УДК 519.685

Глибовець А. М., Гороховський С. С., Шаповалов А. Г.

АГЕНТНІ ОБЧИСЛЕННЯ

В роботі приведено базові концепції розвитку агентних обчислень. Аналізувались основні застосування, архітектури агентних систем, інструментарій створення агентних платформ. Приведено основні переваги та недоліки використання агентного підходу для створення реальних систем обробки інформації. Приведено приклад застосування мобільних агентів для реалізації координаційні моделі у хмарних обчисленнях.

***Ключові слова**: агентні обчислення, агентні системи, агентні платформи, координаційні моделі.*

В работе приведены базовые концепции развития агентных вычислений. Анализировались основные применения, архитектуры агентных систем, инструментарий создания агентных платформ. Приведены основные преимущества и недостатки использования агентного подхода для создания реальных систем обработки информации. Приведен пример применения мобильных агентов для реализации координационных модели в облачных вычислениях.

***Ключевые слова:** агентные вычисления агентные системы, агентные платформы, координационные модели.*

Article describe basic concept of agent computing. It analyzed the basic agents' applications, architecture of agent systems, instruments for agent platforms. It gives the main advantages and disadvantages of using agent approach in creating huge computing systems. As example article provide approach for coordination models in cloud computing based on mobile agents.

***Key words:** Agents computing, agent systems, agent platform, coordinationmodels.*

Вступ

З ростом Інтернету, ГРІД – систем, «хмарних» обчислень використання комп'ютерних технологій значно змінюється [1]. Застосуванням часто доводиться працювати у відкритих системах, тобто системах, де кількість сутностей може змінюватися з часом. Наприклад, хости з'являються й падають, мобільні комп'ютери та пристрої підключаються та відключаються, стають доступними нові версії програмного забезпечення тощо, тобто використання інформаційних технологій характеризується високим динамізмом. Технології семантичного вебу забезпечують доступ до великої кількості інформації з багатьох джерел, але більша частина цієї інформації неструктурована та описана природною мовою. Робота застосувань все частіше стає інтерактивною та колаборативною а задачі моделювання вимагають настільки великої кількості об'єктів взаємодії, для якої жодні методи математичного моделювання непридатні.

Для подолання цих проблем в останні роки все частіше використовуються агентні технології. Агентна парадигма побудована на поняттях штучного інтелекту і теорії розподілених обчислень. Парадигма ґрунтується на абстракції «агента» – компонента програмного забезпечення, який має властивості реактивності, автономії, про-активності та соціальної здатності [2].

Програмний агент може бути визначений як [3]: «... програмне забезпечення, яке функціонує у постійному і автономному режимі в певному середовищі ... і може здійснювати діяльність у гнучкій та інтелектуальній манері, щоб реагувати на зміни в навколишньому середовищі... В ідеалі, агент, який працює безперервно ... повинен навчатися зі свого досвіду. Крім того, агент, який мешкає в середовищі з іншими агентами і процесами, повинен мати можливість спілкуватися і співпрацювати з ними, і, можливо, переїжджати з місця на місце для цих цілей.»

Мобільні агенти були темою дослідження протягом багатьох років, проте ці дослідження здебільшого залишалися в лабораторіях і не поширювалися в комерційних галузях. Розвиток World Wide Web застосувань різко стимулював інтерес до цієї області досліджень, пропонуючи широку низку програм, в яких можна використовувати мобільних агентів. IBM і General Magic були піонерами цієї галузі.

Дослідження в області були переглянуті у 1995-1996 роки, коли Sun Microsystems випустили Java. Хоча Java просто нова інтерпретована мова програмування, вона призначена для мережевих взаємодій і є потужною технологією для мобільного коду. Веб-браузери швидко почали підтримувати Java й IT-спільнота була переконана, що мобільний код швидко стане реальністю.

За цей же період, були представлені численні пропозиції щодо реалізації мобільних агентів. Наприклад, система Lava, що [4] була розроблена у Державному університеті Північної Кароліни. Ця система була орієнтована на проблеми безпеки і в ній було розроблено просту політику безпеки для аплетів. Mitre Corporation [5] також проводила роботу в цій галузі, розробляючи

механізми автентифікації і визначаючи таксономію проблем, пов'язаних з безпекою.

Варто зауважити, що досліджуючи дану область, було виявлено, що парадигма відкритої системи завжди буде мати вади та небезпечні місця, які важко виявити. Частково виходячи з цих міркувань, а також через негативні відгуки на ранні Java-системи, проблеми безпеки були перешкодою широкому розповсюдженню технологій мобільних агентів. Були визначені архітектури безпеки, але вони містили занадто багато залишкового ризику для більшості застосувань. Останні роботи в Університеті м. Талса, наприклад, пропонують використовувати мобільних агентів для інтелектуального аналізу даних. Таке застосування зобов'язує постачальників інформації тримати свої системи «відкритими» для користувачів, більшість з яких невідомі.

Аналізу сучасних тенденцій розвитку агентних обчислень і присвячена дана робота

Програмні мобільні агенти

Програмний агент включає в себе код і дані, необхідні для виконання деяких обчислень, і вимагає агентної платформи для надання обчислювального середовища, в якому він працює. Агенти можуть бути статичними або мобільними. Стаціонарні агенти залишаються в межах однієї платформі, а мобільні агенти здатні до призупинення діяльності на одній платформі, і переміщення на іншу, де вони відновлюють виконання коду. Програмні мобільні агенти визначають нову парадигму для розподілених обчислень. На відміну від клієнт-серверної парадигми, відносини між суб'єктами, як правило, більш динамічні й рівноправні, присутня автономна співпраця.

Рисунок 1 зображує рух агента серед декількох платформ. Платформа, з якої походить агент, називається домашньою (Home platform), і зазвичай є найнадійнішим середовищем для агента. Один або декілька хостів можуть містити агентну платформу, яка може підтримувати декілька місць, де агенти можуть взаємодіяти .

Технології мобільних агентів використовують напрацювання в галузі інтелектуальних агентів, в яких підкреслюються статичні автономні агенти, здатні застосовувати знання про галузь, в якій вони працюють, та розробка програмних систем, здатних підтримувати мобільний код на гетерогенному апаратному забезпеченні (наприклад, Java технології). Інтелектуальні агенти здатні до декомпозиції і вирішення проблеми на основі співробітництва. Агенти спостерігають навколишнє середовище, причини своїх дій та дій інших агентів, взаємодіють з іншими агентами, а також виконують свої дії одночасно з іншими агентами. Взаємодії можуть передавати факти чи переконання за допомогою мовного спілкування агентів і можуть залежати від онтології, щоб досягти загального розуміння ситуації. Значне число мобільних агентних систем були розроблені в університетах і в промисловості. Хоча мобільні агенти зберігають ознаки автономії і співпраці як і інтелектуальні агенти, акцент робиться на характеристиках рухливості, часто спираючись на простий алгоритм мислення і співпрацю в рамках менш складної інтерпретації повідомлень.

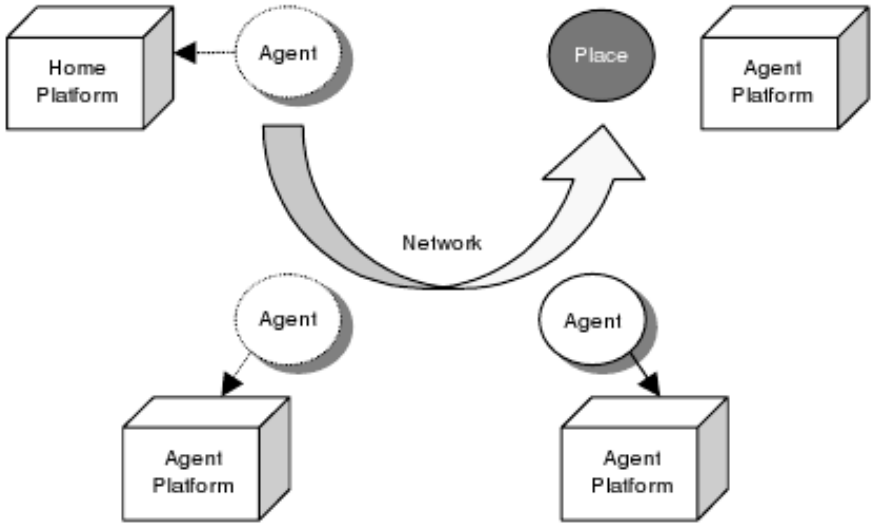

Рис. 1. Рух агента серед декількох платформ

Мали місце серйозні дебати про переваги застосування мобільних агентних систем замість клієнт-серверних систем, обробки транзакцій, і інших сталих технологій. Ця дискусія триває й сьогодні. В остаточному підсумку такі рішення повинні прийматися, враховуючи особливості програмного забезпечення та доцільність технологій для інженерних рішень.

Типова архітектура агентної системи може складатися з таких частин (рис. 2):

Агент-збирач (колектор). Такий агент буде клонований і розподілений по всій мережі. Він патрулює мережу і збирає всі події, що відбуваються на хості, до якого він відноситься. Метою є створення спеціалізованих агентів-збирачів. Ідея полягає в тому, щоб вони були зацікавлені в наборі категорій подій. Так, багато агентів-збирачів може працювати одному й тому ж комп'ютері, залежно від застосованого аналізу. Крім того, можна об'єднати здатності агентів-збирачів в єдиний агент.

Агент-корелятор. Це агент, який буде передавати певну інформацію, назвемо її критичною, до відповідного агента-аналізатора, минаючи агента-менеджера. Комунікаційний протокол за замовчуванням

є централізованим. Це означає, що агент-збирач повинен надіслати звіт менеджеру, який вирішить, чи відправляти дані в аналізатор. Така комунікативна модель не є придатною для онлайн застосування, тому що деякі важливі події повинні бути оброблені аналізатором, як тільки вони відбуваються. Саме тому кожен корелятор буде використовувати набір правил, які чітко визначають потрібні події, умови та агентів-аналізаторів, які відносяться до них.

Агент-аналізатор. Може реалізовувати кілька видів аналізу. Також можна додати поведінковий аналізатор, який буде використовувати свого роду статистичну модель, щоб визначити, що може бути розціненим в якості «нормальної» поведінки системи.

Агент-менеджер. Цей агент отримує інформацію та розподіляє серед агентів аналізаторів. Цей процес комунікації не дозволяє онлайновий аналіз. З цієї причини агенти-корелятори можуть вирішувати, чи спілкуватися безпосередньо з агентами-аналізаторами.

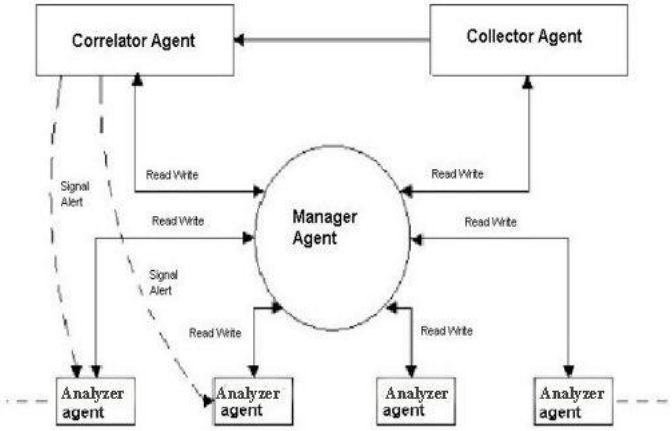

Рис. 2. Типова архітектура агентної системи

Адміністратор може поширювати визначену функціональність на будь-яку кількість хостів у мережі. Кожен вузол може отримувати будь-яку кількість агентів-колекторів, які стежать за всіма подіями, що відбуваються на хості. Всі агенти-колектори доповідають про результати своєї роботи менеджеру , який передає їх до агентів-аналізатор. Критичні події, виявлені колектором, передаються одразу до корелятора (критична подія – будь-яка подія, що є частиною сценарію визначеної функціональності). Корелятор бере на себе відповідальність передавати критичні події, отримані від колектора, до відповідних агентів-аналізаторів. Агент-аналізатор отримує дані про події від менеджера і від корелятора і виконує аналіз, використовуючи якийсь певний метод. Потім аналізатор доповідає про результати своєї роботи менеджеру, а також повідомляє систему про виявленні будь-якої аномалії.

Агенти співпрацюють один з одним, оскільки вони реагують на повідомлення про критичні події від інших агентів. Адміністратор, відповідно до своїх потреб, через зовнішній інтерфейс менеджера може зупинити виконання якихось агентів, послати їх в інші місця і відновити їх діяльність. Агенти можуть самостійно прийняти рішення перенаправити себе. Іншими словами, вони можуть зупинити своє виконання, переміститися в інше місце (хост з відповідною агентною платформою) і перезапустити своє виконання. Крім того, агенти можуть клонувати себе, особливо у випадку збільшення навантаження на мережу.

Агентні системи, фактично є P2P системами: кожен агент – це рівноправний учасник, що потенційно мусить ініціювати спілкування із деяким іншим агентом , а також може надавати такі можливості іншим агентам. Роль комунікації в агентних системах надзвичайно важлива, і вона засновується на наступних трьох характеристиках.

Агенти – активні об’єкти і вони можуть сказати «ні», якщо погано пов’язані між собою. Цей набір взаємозв’язаних властивостей – основа для вибору асинхронної комунікації, основаної на повідомленнях між агентами замість виклику віддалених процедур (remote procedure call, RPC): агент, що хоче спілкування, має лише надіслати повідомлення в певне місце призначення. Така система комунікації дозволяє одержувачу вибирати, які повідомлення обробити в першу чергу, які – пізніше. Вона також дозволяє відправникові контролювати свій потік виконання запитів, а не блокуватися, доки одержувач не прочитає і обробить повідомлення. Нарешті, вона також видаляє будь-яку тимчасову залежність між відправником і одержувачем: одержувач може бути недоступним на той час, коли відправник надсилає повідомлення, або ж його може навіть ще не існувати в той час.

Агенти виконують дії, а спілкування – це лише тип дії. Ставлячи спілкування на той самий рівень, що і дії, агент може, наприклад, виконувати план, що включає як фізичні дії, так і спілкування. Аби зробити спілкування запланованим, впливи і попередні умови для кожної можливої сесії мають бути чітко визначені.

Комунікація несе семантичне значення. Коли агент – об’єкт комунікативної дії (тобто коли він одержує повідомлення), він має мати змогу правильно зрозуміти значення тієї дії, в тому числі й чому ця дія виконується (тобто намір відправника). Таким чином з’являється потреба в існуванні універсальної семантики і стандарту.

Платформа JADE. Суттєвий поштовх практичному використанню техно-логій мобільних агентів дала платформа JADE (Java Agent Development Framework), розроблена в Telecom Italia Lab [6]. JADE – проміжне програмне забезпечення, створене TILAB для розробки розподілених мульти-агентних застосувань, основаних на peer-to-peer архітектурі. Інтелект, ініціатива, інформація, ресурси і контроль можуть повністю поширюватися на мобільні термінали, так як і на комп'ютери, у фіксованій мережі. Середовище може динамічно розвиватися з агентами, які з'являються і зникають в система згідно з потребами і вимогами прикладного програмного середовища. Комунікація між агентами, незважаючи на те, безпровідна чи провідна мережа, повністю симетрична, тобто кожен агент може грати роль як ініціатора, так і відповідача.

JADE слідує таким принципам:

– Інтероперабельність – JADE сумісний із FIPA специфікаціями. FIPA – організація, створена для розвитку і врегулювання стандартів комп'ютерного програмного забезпечення для розподілених і взаємодіючих агентів і агентних системи. Як наслідок, JADE агенти можуть взаємодіяти з іншими агентами, за умови, що вони реалізовують той самий стандарт.

– Однорідність і мобільність – JADE забезпечує гомогенний набір програмних інтерфейсів, який не залежить від базової мережеві і версії Java. JADE середовище виконання надає однакові інтерфейси для J2EE, J2SE і J2ME. Теоретично, розробники програм можуть вирішувати, яке Java середовище використовувати на етапі впровадження.

– Легкість у використанні – складність проміжного програмного забезпечення прихована за простим і інтуїтивним інтерфейсом.

– Філософія «плати тільки за те, що використовуєш» – програмістам не треба використовувати усі функції, що пропонує проміжне програмне забезпечення. Функції, які не використовуються, не вимагають, щоб програмісти знали що-небудь про них, тому не потрібно зайвих обчислювальних дій.

Архітектурна модель

JADE включає як бібліотеки (тобто класи Java), потрібні для розробки програмних агентів, так і середовище виконання, яке надає основні послуги, і яке має бути активне на пристрої перед тим, як агенти зможуть виконуватися (рис.3). Кожен екземляр середовища виконання JADE називається контейнерои (тому що містить агентів). Набір усіх контейнерів називається платформою і забезпечує однорідний прошарок, який приховує для агентів (і для прикладних розробників також) складність і різноманітність підсистем (технічне забезпечення, операційні системи, види мережі, JVM).

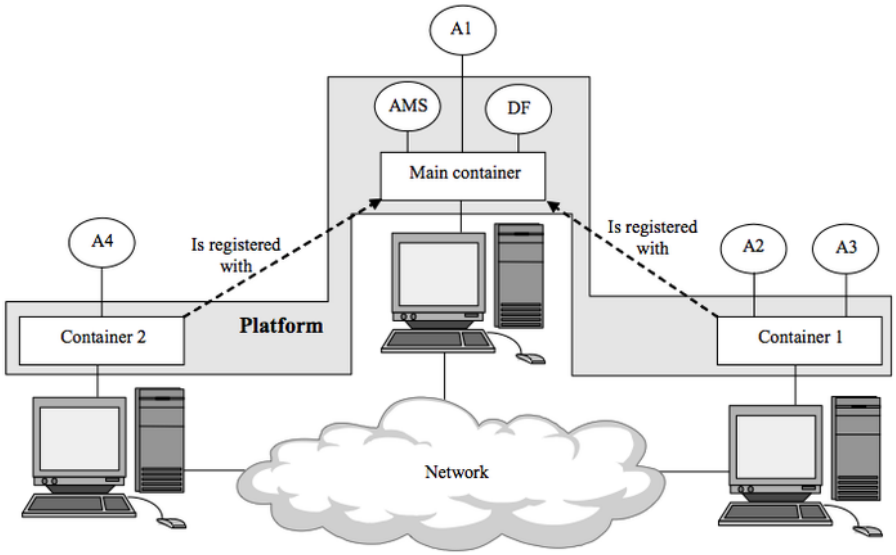

Рис. 3. Архітектурна модель

Кожна платформа повинна мати Головний Контейнер, який має двох спеціальних агентів – AMS і DF.

– AMS (Agent Management System) агент головний у платформі. Тільки цей агент може створювати і видаляти інших агентів, видаляти контейнери, і припиняти роботу платформи.

– Агент DF (Directory Facilitator) реалізує сервіс жовтої преси, який рекламує послуги агентів платформи, так що інші агенти, які вимагають ці послуги, можуть знайти їх.

Функціональна модель

З функціональної точки зору JADE забезпечує базові сервіси, необхідні для розподілених peer-to-peer програм у фіксованому і мобільному середовищах. JADE дозволяє кожному агенту динамічно знаходити інших агентів і зв'язуватися з ними згідно з peer-to-peer парадигмою. З програмної точки зору, кожен агент ідентифікується унікальним ім'ям і забезпечує набір послуг. Він може зареєструвати і змінити свої послуги і/або шукати агентів, що надають послуги, агент може управляти своїм життєвим циклом і, зокрема, спілкуватися з іншими агентами.

Агенти спілкуються, обмінюючись асинхронними повідомленнями, модель комунікації майже універсально прийнятна для розподілених і вільно зв'язаних спілкувань, тобто спілкувань між різнорідними об'єктами, які нічого не знають один про одного. Для того, щоб здійснити спілкування, агент тільки відправляє повідомлення на

місце призначення. Агенти визначаються ім'ям (немає потреби в знанні місця призначення повідомлення) і, як наслідок, немає жодної часової залежності між зв'язком агентів. Відправник і одержувач можуть бути доступний в один і той самий час. Одержувача може навіть не існувати (чи ще не існувати) або відправник може його безпосередньо не знати, а конкретизувати властивість (наприклад «усі агенти, що цікавляться футболом») як місця призначення. Оскільки агенти ідентифікують один одного ім'ям, гаряча зміна їх посилання на об'єкти прозора до застосувань.

Незважаючи на тип комунікацій, безпека зберігається. Для програм, що потребують безпеки, JADE забезпечує належні механізми, щоб автентифікувати і перевірити «права», призначені агентам. Коли треба, застосування може перевірити ідентичність відправника повідомлення і запобігти діям, яким не дозволяється виконувати (наприклад, агенту можна отримати повідомлення від агента, що представляє боса, але не дозволяється відправити повідомлення у відповідь). Усі повідомлення, якими обмінюються агенти, транспортуються у «конверті», який включає тільки інформацію, необхідну транспортному рівню. Це дозволяє кодувати вміст повідомлення окремо від конверта.

Структура повідомлення сумісна з мовою ACL, визначеною FIPA, і включає поля (такі як, наприклад, змінні відображення контексту повідомлення і проміжок часу, протягом якого можна чекати на відповідь), націлені на підтримку складних взаємодій і багаторазових паралельних спілкувань. Для подальшої підтримки виконання складних спілкувань, JADE надає набір скелетонів типових шаблонів взаємодій, щоб виконувати специфічні завдання, як наприклад переговори, аукціони і делегація завдань. Використовуючи ці скелетони (реалізуються як абстрактні класи Java), програмісти можуть звільнитися від тягара проблем синхронізації, таймаутів, станів помилки і, взагалі, усіх аспектів, які непов'язані строго до логіки застосування.

Щоб полегшити створення і управління вмістом повідомлень, JADE забезпечує підтримку автоматично перетворення між форматами для обміну контентом, у тому числі XML і RDF, і форматами для маніпуляції контентом (тобто об'єкти Java). Ця підтримка інтегрована з деякими інструментами створення онтологій.

Для збільшення масштабованості або щоб відповідати середовищу з обмеженими ресурсами, JADE забезпечує можливість виконання багаторазових паралельних завдань в межах тієї ж Java thread. Декілька елементарних завдань, як наприклад, комунікація, можуть потім комбінуватися, щоб сформувати складніші завдання, структуровані як конкуруючі Finite States Machines.

У J2SE і Personal Java середовищах, JADE підтримує мобільність коду і стану виконання. Тобто агент може припинити виконуватися на хості, мігрувати на інший віддалений хост (без необхідності мати вже встановлений код агента на цьому хості), і знову почати виконуватися з моменту, на якому він зупинився (фактично, JADE реалізує форму неслабкої мобільності, тому що стек і лічильник програм не можуть бути збережений в Java). Ця функціональність дозволяє, наприклад розподіл обчислювального навантаження під час виконання, переміщаючи агентів на менш завантажені машини без будь-якого впливу на застосування.

Платформа також включає сервіс імен (кожен агент має унікальне ім'я) і сервіс жовтої преси, яке може бути поширений на багато хостів. Графи можуть бути створені для того, щоб визначити структуровані домени сервісів агента. Інша дуже важлива особливість полягає в наборі графічних інструментів, що підтримують як відлагодження, так і управління/моніторинг фаз життєвого циклу. За допомогою цих інструментів можливо віддалено управляти агентами, навіть якщо вони вже запущені і працюють: спілкування агентів може бути емульоване, повідомлення можна перехоплювати, завдання можуть контролюватися, життєвий цикл агента може бути керованим.

Розглянемо конкретне застосування агентного підходу на такій задачі.

Координаційні моделі у розподілених обчисленнях

Проблема координації ресурсів у комп'ютерній системі виникла ще при розробці перших програмних комплексів, проте по-справжньому актуальною вона стала при переході до розподілених систем. Ще більшу вагу набула координація у розподілених обчисленнях при переході до гетерогенних територіально розподілених систем, оскільки в цьому випадку координаційна модель повинна бути реалізована для вузлів системи з різними типами архітектури, операційних систем тощо. Основними класами координаційних моделей є координація за даними та координація базована на зовнішньому керуванні [7]. При цьому вузли розподіленої системи, опрацьовуючи інформацію, повинні мати доступ до спільних ресурсів, координувати послідовність виконуваних дій, повідомляючи інші вузли або вузол-координатор про зміну свого внутрішнього стану. Ці моделі показали гарні результати при координації процесів у паралельних обчисленнях та гомогенних системах, проте для координації гетерогенних вузлів у територіально розподіленій мережі координаційна компонента повинна брати до уваги досить великі затримки при передачі координаційних примітивів до іншого вузла, необхідність координувати компоненти, що можуть бути реалізовані різними мовами програмування, при цьому надаючи можливість моніторингу, діагностики системи та ефективного керування ресурсами. Все це можна забезпечити, використавши агентно-орієнтований підхід при реалізації координаційної моделі для системи хмарних обчислень. Застосування агентних систем для координації вузлів у системі та ефективного керування ресурсами дозволить розв'язати дві основні проблеми – масштабованості та адаптивності. Існуючі на сьогоднішній день техніки не забезпечують вирішення цих проблем одночасно.

У роботі [7] зазначається, що на практиці зазвичай використовуються координаційні моделі, базовані на обміні даними у якості координаційних примітивів. А control-driven координаційні моделі носять, здебільшого, теоретичний характер.

Проект розподіленої моделі для хмарних обчислень

Проте, із переходом до хмарних обчислень, control-driven координаційні моделі все більше застосовуються для координації вузлів у хмарі завдяки механізму

асинхронності та зменшення навантаження на мережу у порівнянні з координаційною моделлю, базованою на обміні даними. Але у нашій моделі, яка була реалізована на базі агентів, ми залишили можливість обміну повідомленнями між незалежними вузлами у системі.

Нагадаємо, що Cloud вимагає насамперед вирішення таких проблем [1]:

– Масштабованість – Cloud повинен мати змогу надавати клієнту необхідну кількість обчислювальних ресурсів: процесорного часу, дискового простору, тощо.

– Адаптивність – здатність середовища пристосовуватися до потреб клієнта – певний ресурс може бути додано або вилучено із хмари у будь-який час, При зміні навантаження на вузли у хмарі, моніторинговий програмний комплекс повинен проводити реконфігурацію системи для забезпечення більш ефективного використання ресурсів.

– Доступність – оскільки Cloud це розподілена система, то зупинка окремого вузла у системі не повинна впливати на здатність системи функціонувати у звичайному режимі.

Також важливою рисою моделі хмарних обчислень повинна бути гетерогенність, оскільки розподілена система може бути побудована на вузлах з різною конфігурацією наявних обчислювальних ресурсів, у той же час надаючи кінцевому клієнту певну уніфіковану обчислювальну абстракцію. Після аналізу існуючих координаційних моделей для розподілених обчислень у хмарі, для реалізації була обрана control-driven координаційна модель, оскільки це дозволяє зменшити навантаження на мережу, використавши механізм подій та систему передачі повідомлень, уникаючи при цьому надсилання координаційних примітивів та використання спільного простору кортежів, дозволяючи при цьому використати вивільнені ресурси під обчислювальні потреби. Ми пропонуємо використати так агентний підхід для реалізації координаційної моделі (рис.4), оскільки існуючі фреймворки вже мають необхідне підґрунтя для обробки подій, а також реалізовану систему комунікації між агентами на базі системи обміну повідомленнями.

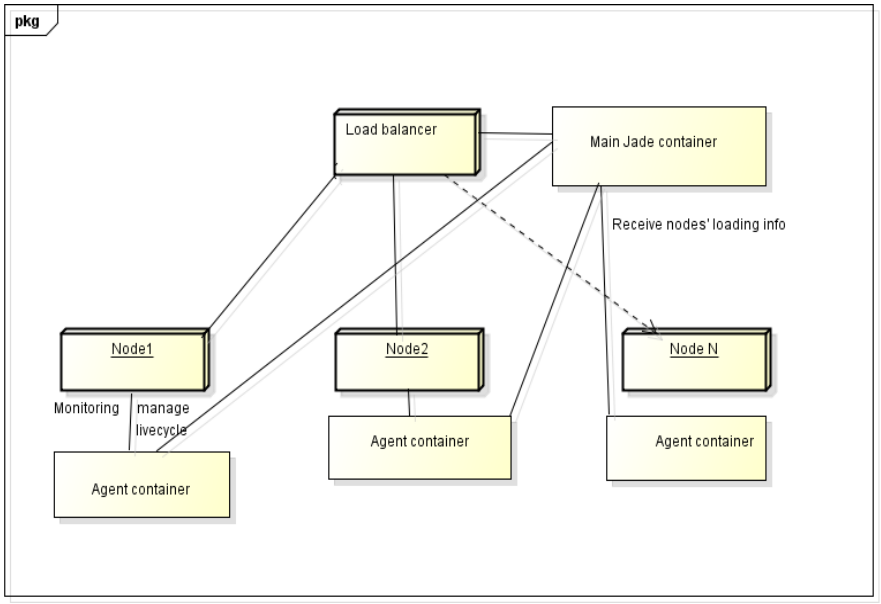

Рис.4. Функціональність реалізації координаційної моделі

Для координації було використано ієрархічний підхід типу керівник-робітники. Агенти робітники відповідальні за моніторинг вузлів системи, а також нотифікацію агента керівника про вивільнення або нестачу певних ресурсів. Також агент робітник відповідальний за керування життєвим циклом підконтрольного вузла. Агент керівник на основі отриманої від робітників інформації керує перерозподілом навантаження між вузлами; використовуючи механізм пересилання серіалізованих об'єктів між агентом та зовнішньою програмою, агент керівник повідомляє балансувальник навантажень про необхідність реконфігурувати параметри навантажень, одночасно запускаючи або зупиняючи обчислювальні вузли.

У якості обчислювальних вузлів було використано веб-сервер Jetty. Вибір було обумовлено можливістю старту та зупинки сервера безпосередньо з Java програми, що дозволило делегувати керування життєвим циклом вузла безпосередньо агенту робітнику. Також перевагами цього сервера є низька потреба у ресурсах та існуючі рішення для кластеризації та підтримки розподіленності сесії користувача. Моніторинг завантаженості вузлів проводився засобами JMX. Для цього було реалізовано систему динамічних JMX бінів, що дозволило отримувати динамічну інформацію про навантаження на сервер, як то: кількість активних сесій користувачів, кількість прийнятих запитів до сервера у одиницю часу, коефіцієнт використання оперативної пам'яті вузлом, тощо. Отримана таким чином інформація надавалася на зовні за допомогою REST веб сервісу та опрацьовувалася агентом на предмет необхідності перерозподілу ресурсів у системі. Безпосередньо балансувальник навантажень було реалізовано на базі неблкуючого асинхронного сервера xsocket, що також є досить легковісним та потребує мінімуму ресурсів. У якості стратегій балансування було обрано Round-robin алгоритм та динамічний Round-robin алгоритм із ваговими коефіцієнтами, отриманими на базі інформації про завантаження окремих вузлів у системі.

Центральний вузол системи містить логіку старту та зупинки головного JADE контейнера. У головному

контейнері стартує агент-керівник. Також у конфігураційному файлі міститься інформація про адресу фізичних комп'ютерів у мережі, на яких може бути розвернута хмара. При необхідності додавання обчислювального ресурсу до хмари, адреса вільного комп'ютера отримується із конфігураційного файла. Після цього агент керівник запускає легковісний контейнер агентів на віддаленій машині на визначеному порту та створює агента робітника. Агент робітник стартує веб сервер на вільному порту та встановлює веб додаток для проведення необхідних обчислень на цей сервер, одночасно починаючи моніторити навнтаження на цей ресурс. Далі агент робітник повідомляє агента керівника про успішний старт обчислювального вузла. Агент керівник завдяки надсиланню серіалізованого повідомлення до балансувальника навантаження, повідомляє його про необхідність здійснити перерозподіл навантаження на вузли системи. Аналогічна схема працює і при необхідності вивільнити обчислювальні ресурси.

Для оцінки ефективності реалізованої координаційної моделі, Cloud систему було протестовано на швидкодію та масштабованість за допомогою виконання обчислень двох розподілених задач: обчислення числа Пі методом Монте-Карло та обчислень на розподіленому рейтрейсері.

Переваги агентного підходу

Ряд переваг використання мобільного коду і обчислювальних парадигм на основі мобільних агентів включають в себе: подолання затримок у мережі, зменшення навантаження на мережу, асинхронне і автономне виконання, динамічна адаптація, функціонування в гетерогенних середовищах, надійна і відмовостійка поведінка.

Подолання затримок у мережі

Мобільні агенти корисні для застосувань, яким необхідно в реальному часі реагувати на зміни в навколишньому середовищі, оскільки вони можуть бути направлені з центрального контролера для здійснення операцій безпосередньо на віддаленій точці. Якщо центральний контролер відправляє повідомлення на вузли у мережі і дає вказівки про те, як реагувати на певну умову або передбачувану дію, то цей підхід є проблематичним. Наприклад, центральному контролеру, можливо, доведеться відповісти на ряд подій по всій мережі і водночас підтримувати свою нормальну роботу, і стати вузьким місцем або точкою відмови працездатності всієї системи. Якщо з'єднання з центральним сервером, на якому розташований контролер, повільне і ненадійне, комунікації у мережі сприйнятливі до неприйнятних затримок. Мобільний агенти, так як вони розподілені по всій мережі, можуть скористатися альтернативними маршрутами комунікації.

Завжди буде швидше відправити повідомлення на вузол у мережі для виконання наперед визначеного резидентного коду, а не відправляти мобільного агента. Однак така архітектура вимагає, щоб всі дії щодо відповідей та реконфігурації були визначені заздалегідь, відтворені і поширені по всій мережі. З цієї точки зору мобільні агенти є більш вдалим рішенням, оскільки вони здатні приймати рішення самостійно, в окремих випадках не покладаючись на центральний вузол. Це зменшує випадки виникнення мертвих місць в мережі через намагання вузла передати інформацію на центральний контролер.

Зменшення навантаження на мережу

Однією з найбільш гострих проблем для сучасних розподілених систем є обробка величезної кількості даних, згенерованих інструментами моніторингу мережевого трафіку і засобами логування хоста. Мобільні засоби дають можливість зменшити навантаження на мережу за рахунок усунення необхідності передачі даних.

Мобільні агенти добре підходять для спеціальних, гнучких, проблем пошуку та аналізу, пов'язаних з декількома розподіленими ресурсами, які вимагають спеціальних завдань, які не підтримуються дата-сервером. Пошук на основі мобільних агентів і підхід до аналізу даних можуть допомогти зменшити мережевий трафік в результаті передачі великих обсягів даних через мережу для місцевої обробки. Замість передачі даних по мережі, мобільні агенти можуть бути надіслані на машину, на якій зберігаються дані, тобто спосіб обчислення посилається до даних, замість переміщення даних для розрахунку, що дозволяє знизити навантаження на мережу для такого сценарію. Очевидно, що передача агента, який менше за розміром, ніж дані, знижує навантаження на мережу. Ці переваги враховуються при проведенні порівняння між зашифрованими легкими мобільними агентами та відносно великими даними, що передаються.

Асинхронне і автономне виконання

Платформи на базі мобільних агентів дозволяють розподіленій системі продовжувати роботу у разі відмови центрального контролера або каналу зв'язку. На відміну від передачі повідомлень або віддаленого виклику процедур, після того, як мобільний агент запускається з домашньої платформи, він може продовжувати працювати автономно, навіть якщо його домашня платформа, з якої він почав працювати, більше недоступна або не підключена до мережі. Неспроможність мобільного агенту спілкуватися з центральним контролером не заважає йому виконувати покладені на нього завдання.

Хоча віддалена діяльність можлива, необхідно вирішити ряд питань. Розподіл функцій центрального контролера між мережевими компонентами є нетривіальним завданням. Ще одна проблема стосується саме мобільних агентів. Наприклад, засновані на Java мобільні агенти, зазвичай завантажують свої файли класів динамічно, в міру необхідності, з їхньої домашньої платформи. Можливість динамічно завантажувати класи має вади у плані безпеки. Якщо домашня платформа недоступна, файли класів можуть бути надані локального або повинні бути знайдені і підключені з віддаленого довіреного хоста. Завантаження класів з віддаленої платформи або з локального хоста є небезпечним. Файли класів можуть бути змінені таким чином, щоб змінити функції агента або навіть «підслуховувати» комунікацію агентів. Схожі проблеми також виникають з різними версіями класів.

Динамічна адаптація

З огляду на те, що конфігурація, топологія і характеристики трафіку мережі змінюються з часом, тести мережі і моніторинг діяльності також повинні змінюватися. Мобільні агенти забезпечують багатосторонню і адаптивну обчислювальну парадигму, оскільки вони можуть бути повернуті додому, відіслані на інший хост, клоновані або відправлені спати (retracted,

dispatched, cloned, or put to sleep), якщо стан мережі чи хоста змінився. Наприклад, якщо завантаження платформи хоста занадто велике, й потужність хоста не відповідає очікуванням агента, агент і його дані можуть переміститися на іншу машину, яка може краще задовольнити обчислювальні потреби. Мобільні агенти можуть поширювати себе серед вузлів в мережі таким чином, щоб підтримувати оптимальну конфігурацію для вирішення специфічної проблеми.

Функціонування в гетерогенних середовищах

Великі мережі підприємств зазвичай складаються з багатьох різних обчислювальних платформ і обчислювальних пристроїв. Одна з найбільших переваг мобільних агентів – реалізація функціональної сумісності на прикладному рівні. Функціональна сумісність на комп'ютерному або транспортному рівні, як наприклад рішення від єдиного виробника, вимагає істотних змін в середовищі вузла. Якщо платформи для мобільних агентів мають бути встановлені на кожному вузлі, самі агенти можуть незалежно налаштовуватися. Оскільки мобільні агенти загалом незалежні від розташування й на транспортному рівні, а залежні тільки від середовища виконання, вони пропонують привабливий підхід для інтеграції гетерогенних систем. Здатність мобільних агентів діяти в гетерогенних обчислювальних середовищах зумовлена віртуальною машиною або інтерпретатором на платформі хоста. Процес з'єднання даних може бути полегшений, якщо запускати мобільні агенти на комутаторах, маршрутизаторах та інших мережевих елементах. Мобільні агенти можуть працювати на будь-якому обчислювальному вузлі, який може прийняти агентну платформу. Здатність агентів діяти в гетерогенних середовищах також забезпечує можливість для простої інтеграції мережевих і вузлових інструментів, діючих на різних платформах.

Надійна і відмовостійка поведінка

Здатність мобільних агентів реагувати динамічно на несприятливі ситуації і події робить простішим розробку стійких розподілених систем. Наприклад, якщо вузол закінчує роботу, усі агенти, що виконуються на цій машині, попереджаються і їм надається час, щоб перейти на інший вузол і продовжити роботу зі збереженням попереднього стану. Підтримка роз'єднаного функціонування й розподілена парадигма виключають деякі сценарії відмови працездатності і дозволяють мобільним агентам пропонувати відмовостійкі характеристики. Незважаючи на особливості мобільних агентів, що збільшують відмовостійкість, треба навести деякі недоліки. Здатність мобільних агентів рухатися від однієї платформи до іншої в гетерогенному середовищі зумовлена використанням віртуальних машин і інтерпретаторів. Віртуальні машини і інтерпретатори, проте, можуть запропонувати тільки обмежену підтримку збереження і відновлення стану через різницю реалізацій.

Звичайні техніки опрацювання помилок недостатні для агентної парадигми. Наприклад, копіювання стану в контрольних точках до і після прибуття, і на завершенні певних транзакцій або подій, може бути потрібне, щоб гарантувати відповідне повернення до початкового (без помилки) стану. Однак, з кожною контрольною процедурою або механізмом збереження відмовостійкості збільшується навантаження на систему. Хоча наявний цілий арсенал технік відмовостійкості, розробник має бути обережним у виборі, бо ці механізми впливають на систему і її функціональність.

Хоча мобільні агенти автономні і здатні до роз'єднаного функціонування, відмова домашньої платформи або інших платформ, на які агенти покладаються, щоб надати послуги безпеки, може серйозно зменшити їх функціональність. Хоча мобільний агент може стати більш відмовостійким при переміщенні на іншу машину, довіра агента до незнайомої платформи накладає обмеження на його функціональність. Розробники платформ мобільних агентів також стикаються з вибором альтернативних варіантів між захистом і відмовостійкістю. Наприклад, для більшої безпеки деякі агентні архітектури, сформовані на централізованих клієнт-серверних моделях, вимагають, щоб агенти повернулися до центрального сервера перед тим, як перейти на інший хост. Очевидно, що в такій ситуації всі агенти вразливі у разі відмови центрального сервера.

Недоліки агентних обчислень

Очевидний недолік використання мобільних агентів – спричинення вразливості в мережі. Рішення на основі агентів, можливо, не виконуються досить швидко. Агенти можуть містити велику кількість коду, роблячи неможливими швидкі передачі між вузлами

Безпека

Проблеми захисту, пов'язані з мобільним кодом, – одна з основних перешкод до поширеного використання цієї технології. Обчислювальна парадигма мобільних агентів породила перелік нових загроз, до яких не можна застосувати звичайні техніки захисту, тому стандартні техніки захисту мають бути змінені або винайдені нові. Загрози можуть бути поділені на чотири широкі категорії: агент до агента, агент до платформи, платформа до агента та інші до платформи. Категорія агента до агента включає безліч загроз, в яких агенти експлуатують слабкості захисту інших агентів або запускають атаки проти інших агентів, те ж саме стосується й категорії агент до платформи. Категорія платформа до агента включає погрози, в яких платформа намагається зламати захист агентів. Категорія інші до платформи – ситуація, коли зовнішні об'єкти, у тому числі агенти і платформи погрожують захисту платформи агента.

Існує додаткова техніка, яку можна застосувати до проблем захисту. Така техніка включає механізми, що управляють доступом до обчислювальних ресурсів, криптографічні методи для закодованого обміну інформацією, криптографічні методи для ідентифікації і автентифікації користувачів, агентів і платформи, і механізми для контролю подій, які відбуваються на агентній платформі.

Продуктивність

Обмеження продуктивності скриптових та інтерпретаційних мов не надають перспективних рішень цієї проблеми, тому що перевага гетерогенності, яку пропонують ці мови, реалізується за рахунок швидкодії. Коли продуктивність є вагомим критерієм, вірогідніше, що розподілена система буде сконструйована з використанням комбінації мобільних агентів, статичних агентів, і інших технологій.

Розмір коду

Агентна система – складне програмне забезпечення. Агенти, які представляють деякі сервіси, можуть містити

велику кількість коду. Якщо передбачається, що ці агенти виконують завдання, які залежать від операційної системи (або навіть від декількох різних), то цей код може бути величезним. Розмір коду мобільних агентів може обмежити їх застосування, оскільки пересилання агентів між вузлами може займати багато часу. Така передача вимагатиме більшого обчислення і мережевих ресурсів. Можливе рішення цієї проблеми – мати статичних агентів, які надають інтерфейс взаємодії агентам, що рухаються між машинами.

Відсутність апріорного знання

Великі мережі підприємств складаються з декількох різних апаратних платформ з різними операційними системами, які мають власні налаштування і програмне забезпечення. Не тривіально для мобільних агентів мати апріорне знання про те, як сконфігурована система і як впорядковані дані, і при цьому залишатися простими. Статичні і менш нестійкі агенти можуть більше знати, як дані впорядковані і який до них доступ, і здатні служити посередниками між мобільними агентами та іншими платформами. Локалізовані дані зручніше опрацьовувати через стандартні програмні інтерфейси.

Обмежені методології

Клієнт-серверна парадигма добре відома і розвинена технологія, але область розподіленого контролю систем на основі мобільних агентів – все ще тема багатьох дослідницьких зусиль. Автономна поведінка агента, включаючи співпрацю з іншими агентами в різних мережевих вузлах, створює динамічне середовище, яке вимагає нових методології і інструментів, щоб належним чином сформулювати і сконструювати агентні системи. Відсутність зрілих методологій робить це завдання важким, але проблему, ймовірно, можна подолати, якщо комерційний попит на агентні системи зросте.

Проблеми програмування і впровадження

Властивості мобільних агентів, такі як переміщення і клонування, додають більше складності до процесу проектування і розробки. Разом з цією додатковою складністю, агентні системи будуть навіть більш схильні до помилок, ніж звичайні. Відсутність інструментів проектування, розробки і підтримки для мобільних агентів, які потрібні для впровадження будь-якого великого проекту з використанням агентних технології, ще більше додає проблем. Розробники агентів і їх адміністратори також потребують кращих механізмів елементу управління ресурсами на агентних платформах.

Висновки

В роботі наведено базові концепції розвитку агентних обчислень. Аналізувались основні застосування, архітектури агентних систем, інструментарій створення агентних платформ. Приведено основні переваги та недоліки використання агентного підходу для створення реальних систем обробки інформації. Приведено приклад застосування мобільних агентів для реалізації координаційні моделі у хмарних обчисленнях.

ЛІТЕРАТУРА

1. Глибовець М.М., Бондар Є.О., Гороховський С.С. Хмарні обчислення. Проблеми і перспективи //Вісник Київського університету. Сер: Фізико-математичні науки. – 2011,–№ 1, –с.74-81.
2. Гороховський С. С. Агентні технології: спроба критичного огляду // Наукові записки. Т. 18: Спеціальний випуск / Національний університет «Києво-Могилянська Академія». – К., 2000. – С. 391—395
3. Bradshaw, Jeffrey M. An introduction to software agents. In Software Agents, edited by J. M. Bradshaw, 3-46. Cambridge, MA: AAAI Press/The MIT Press, 1997.
4. Hansoty, Jatin N., «LAVA: SecureDelegation of Mobile Applets» Master's ThesisNorth Carolina State Univ., 1997. URL: http: //shang.csc.ncsu.edu:80/lava.html
5. Farmer, W.M., J.D. Guttman, and V. Swarup Security for Mobile Agents: Issues and Requirements . In: S. Wakid and J. Davis, eds., Proceedings of the 19th National Information Systems Security Conference, Vol. 2, pp. 591–597, Baltimore Convention Center, Baltimore, Maryland, October 22–25, 1996.
6. http://jade.tilab.com/
7. Глибовець М.М., Гороховський С. С., Стукало М. С. Проблеми координації в розподілених обчисленнях. Комп'ютерні технології. – Миколаїв: Вид.-во ЧДУ ім. Петра Могили, Вип. 95., 2010. – С. 88 – 91.

Дата надходження статті до редколегії 21.04.2012 р.

ГЛИБОВЕЦЬ А. М. – к.ф.-м.н., доцент кафедри мережних технологій факультету інформатики Національного університету «Києво-Могилянська академія».

Коло наукових інтересів: розподілені інтелектуальні пошукові системи, програмні системи підтримки електронного навчання.

ГОРОХОВСЬКИЙ С. С. – к.ф.-м.н., старший науковий співробітник, доцент кафедри інформатики Національного університету «Києво-Могилянська академія», керівник магістерської програми «Інформаційні управляючі системи та технології».

Коло наукових інтересів: теорія структур даних, паралельні й розподілені обчисленням, агентні технології та їх застосування.

ШАПОВАЛОВ А. Г. – магістр факультету інформатики Національного університету «Києво-Могилянська академія».

Коло наукових інтересів: розподілені системи обробки інформації.