УДК 004.032.26; 004.272.86

ТУРЧЕНКО В.О.

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ ГРУПОВОГО НАВЧАННЯ БАГАТОШАРОВОГО ПЕРСЕПТРОНУ НА ПАРАЛЕЛЬНОМУ КОМП'ЮТЕРІ ТА ОБЧИСЛЮВАЛЬНОМУ КЛАСТЕРІ

Паралельний метод групового навчання багатошарового персептрону (БШП) на основі алгоритму зворотного поширення помилки та дослідження ефективності розпаралелення цього методу на паралельному комп'ютері та обчислювальних кластерах представлено в цій статті. Модель БШП та послідовний метод групового навчання описані теоретично. Представлено алгоритмічний опис паралельного методу групового навчання. Ефективність розпаралелення методу досліджена на поступово збільшуваних розмірностях сценаріїв навчання. Результати експериментальних досліджень показали, що (i) обчислювальний кластер з комунікаційним інтерфейсом Infiniband показав кращу ефективність розпаралелення, ніж паралельний комп'ютер загального призначення з ccNuma архітектурою через менші комунікаційні втрати та (ii) ефективність розпаралелення представленого методу є достатньо високою для його успішного застосування на паралельних комп'ютерах та кластерах загального призначення, наявних в сучасних обчислювальних ГРІД-системах.

The development of a parallel method for batch pattern training of a multilayer perceptron with the back propagation training algorithm and the research of its efficiency on general-purpose parallel computer and computational clusters are presented in this paper. The model of a multilayer perceptron and the usual sequential batch pattern training method are theoretically described. An algorithmic description of the parallel version of the batch pattern training method is presented. The efficiency of parallelization of the developed method is investigated on the progressive increasing the dimension of the parallelized problem. The results of the experimental researches show that (i) the computational cluster with Infiniband interconnection shows better values of parallelization efficiency in comparison with general-purpose parallel computer with ccNuma architecture due to lower communication overhead and (ii) the parallelization efficiency of the method is high enough for its appropriate usage on general-purpose parallel computers and clusters available within modern computational grids.

1. Вступ

Штучні нейронні мережі (НМ) є чудовим механізмом для моделювання складних нелінійних систем. Вони є дуже доброю альтернативою традиційним методам рішення складних задач в багатьох сферах включаючи обробку зображень, розпізнавання образів, робототехніку, оптимізацію, моделювання процесів життєдіяльності та інших [1]. В загальному випадку, НМ складається з двох і більше шарів (рівнів) елементарних нейронів з'єднаних синапсами (ваговими коефіцієнтами). Кожен нейрон поточного шару отримує інформацію від нейронів попереднього шару, виконує операцію зваженої суми, використовує її як аргумент для обчислення функції активації та передає результат до нейронів наступного шару. Хоча обчислення в кожному нейроні та НМ загалом є дуже прості, загальне обчислювальне навантаження може бути значним, особливо на етапі навчання НМ (години та дні). Це, без сумніву, є однією з головних перепон до ефективного використання НМ для вирішення прикладних задач. Використання високопродуктивних паралельних комп'ютерів, суперкомп'ютерів, кластерів та обчислювальних ГРІД-систем загального призначення для прискорення етапу навчання НМ є шляхом усунення цієї перепони. Тому розробка паралельних методів навчання НМ та дослідження їх ефективності розпаралелення на таких класах паралельних комп'ютерних систем все ще залишається актуальною науковою проблемою.

Беручи до уваги паралельну природу штучних НМ, багато дослідників досліджували методи їх розпаралелення на спеціалізованому апаратному забезпеченні та трансп'ютерах [2-5], однак ці рішення вимагають наявності згаданих пристроїв для використання широкою науковою спільнотою. В той же час, високопродуктивні паралельні комп'ютери та обчислювальні ГРІД-системи загального призначення широко застосовуються зараз для наукових експериментів та моделювання у віддаленому режимі. Декілька ГРІД-базованих підходів до реалізації НМ представлені в [6-7], однак питання ефективності розпаралелення процесів навчання в них не досліджуються.

Паралельний алгоритм навчання БШП на багатопроцесорному комп'ютері, кластері та обчислювальній ГРІД-системі з використанням бібліотеки MPI (Message Passing Interface) досліджено у [8]. Дослідження проведено на великих моделях НМ, які обробляють велику кількість навчальних векторів (близько 20000), що поступають з Великого Андронного Коллайдера. Однак реалізація розпаралелення відносно «малої» архітектури БШП 16-10-10-1 (16 нейронів у вхідному шарі, два схованих шари по 10 нейронів у кожному та один вихідний нейрон) з 270 внутрішніми налаштовуваними зв'язками (загальна кількість вагових коефіцієнтів та порогів НМ) не забезпечує прискорення розпаралелення через значні втрати часу на передачу повідомлень між паралельними гілками алгоритму, тобто прискорення є меншим, ніж 1 (час виконання паралельної версії програми є більшим від часу виконання послідовної програми, що реалізує той самий алгоритм).

Разом с тим, малі моделі НМ з кількістю зв'язків менше ніж 270, широко використовуються для вирішення практичних задач завдяки кращим узагальнюючим властивостям. В цих моделях кількість нейронів схованих шарів, як правило, є значно меншою від кількості навчальних векторів [1]. Тому розпаралелення малих моделей НМ, що навчаються на не дуже великій кількості вхідних даних, але можуть вимагати великої кількості епох навчання, також є важливою задачею.

В роботах [9-10] автором цієї статті розроблено паралельний метод групового навчання БШП за алгоритмом зворотного поширення помилки та досліджена ефективність його розпаралелення на паралельному комп'ютері загального призначення. Показано, що деталі реалізації паралельного алгоритму насправді відіграють важливу роль. Наприклад, останні версії програмного забезпечення MPI, що мають покращені властивості колективних комунікацій (покращений модуль реалізації колективних функцій пакету Open MPI), показують зменшення комунікаційних втрат, що веде до відповідного підвищення ефективності розпаралелення паралельного алгоритму на обчислювальному кластері [11].

Метою цієї статті є оцінка ефективності розпаралелення розробленого паралельного методу групового навчання за алгоритмом зворотного поширення помилки для БШП на обчислювальних кластерах з комунікаційними інтерфейсами Infiniband та Gigabit Ethernet, та її порівняння з ефективністю розпаралелення на паралельному комп'ютері загального призначення.

2. Метод групового навчання БШП за алгоритмом зворотного поширення помилки

Розпаралелення БШП з стандартним послідовним алгоритмом зворотного поширення помилки не є ефективним на паралельному комп'ютері загального призначення (прискорення менше одиниці) через високі втрати часу на синхронізацію та передачу повідомлень між паралельними процесорами [12]. Тому доцільно використати метод групового навчання, згідно з яким модифікація вагових коефіцієнтів та порогів нейронів здійснюється в кінці кожної епохи навчання, тобто після обробки всіх навчальних векторів з вхідної навчальної вибірки замість модифікації вагових коефіцієнтів та порогів нейронів після обробки кожного навчального вектору в звичайному послідовному режимі навчання.

Вихідне значення тришарового персептрону (рис. 1) визначається згідно з виразом:

$$y = F_3\left(\sum_{j=1}^{N} w_{j3}\left(F_2\left(\sum_{i=1}^{M} w_{ij}x_i - T_j\right)\right) - T\right), \qquad (1)$$

де N – кількість нейронів у схованому шарі, w_{j3} – ваговий коефіцієнт від j-го нейрону схованого шару до вихідного нейрону, w_{ij} – вагові коефіцієнти від i-го нейрону вхідного шару до j-го нейрону схованого шару, x_i – елементи вхідного навчального вектору, T_j – пороги нейронів схованого шару та T – поріг вихідного нейрону [1, 13]. В цій моделі використано стандартну сигмоїдну функцію активації $F(x) = 1/(1+e^{-x})$ для нейронів схованого (F_2) та вихідного (F_3) шарів, однак в загальному випадку ці функції можуть бути різними.

Метод групового навчання за алгоритмом зворотного поширення помилки складається з наступних кроків [13]:

1. Встановити сумарну квадратичну помилку (Sum Squared Error) в необхідне мінімальне значення SSE=$E_{\min}$ та кількість епох навчання t;

2. Проініціалізувати пороги та вагові коефіцієнти нейронів значеннями з відрізку (0…0.5) [13];

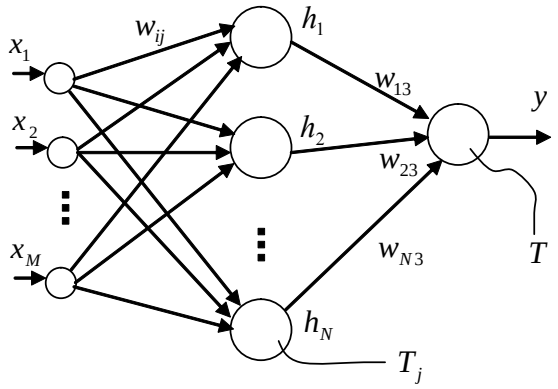


Рис. 1. Структура тришарового персептрону

3. Для вхідного навчального вектору pt виконати наступну навчальну ітерацію:
   3.1. Обчислити вихідне значення БШП $y^{pt}(t)$ згідно з виразом (1);
   3.2. Обчислити помилку вихідного нейрону $\gamma_3^{pt}(t) = y^{pt}(t) - d^{pt}(t)$, де $y^{pt}(t)$ є обчисленим вихідним значенням БШП, а $d^{pt}(t)$ є бажаним вихідним значенням (ціль навчання);
   3.3. Обчислити помилку нейронів схованого шару $\gamma_j^{pt}(t) = \gamma_3^{pt}(t) \cdot w_{j3}(t) \cdot F_3'(S^{pt}(t))$, де $S^{pt}(t)$ - зважена сума вихідного нейрону;
   3.4. Обчислити зміни вагових коефіцієнтів та порогів всіх нейронів та додати їх до відповідних значень змін, отриманих для попередніх навчальних векторів
   $s\Delta w_{j3} = s\Delta w_{j3} + \gamma_3^{pt}(t) \cdot F_3'(S^{pt}(t)) \cdot h_j^{pt}(t)$,
   $s\Delta T = s\Delta T + \gamma_3^{pt}(t) \cdot F_3'(S^{pt}(t))$,
   $s\Delta w_{ij} = s\Delta w_{ij} + \gamma_j^{pt}(t) \cdot F_2'(S_j^{pt}(t)) \cdot x_i^{pt}(t)$,
   $s\Delta T_j = s\Delta T_j + \gamma_j^{pt}(t) \cdot F_2'(S_j^{pt}(t))$, де $S_j^{pt}(t)$ та $h_j^{pt}(t)$ - зважена сума та вихідне значення j-го нейрону схованого шару відповідно;
   3.5. Обчислити SSE використовуючи вираз $E^{pt}(t) = \frac{1}{2}\left(y^{pt}(t) - d^{pt}(t)\right)^2$;
4. Повторити крок 3 вище для кожного навчального вектора pt, де $pt \in \{1,..., PT\}$, PT – розмір усієї вхідної навчальної вибірки;
5. Обчислити вагові коефіцієнти та пороги нейронів використовуючи вирази:
   $w_{j3}(PT) = w_{j3}(0) - \alpha(t) \cdot s\Delta w_{j3}$,
   $T(PT) = T(0) + \alpha(t) \cdot s\Delta T$,
   $w_{ij}(PT) = w_{ij}(0) - \alpha(t) \cdot s\Delta w_{ij}$,
   $T_j(PT) = T_j(0) + \alpha(t) \cdot s\Delta T_j$ де $\alpha(t)$ – крок навчання;
6. Обчислити загальну SSE $E(t)$ для навчальної епохи t використовуючи вираз $E(t) = \sum_{pt=1}^{PT} E^{pt}(t)$;

Якщо значення $E(t)$ більше, ніж бажана мінімальна помилка $E_{\min}$, то збільшити номер навчальної епохи до $t+1$ та знову перейти до кроку 3, в іншому випадку – завершити процес навчання.

3. Паралельний метод групового навчання бшп за алгоритмом зворотного поширення помилки

Аналіз методу групового навчання в розділі 2 вище показує, що послідовне виконання кроків 3.1-3.5 для всіх навчальних векторів у вибірці навчання може бути розпаралелено, тому що операції сумування значень $s\Delta w_{j3}$, $s\Delta T$, $s\Delta w_{ij}$ та $s\Delta T_j$ є незалежними одна від одної. Для розробки паралельного методу необхідно розділити весь об'єм обчислень між головним процесором *Master* (буде виконувати функції призначення частин програмного коду робочим паралельним процесорам та власні обчислення) та робочими процесорами *Workers* (будуть виконувати частини програмного коду паралельно).

Алгоритми процесорів *Master* та *Worker* наведено на рис. 2. *Master* починає функціонування з визначення (i) кількості векторів *PT* у вхідній навчальній вибірці та (ii) кількості процесорів *p*, на яких завантажено паралельну програму на виконання. *Master* розділяє всі навчальні вектори на рівні частини відповідно до кількості робочих процесорів *Workers,* а також призначає одну частину обчислювальної роботи для себе. Після цього *Master* посилає робочим процесорам *Workers* номери визначених навчальних векторів для здійснення навчання.

Кожен робочий процесор *Worker* виконує наступні операції для кожного патерну *pt* зі всіх *PT/p* патернів, що були йому призначені:

- Виконати кроки 3.1-3.5 та 4. Значення часткових сум вагових коефіцієнтів $s\Delta w_{j3}$, $s\Delta w_{ij}$

та порогів $s\Delta T$, $s\Delta T_j$ обчислюються на цьому кроці;

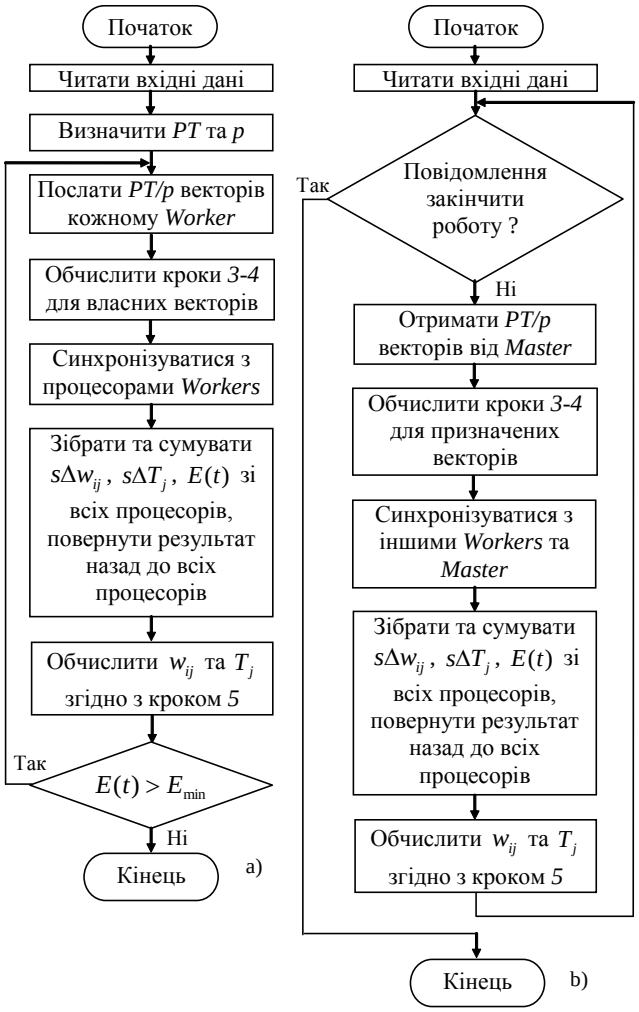


Рис. 2. Алгоритми головного та робочих процесорів Master (a) та Worker (b)

- Обчислити часткові значення SSE для призначених навчальних векторів.

Після обробки всіх призначених векторів кожним процесором виконується тільки одна функція колективної комунікації (вона також здійснює і сумування). Синхронізація з іншими процесорами автоматично забезпечується внутрішньою реалізацією цієї функції колективної комунікації [14]. Однак з алгоритмічної точки зору операція синхронізації показана окремо на рис. 2 перед операцією збору даних. Після збору даних сумарні значення $s\Delta w_{j3}$, $s\Delta T$, $s\Delta w_{ij}$ та $s\Delta T_j$ посилаються до всіх процесорів, що працюють паралельно. Використання тільки однієї функції колективної комунікації, що повертає всі зібрані значення з паралельно працюючих процесорів *Workers* назад до всіх процесорів у групі, дозволяє зменшити комунікаційні втрати паралельного алгоритму. Після отримання всі сумарні значення $s\Delta w_{j3}$, $s\Delta T$, $s\Delta w_{ij}$ та $s\Delta T_j$ поміщаються в локальну пам'ять кожного процесора. Кожен процесор використовує ці сумарні значення для обчислення нових значень вагових коефіцієнтів та порогів згідно з кроком 5 методу. Потім нові значення вагових коефіцієнтів та порогів використовуються на наступній епосі методу навчання. Так як сумарне значення $E(t)$ також отримується в результаті операції збору даних, то головний процесор *Master* вирішує – продовжувати навчання чи ні.

Програмний код розроблено на мові програмування C з використанням стандартних MPI-функцій. Паралельна частина алгоритму починається викликом функції *MPI_Init()*. Функція *MPI_Allreduce()* здійснює (i) збір сумарних по кожному процесору часткових вагових коефіцієнтів $s\Delta w_{j3}$, $s\Delta w_{ij}$ та порогів $s\Delta T$, $s\Delta T_j$, (ii) їх подальше сумування та (iii) відсилку кінцевих (просумованих по всіх векторах та процесорах) значень вагових коефіцієнтів та порогів до всіх інших процесорів, що працюють паралельно в групі. Так як вагові коефіцієнти та пороги фізично розміщуються в сегменті коду програми в різних змінних/матрицях, то з метою використання тільки одного виклику функції *MPI_Allreduce()* доцільно «запакувати» ці різні змінні/матриці в один вектор даних перед відсилкою даних, а на приймачі здійснити обернене «розпакування» отриманих даних у відповідні змінні/матриці. Використання тільки одного фізичного виклику функції *MPI_Allreduce()* дозволяє додатково зменшити комунікаційні втрати паралельного алгоритму. Функція *MPI_Finalize()* завершує паралельний алгоритм.

4. Результати експериментальних досліджень

Експериментальні дослідження розробленого паралельного методу здійснені на трьох паралельних обчислювальних системах загального призначення:

- обчислювальний кластер *Reef*, розміщений у Суперкомп'ютерному та мережевому центрі м. Познань, Польща. Кластер складається з 285-ти обчислювальних вузлів. Для проведення експериментальних досліджень використано групу з 32-х обчислювальних ядер, що розміщуються в чотирьох обчислювальних вузлах. Кожен вузол вміщує два чотири-

ядерних процесора Intel Xeon E5345 з тактовою частотою 2.33 ГГц. Вузли з'єднані між собою інтерфейсами Gigabit Ethernet, Fast Ethernet та Infiniband. Експерименти проведено в режимі, коли для з'єднання між вузлами використано тільки інтерфейс Infiniband. Бібліотека Open MPI 1.2.8 використана для проведення експерименту. В подальшому викладенні цей кластер позначено ідентифікатором "Cluster Infiniband";

- обчислювальний кластер *Battlecat*, розміщений в Лабораторії інноваційних обчислень (ICL) університету шт. Теннессі, США. Кластер складається з одного головного вузла та 7-ми робочих вузлів. Головний вузол складається з двох процесорів Dual Xeon 1.6 ГГц з 4 Мб кеш-пам'яті та 2 Гб локальної пам'яті. Кожен робочий вузол має один двох-ядерний процесор Intel Core 2 Duo 2.13 ГГц з 2 Мб кеш-пам'яті та 2 Гб локальної пам'яті. Вузли з'єднані між собою інтерфейсом Gigabit Ethernet. Бібліотека Open MPI 1.4.1 та 16 обчислювальних ядер використано для проведення експерименту. Цей кластер є важливим для проведення експериментальних досліджень, так як в режимі використання 8-ми та 16-ти ядер кластер має гетерогенну архітектуру як з точки зору обчислювальної потужності ядер, так і з точки зору швидкості передачі даних між ядрами. В подальшому викладенні цей кластер позначено ідентифікаторами "Cluster Gigabit Ethernet" або "CGE";
- паралельний комп'ютер NEC TX7 *Crati*, розташований в Суперкомп'ютерному центрі обчислювальної інженерії, Калабрійський університет, Італія. Комп'ютер складається з 28 одноядерних 64-х розрядних процесорів Intel Itanium 2 з тактовою частотою 1 ГГц та загальною пам'яттю 64 Гб. Кожен процесор має кеш другого рівня розміром 3 Мб. Комп'ютер має архітектуру ccNuma (неоднаковий доступ процесорів до загальної пам'яті з когерентною кеш-пам'яттю) та міжпроцесорний інтерфейс ccNuma link. Бібліотека NEC MPI/EX 1.5.2 та 28 процесорів використано для проведення експерименту. В подальшому викладенні цей комп'ютер позначено ідентифікатором "Computer ccNuma" або "CccNUMA".

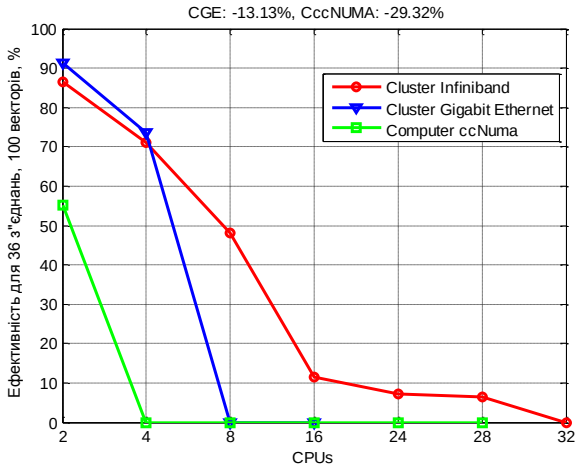
Для проведення експериментальних досліджень використано 8 сценаріїв збільшення розміру БШП та кількості навчальних векторів: БШП 5-5-1 (5 нейронів вхідного шару * 5 нейронів схованого шару = *25 зв'язків* – вагових коефіцієнтів, 5 нейронів схованого шару * 1 вихідний нейрон = ще *5 зв'язків* – вагових коефіцієнтів, всього є 6 (5 схованих та 1 вихідний) нейронів, що здійснюють обробку інформації, – кожен з має поріг = *6 зв'язків* – порогів. Всього в БШП 5-5-1 є *36 налаштовуваних зв'язків*, що змінюють свої значення в процесі його навчання) та 100 навчальних векторів, БШП 10-10-1 (*121 налаштовуваний зв'язок*) та 200 навчальних векторів, БШП 15-15-1 (*256 налаштовуваних зв'язків*) та 400 навчальних векторів, БШП 20-20-1 (*441 налаштовуваний зв'язок*) та 600 навчальних векторів, БШП 30-30-1 (*961 налаштовуваний зв'язок*) та 800 навчальних векторів, БШП 40-40-1 (*1681 налаштовуваний зв'язок*) та 1000 навчальних векторів, БШП 50-50-1 (*2601 налаштовуваний зв'язок*) та 5000 навчальних векторів та БШП 60-60-1 (*3721 налаштовуваний зв'язок*) та 10000 навчальних векторів. Ці сценарії збільшення розміру БШП та кількості навчальних векторів вибрані з метою дослідження ефективності розпаралелення методу як на малих, так і на великих розмірностях обчислювальної задачі. Нейрони схованого та вихідного шарів мали сигмоїдну функцію активації. Кількість епох навчання складала 10^4. Крок навчання був сталий і дорівнював $\alpha(t) = 0.01$. Вирази $S=Ts/Tp$ та $E=S/p\times100\%$ використано для знаходження прискорення S та ефективності розпаралелення E, де Ts – час виконання послідовної програми, Tp – час виконання паралельної версії тієї ж самої програми на p процесорах паралельної обчислювальної системи.

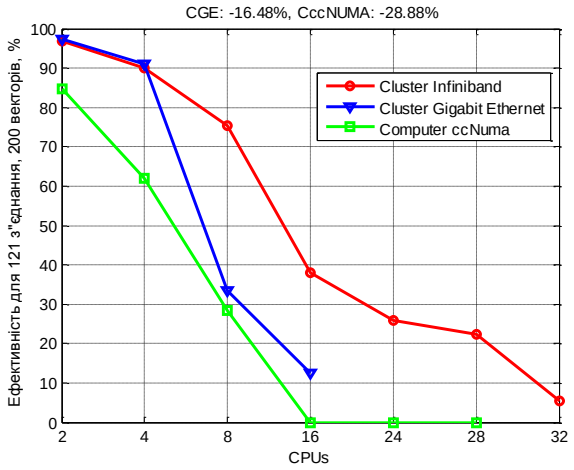
Значення ефективності розпаралелення методу групового навчання БШП за алгоритмом зворотного поширення помилки наведено на рис. 3 для всіх сценаріїв розпаралелення на всіх паралельних обчислювальних системах. Як видно, кращі результати розпаралелення отримані на системі Cluster Infiniband у порівнянні з системами CGE та CccNUMA для всіх сценаріїв, при цьому значення ефективності розпаралелення є приблизно однаковими на двох останніх системах. Тому обчислено усереднені спади ефективності розпаралелення для цих двох систем на тій самій кількості процесорів для кожного сценарію у порівнянні з Cluster Infiniband. Ці спади показані вгорі відповідного результату на рис. 3. Наприклад, для сценарію 36 зв'язків, 100 векторів система Cluster Infiniband переважає Cluster Gigabit Ethernet на 13.13%, а

Computer ccNuma на 29.32% ефективності розпаралелення в середньому.

Аналіз рис. 3 показує, що ефективність розпаралелення підвищується при збільшенні кількості зв'язків в БШП та збільшенні кількості вхідних навчальних векторів через збільшення складності обчислювальної частини алгоритму. Однак ефективність розпаралелення зменшується для того самого сценарію при збільшенні кількості паралельних процесорів через збільшення комунікаційних втрат. Це підтверджується шляхом оцінки часу обчислення та комунікації для 4 найменших сценаріїв на рис. 4. В кожній з 5-ти груп стовпчиків, три стовпчики для кожного сценарію описують обчислювальну (зелений колір) та комунікаційну (жовтий колір) частини алгоритму на відповідних паралельних системах – зліва-направо – Cluster Infiniband, CGE та CccNUMA на 1, 2, 4, 8 та 16 процесорах відповідно. Як видно, система Cluster Infiniband (ліва колонка в кожній групі) показує найменші комунікаційні втрати для всіх сценаріїв, тому вона забезпечує найкращу ефективність розпаралелення.

Отримані значення ефективності розпаралелення є достатньо високі, зокрема навіть для «малих» сценаріїв з 36, 121 та 256 зв'язками система Cluster Infiniband показує прискорення розпаралелення на 16-ти та більше процесорах. Однак з метою підвищення ефективності доцільно малі сценарії задачі розпаралелювати на малій кількості процесорів, тому що час навчання таких сценаріїв не є значним. Аналіз абсолютної тривалості часу виконання на рис. 4 показує, що паралельний комп'ютер CccNuma є швидшим, ніж кластери незважаючи на (i) нижчу тактову частоту процесорів та (ii) нижчі значення ефективності розпаралелення. Цей факт пояснюється внутрішньою оптимізацією програмного коду, реалізованою розробниками комп'ютера NEC TX-7. Результати експериментальних досліджень показали, що у випадку використання обчислювальної ГРІД-системи для ефективного мапування (призначення) процесу паралельного навчання НМ доцільно враховувати не тільки ефективність розпаралелення, але й час виконання задачі також. Це може бути забезпечено шляхом використання брокера ресурсів [15-16], що зараз розробляється в рамках бібліотеки для паралельного навчання НМ PaGaLiNNeT. Цей брокер дозволяє вибрати конкретну паралельну систему з конкретною кількістю паралельних процесорів (серед наявних в ГРІД), що можуть забезпечити певний консенсус між мінімізацією часу навчання НМ та максимізацією ефективності розпаралелення для конкретного сценарію внутрішніх налаштовуваних зв'язків НМ та кількості вхідних навчальних векторів. Наприклад, якщо сценарій БШП 20-20-1 (441 зв'язок) та 600 навчальних векторів поступає на розпаралелення, брокер може вибрати 8 процесорів кластеру Cluster Infiniband (ефективність розпаралелення 92.7%, час обчислення – 6.48 сек.) або 4 процесори паралельного комп'ютера CccNuma (ефективність розпаралелення 92.4%, час обчислення – 5.94 сек.). Такий механізм забезпечує високу ефективність розпаралелення любого сценарію вхідної задачі навчання НМ, так само як і економне використання ГРІД-інфраструктури.





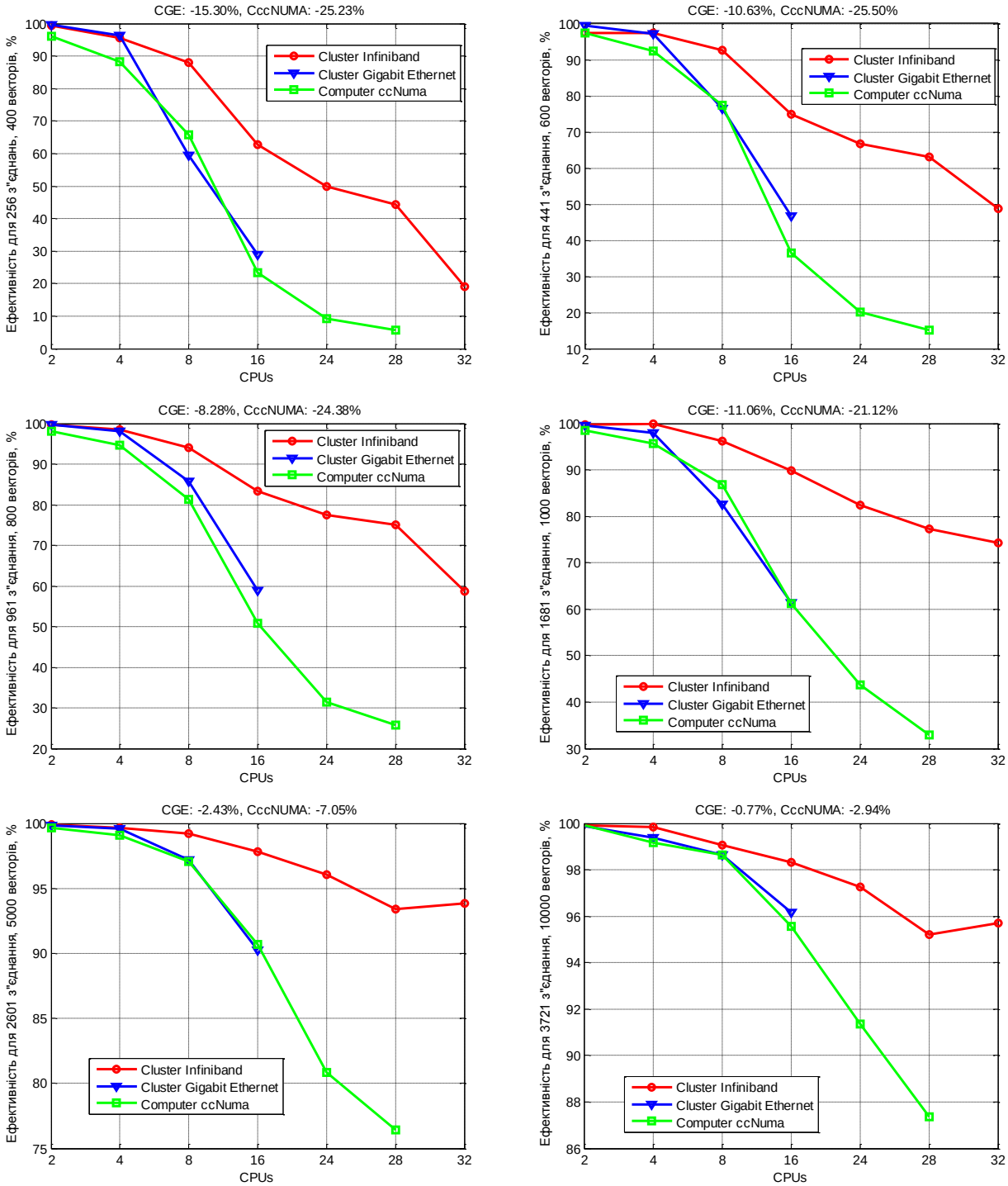
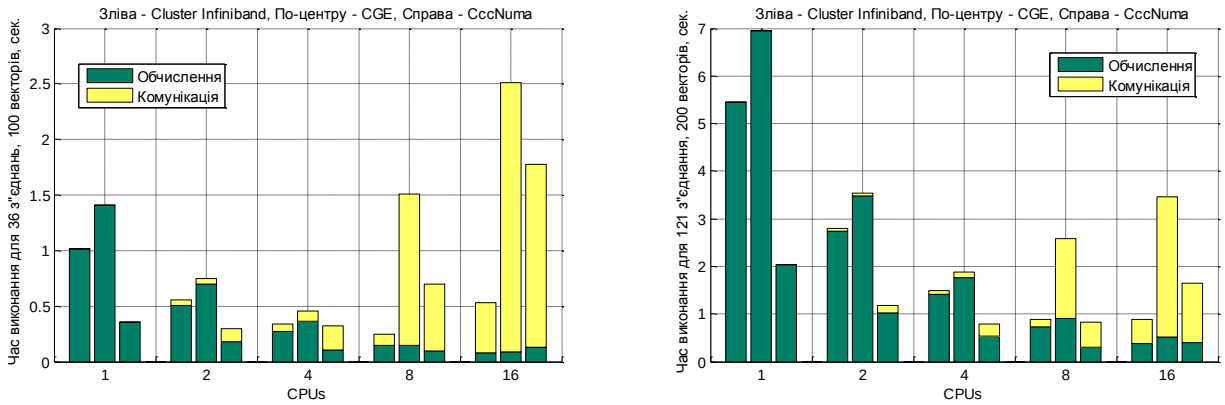


Рис. 3. Ефективність розпаралелення на різних обчислювальних системах



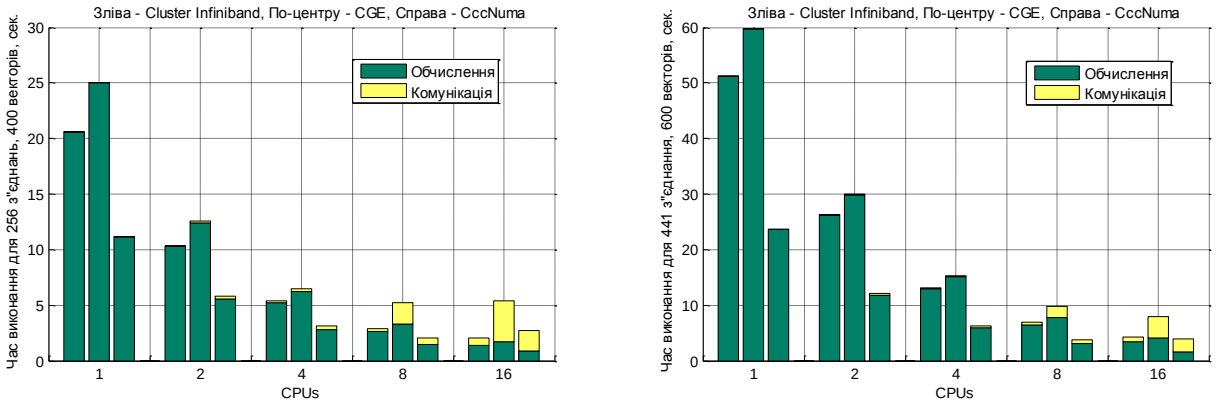


Рис. 4. Аналіз часу виконання обчислювальної та комунікаційної частин алгоритму на різних паралельних системах

Доцільно зазначити, що отримані малі значення часу навчання НМ, особливо для малих сценаріїв, показані в цій статті тільки з метою проведення експериментальних досліджень ефективності розпаралелення розробленого методу. Під час досліджень використано відносну малу кількість епох навчання – 10^4 з урахуванням того, що ефективність розпаралелення методу групового навчання не залежить від кількості епох навчання [17]. Для реальних задач модель БШП вимагає $10^6 \ldots 10^9$ епох для забезпечення достатньо хорошого навчання моделі, в цьому випадку реальний час обчислень буде набагато більшим і застосування паралельних методів навчання стає все більш необхідним.

6. Висновки

В цій статті представлено паралельний метод групового навчання багатошарового персептрону на основі алгоритму зворотного поширення помилки та дослідження його ефективності на обчислювальних кластерах з комунікаційними інтерфейсами Infiniband та Gigabit Ethernet у порівнянні з паралельним комп’ютером ccNuma архітектури. Отримані експериментальні результати показали, що (i) завдяки малим комунікаційним втратам кластер з комунікаційним інтерфейсом Infiniband забезпечує кращу ефективність розпаралелення, ніж суперкомп’ютер з ccNuma архітектурою та (ii) ефективність розпаралелення розробленого паралельного методу є достатньо високою для його ефективного використання на паралельних обчислювальних системах загального приз-начення, наявних у сучасних ГРІД-системах.

Майбутні дослідження доцільно провести в напрямку оцінки ефективності розпаралелення розробленого методу в середовищі серединного програмного забезпечення ГРІД-систем. Експерименти показали, що паралельний комп'ютер з меншою продуктивністю процесорів, але з оптимізованою архітектурою, забезпечує швидке виконання розробленого паралельного методу. Тому для правильного вибору паралельної обчислювальної системи, що має виконувати цей метод в середовищі ГРІД, доцільно ввести додатковий критерій вибору – «вартість» паралельної системи, що може відображати справжню вартість обчислювальної системи, її архітектурні особливості, завантаження задачами та інші необхідні характеристики.

Подяки

Це дослідження фінансовано міжнародною «в’їздною» стипендією ім. Марії К’юрі №221524-908524 “PaGaLiNNeT – Parallel Grid-aware Library for Neural Networks Training" 7-ї Рамкової програми Європейського союзу. Автор також дякує адміністрації та технічним працівникам Суперкомп’ютерного та мережевого центру м. Познань (Польща), Лабораторії інноваційних обчислень (ICL) університету шт. Теннессі (США) та Суперкомп’ютерного центру обчислювальної інженерії Калабрійського університету (Італія) за використання їх обчислювальної ГРІД-інфраструктури.

Список використаних джерел

1. Haykin S. Neural networks and learning machines / S. Haykin. – Prentice Hall, 2008. – 936 p.
2. Mahapatra S., Mahapatra R., Chatterji B. A Parallel Formulation of Back-propagation Learning on Distributed Memory Multiprocessors // Parallel Computing, 1997. – Vol. 22, No. 12. – P. 1661-1675.
3. Hanzálek Z. A Parallel Algorithm for Gradient Training of Feed-forward Neural Networks // Parallel Computing, 1998. – Vol. 24, No. 5 -6. – P. 823-839.
4. Murre J.M.J. Transputers and Neural Networks: An Analysis of Implementation Constraints and Performance // IEEE Transactions on Neural Networks, 1993. – Vol. 4, No. 2. – P. 284-292.
5. Topping B.H.V., Khan A.I., Bahreininejad A. Parallel Training of Neural Networks for Finite Element Mesh Decomposition // Computers and Structures, 1997. – Vol. 63, No. 4. – P. 693-707.
6. Vin T.K., Seng P.Z., Kuan M.N.P., Haron F. A Framework for Grid-based Neural Networks // Proceedings of First International Conference on Distributed Frameworks for Multimedia Applications. – 2005. – P. 246-250.
7. Krammer L., Schikuta E., Wanek H. A Grid-based Neural Network Execution Service Source // Proceedings of 24th IASTED International Conference on Parallel and Distributed Computations and Networking. – 2006. – P. 35-40.
8. De Llano R.M., Bosque J.L. Study of Neural Net Training Methods in Parallel and Distributed Architectures // Future Generation Computer Systems, 2010. – Vol. 26, Issue 2. – P. 183-190.
9. Turchenko V., Grandinetti L. Efficiency Analysis of Parallel Batch Pattern NN Training Algorithm on General-Purpose Supercomputer // Lecture Notes in Computer Science. – Berlin, Heidelberg: Springer-Verlag, 2009. – No. 5518. – P. 223 - 226.
10. Turchenko V., Grandinetti L. Minimal Architecture and Training Parameters of Multilayer Perceptron for its Efficient Parallelization // Proceedings of 5th International Workshop on Artificial Neural Networks and Intelligent Information Processing. – Milan (Italy). – 2009. – P. 79-87.
11. Turchenko V., Grandinetti L., Bosilca G., Dongarra J. Improvement of parallelization efficiency of batch pattern BP training algorithm using Open MPI // Elsevier Procedia Computer Science, 2010. – Vol. 1, Issue 1. – P. 525-533.
12. Turchenko V., Grandinetti L. Efficiency Research of Batch and Single Pattern MLP Parallel Training Algorithms // Proceedings of 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS2009). – Rende (Italy). – 2009. – P. 218-224.
13. Головко В.А. Нейронные сети: обучение, модели и применение. – М.: Радиотехника, 2001. – 256 с.
14. http://www.open-mpi.org/
15. Turchenko V., Grandinetti L. Strategy of Resource Brokering for Efficient Parallelization of MLP Training // Proceedings of the 2010 International Conference on High Performance Computing & Simulation (HPCS 2010). – Caen (France). – 2010. – P. 140-149.
16. Турченко В.О. Методологія брокерування Грід-ресурсів на основі Парето-оптимізації // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2011. – № 1. – С. 312-318.
17. Turchenko V. Scalability of Parallel Batch Pattern Neural Network Training Algorithm // Artificial Intelligence, Journal of National Academy of Sciences of Ukraine, 2009. – Vol. 2. – P. 144-150.