УДК 681.3

А.Ю. Дорошенко, П.А. Іваненко, О.М. Овдій, О.А. Яценко

АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ ПРОГРАМ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ МЕТЕОРОЛОГІЧНОГО ПРОГНОЗУВАННЯ

Розроблено засіб автоматизованого конструювання паралельного коду для середовища OpenMP на основі високорівневих алгебро-алгоритмічних специфікацій. Застосування засобу демонструється на прикладі задачі моделювання циркуляції атмосфери, що представлений як сервіс у складі Інтернет-порталу з надання послуг метеопрогнозу. Здійснена генерація програмного коду та наведено результати експерименту з виконання розробленої паралельної програми прогнозування на мультипроцесорній платформі.

Ключові слова: паралельні обчислення, метеорологічне прогнозування, синтез програм, алгебра алгоритмів, Інтернет-портал.

Вступ

Системи прогнозування з року в рік стають все більш складними, використовувані ними математичні моделі вдосконалюються, обсяги обчислювальних даних стрімко зростають. Не винятком є і задача метеорологічного прогнозування. Враховуючи великий попит на метеорологічні прогнози в різних галузях людської діяльності, проектування та розробка програм для розв'язання цієї задачі є надзвичайно актуальними. Крім того, достовірність та оперативність отримання даних метеорологічних прогнозів мають велике, а іноді і критичне значення, що висуває високі вимоги до швидкісних характеристик таких програм.

Задача метеорологічного прогнозування є складною прикладною обчислювальною задачею з високими вимогами до точності отриманих результатів та жорсткими часовими обмеженнями. Великий обсяг обчислень над великими масивами даних потребує впровадження паралельних реалізацій та забезпечення їх оптимізації.

У попередній роботі авторів [1] було запропоновано реалізацію Інтернет-порталу для надання послуг метеорологічного прогнозування, яка поєднує комплексність використання адекватних фізичних моделей атмосферних процесів з ефективними обчислювальними схемами та методами програмування високопродуктивних обчислень на мультипроцесорних системах, що дає змогу досягати належного ступеня точності, повноти та своєчасності інформації, необхідної для задоволення потреб широкого кола користувачів. Дана робота поширює можливості попередньої на середовище програмування OpenMP з використанням більш потужної мультипроцесорної техніки та розширеної постановки прикладної задачі.

Одним з перспективних напрямів у розробці та дослідженні систем паралельних і розподілених обчислень нині є побудова програмних абстракцій у вигляді алгебро-алгоритмічних мов і моделей, що ставить своєю метою розвиток архітектурно- і мовно-незалежних засобів програмування для мультипроцесорних обчислювальних систем і мереж. У роботах [2–9] були запропоновані теорія, методологія та інструментарій для автоматизованого проектування паралельних програм, що ґрунтуються на засобах високорівневої алгеброалгоритмічної формалізації та автоматизації перетворень програм. Зокрема, розроблено інструментальну систему автоматизації програмування, названу Інтегрованим інструментарієм Проектування та Синтезу програм (ІПС) [2–6]. В системі спільно використовуються три форми подання алгоритмічних знань про предметні області: аналітична (формула в алгебрі алгоритмів), природно-лінгвістична (текстова) та графова (граф-схеми). ІПС ґрунтується на методі діалогового конструювання синтаксично правильних програм,

що орієнтований на виключення виникнення синтаксичних помилок в процесі проектування. Інструментарій забезпечує покрокову розробку програм, починаючи від формальної специфікації і закінчуючи кодом цільовою мовою програмування (Java, C++). У роботах [7–9] розглядається нова версія системи синтезу програм – Онлайновий Діалоговий конструктор Синтаксично Правильних програм (ОДСП). Особливість інструментарію ОДСП полягає у використанні Web-технологій та у розподіленій архітектурі системи.

У попередніх роботах [2–9] системи ІПС та ОДСП були застосовані для розробки паралельних застосувань для багатоядерних центральних процесорів та графічних прискорювачів, а також розподілених та сервісно-орієнтованих програм для Грід. Розроблена інтеграція ІПС та системи переписувальних правил TermWare [5, 10] для автоматизації трансформації паралельних програм. Предметною областю застосування інструментальних засобів, зокрема, є задача метеорологічного прогнозування. Наприклад, у роботі [8] виконане автоматизоване проектування паралельної програми для розв'язання двовимірної задачі конвективної дифузії, яка виникає при математичному моделюванні циркуляції атмосфери в метеорології. У даній статті наведено результати застосування інструментарію для генерації паралельної програми для тривимірного випадку згаданої задачі [11–13]. Розроблена багатопотокова програма реалізована мовою C++ і використовує засоби OpenMP [14]. У роботі також виконане проектування сервісу, який використовує розроблену паралельну програму, і входить до складу вищезгаданого Інтернет-порталу з надання послуг метеорологічного прогнозування.

Запропонований у даній статті підхід є близьким до робіт, що відносяться до алгебраїчного програмування [15], синтезу програм на основі високорівневих специфікацій [16, 17], а також автоматизованої генерації OpenMP програм [18–21] та Java застосувань [22–24]. Відмінність нашого підходу полягає у використанні специфікацій алгебри Глушкова, поданих у природно-лінгвістичній формі, що полегшує розуміння алгоритмів і досягнення необхідної якості програм. Іншою перевагою розроблених засобів є застосування методу діалогового конструювання синтаксично правильних програм, що виключає можливість виникнення синтаксичних помилок у процесі проектування схем.

1. Алгебра алгоритмів та інструментальні засоби автоматизованої розробки програм

В основу пропонованого підходу до проектування паралельних програм покладений апарат систем алгоритмічних алгебр (САА) та їх модифікацій [2]. Модифіковані САА (САА-М) призначені для формалізації процесів мультиобробки, що виникають при конструюванні програмного забезпечення в мультипроцесорних системах. На САА-М ґрунтуються розроблені інструментальні засоби автоматизованого проектування та генерації програм [2–9].

1.1. Основні операції алгебри алгоритмів. САА-М є двоосновною алгеброю $< Op, \mathrm{Pr}; \Omega >$, де Op – множина операторів; Pr – множина логічних умов (предикатів); Ω – сигнатура, що складається з логічних операцій (диз'юнкції, кон'юнкції, заперечення, лівого множення оператора на умову) та операторних операцій (композиції, альтернативи, циклу та ін.). Оператори та предикати можуть бути базисними або складеними.

На САА-М ґрунтується алгоритмічна мова САА/1 [2], призначена для багаторівневого структурного проектування і документування послідовних та паралельних алгоритмів і програм. Перевагою її використання є можливість опису алгоритмів у природно-лінгвістичній формі, зручній для людини, що полегшує досягнення необхідної якості програм. Подання операторів мовою САА/1 називаються САА-схемами.

Далі наведено перелік назв та специфікації основних операторних операцій сигнатури САА-М, поданих у природньо-лінгвістичній формі.

1. Композиція (послідовне виконання операторів):

оператор 1 ПОТІМ *оператор* 2

або

оператор 1; *оператор* 2

2. Альтернатива (умовний оператор):

ЯКЩО *умова* ТО
  оператор 1
ІНАКШЕ
  оператор 2
КІНЕЦЬ ЯКЩО

3. Цикл:

ПОКИ *умова виконання циклу*
ЦИКЛ *оператор*
КІНЕЦЬ ЦИКЛУ

4. Операція виконання одного з *n* операторів за істинності відповідної умови:

ВИБІР ([*умова* 1] → *оператор* 1,
  [*умова* 2] → *оператор* 2,
  ...
  [*умова n*] → *оператор n*)

5. Асинхронна диз'юнкція – паралельне виконання *n* операторів (потоків), де *i* – номер потоку:

ПАРАЛЕЛЬНО (*i* = 0, ..., *n*–1)
(
  оператор
)

6. Синхронізатор, що виконує затримку обчислень доти, поки значення умови не стане істинним:

ЧЕКАТИ *умова*

Приклад використання наведених конструкцій наведено у розділі 2.

Відмітимо, що операції САА-М також можуть бути подані англійською мовою, як, наприклад, у роботі [9].

1.2. Додаткові операції алгебри для об'єктно-орієнтованої парадигми. У сигнатуру САА-М також входять операції, призначені для формалізації основних понять об'єктно-орієнтованого програмування (класів, об'єктів та ін.). Відмітимо, що для сумісності з мовою програмування Java в сигнатуру включені операції для визначення анотацій. Анотації є спеціальною формою синтаксичних метаданих, яка може бути додана у програмний код, і використовується для його аналізу, компіляції або виконання [25]. Анотованими можуть бути пакети, класи, методи, змінні та параметри. Далі наведено перелік основних операцій САА-М для визначення класів, що використовуються у даній роботі.

1. Визначення анотованого класу (в узагальненому вигляді):

Annotated class X implements Y (
  Class name, *Interface name*)
<Annotations>
  Annotation (*Annotation text*);
  Annotation (*Annotation text*);
  ...
<Fields>
  operators
<Methods>
  operators

де *Class name* – ім'я класу, який реалізує інтерфейс *Interface name*; <Annotations>, <Fields>, <Methods> – рядки, після яких необхідно вказати оператори визначення анотацій, полів даних та методів класу, відповідно; *Annotation text* – текст анотації.

2. Анотації специфікуються за допомогою конструкції

Annotation (*Annotation text*),

де *Annotation text* – параметр, у якому зазначається текст анотації.

Прикладом визначення анотації для класу може бути конструкція:

Annotation (*Service*).

3. Ініціалізоване поле даних:

Field initialized (*Modifiers*, *Field type*,
  Field name)
<Initialization>
  operator

Ця конструкція використовується для визначення поля класу, якому присвоюється певне початкове значення. Тут *Modifiers* – список модифікаторів доступу (наприклад, public, static, final і т. п.); *Field*

type – тип даних поля; *Field name* – назва поля; <Initialization> – рядок, після якого потрібно вказати оператор або вираз, значення якого буде використане для ініціалізації поля.

4. Визначення анотованого поля даних:

Field annotated (*Modifiers*, *Field type*, *Field name*)
<Annotations>
Annotation (*Annotation text*);
Annotation (*Annotation text*);
...

Тут словом annotated позначене поле, відмічене анотаціями (наприклад, *Resource*), які необхідно вказати після рядка <Annotations>.

5. Визначення анотованого методу:

Method annotated (*Return type*, *Method name*, *Parameters list*),
<Annotations>
Annotation (*Annotation text*);
Annotation (*Annotation text*);
...
<Body>
operators

Тут *Return type* – тип значення, яке повертає метод; *Method name* – назва методу; *Parameters list* – список формальних параметрів; <Annotations> – рядок, після якого зазначаються анотації до методу (наприклад, *Override*, *Transactional*); <Body> – рядок, після якого вказуються оператори тіла метода.

6. Конструкція для виклику методу екземпляра (об'єкта) класу має вигляд:

Call instance method (*Instance name*, *Method name*, *Arguments*),

де *Instance name* – ідентифікатор екземпляра класу; *Method name* – назва методу класу; *Arguments* – список фактичних параметрів методу.

Приклад використання операцій, розглянутих у даному підрозділі, наведено у розділі 3.

1.3. Інструментальні системи автоматизованої розробки програм. Розроблені інструментальні засоби підтримки проектування та генерації програм ґрунтуються на використанні розглянутих засобів САА-М та методу діалогового конструювання синтаксично правильних програм (ДСП-методу) [2]. На відміну від традиційних синтаксичних аналізаторів, ДСП-метод орієнтований не на пошук і виправлення синтаксичних помилок, а на виключення можливості їх появи в процесі побудови алгоритму. Основна ідея методу полягає в порівневому конструюванні схем зверху вниз шляхом суперпозиції мовних конструкцій САА-М. На кожному кроці конструювання система в діалозі з користувачем надає на вибір лише ті конструкції, підстановка яких у текст алгоритму, що проектується, не порушує синтаксичну правильність схеми. На основі побудованої схеми алгоритму виконується автоматична генерація тексту програми цільовою мовою програмування. Відображення операцій САА-М у текст мовою програмування подане у вигляді шаблонів і зберігається в базі даних інструментарію.

Згаданий підхід був реалізований в системі ІПС, розглянутій у роботах [2–6]. Основними компонентами системи ІПС є такі:

- діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), призначений для діалогового проектування схем алгоритмів та синтезу програм мовами Java та C++;
- редактор граф-схем;
- база даних алгеброалгоритмічних специфікацій, у якій зберігається текст конструкцій САА і базисних елементів схем, а також їх програмні реалізації;
- генератор САА-схем за гіперсхемами.

Для автоматизації виконання трансформацій алгоритмів система ІПС застосовується спільно з системою TermWare [10], що ґрунтується на парадигмі переписувальних правил.

У роботах [7–9] розглядається нова версія системи ІПС, названа онлайновим діалоговим конструктором синтаксично правильних програм. Система ОДСП при-

значена для діалогового проектування, генерації й запуску програм. Основна відмінність інструментарію ОДСП у порівнянні з системою ІПС полягає у сервісно-орієнтованій архітектурі інструментарію та спрямованості на багатокористувальницьке використання інструментарію через Інтернет. Іншою відмінністю є орієнтація системи ОДСП на парадигму об'єктно-орієнтованого програмування, тоді як інструментарій ІПС в основному спрямований на імперативну парадигму.

У попередніх роботах [2–9] засоби САА-М та розроблені інструментальні системи використовувались для автоматизації проектування та генерації паралельних програм для багатоядерних центральних процесорів та графічних прискорювачів, зокрема, у предметній області метеорологічного прогнозування.

У наступних розділах виконаний подальший розвиток створених засобів у напрямку їх застосування для тривимірного випадку задачі прогнозування, а також автоматизації розробки програмного забезпечення для Інтернет-порталу з надання послуг прогнозування погоди.

2. Проектування паралельної схеми алгоритму для розв'язання тривимірної задачі конвективної дифузії

У даному розділі розглядається застосування апарату САА-М та системи ІПС для розробки паралельної програми розв'язання тривимірної задачі конвективної дифузії, яка виникає при математичному моделюванні циркуляції атмосфери в метеорології.

2.1. Математична модель. Постановка задачі конвективної дифузії детально викладена в [11]. У згаданій роботі розглянута модель циркуляції атмосфери, яка описується такими основними фізичними законами:

1) збереження кількості руху

$$\frac{dV}{dt} = -\frac{1}{\rho} gradp - 2\Omega \times V + F_g + \frac{1}{\rho} F_f, \quad (1)$$

2) збереження маси (нестисливість середовища)

$$divV = 0, \quad (2)$$

3) збереження теплової енергії

$$\frac{dT}{dt} = \frac{\delta}{c_V \rho}, \quad (3)$$

4) рівняння стану повітря

$$p = \rho R_C T, \quad (4)$$

де V – вектор швидкості у неінерційній системі координат, ρ – густина повітря, p – атмосферний тиск, Ω – вектор кутової швидкості обертання Землі, F_g – сила тяжіння, F_f – щільність сили тертя, T – абсолютна температура, δ – притік тепла до одиниці об'єму повітря, c_V – питома теплоємність сухого повітря за сталого об'єму, R_C – питома газова стала сухого повітря.

У моделях, що описують макромасштабну циркуляцію, використовують сферичну систему координат [26]. Рівняння (1) – (3) після деяких спрощень матимуть вигляд

$$\frac{\partial u}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial u}{\partial \lambda} + \frac{v}{a}\frac{\partial u}{\partial \varphi} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho a\cos\varphi}\frac{\partial p}{\partial \lambda} + \left(v\sin\varphi - w\cos\varphi\right)\left(2\omega + \frac{u}{a\cos\varphi}\right) + \frac{1}{\rho}F_\lambda, \quad (5)$$

$$\frac{\partial v}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial v}{\partial \lambda} + \frac{v}{a}\frac{\partial v}{\partial \varphi} + w\frac{\partial v}{\partial z} = -\frac{1}{\rho a}\frac{\partial p}{\partial \varphi} - u\left(2\omega + \frac{u}{a\cos\varphi}\right)\sin\varphi - \frac{wv}{a} + \frac{1}{\rho}F_\varphi, \quad (6)$$

$$\frac{1}{a\cos\varphi}\frac{\partial u}{\partial \lambda} + \frac{1}{a}\frac{\partial v}{\partial \varphi} + \frac{\partial w}{\partial z} - \frac{v}{a}tg\varphi = 0, \quad (7)$$

$$\frac{\partial T}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial T}{\partial \lambda} + \frac{v}{a}\frac{\partial T}{\partial \varphi} + w\frac{\partial T}{\partial z} = \frac{\delta}{c_V \rho}, \quad (8)$$

де λ – довгота, φ – широта, z – висота над рівнем моря, $V=(u,v,w)$, a – радіус Землі, ω – швидкість добового обертання Землі, $F_f=(F_\lambda,F_\varphi,F_z)$.

Паралельна чисельна реалізація цієї моделі здійснюється відповідно до трирівневого алгоритму [12], який передбачає розпаралелювання за рівняннями, просторовими напрямками та підобластями. В алгоритмі використовується модифікований адитивно-усереднений метод розщеплення (МАУМ) [13]. До розрахункової області застосовуються покоординатні декомпозиції [27]: розбиття уздовж λ для підзадач за напрямками φ та z; розбиття уздовж φ для підзадач за напрямком λ. Тому в програмі використані два формати подання даних в масивах: основний $[\lambda][\varphi][z]$ та допоміжний $[\varphi][z][\lambda]$. Із рівнянь (5) та (6) визначалися горизонтальні складові руху u та v, із (7) – вертикальна складова w, із (8) – температура T, а із (4) – густина ρ.

2.2. Схема паралельного алгоритму. У даному підрозділі наведена розроблена САА-схема паралельного алгоритму для розв'язання вищерозглянутої задачі конвективної дифузії. Схема сконструйована із використанням інструментальної системи ІПС [2–6]. На основі схеми виконана генерація програмного коду мовою C++ з використанням засобів OpenMP [14]. САА-схема складається з двох складених операторів: "Основна схема" та "Паралельні обчислення", що подані на рис. 1 та рис. 2, відповідно. Згадані складені оператори відповідають функціям *main* та *CalcParallelPart* у програмі мовою C++.

У складеному операторі "Основна схема" виконується виклик операторів ініціалізації даних (параметрів, масивів і т. п.), відліку часу виконання програми, складеного оператора виконання паралельних обчислень, порівняння обчислених значень та збереження результуючих даних у файли. У цій схемі вказані такі змінні: *Proc_Num* – кількість паралельних потоків; *Subdomains* – кількість підобластей; *M_prm* – *m*-параметр МАУМ; *TmKnotsPer12h* – кількість часових точок для 12-годинного періоду; *TmLimCalc* – кінцеве значення часу в умові *m*-циклу; *tau* – часовий крок; *U*, *V* – горизонтальні складові руху, *W* – вертикальна складова; *P* – атмосферний тиск; *T* – абсолютна температура.

СХЕМА ПАРАЛЕЛЬНИЙ АЛГОРИТМ ДЛЯ РОЗВЯЗАННЯ 3-ВИМІРНОЇ ЗАДАЧІ КОНВЕКТИВНОЇ ДИФУЗІЇ

"Основна схема"

==== "Визначити змінну (*Proc_Num*) типу (int) = (1)"
ПОТІМ
"Встановити початкові параметри (*Subdomains*, *M_prm*, *TmKnotsPer12h*, *TmLimCalc*, *tau*)"
ПОТІМ
"Почати відлік часу"
ПОТІМ
"Завантажити дані в масиви для (*P*, *W*, *U*, *V*, *T*)"
ПОТІМ
(*Proc_Num* := "Паралельні обчислення")
ПОТІМ
"Закінчити відлік часу та вивести результат"
ПОТІМ
"Порівняти інтерпольовані та обчислені значення в нормі L2"
ПОТІМ
"Зберегти дані у файли для 3-вимірної задачі конвективної дифузії"

Рис. 1. Початок САА-схеми для задачі конвективної дифузії: складений оператор "Основна схема"

```
"Паралельні обчислення"
==== "Визначити константу (sdm_sz) типу (int) = ((SZ_X1-1)/Subdomains+1)"
     ПОТІМ
     "Визначити змінну (NmbThreads) типу (int)"
     ПОТІМ
     "Визначити масив (pOut[DIR][EVL_MT_VAL][MAX_SUBDMN]) типу (double)"
     ПОТІМ
     "(NmbThreads) := (Subdomains * DIR * EVL_MT_VAL)"
     ПОТІМ
     ПАРАЛЕЛЬНО (proc = 0, ..., NmbThreads-1)
     (
       "Ініціалізація змінних та масивів"
       ПОТІМ
       "Коментарій (***** m-цикл *****)"
       ПОТІМ
       "(n) := (M_prm)"
       ПОТІМ
       ПОКИ '(n * tau) <= (TmLimCalc)'
       ЦИКЛ
            "Встановити значення для U, V, T, P, D, W, F, C та граничні умови"
            ПОТІМ
            ВИБІР ['(dir) = (0)'] →
                     "Обчислити задачі для напрямку (Lam)",
                      ['(dir) = (1)'] →
                      "Обчислити задачі для напрямку (Fi)",
                      ['(dir) = (2)'] →
                      "Обчислити задачі для напрямку (Z)")
            ПОТІМ
            "Поміняти місцями значення в масивах pR1 та pR2"
            ПОТІМ
            ЧЕКАТИ 'Обробка в усіх потоках закінчена'
            ПОТІМ
            "Обчислити середні значення на основі отриманих результатів для
             кожного напрямку"
            ПОТІМ
            ЧЕКАТИ 'Обробка в усіх потоках закінчена'
            ПОТІМ
            "Занести результати в глобальні масиви"
            ПОТІМ
            ЧЕКАТИ 'Обробка в усіх потоках закінчена'
            ПОТІМ
            "Збільшити (n) на (M_prm)"
       КІНЕЦЬ ЦИКЛУ
       ПОТІМ
       "Звільнити пам'ять для масивів для 3-вимірної задачі конвективної дифузії"
     )
     ПОТІМ
     "Повернути значення (NmbThreads)";
```

Рис. 2. Продовження САА-схеми для задачі конвективної дифузії: деталізація складеного оператора "Паралельні обчислення"

Схема "Паралельні обчислення" (див. рис. 2) починається з ініціалізації змінних і масивів, після чого виконується *m*-цикл МАУМ. У тілі *m*-циклу спочатку здійснюється підготовка до основних обчислень: постановка граничних умов, обчислення коефіцієнтів рівнянь, вертикальної складової швидкості і т. п. Далі обчислюються підзадачі для кожного з напрямків λ, φ та z. Потім виконується усереднювання результатів, які відповідають різним напрямкам, зі збереженням граничних умов. У кінці циклу отримані результати записуються в загальні масиви.

Окрім вищезазначених, у схемі паралельних обчислень вказані такі додаткові змінні та константи: *proc* – номер потоку; *NmbThreads* – кількість паралельних потоків; *sdm_sz* – розмір підобластей у напрямку λ; *SZ_X*1 – кількість точок скінченно-різницевої сітки у напрямку λ; *pOut, pR*1, *pR*2 – масиви для зберігання проміжних результатів; *DIR* = 3 кількість напрямків (λ, φ, z); *EVL_MT_VAL* = 3 – кількість метеорологічних величин в еволюційному рівнянні; *MAX_SUBDMN* – максимальна кількість підобластей; *n* – змінна *m*-циклу; *D* – густина повітря; *F* – коефіцієнт рівнянь; *C* – адвекція; *dir* – цілочисельна змінна для зберігання значення поточного напрямку (дорівнює 0 для λ; 1 для φ; 2 для z). Кількість паралельних потоків в алгоритмі встановлюється згідно формули

$$(NmbThreads) := (Subdomains * DIR * * EVL\_MT\_VAL). \quad (9)$$

Як згадувалося вище, на основі побудованої САА-схеми в системі ІПС виконана генерація програмного коду мовою C++ із використанням OpenMP [14]. OpenMP є сукупністю директив компілятора (прагм), бібліотечних процедур і змінних оточення, що призначена для програмування багатопотокових застосувань на багатопроцесорних системах із загальною пам'яттю. У процесі генерації коду ДСП-конструктор інструментарію ІПС використовує шаблони програмних реалізацій операцій САА-М та базисних понять із бази даних. Зокрема, операцію асинхронного виконання потоків (ПАРАЛЕЛЬНО) для задачі, що розглядається, було реалізовано із використанням OpenMP директиви *#pragma omp parallel*. Для реалізації операції синхронізації (ЧЕКАТИ) було застосовано директиву *#pragma omp barrier*, яка виконує затримку обчислень доти, поки всі потоки не завершать свою роботу.

Результати проведеного експерименту з виконання згенерованої програми на мультипроцесорній платформі розглядаються у розділі 4.

3. Проектування онлайнового сервісу для надання послуг метеорологічного прогнозування

У даному розділі проілюстроване застосування засобів алгебри алгоритмів та системи ОДСП на прикладі розробки онлайнового сервісу, який входить до складу сервісно-орієнтованої системи «Оракул-Портал», що призначена для надання послуг метеорологічного прогнозування [1]. Згаданий портал складається з декількох модулів-сервісів і використовує паралельні обчислення на мультипроцесорній платформі. Структура системи детально розглянута у роботі [1]. До складу порталу входять сервіси управління метеорологічними прогнозами та розрахунку прогнозів. База даних (БД) системи містить вихідні метеодані, постачальником яких є метеорологічний центр у м. Офенбах [28], а також дані уточнюючих прогнозів, розрахованих на основі вихідних даних. Значна частина системи написана мовою Java, крім власне паралельних програм для виконання обчислення прогнозу, які розроблені мовою C++ із використанням OpenMP. На даний момент використовуються три варіанти програм для обрахунку прогнозів: двовимірного, тривимірного та періодичного. Вище у розділі 2 даної роботи наведено побудовану САА-схему паралельного алгоритму для автоматизованої генерації програми для тривимірного випадку задачі прогнозування.

Далі на рис. 3 показано побудовану в системі ОДСП схему для Java класу *Prediction3DServiceImpl*, який реалізує один із сервісів системи «Оракул-Портал».

```
 Annotated class X implements Y (Prediction3DServiceImpl, Prediction3DService)
<Annotations>
    Annotation (Service)

<Fields>

    Field initialized (private static final, Logger, LOGGER)
    <Initialization>
        Call instance method (Logger, getLogger, Prediction3DServiceImpl.class);

    Field annotated (public, Prediction3DRepository, prediction3dRepository)
    <Annotations>
        Annotation (Resource)

<Methods>

    Method annotated (Prediction3D, findById, Long id)
    <Annotations>
        Annotation (Override);
        Annotation (Transactional)
    <Body>
        Call instance method (LOGGER, debug, "Trying get prediction3d #" + id +
                                                    " from DB");
        Return (Call instance method (prediction3dRepository, findById, id));

    Method annotated (void, savePrediction, Prediction3D prediction)
    <Annotations>
        Annotation (Override);
        Annotation (Transactional)
    <Body>
        Assign (prediction, Call instance method (prediction3dRepository, save,
                                                    prediction));
        Call instance method (LOGGER, debug,"Prediction3D #"
                                        + prediction.getId() + " has been saved to DB");

    Method annotated (Long, createPrediction, Prediction3D prediction)
    <Annotations>
        Annotation(Override)
    <Body>
        Assign (prediction, Call instance method (prediction3dRepository, save, prediction));
        Return (Call instance method (prediction, getId));

    Method annotated (void, deletePrediction, Long id)
    <Annotations>
        Annotation(Override)
    <Body>
        Call instance method (prediction3dRepository, delete, id);
```

Рис. 3. Схема класу Prediction3DServiceImpl

При конструюванні наведеної схеми використані об'єктно-орієнтовані операції CAA-M, розглянуті у підрозділі 1.2. Клас *Prediction3DServiceImpl* визначено за допомогою конструкції "Annotated class X implements Y", у параметрах якої вказано назву класу та назву реалізованого ним інтерфейсу *Prediction3DService*. Після заголовка класу наведено опис анотації, визначення полів та методів класу.

Полями класу є ініціалізоване поле *LOGGER* та анотоване поле *prediction3DRepository*. Поле *LOGGER* є екземпляром Java класу *Logger*, методи якого призначені для запису повідомлень у журнал. Поле *prediction3DRepository* представляє базу даних порталу.

Клас *Prediction3DServiceImpl* також містить чотири методи. Метод *findById* призначений для пошуку (за номером *id*) обрахованого прогнозу у базі даних. Методи *savePrediction*, *createPrediction* та *deletePrediction* призначені для збереження, створення та видалення прогнозів з БД відповідно.

На основі побудованої схеми в системі ОДСП виконана автоматична генерація програмного коду класу *Prediction3DServiceImpl* мовою Java.

4. Результати чисельних експериментів

Проведено два незалежних експерименти з виконання OpenMP програми для розв'язання тривимірної задачі метеорологічного прогнозування, згенерованої за побудованою у розділі 2 схемою, на двох мультипроцесорних системах. Перша система є одним з вузлів кластера Інституту програмних систем НАНУ і складається із двох чотирьох-ядерних процесорів Quad Core Intel Xeon E5405 із частотою 2 ГГц (всього 8 ядер на вузлі). Як другу мультипроцесорну систему використано один з вузлів суперкомп'ютерного обчислювального комплексу СКІТ-4 Інституту кібернетики імені В.М. Глушкова НАН України [29]. Згаданий вузол містить два шестиядерних процесори Intel Xeon X5675, із частотою 3.07 ГГц (всього 12 фізичних ядер на вузлі; 24 логічних ядра – у режимі *hyper-threading*).

У процесі експерименту були встановлені такі значення параметрів програми:

термін часу, на який виконується розрахунок прогнозу погоди – 12 годин;

m-параметр модифікованого адитивно-усередненого методу розщеплення [11–13] *M_prm* = 10;

кількість точок часової сітки *TmKnotsPer12h* = 4320;

кінцеве значення часу *TmLimCalc* = 43200 сек;

часовий крок *tau* = 10 сек;

кількість підобластей *Subdomains* = 2, кількість потоків *NmbThreads* = 18 (див. формулу (9) у підрозділі 2.2).

Просторова область визначення задачі відповідала наявним архівним даним і становила:

$$\lambda \in [0^\circ, 90^\circ],\ \varphi \in [0^\circ, 90^\circ],$$

$$z \in [8000; 18000].$$

Розмір вхідної скінченно-різницевої сітки обрано $182 \times 182 \times 11$ точок.

У таблиці подано отримані результати виконання програми на 1 та 8 ядрах процесорів кластера ІПС НАНУ та 1 та 12 фізичних ядрах кластера СКІТ-4. Як видно з таблиці, проведені експерименти демонструють гарний ступінь розпаралелюваності обчислень. Краще значення прискорення на кластері СКІТ-4 пояснюється більшою кількістю ядер та наявністю вищезгаданого режиму *hyper-threading*.

Таблиця. Результати виконання паралельної програми для розв'язання задачі конвективної дифузії на кластерах ІПС та СКІТ-4

Мультипроцесорна система	Час $T(1)$ виконання на 1 ядрі, сек	Час $T(N)$ виконання на N ядрах, сек	Мультипроцесорне прискорення, $Sp = \frac{T(1)}{T(N)}$
Кластер ІПС, кількість фізичних ядер $N = 8$	1504,80	259,43	5,80
Кластер СКІТ-4, кількість фізичних ядер $N = 12$	988,20	83,88	11,78

Висновки

Виконане автоматизоване конструювання високорівневих алгебро-алгоритмічних специфікацій програмного забезпечення для розв'язання задачі метеорологічного прогнозування. Прогнозування здійснюється на основі паралельної реалізації задач моделювання циркуляції атмосфери на мультипроцесорній платформі. Виконане проектування сервісу, що входить до складу розроблюваного Інтернет-порталу [1] з надання послуг метеопрогнозу. Виконана генерація програмного коду за побудованими специфікаціями на основі використання розроблених інструментальних засобів автоматизованого проектування та синтезу програм. Перевагою використовуваного в інструментарії підходу до проектування програм є використання мовних конструкцій, близьких до природної мови, а також застосування методу, який забезпечує синтаксичну правильність алгоритмів та програм, що проектуються.

Проведено нові експерименти з виконання розробленої паралельної програми для розв'язання задачі метеорологічного прогнозування на мультипроцесорній платформі, що показали гарний ступінь розпаралелюваності обчислень.

1. *Дорошенко А.Ю., Іваненко П.А., Овдій О.М. та інші.* До створення Інтернет-порталу надання послуг метеорологічного прогнозування на мультипроцесорній платформі // Проблеми програмування. – 2015. – № 3. – С. 24–32.
2. *Андон Ф.И., Дорошенко А.Е., Яценко Е.А. и др.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 631 с.
3. *Дорошенко Е.А., Яценко Е.А.* О синтезе программ на языке Java по алгеброалгоритмическим спецификациям // Проблеми програмування. – 2006. – № 4. – С. 58–70.
4. *Дорошенко А.Е., Жереб К.А., Яценко Е.А.* Формализованное проектирование эффективных многопоточных программ // Там само. – 2007. – № 1. – С. 17–30.
5. *Яценко Е.А.* Интеграция инструментальных средств алгебры алгоритмов и переписывания термов для разработки эффективных параллельных программ // Там само. – 2013. – № 2. – С. 62–70.
6. *Дорошенко А.Ю., Бекетов О.Г., Яценко О.А. та інші.* Розробка сервісно-орієнтованих засобів для запуску паралельних програм на мультипроцесорному кластері // Там само. – 2014. – № 4. – С. 3–14.
7. *Иовчев В.А., Мохница А.С.* Инструментальные средства алгебры алгоритмики на платформе Web 2.0 // Там само. – 2010. – № 2–3. – С. 547–555.
8. *Дорошенко А.Ю., Бекетов О.Г., Яценко О.А. та інші.* Автоматизована генерація паралельних програм для графічних прискорювачів на основі схем алгоритмів // Там само. – 2015. – № 1. – С. 19–28.
9. *Андон Ф.И., Дорошенко А.Е., Яценко Е.А. и др.* Инструментальные средства автоматизации параллельного программирования

на основе алгебры алгоритмов // Кибернетика и системный анализ. – 2015. – № 1. – С. 162–170.
10. *Doroshenko A., Shevchenko R.* A rewriting framework for rule-based programming dynamic applications // Fundamenta Informaticae. – Amsterdam: IOS Press, 2006. – Vol. 72, N 1–3. – P. 95–108.
11. *Черниш Р.І.* Паралельна реалізація моделі макромасштабної циркуляції атмосфери // Вісник Київського національного університету імені Тараса Шевченка: серія фізико-математичні науки. – 2009. – № 2. – С. 155–158.
12. *Черниш Р.І., Тирчак Ю.М., Іваненко П.А.* Побудова паралельного алгоритму чисельного розв'язання багатовимірної задачі моделювання навколишнього середовища // Проблеми програмування. – 2009. – № 1. – С. 85–91.
13. *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.
14. *OpenMP* Application Program Interface [Електронний ресурс]. – Режим доступу: http://www.openmp.org/mp-documents/ OpenMP4.0.0.pdf. – 04.11.2015 р.
15. *Sannella D., Tarlecki A.* Foundations of algebraic specification and formal software development. – Berlin: Springer-Verlag, 2012. – 594 p.
16. *Flener P.* Achievements and prospects of program synthesis // Lecture Notes in Artificial Intelligence. – Berlin: Springer-Verlag, 2002. – Vol. 2407. – P. 310–346.
17. *Gulwani S.* Dimensions in program synthesis // Proc. 12th Int. ACM SIGPLAN symposium on Principles and Practice of Declarative Programming, Hagenberg, Austria (26–28 July, 2010). – New York: ACM, 2010. – P. 13–24.
18. *Raghesh A.* A framework for automatic OpenMP code generation. A project report [Електронний ресурс]. – Режим доступу: http://www. cse.iitm.ac.in/~raghesh/raghesh-a-masters-thesis.pdf. – 04.11.2015 р.
19. *Hu K., Zhang T., Yang Z.* Multi-threaded code generation from Signal program to OpenMP // Frontiers of Computer Science. – Berlin: Springer-Verlag, 2013. – Vol. 7, N 5. – P. 617–626.
20. *PLUTO* – an automatic parallelizer and locality optimizer for multicores [Електронний ресурс]. – Режим доступу: http://pluto-compiler.sourceforge.net/ – 04.11.2015 р.
21. *Par4all* [Електронний ресурс]. – Режим доступу: – http://www.par4all.org/ – 04.11.2015 р.
22. *Kabir M.H.* Automatic construction of Java programs from functional program specifications // International Journal Of Advanced Chemical Science and Applications. – Bhubaneswar: IRD India, 2015. – Vol. 6, N 4. – P. 65–72.
23. *Schoeberl M., Brooks C., Lee E.A.* Code generation for embedded Java with Ptolemy // Proc. 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Waidhofen, Austria (13–15 October, 2010). Lecture Notes in Computer Science. – Berlin: Springer-Verlag, 2010. – Vol. 6399. – P. 155–166.
24. *Herrington J.* Code-generation techniques for Java [Електронний ресурс]. – Режим доступу: http://www.onjava.com/pub/a/onjava/ 2003/09/03/generation.html. – 04.11.2015 р.
25. *Oracle* Java documentation. The Java Tutorials. Lesson: Annotations [Електронний ресурс]. – Режим доступу: https://docs. oracle.com/javase/tutorial/java/annotations/ – 04.11.2015 р.
26. *Белов П.Н.* Практические методы численного прогноза погоды. – Ленинград: Гидрометеорологическое издательство, 1967. – 336 с.
27. *Черниш Р.І.* Покоординатна декомпозиція області для еволюційних задач математичної фізики // Вісник Київського національного університету імені Тараса Шевченка: серія фізико-математичні науки. – 2008. – № 4. – С. 191–194.
28. *Wetter* und Klima – Deutscher Wetterdienst – Startseite (веб-сайт метеорологічного центру у м. Офенбах) [Електронний ресурс]. – Режим доступу: http://www.dwd. de. – 04.11.2015 р.
29. *Суперкомп'ютер* ІК НАН України. – http://icybcluster.org.ua.

References

1. *Doroshenko A.Yu., Ivanenko P.A., Ovdii O.M.* at all. Creation of an Internet portal providing meteorological forecasting services on multi-processor platform // Problems in programming. – 2015. N 3. – P. 24–32 (in Ukrainian).
2. *Andon P.I. et al.* Algebra-algorithmic models and methods of parallel programming. Kiev: Academperiodika, 2007 (in Russian).

3. *Doroshenko, A.Yu. & Yatsenko O.A.* About the synthesis of Java programs by algebra-algorithmic specifications // Problems in programming. – 2006. – N 4. – P. 58–70 (in Russian).
4. *Doroshenko, A.Yu., Zhereb K.A & Yatsenko O.A.* Formalized design of efficient multi-threaded programs // Problems in programming. – 2007. – N 1. – P. 17–30 (in Russian).
5. *Yatsenko O.A.* (2013) Integration of algebra-algorithmic tools and term rewriting for efficient parallel programs development // Problems in programming. – 2013. – N 2. – P. 62–70 (in Russian).
6. Doroshenko A.Yu., Beketov O.G., Yatsenko O.A. at all. Development of the service-oriented soft-ware for launching parallel programs on a multiprocessor cluster // Problems in programming. – 2014. – N 4. – P. 3–14 (in Ukrainian).
7. Iovchev V.O. & Mokhnitsa O.S. Algebra-algorithmic tools on Web 2.0 platform // Problems in programming. – 2010. – N 2. – P. 547–555 (in Russian).
8. *Doroshenko A.Yu., Beketov O.G., Ivaniv R.B. at all.* Automated generation of parallel programs for graphics processing units based on algorithm schemes. Problems in programming. – 2015. – N 1. – P. 19–28 (in Ukrainian).
9. *Andon, P.I., Doroshenko, A.Yu., Beketov, O.G., Iovchev, V.O. & Yatsenko O.A.* (2015) Software tools for automation of parallel programming on the basis of algebra of algorithms // Cybernetics and systems analysis. (1). P. 162–170 (in Russian).
10. *Doroshenko A. & Shevchenko R.* (2006) A rewriting framework for rule-based programming dynamic applications. Fundamenta Informaticae. Amsterdam: IOS Press. 72 (1-3). – P. 95–108.
11. *Chernysh R.I.* (2009) Parallel implementation of atmospheric macroscale circulation model. Bulletin of the University of Kiev, Series: Physics & Mathematics. (2). – P. 155–158 (in Ukrainian).
12. *Chernysh R.I., Tyrchak Yu.M. & Ivanenko, P.A.* (2009) Construction of parallel algorithm for numeric solving multidimensional problem of environmental modeling // Problems in programming. (1). – P. 85–91 (in Ukrainian).
13. *Prusov, V.A., Doroshenko, A.Yu. & Chernysh, R.I.* (2009) A method for numerical solution of a multidimensional convection-diffusion problem // Cybernetics and systems analysis. (1). – P. 100–107 (in Russian).
14. *OPENMP*. (2015) OpenMP Application Program Interface. [Online] Available from: http://www.openmp.org/mp-documents/ OpenMP4.0.0.pdf [Accessed: 4 November 2015].
15. *Sannella D. & Tarlecki A.* (2012) Foundations of algebraic specification and formal software development. Berlin: Springer-Verlag.
16. *Flener P.* (2002) Achievements and prospects of program synthesis. Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag. 2407. – P. 310–346.
17. *Gulwani S.* (2010) Dimensions in program synthesis. In Proc. 12th Int. ACM SIGPLAN symposium on Principles and Practice of Declarative Programming. Hagenberg, Austria, 26-28 July 2010. New York: ACM. – P. 13–24.
18. *Raghesh A.* (2011) A framework for automatic OpenMP code generation. A project report. [Online]. Available from: http://www.cse.iitm.ac.in/~raghesh/raghesh-a-masters-thesis.pdf [Accessed: 4 November 2015].
19. *Hu K., Zhang T., Yang Z.* Multi-threaded code generation from Signal program to OpenMP // Frontiers of Computer Science. – Berlin: Springer-Verlag, 2013. – Vol. 7, N 5. – P. 617–626.
20. *PLUTO-COMPILER.* (2015) *PLUTO – an automatic parallelizer and locality optimizer for multicores.* [Online] Available from: http://pluto-compiler.sourceforge.net/ [Accessed: 4 November 2015].
21. *PAR4ALL.* [Online] Available from: http://www.par4all.org/ [Accessed: 4 November 2015].
22. *Kabir M.H.* Automatic construction of Java programs from functional program specifications // International Journal Of Advanced Chemical Science and Applications. – Bhubaneswar: IRD India, 2015. – Vol. 6, N 4. – P. 65–72.
23. *Schoeberl M., Brooks C., Lee E.A.* Code generation for embedded Java with Ptolemy // Proc. 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Waidhofen, Austria (13–15 October, 2010). Lecture Notes in Computer Science. – Berlin: Springer-Verlag, 2010. – Vol. 6399. – P. 155–166.
24. *Herrington J.* (2003) Code-generation techniques for Java. [Online] Available from: http://www.onjava.com/pub/a/onjava/ 2003/09/03/generation.html [Accessed: 4 November 2015].
25. *ORACLE.* (2015) Java documentation. The Java Tutorials. Lesson: Annotations. [Online]

Available from: https://docs.oracle.com/javase/tutorial/java/annotations/ [Accessed: 4 November 2015].
26. *Belov P.N.* (1967) Practical methods for numerical weather prediction. Saint Petersburg: Hydrometeorological publishing. (in Russian).
27. *Chernysh R.I.* (2008) Coordinate-wise decomposition of area for evolutionary tasks of mathematical physics. Bulletin of the University of Kiev, Series: Physics & Mathematics. (4). – P. 191–194 (in Ukrainian).
28. *DWD* (2015) Wetter und Klima – Deutscher Wetterdienst. [Online] Available from: http://www.dwd.de [Accessed: 4 November 2015].
29. *ICYBCLUSTER.* (2015) Supercomputer of IC [Online] Available from: http://icybcluster.org.ua [Accessed: 4 November 2015].

Одержано 23.11.2015

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень Інституту
програмних систем НАН України,
професор кафедри автоматики і управління в технічних системах НТУУ "КПІ".
Кількість наукових публікацій в
українських виданнях – понад 150.
Кількість наукових публікацій в
іноземних виданнях – понад 30.
Індекс Гірша – 3.
http://orcid.org /0000-0002-8435-1451,

Іваненко Павло Андрійович,
молодший науковий співробітник
Інституту програмних систем
НАН України.
Кількість наукових публікацій в
українських виданнях – понад 30.
Кількість наукових публікацій в
іноземних виданнях – 2.
http://orcid.org/0000-0001-5437-9763.

Овдій Ольга Михайлівна,
молодший науковий співробітник
Інституту програмних систем
НАН України.
Кількість наукових публікацій в
українських виданнях – 16.
Кількість наукових публікацій в
іноземних виданнях – 2.
http://orcid.org/0000-0002-8891-7002.

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник Інституту
програмних систем НАН України.
Кількість наукових публікацій в
українських виданнях – 28.
Кількість наукових публікацій в
іноземних виданнях – 12.
http://orcid.org/0000-0002-4700-6704.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.

E-mail: doroshenkoanatoliy2@gmail.com,
paiv@ukr.net,
olga.ovdiy@gmail.com,
oayat@ukr.net