**ИНСТРУКЦИЯ**

**ПО УСТАНОВКЕ RT.KEYVALUE**

2024

ОГЛАВЛЕНИЕ

[1 Назначение системы и условия применения 3](#_Toc179474023)

[1.1 Назначение RT.KeyValue 3](#_Toc179474024)

[1.2 Условия применения 3](#_Toc179474025)

[1.2.1 Центральный процессор 3](#_Toc179474026)

[1.2.2 Объем оперативной памяти 3](#_Toc179474027)

[1.2.3 Диски 4](#_Toc179474028)

[1.2.4 Общие варианты облачных вычислений (Common Cloud Choices) 5](#_Toc179474029)

[2 Установка RT.KEYVALUE 6](#_Toc179474030)

[2.1 Предустановки 6](#_Toc179474031)

[2.2 Способ установки 6](#_Toc179474032)

[2.2.1 Установка RPM пакетов 6](#_Toc179474033)

[3 КОНФИГУРИРОВАНИЕ 8](#_Toc179474034)

[3.1 Основные свойства среды выполнения 8](#_Toc179474035)

[3.2 Изменение расположения каталогов 9](#_Toc179474036)

[3.3 Переменные среды 9](#_Toc179474037)

[3.4 Логирование 9](#_Toc179474038)

Назначение системы и условия применения

Назначение RT.KeyValue

RT.KeyValue – это NoSQL распределенная база, обладающая высокой масштабируемостью и доступностью без ущерба для производительности. Линейная масштабируемость и проверенная отказоустойчивость на стандартном оборудовании или облачной инфраструктуре делают её идеальной платформой для критически важных данных.

Документ содержит описание, как развернуть продукт и как его сконфигурировать.

Условия применения

До начала работы с RT.KeyValue необходимо учитывать рекомендации к минимальным системным требованиям оборудования.

Как и у большинства баз данных, пропускная способность RT.KeyValue повышается за счет большего количества ядер ЦП, большего объема оперативной памяти и более быстрых дисков. RT.KeyValue может работать на небольших серверах для сред тестирования или разработки минимальный производственный сервер требует не менее 2 ядер и не менее 8 ГБ ОЗУ.

Типичные производственные серверы имеют 8 или более ядер и не менее 32 ГБ ОЗУ.

Центральный процессор

RT.KeyValue работает одновременно с множеством единовременных запросов (как на чтение, так и на запись), используя несколько потоков, работающих на максимально возможном количестве ядер ЦП. Путь записи RT.KeyValue имеет тенденцию быть значительно оптимизированным (запись в журнал фиксации, а затем вставка данных в таблицу памяти), поэтому запись, в частности, обычно связана с процессором. Следовательно, добавление дополнительных ядер ЦП часто увеличивает пропускную способность как чтения, так и записи.

Объем оперативной памяти

RT.KeyValue работает в виртуальной машине Java, которая предварительно выделяет кучу (heap) фиксированного размера (системный параметр Java Xmx). Помимо кучи (heap), RT.KeyValue будет использовать значительные объемы оперативной памяти для метаданных сжатия, фильтров Блума, кешей строк, ключей и счетчиков, а также кеша страниц в процессе. Наконец, RT.KeyValue воспользуется преимуществами кеширования страниц операционной системы, сохраняя файлы недавно использованных частей в ОЗУ для быстрого повторного использования.

Для оптимальной производительности операторы должны тестировать и настраивать свои кластеры в соответствии с их индивидуальной рабочей нагрузкой. Однако основные рекомендации предполагают:

* Всегда следует использовать ОЗУ с ECC, поскольку в RT.KeyValue мало внутренних средств защиты от повреждения битового уровня.
* Куча (heap) RT.KeyValue должна быть не менее 2 ГБ и не более 50% оперативной памяти вашей системы.
* Кучи (heap) размером менее 12ГБ должны учитывать сборку мусора ParNew / ConcurrentMarkSweep.
* Для Кучи размером более 12 ГБ следует учитывать:
* Куча 16ГБ с 8-10ГБ нового поколения, коэффициент выживаемости 4-6 и максимальный порог владения 6
* G1GC

Диски

RT.KeyValue сохраняет данные на диск для двух очень разных целей. Первый - в журнал фиксации, когда делается новая запись, чтобы ее можно было воспроизвести после сбоя или выключения системы. Второй - в каталог данных, когда пороговые значения превышены, и memtables сбрасываются на диск как SSTables.

Журналы фиксации получают каждую запись, сделанную в узел RT.KeyValue, и могут блокировать клиентские операции, но они всегда читаются только при запуске узла. С другой стороны, запись в SSTable (файл данных) происходит асинхронно, но считывается для удовлетворения запросов клиентов. SSTables также периодически объединяются и перезаписываются в процессе, называемом уплотнением. Данные, хранящиеся в каталоге commitlog, - это данные, которые не были постоянно сохранены в каталогах данных SSTable - они будут периодически очищаться после того, как они будут сброшены в файлы данных SSTable.

RT.KeyValue очень хорошо работает как на вращающихся жестких дисках, так и на твердотельных дисках. В обоих случаях отсортированные неизменяемые таблицы SSTables RT.KeyValue допускают линейное чтение, небольшое количество поисков и небольшое количество перезаписей, максимизируя пропускную способность для жестких дисков и срок службы твердотельных накопителей, избегая усиления записи. Однако при использовании вращающихся дисков важно, чтобы журнал фиксации (commitlog\_directory) находился на одном физическом диске (не просто в разделе, а на физическом диске), а файлы данных (data\_file\_directories) были настроены на отдельный физический диск. Отделив журнал фиксации от каталога данных, записи могут выиграть от последовательных добавлений в журнал фиксации без необходимости искать вокруг диска при чтении данных запроса из различных таблиц SSTable на диске.

В большинстве случаев RT.KeyValue предназначена для обеспечения избыточности через несколько независимых недорогих серверов. По этой причине использование NFS или SAN для каталогов данных является антипаттерном, и его обычно следует избегать. Точно так же серверы с несколькими дисками часто лучше обслуживаются с помощью RAID0 или JBOD, чем RAID1 или RAID5 - репликация, предоставляемая RT.KeyValue, устраняет необходимость репликации на уровне диска, поэтому обычно рекомендуется, чтобы операторы использовали дополнительную пропускную способность RAID0, а не защита от сбоев с помощью RAID1 или RAID5.

Общие варианты облачных вычислений (Common Cloud Choices)

Многие крупные пользователи RT.KeyValue работают в различных облаках. Пользователи должны выбрать оборудование, аналогичное тому, что потребуется в физическом пространстве. В EC2 популярные варианты включают:

* инстансы i2, которые обеспечивают как высокое соотношение ОЗУ: ЦП, так и локальные временные твердотельные накопители
* инстансы i3 с дисками NVMe
* EBS работает нормально, если вам нужно легкое резервное копирование и замена
* инстансы m4.2xlarge / c4.4xlarge, которые предоставляют современные процессоры, расширенные возможности работы в сети и хорошо работают с хранилищем EBS GP2 (SSD)

Как правило, производительность диска и сети увеличивается с увеличением размера и поколения экземпляров, поэтому новые поколения экземпляров и более крупные типы экземпляров в каждом семействе часто работают лучше, чем их меньшие или более старые альтернативы.

Установка RT.KEYVALUE

В данном разделе представлена инструкция по развертыванию поддерживаемого выпуска RT.KeyValue на серверах Linux.

RT.KeyValue работает с дистрибутивами Linux, включая (но не ограничиваясь):

* Centos 7.

Это не окончательный и не предписывающий список платформ операционных систем. Однако пользователям будет полезно провести собственные тесты, особенно для менее популярных дистрибутивов Linux. Развертывание в более старых версиях не рекомендуется, если у вас нет опыта работы со старым дистрибутивом в производственной среде.

Предустановки

Установите последнюю версию Java 8, OpenJDK 8. Чтобы убедиться, что у вас установлена правильная версия java, введите java -version .

Для использования cqlsh необходима последняя версия Python 3.6+ или Python 2.7 (поддержка не рекомендуется). Чтобы убедиться, что у вас установлена правильная версия Python, введите python --version.

Способ установки

RT.KeyValue, в качестве способа установки, использует установку пакетов (RPM, YUM)

Для пакетных установок требуются права root. Если вы хотите установить RT.KeyValue с помощью YUM, установите сборку RPM в дистрибутивы.

Установка RPM пакетов

Установка RPM пакета представлена в виде пошаговой инструкции.

Шаг 1. Проверьте установленную версию Java.

Команда:

|  |
| --- |
| $ java -version |

Результат:

|  |
| --- |
| openjdk version "1.8.0\_222"  OpenJDK Runtime Environment (build 1.8.0\_222-8u222-b10-1ubuntu1~16.04.1-b10)  OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode) |

Шаг 2. Добавьте репозиторий RT.KeyValue в файл /etc/yum.repos.d/cassandra.repo (как пользователь root).

Например:

|  |
| --- |
| [RT.KeyValue]  name=RT.KeyValue  baseurl=https://repo.data.tdata.tech/repository/rt.keyvalue\_rpm/ |

Шаг 3. Обновите индекс пакета из источников:

|  |
| --- |
| $ sudo yum update |

Шаг 4. Установите RT.KeyValue с помощью YUM:

|  |
| --- |
| $ sudo yum install cassandra |

|  |
| --- |
| **Примечание.** |
| В процессе установки будет создан новый пользователь Linux cassandra. Сервис RT.KeyValue также будет запущен от имени этого пользователя. |

Шаг 5. Запустите сервис RT.KeyValue:

|  |
| --- |
| $ sudo service cassandra start |

Шаг 6. Проследите за ходом запуска с помощью:

|  |
| --- |
| $ tail -f logs/system.log |

RT.KeyValue готов, когда вы видите такую запись в system.log:

|  |
| --- |
| INFO [main] 2019-12-17 03:03:37,526 Server.java:156 - Starting listening for CQL clients on localhost/127.0.0.1:9042 (unencrypted)... |

Шаг 7. Проверьте статус RT.KeyValue:

|  |
| --- |
| $ nodetool status |

В столбце состояния вывода должно быть указано UN, что означает «Up / Normal».

В качестве альтернативы подключитесь к базе данных с помощью:

|  |
| --- |
| $ cqlsh |

КОНФИГУРИРОВАНИЕ

Расположение файлов конфигурации RT.KeyValue зависит от типа установки:

* пакет: каталог /etc/cassandra.

Файл конфигурации RT.KeyValue по умолчанию, cassandra.yaml, достаточен для изучения простого одноузлового cluster (кластера). Однако все, что выходит за рамки локального запуска кластера с одним узлом, требует дополнительной настройки различных файлов конфигурации RT.KeyValue. Некоторые примеры, требующие нестандартной конфигурации, - это развертывание многоузлового кластера или использование клиентов, которые не работают на узле кластера.

* cassandra.yaml: основной файл конфигурации для RT.KeyValue
* cassandra-env.sh: можно установить переменные среды
* cassandra-rackdc.properties ИЛИ cassandra-topology.properties: установка информации о стойке и центре обработки данных для кластера
* logback.xml: конфигурация ведения журнала, включая уровни ведения журнала
* jvm-\*: несколько файлов конфигурации JVM как для сервера, так и для клиентов
* commitlog\_archiving.properties: установить параметры архивирования для commitlog (журнала фиксации)

Два примера файлов конфигурации также можно найти в ./conf:

* metrics-reporter-config-sample.yaml: настройка того, что будет собирать отчет с метриками
* cqlshrc.sample: как можно настроить оболочку CQL, cqlsh

Основные свойства среды выполнения

Настройка RT.KeyValue выполняется путем установки свойств yaml в файле assandra.yaml. Минимально следует рассмотреть возможность установки следующих свойств:

* cluster\_name: задайте имя вашего кластера.
* seeds: список IP-адресов исходных узлов кластера, разделенных запятыми.
* storage\_port: убедитесь, что порт по умолчанию 7000 не заблокирован брандмауэром.
* listen\_address: адрес прослушивания (listen address) - это IP-адрес узла, который позволяет ему взаимодействовать с другими узлами в кластере. По умолчанию установлено значение localhost. В качестве альтернативы вы можете установить listen\_interface, чтобы сообщить RT.KeyValue, какой интерфейс использовать, и последовательно какой адрес использовать. Установите одно свойство, а не оба.
* native\_transport\_port: убедитесь, что порт по умолчанию 9042 не заблокирован брандмауэром, чтобы клиенты, такие как cqlsh, могли взаимодействовать с RT.KeyValue через этот порт.

Изменение расположения каталогов

Следующие свойства yaml управляют расположением каталогов:

* data\_file\_directories: один или несколько каталогов, в которых расположены файлы данных, такие как SSTables.
* commitlog\_directory: каталог, в котором находятся файлы журнала фиксации.
* saved\_caches\_directory: каталог, в котором находятся сохраненные кеши.
* hints\_directory: каталог, в котором расположены подсказки.

По соображениям производительности, если у вас несколько дисков, рассмотрите возможность размещения файлов журнала фиксации и данных на разных дисках.

Переменные среды

Параметры уровня JVM, такие как размер кучи (heap), можно установить в cassandra-env.sh . Вы можете добавить любой дополнительный аргумент командной строки JVM в переменную среды JVM\_OPTS; при запуске RT.KeyValue эти аргументы будут переданы JVM.

Логирование

Регистратором по умолчанию является logback. По умолчанию он регистрирует:

* Уровень **INFO** в system.log
* Уровень **DEBUG** в debug.log

При работе на переднем плане он также будет регистрироваться на уровне INFO в консоли. Вы можете изменить свойства ведения журнала, отредактировав файл logback.xml или выполнив команду nodetool setlogginglevel.