Server Consolidation in Clouds through Gossiping

Moreno Marzolla       Ozalp Babaoglu       Fabio Panzieri

Università di Bologna, Dip. di Scienze dell'Informazione
{marzolla, babaoglu, panzieri}@cs.unibo.it
http://www.moreno.marzolla.name/
Introduction and Motivations

- The success of the Cloud Computing paradigm has resulted in the creation of large datacenters.
- The electricity bill accounts for a substantial fraction of the total operational costs, and servers are responsible for a substantial fraction of the electrical power consumed.
Electricity use breakdown

Source: Toward energy efficient computing, ACM Queue, http://queue.acm.org/detail.cfm?id=1730791
Virtualization

• Main feature of a Cloud system
  – *Dynamic scalability* (pay-as-you-go economic model)
  – *Virtualization* of resources

• Assumptions
  – Physical resources (servers) are multi-core/multi-processor machines; each server can support up to $C$ VM instances
  – Cloud users request a set of Virtual Machine (VM) instances
  – Users release instances when no longer needed
  – VM instances can be acquired or released at any time
  – The VM monitor supports *live migration* of VMs

• Our goal
  – Minimize energy consumption by *consolidating VMs*
VM Consolidation

(a) Before consolidation

(b) After consolidation of VM₁ and VM₄ to host 2
VM consolidation through gossiping

- Each server hosts the V-MAN daemon
- Daemons maintain an overlay network such that each daemon is connected to at most $K$ other nodes
- Daemons exchange messages only with neighbors
- The overlay is maintained with the Newscast algorithm
Idea

- Node $i$ interacts with each neighbor $j$
- If $i$ has fewer VMs than $j$
  - VMs are migrated from $i$ to $j$
- If $i$ has more VMs than $j$
  - VMs are migrated from $j$ to $i$
Example (1)

- Capacity = 4

![Diagram showing two hosts with VMs](image-url)
Example (1)

- Capacity = 4

I have two VMs
Example (1)

- Capacity = 4
Example (2)

- Capacity = 4
Example (2)

- Capacity = 4

I have two VMs
Example (2)

- Capacity = 4

```
Host 1
VM VM VM VM
Host 2
VM
```

Send me one
1: $i \leftarrow \text{GetProcID}()$

2: \textbf{procedure} ActiveThread
3: \hspace{1em} \textbf{loop}
4: \hspace{2em} Wait $\Delta$
5: \hspace{2em} \textbf{for all} $j \in \text{GetNeighbors}(i)$ \textbf{do}
6: \hspace{3em} Send $\langle H_i \rangle$ to $j$
7: \hspace{3em} Receive $\langle H'_i \rangle$ from $j$
8: \hspace{3em} $H_i \leftarrow H'_i$
9: \hspace{2em} \textbf{end for}
10: \textbf{end loop}
11: \textbf{end procedure}

12: \textbf{procedure} PassiveThread
13: \hspace{1em} \textbf{loop}
14: \hspace{2em} Wait for message $\langle H_j \rangle$ from $j$
15: \hspace{2em} \textbf{if} ($H_i > H_j$) \textbf{then} \textit{\textarrow{Pull from node} $j$}
16: \hspace{3em} $D \leftarrow \min(H_j, C - H_i)$
17: \hspace{3em} Send $\langle H_j - D \rangle$ to $j$
18: \hspace{3em} $H_i \leftarrow H_i + D$
19: \hspace{2em} \textbf{else} \textit{\textarrow{Push to node} $j$}
20: \hspace{3em} $D \leftarrow \min(H_i, C - H_j)$
21: \hspace{3em} Send $\langle H_j + D \rangle$ to $j$
22: \hspace{3em} $H_i \leftarrow H_i - D$
23: \hspace{2em} \textbf{end if}
24: \textbf{end loop}
25: \textbf{end procedure}
Performance assessment

- We implemented V-MAN using the cycle-driven simulator engine provided by PeerSim (peersim.sf.net)

- Parameters:
  - $K=20$ (each node maintains a list of 20 neighbors)
  - $C=8$ (maximum capacity of each host is 8 VMs)
  - Topology is managed using Newscast
  - Length of each simulation run is 20 steps
  - Results are averages of 10 independent simulation runs

- Results:
  - $F_0$ Fraction of empty hosts
  - $F_{0,\text{opt}}$ Optimal fraction of empty hosts
Experiment #1
Static System

![Graph showing static system over time]
Animation

Frame-to-frame animation
Experiment #2
Impact of View Size $K$
Experiment #3 / 1
Impact of Churn

Impact of Churn

\[ \Delta = [-500, 500] \]

\[ \Delta = [-200, 200] \]

\[ \Delta_t = 0 \]

\[ \Delta_t \text{ in } [-200, 200] \]

\[ \Delta_t \text{ in } [-500, 500] \]

No churn

\[ F_{0, \text{opt}} - F_0 \]

Time Step

\[ 2 \]

\[ 4 \]

\[ 6 \]

\[ 8 \]

\[ 10 \]

\[ 12 \]

\[ 14 \]

\[ 16 \]

\[ 18 \]

\[ 20 \]
Experiment #3 / 2
Stopping V-MAN
Experiment #4
Fully Dynamic System
Ongoing work

- Minimize the **size of transferred VM images** (instead of the number of VM instances)
  - Example: what would you migrate?

![Diagram showing Minimize the size of transferred VM images](image-url)
Preliminary results

![Graph showing lot transfer size over time steps]

- Random
- Minimum

Time Step

0 100 200 300 400 500

Lot transfer size
Conclusions

- We propose V-MAN, a totally distributed, gossip-based algorithm for VM consolidation
  - Very fast convergence speed (<5 iterations suffices)
  - Handles churn
  - Resilient to failures

- Ongoing work
  - Minimize size of migrated VM instances (*bin packing* problem?)
  - Penalize migration of long-running VMs
  - Implement V-MAN (in C)

- Source code of V-MAN simulator: [http://www.moreno.marzolla.name/publications/](http://www.moreno.marzolla.name/publications/)
Thank you for your attention!

Questions?
Backup slides
Animation
Animation

Step 001
Animation

Step 002

SUSTAINET 2011, June 20, Lucca, Italy
Animation
Animation
Animation

Step 005
Animation
Animation

Step 007
Animation

Step 008
Animation

Step 009

SUSTAINET 2011, june 20, Lucca, Italy
Animation

Step 010

Back to presentation