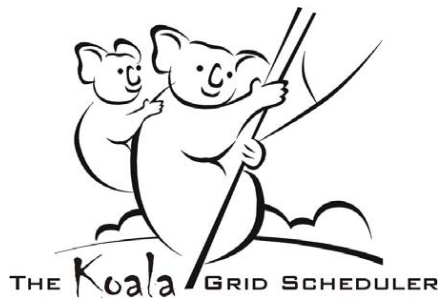


The KOALA Grid Scheduler

Processor and data co-allocation in grids



**Dick Epema, Alexandru Iosup,
Hashim Mohamed, Ozan Sonmez**



october 2006

1

Parallel and Distributed Systems Group



Contents

- Problems in grid scheduling
- The DAS testbed
- A model for co-allocation
- The design of the KOALA co-allocating scheduler
- Some performance results
- Future Work

october 2006

2



Problems in grid scheduling (1): system

1. Grid schedulers usually do not own resources themselves

- they have to negotiate with autonomous local schedulers
- authentication/multi-organizational issues

2. Grid schedulers have to interface to different local schedulers

- some may have support for reservations, others are queuing-based
- some may support checkpointing, migration, etc

3. The set of grid resources is heterogeneous and dynamic

- hardware (processor architecture, disk space, network)
- basic software (OS, libraries)
- grid software (middleware)
- systems management (security set-up, runtime limits)
- **resources may fail!!!!!!!**

Problems in grid scheduling (2): workloads

4. Workloads are heterogeneous and dynamic:

- grid schedulers may not have control over the full workload (multiple submission points)
- different application types
- jobs may have performance requirements

5. Structure of applications

- many different structures (parallel, PSAs, workflows, etc)
- for co-allocation, optimized wide-area communications libraries are needed

Problems in grid scheduling (3): analysis

6. Need for a suite of test applications

- for functionality and reliability testing
- for performance evaluation
- should run on "all grids"

7. Reproducibility of performance experiments

- never the same circumstances
- tools for mimicking same conditions

october 2006

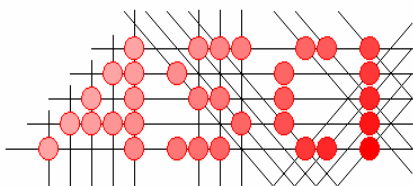
5



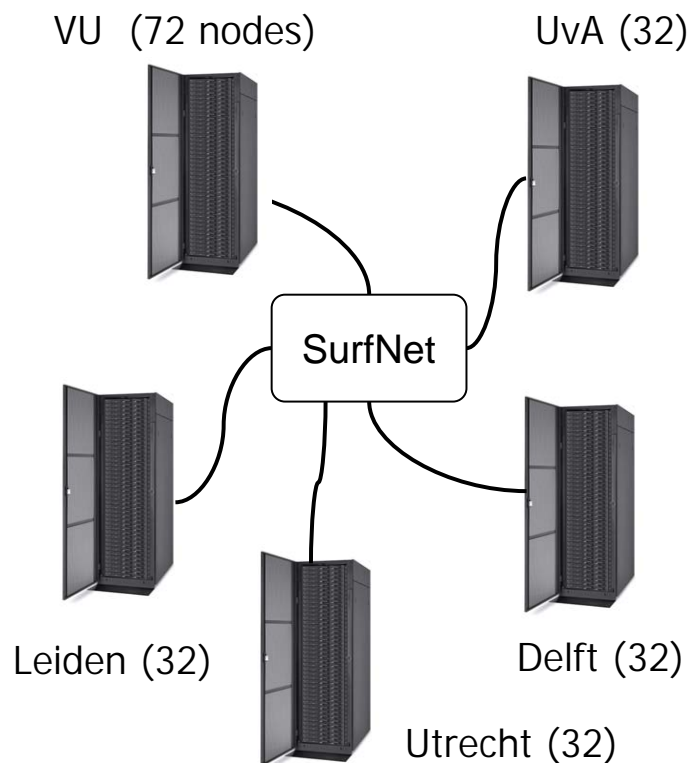
Distributed ASCII Supercomputer (DAS2)

Installed early 2002

- Vendor: IBM
- Dual 1-GHz Pentium-III nodes
- Myrinet/Surfnet
- Linux
- Cluster scheduler: SGE
- Globus installed



Advanced School for Computing and Imaging

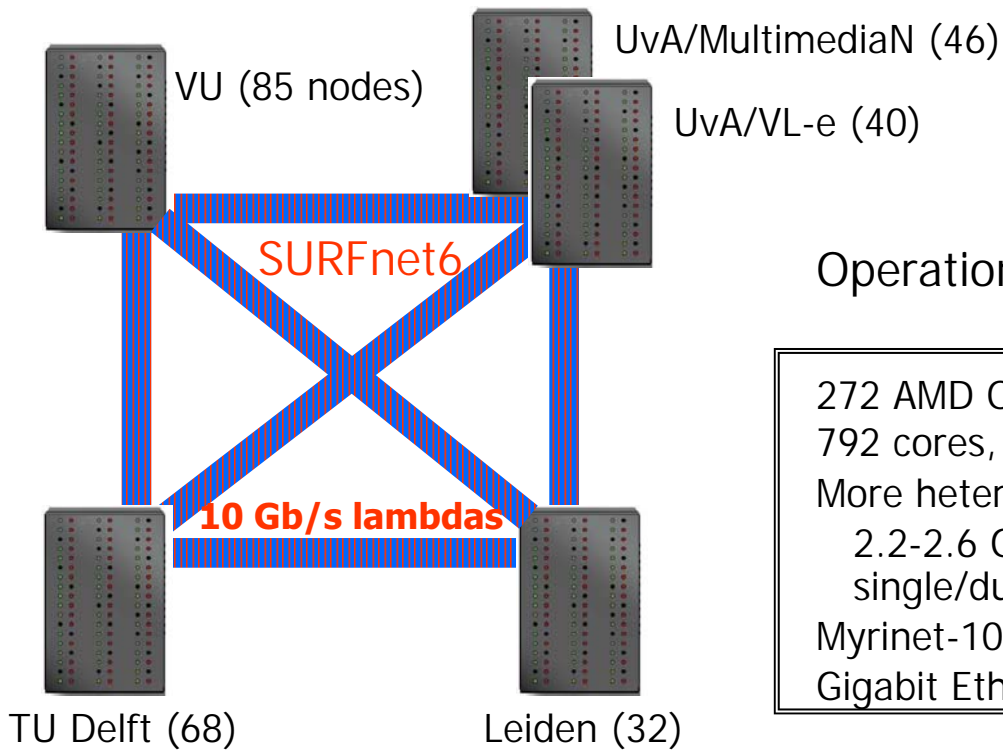


october 2006

6



DAS-3

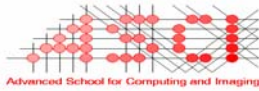


Operational: oct. 2006

272 AMD Opteron nodes
792 cores, 1TB memory
More heterogeneous:
2.2-2.6 GHz
single/dual core nodes
Myrinet-10G (excl. Delft)
Gigabit Ethernet

october 2006

7



Comparison with Grid'5000

- 15 minutes maximum slot at day time
- No constraints for nights and weekend
- No reservations
- No deployment of our own operating system
- Less heterogeneous



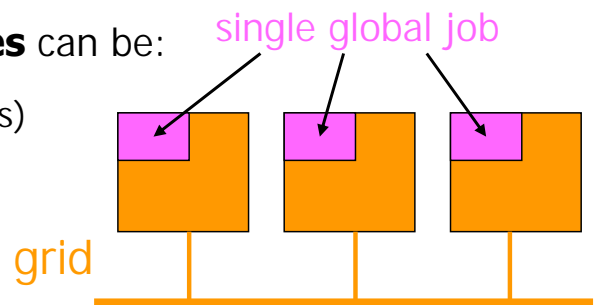
october 2006

8



Co-Allocation (1)

- In grids, jobs may use multiple types of resources in multiple sites: **co-allocation** or **multi-site operation**
- **Reasons:**
 - to use available resources (e.g., processors)
 - to access and/or process geographically spread 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)

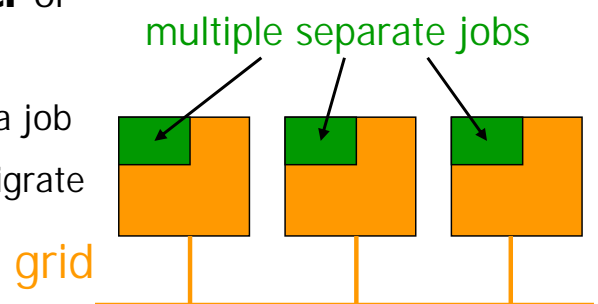


october 2006

9

Co-Allocation (2)

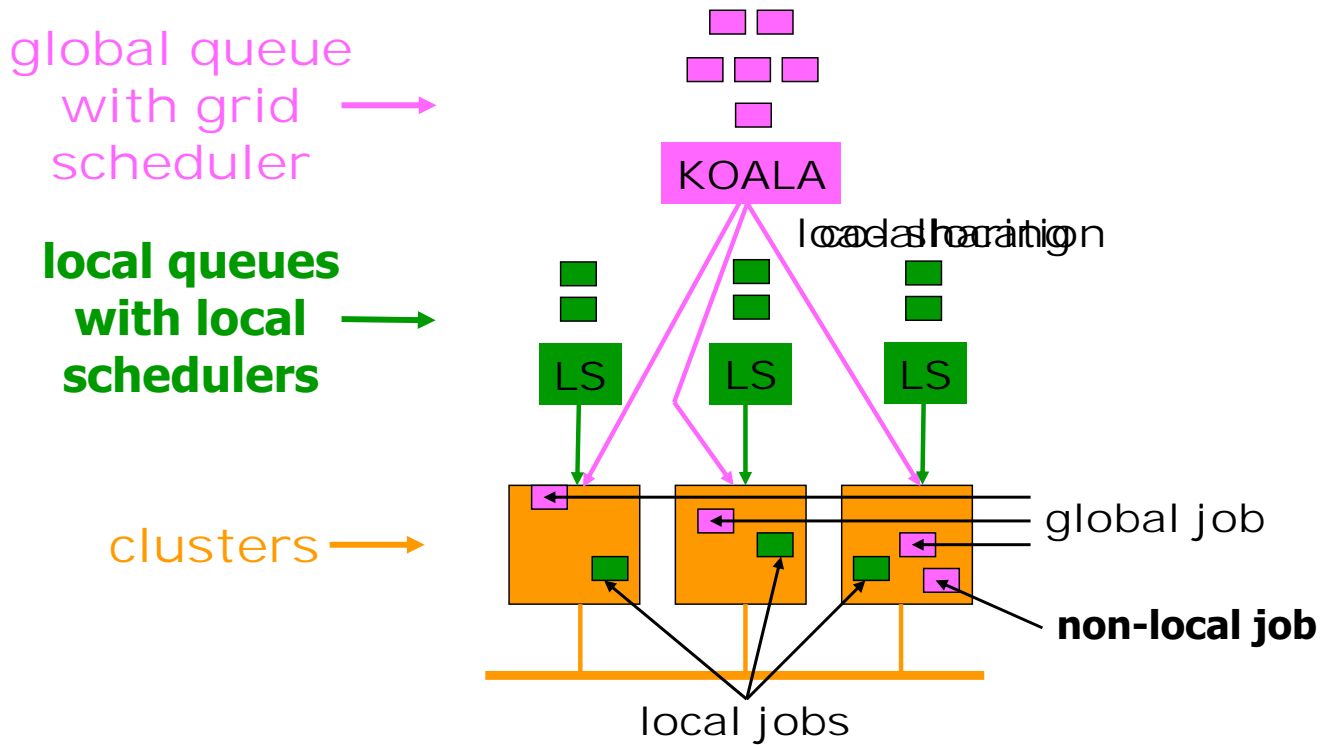
- **Without co-allocation**, a grid is just a big load-sharing device with a **metascheduler** or **superscheduler**:
 - find suitable candidate system for running a job
 - if the candidate is not suitable anymore, migrate
- **With co-allocation:**
 - more difficult resource-discovery process
 - need to coordinate allocations by autonomous resource managers (**local schedulers**)



october 2006

10

A model for co-allocation (1): schedulers

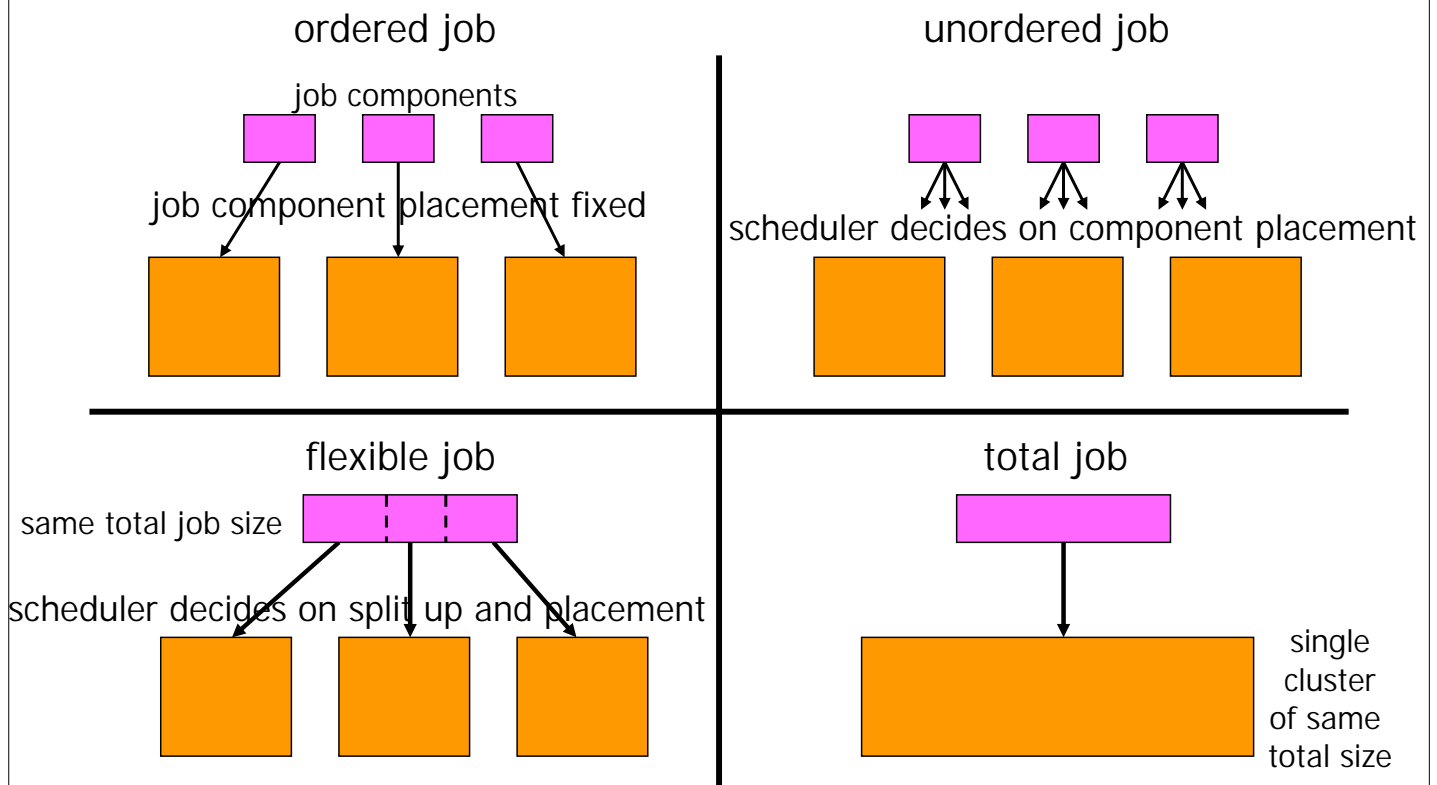


october 2006

11



A model for co-allocation (2): job types



october 2006

12



A model for co-allocation (3): slowdown

- Co-allocated applications are **less efficient** due to the relatively slow wide-area communications
- **Extension factor of a job:**
$$\frac{\text{service time on multicluster}}{\text{service time on single cluster}} \quad (>1 \text{ usually})$$
- Processor co-allocation is a **trade-off** between faster access to more capacity and shorter service times
- Communications libraries may be optimized for wide-area communication

A model for co-allocation (4): policies

- **Placement policies** dictate where the components of a job go
- Placement policies for **unordered jobs:**
 - **Load-aware:** Worst Fit (**WF**)
(balance load in clusters)
 - **Input-file-location-aware:** Close-to-Files (**CF**)
(reduce file-transfer times)
 - **Communication-aware:** Cluster Minimization (**CM**)
(reduce number of wide-area messages)
- Placement policies for **flexible jobs:**
 - **Communication- and queue time-aware:** Flexible Cluster Minimization (**FCM**)
(CM + reduce queue wait time)

Simulations of co-allocation

- Processors only resource considered
- Model has a host of parameters
- **Main conclusions:**
 - Co-allocation is beneficial when the **extension factor ≤ 1.20**
 - **Unlimited co-allocation is no good:**
 - limit the number of job components
 - limit the maximum job-component size
 - **Give local jobs some** but not absolute **priority** over global jobs

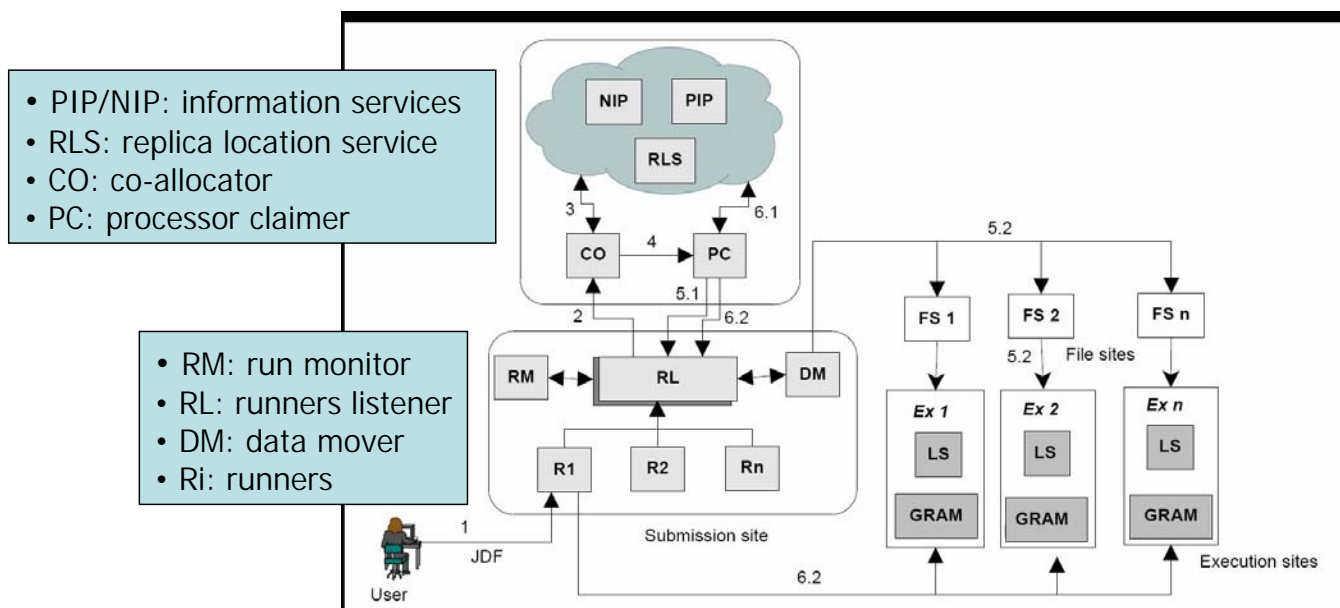
See, e.g.: A.I.D. Bucur and D.H.J. Epema, "Trace-Based Simulations of Processor Co-Allocation Policies in Multiclusters," *HPDC 2003*.

KOALA: a Co-Allocating grid scheduler

- Main goals:
 - 1. processor co-allocation:** (un)ordered/flexible jobs
 - 2. data co-allocation:** move large input files to the locations where the job components will run prior to execution
 - 3. load sharing:** in the absence of co-allocation
 - 4. run alongside local schedulers**
- **KOALA**
 - is written in Java
 - uses Globus components (e.g., RSL and GridFTP)
 - for launching jobs uses its own mechanisms or Globus DUROC
 - has been deployed on the DAS2 in september 2005



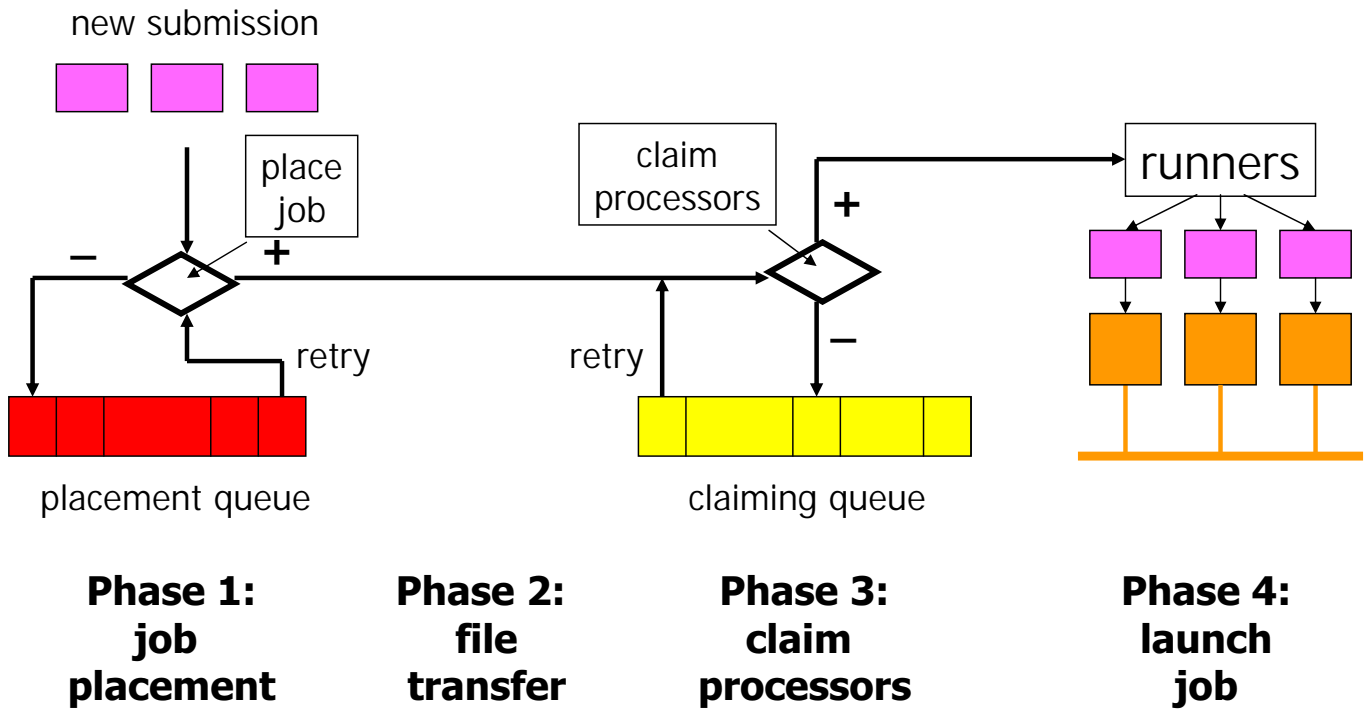
KOALA: the architecture



KOALA: the runners

- The KOALA **runners** are adaptation modules for different application types:
 - set up communication
 - launch applications
- Current runners:
 - **KRunner**: default KOALA runner that co-allocates processors and that's it
 - **DRunner**: DUROC runner for co-allocated MPI applications
 - **IRunner**: runner for applications using the Ibis Java library (Vrije University) for grid applications
 - **MDRunner**: runner for HOC-based applications (University of Muenster)
 - **PSARunner**: coming soon...

KOALA: job flow with four phases

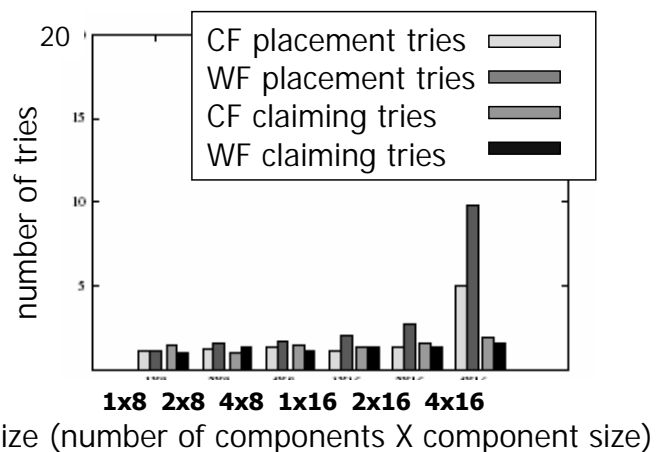
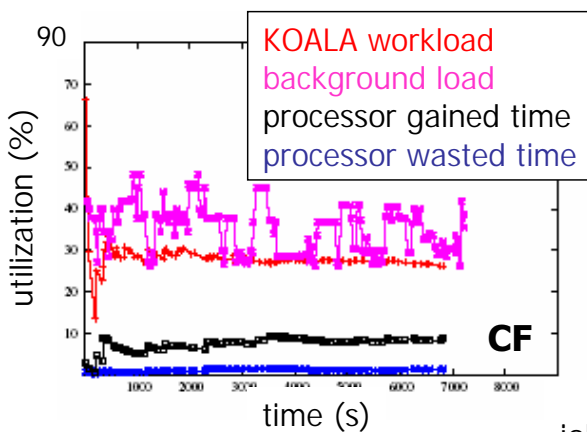


october 2006

19



KOALA: performance results (1)



- With replication (3 copies of input files, 2, 4, or 6 GB)
- Offer a 30% co-allocation load during two hours
- Try to keep the background load between 30% and 40%

See, e.g.: H.H. Mohamed and D.H.J. Epema, "An Evaluation of the Close-to-Files Processor and Data Co-Allocation Policy in Multiclusters," *IEEE Cluster 2004*.

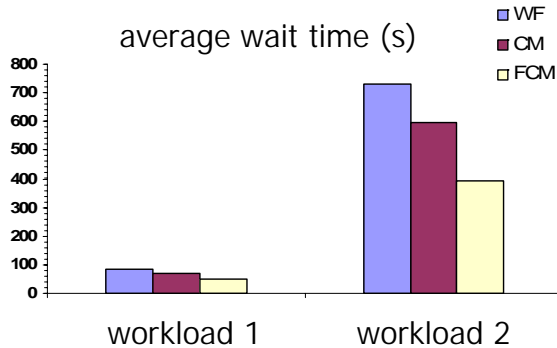
october 2006

20

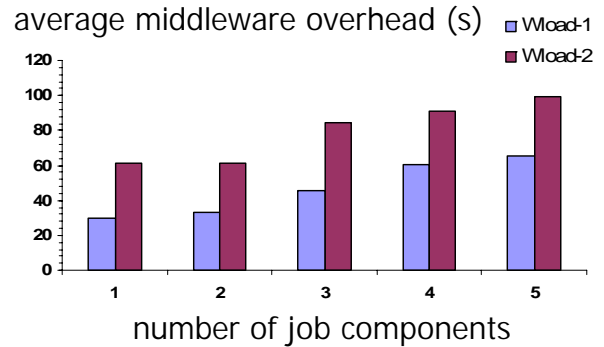
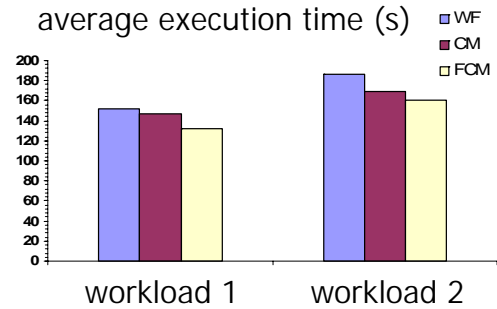


Koala: Performance results (2)

- Communication-intensive applications
- Workload 1: low load
- Workload 2: high load
- Background load: 15-20%



See: O. Sonmez, H.H. Mohamed, D.H.J. Epema, Communication-Aware Job-Placement Policies for the KOALA Grid Scheduler, *2nd IEEE Int'l Conf. on e-Science and Grid Computing*, dec. 2006.



Future work

- Use bandwidth and latency in job placements (lightpaths)
- Test KOALA and its policies in a heterogeneous environment (DAS2 + DAS3 + Grid5000 + ...)
- Deal with more application types (PSAs, ...)
- A decentralized P2P KOALA

Information

- **Publications**

- see PDS publication database at www.pds.ewi.tudelft.nl

- **Web sites:**

- Projects: www.pds.ewi.tudelft.nl/~epema
- KOALA: www.st.ewi.tudelft.nl/koala
- DAS3: www.cs.vu.nl/das3
- VL-e: www.vl-e.nl

