УДК 004.415.2:004.75

СИСТЕМА КЕРУВАННЯ ГРІД-ЗАВДАННЯМИ ВІРТУАЛЬНОЇ ЛАБОРАТОРІЇ, ЩО ГРУНТУЄТЬСЯ НА АСИНХРОННІЙ ОБРОБЦІ ПОДІЙ

А.О. Сальніков

Київський національний університет імені Тараса Шевченка,
01601, м. Київ, вул. Володимирська 64,
тел.: +380 44 2590247; факс: +380 44 5261214; e-mail: manf@grid.org.ua

Розглянуто методики та засоби взаємодії віртуальних лабораторій з грід-інфраструктурами. Розроблена архітектура системи керування грід-завданнями віртуальної лабораторії, що грунтується на асинхронній обробці подій. Створено прототип сервісу, що реалізує запропоновану архітектуру з назвою MAGGIE – Main Architecture Gears for Grid Integrated Environment.

Methods and facilities of interaction between virtual research environments and grid-infrastructures are presented. Asynchronous event-driven architecture for grid-jobs management for building virtual research environments has been developed. Prototype service implementation named MAGGIE – Main Architecture Gears for Grid Integrated Environment – was developed.

Вступ

Визначення грід по-різному сформульовані в літературі, проте всі вони висвітлюють множину характеристик, які формують концепцію системи [1, 2]:

- масштабованість у широких межах (large scale) – система має працювати при зміні кількості ресурсів від одиниць до мільйонів;
- географічний розподіл (geographical distribution) – грід-ресурси можуть знаходитись у різних містах чи країнах;
- гетерогенність (heterogeneity) – як апаратна, так і програмна реалізація різних ресурсів може бути принципово різною;
- розподіл ресурсів (resource sharing) – ресурси грід належать різним незалежним установам;
- розподіл управління (multiple administrations) – кожна установа встановлює власні політики безпеки та права доступу до сервісів ресурсу;
- координування ресурсів (resource coordination) – забезпечення злагодженої роботи для агрегації обчислювальних потужностей, дискового простору систем зберігання тощо;
- прозорий доступ (transparent access) – для кінцевого користувача грід надає єдину точку входу для доступу до агрегованих ресурсів;
- залежний від умов доступ (dependable access) – грід надає ресурси у відповідності до встановлених вимог якості обслуговування (QoS);
- адаптивний доступ (Pervasive access) – грід надає ресурси відповідно до динамічно змінюваного стану системи, де відмова одного з ресурсів є штатною ситуацією.

Така інфраструктура, що побудована для забезпечення доступу до розподілених ресурсів, є лише фундаментом для вирішення ресурсоємних наукових задач в різних предметних галузях. Модель взаємодії дослідників, що вирішують такі задачі з грід-інфраструктурою будується в рамках концепції віртуальних організацій (ВО).

Віртуальна організація – це динамічне об'єднання користувачів, установ, обладнання та сервісів для доступу до ресурсів грід-інфраструктури [1]. ВО є найменшою одиницею з якою взаємодіє кожен з провайдерів ресурсів грід-інфраструктури. Окремі користувачі отримують ресурси виключно як учасники певної ВО, відповідно до встановлених домовленостей. Таким чином агрегація ресурсів, створення єдиної точки входу та встановлені вимоги (резервації, пріоритети завдань, тощо) до ресурсів диктують та забезпечують різні віртуальні організації.

Для вирішення проблеми взаємодії з дослідниками грід-інфраструктура надає набір методів, які забезпечують роботу віртуальних організацій з сервісами грід. Такі методи визначає відкрита архітектура грід-систем (Open Grid Services Architecture – OGSA). Відповідно до відкритої архітектури різні реалізації програмного забезпечення проміжного рівня грід, на яких будуються інфраструктури грід-сервісів, повинні бути інтероперабельними [3].

Проте для використання грід-інфраструктури в наукових дослідженнях вченими, необхідні інтерфейси, які забезпечують прозорий доступ до наявних динамічних ресурсів без необхідності поглиблення в архітектуру системи та знань про множину сервісів.

На сьогодні задача створення автоматизованих систем керування завданнями остаточно не вирішена і саме побудові таких систем присвячена дана робота.

Віртуальні науково-дослідні співтовариства (VRC) та лабораторії (VRE)

Віртуальне науково-дослідне співтовариство (*Virtual Research Community, VRC*) – це група дослідників, можливо розділена широкою географією, які співпрацюють разом через використання інформаційних та телекомунікаційних технологій [4].

Всередині співтовариства дослідники взаємодіють, спілкуються, обмінюються ресурсами, доступом до віддаленого вимірювального обладнання чи обчислювальних машин. Метою VRC є використання віддалених ресурсів настільки ефективно, наскільки це можливо при фізичному доступі до них.

У залежності від завдань та досліджуваних проблем, VRC можуть бути закритими чи відкритими для участі, формалізованими чи неформалізованими, з внутрішньою структурою чи без неї. Розвиваючись, VRC можуть динамічно змінювати такі характеристики, так само як і предмети досліджень [4].

Визначення VRC не прив'язане до використання певних технологій. Існує підмножина VRC, що використовує грід як інструментарій для проведення досліджень. Інтеграція таких VRC відбувається через використання ключової концепції грід – віртуальної організації.

ВО може бути представленням VRC для грід-інфраструктури, або може існувати незалежно. Так само і віртуальне науково-дослідне співтовариство може існувати незалежно, а може бути представлено в грід за допомогою однієї чи декількох ВО. Представлення декількома ВО має історичний аспект (об'єднання декількох ВО) або реалізовано навмисно для розділення напрямків досліджень, що використовують програмне забезпечення з різними вимогами до ресурсів.

Доступ дослідників до сервісів VRC зазвичай відбувається через віртуальну лабораторію.

Віртуальна лабораторія (*Virtual Research Environment, VRE*) – об'єднання онлайн-утиліт, сервісів та ресурсів для ефективного проведення досліджень без прив'язки до певної установи чи організації. Метою VRE є надання зручних інтерфейсів до утиліт та сервісів, які необхідні для дослідження проблеми. Розробка інтерфейсів VRE напрямлена на доступність для дослідників з будь-якої апаратної та програмної платформи та прозоре використання ресурсів інфраструктури.

Віртуальні лабораторії орієнтовані на роботу з дослідниками, що не мають додаткових знань про внутрішню архітектуру та алгоритми роботи програмного забезпечення інфраструктури чи VRE.

Програмні засоби VRE додатково передбачають можливості взаємодії та обміну даними між різними спільнотами дослідників, зазвичай поміж групами, що пов'язані однією областю досліджень чи географічним розташуванням.

Процес дослідження з використанням VRE включає: пошук ресурсів, керування зберіганням даних, аналіз даних, моделювання, спільну роботу, обговорення, публікацію результатів, керування дослідженнями та проектом вцілому. VRE, що слугують інтерфейсами до грід-середовища додатково реалізують специфічні для грід алгоритми: забезпечення авторизації, забезпечення делегації, робота з внутрішньою структурою ВО, робота з сервісами рівня ресурсів та кооперації грід.

Віртуальна лабораторія таким чином слугує шлюзом до інфраструктури грід або інших розподілених ресурсів. Такий шлюз зазвичай представлений в вигляді веб-інтерфейсу, що може бути доступним будь-де.

Автоматизація наукових досліджень в грід

Завдяки своїй розподіленій природі, грід-середовище найбільш ефективне для паралельного виконання великої кількості незалежних завдань, що використовують різні набори вхідних даних чи параметрів. Для роботи багатьох науково-дослідних співтовариств, що відображені на грід-розрахунки за допомогою віртуальних організацій, така структура завдань є природньою. Це включає як проведення досліджень (моделювання) поведінки множини різник об'єктів, так і аналіз одного об'єкта за різних початкових умов чи параметрів [5].

Якщо для проведення наукових досліджень кількість таких завдань до розрахунку сягатиме декількох сотень чи тисяч, то необхідні механізми автоматизації такого процесу. Без автоматизації дослідники змушені вручну контролювати процес підготовки вхідних даних та наборів параметрів для кожного з тисячі завдань, виконувати постановку всіх цих завдань в грід, слідкувати за станом кожної задачі та надсилати заново задачі, що були втрачені. Після успішного закінчення всіх завдань проекту, необхідним кроком є отримання результатів виконання кожної задачі та їх агрегація для подальшого аналізу проблеми.

Більш того, виконання таких досліджень вручну потребує додаткових знань поза предметною областю дослідника, а саме внутрішньої структури та роботи грід-середовища, семантичних мов опису завдань грід, навичок роботи з командним рядком інтерфейсу користувача грід для надсилання завдань, моніторингу, керування та передачі даних.

Ще одним недоліком виконання розрахунків без засобів автоматизації є людський фактор. При зростанні кількості завдань імовірність помилитися у вхідних даних чи втратити прив'язку до ідентифікатора завдання зростає. Таким чином ефективне та надійне проведення наукових досліджень, що потребують проведення інтенсивних розподілених обчислень фактично неможливе без автоматизації [6].

Перед системами автоматизації стоїть завдання вирішення ряду проблем для проведення розрахунків в грід-середовищі:

- авторизація користувачів – визначення параметрів участі користувачів у віртуальній організації за їх ідентифікацією;

- делегація повноважень користувачів – передача повноважень користувача грід-сервісам (наприклад, звертання до елемента зберігання обчислювального елементу від імені власника завдання);
- база даних завдань – керування списками завдань та проектів (включаючи їх ідентифікатори в грід-інфраструктурі, зв'язки між завданнями, авторство, тощо). База даних необхідна для зв'язку між надсиланням завдання, моніторингом та обробкою результатів виконання, що в загальному випадку є незалежними процесами;
- надсилання завдань в грід-інфраструктуру – формування опису завдання відповідно до вказаних критеріїв та параметрів, передача завдання брокеру грід-інфраструктури;
- передача вхідних даних грід-завдань – робота з вхідними файлами та параметрами, забезпечення їх доступності для грід-інфраструктури;
- моніторинг стану грід-завдань – контроль стану завдань в грід-інфраструктурі та кореляція з інформацією в базі даних; контроль за графами зв'язаних потоків завдань (workflow);
- керування завданнями в грід – контроль валідності делегації користувача для виконання завдання, перезапуск втрачених завдань, тощо;
- обробка запитів щодо стану завдань – надання дослідникам інформації про стан розрахунку проекту;
- робота з журналами виконання завдань – надання доступу та аналіз журналів виконання завдань в грід;
- отримання та агрегація результатів обрахунку проекту – надання дослідникам доступу до кінцевого результату дослідження.

Наявні засоби створення віртуальних лабораторій використовують різні алгоритми для реалізації цих ключових елементів автоматизації.

Аналіз засобів створення віртуальних лабораторій

Всі існуючі засоби побудови віртуальних лабораторій можна розділити на засоби загального призначення та специфічні реалізації для певних проектів. Так, наприклад, проекти обробки даних великого адронного колайдера (ALICE, CMS) мають власні системи керування грід-завданнями, що орієнтовані саме на роботу з обладнанням. Об'єктом дослідження роботи є системи, що не прив'язані до конкретного експерименту. Розглянемо найбільш поширені з них – Gridsphere та Lunarc Application Portal.

Gridsphere. Базове середовище розробки (framework) Gridsphere створено об'єднанням дослідників університету Каліфорнія (Сан Дієго), центру суперкомп'ютингу та мереж (Познань) та Інституту Альберта Ейнштейна. Цей проект продовжує розвиватись.

Як серверна, так і клієнтська частина даного продукту основані на технології Java (клієнтська частина – Java Web Start). Серверна частина виконується під керуванням веб-серверу Apache Tomcat. Робочою одиницею для організації обчислень в Gridsphere є «потік задач» (worklflow). Його можна редагувати, запускати на виконання і спостерігати за процесом обчислень на віддалених ресурсах. Права на доступ до них визначаються за сертифікатом користувача.

Процес постановки набору задач на виконання відбувається шляхом синхронного звернення до служби WMS за запитом користувача з інтерфейсу. Gridsphere побудовано за допомогою використання бібліотек набору програмного забезпечення проміжного рівня gLite, тому його використання обмежено лише відповідними інфраструктурами [7].

Worklflow є направленим ациклічним графом, для кожної вершини якого вказано обчислювальний ресурс і програму для обробки даних (задачу), що буде на ньому виконуватись. Ребра цього графа відображають інформаційні канали, що з'єднують вхідні і вихідні точки (що в термінології Gridsphere називаються портами) окремих задач. Вузлом називається обгортка задачі, що містить посилання на виконувану програму, її вхідні і вихідні зв'язки, а також обчислювальний ресурс. Вхідні порти вузлів, що не з'єднані із жодним із вихідних портів інших вузлів представляють вхідні файли даного набору задач. Відповідно вихідні порти, що не під'єднані до жодного із вхідних являють собою вихідні файли завдання.

Метою використання потоків задач є створення складного комплексу завдань, що на основі вхідних файлів проведе ряд операцій для отримання вихідних. При цьому дві задачі, що відносяться до окремих вузлів можуть виконуватись паралельно якщо вони незалежні, тобто одна із них не має прив'язки до даних іншої і всі вхідні дані є доступними. Таким чином, порядок виконання задач визначається структурою направленого графа набору задач.

Авторизація для роботи з порталом реалізована шляхом перевірки імені та паролю, які користувач отримує при реєстрації у системі. Для делегації роботи в грід використовується сервер MyProxy. Параметри участі в ВО не враховуються.

Створення потоків задач та зазначення вхідних файлів та параметрів потоку для кожного обчислення виконується дослідником вручну, що для масових розрахунків не є оптимальним.

Lunarc Application Portal (LAP). Інша розробка порталу, що створена для роботи в грід – Lunarc Application Portal – належить центру наукових та технічних обчислень міста Лунд (Lund Center for Scientific and Technical Computation).

Робота порталу базується на сервері застосувань WebWare для Python, що являє собою набір програмних компонент для створення об'єктно-орієнтованих веб-застосувань. Реалізацією WebWare для Python є модуль, що компілюється для веб-серверу Apache.

Відповідно для створення інтерфейсів користувача до певного програмного забезпечення, надано набір вбудованих бібліотек для мови Python. Бібліотеки дозволяють розробити CGI сценарій інтерфейсу, що інтегрується в портал. Функції постановки завдання в грід також покладаються на створений сценарій і реалізовані в зазначених бібліотеках. Взаємодія здійснюється тільки з грід середовищем, що працює під управлінням ARC версії 0, і вимагає наявності бібліотек інтерфейсу користувача (а саме інтерфейсів ARC API мови Python) [8].

На відміну від GridSphere, структурною одиницею для LAP є одне завдання. Дослідники мають можливість поставити на виконання одне з завдань у відповідності до попередньо створених шаблонів, вручну завдавши вхідні файли та параметри, що так само не сприяє масовим розрахункам. Постановка завдання в грід-інфраструктуру відбувається синхронно по запиту з інтерфейсу користувача шляхом виклику API вбудованого брокеру завдань ARC.

Для того щоб розпочати роботу з порталом необхідно завантажити свій валідний проксі-сертифікат на сервер, що одночасно виконує функції ідентифікації користувача і надає можливість делегувати завдання в грід. Якщо дослідник працює з операційною системою Linux, то функції генерації проксі-сертифікату покладаються на інтерфейс командного рядку ARC.

Для операційної системи Windows розробники створили застосування «Grid Certificate Manager», що дозволяє не тільки генерувати проксі-сертифікати на основі ключів, але й посилати запит на отримання та продовження сертифікату користувача.

Серед українських проектів слід виділити віртуальну лабораторію MolDynGrid та дослідження ВО NetworkDynamics як такі, що ілюструють наявні проблеми синхронного керування завданнями і є цільовими проектами для першого застосування результатів представленої роботи.

MolDynGrid. Віртуальна лабораторія MolDynGrid є флагманським проектом української національної грід-інфраструктури з проведення обчислень в грід-середовищі. Вона вирішує задачі в галузях структурної біології і біоінформатики, які потребують значних витрат машинного часу та оперують великими об'ємами інформації [5].

В цьому проекті було впроваджено методи прозорої авторизації на грід-порталі за допомогою персонального сертифікату користувача імпортованого в веб-браузер та вирішено проблему автоматичного отримання делегації на базі цієї авторизації. Портал представлений в грід-інфраструктуру своїм власним сертифікатом сервісу і звертається до серверу MyProxy як авторизований для отримання делегаії (authorized retriever).

Іншою особливістю MolDynGrid, що дозволила ефективно і прозоро використовувати грід-інфраструктуру було впровадження інтерфейсу віддаленого виклику процедур. Розробка веб-порталу здійснювалась паралельно з модифікацією консольних утиліт запуску завдань. Старий метод роботи досвідчених користувачів через консоль прозоро взаємодіяв з порталом і перехід відбувався поступово.

Запуск завдань також відбувається синхронно в потоці веб-серверу, згідно до запиту користувача через майстер постановки завдання інтерфейсу і є придатним для довготривалих розрахунків динаміки, де кількість завдань маленька. Для задач аналізу, де час розрахунку зменшується, проте кількість їх зростає до сотень необхідність послідовного запуску майстра не є ефективною і зручною для користувачів.

NetworkDynamics. Віртуальна організація NetworkDynamics вирішує задачі моделювання нелінійних систем, а саме поведінки нейронів в рамках реалістичних фізичних моделей. На відміну від MolDynGrid, де моделюється поведінка системи з наперед заданим набором параметрів, NetworkDynamics займається задачею підбору параметрів, які призводять до заданого результату (наприклад, синхронізації нейронів мозку при хворобі Паркінсона). Таким чином кількість задач які необхідно обрахувати для перебору параметрів сягає десятків та сотень тисяч.

При нормальній роботі всіх інформаційних систем процес надсилання однієї задачі в грід складає декілька десятків секунд. Для надсилання 10000 задач система повинна декілька днів виключно надсилати задачі, тому синхронний режим не є зручним та оптимальним [6].

Портал NetworkDynamics використовує методику розділення постановки задачі в грід від надсилання задачі в базу даних порталу. Окремо виконується регулярне завдання що проводить моніторинг, окремо завдання для запуску завдань. Через виконання операцій в різних процесах стає критичним навантаження на базу даних. При декількох тисячах задач система працює нестабільно, тому необхідні методи, що дозволять оптимізувати керування такою кількістю задач.

Ще однією особливістю NetworkDynamics є впровадження масивів задач (Jobset), які формуються з параметризованого опису. Фактично масив задач являє собою worklflow в якому між елементами немає зв'язків.

Параметричний опис містить математичні вирази $\{P_1,P_2,\ldots,P_N\}$ та $\{P_{start}:P_{step}:P_{end}\}$, які задають множину значень та діапазон значень з заданим кроком відповідно. Обробник перетворює такі параметризовані вирази на відповідну кількість окремих завдань, а по їх завершенню проводить агрегацію результатів в межах такого єдиного проекту.

Архітектура системи асинхронного керування завданнями на основі обробки подій

Виходячи з аналізу засобів створення віртуальних лабораторій та попереднього досвіду створення систем керування грід-завданнями в українській національній грід-інфраструктурі (MolDynGrid та NetworkDynamics) можна зробити висновок, що системи послідовної та синхронної обробки завдань не є в достатній мірі масштабованими. Для роботи з проектами, що виконують обрахунок десятків тисяч завдань необхідним є впровадження інших підходів до керування завданнями.

У роботі веб-серверів відомою є «проблема 10000 з'єднань» (англ. C10K problem) – проблема оптимізації програмного забезпечення для обробки запитів великої кількості клієнтів одночасно. Класичні веб-сервери (наприклад Apache), архітектура обробки запитів яких є синхронною не в змозі одночасно обслуговувати 10000 клієнтів. Така проблема була вирішена за допомогою використання асинхронних не блокуючих операцій. Такий веб-сервер як Nginx використовує концепцію асинхронної обробки подій, що представляють запити до веб-серверу. Такий підхід до асинхронного керування подіями може бути застосований і до керування великою кількістю грід-завдань, де кожна подія буде представляти процес постановки, моніторингу, чи обробки стану завдання.

Побудову архітекрутри віртуальних лабораторій доцільно розглядати в концепції «модель-представлення-контролер» (Model-View-Controller, MVC), як найбільш універсального та гнучкого підходу до розробки веб-інтерфесів. У такій архітектурі інтерфейс користувача (представлення) може бути сформованим відповідно до вимог кожної лабораторії, з тематичними елементами дизайну. Внутрішня логіка «моделі» та «контролеру» безпосередньо відповідає за процес обробки запитів користувачів. Саме архітектура «моделі» та «контролеру» реалізує алгоритми взаємодії з грід-середовищем та надає «представленню» достатню кількість інтерфейсів для побудови інтерфейсу для роботи. Інтерфейс до «контролеру» можна використовувати не тільки для побудови веб-інтерфейсу, а і для виконання віддаленого виклику процедур з іншого програмного забезпечення.

Об'єктом дослідження роботи є саме архітектура серверної частини (англ. backend), яка реалізує асинхронний підхід до керування завданнями в грід.

У центрі розробленої архітектури знаходиться черга запитів, що наповнюється подіями з різних джерел (рис. 1). Головним джерелом незалежних вихідних подій є інтерфейс користувача. За приймання таких запитів відповідає потік програми, який обробляє віддалені виклики процедур та відповідно до них додає події в чергу. Архітектура програмного забезпечення проміжного рівня gLite, ARC та UNICORE, що використовуються в світі зазвичай реалізує веб-сервіси, шляхом використання протоколу SOAP, тому найбільш доречним та сумісним з програмною базою грід-систем буде використання саме цього протоколу. Проте завжди можна додати ще один потік, який буде обробляти віддалені виклики процедур, що використовують інший протокол (наприклад, для прозорої міграції програмної бази віртуальної лабораторії MolDynGrid на асинхронну архітектуру керування завданнями).

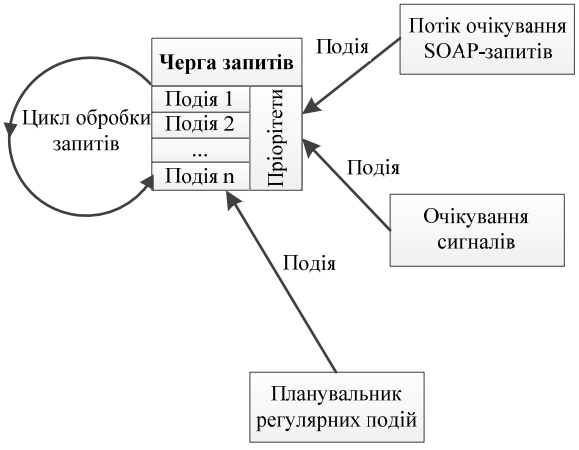

Рис. 1. Додавання подій до черги запитів

Окрім обробки запитів інтерфейсу користувача, ядро системи повинно містити обробники сигналів. За допомогою сигналів буде виконуватись ротація журналів роботи сервісу, перезапуск планувальників, тощо.

Планувальник регулярних подій ініціює надсилання до черги запитів регулярних подій, що працюють з багатьма об'єктами на регулярній основі. Такими подіями є опитування інформаційної системи грід-інфраструктури для синхронізації стану завдань кожного обчислювального елементу, планування надсилання завдань в грід-середовище (у відповідності до графу потоків завдань та пріоритетів виконання), збереження стану об'єктів до бази даних на жорсткому диску (серіалізація).

Події в черзі обробляються у відповідності до пріоритетів всередині циклу обробки запитів (рис. 2). Спочатку обробляються події, що мають найвищий пріоритет потім найменш пріоритетні. Події з найвищим пріоритетом включать планування інших подій та обробку ряду запитів інтерфейсу користувача. Такі події виконуються дуже швидко та потребують якнайшвидшої обробки. Виконання таких високо пріоритетних подій

доцільно виконувати у синхронному режимі, тобто в даній архітектурі (рис. 2) виконувати функцію-обробник події в потоці цикла обробки запитів та одразу повертати результат обробки події з черги.

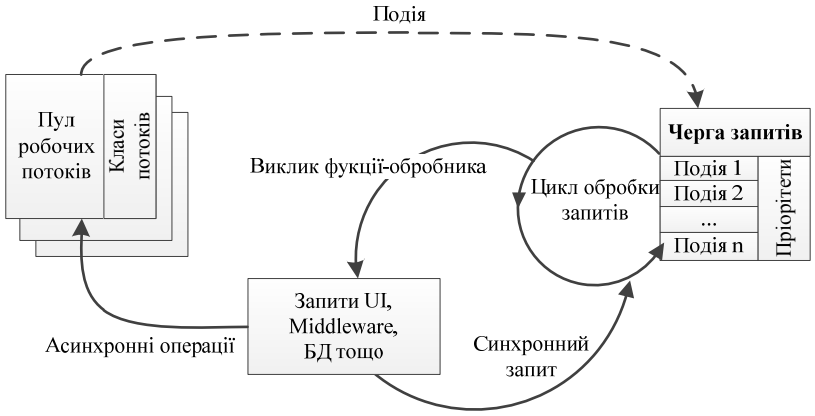

Рис. 2. Синхронна та асинхронна обробка подій з черги

До подій з найнижчим пріоритетом належать трудомісткі операції, такі як постановка задачі на виконання в грід чи опитування інформаційної системи про стан виконання завдання. Операції по роботі з файлами на елементах зберігання – ще один приклад завдань, що виконуються протягом тривалого проміжку часу. Саме такі операції є завадою для ефективного керування десятками тисяч завдань в системах синхронного запуску, через те, що блокують роботу систему на значний час обробки довготривалого запиту. Відповідно для вирішення цієї проблеми в запропонованій архітектурі використовується асинхронна обробка.

Базою асинхронної обробки є пул робочих потоків які обслуговують виконання функцій-обробників подій. У пулі доцільно виділити класи потоків, кожен з яких має певну резервацію під виконання операцій. Так існує клас потоків, що виконують постановку завдань, клас що виконують моніторинг тощо. Таким чином унеможливлюється ситуація коли при постановці на виконання десятків тисяч завдань система буде займатися виключно постановкою завдань через повну утилізацію всіх робочих потоків.

Робочий потік з пулу може запланувати нову подію (показано на рис. 2 пунктирною лінією), що надходить до черги запитів. На рис. 3 показано приклад послідовності обробки подій після надсилання запиту на обробку нового потоку завдань (Workflow, на рисунку WF).

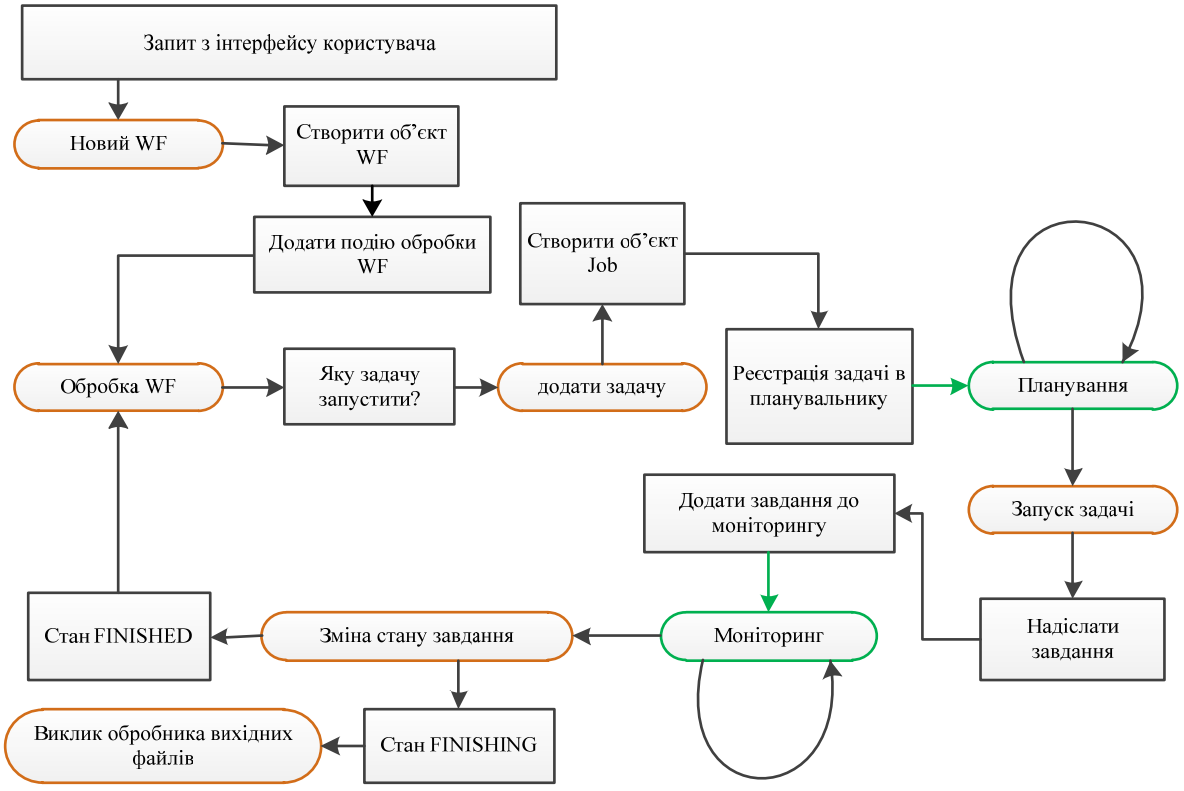

Рис. 3. Схема послідовності подій циклу додавання задачі на розрахунок

Закладання ідеології workflow в архітектуру на етапі проектування є необхідним кроком для досягнення гнучкості роботи сервісу. В простій реалізації потік завдань може складатися з одного завдання, масив завдань (jobset) також може бути представлено потоком завдань, що не мають зв'язків.

Після надходження запиту на виконання нового потоку завдань (рис. 3) до черги запитів додається подія «новий WF». На схемі заплановані події зображені овалами. Обробка події відбувається шляхом виклику функції-обробника (callback), схема роботи якої зображена прямокутниками.

Обробник події «новий WF» створює об'єкт WF та додає нову подію «обробка WF». Таким чином інтерфейс користувача не очікує на тривалий процес синхронної постановки задачі в грід. Обробник події «обробка WF» аналізує об'єкт WF та визначає ті задачі, для яких достатньо вхідних даних для проведення обрахунку. Можливо реалізувати параметризовані потоки завдань, які аналогічно до порталу NetworkDynamics будуть автоматично генерувати необхідну кількість завдань у відповідності до параметричного опису. Для кожного завдання WF, що може бути розрахованим, створюється подія «додати задачу».

Обробник події «додати задачу» створює об'єкт Job та реєструє задачу в планувальнику виконання завдань. Розроблена архітектура не диктує певних алгоритмів планування, проте завданням планувальника є визначення найбільш пріоритетних завдань виходячи з різних факторів (автор, час очікування, тощо). Кількість завдань що одночасно плануються до запуску не має перевищувати кількість робочих потоків сервісу. Виконуючи регулярне планування, для обраних завдань до черги надсилається подія «запуск задачі».

Обробник події «запуск завдання» в робочому потоці виконує передачу завдання до грід-інфраструктури, та отримавши ідентифікатор завдання в грід додає задачу до моніторингу.

Для збільшення ефективності моніторингу ресурсів ARC доцільно використовувати агрегацію запитів до інформаційних систем у відповідності до обчислювального елемента на якому обраховується завдання. При роботі з інфраструктурами gLite використовується єдина інформаційна система Top-BDII, тому такої необхідності немає. Отже наявність декількох обробників моніторингу, використання яких конфігурується збільшить гнучкість системи.

При зміні стану завдання обробник моніторингу додає подію «змін стану завдання». В залежності від стану це може бути перезапуск (завдання в стані «втрачено»), перехід до обробки відповідного WF (стан «finished») чи виклик зовнішньої підключеної програми. Наприклад, за відсутності підтримки автоматичної передачі результатів завдання обчислювальним елементом на елемент зберігання для стану «finishing» доцільно викликати обробник, що виконає передачу за допомогою керування каналом третьою стороною (third-party transfer) протоколу gsiFTP.

Таким чином за допомогою роботи з подіями запропонована асинхронна архітектура дозволить ефективно обробляти велику кількість грід-завдань, уникаючи блокування роботи на довготривалих операціях з грід-інфраструктурою.

Реалізація прототипу системи асинхронного керування грід-завданнями

При проектуванні реалізації системи асинхронного керування грід-завданнями пешим кроком є визначення технологій, які будуть використовуватись при розробці.

Технології та засоби розробки. Для взаємодії з грід-інфраструктурами доцільно використовувати бібліотеки програмного забезпечення проміжного рівня грід, щоб не створювати весь стек протоколів. Існуючі рішення для побудови грід-інфраструктур надають наступні API для наступних мов програмування:

- gLite – об'єднує різні сервіси, що створені мовами C, C++ та Java і надають відповідні API. Доступні для операційної системи Linux, гарантується сумісність з операційними системами сімейства Scientific Linux 5. Робота виключно з сервісами gLite для яких існують відповідні бібліотеки;
- ARC1 – сервіси створені мовою C++, API мовами C++ та Python/Java через систему SWIG. Підтримка будь-яких дистрибутивів операційних систем Linux, UNIX та Windows (за допомогою mingw32). Єдина бібліотека libarcclient для роботи з сервісами ARC1, ARC0, gLite та UNICORE.
- UNICORE – сервіси створені мовою Java, API мовою Java. Незалежні від платформи компоненти мовою Java. Набір бібліотек для роботи з сервісами UNICORE.

Обрано бібліотеку libarcclient продукту ARC1 [9], як таку, що надає інтерфейси для роботи з грід-інфраструктурою під керуванням будь-якого програмного забезпечення проміжного рівня та працює на всіх розповсюджених програмних платформах. Використано мову програмування C++, що дозволяє порівняно з реалізацією на Java використовувати менше оперативної пам'яті, особливо при великій навантаженості задачами для якої і розробляється система.

Для реалізації циклу обробки запитів використано кросплатформенну бібліотеку libevent (рис. 4). Бібліотека пітримує ряд механізмів роботи з подіями, що підтримуються ядрами операційних систем Linux, BSD UNIX, тощо та надає API мовами C та C++. Виклик функцій-обробників відбувається за таймаутом, отриманням UNIX-сигналу та подіями, що пов'язані з файловими дескрипторами [10].

Для обробки запитів віддаленого виклику процедур за протоколом SOAP використано API бібліотеки gSOAP. Критичним для libevent виявилась робота з динамічним додаванням подій до черги з потоків програми відмінних від базового потоку циклу обробки (для роботи з потоками в прототипі використовується бібліотека

pthread). Для вирішення цієї проблеми використано обробники подій файлового дескриптору конвеєру, в який проводить запис потік SOAP-запитів (рис. 4).

Сервіс повинен працювати з різною інформацією, починаючи від авторизації та делегації користувачів, закінчуючи описами задач в грід-інфраструктурі. Технологія зберігання різних даних може значно відрізнятися за вимогами, тому доцільно закласти роботу з різними системами зберігання даних. Два найбільш необхідних підходи - зберігання інформації у вигляді файлів на диску (реалізовано стандартними POSIX-сумісними викликами), та збереження в реляційних базах даних (планується використання технології ODBC, що дозволяє не прив'язуватись до певної СУБД).

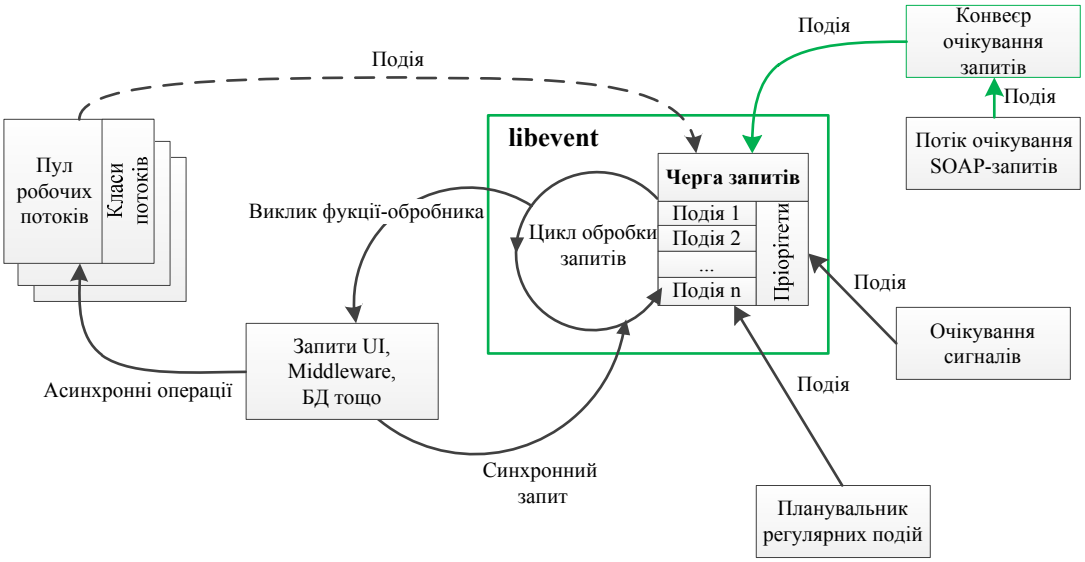

Рис. 4. Використання бібліотеки libevent для обслуговування черги запитів

Функції прототипу. Більшість функцій, що має виконувати сервіс керування завданнями може бути реалізована в залежності від інфраструктури та наявних сервісів кооперації. Так, наприклад, авторизацію на сервері можна виконувати використовуючи сертифікат користувача та протоколи TLS (MolDynGrid та NetworkDynamics), завантаженням проксі-сертифікату (LAP) чи використанню окремих імен користувачів/паролів (GridSphere), тощо. Реалізація взаємодії з грід інфраструктурою напряму залежить від програмного забезпечення проміжного рівня та наявної конфігурації інфраструктури.

Для забезпечення гнучкості системи, реалізацію таких функцій винесено в динамічні бібліотеки. Завдяки використанню об'єктно-орієнтованого підходу до створення програмного забезпечення, а саме інкапсуляції та наслідування інтерфейсів – такий підхід дозволяє відокремити логіку коду від реалізації окремих класів C++, а набір об'єктів, що буде використовуватись в кожному конкретному випадку винести в конфігурацію та підключати динамічно.

Адміністратор сервісу конфігурує необхідні методи авторизації, збираючи систему з окремих розроблених компонент, немов використовуючи конструктор. Така аналогія перекликається з робочою назвою системи, що розроблюється – MAGGIE (Main Architecture Gears for Grid Integrated Environment).

Українська національна грід-інфраструктура побудована за допомогою програмного забезпечення проміжного рівня Nordugrid ARC версій 0 та 1. Тому пріоритетною стала реалізація функцій роботи саме з ARC, що і виконанно в роботі. Перші впровадження системи в роботу плануються для проектів MolDynGrid та NetworkDynamics, що диктує в першу чергу необхідні методи для авторизації та делегації.

Підсумовуючи можна систематизувати створені реалізації класів прототипу MAGGIE:

- ідентифікація за DN сертифікату (перевірка сертифікату здійснюється засобами веб-серверу);
- авторизація з використанням серверу VOMS;
- отримання делегації з MyProxy серверу використовуючи властивість «authorized_retriever»;
- клас роботи з сервісами ARC;
- збереження даних в файлах;
- workflow, що складається з одного завдання;
- планувальник, що використовує принцип round-robin.

Подальше пріорітетне впровадження функцій напрямлено на реалізацію роботи з реляційними базами даних через ODBC та відлагодження коректної зупинки та старту сервісу після зупинки. Реалізація обробників параметризованих workflow та workflow, що містять зв'язки.

Базові класи MAGGIE. В центрі ієрархії класів системи асинхронного керування грід-завдання знаходяться класи DBObject, Logger, Cache та ACL. Більшість об'єктів є нащадками DBObject та реалізують методи serialize та deserialize для зберігання даних класів та їх відновлення з бази даних.

Статичний глобальний екземпляр класу Logger використовується для ведення журналу роботи сервісу та підтримує різні рівні виводу інформації – від критичної до відладки. Клас Logger реалізовано за допомогою ARC::Logger бібліотеки ARC.

У прототип закладено перевірку наявних прав на виконання операцій (клас ACL), проте наразі він дозволяє виконувати всі операції.

У першу чергу для даних ідентифікації, авторизації та делегації використовується кеш інформації за допомогою об'єктів класу Cache.

Відповідно для реалізації функціоналу розроблені абстракті класи (інтерфейси) та їх базові реалізації (дивись вище) для авторизаці (клас Auth), делегації (клас Delegation), роботи з грід-інфраструктурою (клас Middleware), завдання (клас Job), потоку завдань (клас Workflow), а також планування та моніторингу (класи Scheduler та Monitor відповідно).

Робота з грід-інфраструктурою реалізована в класі Infrastructure, що опирається на нащадків класів Middleware та конфігурацію системи (представлену в вигляді класу Config). Клас Infrastructure реалізує такі методи як getJobObject та getMonitorObject, що повертають об'єкти класів які відповідають саме цій грід-інфраструктурі.

Для відокремлення коду від підключених бібліотек використовується окремий простір імен «MAGGIE».

Висновки

Проведений аналіз алгоритмів роботи існуючих засобів створення віртуальних лабораторій показав ключові обмеження використання синхронного підходу до керування завданнями в грід-інфраструктурах, пов'язані з продуктивністю.

Запропонована архітектура системи керування завданнями в грід-інфраструктурах, що ґрунтується на асинхронній обробці подій, вирішує проблему роботи з десятками тисяч завдань з достатнім для практичного застосування рівнем гнучкості та масштабованості.

Для реалізації запропонованої системи доцільно використовувати існуючі API та бібліотеки libarcclient і libevent і мову програмування C++, що підтверджується успішною практичною реалізацією прототипу системи.

Прототип має базові функції необхідні для роботи з завданнями в українській національній грід-інфраструктурі і буде впроваджено в рамках віртуальних організацій MolDynGrid та NetworkDynamics.

Представлена архітектура може бути застосована для використання інших інфраструктур обчислювальних ресурсів, включаючи джерела даних, локальні обчислювальні кластери чи хмарні ресурси.

1. *Foster Ian, Kesselman Carl, Tuecke Steven.* The Anatomy of the Grid – Enabling Scalable Virtual Organizations // International Journal of Supercomputer Applications. – 2001. – Vol. 15. – P. 2001.
2. *Bote-lorenzo Miguel, Dimitriadis Yannis, Gomez-Sanchez Eduardo.* Grid Characteristics and Uses: A Grid Definition // Across Grids 2003, LNCS 2970. – 2003. – P. 291–298.
3. *Foster Ian.* The physiology of the grid: An open grid services architecture for distributed systems integration . – 2002.
4. *Borda Ann.* VRC Final Report / Report of the Working Group on Virtual Research Communities for the OSI e-Infrastructure Steering Group – March 31, 2006. – p.46
5. *Salnikov A.O., Sliusar I.A., Sudakov O.O.* Virtual Laboratory MolDynGrid as a Part of Scientific Infrastructure for Biomolecular Simulations // International J. of "Computing" – 2010. – Vol. 9. – N 4.
6. *Salnikov Andrii, Levchenko Roman, Sudakov Oleksandr.* Integrated Grid Environment for Massive Distributed Computing in Neuroscience. Proceedings Book of the 6-th IEEE International Conference IDAACS 2011, 15-17 September 2011, Prague, Czech Republic, p. 198–202.
7. *Novotny J., Russell M., Wehrens O.* GridSphere: A Portal Framework for Building Collaborations // 1st International Workshop on Middleware for Grid Computing (at ACM/IFIP/USENIX Middleware 2003), Rio de Janeiro, Brazil, June 2003.
8. *Lindemann J.* Lunarc Application Portal, http://laportal.sourceforge.net – 2007.
9. *Ellert M.* Advanced Resource Connector middleware for lightweight computational Grids // Future Gener. Comput. Syst. – 2007. – Vol. 23, N 1. – P. 219–240.
10. *Provos Niels.* Libevent – an event notification library – http://www.monkey.org/~provos/libevent/ – 2011.