Exploiting Spatio-Temporal Tradeoffs for Energy Efficient MapReduce in the Cloud

Talk by
Alexander Bunte

Outline

- MapReduce
  - Cloud, Energy efficiency
- Resource Wastage Metrics
- Efficiency Trade-offs
  - Spatial vs. Temporal
- Algorithms
  - Binning, Placement, ITB
- Evaluation
- Discussion
- Conclusion

MapReduce

- Programming model for processing huge data sets
- Distributed over large cluster of machines
- Highly scalable

MapReduce

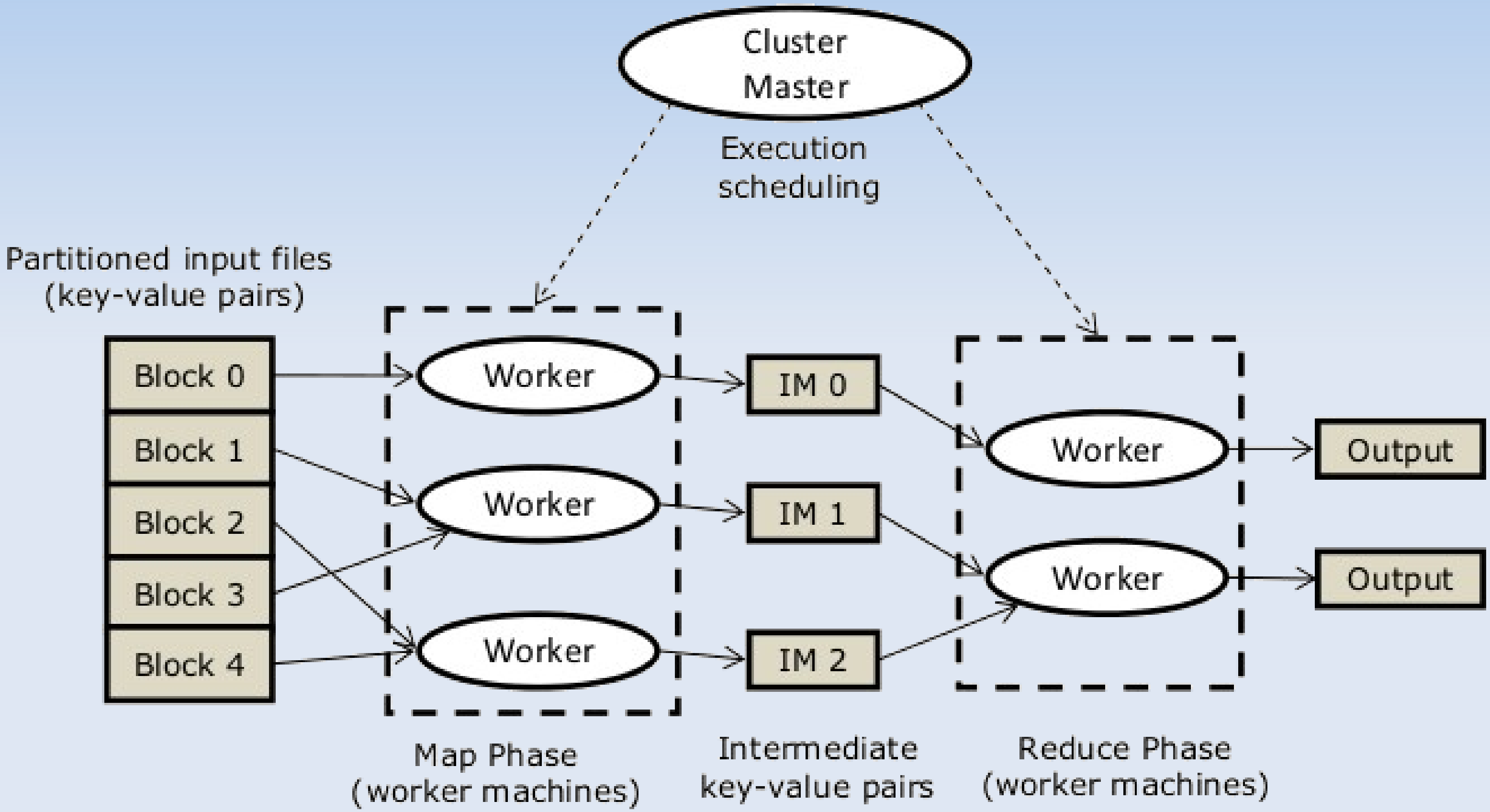
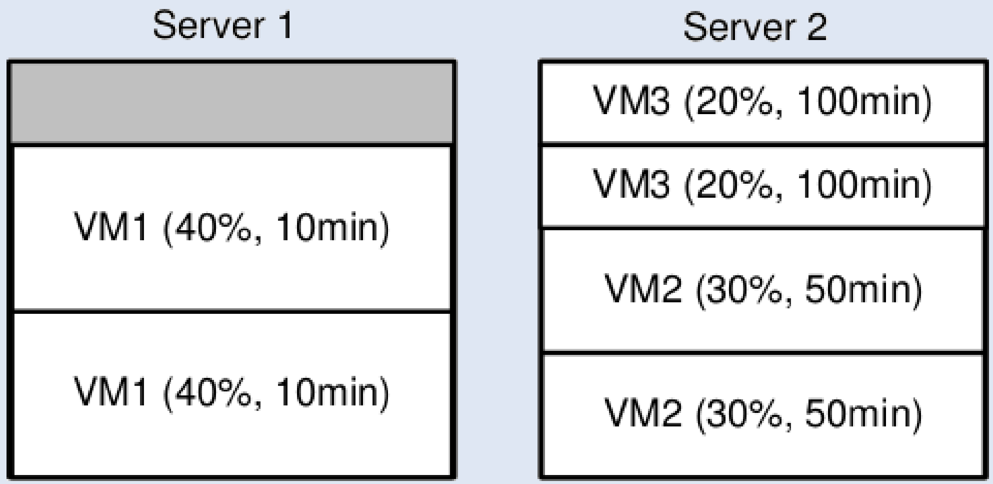


Fig. 1. Illustration of a MapReduce job execution.

MR in the cloud

- Workers are virtual machines
- Several VMs run on one physical node (PM)
- Jobs are executed in their own virtual cluster
- Different types of VMs are available



Energy Efficiency of MR

- Running PMs consume Energy independent of utilization
- Energy consumption may only be effectively decreased by suspending a node
- Minimizing cumulative machine uptime (CMU) should save energy
- For simplicity assume jobs and their runtimes are known in advance

Ressource Wastage Metrics

- $VM_1, \ldots, VM_n : VMs\ hosted\ on\ server\ i$
- $C_i : Resource\ capacity\ of\ server\ i$
- $R_j : Resources\ required\ by\ VM_j$
- $t_j : Runtime\ of\ VM_j$

Resource Wastage Metrics

- Machine Uptime: $$MU_i = \max t_j$$
- Cumulative MU: $$CMU = \sum_{k=1}^{N} MU_k$$
- Spatial Inefficiency: $$SI_i = C_i - \sum_{j=1}^{n} R_j$$
- Time Imbalance: $$TI_i = \max t_j - \min t_j$$

Resource Wastage Metrics

- Spatial Waste: $SW_i = SI_i * MU_i$
- Dead VM Waste: $DW_i = \sum_{j=1}^{N} (MU_i - t_j) * R_j$
- Total Waste: $TW_i = SW_i + DW_i$

Resource Wastage Metrics

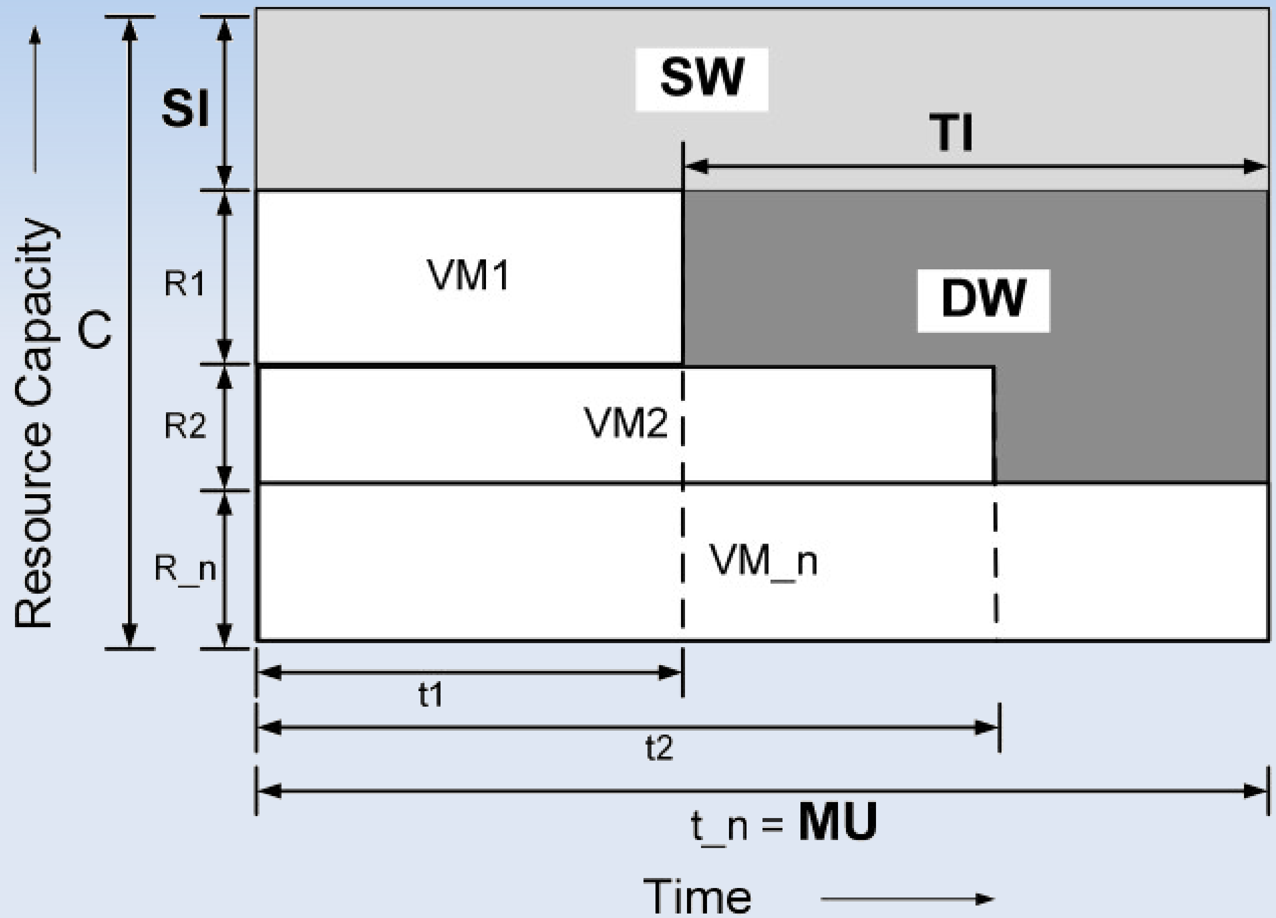


Fig. 6. Resource wastage metrics for a server.

Resource Wastage Metrics

$$MU_i * C_i = UW_i + TW_i$$

UW: Useful Work

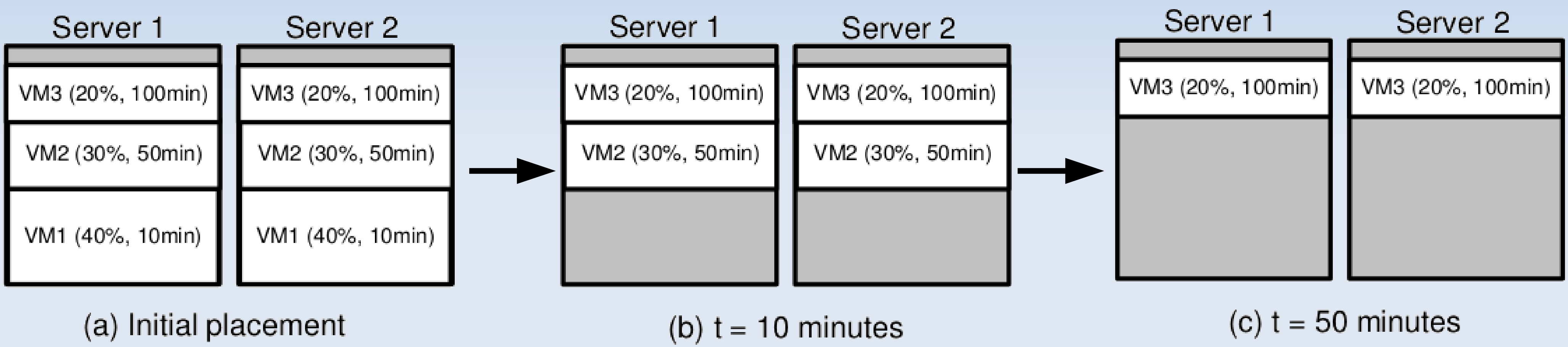
$$\Leftrightarrow \sum_{i=1}^{N} MU_i * C_i = \sum_{i=1}^{N} \left(UW_i + TW_i \right)$$

$$\Leftrightarrow CMU = \frac{\sum_{i=1}^{N} \left(UW_i + TW_i \right)}{C}$$

- This shows that we need to minimize Total Waste (by minimizing Spatial and Temporal Inefficiency) to minimize CMU

Spatial Efficiency

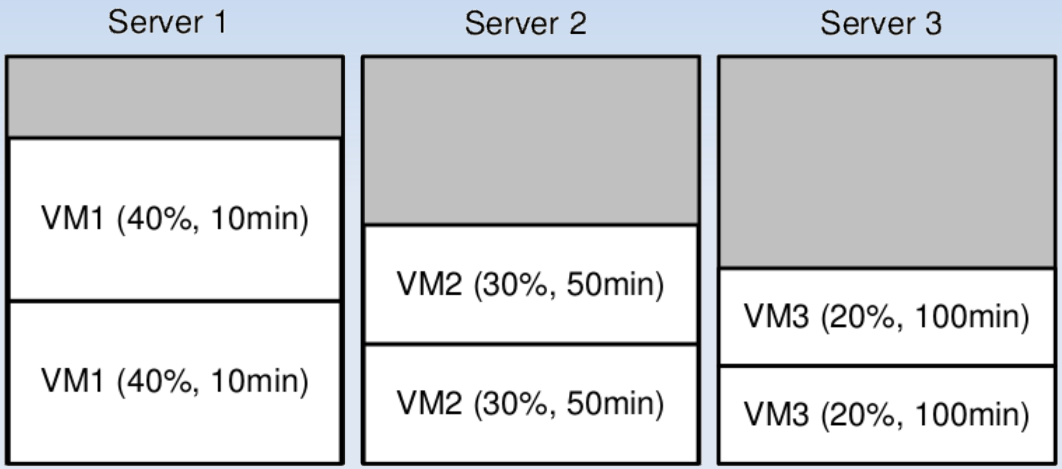
- Place VMs on PMs such that utilization is maximized



- CMU = 100 + 100 = 200
- Spatially-efficient placement results in wasted resources as jobs finish at different times

Temporal Efficiency

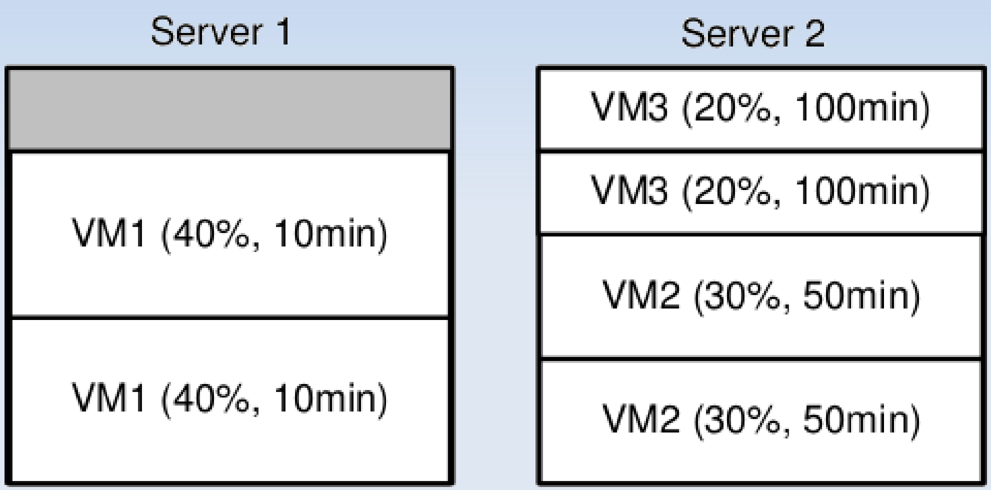
- Place on each PM VMs with the same runtime, so VM can be suspended when jobs finish



- CMU = 10 + 50 + 100 = 160
- Time-balanced placement can lead to lower CMU than spatial-efficient

Spatio-Temporal Efficiency

- Trade-off between spatial and temporal efficiency



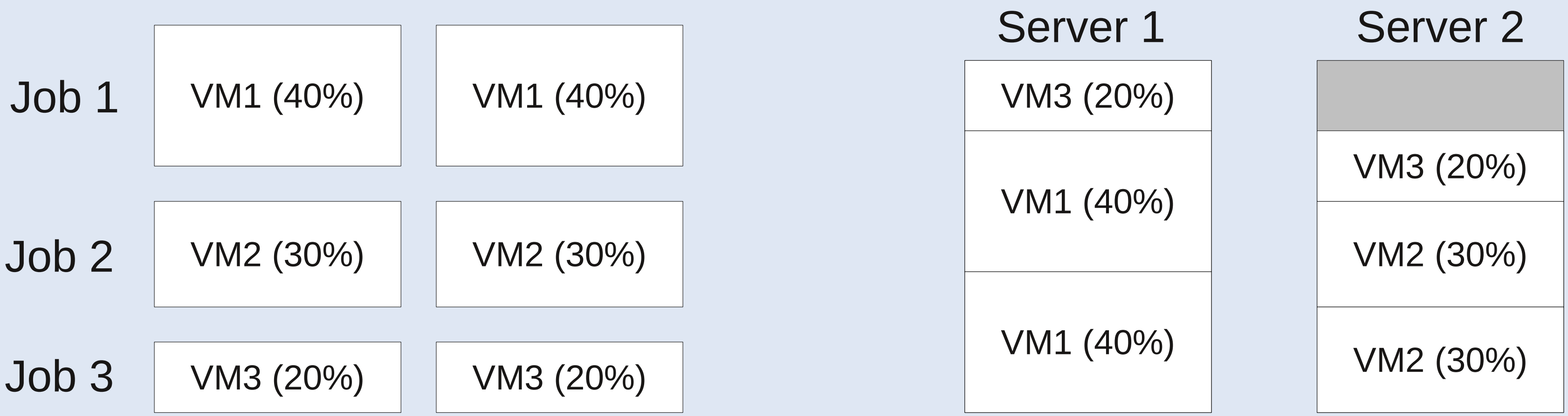
- CMU = 10 + 100 = 110
- An optimal placement needs to be both spatially-efficient as well as time-balanced

Temporal Binning-based Placement

- Binning algorithm
  - Partitions VM's into distinct bins based on runtimes
  - Regulates time-balancing
- Intra-bin placement algorithm
  - Places VM's per bin on the cluster
  - Regulates spatial efficiency
  - Examples: Receipe, First-Fit

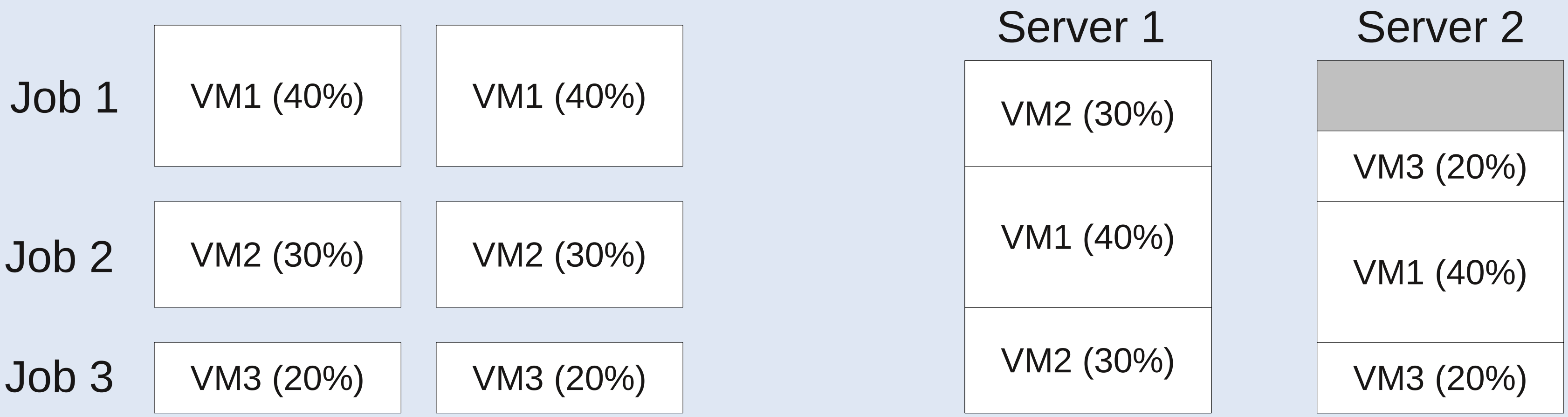
Spatial Placement

- Spatial-First-Fit
  - Place VMs ordered by job in first-fit fashion



Spatial Placement

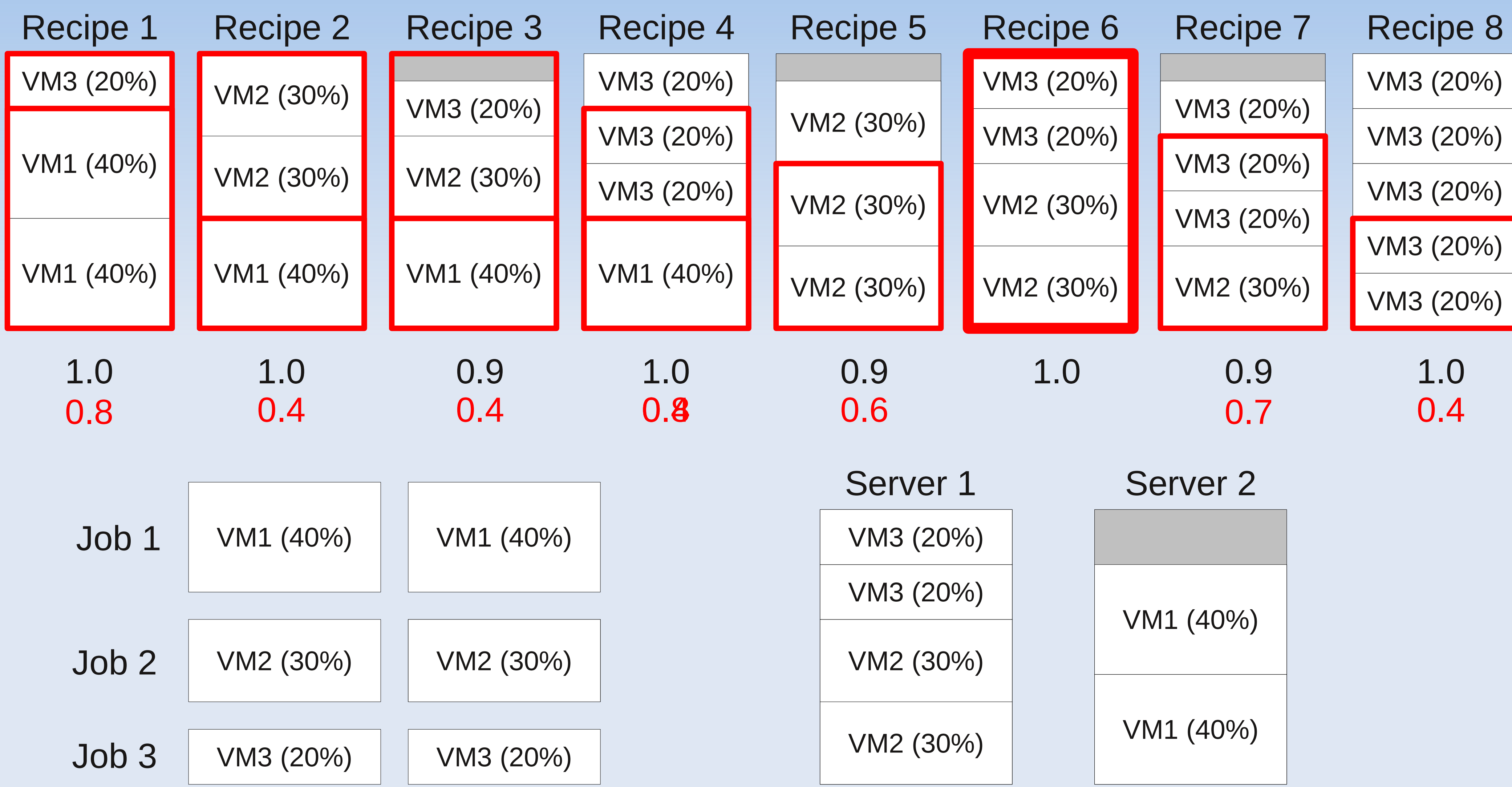
- Random-First-Fit
  - Randomly choose VMs from all jobs and place them in first-fit fashion



Spatial Placement

- Recipe Placement
  - Precompute all possible placements of VMs on a PM
  - Rank recipes by utilization of PM
  - In each step of placement choose highest ranked recipe that matches subset of remaining VMs

Recipe Placement



Binning Algorithms

- Duration-based
  - Runtime range is divided into uniform intervals
  - Each bin is assigned to a distinct interval
  - Bins are potentially skewed

T = 30	1 - 30	31 - 60	61 - 90
	VM1 (10 min)		VM3 (70 min)
	VM1 (10 min)		VM3 (70 min)
	VM2 (30 min)		
	VM2 (30 min)		

Binning Algorithms

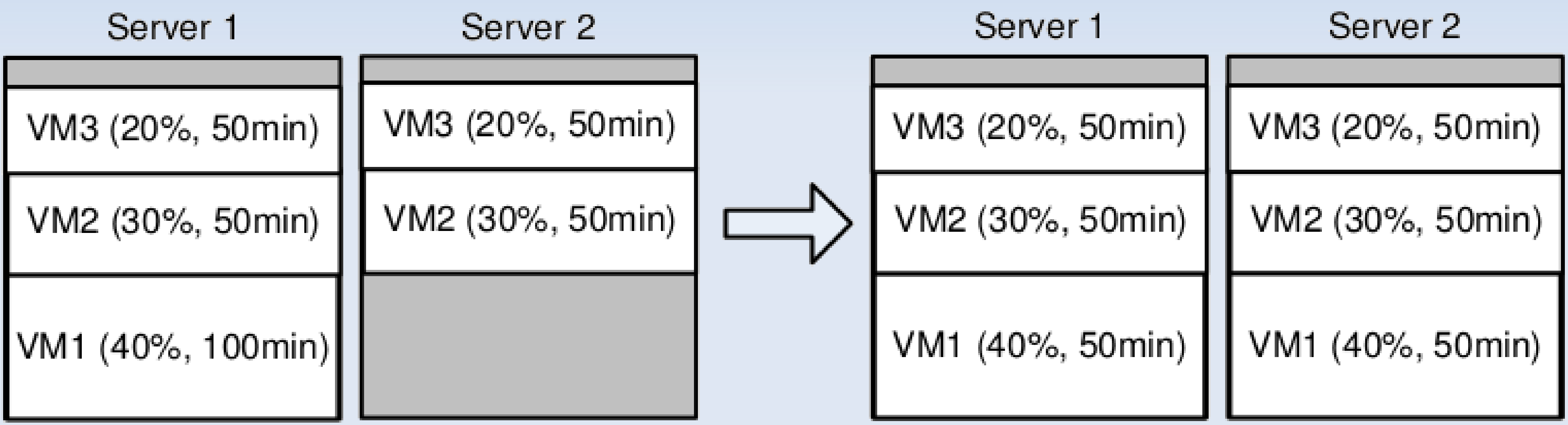
- Cardinality-based
  - Partitions VM's into fixed-size bins
  - Guaranties uniform partitioning

k = 3	Bin 1	Bin 2
	VM1 (10 min)	VM2 (30 min)
	VM1 (10 min)	VM3 (70 min)
	VM2 (30 min)	VM3 (70 min)

Incremental Time Balancing

- Some jobs may have a totally different runtime than others
- Increase size of a virtual cluster to decrease its runtime
- This way cluster utilization may be even more increased
- Side-effect: Better performance

Incremental Time Balancing



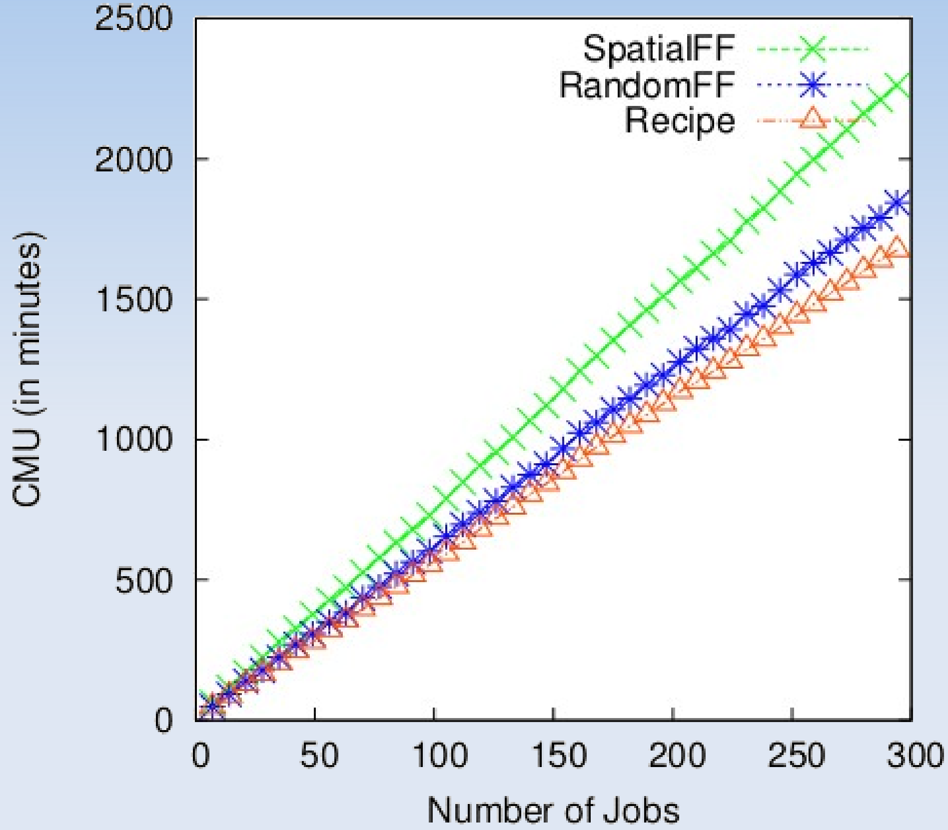
Evaluation

- Simulation framework generates jobs with configurable parameters
  - Number of jobs
  - Deadline and number of VMs range
  - Assignment of VM types for the jobs
- Simulates placement and execution on a cluster

Baseline Algorithms

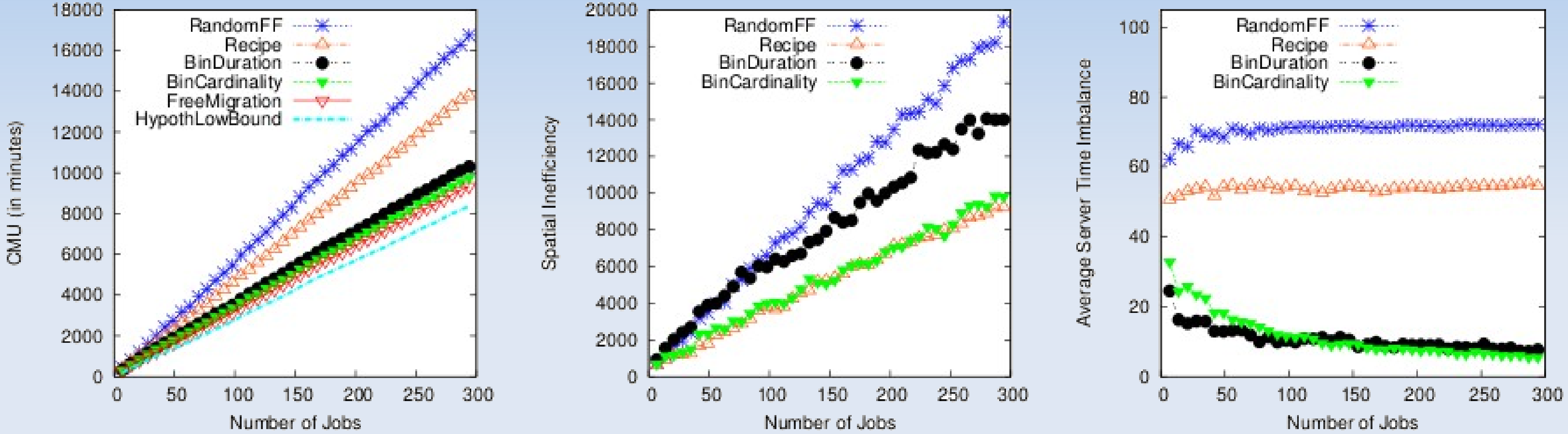
- FreeMigration
  - Migrates VMs while job execution for free
- HypothLowBound
  - Assumes that energy consumption of a node scales perfectly linear with its utilization

Recipe Placement



- Recipe placement achieves better efficiency than first-fit algorithms

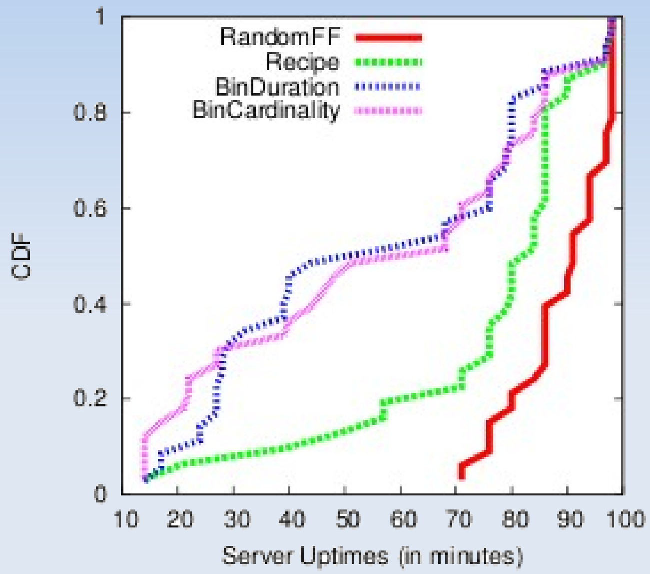
Benchmark of the different Algorithms



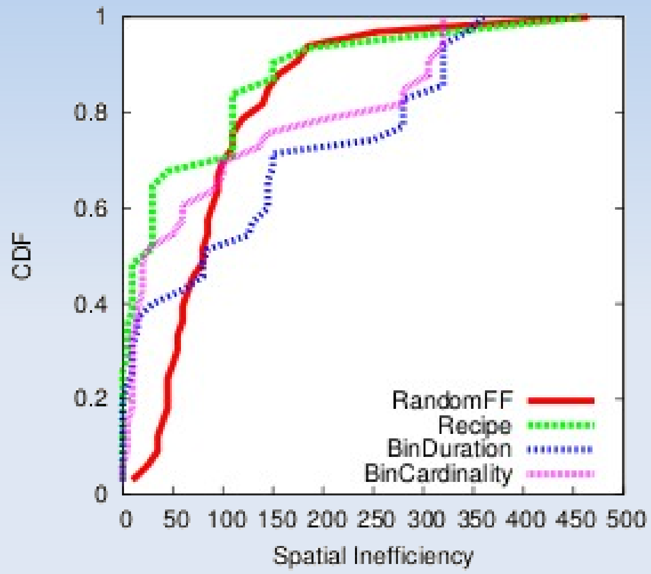
(a) Energy Usage (CMU) (b) Spatial Inefficiency (SI) (c) Temporal Imbalance (TI)

- Spatio-temporal algorithms perform significantly better than spatial-only ones

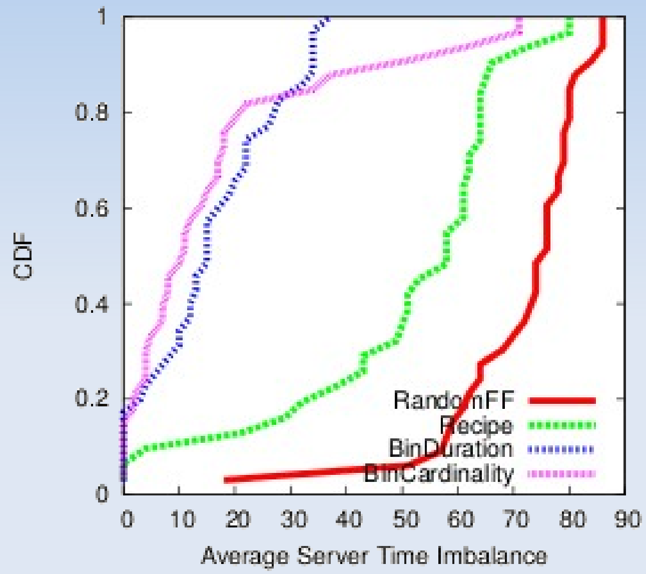
CDFs



(a) Server Uptimes



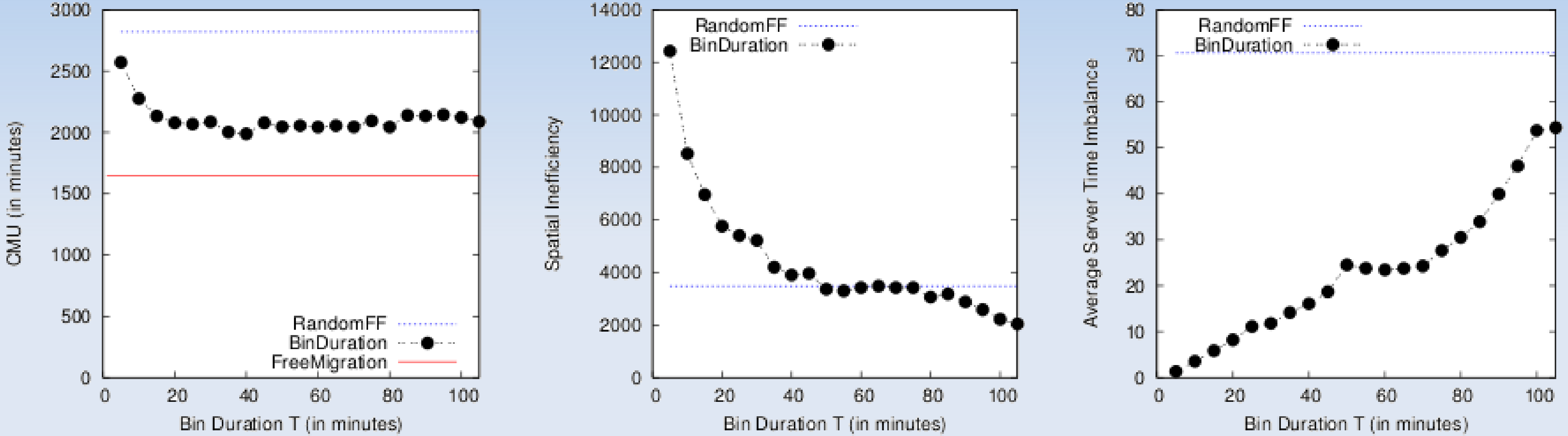
(b) SI



(c) TI

- With RandomFF and Recipe most PMs have a similar (high) uptime

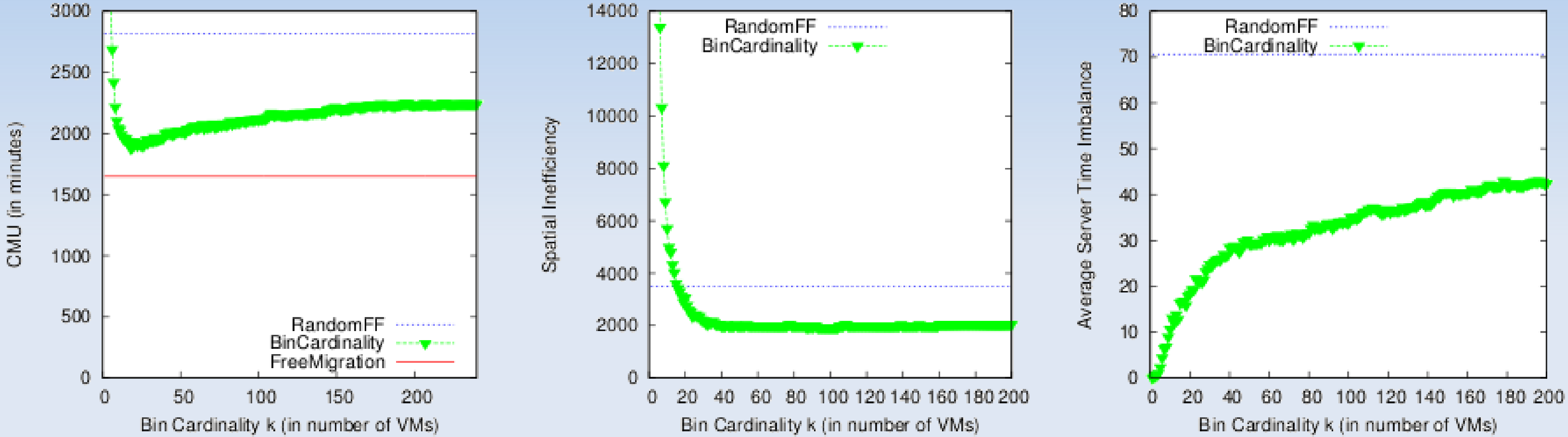
Choice of interval size



(a) Energy Usage (CMU)　(b) Spatial Inefficiency (SI)　(c) Temporal Imbalance (TI)

- Best range T here: 40 minutes

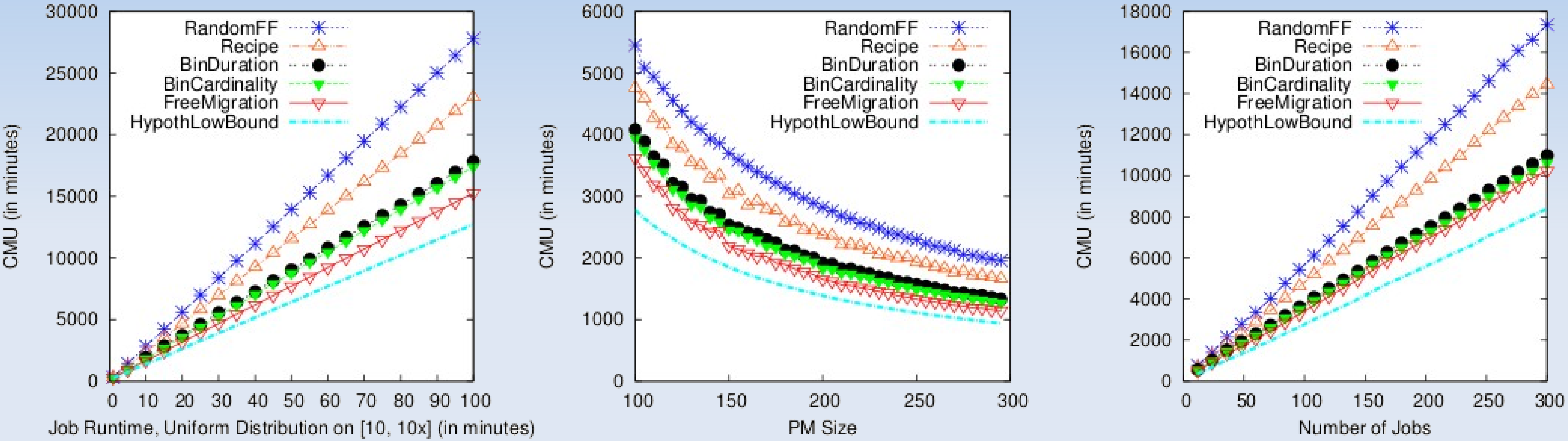
Choice of cardinality



(a) Energy Usage (CMU) (b) Spatial Inefficiency (SI) (c) Temporal Imbalance (TI)

- Optimal cardinality k: 18 VMs

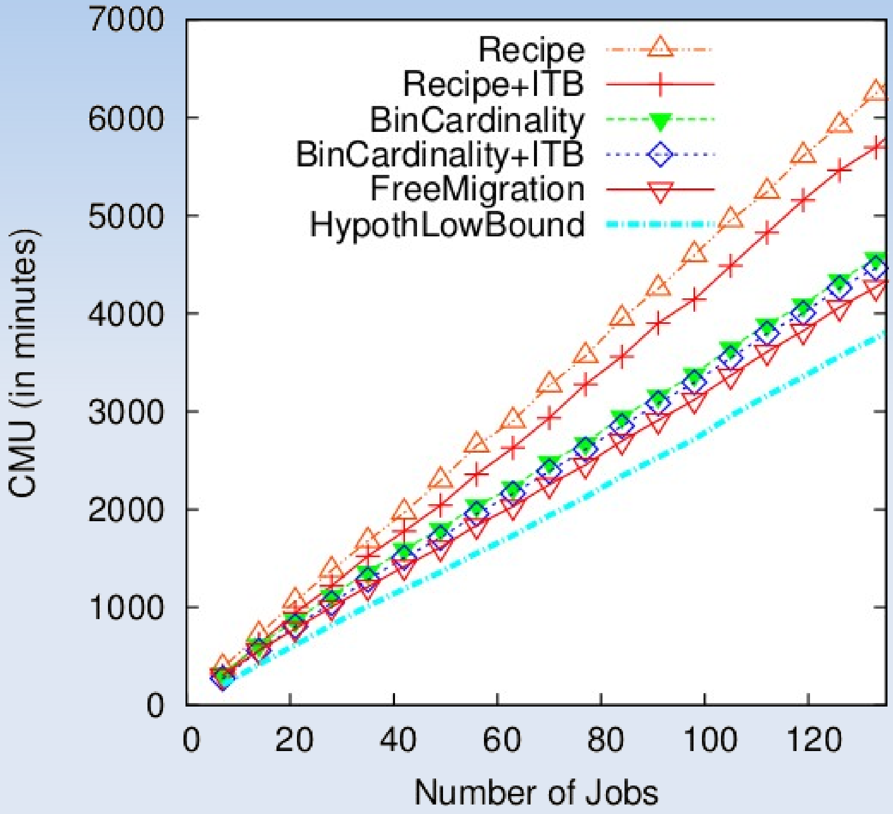
Varying Parameters



(a) Job Runtime Spread (b) PM Size (c) VM Type Assignment

- Algorithms are robust to changes in workload and environment

ITB results



- Incremental time balancing indeed decreases CMU even more

Discussion

- Continuous optimization
- Running MR jobs without virtualization
- Heterogeneous physical resources

Conclusions

- Spatio-temporal algorithms improve utilization of MR on a cluster by 20-35%
- Incremental time balancing adds further gains of up to 15%