FAWKES

Balanced Resource Allocations Across Multiple Dynamic MapReduce Clusters

ACM SIGMETRICS 2014

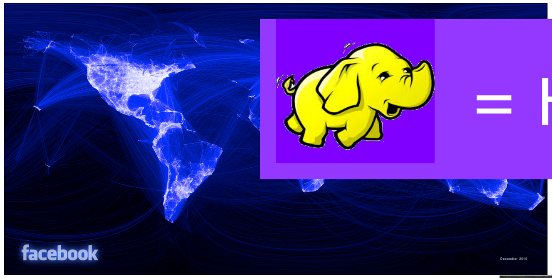
Bogdan Ghiţ, Nezih Yigitbasi, Alexandru Iosup, Dick Epema

Parallel and Distributed Systems
Delft University of Technology
Delft, the Netherlands

TUDelft COMMIT/ PDS Group Challenge the future DELFT DATA SCIENCE

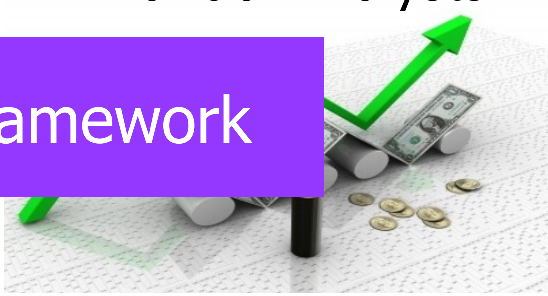
The “big data cake” problem

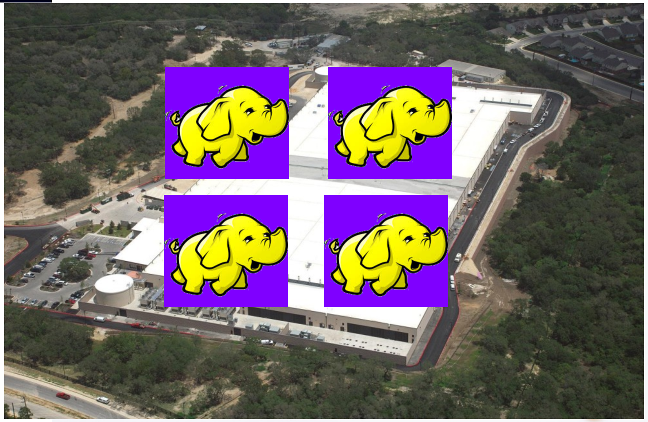
Online Social Networks



= Hadoop / MapReduce framework

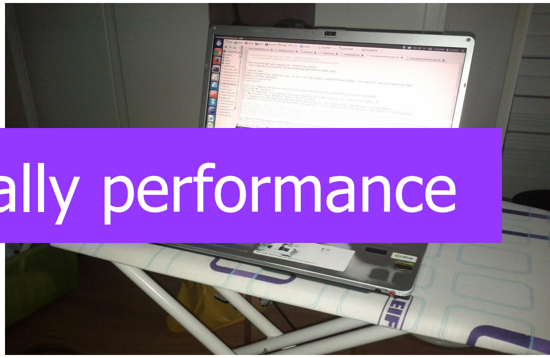
Financial Analysts





Universe Explorers

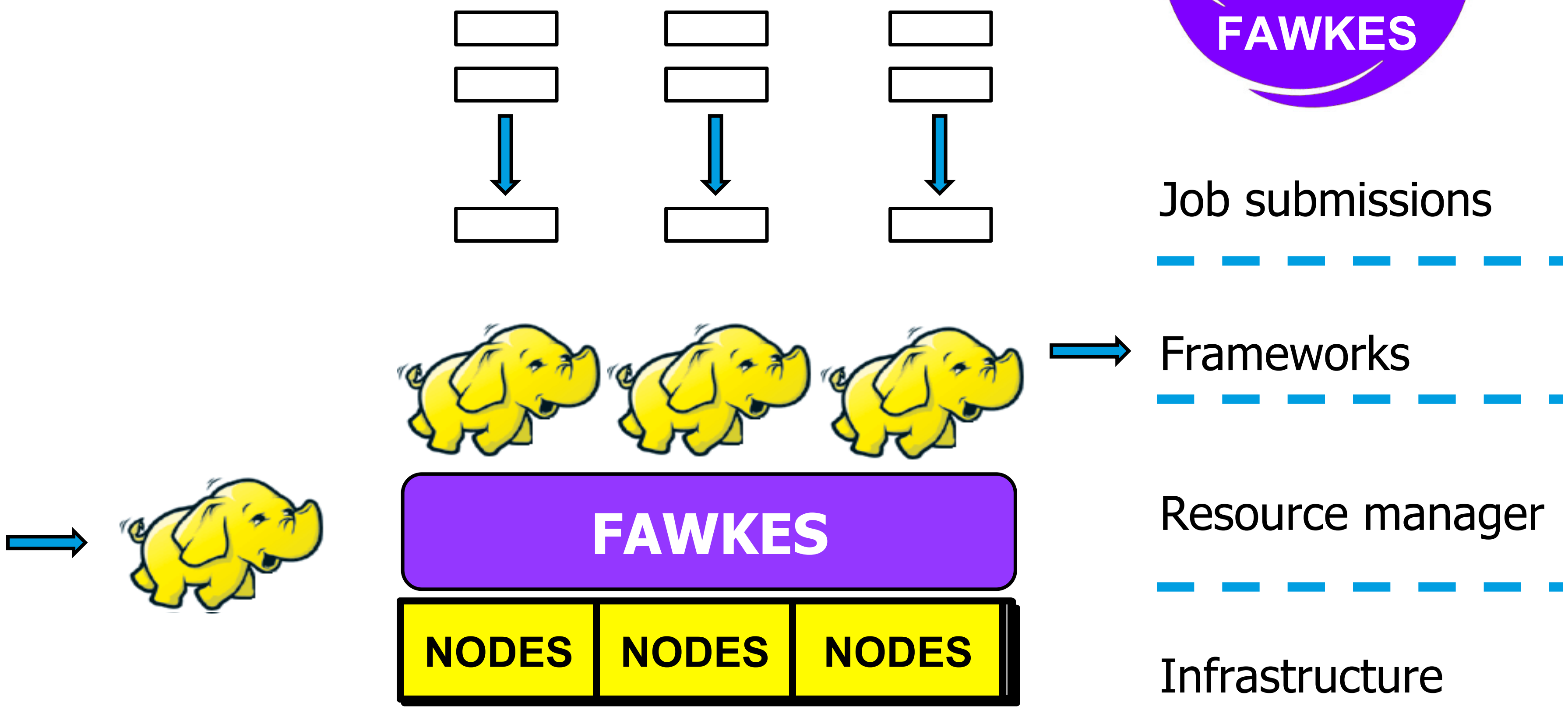
Big Data Enthusiast



Multiple frameworks = Isolation, especially performance

Our solution, FAWKES

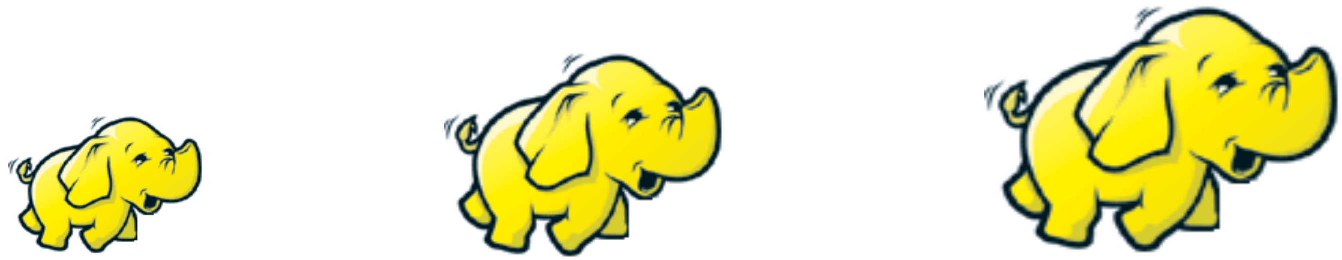
Two-level scheduling architecture



Why dynamic provisioning?

Because workloads may be time-varying:

- Poor resource utilization
- Imbalanced service levels



$w_1 < w_2 < w_3$

Framework size:

$$s_i = \frac{w_i}{w_1 + w_2 + w_3}, \quad i = 1,2,3$$

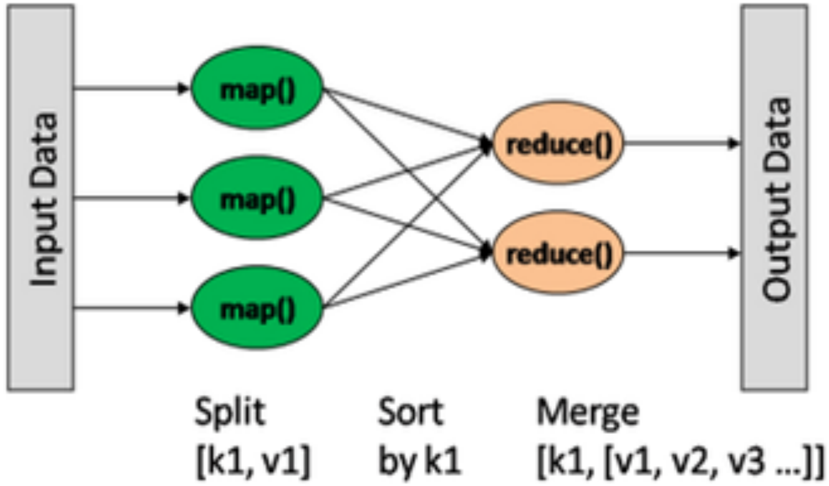
Roadmap

1. Introduction
2. **Dynamic MapReduce**
3. FAWKES operation
4. Experimental setup
5. Results and analysis
6. Conclusions

Dynamic MapReduce

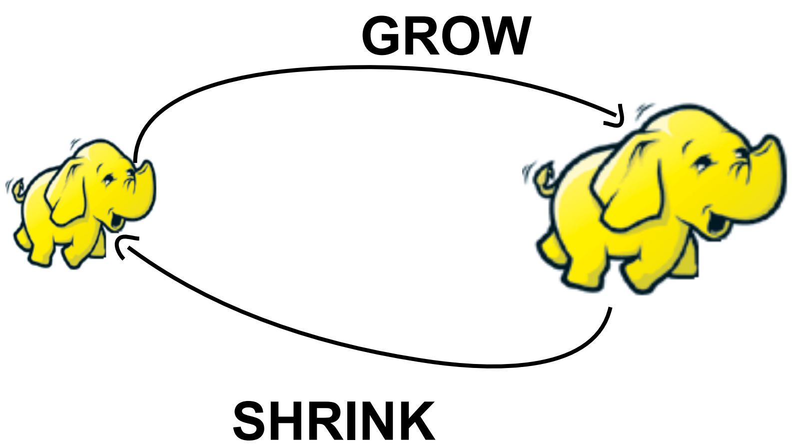
MapReduce framework

- Distributed file system
- Execution engine
- Data locality constraints

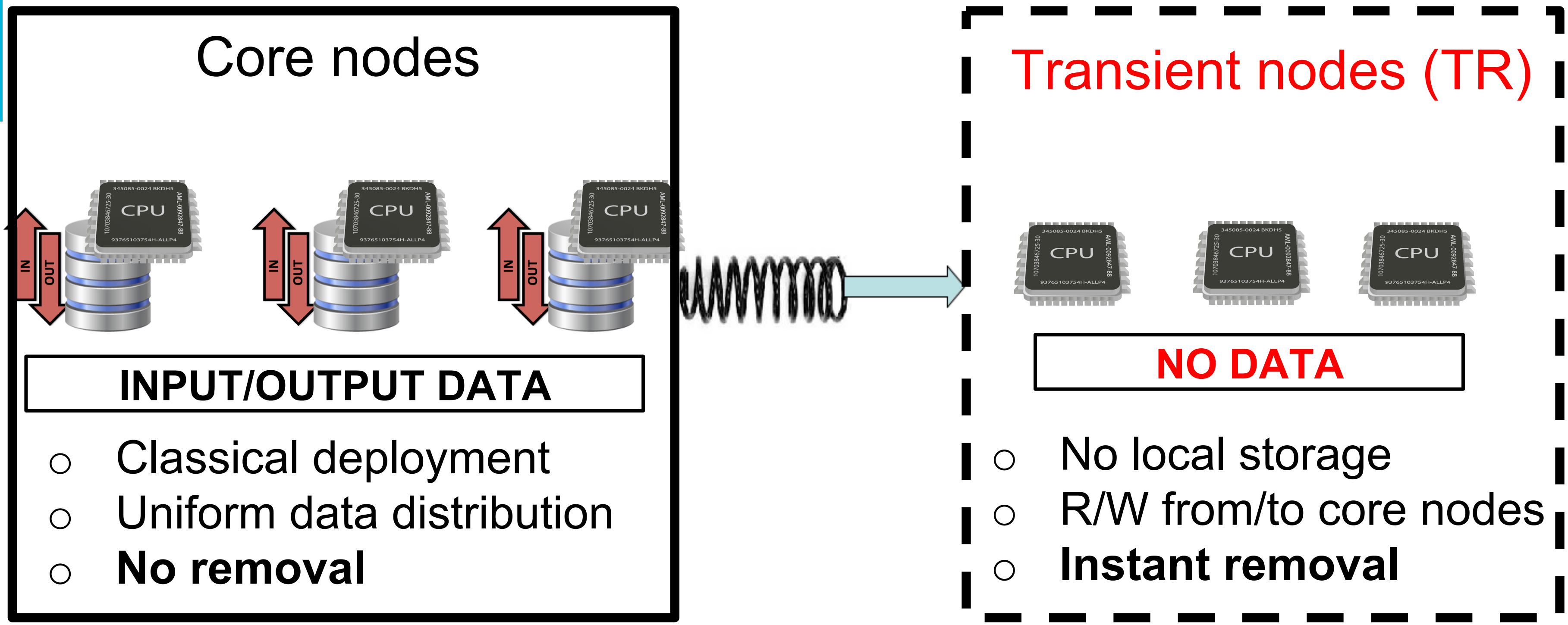


Grow and shrink MapReduce

- Reliable data management
- Fast reconfiguration
- Break data locality



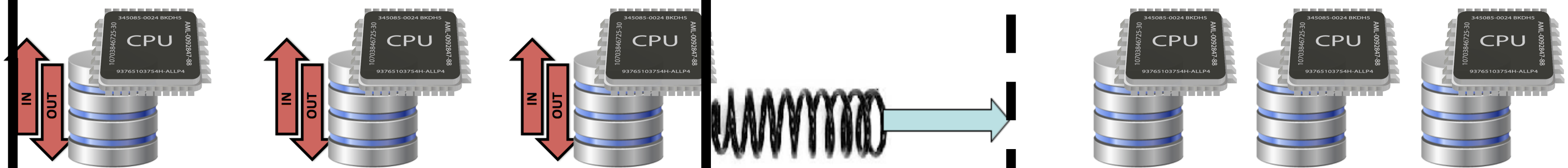
No data locality



Performance?

Relaxed data locality

Core nodes



INPUT/OUTPUT DATA

- Classical deployment
- Uniform data distribution
- **No removal**

Trans-core nodes (TC)

OUTPUT DATA

- Local storage, no input
- Only R from core nodes
- **Delayed removal**

Better performance?

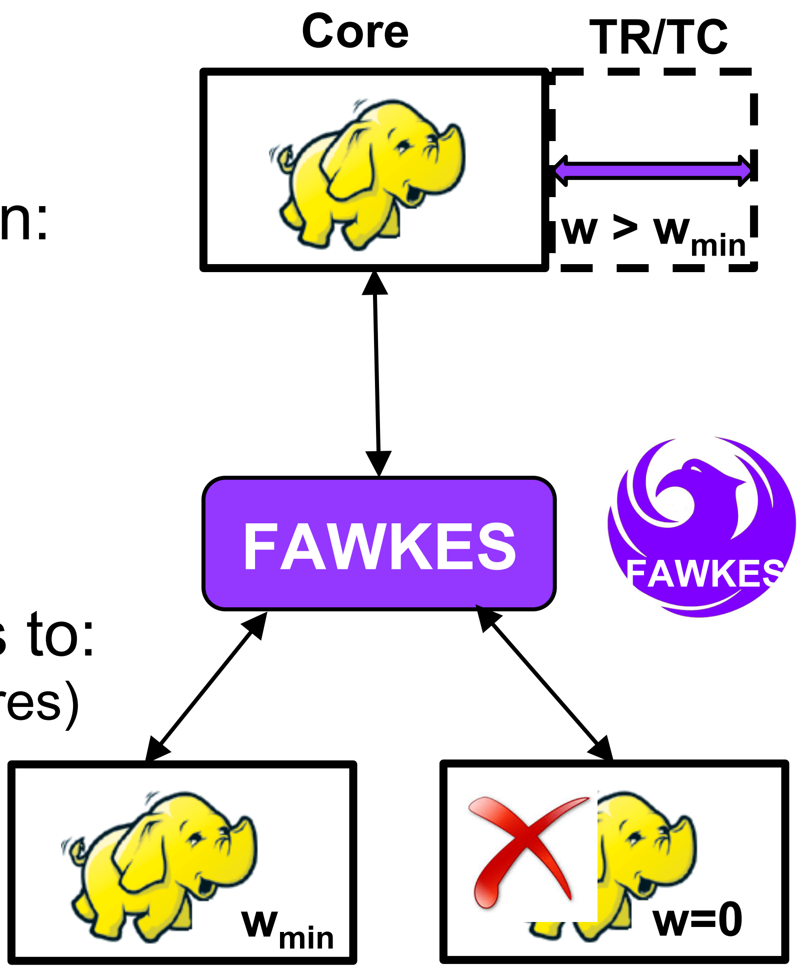
Roadmap

1. Introduction
2. Dynamic MapReduce
3. **FAWKES operation**
4. Experimental setup
5. Results and analysis
6. Conclusions

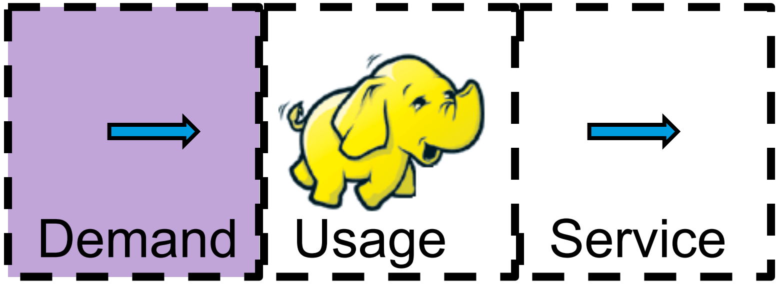
FAWKES in a nutshell

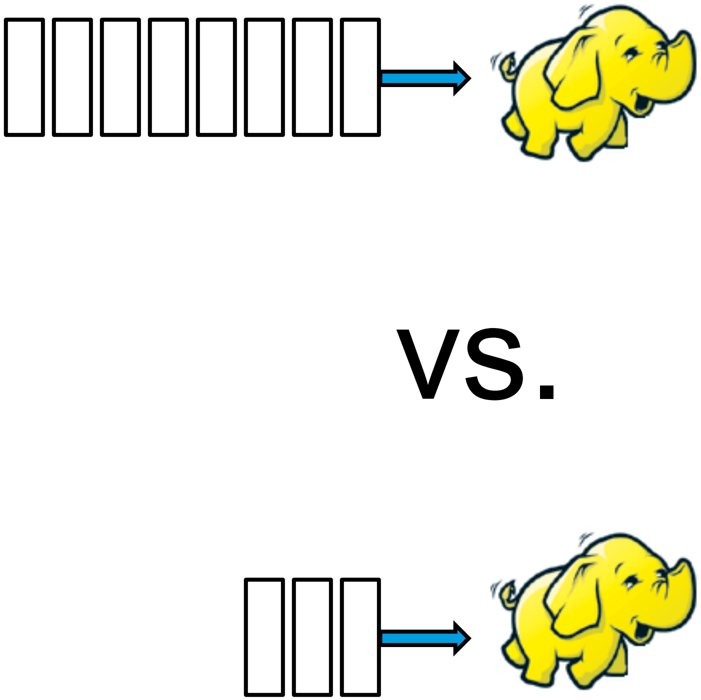
1. Updates dynamic weights when:
   - New frameworks arrive
   - Framework states change

2. Shrinks and grows frameworks to:
   - Allocate new frameworks (min. shares)
   - Give fair shares to existing ones



How to differentiate frameworks? (1/3)

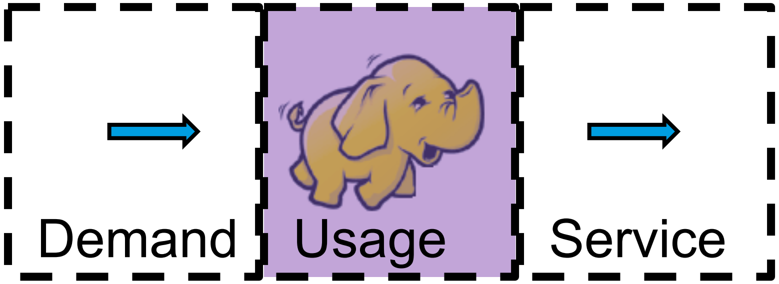




By demand – 3 policies:

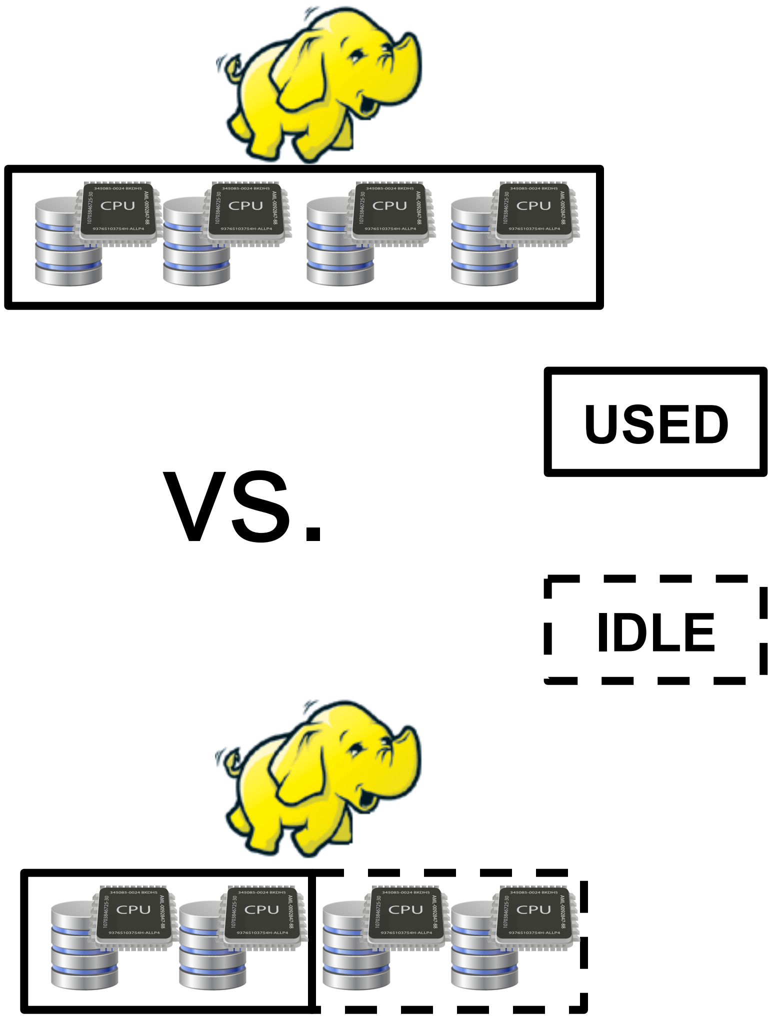
- Job Demand (JD)
- Data Demand (DD)
- Task Demand (TD)

How to differentiate frameworks? (2/3)

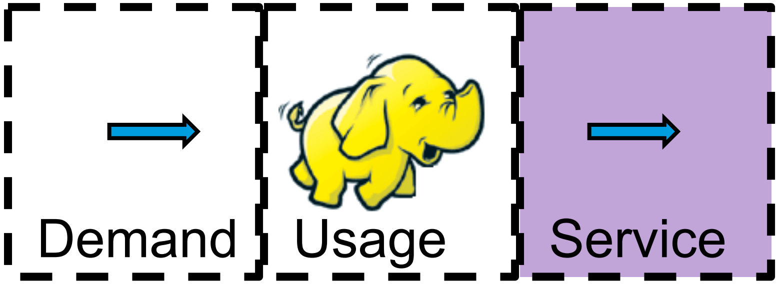


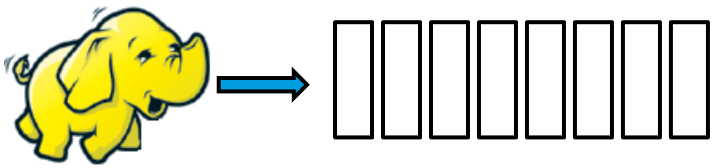
By usage – 3 policies:

- Processor Usage (PU)
- Disk Usage (DU)
- Resource Usage (RU)

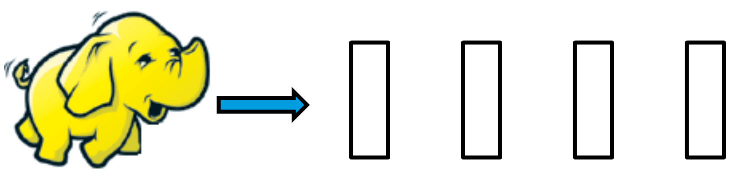


How to differentiate frameworks? (3/3)





vs.



By service – 3 policies:

- Job Slowdown (JS)
- Job Throughput (JT)
- Task Throughput (TT)

Roadmap

1. Introduction
2. Dynamic MapReduce
3. FAWKES operation
4. **Experimental setup**
5. Results and analysis
6. Conclusions

Experimental setup

Distributed ASCI Supercomputer - Version 4

DAS-4 multicluster system:

- 200 dual-quad-core compute nodes
- 24 GB memory per node
- 150 TB total storage
- 20 Gbps InfiniBand

Hadoop deployment:

- Hadoop-1.0 over InfiniBand
- 6 map + 2 reduce slots per node
- 128 MB block size

Overview of experiments:

- Most experiments on 20 nodes
- Up to 60 working nodes
- More than 3 months system time

MapReduce applications

Application	Type	Input	Output
Wordcount (WC)	CPU	200 GB	5.5 MB
Sort (ST)	Disk	200 GB	200 GB
PageRank (PR)	CPU	50 GB	1.5 MB
K-Means (KM)	Both	70 GB	72 GB
TrackerOverTime (TT)	CPU	100 GB	3.9 MB
ActiveHashes (AH)	Both	100 GB	90 KB
BTWorld (BT)	Both	100 GB	73 GB

Synthetic benchmarks:
- HiBench suite
- Single applications
- Random datasets

Real-world applications:
- BTWorld workflow
- 14 Pig queries
- BitTorrent monitoring data

Ghit, Capota, Hegeman, Hidders, Iosup, Epema, "The Challenge of Scaling Complex Big Data Workflows", CCGrid 2014. SCALE Challenge Winner.

Roadmap

1. Introduction
2. Dynamic MapReduce
3. FAWKES operation
4. Experimental setup
5. **Results and analysis**
6. Conclusions

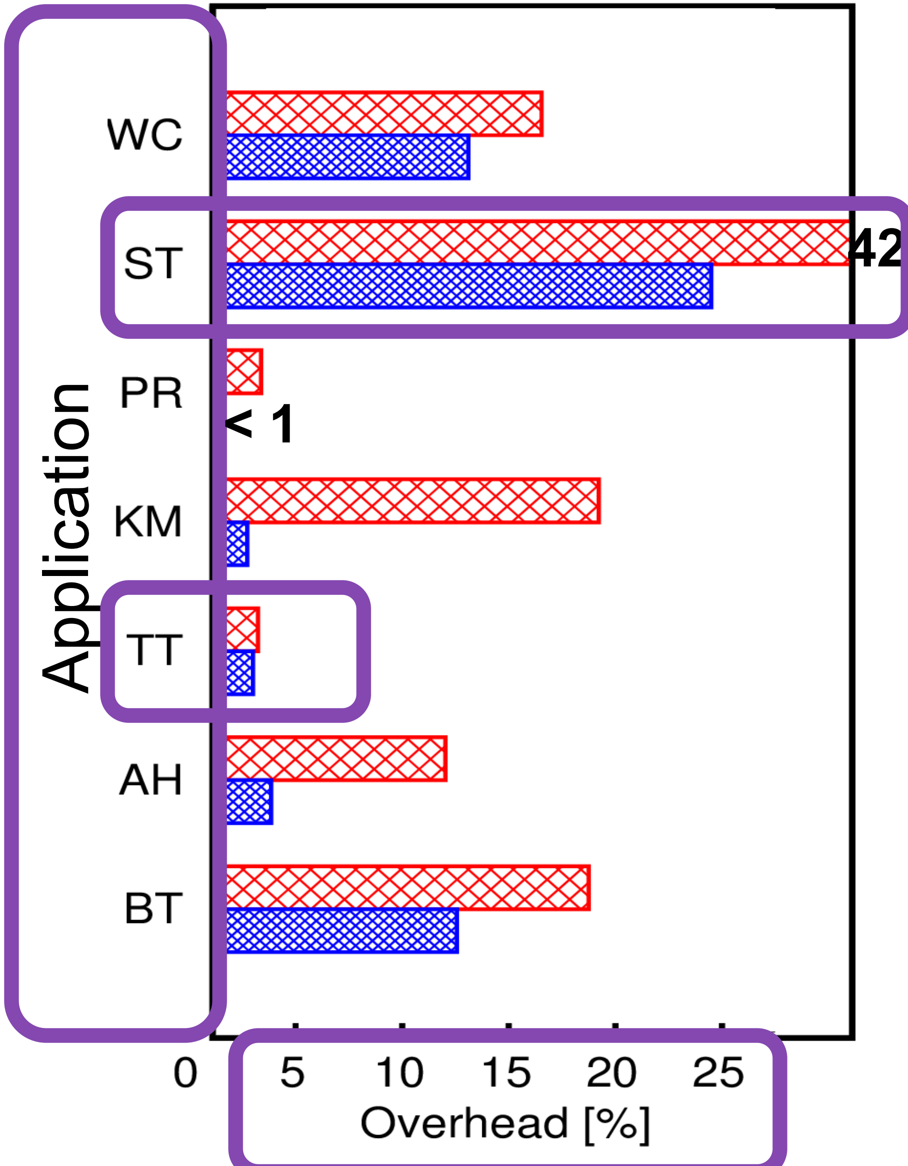
Performance of dynamic MapReduce

10 core + 10xTR
10 core + 10xTC
vs.
20 core nodes

TR - **good** for compute-intensive workloads.

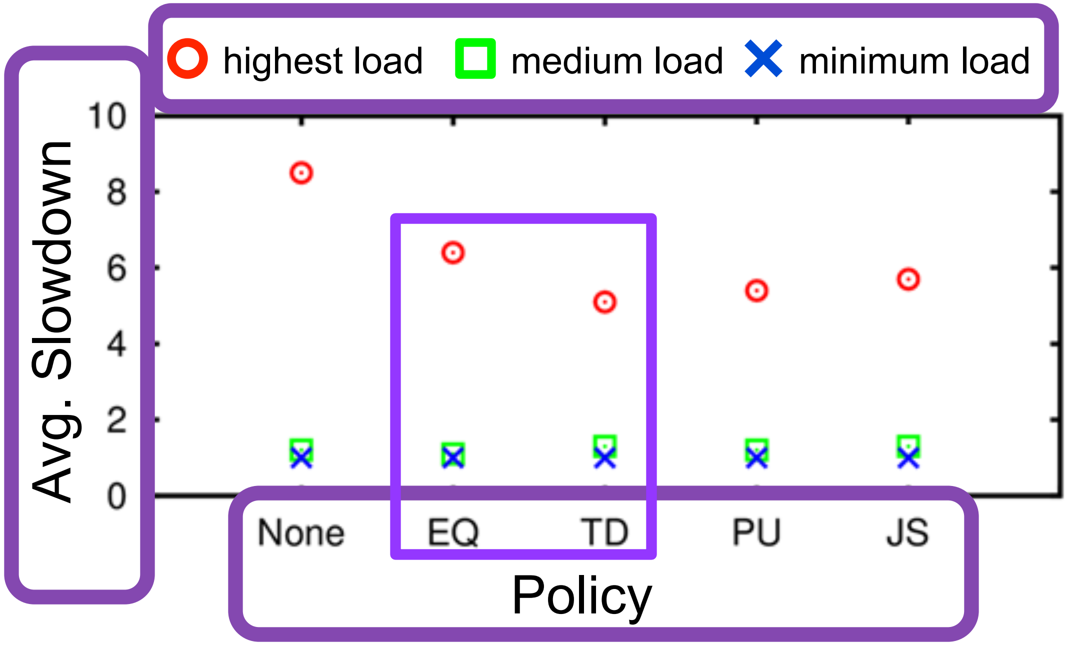
TC - **needed** for disk-intensive workloads.

Dynamic MapReduce:
< 25% overhead



Performance of FAWKES

Nodes	45
Frameworks	3
Min. shares	10
Datasets	300 GB
Jobs submitted	900

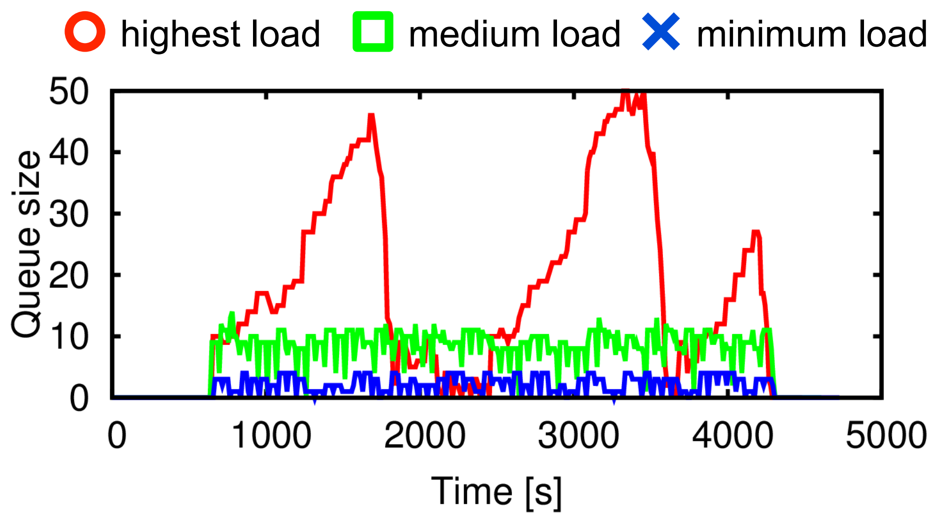


Up to 20% lower slowdown

None – Minimum shares
EQ – EQual shares
TD – Task Demand
PU – Processor Usage
JS – Job Slowdown

FAWKES: behind the scenes

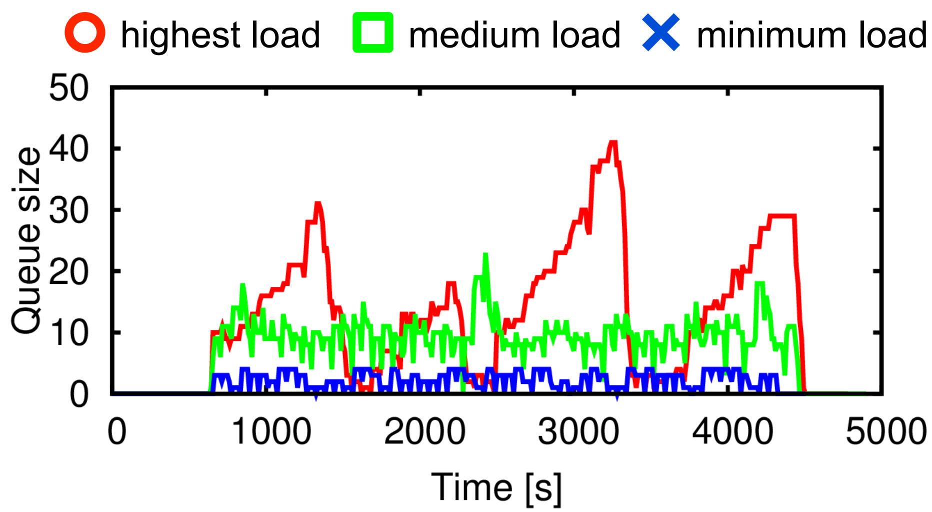
EQ



Utilizations: 60% / 23% / 5%

Imbalanced

TD



Utilizations: 50% / 30% / 8%

More balanced

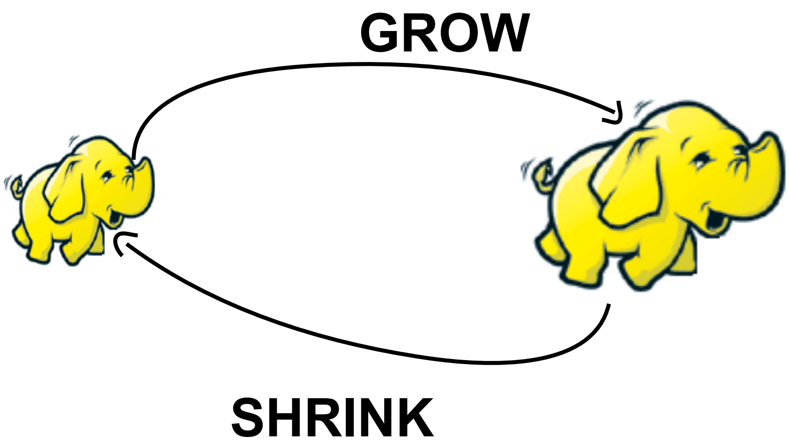
Roadmap

1. Introduction
2. Dynamic MapReduce
3. FAWKES operation
4. Experimental setup
5. Results and analysis
6. **Conclusions**

Take-home message

1. Dynamic MapReduce relaxes data locality
2. FAWKES reduces the imbalance between frameworks
3. More aggressive policies?

FAWKES



Our PDS group

Scheduling and resource management research:

- Systems: multi-cluster systems and clouds
- Applications: workflows, bags-of-tasks, data-intensive, etc.

Contact: B.I.Ghit@tudelft.nl

http://www.pds.ewi.tudelft.nl/ghit/
http://www.pds.ewi.tudelft.nl/research-publications/publications/

Backup slides

Related work

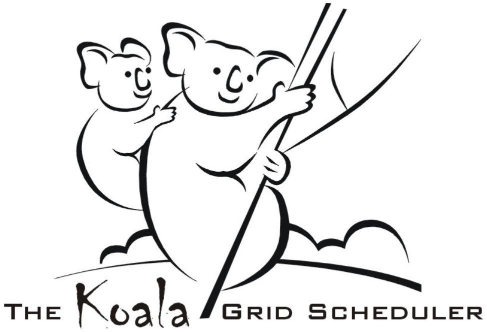
- Resource requests from applications
- Capacity and Fair schedulers

FAWKES uses feedback from system operation

MESOS

- Resource offers to frameworks
- Optimizes for data locality

FAWKES schedules frameworks automatically



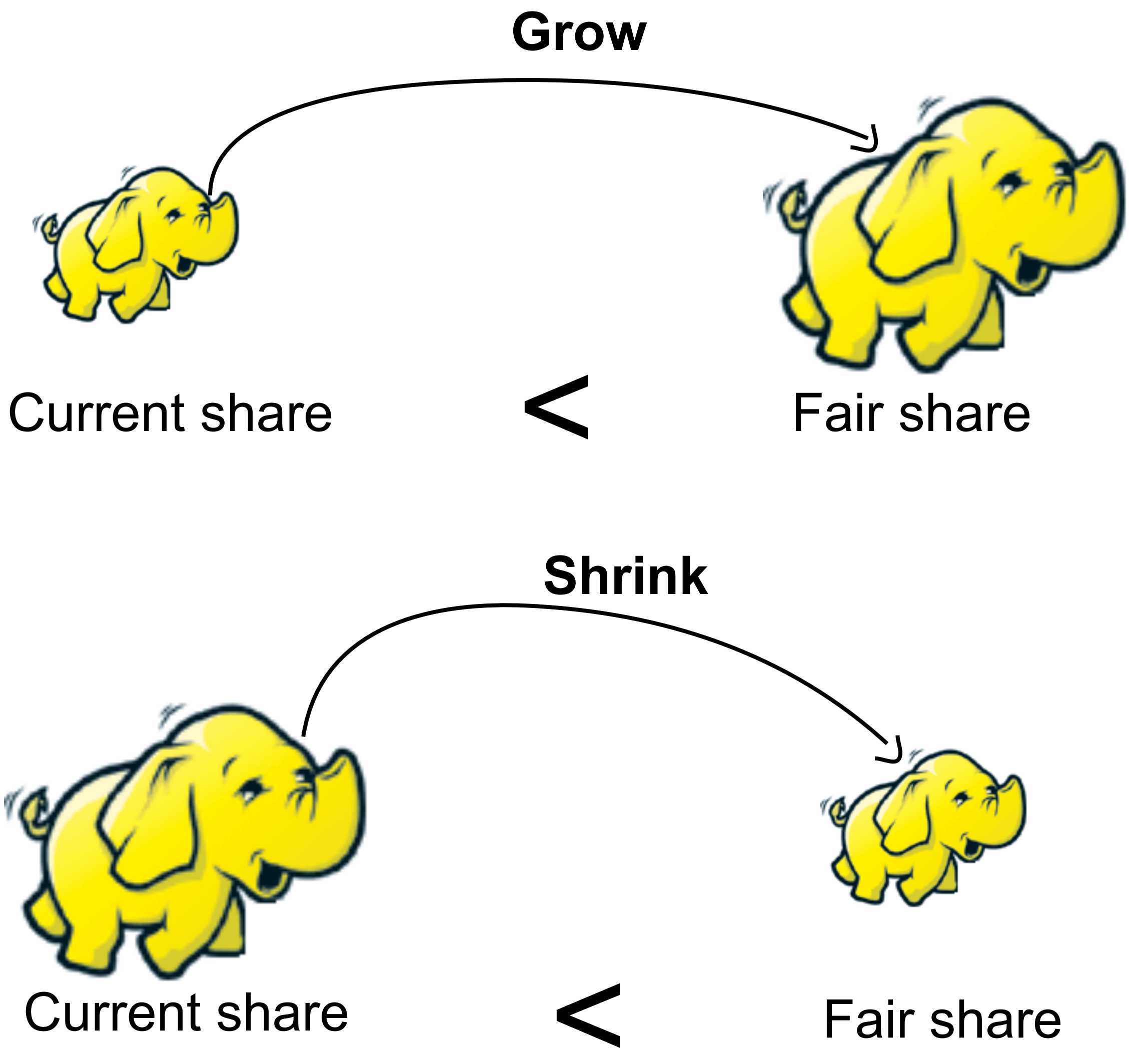
- Grid and cloud scheduler @ TU Delft
- Single applications and frameworks

FAWKES is a research prototype

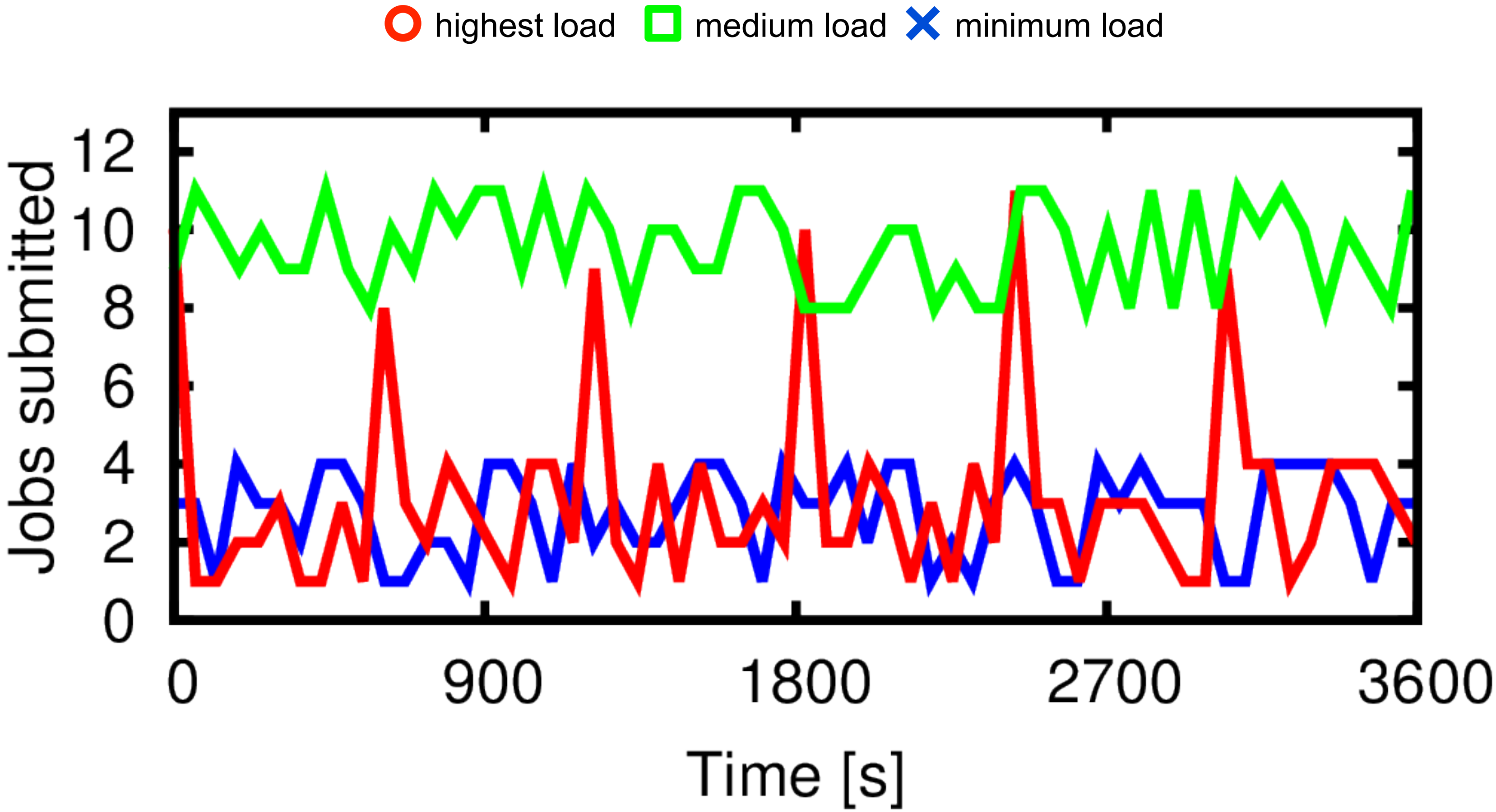
The grow-shrink mechanism

Negative discrimination

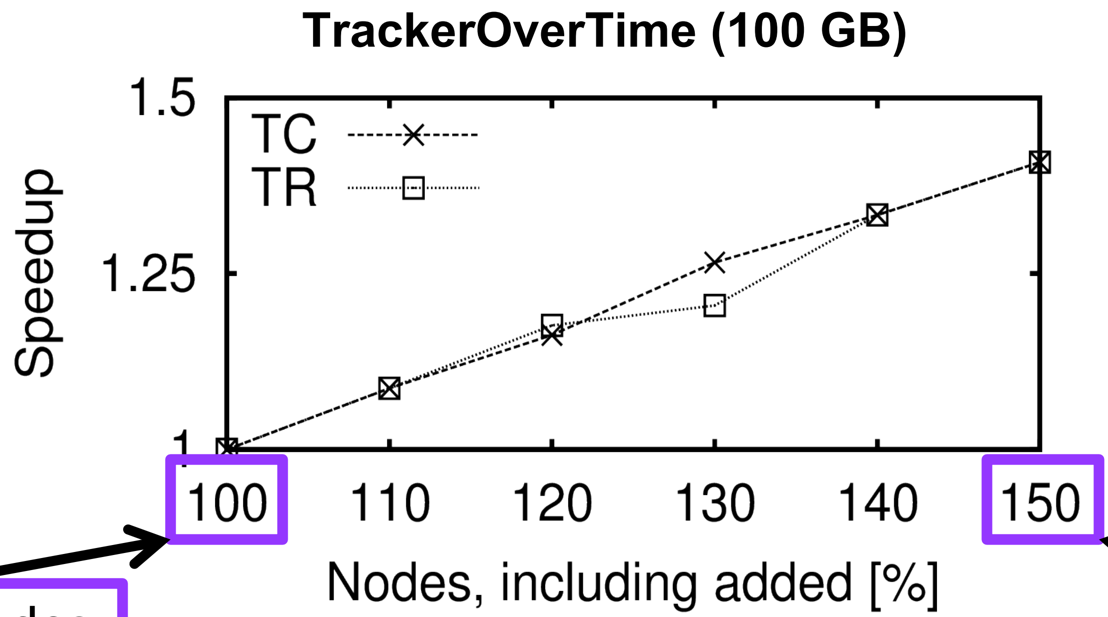
Positive discrimination



Submission patterns



Speedup when growing (1/2)

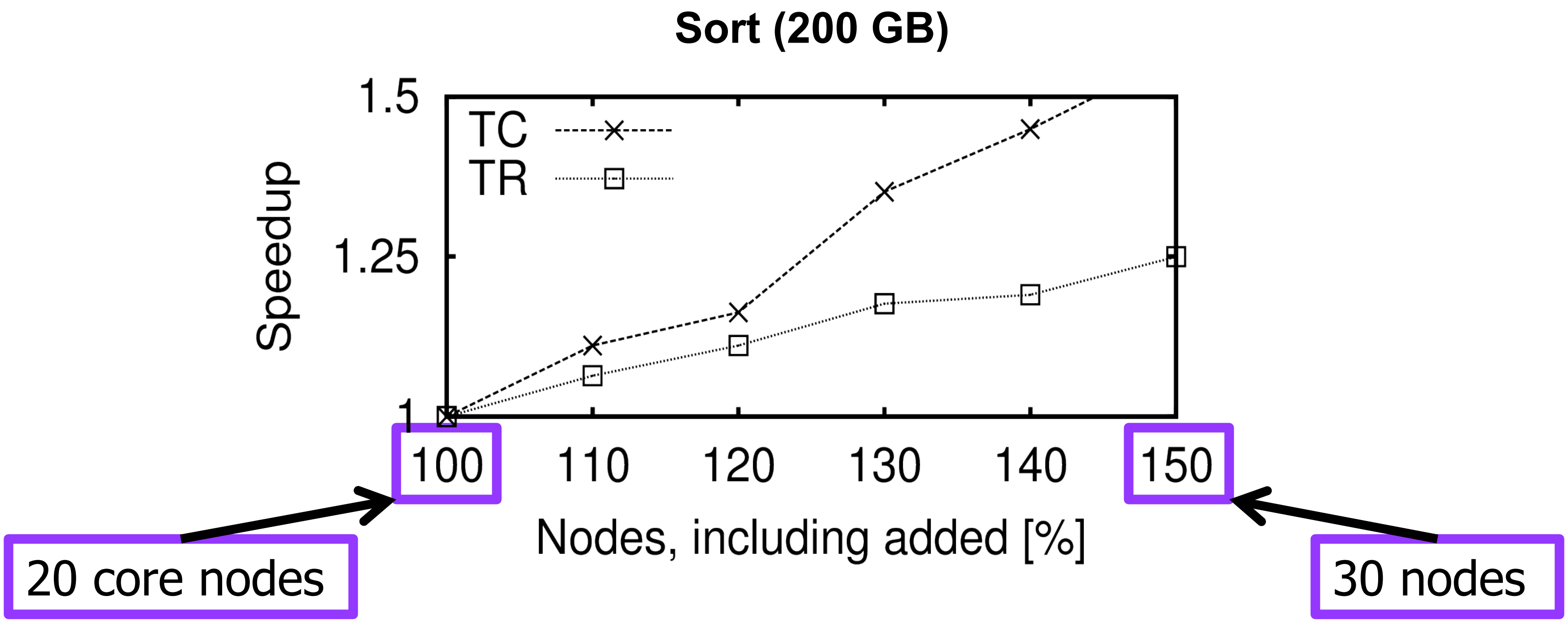


20 core nodes

30 nodes

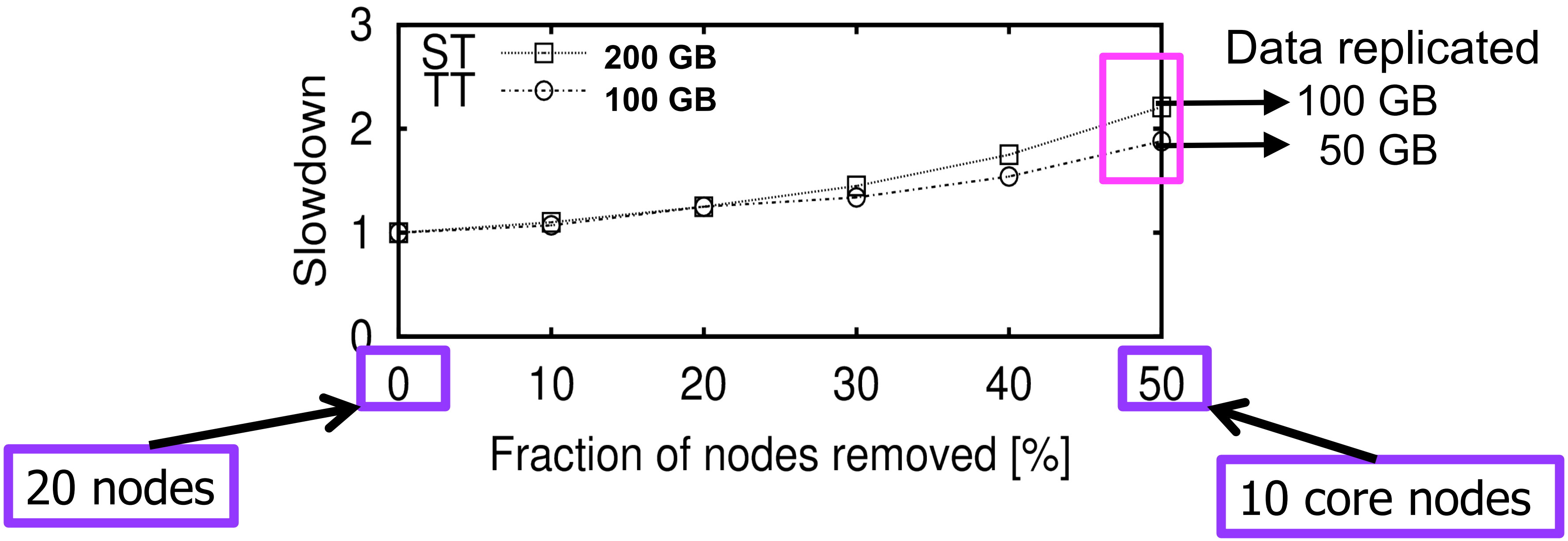
TR nodes deliver good performance for CPU bound workloads

Speedup when growing (2/2)



(Only) TC nodes deliver good performance for disk-bound workloads

Slowdown when shrinking



Job slowdown increases linearly with the amount of replicated data