Scale-out Storage Architectures in the NVM Era

“Evolution or Revolution?”

Sandeep Uttamchandani
Chief Architect, Cloud Storage
VMWare Inc.
Agenda: 3 WHYs

- **Why** Enterprise Storage → Scale-out Architectures
- **Why** NVM ≠ <Yet Another Storage Tier>
- **Why** New Scale-out Design ≠ Clean slate
Exponential Data Growth in Enterprises

“There were 5 Exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days.”

Eric Schmidt, Google 2010 Convention
Enterprise Storage Evolution

- Direct Attached Storage (mid-1980's)
- Networked Storage (mid-1990's)
- Scale-out Storage (early 2000's)
- 2015+
Scale-out Storage Evolution

- Lustre
- IBM GPFS
- VMWare VMFS
- Veritas CFS
- ...

- Cassandra
- HDFS
- MongoDB
- HBase
- ...

- VMWare VSAN
- Ceph
- GlusterFS
- Swift
- ...

- Redis Cluster
- Stanford RamCloud
- Spark/Tachyon
- VoltDB/H-Store
- ...

HPC  Big Data/NoSQL  General Purpose Scale-out  In-memory Scale-out  NVM-tiered
Scale-out Architectures: End of one-size-fits-all
Agenda: 3 WHYs

- Why Enterprise Storage $\rightarrow$ Scale-out Architectures

- Why NVM $\neq$ <Yet Another Storage Tier>

- Why New Scale-out Design $\neq$ Clean slate
Scale-out Design: Object Lookup

Client \[\rightarrow\] Namespaces Manager \[\rightarrow\] Object Manager

Object \[\rightarrow\] \(<\text{Node, File}>^+\)

Directory Lookup Design Pattern
Scale-out Design: Object Lookup

Range-based Lookup Design Pattern
Scale-out Design: Object Lookup

Does this deliver the true value of NVM?

Range-based Lookup Pattern
<table>
<thead>
<tr>
<th>Year</th>
<th>NW RTT</th>
<th>Best IO Latency</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>2.54 ms*</td>
<td>60-80 ms</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>150-300 µs</td>
<td>4-6 ms</td>
<td></td>
</tr>
<tr>
<td>2016+</td>
<td>1-2 µs</td>
<td>100s of ns</td>
<td></td>
</tr>
</tbody>
</table>

*Network is the new bottleneck*

*S. Rumble, et. al. Its time for low Latency. HotOS 2011*
## Namespace Sharding

Object Lookup: ★ Client-aware  ✪ Client-opaque

<table>
<thead>
<tr>
<th>Directory Lookup</th>
<th>GPFS Lustre</th>
<th>HDFS MongoDB</th>
<th>EMC OneFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-based Range Routing</td>
<td>Cassandra Riak</td>
<td>Swift</td>
<td>Redis</td>
</tr>
<tr>
<td>Lookup + Key-based Routing</td>
<td>HBase BigTable</td>
<td>Ceph</td>
<td>RamCloud</td>
</tr>
</tbody>
</table>

Shared Disk  | Big Data/NoSQL  | General Purpose Scale-out  | In-mem Scale-out  | NVM-tiered (Future)
## Namespace Sharding

### Object Lookup:

- **Client-aware**
- **Client-opaque**

### Directory Lookup

- **GPFS**
- **Lustre**

### Key-based Range Routing

- **Cassandra**
- **Riak**
- **Swift**
- **Redis**

### Lookup + Range-based

- **HBase**
- **BigTable**
- **Ceph**
- **RamCloud**

### Shared Disk vs. Big Data/NoSQL vs. General Purpose Scale-out vs. In-mem vs. NVM-tiered (Future)
### Holistic Shifts in Design Constraints

**Application**
- POSIX/ACID Batch-oriented
- Non-POSIX/BASE Real-time

**CPU**
- 1x10MHz
- 16x3GHz

**Memory**
- < 2 MB
- >16GB

**Network**
- 3Mbps
- 10Gbps

**Storage**
- <30MB
- >4TB

- **Beyond traditional block/file POSIX applications**
- **Slow-down of Moore’s law**
- **Bigger & Cheaper**
- **Network is becoming the new latency bottleneck**
- **Emergence of distinct Capacity and Performance Storage Tiers**

**Avg. Node Configuration**

<table>
<thead>
<tr>
<th></th>
<th>1984</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1x10MHz</td>
<td>16x3GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>&lt; 2 MB</td>
<td>&gt;16GB</td>
</tr>
<tr>
<td>Network</td>
<td>3Mbps</td>
<td>10Gbps</td>
</tr>
<tr>
<td>Storage</td>
<td>&lt;30MB</td>
<td>&gt;4TB</td>
</tr>
</tbody>
</table>
The New Storage Hierarchy

mid-1980’s

2015+

Closing the Latency Gap
**Bigger and Cheaper**

**Maximum DRAM in GB**

- **Source:** Forrester Research, *The x86 Server Grows Up And Out* (October 8, 2010)

**DRAM**

- **$/GB**
  - $100
  - $75
  - $50
  - $25
  - $0

- **Source:** Gartner Dataquest, *Forecast: DRAM Market Statistics* (1Q11)
Latency lags Bandwidth*

<table>
<thead>
<tr>
<th></th>
<th>Mid-1980s</th>
<th>2009</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk capacity</td>
<td>30 MB</td>
<td>500 GB</td>
<td>16667x</td>
</tr>
<tr>
<td>Max. transfer rate</td>
<td>2 MB/s</td>
<td>100 MB/s</td>
<td>50x</td>
</tr>
<tr>
<td>Latency (seek &amp; rotate)</td>
<td>20 ms</td>
<td>10 ms</td>
<td>2x</td>
</tr>
</tbody>
</table>

Source: Stanford RamCloud Talk, Feb 2010

*David Patterson, Latency Lags Bandwidth, CACM, 2004
CPU Scaling: Slow-down of Moore’s Law

- CPU Scaling is not exponential anymore
  - 3000% increase from 1995-2004
  - 30-40% increase since 2004

- Multi-core scaling is linear
  - NUMA, locking, sharing latencies, programming models…

- Compute per unit of data is decreasing
Agenda: 3 WHYs

- Why Enterprise Storage → Scale-out Architectures
- Why NVM ≠ <Yet Another Storage Tier>
- Why New Scale-out Design ≠ Clean slate Re-design
What does re-design mean?

Scale-out Storage Architecture

Design of a collection of Micro-Services
## Concurrency Control

<table>
<thead>
<tr>
<th>Pessimistic</th>
<th>Optimistic w/ retry</th>
<th>Optimistic w/ heuristics</th>
<th>Shared Disk</th>
<th>Big Data/NoSQL</th>
<th>General Purpose Scale-out</th>
<th>In-mem</th>
<th>NVM-tiered (Future)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPFS (Distributed LM) VMFS</td>
<td>HDFS</td>
<td>Ceph</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra (CAS) CouchDB (MVCC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra (Merged Updates)</td>
<td>Swift</td>
<td>Redis (CAS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redis (LWW)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
High Availability

Data Replication: ■ State-based ○ Ops-based ◯ Lineage

Active-Active
- CouchDB ■

Active-Passive
- Lustre (NA)
- GPFS (NA)
- Ceph ■
- Swift ■
- Redis ○

Active-Backup
- Cassandra ■
- HBase ■
- RamCloud ■
- Tachyon ◯

Shared Disk
Big Data/NoSQL
General Purpose Scale-out
In-mem
NVM-tiered (Future)
<table>
<thead>
<tr>
<th>In-place</th>
<th>Out-of-place</th>
<th>Versioned</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPFS</td>
<td>HDFS</td>
<td>Cassandra</td>
</tr>
<tr>
<td>Lustre</td>
<td>CouchDB</td>
<td>HBase</td>
</tr>
<tr>
<td>VMFS</td>
<td>Ceph</td>
<td>Swift</td>
</tr>
<tr>
<td></td>
<td>VMware VSAN</td>
<td>Scality</td>
</tr>
<tr>
<td></td>
<td>RamCloud</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NVM-tiered (Future)</td>
</tr>
</tbody>
</table>
Summary

- Why Enterprise Storage $\rightarrow$ Scale-out Architectures
  - Web 2.0 model to handle exponential data growth

- Why NVM $\neq$ <Yet Another Storage Tier>
  - Holistic shifts across compute, network, memory, storage

- Why New Scale-out Design $\neq$ Clean slate
  - Piecemeal evolution of micro-services within a Scale-out Architecture.
Questions?

Sandeep Uttamchandani  
suttamchandani@vmware.com