Cloud computing: a simple picture

• Datacenter provides SW/HW as a service

Datacenter
: Warehouse-Scale Computer

Client A

Client B

Client C

Client D

Client E

Datacenter + Virtualization + Application → Cloud Computing
VM-level analysis is **TOO LIMITED**

Difficult to analyze the performance of apps on VM

<table>
<thead>
<tr>
<th>samples</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>45249</td>
<td>98.6892</td>
</tr>
<tr>
<td>364</td>
<td>0.7939</td>
</tr>
<tr>
<td>36</td>
<td>0.0785</td>
</tr>
<tr>
<td>22</td>
<td>0.0480</td>
</tr>
</tbody>
</table>

System-level analysis is **TOO DIFFICULT**

[150GB sort using 640-thread CPUs using Hadoop @ Sun Microsystems]

Various performance bottlenecks exist.
Commercial cloud solutions are TOO expensive

• Can’t use immature open-source solutions
  – Lack of key features
    ▪ e.g., monitoring, migration, RAS, backup, ...

• Can’t afford commercial solutions
  – costs up to 1000s of dollars per CPU + licensing fees
    (for advanced management features)

  Price for 1,000~10,000 nodes?
  How to modify commercial engines?

PosCloud: Advanced open-source based cloud/big data system @ POSTECH
PosCloud: open-source implementation

- monitoring
- analysis
- deployment
- virtualization
- live migration
- power saving
...

<table>
<thead>
<tr>
<th>Tools</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack</td>
<td>Infrastructure as a Service (IaaS)</td>
</tr>
<tr>
<td>Condor</td>
<td>Workload scheduling</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Scalable file system</td>
</tr>
<tr>
<td>Xen</td>
<td>Virtualization</td>
</tr>
</tbody>
</table>

PosCloud: dynamic resource management
(submitted to USENIX ATC’13)

Reliability, Availability, Serviceability

Quality of Service

Scalable Node Management
PosCloud: VM-aware monitoring

PosCloud: Real-world workloads

- **CloudSuite**
  - **Data Serving**
    - Serving data queries in a scalable noSQL storage system
  - **MapReduce**
    - Scalable machine learning library on Hadoop
  - **Media Streaming**
    - RTP/RTSP streaming server
  - **Software Testing**
    - Automated real-world software testing
  - **Web Serving/Search**
    - Search-oriented dynamic Web server
PosCloud: Real-world workloads

- **SpecVirt**
  - OS: Centos 5.6
  - Virtualization: KVM-83
  - Webserver: Apache 2 with PHP 5
  - Infraserver: Apache 2 with fast-cgi
  - Appserver: Oracle Glassvish v2
  - Mailserver: Dovecot 1.2.17
  - DBserver: PostgreSQL 8
PosCloud: cost-effectiveness

<table>
<thead>
<tr>
<th>Cloud Computing Management</th>
<th>Typical Open-source</th>
<th>Typical Commercial</th>
<th>PosCloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>IaaS Service</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Performance</td>
<td>-</td>
<td>✓</td>
<td>✓+</td>
</tr>
<tr>
<td>Power</td>
<td>-</td>
<td>✓</td>
<td>✓+</td>
</tr>
<tr>
<td>Recovery</td>
<td>-</td>
<td>✓</td>
<td>✓+</td>
</tr>
<tr>
<td>Availability</td>
<td>-</td>
<td>✓</td>
<td>✓+</td>
</tr>
<tr>
<td>Other Features</td>
<td>-</td>
<td>?</td>
<td>✓</td>
</tr>
<tr>
<td>Open-source Platform</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>S/W costs</td>
<td>~$0</td>
<td>1000s of $ per CPU</td>
<td>~$0</td>
</tr>
</tbody>
</table>

Commercial-level services at near zero prices!

Smart device: a simple picture

- Assemble standard components fast and nicely

Short time to market!
High performance!
Long battery life!
Cloud computing: a simple picture

- **Datacenter provides SW/HW as a service**

Datacenter: Warehouse-Scale Computer

Datacenter + Virtualization + Application → Cloud Computing

Why mobile cloud computing?

- **High performance**
  - Let’s borrow the server-scale power
    - E.g., 3D HD game using GPU @ datacenter
  - Let’s take advantage of cloud-scale information
    - E.g., Amazon Silk “cloud” browser’s predictive web caching

- ** Longer battery life**
  - Let’s offloading mobile work to datacenter
    - E.g., Intel CloneCloud, Microsoft MAUI

- **Early time to market**
  - Let the cloud handle development-tricky things
Mobile cloud: task offloading

**What** to offload?
- Functions
- Threads
- processes
- Tasks

**When** to offload?
- High performance
- Long battery life

These are ongoing research issues..
BTW, is the cloud free?

---

The importance of scalable storage

- **Representative enterprise cloud storages**
  - Google File System
  - Amazon S3 (Dynamo)
  - Facebook Haystack
  - Yahoo Walnut / HDFS
  - And many more...

@ 2013 Jangwoo Kim
Evaluating cloud storage system

- **Do existing solutions work well?**
  - Is it really fast?
  - Is it really scalable?
  - Is it really reliable?

- **What we are focusing on**
  - Identifying the performance bottleneck of a system
  - Using architectural support to improve existing storage system

Many research challenges for system architects

OpenStack: cloud software framework
Our storage system testbed

• **OpenStack Swift**
  - Popular open-source object storage system for cloud environments
  - Similar to enterprise storage system’s architecture (Amazon’s Dynamo)
  - Scalable and Reliable

• **State-of-the-art cluster**
  - Intel™ SandyBridge Xeon processors
  - 500TB storage capacity (SSD/HDD)

CPU Performance Analysis

• **Where did CPU cycles go?**
  - Cache misses, branch mispredictions, data dependencies...

  - **Programmer’s** aspects
    • Is my code optimal?
    • Why does this function take too long?

  - **Architect’s** aspects
    • Which parts should be improved?
    • Do we need larger caches?
    • Can we reduce the issue width?
What people have been doing?

- **Performance Counter**
  - Counts various architectural events
  - Very fast, but **inaccurate**

- **Architecture model analysis**
  - Find critical instructions based on arch. knowledge
  - Quite accurate, but **difficult to apply**

- **Simulator**
  - Simulates every events of CPU and memory in cycles
  - Very accurate, but **extremely slow**

---

E.g., critical path theory

- **Typical example of arch. mode analysis**

Performance impact of modifying a critical path?
But, too many cases for too many design choices
GPU Research

- Programming/optimizing GPUs is hard!
  - SIMD programming model
  - Hardware-dependent optimization techniques
  - Programmer-dependent performance variation
  - Program-dependent optimal performance
  - ...

Large-Scale GPU Programming

- Difficult code modification

<table>
<thead>
<tr>
<th>Small-scale</th>
<th>Large-scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>// Perform C = A * B on GPU</td>
<td></td>
</tr>
<tr>
<td>MemcopyCPUtoGPU(matA);</td>
<td></td>
</tr>
<tr>
<td>MemcopyCPUtoGPU(matB);</td>
<td></td>
</tr>
<tr>
<td>CalcMatMultAll(matA, matB, matC);</td>
<td></td>
</tr>
<tr>
<td>MemcopyGPUtoCPU(matC);</td>
<td>// Perform C = A * B on GPU</td>
</tr>
<tr>
<td>stream_t strm[rows * cols];</td>
<td></td>
</tr>
<tr>
<td>for (i = 0; i &lt; rows; i++) {</td>
<td></td>
</tr>
<tr>
<td>for (j = 0; j &lt; cols; j++) {</td>
<td></td>
</tr>
<tr>
<td>int idx = i * cols + j;</td>
<td></td>
</tr>
<tr>
<td>CreateStream(&amp;streams[idx]);</td>
<td></td>
</tr>
<tr>
<td>MemcopyGPUtoGPU(matA[i][...]. strm[idx]);</td>
<td></td>
</tr>
<tr>
<td>MemcopyGPUtoGPU(matB[...][j]. strm[idx]);</td>
<td></td>
</tr>
<tr>
<td>CalcMatMultOne(matA, matB, matC, i, j, strm[idx]);</td>
<td></td>
</tr>
<tr>
<td>MemcopyGPUtoGPU(matC[i][j]. strm[idx]);</td>
<td></td>
</tr>
<tr>
<td>PerformStream(strm[idx]);</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>for (i = 0; i &lt; rows; i++)</td>
<td></td>
</tr>
<tr>
<td>for (j = 0; j &lt; cols; j++)</td>
<td></td>
</tr>
<tr>
<td>SynchronizeStream(strm[i * cols + j]);</td>
<td></td>
</tr>
</tbody>
</table>
Large-Scale GPU Programming

- **Huge performance degradation**
  - **Small-scale**
    - h2d exec. d2h
  - **Large-scale**
    - h2d exec. d2h h2d exec. d2h
      - h2d exec. d2h h2d exec. d2h
        - ...

Summary

- **What we do @ POSTECH**
  - CPU performance modeling
  - Future GPU platform
  - PosCloud + PosMCloud
    - Mobile workload offloading
    - Quality-of-Service guarantee
    - Performance monitoring
    - VM, process, function migration
    - Cloud/Big Data workloads
Question?

Thank You!

Jangwoo Kim
e-mail: jangwoo@postech.ac.kr
http://www.postech.ac.kr/~jangwoo