ПАРАЛЕЛІЗМ В JAVASCRIPT

Черненко М. Ю., Корнієнко Б.Я.

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна

Про переваги паралельного виконання на сьогоднішній день вже можна детально не розповідати, оскільки ця тема стала досить очевидною. Тенденції такі, що навіть мобільні телефони оснащуються багатоядерними процесорами. Згідно ресурсу Steam, згідно ресурсу Steam, який збирає статистику по конфігурації комп'ютерів власних клієнтів, пропорція однопроцесорних, двопроцесорних і чотирипроцесорних персональних ПК змінилася з 14.1%, 56.9% і 27.24% (серпень 2020) до 5.1%, 48.21% і 43.54% (січень 2021). Є й інша тенденція, яка полягає в тому, що велика кількість додатків переходять в браузери (програми перегляду html-документів); це забезпечує деяку універсальність разом з неминучими обмеженнями.

У зв'язку з цим виникає питання про можливість ефективного використання сучасних персональних комп'ютерів за допомогою браузера і JavaScript зокрема. Існує кілька способів виміряти продуктивність JavaScript, можна скористатися тестом від Mozilla(програми перегляду html-документів), котрий називається v8. Цей тест включає в себе багато частин, які покривають потенційні способи використання JavaScript для великої кількості обчислень, але якщо спробувати перевірити ступінь паралелізму цих обчислень, з'ясується, що всі вони виконуються тільки на одному процесорі / ядрі. Проте JavaScript містить модель конкурентного виконання. Спробуємо виконати наступний багатопоточний додаток:

Created by FolderMill | www.foldermill.com

```
function startThread(num) {
   var l=0, dateObj=new Date(),
   startTime=dateObj.getTime();
   if (num < 1) return;
   console.log("Thread " + num + " started");
   setTimeout('startThread('+(num-1)+');', 100);
   while (dateObj.getTime()-startTime<3000) {
   dateObj=new Date();
   l+=Math.random();
   }
   console.log("Thread " + num + " ended (l=" + l + ")");
  }
startThread(10);
```

Рис.1. Багатопоточний додаток

Ця програма створює 10 обчислювальних потоків з інтервалом в 0.1 секунди, кожен з яких працює протягом трьох секунд. Результати дивують, оскільки потоки виконуються повністю послідовно для всіх популярних на даний момент браузерів. У випадку з IE буде потрібно замінити console.log на document.write або alert, але виконання залишається послідовним.

Таким чином, незважаючи на теоретичну можливість, поточні реалізації мови не підтримують виконання декількох обчислювальних процесів одночасно. У випадках, коли це потрібно, можна розбивати завдання на дрібні частини і управляти запуском дрібних частин [1]. В цьому випадку обчислювальні ресурси будуть використані ще менш ефективно через накладних витрат і простою, але поведінка інших елементів сторінки і програми-браузера буде більш адекватним.

Проте можна очікувати, що рано чи пізно така можливість буде реалізована. Розглянемо способи конкурентного виконання обчислювальних потоків в рамках JavaScript.

JQUERY DEFERREDS

Досить популярна бібліотека jQuery дозволяє створювати об'єкти, які допомагають організовувати обчислення по готовності даних. Як правило, в JavaScript готовність даних означає, що закінчився процес їх отримання з

Created by FolderMill | www.foldermill.com

сервера. Концепція не є складною: створюється об'єкт, в який передаються функції, виконувані в разі різних статусів при вирішенні (в разі успіху, неуспіху, в будь-якому випадку). Як правило, такий об'єкт повертається функцією, яка виконує завантаження даних, об'єкт дозволяється за допомогою замикання, контексту реального обробника подій на контекст функції, яка повернула deferred-об'єкт. Функції асинхронних запитів JQuery завжди повертають deferred-об'екти.

```
$.get("test.php").done(function() {
   alert("$.get succeeded"); });
```

Рис.2. AJAX запит типу get

У наведеному прикладі функція, що виконує AJAX запит типу get, повертає deferred-об'єкт, який дозволяє призначити функцію, виконувану в разі успішності запиту. Також можуть бути реалізовані конвеєри з методів JavaScript, що виконуються асинхронно:

```
var defer = $.Deferred(),
 filtered = defer.pipe(function( value ) {
 return value * 2;
 });
defer.resolve( 5 );
filtered.done(function( value ) {
 alert( "Value is ( 2*5 = ) 10: " + value );
});
```

Рис.3. Конвеєр з методів JavaScript

CONCURRENT.THREAD

Ця бібліотека, яка поширюється на вільної ліцензії, дозволяє емулювати багатонитковою виконання програми, розбиваючи програми на невеликі фрагменти. Звичайно, це не може не позначитися на продуктивності.

По-перше, розповімо про саму організацію обчислень: для запуску паралельного процесу використовується функція:

```
Concurrent.Thread.create (function_var, function params ...)
```

Created by FolderMill | www.foldermill.com

Рис.2. Функція запуску паралельного процесу

що саме по собі досить зручно для тих, хто знайомий, наприклад, з потоками .NET.

Крім того, можна скористатися іншим способом і позначити фрагмент JavaScript-коду як багатопотоковою. Попередній приклад привести до такого виду не вийде, оскільки в ньому потрібно явне вказівку запуску різних потоків. Оформимо синхронний HTTP-запит як окремий обчислювальний потік [2]:

```
<script type="text/x-script.multithreaded-js">
  var req = Concurrent.Thread.Http.get(url, ["Accept",
 "*"]);
  if (req.status == 200) {
  alert(req.responseText);
  } else {
  alert(req.statusText);
  }
  </script>
```

Рис.3. Синхронний HTTP-запит як окремий обчислювальний потік

Вище ми згадали зниження продуктивності при використанні цього підходу. Проведемо кілька тестів для того, щоб з'ясувати, наскільки воно істотно. Як обчислювальної задачі візьмемо генерацію випадкових чисел. Не будемо наводити повний текст програми, оскільки це займе досить місця, використовувана завдання - генерація 107 псевдовипадкових чисел. Такий цикл займає приблизно 100 мс в випадку послідовного виконання (в процесі якого блокується робота браузера). У разі паралельного виконання за допомогою даної бібліотеки характерні часи виконання відрізняються на 2 порядки.

Можна відзначити, що час відрізняється досить сильно, що робить неможливим використання цієї бібліотеки в обчислювальних задачах, тому що втрати часу масштабу двох порядків абсолютно не виправдані. Проте, у цього способу є позитивний момент: незалежні обчислення не заважають

Created by FolderMill | www.foldermill.com

один одному. Це означає, що кореляції з кількістю потоків в даному тесті не було виявлено. Тестування вироблялося на Google Chrome 17.

З розглянутих способів організації паралельних процесів на JavaScript тільки Concurrent.Tread підходить для реалізації обчислень. Можливо, розробниками стандартів JavaScript буде обраний якийсь інший спосіб, але в зв'язку з розвитком браузерних додатків все складніше обходити увагою той факт, що процесори стали багатоядерними.

СПИСОК ЛІТЕРАТУРИ

1. Едвардс Дж. Багатопоточність у JavaScript. - 2008. - Доступно за адресою: http://www.sitepoint.com/multi-threading-javascript/
2. Макі Д., Івасакі Х. Багатопоточна структура JavaScript для асинхронної обробки: кандидатська дисертація [Інтернет-документ PDF]. –15 с. - Доступно за адресою: http://mirror.transact.net.au/sourceforge/j/project/js/jsthread/doc/thesis-en.pdf.
3. Корнієнко Б.Я. Дослідження імітаційного полігону захисту критичних інформаційних ресурсів методом IRISK. Моделювання та інформаційні технології. 2018. Вип. 83. С. 34-41.
4. Корнієнко Б.Я. Побудова та тестування імітаційного полігону захисту критичних інформаційних ресурсів. Наукоємні технології. 2017. № 4 (36). С. 316 - 322.
5. Корнієнко Б.Я., Максімов Ю.О., Марутовська Н.М. Прикладні програми управління інформаційними ризиками. Захист інформації. 2012. № 4 (57). С. 60 – 64. DOI: 10.18372/2410-7840.14.3493 (ukr).
6. Корнієнко Б.Я., Галата Л.П. Оптимізація системи захисту інформації корпоративної мережі. Математичне та комп'ютерне моделювання. Серія: Технічні науки, Випуск 19, 2019. - С. 56-62.
7. Korniyenko B., Galata L. Implementation of the information resources protection based on the CentOS operating system. Conference Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON -2019) July 2 – 6, 2019, Lviv, Ukraine. - pp. 1007-1011.

Created by FolderMill | www.foldermill.com

8. Галата Л.П., Корнієнко Б.Я., Заболотний В.В. Математична модель протидії загрозам у системі захисту критичних інформаційних ресурсів. Наукоємні технології, Том 43, № 3, 2019. – С. 300 – 306.
9. Korniyenko B., Galata L., Ladieva L. Research of Information Protection System of Corporate Network Based on GNS3. Conference Proceedings of 2019 IEEE Inernational Conference on Advanced Trends in Information Theory (IEEE ATIT -2019) Dezember 18 – 20, 2019, Kyiv, Ukraine. - pp. 244-248.
10. Korniyenko B., Galata L., Ladieva L. Mathematical model of threats resistance in the critical information resources protection system. CEUR Workshop Proceedings, Selected Papers of the XIX International Scientific and Practical Conference "Information Technologies and Security" (ITS 2019) Kyiv, Ukraine, November 28, 2019. Vol-2577. P.281-291.
11. Корниенко Б.Я. Кибернетическая безопасность – операционные системы и протоколы. ISBN 978-3-330-08397-4, LAMBERT Academic Publishing, Saarbrucken, Deutschland. − 2017. – 122 с.
12. Korniyenko B.Y., Galata L.P. Design and research of mathematical model for information security system in computer network. Наукоємні технології. – 2017, № 2 (34), С. 114 - 118.
13. Корниенко Б.Я. Информационная безопасность и технологии компьютерных сетей : монография. ISBN 978-3-330-02028-3, LAMBERT Academic Publishing, Saarbrucken, Deutschland. − 2016. – 102 с.
14. Korniyenko B., Galata L., Kozuberda O. Modeling of security and risk assessment in information and communication system. Sciences of Europe. – 2016. – V. 2. – No 2 (2). – P. 61 -63.
15. Korniyenko B. The classification of information technologies and control systems. International scientific journal. – 2016. –№ 2. – P. 78 - 81.
16. Корнієнко Б.Я. Інформаційні технології оптимального управління виробництвом мінеральних добрив :монографія. – К.: Вид-во Аграр Медіа Груп, 2014. – 288 с.
17. Korniyenko B., Galata L., Ladieva L. Security Estimation of the Simulation Polygon for the Protection of Critical Information Resources / B. Korniyenko, //CEUR Workshop Proceedings, Selected Papers of the XVIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2018) Kyiv, Ukraine, November 27, 2018, Vol-2318, - P. 176-187, urn:nbn:de:0074-2318-4

Created by FolderMill | www.foldermill.com

18. Korniyenko B.Y., Borzenkova S.V., Ladieva L.R. Research of three-phase mathematical model of dehydration and granulation process in the fluidized bed / B.Y. Korniyenko, S.V. Borzenkova, L.R. Ladieva // ARPN Journal of Engineering and Applied Sciences Volume 14, Issue 12, June 2019, Pages 2329-2332.
19. Zhulynskyi A.A., Ladieva L.R., Korniyenko B.Y. Parametric identification of the process of contact membrane distillation/ A.A. Zhulynskyi, L.R. Ladieva, B.Y. Korniyenko // ARPN Journal of Engineering and Applied Sciences Volume 14, Issue 17, September 2019, Pages 3108-3112.
20. Korniyenko B., Ladieva L. Mathematical Modeling Dynamics of the Process Dehydration and Granulation in the Fluidized Bed. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education III. ICCSEEA 2020. Advances in Intelligent Systems and Computing, vol 1247. Springer, Cham, pp. 18-30. https://doi.org/10.1007/978-3-030-55506-1_2
21. Korniyenko, B., Ladieva, L., Galata, L. Control system for the production of mineral fertilizers in a granulator with a fluidized bed. ATIT 2020 - Proceedings: 2020 2nd IEEE International Conference on Advanced Trends in Information Theory, 2020, pp. 307–310.
22. Ladieva, L., Kozanevych, Z., Klusta, T., Korniyenko, B. System of control of the process of alkylation of benzene with peripene in the liquid phase. ATIT 2020 - Proceedings: 2020 2nd IEEE International Conference on Advanced Trends in Information Theory, 2020, pp. 311–314.

Created by FolderMill | www.foldermill.com