Outline

- Introduction
- Experimental setup
- Profiling
  - Component benchmarks
  - Overall benchmarks
- Power models
  - for a single work node
  - for a cloud system
- Energy-aware scheduling algorithms
  - Placement
  - Re-placement
  - Migrating
- Conclusions
Motivation

- From environmental perspective, it’s *green*:
- From economic perspective, it can minimize operational cost of data centers.
Motivation

- From environmental perspective, it’s green:
- From economic perspective, it can minimize operational cost of data centers.

Interesting facts

- 3-years of operational cost > purchasing cost.
- Annual energy cost of data centers in US:
  - in 2006: 61 billion kWh (1.5% of total electricity consumption).
  - by 2011: 100 billion kWh
  - annual savings in energy consumption by 2011: ~10%
  - techniques: state-of-the-art hardware and virtualization
- Carbon Disclosure Project predicts that annual savings with cloud computing by 2020:
  - $12.3bn in energy cost
  - 85.7 million metric tons of CO2 emissions
How can cloud computing help?

A simple example:
How can cloud computing help?

A simple example:

Traditional use
How can cloud computing help?

A simple example:
How can cloud computing help?

A simple example:

Cloud use: more users
Cloud layers

- **Software as a Service (SaaS)**
- **Platform as a Service (PaaS)**
- **Infrastructure as a Service (IaaS)**
- **Virtualization (Hypervisor)**
- **Server (including software & hardware)**

**Service layers**

- **SaaS**: Provides *cloud applications*, e.g. Gmail, Google Docs, etc.
- **PaaS**: Provides *runtime environment*, e.g. Google App Engine, Windows Azure, etc.
- **IaaS**: Provides *virtual machines*, e.g. Amazon EC2, OpenNebula, etc.
Cloud layers

- **Software as a Service (SaaS)**: Provides cloud applications, e.g., Gmail, Google Docs, etc.
- **Platform as a Service (PaaS)**: Provides runtime environment, e.g., Google App Engine, Windows Azure, etc.
- **Infrastructure as a Service (IaaS)**: Provides virtual machines, e.g., Amazon EC2, OpenNebula, etc.
A typical procedure of VM provision in IaaS cloud
A typical procedure of VM provision in IaaS cloud

1) Cloud client requests a VM from Head Node
2) Head Node prepare and push the request to one of its work node
A typical procedure of VM provision in IaaS cloud

3) Work node creates and starts VM
A typical procedure of VM provision in IaaS cloud

4) Head node assign network address to the VM
A typical procedure of VM provision in IaaS cloud

5) Cloud client accesses the VM
A typical procedure of VM provision in IaaS cloud
Research Questions

Goals

- Keep VMs running on energy-efficient hardware
- Minimize the number of up-running work nodes
Research Questions

Goals
- Keep VMs running on energy-efficient hardware
- Minimize the number of up-running work nodes

Opportunities
- Place VMs on energy-efficient work nodes
- Migrate VMs to 'greener' work nodes if possible
- Explore over-committing on lightly loaded work nodes
Research Questions

Goals

- Keep VMs running on energy-efficient hardware
- Minimize the number of up-running work nodes

Opportunities

- Place VMs on energy-efficient work nodes
- Migrate VMs to 'greener' work nodes if possible
- Explore over-committing on lightly loaded work nodes

Challenges

- What are the power/performance behaviors of VMs and work nodes?
- Explore diversity of application patterns
- How to incorporate them into system-level optimization?
Research Method

Profiling
To characterize the performance & power behavior of a single work node

Power models
- To establish a mathematic description of energy efficiency
- To identify the green parameters

Scheduling algorithms
Sample algorithms to handle:
- placement of new VMs
- overloaded work nodes
- global optimization
Research Method

Profiling
To characterize the performance & power behavior of a single work node

Power models
- To establish a mathematic description of energy efficiency
- To identify the green parameters

Scheduling algorithms
Sample algorithms to handle:
- placement of new VMs
- overloaded work nodes
- global optimization
Research Method

Profiling
To characterize the performance & power behavior of a single work node

Power models
- To establish a mathematic description of energy efficiency
- To identify the green parameters

Scheduling algorithms
Sample algorithms to handle:
- placement of new VMs
- overloaded work nodes
- global optimization
Experimental setup

System Under Test
- Hardware is provided by DAS-4 cluster (UvA)
- Dual-quad-core (8-core) machine

Data logger
- Run on the front-end
- Ganglia

Power meter
Precision: 0.1V and 0.01A
Performance & power profiling

Profiling
- Component test
- Overall test

Power behavior

Power models
- Single work node
- Cloud

Power behavior

Scheduling algorithms
- Placement
- Re-placement
- Migrating
## Summary of Benchmarks

<table>
<thead>
<tr>
<th>Test</th>
<th>Sub-test</th>
<th>Workload generators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component test</td>
<td>CPU</td>
<td>Linpack, Stress, scripts</td>
</tr>
<tr>
<td></td>
<td>Memory</td>
<td>Stress</td>
</tr>
<tr>
<td></td>
<td>HDD</td>
<td>Stress</td>
</tr>
<tr>
<td>Overall test</td>
<td>Floating-point operation</td>
<td>Linpack</td>
</tr>
<tr>
<td></td>
<td>Integer operation</td>
<td>Dhrystone, Fhourstones</td>
</tr>
<tr>
<td></td>
<td>Impact of HT</td>
<td>Linpack</td>
</tr>
</tbody>
</table>
# Summary of Benchmarks

<table>
<thead>
<tr>
<th>Test</th>
<th>Sub-test</th>
<th>Workload generators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component test</td>
<td>CPU</td>
<td>Linpack, Stress, scripts</td>
</tr>
<tr>
<td></td>
<td>Memory</td>
<td>Stress</td>
</tr>
<tr>
<td></td>
<td>HDD</td>
<td>Stress</td>
</tr>
<tr>
<td>Overall test</td>
<td>Floating-point operation</td>
<td>Linpack</td>
</tr>
<tr>
<td></td>
<td>Integer operation</td>
<td>Dhrystone, Fhourstones</td>
</tr>
<tr>
<td></td>
<td>Impact of HT</td>
<td>Linpack</td>
</tr>
</tbody>
</table>
Profiling Component benchmarks

CPU

Gradual increase of CPU load on all available cores

Gradual increase of number of cores, where each core is at its maximum usage

Observations

- Power usage is linear to the CPU load.
- No significant differences in power usage of a VM and its host.
Memory

Varying memory usage

Memory and CPU stress tests

Observations

- Nearly constant power usage of memory
- Variation is less than 10% of total power usage
Overall benchmarks

Floating-point operation (Linpack) test

Observations

- Performance \( \propto \) CPU load (\# of threads).
- Power usage is nearly linear to CPU load.
- Abnormal result for over-committed VM (i.e. with 16 vCPUs).
Impact of Hyper-Threading (HT) technology

Linpack test on an over-committed VM (16 vCPUs)

Observations

- With HT enabled, performance and power usage back to normal.
- HT can handle over-committing on a (overloaded) work node:
  - Over-commit: # of vCPUs > # of physical cores
  - Overload: physical CPU usage approaches its limit
Summary

For CPU- and/or memory-only applications:

Power behavior

\[ P = P_{idle} + c_e U_{cpu} \]

Performance behavior

\[ P = c_p U_{cpu} \]

- \( P_{idle} \) - idle power consumption
- \( U_{cpu} \) - instant CPU load, \( 0 < U_{cpu} \leq \) number of physical cores
- \( c_e, c_p \) - power and performance parameters respectively

Sample values in our case

<table>
<thead>
<tr>
<th></th>
<th>( P_{idle} )</th>
<th>( c_e )</th>
<th>( c_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>90W</td>
<td>7.5W</td>
<td>9.4GFlops</td>
</tr>
</tbody>
</table>

Q. Chen (UvA)  
Towards GreenClouds  
August 2011
Power models

- Profiling
  - Component test
  - Overall test

- Power models
  - Single work node
  - Cloud

- Scheduling algorithms
  - Placement
  - Re-placement
  - Migrating
Assumptions

1. An application has fixed amount of computations for the same input parameters
2. VM lives for applications running on it
   - VM is shut down immediately when idle
   - In this sense, VM has computations as a normal application does.
3. Work node lives for VMs
   - A work node is powered off immediately when idle
4. $c_e$ and $c_p$ are hardware-dependent parameters
   - They are constants for the same work node
Power models for a single work node

Energy usage for a single VM running on a single work node

For an application with:
- fixed amount of computations $G$

running on an work node with:
- $P_{idle}$ - idle power consumption
- $c_e$, $c_p$ - power and performance parameters respectively

the total energy consumption during the application’s runtime $T$ is

$$E_{node} = P_{idle} T + \frac{c_e}{c_p} G$$

regardless of the dynamic CPU usage during the runtime.
Power models for a single work node

Energy usage for a single VM running on a single work node

For an application with:
- fixed amount of computations $G$

running on a work node with:
- $P_{idle}$ - idle power consumption
- $c_e, c_p$ - power and performance parameters respectively

the total energy consumption during the application’s runtime $T$ is

$$E_{node} = P_{idle} T + \frac{c_e}{c_p} G$$

regardless of the dynamic CPU usage during the runtime.
Energy usage for a single VM on a single work node

**Proof**

Suppose the CPU usage at time $t$ is $U_{cpu}(t)$, then

$$G = \int_0^T P(t) dt = \int_0^T c_p U_{cpu}(t) dt \Rightarrow \int_0^T U_{cpu}(t) dt = \frac{G}{c_p}$$
Energy usage for a single VM on a single work node

Proof

Suppose the CPU usage at time $t$ is $U_{cpu}(t)$, then

$$G = \int_0^T \mathcal{P}(t) dt = \int_0^T c_p U_{cpu}(t) dt \Rightarrow \int_0^T U_{cpu}(t) dt = \frac{G}{c_p}$$

Total power consumption during the application’s life time:

$$E_{node} = \int_0^T P(t) dt = \int_0^T (P_{idle} + c_e U_{cpu}(t)) dt$$

$$= P_{idle} T + c_e \int_0^T U_{cpu}(t) dt$$
Energy usage for a single VM on a single work node

**Proof**

Suppose the CPU usage at time $t$ is $U_{cpu}(t)$, then

$$G = \int_0^T P(t)dt = \int_0^T c_p U_{cpu}(t)dt \Rightarrow \int_0^T U_{cpu}(t)dt = \frac{G}{c_p}$$

Total power consumption during the application’s life time:

$$E_{node} = \int_0^T P(t)dt = \int_0^T (P_{idle} + c_e U_{cpu}(t))dt$$

$$= P_{idle} T + c_e \int_0^T U_{cpu}(t)dt$$

By substituting, we have
Energy usage for a single VM on a single work node

**Proof**

Suppose the CPU usage at time \( t \) is \( U_{cpu}(t) \), then

\[
G = \int_0^T P(t)dt = \int_0^T c_p U_{cpu}(t)dt \Rightarrow \int_0^T U_{cpu}(t)dt = \frac{G}{c_p}
\]

Total power consumption during the application’s life time:

\[
E_{node} = \int_0^T P(t)dt = \int_0^T (P_{idle} + c_e U_{cpu}(t))dt
\]

\[
= P_{idle} T + c_e \int_0^T U_{cpu}(t)dt
\]

By substituting, we have

\[
E_{node} = P_{idle} T + \frac{c_e}{c_p} G
\]
Energy usage of multiple VMs on a single work node

When $N$ applications (e.g. $N$ VMs) run on a single work node where application $i$ has computations of $G_i$ during its runtime $(t_i^0, t_i^1]$ ($0 < i \leq N$), the total energy consumption of the work node is

$$E_{node} = P_{idle} T + \frac{C_e}{C_p} \sum_{i=1}^{N} G_i$$

where $T = \big| \bigcup_{0<i\leq N} (t_i^0, t_i^1) \big|$ is the joint life time of all applications running within this work node.

Proof

Similar proof. Please refer pp43-44 of my thesis for details.
Energy usage of multiple VMs on a single work node

When $N$ applications (e.g. $N$ VMs) run on a single work node where application $i$ has computations of $G_i$ during its runtime $[t_0^i, t_1^i]$ ($0 < i \leq N$), the total energy consumption of the work node is

$$E_{node} = P_{idle} T + \frac{c_e}{c_p} \sum_{i=1}^{N} G_i$$

where $T = |\bigcup_{0<i\leq N}(t_0^i, t_1^i)|$ is the joint life time of all applications running within this work node.

Observations

- $T$ depends on $c_p$
- $E_{node}$ is independent to the dynamic CPU usage.
Power model for a cloud

Description

- A work node: $wn_i = (P_{idle}^i, c_e^i, c_p^i)$
- A cloud with $M$ work nodes: $C = \{wn_i|0 < i \leq M\}$
- A VM: $vm_i = (G_i, T_i)$, where $T_i = (t_0^i, t_1^i]$  
- Collection of all VMs: $VM = \{vm_i|0 < i \leq N\}$
- Placement of a VM (many-to-one mapping): $f : VM \rightarrow C$
- All VMs running on $wn_i$: $VM_i = \{vm|f(vm) = wn_i, \forall vm \in VM\}$
- Life of $wn_i$: $T_i = \left| \bigcup_{vm_k \in VM_i} T_k \right|$
Power model for a cloud

Description

- A work node: \( wn_i = (P_{idle}^i, c_e^i, c_p^i) \)
- A cloud with \( M \) work nodes: \( C = \{ wn_i | 0 < i \leq M \} \)
- A VM: \( vm_i = (G_i, T_i) \), where \( T_i = (t_0^i, t_1^i) \)
- Collection of all VMs: \( VM = \{ vm_i | 0 < i \leq N \} \)
- Placement of a VM (many-to-one mapping): \( f : VM \rightarrow C \)
- All VMs running on \( wn_i \): \( VM_i = \{ vm | f(vm) = wn_i, vm \in VM \} \)
- Life of \( wn_i \): \( T_i = \sum_{vm_k \in VM_i} |T_k| \)
Power model for a cloud

Description

- A work node: $wn_i = (P_{idle}^i, c_e^i, c_p^i)$
- A cloud with $M$ work nodes: $C = \{wn_i|0 < i \leq M\}$
- A VM: $vm_i = (G_i, T_i)$, where $T_i = (t_0^i, t_1^i)$
- Collection of all VMs: $VM = \{vm_i|0 < i \leq N\}$
- Placement of a VM (many-to-one mapping): $f : VM \rightarrow C$
- All VMs running on $wn_i$: $VM_i = \{vm|f(vm) = wn_i, vm \in VM\}$
- Life of $wn_i$: $T_i = \big| \bigcup_{vm_k \in VM_i} T_k \big|$
Power model for a cloud

Description

- A work node: $wn_i = (P_{idle}^i, c_e^i, c_p^i)$
- A cloud with $M$ work nodes: $C = \{wn_i|0 < i \leq M\}$
- A VM: $vm_i = (G_i, T_i)$, where $T_i = (t_0^i, t_1^i]$
- Collection of all VMs: $VM = \{vm_i|0 < i \leq N\}$
- Placement of a VM (many-to-one mapping): $f: VM \rightarrow C$
- All VMs running on $wn_i$: $VM_i = \{vm|f(vm) = wn_i, vm \in VM\}$
- Life of $wn_i$: $T_i = |\bigcup_{vm_k \in VM_i} T_k|$
Power model for a cloud

**Description**

- A work node: \( wn_i = (P_{idle}^i, c_e^i, c_p^i) \)
- A cloud with \( M \) work nodes: \( C = \{ wn_i | 0 < i \leq M \} \)
- A VM: \( vm_i = (G_i, T_i) \), where \( T_i = (t_0^i, t_1^i] \)
- Collection of all VMs: \( VM = \{ vm_i | 0 < i \leq N \} \)
- Placement of a VM (many-to-one mapping): \( f : VM \rightarrow C \)
- All VMs running on \( wn_i \): \( VM_i = \{ vm | f(vm) = wn_i, vm \in VM \} \)
- Life of \( wn_i \): \( T_i = | \bigcup_{vm_k \in VM_i} T_k | \)
Power model for a cloud

Description

- A work node: $wn_i = (P_{idle}^i, c_e^i, c_p^i)$
- A cloud with $M$ work nodes: $C = \{ wn_i | 0 < i \leq M \}$
- A VM: $vm_i = (G_i, T_i)$, where $T_i = (t_0^i, t_1^i]$
- Collection of all VMs: $VM = \{ vm_i | 0 < i \leq N \}$
- Placement of a VM (many-to-one mapping): $f : VM \rightarrow C$
- All VMs running on $wn_i$: $VM_i = \{ vm | f(vm) = wn_i, vm \in VM \}$
- Life of $wn_i$: $T_i = | \bigcup_{vm_k \in VM_i} T_k |$
Power model for a cloud

Energy usage of a cloud

\[ E_{\text{cloud}} = \sum_{i=1}^{M} E_{\text{node}}^i = \sum_{i=1}^{M} (P_{\text{idle}}^i T_i + \frac{C_e^i}{C_p^i} \cdot \sum_{v_{m_j} \in VM_i} G_j) \]
Power model for a cloud

Energy usage of a cloud

\[ E_{cloud} = \sum_{i=1}^{M} E^i_{node} = \sum_{i=1}^{M} \left( P^i_{idle} T^i + \frac{c_{e}^i}{c_{p}^i} \cdot \sum_{vm_j \in VM_i} G_j \right) \]

Energy-aware optimization

\[ \min(E_{cloud}) = \min_f \left( \sum_{i=1}^{M} \left( P^i_{idle} T^i + \frac{c_{e}^i}{c_{p}^i} \cdot \sum_{vm_j \in VM_i} G_j \right) \right) \]
Power model for a cloud

Energy usage of a cloud

\[ E_{\text{cloud}} = \sum_{i=1}^{M} E_{\text{node}}^i = \sum_{i=1}^{M} (P_{\text{idle}}^i T_i + \frac{c_e^i}{c_p^i} \cdot \sum_{v_{m_j} \in \text{VM}_i} G_j) \]

Energy-aware optimization

\[ \min(E_{\text{cloud}}) = \min_f \left( \sum_{i=1}^{M} (P_{\text{idle}}^i T_i + \frac{c_e^i}{c_p^i} \cdot \sum_{v_{m_j} \in \text{VM}_i} G_j) \right) \]

Energy efficiency of a work node is jointly defined by \((P_{\text{idle}}, c_e, c_p)\).
Energy-aware scheduling algorithms

Profiling

- Component test
- Overall test

Power models

- Single work node
- Cloud

Scheduling algorithms

- Placement
- Re-placement
- Migrating
Working scenario of an energy-aware scheduler
Placement scheduler

- To place VMs on suitable work nodes
- Triggered when a new VM provision request arrives

Energy-aware placement scheduling algorithm

![Diagram of energy-aware placement scheduling algorithm]

- Active set
- Idle set

Energy efficiency

- Work node A
- Work node B
- Work node C

- VM 6
- VM 4
- VM 8

- n vCPUs
Placement scheduler

- To place VMs on suitable work nodes
- Triggered when a new VM provision request arrives

Energy-aware placement scheduling algorithm

![Diagram showing the process of energy-aware scheduling](image_url)
Migrating scheduler

- Global optimization
- Recommended to be triggered when no work node is idle

Energy-aware migrating scheduling algorithm
Migrating scheduler

- Global optimization
- Recommended to be triggered when no work node is idle

Energy-aware migrating scheduling algorithm
Migrating scheduler

- Global optimization
- Recommended to be triggered when no work node is idle

Energy-aware migrating scheduling algorithm
Re-placement scheduler

- To handle single overloaded work node
- Triggered when a work node is overloaded.
- Only make sense if the overloaded work node is over-committed

Re-placement algorithm for overloaded (and over-committed) work nodes
Re-placement scheduler

- To handle single overloaded work node
- Triggered when a work node is overloaded.
- Only make sense if the overloaded work node is over-committed

Re-placement algorithm for overloaded (and over-committed) work nodes
Re-placement scheduler

- To handle single overloaded work node
- Triggered when a work node is overloaded.
- Only make sense if the overloaded work node is over-committed

Re-placement algorithm for overloaded (and over-committed) work nodes
Conclusions

- Profiled power & performance behaviors for energy efficiency
- Developed our power models for a cloud system
  - Energy consumption is regardless of dynamic resource (CPU & memory) usage
- Identified several key parameters for green hardware
  - Energy efficiency is jointly determined by \((P_{idle}, c_p, c_e)\).
- Proposed novel VM scheduling algorithms for energy-aware clouds
Conclusions

Research outputs

- Extended Ganglia monitor system (Show demo)
- Two academic articles (submitted) and a poster (submitted):

  ICPADS 2011

  CGC 2011

  CGC 2011
Thanks
&
Questions