УДК 681.3

А.Ю. Дорошенко, О.Г. Бекетов, О.А. Яценко, Є.А. Вітряк, Т.О. Павлючин

РОЗРОБКА СЕРВІСНО-ОРІЄНТОВАНИХ ЗАСОБІВ ДЛЯ ЗАПУСКУ ПАРАЛЕЛЬНИХ ПРОГРАМ НА МУЛЬТИПРОЦЕСОРНОМУ КЛАСТЕРІ

Запропоновано підхід до проектування та генерації Grid-сервісів для платформи Globus Toolkit на основі використання алгеброалгоритмічного інструментарію. Розроблено архітектуру та виконано програмну реалізацію Grid-сервісу, що виконує запуск паралельних програм на мультипроцесорному кластері. Розроблений Grid-сервіс застосовано для автоматизації запуску паралельної програми з області метеорологічного прогнозування.

Вступ

На сьогоднішній день як у всьому світі, так і в Україні, є популярними та знаходять своє застосування розподілені комп'ютерні системи на основі використання Grid-технологій, які надають доступ до великих сховищ інформації та інших ресурсів [1]. Головною метою програмування Grid-систем є побудова моделей програмування, інструментальних засобів, методів, які підтримують ефективну розробку переносимих застосувань і дозволяють отримувати високоефективні програми для Grid-систем. Програмування для Grid часто вимагає використання можливостей і властивостей, що лежать за межами простого послідовного програмування, й навіть паралельного або розподіленого програмування. Крім управління операціями над розподіленими структурами даних програміст Grid має керувати обчисленнями в середовищі, що є відкритим, різнорідним та динамічним. Застосування Grid виконуються на різних типах ресурсів, конфігурація яких може змінюватися під час виконання застосувань. Тому крім звичайних обчислень над структурами даних програміст Grid також має проектувати взаємодію між віддаленими сервісами, джерелами даних і ресурсами апаратних засобів. Звідси виникає все більш глибоке розуміння того, що сучасні інструментальні засоби й мови програмування недостатні для підтримки ефективної розробки високопродуктивних програм для Grid-систем. Відмітимо, що зі зростанням складності і динамічності Grid застосувань використання програмних засобів для їх розробки стає суттєвим при настроюванні параметрів застосувань або виявленні дефектів програми.

До інструментальних засобів Grid відноситься, зокрема, Globus Toolkit (GT) [2], призначений для розробки сервісно-орієнтованих розподілених обчислювальних програм та інфраструктур. Для GT існують програмні засоби, що дозволяють спростити розробку Grid-сервісів [3–9]. Наприклад, система Introduce [3, 4] надає користувачу графічний інтерфейс для створення сервісів GT 4.0, а також додавання методів, ресурсів, властивостей та обмежень безпеки. На основі проекту сервісу Introduce генерує каркас клієнтської та серверної частини Grid-застосування. Розроблювачу далі необхідно реалізувати логіку методів сервісу. Однак, написання коду методів є досить складною задачею. У роботі [10] авторами була запропонована автоматизація розробки алгоритмів, що представляють логіку виконання сервісів і програм-клієнтів, а також генерація відповідного коду мовою програмування. Згадана автоматизація здійснюється за допомогою розробленого Інтегрованого інструментарію Проектування та Синтезу програм (ІПС) [11, 12]. У попередній роботі інструментарій ІПС був використаний для розробки Grid-сервісу для GT 4.0.3, який виконував запуск паралельної програми сортування на локальному комп'ютері. Дана робота продовжує згадані дослідження і направлена на розробку сервісно-орієнтованої Grid-програми, що виконує запуск паралельних

програм на віддаленому мультипроцесорному кластері. Проведено експеримент, в якому розроблений Grid-сервіс було застосовано для запуску паралельної програми з області метеорології.

1. Інструментальні засоби автоматизації розробки Grid-сервісів для платформи Globus Toolkit

Globus Toolkit є реалізацією OGSA/OGSI (Open Grid Services Architecture/Open Grid Services Infrastructure) стандарту на архітектуру і функціональність Grid-платформ, яка ґрунтується на технології веб-сервісів [2]. GT забезпечує програмну інфраструктуру, що дає можливість програмам працювати з розподіленими різнорідними обчислювальними ресурсами як з єдиною віртуальною машиною. Інструментарій складається з набору компонентів, що реалізують базові Grid-сервіси, такі як захист, розміщення ресурсу, керування ресурсами і зв'язок.

Для Globus Toolkit існує низка інструментальних засобів, що призначені для автоматизації розробки Grid-сервісів. Стислий огляд деяких з них наведено далі.

Інструментарій Globus Service Build Tools Project [5] містить компонент (розширення) GT4IDE, що вбудовується в середовище розробки програм Eclipse [6], і призначений для полегшення розробки сервісів для GT 4.0. Розширення надає можливість розробнику створити сервіс та додати до нього методи. Система генерує стаб сервісу, який далі потрібно реалізувати розробнику.

У роботах [7, 8] запропоновано метод та систему генерації програмного забезпечення для сервісно-орієнтованих архітектур, що ґрунтується на використанні модельно-орієнтованого підходу MDA (Model Driven Architecture) та Java анотацій.

В роботі [9] розглядається інструментарій автоматизації генерації Grid-сервісів для GT 3.0 із використанням мови UML. Grid-сервіси подаються у вигляді діаграм класів та моделей UML. Система виконує генерацію усіх необхідних файлів, каталогів, вихідного коду та елементів даних розроблюваного сервісу.

Використана у даній роботі система генерації сервісів Introduce [3, 4] має низку переваг над перерахованими вище інструментами. Зокрема, особливостями даної системи є: незалежність від середовища розробки програм; підтримка розробки строго типізованих сервісів; приховування низькорівневих деталей реалізації сервісів; генерація методів як для серверної частини, так і клієнтської. Система Introduce призначена для підтримки трьох основних етапів розробки Grid-сервісів для GT:

1) створення базисної структури сервісу. Розроблювач сервісу описує на верхньому рівні основні властивості сервісу, такі як ім'я та простір імен. Після того як користувач вказав основні властивості, Introduce створює базисну реалізацію сервісу, до якої користувач може потім додавати методи та опції безпеки на етапі зміни сервісу;

2) модифікація сервісу. Етап модифікації дозволяє розроблювачу додавати, видаляти й змінювати методи та властивості сервісу, ресурси, контексти сервісу, а також безпеку на рівні сервісу або методів. На цьому кроці розроблювач може створити строго-типізований інтерфейс сервісу, визначаючи вхідні та вихідні параметри методів сервісу. Після того як операції були додані до сервісу, розроблювач може додавати логіку, що реалізує методи;

3) розгортання. Розроблювач може розгорнути сервіс, що був створений за допомогою Introduce у контейнер Grid-сервісів (наприклад, контейнер сервісів GT).

Для виконання перерахованих трьох кроків розроблювач сервісу використовує графічне середовище Introduce. На основі опису сервісу Introduce генерує каркаси сервісу та відповідної програми-клієнта (мовою Java), які розроблювачу необхідно далі заповнити кодом реалізації.

Отже, система Introduce генерує код сервісу, який не містить реалізації методів.

Створений авторами алгеброалгоритмічний інструментарій [10], що розглядається у наступному розділі, підтримує подальший процес розробки методів – конструювання алгоритмів методів та генерацію відповідного коду мовою програмування.

2. Алгеброалгоритмічний інструментарій проектування та синтезу програм

Розроблена система ІПС є сукупністю програмних засобів, що забезпечують побудову високорівневих специфікацій алгоритмів в режимі порівневого діалогового конструювання, а також генерацію відповідних програм в цільових мовах програмування (C, C++, Java) [11, 12]. В роботі [10] інструментарій було налаштовано на конструювання алгоритмів методів Grid-сервісів і програм-клієнтів, а також генерацію тексту мовою програмування із заповненням каркасного коду, створеного за допомогою програми Introduce. Інструментарій було використано для розробки сервісу, що виконує паралельне сортування на локальному комп'ютері. У даній роботі систему ІПС було застосовано для розробки більш складного сервісу, що виконує запуск паралельних програм на віддаленому мультипроцесорному кластері (див. розділ 3).

В основу інструментарію проектування та генерації програм покладений апарат систем алгоритмічних алгебр (САА) [11], що призначений для формалізованого проектування структурованих послідовних і паралельних алгоритмів та програм. У процесі проектування алгоритмів ІПС інтегрує три форми їх подання – природно-лінгвістичну, алгебраїчну та граф-схемну. Основною формою проектування в ІПС є природно-лінгвістична, яка ґрунтується на алгоритмічній мові САА/1. Основними об'єктами мови САА-схем є абстракції операторів і умов, що можуть бути елементарними (базисними) або складеними. Складені оператори й умови будуються з елементарних за допомогою операцій послідовного і паралельного виконання операторів. До основних операцій відносяться такі:

- `"оператор1" ПОТІМ "оператор2"` (або `"оператор1" ; "оператор2"`) – послідовне виконання операторів;
- `ЯКЩО 'умова' ТО "оператор1" ІНАКШЕ "оператор2" КІНЕЦЬ ЯКЩО` – оператор розгалуження;
- `ПОКИ НЕ 'умова_закінчення_циклу' ЦИКЛ "оператор" КІНЕЦЬ ЦИКЛУ` – оператор циклу;
- `ПАРАЛЕЛЬНО (i = 1, ..., n) ("оператор")` – асинхронне виконання `n` операторів (потоків).

Основними компонентами інструментарію є:

- конструктор, призначений для діалогового проектування схем алгоритмів;
- генератор коду мовою програмування на основі алгоритмів;
- редактор граф-схем;
- база даних алгеброалгоритмічних специфікацій, у якій зберігається текст конструкцій САА і базисних елементів схем та їх програмні реалізації.

За допомогою конструктора користувач системи виконує проектування алгоритмів у діалоговому режимі, вибираючи зі списку заздалегідь заготовлені конструкції (операції САА, базисні оператори та умови). Процес проектування алгоритмів є порівневим (здійснюється зверху-вниз) і представлений у вигляді дерева конструювання. На основі отриманої у процесі конструювання схеми, а також реалізацій елементарних операторів та умов цільовою мовою програмування, що зберігаються в БД ІПС, система ІПС виконує генерацію програми (рис. 1). На вхід генератора надходить також файл, що містить каркасний опис основного класу програми (без реалізацій методів), у який виконується підстановка коду. У випадку розробки сервісно-орієнтованої програми для GT, для генерації каркасного коду використовується інструментарій Introduce. Спосіб підстановки згенерованого коду для методів у каркасний файл вказується у параметрах генерації.

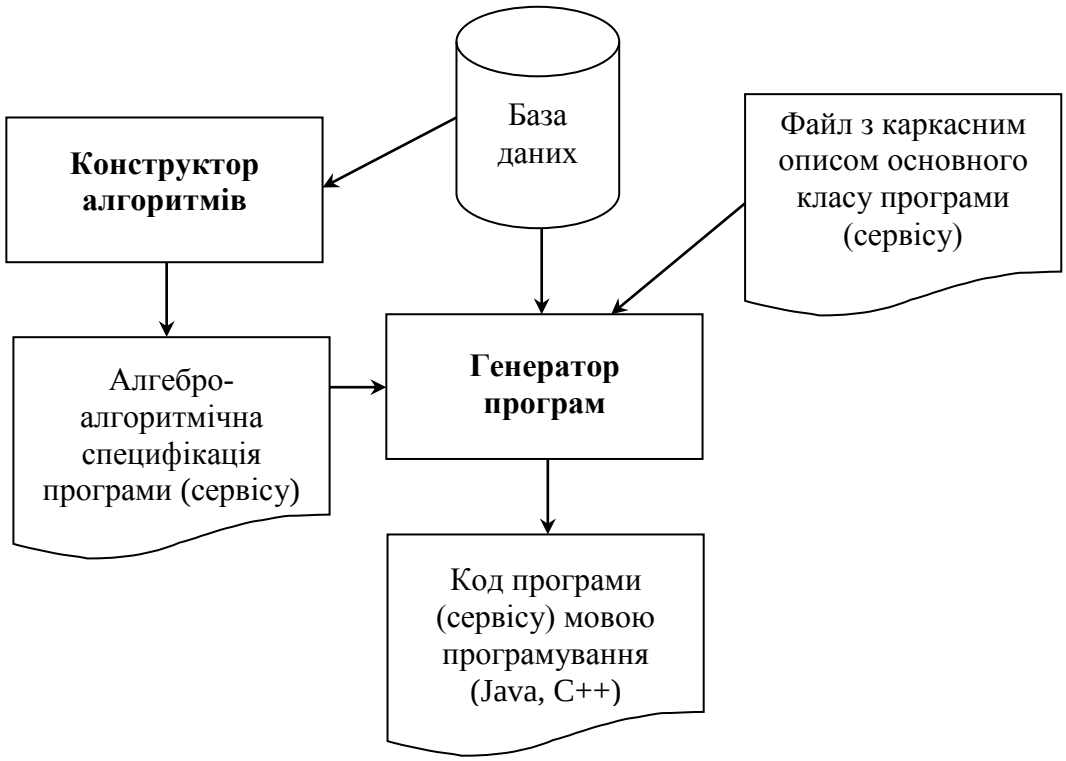

Рис. 1. Процес проектування та генерації програм у системі ІПС

3. Розробка Grid-сервісу для запуску паралельних програм на мультипроцесорному кластері

У даній роботі на основі використання алгеброалгоритмічного інструментарію виконане проектування та генерація коду Grid застосування, що призначене для запуску програм на віддаленому комп'ютері (мультипроцесорному кластері). Структура і схема роботи розробленого Grid-застосування подана на рис. 2. Grid-програма складається з двох основних частин: програми-клієнта та Grid-сервіса (ISS_Service1). Обидві частини програми розміщені на комп'ютерах із встановленою системою Globus Toolkit 4.0.3.

За допомогою графічного інтерфейсу програми-клієнта користувач передає Grid-сервісу необхідні дані про програму, яку необхідно запустити на віддаленому комп'ютері:

- ім'я хосту віддаленого комп'ютера (кластера), ім'я користувача та пароль;
- робочий каталог на кластері, у якому розміщена програма для запуску;
- назва файлу скрипта (командного файлу) для виконання програми;
- список значень параметрів командного рядка для скрипта (наприклад, назва розділу кластера та кількість паралельних процесів).

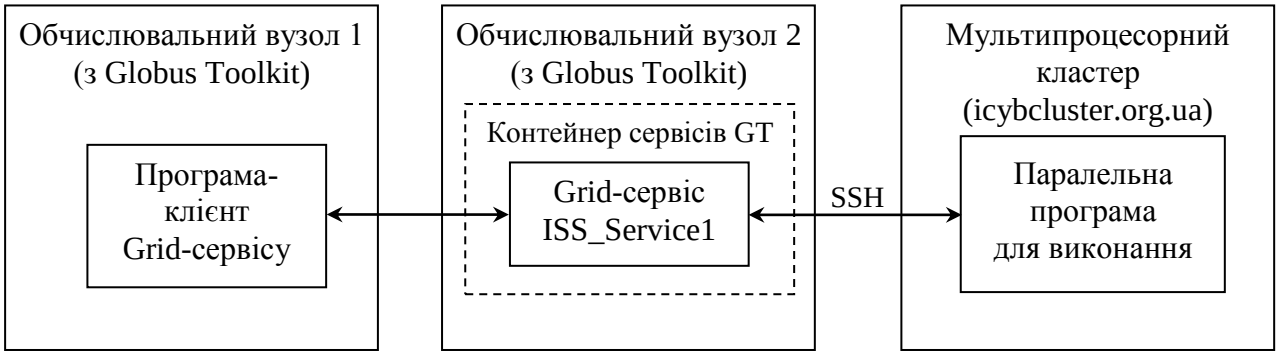

Рис. 2. Структура і схема роботи розробленої Grid програми

Перераховані вище дані, які описують задачу для запуску, зберігаються у текстовому файлі на обчислювальному вузлі 1. Grid-сервіс, відповідно до переданих йому програмою-клієнтом даних з описом задачі, виконує запуск скрипта на вказаному комп'ютері та повертає клієнту текст результату виконання цього скрипта.

Розроблену Grid програму було застосовано для виконання програм на суперкомп'ютері Інституту кібернетики імені В.М. Глушкова НАН України [13]. Відмітимо, що на згаданому кластері встановлено систему керування ресурсами кластера SLURM (Simple Linux Utility for Resource Management) [14]. В рамках цієї системи задачі, що запускаються користувачами кластера, розміщуються в черзі завдань, і кожній задачі присвоюється свій унікальний ідентифікатор. Скрипт для запуску програми, що виконується розробленим Grid-сервісом, містить SLURM команду `sbatch`, яка додає задачу в чергу на виконання [14]. Окрім занесення задач у чергу, Grid-сервіс також може виконувати низку додаткових команд:

- видачу інформації про стан запущених задач (вміст черги);
- відміну задачі;
- перегляд списку файлів у робочому каталозі;
- перегляд вмісту файлів та результатів виконаних задач.

Виконання перерахованих команд також ініціюється програмою-клієнтом.

Grid-сервіс та програму-клієнт було реалізовано мовою Java. На рівні програмного коду Grid-сервіс ISS_Service1 є класом, що містить у собі три методи:

```
String connectToServer(
String hostName,
String userName,
String password,
int client_number)

String executeProgram(
String workingDirectory,
String fileToExecute,
String cmdLineParameters,
boolean locally,
boolean forWindows,
int client_number)

void disconnect(
int client_number)
```

Перераховані методи виконують такі функції.

Метод `connectToServer` виконує з'єднання із сервером `hostName` для користувача `userName`. При успішно встановленому з'єднанні, поточній програмі-клієнту присвоюється номер `client_number`, і інформація про поточну сесію заноситься у масив сесій під відповідним номером. У випадку успішного з'єднання, ця функція повертає текст із номером клієнта, у протилежному випадку метод повертає текст з повідомленням про помилку з'єднання із сервером.

За допомогою методу `executeProgram` виконується запуск скрипта (або команди) `fileToExecute` на сервері у вказаному робочому каталозі для клієнта з номером `client_number`. Метод повертає текст з результатами виконання скрипта.

Метод `disconnect` закриває з'єднання із сервером для вказаної програми-клієнта.

У Grid-сервісі передбачена також можливість запуску програм на локальному комп'ютері, тобто на тому ж, де розташований Grid-сервіс. У цьому випадку методи `connectToServer` та `disconnect` не використовуються, а параметр `locally` у методі `executeProgram` повинен бути встановлений в істинне значення.

Каркас програмного коду мовою Java для Grid-сервісу ISS_Service1 був згенерований за допомогою програми Introduce. Подальше проектування алгоритмів методів мовою CAA та генерація відповідного коду виконана в інструментарії ІПС. На рис. 3 показана CAA-схема Grid-сервісу, побудована в конструкторі алгоритмів інструментарію ІПС. В схемі назви методів зі списком формальних параметрів відділені від реалізації методів ланцюжками символів "=". Текст базисних і складених операторів записаний у подвійних лапках, а базисні умови обрамлені одинарними лапками.

На рис. 4 показано приклад алгоритму виклику методів розглянутого Grid-сервісу у клієнтській програмі. Приклад значень параметрів, що передаються програмі-клієнту, розглядається у розділі 4.

```
СХЕМА ISS_SERVICE1
     "Алгоритми функціонування методів Грід-сервіса
     ISS_Service1. Сервіс виконує запуск програми (або команди)
     на віддаленому або локальному комп'ютері."
   КІНЕЦЬ КОМЕНТАРІЮ

"Дані сервісу"
==== "Визначити масив віддалених сесій (jtv)"

"connectToServer (hostName, userName, password, client_number)"
==== "Визначити масив (res_str) типу (String) зі значеннями
      ({ "", "", "" })";
      "Визначити змінну (host) типу (String) =
      = (userName + "@" + hostName)";
      " Створити нову сесію для клієнта з номером
      (client_number)";
      "Занести число (client_number) в рядок (res_str)";
      ЯКЩО 'Сесія для клієнта (client_number) успішно створена'
      ТО "Встановити з'єднання з сервером (host) для користувача
         (client_number)";
          ЯКЩО 'З'єднання с сервером (host) встановлено'
          ТО "Вивести текст ("Сервіс з'єднався із сервером")
             і змінну (host)"
          ІНАКШЕ "Занести повідомлення про помилку з'єднання в
                 рядок (res_str)";
                  "Видалити сесію для клієнта (client_number)";
                  "Вивести текст (res_str)";
          КІНЕЦЬ ЯКЩО
      ІНАКШЕ
          "Занести повідомлення про помилку створення сесії в
          рядок (res_str)";
          "Вивести текст (res_str)";
      КІНЕЦЬ ЯКЩО
     "Повернути значення (res_str)";

"executeProgram (workingDirectory, fileToExecute, cmdLineParameters, locally,
forWindows, client_number)"
==== "Визначити змінні (res_str, command_str)
     типу (String) = ("")";
     "Підготувати текст команди (command_str) на основі
     параметрів workingDirectory)(fileToExecute)
     (cmdLineParameters)(locally)";
     ЯКЩО 'Виконання на віддаленому комп'ютері'
     ТО "Виконати команду (command_str) на віддаленому
         комп'ютері для клієнта (client_number) і записати
         результат в рядок (res_str)"
     ІНАКШЕ "Виконати команду (command_str) на локальному
           комп'ютері і записати результат в рядок (res_str)"
     КІНЕЦЬ ЯКЩО;
     "Підготувати текст результатів виконання команди (res_str)
     на основі (res_str)(command_str)";
     "Повернути значення (res_str)"

"disconnect (client_number)"
==== "Розірвати  з'єднання з сервером для клієнта
     (client_number)";
     "Видалити сесію для клієнта (client_number)";

  КІНЕЦЬ СХЕМИ ISS_SERVICE1
```

Рис. 3. САА-схема Grid-сервісу ISS_Service1

```
СХЕМА ISS_SERVICE1_CLIENT ====
     "Приклад алгоритму функціонування клієнта Грід-сервіса
      ISS_Service1"
   КІНЕЦЬ КОМЕНТАРІЮ

"Основна схема"
==== "Зчитати вхідні параметри для запуску програми";
     "res := connectToServer(hostName, userName, password,
                             client_number)";
     ЯКЩО 'З'єднання із сервером встановлено (res)'
     ТО
        ПОКИ НЕ 'Завершення роботи із сервером'
           "Зчитування введеної команди";
           "res_text := executeProgram(workingDirectory,
                        fileToExecute, cmdLineParameters,
                        locally, forWindows, client_number)";
           "Вивести текст (res_text)";
        КІНЕЦЬ ЦИКЛУ
        "disconnect(client_number)"
     КІНЕЦЬ ЯКЩО
```

Рис. 4. Приклад спрощеної САА-схеми клієнта Grid-сервісу ISS_Service1

Типова послідовність взаємодії програми-клієнта із Grid-сервісом є такою. На основі переданих користувачем даних про програму, яку необхідно запустити, програма-клієнт спочатку викликає метод Grid-сервіса `connectToServer`. Далі, у випадку успішного з'єднання із сервером (кластером), клієнт викликає метод `executeProgram`. Згаданий метод може бути виконаний необмежену кількість разів. При цьому є можливість змінювати вхідні параметри виконуваної задачі, наприклад, розмір оброблюваних даних, кількість використовуваних процесів, назву розділу кластера тощо. Для завершення роботи із віддаленим сервером програма-клієнт викликає метод `disconnect`.

Після конструювання наведеної вище САА-схеми Grid-сервісу в системі ІПС було виконано генерацію програмного коду сервісу, яка здійснювалася таким чином. Спочатку на вхід генератора ІПС було передано файл, що містив каркасний опис (без реалізацій методів та даних) основного класу сервісу, створеного за допомогою програми Introduce. Далі в параметрах генерації ІПС було вказано співставлення імен складених операторів САА-схеми з іменами методів сервісу. Далі ІПС виконала генерацію тексту даних та методів на основі САА-схеми та автоматично підставила в каркас Grid-сервісу. Для реалізації мовою Java вищерозглянутих методів сервісу було використано Java бібліотеку JSch [15], призначену для програмування взаємодії з віддаленим сервером по SSH.

4. Приклад використання розробленого Grid-сервісу

Як експеримент розглянуто у попередньому розділі Grid-сервіс було використано для автоматизації запуску послідовної та паралельної програм з області метеорологічного прогнозування, що були розроблені в роботі [16]. Згадані програми призначені для вирішення задачі математичного моделювання циркуляції атмосфери. Мета експерименту полягала в тому, щоб отримати час виконання цих програм на різних розмірах оброблюваних даних та обчислити значення мультипроцесорного прискорення.

Розглянемо стисло задачу, що розв'язується в згаданих програмах. Задача подається за допомогою сукупності рівнянь конвективної дифузії [16]

$$\frac{\partial u}{\partial t} + v_1 \frac{\partial u}{\partial x_1} + v_2 \frac{\partial u}{\partial x_2} =$$

$$= \frac{\partial}{\partial x_1}\left(\mu_1 \frac{\partial u}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\mu_2 \frac{\partial u}{\partial x_2}\right) + f\,, \quad (1)$$

при $(x_1, x_2) \in \Omega;\ t \in [0;10]$,

$$u(0, x_1, x_2) = \sin(x_1 + x_2),$$

$$\text{при } (x_1, x_2) \in \Omega,\ t = 0, \qquad (2)$$

$$u(t, x_1, x_2) = u_A(t, x_1, x_2),$$

$$\text{при} (x_1, x_2) \in \partial\Omega,\ t \in [0;10], \qquad (3)$$

де

$$\Omega = [0,1] \times [0,1],$$

$$v_k = \sin(x_k),$$

$$\mu_k = 0.001 + 0.1 * \sin^2(x_k) > 0,$$

$$f(t, x_1, x_2) =$$

$$= (v_1 + v_2 - (1 + 0.1 * (\sin(2x_1) + \sin(2x_2)))) *$$

$$* \cos(x_1 + x_2 - t) + (\mu_1 + \mu_2) * \sin(x_1 + x_2 - t).$$

Тут $u = u(t, x_1, x_2)$ – залежна функція; t – час; x_1, x_2 – координати; Ω – двовимірна просторова область визначення задачі; $f = f(t, x_1, x_2)$ – вільний член рівняння; v_k – коефіцієнт конвекції; μ_k – коефіцієнт дифузії.

Для розв'язання задачі (1)–(3) в програмах застосовується скінченно-різницевий чисельний метод, що детально розглянутий в [16, 17]. Аналітичний розв'язок цієї задачі має вигляд $u_A(t, x_1, x_2) = \sin(x_1 + x_2 - t)$.

Введемо позначення для координат $x_1 \equiv x$, $x_2 \equiv y$, яке буде використовуватися далі. Для вирішення задачі просторова область Ω розбивається на S_x та S_y рівномірних кроків (вузлів) по осям x та y відповідно. Загальна кількість вузлів становить $S_x \times S_y$. Розпаралелювання обчислень для даної задачі ґрунтується на декомпозиції області $S_x \times S_y$ на підмножини та паралельному розв'язанні сукупності підзадач, що визначені на цих підмножинах.

Для розглянутої задачі в [16] було розроблено послідовну та паралельну програмні реалізації мовою C++ для виконання на CPU та GPU відповідно. Реалізація паралельної програми виконана із використанням технології CUDA [18], що дозволяє проводити обчислення на графічних прискорювачах NVIDIA. Розроблений у даній роботі Grid-сервіс було використано для запуску цієї паралельної програми на кластері Інституту кібернетики імені В.М. Глушкова НАН України [13]. Розрахунки проводились із використанням графічного прискорювача NVIDIA Tesla M2075 (448 обчислювальних ядер) та процесора Intel Xeon E5-2670. Для виконання паралельної програми програмі-клієнту були передані такі значення параметрів:

```
hostName = icybcluster.org.ua
locally = false
workingDirectory =
~/helena/convdiff/Meteo_CUDA
fileToExecute = sh run
cmdLineParameters = scit4 128 128
```

Запуск паралельної програми здійснювався за допомогою скрипта `run` (командного файлу оболонки Bash), що розміщений в каталозі, вказаному у змінній `workingDirectory`. Скрипту `run` передаються три параметри командного рядка, значення яких вказуються у змінній `cmdLineParameters`. Перший параметр це назва розділу кластера, на якому необхідно виконати програму (у даному випадку – `scit4`). Другий та третій параметри скрипта – величини S_x та S_y відповідно. Значення S_x та S_y обиралися кратними 64, і такими, що $S_x = S_y$. Чисельні експерименти було виконано для значень розміру задачі від $S_x \times S_y = 64 \times 64$ до $S_x \times S_y = 2048 \times 2048$.

На рис. 5 показана копія екрану програми-клієнта Grid-сервісу із параметрами запуску для задачі, що розглядається. У першому текстовому полі задається ім'я текстового файлу з описом задачі, у якому зберігаються параметри для запуску. У наступних трьох полях вказуються значення цих параметрів, зчитані з файлу:

1) назва робочого каталогу на кластері;

2) назва файлу скрипта;

3) значення параметрів командного рядка.

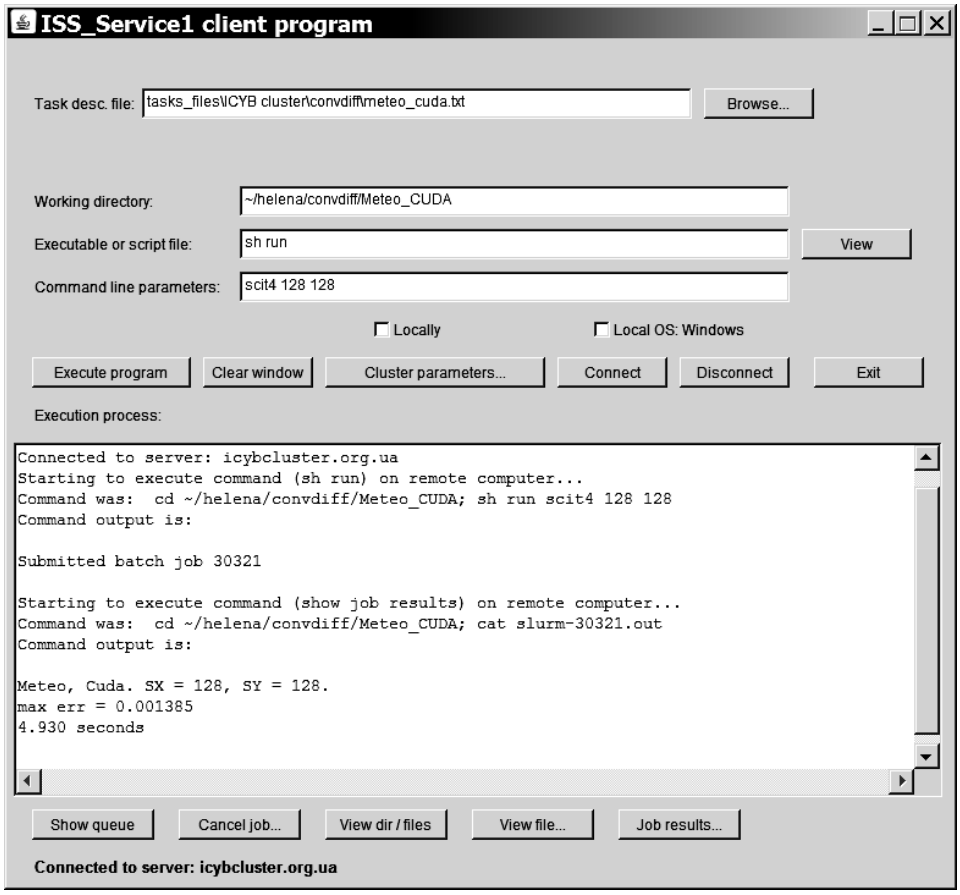

Рис. 5. Вікно програми-клієнта Grid-сервісу із результатами тестового запуску задачі на мультипроцесорному кластері

У полі “Execution process” наведено результати виконання двох команд:

1) занесення задачі у чергу в результаті виконання скрипта `run`. У даному випадку задача була занесена у чергу під номером 30321;

2) перегляд файлу (slurm-30321.out) з текстом, який видає паралельна програма в результаті виконання. Цей текст включає значення параметрів S_x та S_y, величину похибки обчислень та час виконання в секундах. Інші результати розв’язання задачі зберігаються у додаткових файлах.

Як уже згадувалося у розділі 3, Grid-сервіс також забезпечує виконання команд перегляду вмісту черги та відміни виконання завдань.

Аналогічним чином також було здійснено запуск послідовної програми на кластері. У таблиці наведено отримані за допомогою Grid-сервісу значення часу виконання послідовної T_s та паралельної T_p програм у секундах при різних значеннях розміру задачі $S_x \times S_y$, а також обчислене мультипроцесорне прискорення T_s / T_p. На рис. 6 показано діаграму залежності мультипроцесорного прискорення від розміру задачі $S_x \times S_y$.

Менший рівень мультипроцесорного прискорення при невеликих значеннях S_x та S_y пояснюється неповною завантаженістю відеокарти, через що не досягається максимальна ефективність. Максимальне мультипроцесорне прискорення обчислень при $S_x \times S_y = 2048 \times 2048$ становить 69 разів.

Отже, розроблена Grid програма забезпечує автоматизацію запуску та перегляду результатів виконання програм на віддаленому кластері. В подальшому до Grid-сервісу можуть бути додані функції автоматичної обробки отримуваних часових та інших результатів виконання паралельних програм.

Таблиця. Час виконання послідовної T_s та паралельної T_p програм моделювання циркуляції атмосфери для різних значень $S_x \times S_y$, а також значення мультипроцесорного прискорення T_s / T_p

S_x (S_y)	$S_x \times S_y$	T_s, с	T_p, с	T_s / T_p
64	4096	29,000	2,380	12,18
96	9216	53,620	3,240	16,55
128	16384	96,420	4,930	19,56
160	25600	147,960	5,850	25,29
192	36864	255,740	7,040	36,33
288	82944	484,720	12,550	38,62
352	123904	719,950	17,660	40,77
384	147456	850,420	19,840	42,86
416	173056	1040,750	22,580	46,09
480	230400	1344,380	28,020	47,98
544	295936	1709,290	32,970	51,84
640	409600	2400,180	44,270	54,22
768	589824	3439,40	57,93	59,37
1024	1048576	6271,48	93,50	67,07
2048	4194304	25138,65	363,45	69,17

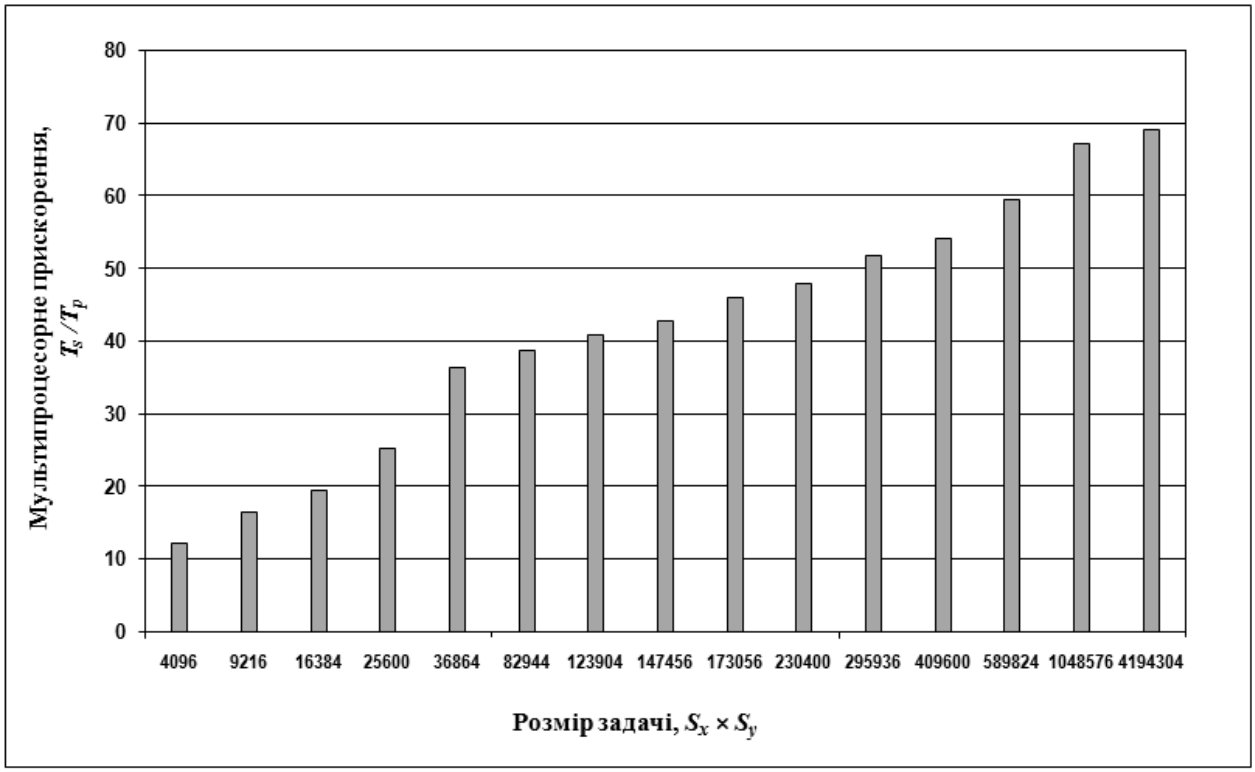

Рис. 6. Залежність мультипроцесорного прискорення T_s / T_p від розміру задачі $S_x \times S_y$ для паралельної програми моделювання циркуляції атмосфери

Висновки

Запропоновано підхід до проектування та генерації Grid-сервісів для платформи Globus Toolkit на основі застосування алгеброалгоритмічного інструментарію. Із використанням інструментарію розроблено алгоритми та виконано генерацію програмного коду Grid-сервісу, що виконує запуск паралельних програм на мультипроцесорному кластері. Особливістю інструментарію є застосування методу конструювання синтаксично правильних програм, який виключає можливість появи синтаксичних помилок у процесі проектування алгоритмів. Іншою особливістю системи є проектування програм із використанням мовних конструкцій, близьких до природної мови. Створену Grid-програму застосовано для автоматизації запуску та перегляду результатів виконання паралельної програми з області метеорологічного прогнозування на відеографічному прискорювачі, що входить до складу кластера. Перспективи виконаної роботи полягають в подальшому розвитку розробленого Grid-сервісу. Розвиток може включати розробку програмних засобів автоматичної обробки результатів виконання паралельних програм за допомогою сервісу.

1. *Jacob B., Brown M., Fukui K., Trivedi N.* Introduction to Grid Computing [Електронний ресурс]. – Режим доступу: http://www.redbooks.ibm.com/redbooks/pdfs/sg246778.pdf. – 20.03.2014 р.
2. *Globus Toolkit* [Електронний ресурс]. – Режим доступу: http://www.globus.org. – 20.03.2014 р.
3. *Introduce* [Електронний ресурс]. – Режим доступу: http://dev.globus.org/wiki/Incubator/Introduce. – 20.03.2014 р.
4. *Hastings S., Oster S., Langella S., Ervin D., Kurc T., Saltz J.* Introduce: An Open Source Toolkit for Rapid Development of Strongly Typed Grid Services // Journal of Grid Computing. – 2007. – Vol. 5, N 4. – P. 407–427.
5. *GSBT* – Globus Service Build Tools [Електронний ресурс]. – Режим доступу: http://sourceforge.net/projects/gsbt/ – 20.03.2014 р.
6. *Eclipse* – The Eclipse Foundation open source community website [Електронний ресурс]. – Режим доступу: http://www.eclipse.org. – 20.03.2014 р.
7. *Smith M., Friese T., Freisleben B.* Model driven development of service oriented Grid applications // Proc. of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW‘06), 2006. – P. 139–146.
8. *Friese, T., Smith M., Freisleben B.* Grid development tools for Eclipse // Proc. of the Eclipse Technology eXchange work-shop (eTX) at ECOOP 2006, Nantes, France, 2006.
9. *Mizuta S., Huang R.* Automation of Grid service code generation with AndroMDA for GT3 // Proc. of the 19th International Conference on Advanced Information Networking and Applications (AINA’05), 2005. – P. 417– 420.
10. *Дорошенко А.Е., Яценко Е.А.* Средства сервисно-ориентированного программирования параллельных программ // Проблеми програмування. – 2009. – № 2. – С. 12–21.
11. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 631 с.
12. *Яценко Е.А.* Интеграция инструментальных средств алгебры алгоритмов и переписывания термов для разработки эффективных параллельных программ // Проблеми програмування. – 2013. – № 2. – С. 62–70.
13. *Суперкомп’ютер* Інституту кібернетики НАН України [Електронний ресурс]. – Режим доступу: http://icybcluster.org.ua. – 20.03.2014 р.
14. *Slurm* Workload Manager [Електронний ресурс]. – Режим доступу: http://www.schedmd.com/slurmdocs/ – 20.03.2014 р.
15. *JSch* – Java Secure Channel – JCraft [Електронний ресурс]. – Режим доступу: http://www.jcraft.com/jsch/ – 20.03.2014 р.
16. *Прусов В.А., Дорошенко А.Ю., Кацалова Л.М., Бекетов О.Г.* Паралельні методи розв’язання двовимірної задачі конвективної дифузії // Вісник Київського національного університету імені Тараса Шевченка. – 2013. – № 4.
17. *Прусов В.А., Дорошенко А.Е., Черныш Р.И.* Метод численного решения многомерной

задачи конвективной диффузии // Кибернетика и системный анализ. – 2009. – № 1. – С. 100–107.
18. *nVidia's* CUDA – NVIDIA Developer Zone [Електронний ресурс]. – Режим доступу: https://developer.nvidia.com/category/zone/cuda-zone – 20.03.2014 р.

Одержано 25.03.2014

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії комп'ютерних обчислень Інституту програмних систем НАН України,

Бекетов Олексій Геннадійович,
аспірант,

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник,

Вітряк Євген Андрійович,
магістрант Національного технічного університету України «КПІ»,

Павлючин Тарас Олександрович,
магістрант Національного технічного університету України «КПІ».

Місце роботи авторів:

Інститут програмних систем
НАН України.
Тел.: (044) 526 3559.
E-mail: dor@isofts.kiev.ua
beketov.oleksii@gmail.com
oayat@ukr.net
pavlyuchin@ukr.net