Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кваліфікаційна наукова
праця на правах рукопису

ФІЛІМОНЧУК ТЕТЯНА ВОЛОДИМИРІВНА

УДК 004.051: 004.75

ДИСЕРТАЦІЯ

МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПОДІЛУ ЗАВДАНЬ В ГЕТЕРОГЕННИХ GRID-СИСТЕМАХ

05.13.06 – інформаційні технології

технічні науки

Подається на здобуття наукового ступеня кандидата наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Науковий керівник
Волк Максим Олександрович, кандидат технічних наук, доцент

Харків – 2017

АНОТАЦІЯ

Філімончук Т.В. Методи та інформаційна технологія розподілу завдань в гетерогенних GRID-системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 «Інформаційні технології». – Харківський національний університет радіоелектроніки, Міністерство освіти і науки України, Харків, 2017.

У дисертаційній роботі запропоновано рішення актуальної науково-практичної задачі розробки методів та інформаційної технології розподілу завдань в гетерогенних GRID-системах. Об'єктом дослідження в роботі виступає процес розподілу завдань в гетерогенних GRID-системах, предметом дослідження – методи планування та інформаційна технологія розподілу завдань в гетерогенних GRID-системах.

Методи дослідження основані на використанні теорії множин – для розробки моделі розподілу завдань в гетерогенних GRID-системах, моделей завдань та обчислювальних ресурсів; загальній теорії систем – для дослідження та розробки методів розподілу завдань в розподілених обчислювальних системах; теорії імітаційного моделювання – для моделювання процесу розподілу завдань в гетерогенних GRID-системах.

В роботі запропоновано модифіковану математичну модель розподілу завдань на обчислювальних ресурсах в гетерогенній GRID-системі, яка розширена за рахунок введення множини методів розподілу та додаткових параметрів при поданні обчислювальних ресурсів та завдань:

- розширення моделі розподілу завдань в GRID-системі за рахунок множини методів розподілу дозволяє здійснювати сукупність експериментів для різних вхідних пулів завдань із подальшим вибором плану розподілу з мінімальними часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів для їх запуску;

- використання у запропонованій моделі коефіцієнту зв'язності дозволяє здійснювати підбір обчислювальних ресурсів з урахуванням мінімізації часу об-

міну даними між задачами в завданні;

- використання у моделі сумарної затримки часу передачі пакетів даних і пропускної здатності каналу зв'язку дозволяють скоротити час виконання пулу завдань, що підвищує ефективність використання обчислювальних ресурсів в GRID-середовищі.

Запропоновано модифікацію методу Backfill із консервативним резервуванням, яка, на відміну від існуючого, дозволяє виконувати розподіл завдань в залежності від зв'язності задач у кожному із завдань. Запропонований метод оперує двома додатковими параметрами: затримкою часу передачі пакетів даних (δ_n) та пропускною здатністю каналу зв'язку (C_n). Залежно від класу завдання, здійснюється підбір обчислювальних ресурсів за допомогою одного із запропонованих параметрів, завдяки чому зменшується час незадіянності обчислювальних ресурсів за рахунок вивільнення каналів зв'язку комп'ютерної мережі.

Розроблено метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів, який, на відміну від існуючих, використовує узагальнений критерій оцінки завдання та імітаційне середовище моделювання GRASS. Це дозволяє підвищити ефективність використання обчислювальних ресурсів GRID-системи за рахунок вибору плану розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів. Впровадження узагальненого критерію оцінки завдання в роботу планувальника (брокера) дозволяє збільшити ефективність його роботи за рахунок зменшення простою обчислювальних ресурсів GRID-системи.

На підставі запропонованої математичної моделі розподілу завдань в GRID-системі, розширенні кортежів завдань та обчислювальних ресурсів, модернізації методу розподілу завдань Backfill з консервативним резервуванням, розробки методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів розроблено інформаційну технологію розподілу завдань, яка була впроваджена в імітаційне середовище моделювання GRASS. Це дозволяє відтворювати процес функціонування елементарних подій, що відбуваються в реальній GRID-системі із збереженням логіки їх взаємо-

дії у поточному часі. Запропонована інформаційна технологія об’єднує процеси передачі, зберігання, збору даних, спостережень за швидкоплинними процесами та процес їх обробки з використанням запропонованих у роботі методів.

Практична значимість отриманих теоретичних результатів дисертаційної роботи підтверджена поліпшенням продуктивності GRID-системи, за рахунок скорочення простою обчислювальних ресурсів. Зокрема, практичне вирішення теоретичних досліджень полягає у такому.

Запропоновано блок, який відповідає за формування додаткових параметрів для найбільш ефективного розподілу завдань. Він включає 2 модуля: модуль згортки кортежу та модуль аналізу зв’язності. Модуль згортки кортежу здійснює обчислення узагальненого критерію оцінки для кожного завдання. Він дозволяє більш продуктивно управляти процесом розподілу завдань на обчислювальних ресурсах і показує, яку частину обчислювального ресурсу займає завдання в процесі його виконання. Результатом роботи цього модуля є перелік обчислювальних ресурсів, на яких кожне завдання може бути розподілено.

Модуль аналізу зв’язності дозволяє здійснювати підбір обчислювальних ресурсів з урахуванням скорочення часу, що витрачається на обмін між задачами у завданні. В сучасних схемах організації розподілу завдань в GRID-системах не враховується характер пакету завдання (зв’язність задач в завданні). Постачальник завдання може вказувати ступінь зв’язності задач в завданні, що дозволяє скоротити час передачі проміжних результатів між задачами завдання. Це, в свою чергу, скорочує час перебування завдання в GRID-системі та збільшує відсоток задіяності обчислювальних ресурсів. Тому в роботі пропонується введення коєфіцієнту зв’язності, завдяки якому відбувається розподіл завдань на обчислювальних ресурсах з урахуванням зменшення простою обчислювальних ресурсів GRID-системи.

Завдання, що надходять до GRID-системи на виконання, характеризуються наступними параметрами: обсяг вхідних і/або вихідних даних, час завантаження та виконання. В роботі пропонується введення вхідних та вихідних вагових коефіцієнтів для кожного завдання. Вони необхідні для оцінки вхідних даних та ре-

зультатів виконання завдання (вихідних даних). На підставі аналізу вагових коефіцієнтів вводиться розподіл завдань, які надходять до системи, на чотири класи:

- великий обсяг вхідних даних та великий обсяг вихідних даних;
- малий обсяг вхідних даних та великий обсяг вихідних даних;
- великий обсяг вхідних даних та малий обсяг вихідних даних;
- малий обсяг вхідних даних та малий обсяг вихідних даних.

Результати, які наведені в роботі (розділ 4), підтверджують припущення, що передача вихідних даних може займати більше часу, ніж обробка вхідних даних. Отже, за наявності таких завдань в GRID-системі, слід враховувати пропускну здатність каналів зв'язку (гетерогенність комп'ютерної мережі), що дозволяє минати вузькі місця в мережі GRID-системи. Це підвищує ефективність використання таких систем.

В роботі розроблено та впроваджено в імітаційне середовище моделювання GRASS модуль розподілу завдань Algoritm Loader, який побудований на множині методів розподілу. Методи розподілу завдань, реалізовані в середовищі GRASS, дозволяють здійснювати моделювання реальних ситуацій, що протікають в GRID-системі. Оскільки в даний час не існує універсального методу розподілу завдань на обчислювальні ресурси, який би давав «найкращий» план розподілу, то необхідно аналізувати вхідні пули завдань та обчислювальних ресурсів в зв'язці. Дуже часто може виникнути ситуація, коли найпростіші методи (LIFO, FIFO) дозволяють отримати виграш за часом виконання пулу завдань на тих обчислювальних ресурсах, які присутні в системі на даний момент часу. І наявність в системі єдиного планувальника, навіть самого кращого (Backfill) не завжди завантажує ресурси GRID-систем на повну потужність, в зв'язку з чим відсоток простою обчислювальних ресурсів буває досить високий. У таких ситуаціях слід робити акцент на класи завдань і будувати плани розподілу з урахуванням різних методів розподілу, що дозволить підвищити ефективність використання обчислювальних ресурсів GRID-системи.

В роботі додана база даних (БД), яка фіксує інформацію про кожне запущене завдання в системі: час виконання завдання на кожному з обчислювальних ре-

сурсів, дані про відмови, дані про стан обчислювальних ресурсів. Завдяки використанню запропонованої БД планувальник має можливість отримувати результати попередніх розподілів завдань за різними методами розподілу, що дозволяє обрати такий план розподілу, який матиме виграш за рядом критеріїв, заздалегідь встановлених користувачем. У процесі моделювання в імітаційному середовищі GRASS всі дані з log-файлів додаються до БД, яку можна використовувати для аналізу статистичної інформації в міру необхідності.

В ході дослідження розробленої інформаційної технології розподілу завдань, було проведено ряд експериментів. Вони довели те що, застосування одного методу розподілу завдань в планувальнику є менш ефективним. Більш ефективним є метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів. Також отримана залежність результатів розподілу (час виконання пулу завдань) від класу завдань. На сьогодні не існує методу розподілу, який для довільного пулу завдань отримував би оптимальний варіант розподілу, однак якщо відомі характеристики обчислювальних ресурсів GRID-системи, вимоги вхідного потоку завдань, то можна провести моделювання, обрати метод розподілу з мінімальним часом виконання для конкретного пулу завдань, та запропонувати його для використання у реальній GRID-системі.

Використання розроблених у дисертаційній роботі методів дає можливість підвищити ефективність використання обчислювальних ресурсів GRID-системи за рахунок зменшення їх простою. Достовірність отриманих практичних результатів підтверджена експериментальними дослідженнями створеного програмного забезпечення, результатами розподілу пулів завдань для реальних GRID-систем. Напрямки теоретичних і практичних досліджень дисертаційної роботи доцільно розвивати в області створення та використання методів розподілу в гетерогенних GRID-системах.

На підставі проведених досліджень і практичної реалізації представлених методів та інформаційної технології розроблено методичне та програмне забезпечення, що використовується в навчальному процесі Харківського національного

університету радіоелектроніки при вивченні дисциплін профілюючого циклу «Інтерфейси паралельного програмування» та «Паралельне моделювання на HPC системах».

Матеріали дисертації достатньо повно викладені у 25 роботах: з них 7 статей у виданнях, які зазначені в переліку фахових видань України з технічних наук (всі праці входять до науково-метричних баз, 1 – до бази Scopus) та 18 тез доповідей міжнародних конференцій.

Ключові слова: розподілені системи, GRID-система, планувальник завдань, брокер, інформаційна технологія, методи розподілу, імітаційне середовище моделювання GRASS, обчислювальні ресурси, завдання.

Список публікацій здобувача

1. Волк М. А., Филимончук Т. В., Гридель Р. Н. Методы распределения ресурсов для GRID-систем. Збірник наукових праць ХУПС. Харків: ХУПС, 2009. №1(19). С. 100–104.

2. Руденко О. Г., Волк М. А, Филимончук М. А., Филимончук Т. В. Архітектура системи моніторингу трафіку в GRID. Право і безпека. Харків: ХНУВС, 2010. №1(33). С. 226–229.

3. Волк М. А, Филимончук М. А., Филимончук Т. В. Модуль распределения заданий в GRID-системах. Системи обробки інформації. Харків: ХУПС, 2012. №2(100). С. 177–182.

4. Волк М. А., Филимончук Т. В., Ал Шиблак Муаз Анализ современного состояния и развития GRID-технологий и языков описания заданий. Збірник наукових праць ХУПС. Харків: ХУПС, 2013. №2 (35). С. 75–81.

5. Волк М. А., Гридель Р. Н., Филимончук Т. В., Ал Шиблак Муаз Формализация процессов распределенной имитации информационных систем. Системи обробки інформації. Харків: ХУПС, 2013. №4(111). С. 89–93.

6. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and

controlling system, 2016. Vol.3/9 (81). P. 45–53. (цитується в Scopus).

7. Волк М. А., Филимончук Т. В. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33–37.

8. Волк М. А., Филимончук М. А., Корниенко Т. В. (Филимончук Т. В.) Анализ использования искусственных иммунных систем в GRID-инфраструктуре. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції. К.: НТУУ «КПІ», 2008. С. 290.

9. Волк М. А., Филимончук М. А., Филимончук Т. В. Анализ алгоритмов распределения ресурсов в имитационных системах моделирования, ориентированных на GRID системы. Проблеми інформатики і моделювання. Матеріали восьмої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2008. С. 20.

10. Волк М. А, Филимончук М. А., Филимончук Т. В. Исследование методов распределения заданий для GRID-систем. Системный анализ и информационные технологии: Материалы XI Международной научно-технической конференции (26-30 мая 2009 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2009. С. 423.

11. Дьяченко К. Ю., Филимончук Т. В. Подключение модулей распределения задач к имитационной модели GRID-системы. Проблеми інформатики і моделювання. Матеріали дев’ятої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2009. С. 43.

12. Волк М. А, Филимончук Т. В., Гридель Р. Н. Имитационная система моделирования GRID-инфраструктуры GRASS. Системный анализ и информационные технологии: Материалы XII Международной научно-технической конференции (25-29 мая 2010 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2010. С. 359.

13. Фесенко М. В., Филимончук Т. В. Анализ алгоритмов распределения заявок в Grid-системах. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали першої науково-технічної конференції. Х.: ДП «ХНДІ ТМ»; К.: ДП «ЦНДІ НіУ», 2010. С. 76.

14. Филимончук М. А., Филимончук Т. В. Использование средств виртуализации при мониторинге трафика для GRID. Інформаційні технології в навігації і

управлінні: стан та перспективи розвитку. Матеріали першої міжнародної науково-технічної конференції. К.: ДП «ЦНДІ НіУ», 2010. С. 28.

15. Волк М. А, Филимончук Т. В. Анализ существующего прикладного программного обеспечения GRID-систем и языков описания заданий. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Матеріали третьої міжнародної науково-технічної конференції. Полтава: ПНТУ; Бєлгород: НДУ «БілДУ»; Харків: ДП «ХНДІ ТМ», Київ: НТУ; Кіровоград: КЛА НАУ, 2013. С. 49.

16. Волк М. А, Филимончук Т. В. Обобщенный критерий оценки задания для технологии планирования заданий в GRID. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам третьей международной научно-практической конференции (24-26 апреля 2013 г., Смоленск). В 3-х томах. Том 2. Смоленск: Смоленский филиал Российского университета кооперации, 2013. С. 172–176.

17. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий по ресурсам. Проблеми інформатизації: Матеріали першої міжнародної науково-технічної конференції. Черкаси: ЧДТУ; Київ: ДУТ; Тольятті: ТДУ; Полтава: ПНТУ, 2013. С. 20–21.

18. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в распределенных системах. Проблеми інформатизації: Матеріали другої міжнародної науково-технічної конференції. Київ: ДУТ; Полтава: ПНТУ; Катовице: Катовицький економічний університет; Париж: Університет Париж VII Вєнсент-Сен-Дєні; Білгород: НДУ «БДУ»; Черкаси: ЧДТУ; Харків: ХНДІТМ, 2014. С. 59–60.

19. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам четвертой международной научно-практической конференции (23-25 апреля 2014 г., Смоленск). В 2-х томах. Том 1. Смоленск: Смоленский филиал Российского университета кооперации, 2014. С. 379–383.

20. Филимончук Т. В., Ткачев В. Н. Информационная технология распределения заданий на вычислительные ресурсы в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам пятой международной научно-практической конференции (11-15 мая 2015 г., Смоленск). В 2-х томах. Том 1 Смоленск: Смоленский филиал Российского университета кооперации, 2015. С. 204–209.

21. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий на вычислительные ресурсы для обработки радиоастрономических данных в GRID-системах. Проблеми інформатизації: Матеріали третьої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2015. С. 23.

22. Волк М. А., Филимончук Т. В. Использование системы моделирования GRASS в задачах распределения заданий в GRID-системах. 20-й Ювілейний міжнародний форум «Радіоелектроніка та молодь у XXI столітті». Харків: ХНУРЕ, 2016. Т.4. С. 171–172.

23. Volk M. O., Filimonchuk T. V. Information technology for job distribution in GRID-systems. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: матеріали шостої міжнародної науково-технічної конференції. Полтава: ПНТУ; Баку: ВА ЗС АР; Кіровоград: КЛА НАУ; Харків: ДП «ХНДІ ТМ», 2016. С. 45–46.

24. Ткачев В. Н., Филимончук Т. В., Митин Д. Е. Использование информационной технологии распределения заданий при обработке больших массивов данных в виртуальных частных облаках. Информационные системы и технологии: Материалы 5-й международной научно-технической конференции (2-17 сентября 2016 г., Харьков). Х.: ДРУКАРНЯ МАДРИД, 2016. С. 333–334.

25. Волк М. А., Филимончук Т. В. Модифицированный метод Backfill с консервативным резервированием. Проблеми інформатизації: Матеріали четвертої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2016. С. 20.

ABSTRACT

Filimonchuk T.V. Methods and information technology for task distribution in heterogeneous GRID-systems. – Qualifying scientific work on the manuscript rights.

A thesis for the candidate degree in technical sciences in the specialty 05.13.06 – information technology. – Kharkov National University of Radio Electronics, Ministry of Education and Science of Ukraine, Kharkov, 2017.

The thesis proposed solutions relevant scientific and practical task of developing methods and information technology division of tasks in heterogeneous GRID-systems. The object of research in the process of serving the tasks distribution in heterogeneous GRID-systems, the subject of research – planning methods and information technology tasks division in heterogeneous GRID-systems.

Methods based on the use of set theory – to develop a model distribution tasks in heterogeneous GRID-systems, models and computing resources tasks; general theory of systems – for research and development of methods for the distribution of tasks in distributed computing systems; theory of simulation – the simulation process for the tasks distribution in heterogeneous GRID-systems.

The paper presents a mathematical model of a modified distribution of tasks to resources in a heterogeneous computing GRID-system, which expanded by introducing a set of methods of distribution and advanced options when submitting computing resources and tasks:

- expansion model distribution tasks in GRID-system through a plurality of distribution methods allows a set of experiments for different input tasks pools selecting distribution plan with minimal runtime pool assignments and minimal idle computing resources to run them;

- used in the proposed model coefficient connectivity allows the selection of computing resources based on minimizing time data exchange between tasks in the job;

- using a model of total delay time packet data bandwidth can reduce the performance of a pool of tasks, which increases the efficiency of computing resources in GRID-environment.

A modification of the method Backfill with conservative reserving that, unlike the existing one, allows performing division of tasks depending on the connectivity problems in each of the tasks. The proposed method operates two additional parameters: time data packet delay (δ_n) and communications bandwidth (C_n). Depending on the class of tasks carried out the selection of computing resources using one of the proposed options, reducing downtime computing resources by freeing up channels of computer network.

Developed the finding method the distribution of tasks with minimal performance objectives of the pool and minimal idle computing resources, which, unlike existing uses generalized criteria assessment tasks and simulation environment simulation GRASS. This allows more efficient use of computing resources of GRID-system by building a distribution plan with minimal performance objectives of the pool and minimal idle computing resources. Implementation of generalized criteria assessment tasks in the job scheduler (broker) can increase its efficiency by reducing idle computing resources of GRID-system.

Based on the proposed mathematical model distribution tasks in GRID-system, expanding tasks tuples and computing resources, modernization of the distribution method with conservative objectives Backfill reservation, developing a method of finding the distribution of tasks with minimal performance objectives of the pool and minimal idle computing resources developed information technology division of tasks, which was implemented in the simulation environment simulation GRASS. This process allows you to play the functioning of elementary events occurring in real GRID-system logic while preserving their interaction at the current time. Information technology integrates the processes of transmission, storage, data collection, fleeting observations of the process and their treatment using the proposed method in the thesis.

The practical significance of the thesis theoretical results confirmed improved performance GRID-system by reducing idle computing resources. In particular, the practical solution is theoretical research in this.

Proposed a unit responsible for the formation of additional options for the most effective distribution of tasks. It includes two modules: tuples convolution module and

connectivity analysis module. The tuples convolution module performs convolution calculation convoy generalized evaluation criteria for each job, which can more efficiently manage the distribution of tasks to computing resources and shows how much of computing resources is the task during its execution. The work result of this module is list for computing resources, where each task can be divided.

Connectivity analysis module allows the selection of computing resources based on reducing time spent on the exchange between tasks in the job. In the current distribution scheme of tasks in GRID-systems package task character is ignored (connectivity problems in the job). Supplier of tasks may indicate a degree of connectivity problems in the job, thereby reducing the transmission of intermediate results between the objectives of the task. This in turn reduces the time spent tasks in GRID-system and increase the employment rate of computing resources. Therefore, the paper proposes the introduction connectivity coefficients by which the distribution of tasks to computing resources considering reducing idle computing resources of GRID-system.

Tasks that comes in GRID-system for execution, characterized by the following parameters: the volume of incoming and/or outgoing data downloading and execution. The paper proposes the introduction of input and output weighting coefficients for each task. They are required to assess the data and results of the assignment (baseline data). On the basis of weight coefficients introduced division of tasks that come into the system, into four classes:

- large amount of input and output data;
- small amount of input data and a large amount of output data;
- large amount of input data and a small amount of output data;
- small amount of input and output data.

The results are given in (section 4) confirm the assumption that the transfer source data may take longer than processing the input data. Consequently, if such tasks exist in GRID-system, should take into account the capacity of communication channels (heterogeneous computer network), which allows pass bottlenecks in the network GRID-system. This increases the efficiency of such systems.

The work developed and implemented in the simulation environment simulation module GRASS distribution of tasks Algoritm Loader, which is built on a set of allocation methods. Methods of distribution of tasks implemented in the GRASS environment, allow the simulation of real situations that occur in GRID-system. Since currently there is no universal method of distribution tasks on computing resources that would give "best" plan of distribution, it is necessary to analyze incoming tasks and pools of computing resources in conjunction. It can often happen that the simplest method (LIFO, FIFO) allow you to gain time perform tasks on a pool of computing resources that are present in the system at this time. And the presence of a single scheduler, even the best (Backfill), do not always load the resources of GRID-full capacity, and therefore the percentage of idle computing resources is high enough. In such situations, it is necessary to focus on class assignments and plan of distribution according to different methods of distribution that will allow more efficient use of computing resources of GRID-system.

In the paper the database (DB) is added, which records information about each task running in the system: performing tasks on each of the computing resources, data rejection, data on computing resources. Using the proposed database scheduler is able to obtain the results of previous allocation of tasks for different methods of distribution that allows you to choose a distribution plan that will benefit in a number of criteria predetermined by the user. In the process of simulation modeling in GRASS among all data from log-files are added to the database that can be used to analyze the statistical information needed.

In the study of developed information technology division of tasks conducted a series of experiments. They proved that the use of one method of distribution tasks scheduler is less effective. More effective is the method of finding the distribution of tasks with minimal performance objectives of the pool and minimal idle computing resources. Also, the results obtained dependence distribution (performing tasks pool) from the class of problems. At present there is no method of distribution, which for any pool of tasks would receive the best option distribution, but if we know the characteristics of computing resources of GRID-system requirements for the incoming

flow of tasks you can carry out simulation, choose the method of distribution with minimal performance for a particular pool of tasks and offer it for use in real GRID-system.

Using developed in the dissertation methods and tools enables more efficient use of computing resources of GRID-systems by reducing their downtime. Reliability of the practical results of experimental studies confirmed established software division pools the results of real problems for GRID-systems. Directions of theoretical and practical research thesis are advisable to develop in the development and use of methods of distribution in heterogeneous GRID-systems.

Based on research and practical implementation of the presented methods and information technology developed methodology and software used in the learning process Kharkiv National University of Radioelectronics in the study subjects cycle profiling «Concurrent programming interfaces», «Parallel simulation on HPC systems».

The thesis materials sufficiently voiced 25 works, including 7 articles in publications that are listed in the list of professional publications of Ukraine in engineering sciences (all articles are members of the scientific-metric database, 1 – in the database Scopus) and 18 abstracts international conferences.

Keywords: distributed systems, GRID-system scheduler, broker, information technology, distribution methods, simulation modeling GRASS environment, computing resources, tasks.

List of publications of the applicant

1. Волк М. А., Филимончук Т. В., Гридель Р. Н. Методы распределения ресурсов для GRID-систем. Збірник наукових праць ХУПС. Харків: ХУПС, 2009. №1(19). С. 100–104.

2. Руденко О. Г., Волк М. А, Филимончук М. А., Филимончук Т. В. Архітектура системи моніторингу трафіку в GRID. Право і безпека. Харків: ХНУВС, 2010. №1(33). С. 226–229.

3. Волк М. А, Филимончук М. А., Филимончук Т. В. Модуль распределения заданий в GRID-системах. Системи обробки інформації. Харків: ХУПС,

2012. №2(100). С. 177–182.

4. Волк М. А., Филимончук Т. В., Ал Шиблак Муаз Анализ современного состояния и развития GRID-технологий и языков описания заданий. Збірник наукових праць ХУПС. Харків: ХУПС, 2013. №2 (35). С. 75–81.

5. Волк М. А., Гридель Р. Н., Филимончук Т. В., Ал Шиблак Муаз Формализация процессов распределенной имитации информационных систем. Системи обробки інформації. Харків: ХУПС, 2013. №4(111). С. 89–93.

6. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol.3/9 (81). P. 45–53. (цитується в Scopus).

7. Волк М. А., Филимончук Т. В. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33–37.

8. Волк М. А., Филимончук М. А., Корниенко Т. В. (Филимончук Т. В.) Анализ использования искусственных иммунных систем в GRID-инфраструктуре. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції. К.: НТУУ «КПІ», 2008. С. 290.

9. Волк М. А., Филимончук М. А., Филимончук Т. В. Анализ алгоритмов распределения ресурсов в имитационных системах моделирования, ориентированных на GRID системы. Проблеми інформатики і моделювання. Матеріали восьмої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2008. С. 20.

10. Волк М. А, Филимончук М. А., Филимончук Т. В. Исследование методов распределения заданий для GRID-систем. Системный анализ и информационные технологии: Материалы XI Международной научно-технической конференции (26-30 мая 2009 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2009. С. 423.

11. Дьяченко К. Ю., Филимончук Т. В. Подключение модулей распределения задач к имитационной модели GRID-системы. Проблеми інформатики і моделювання. Матеріали дев’ятої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2009. С. 43.

12. Волк М. А, Филимончук Т. В., Гридель Р. Н. Имитационная система моделирования GRID-инфраструктуры GRASS. Системный анализ и информационные технологии: Материалы XII Международной научно-технической конференции (25-29 мая 2010 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2010. С. 359.

13. Фесенко М. В., Филимончук Т. В. Анализ алгоритмов распределения заявок в Grid-системах. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали першої науково-технічної конференції. Х.: ДП «ХНДІ ТМ»; К.: ДП «ЦНДІ НіУ», 2010. С. 76.

14. Филимончук М. А., Филимончук Т. В. Использование средств виртуализации при мониторинге трафика для GRID. Інформаційні технології в навігації і управлінні: стан та перспективи розвитку. Матеріали першої міжнародної науково-технічної конференції. К.: ДП «ЦНДІ НіУ», 2010. С. 28.

15. Волк М. А, Филимончук Т. В. Анализ существующего прикладного программного обеспечения GRID-систем и языков описания заданий. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Матеріали третьої міжнародної науково-технічної конференції. Полтава: ПНТУ; Бєлгород: НДУ «БілДУ»; Харків: ДП «ХНДІ ТМ», Київ: НТУ; Кіровоград: КЛА НАУ, 2013. С. 49.

16. Волк М. А, Филимончук Т. В. Обобщенный критерий оценки задания для технологии планирования заданий в GRID. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам третьей международной научно-практической конференции (24-26 апреля 2013 г., Смоленск). В 3-х томах. Том 2. Смоленск: Смоленский филиал Российского университета кооперации, 2013. С. 172–176.

17. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий по ресурсам. Проблеми інформатизації: Матеріали першої міжнародної науково-технічної конференції. Черкаси: ЧДТУ; Київ: ДУТ; Тольятті: ТДУ; Полтава: ПНТУ, 2013. С. 20–21.

18. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в распределенных системах. Проблеми інформатизації: Матеріали дру-

гої міжнародної науково-технічної конференції. Київ: ДУТ; Полтава: ПНТУ; Катовице: Катовицький економічний університет; Париж: Університет Париж VII Вєнсент-Сен-Дєні; Білгород: НДУ «БДУ»; Черкаси: ЧДТУ; Харків: ХНДІТМ, 2014. С. 59–60.

19. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам четвертой международной научно-практической конференции (23-25 апреля 2014 г., Смоленск). В 2-х томах. Том 1. Смоленск: Смоленский филиал Российского университета кооперации, 2014. С. 379–383.

20. Филимончук Т. В., Ткачев В. Н. Информационная технология распределения заданий на вычислительные ресурсы в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам пятой международной научно-практической конференции (11-15 мая 2015 г., Смоленск). В 2-х томах. Том 1 Смоленск: Смоленский филиал Российского университета кооперации, 2015. С. 204–209.

21. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий на вычислительные ресурсы для обработки радиоастрономических данных в GRID-системах. Проблеми інформатизації: Матеріали третьої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2015. С. 23.

22. Волк М. А., Филимончук Т. В. Использование системы моделирования GRASS в задачах распределения заданий в GRID-системах. 20-й Ювілейний міжнародний форум «Радіоелектроніка та молодь у XXI столітті». Харків: ХНУРЕ, 2016. Т.4. С. 171–172.

23. Volk M. O., Filimonchuk T. V. Information technology for job distribution in GRID-systems. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: матеріали шостої міжнародної науково-технічної конференції. Полтава: ПНТУ; Баку: ВА ЗС АР; Кіровоград: КЛА НАУ; Харків: ДП «ХНДІ ТМ», 2016. С. 45–46.

24. Ткачев В. Н., Филимончук Т. В., Митин Д. Е. Использование информационной технологии распределения заданий при обработке больших массивов данных в виртуальных частных облаках. Информационные системы и технологии: Материалы 5-й международной научно-технической конференции (2-17 сентября 2016 г., Харьков). Х.: ДРУКАРНЯ МАДРИД, 2016. С. 333–334.

25. Волк М. А., Филимончук Т. В. Модифицированный метод Backfill с консервативным резервированием. Проблеми інформатизації: Матеріали четвертої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2016. С. 20.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІТ – інформаційна технологія

GRID – технологія, яка заснована на об’єднанні декількох комп’ютерів для вирішення єдиної обчислювально-складної задачі, яка розбита на підзадачі. GRID застосовується для вирішення наукових та математичних задач, що вимагають значних обчислювальних ресурсів

GRASS (GRid Advanced Simulation System) – імітаційне середовище моделювання

БД – база даних

ПЗ – програмне забезпечення

ППЗ – прикладне програмне забезпечення

СПЗ – спеціалізоване програмне забезпечення

ПЗПР – програмне забезпечення проміжного рівня

РОС – розподілена обчислена система

ОС – операційна система

PBS – Portable Batch System

SGE – Sun Grid Engine

LSF – Load Sharing Facility

FCFS – First-Come First-Served

HPF – Highest Priority First

TORQUE – Terascale Open-Source Resource and QUEue Manager

SLURM – Simple Linux Utility for Resource Management

RSL – Resource Specification Language

xRSL – eXtended Resource Specification Language

ClassAd – Classified Advertisement

JDL – Job Description Language

JSDL – Job Submission Description Language

JSON – JavaScript Object Notation

YAML – Yet Another Markup Language

XML – eXtensible Markup Language

MIB – Management Information Base

SNMP – Simple Network Management Protocol

WMI – Windows Management Instrumentation

WMS – Workflow Management System

MCS – Maui Cluster Scheduler

MWM – Moab Workload Manager

ВСТУП

На цей час важко відшукати сферу виробництва, в якій би не застосовувалися інформаційні технології, за допомогою яких здійснюється діяльність окремих компаній. Вони дозволяють автоматизувати та вдосконалити ряд виробничих процесів, що дозволяє полегшити працю, які пов’язані з виконанням небезпечних для життя дій.

Інформаційна технологія (ІТ) – це практична діяльність і прикладна наука, що має справу з даними та інформацією. Прикладом застосування ІТ є збір, обробка, забезпечення безпеки, передача, взаємообмін, уявлення, управління, організація, зберігання та відновлення даних і інформації [1,2].

ІТ застосовуються в освіті, допомагаючи формуванню незалежної інформаційної особистості, підштовхують до прийняття правильних рішень і ефективному використанню інформаційних ресурсів. В.Г. Кремень зазначає, що «XXI століття не тільки висуває нові вимоги до людини, а отже, і до освіти, але і створює нові, раніше небачені можливості для освітньої діяльності. Перш за все, це пов’язано з сучасними інформаційними технологіями, комп’ютерною технікою, яка істотно розширює пізнавальні можливості людини» [3].

Засобом обробки інформаційних потоків виступають комп’ютери, які використовуються у всіх сферах людської діяльності та забезпечують ефективну роботу фахівців. За допомогою комп’ютерів відбувається пошук необхідної інформації, її обробка та аналіз, спрямований на прийняття управлінських рішень, спілкування з партнерами та ін. Великий вплив інформаційні технології зробили на науку, культуру, освіту і навіть суспільно-політичне життя, що призвело до чергового витка розвитку людської цивілізації. Як показує практика, індустріальний етап розвитку людства повільно, але вірно поступається місцем інформаційного етапу [4]. На сьогодні існує безліч ІТ, які допомагають людям у повсякденному житті. Розглянемо GRID-технології, які спрямовані на розподіл завдань по обчислювальних ресурсах.

GRID-система – це узгоджена, відкрита й стандартизована система, яка за-

безпечує гнучкий, безпечний, скоординований розподіл ресурсів в рамках віртуальної організації [5]. З розвитком GRID-технологій з'явилась можливість вирішувати обчислювальні завдання великого обсягу для різних областей, таких як наука, бізнес, техніка, медицина та ін.

Будь-яка GRID-система є неоднорідною (гетерогенної), тому що будується на множені розподілених в просторі обчислювальних ресурсах. Завдання, що надходять на виконання в дану систему, також мають властивість неоднорідності, що ускладнює задачу розподілу. Тому важливе місце в GRID-системах при розподілах відводиться технологіям планування завдань, в обов'язки яких ставиться створення розкладу використання обчислювальних ресурсів. Багато досліджень направлено на створення методу розподілу завдань по обчислювальних ресурсам, який дозволив би мінімізувати час простою ресурсів, скоротити обсяги та час переданої інформації між ними.

Значну роль в дослідженнях цієї області грають роботи вітчизняних та зарубіжних авторів: А.М. Бершадського [6], І.В. Бичкова [7,8], І.А. Голубєва [9], Ж.Б. Кальпеевой [10], Н.В. Покусіна [11], Є.Ю. Селіверстова [12], Б.А. Телесніна [13], D. Agrawal [14], A. Amar, R. Bolze, E .Boix [15], B. Amedro, V. Bodnartchouk [16], L. Baduel, H. Zhao, Sakellariou R [17].

Актуальність теми. Сьогодні широкого поширення набули ресурсомісткі завдання, які для свого вирішення вимагають великої кількості обчислювальних ресурсів. Для таких задач використовуються територіально розподілені обчислювальні системи. Одним із прикладів таких систем є GRID. В основі GRID-систем лежить забезпечення стабільної роботи певного набору служб, побудованих на відкритих стандартах, і проміжного програмного забезпечення [18].

Розподілом обчислювальних ресурсів займається планувальник (брокер), який виступає посередником між постачальниками завдань і обчислювальних ресурсів, він реалізується за допомогою певного методу розподілу. Нині існує низка систем управління й розподілу ресурсів, проте планувальники, які реалізовані в них, не дають ефективного способу розподілу завдань, з їх допомогою користувач може лише скористатися одним простим алгоритмом. Завдання, які надходять на

вхід планувальника, є різнорідними, що створює додаткові складнощі під час їх розподілу. Розробка методу і технології розподілу завдань (з урахуванням ряду додаткових параметрів) на обчислювальні ресурси є актуальною проблемою, оскільки вибір обчислювальних ресурсів визначає ефективність використання всієї системи. Правильний вибір впливає на час простою обчислювальних ресурсів, скорочує час та обсяги переданої між пристроями інформації тощо.

Мета та задачі дослідження. Метою дисертаційної роботи є розробка методів та інформаційної технології розподілу завдань в гетерогенних GRID-системах, які дозволяють зменшити час простою обчислювальних ресурсів системи за рахунок скорочення часу виконання вхідного пулу завдань.

Згідно зі сформульованою метою в дисертаційній роботі необхідно вирішити такі задачі:

- аналіз існуючих методів розподілу завдань в розподілених обчислювальних системах;

- модифікація математичної моделі розподілу завдань в гетерогенних GRID-системах;

- модифікація методу розподілу завдань (Backfill) на обчислювальні ресурси;

- розробка методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів;

- розробка інформаційної технології розподілу завдань на обчислювальні ресурси на основі запропонованої математичної моделі розподілу завдань;

- проведення ряду експериментів з розподілу завдань на основі запропонованої інформаційної технології в системі імітаційного моделювання;

- апробація моделі та методів при вирішенні практичних завдань.

Об’єкт дослідження – процес розподілу завдань в гетерогенних GRID-системах.

Предмет дослідження – методи планування та інформаційна технологія розподілу завдань в гетерогенних GRID-системах.

Методи дослідження базуються на використанні: теорії множин (для роз-

робки моделі розподілу завдань в гетерогенних GRID-системах, моделей завдань та обчислювальних ресурсів), загальної теорії систем (для дослідження та розробки методів розподілу завдань в розподілених обчислювальних системах), теорії імітаційного моделювання (для моделювання процесу розподілу завдань в гетерогенних GRID-системах).

Наукова новизна отриманих результатів. Основні результати, які визначають наукову новизну дисертаційної роботи, полягають у такому:

- вперше запропоновано метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів, який, на відміну від існуючих, використовує узагальнений критерій оцінки завдання та імітаційне середовище моделювання GRASS, що дозволяє підвищити ефективність використання обчислювальних ресурсів GRID-системи за рахунок скорочення часу виконання пулу завдань та простою обчислювальних ресурсів;

- удосконалено метод розподілу завдань на обчислювальні ресурси Backfill з урахуванням трафіку всередині завдання, який, на відміну від існуючих методів, враховує інтенсивність та обсяг потоків даних між задачами у завданні, що дозволяє підвищити ефективність використання GRID-системи за рахунок зменшення часу виконання пулу завдань на розподілених гетерогенних обчислювальних ресурсах;

- набули подальшого розвитку моделі подання завдань і ресурсів у GRID-системі, які, на відміну від існуючих, враховують обсяг мережного трафіку та затримку передачі даних в процесі розподілу і виконання завдань, що дозволяє скоротити час виконання завдання в системі за рахунок усунення втрат за часом, які викликані обміном даних між окремими задачами у завданні.

Особистий внесок здобувача. Всі наукові результати дисертаційної роботи, що виносяться на захист, отримані автором самостійно. У роботах, опублікованих спільно, автору належать такі результати: у роботі [19] розроблена класифікація з точки зору областей застосування методів для різних класів завдань та конфігурацій обчислювальних ресурсів; у роботі [20] наведена архітектура системи моніторингу трафіку в комп’ютерній мережі, яка забезпечує ефективний роз-

поділ завдань за обчислювальними ресурсами GRID-системи; у роботі [21] наведені моделі подання завдань та обчислювальних ресурсів у GRID-системі, запропоновано модуль розподілу завдань на обчислювальні ресурси, який підтримує можливість проведення серії експериментів; у роботі [22] на основі аналізу сучасного стану GRID-технологій, запропоновано класифікацію сучасного прикладного програмного забезпечення, яке використовується у GRID-системах для розподілу завдань; у роботі [23] запропоновано формальний опис процесів у розподілених системах імітаційного моделювання, розглянуто процеси управління локальними та глобальними даними моделей; у роботі [24] запропоновано інформаційну технологію розподілу завдань для GRID-систем, яка ґрунтується на використанні імітаційного середовища моделювання GRASS, а також метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів; у роботі [25] запропоновано удосконалений метод розподілу завдань Backfill з урахуванням трафіку всередині завдання; у роботі [26] проведено аналіз сучасного стану штучних імунних систем і можливостей їх використання в розподілених обчислювальних системах типу GRID; у роботі [27] досліджені методи розподілу ресурсів у гомогенних GRID-структурах та надана їх класифікація; в роботі [28] проведено аналіз найбільш поширених методів планування, які дозволяють оптимізувати час виконання послідовності завдань; у роботі [29] розроблено механізм підключення модулів розподілу заявок до імітаційної системі моделювання GRID на основі плагінів; у роботі [30] розглянуто структуру імітаційного середовища GRASS, проаналізовано склад та функціональне призначення модулів системи; у роботі [31] запропоновано програмні реалізації декількох методів розподілу завдань на обчислювальні ресурси, наведені результати аналізу ефективності даних методів; у роботі [32] обґрунтовано необхідність розробки та вдосконалення методів і засобів віртуалізації при моніторингу трафіку комп’ютерних мереж; у роботі [33] наведені результати аналізу сучасного стану і перспективи розвитку GRID-систем, запропоновано розмежування існуючого прикладного програмного забезпечення за критеріями, які класифікують GRID-систему залежно від поставленої перед користувачем мети; у роботі [34] наведено

аналіз доцільності введення узагальненого критерію для оцінки завдань, який дозволяє підібрати ресурси обчислювальної системи з урахуванням вимог постачальників завдань та власників ресурсів; у роботі [35] запропоновано підхід, який ґрунтується на використанні інформаційної технології розподілення завдань, що дозволяє використовувати сукупність заданих методів розподілу для отримання інформації про розподіл потоку завдань; у роботі [36] запропоновано інформаційну технологію розподілу завдань на обчислювальні ресурси GRID-системи, обґрунтовано використання модуля, який формує додаткові параметри для розподілу завдань; у роботі [37] обґрунтовано доцільність використання існуючих технологій з точки зору підвищення продуктивності системи, описано функціонування технології розподілу завдань на обчислювальні ресурси; у роботі [38] запропоновано інформаційну технологію розподілу завдань на обчислювальні ресурси, яка дозволяє отримати план розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів GRID-системи; у роботі [39] розглянута удосконалена інформаційна технологія розподілу завдань на обчислювальні ресурси в GRID-системі, яка дозволяє здійснювати вибір плану розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів, а також реалізує можливість розширення множини методів розподілу за рахунок додавання нових програмних модулів; у роботі [40] запропоновано алгоритм функціонування імітаційної середовища моделювання GRASS, яке відтворює всі процеси, що відбуваються у реальній GRID-системі; у роботі [41] запропонована інформаційна технологія розподілу завдань на обчислювальні ресурси з великим обсягом вхідних та вихідних даних для GRID-систем, яка заснована на ряді існуючих технологій, що працюють у комплексі і дозволяють знизити трудомісткість процесів використання обчислювальних ресурсів; у роботі [42] запропоновано використання інформаційної технології розподілу завдань для обробки великих масивів вхідних даних у віртуальних приватних хмарах; у роботі [43] запропонована реалізація методу Backfill (mod), який, на відміну від існуючого (Backfill), враховує інтенсивність і обсяг потоків даних між задачами у завданні, що дозволяє підвищити ефективність використання GRID-системи за рахунок

зменшення часу виконання пулу завдань на розподілених гетерогенних обчислювальних ресурсах.

Статті [19,23] та тези доповідей [30] опубліковані спільно із Гріделем Ростиславом Миколайовичем, особистий внесок Гріделя Р.М. наведено в його дисертації на здобуття наукового ступеня кандидата технічних наук [44].

Апробація результатів дисертації. Основні положення дисертаційної роботи доповідалися на таких міжнародних конференціях і форумах:

- 8-й, 9-й Міжнародній науково-технічній конференції «Проблеми інформатики і моделювання» (Харків: 2008, 2009 рр.);

- 10-й, 11-й, 12-й Міжнародній науково-технічній конференції «Систем-ний аналіз та інформаційні технології» (Київ: 2008, 2009, 2010 рр.);

- 1-й Міжнародній науково-технічній конференції «Інформаційні технології в навігації і управлінні: стан та перспективи розвитку» (Київ: 2010 р.);

- 1-й, 3-й, 5-й Міжнародній науково-технічній конференції «Сучасні напрямки розвитку інформаційно-комунікаційних технологій і способів управління» (Київ – Харків 2010 р., Полтава – Бєлгород – Харків – Київ – Кіровоград 2013 р., Полтава – Баку – Кіровоград – Харків 2016 р.);

- 3-й, 4-й, 5-й Міжнародній науково-практичній конференції «Информатика, математическое моделирование, экономика» (Смоленськ: 2013, 2014, 2015 рр.);

- 1-й, 2-й, 3-й, 4-й Міжнародній науково-технічній конференції «Проблеми інформатизації» (Черкаси – Київ – Тольятті – Полтава 2013 р., Київ – Полтава – Катовіце – Париж – Бєлгород – Черкаси – Харків 2014 р., Черкаси – Баку – Бельсько-Бяла – Полтава 2015 р., Черкаси – Баку – Бельсько-Бяла – Полтава 2016 р.);

- 20-й Международный молодёжный форум «Радиоэлектроника и молодежь в XXI веке» (Харків: 2016 р.);

- 5-й Міжнародній науково-технічній конференції «Інформаційні системи та технології» (Коблево: 2016 р.).

На підставі проведених досліджень і практичної реалізації представлених методів та інформаційної технології розроблено методичне та програмне забезпечення, що використовується в навчальному процесі Харківського національного

університету при вивченні дисциплін профілюючого циклу «Інтерфейси паралельного програмування» та «Паралельне моделювання на HPC системах».

Структура дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів основної частини, висновків, списку використаних джерел, додатків.

У ***першому розділі*** на базі вивчення літературних джерел проведено аналіз предметної області. Розглянуто питання розвитку розподілених систем як середовища для розв'язання задач великої обчислювальної складності. Наведено огляд технологій розподілених обчислень та Cloud-технологій, існуючих планувальників завдань, мов опису завдань в GRID-системах, а також наведена постановка задачі на дослідження.

У ***другому розділі*** досліджена існуюча моделі розподілу завдань в GRID-системі, підходи вирішення для розподілу завдань, технології розподілу завдань на обчислювальні ресурси. У ході дослідження було виявлено що, в процесі розподілу завдань на обчислювальні ресурси не враховується параметри, які можуть підвищити ефективність використання ресурсів GRID-системи (пропускна здатність та затримка часу передачі пакету по каналу), а також характер пакета завдання (зв'язність задач у завданні), що дозволить зменшити час перебування завдання в системі. Існуючий метод розподілу Backfill не враховує час, витрачений на передачу вхідних і вихідних даних завдання, і під час його використання немає важелів впливу на хід розподілу конкретного завдання, тому що розподіл здійснюється за принципом «перший відповідний».

Таким чином в роботі ставиться науково-технічна задача розробки нового підходу до розподілу завдань в гетерогенних GRID-системах. Цей підхід орієнтований на використання нової інформаційної технології розподілу завдань та методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів.

У ***третьому розділі*** наведена модифікована математична модель розподілу завдань в GRID-системі, яка розширена за рахунок введення множини методів розподілу і додаткових параметрів в поданні обчислювальних ресурсів та завдань. Підбір обчислювальних ресурсів за рядом параметрів накладає труднощі при роз-

поділі, тому у роботі пропонується згортка параметрів у єдиний критерій оцінки завдання. Цей параметр допомагає відібрати множину обчислювальних ресурсів, на яких може бути запущено завдання, та показує, яку частину ресурсу займає завдання в процесі виконання.

На даний час в системах розподілу завдань на обчислювальні ресурси поширено метод зворотного заповнення Backfill, який запобігає дефрагментацію обчислювальних ресурсів, однак жодним чином не враховує час, який витрачається на передачу вхідних та вихідних даних завдання (даний час безпосередньо залежить від пропускної здатності каналу зв'язку та часу затримки). Модифікований метод (Backfill_mod), який наведено в роботі, дозволяє усунути недоліки, які були виявлені при аналізі, та підвищити ефективність використання обчислювальних ресурсів GRID-системи.

У роботі подано метод пошуку розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів. За допомогою даного методу можливо обрати план розподілу, який дозволяє мінімізувати ряд критеріїв: час виконання пулу завдань, час простою обчислювальних ресурсів і час перебуванням заявок у черзі.

На основі модифікованої математичної моделі розподілу завдань в GRID-системі, модернізації методу Backfill з консервативним резервуванням, розробці методу пошуку найкращого розподілу для пулу завдань була запропонована інформаційна технологія розподілу завдань, яка була впроваджена в середовище моделювання GRASS.

У ***четвертому розділі*** наведена практична реалізація запропонованої інформаційної технології та обґрунтовані переваги її впровадження.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалася відповідно до плану науково-технічних робіт Харківського національного університету радіоелектроніки в рамках держбюджетних тем:

- «Розробка структури Харківського ресурсно-операційного GRID-центру та його ресурсів», договір №9 (28.09.07-31.10.07, 01.12.07-31.12.07) між ХНУРЕ і «ІПСА» НТУУ «КПІ», що виконувалася на підставі договору «ІПСА» НТУУ

«КПІ» з Міністерством освіти і науки України №IT/506-2007, Державної програми «Інформаційні та телекомунікаційні технології в освіті і науці» на 2006-2010 рр. (№ДР 0107U010616);

- «Розробка та дослідження застосування GRID-порталу Харківського ресурсно-операційного GRID-центру», договору №08-22 (08.04.07-27.06.08) і №08-22/9 (01.07.08-30.09.08) між ХНУРЕ і «ІПСА» НТУУ «КПІ», що виконувалася на підставі Договору «ІПСА» НТУУ «КПІ» з Міністерством освіти і науки України №IT/506-2013, Державної програми «Інформаційні та телекомунікаційні технології в освіті і науці» на 2006-2013 рр. (№ДР 0108U008261).

У рамках перерахованих тем здобувачем проведено аналіз концепцій, архітектури, конфігурацій обчислювальних ресурсів, програмного забезпечення сучасних національних проектів GRID, розглянуті цілі та завдання групи NA4. Здійснено підключення до суперкомп'ютерного центру «КПІ» та тестування потужності бібліотек інтерфейсу паралельного програмування PVM на прикладі декількох задач. Результати наукових досліджень використані в науково-технічних звітах про НДР №9, НДР №08-22/9, НДР №08-22.

Практичне значення отриманих результатів. Практична значимість отриманих теоретичних результатів дисертаційної роботи підтверджено поліпшенням продуктивності GRID-системи, за рахунок скорочення простою обчислювальних ресурсів. Зокрема, практичне вирішення теоретичних досліджень полягає у наступному.

Запропоновано інформаційну технологію розподілу завдань на обчислювальні ресурси в GRID-системі, яка на підставі використання множини методів розподілу дозволяє здійснювати вибір розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів.

Запропоновано блок, який відповідає за формування додаткових параметрів для ефективного розподілу завдань. Він включає 2 модуля: модуль згортки кортежу та модуль аналізу зв'язності. Модуль згортки кортежу здійснює обчислення узагальненого критерію оцінки для кожного завдання, що дозволяє більш продуктивно управляти процесом розподілу завдань на обчислювальні ресурси. Модуль

аналізу зв’язності дозволяє здійснювати підбір обчислювальних ресурсів з урахуванням скорочення часу, що витрачається на обмін між задачами у завданні. Якщо задачі в завданні мають високу зв’язність або для даного завдання необхідне пересилання великого обсягу вхідних або вихідних даних, то під час розподілу завдання на виконання акцент буде зроблений на підбір обчислювальних ресурсів аби зменшити час пересилки даних.

Додана база даних (БД), яка фіксує інформацію про кожне запущене завдання: час виконання завдання на кожному з ресурсів, дані про відмови, дані про стан ресурсів. Завдяки використанню запропонованої БД планувальник має можливість отримувати результати попередніх розподілів завдань за різними методами розподілу, що дозволяє обрати такий план розподілу, який матиме виграш за рядом критеріїв, заздалегідь встановлених користувачем. У процесі моделювання всі дані з log-файлів поміщаються до БД, яку можна використовувати для аналізу статистичної інформації в міру необхідності.

Результати дисертаційної роботи впроваджені у Радіоастрономічному інституті НАН України (відділ космічної радіофізики), м. Харків (акт від 30.11.2015 р.) та Харківському національному університеті радіоелектроніки, кафедра електронних обчислювальних машин, м. Харків в процесі проведення лекційних занять і лабораторних робіт з курсів «Паралельне моделювання на HPC системах» та «Інтерфейси паралельного програмування» (акт від 10.01.2017 р.).

Публікації. Матеріали дисертації достатньо повно викладені у 25 роботах: з них 7 статей у виданнях, які зазначені в переліку фахових видань України з технічних наук [19-25] (всі праці входять до науково-метричних баз, 1 – до бази Scopus [24]) та 18 тез доповідей міжнародних конференцій [26-43].

Список використаних джерел у даному розділі наведенj у повному списку використаних джерел під номерами: [1-44].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Місце обчислювальних систем в інформаційних технологіях

Під обчислювальною системою в області інформаційних технологій прийнято розуміти взаємопов'язану сукупність апаратних засобів (процесори, оперативна пам'ять, периферійні пристрої) і програмного забезпечення (ПЗ), які призначені для обробки інформації. Обчислювальні системи мають ряд ознак класифікації [45-48]. Універсальні системи орієнтовані на вирішення широкого кола задач, склад яких заздалегідь не визначений, а спеціалізовані спрямовані на вирішення вузького, певного класу задач. Спеціалізація систем досягається або за рахунок структури системи, або наявністю в її складі спеціального обладнання, для роботи з яким потрібне спеціалізоване програмне забезпечення (СПЗ).

До складу багатомашинних обчислювальних систем входить певна кількість обчислювальних станцій, кожна з яких в своєму складі має оперативну пам'ять та засоби обміну інформацією між собою. Кожна з станцій функціонує з використанням операційної системи (ОС), а обмін між ними здійснюється на рівні взаємодії ОС між собою. Область використання багатомашинних обчислювальних систем – це робота в умовах розпаралелювання за рахунок підвищення надійності обчислювальної системи. Якщо ж в якості елементів обчислювальної системи використовуються не відокремлені обчислювальні структури, а їх компоненти, наприклад, процесори, то таку систему прийнято називати багатопроцесорною [23].

Однорідні обчислювальні системи будуються на однотипних елементах. Вони характеризуються рядом переваг: спрощена розробка та обслуговування технічних та програмних засобів, модернізація та масштабування; є можливість стандартизації та уніфікації з'єднань, процедур взаємодії елементів системи. Неоднорідні обчислювальні системи будуються на різнотипних елементах, які відрізняються за технічними та функціональними характеристиками.

За характером розподілу елементів обчислювальної системи у просторі по-

діляються на системи локального (зосередженого) та розподіленого типів. Розподілені обчислювальні системи (РОС) – це системи з досить складною конфігурацією, елементи якої розосереджені в просторі, що використовують канали зв'язку для обміну інформацією між собою [47]. Локальні системи орієнтовані на мінімальне розосередження елементів обчислювальних систем в обмеженому просторі.

Обчислювальні системи, управління якими здійснюється за допомогою однієї обчислювальної станції (диспетчерської) називаються централізованими. Головними задачами диспетчерської обчислювальної станції є: розподіл навантаження між елементами системи, виділення необхідних ресурсів, контроль за станом цих ресурсів, координація їх взаємодії. Головним недоліком таких систем є те, що у разі непередбаченого припинення функціонування диспетчерської обчислювальної станції, система припиняє обслуговувати користувачів та взаємодіяти з іншими системами. У децентралізованих системах функції управління розподілені між її елементами, тому що кожна обчислювальна станція системи є автономною, а взаємодія між елементами системи встановлюється за допомогою спеціальних наборів команд. У обчислювальних системах з гібридним управлінням використовуються переваги роботи централізованих та децентралізованих обчислювальних систем: перерозподіл функцій в таких системах відбувається в ході обчислювального процесу та залежить від конкретної ситуації.

Оперативний режим роботи обчислювальної системи характеризується тим, що ефективність функціонування обчислювальної системи визначається часом роботи алгоритму. Серед таких систем є системи, які функціонують в реальному часі, тобто використовують наступні обмеження: м'яке та жорстке управління. Обчислювальні системи, що функціонують в реальному часі, характеризуються підвищеними вимогами до надійності функціонування та високим ступенем автоматизації процедур підготовки вхідних даних.

Оскільки інформаційні технології – це сукупність способів і методів застосування засобів обчислювальної техніки при виконанні функцій збору, зберігання, обробки, передачі та використання інформації (ГОСТ 34.003-90), то обчислювальні системи по праву є основоположною їх частиною.

1.2 Розподілені системи як середовище для рішення задач великої обчислювальної складності

На сьогодні існує проблема вирішення складних задач, які вимагають великих обчислювальних та інформаційних ресурсів. У разі виникнення ситуації браку обчислювальних та інформаційних ресурсів системи, для вирішення такої задачі використовуються розподілені обчислювальні системи.

Розподілена система – це набір незалежних обчислювальних станцій, які для користувача виступають єдиною об’єднаною системою, тобто: всі обчислювальні станції є автономними, а для користувачів ця сукупність ресурсів є єдиним пулом (єдиною системою) [49]. Таким чином РОС характеризується рядом особливостей: можливість надавати умови роботи з різними типами пристроїв, тобто бути гетерогенною; бути масштабованою; забезпечення постійного доступу до ресурсів; маскування особливостей комунікацій, тобто вводиться поняття прозорості для РОС (прозорий доступ до ресурсів).

Для забезпечення роботи складових РОС використовують спеціальний програмний рівень, який розташовується між прикладними програмами та різними ОС (рисунок 1.1).

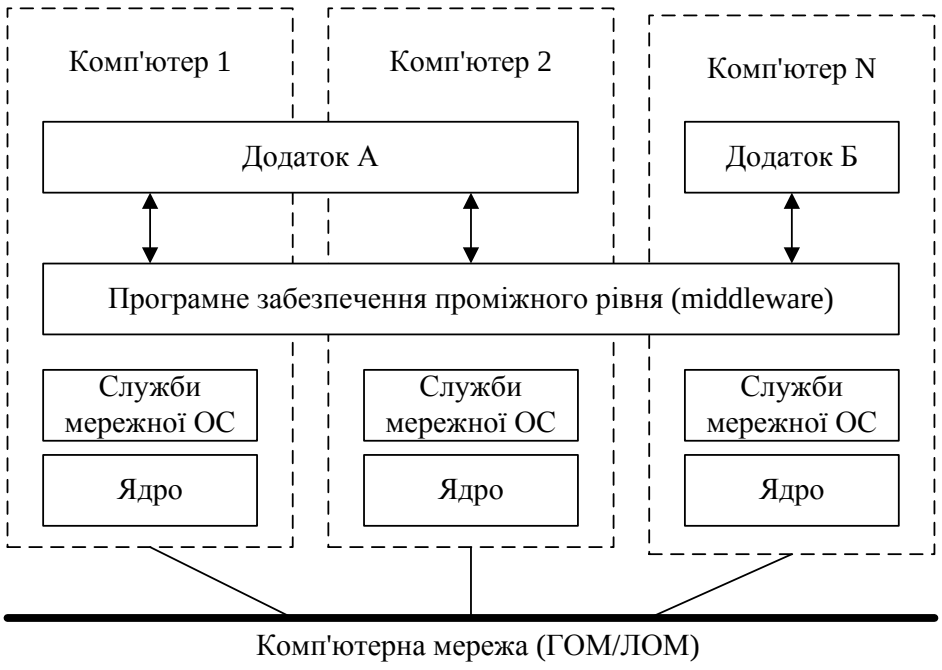

Рисунок 1.1 – Структура програмних складових в РОС

Даний рівень отримав назву програмного забезпечення проміжного рівня (ПЗПР) і орієнтований на реалізацію функціоналу розподіленої системи. Також він забезпечує абстрагування прикладних програм від базових платформ і приховує неоднорідність ресурсів, як від додатків, так і від користувачів [49]. ПЗПР полегшує доступ додатків до обчислювальних ресурсів системи та прискорює процеси взаємодії між ними.

1.3 Огляд Cloud-технологій та технологій розподілених обчислень

Перші розподілені обчислення були запущені для роботи мережі з середини 70-х років [50]. Однак це були лише перші спроби, які носили поодинокий характер. В даний час розрізняють три етапи розвитку розподілених обчислень. Перший етап прийнято датувати початком 1990-х років – це становлення розподілених обчислень. Перші проекти, пов’язані з розподіленими обчисленнями, були орієнтовані на обчислювальні ресурси суперкомп’ютерів (проекти FAFNER, I-WAY) [49].

У 1998 році починається другий етап розвитку: вводиться визначення терміну GRID, розподілені системи орієнтуються на масивні обсяги передачі інформації та обчислювальні витрати (проекти Globus, SETI@home (P2P)). Ян Фостер виводить три вимоги, яким повинні задовольняти GRID-системи: гетерогенність, масштабованість, адаптованість [5]. У 2001 році відбувається переорієнтація GRID в сторону створення віртуальних організацій: розвивається сервісно-орієнтований підхід (SOA), який дозволяє гнучко використовувати одні й ті ж обчислювальні ресурси багатьма користувачами; спостерігається автоматизація методів управління ресурсами (Globus, OGSA, WSRF). На даному етапі розвитку вводиться система класифікації РОС за двома напрямками [50]:

- розмір системи та засіб її адміністрування (кластер, корпоративні обчислювальні системи, GRID-система);

- функціональність системи (Computational Grid, Data Grid, Informational Grid, Hybrid Grid, Semantic Grid, передатні системи).

Під кластером прийнято розуміти об'єднання декількох однорідних обчислювальних станцій, які розглядаються як самостійні одиниці, що мають ряд специфічних властивостей.

Областю використання корпоративних обчислювальних систем є великі фінансові та виробничі компанії. Завдяки цим системам автоматизується управлінська діяльність за рахунок організації інформаційних систем, які дозволяють обслуговувати велику кількість користувачів в рамках одного напрямку (біржі, банківські системи, системи бронювання та продаж квитків на транспорт та ін.). Адміністрування в таких випадках носить «ручний режим», за рахунок невеликого розміру подібних систем, проте, як правило, з'являється потреба в автоматизації деяких процесів.

GRID-система – це узгоджена, відкрита та стандартизована система, яка забезпечує гнучкий, безпечний, скоординований розподіл ресурсів в рамках віртуальної організації [5]. Адміністрування в таких системах вбудовується в проміжне програмне забезпечення (ППЗ), що дозволяє представити даний процес автоматизованим. Розрізняють основні класи GRID-систем: Computational Grid, Data Grid, Informational Grid, Hybrid Grid, Semantic Grid.

Computational Grid – це система, яка орієнтована на розподілені обчислення. Data Grid застосовується у разі необхідності обробки великих потоків даних. Використання Informational Grid пов'язано з інтеграцією великих інформаційних сховищ даних. Hybrid Grid – це поєднання Computatuinal / Data Grid та Informational Grid, яке дозволяє використовувати переваги вищевказаних систем і створює ресурс, що володіє значними показниками продуктивності, які орієнтовані під певний клас задач. Інструментарій Semantic Grid використовується у разі опису семантики ресурсів будь-якої архітектури GRID [51].

Прикладами РОС служать системи, які поєднують обчислювальні ресурси високої продуктивності [52-54]; системи, які використовують процесорний час станцій, що простоюють [55,56]; Clouds Computing [57]. Протягом останніх 10 років були введені в експлуатацію наступні проекти: EGEE, NorduGrid, TeraGrid, Open Scince Grid. Наразі існують такі основні GRID-сайти: Folding@home (дослі-

джується згортання білків), Hydrogen@Home (триває пошук нових джерел водню на органічній основі для екологічно чистої енергетики), Clean Energy (триває розвиток принципово нового матеріалу для сонячних батарей), Docking@Home (здійснюється тестування малих молекул для нових ліків за допомогою молекулярного докінгу), AstroGrid@Home (обробка даних, які були отримані в результаті радіо- та оптичної астрономії) та ін.

В епоху інформаційних технологій без GRID-інфраструктури неможна уявити ряд наукомістких додатків, які вимагають високопродуктивних обчислень та багатопоточну (паралельну) обробку інформації. У зв'язку з цим, створення розподілених GRID-додатків – це актуальна задача, яка є складним процесом в порівнянні зі створенням звичайних послідовних програмних систем [52,58].

GRID-система – це сукупність розрізнених ресурсів (обчислювальних, інформаційних та ін.), що використовуються для вирішення наукових і математичних задач, які орієнтовані на високопродуктивні кластери та суперкомп'ютери [5,18,52,53]. На початку свого становлення GRID-системи використовувалися тільки для наукових або інженерних додатків. Однак згодом вони отримали широке застосування в комерційній інфраструктурі для вирішення завдань економічного прогнозування, розробки та вивчення властивостей ліків, криптоаналізу, сейсмоаналізу, радіоастрономії та ін.

Виділяють наступні області, в яких доцільно використовувати GRID-обчислення:

- моделювання складних процесів та систем:

а) моделювання структури та динаміки білків (біомедицина);

б) інтерактивне моделювання (кліматологія);

в) моделювання умов землетрусів в складних тривимірних геологічних моделях (фізика суші);

г) прогнозування поведінки вулканів (фізика суші);

- обробка, зберігання та аналіз масивів даних:

а) реконструкції реальних подій (фізика високих енергій);

б) аналіз медичних зображень (біомедицина);

в) аналіз великих макромолекулярних комплексів (біомедицина);

г) аналіз характеристик озонового шару (моніторинг атмосфери);

д) візуалізація великих масивів даних (радіо- та оптична астрономія).

GRID-технології активно використовуються в астрофізиці (MAGIC, Planck, ANTARES, NEMO), молекулярному моделюванні (CHARON, CompChem), нанотехнології, термоядерному синтезі, археології та в інших напрямках. Завдяки GRID-інфраструктурі спрощується співробітництво між географічно розподіленими спільнотами, що дозволяє спільно користуватися комп'ютерними ресурсами та даними. Тому в даний час число нових GRID-проектів постійно росте [58-61].

Хмара – це об'єднання комп'ютерів, які належать одному провайдеру, проте споживачі можуть орендувати доступ до цих обчислювальних ресурсів (Amazon's Elastic Compute Cloud, Google App Engine, IBM's Enterprise Data Centre Etc).

Якщо порівнювати GRID-системи та хмари, то варто визначити їх загальні риси: обидві забезпечують доступ до розподілених ресурсів та сервісів для користувачів, зменшують вартість організації обчислень, покращують їх надійність, відмовостійкість та гнучкість. При цьому GRID-системи орієнтовані на використання прикладних програм, а хмари – на використання сервісів [62].

Однак, варто виділити загальні риси та відмінності GRID-систем та хмар. У переважній більшості у якості провайдерів GRID-систем виступають співтовариства дослідних інститутів та університетів, які розташовані по всьому світу, а в хмарі – сервіс-орієнтовані організації, переважною більшістю – комерційні. Користувачами GRID-систем є віртуальні організації (ВО), які розміщені по всьому світу і складаються з співтовариств дослідних інститутів та університетів, а хмари використовуються малими та середніми комерційними фірмами, яким необхідні великі обчислювальні потужності. GRID-центри розташовуються в обчислювальних центрах, що можуть бути розподіленими за географічною ознакою, а хмари дислокуються в обчислювальних центрах провайдерів. GRID-системи були впроваджені для вирішення ряду завдань з обмеженим часом виконання, які генерують великі обсяги інформації, а хмари орієнтована на довгострокові сервіси та задачі з великим часом виконання [62].

Перевагами GRID-систем є:

- співпраця (GRID-система надає платформу для розподіленого співробітництва вчених);

- прозорість (GRID-технології є потенціально доступними для кожного, що надає гарантії використання даних ресурсів, тобто робить процеси прозорими);

- гнучкість (GRID-система розосереджена на множині обчислювальних систем, що зменшує ризик в разі відмови однієї з них).

Недоліками GRID-систем є:

- низька надійність (GRID-система базується на множині розподілених сервісів, які підтримуються персоналом на різних рівнях обслуговування);

- висока складність (побудова та експлуатація GRID-систем вимагає наявність спеціальних знань у обслуговуючого персоналу) [62].

Перевагами хмарних технологій є:

- гнучкість (користувач може масштабувати задіяні ресурси в режимі реального часу);

- висока надійність (провайдер ресурсів бере на себе зобов'язання щодо забезпечення якості наданої послуги: резервування, відмовостійкості);

- простота використання (сучасні хмарні технології характеризуються «дружнім» інтерфейсом).

Хмарні обчислення – це динамічно масштабований засіб доступу до зовнішніх обчислювальних ресурсів у вигляді сервісу, при цьому користувачеві не потрібно ніяких особливих знань про інфраструктуру хмари або навичок управління цією хмарною технологією [62-64].

Хмарна модель характеризується такими властивостями [65]:

- самообслуговування на вимогу (On-demand self-service) – користувач отримує доступ до ресурсів за потребою без взаємодії з провайдером послуг;

- широкий мережний доступ (Broad network access) – користувач отримує обчислювальні ресурси по мережі за допомогою стандартних механізмів, а також за допомогою тонких або товстих клієнтів;

- об'єднання ресурсів в пули (Resorce pooling), які динамічно змінюються,

відповідно до запитів користувачів;

- миттєва еластичність (Rapid elasticity) в залежності від динаміки запитів;

- вимірювані сервіси (Measured service): провайдер послуг веде автоматичне відстеження та контроль ресурсів, що забезпечує прозорість як для провайдера, так і для користувача.

Існує 4 моделі розгортання хмарних обчислень: приватна, публічна, комунальна (громадська) та гібридна [65].

1.4 Аналіз методів розподілу завдань в РОС

Головною задачею РОС, є задача розподілу обчислювальних ресурсів між завданнями [66-69], які надходять до обчислювальної системи. Процес планування може здійснюватися в двох режимах: статично або динамічно (в процесі надходження завдань на вхід системи). Функцію планування виконує спеціальна програма – планувальник (брокер) [70]. Він керує потоком завдань, які надходять від користувачів та проводить їх розподіл за наявними ресурсами. Це можливо виконати за допомогою інформації, яку надає користувач. Важливу роль в роботі РОС займають методи розподілу завдань на обчислювальні ресурси. Результатом роботи їх є розклад виконання завдань, які надійшли до обчислювальної системи.

На сьогодні не існує універсального методу, який здійснює розподіл для будь-якого класу задач та сукупності доступних ресурсів [19,71,72]. Як правило, виникає необхідність враховувати властивості конкретного класу задач, для яких даний метод використовується. Тому задача планування в GRID-системах відноситься до класу NP-повних задач [73]. Для її вирішення не існує поліноміальних алгоритмів. Однак існування таких задач змушує шукати шляхи, за допомогою яких можливе подолання даних труднощів. Наприклад, виділяють основні напрямки: знаходження рішень за допомогою евристичних методів, поліпшення переборних методів за рахунок введення ряду обмежень або умов, динамічне програмування, збір статистичних даних про виконані завдання (для навчання нейронної мережі).

1.4.1 Точні методи розподілу завдань

Метод критичного шляху (Critical path method) дозволяє скласти розклад та оцінити часові витрати, які можуть виникнути при розподілі завдань на обраний обчислювальний ресурс; здійснити попередній розрахунок часу закінчення завдання, що дозволяє прийняти рішення користувачу щодо подальших дій. Однак для цього методу обов'язковим параметром є наявність для кожного завдання тривалості його виконання. У зв'язку з тим, що постачальник завдань не завжди може оцінити час виконання задач із завдання, використання цього методу в GRID-системах не завжди є доцільним. Метод критичного шляху має дві модифікації стохастичну [74] та динамічну [75], які містять ймовірнісні припущення про час виконання завдання, що дозволяє використовувати їх в РОС.

Лінійне програмування – це ряд методів, які орієнтовані на рішення задач про пошук екстремумів лінійних функцій на множинах n-мірного векторного простору (задаються системами лінійних рівнянь та нерівностей) [76]. При вирішенні задачі розподілу завдань вводиться обов'язкове обмеження: рішення повинне бути цілочисельним, тобто в разі вибору ресурсу може бути лише два рішення (ресурс зайнятий (1) або ресурс вільний (0)). Ефективним для вирішення таких задач є алгоритм на основі рангового підходу [77-81], який заснований на розбитті n-мірного одиничного кубу на n рангів, кожен вектор якого має однакове число одиниць та нулів. До переваг такого підходу можливо віднести простоту та наочність представлення рішення задачі, поліноміальну складність, можливість реалізації на паралельних обчислювальних структурах.

Симплекс-метод – це один з інструментів для вирішення задачі лінійного програмування [78]. Недоліком цього методу є його експоненціальна складність, але він дозволяє за кінцеве число ітерацій знайти оптимальне рішення. В якості обмежень можуть бути використані як рівності, так і нерівності. Внаслідок того, що в даному випадку використовуються нерівності типу ≤ 1, то даний метод є ефективним, тобто рішення завжди є цілочисельним.

Динамічне програмування – це клас методів, які засновані на покроковому

виконанні завдання: на кожному кроці з множини допустимих рішень обирається одне, яке оптимізує цільову функцію [73]. Ці методи слід використовувати у тому випадку, якщо задачу можна розбити на однакові підзадачі, кожна з яких вирішується тільки один раз, а результат її вирішення заноситься до таблиці. Перевагами даного напрямку є те, що він дозволяє знайти глобальне оптимальне рішення, а рівняння Беллмана є зручним для програмування. Недоліком є розмірність задачі, тому що є необхідність зберігати результати оптимізації всіх етапів.

У разі вибору переборних методів, множина рішень є кінцевою величиною, тому що при повному переборі всіх можливих рішень задачі можна знайти найкраще з рішень. Недоліком цього методу є велика кількість розглянутих варіантів. Перевагою методу є його простота та високі показники ефективності для задач з малим числом вхідних даних або, коли є можливість звести велику кількість вхідних параметрів до одного. Але при збільшенні числа параметрів задачі збільшується число допустимих рішень та час пошуку оптимального рішення; таким чином, використання даного методу стає неефективним. Прискорити процес пошуку можливо за рахунок відкидання підмножини неоптимальних рішень, наприклад, особою, що приймає рішення. Така модернізація знайшла відображення в методі гілок та меж [82].

1.4.2 Наближені методи

В даний час при вирішенні задач планування широке використання отримали евристичні методи [83-86], правильність яких строго не доводиться, однак рішення, які отримані за їх допомогою, дають позитивні результати. Під евристикою розуміють не повністю математично обґрунтований, але корисний алгоритм. Евристика не гарантує знаходження оптимального та заздалегідь існуючого рішення та в окремих випадках може мати невірне рішення. При використанні евристичних методів для розподілу завдань в якості евристик слід задавати повний перелік правил планування [87-89].

Жадібні алгоритми – це прості, швидкі алгоритми, які на кожному кроці ро-

блять локально оптимальний вибір і надалі цей вибір не відміняється [90]. Жадібні алгоритми доцільно використовувати на початковому етапі, тобто при формуванні первинного плану. Однією із властивостей жадібного алгоритму є те, що з його допомогою вдається знайти оптимальне рішення, однак це буває не завжди.

Еволюційні алгоритми – це евристичні алгоритми пошуку, які використовуються для рішення задач оптимізації шляхом послідовного варіювання параметрів, які знаходяться [91]. Цей клас алгоритмів заснований на принципах природного відбору. В даний час існує декілька типів еволюційних алгоритмів: алгоритми еволюційних стратегій, еволюційного програмування, генетичні алгоритми [92,93], алгоритми генетичного програмування [94]. Дані алгоритми не завжди дають оптимальні рішення. Використання цього класу алгоритмів доцільно в тому випадку, коли спосіб точного рішення задачі невідомий, або, коли спосіб для точного рішення існує, проте він дуже складний в реалізації і вимагає великих затрат часу та обчислювальних потужностей.

Широке застосування набирають нейронні мережі, які успішно застосовуються в самих різних областях: медицині, бізнесі, фізиці та ін. Їх використовують там, де необхідно вирішувати задачі управління або класифікації, так як вони дозволяють відтворювати надзвичайно складні залежності. Так як нейронні мережі – нелінійні, то їх можливо використовувати в задачах, де лінійна апроксимація незадовільна, а наявні лінійні моделі дають лише наближені результати. Задачі лінійного програмування дуже часто стикаються з проблемою, відомою як «прокляття розмірності», яка не дозволяє моделювати лінійні залежності в разі великого числа змінних. Для нейронних мереж ця проблема є цілком вирішуваною. Нейронні мережі характеризуються функцією навчання: користувач підбирає дані, а після цього запускається алгоритм навчання [95].

Використання нейронних мереж в GRID-системах має ряд переваг:

- у ході роботи накопичуються статистичні дані, які в подальшому можуть бути використані для розподілу завдань;

- немає необхідності кожного разу під час додавання завдання заново здійснювати процес прогнозування та витрачати додатковий час на отримання проміж-

жних даних;

- після навчання мережа здатна спрогнозувати значення якоїсь послідовності на основі декількох попередніх значень і/або якихось існуючих на даний час чинників.

Перевагою використання нейронних мереж є висока швидкість та точність отримання результатів, проте існує декілька недоліків: складність аналізу, особливості програмної та апаратної реалізації.

1.4.3 Найпростіші алгоритми планування

First-Come First-Served (FCFS) – це найпростіший алгоритм планування, при якому розподіл завдань на ресурси здійснюється в тій послідовності, в якій вони надійшли до обчислювальної системи. Перевагою алгоритму є простота реалізації. Серед недоліків варто зазначити: середній час очікування та середній час виконання завдання залежать від порядку розташування завдань в черзі [96]. Якщо на початку черги знаходяться завдання, які вимагають для виконання великої кількості ресурсів, то може виникнути ситуація, при якій інші завдання, що вимагають менший обсяг ресурсів, будуть не задіяними за умови, що для їх виконання є необхідні ресурси.

При використанні алгоритму Highest Priority First (HPF) ресурс надається для виконання того завдання, яке має найвищий пріоритет. Однак можуть виникати ситуації, коли високопріоритетні завдання можуть задіяти обчислювальні ресурси на тривалий час, внаслідок чого завдання з більш низьким пріоритетом не отримають ресурси. Так само існують випадки, коли для виконання високопріоритетного завдання в системі немає необхідного ресурсу. У разі виникнення такої ситуації завдання знаходиться в черзі та очікує звільнення ресурсу. Однак для менш пріоритетного завдання ресурс є, але це завдання не може зайняти його через те, що попереду нього є завдання з більш високим пріоритетом. Таким чином, виникає ситуація нераціонального використання обчислювальних ресурсів. Обов’язковою умовою використання методу HPF є необхідність визначення па-

раметрів, які стануть за основу для формування пріоритету [19].

Класичний метод зворотного заповнення BackFill використовується для вирішення задачі розподілу на базі великих багатопроцесорних систем [97]. Ідея методу полягає в наступному: обчислювальні ресурси виділяються для завдань не безпосередньо в момент їх звільнення, а завчасно, тобто планувальник будує план розподілу ресурсів. Перевагою цього методу є те, що він гарантує отримання ресурсів для високопріоритетних завдань за мінімально можливий час та допускає порушення порядку черги, що дозволяє завданням з більш низьким пріоритетом задіювати обчислювальні ресурси, які на даний час не використовуються жодним із завдань. Ці дії спрямовані на підвищення коефіцієнта загального навантаження обчислювальних ресурсів GRID-системи [98-100].

1.5 GRID-ініціативи та проекти в світі

Ян Фостер у своїй роботі [101] ввів розмежування GRID-систем по науковим та комерційним результатам за допомогою 3-х критеріїв. В якості цих критеріїв виступають визначення, відповідно до яких GRID – це система, яка:

- координує використання ресурсів при відсутності централізованого управління ними;
- використовує стандартні, відкриті, універсальні протоколи та інтерфейси;
- нетривіальним чином забезпечує високоякісне обслуговування.

Для управління GRID-системою необхідно ППЗ, яке реалізує уніфіковані механізми доступу до ресурсів GRID. Введемо умовний розподіл ППЗ за трьома напрямками. До перших віднесемо ПЗ, яке відповідає критеріям, перерахованим в роботі [101]: Globus toolkit, NurduGrid ARC, gLite (є основним ППЗ, за допомогою якого будують свої дослідження вчені, в тому числі вчені України).

Другий напрямок – системи Condor, Unicore, Legion – це доступні та стабільні середовища, які використовуються для організації розподілених обчислень, однак вибір певного середовища залежить від функцій, які воно буде виконувати.

До третього напряму належать GRID-полігони, які створюються для вирі-

шення конкретної задачі на визначеному типі апаратного та програмного забезпечення. В даний час до GRID-полігонів можна віднести наступні проекти: EGEE [102], Tera Grid [103], NorduGrid [104,105], Open Science GRID (OSG) [106], MAPPER [107]. Українські лабораторії мають машини, які підключені до GRID EGEE, NorduGrid.

В роботі [22] наведено аналіз сучасного стану GRID-технологій, проаналізовано переваги та недоліки запропонованого ринком ППЗ, головною функцією якого є надання користувачеві зручного інтерфейсу між додатком та необхідними обчислювальними ресурсами.

1.6 Огляд технологій існуюччого прикладного програмного забезпечення

В даний час існує множина посилань на статті [22,108-110], в яких авторами ставиться мета проаналізувати переваги та недоліки існуючого ППЗ.

1.6.1 Планувальник Condor

Широке поширення набуло ПЗ Condor [111], яке було створено в університеті штату Вісконсін (США) для об’єднання комп’ютерів університету. Це ПЗ було використане для вирішення ряду завдань, які вимагають велику кількість обчислювальних ресурсів. Condor дозволяє розподіляти завдання як на відчужені, так і не невідчужені ресурси, що дозволило ефективно використовувати їх обчислювальні потужності.

У 2012 році Condor був перейменований в HTCondor. ПЗ HTCondor є доступним для користувачів та підтримує механізми міграції завдань, призначення пріоритетів, створення контрольних точок, віддалений виклик процедур, а також стратегію планування та моніторингу завдань.

У планувальнику HTCondor використовується: потужна мова опису ресурсів, механізми безпеки та підтримання цілісності даних, забезпечується надійна передача інформації між компонентами системи, гарантується захист даних кори-

стувача та власника ресурсу. Одним з можливих способів аутентифікації в системі є GSI аутентифікація на основі публічних ключів з використанням сертифікатів. На відміну від традиційних пакетних систем HTCondor дозволяє здійснювати рівномірне завантаження обчислювальних ресурсів для запуску завдань, які надійшли до системи [112] та має гнучку систему розподілу ресурсів, яка реалізована за допомогою мови ClassAd (Classified Advertisement).

Перевагою ПЗ HTCondor є його постійне оновлення. На даний час планувальник HTCondor може бути встановлений на різні апаратні платформи: Intel x86, HP PA-RISC, Sun, PowerPC, Itanium IA64 та на всі ОС: Windows, RedHat Linux, Solaris, Debian Linux, Fedora Core, HPUX, Macintosh OS, AIX та ін. [113].

Недолік ПЗ HTCondor полягає в тому, що серед вимог до адміністратора GRID-системи додається значний обсяг роботи, який вимагає ручного опису стратегій планування на всіх рівнях функціонування системи, тому що дана система не має властивостей адаптивності до умов, що були змінені.

1.6.2 Система DIET

Система DIET [15,114] (рисунок 1.2) використовується для організації розподілених обчислень для задач великої розмірності. Архітектура системи, включає в себе ряд компонентів:

- клієнт (Client) – додаток, який надходить в систему на виконання;
- керуючий агент (Master Agent, MA) – працює із запитом від клієнта, збирає дані про вільні обчислювальні ресурси, знаходить необхідний ресурс та передає його адресу клієнту;
- місцевий агент (Local Agent, LA) – орієнтований на передачу запитів та інформації між керуючим агентом та обчислювальним ресурсом, відповідає за локальне планування в рамках конкретної обчислювальної системи;
- серверна служба (Server Deamon, SeD) – встановлюється на кожному ресурсі для моніторингу інформації його стану (показник завантаженості, показник задіяності ресурсів, тип вирішуваних завдань).

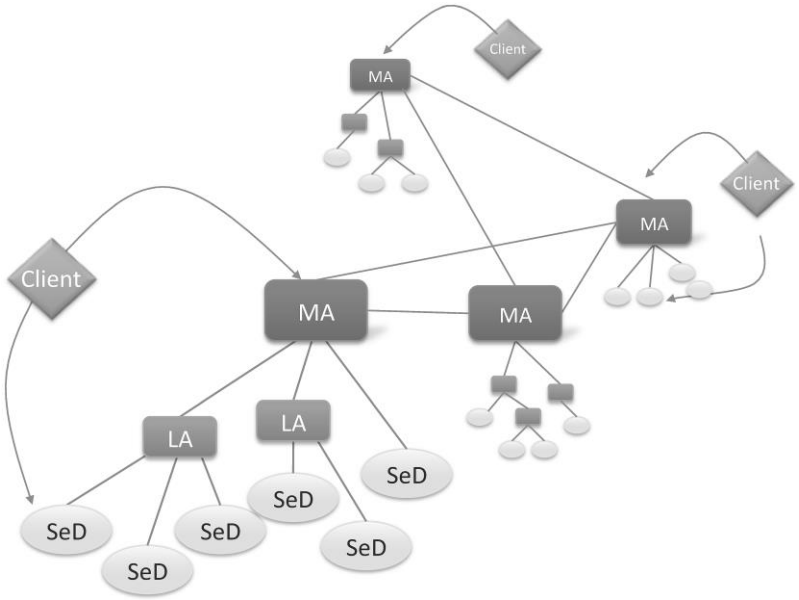

Рисунок 1.2 – Архітектура системи DIET

При надходженні до системи запиту від клієнта – система аналізує інформацію, що надходить від серверних служб до керуючого агенту про стан обчислювальних ресурсів. Всі дані, що надходять на керуючий агент, сортуються залежно від обраного критерію оптимізації. Однак ця функція може бути відключена, і система DIET вибере обчислювальний ресурс випадковим чином. Розробники DIET акцентують увагу на те, що ця система проста у використанні, тому що не вимагає введення інформації про ресурсні вимоги завдань. Її перевагою є те, що вона дозволяє модифікувати свій планувальник шляхом підключення до нього додаткових плагінів, які дозволяють розширювати множину метрик. Однак в даному випадку при розробці плагінів проблема планування перекладається на розробників, що є суттєвим недоліком, бо це вимагає наявність специфічних знань у адміністратора системи, який буде вносити зміни для кожної з прикладних задач.

1.6.3 Системи управління ресурсами PBS, TORQUE та SLURM

Portable Batch System (PBS) – це система управління ресурсами в високопродуктивних обчислювальних кластерах під керуванням ОС UNIX/Linux. На даний час існує безкоштовна версія OpenPBS та комерційна – PBS Pro [115].

У складі PBS існує власний планувальник (алгоритм FIFO), який дозволяє

використовувати різноманітні політики розподілу завдань. Планувальник PBS дозволяє вводити обмеження для завдань, які надходять у систему, здійснювати постановку, вивантаження та перерозподіл завдань в системі, призначати пріоритети завдань. В ході роботи постійно відбувається збір відомостей про обчислювальні ресурси. В системі PBS використовується кілька черг, які відповідають набору пріоритетів (чим більше часу завдання знаходиться в черзі, тим вище його пріоритет виконання). У черзі завдання упорядковані відповідно до часу надходження і мають межу очікування – 24 години.

Перевагою PBS є те, що вона групує ресурси в набори віртуальних вузлів за певними ознаками. За їх допомогою в PBS реалізується попередня резервація часу для вхідних завдань, що дозволяє більш ефективно налаштовувати цю систему. Планувальник PBS орієнтований на балансування завантаження обчислювального ресурсу, тобто якщо буде встановлено значення максимального завантаження, то планувальник буде розподіляти завдання на обчислювальні ресурси так, аби сумарне завантаження не перевищувало попередньо задане значення.

TORQUE (Terascale Open-Source Resource and QUEue Manager) – це одна з сучасних версій PBS, яка поширюється під вільною ліцензією OpenPBS Software License [116] і орієнтована на роботу під UNIX-подібними ОС. Даний менеджер дозволяє управляти ресурсами (початок виконання, зупинка або видалення завдання з черги) та здійснювати моніторинг системи (стан обчислювальних вузлів, інформація про поточний стан завдання тощо). Завдання, що надходять в систему, отримують пріоритет, а планувальник вибирає з черги саме те завдання, що має найвищий пріоритет, і відповідно до обраної політики планування, розподіляє їх між обчислювальними ресурсами системи.

SLURM (Simple Linux Utility for Resource Management) – це надійна, відкрита, добре масштабована система управління ресурсами кластера, яка орієнтована на роботу для різнотипних Linux-кластерів.

Системи управління ресурсами TORQUE та SLURM схожі. На кожному обчислювальному ресурсі встановлюється сервіс Machine Oriented Miniserver (для TORQUE) або SLURM Daemon (для SLURM), який здійснює управління обчис-

лювальними ресурсами даного вузла. Також існує центральний сервіс PBS Server (для TORQUE) або SLURM Control Daemon (для SLURM), який управляє чергою завдань, взаємодіє з планувальником ресурсів, координує роботу обчислювальних ресурсів та здійснює моніторинг їх стану. В системі SLURM для забезпечення відмовостійкості передбачений додатковий екземпляр центрального сервісу, який дублює стан головного. В якості планувальника ресурсів в обох системах використовується планувальник FIFO, однак є можливість підключення зовнішнього модуля, наприклад, Maui Cluster Scheduler (MCS) або Moab Workload Manager (MWM).

1.6.4 Планувальники Maui и Moab

Maui – це планувальник з відкритим кодом, який використовується в паралельних та розподілених обчислювальних системах. Maui орієнтований на роботу з UNIX подібними ОС [117]. Перевагами планувальника є: робота з великим набором методів планування, які може налаштовувати адміністратор; зменшення часу очікування завдань в черзі (за рахунок використання методу Backfill); можливість використання розширеної статистики: завантаженості окремих обчислювальних ресурсів, системи в цілому, стану конкретного завдання, стану черг; використання автоматичної системи визначення пріоритетів завдань.

Планувальник Maui взаємодіє з системою управління ресурсами, яка сповіщає його про стан обчислювальних ресурсів в системі. Вимоги, що виставляються завданням до ресурсу, асоціюється в Maui з поняттям «клас» (може бути кілька класів, які згруповані за окремими атрибутами). У зв’язку з цим для кожного сформованого «класу» планувальник Maui виділяє окрему чергу.

На кожному кроці планування Maui збирає статистику системи управління ресурсами щодо стану обчислювальних ресурсів, навантаження системи в цілому та обрану політику планування. Далі він отримує множину завдань, які можуть бути виконані та сортує їх відповідно до пріоритету. Пріоритет визначається на підставі інформації про розмір завдання, часу знаходження його в черзі, власника

завдання та ін. Для кожного завдання здійснюється підбір обчислювального ресурсу, що його задовольняє, з подальшим його запуском. Далі здійснюється сповіщення системи управління ресурсами про недоступність деяких ресурсів на момент виконання конкретного завдання на них.

Перевагою планувальника Maui є підтримка функції резервування ресурсів, яка реалізується за рахунок задання тимчасових інтервалів і списків доступу. Недоліками виступають: необхідність вказівки точних умов вибору обчислювальних ресурсів, задання чітких обмежень щодо політики доступу, визначення правил роботи з чергами, формування «класів» завдань.

З 2005 року проект отримує назву Moab Cluster Suite. Використання планувальника Moab [118] направлено на рівномірне та ефективне використання ресурсів для завдань, що надійшли до системи. Планувальник Moab може бути інтегрований в TORQUE та SLURM і працює в двох режимах: в режимі опитування системи управління обчислювальними ресурсами та в режимі подій, що надходять.

На кожному кроці планувальник Moab звертається до системи управління ресурсами з метою збору статистики про стан обчислювальних ресурсів, завантаження системи в цілому, переліку заданих на даний момент політик планування [118]. На вхід системи Moab подається перелік завдань, які він сортує відповідно до відносного пріоритету. Даний пріоритет Moab вираховує сам на підставі інформації, яка надходить з системи: розмір, власник завдання, час знаходження завдання в системі та ін. Після того як завдання впорядковані за пріоритетом відбувається підбір обчислювальних ресурсів відповідно до потреб завдання. Якщо для всіх задач даного завдання підібрані обчислювальні ресурси, то завдання відправляється на виконання, а до системи управління ресурсами надходить повідомлення про те, що задіяні ресурси для виконання завдання не можуть задіяними в подальшому розподілі.

Перевагою планувальника Moab є гнучкість в налаштуванні розподілу ресурсів за рахунок того, що в ньому використовується модульний підхід. Однак цей підхід є і недоліком, тому що для кожного завдання адміністратору системи необхідно задавати ресурсні вимоги.

Хоча планувальник Moab в даний час має більш значні переваги щодо планувальника Maui [119,120], використання планувальника Maui в системах розподілу ресурсів є пріоритетним.

1.7 Моделювання процесів в GRID-системах

GRID-система – це сукупність розрізнених ресурсів (обчислювальних, інформаційних та ін.), які використовуються для вирішення наукових і математичних задач, орієнтованих на високопродуктивні кластери та суперкомп'ютери. Однак створення GRID-системи вимагає значних матеріальних ресурсів. Економічно-обґрунтованим варіантом є побудова моделі, яка дозволить розробити структуру РОС з урахуванням задач, які вона вирішує, та досліджувати її властивості.

Моделювання роботи системи – це складова процесу прогнозування продуктивності та розробки ефективних механізмів розподілу завдань, що надходять в систему, які дозволяють зменшити час простою їх в чергах [121,122]. Розподіл ресурсів – це процес, що включає в себе ряд етапів: визначення множини ресурсів, які відповідають заданим вимогам для вирішення завдання; резервування ресурсів; планування виконання завдань; моніторинг системи.

Для досягнення цих цілей необхідна розробка ефективного алгоритму розподілу ресурсів та планування їх використання, що спричиняє за собою необхідність багаторазового повторення ряду експериментів. Проведення таких експериментів часто є неможливим через розподіленість та гетерогенність GRID-систем:

- ресурси системи належать різним користувачам, що ускладнює управління експериментом;

- потреби користувачів та можливості постачальників непостійні в часі, що ускладнює проведення повторного експерименту;

- створення працездатної GRID-системи для потреб проведення конкретного експерименту протягом тривалого часу за умови використання великої кількості ресурсів є довгим та економічно необґрунтованим процесом.

Отже, моделювання є оптимальним рішенням, яке дозволяє досліджувати

GRID-систему та проводити аналіз її поведінки як системи в цілому, так і окремих частин. Для моделювання GRID-систем застосовують різні підходи: побудова аналітичних та статистичних моделей [52]. Властивості елементів та систем задаються в аналітичній формі з урахуванням відокремленості одних параметрів від інших, лінійності ряду залежностей, таблиці переходів між станами та ін. Однак в деяких випадках доводиться вводити припущення, які призводять до значних відмінностей між моделлю та об'єктом і виникненню помилок в процесі опису системи. Тому для цих цілей підходить імітаційне моделювання, яке є одним із інструментів для дослідження поведінки реальних систем. Воно ґрунтується на тому, що математична модель відтворює процес функціонування елементарних подій, які протікають протягом деякого часу в системі зі збереженням логіки їх взаємодії. Часто постановка реальних експериментів є важкою, тоді єдиним рішенням залишається побудова комп'ютерної моделі [123]. Така модель дозволяє проводити обчислювальні експерименти, метою яких є збір, аналіз та інтерпретація результатів моделювання, а також зіставлення отриманих результатів з реальною поведінкою досліджуваного об'єкта [21].

Перевага імітаційного моделювання – це його універсальність, тому що з його допомогою можливо досліджувати системи довільної складності та не обмежувати рівень деталізації моделі. На рівні алгоритмів є можливість відтворення складних зв'язків між елементами системи, а також дослідження процесів їх функціонування.

1.8 Аналіз засобів програмної реалізації завдань в GRID-системах

Важливе місце в організації GRID-систем займає функція управління завданнями, за яку відповідає система управління робочими потоками WMS (Workflow Management System) [124]. До її складу входить набір програмних компонентів, що виконують розподіл завдань за наявними ресурсами, а також управління ними. Будь-яка реалізація цієї системи використовує специфічну мову для опису задач і управління робочими потоками. Файли опису завдання не є стандар-

тизованими, тому в різних GRID-системах можуть використовуватися різні формати опису.

Як правило, опис завдання містить наступну інформацію:

- інформація про постачальника завдань зберігається в обліковому запису системи та може бути використана для надання пріоритету завданням або визначення класу ресурсів, на якому дане завдання може виконуватися;

- вимоги завдання до ресурсів включають в себе обумовлений заздалегідь перелік обмежень для програмного та апаратного забезпечення;

- цілі розподілу можуть виступити в якості обмеження при виборі методів планування та ресурсів, на яких завдання буде запущено.

Файл опису завдання може містити додаткову інформацію: ім’я завдання, формат виведення вихідних даних та помилок в разі їх виникнення, місце зберігання вхідних та вихідних даних, вимоги щодо одночасного запуску завдань тільки на одному обчислювальному ресурсі. Існує множина форматів опису завдань: RSL/xRSL, ClassAd, JDL, JSON, YAML, XML, JSDL/JSDL-WG та ін.

Мови RSL (Resource Specification Language) та xRSL (eXtended Resource Specification Language) використовуються для передачі інформації та вимог до віддаленого ресурсу [125,126]. У якості вимог можуть виступати наступні параметри: архітектура, оперативна пам’ять, кількість процесорів та ін. Версії 1.0 та 1.1 мови RSL використовуються в системі GRAM5 на платформі Globus Toolkit, а мова xRSL застосовується в платформі ARC. Відмінність мов RSL та xRSL полягає в наявності нових атрибутів, рівнях специфікації (User-side RSL – набір атрибутів, що задаються користувачем у файл-специфікації; GridManager-side RSL – набір атрибутів з призначеного для користувача файл-специфікації, перероблених за певними правилами для передачі безпосередньо в GridManager).

Мова RSL містить два типи параметрів, які розрізняються за змістом і способом обробки: обмеження на ресурси, які формуються за допомогою завдання імен атрибутів та локальні параметри, які містять інформацію про завдання. Далі по заданих атрибутам здійснюється підбір відповідних ресурсів за допомогою спеціалізованого брокера, який орієнтується на дані про поточний статус ресурсів,

які отримані від інформаційного сервісу. Результатом роботи брокера є множина локальних менеджерів ресурсів, що підходять для виконання певного завдання.

ClassAd (Classified Advertisement) – є однією з перших мов опису завдань, яка розроблена для систем управління робочими потоками (заснованих на Condor). Мова ClassAd дозволяє користувачу створювати описи ресурсів та завдань, які надходять на виконання. Мова є масштабованою, що дозволяє легко додавати нові атрибути. В Condor на основі мови ClassAd реалізований механізм, який здійснює пошук ресурсів для кожного завдання з черги завдань, шляхом зіставлення інформації про обчислювальні ресурси та вимоги завдань. Мова ClassAd стала платформою для побудови мови JDL [127].

Job Description Language (JDL) – одна з високорівневих мов опису завдань в GRID-системах. Атрибутами JDL є запити, що визначають графік роботи та розподіл завантаження в обчислювальній системі. Мова використовується для опису задач в системах управління робочими потоками для платформи gLite: Computing Resource Execution And Management (CREAM) та WMS.

Атрибути мови JDL бувають обов'язковими та другорядними. У разі відсутності обов'язкових атрибутів WMS не зможе виконати пошуковий запит, а для другорядних атрибутів система самостійно визначить ряд значень за замовчуванням для обробки запиту [128].

Для опису завдань в UNICORE використовується мова JSON (JavaScript Object Notation) – текстовий формат обміну даними, в основі якого лежить використання мови JavaScript [129]. У мові є інтуїтивно-зрозумілий графічний інтерфейс, що полегшує користувачеві роботу з ним. Мова будується на основі двох універсальних структур даних: наборі пар ключ-значення та пронумерованому наборі значень, які реалізовані за допомогою наступних форм: об'єкт, одновимірний масив, значення, рядок.

YAML (Yet Another Markup Language) – формат для структурування даних в файлі [130]. Ця мова розглядалася як конкурент XML, але пізніше була перейменована з метою привернення уваги на, власне, дані, а не розмітку документів. Мова є масштабованою, оптимізована для роботи в конфігураційних файлах, серіалі-

зації даних, log-файлах, різних типів повідомлень і фільтрації даних.

XML (eXtensible Markup Language) – це платформо-незалежна розширювана мова подання даних, яка розроблялась для розміщення інформації в мережі Інтернет [131]. За допомогою мови XML вирішується ряд задач, наприклад, подання документів будь-якого типу в структурованому (ієрархічному) вигляді, сортування, фільтрація та пошук інформації. Стандартом визначено два рівні XML: вірно сформований та дійсний документ. Перший тип підпорядковується загальним правилам синтаксису XML (кожен елемент XML повинен містити початковий і кінцевий тег, будь-який вкладений елемент повинен бути повністю визначений всередині елемента, до складу якого він входить, документ повинен мати лише один елемент верхнього рівня). Імена елементів XML є чутливими до регістру.

Другий тип документу підпорядковується семантичним правилам (більш сувора додаткова перевірка коректності документа на відповідність встановленим правилам). Перераховані правила розробляються як самим користувачем, так і сторонніми розробниками. Обумовлені правила зберігаються в спеціальних файлах (схемах), де детально описана структура XML: допустимі назви елементів, атрибутів документу та ін.

Мова опису подання завдань Job Submission Description Language (JSDL) – це мова, яка призначена для опису вимог завдань до обчислювальних ресурсів. У мові JSDL присутній словник та нормативна XML-схема, які спрощують опис вимог [132]. Базовий словник використовує поняття, які присутні в ряді існуючих систем: Condor, Globus Toolkit, Load Sharing Facility (LSF), Uniform Interface to Computing Resources (Unicore) та ін.

Для високопродуктивних обчислювальних систем розроблено розширення JSDL-WG, яке спирається на використання елементів мови XML. Перевагою цього розширення є можливість перевірки структури та типів даних на відповідність заданій схемі. Елементи JSDL поділяються на три категорії: вимоги для ідентифікації завдання, вимоги на обчислювальні ресурси та вимоги на дані.

У якості елементів, які є ідентифікаторами завдання, виступають: ім’я завдання; ім’я проекту, в який входить завдання; опис завдання. При описі ресурсів

можна задавати такі дані:

- набір вузлів, які можуть бути обрані для запуску завдання;
- тип файлової та операційної системи, архітектура центрального процесора;
- діапазон швидкодії для кожного центрального процесора;
- діапазон кількості центральних процесорів, які необхідні кожному індивідуальному ресурсу;
- максимальна ємність системи збереженні даних окремого ресурсу;
- діапазон, що визначає необхідне загальне число секунд роботи центрального процесора для всіх пристроїв, які використовуються для виконання завдання;
- необхідна кількість процесорів, які необхідні для розміщення завдання;
- кількість встановленої фізичної пам'яті на всіх виділених ресурсах для всього завдання;
- обсяг віртуальної пам'яті для всього завдання на всіх ресурсах;
- загальна кількість ресурсів, які необхідні для виконання завдання.

При описі даних визначаються файли, які повинні бути переміщені на виконуючий вузол (вхідний потік) і файли, які необхідно взяти з виконуючого вузла (вихідний потік). Вхідні файли переміщаються до початку виконання завдання, вихідні файли після закінчення роботи завдання.

Мова JSDL використовується на платформах UNICORE та ARC [133].

1.9 Висновки по розділу та постановка задач дисертаційного дослідження

В даний час РОС – це повністю сформована сфера для високопродуктивних обчислень. У процесі розвитку інформаційних технологій з'являються нові умови щодо розробки та впровадження нових концепцій побудови РОС, розширення класів розв'язуваних задач та їх уніфікація.

Одним з класичних підходів в організації швидких обчислень є технологія GRID-обчислень. GRID-системи характеризуються різноманіттям способів реалізації, взаємодій, компонентів, способів виконання різних обчислень. Однією з актуальних задач є розробка прикладного програмного забезпечення, головною фу-

нкцією якого є надання користувачеві GRID-системи зручного інтерфейсу між додатком і необхідними обчислювальними ресурсами.

Велика ніша в даному питанні відводиться програмам, які здійснюють розподіл завдань по ресурсах – планувальникам (брокерам). Основна задача планувальника полягає в побудову плану розподілу, який задовольняє вимогам постачальників завдань (тобто при формуванні плану необхідна оптимізація параметрів цільової функції, за рахунок яких відбувається скорочення часу виконання завдань, що надійшли до системи).

Відомі планувальники [16,17,83,87,93,134-138] мають ряд недоліків. Головним з яких – є орієнтація на конкретний клас задач. Це пояснюється тим, що дані планувальники створювалися для кластерних систем, які в свою чергу організовувалися в GRID-кластери, а завдання продовжували виконуватися локально. Більшість сучасних планувальників не приймають до уваги ціну використання обчислювального ресурсу, що призводить до неефективного використання ресурсів обчислювальних систем.

Таким чином ставиться науково-технічна задача розробки нового підходу до розподілу завдань в гетерогенних GRID-системах, який орієнтований на використання нової інформаційної технології розподілу завдань. Даний підхід повинен враховувати та усувати недоліки в існуючих рішеннях задач розподілу завдань, які були виявлені в ході порівняльного аналізу.

Для досягнення поставленої мети необхідно вирішити наступні наукові та практичні задачі:

- аналіз існуючих методів розподілу завдань в розподілених обчислювальних системах;

- модифікація математичної моделі розподілу завдань в гетерогенних GRID-системах;

- модифікація методу розподілу завдань (Backfill) на обчислювальні ресурси;

- розробка методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів;

- розробка інформаційної технології розподілу завдань на обчислювальні ресурси на основі запропонованої математичної моделі розподілу завдань;

- проведення ряду експериментів з розподілу завдань на основі запропонованої інформаційної технології в системі імітаційного моделювання;

- апробація моделі та методів при вирішенні практичних завдань.

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [5,15-19,21-23,45-138].

2 ДОСЛІДЖЕННЯ МОДЕЛІ ПОДАННЯ ДАНИХ І МЕТОДІВ РОЗПОДІЛУ ЗАВДАНЬ У ГЕТЕРОГЕННИХ GRID-СИСТЕМАХ

2.1 Дослідження існуючої моделі подання завдань і ресурсів в GRID-системі

GRID-система – це складний об'єкт, який містить сукупність взаємопов'язаних різнорідних ресурсів, функціонування яких підпорядковується ряду заданих правил [58,60,72,96-99]. Однією з головних задач такої системи є узгоджений розподіл ресурсів для вирішення завдань, які надходять до неї.

Модель розподілу завдань в GRID-системі може бути побудована на основі: множини обчислювальних ресурсів R, множини завдань Z і методу розподілу q, тобто $G = \{R, Z\}$.

Завдання, які поступають до GRID-системи, утворюють потік $\{Z_i, i = 1, 2, .., M\}$, де i – це порядковий номер завдання, а M – кількість завдань. Кожне завдання характеризується рядом параметрів, які необхідні для їх виконання в GRID-середовищі (2.1):

$$Z_i = \{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z\}, \ \forall i = 1..M, \qquad (2.1)$$

де ar_i (architecture) – архітектура процесора;

os_i (operating system) – операційна система;

pc_i (processor count) – кількість процесорів;

ps_i (processor speed) – швидкодія процесорів;

ms_i (memory size) – обсяг оперативної пам'яті;

dc_i (disk capacity) – доступний обсяг системи збереження данних;

pr_i (priority) – пріоритет завдання.

Завдання – це пакет задач, які споріднені за певною ознакою. Тобто, кожне завдання – це програма, яка запускається на виконання в GRID-середовищі. Всі задачі, які входять в завдання, повинні запускатися на виконання одночасно. Якщо завдання підлягає декомпозиції у розрізі виконання на декількох обчислюва-

льних ресурсах, то виконання такого завдання можливе тільки в тому випадку, якщо для всіх задач із завдання будуть підібрані обчислювальні ресурси, що задовольняють вимогам постачальників.

Обчислювальні ресурси в GRID-системі можна представити у вигляді множини $\{R_j, j = 1, 2, .., N\}$, де j – номер обчислювального ресурсу, а N – число ресурсів в GRID-системі. Будь-який обчислювальний ресурс, що є складовою GRID-системи, можна представити у вигляді кортежу (2.2):

$$Rj = \{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r\}, \ \forall j = 1..N, \qquad (2.2)$$

Сукупність обчислювальних ресурсів є динамічною сутністю: відбувається видалення або додавання обчислювального ресурсу до GRID-системи, змінюються характеристики ресурсів. Ресурсами можуть бути: обчислювальні кластери, ноди, інші обчислювальні структури (наприклад, робочі станції, які надають користувачу доступ до оперативної та віртуальної пам’яті, дискового простору, а також процесорів).

У різних системах розподілу завдань вхідні кортежі (2.1 та 2.2) можуть мати несуттєві відмінності. Це пов’язано з тим, що структура GRID-систем з часом може набувати нових характеристик, що задовольняють додатковим вимогам постачальників завдань.

2.2 Дослідження існуючих підходів вирішення розподілу завдань в GRID-системі

2.2.1 Дослідження технологій розподілу завдань на обчислювальні ресурси

Вирішення задачі розподілу завдань на обчислювальні ресурси є основною для GRID-систем. Вона полягає в побудові ефективного плану розподілу завдань на множину обчислювальних ресурсів GRID-системи.

Задача розподілу завдань в GRID-системі може бути представлена у вигляді повного орієнтованого дводольного графа (рисунок 2.1). Вагами дуг графа W_{ij}

(обмеження на виконання) є характеристики, які задають постачальники завдань, та параметри обчислювальних ресурсів (кількість процесорів, обсяг ОЗУ, швидкодії процесора, обсяг системи збереження даних, пропускна здатність каналу зв'язку, тип операційної системи, архітектура процесорів).

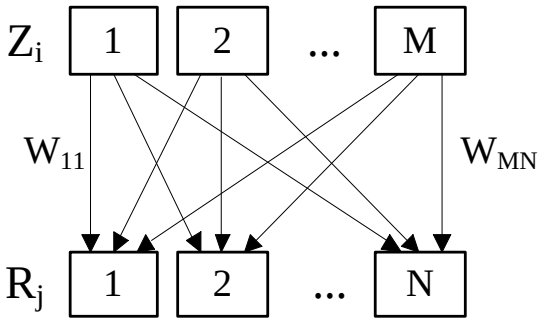

Рисунок 2.1 – Граф задачі розподілу завдань

Сукупність параметрів може бути доповненою в залежності від вимог постачальників завдань та ресурсів. Це породжує велику кількість варіантів вирішення поставленої задачі розподілу. Таким чином, зростає складність пошуку оптимального рішення.

Рішення задачі розподілу завдань на обчислювальні ресурси GRID-системи зводиться до наступної послідовності дій:

- за допомогою методу розподілу (реалізований у брокері) потік завдань Z_i розподіляється на множину обчислювальних ресурсів R_j враховуючи всі вимоги постачальників завдань;

- якщо завдання Z_i не може бути розподілено (в системі немає обчислювальних ресурсів, які задовольняють виконанню завдань), система приймає рішення відмовити в обслуговуванні;

- якщо завдання Z_i може бути розподілено, системою надається обчислювальний ресурс R_j для його виконання протягом часу, який задається постачальником завдання;

- якщо обчислювальний ресурс вивільняється раніше наданого часу, то відбувається перерозподіл ресурсів;

- у всіх інших випадках перерозподіл ресурсів відбувається за стандартною схемою.

2.2.2 Дослідження технологій розподілу задач завдання на обчислювальних ресурсах

Існують випадки, коли завдання не може бути виконано на одному обчислювальному ресурсі. У такому разі воно має бути розподіленим на множину обчислювальних ресурсів, які задовольняють вимогам постачальників завдань.

В сучасних схемах організації розподілу завдань в GRID-системах не враховується характер пакету завдання (зв'язність задач в завданні). Постачальник завдання може вказувати ступінь зв'язності задач в завданні, що дозволяє скоротити час передачі проміжних результатів між задачами завдання. Це, в свою чергу, скорочує час перебування завдання в GRID-системі та збільшує відсоток задіяності обчислювальних ресурсів.

Задачі в завданні можуть мати зв'язність трьох типів: незв'язані, слабозв'язані та сильнозв'язані.

Якщо задачі в завданні незв'язані між собою (ca=0), то при підборі обчислювальних ресурсів, необхідних для виконання завдання, виконуються наступні дії: дані про обчислювальні ресурси, які прийшли на вхід модуля аналізу зв'язності у вигляді відповідної множини, із модулю згортки кортежу залишається незмінною за критерієм цілісності, а потім подається на вхід брокеру для розподілу у відповідності від раніше обраного методу розподілу завдань.

Якщо задачі в завданні слабозв'язні ($ca \leq 0.3$), то при підборі обчислювальних ресурсів, необхідних для виконання завдання, виконуються наступні дії: враховуючі, що множина ресурсів з модулю згортки кортежу залишається незмінною, ресурсам, які знаходяться в одному місці (кластері) віддається перевага (для них встановлюється мітка). Тобто, якщо при розподілі завдань, вищевказані ресурси – вільні, то планувальник для виконання завдання задіє помічені ресурси.

Якщо задачі в завданні сильнозв'язні ($0,3 < ca \leq 1$), то з множини ресурсів будуть видалені лише ті, які були підібрані для виконання завдання із різних ресурсів (наприклад, $R_1 + R_5$). Це пояснюється тим, що розподіл таких завдань необхідно виконувати тільки на одному обчислювальному ресурсі. Якщо на момент

запуску завдання ресурс, який би задовольнив вимогам постачальника – відсутній в GRID-системі, то виконання завдання унеможливлюється та повертається постачальнику завдання з метою зміни характеристик обчислювальних ресурсів, необхідних для виконання завдання, або для перекваліфікації загальної задачі у площині розпаралелювання, власно, задач із завдання. Якщо обчислювальний ресурс, який задовольняє вимогам постачальника, присутній, але на даний момент зайнятий виконанням іншого завдання, то виконання поточного завдання буде призупинено до звільнення ресурсу; якщо час очікування перевищує часовий поріг, встановлений адміністратором системи, то брокер, проаналізувавши попередні запуски однотипних завдань, приймає рішення щодо реалізації механізму розпаралелювання задач із завдання на вільних обчислювальних ресурсах.

2.3 Дослідження існуючих планувальників

Важливе місце в GRID-системах займають планувальники завдань, які будують розклад на підставі інформації, що надходить від постачальників завдань та ресурсів. На даний час відомо більше двадцяти планувальників (таблиця 2.1): Portable Batch System (PBS) [115], Sun Grid Engine (SGE) [139], TORQUE [116], Condor [112], Load-Lever [140], MAUI Scheduler [119], Load Sharing Facility (LSF) [141] та ін.

Існуюча більшість планувальників (таблиця 2.1) характеризується такими недоліками:

- закритість вихідного коду та платне ліцензування;
- жорстка прив’язка у відношенні до конкретних постачальників завдань;
- низький показник живучості (неможливість відновлення роботи після збоїв у роботі);
- відсутність функції міграції завдань між обчислювальними ресурсами.

В сучасних GRID-системах використовується пріоритетне планування, яке має два різновиди класів.

Таблиця 2.1 – Порівняльна характеристика існуючих планувальників

	OpenPBS	PBS Pro	Sun Grid Engine (SGE)	TORQUE	Condor	Load-Leverer
Вихідний код	–	–	+	+	+	–
Безкоштовне використання в некомерційних цілях	+	–	+	+	+	–
Підтримка UNIX-платформ	+	+	+	+	+	+
Підтримка Windows-платформ	–	+	+	–	+	–
Підтримка MPI	+	+	+	+	+	+
Відновлення завдань після збою	–	+	+	–	+	+
Міграція завдань	–	+	+	–	+	+
Підтримка Maui	+	+	+	+	–	+
Виробник	Altair Engineering, Inc	Altair Engineering, Inc	Sun Micro-systems, Inc	Cluster Resources Inc.	HT Condor	IBM Corp
Дата і номер останньої версії	червень 2001, v.2.3.16	червень 2015, v.13.0	жовтень 2012, v.6.2.48	серпень 2016, v.6.0.2	вересень 2016, v.8.4.9	грудень 2012, v.5.1.0

До першого класу відносяться методи, які орієнтовані на виділення обчислювальних ресурсів для завдань з черги за поточним станом обчислювальних ресурсів. Використання даних методів має такі недоліки: накопичення обчислювальних ресурсів, які необхідні для запуску завдання, призводить до їх тривалої незадіянності, так як задачі, що входять до завдання, повинні бути запущені одночасно; внаслідок того, що відсутня інформація про час вивільнення обчислювальних ресурсів, унеможливлюється застосовування функції попереднього резервування. При вирішенні задач, які не вимагають розпаралелювання (слабозв'язні), може виникати ситуація, коли високопріоритетні завдання захоплюють ресурси на тривалий час, а це призводить до того, що низькопріоритетні завдання можуть не отримати ресурси взагалі. Тобто кожен раз, коли необхідний ресурс (для низькопріоритетного завдання) звільняється – до GRID-системи надходить високопріоритетне завдання та займає його для виконання. У іншій ситуації, коли для виконання високопріоритетного завдання немає необхідного ресурсу, воно знаходиться у черзі завдань та очікує вивільнення обчислювального ресурсу (ресурсів). Тепер, хоча для низькопріоритетного завдання з'являється ресурс, який задовольняє його вимогам, – це завдання не може зайняти ресурс через те, що в черзі завдань перед ним знаходиться високопріоритетне завдання.

До другого класу відносяться методи, які використовують для розподілу завдань заздалегідь сформований розклад розподілу. Класичним представником цього класу є метод BackFill [100], який дозволяє реалізувати функцію попереднього резервування ресурсів, що зменшує відсоток простою обчислювальних ресурсів.

Підсумовуючи, можна виділити головний недолік – жорстка прив'язаність до вирішення конкретного класу задач (вузькоспеціалізованість). У зв'язку з цим виникають додаткові складнощі при розподілі таких завдань, тому що на даний момент відсутній універсальний метод розподілу завдань.

Проведений аналіз дозволяє зробити висновок, що:

- існуючі планувальники не пропонують користувачу найбільш ефективного способу розподілу завдань. Тобто за їх допомогою користувач може лише скорис-

татися одним простим методом розподілу, який виконується за умови, що всі завдання та обчислювальні ресурси GRID-системи відповідають шаблону заданого опису;

- в даний час не існує універсального планувальника завдань через те, що завдання, які надходять до GRID-системи є різнорідними, а при їх розподілі необхідно враховувати ряд додаткових параметрів.

Отже, необхідно розробити сукупність заходів щодо створення нового підходу до розподілу завдань та забезпечуючої інформаційної технології [38]. Ці дії мають враховувати та усувати недоліки в існуючих рішеннях задачі ефективного розподілу завдань, які були виявлені в ході порівняльного аналізу (таблиця 2.1).

2.4 Дослідження різновидів методу Backfill

Метод зворотнього заповнення Backfill використовує інформацію про поточний стан черги та стан окремо взятих обчислювальних ресурсів. Всі незадіяні ресурси об’єднуються в області (вікна), які сортуються по ширині. З черги завдань обирається завдання, яке задовольняє ширину вікна, та запускається на обчислювальному ресурсі. Зарезервовані завдання мають найбільший пріоритет в черзі завдань відносно обраного вікна.

Існує два різновиди методу Backfill: агресивне та консервативне резервування [100]. Консервативне резервування гарантує для високопріоритетних завдань виділення обчислювальних ресурсів за мінімально можливий час і дозволяє низькопріоритетним завданням використовувати незадіяні ресурси, що підвищує коефіцієнт загального завантаження GRID-системи.

При використанні агресивного резервування обчислювальні ресурси з пулу GRID-системи виділяються тільки для високопріоритетних завдань. Недолік агресивного резервування полягає у тому, що не враховуються завдання, які не брали участь в процесі резервування ресурсів взагалі.

Вхідними даними для консервативного та агресивного резервування методу Backfill виступають:

- множина завдань в черзі із зазначенням вимог до обчислювальних ресурсів і інтервал часу для їх виконання $\{Z_i, i = 1, 2, ..., M\}$;

- кількість завдань, які виконуються у поточному часі, із зазначенням задіяних обчислювальних ресурсів та очікуваним часом завершення виконання $z_i^{run} \in \{Z_i, i = 1, 2, ..., M\}$;

- кількість незадіяних обчислювальних ресурсів $r_j^{free} \in \{R_j, j = 1, 2, ..., N\}$.

На першому кроці консервативного резервування відбувається генерація профілю використання процесорів (CPU) для виконувальних завдань у поточному часі. Для цього необхідно:

- впорядкувати список виконувальних завдань у поточному часі z_i^{run}, відповідно до їх часу закінчення rt, $\{Z \mid sort(rt_i, \forall i = 1...M)\}$;

- час виконання завдань зі списку t розбити на періоди, які відповідають часу закінчення завдань τ. Для кожного періоду записується кількість CPU, які використовуються в цьому періоді, $t \subset \tau, \forall \tau := pr \mid R_j, \forall j = 1...N$. Після чого формується профіль використання $Prof \subset \{\tau\}$.

На другому кроці відбувається розподіл завдань з черги, для чого:

- черга завдань обходиться в порядку їх надходження;

- для кожного завдання, аналізується профіль $Prof$ і знаходиться перша точка, де достатня кількість CPU для виконання завдання $pr_i \le pr_{prof}$. В результаті цих дій отримується точка прив'язки. Починаючи з цієї точки, триває аналіз профілю $Prof$ до точки закінчення завдання та визначається, чи будуть доступні CPU протягом необхідного часу:

а) якщо «так», то відбувається оновлення профілю з урахуванням того що ці CPU будуть зайняті;

б) якщо «ні», то виконується аналіз наступної точки прив'язки та повторяється перевірка доступності CPU;

- перше завдання, яке задовольняє вимогам, може бути відправлено на виконання з близько нульовою затримкою.

На першому кроці агресивного резервування відбувається підбір вікна та додаткових обчислювальних ресурсів. Для цього необхідно:

- впорядкувати список виконувальних завдань у поточному часі z_i^{run}, відповідно до їх часу закінчення rt, $\{Z \mid sort(rt_i, \forall i = 1...M)\}$;

- сформувати достатню кількість обчислювальних ресурсів для виконання перщого завдання з черги шляхом знаходження обчислювальним ресурсам, що задовольняють вимогам завдання;

- виділити вікно з інтервалу часу, який було знайдено для виконання завдання на обчислювальних ресурсах;

- якщо дане вікно включає надлишок обчислювальних ресурсів, чим необхідно для виконання завдання, то різницеві ресурси будуть додатковими.

На другому кроці відбувається підбір завдання для розподілу, для цього:

- черга завдань обходиться в порядку їх надходження;

- для кожного завдання, перевіряється одна з таких умов:

а) для виконання завдання потрібно менше обчислювальних ресурсів, ніж є на даний момент у вікні, а час закінчення знаходиться в межах вікна;

б) для виконання завдання потрібна загальна кількість вільних і додаткових обчислювальних ресурсів;

- перше знайдене завдання відправляється на виконання.

2.5 Дослідження існуючих способів підрахунку мережного трафіку в GRID-системах

Мережний трафік – це обсяг інформації, яка передається через комп’ютерну мережу за певний період часу. Кількість трафіку може бути виміряна в пакетах, бітах, байтах і їх похідних [142]. Керуюча інформація пакетів даних містить адреси відправника та одержувача, коди виявлення помилок та інформацію про черговості. Сумарна довжина блоку керуючої інформації становить 2-10% від загальної довжини пакета.

Часто виникають ситуації, коли для вирішення тієї чи іншої задачі необхід-

ний збір і подальший аналіз різноманітної статистики в діючих мережах (швидкість, обсяги переданих даних та ін.). Для збору такої статистики можна використовувати різноманітне ПЗ, побудоване на відомих алгоритмах [143]. Часто виникає необхідність введення обмежень на збір даних, які необхідні для аналізу.

Одним із важливих параметрів є загальний час виконання завдання, яке визначається за формулою (2.3):

$$T_з = \sum_{i=1}^{n} T_i , \tag{2.3}$$

де T_i – час виконання завдання;

n – кількість задач у завданні.

Час виконання задачі складається з часу, який необхідний для обчислень (t_i^o) і, власне, часу, який необхідний для передачі інформації ($t_i^п$). Дана закономірність представлена у формулі (2.4):

$$T_i = t_i^o + t_i^п . \tag{2.4}$$

Час, який потрібний для обчислень (t_i^o) визначається як частка від кількості елементарних операцій в завданні ($k_i^{оп}$) і продуктивність процесора (p_i), яка має розмірність, в свою чергу, – операції за секунду (2.5).

$$t_i^o = \frac{k_i^{оп}}{p_i}, \tag{2.5}$$

Час, який необхідний для передачі інформації ($t_i^п$) визначається як сума затримки часу передачі ($t_i^з$) і номального часу передачі інформації (t_i^{tr}) (2.6):

$$t_i^п = t_i^з + t_i^{tr} . \tag{2.6}$$

Час передачі даних визначається, як частка від ділення кількості переданих

в завданні біт ($k_i^{б}$) на ширину пропускної можливості каналу зв'язку (b_i) (2.7):

$$t_i^{tr} = \frac{k_i^{б}}{b_i}. \tag{2.7}$$

Таким чином, загальний час виконання завдання може бути представлено в наступному вигляді (2.8):

$$T_з = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \left(\frac{k_i^{оп}}{p_i} + t_i^{з} + \frac{k_i^{б}}{b_i} \right) \tag{2.8}$$

Вираз (2.8) є справедливим у тому випадку, якщо задачі, що входять до складу завдання виконуються послідовно на одному ресурсі. При розпаралелюванні завдань на обчислювальних ресурсах, вираз (2.4) приймає вигляд (2.9):

$$T_з = \max_{i=1}^{n} T_i = \max_{i=1}^{n} \left(\frac{k_i^{оп}}{p_i} + t_i^{з} + \frac{k_i^{б}}{b_i} \right) \tag{2.9}$$

Існують наступні інструментарії обліку та аналізу мережного трафіку [142]:

- з використанням SNMP-протоколу;
- з використанням служби WMI;
- установка агентів на віддалені комп'ютери;
- з використанням протоколу NetFlow;
- з використанням аналізаторів трафіку.

При використанні SNMP-протоколу не потрібна установка додаткового ПЗ на призначені для користувача обчислювальні станції. Програма обліку трафіку ставиться тільки на комп'ютер адміністратора (SNMP-менеджер). На віддалених машинах (SNMP-агентів) необхідно лише налаштування роботи служби SNMP. При роботі SNMP-протоколу використовується керуюча база даних MIB (Management Information Base). MIB – це набір змінних, що характеризують стан агента. Для того щоб проконтролювати роботу певного мережного пристрою не-

обхідно отримати доступ до його бази MIB і проаналізувати значення ряду змінних. Перевагою даного протоколу є можливість використання його на платформах ОС Windows та UNIX/Linux.

WMI (Windows Management Instrumentation) – це інструментарій управління операційної системи Windows, який використовується для централізованого управління і спостереження за роботою окремих частин комп'ютерної мережі. Облік трафіку за допомогою WMI не вимагає установки додаткових модулів на віддалені комп'ютери. WMI – це розширена та адаптована для платформи Windows реалізація стандарту WBEM. В основі даного стандарту лежить концепція створення універсального інтерфейсу для моніторингу та управління різними системами та компонентами розподіленого інформаційного середовища інформаційної системи. Для звернення до об'єктів WMI використовується специфічна мова запитів WQL, яка реалізуються за допомогою WSH-скриптів.

Якщо унеможливлюється використання SNMP-протоколів та служби WMI в комп'ютерній мережі, то є можливість аналізувати та вести облік трафіку за допомогою агентної системи контролю. Наприклад, такими агентами можуть бути програмні модулі системи моніторингу Zabbix. Вона орієнтована на відстеження статусів різноманітних сервісів мережного обладнання, комп'ютерної мережі та серверів. В системі Zabbix підтримується кілька видів моніторингу і в її архітектурі можна виділити наступні складові:

- Zabbix-сервер, що виконує періодичні запити на отримання даних, їх обробку та аналіз, а також запускає скрипти оповіщення;

- БД, які сформовані за допомогою мов MySQL, Jracle, PostgreSQL, SQLite;

- web-інтерфейс, який реалізований на мові PHP;

- Zabbix-агент, який встановлюється на віддалених об'єктах і запускається для отримання інформації та подальшої передачі її на Zabbix-сервер.

Облік мережного трафіку за допомогою протоколу NetFlow, був розроблений компанією Cisco і використовується для збору інформації про мережний трафік лише всередині комп'ютерної мережі. Принцип роботи протоколу NetFlow полягає в зборі статистики про передані пакети в мережі. Вся накопичена і про-

аналізована інформація зберігається в спеціальному буфері ОС моніторингової обчислювальної станції. Перевагою даного підходу є можливість вести облік мережного трафіку у великих інфраструктурних рішеннях зі складною та територіально-розподіленою інфокомунікаційною мережею. Недоліком є те, що використання даного підходу можливо лише у тому випадку, якщо використовується спеціалізоване дороге обладнання, яке підтримує протокол NetFlow.

Іншим напрямом для підрахунку мережного трафіку є аналізатори трафіку. Зібрані пакети у програмних модулях аналізуються і на підставі отриманих результатів відбувається діагностика та усунення виникаючих проблем у комп'ютерній мережі. Існуючі аналізатори трафіку дозволяють:

- виявляти закільцьований трафік в мережі, який збільшує мережне завантаження каналів зв'язку та усувати його;

- виявляти в мережі анамальне ПЗ, що вносить збурення в роботу мережі;

- здійснювати аналіз незашифрованого трафіку з метою верифікації паролів або іншої інформації щодо пакетного вмісту даних;

- виявляти і локалізувати помилку конфігурації або несправність мережі.

Використання в комплексі декількох напрямів для обліку трафіку дозволяє отримати повну картину про роботу комп'ютерної мережі та її окремих складових. Облік мережного трафіку окремо по кожному з протоколів, аналіз отриманих результатів, відображення інформації у вигляді таблиць, графіків дозволяє виявити місця мережі, які характеризуються високим рівнем циркуляції трафіку та запобігти виникненню можливих факторів, які можуть спричинити непрацездатність окремих вузлів або всієї мережі.

2.6 Дослідження існуючих способів підрахунку пропускної здатності та затримки передачі пакетів даних у каналі зв'язку

Існують різні типи завдань, які можуть бути виконані в GRID-системі. Наприклад, завдання рендерингу відеопотоку, яке характеризується великим обсягом вхідних і вихідних даних. Під рендерингом відеопотоку розуміється процес

отримання відеоматеріалу на основі побудованої раніше моделі за допомогою спеціалізованої комп'ютерної програми. Модель будується на підставі переліку заданих раніше об'єктів: геометричне розташування, освітлення, кут огляду, ряд текстур тощо. Для виконання такого завдання GRID-системі необхідно передати вхідні дані (сцену) до всіх підібраних обчислювальних ресурсів. Однак в цьому випадку виникає ситуація, коли об'єм моделі є занадто ємним і передача на обчислювальні ресурси може займати тривалий час, який відкладе початок виконання всіх завдань з черги.

У такому випадку слід аналізувати канали зв'язку, які використовуються для передачі даних, і відбирати лише ті, які дозволять скоротити час передачі вхідних та вихідних даних. Для аналізу каналів зв'язку слід звернути увагу на два параметри: пропускну здатність каналу (bandwidth) та затримку передачі пакетів (delay).

Пропускна здатність визначає ширину каналу зв'язку, яка вимірюється в bit/s (bps), kbit/s (kbps), mbit/s (mbps) і показує, який максимальний обсяг даних може бути переданий по каналу за одиницю часу.

На сьогодні існує ряд інструментів, які дозволяють вимірювати пропускну здатність каналу зв'язку. В межах локальної мережі – це інструменти netperf та ttcp. Для глобальної мережі Інтернет існують «тести швидкості», які доступні для інтерактивного використання на загальнодоступних web-сайтах. Скориставшись даними тестами можна зробити висновок, що пропускна здатність постійно варіюється і виміряти її точно дуже складно, тобто це динамічна величина. Це пов'язано з тим, що мережна архітектура включає кількість рівнів вузлів та ПЗ, а також має ряд часових характеристик проходження інформації через них.

Затримка передачі пакета даних в каналі зв'язку, також вносить свої корективи в роботу мережі. Даний параметр може бути співставленно з затримками в процесі передачі даних в мережі. Самі дані розрізняються також за видами (еластичні та нееластичні). Час передачі пакету між обчислювальними станціями розраховується по формулі (2.10).

$$T_t = \frac{ps}{bw}, \qquad (2.10)$$

де ps (packet size) – розмір пакету даних;

bw (bandwidth) – пропускна здатність каналу зв’язку.

Наприклад, час передачі пакету даних розміром 64 байти в каналі з пропускною здатністю 64 kbit/s складає: ps=64*8=512 bit, Tt=512/64000=0,008 с.

Наведений приклад показує, що затримка і пропускна здатність взаємопов’язані між собою. Якщо теоретична величина пропускної здатності фіксована, то практична (ефективна пропускна здатність) змінюється на інтервалі часу, тому що на неї впливає загальний фон циркуляції даних в мережі. Занадто велика величина затримки за занадто короткий проміжок часу породжує таке явище як затор. Він перешкоджає повному заповненню каналу зв’язку даними, що значно знижує ефективність пропускної здатності каналу за рахунок повторних передач. Справедливо і зворотне твердження: якщо реальне завантаження каналу зв’язку наближається до 100%, а користувачі комп’ютерної мережі продовжують пеоредавати потоки даних, можна помітити суттеве зростання часу на передачу пакетів даних. Це відбувається в результаті того, що пакет займає місце в черзі, замість того, щоб потрапити в канал передачі, а як наслідок відбувається переповнення, власне, самої черги.

Для вимірювання затримки передачі пакету існують такі мережні інструменти, як ping та traceroute. Вони дозволяють виміряти затримку за допомогою визначення часу, який потрібен для того, щоб певний мережний пакет пройшов шлях від початкової точки до місця призначення і назад через всі маршрутизатори мережі.

Traceroute (Tracert) – це мережна команда, яка дозволяє оцінити довжину шляху проходження пакету від заданої точки до даного ресурсу. Дана інформація показує кількість проміжних маршрутизаторів через які проходить пакет і дозволяє оцінити працездатність каналу зв’язку до необхідного вузла. При занадто довгих маршрутах (більше 25 кроків) зростає ймовірність того, що один з проміжних маршрутизаторів на шляху проходження даних в мережі буде перевантажений,

внаслідок чого швидкість передачі даних суттєво знизиться. Якщо при виконанні команди ping пакети не доходять до потрібного вузла, то за допомогою команди tracert можна дізнатися, на якому саме маршрутизаторі відбувається розрив маршруту передачі даних.

Існують також інші інструменти для виявлення нестабільних мережних вузлів, сегментів мережі з низкою пропускною здатністю, інформація про які зберігається в БД. Якщо протягом певного часу ситуація не змінюється, то можуть прийматися рішення щодо надбудови віртуалізованих рішень для створення нових маршрутив передачі даних.

2.7 Висновки по розділу

В ході дослідженні моделі розподілу завдань на обчислювальних ресурсах та аналізу відомих рішень було встановлено що:

- не існує універсального планувальника завдань, який дозволив би розподіляти завдання на ресурси таким чином, щоб мінімізувати простої обчислювальних ресурсів в GRID-системі;

- при розподілі завдань на обчислювальних ресурсах планувальники не враховують характер пакету завдання (зв’язність задач в завданні), що дозволяє зменшити час перебування завдання в черзі завдань;

- найбільш використовуваний при розподілі завдань в GRID-системах є метод Backfill, який запобігає дефрагментації обчислювальних ресурсів, однак жодним чином не враховує час, який витрачається на передачу вхідних та вихідних даних завдання (даний час безпосередньо залежить від пропускної здатності каналу зв’язку та часу затримки передачі пакету по каналу зв’язку);

- при використанні методу Backfill відсутня можливість здійснювати вплив на процес розподілу завдань, із-за реалізації у ньому принципу «перший відповідний».

На підставі зробленого аналізу можна зробити висновок, що необхідно розробити сукупність заходів щодо створення нового підходу до розподілу завдань

та відповідної інформаційної технології, яка буде реалізувати його.

Для реалізації даного підходу необхідно:

- ввести класифікацію завдань за показником переданого обсягу інформації, що дозволить ефективно використовувати обчислювальні ресурсів GRID-системи за рахунок скорочення відсотку їх простою;

- реалізувати в середовищі моделювання GRASS множину методів розподілу, які дозволять провести ряд експериментів для різних пулів завдань з метою підтвердження припущення про вплив класів завдань на час розподілу вхідного пула;

- розробити метод, який спрямований на пошук розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів на підставі сукупності задіяних критеріїв. Це дозволить сформувати пропозиції для реальної GRID-системи щодо розподілу завдань, який буде враховувати не тільки скорочення часу виконання пулу завдань, а й на ефективно максимальну завантаженість обчислювальних ресурсів GRID-системи.

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [38,58,60,72,96-100,112,115,116,119,139-143].

3 ПОБУДОВА МОДЕЛІ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПОДІЛУ ЗАВДАНЬ НА ОБЧИСЛЮВАЛЬНИХ РЕСУРСАХ

Інформаційна технологія (ІТ) – це практична діяльність і прикладна наука, що мають справу з даними та інформацією [1]. Суть інформаційної технології полягає в зборі, обробці, забезпеченні безпеки, передачі, взаємообміні, управлінні, організації, зберіганні та відновленні даних та інформації. Отже, інформаційна технологія – це сукупність методів, технічних і програмних засобів, які об'єднуються для збору, зберігання, передачі, обробки та відображення інформації. Основною метою інформаційної технології є отримання інформації для аналізу її експертом (або експертної системою), який на її основі приймає рішення щодо виконання певних дій. Інформаційні технології використовуються для зниження трудоємності обчислювальних процесів на інформаційних ресурсах, а також для підвищення їх надійності.

3.1 Розробка моделі розподілу завдань в гетерогенній GRID-системі

На підставі проведеного аналізу в розділі 2, розширимо модель розподілу завдань в GRID-системі за рахунок додання множини методів розподілу (3.1):

$$Q_k = \{mn_k, lp_k\}, \ \forall k = 1..K, \tag{3.1}$$

де mn – метод розподілу завдань на обчислювальних ресурсах;

lp – перелік вхідних параметрів, що враховуються при розподілі.

Тепер модель розподілу завдань в GRID-системі будується на підставі трьох множин: обчислювальних ресурсів R, завдань Z та методів розподілу Q: $G = \{R, Z, Q\}$.

Також були розширені кортежі 2.1 та 2.2 за рахунок параметрів, які дозволять усунути виявлені та проаналізовані у другому розділі недоліки та підвищити ефективність використання обчислювальних ресурсів GRID-системи (3.2-3.3):

$$Z_i = \left\{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z, ca_i^z, rt_i^z\right\}, \ \forall i = 1..M, \quad (3.2)$$

$$R_j = \left\{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r, bw_j^r, d_j^r\right\}, \ \forall j = 1..N, \quad (3.3)$$

де ca_i (coefficient of association) – коефіцієнт зв'язності задач в завданні;

rt_i (run time) – час виконання завдання (час використання ресурсу);

bw_j (bandwidth) – пропускна здатність каналу зв'язку (від брокера до ресурсу);

d (delay) – затримка часу передачі пакету даних в каналі зв'язку.

Недоліком GRID-систем є використання єдиного брокера, який орієнтований на певний клас задач і при розподілі завдань використовує єдиний алгоритм розподілу [19,27,28].

При проведенні експериментів було використано імітаційне середовище моделювання GRASS [30,40]. Встановлено, що воно дозволяє проводити експеримент тільки з одним методом розподілу (FIFO). Для збільшення продуктивності середовища GRASS був запропонований модуль розподілу завдань (Algorithm Loader) [21], який підтримує можливість проведення серії експериментів. Цей модуль оперує декількома методами розподілу: First-Come First-Served (FCFS), Last In First Out (LIFO), Highest Priority First (HPF), метод лінійного програмування (Simplex), метод розподілу по звільнився ресурсу (Smart), метод зворотнього заповнення (Backfill), а також метод розподілу завдань на обчислювальні ресурси з урахуванням мережного трафіку (Backfill_mod). Впровадження модуля розподілу завдань в середовище GRASS дозволяє в рамках одного експерименту переглянути динаміку зміни ефективності системи та визначити її вузькі місця.

3.2 Побудова системи показників на підставі узагальненого критерію оцінки завдання

Розподіл завдань на ресурсах є складною задачею із-за наявної необхідності врахування впливу багатьох параметрів на якість розподілу. Тому задачу розподілу спрощують за рахунок скорочення кількість критеріїв розподілу, залишаючи

тільки найбільш впливові на якість розподілу, або вводять поняття пріоритету завдання.

З метою поліпшення якості роботи планувальника завдань, пропонується ввести узагальнений критерій оцінки завдання щодо можливості його використання на ресурсах обчислювальної системи, що дозволяє враховувати всі необхідні параметри ресурсів і завдань [34].

Будь-яке завдання, яке надходить до GRID-системи, описується за допомогою кортежу (3.2), який включає в себе весь перелік вимог, необхідних для запуску і коректної роботи. Для планувальника важливими параметрами завдання є вимоги до ресурсів, що описані кортежем (3.3). З'являється задача визначення головного параметру, тобто по якому з параметрів здійснювати оптимізацію.

В даний час існує ряд задач, які необхідно аналізувати по сукупності параметрів, що створює труднощі для процесу розподілу. У таких випадках раціонально використовувати методи розв'язання багатокритеріальних задач оптимізації. Застосування цих методів має на увазі об'єднання ряду параметрів в єдиний критерій. Об'єднання (згортка) параметрів проводиться евристичними методами, тому з математичної точки зору, не існує рішення таких задач, яке б задовольняло всім вимогам. Найчастіше для вирішення багатокритеріальних задач оптимізації використовують узагальнений (інтегральний) критерій [76,144]. Його суть полягає в тому, що часткові критерії $F_i(X), i = \overline{1,n}$ об'єднуються в один інтегральний критерій $F(X) = \Phi(F_1(X), F_2(X), ..., F_n(X))$. Потім знаходиться максимум або мінімум даного критерію. Залежно від того, яким чином часткові критерії об'єднуються в узагальнений, розрізняють наступні їх види: адитивний, мультиплікативний, максимінний (мінімаксний). Перевагою використання запропонованого підходу є те, що завжди вдається визначити оптимальний варіант рішення поставленої задачі. Серед їх недоліків варто виділити:

- труднощі (суб'єктивізм) у визначенні вагових коефіцієнтів;

- адитивний коефіцієнт не зав'язаний на об'єктній ролі часткових коефіцієнтів і тому його можна представити як формальний математичний прийом;

- адитивним коефіцієнтом визначається взаємна компенсація часткових кри-

теріїв, тобто зменшення одного з них може бути компенсовано збільшенням іншого.

Узагальнений адитивний критерій знаходится шляхом додавання нормованих значень часткових критеріїв [144] (3.4):

$$F = \sum_{l=1}^{n} C_l \frac{F_l}{F_l^0} = \sum_{l=1}^{n} C_l f_l, \qquad (3.4)$$

де n – кількість поєднуваних часткових критеріїв;

C_l – ваговий коефіцієнт l-го часткового критерію, $\sum_{l=1}^{n} C_l = 1$;

F_l – числове значення l-го часткового критерію;

F_l^0 – l-й нормуючий дільник;

f_l – нормоване значення l-го часткового критерію.

Часткові критерії часто мають різну фізичну природу та різну розмірність, отже, складати їх некоректно. Тому числові значення окремих критеріїв діляться на деякі нормуючі дільники. Розмірності самих часткових критеріїв і відповідних нормуючих дільників однакові, отже, узагальнений адитивний критерій є безрозмірною величиною.

Аналогічним чином отримують узагальнений мультиплікативний критерій, в якому замість складання виконують функцію перемноження часткових критеріїв. В якості часткових критеріїв можуть виступати різні параметри завдань та обчислювальних ресурсів: P_i^Z – значення і-го параметру завдання, P_i^R – значення і-го параметру обчислювального ресурсу.

Параметри кортежу завдань (3.2), якими оперує GRID-система, можна розділити на якісні та кількісні. Якісні параметри можуть бути представлені у вигляді множини дискретних значень:

- P_{ar} (architecture) – архітектура процесора:

$P_{ar} \in \{0 – \text{одноядерна}, 1 – \text{багатоядерна}\}$;

- P_{os} (operating system) – операційна система:

$P_{os} \in \{0 - \text{Windows XP}, 1 - \text{Windows 7}, 2 - \text{Linux}, 3 - \text{FreeBSD}\}$.

Узагальненим критерієм для якісних параметрів зручно прийняти мультиплікативний критерій виду (3.5), де P_i визначається виразом (3.6):

$$F_{quality} = \prod_{i=1}^{n} P_i, \tag{3.5}$$

$$P_i = \begin{cases} 1, \text{ якщо } P_i^Z = P_i^R \\ 0, \text{ якщо } P_i^Z \neq P_i^R \end{cases}. \tag{3.6}$$

$F_{quality} \in \{0,1\}$ при $F_{quality} = 1$ – завдання задовольняє якісними критеріями ресурсу.

Кількісні параметри – це фізичні характеристики, які для завдань є необхідними, а для ресурсів – певними (заданими). Приклади кількісних параметрів:

- P_{pc} (processor count) – кількість процесорів (ядер);
- P_{ps} (processor speed) – швидкодія процесору;
- P_{ms} (memory size) – обсяг оперативної пам’яті;
- P_{dc} (disk capacity) – обсяг жорсткого диску;
- P_{bw} (bandwidth) – необхідна пропускна здатність каналу зв’язку для організації обміну між елементами GRID-системи при виконанні завдань;
- P_d (delay) – сумарна затримка часу передачі пакету.

Узагальнений критерій для кількісних параметрів можна представити у вигляді адитивного критерію (3.7):

$$F_{quantity} = \begin{cases} \sum_{i=1}^{n} C_i P_i, \text{ якщо } P_i \leq 1 \\ 0, \quad \text{якщо } P_i > 1 \end{cases}, \tag{3.7}$$

де P_i – нормоване значення і-го параметру, $P_i = \dfrac{P_i^Z}{P_i^R}$.

Підсумковим узагальненим критерієм оцінки завдання, що враховує якісні та кількісні параметри, є вираз (3.8):

$$F = Fquality \cdot Fquantity. \quad (3.8)$$

Діапазон значень узагальненого критерію лежить в межах [0; 1]: якщо F=0 – це завдання не може бути виконане на даному обчислювальному ресурсі, якщо F=1 – завдання повністю використовує всі можливості ресурсу.

Узагальнений критерій оцінки завдання допомагає відібрати множину ресурсів, на яких може бути виконано конкретне завдання, а також показує, яку частину ресурсу займає завдання в процесі виконання.

Розглянемо використання узагальненого критерію оцінки завдання на прикладі. Нехай є загальна кількість завдань, що вимагають виконання M. Тоді F_m – це значення узагальненого критерію m-го завдання, де $m = \overline{1, M}$.

Якщо ресурс має незадіяні потужності, то на ньому можна виконувати декілько завдань одночасно. Наприклад, якщо у одного завдання значення узагальненого критерію $F_1 = 0.5$, а в іншого – $F_2 = 0.34$, то ці завдання можна розподілити для одночасного виконання на обчислювальному ресурсі, тому що F = F1 + F2 = 0.84, а 0.84 <1.

Нехай N – це кількість доступних для розподілу ресурсів, тоді завдання, яке найменш завантажує n-й ресурс ($n = \overline{1, N}$) буде мати мінімальне значення узагальненого критерію (3.9):

$$F^n_{m_{min}} = \min(F^n_m), \quad (3.9)$$

де F^n_m – значення узагальненого критерію оцінки m-го завдання для n-го ресурсу.

Таким чином, можна використовувати узагальнений критерій оцінки завдання для управління процесом розподілу завдань на обчислювальних ресурсах GRID-системи.

Додаткову складність вносить процес визначення значень вагових коефіцієнтів часткових критеріїв. Можна передати виконання цього процесу постачальнику завдань. Це пояснюється тим, що він має відомості про важливість тих чи інших часткових критеріїв, що характеризують це завдання. Також процедуру ви-

значення вагових коефіцієнтів можна покласти на системного адміністратора GRID-системи. Є можливим спільного визначення вагових коефіцієнтів обома учасниками процесу. У будь-якому випадку, розробка процедури визначення значень вагових коефіцієнтів часткових критеріїв, вимагає більш глибших досліджень для отримання оптимального результату.

3.3 Розробка модифікованого методу Backfill з консервативним резервуванням

При розподілі завдань на кожному кроці необхідно вирішувати дві задачі. Перша задача – це створення пулу ресурсів за вимогами постачальників завдання. Друга задача – це обрання ресурсів, які оптимізують час виконання заданого завдання, що призводить до підвищення продуктивності GRID-системи в цілому за рахунок скорочення простою обчислювальних ресурсів.

В даний час при розподілі ресурсів не існує відомих важелів впливу на хід розподілу конкретного завдання, тому що процес розподілу здійснюється за принципом «перший відповідний за вимогами». В даному випадку не враховуються пріоритети інших користувачів. Постачальники завдань не можуть точно оцінити час його виконання, отже, в системі часто виникають простої ресурсів. Також жодним чином не враховується зв’язність задач в завданні, за рахунок якої можна отримувати виграш за часом за рахунок оптимального розподілу обчислювальних ресурсів.

На сьогодні існує ряд завдань, які оперують великим обсягом вхідних даних (Big Date): наприклад, задача рендерінгу відеопотоку, задача обробки статистичної інформації, отриманої від радіо- та астрономічних спостережень (за рахунок перетворення Фур’є) тощо. Однак найчастіше при розподілі завдань на обчислювальних ресурсах не відбувається аналіз вхідних і вихідних даних, які використовуються для виконання завдання. Пропонується модифікований метод розподілу, який дозволяє врахувати сукупність різноманітних недоліків, що дозволить поліпшити план розподілу завдань.

На першому кроці необхідно проаналізувати вхідні вимоги і сформувати критерій оптимізації, який обчислюється на підставі вподабань постачальника завдань. Постачальник завдання вказує переваги за характеристиками обчислювального ресурсу, визначає вагові коефіцієнти для заданих параметрів. Цей критерій використовується для пошуку відповідних обчислювальних ресурсів і показує, яку частину обчислювального ресурсу займає завдання.

В даний час в розподілених системах поширений алгоритм Backfill, який має багато переваг і дозволяє запобігти дефрагментації обчислювальних ресурсів. Однак раціонально його використовувати або в зв'язці з додатковими евристиками розподілу, або його модифікацію. Так можна взяти до уваги роль мережного трафіку, тобто при розподілі завдань варто враховувати оцінку сумарної затримки часу передачі пакету в каналі зв'язку та його пропускну здатність. Ці два параметри дозволяють скоротити час на передачу результатів між окремо взятими ресурсами та зменшити мережний трафік за рахунок розвантаження каналів зв'язку [25,43].

Вхідними даними для запропонованого методу є:

- список завдань в черзі з зазначенням вимог до ресурсів $\{Z_i, i = 1, 2, .., M\}$;

- перелік виконуваних завдань із зазначенням задіяних обчислювальних ресурсів і очікуваним часом закінчення виконання завдання $z_i^{run} \in \{Z_i, i = 1, 2, .., M\}$;

- список ресурсів, які були підібрані для запуску кожного завдання $\{R_j^r, j = 1, 2, .., K\}$, згідно з узагальненим критерієм оцінки завдання.

Результатом роботи модифікованого методу Bacfill є отримання для кожного завдання Z_i «вікна для запуску».

На першому кроці резервування відбувається сортування списку запущених завдань z_i^{run} відповідно до очікуваного часу закінчення rt, $\{Z \mid sort(rt_i, \forall i = 1..M)\}$;

- час виконання завдань зі списку t розбивається на періоди, відповідно часу закінчення завдань τ. Для кожного періоду записується кількість обчислювальних ресурсів, які використовуються в цьому періоді, $t \subset \tau, \forall \tau := pr \mid R_j, \forall j = 1..N$. Як результат – отримуємо профіль використання $Prof \subset \{\tau\}$.

На другому кроці відбувається розподіл завдання, для чого:

- завдання з черги упорядковано відповідно до критерію зв'язності;

- на підставі множини $\left\{R_j^r,\ j=1,2,..,K\right\}$ формується список вікон, які можуть бути використані для запуску завдання – $W_{i\in 1..M}^{run}$;

- для кожного отриманого вікна формується перелік доступних обчислювальних ресурсів (CPU);

а) для кожної пари (CPU-брокер) визначається затримка часу передачі пакету даних по каналу (δ_n);

б) для кожної пари (CPU-брокер) визначається пропускна здатність каналу зв'язку (C_n);

- для запуску задач із завдання $a, z_a \in Z_i$ набирається необхідна кількість CPU, мінімізуючи один з параметрів:

а) якщо завдання має великий обсяг переданих вхідних даних і відповідно великий обсяг вихідних даних, то підбираються маршрути передачі даних з найбільшими значеннями пропускної здатності – C_n;

б) якщо для запуску завдання не потрібна передача великого обсягу даних, то підбираються маршрути передачі даних з найменшим значенням δ_n;

- по кожному отриманому вікну підсумовуються значення попарних відстаней і визначається найменше з них – $D_{min} = \min_{i\in 1..M} D_{W_i^{run}}$ – це і є найкращий результат розподілу.

Після того як завдання отримало вікно на виконання, відбувається оновлення профілю $\mathrm{Prof} \subset \{\tau\}$ з урахуванням того, що ці процесори будуть задіяні.

3.4 Розробка методу пошуку розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів

На сьогодні актуальною задачею в галузі GRID-систем є задача ефективного розподілу завдань за наявними обчислювальними ресурсами [93,134,145]. Вона

полягає в знаходженні такого набору ресурсів, при якому критерії оптимальності набудуть екстремального значення, тобто система буде функціонувати ефективно.

За допомогою запропонованого методу є можливість виконати моделювання роботи методів розподілу, реалізованих у модулі розподілу потоку завдань (Algorithm Loader) [21] в середовищі GRASS [30,40] і вибрати кращий з них, мінімізувавши ряд критеріїв (час виконання пулу завдань, час простою обчислювальних ресурсів, час перебуванням завдань в черзі) [24].

Ефективність вибору методу розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів GRID-системи визначається системою критеріїв і правил їх порівняння. Вибір плану розподілу заснований на аналізі, реалізованих в середовищі моделювання GRASS методів розподілу. Аналіз є побудованим на порівнянні значень запропонованих критеріїв по кожному з методів. Метод, який дає кращий результат за сукупністю всіх критеріїв, буде обраний результуючим для розглянутого пулу завдань.

Вказані критерії можуть бути використані для аналізу повноти завантаженості обчислювальних ресурсів. Кожен з представлених критеріїв не дає картини розподілу, однак використання множини критеріїв в сукупності, дозволить рівномірно завантажити обчислювальні ресурси GRID-системи та уникнути простоїв.

Розглянемо роботу запропонованого методу за допомогою імітаційної середовища моделювання GRASS. Для визначення плану розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простєм обчислювальних ресурсів необхідно провести ряд експериментів. Вхідні дані для експерементів формуються заздалегідь за певними правилами (завантажуються з реальної GRID-системи). В ході проведення експериментів відбувається збір статистичної інформації по кожному із заданих критеріїв для кожного методу розподілу. Для всіх методів розподілу використовуються одні й ті ж вхідні пули завдань і обчислювальних ресурсів. Запропонований метод пошуку плану розподілу включає в себе наступні етапи.

Етап 1: завантаження пулів завдань і ресурсів з реальних GRID-систем:

- $Z_i = \{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z, ca_i^z, rt_i^z\}, \ \forall i = 1..M;$

- $R_j = \{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r, dc_j^r, bw_j^r, d_j^r\}, \ \forall j = 1..N$.

Етап 2: вибір методів розподілу з множини $Q_k = \{mn_k, lp_k\}, \ \forall k = 1..K$.

Етап 3: запуск процесу моделювання в середовищі GRASS по кожному методу розподілу ($mn \in Q$):

а) отримуємо множину планів розподілу (Plank), $f : Q_k \rightarrow Plan_k, \ \forall k = 1..K$;

б) отримуємо множину часів виконання планів (T_k) по кожному методу розподілу ($mn \in Q$) $f : Plan_k \rightarrow T_k, \ \forall k = 1..K$;

в) визначаємо відсоток простою обчислювальних ресурсів і середній час очікування завдання в черзі по кожному із запропонованих методів розподілу.

Етап 4: аналіз log-файлів для всіх методів розподілу ($mn \in Q$) $f : Q_k \rightarrow T_k$, $\forall k = 1..K$.

Етап 5: обрання плану розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів GRID-системи на підставі прийнятих правил і обмежень: $plan = \min t_k, \ \forall k = 1..K$.

Методи розподілу завдань, реалізовані в середовищі GRASS, дозволяють здійснювати моделювання реальних ситуацій, що відбуваються в GRID-системі. Оскільки не існує універсального методу розподілу завдань на обчислювальні ресурси, який би давав кращий план розподілу, то необхідно аналізувати вхідні пули завдань і ресурсів в зв'язці.

Інколи виникає ситуація, коли найпростіші методи (LIFO, FIFO) дозволяють отримати виграш за часом виконання пулу завдань на тих обчислювальних ресурсах, які присутні в системі на даний момент часу. І наявність в системі єдиного планувальника (наприклад, Backfill), не завжди завантажує ресурси GRID-системи на повну потужність, в зв'язку з чим відсоток незадіяності обчислювальних ресурсів буває досить високий.

У таких ситуаціях слід робити акцент на класи завдань і будувати плани розподілу з урахуванням різних методів розподілу, що дозволяє підвищити ефективність використання обчислювальних ресурсів GRID-системи.

3.5 Розробка інформаційної технології розподілу завдань на обчислювальні ресурси

Згідно з визначеннями, які надані в ГОСТ ИСО/МЭК 2382-1-99 та ГОСТ ИСО/МЭК 27033-1-2011 інформаційна технологія – це сукупність методів, технічних і програмних засобів, які об'єднуються для збору, зберігання, передачі, обробки та відображення інформації. Основною метою інформаційної технології є отримання інформації для подальшого аналізу її експертом (або експертної системою), який приймає рішення щодо виконання певних дій.

На підставі запропонованої математичної моделі розподілу завдань в GRID-системі (підрозділ 3.1), розширенні кортежів завдань і обчислювальних ресурсів (підрозділ 3.1), модернізації методу розподілу завдань Backfill з консервативним резервуванням (підрозділ 3.3), розробки методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів (підрозділ 3.4) розроблено інформаційну технологію розподілу завдань (рисунок 3.1), яка була впроваджена в імітаційне середовище моделювання GRASS. Вона об'єднує процеси передачі, зберігання, збору даних, спостережень за швидкоплинними процесами та процес їх обробки з використанням запропонованих у дисертаційній роботі методів.

Вхідними даними для запропонованої технологи виступають кортежі завдань (3.2) та обчислювальних ресурсів GRID-системи (3.3). Ці кортежі формуються за обумовленими заздалегідь правилами та містять необхідні характеристики для формування розкладу виконання завдань на обчислювальних ресурсах. В ході роботи з вхідними даннямі відбувається згортання кортежу параметрів завдання (3.2), що дозволяє підвищити коефіцієнт використання обчислювальних ресурсів GRID-системи. В ході моделювання збирається інформація про розподіли по ряду запропонованих методів і заноситься до БД. До цієї інформації надається доступ програмному модулю системи моделювання GRASS, який реалізує метод пошуку розподілу з мінімальним часом виконання вхідного пулу завдань та мімнімальним простоєм обчислювальних ресурсів GRID-системи.

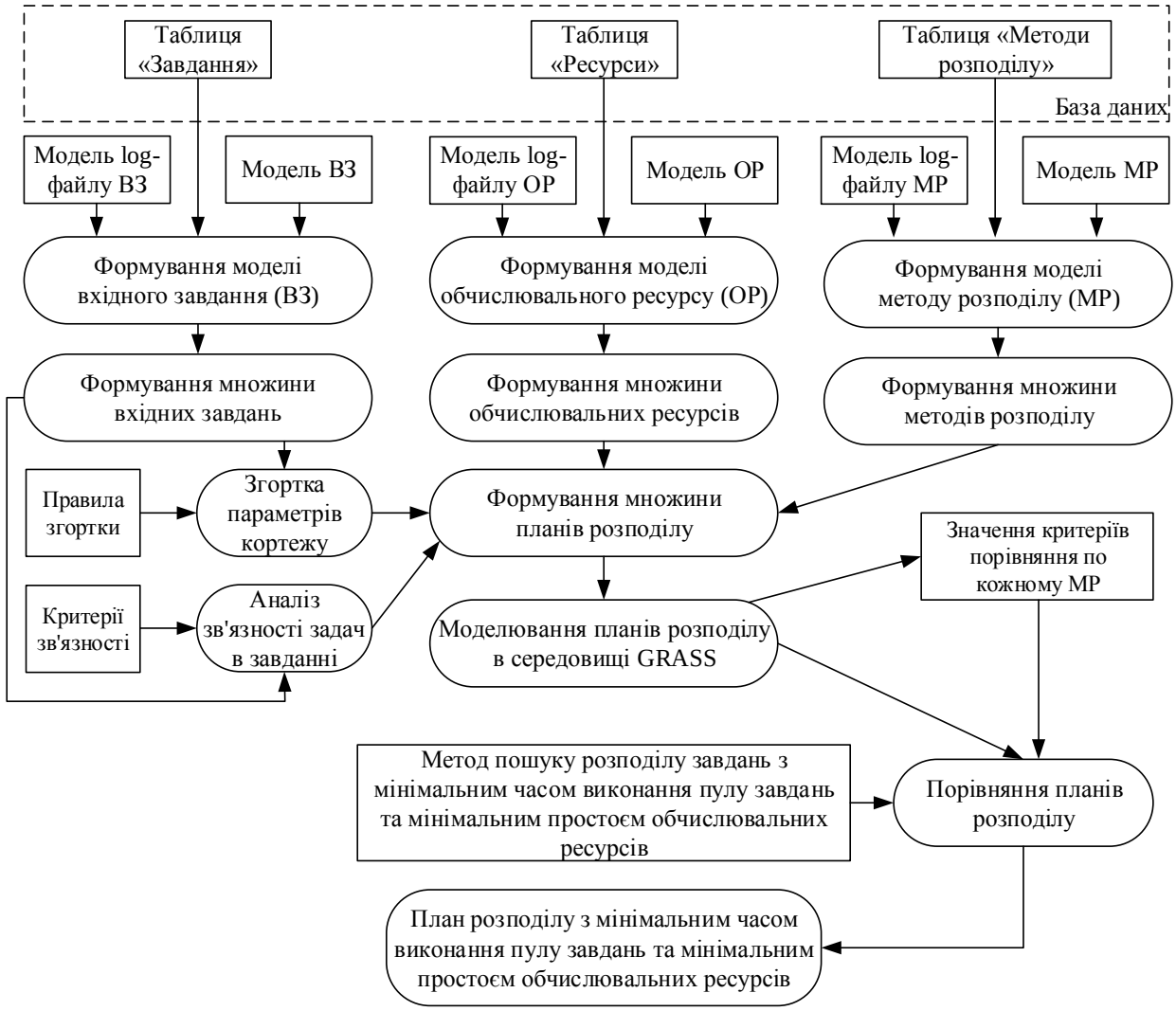

Рисунок 3.1 – Запропонована інформаційна технологія розподілу завдань на обчислювальні ресурси GRID-системи

Розглянемо етапи роботи запропонованої інформаційної технології розподілу завдань в гетерогенній GRID-системі, яка використовує імітаційне середовище моделювання GRASS (рисунок 3.2).

Етап 1: завдання параметрів експерименту. Для запуску експерименту налаштовується конфігураційний файл plugins.xml, який описує взаємозв’язки між іменами модулів в системі, назвами файлів і бібліотек [146]. У файлі plugins.xml можна задавати параметри, що передаються модулям при завантаженні для завдання режиму роботи або ініціалізації внутрішніх значень: для завдань, обчислювальних ресурсів і методів розподілу. Важливим кроком є вибір, власне, методу розподілу. Процес моделювання розподілу завдань на ресурсах буде запущений після перевірки параметру tasks_count плагіну Algorythm-Loader.

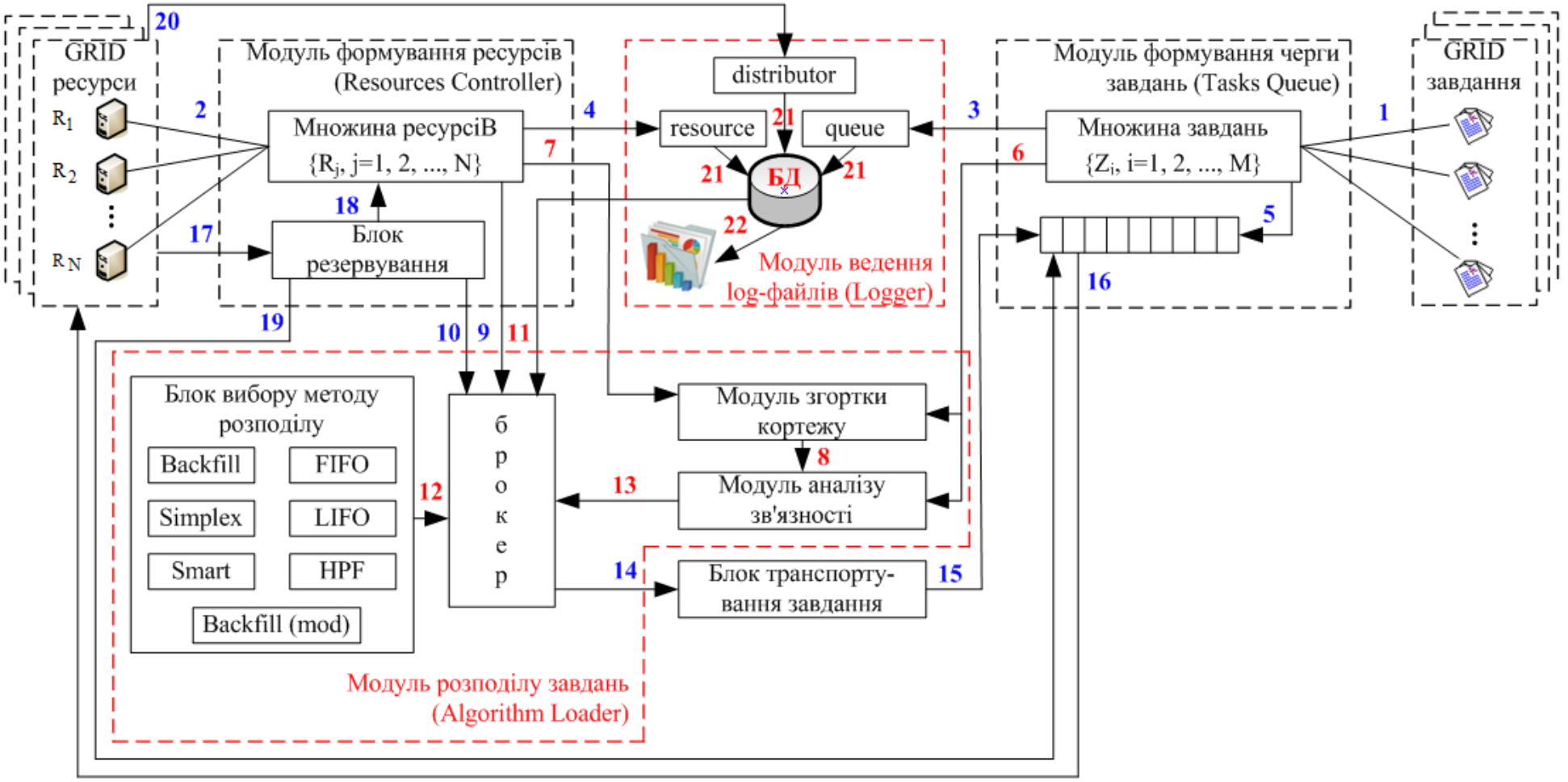

Рисунок 3.2 – Впровадження інформаційної технологія розподілу завдань у гетерогенній GRID-системі в імітаційне середовище моделювання GRASS

Як тільки кількість завдань в черзі перевищить значення, вказане в параметрі tasks_count, відбувається виклик методу розподілу, визначеного в параметрі DistributionAlgorythm. Відповідно до обраного методу розподілу [21] відбудеться підбір ресурсів для кожного із завдань в черзі. Модуль розподілу завдань містить ряд методів, що імітують роботу різних методів розподілу, кожен з яких використовує свій набір параметрів для розподілу. Перед запуском експерименту можливий вибір різного числа методів розподілу.

Після закінчення процесу моделювання розподілу завдань в log-файлах експерименту (а також в БД) буде знаходитися інформація по кожному з методів розподілу, на основі якої можна прийняти рішення у виборі кращого плану розподілу для вхідного пулу завдань. Результатом роботи першого етапу є зчитування та декодування конфігураційного файлу plugins.xml, завантаження основних плагінів системи для запуску процесу моделювання.

Якщо в ході аналізу виникне ситуація, в якій необхідні будуть додаткові результати моделювання для інших методів розподілу, експеримент доведеться повторювати, щоб не була порушена його чистота.

Етап 2: завантаження в середовище моделювання GRASS інформації про обчислювальні ресурси та завдання. Завдання, яке надходить до системи – це пакет задач, який представлений у вигляді окремої виконувананої програми. Процесор (ядро) у кожен момент часу може виконувати тільки одну задачу і завдання може бути запущено на виконання тільки в тому випадку, якщо для всіх задач завдання підібрані обчислювальні ресурси. Будь-яке завдання, яке надходить до середовища моделювання GRASS, можна розбити на дві складові: характеристики (параметри) завдання і тіло завдання (у вигляді .exe файлу, файли вхідних даних, БД тощо).

Завдання, що надходять до середовища моделювання GRASS, утворюють потік (зв’язок 1, рисунок 3.2) $\{Z_i, i = 1, 2, .., M\}$, де i – номер завдання, а M – кількість завдань. Надходячі до системи, відбувається поділ завдання по запропонованим вище складових. На початковому етапі для системи головне значення має інформація про завдання (3.2), яка оперує даними необхідними для пошуку обчи-

слювальних ресурсів, придатних для запуску [21]. Файл tasks_data.xml, що містить дану інформацію, приведений до заданого заздалегідь формату та дозволяє здійснити підбір конкретного ресурсу для виконання. Середовище моделювання GRASS дозволяє не тільки довантажувати сформований заздалегідь файл завдань за допомогою плагіна SimpleTasksManager, але і в ході процесу розподілу завантажити генератор формування завдань, використовуючи плагін SimpleTasks-Generator з файлу tasks_model.xml. Одночасно із завданнями до середовища моделювання надходить інформація про доступні ресурси (зв'язок 2, рисунок 3.2) $\{R_j, j = 1, 2, .., N\}$, де $-$ jпорядковий номер ресурсу, а N – число ресурсів в GRID-системі. Ресурси, як і завдання, приведені до заданого формату в файлі resources_data.xml. Так само, як і при формуванні завдань, можна скористатися плагінами SimpleResourcesManager та SimpleResourcesGenerator. Перший плагін використовується для завантаження ресурсів з файлу resources_data.xml, який був сформований заздалегідь, другий плагін завантажить обраний генератор ресурсів, який вказаний у файлі конфігурації resources_model.xml. Генератори формування обчислювальних ресурсів аналогічні генераторам завдань.

Паралельно відбувається виконання 8-го етапу, який відповідає за ведення log-файлів в середовищі GRASS.

Етап 3: формування додаткових параметрів для ефективного розподілу завдань [24]. Всі завдання, що надійшли до системи, поміщаються в чергу завдань (зв'язок 5, рисунок 3.2), і паралельно відбувається передача інформації по кожному завданню (у вигляді кортежу 3.2) в модулі згортки кортежу та аналізу зв'язності (зв'язок 6, рисунок 3.2). Модуль згортки кортежу здійснює обчислення узагальненого критерію оцінки для кожного завдання, який дозволить більш ефективно керувати процесом розподілу завдань на обчислювальних ресурсах і покаже, яку частину обчислювального ресурсу займає завдання в процесі його виконання [34]. Одночасно до даного модулю відбувається передача інформації про обчислювальні ресурси, які наявні в системі (зв'язок 7, рисунок 3.2). Результатом роботи модуля згортки кортежу є перелік ресурсів, на яких кожне завдання може бути розподілено.

Далі інформація надходить до модулю аналізу зв’язності (зв’язок 8, рисунок 3.2), де відбувається аналіз задач із завдання. Якщо задачі в завданні мають високу зв’язність (необхідний обмін інформацією між задачами в ході їх виконання) або для даного завдання необхідне пересилання великого обсягу вхідних / вихідних даних, відбудеться виклик методу checkQueueStrict (), який порівняє вимоги завдання з наявними обчислювальними можливостями і здійснить підбір обчислювальних ресурсів таким чином, щоб зменшити час передачі даних між задачами в завданні.

Якщо в системі присутні обчислювальні ресурси для запуску даного завдання, то воно залишиться в черзі та набуде стану Waiting і для нього в подальшому будуть обрані обчислювальні ресурси. Якщо таких ресурсів не виявиться, то завдання набуде стану Cancelled, після чого воно буде видалене з черги [21].

Якщо задачі в завданні не пов’язані між собою, то в системі запуститься метод checkQueueBase(), який аналогічно методу checkQueueStrict() здійснить підбір обчислювальних ресурсів з можливістю розподілу окремих задач із завдання на різних обчислювальних ресурсах і завдання залишиться в черзі в стані Waiting.

Використання інформації, що передається з модулю аналізу зв’язності, дозволяє здійснювати підбір обчислювальних ресурсів з урахуванням зменшення часу на передачу вхідних і вихідних даних для завдання.

Етапи 4-6: робота модулю розподілу завдань. На етапі 4 до модулю розподілу завдань (Algorithm Loader), надходить інформація, що дозволяє прийняти рішення для розподілу завдання на обчислювальний ресурс або ресурси, виконання на яких буде для нього оптимальним, за апаратним показниками і за характеристиками продуктивності. Брокеру для розподілу завдань необхідно отримати наступні дані:

- інформацію про обчислювальні ресурси (3.3), які присутні в системі (зв’язок 9, рисунок 3.2);

- інформацію про обчислювальні ресурси, що є задіяними (зв’язок 10, рисунок 3.2) і час їх вивільнення;

- інформацію з БД про попередні запуски завданнь (зв’язок 11, рисунок 3.2);

- метод розподілу (зв’язок 12, рисунок 3.2);

- інформацію про зв’язність задач в завданні (зв’язок 13, рисунок 3.2).

На основі отриманих даних здійснюється побудова плану розподілу завдань на обчислювальних ресурсах GRID-системи [36-39]. Результатом роботи етапу 4 є отримання множини планів розподілу за обраними заздалегідь методам розподілу.

Далі (етап 5) отримані плани розподілу направляються до середовища моделювання GRASS, де здійснюється їх запуск з метою отримання результатів, які в подальшому будуть проаналізовані для пошуку кращого плану розподілу. Результатом роботи п’ятого етапу є множина часів виконання пулу завдань з кожного методу розподілу, а також ряд значень критеріїв, які вираховуються для кожного методу.

Вибір плану розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів здійснюється на шостому етапі на підставі прийнятих заздалегідь критеріїв відбору (зв’язок 14, рисунок 3.2), які для кожного завдання визначають обчислювальні ресурси, відповідні за вказаними раніше вимогами. Будь-яке завдання, яке надійшло до середовища моделювання GRASS, буде розподілено. Винятком може бути випадок, коли в системі не виявиться необхідних обчислювальних ресурсів. У цьому разі завданням буде відмовлено у виконанні, і вони будуть відправлені постачальнику для зміни вимог.

Етап 7: процес передачі завдання на обчислювальний ресурс або ресурси, зазначені в плані розподілу. Для цього з блоку, що відповідає за транспортування, здійснюється запит до черги завдань (зв’язок 15, рисунок 3.2), який, відповідно до плану розподілу, вибирає конкретне завдання і привласнює йому стан Running. Далі завдання (друга складова) відправляється до обраного обчислювального ресурсу для його виконання (зв’язок 16, рисунок 3.2). Оскільки в системі сталися події з ресурсами (в даному випадку: обрано ресурс для виконання завдання), то необхідно сповістити систему про це (зв’язок 17, рисунок 3.2). Блок резервування передає інформацію системі, про те, що повністю задіяний обчислювальний ресурс не може бути використаний для подальшого запуску до моменту закінчення виконання на ньому завдання (зв’язок 18, рисунок 3.2).

Коли знову відбудуться дії з ресурсами (наприклад, станеться вивільнення ресурсу), блок резервування отримає інформацію про це (зв'язок 17, рисунок 3.2) і сповістить систему про можливість його подальшого використання в розподілі (зв'язок 18, рисунок 3.2), а також відправить повідомлення до черги завдань для видалення його з черги в зв'язку з закінченням його виконання (зв'язок 19, рисунок 3.2).

Етап 8: збір статистичної інформації про експеримент. Для отримання статистичної інформації про розподіл та аналіз її в подальшому, відбувається запуск модуля, який відповідає за формування log-файлів. У середовищі GRASS за збір статистики відповідає модуль ведення log-файлів (Logger). Даний модуль надає можливість гнучкого та централізованого ведення логів для всіх плагінів середовища моделювання GRASS.

У системі є кілька типів log-файлів, які фіксують ряд дій, що дозволяє при необхідності швидко отримати необхідну інформацію з БД з метою усунення можливих помилок, побудови статистичних моделей тощо.

При надходженні завдань до системи відбувається автоматичний запис даних про завдання та його вимоги в файл queue.log (зв'язок 3, рисунок 3.2), додання обчислювальних ресурсів в систему також призводить до формування файлу resources.log (зв'язок 4, рисунок 3.2). Аналогічно завданням, обчислювальні ресурси мають параметри, за якими в подальшому буде здійснено їх підбір. У момент запуску або закінчення виконання завдання на обчислювальному ресурсі відбувається фіксація інформації про дані дії в файлі raspred.log (зв'язок 20, рисунок 3.2), тобто відбувається дозапис даного файлу. Даний файл є масштабованим, тому що містить всю послідовність дій, які відбуваються в системі. Всі записи з log-файлів надходять до БД системи (зв'язок 21, рисунок 3.2), дані з якої можуть бути використані в подальшому для здійснення вибірки по заданих параметрах або для графічного відображення інформації за обраний проміжок часу (зв'язок 22, рисунок 3.2) .

Процес моделювання по кожному методу розподілу триває до тих пір, поки в черзі присутні завдання.

3.6 Побудова абстрактної моделі інформаційної технології розподілу завдань на обчислювальних ресурсах

UML – це відкритий стандарт, який може бути використаний для побудови абстрактної моделі системи, що розробляється за допомогою графічних позначень і дозволяє розробникам ПО досягти ряд угод в графічних позначеннях для представлення низки загальних понять (компонент, клас, поведінку, агрегація) [147].

В UML використовуються наступні види діаграм:

- діаграма прецедентів (use case);
- діаграма класів (class);
- діаграма станів (statechart);
- діаграма діяльності (activity);
- діаграма послідовності (sequence);
- діаграма кооперації (collaboration);
- діаграма компонентів (component);
- діаграма розгортання (deployment).

У поточну версію UML 2.5 був внесений ряд поліпшень: змінилося зовнішнє уявлення діаграм, уточнена та розширена семантика мови для підтримки методології Model Driven Development (MDD), удосконалилися окремо взяті нотації.

Не існує строгого переліку діаграм, які необхідно будувати при розробці системи. Існують рекомендації, які вказують, що необхідно приділити увагу тим діаграмам, які в повній мірі відображають архітектуру інформаційної технології. Складну систему необхідно представити у вигляді набору невеликих і незалежних моделей-діаграм, що допоможе надалі уникнути конфліктів між їх окремими частинами на етапі «збирання». Найбільш важливі прецеденти можна виділити в ході побудові моделі взаємодії користувачів з інформаційною технологією. Діаграми класу використовуються при проектуванні системи для уточнення її структури, тому що вони містять інформацію про атрибути та перелік послуг, які надаються об’єктами даного класу. На рисунку 3.3 наведені рекомендації щодо вибору UML діаграм, системи що розробляється.

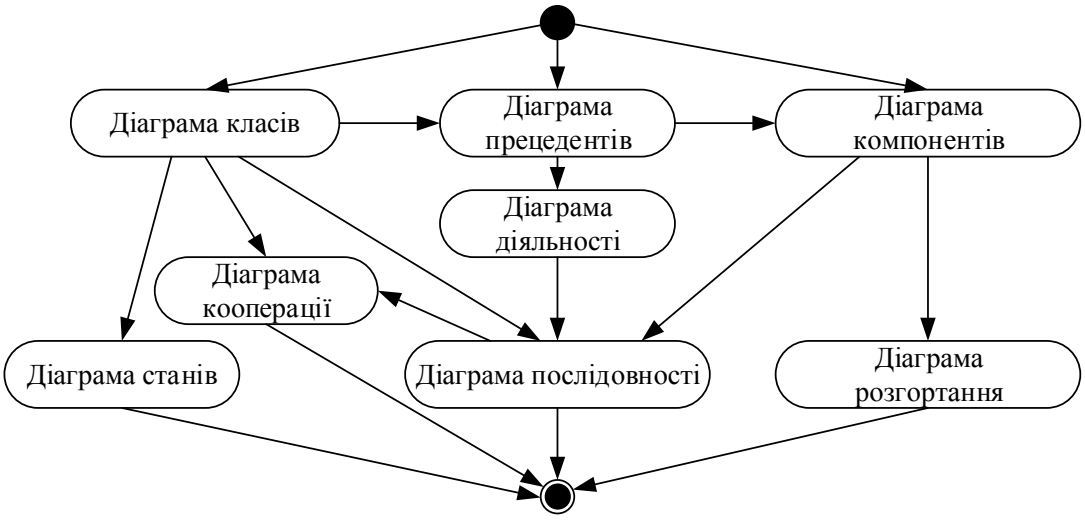

Рисунок 3.3 – Рекомендації з побудови UML діаграм

Нижче наведено ряд діаграм, які дозволять візуалізувати запропоновану систему з різних точок зору: діаграма прецедентів (рисунок 3.4), діаграма активності (рисунок 3.5) та діаграма послідовностей (тимчасова діаграма) (рисунок 3.6).

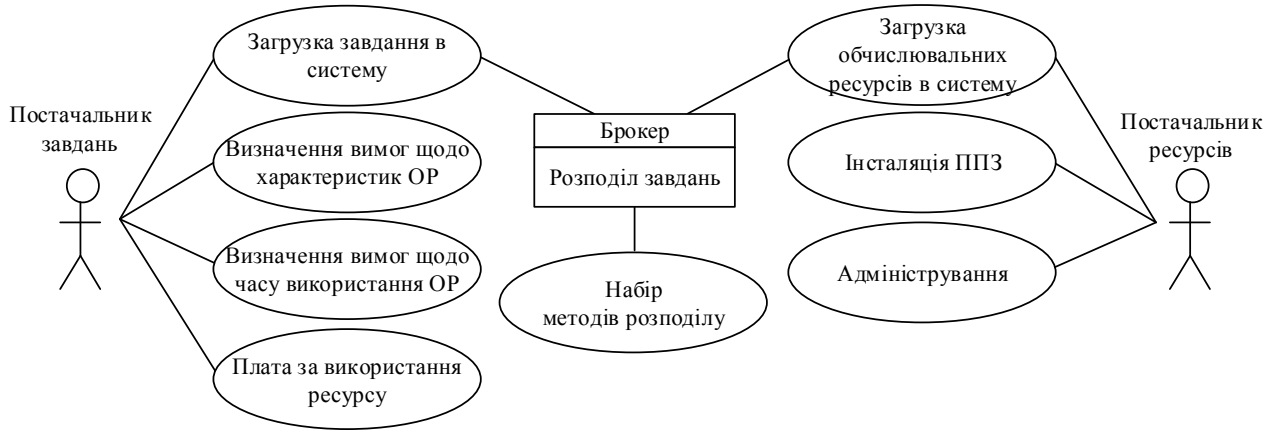

Рисунок 3.4 – Діаграма прецедентів

Діаграма прецедентів – це діаграма, яка відображає відносини між акторами та прецедентами і визначає функціональні вимоги до розроблюваної системі [147]. На рисунку 3.4 показана діаграма прецедентів запропонованої інформаційної технології розподілу завдань на обчислювальни ресурси GRID-системи. Діаграма прецедентів відіграє важливу роль у моделюванні поведінки компонентів технології, показує низку прецедентів, акторів і відносин між ними. Дані діаграми використовуються для візуалізації та документування поведінки окремо взя-

того елементу, тому що полегшують розуміння систем, підсистем або класів, представляючи погляд ззовні на те, як дані елементи можуть бути використані у відповідному контексті. Крім того, такі діаграми важливі для тестування виконуваних систем в процесі прямого проектування і для розуміння їх внутрішнього устрою при зворотньому проектуванні [147].

Розглянемо розроблену технологію розподілу завдань на обчислювальні ресурси, вказавши етапи технології в прив'язці до діаграми дій, декомпозувавши послідовні та паралельні етапи (рисунок 3.5).

Етап 1 відноситься до послідовного етапу, тому що дозволяє сформувати спрямованість і наповнюваність експерименту. Результатом роботи пешого етапу є зчитування та декодування конфігураційного файлу plugins.xml, завантаження основних плагінів системи для запуску процесу розподілу.

Етап 2 може бути розподілений на дві паралельні складові, тому що формування (завантаження) завдань та обчислювальних ресурсів жодним чином не залежать один від одного. Як тільки почнеться завантаження вхідних даних, паралельно запускається етап 8. Роботу восьмого етапу можна назвати фоновою, тому що даний етап працює постійно: відбувається збереження статистичних даних (формування log-файлів, дозапис в БД даних про експерименти).

Діаграма активностей – це послідовність протяжних за часом дій. Однак слід враховувати, що кожна дія може складатися з більш простих дій (етап 3): для кожного завдання, яке надійшло на вхід системи, відбувається згортання кортежу (модуль згортки), а також аналіз зв'язності задач в завданні (модуль аналізу зв'язності). Результатом роботи модуля згортки кортежу є перелік ресурсів, на яких кожне завдання може бути розподілено, модуль аналізу зв'язності надасть вибірку обчислювальних ресурсів по кожному з завдань, що надійшли [24,41]. Отримання перерахованих параметрів для кожного завдання здійснюється паралельно, однак в діаграмі дій (етап 3) показано, що для кожного завдання послідовно повинні бути сформовані ці дані.

Наступні три етапи є послідовними, проте на кожному етапі відбувається паралельна робота за різними методами розподілу.

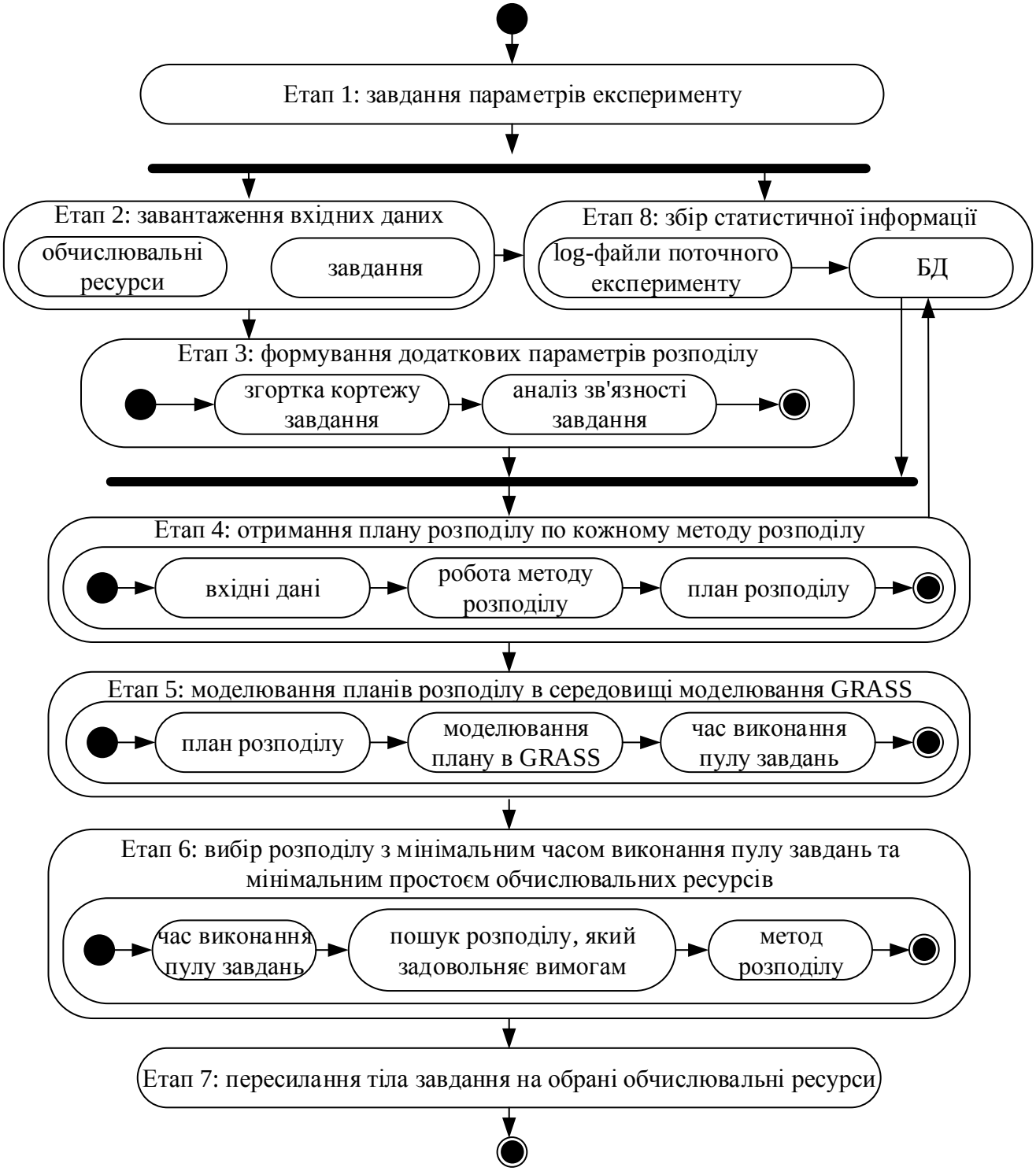

Рисунок 3.5 – Діаграма активності інформаційної технології розподілу завдань

На четвертому етапі по кожному методу розподілу відбувається отримання плану розподілів. Далі (етап 5) отримані плани моделюються в середовищі GRASS для отримання часу виконання пулу завдань. Після чого (етап 6) з урахуванням заданих експертом (системним адміністратором) критеріїв відбувається

аналіз, результатом якого є вибір плану з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів для конкретного пулу завдань.

На сьомому етапі відбувається пересилка завдання до обраного обчислювального ресурсу. В ході передачі завдання здійснюються сповіщення всієї системи: з черги завдань видаляється розподілене завдання, обчислювальні ресурси отримують статус «зайняті», в блок резервування надходить інформація про їх зайнятість на певний час, який необхідний для виконання завдання, відбувається запис даних дій в log-файли (етап 8). Статистична інформація в процесі моделювання надходить в відповідні таблиці БД, вона використовується для прийняття рішень в процесі розподілу, а також допомогає провести аналіз завантаженості GRID-системи для подальшого зменшення коефіцієнту простою її ресурсів.

Для більш зрозумілої роботи системи необхідний не тільки алгоритм послідовності дій, а й обмін повідомленнями між її об’єктами. У цьому може допомогти діаграма кооперації (взаємодії). У версії UML 2.0 з’явилася тимчасова діаграма, яка є різновидом діаграм взаємодії. Тимчасова діаграма – це сукупність діаграм станів і послідовності, призначена для відображення стану окремих об’єктів системи в ході взаємодії з часу [147]. На рисунку 3.6 наведена тимчасова діаграма запропонованої інформаційної технології розподілу завдань.

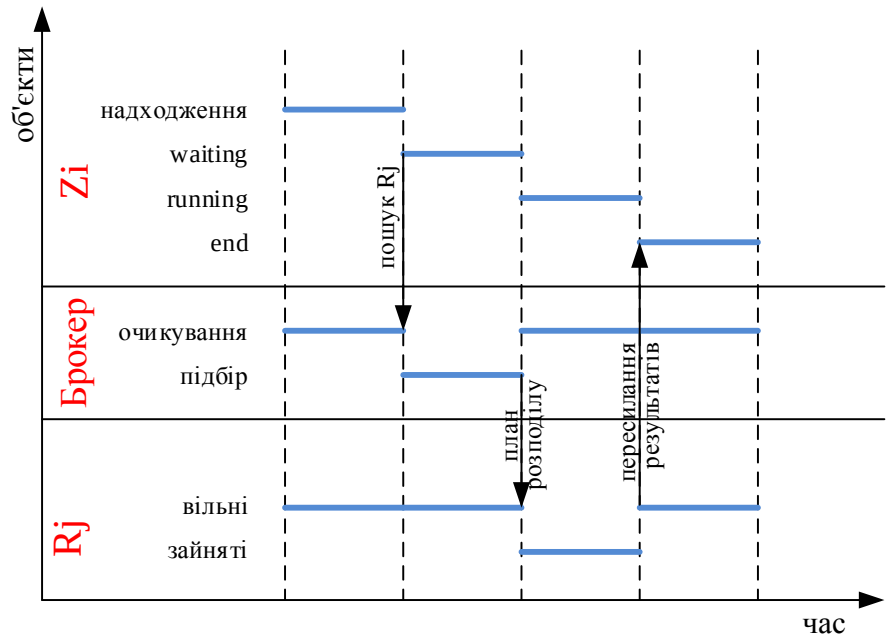

Рисунок 3.6 – Тимчасова діаграма інформаційної технології розподілу завдань

Основний упор в тимчасовій діаграмі робиться на наочне зображення статків потоків завдань та обчислювальних ресурсів та як вони змінюються в часі.

3.7 Розробка підсистеми моніторингу мережного трафіку в GRID-системі

В сучасних GRID-системах при розподілі завдань на обчислювальних ресурсах орієнтуються на один критерій (наприклад, продуктивність кластера). Доповнюючими сутностями критерію можуть бути задані умови обмеження типу «обсяг пам'яті не менше, ніж ...». При цьому абсолютно не враховуються такі параметри як пропускна здатність каналу зв'язку, його завантаження та затримка передачі даних в каналі.

З огляду на практичні напрацювання, відомо що, доставка вхідних даних на відібраний обчислювальний ресурс або отримання вихідних даних займають певний відрізок часу суміжний з часом виконання завданнь. Рішенням даної задачі може бути: використання інфокомунікаційних мереж, які надають послуги за принципом «з максимальним зусиллям» (best effort), що вимагає певного запасу по пропускній здатності. Іншим рішенням є зменшення часу доставки завдання до обчислювального ресурсу із забезпеченням заданої якості обслуговування, яке передбачає знання про завантаження проміжних елементів мережі. Оперування даною інформацією дозволяє доставити завдання до відібраного обчислювального ресурсу та повернути результат виконання, за часом не більше заданого. Тому для моніторингу завантаження проміжних елементів мережі необхідні відповідні мережні інструменти.

В даний час в комп'ютерних мережах використовують ряд методів для моніторингу мережного трафіку:

- моніторинг на канальному рівні (SNMP);
- моніторинг на мережному, частково прикладному рівні (RMON);
- моніторинг мережі «в цілому».

Однак використання даних методів моніторингу має ряд особливостей, які ускладнюють моніторинг мережного трафіку для GRID-систем великих розмірів.

Використання SNMP (Simple Network Management Protocol) і RMON (Remote MONitoring) потребує підтримки їх за допомогою додаткового мережного обладнання, що призводить до необхідності збільшення продуктивності наданого обладнання, а значить і його вартості [142]. Найчастіше виробники в своїх пристроях частково реалізують можливості SNMP та RMON, тому що зазначені технології завантажують своїм трафіком канали зв'язку. Тому виробники мережного обладнання створюють власні технології моніторингу. В даний час в комп'ютерних мережах функціонують протоколи NetFlow (Cisco Systems) та sFlow (Hewlett-Packard), специфікації яких є відкритими. Це призвело до їх широкого поширення. Особливістю протоколів NetFlow та sFlow є використання колектора, в функції якого входить збір і обробка інформації про мережний трафік від агентів сегмента мережі, що істотно знижує обсяг трафіку, який передається для моніторингу. Відмінність протоколу sFlow від NetFlow полягає в тому, що він орієнтований на аналізі статистичної вибірки пакетів, що також знижує обсяг службового трафіку.

Вищенаведені технології, що використовуються для моніторингу мережного трафіку в GRID-системах, стикаються з рядом проблем:

- територіальна розподіленість обчислювальних ресурсів призводить до зростання проміжних елементів мережі;

- проміжні елементи (підмережі) можуть належати різним власникам, які реалізують відмінні один від одного підходи моніторингу мережного трафіку та забезпечення доступу для отриманої інформації;

- унеможливлення контролю за стиками мереж провайдерів;

- кросплатформність мережного обладнання та програмного забезпечення;

- вартість готових рішень зависока, а набір функцій, які вони надають користувачам вузькоспеціалізовані.

У зв'язку з цим задача моніторингу мережних ресурсів GRID-систем і мережного трафіку зокрема є актуальною задачею для забезпечення необхідної якості обслуговування при передачі великих обсягів даних між різними обчислювальними ресурсами (QoS).

Запропонована архітектура підсистеми моніторингу має характеризуватись такими функціями:

- постійний збір статистичної інформації від агентів мережі та передача підсумкових результатів;

- моніторинг мережного трафіку за основними мережними протоколами різних рівнів OSI та програмно-апаратних платформ;

- вирішення питань, пов’язаних з доступом до мережних ресурсів різних провайдерів;

- прогнозування завантаженості мережі на підставі поточної та зібраної інформації про її стан.

Розробка системи моніторингу, з перерахованими вище можливостями, призводить до значних затрат. При цьому дана система повинна враховувати особливості запропонованої інформаційної технології (підрозділ 3.5). Отже, для зменшення витрат, пов’язаних з розробкою системи моніторингу трафіку, рекомендується максимально використовувати існуючі стандарти технологій моніторингу мережевого трафіку та розробити інтерфейс, який використовує різні технології моніторингу мережного трафіку в єдиній системі. Архітектура запропонованої підсистеми моніторингу мережного трафіку [20,32] показана на рисунку 3.6.

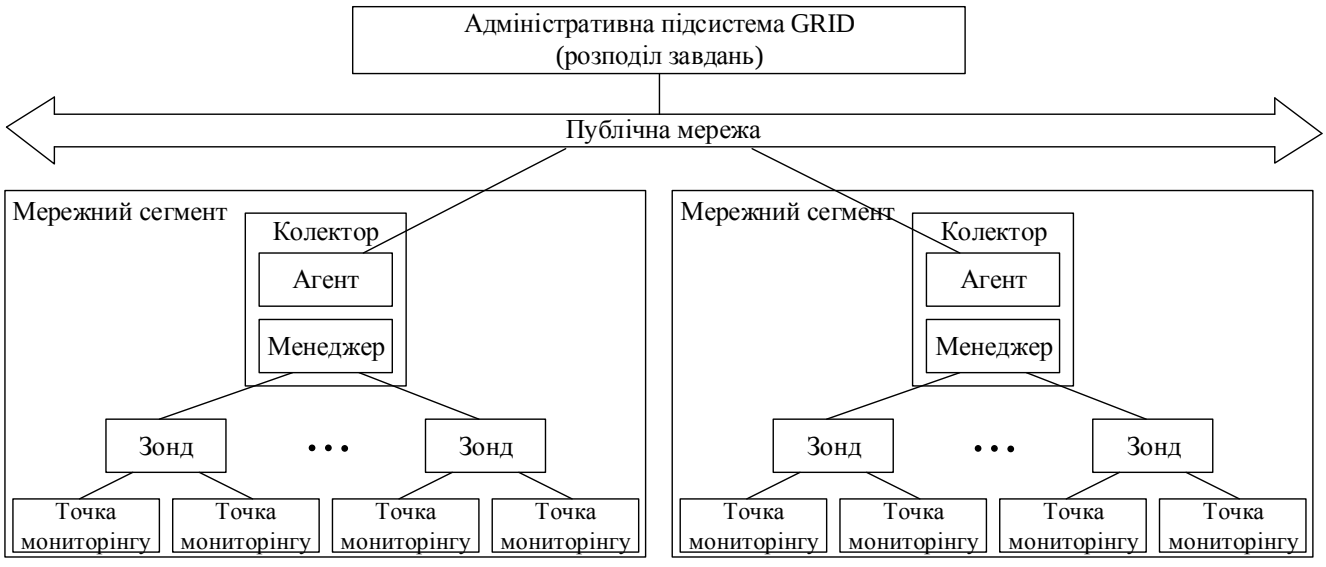

Рисунок 3.6 – Архітектура підсистеми моніторингу мережевого трафіку

Точка моніторингу – це агент моніторингу трафіку (SNMP, RMON), який розміщений в необхідних мережних сегментах. Зонд – це інтерфейс для точок моніторингу, які розроблені за різними мережними технологіями. Він виконує функцію фільтрації, що надходить від агентів, інформації про мережний трафік, за раніше заданим критеріями.

Колектор складається з двох частин: агента та менеджера. Менеджер відповідає за функцію збору, аналізу та зберігання інформації, отриманої від зондів, а також надає інтерфейс для налаштування конкретного зонду. Агент призначений для зв'язку мережного сегменту моніторингу з адміністративною підсистемою GRID-середовища через загальну мережу. Він виконує функції аунтефікації та розподілу доступу до запитуваної інформації.

Функціями адміністративної підсистеми GRID-середовища є:

- налаштування колекторів у відповідності зі стратегією моніторингу;

- опитування колекторів для отримання необхідної інформації про мережний трафік;

- прогнозування стану мережі;

- розподіл завдань по мережним сегментам (обчислювальним ресурсам) з урахуванням отриманої від колекторів інформації.

Запропонована архітектура системи моніторингу мережного трафіку в інфокомунікаційних мережах дозволяє підвищити ефективність розподілу завдань на обчислювальних ресурсах в GRID-системі за рахунок врахування додаткових факторів, які орієнтовані на завантаженість каналів передачі даних і їх пропускну здатність. Запропонована система моніторингу може бути реалізована як надбудова в програмному забезпеченні планувальника завдань в GRID-системах [20].

3.8 Розробка функціональної моделі модуля розподілу потоку завдань

На рисунку 3.7 представлена структурна схема розробленого модуля розподілу потоку завдань на обчислювальних ресурсах Algorithm Loader. Вона складається з чотирьох окремих блоків, а для коректної роботи використовує потоки за-

вдань і обчислювальних ресурсів, які формуються поза модулем згідно раніше заданих правил (кортежі 3.2, 3.3) [21].

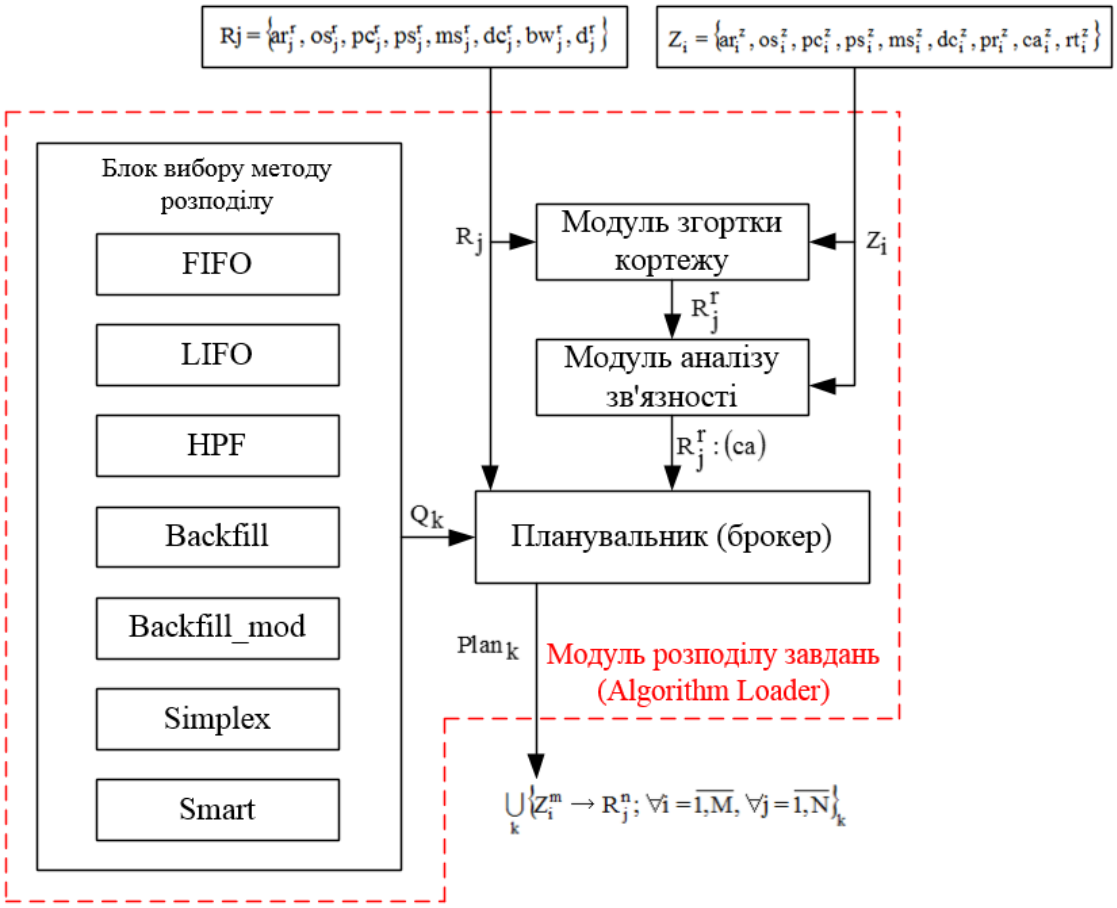

Рисунок 3.7 – Структура розробленого модуля розподілу потоку завдань на обчислювальних ресурсах (Algorithm Loader)

У середовищі моделювання GRASS було прийнято ряд правил, одним з яких є, те, що завдання, у міру проходження у системі може змінювати свій стан. Так, завдання у черзі характеризуються одним з восьми значень:

- Failed – завдання завершено з помилкою;
- Finished – завдання завершено успішно;
- Invalid – некоректне завдання або помилка роботи системи;
- Running – завдання виконується на обчислювальному ресурсі;
- Timeout – завдання знято з виконання у відповідності до тайм-ауту;
- Waiting – завдання очікує розподілу в черзі;
- Cancelled – виконання завдання скасовано через відсутність в системі зая-

влених постачальником завдань вимог щодо обчислювальних ресурсів;

- Planned – завдання заплановано для виконання при виборі методів Backfill та Backfill_mod.

В процесі надходження завдань до системи відбувається постійна перевірка наявності завдань у черзі. Якщо кількість завдань стану Waiting дорівнює чи перевищує параметр tasks_count, то надходить повідомлення про те, що ці завдання можуть бути обслуговані даною системою і, відповідно, до обраного методу (методів) розподілу відбудеться побудова плану (планів) запуску завдань на обчислювальних ресурсах.

Блок методів розподілу містить наступні методи: First-Come First-Served (FCFS), Last In First Out (LIFO), Highest Priority First (HPF), метод зворотнього заповнення (Backfill), модифікований метод зворотнього заповнення з урахуванням пропускної здатності і ширини каналу зв'язку (Backfill_mod), метод лінійного програмування (Simplex), метод розподілу за вивільними ресурсами (Smart). Кожен з представлених методів використовує свій набір параметрів для розподілу. Винятками є методи FCFS і LIFO, тому що – це найпростіші алгоритми розподілу і для їх розподілу не потрібно формування додаткових параметрів.

Після активації модулів Tasks Generators (формування черги) та Resources Generators (формування ресурсів), починає роботу модуль згортки кортежу, який здійснює обчислення узагальненого критерію оцінки для кожного завдання (3.2), що дозволяє більш продуктивно керувати процесом розподілу завдань на обчислювальних ресурсах.

Результатом роботи модуля згортки кортежу за кожним завданням є множина $\Omega_{Z_i} = \left(R_j^r \to part_j\right), \quad j = 1, ..., K$, що складається з підмножини $\left\{R_j^r,\ j = 1, ..., K\right\}$ (обчислювальні ресурси, які були відібрані з кожного завдання на підставі узагальненого критерію оцінки завдання) та $\left\{part_j,\ j = 1, ..., K\right\}$ (підмножина коефіцієнтів використання обчислювальних ресурсів, що показує, яку частину обчислювального ресурсу буде займати завдання в процесі його виконання).

За допомогою модуля аналізу зв'язності є можливість аналізу зв'язності за-

дач в завданні. Якщо задачі в завданні мають високу зв’язність (необхідний інтенсивний обмін інформацією між задачами в завданнями в ході їх виконання) або для даного завдання необхідна передача великого обсягу вхідних або вихідних даних, то при розподілі задач із завдання пріоритет буде поставлений на підбір обчислювальних ресурсів таким чином, аби зменшити час на передачу даних між ними.

Для розподілу завдань на обчислювальні ресурси планувальнику (брокеру) необхідно отримати інформацію про стан системи в цілому та про вимоги завдань. На підставі цієї інформації будується план, який для кожного завдання визначає обчислювальний ресурс, що відповідає за раніше вказаними вимогам. Якщо перед початком експерименту в файлі plugins.xml було обрано кілька методів розподілу, то таких планів буде теж декілька (відповідно до обраних методів розподілу). Також в ході експерименту збирається статистична інформація про завантаженість системи, час перебування завдання в системі та черзі.

Блок транспортування завдання до обчислювального ресурсу здійснює, власне, доставку завдання до обраного ресурсу, присвоює завданням стан Running іта резервує ресурс (ресурси) на час виконання завдання. Після закінчення часу, запланованого на виконання завдання, плагіном Distributor виконується метод terminateTask(), який відповідає за видалення завдання із черги та повернення обчислювальних ресурсів для подальшого використання в GRID-системі.

3.9 Розробка структури бази даних і використання її в середовищі моделювання GRASS

На сьогодні для успішного функціонування організацій, потрібна розвинена інформаційна система, яка реалізує автоматизований процес збору та обробки даних. Одним з найбільш зручних засобів для раціонального та ефективного зберігання інформації є бази даних.

База даних – це організована за певними правилами сукупність даних про процесах, об’єкти, явища і події, які відносяться до заданої області [148]. Додатки

БД забезпечують користувачам надійний захист інформації від випадкової втрати або псування, а також забезпечують механізмами для пошуку інформації.

БД організовується таким чином, щоб мати можливість забезпечувати інформаційні запити користувачів, розподілено зберігати ряд даних. Більшість сучасних баз даних мають реляційну структуру. У таких БД всі дані можуть бути представленими у вигляді простих таблиць, які розбиті на рядки та стовпці, на перетині яких розташовуються дані. Таблиці в реляційних базах даних мають наступні властивості:

- таблиця не може мати два однакові рядки;
- стовпчики розташовуються в порядку, заданому при створенні таблиці;
- кожен стовпець має своє унікальне ім’я та конкретний тип;
- на перетині рядків і стовпців може перебувати тільки атомарне значення (одне значення, яка не перебуває в групі значень).

Розробка структури бази даних – це одна з найважливіших задач, що вирішуються при проектуванні БД. Структура БД включає в себе: набір форм, таблиць, зв’язків між ними; є основним проектним рішенням при створенні додатків з використанням БД. Створена структура БД описується на мові визначення даних системи управління базами даних (СКБД). Будь-яка СУБД дозволяє виконувати ряд операцій з даними:

- додання (видалення) записів у таблиці;
- оновлення окремо взятих полів в одній або кількох записах таблиць;
- пошук записів за заданими запитами.

Проектування БД починається з побудови концептуальної моделі, яка відображає предметну область. Далі відбувається перетворення концептуальної моделі в реляційну. Для цього необхідно перерахувати наявні таблиці із зазначенням первинних ключів, а також провести нормалізацію.

Первинним ключем (primary key – PK) виступає стовпець, значення якого у всіх рядках таблиці різні. Первинні ключі можуть бути двох типів: логічні та сурогатні. Перевагою використання сурогатних ключів у відношенню до логічних є можливість абстрагувати ключі від реальних даних. Сурогатний ключ – це додат-

кове поле в базі даних, тобто порядковий номер запису.

Зовнішнім ключем (foreign key – FK) виступає стовпець (стовпці), які застосовується для примусового встановлення зв’язку між даними в двох різних таблицях, тобто описують їх первинний ключ.

Нормалізація – це процес організації даних в БД, який включає в себе створення таблиць та встановлення зв’язків між ними відповідно до обумовлених правил, які спрямовані на забезпечення захисту даних. Запропоновані дії роблять базу даних більш гнучкою, усувають надмірність і неузгоджені залежностей.

Після того, як концептуальна модель перетворена у реляційну, необхідно її реалізувати в конкретній СУБД. Основним об’єктом СУБД є таблиці. Побудова і заповнення таблиць починається з тих таблиць, які є головними, в даному випадку це таблиці «Завдання» та «Ресурси». Далі відбувається підключення до них додаткових таблиць, які конкретизують (уточнюють) інформацію.

Розробка таблиць БД починається з опису таблиць, тобто необхідно конкретизувати структуру таблиці. Під цим розуміється: опис найменувань, типів полів, ряд додаткових характеристик (формат і критерії перевірки даних, що вводяться). Крім опису структури таблиць задаються типи зв’язків між ними, які дозволяють регулювати відносини між об’єктами предметної області.

На завершальному етапі процесу проектування структури таблиці відбувається визначення ключів та індексів. Після створення опису структури таблиці необхідно перейти в режим безпосереднього введення в неї даних. На основі створених таблиць в подальшому створюються запити, звіти, призначені для користувача форми, що і складає реляційну базу даних.

Запит – це вимога, яку висуває користувач на вилучення інформації з таблиць бази даних, на внесення змін в базу даних, а також на виконання обчислень над даними. СУБД дозволяють створювати запити трьох типів: запити на вибірку, перехресні запити, запити на внесення змін в базу даних. Розглянемо ці типи.

Результатом запиту на вибірку є динамічна таблиця, яка може бути переглянута та проаналізована у подальшому. Запит на вибірку дозволяє:

- включати в результуючу таблицю поля з однієї або декількох таблиць в

потрібному порядку;

- вибирати записи, що задовольняють умовам відбору;
- здійснювати обчислення над полями бази даних.

На рисунку 3.8 показана структура бази даих, яка була розроблена.

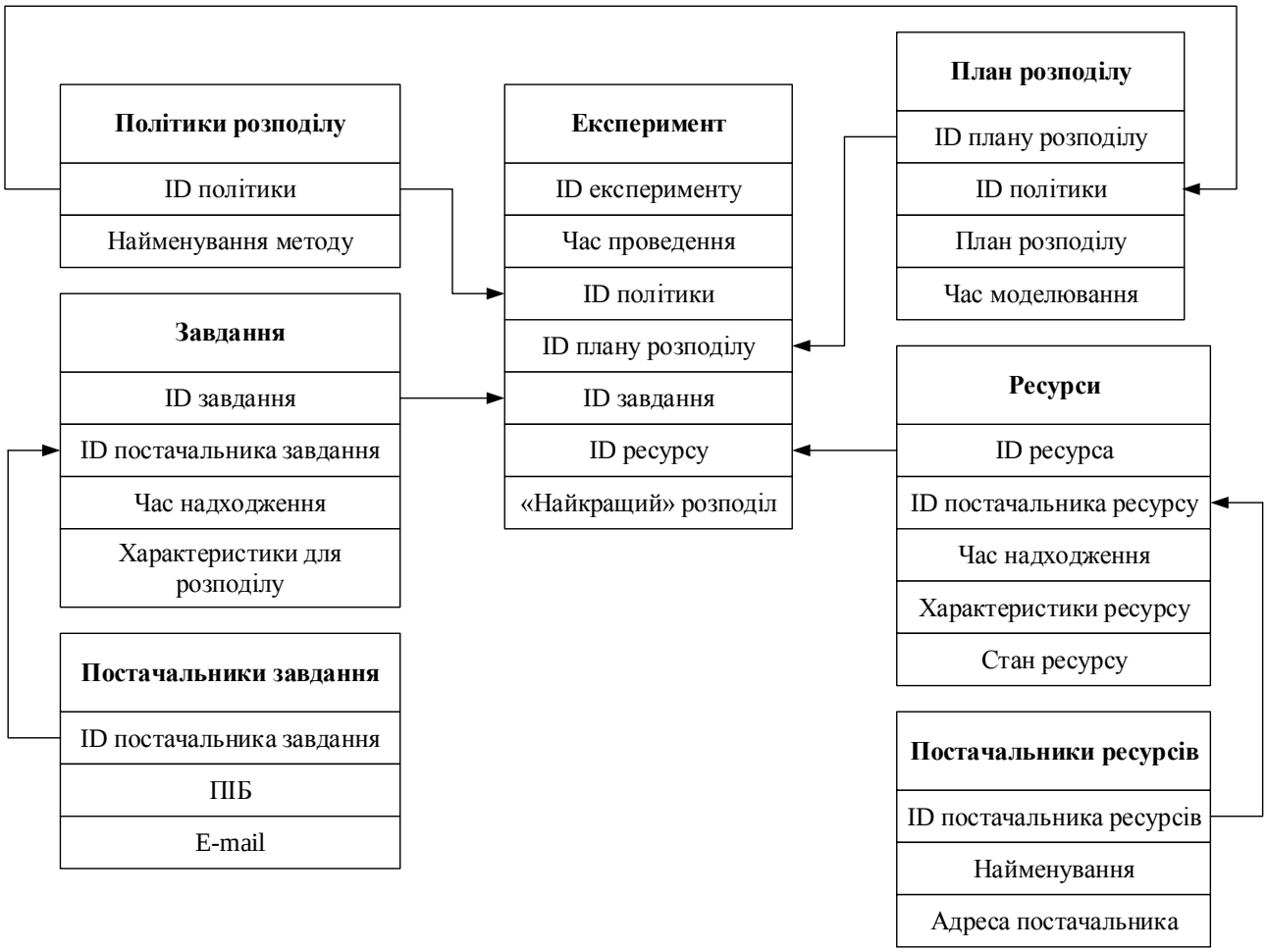

Рисунок 3.8 – Структура запропонованої бази даних, яка використовується в середовищі моделювання GRASS

Результатом роботи перехресного запиту є відображення результатів підсумкових статистичних розрахунків над значеннями деякого поля.

У процесі проведення експериментів можуть виникнути ситуації, коли необхідно здійснити аналіз отриманих результатів. Для таких цілей в системі є модуль скриптових інтерфейсів (Script Interfaces), який дозволяє користувачу сформувати запити, необхідні для отримання необхідної інформації.

Сучасні GRID-системи працюють під управлінням ОС Scientific Linux. Да-

ний дистрибутив містить в собі набір програмних компонентів, які необхідні для організації серверної інфраструктури. На базі Scientific Linux можуть бути створені різні структури: поштовий сервер, сервер збереження даних, web-сервер, сервер БД тощо. До складу дистрибутива входить ряд компонентів, які необхідні для створення кластерної інфраструктури різного призначення: обчислювальних кластерів, кластерів високої надійності і кластерів з балансуванням навантаження. Як правило, сервер БД в Scientific Linux реалізується за допомогою MySQL-системи (або PostgreSQL) керування базами даних, ПО яке є відкритим, що дозволяє застосовувати і модернізувати його під свої завдання.

3.10 Висновки по розділу

Була запропонована модифікована модель розподілу завдань на обчислювальних ресурсах у GRID-системі (розширена за рахунок введення множини методів розподілу та додаткових параметрів при поданні обчислювальних ресурсів і завдань):

- використання у запропонованій моделі коефіцієнту зв’язності дозволяє здійснювати підбір обчислювальних ресурсів з урахуванням мінімізації часу на обмін даними між завданнями;

- введення в модель сумарної затримки часу передачі пакету в каналі зв’язку та його пропускної здатності дозволяє скоротити час виконання пулу завдань і підвищує ефективність використання обчислювальних ресурсів в GRID-системі;

- розширення моделі розподілу завдань в GRID-системі за рахунок множини методів розподілу дозволяє здійснювати серії експериментів для різних пулів завдань з подальшим вибором плану розподілу з мінімальним часом виконання цих пулів завдань та мінімальним простоєм обчислювальних ресурсів.

При розподілі завдань на обчислювальних ресурсах на якість розподілу впливає ряд параметрів. Для отримання плану розподілу з мінімальним часом виконання цих пулів завдань та мінімальним простоєм обчислювальних ресурсів, необхідно скоротити кількість критеріїв розподілу, залишаючи тільки найбільш

важливі. На підставі цього твердження пропонується здійснити згортку кортежу (3.2) з використанням узагальненого критерій оцінки завдання, за рахунок чого спрощується процедура підбору обчислювальних ресурсів для запуску завдання.

Запропоновано модифікацію методу Backfill з консервативним резервуванням, яка, на відміну від існуючого, дозволяє розподіляти завдання в залежності від зв'язності завдань в кожному із завдань. Запропонований метод оперує двома додатковими параметрами: затримкою часу передачі пакету по мережі (δ_n) та пропускною здатністю каналу зв'язку (C_n). Залежно від класу завдання, здійснюється підбір обчислювальних ресурсів за одним із запропонованих параметрів, завдяки чому зменшується час простою обчислювальних ресурсів (за рахунок розвантаження каналів зв'язку).

Розроблено метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів. Результати експериментів у середовищі моделювання GRASS підтверджують припущення, що запропоновані методи розподілу завдань для різних вхідних пулів завдань і обчислювальних ресурсів можуть давати виграш за часом та скорочувати простої обчислювальних ресурсів.

Запропоновано інформаційну технологію розподілу завдань на обчислювальні ресурси для GRID-систем, яка впроваджена в імітаційне середовище моделювання GRASS. Дане рішення дозволяє проводити обчислювальні експерименти, які реалізують різні методи розподілу, з подальшим вибором ефективного рішення на основі збору, аналізу та інтерпретації результатів моделювання.

Організація моніторингу мережних ресурсів GRID-систем є важливою проблемою для забезпечення якості обслуговування при передачі великих обсягів даних між різними мережними обчислювальними ресурсами. В роботі розроблено архітектуру системи моніторингу, яка дозволяє підвищити ефективність використання ресурсів за рахунок ряду параметрів, які враховують завантаженість каналів зв'язку та їх пропускну здатність.

Розроблено та впроваджено в імітаційне середовище моделювання GRASS

модуль розподілу завдань (Algoritm Loader), який грунтується на множині методів розподілу. Завдяки його доданню стало можливим створення ряду обчислювальних експериментів, які дозволяють здійснити збір, аналіз результатів моделювання, а також зіставлення отриманих результатів з реальною поведінкою досліджуваного об’єкту. Модуль розподілу завдань, який оперує великою кількістю методів розподілу, дозволяє здійснити моделювання розподілу для конкретного пулу завдань різними методами розподілу та проаналізувати результати моделювання.

Для збору статистичної інформації та подальшого її аналізу була розроблена структура бази даних, яка включає в себе набір таблиць та зв’язків між ними. Створена БД описується мовою визначення даних системи (MySQL).

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [1,19-21,24,25,27,28,30,32,34,36-41,43,76,93, 134,142,144-148].

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ, ЯКА БУЛА ЗАПРОПОНОВАНА, В ПРОЕКТІ GRASS

4.1 Обґрунтування вимог до архітектури середовища моделювання GRID-системи

Архітектура середовища моделювання GRID-системи повинна задовольняти ряду вимог: розширюваності, масштабованості, продуктивності та тестопридатності.

Вимога щодо розширення – це умова додання нових функцій або заміни деяких компонентів системи без внесення змін до інших підсистем. Завдяки розширюваності є можливість використання різних модифікацій існуючих компонентів системи з метою порівняння ефективності їх роботи у початковому стані (для досягнення чистоти експерименту).

Для більшості компонентів середовища моделювання GRID-систем (черга заявок, планувальник та ін.) є несуттєвою ознака, за якою працює дане середовище моделювання в локальному чи розподіленому режимі. Компоненти повинні бути ізольовані відносно підсистеми управління ресурсами та без змін працювати в обох режимах: консольному та графічному. Тобто система повинна мати можливість здійснювати налагодження та первинне тестування в локальному середовищі, а кінцеве тестування – в розподіленому. Така реалізація досягається за рахунок масштабованості.

Також базові компоненти середовища моделювання мають бути максимально продуктивними, щоб забезпечити можливість моделювання роботи GRID-системи при великих навантаженнях.

Тестопридатність GRID-системи полягає в тому, що її архітектура повинна мати гнучкість щодо процедур тестування окремих компонентів та всієї обчислювальної системи. Це реалізує високі вимоги щодо якості програмного коду, а також стабільну та надійну роботу системи в цілому.

4.2 Обґрунтування застосування модульної архітектури на прикладі системи GRASS

Існує ряд пакетів моделювання GRID-систем. Найбільш популярними з них є: Bricks, OptorSim, GridSim [149], SimGrid, GangSim [149]. Також існують середовища емуляції GRID-систем: Grid eXplorer, MicroGrid [149] та ін.

Порівняльний аналіз архітектури середовищ моделювання GRID-систем виявив ряд переваг та недоліків. До недоліків слід віднести: вузьку спеціалізацію цих середовищ, обмеженість архітектури, відсутність доступних відкритих версій. У деяких середовищах при проведенні експериментів користувач стикається з неможливістю проведення експерименту в зв’язку з тим, що середовище має ряд обмежень на кількість одночасно існуючих елементів в GRID-системі. Також від користувача вимагають знань спеціалізованих мов програмування. Ці недоліки знижують ефективність роботи з даними середовищами, тому актуальною задачею є розробка власної архітектури, яка усуває недоліки перерахованих систем. Тому для усунення недоліків доцільно використання модульної архітектури, яка має низку переваг:

- гнучкість налаштування;
- простота модифікації;
- спрощення розробки окремих компонентів;
- спрощення процедур тестування;
- наявність можливостей мати кілька реалізацій одного модуля;
- можливість повторного використання програмного коду;
- «ліниве» завантаження.

Гнучкість налаштування передбачає можливість функціонування GRID-системи з різним складом модулів за кількісними та якісними критеріями. Наприклад, додаток може бути запущено без перекомпіляції в графічному або консольному режимі в залежності від встановлених модулів візуалізації. Аналогічним чином можна запускати локальну або мережну версію програми, інтерактивну версію або версію, яка запускається в консольному режимі та створює звіт. Набір мо-

дулів має бути достатнім для виконання поставленої задачі, тобто якщо будь-який критично необхідний модуль відсутній, то коректна робота GRID-системи даної конфігурації унеможливлюється.

Зміна технологій, розробка нових методів та алгоритмів, а також виправлення помилок у готових реалізаціях призводять до модифікації існуючих GRID-систем. З використанням модульної архітектури для оновлення GRID-системи достатньо замінити версію компонента. При цьому немає необхідності вносити зміни в інші компоненти системи, так як модифікація передбачає цю процедуру.

Використовуючи модульну архітектуру, можна легко реалізувати один з основних принципів програмування – декомпозицію задачі, де кожна з підзадач реалізується окремим модулем. Відповідно, різні модулі можуть бути реалізовані різними розробниками незалежно один від одного. Крім того, у великій GRID-системі важливою є можливість перекомпіляції одного модуля без модифікації інших, що значно зменшує затрати за різними показниками.

Модульна архітектура дозволяє легко проводити тестування окремо взятого модуля та всієї системи. Для цього необхідно створити відповідні тестові модулі, що містять вихідні дані, результати тестування, механізми порівняння отриманих даних з еталонними та конфігурувати систему таким чином, аби вона взаємодіяла з ними. Також можуть бути створені службові тестові модулі, які виконують функцію зменшення надмірності загального програмного коду. Присутня функція порівняння між собою результатів роботи різних версій обраного логічного елементу (алгоритму) GRID-системи для перевірки коректності їх роботи. Маючи декілька модулів, що реалізують одну й ту ж логіку, є можливість по черзі ініціалізувати їх для однакових задач за рахунок зміни складу системи. Крім того, модульна архітектура дозволяє динамічно (в режимі реального часу) вибирати оптимальний (за часом виконання) модуль в залежності від вихідних даних або наданих ресурсів.

Модульна архітектура дозволяє використовувати відомі реалізації в різних проектах, що реалізовані на основі однакових або різнотипних апаратних комплектаціях. Це можуть бути компоненти, які найбільш часто використовуються, на-

приклад системи ведення журналів, логів, статистики, менеджери пам’яті, компоненти графічного інтерфейсу користувача (GUI) та ін.

«Ліниве» завантаження – це істотна перевага модульної системи, яка дозволяє раціонально використовувати оперативну пам’ять обчислювальної системи, а також час запуску програми. Це означає, що модуль не буде задіяним до тих пір, доки не з’явиться нагальна необхідність. Більшість модулів системи виконують це правило, однак інколи виникає необхідність завантаження деяких модулів при завантаженні програмної системи для виконання інших задач, включаючи виклик інших компонентів.

Середовище GRASS – це проект з відкритим вихідним кодом, який реалізує модульне середовище моделювання GRID-систем. Особливостями проекту є: кросплатформність, використання мови програмування C++, бібліотек Qt4, Boost [30,40]. Середовище моделювання GRASS наочно демонструє переваги модульної архітектури, має консольний і графічний інтерфейс користувача. Графічний інтерфейс дозволяє спостерігати за процесом моделювання в режимі реального часу. Зокрема, він демонструє: статистику роботи середовища моделювання, стан черги завдань та обчислювальних ресурсів, виконання завдань ресурсами. Крім інтерактивного спостереження за процесом моделювання, GRASS дозволяє складати звіт для подальшого більш детального його аналізу. Слід зазначити, що у середовищі є плагін, який відповідає за формування log-файлів. Ці файли розрізнені і для формування більш докладного звіту або вибірки необхідно заздалегідь створювати спеціалізовані скрипти до плагінів.

4.3 Особливості реалізації модульної архітектури середовища GRASS

Середовище моделювання GRASS складається з ядра та модулів (плагінів). Ці складові підключаються у динамічному режимі. Кожен модуль виконує свою вузькоспеціалізовану задачу, звертаючись при необхідності до інших модулів системи. Ядро підтримує інструментарій міжмодульної взаємодії, а також забезпечує початкове завантаження (старт) та конфігурацію системи.

Кожен модуль має унікальний строковий ідентифікатор (ім’я та ID). Він також формує набір інтерфейсів взаємодії з ним для інших компонентів системи. Кожен інтерфейс модуля має ім’я та може бути застосований до інших модулів. Таким чином, для отримання будь-якого інтерфейсу модуля необхідно знати його ім’я та ідентифікатор інтерфейсу взаємодії.

Модуль в GRASS – це бібліотека, яка динамічно підключається [29]. В ОС сімейства Microsoft Windows використовується модифікація GRASS – dynamic linked library (dll); в UNIX-подібних ОС – shared objects (so). Ця бібліотека реалізує фабричний метод (factory method), який створює екземпляр класу модуля. Клас модуля реалізує інтерфейс Framework::IPlugin, що забезпечує універсальність роботи з ним, не враховуючи особливості програмної реалізації. Цей інтерфейс включає в себе базові операції, які кожен модуль зобов’язаний надавати системі:

- отримання імені модуля;
- отримання інтерфейсу з заданим ім’ям (ідентифікатором);
- ініціалізація.

Кожен інтерфейс модуля повинен бути успадкований від стандартного класу Framework::IPluginInterface. Він дозволяє уніфікувати всі інтерфейси модулів в системі. Основним запитом від модуля є отримання імені інтерфейсу у вигляді строкового ідентифікатора. Загальна структурна схема взаємодії модулів складається з декількох інтерфейсів та класів, які їх реалізують.

Запит інтерфейсу одного модуля іншому виглядає наступним чином. Інтерфейс модуля, який створює запит, викликає метод ядра getPluginInterface(), передає йому в якості аргументів ім’я модуля та ім’я (ідентифікатор) його інтерфейсу. Ядро перевіряє наявність модуля у відповідності до заданого ім’я та його стан. Якщо модуль не знайдено або вже була невдала спроба завантажити його, повертається помилка. Якщо це перше звернення до модуля, виконується його завантаження в пам’ять, ініціалізація та конфігурація. Якщо завантаження пройшло успішно, ядро викликає метод getPluginInterface() та отримує необхідний інтерфейс модуля по імені (ідентифікатору). У разі успішного запиту ядро виконує перевір-

ку відповідності імені отриманого інтерфейсу та повертає його, при цьому виконується перетворення його до необхідного типу C++.

Отримання доступу до інтерфейсів інших плагінів відбувається через головний інтерфейс ядра Framework::IFramework. Він реалізується класом Framework:: Framework, який здійснює обробку конфігураційного файлу та початкове завантаження плагінів. Інтерфейс Parameters::IParameters, а також його реалізація Parameters::Parameters призначені для передачі модулям, які підключаються, початкових параметрів, які задані у файлі конфігурації.

Для створення гнучкої модульної системи необхідно мати можливість простої зміни набору плагінів та їх параметрів без модифікації ядра або самих бібліотек модулів. Для цього в GRASS використовується система конфігураційних файлів на основі XML. Дані файли описують взаємозв'язок між іменами модулів в системі та назвами файлів їх бібліотек; дозволяють задавати параметри, що передаються модулям при завантаженні, що можуть бути використані для завдання режиму роботи або ініціалізації внутрішніх значень.

4.4 Схема взаємодії модулів в середовищі моделювання GRASS

Імітаційне середовище моделювання GRASS дозволяє: відтворювати всі процеси, що відбуваються в реальній GRID-системі; проводити експерименти за допомогою яких можна перевіряти гіпотези щодо поведінки реальної GRID-система. GRASS є макетом для налагодження модуля розподілу завдань (Algoritm Loader). Цей модуль був розроблений під час практичної реалізації результатів дисертаційних досліджень. У структурі середовища GRASS можна виокремити три складові частини (блоки) [145]: програмне забезпечення завдань, віртуальної організації та обчислювальних ресурсів (рисунок 4.1).

Блок «ПЗ завдань» відповідає за функцію моніторингу стану системи, управління її роботи (додання та видалення завдань з черги, балансування навантаження між наявними ресурсами, вибір методів генерації надходження завдань в систему та ін.).

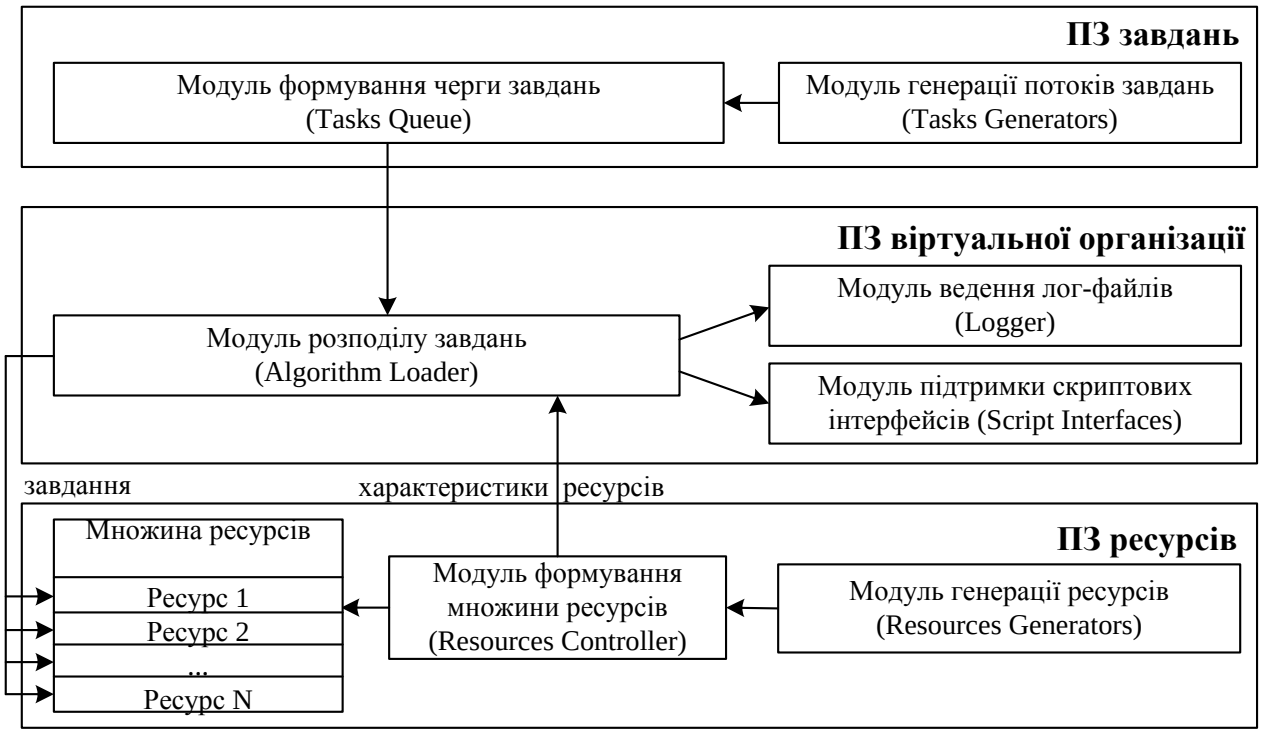

Рисунок 4.1 – Схема взаємодії модулів середовища моделювання GRASS

Блок «ПЗ віртуальної організації» – це центральний компонент середовища, який забезпечує контроль над підключеними обчислювальними ресурсами системи та чергою завдань, що надходять на її вхід, а також збір та обробку статистичної інформації.

Модуль розподілу завдань (Algoritm Loader) є частиною блоку «ПЗ віртуальної організації». Головною задачею даного модулю є завантаження методів розподілу при виникненні в системі подій, що вимагають перерозподілу завдань по ресурсам (додавання нового завдання або обчислювального ресурсу, відключення існуючого ресурсу та ін.).

Блок «ПЗ ресурсів» забезпечує управління обчислювальними ресурсами, збирає та надає іншим компонентам середовища інформацію про стан ресурсів (обсяг вільної фізичної пам’яті, середню завантаженість ресурсу та ін.).

Модуль генерації потоків завдань (Tasks Generators) виконує функцію моделювання вхідного потоку завдань, що надходить до середовища GRASS, відповідно із заданим законом розподілу.

Модуль формування черги завдань (Tasks Queue) виконує функцію моделювання черги завдань в середовищі GRASS, забезпечує зберігання та надання дос-

тупу до неї іншим модулям, збирає статистику знаходження завдань в черзі та їх стан. Завдання можуть бути: згенеровані за допомогою модуля Tasks Generators та завантажені із зовнішнього файлу.

Модуль генерації ресурсів (Resources Generators) виконує функцію моделювання множини обчислювальних ресурсів у відповідності до визначених правил: підключення нових, відключення існуючих (моделювання несправностей).

Модуль формування множини ресурсів (Resources Controller) виконує функцію забезпечення доступу до наявних обчислювальних ресурсів середовища GRASS. Обчислювальні ресурси, аналогічно потоку завдань, можуть бути, як згенеровані за допомогою модуля Resources Generators, так і завантажені із зовнішнього файлу.

Модуль ведення log-файлів (Logger) виконує функцію надання гнучкого та централізованого ведення логів для всіх плагінів середовища моделювання. В середовищі представлено три типи log-файлів: завданя, обчислювальні ресурси та розподіл завдань по ресурсам. Це зроблено для зручності аналізу статистичної інформації. Практична реалізація полягає в використанні модуля скриптових інтерфейсів (Script Interfaces), який дозволяє створювати об’єкти обробки скриптів з налаштованим оточенням для взаємодії з внутрішніми компонентами та модулями середовища GRASS.

4.5 Особливості програмної реалізації конфігураційного файлу plugins.xml в середовищі моделювання GRASS

Файл plugins.xml – це головний конфігураційний файл формату xml, у якому містяться всі параметри налаштування середовища моделювання GRASS. Він знаходиться в каталозі «config» каталогу «Debug» файлової структури середовища GRASS (рисунок 4.2).

У файлі plugins.xml приведені у відповідність імена модулів в системі та назви файлів і бібліотек.

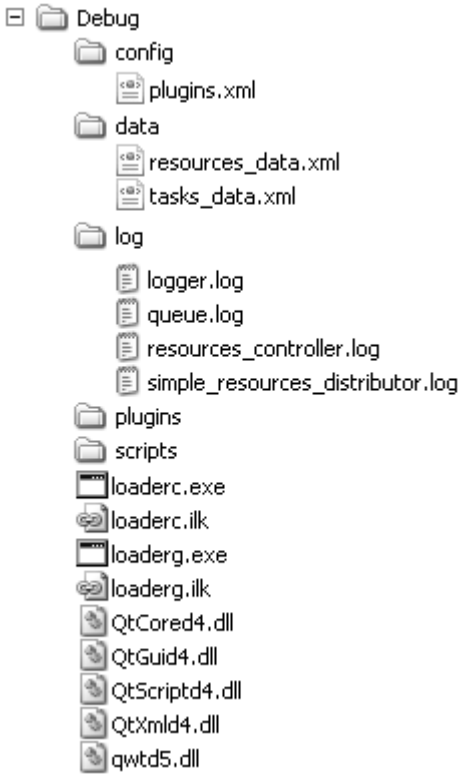

Рисунок 4.2 – Структура каталогу «Debug»

Особливістю структури файла plugins.xml є його розподіл на кілька розділів, в кожному з яких зберігаються посилання на плагіни, що використовуються для функціонування середовища:

- General;
- GUI Management and visualization;
- Resources generation/storing;
- Tasks generation/storing;
- Distribution algorithms & algorithm selector;
- Script plugin interfaces factories;
- Scripting plugins.

Синтаксис мови XML дозволяє описувати та підключати наявні в системі плагіни без специфічних знань адміністратора системи. Наприклад, визначення плагіну «FileName» (приклад 4.1).

```
<plugin name="Name" filename="FileName" loadatstartup="1">
```

Приклад 4.1 – Визначення плагіна «FileName»

У прикладі 4.1 присутні наступні параметри:

- name – ім’я плагіну, яке відповідає виклику функції getName();

- filename – ім’я директорії, де зберігається даний пагін;

- loadatstartup – перемикач, який вказує на необхідність завантаження (початку виконання) плагіну при завантаженні системи («1» / «yes» / «true») або «ручне» завантаження («0» / «no» / «false»).

4.5.1 Реалізація взаємодії компонентів у середовищі

У розділі «General» зберігаються посилання на плагіни для всього середовища моделювання. Наприклад, Logger, CommandsManager, ScriptPlugin та ін. [42]. Плагін Logger – це компонент, який забезпечує гнучкість та централізацію ведення log-файлів для всіх плагінів середовища, дозволяє створювати спеціалізовані повідомлення, на кшталт, Error, Warning, Info, Debug, а також їх сортування за ознаками [21]. Даний плагін надає користувачу можливість налаштування формату виводу та фільтрації повідомлень, забезпечує функцію виділення найбільш важливих з них.

Плагін Commands Manager – це компонент, який керує обробкою команд в середовищі GRASS: ввід команд в консолі, або дія, яку може виконати користувач за допомогою графічного інтерфейсу (відкриття файлу або зміна налаштувань).

Плагін ScriptPlugin – це компонент, який відповідає за формування та запуск скриптів, які необхідні для формування статистики роботи деяких компонентів середовища. Він виконує функції збереження моделей ресурсів, завдань, та є парсером log-файлів середовища.

4.5.2 Реалізація візуалізації статистики роботи середовища моделювання

У розділі «GUI Management and visualization» зберігаються посилання на сукупність плагінів, які відповідають за візуалізацію статистики роботи середовища моделювання GRASS. Також відображається стан черг завдань і ресурсів, демон-

струється виконання завдань обчислювальними ресурсами.

Група плагінів GUI Support Plugins – це компоненти, що забезпечують підтримку різних інструментів графічного інтерфейсу. Наприклад, QueueAdaptor, GUIObjectsMonitor, GUIStatisticsMonitor, GUIPluginsInfo, UIManager та ін.

За допомогою плагіну QueueAdaptor здійснюється візуалізація статистичної інформації у вигляді графіків (параметр refresh_time відповідає за період оновлення (вказується в мс)). В прикладі 4.2 наведено засіб визначення даного плагіна в файлі plugins.xml.

```
     <plugin name = "QueueAdaptor" filename = "queue_adaptor"
loadatstartup = "no">
     <parameter refresh_time = "1000" />
     </plugin>
```

Приклад 4.2 – Завантаження плагіна QueueAdaptor

Приклад 4.2 показує, що даний плагін не завантажується при старті системи, а період оновлення інформації становить 1000 мс, тобто 1 с.

Плагіном GUIObjectsMonitor виконується відображення списку обчислювальних ресурсів та завдань, стан зайнятості ресурсів. Якщо параметр use_colors=«yes», це означає, що всі завдання, які виконуються в даний момент часу, будуть виділені різними кольорами. Це спрощує контроль за виконанням певних завдань.

Плагін GUIStatisticsMonitor – це компонент, який відповідає за виведення різної статистичної інформації у вигляді таблиці (параметр refresh_time відповідає за період оновлення інформації та визначається в мс).

Плагіном GUIPluginsInfo виконується відображення спеціальної таблиці з інформацією про стан середовища, наприклад, інформація про плагіни, які виконуються в даний момент часу.

Плагін UIManager – це компонент, який відповідає за збереження та відновлення стану GUI при завантаженні середовища. Параметр preference_file визначає шлях до файлу, в якому зберігається інформація про налаштування головного вікна середовища.

4.5.3 Моделювання обчислювальних ресурсів

Розділ «Resources generation/storing» відповідає за моделювання обчислювальних ресурсів в середовищі GRASS. Модель ресурсів описує набір деяких окремих обчислювальних станцій або кластерів з різними параметрами. Наявність функціональності моделювання ресурсів дозволяє динамічно генерувати ресурси в системі, а також моделювати їх поведінку (завершення роботи у разі виникнення несправності, моделювання часу відновлення, поновлення роботи ресурсу). Це реалізовано за допомогою двох плагінів: SimpleResourcesManager та SimpleResourcesGenerator (приклад 4.3).

```
     <plugin name = "SimpleResourcesManager" filename =
"simple_resources_manager" loadatstartup = "yes" >
     <parameter input_file="./data/resources_data.xml"/>
     </plugin>
     <plugin name = "SimpleResourcesGenerator" filename =
"simple_resources_generator" loadatstartup = "no" >
     <parameter model_file="./data/resources_model.xml"/>
     </plugin>
```

Приклад 4.3 – Плагіни SimpleResourcesManager та SimpleResourcesGenerator

Плагін SimpleResourcesManager використовується для додання до середовища моделювання реальних (GRID-система) обчислювальних ресурсів з конфігураційного файлу. Дані конфігураційного файлу зберігаються в форматі .xml за адресою, яка вказана в параметрі input_file – resources_data.xml. Якщо користувачу необхідно провести повторний експеримент з даними, які були згенеровано раніше, то в даному плагіні параметр loadatstartup задається значенням «yes». У прикладі 4.4 показана структура конфігураційного файлу resources_data.xml, який містить інформацію про два обчислювальні ресурси (name1 та name2).

```
<resources>
     <resource name="name1" time="480">
          <parameter cpu_arch="AMD"/>
          <parameter cpu_count="1"/>
          <parameter cpu_speed="1000"/>
          <parameter os="Linux 2.6.2"/>
          <parameter physical_memory="16"/>
     </resource>
```

```
    <resource name="name2" time="115">
        <parameter cpu_arch="Intel"/>
        <parameter cpu_count="2"/>
        <parameter cpu_speed="1300"/>
        <parameter os="Linux 2.4.2"/>
        <parameter physical_memory="32"/>
    </resource>
</resources>
```

Приклад 4.4 – Файл resources_data.xml

Тег «resources» є кореневим і містить всі дані про обчислювальні ресурси, які необхідні для ініціалізації процесу моделювання. Усередині кореневого тегу розташовані кілька тегів «resource», кожен з яких описує конкретний ресурс. Тег «parameter» описує параметри ресурсу: архітектуру процесора (cpu_arch), кількість процесорів (cpu_count), швидкість процесора (cpu_speed), операційну систему (os) та обсяг фізичної пам'яті (physical_memory). Набір параметрів моделювання для ресурсів може бути розширено.

Якщо параметр loadatstartup плагіна SimpleResourcesGenerator задається зі значенням «yes», то буде виконана ініціалізація генератора ресурсів, конфігурація якого налаштовується у файлі resources_model.xml, а шлях до нього вказується в параметрі model_file.

В середовищі існує п'ять різновидів генераторів:

- Constant – генератор константних значень;
- Uniform – генератор рівномірного розподілу;
- Normal – генератор нормального розподілу;
- Exponential – генератор значень з експоненціальним розподілом;
- генератори строкових значень зі списку:
    а) лінійний – «ListLinear»;
    б) випадковий – «ListRandom».

Константний генератор – це найпростіший вид генератору, який завжди формує константне значення. Це значення задається у конфігурації (приклад 4.5).

```
<priority generator=«Constant»>
    <double value=«100»/>
</priority>
```

Приклад 4.5 – Визначення параметру за допомогою генератора Constant

В даному прикладі параметр «priority» задається за допомогою константного генератора зі значенням 100. Це значення задається параметром генератору «value» типу «double» (знаходиться в тегі відповідного до типу).

Параметр «timeout» (приклад 4.6) задається за допомогою генератора рівномірного розподілу.

```
<timeout generator=«Uniform»>
     <double a=«1500»/>
     <double b=«2300»/>
</timeout>
```

Приклад 4.6 – Визначення параметру за допомогою генератора Uniform

Параметри генератора «a» та «b» типу «double» – це ліва та права межа інтервалу, на якому випадкова величина отримує свої значення. В даному прикладі випадкова величина буде генеруватися в межах від 1500 до 2300.

Параметр «Memory» (приклад 4.7) задається за допомогою генератора, який розподілений по експоненціальному закону.

```
<Memory generator=«Normal»>
     <double mean=«1000000»/>
     <double sigma=«5000»/>
</Memory>
```

Приклад 4.7 – Визначення параметру за допомогою генератора Normal

Параметр «mean» – це середнє значення (математичне очікування) випадкової величини, а «sigma» – дисперсія. В даному прикладі величина буде генеруватися за нормальним законом з параметрами математичного очікування 1000000 та дисперсією 5000.

Параметр «delay» (прикладі 4.8) формується генератором випадкових чисел, який розподілений по експоненціальному закону.

```
<delay generator=«Exponential»>
     <double gamma=«0.005»/>
</delay>
```

Приклад 4.8 – Визначення параметра за допомогою генератора Exponential

Параметр «gamma» є основним параметром експоненціального розподілу.

Це середнє число заяв за секунду. В даному прикладі випадкова величина буде генеруватися з параметром 0,005.

У середовищі моделювання GRASS існує два види генераторів строкових значень зі списку: лінійний (ListLinear) та випадковий (ListRandom). В якості параметру генератора задається список строкових значень (приклад 4.9), значення параметра вибирається з цього списку лінійно (послідовно) або випадково. Дані списки можуть бути доповнені.

```
<os generator=«ListLinear»>
     <parameter values=«stringlist»>
          <x value=«Linux 2.6.2»/>
          <x value=«Windows 2000»/>
          <x value=«Windows XP»/>
     </parameter>
</os>
<os generator=«ListRandom»>
     <parameter values=«stringlist»>
          <x value=«Linux 2.6.2»/>
          <x value=«Windows 2000»/>
          <x value=«Windows XP»/>
     </parameter>
</os>
```

Приклад 4.9 – Використання генераторів строкових значень

4.5.4 Моделювання вхідних завдань

Розділ «Tasks generation/storing» відповідає за моделювання потоків завдань, які надходять в середовище з різною інтенсивністю та параметрами. Наприклад, Queue, SimpleTasksManager та SimpleTasksGenerator.

Плагін Queue – це компонент, який моделює чергу завдань в середовищі, а також відповідає за організацію зберігання та надання доступу до черги іншим модулям середовища (приклад 4.10).

```
<plugin name = "Queue" filename = "queue" >
      <parameter task_parameters_customizer = "stringlist">
         <string value = "AlgorithmLoader" />
         <string value = "ResourcesController" />
      </parameter>
</plugin>
```

Приклад 4.10 – Плагін Queue

Функціональність формування завдань реалізована за допомогою двох плагінів: SimpleTasksManager та SimpleTasksGenerator (приклад 4.11).

```
     <plugin name = "SimpleTasksManager" filename = "simple_tasks_manager"
loadatstartup = "yes" >
           <parameter input_file="./data/tasks_data.xml"/>
     </plugin>
     <plugin name = "SimpleTasksGenerator" filename =
"simple_tasks_generator" loadatstartup = "no" >
           <parameter mode="PUSH_AND_WRITE"/>
           <parameter model_file="./data/tasks_model.xml"/>
           <parameter output_file="./last_data/tasks_data.xml"/>
     </plugin>
```

Приклад 4.11 – Плагіни SimpleTasksManager та SimpleTasksGenerator

Плагін SimpleTasksManager – це компонент, який призначений для генерації потоку завдань на основі вхідного конфігураційного файлу (приклад 4.11). Дані цього файлу зберігаються в форматі .xml за адресою, яка вказана в параметрі input_file – tasks_data.xml. Якщо користувачу необхідно провести повторний експеримент з даними, яки були згенеровані раніше, то в даному плагіні параметр loadatstartup задається зі значенням «yes». У прикладі 4.12 наведена структура конфігураційного файлу tasks_data.xml.

```
<tasks>
     <task time="1000">
          <parameter cpu_arch="AMD"/>
          <parameter cpu_count="1"/>
          <parameter cpu_speed="1000"/>
          <parameter os="Linux 2.6.2"/>
          <parameter physical_memory="16"/>
          <parameter priority ="18"/>
          <parameter timeout="180000"/>
     </task>

     <task time="2000">
          <parameter cpu_arch="Intel"/>
          <parameter cpu_count="2"/>
          <parameter cpu_speed="1300"/>
          <parameter os="Linux 2.4.2"/>
          <parameter physical_memory="32"/>
          <parameter priority ="20"/>
          <parameter timeout="60000"/>
     </task>
     </tasks>
```

Приклад 4.12 – Структура файлу tasks_data.xml

В даному прикладі описані два завдання: перше завдання надійде до системи з затримкою в 1000 мс, а друге – через 2000 мс після першого. Вимоги, що ставляться завданням до ресурсів, а також параметри завдання задаються в тегах «parameter».

Тег «tasks» – є кореневим. Він містить всі дані для моделювання, які описують кожне окреме завдання. Параметр тега «time» визначає затримку надходження наступного завдання відносно попереднього (задається в мс).

Потім відбувається, власне, процес моделювання затримок надходження відповідних завдань та додання їх до середовища. За допомогою параметру QTimer планується виклик методу timerTick(), з деякою затримкою, в якому створюється новий об'єкт завдання класу Task::Task. Він додається до черги завдань.

Плагін SimpleTasksGenerator – це компонент, який відповідає за створення завдань для середовища. Плагін SimpleTasksGenerator містить п'ять генераторів (ідентичних плагіну SimpleResourcesGenerator). Генератор завдань має кілька режимів роботи:

- PUSH_AND_WRITE – генерує завдання до середовища та записує їх у вихідному файлі для подальшого використання (повторного експерименту);

- PUSH_ONLY – генерує завдання, ставить їх в чергу, але не записує їх у вихідному файлі (режим моделювання в реальному часі);

- WRITE_ONLY – генерує завдання тільки для запису до файлу (тестовий режим для перевірки правильності роботи середовища).

4.5.5 Реалізація методів розподілу завдань

Розділ «Distribution algorithms & algorithm selector» реалізує методи розподілу завдань на обчислювальні ресурси в середовищі моделювання GRASS. Ці методи представлені у вигляді плагіну Algorithm Loader.

Плагін Algorithm Loader – це компонент, який відповідає за завантаження методів розподілу при виникненні в системі подій, що вимагають перерозподілу

завдань на ресурси (наприклад, додавання нового завдання або ресурсу, відключення існуючого ресурсу та ін.) (приклад 4.13).

```
     <plugin name = "AlgorithmLoader" filename = "algorithm_loader"
loadatstartup = "yes">
           <parameter tasks_count="10"/>
           <parameter algorithm="BackFillDistributionAlgorithm" />
           <parameter strict="0"/>
      </plugin>
     <plugin name = "SmartDistributionAlgorithm" filename =
"smart_distribution_algorithm" />
     <plugin name = "FifoDistributionAlgorithm" filename =
"fifo_distribution_algorithm" />
     <plugin name = "LifoDistributionAlgorithm" filename =
"lifo_distribution_algorithm" />
     <plugin name = "SimplexMetod" filename = "simplex_metod" />
     <plugin name = "PriorityDistributionAlgorithm" filename =
"priority_distribution_algorithm" />
     <plugin name = "BackFillDistributionAlgorithm" filename =
"backfill_distribution_algorithm" />
     <plugin name = "BackFillDistributionAlgorithm_mod" filename =
"backfill_distribution_algorithm_mod" />
```

Приклад 4.13 – Плагін Algorithm Loader

Плагін Algorithm Loader містить три параметри: tasks_count, algorithm, strict.

Параметр tasks_count визначає гранична кількість завдань у черзі: коли кількість завдань в черзі перевищує задане адміністратором системи значення, запускається метод планування завдань, який вказаний в параметрі algorithm.

Параметр algorithm визначає метод розподілу завдань на ресурси. В середовищі реалізовані наступні методи:

- FifoDistributionAlgorithm – метод розподілу завдань, який заснований на принципі FIFO;

- LifoDistributionAlgorithm – метод розподілу завдань, який заснований на принципі LIFO;

- SmartDistributionAlgorithm – метод розподілу завдань по вивільненню ресурсів (після того як обчислювальний ресурс звільняється, його займає перше завдання, яке має задовольняє відповідним вимогам);

- PriorityDistributionAlgorithm – метод розподілу завдань за пріоритетами (першими будуть оброблені завдання з більш високим пріоритетом);

- SimplexMetod – метод розподілу завдань, який заснований на вирішенні задачі лінійного програмування;

- BackFillDistributionAlgorithm – метод розподілу завдань на основі методу зворотного заповнення BackFill;

- BackFillDistributionAlgorithm_mod – модифікація консервативного резервування для методу BackFill з урахуванням пропускної здатності та затримки передачі пакетів даних в комп'ютерній мережі.

Третім з регуляторів експерименту є змінна strict, яка дозволяє проводити експеримент у двох напрямах: з урахуванням розпаралелювання завдання на різні ресурси та без нього (в залежності від коефіцієнта зв'язаності).

4.6 Програмна реалізація узагальненого критерію оцінки завдань

Узагальнений критерій оцінки завдання розраховується в модулі згортки кортежу [34]. Його функціональність є складовою частиною інформаційної технології розподілу завдань. Для отримання множини ресурсів по кожному з вхідних завдань необхідно виконати дії, згруповані за наступними етапами.

Етап 1: завантаження вихідних даних. Вихідними даними для даного модуля є два файли із середовища GRASS: файл ресурсів (resources_data.xml) та множину завдань (tasks_data.xml). Постачальник завдань задає: вимоги до обчислювальних ресурсів, вагові коефіцієнти для кількісних параметрів завдання (рисунок 4.3).

Етап 2: задання часткових критеріїв по кожному завданню. Вагові коефіцієнти по кожному кількісному параметру завдання необхідні для визначення часткових критеріїв, на підставі яких проводиться розрахунок узагальненого критерію оцінки завдання.

Етап 3: отримання результатів згортки кортежу. На виході модулю згортки кортежу для кожного завдання формується множина обчислювальних ресурсів, які задовольняють виконанню завдань Ω_{Z_i} (4.1) :

$$\Omega_{Z_i} = \left(R_j^r \rightarrow part_j\right), \text{ при } j = \overline{1, K}, \qquad (4.1)$$

де $\{R_j^r\}$ – підмножина обчислювальних ресурсів, відібраних по кожному завданню на підставі узагальненого критерію оцінки завдання;

$\{part_j\}$ – підмножина коефіцієнтів використання обчислювальних ресурсів. Вона вказує на задіяну частину обчислювального ресурсу при виконанні завдання (рисунок 4.5).

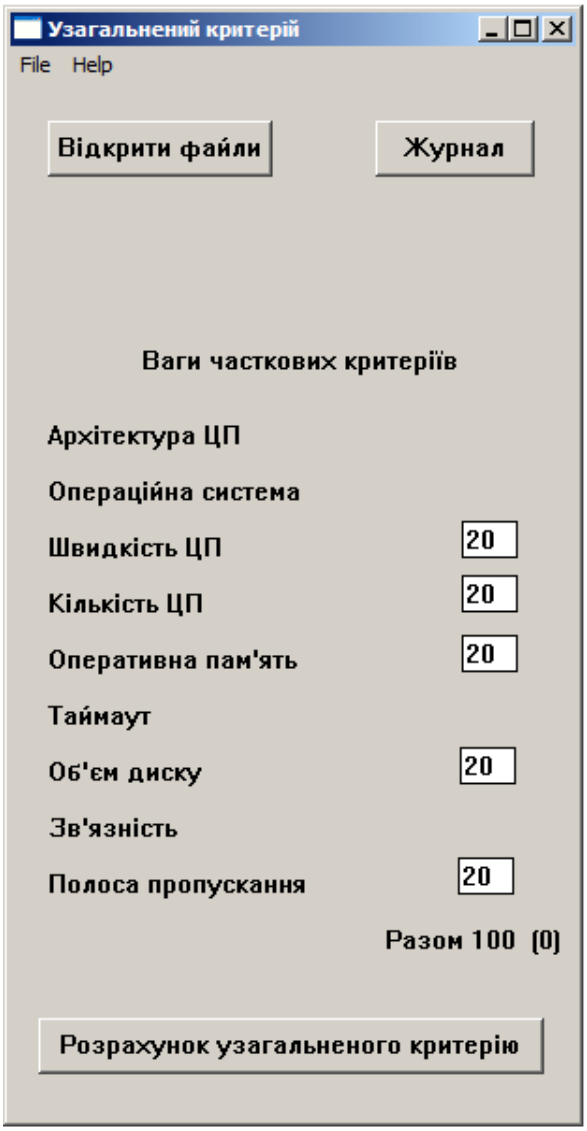

Рисунок 4.3 – Вікно задання часткових критеріїв за кожним завданням

Етап 4: передача множини Ω_{Z_i} на вхід модуля аналізу зв’язності. Модуль згортки кортежу (рисунок 4.4) виконує розрахунок узагальненого критерію для

завдання. Наприклад, для виконання завдання Z_1 підходять обчислювальні ресурси R_3, R_5, R_7 та R_9. Коефіцієнт використання ресурсу R_3 становить 0,52. Це означає що незадіяні обчислювальні потужності (1,0 – 0,52 = 0,48) можуть бути використаними для виконання інших завдань, тобто кожен з обраних ресурсів R_j має достатній запас обчислювальної потужності, що дозволяє виконувати на кожному з них декілька завдань одночасно (паралельно).

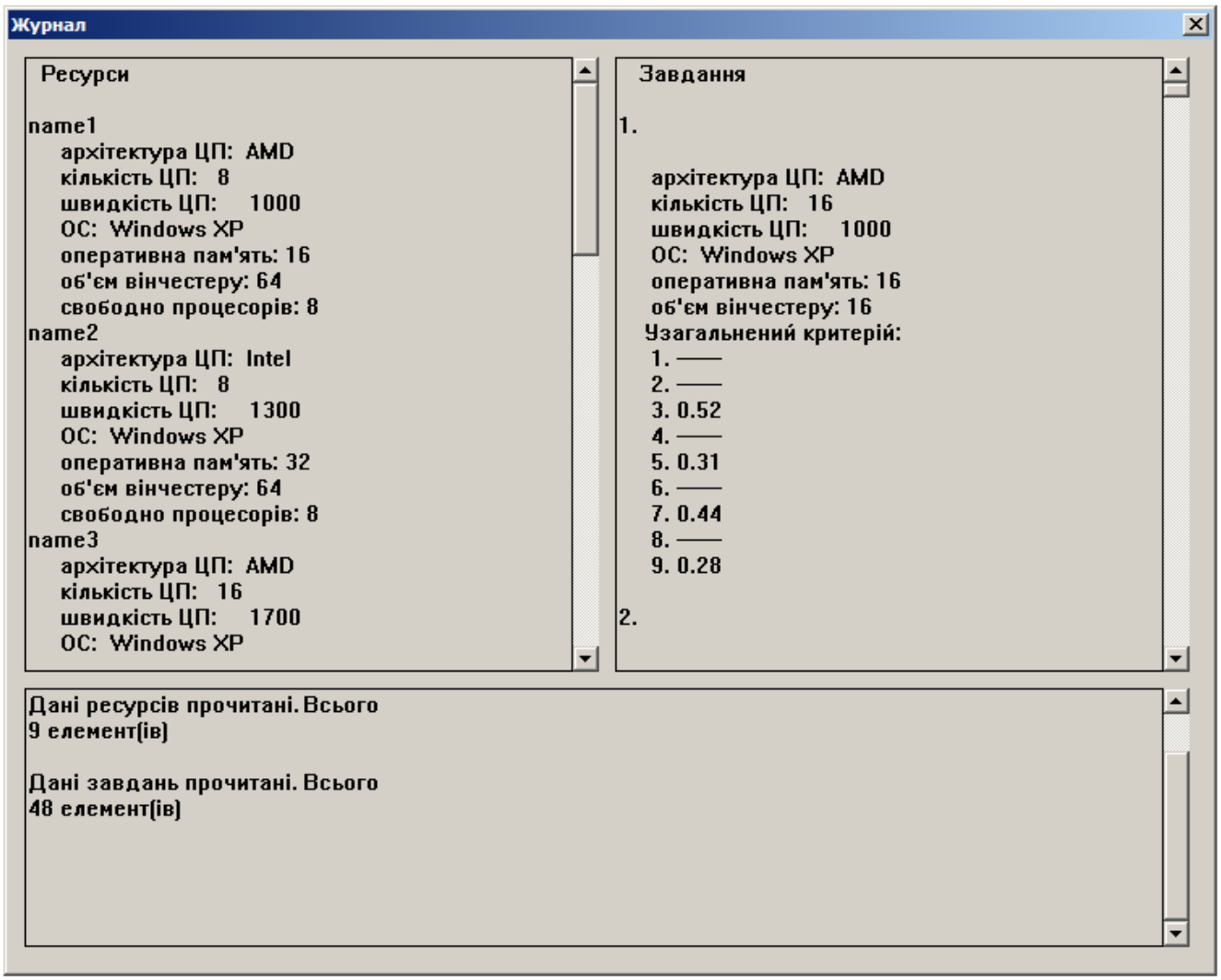

Рисунок 4.4 – Вікно журналу для розрахунку узагальненого критерію завдання

Отже, при розподілі завдань на обчислювальних ресурсах, адміністратору системи необхідно враховувати коефіцієнт використання обчислювальних ресурсів завданнями. Оперування цими коефіцієнтами дозволяє ефективно використовувати обчислювальні ресурси (мінімізація часу їх незадіяності в системі).

В результаті роботи даного модуля формується множина, яка задовольняє

вимогам для виконання завдань (враховуючи вимоги постачальника). При виконанні завдання є вірогідність наступної ситуації: обчислювальна система не задовольняє вимогам постачальника; задачі в завданні – не пов’язані між собою. У такому випадку їх необхідно розподілити по різним вільним обчислювальним ресурсам – це гарантує виконання завдання в цілому.

Перевагою запропонованого модуля є можливість створення пулу обчислювальних ресурсів, які придатні для виконання задач із завдання з урахуванням їх зв’язності. Використання цього модуля при розробки плану розподілу враховує інтенсивність та обсяг потоків даних між задачами у завданні, що підвищує ефективність використання обчислювальних ресурсів GRID-системи за рахунок скорочення часу незадіянності обчислювальних ресурсів в розподілених гетерогенних системах.

4.7 Практичне застосування інформаційної технології розподілу завдань на обчислювальні ресурси в середовищі моделювання GRASS

Враховуючи виконані налаштування конфігураційного файлу plugins.xml (підрозділ 4.5), виконується ініціалізація (завантаження) середовища моделювання GRASS [37]. Після цього є можливість спостерігати за його роботою за допомогою додатку візуалізації (рисунок 4.5), який містить кілька вікон (активні або пасивні).

На рисунку 4.5 в додатку візуалізації відкрито два активних вікна: у Tasks Info відображена черга завдань, які надходять до системи, із зазначенням вимог постачальників завдань; а Resources Info відображає перелік наявних в GRASS середовищі обчислювальних ресурсів. У ході надходження завдань до GRASS середовища відбувається додання їх до черги, яка відображається в вікні Tasks Info. Аналогічно механізму додання завдань, до GRASS середовища можна додавати обчислювальні ресурси. У цьому випадку вони будуть відображатися в вікні додатку Resources Info.

Під час роботи додатку візуалізації, задачі кожного завдання мають один

колір, що дозволяє в режимі реального часу спостерігати за «місцем виконання» (обчислювальний ресурс) конкретної задачі. Це реалізує концепцію дружелюбного інтерфейсу програмного забезпечення.

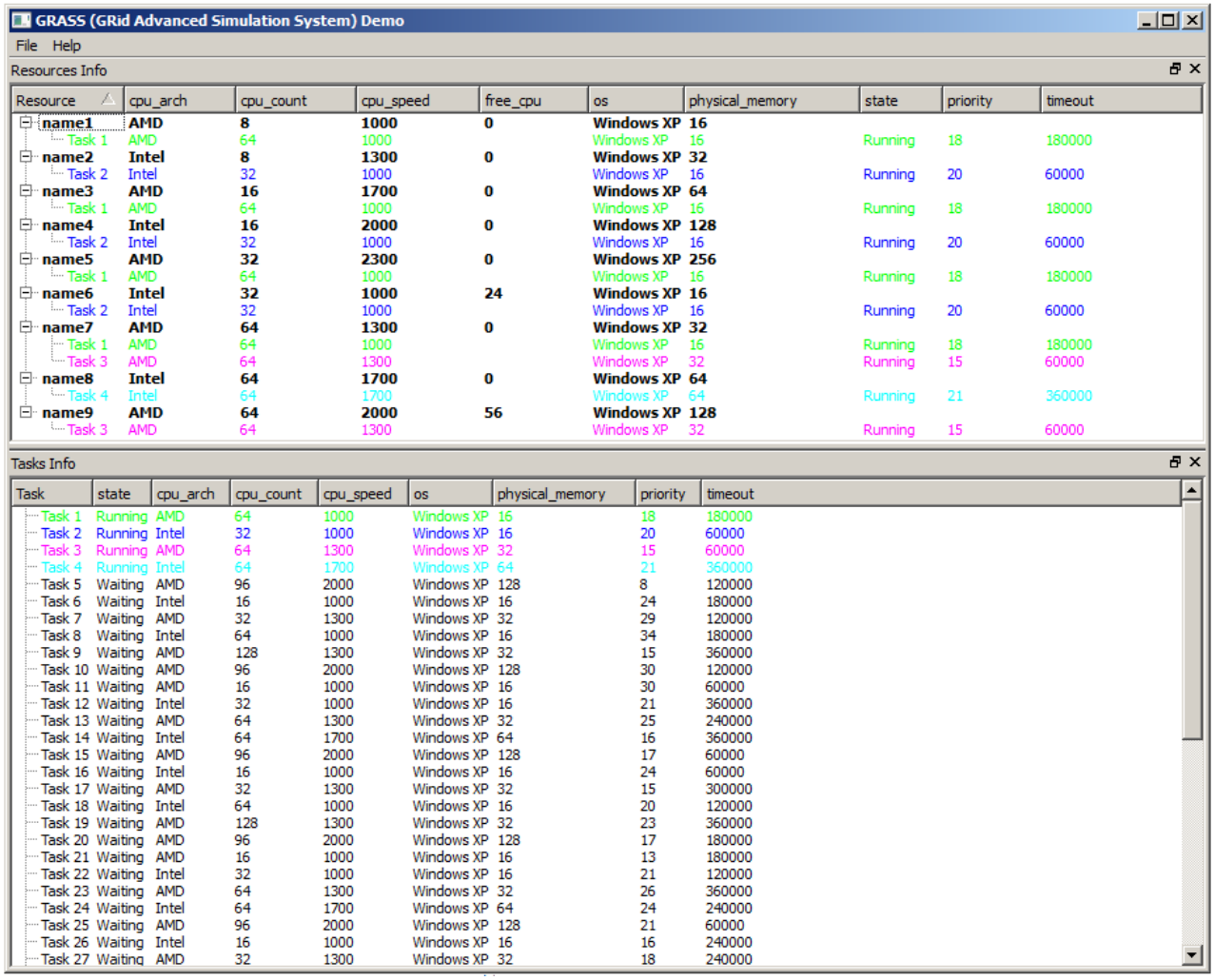

Рисунок 4.5 – Візуалізація черг та процесу виконання задач із завдань в GRASS

Після завершення виконання завдання, обчислювальний ресурс звільняється, а завдання залишає чергу. Під час роботи середовища GRASS здійснюється моніторинг, власне, середовища, а також графічне відображення процесу моделювання в реальному часі (рисунок 4.6).

Отримана статистична інформація в результаті моніторингу може бути використана для аналізу вибірок в залежності від заданих критеріїв. В середовищі GRASS для роботи зі статистичною інформацією були введені наступні змінні:

- Pushed – кількість завдань, які були додані до черги під час моделювання

(при кожному доданні завдання, інкрементується значення лічильника нових завдань);

- Removed – кількість завдань, які були видалені з черги під час моделювання (тобто були виконані). При кожному виконанні завдання, інкрементується значення лічильника виконаних завдань;

- Ts – час роботи GRASS середовища. Змінна приймає значення в залежності від часу виконання всіх завдань (до моменту примусової зупинки роботи додатку моделювання).

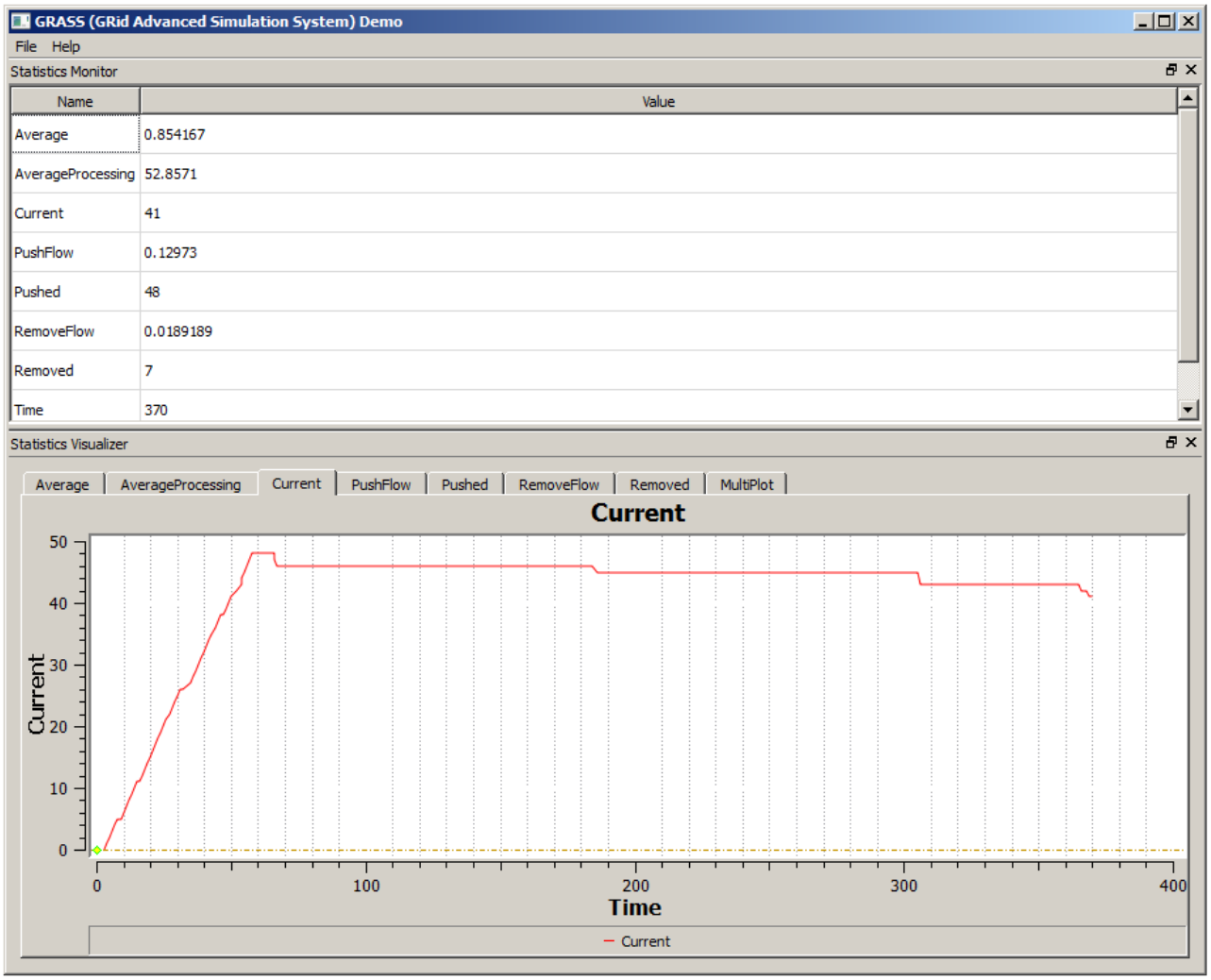

Рисунок 4.6 – Візуалізація моніторингової системи середовища GRASS

На підставі вище вказаних параметрів можна побудувати ряд графіків, які відображають роботу GRASS середовища за наступними показниками.

Графік PushFlow показує щільність потоку завдань, які надходять до черги.

Значення даного параметра визначається наступним чином (4.2):

$$PushFlow = \frac{Pushed}{Ts}. \quad (4.2)$$

Графік RemoveFlow показує щільність потоку завдань, які покидають чергу (були виконані). Значення даного параметра визначається наступним чином (4.3):

$$RemoveFlow = \frac{Removed}{Ts}. \quad (4.3)$$

Графік Average відображає значення поточної завантаженості черги. Значення даного параметра визначається наступним чином (4.4):

$$Averade = \frac{(Pushed - Removed)}{Pushed}. \quad (4.4)$$

Графік Current відображає значення поточної кількості завдань в черзі. Значення даного параметра визначається наступним чином (4.5):

$$Current = Pushed - Removed. \quad (4.5)$$

Графік AverageProcessing відображає значення середнього часу обробки одного завдання. Значення даного параметра визначається наступним чином (4.6):

$$Averade\ Processing = \frac{Ts}{Removed}. \quad (4.6)$$

Всі вище наведені графіки будуються в реальному часі. Перехід між графіками здійснюється за допомогою закладок вікна Statistic Visualizer додатку моделювання GRASS середовища. У вікні Statistic Monitor (рисунок 4.6) показані змінні, які набувають значень в процесі моделювання. Інформація про ці дані накопичується в log-файлах.

4.8 Класифікація завдань за обсягами даних, що були передані

Завдання, що надходять в GRID-систему на виконання, характеризуються наступними параметрами: обсяг вхідних і/або вихідних даних, час завантаження та виконання. В роботі пропонується введення вагових коефіцієнтів для кожного завдання (під час постановки експериментів). Вони необхідні для оцінки вхідних даних та результатів виконання завдання (вихідних даних).

Нехай D_{in} – це вхідний ваговий коефіцієнт, який визначається виразом 4.7:

$$D_{in} = \begin{cases} 1, t_c \geq t_з \\ 0, t_c < t_з \end{cases}, \tag{4.7}$$

де t_c – вхідне навантаження (інтервал часу від початку завантаження даних до початку виконання завдання);

$t_з$ – час виконання завдання.

Вхідне навантаження t_c завдання визначається виразом 4.8:

$$t_c = \frac{V_c}{C_{in}} \tag{4.8}$$

де V_c – вхідний обсяг даних завдання для завантаження;

C_{in} – вхідна пропускна здатність каналу зв’язку.

Аналогічно визначається вихідний коефіцієнт (4.9) та вихідне навантаження (4.10) завдання:

$$D_{out} = \begin{cases} 1, t_ф \geq t_з \\ 0, t_ф < t_з \end{cases}, \tag{4.9}$$

$$t_ф = \frac{V_ф}{C_{out}} \tag{4.10}$$

де $t_ф$ – вихідне навантаження (час від закінчення завдання до вивантаження даних);

$V_ф$ – вихідний обсяг даних завдання для вивантаження;

C_{out} – вихідна пропускна здатність каналу зв'язку.

Тоді на підставі виразів 4.7-4.10 всі завдання, що надходять до GRID-системи, можна умовно поділити на чотири класи:

- великий обсяг вхідних даних $D_{in} = 1, t_c \geq t_з$ та великий обсяг вихідних даних $D_{out} = 1, t_ф \geq t_з$;

- малий обсяг вхідних даних $D_{in} = 0, t_c < t_з$ та великий обсяг вихідних даних $D_{out} = 1, t_ф \geq t_з$;

- великий обсяг вхідних даних $D_{in} = 1, t_c \geq t_з$ та малий обсяг вихідних даних $D_{out} = 0, t_ф < t_з$;

- малий обсяг вхідних даних $D_{in} = 0, t_c < t_з$ та малий обсяг вихідних даних $D_{out} = 0, t_ф < t_з$.

Наприклад, для рендерінгу відеофайлу тривалістю 15 хвилин з наступними вихідними даними: вхідна сцена має ємність 300 МБ, розмір 1 кадру 0,5 Мб, пропускна здатність каналу передачі даних 100 Мб/с, 1 секунда відеофайлу складається з 25 кадрів – виконується наступна послідовність дій.

Крок 1: 25 кадрів за секунду * 0,5 Мб = 12.5 Мб/с – 1 секунда відеофайлу.

Крок 2: 12,5 Мб/с * 60 с * 15 хв = 11250 Мб – розмір відеофайлу на виході.

Крок 3: 300 МБ * 8 біт = 2400 Мбіт – обсяг вхідного файлу (сцени).

Крок 4: 2400 Мбіт / 100 Мбіт / сек = 24 сек – час, необхідний для передачі вхідного файлу на обчислювальні ресурси. Де-факто, це розрахункова величина часу, тому що реальна пропускна здатність каналу передачі даних залежить від завантаження каналу передачею інших даних. Для узагальнення візьмемо фактично гарантовану пропускну здатність каналу передачі даних 70%. Враховуючи цей факт, вхідна сцена буде передаватися на обчислювальний ресурс протягом ~ 40 с.

Крок 5: 11250 Мб * 8 біт = 90000 Мбіт – обсяг вихідного відеофайлу.

Крок 6: 90000 Мбіт / 100 Мбіт / сек = 900 сек – час, який необхідно витратити для передачі вихідного файлу з обчислювальних ресурсів користувачу. На-

справді, ~ 1500 с (25 хв) – це час, який витрачається на передачу результату 15-хвилинного відеофайлу.

У таблиці 4.1 показана залежність результату рендерінгу однієї секунди відеофайлу від величини його кадру для різних форматів якості, а в таблиці 4.2 показана залежність розміру відеопотоку для різних частот проходження кадрів від часу рендерінгу відеофайлу (без стиснення).

Таблиця 4.1 – Залежність розміру 1 секунди відеофайлу від величини кадру

Розмір кадру, Кб		Частота проходження кадрів			
		25	30	50	60
171	Розмір секунди відео, Кб	4275	5130	8550	10260
381		9525	11430	19050	22860
120		3000	3600	6000	7200
246		6150	73820	12300	14760
890		22250	26700	44500	53400
1008		25200	30240	50400	60480
2058		51450	61740	102900	123480
7933		198325	237990	396650	475980
33822		845550	1014660	1691100	2029320

Таблиця 4.2 – Залежність розміру відеопотоку для різних частот проходження кадрів від часу

Розмір кадру, Кб		Частота проходження кадрів			
		25	30	50	60
		15 хвилин	30 хвилин	45 хвилин	60 хвилин
171	Розмір відео, Мб	3757,324	9017,578	22543,95	36070,31
381		8371,582	20091,8	50229,49	80367,19
120		2636,719	6328,125	15820,31	25312,5
246		5405,273	12972,66	3231,64	51890,63
890		19555,66	46933,59	117334	187734,4
1008		22148,44	53156,25	132890,6	212625
2058		45219,73	108527,3	271318,4	434109,4
7933		174309,1	418341,8	1045854	1673367
33822		743159,2	1783582	4458955	7134328

Результати, які показані в таблицях 4.1. та 4.2, дозволяють зробити припущення, що зі збільшенням розміру кадру та часу рендерінга відеофайлу, збільшується розмір вихідного файлу. Даний відеофайл постачальник завдання повинен отримати назад, однак доки він не буде переданий, обчислювальний ресурс не може бути використаний GRID-системою для інших завдань. Існують ситуації, коли передача вихідних даних займає більше часу, ніж обробка вхідних даних. Отже, за наявності таких завдань в GRID-системі, слід враховувати пропускну здатність каналів зв'язку (гетерогенність комп'ютерної мережі), що дозволяє минати вузькі місця в мережі GRID-системи. Це підвищує ефективність використання таких систем.

4.9 Аналіз результатів дослідження застосування запропонованої інформаційної технології

В імітаційне середовище моделювання GRASS були завантажені пули з реальної GRID-системи, які використовувалися під час обробки радіоастрономічних даних на GRID-кластері в РІ НАНУ [39,147]: пул завдань та пул обчислювальних ресурсів. Пул завдань включає в себе завдання, кожне з яких описано кортежем (3.2), пул ресурсів сформований відповідно до кортежу (3.3). Використовуючи різні методи розподілу (3.1), які були реалізовані в імітаційному середовищі моделювання GRASS, було проведено моделювання та отримані відповідні результати. Рішення про вибір методу розподілу з мінімальним виконанням пулу завдань та мінімальним простоєм обчислювальних ресурсів GRID-системи приймається на основі порівняльного аналізу.

На рисунку 4.7 показано час виконання фіксованого пулу завдань для всіх методів, які реалізовані в середовищі GRASS (в модулі Algorithm Loader), а на рисунку 4.8 – показник незадіянності обчислювальних ресурсів по кожному із запропонованих методів.

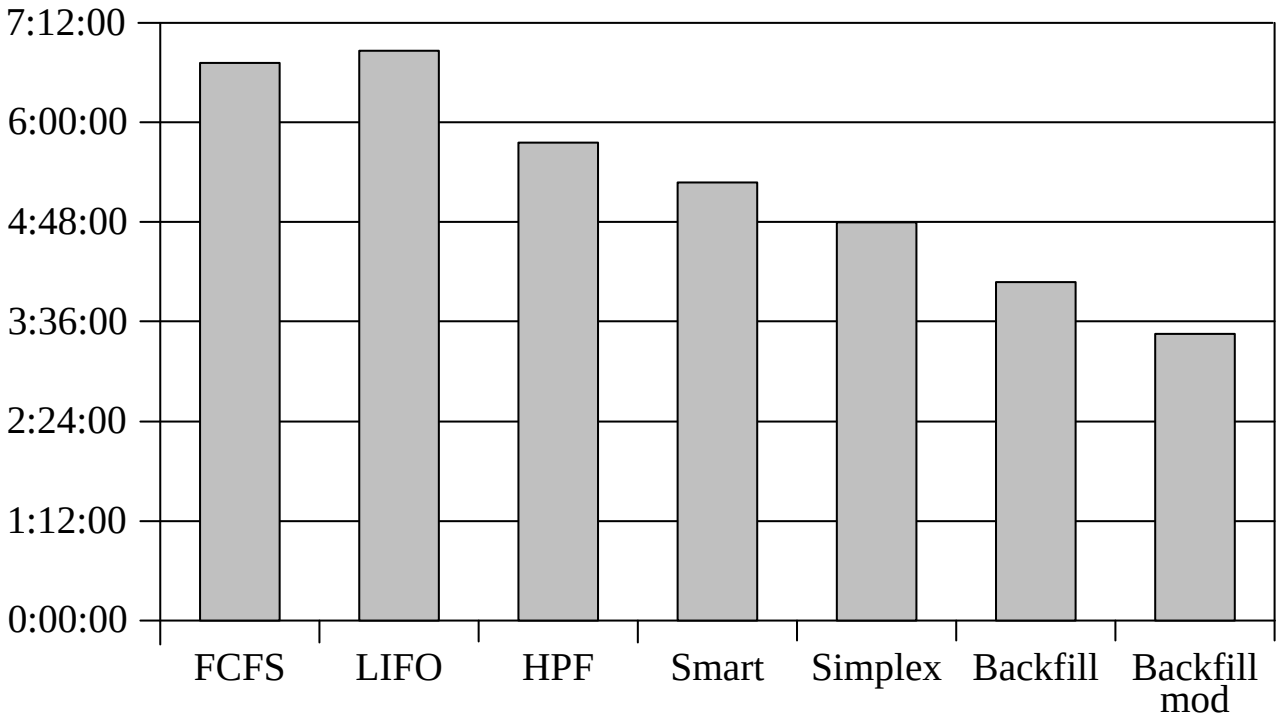

Рисунок 4.7 – Час виконання пулу завдань в середовищі GRASS

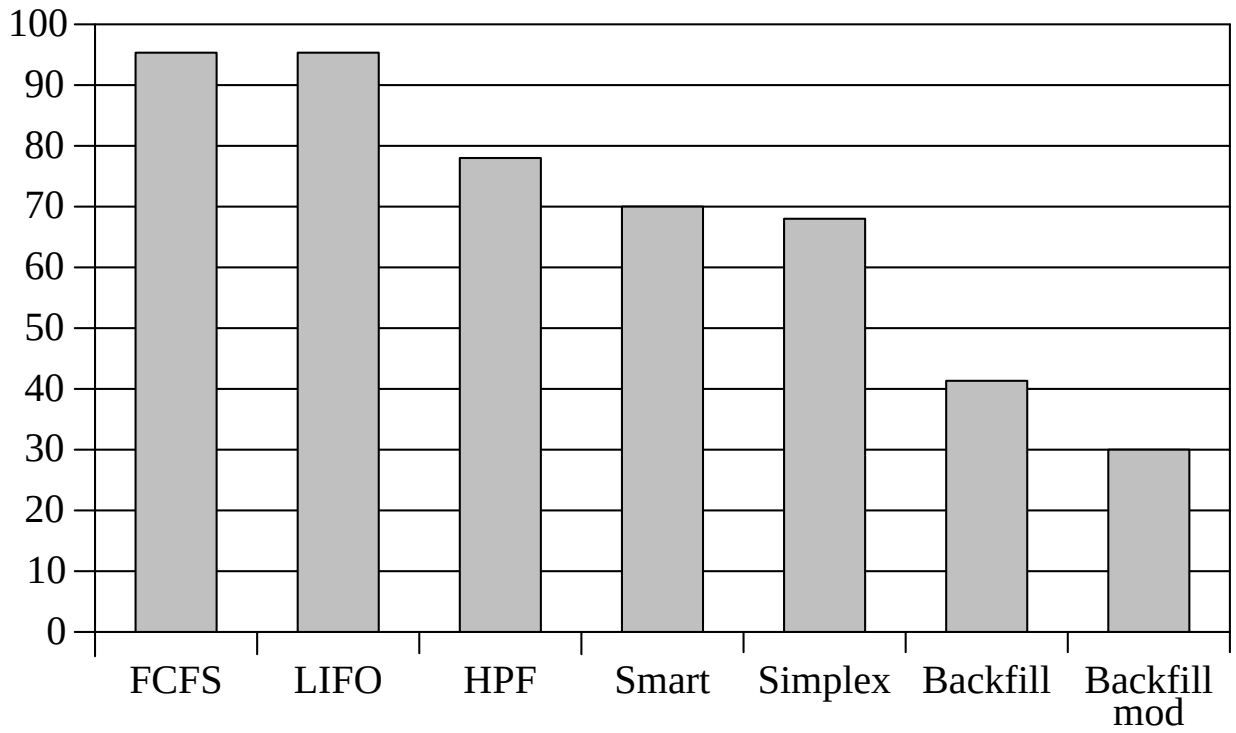

Рисунок 4.8 – Показник незадіянності обчислювальних ресурсів в середовищі GRASS

У таблиці 4.3 та на рисунку 4.9 наведені результати моделювання роботи всіх методів в середовищі GRASS відповідно до наведеної класифікації завдань.

Таблиця 4.3 – Час роботи методів розподілу в імітаційному середовищі моделювання GRASS для різних класів задач

Методи розподілу	Класи задач			
	1-й	2-й	3-й	4-й
FCFS	6:50:21	5:16:28	5:23:53	2:32:42
LIFO	6:54:43	5:58:50	6:01:52	2:37:53
HPF	6:33:45	6:00:35	6:08:32	2:40:22
Smart	6:28:55	6:03:51	6:10:44	3:09:49
Simplex	6:33:46	5:33:46	4:56:21	2:57:55
Backfill	5:19:54	5:01:49	3:28:43	3:42:46
Backfill (mod)	4:48:39	3:58:23	2:47:33	3:48:01

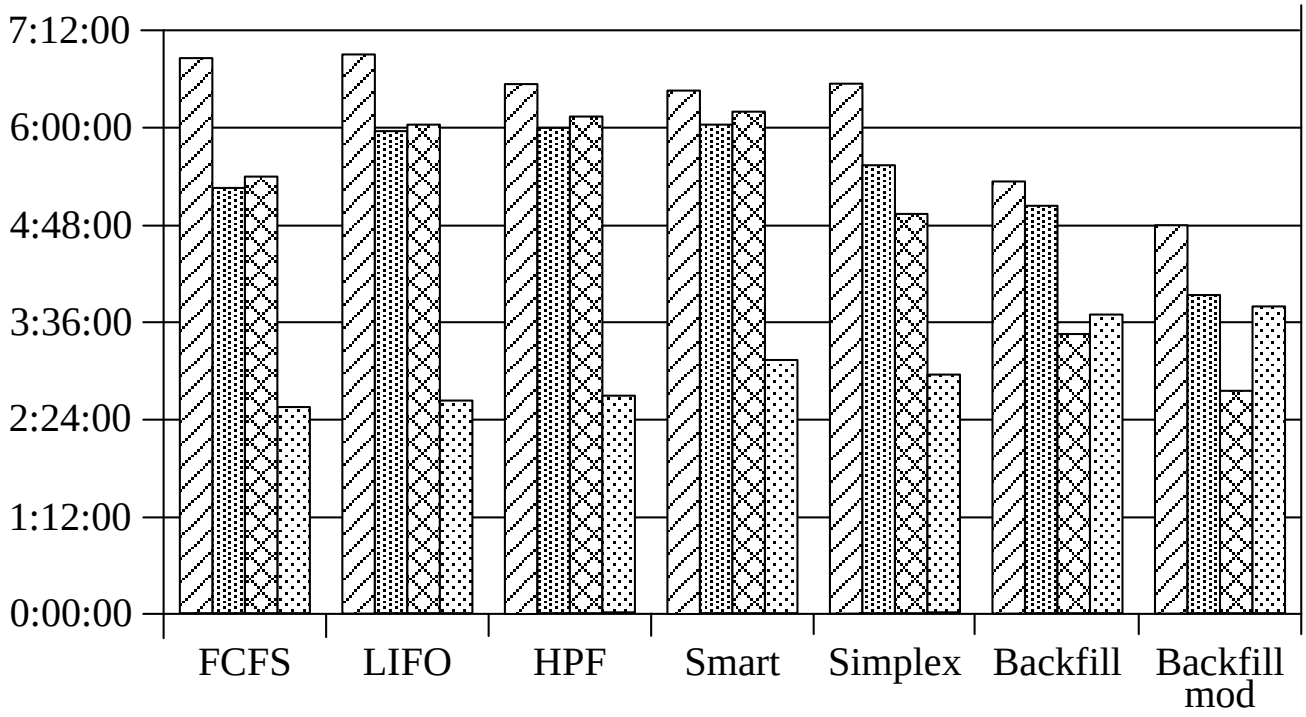

Рисунок 4.10 – Час роботи методів розподілу імітаційної середовища моделювання GRASS в залежності від класу завдань

Практична реалізація запропонованих методів та інформаційної технології в GRID-системі Радіоастрономічного інститут НАН України має місце при вирішенні задачі визначення часових запізнень у гравітаційно лінзованих квазарах, який надає процедуру усунення впливу подій мікролінзування, що грунтується

на фундаментальних властивостях репрезентації квадратично інтегрованих функцій у вигляді розкладання за ортогональними поліномами [150].

Високошвидкісні обчислення знайшли місце в обчисленні крос-кореляційних функцій для пар кривих блиску компонентів, представлених своїми апроксимаціями. Для усунення крайових ефектів при обчисленні крос-кореляційних функцій використана процедура, аналогічна процедурі обчислення локально-нормованої дискретної кореляційної функції для визначення тимчасового запізнювання в першій лінзі Q0957+561 в радіодіапазоні. Далі в якості оцінки тимчасового запізнювання для кожної даної пари приймався зсув у часі, при якому функція крос-кореляції досягає максимуму. Описаний метод реалізований на мові програмування IDL, а при його розподілі була застосована розроблена інформаційна технологія.

Розроблена інформаційна технологія була застосована при апробації вищевказаного методу визначення тимчасових запізнень для тестування семирічних кривих блиску гравітаційно лінзірованного квазара HE 0435-1223.

Впровадження даної інформаційної технології дозволило скоротити час визначення запізнень в гравітаційно лінзірованних квазарах, що надає просту та прозору процедуру усунення впливу подій мікролінзування, яка заснована на фундаментальних властивостях уявлення квадратично інтегрованих функцій у вигляді розкладання по ортогональних поліномами. При цьому, завдяки використанню запропонованої інформаційної технології, вдалося скоротити час обробки даних до 22% в GRID-середовищі, зокрема, GRID-кластері Радіоастрономічного інституту Національної академії наук України. В цілому це дозволило скоротити сумарний час за визначенням необхідних параметрів до 25% [150].

Розроблена інформаційна технологія планується до подальшого використання при обробці радіоастрономічних даних з метою визначення тимчасових запізнень в гравітаційно лінзірованних квазари з використанням хмарних технологій в рамках поставлених наукових завдань відділу космічної радіофізики РІ НАНУ.

В ході дослідження розробленої інформаційної технології розподілу завдань на обчислювальні ресурси, було проведено ряд експериментів. Під час проведення цих експериментів була отримана залежність результатів розподілу (час виконання пулу завдань) від класу задач [24,39]. Вони довели те що, застосування одного методу розподілу завдань в планувальнику є менш ефективним ніж множини. Більш ефективним варіантом є вібір оптимального плану розподілу за допомогою запропонованого методу пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів з множини методів розподілу, тому що завдання, які надходять до GRID-системи є різнорідними. В даний час не існує методу розподілу, який для будь-якого пулу завдань мав би оптимальний варіант розподілу за часом виконання. Однак, якщо відомі характеристики обчислювальних ресурсів GRID-системи, вимоги вхідного потоку завдань, то можна провести моделювання, обрати оптимальний метод розподілу для пулу завдань та запропонувати його реальній GRID-системі в якості найбільш ефективного рішення.

4.10 Висновки по розділу

В ході виконання роботи було модернізовано імітаційне середовище моделювання GRASS шляхом впровадження інформаційної технології розподілу завдань для гетерогенної GRID-системи. За рахунок того, що архітектура середовища GRASS є модульною, це дозволило проводити операції з його масштабування.

Впровадження розробленої інформаційної технології імітаційна середа моделювання GRASS надало можливість:

- паралельно моделювати реальний процес розподілу завдань на обчислювальних ресурсах в GRID-системі за кожним методом розподілу;

- обрання плану розподілу для пулу завдань на основі заданих експертами критеріїв та правил, які були отримані в ході моделювання (мінімальним часом виконання пулу завдань і мінімальним простоєм обчислювальних ресурсів);

- враховувати зв’язність задач в завданні та клас завдань при розподілі їх на обчислювальних ресурсах GRID-системи.

У ході дослідження було проведено ряд експериментів з розподілу завдань на обчислювальних ресурсах для різних методів розподілу з різними пулами завдань та обчислювальних ресурсів. Результати, які були отримані в ході експериментів, свідчать про скорочення часу виконання пулу завдань до 24% і підвищення показників ефективності використання обчислювальних ресурсів до 32% для ряду методів розподілу, які реалізовані в імітаційному середовищі моделювання GRASS.

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [21,24,29,30,34,37,39,40,145,147,149,150].

ВИСНОВКИ

В даний час поява нових інформаційних технологій, таких як хмарні обчислення, GRID, гібридні кластерні системи, зумовило якісне розвиток технологій розв'язання задач великої обчислювальної розмірності. До найбільш перспективних напрямків, сьогодні можна віднести технології паралельних розподілених обчислень, які є основою GRID-обчислень.

Важливе місце в GRID-системах при розподілі відводиться планувальникам завдань, в обов'язки яких ставиться створення розкладу використання обчислювальних ресурсів. Однак розглянуті планувальники мають ряд недоліків, і не існує універсального по ефективності методу, який розподіляє будь-які класи задач, враховуючи при цьому всі особливості наявних обчислювальних ресурсів.

У дисертаційній роботі пропонується нове вирішення актуальної науково-практичної задачі розробки методів та інформаційної технології розподілу завдань в гетерогенних GRID-системах. В результаті виконання роботи отримані нові наукові та практичні результати.

1. Проведено аналіз методів розподілу завдань в гетерогенних GRID-системах, який виявив, що існуючі планувальники мають ряд недоліків, головний з яких – це орієнтація на конкретний клас задач.

2. Модифіковано математичну модель розподілу завдань у GRID-системі за рахунок введення множини методів розподілу і додаткових параметрів у поданні обчислювальних ресурсів і завдань. Врахування зв'язності задач у завданні дозволяє скоротити час виконання пулу завдань у системі за рахунок усунення втрат за часом, які викликані обміном даних між окремими задачами в завданні. Множина методів розподілу дозволяє проведення серії експериментів для подальшого вибору плану розподілу з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів.

3. Модифіковано метод розподілу завдань на обчислювальні ресурси Backfill з урахуванням трафіку всередині завдання, який, на відміну від існуючих методів, враховує інтенсивність і обсяг потоків даних між задачами в завданні, що

дозволяє підвищити ефективність використання GRID-системи за рахунок зменшення часу виконання пулу завдань на розподілених гетерогенних обчислювальних ресурсах.

4. Вперше запропоновано метод пошуку розподілу завдань з мінімальним часом виконання пулу завдань та мінімальним простоєм обчислювальних ресурсів, який, на відміну від існуючих, використовує узагальнений критерій оцінки завдання та імітаційне середовище моделювання GRASS, що дозволяє підвищити ефективність використання обчислювальних ресурсів GRID-системи за рахунок скорочення часу виконання пулу завдань та простою обчислювальних ресурсів. Впровадження узагальненого критерію оцінки завдання в роботу планувальника дозволяє збільшити ефективність його роботи за рахунок зменшення простою обчислювальних ресурсів GRID-системи.

5. Розроблено інформаційну технологію розподілу завдань, яка була впроваджена в імітаційне середовище моделювання GRASS, яке дозволяє відтворювати процес функціонування елементарних подій, що відбуваються в реальній GRID-системі із збереженням логіки їх взаємодії у поточному часі. Запропонована інформаційна технологія об’єднує процеси передачі, зберігання, збору даних, спостережень за швидкоплинними процесами та процес їх обробки з використанням запропонованих у дисертаційній роботі методів.

6. В ході дослідження розробленої інформаційної технології було проведено ряд експериментів з розподілу завдань на обчислювальні ресурси для різних методів розподілу та з різними пулами завдань і обчислювальних ресурсів. Результати, отримані в ході експериментів, свідчать про скорочення часу виконання пулу завдань до 24% і підвищення ефективності використання обчислювальних ресурсів до 32% для ряду методів розподілу, реалізованих в імітаційному середовищі GRASS.

7. Проведено апробацію методів та інформаційної технології розподілу завдань на обчислювальні ресурси при вирішенні практичних задач. Впровадження запропонованої інформаційної технології дозволило скоротити час визначення запізнень в гравітаційно лінзірованних квазарах, що надає просту та прозору проце-

дуру усунення впливу подій мікролінзування, яка заснована на фундаментальних властивостях уявлення квадратично інтегрованих функцій у вигляді розкладання по ортогональних поліномам. При цьому, завдяки використанню розробленої інформаційної технології, вдалося скоротити час обробки даних до 22% в GRID-середовищі, зокрема, GRID-кластері Радіоастрономічного інституту Національної академії наук України. Розроблена інформаційна технологія була застосована при апробації вищевказаного методу визначення тимчасових запізнень для тестування семирічних кривих блиску гравітаційно лінзірованного квазара HE 0435-1223. В цілому це дозволило скоротити сумарний час за визначенням необхідних параметрів до 25%. Дана інформаційна технологія планується до подальшого використання при обробці радіоастрономічних даних з метою визначення тимчасових запізнень в гравітаційно лінзірованних квазари з використанням хмарних технологій в рамках поставлених наукових завдань відділу космічної радіофізики РІ НАНУ.

8. Розроблені програмні засоби, що реалізують запропоновані методи, можуть бути використані в наукових організаціях України та світу, де виникає необхідність оперативної передачі та паралельної обробки великих масивів даних.

9. Матеріали дисертаційної роботи використовуються в навчальному процесі на кафедрі електронних обчислювальних машин Харківського національного університету радіоелектроніки в процесі проведення лекційних занять і лабораторних робіт з курсів «Паралельне моделювання на HPC системах» та «Інтерфейси паралельного програмування».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ГОСТ ИСО/МЭК 2382-1-99 (ISO/IEC 2382-1:1993). Межгосударственный стандарт. Информационная технология. Словарь. Часть 1. Основные термины. Межгосударственный совет по стандартизации, метрологии и сертификации. Минск, 1999. 40 с.

2. ГОСТ ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции. Федеральное агентство по техническому регулированию и метрологии. М.: Стандартинформ, 2012. 73 с.

3. Кремень В. Г. Філософія освіти XXI століття. Педагогіка і психологія, 2003. № 1. С. 6–16.

4. Исаев Г. Н. Информационные технологии: учебное пособие. М.: Омега-Л, 2015. 464 с.

5. Foster I., Kesselman C. The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 2004. 748 p. URL: https://books.google.com.ua/books?hl=en&lr=&id=0l5gm6o3vrMC&oi=fnd&pg=PP1&dq=info:PUykcjyXRb8J:scholar.google.com&ots=hpLguSHund&sig=Mml_ByinupNJMtHO7VCacZuZG1k&redir_esc=y#v=onepage&q&f=false.

6. Бершадский А. М., Курилов Л. С., Финогеев А. Г. Исследование стратегий балансировки нагрузки в системах распределенной обработки данных. Известия высших учебных заведений. Поволжский регион. Технические науки. Пенза: Государственное образовательное учреждение высшего образования «Пензенский государственный университет», 2009. №4. С. 38–48.

7. Бычков И. В., Опарин Г. А., Феоктистов А. Г., Богданова В. Г., Пашинин А. А. Сервис-ориентированное мультиагентное управление распределенными вычислениями на основе мультиагентніх технологий. Известия Южного федерального университета. Технические науки. Ростов-на-Дону: Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Южный федеральный университет», 2014. № 12. 17–27.

8. Бычков И. В., Опарин Г. А., Феоктистов А. Г., Кантер А. Н. Мультиагентный алгоритм распределения вычислительных ресурсов на основе экономического механизма регулирования их спроса и предложения. Вестник компьютерных и информационных технологий. М.: Издательский дом «СПЕКТР», 2014. №1. С. 39–45.

9. Голубев И. А. Планирование задач в распределенных вычислительных системах на основе метаданных: дис. … канд. техн. наук: 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина). Санкт-Петербург, 2014. 135 с.

10. Кальпеева Ж. Б.Модели и методы организации вычислительных процессов в распределенной облачной среде: дисс. … докт. философии (PhD): 6D070400 «Вычислительная техника и программное обеспечение». Казахский национальный исследовательский технический университет им. К. И. Сатпаева. Алмааты, 2014. 136 с.

11. Покусин Н. В. Балансировка нагрузки, распределенной гетерогенной вычислительной системы в условиях априорной неопределенности о характере входного потока заявок. Интернет-журнал «Науковедение», 2013. №3. URL: http://naukovedenie.ru/PDF/82tvn313.pdf.

12. Селиверстов Е. Ю. Обзор методов решения задачи планирования параллельных алгоритмов. Электронный научно-технический журнал «Инженерный вестник», 2014. №12. С. 541–555. URL: http://engbul.bmstu.ru/doc/746179.html.

13. Телеснин Б. А. Методы и средства организации обработки потоковой информации на распределенных гетерогенных вычислительных комплексах: дис. ... канд. техн. наук: 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Федеральное государственное научное учреждение научно-исследовательский «Специализированные вычислительные устройства защиты и автоматики» (ФГНУ НИИ «Спецвузавтоматика»). Ростов на Дону, 2009. 134 с.

14. Agrawal D., Jaiswal L. H., Singh I., Chandrasekaran K. An Evolutionary

Approach to Optimizing Cloud Services. Computer Engineering and Intelligent System, 2012. 3:4. pp. 47–54.

15. Amar A., Bolze R., Boix E. DIET 2.8 user's manual, 2011. URL: http://graal. ens-lyon.fr/ ~diet/download/doc/UsersManualDiet2.8.1.pdf.

16. Amedro B., Bodnartchouk V., Baduel L. ProActive Scheduling v.3.3.2 user's manual, 2013. 152 p.

17. Sakellariou R., Zhao H. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. 18th International Parallel and Distributed Processing Symposium, 2004. 13 p.

18. Демичев А. П., Ильин В. А., Крюков А. П. Введение в грид-технологии. М.: Препринт НИИЯФ МГУ, 2007. 87 с.

19. Волк М. А., Филимончук Т. В., Гридель Р. Н. Методы распределения ресурсов для GRID-систем. Збірник наукових праць ХУПС. Харків: ХУПС, 2009. №1(19). С. 100–104.

20. Руденко О. Г., Волк М. А, Филимончук М. А., Филимончук Т. В. Архітектура системи моніторингу трафіку в GRID. Право і безпека. Харків: ХНУВС, 2010. №1(33). С. 226–229.

21. Волк М. А, Филимончук М. А., Филимончук Т. В. Модуль распределения заданий в GRID-системах. Системи обробки інформації. Харків: ХУПС, 2012. №2(100). С. 177–182.

22. Волк М. А., Филимончук Т. В., Ал Шиблак Муаз. Анализ современного состояния и развития GRID-технологий и языков описания заданий. Збірник наукових праць ХУПС. Харків: ХУПС, 2013. №2 (35). С. 75–81.

23. Волк М. А., Гридель Р. Н., Филимончук Т. В., Ал Шиблак Муаз. Формализация процессов распределенной имитации информационных систем. Системи обробки інформації. Харків: ХУПС, 2013. №4(111). С. 89–93.

24. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol.3/9 (81). pp. 45–53. (цитується в Scopus).

25. Волк М. А., Филимончук Т. В. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33–37.

26. Волк М. А., Филимончук М. А., Корниенко Т. В. (Филимончук Т. В.) Анализ использования искусственных иммунных систем в GRID-инфраструктуре. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції. К.: НТУУ «КПІ», 2008. С. 290.

27. Волк М. А., Филимончук М. А., Филимончук Т. В. Анализ алгоритмов распределения ресурсов в имитационных системах моделирования, ориентированных на GRID системы. Проблеми інформатики і моделювання. Матеріали восьмої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2008. С. 20.

28. Волк М. А, Филимончук М. А., Филимончук Т. В. Исследование методов распределения заданий для GRID-систем. Системный анализ и информационные технологии: Материалы XI Международной научно-технической конференции (26-30 мая 2009 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2009. С. 423.

29. Дьяченко К. Ю., Филимончук Т. В. Подключение модулей распределения задач к имитационной модели GRID-системы. Проблеми інформатики і моделювання. Матеріали дев’ятої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2009. С. 43.

30. Волк М. А, Филимончук Т. В., Гридель Р. Н. Имитационная система моделирования GRID-инфраструктуры GRASS. Системный анализ и информационные технологии: Материалы XII Международной научно-технической конференции (25-29 мая 2010 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2010. С. 359.

31. Фесенко М. В., Филимончук Т. В. Анализ алгоритмов распределения заявок в Grid-системах. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали першої науково-технічної конференції. Х.: ДП «ХНДІ ТМ»; К.: ДП «ЦНДІ НіУ», 2010. С. 76.

32. Филимончук М. А., Филимончук Т. В. Использование средств виртуализации при мониторинге трафика для GRID. Інформаційні технології в навігації і управлінні: стан та перспективи розвитку. Матеріали першої міжнародної науко-

во-технічної конференції. К.: ДП «ЦНДІ НіУ», 2010. С. 28.

33. Волк М. А, Филимончук Т. В. Анализ существующего прикладного программного обеспечения GRID-систем и языков описания заданий. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Матеріали третьої міжнародної науково-технічної конференції. Полтава: ПНТУ; Бєлгород: НДУ «БілДУ»; Харків: ДП «ХНДІ ТМ», Київ: НТУ; Кіровоград: КЛА НАУ, 2013. С. 49.

34. Волк М. А, Филимончук Т. В. Обобщенный критерий оценки задания для технологии планирования заданий в GRID. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам третьей международной научно-практической конференции (24-26 апреля 2013 г., Смоленск). В 3-х томах. Том 2. Смоленск: Смоленский филиал Российского университета кооперации, 2013. С. 172–176.

35. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий по ресурсам. Проблеми інформатизації: Матеріали першої міжнародної науково-технічної конференції. Черкаси: ЧДТУ; Київ: ДУТ; Тольятті: ТДУ; Полтава: ПНТУ, 2013. С. 20–21.

36. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в распределенных системах. Проблеми інформатизації: Матеріали другої міжнародної науково-технічної конференції. Київ: ДУТ; Полтава: ПНТУ; Катовице: Катовицький економічний університет; Париж: Університет Париж VII Вєнсент-Сен-Дєні; Білгород: НДУ «БДУ»; Черкаси: ЧДТУ; Харків: ХНДІТМ, 2014. С. 59–60.

37. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам четвертой международной научно-практической конференции (23-25 апреля 2014 г., Смоленск). В 2-х томах. Том 1. Смоленск: Смоленский филиал Российского университета кооперации, 2014. С. 379–383.

38. Филимончук Т. В., Ткачев В. Н. Информационная технология распределения заданий на вычислительные ресурсы в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам пятой международной научно-практической конференции (11-15 мая 2015 г., Смоленск). В 2-х томах. Том 1. Смоленск: Смоленский филиал Российского университета кооперации, 2015. С. 204–209.

39. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий на вычислительные ресурсы для обработки радиоастрономических данных в GRID-системах. Проблеми інформатизації: Матеріали третьої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2015. С. 23.

40. Волк М. А., Филимончук Т. В. Использование системы моделирования GRASS в задачах распределения заданий в GRID-системах. 20-й Ювілейний міжнародний форум «Радіоелектроніка та молодь у XXI столітті». Харків: ХНУРЕ, 2016. Т.4. С. 171–172.

41. Volk M. O., Filimonchuk T. V. Information technology for job distribution in GRID-systems Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: матеріали шостої міжнародної науково-технічної конференції. Полтава: ПНТУ; Баку: ВА ЗС АР; Кіровоград: КЛА НАУ; Харків: ДП «ХНДІ ТМ», 2016. С. 45–46.

42. Ткачев В. Н., Филимончук Т. В., Митин Д. Е. Использование информационной технологии распределения заданий при обработке больших массивов данных в виртуальных частных облаках. Информационные системы и технологии: Материалы 5-й международной научно-технической конференции (2-17 сентября 2016 г., Харьков). Х.: ДРУКАРНЯ МАДРИД, 2016. С. 333–334.

43. Волк М. А., Филимончук Т. В. Модифицированный метод Backfill с консервативным резервированием. Проблеми інформатизації: Матеріали четвертої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ, 2016. С. 20.

44. Гридель Р. Н. Моделі, методи та інформаційна технологія аналізу розподілених програмних моделей GRID-систем : дис. ... канд. техн. наук : 05.13.06 «Информационные технологи»; Министерство образования и науки, Харьковский национальный университет радиоэлектроники. Харьков, 2015. 169 с.

45. Кандаурова Н. В., Яковлев С. В., Яковлев В. П., Чеканов В. С. Вычислительные системы, сети и телекоммуникации: Учебное пособие. 2-е изд., М.: ФЛИНТА, 2013. 344 с.

46. Дрейвс Ю. Г. Организация ЭВМ и вычислительных систем: Учебник для вузов. М.: Высшая школа, 2006. 501 с.

47. Пятибратов А. П., Гудыно Л. П., Кириченко А. А. Вычислительные системы, сети и телекоммуникации: Учебник. 2-ое изд., перераб. и доп. М.: Финансы и статистика, 2004. 512 с.

48. Михайлов Б. М., Халабия Р. Ф. Классификация и организация вычислительных систем: Учебное пособие. М.: МГУПИ, 2010. 144 с.

49. Таненбаум Э., Ван-Стеен М. Распределенные системы. Принципы и парадигмы. Спб.: Питер, 2003. 877 с.

50. Родин А. В., Бурцев В. Л. Параллельные или распределенные вычислительные системы? Научная сессия МИФИ-2006. Т.12. Информатика и процессы управления. Компьютерные системы и технологии, 2006. С. 151–155.

51. Zhaohui Wu, Jiaoyan Chen. Semantic Grid: Model, Methodology, and Applications. URL: http://www.springer.com/la/book/9783540794530#aboutAuthors.

52. Попков Ю. С. Макросистемы и Grid-технологии: моделирование динамических стохастических сетей. Проблемы управления, 2003. №8 С. 10–20.

53. Foster I. The anatomy of the Grid: enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 2001. Vol. 15(3). pp. 200–222.

54. Foster I., Kesselman C., Nick J., Tuecke S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Computer Networks: The International Journal of Computer and Telecommunications Networking, 2002. Vol. 40(1). pp. 5–17.

55. Филамофитский П. П. Система поддержки метакомпьютерных расчетов X-Com: архитектура и технология работы. Вычислительные методы и программирование, 2004. Т.5. №2 С. 123–137.

56. Heien E. M., Anderson P. D., Hagihara K. Computing Low Latency Batches with Unreliable Workers in Volunteer Computing Environments. Journal of Grid Computing, 2009. Vol. 7(4). pp. 501–519.

57. Brian H. Cloud computing. Communications of the ACM, 2008. Vol. 51(7). pp. 9–11.

58. Петров М. Ю. Опыт построения и использования в Грид-технологиях кластера высокопроизводительных параллельных вычислений. Труды СПИИРАН, 2003. Вып. 1. С. 41–48.

5. Милюков В. В., Сосновский Ю. В. Моделирование фрагментов Grid-системы в симуляторе GridSim. Оптимізація виробничих процесів, 2013. № 14. С. 218–222.

60. Дунаев А. В., Ларченко А. В., Бухановский А. В. Моделирование параллельных вычислительных процессов среде Грид на примере Intel Grid Programming Environment. Параллельные вычислительные технологии (ПаВТ’2008): Труды международной научной конференции (28 января – 1 февраля 2008 г., г. Санкт-Петербург). Челябинск: Изд. ЮУрГУ, 2008. С. 390–394.

61. Литвинов В. В., Стеценко І. В. Управління розподіленими ресурсами грід-системи. Математичні машини і системи. № 2. Том 1, 2012. С. 3–12.

62. Петренко А. І. Хмарні і Грід обчислення для е-науки. Системный анализ и информационные технологии: 14-я международная научно-техническая конференция «САИТ-2012» (24 апреля 2012, Киев). К.: УНК «ИПСА «НТУУ «КПИ», 2012. С. 294–295.

63. Peter Mell, Timothy Grance The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. National Institute of Standards and Technology Special Publication, 2011. P.7. pp. 800–145.

64. Михайлов П. А., Радченко Г. И. Методы моделирования и оценки производительности облачных систем. Вестн. ЮУрГУ. Серия: Вычислительная мате-

матика и информатика. №3. Том 3, 2014. С. 109–123.

65. Батура Т. В., Мурзин Ф. А., Семич Д. Ф. Облачные технологии: основные понятия, задачи и тенденции развития. Программные продукты и системы и алгоритмы. №1, 2014. URL: http://swsys-web.ru/cloud-computing-basic-concepts-problems.html.

66. Бычков И. В., Опарин Г. А., Феоктистов А. Г., Корсуков А. С. Распределение заданий в интегрированной кластерной системе на основе их классификации. Вычислительные технологии. №2. Том 18, 2013 – С. 25–32.

67. Агаларов М. Я., Агаларов Я. М. Оптимизация квазистатического плана распределения вычислительных ресурсов специализированной вычислительной системы. Информационные технологии и вычислительные системы. №3, 2012, С. 43–54.

68. Шаповалов Т. С. Планирование выполнения заданий в распределенных вычислительных системах с применением генетических алгоритмов: дис. … канд. техн. наук : 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Институт динамики систем и теории управления СО РАН. Иркутск, 2011. 146 с.

69. Замятин А. А. Неманипулируемый для пользователей механизм распределения процессорного времени между неделимыми заданиями. Параллельные вычислительные технологии (ПаВТ’2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). Челябинск: Издательский центр ЮУрГУ, 2013. С. 127–135.

70. Аветисян А. И., Гайсарян С. С., Грушин Д. А., Кузюрин Н. Н., Шокуров А. В. Эвристики распределения задач для брокера ресурсов Grid. Труды Института системного программирования. Т 5, 2004. С. 269–280.

71. Красавина А. К. Исследование методов и алгоритмов распределения задач между исполнителями в системах управления проектами. International Scientific. Practical Conference «Innovative information technologies». С. 134–138.

72. Кошкарева Ю. И., Третьякова А. А. Моделирование подходов к распределению ресурсов и оценка времени распределения задач в облачных грид систе-

мах. Седьмая всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика» (ИММОД-2015): Труды конф., 21–23 окт. 2015 г., Москва: в 2 т. / Ин-т проблем упр. им. В.А. Трапезникова Рос. Акад. наук. Т 1. М.: ИПУ РАН, 2015. С. 267-274.

73. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 1999. 960 с.

74. Neumann K. Stochastic project networks. Temporal analysis, scheduling and cost minimization. Berlin: Springer-Verlag, 1990. 237 p.

75. Blazewicz J., Lenstra J.K. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics. Vol. 5, 1983. pp. 11–24.

76. Хемди А. Таха Введение в исследование операций, 7-е издание: Пер. с англ. М.: Издательский дом «Вильямс», 2005. – 912 с.

77. Пономаренко В. С., Листровой С. В., Минухин С. В., Знахур С. В. Методы и модели планирования ресурсов в GRID-системах. Монография. Х.: ВД «ІНЖЕК», 2008. 408 с.

78. Листровой С. В., Яблочков С. В. Метод решения задачи определения минимальных вершинных покрытий и независимых максимальных множеств. Электронное моделирование. № 2. Том 25, 2003. С. 31–40.

79. Пономаренко В. С., Листровой С. В. Метод решения задачи о минимальном покрытии как средство планирования в GRID. Проблемы управления. №3, 2008. С. 78–84.

80. Листровой С. В., Гуль А. Ю. Метод решения задачи о минимальном покрытии на основе рангового подхода. Электронное моделирование. №1, 1999. С. 58–70.

81. Минухин С. В. Подходы к организации планирования распределением ресурсов в GRID-системах. Системи обробки інформації. Харків: ХУПС, 2009. №3(77). С. 49–54.

82. Nemhauster G. L., Pruul E. A., Rushmeier R. A. Branch-and-bound and Parallel Computation: a Historical Note. Oper. Res. Let. №7, 1988. pp. 65–69.

83. Boctor F. F. Some e-cient multi-heuristic procedures for resource-constrained project scheduling. European journal of operational research. Vol. 49(1), 1990. pp. 3–13.

84. Harvey W. D., Ginsberg M. L. Limited discrepancy search. Proceding of the International Joint Conference on Artificial Intelligence (IJCAI-95). Montreal, 1995. pp. 607–613.

85. Patterson J. H. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. Management Science, 1984. Vol. 30(7). pp. 854–867.

86. Sampson S. E., Weiss E. N. Local search techniques for the generalized resource constrained project scheduling problem. Naval Research Logistics, 1993. Vol. 40(5). P. 665.

87. Panwalker S., Iskander W. A survey of scheduling rules. Operations Research, 1997. Vol. 25(1). pp. 45–61.

88. Devis E. W. Patterson J. H. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. Management Science, 1975. Vol. 21(8). pp. 944–955.

89. Devis E. W., Heidorn G. E. An algorithm for optimal project scheduling under multiple resource constraints. Management Science, 1971. Vol 17(12). pp. 803–817.

90. Cai Canonical X. Coin Systems for Change-Making Problems. Proceedings of the Ninth International Conference on Hybrid Intelligent Systems, 2009. pp. 499–504.

91. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs (3rd Edn.). Springer Science & Business Media, 1996. 387 p.

92. Шаповалов Т. С. Генетический алгоритм составления расписания запуска параллельных заданий в GRID. Многопроцессорные вычислительные системы, 2010. № 4 (26). С. 115–126.

93. Шаповалов Т. С. Применение генетических алгоритмов для поиска оптимального расписания заданий в Grid. Труды международной конференции «Параллельные вычислительные технологии». Челябинск: Изд. ЮУрГУ, 2008.

С. 500–505.

94. Липинский Л. В. Семенкин Е. С. Применение алгоритма генетического программирования в задачах автоматизации проектирования интеллектуальных информационных технологий. Вестник Сибирского государственного аэрокосмического университета им. ак. М.Ф. Решетнева. Вып. 3 (10), 2006. С. 22–26.

95. Назаров А. В., Лоскутов А. И. Нейросетевые алгоритмы прогнозирования и оптимизации систем. Наука и техника, 2003. 384 с.

96. Коваленко В. Н., Коваленко Е. И., Корягин Д. А., Семячкин Д. А Управление параллельными заданиями в гриде с неотчуждаемыми ресурсами. Препринт №63. Москва: ИПМ РАН, 2007. С. 1–28.

97. Коваленко В. Н. Семячкин Д. А. Использование алгоритма Backfill в ГРИД. Распределенные вычисления и Грид-технологии в науке и образовании: труды международной конференции, Дубна, 29 июня – 2 июля 2004 г., Россия. Дубна, 2004, С. 139–144.

98. Березовский П. С., Коваленко В. Н.. Состав и функции системы диспетчеризации заданий в гриде с некластеризованными ресурсами. Препринт №67. Москва: ИПМ им. М. В.Келдыша РАН, 2007. URL: http://www.keldysh.ru/papers/2007/prep67/prep2007_67.html.

99. Полежаев П. Н. Об эффективности алгоритмов планирования задач управления потоками данных облачных грид-систем. Вестник Оренбургского государственного университета. Выпуск № 3 (164), 2014. С. 168–172.

100. Mu’alem A.W. and Feitelson D.G. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans. Parallel & Distributed Syst. 12(6), 2001. pp. 529–543.

101. Foster I. What is the Grid? A Three Point Checklist. GRIDToday, 2002. URL: https://www.researchgate.net/publication/215757948_What_is_the_Grid_A_Three_Point_Checklist.

102. Enabling Grids for E-sciencE project (EGEE). URL: http://eu-egee-org.web.cern.ch/ eu-egee-org/index.html.

103. TeraGrid (SUPERCOMPUTING NETWORK). URL: https://www.britan-

nica. com/topic/TeraGrid.

104. Eerola P. The NorduGrid production Grid infrastructure, status and plans. Proc. 4th Intel Workshop on Grid Computing (GRID'03). IEEE CS Press, 2003. pp. 158–165.

105. NorduGrid. Grid Solution for Wide Area Computing and Data Handling. URL: http://www.nordugrid.org.

106. Open Science Grid (OSG). URL: https://www.opensciencegrid.org.

107. Multiscale APPlications on European e-Infrastructures (MAPPER). URL: http://www.mapper-project.eu/web/guest/home.

108. Фраленко В. П., Агроник А. Ю. Средства, методы и алгоритмы эффективного распараллеливания вычислительной нагрузки в гетерогенных средах. Программные системы: теория и приложения, 2015. 6:3(26). С.73–92.

109. Юрич М. Ю. Анализ систем управления заданиями в рамках GRID. Радіоелектроніка. Інформатика. Управління, 2010. №1 (22). С. 126–134.

110. Lovas R., Афанасьев А. П., Волошинов В. В., Посыпкин М. А., Сухорослов О. В., Храпов Н. П. Увеличение вычислительной мощности распределенных систем с помощью грид-систем из персональных компьютеров. Параллельные вычислительные технологии (ПаВТ'2011): труды международной научной конференции. Челябинск: Издательский центр ЮУрГУ, 2011. С. 6–14.

111. HTCondor: Hight Throughput Computing. URL: https://research.cs.wisc.edu/htcondor.

112. HTCondor Version 8.0.0 Manual. University of Wisconsin–Madison: Center for High Throughput Computing, 2013. 1053 p.

113. Голубовский А. В., Гиоргизова-Гай В. Ш Построение одноуровневой Грид-системы на платформе Condor. Материалы Международной научно-технической конференции SAIT 2011,Киев, 23–28 мая 2011 г. К.: УНК «ИПСА» НТУУ «КПИ», 2011. С. 364.

114. Abdelkader A., Caniou Y., Caron E. et al. DIET 2.8. User's manual. Inria, ENS-Lyon, UCBL, SysFera, 2011. URL: http://graal.ens-lyon.fr/~diet/UsersManual-DIET2.8/UsersManual.html.

115. PBS Works 10 Enabling On-Demand Comp. URL: http://www.pbsgridworks.com/images/solutions-en-US/PBS_10.2Highlights_letr_REVISE-WEB.pdf.

116. TORQUE Resource Manager Guide. URL: http://www.clusterresources.com/products/torque-resource-manager.php.

117. Maui v.3.2 Administrator's Guide. Adaptive Computing Enterprises, 2011. P. 287.

118. Moab Workload Manager v.7.2.4 Administrator Guide. Adaptive Computing Enterprises, 2013. P. 1136.

119. Maui Scheduler™. Administrator's Guide. URL: http://docs.adaptivecomputing.com/maui.

120. Moab Workload Manager Administrator's Guide. Appendix K: Migrating from Maui 3.2. URL: http://www.clusterresources.com/products/mwm/docs/a.kmaui-migrate.shtml.

121. Кореньков В. В., Нечаевский А. В., Ососков Г. А., Пряхина Д. И., Трофимов В. В., Ужинский А. В Моделирование грид-облачных сервисов проекта NICA как средство повышения эффективности их разработки. Компьютерные исследования и моделирование, 2014. Т. 6. № 5. С. 635–642.

122. Афанасьев А. П., Посыпкин М. А., Заикин О. С., Семёнов А. А., Манзюк М. О., Бычков И. В. Концепция многозадачной грид-системы с гибким управлением свободными вычислительными ресурсами суперкомпьютеров. Национальный Суперкомпьютерный Форум (НСКФ-2014). Россия, Переславль-Залесский, ИПС имени А.К. Айламазяна РАН (25-27 ноября 2014 г). URL: http://2014.nscf.ru/TesisAll/0_NSCF_Plenar/12_189_BichkovIV.pdf.

123. Грушин Д. А., Поспелов А. И. Система моделирования Grid: реализация и возможности применения. Труды Института системного программирования РАН, 2010. Т. 18. С. 243–260.

124. Ferreira A., Pozzi G. Workflow Management Systems: Exercise Book. Societa Editrice Esculapio, 2012. 136 p.

125. Курбин С. Изменения и дополнения в RSL. RS-CLUB: Информационно-аналитическое издание компании R-Style Softlab, 1996. №1, С. 50–55.

126. Курбин С. Способы решения проблем в языке RSL. RS-CLUB: Информационно-аналитическое издание компании R-Style Softlab, 1996. №2. С. 42–43.

127. Condor Version 7.6.10 Manual. URL: http://research.cs.wisc.edu/htcondor/manual/v7.6.10.

128. Job Submission Description Language (JSDL). Specification, Version 1.0. URL: https://www.ogf.org/documents/GFD.56.pdf .

129. Введение в JSON. URL: http:// json.org/json-ru.html.

130. Yaml Is Not Markup Language (YAML). URL: http://www.webcitation.org/65LPKtdaB.

131. Extensible Markup Language (XML) 1.0. URL: https://www.w3.org/TR/2004/REC-xml-20040204.

132. Журавлёв Е. Е., Корниенко В. Н., Олейников А. Я., Широбокова Т. Д. Разработка первого национального стандарта для обеспечения интероперабельности в Грид-среде. Журнал радиоэлектроники №2, 2011. URL: http://jre.cplire.ru/alt/feb11/4/text.html.

133. Boardman R., Crouch S., Mills H., Newhouse S., Papay J. and the OMII-UK team. Towards Grid Interoperability. Open Middleware Infrastructure Institute UK (OMII-UK) University of Southampton. URL: https://eprints.soton.ac.uk/265165/1/781.pdf.

134. Петренко А. І., Свистунов С. Я., Свірін П. В. Брокер ресурсів для Nordugrid ARC із використанням прогнозування часу початку виконання. Моделювання та інформаційні технології, 2013. Вип. 68. С. 170–176.

135. Venugopal S., Buyya R., Winton L. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. Proceedings of the 2nd International Workshop on Middleware for Grid Computing. URL: http://arxiv.org/ftp/cs/papers/0405/0405023.pdf.

136. Möser, F. Integration von Optimierungsalgorithmen in den Grid Resource Broker GORBA. URL: https://www.iai.kit.edu/www-extern/fileadmin/Image_Archive/simulation/optimierung/SuCo/pdf/Moe09.pdf.

137. Venugopal S., Buyya R., Winton L. A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids. Concurrency and Computation: Practice and Experience, 2006. Vol. 18. Issue 6. pp. 685–699.

138. Atakan D. Biobjective Scheduling Algorithms for Execution Time-Reliability Trade-off in Heterogeneous Computing Systems. The Computer Journal, 2005. Vol. 48(3). pp. 300–314.

139. Официальный сайт компании Oracle Corp. URL: https://www.oracle. com/sun/index.html.

140. IBM Tivoli Workload Scheduler LoadLeveler. URL: http://www.interface.ru/home.asp?artId=6283.

141. Load Sharing Facility (LSF). URL: http://help.unc.edu/help/lsf-load-sharing-facility.

142. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы. 5-е издание. СПб.: БХВ-Петербург, 2016. 992 с.

143. Замятина Е. Б., Мерзляков Д. В., Семеновых А. А. Языковые и программные средства для многомодельного исследования имитационных моделей компьютерных сетей. Седьмая всероссийская научено-практическая конференция «Имитационное моделирование. Теория и практика» (ИММОД-2015). С. 86–94.

144. Брахман Т. Д. Многокритериальность и выбор альтернативы в технике. М.: Радио и связь, 1984. 288 с.

145. Ткачев В. Н. Захаренко В. В., Васильева Я. Ю., Царин Ю. А., Банникова Е. Ю., Илюшин В. В., Саваневич В. Е., Герасименко О. В., Анненков А. Б., Кулишенко С. Ф., Кулишенко Д. Ф. Использование грид-технологий в решении задач радиофизики и радиоастрономии. Радиофизика и радиоастрономия, 2013. Т. 18, № 2. С. 176–188.

146. Волк, М. А. Структура программного комплекса имитационного моделирования элементов GRID-систем для научных исследований. Системи обробки інформації, 2009. Вип. 3. С. 125–128.

147. Буч Г., Рамбо Д., Джекобсон А. Язык UML: руководство пользователя. URL: http://kaf401.rloc.ru/TRPO/UMLBooch/UMLBoochContent.htm.

148. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих: Пер. с англ. СПб.: БХВ-Петербург, 2013. 816 с.

149. Quetier B., Capello F. A survey of Grid research tools: simulators, emulators and real life platform. In: 17th IMACS World Congress. Scientific Computation. Applied Mathematics and Simulation. Paris, France, 2005. URL: http://citeseerx.ist. psu.edu/viewdoc/download?doi=10.1.1.106.8816&rep=rep1&type=pdf.

150. Цветкова В. С., Шульга В. М., Бердина Л. А. Измерение временных запаздываний в гравитационно линзированных квазарах при наличии микролинзирований. Радиофизика и радиоастрономия, 2014. Т. 19. № 4. С. 307–316.

ДОДАТОК А

Акти впровадження результатів дисертаційної роботи

Національна академія наук України
Радіоастрономічний інститут
РІ НАНУ
вул. Червонопрапорна,4 м.Харків, 61002
тел 38(057)315 11 27, факс 38(057)706 14 15
E-mail: rian@rian.kharkov.ua
Web: http://rian.kharkov.ua
Код ЄДРПОУ 02772020

ЗАТВЕРДЖУЮ
Директор РІ НАН України
академік НАН України
_____Л.М. Литвиненко
«30 » «листопада » 2015 р.

АКТ

О внедрении результатов научной деятельности ассистента кафедры электронных вычислительных машин ХНУРЭ Филимончук Татьяны Владимировны.

Ассистент кафедры ЭВМ ХНУРЭ Филимончук Т.В. разработала информационную технологию распределения заданий на вычислительные ресурсы в GRID-системах.

Внедрение данной информационной технологии в виде программных средств позволило сократить время определения запаздываний в гравитационно линзированных квазарах, что предоставляет простую и прозрачную процедуру устранения влияния событий микролинзирования, основанную на фундаментальных свойствах представления квадратично интегрируемых функций в виде разложения по ортогональным полиномам.

При этом, благодаря использованию разработанной информационной технологии Филимончук Т.В., удалось сократить время обработки данных до 22% в ГРИД-среде, в частности, на ГРИД-кластере Радиоастрономического института Национальной академии наук Украины.

Разработанная информационная технология была применена при апробации вышеуказанного метода определения временных запаздываний для тестирования семилетних кривых блеска гравитационно линзированного квазара HE 0435-1223.

В целом это позволило сократить суммарное время по определению необходимых параметров до 25%.

Разработанная информационная технология планируется к дальнейшему использованию при обработке радиоастрономических данных с целью

определения временных запаздываний в гравитационно линзированных квазарах с использованием облачных технологий в рамках поставленных научных задач отдела космической радиофизики РИ НАНУ.

Акт составлен для представления в специализированный ученый совет по защите диссертаций и не является основанием для финансовых расчетов.

Заведующий отделом космической радиофизики РИ НАНУ, д.ф.-м.н., с.н.с. — В.П. Тишковец

Научный сотрудник отдела космической радиофизики РИ НАНУ к.ф.-м.н. — Л.А. Бердина

Инженер-исследователь, системный администратор РИ НАНУ — В.Н. Ткачев

Подписи удостоверяю.

Ученый секретарь РИ НАНУ, к.ф.-м.н., с.н.с. — А.П. Удовенко

УКРАЇНА НАЦІОНАЛЬНА АКАДЕМІЯ НАУК
Радіоастрономічний інститут
Ідентифікаційний код 02772020

«ЗАТВЕРДЖУЮ»
В.о. проректор з науково-методичної
роботи Харківського національного
університету радіоелектроніки
_____ Рубан І.В.
«__» _____ 2017 р.

АКТ

про використання в навчальному процесі результатів дисертаційної роботи на тему: «Методи та інформаційна технологія розподілу завдань в гетерогенних GRID-системах» здобувача кафедри електронних обчислювальних машин Харківського національного університету радіоелектроніки Філімончук Тетяни Володимирівни

Комісія у складі:

Голови: завідувача кафедри електронних обчислювальних машин д.т.н., проф. Міхаль О.П.

Членів комісії: начальника навчального відділу ХНУРЕ к.т.н., доц. Свид І.В., к.т.н., доц. кафедри електронних обчислювальних машин Ляшенка О.С.

встановила, що результати наукових досліджень реалізовано в навчальному процесі Харківського національного університету радіоелектроніки на кафедрі електронних обчислювальних машин (протокол засідання кафедри ЕОМ №7 від 06.01.2017 р.).

Розглянувши матеріали роботи та організацію навчального процесу на кафедрі, комісія відзначає, що при проведенні лекційних занять та лабораторних робіт з курсів «Інтерфейси паралельного програмування» та «Паралельне моделювання на HPC системах» використані наступні результати дисертаційної роботи:

- модель розподілення завдань на обчислювальні ресурси у GRID-системах, яка враховує властивості мережевого трафіку всередині завдань;
- метод пошуку найкращого розподілу для пулу завдань з використанням імітаційного середовища моделювання, що дозволяє отримати оптимальний варіант розподілу з мінімальним часом простою обчислювальних ресурсів на підставі результатів моделювання поведінки GRID-системи при різних алгоритмах розподілу завдань.

Завідувач кафедри електронних обчислювальних машин О.П. Міхаль

Начальник навчального відділу І.В. Свид

Доцент кафедри електронних обчислювальних машин О.С. Ляшенко

ДОДАТОК Б

Список публікацій здобувача

1. Волк М. А., Филимончук Т. В., Гридель Р. Н. Методы распределения ресурсов для GRID-систем. Збірник наукових праць ХУПС. Харків: ХУПС, 2009. №1(19). С. 100–104.

2. Руденко О. Г., Волк М. А, Филимончук М. А., Филимончук Т. В. Архітектура системи моніторингу трафіку в GRID. Право і безпека. Харків: ХНУВС, 2010. №1(33). С. 226–229.

3. Волк М. А, Филимончук М. А., Филимончук Т. В. Модуль распределения заданий в GRID-системах. Системи обробки інформації. Харків: ХУПС, 2012. №2(100). С. 177–182.

4. Волк М. А., Филимончук Т. В., Ал Шиблак Муаз Анализ современного состояния и развития GRID-технологий и языков описания заданий. Збірник наукових праць ХУПС. Харків: ХУПС, 2013. №2 (35). С. 75–81.

5. Волк М. А., Гридель Р. Н., Филимончук Т. В., Ал Шиблак Муаз Формализация процессов распределенной имитации информационных систем. Системи обробки інформації. Харків: ХУПС, 2013. №4(111). С. 89–93.

6. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment. Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol.3/9 (81). P. 45–53. (цитується в Scopus).

7. Волк М. А., Филимончук Т. В. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33–37.

8. Волк М. А., Филимончук М. А., Корниенко Т. В. (Филимончук Т. В.) Анализ использования искусственных иммунных систем в GRID-инфраструктуре. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції. К.: НТУУ «КПІ», 2008. С. 290.

9. Волк М. А., Филимончук М. А., Филимончук Т. В. Анализ алгоритмов распределения ресурсов в имитационных системах моделирования, ориентированных на GRID системы. Проблеми інформатики і моделювання. Матеріали восьмої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2008. С. 20.

10. Волк М. А, Филимончук М. А., Филимончук Т. В. Исследование методов распределения заданий для GRID-систем. Системный анализ и информационные технологии: Материалы XI Международной научно-технической конференции (26-30 мая 2009 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2009. С. 423.

11. Дьяченко К. Ю., Филимончук Т. В. Подключение модулей распределения задач к имитационной модели GRID-системы. Проблеми інформатики і моделювання. Матеріали дев'ятої міжнародної науково-технічної конференції. Х.: НТУ «ХПІ», 2009. С. 43.

12. Волк М. А, Филимончук Т. В., Гридель Р. Н. Имитационная система моделирования GRID-инфраструктуры GRASS. Системный анализ и информационные технологии: Материалы XII Международной научно-технической конференции (25-29 мая 2010 г., Киев). К.: УНК «ИПСА» НТУУ «КПИ», 2010. С. 359.

13. Фесенко М. В., Филимончук Т. В. Анализ алгоритмов распределения заявок в Grid-системах. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Матеріали першої науково-технічної конференції. Х.: ДП «ХНДІ ТМ»; К.: ДП «ЦНДІ НіУ», 2010. С. 76.

14. Филимончук М. А., Филимончук Т. В. Использование средств виртуализации при мониторинге трафика для GRID. Інформаційні технології в навігації і управлінні: стан та перспективи розвитку. Матеріали першої міжнародної науково-технічної конференції. К.: ДП «ЦНДІ НіУ», 2010. С. 28.

15. Волк М. А, Филимончук Т. В. Анализ существующего прикладного программного обеспечения GRID-систем и языков описания заданий. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Матеріали третьої міжнародної науково-технічної конференції. Полтава: ПНТУ; Бєлгород: НДУ «БілДУ»; Харків: ДП «ХНДІ ТМ», Київ: НТУ; Кіровоград: КЛА НАУ, 2013. С. 49.

16. Волк М. А, Филимончук Т. В. Обобщенный критерий оценки задания для технологии планирования заданий в GRID. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам третьей международной научно-практической конференции (24-26 апреля 2013 г., Смоленск). В 3-х томах. Том 2. Смоленск: Смоленский филиал Российского университета кооперации, 2013. С. 172–176.

17. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий по ресурсам. Проблеми інформатизації: Матеріали першої міжнародної науково-технічної конференції. Черкаси: ЧДТУ; Київ: ДУТ; Тольятті: ТДУ; Полтава: ПНТУ, 2013. С. 20–21.

18. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в распределенных системах. Проблеми інформатизації: Матеріали другої міжнародної науково-технічної конференції. Київ: ДУТ; Полтава: ПНТУ; Катовице: Катовицький економічний університет; Париж: Університет Париж VII Вєнсент-Сен-Дєні; Білгород: НДУ «БДУ»; Черкаси: ЧДТУ; Харків: ХНДІТМ, 2014. С. 59–60.

19. Волк М. А., Филимончук Т. В. Информационная технология управления заданиями в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам четвертой международной научно-практической конференции (23-25 апреля 2014 г., Смоленск). В 2-х томах. Том 1. Смоленск: Смоленский филиал Российского университета кооперации, 2014. С. 379–383.

20. Филимончук Т. В., Ткачев В. Н. Информационная технология распределения заданий на вычислительные ресурсы в GRID-системах. Информатика, математическое моделирование, экономика: Сборник научных статей по итогам пятой международной научно-практической конференции (11-15 мая 2015 г., Смоленск). В 2-х томах. Том 1 Смоленск: Смоленский филиал Российского университета кооперации, 2015. С. 204–209.

21. Волк М. А., Филимончук Т. В. Информационная технология распределения заданий на вычислительные ресурсы для обработки радиоастрономических

данных в GRID-системах. Проблеми інформатизації: Матеріали третьої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2015. С. 23.

22. Волк М. А., Филимончук Т. В. Использование системы моделирования GRASS в задачах распределения заданий в GRID-системах. 20-й Ювілейний міжнародний форум «Радіоелектроніка та молодь у XXI столітті». Харків: ХНУРЕ, 2016. Т.4. С. 171–172.

23. Volk M. O., Filimonchuk T. V. Information technology for job distribution in GRID-systems. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: матеріали шостої міжнародної науково-технічної конференції. Полтава: ПНТУ; Баку: ВА ЗС АР; Кіровоград: КЛА НАУ; Харків: ДП «ХНДІ ТМ», 2016. С. 45–46.

24. Ткачев В. Н., Филимончук Т. В., Митин Д. Е. Использование информационной технологии распределения заданий при обработке больших массивов данных в виртуальных частных облаках. Информационные системы и технологии: Материалы 5-й международной научно-технической конференции (2-17 сентября 2016 г., Харьков). Х.: ДРУКАРНЯ МАДРИД, 2016. С. 333–334.

25. Волк М. А., Филимончук Т. В. Модифицированный метод Backfill с консервативным резервированием. Проблеми інформатизації: Матеріали четвертої міжнародної науково-технічної конференції. Черкаси: ЧДУТ; Баку: ВА ЗС АР; Бельсько-Бяла: УТіГН; Полтава: ПНТУ; 2016. С. 20.