ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 004.75
DOI: 10.17586/0021-3454-2017-60-3-234-238

ГОРИЗОНТАЛЬНОЕ МАСШТАБИРОВАНИЕ БАЗЫ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ КОНСИСТЕНТНОГО ХЕШИРОВАНИЯ

А. С. Зернов, А. А. Ожиганов

Университет ИТМО, 197101, Санкт-Петербург, Россия
E-mail: alexeyworking@gmail.com

Рассматривается задача организации распределенного хранения больших объемов данных в крупномасштабных кластерных системах. Представлен способ распределения данных по узлам кластера с использованием консистентного хеширования. Описаны базовый метод консистентного хеширования и усовершенствованный метод с использованием виртуальных узлов.

Ключевые слова: *распределенные хранилища, масштабирование базы данных, массивы данных, шардинг, высоконагруженные системы*

Введение. В настоящее время множество различных компаний сталкиваются с проблемами, связанными с постоянным увеличением объема данных, что отрицательно сказывается на производительности сервера базы данных. В отличие от „вертикального“ масштабирования (повышения производительности основного сервера) при „горизонтальном“ масштабировании серверы объединяются в кластер и вся нагрузка распределяется между ними [1]. Однако в такой распределенной системе возникает проблема равномерного распределения данных внутри кластера, а также перераспределения данных при добавлении или удалении одного из серверов [2].

Шардирование реляционной базы данных. Основным методом горизонтального масштабирования базы данных является шардинг [3] — разбиение большого массива данных на малые блоки и дальнейшее распределение их по узлам кластера. В случае реляционной базы данных такими блоками являются строки таблиц, которые при шардинге хранятся на различных узлах [4]. Строка представляет собой множество элементов с уникальным идентификатором (первичным ключом):

$$\mathtt{table(key\_id)} = \{\mathtt{elem}_1, \mathtt{elem}_2, \ldots, \mathtt{elem}_n\}.$$

Основная задача при реализации шардинга — выбор механизма для определения узла, на котором будет храниться та или иная строка таблицы. Как известно [5], первой функцией определения принадлежности строки к конкретному узлу кластера является хеш-функция первичного ключа строки с последующим получением результата деления значения хеш-функции по модулю числа узлов в кластере, т.е.

$$\mathtt{shard\_id} = \mathtt{hash(key\_id)\%N},$$

где `shard_id` — идентификатор узла, `key_id` — идентификатор строки (первичный ключ), для которого считается значение хеш-функции, `N` — число узлов кластера [5].

Особенности консистентного хеширования. Функция, описанная выше, обеспечивает равномерное распределение ключей по узлам кластера, однако при добавлении или удалении узла могут возникнуть проблемы. Изменение числа узлов (т.е. числа `N`) приведет к практически полной потере соотношения между ключами и узлами кластера и соответственно к необходимости заново перераспределять все ключи [6].

Решением данной проблемы является механизм консистентного хеширования, особенность которого заключается в возможности избежать полного перераспределения ключей при изменении числа узлов. Это означает, что перемещению будут подвержены только `K/N` ключей (где `K` — общее число ключей); также возможна ситуация, когда один из узлов может выйти из строя. В этом случае консистентное хеширование минимизирует влияние отказов узлов кластера и обеспечивает ускоренное восстановление системы после сбоя, так как перераспределению будут подвергнуты только те ключи, которые хранились на нефункционирующем узле [7].

Базовый метод консистентного хеширования. Базовый метод консистентного хеширования заключается в следующем: вся область значений используемой хеш-функции представляется в виде замкнутой окружности. На рис. 1, *а* показан пример окружности, представляющей собой нормированную область, приведенную к интервалу [0, 255] значений хэш-функции.

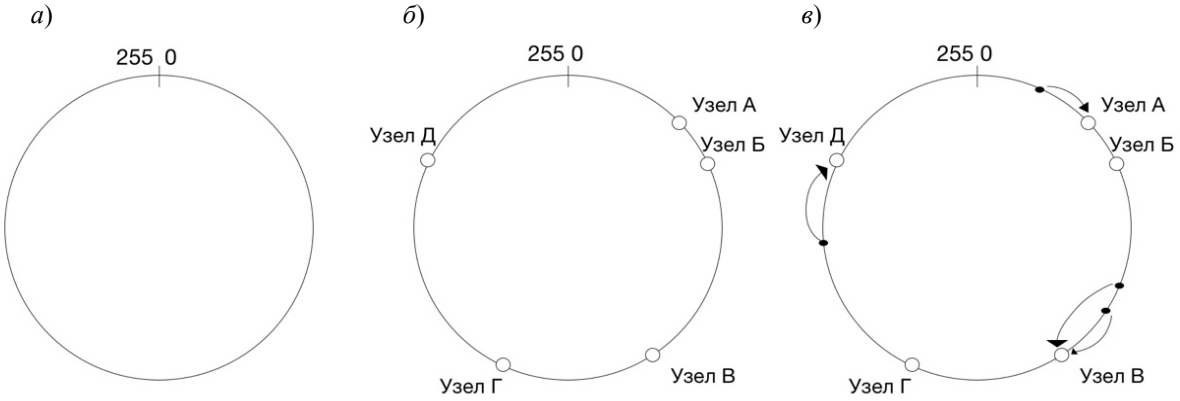

Рис. 1

Каждый узел кластера ассоциируется с точкой на этой окружности, т.е. с некоторым значением хэш-функции, которое может быть получено в результате ее применения к идентификатору узла. При этом хэш-функция может быть не полностью идентична той, с помощью которой будет вычислено значение для ключа строки, однако она должна соответствовать условию

$$\texttt{hash1(shard\_id)} \in \texttt{hash2(key\_id)},$$

где `hash1` — хеш-функция, применяемая к идентификатору узла, `hash2` — хеш-функция, применяемая к идентификатору строки.

Таким образом узлы распределяются на окружности (рис. 1, *б*).

Как и в случае с узлами, каждому ключу строки присваивается значение, соответствующее результату вычисления хеш-функции, и, следовательно, некоторое положение на окружности [8]. Между значением узла и значением ключа на окружности нет принципиальной разницы, поэтому для устранения коллизий необходимо, чтобы их идентификаторы отличались друг от друга [9].

Два соседних узла на окружности образуют дугу, за которую „отвечает“ узел, находящийся по ходу часовой стрелки первым. Если идентификатор строки, подвергшейся хешированию, входит в область значений на дуге, то между ними устанавливается связь (см. рис. 1, *в*).

При удалении узла из кластера или его добавлении происходят следующие действия:

— если один из узлов удаляется из кластера, то ближайший к нему узел по ходу часовой стрелки берет за него ответственность (рис. 2, *а*);

— при добавлении узла в кластер новый узел берет на себя ответственность за ближайший против хода часовой стрелки интервал (рис. 2, *б*).

а) *б*)

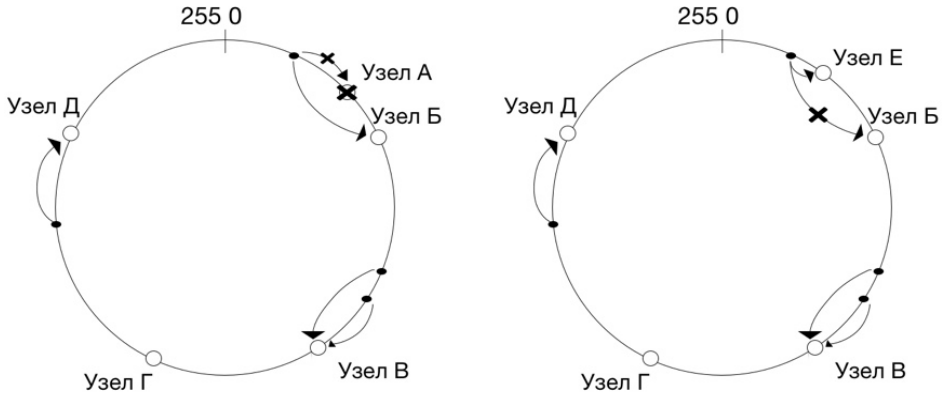

Рис. 2

Усовершенствованный метод консистентного хеширования. Базовый метод консистентного хеширования имеет ряд недостатков. Во-первых, так как узлы на окружности расположены случайным образом (см. рис. 1, *в*), то размеры дуг, за которые они отвечают, могут существенно различаться, а значит, некоторые узлы могут быть перегружены данными. Во-вторых, базовый метод не учитывает случай, когда узлы могут быть неравнозначны и иметь разную производительность.

Во избежание этих недостатков необходимо использовать концепцию „виртуального узла“. Данная концепция заключается в том, что вместо ассоциации узла с единственной точкой на окружности, каждый узел ассоциируется с множеством точек на ней. На рис. 3 заштрихованными точками показаны равномерно расположенные на окружности виртуальные узлы.

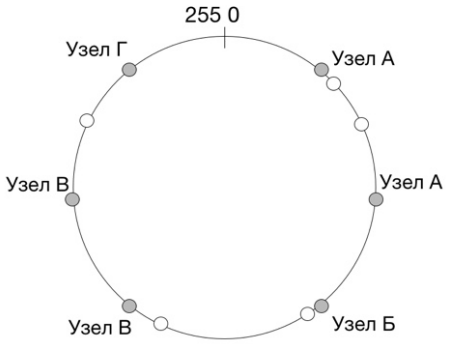

Рис. 3

Каждый физический узел может нести ответственность более чем за один виртуальный узел. Так как виртуальные узлы расположены на окружности равномерно, то если один из узлов становится недоступным, вся нагрузка равномерно перераспределяется на остальные. На основе параметров конкретного физического узла определяется число виртуальных узлов, за которые он несет ответственность, — чем больше производительность физического узла, тем больше виртуальных узлов следует с ним ассоциировать.

Заключение. Использование предложенного метода консистентного хеширования позволяет уменьшить нагрузку в высоконагруженных системах, равномерно распределить большой объем данных по узлам кластера, а также добавлять и удалять узлы кластера без полного перераспределения данных.

СПИСОК ЛИТЕРАТУРЫ

1. *Wang W., Zhang Z.* Balanced partition scheme for distributed caching systems to solve load imbalance problems // ACM SIGSOFT Software Engineering Notes. 2012. Vol. 37. P. 4—5. DOI: 10.1145/2382756.2382772.
2. *Tamer Özsu M., Valduriez P.* Principles of Distributed Database Systems. N. Y.: Springer-Verlag, 2011. P. 71—89.
3. *Chhanda R.* Distributed Database Systems. New Jersey, USA: Pearson, 2009. P. 119—133.
4. Dynamo: Amazon's Highly Available Key-value Store / Amazon.com, Inc., 2007. P. 208—212 [Электронный ресурс]: <http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf>.
5. *Fan L., Cao P., Almeida J., Broder A. Z.* Summary Cache: a Scalable Wide-Area Web Cache Sharing Protocol / Computer Science Department, Univ. of Wisconsin, Madison, USA, 1998. P. 287—292.
6. *Karger, Lehman E., Leighton T., Levine M., Lewin D., Panigrahy R.* Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web // Proc. of the 29th Annual ACM Symp. on Theory of Computing. 1997. P. 11—15.
7. *Swaminathan S.* Amazon dynamoDB: a seamlessly scalable non-relational database service // Proc. of the ACM SIGMOD Intern. Conf. on Management of Data, USA, ACM, 2012. P. 208—210. DOI: 10.1145/2213836.2213945.
8. *Aspnes J., Safra M., Yin Y.* Ranged hash functions and the price of churn // Proc. of the 19th Annual ACM-SIAM Symp. on Discrete Algorithms, SODA '08. 2008. P. 1066—1075.
9. *Haiying S., Cheng-Zhong Xu.* Hash-based proximity clustering for load balancing in heterogeneous DHT networks // Proc. of the 20th Intern. Conf. on Parallel and Distributed Processing. USA: IEEE Computer Society, Washington, 2006. P. 39—40.

Сведения об авторах

Алексей Сергеевич Зернов — студент; Университет ИТМО; кафедра вычислительной техники; E-mail: alexeyworking@gmail.com

Александр Аркадьевич Ожиганов — д-р техн. наук, профессор; Университет ИТМО; кафедра вычислительной техники; E-mail: ojiganov@mail.ifmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 05.07.16 г.

Ссылка для цитирования: *Зернов А. С., Ожиганов А. А.* Горизонтальное масштабирование базы данных с использованием консистентного хеширования // Изв. вузов. Приборостроение. 2017. Т. 60, № 3. С. 234—238.

HORIZONTAL SCALING OF THE DATABASE USING CONSISTENT HASHING

A. S. Zernov, A. A. Ozhiganov

ITMO University, 197101, St. Petersburg, Russia
E-mail: alexeyworking@gmail.com

The problem of the organization of distributed storage of large amounts of data in large-scale cluster systems is considered. A method of data distribution across cluster nodes using consistent hashing is presented. The basic method of consistent hashing, as well an improved method using virtual nodes are described.

Keywords: distributed storages, database scalability, datasets, sharding, high-loaded systems

Data on authors

Alexey S. Zernov — Student; ITMO University; Department of Computer Science; E-mail: alexeyworking@gmail.com

Aleksander A. Ozhiganov — Dr. Sci., Professor; ITMO University; Department of Computer Science; E-mail: ojiganov@mail.ifmo.ru

For citation: *Zernov A. S., Ozhiganov A. A.* Horizontal scaling of the database using consistent hashing // Journal of Instrument Engineering. 2017. Vol. 60, N 3. P. 234—238 (in Russian).

DOI: 10.17586/0021-3454-2017-60-3-234-238