МЕТРИКИ ДЛЯ ДИНАМИЧЕСКОГО МАСШТАБИРОВАНИЯ БАЗ ДАННЫХ В ОБЛАЧНЫХ СРЕДАХ*

УДК 004.4

Александр Викторович Бойченко,
к.т.н., начальник УНИР, Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-60-11
E-mail: ABoichenko@mesi.ru

Дмитрий Константинович Рогожин,
аспирант, инженер-программист научно-образовательного центра УНИР, Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-82-33
E-mail: DRogojin@mesi.ru

Дмитрий Геннадьевич Корнеев,
к.э.н., начальник отдела фандрайзинга УНИР, Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-60-11
E-mail: DKorneev@mesi.ru

В статье проводится анализ основных методов масштабирования баз данных (репликация, шардинг) и их поддержки на уровне популярных реляционных СУБД и NoSQL решений с различными моделями данных: документо-ориентированной, ключ-значение, поколоночной, графовой. Приводится оценка возможности современных облачных решений и модель для организации динамического масштабирования в облачной инфраструктуре. Выделяются группы базовых метрик, характеризующих параметры функционирования СУБД и технических средств, а также определяются цели и группы интегральных метрик, необходимых для реализации адаптивных алгоритмов динамического масштабирования баз данных в облачной инфраструктуре.

Ключевые слова: динамическое масштабирование, облачные вычисления, NoSQL, шардинг, репликация, метрики функционирования БД.

Alexander V. Boichenko,
PhD, Head of the Research Department, Moscow State University of Economics, Statistics and Informatics (MESI)
Тел.: (495) 442-60-11,
E-mail: ABoichenko@mesi.ru

Dmitry K. Rogojin,
Post-graduate student, Software developer, Moscow State University of Economics, Statistics and Informatics (MESI)
Тел.: (495) 442-82-33,
E-mail: DRogojin@mesi.ru

Dmitry G. Korneev,
PhD, Head of Fundraising Division, Moscow State University of Economics, Statistics and Informatics (MESI)
Тел.: (495) 442-60-11,
E-mail: DKorneev@mesi.ru

METRICS FOR DYNAMIC SCALING OF DATABASE IN CLOUDS

This article analyzes the main methods of scaling databases (replication, sharding) and their support at the popular relational databases and NoSQL solutions with different data models: a document-oriented, key-value, column-oriented, graph. The article provides an assessment of the capabilities of modern cloud-based solution and gives a model for the organization of dynamic scaling in the cloud infrastructure. In the article are analyzed different types of metrics and are included the basic metrics that characterize the functioning parameters and database technology, as well as sets the goals of the integral metrics, necessary for the implementation of adaptive algorithms for dynamic scaling databases in the cloud infrastructure. This article was prepared with the support of RFBR grant № 13-07-00749.

Keywords: dynamic scaling, cloud computing, NoSQL, sharding, replication , database metrics.

1. Введение

В условиях постоянно изменяющейся нагрузки, характерной, в частности для решений *SaaS* и *DBaaS*, способность сервиса к быстрому масштабированию в облачной инфраструктуре является главным преимуществом перед классическим подходом доставки сервисов. Эта способность позволяет повысить отказоустойчивость, уровень доступности и производительность сервиса путем перехода от статического выделения ресурсов сервису к динамическому выделению по требованию в соответствии с текущей нагрузкой. [1] Рост интереса к горизонтальному масштабированию баз данных привел к появлению нового течения в хранении и обработке данных – *NoSQL* решениям, большинство из которых изначально проектировались с поддержкой устойчивости к разделению данных по сети. Классические реляционные СУБД: *Oracle*, *MySQL*, *MS SQL* Server также стремятся к поддержке устойчивости к разделению по сети в специальных «кластерных» версиях своих продуктов, жертвуя полноценными *ACID* транзакциями в пользу eventual consistency – «согласованность в конечном итоге» [4].

В условиях предоставления инфраструктуры для программных сред по требованию, актуальными стали вопросы возможности динамического масштабирования баз данных. Под динамическим масштабированием подразумевается свойство системы программно регулировать ресурсы, не требуя оперативного вмешательства пользователя. Динамическое масштабирование позволяет снизить время реакции на инциденты, увеличить уровень доступности сервиса и, тем самым, повысить качество предоставляемых сервисов. Возможности динамического масштабирования серверов на основе набора триггеров и метрик предоставляют своим клиентам такие компании, как *Amazon* (на основе решений *CloudWatch* и *ElasticCloud*), *Rackspace* (на основе решений *Rackspace Cloud Monitoring* и *Otter*). Создаваемая система *TIRAMOLA* [10] – совместный труд греческих университетов *National Technical University of Athens* и *Ionian University* – позволяет динамически масштабировать некоторые *NoSQL* СУБД с поколоночной моделью данных. Однако, общие методики динамического масштабирования БД с различными моделями данных, основанные на комплексном анализе метрик функционирования БД, в настоящий момент проработаны недостаточно. В настоящее время для масштабирования обычно используются метрики, основанные только на показателях функционирования технических устройств. Таким образом, разработка интегральных метрик, одновременно учитывающих параметры функционирования БД и технических средств, для возможности динамически выполнять масштабирование СУБД в облачной инфраструктуре является актуальной темой исследований.

2. Анализ методов масштабирования БД

Долгое время вертикальное масштабирование было основным методом повышения производительности баз данных. Но, зачастую, вертикальное масштабирование не решает возникших в БД проблем и становится неоптимальным методом, поэтому сегодня для повышения производительности баз данных все чаще используют горизонтальное масштабирование. В на-

* Статья подготовлена при поддержке гранта РФФИ № 13-07-00749.

стоящее время среди существующих методов горизонтального масштабирования, имеющих поддержку в реляционных БД и *NoSQL* решениях, принято выделять репликацию, секционирование (партицирование) и шардинг. [2, 3]

Репликация – это процесс, под которым понимается копирование данных из одного источника на множество других и наоборот. Существует около 10 видов репликации серверов баз данных: мульти-мастер (*multi-master*, *master-master*), мастер-подчинённый (*master-slave*), мульти-источник (*multy-source*), мастер-подчинённый-мастер (*master-slave-master*), мульти-сервер (*multi-server parallel query execution*), сектор-подчинённый (*mirror data partitioning*) и др. [5] Несмотря на то, что репликация теоретически является бесконечно наращиваемым решением, она способна решить только проблемы чтения данных из БД. Увеличение числа серверов становится нецелесообразным, когда появляются проблемы с записью данных в БД. [3]

Секционирование (*партицирование*) – средство масштабирования многих современных реляционных баз данных, которое позволяет разбивать таблицы, индексы и индекс-таблицы на части, таким образом, обеспечивая контроль и доступ к данным объектам базы данных на более низком уровне. Каждая из этих частей объекта базы данных называется секцией (или подсекцией для составных секционированных объектов. Таблицы секционируются с использованием «ключа секционирования», набора столбцов, определяющих, в какой секции будет располагаться заданная запись. Более подробно механизмы реализации секционирования описаны в [6, 7]. Существенным минусом механизма секционирования применительно к горизонтальному масштабированию уровня БД является то, что секционирование не выходит за рамки одного сервера. То есть, полученные в результате секции объектов физически располагаются на том же сервере.

Ш*ардинг* – разделение данных на уровне ресурсов – процесс, по смыслу схожий с секционированием, однако полученные в результате разделения объекты разносятся по разным серверам БД. Принято выделять *вертикальный шардинг*, при котором таблица(коллекция) выносится на другой сервер целиком, и *горизонтальный шардинг*, при котором на разные сервера выносятся части одной таблицы (коллекции). Подобно процессу секционирования, при шардинге выбирается «ключ шардирования», определяющий, на каком сервере будут располагаться части данных. Логика поиска сервера, на котором располагаются необходимые данные, заимствована из алгоритма разбиения пространства ключей DHT (англ. *Distributed Hash Table* – «распределённая хеш-таблица»). [8] В основе этого способа разбиения лежит функция $\delta(k_1, k_2)$, определяющая абстрактное понятие расстояния между ключами k_1 и k_2. Каждому узлу присваивается единичный ключ, называемый его идентификатором (*ID*). Узел с *ID* i_n владеет всеми ключами k_m, для которых i_n – самый ближайший *ID*, вычисленный с помощью $\delta(k_m, i_n)$. [9] Наиболее удобным является механизм так называемого авто-шардинга (*auto-sharding*), при котором реализация логики процесса шардинга осуществляется средствами самой СУБД. В таком случае нет необходимости реализовывать ее самостоятельно в рамках программного кода приложения или применять сторонние решения в качестве дополнительного промежуточного слоя. Помимо оптимизации операций чтения – читаются данные из более узкого набора, а не из всей таблицы (коллекции) целиком, шардинг решает также проблемы записи в БД.

Выбор того или иного средства для обеспечения масштабирования

Таблица 1.

Реализации репликации и шардинга в современных СУБД.

База данных	Режимы репликация	Механизм шардинга
Документо-ориентированные		
MongoDB	Мастер-подчиненный	*Auto-sharding* (на основе “осколков” (*Shards*) и “кусков” (*Chunks*)) [12]
CouchDB	Мульти-мастер	Встроенной поддержки нет. Осуществляется сторонними решениями [11]: *CouchDb-Lounge*, *BigCouch*, *Gizzard*
Ключ-значение		
Riak	Мульти-мастер	*Auto-sharding* (на основе *Ring Partitions*)
Redis	Мастер-подчиненный	Встроенной поддержки нет. Осуществляется сторонними решениями [15]: *Redis Cluster*, *Twemproxy*, *Predis*
Поколоночные (Столбцовые)		
HBase	Мастер-подчиненный	*Auto-sharding* (на основе Регионов (*Regions*))
Cassandra	Мульти-мастер	*Auto-sharding* (на основе *Ring Partitions*) [8]
Графовые		
Neo4j	Мастер-подчиненный	Полноценной поддержки нет. Частично реализован в технологии *Cache Sharding* [18]
OrientDB	Мульти-мастер	Нет
Реляционные		
Oracle Database	Мульти-мастер, Мастер-подчиненный	Только для *Oracle Real Application Clusters* [16]
MySQL	Мульти-мастер, Мастер-подчиненный, «Круговая» репликация	*Auto-Sharding* (только в версии *MySQL Cluster*) [19]
MS SQL Server	Мастер-подчиненный, *Multi-source*	*Data-Dependent Routing* (для *MS SQL Server*), *Federation* (для *SQL Azure*) [13]
PostgreSQL	Мульти-мастер, Мастер-подчиненный	Встроенной поддержки нет. Осуществляется сторонними решениями [14]: *PL/Proxy*, *HadoopDB*, *PgBouncer*.

уровня БД зависит в первую очередь от комплексного анализа состояния БД и технических средств и выявления возможной точки отказа. Зачастую, при очень высокой нагрузке, необходимо применить вышеописанные методы масштабирования в комплексе. Выбор какого-то конкретного метода зависит не только от поставленной задачи, но и от поддержки метода на уровне самой СУБД. В таблице 1 приведены сведения о поддержке и реализации репликации и шардинга в современных СУБД с различными моделями данных: документо-ориентированной, ключ-значение, поколоночной, графовой и реляционной. Поскольку механизмы секционирования не имеют прямого отношения к организации масштабируемого кластера в облачной инфраструктуре, в данной таблице и далее они не рассматриваются.

Как видно из таблицы, большинство *NoSQL* решений (кроме графовых БД) обладают встроенными механизмами автоматического шардинга в базовой поставке, в отличие от реляционных БД, где шардинг доступен только в специальных версиях. Такое ограничение в реляционных БД волне логично – проектировщику БД предоставляется выбор: что важнее – полноценная поддержка транзакций в нераспределенной среде или поддержка устойчивости к разделению по сети с согласованностью в конечном итоге (ограничения теоремы *CAP* [17]). Для графовых БД отсутствие шардинга объясняется теми же ограничениями, поскольку большинство СУБД, основанных на графовых моделях, имеют полноценную поддержку *ACID* транзакций подобно традиционным реляционным базам данных. [18] В свою очередь, реляционные СУБД имеют более широкий диапазон выбора методов репликации, апробированных в процессе многолетнего использования реляционных БД.

3. Масштабирование в облачной инфраструктуре

Динамическое масштабирование в облачной инфраструктуре возможно благодаря использованию технологии виртуализации и предоставляемому облачной платформой открытому *API*. Виртуализация позволяет выделять ресурсы сервису по требованию в виде порций – виртуальных машин, а открытое *API* позволяет это делать программными средствами. Большое распространение получила библиотека *euca2ools* [20], совместимая с публичным облаком *Amazon AWS* и приватными облаками на базе *Eucalyptus* и *OpenStack*. Используя *API* облачной платформы напрямую или через *euca2ools*, клиенты могут производить запуск и остановку виртуальных машин в облаке, основываясь на собственных алгоритмах выделения ресурсов по требованию. Для реализации этих возможностей проектируемая система динамического масштабирования должна содержать как минимум систему мониторинга и систему управления масштабированием, содержащую подсистемы управления виртуальными машинами и кластером СУБД (рисунок 1).

Большинство реализованных систем динамического масштабирования используют для организации процесса масштабирования метрики, основанные на базовых показателях жизнедеятельности сервера, на котором располагается база данных: информацию о процессоре, оперативной памяти, жестком диске и сети. В частности, система динамического масштабирования кластера поколоночных (столбцовых) СУБД *TIRAMOLA* [10] – совместная разработка греческих университетов *National Technical University of Athens* и *Ionian University* – использует в качестве основной метрики текущую загрузку процессора (*CPU Usage*). Подсистема динамического масштабирования системы *TIRAMOLA* реализует следующий алгоритм. В случае, когда хотя бы для одного из серверов в кластере верно условие *CPU Usage* > 40%, срабатывает триггер добавления виртуальных машин в кластер. В случае, когда для всех серверов кластера справедливо условие *CPU Usage* < 15%, срабатывает триггер высвобождения виртуальных машин из кластера. В [10] приводятся положительные результаты работы такого алгоритма для СУБД *HBase*, *Cassandra* и *Riak*, имеющих схожую реализацию механизмов авто-шардинга на основе *Ring Partitions*. Однако, на наш взгляд, базовых метрик жизнедеятельности серверов недостаточно для комплексного подхода к масштабированию БД с различными моделями данных. В частности, имея в распоряжении эти только эти метрики, невозможно точно выяснить, какой из методов масштабирования – репликацию или шардинг – необходимо выбрать для увеличения производительности кластера СУБД. Невозможно также определить «кандидата на шардинг» – таблицу или коллекцию данных, требующую разделения по сети, и ключ шардинга. Для адаптивных алгоритмов динамического масштабирования требуются более жесткие требования к системе мониторинга, соответственно, более широкий набор метрик функционирования серверов СУБД.

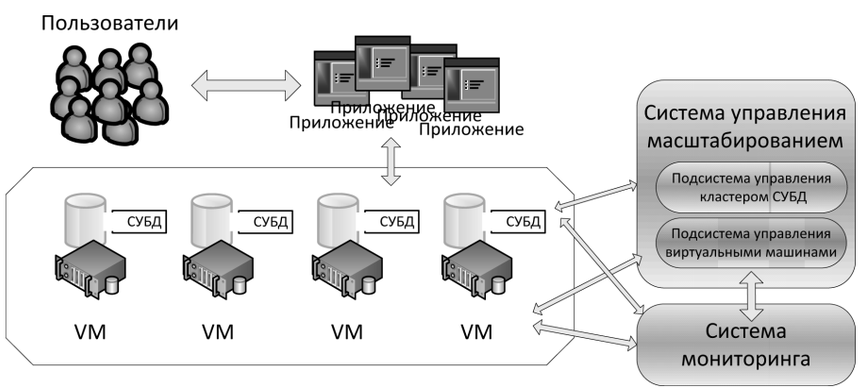

Рис. 1. Модель архитектуры системы динамического масштабирования

4. Метрики, используемые для масштабирования

Современные системы мониторинга обладают широкими возможностями для наблюдения за показателями СУБД. Многие из них предоставляют модули для мониторинга популярных БД в базовой комплектации или, в противном случае, расширение функционала через сторонние плагины. В частности, плагин *mikoomi* [22] для популярной системы мониторинга *Zabbix* поддерживает около 80 метрик для документо-ориентированной СУБД *MongoDB*. Такое большое количество показателей связано, в первую очередь, со сложной архитектурой любой из современных СУБД. Примерно такое

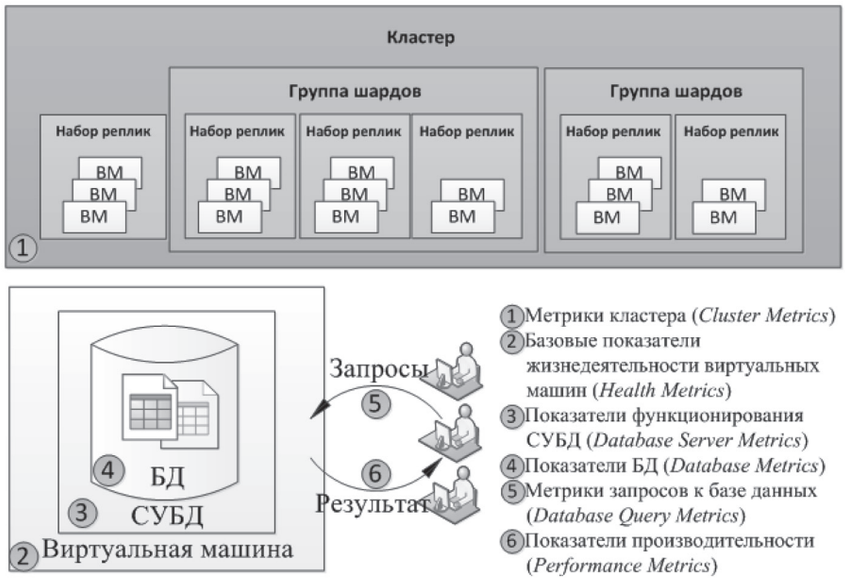

Рис. 2. Группы базовых метрик в разрезе масштабируемого кластера СУБД

же число метрик приведено в [21] для нескольких популярных реляционных СУБД.

В контексте статьи метрики, значения которых предоставляются непосредственно системой мониторинга, будем называть базовыми. Поскольку кластер БД, как объект мониторинга, является комплексным, получаемые метрики целесообразно разделить на логические группы. Основываясь на архитектуре работы кластера СУБД и взаимодействии с ним пользователей, можно выделить следующие группы базовых метрик, имеющие отношение к соответствующим уровням архитектуры (рисунок 2). В зависимости от используемой в БД модели данных термины исследуемых объектов несколько отличается друг от друга, поэтому в качестве обобщения далее в статье будут использованы термины реляционной модели данных.

1. Метрики кластера *(Cluster Metrics)*. К метрикам кластера предлагается относить показатели, характеризующие ресурсы кластера и логических групп внутри кластера, в том числе:

− **Общие метрики кластера**: количество виртуальных машин в кластере и объем выделенных кластеру в настоящий момент ресурсов (количество виртуальных процессоров, объем оперативной памяти, количество и объем дисковых разделов);

− **Метрики групп репликации**: количество групп репликации, количество серверов в каждой группе репликации, потребляемый группой репликации объем ресурсов (количество виртуальных процессоров, объем оперативной памяти, количество и объем дисковых разделов);

− **Метрики групп шардинга**: количество групп шардинга, количество серверов в каждой группе шардинга, потребляемый группой шардинга объем ресурсов (количество виртуальных процессоров, объем оперативной памяти, количество и объем дисковых разделов).

2. Базовые показатели жизнедеятельности виртуальных машин *(Health Metrics)*. Под базовыми показателями жизнедеятельности виртуальных машин предлагается понимать моментальные абсолютные и относительные метрики ресурсов конкретной виртуальной машины кластера, в том числе:

− **Общие метрики виртуальной машины**: доменное имя, количество выделенных виртуальных процессоров, объем выделенной оперативной памяти, количество и объем разделов жестких дисков;

− **Метрики утилизации *(Usage)*** виртуальных процессоров, оперативной памяти, виртуальной памяти (*swap*), сетевого траффика, системы ввода-вывода и пространства жестких дисков;

− **Метрика статуса процесса СУБД**, характеризующая состояние выполнения процесса: «запущен» или «остановлен».

3. Показатели функционирования СУБД *(Database Server Metrics)*. К этим показателям предлагается относить метрики, характеризующие состояние системы управления базами данных, в том числе:

− **Общие метрики СУБД**: тип СУБД (реляционная, документная, поколоночная, ключ-значение или графовая), название и версия СУБД, пути к каталогам с данными и логам, флаг «только для чтения»;

− **Метрики обслуживаемых баз данных**: список баз данных, физический размер всех баз данных, общее количество объектов во всех базах данных, общее количество таблиц во всех базах данных;

− **Метрики репликации**: флаг принадлежности сервера СУБД к группе репликации, Идентификатор группы репликации, список имен всех серверов данной группы репликации, количество серверов данной группы репликации;

− **Метрики шардинга**: флаг принадлежности сервера СУБД к группе шардинга, идентификатор группы репликации, список шардов в группе шардинга, общее число шардов в группе шардинга, список шардированных таблиц во всем кластере СУБД, общее число шардированных таблиц в кластере СУБД.

4. **Показатели базы данных *(Database Metric)***. К этим показателям предлагается относить метрики, характеризующие состояние конкретной базы данных и её объектов: таблиц, индексов и др., в том числе:

− **Общие метрики БД:** количество таблиц, список таблиц, количество индексов, список индексов, количество экстентов, количество объектов, средний размер объекта, объем данных в базе, объем данных на диске;

− **Метрики таблиц и индексов:** количество записей в таблице, объем записей в таблице, список полей таблицы, первичный ключ, список внешних ключей, количество внешних ключей, объем и тип индекса.

5. Метрики запросов *(Query Metrics)*. Современные СУБД предоставляют средства профилирования и логирования запросов к базе данных. На основе этой информации для каждого выполняемого запроса можно определить:

− **Общие метрики запросов:** общее число запросов за период времени, количество операций выборки, встав-

ки, изменения и удаления за период времени;

− **Метрики запроса:** тип запроса (*select, insert, update, delete*, изменение схемы), список затрагиваемых таблиц, количество измененных строк, флаг «медленный запрос», используемый индекс.

6. Метрики производительности *(Performance Metrics).* Метрики производительности являются наиболее важными при оценке качества работы кластера БД. К ним можно отнести:

− **Метрики подключений**: количество доступных подключений, количество активных в настоящий момент подключений, количество операций login/logout в единицы времени;

− **Метрики пропускной способности *(Throughput)*:** объем входящего/исходящего траффика СУБД за период времени;

− **Метрики журналирования:** количество и объем операций журналирования за единицу времени;

− **Метрики эффективности индексов:** общее число обращений к индексам за единицу времени, число промахов (Miss) при попытке использовать индекс за единицу времени, общее число сбросов индексов;

− **Метрики отклика *(Response):*** время выполнения операций каждого типа (*select, insert, update, delete*, изменение схемы), время выполнения каждого из запросов в отдельности.

5. События, учитываемые при масштабировании

Помимо базовых метрик, система мониторинга должна фиксировать происходящие в кластере события, связанные с изменением состояния кластера СУБД:

− **События кластера:** создана виртуальная машина, удалена виртуальная машина, создана группа шардинга, удалена группа шардинга, создана группа репликации, удалена группа репликации;

− **События виртуальных машин:** виртуальная машина не отвечает;

− **События СУБД:** добавлен новый узел группы репликации, удален узел группы репликации, добавлен новый узел группы шардинга, удален узел группы шардинга, создана база данных, удалена база данных;

− **События БД:** создана таблица, удалена таблица, создан индекс, удален индекс, изменена структура таблицы.

6. Назначение интегральных метрик

В контексте статьи под интегральными метриками понимаются метрики, значение которых вычисляется на основе полученных с помощью системы мониторинга базовых метрик всех описанных выше групп. Целями интегральных метрик являются:

− Отслеживание полного отказа серверов;

− Оценка увеличения или снижения нагрузки на кластер БД и/или его участки - группы репликации и группы шардинга;

− Оценка снижения производительности кластера БД и/или его участков – групп репликации и групп шардинга;

− Определение узких мест кластера, повлиявших на снижение производительности;

− Определение целесообразности применения одного из методов масштабирования:

  o Определение участка, нуждающегося в вертикальном масштабировании;

  o Определение участка, нуждающегося дополнительной реплике или уменьшении числа существующих реплик;

  o Определение участка, нуждающегося в дополнительном шарде или уменьшении числа существующих шардов;

  o Определение оптимального ключа шардинга в случае необходимости шардинга;

− Оценка повышения производительности кластера за счет управляющего воздействия системы динамического масштабирования;

− Прогнозирование изменений в работе кластера исходя из полученных ранее данных.

Таким образом, интегральные метрики должны наиболее полно отражать состояние кластера БД и происходящие в нем процессы, в том числе, в динамике. Для достижения этих целей в настоящий момент предлагается выделение интегральных метрик, отражающих:

− Средние значения базовых показателей жизнедеятельности кластера БД и входящих в его состав групп шардинга и репликации;

− Динамику изменения средних значений базовых показателей жизнедеятельности кластера БД и входящих в его состав групп шардинга и репликации за период времени;

− Динамику изменения производительности каждой конкретной виртуальной машины кластера БД;

− Динамику абсолютного и относительного приращения объемов базы данных;

− Динамику вертикального и горизонтального, абсолютного и относительного приращения объемов объектов базы данных;

− Динамику интенсивности обращений к базе данных;

− Динамику интенсивности конкретных запросов к базе данных;

− Динамику показателей производительности кластера СУБД.

7. Направления для продолжения исследований

В настоящей статье были проанализированы основные методы масштабирования баз данных и их поддержка на уровне популярных реляционных СУБД и *NoSQL* решений с различными моделями данных: документо-ориентированной, ключ-значение, поколоночной, графовой. Оценены возможности современных облачных решений и приведена модель организации динамического масштабирования. Выделены группы базовых метрик, характеризующих параметры функционирования БД и технических средств, а также определены цели и группы интегральных метрик, необходимых для реализации адаптивных алгоритмов динамического масштабирования баз данных.

Для продолжения исследований алгоритмов динамического разделения данных на уровне ресурсов (шардинга) и репликации для организации автоматического масштабирования мультитенантной базы данных в облачной инфраструктуре предполагается выделить следующие направления:

− Детализация и формализация интегральных метрик функционирования кластера мультитенантных баз данных в облачной инфраструктуре;

− Оценка возможности использования указанных показателей в алгоритмах для организации динамического масштабирования мультитенантных баз данных в облачной инфраструктуре.

Литературы

1. Idziorek J. Discrete event simulation model for analysis of horizontal scaling in the cloud computing model /Proceedings of the 2010 Winter Simulation Conference.

URL: http://www.informs-sim.org/wsc10papers/278.pdf (дата обращения: 07.11.2013).

2. Рогожин Д. Средства масштабирования в облачной инфраструктуре / III научно-практическая конференция молодых ученых “Инновационное развитие российской экономики”. 10 декабря 2012г. / Сборник научных трудов. – М.: МЭСИ, 2012. – Предм. указ.: с. 140 – 148.

3. Дэн Голотюк. Шардинг, партиционирование, репликация - зачем и когда? / URL: http://highload.com.ua/index.php/2009/05/06/шардинг-партиционирование-репликац/ (дата обращения: 07.11.2013)

4. Burckhardt S., Leijen D. Eventually Consistent Transactions/ URL: http://research.microsoft.com/pubs/158085/ecr-esop2012.pdf / (дата обращения: 07.11.2013)

5. Косьмина Я.О. Репликация. Master-slave, кластер и прочее. / URL: http://freehabr.ru/blog/database/1119.html (дата обращения: 07.11.2013).

6. Oracle® Database VLDB and Partitioning Guide. Part2: Partitioning Concepts / URL: http://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm (дата обращения: 07.11.2013)

7. SQL Server 2012. Create Partitioned Tables and Indexes/ URL: http://technet.microsoft.com/en-us/library/ms188730.aspx (дата обращения: 07.11.2013)

8. Ломакин А. Apache Cassandra, part 1 – principles, data model. Презентация Exigen Services 28 июня 2011. / URL: http://www.slideshare.net/lomakin.andrey/apache-cassandra-part-1-principles-data-model (дата обращения: 07.11.2013)

9. Распределённая хеш-таблица. Материал из Википедии — свободной энциклопедии / URL: http://ru.wikipedia.org/wiki/Распределённая_хеш-таблица (дата обращения: 07.11.2013)

10. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM ‘11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385-2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (дата обращения: 07.11.2013)

11. CouchDB. The Definitive Guide. Clustering. / URL: http://guide.couchdb.org/draft/clustering.html (дата обращения: 07.11.2013)

12. Sharding and MongoDB. MongoDB Documentation Project. November 01, 2013/ URL: http://docs.mongodb.org/v2.4/MongoDB-sharding-guide.pdf (дата обращения: 07.11.2013)

13. MSDN. Scaling Out SQL Server. / Microsoft Corporation, April 2012. / URL: http://msdn.microsoft.com/en-us/library/aa479364.aspx (дата обращения: 07.11.2013)

14. Васильев А.Ю. Работа с Postgresql: настройка, масштабирование, - Справочное пособие, 2010. / URL: http://postgresql.ru.net/pgtune/postgresql.html#SECTION00600000000000000000 (дата обращения: 07.11.2013) / Глава 5. Шардинг.

15. Partitioning: how to split data among multiple Redis instances./ URL: http://redis.io/topics/partitioning (дата обращения: 07.11.2013)

16. Oracle Real Application Clusters (RAC). An Oracle White Paper, June 2013 / URL: http://www.oracle.com/technetwork/products/clustering/rac-wp-12c-1896129.pdf?ssSourceSiteId=ocomen (дата обращения: 07.11.2013)

17. Seth Gilbert, Nancy Lynch. Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. - Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. URL: http://people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf (дата обращения: 07.11.2013)

18. How To Use Cache Sharding: Scale Out Neo4j / FromDev: A Technology Blog About Programming, Web Development, Tips, Tutorials and Books Recommendation for Developers / 14.10.2013 / URL: http://www.fromdev.com/2013/10/neo4j-cache-sharding-scale-out.html (дата обращения: 07.11.2013)

19. Guide to Scaling Web Databases with MySQL Cluster. Accelerating Innovation on the Web and in the Cloud, A MySQL® White Paper, June 2013 / URL: http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/ (дата обращения: 07.11.2013)

20. Euca2ools. Сайт проекта Eucalyptus / URL: http://www.eucalyptus.com/download/euca2ools (дата обращения: 07.11.2013)

21. Oracle® Enterprise Manager. System Monitoring Plug-in Metric Reference Manual for Non-Oracle Database Management, Release 12 (12.0), B28748-11, November 2010. / URL: http://docs.oracle.com/cd/E11857_01/em.111/b28748.pdf (дата обращения: 07.11.2013)

22. Mikoomi. Developing useful monitoring plugins for Zabbix. Chapter 3: MongoDB Plugin. May 5, 2011. / URL: https://code.google.com/p/mikoomi/wiki/03 (дата обращения: 07.11.2013)

References

1. Idziorek J. Discrete event simulation model for analysis of horizontal scaling in the cloud computing model / Proceedings of the 2010 Winter Simulation Conference. URL: http://www.informs-sim.org/wsc10papers/278.pdf (date of request: 07.11.2013).

2. Rogogin D. Solution of database scaling in cloud infrastructure / III Scientific and Practical Conference of Young Scientists «Innovative development of Russian economy.» 10 December 2012. / Moscow: MESI, 2012. - p. 140 - 148.

3. Den Golotyuk. Sharding, Partitioning, Replication – Why and When? / URL: http://highload.com.ua/index.php/2009/05/06/ Sharding-Partitioning-Replication / (date of request: 07.11.2013)

4. Burckhardt S., Leijen D. Eventually Consistent Transactions/ URL: http://research.microsoft.com/pubs/158085/ecr-esop2012.pdf / (date of request: 07.11.2013)

5. Kosmina Y. Replication. Master-slave, cluster and so on. / URL: http://freehabr.ru/blog/database/1119.html (date of request: 07.11.2013).

6. Oracle® Database VLDB and Partitioning Guide. Part2: Partitioning Concepts / URL: http://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm (date of request: 07.11.2013)

7. SQL Server 2012. Create Partitioned Tables and Indexes/ URL: http://technet.microsoft.com/en-us/library/ms188730.aspx (date of request: 07.11.2013)

8. Lomakin A. Apache Cassandra, part 1 – principles, data model. Presentation Exigen Services 28/06/2011. / URL: http://www.slideshare.net/lomakin.andrey/apache-cassandra-part-1-principles-data-model (date of request: 07.11.2013)

9. Distributed hash table. Wikipedia / URL: http://ru.wikipedia.org/wiki/Distributedhashtable. (date of request: 07.11.2013)

10. Tsoumakos D., Konstantinou I., Boumpouka C. On the Elasticity of NoSQL Databases over Cloud Management Platforms. / Proceeding CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 2385-2388, ISBN: 978-1-4503-0717-8, / URL: http://www.cslab.ntua.gr/~ikons/elastic_nosql.pdf (date of request: 07.11.2013)

11. CouchDB. The Definitive Guide. Clustering. / URL: http://guide.couchdb.org/draft/clustering.html (date of request: 07.11.2013)

12. Sharding and MongoDB. MongoDB Documentation Project. November 01, 2013/ URL: http://docs.mongodb.org/v2.4/MongoDB-sharding-guide.pdf (date of request: 07.11.2013)

13. MSDN. Scaling Out SQL Server. / Microsoft Corporation, April 2012. / URL: http://msdn.microsoft.com/en-us/library/aa479364.aspx (date of request: 07.11.2013)

14. Vasiliev A.U. Working in Postgresql: adjustment, scaling/ Reference manual, 2010. / URL: http://postgresql.ru.net/pgtune/postgresql.html#SECTION00600000000000000000 (date of request: 07.11.2013) / Part 5. Sharding.

15. Partitioning: how to split data among multiple Redis instances./ URL: http://redis.io/topics/partitioning (date request: 07.11.2013)

16. Oracle Real Application Clusters (RAC). An Oracle White Paper, June 2013 / URL: http://www.oracle.com/technetwork/products/clustering/rac-wp-12c-1896129.pdf?ssSourceSiteId=ocomen (date of request: 07.11.2013)

17. Seth Gilbert, Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. - Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. URL: http://people.csail.mit.edu/sethg/pubs/BrewersConjecture-SigAct.pdf (date of request: 07.11.2013)

18. How To Use Cache Sharding: Scale Out Neo4j / FromDev: A Technology Blog About Programming, Web Development, Tips, Tutorials and Books Recommendation for Developers / 14.10.2013 / URL: http://www.fromdev.com/2013/10/neo4j-cache-sharding-scale-out.html (date of request: 07.11.2013)

19. Guide to Scaling Web Databases with MySQL Cluster. Accelerating Innovation on the Web and in the Cloud, A MySQL® White Paper, June 2013 / URL: http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/ (date of request: 07.11.2013)

20. Euca2ools. Сайт проекта Eucalyptus / URL: http://www.eucalyptus.com/download/euca2ools (date of request: 07.11.2013)

21. Oracle® Enterprise Manager. System Monitoring Plug-in Metric Reference Manual for Non-Oracle Database Management, Release 12 (12.0), B28748-11, November 2010. / URL: http://docs.oracle.com/cd/E11857_01/em.111/b28748.pdf (date of request: 07.11.2013)

22. Mikoomi. Developing useful monitoring plugins for Zabbix. Chapter 3: MongoDB Plugin. May 5, 2011. / URL: https://code.google.com/p/mikoomi/wiki/03 (date of request: 07.11.2013)