УДК 004.421.2

РЕАЛИЗАЦИЯ ИНФРАСТРУКТУРЫ РАСПРЕДЕЛЕННОЙ ХЕШ-ТАБЛИЦЫ В РАМКАХ КЛАСТЕРНОЙ СИСТЕМЫ DNS

С. Н. Шилов, С. Д. Кургалин, А. А. Крыловецкий

Воронежский государственный университет

Поступила в редакцию 20.10.2012 г.

Аннотация. Данная статья посвящена проблеме распределения нагрузки на узлы в DNS кластере. В рамках статьи рассматривается метод построения подсистемы распределения нагрузки, основанный на распределенной хеш-таблице (Distributed Hash Table, DHT). Статья предоставляет детальное описание принципов построения указанной подсистемы и численные результаты кластерных измерений.

Annotation. This article is devoted to a load distribution problem on clusters in DNS a cluster. Within article the method of creation of a subsystem of the load distribution, based on the distributed hash table (Distributed Hash Table, DHT) is considered. Article provides the detailed description of the principles of creation of the specified subsystem and numerical results of cluster measurements.

Ключевые слова: кластер, кластеризация, распределенная хеш-таблица, распределение нагрузки, балансировка нагузки, система доменных имен .

Keywords: cluster, clustering, distributed hash table, load distribution, load balancing, domain name system.

ВВЕДЕНИЕ

Данная работа реализуется в рамках кластерной системы. Со второй половины XX века кластеризация проявила себя как очень перспективное направление развития компьютерных систем. История создания кластеров неразрывно связана с ранними разработками в области компьютерных сетей. Одной из причин для появления скоростной связи между компьютерами стали надежды на объединение вычислительных ресурсов.

Кластер состоит из трех основных компонентов: собственно вычислителей (обработчиков) – компьютеров, образующих узлы кластера, сети, объединяющей эти узлы, и программного обеспечения, заставляющего всю конструкцию работать в стиле единого компьютера. В роли вычислительных узлов могут выступать компьютеры различного класса: от самого обычного персонального компьютера до современного многопроцессорного сервера. Причем количество узлов в кластере практически ничем не ограничено.

Так как кластер состоит не менее чем из двух компьютеров (а в большинстве случаев их гораздо больше), то для обеспечения эффективности функционирования возникают вопросы балансировки нагрузки на узлы кластера, а также принципа выбора узла для какой-либо обработки. В зависимости от варианта реализации кластера, его назначения и обрабатываемых данных, выбираются различные принципы балансировки нагрузки.

Поставленная в данной работе задача состоит в реализации системы распределения нагрузки в кластере DNS.

Кластеризация системы DNS проектировалась следующим образом. Каждый узел кластера должен являться точкой входа в кластер. При поступлении запроса на какой-либо узел кластера, узел должен был решить, обработать ли ему поступивший запрос, или же передать его на обработку какому-либо другому узлу кластера, и, дождавшись результата обработки, отправить его в ответ на первоначальный запрос. Причем каждый из узлов кластера должен получать примерно одинаковое количество запросов на обработку, т.е. возникает вопрос балансировки нагрузки в кластере. Причем балансировка должна происходить с учетом определенных особенностей системы DNS. Начнем с того, что каждый DNS сервер содержит кэш DNS записей. При поступлении запроса на разреше-

ние домена сервер сначала ищет соответствующую запись в своем кэше, если находит – сразу отдает ответ запрашиваемому клиенту, иначе он начинает запрашивать вышестоящие серверы DNS, которые возвращают ответ, этот ответ сервер кэширует и возвращает клиенту. При следующем запросе этого домена сервер уже будет содержать соответствующую запись и сразу вернет ответ клиенту. Каждый узел кластера представляет собой DNS сервер со своим отдельным кэшем, и если узел будет обслуживать только определенные домены, то он будет максимально использовать свой кэш и редко обращаться к вышестоящим серверам, что крайне положительно скажется на производительности каждого узла в частности и кластера в общем. Т.е. система распределения нагрузки должна не просто равномерно распределять запросы среди узлов кластера, но, в то же время, ретранслировать запросы с определенными доменами определенным узлам кластера.

Отсюда же вытекают требования к масштабируемости кластера. При выходе узла из системы запросы с доменами, которые он должен обслуживать, необходимо равномерно перераспределить между оставшимися узлами, а при обратном входе в систему он должен «вернуть» себе именно эту часть запросов и в дальнейшем их обслуживать.

Решением поставленных задач может служить инфраструктура распределенной хеш-таблицы. Таким образом, цель данной работы – реализация инфраструктуры распределенной хеш-таблицы, обслуживающей данный DNS кластер. Основной ее задачей является выбор узла для обработки запрашиваемых данных. Выбор должен производиться таким образом, чтобы обеспечить распределение нагрузки на узлы кластера. Также, реализуемая система DHT должна обеспечивать масштабируемость кластера, корректным образом обрабатывать вход и выход узлов из системы.

АНАЛИЗ ПРОБЛЕМЫ

Первые четыре DHT системы были предложены в 2001 году. Это системы Content Addressable Network (CAN), Chord, Pastry и Tapestry. Все эти оригинальные протоколы обеспечивают функционирование распределенных децентрализованных систем посредством использования распределенных хеш-таблиц. Особое влияние на реализуемую в рамках данной работы систему DHT оказала распределенная система хранения информации Dynamo. Dynamo является быстрым, высоконадежным, распределенным хранилищем информации, представленной в виде пар ключ-значение. Некоторые моменты в части управления и использования области значений хеш-функции в реализуемой системе и системе Dynamo схожи, хотя и присутствуют существенные различия, обусловленные спецификой задачи.

Распределенная хеш-таблица (Distributed Hash Table или сокращенно DHT) – это инфраструктура, которая может быть использована для построения многих комплексных сервисов, таких как распределенные файловые системы, пиринговое распространение файлов и системы распространения контента, кооперативный web-кэш, многоадресная доставка (multicast), anycast, сервисы доменных имен, системы мгновенных сообщений и т.д. Иначе говоря, DHT – это класс децентрализованных распределённых систем, которые обеспечивают поисковый сервис, похожий по принципу работы на таблицу хешей. Основная задача DHT состоит в отображении хеш-значения, полученного от определенного параметра (в нашем случае это запрашиваемый домен), в узел системы.

Структура DHT может быть разбита на несколько основных компонентов. Она основывается на абстрактном пространстве ключей (keyspace). Схема разбиения пространства ключей распределяет принадлежность ключей среди участвующих узлов. Исторически первой функцией распределения нагрузки на основе хеш-функции была функция модуля:

$$Id = hash(key) \ \% \ N,$$

где Id – номер узла, key – значение, от которого считается хеш, N – количество функционирующих узлов кластера. Такая функция обеспечивает равномерное распределение ключей по узлам, однако проблемы возникают при переконфигурировании кластера: изменение количества узлов приводит к перемещению значительной части ключей по узлам, что эквивалентно потере значительной части ключей, потери актуальности кэшей узлов и их полному перезаполнению. Данный подход совершенно неприменим к реализуемой системе.

Альтернативой для данной функции является механизм консистентного хеширования (consistent hashing), который при переконфигурации кластера сохраняет положение ключей

по узлам. Основная идея данного подхода состоит в следующем: все пространство значений хеш-функции (например, в случае 64-х битной хеш-функции область ее значений будет составлять отрезок $[0, 2^{64}]$) представляется в виде кольца, в котором пространство «склеивается» в своем максимальном и минимальном значениях (0 и 2^{64}). С каждым узлом ставится в соответствие идентификатор – число в пределах пространства хеш-функции. Таким образом, каждый узел получает некоторое положение на кольце. Ключевым моментом здесь является то, что между идентификатором узла и ключом, получающемся в результате вычисления хеш-функции от нужного значения, нет никакой принципиальной разницы. Два соседних на кольце узла образуют область, за которую отвечает один из них (с большим или меньшим идентификатором). Каждый элемент данных идентифицируется ключом (путем вычисления хеш-функции) и получает некоторую позицию на кольце. Далее по кольцу ищется первый встретившийся по часовой (или против часовой) стрелке идентификатор узла. Так элемент данных связывается с определенным узлом (рис. 1).

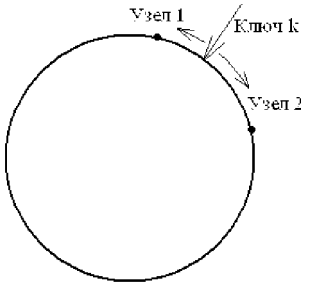

Рис. 1. Отображение элемента данных в узел кластера посредством ключа k

В нашем конкретном случае элемент данных – запрашиваемый домен.

Основным вопросом при реализации DHT является схема разбиения пространства ключей между участниками кластера.

Самый простой способ деления пространства – равномерно разбить его на количество узлов и каждому узлу назначить участок. Но данный подход имеет ряд недостатков: при выходе узла область его ответственности практически полностью перейдет только одному узлу, также и при входе, никаким образом не учитывается неравномерность распределения получаемых хеш-значений и т.д. Содержание кэша каждого узла полностью зависит от того, за какую часть пространства отвечает узел. Соответственно, если после изменения состава участников кластера пространство ответственности узла претерпит существенные изменения, то большая часть его кэша станет неактуальной, и при подавляющем большинстве запросов узел, поняв, что в кэше не содержится запрашиваемых данных, будет для разрешения домена обращаться к вышестоящим серверам, производить рекурсивные запросы, дожидаться ответов и т.д. Все это критическим образом скажется на производительности кластера.

Следовательно, если представлять проблему графически на рис. 2, для каждого узла необходимо добиться максимального пересечения его пространства до изменения, и его пространства после изменения состава участников кластера.

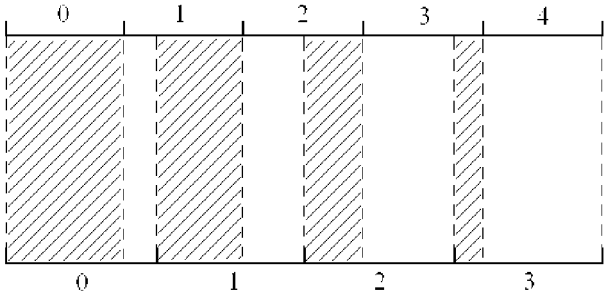

Рис. 2. Пересечение пространств узлов до и после изменения состава участников в линейном представлении (верхняя и нижняя горизонтальные линии – хеш-пространство, разделенное между узлами 0, 1, 2 и т.д.)

Ситуация выхода узла под номером 4 из кластера, состоящего из узлов 0, 1, 2 и т.д., на рис. 2 для удобства восприятия изображена в линейном представлении. Как видно, размеры пересечения пространств для каждого узла крайне неодинаковы. Наибольшее пересечение получилось для узла 0, большая часть его кэша останется актуальной, таким образом, изменение состава участников не сильно повлияет на скорость его работы. Но этого нельзя сказать про другие узлы. Высвободившийся участок вышедшего узла 4 полностью перешел только одному узлу под номером 3, причем этому узлу придется практически полностью обновить свой кэш, что существенно скажется на времени. Меньше всего обновлять свой кэш придется узлу 0, узлу 1 – уже больше и т.д. по

нарастающей. В итоге получается большая неравномерность в перераспределении освободившегося пространства. Следовательно, нужен более гибкий способ разделения пространства.

РЕАЛИЗАЦИЯ

После проведенных исследований и расчетов, была реализована следующая концепция распределения области значений хеш-функции. В данной системе используется хеш-функция, вычисляющая 64-битный ключ. Таким образом, мы имеем пространство ключей размером 2^{64}. Все пространство разбивается на 2^{32} одинаковых частей, которые будем называть чанками. В свою очередь каждый чанк в равных частях делится на количество «живых» узлов кластера.

Для того чтобы при выходе узла из состава кластера его участки не переходили одним и тем же соседним узлам, порядок следования отрезков ответственности узлов должен быть неодинаковым (для равномерности перераспределения запросов при выходе узла из системы). Но при этом, например, для 10 узлов существует 10! вариантов перестановок, что сложно для вычисления, хранения и поиска искомого узла, что в конечном итоге критично сказывается на скорости работы системы. Также это решение практически не масштабируется и плохо ведет себя при изменении состава кластера, т.к. в каждом чанке порядок следования отрезков ответственности узлов не должен меняться при выходе или входе узла, что очень сложно поддерживать при таком решении.

Поэтому решено было ввести 512 вариантов распределения узлов в чанке, повторяющихся циклично на всем пространстве, и хранить их в двумерном массиве. Варианты распределения вычисляются с помощью функции rand(). Причем для того, чтобы узлы не теряли своего взаимного положения (т.к. в любом случае приходится нормировать случайное значение на количество узлов в кластере), применяется следующая схема: на пространстве, превышающем максимальное предполагаемое количество узлов (например, 1024), размер которого должен обязательно оставаться неизменным, каждый узел через rand() получает свою позицию, исходя из своего IP и номера варианта распределения.

Таким образом, мы получаем «детерминированную случайность»: с одной стороны, позиции узлов получаются случайным образом, но с другой стороны, это случайное значение не меняется для данного чанка. Распределив узлы на промежуточном пространстве, «собираем» их в массив в порядке следования и получаем вариант распределения. Данная операция выполняется для каждого варианта (в существующей реализации – 8 раз). Получаем таблицу узлов, в соответствии с которой строится функция отображения ключей в узлы. При изменении состава участников кластера таблица обновляется.

Также, в данной системе реализована репликация хеша, т.е. взаимное перекрытие отрезков ответственности узлов, в соответствии с которым определяется не только узел, которому будет послан запрос на обработку, но и узел, чей отрезок имеет с ним зону перекрытия. При этом он только закэширует данные. Это сделано для того, чтобы при выходе узла из кластера перекрывающие его узлы уже частично содержали его кэш, что положительно скажется на скорости обработки данных. Причем области перекрытия задаются в процентном соотношении.

В ходе тестирования системы распределения нагрузки за основу были взяты реальные данные – 5000000 уникальных доменов из запросов, принятых за две недели работы системой DNS. Тестовые запуски проводились для различного количества узлов кластера: два, три, пять и десять узлов. В каждом из тестов к системе были обращены 5000000 запросов с собранными уникальными доменами.

На рис. 3 в виде графика представлены результаты проведенных тестов для 5-ти узлов в кластере, где эффективность распределения

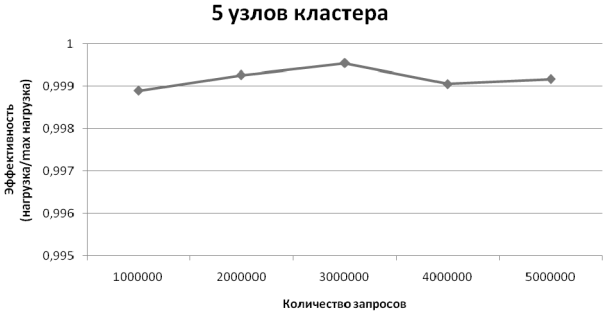

Рис. 3. Показатели эффективности для 5-ти узлов кластера

нагрузки определяется как отношение среднего числа запросов, обслуживаемых каждым узлом, к максимальному количеству запросов, принятых самым нагруженным узлом. По горизонтали обозначено различное количество запросов к системе с шагом 1000000 (1000000, 2000000, 3000000 и т.д.), по вертикали – эффективность в виде отношения средняя нагрузка/максимальная нагрузка.

Как видно из представленного графика каждый узел системы получает практически одинаковое количество запросов. Расчеты показывают, что в среднем отклонение составляет около 0.07% (для запросов с уникальными доменами).

ЗАКЛЮЧЕНИЕ

В заключение можно сказать, что в ходе проведенной работы была полностью реализована инфраструктура распределенной хеш-таблицы, обслуживающая кластер DNS.

Разработка данной системы производилась в несколько этапов:

- анализ проблемы и технического задания;
- изучение теоретического материала, касающегося распределенных хеш-таблиц и их реализации;
- рассмотрение существующих решений и реализаций, анализ возможности их частичного применения;
- разработка общей архитектуры системы;
- подбор хеш-функции нужной разрядности и исследование ее характеристик;
- непосредственная реализация системы: реализация функций инициализации и деинициализации системы, построения внутренних структур данных и их корректировки при ситуации изменения состава участников кластера, функции вычисления узлов для обработки и репликации, организация взаимодействия с внешними системами, обслуживающими кластер;
- тестирование и оценка эффективности.

В ходе тестирования были получены высокие показатели эффективности системы. Как было сказано выше, в среднем отклонение в балансировке нагрузки от абсолютной равномерности составило 0.07%, что для данной системы является вполне удовлетворительным. В целом, реализованная система DHT обладает высокими показателями распределения нагрузки, надежности, быстродействия, гибкости поведения при изменении состава участников кластера. Также реализован механизм репликации DNS записей.

В данный момент система успешно функционирует в составе DNS кластеров, расположенных в Европе и США.

СПИСОК ЛИТЕРАТУРЫ

1. *DeCandia G., Hastorun D., Jampani M., Kakulapati G., Lakshman A., Pilchin A., Sivasubramanian S., Vosshall P., Vogels W.* 2007. Dynamo: Amazon's Highly Available Key-value Store. In Proceedings of the 21st ACM Symposium on Operating Systems Principles (Skamania Lodge Stevenson, WA, USA, October 14–17, 2007). SOSP '07. ACM Press, New York, NY, 205–218.

2. *Fox A., Gribble S. D., Chawathe Y., Brewer E. A., Gauthier P.* 1997. Cluster-based scalable network services. In Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles (Saint Malo, France, October 05 – 08, 1997). W. M. Waite, Ed. SOSP '97. ACM Press, New York, NY, 78–91.

3. *Karger D., Lehman E., Leighton T., Panigrahy R., Levine M., Lewin D.* 1997. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing (El Paso, Texas, United States, May 04 – 06, 1997). STOC '97. ACM Press, New York, NY, 654–663.

4. Ali Ghodsi. Distributed k-ary System: Algorithms for Distributed Hash Tables. KTH-Royal Institute of Technology, 2006, 1–15, 23–40.

5. *Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph and John D. Kubiatowicz* 2004. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal On Selected Areas In Communication, Vol. 22, No. 1, January 2004, 41–48.

6. *Шилов С.Н., Крыловецкий А.А.* XI международная конференция «Информатика», том 2, 2011, 448–454.

Шилов С. Н. – аспирант кафедры цифровых технологий Воронежского государственного университета. E-mail: shilov_sn88@mail.ru

Shilov S. - Postgraduate student of Department of Digital Technologies, Voronezh State University, e-mail: shilov_sn88@mail.ru

Кургалин Сергей Дмитриевич – доктор физико-математических наук, заведующий кафедрой цифровых технологий факультета компьютерных наук Воронежского государственного университета. Тел.: (473) 2-208-384. E-mail: kurgalin@bk.ru

Крыловецкий Александр Абрамович – кандидат физико-математических наук, доцент кафедры цифровых технологий Воронежского государственного университета. E-mail: aakryl@cs.vsu.ru

Kurgalin Sergey Dmitrievich – Doctor of Physical and Mathematical Science, Head of the Digital Technologies Department of Computer Science Faculty of Voronezh State University. Tel.: (473) 2-208-384. E-mail: kurgalin@bk.ru

Krylovetsky A. A. – Ph.D. in Physics and Mathematics, Associate Professor of Department of Digital Technologies, Voronezh State University. E-mail: aakryl@cs.vsu.ru