Exploiting Spatio-Temporal Tradeoffs For Energy-aware MapReduce in the Cloud

Michael Cardosa, Aameek Singh,
Himabindu Pucha, Abhishek Chandra

Present by Leping Wang
3/5/2012

Outline

- **Background and Introduction**
- **System model and Problem Definition**
- **Spatial Placement Algorithm**
- **Spatial-Temporal Placement Algorithm**
- **Evaluation**
- **Conclusions and Future Work**

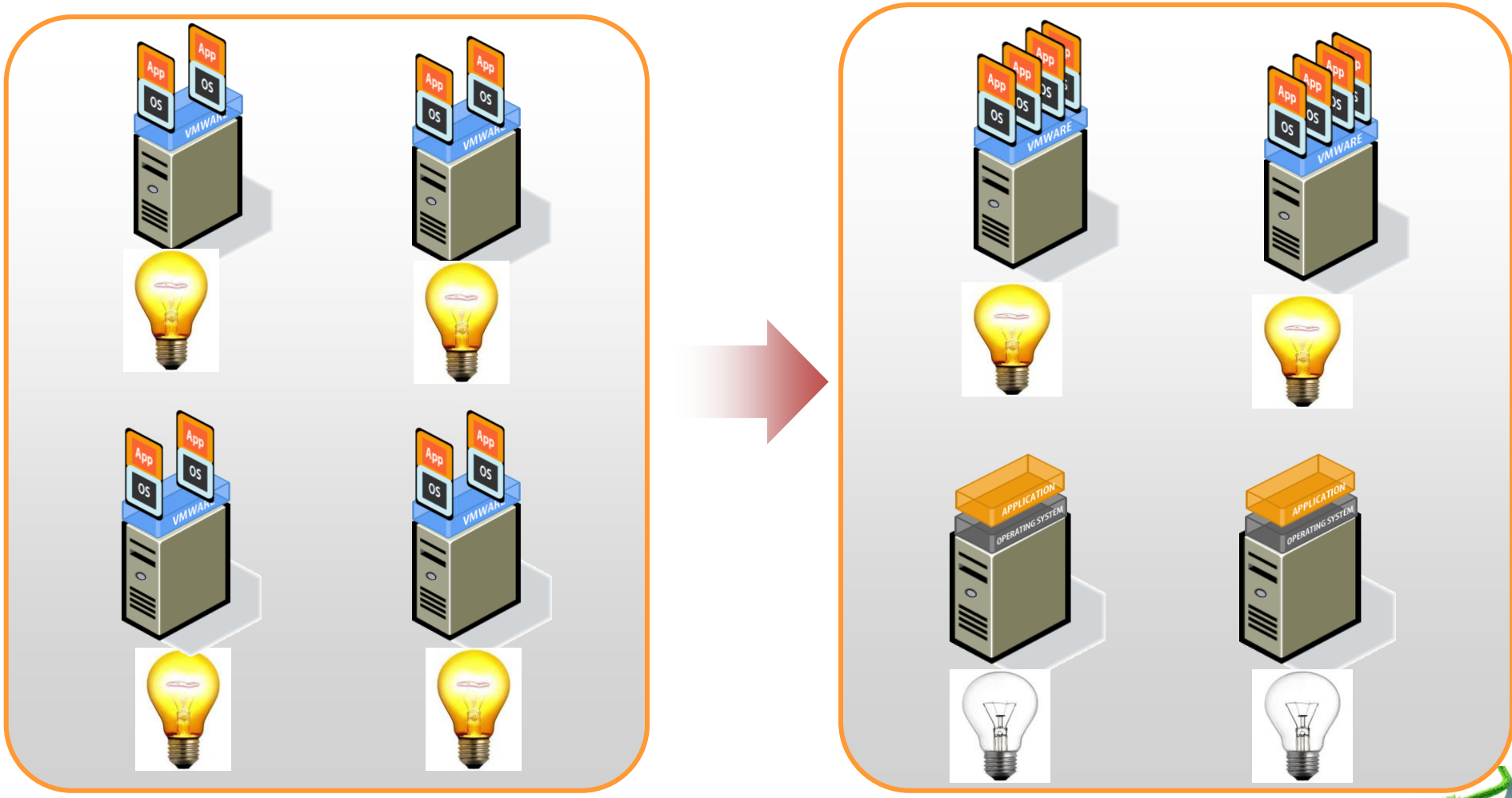
Background and Introduction

- **MapReduce has emerged as the most popular computing paradigm for large-scale data processing and analysis**
- **To improve the energy efficiency of Mapreduce paradigm is very important**
- **Unique opportunity to reduce energy consumption in a cloud environment:**
  - Reduce the amount of available parallelism

    ➔ Reduce the cumulative machines uptime (CMU)

    ➔ Conserve total energy

Background and Introduction



System model and Problem Definition

- **MapReduce jobs run on a virtualized environment**
- **Each job is assigned a type of VM for its cluster.**
- **The cloud operator can transparently increase the size of the virtualized clusters to utilize the physical machines.**
- **Using history data, we can accurately (less than 10% error) estimate the completion time of the jobs.**

System model and Problem Definition

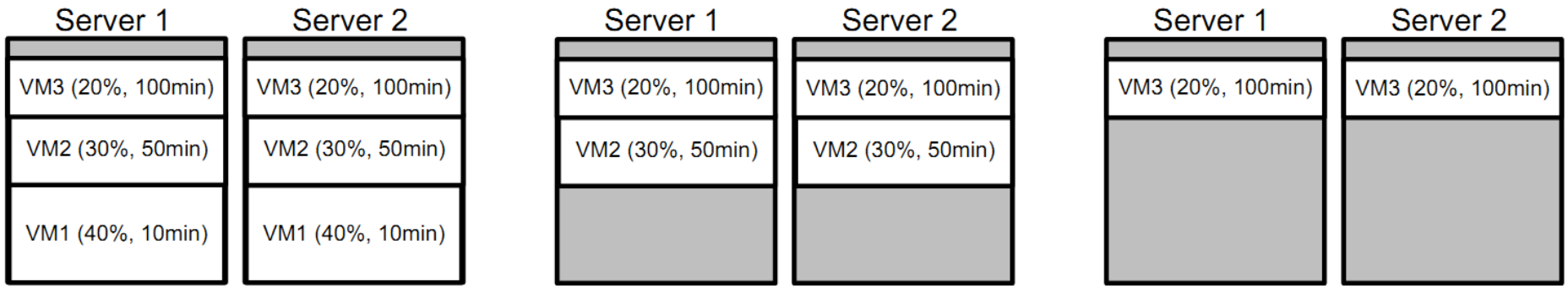
- **Problem Statement:**
  - All machines consume an equal amount of power no matter what utilization a server is running at
  - To control energy by turning machines on/off
  - DVFS is not considered
  - Given a set of Mapreduce jobs, minimized the cumulative machine uptime (CMU) of all physical machines in the cluster

Spatial Placement Algorithm

- **Spatial placement problem:**
  - How to place the MapReduce VMs across the datacenter so as to prevent underutilization and to minimize wasted resources.
  - The spatial placement can maps to a multi-dimensional bin packing problem, hence is NPC
  - Recipe Placement Algorithm:
    - Only a small set of possible VM types
    - Execute exhaustive search to get all possible VM placement configuration
    - One-time computation, the complexity is $O(m^n)$
    - Rank all recipes according to the resource utilization

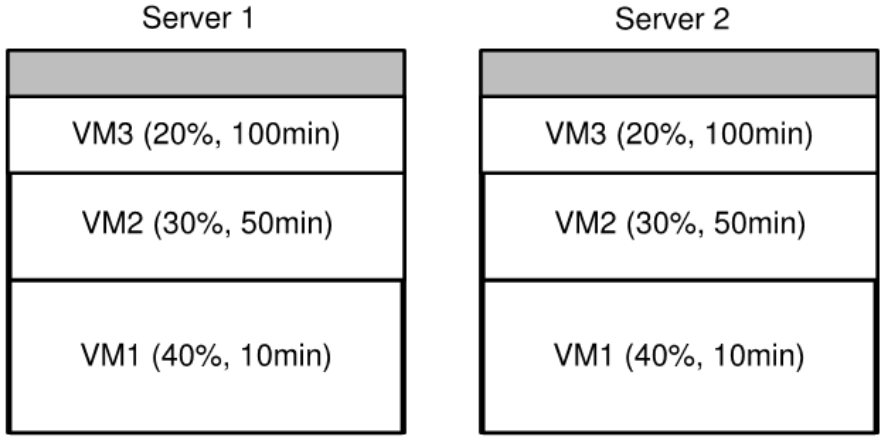
Spatial Placement Algorithm

- **The problem of recipe placement algorithm :**
  - Good at spatial fitting in terms of spare capacity
  - May result in lot of wasted resource if co-placed VMs belong to jobs with widely differing runtimes
- **Examples:**

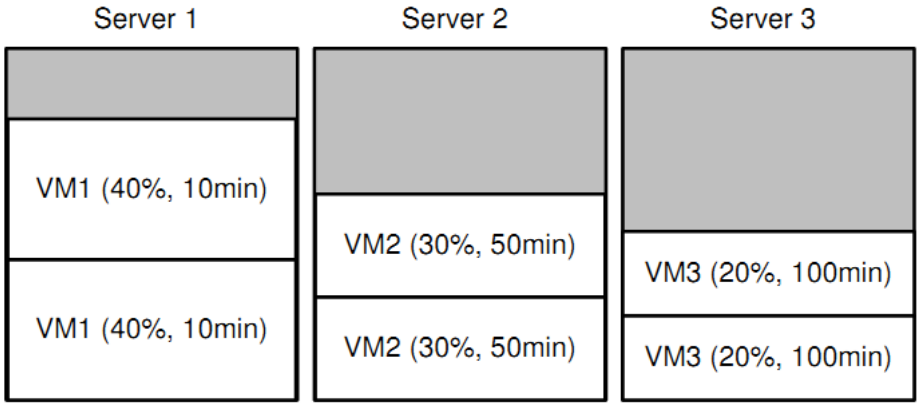


Spatial Placement Algorithm

- **Space-Time Tradeoff in VM Placement**

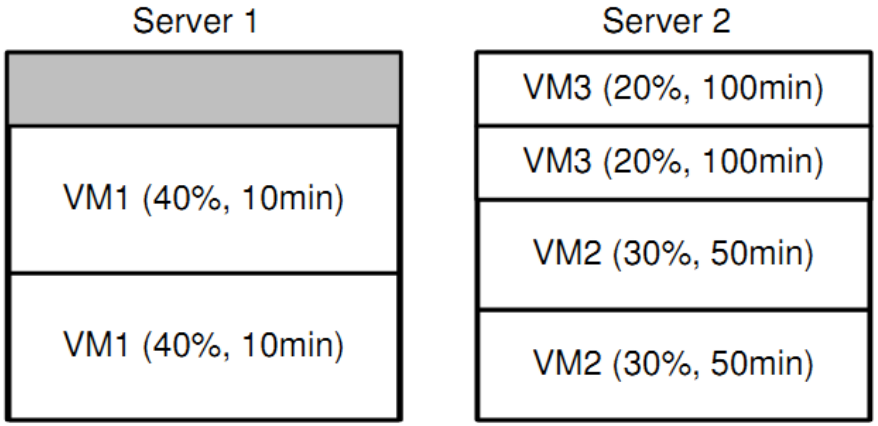


(a) Spatially-efficient VM placement results in wasted resources as jobs finish at different times.



(b) A time-balanced placement algorithm can have a lower CMU than a spatially-efficient algorithm.

CMU = 200



(c) A placement needs to be both spatially-efficient as well as time-balanced to achieve a low CMU.

CMU = 110

CMU = 160

Spatial Placement Algorithm

- **There is a space-time tradeoff in achieving energy-efficiency for MapReduce jobs.**

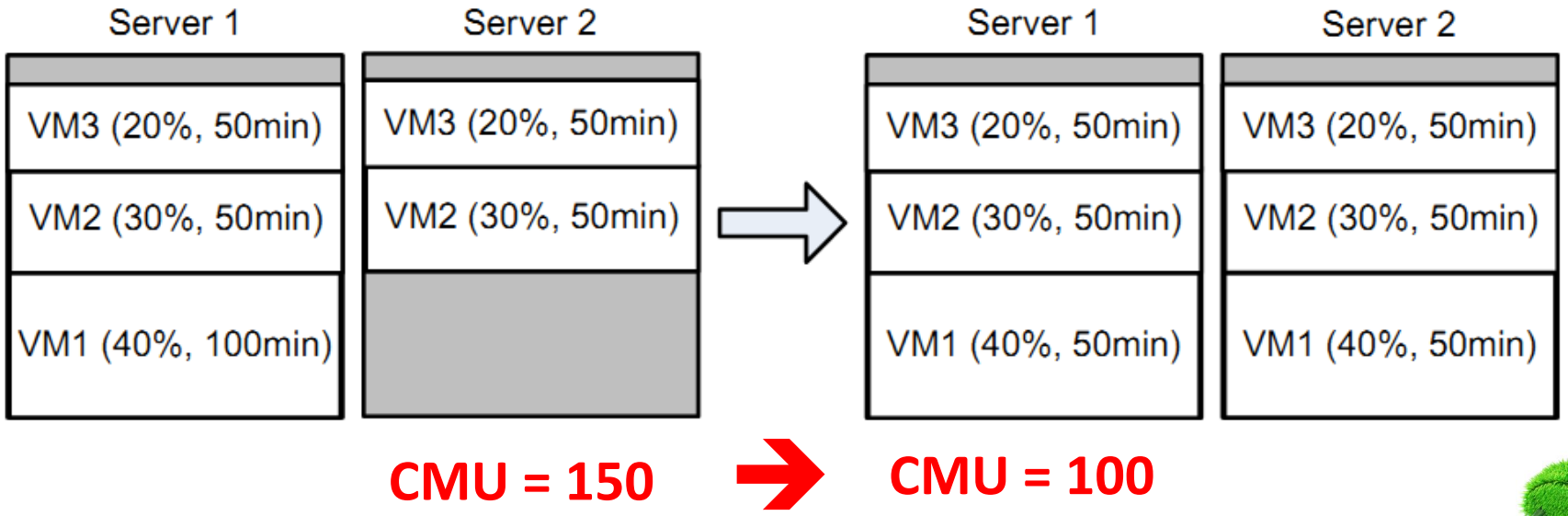
- **I think authors of this paper took improper examples when they talked about the space-time tradeoff. Example c is the most efficient, both in spatial placement and temporal placement**

Spatial-Temporal Placement Algorithm

- **Temporal Binning-based Placement**
  - **Binning algorithm:** VMs are grouped together based on their runtimes into distinct bins for placement
    - **Duration-based binning:**
      1. The runtimes of VMs within each bin are within time **T**
      2. May result in highly uniform bins, if a large number of jobs have similar runtimes.
    - **Cardinality-based binning:**
      Partitions VMs into distinct bins ordered by their runtimes, it places ***k*** VMs within each bin
  - **Intra-bin placement algorithm:** how to place each bin on the available servers
    - Using recipe-based algorithm for this paper
    - Using other algorithm(random, first-fit…) for baseline placement

Spatial-Temporal Placement Algorithm

- **Incremental Time Balancing:**
  - Provision more VMs for longer running jobs to reduce their runtimes, having their VMs finish closer in time to other co-placed VMs belonging to shorter runtime jobs. Thus, potentially decrease CMU.
  - Improve the performance of the Mapreduce



- **Incremental Time Balancing:**
  - **Incremental time balancing(ITB) algorithm**: to determine the optimal virtual cluster size for each job that minimizes the energy usage
  - This hill-climbing algorithm starts with an initial placement configuration obtained from one of the placement algorithm described above.
  - At each step select the longest runtime job of each server as a candidate for time balancing. By provisioning additional VMs for this job at a time, we can find the possible energy-efficient placement

Evaluation

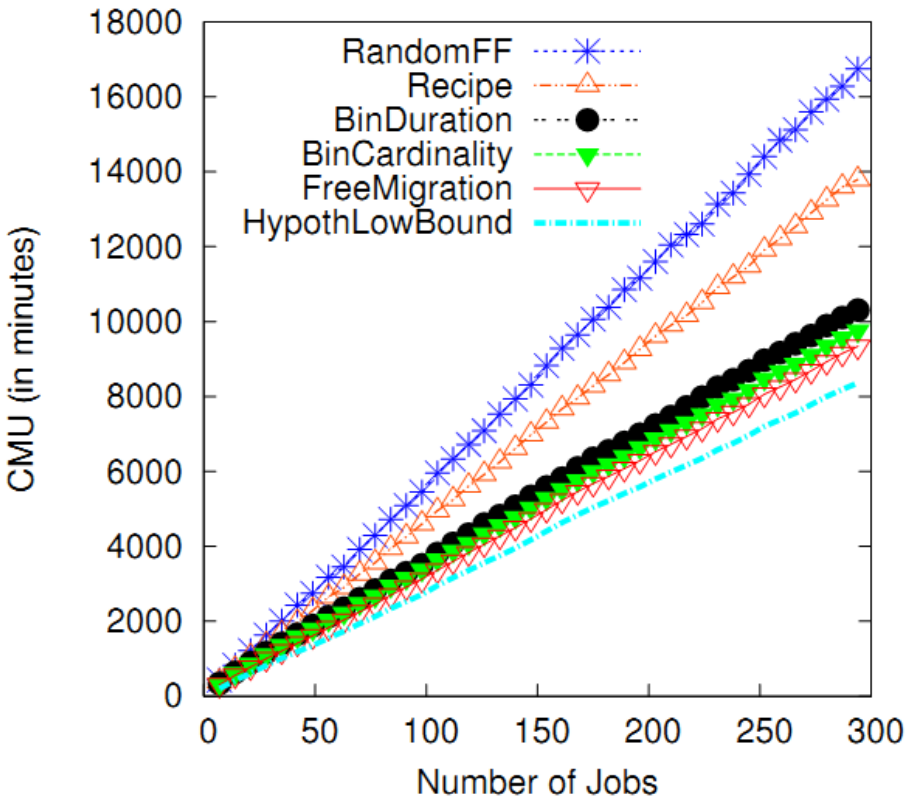
- **A simulation framework simulates a large cloud datacenter**
- **50 Mapreduce jobs,**
- **The minimum number of VMs required is uniform on [1,10]**
- **A round-robin assignment is used to assign VM types to successive jobs**
- **Estimated completion times for each job is taken from uniform distribution on [10,100] minutes**
- **3 types of resources, CPU, memory, storage**
- **7 VM types with pre-set resource configurations**

Evaluation

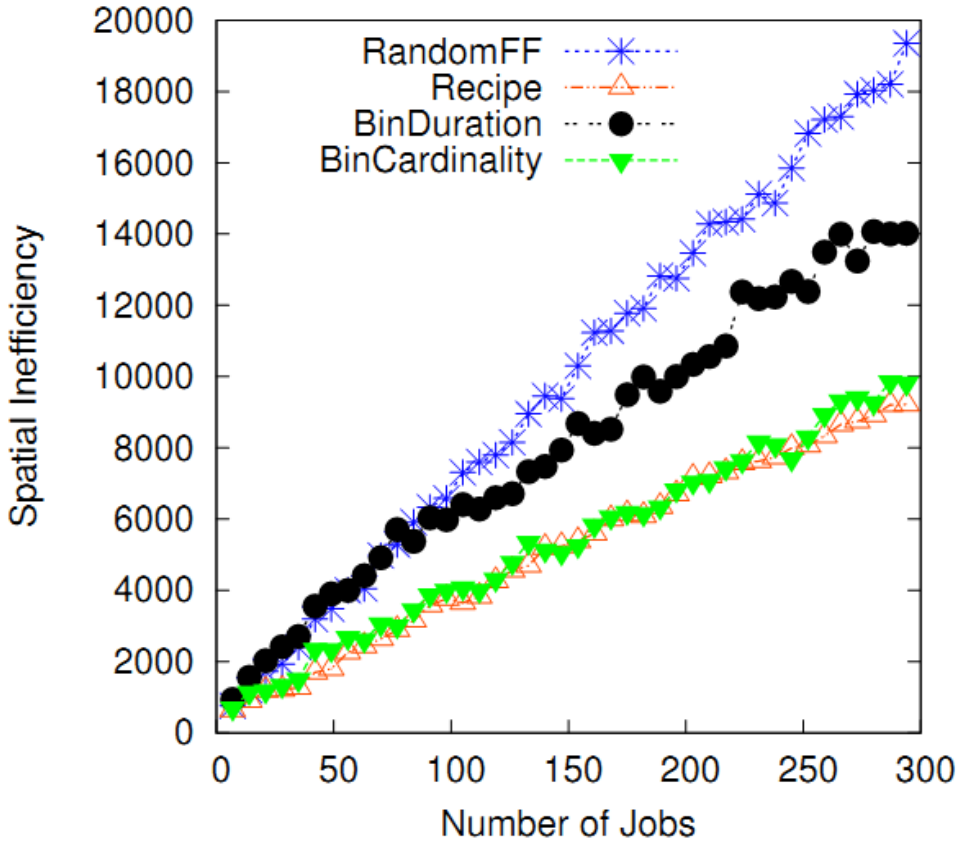
- **Spatial Alorithms:**
  - **RandomFF:** VMs randomly shuffled before a first-fit placement is performed
  - **Recipe**: Recipe-based spatial packing
- **Spatio-Temporal Algorithm**
  - **BinDuration:** Duration-based binning
  - **BinCardinality:** Cardinality-based binning
  - **+ITB**: incremental time balancing
- **Two hypothetical Baseline Algorithms**
  - **FreeMigration:** VMs can be replaced every time a job finishes.
  - **HypothLowBound:** A server's energy consumption is proportional to its resource utilization

Evaluation

- **Spatio-Temporal Tradeoffs**



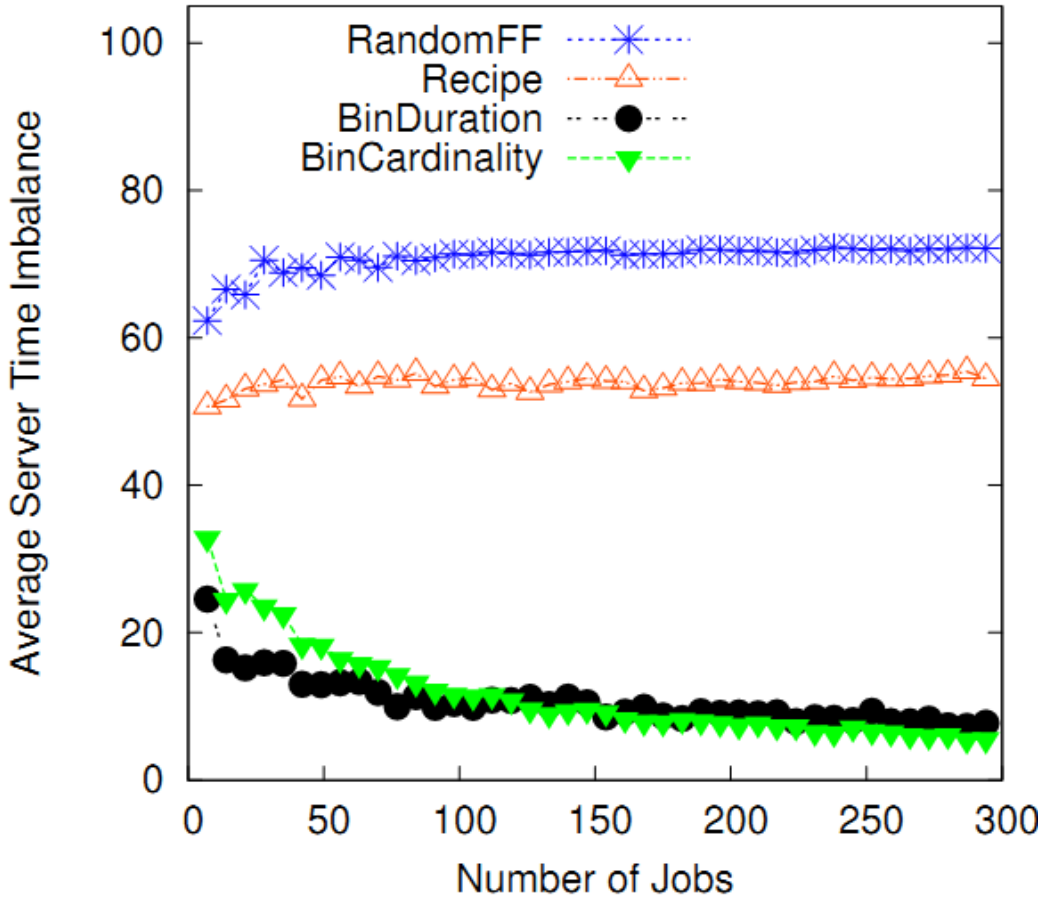
(a) Energy Usage (CMU)



(b) Spatial Inefficiency (SI)

Evaluation

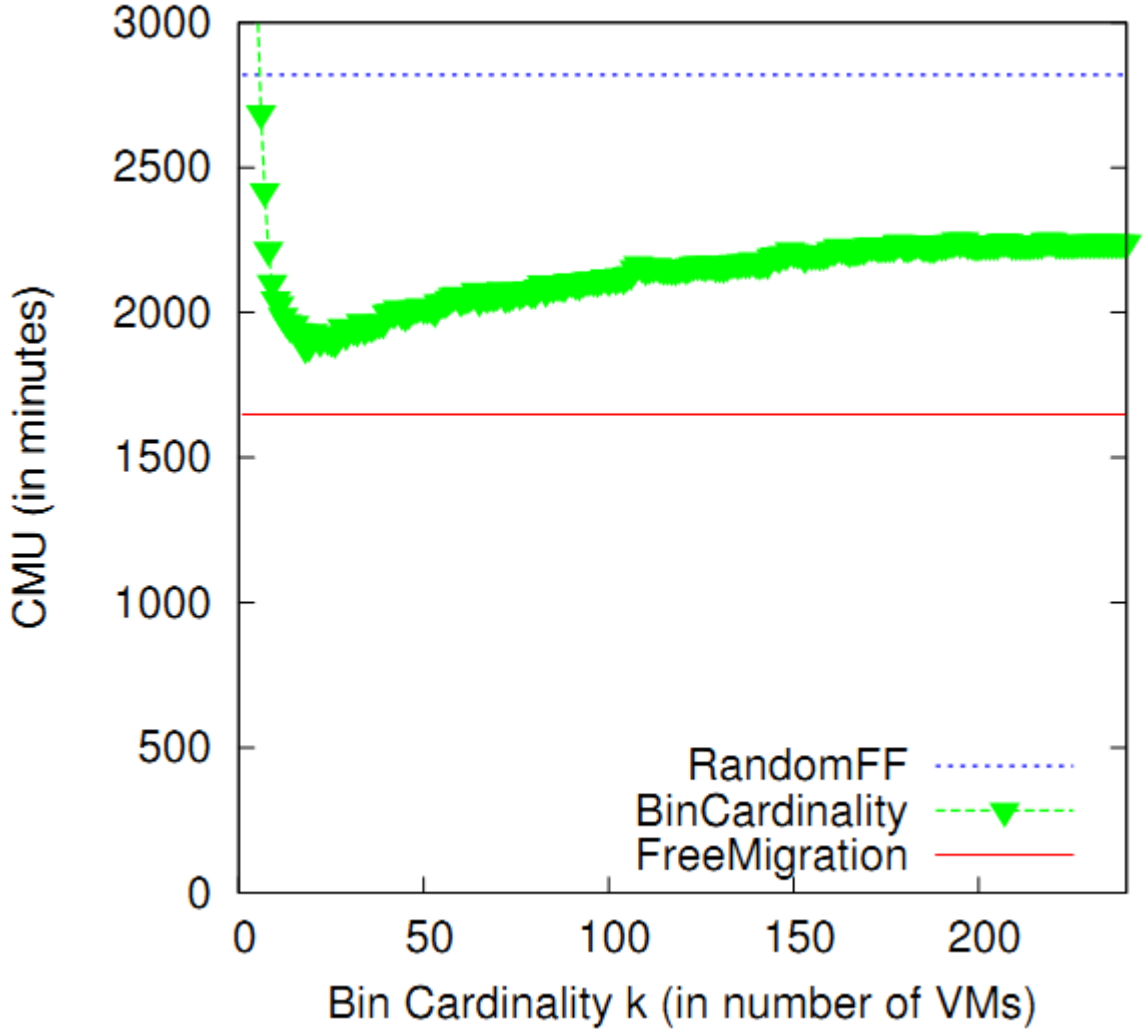
- **Spatio-Temporal Tradeoffs**



(c) Temporal Imbalance (TI)

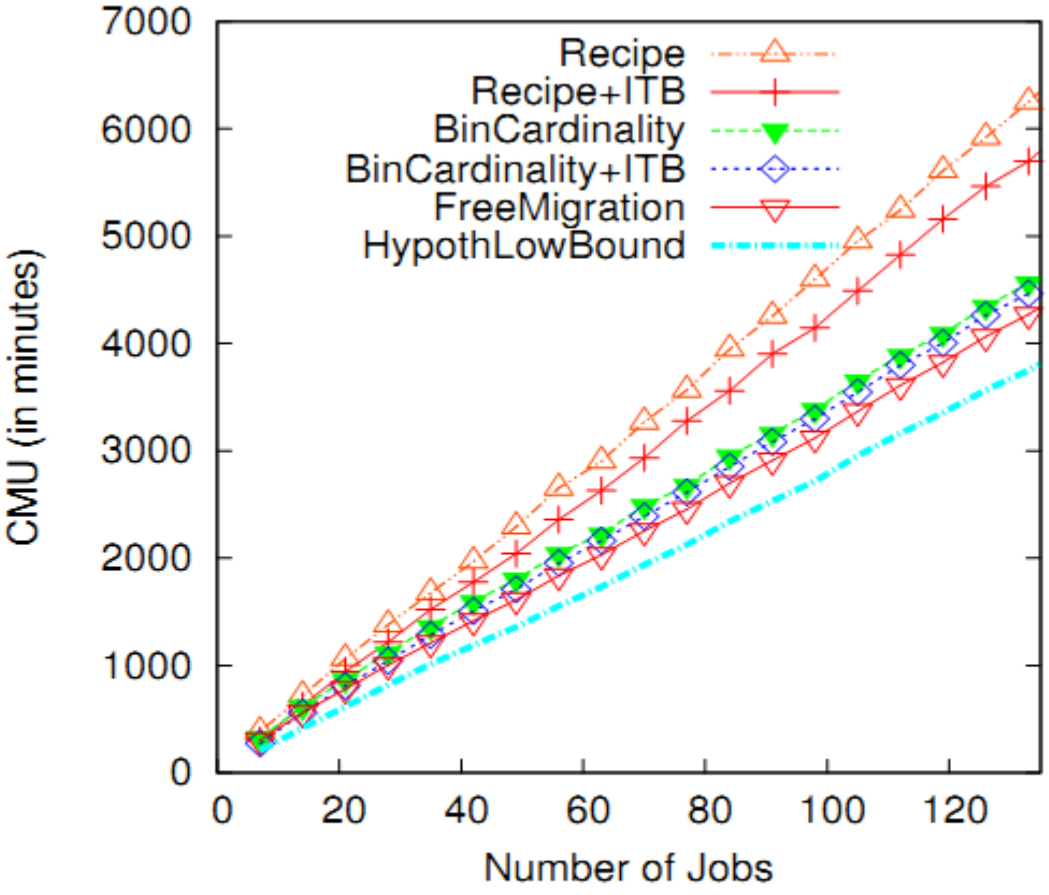
Evaluation

- **Impact of Bin Size**



Evaluation

- **Incremental Time balancing**



(a) Incremental Time Balancing provides an improvement over the base algorithm.

Conclusions and Future Work

- **Our algorithms exploiting the spatio-temporal tradeoffs(BinCardinality, BinDuration) perform 20-35% better than spatial algorithm (RandomFF, Recipe)**
- **ITB further improves our algorithm by up to 15% while reducing Mapreduce job runtimes between 5-35%**
- **Our techniques could also be extended to other environments**
- **Considering DVFS**

Questions and comments