Monitoring Your Cluster Using statd

A Xpand instance obtains statistics using statd (/opt/clustrix/bin/statd.py), a cluster-wide statistics aggregation and tracking mechanism. The resulting statistics are used for the graphical displays within the XpandGUI Administration UI and may also be queried directly using SQL.

The statd script runs on each node but only active instance is on the node with the highest node ID and collects statistics from internal system tables and populates the tables in the clustrix_statd database:

- hotness_current: table and index read and write stats collected every 5 minutes
- hotness_history: table and index read and write stats aggregated into 5 minute stats with stats older than one day rolled into hourly stats, kept for 7 days
- qpc_current: query stats collected every minute
- qpc_history: query stats aggregated into 5 minute stats, with stats older than one day rolled into hourly stats, kept for 7 days
- statd_current: cluster stats collected every 30 seconds
- statd_history: cluster stats aggregated into 5 minute stats, with stats older than one day rolled into hourly stats, kept for 7 days.

The frequency of statistics gathering is tunable via the statd_config table (see below), but we recommend to leave them with their default values.

In addition, the statd_metadata table contains the current and historical statd IDs and names that can be used to join back to the `statd_history` data.

Querying statd Statistics

Sample statd_current query

```
sql> SELECT name, value
     FROM clustrix_statd.statd_current
     NATURAL JOIN clustrix_statd.statd_metadata
     WHERE name LIKE '%nodes_in_quorum%'
     ORDER BY name;
```

Sample statd_history query

```
sql> SELECT timestamp, value, name
     FROM clustrix_statd.statd_history
     NATURAL JOIN clustrix_statd.statd_metadata
     WHERE name LIKE 'clustrix.cpu.load_avg%'
       AND timestamp BETWEEN '2019-01-01 09:30:01'
       AND '2019-05-15 10:30:01'
     ORDER BY timestamp, name;
```

Sample qpc_history query

The query below looks at queries that have high average rows read between the specified timestamps as queries that read many rows may be candidates for optimizing.

```
sql>  SELECT sum(exec_count) as exec_count,
       round(avg(avg_rows_read)) as avg_rows_read,
       sum(rows_read) as rows_read,
       round(avg(avg_exec_ms)) as avg_exec_ms,
       left(statement,60)
     FROM clustrix_statd.qpc_history
     WHERE timestamp BETWEEN '2019-06-01 00:00:00'
     AND '2019-06-01 23:59:59'
     GROUP BY 5
 ORDER by 2 desc
     LIMIT 20;
```

Sample hotness_history query

The query below looks at tables and indexes that have the most reads over the last 24 hours:

```
sql> SELECT Sum(reads) AS reads,
       Sum(writes) AS writes,
       DATABASE,
       `table`,
       `index`,
       total_replicas
     FROM clustrix_statd.hotness_history
     WHERE timestamp >now() - INTERVAL 1 day
     AND DATABASE NOT IN ('clustrix_statd', '_replication', 'clustrix_ui', 'system', 'clustrix_dbi')
     GROUP BY `table`, `index`
     ORDER BY 1 desc
     LIMIT 50;
```

Useful Statistics

There are numerous statistics collected for your cluster. Use the following query to determine all available statd statistics:

```
sql> SELECT name
     FROM clustrix_statd.statd_metadata
     ORDER BY name;
```

The following tables highlight useful statistics. Additional stats can be found at statd Metrics.

Performance

Statistic	Notes
clustrix.qps clustrix.tps	The statistics for both queries per second (qps) and transactions per second (tps) will be the same if explicit transactions were not used.
clustrix.cpu.load_avg.node.<node#> clustrix.cpu.load_max clustrix.cpu.load_min	Indicates average CPU load across all CPU cores per node: and load of busiest and least busy CPU core in the cluster. Note that core 0 is reserved for certain activities and thus will often reflect a lower CPU utilization relative to other cores, which will intern affect these averages. To see more types of cpu stats available: SELECT name, value FROM statd_current NATURAL JOIN statd_metadata WHERE name LIKE '%clustrix.cpu%' ORDER BY name;
clustrix.stats.Com_delete clustrix.stats.Com_insert clustrix.stats.Com_update clustrix.stats.Com_select clustrix.stats.Com_alter_<type>	Provide execution counters for each operation. Provide the total time spent executing each operation. Taking the delta of Com_<operation>_us and dividing by the delta of Com_<operation> will thus provide the average response time for that delta. To see more types of Com stats available for operations: SELECT name, value FROM statd_current NATURAL JOIN statd_metadata WHERE name LIKE '%clustrix.stats.Com%' ORDER BY name;
clustrix.stats.executing_sessions	The statistics for number of sessions executing.

Replication

Statistic	Notes
clustrix.replication.slave.relay_log_bytes.<slave name>	Size of the slave relay log in bytes.
clustrix.replication.slave.seconds_behind_master.<slave name>	Seconds behind master.

Data Protection

Statistic	Notes
clustrix.cluster.nodes_in_quorum	Nodes participating in current group.
clustrix.cluster.total_nodes	All the nodes the cluster knows about, including down/unavailable.

clustrix.rebalancer.rebalancer_reprotects clustrix.rebalancer.rebalancer_deletes clustrix.rebalancer.rebalancer_rebalance clustrix.rebalancer.rebalancer_redistributes clustrix.rebalancer.rebalancer_request clustrix.rebalancer.rebalancer_reranks clustrix.rebalancer.rebalancer_splits	Number of Rebalancer actions in progress for a given type.

Cluster Capacity

The following statistics are kept as percentages and are thus in the range of 0-100

Statistic	Notes
clustrix.capacity.disks.avg_used_percent	Average percentage of permanent space allocated across the disks in the system.
clustrix.capacity.disks.max_used_percent	Highest percentage of permanent space allocated across all the disks in the system.
clustrix.capacity.disks.min_used_percent	Lowest percentage of permanent space allocated across all the disks in the system.

Tuning the Frequency of statd Collection

Increasing frequency of statd collection can impact the performance of your cluster.

The frequency of statistics collection is configurable within the statd_config table. Contact Xpand Support for assistance in modifying these values.

```
sql> select * from statd_config order by name;
+--------------------------------------+-------+
| name                                 | value |
+--------------------------------------+-------+
| qpc_current_interval_s               |    60 |
| qpc_history_interval_s               |   300 |
| qpc_history_top_n                    |   100 |
| qpc_rollup_interval_s                |  3600 |
| repartition_history_interval_s       | 43200 |
| SpaceUsedBreakdown_current_interval_s |   900 |
| stats_current_interval_s             |    30 |
| stats_history_interval_s             |   300 |
+--------------------------------------+-------+
8 rows in set (0.00 sec)
```

Data Retention

The hotness_history, qpc_history and statd_history tables are partition tables and data older than 7 days are dropped. If you need to longer data retention, you can have a cron job to extract the data accordingly and store it elsewhere.

Third Party Plugins

See also information for using:

- Nagios plugin
- Cacti plugin
- Zabbix
- Grafana