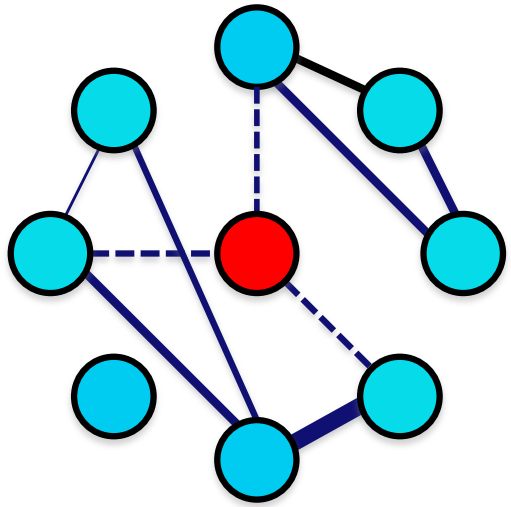
# Dynamic Resource Provisioning for Application Frameworks in Datacenters

**Dick Epema**

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

and

**System Architecture and Networking Group**
Eindhoven University of Technology
Eindhoven, the Netherlands



**10 March 2015**



TU Delft
Delft University of Technology

# The Parallel and Distributed Systems Group: People

Dick Epema

Scheduling
Cloud Computing
P2P systems
Online Social Networks

Alexandru Iosup

Cloud Computing
Big Data
Online gaming
P2P systems

Johan Pouwelse

P2P systems
Video distribution
Online Social Netw.
Applied security

Henk Sips

HPC systems
Parallel computing
Multi-core
P2P systems



TU Delft

# Delft University of Technology: Faculty of EE, Math, and CS (EEMCS)

TU Delft:

- 20,000 BSc+MSc students
- 2,000 PhD students

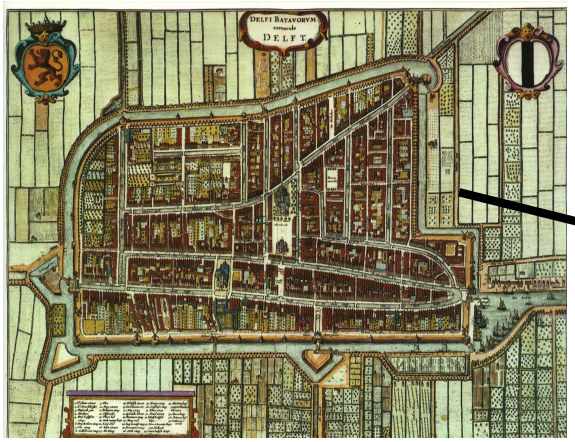
EEMCS:

- 180 scientific staff
- 500 PhD students



TU Delft

# Delft – the Netherlands – Europe







TU Delft

# Delft

the old church

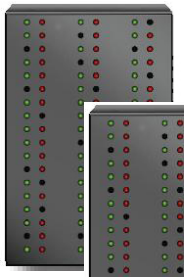
the “new” church



TU Delft

# Our experimental testbed: DAS-4

VU (148 CPUs)



UvA/MultimediaN (72)

UvA (32)

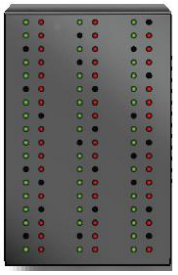
SURFnet6


DAS5 on the way

- Q2 2015
- 400 8-core CPUs
- FDR Infiniband

**10 Gb/s lambdas**

TU Delft (64)



Astron (46)

Leiden (32)



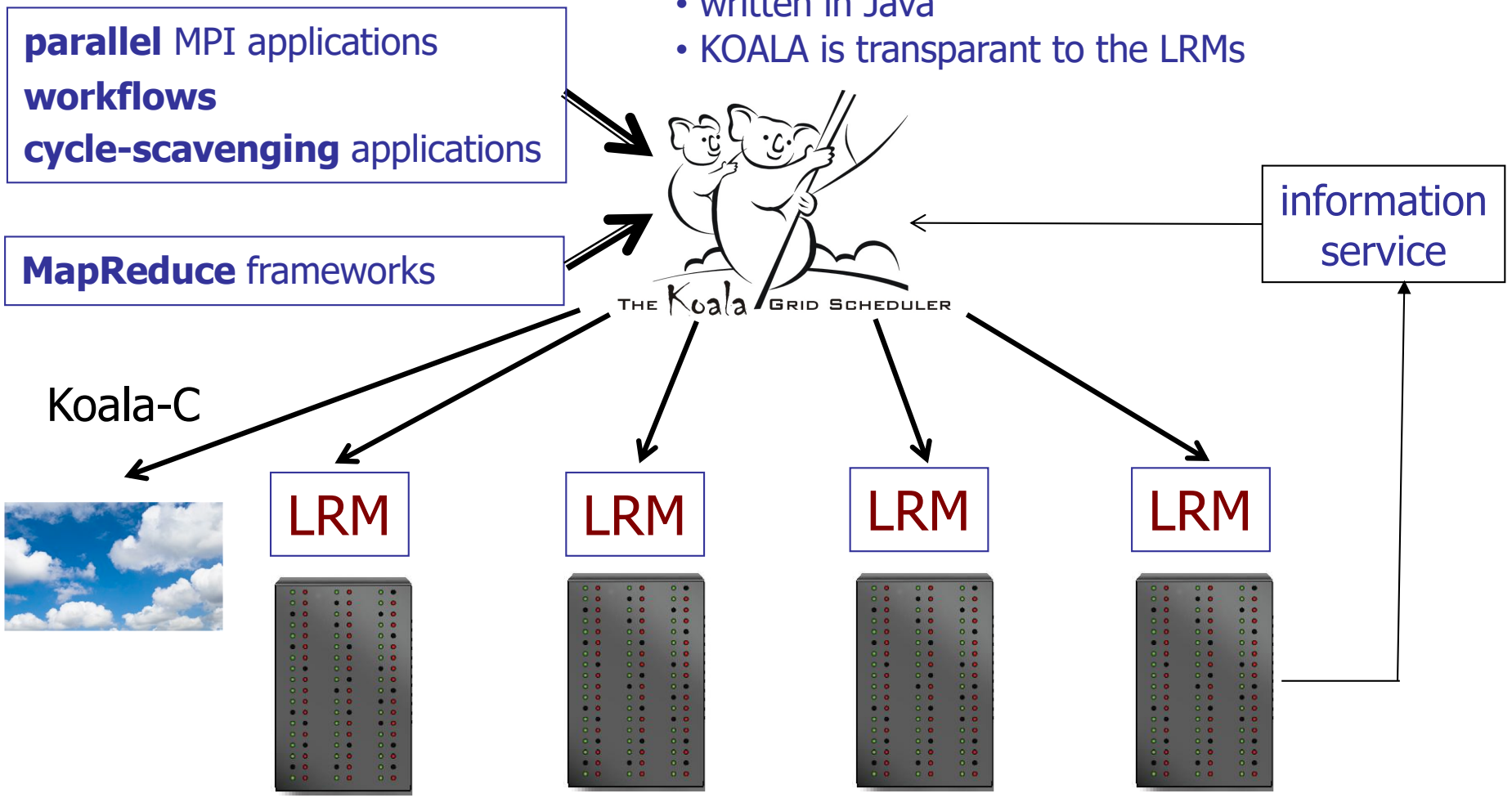
Advanced School for Computing and Imaging

NWO Netherlands Organisation for Scientific Research

TU Delft

# The KOALA multicluster scheduler

- KOALA is our research vehicle for scheduling research
- deployed on DAS generations since 2005
- written in Java
- KOALA is transparant to the LRMs






TU Delft

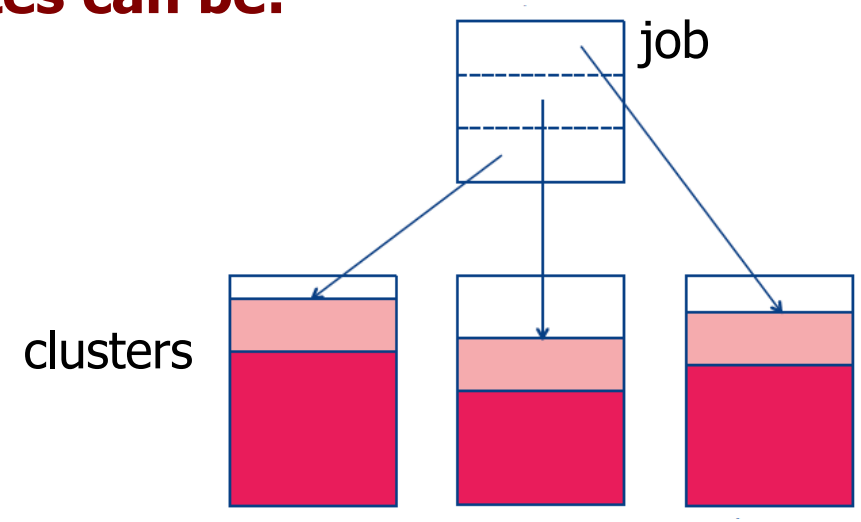
# KOALA: the runners

- The KOALA **runners** are **adaptation modules** for different application types:
  - set up communication / name server / environment
  - launch applications + perform application-level scheduling
  - scheduling policies
- **Current runners:**
  - **CSRunner**: for **cycle-scavenging** applications (PSAs)
  - **Mrunner**: for **malleable parallel applications**
  - **OMRunner**: for **co-allocated parallel** OpenMPI applications
  - **Wrunner**: for **co-allocated workflows**
  - **MR-runner:** for **MapReduce** applications

H.H. Mohamed and D.H.J. Epema, "KOALA: A Co-Allocating Grid Scheduler," *Concurrency and Computation: Practice and Experience*, Vol. 20, 1851-1876, 2008.

# Processor co-allocation (1)

- **Reasons**:
  - to benefit from available resources (e.g., processors, data)
  - application characteristics (e.g., simulation in one location, visualization in another)
- **Resource possession in different sites can be:**
  - simultaneous (e.g., parallel applications)
  - coordinated (e.g., workflows)
- **With co-allocation**:
  - need to **coordinate allocations** by autonomous resource managers



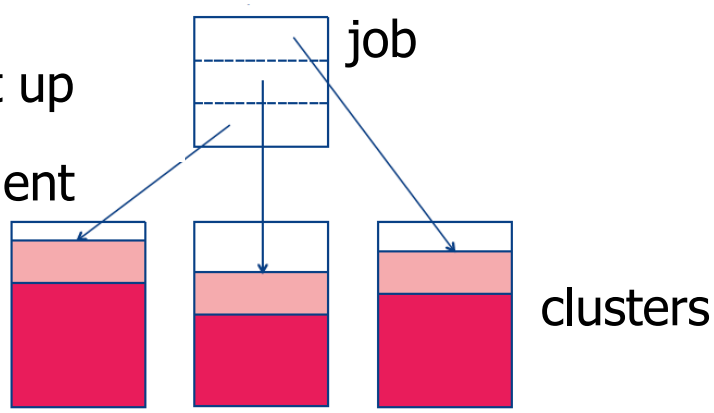

A.I.D. Bucur and D.H.J. Epema, "Scheduling Policies for Processor Co-Allocation in Multicluster Systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 18, pp. 958-972, 2007.

# Co-allocation for parallel applications (2)

## non-fixed job

**user** decides on job split up

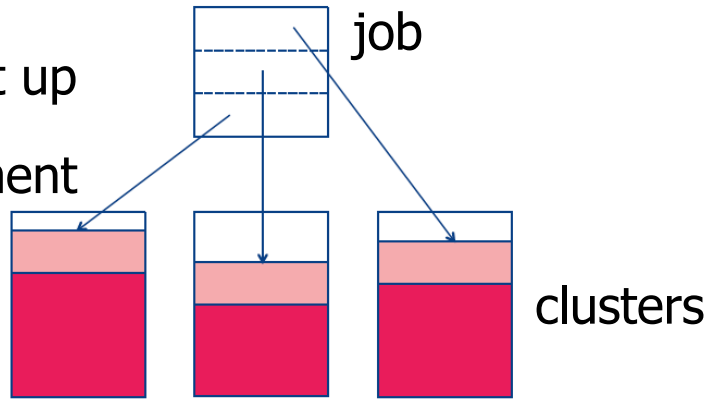
**scheduler** decides on placement




## flexible job

**scheduler** decides on job split up

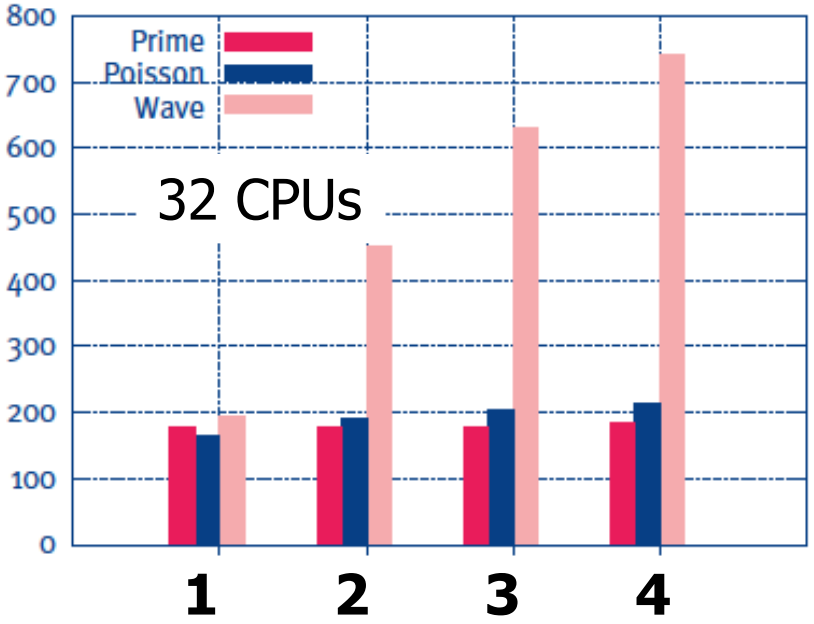
**and** on placement

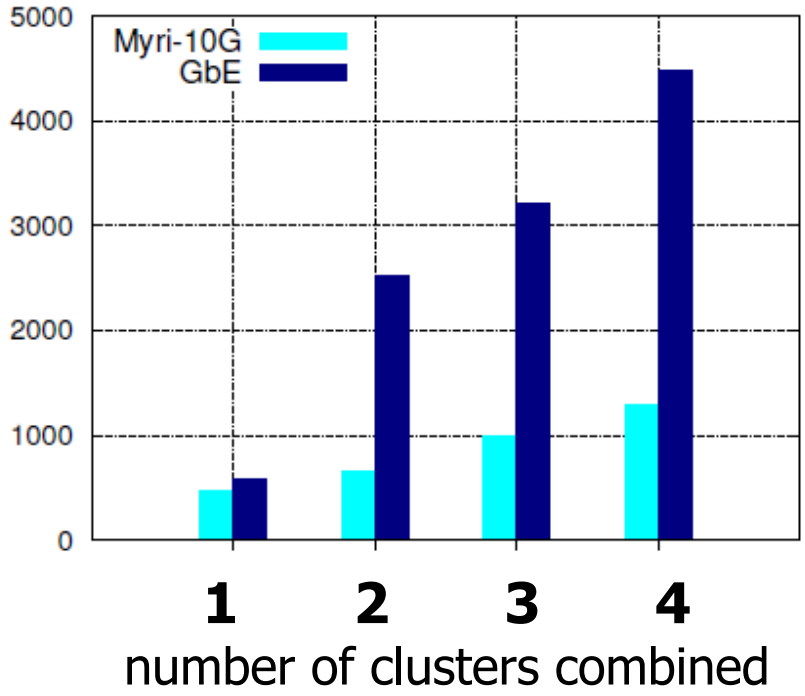


# Co-allocation (3): wide-area communication

- Co-allocated parallel applications are **less efficient** due to the relatively **slow wide-area communications**

execution time (seconds)




O.O. Sonmez, H.H. Mohamed, and D.H.J. Epema, "On the Benefit of Processor Co-Allocation in Multicluster Grid Systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 21, 778-789, 2010.

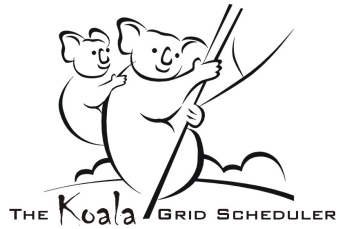
# Co-allocation (4): slowdown

- **Slowdown of a job**:

$$\frac{\text{execution time on multicluster}}{\text{execution time on single cluster}} \quad \textbf{(>1 usually)}$$

- Processor co-allocation is a **trade-off** between
  - \+ **faster access to more capacity**, and higher utilization
  - – **longer execution times**

# Co-allocation (5): scheduling policies

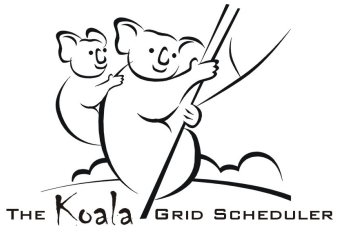



- **Placement policies for non-fixed jobs**:
  1. **Load-aware**: Worst Fit (**WF**)
     (balance load in clusters)
  2. **Input-file-location-aware**: Close-to-Files (**CF**)
     (reduce file-transfer times)
  3. **Communication-aware**: Cluster Minimization (**CM**)
     (reduce number of wide-area messages)
- **Placement policies for flexible jobs**:
  1. **Communication-aware**: Flexible Cluster
     (CM for flexible) Minimization (**FCM**)
  2. **Network-aware**: Communication-Aware (**CA**)
     (take latency into account)



TU Delft

# Co-allocation (6): simulations/analysis




- **Model has a host of parameters**
- **Main conclusions:**
  - co-allocation is beneficial when the **slowdown ≤ 1.20**
  - **unlimited co-allocation is no good**:
    - limit the number of job components
    - limit the maximum job-component size
- **Mathematical analysis for maximal utilization**
  - assessment of "gaps" in the schedule due to parallelism

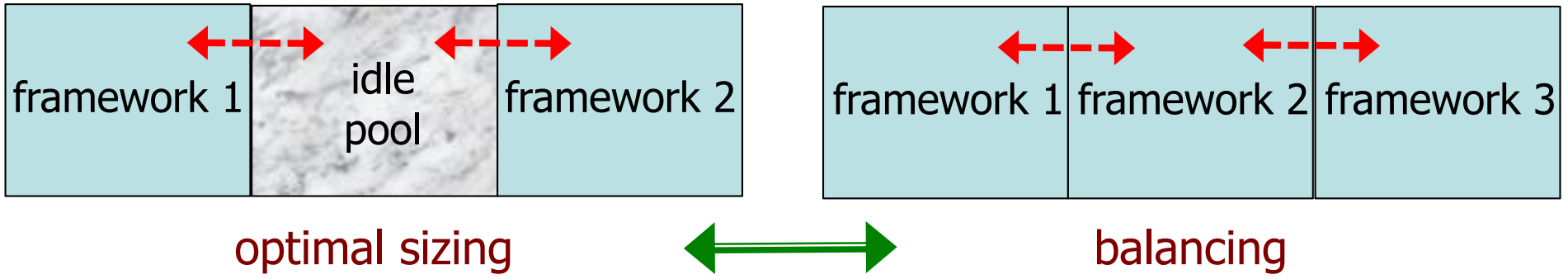
A. Bucur and D.H.J. Epema, "The Maximal Utilization of Processor Co-Allocation in Multicluster Systems," *IPDPS 2003*



TU Delft

# Scheduling frameworks

- **Reduce**
  - **scheduling overhead** of centralized scheduler
  - **complexity** of centralized scheduler
- **Provide isolation among frameworks**
- **KOALA**
  - requests large chunk of a cluster and
  - allocates parts of it to frameworks
- **Two models:**

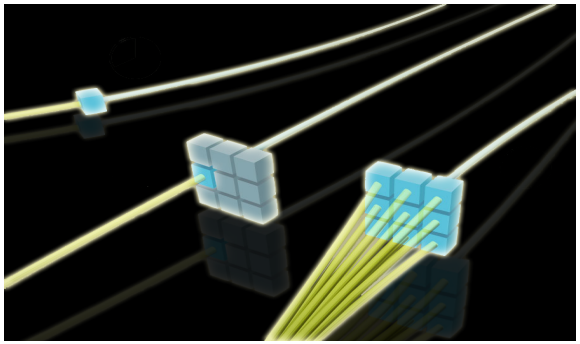





TU Delft

# Types of Isolation

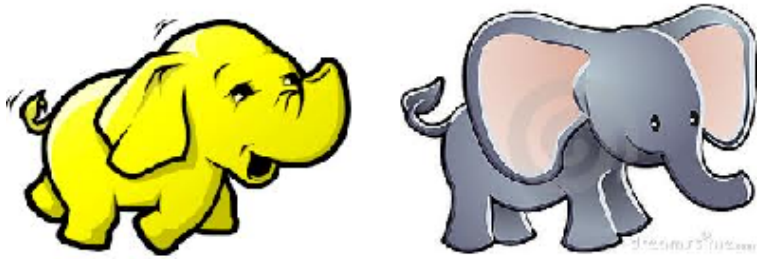
## Performance isolation



## Failure isolation

## Data isolation

## Version isolation

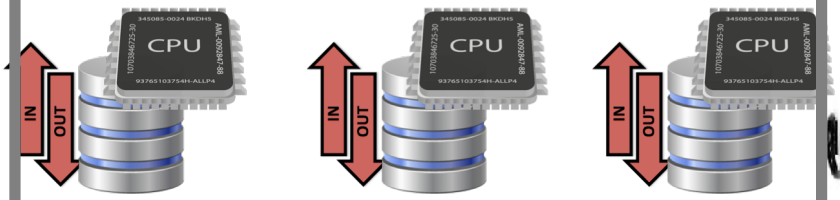




TU Delft

# Resizing MapReduce: no data locality

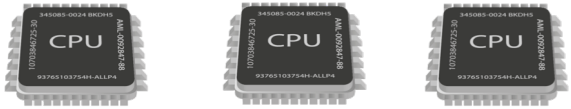

## Core nodes




**INPUT/OUTPUT DATA**

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

## Transient nodes (TR)




**NO DATA**

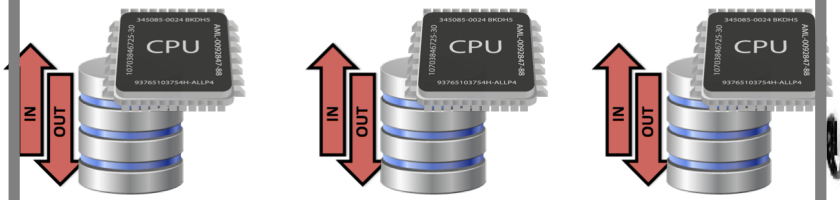
- No local storage
- R/W from/to core nodes
- **Instant removal**



TU Delft

# Resizing MapReduce: relaxed data locality

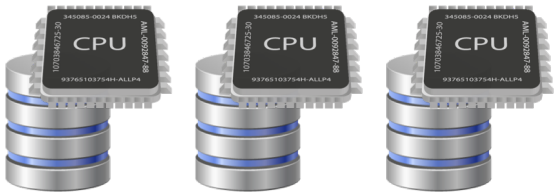

## Core nodes




**INPUT/OUTPUT DATA**

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

## Trans-core nodes (TC)



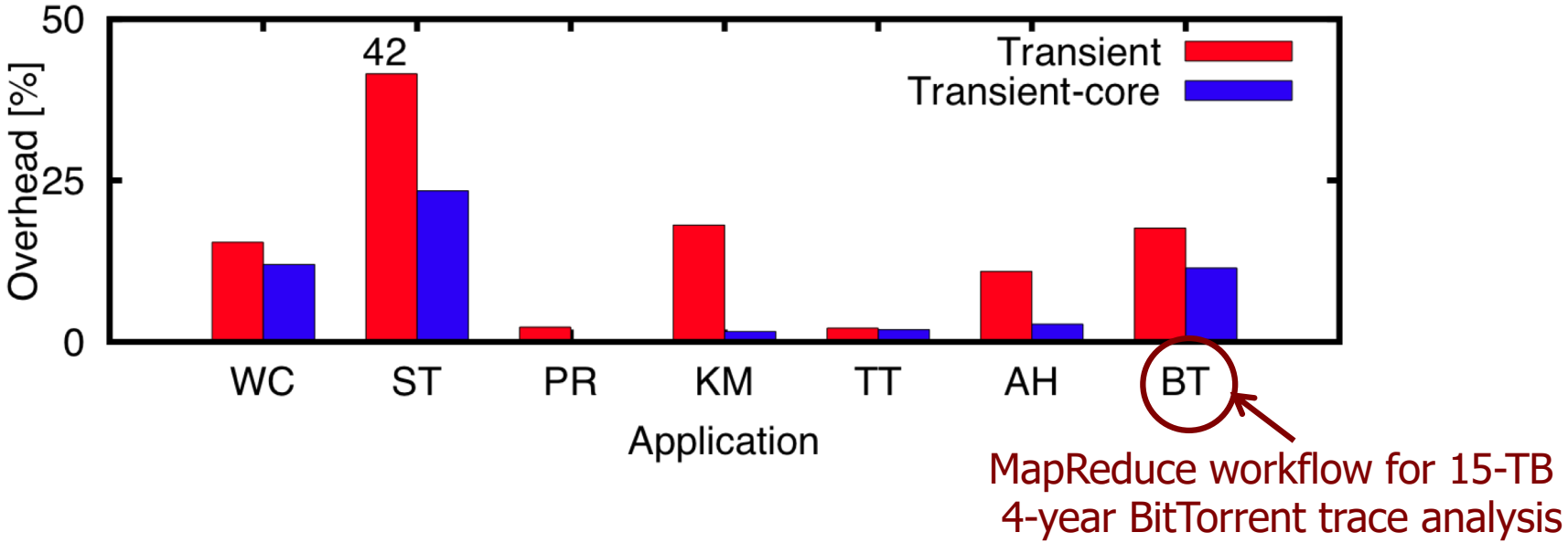

**OUTPUT DATA**

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



TU Delft

# Performance of no versus relaxed data locality




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

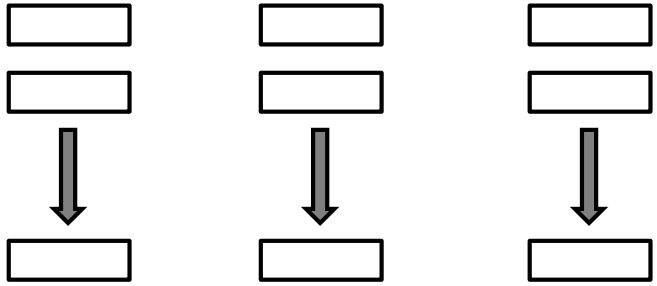
B.I. Ghit, M. Capota, T. Hegeman, J. Hidders, D.H.J. Epema and I. Iosup, "V for Vicissitude: The Challenge of ScalingComplex Big-DataWorkflows," **winner SCALE Challenge** at *CCGrid 2014*



TU Delft

# Balancing Allocations with FAWKES


Two-level scheduling architecture

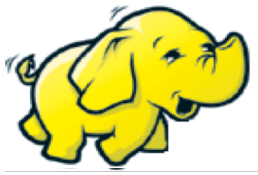


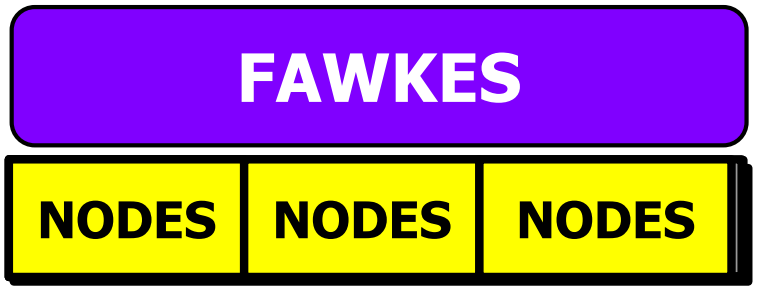
Job submissions

Frameworks

Resource manager

Infrastructure



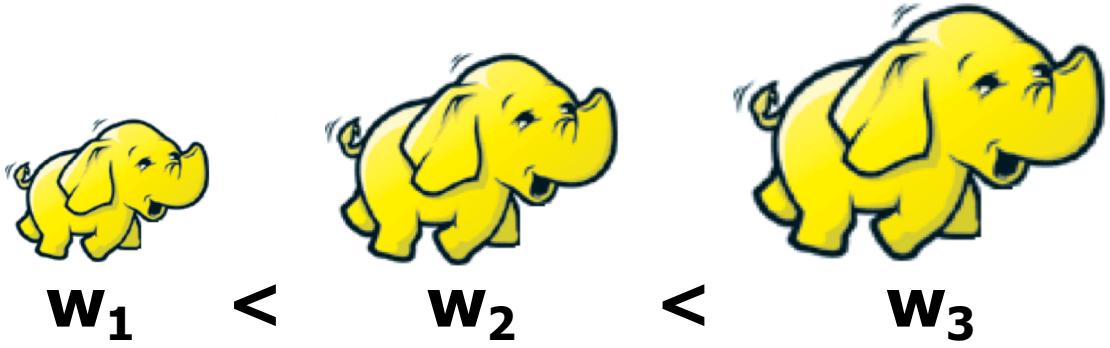


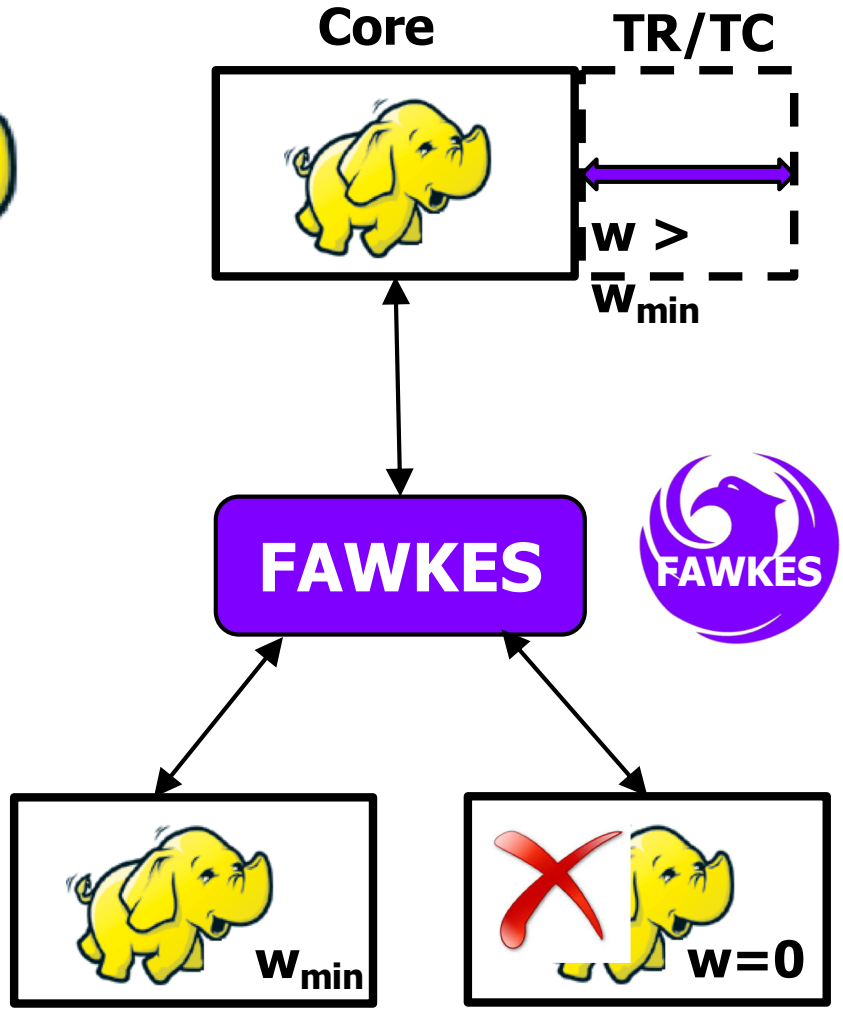

B.I. Ghit, A. Iosup, and D.H.J. Epema, "Balanced Resource Allocations across Multiple Dynamic MapReduce Clusters," *ACM Sigmetrics 2014.*



TU Delft

# FAWKES in a nutshell



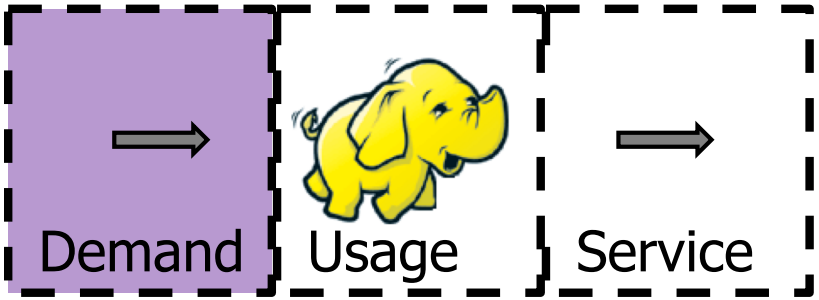

1. Updates dynamic weights when:
   - new frameworks arrive
   - framework states change

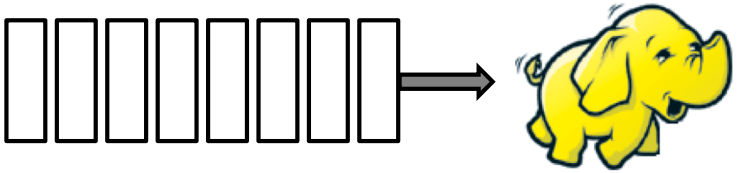
2. Shrinks and grows frameworks to:
   - allocate new frameworks (min. shares)
   - give fair shares to existing ones



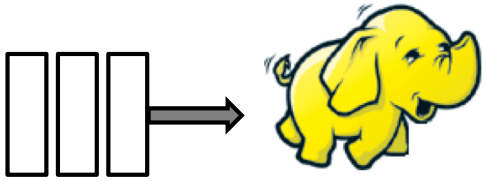
TU Delft

# How to differentiate frameworks? (1/3)



versus



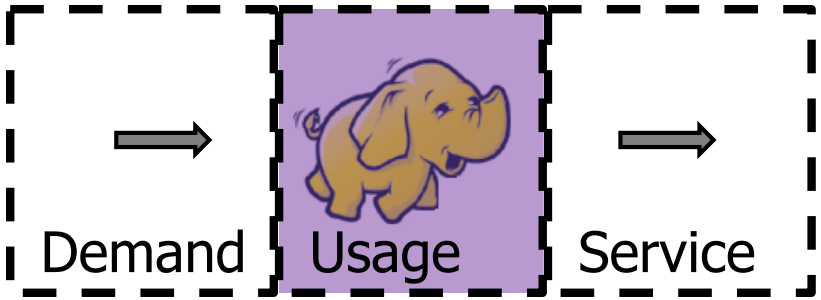
By **demand** – 3 policies:

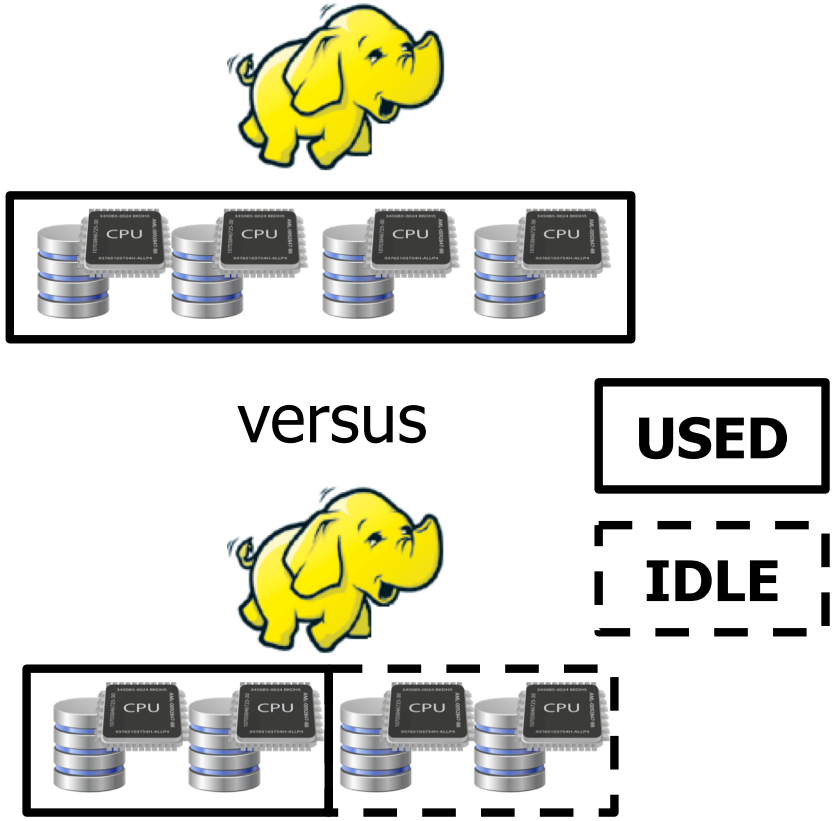
- Job Demand (JD)
- Data Demand (DD)
- Task Demand (TD)



TU Delft

# How to differentiate frameworks? (2/3)



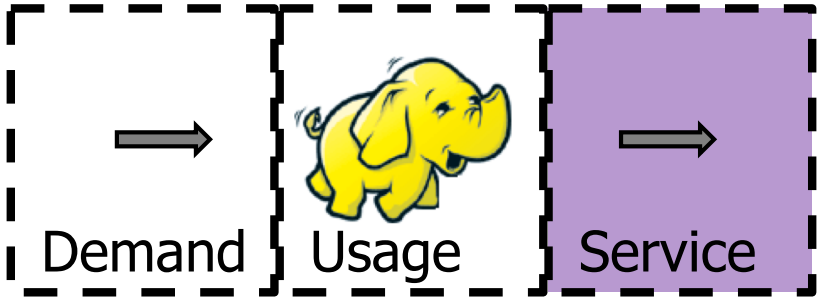

By **usage** – 3 policies:

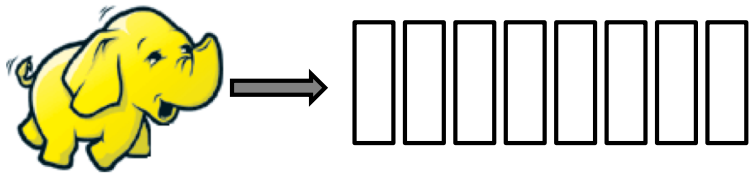
- Processor Usage (PU)
- Disk Usage (DU)
- Resource Usage (RU)



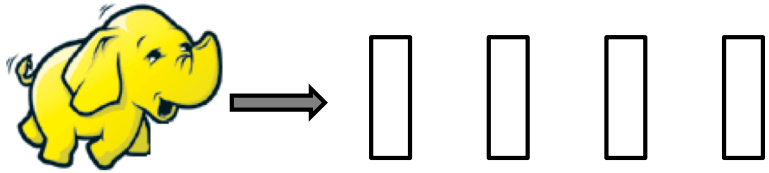
TU Delft

# How to differentiate frameworks? (3/3)



versus



By **service** – 3 policies:

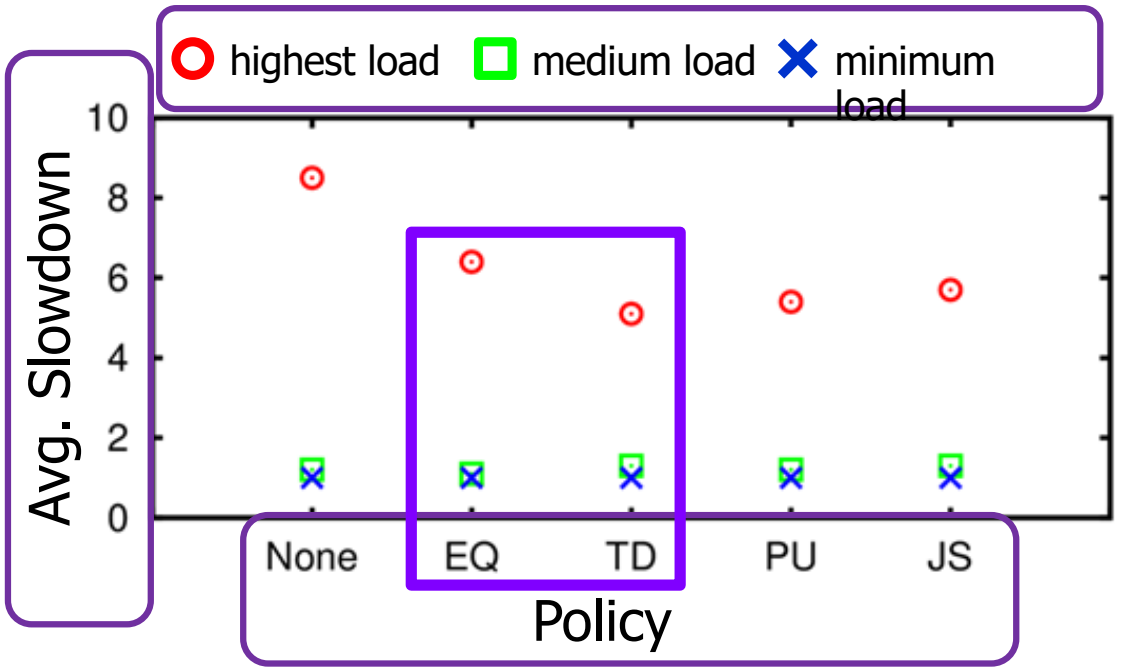
- Job Slowdown (JS)
- Job Throughput (JT)
- Task Throughput (TT)



TU Delft

# Performance of FAWKES

| | |
|---|---|
| Nodes | 45 |
| Frameworks | 3 |
| Minimum 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

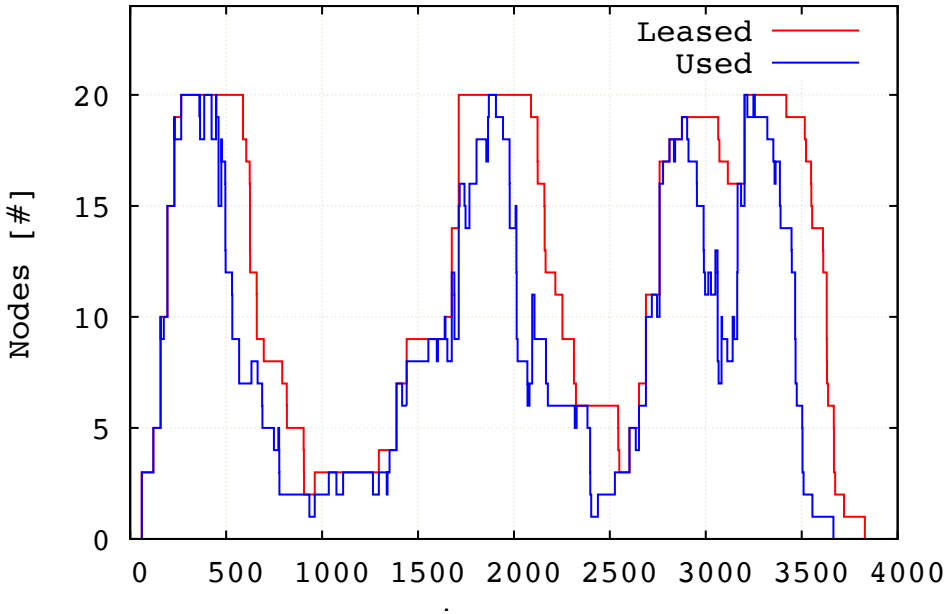


TU Delft

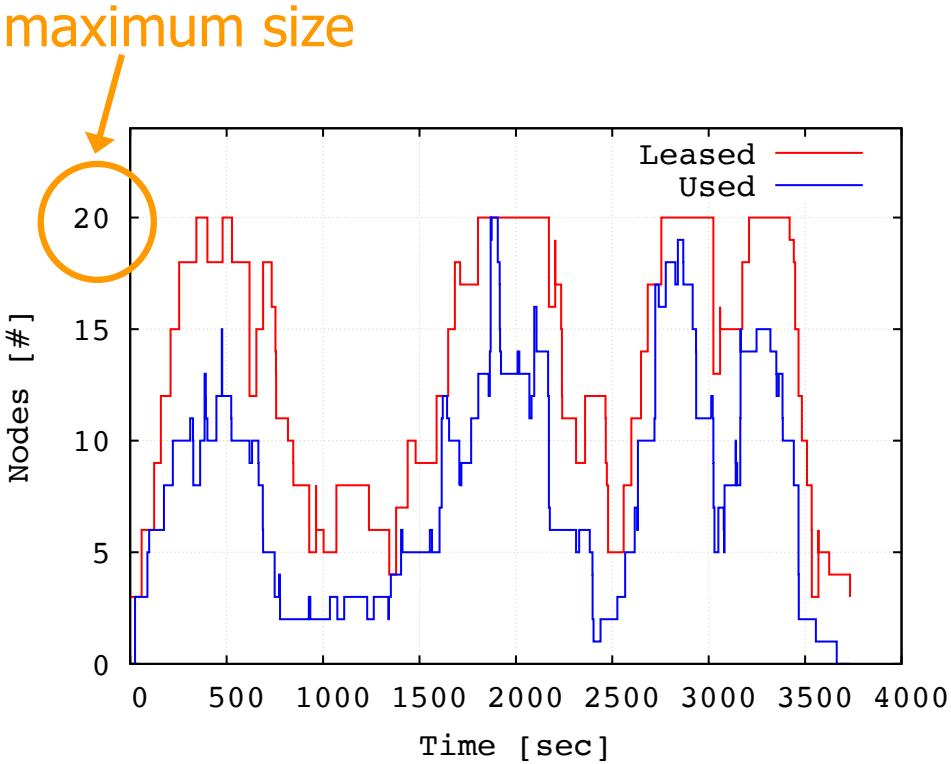
# Optimal sizing (1)

- **Fluent** is a component-based framework
  - jobs consist of **batches of identical video applications** with identical runtimes
  - **admission control**: jobs require immediate/fast start
  - metric: **reject rate** (of all applications across all jobs)
- **OnDemand** policy:
  - framework **initiative**
  - explicit grow and shrink requests to KOALA
  - grow because of new job that doesn't fit
  - shrink after some idle time of resources
- **Proactive** policy:
  - KOALA **initiative**
  - maintain utilization (used/allocated) between lower and upper bound (periodic check)

# Optimal sizing (2)




**OnDemand**



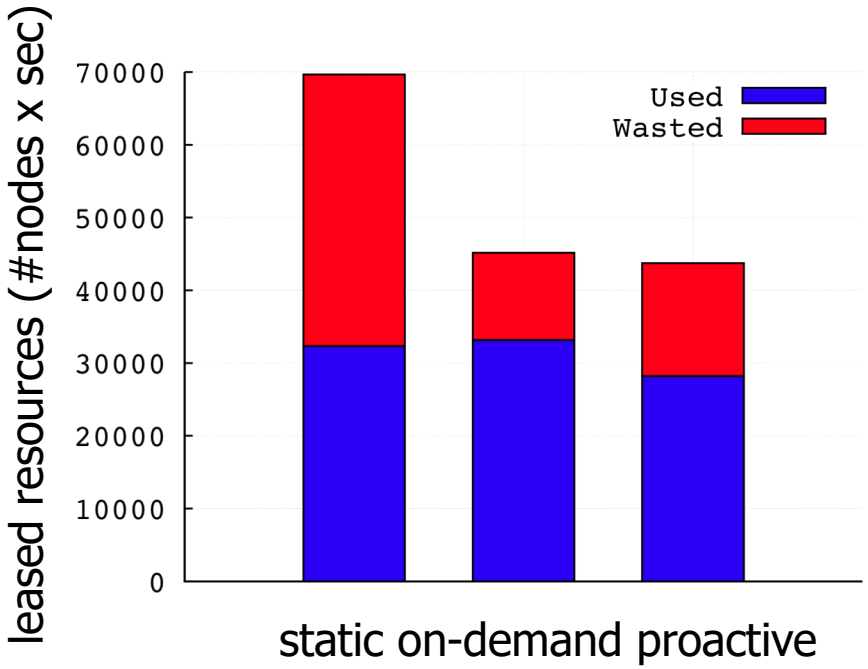

**Proactive (util. 40-50%)**

A. Kuzmanovska, R.H. Mak, and D.H.J. Epema, "Scheduling Workloads of Workflows with Unknown Task Runtimes," *Workshop Job Scheduling Strategies for Parallel Processing*, May 2014



TU Delft

# Optimal sizing (3)




| policy | reject rate (%) | utilization (%) |
|---|---|---|
| static | 13 | 46 |
| on-demand | 13 | 73 |
| pro-active | 21 | 65 |



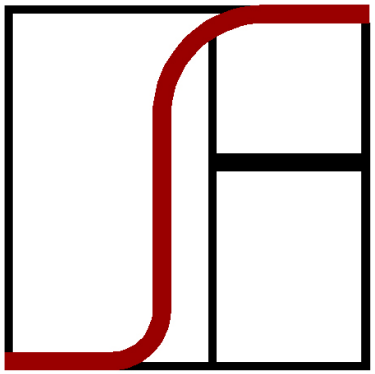
TU Delft

# Other stuff (1): the Failure Trace Archive

and several other

- **Motivation**: (components of) large-scale systems fail
  - no generally accepted failure models
  - no standard way to share failure traces
- The **Failure Trace Archive** is a repository of failure traces of parallel and distributed systems with analysis tools to
  - **understand** failure patterns
  - **facilitate design** of fault-tolerant algorithms
  - **improve reliability** of distributed systems

**http://fta.inria.fr**



D. Kondo, B. Javadi, A. Iosup, and D.H.J. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," *10th IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGRID10)*, May 2010 (**best-paper award**).



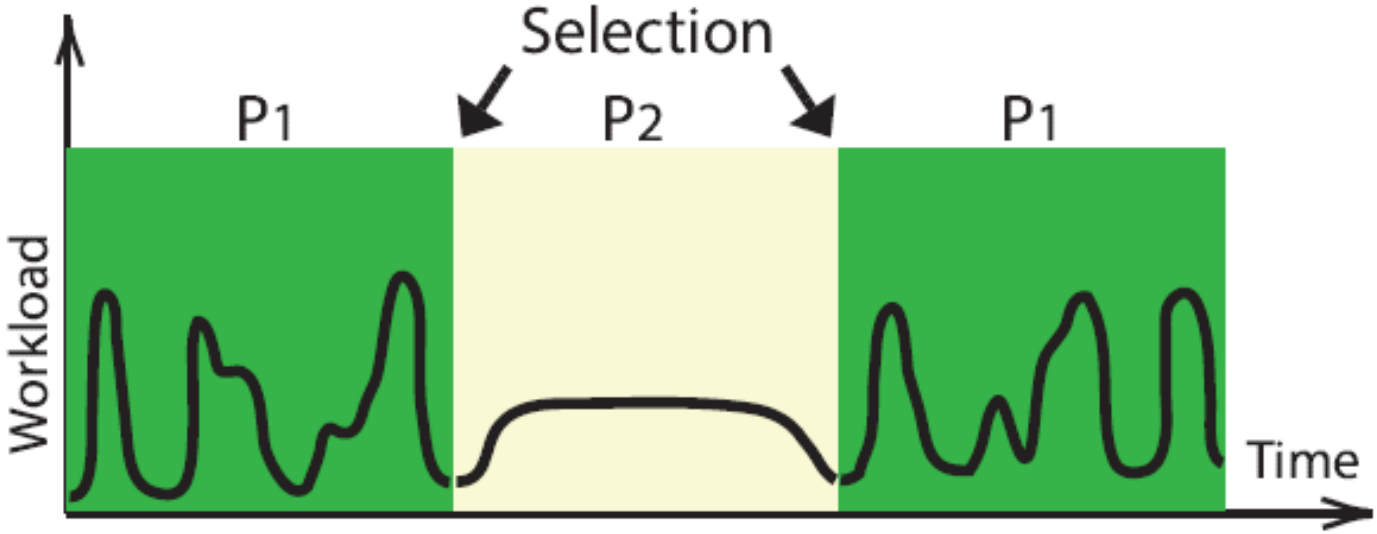
TU Delft

# Other stuff (2): portfolio scheduling (1)

- **Old scheduling aspects**
  - workloads evolve over time
  - no one-size-fits-all policy: hundreds of policies exist, each good for specific conditions
- **New scheduling aspects**
  - new workloads
  - new data center architectures
  - new cost models
- **Issues:**
  - developing a scheduling policy is risky and ephemeral
  - selecting a scheduling policy for your data center is difficult
  - combining the strengths of multiple scheduling policies is …

K. Deng, J. Song, K. Ren, and A. Iosup, "Exploring Portfolio Scheduling for Long-term Execution of Scientific Workloads in IaaS Clouds," *SuperComputing* 2013

# Other stuff (2): portfolio scheduling (2)



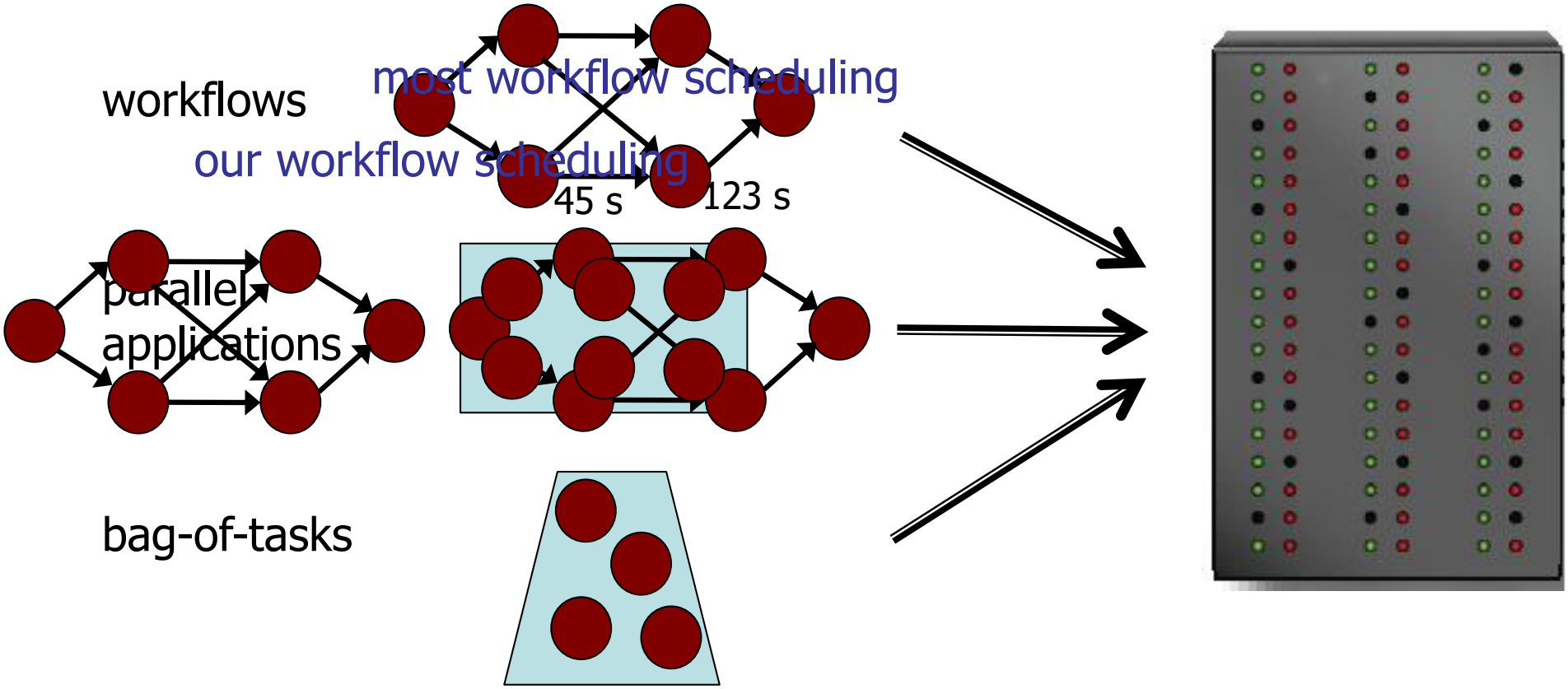

- Create a set of scheduling policies
  - resource provisioning and allocation policies
- Online selection of the active policy, at important moments
  - periodic selection
  - change in pricing model
  - change in datacenter architecture



TU Delft

# Other stuff (3): workflow scheduling (1)

real workloads




A. Ilyushkin, B.I. Ghit, and D.H.J. Epema, "Scheduling Workloads of Workflows with Unknown Task Runtimes," *15th IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGRID15)*, May 2015

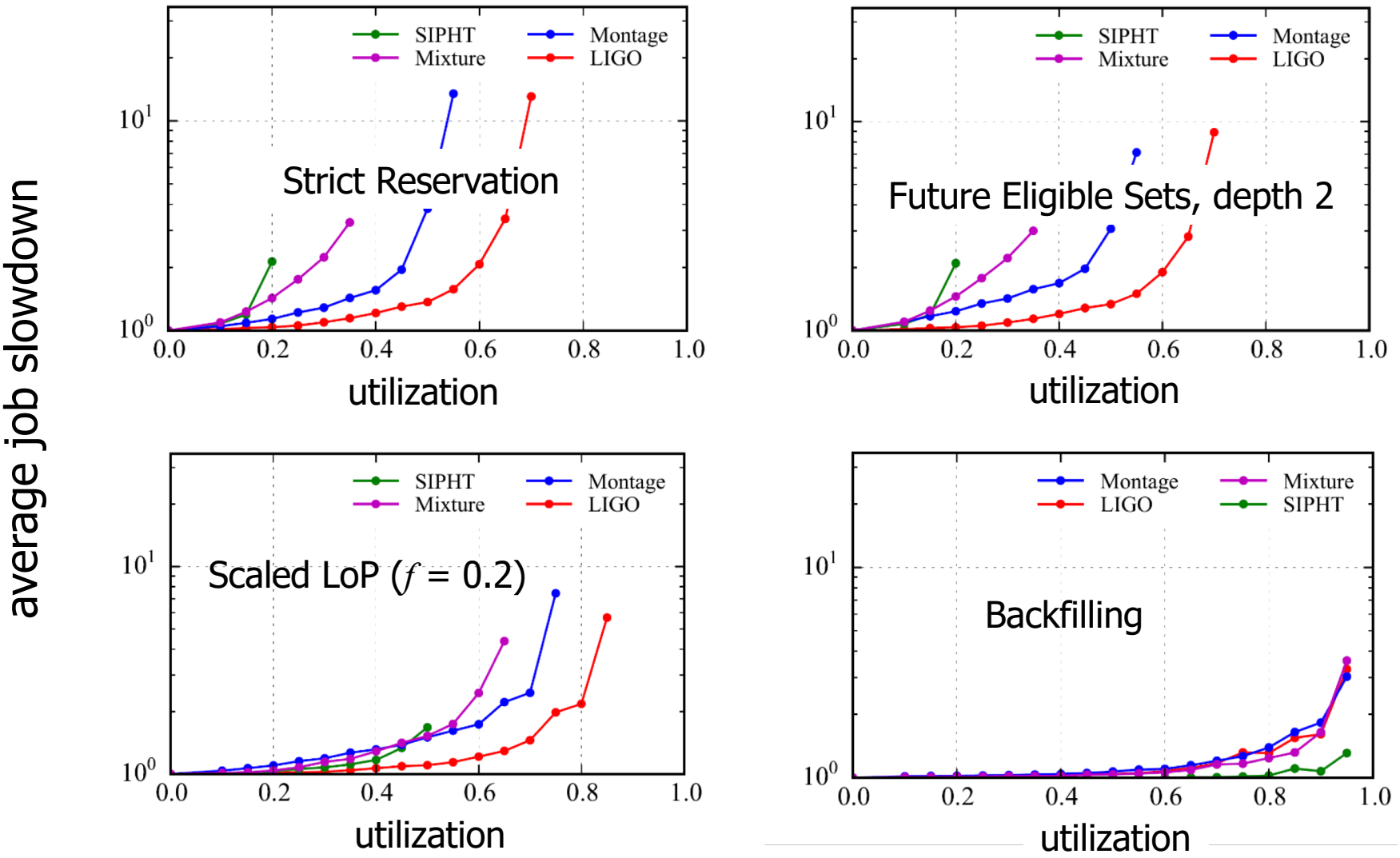


TU Delft

# Other stuff (3): workload scheduling (2)

- **Research question**
  - how to schedule **workloads of workflows** with **unknown** task runtimes?
- **Reserving processors for job(s) at the head of the queue**
  - reduces time in service
  - but increases wait time
- **Policies**
  - strict reservation (reserve for **maximum Level of Parallelism**)
  - scaled LoP (reserve only for fraction of max. LoP)
  - future eligible sets (look number of steps into the future)
  - (unrestricted) backfilling
- **Metric**
  - job slowdown

# Other stuff (3): workload scheduling (3)






TU Delft

# The Tribler BitTorrent-based P2P client


## Tribler

- Considers peers as really representing **actual users**
- Adds **social-based** functionality (e.g., taste buddies)
- Uses an **epidemic protocol** for decentralized peer and content discovery
- Peers keep a **MegaCache** with information on the whole system
- Was **first released** on 17 March 2006 (1,500,000+ downloads)
- Has channels, a reputation system, a new transport protocol (IETF)
- Is our **research vehicle** for P2P research
- Current focus: privacy, trust, and anti-censorship

J.A. Pouwelse, P. Garbacki, A. Iosup, D.H.J. Epema, H.J. Sips, M. van Steen, et 4 al., "Tribler: A Social-Based Peer-to-Peer System," *Concurrency and Computation: Practice and Experience*, Vol. 20, pp. 127-138, 2008.

# Next March in Delft


- Home
- Important Dates
- Call For Contributions
  - TXT version
  - PDF version

## Welcome to the 7th ACM/SPEC International Conference on Performance Engineering

The International Conference on Performance Engineering (ICPE) provides a forum for the integration of theory and practice in the field of performance engineering. ICPE is an annual joint meeting that has grown out of the ACM Workshop on Software Performance (WOSP) and the SPEC International Performance Engineering Workshop (SIPEW). It brings together researchers and industry practitioners to share ideas, discuss challenges, and present results of both work-in-progress and state-of-the-art research on performance engineering of software and systems.

## Sponsors

TU Delft

Delft University of Technology

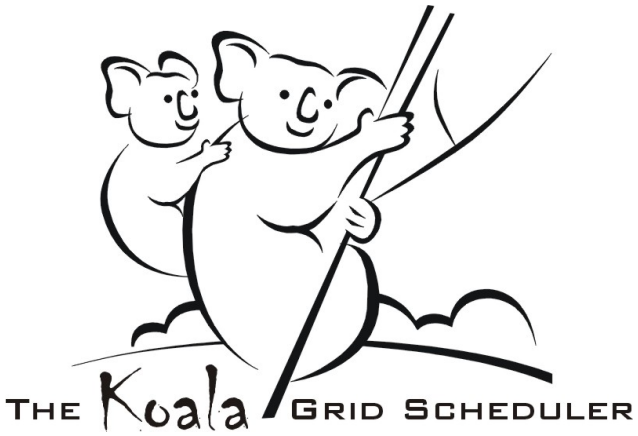
General Chair: Alex Iosup



TU Delft

# More information

- **Publications**
  - see PDS publication database at publications.st.ewi.tudelft.nl
- **Home pages**:
  - www.pds.ewi.tudelft.nl/epema
  - www.pds.ewi.tudelft.nl/~iosup
  - www.pds.ewi.tudelft.nl/pouwelse
- **Web sites:**
  - KOALA: www.st.ewi.tudelft.nl/koala
  - DAS4: www.cs.vu.nl/das4
  - FTA: fta.inria.org (failure trace archive)
  - Tribler: www.tribler.org







TU Delft

# Our research tag cloud




TU Delft