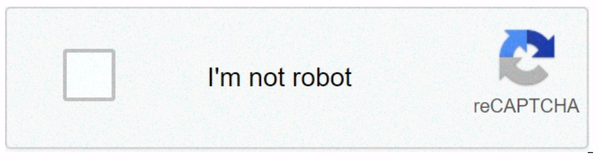

Continue

Slayer raining blood acoustic tab

Get Cassandra: The Definive Guide, 2nd Edition now with O'Reilly's online learning. O'Reilly members experience live online training, books, videos and digital content from more than 200 publishers. Imagine what you can do if scalability wasn't a problem. Use this policy guide to learn how the Cassandra database management system processes hundreds of tediths of data that are still highly available in multiple data centers. Updated to cassandra 3.0, this extended second edition includes technical details and practical examples that you need a database to work on in a production environment. View/send Errata Download sample code Foreword foreword 1. Outside relational databases 2. Introducing Cassandra 3. Installing Cassandra 4. Cassandra's polling language 5. Data modelling 6. Cassandra Architecture 7. Setting up Cassandra 8. Customers 9. Reading and writing data 10. Monitoring 11. Maintenance 12. Adjusting performance 13. Security on The 14th. Enable and integrate index book AWS CertifiEd Solutions Architects Study Guide by Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut, Kevin E. Kelly, Sean Senior, John Stamper Validate your AWS skills. This is your chance to take the next step in your career ... Book Modern Java in action Raoul-Gabriel Urma, Alan Mycroft, Mario Fusco modern java connect operations With java new features with practical applications. Use... book Software Engineering At Google by Titus Winters, Tom Manshreck, Hyrum Wright Today, software engineers need to know not only how to program efficiently, but also how to ... Aditya Bhargava's book Grokking Algorithms, Aditya Y. Bhargava Grokking Algorithms is a friendly insight into this core topic of computer science. That's where you learn ... Cassandra's coming forward has helped develop and develop core technology for understanding databases, understanding Cassandra and applying it to real-life work by comparing traditional, ly national databases. IMS, RDBMSs, NoSQL. Horse, car, aeroplane.2 guarantees data consistency, relationship databases are used for transactions, transaction handling is lock-dependent, there is a heavier load d application layer for optimization, such as indexing, sharing complex SQL, and stored procedure e adding cache layers such as memcached, Redis, EHcache, etc., more serious consistency issues back into the database, denormalization (contrary to relationship data principles), such as a data redundancy ratio data model, also solves only some specific problems, not the silver bullet of data storage. But the relationship database is the most researched and used data model, be it a company or a university. It seems to be a data model that's everywhere. Key value type: Redis, Memccached Get Cassandra: The Definive Guide, 2nd Edition now e-learning. O'Reilly members experience live online training as well as books, videos and digital content over 200,200 by Jeff Carpenter and Eben Hewitt Copyright © 2016 Jeff Carpenter, Eben Hewitt. All rights reserved. Printed in the United States. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. O'Reilly's books can be purchased for educational, business or promotional purposes. Online versions are also available for most items (. For more information, please contact our Corporate/Institutional Sales Department: 800-998-9938 or corporate@oreilly.com. Reporting by Mike Loukides and Marie Beaugureau Production Editor: Colleen Cole Copyeditor: Jasmine Kwityn Proofreader: James Fraleigh Indexer: Ellen Troutman-Zaig Interior Designer: David Futato Cover Designer: Karen Montgomery Illustrator: Rebecca Demarest June 2 016: Second edition 2010-11-12: First release 2016-06-27: Second release 2017-04-07: Third release see more information about the release. The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. Cassandra: The Definitive Guide, the cover image and the associated trade dress are trademarks of O'Reilly Media, Inc. While publishers and authors have used in good faith their efforts to ensure that the information and instructions contained in this work are accurate, the publisher and authors are not liable for errors or omissions, including, without limitation, liability for damages ... Get Cassandra: The Definitive Guide, 2nd Edition now with O'Reilly's online learning. O'Reilly members experience live online training, books, videos and digital content from more than 200 publishers. Book title: Cassandra: The Definitive Guide, 2nd Edition Author: Eben Hewitt, Jeff Carpenter ISBN-10: 1491933666 Year: 2016 Pages: 370 Language: English File size: 7.8 5 T File format: PDF Imagine what you could do if scalability wasn't a problem. With this practical guide, you will learn how the Cassandra database management system processes hundreds of teratabytes of data and remains highly available in multiple data centers. This extended second version, updated for Cassandra 3.0, provides technical details and practical examples that you need a database to work on in a production environment. Authors Jeff Carpenter and Eben Hewitt demonstrate the benefits of Cassandra's non-relational design, paying special attention to data modelling. If you're a developer, DBA, or application architect who wants to solve a database scaling issue or your future-proof app, this guide will help you take advantage of Cassandra's speed and flexibility. About Cassandra's distributed and distributed structure Use Cassandra Query Language (CQL) and cqlsh – CQL user interface Create a workable data model and compare it to the corresponding relational model Develop sample applications using client drivers for languages such as Python and and and Discover cluster topology and learn how to how nodes exchange information Maintain high performance in your cluster Enable Cassandra on site, in the cloud, or with Docker Integrate Cassandra with Spark, Hadoop, Elasticsearch, Slier and Luc Downloadene PDF NOTIFICATION: IF THE DOWNLOAD LINK IS BROKEN REPORT US AT (email protected) In this chapter, you will learn how to use various tools to track and understand important events in the Cassandra cluster. We look at some simple ways to see what's going on, such as changing the posting level and understanding output. Cassandra also has built-in support for Java Management Extensions (JMX) extensions, providing a versatile way to track your Cassandra nodes and the Java environment behind them. With JMX, we can see the health and continuous events of the database and even interact with it remotely to adjust certain values. JMX is an important part of Cassandra, and we spend a few time making sure we know how it works and what Cassandra monitors and treats with JMX. Let's start! Logging The easiest way to understand what is going on in the database is to change the posting level only to make the output more detailed. This is useful in developing and learning cassandra's behind-the-scenes drawings. Cassandra uses the Java (SLF4J) api to log and log back on a simple logging faça as an implementation. The SLF4J offers the look and feel of various log frames, including logback, Log4J, and Java's built-in log (java.util.logging). For more information about Logback, . By default, cassandra's server log level is set to INFO, which does not provide detailed information about cassandra agglomerations. It prints only basic update updates as follows: INFO (main) 2015-09-19 09:40:20, 215 Cassandra Daemon.java:19 149 - Host name: Carp-iMac27.local INFO (main) 2015-09-19 09:40:20,233 YamlConfi GurationLoader.java:92 - Download settings from file: //Users/jeff/Cassandra/apache-cassandra-2.1.8/conf/cassandra.yaml INFO (main) 2015-09-19 09:40:20,333 YamlConfigurationLoader.java:135 - Node Assembly... When you start Cassandra at the terminal, this output is kept running in the terminal window by skipping the -f flag for the program (so that the output appears at the front of the terminal window). But Cassandra also writes these logs to the physical file so that you can check later. By changing the posting level to DEBUG, we can see more clearly the functions the server is working on, rather than just looking at these step updates. To change the posting level, open the file /conf/logback.xml and <root level=INFO> <appender-ref=FILE></appender-ref> <appender-ref=STDOUT></appender-ref> </root> find the section that reads as follows: Change the first line to display it After you complete the change and <root level=DEBUG> save the file, Cassandra will soon start printing a debugging level posting statement. This is because the default posting is configured to check the profile every minute as follows: </root> </root> True > now Cassandra's work allows us to see more action. This will give you an accurate idea of what Cassandra does and when, which is useful for troubleshooting. But it also helps to simply understand what Cassandra is doing to sustain herself. Adjusting production records In production, of course, you need to adjust the posting level back to WARN or ERROR because detailed printing slows down significantly. By default, Cassandra log files are stored in the Logs folder in cassandra's installation folder. To change the location of the log folder, locate the following entry in the logback.xml file, and then select a different file name: <file>$(cassandra.logdir.log system.log</file> Missing log files If you do not see log files at the specified location, make sure that you are the owner of the folder or that you have at least the correct read/write permissions. Cassandra won't tell me if it can't sign in. It can't be written. The same goes for data files. Other .xml support scrolling log files. By default.log can scroll to the archive when they are 20 MB in size. Each log file archive is compressed in zip format and named after the pattern system .log.1.zip, system .log.2 .zip, etc. Tailing You don't need to start Cassandra with a desktop switch to view scroll logs. You can also start it without the -f switch, and then drag the log. The print is not unique to Cassandra; It is a small program that is offered in the Linux version so that you can view the new values printed on the console because they are attached to the file. To track the log, follow these steps to start Cassandra: $bin/cassandra then open another console, type the tail command and enter it in a location passed to a specific file as follows: $tail -f $CASSANDRA? HOME/logs/system.log f option means company, and when Cassandra prints the data to a physical log file, the heel prints it on the screen. You can stop following by pressing Ctrl-C.C. If you are using Windows, you can do the same, but Windows itself does not include tail programs. So, to do this, you need to download and install Cygwin, a free open source Bash shell simulator. Cygwin allows you to use the Linux-style interface and use various Linux tools on Windows. You can then start Cassandra regularly and follow the log files with the following command: $tail -f %CASSANDRA_HOME% of the console output.log which displays the output in the console in the same way as the preset. Checking log files After you perform server debugging logging, you can see that more help can be provided during debugging. For example, here we can see the output when you use cqlsh to enter simple values into the database: cqlsh> INSERT INTO hotels (ID, name, phone, address) ARVOT (AZ123, Comfort Suites Old Town Scottsdale, ... (480) 946-1111, katu: 3275 N. Drinkwater Blvd., ... kaupunki : Scottsdale, valtio : AZ, zip_code: 85251) DEBUG (SharedPool-Worker-1) 2015-09-30 2015-09-30 Message.java:506 - Received: OPTIONS, v-4 DEBUG (SharedPool-Worker-1) 2015-09-30 06:21:41,410 Message.java:525 - Reply: SUPPORTED COMPRESSION (snappy, lz4), CQL_VERSION 2015-09-30 06:21:42,082 Message.java:506 - Received: QUERY INSERT INTO hotels.hotels (id, name, phone, address) VALUES ('AZ123', 'Comfort Suites Old Town' Scottsdale, (480) 946-1111, street: 3275 N. Drinkwater Blvd., city : Scottsdale, stating : AZ, zip_code: 85251); 2015-09-30 06:21:42,086 AbstractReplicationStrategy.java:87 - Clear cached endpoints DEBUG (SharedPool-1) 2015-0930 06:21:21 42,087 Trace .java:155 - Request Full DEBUGGING (SharedPool-Worker-1) 2015-09-30 06:21:42,087 Message.java:525 - Reply: RESULT, BLANK, because it works in one node cluster, this output has less pereper performance than in other cases. If we then download the line with a simple query: cqlsh> select s hotel.hotels; The server log stores this query as follows: DEBUG (Sharepool-Worker-1) 2015-09-30 06:27:27,392 Message.java:506 - Received: SELECT QUERY s from hotel.hotels; pageSize s 100, v s4 DEBUG s SharedPool-Worker-1 s 2015-09-30 06:27:27,395 StorageProxy.java:2021 - Predicted result line range: 0.0; requested rows: 100, ranges.size(): 257; Concurrent area requests: 1 DEBUG (Sharepool-Worker-1) 2015-09-30 06:27:27,401 ReadCallback.java:141 - Read: 0 ms. DEBUG (SharedPool-Worker-1) 2015-09-30 06:27:27,401 Trace .java :155 - request a full DEBUG (SharedPool-Worker-1) 2015-09-30 06:27:27,401 Message.java:525 - Reply: LINES (hotel, hotels), org.apache.cassandra.db.marshal.UUIDType address (hotel, hotels), org.apache.cassandra.db.marshal.userType (hotel, 61646472657373, 7374726565574: org.apache.cassandra.db.marshal.UTF8Type, 63697479:. rg.apache.cassandra.db.marshal.UTF8Type, 7374617465: org.apache.cassandra.db.marshal.5 UTC8Type, 7a69705f636f6465: org.apache.cassandra.db.marshal.Int32Type) Hotels), org.apache.cassandra.db.marshal.UTF8Type, sphone (hotel, hotels), org.apache.cassandra.db.marshal.UTF8Type, away (hotel, hotels), org.apache.cassandra.db.marshal.setType (org.apache.cassandra.db marshal. UUIDType) . . 452d27e1-804e-479b-aeaf-61d1fa31090f 3275 N. Drinkwater Blvd.: Scottsdale: AZ:85251 Comfort Suites Scottsdale Old Town (480) 946-1111 Null As you can see, the server uploads each column of disk format data we request to be grouped through the category responsible. The DEBUGGING log level should provide you with sufficient information to track server work. This section explores how Cassandra uses Java Management Extensions (JMX) to remotely manage servers. JMX started with Java Specification Request (JSR) 160 and has been a key part of Java since version 5.0. JMX add-on JMX is a Java API that provides application management in two main ways. First, with JMX, you can understand your app's health and overall performance in terms of memory, threading, and CPU usage - these contents are usually available for any Java app. Secondly, JMX allows you to use certain aspects of the application that have been detected. Instrumentation is an app code wrapper that provides hooks from application to JVM so that JVM can collect data that can be used by external tools. Such tools include tracking agents, data analysis tools, analysers, etc. JMX not only allows you to view such data, but also manages your app during a manageable time by updating values when enabled. JMX is typically used for various application control functions, such as low available memory recognition, including the size of thread data in each pile's build mode, such as lock detection, thread spikes, and current active threads Detailed class loader trace Log level management General information such as application uptime and active class paths Many popular Java apps use JMX to detect, including JVM itself, Tomcat application servers, and Cassandra. The description of the JMX architecture is shown in Figure 10-1. JMX architecture is simple. The JVM collects data from the underlying operating system. The JVM itself has been tested, so as mentioned earlier, many of its features are publicly manageable. An recognizable Java application, such as Cassandra, runs on this basis and exposes some of its functions as manageable objects. JDK includes an MBean server that provides recognized functionality to the JMX management application through a remote protocol. JVM also provides administrative features using a simple network monitoring protocol (SNMP), which can be useful if you use SMTP monitoring tools such as Nagios or Zenoss. However, you can only control what a developer can control for you. Fortunately, Cassandra developers have been able to identify most database modules, and managing them with JMX is easy. The tools of this Java application are executed by wrapping the application code that you want JMX to connect to the managed bean. Connect to cassandra jconsole via JConsole with a standard Java Development Kit. It provides a graphical user interface client for working with MBeans, which can be used for local or remote management. Connecting to Cassandra on JMX port to JConsole. To do this, open a new end and type: >jconsole When you run the jconsole, you will see a sign-in screen similar to that shown in Figure 10-2. From here, if you are monitoring nodes on the same machine, double-click org.apache.cassandra.service.Cassandra daemon in the Local Process section. If you want to nodes on other computers, click the Remote Processes option button, and then type the host and port you want to connect to. By default, cassandra JMX is sent at port 7199, so you can enter a value similar to the one shown here and click Connect: >Accident:7199 >Accident:7199 By default, Cassandra runs only when JMX is enabled for local use. To enable remote access, edit the file/cassandra-env-.sh (or cassandra.ps1 on Windows). In the file section, locate JMX, which contains options for managing JMX ports and other local/remote connection settings. When the connection to the server is connected, the default view contains four master classes of server space that are constantly updated: use of the build memory This shows the total amount of memory available to Cassandra and the amount of memory it is currently using. Downloaded categories in Cassandra. For such an effective programme, this figure is relatively small; Cassandra usually requires less than 3,000 courses. Compare it to programs like Oracle WebLogic, which typically loads about 24,000 categories. CPU Usage This shows the percentage of processors used by Cassandra. You can use the switch to adjust the time interval displayed in the chart. To see a more detailed view of cassandra's use of the Java pile and non-pile memory, click the Memory tab. By changing the values in the chart in the drop-down list, you can view Cassandra's memory in detail. If you think it's necessary, you can also (try) force the collection of garbage. You can connect to multiple JMX agents at once. Just select the file that → connection... And repeat the steps to connect to another Cassandra nodes that are running so that you can view multiple servers at once. Other JMX customers When looking for a JMX client, JConsole is a simple choice as it is easy to use and comes with JDK. But this is just a potential JMX customer - there are many other customers available. Here are some examples of customers who might meet your needs: Oracle Java Mission Control and Visual VM tools are also available with Oracle JDK, providing more powerful metrics, diagnostics, and visualizations for memory usage, threading, garbage collection, and more. The main comparison between the two is that the Visual VM is an open source project available under a GNU license, while Mission Control offers deeper integration with Oracle JVM through a framework called air traffic control. Java Mission Control can be JAVA_HOME/bin/jmc and Visual VM command for $JAVA_HOME/bin/jvisualvm. Both are suitable for development and production environments. The MX4J Management Extensions for Java (MX4J) project provides open source implementation for JMX, including tools like HTTP/HTML, such as JMX's embedded web interface. This allows you to interact with JMX through a standard Web browser. To integrate MX4J with Cassandra Setup, download the mx4j_tools.jar library, save the JAR file to the lib folder installed by Cassandra, and MX4J_ADDRESS and MX4J_PORT conf/cassandra envin asetukset.sh Jmxterm Jmxterm is a command-line JMX client that allows access to the JMX server without graphical interface. This is especially useful when working in a cloud environment where graphics tools are often more resource-intensive. Jmxterm is an open source Java project offered by Cyclops Group. IDE integration You can also find JMX customers integrated with the popular IDE; For example, eclipse jmx. MBea Overview Overview Beans representing a single resource to manage in the JVM. MBean interacts with the MBean server so that its functions are remotely accessed. JConsole is viewed in chapters 10-3. In this diagram, you will see tab windows that provide a general view of the threads, memory, and processor of each app, as well as a more detailed MBeans tab that reveals the possibility of interacting more in more detail with the MBean revealed by the app. For example, in the photo, we decide to look at the peak value of the threads. You can see that many other aspects of app detection are also available. Many aspects of the application or JVM can be detected, but they can be disabled. The thread dispute is an example of a potentially useful MBean that is closed in the JVM by default. These aspects are useful with debugging, so if you see MBean that you think can help you solve the problem, continue and enable it. Keep in place, however, that nothing is free, and it's a good idea to read about JavaDoc MBean, which you want to enable you to understand the potential impact on performance. For example, measuring processor time per thread is an example of useful but expensive MBean activity. MBean Object Name Policy When registering MBean on an MBean server, it specifies the name of the MBean object used to identify the JMX client. The object name consists of a list of domains followed by a pair of key values, at least one of which must identify the type you want to identify. A typical way is to choose a domain that resembles the name of the MBean java package and name the name of the MBean interface by type (minus MBean), but this is not strictly required. For example, the thread properties we've seen before appear under the JConsole java.lang.Threading heading and are implemented by java.lang.management. ThreadMXBean interface class that registers MBean under object name java. lang.type . . . Thread. When we discuss the different MBeans rules in this chapter, we recognize the name and interface of the MBean object, which allows you to navigate between the JMX client and Cassandra's source code. Some simple values in the app are revealed as features. One example is Thereading > PeakThreadCount, which reports only the values that MBean stores for the maximum number of threads used by the app at a time. You can update to see the latest values, but this can almost be done with it. Since such values remain internally within the JVM, it makes no sense to set it externally (it is derived from actual events and cannot be determined). But the other MBeans are configurable. They allow a JMX agent to use functions to obtain and set values. You can tell whether to allow MBea values to be set by viewing the values you want to enter. If it's untrue, you'll see a legend that indicates read-only and if so, you'll see one or more field that you can use to add a new value and a button to update it. An example of this is ch.qos.logback.classic.jmx.JMXConfigurator beans as shown in Table 10-4. accordance with article 10 of For JMXConfigurator, this class implements an interface called JMXConfigurator MBean that packages it for detection. To find out the correct parameters of setLoggerLevel, we check the javaDoc interface for this interface, which is available on . When you view the documentation, you can see that p0 represents the name of the log you want to change, and p1 represents the recorder that is set to the record level. Some MBeans return values for the javax.management.openmbean.CompositeDataSupport property. This means that these are not simple values that can be displayed in one field, such as LoadedClassCount, but multiple values. One example is > HeapMemoryUsage, which offers multiple data points and therefore has its own view. A different MBean function not only displays values or lets you set values, but allows you to do something useful. DumpAllThreads and resetPeakThReadCount are two such functions. Now we're about to start following and running Cassandra. Cassandran MBeans can use the exposed MBean to control Cassandra when connected to JMX agents such as JConsol. To do this, select the MBeans tab. In addition to the standard JAVA projects available for each agent, there are several Cassandra packages that contain manageable beans organized by package names starting with org.apache.cassandra. We are not talking about all this, but we need to focus on a few advantages. Many of Cassandra's categories are revealed to be MBeans, which actually means that they implement a custom interface that describes the functions and functions to be implemented that the JMX agent is hooked on. The steps to make MBean work are essentially the same. If you want JMX to enable features that aren't already in use, modify the source code according to this outline, and you'll be open for business. For example, let's look at Cassandran ActionManager and how it uses MBean from org.apache.cassandra.db.compaction. This is the definition of the ActionManagerMBean class. comments are omitted from the shortness: public interface CompactionManagerMBean Public Catalog <> <String, string=>> getCompactions ();<String> public List Get Action Summary (); public Tabular Data get ActionHistory (); public void forceUserDefinedCompaction(String dataFiles); public void stopCompaction (String Type); public void stopCompaction By Id; public void stopCompaction By Id; public int getCoreCompactor Threads (); the public cancellation seriesCoreCompactor Threads; public int get MaximumCompactor Threads (); maximcompator threads public voiding set; public int getCoreValidation Threads (); public void setCoreValidation threads (int number); public int get Threads (); maximvalidator threads public voiding set; As you can see from the definition of the MBean interface, there is no magic to talk about. This is only a global interface that specifies a set of actions that are vulnerable to CompMManager implementation and must be supported by JMX. This usually means storing metadata once regular operations have completed their work. </String></String,> </String></String,> CompactionManager class itself registers and deletes registrations of JMX features it maintains locally on the MBean server: public static final String MBEAN_OBJECT_NAME sorg.apache.cassandra.db: write compactionManager; // ... static instance of the new CompactionManager (); MBean Server mbs s ManagementFactory.getPlatformMBeanServer (); try smbs.registerMBean (instance, new ObjectName (MBEAN_OBJECT_NAME); Please note that MBean is registered with org.apache.cassandra .db, and its type is ActionManager. Features and features revealed by this MBean appear in the JMX client organization org.apache.cassandra.db> CompactionManager. The implementation completes all the work it intends to do and then implements the methods needed to communicate only with the MBean server. For example, this is the ActionManager implementation of StopCompaction() operations: public void stopCompaction (String type) s OperationType operationType.valueOf (type); CompactionManager iteration is running during compression, which stops each compression with the specified type. Javadoc tells us that valid types are COMPACTION, VALIDATION, CLEANUP, SCRUB, and INDEX_BUILD. When we view OperationManagerMBean at JConsole, we can select an action and view the StopCompaction() function as shown in Fig. 10-5. We can enter one of the above types and ask for the package to be discontinued. In the following sections, we take a look at the features that JMX can use to track and manage. Database MBean These are the Cassandra classes associated with the core database itself, which are exposed to customers of the domain org.apache.cassandra .db domain. This domain has many MBeans, but we focus on a few central MBeans related to storage, cache, log tying and table storage. The storage service MBean Since Cassandra is a database, it is basically a very complex storage program, so one of the first places you want to see when you have a problem is org.apache.cassandra.service.StorageServiceMBean. This allows you to check operamode and report normally if all goes well (other possible states are Leave, Join, Retire, and Client). You can also view the current set of active nodes in the farm and the set of nodes that cannot be used. If node is not available, Cassandra will indicate the IP address in the Unreachable Nadit property. If you want to change cassandra's log level while driving without interrupting the service (as shown in the previous example), you can call the SetLoggingLevel (String Class Qualifier, String level) method. For example, assume that you have set cassandra's log level to DEBUGGING because you are fixing the problem. Some of the methods described here allow you to try to help and then you may want to make the log-level system less important. To do this, go to the JMX client's StorageService service, such as JConsole JConsole INFO 03:08:20 set the log level to INFO for classes with the symbol 'org.apache.cassandra.gms.gossiper' (if the level does not display 'INFO', the log cannot parse 'INFO') after calling LoggingsetLevel , we will receive an inFO printout instead of DEBUG class statements. You can understand the amount of data stored in a slag node using the getLoadMap() method, which returns a Java Map with an IP address key that contains the corresponding storage load value. You can also use the string keyspace feature to use the percentage of data in the keyspace for each node. If you are looking for a key, you can use the getNaturalEndpoints function. Enter a table name and key to find an endpoint value for it, and it returns the list of IP addresses responsible for storing the key. GetRangeToEndpointMap also lets you get a map that describes the area-to-end point of ring topology. If you feel brave, you can call the truncated () function in the key mode provided for the given table. If all nodes are available, this action deletes all data in the table, but leaves its definition as it is. StorageServiceMBean provides you with a number of standard maintenance activities, including resumeBootstrap(), joinRing (), repairAsync (), drain (), removeNode (), decommission () and starting and stopping gossip, local broadcasts, and frugality (until Trift is finally removed). Understanding the available maintenance functions is important to maintain the health of the cluster, which we cover in more detail in Chapter 11. Storage agent MBean, as we learned in Chapter 6, org.apache.cassandra.service.StorageProxy provides StorageService with a layer to handle communication between customer requests and nodes. StorageProxyMBean provides the ability to check and set time different values for various functions, such as read and write. This MBean also provides access to the prompt switching settings, such as the maximum time window to save the prompt. Tip exchange statistics include getTotalHints() and getHintsInProgress(). Use to disable HintsForDC () on prompts for a specific node. You can also enable or disable this node to participate in the prompt to switch through setHintedHandoffEnabled() status or to check the current status with getHintedHandoffEnabled(). These are used separately in soldetol enablehandoff commands, disablehandoff, and statushandoff commands. You can use ReloadTriggerClasses() to install a new trigger without restarting the node. ColumnFamilyStoreMBean Cassandra is an org.apache.cassandra.db>Store registry instance org.apache.cassandra.db.ColumnFamily Store MBean stored in nodes under tables (formerly CoolumnFamilies). ColumnFamily StoreMBean provides access to each table and packaging settings. This allows you to temporarily override these settings on a specific node. When the node is restarted, these values are returned to the values specified in the table structure. MBean also reveals a large amount of data from a data store on the disk of table nodes. data store. EstimateKeys() provides an estimate of the number of partitions stored in this node. In summary, this information knows whether calling forceMajorCompaction() in this table could help free up space in this node and improve reading power. There is also a trueSnapshotsSize() function that allows you to specify the size of SSTable shapshot samples that are no longer active. A higher value here means that you can consider deleting these snapshots, possibly after making an archived copy. Because Cassandra stores indexes as tables, each index column also has a Column Family Store MBean instance that can be used in org.apache.cassandra.db> IndexTables (formerly IndexColumnFamilies) operations with the same features and functionality. CacheServiceMBean org.apache.cassandra.service.CacheServiceMBean provides access to the Cassandra key cache, line cache, and counter cache under the domain org.apache.cassandra.db> cache. The available data for each cache includes the maximum size and duration of the cached item, as well as the ability to void each cache. CommitLogMBean org.apache.cassandra.db.commitlog.CommitLogMBean reveals features and functions to help you understand the current state of the verification log. Commit LogMBean also reveals a recovery function() that can be used to restore database status from files in the archived shipping log. The default settings for validation log recovery commitlog_archiving.properties file, but can be overwritten by MBean. We learned more about data recovery from Chapter 11. Compression Manager MBean We've studied how it interacts with JMX in the source code of org.apache.cassandra.db.compaction.CompactionManagerMBean, but we're not really talking about what it's for. With this MBean, we can get statistics on previously made packages as well as the ability to force the compression of certain SSTable files that we identify by calling the ForceUserDefinedCompaction method in the Company Manager class. This MBean is utilized by the nodetool command, including compact, sealing histories and sealing statistics. Calf MBeans Cassandra offers two MBeans to monitor and determine whistleblower behavior. org.apache.cassandra.locator.EndpointSnitchInfoMBean provides the name of the telin and data center and the name of the calves in use. If you are using Dynamic Endpoint Snitch, you register at org.apache.cassandra.locator.FynamicEndpoint SnitchMBean. This MBean reveals the ability to reset the bad thresholds that a calf can use to mark nodes offline and lets you see the dots for each node. HintedHandoffManagerMBean In addition to the implicit switching functions mentioned earlier in Thestorage Service MBean also provides more detailed control over fast switching via org.apache.cassandra.db.HintedHandOffManagerMBean. MBean reveals the ability to enumer nodes that store prompts by calling listEndpointsPendingHints(). You can then force the prompt to be passed to a node in scheduleHintDelivery() rules, or use deletehHintsForEndpoint() to remove a prompt stored for a specific node. Besides, you can use pauseHint pauseHint Repeat tip exchange management org.apache.cassandra.hints.HintsService reveals HintsServiceMBean in domainorg.apache.cassandra.hints>IncesService. This MBean provides interrupt and reconnection and removes prompts that are stored on all nodes or nodes recognized by IP addresses. Because there is a lot of overlap between StorageService MBean, HintedHandOffManagerMBean, and HintsServiceMBean, consolidation of these functions is likely in future versions. MBean org.apache.cassandra.net domain includes MBean, which helps manage cassandra's network-related functions, such as Phi bug detection and gossip, messaging services, and stream management. FailureDetectorMBean org.apache.cassandra.gms.FailureDetectorMBean offers features that describe the status of other nodes and Phi points, as well as Phi sentencing thresholds. GossiperMBean org.apache.cassandra.gms.GossiperMBean provides access to Gossiper's work. We have discussed how StorageServiceMBean reports which nodes are available. Based on this list, you can call gossiperMBean getEndpointDowntime() to specify how long the specified node has been closed. Outages are measured from the perspective of the MBean node we have checked, and the value is reset when the node returns to the network. Cassandra uses this function internally to understand how long it can wait for the prompt to drop. The GetCurrentGenerationNumber() action returns the creation number associated with a specific node. The build number is included in gossip messages exchanged between nodes to distinguish between the current status of the node and the pre-restart mode. When the node is active, the creation number remains the same and increases each time the node restarts. Gossiper maintains it with a time stamp. AssassinateEndpoint() is trying to remove nodes from the ring by telling other nodes that the node has been permanently removed, such as the concept of character assassination in human gossip. The assassination node is the last maintenance step when the node cannot be removed correctly from the cluster. The Nodetool astice command uses this function. StreamManagerMBean org.apache.cassandra.streaming.StreamManagerMBean allows us to see the flow activity between the node and its equivalent. Here are two basic ideas: a flow source and a flow target. Each node can stream its data to another node for load balancing, which is supported by the StreamManager class. StreamManagerMBean provides the necessary view of data that moves between cluster nodes. StreamManagerMBean supports two modes of operation. GetCurrentStreams() provides a snapshot of current incoming and outgoing streams, and MBean also publishes notifications related to changes in workflow mode, such as initialization, completion, or failure. You can subscribe to these notifications and observe when streaming takes place. So when you work with StorageServiceMBean, if you're worried that the node won't receive the information it deserves, or if the node is unbalanced or even inappropriate, working with two MBeans will give you an idea of what's going on. Cluster. Meters MBeans MBeans The variable reported by Cassandra includes the following: a buffer pool variable that describes Cassandra's memory usage. CQL metrics, including the number of prepared and regular bank statement settlements. The dimensions of cached keys, rows, and counter caches, such as the number and capacity of entries, and hit and miss counts. Customer metrics, including the number of connected customers, and information about customer requests, such as latency, failure, and timeouts. Send logmeters, including log size and statistics for pending and completed tasks. Packing meters, including the total number of condensed bytes and the statistics to be processed and summarized. Connection meters for each node in the cluster, including gossip. Rejected message meters used as part of node tpstats. Read the corrective meters to describe the number of background corrections made over time and the number of times read corrections are blocked. Save variables, including running prompt counts and total prompts. Thread pool variables, including the activity of each thread pool, completed and blocked tasks. Table variables, such as cache, memtables, SSTables and Bloom filter usage, and delays in various read and write operations, are reported every 1 minute, 5 minutes, and 15 minutes. Key mode variables that aggregate variables in tables in each keyspace. Cassandra uses these variables through JMX using dropwizard Metrics open source Java libraries. Cassandra uses the Metrics library to register its meters, which in turn reveals them with the MBeans domain org.apache.cassandra.metrics. The Soldetool command uses many of these indicators, such as tpstats, histograms, and proxy combinations. For example, tpstats are simply a measure of thread reserves and deleted messages. Thread

MBeans org.apache.cassandra.internal domain includes MBean, which allows you to specify the thread pool associated with each step of Cassandra's event-driven architecture (SEDA). These steps include antientropystage, GossipStage, InternalResponseStage, MigrationStage and more. Read the fix for MBea's Historical Causes ReadRepairStage MBean is located under the org.apache.cassandra.request domain, not org.apache.cassandra.internal. Thread reserves are implemented through the JMXEnabled ThreadPool Executor and JMXEnabled Scheduled ThreadPool Executor categories of the org.apache.cassandra .concurrent package. Each step of MBean implements the JMXEnabled SchederEdPoolExecutorMBean interface, which allows you to view and specify the number of kernel threads and the maximum number of threads in each thread pool. The service MBean GCInspectorMXBean unveils a single function, getAndResetStats(), which retrieves and resets the garbage collectors cassandra collects with JVM. This MBean org.apache.cassandra.service domain. The secure MBean org.apache.cassandra.auth domain contains security-related MBeans grouped by the same Java package name. Starting with version 3.0, it consists of the License Cache MBean, which is revealed to customers as org.apache.cassandra.auth.PermissionsCache. We are discussing this MBean in Chapter 13. Monitor with nodes Nodetool comes with Cassandra, which can be found / bin. This is a command-line program that provides a versatile way to view, understand, and modify a cluster. The node allows you to get limited statistics about the farm, see the scope of maintenance for each node, move data from one node to another, remove activation nodes, and even fix problematic nodes. Many of the tasks in soldetol and JMX overlap with the features of the JMX interface. That's because behind the scenes, nodetool uses a name called org.apache .cassandra .tools. NodeProbe's utility class calls JMX. So, JMX does real work, the NodeProbe class is used to connect to a JMX agent and retrieve data, and the NodeCmd class is used to render it in an interactive command line interface. nodetool uses the same environmental settings as Cassandra daemon: bin/cassandra.in.sh and conf/cassandra-env.sh unix (or bin/cassandra.in.bat and conf/cassandra-env.ps1). Log logging .xml conf/logback.xml-tools file. Starting a node is an easy task. Just open the terminal, go, and then type the following command: bin/nodetool will help this cause the program to print a list of available commands, several of which we will cover for now. Driving a node jaol without parameters corresponds to a help command. You can also use the name of a specific command to perform additional instructions. Connecting to a specific node In addition to the Help command, you must connect the node to the Cassandra node to access the data in the node or cluster. The -h switch can be used to identify the IP address of the node that you want to connect to nodes with nodes. If the IP address is not specified, the tool attempts to connect to the default port on the local computer, which is the sample approach that is adopted in this chapter. Getting cluster data You can get different information about the farm and its nodes that we cover in this section. Get basic information about a single node or all the nodes in the participating ring. Describecluster describecluster command prints basic farm information, including names, whistleblowers and partitioners: $bin/nodetool describecluster Cluster Information: Name: Test Cluster Snitch: org.apache.cassandra.locator.DynamicEndpointSnitch Partitioner: org.apache.cassandra.dhmmur 3PartPartitioner Schema versions: 2d4043cb-2124-3589-b2d0-375759b9dd0a: s127.0.0.1, 127.0.0.2, 127.0.0.3 The last part of the output is particularly important for identifying the discord or patterns between nodes in the table definition. When Cassandra changes through cluster progression mode, any differences are usually resolved quickly, so delayed pattern differences usually indicate that the restartable node is down or non-flammable. A more direct way to identify cluster nodes and their locations is to use $ bin/nodetool status Datacenter: datacenter1 . . . . . 103.82 KB 256 ? 31d9042b-6603-4040-8aac-fef0a235570b UN 127.0.0.2 110.9 KB caad1573-4157-43d2-a9fa-88f79344683d UN 127.0.0.3 109.6 KB 256 ? The e78529c8-ee9f-46a4-8bc1-3479f9a1860 mode is organized by data centers and scaffolding. The status of each node is identified by a two-character code in which the first character indicates whether the node has started (currently available and ready for query) or down, and the second character indicates the status or operating state of the node. The load column specifies the amount of data held by each node. My column indicates the effective percentage of the id range owned by the node and considers replication. Because we do not specify a key state, and because different key states in this farm have different replication policies, the node cannot calculate a meaningful ownership percentage. The Data Information command prompts nodes to connect to a single node and retrieve basic information about their current status. Anna vain sen solmun osoite, johon haluat tietoja: $ bin / nodetool -h 192.168.2.7 info ID: 197efa22-ecaa-40dc-a010-6c105819bf5e Gossip Active : true Thrift Active : false Native Transport Active: 301.17 MB Generation Ei : 1447444152 Käyttöaika (sekuntia): 1901668 Kasamuisti (Mt): 395.03 / 989.88 Pois kasamuistista (Mt) : 2.94 Data Center : Datacenter1 Rack : Rack1 Poikkeukset: 0 Välimuistiavain : merkinnät 85, koko 8,38 KB, KT kapasiteetti 49 Mt, 47958 osumaa, 48038 pyyntöä, 0,998 viimeisin osumanopeus, 14400 tallennusjakso sekunteina Rivirivi : merkinnät 0, koko 0 tavua, kapasiteetti 0 tavua, 0 osumaa, 0 pyyntöä, NaN:n viimeisin osumanopeus, 0 tallennusjakso sekunteina Counter Cache: 0, koko 0 tavua, kapasiteetti 24 Mt, 0 osumaa, 0 pyyntöä, NaN:n viimeisin osumanopeus, 72000 jaksoa käännässä: (kutsu -T/tokens nähdäksesi Ilmoitettuja tietoja ovat solmun muisti ja levyn käyttö (kuormitus) sekä Cassandran tarjoamista eri palveluista. You can also check the statusgossip, statusthrift, statusbinary, and statushandoff for node commands for each service status (note that switching mode is not part of the data). Ring To specify ring nodes and their position with a node command, follow these steps: $bin/nodetool ring Datacenter: Datacenter1 , Address Rack State Loads Token 92082375882789476801 192.168.2.5 rack1 Normal Up 243.6 KB? -9203905334627395805 192.168.2.rack1 Up Normal 243.6 KB ? -9145503818225306830 192.168.2.7 rack1 Up Normal 243.6 KB ? -9091015424710319286 This output is organized by vnodes. Here we can see the IP address of all the nodes in the ring. In this case, we have three nodes, all of which have been started (currently available and ready for query). The load column specifies the amount of data held by each node. The output of the description command is similar, but it is arranged around the scope of the id. Other useful space commands offered by the Node Center include: and setLoggingLevels commands enable the use of Logback, as we discussed earlier You can also use the nodetool of statistical contacts to collect statistics on server status at the aggregate level and at the level of specific key states and tables. The two most commonly used commands are tpstas and table stats, which we check now. The TPSTATS TPSTATS tool provides information about the spiral pool maintained by Cassandra. Cassandra is very co-standard and optimized for multiprocessor/multi-core machines. In addition, Cassandra has internally step-by-step event-driven architecture (SEDA), so understanding the behavior and health of thread pools is important for Cassandra's good maintenance. To find statistics about thread pools, use the tpstats command to run the nodetool: $bin/nodetool tpstats Pool Name Active Pending Completed Completed Blocked All time blocked ReadStage 0 0 21 6 0 0 MutationStage 1 0 3637 0 0 CounterMutationStage 0 0 0 0 0 ViewMutationStage 0 0 0 0 Gossip Stage 0 0 0 0 RequestResponseStage 0 0 0 0 0 AntiEntropyStage 0 0 0 0 0 0 0 MigrationStage 0 0 0 0 MiscStage 0 0 0 0 0 0 InternalResponseStage 0 0 0 0 ReadRepairStage 0 0 0 Korjausstage 0 Message Type Drop read 0 RANGE_SLICE 0 _TRACE 0 TIP 0 MUTATION 0 COUNTER_MUTATION 0 BATCH_STORE 0 BATCH_REMOVE 0 REQUEST_RESPONSE 0 PAGED_RANGE 0 READ_REPAIR 0 The printout top displays the task information for each Cassandra thread reserve. You can see directly where the number of operations is and whether they are active, pending, or completed. This output was captured during the writing operation, so it is shown that there are active tasks in The NationalStage. The bottom of the printout indicates the number of messages deleted for the node. Dropped messages are cassandra's load load implementation indicator that each node uses to protect itself when it receives more requests than it can handle. For example, messages between nodes that receive a node, rpc_timeout not processed in the node, are dropped instead of processed because the coordination node no longer waits for a response. If you see a large number of zeroes in the output of blocked tasks and dropped messages, there is very little activity on the server or Cassandra does a great job of maintaining the load. Many values, which are not zero, indicate that Cassandra is difficult to follow and may indicate the need for some of the techniques described in Chapter 12. You can use table statistics to view summary statistics for key states and tables by using the tablestats command. You can also recognize the command from the previous name cfstats. Here's an example of the hotel's key space output: $bin/nodetool tablestats hotel Keyspace: Hotel Read Count: 8 Read Latency: 0.617 ms. Writing volume: 13 Writing latency: 0.13330769230769232 ms. Pending rinses: 0 Table: Hotels SSTable count: 3 Used mode (live): 16601 Total used mode: 16601 Mode used by snapshots 0 Accumulated memory used (total): 140 SSTable Compression: 0.62773226277723 rating): 19 Number of Memtable cells: 8 Memtable data size: 792 Memtable off heap memory: 0 Memtable switch count: 1 Local number: 8 Local number late: 0,680 ms Local write count: 13 Local writing local latency: 0,148 0,148 Pending flushes: 0 Wrong bloom filter positives: 0 Wrong bloom filter ratio: 0.00000 Used Bloom filter mode: 56 Bloom filter out of used pile memory: 32 Index summary of the pile memory used: 84 Compression data for used pile memory: 24 rearranged minimum partition data: 30 rearranged maximum partition radars: 149 compressed partition radars: 87 Average living cells per slice (last). five minutes): 1.0 Maximum number of live cells per slice (last five minutes): 1 Average tombstones per slice (last five minutes): 1.0 Headstones per slice: 1 Here we omitte the output of other tables in the key space , so we can focus on the hotel table; Create the same statistics for each table. We can see the read and write delay and the total number of read and write at the key state and table level. We can also see more information about cassandra's internal structure of each table, including memtables, Bloom filters and SSTables. Summary This chapter describes ways to track and manage Cassandra clusters. In particular, we discussed JMX in detail and learned about the different functions Cassandra offers for MBean servers. We learned how to use JConsole and nodetol to see what's going on in the Cassandra cluster. You are now ready to learn how to perform routine maintenance tasks in the Cassandra cluster to make it healthy. You are now ready to learn how to perform routine maintenance tasks in the Cassandra cluster to make it healthy.