Recognizing Platform Limits

This section describes the potential limiting platform factors on cluster performance, how to measure whether a cluster is approaching or exceeding those limits, and options available to remedy such conditions. "Platform factors" refer to hardware resources such as CPU, memory, disk, and network I/O subsystems. For potential software-related factors, please see Managing Data Distribution, Load Balancing ClustrixDB with HAProxy, or Understanding the ClustrixDB Explain Output.

.

CPU Load

A common cause of overall degraded performance within ClustrixDB is CPU contention. In the ideal case, this occurs when a cluster reaches maximum TPS for a given workload with the current number of nodes: all CPU cores are busy, and additional load results in increased query latency. The solution here is to add more nodes to the cluster, providing additional compute, memory, and storage capacity.

CPU Imbalance

There are other cases where CPU contention becomes a bottleneck even though the cluster is not being fully utilized; that is, load is not optimally balanced across the cluster. This can be due to external factors such as an inefficient query, or client connections being poorly distributed across nodes (if not connecting through the VIP). A sub-optimal configuration could also be a culprit, such as having a table that is not distributed evenly across the cluster, although the system goes to great lengths to automatically manage this. .

Monitoring CPU Load

CPU load reflects the busyness of each node's CPU cores in the ClustrixDB cluster.

The SHOW LOAD command gives the current average load across all cores of all nodes (disregarding core 0, which is reserved for specific tasks). On a well-balanced system, the SHOW LOAD output gives a good indication of the current overall system utilization.

The table system.cpu_load provides a finer-grained view of CPU utilization, breaking out the individual CPU cores on each node. It shows both a load column, which is an instantaneous measure of current load, as well as total_busy, which counts up seconds of busy time (from database startup). When the CPU is 100% busy, load is 1, and total_busy increments by 1 each second. total_busy thus can provide a better measure of overall utilization.

The statistics gathering process statd (described in Monitoring Your Cluster Using statd) collects the system.cpu_load value and generates a delta reflecting load over the interval, in the statistic clustrix.cpu.load.node.N.cpu.N The instantaneous cpu minimum, average, and maximum are also collected from SHOW LOAD and stored in clustrix.cpu.load_min, clustrix.cpu.load_avg, and clustrix.cpu.load_max.

Querying the database statd will give you an indication of how your system has been performing over a period of time. Uneven node CPU utilization should be investigated, as an imbalanced load will lead to higher latency and lower throughput for a given cluster size. The most common cause of imbalanced load is a poorly distributed index (see Managing Data Distribution).

ClustrixGUI has a CPU Utilization graph that shows the min, max, and average CPU usage over all nodes for the past 24 hours and separately displays the instantaneous average CPU usage per node. Clusters using over 80% of their cpu usage would benefit from additional capacity and may experience higher latencies as a result of CPU starvation. See Expanding Your Cluster - Flex-Up.

Memory and Disk I/O

Buffer Manager Miss Rate

Buffer Manager Miss Rate, shown as bm_miss_rate in SHOW LOAD, indicates how often read operations miss the buffer cache and must instead go to disk. For a moderately loaded system with spinning disks, a bm_miss_rate over 2% may correlate with poor system performance. Persistent high bm_miss_rate indicates that the working set (rows of tables and indexes regularly accessed by your workload) exceeds the total buffer cache (memory) available across all nodes. This can result in higher query latency.

As described in Data Distribution-Cache Efficiency, cache is additive for ClustrixDB nodes. It is thus possible to reduce the bm_miss_rate for a given workload by adding more nodes to the cluster (data will need to be redistributed to the newly added nodes before this is effective). While incurring more downtime, it is of course also possible to add more memory to the existing nodes to increase total cache size, and thus reduce the bm_miss_rate.

bm_miss_rate may spike due to user queries accessing less common row data, for instance, some analytic query reading historical data not normally included in the working set, or a backup task such as mysqldump running.

Disk Latency and Throughput

The actual cost of buffer manager misses depends upon disk latency. For flash/SSDs, random read I/O is quite fast while spinning disk is relatively slow. Thus on an SSD system, bm_miss_rate may far exceed 2% without appreciable performance impact. Examining disk latency metrics in conjunction with b m_miss_rate can help pinpoint the cause of slowness to the disk I/O subsystem.

The following are the most useful metrics collected by statd related to disk latency and throughput:

- clustrix.io.vdev.read_latency_us.node.N.vdev.N and clustrix.io.vdev.write_latency_us.node.N.vdev.N indicate the response time for reads and writes to the vdev of each node. We are typically most concerned with vdev 2, which corresponds to the primary data store of each node. High latencies here, in conjunction with bm_miss_rate over 2%, will typically result in poor query response time (while CPU utilization remains relatively low).
- clustrix.io.disk.pct_utilization.node.N.disk.X provides a disk utilization metric for individual physical disks hosting the vdev file (e.g. through an md RAID device). It is calculated similarly to percent utilization of sar or iostat; the percentage of elapsed time that this device was servicing I/O, where a value nearing 100% indicates the disk is saturated. If any one disk shows significantly higher values than others, it may be performing poorly (for example, an old SSD which has gone through too many write cycles).
- clustrix.io.vdevs.bytes_read_per_sec and clustrix.io.vdevs.bytes_written_per_sec provide a cluster-wide measure of disk throughput from the database to the storage system. Significant increases in these values, coupled with an increase in query latency, may be indicative of an I/O bottleneck.

Using SHOW LOAD

SHOW LOAD provides a disk utility metric that is an average of the percent utilization (clustrix.io.disk.pct_utilization.node.N.disk.X, described above) over all disks in all nodes.

Also, note that SHOW LOAD reflects read and write activity over the last 15 seconds. Writes are typically buffered on each node and then written in periodic checkpoints, so regular spikes are to be expected. If write load is consistently high, however, this indicates that checkpoints are not flushing all writes before the next one begins, and this could indicate a write saturation condition which should be investigated by Clustrix Support.

Using sar

To more deeply investigate disk I/O subsystem performance, you can use a tool such as sar. The statd metrics noted above expose much of the same information, however, sar easily allows more frequent polling of this information.

sar -b will provide a global view of reads and writes of buffers from and to all disks in the system. It gives a gross indicator of disk utilization on a per-node basis.

```
root@ip-10-76-3-87:~$ sar -b 5
Linux 2.6.32-358.14.1.el6.x86_64 (ip-10-76-3-87)    09/25/2013   _x86_64_    (4 CPU)

07:06:13 PM      tps    rtps     wtps   bread/s   bwrtn/s
07:06:18 PM  3143.40  374.40  2769.00  22281.60  19230.40
07:06:23 PM  3861.28  671.86  3189.42  41255.09  22692.22
07:06:28 PM  2556.43  375.10  2181.33  22207.23  14547.79
07:06:33 PM  3208.38  526.15  2682.24  32175.65  15326.15
07:06:38 PM  2202.00  502.00  1700.00  31121.76   9654.29
07:06:43 PM  2572.40  402.20  2170.20  24441.60  17152.00
07:06:48 PM  1290.18  285.37  1004.81  17590.38   5861.32
07:06:53 PM  3287.82  553.69  2734.13  34430.34  20011.18
```

These numbers by themselves are not immediately useful, as one needs to understand the baseline performance of the disk subsystem of the particular platform.

sar -d -p will provide a number of metrics for each storage device on the system, some of which are immediately useful:

```
root@ip-10-76-3-87:~$ sar -d -p 5
Linux 2.6.32-358.14.1.el6.x86_64 (ip-10-76-3-87)    09/25/2013   _x86_64_    (4 CPU)
07:09:37 PM    DEV      tps  rd_sec/s  wr_sec/s avgrq-sz avgqu-sz   await  svctm   %util
07:09:42 PM xvdap1     0.00      0.00      0.00     0.00     0.00    0.00   0.00    0.00
07:09:42 PM   xvdb   421.69   4820.88   3129.32    18.85     1.06    2.52   1.41   59.32
07:09:42 PM   xvdc   391.97   4473.90   2923.69    18.87     0.90    2.31   1.37   53.88
07:09:42 PM   xvdd   519.28   4986.35   3868.27    17.05     1.02    1.96   1.12   58.13
07:09:42 PM   xvde   453.21   4268.27   3529.32    17.21     0.81    1.80   1.08   49.10
07:09:42 PM    md0  3112.65  18562.25  13452.21    10.29     0.00    0.00   0.00    0.00

07:09:42 PM    DEV      tps  rd_sec/s  wr_sec/s avgrq-sz avgqu-sz   await   svctm   %util
07:09:47 PM xvdap1     0.20      3.19      0.00    16.00     0.00    7.00    7.00    0.14
07:09:47 PM   xvdb   470.86   4804.79   3164.87    16.93     1.04    2.22    1.27   59.72
07:09:47 PM   xvdc   518.56   4502.99   3580.04    15.59     0.86    1.65    0.99   51.28
07:09:47 PM   xvdd   373.45   4534.93   2420.76    18.63     0.91    2.44    1.38   51.68
07:09:47 PM   xvde   348.70   5148.10   2146.11    20.92     0.95    2.73    1.54   53.71
07:09:47 PM    md0  2998.60  19003.59  11310.18    10.11     0.00    0.00    0.00    0.00
```

Of particular interest here are the average queue size (avgqu-sz) and utilization level (%util). If queue size is regularly greater than 2, or utilization exceeds 75%, it is likely that the workload is bottlenecked on disk I/O. These numbers should be useful even without having first established a performance baseline (as is the case with sar -b).

Search "linux sar" for more information on running and interpreting sar output. Note that iostat can also be used to provide similar information (both sar and iostat are based on information collected by the kernel in /proc/diskstats).

Network Throughput and Latency

Databases are not typically network-bound (as compared to a file server), however, a clustered database system does rely upon low latency links between nodes. For most workloads, communication between nodes does not consume large amounts of bandwidth, however, high message rates are possible; the OS TCP layer typically does a good job of avoiding network congestion. However, problems may arise where the same interface is servicing both large numbers of client connections as well as internode traffic, especially if there is a virtualization layer involved doing some software switching; in benchmark testing we have seen such factors provide an effective limit on transaction throughput, while standard throughput tests (MB/s) imply plenty of bandwidth is available.

Below we discuss two methods to assess whether the network presents a performance bottleneck.

Internode Latency

The virtual relation system.internode_latency shows the round-trip latency for communication between database processes on each node.

```
sql> select * from internode_latency order by 1,2;
+--------+----------+------------+
| nodeid | dest_nid | latency_ms |
+--------+----------+------------+
|   1    |    1     |    0.05    |
|   1    |    3     |   0.313    |
|   1    |    4     |   0.419    |
|   3    |    1     |   0.329    |
|   3    |    3     |   0.081    |
|   3    |    4     |   0.415    |
|   4    |    1     |   0.495    |
|   4    |    3     |   0.421    |
|   4    |    4     |   0.083    |
+--------+----------+------------+
9 rows in set (0.00 sec)
```

Note that this does include time for the database process to receive and respond to the ping, so to test just network latency, run the test on an idle cluster. But this also means an overloaded database process will typically result in higher reported internode latency times, so you can use internode_laten cy on a busy system to detect a generally underperforming node. In this case, you would typically see a pattern where one node is reported as having higher latency from other nodes while latency is low from that node to other nodes:

```
sql> select * from internode_latency order by 1,2;
+--------+----------+------------+
| nodeid | dest_nid | latency_ms |
+--------+----------+------------+
|   1    |    1     |   0.051    |
|   1    |    3     |   0.285    |
|   1    |    4     |   10.888   | <<==
|   3    |    1     |   0.425    |
|   3    |    3     |   0.057    |
|   3    |    4     |   8.818    | <<==
|   4    |    1     |   0.487    |
|   4    |    3     |   0.457    |
|   4    |    4     |   0.156    |
+--------+----------+------------+
9 rows in set (0.01 sec)
```

Note that the condition above was forced by running multiple CPU intensive processes from the linux shell on the node (in this case, gzip).

Network Throughput

Network statistics are collected by statd and provided as clustrix.io.network.*.node.*.if.*. Examples of the most useful of these are:

- clustrix.io.network.rx_bytes.node.1.if.eth0
- clustrix.io.network.rx_packets.node.1.if.eth0
- clustrix.io.network.tx_bytes.node.1.if.eth0
- clustrix.io.network.tx_packets.node.1.if.eth0

These are raw counters. A delta can be generated by a third-party monitoring tool such as Cacti, Nagios, or Zabbix.

Third-party tools are also available to monitor bandwidth utilization, such as bwm-ng. These can be particularly useful for real-time monitoring during high-bandwidth workloads such as backup, restore, or parallel data ingest from multiple clients.

It is not typical for network bandwidth to be a concern for ClustrixDB. We have only observed problems with high-concurrency benchmarks running in virtualized environments encountering a packet per second limitation of the VM platform.