Elasecutor: Elastic Executor Scheduling in Data Analytics Systems

Libin Liu, Hong Xu

City University of Hong Kong

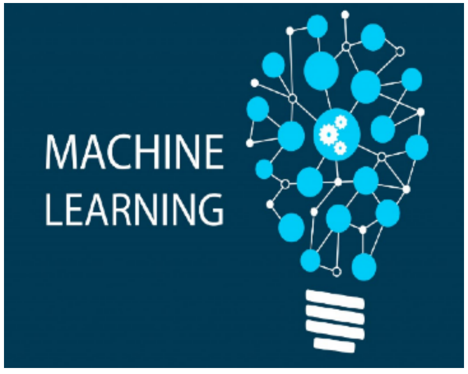
ACM Symposium on Cloud Computing 2018

Data Analytics Systems

- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG

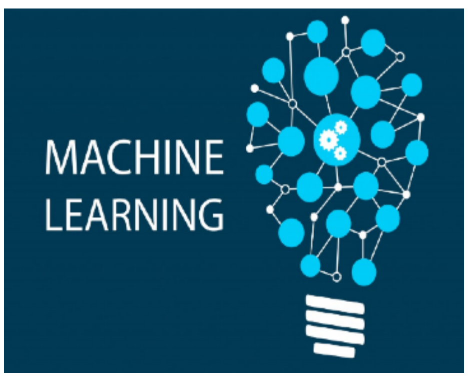
Data Analytics Systems

- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG



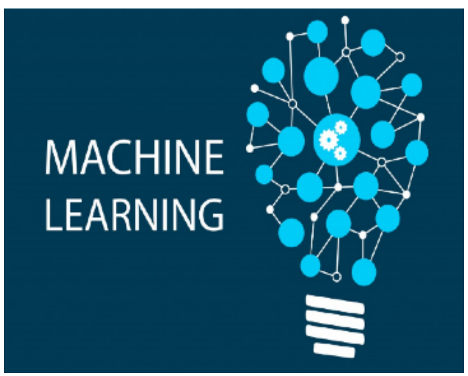
Data Analytics Systems

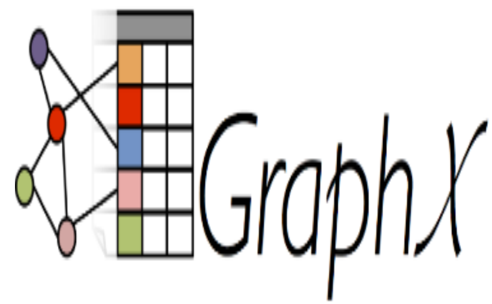
- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG



Data Analytics Systems

- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG



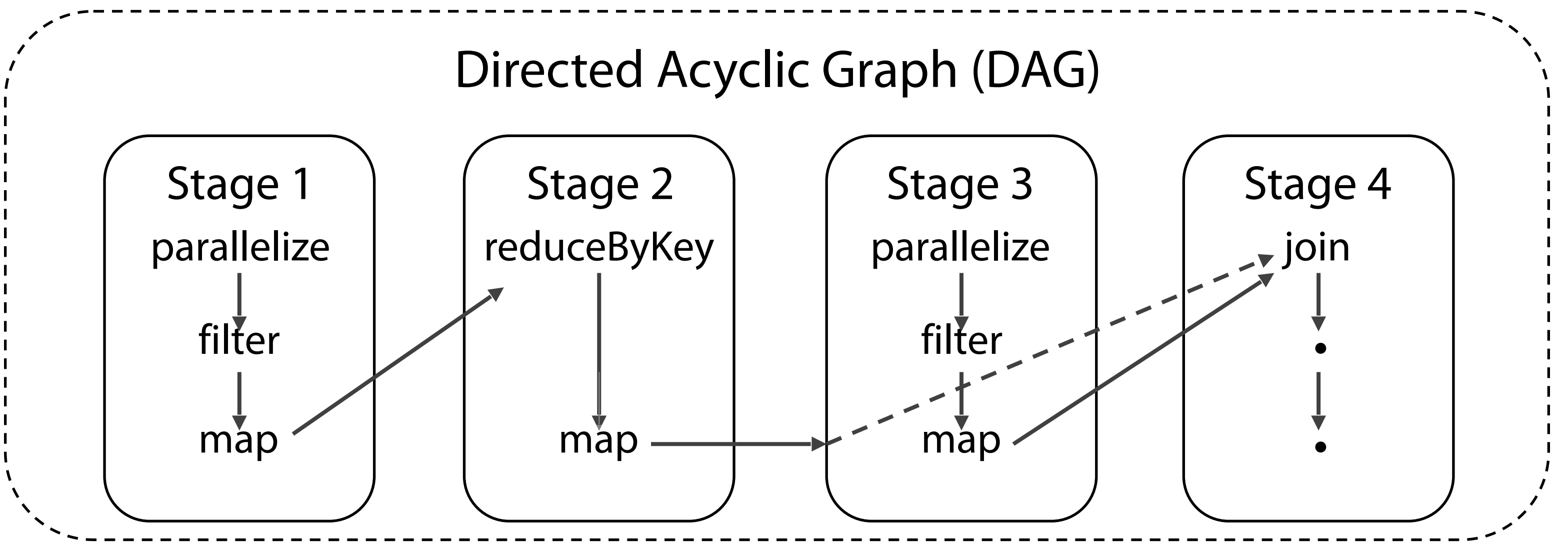


Data Analytics Systems

- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG

Data Analytics Systems

- Various workloads running in data analytics systems concurrently
- The workflow of an analytics application can be expressed as a DAG



Resource Scheduling

- Resource schedulers for various objectives, e.g., fairness, cluster utilization, application completion time, etc.

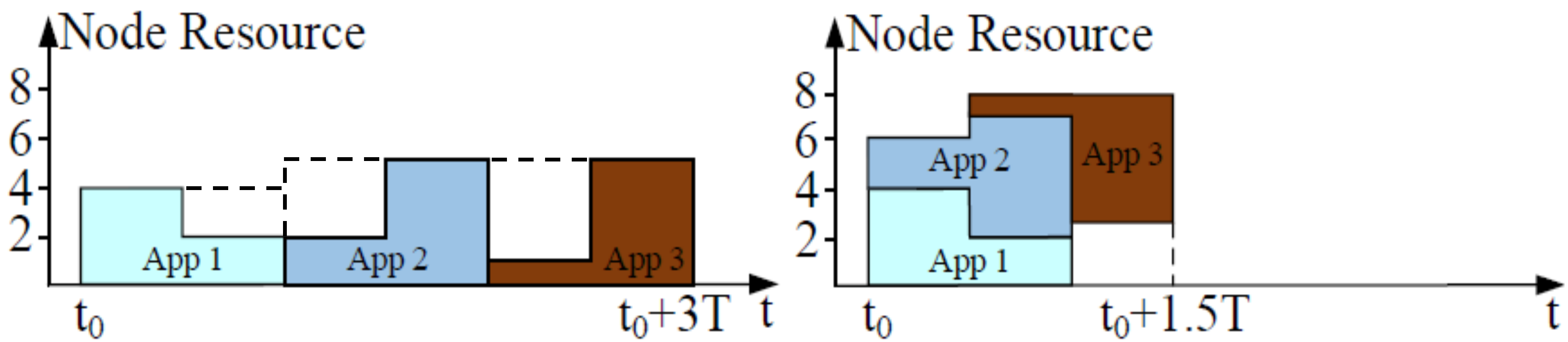
Resource Scheduling

- Resource schedulers for various objectives, e.g., fairness, cluster utilization, application completion time, etc.

Efficient resource scheduling is an important and practical issue in data analytics systems

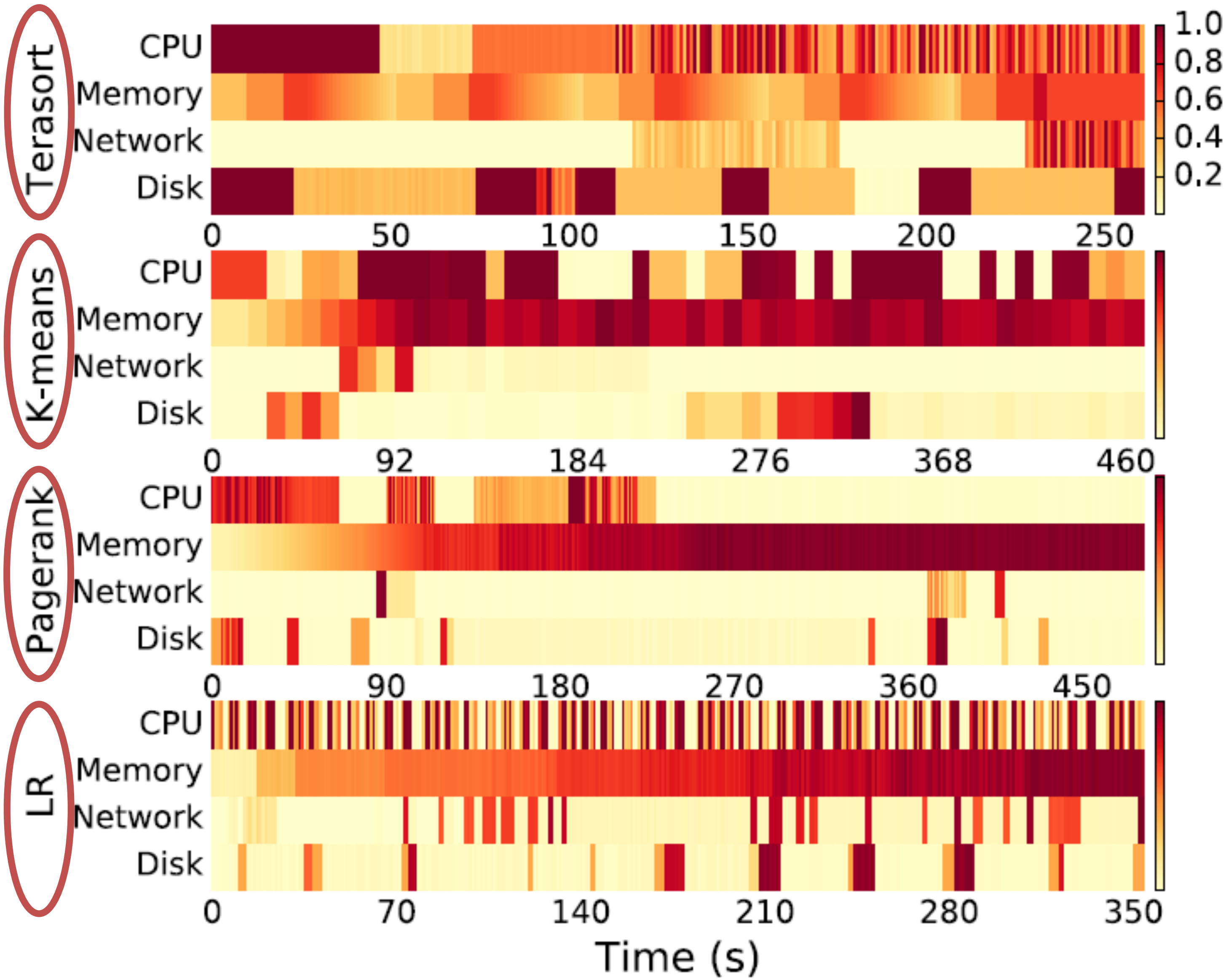
Current Solutions

- Static allocation according to peak demands
- "Task-based" resource schedulers adopted in "executor-based" systems
- Assign executors to machines randomly

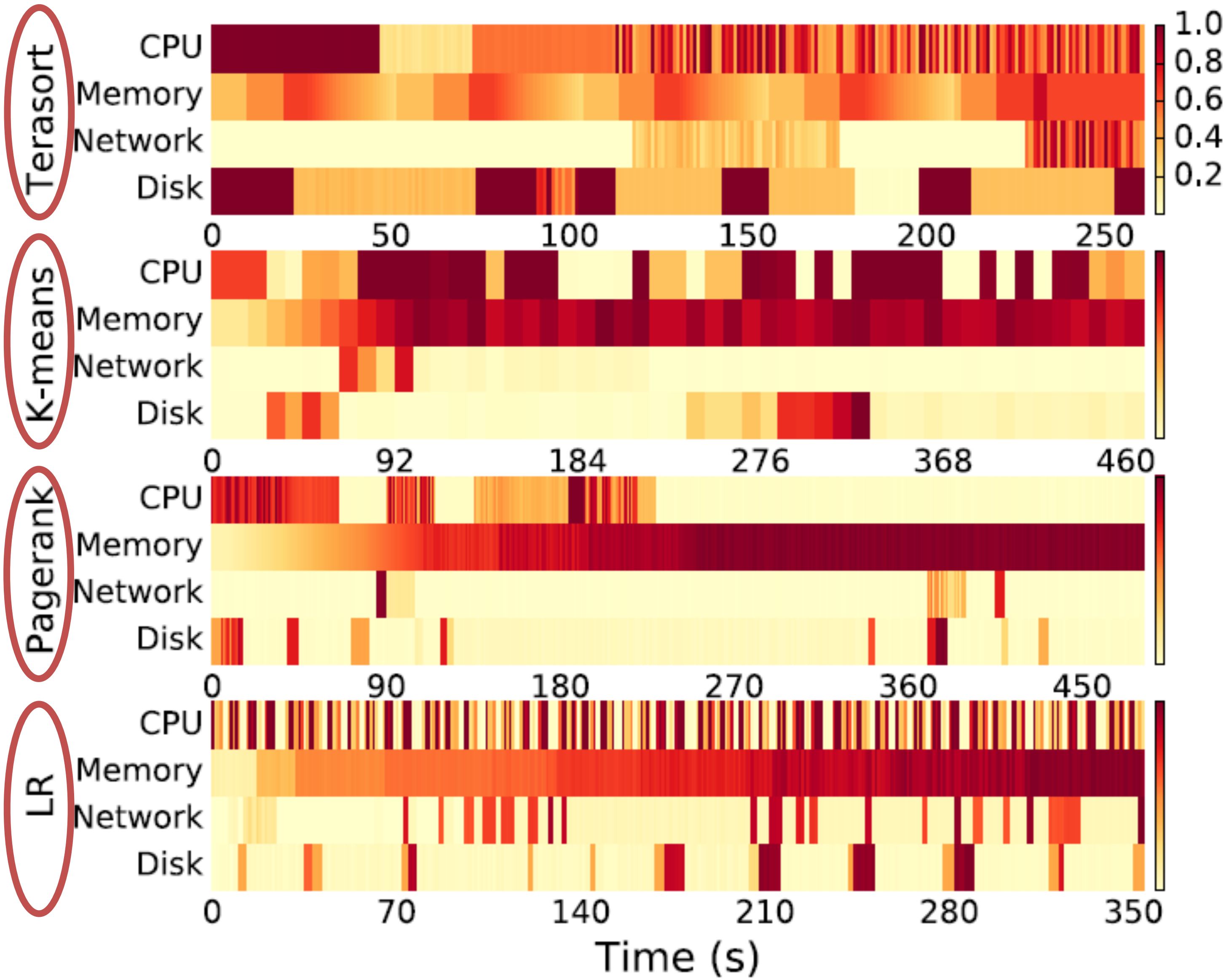


(a) Static allocation **(b)** Elastic allocation

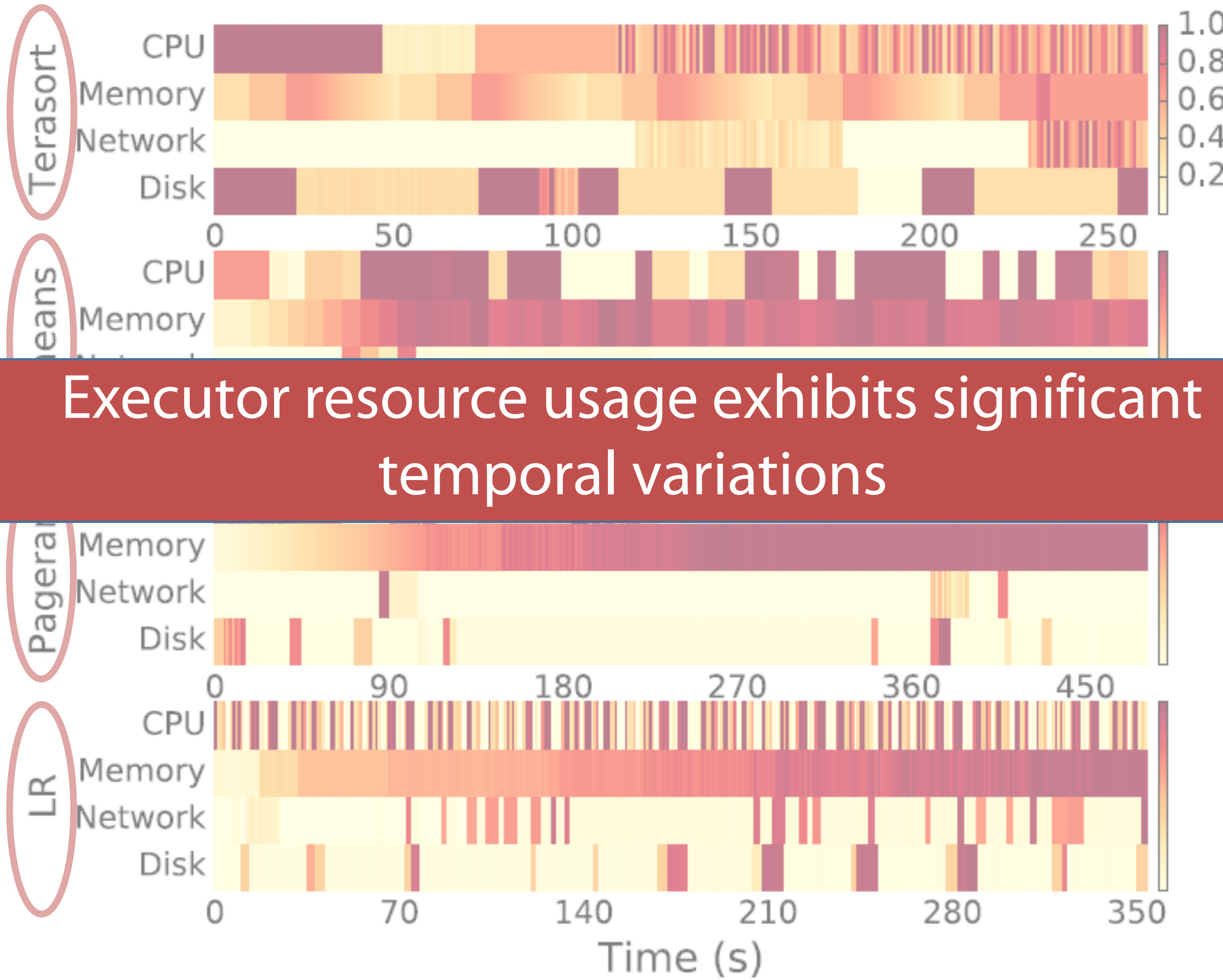
Need for an Elastic Scheduler



Need for an Elastic Scheduler



Need for an Elastic Scheduler



Executor resource usage exhibits significant temporal variations

Need for an Elastic Scheduler

Resource	CPU	Memory	Network	Disk
Terasort				
Peak/Avg.	1.8	1.7	6.2	1.5
Peak/Trough	60	3.3	237	6.1
K-means				
Peak/Avg.	1.7	1.2	11.5	5.6
Peak/Trough	75	6	53	100
Pagerank				
Peak/Avg.	3.9	1.3	20.2	9.1
Peak/Trough	50	11.5	119	50
Logistic Regression				
Peak/Avg.	2.1	1.4	5.5	6.1
Peak/Trough	50	12	409.6	42.5

Need for an Elastic Scheduler

Resource	CPU	Memory	Network	Disk
Terasort				
Peak/Avg.	1.8	1.7	6.2	1.5
Peak/Trough	60	3.3	237	6.1
K-means				
Peak/Avg.	1.7	1.2	11.5	5.6
Peak/Trough	75	6	53	100
Pagerank				
Peak/Avg.	3.9	1.3	20.2	9.1
Peak/Trough	50	11.5	119	50
Logistic Regression				
Peak/Avg.	2.1	1.4	5.5	6.1
Peak/Trough	50	12	409.6	42.5

Need for an Elastic Scheduler

Resource	CPU	Memory	Network	Disk
Terasort				
Peak/Avg.	1.8	1.7	6.2	1.5
Peak/Trough	60	3.3	237	6.1
K-means				
Peak/Avg.	1.7	1.2	11.5	5.6
Peak/Trough	75	6	53	100
Pagerank				
Peak/Avg.	3.9	1.3	20.2	9.1
Peak/Trough	50	11.5	119	50
Logistic Regression				
Peak/Avg.	2.1	1.4	5.5	6.1
Peak/Trough	50	12	409.6	42.5

Need for an Elastic Scheduler

Resource	CPU	Memory	Network	Disk
Terasort				
Peak/Avg.	1.8	1.7	6.2	1.5
Peak/Trough	60	3.3	237	6.1
K-means				
Pagerank				
Peak/Avg.	3.9	1.3	20.2	9.1
Peak/Trough	50	11.5	119	50
Logistic Regression				
Peak/Avg.	2.1	1.4	5.5	6.1
Peak/Trough	50	12	409.6	42.5

Static allocation using peak demands would cause severe resource wastage and performance issues

Our Idea

Dynamically allocate and explicitly size resources to executors over time, and strategically assign executors to machines

Our Idea

Dynamically allocate and explicitly size resources to executors over time, and strategically assign executors to machines

Our Idea

Dynamically allocate and explicitly size resources to executors over time, and strategically assign executors to machines

↓

Elasecutor, a novel executor scheduler for data analytics systems

Outline

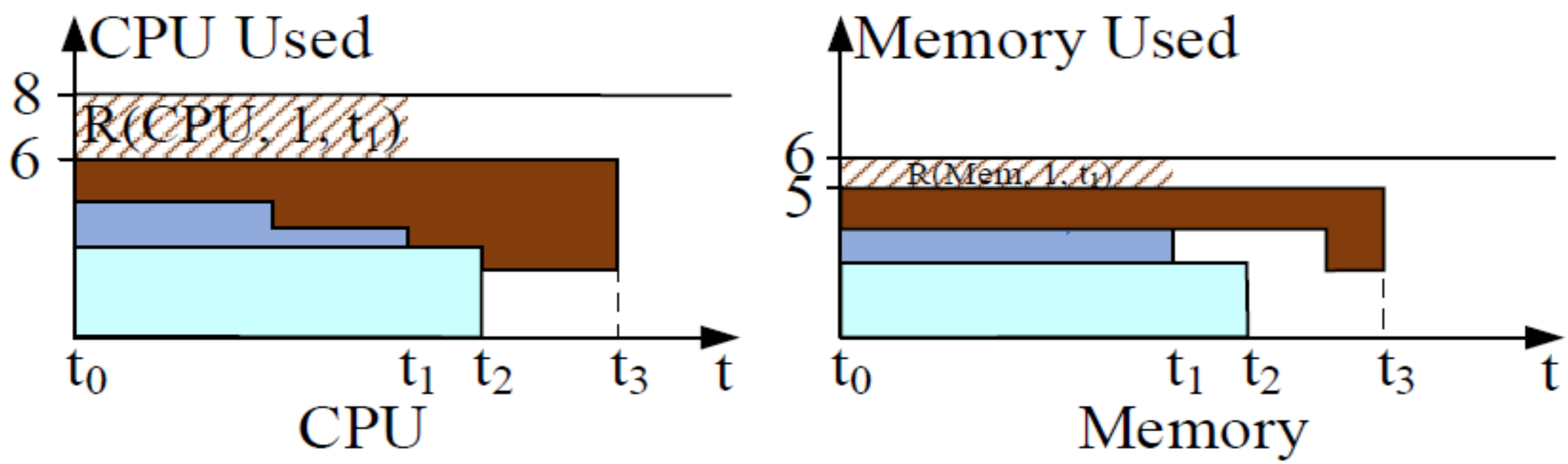
- Motivation
- Elasecutor Design
  - Elastic Executor Scheduling
  - Demand Prediction
  - Dynamic Reprovisioning
- Implementation
- Evaluation
- Conclusion

Elastic Executor Scheduling

- Challenge
  - Scheduling executors with their multi-resource demand time-series
  - Multi-dimensional packing
  - APX-hard
  - Analyzed in detail in section 3.2.1
- Objective
  - Minimizing makespan
  - i.e., avoid resource underutilization and minimize machine-level resource fragmentation

Elastic Executor Scheduling - DRR

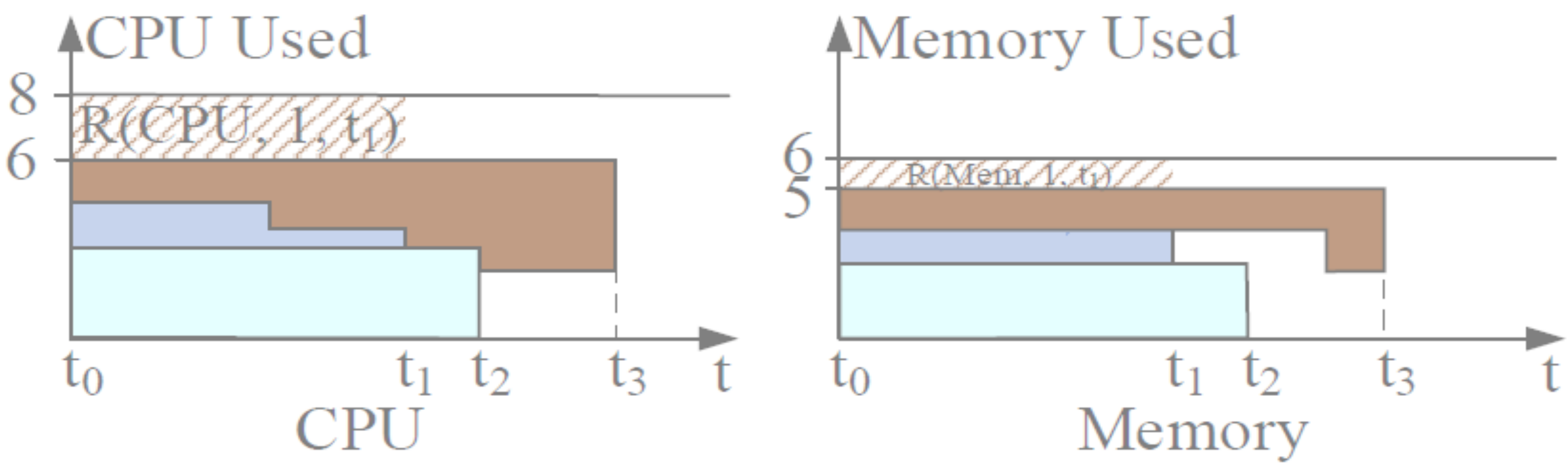
- Dominant Remaining Resource: "dominant" = "maximum"
- An example: We select as the time point to calculate DRR for machine 1. and , and its *DRR* is



Elastic Executor Scheduling - DRR

- Dominant Remaining Resource: "dominant" = "maximum"
- An example: We select as the time point to calculate DRR for machine 1. and , and its *DRR* is

DRR is defined as the maximum remaining resource along the time dimension up to time t



Why DRR

- Convert multi-dimensional metrics into scalars
- Better reflect resource utilization
  - "Maximum", not "Minimum"
- Better than alternative metric TRC
  - TRC sums up the relative remaining capacity of each resource

Design Choices	Makespan			ACT		
	Average	Median	Stdev.	50th	90th	99th
DRR vs. TRC	11.5%	13.4%	5.7%	2.3%	6.1%	11.0%

Improvement of DRR over TRC as an alternative metric for executor placement

Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR

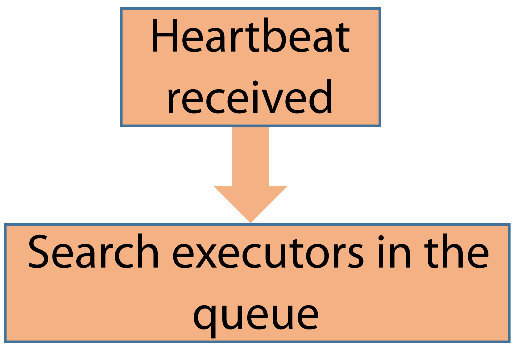
Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR

Heartbeat received

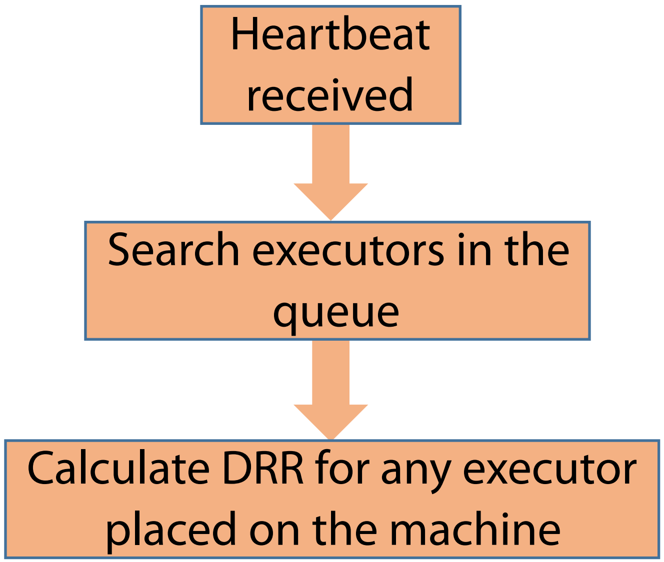
Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR



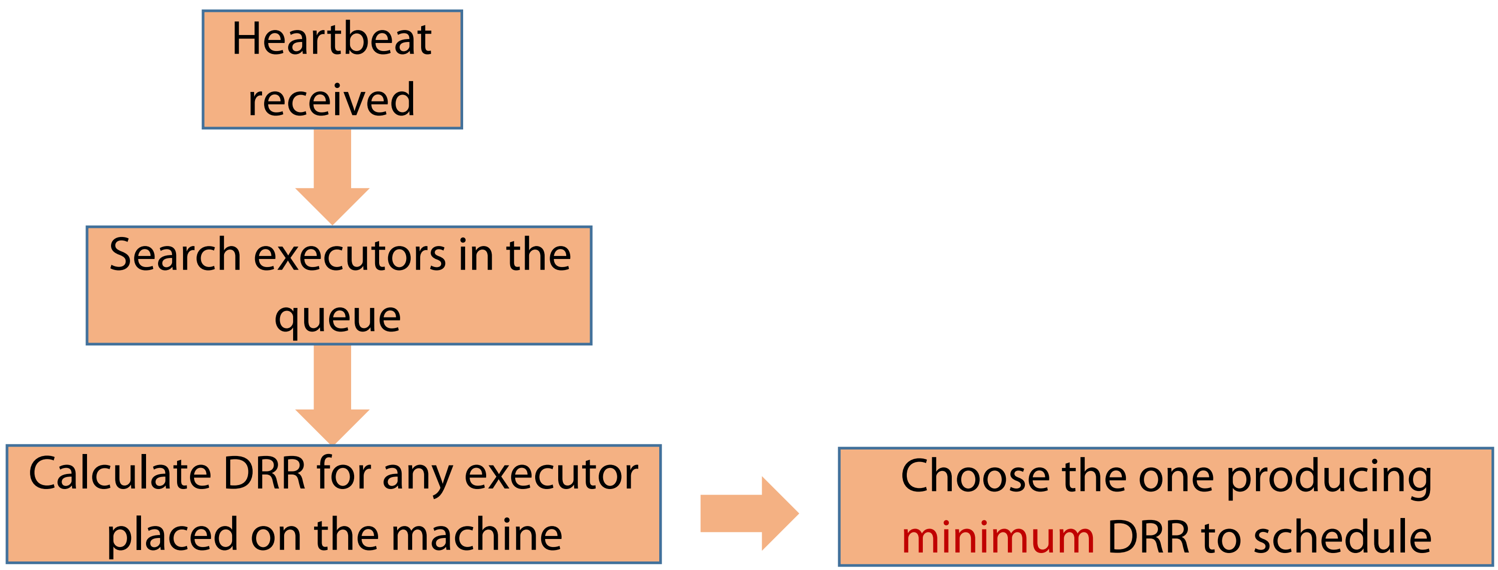
Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR



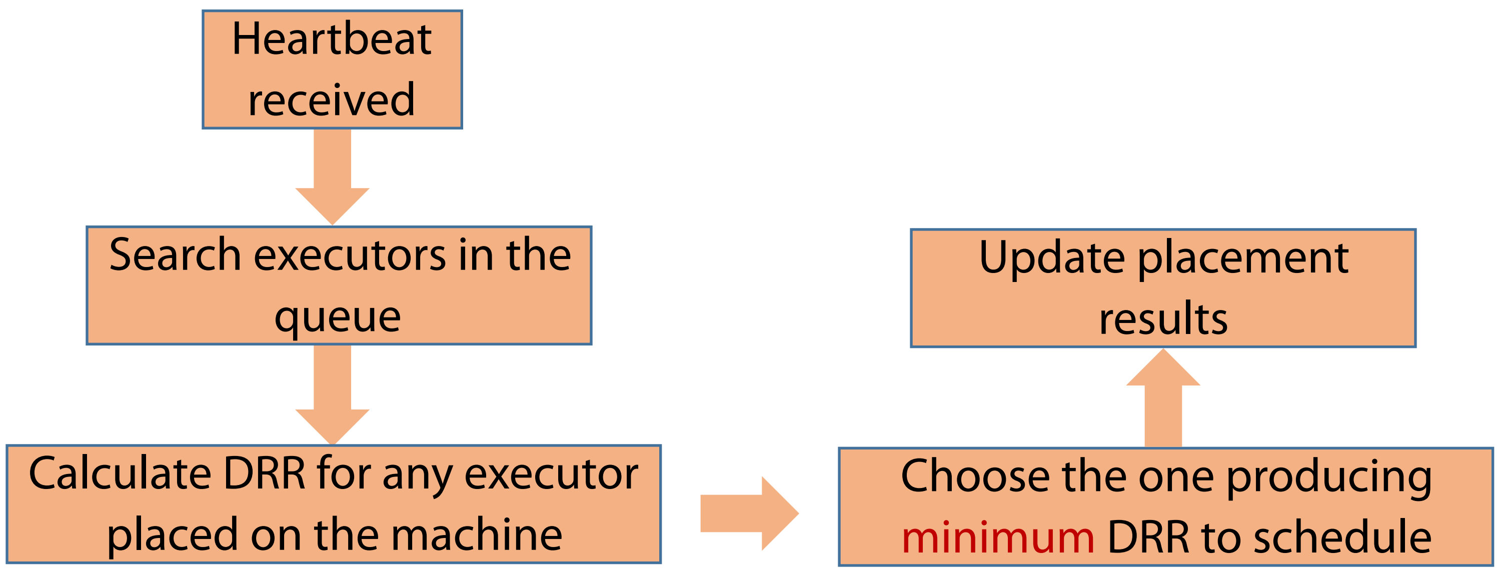
Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR



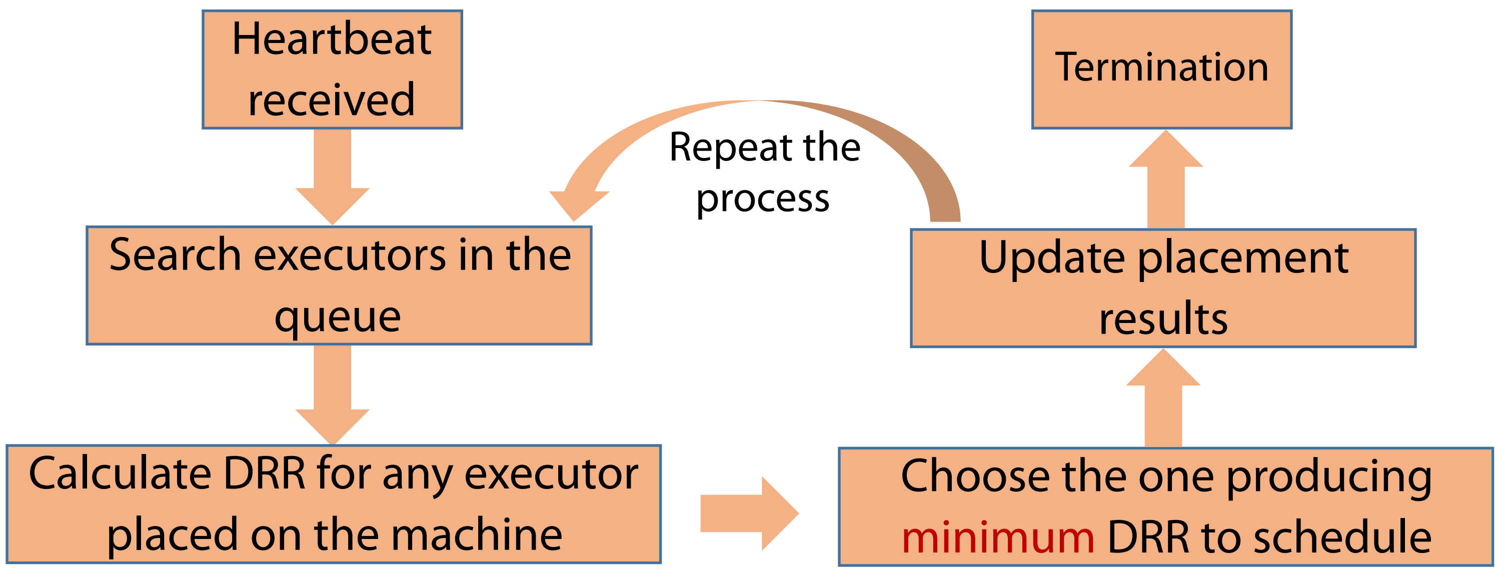
Elastic Executor Scheduling - MinFrag

- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR

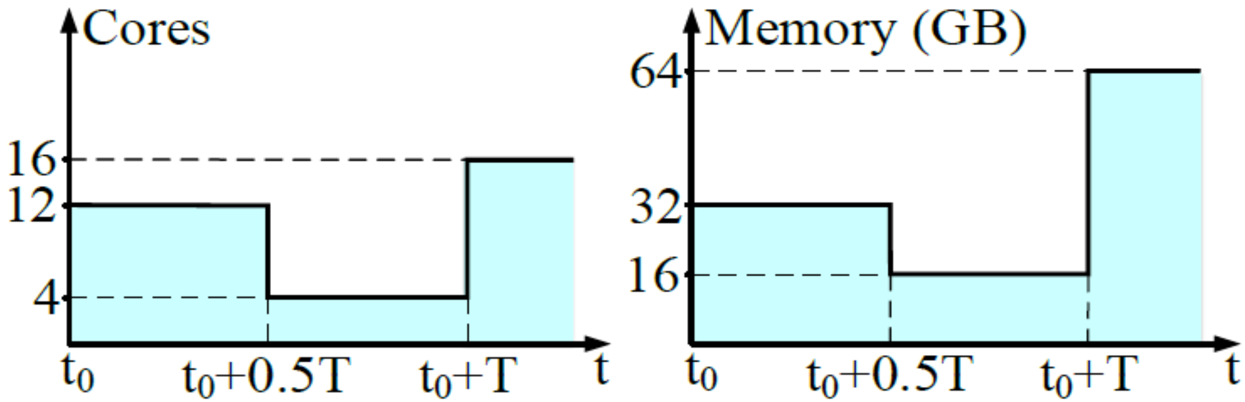


Elastic Executor Scheduling - MinFrag

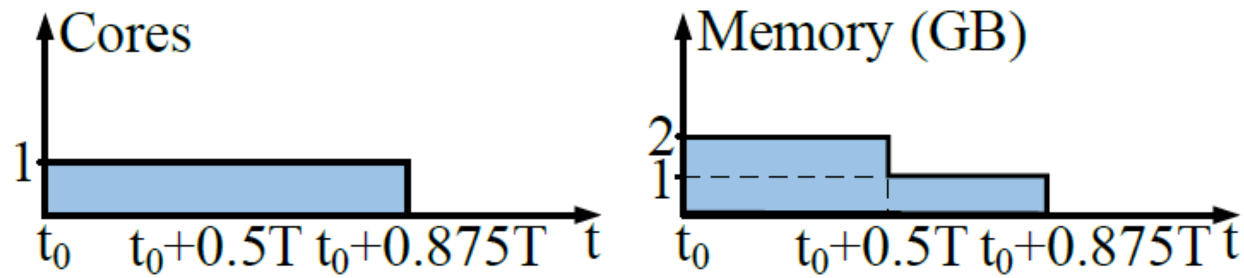
- Base on BFD (Best Fit Decreasing)
- Iteratively assigning the "largest" executor to a machine that yields the minimum DRR



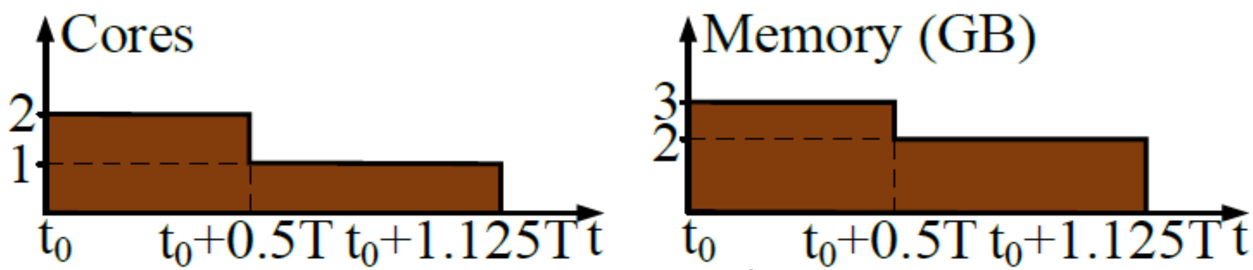
Elastic Executor Scheduling - MinFrag



(a) Available resources of machine

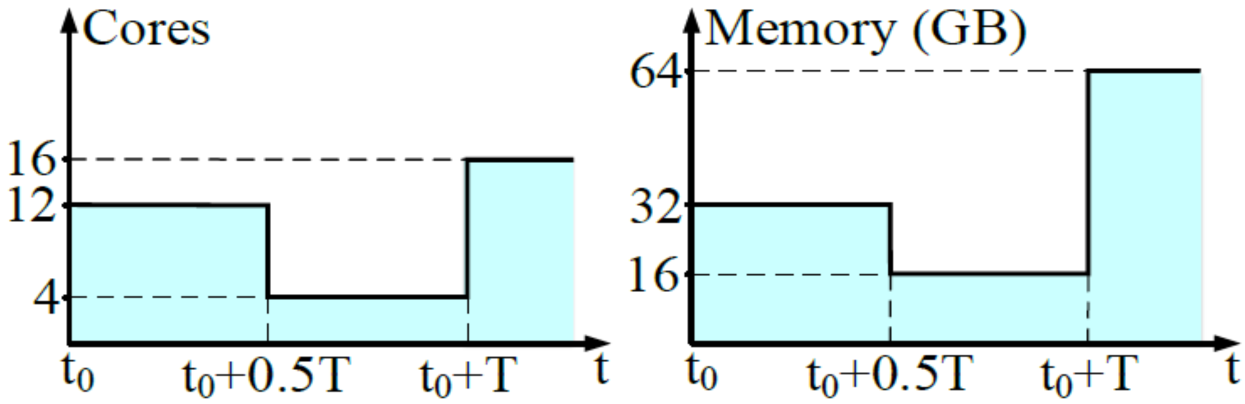


(b) Resource demands of executor 1

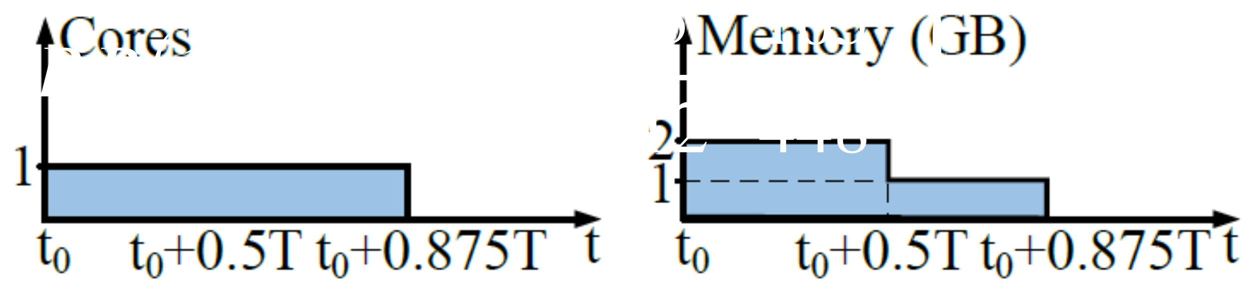


(c) Resource demands of executor 2

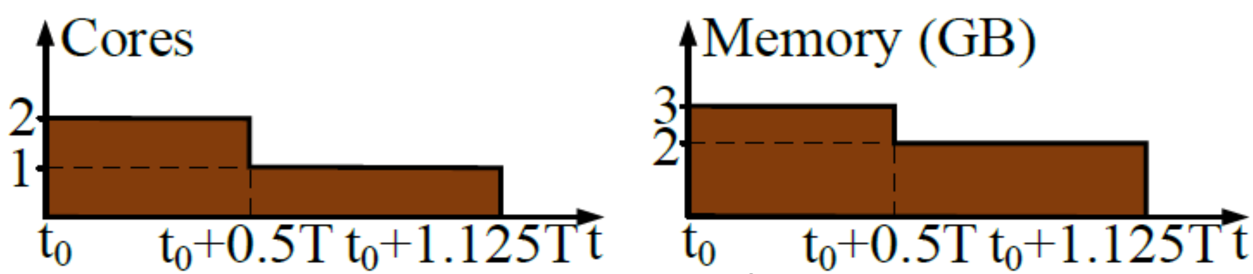
Elastic Executor Scheduling - MinFrag



(a) Available resources of machine

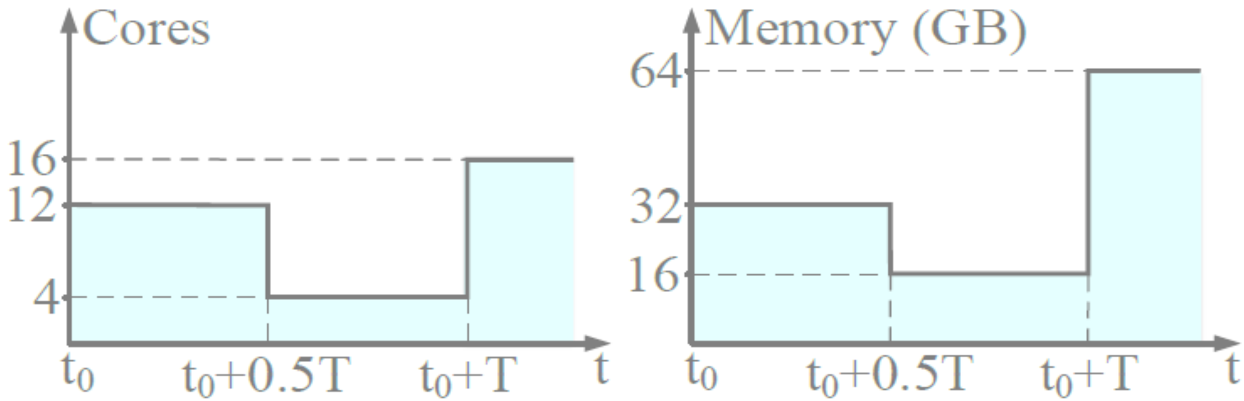


(b) Resource demands of executor 1

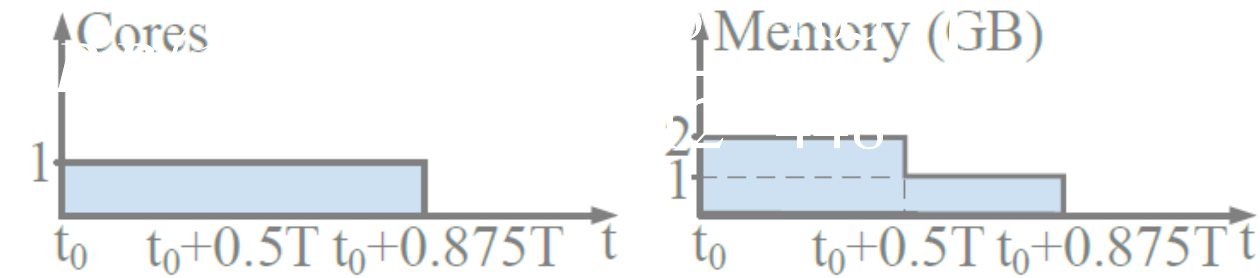


(c) Resource demands of executor 2

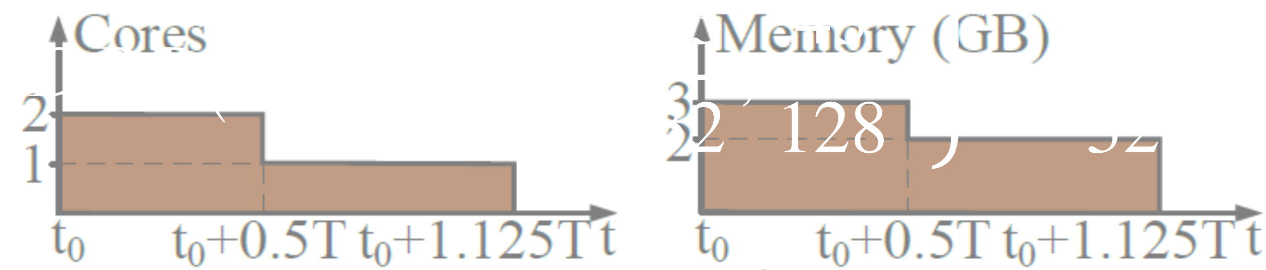
Elastic Executor Scheduling - MinFrag



(a) Available resources of machine



(b) Resource demands of executor 1



(c) Resource demands of executor 2

Prediction Module

- Recurring workloads
  - Average resource time series of the latest 3 runs as the prediction result
- New workloads
  - Support Vector Regression

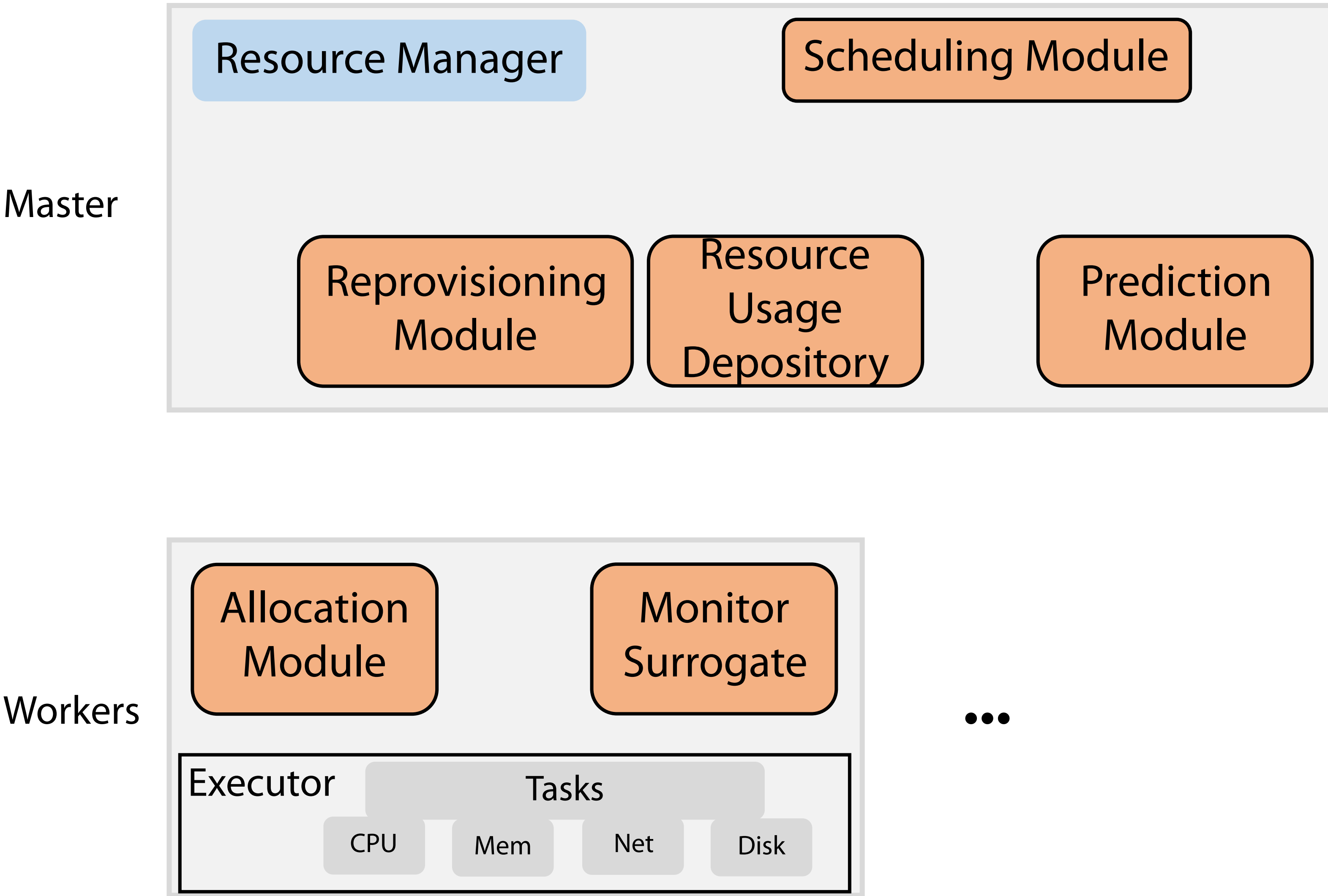
Dynamic Reprovisioning

- To prevent possible prediction errors and unpredicted issues
- Mechanism
  - Monitoring stage execution time
  - Once observing longer than 1.1x expected one
  - Allocating all remaining resource to the executor for one monitoring period

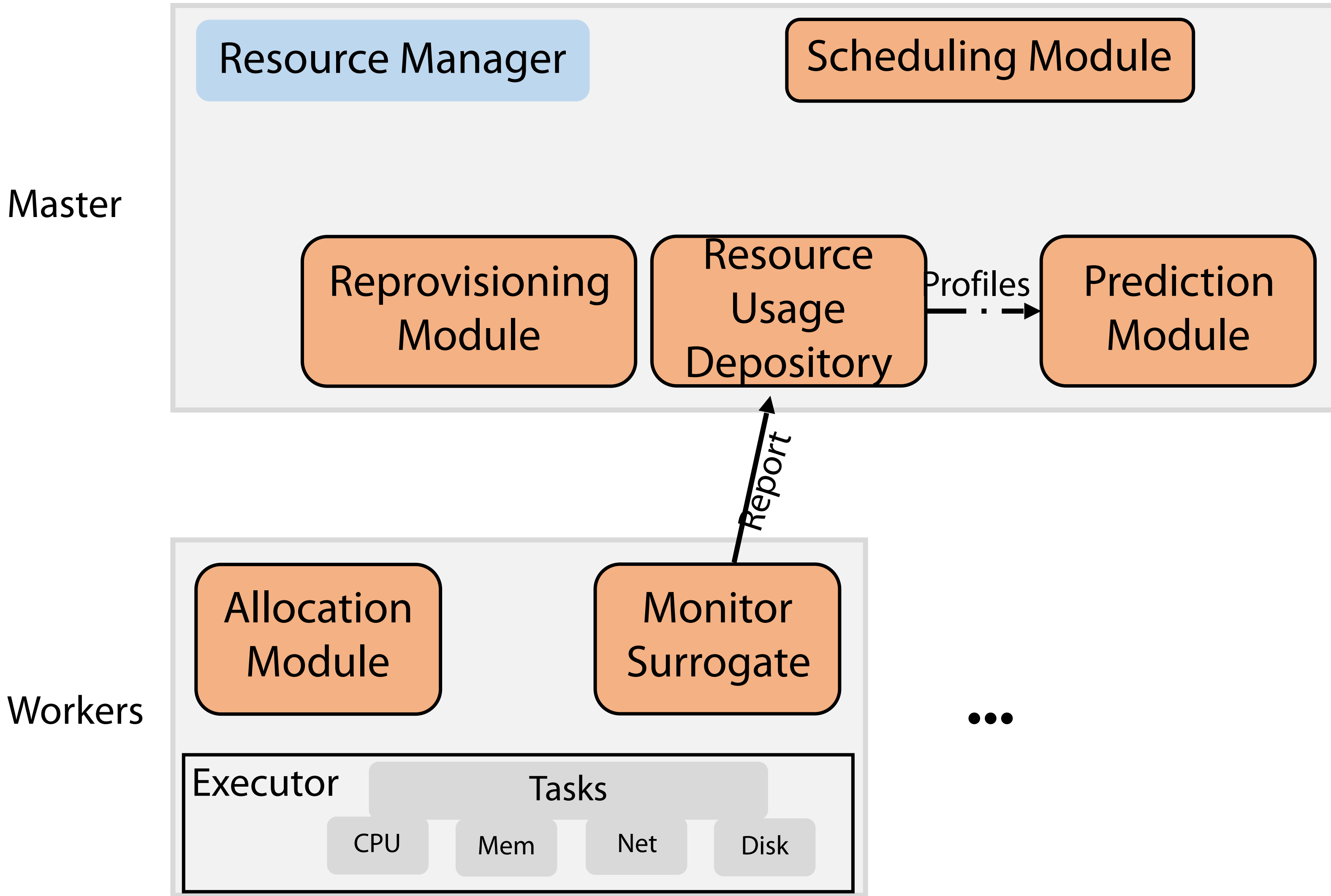
Implementation

- Spark 2.1.0
- Allocation Module (Cgroups, modified OpenJDK)
- Scheduling Module
- Resource Usage Depository
- Reprovisioning Module
- Prediction Module
- Monitor Surrogate

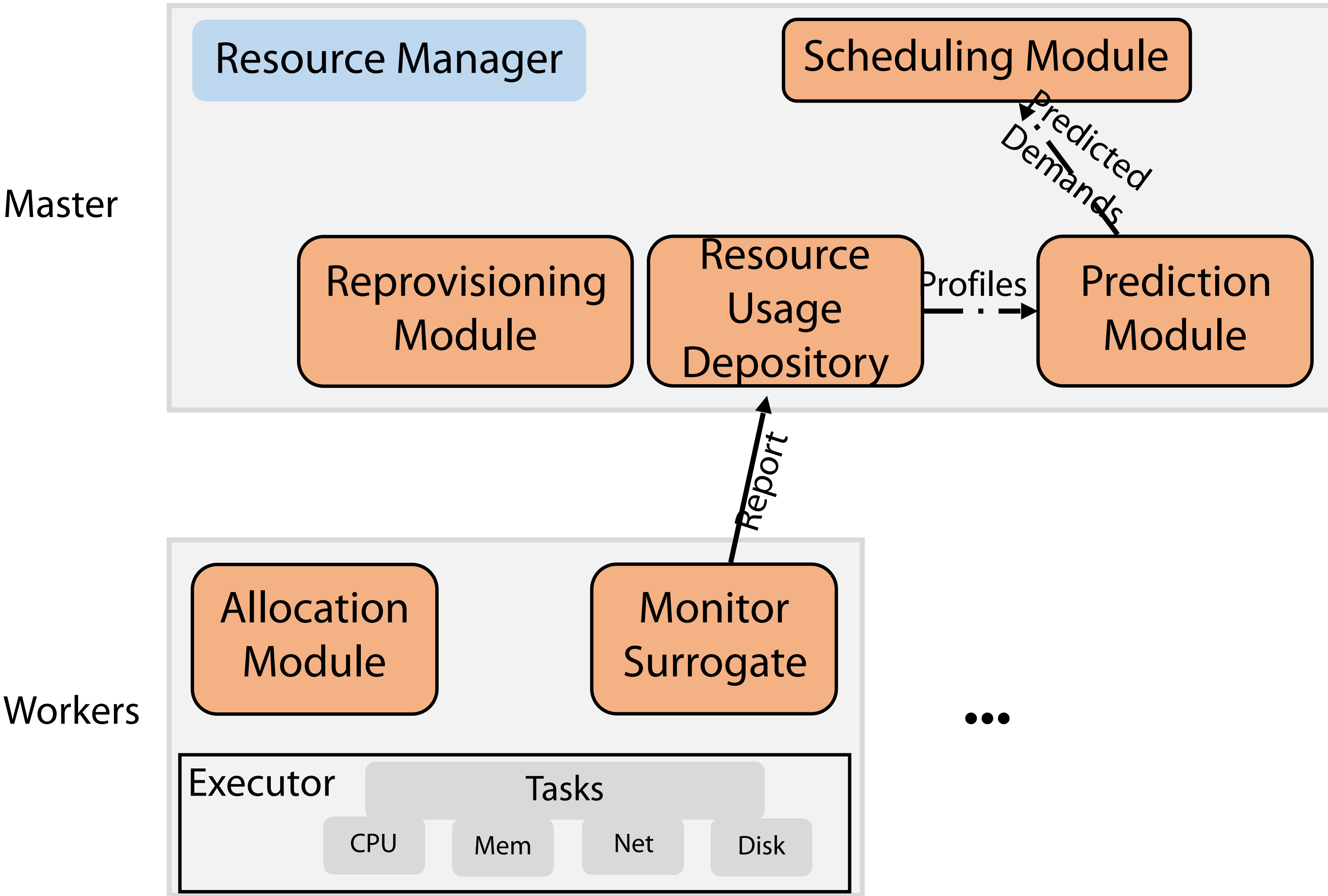
Elasecutor System



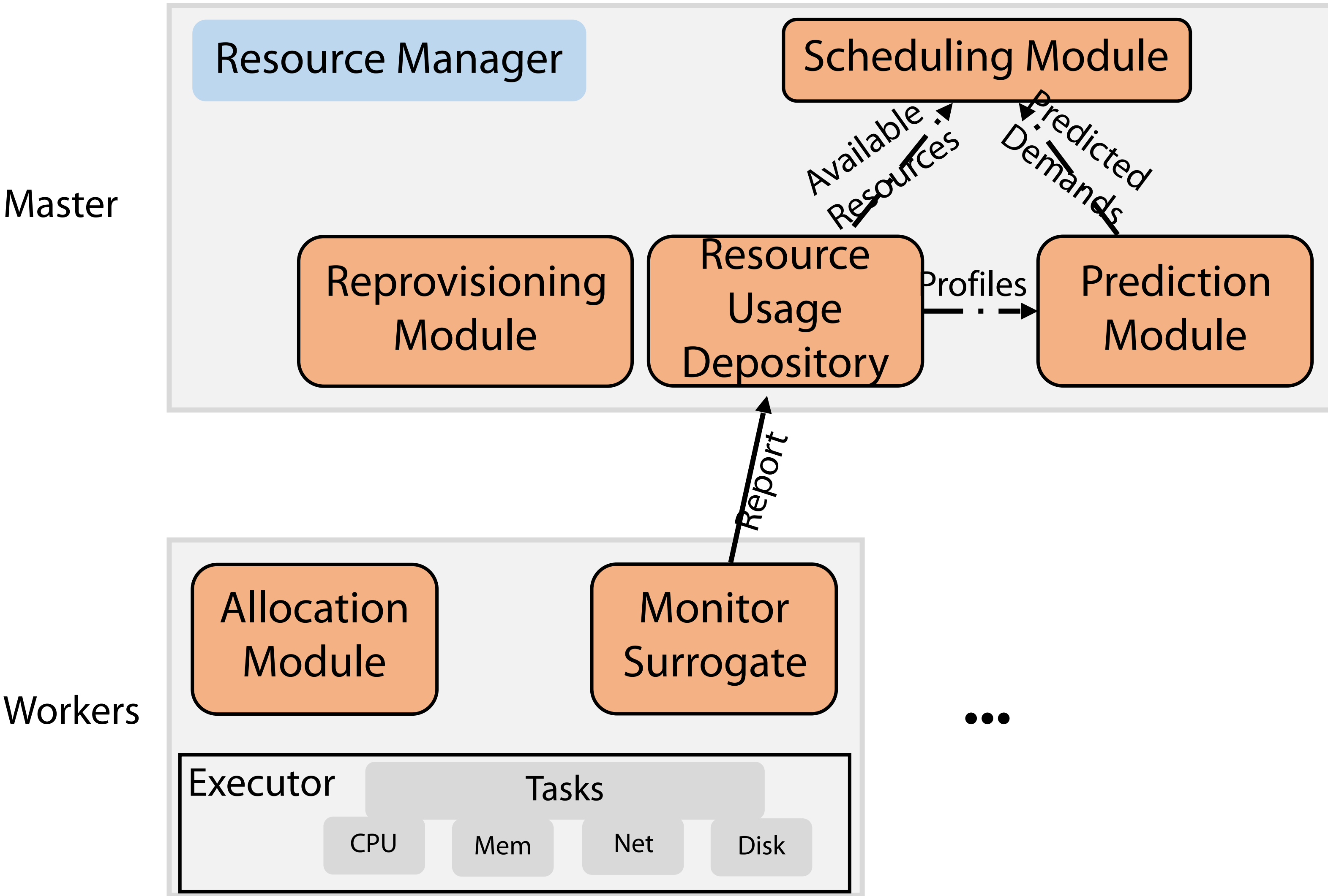
Elasecutor System



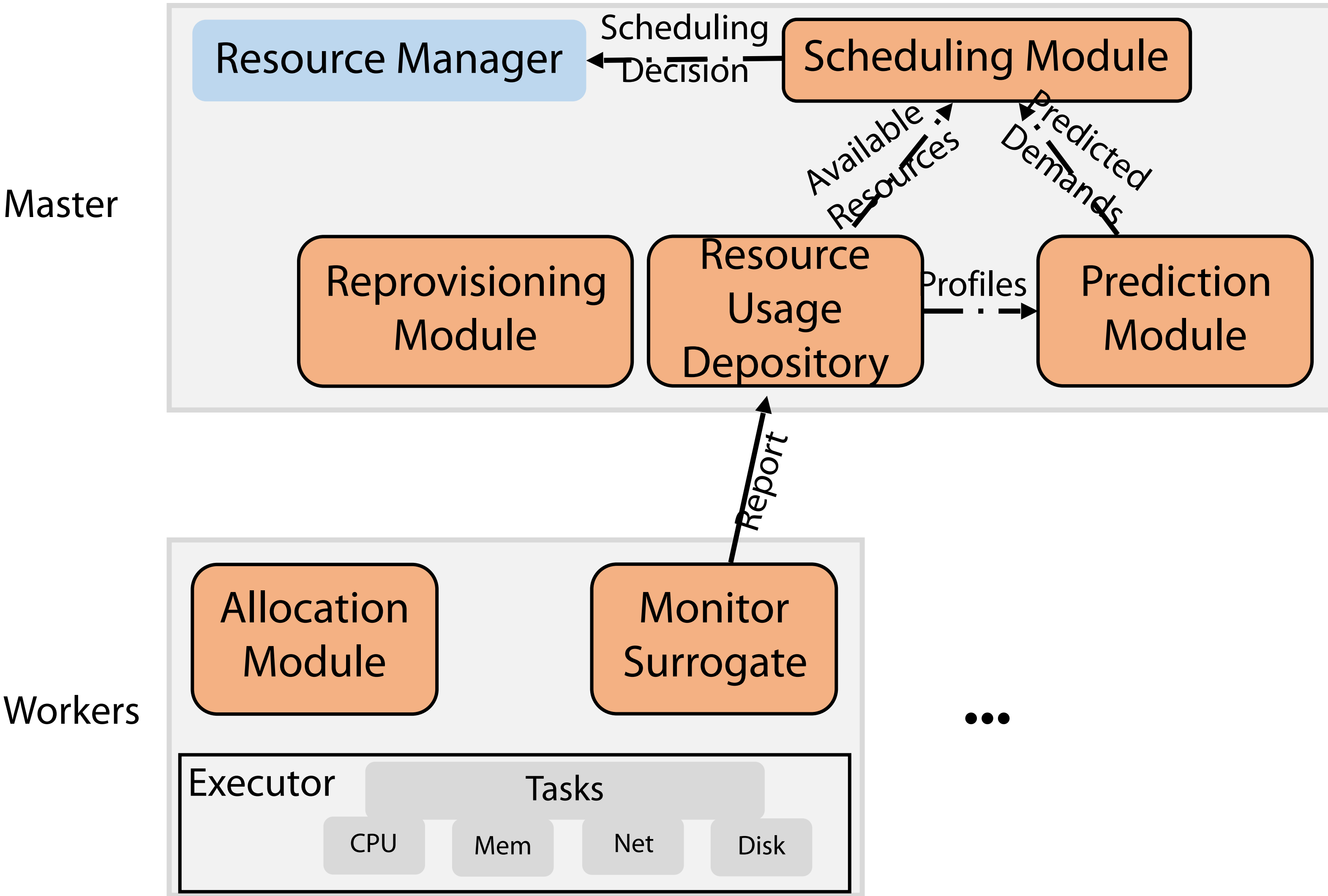
Elasecutor System



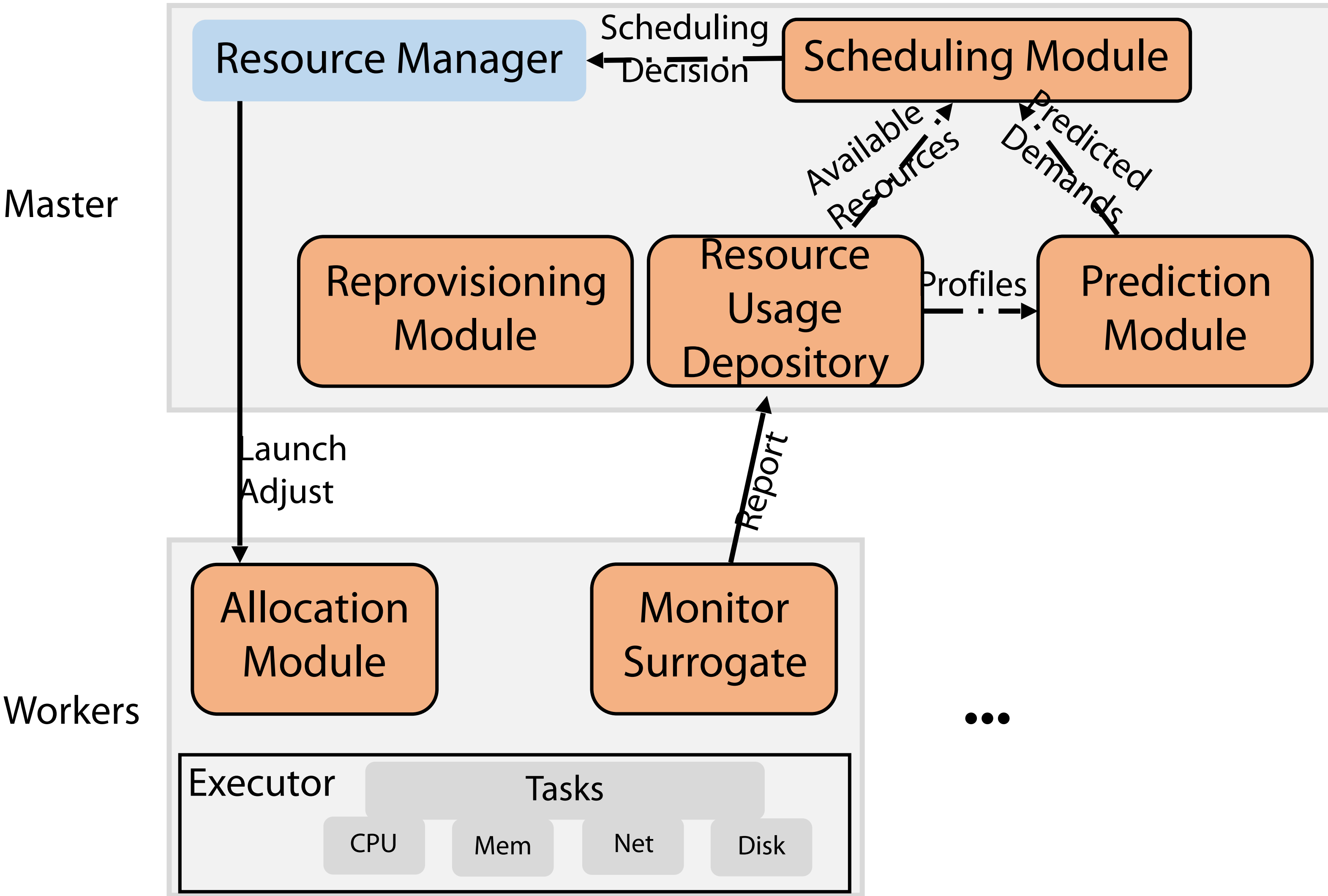
Elasecutor System



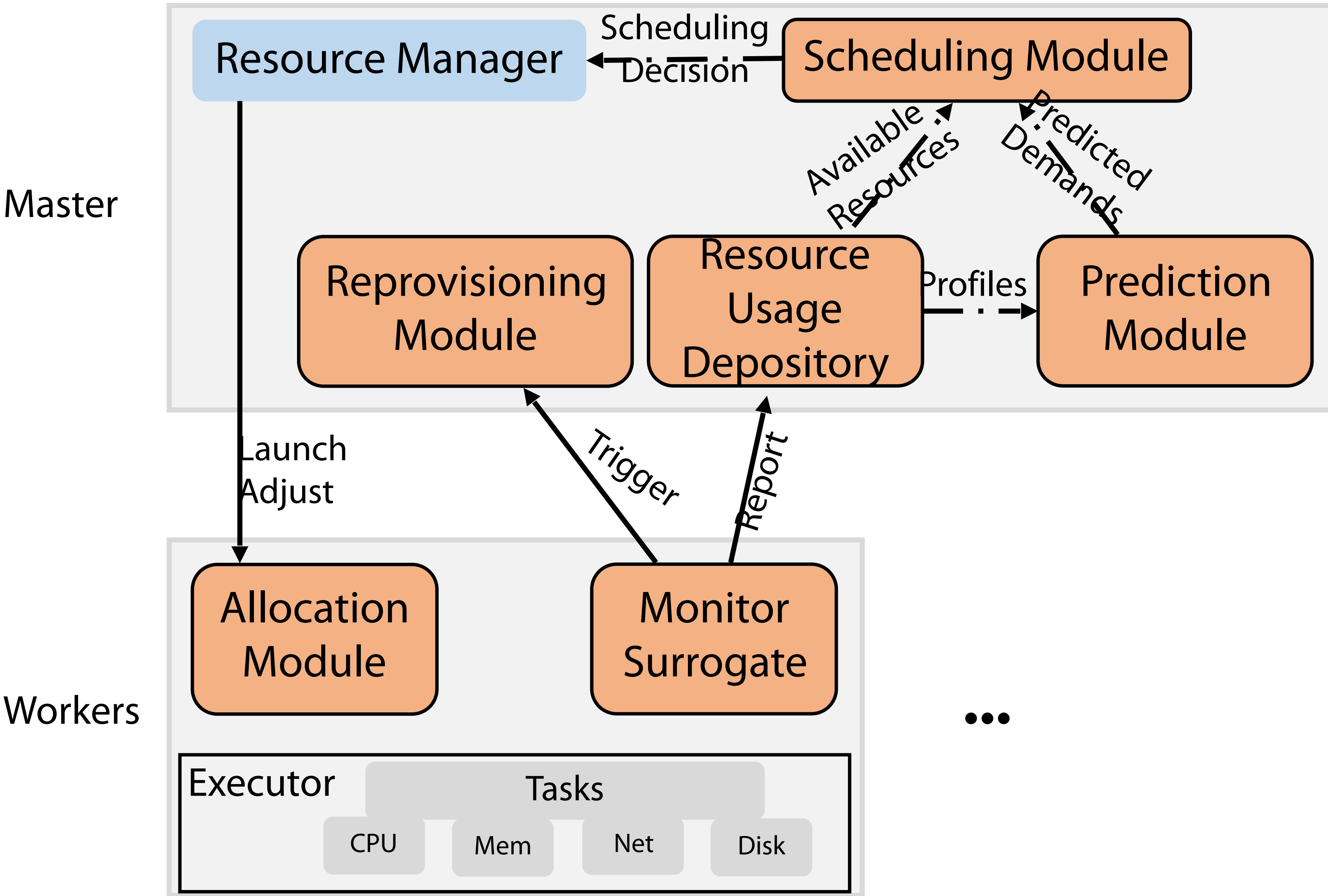
Elasecutor System



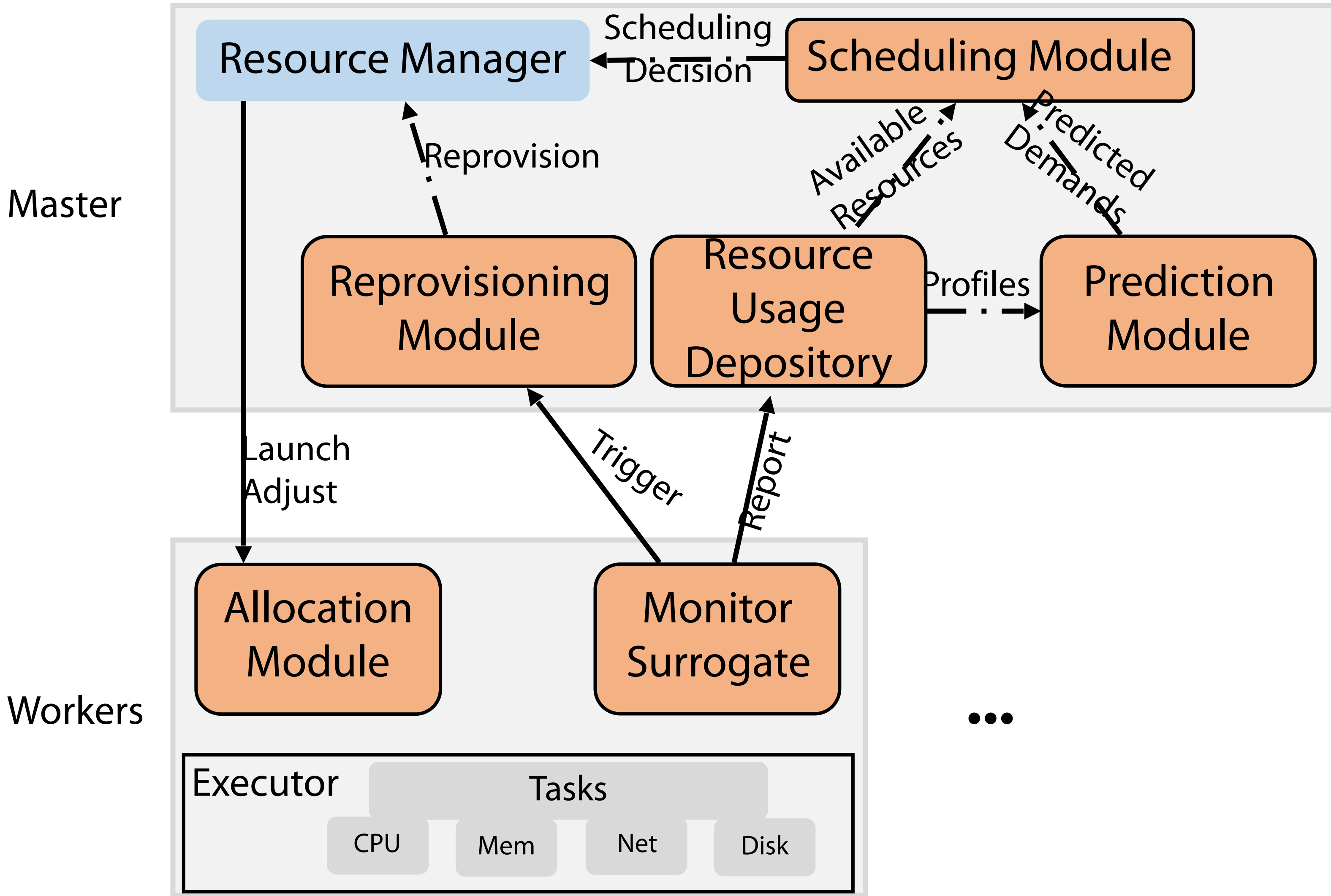
Elasecutor System



Elasecutor System



Elasecutor System



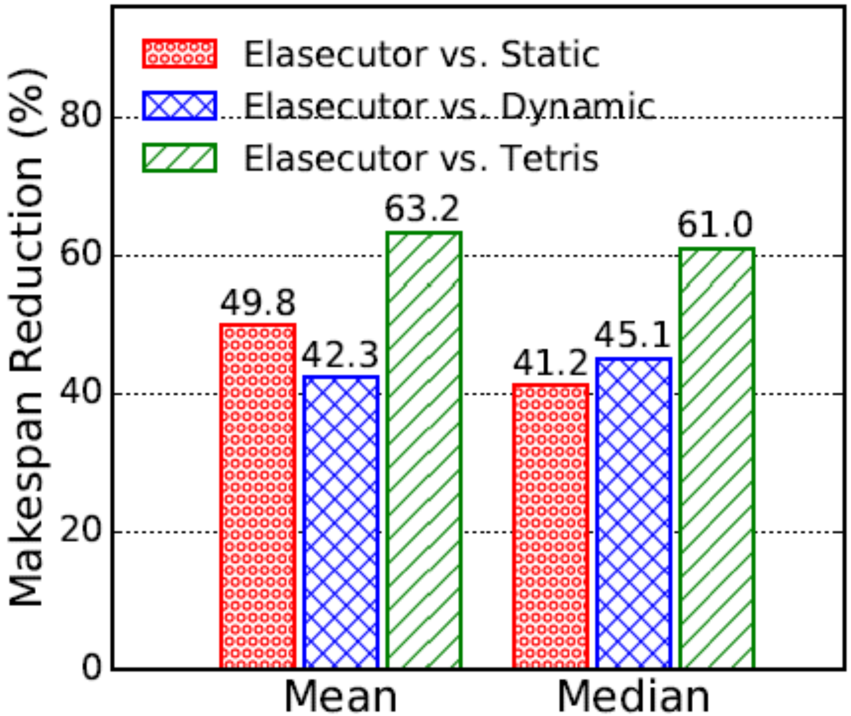
Testbed Experiments

- Testbed Setup
  - 35 dell servers
  - Each server with two CPUs, 64GB RAM, and a quad-port 10GbE NIC
  - A 10GbE Switch
- Methodology
  - 120 recurring applications with different workloads, input data sizes, and resource settings
  - 12 new applications
  - Arriving according to a Poisson process

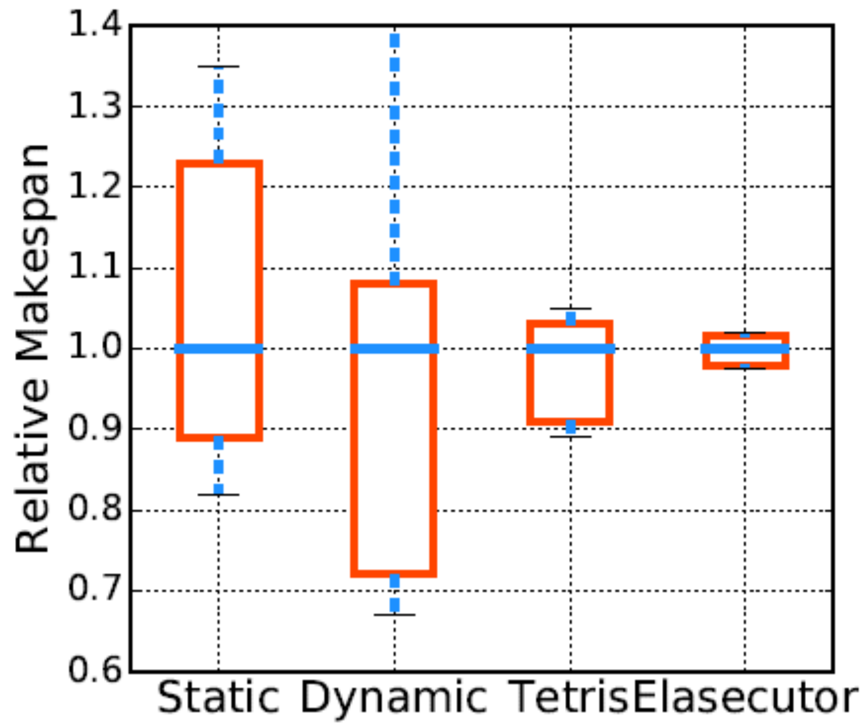
Schemes Compared

- Static
  - Statically allocating CPU and memory for each executor based on peak demands
  - Launching a fixed number of executors
- Dynamic
  - Scaling the number of executors dynamically,
  - each executor allocated a multiple of <1 core, 2GB RAM>
- Tetris (SIGCOMM'14)
  - Allocating peak demanded resources to executors
  - BFD-like algorithm for executor placement

Evaluation - Makespan



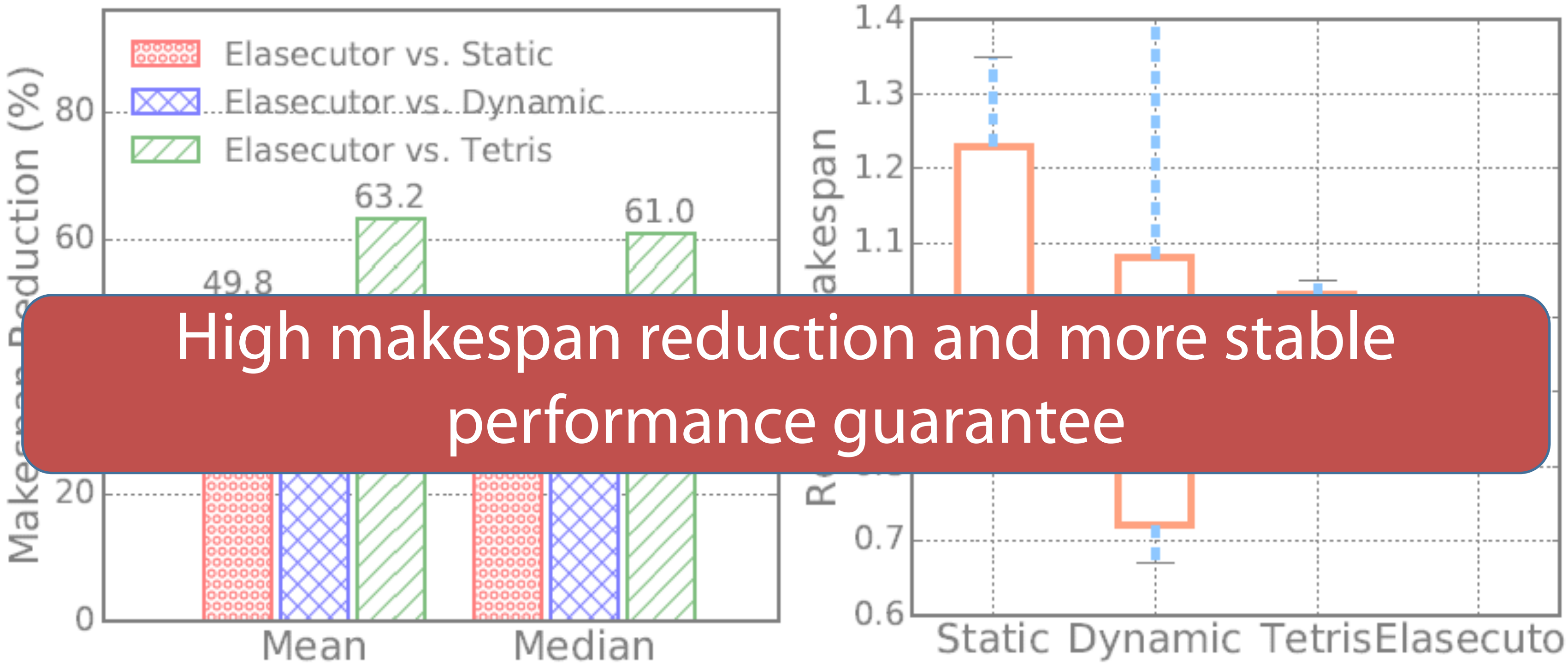
(a) Makespan Reduction



(b) Stability of makespan

Makespan measures the total time used to complete all applications

Evaluation - Makespan

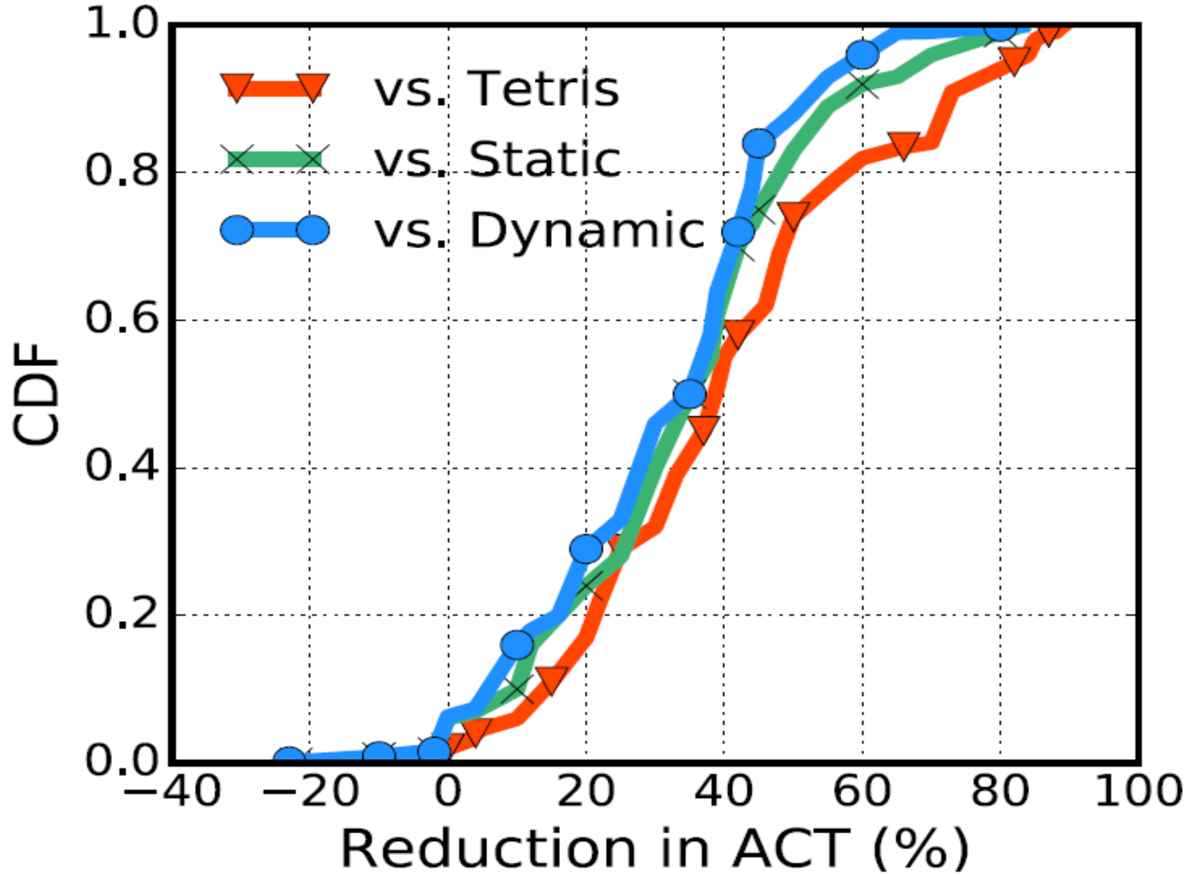


High makespan reduction and more stable performance guarantee

(a) Makespan Reduction (b) Stability of makespan

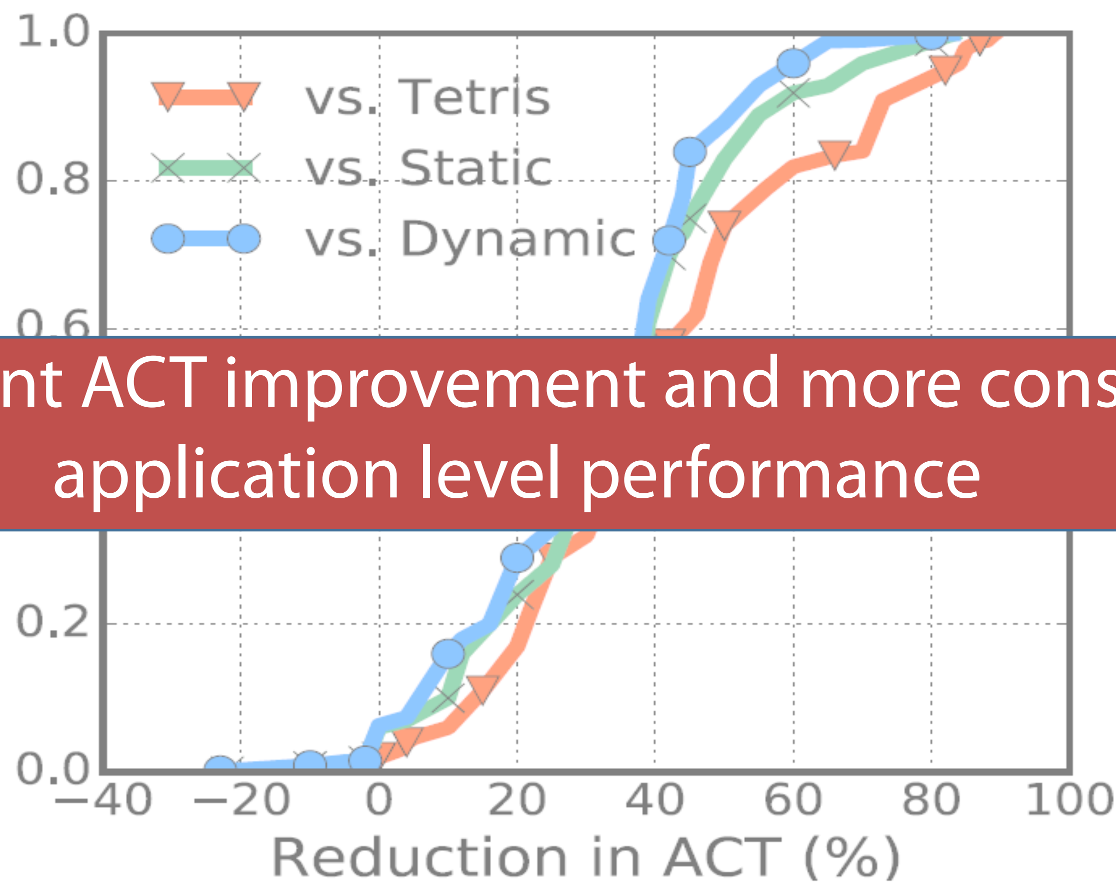
Makespan measures the total time used to complete all applications

Evaluation - ACT



The CDFs of reduction in application completion time

Evaluation - ACT



Significant ACT improvement and more consistent application level performance

The CDFs of reduction in application completion time

Evaluation - Resource Utilization

Utilization Improvement (%)	CPU	Memory	Network	Disk I/O
Elasecutor vs. *Static*	43.4	29.5	40.8	25.4
Elasecutor vs. *Dynamic*	27.2	22.6	33.4	40.0
Elasecutor vs. *Tetris*	28.6	25.2	55.6	43.9

Elasecutor's average utilization improvement over other policies

Evaluation - Resource Utilization

Utilization Improvement (%)	CPU	Memory	Network	Disk I/O
Elasecutor vs. *Static*	43.4	29.5	40.8	25.4
Elasecutor vs. *Dynamic*	27.2	22.6	33.4	40.0
Elasecutor vs. *Tetris*	28.6	25.2	55.6	43.9

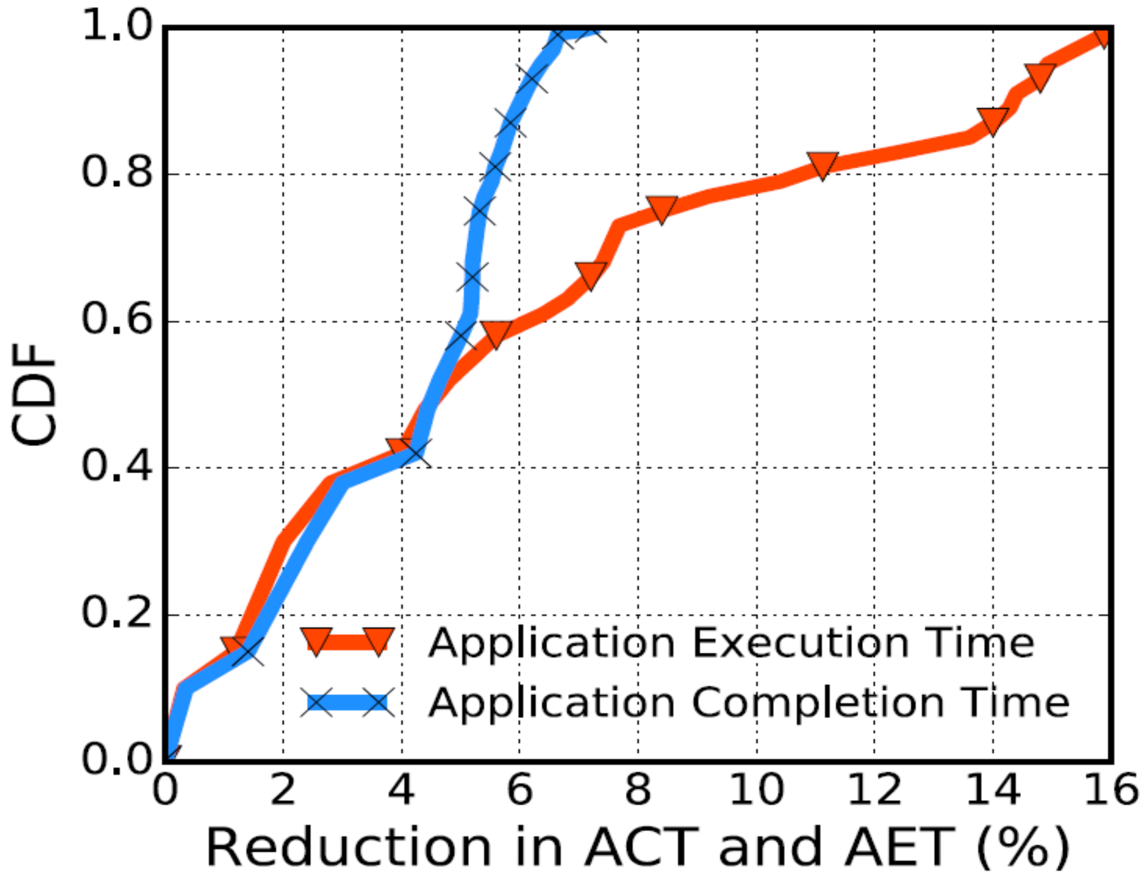
Elasecutor's average utilization improvement over other policies

Evaluation - Resource Utilization

Utilization Improvement (%)	CPU	Memory	Network	Disk I/O
Elasecutor vs. *Static*	43.4	29.5	40.8	25.4
Elasecutor vs. *Dynamic*	27.2	22.6	33.4	40.0
Elasecutor vs. *Tetris*	28.6	25.2	55.6	43.9

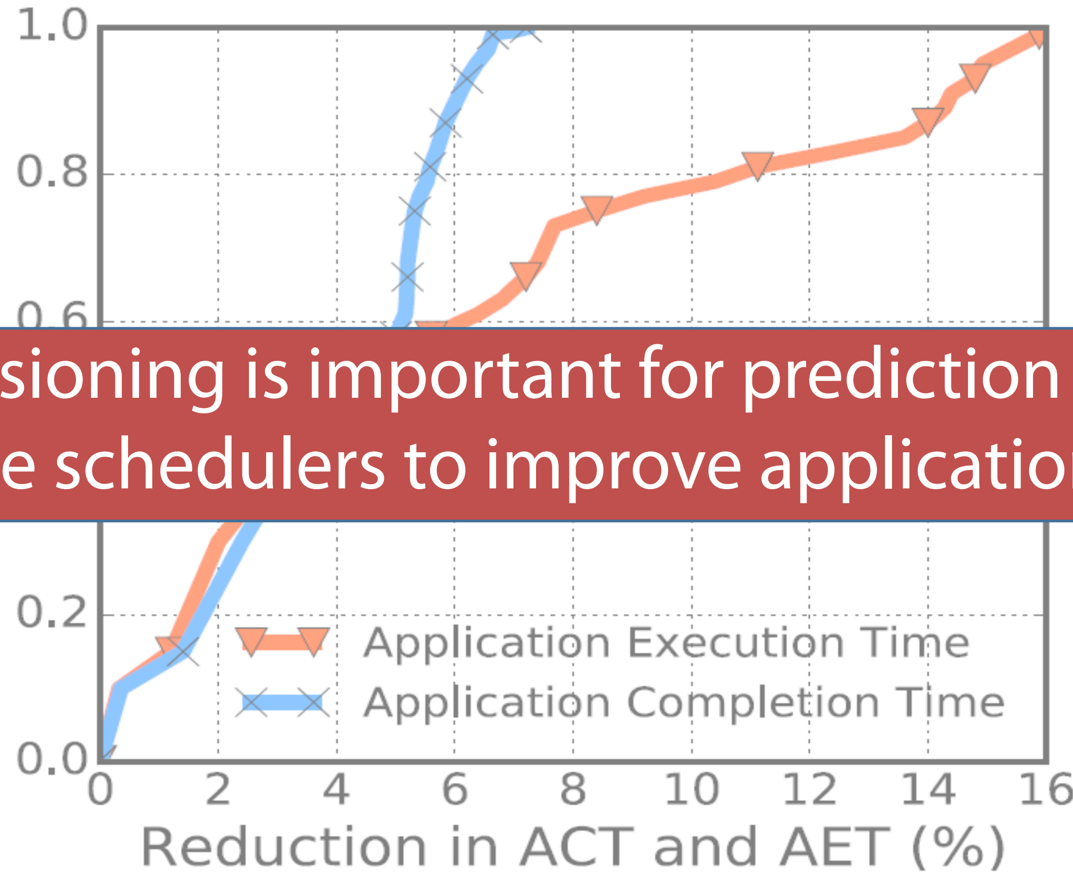
Elasecutor's average utilization improvement over other policies

Evaluation - Microbenchmark



CDFs of reductions in ACT and AET by comparing Elasecutor with and without reprovisioning module

Evaluation - Microbenchmark



Reprovisioning is important for prediction based resource schedulers to improve application QoS

CDFs of reductions in ACT and AET by comparing Elasecutor with and without reprovisioning module

Conclusion

- Elasecutor
  - Elastically allocating resources to avoid overallocation
  - Placing executors strategically to minimize multi-resource fragmentation
- Experiment results
  - Reducing makespan by more than 42% on average
  - Reducing the median application completion time by up to 40%
  - Improving cluster resource utilization by up to 55%

Thanks!
Q & A

Overhead

Resource Consumption	CPU	Memory	Total executor profile size
Monitor surrogate	0.3%	0.1%	12.1 KB

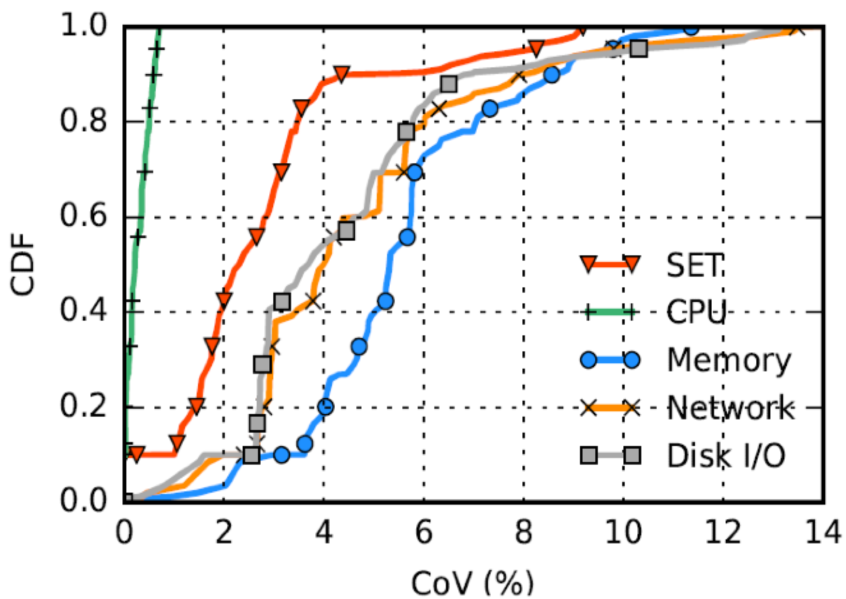
Monitor surrogate ‘s resource consumption

Time to process (ms)	Unmodified Spark	Elasecutor
Worker heartbeat	~0.031	~0.035
Application driver heartbeat	~0.153	~0.155

Resource scheduler’s processing delay

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight



The CDFs of coefficient of variations

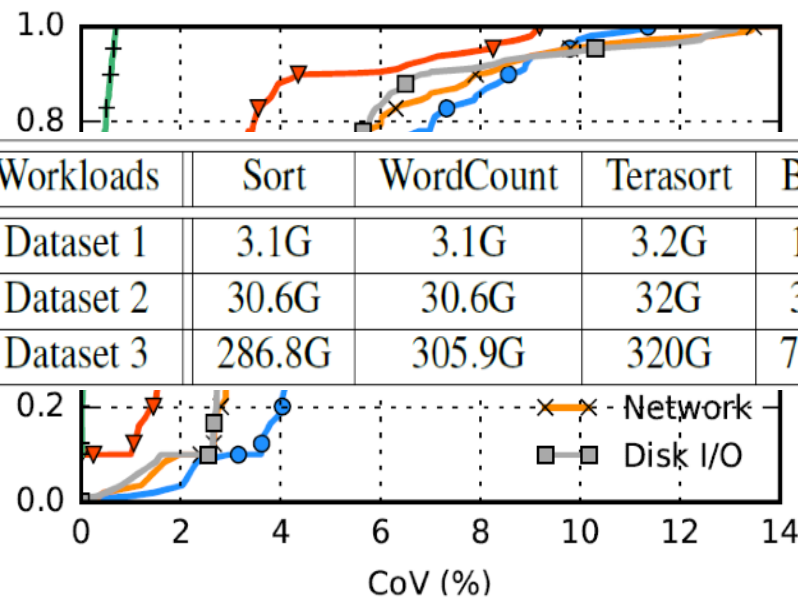
CoV Statistics (%)	Percentiles			
	10th	50th	90th	99th
SET	0.7	2.6	5.5	9.1
CPU	0	0.3	0.6	0.7
Memory	3.1	5.6	8.6	11.0
Network	2.4	4.2	7.9	13.4
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight

Workloads	Sort	WordCount	Terasort	Bayes	K-means	LR	PageRank	NWeight
Dataset 1	3.1G	3.1G	3.2G	1.8G	3.7G	7.5G	1.7M	37.5M
Dataset 2	30.6G	30.6G	32G	3.5G	18.7G	22.4G	247.9M	294.5M
Dataset 3	286.8G	305.9G	320G	70.1G	37.4G	37.3G	2.8G	2.7G



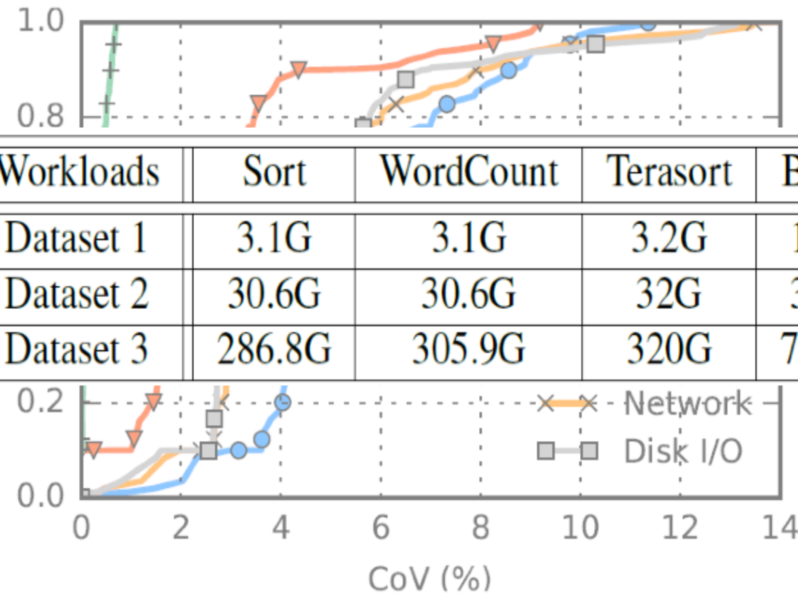
The CDFs of coefficient of variations

	Percentiles			
Network	[illegible]	[illegible]	[illegible]	[illegible]
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight



Workloads	Sort	WordCount	Terasort	Bayes	K-means	LR	PageRank	NWeight
Dataset 1	3.1G	3.1G	3.2G	1.8G	3.7G	7.5G	1.7M	37.5M
Dataset 2	30.6G	30.6G	32G	3.5G	18.7G	22.4G	247.9M	294.5M
Dataset 3	286.8G	305.9G	320G	70.1G	37.4G	37.3G	2.8G	2.7G

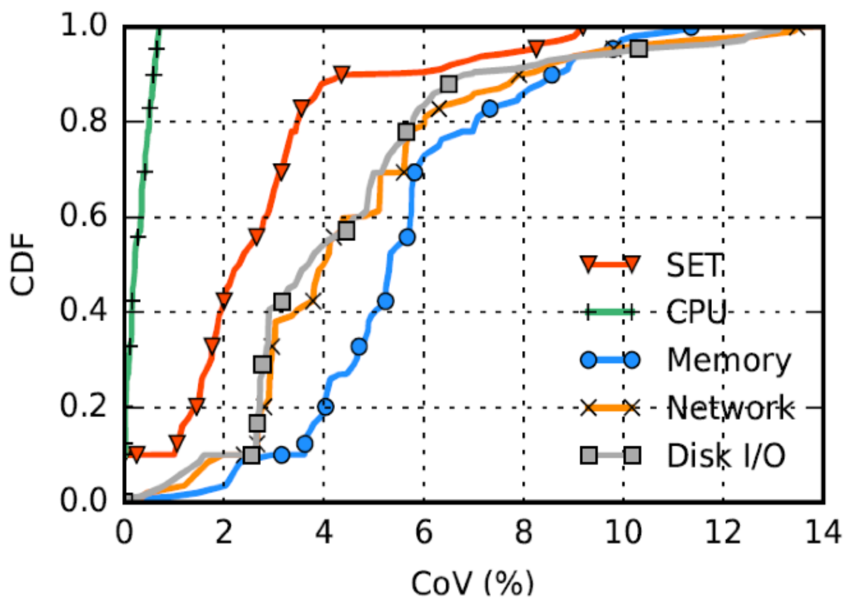
	Percentiles			
Disk I/O	2.5	2.9	6.8	12.9

The CDFs of coefficient of variations

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight



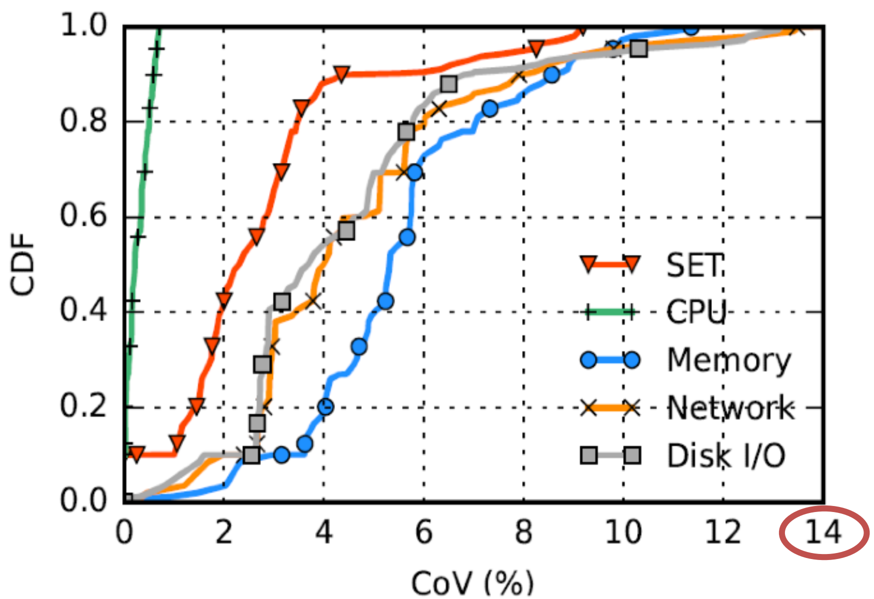
The CDFs of coefficient of variations

CoV Statistics (%)	Percentiles			
	10th	50th	90th	99th
SET	0.7	2.6	5.5	9.1
CPU	0	0.3	0.6	0.7
Memory	3.1	5.6	8.6	11.0
Network	2.4	4.2	7.9	13.4
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight



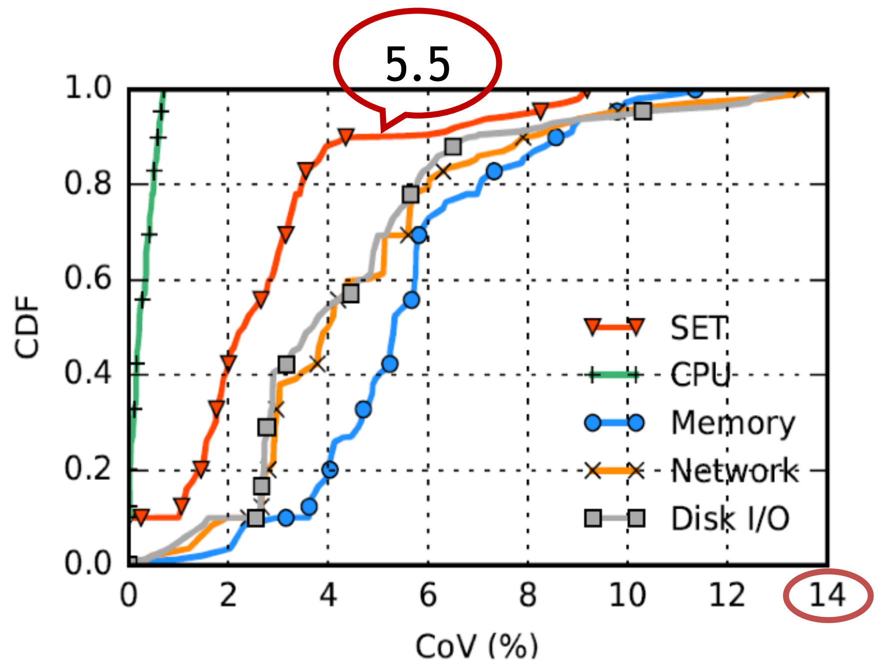
The CDFs of coefficient of variations

CoV Statistics (%)	Percentiles			
	10th	50th	90th	99th
SET	0.7	2.6	5.5	9.1
CPU	0	0.3	0.6	0.7
Memory	3.1	5.6	8.6	11.0
Network	2.4	4.2	7.9	13.4
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight



The CDFs of coefficient of variations

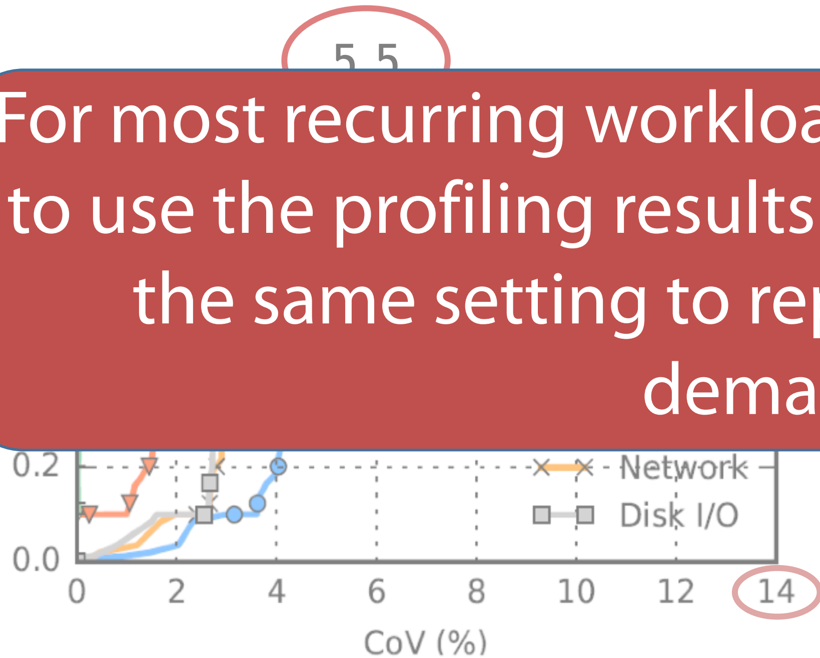
CoV Statistics (%)	Percentiles			
	10th	50th	90th	99th
SET	0.7	2.6	5.5	9.1
CPU	0	0.3	0.6	0.7
Memory	3.1	5.6	8.6	11.0
Network	2.4	4.2	7.9	13.4
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Predictability

- Workloads: Sort, WordCount, Terasort, Bayes, K-means, LR, PageRank, NWeight

For most recurring workloads, it is accurate enough to use the profiling results from previous runs with the same setting to represent the resource demands

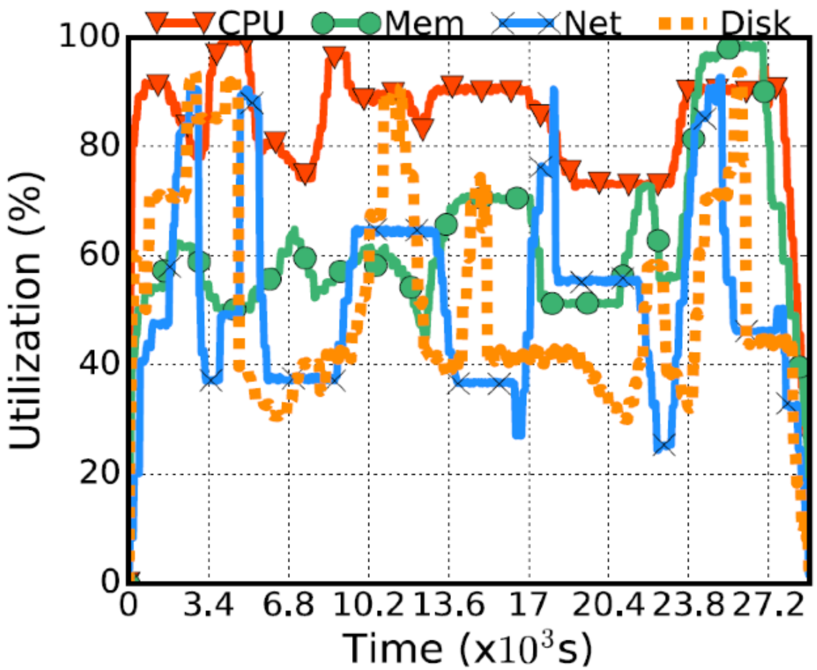


The CDFs of coefficient of variations

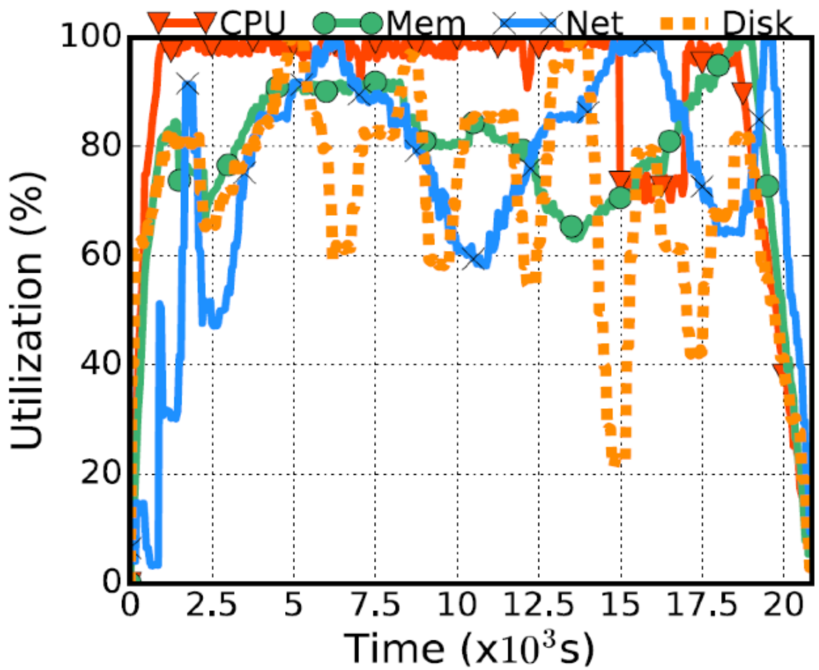
Disk I/O	2.5	2.9	6.8	12.9

Statistical analysis of CoVs

Resource Utilization

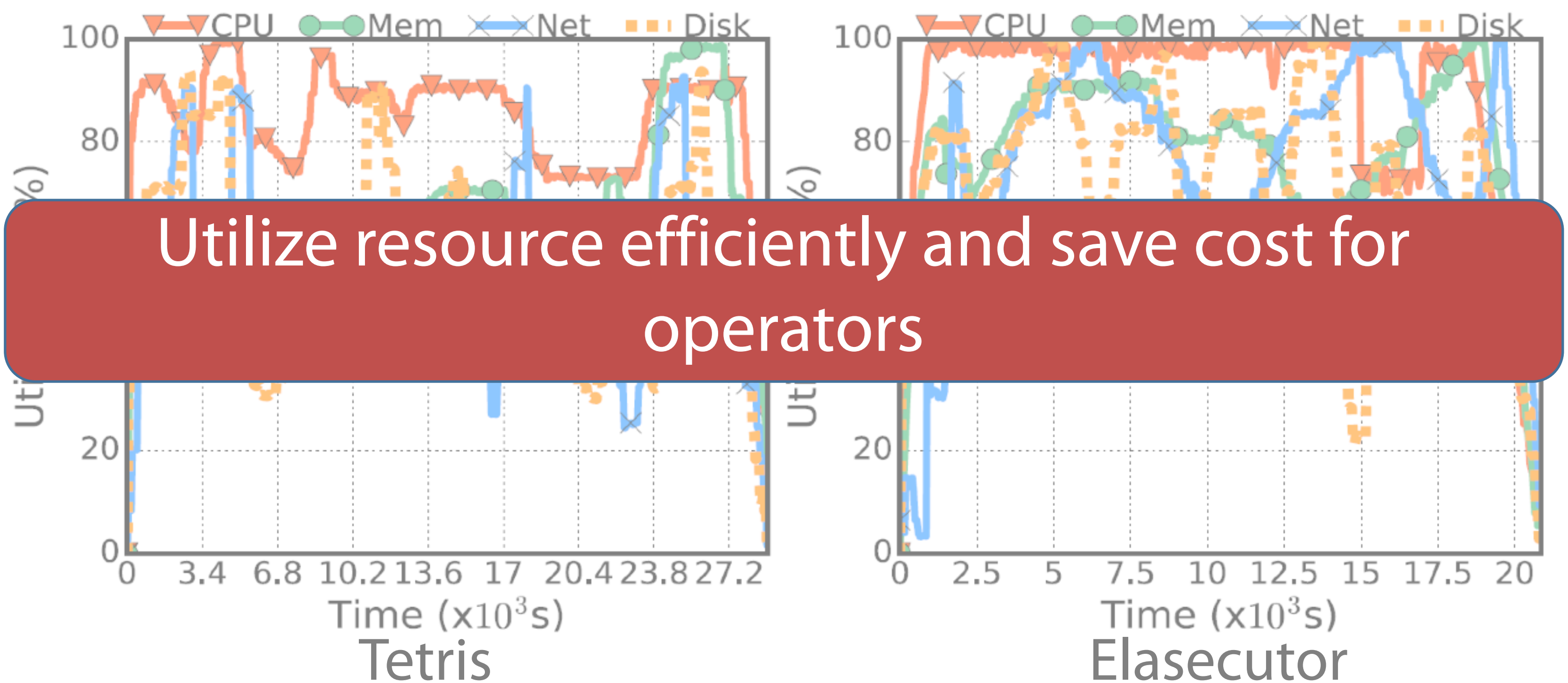


Tetris



Elasecutor

Resource Utilization



Utilize resource efficiently and save cost for operators