Dynamic Scheduling of Hadoop Clusters in Datacenters

Bogdan Ghiţ

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

Born in Bucharest, Romania

PhD Candidate at TU Delft

- supervised by **Dick Epema**
- *scheduling in clusters and performance of MapReduce*

Experimental research:

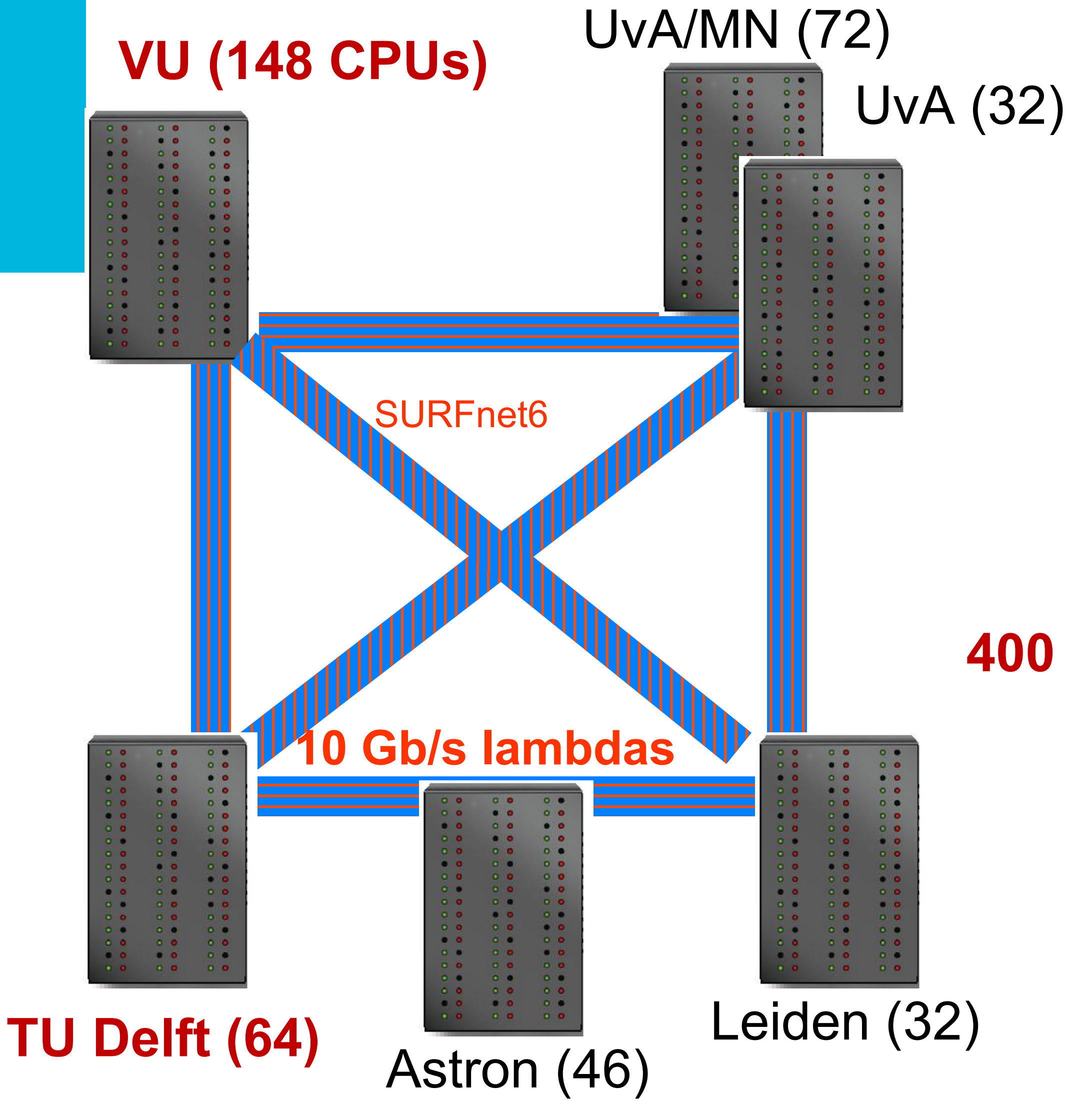
- **design models** by theoretical study and analysis
- **validate models** by implementation and experimentation

COMMIT/

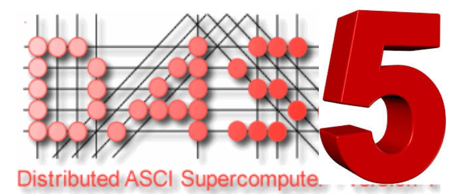
DELFT DATA SCIENCE

PDS Group

Our experimental testbed: DAS-4



10+ years of system research
300+ scientists as users

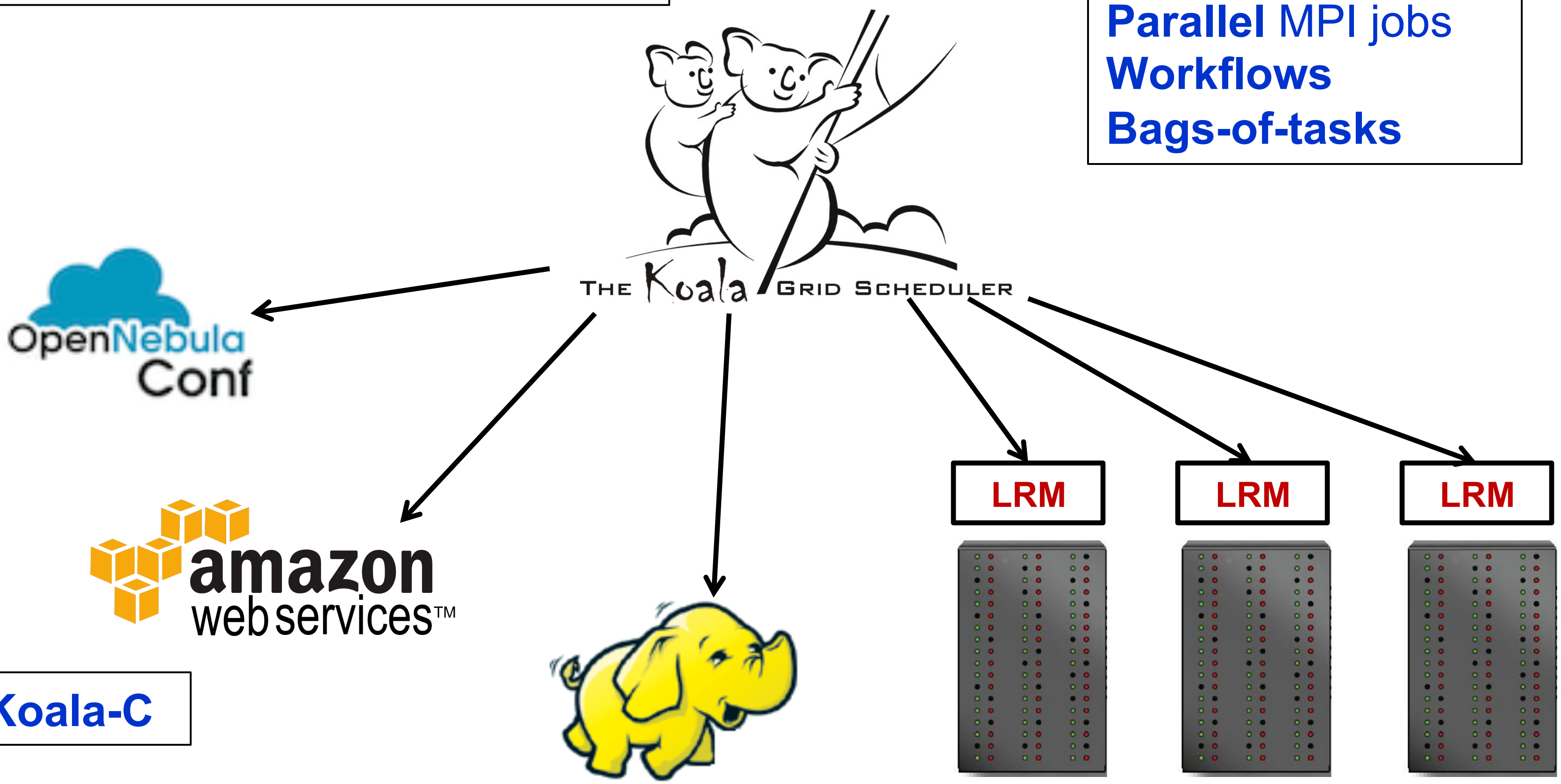


400 ~~200~~ dual-quad-core compute nodes
24 GB memory per node
150 TB total storage
20 Gpbs ~~QDR~~ FDR InfiniBand network

The KOALA multicluster scheduler

Our research vehicle
Deployed on DAS since 2005

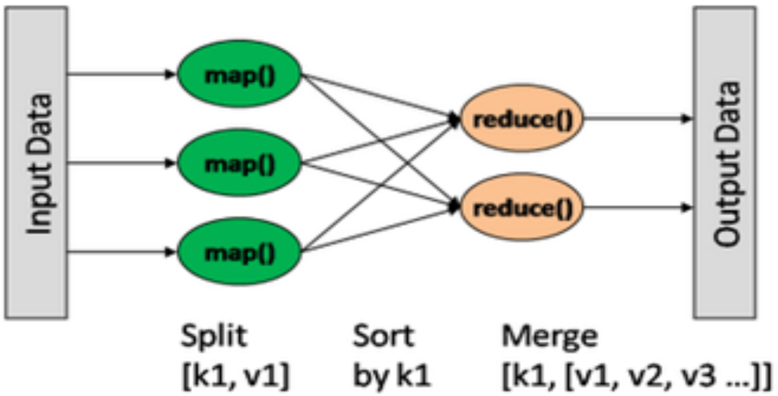
Parallel MPI jobs
Workflows
Bags-of-tasks



Koala-C

This talk: MapReduce frameworks

Hadoop and MapReduce



MapReduce

- Two-phase processing
- Data locality constraints
- Inter-task dependency

Open-source software

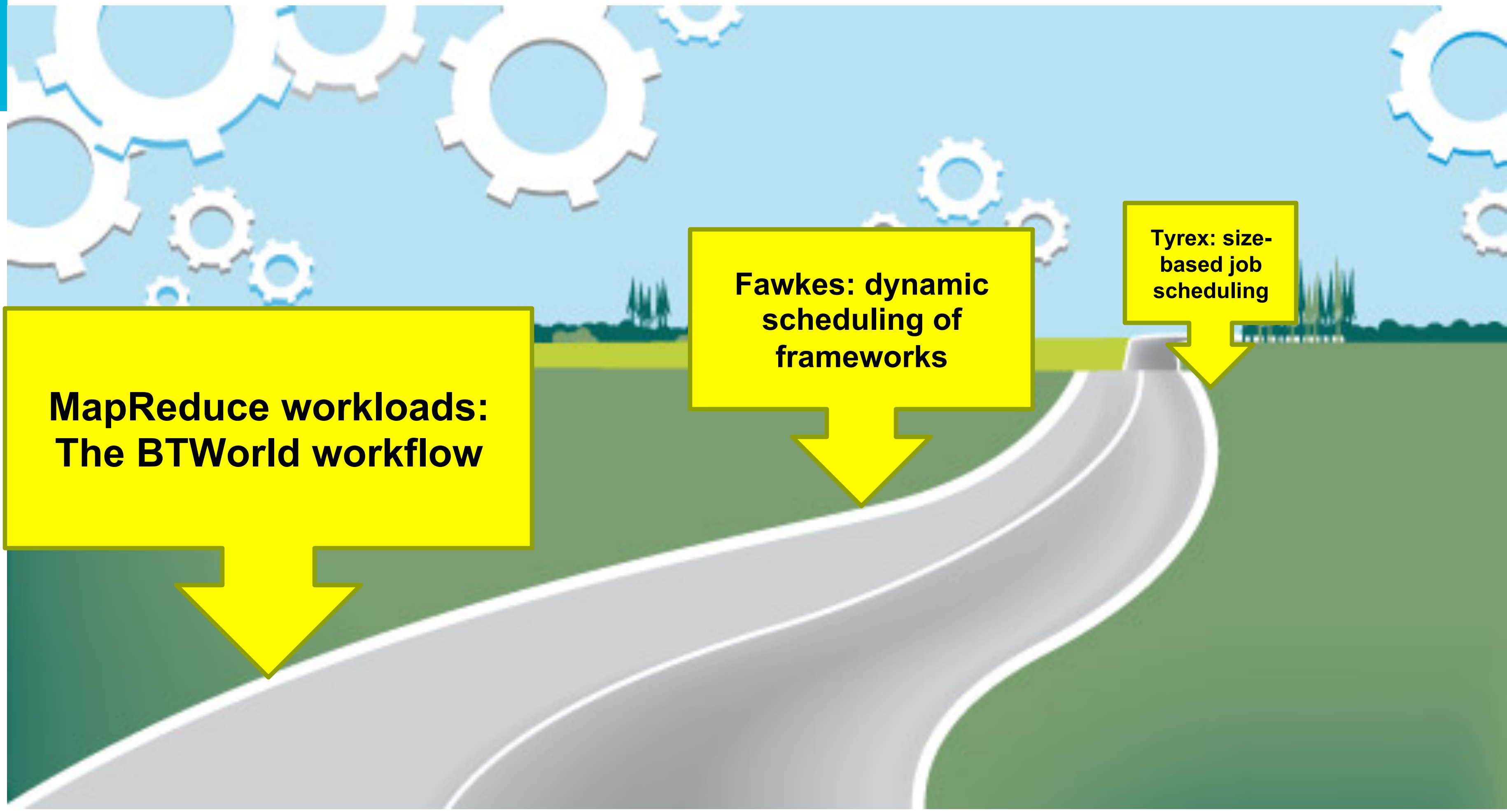
- **HDFS** – high-throughput access to data
- **MapReduce** – parallel data processing
- **Yarn –** cluster resource management

Distributed ASCI Supercomputer - Version 4

In our experiments

- 6 map slots vs. 2 reduce slots
- 128 MB per data block
- 3 replicas of each data block
- 3 GB memory per task
- InfiniBand network

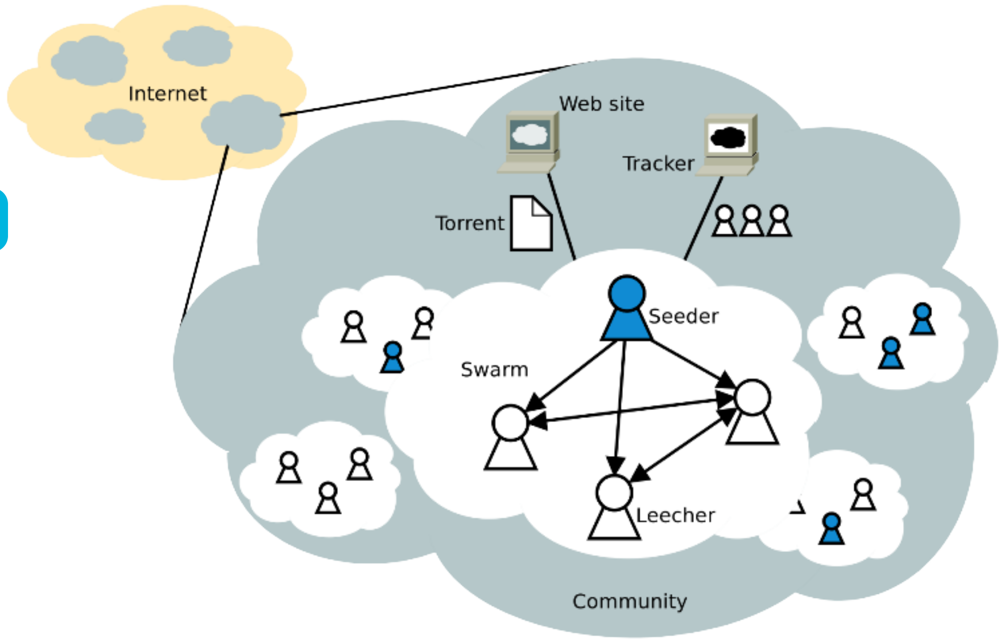
Roadmap


MapReduce workloads: The BTWorld workflow
Fawkes: dynamic scheduling of frameworks
Tyrex: size-based job scheduling

The BTWorld project: a typical big data use case (1/2)

BitTorrent
- Most used protocol on Internet
- Over 100 million users

	Upstream	
Rank	Application	Share
1	BitTorrent	48.10%
2	YouTube	7.12%
3	HTTP	5.74%
4	Skype	4.96%
5	Facebook	3.54%
6	Netflix	2.83%
7	SSL	2.47%
8	eDonkey	1.12%
9	Dropbox	1.12%
10	RTMP	0.85%
		77.83%

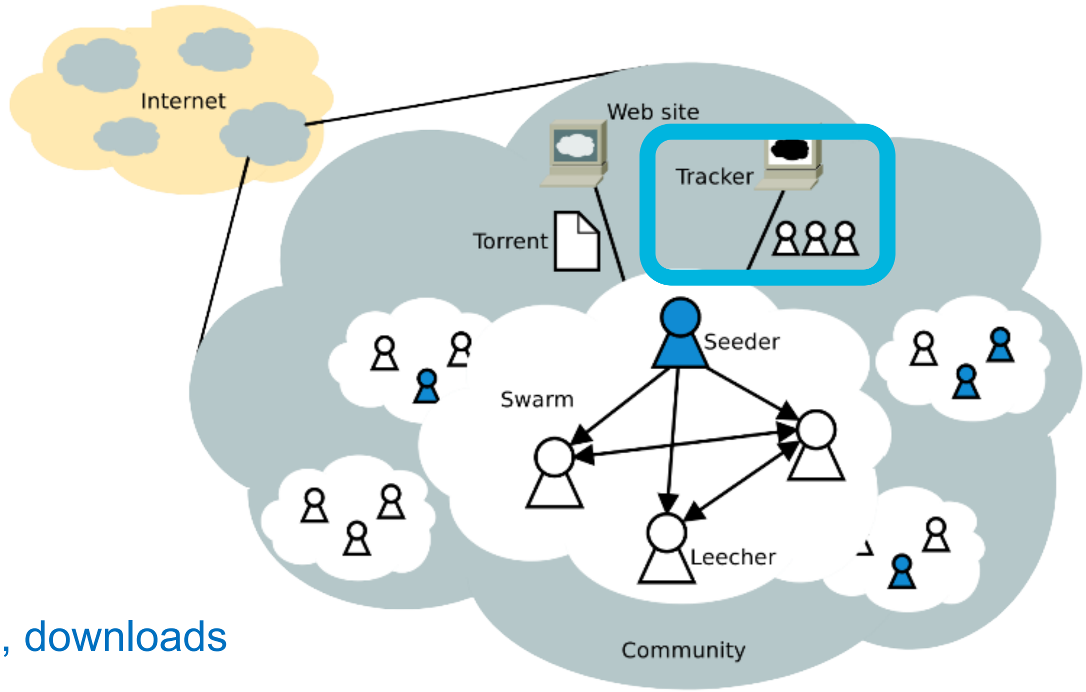


The BTWorld project: a typical big data use case (2/2)

Our approach
- monitor servers not users

Collected data
- over 15 TB since 2009
- 1 file / tracker / sample

Multi-record files
- timestamp: logging time
- hash: unique id for content
- tracker: unique id for server
- info per file: seeders, leechers, downloads

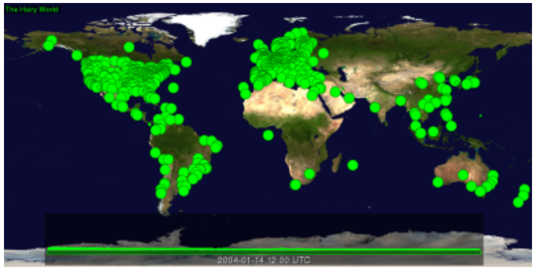


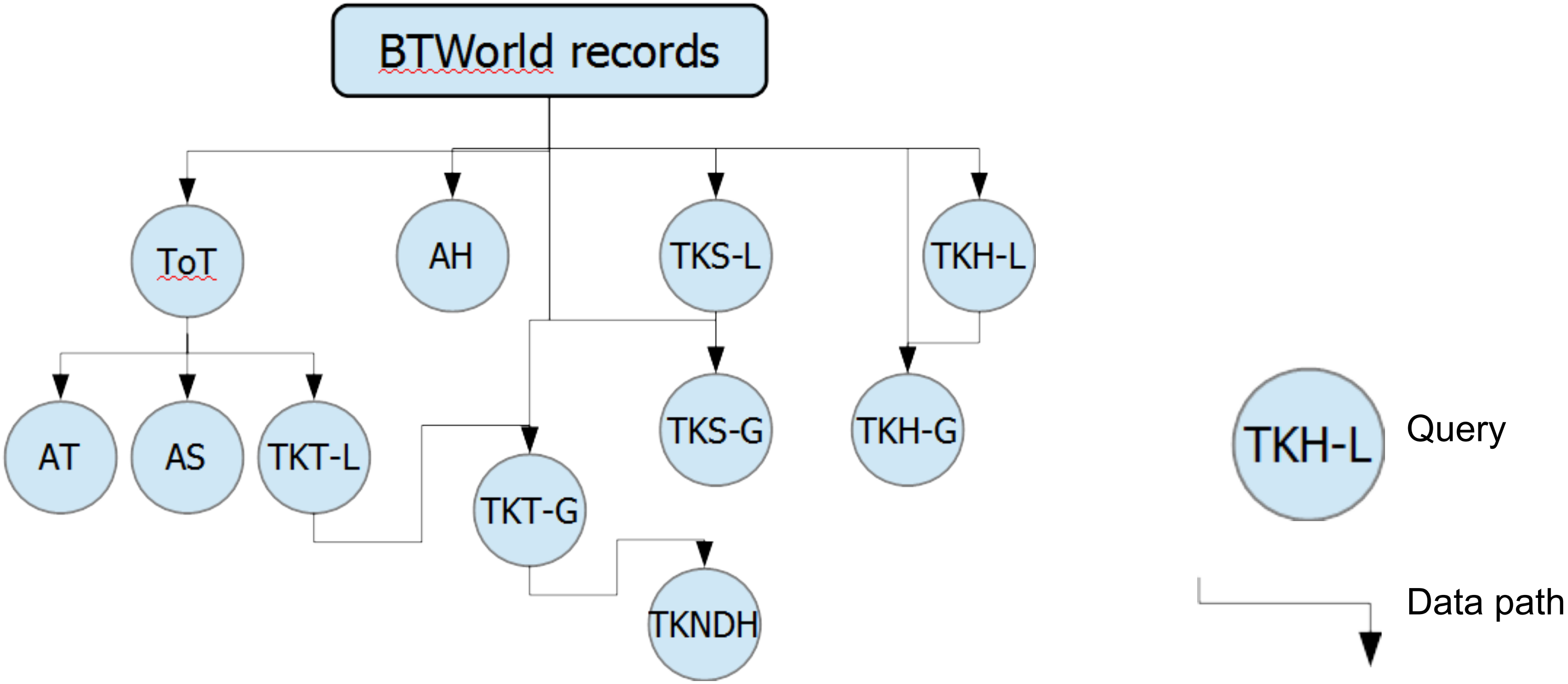
M. Wojciechowski, M. Capota, J. Pouwelse, A. Iosup, "Towards observing the global BitTorrent file-sharing network", ACM HPDC 2010

The BTWorld workflow (1/4)

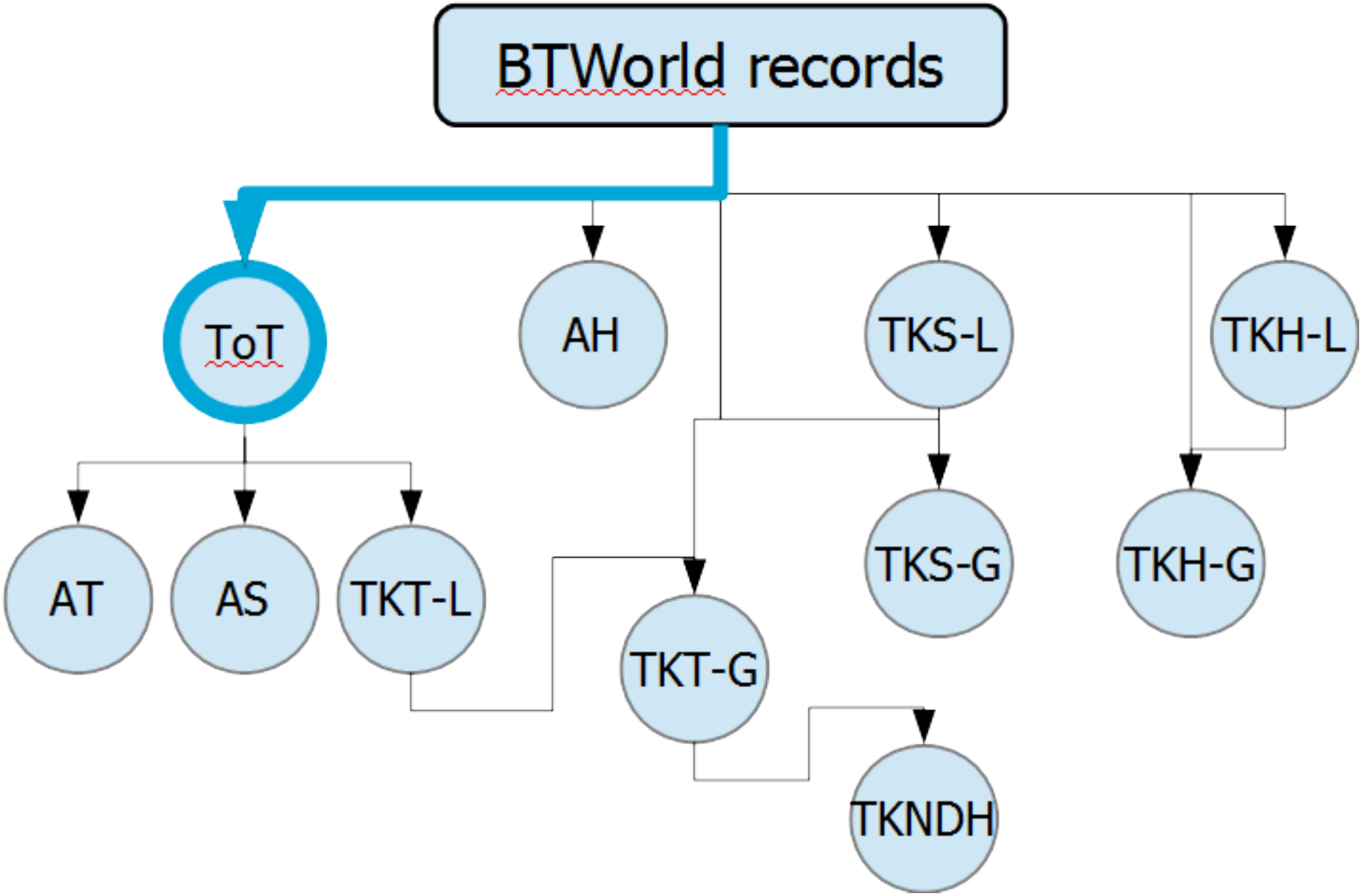
Analyst questions

- How does the number of peers evolve over time?
- How long are files available?
- Did the legal bans and tracker take-downs impact BT?
- How does the location of trackers evolve over time?

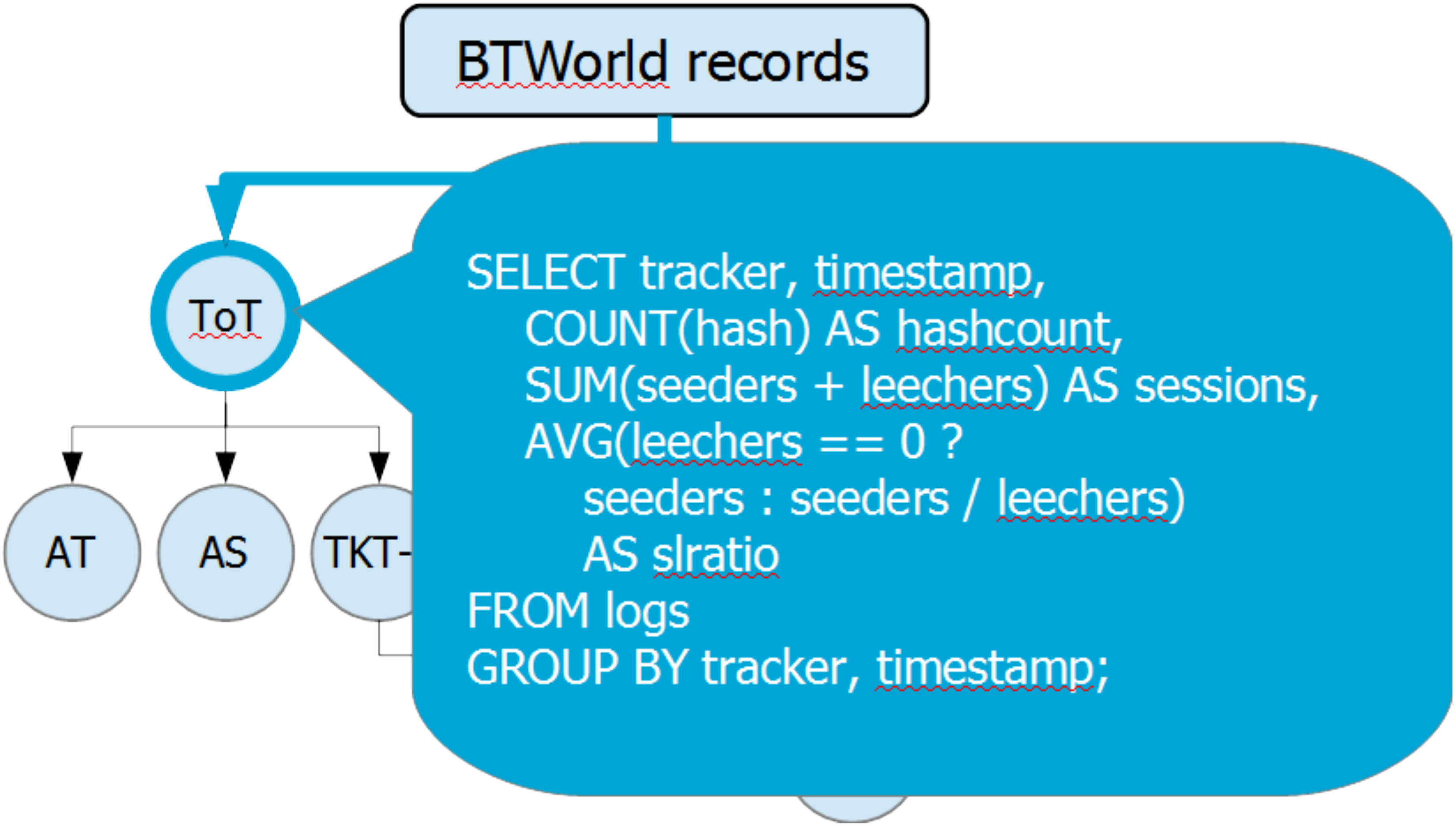




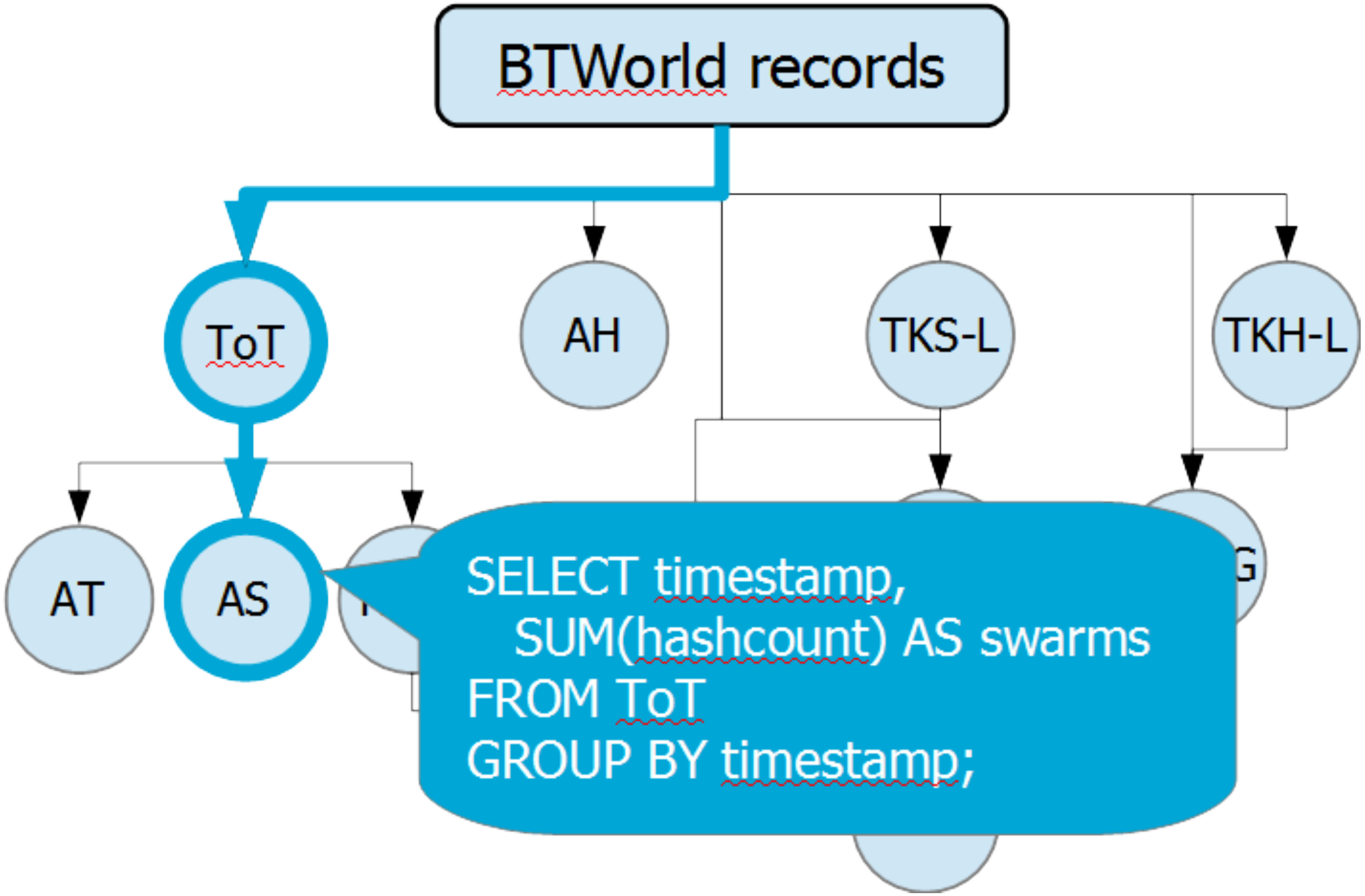
The BTWorld workflow (2/4)



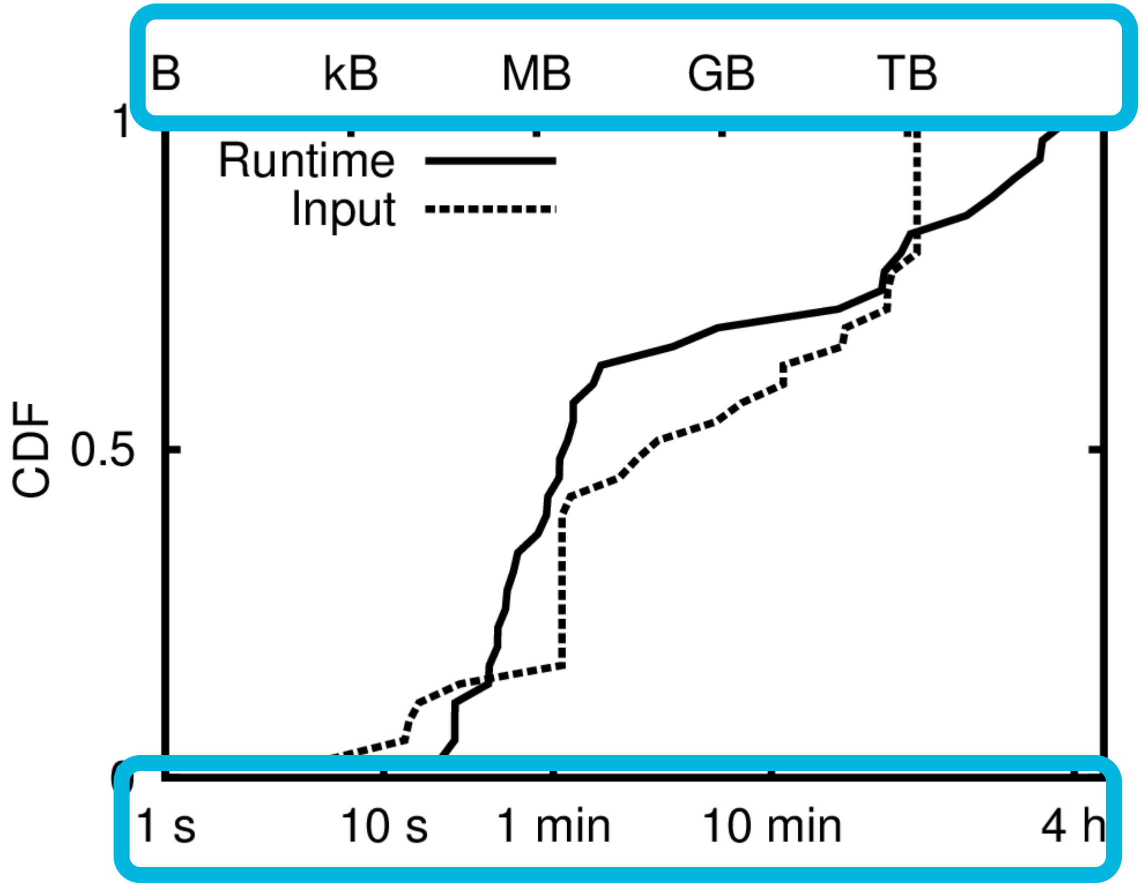
The BTWorld workflow (3/4)



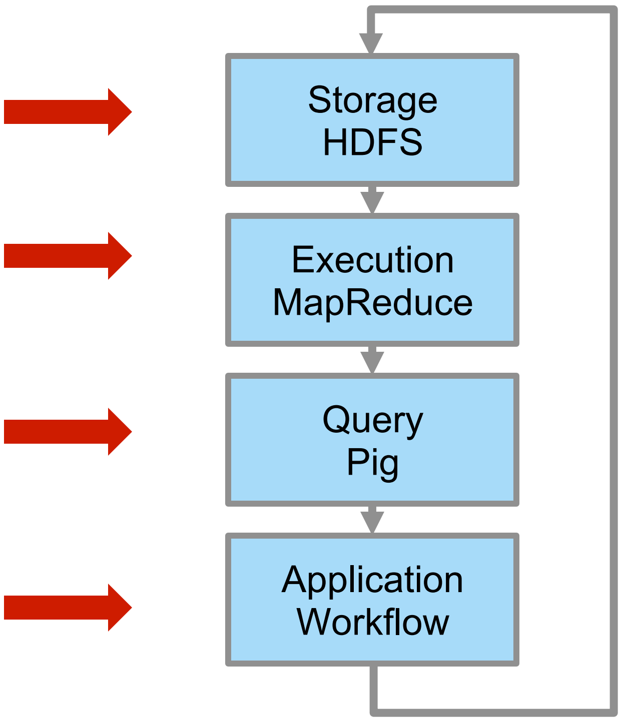
The BTWorld workflow (4/4)



Variety in job input size and job runtime



Platform optimisations



HDFS

- Data pre-processing
- Reduced replication

MapReduce

- Task memory versus number of tasks
- Stalled reduce execution

Pig

- Not enough operators
- Adaptive scheduling of reduce tasks

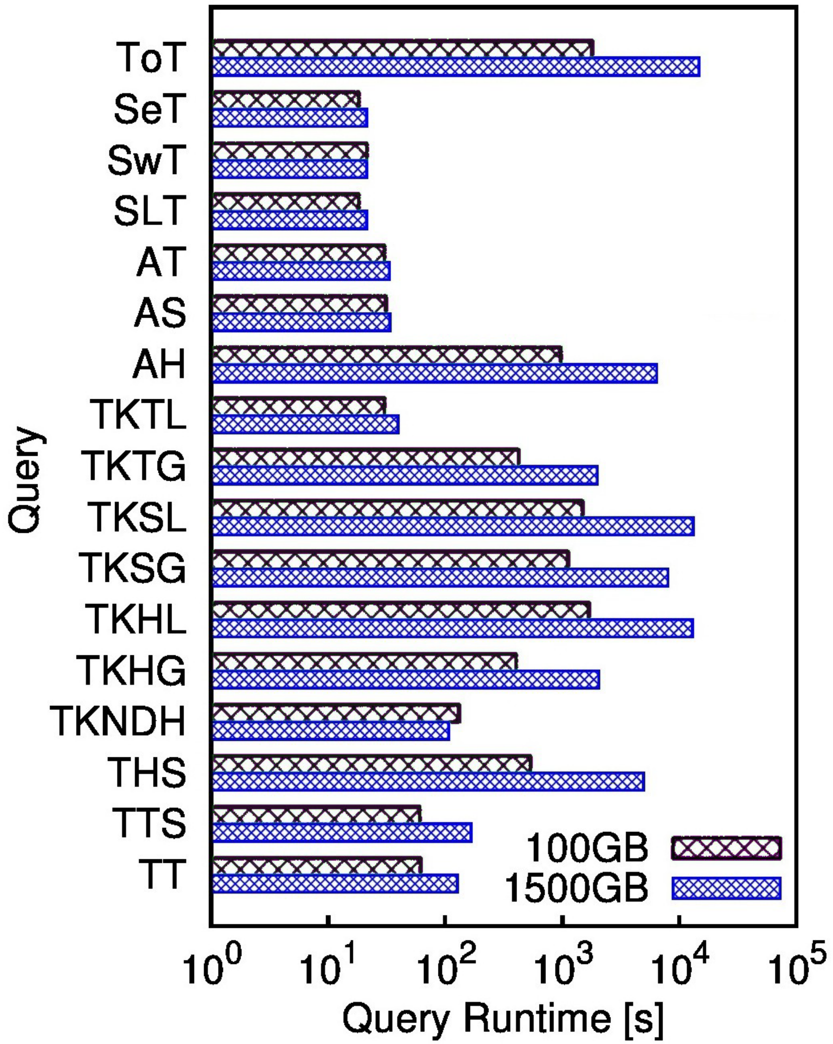
Workflow

- Reuse intermediary data
- Extract execution patterns

B.I. Ghit, M. Capota, T. Hegeman, D.H.J. Epema, A. Iosup, "V for Vicissitude: The Challenge of Scaling Complex Big Data Workflows" , **winner SCALE Challenge** at CCGrid 2014.

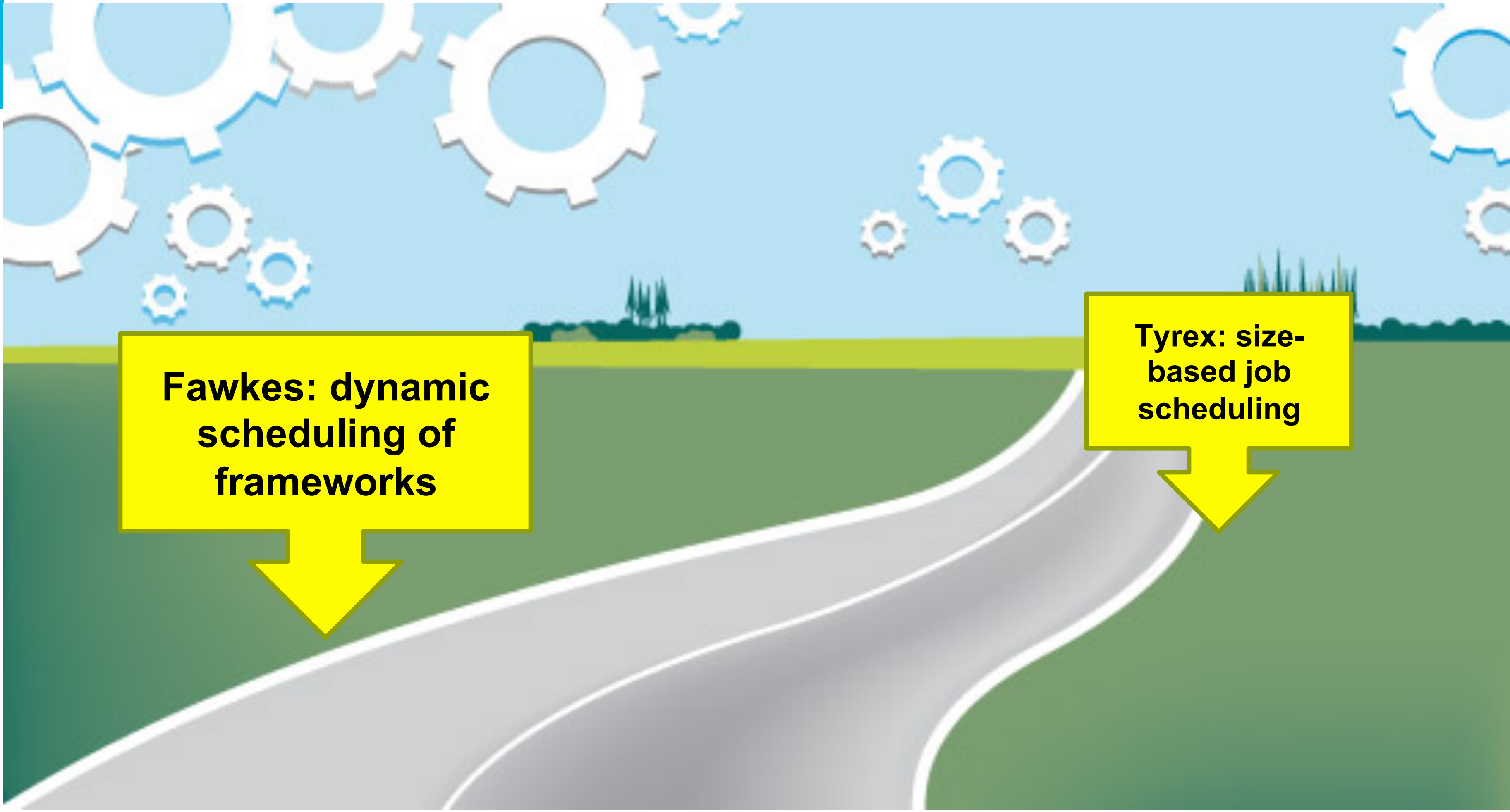
Long versus short

Nodes	24
Map slots	92
Reduce slots	92
Memory per task	6 GiB
Total memory	552 GiB
HDFS replication	2



Short queries are relatively scale-free
Long queries do not scale linearly

Roadmap



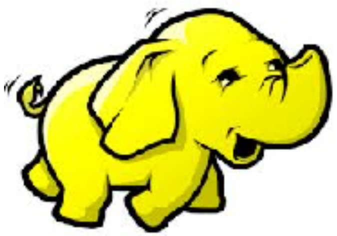
Fawkes: dynamic scheduling of frameworks
Tyrex: size-based job scheduling

Why multiple frameworks?

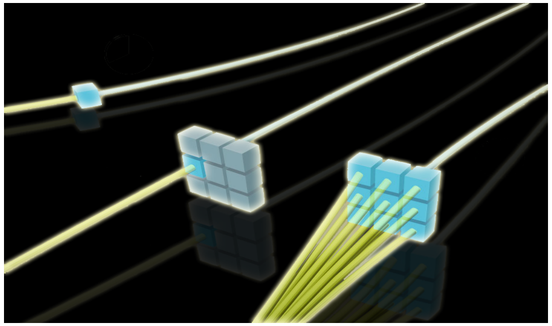
Data isolation

Failure isolation

Version isolation



Performance isolation

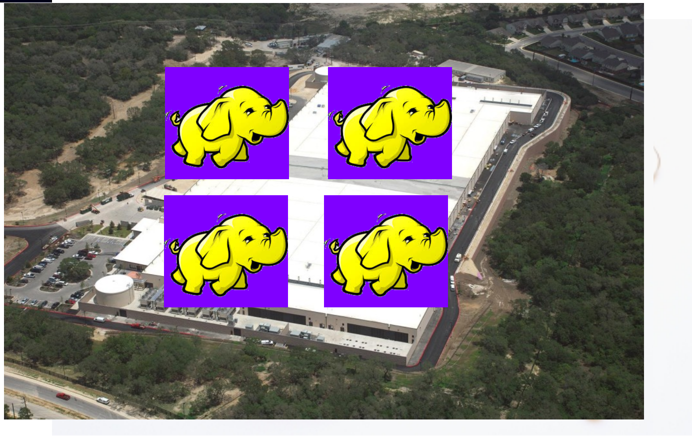


Appealing for companies and users
Difficult to **achieve** and to **define**

The “big data cake” in datacenters

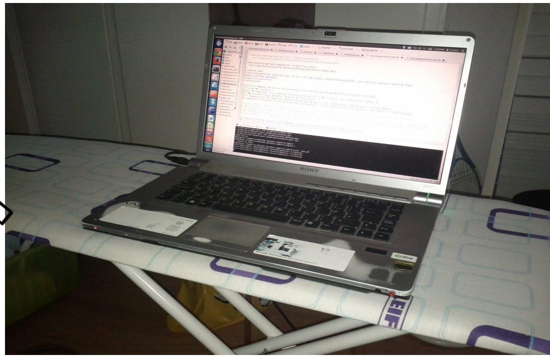
Online Social Networks

Financial Analysts



Universe Explorers

Big Data Enthusiast



Scheduling Frameworks (1/2)

Monolithic schedulers:

- **single, centralized** scheduling algorithm

Allocate resources to frameworks

THE Koala GRID SCHEDULER

Two-level schedulers:

- **lower** scheduling overhead
- **flexibility** and **parallelism**
- **isolation** among frameworks
- **transparent** job submission

Internal job scheduling

Framework 1

Framework 2

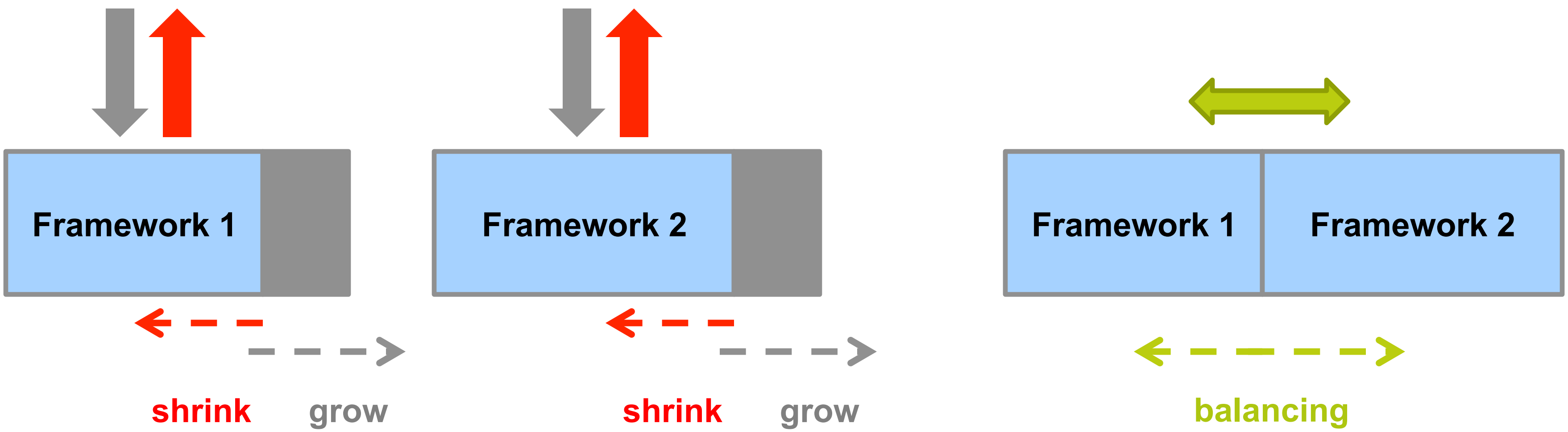
Scheduling Frameworks (2/2)

Resource offers

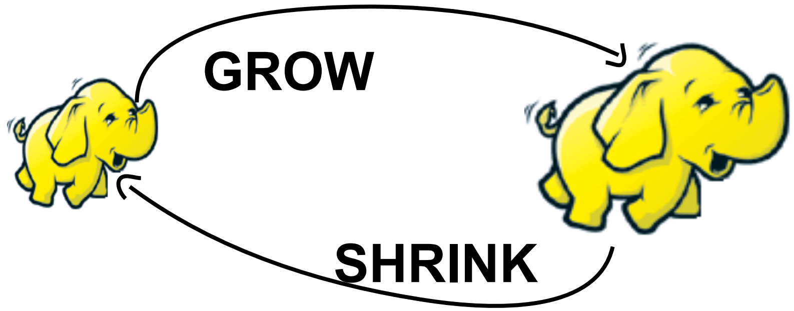
- Frameworks accept/reject offers
- Best-effort strategy

Resource balancing

- Koala has access to the global state
- Koala finds the optimal configuration

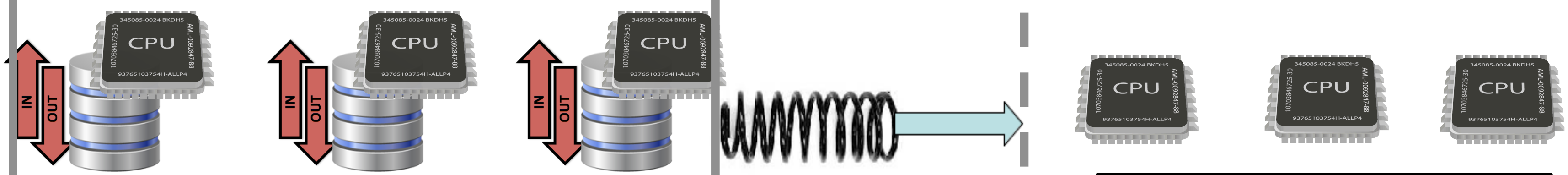


Resizing MapReduce: no data locality



Core nodes

Transient nodes (TR)



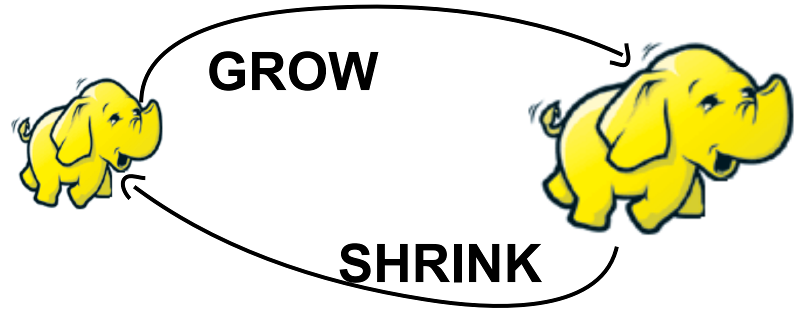
INPUT/OUTPUT DATA

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

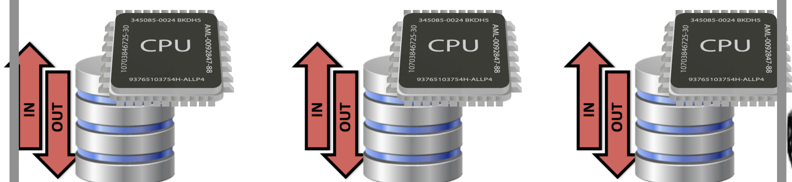
NO DATA

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

Resizing MapReduce: relaxed data locality

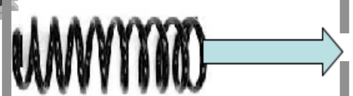


Core nodes

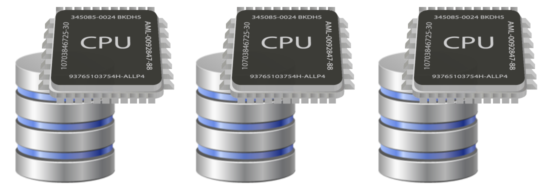


INPUT/OUTPUT DATA

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



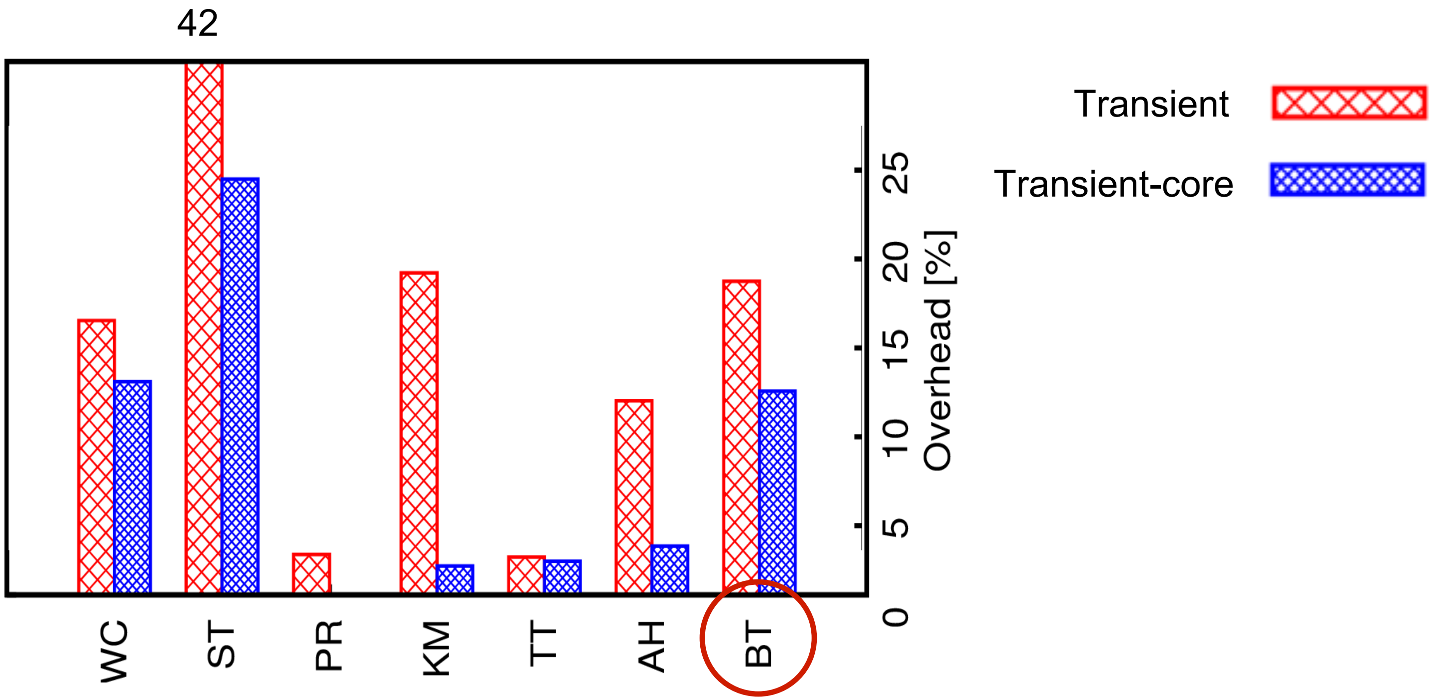
Trans-core nodes (TC)



OUTPUT DATA

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

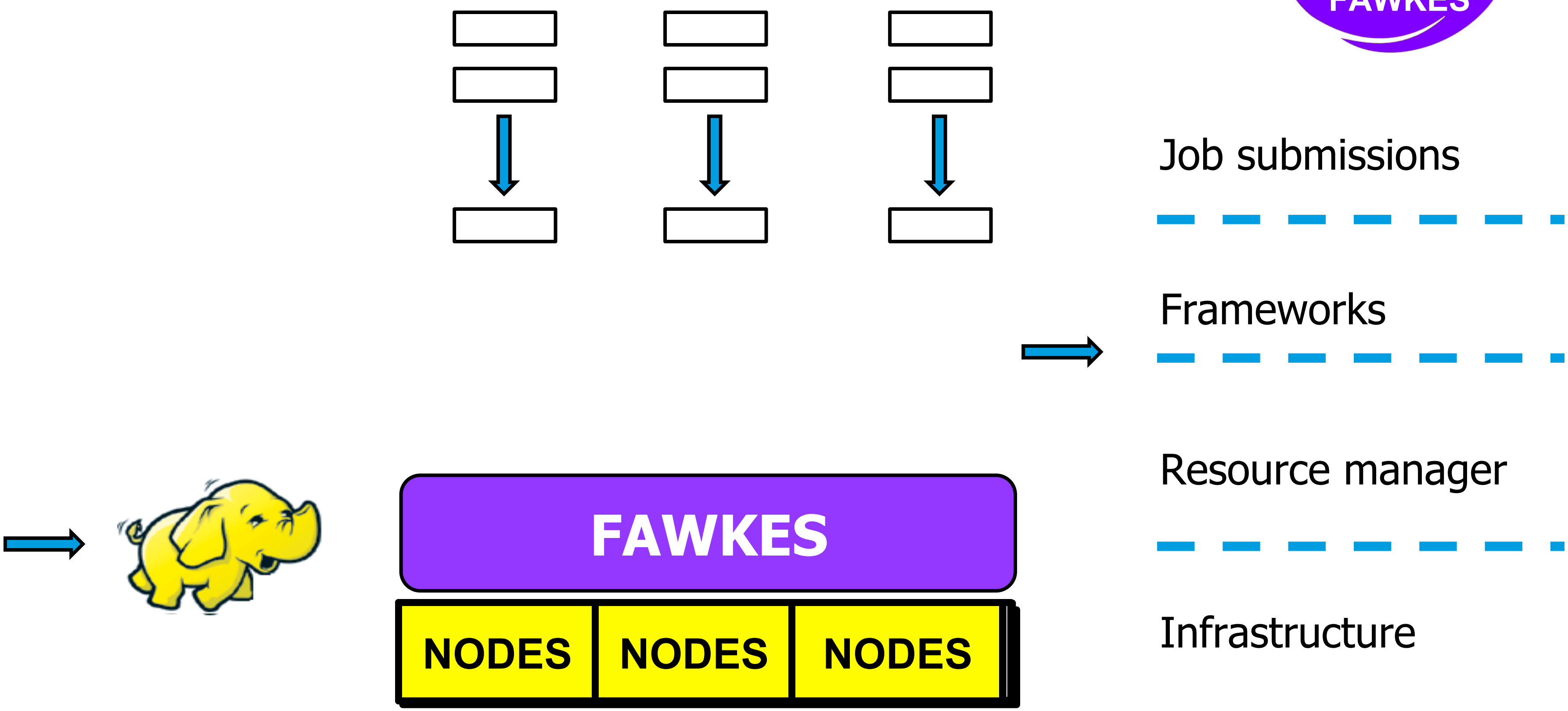
Performance of no versus relaxed data locality



Complete workflow on 100 GB

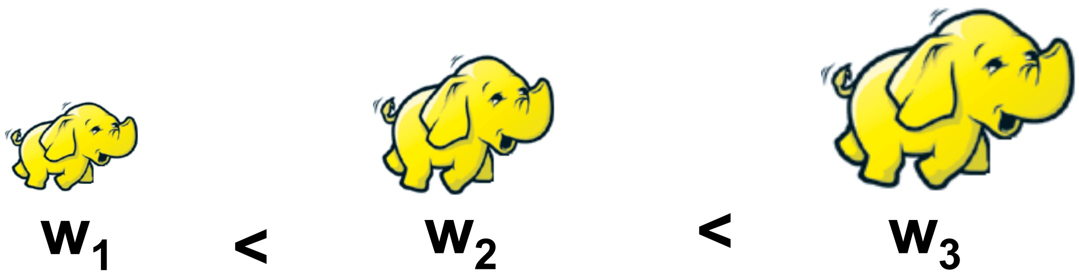
- Single-application performance overhead
- 10 core nodes + 10 transient/transient-core nodes

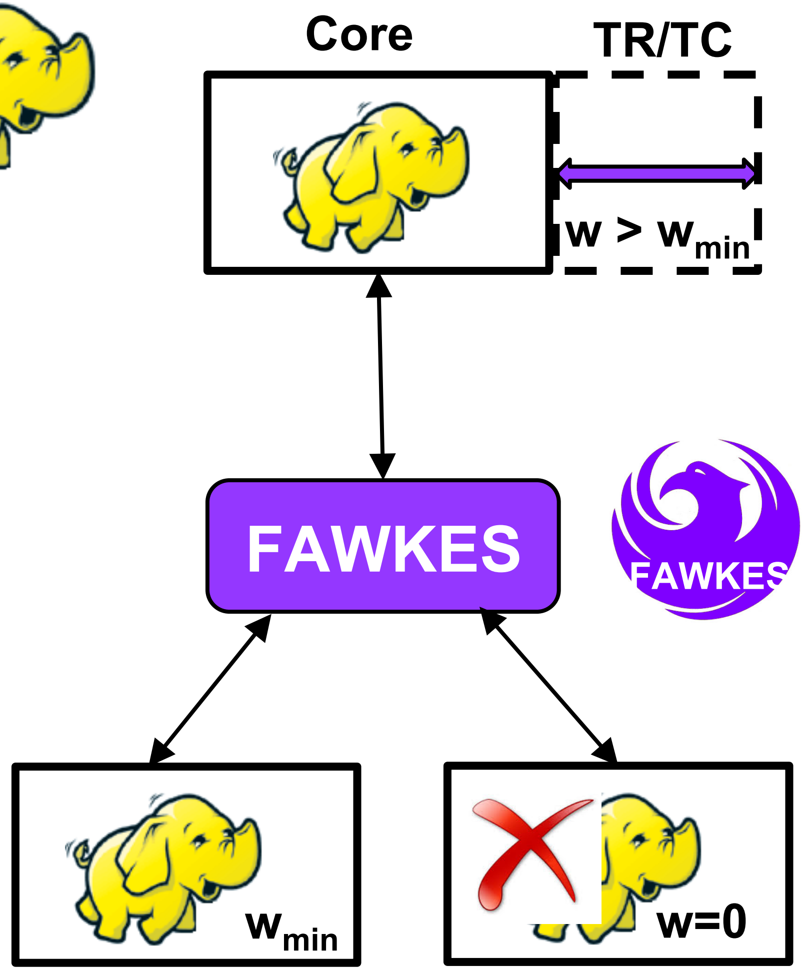
Dynamic scheduling with FAWKES



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

Balancing Allocations with FAWKES

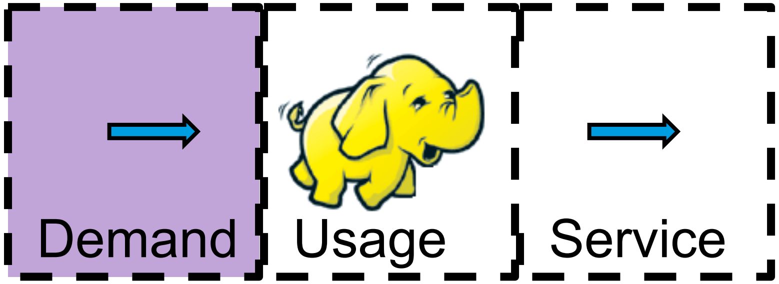


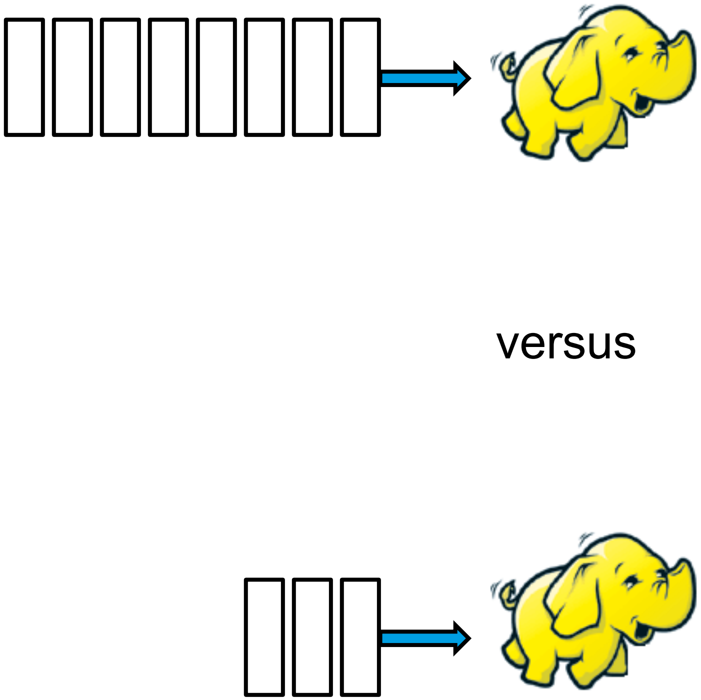


1. Updates dynamic weights when:
   - new frameworks arrive
   - framework states change

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

How to differentiate frameworks (1/3)

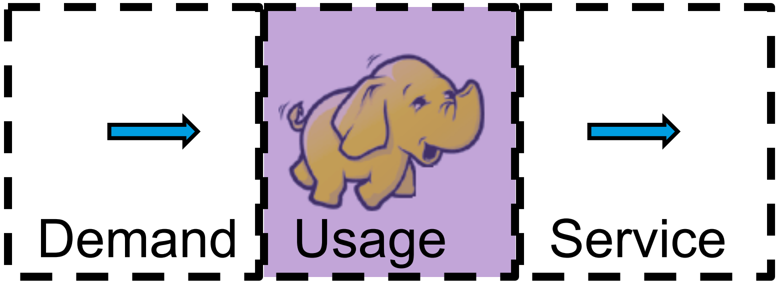




By demand – 3 policies:

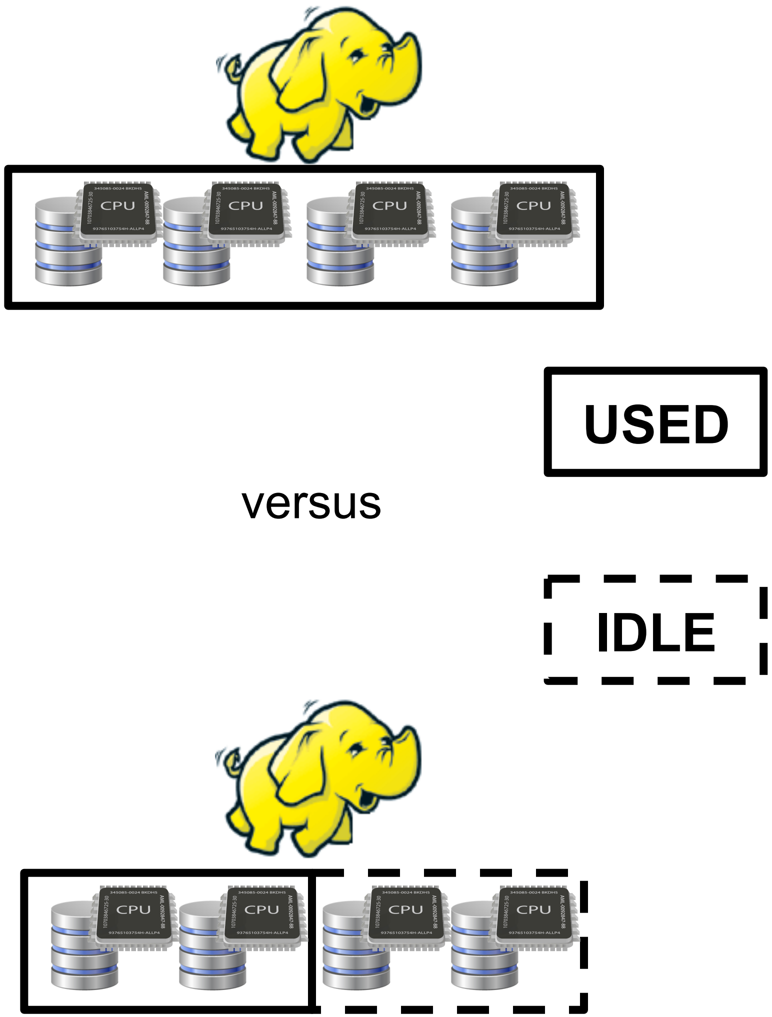
- Job Demand (JD)
- Data Demand (DD)
- Task Demand (TD)

How to differentiate frameworks (2/3)

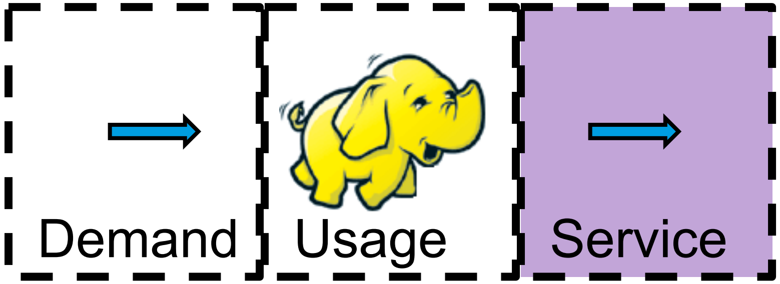


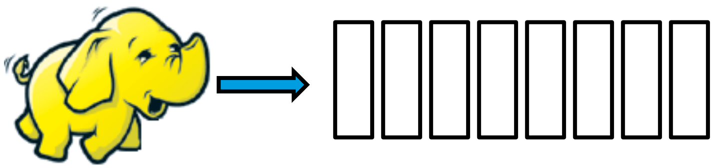
By usage – 3 policies:

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

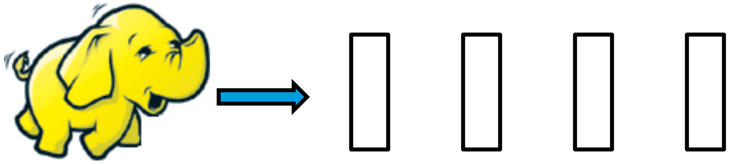


How to differentiate frameworks (3/3)





versus



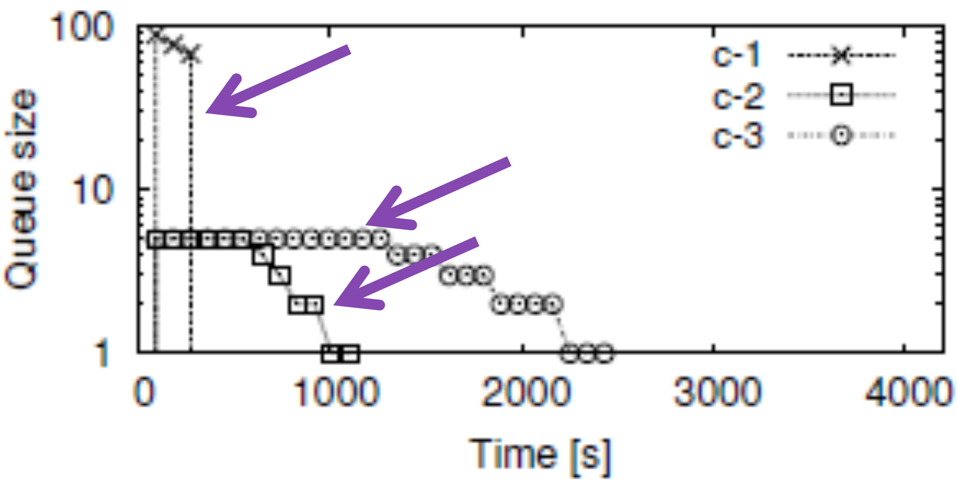
By service – 3 policies:

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

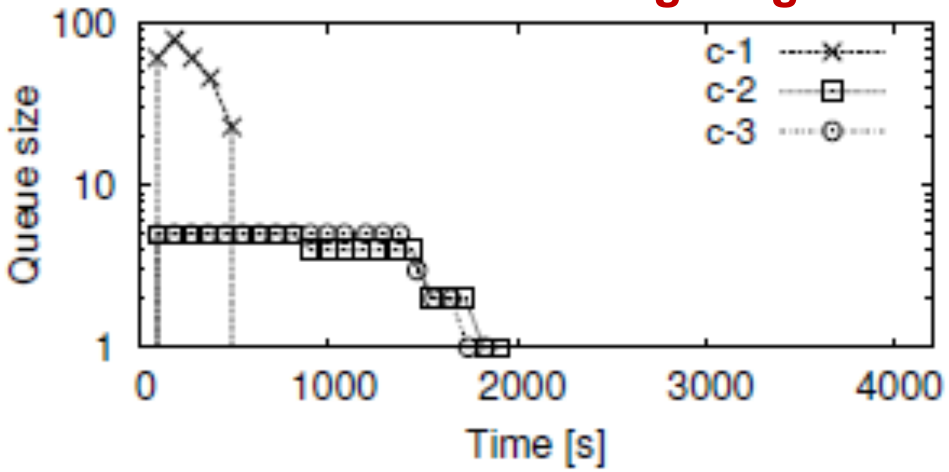
Performance of FAWKES (1/2)

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

FAWKES with static allocation



FAWKES with TD weighting

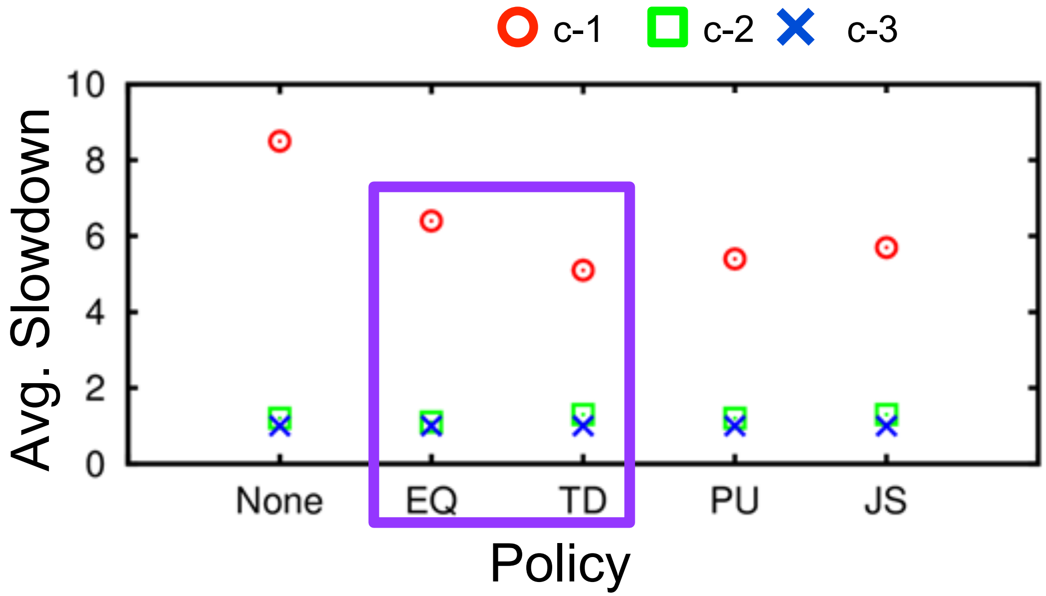


Closed system

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

Performance of FAWKES (2/2)

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



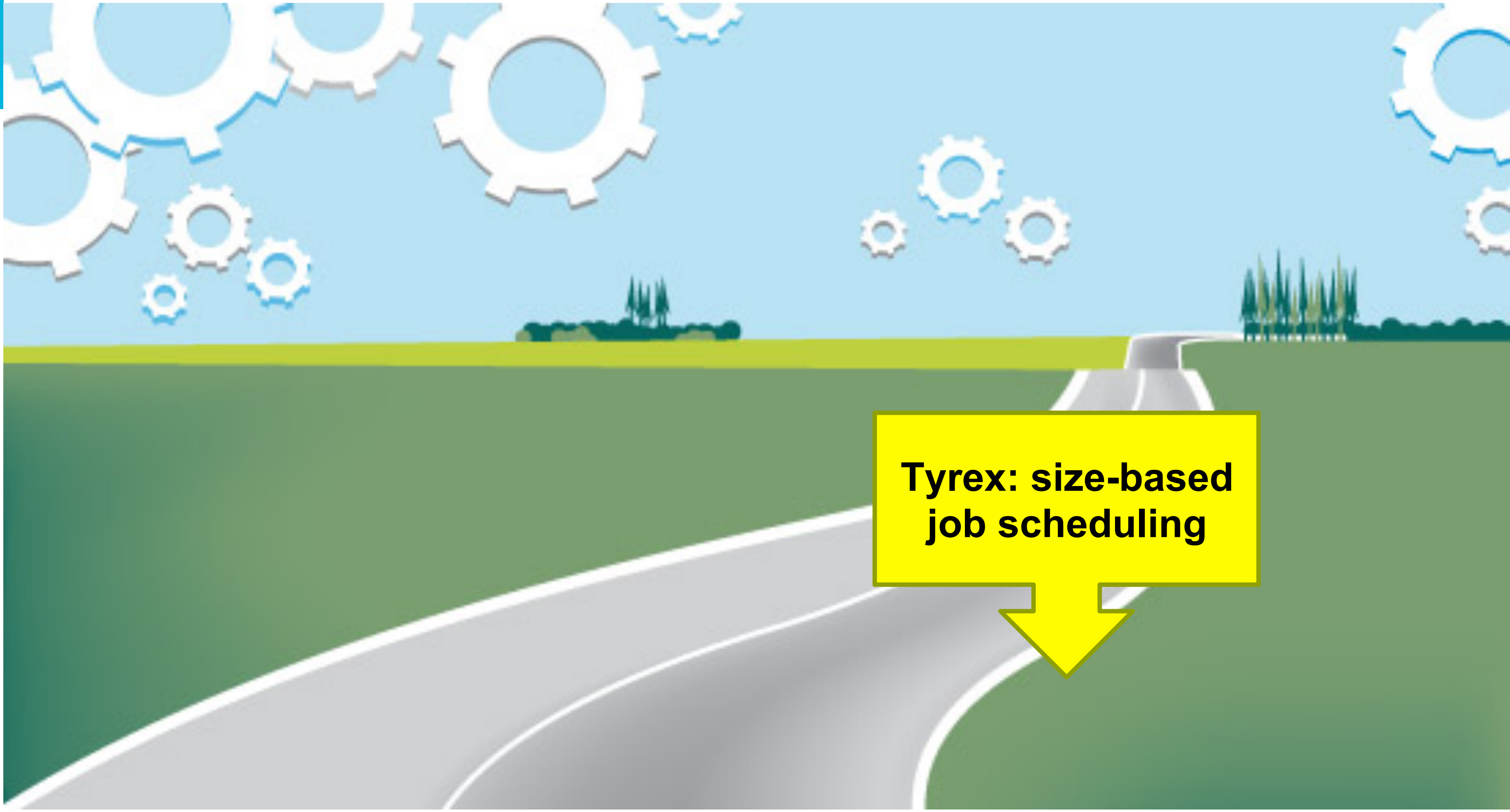
Open system

- 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

Roadmap

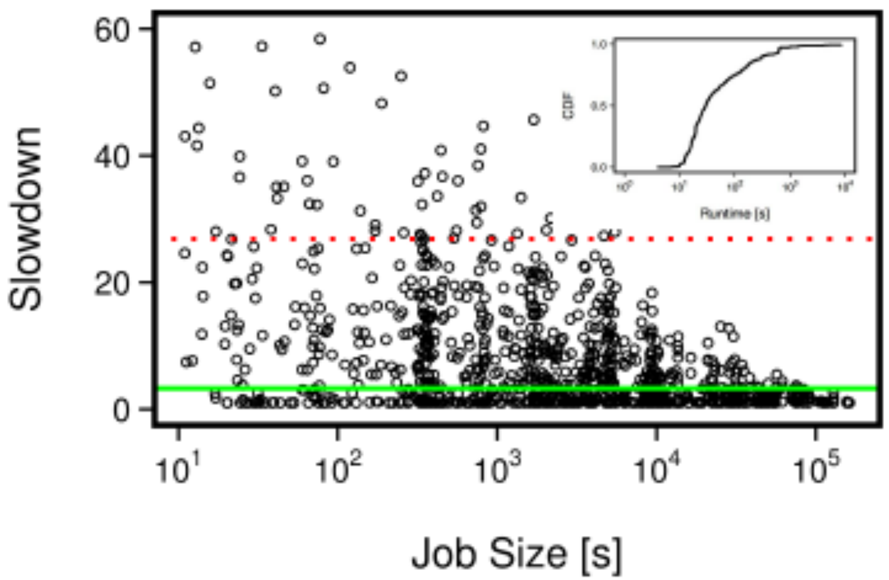


Job scheduling in MapReduce

MapReduce workloads

- skewed job size distributions
- high job size variability
- short jobs prevail, but long jobs dominate
- challenging for existing schedulers

FIFO with a Facebook trace

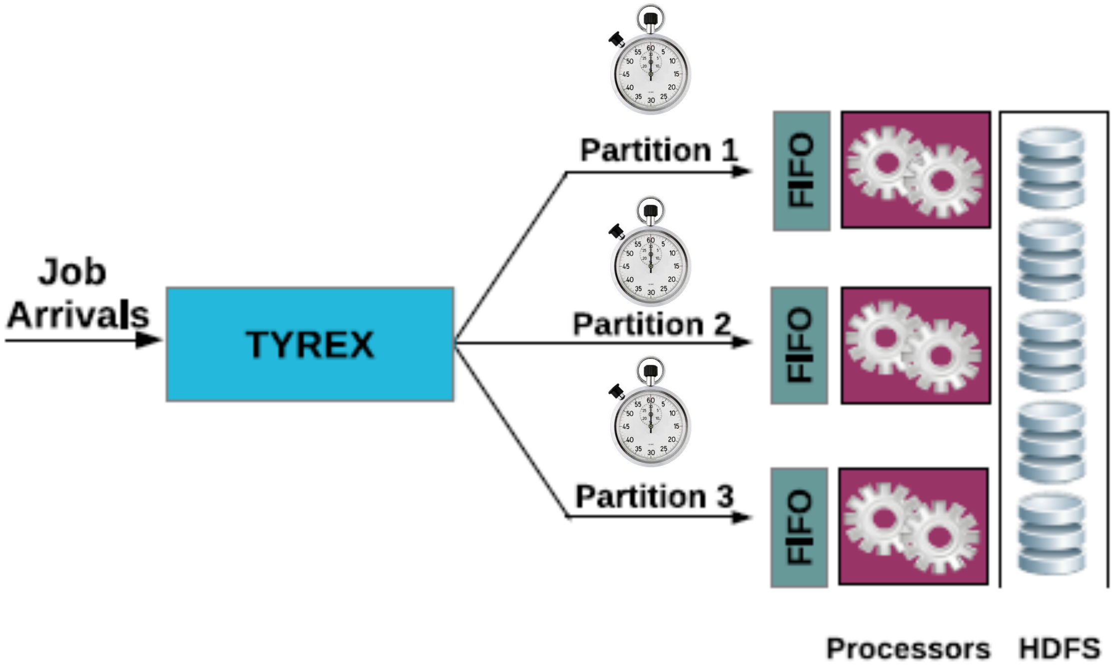


We need some form of isolation within a single framework

Size-based scheduling with Tyrex

Based on TAGS policy for distributed servers

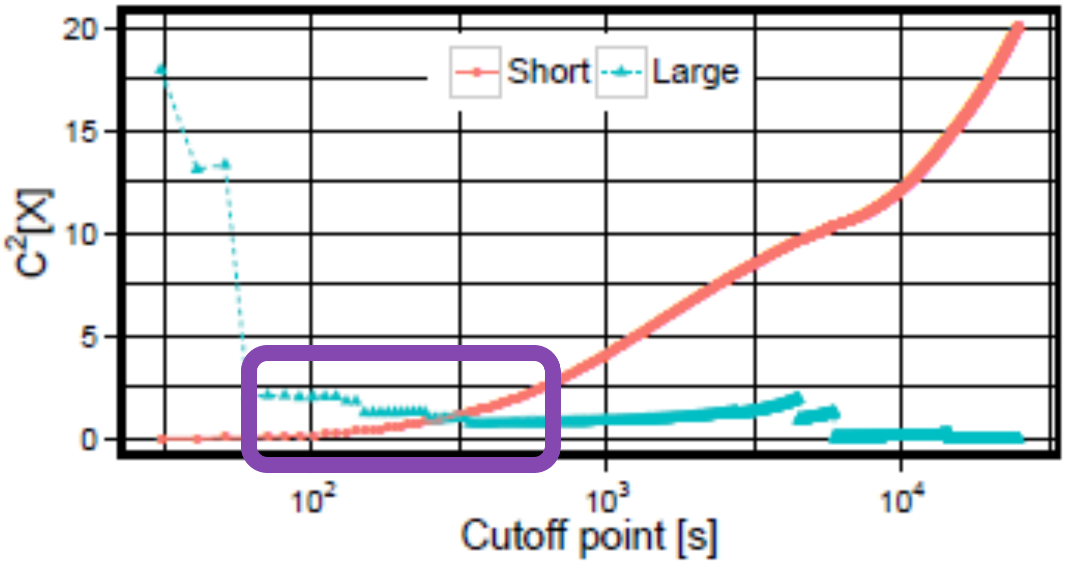
- Partition capacities
- Elastic parallel jobs
- No killing



Dynamic timers with Tyrex

Optimal timers

- no closed forms
- complex expressions for Pareto distributions
- significant human effort to find them
- may change over time



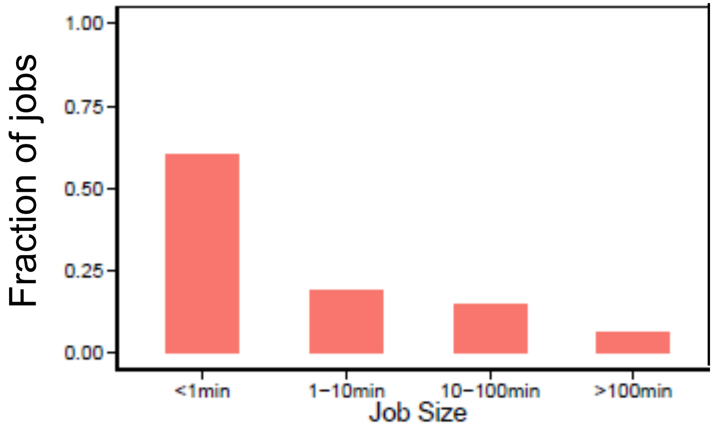
The SplitTAGS policy

Partial job size = sum of completed task runtimes
Remaining job size = non-completed tasks

Find the timer to minimize the maximum variability

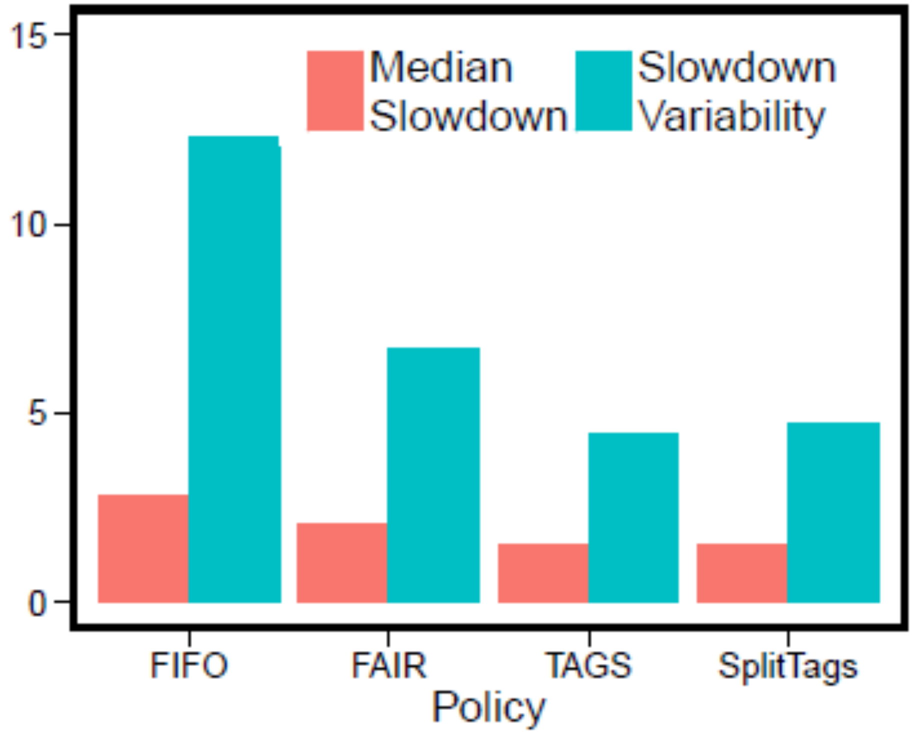
- SCV of remaining job sizes
- Preempt all jobs with **partial sizes** larger than the timer

Experimental setup



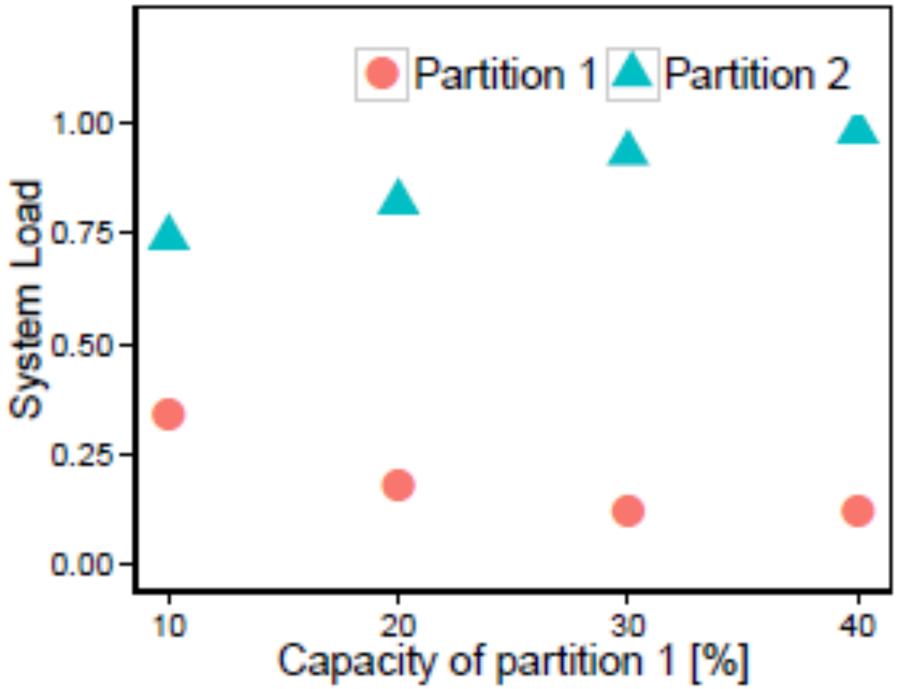
Statistics	HVW	MVW	LVW
Total jobs		300	
BTWORLD jobs	33	45	10
Total maps	6,139	11,866	30,576
Total reduces	788	1,368	3,089
Temporary data [GB]	573	693	1,062
Persistent data [GB]	100	92	303
Total CPU time [h]	63.6	124.6	306.9
Total runtime [h]	3.51	3.98	5.31

Performance of Tyrex



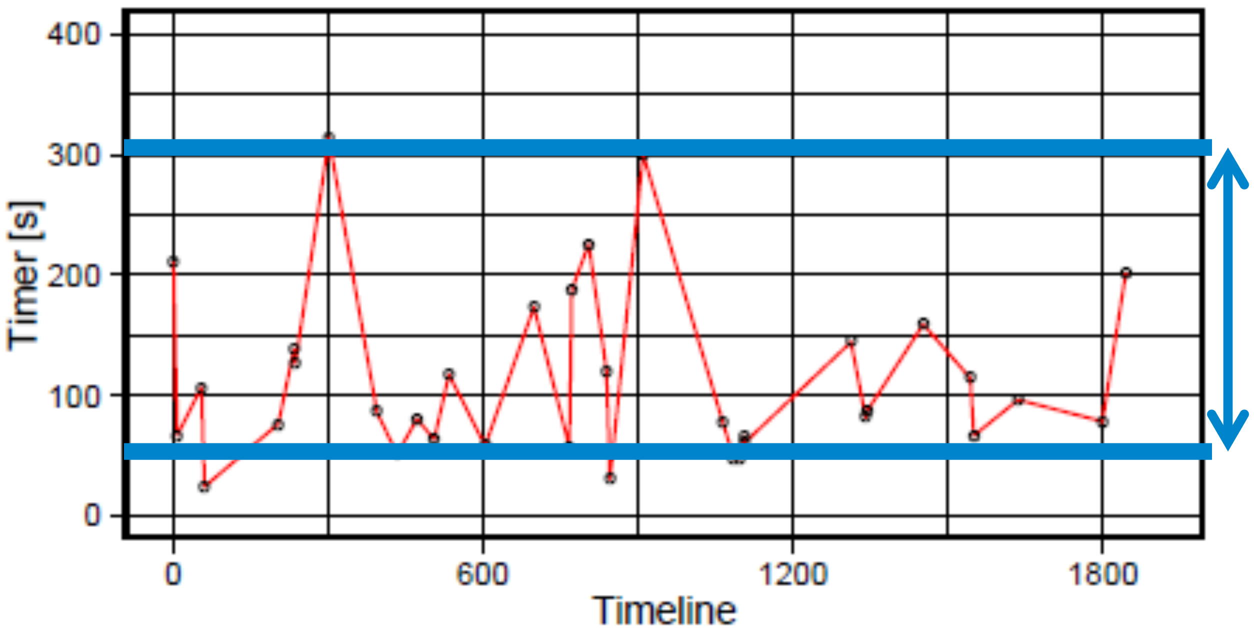
TAGS and SplitTags offer considerable improvements

To balance or not to balance?



Very low load conditions in partition 1

Stability of dynamic timers



1 change per minute at 70% load
Stays in the range of 50-300 seconds

Conclusions

FAWKES

BTWorld workflow
- benchmarking MapReduce systems
- representative for MapReduce workloads

Fawkes mechanism
- automatic deployment and elastic data-processing
- reduces the imbalance between frameworks

Tyrex scheduler
- job isolation by means of timers
- very good slowdown performance (with and without timers)

Distributed ASCI Supercomputer - Version 4

The Koala Grid Scheduler

Our research tag cloud

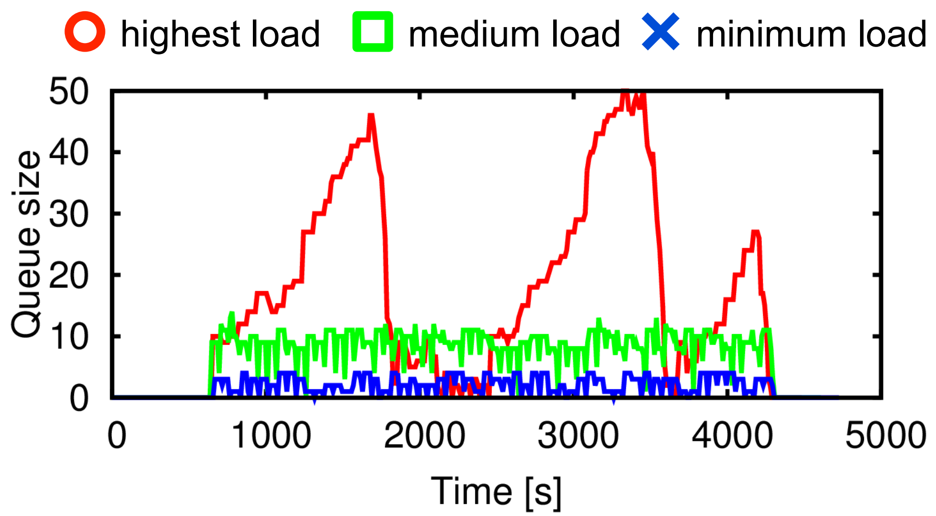
More information

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

Backup slides

FAWKES behind the scenes

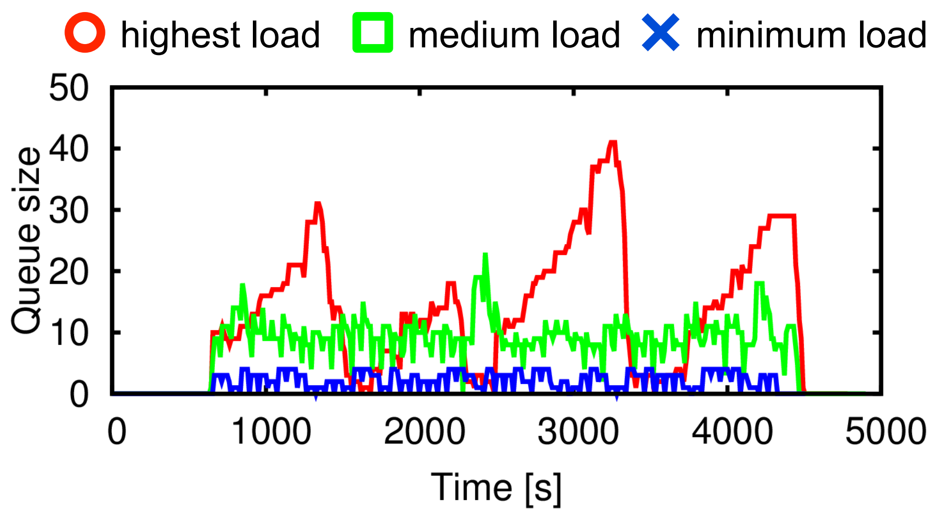
EQ



Utilizations: 60% / 23% / 5%

Imbalanced

TD



Utilizations: 50% / 30% / 8%

More balanced

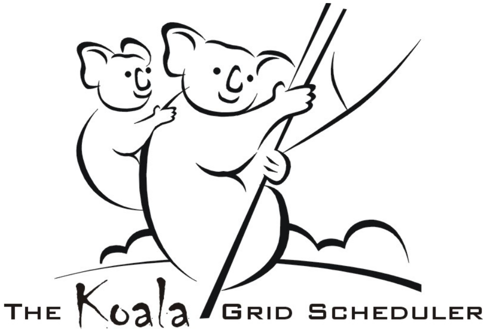
Contrasting the frameworks

- Resource requests from applications
- Capacity and Fair schedulers

FAWKES uses feedback from system operation

- Resource offers to frameworks
- No fairness guarantees

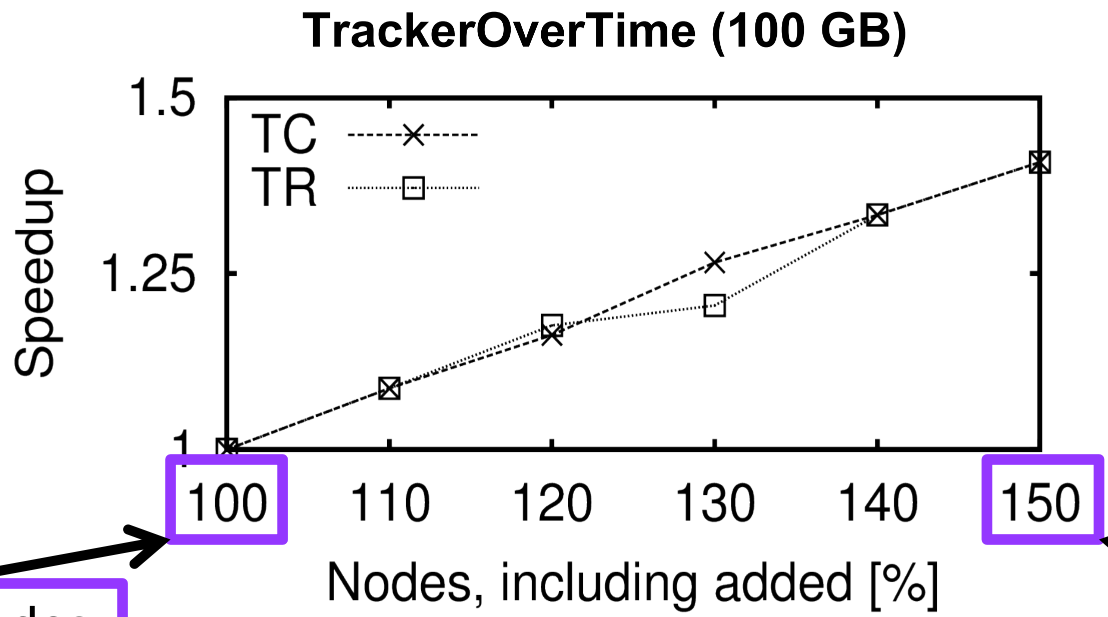
FAWKES schedules frameworks automatically



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

FAWKES is a research prototype

Speedup of growing

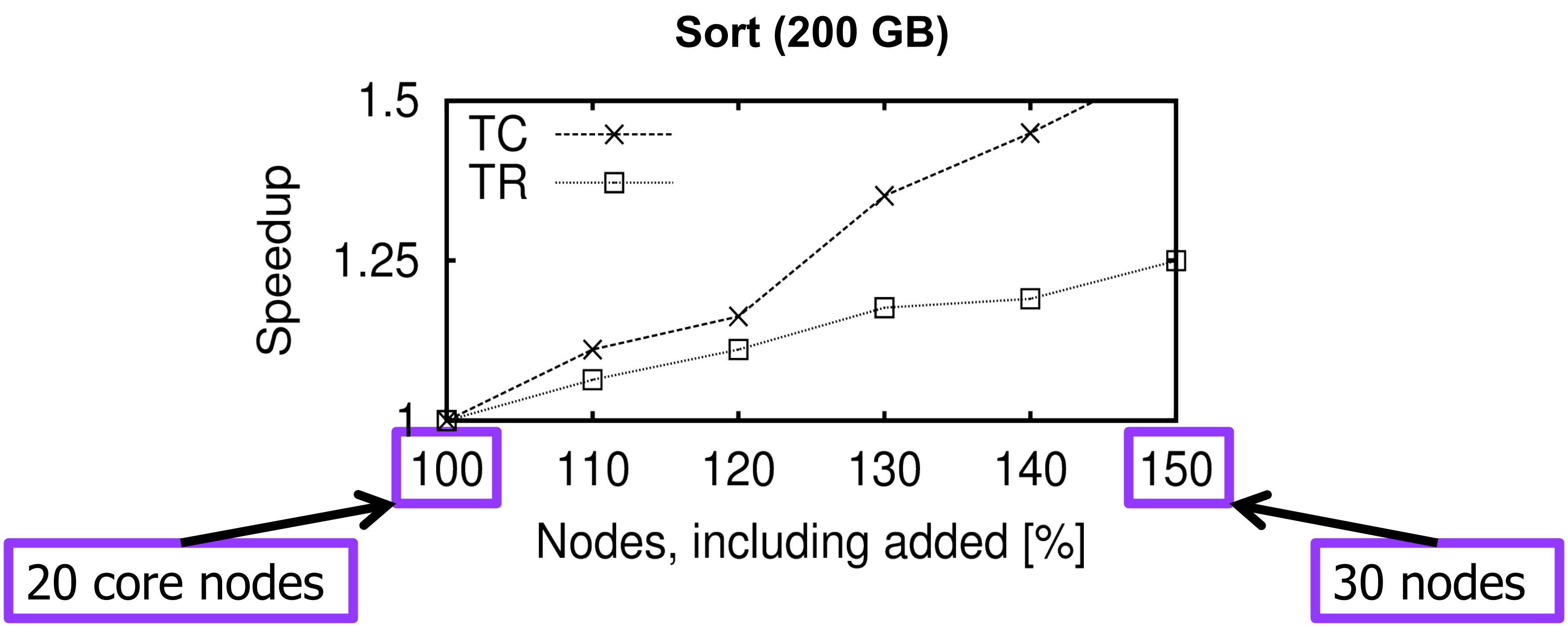


20 core nodes

30 nodes

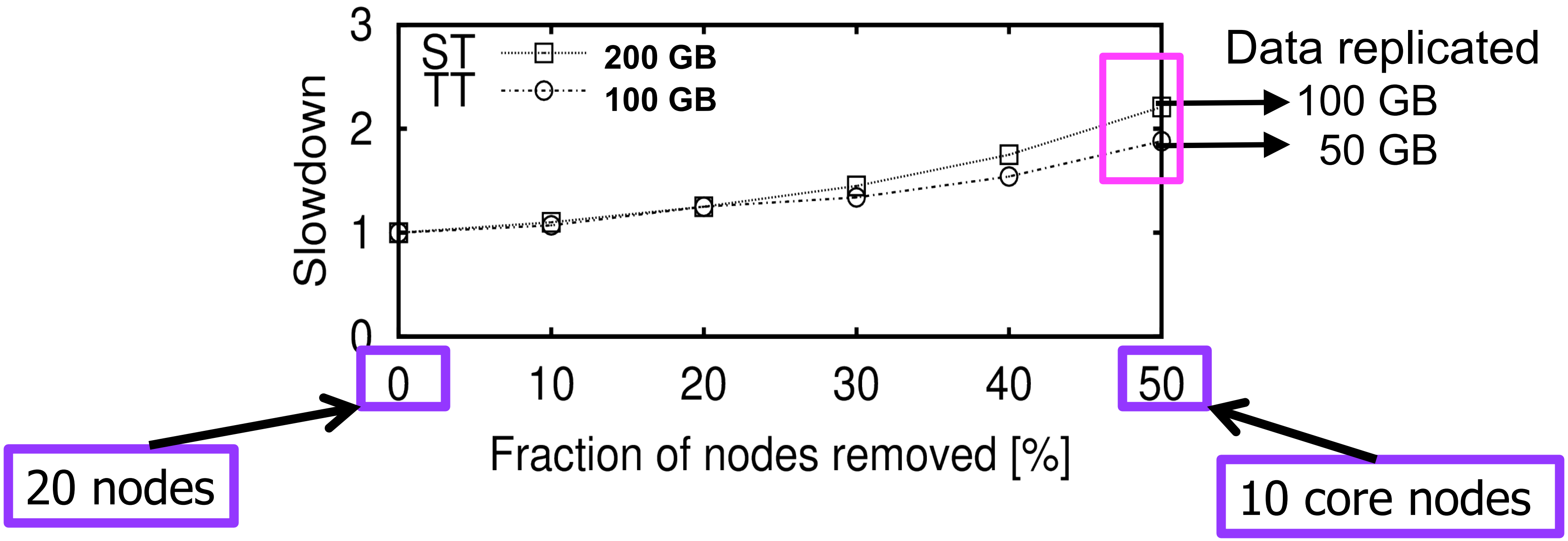
TR nodes deliver good performance for CPU bound workloads

Speedup of growing



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

Speedup of shrinking



Job slowdown increases linearly with the amount of replicated data