УДК 004.422.833

Н. П. Цыганенко, асп. (БГТУ, г. Минск)

КЭШИРОВАНИЕ ДАННЫХ В ВЕБ-ПРИЛОЖЕНИЯХ НА ОСНОВЕ NOSQL-ХРАНИЛИЩ

Основная цель веб-приложений и других интернет ресурсов в конечном итоге заключается в предоставлении посетителям информации в различных формах (текст, графика, видео и т.д.). Интернет пространство сегодня – это насыщенная данными среда. Поэтому, в условиях жесткой конкуренции за внимание пользователей, проблема скорости получения пользователем информации более чем актуальна. Ее решением является минимизация времени между запросом, посылаемым на сервер, и моментом полной загрузки ответного сообщения с запрашиваемым контентом [1].

Одной из точек минимизации времени получения информации является канал между серверным веб-приложением и реляционной базой данных. Для ускорения данного канала используют механизм кэширования. Суть кэширования заключается в том, что часть информации из реляционной базы данных дублируется в хранилище с более высокой скоростью доступа – кэше. В момент, когда пользователь запрашивает информацию у веб-приложения сначала проверяется кэш. В случае если в нём необходимых данных не оказалось, запрос перенаправляется к реляционной базе данных. Таким образом в случае, если необходимая информация находится в кэше, то получается значительный выигрыш во времени, по сравнению с простым использованием базы данных [2].

Двумя наиболее важными характеристиками эффективности системы кэширования являются скорости доступа к хранимой информации и вероятность попадания в кэш. Скорость доступа к информации – это время от запроса информации веб-приложением у системы кэширования до полного получения необходимых данных. Вероятность попадания – это процент успешных разрешений запросов на получение определенной информации к их общему числу [3].

Величина, на которую увеличивается скорость доступа к информации, зависит от типа хранилища на базе которого строится система кэширования. Двумя наиболее популярными вариантами хранилищ кэша являются оперативная память(RAM) и нереляционные базы данных (NoSQL).

Наибольшей скоростью доступа обладают хранилища, построенные на базе оперативной памяти. Но такие системы обладают малой вместимостью, поэтому могут использоваться только для приложений

с небольшим объемом данных. Если использовать такие системы для больших приложений, то вероятность попадания в кэш будет минимальной и произойдёт снижение скорости доступа к информации по сравнению с обычным использованием реляционных баз данных [4].

Оптимальным вариантом для веб-приложений является использование в качестве хранилища кэша нереляционных баз данных. Системы на основе NoSQL-хранилищ обладают более высокой скоростью доступа, по сравнению с реляционными базами данных, а также способны вместить большие объемы информации, так как располагаются на жестких дисках. Хотя нереляционные базы данных имеют преимущество в скорости работы, однако, для большинства веб-приложений они не подходят в качестве основного хранилища данных. Это обусловлено тем, что реляционные базы данных позволяют удобней отображать структуру предметной области и имеют мощный математический аппарат в виде реляционной алгебры.

Для увеличения вероятности попадания кэш используются различные алгоритмы вытеснения и предварительной загрузки данных.

В настоящее время нет спецификаций и стандартов по построению эффективных систем кэширования, что приводит к неэффективности, плохой масштабируемости, невозможности повторного использования и неподдерживаемости систем кэширования в веб-приложениях. Таким образом, для обеспечения наиболее высокой скорости доступа к информации веб-приложений необходима разработка общей архитектуры систем кэширования, а также методов оптимизации её работы. Важной особенностью хорошей системы кэширования является инкапсуляция работы с реляционной базы данных таким образом, что веб-приложение ничего не знает о том, что информация кэшируется. Таким образом для работы с данными у веб-приложения по-прежнему остается единый интерфейс.

ЛИТЕРАТУРА

1. Esposito, D. Microsoft .NET - Architecting Applications for the Enterprise.Redmond: Microsoft Press, 2014. 416 p.

2. Watson, B. Writing High-Performance .NET Code. Redmond: Microsoft Press, 2014. 280 p.

3. Millett, S. Professional ASP.NET Design Patterns. Indianapolis: Wrox, 2010. 720 p.

4. Da Silva, M. Redis Essentials. NY: Packt Publishing, 2015. 230 p.