Toward More Scalable Off-line Simulations of MPI applications

Frédéric Suter

IN2P3 Computing Center
Avalon, Inria, ENS Lyon

CCDSC
September 4, 2014

PICS 5473

What the Fox might want

- Analyze and understand the performance behavior of MPI applications
  - Detection of bottlenecks, load imbalance, undesired behaviors, . . .
- But also go further than what profiles allow for
  - Visualization, debugging, . . .
- At large scale
- And even on unavailable hypothetical configurations
  - Larger scale, different network interconnect or topology, . . .
- In a controlled and reproducible way, . . .

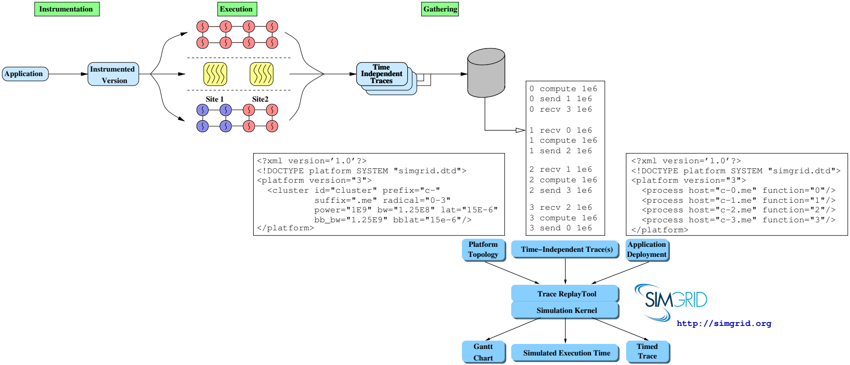
What the Fox might want

- Analyze and understand the performance behavior of MPI applications
  - Detection of bottlenecks, load imbalance, undesired behaviors, . . .
- But also go further than what profiles allow for
  - Visualization, debugging, . . .
- At large scale
- And even on unavailable hypothetical configurations
  - Larger scale, different network interconnect or topology, . . .
- In a controlled and reproducible way, . . .

- Hey, the fox needs scalable and accurate off-line simulation!!

Time-Independent Trace Replay in a Nutshell

- Project developed within the SimGrid framework
  - Available since release 3.8 (Oct. 2012)
- Custom instrumentation ⇒ traces without any time-related information
- Multiple scalable acquisition modes
- Replay based on SMPI (on-line simulation of MPI module)



```
0 compute 1e6
0 send 1 1e6
0 recv 3 1e6

1 recv 0 1e6
1 compute 1e6
1 send 2 1e6

2 recv 1 1e6
2 compute 1e6
2 send 3 1e6

3 recv 2 1e6
3 compute 1e6
3 send 0 1e6
```

```
<?xml version=’1.0’?>
<!DOCTYPE platform SYSTEM "simgrid.dtd">
<platform version="3">
  <cluster id="cluster" prefix="c-"
           suffix=".me" radical="0-3"
           power="1E9" bw="1.25E8" lat="15E-6"
           bb_bw="1.25E9" bblat="15e-6"/>
</platform>
```

```
<?xml version=’1.0’?>
<!DOCTYPE platform SYSTEM "simgrid.dtd">
<platform version="3">
  <process host="c-0.me" function="0"/>
  <process host="c-1.me" function="1"/>
  <process host="c-2.me" function="2"/>
  <process host="c-3.me" function="3"/>
</platform>
```

ScalaTrace in a Nutshell

- Project developed at North Caroline State University
  - Team led by Franck Müller
  - Current version: v2.2
- Advanced compression techniques
  - Detect repetitive patterns in regular codes
  - based on (recursive) RSDs (`RSD1: <100, MPI_Send1, MPI_Recv1>`)
  - Intra- and Inter-nodes compression
- Preserve structural information and temporal event order
- Store delta times in balanced histograms
- Support of irregular applications
  - Histograms for communication parameters too
- Several spin-off tools
  - ScalaExtrap: trace extrapolation from small instances
  - ScalaBenchGen: mock creation from actual traces
  - ScalaJack: memory aspects

Pros and Cons

Time-Independent Trace Replay

- ☺ Decouple acquisition from replay ⇒ Improves scalability
- ☺ Leverages SMPI network models ⇒ Improves accuracy
- ☹ Verbose trace format ⇒ Limits scalability
- ☹ Unique instruction rate for the whole application ⇒ Limits accuracy

ScalaTrace

- ☺ Ultra Compact trace format ⇒ Improves scalability
- ☺ Identifies sub-parts of the applications without extra-instrumentation ⇒ good for calibration
- ☹ No simulated replay ⇒ Limits scalability
- ☹ Timed traces ⇒ Limits acquisition to homogeneous platforms

This work

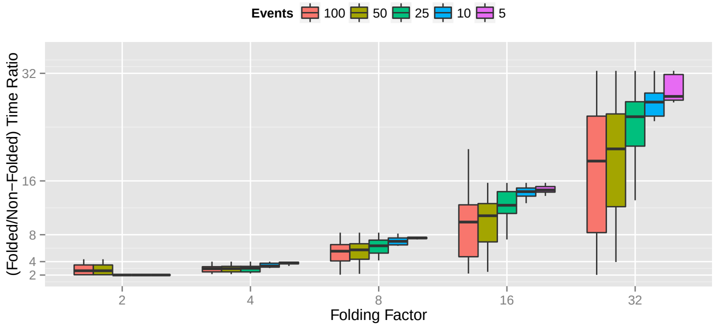
- Combine strenghts of both tools
  - Be limited neither by acquisition platform nor trace size
- Improve of calibration method
  - By leveraging the RSDs

Outline

- Motivation and Background
- Making ScalaTrace Time-Independent
  - Motivations
  - Implementation
  - Results
- Replaying ScalaTrace's Traces in Simulation
- Conclusion

Motivations

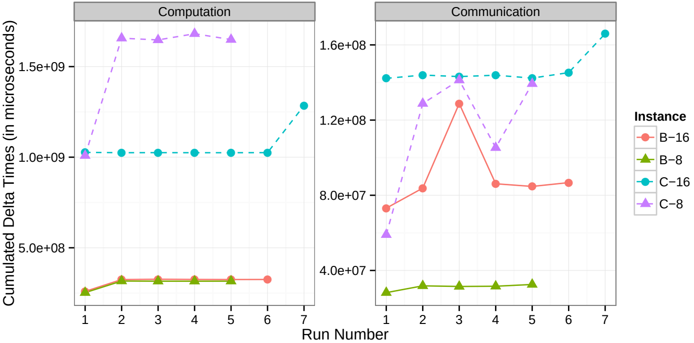
- Claim: Time-related information
  - Limits scalability: No folded or composite acquisitions



- Time is not stretched uniformly across events
- Only the top 5% most time-consuming events follow that trend

Motivations

- Claim: Time-related information
  - Can be impacted by external factors



- Computation time changes with number of cores used
- Communication time might be impacted by nearby jobs

Implementation

ScalaTrace *delta time* logging

- Wrappers on MPI calls all have pre and post stubs
  - Pre: `Stat::RecordStat` ⇒ `StatTime::end` ⇒ `gettimeofday` + delta comp.
    - End of CPU burst
  - Post: `Stat::ResetStat` ⇒ `StatTime::start` ⇒ `gettimeofday`
    - New CPU burst starts

Going Time-Independent

- Create a new `StatInst` class based on the `StatTime` class
  - `StatInst::end` ⇒ `PAPI_accum_counters` + delta computation
  - `StatInst::start` ⇒ `PAPI_accum_counters`
- Initialization of the `PAPI_TOT_INS` counter in `MPI_Init` wrapper
- Destruction in the `MPI_Finalize` wrapper
- Note: `StatInst` class used just for CPU bursts
  - Time is still the metric for communications
  - . . . which is ignored during simulation
- Switch from Timed to Time-Independent with a compilation flag

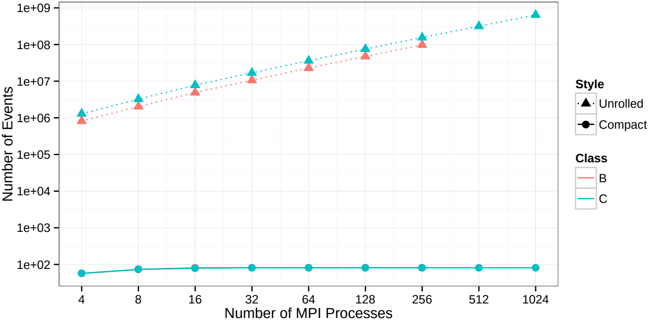
Trace Size

Motivation recall

- ☹ Time-Independent Trace Replay: Verbose trace format ⇒ Limits scalability
- ☺ ScalaTrace: Ultra Compact trace format ⇒ Improves scalability

Compact AND Time-Independent

- ▶ Numbers of events in ScalaTrace-TI and original TI traces (NPB LU)



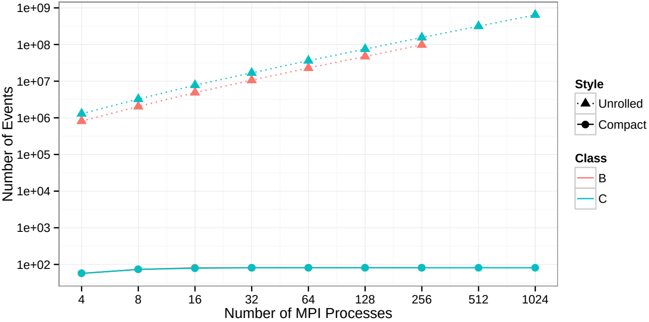
Trace Size

Motivation recall

- ☹ Time-Independent Trace Replay: Verbose trace format ⇒ Limits scalability
- ☺ ScalaTrace: Ultra Compact trace format ⇒ Improves scalability

Compact AND Time-Independent

- Numbers of events in ScalaTrace-TI and original TI traces (NPB LU)
  - ☺ Unrolling ones leads to the others



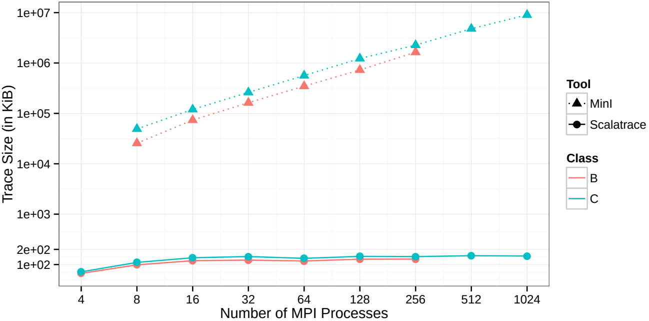
Trace Size

Motivation recall

- ☹ Time-Independent Trace Replay: Verbose trace format ⇒ Limits scalability
- ☺ ScalaTrace: Ultra Compact trace format ⇒ Improves scalability

Compact AND Time-Independent

- Directly impacts the trace sizes
  - From increasing numbers of MB to near-constant numbers of KB



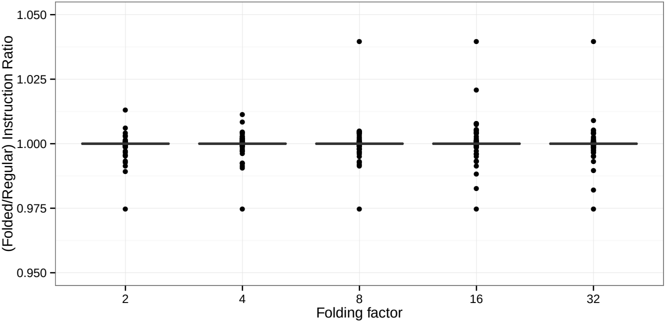
Folded Acquisition

Motivation recall

- ☺ Time-Independent Traces: Folded/Composite modes ⇒ Improves scalability
- ☹ ScalaTrace Timed traces ⇒ Limits scalability

Impact of folding

- On stored numbers of instructions: None!
  - Whatever the number of cores used in a node



Folded Acquisition

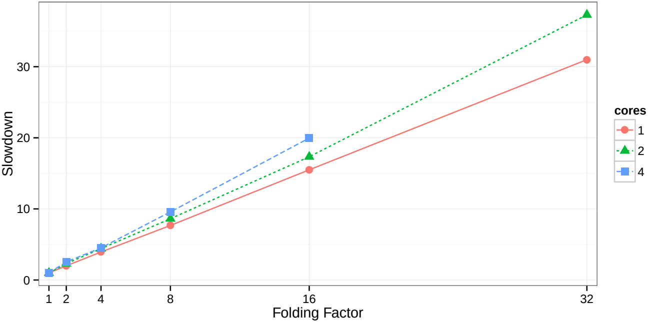
Motivation recall

☺ Time-Independent Traces: Folded/Composite modes ⇒ Improves scalability

☹ ScalaTrace Timed traces ⇒ Limits scalability

Impact of folding

- On acquisition time: larger when all cores are used
  - but same trace as output



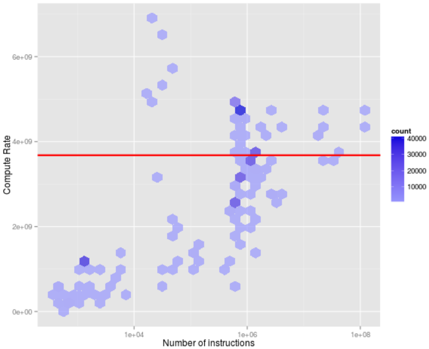
Outline

- Motivation and Background
- Making ScalaTrace Time-Independent
- Replaying ScalaTrace's Traces in Simulation
  - Motivations
  - Implementation
  - Results
- Conclusion

Motivations

Calibration of Time-Independent Replay

- Could be improved
  - ☺ Beyond already good accuracy
  - ☹ A single instruction rate for the whole execution
- Could we leverage ScalaTrace traces' structure?



ScalaTrace Replay

- Is a live replay
  - ☹ Requires a platform at scale
  - ☹ Prevents the exploration of many what-if scenarios

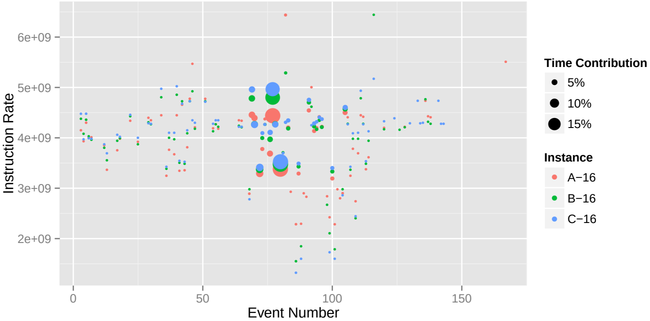
Implementation

- ScalaReplay is an MPI program
  - It can be seamlessly simulated with SMPI without any modification
  - Just have to replace `mpicxx` and `mpirun` by `smpicc` and `smpirun`
  - Thanks to automatic privatization of global variables
- But . . .
  - We don't want to simulate the replay code between MPI calls

    ⇒ Slight modification to SMPI to ignore them
  - Simulate CPU bursts of original application instead
    - Timed: `smpi_execute(delta_time)`
    - Time-Independent: `smpi_execute_flops(inst_number)`
  - ☹ Not so perfect as it seems
    - Troubles with histo replay
    - Had to fall back to normal replay mode

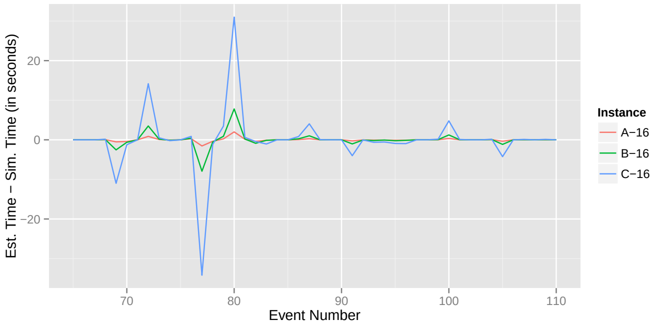
Results

- Still an ongoing work
    - Results are yet to come . . .
- What about instruction rate?
    - Less events (from millions to a hundred)
    - Can compare instances ⇒ possible extrapolation
    - Can identify (and focus on) “big players”



Results

- Still an ongoing work
  - Results are yet to come ...
- What about instruction rate?
  - Single rate doesn't seem a bad idea after all ...
  - Estimated time: use individual rate per event
  - (Hypothetical simulated time: use globally averaged rate



- More investigation is still needed

Conclusions

Time-Independent Trace Replay

- ☺ Decouple acquisition from replay ⇒ Improves scalability
- ☺ Leverages SMPI network models ⇒ Improves accuracy
- ☹ Verbose trace format ⇒ Limits scalability
- ☹ Unique instruction rate for the whole application ⇒ Limits accuracy

ScalaTrace

- ☺ Ultra Compact trace format ⇒ Improves scalability
- ☺ Identifies sub-parts of the applications without extra-instrumentation ⇒ good for calibration
- ☹ No simulated replay ⇒ Limits scalability
- ☹ Timed traces ⇒ Limits acquisition to homogeneous platforms

H. Casanova, F. Desprez, G. Markomanolis, and F. Suter. Simulation of MPI Applications with Time-Independent Traces. In Concurrency and Computation: Practice and Experience, 2014 (In press). http://onlinelibrary.wiley.com/doi/10.1002/cpe.3278/pdf

P. Bédaride, A. Degomme, S. Genaud, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, F. Suter and B. Videau. *Toward Better Simulation of MPI Applications on Ethernet/TCP Networks*. In Proceedings of the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), Denver, CO, Nov 2013. http://hal.inria.fr/hal-00919507

H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. *Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms*. JPDC, 74(10):2899-2917, Oct. 2014. http://www.sciencedirect.com/science/article/pii/S0743731514001105

Conclusions

Time-Independent Trace Replay

- ☺ Decouple acquisition from replay ⇒ Improves scalability
- ☺ Leverages SMPI network models ⇒ Improves accuracy

- ☹ Unique instruction rate for the whole application ⇒ Limits accuracy

ScalaTrace

- ☺ Ultra Compact trace format ⇒ Improves scalability
- ☺ Identifies sub-parts of the applications without extra-instrumentation ⇒ good for calibration
- ▸ Promising step forward toward more scalable off-line simulation
- ▸ But be patient Fox, long is the way!

H. Casanova, F. Desprez, G. Markomanolis, and F. Suter. Simulation of MPI Applications with Time-Independent Traces. In Concurrency and Computation: Practice and Experience, 2014 (In press). http://onlinelibrary.wiley.com/doi/10.1002/cpe.3278/pdf

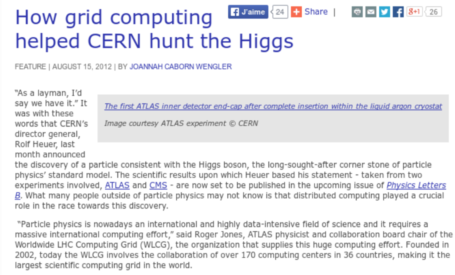
P. Bédaride, A. Degomme, S. Genaud, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, F. Suter and B. Videau. *Toward Better Simulation of MPI Applications on Ethernet/TCP Networks*. In Proceedings of the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), Denver, CO, Nov 2013. http://hal.inria.fr/hal-00919507

H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. *Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms*. JPDC, 74(10):2899-2917, Oct. 2014. http://www.sciencedirect.com/science/article/pii/S0743731514001105

Patrick's Slide

"Nothing good never came out from the Grid"

P. Geoffray



How grid computing helped CERN hunt the Higgs

FEATURE | AUGUST 15, 2012 | BY JOANNAH CABORN WENGLER

The first ATLAS inner detector end-cap after complete insertion within the liquid argon cryostat

Image courtesy ATLAS experiment © CERN

"As a layman, I'd say we have it." It was with these words that CERN's director general, Rolf Heuer, last month announced the discovery of a particle consistent with the Higgs boson, the long-sought-after corner stone of particle physics' standard model. The scientific results upon which Heuer based his statement - taken from two experiments involved, ATLAS and CMS - are now set to be published in the upcoming issue of *Physics Letters B*. What many people outside of particle physics may not know is that distributed computing played a crucial role in the race towards this discovery.

"Particle physics is nowadays an international and highly data-intensive field of science and it requires a massive international computing effort," said Roger Jones, ATLAS physicist and collaboration board chair of the Worldwide LHC Computing Grid (WLCG), the organization that supplies this huge computing effort. Founded in 2002, today the WLCG involves the collaboration of over 170 computing centers in 36 countries, making it the largest scientific computing grid in the world.

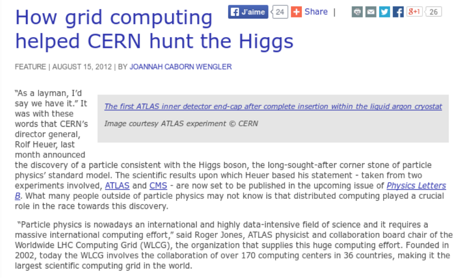
"Three most important elements in today's accomplishment are: LHC experiment, the detectors, and the global Grid."

R. Heuer, CERN's Director General

Patrick's Slide

"Nothing good never came out from the Grid"

P. Geoffray



How grid computing helped CERN hunt the Higgs

FEATURE | AUGUST 15, 2012 | BY JOANNAH CABORN WENGLER

The first ATLAS inner detector end-cap after complete insertion within the liquid argon cryostat

Image courtesy ATLAS experiment © CERN

"As a layman, I'd say we have it." It was with these words that CERN's director general, Rolf Heuer, last month announced the discovery of a particle consistent with the Higgs boson, the long-sought-after corner stone of particle physics' standard model. The scientific results upon which Heuer based his statement - taken from two experiments involved, ATLAS and CMS - are now set to be published in the upcoming issue of *Physics Letters B*. What many people outside of particle physics may not know is that distributed computing played a crucial role in the race towards this discovery.

"Particle physics is nowadays an international and highly data-intensive field of science and it requires a massive international computing effort," said Roger Jones, ATLAS physicist and collaboration board chair of the Worldwide LHC Computing Grid (WLCG), the organization that supplies this huge computing effort. Founded in 2002, today the WLCG involves the collaboration of over 170 computing centers in 36 countries, making it the largest scientific computing grid in the world.

"Three most important elements in today's accomplishment are: LHC experiment, the detectors, and the global Grid."

R. Heuer, CERN's Director General

- Nothing came out . . . but the "particle of God" ;-)