SimGrid/SMPI Status Report

Arnaud Legrand
CNRS/INRIA MESCAL/University of Grenoble

The SimGrid Team

INRIA-Illinois Joint Laboratory on Petascale Computing
June 11, 2013

SMPI - Introduction

What it is

- Partial Implementation of MPI on top of Simgrid
  http://simgrid.gforge.inria.fr/
- Aims at replacing transparently MPI libraries without changing user's code
- Open source and use sound validated models

Goal

- Simulate the behavior of applications on any kind of system/cluster
- Allow developers to debug (gdb, valgrind) their code on their laptop
- Allow scaling studies and understand platform parameters and limitations

Achievements

- 88 of the most used MPI functions are implemented at this time (mostly from MPI 1.1)
- 90+ different collective algorithms
- Execution mode #1: **Direct execution (online)**
  - Need to use the same compilation toolchain to avoid wrong estimations
  - Possibility to share memory between processes (saves memory)
  - Possibility to profile and inject timings for loops (saves time)
- Execution mode #2: **Trace injection (offline)**
  - Capture a trace with Tau/Paraver/...
  - Replay the trace in the simulator

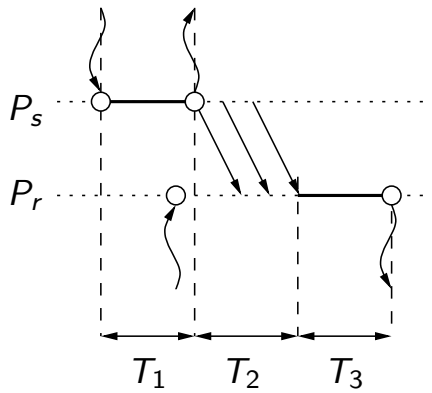
Outline

- Introduction
- Classical Network Models
- SMPI (In) Validation
  - Point-to-point Communications
  - Bandwidth Saturation and Topology
  - NAS PB
  - Real Application
- Conclusion

The LogP family[CKP+93, AISS95, KBV00, IFH01]

Such models were initially meant to **design algorithms**

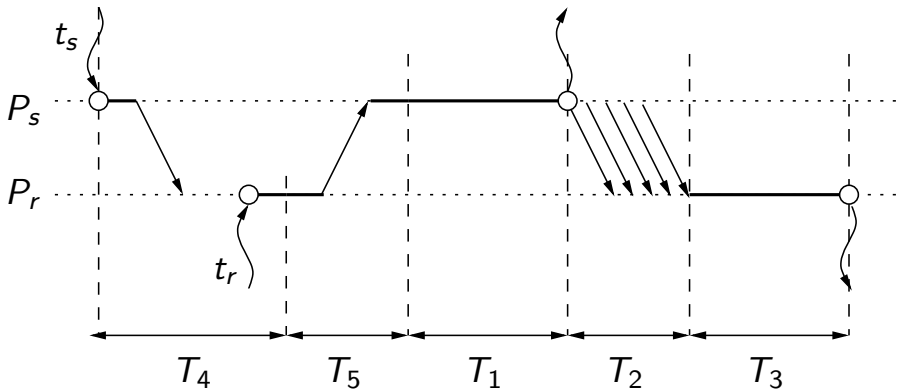
The LogGPS model[IFH01] in a nutshell



(a) Asynchronous mode ($k \leq S$).

Routine	Condition	Cost
`MPI_Send`	$k \leq S$	T_1
	$k > S$	$T_4 + T_5 + T_1$
`MPI_Recv`	$k \leq S$	$\max(T_1 + T_2 - (t_r - t_s), 0) + T_3$
	$k > S$	$\max(o + L - (t_r - t_s), 0) + o +$ $T_5 + T_1 + T_2 + T_3$
`MPI_Isend`		o
`MPI_Irecv`		o

(b) LogGPS modeling of MPI routine costs.



(c) Rendez-vous mode ($k > S$).

$$T_2 = \begin{cases} L + kg & \text{if } k < s \\ L + sg + (k-s)G & \text{otherwise} \end{cases}$$

$$T_1 = o + kO_s \qquad T_3 = o + kO_r$$

$$T_4 = \max(L + o, t_r - t_s) + o \qquad T_5 = 2o + L$$

(d) Partial piecewise linear models

Flow-level Model

Flow-level models A communication (flow) is simulated as a single entity

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Flow-level Model

Flow-level models A communication (flow) is simulated as a single entity

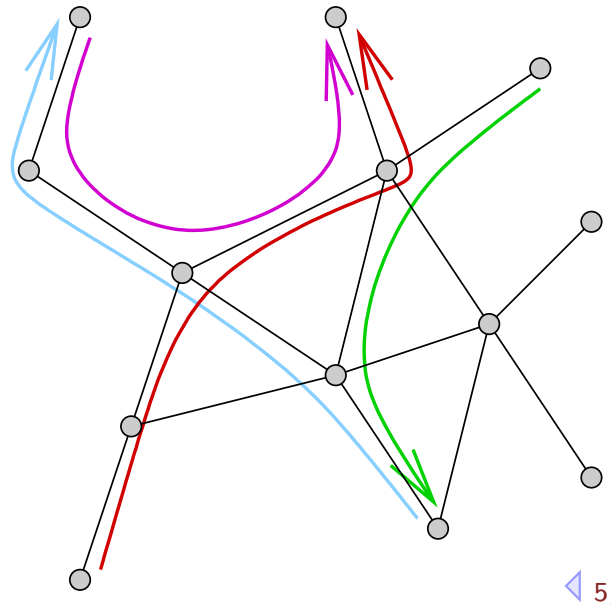
$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\sum_{\text{if flow i uses link j}} \rho_i \leq C_j$



Flow-level Model

Flow-level models A communication (flow) is simulated as a single entity

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

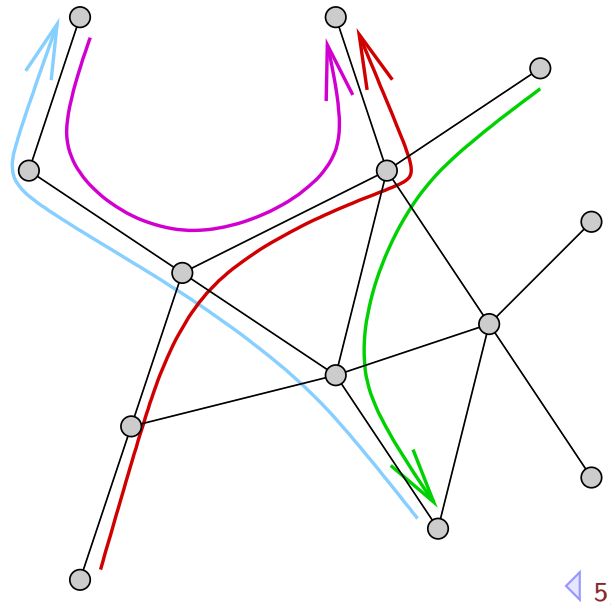
Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\sum_{\text{if flow i uses link j}} \rho_i \leq C_j$

Objective function

- Max-Min $\max(\min(\rho_i))$
- or other fancy objectives
  e.g., Reno $\sim \max(\sum \log(\rho_i))$



Flow-level Model

Flow-level models A communication (flow) is simulated as a single entity

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Assume steady-state and **share bandwidth** every time a new flow appears or disappears

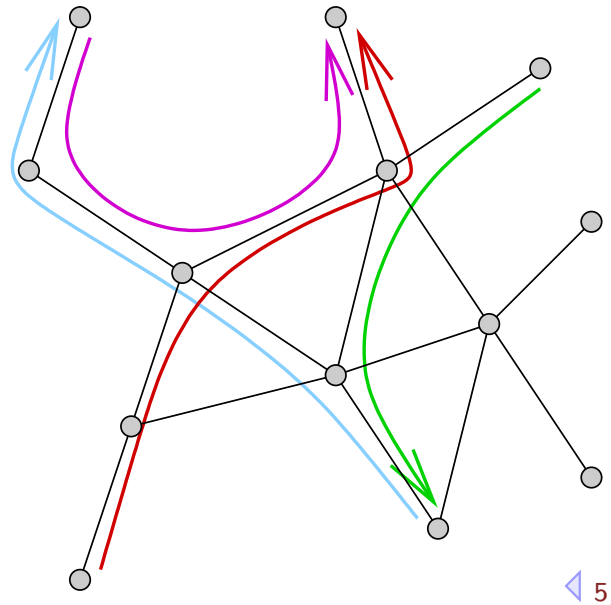
Setting a set of flows $\mathcal{F}$ and a set of links $\mathcal{L}$

Constraints For all link j: $\sum_{\text{if flow i uses link j}} \rho_i \leq C_j$

Objective function

- Max-Min $\max(\min(\rho_i))$
- or other fancy objectives
  e.g., Reno $\sim \max(\sum \log(\rho_i))$

Seamlessly account for topology



SimGrid Validity Results until 2011

SimGrid validity: Research focus since 2002

- 2002 *Sound model* proposed ⇒ Validity *checked on a few simple scenarios*.
- 2007- *Error evaluation* starts ⇒ Identify (and solve) model's weaknesses

SimGrid Validity Results until 2011

SimGrid validity: Research focus since 2002

- 2002 *Sound model* proposed $\Rightarrow$ Validity *checked on a few simple scenarios*.
- 2007- *Error evaluation* starts $\Rightarrow$ Identify (and solve) model's weaknesses

Settings: *Synthetic* App. + *Synthetic* WAN. Compare against *GTNetS*

- Errors were hunted down + unexpected phenomenon were understood
- Sharing mechanism from theoretical literature experimentally proved wrong

$\rightsquigarrow$ The model and its instanciation were considerably improved

- SimGrid and packet-level simulators now mostly diverge in extreme cases

SimGrid Validity Results until 2011

SimGrid validity: Research focus since 2002

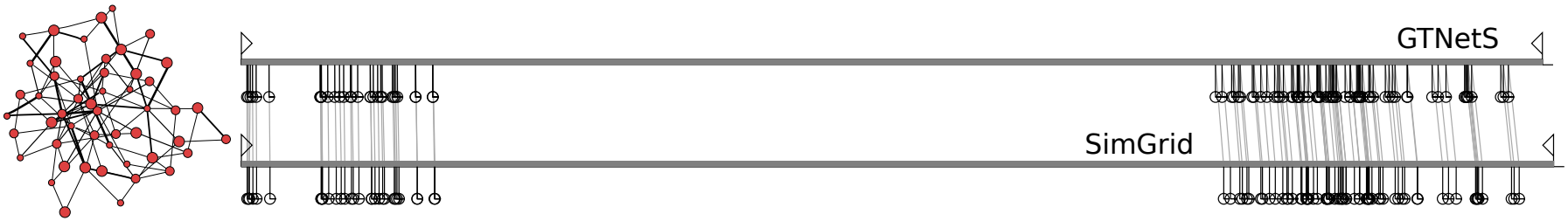
- 2002 *Sound model* proposed ⇒ Validity *checked on a few simple scenarios*.
- 2007- *Error evaluation* starts ⇒ Identify (and solve) model's weaknesses

Settings: *Synthetic* App. + *Synthetic* WAN. Compare against *GTNetS*

- Errors were hunted down + unexpected phenomenon were understood
- Sharing mechanism from theoretical literature experimentally proved wrong

⇝ The model and its instanciation were considerably improved

- SimGrid and packet-level simulators now mostly diverge in extreme cases



SimGrid Validity Results until 2011

SimGrid validity: Research focus since 2002

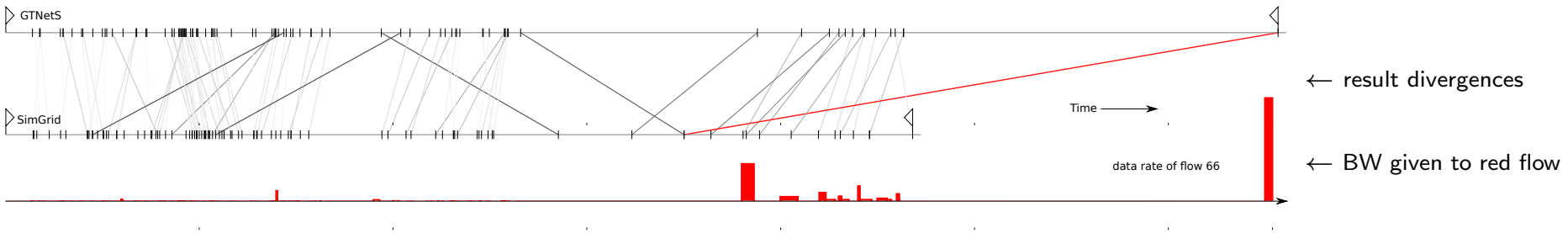
- 2002 *Sound model* proposed ⇒ Validity *checked on a few simple scenarios*.
- 2007- *Error evaluation* starts ⇒ Identify (and solve) model's weaknesses

Settings: *Synthetic* App. + *Synthetic* WAN. Compare against *GTNetS*

- Errors were hunted down + unexpected phenomenon were understood
- Sharing mechanism from theoretical literature experimentally proved wrong

⇝ The model and its instanciation were considerably improved

- SimGrid and packet-level simulators now mostly diverge in extreme cases



In this scenario, GTNetS and SG agree on termination date of most flows. The most diverging gets no bandwidth for a while although all others are done.

SimGrid Validity Results until 2011

SimGrid validity: Research focus since 2002

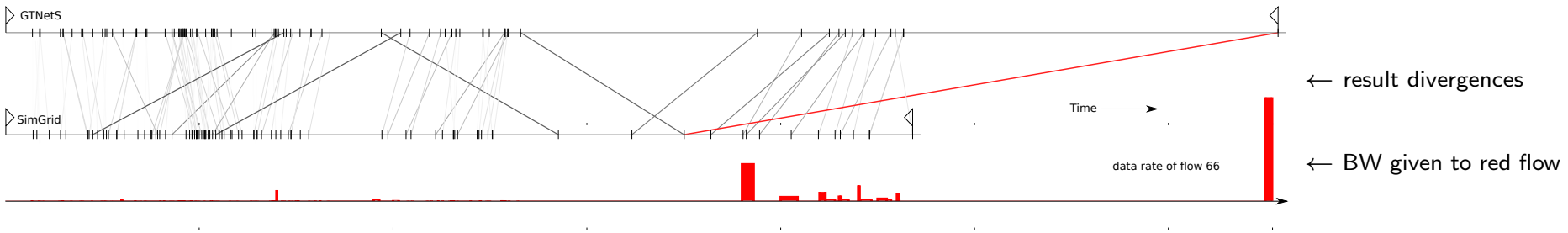
- 2002 *Sound model* proposed ⇒ Validity *checked on a few simple scenarios*.
- 2007- *Error evaluation* starts ⇒ Identify (and solve) model's weaknesses

Settings: *Synthetic* App. + *Synthetic* WAN. Compare against *GTNetS*

- Errors were hunted down + unexpected phenomenon were understood
- Sharing mechanism from theoretical literature experimentally proved wrong

⇝ The model and its instanciation were considerably improved

- SimGrid and packet-level simulators now mostly diverge in extreme cases



In this scenario, GTNetS and SG agree on termination date of most flows. The most diverging gets no bandwidth for a while although all others are done.

Such **fluid models can account** for TCP key characteristics

- slow-start
- flow-control limitation
- RTT-unfairness
- cross traffic interference

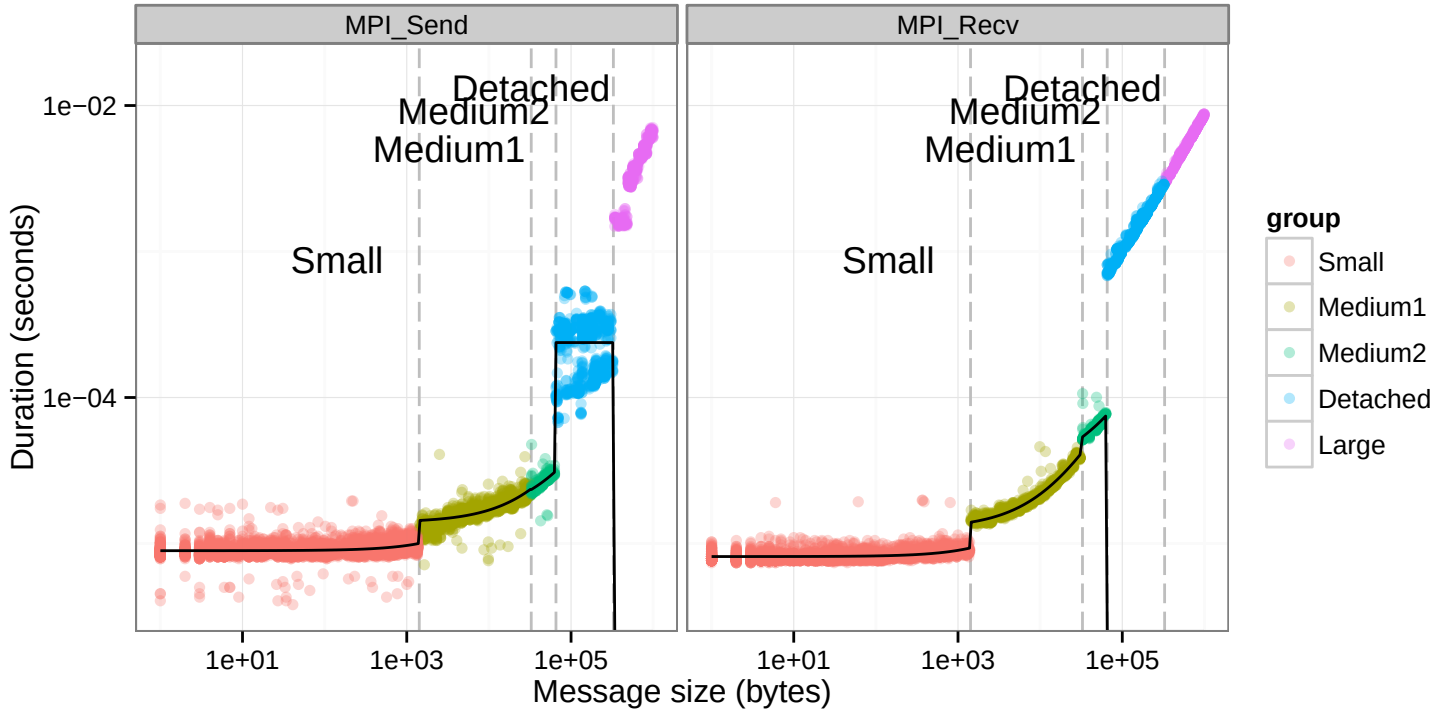
They are a very reasonable approximation for most LSDC systems

Outline

- Introduction
- Classical Network Models
- SMPI (In) Validation
  - Point-to-point Communications
  - Bandwidth Saturation and Topology
  - NAS PB
  - Real Application
- Conclusion

Reality is much more complicated than LogGPS

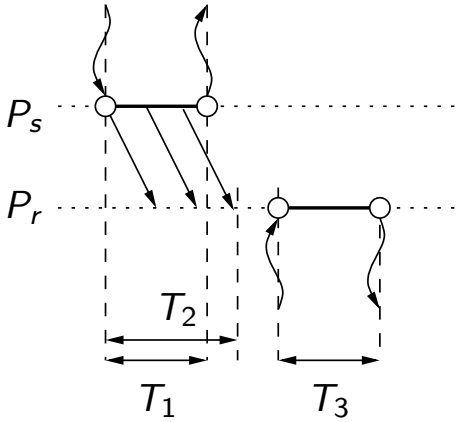
Focus on TCP for now as it seems challenging and would be useful in the European **Mont-Blanc** project (toward Exascale using low-power embedded technology).



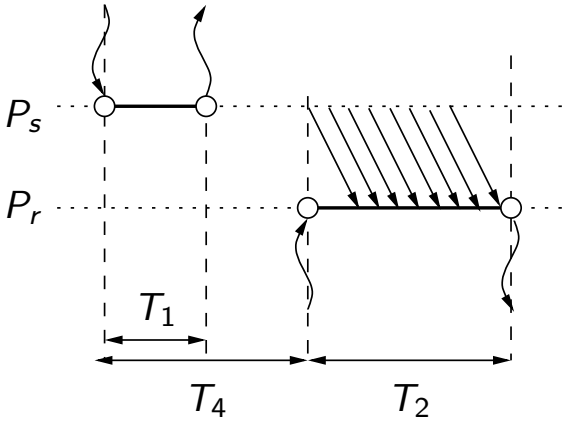
We do not assume such model and instead add what seems required and trim useless parts

SMPI model for P2P communication

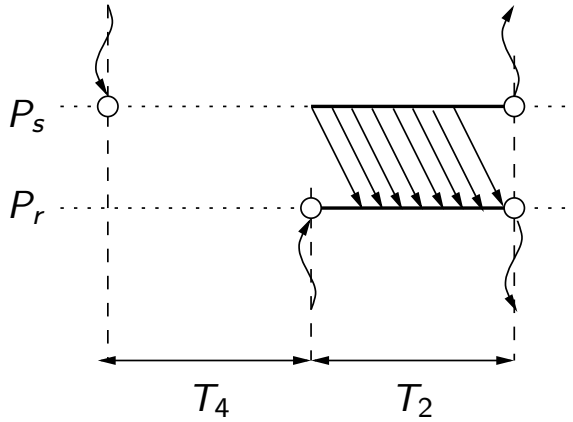
SMPI Asynchronous mode ($k \leq S_a$)



SMPI Detached mode ($S_a < k \leq S_d$)



SMPI Synchronous mode ($k > S_d$)



- Simple MPI program, 6 series of randomized tests, 1000 different messages sizes from 1 Byte to 1GB:
- R script automatically computes latency, bandwidth, timings and generates XML Simgrid platform parameters

```
<prop id="smpi/os" value="1:8.75118726019245e-06:7.09598480584951e-10;
  1420:1.38989305424406e-05:2.18111838119125e-10;
  65536:0.000193970854779561:-4.82025737428887e-11;
  327680:0:0"/>
...
<prop id="smpi/async_small_thres" value="65536"/>
<prop id="smpi/send_is_detached_thres" value="327680"/>
```

Not doing so can be particularly harmful

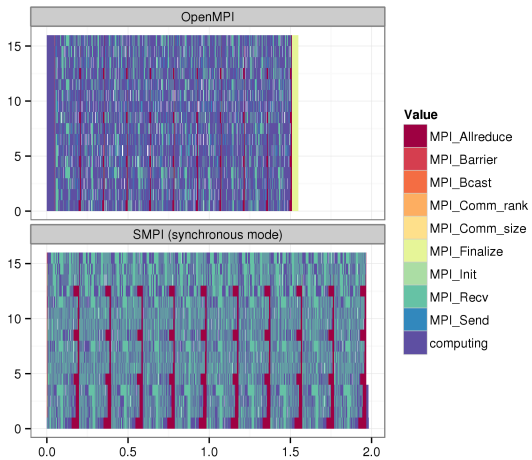
MPI Oddities and Cluster Peculiarities

- Protocol switch (1500, 65k, 327k, ...),
- Noisy areas and complex synchronization
- New distinctions (e.g., MPI_Send vs. MPI_Isend for small messages) appear when changing cluster
- Weird SendRecv behavior in the middle phase of pairwise AllToAll

Not doing so can be particularly harmful

MPI Oddities and Cluster Peculiarities

- Protocol switch (1500, 65k, 327k, ...),
- Noisy areas and complex synchronization
- New distinctions (e.g., MPI_Send vs. MPI_Isend for small messages) appear when changing cluster
- Weird SendRecv behavior in the middle phase of pairwise AllToAll



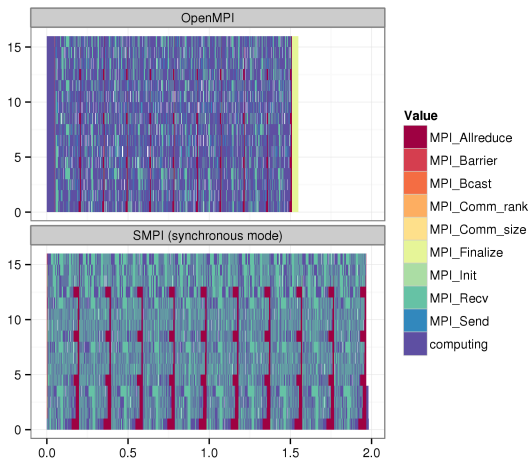
1

1. Need to account for **eager** mode!

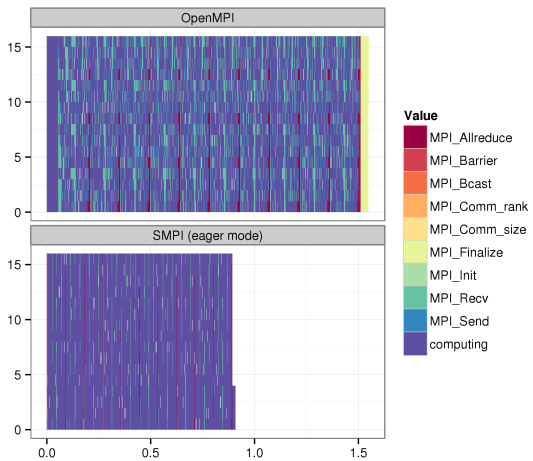
Not doing so can be particularly harmful

MPI Oddities and Cluster Peculiarities

- Protocol switch (1500, 65k, 327k, ...),
- Noisy areas and complex synchronization
- New distinctions (e.g., MPI_Send vs. MPI_Isend for small messages) appear when changing cluster
- Weird SendRecv behavior in the middle phase of pairwise AllToAll



1



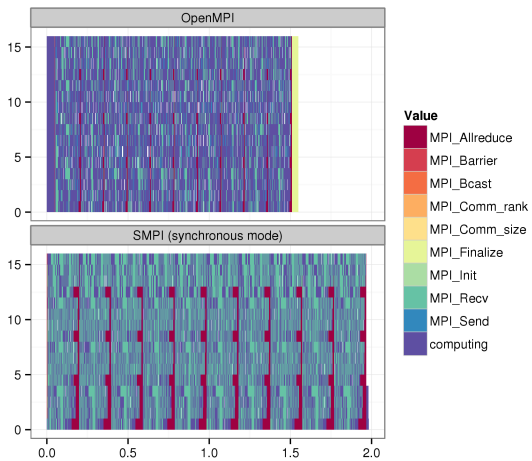
2

1. Need to account for **eager** mode!
2. The **overhead** of syscalls and memory copies is not negligible

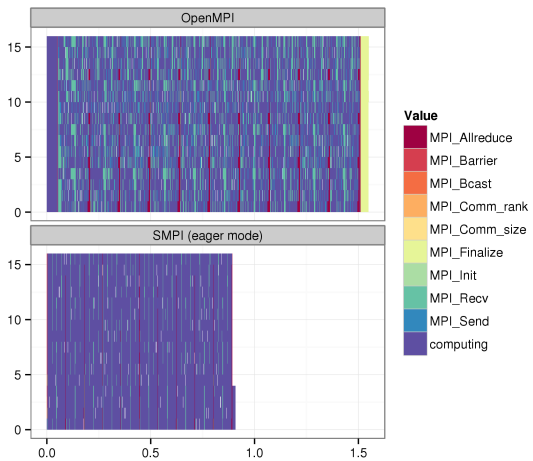
Not doing so can be particularly harmful

MPI Oddities and Cluster Peculiarities

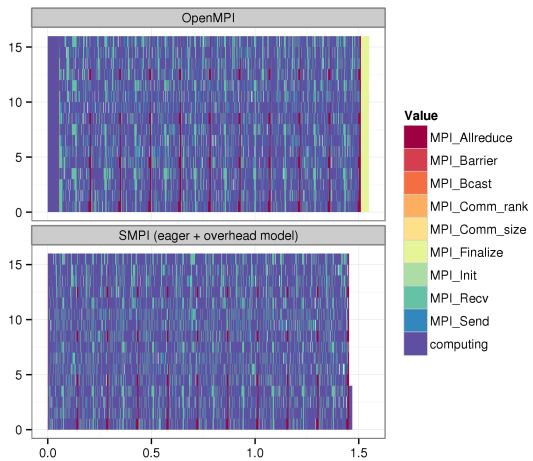
- Protocol switch (1500, 65k, 327k, ...),
- Noisy areas and complex synchronization
- New distinctions (e.g., MPI_Send vs. MPI_Isend for small messages) appear when changing cluster
- Weird SendRecv behavior in the middle phase of pairwise AllToAll



1



2



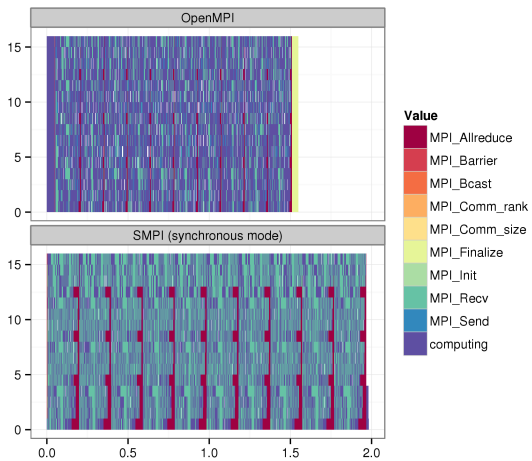
3

1. Need to account for **eager** mode!
2. The **overhead** of syscalls and memory copies is not negligible
3. Ok now but simple modeling error $\rightsquigarrow$ gross inaccuracies

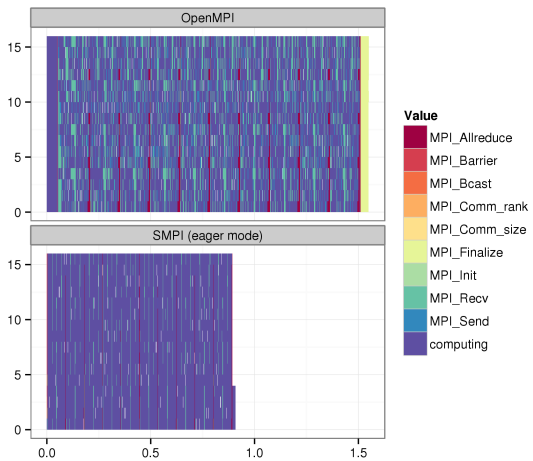
Not doing so can be particularly harmful

MPI Oddities and Cluster Peculiarities

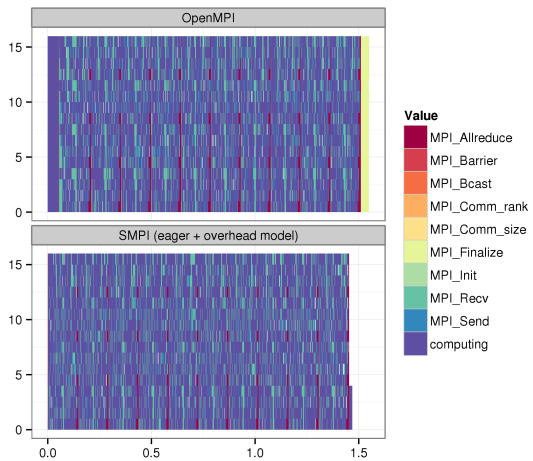
- Protocol switch (1500, 65k, 327k, ...),
- Noisy areas and complex synchronization
- New distinctions (e.g., MPI_Send vs. MPI_Isend for small messages) appear when changing cluster
- Weird SendRecv behavior in the middle phase of pairwise AllToAll



1



2



3

1. Need to account for **eager** mode!
2. The **overhead** of syscalls and memory copies is not negligible
3. Ok now but simple modeling error $\rightsquigarrow$ gross inaccuracies

Hiding errors is easy: consider makespan only and overfit model parameters

Modeling Saturation on G5K cluster

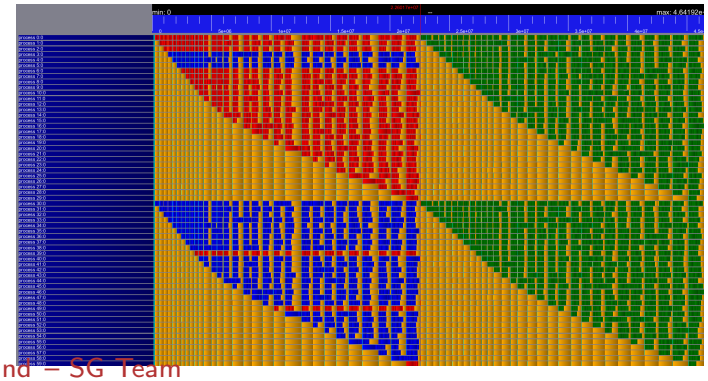
Experimental Setup

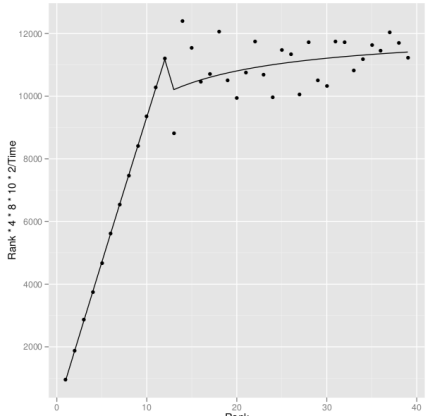
We used the graphene cluster of the Grid'5000 experimental testbed:

- 144 2.53GHz Quad-Core Intel Xeon x3440 nodes
- Four cabinets interconnected by a hierarchy of 10 Gigabit Ethernet switches

Main issue

- Simple collective operations are not too sensitive to bandwidth saturation
- AllToAll stress the network all way long
- Contention may occur within or between cabinets
- Identified issues:
  - Only 65% of max bandwidth (fullduplex $2B$) with `MPI_SendRecv`
  - No saturation within cabinets but similar limitation between cabinets
  - Nodes and cabinet interconnection have three links: up, down, limiter

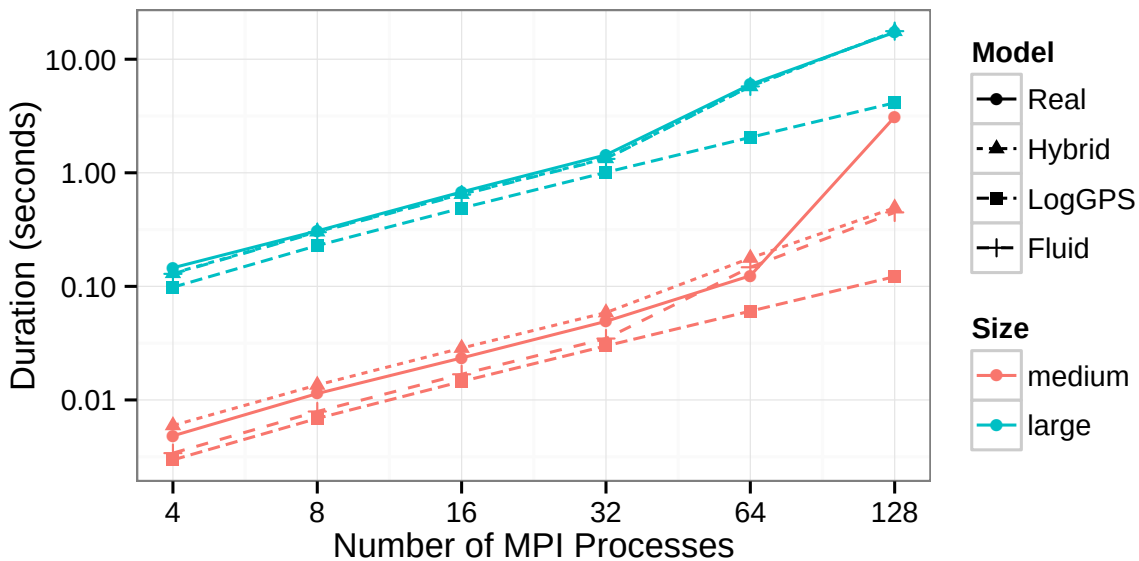




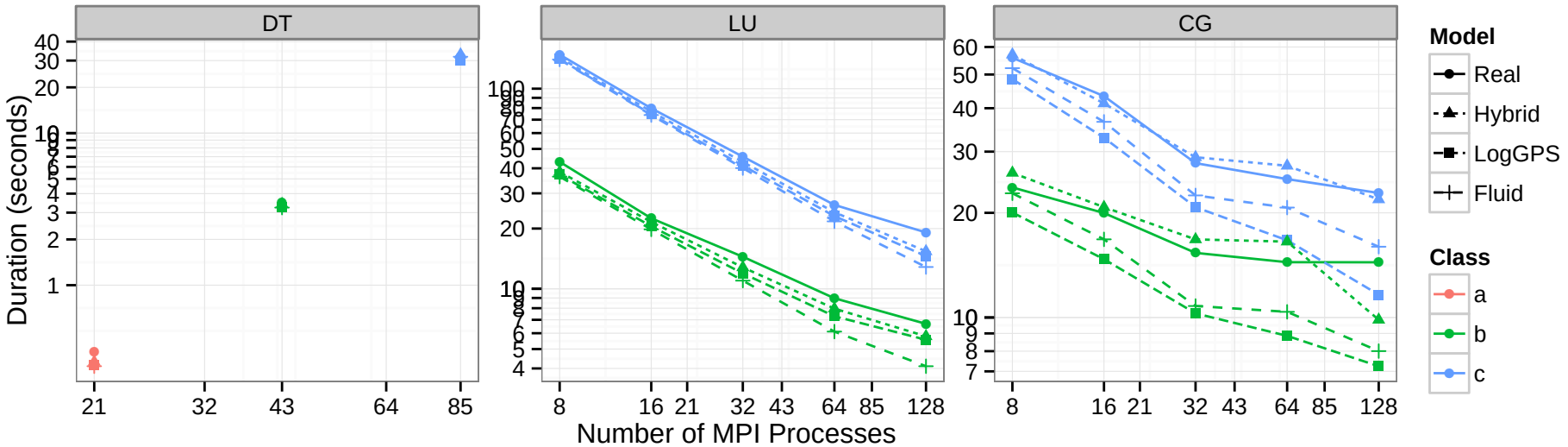
Modeling Collective Communications

Some projects propose to use simple analytic formula. This seems a little naive.

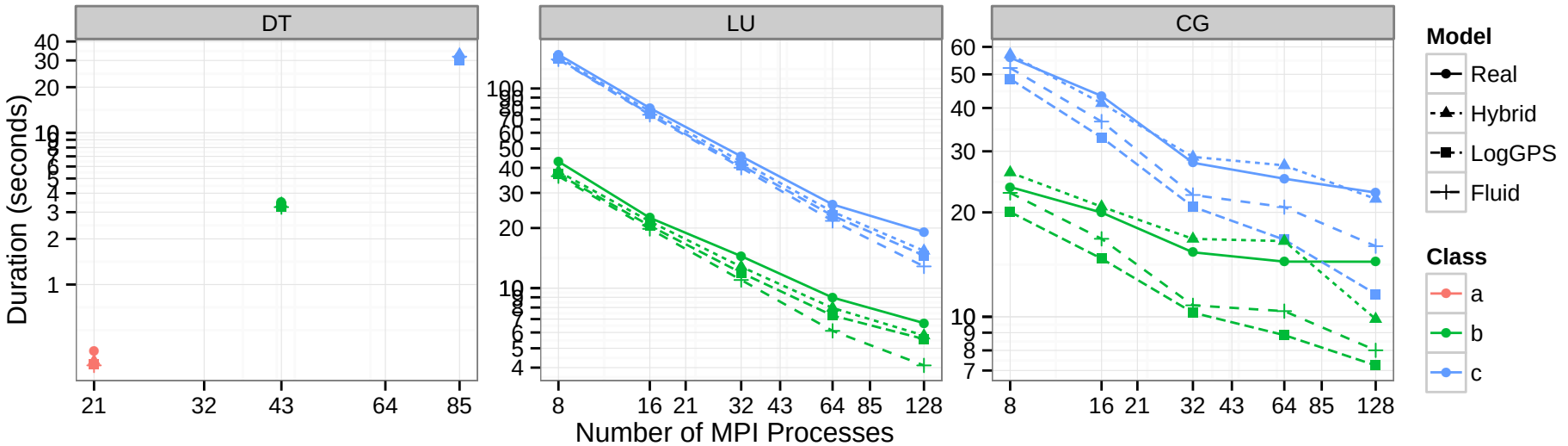
- Real MPI applications use several implementations for each collective, and select the right one at runtime
  - 2300 lines of code for the AllReduce in OpenMPI!!!
- Our initial SMPI versions had only one simple implementation for each one (except alltoall, which had 3):
  - StarMPI: large collection of implementations for collectives, adaptative selector
  - SMPI now: StarMPI's collectives reused,
  - 90+ collective algorithms but only one selected at each run (no adaptation)
  - Future work: *steal* MPICH and OpenMPI selector

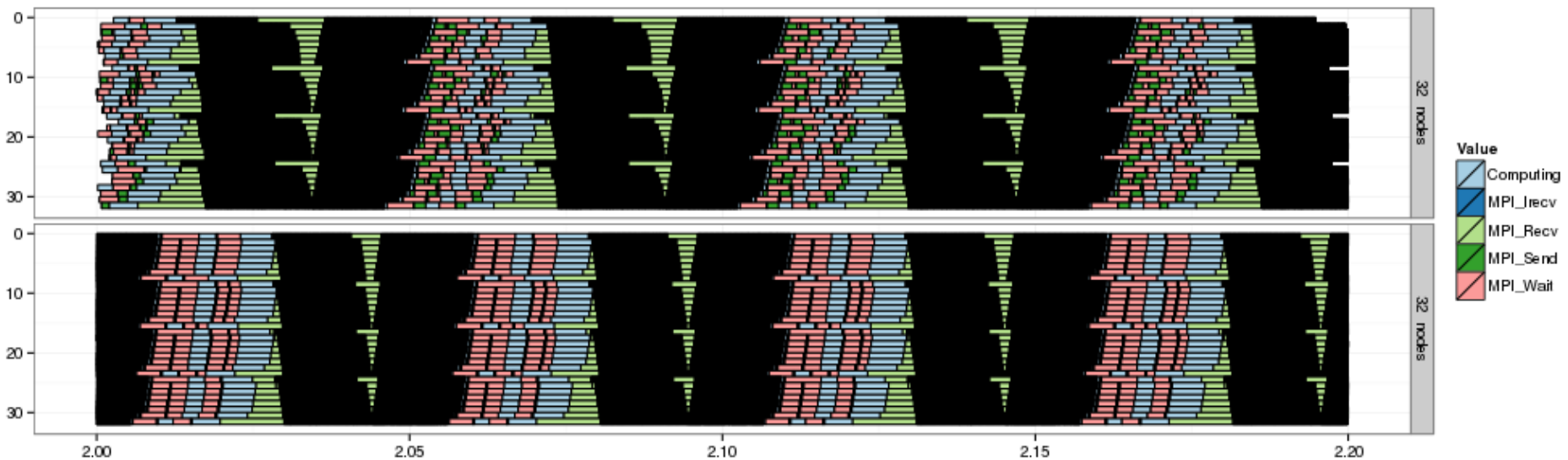


(In)Validation of SMPI with NAS PB



(In)Validation of SMPI with NAS PB

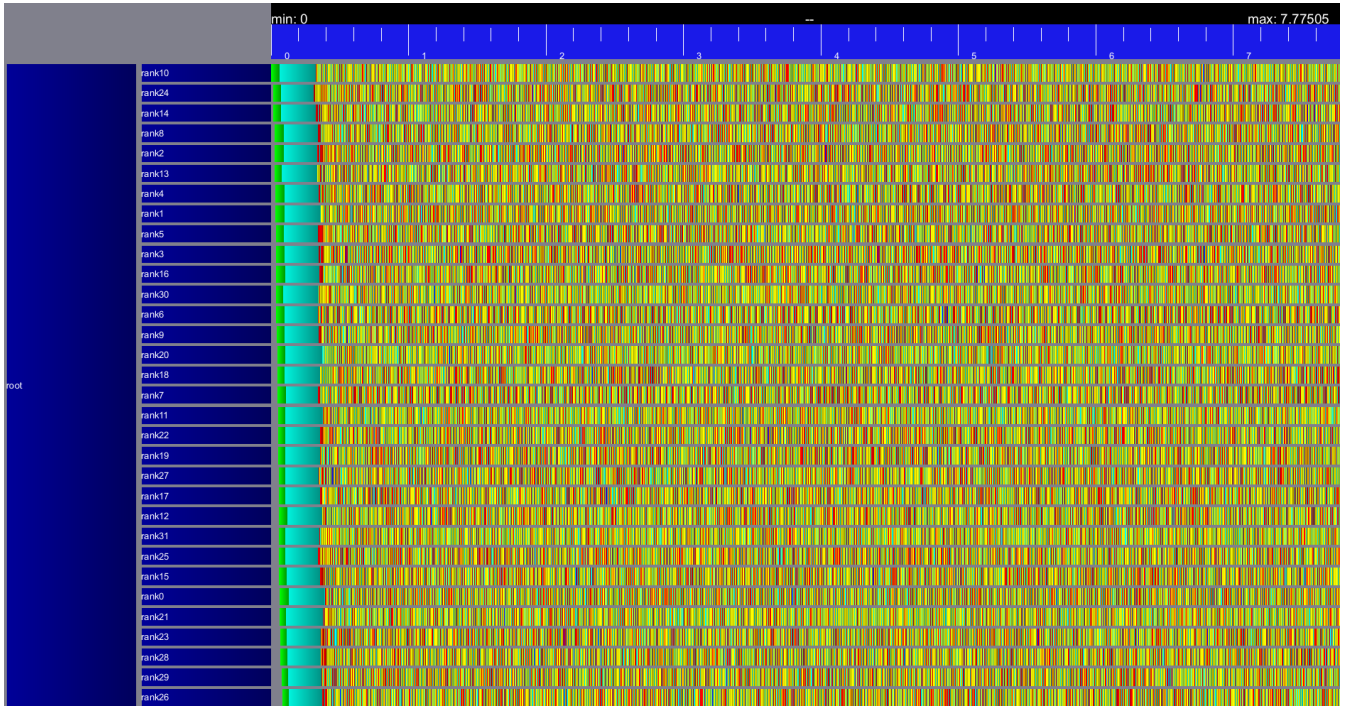




Zoom on 1 second of the LU benchmark with 32 processes

(In)Validation of Real Life with NAS PB

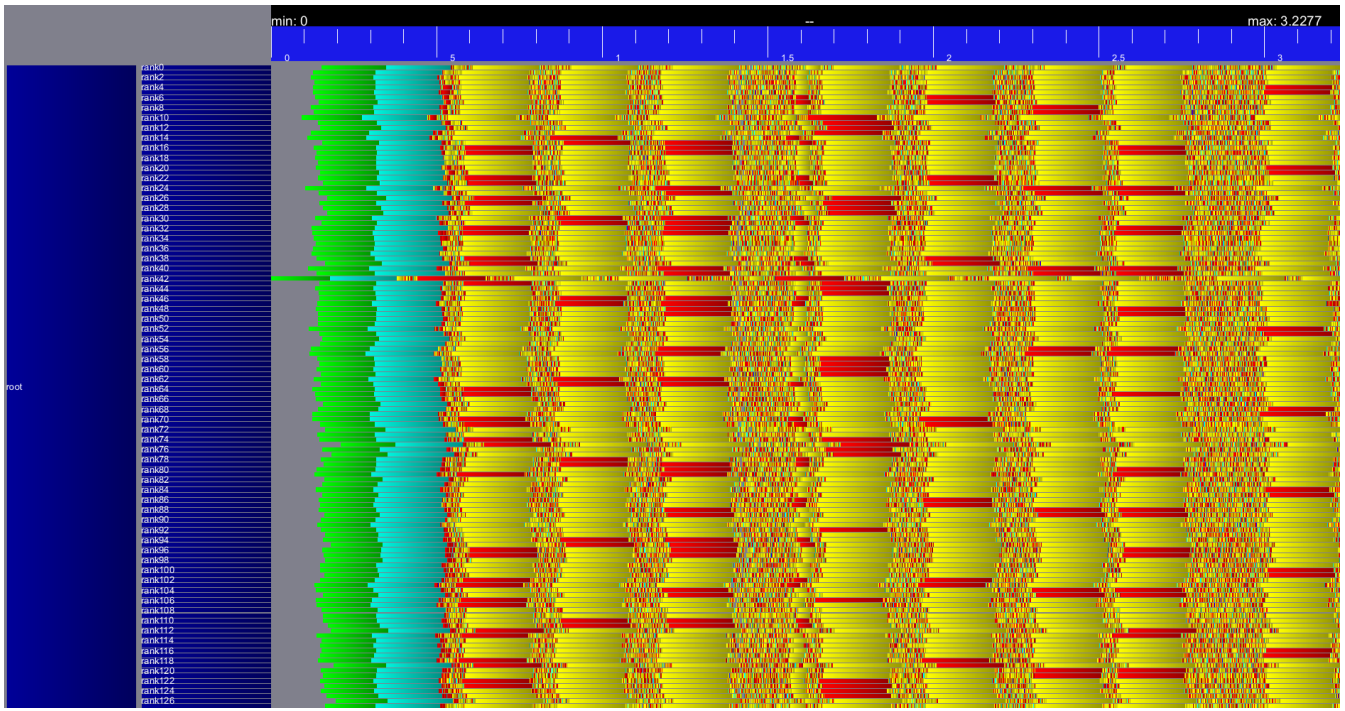
CG 32 nodes, red = send, yellow = wait



1. Communication time (32) $\approx$ a few micro seconds

(In)Validation of Real Life with NAS PB

CG 128 nodes, red = send, yellow = wait



1. Communication time (32) $\approx$ a few micro seconds
2. Communication time (128) $\approx$ sometimes 200 ms!!!
   - Occurs **24 times** leading to a delay of 4.86s out of 14.4s!!!
   - Removing it would lead to the correct estimation
   - Identified to be **TCP RTO** that also arise in the cloud context ("**TCP Incast** Throughput Collapse")

BigDFT

BigDFT in a nutshell

- Density Functional Theory (DFT) code (electronic structure simulation)
- Test application in the European Mont-Blanc project
- Heavily relies on collective operations

Online Simulation issues

- Global variables (Fortran Code, manual privatization with `openmp`), configuration files
- Get rid of computation checks (ruined by computation and memory folding)
- Use different set of collective operations depending on size, instance, ...

BigDFT

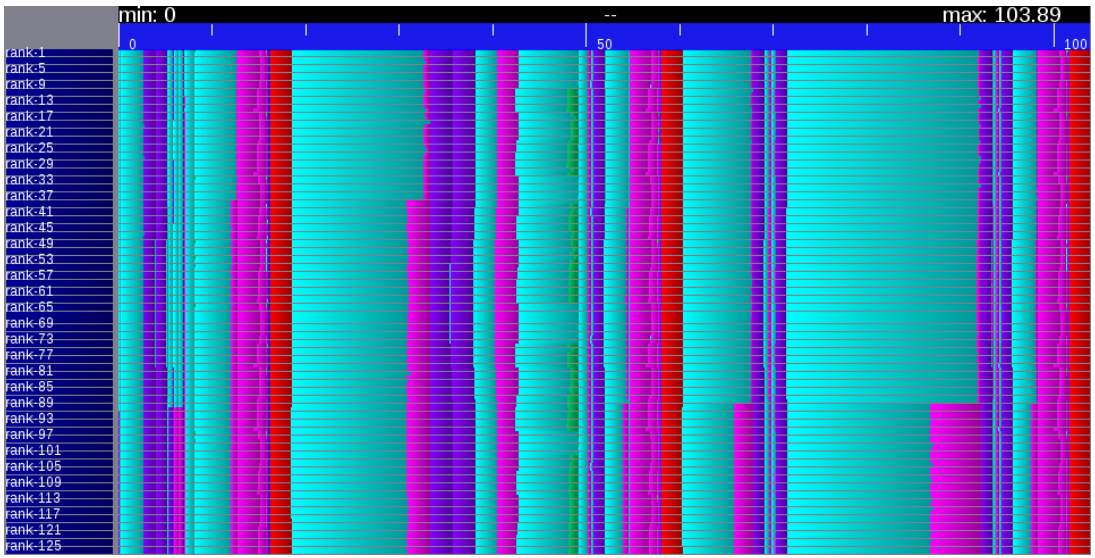
BigDFT in a nutshell

- Density Functional Theory (DFT) code (electronic structure simulation)
- Test application in the European Mont-Blanc project
- Heavily relies on collective operations

Online Simulation issues

- Global variables (Fortran Code, manual privatization with `openmp`), configuration files
- Get rid of computation checks (ruined by computation and memory folding)
- Use different set of collective operations depending on size, instance, ...

First results



Simulated run

BigDFT

BigDFT in a nutshell

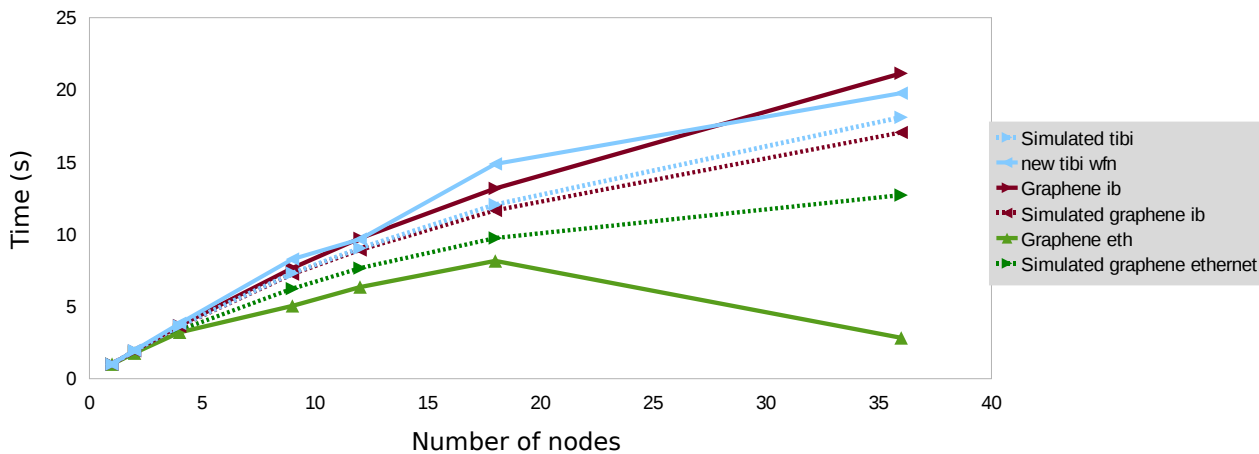
- Density Functional Theory (DFT) code (electronic structure simulation)
- Test application in the European Mont-Blanc project
- Heavily relies on collective operations

Online Simulation issues

- Global variables (Fortran Code, manual privatization with `openmp`), configuration files
- Get rid of computation checks (ruined by computation and memory folding)
- Use different set of collective operations depending on size, instance, ...

First results

- InfiniBand
- Tibidabo (Mont-Blanc ARM cluster with Ethernet 10G)



Scaling experiment

Outline

Future Work in SMPI

Main concerns of the SimGrid project

- Validity: Get realistic results (controlled experimental bias)
- Scalability: Simulate *fast enough* problems *big enough*
- Associated tools: campaign mgmt, result analysis, settings generation, ...
- Applicability: If it doesn't simulate what is important to you, it's void
- Open Source: We do our best for user support; Coding sprint a week ago

Future Work in SMPI

Main concerns of the SimGrid project

- Validity: Get realistic results (controlled experimental bias)
- Scalability: Simulate *fast enough* problems *big enough*
- Associated tools: campaign mgmt, result analysis, settings generation, ...
- Applicability: If it doesn't simulate what is important to you, it's void
- Open Source: We do our best for user support; Coding sprint a week ago

Important concerns for SMPI

- Tried to use a Reproducible Research approach. We need to set up a clean reproducible experimental workflow and a trace repository
- Test with other network models and other architectures and with shared memory
- Test with other applications: Sweep3D, Linpack, SpecFEM3D, ...
- Privatization is still not automatic, hence requires both SMPI and application expertise
- Noise characterization and deterministic injection
- Scale to more than only a few hundred nodes with real applications (we have real BigDFT traces up to 1024 nodes, infiniband)

SMPI and JLPC

Potential collaborations in the joint lab

LogGOPSim Loading GOAL/CDAG in SimGrid is trivial

- Would allow to simulate seamlessly network hierarchy and contention
- What about injecting system noise? Failures?
- Using SMPI to evaluate topology-aware collective communications?
- A torus network has been implemented by a Master at UIUC

SMPI and JLPC

Potential collaborations in the joint lab

LogGOPSim Loading GOAL/CDAG in SimGrid is trivial

- Would allow to simulate seamlessly network hierarchy and contention
- What about injecting system noise? Failures?
- Using SMPI to evaluate topology-aware collective communications?
- A torus network has been implemented by a Master at UIUC

BigSim Seems more resource demanding and our trials at running it were... painful

- Inject traces from BIGSim
- May want to benefit from hierarchical fluid network models
- Has nice tricks for variable privatization
- Handles distributed execution while it is WIP in SG

SMPI and JLPC

Potential collaborations in the joint lab

LogGOPSim Loading GOAL/CDAG in SimGrid is trivial

- Would allow to simulate seamlessly network hierarchy and contention
- What about injecting system noise? Failures?
- Using SMPI to evaluate topology-aware collective communications?
- A torus network has been implemented by a Master at UIUC

BigSim Seems more resource demanding and our trials at running it were... painful

- Inject traces from BIGSim
- May want to benefit from hierarchical fluid network models
- Has nice tricks for variable privatization
- Handles distributed execution while it is WIP in SG

Visualization tools (UFRGS, Brazil) need **new** tools with both spatial and temporal **aggregation** capabilities

http://simgrid.gforge.inria.fr/


Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman.
LogGP: Incorporating Long Messages Into the LogP Model – One Step Closer Towards a Realistic Model for Parallel Computation.
In *Proc. of the 7th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 95–105, Santa Barbara, CA, 1995.


David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken.
LogP: Towards a Realistic Model of Parallel Computation.
In *Proc. of the fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP)*, pages 1–12, San Diego, CA, 1993.


Fumihiko Ino, Noriyuki Fujimoto, and Kenichi Hagihara.
LogGPS: a Parallel Computational Model for Synchronization Analysis.
In *Proc. of the eighth ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming (PPoPP)*, pages 133–142, Snowbird, UT, 2001.


Thilo Kielmann, Henri E. Bal, and Kees Verstoep.
Fast Measurement of LogP Parameters for Message Passing Platforms.
In *Proc. of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, IPDPS '00, pages 1176–1183, London, UK, UK, 2000. Springer-Verlag.