Подход к работе сайтов с данными в оперативной памяти и на жёстком диске

М.С. Герман[1], А.В. Ермаков[1], С.А. Ермаков[1]

1 ИПМ им.М.В.Келдыша РАН

Аннотация. Описан пример ускорения построения вэб-страницы за счёт распределённого хранения данных в оперативной памяти и на жёстком диске.

Ключевые слова: вэб-страница, CMS, динамическая часть

Approach to work of sites with data in RAM and on the hard disk

M.S. German, A.V. Ermakov, S.A. Ermakov

Keldysh Institute of Applied Mathematics RAS

Abstract. An example of accelerating the construction of a web page due to distributed data storage in RAM and on a hard disk is described.

Keywords: webpage, CMS, dynamic part

Активное развитие интернета в последние десятилетия привело к быстрому развитию программ, позволяющих упростить разработку и создание сайтов. На рынке стали появляться не только новые языки вэб-программирования, но и готовые наборы инструментов для быстрого создания сайта. Такое наборы инструментов называются Content management system (далее CMS). Назначение подобной системы – дать возможность разработчику (web-программисту) как можно проще и быстрее создать сайт, после чего останется только наполнить его контентом.

Рынок интернета развивается стремительно и, как следствие, потребности посетителей сайтов так же растут. Интернет-сайты в настоящее время выполняют не только функции предоставления информации, но дают возможность организации обратной связи, электронного документооборота, оплаты услуг, бронирования номеров гостиниц, покупки билетов и т.д. Сайты стали богаче не только по функциональным возможностям, но и по дизайну – появились адаптивные версии вёрстки под планшетные устройства, мобильные телефоны, широкоформатные мониторы. Повышение сложности сайтов

напрямую повлияло на развитие CMS – появилась необходимость разделения систем по направлениям, с различными наборами инструментов.

На практике, CMS удобна для небольших сайтов, которые не требуют активного развития и адаптации под рынок Интернета, поскольку основное назначение CMS – быстрое создание сайта, без учёта возможного его дальнейшего изменения. CMS поддерживают техническую документацию, чтобы дальнейшее изменение готовой CMS не было столь сложной задачей.

Для различных направлений сайтов (интернет-магазин, сайт-визитка и др.) требуется различный функционал. CMS обычно разделяют на обязательную неизменную часть для любых направлений сайтов (далее – ядро CMS) и необязательные части, которые могут отсутствовать или быть различными для направлений. Необязательные части могут работать в рамках одной CMS (далее такие части будем называть модулями CMS), так и взаимодействовать с другими CMS и программами на вэб-сервере (далее - инструменты). Оптимизация и введение новых открытых технологий в работу ядра CMS и модулей является правилом хорошего тона. На практике, ядро CMS не нуждается в частом обновлении: любой сайт успешно работает и на изначально созданном ядре системы. Любые серьёзные изменения основного кода CMS могут повлиять на работоспособность всех модулей и инструментов, которые были установлены на сайтах до изменений. Получается, при изменении ядра CMS (например, оптимизация, которая немного уменьшает нагрузку на вэб-сервер или экономит место в памяти) необходимо обеспечить полную совместимость всех существующих к нему модулей, в том числе написанных сторонними разработчиками. Это трудозатратно, если, например, изменения в ядре были связаны с архитектурой. Следовательно, изменять работу ядра CMS просто невыгодно. Поэтому новые улучшения CMS, которые появлялись уже после начала эксплуатации какой-либо системы не входили в состав ядра, а подключались как отдельные модули или инструменты.

Примером такого инструмента является Memcached [3], который обеспечивает кэширование в оперативную память (далее RAM). Основное назначение - по ключу положить данные в память и быстро их вернуть по соответствующему запросу. Этот инструмент был реализован Брэдом Фитцпатриком (Brad Fitzpatrick) в 2003 году в рамках работы над проектом сайта LiveJournal. Он использовал Memcached для разгрузки базы данных от запросов при формировании веб-страниц. В настоящее время его применяют во многих крупных проектах, например, Wikipedia, YouTube, Facebook и другие.

Аналогичным инструментом является Redis [4], появившийся в 2009 году. Он имеет более расширенный функционал: поддерживает сложные типы данных (например, списки), поддерживает высокоуровневые операции (например, сортировка), имеет встроенную поддержку репликации данных на другие сервера с целью защитить данные и повысить скорость записи/чтения данных при нагрузке. К сожалению, инструмент не поддерживается ОС Windows.

Существенным минусом Memcached и Redis является необходимость отслеживать актуальность данных и хранить информацию о ключах, по которым данные можно получить. При существенном масштабировании сайта это сильно усложняет сам процесс вэб-разработки страниц.

Но хранение данных возможно не только в оперативной памяти, но и на жёстком диске.

В рамках данной работы был предложен новый подход, который позволяет при изменении данных автоматически поддерживать её актуальность как в оперативной памяти, так и в других местах, где данная информация хранится. Если на сайте используется кэширование, которое зависит от изменённой информации, то этот кэш будет вычислен заново.

Поскольку RAM является более быстрой, предложено в оперативную память реплицировать информацию, которая хранится в базе данных, а на жёстком диске хранить вычисленные вэб-страницы или их части (далее кэш), т.к. они занимаю больше места.

Чтобы эффективно использовать память жёсткого диска для хранения вычисленных страниц, предлагается каждую вэб-страницу поделить на статичную часть, которая не меняется, и динамические части, которые зависят от параметров загрузки. Отдельное хранение вычисленных частей в памяти жёсткого диска позволяет их использовать повторно на разных страницах, экономя память и вычислительные мощности.

Рассмотрим подробнее, от чего может меняться состояние динамических частей (далее - параметры загрузки):

- url-запросы,
- тип запроса (GET или POST),
- cookies-файлы (например, если сайт хранит данные о поведении пользователя в cookies и на вэб-странице надо реализовать раздел «Вы смотрели ранее» на основании этих данных),
- текстовый идентификатор, позволяющий различным динамическим частям использовать общий кэш. Если у двух разных динамических частей будет одинаковый идентификатор, то место хранения в памяти жёсткого диска будет одинаковым при одинаковых параметрах загрузки. Это позволяет вычислять динамические блоки один раз, после чего использовать уже вычисленный кэш,
- уникальный идентификатор посетителя, например, логин. Позволяет кэшировать индивидуальное для каждого посетителя состояние динамических частей.

В рамках данной разработки всю логику динамического блока решено выделить в отдельный класс (далее - модуль). Модуль кэширует на жёсткий диск свои результаты для каждого параметра запроса, так же, как и динамическая часть. Результатом вычисления модуля является список из следующих данных:

- HTML код,
- javascript-код,
- javascript-код, который запускается по событию onload при загрузке страницы,
- CSS стиль,
- POST ответ в виде текстовой строки,
- breadcrumb – данные, позволяющие формировать хлебные крошки.

Модуль не просто отображает данные, но и благодаря обработчику POST-запросов может отправлять запросы к вэб-серверу и получать ответы без перезагрузки страницы (технология AJAX [5]). Если данные, передаваемые с помощью AJAX, не зависят от параметров загрузки и меняют отображение модуля, то кэш модуля будет неизменным, ведь изначальный код страницы не именился. Например, если модуль отображает динамичную файловую структуру: слева список папок, а при клике на папку появляются вложенные в папку файлы (рис. 1).

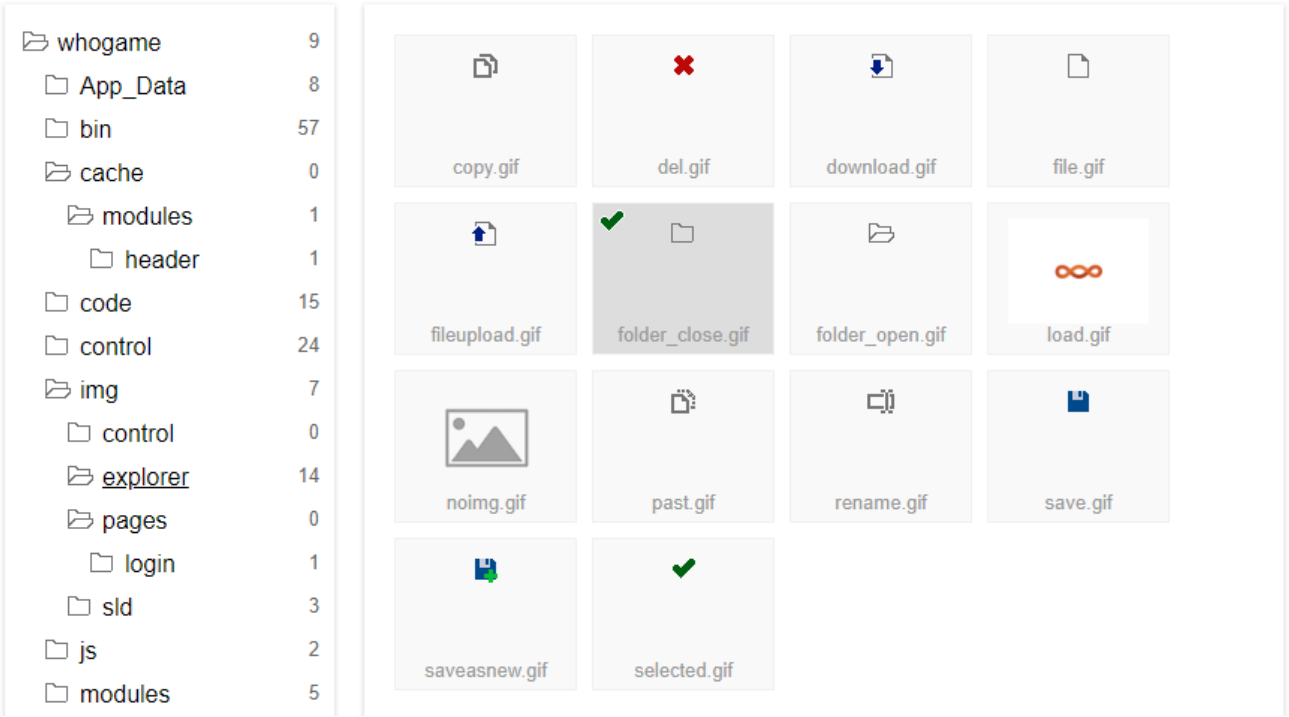

Рис. 1. Модуль, отображающий файловую систему

В итоге, если состояние модулей для определённых параметров запроса уже были вычислены ранее, скорость загрузки страницы будет максимально быстрой, поскольку не потребуется никаких вычислений. Рассмотрим ситуацию, как можно ускорить и сам процесс вычисления модулей.

Обычно вычисления модуля основывается на запросе данных из места их хранения (обычно это база данных SQL) и дальнейшими операциями с этими данными. Считаем, что вычисления уже оптимизированы разработчиком модуля. Необходимо ускорить время получения данных из базы. Одно из

решений – часть данных, которые часто запрашиваются, дублировать в оперативной памяти, на подобии инструментов Redis и Memcache, описанных ранее.

Обычно вэб-программист отвечает за серверную и клиентскую часть и не занимается вопросами системной работы CMS. Поэтому при разработке html-страницы ему не принципиально, где хранятся данные. CMS должна самостоятельно понимать, как максимально быстро вернуть информацию по соответствующему запросу.

В нашем проекте была реализована собственная система обработки данных, которая часть информации в момент первой загрузки сайта помещает в RAM. Какая именно информация будет в оперативной памяти указывает вэб-программист один раз при начальной настройке сайта. Вся дальнейшая работа вэб-разработчика с данными основывается на обобщённом запросе к CMS с указанием, какие данные он хочет использовать без уточнения места хранения. Система запрашивает информацию из оперативной памяти, а если её там нет, запрашивает её в базе данных.

В некоторых CMS (в том числе предлагаемой) используется модель хранения данных Сущность-Атрибут-Значение (Entity–Attribute–Value (EAV)). Такая модель имеет ряд преимуществ:

1. позволяет изменять атрибуты и параметры для сущности без изменения структуры базы данных,
2. позволяет избежать избыточности при хранении данных.

В данной работе пришлось столкнуться с некоторыми проблемами:

1. усложнение запросов выборки по атрибутам, которые требуют больше времени из-за невозможности применения стандартных способов индексации,
2. необходимость отслеживания дублирующих связей,
3. необходимость отслеживания связей между несуществующими записями таблицы.

Указанные выше проблемы были решены в рамках данной работы. Рассмотрим подробнее пример предложенной структуры SQL для хранения данных пользователей (рис. 2).

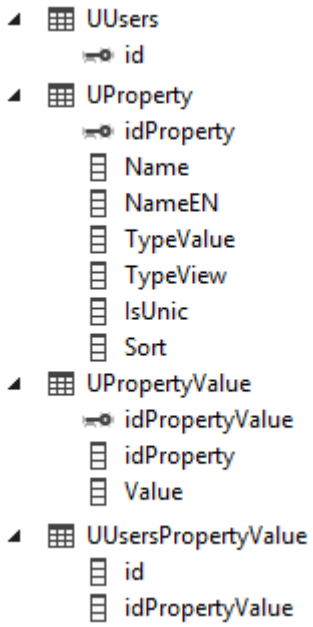

Рис. 2. Система хранения данных о пользователях в рамках модели EAV

Таблица UUsers хранит id пользователей.

Таблица UProperty хранит данные о самих атрибутах. Например, для у атрибута «Адрес» есть (Рис. 3):

- уникальный идентификатор «idProperty» равный «2»,
- указано имя на русском языке,
- название латинице,
- тип значений для этого атрибута,
- способ отображения параметра (это может быть, как обычное текстовое поле, так и слайдер в двумя ползунками),
- должны ли быть значения адресов уникальными,
- сортировка атрибута относительно других атрибутов пользователя.

	idProperty	Name	NameEN	TypeValue	TypeView	IsUnic	Sort
	2	Группа	usersgroup	0	3	False	0
	3	Адрес	adress_country	1	0	False	55

Рис. 3. Система хранения данных о пользователях в рамках модели EAV

Таблица UPropertyValue хранит значения атрибутов.

Таблица UUsersPropertyValuехранит связи между атрибутами и пользователями.

Чтобы выполнить выборку данных максимально быстро мы предлагаем сделать один сложный запрос с использованием оператора JOIN вместо несколько простых. Команда INNERJOIN позволяет SQL обрабатывать таблицы в порядке, отличном от указанного в запросе. Это дает возможность базе данных максимально быстро вернуть результат.

Сама функция запроса возвращает сущности, которые удовлетворяют требования к атрибутам и их значениям, например:

- требования к сущности о наличии хотя бы одного значения этого атрибута,
- проверка о наличии/отсутствии NULL среди значений,
- наличие у определённого атрибута значения, которое содержит указанную строку или полностью совпадает.

Сложность таких запросов в том, что мы должны их сначала автоматически сгенерировать в зависимости от указанных требований к атрибутам, а потом уже выполнять.

Чтобы в модели EAV строить такие запросы в базу данных для любого количества атрибутов с любыми требования к их значениям был создан специальный компоновщик запросов для всех возможных условий.

Сохранять результаты запросов к базе данных и всех их вариаций не имеет смысла – слишком большое количество подмножеств результатов может быть. Уменьшить число запросов к базе данных можно с помощью заранее подгруженных таблиц в RAM.

Обычно при создании сайта вэб-программист заранее понимает, как часто к сущностям будут применяться сложные запросы с выборками. Поэтому было предложено вэб-программисту самому решать, какие сущности подгружать в оперативную память, а какие нет.

Данные хранятся в оперативной памяти в другом формате, более удобном для получения различных выборок. По сути, мы храним список сущностей, к которым уже привязаны их атрибуты и значения. Поэтому запросы на выборку происходят гораздо проще, так как мы можем сразу оценить, подходит ли сущность под требования, минуя получение промежуточных результатов. Такая система хранения данных в RAM является избыточной, ведь параметры повторяются, но это плата за скорость их обработки.

Также дополнительно подгружается список всех параметров и значений для ускоренного получения списка всех атрибутов и их значений без привязки к сущностям.

На основе предложенных подходов был разработан сайт http://chik-chik.ru/, который показал в среднем хорошие результаты в нагрузочном тестировании по сравнению с другими крупными сайтами (см. табл. №1). Для тестирования мы выбирали сервера, близкие к физическому расположению вэб-сервера сайта.

	min, мс	max, мс	**avg, мс**	size, мб	Html/js/img %
chik-chik.ru	170	6 930	**231**	0.7 мб	11 / 12 / 74 %
amazon.com	28	999	**316**	6.7 мб	2 / 23 / 70 %
ozon.ru	1 040	24 410	**1 440**	2.7 мб	1 / 35 / 53 %
yandex.ru	599	1 760	**717**	0.9 мб	10 / 32 / 15 %

Таблица №1. Скорость загрузки страниц по данным Loadimpact [6]

В таблице указана минимальная (столбец min) и максимальная (столбец max) скорость загрузки страницы, но их нельзя считать объективным показателем для сравнения. В данном случае стоит считать среднее значение (столбец avg) на всём периоде тестирования, которые уже предоставляет Loadimpact. Стоит так же обратить внимание на специфику самих сайтов: тестируемый прототип является интернет-магазином и на главной странице подгружает определённые товары по определённому алгоритму. Но в отличии от аналогичных страниц у Amazon.com (крупнейший интернет-магазин в Мире) и Ozon.ru (крупнейший интернет-магазин в России), размер передаваемых данных у chik-chik.ru в разы меньше. Поэтому пришлось добавить Yandex.ru (популярнейший поисковик в России) с похожим размером страницы для частоты эксперимента. Прототип, реализованный с помощью предложенного подхода остался лидером по средней скорости загрузки.

Заключение

В данной работе были предложены новые подходы к работе сайтов с данными в оперативной памяти и на жёстком диске. Был разработан алгоритм, при котором вэб-программист запрашивает данные без уточнения их физического места хранения (RAM или ROM). Был разработан первый прототип на основе предложенных подходов, в настоящее время готовится вторая версия. На основе разработанного прототипа были получены положительные результаты нагрузочного тестирования сайта по сравнению с такими известными сайтами, как Amazone.com, Ozon.ru и Yandex.ru.

Программа для ЭВМ “CMS с автоматическим кэшированием данных” зарегистрирована Федеральной службой по интеллектуальной собственности в реестре программ для ЭВМ, в результате чего выдано свидетельство о государственной регистрации программы для ЭВМ No 2018660021 от 20 июля 2018 г.

Статья подготовлена при поддержке Российского фонда фундаментальных исследований, проекты 16-01-00347-а, 18-07-00841-а, 18-07-01292-а.

Литература

1. CMS для электронной коммерции. — URL: https://www.magento.com/
2. CMS общего назначения. — URL: https://www.1c-bitrix.ru/products/cms/
3. Сервис кэширования данных в оперативной памяти Memcached. — URL: https://memcached.org/
4. Хранилище данных типа «ключ — значение» Redis. — URL: https://redis.io/
5. AJAX - Подход к построению интерактивных пользовательских интерфейсов веб-приложений . — URL: https://www.w3.org/TR/XMLHttpRequest/
6. Инструмент для нагрузочного тестирования сайтов. — URL: https://loadimpact.com/

References

1. CMS for e-commerce. — URL: https://www.magento.com/
2. CMS general purpose. — URL: https://www.1c-bitrix.ru/products/cms/
3. Free & open source, high-performance, distributed memory object caching system Memcached. — URL: https://memcached.org/
4. Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. — URL: https://redis.io/
5. AJAX - Asynchronous JavaScript And XML. — URL: https://www.w3.org/TR/XMLHttpRequest/
6. Performance testing for Developers. — URL: https://loadimpact.com/