УДК 004.4

Бойко Ю. В.,
канд. фіз.-мат. наук, ІОЦ Київського національного університету імені Тараса Шевченка, м. Київ, Україна
boyko@univ.kiev.ua
Глибовець М. М.,
д-р фіз.-мат. наук, професор, Національний університет «Києво-Могилянська академія», м. Київ, Україна
glib@ukma.kiev.ua
Кривий С. Л.,
д-р фіз.-мат. наук, професор, Київський національний університет імені Тараса Шевченка, м. Київ, Україна
sl.krivoi@gmail.com
Погорілий С. Д.,
д-р техн. наук, професор, Київський національний університет імені Тараса Шевченка, м. Київ, Україна
sdp@univ.net.ua
Ролік О. І.,
д-р техн. наук, Національний технічний університет України «Київський політехнічний інститут», м. Київ, Україна
arolick@gmail.com
Теленик С. Ф.,
д-р техн. наук, професор, Національний технічний університет України «Київський політехнічний інститут», м. Київ, Україна
telenyk@acts.kpi.ua
Єршов С. В.,
д-р фіз.-мат. наук, с.н.с., Інститут кібернетики імені В. М. Глушкова НАН України, м. Київ, Україна
sershv@ukr.net

МОДЕЛЮВАННЯ КОМПОНЕНТІВ БАЗОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПОДІЛЕНИХ СЕРЕДОВИЩ

При розробці розподілених середовищ необхідно дослідити формальними методами основні властивості створюваного середовища. Компонентні моделі середовища повинні мати універсальний характер з метою якнайбільше абстрагуватися від конкретних предметних областей застосування та конкретних програмних реалізацій засобів підтримки таких середовищ. Запропоновано нову універсальну модель на основі транзиційних систем для моделювання роботи середовищ мережного типу та оптимізації управління такими середовищами; розроблено новий алгоритм визначення зупинки розподіленої асинхронної системи, досліджено коректність алгоритму визначення зупинки. Проведено порівняльний аналіз моделей координації за параметрами, що є найважливішими для розробки розподілених систем хмарних обчислень. Побудовано зовнішньо керовану модель для координації вузлів у хмарі, що реалізована на основі агентного підходу і забезпечує масштабованість та високу адаптивність розподілених середовищ.

Ключові слова: *розподілене середовище; транзиційна система; координація; хмарні обчислення; мережа Петрі; агенти.*

1. Вступ

При розробці високопродуктивних розподілених середовищ основним є створення математичної моделі такого середовища, яка дає можливість дослідити формальним методами основні властивості створюваного середовища. До найважливіших властивостей моделей належать правильність функціонування і взаємодія з внутрішніми компонентами та зовнішнім

середовищем. Найбільш популярними математичними моделями на сьогодні виступають автоматні і мережеві моделі. Це пов'язано з тим, що для таких моделей створено алгоритмічну основу, що дозволяє в автоматичному або напівавтоматичному режимі досліджувати їх властивості. До таких математичних моделей належать транзиційні системи, як математичні моделі найбільш загального типу. Отже, першим кроком при проектуванні і побудові ефективних розподілених середовищ колаборативного типу є створення формальної математичної моделі. Очевидним є й те, що це мусить бути мережна модель за рахунок необхідності використання можливостей мережі Інтернет [1]. Ця модель має досить універсальний характер з метою якнайбільше абстрагуватися від конкретних предметних областей застосування та конкретних програмних реалізацій засобів підтримки таких середовищ. Вирішення проблем, пов'язаних з координацією на базі абстрактної моделі, допомагає згодом побудувати програмні реалізації, що успадкують аналогічні властивості [2–5]. Окрім загальності, ця модель дає можливість при реалізаціях не приймати жорсткі проектні рішення, що забезпечує надалі високу гнучкість і застосовність програмних проектів [6–7].

2. Математичні моделі розподіленої системи

За основу специфікації структури колаборативного середовища в комп'ютерній мережі взята модель у вигляді транзиційної системи.

Означення 1. Транзиційною системою (ТС) називається п'ятірка $A = (S,T,f,g,s_0)$, де

- S – скінченна або нескінченна множина станів;
- T – скінченна або нескінченна множина переходів;
- f і g – функції із S в T, які ставлять у відповідність кожному переходу $t \in T$ два стани $f(t)$ і $g(t)$, які називаються відповідно початком і кінцем переходу;
- s_0 – початковий стан ТС.

При зображенні ТС використовується орграф, вершинами якого є стани ТС, а дугами – переходи. Як правило переходи в ТС мають позначки (така ТС називається позначеною ТС (ПТС)), які являють собою назву дії, що виконується в ТС при спрацюванні цього переходу.

У якості математичної моделі колаборативного середовища взята транзиційна модель Х. Х. Гарсіа-Луна-Ачевеса і Г.-П. Доммеля. В ній колаборативне середовище подано як набір $\Gamma = <S, U, R, F>$, де S – множина сеансів, U – множина користувачів, R – набір спільних ресурсів, F – множина рівнів, що керують ресурсами. Кожна з цих сутностей має свій набір атрибутів (рис. 1).

Рис. 1. Зв'язок між складовими системи

Оскільки дана модель надає детальну структуру формальної специфікації та можливість перевірки властивостей колаборативних систем, саме вона є основою при побудові нашої моделі колаборативного розподіленого середовища. Нижче розглядається, як за допомогою скінченних автоматів та мереж Петрі можна змоделювати ТС контро́лерів для чотирьох складових системи: сеансу, користувача, ресурсу та рівня [8] (рис. 2)

Моделювання починається з опису мережної моделі контро́лерів. Для зручності побудови мережної моделі спочатку уточнюється опис її роботи на рівні специфікації вимог до функціонування контролера (див. розділ 3) або на рівні процедурного подання автоматної моделі. Користувачі репрезентуються в системі своїми профілями. Під профілем користувача ми розуміємо його «особову справу», що зберігається на головному сервері мережі. Профіль визначає права користувача в межах даної мережі та слугує джерелом інформації про поточні дії користувача. Уточнено і модель профіля користувача.

Контро́лер ресурсу відповідає за надання ресурсу за запитом та за коректне його вилучення (в разі аварійних ситуацій у роботі системи). Протокол рівневого контролю домагається доступу до спільно використовуваних об'єктів, надаючи доступ до рівнів відповідно до політики, визначеної для групи обслуговування. Фактично зайняття рівня рівносильне одержанню права діяти. Саме через контро́лер рівня відбуваються практично всі дії з ресурсом (окрім створення, яке робиться безпосередньо).

На базі автоматної моделі будуються відповідні моделі з допомогою мереж Петрі. Таке перетворення традиційне, тим не менш, було виявлено певні нюанси. Наприклад, моделювання механізму прибирання неактуальних повідомлень, який продукує повідомлення про звільнення/видалення ресурсу, вимагало введення додаткових специфічних переходів. Це потрібно для запобігання виникненню глухих кутів у роботі контро́лера.

Як відомо, апарат мереж Петрі не лише полегшує моделювання складних систем, а й дає змогу провести

глибокий аналіз властивостей системи (наприклад безпечність системи, чи тупіковість), що моделюється.

Для моделювання колаборативної системи в цілому потрібно внести користувачів у запропоновану раніше модель.

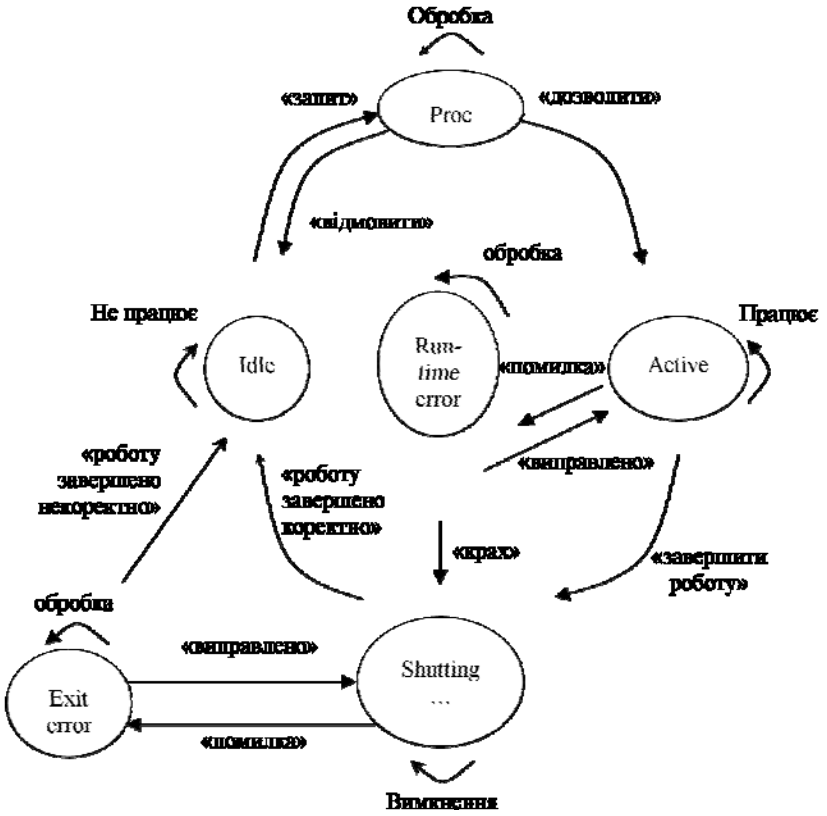

Рис. 2. ТС схеми контро́лера сеансу

Для дослідження системи на наявність властивостей безпечності або тупиковості використано метод побудови дерева (графа) досяжності станів. У результаті такої побудови встановлені такі властивості наведеної моделі:

Теорема 1. *Мережа Петрі, яка моделює а) контро́лер сеансу; б) профіль користувача; в) контро́лер ресурсу; г) контро́лер рівня, обмежена і жива.*

Ця теорема встановлює правильність функціонування лише окремих компонентів колаборативної системи, а тепер необхідно дослідити властивості загальної моделі, зібраної з цих частин. Справа в тому, що в процесі об'єднання мереж окремих складових в одну мережу постає низка проблем. По-перше, це забезпечення контролю команд, які віддає користувач. Через недетерміновану природу мереж Петрі цілком імовірна ситуація, коли користувач буде постійно віддавати системі одну й ту саму команду, наприклад, «створити ресурс», або ж віддаватиме команди в неправильній послідовності (спочатку «вилучити ресурс», потім «створити ресурс» і т. д.). Для уникнення подібних ситуацій потрібно ввести додаткові умови. Прив'язки дозволу на віддачу певної команди до одержаного від системи повідомлення про виконання попередньої операції недостатньо.

Для розв'язання даної проблеми вводиться поняття синхронного добутку ТС, який являє собою композицію ТС окремих компонентів і дає можливість звести задачу аналізу зібраної системи до аналізу відповідної ТС [9–11]. Для спрощення моделі ми взагалі відмовилися від команд та повідомлень, замінивши відповідні послідовності переходів безпосередніми зв'язками між станами контро́лерів, яким віддається команда. При цьому поточний стан підлеглого контро́лера визначає можливі команди, які йому може віддати користувач. Розглянемо поняття синхронного добутку ТС і пояснимо прикладом його побудову.

Нехай $A_1,...,A_n$ транзиційні системи, де $A_i = (S_i, T_i, f_i, g_i, s_0^i)$, $i=1,...,n$.

Означення 2. Обмеженням синхронізації називається підмножина **T** множини:

$$(T_1 \cup \{\varepsilon\}) \times ... \times (T_n \cup \{\varepsilon\}) \setminus \{(\varepsilon,...,\varepsilon)\},$$

де ε – тотожний перехід, який означає відсутність якої-небудь дії в ТС. Елементи із T називаються *глобальними переходами*. Якщо $t = (t_1,...,t_n) \in T$ і $t_i \neq \varepsilon$, то говорять, що ТС A_i бере участь у переході. Кортеж $A = (A_1,...,A_n,T)$ називається синхронним добутком ТС $A_1,...,A_n$.

Для ілюстрації цієї побудови обмежимося розглядом прикладу побудови синхронного добутку двох автоматів: які представляються ТС, а отримана ТС, у свою чергу, моделюється мережею Петрі. В прикладі наводяться невеликі розміри ТС, але сенс побудови від цього не змінюється.

Приклад моделювання синхронного добутку ТС мережею Петрі. Нехай дано два скінченні *X*-автомати своїми графами переходів:

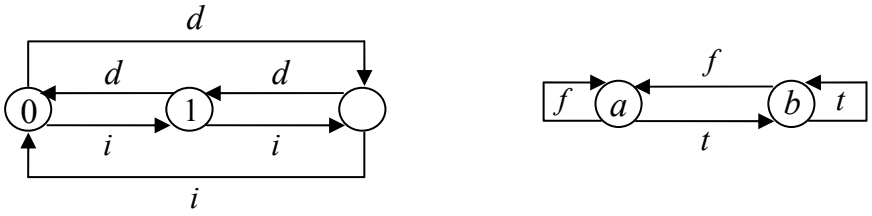

Рис. 3. Х-автомати A_1, A_2

Відповідні ТС мають вигляд:

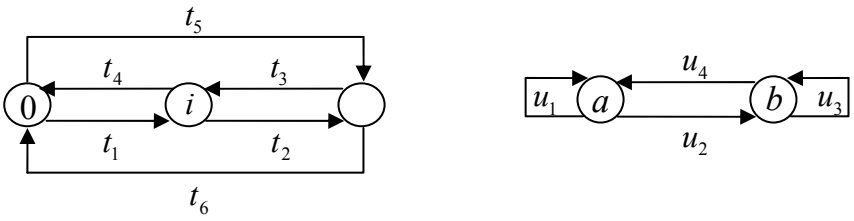

Рис. 4. ТС A_1, A_2, які відповідають Х-автоматам

У цих ТС множини переходів є такими:

$$t_1 = (0,i,1),\ t_2 = (1,i,2),\ t_3 = (2,d,1),\ t_4 = (1,d,0),$$
$$t_5 = (0,d,2),\ t_6 = (2,i,0),$$
$$u_1 = (a,f,a),\ u_2 = (a,t,b),\ u_3 = (b,t,b),\ u_4 = (b,f,a).$$

Припустимо, що в процесі проектування були задані такі обмеження синхронізації:

$$T = \{(t_1,\varepsilon),(t_2,\varepsilon),(t_3,u_2),(t_4,u_4),(t_5,\varepsilon),(t_5,u_1),(\varepsilon,u_3)\}.$$

Тоді синхронний добуток ТС $A = (A_1, A_2, T)$ моделюється такою мережею Петрі:

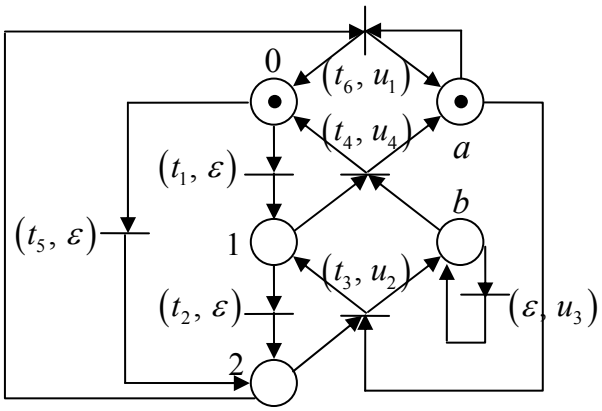

Рис. 5. Мережа Петрі, яка моделює синхронний добуток ТС

Нехай початкова розмітка МП ставить у місця 0 і *a* по одній фішці.

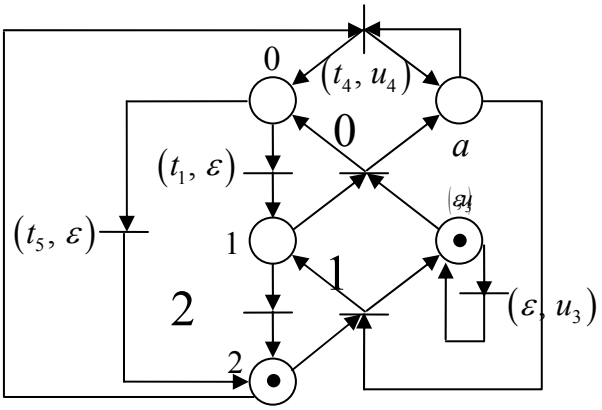

Рис. 6. Розмітка, яка є глухим кутом в МП

Очевидно, що за цієї розмітки МП має глухий кут, до якого приводить така послідовність глобальних переходів: $(t_5,\varepsilon),(t_3,u_2),(t_2,\varepsilon)$. Дійсно, ця послідовність дає розмітку МП, за якої неможливий жоден перехід у цій МП.

За допомогою синхронного добутку ТС и його моделювання МП розв'язується друга проблема, пов'язана з виходом користувача із системи. Користувач на момент виходу може володіти якимось ресурсом, і доки він не звільнить його своєю командою, ніхто інший не зможе до нього доступитися. Тому на вихід користувача з системи також було накладено додаткові умови. Задля полегшення подальшого аналізу мережі системи було проведено оптимізацію шляхом заміни деякої послідовності позицій та переходів одним переходом, спрацювання якого реалізувало той самий сценарій.

Для аналізу цієї мережі використано програмні засоби побудови дерева досяжності для мережі. На основі його аналізу доведено:

Теорема 2. МП, яка моделює колаборативну систему з *n* користувачами обмежена і жива.

Запропонована модель колаборативного середовища є базисом як для досліджень властивостей вже наявних систем організації віддаленої взаємодії користувачів, так і для проектування нових систем. Завдяки тому, що для побудови моделі було використано мережі Петрі, її можна розширювати за рахунок додавання нових користувачів та ресурсів, передбачення можливості одночасного існування двох або більше сеансів зв'язку. Крім того, для ґрунтовнішого дослідження системи її можна перевести на мову часових або стохастичних мереж Петрі.

При створенні розподілених систем перед розробниками постає низка вимог, яким має задовольняти система: гетерогенність або різнорідність, відкритість, безпека, можливість розширення, значний паралелізм, широка прозорість і ефективна обробка збоїв. Зрозуміла важливість останньої вимоги. Тому запропоновано алгоритм визначення зупинки розподіленої асинхронної системи [12; 13].

При розробці розподіленої системи необхідно врахувати всі можливі типи збоїв системи і реалізувати адекватну поведінку системних компонентів у разі їх виникнення. Для цього уточнено модель Н. Лінча розподіленої системи асинхронного типу, обґрунтовано зручність її використання для розв'язання проблеми збоїв.

Нехай існує деяка розподілена система асинхронного типу, що складається із *n* процесів, які пронумеровані від 0 до *n*-1. Кожен процес може відправляти повідомлення за допомогою буферизованих каналів зв'язку до інших процесів, тобто існує (*n*-1)2 каналів зв'язку між кожною парою процесів відповідно. Спочатку всі процеси активні. Процес може посилати повідомлення, тільки тоді коли він активний. Який завгодно активний процес може вирішити перестати бути активним, але не навпаки. При отриманні повідомленя від іншого процесу неактивний процес стає знову активним.

Розподілена асинхронна система – це множина або набір поведінок (*behaviors*), кожен елемент ***s*** цієї множини складається в свою чергу із послідовності станів. Завдяки запропонованій моделі було уточнено визначення основної системи.

Для побудови алгоритму визначення зупинки системи було розширено визначення основної системи:

SystemTn in **N**(act *in* **Z**n→**B**, chan in **Z**n x **Z**n→**Seq**(MSG), term in **B**)

Основна ідея алгоритма полягає в наступному. Якщо процес 0 (або головний процес) хоче визначити зупинку системи, він відправляє повідомлення останньому процесу n-1. Якщо неактивний процес i > 0 отримує таке повідомлення, він просто пересилає це повідомлення процесу i–1. Активний процес тримає це повідомлення доти, поки сам не стане неактивним. Далі він також відправляє повідомлення процесу i–1. Можна сказати, що це повідомлення є «сигналом» (token), який циркулює у мережі і сигналізує про закінчення роботи для кожного процесу, через який проходить цей сигнал.

Значна розповсюдженість розподілених асинхронних систем і важливість алгоритму визначення їх зупинки потребує формального доведення коректності алгоритму. Доведення правильності алгоритму, шляхом послідовного уточнення тверджень, додає значну долю впевненості, що алгоритм відповідає основним вимогам до нього. Основна складність доведення полягає в тому, що на кожному етапі функціонування система може згенерувати дію, яка може призвести до помилки роботи всієї системи. Перш за все, доведено, що кожна можлива поведінка розширеної системи також є можливою поведінкою базової системи і навпаки, кожна поведінка базової системи можлива в системі визначення зупинки. Це так звана інваріантність властивостей базової моделі відносно надбудови.

Для доведення того факту, що алгоритм є коректним використано доведення теореми:

Теорема 3. (*Коректність*). Якщо алгоритм визначає, що система зупинена, то система дійсно зупинена, тобто для всіх n, act, chan, term:

SystemT$_n$(act, chan, term) => [](term => **terminated$_n$**(act, chan))

Ґрунтовне дослідження проблеми зупинки асинхронних розподілених систем дає обґрунтування для ефективних реалізацій.

3. Координація в розподілених системах і обчисленнях.

Використання всього потенціалу масштабних розподілених систем вимагає наявності програмних моделей, які явно оперують поняттями паралельної співпраці між дуже великим числом активних сутностей, які складають єдине застосування [14–16]. Така потреба призвела до проектування та впровадження декількох координаційних моделей разом із мовами, які підтримують ці моделі. Майже всі ці схеми створювалися для того, щоб надати розробникам каркас (framework), який би покращував модульність, сприяв би повторному використанню компонентів (послідовних чи паралельних), підвищував би переносимість та мовну сумісність. Однак ці моделі відрізняються в тому, як саме визначається поняття координації, що конкретно координується, якими засобами досягається коорди-

нація, і які метафори застосовані для представлення цих понять [17].

Розробка та впровадження великих і складних систем, таких як авіадиспетчерська служба, транспортна навігація, інтелектуальний пошук та обробка даних, показали, що єдина мова програмування чи системна архітектура не в змозі охопити всі можливі ситуації, які виникають при розробці такого складного і багатофункціонального застосування [18]. Підхід до програмування великих застосувань було знайдено в ідеях багатомовності та гетерогенності. Багатомовне програмування забезпечує зв'язок між застосуваннями та ізолює небажану взаємодію. Під гетерогенністю ми розуміємо підтримку різних моделей обчислень.

Концепція координації тісно пов'язана з багатомовністю та гетерогенністю. Розробку паралельного чи розподіленого застосування можна представити як розробку двох окремих частин: власне обчислювальна частина об'єднує процеси, відповідальні за обробку даних, а координаційна частина бере на себе комунікацію та кооперацію між процесами. Тому координацію можна використати для відділення обчислень у розподіленому чи паралельному застосуванні від власне комунікації, що дозволяє проводити розробку двох частин окремо і потім об'єднати результати роботи. Оскільки координаційний компонент системи відділено від обчислювального, то останній можна вважати «чорною скринькою», і тому конкретна мова, використана для обчислень, не має значення для координаційного механізму. Очевидно, оскільки мови для обчислень можуть бути різними, координація заохочує використовувати гетерогенні архітектури.

Відповідно, координаційна модель – це той засіб, який склеює окремі процеси в єдиний ансамбль.

Координаційні моделі можна розділити за багатьма критеріями, але найбільш поширена класифікація на дві категорії: data-driven (орієнтовані на задачу) або control-driven (орієнтовані на процеси) координаційні підходи.

Основною особливістю моделей та мов data-driven координації є той факт, що стан обчислень на будь-який момент часу визначається даними, що отримуються або передаються та фактичною конфігурацію скоординованих компонентів. Моделі data-driven координації, як правило, надають деякі координаційні примітиви (у поєднанні з координаційною метафорою), які змішуються з чисто обчислювальною частиною коду. Ці примітиви зручно інкапсулюють комунікаційні і конфігураційні аспекти обчислень.

В основу моделі control-driven координації покладено спільний простір даних, який являє собою асоціативну структуру даних. Спільний простір даних відповідає за зберігання інформації, такої як кортежі.

Основна різниця між моделями data-driven та control-driven координації полягає у тому, що в останньому випадку існує майже повне розділення координації та власне обчислень.

Принцип роботи control-driven координації ґрунтується на взаємодії постачальника (producer) та споживача (consumer) даних. Ця взаємодія визначається як потік або канал даних між вихідними портами постачальників і вхідними портами споживачів.

Крім стилістичних відмінностей між двома основними координаційними моделями, які впливають на ступінь розділення обчислювальних та координаційних частин, кожна категорія підходить для різних типів застосувань. Модель control-driven координація, як правило, використовується в основному для розпаралелювання обчислювальних задач. Підхід control-driven координації зазвичай використовується для моделювання систем. Отже, представники першої категорії, як правило, намагаються координувати дані, у той час представники другої намагаються координувати сутності (які, можуть бути не тільки звичайними процесами, а й пристроями, компонентами системи і т. д.).

Проект розподіленої моделі для хмарних обчислень. 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 порівняно з координаційною моделлю, базованою на обміні даними. Але в моделі, яка реалізована на базі агентів, є можливість обміну повідомленнями між незалежними вузлами в системі.

Нагадаємо, що Cloud вимагає насамперед вирішення проблем масштабованості, адаптивності та доступності [19]. Ще однією важливою рисою моделі хмарних обчислень мусить 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 наявних координаційних моделей для розподілених обчислень у хмарі, для реалізації було обрано 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. Застосовано агентний підхід для реалізації координаційної моделі (рис. 7), оскільки наявні фреймворки вже мають необхідне підґрунтя для обробки подій, а також реалізовану систему комунікації між агентами на базі системи обміну повідомленнями.

Для координації використано ієрархічний підхід типу керівник-робітники. Агенти робітники, відповідальні за моніторинг вузлів системи, а також нотифікацію агента керівника про вивільнення або нестачу певних ресурсів. Агент-робітник відповідальний також 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.

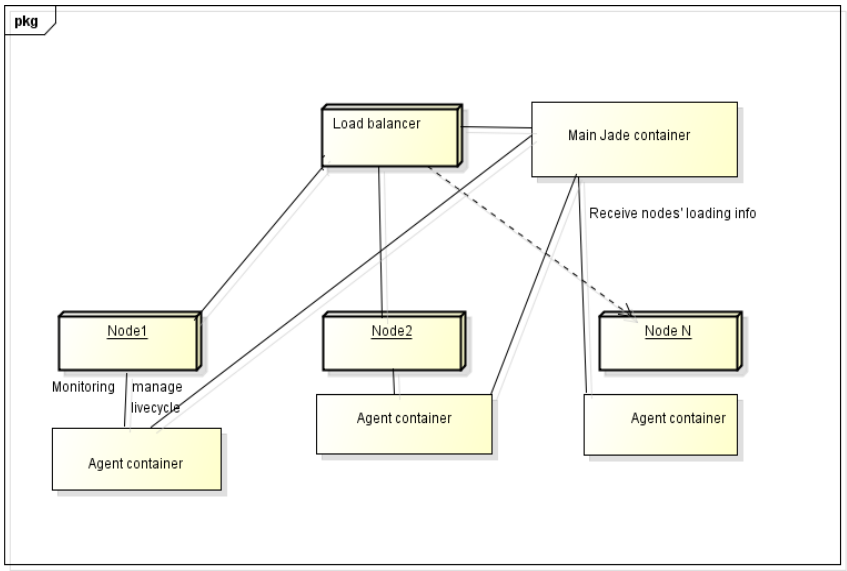

Рис. 7. Функціональність реалізації координаційної моделі

У якості обчислювальних вузлів використано веб-сервер Jetty. Моніторинг завантаженості вузлів проводився засобами JMX. Безпосередньо балансувальник навантажень реалізовано на базі неблокуючого асинхронного сервера xsocket, що також є досить легковісним та потребує мінімуму ресурсів. У якості стратегій балансування обрано Round-robin алгоритм та динамічний Round-robin алгоритм із ваговими коефіцієнтами, отриманими на базі інформації про завантаження окремих вузлів у системі.

Ряд переваг використання мобільного коду і обчислювальних парадигм на основі мобільних агентів включають у себе: подолання затримок у мережі, зменшення навантаження на мережу, асинхронне і автономне виконання, динамічну адаптацію, функціонування в гетерогенних середовищах, надійну і відмовостійку поведінку.

Виконуючи розподілене обчислення, клієнт надсилає кількість симуляцій, які необхідно виконати на кожному з обчислювальних вузлів. Оскільки число таких запитів до кожного з вузлів може бути досить великим, то при закінченні певного ресурсу агент-робітник сигналізує агенту керівникові про необхідність динамічно додати обчислювальних ресурсів у систему у випадку наявності вільних комп'ютерів. Потім отриману в результаті обчислень кількість вдалих спроб обчислювальний вузол повертає клієнту через веб-сервіс. З точки зору клієнта обчислювальний процес виглядає досить прозоро: клієнт робить запити на єдину веб-адресу. При цьому за динамічний менеджмент ресурсів відповідає логіка, імплементована в агентах на базі Control-driven координаційної моделі.

При цьому отримуємо можливість регулювати рівень абстракції моделі від PSM, залежно від конкретної ситуації.

Висновки

У статті запропоновано та досліджено цілісні підходи до побудови моделі та програмної реалізації систем підтримки розподілених інтелектуальних середовищ (РІС) з організацією ефективної взаємодії між підсистемами власне управління та управління контентом; запропоновано нову інтерпретацію автоматної моделі для моделювання роботи РІС мережного типу та її застосування для дослідження оптимізації управління такою системою; розроблено новий алгоритм визначення зупинки розподіленої асинхронної системи, досліджена коректність алгоритму.

Проведено порівняльний аналіз моделей за параметрами, що є найважливішими для розробки розподіленої системи хмарних обчислень та обрано зовнішньо керовану координаційну модель для координації у хмарі, що реалізована за допомогою агентного підходу. Перевагами розробленої системи, що використовує агентний підхід для координації вузлів у системі, є масштабованість та висока адаптивність.

ЛІТЕРАТУРА

1. Глибовець М. М. Програмні засоби створення і супроводу розподіленого навчального середовища / М. М. Глибовець, І. В. Сергієнко, С. С. Гороховський, А. Глибовець. – М. – К. : НаУКМА; Аграр Медіа Груп, 2012. – 710 с.
2. Глибовец Н. Н. Структурированные данные и семантическая паутина: технологии Wiki / А. Н. Глибовец, Н. Н. Глибовец, Д. Е. Покопцев, М. О. Сидоренко // Проблемы программирования. – 2013. – № 1. – С. 45–67.
3. Погорілий С. Д. Технологія віртуалізації. Динамічна реконфігурація ресурсів обчислювального кластера / С. Д. Погорілий, І. В. Білоконь, Ю. В. Бойко // Математичні машини і системи. – 2012. – № 3.– С. 3–18.
4. Погорілий С. Д. Комп'ютерні мережі. Апаратні засоби та протоколи передачі даних. Підручник серії Автоматизація наукових досліджень за редакцією академіка АПН України Третяка О. В. / С. Д. Погорілий, Д. М. Калита. – К. : ВПЦ Київський університет. – 2007. – С. 456.
5. Погорілий С. Д. Концепція створення гнучких гомогенних архітектур кластерних систем / С. Д. Погорілий, Ю. В. Бойко, Д. Б. Грязнов, О. Д. Ломакін, В. А. Мар'яновський // Проблеми програмування. – 2008. – № 2–3. – С. 84–90.

6. Бойко Ю. В. Формування логіки внутрішньої міжрівневої взаємодії в багатоланковій системі / Ю. В. Бойко, С. Д. Погорілий, О. В. Коваленко // Проблеми програмування. – 2008. – № 2–3. – С. 91–96.
7. Теленик С. Ф. Управління ресурсами центрів оброблення даних при виділених серверах / С. Ф. Теленик, О. Ролік, М. Букасов, К. Крижова // Автоматика. Автоматизація. Електротехнічні комплексі та системі. – 2009. – № 2 (24). – С. 122–136.
8. Глибовець М. М. Формальна модель координаційно-орієнтованої мережі для колаборативної системи навчання / М. М. Глибовець, Д. К. Гломозда // Проблеми програмування. – 2006. – № 2–3. – С. 402–412.
9. Крывый С. Л. Верификация программ: состояние, проблемы, результаты / С. Л. Крывый, А. Н. Максимец // Кибернетика и системный анализ. – 2013. – № 6. – С. 3–14.
10. Крывый С. Л. Верификация программ: состояние, проблемы, результаты / С. Л. Крывый, А. Н. Максимец // Кибернетика и системный анализ. – 2014. – № 1. – С. 11–20.
11. Кривий С. Л. Дискретна математика / С. Л. Кривий. – Букрек : Чернівці. – 2014. – 576 с.
12. Глибовец Н. Н. Корректность алгоритма останова распределенной асинхронной системы / Н. Н. Глибовец, С. В. Комличенко // Компьютерная математика. – 2003. – № 2. – С. 137–144.
13. Глибовец Н. Н. Алгоритм определения останова распределенной асинхронной системы / Н. Н. Глибовец, С. В. Комличенко // Компьютерная математика. – 2003. – № 1. – С. 22–36.
14. Глибовець М. М. Проектирование инструментария сетевого программного продукта / М. М. Глибовець, С. С. Гороховский // Проблеми програмування. – 2010. – № 2–3. – С. 102-106.
15. Глибовец Н. Н. Упрощенная инфраструктура для трансформации XML-моделей / Н. Н. Глибовец, В. М. Федорченко // Кибернетика и системный аналіз. – 2010. – № 1. – С. 105–111.
16. Глибовец Н. Н. Расширение языка программирования Scala средствами параллелизма и распределенности с помощью координационной системы Linda / Н. Н. Глибовец, С. С. Гороховський, М. С. Стукало // Кибернетика и системный анализ. – 2010. – № 2. – С. 55–59.
17. Глибовець М. М. Хмарні обчислення. Проблеми і перспективи / М. М. Глибовець, Є. О. Бондар, С. С. Гороховський // Вісник Київського університету. Серія «Фізико-математичні науки». – 2011. – № 1. – С. 74–81.
18. Глибовець М. М. Агентні обчислення / М. М. Глибовець, С. С. Гороховський, А. Г. Шаповалов // Наукові праці ЧДУ імені Петра Могили. Серія: Комп'ютерні технології. – 2012. – Вип. 179, Т. 191. – С. 54–63.

Бойко Ю. В., *ИВЦ Киевского национального университета имени Тараса Шевченко, г. Киев, Украина*
Глыбовец Н. Н., *Национальный университет «Киево-Могилянская академия», г. Киев, Украина*
Крывый С. Л., *Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина*
Погорелый С. Д., *Киевский национальный университет имени Тараса Шевченко, г. Киев, Украина*
Ролик А. И., *Национальный технический университет Украины «Киевский политехнический институт», г. Киев, Украина*
Теленик С. Ф., *Национальный технический университет Украины «Киевский политехнический институт», г. Киев, Украина*
Ершов С. В., *Институт кибернетики имени В. М. Глушкова НАН Украины, г. Киев, Украина*

МОДЕЛИРОВАНИЕ КОМПОНЕНТОВ БАЗОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РАСПРЕДЕЛЕННЫХ СРЕД

При разработке распределенных сред необходимо исследовать формальными методами основные свойства создаваемой среды. Компонентные модели среды должны иметь универсальный характер с целью как можно больше абстрагироваться от конкретных предметных областей применения и конкретных программных реализаций средств поддержки таких сред. Предложена новая универсальная модель на основе транзиционных систем для моделирования работы сред сетевого типа и оптимизации управления такими средами; разработан новый алгоритм определения остановки распределенной асинхронной системы, исследована корректность алгоритма определения остановки. Проведен сравнительный анализ моделей координации по параметрам, которые являются важнейшими для разработки распределенных систем облачных вычислений. Построено внешне управляемую модель для координации узлов в облаке, которая реализована на основе агентного подхода и обеспечивает масштабируемость и высокую адаптивность распределенных сред.

Ключевые слова: *распределенная среда; транзиционная система; координации; облачные вычисления; сеть Петри; агенты.*

Boyko Yu. V., *ITC of Kyiv National Taras Shevchenko University, Kyiv, Ukraine*
Glybovets M. M., *National University of «Kyiv-Mohyla Academy», Kyiv, Ukraine*
Kryvyi S. L., *Kyiv National Taras Shevchenko University, Kyiv, Ukraine*
Pogorily S. D., *Kyiv National Taras Shevchenko University, Kyiv, Ukraine*
Rolik O. I., *National Technical University of Ukraine «Kyiv Polytechnical Institute», Kyiv, Ukraine*
Telenik S. F., *National Technical University of Ukraine «Kyiv Polytechnical Institute», Kyiv, Ukraine*
Yershov S. V., *Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine, Kyiv, Ukraine*

MODELING OF COMPONENTS OF BASIC SOFTWARE FOR DISTRIBUTED ENVIRONMENTS

Investigation with formal methods becomes one of the latest trends when verifying basic properties of developed distributed environment. The complexity growth of such environments in the last years has rocketed the interest in developing formal models based on transition systems. The abstraction from specific application domains and specific

software implementations tools for collaborative distributed environments introduces a degree of universality that has to be handled in addition to the correct operation of components and coordination of distributed components. In order to adequately deal with coordination on the basis of an abstract model we propose the concept of synchronous product of transition systems, which enables to reduce the problem of analysis of component distributed system to analysis of the product system. The method uses a reachability tree (graph) for Petri nets that corresponds to transition systems, to explore safety and deadlock-freeness properties. In order to adequately provide model of collaborative distributed environments, Petri nets that are able to deal with many users, are bounded and alive. A new algorithm for determining the stopping of asynchronous distributed systems is developed; correctness of the algorithm for determining a stop is explored. The good performance of this approach is supported by control-driven coordination model, as it can reduce the load on the network, using the mechanism of events and messaging system while avoiding the use of common space of tuples. A structure «manager-workers» of agents is responsible for dynamic resource management in control-driven coordination model. The results obtained show that externally managed model for coordinating components in the cloud, which is implemented through agent-based approach, provides high scalability and adaptability of distributed environments.

Key words: *Distributed environment; Transition system; Coordination; Cloud computing; Petri nets; Agents.*

© Бойко Ю. В., Глибовець М. М., Кривий С. Л.,
Погорілий С. Д., Ролік О. І., Теленик С. Ф.,
Єршов С. В., 2014

Дата надходження статті до редколегії 18.12.2014 р.