УДК 004.451.24

СТІРЕНКО С.Г.

РЕАЛІЗАЦІЯ МЕТОДУ ВИЗНАЧЕННЯ МОМЕНТУ ВІДКЛАДЕННЯ ПЕРЕСИЛАННЯ ДАНИХ В КОМП'ЮТЕРНИХ КЛАСТЕРНИХ СИСТЕМАХ

В роботі розглядаються реалізація методу визначення моменту відкладення передачі даних, що ефективно використовуються під час виконання балансування навантаження. На основі проведених експериментів та аналізу їх результатів можна стверджувати, що метод визначення моменту відкладання пересилання даних може бути ефективно застосовано для підвищення функціонування паралельних кластерних систем.

We consider the implementation of the method for determining the date of the deposition of data that are being used while performing load balancing. Based on the experiments and analyze their results could be argued that the method of determination of a data transmission delay can be effectively used to improve the functioning of parallel cluster systems.

Особливості реалізації в кластерних системах

Основними системами з локальною пам'яттю на даний момент є кластерні системи, що складаються з вузлів, кожен з яких має спільну пам'ять для декількох процесорів, з'єднаних високопродуктивною мережею. Переважна більшість існуючих кластерних систем надає доступ до низькорівневого інтерфейсу передачі повідомлень MPI як до інтерфейсу програмування обчислювальної системи. Структура цього інтерфейсу дозволяє абстрагувати апаратні особливості тієї чи іншої системи, зокрема систему зв'язків та нестрогу неоднорідність обчислювальних вузлів, дозволяючи виконувати обчислення на будь-якому з них та організовувати передачу даних між будь-якою парою вузлів. Таким чином, MPI є фактично інтерфейсом до більшості кластерних систем, відповідно доцільним є використовувати його для безпосередньої передачі даних, тобто ввести додатковий рівень абстракції над MPI. Змінення поведінки безпосередньо інтерфейсу MPI небажана, оскільки може вплинути на вже існуючі програми з його використанням, які розраховують на відому поведінку.

Додатковий рівень абстракції може також забезпечити підтримку моделі, що використовує підзадачі зі шляхами розбиття для забезпечення прогнозування часу передачі даних між вузлами та часу виконання обчислень підзадачі. На цьому рівні можна виконувати збір різного роду статистичної інформації із застосуванням інтерфейсу визначення продуктивності MPI performance API, який надає платформонезалежні методи доступу до інформації про показники продуктивності, що є вкрай важливим для підтримки достатньо великої кількості обчислювальних систем. Робота на тому ж самому рівні, що й MPI, вимагала б повторної реалізації тих самих механізмів. Аналогічним чином, додатковий рівень абстракції з використанням поділу задач на частини був запропонований в технології Intel Threading Building Blocks [1] для обчислювальних систем зі спільною пам'яттю, однак його модель не передбачає необхідність передачі даних. Тому доцільно зберігати сумісність інтерфейсів з TBB, якщо можливо. З точки зору доступних користувачу дій, основними механізмами розпаралелювання є паралельні згортка та сканування, причому конкретна реалізація паралельного обчислення не визначається, тому існує можливість використовувати такий самий інтерфейс. З іншого боку, підтримка атомарних операцій або взаємного виключення технічно неможлива в системах з локальною пам'яттю, тому ця частина інтерфейсу TBB не має підтримуватись.

Таким чином найбільш доцільним підходом до реалізації запропонованого відкладення передачі даних [2,3,4] є введення додаткового рівня абстракції над MPI, який дозволить виконувати необхідні дії та вимірювати певні метрики ефективності. Це також дозволить користувачу обчислювальної системи описувати способи передачі даних з використанням стандартизованих засобів MPI, однак момент безпосереднього виклику цих засобів буде визначатися системою під час виконання.

На цьому рівні необхідно реалізувати підтримку необхідних механізмів для відкладення передачі даних. Оскільки на даний

момент безпосередньо MPI не надає подібної можливості окрім дуже обмеженої підтримки буферизованих передач, необхідно ввести додатковий керуючий процес, який буде оброблювати запити за передачу даних. За умови відкладення початку передачі даних, у процес, що має їх приймати, замість безпосередньо даних відправляється спеціальна структура даних – дескриптор, – яка містить інформацію про обсяг даних та їх розташування на деякому вузлі. Ця структура може бути використана для виконання явного запиту даних, якщо вони необхідні раніше, ніж були передані. Також такий підхід дозволяє передавати лише дескриптори, розміри яких зазвичай значно менше обсягів даних, що передаються, при перенесенні обчислень між вузлами. У керуючому процесі підтримується впорядкований за часом список відкладених передач. Для визначення часу використовуються високоточні таймери, наявні в сучасних процесорах, роздільна здатність яких досягає наносекунд. [5] Так як в рамках вузла використовується система, побудована за принципами розподілу часу, необхідно врахувати її особливості під час ініціювання початку передачі даних. Для цього виявляється кількість процесів, що знаходяться в активному стані. Якщо передача даних має бути ініційована менш ніж через час, що дорівнює отриманій кількості процесів на квант часу, передачу можна ініціювати негайно.

Типи задач, для яких виконувалось тестування

Для демонстрації принципів роботи запропонованих підходів відкладення початку передачі даних необхідно застосувати їх до ряду типових обчислювальних задач, які виконуються на системах з локальною пам’яттю. Оскільки запропонована для досліджень реалізація використовує додатковий рівень абстракції над низькорівневим інтерфейсом передачі повідомлень, неможливо безпосередньо використати її для існуючих програм із застосуванням MPI, а необхідно виконати їх адаптацію. Тому було обрано ряд типових задач, які мають характерні особливості, поведінку та схеми взаємодії, та адаптовано для використання дослідницької реалізації.

З точки зору організації взаємодії та обсягів даних, що передаються між вузлами обчислювальної системи з локальною пам’яттю, а також обсягів пам’яті, що використовуються у вузлах, можна виділити наступні широкі класи задач.

а) Задачі з рівномірним використанням пам’яті на вузлах та рівномірним обсягом даних, що передаються між вузлами. При цьому, взаємодія може відбуватись лише між обраними парами вузлів, однак необхідною умовою є стабільність середньої кількості переданих даних за певний проміжок часу між всіма парами вузлів. Прикладами таких задач є операції лінійної алгебри із щільними векторами, матрицями та тензорами, що зберігаються простим способом, зокрема матрично-тензорна алгебра, одно- та багатовимірні згортки, гаусово розмиття, дискретні перетворення в системах ортогональних функцій, зокрема перетворення Фур’є; розв’язок диференційних рівнянь на рівномірних сітках методами кінцевих різниць тощо.

б) Задачі, в паралельних реалізаціях яких комунікація між процесами здійснюється відповідно до певного невипадкового закону, зокрема поліноміальному, причому обсяги комунікації між певними парами процесів значно більше ніж між іншими парами процесів. Використання пам’яті у вузлах також може бути нерівномірним, але невипадковим, причому не є обов’язково безпосередньо пов’язаним з обсягами комунікації. До цього класу можуть бути віднесені задачі чисельного математичного аналізу, зокрема чисельного інтегрування зі складними характеристиками сходимості; ітераційні методи, що виконуються на інтервалах до сходимості, зокрема методи розв’язку систем лінійних рівнянь; розв’язок диференційних рівнянь методами кінцевих різниць на кінцевих елементів з адаптивними або нерівномірними сітками тощо.

в) Задачі, в яких обсяги комунікації між процесами при паралельному розв’язані можуть варіюватися випадково в залежності від значень вхідних даних, а не від їх обсягу, або задачі, в яких обсяги використаної пам’яті можуть випадковим чином залежати від значень вхідних даних або результатів, отриманих при попередніх обчисленнях. Прикладами таких задач є задачі лінійної алгебри над довільними розрідженими матрицями з випадковим заповненням; ймовірнісні методи, зокрема методи Монте-Карло; моделювання випадкових процесів; деякі обчислювально складні задачі на графах.

Така класифікація не претендує на повноту або покриття всіх можливих обчислювальних задач, але відображає достатньо велику кількість типових обчислень, що виконуються

на кластерних системах. Крім того, вона дозволить достатньо швидко оцінити ефективність застосування запропонованих підходів до відкладення передачі даних без необхідності реалізовувати велику кількість задач із застосуванням цих підходів.

Для проведення експерименту з ціллю підтвердження впливу на коефіцієнт ефективності запропонованих підходів та перевірки висунутих гіпотез, було обрано та реалізовано із застосуванням моделі розбиття на підзадачі та зв'язані з ними дані, по дві наступні задачі кожного класу:

а) виконання дискретного перетворення Фур'є; чисельне розв'язання диференційного рівняння квантової оптики методом кінцевих різниць з рівномірною сіткою;

б) чисельне інтегрування методом кривих вищих порядків; чисельне розв'язання диференційного рівняння квантової оптики методом кінцевих різниць з адаптивної сіткою;

в) знаходження власних чисел та векторів розрідженої матриці з випадковим заповненням; моделювання випадкового фізичного процесу методом Монте-Карло.

Обрані задачі використовуються у багатьох більш складних застосуваннях обчислювальних систем для розв'язку наукових та технічних задач, тому можливість збільшення коефіцієнту ефективності їх розв'язання дозволить також збільшити коефіцієнти ефективності виконання обчислень, які активно використовують розв'язання цих задач.

Крім того, оскільки обрані задачі та особливо їх конкретні можуть мати чітко виражені особливості організації взаємодії між паралельними процесами, в яких виконується їх обчислення, та схеми використання пам'яті, додатково було побудовано програмні моделі задач кожного класу.

Так для моделювання задачі класу (а) для кожної пари процесів визначається, чи буде виконуватися моделювання взаємодії між ними: якщо рівномірно розподілене на проміжку [0; 1) випадкове значення r перевищує певне порогове значення $r_{thresh} = 0.75$, то процеси вважаються взаємодіючими. Обрання порогового значення може залежати від кількості паралельних процесів, оскільки зазвичай у разі великої кількості процесів, взаємодія відбувається лише між окремими парами процесів. Кожний процес, що моделюється, являє собою чергування фази обчислень та фази передачі даних. Для моделювання обчислень обирається час їх виконання tc як випадкове значення, розподілене за нормальним законом $N(\mu,\sigma)$, зі значеннями $\mu = 10$ мс, $\sigma = 1$ мс, після чого в процесі виконується цикл, умовою виходу з якого є досягнення встановленого часу t_c. В циклі відбувається звернення до випадкових адрес пам'яті, яка доступна даній програмі, причому наступна адреса обирається як випадкова величина, розподілена за нормальним законом, з математичним очікуванням рівним наступній адресі для задовільнення принципу локальності звернення до даних. Під час обчислень виділяється певний обсяг пам'яті M, який обчислюється як $M = \frac{19m}{20} + m_{rand}$, де $m = 2^{29}$ байт – фіксований обсяг виділеної пам'яті, а m_{rand} – випадкова компонента обсягу пам'яті, що розподілена за нормальним законом $N(\frac{m}{10}, \frac{m}{100})$ та має на меті урахування побічних факторів, що можуть впливати на обсяги виділеної пам'яті. Виділена пам'ять заповнюється випадковими даними. Моделювання фази взаємодії виконується наступним чином: з використанням рівномірно розподіленого випадкового значення та порогу rthresh визначається, чи буде відбуватися взаємодія між парою процесів в даній фазі, якщо так, випадковим чином визначається процес-джерело та процес-приймач даних. Після чого визначається обсяг даних, які будуть передаватися $S = \frac{19m}{20} + s_{rand}$, де $s = 2^{22}$ байт – фіксований обсяг даних, що будуть передаватися, а s_{rand} – його випадкова компонента, розподілена за нормальним законом аналогічним до обсягів пам'яті. Виконується передача необхідного обсягу даних між вузлами, значення даних генеруються випадково.

Моделювання задачі класу (б) аналогічним до (а) чином визначає пари процесів, між якими відбуватиметься взаємодія. Однак в цьому випадку необхідно внести нерівномірність обсягів використаної пам'яті та даних. Для цього перед початком моделювання обирається два поліноми PM та PS від номеру процесу i: $P_4(i) = p_4 i^4 + p_3 i^3 + p_2 i^2 + p_1 i + p_0$, де p_i – коефіцієнти, що є випадковими значеннями, розподіленими за нормальним розподілом на проміжку [0; 3). В подальшому поліном $P^M(i)$ використовується для визначення обсягів пам'яті, що виділяються в i-тому вузлі, а поліном $P^S(i)$ – для визначення обсягів даних, що мають бути передані з i-го вузла у певний інший. Для отримання абсолют-

них значень обсягів даних значення поліномів домножаться на $\frac{10m}{P^M(N)}$ та $\frac{10s}{P^S(N)}$ відповідно, де N – кількість процесів.

Моделювання задач класу (в) відбувається наступним чином. На кожній фазі передачі даних для кожної пари процесів обирається чи буде відбуватися між ними взаємодія аналогічно до (а), однак на відміну від нього, будь-яка пара процесів потенційно може виконувати обмін даними. Моделювання процесу обчислень також відбувається за тими самими принципами, однак обсяги виділеної пам'яті визначаються як випадкова величина, розподілена за рівномірним законом на проміжку $[0.1m, 10m)$. Аналогічно обсяги даних, що будуть передані між кожною парою вузлів обираються як незалежні випадкові значення, розподілені за рівномірним законом на проміжку $[0.1s, 10s)$.

Наведені конкретні значення m та s, а також проміжки, у яких вибираються випадкові значення для обсягів пам'яті та даних, обрані у відповідності до статистики виконання задач на кластері Суперкомп'ютерного центру НТУУ «КПІ», типової конфігурації обладнання для обчислювальних систем такого типу (1 Гб оперативної пам'яті на один обчислювальний процес, та використання 10 Гб/с зв'язку між вузлами) а також з урахуванням загальної статистики [5], зокрема різниця між мінімальним та максимальним обсягом даних обрана як два десяткових порядки.

Використання пам'яті у вузлах системи

Процес виконання обчислень на обчислювальній системі з локальною пам'яттю передбачає насамперед необхідність зберігання даних в локальній пам'яті одного з вузлів цієї системи. Оскільки дані є необхідними для виконання обчислень або є їх результатом, зменшити обсяги пам'яті шляхом відмови від їх зберігання неможливо, за винятком випадків відміни обчислень, розглянутих у другому розділі. Однак, у час між готовністю даних до передачі в одному вузлі та їх безпосереднім використанням у іншому вузлі, існує можливість обрати один з них для зберігання цих даних, що в даній роботі виконувалось з огляду на обсяг пам'яті, зайнятої на кожному вузлі з ціллю його мінімізації. Зберігання занадто великих обсягів може непрямо впливати на ефективність виконання обчислень, а в деяких випадках навіть привести до зупинки обчислень через перевищення доступних обсягів пам'яті, так як ряд обчислювальних систем з локальною пам'яттю, зокрема великі кластерні системи, можуть не мати вузлів, обладнаних дисками, що використовуються для підкачки даних в основну пам'ять.

На рис. 1 зображено обсяги пам'яті, що в середньому були використані одним процесом за весь час роботи програми, усереднена для всіх програм, для яких виконувалось тестування, та згрупована за способами визначення тривалості відкладення передачі.

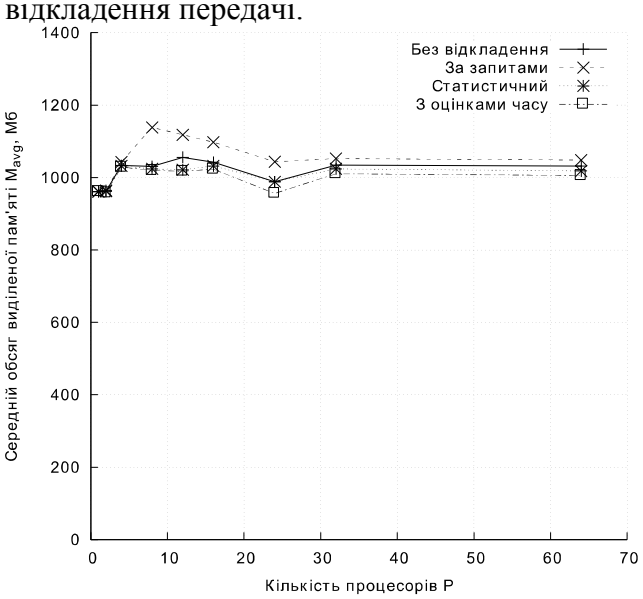

Рис. 1. Середній за час виконання обчислень та за всіма процесами обсяг виділеної пам'яті в перерахунку на один процес

Окремо зазначимо, що обсяг фізичної пам'яті на вузлах системи, яка використовувалась для тестування, складає 1024 Мб. на кожний наявний процесор. Програми виконання обчислень отримували цю інформацію від системи керування задачами на кластері, однак не враховували обсяги пам'яті, що використовуються системними процесами. Крім того, оскільки один вузол обчислювальної системи містить декілька процесорів, а обчислення могли виконуватись на одному або меншій кількості, існувала можливість використовувати всю пам'ять, наявну у вузлі без задіяння механізмів підкачки.

Відкладення передачі даних до запиту на їх використання прогнозовано збільшило середній обсяг використаної пам'яті у вузлах, оскільки дані, передачу яких відкладено, необхідно зберігати у джерелі. Слід відмітити, що при меншій кількості процесорів, обсяг зайнятої пам'яті суттєво збільшився, в деяких випадках збільшення склало до 13%. Це може бути пояснено більшими обсягами даних, що мають передаватися між парами вузлів, якщо їх невелика кількість, та меншою кількістю вузлів, які мог-

ли вже запитати дані на момент їх готовності. Загалом, використання підходу за запитами збільшує середні обсяги використаної пам'яті на 2 - 4%, однак в деяких випадках може зменшити пікові обсяги виділення пам'яті.

Використання підходу з відкладенням початку передачі на час, що визначається статистично дозволило незначно зменшити середній обсяг використаної пам'яті, на 1 - 2%, що викликано здебільшого наявністю правила про заборону відкладення при значній різниці в обсягах виділеної пам'яті. Використання підходу із застосуванням оцінок часу передачі даних та виконання дозволило зменшити обсяги використаної пам'яті до 5.3% при наявності балансування навантаження в системах з лінеарізуємими залежностями швидкостей передачі даних, та до 2.7% в системах з довільними залежностями швидкостей передачі даних.

Провал в обсягах використаної пам'яті для 24 процесорів, найбільш ймовірно пояснюється нерівномірністю розбиття на підзадачі, оскільки використовуваний алгоритм розділення на половини має продукувати кількість підзадач з приблизно однаковими обсягами даних, що є певним ступенем двійки.

За відсутності балансування навантаження у задачах, які не допускають відміну обчислень, середні обсяги використаної пам'яті залишилися незмінними.

Час очікування введення та виведення

Очікування початку передачі даних за мережею, на відміну від часу безпосередньої передачі даних, передбачає відсутність виконання обчислень через неготовність даних, що в свою чергу може суттєво вплинути на час виконання обчислень та на коефіцієнт ефективності.

На рис. 2 наведено залежність відношення часу очікування введення та виведення, до якого входить як час передачі даних мережею, так і робота з дисковою підсистемою, за її наявності на локальних вузлах, до загального часу виконання програми від кількості процесорів, що використовуються для обчислень. Тобто за віссю ординат знаходиться відсоток часу, який, в середньому для всіх процесів, займає час очікування введення та виведення від часу виконання обчислень. При цьому на графіку наведено усереднене для всіх задач та способів визначення часу передачі даних значення. Доцільно зображати саме відсоток, а не абсолютні значення, оскільки тривалість виконання обчислень сильно змінюється зі зміною кількості процесорів, причому різним чином для різних класів задач.

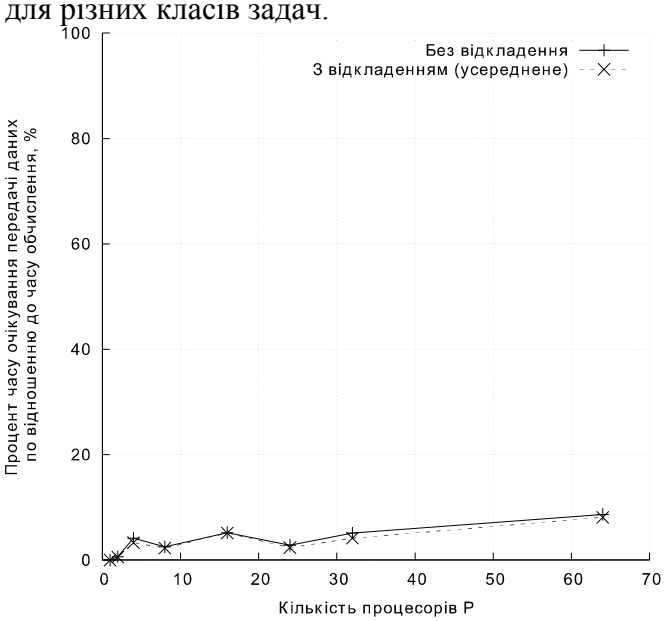

Рис. 2. Відношення часу очікування передачі даних до загального часу виконання програми, усереднене за всіма процесами

При невеликих кількостях процесорів, що використовуються для виконання обчислень, відсоток часу очікування передачі даних майже не змінився. Це пояснюється відносно великим часом виконання обчислень по відношенню до очікування. З іншого боку, час очікування передачі даних із застосуванням відкладення передачі даних зазнав зменшення до 43% по відношенню до цієї ж величини без відкладення, однак це не мало суттєвого впливу на ефективність виконання обчислень. Із збільшенням кількості процесорів, на яких виконуються обчислення, вплив на відсоток часу, що займає очікування введення та виведення, стає більш суттєвим завдяки двом факторам: по-перше, при великій кількості процесорів збільшується кількість пар вузлів, що комунікують один з одним, та потенційно створюють ситуації очікування, і по-друге, зменшується загальний час виконання обчислень і очікування має більш суттєве значення. Крім того, зростає час очікування введення та виведення через збільшення варіантів обрання пар процесів для передачі даних. При використанні 32 процесорів відсоток часу очікування введення та виведення по відношенню до загального часу виконання обчислень було зменшено з 5.1% до 4.1%. Очікується, що з подальшим збільшенням кількості процесорів за умови ба-

лансування навантаження і невпорядкованих передач, різниця в часі очікування лише збільшуватиметься.

Вплив на коефіцієнт ефективності

Покращення двох вищерозглянутих факторів: обсягів використаної у вузлах пам'яті та часу очікування введення та виведення даних – мали на меті вплинути на ефективність виконання обчислень, яка в свою чергу виражається як у швидкості виконання обчислень, так і у завантаженості апаратного забезпечення. Необхідно розглянути вплив на ефективність виконання обчислень окремо для різних класів задач, оскільки вони самі по собі демонструють значення коефіцієнту ефективності, що суттєво відрізняються, особливо при масштабуванні за кількістю процесорів, на яких виконуються обчислення. Тим не менш, необхідно згрупувати результати, отримані для різних обмежень на швидкості передачі даних між вузлами.

На рис. 3 зображено яким чином змінюється коефіцієнт ефективності в задачах, які обмінюються однаковими обсягами даних між обраними парами вузлів. Неважко помітити, що коефіцієнт ефективності незначно спадає при збільшенні кількості процесорів завдяки рівномірності розподілу обсягів даних, що дозволяє статистично їх зпрогнозувати. При невеликих кількостях процесорів, найбільш суттєве збільшення коефіцієнту ефективності досягається при використанні відкладення на статичний визначений час, при цьому із збільшенням кількості процесорів таке відкладення стає менш ефективним, зокрема для 64 процесорів його застосування взагалі стає недоцільним через зменшення коефіцієнту ефективності. Це має бути пояснене двома факторами: з одного боку, зі збільшенням кількості процесорів зменшується мінімальний розмір блоку даних, для яких виконується обчислення, тобто зменшується час виконання обчислень, а отже статично заданий час стає порівняно більшим; з іншого боку, більша ймовірність того, що один із процесорів надішле запит на використання даних за проміжок часу, на який була відкладена їх передача.

З іншого боку, застосування підходів, що базуються на зборі статистичної інформації про виконання обчислень та передачі даних демонструють незначне збільшення коефіцієнту ефективності в одних випадках та зменшення в інших. Зменшення переважно спостерігається при виконанні обчислень, в яких відбувається передача фіксованих обсягів даних між впорядкованими парами процесів, якщо канали передачі даних мають майже однакові характеристики. Разом з тим при використанні каналів передачі, що мають суттєво різні обмеження, застосування таких підходів дозволяє збільшити коефіцієнт ефективності.

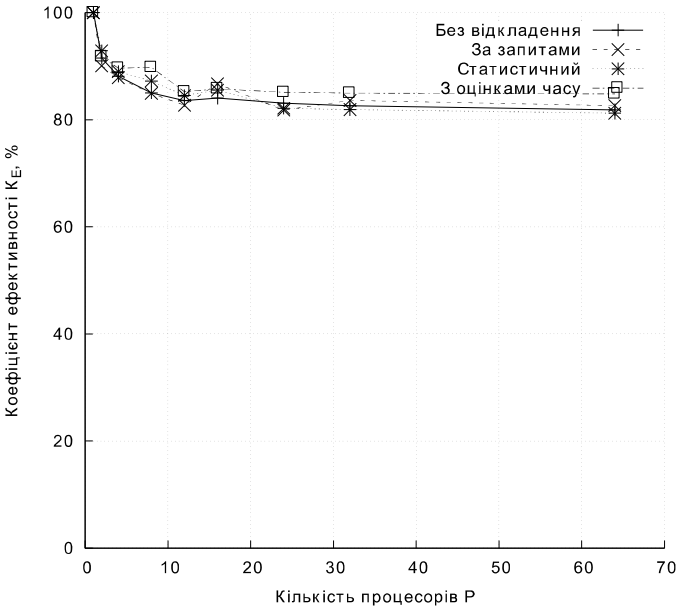

Рис. 3. Залежність коефіцієнту ефективності від кількості процесорів для обчислень з однаковими обсягами даних, що передаються між вузлами

Застосування підходу з оцінками часу майбутньої передачі даних дозволяє збільшити коефіцієнт ефективності на 4 - 6%, оскільки ймовірнісна модель, що використовується для оцінки легко адаптується до рівномірних обсягів та типів даних, що передаються, тому оцінки часу передачі, виконані з використанням цієї моделі, близькі до реальної тривалості передачі даних.

На рис. 4 показано зміну коефіцієнту ефективності при виконанні обчислень на різній кількості процесорів у випадку задач, в яких обсяг обчислень або обсяги даних, що передаються між вузлами, можуть поліноміально залежати від номерів вузлів. Результати згруповані за підходами, що використовувалися для обчислення тривалості відкладення передачі даних, та усереднені за усіма задачами, що входять до даного класу. Слід зазначити, що експерименти виконувались для задачі чисельного інтегрування, для якої обсяги обчислень змінюються лінійно на всьому проміжку, та для чисельного розв'язання диференційного рівняння в двовимірному просторі, в якому обсяги даних, що передаються, змінюються квад-

ратично від центру області розв'язання. Програмна модель, з іншого боку, дозволила моделювати виконання обчислень з поліноміальними залежностями до п'ятого ступеню, як описано в попередньому розділі.

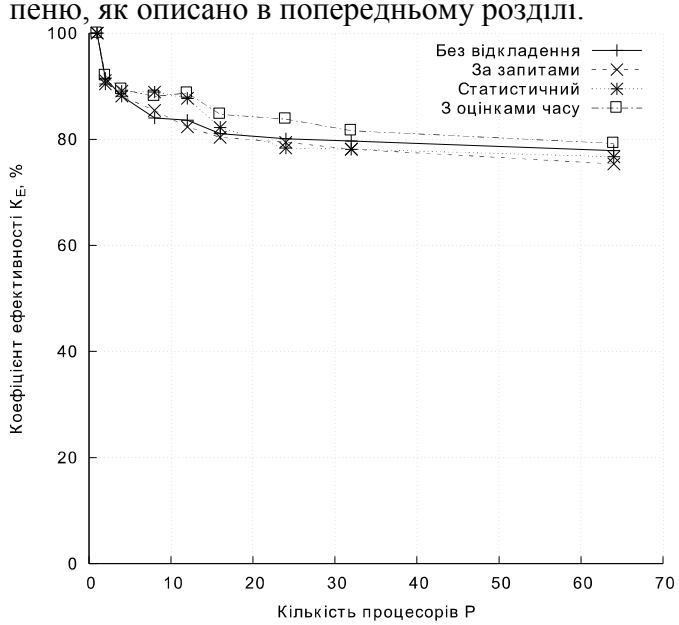

Рис. 4. Залежність коефіцієнту ефективності від кількості процесорів для обчислень з поліноміальною залежністю обсягів даних від номеру процесу

Для невеликої кількості процесорів, до шістнадцяти, усі підходи до обчислення тривалості відкладення початку передачі даних демонструють приблизно однакове збільшення коефіцієнту ефективності. Найбільш істотне збільшення, з 83.5% до 88.7% зафіксовано при використанні 12 процесорів. Підхід до визначення тривалості відкладення за запитами зі збільшенням кількості процесорів призводить до зменшення коефіцієнту ефективності, навіть у порівнянні з відсутністю відкладення передачі даних через збільшення часу очікування передачі. Тим не менш, при невеликій кількості процесорів через більш суттєву нерівномірність розподілу обсягів даних, що передаються, цей підхід дозволяє збільшити коефіцієнт ефективності на 0.2% завдяки перерозподілу обсягів виділеної пам'яті.

Статистичний підхід при великій кількості процесорів може також призвести до незначного зменшення коефіцієнту ефективності, так як передачі даних можуть виконуватися між більшою кількістю пар процесорів що вимагає більш тривалого часу збирання статистики для визначення часу передачі, тому статистичні оцінки можуть бути невірними. Також зі збільшенням кількості процесорів зменшується перевага від використання відкладених передач даних у відносних одиницях, що не перешкоджає однак подальшому зменшенню часу виконання обчислень.

Зміна коефіцієнту ефективності, усередненого за всіма задачами, для яких виконувались експерименти, та згрупована за підходами до визначення тривалості відкладення передачі даних, в залежності від кількості процесорів представлена на рис. 5. Слід відмітити більш різкий спад ефективності при нарощуванні кількості процесорів через явну нерівномірність обсягів обчислень або даних, що передаються в цьому випадку.

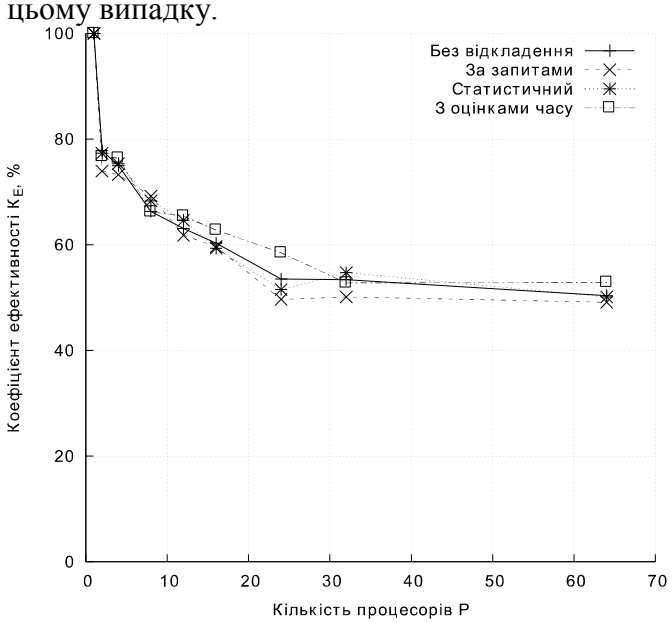

Рис. 5. Залежність коефіцієнту ефективності від кількості процесорів для обчислень, в яких обсяги даних не залежать від номеру процесу

Застосування підходу визначення тривалості відкладення передачі даних за запитами майже в усіх випадках призвела до зменшення коефіцієнту ефективності, яке не було нівельоване ефектами від зміни обсягів пам'яті через випадковий вплив на них в процесі обчислень. Для невеликої кількості процесорів, більшість методів дозволяють отримати коефіцієнти ефективності в межах ±1% у порівнянні до відсутності відкладення, що дозволяє використовувати запропоновані підходи навіть у цьому випадку без суттєвих втрат ефективності.

Використання підходу з оцінками часу при наявності 8 або менше процесорів, здебільшого призвело до зменшення коефіцієнту ефективності, що пояснюється неадаптованістю моделі оцінки до випадкових обсягів даних, які пере-

силаються, тому ймовірнісна модель оцінки могла зафіксувати неіснуючу залежність між обсягами даних. Однак зі збільшенням кількості процесорів, завдяки збільшенню кількості факторів, що впливають на процес обчислень загалом, оцінка часу передачі даних наблизилась до нормального розподілу, що дозволило підходу з оцінками часу демонструвати збільшення ефективності до 4.6%.

Застосування статистичного підходу без оцінок в деяких випадках дозволило збільшити коефіцієнт ефективності до 1.3%, що однак не є загальною тенденцією для всіх обчислювальних задач, для яких проводились експерименти.

Висновки

Запропоновані підходи до відкладання початку передачі даних між вузлами обчислювальної системи на певний час з огляду на збільшення ефективності та оптимізацію використання пам'яті можуть бути реалізовані як додатковий рівень абстракції над низькорівневим інтерфейсом передачі повідомлень MPI, оскільки останній являє собою інтерфейс для організації взаємодії між вузлами в більшості сучасних кластерних систем. Крім того, це дозволить використовувати існуючі оптимізовані реалізації MPI та дозволить користувачеві системи описувати спосіб передачі даних із використанням відомих засобів. При цьому також доцільно реалізувати одну із абстракцій більш високого рівня над передачею повідомлень, зокрема абстракцію розбиття і об'єднання підзадач, яка дозволить у більш простому вигляді описувати структуру задач для подальших експериментів.

Реалізація для сучасних обчислювальних систем з локальною пам'яттю має ряд особливостей, зокрема необхідність урахувати під час реалізації наявність в системі процесів, що виконують функції операційної системи. Також можна урахувати оптимізації реалізацій низькорівневого інтерфейсу передачі повідомлень для випадків, коли передача повідомлень здійснюється між процесами, які фактично мають доступ до спільної пам'яті. В цьому випадку недоцільно виконувати відкладення, оскільки безпосередньої передачі даних не відбувається, а лише надається доступ до певної частини пам'яті для іншого процесу на тому ж самому вузлі.

Для реалізації передачі даних за запитом запропоновано передавати короткі дескриптивні структури даних, в яких міститься інформація про реальне місцезнаходження даних, їх структуру ту та обсяг. При цьому, у випадку переміщення процесу обчислень на інший вузол, необхідно перемістити лише цей дескриптор, а безпосередньо дані можуть переміщені пізніше. Додатково, у моделях, що підтримують розбиття задачі за даними, наприклад у задачах масової обробки даних, існує можливість алгоритмічно визначати нові дескриптори без звернення безпосередньо до даних, якщо це допускається обчислювальною задачею.

Щоб отримати статистику виконання обчислень на реальній системі, використовується ряд інтерфейсів доступу до апаратних лічильників продуктивності, а також інструментування викликів інтерфейсу передачі повідомлень. Дані про виконання обчислень збираються безпосередньо під час їх виконання та зберігаються у файл трасування для подальшої обробки, оскільки аналіз під час виконання обчислень може вплинути на вимірювання через необхідність проведення складних обчислень.

Список посилань

1. Pheatt, Chuck. Intel ® threading building blocks / Chuck Pheatt //Journal of Computing Sciences in Colleges.– 2008.– Vol. 23, no. 4.– P. 298–298.
2. Стіренко С.Г. Метод перебалансування навантаження в кластерних комп'ютерних системах в динамічному режимі // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2014. –№. 60. – С. 55-59.
3. Стіренко С.Г. Модель технології програмування паралельних систем // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.58. – С.158-164.
4. Стіренко С.Г. Організація відкладених обчислень в кластерних системах // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.59. – С.12-16.
5. Hennessy, John L. Computer architecture: a quantitative approach / John L Hennessy, David A Patterson. – Amsterdam, Noord-Hooland, Netherlands: Elsevier, 2011. – 824 p.