ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, СИСТЕМНИЙ АНАЛІЗ ТА КЕРУВАННЯ

УДК 004.75

А.І. Петренко, Б.В. Булах

ЗАСТОСУВАННЯ WORKFLOW-СИСТЕМ ДЛЯ ПОТРЕБ СУЧАСНИХ НАУКИ ТА ІНЖЕНЕРІЇ

The paper considers the modern research direction in scientific and engineering software called the workflow systems. The objective of this paper is to evaluate the relevance of the workflow approach for solving pressing scientific and engineering problems. The comparative analysis of the number of successfully developed workflow projects is made, embracing a wide range of solutions from data analysis environments to grid portals. The advantages and shortcomings of these projects are evaluated and alternative architecture decisions are studied. We also consider the approaches to user interface development, the workflow components implementation, control and data flow organization, scenario analysis. Based on the analysis results, promising solutions using the open standards are proposed and the tendencies of further development of this direction are also covered.

Вступ

Еволюція сучасних наукових та інженерних досліджень нерозривно пов'язана з інтенсивним розвитком інформаційних технологій. Сьогодні проведення моделювань *in silico* замість реальних експериментів чи обробка експериментальних даних на високопродуктивних обчислювальних ресурсах сприймаються як належні. Розвиток мережі Інтернет та супутніх веб-технологій уможливили вільний і швидкий обмін актуальною інформацією не залежно від географічних відстаней. На допомогу науковцям та інженерам приходять нові технології ефективного використання гетерогенних розподілених обчислювальних ресурсів і сховищ даних, відомі як грід-технології. Постійне нарощення обчислювальних потужностей спричинило так званий "інформаційний бум", коли кількість накопичених експериментальних і розрахункових даних стрімко зростає і навіть перевищує можливості наявних файлових сховищ. Таким чином, ми є свідками становлення парадигми "електронної науки" (англ. e-Science) [1], яка позначає широкомасштабне залучення комп'ютерних обчислень у наукові дослідження, використання розподілених обчислювальних середовищ та інтелектуальних систем, організацію співпраці вчених різних установ і країн через "віртуальні лабораторії".

Однак грід і споріднені з ним клауд-технології є лише потужними механізмами залучення віддалених ресурсів для *виконання* обчислювальних кроків масштабних числових експериментів, і без додаткових автоматизованих засобів *проектування* таких експериментів їх використання залишається складною задачею для більшості дослідників. Як правило, складні експерименти вимагають узгодженого виконання цілих ланцюжків обчислювальних кроків, а тому актуальними є проблеми автоматизації компонування таких ланцюжків, контролю за ходом багатокрокових обчислень, передачі даних між цими кроками тощо. Виконання обчислень у грід вимагає від користувача обізнаності в конкретному програмному забезпеченні грід (далі – програмне забезпечення проміжного шару (ПЗПШ)), а тому важливою задачею є надання користувачу простого, доступного, інтуїтивно зрозумілого інтерфейсу, що приховав би специфічні особливості керування грід-задачами та дав можливість зосередитись на розв'язанні наукових і прикладних задач, а не задач адміністрування.

Вирішення названих вище проблем може базуватися на використанні так званих workflow-систем, ключовим об'єктом яких є абстракція "workflow" (найбільш поширені варіанти перекладу – потік робіт, робочий потік, маршрут, каскад або ланцюг) – обчислювальний сценарій, що складається з окремих обчислювальних кроків, поєднаних у певну послідовність виконання [2, 3].

Постановка задачі

Мета статті – оцінити придатність workflow-систем для розв'язання задач сучасної науки та інженерії, висвітлити переваги та недоліки отримуваних розв'язків, проаналізувати накопичений досвід побудови існуючих систем, вказати шляхи подальшого розвитку даного підходу.

Вихідні положення

Під "потоком робіт" в загальному випадку надалі будемо розуміти множину кроків, кожен з яких пов'язаний із виконанням певної дії (наприклад, обчислень), і переходів між ними,

що вишиковують виконання цих кроків у певну визначену послідовність. Визначений таким чином потік робіт може бути поданий орієнтованим графом, де вершини позначають окремі кроки потоку, а ребра – переходи між ними.

Використовуючи таку абстракцію для моделювання реальних послідовностей обчислень, слід зважати на певні особливості останніх: кроки обчислень виконуються над наборами даних із формуванням "конвеєрів" обробки даних; порядок виконання кроків і порядок передачі даних між кроками можуть не збігатися.

Виділяють [2] два типи переходів між кроками: керуючі переходи (позначають порядок виконання кроків потоку) та канали передачі даних (позначають маршрути передачі даних від кроків-постачальників даних до кроків-споживачів цих даних). Переходи першого типу утворюють *потік керування*, а переходи другого типу, відповідно, – *потік даних* (рис. 1).

Говорячи про обчислювальні потоки, слід зробити деякі уточнення. Як кроки цих потоків зазвичай виступають програмні компоненти, які приймають вхідні дані певного формату через набір вхідних портів, виконують обчислювальні дії та видають на вихід дані певного формату через набір вихідних портів. Цієї термінології дотримуватимемося й надалі.

Під системою керування робочими потоками (СКРП (англ. WFMS [3])) розуміється програмна система, яка відповідає за автоматизоване проектування та виконання потоків робіт, описаних певною вхідною мовою (текстовою або графічною).

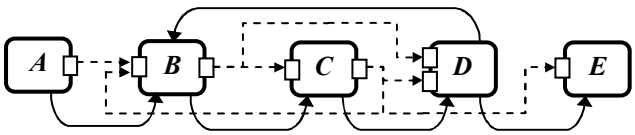

Рис. 1. Потік робіт: блоки *A*–*E* – кроки (компоненти) потоку, ⟶ – потік керування, - - - - - ▸ – потік даних

Приклади систем організації сценаріїв обчислень

Наукові або інженерні сценарії обчислень мають свою специфіку. По-перше, вони складаються на вимогу дослідників, які зазвичай не є фахівцями з програмування. Навіть мови програмування високого рівня є для них занадто складним засобом розроблення серйозних числових експериментів, і без допомоги програмістів дослідникам не обійтися. Відносно простіший засіб організації потоків робіт – скриптові мови. До них можна віднести стандартні засоби операційної системи (bat-файли для ОС родини MS Windows чи shell-скрипти для Unix-подібних ОС тощо). Гнучкості такого підходу для організації конвеєра обчислень цілком вистачає, проте недолік той самий: дослідник має володіти хоча б базовими навичками програміста й адміністратора. Друга характерна особливість масштабних наукових та інженерних обчислень – потреба у високопродуктивних обчислювальних ресурсах, що означає необхідність взаємодії зі спеціалізованим ПЗПШ. СКРП є прикладом інструментарію, який добре узгоджується з названою вище специфікою сучасних наукових та інженерних обчислень.

Як не дивно, багато СКРП, які згодом набули міждисциплінарного статусу, початково розроблялися для потреб конкретної спільноти науковців або інженерів [2]. Серед наукових застосувань, для яких активно розроблялися СКРП, провідні місця належать астрономії, комбінаторній хімії, біоінформатиці. Обробка даних фізичних експериментів на прискорювачі ВАК у ЦЕРНі, яка здійснюється за допомогою грід-технології, є також прикладом масштабного розгалуженого потоку робіт. Такі інженерні задачі, як обробка сигналів, моделювання електронних, механічних, гібридних систем і багато інших, також успішно розв'язуються за допомогою засобів workflow-систем. Простежити еволюцію та характерні особливості workflow-системи для наукових та інженерних задач доцільно на кількох прикладах.

Задачі у сфері астрономії перш за все пов'язані з необхідністю обробки численних зображень (знімків небесних об'єктів у широкому електромагнітному спектрі). Серед основних обчислювальних проблем – виконання перетворень різних за форматами, масштабами, спектральним діапазоном знімків і розпізнавання образів. Прикладом СКРП, яка покликана вирішити першу з цих проблем, є Montage [4, с. 19–38]. Призначений для однієї конкретної задачі, Montage пропонує своїм користувачам обмежений набір виконуваних модулів, які можна запускати окремо, або ж дає змогу скористатися функціональністю грід-порталу. Можливості користувачів довільно компонувати обчислювальні кроки з перетворення зображень можна вважати недостатніми, однак за умови використання грід-порталу доступу створення повноцінного потоку робіт відбувається

неявно на основі вхідних параметрів задачі (вхідний набір даних, параметри перетворення, такі як координатна система, проекція та ін.). Результуючий "абстрактний" сценарій у вигляді ациклічного орієнтованого графа (англ. DAG) може бути перетворений (за допомогою ПЗ Pegasus) на сценарій для планувальника Condor і запущений у грід-середовищі TeraGrid.

СКРП застосовуються й у більш "наближеній до Землі" сфері досліджень – метеорології. Задачею спільного проекту восьми дослідницьких установ США зі створення зв'язаних середовищ для атмосферних досліджень (LEAD) [4, с. 126–142] є зміна парадигми передбачення погоди: від статичного процесу за усталеним розкладом – до адаптивного сценарію, що керується змінами погодних умов. Загальний потік робіт можна коротко описати так: вхідні дані надходять до системи від численних датчиків, після чого асимілюються та передаються для розрахунку прогнозу погоди; результат передбачення візуалізується у вигляді зображень. Кожен із цих укрупнених кроків виконується з використанням одного або більше програмних компонентів. Задача проекту LEAD – додати адаптивності сценарію на кожному з кроків (варіювання кроку розрахункової сітки для окремих географічних областей, динамічна вибірка датчиків-постачальників даних, планування обчислювальних ресурсів, композиціювання потоків), що означає відхід від фіксованого потоку на основі скриптів і перехід до повноцінної СКРП, з якою може легко взаємодіяти кожен дослідник. Архітектура системи розрахована на обслуговування таких категорій користувачів: розробники моделей, які можуть змінювати вихідні коди програмних компонентів; науковці-експериментатори, які можуть компонувати потоки робіт і виконувати їх над різними наборами даних; студенти, які запускають стандартні потоки в навчальних цілях; відвідувачі без спеціальних знань, які зазвичай лише переглядають результати вже розрахованих прогнозів.

Архітектура системи є сервісно-орієнтованою та багатошаровою: робота організована через веб-портал доступу для користувачів, який спирається на веб-сервіси керування потоками, даними та інформаційні сервіси. Останні в свою чергу взаємодіють із грід-сервісами ПЗПШ Globus Toolkit і OGSA-DAI. Компоненти робочих потоків також "загорнуті" як веб-сервіси. Для візуального проектування потоків робіт використовується редактор X-Baya.

Для ілюстрації складності робочих потоків LEAD коротко розглянемо приклад типового потоку передбачення погоди [4, с. 134]. Під час фази попередньої підготовки завантажуються дані про рельєф місцевості та характеристики поверхні (два сервіси). Протягом фази аналізу ці дані разом із даними поточного прогнозу інтерполюються на тривимірну сітку (сервіс-інтерполятор) та направляються до сервісу аналізу даних, куди надходять також дані від датчиків, радарів і супутникові дані (постачаються ще трьома сервісами). Результати аналізу перевіряються на наявність сигнатур ураганів за допомогою засобів інтелектуального аналізу даних ADaM. Якщо у певному регіоні ураган виявлено, то запускається сценарій його передбачення. Під час виконання фази передбачення запускається сервіс-компонувальник, що будує конфігурацію для виконання ітерацій передбачення. Відбувається перетворення форматів даних між моделями для аналізу та передбачення. Паралельно запускаються до сотень екземплярів сервісу передбачення, результати якого подаються у вигляді тривимірних графіків на етапі візуалізації.

Сервісно-орієнтований підхід, як видно з попереднього прикладу, дав змогу розділити задачі створення спеціалізованих програмних компонентів, проектування сценаріїв та їх виконання. Одразу кілька профільних проектів для окремого розв'язання останніх двох задач ініціювали розроблення універсальних середовищ проектування та виконання робочих потоків: так, СКРП Triana [5] була початково розроблена в рамках проекту з виявлення гравітаційних хвиль GEO600, а СКРП Kepler [6] завдячує своїм виникненням проекту з моделювання паралельних систем Ptolemy II. Надалі ці СКРП набули статусу міждисциплінарних проектів.

Triana Workflows є розробкою Університету Кардіффа. На сьогодні до бібліотеки системи входить більше 500 компонентів для обробки сигналів, зображень, звуку і статистичного аналізу. Крім згаданого GEO600, серед проектів, що використовували Triana, – проекти з дослідження біорізноманіття BiodiversityWorld, медичне моделювання GEMMS (Grid-Enabled Medical Simulation Services), а також проекти з інтелектуального аналізу даних Data-Mining Grid і FAEHIM.

Архітектура Triana є типовою для "універсальних" СКРП: графічний редактор, бібліоте-

ка компонентів потоку, засоби керування виконанням. Компоненти в цій системі іменуються "функціональними одиницями" (units), порти яких з'єднуються між собою через "кабелі" (cables). Передача даних на вхідний порт компонента ініціюватиме його виконання. Цікавою особливістю є відсутність вбудованої підтримки конструкцій потоку керування: наприклад, цикли та розгалуження організуються за допомогою окремих керуючих "одиниць" [2, 5].

Розробку СКРП Kepler ініціювали представники одразу кількох наукових установ США (каліфорнійські університети Девіса, Санта-Барбари і Сан-Дієго, 2002 р.). Нині коло дисциплін, які підтримує система, надзвичайно широке: екологія, молекулярна біологія, генетика, фізика, хімія, океанографія, комп'ютерні науки та ін. Серед проектів, що використовують Kepler, – філогенетичні дослідження pPOD, аналіз даних від наукового обладнання в реальному часі REAP (Real-time Environment for Analytical Processing) та інші застосування з аналізу даних і моделювання [2, 6].

Taverna Workbench [7] є одним із головних елементів проекту розвитку е-науки myGrid, виконавцями якого є цілий ряд британських наукових закладів. За розвиток системи керування робочими потоками Taverna відповідає група з університету Манчестера. Первинною метою системи було задовольнити потреби біоінформатиків у засобі побудови робочих потоків з численних віддалених веб-сервісів.

Крім орієнтованості на веб-сервіси, до архітектурних особливостей також слід віднести можливість віддаленого контролю за виконанням потоків через сервер Taverna Server. Середовище розробки дає можливість контролювати синтаксичну коректність описів потоків, повторно використовувати потоки, ієрархічно включати потоки в інші потоки. СКРП також пропонує чимало засобів для діагностики, відлагодження та контролю за виконанням, серед яких: моніторинг, збереження історії виконання, призупинення виконання потоку, повторення запитів при відмовах окремих сервісів та ін.

Зазначені системи (та їх численні аналоги) відображають десятилітній досвід розроблення СКРП, будучи відповіддю на актуальні потреби дослідників у інструментах організації числових експериментів. Аналіз такого досвіду – необхідний крок на шляху до розроблення інтелектуальних СКРП нового покоління, що спиратимуться на сучасні досягнення в галузях веб-технологій і семантичних систем.

Архітектурні складові СКРП

Очевидно, що архітектура СКРП має проектуватися з міркувань підтримки продуктивної роботи користувача на всіх фазах життєвого циклу потоку робіт [8]. "Науковий" або "інженерний" робочий потік в процесі свого існування проходять кілька стадій (рис. 2). Робота починається із формулювання гіпотези або загальної ідеї числового експерименту, після чого дослідник переходить до *фази проектування*. Ефективність проектування робочих потоків напряму залежить від можливостей повторного використання готових напрацювань: повторного використання робочих потоків, їх фрагментів і компонентів, де це можливо; використання та наповнення бібліотек готових шаблонів,

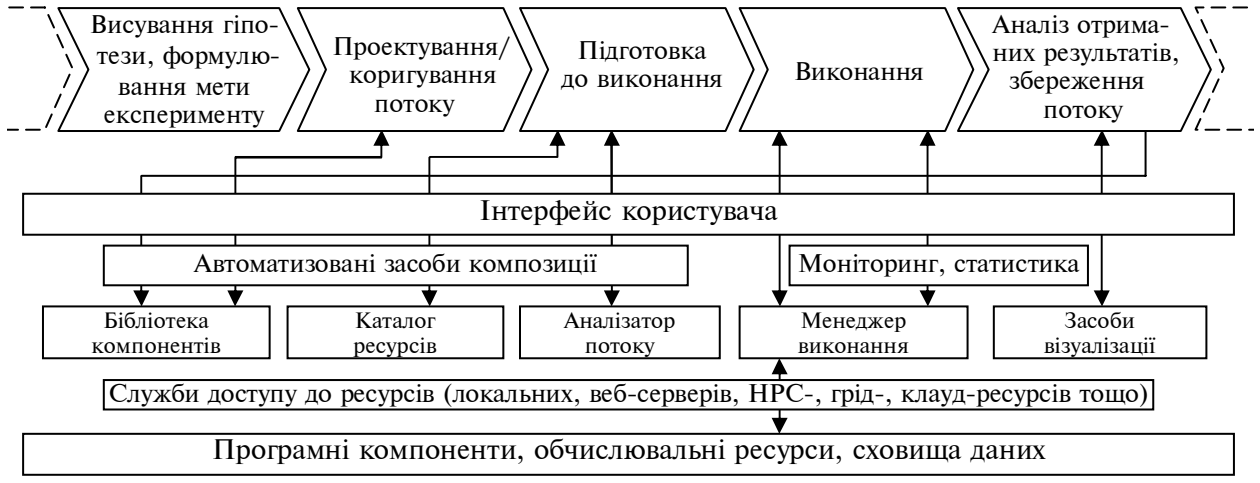

Рис. 2. Етапи життєвого циклу потоку робіт для наукових та інженерних застосувань і їх зв'язок з основними компонентами архітектури СКРП

компонентів і даних. Проектувальник визначає логічну структуру експерименту, вибирає програмні реалізації його кроків, зв'язує їх переходами, вирішує питання передачі даних. На виході користувач отримує абстрактний робочий потік – прототип майбутнього числового експерименту.

Для успішного виконання названих вище операцій СКРП має надавати розвинутий, бажано графічний інтерфейс користувача з можливістю доступу до репозитарію готових компонентів. На цій стадії подальша автоматизація кроку проектування можлива за рахунок залучення семантичних технологій, що дасть змогу автоматизувати пошук компонентів, узгодження форматів даних, композицію компонентів для досягнення поставленої мети [9].

Фаза проектування поступово переходить у *фазу підготовки* потоку, на якій розробник потоку готує вхідні параметри та завантажує дані для числового експерименту, вибирає (якщо є така можливість) ресурси (апаратні, програмні, джерела даних тощо) або вимоги до них (ЦП, пам'ять, ОС, паралельне виконання, виконання у грід і т.д.). Побудова конкретизованого сценарію можлива і без участі користувача, коли він делегує цю задачу інтелектуальним компонентам СКРП. Перед виконанням підготовлений (іншими словами – конкретизований) опис потоку бажано проаналізувати (автоматизованими засобами) на наявність вад: синтаксичних, структурних (неініціалізовані змінні, недопустимі переходи), логічних ("вічні" цикли, незадіяні відгалуження, тупикові гілки, взаємні блокування), та оцінити очікуваний час виконання і ймовірність збоїв (надійність).

На *етапі виконання* потоку готовий опис потоку передається до менеджера потоку, який відповідає за узгоджене виконання кроків потоку згідно з описом, передачу даних від постачальників до споживачів, моніторинг виконання та постачання користувача актуальною інформацією про хід виконання, ведення журналу та корисної статистики, повернення результатів роботи та статистичних даних користувачу.

Після отримання результатів дослідник має оцінити їх адекватність (із залученням процедур візуалізації результатів розрахунків) та ухвалити рішення про необхідність повторення експерименту або збереження потоку в бібліотеці готових компонентів, після чого життєвий цикл потоку замикається.

Аналіз результатів зазвичай відбувається за такими напрямами: адекватність (відповідність отриманих результатів експерименту очікуваним у рамках вхідної моделі), надійність (помилки, збої, виключні ситуації на різних кроках потоку, їх причини, усунення "вузьких місць" у потоці), ефективність (час виконання, шляхи його оптимізації). Варто також зазначити, що аналіз потоку на наявність вад, надійність та ефективність (як до, так і після експерименту) потребує формального опису структури потоку.

Тож зважаючи на особливості описаного сценарію роботи користувача, можна виділити такі елементи типової СКРП для наукових та інженерних застосувань: функціональний та інтуїтивний інтерфейс користувача, бібліотека програмних модулів, готових до використання як компоненти потоку, засоби автоматизованої композиції та аналізу потоків, менеджер виконання (аналоги цих підсистем наявні в більшості розглянутих вище прикладах СКРП). Доцільно детальніше розглянути специфіку кожної з цих підсистем.

Інтерфейс користувача

Багато існуючих СКРП пропонують своїм користувачам повноцінне інтегроване середовище розв'язання задач, що підтримує повний життєвий цикл потоку [2]. А графічний редактор потоку вже є стандартом де-факто для сучасних рішень (редактори названих вище Triana, Kepler, Taverna, XBaya є складником багатьох спеціалізованих workflow-проектів). Основними елементами редактора зазвичай є: робоча ділянка для розміщення елементів потоку, бібліотека доступних компонентів-реалізацій кроків потоку, вікно редагування властивостей вибраних елементів, меню з переліком операцій над потоком.

Зазвичай такі системи проектування (або принаймні їх клієнтська частина) є окремо інстальованими програмами та пропонують стандартний віконний інтерфейс. Утім більш гнучким і перспективним може виявитись поєднання веб-інтерфейсу, створеного засобами побудови RIA-додатків, із віддаленою серверною частиною. Це звільнить користувача від необхідності встановлення та оновлення версій програми всюди, де потрібно працювати з системою (як клієнт виступає веб-браузер із підтримкою Flash, Java чи подібних їм технологій), і дасть змогу розробникам системи опера-

тивніше вдосконалювати її та підтримувати в робочому стані (виключає потребу підтримки різних версій). Крім того, веб-сервер доступу може бути єдиною точкою входу до системи, що дасть можливість спростити облік користувачів та створення "віртуальних лабораторій" для колективної роботи.

З іншого боку, такий підхід вимагатиме наявності доступу до мережі Інтернет не лише під час виконання потоків, а й на інших фазах їх життєвого циклу, що не завжди є доцільним і зручним, особливо якщо робочий потік не потребує виконання масштабних обчислень на віддалених ресурсах і може виконуватися локально.

Окреме питання – графічна нотація, на яку орієнтується редактор потоків. Очевидно, що набір доступних примітивів має бути: достатнім для конструювання потоків зі складною конфігурацією (цикли, розгалуження), інтуїтивно зрозумілим користувачу навіть без спеціальної підготовки. На сьогодні не існує стандартних нотацій для опису потоків робіт наукового або інженерного спрямування, тож найчастіше такі редактори пропонують власну нотацію, яка базується на орієнтованому графі: кроки потоку позначаються геометричними примітивами – вершинами графу, а ребра-переходи – стрілками (рис. 3). Окремих позначень для елементів, які визначають потік керування, може не передбачатися взагалі (цю роль виконують спеціалізовані компоненти потоку) або вони базуються на стандартних нотаціях типу UML [10] (UML Activity Diagram), нотаціях опису бізнес-процесів (BPMN), спеціалізованих мовах типу YAWL [2] тощо.

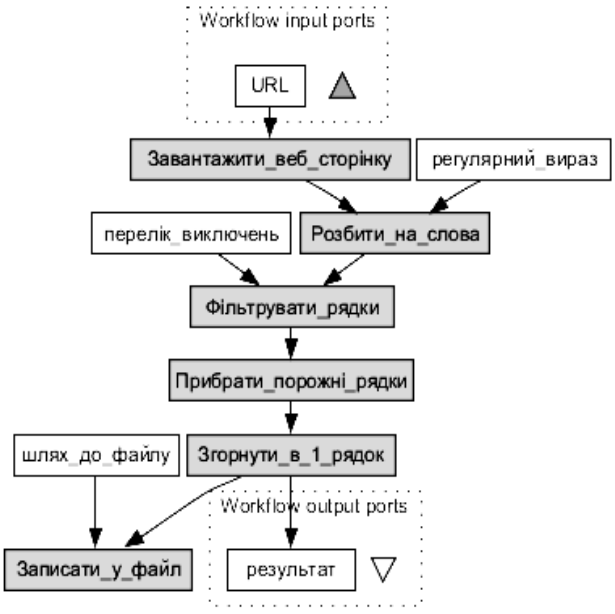

Рис. 3. Графічний опис фрагмента потоку робіт із текстової обробки веб-сторінок, розробленого в редакторі Taverna Workbench 2.2 [7]: сервіси потоку (прямокутні блоки) сполучені стрілками, що формують DAG-послідовність виконання; світліші блоки позначають константи, вхідні та вихідні дані

Компоненти потоків робіт

Типовий компонент робочого потоку призначений, як вже зазначалося, для виконання одного кроку потоку (тобто має конкретну функціональність) і взаємодіє з іншими компонентами через вхідні та вихідні порти. Порти, як правило, розраховані на передачу даних певного формату, що накладає обмеження на можливість довільного сполучення компонентів: поєднані потоком даних компоненти мають бути сумісними за своїми входами та виходами. В рамках моделей деяких СКРП компоненти агностичні до даних, тобто один і той самий компонент може підтримувати різні формати даних (Kepler) і їх різну розмірність (скаляри, масиви) (Taverna) [2, 8].

Існують два добре відомих підходи до композиції таких компонентів, як оркестровка і хореографія (з англ. orchestration та choreography). При першому підході всі операції з виклику компонентів потоку та передачі даних між ними здійснюються централізовано окремим "менеджером-диригентом". При другому підході самі компоненти мають піклуватися про виклик своїх "сусідів" у потоці та передачу їм даних, що означає децентралізований сценарій виконання.

Більшість сучасних СКРП використовують перший підхід, оскільки він надає більше контролю за виконанням та використовує незалежні один від одного (слабко зв'язані, а значить – більш універсальні) компоненти. Характерним представником такого підходу є система Kepler. Особливістю архітектури Kepler є те, що для кожного робочого потоку встановлюється своя модель обчислень із відповідним менеджером керування (термінами Kepler – director, тобто "директор" або "режисер"). Поняття "режисер" є ключовим для системи: якщо компоненти потоку (які тут називаються "акторами") та їх зв'язки становлять модель потоку, то режисери визначають модель обчислень. Актори діють лише за командою режисера, виконуючи операції над вхідними даними та повертаючи

результати через вихідні порти. Лише режисер, відповідно до своєї моделі обчислень, ухвалює рішення, коли і як саме викликати того чи іншого актора. Такий підхід може називатися поведінковим поліморфізмом: одні й ті ж самі актори з одними й тими самими залежностями в потоці можуть викликатися різними режисерами у різний спосіб. Наприклад, вони можуть викликатися прямо залежно від потоку даних (в одному послідовному потоці чи в кількох паралельних потоках для розподіленої обробки) або ж відповідно до поняття часу (для залежних від часу задач моделювання). Завдяки цьому така СКРП добре підходить як для класичних задач обробки даних, так і для задач моделювання динамічних систем.

Орієнтація на єдині стандарти надала б змогу залучати до СКРП компоненти сторонніх розробників (навіть розроблені для інших СКРП), що значно розширило б її функціональні можливості, а розробників компонентів звільнило б від необхідності підтримки різних форматів для різних систем керування робочими потоками. Прикладом такого стандарту для віддалених компонентів можуть слугувати вебсервіси, що добре себе зарекомендували як реалізація сервісно-орієнтованої архітектури (СОА) [11] у веб. Згідно з парадигмою СОА, веб-сервіси мають "контракт" (WSDL-опис) – стандартизований опис їх інтерфейсу, що розв'язує задачу автоматичного виклику сервісів та, частково, – автоматизації їх композиції. Існують також веб-стандарти на мови опису композиції веб-сервісів (такі як WS-BPEL [12]), що робить веб-сервіси цілком придатними для використання як компоненти робочих потоків для наукових та інженерних застосувань. Орієнтація на веб-сервіси полегшує залучення грід-обчислень для виконання потоку використанням грід-сервісів.

Так, розробники СКРП Taverna свідомо вибрали WS-інтерфейс як стандартний відкритий інтерфейс компонентів, не нав'язуючи власний внутрішній опис. Головним компонентом робочих потоків є саме веб-сервіс: SOAP/WSDL або REST чи спеціалізовані сервіси проектів BioMart, BioMoby, SoapLab та ін.

Прикладом універсалізму в підтримці різноманітних інтерфейсів компонентів є СКРП Triana. Крім базових Java-компонентів, як функціональні одиниці можуть виступати компоненти з такими інтерфейсами, як JXTA, P2P sockets, веб-сервіси, грід-сервіси. Ця різноманітність підтримується через проміжні інтерфейсні шари GAP і GAT (Grid Application Toolkit проекту Gridlab). Кожна функціональна одиниця може бути відредагована, перекомпільована та заново внесена до бібліотеки компонентів. Також цілий потік може бути згрупований як одна одиниця потоку – для підтримки ієрархічного "вкладання" потоків один в інший. Для реєстрації одиниць потоку використовується XML-опис.

Одним із перспективних напрямів майбутнього розвитку СКРП є задача *автоматичного компонування потоку*. Її суть полягає в тому, що користувач звільняється від необхідності здійснення ручного пошуку компонентів та їх сполучення і натомість має лише визначити вимоги до всього потоку: мету, вхідні та вихідні дані, обмеження на використані компоненти тощо. В такій постановці цю задачу можна розв'язати, залучаючи семантичний опис компонентів на базі розгорнутих онтологій і базу знань із правилами побудови потоків [9]. Слід зазначити, що використання метаданих вже зараз використовується у СКРП для розв'язання задач перевірки типів і автоматизованого пошуку компонентів.

Потоки керування і потоки даних

З урахуванням специфіки наукових та інженерних потоків більшість із наявних СКРП оперують в першу чергу потоками даних (згадати хоча б концепцію "кабелів даних" СКРП Triana). Тому потік керування часто є лише похідним від потоку даних. Причому при неявній побудові потоку керування за вказаними маршрутами даних можливі як прямий push-підхід, коли кроки виконуються в напрямку передачі даних від вихідного постачальника до його прямих та опосередкованих споживачів, так і зворотній pull-підхід, коли послідовність виконання кроків заздалегідь будується в зворотному порядку: від кінцевого споживача даних до всіх задіяних постачальників даних [2].

Один із піонерських workflow-проектів для наукових застосувань Discovery Net [2] мав характерну особливість: шар потоку керування явно відокремлювався від потоку даних (окремі фрагменти потоків даних вбудовувались як складові блоки потоку керування). Таким чином, потік керування координував виконання підпотоків даних, причому комунікації відсутні як між шарами потоку керування та потоку даних, так і між окремими потоками даних, поєднаних потоком керування.

Особливістю потоку даних Taverna є специфіка використання тих самих сервісів-акторів як для окремих екземплярів вхідних даних, так і для їх масивів. Нехай сервіс виконує деяку функцію f над вхідними даними x, тоді на виході матимемо $f(x)$. Якщо ж на етапі проектування потоку визначається, що на вхід буде подано масив даних, тобто редактор "знає", що $x = [x_1, ..., x_n]$, то результатом буде вже $[f(x_1),...,f(x_n)]$, що досягається неявним конструюванням цілого цик-лу викликів сервісу та збором їх результатів. Якщо ж на вхід функції подаються два масиви $[x_1,...,x_n]$ і $[y_1,...,y_n]$, то результатом буде або скалярний добуток виду $[f(x_1,y_1), f(x_2,y_2),...,f(x_n,y_n)]$, або (за вибором користувача) векторний добуток $[f(x_1,y_1),..., f(x_1,y_n),...,f(x_n,y_1),...,f(x_n,y_n)]$. Таким чином, компоненти потоку Taverna не потребують окремих операцій для скалярів і множин [2, 7].

Як у потоці даних, так і в потоці керування можна виділити деякі типові примітиви, так звані *шаблони робочих потоків* (аналогічно до шаблонів проектування у програмуванні), що також є предметом спроб стандартизації. Розроблення мови YAWL і супутнього інструментарію спиралося саме на такі дослідження. Ресурс [13] повністю присвячено аналізу простих і складних шаблонів та перевірці їх реалізації у СКРП.

Дослідники виділяють різні категорії шаблонів: шаблони потоку керування, потоку даних, шаблони ресурсів, обробки виключень. Шаблони також можна поділяти за їх складністю на прості (базові) та складні (похідні). Якщо базові шаблони зазвичай мають вбудовану підтримку в більшості систем керування потоками, то більш ускладнені варіанти переважно реалізовуються через базові, причому не завжди однаково у різних системах. Прикладами базових шаблонів потоку керування є: послідовність, паралельне розгалуження, синхронізація, виключний вибір, просте злиття. Серед більш складних шаблонів – множинний вибір, множинне злиття, критична секція, відкладений вибір, цикли, тригери та ін.

Повернемось до простого прикладу (див. рис. 1). Потік робіт є прототипом цілком реальних обчислювальних сценаріїв: нехай крок A позначає генератор (джерело) вхідних даних для аналізу, B – тимчасове сховище даних, C – процедуру аналізу даних, D – перевірку якості аналізу, E – візуалізатор результатів. Потік керування збігається з потоком даних між кроками A і B (передача вхідних даних до сховища), але на інших ділянках вони розбіжні: дані зі сховища A постачаються одразу на кілька наступних кроків (C, D), а результати аналізу C постачаються одразу до процедури оцінки D, візуалізатора E та знову до сховища B. Компонент D виконує роль вибору за умовою: він направляє потік керування або на повторний аналіз (замикаючи кроки B, C, D у цикл), або на відображення результатів E. Цікаво, що навіть такий простий приклад може бути нездійсненним у СКРП, які орієнтуються на DAG-модель потоку, що не підтримує цикли.

Аналіз потоків робіт

Задачами СКРП на етапі підготовки до виконання є перевірка сумісності опису потоку із прийнятою моделлю та аналіз потоку на можливі вади логічного характеру, що впливають на ефективність виконання або відмовостійкість. Якщо з першою з цих задач труднощів, як правило, не виникає, то реалізація інтелектуального аналізатора, здатного надавати слушні поради користувачу щодо довільного спроектованого потоку, є не настільки тривіальною. Зазвичай такий аналіз вимагає математичного опису моделі потоку [14]. Розглянемо кілька можливих підходів до формального опису потоків робіт.

Теорія автоматів. Цей підхід відштовхується від подання всіх компонентів потоку як кінцевих автоматів ($\sum$, Q, δ, q_0, F), де $\sum$ – вхідний алфавіт (переходи), Q – множина станів, δ – функція переходів, q_0 і F – початковий і кінцеві стани. Так, автомат для компонентів потоку, що обмінюються повідомленнями, може містити переходи "отримання повідомлення із вхідної черги" або "відправка вихідного повідомлення", а автомати компонентів, що враховують можливість збоїв, матимуть два кінцевих стани, що відповідають успішному та аварійному завершенню та ін. Поєднуючи окремі автомати, можна змоделювати весь потік робіт і дослідити його на наявність блокувань, перегонів, невизначеностей тощо, тобто перевірити (верифікувати) опис конкретного потоку на відповідність очікуваній поведінці [15].

Мережі Петрі. Добре розвинена теорія мереж Петрі є досить популярним засобом аналізу дискретних розподілених систем. Мережі Петрі можуть використовуватись для виявлен-

ня циклів і тупикових гілок в описі потоків робіт. При моделюванні потоку робіт позиції мережі Петрі відповідають крокам потоку, переходи мережі – переходам у потоці, послідовність виконання потоку задається дугами мережі Петрі. Тобто реальний робочий потік перед аналізом потребує перекладу на модель мережі Петрі та подальше дослідження цієї моделі. Можливий повний переклад елементів описової мови BPEL примітивами мереж Петрі [16]. Серед основних методів дослідження моделей, описаних мережею Петрі, – аналітичний, графічний, за допомогою еквівалентних перетворень, побудова дерева досяжності.

Числення процесів. Числення процесів (або алгебра процесів) – ще один альтернативний підхід для формального моделювання конкурентних систем. Існує чимало алгебр процесів (CSP, CSS, π-числення), але всім їм властиве використання для опису складних процесів невеликого набору примітивів та операцій над ними із визначеними алгебричними законами, що дає можливість здійснювати аналіз описів процесів і судити про їх еквівалентність. Для прикладу наведемо мінімалістичний опис процесу в π-численні у нотації Бекуса–Наура:

$$P ::= 0 \mid P|Q \mid !P \mid (vx)P \mid x(y).P \mid x\langle y\rangle.P,$$

де 0 – "нульовий" процес, виконання якого завершено; $P|Q$ – конкурентне виконання процесів P і Q; $!P$ – реплікація, процес, який завжди може створити нову копію; $(vx)P$ – створення нового імені; $x(y).P$ – процес, що очікує повідомлення (ім'я) y, яке було відіслане через канал зв'язку x; $x\langle y\rangle.P$ – позначає, що ім'я y передається через канал x перед тим, як продовжити процес як P.

π-числення (π-calculus) є одним із відносно нових і популярних формалізмів з родини алгебр процесів, що дає можливість опису конкурентних процесів, конфігурація яких може змінюватись під час роботи. Чимало уваги приділяється аналізу потоків робіт за допомогою π-числення [14, 17].

Виконання потоків робіт

Первинною задачею СКРП залишається автоматичне виконання сценаріїв обчислень, створених у самій системі або ж переданих їй у готовому вигляді. Серед вимог, які ставляться для СКРП наукових та інженерних обчислень, можна назвати такі:

- віддалене виконання, що не потребує постійної присутності користувача в системі (дає можливість переривати сеанс роботи, не перериваючи тривалих розрахунків). Означає необхідність серверної реалізації менеджера виконання;
- використання високопродуктивних ресурсів (у т.ч. в грід і клауд). Означає необхідність взаємодії із засобами різних ПЗПШ, при чому в такий спосіб, що не потребує обізнаності користувача у специфіці конкретного ПЗПШ;
- передбачені дії у випадку виняткових ситуацій під час виконання (програмних та апаратних збоїв, відсутності зв'язку із компонентами потоку тощо). Можливі статичні (окремі передбачені гілки виконання) та динамічні (автоматична адаптація потоку) сценарії реагування на виняткові ситуації;
- збереження журналу про хід обчислень та історії обробки даних для подальшого аналізу разом із самими результатами розрахунків.

Підходи до реалізації СКРП

Аналіз досвіду існуючих рішень у сфері workflow-систем дає можливість зробити деякі узагальнення (коротку порівняльну характеристику деяких найбільш популярних некомерційних СКРП наукового та інженерного спрямування наведено в табл. 1 і 2).

Більшість СКРП із підтримкою візуального проектування потоків робіт орієнтуються на власні редактори, інтегровані в програмні середовища, традиційно виконані як локальні програми для ПК. Як зазначалося, веб-інтерфейс може бути досить вдалим вирішенням для серверних реалізацій СКРП, і відсутність належної уваги до веб-технологій з боку розробників СКРП можна пояснити тривалою історією розробки та розвитку популярних СКРП, а також небажанням обмежувати користувачів необхідністю постійного доступу до мережі Інтернет. Розвиток технологій створення RIA-додатків (Flash, JavaFx, MS Silverlight) дає можливість сподіватися на появу розвинутих веб-середовищ керування робочими потоками нового покоління.

Окремі кроки потоків робіт у більшості СКРП реалізуються компонентами трьох типів: власними локальними програмними компонентами потоків (можуть бути "обгортками" для виклику віддаленого функціонала); віддаленими

***Таблиця* 1.** Порівняльна характеристика деяких некомерційних наукових та інженерних СКРП: архітектурні вирішення

Назва, поточна версія	Призначення, архітектурні особливості	Компоненти потоку, інтерфейси	Механізми залучення грід-обчислень
Triana 4.0 [5]	Інтегроване середовище для обробки даних, розроблене як Java-програма	Програмні модулі з інтерфейсами: JXTA, P2P-sockets, WS, WSRF	Через проміжні програмні шари GAP і Gridlab GAT
Kepler 2.1 [6]	Інтегроване середовище для аналізу даних і моделювання, розроблене як Java-програма	Java-модулі – "актори". Наявні актори-обгортки для виклику веб-сервісів	Через використання спеціалізованих "акторів" для взаємодії з ПЗПШ GT, Unicore, gLite
Taverna 2.2 [7]	Середовище проектування та виконання потоків веб-сервісів. Власний підхід до "оркестровки" сервісів (не BPEL)	Веб-сервіси і допоміжні локальні операції	Через використання WSRF-сервісів і плагінів для виклику сервісів Unicore, gLite, ARC
Askalon 2-beta [18]	Середовище розроблення, аналізу та виконання потоків грід-задач. Багатокомпонентна надбудова над ПЗПШ GT	Грід-задачі (зареєстровані в системі описи грід-задач)	Вбудована взаємодія планувальника та менеджера виконання із сервісами GT
P-Grade 2.10 [19]	Грід-портал: веб-портал на базі портлетів для організації грід-обчислень. Використовує Condor DAGman для виконання потоків	Грід-задачі (опис задачі становлять: виконуваний файл, аргументи, вхідні/вихідні порти-файли тощо)	Використання засобів ПЗПШ GT, ARC, gLite і спеціалізованих під них скриптів для виконання потоків грід-задач
Trident 1.2.1 [20]	Середовище наукових обчислень, базується на можливостях платформи Windows Workflow Foundation	WF-activities платформи .Net (стандартні та розроблені)	Прямо не підтримуються. Можливий запуск на кластері під ОС MS Windows HPC Server 2008

***Таблиця* 2.** Порівняльна характеристика деяких некомерційних наукових та інженерних СКРП: особливості проектування потоків робіт

Назва, поточна версія	Графічний редактор	Особливості потоку керування	Особливості потоку даних
Triana 4.0 [5]	Власний ("функціональні одиниці", сполучені "кабелями" даних)	Паралельне виконання, розгалуження та цикли організуються окремими компонентами потоку	Перевірка типів даних. Потік даних визначає потік керування
Kepler 2.1 [6]	Власний ("актори", сполучені переходами + "режисер" потоку)	Спосіб і час виклику "акторів" визначається "режисером" (різні моделі для обробки даних і моделювання)	Перевірка типів даних. Поліморфізм "акторів" відносно типів і розмірності (скаляри, масиви) даних
Taverna 2.2 [7]	Власний (DAG, що сполучає веб-сервіси та локальні операції)	Модель потоку – ациклічний граф, але цикли неявно організуються при виконанні за необхідності обробки масивів	Перевірка типів даних. Робота з масивами даних через циклічний виклик сервісів потоку
Askalon 2-beta [18]	Власний (редактор UML-діаграм) – локальна та браузер-версії	Підтримуються цикли, перехід за умовою, паралельні гілки виконання	Потік даних може визначатися окремими дугами графу, як це передбачено UML AD 2.0
P-Grade 2.10 [19]	Власний (граф залежностей грід-задач)	Простий ациклічний граф, що описує залежності між грід-задачами	Доступний режим одночасного запуску кількох екземплярів потоку для обробки різних наборів даних
Trident 1.2.1 [20]	Власний (ієрархічний графо-блочний опис "дій") – локальна та браузер-версії	Підтримуються паралельні гілки виконання, перехід за умовою	Перевірка типів даних на входах і виходах activity-компонентів

компонентами (наприклад, веб-сервіси); грід-задачами (в системах керування потоками грід-обчислень). Прикладами кожного з цих підходів є СКРП Kepler, Taverna і Askalon відповідно. Остання з названих систем є представником окремого підвиду СКРП – систем керування потоками грід-задач, які характеризуються виключною орієнтацією на роботу в грід-середовищі. Такі СКРП найчастіше є складовими т.зв. грід-порталів [21] – веб-порталів, що спеціалізуються на організації доступу до грід-ресурсів та виконанні грід-обчислень.

Інтерес до концепції workflow все більше зростає, що зумовило появу цілих програмних платформ засобів розроблення workflow-рішень. Прикладом такої платформи є Windows Workflow Foundation, що входить до складу останніх версій Microsoft .Net Framework. Незважаючи на орієнтацію на платформу MS Windows, дане рішення успішно застосоване у СКРП Trident, доступній під відкритою ліцензією Apache 2.0.

Зважаючи на заслужене зростання популярності сервісно-орієнтованої архітектури та розвиток концепції семантичної СОА, саме така архітектура на сьогодні вбачається найбільш актуальним рішенням для створення інтероперабельних розподілених систем. Тому системи, засновані на стандартах веб-сервісів і стандартних мовах їх композиції (напр., WS-BPEL) мають непогані перспективи для подальшого розвитку в інтелектуальних середовищах розв'язання задач, які базуватимуться на автоматичній композиції потоків робіт [9].

Висновки

У статті розглянуто окремі аспекти систем керування потоками робіт для наукових та інженерних обчислень як важливого компонента концепції "електронної науки". Проаналізовано існуючий програмний інструментарій, вказано перспективні архітектурні вирішення та шляхи подальшого вдосконалення даного виду програмного забезпечення, яке успішно довело свою практичну користь для дослідників у різних галузях науки та інженерії в рамках численних проектів.

Серед підходів до побудови сучасних систем керування обчислювальними сценаріями, які орієнтуються на нинішні стандарти веб-технологій, окремої уваги заслуговує ідея поєднання workflow-концепції для подання сценаріїв і стандартного інструментарію компонування веб-сервісів для забезпечення їх виконання. Орієнтація системи на існуючі стандарти, модульність її архітектури дадуть можливість розвивати її функціональність синхронно з подальшим розвитком веб-технологій і стандартів. Надалі планується вдосконалення запропонованої системи за допомогою залучення семантичних технологій для автоматичної композиції сценаріїв і впровадження розвинутих процедур аналізу спроектованих рішень.

1. *Згуровський М.З., Петренко А.І.* Grid-технології для е-науки і освіти // Наукові Вісті НТУУ "КПІ". – 2009. – № 2. – С. 10–17.
2. *Curcin V., Ghanem M.* Scientific workflow systems – can one size fit all? // Proc. of Biomedical Engineering Conference CIBEC 2008. – 2008. – P. 1–9.
3. *Yu J., Buyya R.* A Taxonomy of Workflow Management Systems for Grid Computing // J. of Grid Computing. – 2005. – **3**, N 3. – P. 171–200.
4. *Workflows* for e-Science. Scientific Workflows for Grids / I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (Eds.). – New York: Springer, 2007. – 530 p.
5. *Triana* – Open Source Problem Solving Software. – http://www.trianacode.org/
6. *Kepler* Project. – https://kepler-project.org/
7. *Taverna* Workflow Management System. – http://www.taverna.org.uk/
8. *Scientific* Workflows: Business as Usual? / B. Ludäscher, M. Weske, T.M. McPhillips, S. Bowers // Proc. of the 7th Int. Conference on Business Process Management BPM. – Ulm, Germany, 2009. – P. 31–47
9. *Петренко А.І., Булах Б.В., Хондар В.С.* Семантичний Грід для науки і освіти. – К.: Політехніка, 2010. – 184 с.
10. *Towards* an UML Based Graphical Representation of Grid Workflow Applications / S. Pllana, T. Fahringer, J. Testori et al. // Proc. of the 2nd European Across Grids Conference, Nicosia, Cyprus, January 2004. – Springer-Verlag, 2004. – P. 149–158.
11. *Erl T.* Service-Oriented Architecture: Concepts, Technology & Design. – New York: Prentice Hall / PearsonPTR, 2005. – 792 p.
12. *Web* Services Business Process Execution Language. – http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
13. *Workflow* Pattrens. – http://www.workflowpatterns.com/
14. *Oren E., Haller A.* Formal Frameworks for Workflow Modelling / Technical Report 2005-04-07. – Digital Enterprise Research Institute, 2005. – 21 p.

15. *Fu X., Bultan T., Su J.* Analysis of interacting BPEL Web Services // Proc. of the 13th Int. Conference on World Wide Web. – New York, ACM Press. – 2004. – P. 621–630.
16. *Formal* semantics and analysis of control flow in WS-BPEL / C. Ouyang, E. Verbeek, W. van der Aalst et al. // Science of Computer Programming. – 2007. – **67**, N 2-3. – P. 162–198.
17. *Weidlich M., Decker G., Weske M.* Efficient Analysis of BPEL 2.0 Processes using pi-Calculus // Proc. of the IEEE Asia-Pacific Services Computing Conference (APSCC'07). – Japan, 2007. – P. 266–274.
18. *Askalon* Programming Environment for Grid Computing. – http://www.dps.uibk.ac.at/projects/askalon/
19. *P-GRADE* Grid Portal. – http://portal.p-grade.hu/
20. *Project Trident*: A Scientific Workflow. – http://tridentworkflow. codeplex.com/
21. *Special* Issue: Portals for life sciences – providing intuitive access to bioinformatic tools / S. Gesing, J.V. Hemert, P. Kacsuk, O. Kohlbacher // Concurrency and Computation: Practice and Experience. – 2011. – N 23. – P. 223–234.

Рекомендована Радою
Навчально-наукового комплексу
“Інститут прикладного системного
аналізу” НТУУ “КПІ”

Надійшла до редакції
10 червня 2011 року