Achieving Fairness and High-Performance in Datacenter Scheduling

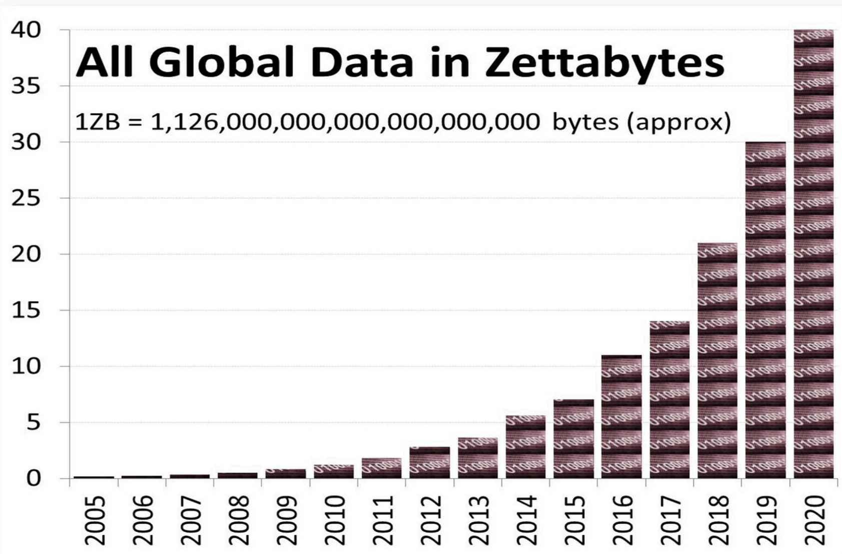
Bogdan Ghit

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

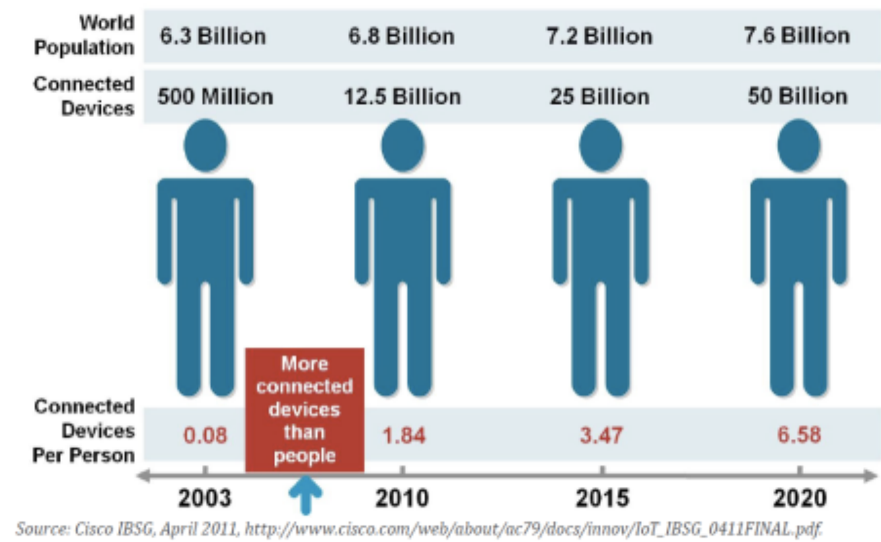
TU Delft

Research context

Growing volumes of data and users.



From UNECE Statistics



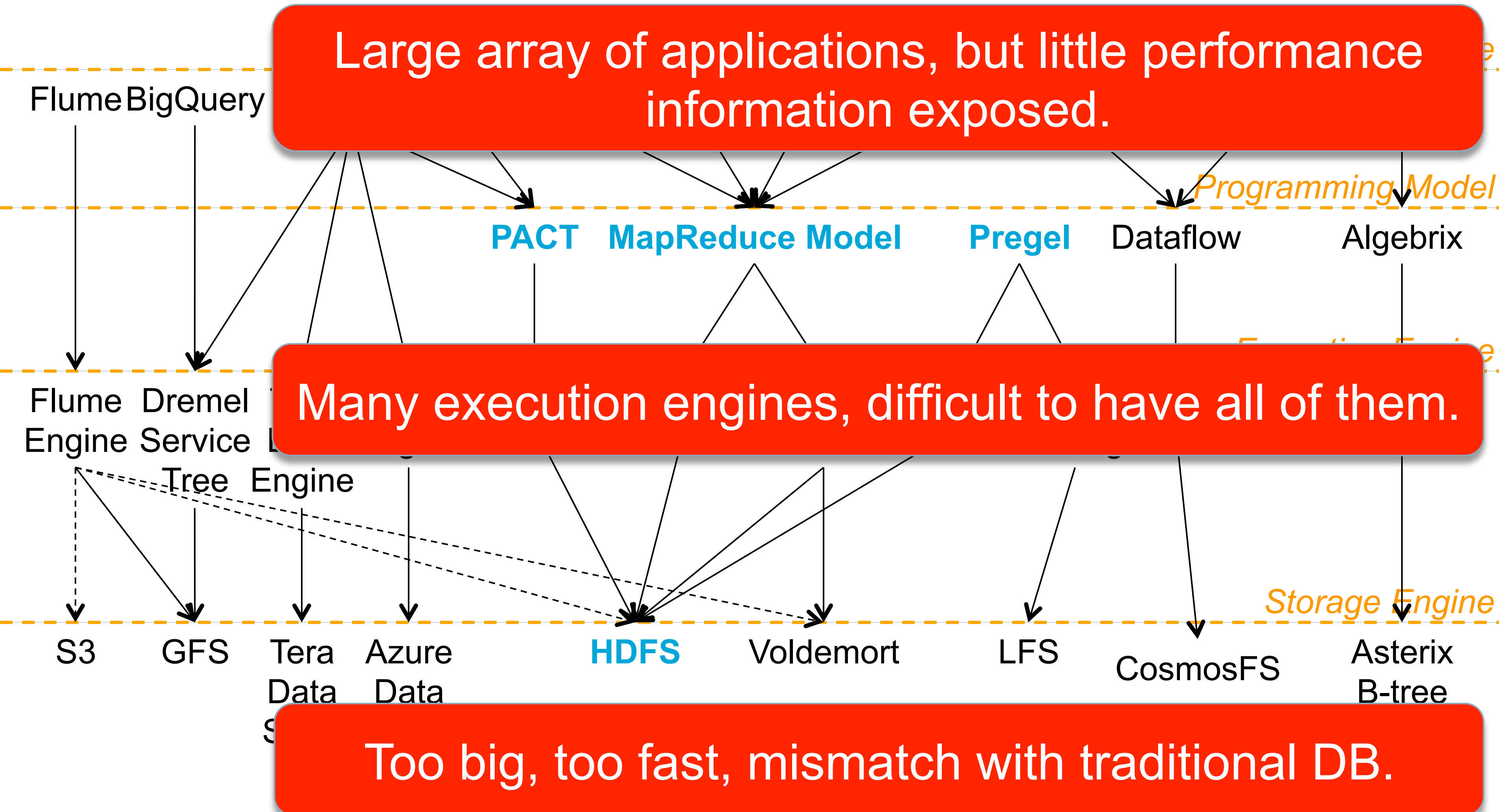
From Cisco IBSG

Applications run on clusters of thousands of nodes:

- Web search
- Social networks
- Apple's Siri

TU Delft

What is big data?



Large array of applications, but little performance information exposed.

Many execution engines, difficult to have all of them.

Too big, too fast, mismatch with traditional DB.

Adapted from: Dagstuhl Seminar on Information Management in the Cloud, http://www.dagstuhl.de/program/calendar/partlist/?semnr=11321&SUOG

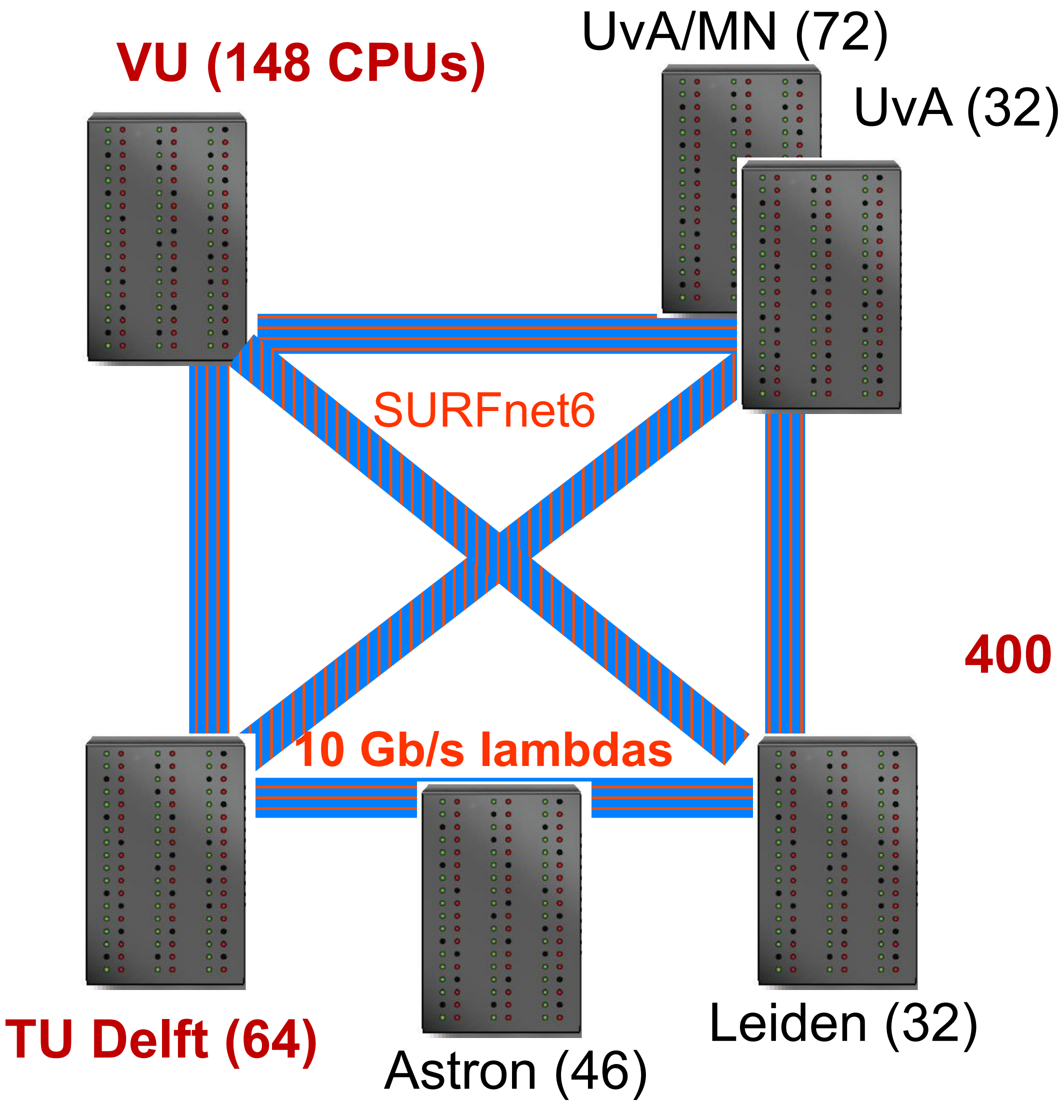
In this talk

(1) Designing Fawkes, a scheduling system for dynamic (re-)allocation of the datacenter resources to multiple (groups of) users.

(2) Analyzing fundamental scheduling problems in datacenters: performance isolation, resource partitioning, fairness.

(3) Designing Tyrex, a scheduling system that reduces the job slowdown variability in data-intensive workloads.

The experimental testbed: DAS



- 10+ years of system research
- 300+ scientists as users

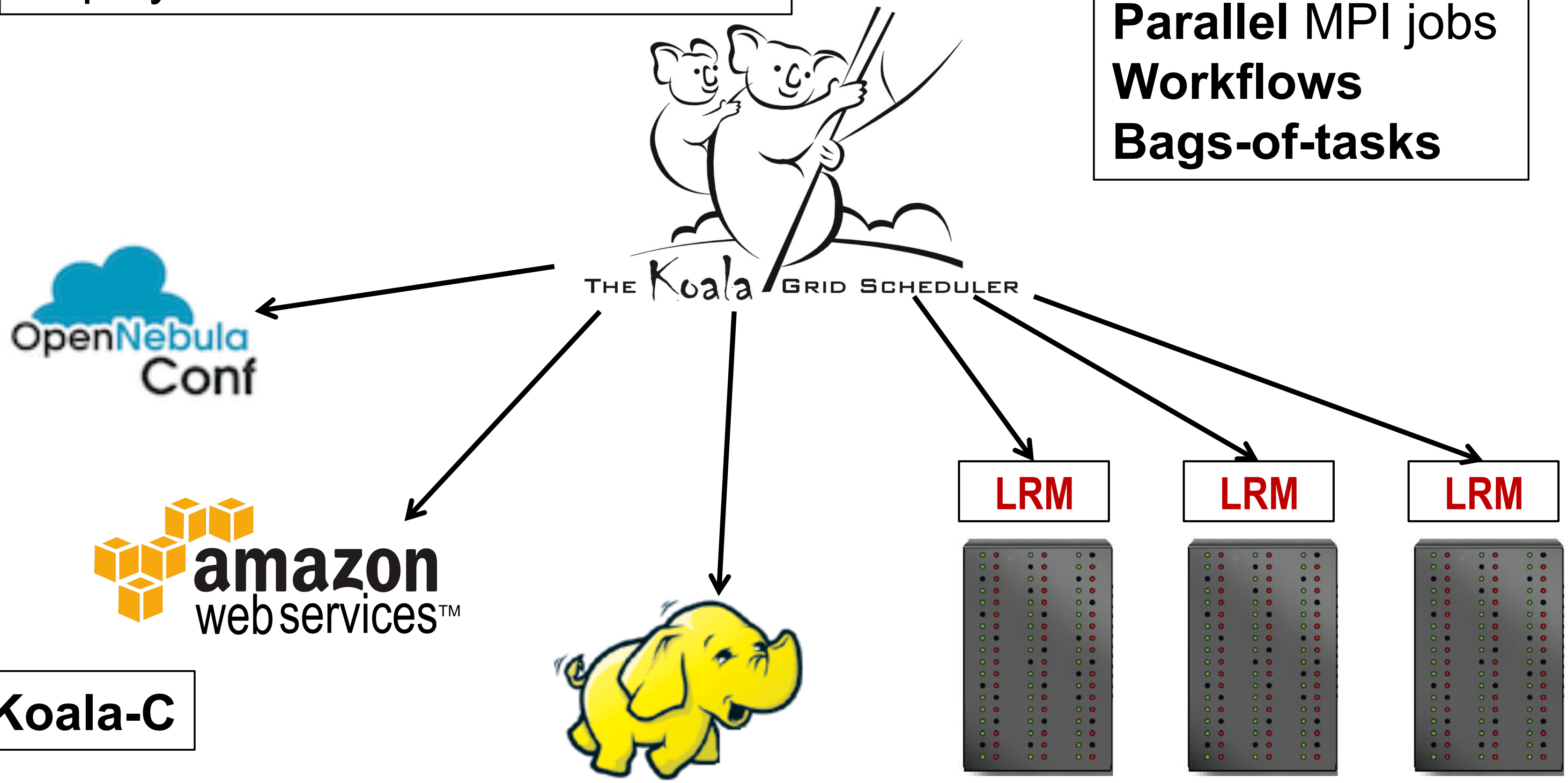
Distributed ASCI Supercomputer 5

400 2~~00~~ dual-quad-core compute nodes
24 GB memory per node
150 TB total storage
20 Gpbs Q~~D~~R InfiniBand network
FDR

The KOALA multicluster scheduler

Our research vehicle
Deployed on DAS since 2005

Parallel MPI jobs
Workflows
Bags-of-tasks



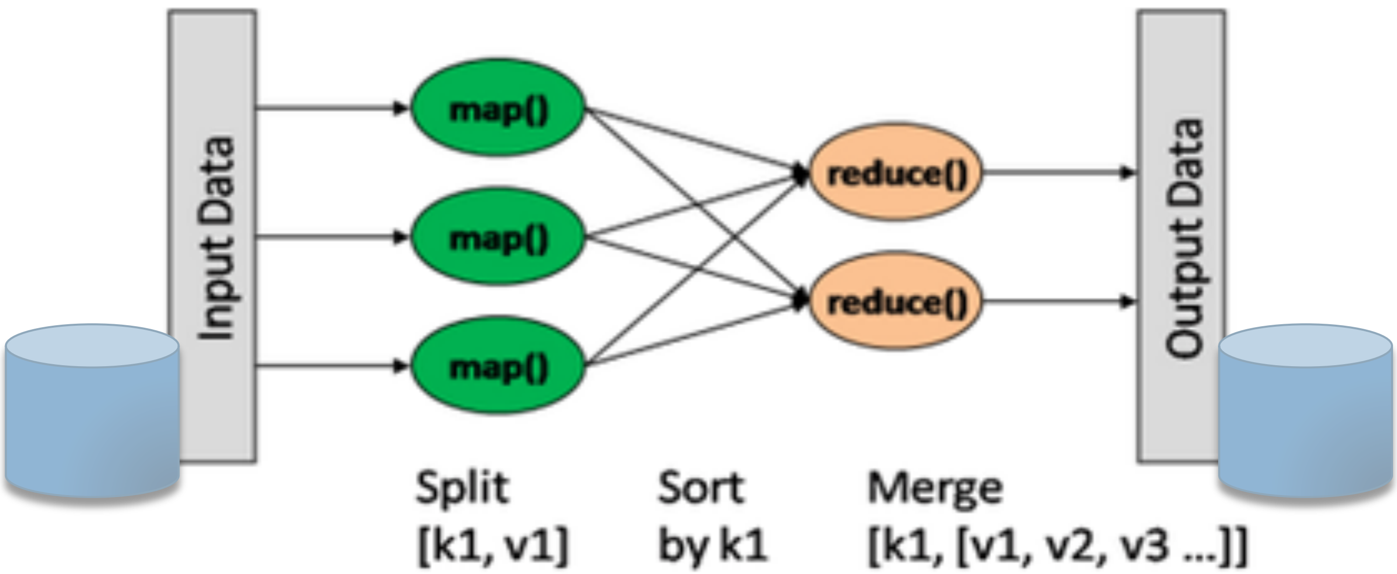
Koala-C

This talk: MapReduce frameworks

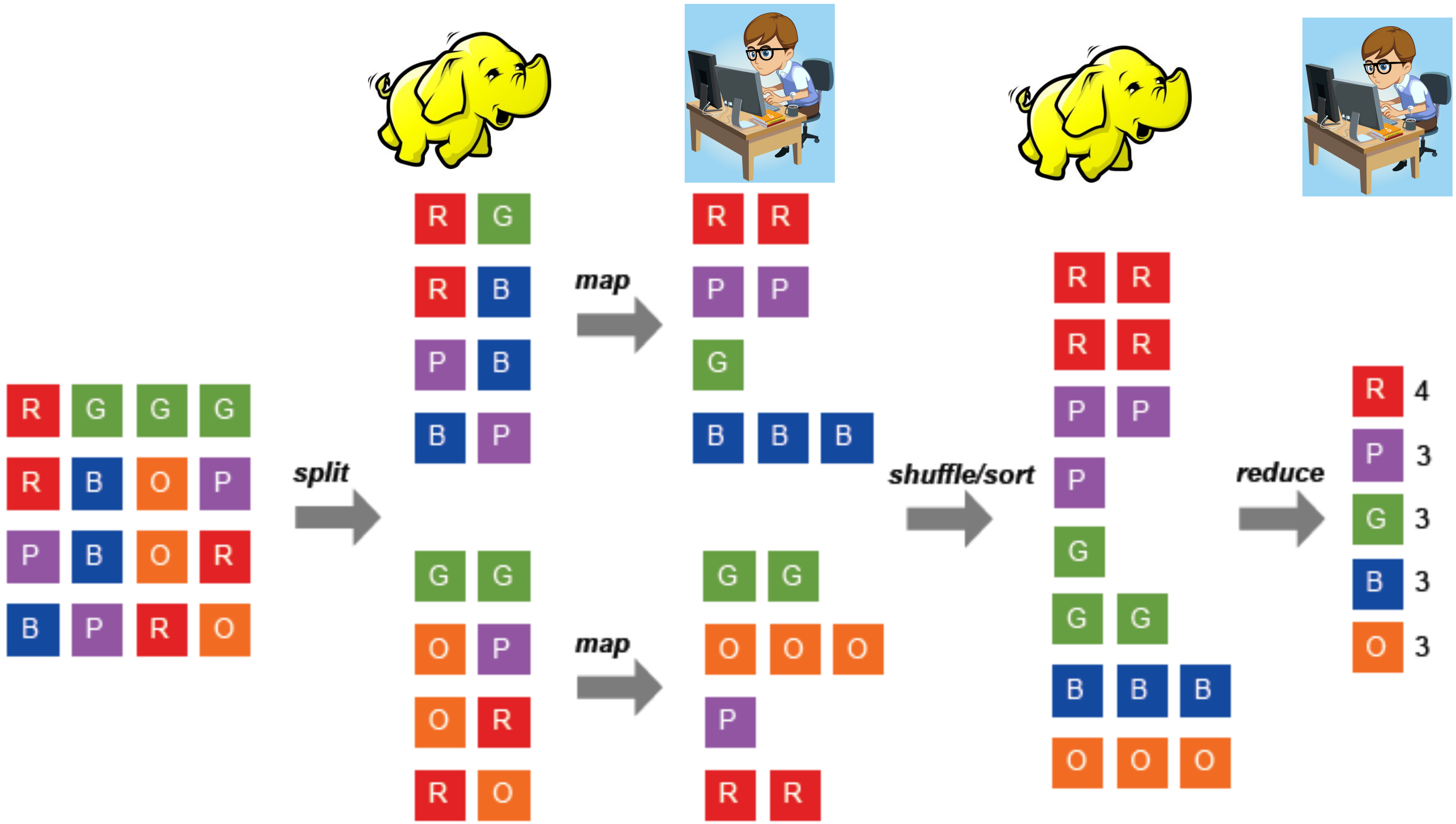
The MapReduce framework

Programming model

- Transforms data flowing from stable storage to stable storage.
- Jobs are split into tasks that run on slots.



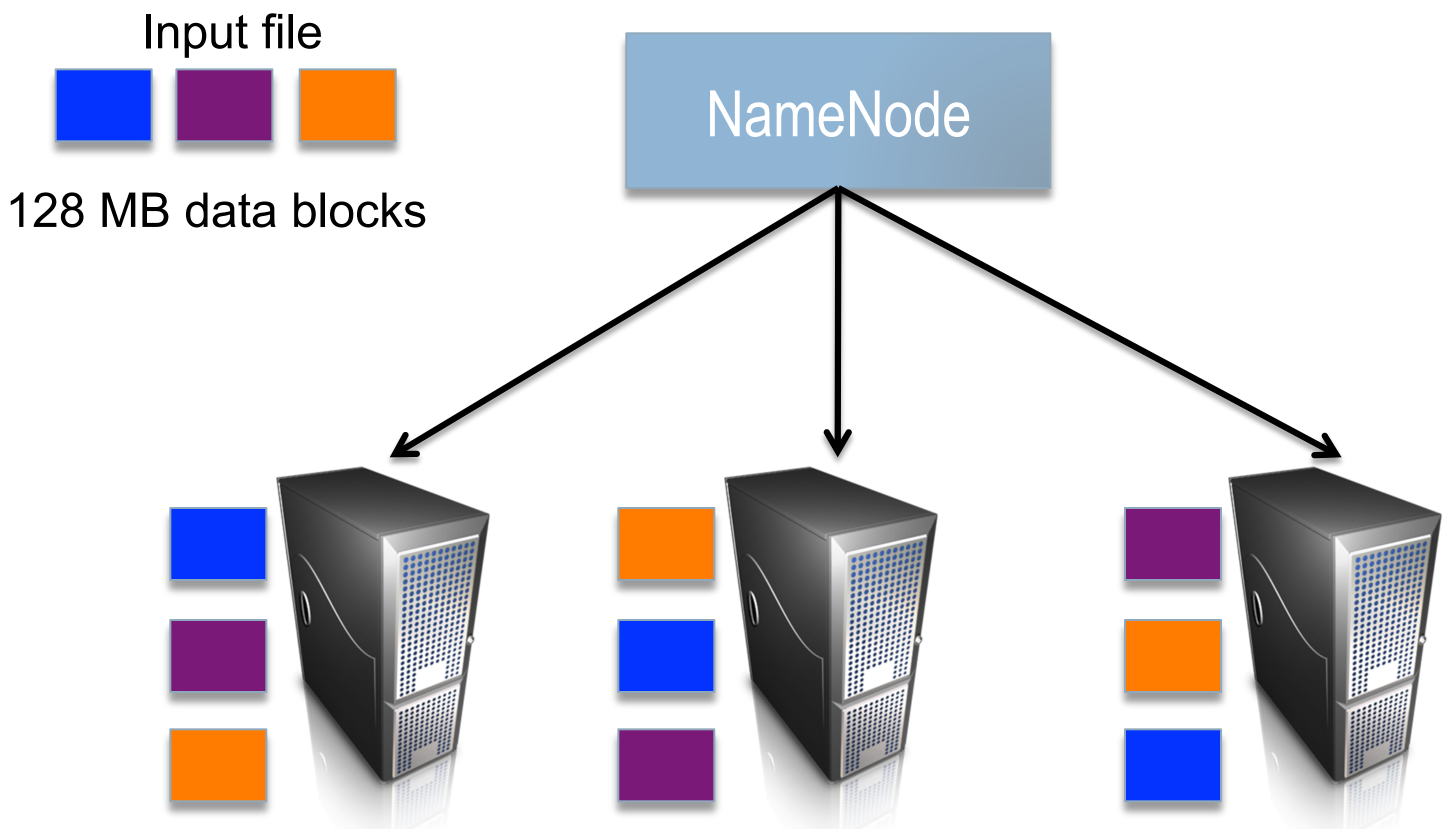
MapReduce explained



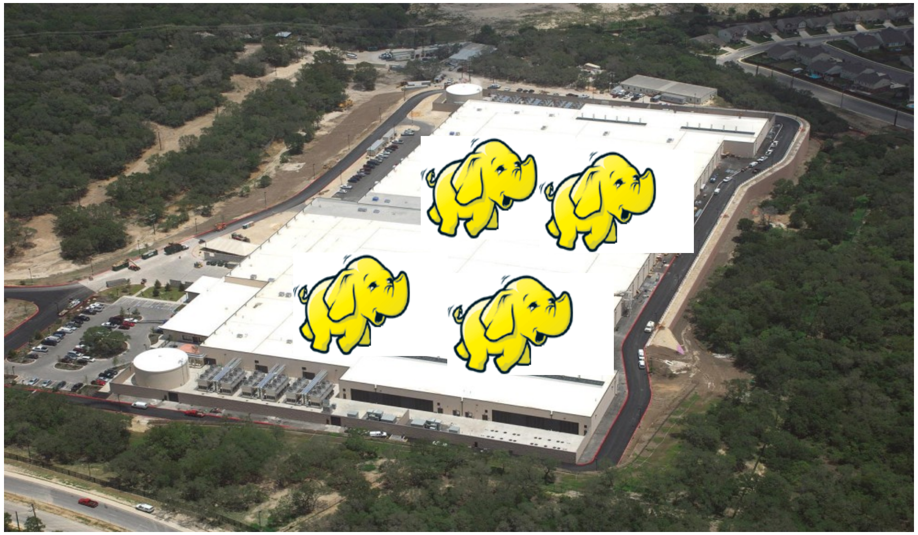
Inside the elephant: the HDFS

Traditional assumptions and goals:

- "HDFS apps. need a write-once-read-many access model for files."
- "Hardware failure is the norm rather than the exception."
- "Moving computation is cheaper than moving data."



Multiple users need multiple frameworks



Data isolation
Failure isolation
Version isolation

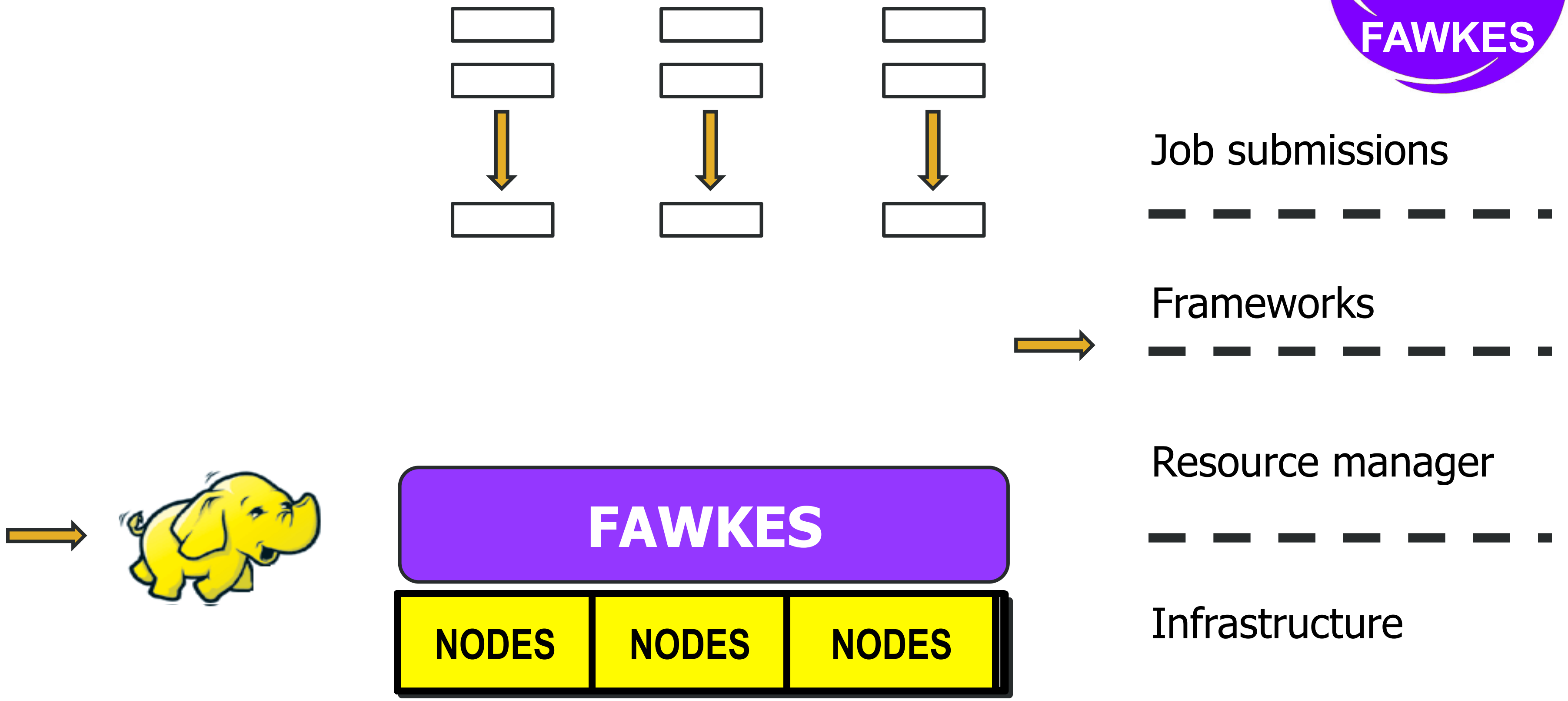
Performance isolation

- Appealing to companies and users
- Difficult to achieve and define
- No one framework optimal for any user
- Dynamic infrastructure for data processing

TU Delft

Dynamic Big Data Processing

Fawkes = elastic MapReduce via two-level scheduling

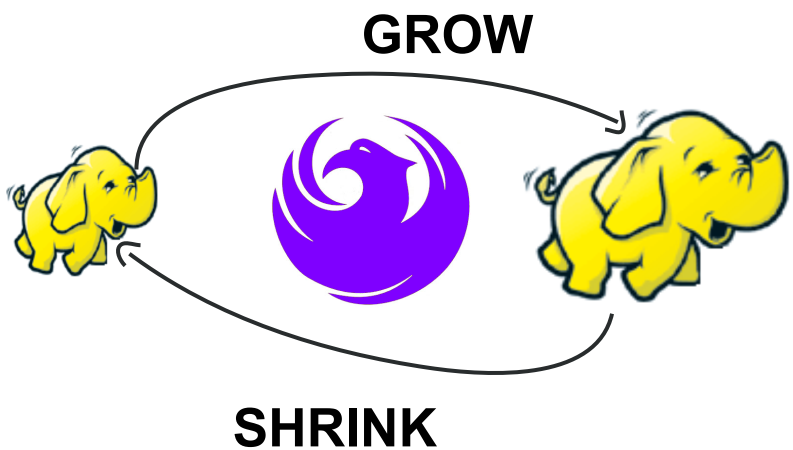


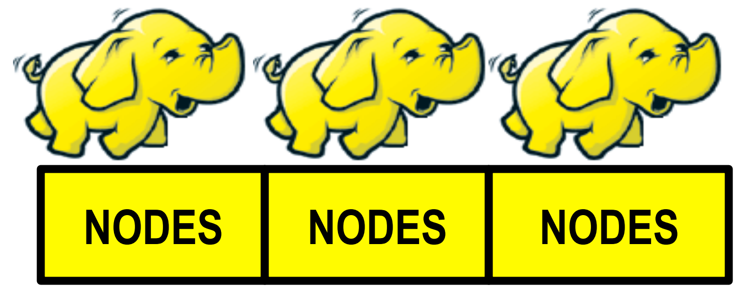
B.I. Ghit, N. Yigitbasi, A. Iosup, D.H.J. Epema, “Balanced Resource Allocations across Multiple Dynamic MapReduce Clusters”, ACM Sigmetrics 2014.

Elastic MapReduce

Because workloads may be time-varying:

- Poor resource utilization
- Imbalanced service levels





Growing and shrinking MapReduce:

- Distributed file system
- Execution engine
- Data locality constraints

TU Delft

How hard it really is?

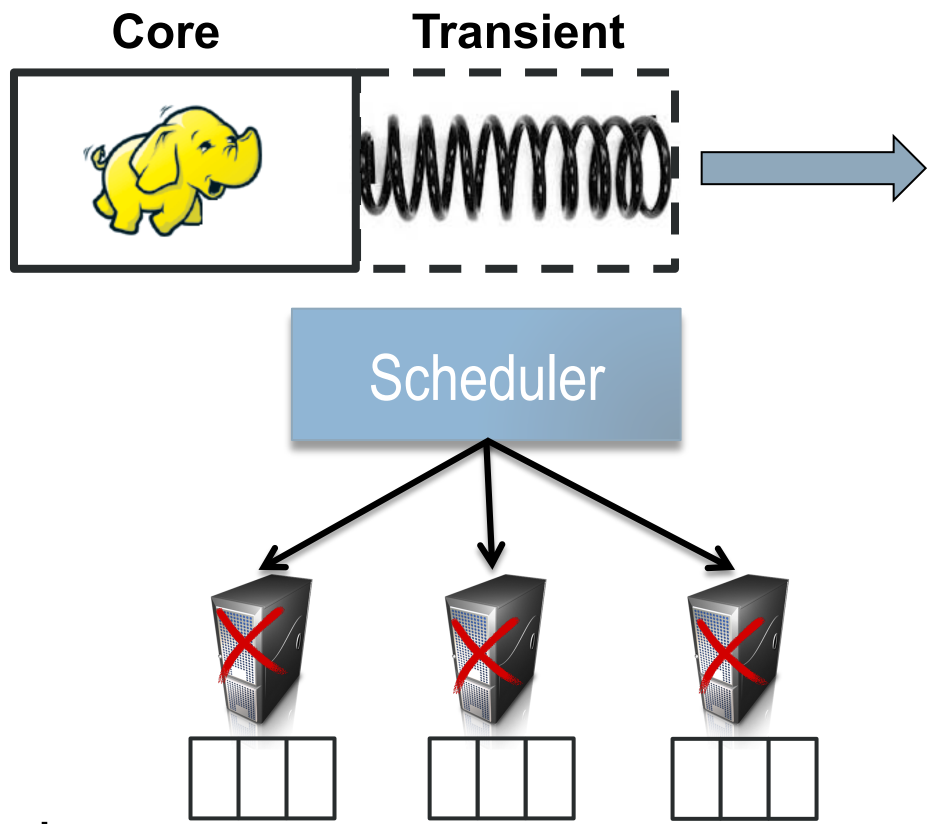
1. Distributed file system

- Big data is hard to move
- We need a fixed core extended

by transient nodes (*data locality*)

2. Execution engine

- Re-scheduling killed tasks
- We need to control the frequency of reconfigurations (*policies*)

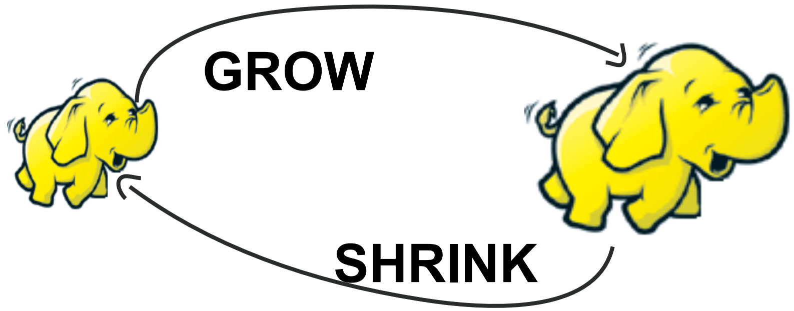


Growing and shrinking MapReduce:

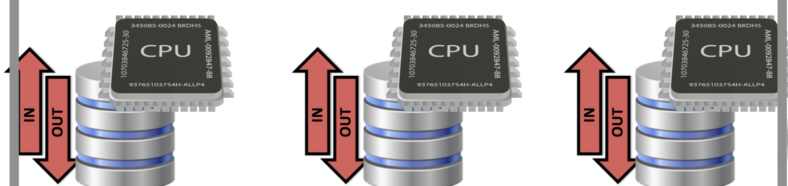
(1) Break data locality

(2) Policies to differentiate frameworks

Resizing MapReduce: no data locality

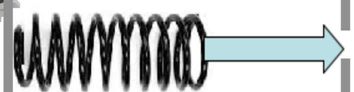


Core nodes

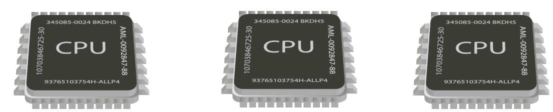


INPUT/OUTPUT DATA

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



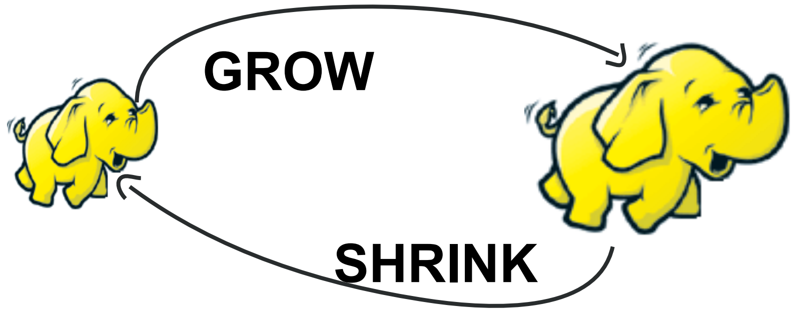
Transient nodes (TR)



NO DATA

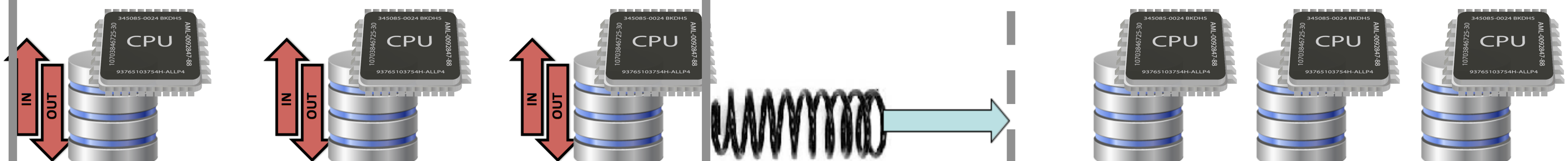
- No local storage
- R/W from/to core nodes
- **Instant removal**

Resizing MapReduce: relaxed data locality



Core nodes

Trans-core nodes (TC)



INPUT/OUTPUT DATA

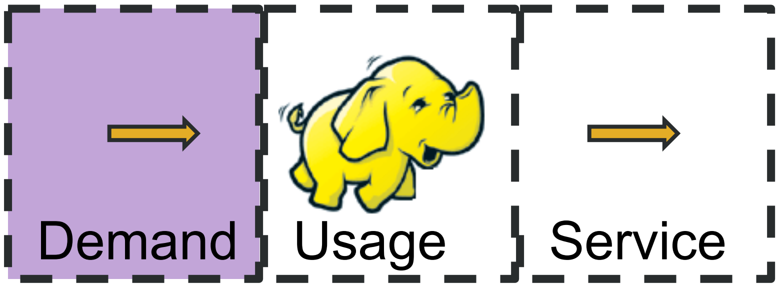
- Classical deployment
- Uniform data distribution
- **No removal**

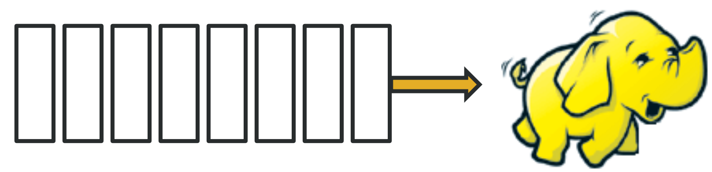
OUTPUT DATA

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

TUDelft

How to differentiate frameworks (1/3)

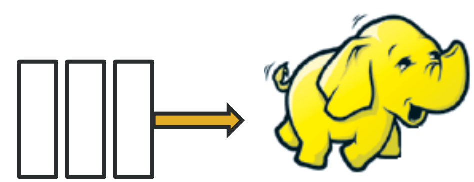




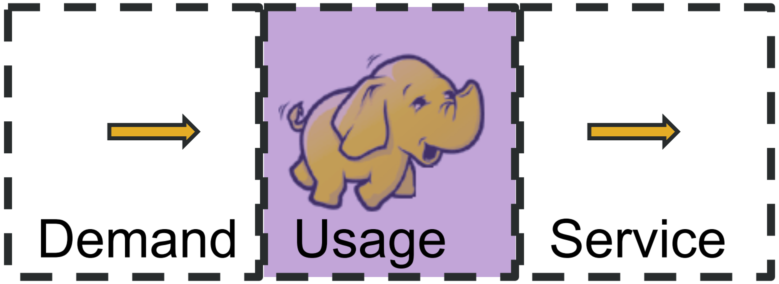
By demand – 3 policies:

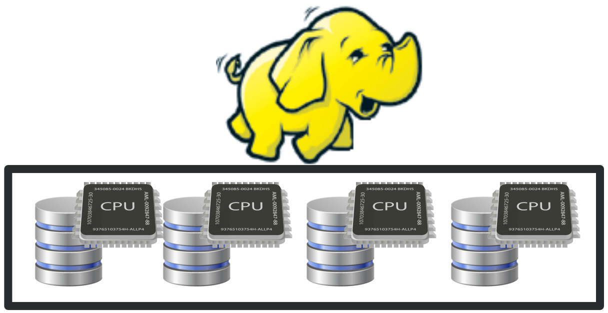
- Job Demand (JD)
- Data Demand (DD)
- Task Demand (TD)

versus



How to differentiate frameworks (2/3)

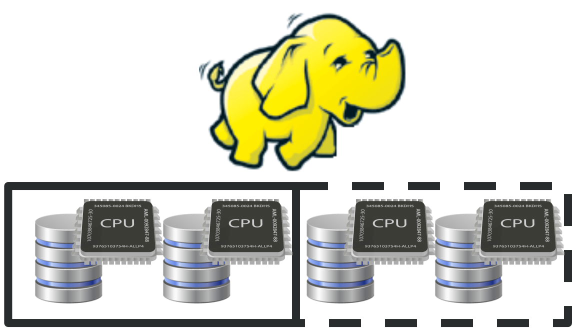




USED

versus

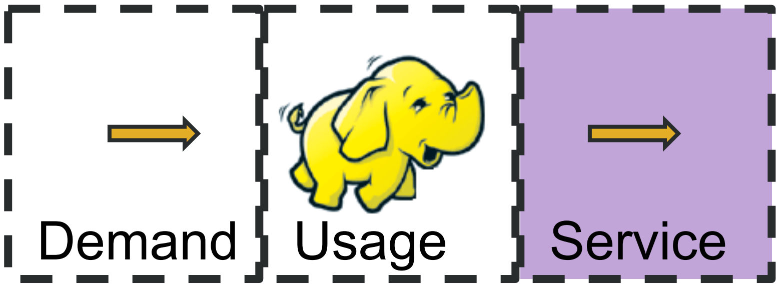
IDLE

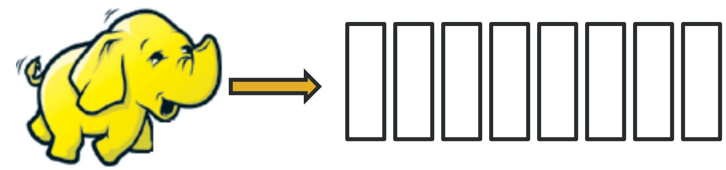


By usage – 3 policies:

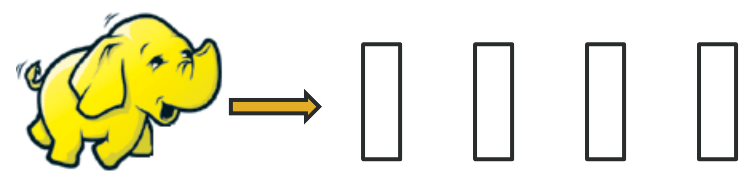
- Processor Usage (PU)
- Disk Usage (DU)
- Resource Usage (RU)

How to differentiate frameworks (3/3)





versus



By service – 3 policies:

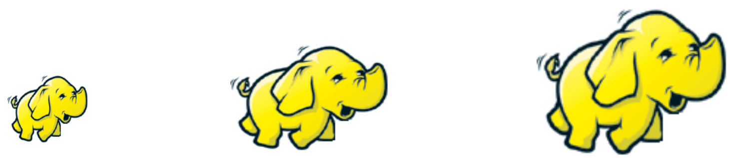
- Job Slowdown (JS)
- Job Throughput (JT)
- Task Throughput (TT)

TU Delft

Fairness or balanced service levels

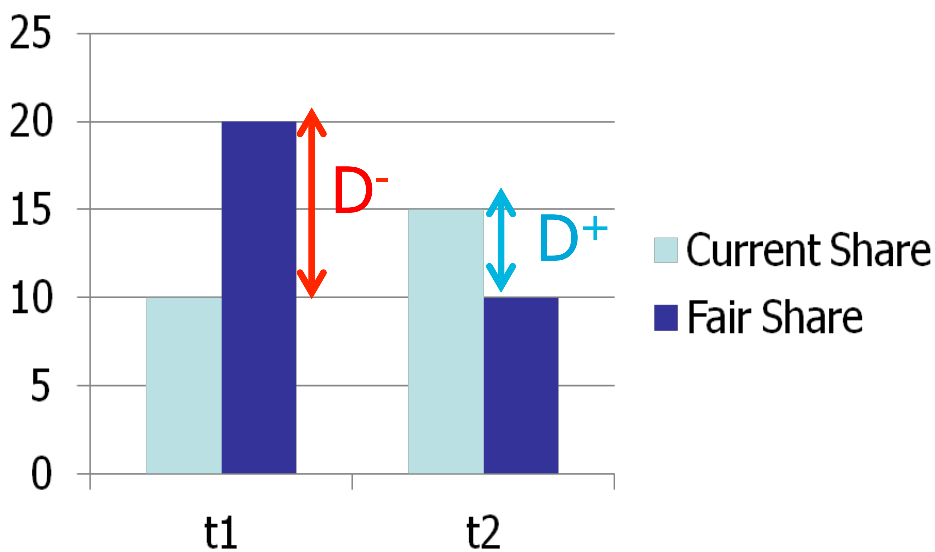
MR framework shares are proportional to their weights

- Weights are set from the system operation
- Temporal discrimination = *current share – entitled share*



w_1 < w_2 < w_3

$$s_i = \frac{w_i}{\sum w_j}$$



Measure of imbalance:

$$D_i(t_1, t_2) = \int_{t_1}^{t_2} (c_i(t) - w_i(t))dt$$

$$Var(D) > \tau$$

The grow-shrink mechanism

(1) Admission policy

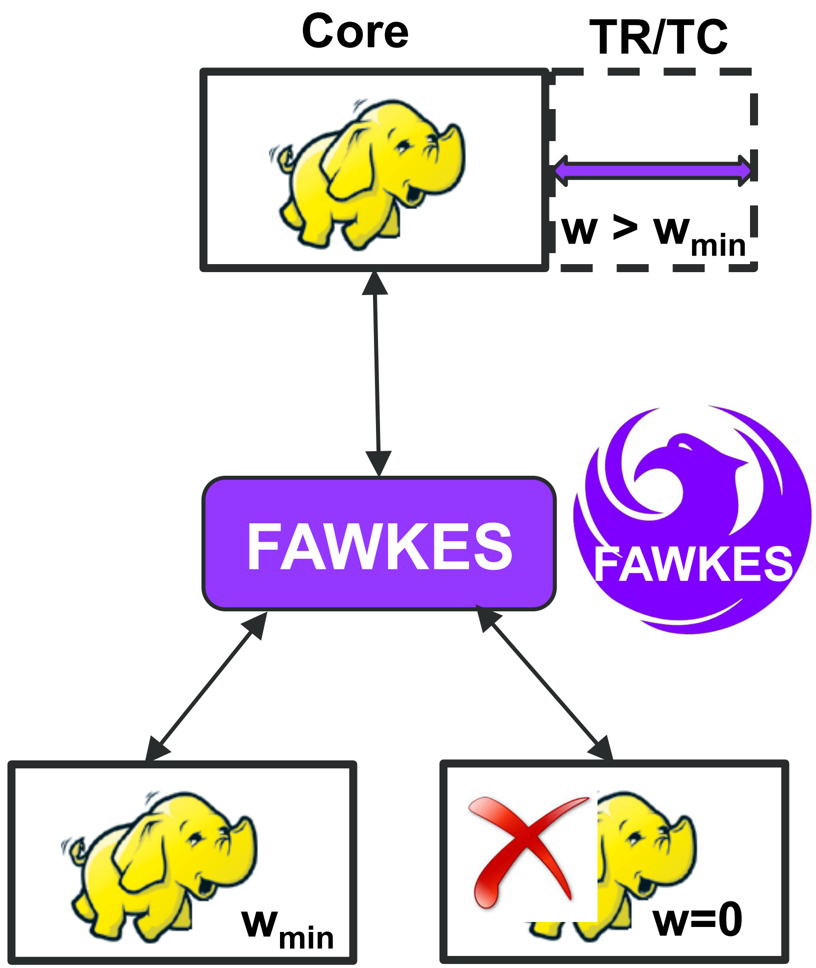
- Min. share guarantees
- Queue it if no free capacity

(2) Growing Mechanism

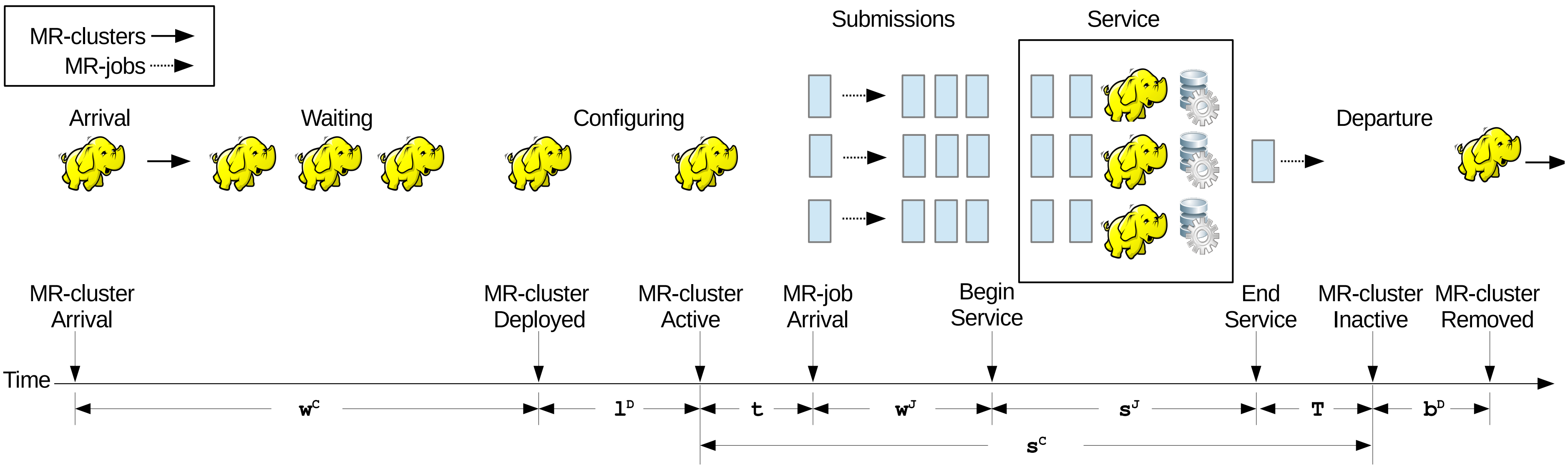
- Frameworks below their fair shares
- No locality – TR nodes
- Relaxed locality – TC nodes

(3) Shrinking Mechanism

- Frameworks above their fair shares
- Instant preemption – TR nodes
- Delayed preemption – TC nodes



It's a complex system



Our methodology to evaluate the system:

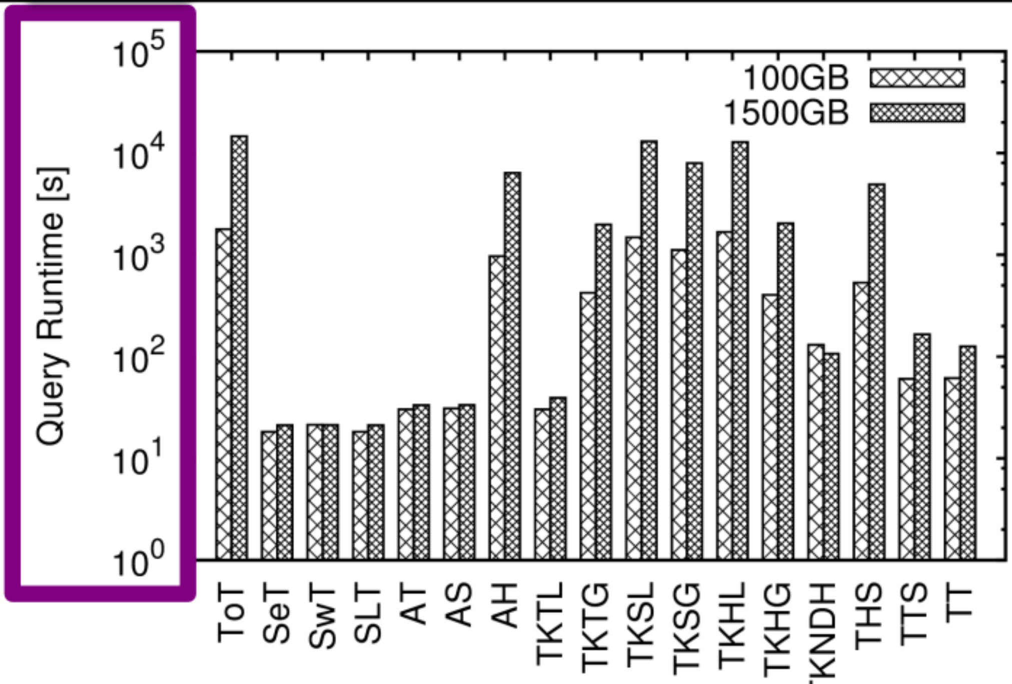
1. Design relevant workloads
2. Evaluate separate aspects of the system
3. Evaluate the full system

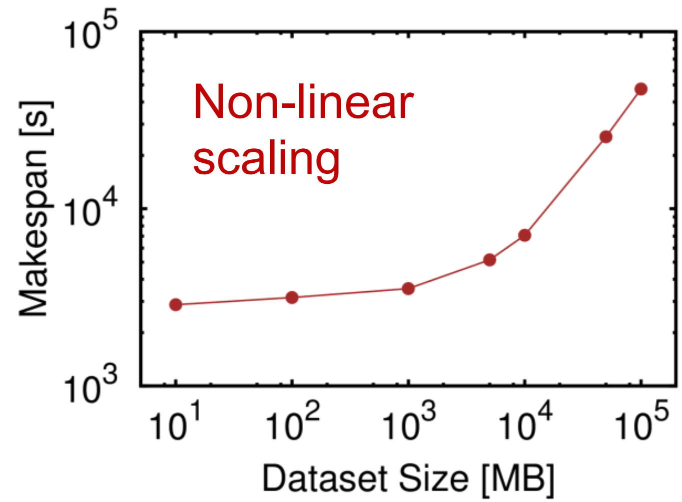
More than 60,000 hours system time!

TU Delft

MapReduce workloads

	Queries/Jobs	Workload Diversity	Data Set	Data Layout	Data Volume
MRBench [15]	business queries	high	TPC-H	relational data	3 GB
N-body Shop [14]	filter and correlate data	reduced	N-body simulations	relational data	50 TB
DisCo [6]	co-clustering	reduced	Netflix [29]	adjacency matrix	100 GB
MadLINQ [7]	matrix algorithms	reduced	Netflix [29]	matrix	2 GB
ClueWeb09 [30]	web search	reduced	Wikipedia	html	25 TB
GridMix [16], PigMix [17]	artificial	reduced	random	binary/text	variable
HiBench [31], PUMA [32]	text/web analysis	high	Wikipedia	binary/text/html	variable
WL Suites [12]	production traces	high	-	-	-
BTWorld	**P2P analysis**	**high**	**BitTorrent logs**	**relational data**	**14 TB**

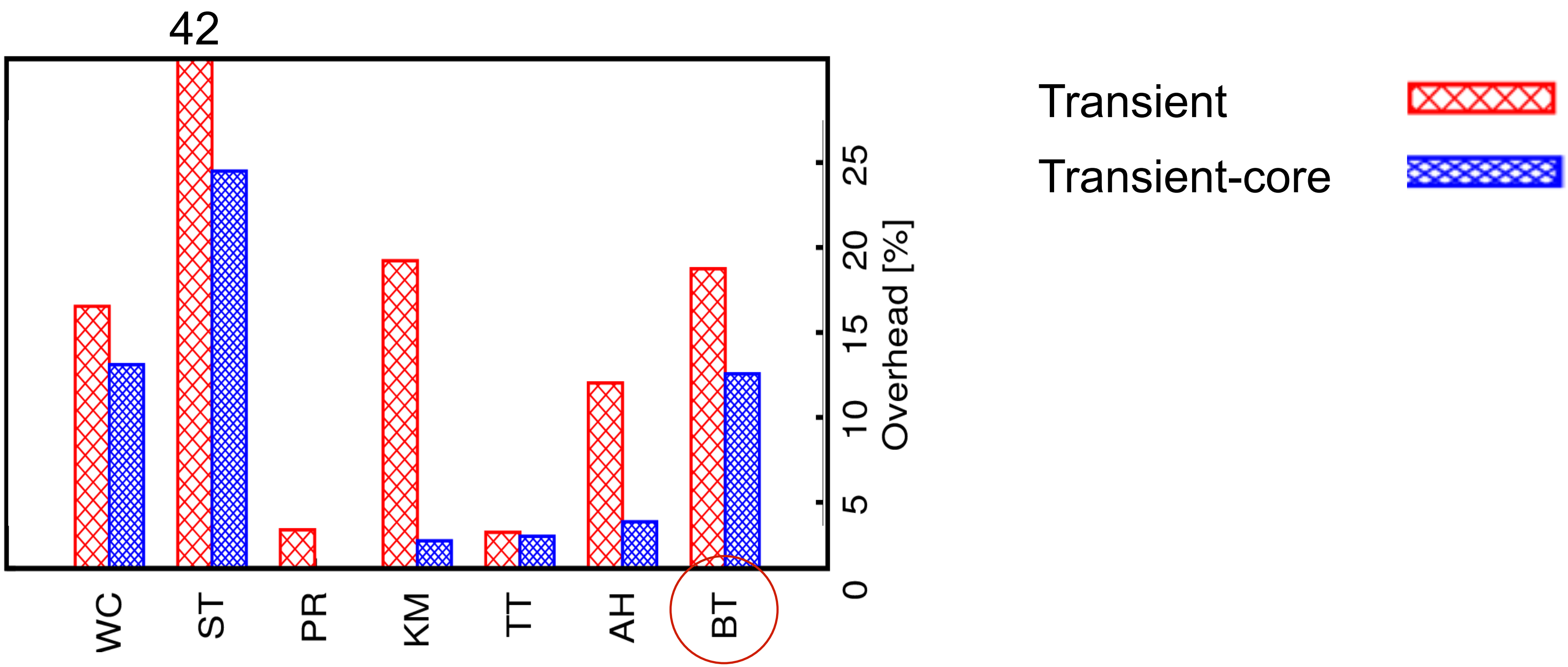




B. Ghit, M. Capota, T. Hegeman, D. Epema, A. Iosup. V for Vicissitude: The Challenge of Scaling Complex Big Data Workflows. In ACM/IEEE CCGrid (Winner of Scale Challenge 2014)

TUDelft

Performance of no versus relaxed locality



Complete workflow on 100 GB

- Single-application performance overhead
- 10 core nodes + 10 transient/transient-core nodes

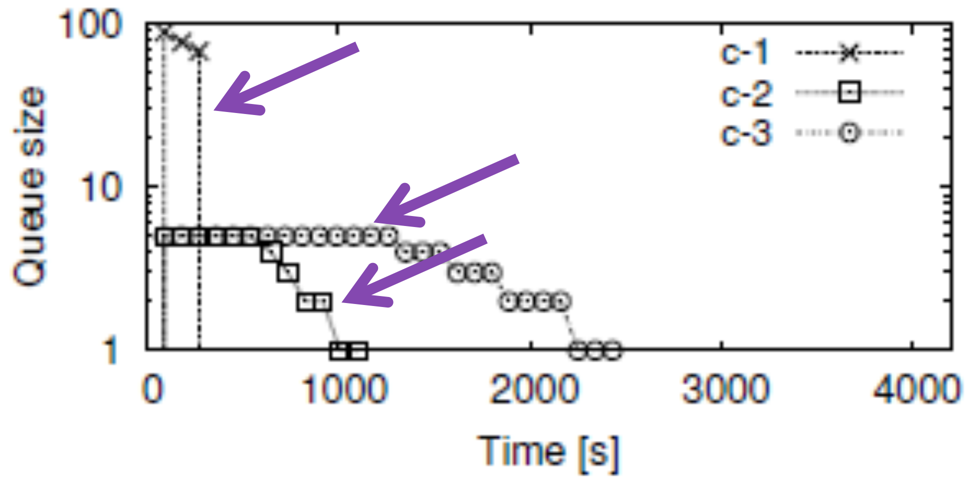
Performance of Fawkes: closed system

Nodes	45
Frameworks	3
Min. shares	10
Datasets	200 GB
Jobs submitted	100

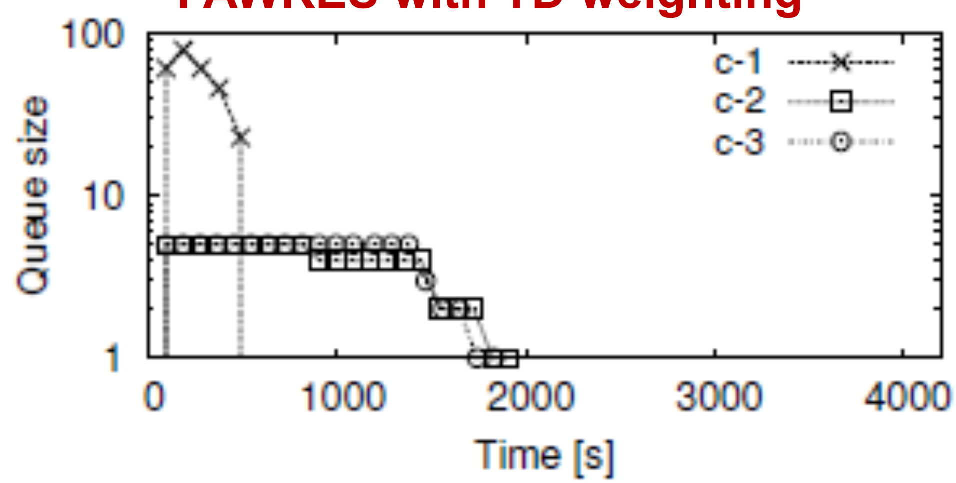
Closed system

- c-1: 90 x 1 GB sort jobs
- c-2: 5 x 50 GB sort jobs
- c-3: 5 x100 GB sort jobs

FAWKES with static allocation

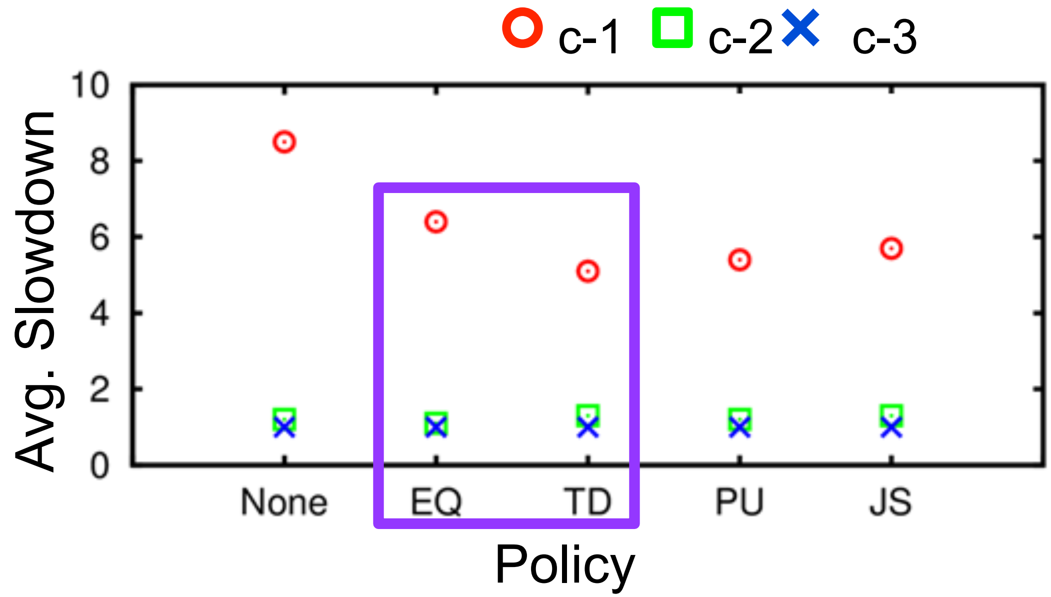


FAWKES with TD weighting



Performance of Fawkes: open system

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



Open system

- Poisson arrivals
- c-1: 1 – 100 GB Wordcount and Sort jobs
- c-2, c-3: 1 GB Wordcount and Sort jobs

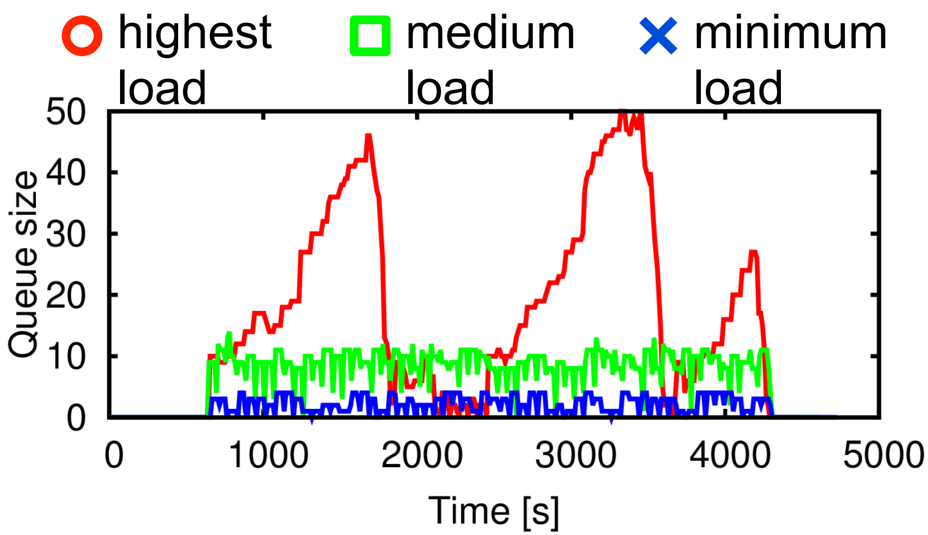
None – Minimum shares
EQ – EQual shares
TD – Task Demand
PU – Processor Usage
JS – Job Slowdown

Up to 20% lower slowdown

TUDelft

Fawkes behind the scenes

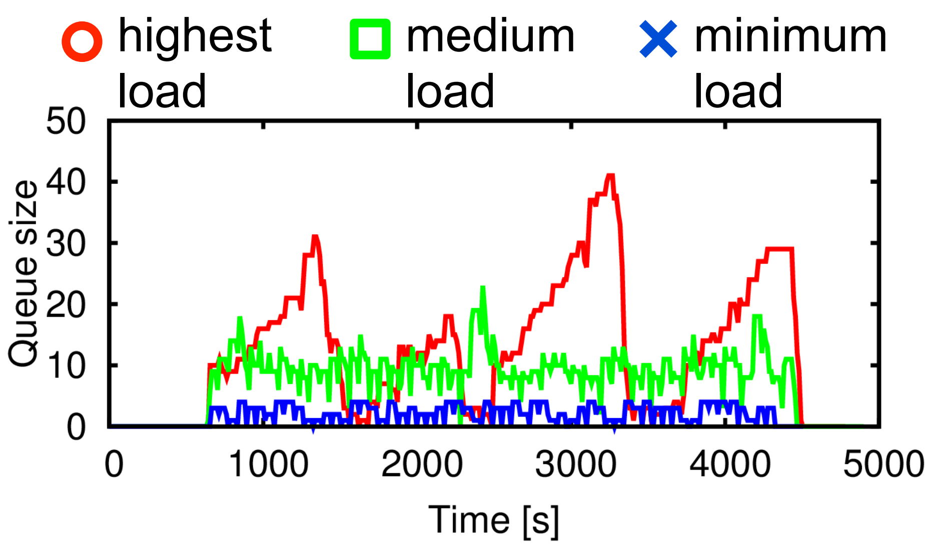
EQ



Utilizations: 60% / 23% / 5%

Imbalanced

TD



Utilizations: 50% / 30% / 8%

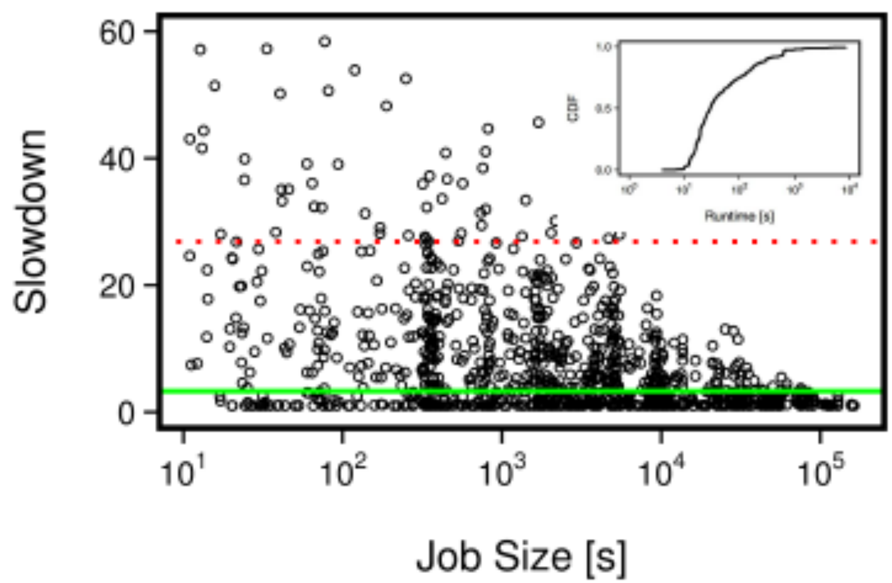
More balanced

Can we do better?

MapReduce workloads

- Challenging for existing schedulers
- High job size variability
- Short jobs prevail, but long jobs dominate

FIFO with a Facebook trace

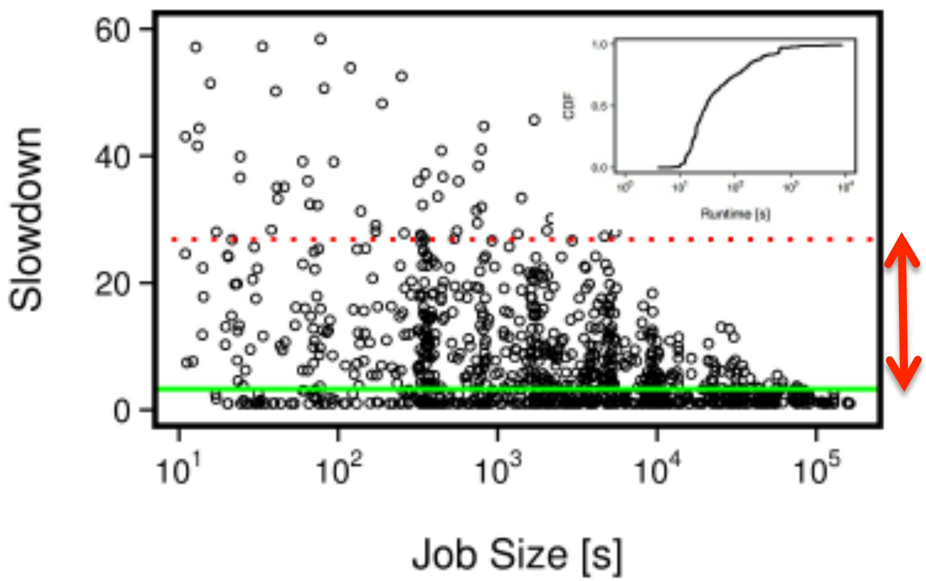


Job slowdown variability

Definitions:

- *Job size* = sum of its task runtimes
- *Job slowdown* = ratio between the sojourn time and the runtime in isolation
- *Job slowdown variability* = ratio between job slowdown at the 95th percentile and the median job slowdown

FIFO with a Facebook trace



B.I. Ghit and D.H.J. Epema, “Reducing Job Slowdown Variability for Data-Intensive Workloads”, IEEE MASCOTS 2015.

Size-based scheduling

Previous work:

- PS has job slowdown independent of job size: $E[S(x)] = \dfrac{1}{1-\rho}$
- SRPT is response time-optimal, but jobs may starve.

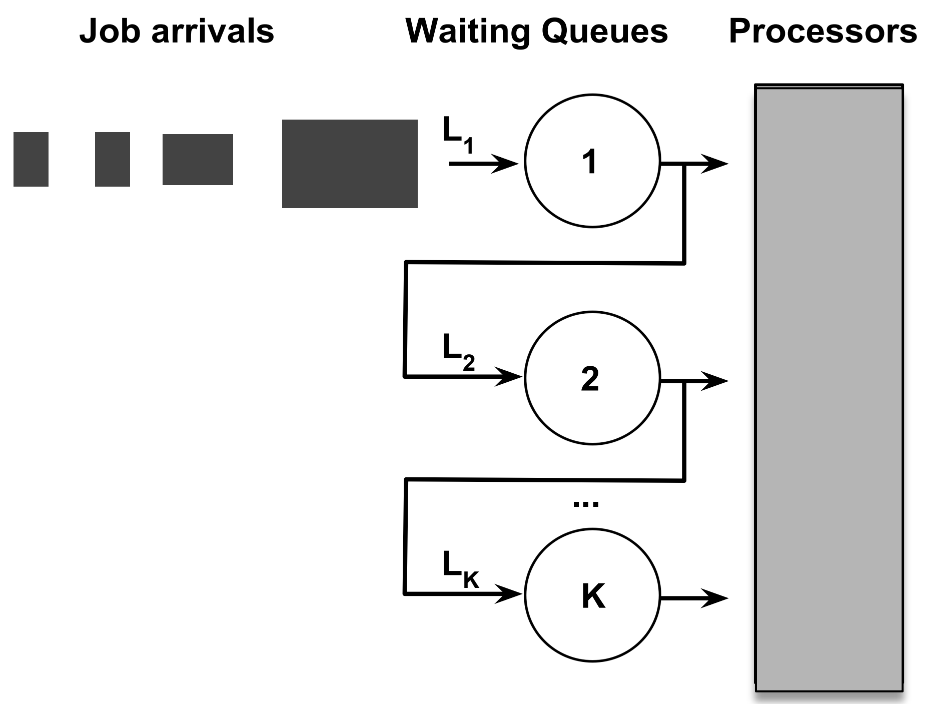
Partitions AND Feedback	Only Feedback
Only Partitions	None

Main mechanisms:

1. Logical partitioning
   - Allocate processors to disjoint partitions
   - Restrict amount of service offered to jobs
2. System feedback
   - Job preemption in a work-conserving way
   - Pause/resume jobs using HDFS

The FBQ policy

Partitions	NO
Feedback	YES

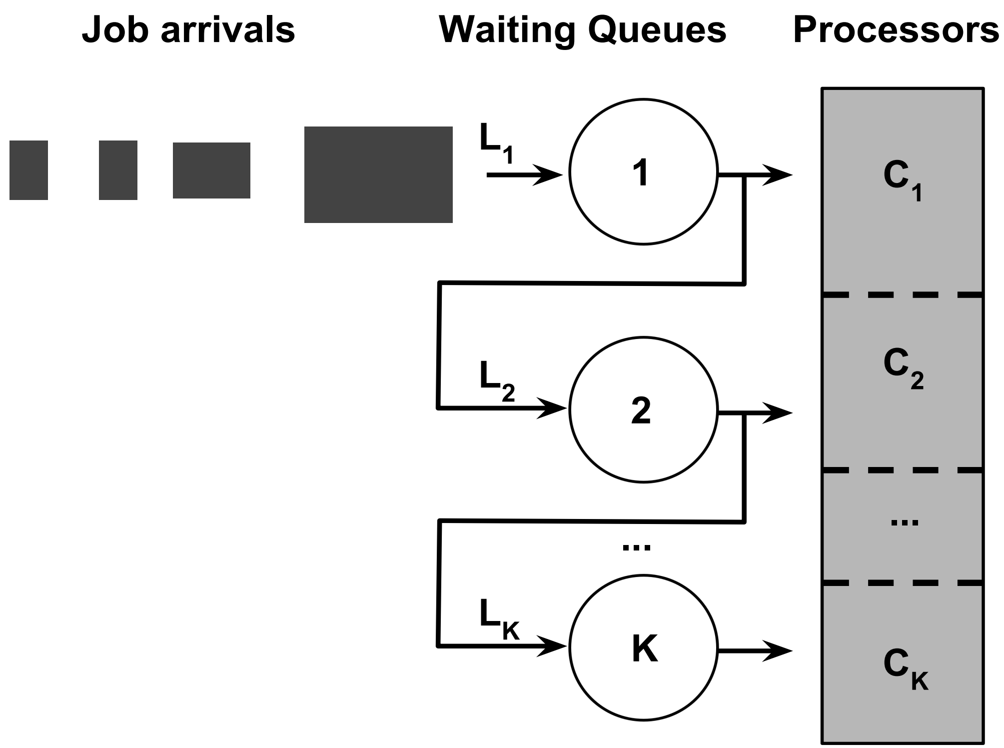


- Uses feedback, but no resource partitioning
- When job reaches queue time limit, then it is paused and moved to a lower priority queue.

L.E. Schrage, “The M/G/1 queue with feedback to lower priority queues”, Management Sciences, 1967.

The TAGS policy

Partitions	YES
Feedback	YES

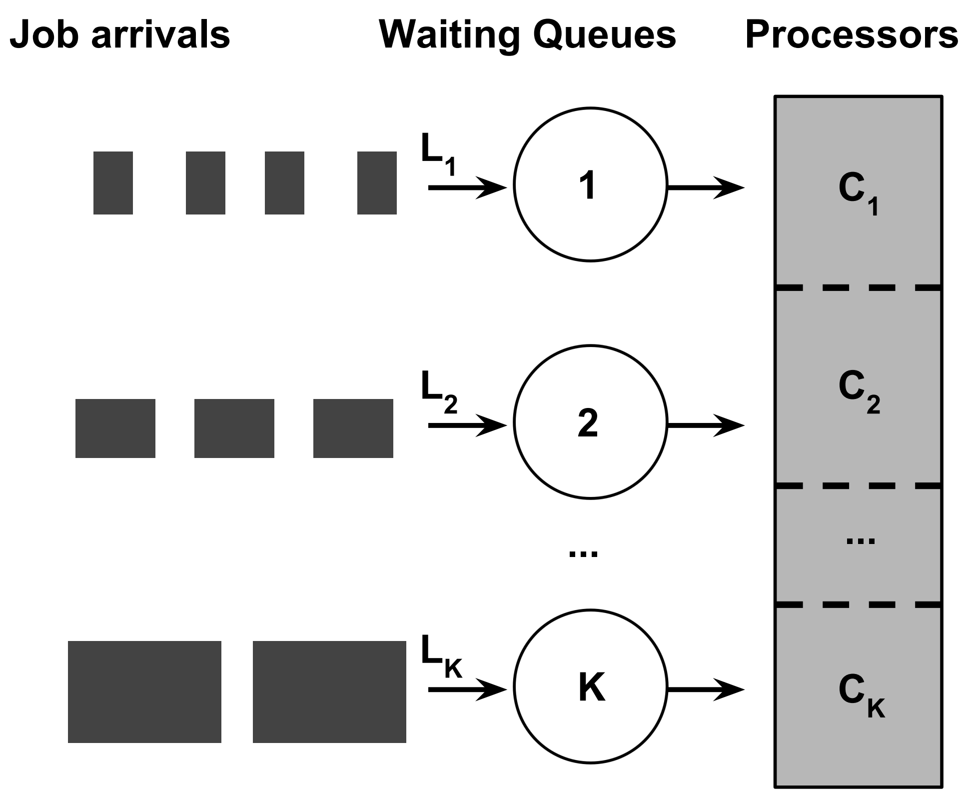


- Uses feedback, but each queue has its own partition
- When job reaches queue time limit, then it is paused and resumed at the next queue.

M. Harchol-Balter, “Task assignment with unknown durations”, Distributed Computing Systems, 2001.

The SITA policy

Partitions	YES
Feedback	NO



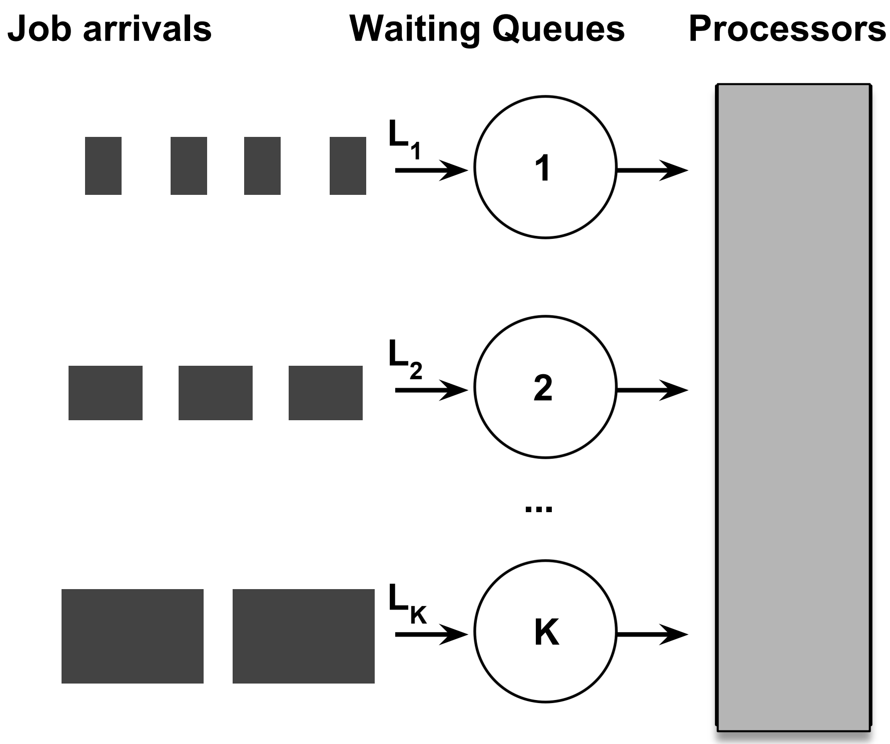
+PREDICTION

- Per-queue resource partitions, but no feedback
- Dispatch jobs to queues based on their sizes

M. Harchol-Balter et. al., "On choosing a task assignment policy for a distributed server system", Parallel and Distributed Computing, 1999.

The COMP policy

Partitions	NO
Feedback	NO



+PREDICTION

- No resource partitioning, no feedback
- Append to queue m+1 if larger than m of the last K-1 completed jobs

Jian Tan et. al., “Adaptive scalable comparison scheduling”, SIGMETRICS, 2007.

Contrasting the policies

Previous work

- Single or distributed-server model
- Simple, rigid non-preemptive jobs
- Wasted work by killing jobs

Our work

- Datacenters with very large capacity
- Malleable MapReduce jobs
- Work-conserving approach

Policy	Queues	Partitions	Feedback	Job Size	Param.
FIFO	single	no	no	unknown	0
FBQ	multiple	no	yes	unknown	K
TAGS	multiple	yes	yes	unknown	$2K-1$
SITA	multiple	yes	no	predicted	$2K-1$
COMP	multiple	no	no	compared	1

TUDelft

Simulator validation (1/2)

Apache Mumak with two main improvements:

- Accurately modeling of the shuffle phase
- Removal of the periodic heartbeat in JT-TT communication

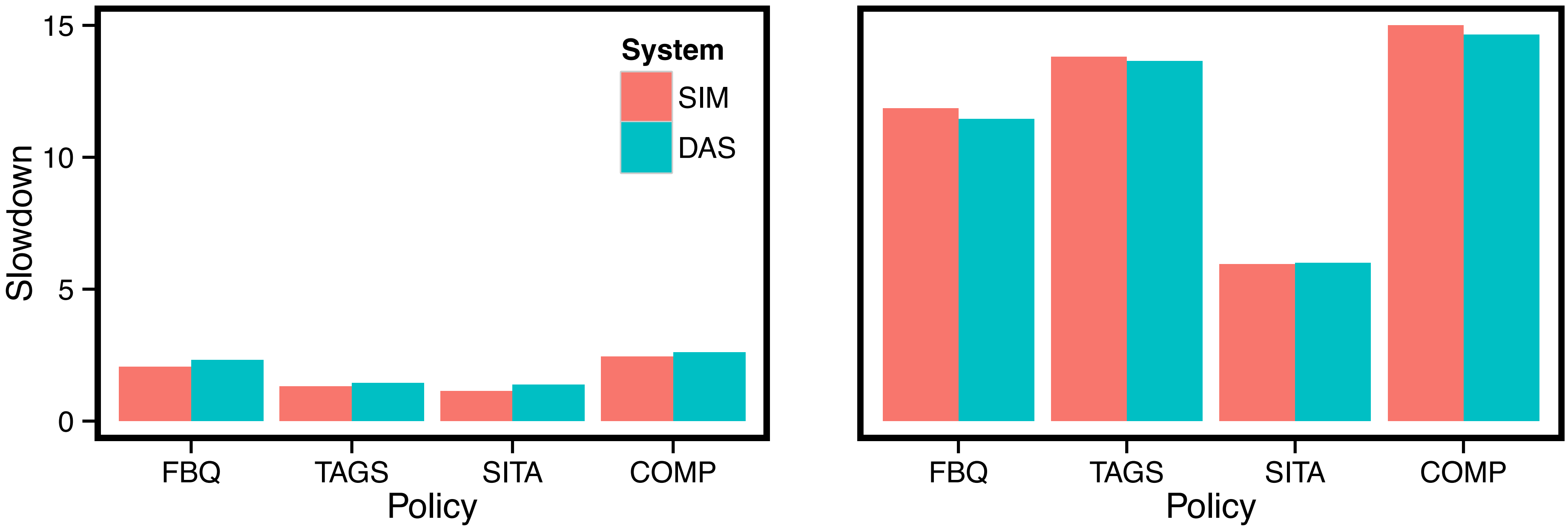
Mumak versus Hadoop on DAS-4

- 10 nodes with 6 map slots and 2 reduce slots
- Single jobs: Grep, Sort, Wordcount

Applications	Maps	Reduces	Job Size [s]	SIM [s]	DAS [s]	Jobs
GREP	2	1	63.14	36.10	43.26	26
SORT	4	1	60.20	32.70	39.97	4
WCOUNT	4	1	126.14	42.04	49.73	4
GREP	50	5	155.32	42.83	53.18	4
WCOUNT	100	10	3,790.46	86.80	93.62	3
SORT	200	20	5,194.64	149.92	156.89	3
GREP	400	40	15,697.18	233.63	239.21	3
WCOUNT	600	60	26,662.53	579.73	589.02	3

TUDelft

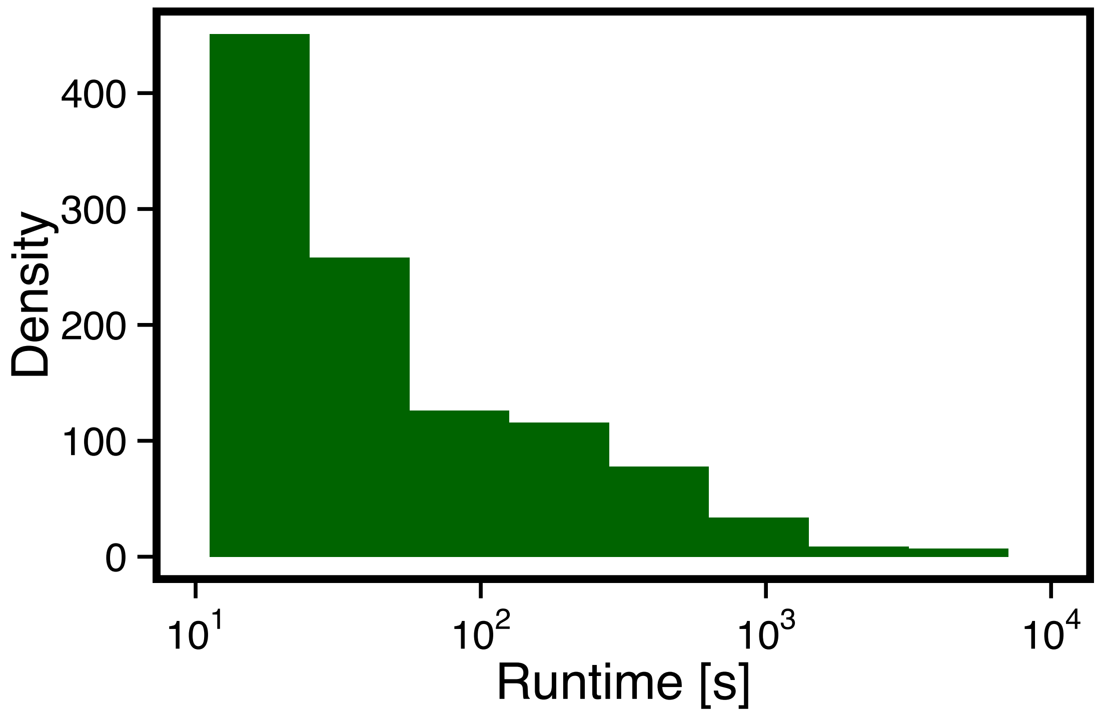
Simulator validation (2/2)



Median **95^{th}**

- Workload of 50 jobs, sys. load of 0.7
- Less than 1% error between SIM and DAS

Facebook workload (1/2)

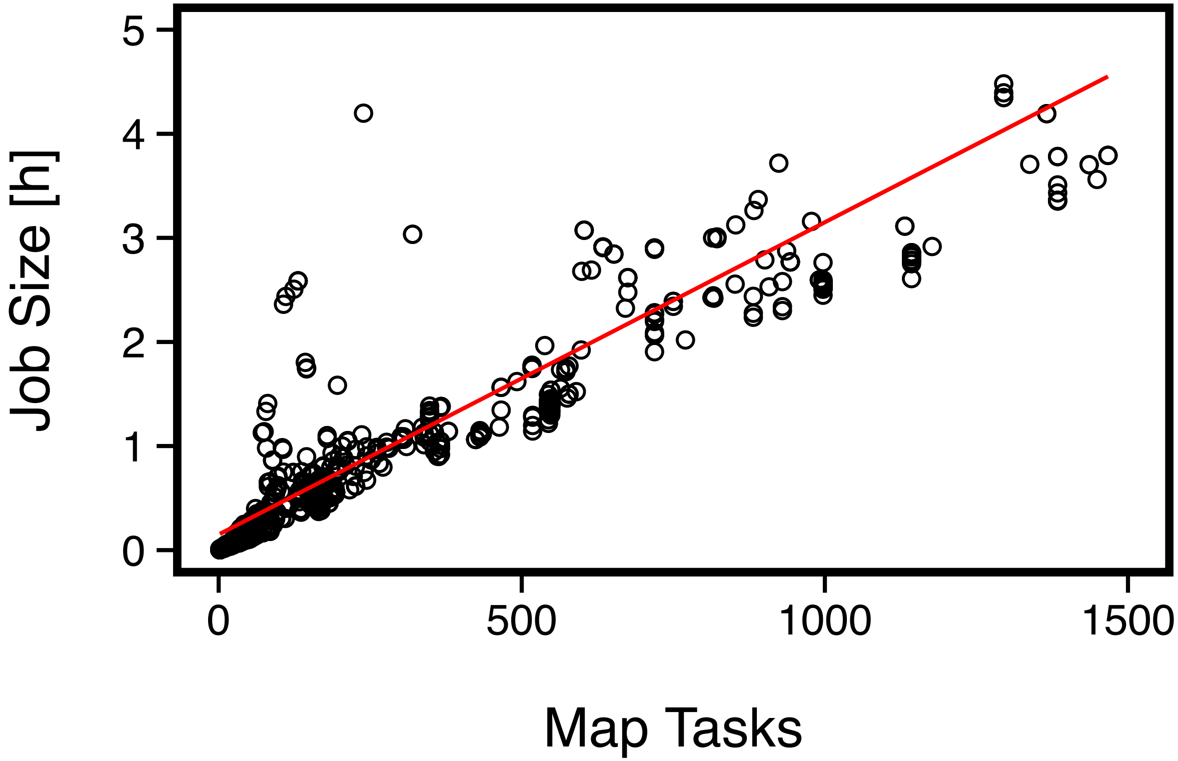


- Workload of 60 h of simulated time
- Very variable distribution: CV^2=16.35
- Mumak with 100 simulated nodes

- Less than 8% of the jobs = 50% of the total load

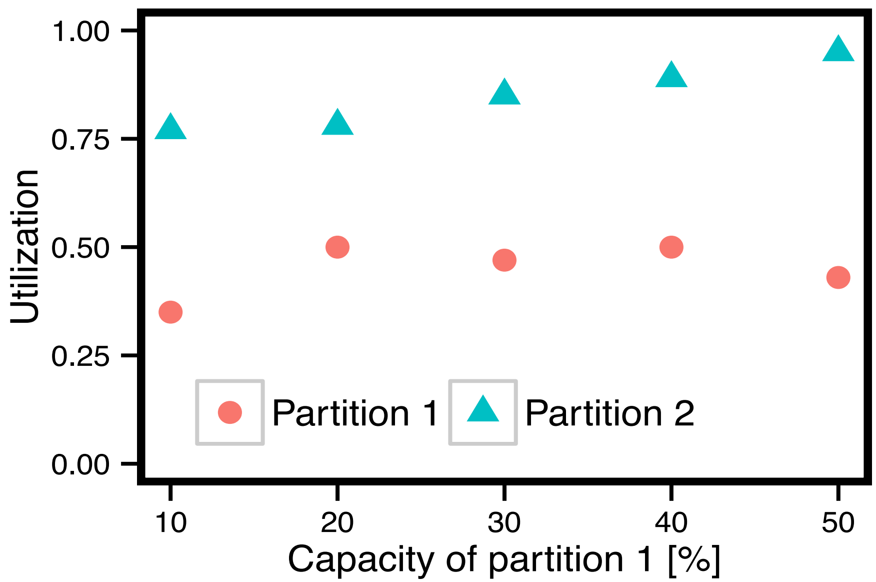
TUDelft

Facebook workload (2/2)

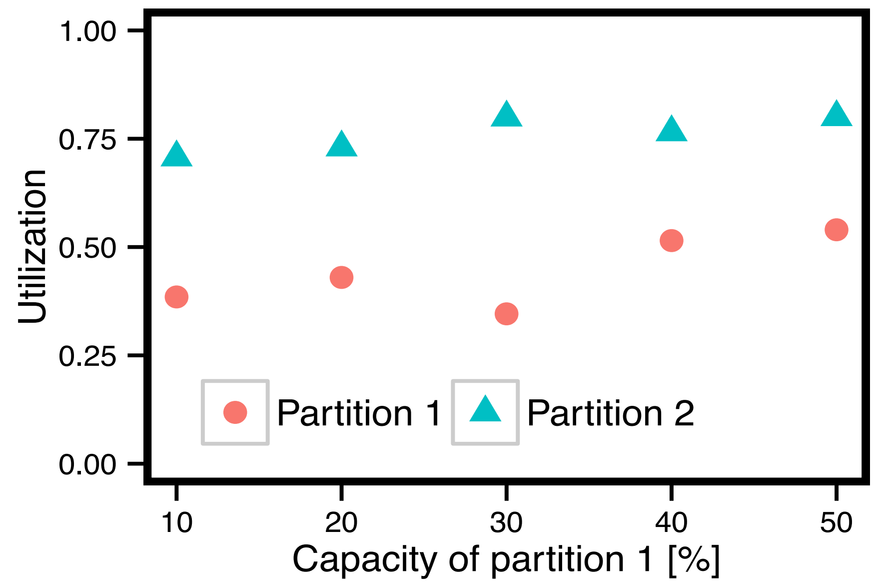


- Strong correlation between job input size and job proc. requirement

Load unbalancing



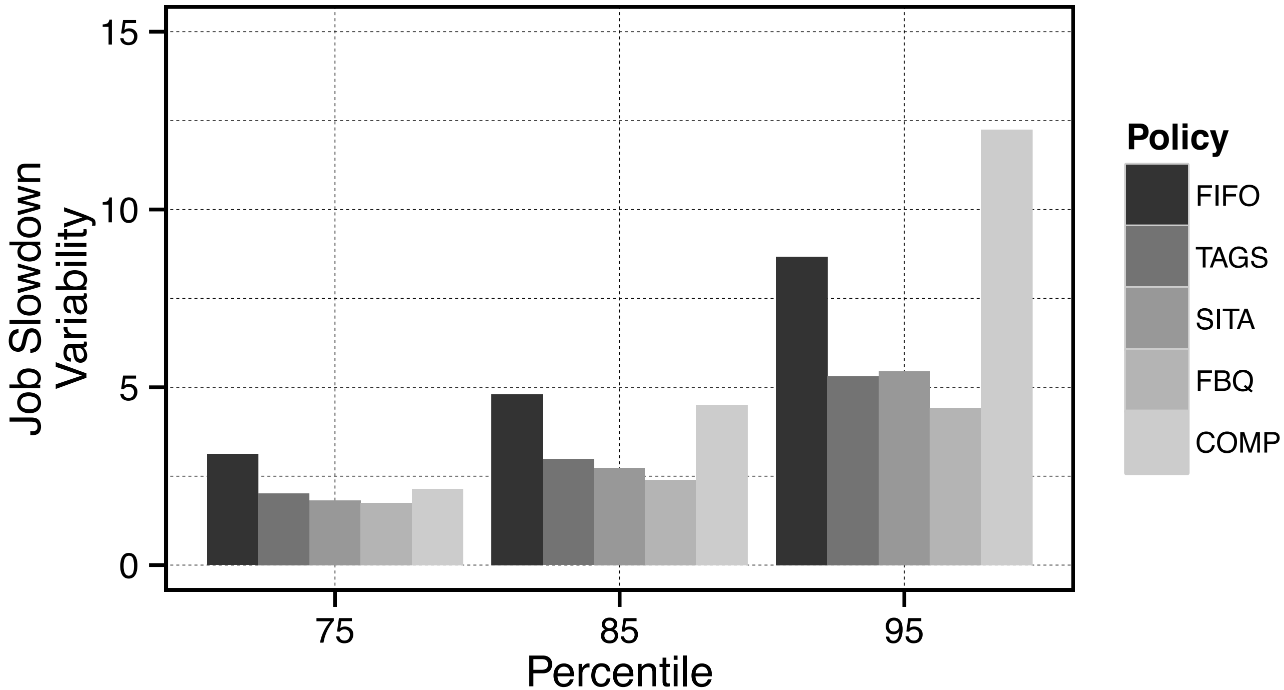
TAGS



SITA

- Partition 1 has significantly lower load than partition 2
- Higher load in partition 2 with TAGS than with SITA

Fairness analysis (1/2)



- All policies improve over FIFO
- TAGS and SITA shift variability to partition 2

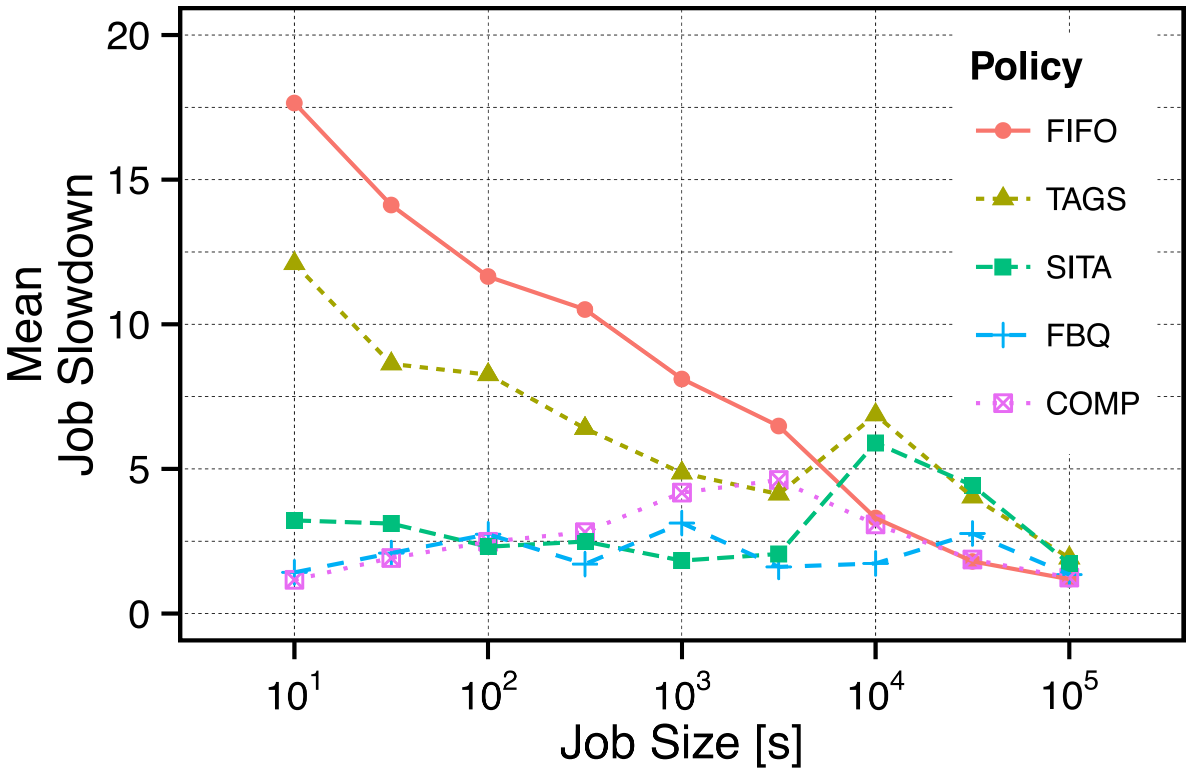
FBQ < SITA < TAGS < COMP < FIFO

Best

Worst

TU Delft

Fairness analysis (2/2)



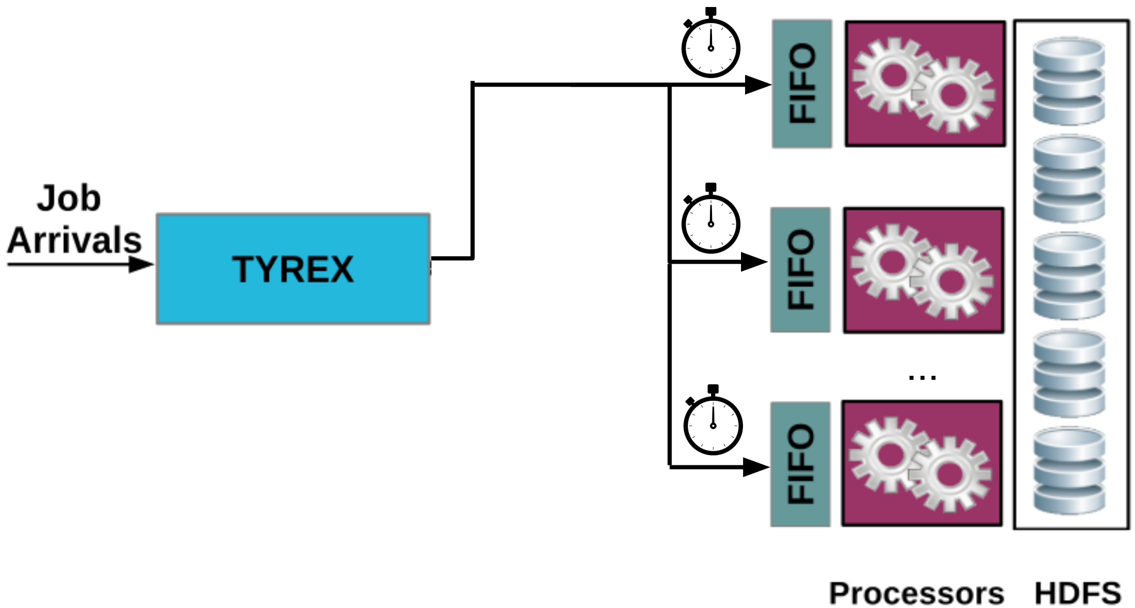
FBQ < SITA < COMP < TAGS < FIFO

Best

Worst

TU Delft

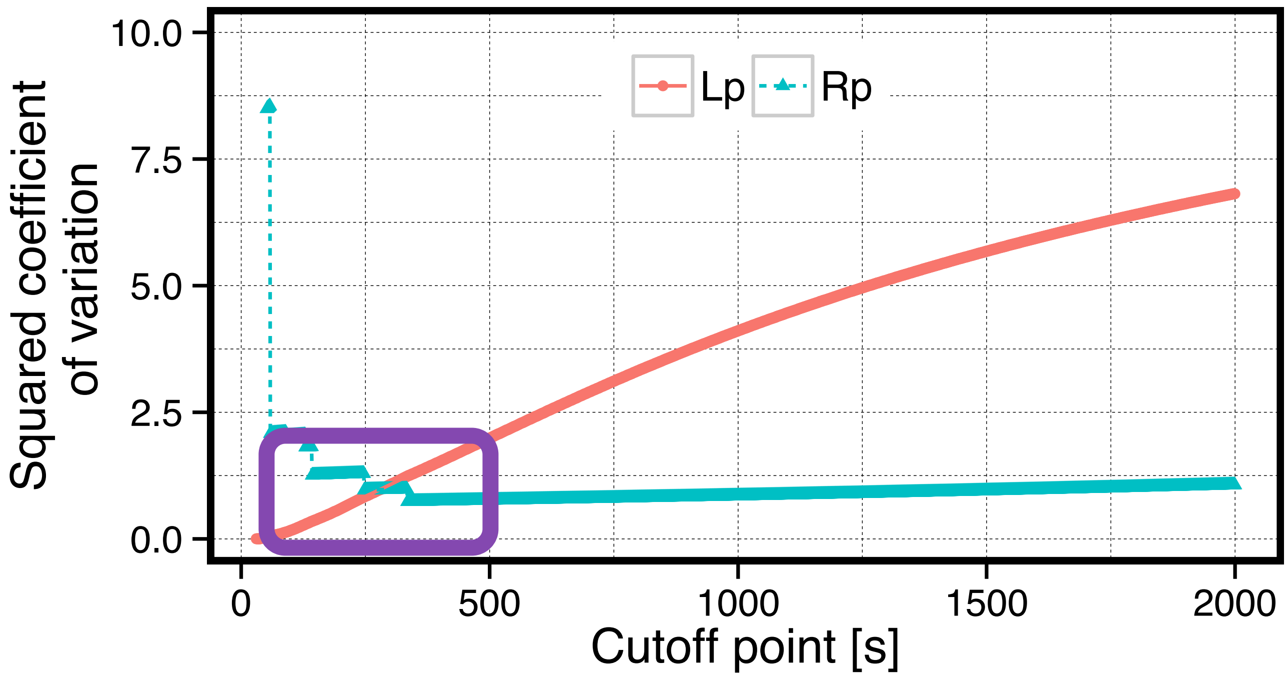
Tyrex: size-based resource allocation



- Based on previous design guidelines
- The cutoffs do not have closed forms
- May need to be recomputed frequently

Workload analysis

Migrate jobs that are likely to be way larger than the rest

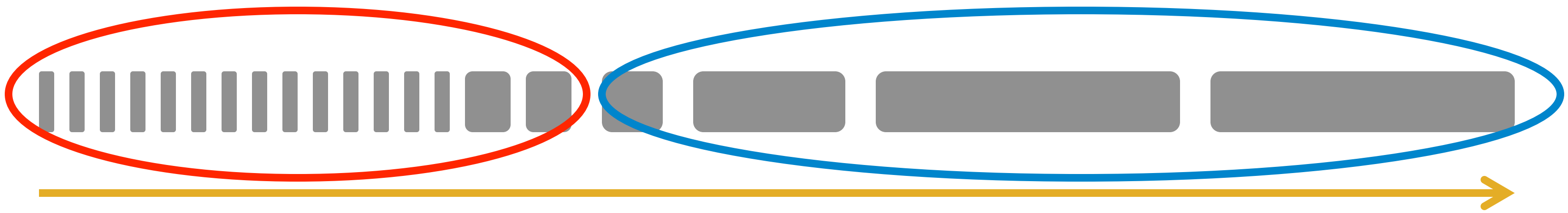


$L_p = \min(X, p)$
$R_p = X - p$

- Reduce the imbalance between Lp and Rp
- Aim for squared CV lower than 2 in any partition

TU Delft

The DynamicTags policy



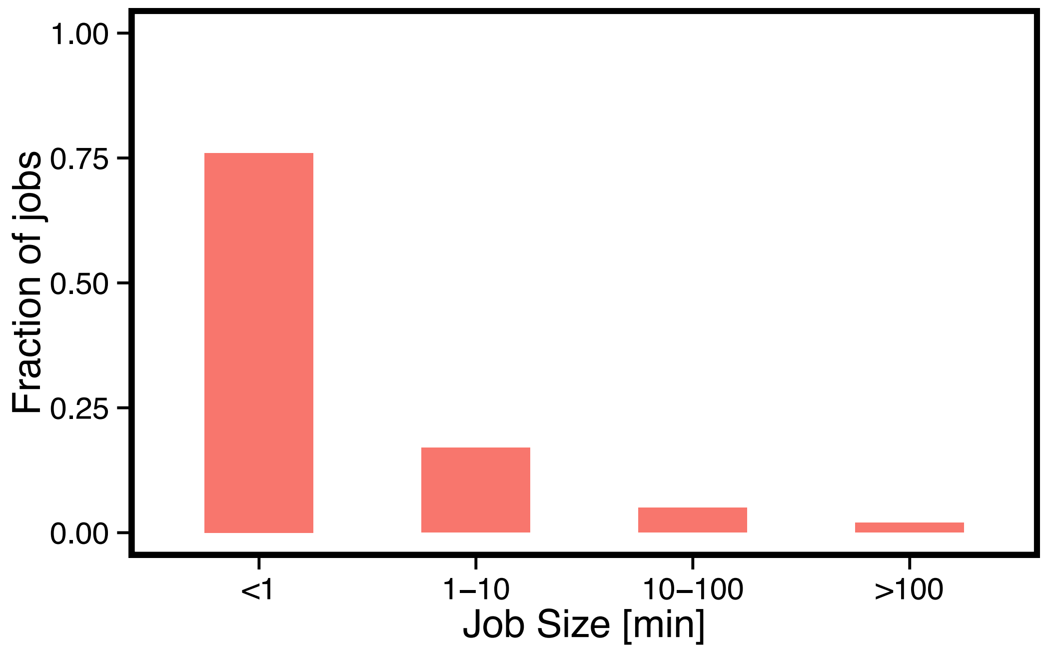
X = distribution of the current partial job size

- L_p captures the notion of young jobs
- R_p represents the residual lifetime of jobs
- **Optimal cutoff point p:** $CV^2(L_p) = CV^2(R_p)$
- Old jobs with large residual lifetimes are migrated

When the squared CV in a partition is higher than 2, then migrate all jobs that exceed the optimal cutoff point

TU Delft

Real-world workloads (1/3)

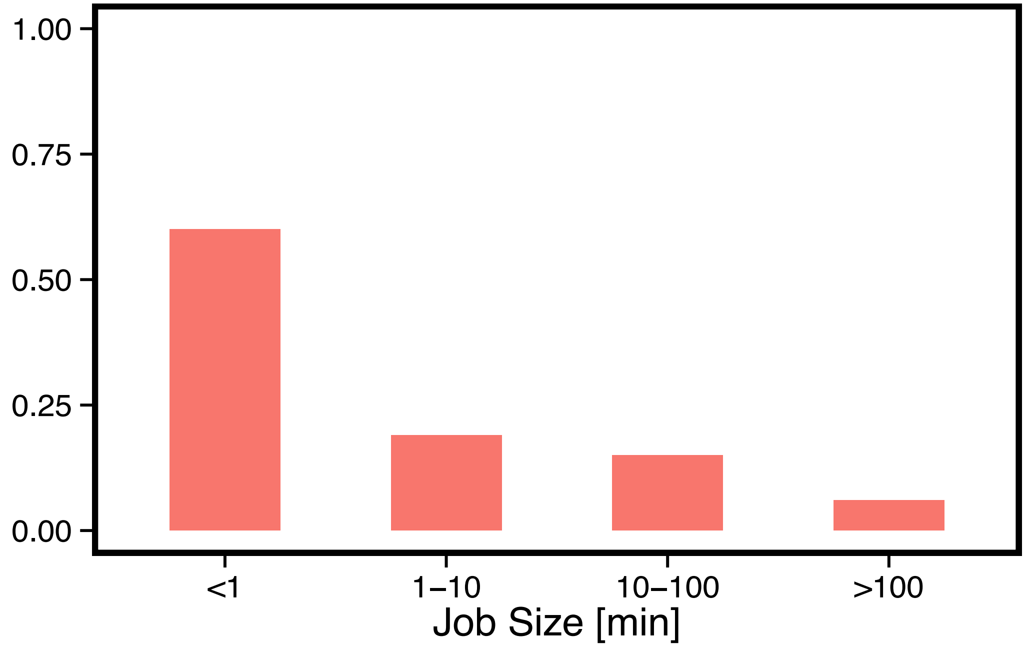


HVW
CV^2=20

Statistics	HVW	MVW	LVW
Total jobs		300	
BTWORLD jobs	33	45	10
Total maps	6,139	11,866	30,576
Total reduces	788	1,368	3,089
Temporary data [GB]	573	693	1,062
Persistent data [GB]	100	92	303
Total CPU time [h]	63.6	124.6	306.9
Total runtime [h]	3.51	3.98	5.31

TUDelft

Real-world workloads (2/3)

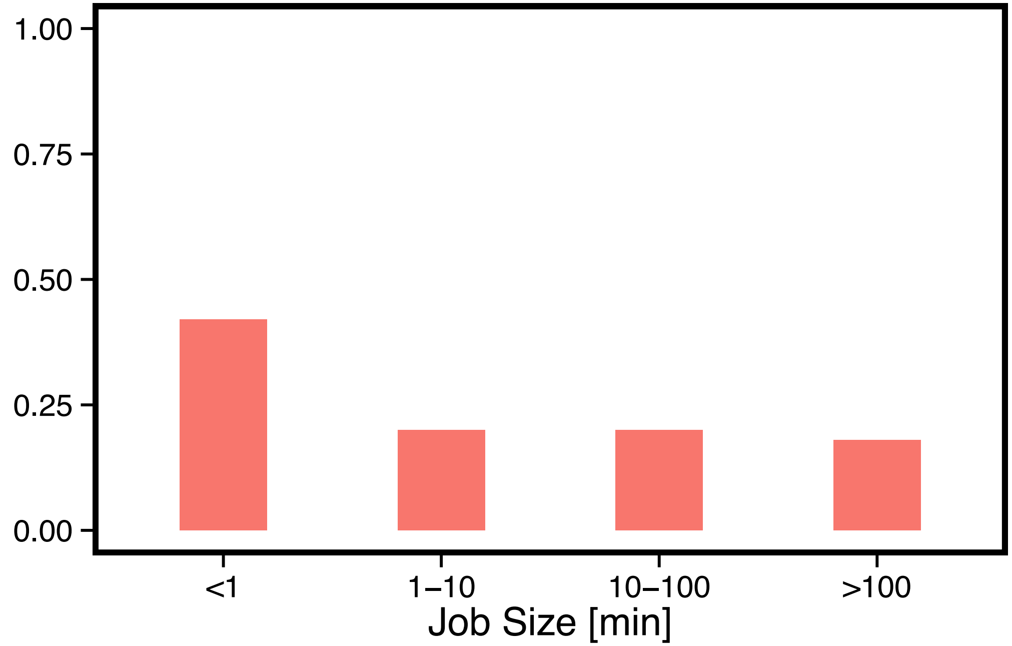


MVW
$CV^2=10$

Statistics	HVW	MVW	LVW
Total jobs		300	
BTWORLD jobs	33	45	10
Total maps	6,139	11,866	30,576
Total reduces	788	1,368	3,089
Temporary data [GB]	573	693	1,062
Persistent data [GB]	100	92	303
Total CPU time [h]	63.6	124.6	306.9
Total runtime [h]	3.51	3.98	5.31

TU Delft

Real-world workloads (3/3)

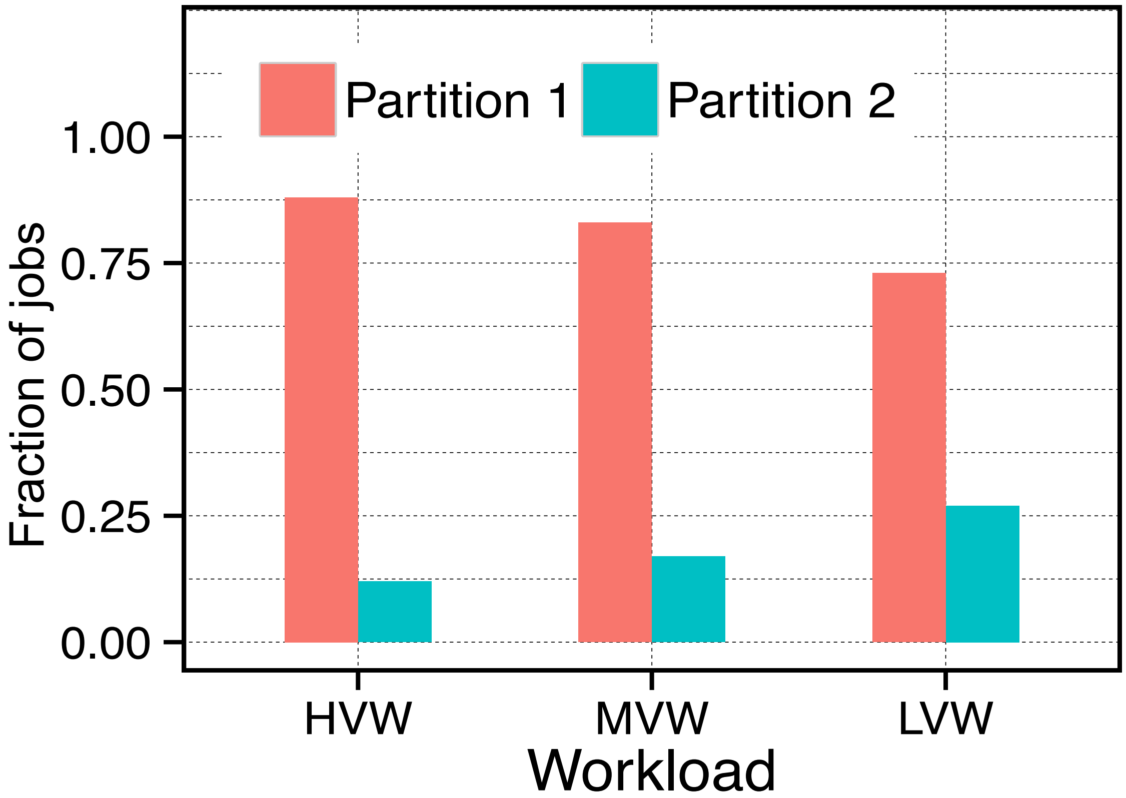


LVW
CV^2=4

Statistics	HVW	MVW	LVW
Total jobs		300	
BTWORLD jobs	33	45	10
Total maps	6,139	11,866	30,576
Total reduces	788	1,368	3,089
Temporary data [GB]	573	693	1,062
Persistent data [GB]	100	92	303
Total CPU time [h]	63.6	124.6	306.9
Total runtime [h]	3.51	3.98	5.31

TU Delft

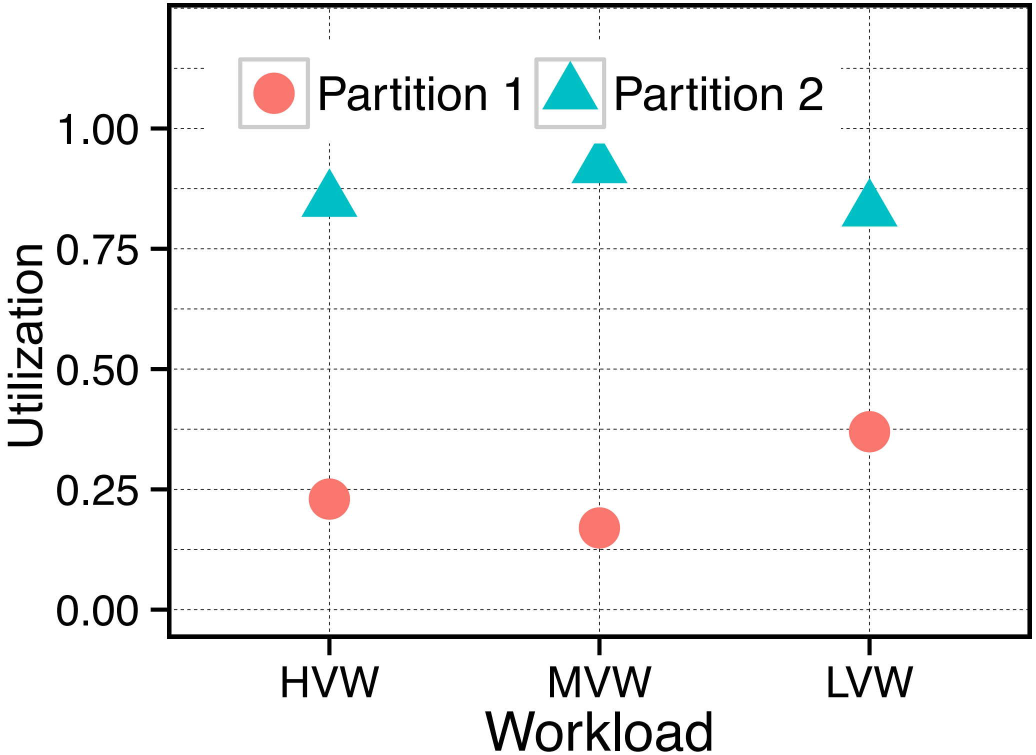
Fraction of jobs completed per partition



C_1 = 30%

- As the workload variability decreases, Tyrex migrates more jobs to partition 2.

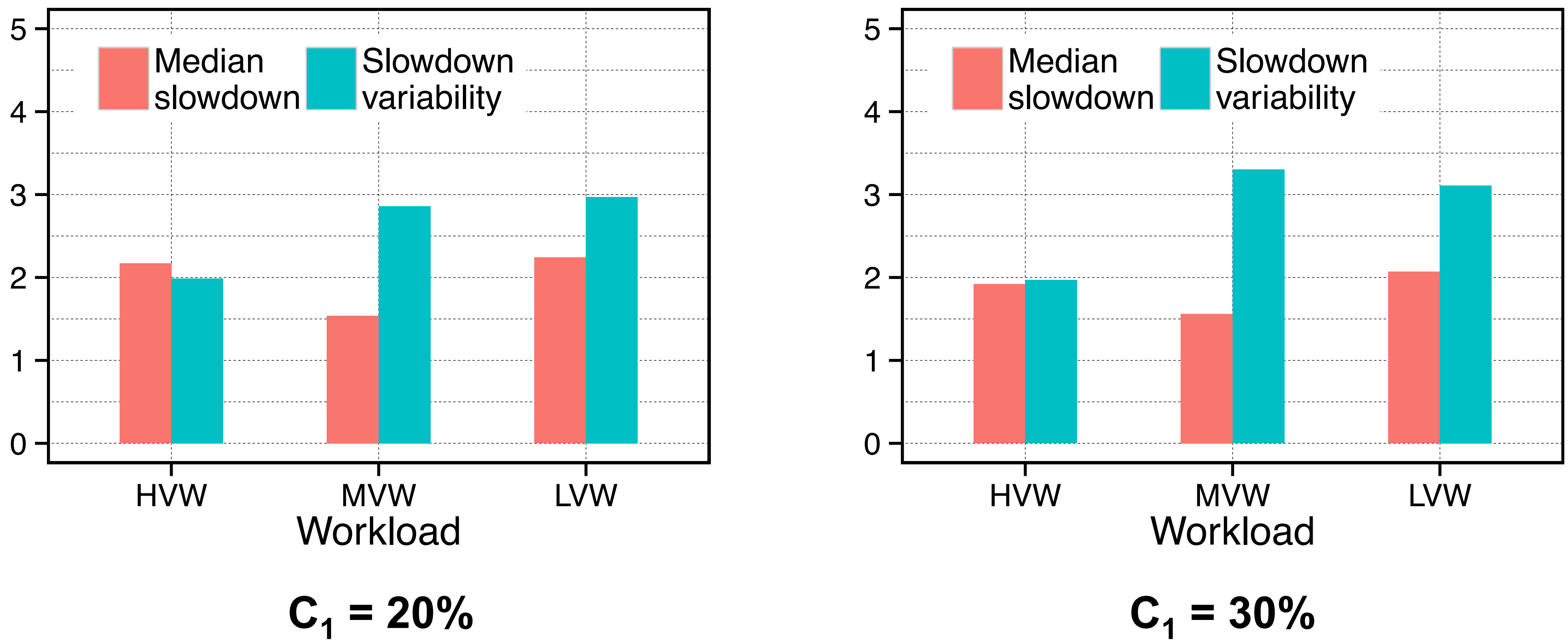
Load distribution across partitions



C_1 = 30%

- Tyrex is rather aggressive in migrating jobs to partition 2

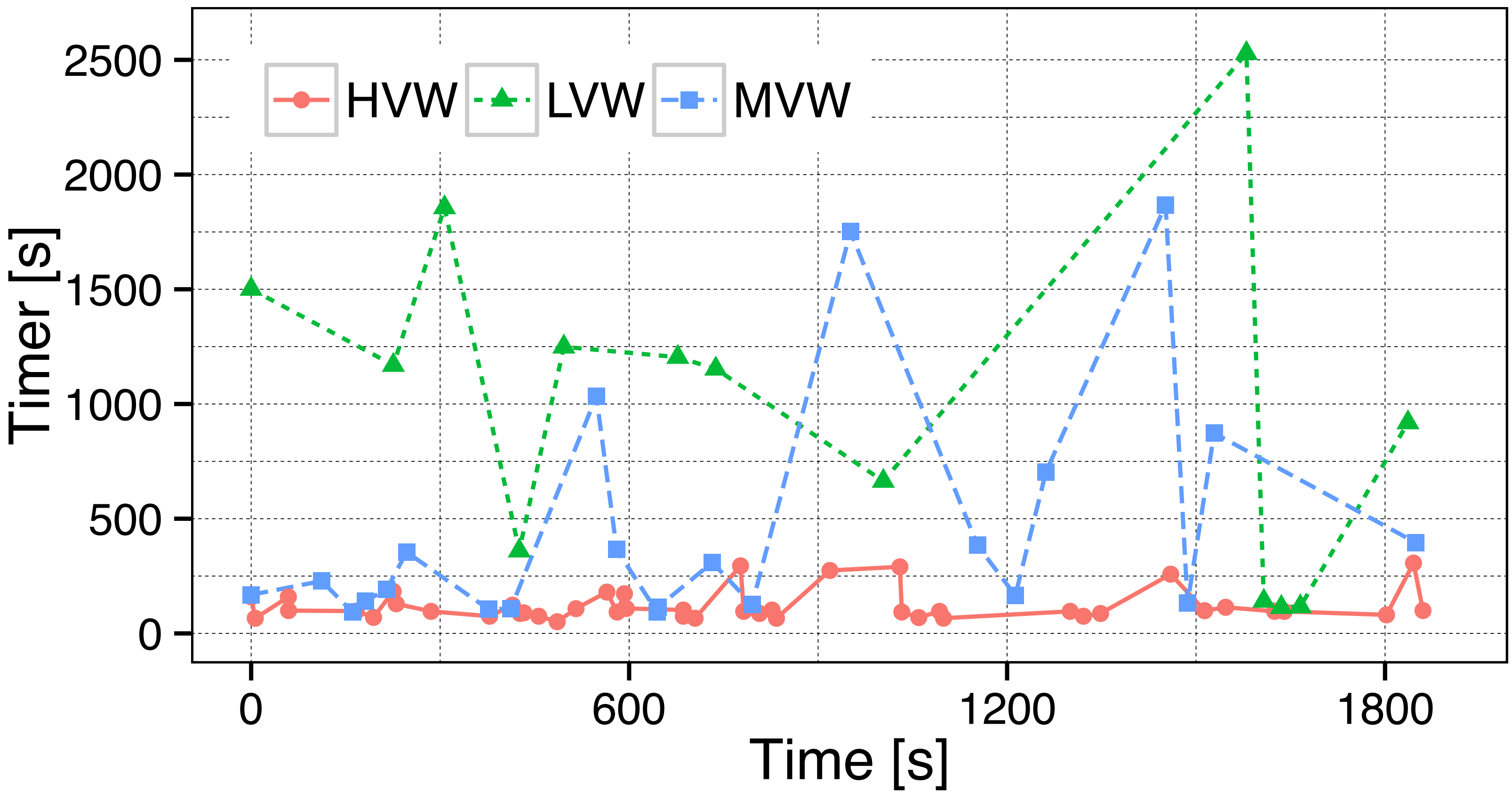
Slowdown performance of Tyrex



- Good slowdown performance for all workloads
- Similar improvements no matter the partition sizes

TUDelft

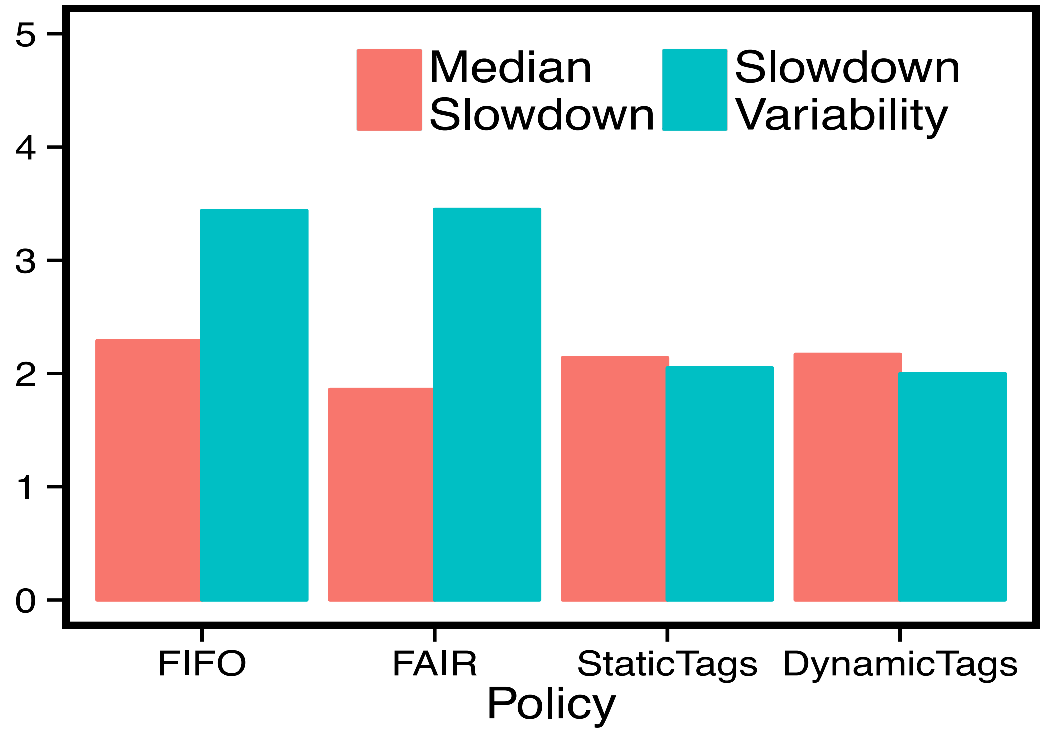
Dynamic timers



C_1 = 30%

- Converges to lower values for more variable distributions
- Exactly the range of values that equalizez the squared CV

Improvements from Tyrex



HVW
$CV^2=20$

C_1 = 20%

- Tyrex cuts in half the job slowdown variability when compared to FIFO and FAIR

TU Delft

Key takeaways

Big data = system of systems

- The stack of systems exposes many trade-offs
- Both fairness and performance are important
- Both simulation and experimentation are needed

In this talk

- New MR abstraction for elastic data processing
- Fawkes balances allocations even for highly imbalanced workloads
- Two main techniques to reduce the job slowdown variability
- Tyrex delivers competitive performance with the optimal parameter setting

Our research group

More information

- www.publications.st.ewi.tudelft.nl
- www.pds.ewi.tudelft.nl/ghit
- www.pds.ewi.tudelft.nl/epema

TODO

TODO

TODO