DreamServer: Truly On-Demand Cloud Services

Thomas Knauth, Christof Fetzer
TU Dresden, Germany
The bigger picture

- want to power off physical servers to save energy
- high baseline energy consumption at low utilization yields high savings

The appealing properties of cloud computing

- “infinite” scalability
- pay as you go only for what you are used
- metered resources
- accessed over the network
Inflexible for small customers

- sporadic usage not well supported
- service only a couple of requests per hour
- geographically local user base with diurnal usage pattern
- service idle for most of the time
How to define idleness?

- system idle whenever not processing a request
- build in some slack, i.e., idle if not processed a request for X seconds
How fast can you activate resources?

- physical server: 10 sec
- cloud VM: 10s up to minutes
- boot VM: 10 to 20 sec
- resume VM: seconds up to 10s of seconds

Lazy and hybrid virtual machine resume

• only read checkpoint as needed, instead of eagerly at the beginning

• destroys sequential access pattern

• hybrid: identify “important” parts to read sequentially; read rest on demand
What parts are important?

• working set as defined by P. Denning
• trace page accesses dynamically (Zhang et al.)
• resume set: pages accessed during previous resume
Implementation Details

• QEMU/KVM 1.4

• replaced memory allocation to use mmap() for lazy resume

• combine mmap() with remap_file_pages() to create non-linear mapping to exploit sequential read performance
Lazy and hybrid resume: the gory details

- lazy:
  1. `mmap()` file

- hybrid:
  1. `mmap()`
  2. remap pages
  3. read resume set
Overall system architecture

• must interpose between client and server; many options exist

• we used a modified proxy to track active and idle services

• see our HotSDN poster for SDN-based solution
Experimental setup

- virtual machine host (4-core AMD, 12 GiB memory, one HDD, one SSD)
- remote storage node (6-core AMD, 16 GiB memory, one HDD, one SSD)
- switched gigabit Ethernet
- VMs with seven different HTTP-services
Resume from local HDD

![Graph showing time in seconds for different applications]

<table>
<thead>
<tr>
<th>Application</th>
<th>eager</th>
<th>lazy</th>
<th>hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Django</td>
<td>3.8 s</td>
<td>1.6 s</td>
<td>5.3 s</td>
</tr>
<tr>
<td>Jenkins</td>
<td>6.3 s</td>
<td>3.2 s</td>
<td>6.3 s</td>
</tr>
<tr>
<td>MediaWiki</td>
<td>14.5 s</td>
<td>3.2 s</td>
<td>14.5 s</td>
</tr>
<tr>
<td>OwnCloud</td>
<td>10.4 s</td>
<td>3.8 s</td>
<td>10.4 s</td>
</tr>
<tr>
<td>Rubis</td>
<td>4.3 s</td>
<td>4.3 s</td>
<td>4.3 s</td>
</tr>
<tr>
<td>Trac</td>
<td>7.8 s</td>
<td>4.1 s</td>
<td>4.1 s</td>
</tr>
<tr>
<td>Wordpress</td>
<td>13.5 s</td>
<td>4.2 s</td>
<td>4.2 s</td>
</tr>
</tbody>
</table>

Friday 16 May 14
Resume from local SSD

![Comparison of time taken for different services]

- **Eager**: Faster initial load but may require more resources.
- **Lazy**: Slower initial load but more efficient in terms of resources.
- **Hybrid**: Balances the efficiency of eager and lazy approaches.

Friday 16 May 14
Resume from remote SSD

Friday 16 May 14
Summary

• implemented lazy and hybrid VM resume for QEMU/KVM using little known remap_file_pages() system call

• it’s possible to resume a full OS-level virtual machine in less than one second

• one possible application is for truly on-demand cloud services which are only activated on incoming requests
Open access

• http://bitbucket.org/tknaught/dreamserver
EOP
Backup - non-HTTP services