AlGorille: Algorithms for the Grid

Jens Gustedt

INRIA Nancy – Grand Est
AlGorithmes pour la Grille

INRIA project team since 2007

The Team

Permanents:

S. Contassot-Vivier	(Pr. UL)	
J. Gustedt	(DR INRIA)	
M. Quinson	(As. Pr. UL)	
L. Nussbaum	(As. Pr. UL)	Nancy

Associates:

S. Vialle	(Pr. SUPÉLEC)	Metz
S. Genaud	(As. Pr. U Stbg)	
J. Gossa	(As. Pr. U Stbg)	
		Strasbourg

The Team

2008

Permanents:

S. Contassot-Vivier (Pr. UL)
J. Gustedt (DR INRIA)
M. Quinson (As. Pr. UL)
L. Nussbaum (As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud (As. Pr. U Stbg)
J. Gossa (As. Pr. U Stbg)

Strasbourg

The Team

2008–2012

Permanents:

S. Contassot-Vivier	(Pr. UL)
J. Gustedt	(DR INRIA)
M. Quinson	(As. Pr. UL)
L. Nussbaum	(As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud	(As. Pr. U Stbg)
J. Gossa	(As. Pr. U Stbg)

Strasbourg

The Team

2008–2012

Permanents:

S. Contassot-Vivier (Pr. UL)
J. Gustedt (DR INRIA)
M. Quinson (As. Pr. UL)
L. Nussbaum (As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud (As. Pr. U Stbg)
J. Gossa (As. Pr. U Stbg)

Strasbourg

The Team

2008–2012

Permanents:

S. Contassot-Vivier (Pr. UL)
J. Gustedt (DR INRIA)
M. Quinson (As. Pr. UL)
L. Nussbaum (As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud (As. Pr. U Stbg)
J. Gossa (As. Pr. U Stbg)

Strasbourg

The Team

2008–2012

Permanents:

S. Contassot-Vivier (Pr. UL)
J. Gustedt (DR INRIA)
M. Quinson (As. Pr. UL)
L. Nussbaum (As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud (As. Pr. U Stbg)
J. Gossa (As. Pr. U Stbg)

Strasbourg

The Team

2012

Permanents:

S. Contassot-Vivier	(Pr. UL)
J. Gustedt	(DR INRIA)
M. Quinson	(As. Pr. UL)
L. Nussbaum	(As. Pr. UL)

Nancy

Associates:

S. Vialle (Pr. SUPÉLEC)

Metz

S. Genaud	(As. Pr. U Stbg)
J. Gossa	(As. Pr. U Stbg)

Strasbourg

The Team

End 2012

Permanents:

S. Contassot-Vivier	(Pr. UL)
J. Gustedt	(DR INRIA)
M. Quinson	(As. Pr. UL)
L. Nussbaum	(As. Pr. UL)

Nancy

Associates:

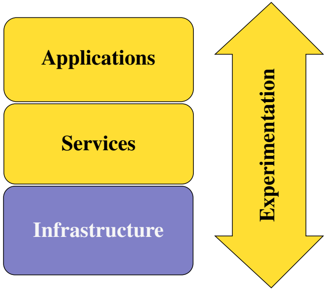
S. Vialle	(Pr. SUPÉLEC)

Metz

S. Genaud	(As. Pr. U Stbg)
J. Gossa	(As. Pr. U Stbg)

Strasbourg

Parallel and distributed computing: a layered approach



- Goal: Executing applications over a large-scale distributed infrastructure
- Generic services can be shared accross middleware solutions
- Our work: applications and services
- Experiments mandatory for validation

Outline

Outline

Structuring applications for scalability

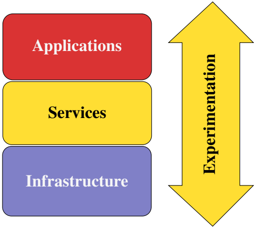
executive summary

Challenge

Efficient parallel applications for
hierarchical and heterogeneous systems

Our approach

- Programming Models high level, easy to use
- theoretical foundations and proofs
- efficient and portable realizations



Structuring applications for scalability

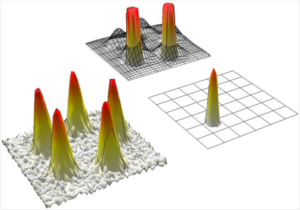
executive summary

Application related

- Collaborations with application domains:
  physics, geology, biology, medicine, machine learning, finance

- Large scale cellular automata for fine grained applications

  InterCell project
  (regional collaboration)

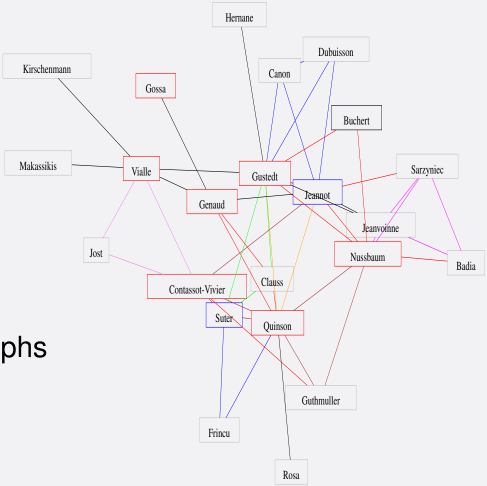


- Efficient linear algebra on accelerators
  A sparse linear solver for GPUs
  (*T. Jost, S. Contassot-Vivier and S. Vialle, book chapter, 2010*)

Structuring applications for scalability
executive summary

Foundations

- Convergence detection of asynchronous iterative computations (*VECPAR, 2008*)
- Random genesis of large graphs (*Physica A, 2011*)



Transparent Resource Management

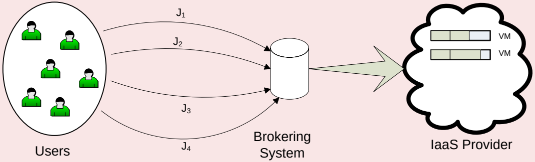
executive summary

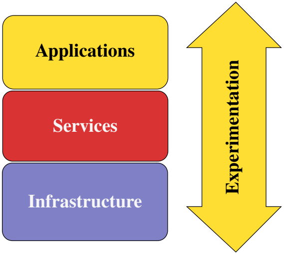
Challenge

Optimize resource usage for different actors with multiple objectives

Our approach:

1. Resource management algorithms
2. Implemented as services
3. Plugged into middleware





Transparent Resource Management

executive summary

Contributions:

- Fault tolerance: automatic and user-guided services (*IJPP, 2009*)
- Scheduling in unreliable environments: robustness vs. makespan (*IEEE TPDS*)
- Client-side resource provisioning: elastic Clouds (*CLOUD, 2011*)

Experimental validation

executive summary

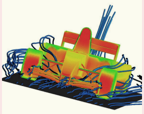
Challenge

Evaluation and validation of distributed algorithms and applications

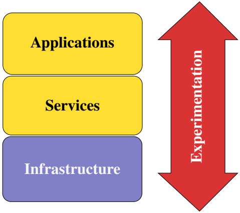
Testbed experiments

Grid'5000

Simulation



SIMGRID



Experimental validation

contributions – Experimentation on experimental testbeds

Grid'5000: one of the most advanced testbeds world-wide

- Major role in testbed design and evolutions
- Development and maintenance of key software (Kadeploy)

Advanced experimentation techniques

- Long experience on emulation (Wrekavoc → Distem)
- Preliminary work on orchestration of large experiments
- SCALE Challenge 2012 finalist
  4000 VM in 1 hour with Kadeploy on 668 Grid'5000 nodes

Experimental validation

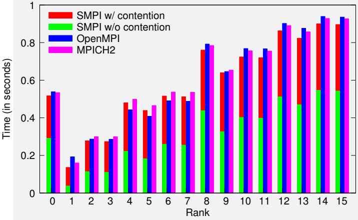
contributions – Simulation of Large-Scale Distributed Applications

SimGrid as a Scientific Instrument

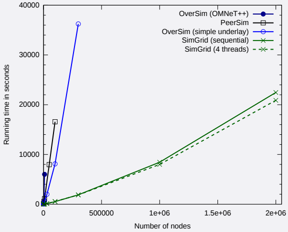
- Tool in Top 3 World-wide for Grids and P2P studies
- We are leading this project (in collab with Mescal and Avalon)
  - Other Inria teams involved: Cepage, Mascotte, Ascola, Runtime, . . .

Simulation as a Scientific Object

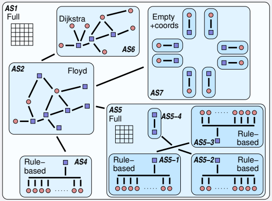
Validated



Scalable



Modular

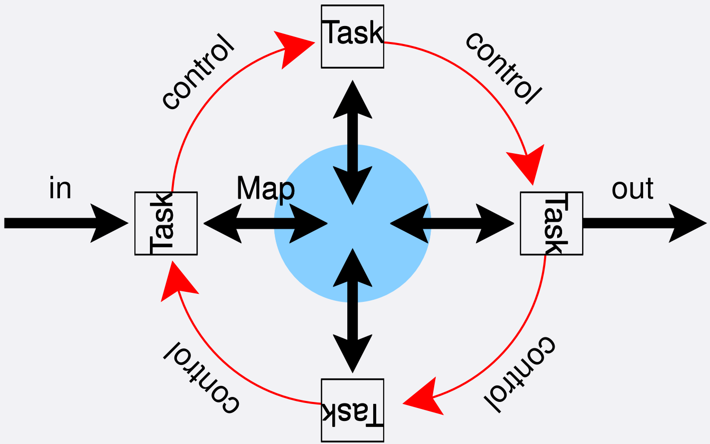


Outline

Focus I: Control- and Dataflow in Applications

problem statement

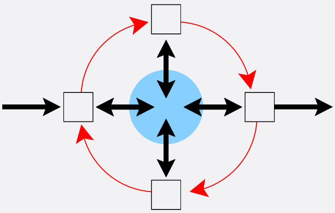
Control- and Dataflow: shared memory

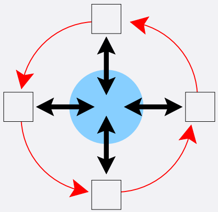


Focus I: Control- and Dataflow in Applications

problem statement

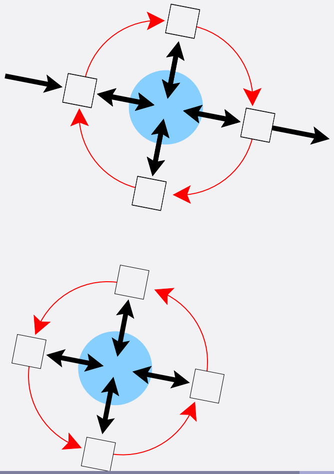
Control- and Dataflow: add message passing





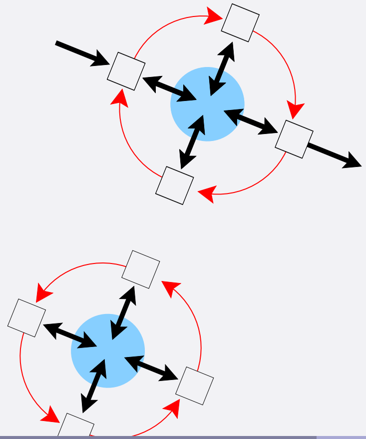
Focus I: Control- and Dataflow in Applications
problem statement

Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications
problem statement

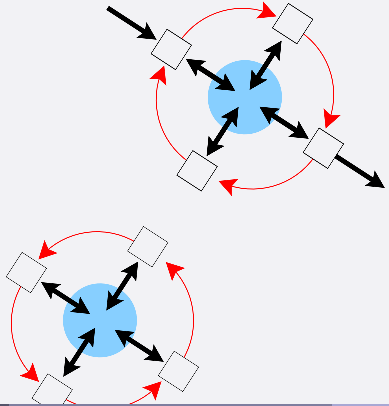
Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications

problem statement

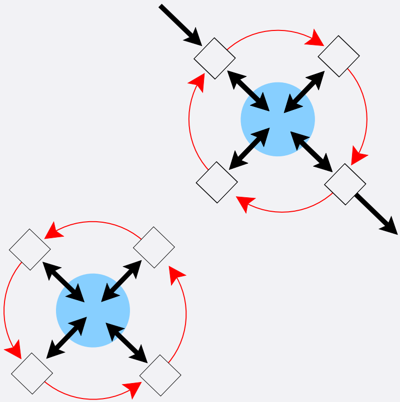
Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications

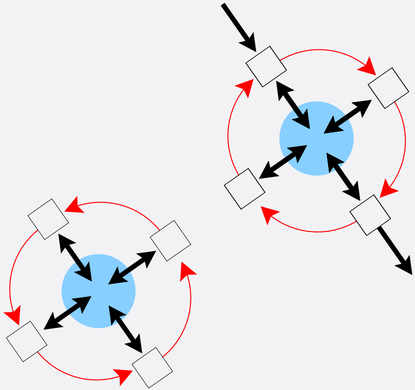
problem statement

Control- and Dataflow: add message passing



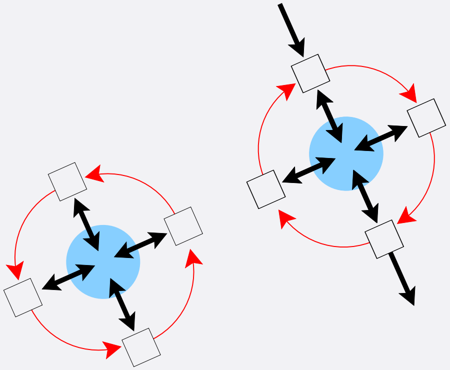
Focus I: Control- and Dataflow in Applications
problem statement

Control- and Dataflow: add message passing



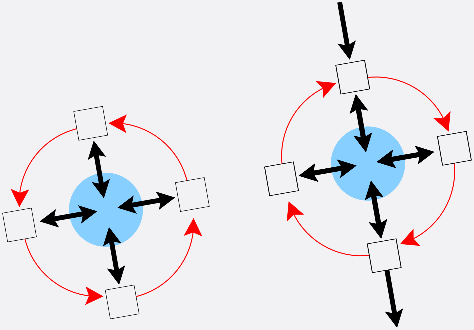
Focus I: Control- and Dataflow in Applications
problem statement

Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications
problem statement

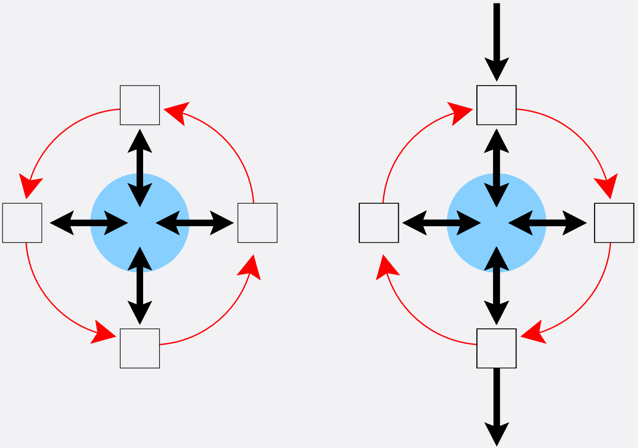
Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications

problem statement

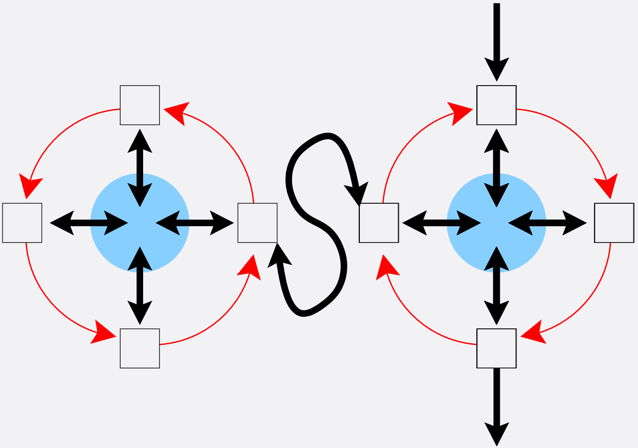
Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications

problem statement

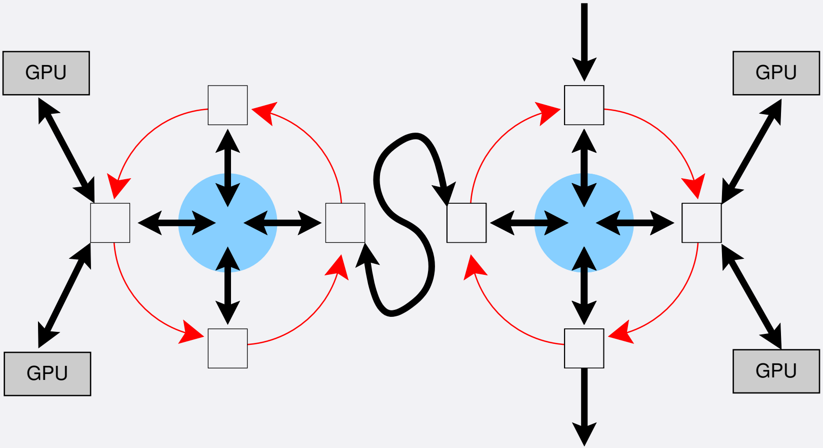
Control- and Dataflow: add message passing



Focus I: Control- and Dataflow in Applications

problem statement

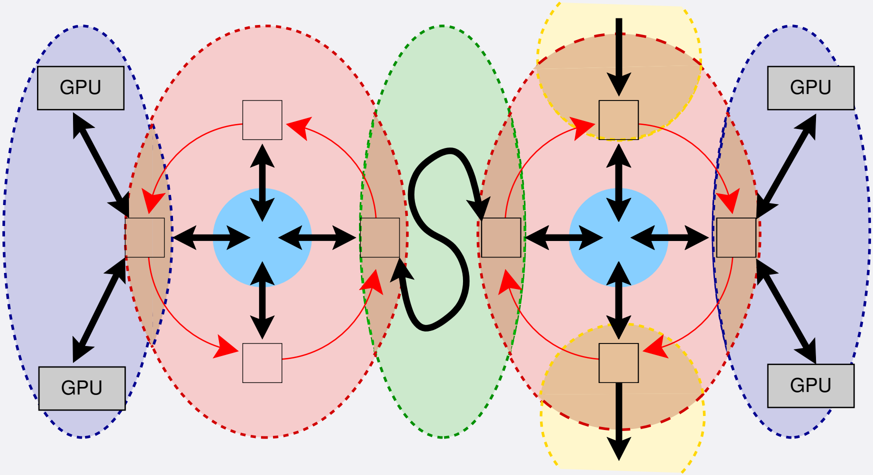
Control- and Dataflow: add accelerators or other devices



Focus I: Control- and Dataflow in Applications

problem statement

Control- and Dataflow: several domains of control



Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is

deadlock-free and fair.

Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is ▢

deadlock-free and fair.

Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

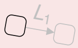
- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is

deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

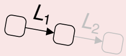
A unifying model: Ordered Read-Write Locks – ORWL

- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

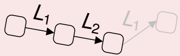
- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is

deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

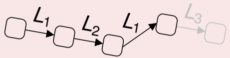
A unifying model: Ordered Read-Write Locks – ORWL

- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

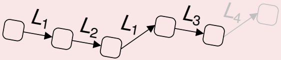
- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is

deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

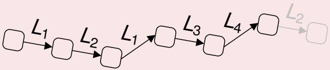
A unifying model: Ordered Read-Write Locks – ORWL

- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

contribution

A unifying model: Ordered Read-Write Locks – ORWL

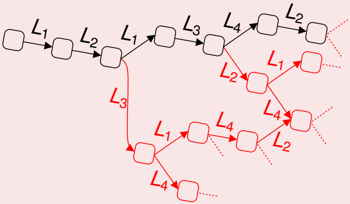
- a resource comprises location, control and data access
- access is regulated: handles, FIFO, read or write
- the typical sequence of access is

  `request` $\Longrightarrow$ `acquire` $\Longrightarrow$ `map` $\Longrightarrow$ `release`
- precise notions of iterative access and critical section

Theorem (Clauss & Gustedt 2010)

Any iterative ORWL algorithm is

deadlock-free and fair.



Focus I: Control- and Dataflow in Applications

ongoing and beyond

A portable implementation: Ordered Read-Write Locks – ORWL

Simple: only a handful concepts, a local view per task

Efficient: good overlap, low overhead

Portable: modern C, C11 (atomics, threads), sockets

Compatible: OpenMP, user threads, CUDA

Extensible: accelerators (CUDA in the works), devices (camera)

Application

Focus I: Control- and Dataflow in Applications
ongoing and beyond

A portable implementation: Ordered Read-Write Locks – ORWL

Application

2011- American Option Pricer on GPU cluster (S. Vialle & L. Abbas-Turki)

2012- Radiative transfer equations (F. Asllanaj)

2013- HPC for numerical simulations of plasma physics
LabEx IRMIA, Strasbourg (CALVI/TONUS)

2014- integrate to system level
INRIA Lab on multi-cores and accelerators (CAMUS)

Focus 2: Simulated MPI (SMPI)

problem statement

Goal: Online simulation of unmodified MPI application within SimGrid

- Algorithm prototyping; Platform dimensionning; What-if analysis ...

PB 1: Enable this mode of MPI execution

- (partially) Reimplement MPI on top of SimGrid
- Fold MPI processes as threads
- Allow to manually factorize data memory

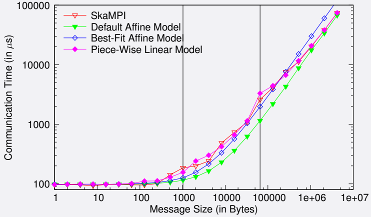
PB 2: Useless if not realistic enough

- Improve model $\rightsquigarrow$ piece-wise linear model
  Accurate also for small messages
- Preserve good modeling of network contention

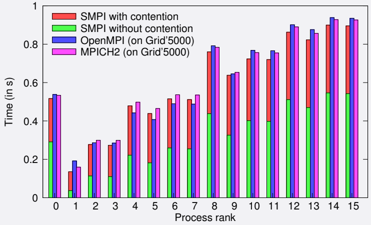
Focus 2: Simulated MPI (SMPI)

contribution

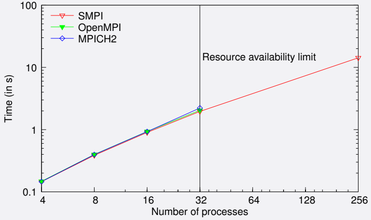
Realism: Point to Point



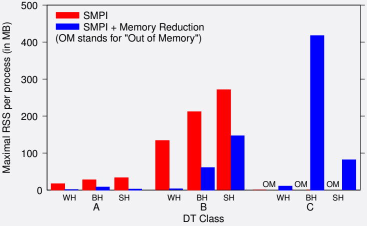
Scatter 16 procs



Scalability: Scatter (4 MiB/msg)



Reduced Footprint of DT



Focus 2: Simulated MPI (SMPI)

ongoing and beyond

Improve the enabling of MPI simulation

- Simulate 10^6 MPI Linpack processes within SimGrid?
- Distribute simulation to achieve this size-up

Push the validity limit further

- Model global communications (OpenMPI vs. MPICH2)
- Model CPU and memory performance (with MESCAL team)

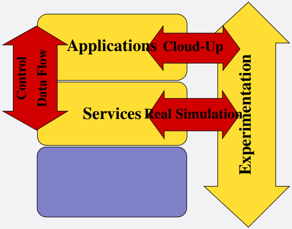
Vision

- Be the best alternative to simulate ExaScale Systems
- ANR SONGS project coordinates these efforts (7 INRIA teams)

Outline

A lot of Perspectives

integrate modeling, algorithm design and experiments



mutual feedback

1. "applicative control flow" and "scheduling"
2. service API and online simulation
3. performance guarantees on all scales

organizational framework

Structuring applications for scalability

perspectives

Seamless Efficiency Engineering at All Scales

- Provide an integrated view of application programming:

  from efficient programming on a PC ...

  ... HPC on a large scale

- Don't forget the users in the middle
  - can't afford to launch an industrial project for each application
  - have them use a cluster, mainframe, Cloud ... occasionally

Seamless Efficiency Engineering at All Scales

- Concentrate our efforts around Control- and Dataflow (ORWL)
  - develop and extend API application programming
  - connect ORWL into the Cloud
  - connect ORWL to the system level

Transparent Resource Management
perspectives

Transparent Cloud Resources Brokering

- Operate clouds as black boxes, on behalf of the client
  - From usage to price
  - Comprehensive run management
- Key Issues
  - Adaptive resource provisioning strategies
  - Characterization of user workloads in Clouds
  - Complex cost models

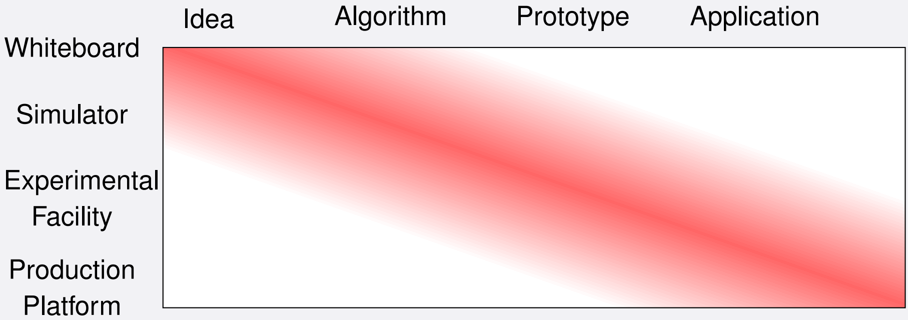
Cloud Exploitation

- Cloud simulation: SimCloud to interface with SimGrid analyze complex setups through simulation
- HPC-Cloud: Experiment and adapt our effort for HPC to Clouds

Experimental validation

perspectives

No Experimental Methodology is Sufficient: We Need Them All

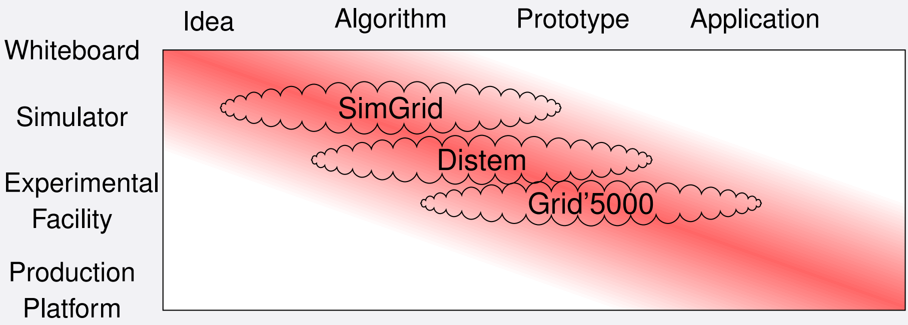


One Workbench to Rule Them All

- Leverage our expertise on all methodologies to combine them

Experimental validation
perspectives

No Experimental Methodology is Sufficient: We Need Them All

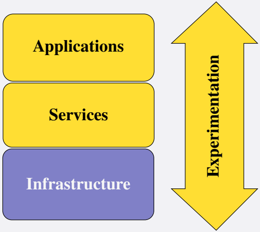


One Workbench to Rule Them All

- Leverage our expertise on all methodologies to combine them

Conclusion

AlGorille project



Focus on Algorithms

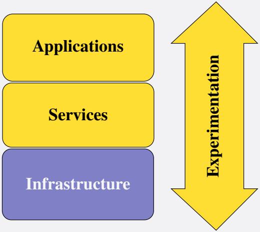
1. structuring application for scalability
2. transparent resource management
3. experimental validation

AlGorille team

- young dynamic team, primarily university staff
- important engineering support by INRIA and ANR
- people on three sites (and it works!)
- University of Lorraine will be recruiting in 2013

Conclusion

AlGorille project



Focus on Algorithms

1. structuring application for scalability
2. transparent resource management
3. experimental validation

AlGorille team

- young dynamic team, primarily university staff
- important engineering support by INRIA and ANR
- people on three sites (and it works!)
- University of Lorraine will be recruiting in 2013