Designing Performance Monitoring Tool for NoSQL Cassandra Distributed Database

Prasanna Bagade, Ashish Chandra, Aditya B.Dhende
Pune Institute of Computer Technology
University of Pune
Pune

ABSTRACT: *The popularity of NoSQL databases (especially Cassandra) has been increasing day by day. Now, as many companies are developing Cassandra applications, they may need new tools to monitor database performance efficiently. Developers have difficulty optimizing something they can't see. When problems related to performance occur and proper analysis is needed, the statistical data generated by monitoring tool will be of a lot help. To optimize NoSQL applications, developers need to have an idea about how the database is behaving in different working scenarios. Cassandra is easy to configure, but for the proper performance tuning it is necessary to study the performance requirements for a particular application. This can be judged by monitoring tool. The paper describes the design of such monitoring tool and the results generated ie. statistics and graphs. The tool will be used primarily for low end machines as they are cost effective.*

Keywords: Distributed Databases, NoSQL Databases, Database Performance, Parameters of Performance

Received: 22 December 2011, Revised 10 February 2012, Accepted 17 February 2012

1. Introduction

Cassandra is NoSQL distributed database system which is known for managing large amount of distributed data. It provides high availability without single point of failure, the reason behind this is that it treats failure of node as norm rather than exception. It is also famous for high write throughput without harming read efficiency. As it is distributed database it replicates data to keep search latency small. Every keyspace when created, it is assigned a replication factor. Cassandra provides replication polices namely rack aware, rack unaware and data center aware [1].

The data model of Cassandra is column oriented, columns together form column family. Column family is nothing but collection of columns associated with the key. Column has a name, value and timestamp. Different rows in the same column family may not have same number/type of columns.

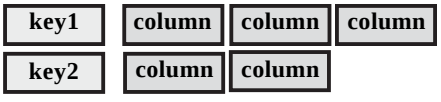

Figure 1. Data model of Cassandra -I

Super- column family is like column family within a column family. Every super column family is the collection of similar/related columns [3].

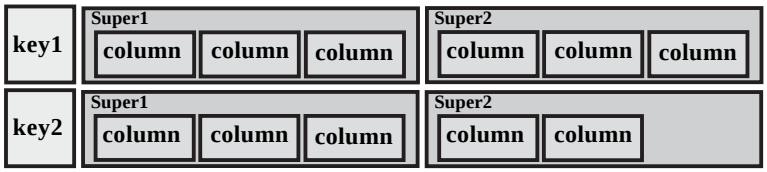

Figure 2. Data model of Cassandra -II

Cassandra dynamically partitions the data across the cluster and it provides different partitioning methods like random partitioner and order preserving partitioner. Cassandra is much easier to configure compared to other distributed database. It also allows fine performance tuning as per changing requirements. Mainly Cassandra system can be contains three layers – core layer, middle layer and top layer.The top layer is allows efficient, consistent reads and writes using a simple API. Cassandra provides simple queries insert, get & delete. The Cassandra API is made up of simple getter and setter methods and has no reference to the database distributed nature. Hinted hand-off is also the part of top layer. This occurs when a node goes down - the successor node becomes a coordinator and temporarily receives and stores write activities (hint) for downed node. When downed node becomes live, this information is given(handed) by coordinator node to live node.

The middle layer contains functions for handling the data that is being written into the database. Compaction is the process which tries to combine keys and columns to increase the performance of the system by freeing the memory. The different ways of storing data such as Memtable and SSTable are also handled here [3].

The core layer deals with the distributed nature of the database, and contains functions for communication between nodes, the state of the cluster as a whole (including failure detection) and replication between nodes [2].

Core	Middle	Top
Messaging service Failure detection Cluster state Partitioner Replication	Indexes Compaction Commit log Memtable SSTable	Hinted handoff Read repair Monitoring Admin tools

2. The Design

The nodetool utility in Cassandra allows to collect Cassandra performance statistics. Also commands like TOP, SAR are useful to collect stastistics. As there is built in support of performance counters that provides information about how system is doing. Recoding the information from these counters is very much necessary for troubleshooting in the development phase of application. The main performance parameters we should consider here are CPU utilization, Memory utilization, Thread Pool Statistics, Read Write counts & latencies for Column Family, Read Write counts & latencies for Keyspace. So here we have to write total five shell scripts to collect the statistics repeatedly after some interval of time and store it in the file. We have to write a program which will read the files and display statistics graphically.

2.1 GUI

The GUI consists of panel of buttons & some textboxes to choose which statistics user wants to see.

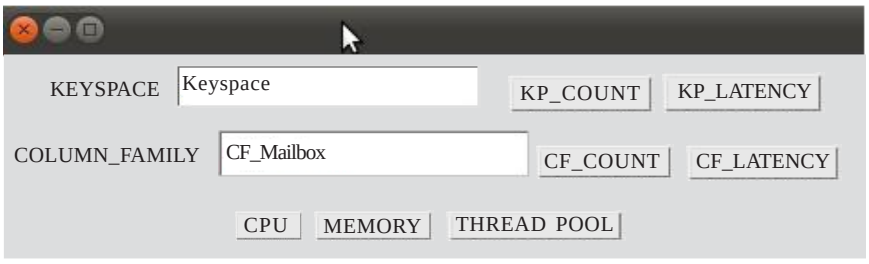

Figure 3. GUI

2.2 Thread Pool Statistics

Cassandra is implemented by Staged Event Driven Architecture (SEDA), therefore instead of spawning thread per request, requests are transferred between queues of bounded size called thread pools. If these pools are filled, requests get backlogged which effects in delays or exceptions. We can diagnose this problem by collecting thread pool statistics (tpstats). It gives the number of operations active, pending and completed at different stages. Any stage that has the non zero number is in the active or pending column is backlogged. A healthy system shows near zero at most times in all stages for both the active and pending stages. (Figure 4)

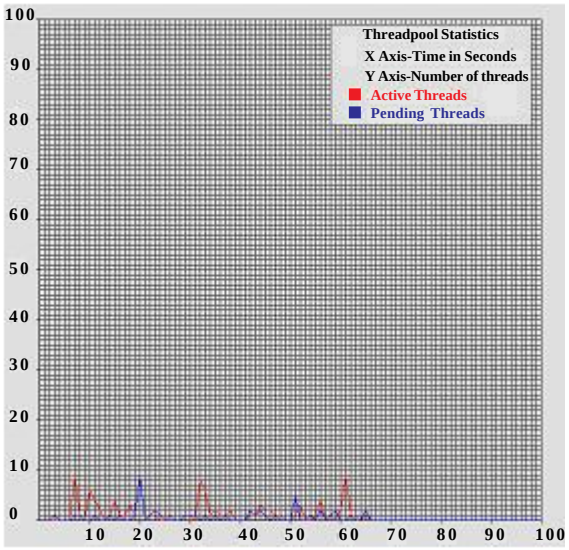

Figure 4. Threadpool Statistics

The graph shows total active & pending thread count including all the stages.

2.3 Column Family Statistics

Each column family has a number of performance counters that provide proper diagnostics. The cfstats (column family statistics) option shows a high level summary of the column family information. The main parameters are Read Count, Write Count & Read Latency, Write Latency. The graph will show Reads/Writes per second & Read/Write latency over time. The more requests on a given column family, the more system resources are being used. Use this information to: Correlate with Disk and CPU usage graphs to determine approximately how many operations per second a system can support, Track these values over time to monitor usage patterns. Latency is an important factor when serving data to clients.Cassandra tracks latency, it counts time the request is received to the time it is found or inserted to disk. During read operations, Cassandra tracks information on the cumulative time spent searching for data and makes this available as TotalReadLatencyMicros. Use this information to Ensure that latency stays within acceptable values & Compare this latency with results from yesterday or last week.

2.4 Count Graph of Column Family

The graph shows Figure 5.

2.5 Latency Graph of Column Family

The graph shows Figure 6.

2.6 Keyspace Statistics

The keyspace contains many Column Families. Hence the main parameters for keyspace are Read Count, Write Count & Read Latency, WriteLatency. They represent the values for whole keyspace.

2.7 CPU

The percentage CPU utilization by Cassandra process is obtained by Top command. This graph is used to check whether CPU is under load or not. (Figure 7)

2.8 Memory

The percentage memory utilization by Cassandra process is obtained by Top command. This graph is used to check whether memory is under load or not. (Figure 8)

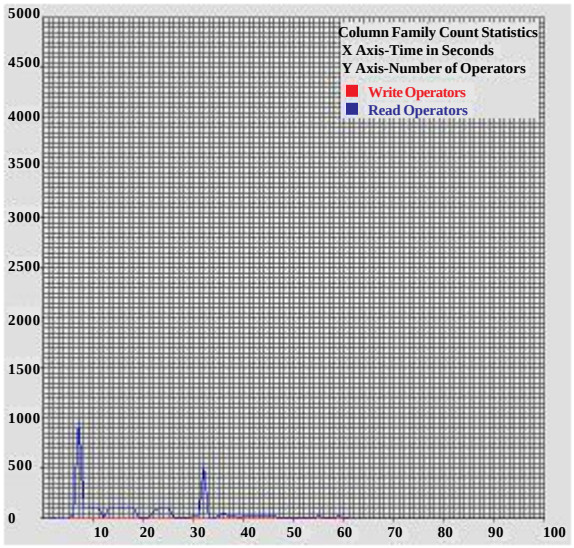

Figure 5. Count Graph of Column Family

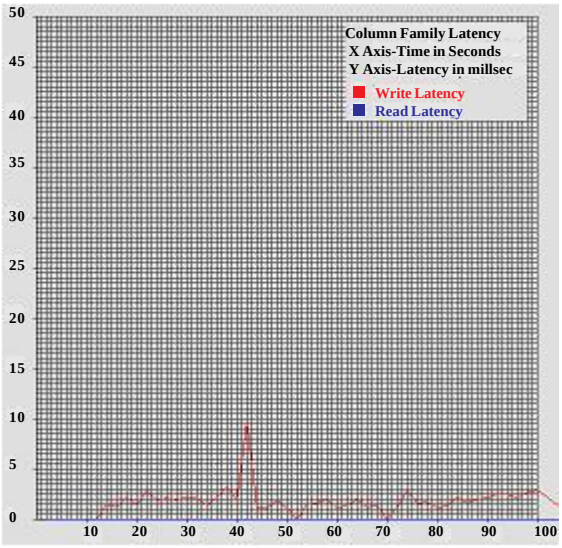

Figure 6. Latency Graph of Column Family

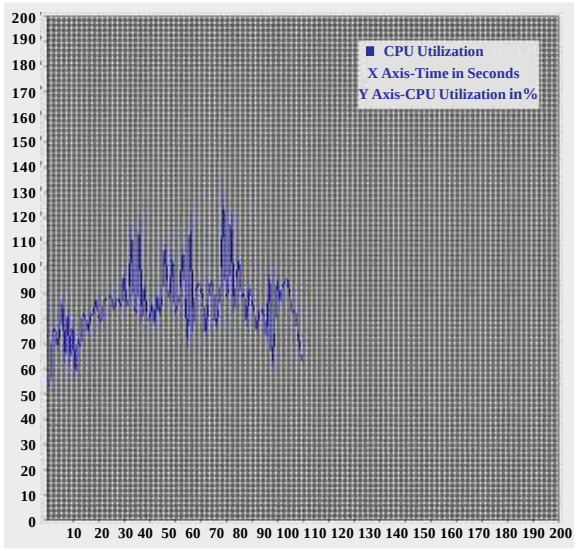

Figure 7. CPU utilization by Cassandra process

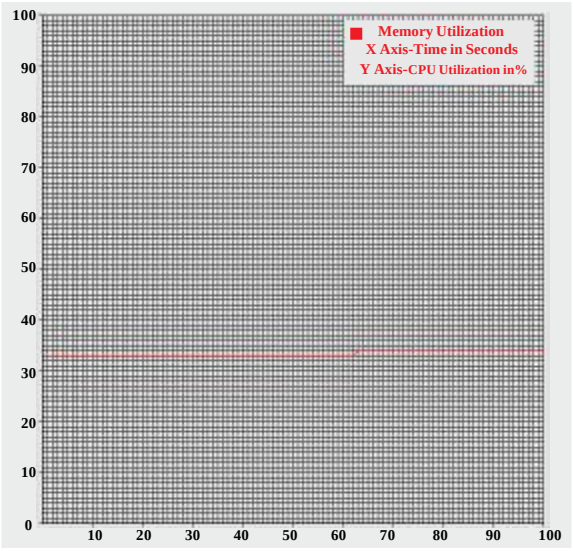

Figure 8. Memory utilization by Cassandra process

The low end machines are cost effective than high end machines. So many small application designing organizations will go for low end machines. Firstly it is necessary to create an application which will fire queries on database &whose query load can be adjustable as per requirement. Then the monitoring tool will be run on each node present in the cluster. The query load will be increased/decreased, the nodes will be added/removed and simultaneously the change in performance will be observed. Based on observations the user will be able to configure performance tuning parameters correctly. It is necessary to show the graph of performance parameters against time or transactions per second.

3. Mathematical Model

Let S be the system which collects the statistics and stores it.

$$S = \{F,R, I, CPU, MEM, CF, TP, CF, G\}$$

Where,

$$F = \{F1, F2, F3, F4, F5\}$$

This is the set of functions which collect & store statistics in respective result sets.

$$R = \{R1, R2, R3, R4, R5\}$$

I = time interval δt after which statistics is collected again and again.

CPU - percentage CPU utilization.

F1()- function Collects & Stores CPU in R1

MEM- Memory Utilization

F2() -function Collects & Stores M in R2

$$TP = \{St1, St2, \ldots\ldots\ldots\ldots, St14\}$$

TP is threadpool statistics where '*St*' means the '*stage*' of operation(as explained in previous section).

$$St = \{A, P, C\}$$

A, P & C are the number of tasks that are active, pending & completed respectively in a stage '*St*'

F3()-function Collects & Stores TP in R3

$$CF = \{KSP, CF1, CF2, \ldots\ldots\ldots, CFn\}$$

CF is column family statistics where KSP is Keyspace statistics &CFx stands for statistics of x'th column family in that keyspace.

KSP = {RC, RL, WC, WL}

RC - read count

RL - read latency

WC - write count

WL - write latency

CFx = {N, SS, MT, RW, CA, CO}

N - Name of column family

SS = {SSTC, SUL, SUT}

SSTC - SSTable Count

SUL - Space used(live)

SUT - Space used(total)

MT = {MTC, MTD, MTS}

MTC - Memtable column count

MTD - Memtable data size

MTS - Memtable switch count

RW = {RC, RL, WC, WL, PT}

RC - Read Count

RL - Read Latency

WC - Write Count

WL - Write Latency

PT- Pending Tasks

$$CA = \{KCC, KCS, KCHR, RC\}$$

KCS- Key Cache Size

KCHR- Key Cache Hit Rate

RC- Row Cache

$$CO = \{CMNS, CMXS, CMES\}$$

CMNS- Compacted Row minimum size

CMXS- Compacted Row Maximum Size

CMES- Compacted Row Mean Size

F4()-function Collects & Stores KSP in R4

F5()-function Collects & Stores RW in R4

Now, after every δt set of functions F, all functions can work simultaneously so as to produce result set R. So any x'th result in set R is integration of x'th function in set F over the time T1 to T2.

$$Rx = \int Fx()$$

$$G = \{G1, G2, G3, G4, G5\}$$

This is the set of graph drawing functions which take Result as input and work simultaneously to draw graphs.

G1 (R1) - draws CPU utilization with respect to time

G2 (R2) - draws memory utilization with respect to time

G3 (R3) - draws tpstats with respect to time

G4 (R4) - draws Keyspace related attributes with respect to time.

G5 (R4) - draws RW (read write attributes) with respect to time for each column family seperately.

4. Limitations

The system will fail in following scenarios

1) Network failure: The network failure stands for breakage or saturation of link between two nodes and if the node goes down due to power failure.

2) System Crash: The System may crash due to excessive load on a specific machine.

5. Future Work

The future work includes adding some more performance parameters to this tool. The limitations mentioned should be overcome. Also it will be good to design an intelligent system that will analyze the statistics generatedto suggest the most suitable settings as per the real time environment. The graphical user interface can be made efficient so that it will properly convey changes in performance.

6. Conclusion

NoSQL technologies enable enterprises to deliver rich application services to more users than ever before. So its necessary to design a performance monitoring tool which will help to make decisions to optimize performance as per different environments. In the paper we saw the design of the monitoring tool. With the graphs generated by monitoring tool we are able to get picture about performance of the database & this helps to make decisions about storage strategy of database.

References

[1] Prashant Malik, AvinashLakshman, (2009). *Cassandra – a decentralized structured storage system.* The 3rd ACM SIGOPS International Workshop on Large Scale DistributedSystems and Middleware (LADIS 09), October.

[2] Bingwei Wang, Si Peng, Xiaomeng Zhang, Mark Bownes, Rob Paton, FarshidGolkarihagh, (2010) *Cassandra as used by Facebook,* December 15.

[3] NabeelAhamedAkheel, *Cassandra.*

[4] DietrichFeatherston, *Cassandra: Principles and Application.*

[5] PrasannaBagade, Ashish Chandra, Aditya Dhende. *Performance Monitoring Tool for NoSQL Column Oriented Distributed Database (Cassandra).*