Performance Evaluation and Prediction of Parallel Applications

Georgios Markomanolis

Avalon Team, INRIA, LIP,
Ecole Normale Superiéure de Lyon, France

Ph.D. Defense
January 20th 2014

Under the supervision of Frédéric Desprez and Frédéric Suter

Dimensioning Through Simulation

- User and administrator expertise is not enough
- Decisions can cost a lot of money

⇒ Need for objective indicators by exploring various “what-if” scenarios

- Simulation has many advantages
  - Less simplistic than theoretical models
  - More reproducible than running on production systems
  - Execution on real platform can be time and money consuming
- Focus on non adaptive MPI applications
- Two complementary approaches
  - On-line: execute the application with some simulated parts
  - Off-line: replay an execution trace

Dimensioning Through Simulation

- User and administrator expertise is not enough
- Decisions can cost a lot of money

⇒ Need for objective indicators by exploring various “what-if” scenarios

- Simulation has many advantages
  - Less simplistic than theoretical models
  - More reproducible than running on production systems
  - Execution on real platform can be time and money consuming
- Focus on non adaptive MPI applications
- Two complementary approaches
  - On-line: execute the application with some simulated parts
  - Off-line: replay an execution trace

Time-Independent Trace Replay

- Post-mortem analysis (or off-line simulation) of MPI applications
  - Well covered field
  - Mainly profiling tools
  - Unexpected behaviors and performance bottlenecks detection
  - TAU, Scalasca, Vampir, SCORE-P, ...
- Usually based on timed traces
  - Create a tight link between trace to acquisition environment

Time-Independent Trace Replay

- Post-mortem analysis (or off-line simulation) of MPI applications
  - Well covered field
  - Mainly profiling tools
  - Unexpected behaviors and performance bottlenecks detection
  - TAU, Scalasca, Vampir, SCORE-P, . . .
- Usually based on timed traces
  - Create a tight link between trace to acquisition environment
- Proposition: get rid off the timestamps
  - Trace volumes only
  - Numbers of instructions for computations
  - Message sizes for communications
- Goals
  - Get environment oblivious traces
  - Decouple acquisition from actual replay

Time-Independent Traces

```
for (i=0; i<4; i++){
 if (myId == 0){
  /* Compute 1M instructions */
  MPI_Send(..., (myId+1));
  MPI_Recv(...);
 } else {
  MPI_Recv(...);
  /* Compute 1M instructions */
  MPI_Send(..., (myId+1)% nproc);
 }
}
```

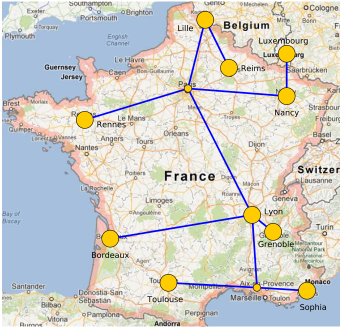
- list of actions performed by each process
- Action described by
  - id of the process
  - type, e.g., computation or communication
  - volume in instructions or bytes
  - some action specific parameters

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

1 init
1 recv 0
1 compute 1e6
1 send 2 1e6
1 finalize

2 init
2 recv 1
2 compute 1e6
2 send 3 1e6
2 finalize

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

Experimental Environments

- NAS Benchmarks:
  - EP: An embarrassingly parallel kernel.
  - DT: Communication with large messages using quad-trees
  - LU: Solve a synthetic system of nonlinear PDEs
  - CG: Conjugate gradient method
  - MG, FT, IS, BT, SP (not tested)

- Grid'5000: 24 clusters, 1,169 nodes, 8,080 cores (July 2013)



Contributions

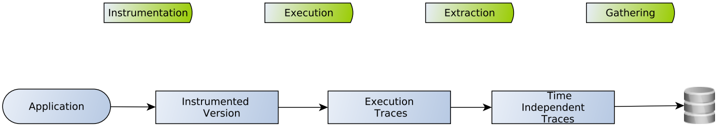
- An original approach that totally decouples the acquisition of the trace from its replay
- Several original scenarios that allow for the acquisition of large execution traces
- Study the state of the art and open source profiling tools
- A new profiling tool based on our framework requirements
- A trace replay tool on top of a fast, scalable and validated simulation kernel
- A complete experimental evaluation of our off-line simulation framework

Outline of the Talk

Trace Acquisition Process

```
for (i=0; i<4; i++){
 if (myId == 0){
 /* Compute 1M
 instructions */
  MPI_Send(..., (myId+1));
  MPI_Recv(...);
 } else {
  MPI_Recv(...);
 /* Compute 1M
 instructions*/
  MPI_Send(...,
       (myId+1)% nproc);
 }
}
```

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

1 init
1 recv 0
1 compute 1e6
1 send 2 1e6
1 finalize

2 init
2 recv 1
2 compute 1e6
2 send 3 1e6
2 finalize

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



Evaluation of Profiling Tools - Results

	Profiling features #criteria 8	Quality of output #criteria 3	Space and Time Overheads #criteria 8	Quality of Software #criteria 11	**Total**
PerfBench	2	0	0	5	**7**
PerfSuite	2	0	0	10	**12**
MpiP	2	0	0	11	**13**
IPM	3	0	0	11	**14**
MPE	4	1	2	10	**17**
PAPI	4	3	6	11	**24**
Extrae	7	2	5	11	**25**
VampirTrace	7	2	5	11	**25**
MinI	7	3	6	10	**26**
TAU	8	2	5	11	**26**
Scalasca	6	2	8	11	**27**
Score-P	7	2	8	11	**28**

Choosing an instrumentation method

Contenders

- TAU-full: Selective instrumentation
- TAU-reduced: Selective instrumentation by instrumenting only MPI calls

```
BEGIN_FILE_EXCLUDE_LIST
*
END_FILE_EXCLUDE_LIST
```

+

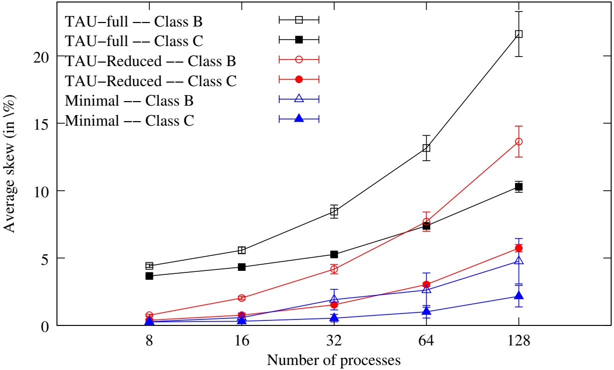
```
-optTauSelectFile=/path/exclude.pdt
```

- MinI: Combination of PMPI library with PAPI support

Metrics

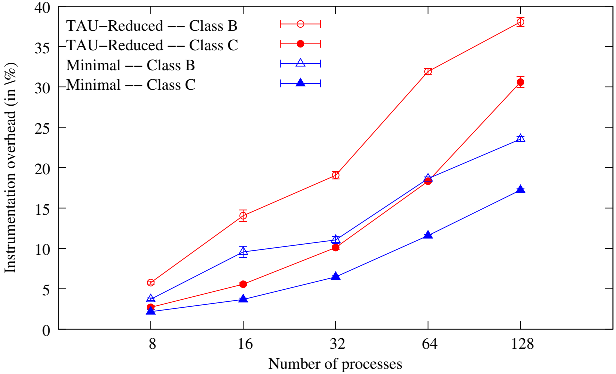
- Skew is the discrepancy in instruction count between a run of the instrumented application and a run of uninstrumented application due to the instrumentation code
- Overhead, the execution time increase due to the execution of the instrumentation code

Instrumentation Skew



- We call “original“ the version with two PAPI calls inserted at the beginning/end of the LU computation
- TAU-full leads to instrumentation skew from 3.66% to 21.62%
- MinI achieves instrumentation skew less than 5%

Instrumentation Overhead

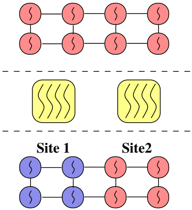


- On average MinI has 1.6 times less instrumentation overhead than TAU-Reduced
- For Mini the instrumentation overhead is up to 23.5%
- MinI produces directly Time-Independent trace files

Execution

Four different acquisition modes

- Regular: one process per CPU
  - Limited scalability
- Folded: more than one process per CPU
  - Acquisition of traces for larger instances
  - Limited by the available memory
- Composite: CPUs don't necessarily belong to one cluster
  - Many nodes available
- Composite and Folded: combination of the previous modes



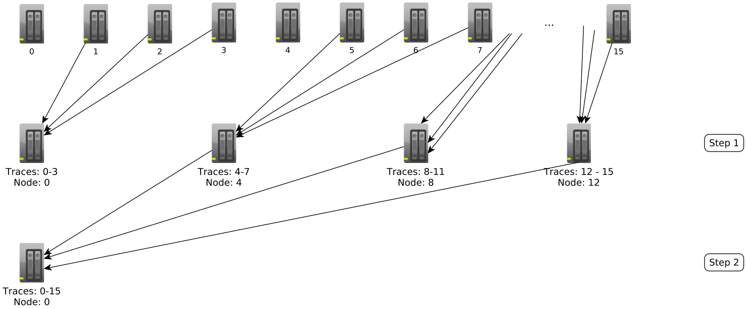
Execution

	Acquisition mode	R	F-2	F-16	C-2	CF-(2,4)
	Number of nodes	64	32	4	(32,32)	(8,8)
LU	Execution Time (in sec.)	11.52	24.45	148.95	23.8	72.14
	Ratio to regular mode	1	2.12	12.92	2.09	6.26

- Linear increase with folded factor
  - 16 processes per CPU ⇒ 13 times bigger execution time
- Increase the number of the sites ⇒ bigger overhead
- A trace tool produces traces with erroneous timestamps
- All the traces are identical with variations less than 1%
  - Acquisition and replay are totally decoupled

Trace Gathering

- The replay tool requires for the traces to be located on the same hard disk
- K-nomial tree reduction
  - $\log_{(K+1)} N$ steps, where N is the total number of files, and K is the arity of the tree



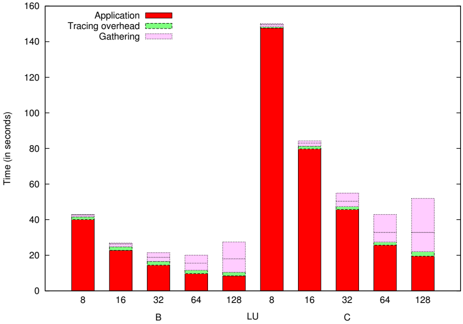
- For benchmark LU, classes B, C and 64 nodes, 2 - 12.58 times faster than Kaget tool.

Analysis of Trace sizes

#Processes	Trace size (in MB) for LU benchmark					
	TAU-full		TAU-reduced		MinI	
	Class B	Class C	Class B	Class C	Class B	Class C
8	334	531	188	298	29.6	48
16	741	1,200	450	714	72	116.8
32	1,600	2,500	973	1,600	159	255
64	3,200	5,100	2,100	3,300	339	550
128	6,600	11,000	4,300	6,800	711	1,200

- TAU_Full $>>$ TAU_Reduced $>>$ MinI
- More information $\rightarrow$ essential information
- Size related to number of actions
  - $\sim$ 15 characters/action, depend on the type of action

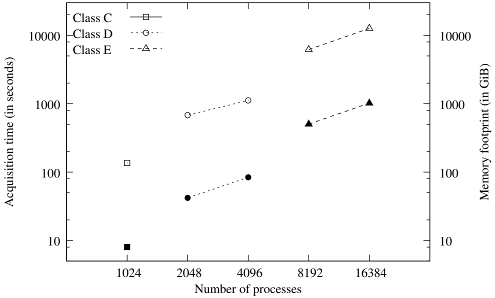
Distribution of the acquisition time



- Time to gather the traces up to 62.02% of the acquisition time
- Horizontal line indicates the gather time of compressed files
- Tracing overhead between 1.75% and 10.55%

Extreme folding

	Time in minutes / Memory footprint in GiB			
Instance	TAU Reduced	Scalasca	Score-P	Minimal Instrumentation
B - 256	2.58 / 11	2.1 / 2.8	1.75 / 4	1.9 / 1.65
C - 1024	N/A	16.3 / 12.9	26.2 / 31	12.9 / 7.95
D - 256	81.8 / 40	55.2 / 16.9	72.16 / 32	47.4 / 15.4



- TAU demands a lot of memory
- Scalasca is efficient but does not provide the exact Time-Independent trace format
- Score-P is getting improved
- MinI tool operates as expected according to our requirements

Large Scale Experiment: LU - E - 16k

Folded

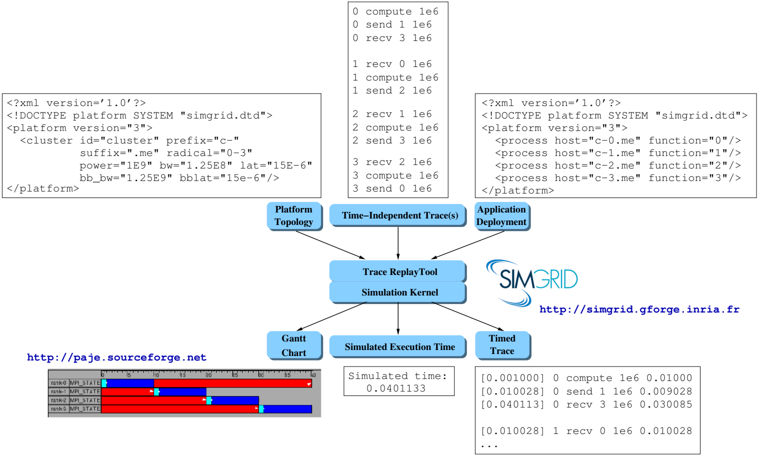
- *StRemi* cluster, 40 nodes, 960 cores, 48GB memory per node
- More than 400 MPI processes per node
- Execution time 3.5 hours, 1 TB of memory

Composite and Folded

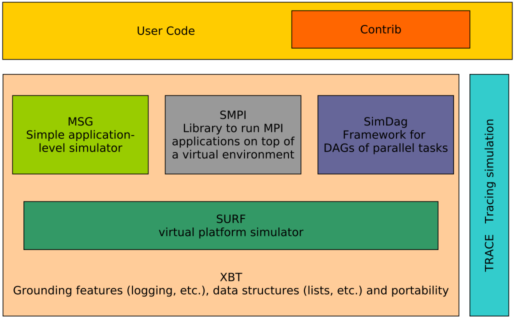
- 778 nodes, 18 clusters, 9 geographically distant sites
- Folded factor based on the memory node
- 1.45 TB Time-Independent traces
- Less than 1.5 hour to execute the instrumented application (53 minutes) and gather the compressed trace files (16 minutes)

Outline of the Talk

Trace Simulated Replay



SimGrid in One Slide



- Three user APIs
  - SimDag: heuristics as DAG of (parallel) tasks
  - MSG: heuristics as Concurrent Sequential Processes
  - SMPI: simulate MPI real applications
- Under the hood
  - SURF: The simulation kernel (full of deeply investigated models)
  - XBT: bundle of useful stuff (data structures, logging, . . .)

Computation Calibration and Cache Usage

- Platform file contains instruction rate of CPUs

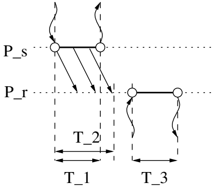
```
<cluster id="AS_cluster" prefix="c-" suffix=".me" radical="0-3"
         power="1E9" bw="1.25E8" lat="15E-6" bb_bw="1.25E9"
bb_lat="15E-6"/>
```

- Calibration procedure
  - Execute a small instrumented instance of the target application
    - Typically Class A on 4 processes
  - Compute the instruction rate for every event
  - Compute a weighted average of the instruction rates for each process
  - Compute the average instruction rate for all the process set
  - Do it five times
- A single instruction rate for everything
  - Small instance ⇒ data fit in L2 cache
  - Larger instance ⇒ exceed L2 capacity ⇒ lower rate!
- We take cache usage into account during calibration

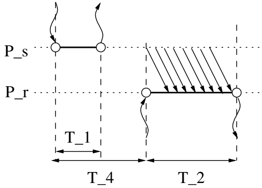
Impact on XML description

```
...
<cluster  id="AS_sgraphene1"
 prefix="graphene-" suffix=".nancy.grid5000.fr" radical="0-38"
 power="3.68E9" bw="1.25E8" lat="15E-6"  bb_bw="1.25E9" bb_lat="15E-6"/>
<cluster  id="AS_sgraphene2"
 prefix="graphene-" suffix=".nancy.grid5000.fr" radical="39-73"
 power="3.68E9" bw="1.25E8" lat="15E-6"  bb_bw="1.25E9" bb_lat="15E-6"/>

<link id="switch-graphene" bandwidth="1.25E9" latency="5E-4"/>

<ASroute src="AS_sgraphene1" dst="AS_sgraphene2"
gw_src="graphene-AS_sgraphene1_router.nancy.grid5000.fr"
gw_dst="graphene-AS_sgraphene3_router.nancy.grid5000.fr">
                <link_ctn id="switch-graphene"/>
</ASroute>
    ...
```

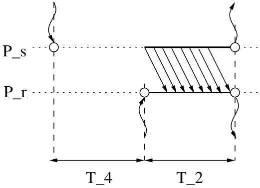
The "hybrid" network model of SMPI



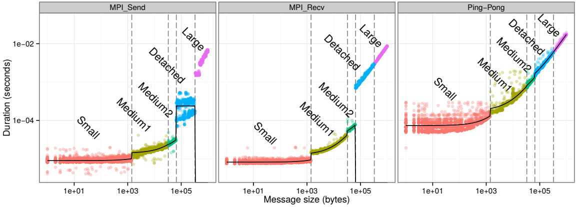
Asynchronous
($k \leq S_a$)



Detached
($S_a < k \leq S_d$)



Synchronous
($k > S_d$)



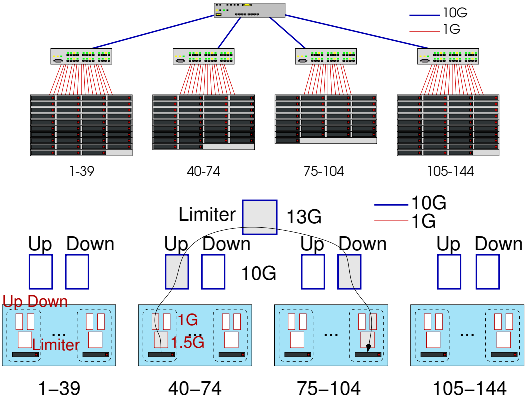
- It is piece wise linear and discon- tinuous

Impact on XML description

```
<config id="General">
  <prop id="workstation/model" value="compound"/>
  <prop id="network/model" value="SMPI"/>

  <prop id="smpi/async_small_thres" value="65536"/>
  <prop id="smpi/send_is_detached_thres" value="327680"/>

  <prop id= "smpi/os"
      value="0:8.93009e-06:7.654382e-10; 1420:1.396843e-05:2.974094e-10;
      32768:1.540828e-05:2.441040e-10; 65536:0.000238:0;327680:0:0"/>

  <prop id= "smpi/or"
       value="0:8.140255e-06:8.3958e-10; 1420:1.2699e-05:9.092182e-10;
       32768:3.095706e-05:6.956453e-10; 65536:0:0; 327680:0:0"/>

    <prop id="smpi/bw_factor"
       value="0:0.400977; 1420:0.913556; 32768:1.078319; 65536:0.956084;
327680:0.929868"/>

    <prop id= "smpi/lat_factor"
       value="0:1.35489; 1420:3.437250; 32768:5.721647;65536:11.988532;
327680:9.650420"/>
  </config>
     ...
```

SMPI - Hybrid Model

- UP/DOWN links to share the available bandwidth separately in each direction
- Limiter link shared by all the flows to and from a processor



Impact on XML description

```
<cluster id="AS_sgraphene1" prefix="graphene-"
    suffix=".nancy.grid5000.fr" radical="0-38" power="3.68E9"
    bw="1.25E8" lat="15E-6"
        sharing_policy="FULLDUPLEX" limiter_link="1.875E8"
       loopback_lat="1.5E-9" loopback_bw="6000000000"></cluster>

 <link id="switch-backbone1" bandwidth="1162500000" latency="1.5E-6"
    sharing_policy="FULLDUPLEX"/>
 <link id="switch-backbone2" bandwidth="1162500000" latency="1.5E-6"
    sharing_policy="FULLDUPLEX"/>

 <link id="explicit-limiter1" bandwidth="1511250000" latency="0"
    sharing_policy="SHARED"/>
 <link id="explicit-limiter2" bandwidth="1511250000" latency="0"
    sharing_policy="SHARED"/>

    <ASroute src="AS_sgraphene1" dst="AS_sgraphene2"
         gw_src="graphene-AS_sgraphene1_router.nancy.grid5000.fr"
         gw_dst="graphene-AS_sgraphene2_router.nancy.grid5000.fr"
     symmetrical="NO"
        <link_ctn id="switch-backbone1" direction="UP"/>
        <link_ctn id="explicit-limiter1"/> <link_ctnid="explicit-limiter2"/>
        <link_ctn id="switch-backbone2" direction="DOWN"/>
```

Trace Replay Tool

- Based on SMPI
  - Complete MPI implementation
  - Better handling of eager-mode
  - Factoring the efforts
- Implementation of the *send* action

```
static void action_send (const char *const *action){
 int to = atoi(action[2]);
 double size = parse_double(action[3]);
 smpi_mpi_send (NULL, size, MPI_BYTE, to, 0, MPI_COMM_WORLD);}
```

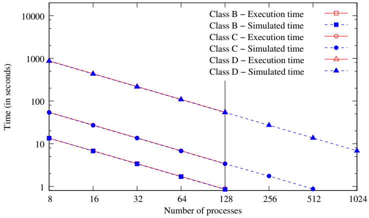
- A user has to execute the *smpi_replay* tool

```
int main(int argc, char *argv[]){
 smpi_replay_init(&argc, &argv);
 smpi_action_trace_run();
 smpi_replay_finalize();
 return 0;}
```

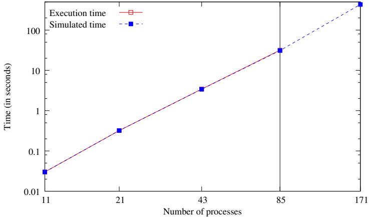
- *smpirun* is used to execute the simulator

```
smpirun -np 8 -hostfile hostfile -platform platform.xml \
        ./smpi_replay trace_description
```

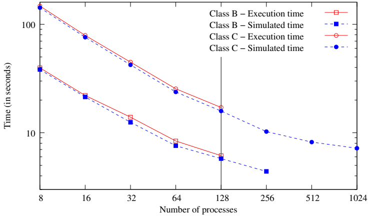
Simulation Accuracy - Latest Framework



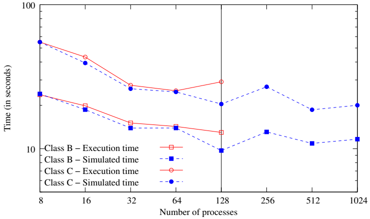
EP



DT



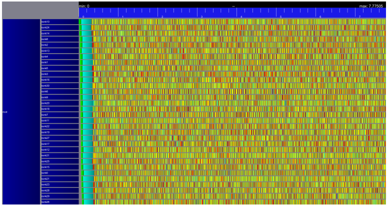
LU

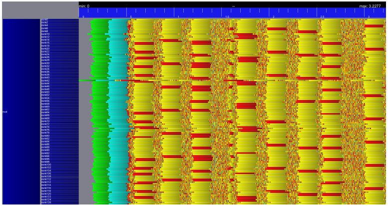


CG

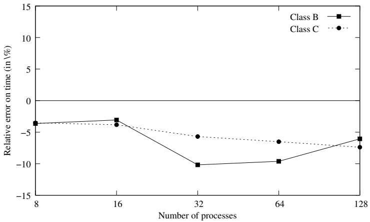
Source of inaccuracy for CG

- Two seconds Gantt-chart of the real execution of a class B instance of CG for 32 processes on left side and 128 processes on right side
- Massive switch packet drops lead to 0.2s timeouts in TCP for 128 processes

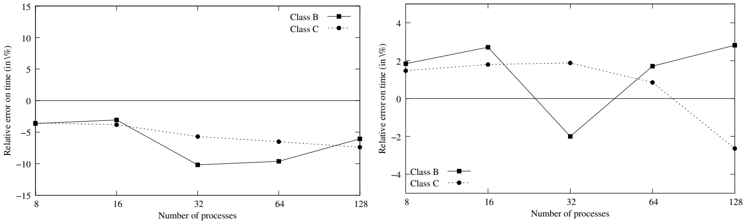




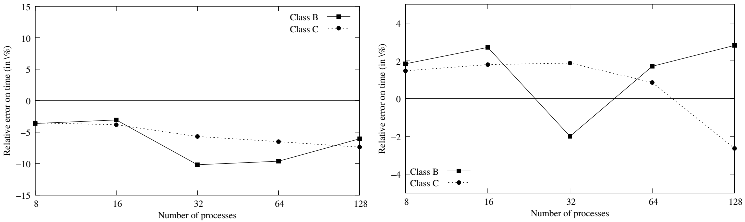
Source of inaccuracy for LU

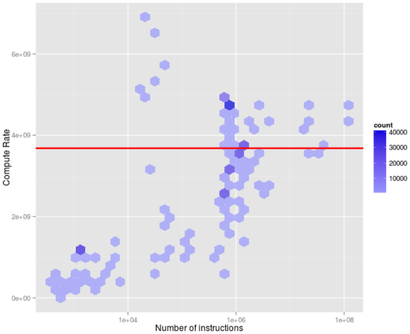


Source of inaccuracy for LU

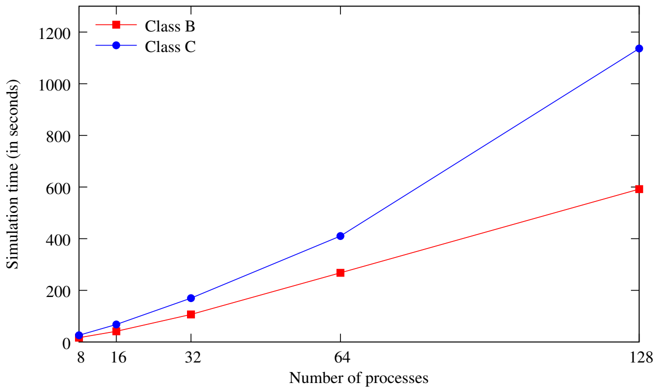


Source of inaccuracy for LU





Simulation Time



- 8.6 seconds for one million actions up to 64 processes
- 13 seconds for one million actions on 128 processes
- Almost 8 days would be needed to replay 1.45 TB of LU-E with 16k processes

Outline of the Talk

Conclusions

- The Time-Independent Trace Replay Framework was presented by detailing and evaluation its two main parts: acquisition and replay
- The acquisition procedure is decoupled from its replay, thus the acquired Time-Independent traces can be simulated with various scenarios
  - We can use the same Time-Independent traces with future SimGrid network models e.g., Infiniband
- We implemented a profiling tool respecting our requiremets which is more efficient than other available ones

Perspectives

Short term

- More MPI calls can be supported
- Framework for automatic acquisition of the traces

Long term

- Simulation of real applications
- Simulation of larger platforms
- Decrease simulation time and trace sizes
- Investigate our framework model with regard to multicore processors

Difficult

- A new computation model could be introduced

Publications

- Simulation of MPI Applications with Time-Independent Traces, Concurrency and Computation: Practice and Experience, under revision
- Toward Better Simulation of MPI Applications on Ethernet/TCP Networks, Proceedings of the 4th International Workshop on Performance Modeling, Benchmarking and Simulation of High PerformanceComputer Systems (PMBS), 2013
- Improving the Accuracy and Efficiency of Time-Independent Trace Replay, Proceedings of the 3rd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), 2012
- Assessing the Performance of MPI Applications Through Time-Independent Trace Replay, Proceedings of the 2nd International Workshop on Parallel Software Tools and Tool Infrastructures (PSTI), 2011
- Evaluation of Profiling Tools for the Acquisition of Time Independent Traces, Technical Report, RT-0437, INRIA
- Time-Independent Trace Acquisition Framework – A Grid'5000 How-to, Technical Report, RT-0407, INRIA