УДК 681.322

Шквар Є.О.

ІНТЕГРОВАНА ГІБРИДНА ТЕХНОЛОГІЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Постановка проблеми. Головною рисою сучасного стану розвитку обчислювальних систем і їх компонентів є чітко визначена орієнтація на розвиток і застосування паралельних обчислювальних технологій. Саме такий підхід є максимально ефективним шляхом масштабування продуктивності обчислень у порівнянні з ще донедавна застосовуваним нарощуванням частоти роботи елементної бази і, зокрема, процесорів. Існує широке коло задач, пов'язаних з виконанням ресурсномістких обчислень при моделюванні різноманітних процесів, що потребує, зокрема, обробки та візуалізації значних обсягів інформації. Останнє часто ускладнюється необхідністю здійснення обробки інформації в масштабі реального часу, що потребує як значних комп'ютерних потужностей, так і відповідних алгоритмів, що спроможні їх ефективно утилізувати. Одним із активних споживачів надмірних комп'ютерних ресурсів є обчислювальна гідроаеродинаміка – одна з наймолодших галузей сучасної науки, яка протягом останнього півстоліття переживає інтенсивний розвиток, обумовлений різким зростанням потужності та доступності обчислювальної техніки та значним зменшенням собівартості обчислень. Нові обчислювальні можливості дозволяють формулювати нові постановки відомих актуальних задач, оскільки, як добре відомо, надмірна складність процесів, що протікають при русі рідини або газу суттєво обмежують можливості отримання аналітичних розв'язків лише досить простими розрахунковими випадками [1-3]. Саме тому підвищення досконалості математичних моделей разом із зростанням ефективності відповідних розрахункових методів залишається однією з найактуальніших задач у цій сфері досліджень. Сучасні задачі моделювання обтікання ґрунтуються на застосуванні чисельних методів інтегрування системи рівнянь Нав'є-Стокса або її спрощених варіантів. Нелінійність цієї системи та еліптичність за просторовими координатами рівнянь руху обумовлює необхідність використання ітераційних методів її розв'язування. Турбулентність, процеси хімічної та міжфазної взаємодії значно ускладнюють моделі течій, що вивчаються, і суттєво уповільнюють відповідні розрахункові методи. Тому шляхи розвитку обчислювальних методів і технологій в галузі гідроаеромеханіки безпосередньою мірою обумовлені потужністю та швидкодією наявної обчислювальної техніки і, перш за все визнаються задіяними методами нарощування продуктивності останньої. Спроба якомога кращої адаптації деяких існуючих класичних методів числових розрахунків до розгалужених архітектур сучасних кластерних комп'ютерних систем з багатопроцесорними чи багатоядерними вузлами і складає загальну постановку проблеми даного дослідження.

Аналіз відомих здобутків у галузі побудови паралельних алгоритмів. Оскільки розвиток мікроелектронної елементної бази не встигає за практичними потребами, суттєвою можливістю підвищення продуктивності комп'ютерів є використання паралельних архітектур шляхом використання в одному комп'ютері не одного, а двох чи більшої кількості арифметичних пристроїв – процесорів. Відповідна архітектура отримала назву симетричної мультипроцесорності (SMP-Symmetric MultiProcessing). Сучасні технології розвитку мініатюризації мікрочіпів пропонують ще один інноваційний підхід – багатоядерні процесори, що містять в одному чипі кілька ідентичних арифметич-

них пристроїв – процесорних ядер (CMP – Chip-Level MultiProcessing). Сучасні обчислювальні пакети, орієнтовані на розв'язування задач моделювання течій, наприклад PHOENICS чи ANSYS, як правило, передбачають можливість їх застосування на багатопроцесорних або багатоядерних комп'ютерах. Але досягнення бажаного зростання швидкості проведення розрахунків шляхом орієнтації на обчислювальну техніку паралельної архітектури зіштовхується з рядом проблем. Традиційно, ефективність паралелізму нарощуванням кількості арифметичних пристроїв є обмеженою пропускною здатністю спільної оперативної пам'яті, що мусить, наскільки можливо, одночасно обробляти їхні паралельні запити. Тому найрозповсюдженіші конструктивні рішення містять не більш ніж 4 процесори, а сучасні процесори масового використання – не більш ніж 4 ядра.

Створення програм під такі архітектури ґрунтується на багатопоточній технології, яка є порівняльно нескладною і незатратною в процесі її виконання. Вона підтримується у тому чи іншому вигляді усіма сучасними компіляторами з найуживаніших мов програмування. Отже конструктивний фактор виступає як найсуттєвіший з тих, що гальмують можливості масштабування продуктивності обчислень на основі такого підходу. Водночас слід зауважити, що саме ця технологія масових багатопоточних обчислень у дещо адаптованому варіанті стала стандартом при обробці та побудові графічних зображень відеопроцесорами графічних карт і сьогодні інтенсивно поширюється фірмами-розробниками відеочипів (Nvidia, AMD (ATI)) на різноманітні прикладні обчислення не лише графічної сфери. Графічні ресурсномісткі обчислення сьогодні є вкрай прогресивними враховуючи сукупну обчислювальну продуктивність відеочипів, яка набагато перевищує можливості сучасних процесорів, але, враховуючи на досить усталену специфіку і обмежену кількість команд відеопроцесорів, відмінність способу їх взаємодії з відеопам'яттю і обмеженість останньої треба при програмуванні прикладних задач разом із потужностями графічної карти застосовувати також і ресурси головного процесора та оперативної пам'яті. Все це разом із потребою створення ефективного паралельного коду значно ускладнює прикладне програмування необхідністю досягнення ще й вдалого перерозподілу обчислювальної роботи між різними компонентами комп'ютера.

Інший підхід полягає в об'єднанні деякої кількості комп'ютерів швидкодіючим мережним інтерфейсом і застосування на цій розгалуженій системі однієї з існуючих технологій розподілених обчислень. Одним з прообразів таких систем був проект Beowulf, започаткований в 1993 р. Ця архітектура отримала назву мультикомп'ютерної або кластерної (MPP – Massively Parallel Processing) і набула популярності через порівняльну дешевизну її стандартних компонентів та гарну спроможність до нарощування ресурсів, оскільки збільшення кількості арифметичних пристроїв супроводжується пропорційним зростанням розподіленої оперативної пам'яті. Сьогодні це один із популярних шляхів масштабування продуктивності обчислень, через що близько 60 % найпотужніших суперкомп'ютерів світу втілюють саме кластерний принцип побудови. Найсуттєвіший недолік тут випливає саме з необхідності адресації масиву розподіленої пам'яті і потреби забезпечити при розв'язанні задачі ефективний обмін проміжними даними між вузлами кластерної системи. У зв'язку з цим до мережного інтерфейсу висуваються вимоги не лише високої швидкодії, а й мінімізації затримок (низької латентності) при передачі даних між вузлами, що часто обумовлює його досить складну розгалужену між вузлами топологію. Різниця в принципі побудови двох розглянутих вище архітектур вимагає різних підходів до алгоритмізації та програмування задачі. Сучасні

кластерні системи є комбінаціями цих архітектур, оскільки кожен з вузлів, як правило, містить кілька багатоядерних процесорів, що робить кластерну установку компактнішою та зменшує навантаження на мережу через зниження сукупних обсягів потоків обміну даними між вузлами. Програмування ж задач традиційно здійснюється за принципами, прийнятими для розподілених систем, згідно яких кілька процесорів одного вузла чи навіть кілька ядер одного процесора обробляють незалежні процеси, кожен з яких опрацьовує відповідну ділянку виділеної йому спільної у даному випадку оперативної пам'яті як ізольовану від інших. Обмін між цими суміжними ділянками здійснюється через мережний інтерфейс, на що непродуктивно використовуються ресурси процесорів, які разом з неминучими затримками у результаті цієї передачі призводять до падіння продуктивності обчислювальної системи у цілому.

У цьому зв'язку набуває актуальності проведення досліджень з метою узгодження існуючих принципів багатопоточного та розподіленого програмування та їх адаптації до комбінованої архітектури сучасних обчислювальних кластерів.

Метою даної роботи є оцінювання ефективності застосування комбінованого методу розпаралелювання обчислень, що втілює елементи програмування, орієнтовані на архітектури як MPP, так і SMP (CMP) обчислювальних систем для двох класичних задач обчислювальної математики та гідроаеромеханіки, а саме: а) розв'язування систем лінійних алгебраїчних рівнянь (СЛАР) великих порядків, і б) моделювання течії в'язкої нестисливої рідини, що формується в замкненій порожнині у формі прямокутного паралелепіпеда. Варто зауважити, що значною мірою ці обидві задачі пов'язані між собою, оскільки при використанні числових методів, які ґрунтуються на скінченно-різницевій апроксимації диференціальних рівнянь, задача розв'язування останніх замінюється задачею знаходження значень шуканих функцій у вузлових точках різницевої сітки шляхом відшукання розв'язку СЛАР. Відмінність полягає у тому, що в останньому випадку при побудові методу розрахунку можна, як правило, ефективно використати особливості структури отриманих у результаті скінченно-різницевої дискретизації матриць СЛАР і застосовувати відповідні економічні числові методи.

Методика дослідження полягає в плануванні та здійсненні числового експерименту з порівняльної оцінки ефективності багатопоточного (SMP чи CMP орієнтованого), розподіленого (MPP орієнтованого) та комбінованого алгоритмів при змінній завантаженості наявних обчислювальних ресурсів і використанні кількох конфігурацій мережевого інтерфейсу з різною пропускною здатністю.

Стисла характеристика використаної кластерної установки. У даному дослідженні використано кластерну систему, побудовану з вісьмох вузлів, об'єднаних мережею топології "зірка", яку складали гігабітні PCI мережні карти виробництва Intel PRO/1000MT та комутатор D-Link DGS-1008D (Gigabit Ethernet) або D-Link DES-1008D(Fast Ethernet)). Кожен з них містив двоядерний процесор AMD Athlon 64 X2 3800+ з тактовою частотою 2.0 ГГц та 2 ГБ оперативної пам'яті DDR-400, яка працювала у двоканальному режимі. Кластер працював під керуванням операційної системи Windows XP x64, а міжвузловий інтерфейс забезпечувався пакетом MPICH2 (ver. 1.1b1).

Задача розв'язування СЛАР великого порядку. Задача розв'язування СЛАР великого порядку. Розглянемо неоднорідну СЛАР, задану N рівняннями виду:

$$\sum_{j=1}^{N} a_{ij} x_j = b_i \,, \tag{1}$$

де $i = \{1,..,N\}$ – номер рівняння, $j = \{1,..,N\}$ – номер невідомої; a_{ij}, b_j – коефіцієнти та вільні члени СЛАР; x_j – невідомі.

Як відомо, для систем великого порядку ($N > 1000$) внаслідок помилок округлення прямі методи перестають бути ефективними і є доцільність у використанні методів ітераційного розв'язання СЛАР. Згідно одному з них – методу Гауса-Зейделя, який одночасно втілює простоту і ефективність розрахункового алгоритму, невідомі відшукуються за правилом

$$x_j^k = \begin{cases} \dfrac{1}{a_{jj}}\left(b_j - \sum\limits_{m=j+1}^{N} a_{jm} x_m^{k-1}\right) & при \quad j = 1; \\ \dfrac{1}{a_{jj}}\left(b_j - \sum\limits_{m=1}^{j-1} a_{jm} x_m^{k} - \sum\limits_{m=j+1}^{N} a_{jm} x_m^{k-1}\right) & при \quad 1 < j < N; \\ \dfrac{1}{a_{jj}}\left(b_j - \sum\limits_{m=1}^{j-1} a_{jm} x_m^{k}\right) & при \quad j = N, \end{cases} \qquad (2)$$

що застосовується послідовно до кожного з рівнянь $j = \{1,..,N\}$ і нескладно алгоритмізується при створенні послідовного коду. Верхнім індексом k тут позначено номер поточної ітерації. При розробці паралельного алгоритму розіб'ємо СЛАР на рівні частини згідно кількості доступних вузлів (процесорів чи ядер) R. Число рівнянь у кожній з них дорівнюватиме $S = \text{int}(N / R)$. При нецілому діленні остання частина рівнянь СЛАР коригується таким чином, щоб включити останнє рівняння системи з номером N.

Багатопоточна реалізація розпаралелювання ітераційного розв'язання СЛАР. Алгоритмізація і програмування паралельного методу обрахунку на комп'ютері симетричної мультипроцесорної (SMP) архітектури (тобто при наявності кількох процесорів чи процесорних ядер, які опрацьовують розміщені в спільній оперативній пам'яті дані) спираються на технологію генерації потрібної кількості потоків, кожен з яких обробляє відповідну частину рівнянь СЛАР. Як правило, кількість потоків повинна відповідати числу наявних (або доступних) для використання процесорних елементів чи ядер. Ніяких додаткових витрат окрім виділення програмою R потоків тут не передбачається. Самі ж ці витрати є вкрай незначними у порівнянні з часом виконання цими потоками обчислювальної роботи по обробці рівнянь $i = \{L,...,L+S\}$ (L – номер потоку) за формулою (2) за умови, що порядок СЛАР не є досить малим. Таким чином, при алгоритмізації і програмуванні задачі розв'язання СЛАР за методом Гауса-Зейделя з розрахунком на багатопоточну технологію можна очікувати на близьку до ідеальної її реалізацію з точки зору ефективності розпаралелювання. При існуванні можливості задіяти технологію HyperThreading, що розробляється і удосконалюється компанією Intel, можна очікувати ще більш ефективної додаткової утилізації наявних процесорних ресурсів. Проблема масштабування цієї технології полягає лише в обмеженій кількості арифметичних пристроїв (процесорів чи їх ядер), що можуть при такому підході бути одночасно задіяні, а основним джерелом гальмування її ефективності тут виступатиме обмежена пропускна спроможність каналу процесори (ядра) – оперативна пам'ять.

Реалізація паралельного ітераційного розв'язання СЛАР на МРР комп'ютерах розподіленої архітектури (кластерах). Розглянемо більш ретельно випадок алгоритмізації цього ж методу при застосуванні кластерної обчислювальної системи з розгалуженими процесорами і оперативною пам'яттю. Припустимо аналогічно попередньому випадку, що кількість вузлів обчислювального кластера також дорівнює R. У цьому випадку кожна з R частин СЛАР обробляється окремо обчислювальним процесом, що генерується на відповідному вузлі кластера. У цьому випадку вектор поточних значень невідомих x_j^{k-1}, $j = \{1,...,N\}$ з попередньої ітерації на ітерації k мусить паралельно бути доступним для усіх процесів (вузлів), але уточнюватися він може кожним із процесів шляхом використання вже здобутої на поточній ітерації k інформації лише частково – виключно в межах відповідної частини рівнянь, що опрацьовується кількома потоками конкретного процесу за умови багатопроцесорних чи багатоядерних вузлів кластера, тобто окремо для кожного набору $j = \{M,...,M+S\}$ (M – номер процесу). Таким чином, реалізувати у повному обсязі перевагу методу Гауса-Зейделя перед методом Якобі тут вже є можливим лише частково, тобто лише в межах кожного з вузлів. Метод Якобі, як відомо, не використовує на поточній ітерації k вже уточнених значень невідомих x_j^k, тобто базується на наступній формулі

$$x_j^k = \frac{1}{a_{jj}}\left(b_j - \sum_{m=1}^{j-1} a_{jm}x_m^{k-1} - \sum_{m=j+1}^{N} a_{jm}x_m^{k-1}\right), \qquad (3)$$

а спроба його адаптації під розгалужену кластерну архітектуру з багатопроцесорними чи багатоядерними вузлами приводить до наступної модифікації

$$x_j^k = \frac{1}{a_{jj}}\left(b_j - \sum_{m=1}^{j-1} a_{jm}x_m^{k-1} - \sum_{m=j+1}^{N} a_{jm}x_m^{k-1}\right) \text{ при } j = M;$$

$$x_j^k = \frac{1}{a_{jj}}\left(b_j - \sum_{m=1}^{M-1} a_{jm}x_m^{k-1} - \sum_{m=M}^{j-1} a_{jm}x_m^{k} - \sum_{m=j+1}^{N} a_{jm}x_m^{k-1}\right) \text{при } M < j < M+S; \qquad (4)$$

$$x_j^k = \frac{1}{a_{jj}}\left(b_j - \sum_{m=1}^{M-1} a_{jm}x_m^{k-1} - \sum_{m=M}^{M+S} a_{jm}x_m^{k} - \sum_{m=M+S+1}^{j-1} a_{jm}x_m^{k-1} - \sum_{m=j+1}^{N} a_{jm}x_m^{k-1}\right)$$

$$\text{при } M+S < j = N.$$

Отже, при розв'язуванні СЛАР кластерною обчислювальною системою отримуємо дещо уповільнений алгоритм (4), який займатиме по ефективності проміжне положення між алгоритмами методу Гауса-Зейделя (2) і Якобі (3). Ефективність врахування вже здобутої на поточній ітерації інформації буде тим меншою, чим більша кількість процесів використовуватиметься при проведенні обчислень і навпаки, тим більшою, чим більша кількість потоків у межах кожного з процесів буде задіяною. Водночас, слід зауважити, що найсуттєвішим технологічним обмеженням багатопоточної технології паралельних обчислень може виступити пропускна здатність каналу процесор-

оперативна пам'ять. З іншого боку, реалізація розгалуженого алгоритму, орієнтованого під розподілену кластерну архітектуру, передбачає після здійснення кожної з ітерацій необхідність передачі отриманою кожним процесом (вузлом) інформації по уточненню частини значень x_j^k, $j = \{1,...,N\}$ решті процесів, що також зумовлює додаткові витрати як часу так і процесорних ресурсів для реалізації цього обміну даними.

З наведених двох концептуально різних принципів реалізації паралелізму ітераційного розв'язання СЛАР при орієнтації на системи з SMP та розподіленою (кластерною) архітектури випливає висновок про те, що як при застосуванні багатопоточних алгоритмів, так і при використанні технологій розподілених обчислень існують різні за природою виникнення джерела гальмування процесу паралельних обрахунків. Враховуючи на те, що сьогодні спостерігається інтенсивний розвиток багатоядерних технологій процесоробудування і їх конвергенція з традиційними кластерними архітектурами, вузли яких стають не лише багатопроцесорними, що досить типово, а й багатоядерними, є актуальним враховувати ці тенденції і нові архітектурні особливості, а також оцінити додаткові витрати, обумовлені розгалуженням алгоритму. Для вирішення цієї задачі було створено програму мовою Fortran, що реалізує розбиття системи заданого порядку N на частини, які у свою чергу обробляються відповідними процесами згідно формули (4). Передачу інформації між процесами реалізовано шляхом використання інструкцій інтерфейсу передачі повідомлень (Message Passing Interface – MPI). Для трансляції програми у виконавчий код використовувався компілятор, розроблений компанією Intel (ver. 11).

Результати дослідження ефекту розпаралелювання при розв'язанні СЛАР. При виконанні дослідження використовувалася наступна стратегія: коефіцієнти СЛАР генерувалися випадковим чином, для полегшення аналізу ефективності обчислень і мінімізації впливу порядку СЛАР та описаної вище неможливості реалізувати алгоритм Гауса-Зейделя при розподілених обчисленнях у повній мірі, замість умови виходу з ітераційного процесу по досягненню наперед заданої точності використовувалася умова здійснення фіксованого числа ітерацій $k = 1000$. Кожен розрахунок здійснювався п'ять разів, після чого знаходилося середнє арифметичне здобутих результатів. Першим етапом дослідження, представленим рис. 1, було визначення ефективності розпаралелювання за технологією MPI при розв'язанні СЛАР у залежності від її порядку, яка оцінювалася кількісно коефіцієнтом прискорення, тобто тобто часткою часу, необхідного для проведення обчислень за паралельним алгоритмом по відношенню до послідовного алгоритму розв'язування тієї самої задачі. З метою забезпечення більш наочного аналізу витрат, обумовлених мережевим інтерфейсом, при аналізі ефективності розв'язання СЛАР на цьому етапі дослідження використовувався Fast Ethernet (100 Mbps/s). З наведеної ілюстрації випливає, що застосування технології MPI розпаралелювання не є ефективним при порядках СЛАР, менших за 1000. При $N = 2000$ максимальний ефект від застосування розпаралелювання дорівнював подвійному прискоренню обчислень при 3-х задіяних процесорах, але подальше їх збільшення до 4-х ситуації не покращувало, а подальше збільшення кількості використаних процесорів призводило до деякого (близько 10 %) погіршення найкращого результату.

Описаний ефект немонотонної поведінки залежності коефіцієнта прискорення від кількості використаних процесорів спостерігається і для більших порядків СЛАР ($N = 4000,\ 6000$), але екстремум зростає і зсувається в напрямку більшої кількості процесорів. Отже збільшення порядку СЛАР є сприятливим фактором для більш ефективної реалізації технологій паралельних обчислень на розподіленій системі. При $N = 7000$ можна спостерігати в усьому дослідженому діапазоні восьми процесорів монотонне з незначним відхиленням від лінійної залежності зростання ефективності обчислень. Але, навіть і в цьому випадку, якщо при числі процесорів $R = 2$ маємо прискорення в 1.7 разів, тобто витрати на реалізацію технології паралелізму обчислень складають 30 %, то при використанні $R = 8$ процесорів втрати вже становлять 43.5 %, що обумовлено як зростанням кількості операцій обміну, так і більшим сукупним обсягом даних, що передаються між вузлами кластеру і обумовленою цим додатковою завантаженістю процесорів, які замість розв'язання прикладної задачі здійснюють додаткові операції по обслуговуванню її належного функціонування на розподіленій обчислювальній системі.

Більш детальну інформацію стосовно протікання процесів міжвузлового обміну даними можна отримати з аналізу завантаженості мережного інтер-фейсу (Fast Ethernet) при розв'язанні даної задачі, представленої на рис. 2 (другий етап дослідження). Ці дані знімалися одночасно на кількох вузлах з диспетчеру задач Windows XP безпосередньо в процесі проведення обрахунків.

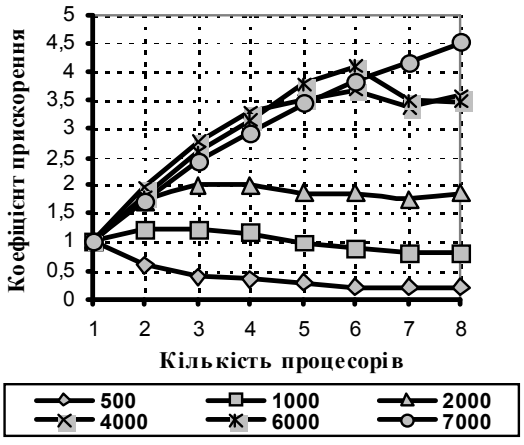

Рисунок 1 – Порівняння залежності коефіцієнту прискорення обчислень від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування (Fast Ethernet)

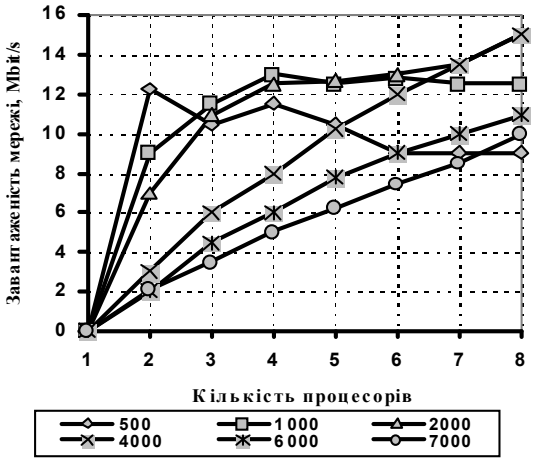

Рисунок 2 – Порівняння залежності завантаженості мережі Fast Ethernet від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування

З наведеного співставлення випливає, що мережа при $R \le 8$ процесорів не використовує більш ніж 15 % своєї максимальної пропускної здатності в усьому діапазоні досліджених порядків СЛАР, але збільшення порядку призводить до більш лінійної залежності між завантаженістю мережі та кількістю задіяних процесорів. При значних порядках СЛАР $N \ge 4000$ спостерігається тенденція поступового зменшення завантаженості мережі при одному і тому ж числі використаних процесорів $R = const$, що,

скоріше за все, обумовлено зростанням сукупного часу на здійснення однієї ітерації у порівнянні з долею часу, що витрачається мережею на міжвузловий обмін.

Ще однією дослідженою вагомою характеристикою стала завантаженість процесора під час процесу паралельного розв'язання СЛАР різних порядків при різній кількості задіяних процесорів (третій етап дослідження). Відповідні результати (рис. 3), як і у попередньому випадку, знімалися безпосередньо в процесі проведення обрахунків одночасно на кількох вузлах з диспетчеру задач Windows XP.

Отримані результати дають змогу пояснити причину низької ефективності розв'язання СЛАР порівняльно невисокого порядку ($N \le 2000$) при зростанні кількості використаних процесорів. У цьому випадку спостерігається різке падіння завантаженості процесорів, яке вже при $R \ge 4$ не перевищує 55 %, тобто розподілена задача неспроможна завантажити усі процесори обчисленнями і вони переважно простоюють між ітераціями, очікуючи завершення порівняльно повільних операцій міжвузлового обміну, який, як це слідує з рис. 2, також не в змозі завантажити мережевий інтерфейс більщ ніж на 12–15 %. Збільшення порядку СЛАР ($N \ge 4000$) є сприятливим щодо утилізації ресурсів процесора, оскільки співвідношення між витратами часу, необхідними для проведення обчислень і для здійснення міжвузлового обміну, схиляється в бік перших. У результаті сукупна завантаженість процесора зростає і негативна роль витрат, обумовлених операціями міжвузлової передачі даних, зменшується.

Збільшення числа задіяних процесорів у випадку великих порядків СЛАР ($N \ge 4000$) призводить до практично лінійного зменшення завантаженості процесорних ресурсів кластера, але, як це слідує з рис. 3, втрати процесорної ефективності при $R = 8$ не перевищують 40 %.

Таким чином, отримані результати дозволяють проаналізувати внесок досліджених складових витрат комп'ютерних ресурсів в ефективність розв'язування СЛАР ітераційним методом Гауса-Зейделя і дійти висновку про те, що більшої ефективності і на більшій кількості процесорів можна очікувати від розпаралелювання при зростанні порядку СЛАР. Зменшити витрати обчислювальної ефективності кластера можна при якомога більшому використанні мультипроцесорних чи багатоядерних потужностей кожного з вузлів шляхом застосування багатопоточних технологій програмування поряд з методами розподілених обчислень, що сприятиме мінімізації міжвузлового обміну даними і, як наслідок, збільшенню ефективності функціонування паралельних алгоритмів.

Наступний етап дослідження стосувався здійснення тих самих обрахунків, але при заміні комутації вузлів з Fast Ethernet на більш продуктивний Gigabit Ethernet (1000 Mbps). Окрім вдесятеро більшої пропускної здатності він характеризується ще й меншою латентністю, що позитивно впливає на взаємодію вузлів при розв'язуванні паралельної задачі. Рис. 4–6 наочно ілюструють результат цього ефекту, оскільки коефіцієнт прискорення обчислень (рис. 4) вже демонструє зростання починаючи вже з $N \ge 1000$, а не $N \ge 4000$, як це було для Fast Ethernet комунікаційного інтерфейсу. При кількості задіяних процесорів $R = 5$ значення коефіцієнта прискорення при $N = 7000$ збільшилося з 3.44 для Fast Ethernet до 4.55 для Gigabit Ethernet, отже втрати на реалізацію паралелізму для цього розрахункового випадку зменшилис з 31.2 % до 9 % виключно за рахунок зміни продуктивності інтерфейсу обміну між вузлами.

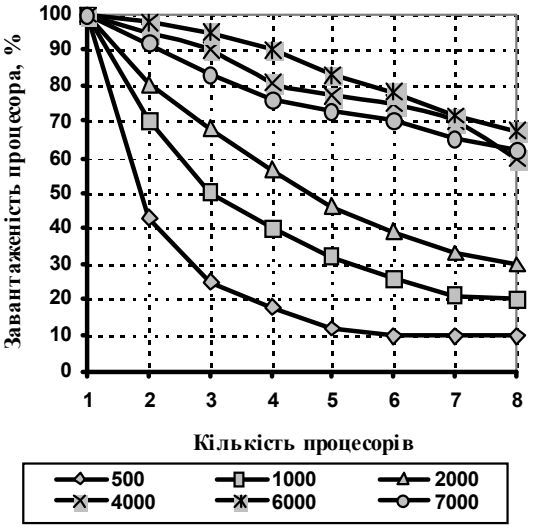

Рисунок 3 – Порівняння залежності завантаженості процесора від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування (Fast Ethernet)

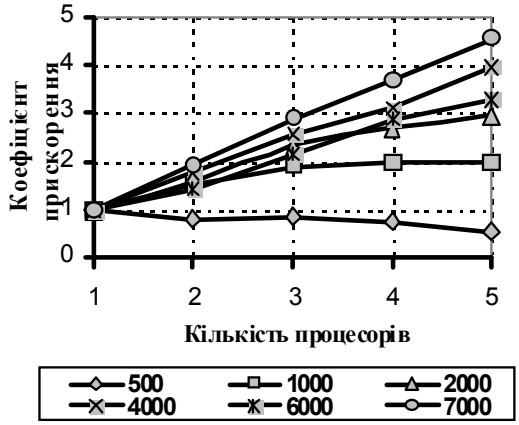

Рисунок 4 – Порівняння залежності коефіцієнту прискорення обчислень від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування (Gigabit Ethernet)

Залежності набули більш лінійного вигляду, отже продуктивність обчислень зростає в більшій мірі пропорційно до кількості задіяних процесорів. Одночасно суттєво (до 7.7 разів при $R = 5$) зросла інтенсивність обміну даними між вузлами, що ілюструється рис. 5. При максимально теоретично можливому відношенні у 10 разів, що визначається гранично можливими специфікаціями використаних мережних інтерфейсів, отриманий показник є обнадійливим позитивним результатом.

Більш ефективна взаємодія знаходить своє безпосереднє віддзеркалення і в різкому зростанні завантаженості процесорів, що демонструє рис. 6. Якщо для СЛАР з і $R = 5$ процесори відпрацьовували лише 83 % своєї максимальної продуктивності за умови використання Fast Ethernet, то заміна останнього на Gigabit Ethernet забезпечує вже 95 % утилізації ресурсів процесорів, тобто у цьому випадку мережевий інтерфейс вже в значно меншій мірі стає вузьким місцем технології розподілення обчислень при ітераційному розв'язання СЛАР.

Отже, усі компоненти обчислювальної кластерної системи при застосуванні більш швидкодіючого інтерфейсу працюють більш злагоджено.

Подальшу перевірку цього результату буде здійснено шляхом розгляду більш конкретної аерогідродинамічної задачі.

Постановка задачі, вихідні рівняння. Розглянемо течію в'язкої нестисливої рідини, що формується в замкненій порожнині у формі прямокутного паралелепіпеда. Верхня кришка порожнини вважається рухомою, швидкість її руху u_H є сталою заданою величиною. В результаті в порожнині утворюється циркуляційний рух рідини, що її заповнює, властивості якого залежать від геометричних особливостей і режимних параметрів формування течії, і, зокрема, від кута, під яким рухається верхня кришка по відношенню до бічних граней порожнини. Обмежимо розгляд ламінарним режимом і порівняльно невеликим значенням числа Рейнольда ($\mathrm{Re} = 200$), що дозволить уникнути проблем, пов'язаних із забезпеченням збіжності ітераційних розрахункових процедур і зосередитися на побудові розгалужених алгоритмів обчислень.

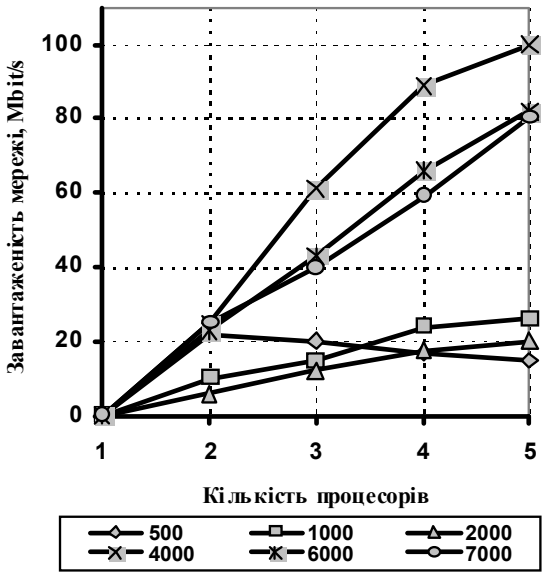

Рисунок 5 – Порівняння залежності завантаженості мережі Gigabit Ethernet від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування

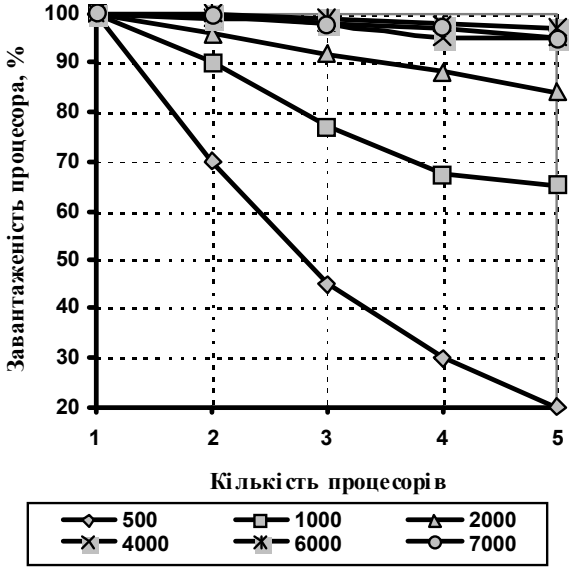

Рисунок 6 – Порівняння залежності завантаженості процесора від кількості задіяних процесорів при розв'язанні СЛАР різного порядку шляхом застосування MPI технології паралельного програмування (Gigabit Ethernet)

Числове розв'язування цієї задачі ґрунтується на системі рівнянь Нав'є-Стокса, яка в просторовій декартовій системі координат (x, y, z) після перетворень, притаманних методу SIMPLE (Semi Implicit Method for Pressure-Linked Equations), може бути подана у наступній узагальненій формі:

$$\frac{\partial}{\partial x}A_x\overline{\varphi}+\frac{\partial}{\partial y}A_y\overline{\varphi}+\frac{\partial}{\partial z}A_z\overline{\varphi}=S_\varphi+\frac{\partial}{\partial x}\left(F\frac{\partial\overline{\varphi}}{\partial x}\right)+\frac{\partial}{\partial y}\left(F\frac{\partial\overline{\varphi}}{\partial y}\right)+\frac{\partial}{\partial z}\left(F\frac{\partial\overline{\varphi}}{\partial z}\right), \qquad (5)$$

де $\varphi=\{u,v,w,p\}$ – узагальнена розрахункова змінна; A_x, A_y, A_z – коефіцієнти конвективного переносу в напрямках відповідних координатних осей x, y, z; F – дифузійний коефіцієнт; S_φ – джерельний член. Обезрозмірення декартових координат x, y, z проводиться по характерному розміру розрахункової області L, за який прийматимемо висоту порожнини, а саме: $\overline{x}=x/L$, $\overline{y}=y/L$, $\overline{z}=z/L$. Отже, просторова розрахункова область D характеризуватиметься значеннями: $0\le x\le x_{\max}$, $0\le y\le y_{\max}$, $0\le z\le L$. Обезрозмірення компонент швидкості є наступним: $\overline{u}=u/u_H$, $\overline{v}=v/u_H$, $\overline{w}=w/u_H$, тиск обезрозмірюється згідно формули $\overline{p}=p/\rho u_H^2$, де ρ – густина. Параметри рівняння визначаються залежно від розрахункової змінної згідно таблиці 1.

Система визначених таблицею 1 чотирьох рівнянь розв'язується з наступними граничними умовами:

На обтічних поверхнях (стінках порожнини):

$$\overline{u}=0,\ \overline{v}=0,\ \overline{\omega}=0. \qquad (6)$$

Таблиця 1 – Структурні елементи рівняння (5)

φ	A_x	A_y	A_z	F	S_φ
u	$\bar{u}$	$\bar{v}$	$\bar{w}$	$\bar{\nu} = \frac{1}{\mathrm{Re}}$	$-\frac{\partial \bar{p}}{\partial \bar{x}}$
v	$\bar{u}$	$\bar{v}$	$\bar{w}$	$\bar{\nu} = \frac{1}{\mathrm{Re}}$	$-\frac{\partial \bar{p}}{\partial \bar{y}}$
w	$\bar{u}$	$\bar{v}$	$\bar{w}$	$\bar{\nu} = \frac{1}{\mathrm{Re}}$	$-\frac{\partial \bar{p}}{\partial \bar{z}}$
p'	0	0	0	1	За методом SIMPLE

На кришці з урахуванням її рухомості:

$$\bar{u} = \bar{u}_{Hx} = const\ ,\ \bar{v} = 0\ ,\ \bar{w} = \bar{u}_{Hz} = const\ . \qquad (7)$$

Тиск визначається в процесі ітераційного відшукання розв'язку за методом SIMPLE.

Математичні подробиці побудови паралельних алгоритмів. Зв'яжемо осі координат з ребрами порожнини і покриємо просторову розрахункову область D сіткою, в вузлах якої будемо обчислювати значення розрахункових змінних. Для даної модельної задачі з метою спрощення використаємо рівномірну сітку, причому кількість вузлів сітки в обох напрямках візьмемо однаковою. Розіб'ємо побудовану сітку на блоки вздовж останнього напрямку z так, щоб вузли на межах блоків перекривалися, тобто межовий вузол одного блоку є внутрішнім для сусіднього з ним і навпаки. Скінченно-різницеву апроксимацію рівняння (5) для кожної зі змінних представимо на п'ятиточковому шаблоні і організуємо для кожного з рівнянь системи (табл. 1) послідовний обхід всіх внутрішніх точок для окремо виділеного блоку згідно розрахункової процедури SIMPLE. На кожному кроці ця процедура виконується для кожного з блоків, а межові значення розрахункових змінних корегуються залежно від використаної технології розпаралелювання. У дослідженні одночасно використано два підходи: OpenMP [5] та MPI [6]. Перший з них (OpenMP) орієнтований на багатопоточне програмування і є ефективним в багатопроцесорних чи багатоядерних системах. Другий підхід (MPI – Message Passing Interface) є ефективним для розподілених обчислювальних систем (кластерів). Ці технології є принципово відмінними і, певною мірою, конкуруючими.

У даній роботі здійснено спробу їх узгодження шляхом комбінування (рис. 7), при якому уся розрахункова область ділиться пропорційно кількості вузлів кластера і утворені підобласті обробляються відповідними процесами інтерфейсу MPI з притаманним йому міжвузловим обміном проміжними результатами обчислень на внутрішніх суміжних межах.

У свою чергу, кожна з цих підобластей ділиться пропорційно кількості арифметичних пристроїв вузла, що визначається добутком кількості процесорів у вузлі та кількості ядер кожного процесора. Кожен з цих елементів області опрацьовується відповідним ядром процесора шляхом генерації OpenMP потоків, які, на відміну від MPI процесів, працюють зі спільною оперативною пам'яттю кластерного вузла і тому не потребують обміну даними між собою. Такий підхід, на відміну від реалізації традиційної стратегії виключно MPI програмування (як це було виконано вище для СЛАР), дозволяє уникнути додаткових витрат ресурсів і затримок, обумовлених непродуктивним і зайвим у цьому випадку обміном даних в межах кожного з вузлів кластерної системи.

Загальна характеристика застосованої кластерної установки. Для проведення даного етапу досліджень використано кластерну систему, побудовану з п'ятьох вузлів, об'єднаних гігабітною мережею. Кожен з них містив двоядерний процесор Intel Core 2 Duo E6400 з тактовою частотою 2.13ГГц та 2 ГБ оперативної пам'яті DDR2-800. Для тестування суто багатопоточної OpenMP технології було застосовано комп'ютер на базі чотирьохядерного процесора Intel Core 2 Quad Q6600 з тактовою частотою 2.4 ГГц та 4 ГБ оперативної пам'яті DDR2-800.

Результати досліджень. Проаналізуємо особливості реалізації кожної з розглянутих вище технологій розпаралелювання стосовно сформульованої задачі. На рис. 8 представлені результати обчислень даної задачі за OpenMP технологією, збільшення потужності сітки (трикутники – 102 вузли в кожному напрямку; квадрати – 202 вузли; кола – 302 вузли; ромби – 402 вузли).

Як це випливає з отриманих даних, збільшення обсягів обчислень погіршує ефективність розпаралелювання розрахунків, яке тут, як і раніше, оцінювалося коефіцієнтом прискорення. Пояснення цього ефекту полягає в тому, що при зростанні обчислювальної роботи збільшується і кількість даних, які знаходяться в оперативній пам'яті і одночасно опрацьовуються усіма потоками, що обмежується пропускною здатністю каналу процесор-оперативна пам'ять.

При застосуванні суто MPI технології та проведенні розрахунків на відповідній розподіленій кластерній системі маємо протилежну тенденцію (рис. 9, 10, трикутники): коефіцієнт прискорення обчислень зростає по мірі збільшення розмірності задачі. Тлумачення цього результату полягає в тому, що при збільшенні кількості вузлів n в кожному з напрямків навантаження на обчислювальні ресурси вузла зростає пропорційно n^3, а міжвузловий обмін даними з суміжних граней під областей збільшується лише пропорційно n^2. Отже, доля ресурсів, яку треба витратити на непродуктивний, але необхідний елемент цієї технології неухильно зменшується в обсязі усієї обчислювальної роботи по мірі зростання потужності сітки.

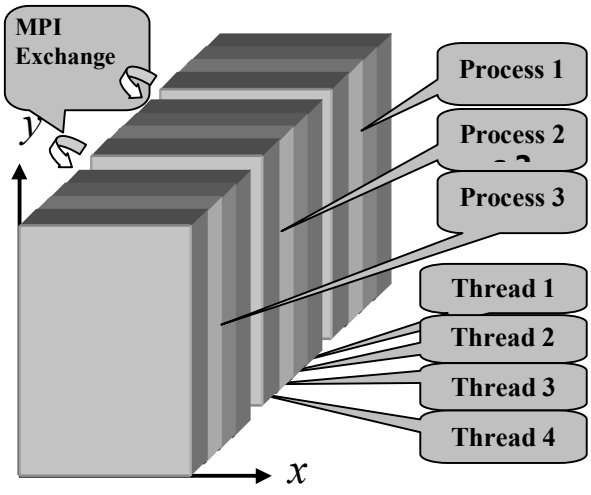

Рисунок 7 – Комбінована декомпозиція розрахункової області при використанні MPI та OpenMP технологій паралельних обчислень

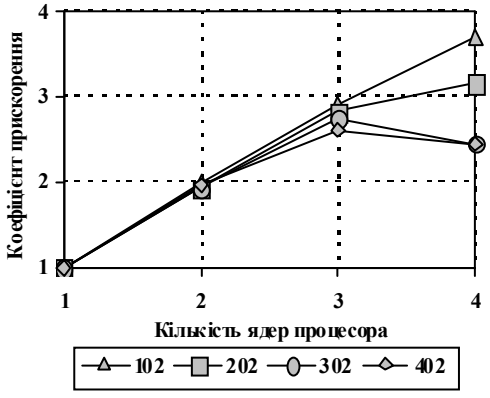

Рисунок 8 – Залежність коефіцієнту прискорення обчислень від розмірності сітки та кількості задіяних ядер процесора при реалізації OpenMP технології багатопоточного програмування

Реалізована у даному дослідженні ідея автора по побудові гібридного алгоритму, який містить в собі дві проаналізовані вище технології, як це випливає з рис. 9, 10 (кола), виявилася спроможною одночасно використати переваги і нівелювати недоліки двох її складових.

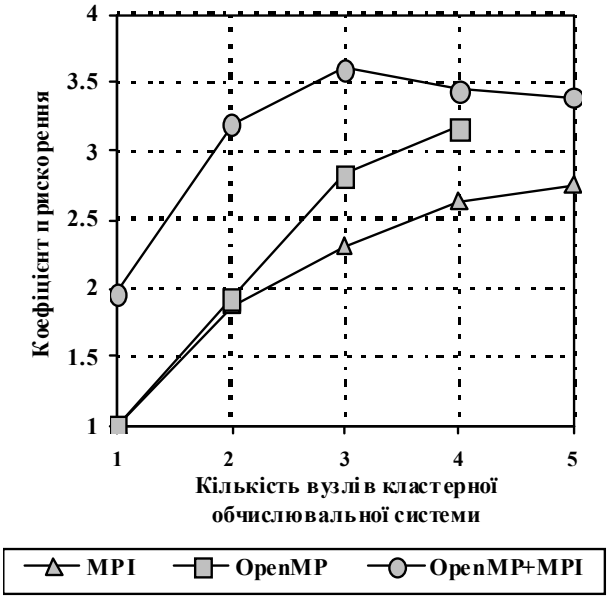

Рисунок 9 – Порівняння залежності коефіцієнту прискорення обчислень при реалізації різних технологій паралельного програмування: MPI, OpenMP та OpenMP+MPI $N = 202^3$ вузлів

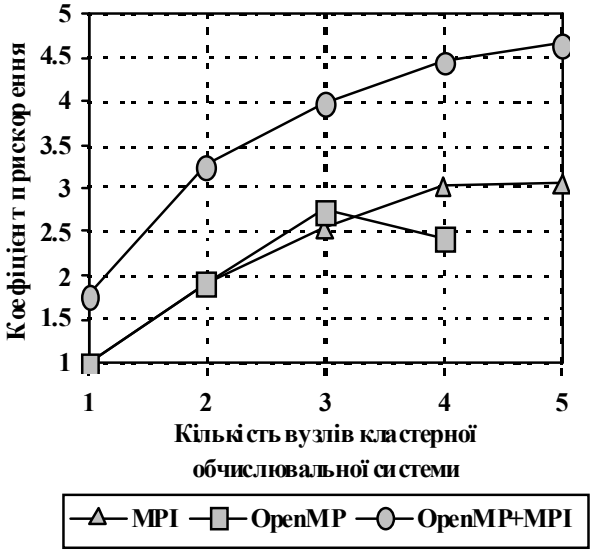

Рисунок 10 – Порівняння залежності коефіцієнту прискорення обчислень при реалізації різних технологій паралельного програмування: MPI, OpenMP та OpenMP+MPI. $N = 302^3$ вузлів

За рахунок використання в межах вузла лише OpenMP підходу, вдається, як зазначалося вище, зменшити непродуктивний обмін даними, а також знизити навантаження на процесори та мережний інтерфейс шляхом мінімізації міжвузлового обміну даними. З іншого боку, завдяки використанню MPI декомпозиції, зменшується навантаження на кожен з вузлів кластера, результатом чого є зростання ефективності багатопоточної обробки. Таким чином, розроблений гібридний OpenMP+MPI підхід до розпаралелення обчислень є більш ефективним у порівнянні з кожною з технологій, застосованих окремо. З наведених на рис. 9, 10 результатів слідує, що на п'ятивузловому кластері з двоядерними вузлами вдалося отримати прискорення розв'язування задачі моделювання течії в просторовій порожнині в 4,66 разів при загальній кількості вузлів сітки $27,5 \cdot 10^6$.

Висновки

1. Проаналізовано ефективність розпаралелювання ітераційного методу розв'язування СЛАР по завантаженню процесорів, комунікаційного інтерфейсу та коефіцієнту прискорення обчислень у цілому для двох типів мереж, що об'єднують вузли кластера (Fast та Gigabit Ethernet);

2. Розроблено гібридний метод розпаралелювання обчислень, що об'єднує MPI та OpenMP технології розподіленого та багатопоточного програмування;

3. Виконано адаптацію цього метода до моделювання течії в порожнині у формі прямокутного паралелепіпеда;

4. Отримані результати продемонстрували переваги розробленого методу побудови паралельних алгоритмів перед кожною з технологій, що є його складовою з точки зору ефективності розпаралелювання;

5. Реалізація цього методу при алгоритмізації та програмуванні не потребує зна-

чного ускладнення та використання суттєвих додаткових ресурсів, забезпечуючи в результаті кращу адаптацію до сучасних архітектур кластерних обчислювальних систем та зменшення витрат на чисельні експерименти;

6. Подальше удосконалення цього метода полягатиме у його застосуванні до моделювання турбулентних течій, зокрема, на основі методу моделювання динаміки великих вихорів.

Література

1. Лойцянский Л.Г. Механика жидкости и газа / Л.Г. Лойцянский – М.: "Наука", 1987. – 840 с.

2. Ламб Г. Гидродинамика/ Г. Ламб – М.: Гостехиздат, 1947. – 928 с.

3. Шлихтинг Г. Теория пограничного слоя/ Г. Шлихтинг – М.: Изд. иностр. лит., 1959. – 528 с.

4. Kawamura H, Abe H. DNS of Turbulent Heat Transfer in Channel Flow With Respect to Reynolds-Number Effect/ H. Kawamura, H. Abe // Proc. of the 2nd Engineering Foundation Conference in Turbulent Heat Transfer. – Manchester, UK, 1998. – V. 1 – P. 1–15 – 1–22.

5. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие/ **А.С.** Антонов – М.: МГУ, 2009. – 77 с.

6. Matloff's N. MPICH/MPICH2 MPI Tutorial/ N. Matloff's Електронний ресурс. Режим доступу: http://heather.cs.ucdavis.edu/~matloff/MPI/NotesMPICH.NM.html.

УДК 681.322

Шквар Е.А.

ИНТЕГРИРОВАННАЯ ГИБРИДНАЯ ТЕХНОЛОГИЯ ПАРАЛЛЕЛЬНЫХ ВИЧИСЛЕНИЙ

Выполнен анализ эффективности применения интегрированной гибридной параллельной технологии ресурсоемких вычислений, являющейся адаптированной к применению на кластерных вычислительных системах с многопроцессорными или многоядерными узлами, которые одновременно обеспечивают симметричный и распределенный виды мультипроцессорной обработки. Приведены примеры применения предложенного подхода, проанализированы причины замедления масштабирования производительности вычислений по мере наращивания вычислительной мощности распределенной системы. Определены приоритетные направления применения разработанного вычислительного метода.

Shkvar Ye.O.

INTEGRATED HYBRID TECHNOLOGY OF PARALLEL COMPUTATIONS

The efficiency of integrated hybrid parallel technology of resource-intensive computations is analyzed. It is oriented on the use of cluster computer systems with multiprocessor or multicore nodes, which are able to realize simultaneous symmetric and distributed types of multiprocessing. The examples of the proposed approach are presented; the reasons of computational productivity scaling delay as a result of growing the processing power of distributed system are analyzed. The priority guidelines of the developed computational method are worked out.