УДК 539.3
DOI: 10.31891/2219-9365-2020-67-1-3

ВАСИЛЬЧЕНКО І. П.
Уманський національний університет садівництва
САЧАНЮК-КАВЕЦЬКА Н. В.
Вінницький національний технічний університет
БАРАНЕНКО Р. В.
Уманський національний університет садівництва

ТЕХНОЛОГІЇ РОЗПОДІЛЬНИХ СИСТЕМ ТА ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

У цій роботі представлена структура та функціональність zFunction, що є адаптивною розподіленою обчислювальною платформою, яка підтримує зручну модель програмування для розробки додатків для паралельної обробки. Це дозволяє розробникам розробляти програмне забезпечення як би вони програмують для одного комп'ютера, а потім він автоматично піклується про розподіл даних та паралелізацію завдань на різних вузлах кластера або декількох ядрах центрального процесора. Zфункція таким чином істотно покращує продуктивність складного розподіленого програми, що обробляють велику кількість даних у режимі реального часу, це критично важливі системи.

У цій роботі використано репрезентативне тематичне дослідження з домену фінансових послуг, щоб показати, як zFunction може отримати вигоду від цих типів програм.

Ключові слова: інженерна мережа, інструменти тестування, моделі програмування, безперервність бізнесу, інформаційні технології.

VASYLCHENKO IVAN
Uman National University of Horticulture
SACHANIUK-KAVETS'KA NATALIA
Vinnytsia National Technical University
BARANENKO ROMAN
Uman National University of Horticulture

DISTRIBUTION SYSTEMS AND PARALLEL COMPUTING TECHNOLOGIES

High performance computing (HPC) has recently become important in several sectors, including science and manufacturing.

To build powerful systems requires a combination of software models to increase system parallelism, especially dual and three-tier programming models to increase parallelism in disparate systems, including CPUs and GPUs.

However, building systems with different programming models is error-prone and complex, and difficult to verify. In addition, testing parallel programs is already a difficult task, as parallel errors are difficult to detect due to the indeterminate behavior of the parallel application. Integrating multiple programming models into a single application complicates testing, as this integration can lead to new types of errors.

Despite efforts to create tools for testing parallel systems, much remains to be done, especially when testing heterogeneous and multilevel programming models.

This paper presents the structure and functionality of zFunction, which is an adaptive distributed computing platform that supports a convenient programming model for developing applications for parallel processing. This allows developers to develop software as if they were programming for a single computer, and then it automatically takes care of data distribution and task parallelization on different nodes of the cluster or multiple CPU cores. The z function thus significantly improves the performance of a complex distributed program that processes large amounts of data in real time, these are critical systems.

This paper uses a representative case study in the field of financial services to show how zFunction can benefit from these types of programs.

Key words: engineering network, testing tools, programming models, business continuity, information technologies.

Постановка проблеми. Високопродуктивні обчислення (HPC) останнім часом набувають важливого значення в декількох секторах, включаючи наукову та виробничу галузі.

Для побудови потужних систем необхідна комбінація програмних моделей для збільшення паралельності системи, особливо подвійних та трирівневих моделей програмування для збільшення паралельності в різнорідних системах, що включають центральні процесори та графічні процесори.

Однак побудова систем з різними моделями програмування є схильною до помилок і складною, а також важко перевірити. Крім того, тестування паралельних програм вже є складним завданням, оскільки паралельні помилки важко виявити через невизначену поведінку паралельного додатка. Інтеграція декількох моделей програмування всередину одного додатку ускладнює тестування, оскільки ця інтеграція може спричинити помилки нового типу.

Незважаючи на зусилля, спрямовані на створення інструментів тестування паралельних систем, ще потрібно зробити багато роботи, особливо при тестуванні різнорідних та багаторівневих моделей програмування.

Аналіз останніх досліджень і публікацій. Високопродуктивні обчислювальні технології (HPC) в даний час є частиною всіх наукових та виробничих секторів, що зумовлені вдосконаленням машин HPC, особливо з урахуванням зростаючої уваги до суперкомп'ютерів Exascale, що, як вважають, можливо здійснити до 2022 року в рамках різних досліджень.

Це постійне вдосконалення являє собою складне завдання побудови масивно паралельних систем, які можна використовувати в цих супер машинах. Паралельні програми повинні бути без помилок, щоб задовольнити вимоги та переваги можливостей використовуваної моделі програмування. Тестувати такі програми дуже важко через їх величезні розміри, мінливу поведінку та інтеграцію між різними моделями програмування в одному додатку.

Мета роботи. Дослідити існуючі інструменти тестування, які перевіряють паралельні системи для виявлення помилок під час роботи та класифікуємо перевірені засоби тестування за різними категоріями та підкатегоріями на основі використовуваних методів тестування, моделей цільового програмування та виявлених помилок під час виконання.

Постановка задачі. Цей документ намагається дослідити різні доступні засоби тестування та методи, які виявляють помилки під час виконання в паралельних додатках, що використовують моделі програмування, включаючи однорідні та неоднорідні системи.

Розглянути та вивчити різні інструменти та техніки залежно від різних факторів та характеристик. Цей огляд досліджуватиме, що ще потрібно зробити при тестуванні паралельних систем та напрямки майбутніх досліджень у цій галузі.

Виклад основного матеріалу. Паралельне програмування стає все більш важливим для дослідників та розробників широкомасштабних розподілених та паралельних застосувань у ряді доменів, включаючи оцінку та моделювання фінансових ризиків (наприклад, оцінку ризику вартості та історичні розрахунки), прийняття рішень у реальному часі. створення на основі алгоритмічного зворотного зв'язку (наприклад, створення ринку, арбітраж електронних стратегій та високочастотна торгівля), а також обробка, архівування, зберігання та пошук сховищ вмісту для корпоративних систем управління контентом (наприклад, веб-сайтів новин та веб-енциклопедій).

З появою товарних багатоядерних процесорів та систем хмарних обчислень дослідникам та розробникам також потрібні новіші методи паралельного програмування, які можуть максимально використовувати такі системи. Традиційні методи паралельного програмування, такі як передача повідомлень [8] та обчислювальна мережа загальної пам'яті [15], були застосовані дослідниками в університетах та національних лабораторіях для розробки та розгортання розподілених та паралельних додатків масштабів підприємств. Однак паралельна розробка додатків залишається складною проблемою, у сфері широкомасштабної розробки розподілених та паралельних додатків, де традиційні технології обчислювальних мереж не можуть бути застосовані через наступні обмеження:

- Складні моделі програмування, яким не властива підтримка для таких функцій, як виявлення вузлів, розповсюдження даних, балансування навантаження та контроль паралельності. Заявки, написані нами-Такі методи не добре масштабуються для складних критично важливих систем.
- Традиційні технології обчислювальних мереж не є агресивними на платформі.
- Існує крута крива навчання, задіяна в освоєнні цих паралелей паралельного програмування.

Для вирішення цих обмежень розробили адаптивне розподілене обчислювальне проміжне програмне забезпечення під назвою zFunction, яке покращує широкомасштабні розподілені та паралельні програми, створюючи адаптивні обчислення в режимі реального часу та розподілені обчислення за запитом. zFunction надає наступні можливості для дослідників та розробників:

- Настроюване проміжне програмне забезпечення, чиї підключаються служби автоматизують багато втомливих та схильних до помилок заходів, пов'язаних із мережевим програмуванням, включаючи обробку різних мережевих протоколів, (де) маршування, відмовостійкість, створення потоків та керування ними, та вдосконалене балансування навантаження через мережу обчислювальних серверів.
- Децентралізована архітектура програмного забезпечення, яка не має жодних точок відмови.
- Пряма модель паралельного програмування, що дозволяє розробникам складних великомасштабних додатків (наприклад, спільних програм фінансування та обробки даних) -підписати програмне забезпечення, яке працює в кластері комп'ютерів так, ніби вони програмують для одного комп'ютера Паперової організації.

Програми обчислювального фінансування, що включають масові моделювання, добре підходять для розподілу та розпаралелювання. На жаль, забороняючі зусилля, необхідні для паралелізації цих застосувань за допомогою традиційних механізмів, обмежили рух фінансової галузі в цьому напрямку.

Оскільки ринки все більше домінують за допомогою електронних торгових систем, ефективність роботи в режимі реального часу стає дедалі критичнішим фактором прийняття своєчасних торгових рішень.

Типовою проблемою, що зустрічається в галузі фінансових послуг, є оцінка ризиків. Методи Монте-Карло [9], що спираються на моделювання, засновані на гіпотетичних сценаріях поведінки ринку, виявились досить корисними при розрахунках ризиків, особливо для портфелів із похідними фінансовими інструментами. Однак обчислювальна інтенсивність таких методів, як правило, обмежує частоту їх використання.

Отже, суттєва вигода від підвищення ефективності таких обчислень. Решта цього розділу представляє тематичне дослідження фінансового аналізу, засноване на обчисленні avalue-at-risk (VaR) [12] з використанням моделювання Монте Карло для демонстрації ключових проблем проектування розробки паралельних програм обчислювального фінансування.

У фінансовій математиці та управлінні фінансовими ризиками Valueat Risk (VaR) є широко використовуваним показником ризику збитків для конкретного портфеля фінансових активів. Для даного портфеля, вірогідності та часового горизонту, VaR визначається як порогове значення, таким чином, що вірогідність того, що втрата від ринкової вартості на портфелі за даний часовий горизонт перевищує це значення (припускаючи нормальний ринок і відсутність торгів у портфелі) є даною Рівень ймовірності. Для портфелів, що включають традиційні інструменти, такі як акції, розроблено ефективні та обчислювально-економні аналітичні методи (наприклад, метод дисперсії та коваріації [11, 7]) для розрахунку вартості ризику. Найважливіше, що такі методи спираються на важливі припущення про характер розподілу збитків, включаючи умову, що це нормальний розподіл. Отже, такі алгоритми не можна застосовувати до портфелів, що містять екзотичні інструменти (наприклад, опціони та інші похідні інструменти), а менеджери ризиків повинні вдаватися до більш узагальнених методів.

Завдяки своїй загальності методи Монте-Карлома часто використовуються для розрахунків VaR для портфелів з опціями . Замість того, щоб моделювати майбутню ефективність портфеля на суто теоретичних обговореннях, такі методи моделюють великий, репрезентативний набір можливих сценаріїв продуктивності, а потім базують вимірювання VaR на підсумках результатів.

Одним із варіантів таких методів - проектуванням портфелів з опціонами - є використання історичних показників базових цінних паперів опціонів у випадково обраних проміжках часу для формування правдоподібних сценаріїв майбутніх цінних паперів.

Оскільки існують потужні алгоритми (такі як модель Блека-Шоулза [5] та модель біноміального дерева [6]) для прогнозування опціонних цін на основі базових цін, такі сценарії можуть бути розширені, щоб генерувати прогнози щодо ефективності всіх позицій у портфелі, і, отже, загальна ефективність портфоліо. Хоча такі методи Монте Карло (ми використовуємо методи Монте Карло на основі історичного моделювання) є загальними та універсальними, вони надзвичайно обчислювальні. Зокрема, алгоритми виведення цін на опціони з тих, що лежать в їх основі, цілком задіяні, а обчислювальні витрати поєднуються з тим фактом, що багато таких прогнозів необхідно розрахувати, щоб сформувати досить велику кількість сценаріїв для надійного статистичного аналізу. Дійсно, обчислювальні витрати на такі розрахунки часто є фактором, що обмежує їх ширше використання у фінансовій галузі.

Типова послідовна реалізація VaRCalculation де вхідні дані та результати для обчислень часто розміщуються в додатках, таких як електронні таблиці Microsoft Excel, а обчислення контролюються Visual Basic для Applicationsscript. Актуальні розрахунки можуть виконуватися або в процесорі Excel, делегованому іншому процесу в конфігурації клієнт / сервер. Цей документ зосереджується на репрезентативному обчисленні VaR, де обчислювальним завданням є оцінка одноденного значення ризику для портфеля з позиціями в акції, індексу та ряду американських опціонів на ці цінні папери.

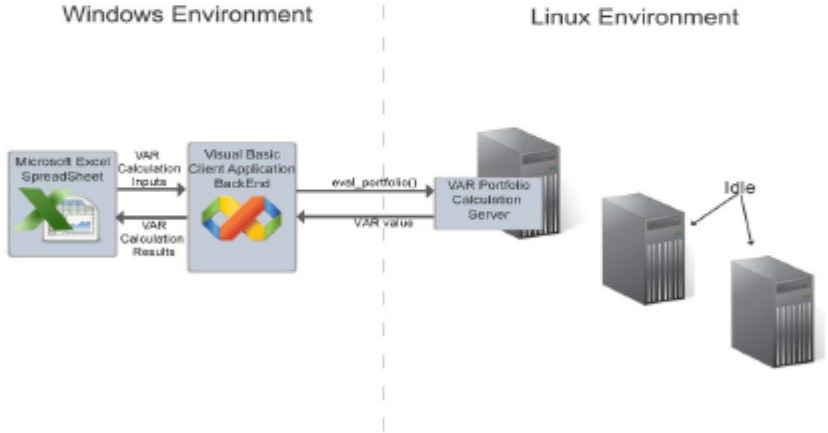

Рис. 1. Архітектура послідовного обчислення

На рисунку 1 показана архітектура такого послідовного обчислення VaR, де програма aVisual Basicclient обчислює VaR для набору портфоліо (вхідні дані яких зберігаються в Microsoft Excelsprepreheetheet) шляхом виклику повторних запитів в бібліотеці оцінки VaR, розміщеній на сервері Linux. Карлосимуляція досягає цього розрахунку VaR за допомогою (1) обчислення вартості портфеля в кінці часового горизонту за великої кількості сценаріїв поведінки на ринку та (2) кількісне визначення максимальних збитків, що очікуються з заданою ймовірністю (тобто довірчий інтервал моделювання), 30 акцій та один фонд відстеження індексів відповідають нашому портфелю. Ми збираємо історію щоденних повернень запасів за два роки, що передують даті обчислення VaR, у таблиці Microsoft Excelspreadsheet. Наступним кроком є створення 1000 нечітких сценаріїв шляхом випадкового вилучення 1000 наборів повернень та обчислення базового значення VaR за 1 день.

Як показано на малюнку 1, функція обчислення VaR вбудована в бібліотеку, розміщену на машині Linux. Аналогічним чином, клієнтом для цього обчислення є anExcelspreadsheet, а виконувана програма aVisual Basicclient допомагає таблиці Excelspreadsheets робити віддалені виклики в бібліотеці Linux.

Виклики проектування паралелізації розрахунку VaR У нашому прикладі ми маємо американські варіанти з дискретними роздільними ідеями для окремих запасів, оскільки для них не існує рішення закритої форми, і для розрахунку ціни потрібно використовувати трудомісткі біноміальні дерева.

Отже, наступні обчислювальні кроки беруть участь у обчисленні 1-денного VaR: (1) випадковим вибором історичних дат, (2) застосуванням повернень за кожну історичну дату до базових цін на початкову дату, щоб отримати сценарій для базових цін в кінці часовий горизонт, (3) для кожного сценарію цін на базові цінні папери, оцінка цін усіх опціонів у портфелі, та (4) після оцінки всіх сценаріїв, формування імітованого розподілу значень портфеля та обчислення VaR. розрахунки логічно незалежні і досить численні в рамках аналізу єдиного портфеля, цей розрахунок VaR має велику паралельність, яку можна використати, і його легко розпаралелювати.

Хоча паралелізація забезпечує реалістичний та економічний спосіб поліпшити ефективність суханалізу, звичайні реалізації цього паралельного розрахунку мають ряд конструктивних проблем при розвантаженні послідовних розрахунків паралельно працювати на сотнях або тисячах розподілених обчислювальних серверів.

Завдання 1: Виявлення та адресація віддалених обчислювальних серверів для розподілених обчислень. Якщо розробники додатків пишуть вихідний код вручну для паралельного програмування, їм доведеться ідентифікувати IP-адреси клієнта та серверних машин, визначати багатоадресні адреси, а також обробляти варіанти, пов'язані з базовим мережевим стеком, для передачі запитів та відповідей через мережі.

Більше того, цей процес буде повторюватися щоразу, коли змінюються основні платформи, наприклад, вхідні дані можуть бути переміщені з таблиці Excel до бази даних, або сервер обчислення VaR може бути перемішений з Linuxhost на aSolarishost.

Незалежно від цих змін у топології мережі, апаратного забезпечення та платформи, дані обчислення VaR повинні розподілятися, а розподілені обчислення повинні виконуватися. Як описано вище, вручну змінювати вихідний код для обробки таких складних випадків використання важко.

Завдання 2: Поширення даних для віддаленого розподіленого обчислення. Розподілені обчислення передбачають перетворення внутрішнього стану програми (наприклад, вхід для обчислення VaR, що зберігається в електронній таблиці Excel), у зовнішній формат, який можна передати через мережу на віддалені обчислювальні сервери. Техніка програмування, яка використовується для здійснення цього перетворення, називається маршалінгом, а зворотний процес перетворення зовнішнього формату даних у внутрішній формат даних називається демаршалінг.

Історично розробники додатків вручну писали (де) маршируючий код для задоволення розподілених обчислювальних вимог обчислень VaR. Цей (де) маршовий код сильно залежить від формату даних, що надсилаються, і платформ, що розміщують процеси клієнт-сервер, що ускладнює ручну розробку вихідного коду.

Завдання 3: Ефективне розповсюдження віддалених обчислень для ефективного управління ресурсами в мережі. Після того, як розробники додатків розробляють рішення завдань 1 і 2 вище, для розповсюдження запитів між різними серверами обчислень необхідні інтелектуальні алгоритми планування та розподілу запитів. Ефективне розповсюдження запитів гарантує, що (1) ефективно використовуються всі апаратні ресурси, (2) віддаленим обчисленням не заважає дисбаланс навантаження між серверами обчислень, а також (3) клієнти захищені від неоднорідних можливостей апаратного та програмного забезпечення.

Завдання 4: Відмовостійкість та застосування прозорого виявлення несправностей та відновлення. Коли віддалені обчислювальні сервери виконують дуже сильні складні розрахунки додатків, апаратні збої можуть зірвати обчислення. З такими типами відмов потрібно боротися з необхідністю, оскільки і обчислювальний сервер (и), і зв'язки зв'язку можуть бути недоступними. Розробка вихідного коду для забезпечення відмовостійкості може включати написання коду для виявлення несправностей, ідентифікацію

запитів, які обчислював сервер, що не працює, повторне надсилання цих запитів на альтернативний сервер та вжиття заходів щодо омолодження, таких як перезапуск невдалих серверів.

Питання коду інфраструктури відмовостійкості для кожного додатка є складним та схильним до помилок процесом, що ускладнює розробникам програм швидке розпаралелювання існуючих фінансових програм.

Завдання 5: Управління паралельністю. Обчислювальні програми для фінансування, такі як обчислення VaR у нашому прикладі, часто дуже обчислювальні. Тому ці програми можуть отримати значну користь від належного управління паралельністю, коли всі ядра в багатоядерному процесорі ефективно використовуються для оптимізації розрахунків. Програмування цих проблем вимагає від розробників додатків явного управління паралелізмом шляхом створення потоків та синхронізації цих потоків з повідомленнями та блокуваннями. Цей процес необхідно повторити для кожної платформи, оскільки API програмування потоків відрізняються від платформи до платформи, наприклад, відмінності в API потоків між Windows і Linux. В ідеалі розробники додатків повинні розробляти вихідний код на основі способу агностики платформи, щоб запити на додатки могли бути оптимізовані залежно від наявності одноядерних та багатоядерних процесорів. може вирішити описані вище розподілені та паралельні проблеми розробки додатків, пов'язані з великомасштабними програмами обчислювального фінансування. інтенсивні програми в мережевому середовищі.

- Test Configuration Environment (TCE) - це утиліта конфігурації програми, яка виявляє, перевіряє та управляє всіма програмами в розгортанні. Він управляє обчислювальними серверами, клієнтами та утилітами моніторингу та забезпечує IP-адреси та багатокадрові адреси для розподіленого середовища виконання.
- zNet - це оптимізована система балансування навантаження, пов'язана з клієнтськими програмами і, отже, знаходиться в адресному просторі клієнта. zNet автоматично розподіляє обчислення на всіх доступних серверах, прозоро паралелізує виконання масштабованим, відповідальним та ресурсозберігаючим способом, а також покращує продуктивність на порядки порівняно зі звичайними техніками програмування.
- zEngine, який є обчислювальним серверним контейнером, який встановлюється і запускається на (потенційно неоднорідних) цільових машинах. Це контейнер, в якому фактично виконуються паралелізовані обчислення. ZEngine використовує основні механізми планування операційної системи (тобто створення потоків, синхронізація та управління потоками, що знають ядро), щоб максимізувати використання процесора, виконуючи екземпляр паралелізованої функції на кожному ядрі (загальна практика полягає в тому, щоб запускати якомога більше zEngineinstance на кожному хості, оскільки є ядра процесора).
- zPluginbuilder - це утиліта, яка використовується для адаптації бібліотеки послідовних клієнтів у паралелізуються бібліотеки плагінів, які можуть розпаралелювати складні обчислення за допомогою проміжного програмного забезпечення zNet.
- zAdmin, який є утилітою для управління (тобто моніторинг, інсталювання, запуск та зупинка) ресурсів та програм у системі або графічно, або за допомогою командного рядка.

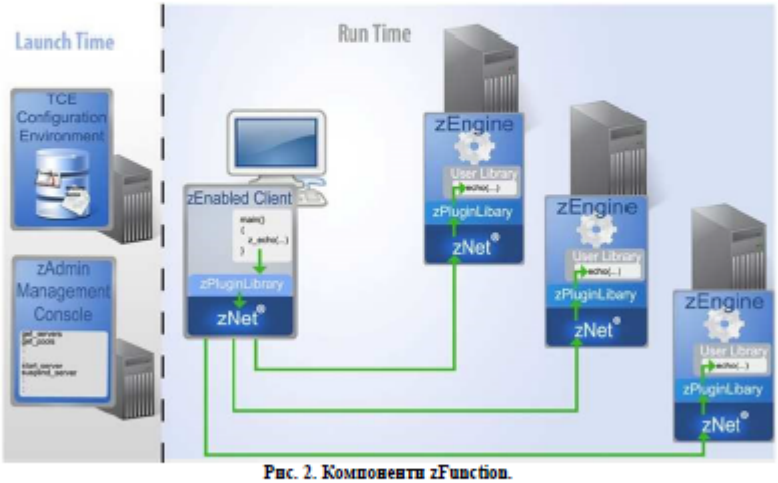

Рис. 2. Компоненти zFunction.

Увімкнення за допомогою zFunctionAdapters та zPluginLibrariesБудь-яке послідовне застаріле додаток, яке виконує складні обчислення на великих наборах даних, можна розпаралелювати за допомогою zFunction. Паралелізація послідовного додатка (який ми називаємо zEnabling) передбачає кроки для зв'язку програми з проміжним програмним забезпеченням zFunction, що прозоро охоплює проблеми розподіленої та паралельної обробки від додатків.

ThezEnablingprocess захищає розробників додатків від проблем низького рівня розподілу, таких як виявлення, адресація, (de) маршируючи запити та відповіді та займаючись змінами в стеках мережевого протоколу, що дозволяє відмовитись, так що програми можуть безперешкодно взаємодіяти з будь-якою платформою та мовою програмування.

zEnabledapplication містить еквівалент zFunctionAdapterz_F для кожної функції, що паралелізується. Розробникам клієнтських додатків потрібно лише замінити виклики toF з викликами toz_F для розпаралелювання.

ZFunctionAdapterz_F - це проксі-сервер на стороні клієнта, який прозоро розподіляє асинхронні запити на zEngines, забезпечуючи тим самим адаптивні, розподілені та високопродуктивні обчислення на попит на клієнтські програми. zFunction використовує інструмент zPluginBuilder для бібліотек zEnablinguser.

Вхідними даними інструменту zPluginBuilder є файл XML, що описує функцію F, її вхідні параметри, вихідні параметри та розташування бібліотеки, що містить визначення функції F (показано рисунка 3). .

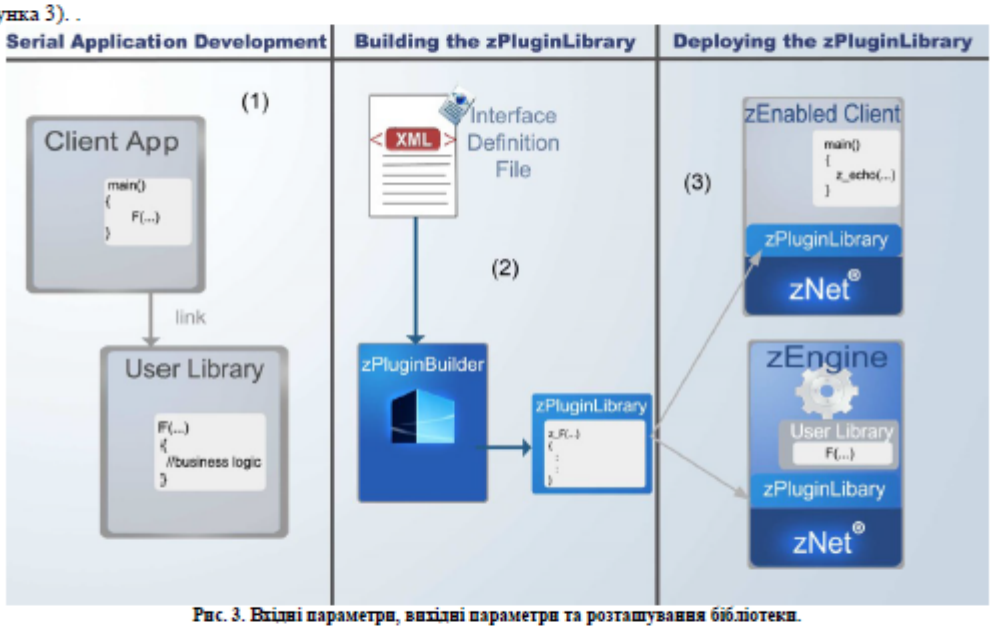

Рис. 3. Вхідні параметри, вихідні параметри та розташування бібліотеки.

Вихід - бібліотека(називається zPluginLibrary) з реалізацією zFunctionAdapterz_Fконформ, що відповідає тому ж інтерфейсу, що і вихідна функція F. Створена zPluginLibrary пов'язана як клієнтською програмою, так і zEngine (див. праву сторону малюнка 3). На сервері zPluginLibrary просто делегує дзвінки, зроблені з клієнтської zPluginLibrary (від імені клієнтських програм), до функціїFdefined у бібліотеці, створеній службою розробників. Таким чином, користувачі zFunction, маючи мінімальну кількість зусиль для розробки, отримують універсальну паралелізовану програму якісного виробництва, яку можна розгорнути в мережі паралельних обчислювальних вузлів компоненти, показані на рис. 2, стосуються розподілених ключів та розпаралелізують завдання проектування додатків.

Вирішення проблеми 1: Надання інформаційної послуги для виявлення та адресація віддалених обчислювальних серверів. Конфігураційне середовище (TCE) діє як інформаційна служба для zFunction та завантажує всі програми в мережі. Інші компоненти у розгортанні zFunction (включаючи клієнтів та zEngines, які виконують віддалені обчислення) реєструються в TCE під час запуску. Цей процес дозволяє TCE визначати параметри мережі, такі як IP-адреси хоста, ідентифікація підмережі мережі, адреси багатоадресної передачі. TCE використовує протокол рукостискання, який забезпечує мережеву інформацію для всіх компонентів zFunction, завдяки чому додатки можуть спілкуватися між собою під час виконання, не співпрацюючи з TCE.

zFunction дозволяє розробникам додатків оптимізувати роботу системи, забезпечуючи гнучкі механізми розповсюдження даних. zFunctionclients не надсилають дані з кожним запитом; натомість дані

надсилаються лише один раз, і з кожним запитом zFunction надсилає посилання на кожен сервер, де дані можна знайти. Більше того, якщо нові дані потрібно оновлювати в середині обчислень, zFunction також надає механізм сигналізації всіх серверів і дозволу їм досягти загального знімка або контрольної точки, отримати нові вхідні дані від клієнта, а потім відновити обчислення.

ZFunction надає утиліту zPluginBuilder, яка автоматично генерує zPluginLibraries, що служать адаптерами між загальним проміжним програмним забезпеченням zFunction та конкретними програмами клієнт / сервер. Ці адаптери випускають ефективний (де) маршалінговий код, що дозволяє проміжному програмному забезпеченню zFunction прозоро підтримувати віддалену комунікацію на різнорідних платформах і мережах.

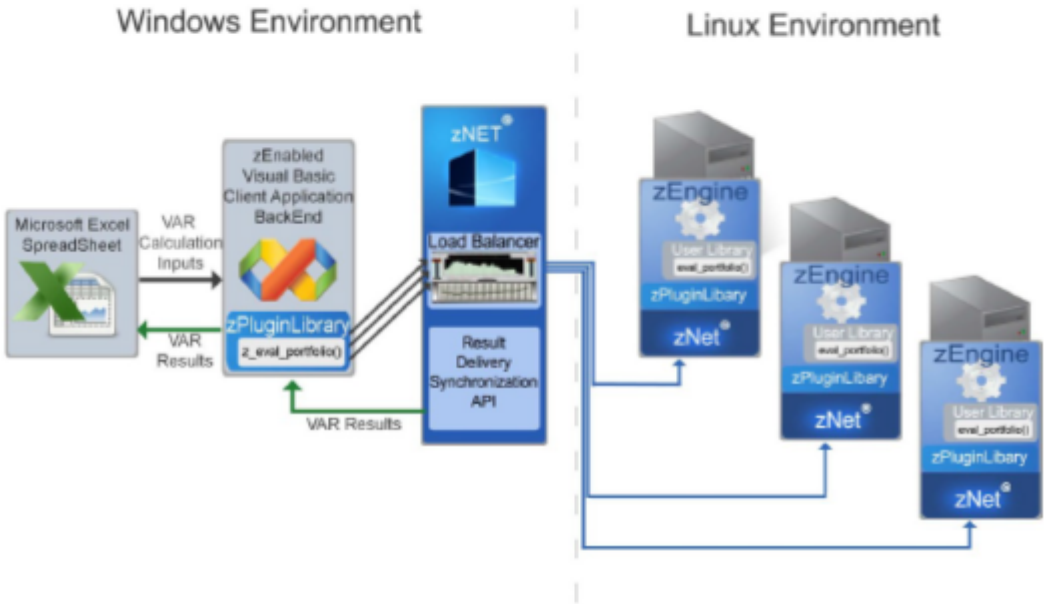

Рис. 4. Рівномірний розподілу роботи між існуючими комп'ютерними серверами.

Вирішення проблеми 3: Забезпечення ефективного управління ресурсами віддалених обчислювальних серверів. Коли повторні запити клієнта zEnabled надсилаються в пул серверів, інтелектуальний балансувальний балансир проміжного програмного забезпечення zFunction використовується для рівномірного розподілу роботи між існуючими комп'ютерними серверами в режимі реального часу, як показано на малюнку 4. Розподіляючи обчислення рівномірно на всіх доступних серверах, zFunctionмаксимізує розподіл ресурсів для критично важливих додатків, а також гарантує, що апаратні ресурси використовуються в повній мірі.

Вирішення проблеми 4: Забезпечення прозорості для багаторівневого багатошарового захисту. zFunction також забезпечує виконання додатків незалежно від збоїв обладнання та прозоро забезпечує відновлення за замовчуванням та відмову, повторно виконуючи запити на серверах, які все ще працюють. zFunction відстежує історію виконання кожного запиту і на яку zEngine було надіслано запит. Коли zFunction виявляє, що zEngine вийшов з ладу, він автоматично відправляє запит на новий або омолоджений двигун і гарантує, що обчислення виконуються незалежно від збоїв обладнання. -ures.

Вирішення проблеми 5: Забезпечення неявної масштабованості за допомогою багатопоточності, що усвідомлює основи. zФункція досягає розпаралелювання за допомогоювиконання декількох екземплярів паралелізованої функції програми одночасно в процесах zEngine, що виконуються на різних машинах у мережі. zFunction забезпечує неявну підтримку паралельності та автоматично створює потоки для розподілу запитів на різні сервери, а також синхронізує ці потоки за допомогою mes-sage та блокування.

Оновлену програму VaR, яка використовує zFunc -tion для паралелізації обчислень на портфелі акцій та опціонів, представлених як електронна таблиця Microsoft Excels. Як обговорювалося вище, проміжне програмне забезпечення zFunction дозволяє ефективно роз'єднати клієнтський код і реалізації функції, що паралелізується, щоб вони могли працювати на різних платформах, наприклад, Windows для клієнта та Linux для серверів. Більше того, різні частини програми theVaR також можуть бути написані різними мовами, наприклад, Vi-sual Basic для клієнта та C / C ++ для серверів. Інтерфейс між клієнтом та

функцією розпаралелювання вказаний стисло у файлі опису інтерфейсу на основі XML Потім інструмент zPluginBuilder використовується для створення бібліотек плагінів, що підходять як для клієнтської програми, так і для реалізації сервера zEngine. Для цього прикладу клієнтом є Microsoft Excelsprepreheet, який використовує COM-інтерфейс (через Visual Basic) для інтеграції з проміжним програмним забезпеченням zFunction. І навпаки, сервери розгортаються в пулі zEngines, що розгортає бібліотеки Linux C / C ++ для розрахунку VaR. 4.1z Впровадження клієнта на основі функцій Клієнтський код цього додатка міститься в Microsoft Excelworkbook, який містить дані про сценарії поведінки на ринку (див. Рис. 1), що є природним і типовим середовищем для фінансових розрахунків із великим обсягом даних.

Як показано на малюнку 5, клієнт тому може легко підтримувати застарілу послідовну реалізацію розрахунку, написаного у програмі Visual Basic для додатків. На цьому малюнку також показано, як theEEnabledapplication вимагає двох поверхневих перетворень коду на стороні клієнта, вбудованого в електронну таблицю:

- Обчислення надсилаються асинхронно, а функція зворотного виклику з роздільною швидкістю отримує відповіді від віддалених zEng-ines і заповнює цільові клітинки в результаті електронна таблиця з цими відповідями.
- Інструмент zPluginBuilder використовується для генерації клієнтського COM-інтерфейсу, що дозволяє викликати відповідну функцію з коду Visual Basicclient.

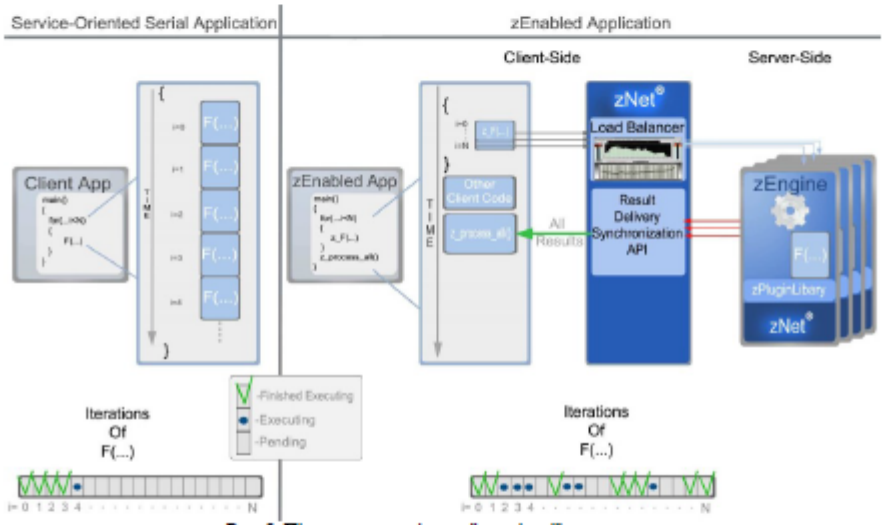

Рис. 5. Підтримка послідовної реалізації розрахунку

Реалізація сервера на основі функцій При розробці серверного коду isEnabledapplication необхідно лише (1) створити користувацьку бібліотеку, що містить функцію, що паралізується, та (2) описати інтерфейс функції за допомогою файлу опису інтерфейсу (у XML). Інструмент zPluginBuilder використовується для створення бібліотеки плагінів, яку можна динамічно завантажувати в zEngines, що працюють на будь-якій підтримуваний платформі, як показано на рис. 3. Реалізація сервера складається з функції aneval_portfolio (), вхідні параметри якої включають визначення портфеля та ціну акцій сценарію та використовує біноміальну модель ціноутворення для оцінки всіх варіантів у портфелі. Кінцевим результатом обчислення для одного сценарію є вартість портфеля. Таким чином, кожен сценарій (який визначається окремим набором гіпотетичних цін на акції в кінці часового горизонту моделювання) дає незалежний та паралельний розрахунок вартості портфеля.

Як показано на малюнку 5, zEngines, призначені для обчислення, можуть ефективно виконувати всі ці незалежні сценарії обчислень, за допомогою клієнтського балансу навантаження, інтегрованого в проміжне програмне забезпечення zNet, розподіляючи роботу автоматично.Переваги застосування zFunction до тематичного дослідження VAR Автоматизований процес активації zEnabling вирішує всі проблеми, наведені вище, з якими стикаються розробники обчислювальних програм для фінансування.

Як показано на малюнку 5, для вирішення проблем 1, 3, 4 та 5 проміжне програмне забезпечення zFunction автоматично забезпечує виявлення, адресацію, балансування навантаження, виявлення несправностей та механізми відновлення, гарантуючи, що всі клієнтські запити, передані йому, будуть виконуватися в будь-якому випадку, незалежно від зв'язку або відмови сервера. Ця відмовостійкість забезпечується компонентами zFunction по обидва боки мережі, що не вимагає зусиль розробника додатків.

Специфічна для програми zFunction, згенерована zPluginBuilder, також капсулює проблеми, пов'язані з надійними розподіленими обчисленнями за інтерфейсом, подібним до інтерфейсу, аналогічного функції синхронної розпаралелізації, тим самим підвищуючи рівень абстракції програмування, досвідчений розробниками додатків. на zFunction з пов'язаною роботою з паралельної розробки та розгортання додатків.

Спеціально-орієнтоване програмування (AOP). Нещодавня робота зосереджена на використанні AOP [13] для відокремлення питань розпаралелювання щодо вихідного коду, специфічного для застосування [10, 17, 14]. Однак для того, щоб забезпечити можливості в режимі реального часу, такі як відмовостійкість, балансування навантаження та розповсюдження даних за допомогою AOP, потрібно використовувати новіші технології, що підтримують композицію аспектів.

zFunction забезпечує всі ці переваги з мінімальними модифікаціями існуючих додатків. Проміжне програмне забезпечення для обчислювальної мережі. Багато проектів досліджували ідею використання розподілених обчислювальних архітектур для прискорення складних обчислень за кластером комп'ютерів.

Деякі добре відомі приклади включають проекти SETI @ Home [3] та BOINC [2], в яких використовуються недостатньо використовувані мережеві процесори для виконання загальних завдань.

Аналогічним чином, Frontier (www.frontier.com) надає програмне забезпечення для мереж для використання доступних процесорів для прискорення паралельних програм.

Загалом, у цих підходах клієнтські вузли спілкуються через централізований головний вузол для подання завдань, що може збільшити затримку, створити вузькі місця у роботі та дає єдину точку відмови. На відміну від цього, zFunction забезпечує надзвичайно оптимізовану та децентралізовану інфраструктуру проміжного програмного забезпечення для розпаралелювання додатків, міжпроцесорного зв'язку та розподілу даних.

Проміжне програмне забезпечення для прискорення застосувань фінансового інжинірингу. Попередня робота також була зосереджена на розробці та / або застосуванні архітектур сітки та мережевих додатків для застосування фінансових послуг. Наприклад, [16] обговорюється практичний досвід, пов'язаний з проблемами управління даними та продуктивністю, що виникають при розробці додатків для фінансових послуг у суперкомп'ютері IBM Bluegenesuper [1].

Аналогічним чином, PicsouGrid [4] - це відмовостійка та багатопарадигмна програмна архітектура сітки для прискорення фінансових обчислень у великій сітці. Інші системи, що базуються на сітці, включають обчислення платформи (www.platform.com), Data-Synapse (www.datasynapse.com) та Microsoft HPC (www.microsoft.com/hpc), які забезпечують розподілене програмне середовище для фінансових розрахунків. zFunction відрізняється від цих технологій простотою використання та інтеграції, ефективністю роботи в режимі реального часу, здатністю обробляти як малі, так і великі обчислення, підтримкою портативних архітектур та платформ, а також своїми вдосконаленими функціями паралельного програмування, такими як прозора додаткова відмовостійкість, балансування навантаження та неявне програмування потокової спільної пам'яті.

Висновки. Цей документ продемонстрував можливості проміжного програмного забезпечення zFunction, яке може паралелізувати складні обчислювальні та інтенсивно розподілені програми з використанням спрощеного режиму програмування, що створює адаптивне середовище розподіленого компіляційного середовища в режимі реального часу, що захищає від несправностей, на вимогу zFunction добре підходить для доменних мереж, де складні розрахунки в реальному часі потрібні швидко передбачувати, і які можуть отримати вигоду від розподіленої обробки робочого навантаження через (потенційно неоднорідну) мережу. Це може значно підвищити продуктивність таких систем за низьку ціну, дозволяючи додаткам паралельно працювати на апаратному забезпеченні COTS, робочих столах, кластерах та хмарі.

Слід докласти значних зусиль для тестування паралельних систем та виявлення помилок під час виконання, але особливо для систем, що використовують різнорідні моделі програмування, а також дворівневі та трирівневі моделі програмування. Інтеграція різних моделей програмування в одну і ту ж систему потребуватиме нових методів тестування для виявлення помилок під час виконання в неоднорідних паралельних системах Для досягнення хороших систем, які можна використовувати в суперкомп'ютерах, слід зосередитись на тестуванні систем через їх паралельну природу. Крім того, ці інструменти тестування повинні інтегрувати декілька методів тестування і паралельно працювати над виявленням помилок під час виконання шляхом створення тестових потоків залежно від кількості потоків додатків.

Як результат, використання паралельних гібридних методів збільшить час тестування та охопить широкий спектр помилок під час виконання.

References

1. et. al. Allen, F. Blue gene: a vision for protein science using a petaflop supercomputer. IBM Syst. J., 40(2):310–327, 2011.
2. D. P. Anderson. Boinc: A system for public-resource computing and storage. In GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pages 4–10, Washington, DC, USA, 2014. IEEE Computer Society.

3. D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. Commun. ACM, 45(11):56–61, 2016.

4. S. Bezzine, V. Galtier, S. Vialle, F. Baude, M. Bossy, V. D. Doan, and L. Henrio. A fault tolerant and multi-paradigm grid architecture for time constrained problems. application to option pricing in finance. In E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, page 49, Washington, DC, USA, 2006. IEEE Computer Society.

5. F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. The Journal of Political Economy, 81(3), May 1973.

6. J. C. Cox, S. A. Ross, and M. Rubinstein. Option Pricing: A Simplified Approach. Journal of Financial Economics, 4, 1979.

7. D. Duffie and J. Pan. An Overview of Value At Risk. The Journal of Derivatives, 4(3), Apr. 1997.

8. M. Forum. Message Passing Interface Forum. www.mpi-forum.org.

9. P. Glasserman. Monte Carlo Methods in Financial Engineering (Stochastic Modeling and Applied Probability). Springer Verlag, 2017.

10. B. Harbulot and J. R. Gurd. Using aspectj to separate concerns in parallel scientific java code. In AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development, pages 122–131, New York, NY, USA, 2016. ACM.

11. J. C. Hull. Risk Management and Financial Institutions. Prentice Hall, Upper Saddle River, NJ, 2006.

12. P. Jorion. Value at Risk: The New Benchmark for Managing Financial Risk. McGraw-Hill, New York, NY, third edition, 2016.

13. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In Proceedings of the 11th European Conference on Object-Oriented Programming, pages 220–242, June 1997.

14. M. E. F. Maia, P. H. M. Maia, N. C. Mendonca, and R. M. C. Andrade. An aspect-oriented programming model for bag-of-tasks grid applications. In CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pages 789–794, Washington, DC, USA, 2007. IEEE Computer Society.

15. OpenMP. OpenMP Home Page. www.openmp.org.

16. T. Phan, R. Natarajan, S. Mitsumori, and H. Yu. Middleware and performance issues for computational finance applications on blue gene/l. Parallel and Distributed Processing Symposium, International, 0:371, 2019.

17. J. Sobral. Incrementally developing parallel applications with aspectj. Parallel and Distributed Processing Symposium, International, 0:95, 2016.