Список использованных источников:

1. Берг О.Я. Физические основы теории прочности бетона и железобетона / Берг О.Я. – М.: Госстройиздат, 1962. – 96 с.

2. Прочность и деформативность бетона средних классов по результатам испытаний / С.Д. Семенюк, И.С. Фролков, М.Г. Мамочкина, Г.А. Дивакова // Вестник Белорусско-Российского университета. – Могилев, 2013. – Вып. № 3 – С. 92–100.

3. Ватуля Г.Л. Определение механических характеристик конструкций с помощью глубинных датчиков. / Ватуля Г.Л., Галагуря Е.И., асп. Петренко Д.Г. // Збірник наукових праць УкрДАЗТ. – 2013 – Вип. 138.– С. 231-235.

Шліхтенко Н.М., Ярмак Д.О.

студенти,
Національний технічний університет України
«Київський політехнічний інститут»

ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ ЗАСОБАМИ JAVASCRIPT

Сучасні задачі моделювання та обробки даних, що супроводжують інженерну та наукову діяльність, відбирають дуже багато часу і потребують великого обсягу комп'ютерної пам'яті для їх розв’язання. Але з виникненням технологій, що дозволяють створювати відносно недорогі та досить потужні процесори, почався перехід на багатопроцесорні системи, що значно спрощує вирішення таких задач.

Традиційно програми розробляються для послідовних обчислень. Для розв'язку певної задачі складається алгоритм, який реалізовується у вигляді послідовності інструкцій. Ці інструкції виконуються процесором одного комп'ютера. В кожен момент часу може виконуватись тільки одна інструкція, і лише після завершення її виконання починається виконання наступної. У паралельному ж програмуванні одночасно використовують кілька обчислювальних елементів для розв'язання однієї задачі. Це стає можливим завдяки розбиттю задачі на підзадачі, кожна з яких може бути вирішена незалежно [1].

Таким чином, паралельні обчислення – це такий спосіб організації комп'ютерних обчислень, при якому програми розробляються як набір взаємодіючих обчислювальних процесів, що працюють паралельно (одночасно) [2]. Основна складність при проектуванні паралельних програм – забезпечити правильну послідовність взаємодій між різними обчислювальними процесами, а також координацію ресурсів, що розподіляються між процесами.

Розвиток сучасних мікропроцесорів дозволяє виділити три базові напрямки з точки зору паралельних обчислень. По-перше, це багатоядерність (англ. multicore), коли декілька ядер знаходяться у корпусі одного процесора. По-друге, це технології з явним паралелізмом команд (англ. Explicitly Parallel Instruction Computing – EPIC). Останній напрямок – це багатопотоковість (англ.

Thread Level Parallelism – TLP), коли в кожному ядрі процесора виконується одночасно декілька потоків, конкуруючих між собою [3].

Паралельне програмування стає домінуючою парадигмою комп'ютерній архітектурі в основному саме у формі багатоядерних процесорів. І хоч використання можливостей паралелізму стало уже звичайною практикою в програмуванні, є мови програмування, в яких паралелізм застосовується не так активно, як у інших. До таких мов можна віднести JavaScript – найпопулярнішу мову веб-програмувння. Проте, враховуючи неймовірно стрімкий розвиток веб-технологій, саме у цієї мови дуже великі перспективи у застосуванні паралельних обчислень.

Існує декілька аспектів, що не дозволяють переносити деякі JavaScript програми на сторону клієнта (наприклад, для того, щоб зменшити навантаження на сервери). До них відносяться сумісність браузерів, статична типізація, доступність та ефективність. Проте, оскільки розробники браузерів постійно підвищують швидкість функціонування платформ JavaScript, ці проблеми поступово зникають.

Однією із проблем, пов'язаних з JavaScript, залишається специфіка самої мови. JavaScript – це однопоточне середовище, в якому кілька скриптів не можуть виконуватися одночасно. Наприклад, нехай є сайт, на якому необхідно управляти подіями інтерфейсу користувача, надсилати запити і обробляти великі обсяги даних API і працювати з моделлю DOM. На жаль, всі ці дії не можуть виконуватися одночасно через обмеження браузерів у середовищі виконання JavaScript.Тому скрипти виконуються в межах одного потоку [4].

Розробники можуть імітувати паралельне виконання коду за допомогою таких засобів, як методи setTimeout() і setInterval(), обробників подій, а також технології XMLHttpRequest. Всі ці функції виконуються асинхронно, але відсутність блокування не обов'язково означає паралельне виконання. Асинхронні події обробляються після повернення з поточного скрипта, що виконується. Але віднедавна в HTML5 з’явився більш зручний спосіб реалізації паралельного виконання коду – технологія Web Worker.

Специфікація об'єктів Web Worker визначає API для створення фонових скриптів у веб-додатках. Ці об'єкти дозволяють запускати у «фоновому режимі» довготривалі скрипти для виконання завдань, що вимагають великого обсягу обчислень, не блокуючи інтерфейс користувача або інших скриптів, що відповідають за взаємодією користувача з системою.

Об'єкти Web Worker запускаються в окремому потоці і використовують потокову передачу повідомлень для реалізації паралельності. Вони ідеально підходять для оновлення інтерфейсу, забезпечення його ефективності та реалізації оперативної взаємодії з користувачами. Саме ця особливість дозволяє використовувати даний підхід для реалізації паралельних алгоритмів на JavaScript.

Хоч і кожний скрипт кожного Web Worker виконується в окремому потоці операційної системи, точки зв'язку з іншими потоками ретельно контрольовані. У всіх об'єктів Web Worker немає доступу до непотокобезпечних компонентів і дані між потоками передаються за допомогою надсилання повідомлень з

даними в серіалізованому вигляді. Все це дозволяє уникати більшості проблем паралельного програмування при роботі з мовою JavaScript.

Повідомлення, що передаються між головною сторінкою та об'єктами Web Worker, копіюються, але загальний доступ до них не надається. В результаті кожного разу при передачі даних створюються їхні дублікати. Більшість браузерів реалізують цю функцію шляхом автоматичного кодування і декодування значень в кінцевих точках маршруту. Така поведінка зменшує ефективність паралельних алгоритмів, які потребують частої передачі великих об’ємів даних між об'єктами Web Worker [5].

Для збільшення ефективності при надсиланні великих структур даних в контекст Web Worker і назад у головний потік було створено додатковий спосіб надсилання повідомлень при використанні об’єктів, що надсилаються. При надсиланні цих об’єктів з одного контексту в інший не відбувається жодного копіювання даних, що забезпечує значне збільшення швидкості обміну великими повідомленнями. Але для уникнення проблем з одночасним доступом до спільної пам’яті контекст, з якого було надіслано об’єкт, втрачає доступ до нього, оскільки право доступу передається іншому контексту. До таких обєктів належить ArrayBuffer, що надає можливість надсилати всі види типізованих масивів між об’єктами Web Worker. Типізовані масиви наразі є зручним типами даних при реалізації алгоритмів над великими об’ємами даних [6].

Веб-технології розвиваються зі стрімкою швидкістю, і на сьогодні важко уявити своє життя без Інтернету. Багато задач, що раніше потребували складних обчислень і, відповідно, спеціальних програм та комп’ютерних пристроїв, сьогодні стали доступними звичайним користувачам персональних комп’ютерів.

Використання можливостей паралелізму стало уже звичайною практикою в програмуванні, проте у світі веб-розробки ці можливості лише починають з’являтись. І саме у JavaScript – найрозповсюдженішої мови веб-програмування – є великі перспективи в цьому напрямку.

Список використаних джерел:

1. Гергель В.П., Лабутина А.А. Учебно-образовательный комплекс по методам параллельного программирования // Нижний Новгород, 2007. – 138 с.

2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления // СПб: БХВ-Петербург, 2002. – 608 с.

3. Интернет Университет Информационных Технологий, Параллельное программирование с использованием OpenMP. [Електронний ресурс] – Режим доступу http://www.intuit.ru/department/se/openmp/class/free/status/

4. Использование Web Workers. [Електронний ресурс] – Режим доступу https://developer.mozilla.org/ru/docs/DOM/Using_web_workers

5. Основные сведения об объектах Web Worker. [Електронний ресурс] – Режим доступу http://www.html5rocks.com/ru/tutorials/workers/basics/

6. Transferable Objects: Lightning Fast! [Електронний ресурс] – Режим доступу https://developers.google.com/web/updates/2011/12/Transferable-Objects-Lightning-Fast