Моделе-орієнтоване програмування нечітких агентів для високопродуктивних обчислювальних систем

І.М.Парасюк[1], С.В.Єршов[1]

[1] *Інститут кібернетики ім.В.М.Глушкова НАН України, просп. Академіка Глушкова 40, Київ ,Україна*

ivpar1@i.com.ua, sershv@ukr.net

Анотація. *Представлено новий трансформаційний підхід, який застосовується при моделе-орієнтовному програмуванні нечітких агентів для високопродуктивних обчислювальних систем. Для зменшення витрат на комунікацію між агентами запропоновано метод, що ґрунтується на генерації віртуальних блоків агентів над ієрархією типів пам'яті кластерної системи і враховує основні особливості платформ реалізації, таких як CUDA та MPI. Розроблені основні перетворення, що дозволяють згенерувати нечіткі платформно-незалежні моделі внутрішнього агентного управління та пов'язані з ними рольові моделі для високопродуктивних обчислювальних систем. В результаті, початкові моделі збагачують інформацією, що поступово приводить до реалізації високопродуктивної мультиагентної системи. Даний підхід використовувався для розробки мультиагентної системи моделювання трансферу (купівля/продаж) матеріальних ресурсів в ринкових умовах.*

Ключові слова

Моделе-орієнтоване програмування, нечіткі агенти, кластерна система, MPI, CUDA.

1 Вступ

В даний час мультиагентні системи являють собою один з найбільш складних видів програмного забезпечення для високопродуктивних обчислювальних систем, розробка яких ускладнюється у зв'язку з: 1) відсутністю єдиної метамоделі, в рамках якої описується функціонування агентів (наприклад, розглядаються деліберативні, реактивні та інші види агентів); 2) відсутністю загальноприйнятих "стандартних" платформ і мов реалізації. Агентно-орієнтоване моделювання процесів в найбільш складних системах (соціальних, економічних тощо) вимагає одночасного виконання до 10^{13} програм-агентів [1]. Програмування інтелектуальних агентів для сучасних кластерних систем значно ускладнюється необхідністю враховувати не тільки функціональні вимоги, що ґрунтуються на паралельному виконанні алгоритмів прийняття рішень агентами, але й особливості парадигми та специфічні деталі програмного забезпечення технологічної платформи паралельного виконання та комунікації в середовищі високопродуктивної системи такої як Message Passing Interface (MPI), Common Unified Data Architecture (CUDA) тощо.

Для подолання вищезазначених проблем нами запропонований моделе-орієнтований підхід до програмування та розробки мультиагентних систем, що функціонують в складі сучасних та перспективних кластерних систем, зокрема СКІТ-3 [2]. Моделе-орієнтоване (трансформаційне, генеративне) програмування ґрунтується на поняттях платформно-незалежної і платформно-залежної моделей (ПНМ, PIM, platform-independent та ПЗМ, PSM, platform-specific model). ПНМ визначені на вищому рівні абстракції, ніж ПЗМ. Незалежна від платформи модель – представлення системи з незалежної від платформи точки зору, в той час як ПЗМ визначена як вид системи з точки зору певної платформи програмування. Використання цих понять дозволяє позбутися технологічних і технічних деталей, які несуттєві для фундаментальної функціональності мультиагентної системи. Полегшується перенесення програмного забезпечення на іншу перспективну платформу і його модифікація, так як при цьому можна використовувати стару платформно-незалежну модель і розробляти знову тільки платформно-залежну.

Головна перевага моделе-орієнтованого підходу полягає в тому, що з його допомогою можна значно прискорити розробку мультиагентних систем для кластерних ЕОМ, незважаючи на те, що потрібно створити

кілька моделей замість однієї. Це досягається за рахунок автоматизованої генерації платформно-залежної моделі за платформно-незалежною за допомогою спеціально розроблених трансформацій (які виражаються як набір правил). Тому процес переходу до програмного коду мультиагентних систем, що ґрунтується на конкретній платформі кластерної системи, може бути формалізований за допомогою таких трансформацій.

Мета цієї доповіді – дослідження і обґрунтування трансформаційного підходу до розробки інтелектуальних агентів з використанням нечітких моделей для високопродуктивних середовищ кластерних систем, розробка трансформаційних засобів для специфікації нечітких агентів, і вивчення архітектурних особливостей нечітких агентів, що породжуються для перспективних платформ. Зазначені проблеми тематично вписуються у подальший розвиток досліджень в напрямку становлення моделе-орієнтованого підходу до розробки програмного забезпечення високопродуктивних мультиагентних систем [3–5].

2 Платформа виконання агентів для кластерної системи СКІТ

В якості цільової платформи виконання згенерованих систем використовується кластерна система СКІТ-3 [2]. Для підтримки взаємодії інтелектуальних агентів, була створена спеціальна бібліотека, що включає ряд шаблонів, класів і методів для організації поведінки агентів – циклічного, одноразового та багаторазового, а також для обміну різними типами асинхронних повідомлень з використанням MPI у якості носія. Варіант бібліотеки реалізовано для використання можливостей програмно-апаратного стеку CUDA. Бібліотека надає можливість організувати модель системи як віртуальну сітку (grid) агентів, в якій агентам дозволяється взаємодіяти тільки на обмежену дистанцію (кількість кроків за сіткою). Через розподіл глобального стану між процесорами, частина стану окремих агентів може зберігатися в сусідніх агентів. При циклічному виконанні алгоритмів нечіткого виведення елементи цього стану можуть запитуватися до початку виконання наступної ітерації. Спеціальний примітив використовується для бар'єрної синхронізації виконання ітерацій (кроків виконання) нечітких агентів. Однак, такий підхід не враховує різноманітності типів пам'яті та час затримки передачі повідомлень між агентами, які використовують такі типи пам'яті. Тому, для зменшення витрат на комунікацію нами запропоновано метод, що ґрунтується на віртуальній сітці агентів над ієрархією типів пам'яті агентів.

Віртуальна сітка агентів розбивається на блоки агентів, що закріплені за окремими елементами обробки (CPU, GPU). Кожний такий блок складається з $L \times L$ агентів, що оточені E шарами, що представляють агенти у сусідніх блоках. Завдяки цьому, обчислення у межах локального блоку можуть бути подовжені на E ітерацій, після чого потрібна передача даних між блоками. Оскільки блок розміру $L + 2E$ містить усю інформацію, яка необхідна для локальних обчислень, це спричиняє зменшення кількості обмінів у сітці. Витрати на комунікацію після E ітерацій можуть бути представлені по-різному для CPU та GPU відповідно $C_{CPU} = c_w(L^2 - (L-2E)^2) + c_r((L+2R)^2 - L^2)$, $C_{GPU} = c_w(L^2) + c_r((L+2R)^2 - L^2)$, де c_w, c_r - витрати часу на читання (запис) даних одного агента в середньому.

При даному підході на найнижчому рівні ієрархії розташовані агенти, що виконуються як потоки CUDA. Навіть на цьому рівні схема зменшення витрат на комунікацію досягається завдяки використанню блоків агентів $L + 2E$ у якості блоків CUDA. Вище знаходяться блоки агентів на окремих GPU, синхронізація між якими здійснюється при послідовному виконанні ядер CUDA. На найвищому рівні найбільші блоки агентів розподіляються між вершинами, комунікація між якими здійснюється на основі MPI. Після E_{GPU} ітерацій, для блоків агентів з відповідними рангами здійснюються неблокуючі виклики MPI_Irecvs та MPI_Isends, що оновлюють стани $(L + 2R)^2 - L^2$ агентів, після чого їх виконання продовжується.

3 Основні моделі та трансформації

Основні процеси моделе-орієнтованого програмування – це розробка (виявлення та аналіз) вимог, проектування агентів і реалізація агентів мультиагентної системи. Значення окремих моделей інтелектуальних агентів не однаково для різних процесів. При цьому на етапах розробки вимог і проектування створюються основні платформно-незалежні моделі: початкова модель нечіткого агента, початкова рольова модель, уточнена рольова модель і модель внутрішнього агентного управління (МВУ). Платформно-залежні моделі містять моделі нечіткого агента, які відповідають усім деталям текстового представлення. Природно, що такі моделі є об'єктом процесів конструювання та тестування. В даний час такі моделі розроблені для таких платформ як FCL [6], JADE і для розробленої нами бібліотеки класів нечітких агентів у середовищі суперкомп'ютера СКІТ-3 [2].

Метамоделі, моделі і трансформації, що застосовуються в процесах моделе-орієнтованої розробки нечітких мультиагентних систем, описані на основі фреймворку Eclipse Modeling Framework, який використовує для

побудови моделей метамодель вищого рівня Ecore. Засоби опису метамоделей Ecore призначені для підтримки часу виконання, такої як повідомлення про зміни, засоби для збереження моделей на основі стандарту XMI (XML Metadata Interchange), і ефективний інтерфейс для операцій над об'єктами Ecore. Аналогічна технологія – Meta-Object Facility (MOF). Однак, метамодель EMF простіша з точки зору внутрішньої структури.

Початковою при трансформаційної розробці агентів може служити платформно-незалежна модель нечітких правил. Для представлення системи нечітких правил розроблена метамодель Ecore, що містить всі необхідні поняття для конструювання систем нечітких правил виводу, таких як системи Мамдані або системи Такаги–Сугено. В якості основних класів (понять) обрані нечітке правило, нечітка змінна, нечітке значення, нечітка множина і пов'язаний з нею лінгвістичний терм. Кожне правило містить набір антецедентів і консеквентів, яким відповідають нечіткі значення, пов'язані в свою чергу з нечіткими змінними, які описують набір допустимих лінгвістичних значень та їх функцій належності. Різні способи виконання нечітких правил враховуються за допомогою конкретних класів FuzzyRule-Executor. Це дозволяє розширювати дану метамодель іншими видами нечітких систем, не зачіпаючи при цьому інші моделі.

Однак, задавати початкову модель правил в текстовому вигляді або базовими засобами дерева об'єктів Ecore було б не наочно і ускладнювало б її сприйняття. Для введення і редагування таких моделей створений спеціальний графічний редактор з засобами навігації за нечіткими правилами і композиції правил заміни (видалення і додавання нечітких змінних, множин, функцій належності, включення змінних в антецеденти і консеквенти правил тощо). Створення такого редактора проводилося в рамках методології предметно-орієнтованого моделювання (domain-specific modeling), що ґрунтується на систематичному використанні візуальних (заснованих на графах) предметно-орієнтованих мов моделювання для представлення різних аспектів системи. Абстрактний синтаксис для моделювання нечітких агентів задається метамоделлю в системі EMF. Для опису конкретного візуального синтаксису діаграм використовувався фреймворк GMF (Graphical Modeling Framework). На його основі породжується повний код програми-редактора діаграм.

Визначення правил трансформацій, що відображає моделі, що створюються за допомогою згенерованого редактора, здійснювалося з використанням мови Query-View-Transformation (QVT), яка розроблена під керівництвом OMG. Хоча стандартне визначення QVT призначено для відображень над елементами MOF-сумісних метамоделей, реалізація, яка застосовувана в Eclipse, заснована на метамоделі Ecore. QVT визначає не одну, а три предметно-орієнтованих мови: декларативну Relations, ядро Core і мову відображень OperationalMappings. В даний час саме QVT/OperationalMappings забезпечує найбільшу практичну пропрацьованість і найбільший ступінь сумісності з концепціями в рамках моделе-орієнтованої архітектури.

4 Рольові моделі та моделі внутрішнього агентного управління

В процесі розробки вимог також визначається початкова рольова модель (РМ) агентів. Передбачається що інтелектуальний агент зобов'язується виконати одну або кілька ролей в перебігу свого життєвого циклу. Однією з особливостей розробки агентів є те, що види діяльності агента пов'язані із роллю, а не з системою. Крім того, після визначення можливостей агентів і розкладання їх на прості діяльності, необхідно визначити динамічну композицію з цих видів діяльності за кожною роллю таким чином, що агент досягає поставленої мети.

В процесі проектування генеруються додаткові ролі агента, що уточнюють початкову рольову модель. У загальній ситуації, відповідно до шаблона проектування постачальник/споживач для кожної нечіткої змінної, що входить в початкову модель правил, породжуються дві додаткові ролі. Визначені раніше ролі зв'язуються з вибраними типами агентів, що складають проектовану мультиагентну систему і породжують окремі (типізовані) екземпляри агентів. При необхідності розміщення агентів на одній вершині для економії комунікаційних ресурсів їх типи можуть бути суміщені, що призводить до підсумовування (накладання) ролей, які вони мають виконувати. В процесі проектування до уточненої рольової моделі застосовуються трансформації, що дозволяють отримати специфікації моделей внутрішнього агентного управління – по одній для кожної згенерованої ролі. Для розробки такої трансформації модель-модель (M2M) використано мову QVT/OperationalMappings.

Багато методологій розробки (як Tropos або INGENIAS) нав'язують специфічні ментальні моделі агента. На відміну від цього, МВУ ґрунтується на нечітких схемах станів і не робить ніяких додаткових припущень стосовно моделі комунікації, процесу виведення чи ментального стану (переконання-бажання-наміри) агентів.

Роботою окремих агентів можна управляти з урахуванням ступеня досяжності таких "розмитих" станів. МВУ задається нечіткою схемою переходів, яка представлена нечітким мультиграфом $G = (Q, T, \mu_G, \Sigma)$, де $Q = \{q_1, q_2, ..., q_n\}$ – множина станів (вершин графа), де q_1 – початковий стан; $T = \{t_1, t_2, ..., t_n\}$ – множина

переходів (дуг) графа; $\mu_G : T \to [0,1]$ – функція належності переходів даній схемі; $\Sigma : T \to A$ – функція розмітки переходів подіями з деякого кінцевого алфавіту подій (повідомлень) *A*.

5 Генерація коду мультиагентної системи

Породження коду мультиагентної системи складається з двох трансформацій – виконання ряду перетворень модель-модель з метою отримання платформно-специфічної моделі нечітких агентів і запуску подальших трансформацій "модель-код" для отримання тексту програми цільового агента.

FCL – єдино стандартизована предметно-орієнтована мова для специфікації систем нечіткого логічного виведення [6]. Її основна перевага – наочність і простота, а також наявність переносимих реалізацій (у вигляді інтерпретаторів) мовами Java і C++. Однак, оскільки FCL не забезпечує всієї функціональності, необхідної для функціонування агентів, породжується код мовою C++ для кластерної платформи, який містить виклики інтерпретатора FCL. Більшість відомих платформ (Jason, 3APL, JACK тощо) мультиагентних систем підтримує тільки архітектуру BDI (Belief-Desire-Intention), яка втілює один з основних видів деліберативних агентів і містить явно представлену структуру даних, що відповідає цим трьом зазначеним властивостям міркувань, які застосовуються агентом при вирішенні задач. Привабливість BDI-підходу знижується, оскільки був реалізований ряд інших моделей (реактивні, мотивовані), що дозволяють усунути проблеми BDI-агентів [3].

Трансформація МВУ в код агента для цільової платформи – перетворення "модель-текст", що створює клас агента і кілька класів Behaviour (Поведінка) для кожної моделі МВУ. Особливістю функціонування породжуваних агентів для CKIT є те, що на відміну від JADE не використовується мова ACL (агенти на кластері не взаємодіють з іншими відкритими платформами), замість цього генерується код, що виконує маршалінг/демаршалінг основних типів повідомлень. Також, зв'язування сервісів з агентами-постачальниками виконується під час компіляції, а не динамічно, що дозволяє скоротити витрати на обмін повідомленнями.

Для генерації текстових файлів програм-агентів розроблені перетворення "модель-текст" на основі текстових шаблонів платформи XPand, що запускаються в контексті компонентів EMFT Workflow. Такі перетворення генерують методи поведінки агента для отримання і відправки повідомлень і композитні поведінки нечіткої схеми переходів, які координують виконання простих поводжень.

6 Висновки

Таким чином, нами представлено новий трансформаційний підхід, який застосовується при моделе-орієнтовному програмуванні нечітких агентів для високопродуктивних обчислювальних систем. Для зменшення витрат на комунікацію між агентами нами запропоновано метод, що ґрунтується на генерації віртуальних блоків агентів над ієрархією типів пам'яті кластерної системи. Дотримуючись моделе-орієнтованого підходу, розроблені основні перетворення, що дозволяють згенерувати нечіткі платформно-незалежні моделі внутрішнього агентного управління та пов'язані з ними рольові моделі для високопродуктивних обчислювальних систем. Важливо, що запропонований підхід враховує основні особливості програмно-апаратних платформ, таких як CUDA та MPI. В результаті, початкові моделі збагачують інформацією, що поступово приводить до реалізації високопродуктивної мультиагентної системи. Даний підхід використовувався для розробки мультиагентної системи моделювання трансферу (купівля/продаж) матеріальних ресурсів в ринкових умовах, що включає паралельне виконання 10^4 агентів.

Посилання

[1] Macal C., North M. Agent-based modeling and simulation for exascale computing. – SciDAC Review, Summer 2008. – P. 34–41.
[2] Суперкомпьютеры ИК НАН Украины. – http://icybcluster.org.ua.
[3] Парасюк И.Н., Ершов С.В. Нечеткие модели мультиагентных систем в распределенной среде // Проблеми програмування. – 2010. – № 2-3. – С. 330–339.
[4] Парасюк И.Н., Ершов С.В. Моделе-ориентированная архитектура нечетких мультиагентных систем // Компьютерная математика. – 2010. – № 2. – С. 62–74.
[5] Ершов С.В. Трансформационный подход к разработке адаптивных интеллектуальных агентов на основе нечетких схем переходов // Компьютерная математика. – 2011. – № 1. – С. 69–78.
[6] International Electrotechnical Commission (IEC). Technical Committee No. 65: Industrial Process Measurement and Control. IEC 1131 – Programmable Controllers Part 7 – Fuzzy Control Programming. – http://www.fuzzytech.com/binaries/ieccd1.pdf