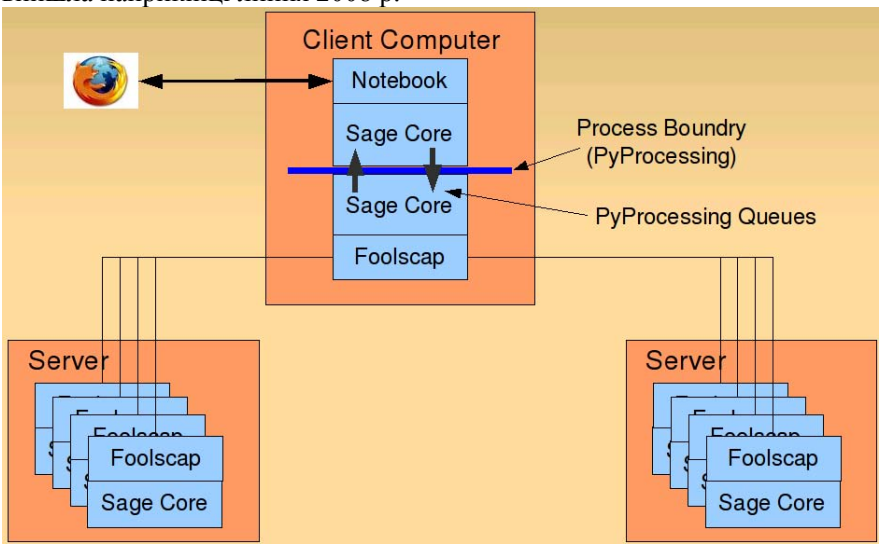
РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ У WEB-СКМ SAGE

О.П. Поліщук[α], С.В. Шокалюк[β], С.В. Середа
Україна, м. Кривий Ріг, Криворізький державний педагогічний університет
[α] apol@cabletv.dp.ua
[β] ksv_ipm@mail.ru

Традиційно розподілені обчислення реалізуються за клієнт-серверною технологією із застосуванням або низькорівневого інтерфейсу (pthread, OpenSSL), або високорівневих комунікаційних бібліотек (MPI, OpenMP). При цьому програмування самої обчислювальної задачі також вимагає застосування бібліотек для математичних обчислень з високою точністю (Boost, GMP). Частково автоматизувати процес створення розподілених додатків можуть метаоболонки, в яких необхідно реалізувати лише обчислювальний алгоритм у вигляді динамічної бібліотеки (див., наприклад, [1]).

Застосування систем комп'ютерної математики (особливо таких всеохоплюючих, як Web-СКМ SAGE) дозволяє зосередитись на обчислювальному алгоритмі замість деталей реалізації. Необхідна функціональність може бути створена внутрішньою мовою SAGE – Python, а наявність інтерфейсу Cython дозволяє перетворити інтерпретовану програму на компільовану динамічну бібліотеку. Звідси до розподіленої системи – один крок, який і був зроблений у новій версії системи, SAGE 3.0.6, що вийшла наприкінці липня 2008 р.



В новому пакеті DSAGE (Distributed SAGE) клієнтським комп'ютером виступає сервер SAGE, що отримує запит від Web-браузера за протоколом HTTP. Локальними серверами, що виконують за запитами клієнта окремі частини завдання, виступають інші системи, на яких має працювати ядро SAGE. Подібна організація використовується в MPICH, проте тут користувачеві не потрібно явно вказувати сервери; крім того, дана реалізація завдяки застосуванню інтерпретованої мови є незалежною від ОС та апаратної платформи (компіляція засобами Cython виконується на локальних серверах після отримання завдання).

Функціональність DSAGE реалізована в класі `DistributedSage`. Перед першим його застосуванням необхідно виконати конфігураційну утиліту `dsage.setup()`, що створює бази даних, приватний та публічний ключі для аутентифікації та SSL-сертифікат для сервера. Окремо конфігурування можна зробити за допомогою `dsage.setup_server()`, `dsage.setup_client()`, та `dsage.setup_worker()`.

Наступний крок – запуск:

```
sage: D=dsage.start_all(workers=20)
```

У прикладі запускаються сервер та 20 одночасно працюючих процесів. Також можна встановити номер порту (за замовчанням – 8081), рівень деталізації логу тощо. Окремий запуск сервера та робочої процедури виконується викликом `dsage.server()` та `dsage.worker()`.

Для виконання обчислень достатньо передати текст програми у `D`:

```
sage: def f(n): # створюємо функцію
        return n*n
# обчислюємо f(25)
sage: j = d.eval_function(f, ((25,),{}), job_name='square')
sage: j # отримуємо результат
625
sage: print j.wall_time # виводимо час виконання
0:00:00.144780
# виконуємо функцію для певного діапазону даних
sage: jobs = d.map(f, [10..20])
sage: jobs # отримуємо результат
[100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400]
```

Завершають роботу виклики `kill_server` та `kill_worker` (або `kill_all`).

Література:

1. Чумак Д.О., Семеріков С.О. Розробка програмного комплексу для метакомп'ютерних обчислень // Проблеми підготовки та перепідготовки фахівців у сфері інформаційних технологій. Матеріали V Міжнародної науково-технічної конференції "Комп'ютерні технології в будівництві": Київ–Севастополь, 18-21 вересня 2007 р. – Кривий Ріг, 2008. – С. 102–103.