УДК: 681.3

АВТОМАТИЧНО НАЛАГОДЖУВАНИЙ ПАРАЛЕЛЬНИЙ АЛГОРИТМ ЧИСЕЛЬНОГО РОЗВ'ЯЗАННЯ БАГАТОВИМІРНОЇ ЗАДАЧІ МОДЕЛЮВАННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

П.А. Іваненко, А.Ю. Дорошенко, Л. Суслова, Р.І. Черниш

Інститут програмних систем НАН України, проспект Академіка Глушкова, 40.
Київ, тел: 050 58064-13 paiv@ukr.net
Український гідрометеорологічний інститут,
Київ, проспект Науки, 37, chernysh@uhmi.org.ua
Національний технічний університет України «КПІ», Київ, проспект Перемоги, 32, корпус 18.

Розглядається процес створення авто-тюнера для задачі чисельного моделювання з використанням модифікованого адитивно-усередненого розщеплення з покоординатною декомпозицією області. Наведені результати роботи авто-тюнера на декількох різних платформах.

This article describes construction of auto-tuner for numeric solving multidimensional problem of environmental modeling parallel algorithm and provides analysis of results of auto-tuner execution on two different computing systems.

Вступ

У зв'язку з тим, що подальше нарощення тактової частоти процесорів стало технічно неможливим, перехід до багатоядерної архітектури процесорів з використанням «енергоефективних» процесорів меншої частоти став джерелом для подальшого нарощування обчислювальних потужностей. Цей перехід дав новий поштовх розвитку паралельних обчислень в цілому. Тому зараз індустрія інформаційних технологій та розробники прикладного програмного забезпечення роблять акцент на концепціях паралельних та розподілених обчислень. Однією з головних проблем при розробці програм, орієнтованих на багатоядерні середовища, було й залишається ефективність виконання на різних багатоядерних архітектурах. Незначні на перший погляд відмінності, такі як кількість одночасно виконуваних потоків, розмір кеша та час доступу до пам'яті, у сукупності дають значну різницю у швидкодії паралельних програмних додатків у різних середовищах.

Потенційним вирішенням цієї проблеми є так звані «авто-тюнери» [1]. Основною ідеєю їх підходу є те, що параметри, які впливають на швидкодію програм (такі як кількість потоків, «зернистість» декомпозиції даних, тощо), потрібно зробити налагоджуваними і їх оптимальні значення автономно знайде «авто-тюнер» у цільовому середовищі. Іноді навіть є доцільним створення декількох різних реалізацій алгоритмів, якщо вони в залежності від архітектури платформи можуть забезпечити значну різницю у швидкодії.

У даній роботі на прикладі паралельної реалізації багатовимірної задачі моделювання навколишнього середовища буде показано процес створення «авто-тюнера» й проаналізовано перспективність використання такого підходу в цілому.

Постановка задачі

Об'єктом оптимізації авто-тюнера було обрано паралельну реалізацію спрощеного алгоритму короткотермінового прогнозування стану навколишнього середовища. Спрощення полягало в тому, що розглядався двовимірний варіант задачі, й обраховувався тільки один параметр – швидкість вітру. Повне формулювання задачі, а також алгоритм розв'язку можна знайти в [2, 3]. Далі розглядається обране спрощення.

Розглядаємо задачу на ізобаричній поверхні :

$$\frac{\partial u}{\partial t}+\frac{v}{a}\frac{\partial u}{\partial \varphi}+\frac{u}{a\cos\varphi}\frac{\partial u}{\partial \lambda}=-\frac{g}{a\cos\varphi}\frac{\partial h}{\partial \lambda}+\frac{\partial}{\partial \lambda}\left(\frac{\mu}{\rho(a\cos\varphi)^2}\frac{\partial u}{\partial \lambda}\right)+\frac{\partial}{\partial \varphi}\left(\frac{\mu}{\rho a^2}\frac{\partial u}{\partial \varphi}\right)+$$

$$+v\left(\frac{u\,tg\,\varphi}{a}+2\Omega\sin\varphi\right), \qquad (1)$$

$$\frac{\partial v}{\partial t}+\frac{v}{a}\frac{\partial v}{\partial \varphi}+\frac{u}{a\cos\varphi}\frac{\partial v}{\partial \lambda}=-\frac{g}{a}\frac{\partial h}{\partial \varphi}+\frac{\partial}{\partial \lambda}\left(\frac{\mu}{\rho(a\cos\varphi)^2}\frac{\partial v}{\partial \lambda}\right)+\frac{\partial}{\partial \varphi}\left(\frac{\mu}{\rho a^2}\frac{\partial v}{\partial \varphi}\right)$$

$$-u\left(\frac{u\,tg\,\varphi}{a}+2\Omega\sin\varphi\right), \qquad (2)$$

де: λ, φ – довгота та широта, рад; u, v – горизонтальні складові швидкості вітру, м/с; h – висота геопотенціалу, м; ρ – густина повітря, кг/куб. м; $\Omega = 7.292 \cdot 10^{-5}$ – кутова швидкість обертання Землі, рад/с; $a = 6.373 \cdot 10^6$ м – усереднений радіус Землі; $g = 9.81 м/с^2$ – прискорення вільного падіння.

Розщеплена за напрямками задача має вигляд

$$\frac{1}{2}\frac{\partial u_1}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial u_1}{\partial \lambda} = \frac{\partial}{\partial \lambda}\left(\frac{\mu_h}{\rho(a\cos\varphi)^2}\frac{\partial u_1}{\partial \lambda}\right) - \frac{g}{a\cos\varphi}\frac{\partial h}{\partial \lambda}, \tag{3}$$

$$\frac{1}{2}\frac{\partial u_2}{\partial t} + \frac{v}{a}\frac{\partial u_2}{\partial \varphi} = \frac{\partial}{\partial \varphi}\left(\frac{\mu_h}{\rho a^2}\frac{\partial u_2}{\partial \varphi}\right) + v\left(\frac{u\,tg\,\varphi}{a} + 2\Omega\sin\varphi\right), \tag{4}$$

$$\frac{1}{2}\frac{\partial v_1}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial v_1}{\partial \lambda} = \frac{\partial}{\partial \lambda}\left(\frac{\mu_h}{\rho(a\cos\varphi)^2}\frac{\partial v_1}{\partial \lambda}\right) - u\left(\frac{u\,tg\,\varphi}{a} + 2\Omega\sin\varphi\right) \tag{5}$$

$$\frac{1}{2}\frac{\partial v_2}{\partial t} + \frac{v}{a}\frac{\partial v_2}{\partial \varphi} = \frac{\partial}{\partial \varphi}\left(\frac{\mu_h}{\rho a^2}\frac{\partial v_2}{\partial \varphi}\right) - \frac{g}{a}\frac{\partial h}{\partial \varphi}, \tag{6}$$

де

$$\mu_h = s\left(k_0 + 0.08a^2\left(\Delta\lambda^2\cos^2\varphi + \Delta\varphi^2\right)\sqrt{D_c^2 + D_d^2}\right),$$

$$D_c = \frac{\partial u}{\partial(a\lambda\cos\varphi)} - \frac{\partial v}{\partial(a\varphi)} - \frac{v\,tg\,\varphi}{a},$$

$$D_d = \frac{\partial v}{\partial(a\lambda\cos\varphi)} + \frac{\partial u}{\partial(a\varphi)} + \frac{u\,tg\,\varphi}{a},$$

$k_0 = 5 \cdot 10^4 м^2/с$ – стала величина; S – параметр варіації рівня дисипації в моделі, (3,…, 4),

Метод розщеплення

Використовувалася модифікована схема розщеплення [4, 5]. Нехай обрано часовий крок для розв'язання задачі $\tau = T/N$, де T – загальний час визначення задачі, N – загальна кількість кроків. Вважатимемо, що число N не є простим: $N = mQ$. Тоді розв'язання задачі зводиться до послідовності Q штук m-циклів, що мають таку структуру:

1. Ставимо початкові умови для (3), (5) та (4), (6): $u_1^{qm+0} = u_2^{qm+0} = u^{qm}$, $v_1^{qm+0} = v_2^{qm+0} = v^{qm}$. Також задаємо крайові умови та коефіцієнти для моменту часу $t = qm\tau$ (момент початку поточного m-циклу) і вважаємо їх сталими протягом наступних m кроків (таке припущення є справедливим в силу повільності процесів щодо часового кроку $m\tau$).

2. Розв'язуємо задачі (3), (5) та (4), (6) на проміжку $[qm\tau; (q+1)m\tau]$ із кроком τ (перший етап m-циклу), тобто

$$u_k^{qm+0} \to u_k^{qm+1} \to \ldots \to u_k^{qm+m-1} \to u_k^{qm+m}, \tag{7}$$

$$v_k^{qm+0} \to v_k^{qm+1} \to \ldots \to v_k^{qm+m-1} \to v_k^{qm+m}. \tag{8}$$

3. Усереднюємо отримані результати для кожної функції (другий етап):

$$u^{(q+1)m} = \frac{u_1^{qm+m} + u_2^{qm+m}}{2}, \qquad v^{(q+1)m} = \frac{v_1^{qm+m} + v_2^{qm+m}}{2}. \tag{9}$$

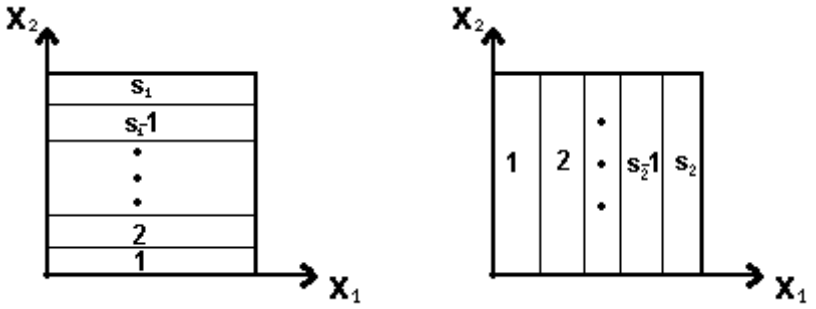
Цикл завершено.

Слід зауважити, що збільшення значення параметра m призводить, з одного боку, до зменшення часу розв'язання задачі, а з іншого – до втрати точності розв'язку. Рекомендований діапазон вибору m є 1, ..., 10.

Геометрична декомпозиція області

Окрім розпаралелення задачі за рівняннями та координатними напрямками існує можливість розпаралелення за підобластями (геометричне розпаралелення).

Було використано декомпозицію на рис. 1, а для рівнянь (3) та (5), а декомпозицію на рис. 1, б – для рівнянь (4) та (6). Такий підхід [6] дозволив уникнути проблеми постановки крайових умов для підобластей всередині області визначення задачі (де немає крайових умов) та здійснити незалежний розрахунок в межах першого та другого етапів. Цілком природно вибрати $s = s_1 = s_2$, тоді кількість задіяних процесорів становитиме $4s$.



а б

Рис. 1. Декомпозиція, що залежить від координатного напрямку рівняння $\left(x_1 = \lambda,\ \ x_2 = \varphi\right)$.

Схематично для одного рівняння розпаралелювання буде виглядати наступним чином (рис. 2).

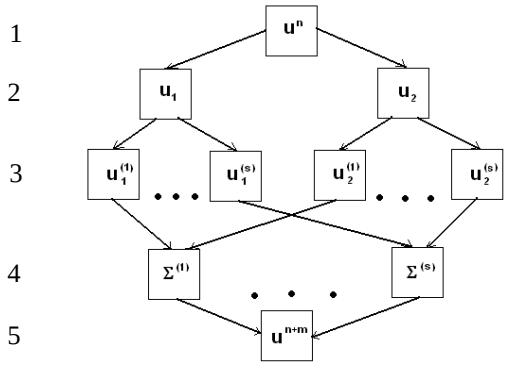


Рис. 2. Структура розпаралелення розрахунків у m-циклі

Алгоритм (згідно з рис. 2).

1. Маємо відомі значення u^n та v^n, де $n = qm$, $q = 1,\ldots,Q$ – номер поточного m-циклу. Обчислюємо значення усіх коефіцієнтів рівнянь (3), (5) та (4), (6).

2. Ставимо крайові умови для моменту часу $n = (q+1)m$ та початкові умови. Всі обчислені умови та коефіцієнти на кроках 1, 2 вважаємо сталими впродовж усього m-циклу.

3. (m-цикл – перший етап). У кожній із s підобластей здійснюємо m послідовних переходів (обчислень) за ланцюгами (7) та (8) – тут результат попереднього переходу є початковою умовою для наступного.

4. (m-цикл – другий етап). У кожній із S підобластей результат останнього m-го переходу $u_k^{(r),n+m}$ та $v_k^{(r),n+m}$ (де $r = 1,\ldots,s$ – номер підобласті, $k = 1,2$) усереднюємо за формулою середнього арифметичного (9) для u та v. У результаті отримуємо значення $u^{(r),n+m}$ та $v^{(r),n+m}$.

5. (Крок не обов'язковий). Поєднуємо результат кроку 4 в одну просторову область, одержуємо u^{n+m} та v^{n+m}. Перехід на крок 1 при $n = (q+1)m$.

Автоматичне налагодження паралельних програм

Дуже часто оптимізацію паралельної програми виконують на етапі розробки. Такий підхід майже ніколи не дозволяє створити ефективну для будь-якої обчислювальної платформи реалізацію, оскільки дуже часто оптимізація для окремої обчислювальної машини призводить до сповільнення програми на інших машинах. Також слід зважати на час, який затрачується на оптимізацію в «ручному режимі».

Досить тривалий час популярною альтернативою було використання паралельних компіляторів. Цей підхід є достатньо ефективним і дає добрі результати, проте подальший розвиток таких компіляторів затрудняється великою різноманітністю архітектур паралельних середовищ. Розширення компіляторів оптимізаційними стратегіями під кожну окрему платформу становиться майже нездійсненною задачею, тому нещодавно був запропонований альтернативний підхід [1] – використання авто-тюнерів.

Авто-тюнери є перспективним систематичним підходом, за яким паралельні програми створюються максимально універсальними й портативними, при цьому їх швидкодія залишається на рівні оптимізованих «вручну» програм. Авто-тюнер – бібліотека чи незалежний програмний додаток, який динамічно виконує параметризовану програму декілька разів і систематично досліджує область пошуку параметрів. Він намагається на цільовій платформі підібрати найбільш оптимальну конфігурацію параметрів. Підхід, який пропонують авто-тюнери, є ефективним, коли швидкодія паралельної програми суттєво залежить від декількох параметрів та є гарною альтернативою «ручному» перебору цих параметрів, особливо, коли оптимальна конфігурація не є інтуїтивно-очевидною.

У загальному випадку роботу авто-тюнера відображає діаграма на рис. 3.

Рис. 3. Цикл роботи авто-тюнера

У процесі підбору параметрів у найпростішому випадку виконується повний перебір значень параметрів з допустимого діапазону, а також перевіряються всі доступні варіації алгоритмів. Параметри й алгоритми, з використанням яких були отримані найкращі результати швидкодії, зберігаються й використовуються надалі як оптимальні. Така методика підпадає в категорію «авто-тюнери як бібліотеки» [7]. Проте повний перебір параметрів займає багато часу, і в багатьох випадках простір пошуку можна мінімізувати з використанням евристик [1, 7, 8].

При проектуванні й реалізації будь-якого паралельного алгоритма доцільним є використання так званих «патернів» паралельного програмування [9] – певних параметричних шаблонів проектування, що дозволяють використовувати вдалі рішення типових задач у специфічному контексті. Застосування патернів у більшості випадків дозволяє зекономити час на створенні власного рішення. Далі будуть присутні посилання на назви використаних патернів, інформацію про які можна знайти у [9].

Для обраної задачі, як вже було вказано, найоптимальнішим є одночасне використання операторного й геометричного розщеплення (надалі називатимемо цей алгоритм скорочено A1), за якого для обчислень необхідно 4*S процесорів (S – кількість підобластей у геометричній декомпозиції області обчислень).

Для реалізації геометричної декомпозиції області доцільно використати класичний структурний патерн Geometric decomposition pattern [9]. Цей патерн створений для вирішення проблеми організації роботи алгоритмів з вхідними даними, які було розбито на підобласті для паралельної обробки. Він дуже добре підходить для A1, схема якого для одного рівняння вже була проілюстрована на рис. 2, оскільки вхідні дані зберігаються у вигляді масивів й використання модифікованого адитивно-усередненого розщеплення дозволило позбутися будь-яких залежностей по даним між підобластями. Тобто в межах одного m-циклу не потрібно організовувати жодних операцій обміну.

Одним з основних аспектів цього патерну є підбір «зернистості» розбиття, а саме має бути присутньою можливість зміни розмірності підобластей на етапі виконання для емпіричного пошуку оптимального варіанта на цільовій обчислюваній системі. Ця задача була доручена авто-тюнеру.

Проте A1 не може претендувати на загальність, оскільки кількість одночасно незалежно виконуваних процесів у системі може бути не кратною 4. Тому доцільним є створення варіації алгоритму, яка б не мала такого обмеження. Для цього зручно використати патерн The Shared Queue pattern [9], який вирішує проблему паралельної обробки спільної черги (структура даних) й добре працює с патерном Master/Worker [9]. У цій реалізації декомпозицією області, а також обчисленням крайових умов буде займатися один головний процес (Master). Він буде заповнювати чергу готовими для обчислень підобластями, після чого усі процеси (Workers) будуть незалежно оброблювати цю чергу (надалі називатимемо цей алгоритм скорочено A2).

Як і для A1, задачею авто-тюнера буде підбір розмірності розбиття таким чином, щоб кількість підобластей була більшою за кількість обчислювальних процесів, але достатньо великої розмірності, щоб витрати на синхронізацію окупилися.

Чисельний експеримент

Паралельна реалізація задачі була створена засобами OpenMP. Експериментальні значення параметрів:

1. Величина часового кроку $\Delta t = 0.5$ с.

2. Значення m-параметра для методу розщеплення було взято рівним 10, оскільки при більших значеннях величина похибки обчислень виходить за межі 5 %.

3. Обчислення проводилися на сітці розмірністю 240*240.

Експеримент було проведено на двох ЕОМ:

1. Гомогенна ЕОМ з двома Quad Core Intel Xeon E5405 із частотою 2 ГГц та архітектурою Intel EM64T. Загальний обсяг оперативної пам'яті становить 16 ГБайт. Доступ до спільної пам'яті однорідний.

2. Гомогенна ЕОМ з 24 процесорами Dual Core Intel Itanium 2 Series 9000 із частотою 1.6 ГГц та архітектурою IA64. Система доступу до пам'яті NUMAlink 4, топологія доступу fat-tree, загальний обсяг оперативної пам'яті становить 96 ГБайт.

Мультипроцесорне прискорення

$$W(k) = T_1 / T_k,$$

де T1 й Tk – час обчислень на одному і k процесорах відповідно.

Коефіцієнт ефективності паралельного алгоритму

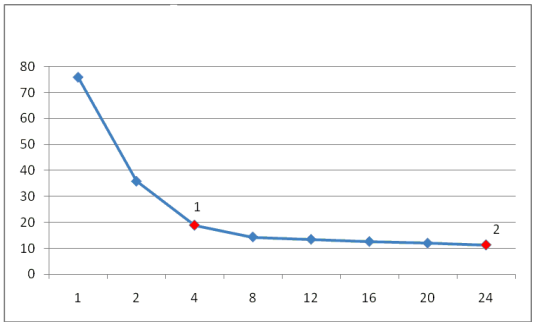
$$E(k) = \frac{W(k)}{N_{проц}}.$$

На першій обчислювальній системі авто-тюнер знайшов наступні оптимальні параметри:

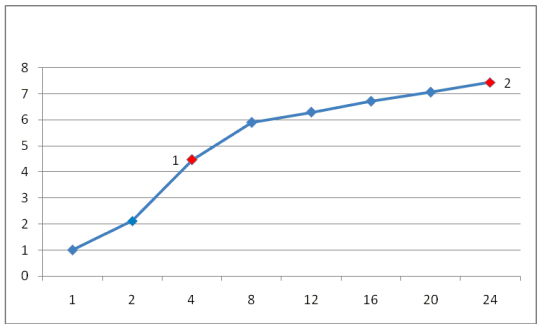
Алгоритм, №	Кількість задіяних процесорів	Кількість підобластей обчислення	Прискорення	Ефективність, %
1	8	2	5,45	68
2	8	8	4,87	61

Тобто, як видно з результатів, обидва алгоритми показали найкращі результати прискорення й ефективності при використанні усіх наявних процесорів. Часові затрати вийшли приблизно рівними: А1 виконав обчислення на 4 % швидше. Слід зауважити, що при цих же параметрах часові затрати на обчислення були мінімальними, чого не трапилося на другій ЕОМ.

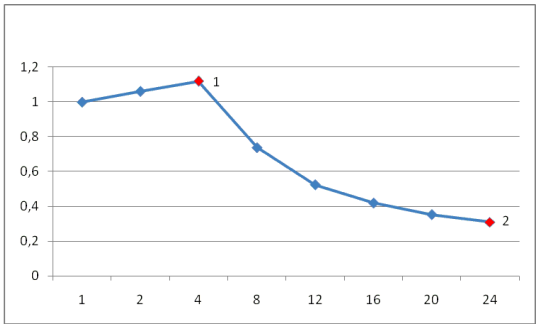
Розглянемо детальніше результати роботи авто-тюнера на другій ЕОМ (рис. 4, А1).



а



б



в

Рис. 4

Найменший час обчислень було отримано при використанні максимальної кількості процесорів (на рисунках точка № 2) (для тестування було обмежено 24) і, відповідно геометричної декомпозиції на 6 підобластей. Проте з точки зору ефективності оптимальний параметр авто-тюнер визначив інший – обчислення на 4 процесорах (на рисунках точка № 1). Значення коефіцієнта ефективності більше 1 (> 100 %) у цій точці пояснюється тим, що алгоритм А1, як було зазначено раніше, не є ефективним при використанні кількості процесорів, не кратної 4. Падіння ефективності при використанні 24 процесорів пояснюється тим, що при зменшенні «зернистості» підобластей обчислення затрати на породження та синхронізацію потоків значно зменшили виграш від паралелізму. Схожі результати було отримано для А2. Нагадаємо, що для нього проводився пошук оптимальної розмірності обчислення за фіксованої кількості процесорів.

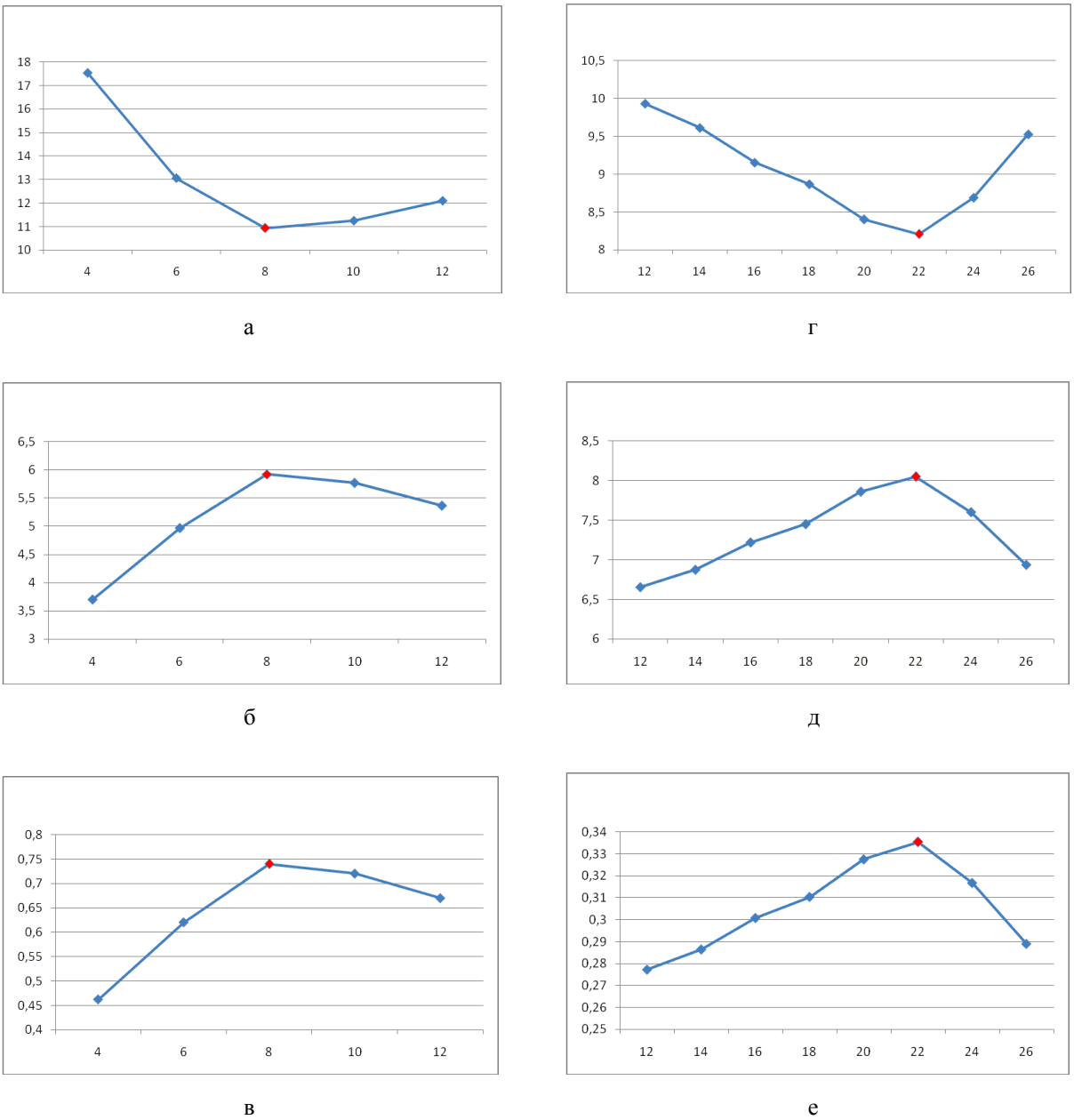


Рис. 5

На рис. 5, а, б, в показані результати обчислень на 8 процесорах. У цій серії тестів була досягнута максимальна ефективність (78 %) А2 у точці, що відповідає геометричній декомпозиції на 8 підобластей. На рис. 5, г, д, е показані результати обчислень на 24. У точці, що відповідає геометричній декомпозиції на 22 підобластях, алгоритм показав найбільшу швидкодію. У цілому для другої ЕОМ А2 виконувався швидше (на 24 %), ніж А1.

З результатів видно, що для отримання вищої ефективності на більшій кількості процесорів потрібно проводити обчислення на області більшої розмірності.

Висновки

Було розглянуто ключові моменти процесу створення авто-тюнера для прикладної задачі. На прикладі реалізації паралельного алгоритму для обраної задачі було використано поєднання алгоритмічного розщеплення задачі за просторовими напрямками із покоординатною декомпозицією області. Це дозволило отримати сукупність підзадач, які не потребують жодних узгоджень розв'язків через обміни даними на межах підобластей, що значно підвищило ефективність обчислень.

Наведено результати роботи авто-тюнера на двох різних ЕОМ. Той факт, що для них оптимальними виявилися різні алгоритми розв'язання задачі, додатково ілюструє доцільність створення авто-тюнерів. Їх використання для програм, які плануються виконувати на ЕОМ з різними архітектурами, дозволяє значно зекономити час на етапі оптимізації.

1. *K. Asanovic et al*. The Landscape of Parallel Computing Research: A View From Berkeley. Technical Report, University of California, Berkeley, 2006.
2. *Белов П.Н.* Практические методы численного прогноза погоды / Белов П.Н. – Л.: Гидрометеорологическое изд-во, 1967. – 336 с. – (Математическое моделирование в метеорологии).
3. *Белов П. Н.* Численные методы прогноза погоды / Белов П. Н., Борисенков Е. П., Панин Б. Д. – Л. : Гидрометеоиздат, 1989. – 376 с.
4. *Прусов В.А., Дорошенко А.Е., Черныш Р.И.* Выбор параметра модифицированного аддитивно-усредненного метода // Кибернетика и системный анализ. – 2009. – № 4. – С. 98–105.
5. *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.
6. *Черниш Р. І.* Покоординатна декомпозиція області для еволюційних задач математичної фізики / Р .І. Черниш. // Вісник Київ. нац. ун-ту імені Тараса Шевченка: Сер. Фізико-математичні науки. – 2008. – № 4. – С. 191–194.
7. *Thomas Karcher, Christoph Schaefer, Victor Pankratius.* Auto-Tuning Support for Manycore Applications - Perspectives for Operating Systems and Compilers, Technical Report, University of Karlsruhe, Karlsruhe, Germany, 2009.
8. *C. A. Schaefer, V. Pankratius, and W. F. Tichy.* Atune-IL: An instrumentation language for auto-tuning parallel applications. Technical Report, University of Karlsruhe, 2009.
9. *T. Mattson, B. Sanders, B.Massingill.* Patterns for Parallel Programming. Addison-Wesley Professional, Reading, MA, 2004.