Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers

Anton Beloglazov and Rajkumar Buyya
CLOUDS Lab, Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia

MGC '10 Proceeding of the 8th International Workshop on Middleware for Grids, Clouds and e-Science

Outline

- Introduction
- Related Work
- System Model
- Allocation Policies
- Evaluation
- Conclusion and Future Work

Introduction

- The high energy consumption of Cloud is not just in the amount of computing resources used, but rather lies in the inefficient usage of these resources.

- But efficient resource management in Cloud is not trivial: The workload is highly variable, causing dynamic resource usage patterns.

- We propose a novel approach for dynamic consolidation of VMs, which is able to reduce energy consumption and maintain the level of SLA violation as low as 1%.

Related Work

- Kusic et al. use Limited Lookahead Control (LLC)
  - Simulation-based learning
  - Not heuristic-based

- VMware Distributed Power Management
  - Lower threshold 45% and upper threshold 81%
  - However fixed threshold is not suitable for dynamic and unpredictable system workloads.

System Model (1/4)

- Power Model
  - The power consumption can be described by a linear relationship between the power consumption and CPU utilization even when DVFS is applied.

$$P(u) = k * P_{max} + (1 - k) * P_{max} * u$$

  - Studies show an idle server consumes approximately 70% of the full power consumption.

$$P(u) = P_{max} * (0.7 + 0.3u)$$

System Model (2/4)

- Power Model (cont.)
  - The utilization may change over time due to the workload variability: $u(t)$

$$E = \int_t P\big(u(t)\big)\, dt$$

  - To reduce the energy consumption, our approach is to improve the CPU utilization of physical nodes in a data center.

System Model (3/4)

- Cost of VM Live Migration
  - The average performance degradation can be estimated as approximately 10% of CPU utilization.
  - Each VM migration may cause some SLA violation. Therefore it's crucial to minimize the number of VM migrations.

$$T_{m_j} = \frac{Memory_j}{Bandwidth_j}$$

$$U_{d_j} = 0.1 * \int_{t_0}^{t_0+T_{m_j}} u_j(t)\, dt$$

System Model (4/4)

- SLA Violation Metric

$$SLA = \frac{\sum_{j=1}^{M} \int_t U_{r_j}(t) - U_{a_j}(t)\, dt}{\sum_{j=1}^{M} \int_t U_{r_j}(t)\, dt}$$

A fraction of the difference between the requested MIPS and the actually allocated MIPS relatively to the total requested MIPS.

It represents the percentage of the CPU performance that has not been allocated when demanded relatively to the total demand.

Allocation Policies (1/5)

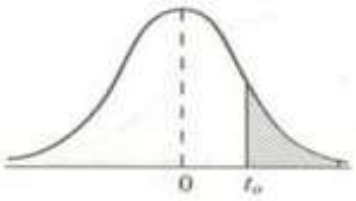
- Fixed Utilization Thresholds
  - Single Threshold (ST)
    - Keeping the total utilization of CPU below this threshold.
  - Minimization of Migrations (MM)
  - Highest Potential Growth (HPG)
  - Random Choice (RC)
    - Upper and lower threshold for hosts
    - Keeping the total utilization of CPU between these thresholds.
    - If falls below the lower threshold, all VMs have to be migrated from this host and the host has to be switched off.
    - If exceeds the upper threshold, some VMs have to be migrated from this host to reduce the CPU utilization.

Allocation Policies (2/5)

- Dynamic Utilization Thresholds
  - CPU utilization of each VM can be described by u_j. (Sample history records)
  - Use *Student's t-distribution* to model the distribution of the CPU utilization of hosts.
  - Upper threshold for host i
    - $\overline{U_i} = \sum_{j=1}^{m} \overline{u_j}, S_{U_i} = \sqrt{\sum_{j=1}^{m} s_{u_j}^2}$
    - $T_{u_i} = 1 - \left(\left(t_{n-1}(P_{uu}) * S_{U_i} + \overline{U_i}\right) - \left(t_{n-1}(P_{ul}) * S_{U_i} + \overline{U_i}\right)\right)$
  - Lower threshold for all hosts
    - $\overline{U} = \frac{1}{N} \sum_{i=1}^{N} \overline{U_i}, S_U = \frac{1}{N} \sqrt{\sum_{i=1}^{N} (\overline{U_i} - \overline{U})^2}$
    - $T_l = \begin{cases} \overline{U} - t_{n-1}(P_l) * S_U, if < U_l \\ U_l, otherwise. \end{cases}$
    - We set $U_l = 30\%$ found by previous work

Allocation Policies (3/5)

A.3 t 分配之臨界值



ν	α 0.10	0.05	0.025	0.01	0.005
1	3.078	6.314	12.706	31.821	63.657
2	1.886	2.920	4.303	6.965	9.925
3	1.638	2.353	3.182	4.541	5.841
4	1.533	2.132	2.776	3.747	4.604
5	1.476	2.015	2.571	3.365	4.032
6	1.440	1.943	2.447	3.143	3.707
7	1.415	1.895	2.365	2.998	3.499
8	1.397	1.860	2.306	2.896	3.355
9	1.383	1.833	2.262	2.821	3.250
10	1.372	1.812	2.228	2.764	3.169
11	1.363	1.796	2.201	2.718	3.106
12	1.356	1.782	2.179	2.681	3.055
13	1.350	1.771	2.160	2.650	3.012
14	1.345	1.761	2.145	2.624	2.977
15	1.341	1.753	2.131	2.602	2.947
16	1.337	1.746	2.120	2.583	2.921
17	1.333	1.740	2.110	2.567	2.898
18	1.330	1.734	2.101	2.552	2.878
19	1.328	1.729	2.093	2.539	2.861
20	1.325	1.725	2.086	2.528	2.845
21	1.323	1.721	2.030	2.518	2.831
22	1.321	1.717	2.074	2.508	2.819
23	1.319	1.714	2.069	2.500	2.807
24	1.318	1.711	2.064	2.492	2.797
25	1.316	1.708	2.060	2.485	2.787
26	1.315	1.706	2.056	2.479	2.779
27	1.314	1.703	2.052	2.473	2.771
28	1.313	1.701	2.048	2.467	2.763
29	1.311	1.699	2.045	2.462	2.756
inf.	1.282	1.645	1.960	2.326	2.576

Ex:

$90\% = 1 - 0.9 = 0.10$

$t_6(0.10) = 1.440$

Allocation Policies (4/5)

- VM Reallocation

Algorithm 1: Dynamic Thresholds (DT)

```
1  Input: hostList, vmList   Output: migrationList
2  vmList.sortDecreasingUtilization()
3  foreach h in hostList do
4      hUtil ← h.util()
5      bestFitUtil ← MAX
6      while hUtil > h.upThresh() do
7          foreach vm in vmList do
8              if vm.util() > hUtil − h.upThresh() then
9                  t ← vm.util() − hUtil + h.upThresh()
10                 if t < bestFitUtil then
11                     bestFitUtil ← t
12                     bestFitVm ← vm
13             else
14                 if bestFitUtil = MAX then
15                     bestFitVm ← vm
16                 break
17         hUtil ← hUtil − bestFitVm.util()
18         migrationList.add(bestFitVm)
19         vmList.remove(vm)
20     if hUtil < lowThresh() then
21         migrationList.add(h.getVmList())
22         vmList.remove(h.getVmList())
23 return migrationList
```

Allocation Policies (5/5)

- VM Placement
  - MBFD (Modified Best Fit Decreasing)

Algorithm 2: Modified Best Fit Decreasing (MBFD)

```
1  Input: hostList, vmList   Output: allocation of VMs
2  vmList.sortDecreasingUtilization()
3  foreach vm in vmList do
4      minPower ← MAX
5      allocatedHost ← NULL
6      foreach host in hostList do
7          if host has enough resource for vm then
8              power ← estimatePower(host, vm)
9              if power < minPower then
10                 allocatedHost ← host
11                 minPower ← power
12     if allocatedHost ≠ NULL then
13         allocate vm to allocatedHost
14 return allocation
```

Evaluation (1/7)

- CloudSim 2.0
  - 1 data center
  - 1500 heterogeneous physical nodes
    - 1 CPU core (2000, 2500, 3000, 3500 MIPS)
    - 16GB RAM
    - 10GB/s network
    - 1TB storage
  - 500 heterogeneous VM requests
    - 1 CPU core (1000, 2000, 2500, 3250 MIPS)
    - 1GB RAM
    - 100Mb/s network
    - 1GB storage
  - Each VM uses CoMon project real workload data randomly from one of the servers.

Evaluation (2/7)

- Parameters of the dynamic thresholds
  - P_l: Probability for the lower threshold from 90% to 99%
  - P_{ul}: The lower bound of the probability interval for upper threshold from 90% to 98%
  - P_{uu}: The upper bound of the probability interval for upper threshold from 95% to 99.9%
- Compare two paired parameters
  - DT-90-90-95
    - The least energy consumption
  - DT-99-98-99.9
    - The least SLA violation and number of VM migrations

Evaluation (3/7)

Table 1: DT-99-98-99.9 against DT-90-90-95

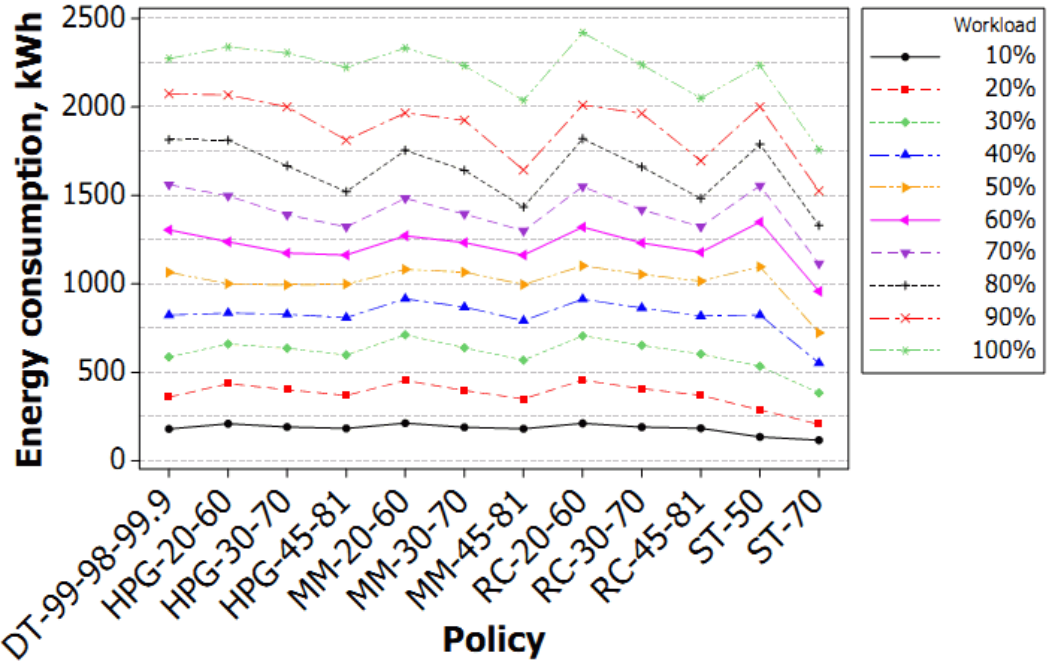
Parameters	Energy	SLA viol.	VM migr.
99-98-99.9	1204 kWh	**0.96%**	**20577**
90-90-95	**1154 kWh**	1.47%	37758
Difference	50 kWh	-0.51%	-17180

Evaluation (4/7)

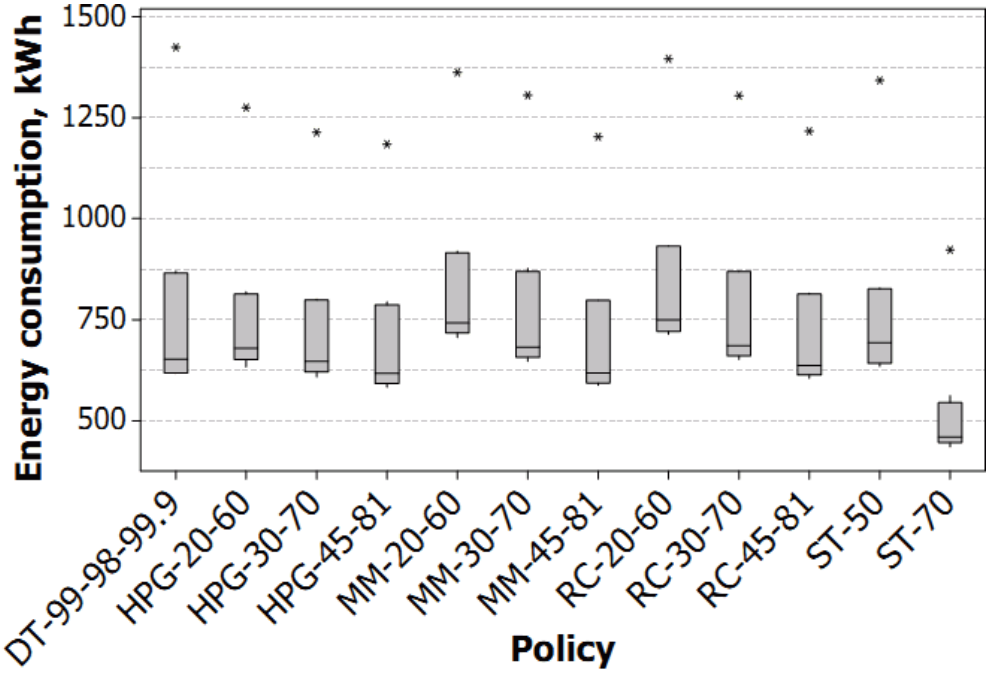
Table 2: DT-99-98-99.9 against the other algorithms

Algorithm	Energy	SLA viol.	VM migr.
DT-99-98-99.9	1204 kWh	**0.96%**	**20,577**
MM-20-60	1218 kWh	3.15%	100,339
MM-30-70	1158 kWh	3.73%	101,661
MM-45-81	1046 kWh	5.88%	125,199
ST-50	1180 kWh	3.60%	270,766
ST-60	**866 kWh**	5.8%	319,991
DVFS	1847 kWh	–	–
NPA	8975 kWh	–	–

Evaluation (5/7)

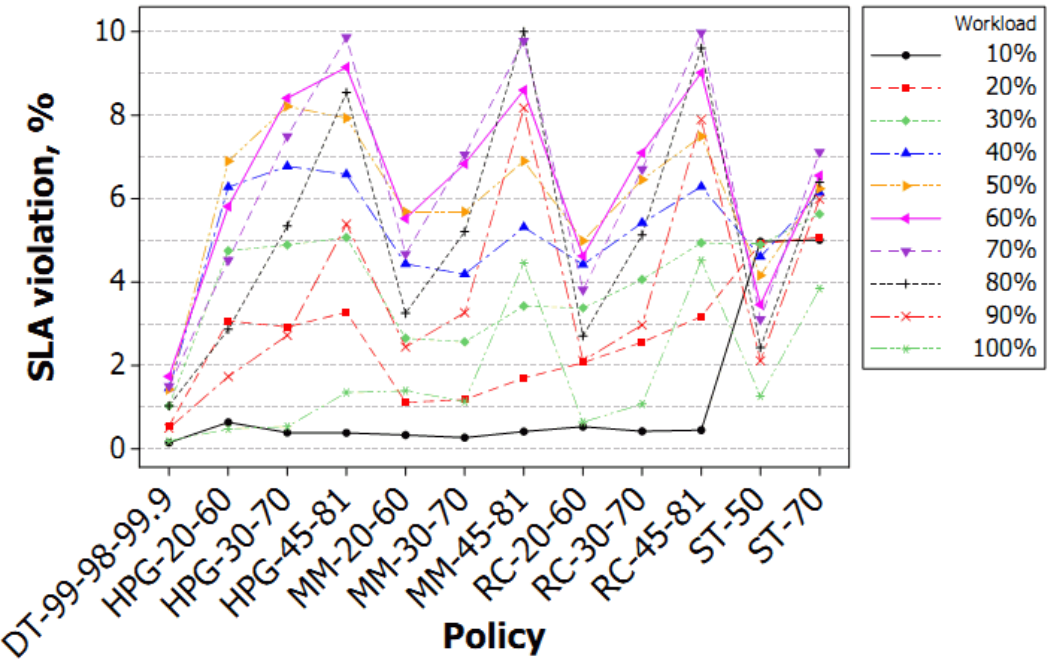


(a) Energy consumption (categories)

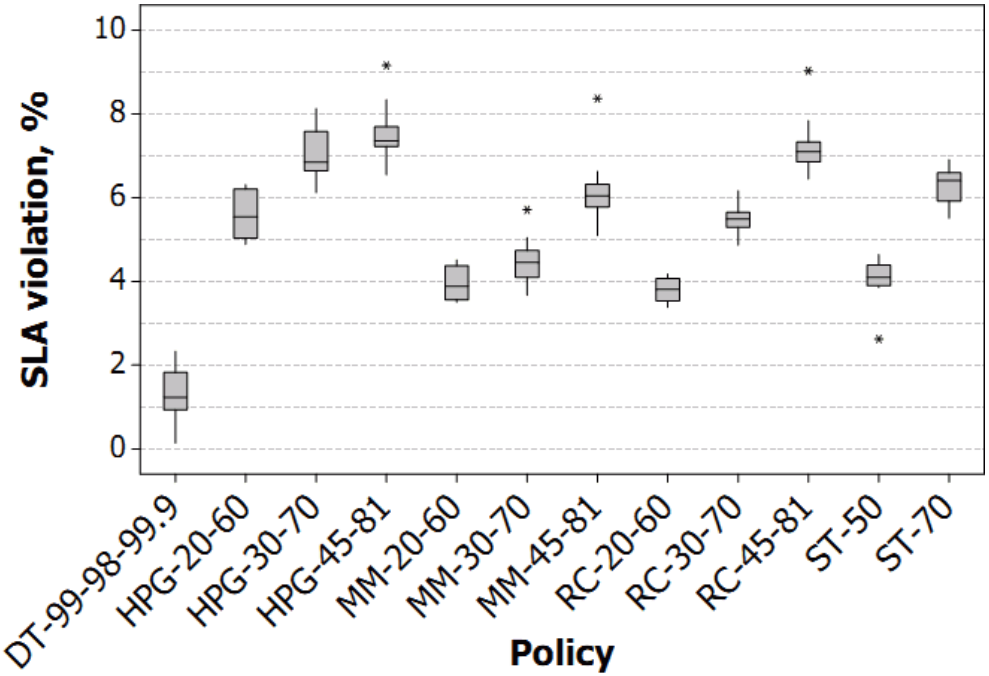


(b) Energy consumption (daily)

Evaluation (6/7)

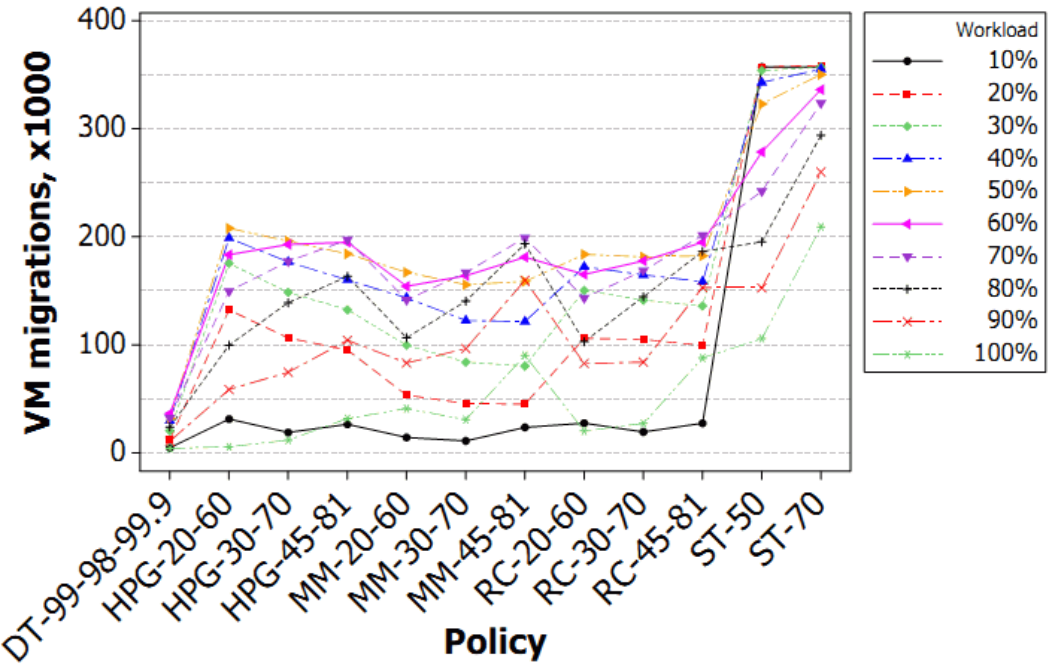


(c) SLA violation (categories)

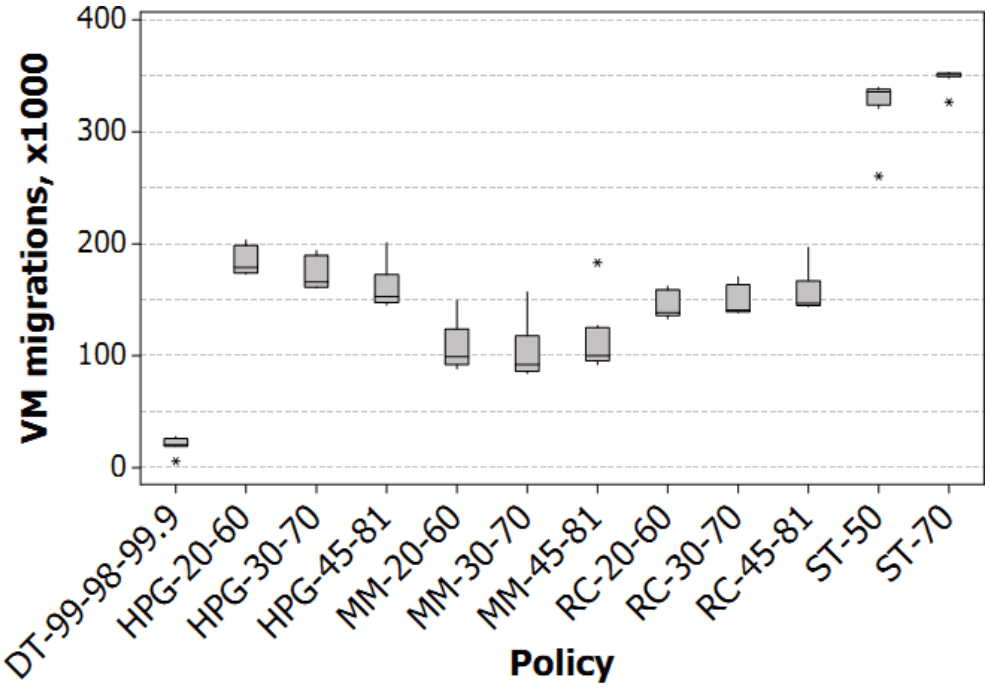


(d) SLA violation (daily)

Evaluation (7/7)



(e) VM migrations (categories)



(f) VM migrations (daily)

Conclusion and Future Work

- We have proposed a novel technique for the energy-efficient threshold-based dynamic consolidation of VMs with auto-adjustment of the threshold values.

- The experimental results show the proposed technique outperforms other policies in terms of SLA violation and number of VM migrations.

- Future work focuses on multi-core CPU architectures, and multiple system resources such as memory and network.

- We plan to implement it on a real-world Cloud platform, such as Aneka.