ПРОТОТИП СИСТЕМЫ ХРАНЕНИЯ СЕССИИ ВЕБ-БРАУЗЕРА НА ОСНОВЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Степанов Максим Александрович

студент, Новосибирский государственный университет, РФ, г. Новосибирск

Аннотация. В статье представлены результаты анализа существующих систем хранения данных сессии в браузерах с открытым исходным кодом - Mozilla Firefox и Chromium. В процессе анализа были найдены причины низкой производительности обеих систем и предложена технология хранения данных сессии, способная превзойти существующие подходы. В результате на основе браузера Chromium была реализована новая система хранения данных, которая превзошла старую по производительности в 2 раза. Таким образом была обоснована целесообразность использования новой системы в качестве основы хранения данных сессии в Chromium.

Ключевые слова: Chromium, Mozilla Firefox, Sqlite, сессии, реляционные базы данных, браузеры.

Введение

В современных браузерах востребованной среди пользователей функцией является восстановление сессии при запуске браузера после закрытия. Это позволяет продолжить работу с того же места, на котором пользователь остановился в предыдущем сеансе работы. В данном случае под сессиями будем понимать различные данные о состоянии браузера - открытые окна, вкладки и ссылки в них, а также последовательности переходов по ссылкам для каждой вкладки.

В современных браузерах для хранения сессий применяются ряд подходов. В них основным решением является сохранение информации о состоянии браузера в определенном формате в файле. Такой подход требует полной перезаписи файла даже при незначительных изменениях в состоянии браузера. Это при больших размерах сессии, может занимать длительное время.

Альтернативным способом для хранения данных сессии является реляционная база данных Sqlite, в которой будет перезаписываться только измененные части файла данных при сохранении.

Обзор подходов к хранению сессий

В настоящее время используются различные методы хранения данных сессии. Рассмотрим два таких способа, используемые в самых популярных на данный момент браузерах с открытым исходным кодом Chromium и Mozilla Firefox.

Хранение сессий в браузере Chromium

В браузере Chromium сессия хранится в виде последовательности команд. Каждая команда сессии содержит информацию о некотором атомарном изменении состояния браузера, например, о переходе по ссылке или о изменении положения окна браузера. Для сохранения

команд между запусками браузер последовательно записывает их в файл на диске, называемый файлом сессии. [1, с. 1]

Во время восстановления сессии браузер считывает команды из файла и последовательно применяет их к пустой сессии браузера без открытых окон.

Для того, чтобы сохранять изменения, которые происходят в процессе использования браузера (переходы по ссылкам, открытие новых вкладок и окон, закрытие старых), в конец файла сессии записываются новые команды, соответствующие этим изменениям. При длительной работе браузера изменения накапливаются, что приводит в увеличению размера файла.

Интенсивный рост размера файла сессии может привести к тому, что директория, содержащая пользовательские данные, будет занимать слишком много места на диске. Другой проблемой является то, что при старте необходимо считывать и интерпретировать файл сессии, что при больших его размерах может приводить к задержке на старте браузера. Для того, чтобы избежать этого, файл сессии перезаписывается каждый раз после 250 записанных команд с момента последней перезаписи.

В процессе перезаписи в новый файл сессии записываются только те команды, которые имеют непосредственное отношение к текущему состоянию браузера. Так, например, вместо нескольких последовательных команд изменения положения окна на экране будет записана только одна - та, которая была сохранена последней.

Основным недостатком такой системы является частая перезапись файла сессии, которая превращает изменение нескольких байт хранимой информации о сессии в запись на диск файла размером от килобайта, до нескольких мегабайт. Это может привести к увеличению времени отклика на компьютерах с малопроизводительными жесткими дисками или в случае высокой текущей нагрузки на диск от других приложений пользователя.

Хранение сессий в браузере Firefox

Система хранения сессии в браузере Firefox использует алгоритм, подобный описанному выше. Но, в отличие от Chromium, в файлах сессии хранятся не последовательности сериализованных команд, а JSON представление состояния браузера на момент записи файла [2, с. 1]. Помимо того, что для хранения одних и тех же данных в формате JSON, в общем случае, потребуется больше места на диске [6, с. 3], по сравнению с бинарным форматом данных, возникает также другая проблема, которая связана с внесением изменений в существующий JSON файл. Наиболее вероятно, что для сохранения изменений потребуется полностью перезаписать файл на диск. Если в процессе записи произойдёт сбой в работе приложения или операционной системы, то будет невозможно восстановить сессию, вследствие нарушения формата хранения данных либо же некорректности записанных данных.

Для того, чтобы не потерять состояние сессии при таком сбое, старые файлы сессии не удаляются, и при восстановлении сессии используется последний успешно записанный файл. Хотя такие файлы не будут хранить актуальную информацию, это позволит восстановить значительную часть открытых вкладок, если только они не были открыты непосредственно перед сбоем.

Недостатки системы хранения сессии браузера Firefox выражены ещё сильнее, чем в Chromium, так как запись большого файла происходит после каждого изменения состояния сессии. Для того, чтобы избежать слишком частой записи файла, Firefox по-умолчанию выполняет регулярное сохранение с интервалом в 15 секунд [3, с. 1], чтобы накопить изменения и не создавать постоянную нагрузку на диск компьютера.

Хранение сессий в виде БД

Для решения выявленных проблем обеих систем была исследована возможность создания системы хранения сессии для браузера Chromium, использующая библиотеку встраиваемой

базы данных Sqlite [5, с. 1], для хранения данных на диске.

Прототип разрабатывался на базе форка Chromium, расположенного на хостинге GitHub. Новая система полностью интегрирована в исходный код браузера Chromium. Получившийся форк поддерживает как использование старой, так и новой системы хранения сессии.

Для хранения информации о сессии используются 3 таблицы: “windows”, “tabs” и “navigations”.

Ниже представлено краткое описание основных отношений:

Таблица “windows” предназначена для хранения свойств, принадлежащих одному открытому окну браузера, которые будут восстановлены при запуске браузера

Таблица “tabs” содержит свойства, присущие одной открытой вкладке в браузере. Для каждой вкладки хранится некоторое количество переходов по ссылкам, каждый из которых хранится в виде записи в таблице “navigations”.

Данные об окнах, вкладках, группах вкладок и навигациях хранятся в виде таблиц в файле базы данных Sqlite, при изменении состояния браузера, актуализация хранимых данных происходит посредством запросов UPDATE, INSERT или DELETE к базе данных на языке SQL.

Рассмотрим в качестве примера следующую ситуацию: пользователь открыл 2 окна браузера, в каждом из которых открыто несколько вкладок. В момент создания каждому окну и вкладке присваиваются уникальные идентификаторы. Пусть теперь пользователь с помощью компьютерной мыши перетаскивает вкладку с идентификатором “5” из окна с идентификатором “4” в окно с идентификатором “1” на самую левую позицию. Для того, чтобы сохранить это изменение необходимо выполнить в базе данных сессии SQL запрос, представленный на рисунке 1.

```
UPDATE tabs
SET window_id=1, tab_visual_index=0
WHERE id=4
```

Рисунок 1. SQL запрос для обновления данных о положении вкладки в окне

Здесь window_id - это внешний ключ таблицы “tabs”, ссылающийся на запись в таблице “windows”, tab_visual_index - атрибут таблицы “tabs”, означающий индекс вкладки, id - первичный ключ таблицы “tabs”.

Этот запрос обновляет информацию об окне, в котором открыта вкладка, а также её индекс, то есть порядковое положение среди других вкладок в этом окне. Индексы вкладок начинаются с 0 и возрастают слева направо, поэтому для самой левой вкладки в окне в качестве значения столбца tab_visual_index должно быть установлено значение 0.

Пусть теперь пользователь решил развернуть окно с идентификатором “1” на весь экран при размере экрана 1920x1080. Для того, чтобы это действие было восстановлено после перезапуска браузера, необходимо выполнить SQL запрос, представленный на рисунке 2:

```
UPDATE windows
SET x=0, y=0, width=1920, height=1080, show_state=3
WHERE id=1
```

Рисунок 2. SQL запрос для обновления данных о положении окна на экране

Здесь, x, y, width, height и show_state - атрибуты таблицы “windows”, характеризующие положение окна на экране, а id - первичный ключ этой таблицы.

Значение “3” для столбца show_state означает, что пользователь нажал кнопку “Развернуть” в углу окна.

Для определения эффективности новой системы были произведены необходимые испытания.

Методика испытаний

В браузере Chromium для проведения большинства тестов используется тестовый фреймворк Google* Test [5, с. 1]. Также для Google* Test в репозитории Chromium уже реализовано большое количество вспомогательных классов и функций. Так как прототип системы был реализован на основе форка Chromium, то все эти вспомогательные классы и функции были доступны для использования. Поэтому для измерения производительности был также использован Google *Test.

В Google* Test тесты состоят из двух частей — тестового класса (test suite) и тестового метода (test case). Тестовый класс позволяет задавать произвольные предусловия для тестируемой ситуации. Сама же тестовая ситуация описывается внутри тестового метода в виде обычной функции, написанной на языке C++. Внутри этой функции также допускается использование открытых и защищенных методов тестового класса. Таким образом, тестовые классы Google * Test позволяют многократно применять часто используемые предусловия и управлять их созданием посредством наследования.

Наиболее популярным предусловием, используемым для тестирования Chromium, является запущенный браузер со всеми необходимыми для его работы процессами [8, с. 1]. Для этого в библиотеках браузера Chromium есть тестовый класс InProcessBrowserTest. С помощью этого класса перед началом непосредственно теста воспроизводится процесс обычного старта браузера, подобно случаю, когда пользователь запускает браузер через ярлык на рабочем столе.

После того, как начальная процедура запуска завершилась, исполняется тестовый метод, в котором происходят основные проверки теста. В нашем случае необходимо просимулировать взаимодействие пользователя с приложением и записать метрики, которые связаны с работой системы хранения данных сессии.

В качестве метрик выбраны:

- время выполнения задач хранилища сессии в фоновом потоке worker-thread;
- количество байт данных, записанных на диск;
- количество обращений на запись к диску;
- время записи данных на диск.

В качестве тестовых сценариев было ли выбраны следующие:

- перемещение вкладок внутри окна - в тесте симулируется изменение положения вкладки с открытым сайтом внутри браузера - раз в 100 миллисекунд самая правая открытая вкладка перемещается в самую левую позицию в окне;

- изменение размеров и положения окна на экране - раз в 10 миллисекунд положение окна на экране меняется на случайное новое.

Такой выбор обоснован тем, что эти на этих сценариях сильнее всего обнаруживаются недостатки системы хранения данных сессии в браузере Chromium.

Система хранения, основанная на файлах сессии, сохраняет изменения раз в 2,5 секунды, поэтому для того, чтобы воспроизвести проблему с большим количеством перестроений файла сессии были добавлены интервалы ожидания между действиями. В противном случае все изменения попали бы в одну и ту же перезапись файла. Следует заметить, что количество операций записи будет возрастать с увеличением промежутка между действиями пользователя, поэтому выбран небольшой интервал времени для того, чтобы замерить большее количество изменений состояния сессии.

Для получения метрики времени выполнения задач в систему добавлен код для записи измерений. Вычисление продолжительности выполнения той или иной задачи происходит с применением системных часов высокой точности.

Для измерения метрик работы с диском было использовано приложение Process Monitor для ОС Windows. Оно позволяет отслеживать все операции с диском, а также фильтровать и агрегировать их [7, с. 145].

Для того, чтобы замерить операции записи на диск фильтры были настроены так, чтобы сохранять только события записи в файлы сессий или в файлы базы данных сессии. После этого был выполнен запуск теста, в процессе выполнения кода которого были отслежены все интересующие нас операции записи.

Суммарные данные за всё время работы теста получены с помощью функции агрегации.

Все запуски теста производились на компьютере со следующими характеристиками:

- операционная система Windows 10;
- процессор Intel Core i7 13700k, 3400 МГц;
- оперативная память 32 GB DDR5, 4800 МГц;
- жесткий диск Samsung 980 Pro.

Результаты измерений

Результаты измерений, проведённые для сессии, состоящей из одного окна с 50 открытыми вкладками приведены в таблице ниже.

Таблица.

Результаты измерений производительности

Тип эксперимента	**Перемещение вкладок**		**Перемещение окна**	
Система хранения	старая	новая	старая	н
Время worker-thread	1693 мс	725 мс	135 мс	3
Объём записи	1,93 Мб	1,02 Мб	366,4 Кб	1
Количество операций записи	200247	3276	10953	
Суммарное время записи на диск	0,3755 с	0,022 с	0,0783 с	0

На основании результатов можно заключить, что новая система более эффективно использует диск персонального компьютера. Так, объём записи на диск уменьшился приблизительно в 2 раза в обоих экспериментах, а время, затраченное на непосредственную запись данных на диск уменьшилось в 7-20 раз.

Эти эффекты достигаются за счёт рационального использования базой данных Sqlite дискового хранилища путём записи только измененных блоков данных. Это приводит к уменьшению количества операций записи, за счёт отсутствия перезаписи данных, не претерпевших изменений. Дополнительно Sqlite хранит многие данные и запросы в кэше, тем самым снижая накладные расходы на использование базы данных.

Также время исполнения задач на фоновом потоке различается минимум в 2 раза. Это происходит из-за частой перезаписи данных на диск для подхода с файлами сессии. Можно заметить, что старая реализация Chromium записывает команды сессии отдельными операциями с файловой системой.

Заключение

В данной работе выполнен анализ реализации механизма сессий в двух самых популярных современных браузерах (Chromium и Mozilla Firefox) с точки зрения производительности. Обе системы неэффективно используют диск пользователя при сохранении изменений.

Был предложен новый подход на основе базы данных, устраняющий наиболее существенные проблемы из выявленных при анализе. Реляционная база данных Sqlite записывает на диск только модифицированные данные, что позволяет получить преимущество над другими реализациями.

Результаты сравнения скорости работы механизма сессий на основе базы данных позволяют ожидать ускорение операций обновления на 50% на некоторых сценариях использования браузера по сравнению с существующими реализациями.

Таким образом, в предлагаемой системе решена проблема, связанная с частой перезаписью данных на диск. В дальнейшем необходимо продолжить исследовать производительность на различных сценариях, в том числе производительность чтения данных.

В настоящее время идет работа над реализацией нового подхода в рамках форка Chromium на хостинге проектов с исходным кодом GitHub, расположенного по адресу https://github.com/MaximuSG140/chromium/pull/1.

Список литературы:

1. Chrome Session and Tabs Files (and the puzzle of the pickle). - [Электронный ресурс] - Режим доступа. -URL: https://digitalinvestigation.wordpress.com/2012/09/03/chrome-session-and-tabs-files-and-the-puzzle-of-the-pickle/ (Дата обращения 25.04.2024)

2. Analysing Firefox Session Restore data. - [Электронный ресурс] - Режим доступа. -URL: https://www.foxtonforensics.com/blog/post/analysing-firefox-session-restore-data-mozlz4-jsonlz4 (Дата обращения 25.04.2024)

3. Change Firefox session store interval to save your SSD. - [Электронный ресурс] - Режим доступа. -URL: https://www.mahal.org/2016/10/change-firefox-session-store-interval-save-ssd/ (Дата обращения 25.04.2024)

4. GoogleTest* Primer. - [Электронный ресурс] - Режим доступа. -URL: https://chromium.googlesource.com*/external/github.com/google*/googletest*/+/HEAD/docs/primer.md (Дата обращения 26.04.2024)

5. SQLite Home Page. - [Электронный ресурс] - Режим доступа. -URL: https://sqlite.org/index.html (Дата обращения 25.04.2024)

6. Maeda, Kazuaki. Performance evaluation of object serialization libraries in XML, JSON and binary formats.- 2012 - 10.1109/DICTAP.2012.6215346.

7. M. Russinovich, A. Margosis. Troubleshooting With The Windows Sysinternals Tools - [Электронный ресурс] - Режим доступа. -URL: https://archive.org/details/troubleshooting-with-the-windows-sysinternals-tools (Дата обращения 29.04.2024)

8. Browser Tests. - [Электронный ресурс] - Режим доступа. -URL: https://www.chromium.org/developers/testing/browser-tests/ (Дата обращения 02.04.2024)

*(По требованию Роскомнадзора информируем, что иностранное лицо, владеющее информационными ресурсами Google является нарушителем законодательства Российской Федерации – прим. ред.)