Analytics in the cloud

Dow we really need to reinvent the storage stack?

R. Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, Renu Tewari
Data-Intensive Internet Scale Applications

Typical Applications

- Web-scale search, indexing, mining
- Genomic sequencing
- brain-scale network simulations
Data-Intensive Internet Scale Applications

- **Key Requirements**
  - Scale to very large data sets
  - Platform needs to scale to 1000’s of nodes
  - Built of commodity hardware for cost efficiency
  - Tolerate failures during “every” job execution
  - Support data shipping to reduce network requirements
MapReduce for analytics

- MapReduce is emerging as a model for large-scale analytics application
- Important design goals are extreme-scalability and fault-tolerance
- Storage layer is separated and has well-defined requirements

MapReduce Data-store requirements

- Provide a hierarchical namespace with directories and files
- Allow applications to read/write data to files
- Protect data availability and reliability in the face of node and disk failures
- Provide high bandwidth access to reasonably-sized chunks of data to all compute nodes (not necessarily all-to-all)
- Provide chunk access-affinity information to allow proper scheduling of tasks
## Data store options: Cluster FS Vs Specialized FS

<table>
<thead>
<tr>
<th></th>
<th>Specialized FS</th>
<th>Cluster FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Commodity hardware compliant</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Data store options: Cluster FS Vs Specialized FS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specialized FS</th>
<th>Cluster FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Commodity hardware compliant</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Traditional application support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mature management tools</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Data store options: Cluster FS Vs Specialized FS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specialized FS</th>
<th>Cluster FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Commodity hardware compliant</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Traditional application support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mature management tools</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Tuned for Hadoop</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Modifying a Cluster Filesystem for MapReduce

- **GPFS**
  - Mature filesystem - many large production installations
  - High performance, Highly scalable
  - Reliability features focused on SAN environments
    - Supports rack-aware 2-way replication
  - POSIX interface
  - Supports shared disk (SAN) and shared-nothing setups
  - Not optimized for MapReduce workloads
    - Does not expose data location information
    - largest block size = 16 MB

- **Changes for Hadoop:**
  - Make blocks bigger
  - Let the platform know where the big blocks are
  - Optimize replication and placement to reduce network usage
Key change: Metablocks

- Works for many workloads
  - Small FS blocks (eg: 512K)
  - Large Application blocks (eg: 64M)

- New allocation scheme
  - Metablock size granularity for wide striping

- Block map operates on large Metablock size

- All FS operations operate on small regular block size

- Additional changes to provide block location information and “write affinity”
MapReduce performance

Test bed

iDataPlex: 42 nodes
8 cores, 8GB RAM
4+1 disks per-node

Hadoop: version 0.18.1
GPFS: version pre3.3

16 nodes
160 GB data
(replication factor = 2)
Impact on traditional workloads

- **iDataPlex**: 42 nodes, 8 cores, 8GB RAM, 4+1 disks per-node
- **GPFS**: version pre3.3, Bonnie filesystem benchmark
Things that didn’t work

- Large filesystem block-size
- Turn-off Prefetching
- Create alignment of records to block boundaries

![Graph showing normalized random and sequential read performance](image-url)
Advantages of traditional filesystems

- Traditional filesystems have solved many hard problems like access control, quotas, snapshots ...
- Allow traditional and MapReduce applications to share the same input data.
- Exploit Filesystem tools & scripts based on “regular” filesystems.
- Re-use of Backup/Archive solutions built around particular filesystems.
- Mixed analytics pipelines.

Using a MapReduce-specific filesystem (e.g. HDFS):

- Crawl
- Load
- Analyze
- Output
- Serve

Crawler writes to a traditional filesystem
Into mapreduce filesystem
Back to traditional filesystem

Using a general-purpose filesystem (e.g. GPFS):

- Crawl
- Analyze
- Serve
Conclusion

- MapReduce platforms can use traditional filesystems without loss of performance.
- There are important reasons why traditional filesystems are attractive to users of MapReduce.