Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing

Anton Beloglazov

Supervisor: Prof. Rajkumar Buyya

The Cloud Computing and Distributed Systems (CLOUDS) Lab
CIS Department, The University of Melbourne

THE UNIVERSITY OF MELBOURNE

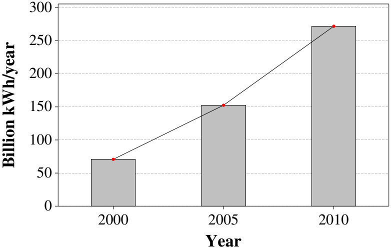
PhD Completion Seminar

MELBOURNE CLOUDS LAB

Cloud Data Centers

- Delivering computing resources on-demand over the Internet
- Hundreds of thousands of servers worldwide
- Amazon EC2 2012
  - 450,000 servers [Liu, 2012]
  - 9 regions
- High energy consumption and CO_2 emissions [Koomey, 2011]
  - 2005-2010: 56% increase in energy consumption
  - 2% of global CO_2 emissions [Gartner, 2007]

Google's data center [Google, 2012]

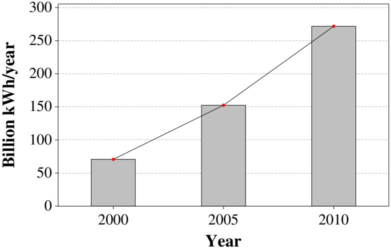


Worldwide data center energy consumption 2000-2010 [Koomey, 2011]

CLOUD DATA CENTERS

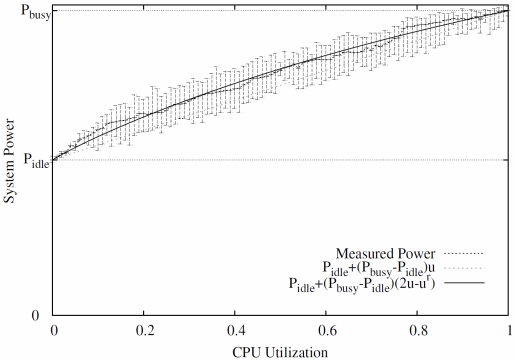
- Delivering computing resources on-demand over the Internet
- Hundreds of thousands of servers worldwide
- Amazon EC2 2012
  - 450,000 servers [Liu, 2012]
  - 9 regions **+ Sydney (2012)!**
- High energy consumption and CO_2 emissions [Koomey, 2011]
  - 2005-2010: 56% increase in energy consumption
  - 2% of global CO_2 emissions [Gartner, 2007]

Google's data center [Google, 2012]



Worldwide data center energy consumption 2000-2010 [Koomey, 2011]

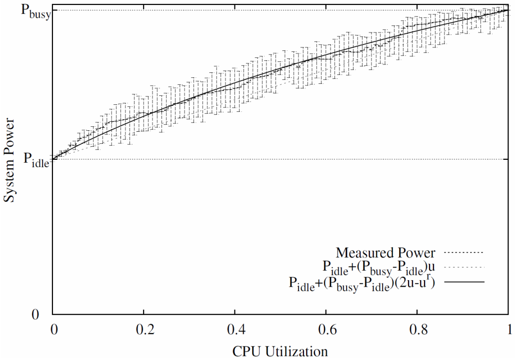
Sources of Energy Waste



Server power consumption depending on the CPU utilization [Fan, 2007]

1. Infrastructure efficiency
   - Facebook's Oregon data center PUE = 1.08 [Open Compute, 2012]
   - 91% of energy is consumed by the computing resources
2. Resource utilization
   - Average CPU utilization: < 50% [Barroso, 2007]
   - Low server dynamic power range: 30% [Fan, 2007]

SOURCES OF ENERGY WASTE

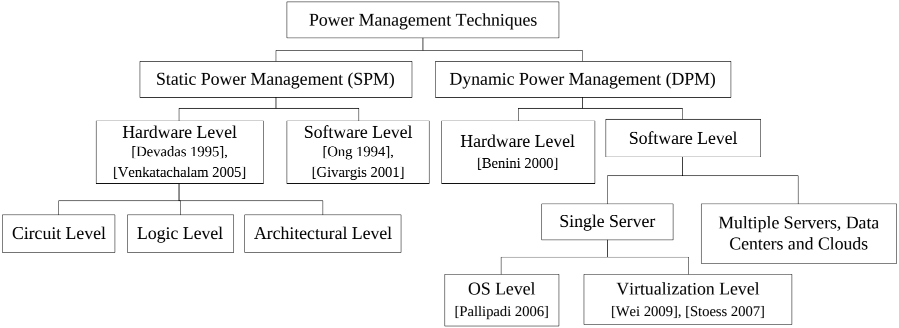


Server power consumption depending on the CPU utilization [Fan, 2007]

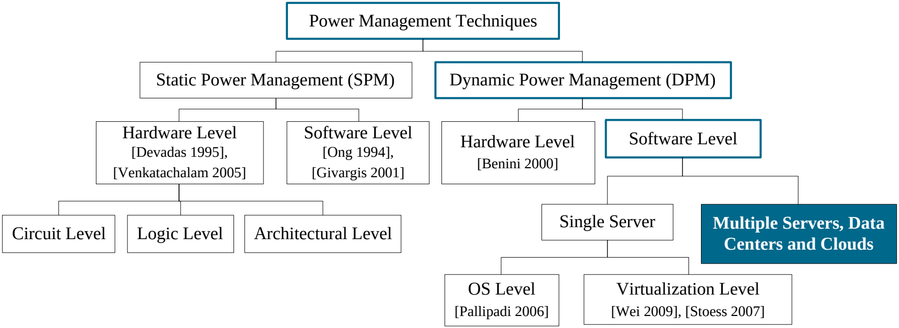
1. Infrastructure efficiency
   - Facebook's Oregon data center PUE = 1.08 [Open Compute, 2012]
   - 91% of energy is consumed by the computing resources
2. Resource utilization
   - Average CPU utilization: < 50% [Barroso, 2007]
   - Low server dynamic power range: 30% [Fan, 2007]

Solution – sleep mode!
450 W $\rightarrow$ 10 W in 300 ms

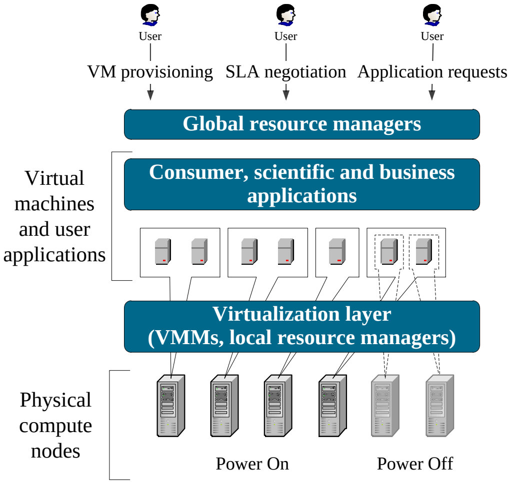
A TAXONOMY OF ENERGY-EFFICIENT COMPUTING



A TAXONOMY OF ENERGY-EFFICIENT COMPUTING



DYNAMIC CONSOLIDATION OF VIRTUAL MACHINES



- Adjusts the number of active hosts according to the resource demand
- Improves the power proportionality
- 2 basic processes
  - VM consolidation
  - VM deconsolidation
- Nathuji 2007
  Raghavendra 2008
  Verma 2008
  Kusic 2009
  Hermenier 2009

Infrastructure as a Service – Properties

1. Large scale
   - Amazon EC2: ≈450,000 servers
   - Rackspace: ≈85,000 servers
   - Scalability and fault-tolerance are required
2. Multiple independent users
   - On-demand VM provisioning
   - Full access and permissions
   - VM provisioning time is unknown
3. Unknown mixed workloads
   - Web, HPC applications
   - The provider is unaware of the application workloads
4. Quality of Service (QoS) guarantees
   - Currently, the performance in IaaS is not guaranteed
   - Existing metrics: availability, response time, deadlines
   - Workload independent QoS are required

RESEARCH QUESTIONS

1. How to define workload-independent QoS requirements?
2. When to migrate VMs?
3. Which VMs to migrate?
4. Where to migrate the VMs selected for migration?
5. When and which physical nodes to switch on/off?
6. How to provide scalability and fault-tolerance?

Thesis Contributions

1. A taxonomy and survey of energy-efficient computing
   - ***Advances in Computers 2011***
2. Competitive analysis of dynamic VM consolidation
   - ***CCPE 2012***
3. Novel heuristics for dynamic VM consolidation
   - ***FGCS 2012, CCPE 2012***
4. The Markov host overload detection algorithm
   - ***TPDS 2013***
5. A software framework for dynamic VM consolidation
   - ***SPE 2013 (in prep.)***

Thesis Contributions

1. A taxonomy and survey of energy-efficient computing
   - *Advances in Computers 2011*
2. Competitive analysis of dynamic VM consolidation
   - *CCPE 2012*
3. Novel heuristics for dynamic VM consolidation
   - ***FGCS 2012, CCPE 2012***
4. The Markov host overload detection algorithm
   - ***TPDS 2013***
5. A software framework for dynamic VM consolidation
   - ***SPE 2013 (in prep.)***

Outline

OUTLINE

DISTRIBUTED APPROACH: 4 SUB-PROBLEMS

1. Host underload detection
2. Host overload detection
3. VM selection
4. VM placement

DISTRIBUTED APPROACH: 4 SUB-PROBLEMS

1. Host underload detection
2. Host overload detection
3. VM selection
4. VM placement

Scalability and fault-tolerance $\rightarrow$ distribution and replication

OUTLINE

Overload Time Fraction (OTF)

$$OTF(u_t) = \frac{t_o(u_t)}{t_a}$$

- u_t – the CPU utilization threshold distinguishing the non-overload and overload states of a host
- t_o – the time, during which the host has been overloaded, which is a function of u_t
- t_a – the total time, during which the host has been active

Aggregate Overload Time Fraction (AOTF)

$$AOTF(u_t) = \sum_{h \in \mathcal{H}} \frac{t_o(h, u_t)}{t_a(h)}$$

- u_t – the CPU utilization threshold distinguishing the non-overload and overload states of a host
- $\mathcal{H}$ – the set of compute hosts
- h – a compute host
- $t_o(h, u_t)$ – the overload time of the host h, which is a function of u_t
- $t_a(h)$ – the total activity time of the host h

OUTLINE

HOST UNDERLOAD DETECTION

A simple algorithm for simulation purposes:

Input: Hosts, VMs
Output: A decision on whether the host is underloaded

```
Place all VMs from the current host on other hosts
if a feasible placement exists then
    return true
return false
```

HOST OVERLOAD DETECTION

- A static CPU utilization threshold (THR)
- Adaptive threshold-based algorithms:
  - Adjust the threshold depending on the strength of deviation of the CPU utilization
  - Median Absolute Deviation (MAD): $u_n > 1 - s \times MAD$
  - Interquartile Range (IQR): $u_n > 1 - s \times IQR$
- Local regression-based algorithms: $s \times \widehat{u}_{n+1} >= 1$
  - Local Regression (LR)
  - Robust Local Regression (LRR)

VM SELECTION

- Minimum Migration Time (MMT)
  - Estimate the VM migration time as RAM/BW
- Random Selection (RS)
  - Randomly select a VM
- Maximum Correlation (MC)
  - Select the VM that maximizes the multiple correlation coefficient

ALGORITHMS: VM PLACEMENT

- A modification of the Best Fit Decreasing (BFD) algorithm, which uses no more than $(11/9 \times OPT + 1)$ bins [Yue, 1991]
- Extensions:
  - A constraint on the amount of RAM required by the VMs
  - An inactive host is only activated when a VM cannot be placed on one of the already active hosts
- The worst-case complexity: $(n + m/2)m$
  - n – the number of physical nodes
  - m – the number of VMs to be placed
  - The worst case occurs when every VM to be placed requires a new inactive host to be activated

Performance Metrics

$$AOTF(u_t) = \sum_{h \in \mathcal{H}} \frac{t_o(h, u_t)}{t_a(h)}$$

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{d_j}}{C_{r_j}}$$

$$SLAV = AOTF \times PDM$$

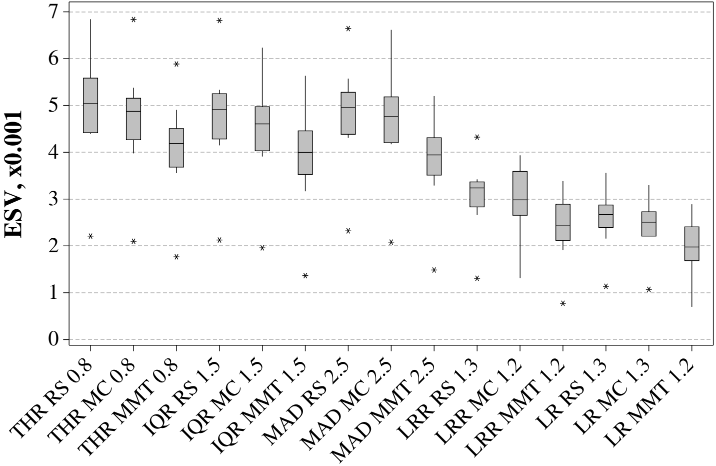
$$ESV = E \times SLAV$$

- PDM – Performance Degradation due to Migrations
- M – the number of VMs
- C_{d_j} – the estimate of the performance degradation of the VM j caused by migrations
- C_{r_j} – the total CPU capacity requested by the VM j
- SLAV – SLA Violation
- ESV – the Energy - SLA Violation combined metric

EXPERIMENTAL SETUP

- Simulation using CloudSim [Calheiros, 2011]
- Power consumption data from SPECpower:
  - 400 × HP ProLiant ML110 G4 (2 cores × 1860 MHz, 4 GB)
  - 400 × HP ProLiant ML110 G5 (2 cores × 2660 MHz, 4 GB)
- VM CPU utilization traces from PlanetLab [Park, 2006]
  - Collected every 5 minutes during 10 days of 03-04/2011
  - 898-1516 VMs per day

SIMULATION RESULTS: ESV



SIMULATION RESULTS: SUMMARY

Simulation results of the best algorithm combinations and benchmark algorithms (median values)

Policy	**ESV**($\times 10^{-3}$)	**Energy** (kWh)	**SLAV**($\times 10^{-5}$)
DVFS	0	613.6	0
THR-MMT-1.0	20.12	75.36	25.78
THR-MMT-0.8	4.19	89.92	4.57
IQR-MMT-1.5	4.00	90.13	4.51
MAD-MMT-2.5	3.94	87.67	4.48
LRR-MMT-1.2	2.43	87.93	2.77
LR-MMT-1.2	1.98	88.17	2.33

CONCLUSIONS

1. Dynamic VM consolidation algorithms significantly outperform static allocation policies, such as DVFS
2. The MMT policy produces better results compared to the MC and RS policies: minimization of the VM migration time is more important than the correlation
3. Host overload detection algorithms based on local regression outperform the threshold based algorithms due to a decreased level of SLA violations and the number of VM migrations

Outline

HOST OVERLOAD DETECTION

- Host overload detection has direct influence on the QoS
  - Since host overloads cause resource shortages and performance degradation of applications
- Current algorithms have no direct control over the QoS
  - Only by tuning the algorithm parameters
- Overload detection is done by each host independently

QUALITY OF DYNAMIC VM CONSOLIDATION

$$H = \frac{1}{n}\sum_{i=1}^{n} a_i \rightarrow min$$

- H – the mean number of active hosts over n time steps
- a_i is the number of active hosts at the time step $i = 1, 2, \ldots, n$
- A lower value of H represents a better quality of VM consolidation

Quality of Dynamic VM Consolidation

$$E[H^*] \propto \frac{np^2}{2E[T]}\left(1 + \frac{n}{E[T]}\right),$$

therefore, $E[T] \rightarrow max$

- $E[H^*]$ – the mean number of active hosts switched on due to VM migrations initiated by the host overload detection algorithm over n time steps
- p – the probability that an extra host has to be activated to migrate a VM from an overloaded host
- $E[T]$ is the expected time between migrations from overloaded hosts

THE HOST OVERLOAD DETECTION PROBLEM

$$t_a(t_m) \to \max$$
$$\frac{t_o(t_m)}{t_a(t_m)} \leq M$$

- The problem is limited to a single VM migration
- $t_a(t_m)$ – the time until migration, which is a function of t_m
- t_m – the VM migration start time
- $t_o(t_m)$ – the time, during which the host has been overloaded, which is a function of t_m and u_t
- M – the limit on the maximum allowed OTF value, which is a QoS goal expressed in terms of OTF

OUTLINE

THE OPTIMAL OFFLINE ALGORITHM

Input: A system state *history*
Input: M, the maximum allowed OTF
Output: A VM migration time

while *history* is not empty **do**
  if OTF of *history* $\leq M$ **then**
    return the time of the last *history* state
  else
    drop the last state from *history*

Outline

THE HOST MODEL

- Consider the time period until the first VM migration
- States are assigned to N CPU utilization intervals
- E.g., a host is overloaded if the CPU utilization $\geq 80\%$
  - The state space $\mathcal{S}$ of the DTMC contains 2 states
  - State 1: [0%, 80%)
  - State 2: [80%, 100%]
- Assuming the workload is known, a matrix of transition probabilities $\mathbf{P}$ can be estimated for $i, j \in \mathcal{S}$:

$$\widehat{p}_{ij} = \frac{c_{ij}}{\sum_{k \in \mathcal{S}} c_{ik}}$$

  - c_{ij} – the number of transitions between states i and j

The Host Model

- To model VM migrations we add an *absorbing state*
- A state k is absorbing if no other state can be reached from it, i.e., $p_{kk} = 1$
- The resulting extended state space is $\mathcal{S}^* = \mathcal{S} \cup \{(N+1)\}$
- Then, the control policy is represented by the transition probabilities to the absorbing state $(N+1)$

The Host Model

- The extended matrix of transition probabilities $\mathbf{P}^*$:

$$\mathbf{P}^* = \begin{pmatrix} p^*_{11} & \cdots & p^*_{1N} & m_1 \\ \vdots & \ddots & \vdots & \vdots \\ p^*_{N1} & \cdots & p^*_{NN} & m_N \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$p^*_{ij} = p_{ij}(1 - m_i), \quad \forall i, j \in \mathcal{S}$$

- *Closed-form* equations for the expected time until absorption spent in each state can be obtained, where the unknowns are the required $m_1, m_2, \ldots, m_N$:

$$L_1(\infty), L_2(\infty), \ldots, L_N(\infty)$$

The Optimization Problem

$$\sum_{i \in \mathcal{S}} L_i(\infty) \to \max$$

$$\frac{T_m + L_N(\infty)}{T_m + \sum_{i \in \mathcal{S}} L_i(\infty)} \leq M$$

- $L_i(\infty)$ – the expected time until absorption spent in the state i
- $L_N(\infty)$ – the expected time until absorption spent in the overload state N
- T_m – the VM migration time
- M – the limit on the maximum allowed OTF value

The Control Policy

- The solution of the optimization problem are the probabilities of transitions to the absorbing state $(N+1)$, $m_1, m_2, \ldots, m_N$
- A VM is migrated with the probability m_i, where $i \in \mathcal{S}$ is the current state
- The control policy is *deterministic* if: $\exists k \in \mathcal{S} : m_k = 1$ and $\forall i \in \mathcal{S}, i \neq k : m_i = 0$
- Otherwise the policy is *randomized*

THE MHOD-OPT ALGORITHM

Input: Transition probabilities
Output: A decision on whether to migrate a VM

```
Build the objective and constraint functions
Invoke the brute-force search to find the m vector
if a feasible solution exists then
    Extract the VM migration probability
    if the probability is < 1 then
        return false
return true
```

The MHOD Algorithm

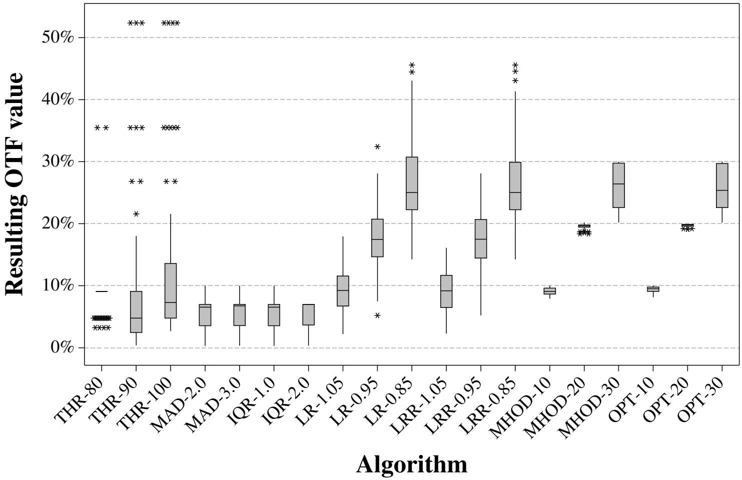
Input: A CPU utilization history
Output: A decision on whether to migrate a VM

1: **if** the CPU utilization history size $> T_l$ **then**
2: Convert the last CPU utilization value to a state
3: Invoke the Multisize Sliding Window estimation [Luiz, 2010] to obtain transition probability estimates
4: Invoke the MHOD-OPT algorithm
5: **return** the decision returned by MHOD-OPT
6: **return false**

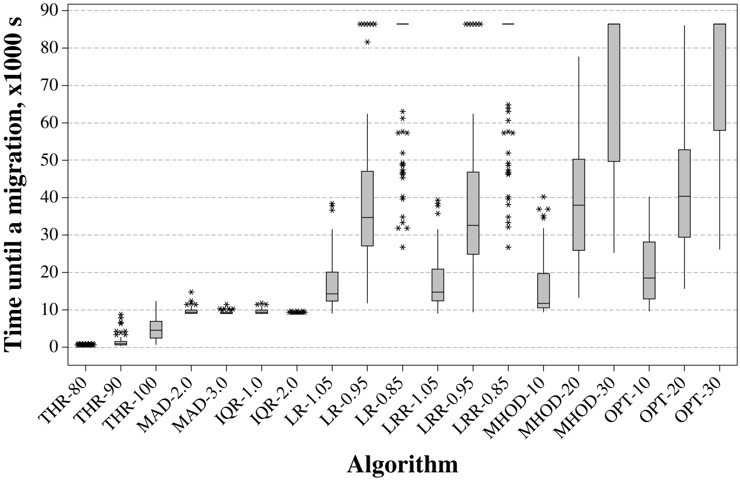
Experimental Setup

- Simulated a single host: 4 cores $\times$ 3 GHz
- 4 VM instance types: 1.7 GHz, 2 GHz, 2.4 GHz, and 3 GHz
- VM CPU utilization traces from PlanetLab [Park, 2006]
  - Collected every 5 minutes during 10 days of 03-04/2011
- 100 different sets of VMs
  - The max OTF after the first 30 time steps is 10%
  - The min overall OTF is 20%
- A simulation is run until the first VM migration

Simulation Results: OTF



Simulation Results: Time Until a Migration



Simulation Results: MHOD vs LRR

Paired T-tests for comparing the time until a migration

Alg. 1 ($\times 10^3$)	**Alg. 2** ($\times 10^3$)	**Diff.** ($\times 10^3$)	***p*-value**
MHOD (39.64)	LR (44.29)	4.65 (2.73, 6.57)	< 0.001
MHOD (39.23)	LRR (44.23)	5.00 (3.09, 6.91)	< 0.001

Simulation Results: MHOD vs OPT

	OPT	MHOD	Difference	*p*-value
OTF	18.31%	18.25%	0.06% (-0.03, 0.15)	$= 0.226$
Time	45,767	41,128	4,639 (3617, 5661)	< 0.001

- Relatively to OPT, the time until a migration produced by the MHOD algorithm converts to 88.02% with 95% CI: (86.07%, 89.97%)

CONCLUSIONS

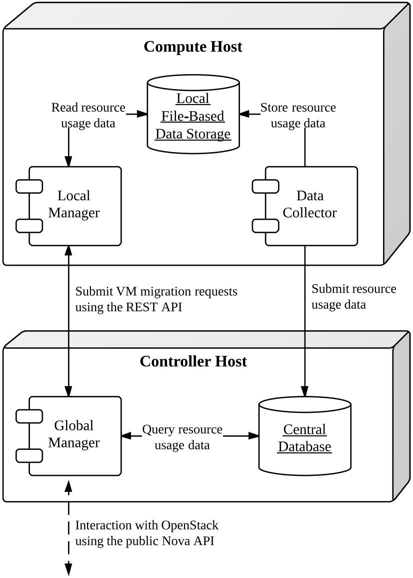
1. MHOD on average provides approximately 88% of the time until a VM migration produced by OPT
2. MHOD leads to approximately 11% shorter time until a migration than the LRR algorithm, while satisfying QoS constraints
3. The MHOD algorithm enables explicit specification of a desired QoS goal to be delivered by the system through the OTF parameter, which is successfully met by the resulting value of the OTF metric

Outline

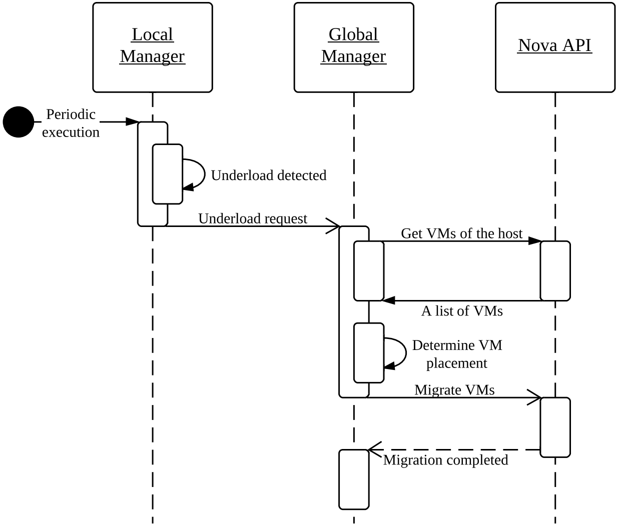
DYNAMIC VM CONSOLIDATION FRAMEWORK

- An extension for the OpenStack Cloud platform
  - Supported by the industry: Rackspace, NASA, IBM, etc.
  - Scalable and fault-tolerant: loose coupling + replication
- The framework transparently attaches to existing OpenStack deployments with no configuration changes
- Interaction with OpenStack through public APIs
- Configuration-based substitution of algorithms
- Open source, released under the Apache 2.0 license: `http://openstack-neat.org/`
  - Neat – (adjective) arranged in an orderly, tidy way

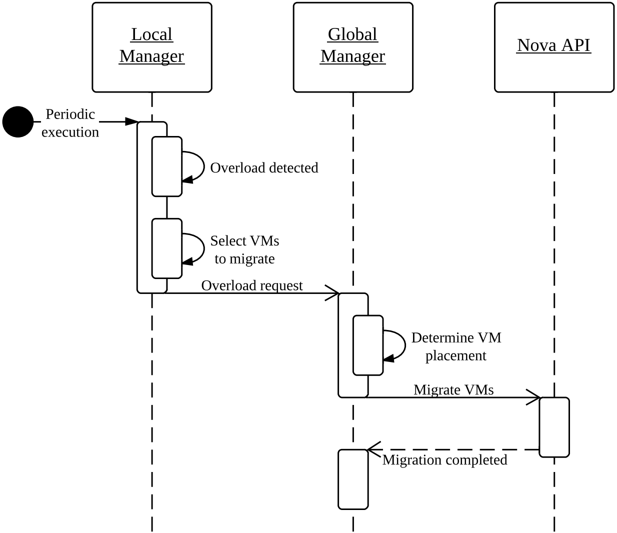
FRAMEWORK COMPONENTS



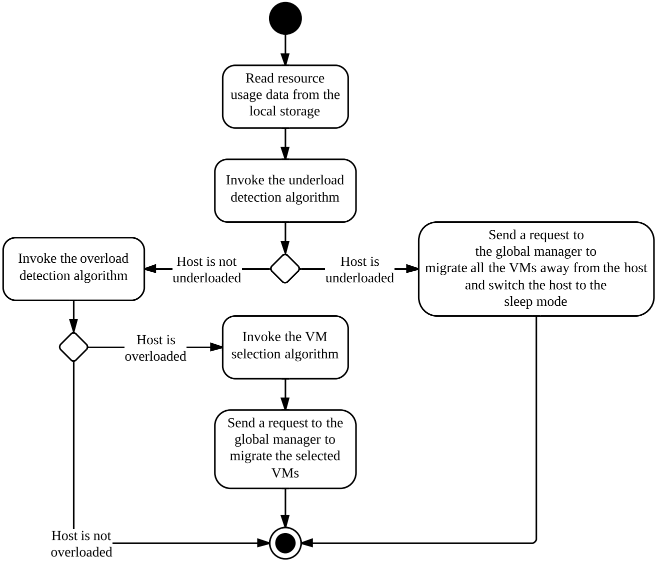
THE GLOBAL MANAGER: UNDERLOAD



The Global Manager: Overload



THE LOCAL MANAGER



IDLE TIME FRACTION (ITF)

$$ITF = \frac{t_i}{t_a}$$

$$AITF = \sum_{h \in \mathcal{H}} \frac{t_i(h)}{t_a(h)}$$

- t_i – the time, during which the host has been idle
- t_a – the total time, during which the host has been active
- $\mathcal{H}$ – the set of compute hosts
- h – a compute host

DYNAMIC VM CONSOLIDATION ALGORITHMS

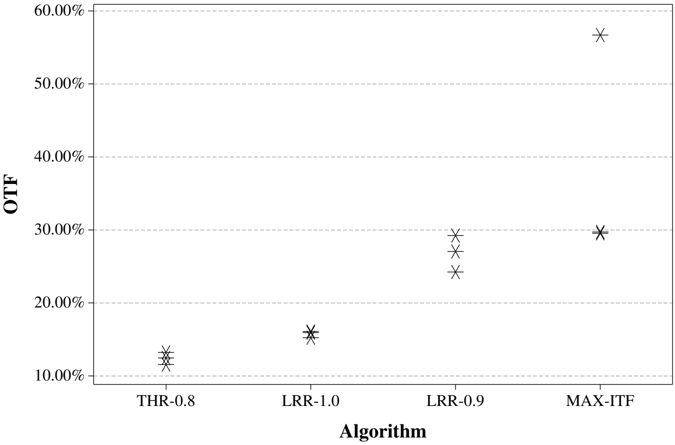
- Host underload detection
  - The averaging threshold-based underload detection algorithm
- Host overload detection
  - MAX-ITF – a base line algorithm, which never detects host overloads leading to the maximum ITF
  - THR – the averaging threshold-based algorithm
  - LRR – the robust local regression algorithm
  - MHOD – pending
- VM selection
  - The min migration time max CPU utilization algorithm
- VM placement
  - The BFD-based algorithm with CPU utilization averaging

OUTLINE

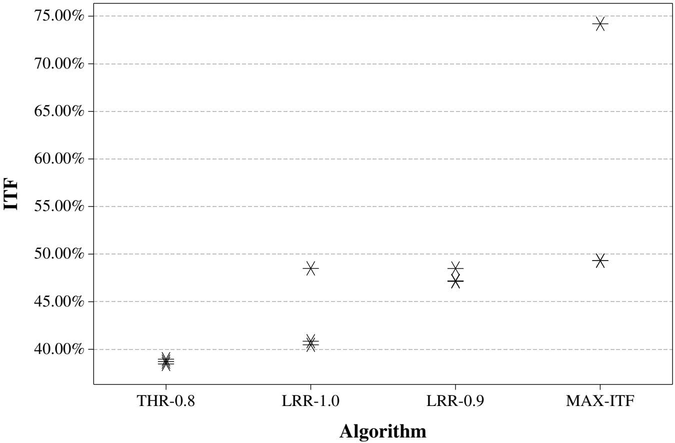
Experimental Setup

- 4 compute nodes, 1 controller node
  - 4 x IBM System x3200 M3 (8 threads × 2800 MHz, 4 GB)
  - 1 x Dell Optiplex 745 (2 threads × 2400 MHz, 2 GB)
- 28 VMs
  - 1 virtual CPU
  - 128 MB RAM
  - Ubuntu 12.04 Cloud Image
- VM CPU utilization traces from PlanetLab [Park, 2006]
  - Collected every 5 minutes during 10 days of 03-04/2011
  - At least 10% of time the CPU utilization is lower than 20%
  - At least 10% of time the CPU utilization is higher than 80%
- Each experiment is 24 hour long × 3
- No sleep mode – AITF

EXPERIMENTAL RESULTS: AOTF



EXPERIMENTAL RESULTS: AITF



EXPERIMENTAL RESULTS: SUMMARY

- Server power consumption [Meisner, 2009]:
  - 450 W – the fully utilized state
  - 270 W – the idle state
  - 10.4 W – the sleep mode

Algorithm	AOTF	AITF	Energy savings
THR-0.8	12.4% (10.3, 14.5)	38.7% (38.1, 39.3)	26.80%
LRR-1.0	15.7% (14.6, 16.9)	43.3% (32.0, 54.5)	30.36%
LRR-0.9	26.8% (20.6, 33.1)	47.6% (45.7, 49.5)	32.98%
MAX-ITF	38.7% (0.0, 77.5)	57.6% (21.9, 93.3)	40.96%

CONCLUSIONS

- OpenStack Neat is transparent to the base OpenStack installation by interacting with it using the public APIs
- The framework can be customized to use various implementations of VM consolidation algorithms
- On a 4-node testbed the energy consumption has been reduced by up to 30% with a limited application performance impact of 15% OTF
- Iterations of the components take a fraction of a second
- The request processing of the global manager takes on average 20 to 40 seconds required for VM migration
- OpenStack Neat is released under the Apache 2.0 license: `http://openstack-neat.org/`

Outline

Conclusions

- Dynamic VM consolidation significantly reduces energy consumption by adjusting the number of active servers
- Scalability and fault-tolerance are crucial in large-scale IaaS
- The proposed approach is distributed, scalable, and efficient in managing the energy-performance trade-off
- The proposed approach allows the system administrator to explicitly specify workload-independent QoS constraints
- On a 4-node testbed, the estimated energy savings are up to 30% with a limited performance impact
- The implemented OpenStack Neat framework is open source: `http://openstack-neat.org/`

Future Directions

- Replicated Global Managers
  - Achieving the complete distribution
- VM Network Topologies
  - Taking into account network communication between VMs
- Thermal-Aware Dynamic VM Consolidation
  - Taking into account the server temperature and cooling
- Dynamic and Heterogeneous SLAs
  - Handling per-user SLAs, which may vary over time
- Power Capping
  - Constraining the overall power consumption by servers

Selected Publications

1. **Anton Beloglazov**, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers, Marvin V. Zelkowitz (editor)*, 82:47-111, 2011
2. **Anton Beloglazov**, Jemal Abawajy, and Rajkumar Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems (FGCS)*, 28(5):755-768, 2012
3. **Anton Beloglazov** and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurrency and Computation: Practice and Experience (CCPE)*, 24(13):1397-1420, 2012
4. **Anton Beloglazov** and Rajkumar Buyya, "Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2013 (in press, accepted on August 2, 2012)
5. **Anton Beloglazov** and Rajkumar Buyya, "OpenStack Neat: A Framework for Dynamic Consolidation of Virtual Machines in OpenStack Clouds," *Software: Practice and Experience (SPE)*, 2013 (in preparation)

ACKNOWLEDGEMENTS

- Supervisor: Prof. Rajkumar Buyya
- PhD committee:
  - Prof. Chris Leckie
  - Dr. Rodrigo Calheiros
  - Dr. Saurabh Garg
- Past and current members of the CLOUDS Laboratory
- Friends and colleagues from the CSSE/CIS department

MORE INFORMATION

- Thesis:
  http://beloglazov.info/thesis.pdf
- Slides:
  http://beloglazov.info/thesis-slides.pdf
- More information and publications:
  http://beloglazov.info

Thank you all for coming! Any questions?

Appendix

References I

- Google
  Google data centers.
  http://www.google.com/datacenters/

- H. Liu
  Amazon data center size.
  http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/

- J. G. Koomey
  Growth in data center electricity use 2005 to 2010.
  Analytics Press, 2011.

References II

- Gartner, Inc.
  Gartner estimates ICT industry accounts for 2 percent of global CO_2 emissions.
  http://www.gartner.com/it/page.jsp?id=503867

- The Open Compute Project
  Energy Efficiency.
  http://opencompute.org/about/energy-efficiency/

- X. Fan, W. D. Weber, and L. A. Barroso
  Power provisioning for a warehouse-sized computer.
  34th Annual International Symposium on Computer Architecture (ISCA), 2007

References III

- L. A. Barroso and U. Holzle
  The case for energy-proportional computing.
  Computer, 40(12):33–37, 2007

- K. S Park and V. S Pai
  CoMon: a mostly-scalable monitoring system for PlanetLab.
  ACM SIGOPS Operating Systems Review, 40(1):65–74, 2006

- R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya
  CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms.
  Software: Practice and Experience, 41(1):23–50, 2011

References IV

D. Meisner, B.T. Gold, and T.F. Wenisch
PowerNap: Eliminating server idle power.
ACM SIGPLAN Notices, 44(3):205–216, 2009

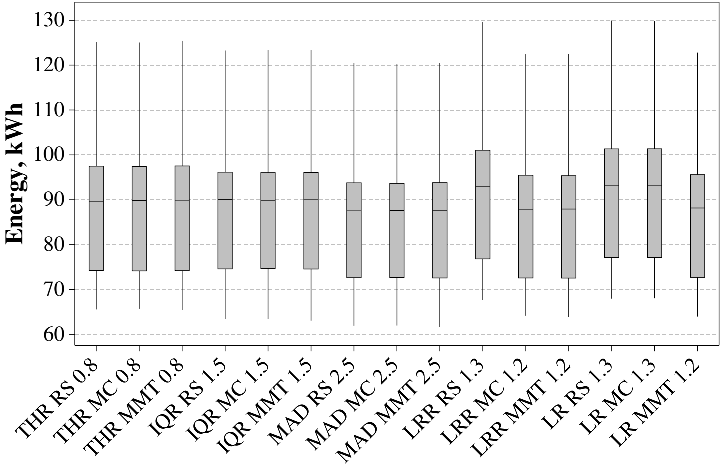
M. Yue
A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1,for all L for the FFD bin-packing algorithm.
Acta Mathematicae Applicatae Sinica, 7(4):321–331, 1991

S. O. D. Luiz, A. Perkusich, and A. M. N. Lima
Multisize Sliding Window in Workload Estimation for Dynamic Power Management.
IEEE Transactions on Computers, 59(12):1625–1639, 2010

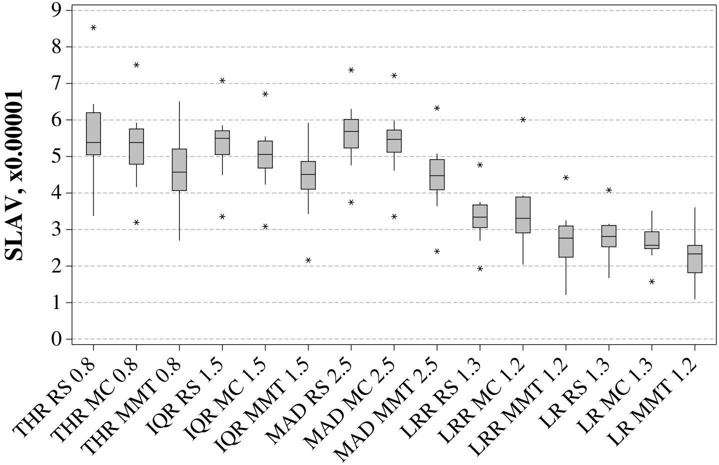
I Wish I Used * From the Beginning of my PhD

- Linux / Xmonad
- Emacs
- LaTeX
- R
- Python (more productive than Java)
- Git
- Haskell – still have not started using…

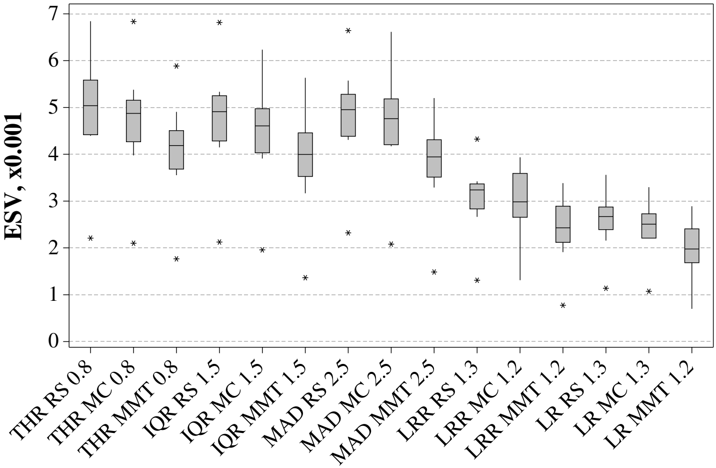
SIMULATION RESULTS: ENERGY



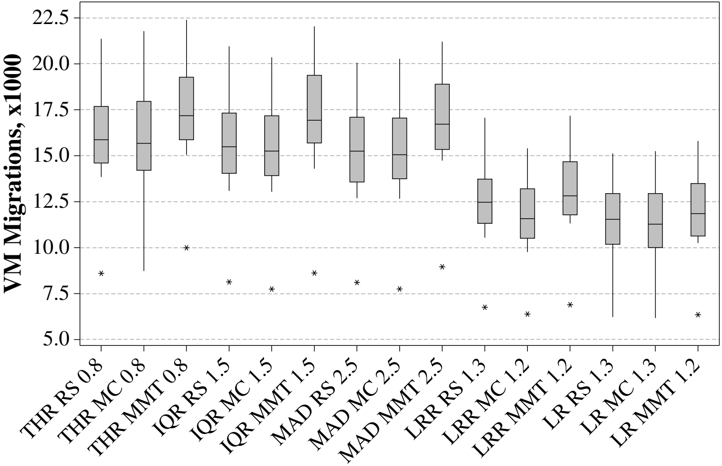
SIMULATION RESULTS: SLAV



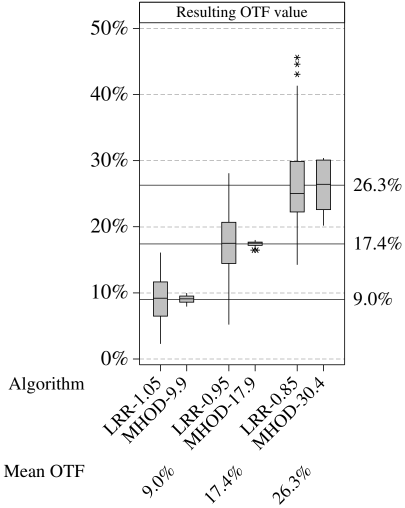
Simulation Results: ESV

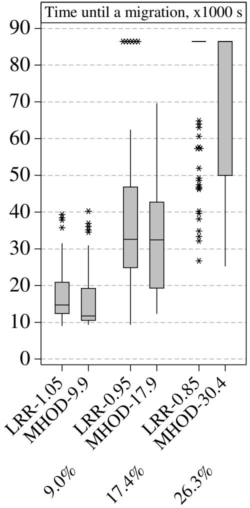


Simulation Results: VM Migrations



SIMULATION RESULTS: MHOD VS LRR





Algorithms: Host Underload Detection

The averaging threshold-based underload detection algorithm:

Input: *threshold*, *n*, *utilization*
Output: Whether the host is underloaded

1: **if** *utilization* is not empty **then**
2: *utilization* $\leftarrow$ last *n* values of *utilization*
3: *meanUtilization* $\leftarrow$ sum(*utilization*) / len(*utilization*)
4: **return** *meanUtilization* $\leq$ *threshold*
5: **return** **false**

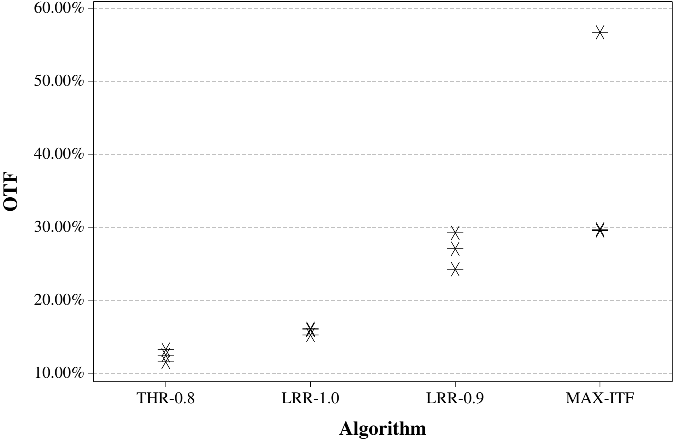
Algorithms: VM Selection

The min migration time max CPU utilization algorithm:

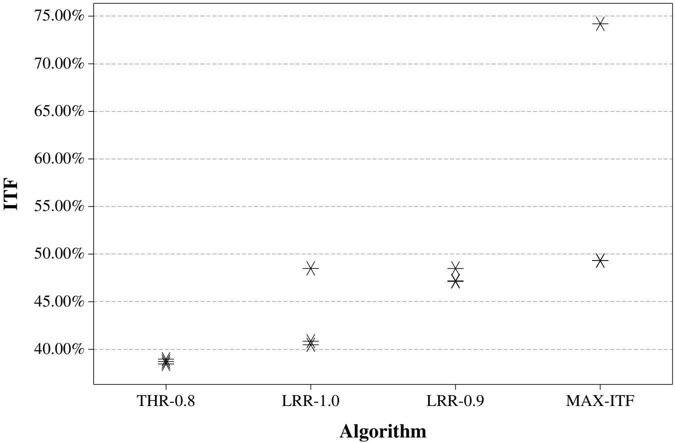
Input: n, $vmsCpuMap$, $vmsRamMap$
Output: A VM to migrate

```
minRam ← min(values of vmsRamMap)
maxCpu ← 0
selectedVm ← None
for vm, cpu in vmsCpuMap do
    if vmsRamMap[vm] > minRam then
        continue
    vals ← last n values of cpu
    mean ← sum(vals) / len(vals)
    if maxCpu < mean then
        maxCpu ← mean
        selectedVm ← vm
return selectedVm
```

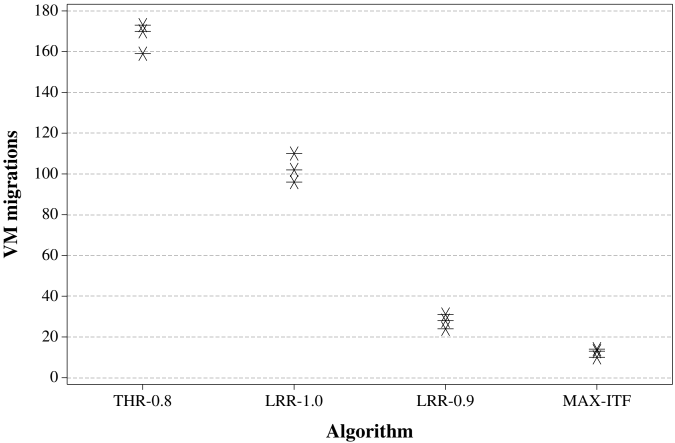
Experimental Results: AOTF



Experimental Results: AITF



Experimental Results: VM Migrations



Experimental Results: Energy Consumption

- Server power consumption [Meisner, 2009]:
  - 450 W – the fully utilized state
  - 270 W – the idle state
  - 10.4 W – the sleep mode

Algorithm	Energy, kWh	Base energy, kWh	Energy savings
THR-0.8	25.18	34.39	26.80%
LRR-1.0	23.51	33.76	30.36%
LRR-0.9	22.23	33.16	32.98%
MAX-ITF	18.76	31.78	40.96%