Proposals in Bottlenecks

Yu Yang

20160824
Agenda

• Bottlenecks Colorado Discuss
  • Colorado Release Report
    • Colorado Stable Branch
    • Release Meeting Report
  • Bottlenecks Colorado Testing Framework (Rubbos example)
• Proposals in Bottlenecks (Draft)
  • Goals and Scope (Draft)
  • The Proposals (Draft)
  • Metrics & Tools Discussion (Draft)
  • Work Plan Discussion for the proposal (Draft)
Bottlenecks Colorado Discussion

• August 15-22, Milestone window period
• MS9: Stable branch created
  – Projects are branched from main
  – Commits are limited to critical issues
  – Commits must be cherry-picked
• MS10: Documentation completed
  – Updated
  – Reviewed
  – Verified
  – Committed to repo
  – Note: DOES NOT include test results (see "Formal test execution completed" below)
Bottlenecks Colorado Discussion

- Colorado Release Meeting 0816
  - Discussion of Release Meeting
    - stable branch
    - documentation
    - scenario status
    - Colorado 2.0 and 3.0 milestone planning
Bottlenecks Colorado Discussion

- Bottlenecks
  - config
  - docs
  - puppet_manifests
  - rubbos
  - run_rubbos_internal.sh
- run_rubbos.py
- run_rubbos.sh

Jump Server

Rubbos Controller

Git Repo

Artifacts Repo

DB for dashboard

dashboard

DUT

client

......

Web Server

Application Server

Database Server

NFV Infrastructure

I : vm ready
II : run_rubbos_internal.sh
Goals and Scope (Draft)

• Goals
  • Enhance interaction with other project
  • Feedback development suggestions to upstream
  • Improve the performance of OPNFV reference platform

• Scope
  • OPNFV Testing projects
  • OPNFV Feature projects
  • Modeling (Profile the testing behaviors), Testing and Data analysis
  • Parameters choosing and Algorithms
Proposals in Bottlenecks (Draft)

1. Classified bottlenecks
2. Feedback bottlenecks
3. Upstream Develop
4. Performance Improvement

Bottlenecks

Test Cases
- Network
- Storage
- Compute
- Middleware
- APP

Bottlenecks Testing Results

Yardstick

DB

OPNFV Reference Platform
Metrics & Tools Discussion (Draft)

• Target
  – Metrics Set for Specific Bottlenecks
  – Feature testing results could be organized into different metrics sets to find the bottlenecks

• Monitoring
  – Compute: latency, utilization of CPU, cache size, etc.
  – Network: throughput, number of connection, packet delay, etc.
  – Storage: memory available mbytes, pages/sec, idle time, etc.
  – Midware: concurrent request, response speed, throughput, etc.
  – APP: scale in/out, scale up/down, throughput, latency, etc.
# Metrics from Yardstick

<table>
<thead>
<tr>
<th>Performance/Speed</th>
<th>Capacity/Scale</th>
<th>Reliability/Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency for random memory access</td>
<td>Number of cores and threads</td>
<td>Processor availability (Error free processing time)</td>
</tr>
<tr>
<td>Latency for cache read/write operations</td>
<td>Available memory size</td>
<td>Memory availability (Error free memory time)</td>
</tr>
<tr>
<td>Processing speed (instructions per second)</td>
<td>Cache size</td>
<td>Processor mean-time-to-failure</td>
</tr>
<tr>
<td>Throughput for random memory access (bytes per second)</td>
<td>Processor utilization (max, average, standard deviation)</td>
<td>Memory mean-time-to-failure</td>
</tr>
<tr>
<td></td>
<td>Memory utilization (max, average, standard deviation)</td>
<td>Number of processing faults per second</td>
</tr>
<tr>
<td></td>
<td>Cache utilization (max, average, standard deviation)</td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput per NFVI node (frames/byte per second)</td>
<td>Number of connections</td>
<td>NIC availability (Error free connection time)</td>
</tr>
<tr>
<td>Throughput provided to a VM (frames/byte per second)</td>
<td>Number of frames sent/received</td>
<td>Link availability (Error free transmission time)</td>
</tr>
<tr>
<td>Latency per traffic flow</td>
<td>Maximum throughput between VMs (frames/byte per second)</td>
<td>NIC mean-time-to-failure</td>
</tr>
<tr>
<td>Latency between VMs</td>
<td>Maximum throughput between NFVI nodes (frames/byte per second)</td>
<td>Network timeout duration due to link failure</td>
</tr>
<tr>
<td>Latency between NFVI nodes</td>
<td>Network utilization (max, average, standard deviation)</td>
<td>Frame loss rate</td>
</tr>
<tr>
<td>Packet delay variation (jitter) between VMs</td>
<td>Number of traffic flows</td>
<td></td>
</tr>
<tr>
<td>Packet delay variation (jitter) between NFVI nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential read/write IOPS</td>
<td>Storage/Disk size</td>
<td>Disk availability (Error free disk access time)</td>
</tr>
<tr>
<td>Random read/write IOPS</td>
<td>Capacity allocation (block-based, object-based)</td>
<td>Disk mean-time-to-failure</td>
</tr>
<tr>
<td>Latency for storage read/write operations</td>
<td>Block size</td>
<td>Number of failed storage read/write operations per second</td>
</tr>
<tr>
<td>Throughput for storage read/write operations</td>
<td>Maximum sequential read/write IOPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum random read/write IOPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disk utilization (max, average, standard deviation)</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Bottlenecks</td>
<td>Metrics Set</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| M&T List |                          | https://cloud.google.com/monitoring/api/metrics  
| Compute | Short of Processor | (System\%Total processor time, Processor \%Processor Time, system\Processor Queue Length) | Metrics 2 is for SQL Server  
PPT is to avoid memory shortage  
SPQL is to trace LB of processors |
|          | latency       | reponse time | Metrics 1 is for web server  
Metrics 2 is for |
|          | throughput    | (reponse time, %package loss) | where the network congestion occur and throughput reaches it bottleneck |
| Storage  | Short of Memory (Memory Available MBytes) | (Page Reads/Sec, Page/Sec) | PS is not necessarily lack of memory when it is high, maybe an application sequentially reading a memory mapped file |
|          | memory leak (Memory Available MBytes) | (%Disk Time, Page Reads/Sec, Avg.Disk Queue Length) | Short of memory will cause using Disk Cache |
|          | I/O           | (PhysicalDisk/%Disk time, PhysicalDisk/%Idle Time, Physical Disk\ Avg.Disk Queue Length, Disk sec/Transfer) | Only DT is high, then Disk is not the bottlenecks. PRS is to avoid memory shortage |

More are planed to develop and under Discussion
Work Plan Discussion for the proposal (Draft)

- Adding testing suite to Bottlenecks projects
  - Jenkins job and proposed test suite
  - Code structure in the Bottlenecks repo
- Determine metrics set and tools for the initial setup
  - Compute: Short of Processor
  - Network: bandwidth, latency and throughput
  - Storage: Short of Memory, memory leak, I/O
Work Plan Discussion for the proposal (Draft)
Some Storage Metrics

- **Capacity utilisation**: in terms of percent/GB of space used, as well as subcategories such as raw, formatted, free, allocated or allocated not used
- I/O per second (IOPS)
- Bandwidth
- **Latency**
- Access time: read, write, random
- **Energy usage**: from macro (subsystem) to micro (device or component)
- **Mean time between failure (MTBF)**
- Mean time to repair or replace (MTTR) failed subsystems/components
- **Recovery point objective (RPO)**: The point in time to which you want data restored
- **Recovery time objective (RTO)**: The time period in which data to the point required by the RPO must be restored.