2. Джереми Гибсон Бонд, Unity в C# геймдев от идеи до реализации / Джереми Гибсон Бонд //Питер. – 2019. – № 2. – С. 329–930.

УДК 004.042

ПАРАЛЛЕЛИЗМ В JAVASCRIPT

Мелихов В.А., студент,
Шнитко А.В., студент
Белорусский национальный технический университет
Минск, Республика Беларусь
Научный руководитель: канд.техн.наук, доцент Дробыш А.А.

Аннотация:
Рассматриваются проблемы использования параллелизма в JavaScript. Продемонстрированы различные способы применения параллелизма.

О выгодах параллельного исполнения процессов на сегодняшний день уже даже не нужно подробно рассказывать, поскольку эта тема многократно освещалась и оказалась достаточно очевидной. На сегодня тенденции таковы, что даже мобильные телефоны оснащаются многоядерными процессорами.

Статистика ресурса Steam демонстрирует явную тенденцию роста числа многопроцессорных систем. К примеру, только 3.1% пользователей владеют однопроцессорной конфигурацией (в 2014 году данное число составляло почти 20%).

Есть и другая тенденция: огромное число приложений переходят в браузеры. Это позволяет обеспечить некоторую универсальность приложения и простоту поддержки, однако, также и неизбежные ограничения.

В связи с этими тенденциями возникает вопрос о возможности эффективного использования ресурсов браузера и JavaScript, в частности – о параллельном исполнении процессов.

Существует исчерпывающее количество способов измерить производительность JavaScript. Эти тесты включают в себя много частей, которые покрывают потенциальные способы использования JavaScript для большого количества вычислений, но, если попытать-

ся проверить степень параллелизма этих вычислений, выяснится, что, вне зависимости от количества ядер процессора, все они исполняются только на одном процессоре/ядре системы.

Тем не менее JavaScript содержит модель конкурентного исполнения. При попытке выполнения многопоточного приложения в ходе тестирования можно заметить, что потоки выполняются полностью последовательно для всех популярных на данный момент браузеров. Таким образом, несмотря на теоретическую возможность, текущие реализации языка не поддерживают исполнение нескольких вычислительных процессов одновременно. Рассмотрим способы конкурентного выполнения в рамках JavaScript.

CONCURRENT.THREAD – это библиотека, распространяемая по свободной лицензии, позволяет эмулировать многопоточное исполнение программы, путем разбиения программы на отдельные модули. Конечно же такое разделение не может не сказаться на производительности. Мы провели несколько тестов для того, чтобы выяснить, насколько оно существенно. В качестве вычислительной задачи возьмём генерацию случайных чисел. Используемая задача – генерация 10^7 псевдослучайных чисел. Такой цикл занимает около 100 мс в случае последовательного выполнения. В случае параллельного исполнения при помощи рассматриваемой библиотеки характерные времена исполнения отличаются на 2 порядка. Таким образом можно отметить, что время выполнения задачи отличается крайне сильно, что делает немыслимым использование этой библиотеки в вычислительных задачах, поскольку потери времени совершенно не оправданы. Тем не менее, задача используемого ресурса выполняется: независимые вычисления не мешают друг другу. Это значит, что корреляции с количеством потоков в данном тесте не было обнаружено.

MULTITHREADING – другой подход [2], включающий в себя отдельный компилятор и средства для отладки многопоточных приложений на JavaScript. Курс ресурса в первую очередь направлен не на оптимизацию производительности кода, а на альтернативную модель описания параллельных процессов. Синхронизация в данном случае осуществляется посредством отправки сообщений, которые компилируются в обычный «ванильный» JavaScript-код при помощи Java-приложения. Суть в том, что код приложения разбивается на части между синхронизациями, а для синхронизаций гене-

рируется дополнительный код. Как уже упоминалось, для готовых программ создан отладчик, который позволяет проследить за синхронизацией. Такое решение отлично подходит для учебных целей, но не для решения реальных задач, поскольку является крайне ресурсозатратным и требует перекомпиляции при каждом изменении участка кода.

JQUERY DEFERREDS5 - популярная библиотека jQuery, которая решила вопрос многопоточности, путем создания объектов, которые помогают организовывать вычисления по готовности данных. Концепция довольно проста: создаётся объект, в который передаются функции, выполняемые в случае различных статусов при определенном условии. Как правило, такой объект возвращается функцией, которая загружает и данные, объект разрешается при помощи замыкания, контекста реального обработчика событий на контекст функции, вернувшей deferred-объект.

Из рассмотренных нами способов организации параллельных процессов на JavaScript только Concurrent.Tread подходит для реализации вычислений. Скорее всего, разработчиками стандартов будет выбран совершенно иной способ реализации параллелизма в JavaScript, но в связи с развитием браузерных приложений всё сложней обходить вниманием тот факт, что процессоры стали многоядерными.

Список используемых источников

1. Edwards J. Multi-threading in JavaScript. – 2008. – Available at: http://www.sitepoint.com/multi-threading-javascript/ (accessed on 19.03.2021)

2. Petitpierre C. Multithreading for Javascript. Available at: http://ltiwww.epfl.ch/sJavascript/(accessed on 19.03.2021).

3. Maki D., Iwasaki H. JavaScript Multithread Framework for Asynchronous Processing: PhD thesis [online pdf document]. – 15 p. – Available at: http://mirror.transact.net.au/sourceforge/j/project/js/jsthread/doc/thesis-en.pdf.