Н. В. ЕРМАКОВ, С. А. МОЛОДЯКОВ
Санкт-Петербургский политехнический университет Петра Великого, г. Санкт-Петербург

МОДЕЛЬ КЭШИРОВАНИЯ ДЛЯ СИСТЕМЫ БЫСТРОГО ДОСТУПА ФАЙЛОВ

В настоящей работе рассмотрена прикладная система хранения. Эта система эффективно хранит данные в зависимости от их особенностей. Основное внимание уделено вопросу кэширования данных на уровне прикладного сервера. Чтобы ускорить получение данных, мы предлагаем использовать двухсекционный кэш. В первой секции находятся результаты запросов к базе данных, а во второй – данные из файловой системы. Для исследований были разработаны аналитическая и имитационная модели кэширования. Приводятся графики исследования моделей в зависимости от размера кэша и других параметров.

Введение. За многие годы работы разные учреждения и предприятия накопили большие объемы информации, которые продолжают увеличиваться [1], [2]. Дополнительные возможности, которые предоставляет большое количество данных, требуют разработки новых систем хранения данных.

Файловые системы обычно обеспечивают хранение слабо структурированной и неструктурированной информации. Системы управления базами данных хранят структурированную информацию и решают множество проблем, которые затруднительно или вообще невозможно решить при использовании файловых систем [3]–[5]. Файловая система лучше подходит для хранения большого количества неструктурированных данных, чем база данных [6]. Поэтому, если необходимо хранить и структурированные и неструктурированные данные, файловая система используется вместе с SQL, NoSQL базой данных или поисковым движком. У поисковых движков меньше время поиска, но больше время вставки по сравнению с другими базами данных, например, с MongoDB [7].

Одним из методов повышения производительности системы хранения данных является кэширование. Аналитическое и имитационное моделирование системы кэширование применяется для оценки эффективности в зависимости от различных параметров. При аналитическом моделировании могут применяться модели на основе цепей Маркова [8]–[10]. Для имитационного моделирования различных протоколов когерентности кэшей (MSI, MESI и других) применяются модели на основе диаграмм состояний [11], [12].

Прикладная система хранения данных была разработана для компании EPAM. Она предназначена для хранения информации о фармацевтических исследованиях. Система состоит из файловой системы, базы данных, поискового движка и прикладной программы, размещенной на сервере. Вся необходимая информация классифицируется и распределяется. Неструктурированные данные хранятся в файловой системе, структурированные – в базе данных, не меняющиеся – в поисковом движке. Результаты часто используемых запросов к базе данных кэшируются.

Предлагаемый доклад посвящен исследованию возможностей повышения производительности системы хранения данных за счет более эффективного использования кэширования. Для исследований были разработаны аналитическая и имитационная модели двухсекционного кэша. В этот кэш помещаются результаты запросов к базе данных и файлы из файловой системы.

Аналитическая модель кэширования. На рис. 1 приведена схема кэширования в системе хранения данных. Клиенты запрашивают у сервера приложений файлы по некоторому заданному условию. Сервер приложений запрашивает список файлов, удовлетворяющих этому условию, из базы данных. Результаты запроса помещаются в сегмент кэша S1. Затем сервер приложений запрашивает требуемые файлы из файловой системы. Запрошенные файлы помещаются в сегмент кэша S2. Запрошенные данные также могут помещаться в кэши клиентов L1. Если данные могут потребоваться нескольким клиентам, то они могут быть помещены в кэши проски-серверов L2.

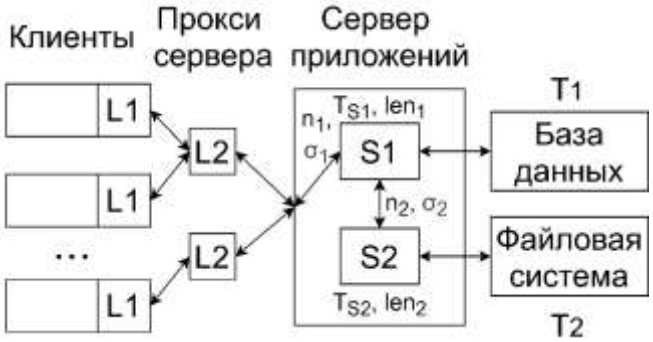

Рис. 1. Схема кэширования в системе хранения данных

T_{S1} и T_{S2} – среднее время чтения из сегментов кэша S1 и S2. T_1 и T_2 – среднее время чтения из базы данных и файловой системы. len_1 и len_2 – это размеры сегментов кэша S1 и S2. n_1 – количество запросов в базу, n_2 – количество запрошенных файлов. В реальных системах хранения данных одни данные всегда запрашиваются чаще других. Предположим, что «номера» запросов к базе данных и «номера» запрошенных файлов от пользователей распределены по нормальному закону. σ_1 и σ_2 – среднеквадратичное отклонение «номеров» запросов к базе данных и файлам. «Номера», которые находятся рядом с математическим ожиданием, запрашиваются чаще.

Предположим, что в кэше всегда хранятся наиболее часто запрашиваемые данные с «номерами» от $\mu - len/2$ до $\mu + len/2$, где: len – размер кэша, μ – математическое ожидание, σ – среднеквадратичное отклонение. Тогда выражение (1) – это вероятность попадания в кэш.

$$p_{hit} = F\left(\mu + \frac{len}{2}\right) - F\left(\mu - \frac{len}{2}\right) = \text{erf}\left(\frac{len}{2\sqrt{2}\sigma}\right) \quad (1)$$

Выражение (2) – это общее время работы системы.

$$T_{total} = n_1(p_{hit1}T_{S1} + (1 - p_{hit1})T_1) + n_2(p_{hit2}T_{S2} + (1 - p_{hit2})T_2) \quad (2)$$

Выражение (3) – это общее время работы системы при отсутствии кэширования.

$$T_{without\ cache} = n_1T_1 + n_2T_2 \quad (3)$$

Выражение (4) – это выгода от использования кэширования. Выгода зависит от отношения запросов в базу и к файлам $n1/n2$, отношения длины кэша и среднеквадратичного отклонения len/σ для каждого сегмента, времени чтения из базы T_1, файловой системы T_2 и сегментов кэша T_{S1} и T_{S2}.

$$gain = \frac{T_{without\ cache}}{T_{total}} = \frac{\frac{n_1}{n_2}T_1 + T_2}{\frac{n_1}{n_2}\left(\text{erf}\left(\frac{len_1}{2\sqrt{2}\sigma_1}\right)T_{S1} + \left(1 - \text{erf}\left(\frac{len_1}{2\sqrt{2}\sigma_1}\right)\right)T_1\right) + \left(\text{erf}\left(\frac{len_2}{2\sqrt{2}\sigma_2}\right)T_{S2} + \left(1 - \text{erf}\left(\frac{len_2}{2\sqrt{2}\sigma_2}\right)\right)T_2\right)} \quad (4)$$

Иммитационная модель кэширования. На рис. 2 приведена функциональная схема разработанной модели кэширования, созданной в среде MATLAB/Simulink. Модель состоит из двух генераторов чисел с нормальным распределением, двух моделей кэша, и части, подсчитывающей выгоду от кэширования. В модели можно задать среднеквадратичное отклонение генераторов, размер сегментов кэша, среднее время чтение при попадании в кэш и промахе. Считаем, что среднее время чтения при попадании в первый сегмент кэша равно 1. Для подсчета выгоды среднее время без кэширования $T_1 + n_2/n_1 * T_2$ умножается на текущее время и делится на общее время с кэшированием.

Входные параметры моделей кэшей – это последовательность номеров (вход Source) и размер кэша (рис. 3). Блок «Cache model» хранит кэшированные номера со входа «Source». Выход «result» блока «Cache model» равен 1, если в кэше есть номер со входа «get», и 0, если нет. В модели реализована политика вытеснения LRU (least recently used). Блок «Priority queue» хранит время последнего запроса номера. С задержкой в один такт запрошенный номер добавляется в «Cache model» и «Priority queue». Если вход «Pop» = true, то на выходе «Out» будет номер с минимальным временем. Неиспользованный дольше всех номер вытесняется из

очереди и кэша, если до этого был кэш промах (result = 0) и размер кэша равен максимальному. Два счетчика считают количество кэш попаданий и промахов.

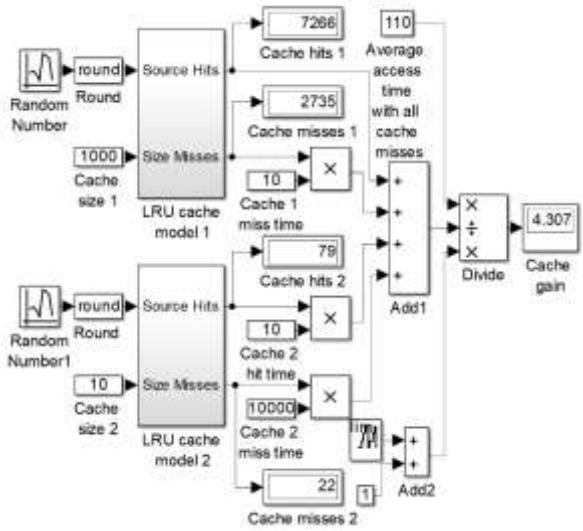

Рис. 2. Модель кэширования в среде MATLAB/Simulink

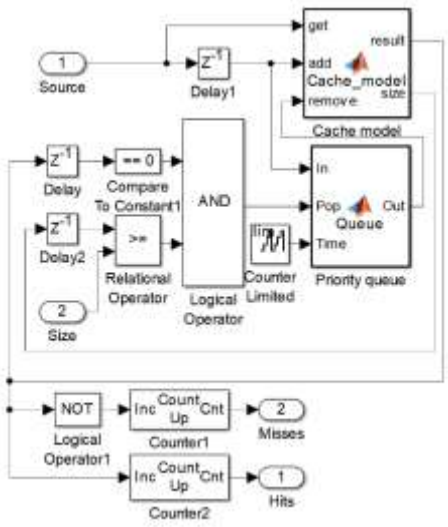

Рис. 3. Блок «LRU cache model»

Результаты. Пусть средний размер файла – 1 Гб, средний размер ответа из базы – 100 байт, размер сегмента кэша для базы данных – 100 КБ, для файлов – 10 Гб. В среднем в кэш помещается $len_1 = 1000$ запросов к базе и $len_2 = 10$ файлов. Количество запросов к базе равно $n_1 = 10000$, а к файлам – $n_2 = 100$. Среднеквадратичное отклонение запросов к базе равно $\sigma_1 = 100\sqrt{10}$, к файлам – $\sigma_2 = \sqrt{10}$. Вызывается около 1000-2000 различных запросов к базе данных и 10–20 файлов, т. к. 99.7 % запросов попадают в промежуток от -3σ до +3σ. Среднее время скорости чтения запроса из кэша базы данных равно $T_{S1} = 1$, из базы – $T_{S2} = 10$, из кэша файлов – $T_1 = 10$, из файловой системы – $T_2 = 10000$. Исследуем зависимость выгоды от использования кэша в зависимости от одного параметрах при остальных фиксированных параметрах и сравним с выгодой по (4).

На рис. 4 приведена зависимость выгоды от размера второго сегмента кэша len_2. При $len_2 \geq 14$ выгода перестает увеличиваться и равна 6.3. Это связано с тем, что кэш изначально пустой. Поэтому вначале всегда будут кэш промахи, даже размер кэша больше всех различных запрашиваемых данные. Также политика вытеснения LRU хуже идеальной политика вытеснения в формуле. Мы ожидаем, что выгода, рассчитанная по формуле, всегда будет больше, чем выгода, полученная при моделировании.

На рис. 5 приведена зависимость выгоды от времени доступа к базе T_2, когда происходит кэш промах во втором сегменте. Чем больше время, тем больше выгода.

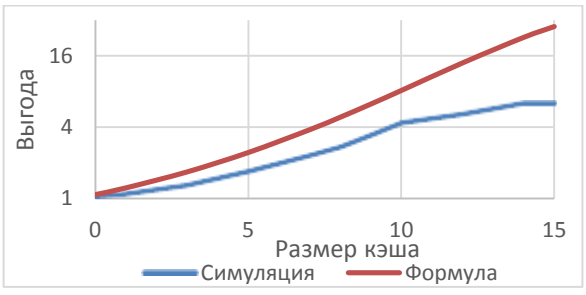

Рис. 4. Зависимость выгоды от размера кэша S2

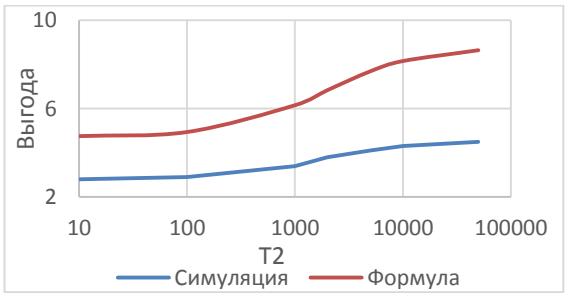

Рис. 5. Зависимость выгоды от времени доступа T2

Заключение. В этой статье рассмотрены аналитическая и имитационная модели двухсекционного кэша. Этот кэш может применяться для систем хранения данных, состоящей из нескольких компонентов, например, из базы данных и файловой системы. В аналитической модели рассмотрен идеальный случай, и поэтому она показывает лучшие результаты. Разработанные модели могут быть использованы для определения оптимальных размеров кэша

в системе хранения данных в зависимости от распределения запросов к системе. Кэширование может существенно ускорить получение данных, которые часто запрашиваются и редко изменяются.

ЛИТЕРАТУРА

1. Laboshin L.U., Lukashin A.A., Zaborovsky V.S. The Big Data approach to collecting and analyzing traffic data in large scale networks. In: Procedia Computer Science, 2017. Vol. 103, P. 536–542. doi: 10.1016/j.procs.2017.01.048
2. Voinov N., Garzon K. R., Nikiforov I., Drobintsev P. Big Data processing system for analysis of GitHub events. In: 2019 XXII International Conference on Soft Computing and Measurements (SCM), 2019. P. 187–190. doi: 10.1109/SCM.2019.8903782
3. Coronel C., Morris S. Database systems: design, implementation, and management. Boston, USA: Cengage Learning, 2019.
4. Ramakrishnan R., Gehrke J. Database management systems. McGraw Hill, 2003.
5. Connolly T., Begg C. Database systems: practical approach to design, implementation, and management. Harlow: Pearson Education Limited, 2015.
6. Sears R., Van Ingen C., Gray J. To blob or not to blob: Large object storage in a database or a filesystem? Microsoft Technical Report, 2007.
7. Abubakar Y., Adeyi T. S., Auta I. G. Performance evaluation of NoSQL systems using YCSB in a resource austere environment. In: International Journal of Applied Information Systems, 2014. Vol. 7, P. 23–27. doi:10.5120/ijais14-451229
8. Ben-Ammar H., Hadjadj-Aoul Y., Rubino G., Ait-Chellouche S. On the performance analysis of distributed caching systems using a customizable Markov chain model. In: Journal of Network and Computer Applications, 2019. Vol. 130, P. 39–51. doi: 10.1016/j.jnca.2019.01.011
9. Chen C., Beltrame G. An adaptive Markov model for the timing analysis of probabilistic caches. In: ACM Transactions on Design Automation of Electronic Systems (TODAES), 2017. Vol. 23, P. 1–24. doi: 10.1145/3123877
10. Ammar H.B., Chellouche S.A., Aoul Y.H. A Markov chain-based Approximation of CCN caching Systems. In: 2017 IEEE Symposium on Computers and Communications (ISCC), 2017. P. 327–332. doi: 10.1109/ISCC.2017.8024551
11. Kehagias D., Raptis I. An Android-based MESI cache coherence simulator. In: The 5th International Virtual Conference on Advanced Scientific Results, 2017, P. 194–199. doi: 10.18638/scieconf.2017.5.1.422
12. Sensfelder N., Brunel J., Pagetti C. Modeling cache coherence to expose interference. ECRTS, 2019. doi: 10.4230/LIPIcs.ECRTS.2019.18

N.V. Ermakov, S.A. Molodyakov, (Peter the Great St.Petersburg Polytechnic University, St. Petersburg)

A Caching Model for a Quick File Access System

In this paper, we consider the developed storage system. This system effectively stores data depending on their features. The main attention is paid to the issue of data caching at the application server level. To speed up data retrieval, we suggest using a two-section cache. The first section contains the results of queries to the database, and the second contains data from the file system. Analytical and simulation caching models have been developed for research. The graphs of model research depending on the cache size and other parameters are provided.