Assessing the Performance of MPI Applications Through Time-Independent Trace Replay

F. Desprez[1] **G. Markomanolis**[1] M. Quinson[2] F. Suter[3]

[1]INRIA, LIP, ENS de Lyon,
Lyon, France

[2]Nancy University, LORIA, INRIA,
Nancy, France

[3]Computing Center, CNRS, IN2P3,
Lyon-Villeurbanne, France

PSTI 2011

Outline

Introduction and motivation

Introduction

- Dimensioning of compute clusters
- Simulation Frameworks:
  - off-line simulation
    - replay an execution trace
  - on-line simulation
    - a part of the application is simulated
- The current framework follows the off-line approach

Motivation

- Off-line simulation is usually based on timed traces
  - Dependency on the machine
- New approach
  - Do not use timestamps
  - Decouple the acquisition of the trace from its replay
- Applies on regular data-independent MPI applications

Time-Independent Trace Format

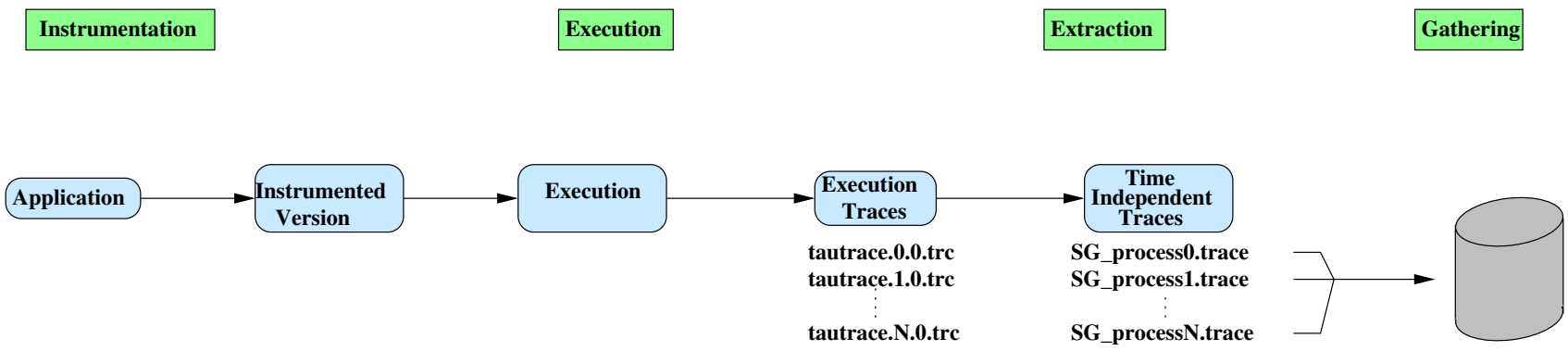
```
for (i=0; i<4; i++){
 if (myId == 0){
  /* Compute 1M instructions */
  MPI_Send(1MB,..., (myId+1));
  MPI_Recv(...);
 } else {
  MPI_Recv(...);
  /* Compute 1M instructions */
  MPI_Send(1MB,..., (myId+1)% nproc);
 }
}
```

```
p0 compute 1e6
p0 send p1 1e6
p0 recv p3

p1 recv p0
p1 compute 1e6
p1 send p2 1e6

p2 recv p1
p2 compute 1e6
p2 send p3 1e6

p3 recv p2
p3 compute 1e6
p3 send p0 1e6
```

Each action is constituted by:

- id of the process
- type, e.g., computation or communication
- volume in flops or bytes
- action specific parameters

Trace Acquisition Process



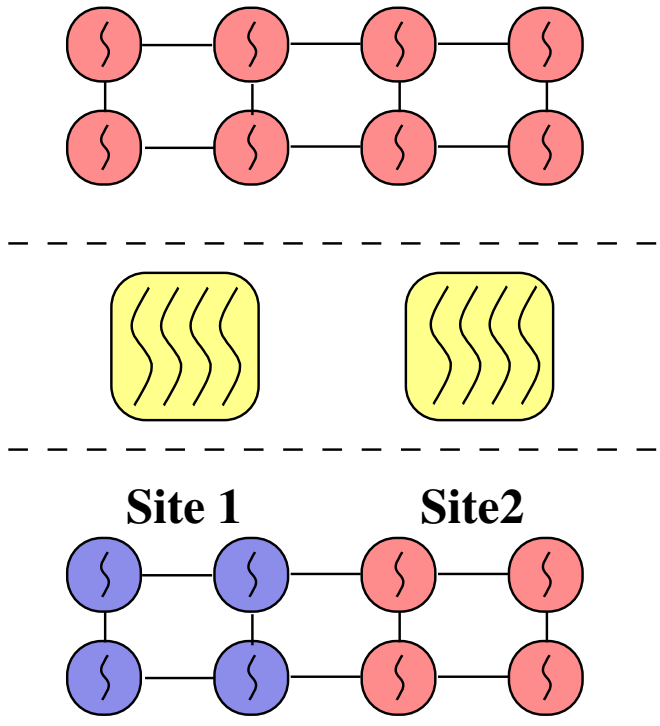
Instrumentation

- Need for a suitable tracing tool
- TAU Performance System is a profiling and tracing tookit:
  - Record every MPI message
  - Measuring the instructions through PAPI
  - Selective instrumentation

```
1       call TAU_ENABLE_INSTRUMENTATION()
2       call ssor(itmax)
3       call TAU_DISABLE_INSTRUMENTATION()
```

Acquisition modes

- **Regular mode**: one process per CPU
  - Limited scalability
- **Folding mode:** more than one process per CPU
  - Acquisition of traces for larger instances
  - Limited by the available memory
- **Scattering mode:** CPUs do not necessarily belong to the same cluster
  - Many nodes available
- **Scattering and Folding:** the combination of Folding and Scattering mode



- The trace remains the same for all the modes
- Acquisition and replay are totally decoupled

Post-processing of the Execution Traces I

After the execution of an instrumented application, there are:

- trace files (1 per process), binary files
- event files (1 per process), text files

Example of event file:

```
49 MPI 0 "MPI_Send()  " EntryExit
1 TAUEVENT 1 "PAPI_TOT_INS" TriggerValue
```

Need to:

- *Extract* a time-independent trace from the trace and event files
- *Gather* the extracted traces on a single node

Post-processing of the Execution Traces II

- TAU provides the Trace Format Reader library for extracting events
  - 11 callback functions to implement

```
1 0 1.42947e+06 EnterState      49
1 0 1.42947e+06 EventTrigger     1  164035532
1 0 1.4295e+06  EventTrigger    46  163840
1 0 1.4295e+06  SendMessage   0 0  163840  1 0
1 0 1.4299e+06  EventTrigger     1  164035624
1 0 1.4299e+06  LeaveState      49
```

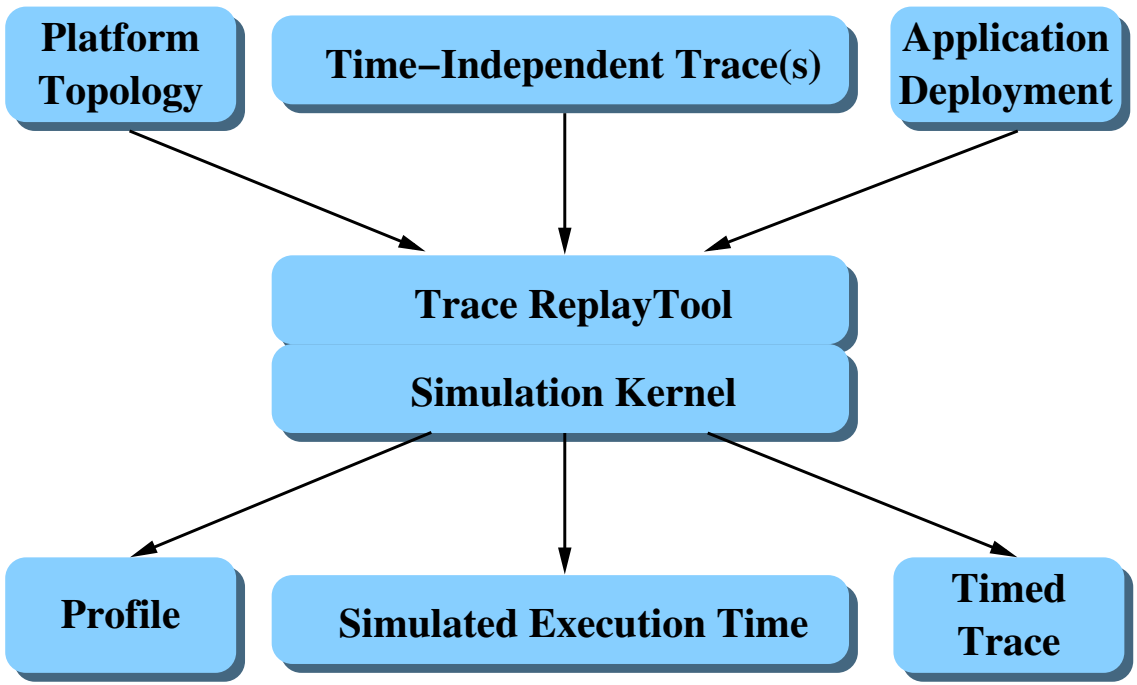
Time independent trace:

```
p1 send p0 163840
```

- Use a K-nomial tree to gather traces on a single node

Trace Replay with SimGrid

Inputs and outputs of the SIMGrid trace replay framework



Platform and Deployment files

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "simgrid.dtd">
<platform version="3">
 <AS id="AS_mysite" routing="Full">
  <cluster id="AS_mycluster"
           prefix="mycluster-" suffix=".mysite.fr"
           radical="0-3" power="1.17E9"
           bw="1.25E8" lat="16.67E-6"
           bb_bw="1.25E9" bb_lat="16.67E-6"/>
 </AS>
</platform>
```

```
<!DOCTYPE platform SYSTEM "simgrid.dtd">
<platform version="3">
  <process host="mycluster-0.mysite.fr"
           function="p0"/>
  <process host="mycluster-1.mysite.fr"
           function="p1"/>
  <process host="mycluster-2.mysite.fr"
           function="p2"/>
  <process host="mycluster-3.mysite.fr"
           function="p3"/>
</platform>
```

Trace replay tool

Using MSG API:

1. A function for every action, for instance:

```
1  static void compute(xbt_dynar_t action){
2    char *amount = xbt_dynar_get_as(action,
3                        2, char *);
4    m_task_t task = MSG_task_create(NULL,
5                        parse_double(amount),
6                        0, NULL);
7    MSG_task_execute(task);
8    MSG_task_destroy(task);
9  }
```

2. Register the function:

```
MSG_action_register("compute", compute);
```

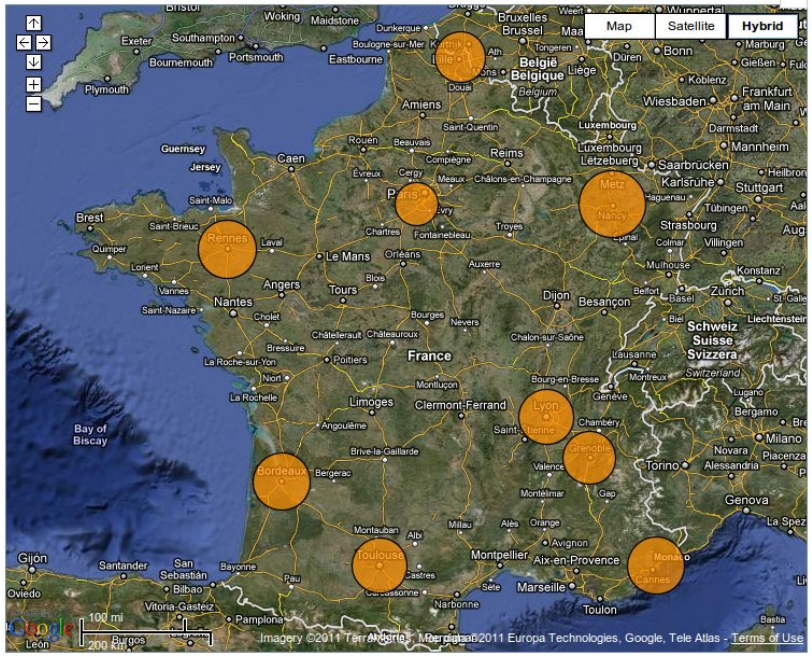
3. Call the function `MSG_action_trace_run`

```
<process host="mycluster-0.mysite.fr" function="p0">
<argument value="SG_process0.trace"/>
</process>
```

Calibration

- Computation
  - Execute a small instrumented instance of the target application
  - Compute the instruction rate for every event
  - Compute a weighted average of the instruction rates for each process
  - Compute the average instructions rate for all the process set
  - Compute an average over these five runs
- Communication
  - Bandwidth: We use the nominal value of the links
  - SkaMPI for measuring the latency
  - Piece-wise linear model used by SIMGrid dedicated to MPI communications:
    - Latency and bandwidth correction factors

Grid'5000 and software



Total: 10 sites, 1532 nodes, 8440 cores (September 2011, with Reims site)

One of the key concept of the Grid'5000 experimental platform is to offer its users the capacity to deploy their own system image at will

Debian Lenny image: Kernel (v2.6.25.9), Perfctr driver (v2.6.38), TAU (v2.18.3), PDT (v3.14.1), PAPI (v3.7.0), NAS Parallel Benchmarks (v3.3), OpenMPI (v1.3.3), SimGrid (v3.6-r9800)

Benchmarks, Clusters

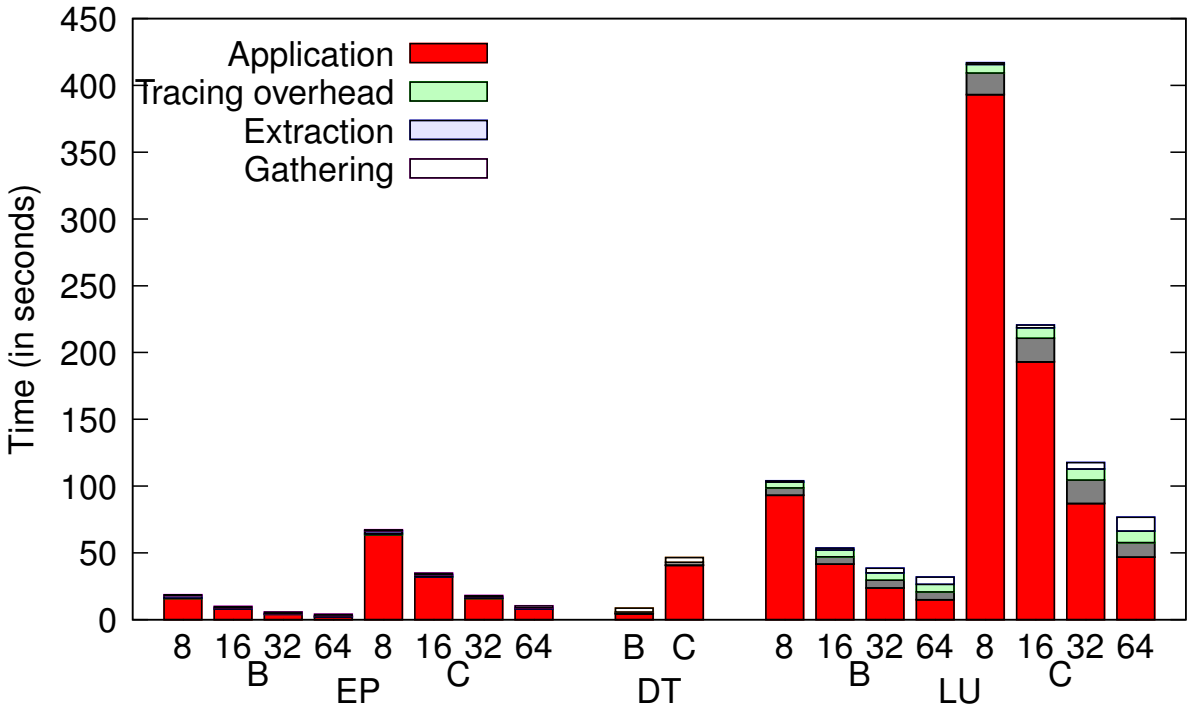
NAS Parallel Benchmarks (NPB):

- Embarrassing Parallel (EP), Data Traffic (DT), LU factorization (LU) programs
- 7 different *classes*, denoting different problem sizes: S (the smallest), W, A, B, C, D, and E (the largest)

Clusters:

- *Bordereau*: 93 2.6GHz Dual-Proc, Dual-Core AMD Opteron 2218 nodes, 4GiB RAM, single 10 Gigabit switch
- *Gdx*: 86 2.0 GHz Dual-Proc AMD Opteron 246 scattered across 18 cabinets, 2GiB RAM

Acquisition time



- The worst value for the percentage of extraction and gathering steps is 49.31% for EP, class B with 64 processes
- Many “what-if” scenarios can be applied with one acquisition

Evolution of the execution time

Evolution of the execution time of instrumented class B instance of EP, LU executed by 64 processes and 43 ones for DT with regard to the acquisition mode

	Acquisition mode	R	F-2	F-4	F-8	F-16	F-32	S-2	SF-(2,2)	SF-(2,4)	SF-(2,8)	SF-(2,16)
	Number of nodes	64	32	16	8	4	2	(32,32)	(16,16)	(8,8)	(4,4)	(2,2)
LU	Execution Time (in sec.)	20.73	52.96	88.66	179.07	347.27	689.18	37.54	79.19	134.05	277.25	505.64
LU	Ratio to regular mode	1	2.55	4.28	8.64	16.75	33.25	1.81	3.82	6.47	13.37	24.39
EP	Execution Time (in sec.)	1.99	4.06	8.33	16.29	36.42	69.05	2.67	5.38	10.77	21.52	43.09
EP	Ratio to regular mode	1	2.04	4.18	8.18	18.3	34.7	1.34	2.7	5.41	10.81	21.65

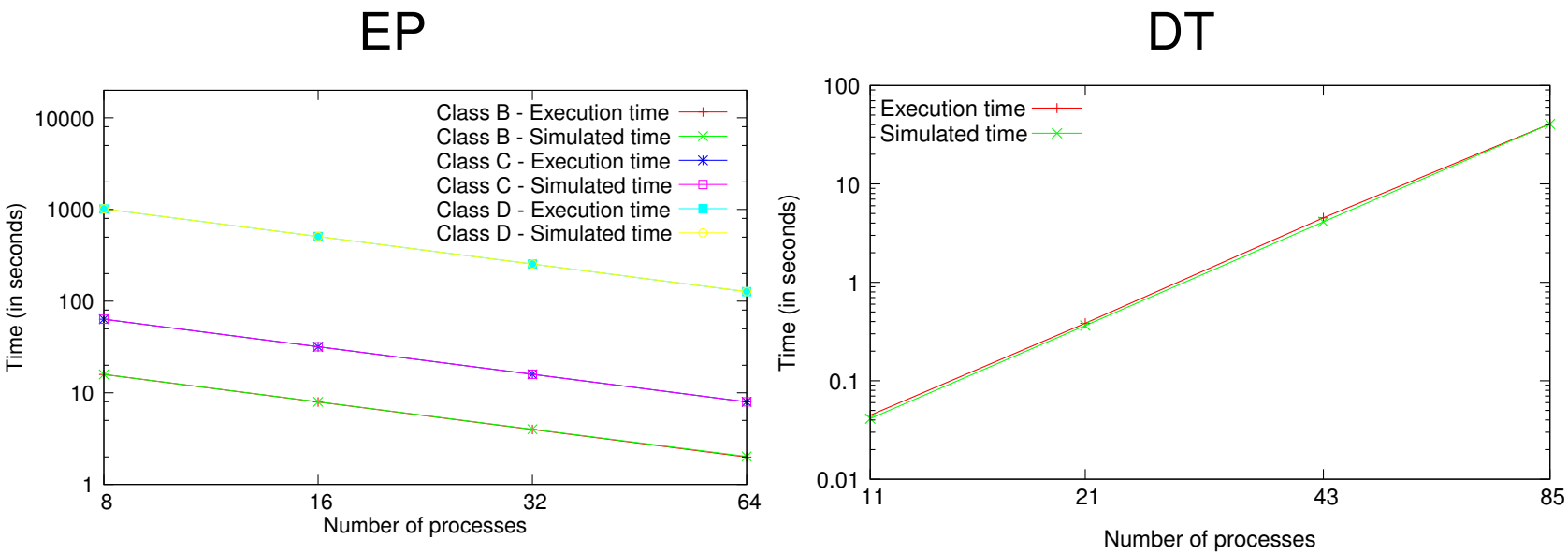
	Acquisition mode	R	F-2.625	F-5.25	F-10.5	F-21	S-2	SF-(2,2.625)	SF-(2,5.25)	SF-(2,10.5)	SF-(2,21)
	Number of nodes	43	16	8	4	2	(22,21)	(8,8)	(4,4)	(2,2)	(1,1)
DT	Execution Time (in sec.)	4.56	12.62	18.67	32.2	58.8	10.66	31.277	41.06	47.49	66.93
DT	Ratio to regular mode	1	2.76	4.09	7.06	12.89	2.33	6.86	9	10.4	14.67

- Linear increase with folding factor for EP and LU
- A trace tool produces traces with erroneous timestamps
- Same simulated time for all the acquisition modes with variations less than 1%

Analysis of Trace Sizes (LU benchmark)

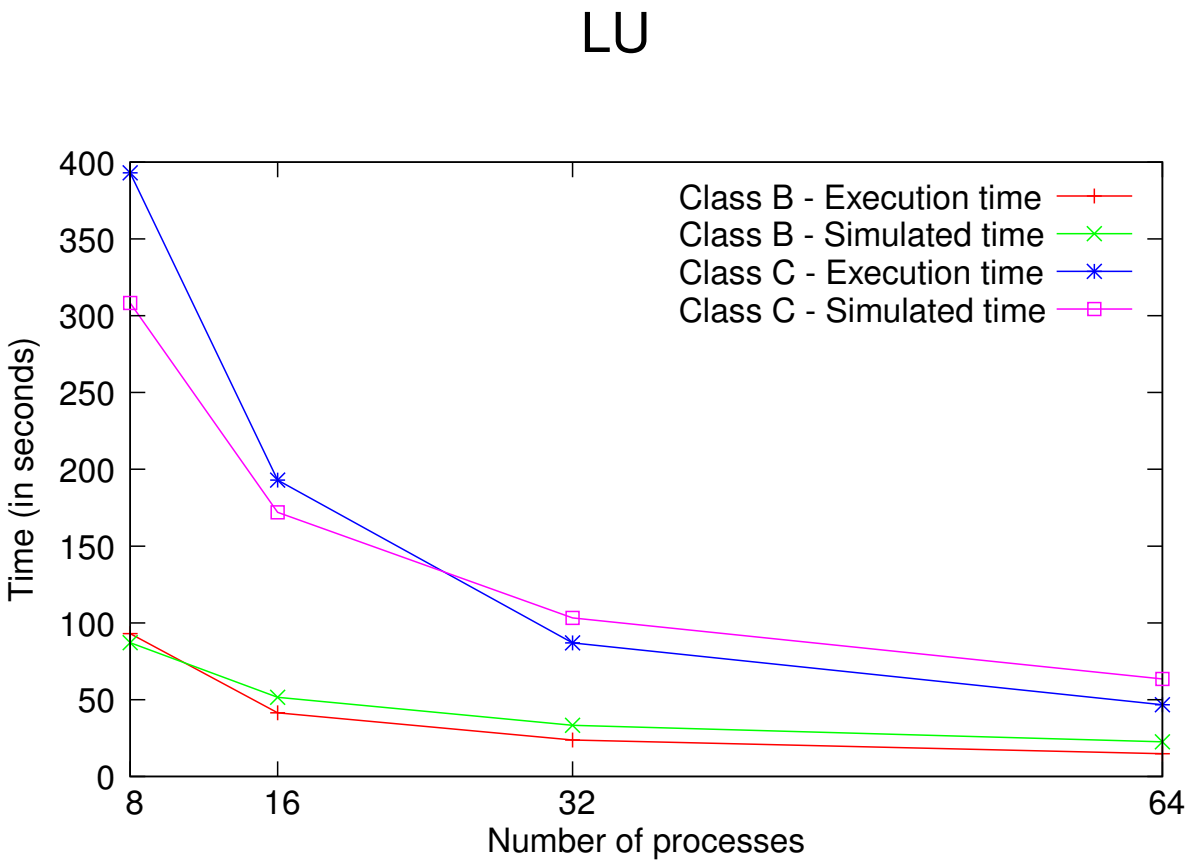
#Processes	Trace size (in MiB)		#Actions (in millions)	
	Class B	Class C	Class B	Class C
8	29.9	48.4	2.03	3.23
16	72.6	117	4.87	7.75
32	161.3	256.8	10.55	16.79
64	344.9	552.5	22.73	36.17

Accuracy of Time-Independent Trace Replay I



- The relative errors are low enough (worst case 1.7% for EP and 7.27% for DT)

Accuracy of Time-Independent Trace Replay II

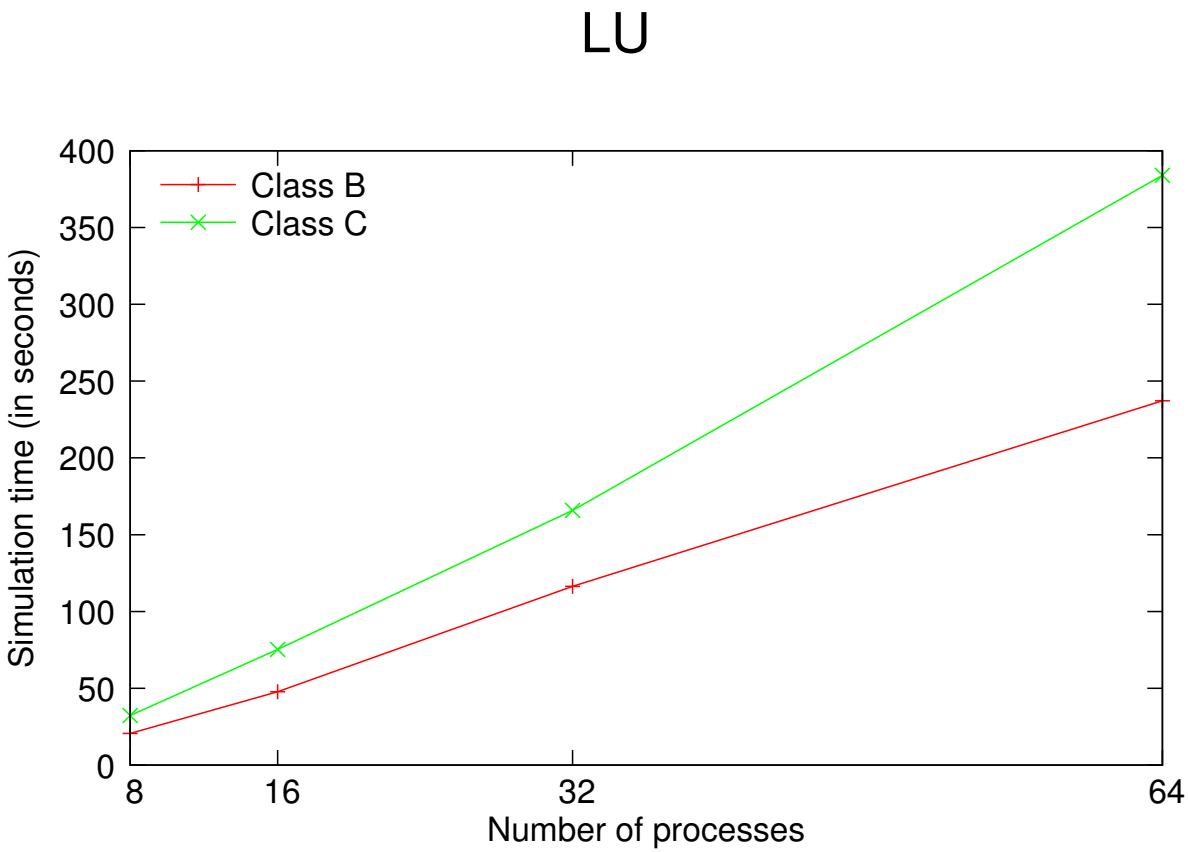


- The local relative error may be quite high and not constant
- Trends are correctly predicted
- Calibration issues

Acquiring a Large Trace

- Executing the acquisition process for a Class D instance on 1,024 processes
- We only use 32 nodes, 128 individual cores, and a folding factor of 8
- EP benchmark: less than 86 seconds to acquire the 4.1 MiB time-independent trace
- LU benchmark: less than 25 minutes to acquire the 32.5 GiB time-independent trace
- Memory constraints prevent the folding of the DT benchmark for the current nodes

Simulation Time



- The speed of simulating actions is around to 94205 actions/second
- Linear with the number of processes
- The simulation time for EP and DT benchmarks is less than 0.1s

Conclusion and Future Work

Conclusion

- Time-independent traces for the off-line simulation of MPI applications
- Tools for our Framework:
  - TAU is a well known profiling tool since 1997
  - SIMGrid
- Some issues have to be solved

Future Work

- Solve the accuracy and simulation time issues
- We also aim at exploring techniques to reduce the size of the traces
- Compare off-line simulations results with those produced by on-line simulators

Thank you!
Questions?