УДК 004.3

**Бойчук В.О.,**
кандидат технічних наук, доцент
**Муляр І. В.,**
кандидат технічних наук, доцент
**Царьов Ю. О.,**
кандидат психологічних наук, с.н.с.

# БІОКОМП'ЮТЕРИ ТА МЕТОДИКИ ЇХ ПРОГРАМУВАННЯ

*Розглянуто традиційні моделі програмування для паралельних систем та перспективи розвитку паралельних обчислювальних систем. На основі класифікації сучасних паралельних систем показані недоліки сучасних підходів паралельного програмування. Розглянуто сучасні паралельні системи, які створені на основі запозичення принципів функціонування з біології. Проведений детальний огляд методик їх програмування. Зроблено висновки про їх недоліки й запропоновано принципи розробки паралельних програм для таких обчислювальних систем.*

***Ключові слова:*** *паралельні системи, дрібнозернистий паралелізм, клітинні комп'ютери, клітинний автомат, аморфні обчислення.*

*В статье рассмотрены традиционные модели программирования для параллельных систем и перспективы развития параллельных вычислительных систем. На основе классификации современных параллельных систем показаны недостатки современных подходов параллельного программирования. Рассмотрены современные параллельные системы, которые созданы на основе заимствования принципов функционирования из биологии. Проведен детальный обзор методик их программирования. Сделаны выводы об их недостатках и предложены принципы разработки параллельных программ для таких вычислительных систем*

***Ключевые слова:*** *параллельные системы, мелкозернистый параллелизм, клеточные компьютеры, клеточный автомат, аморфные вычисления.*

*Paper describes traditional programming models for parallel systems and the prospects for the parallel computing systems development. On the basis of the classification of modern parallel systems several lacks of modern approaches to parallel programming are shown. Modern parallel systems , which are based on the functioning principles borrowing from biology, are described. A detailed overview of the methods of their programming was done. The principles of the parallel programs development for these computing systems were proposed.*

***Keywords:*** *parallel systems, finegrained parallelism, cellular computers, cellular automata, amorphous computing.*

Традиційно у програмуванні для паралельних систем існують дві моделі паралелізму: із загальною пам'яттю й заснована на обміні повідомленнями.

Більшість існуючих мов використовують модель із загальною пам'яттю. Це обумовлюється переважаючою на сьогодні архітектурою комп'ютерів. Деякі мови побудовані на основі обміну повідомленнями, до цієї групи відносяться Occam, Erlag і Oz. У моделі паралелізму, заснованій на обміні повідомленнями, постулюється: загальної пам'яті немає, всі обчислення повинні вироблятися в ізольованих процесах. Єдиний спосіб обміну інформацією – асинхронний обмін повідомленнями. Такий підхід позбавляє розробника важко виявлюваних помилок: взаємних блокувань і станів гонок. Крім цього, обмін повідомленнями між процесами дає додаткові можливості для збільшення надійності та масштабованості розроблюваних систем.

Природа – живий приклад складної розподіленої паралельної системи, яка використовує принципи обміну повідомленнями. Мурашник, бджолиний рій, клітини організму є зразками систем, де безліч досить простих організмів колективно вирішують складні завдання, не вдаючись при цьому до використання загальної пам'яті. Інженерам вже зараз доводиться вирішувати подібні завдання, а з розвитком штучного інтелекту, сенсорних мереж, нанотехнологій їхня частка буде тільки збільшуватися.

Як видно з цих наочних прикладів, природні паралельні системи, хоча і використовують механізми обміну повідомленнями, можуть мати ступінь паралелізму куди більш високий, ніж сучасні і плановані багатоядерні процесори.

І тому вже зараз стають актуальними задачі створення паралельних комп'ютерних розподілених систем, що не підходять під існуючі шаблони і створення парадигм програмування для цих систем.

Для цього паралельні обчислювальні системи треба класифікувати за обчислювальними елементами (процесорами) і зв'язками (комунікаціями) між цими елементами.

Паралельні системи за обчислювальними елементами (процесорами) можна класифікувати наступним чином:

• зі складними обчислювальними елементами з великою кількістю виконуваних операцій. Наприклад, до таких елементів можна віднести ядра стандартних багатоядерних процесорів ;

• з елементарними обчислювальними елементами – з обмеженим числом виконуваних операцій, але дешевих у виробництві;

• з невеликою кількістю обчислювальних елементів у системі, менше 1000;

• з великою кількістю обчислювальних елементів у системі, більше 1000;

• зі стабільною кількістю обчислювальних елементів у системі;

• з динамічно змінюваною кількістю обчислювальних елементів у системі;

Можна виділити наступні основні типи комунікацій в паралельних обчислювальних системах :

• локальні – кожен елемент пов'язаний із невеликим набором інших елементів;

• глобальні – кожен елемент пов'язаний із великим числом інших елементів;

• структуровані – елементи системи утворюють регулярну структуру (наприклад, з топологією решітки);

• неструктуровані – елементи пов'язані довільним графом ;

• статичні – схема комунікацій не змінюється з плином часу;

• динамічні – схема комунікацій змінюється в процесі виконання програми;

• синхронні – відправник і одержувач даних координують обмін ;

• асинхронні – обмін даними не координується .

Стандартні широко використовувані обчислювальні системи, як правило, містять невелику кількість обчислювальних простих елементів, наприклад багатоядерні процесори, де кількість ядер не перевищує 100. Кількість їх стабільна. Комунікації між цими елементами глобальні, як правило, через загальну пам'ять, структуровані і статичні.

І, навпаки, сучасні паралельні системи можуть містити величезну кількість (до декількох мільйонів) обчислювальних простих елементів, кількість яких може динамічно змінюватись, як через вихід із ладу, так і з інших причин. Комунікації між цими елементами локальні, неструктуровані, асинхронні й динамічні.

Для функціонування цих систем відповідно можуть бути використані такі терміни:

– великоблочний паралелізм властивий обчислювальним системам, складеним з невеликого числа (десятки, сотні) відносно складних процесорних елементів, з'єднаних статичною, структурованою мережею зв'язку того чи іншого виду.

– дрібнозернистий паралелізм властивий обчислювальним системам, складеним із величезного числа (десятків, сотень тисяч) відносно простих процесорних елементів. Зв'язки між процесорними елементами нерегулярні і дуже часто організовані за принципом близькодії. Як правило, такі системи вузькоспеціалізовані. Термін "дрібнозернистий паралелізм" говорить про елементарність і швидкоплинність окремої обчислювальної дії. Характерною рисою дрібнозернистого паралелізму є велика і приблизно однакова інтенсивність паралельних обчислень та обміну інформацією.

Розглянемо характерні риси деяких із реалізованих паралельних систем дрібнозернистого паралелізму на основі запозичення принципів функціонування з біології і їх методики програмування [1].

Клітинні комп'ютери являють собою колонії різних мікроорганізмів, що самоорганізуються, в геном яких вдалося включити певну логічну схему, котра могла б активізуватися за присутності певної речовини [2]. Для цієї мети ідеально підійшли б бактерії, ємність з ними і була б біокомп'ютером. Такі комп'ютери дешеві у виробництві.

Головна властивість такого комп'ютера полягає в тому, що кожна його клітина являє собою мініатюрну хімічну лабораторію. Якщо біоорганізм запрограмований, то він просто виробляє потрібні речовини. Досить виростити одну клітку, що володіє заданими якостями, і можна легко і швидко виростити тисячі клітин з такою ж програмою.

Однак поряд з перевагами клітинні комп'ютери мають і суттєві недоліки, такі як:

– організація всіх клітин в єдину працюючу систему;

– складнощі зі зчитуванням результатів;

– низька точність обчислень, пов'язана з виникненням мутацій, прилипанням молекул до стінок судин і т.д.;

– неможливість тривалого зберігання результатів обчислень у зв'язку з розпадом ДНК протягом часу.

При програмуванні клітинних комп'ютери можуть бути використані клітинні автомати, які були запропоновані фон Нейманом як універсальна обчислювальна

модель паралельних обчислень. Клітинний автомат – дискретна динамічна система, що є сукупністю однакових клітин, однаковим чином з'єднаних між собою. Усі клітини утворюють решітку клітинного автомата. Решітки можуть бути різних типів, відрізняючись як за розмірністю, так і за формою клітин. Кожна клітина є кінцевим автоматом, стан якого визначаються станами сусідніх клітин і її власним станом.

При програмуванні клітинних автоматів використовують такі їх властивості.

1. Зміни значень всіх клітин відбуваються одночасно після визначення нового стану кожної клітини решітки.

2. Решітка однорідна.

3. Взаємодії локальні. Лише, як правило, сусідні клітини здатні вплинути на дану клітку.

4. Множина станів клітини кінцева.

У більшості випадків клітинний автомат є лише математичним уявленням, яке дозволяє у простий чисельний спосіб вирішувати завдання визначення поведінки складних систем, що складаються з дуже великої кількості однакових елементів, що взаємодіють за простими законами. Також обмеженням цієї моделі є те, що вона використовує структуровані і статичні комунікації між клітинами.

Ще одна модель дрібнозернистого паралелізму – аморфні обчислення з'явилися як область досліджень в середині 1990-х, на основі таких чинників [3]:

розвитку моделей клітинних автоматів;

перспективи отримання майже безкоштовних елементарних процесорів у великих кількостях.

Мета аморфних обчислень полягає в тому, щоб визначити організаційні принципи і створити технології програмування для отримання заздалегідь заданої поведінки зі співпраці великої кількості ненадійних елементів, які розміщені невідомим і змінним в часі способом. Важливість аморфних обчислень сьогодні пов'язана з появою нових технологій, які можуть слугувати основою для систем обробки інформації величезної потужності за дуже низьку ціну, якби тільки можна було б оволодіти методами їх програмування.

Аморфні комп'ютери мають наступні властивості:

1. Окремі процесори ідентичні і виробляються масово. Це дозволяє дешево виготовляти велику кількість процесорів. Кожен процесор повинен мати генератор випадкових чисел, щоб відрізняти себе від інших.

2. Процесори не мають ніякої початкової інформації про своє розташування, орієнтацію в просторі чи про особливості своїх сусідів. Процесори повинні самі виявляти своє розташування і позицію.

3. Процесори працюють асинхронно й мають однакову тактову частоту. Аморфний комп'ютер не допускає синхронізацію процесорів, тому що може бути важко чи неефективно забезпечувати синхронізацію в певних фізичних середовищах.

4. Всі процесори запрограмовані однаково, хоча в кожного елемента є засоби для того, щоб зберігати поточний стан і генерувати випадкові числа. Можуть також бути особливі елементи, які були встановлені у специфічний стан.

5. Процесори розміщені щільно, але випадково. Випадкове розміщення рівномірне.

6. Процесори ненадійні. Для того, щоб виготовляти велику кількість процесорів дешево, неможливо тестувати кожен окремий процесор. Таким чином, певна кількість процесорів неминуче не будуть працювати. Безвідмовність досягається через надмірність.

7. Процесори зв'язуються тільки локально й не мають постійного зв'язку. Процесори повинні зв'язуватися з фізично близькими сусідами через локальний механізм передачі повідомлень, хоча конкретний механізм залежить від основи на якій розміщені процесори. Радіус зв'язку передбачається набагато меншим, ніж весь простір, зайнятий аморфним комп'ютером.

8. Комунікація передбачається ненадійною й у відправника немає жодної гарантії, що повідомлення буде отримане. У електронному аморфному комп'ютері елементи могли б спілкуватися за допомогою радіохвиль короткої дії, тоді як біоінженерні клітини могли б спілкуватися за допомогою хімічних сигналів.

9. Кожен елемент має невелику обчислювальну потужність і невеликий об'єм пам'яті. Елементи можуть містити датчики й пересувні механізми й у деяких випадках можуть бути повністю мобільними.

Крім того, при розгляді функціонування таких комп'ютерів часто вводяться наступні обмеження щодо фізичних характеристик аморфного комп'ютера.

1. Аморфний комп'ютер існує на плоскій двовимірній поверхні. Це припущення зроблене для спрощення аналізу.

2. Модель зв'язку припускає, що всі процесори мають обмежений радіус передачі інформації приблизно одного й того ж самого встановленого розміру й спільно використовують єдиний канал. Як наслідок, можуть відбуватися колізії, коли два процесори, області передачі інформації яких перекриваються, посилають повідомлення одночасно. Одержувач може виявити колізію, а відправник – ні (рис. 1).

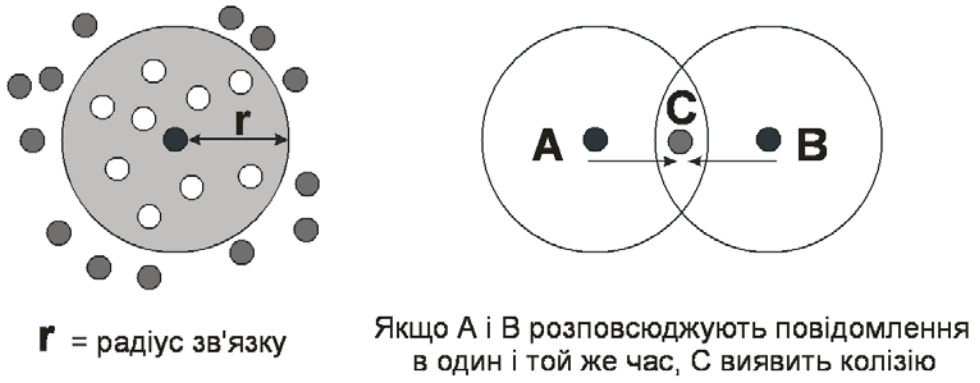


Рис. 1. Взаємодія елементів аморфного комп'ютера

Припускається, що число елементів може бути дуже великим (порядку $10^6$ – $10^{12}$). Алгоритми, розроблені для аморфних комп'ютерів, мають бути відносно незалежними від числа частинок, продуктивність повинна погіршуватись плавно, зі зменшенням кількості частинок. Таким чином, весь аморфний комп'ютер може бути розцінений як обчислювальна система з масовим паралелізмом і попередні дослідження в обчисленнях з масовим паралелізмом, таких як дослідження клітинних автоматів, є одним із джерел ідей для розвитку аморфних комп'ютерів.

Аморфні обчислення відрізняються від досліджень у клітинних автоматах, тому що аморфні механізми мають бути незалежними від конфігурації, ненадійними і асинхронними.

Першим кроком до реалізації аморфних комп'ютерів стала технологія мікромеханічного виготовлення електронних компонентів, яка була розроблена ще в минулому десятилітті. Вона інтегрує логічні канали, мікродатчики, виконавчі механізми і мережі комунікації на однокристальній схемі. Ці компоненти можуть бути виготовлені надзвичайно дешево, за умови, що не всі чіпи повинні працювати коректно й що немає жодної потреби упорядкувати ці мікросхеми в точні геометричні форми або встановлювати точні з'єднання між ними. Дослідники уявляють елементи "розумного пилу", достатньо малі, щоб переноситись повітряними потоками, і які б формували хмари сенсорних частинок, здатних обмінюватись інформацією. Такі хмари датчиків досить важко реалізувати, але мережі між частинками міліметрового масштабу тепер комерційно доступні для засобів контролю над середовищем.

Одна з відомих методик для програмування аморфних комп'ютерів використовує принцип дифузії. Деякий елемент (вибраний деяким випадковим чином) розповсюджує повідомлення. Це повідомлення, отримане кожним з її сусідів, роздається їхнім сусідам, і так далі – для утворення хвилі, яка розповсюджується по всій системі. Повідомлення має лічильник, і кожна частинка зберігає його значення і збільшує його перед передачею. Щойно елемент зберігає своє значення лічильника, він припиняє передавати повідомлення й ігнорує наступні повідомлення. Така хвиля дає кожному елементу оцінку про відстань від джерела. Вона може також утворювати регіони керованого розміру, передаючи повідомлення тільки тоді, коли значення лічильника більше заданого порогу. Дві такі лічильні хвилі можуть бути об'єднані, щоб визначити ланцюжок частинок між двома заданими елементами. Мотивом для такої методики послужила хімічна градієнтна дифузія, яка є основоположним механізмом в біології.

Для програмування аморфних комп'ютерів була розроблена мова зростаючої точки (GPL) (Coore's Growing Point Language [4]. Використання цієї мови демонструє, що аморфний комп'ютер може генерувати досить складні шаблони. Прикладом може слугувати шаблон, що представляє структуру з'єднання довільного електричного ланцюга, як показано на рис. 2.

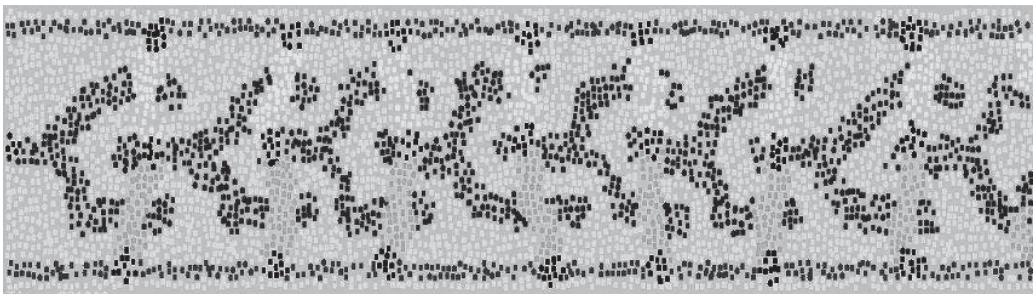

Рис. 2. Приклад з'єднання довільного електричного ланцюга, утвореного мовою зростаючої точки

За основу GPL був взятий принцип з ботаніки, який заснований на зростаючій точці і тропізмі. Зростаюча точка – місце діяльності в аморфному ком-

п'ютері. Зростаюча точка поширюється через комп'ютер, передаючи її діяльність від одного обчислювального елементу до сусіднього. Коли зростаюча точка проходить через комп'ютер, це проводить диференціювання поведінок частинок, через які вона проходить. Частинки виділяють "хімічні" сигнали, лічильні хвилі яких визначають градієнти, і вони привертають або відвертають зростаючі точки, напрямок яких визначений програмним тропізмом. Цих механізмів достатньо, щоб аморфні комп'ютери могли генерувати будь-які довільні, наперед визначені, структурні шаблони за певною топологією. Проте, на відміну від реальної біології, щойно шаблон був створений, немає жодного чіткого механізму для підтримки його за змін матеріалу. Крім того, із програмної точки зору немає жодного чіткого способу скласти форми, складаючи зростаючі точки. Пізніше було показано, як GPL може бути розширена, щоб утворювати довільно великі шаблони, такі як текстові рядки.

Розглянувши особливості обчислювальних систем із дрібнозернистим паралелизмом, можна визначити низку загальних проблем.

1. Досить важко забезпечити впорядковане функціонування обчислювальних елементів, кількість яких може досягати декількох мільйонів. Як видно з аморфних комп'ютерів, централізоване керування такими наборами елементів намагаються здійснювати через принципи випадкової локальної взаємодії, використовуючи механізм градієнтів, імітуючи взаємодію колоній найпростіших організмів у біології. Це досить повільно і неефективно.

2. Передбачається, що кожен обчислювальний елемент в біокомп'ютерах виконує порівняно прості стандартні операції на основі локальної взаємодії. А це також обмежує обчислювальні можливості й універсальність таких систем.

3. Вибіркове програмування потрібних обчислювальних елементів або групи елементів неможливе через ускладнений доступ до цих елементів й часто є недоцільним через те, що вони не можуть виконувати складні операції.

Відповідно, для ефективного програмування біокомп'ютерів впроваджувати нові методології програмування, які б дозволяли вибірково програмувати й керувати конкретними обчислювальними елементами або їх групами. Також обчислювальні елементи мають виконувати не тільки елементарні локальні операції. Якщо використати аналогію біологічного багатоклітинного організму, окремі клітини мають надскладну структуру і функції. Також вони існують не тільки локально взаємодіючи із сусідами, але керуються нервовою і гуморальною системою. Це дає змогу їм функціонувати узгоджено йвиконувати потрібні функції.

Таким чином, на сучасному етапі розвитку паралельних систем і паралельного програмування постала необхідність у розробці нової методики програмування. Основною рисою цієї методики є комбінування випадкових локальних взаємодій множин обчислювальних елементів із централізованим керуванням ними.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. L. Edelstein-Keshet Mathemathical Models in Biology. McGraw Hills, New-York,1988.

2. *Богданов А.* Способы организации высокопроизводительных процессоров. Клеточные и ДНК-процессоры. Коммуникационные процессоры / А. Богданов [Електронний ресурс]. – Режим доступу : http : // www.intuit.ru/studies/courses/45/45/lecture/1348.

3. Abelson, H.; Allen, D.; Coore, D.; Hanson, C.; Homsy, G.;Knight, T.; Nagpal, R.; Rauch, E.; Sussman, G.; and Weiss, R. 2000. Amorphous computing. Communications of the ACM 43(5).

5. Alyssa Morgan, Daniel Coore: Extending the growing point language to self-organise patterns in three dimensions. GECCO (Companion) 2013: 109–110.

Отримано 18.10.2013.