УДК 681.3.06

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ ОЦІНКИ І КЛАСИФІКАЦІЇ СТАНІВ СКЛАДНИХ СИСТЕМ НА БАЗІ НЕЧІТКИХ ДАНИХ ТА ЗНАНЬ У ВИСОКОПРОДУКТИВНОМУ ПАРАЛЕЛЬНОМУ СЕРЕДОВИЩІ

І.М. Парасюк, С.В. Єршов, Є.С. Карпінка, О.В. Верьовка

Інститут кібернетики ім. В.М. Глушкова НАН України
03680, Київ-187, проспект Академіка Глушкова, 40,
т.: 526-64-22, e-mail: ivpar1@i.com.ua

Дається огляд архітектури та інструментальних засобів розробки сучасних інформаційних технологій у високопродуктивному паралельному середовищі. Описані основні модулі інформаційної технології оцінки і класифікації станів складних систем на основі нечітких даних та методи розпаралелювання її задач на кластері, які включають функціональну декомпозиції та декомпозиції за гіпотезами. Як приклад, наведено застосування цієї технології для оцінки і класифікації станів економічної системи, що представлена у нечіткому інформаційному просторі із дзвоноподібними функціями належності на основі розширення моделі Байєса на випадок нечітких оцінок вірогідностей у середовищі системи СКІТ.

The review of architecture and tools for development of modern information technologies in a high-performance parallel environment is given. The basic modules of information technology of estimation and classification of the states of the complex systems on the basis of fuzzy data and methods of parallelization of tasks on a cluster are described, which include the methods of functional decompozition and decompozition by hypotheses. As an example, an application of above technology for estimation and classification of the states of the economic system, which is presented in fuzzy information space with bell-like functions of belonging on the basis of expansion of Bayes model in case of fuzzy estimations of probabilities in the environment of the SKIT system is resulted.

Вступ

Високопродуктивні паралельні середовища в нинішній час широко використовуються для комп'ютеризації складних прикладних задач, що вимагають неприйнятних затрат часу на їх розв'язання [1]. До них можна віднести і задачі класифікації станів складних систем на базі нечітких даних та знань, у яких потреба надпотужних обчислень зростає у зв'язку із суттєвим ускладненням інтерполяційних процедур, що використовуються при виведенні висновків [2, 3].

Цілком зрозуміло, що процес створення інформаційних технологій для таких обчислювальних середовищ, повинен враховувати особливості, як архітектур і наявних інструментальних засобів таких середовищ, так і специфіку самих предметних областей, на комп'ютеризацію яких ці технології орієнтовані.

В даній роботі узагальнюється досвід авторів, накопичений при створенні інформаційної технології оцінки і класифікації станів складних систем на базі нечітких даних та знань, що розроблені в Інституті кібернетики ім. В.М. Глушкова НАН України для функціонування у складі кластерної системи СКІТ [1].

Основна увага приділяється організації взаємодії окремих модулів інформаційної технології, методам розпаралелювання задач класифікації станів складних систем на кластері, які включають гнучкі способи декомпозиції та організації обчислювального процесу. Функціонування зазначеної технології на кластері проілюстровано на прикладі оцінки і класифікації станів економічної системи, що представлена у нечіткому інформаційному просторі із дзвоноподібними функціями належності на основі розширення моделі Байєса на випадок нечітких оцінок вірогідностей.

Архітектура високопродуктивного паралельного середовища

У процесі становлення суперкомп'ютерних технологій привабливою і перспективною стала ідея підвищення продуктивності архітектури обчислювальної системи за рахунок збільшення числа процесорів [4].

Перші промислові зразки високопродуктивного паралельного середовища з'явилися на базі векторно-конвейєрних комп'ютерів у середині 80-х років. У подібних комп'ютерах об'єднувалося від 2 до 16 процесорів, які мали рівноправний (симетричний) доступ до спільної оперативної пам'яті. У зв'язку з цим вони отримали назву симетричні мультипроцесорні системи (Symmetric Multi-Processing - SMP).

Як альтернатива таким дорогим мультипроцесорним системам на базі векторно-конвейєрних процесорів була запропонована ідея будувати еквівалентні за потужністю багатопроцесорні системи з великого числа дешевих мікропроцесорів, що серійно випускаються. Виявилося, що SMP архітектура має дуже обмежені можливості по нарощуванню числа процесорів у системі через різке збільшення числа конфліктів при зверненні до спільної шини пам'яті. У зв'язку з цим виправданою представлялася ідея забезпечити кожний процесор власною оперативною пам'яттю, перетворюючи комп'ютер на об'єднання незалежних обчислювальних вузлів. Такий підхід значно збільшив ступінь масштабованості багатопроцесорних систем, але у свою чергу вимагав розробки спеціального способу обміну даними між обчислювальними вузлами, який реалізується у вигляді

механізму передачі повідомлень (Message Passing). Комп'ютери з такою архітектурою є найяскравішими представниками MPP систем (MPP - Massively Parallel Processing, системи з масовим паралелізмом). У даний час ці два напрями (або певні їх комбінації) є домінуючими в розвитку суперкомп'ютерних технологій.

Чимось середнім між SMP і MPP є NUMA-архітектура (Non Uniform Memory Access), в якій пам'ять фізично розділена, але логічно загальнодоступна. При цьому час доступу до різних блоків пам'яті стає неоднаковим. У даний час розвиток суперкомп'ютерних технологій здійснюється чотирьома основними напрямками: векторно-конвейєрні суперкомп'ютери, SMP системи, MPP системи і кластерні системи.

Кластерні системи стали логічним продовженням розвитку ідей, закладених в архітектурі MPP систем. Якщо процесорний модуль в MPP системі є закінченою обчислювальною системою, то як обчислювальні вузли використовують звичайні комп'ютери, що серійно випускаються. Привабливою рисою кластерних технологій є те, що вони дозволяють для досягнення необхідної продуктивності об'єднувати в єдині обчислювальні системи комп'ютери самого різного типу. Широке розповсюдження кластерні технології отримали, як засіб створення високопродуктивного обчислювального середовища з складових частин масового виробництва, що значно здешевлює вартість обчислювальної системи.

Переваги кластерної системи перед набором незалежних комп'ютерів очевидні. По-перше, розроблено багато диспетчерських систем пакетної обробки завдань, що дозволяють послати завдання на обробку кластеру в цілому, а не якомусь окремому комп'ютеру. Ці диспетчерські системи автоматично розподіляють завдання по вільних обчислювальних вузлах або буферизують їх за відсутності таких, що дозволяє забезпечити більш рівномірне й ефективне завантаження комп'ютерів. По-друге, з'являється можливість сумісного використання обчислювальних ресурсів декількох комп'ютерів для вирішення однієї задачі. Кожний з комп’ютерів може функціонувати під управлінням своєї власної операційної системи. Частіше за все використовуються стандартні операційні системи: Linux, FreeBSD, Solaris, Windows NT.

При створенні кластерів можна виділити два підходи. Перший підхід застосовується при створенні відносно невеликих за показниками продуктивності кластерних систем. В кластер об'єднуються повнофункціональні комп'ютери, які продовжують працювати і як самостійні одиниці. Другий підхід застосовується в тих випадках, коли цілеспрямовано створюється могутній обчислювальний ресурс. Тоді системні блоки комп'ютерів компактно розміщуються, а для управління системою і для запуску задач виділяється один або декілька повнофункціональних комп'ютерів, званих хост-комп'ютерами. Саме з цією метою у 2004–2005 рр. в Інституті кібернетики ім. В.М. Глушкова розроблено високопродуктивне середовищє кластерної системи СКИТ [1].

Різноманітність існуючих високопродуктивних паралельних середовищ вимагала ввести для них певну класифікацію. Одна з перших класифікацій, посилання на яку найбільш часто зустрічаються в літературі, була запропонована М. Флинном у кінці 60-х років минулого століття. Вона базується на поняттях двох потоків: команд і даних. На основі кількості цих потоків виділяється чотири класи архітектури: SISD (Single Instruction Single Data) – єдиний потік команд і єдиний потік даних; SIMD (Single Instruction Multiple Data) – єдиний потік команд і множинний потік даних; MISD (Multiple Instruction Single Date) – множинний потік команд і єдиний потік даних; MIMD (Multiple Instruction Multiple Date) – множинний потік команд і множинний потік даних. У цій класифікації всі види сучасних високопродуктивних середовищ належать одному классу – MIMD. Проте, терміни, які використовуються в ній, достатньо часто згадуються в літературі за паралельними обчисленнями [4–6].

Інструментальні засоби розробки інформаційних технологій у високопродуктивному паралельному середовищі

Ефективність використовування високопродуктивного середовища для задач класифікації станів складних систем у вирішальному ступені залежить від складу і якості програмного забезпечення, встановленого на них. У першу чергу це стосується програмного забезпечення, призначеного для розробки прикладних програм класифікації у нечіткому інформаційному просторі.

Останніми роками все більш популярною стає система програмування OpenMP [4, 6], що є багато в чому узагальненням і розширенням цих наборів директив. Інтерфейс OpenMP задуманий як стандарт для програмування в моделі із спільною пам'яттю. Вся програма розбивається на послідовні і паралельні області. Всі послідовні області виконує головна нитка, породжувана при запуску програми, а при вході в паралельну область головна нитка породжує додаткові нитки. Слід зазначити, що наявність загальної пам'яті не перешкоджає використанню технологій програмування, розроблених для систем з розподіленою пам'яттю. Разом з тим незалежними розробниками програмного забезпечення була запропонована велика кількість реалізацій механізму передачі повідомлень, незалежних від конкретної платформи. Самими відомими з них є EXPRESS компанії Parasoft і комунікаційна бібліотека PVM (Parallel Virtual Machine) [4, 5].

У 90-х р. був прийнятий до реалізації стандарт механізму передачі повідомлень MPI (Message Passing Interface) [5, 7]. Реалізації MPI успішно працюють не тільки на класичних MPP системах, але також на SMP системах і на мережах робочих станцій (у тому числі і неоднорідних). MPI – це бібліотека функцій, що забезпечує взаємодію паралельних процесів за допомогою механізму передачі повідомлень. Підтримуються інтерфейси для мов C і FORTRAN. Останнім часом забезпечена також підтримка мови C++. Бібліотека включає множину функцій передачі повідомлень типу точка-точка, розвинутий набір функцій для виконання

колективних операцій і управління процесами паралельного застосування. Основна відмінність MPI від попередників у тому, що явно вводяться поняття груп процесів, з якими можна оперувати, як з кінцевими множинами, а також областей зв'язку та комунікаторів, що описують ці області зв'язку. Це надає програмісту дуже гнучкі засоби для написання ефективних паралельних програм. У даний час MPI є основною технологією програмування для багатопроцесорних систем з розподіленою пам'яттю [4, 5, 7].

Не зважаючи на значні успіхи в розвитку технології програмування з використанням механізму передачі повідомлень, трудомісткість розробки програмного забезпечення з використанням цієї технології все-таки дуже велика.

Альтернативний підхід надає парадигма паралельної обробки даних, яка реалізована в мові високого рівня HPF [6]. Від програміста вимагається тільки задати розподіл даних по процесорах, а компілятор автоматично генерує виклики функцій синхронізації та передачі повідомлень. На першому етапі HPF-програма перетвориться в стандартну Фортран-програму, доповнену викликами комунікаційних підпрограм, на другому – відбувається її компіляція стандартними компіляторами. Також, слід зазначити систему компіляції Adaptor, розроблену німецьким Інститутом алгоритмів і наукових обчислень, і систему розробки паралельних програм DVM, створену в Інституті прикладної математики ім. М.В. Келдиша РАН. Але в більшості випадків ефективність паралельних HPF-програм значно нижче, ніж MPI-програм.

Принципи організації паралелізму у високопродуктивному середовищі

Паралельні технології на високопродуктивних MPP-системах допускають дві моделі програмування (схожі на класифікацію М. Фліна): SPMD (Single Program Multiple Date) – на всіх процесорах виконуються клони (копії) однієї програми, які обробляють різні блоки даних (див. також [8]); MPMD (Multiple Program Multiple Date) – на процесорах виконуються різні програми.

Другий варіант іноді називають функціональним розпаралелюванням. Такий підхід, зокрема, використовується в інформаційних технологіях, коли кожна множина елементів даних має проходити кілька етапів обробки, які виконуються на окремих процесорах. Проте такий підхід має вельми обмежене застосування, оскільки організувати достатньо довгий конвеєр, та ще з рівномірним завантаженням всіх процесорів, вельми складно. Фактично, в [5] показано, яким способом архітектуру MPMD можна промоделювати у системах SPMD.

Найпоширенішим режимом роботи на системах з розподіленою пам'яттю є завантаження в деяку кількість процесорів клонів деякої програми. Розробка паралельної програми має на увазі розбиття задачі на P підзадач, кожна з яких розв'язується на окремому процесорі. Таким чином, спрощену схему паралельної програми, що використовує механізм передачі повідомлень, можна представити таким чином:

```
if (proc_id == 0) {
   task1
}
IF (proc_id == 1) {
   task2
}
...
result = reduce(result_task1, result_task2, ...)
```

Тут proc_id – ідентифікатор процесора, а функція reduce формує певний глобальний результат на основі локальних результатів, отриманих на кожному процесорі. В цьому випадку клони програми виконуватимуться на *P* процесорах, на кожному із яких вирішуватиметься певна підзадача. Якщо розбиття на підзадачі достатньо рівномірне, а накладні витрати на обміни не дуже великі, то можна чекати близького до P коефіцієнта прискорення рішення задачі.

Питання розподілу даних по процесорах і зв'язок цього розподілу з ефективністю паралельної програми є основним питанням паралельного програмування. Розподіл масивів по процесорах дозволяє вирішувати значно більш об'ємні задачі , але тоді на перший план виступає проблема мінімізації пересилок даних.

Розглянемо, якими властивостями має володіти багатопроцесорна система MPP типу для виконання на ній паралельних програм. Мінімально необхідний набір це такий:

- процесори в системі повинні ідентифікуватись унікальними номерами;
- існує функція ідентифікації процесором самого себе;
- існують функції обміну між двома процесорами: посилка повідомлення одним процесором і прийом повідомлення іншим процесором.

Парадигма передачі повідомлень має на увазі асиметрію функцій передачі і прийому повідомлень. Ініціатива ініціалізації обміну належить передаючій стороні. Приймаючий процесор може прийняти тільки те, що йому було послане. В комунікаційних операціях типу точка-точка завжди беруть участь не більше двох процесів: передаючий і приймаючий. Блокуючі функції мають на увазі вихід з них тільки після повного закінчення операції, тобто процес, який викликає функцію, блокується, поки операцію не буде завершено. Неблокуючі функції мають на увазі поєднання операцій обміну з іншими операціями, тому неблокуючі функції передачі і прийому по суті справи є функціями ініціалізації відповідних операцій. Для опиту завершеності операції (і завершення) вводяться додаткові функції.

Як для блокуючих, так і неблокуючих операцій MPI підтримує чотири режими виконання. Ці режими стосуються тільки функцій передачі даних, тому для блокуючих і неблокуючих операцій є по чотири функції посилки повідомлення. В таблиці наведено імена базових комунікаційних функцій типу точка-точка бібліотеки MPI, шо використовуються окремими модулями інформаційної технології.

Список комунікаційних функцій MPI типу точка-точка Таблиця

Режими виконання	З блокуванням	Без блокування
Стандартна посилка	MPI_Send	MPI_Isend
Синхронна посилка	MPI_Ssend	MPI_Issend
Буферизуюча посилка	MPI_Bsend	MPI_Ibsend
Узгоджена посилка	MPI_Rsend	MPI_Irsend
Прийом інформації	MPI_Recv	MPI_Irecv

До імен базових функцій Send/Recv додаються різні префікси: S (synchronous) – синхронний режим передачі даних, операція закінчується тільки тоді, коли закінчується прийом даних; B (buffered) – буферизуючий режим передачі даних, операція посилки закінчується, коли дані поміщені в цей буфер; R (ready) – погоджений або підготовлений режим передачі даних, операція передачі даних починається тільки тоді, коли приймаючий процесор виставив ознаку готовності прийому даних, ініціювавши операцію прийому; I (immediate) – неблокуючі операції.

Використання в інформаційній технології класифікації тільки неблокуючих комунікаційних операцій підвищує безпеку з погляду виникнення тупикових ситуацій, а також збільшує швидкість роботи програм за рахунок поєднання виконання обчислювальних і комунікаційних операцій. Ці задачі розв'язуються розділенням комунікаційних операцій на дві стадії: ініціацію операції і перевірку завершення операції.

Реалізація інформаційної технології на кластерній системі СКІТ

Процес класифікації стану складних систем у високопродуктивному кластерному середовищі включає:

– завантаження знань та фактів, які зберігаються у файлах;

– паралельне обчислення нечітких оцінок вірогідностей гіпотез на окремих вузлах кластерної системи;

– отримання і об’єднання результатів окремих вузлів;

– представлення результатів класифікації та її візуалізація.

Основні етапи функціонування інформаційної технології показані на рис. 1.

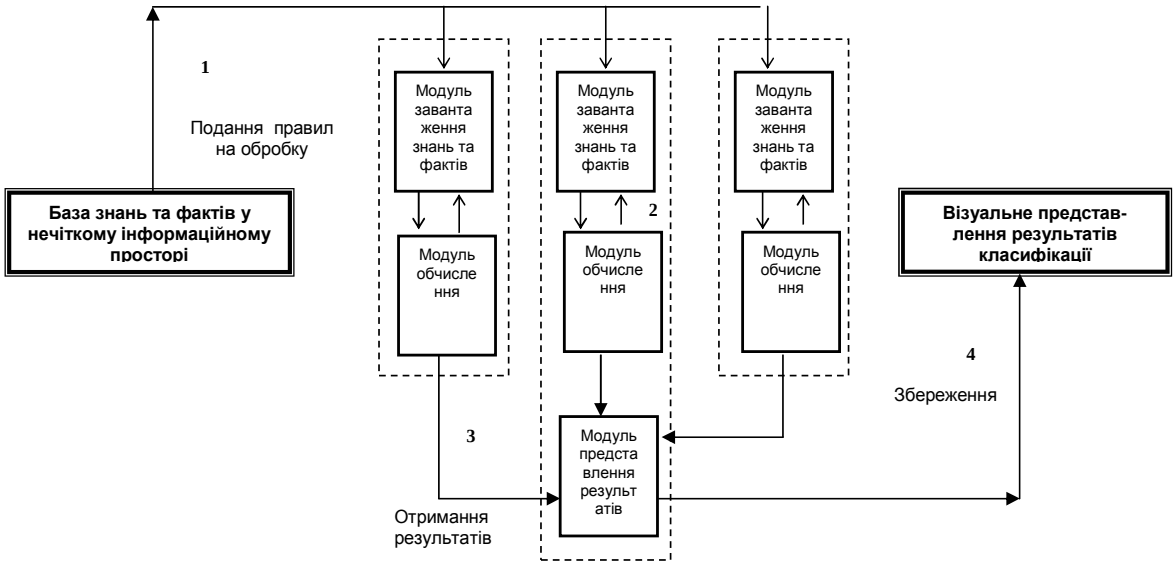

Рис. 1. Основні модулі інформаційної технології класифікації стану складної системи структура програми

Усі паралельні процеси програм завантажують знання та факти з трьох файлів: файлу опису симптомів, файлу опису гіпотез та файлу правил виведення, а також в окремих файлах, які представляють значення певних дискретних функцій належності, що використовуються при здійсненні нечіткого байєсівського виведення. Завантаження відбувається усіма паралельними процесами одночасно.

На першому етапі роботи всі процеси задачі завантажують дані із файлів представлення симтомів, гіпотез та правил у нечіткому поданні. Завантаження відбувається одночасно усіма паралельними процесами – повністю або вибірково, якщо певні дані не будуть використані даним процесором.

У файлі представлення значень симптомів описуються номери, назви симптомів та лінгвістичні значення, кожне із яких відповідає суб'єктивному фактору визначеності для кожного симптому і задане за допомогою відповідної функції належності в окремому файлі:

< порядковий_номер, назва_симптому, лінгвістичне значення>.

У файлі опису можливих гіпотез задаються назви окремих гіпотез та назви апріорних (початкових) функцій належності окремих гіпотез:

< порядковий_номер, назва_гіпотези, назва функції належності>.

У файлі правил виведення задані нечіткі правила виведення у такому форматі:

<номер гіпотези, номер симптому , назва функції належності для P(E|H), назва функції належності для P(E|not H)>.

Розпаралелювання організовано у вигляді SPMD архітектури, відповідно до якої на усіх вузлах кластера виконується одна і та сама програма. Роль, яку кожний модуль обчислення грає у розподіленому обчисленні, визначається рангом — унікальним номером, що дається при запуску.

При обчисленні апостеріорних ймовірностей результати стосовно кожної гіпотези надсилаються координуючому процесу (з номером 0), який відповідає за їх збереження та об'єднання в єдиному масиві результатів. Для пересилання результатів використовуються комунікаційні операції MPI типу точка-точка та файлова система LAM MPI. Для перевірки завершення обчислень усіма процесорами, які виконують обчислення, використовується бар'єрна синхронізація MPI_Barrier.

На останньому етапі, після того як обчислення $P^*(H)$ закінчені усіма процесами, проводиться класифікація гіпотез за рівнем достовірності. Для цього проводиться дефузифікація значення нечіткої змінної $P^*(H)$ для кожної гіпотези методом "центра тяжіння", і відсортовані чіткі апостеріорні оцінки гіпотез видаються користувачу у порядку зменшення. Для візуалізації нечітких функцій належності можуть бути використані як спеціальні програмні продукти, такі як IMB DataViz, Visio Pro, так і математичні пакети загального призначення MathCAD та MathLab [9]. Для спрощення процесу візуалізації нечіткої бази знань та результатів було створено просту графічну оболонку, призначення якої – введення експертом та наочне подання функцій належності, що представлені у файловому форматі. Зазначена оболонка дозволяє оперативно завантажувати за протоколом FTP файли із обрахованими значеннями функцій належності, здійснювати перегляд та оцінювання їх узагальнених характеристик, таких як границі носія, інтервал довіри, "дефузифіковане" за певним алгоритмом значення тощо.

Основні класи гіпотез: найбільш імовірна гіпотеза, домінуючі гіпотези, актуальні гіпотези, відхилені гіпотези. Результатом роботи програми нечіткої байєсівської класифікації гіпотез є список значень апостеріорних ймовірностей найбільш вірогідних гіпотез, а також список значень апостеріорних оцінок ймовірностей актуальних гіпотез. Результати класифікації записуються у вихідному файлі, розташування якого визначається у паспорті задачі.

Распаралелювання програмних засобів інформаційної технології зводиться до процесу декомпозиції задачі на незалежні процеси, що не вимагають послідовного виконання і можуть, відповідно, бути виконані на різних процесорах системи СКІТ паралельно. Існує три основних варіанти декомпозиції: проста декомпозиція (trival), функціональна (functional) і декомпозиція даних.

Тривіальна декомпозиція – найбільш простий тип декомпозиції. Застосовується вона в тому випадку, коли різні копії лінійного коду можуть виконуватися незалежно одна від одної і не залежать від результатів, що отримані у процесі виконання інших копій коду. В цьому випадку основна програма, одержавши різні початкові параметри, може бути запущена на різних процесорах кластера.

При функціональній декомпозиції вихідна задача розбивається на ряд послідовних задач, що можуть бути виконані на різних вузлах кластера не залежно від проміжних результатів, але строго послідовно. Кожна з цих задач може бути виконана на окремому вузлі кластера. Загальний час обробки всього масиву елементів даних помітно знижується за рахунок того, що йде обробка відразу кількох послідовних елементів масиву даних.

За наведеним сценарієм дані обробляються в режимі конвеєра. На програміста, що обрав функціональний тип декомпозиції задачі, лягає обов'язок не тільки стосовно вибору декомпозованих функцій, але і по організації роботи паралельних частин програми в режимі конвеєра, тобто правильної організації процедури одержання вихідних даних від попереднього процесу і передачі оброблених даних наступному процесу.

На відміну від функціональної декомпозиції, коли між процесорами розподіляються різні задачі, декомпозиція даних припускає виконання на кожному процесорі однієї і тієї ж задачі, але над різними наборами даних. Частини даних спочатку розподілено між процесорами, що обробляють їх, після чого результати складаються деяким чином в одному місці (звичайно на консолі кластера). Дані повинні бути розподілені так, щоб обсяг роботи для кожного процесора був приблизно однаковий, тобто декомпозиція має бути збалансованою. У випадку дисбалансу ефективність роботи кластера може бути знижена.

У випадку, коли область даних задачі може бути розбита на окремі непересічні області, обчислення в яких можуть йти незалежно, маємо регулярну декомпозицію. Дуже багато задач базуються на обробці великих масивів даних, структурованих у регулярні структури (матриці або послідовності). У тому випадку, коли оброблювана структура даних може бути розбита на регулярний (непересічний) масив підструктур (областей), задача може бути розподілена між процесорами кластера і вирішена в паралельному режимі. Це або дозволить

скоротити час рішення задачі, або поставити задачу для більшого масиву даних (наприклад зробити різницеву сітку більш дрібною). Для задачі байєсівської класифікації декомпозиція даних є найбільш придатним способом підготувати програму для виконання на паралельній машині.

При розробці паралельного алгоритму байєсівського методу класифікації у нечіткому інформаційному просторі використовувалися два типи декомпозиції: регулярна декомпозиція на вхідному масиві даних та функціональна декомпозиція.

Процес регулярної декомпозиції за даними розбиває всю множину симптомів $E = \{E_1, E_2, ..., E_n\}$ на множини симптомів $E_i(H_i)$, від яких залежить обчислення апостеріорних ймовірностей окремих гіпотез H_i, $i = 1,..., m$ (рис. 2).

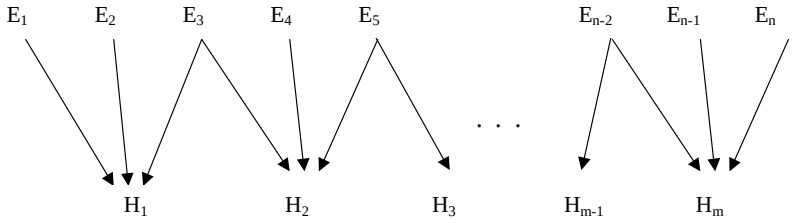

Рис. 2. Регулярна декомпозиція за гіпотезами

При цьому множини симптомів $E_i(H_i)$ можуть частково перетинатися, утворюючи так звані граничні симптоми, значення яких враховують у розрахунку кількох гіпотез одночасно. Обчислення для кожної гіпотези можуть виконуватися паралельно, як окремий процес, на окремому комп'ютері кластерної машини. Оскільки кількість вільних процесорів *N* кластерної машини найчастіше буває меншою ніж кількість гіпотез у реальних задачах байєсівського виведення, кожен процесор має обраховувати апостеріорні ймовірності *P**(*H*) для приблизно однакової кількості гіпотез (рис. 3). У цьому полягає принцип регулярності декомпозиції по множині гіпотез.

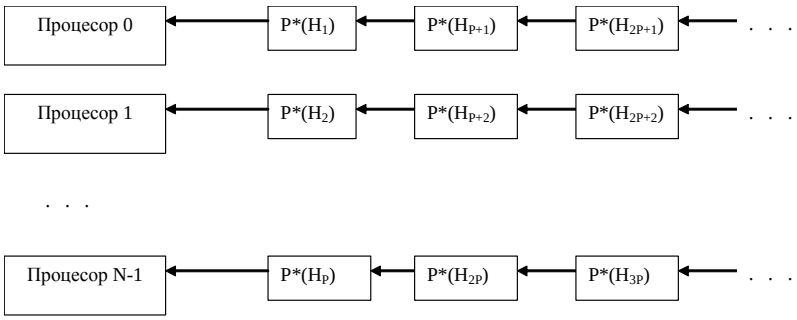

Рис. 3. Обчислення ймовірностей P*(H) окремими процесорами кластерного комп'ютера

Як видно з рис. 3, процесор з номером I має обчислювати ймовірності (I+1), … (2N + 1), (3N +1), де N – кількість процесорів, на яких було запущено програму. При цьому процесор з номером 0 бере на себе функцію координатора – його мета отримати результати обчислень *P**(*H*) від кожного процесора та вивести результати у вихідний файл. При цьому обчислення *P**(*H*) виконується ітеративно за списком симптомів, від яких залежить дана гіпотеза. Одночасно з обчисленням *P**(H_i) проводиться обчислення значень, які використовуються в координуючому процесі для класифікації гіпотез за значеннями, що надійшли від усіх інших процесорів. Користувачу видаються значення найбільш ймовірних та домінуючих гіпотез.

Функціональна декомпозиція, яку застосовують у програмах байєсівського оцінювання, використовує властивості формули виведення, яку можна переписати у наступному вигляді:

$$P*(H/E) = Q(P(H/E), P(H/\overline{E})), \quad (1)$$

де Q – узагальнена інтерполяційна обчислювальна процедура байєсівського оцінювання. Значення $P(H|E)$ та $P(H|\overline{E})$ обчислюються фактично незалежно на кожній ітерації, вони залежать тільки від значення *P**(*H*) на попередній ітерації обчислення гіпотези. Тому функціональна декомпозиція – це декомпозиція обчислення *P**(*H*|*E*) на кожній ітерації для кожної гіпотези H. Значення $P(H|E)$ обчислює той самий процесор, який виконує обчислення *P**(*H*) – процесор з непарним номером. Для обчислення $P(H|\overline{E})$ використовують процесори з парними номерами. Фактично для обчислення значення *P**(*H*) для однієї гіпотези вимагається використання пари процесорів – парного та непарного. Такий спосіб використання процесорів кластерного комп'ютера є найбільш збалансованим, оскільки парний процесор може починати обчислення нової ітерації не дочікуючись завершення роботи непарного на попередній ітерації.

Головну програму для реалізації алгоритму класифікації написано мовою C++ з використанням бібліотеки шаблонів STL [10]. Фрагмент головного модуля програми показано на рис. 4.

```
int main(int argc, char** argv)
{
// Завантаження даних в оперативну пам'ять з файлів Symptoms.txt, Hypotheses.txt та
RulesConclusion.txt
    DATA_DOWNLOAD_MODULE();

    int me, size;

    #ifdef PARALLEL
       MPI_Init(&argc, &argv); // ініціалізація MPI
       MPI_Comm_rank(MPI_COMM_WORLD, &me);
// обчислення поточного номера процесу
       MPI_Comm_size(MPI_COMM_WORLD, &size);
// визначення кількості запущених процесів
       me++; // для зручності нумерацію починаємо з 1
    #endif

    #ifndef PARALLEL
       if (argc>1) {
         Accuracy_ = atoi(argv[1]);
       } else
         Accuracy_ = 30;   // Точність за умовчанням
    #else
       Accuracy_ = ACCURACY;
    #endif

    cout << "Accuracy is " << Accuracy_ << "."<<"\n";

       CLASSIFY_MODULE(me, size); // Головна послідовність обчислень програми

    if (me==1)SORTING_MODULE(); // Процесор з номером 1 здійснює сортування
                                // апостеріорних значень нечітких гіпотез
    #ifdef PARALLEL
       MPI_Finalize(); // Завершення роботи з MPI
    #endif
    return 0;
}
```

Рис. 4. Фрагмент головної частини програми класифікації

Функція визначення числа процесів в області зв'язку MPI_Comm_size дозволяє визначити кількість процесорів, на яких запущено задачу. Функція визначення номера процесу MPI_Comm_rank повертає номер поточного процесу, який виконується. Дана інформація використовується для динамічної декомпозиції (за даними та функціональна) задачі класифікації у нечіткому просторі. Мінімальна кількість процесорів, на якій працює даний алгоритм – один.

Обмін між парним та непарним процесорами реалізовано за допомогою комунікаційних операцій типу точка-точка, у яких завжди беруть участь не більш двох процесів: передавальний і приймаючий.

Приклад ралізації інформаційної технології класифікації станів економічної системи

Однією із задач, що розв'язувалась у вищеописаному комп'ютерному середовищі є задача оцінки і класифікації станів деякої економічної системи, що представлена у нечіткому інформаційному просторі з дзвоноподібними функціями належності [2, 3]. Основними об'єктами такої системи виступають гіпотези-стани економіки та окремі свідчення-показники економічного розвитку даної системи. Ймовірнісна базова модель відношень між цими об'єктами – це аналог формули Байєса на випадок нечітких оцінок вірогідностей.

Представимо основні компоненти таких моделей, що використовуються при реалізації поставленої задачі.

Для цього використаємо позначення:

$\mu_A(x)$ – функція належності нечіткого числа A [11, 12];

$\mu_{P(A)}(x)$ – функція належності нечіткої оцінки вірогідності P(A);

E_i $(i=\overline{1,I})$ – можливі гіпотетичні стани, в яких може перебувати досліджувана економічна система;

G_j $(j=\overline{1,J})$ – економічні показники (свідчення) досліджуваної системи;

S_{jk} $(j=\overline{1,J}\,;\ k=\overline{1,K})$ – рівень (стан), до якого можна віднести досягнуті поточні значення показника G_j.

База знань відображає проведені експериментальні дослідження економічних станів системи і їх показників і включає $\forall E_i$, S_{jk} відповідні функції належності: $\mu_{P^0(E_i)}(x)$, $\mu_{P(S_{jk}|E_i)}(x)$), $\mu_{P(S_{jk}|Ei)}$ – (x) . Індекс „ноль" показує, що дані оцінки є початкові та апріорні.

Величини $P(S_{jk}| E_i)$ та $P(S_{jk}| \overline{E}_i)$ – суть правдоподібності (умовні вірогідності).

У базах знань ці величини розглядаються в якості правил-продукцій. Зауважимо, що величини $\mu_{P(S_{jk})}(x)$ у поточний момент часу вважаються невідомими. Їх прийнято [3] оцінювати наближено з допомогою нечіткої оцінки міри невизначеності станів. У якості такої міри перебування показників G_j у стані S_{jk}, як правило, виступають позиції $\sigma=\{\sigma_1,\sigma_2,\ldots,\sigma_L\}$ на деякій дискретній шкалі σ, наприклад (+5, +4, +3, +2, +1,0,-1,-2,-3,-4,-5). У даній реалізації – це певні нечіткі числа із функціями належності $\mu_{\sigma_i}(x)$, що ідентифіковані певними лінгвістичними термами. Нами використовуються відповідно такі терми: {**впевнено ТАК**, **напевно ТАК**, **вірогідно ТАК**, **можливо ТАК**, **сумнівно ТАК, не знаю, сумнівно НІ, вірогідно НІ**, **можливо НІ, напевно НІ, впевнено НІ**}.

Лабораторна база знань, на якій випробовується ймовірнісна модель економічної системи, включає такі гіпотези (стани) E_i:

{**максимальне піднесення, стрімке піднесення, стійке піднесення, піднесення, повільне піднесення (повільний розвиток), стагнація, повільне падіння, падіння, стійке падіння, стрімке падіння, максимальне падіння**}.

Кожен із показників G_j соціально-економічного розвитку (**валовий внутрішній продукт, капітальні вкладання, експорт товарів та послуг, індекс споживчих цін, рівень інфляції, сальдо зведеного бюджету** та інші) оцінюється в станах S_{jk} :

{**максимальне зростання, стрімке зростання, високі темпи зростання, зростання, повільне зростання, депресія, повільне падіння, падіння, високі темпи падіння, стрімке падіння, максимальне падіння**}

відповідно до позиції на нечіткій шкалі.

На рис. 5 показано експериментальну функцію належності $\mu_{P^0(E_i)}(x)$ окремого екземпляра нечіткої величини із бази знань. Екземпляри нечітких чисел $P(S_{jk}|E_i)$ та $P(S_{jk}|\overline{E_i})$ приведені у символьному вигляді. Так нечіткі числа $P(\textit{валовий внутрішній продукт}:\textit{повільне зростання}\,|\,\textit{повільний розвиток})$, та $P(\textit{валовий внутрішній продукт}:\textit{повільне зростання}\,|\,\textit{не повільний розвиток})$, подані виразами (2), (3) відповідно.

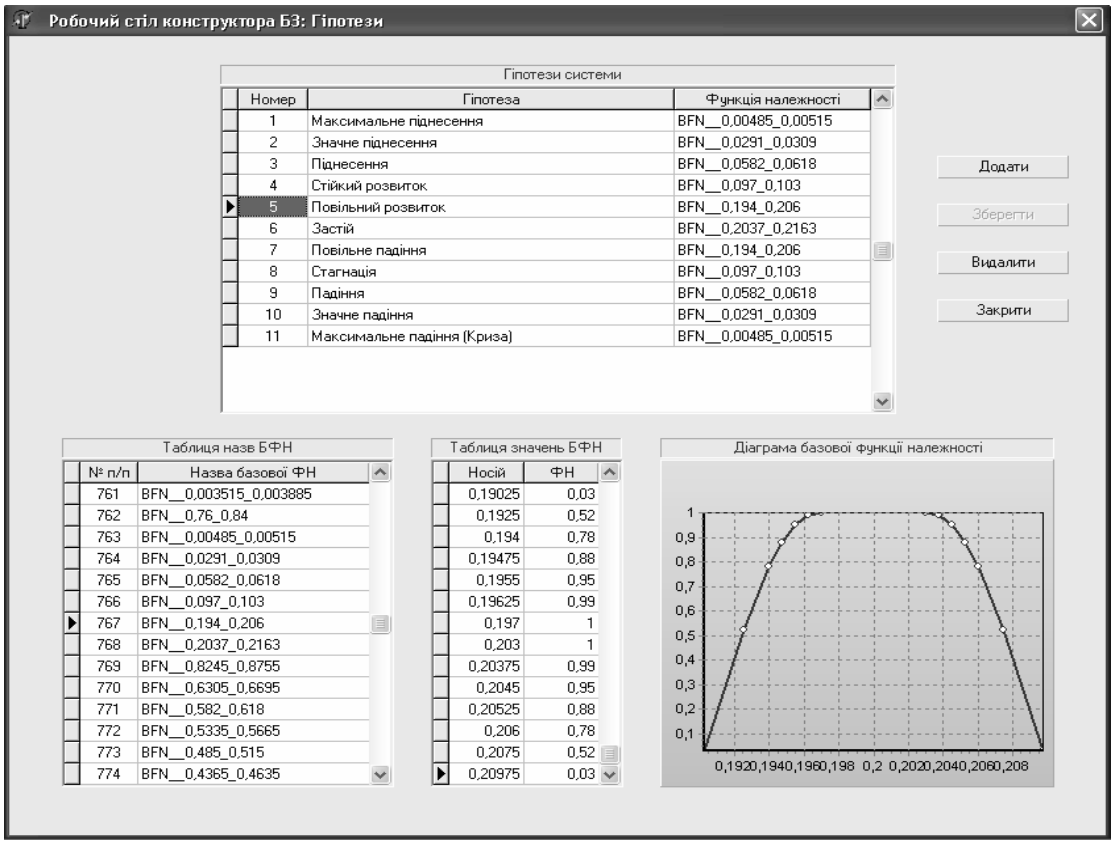

Рис. 5. Функція належності $\mu_{P^0(\text{повільний розвиток})}(x)$

Аналогічним чином приведено екземпляр нечіткого числа σ_i для відповідної нечіткої оцінки. Зокрема, нечітке число **напевно ТАК** подане виразом (4).

$$P^0(E_i) = \frac{0,03}{0,8085625} + \frac{0,52}{0,818125} + \frac{0,78}{0,8245} + \frac{0,88}{0,8276875} + \frac{0,95}{0,830875} + \frac{0,99}{0,8340625} + \frac{1}{0,83725} + \frac{1}{0,86275} + \frac{0,99}{0,8659375} + \frac{0,95}{0,869125} + \frac{0,88}{0,8723125} + \frac{0,78}{0,8755} + \frac{0,52}{0,881875} + \frac{0,03}{0,8914375}, \quad (2)$$

$$P(S_{jk}|E_i) = \frac{0,03}{0,475625} + \frac{0,52}{0,48125} + \frac{0,78}{0,485} + \frac{0,88}{0,486875} + \frac{0,95}{0,48875} + \frac{0,99}{0,490625} + \frac{1}{0,4925} + \frac{1}{0,5075} + \frac{0,99}{0,509375} + \frac{0,95}{0,51125} + \frac{0,88}{0,513125} + \frac{0,78}{0,515} + \frac{0,52}{0,51875} + \frac{0,03}{0,524375}, \quad (3)$$

$$P(S_{jk}|\overline{E}_i) = \frac{0,03}{0,7} + \frac{0,52}{0,73} + \frac{0,78}{0,75} + \frac{0,88}{0,76} + \frac{0,95}{0,77} + \frac{0,99}{0,78} + \frac{1}{0,79} + \frac{1}{0,81} + \frac{0,99}{0,82} + \frac{0,95}{0,83} + \frac{0,88}{0,84} + \frac{0,78}{0,85} + \frac{0,52}{0,87} + \frac{0,03}{0,9}. \quad (4)$$

Екземпляр апостеріорної нечіткої оцінки стану досліджуваної системи показано на (рис 6.). Згідно із цією оцінкою даний стан економіки належить до найбільш вірогідної гіпотези класу **домінуючі гіпотези,** а приведені параметри функції належності характеризують окремі її позиції.

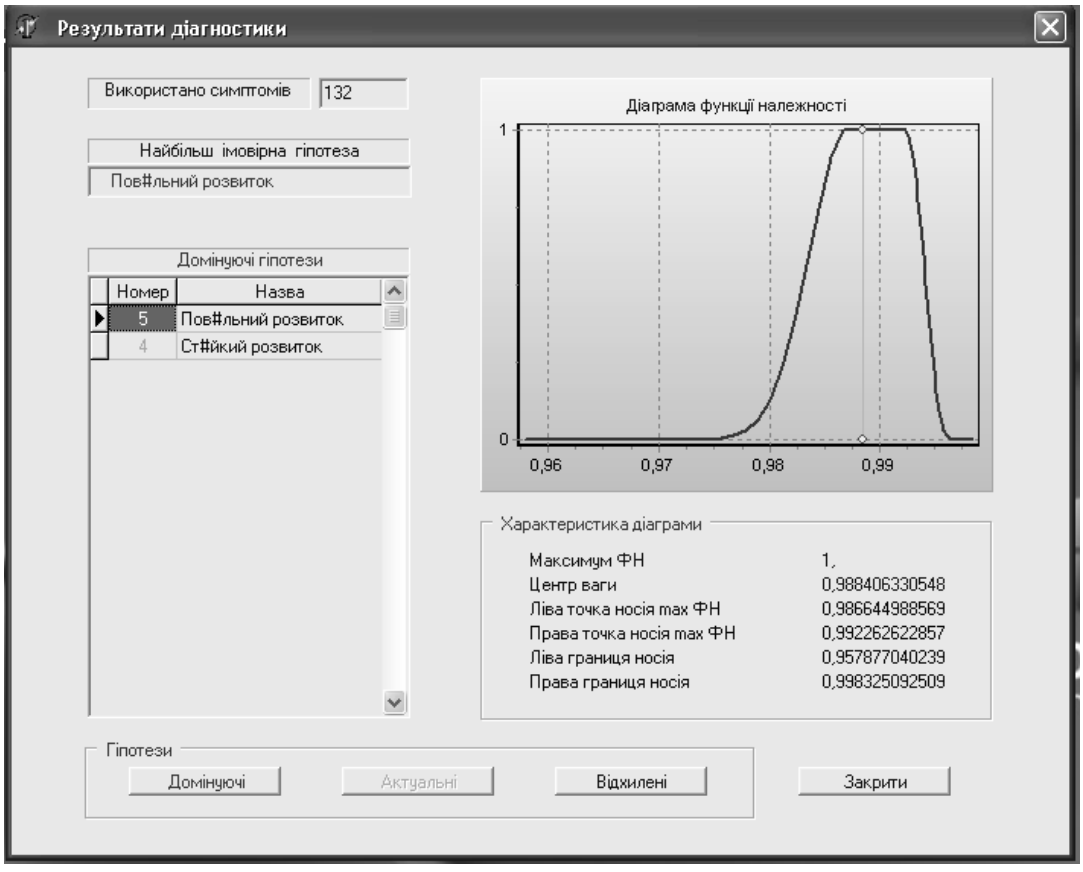

Рис. 6. Функції належності $\mu_{\text{Р(повільний розвиток)}}(x)$

Висновки

Проведено експериментальну перевірку ефективності цієї інформаційної технології в кластерних середовищах родини СКІТ із точністю 100 (кількості точок, якими задаються обчислення значень функцій належності) прискорення часу виконання цих алгоритмів – у 5,9 разів на 8 процесорах, у 10,25 разів на

16 процесорах та у 16,4 разів на 32 процесорах при застосуванні регулярної декомпозиції на вихідному масиві даних у порівнянні з однопроцесорними варіантами алгоритму (рис. 7).

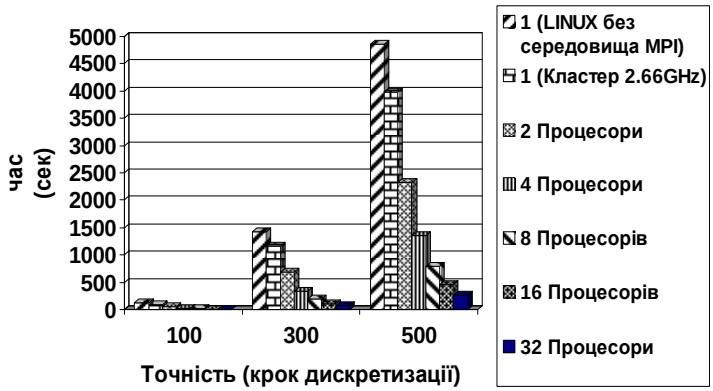

Рис. 7. Часові характеристики паралельних алгоритмів нечіткої байєсівської класифікації

Отримані результати свідчать про виконання закону Амдала [4] для паралельних систем з масовим паралелізмом. Для таких систем непаралельна частина коду утворюється за рахунок операторів, виконання яких дублюється усіма процесорами. Оскільки реальне прискорення складає приблизно 1,7 рази при збільшенні кількості процесорів удвічі, із таблиці 1 випливає, що доля послідовних обчислень у реалізованому алгоритмі класифікації складає приблизно 20 %, а доля паралельних обчислень – 80 %.

Програмна реалізація інформаційної технології є високонадійною, вона логічно завершується та видає результати при будь-яких допустимих значеннях лінгвістичних змінних суб'єктивних факторів визначеності симптомів. Дана технологія крім поточного стану дозволяє оцінити також вірогідність прогнозних станів економічної системи.

1. *Сергієнко І.В., Коваль В.М., Сав'як В.В.* Сучасні високошвидкісні обчислювальні системи. – http://www.ustar.com.ua.
2. *Веревка О.В., Парасюк И.Н.* Математические основы построения нечетких байесовских механизмов вывода // Кибернетика и системный анализ. – 2002. – № 1. – С. 105–117.
3. *Веревка О.В., Заложенкова И.А., Парасюк И.Н.* Байесовские процедуры диагностики для различных типов неопределенности информации // Проблемы программирования. – 2002. – № 1-2. – С. 328–333.
4. *Немнюгин С.А.* Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.
5. *Хьюз К., Хьюз Т.* Параллельное и распределенное программирование на C++.: Пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 672 с.
6. *Богачев К.Ю.* Основы параллельного программирования. – М.: БИНОМ. Лаборатория базовых знаний, 2003. – 342 с.
7. *MPI* Forum. - http://www.mpi-forum.org.
8. *Парасюк И.Н.* Об одном алгоритме асинхронного управления вычислительным процессом в ППП на многопроцессорных вычислительных системах // Пакеты прикладних программ и численные методы. – Киев: Институт кибернетики им. В.М.Глушкова АН УССР, 1988. – С. 51–55.
9. *Дорошенко А.Е., Рухлис К.А.* Параллельное программирование задач визуализации научных данных // Проблемы программирования. – 2004. – № 2-3. – С. 285–295.
10. *Плаугер П., Степанов А., Ли М., Массер Д.* STL – стандартная библиотека шаблонов C++: Пер. с англ. – СПб: "БХВ-СПб", 2004. – 656 с.
11. *Zadeh L.A.* Fuzzy Probabilities and Their Role Decision Analysis // Proc. of. IFAC Symp. "Theory and Appl. Of Digital Control". – New Delhi: IFAC, 1982. – P. 15–21.
12. *Прикладные* нечеткие системы: Пер. с япон. / Под ред. Т.Тэрано, К.Асан, М.Сугэно. – М.: Мир, 1993. – 366 с.