Resource Management in Cloud Computing Systems
A 40+ member group. Past and current funding from the Australian Research Council, CISCO, ERICSSON, IBM, Microsoft, Sun, Smart Internet CRC, NICTA, DSTO and CSIRO.

The Centre’s mission is to establish a **streamlined research, technology exploration and advanced training program**. It will be a leading centre to undertake collaborative multi-disciplinary research in support of distributed and high performance computing and related industry to enable advances in information technology and other application domains.

The Centre focuses currently on several themes which build on existing strengths at Sydney University:

- Algorithmics and Data Mining
- Cloud Computing and Green ICT
- Internetworking
- Service Computing
- Distributed Computing Applications
Outline

› Resource Abundance in Clouds
  - Source of inefficiency or opportunity of efficiency?
  - Inefficiency of current practices in resource management

› Holistic Approach to Optimization of Cloud Efficiency
  - Data center level efficiency
  - Individual node/resource level efficiency
  - Capturing trade-off between cost and performance

› Conclusion
Take Home Message

Source: http://www.flickr.com/photos/56104473@N04/5190273185/sizes/l/in/photostream/
There is a need for different usage/application models for cloud computing environments.

Resource allocation in clouds involves a number of very complex issues that will be around for some time.

A fertile research area with many directions.
Gartner’s Strategic Technology Trends for 2015

Merging the Real World and the Virtual World
- Computing Everywhere
- The Internet of Things
- 3D Printing

Intelligence Everywhere
- Advanced, Pervasive, and Invisible Analytics
- Context-Rich Systems
- Smart Machines

The New IT Reality Emerges
- Cloud/Client Computing
- Software-Defined Applications and Infrastructure
- Web-Scale IT
- Risk-Based Security and Self-Protection

gartner.com/SmarterWithGartner

© 2014 Gartner, Inc. and/or its affiliates. All rights reserved.
‘Efficiency’ of Resource Abundant Clouds

› Resource Efficiency ≠ Resource Utilization

› Definition of ‘Efficiency’
  - Minimum resource provisioning level
  - Maximum resource utilization
  - Meeting performance requirements (or high performance/throughput)

› Resource utilization
  - The number of active resources over time (system/data-center level)
  - The actual resource usage (e.g., CPU utilization)

› We aim to identify the minimum level of resource provisioning that maximizes resource utilization meeting performance requirements
Efficiency of ‘Resource Abundant Clouds’

The National Security Administration (NSA) data center

Google data center
Efficiency of ‘Resource Abundant Clouds’

**Computer Air Handling Unit (CRAC)**
- Up To 30 Ton Sensible Capacity Per Unit
- Air Discharge Can Be Upflow Or Downflow Configuration
- Downflow Configuration Used With Raised Floor To Create A Pressurized Supply Air Plenum With Felt Supply Diffusers

**Individual Colocation Computer Cabinets**
- Typ. Cabinet Footprint (28”W x 36”D x 84”H)
- Typical Capacities Of 1750 To 3750 Watts Per Cabinet

**Emergency Diesel Generators**
- Total Generator Capacity = Total Electrical Load To Building
- Multiple Generators Can Be Electrically Combined With Paralleling Gear
- Can Be Located Indoors Or Outdoors At Grade Or On Roof
- Outdoor Applications Require Sound Attenuating Enclosures

**Fuel Oil Storage Tanks**
- Tank Capacity Dependent On Length Of Generator Operation
- Can Be Located Underground Or At Grade Or Indoors

**UPS System**
- Uninterruptible Power Supply Modules
- Up To 1000 kVA Per Module
- Cabinets And Battery Strings Or Rotary Flywheels
- Multiple Redundancy Configurations Can Be Designed

**Electrical Primary Switchgear**
- Includes Incoming Service And Distribution
- Direct Distribution To Mechanical Equipment
- Distribution To Secondary Electrical Equipment Via UPS

**Heat Rejection Devices**
- Drycoolers, Air Cooled Chillers, Etc.
- Up To 400 Ton Capacity Per Unit
- Mounted At Grade Or On Roof
- N+1 Design

**Pump Room**
- Used To Pump Condenser/Chilled Water Between Drycoolers And CRAC Units
- Additional Equipment Includes Expansion Tanks, Glycol Feed System
- N+1 Design (Standby Pump)

**Power Distribution Unit (PDU)**
- Typical Capacities Up To 223 kVA Per Unit
- Redundancy Through Dual PDU’s With Integral Static Transfer Switch (STS)

**Colocation Suites**
- Modular Configuration For Flexible Suite Sq.Ft. Areas
- Suites Consist Of Multiple Cabinets With Secured Partitions (Cages, Walls, Etc.)
Inefficiency of Current Practices: Data Center Level

- Data center utilization is mostly below 10%\(^1\) due to over-provisioning
- Idle servers still consume more than 50% of peak power draw\(^2\)
- Average lifespan of servers is 3 years
- Energy costs are soaring
- Public cloud services are often charged by resource hours (partial hours are a source of cost inefficiency)

---


Inefficiency of Current Practices: Data Center Level

walmart.com

sydney.edu.au

naver.com
Inefficiency of Current Practices:
Individual Resource Level

**CPU utilization of scientific workflow**

**CPU utilization of MapReduce job**

**Write rate (I/O resource usage) of MapReduce job**
Inefficiency of Current Practices: individual resource level

Visualization of executing Montage astronomical scientific workflow
Ways to Improve Efficiency: Data Center Level

- Dynamic, adaptive resource provisioning by exploiting **elasticity** in the cloud

Source: Energy Efficiency and Cloud Computing by D. Patterson in Microsoft Research Faculty Summit 2009
Optimizing Clouds

Source: http://www.flickr.com/photos/ibm_media/2071286721/
Optimizing the Efficiency of Clouds: Our Solutions

Resource Efficient Workflow Scheduling


High Performance/Throughput Computing Applications


Many applications in science and engineering are becoming increasingly large-scale and complex.

These applications are often amalgamated in the form of workflows.

**Montage:**
- Astronomical image mosaic engine

**Epigenomics:**
- Genome sequence processing

**CyberShake:**
- Earthquake hazards characterization

**SIPHT:**
- Search for untranslated RNAs (sRNAs)
Many applications in science and engineering are becoming increasingly large-scale and complex.

These applications are often represented in the form of workflows.

1 worker node for 1000 hours ≠ 1000 worker nodes for 1 hour.
Optimizing the Efficiency of Clouds: Resource Efficient Workflow Scheduling

Resource allocation and scheduling with abundant resources
Running scientific workflows

- Montage: an astronomical image mosaic engine
  - stitches together multiple input images to create custom mosaics of the sky
  - A 6.0 Degree Montage workflow contains 8,596 jobs, 1,444 input files with a total size of 4.0 GB and 22,850 intermediate files with a total size of 35GB.
Running scientific workflows

- How many resources are needed for a given workflow application?
Optimizing the Efficiency of Clouds: Resource Efficient Workflow Scheduling

Traditionally

[Diagram with workflow and cloud servers]
Traditionally
Optimizing the Efficiency of Clouds: Resource Efficient Workflow Scheduling
Today
Resource efficient solution
Workflow scheduling with abundant resources

- How many resources are needed for a given workflow application?
- \#resources used tends to be dominated by the (maximum) width of DAG
Our solution (stretch out and compact)

- CPF (Critical Path First): stretch out the schedule to preserve critical path length (the shortest possible time of completion) using as many resources.

- MER (Maximum Effective Reduction): Compact the schedule by rearranging tasks making use of idle/inefficiency slots present due to precedence constraints.
Stretch out: Critical Path First (CPF)

- Critical path length can be proactively preserved by assigning all CP tasks on a particular resource (or CP resource) ‘at the beginning’ and then scheduling remaining tasks.
Schedule compaction (Maximum Effective Reduction or MER)

- Makespan minimization and resource usage reduction are conflicting objectives
- Resource efficiency can be improved by resolving (or at least relieving) the conflict
- **How?**
  - The inefficiency in resource usage of workflow schedule (i.e., idle slots) should be better exploited
Optimizing the Efficiency of Clouds: Resource Efficient Workflow Scheduling

Schedule compaction (Maximum Effective Reduction or MER)
- The difference between resource usage reduction (RUR) and makespan increase (MI) in a resulting consolidated schedule as compared to the original output schedule

\[
\text{Effective Reduction (ER)} = \frac{|R^0| - |R^*|}{|R^0|} - \frac{|ms^*| - |ms^0|}{|ms^0|}
\]

- $|R^0|$: resources used in the original schedule
- $|R^*|$: resources used in the consolidated schedule
- $ms^0$: the original makespan
- $ms^*$: the makespan after consolidation
Experimental Evaluation
- Intel 40-core machine with 4 10-core Intel 2.4GHz Xeon processors
- Five real-world scientific workflows (50 - 6,000 tasks/job)
  - CyberShake, Epigenomics, LIGO, Montage and SIPHT

Evaluation metrics
- Makespan
- #Resources used
- Algorithm running time
Results: Makespan increase w.r.t resource usage reduction
Results: effective reduction w.r.t. different apps and algorithms
› Results: scheduling time

The diagram shows the results of scheduling time for various workflows and compaction overheads. The x-axis represents different workflows: CyberShake, Epigenomics, LIGO, Montage, and SIPHT. The y-axis represents scheduling time in milliseconds (ms). The overheads include EFT, CPF, CPOP, and DCP compaction overheads.
Optimizing the Efficiency of Clouds:
Our Solutions

› Resource Efficient Workflow Scheduling


› High Performance/Throughput Computing Applications


Scientists need to run these workflows with different parameters repeatedly, or use a combination of different workflows to achieve an ultimate goal.

A workflow ensemble represents an entire scientific analysis as a set of interrelated but independent workflow applications.

An ensemble of 200 6.0 degree Montage workflows:
- 1,717,200 jobs
- 288,800 input files and 4,570,000 intermediate files, and
- Approximately 7 TB data footprint

We need an efficient “cloud-ready” workflow execution system for effectively dealing with resource allocation, data staging and execution coordination.
Optimizing the Efficiency of Clouds: Executing Large-scale Workflow Ensembles

› DEWE (Distributed Elastic Workflow Execution)
  - Open-source project supported by AWS Education Research Grant (https://bitbucket.org/lleslie/dwf/wiki/Home)
DEWE (Distributed Elastic Workflow Execution)

- The workflow visualization toolkit takes a workflow execution trace file as the input, and produces a scalable vector graph (SVG) or PDF representing the resource consumption status during the execution.
Optimizing the Efficiency of Clouds: Executing Large-scale Workflow Ensembles

DEWE (Distributed Elastic Workflow Execution)

- The workflow visualization toolkit takes a workflow execution trace file as the input, and produces a scalable vector graph (SVG) or PDF representing the resource consumption status during the execution.
DEWE vs. Pegasus (well-known workflow execution system)

- Resource consumption of multiple 6.0 degree Montage workflows on Amazon EC2 c3.8xlarge instance
DEWE evaluation

- Node Performance Index \( P \) is used after profiling

\[
P = \frac{W}{N \times T}
\]

\( W \): the number of workflows

\( N \): the number of worker nodes

\( T \): the execution time needed for \( N \) workflows

Then, we can estimate the number of worker nodes needed to execute a large scale workflow ensemble with deadline constraints using the following formula:

\[
N = \frac{W}{P \times T}
\]
DEWE evaluation

- Cluster configurations

<table>
<thead>
<tr>
<th>Cluster</th>
<th>#Nodes</th>
<th>#vCPUs</th>
<th>Memory (TB)</th>
<th>Storage (TB)</th>
<th>Price (USD/hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3.8xlarge</td>
<td>40</td>
<td>1280</td>
<td>2.40</td>
<td>25.6</td>
<td>67.2</td>
</tr>
<tr>
<td>r3.8xlarge</td>
<td>25</td>
<td>800</td>
<td>6.10</td>
<td>16.0</td>
<td>70.0</td>
</tr>
<tr>
<td>i2.8xlarge</td>
<td>23</td>
<td>768</td>
<td>5.61</td>
<td>147.2</td>
<td>156.7</td>
</tr>
<tr>
<td>i2.8xlarge B</td>
<td>10</td>
<td>320</td>
<td>2.44</td>
<td>64.0</td>
<td>68.2</td>
</tr>
</tbody>
</table>

- Workflows
  - 50 - 200 6.0 degree Montage workflows
  - Deadline constraint: 1 hour
DEWE evaluation

- Results:
  - By adopting the pulling approach in our solution system, much of scheduling overhead can be removed as a majority of tasks in scientific workflows often exhibit homogeneity in their resource consumption pattern and acquiring a large number of homogeneous public cloud resources is easily possible.
  - 80% speed-up compared to Pegasus
  - Cost and deadline compliance can be achieved
Optimizing the Efficiency of Clouds: Our Solutions

› Resource Efficient Workflow Scheduling


› High Performance/Throughput Computing Applications


Why cloud bursting?

- Many organizations already operate their own computing facilities, called private clouds or data centres.

- Multi-cloud model is practical and realistic in many scenarios:
  - Security is a major concern (compared to cloud sourcing)
  - Workloads exhibit different characteristics
  - Sporadic workload surges occur (a major source of over provisioning, inefficient resource usage)
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

› Tools for cloud bursting

- Eucalyptus
- OpenStack
- OpenNebula.org
- Ganeti
- Bright Computing
Different users have a diverse set of applications possibly with different objectives, e.g., performance/time, cost, etc.

Cloud providers offer a number of different services:
- E.g., Standard, High-CPU, High-Memory, Compute Cluster, GPU Cluster

Usage is typically charged by the hour.

Cost to performance ratio (cost efficiency) may vary significantly by scheduling and resource allocation.
Private system often gets overwhelmed by resource requirement of bag-of-tasks (BoT) applications

- BoT applications are common in science and engineering
  - Monte Carlo simulations
  - CycleCloud: more than 10 machine years
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

Cloud bursting with BoT applications

- Multi-cloud model
- Public and private cloud resources: \( \{s_1, s_2, \ldots, s_k\} \) and \( \{c_1, c_2, \ldots, c_k\} \)
- BoT application model
- Set of \( n \) tasks
- \( P_i \): amount of time required to complete, unknown in advance
- If task \( j \) run on machine \( i \), it takes \( P_j / s_i \) to finish.

- Objective function
  - User has two conflicting objectives of minimizing cost and maximizing performance (minimizing makespan)
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

- Closer look to objective function
  - Pareto optimality effectively captures the trade off between two conflicting objectives
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

- **PANDA (PAreto Nnear-optimal Deterministic Aproximation)**
  - A fully polynomial time approximation scheme (FPTAS) with input size $n$ and approximation factor $\varepsilon$

- **Four major steps**
  - Pre-processing
    - Tasks are pre-processed for their lengths to be equalized
  - Task selection with trimming
    - Tasks are selected by solving subset sum problem
  - Task assignment
    - Each machine gets its workload (optimal #tasks)
  - Solution refinement
    - A task currently assigned to a slow resource is moved to a faster resource such that the time required by the faster resource does not incur any extra cost
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

› Optimal task assignment: **integer programming**

\[
\begin{align*}
\min z &= \sum_{i \in \Gamma} L_i c_i \left[ \frac{x_i P}{s_i} \right] \frac{x_i P}{s_i} + L_v c_v \left( \frac{x_v P}{s_v} \right)^2 \\
\text{s.t.} \quad \sum_{i \in \Gamma} L_i x_i + L_v x_v &= n \\
x_v, x_i &\in \mathbb{Z}^\geq 0
\end{align*}
\]

› Optimal solution for relaxed problem:

\[
x_i = \frac{nL_i}{\alpha_i} \sum_{j \in \Gamma \cup \{v\}} L_j^2 \frac{\alpha_j}{\alpha_i}, \quad \forall i \in \Gamma \cup \{v\}
\]

\[
\alpha_i = L_i c_i P^2 / s_i^2, \text{ for } i \in \Gamma \cup \{v\}
\]
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

Algorithm 3: Approximate task assignment

```
input : n, ε, π = \{P_1, P_2 \cdots P_n\}, L_i, c_i, s_i; \forall i \in \Gamma \cup \{v\}
output: \pi_i; a partition scheme of π while the sum of numbers in \pi_i approximates \(x_i^*\);

begin
   Set \(\mu\) to a small real number, e.g., 1;
   Let \(n^* = \sum_{i=1}^{n} \frac{P_i}{\mu}\);
   \(x^* \leftarrow FindOptimal(n^*, \mu, L_i, c_i, s_i); \quad //\text{Algorithm 1}\)
   Sort(\(x^*,\text{descending}\));
   \(\pi' \leftarrow \pi;\)
   for \(i = 1 \cdots k + 1\) do
      \(\ell_0 \leftarrow \langle 0 \rangle\)
      for \(j = 1 \cdots |\pi'|\) do
         \(\ell_j \leftarrow MergeList(\ell_{j-1}, \ell_{j-1} + P_j)\)
         \(\ell_j \leftarrow Trim(\ell_j, \frac{\epsilon}{2|\pi'|}) \quad //\text{Algorithm 2}\)
         Remove elements from \(\ell_j\) for which the size is greater than \(\frac{x_i^*}{1 - \frac{c_i^*}{2|\pi'|}}\)
      end
   end
   Let \(\pi_i^*\) be the nearest value to \(x_i^*\) in \(\ell_j\);
   \(\pi_i \leftarrow\text{set of numbers whose sum is equal to } \pi_i^*;\)
   \(\pi' \leftarrow \pi' - \pi_i;\)
end
Run the refinement process in Algorithm 1.
```

Pre-processing:

Task selection:

Task assignment

Refinement
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

› Experimental evaluation

› We modeled ISOMAP as a real-world BoT application.
  - consists of tens of thousands of (CPU-intensive) tasks.
  - each task runs for seconds or up to tens of minutes.
  - Job sizes in million seconds (Ms): {1 Ms, 5 Ms, 10 Ms, 17 Ms}

› Multi-cloud setting

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Res. Type</th>
<th>Proc. Capacity</th>
<th>Hourly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>1</td>
<td></td>
<td>$0.080</td>
</tr>
<tr>
<td>Amazon EC2</td>
<td>c1.medium</td>
<td>5</td>
<td>$0.165</td>
</tr>
<tr>
<td>US East (VA)</td>
<td>m1.large</td>
<td>4</td>
<td>$0.320</td>
</tr>
<tr>
<td>US East (VA)</td>
<td>c1.xlarge</td>
<td>20</td>
<td>$0.660</td>
</tr>
<tr>
<td>Private</td>
<td>4x10-core Xeon</td>
<td>10</td>
<td>$0.320</td>
</tr>
</tbody>
</table>
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

- Pareto frontier reached (1) theoretically, (2) by PANDA, and (3) by a modified List heuristic

$L_i = 5$, $\varepsilon = 0.1$, and job size = 10Ms (on m1.small)
Optimizing the Efficiency of Clouds: A Case for HPC/HTC applications

Average values of makespan and total cost with respect to different sizes of BoT applications.

<table>
<thead>
<tr>
<th>BoT size</th>
<th>$L_{ist_{\eta}}$ ms(h)</th>
<th>cost</th>
<th>PANDA ms(h)</th>
<th>cost</th>
<th>Optimal ms(h)</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M.s.</td>
<td>2.2</td>
<td>66.0</td>
<td>1.8</td>
<td>58.5</td>
<td>1.5</td>
<td>58.5</td>
</tr>
<tr>
<td>5M.s.</td>
<td>4.2</td>
<td>118.4</td>
<td>3.6</td>
<td>117.2</td>
<td>3.3</td>
<td>117.2</td>
</tr>
<tr>
<td>10M.s.</td>
<td>5.5</td>
<td>153.5</td>
<td>4.9</td>
<td>146.0</td>
<td>4.5</td>
<td>146.0</td>
</tr>
<tr>
<td>17M.s.</td>
<td>9.7</td>
<td>241.7</td>
<td>8.2</td>
<td>215.6</td>
<td>7.9</td>
<td>192.8</td>
</tr>
</tbody>
</table>

$L_i = 20, \varepsilon = 0.1$ on m1.small
Unknown task execution times

› **PESU (Pareto Efficient Scheduling with Uncertainty)**
  - We devise a dynamic resource allocation solution with a hybrid task running time estimation technique based on a feedback control mechanism

› Three phases
  - Estimation
    - estimates the execution time of each task using existing estimation techniques
  - Pareto-efficient point generation
    - Generates possible Pareto-efficient schedules
  - Resource allocation
    - Allocates resources for the selected Pareto-efficient point
Unknown task execution times

› PESU
Running time estimation

We use existing estimation techniques (e.g., ATOM, Pin, and Valgrind) in an iterative fashion

1. Add several breakpoints to each task
2. Assign an accurate weight to each tool by monitoring and comparing the actual running time of breaking points
3. Divide the whole time horizon into equal intervals
4. At the beginning of each interval, a monitoring phase happens:
   - the actual revealed running time and the estimated running time are compared to evaluate the accuracy of each estimation tool.
Experimental evaluation: Unknown task execution times

› We modeled ISOMAP as a real-world BoT application.

<table>
<thead>
<tr>
<th>Type</th>
<th>No. Tasks</th>
<th>Task Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BoT size, Task Running Time)</td>
<td>$10k \times 2^x$</td>
<td>$2^x$ (minute)</td>
</tr>
<tr>
<td>LS (Large, Short)</td>
<td>$x \sim \text{Wbl}(1.7,2)$</td>
<td>$x \sim \text{U}(0,3)$</td>
</tr>
<tr>
<td>LL (Large, Long)</td>
<td>$x \sim \text{Wbl}(1.7,2)$</td>
<td>$x \sim \text{N}(3.5,3)$</td>
</tr>
<tr>
<td>LM (Large, Mixture)</td>
<td>$x \sim \text{Wbl}(1.7,2)$</td>
<td>$x \sim \text{N}(1.8,3)$</td>
</tr>
</tbody>
</table>

› Multi-cloud setting

<table>
<thead>
<tr>
<th>Cloud</th>
<th>Res. Type</th>
<th>Proc. Capacity</th>
<th>Hourly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>m1.small</td>
<td>1</td>
<td>$0.080</td>
</tr>
<tr>
<td></td>
<td>c1.medium</td>
<td>5</td>
<td>$0.165</td>
</tr>
<tr>
<td>US East (VA)</td>
<td>m1.large</td>
<td>4</td>
<td>$0.320</td>
</tr>
<tr>
<td></td>
<td>c1.xlarge</td>
<td>20</td>
<td>$0.660</td>
</tr>
<tr>
<td>Private</td>
<td>4x10-core Xeon</td>
<td>10</td>
<td>$0.320</td>
</tr>
</tbody>
</table>
Results: **Unknown task execution times**

- Comparison of makespan and cost

**Short tasks**

- **Short + Long tasks**

- **Long**

![Graphs showing comparison of makespan and cost for short tasks, short + long tasks, and long tasks.](image)
Simple ideas, but hard to implement!!!!

Ideas are easy. Implementation is hard.

Guy Kawasaki
Conclusion

› Today, with advances in **VM techniques** and the advent of **multi-/many-core processors**, resources are ever abundant

› Computing and data processing needs continuously increase

› Simply expanding resource capacity has resulted in poor resource utilization, i.e., average data center utilization is 10-30% or less

› Adaptive resource management for typical workloads in clouds are essential
  - Workflows: Maximization of resource utilization with min performance impact
  - HPC/HTC apps: Capturing trade-off between cost and performance
Sample of current research projects

› Cost Efficiency of the Data Centre
  - Cost reductions and profit increases (e.g. game theoretic methods)
  - Pay-as-you-go pricing, pricing dynamics

› Implications of multi tenancy
  - Resource virtualization → Resource contention (migrate VMs?)
  - Current SLAs: only availability (need to consider performance?)

› Scheduling and resource allocation as a cost efficient solution (energy minimization)
  - Exploitation of application characteristics (e.g. data locality, latency, quality of service, execution time)
  - Explicit consideration of user experience/satisfaction
  - Map reducing applications, tuning Map reducible applications.
  - Hybrid clouds, cloud bursting for execution time, energy efficiency, pricing, privacy
Other recent work


Thank you