www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MUSA Tutorial
Session 1: Advanced concepts and enviroment introduction

Francesc Martínez
RoMoL at BSC-CNS

Barcelona, 11th May 2018

Tutorial objectives

Session 1

- MUSA Introduction
- Run a high level simulation (Burst mode)

Session 2

- Understand MUSA configurability
- Run a more detailed simulation (Memory mode)
- Compare results against a native execution

Session 3

- Perform a design space exploration simulation

Session 1 Steps

- Step 1: MareNostrum4 and MUSA Introduction
- Step 2: Environment Setup
- Step 3: Execute the benchmark natively
- Step 4: Generate a Burst mode trace
- Step 5: Simulate the Burst mode trace
- Step 6: Integrate the simulation results
- Result Analysis

Step 1: MareNostrum & MUSA Introduction

Login on MareNostrum:

```
$> ssh -X user@mn1.bsc.es
```
(can be mn1, mn2 or mn3)

- username: nct010[27-28], nct010[46-68]
- password: ALUM.ARMAPR18.0XX
- Working folder: /gpfs/projects/nct01/nct010XX

SLURM system:

```
$> sbatch script_name.bash
```
To submit scripts to the queue

```
$> squeue
```
To query the queue system

Step 1: MareNostrum & MUSA Introduction

Loading MUSA:

```
$> module use -a /apps/ROMOL/musa/2.8/module/
$> module purge && module load musa
$> module list
```

This loads:

- Simulators: Tasksim(+ramulator), Dimemas.
- Compiler: Mercurium, GCC 7.1.0
- OpenMP/OmpSs runtime: Nanos++
- Libraries: OpenMPI, Extrae
- Other: DynamoRIO

Step 1: MareNostrum & MUSA Introduction

- Run SP-MZ natively on MareNostrum4
- Generate a Burst-mode trace
- Run a Burst-mode simulation
- Analyze the results
- Generate an Scalability graph (Native vs. Burst simulation)

Step 2: Enviroment Setup

```
$> mkdir musa_tutorial && cd musa_tutorial
$> musa_start.bash
```

- Copies initial scripts:
  - job_tracer.bash — Generates Burst-mode trace
  - job_tracer_memory.bash — Generates Memory-mode trace
  - generate_speedup_graph.bash — Generates Speed-up graph
  - native_run.bash — Execute the benchmark and time it
  - tutorial_instructions/ — Folder with step by step guide

Step 2: Enviroment Setup

- If you get lost:

```
$> musa_fallback_recovery.bash $session $step
```

Step 2: Enviroment Setup

```
$> musa_load_tracing_example.bash
```

- Copies and links SP:
  - Input class B, 4 MPI ranks
  - MPI+OpenMP
  - Compiled with Mercurium
  - Uses Nanos++ as OpenMP runtime
  - Contains an instrumented version (sp-mz.B.4_instr)

Step 3: Native run

- Execute SP-MZ.B.4 natively:

```
$> sbatch native_run.bash
```

- Runs with 1, 2, 4, 8 and 16 threads per rank
- Check execution finish in the SLURM queue:

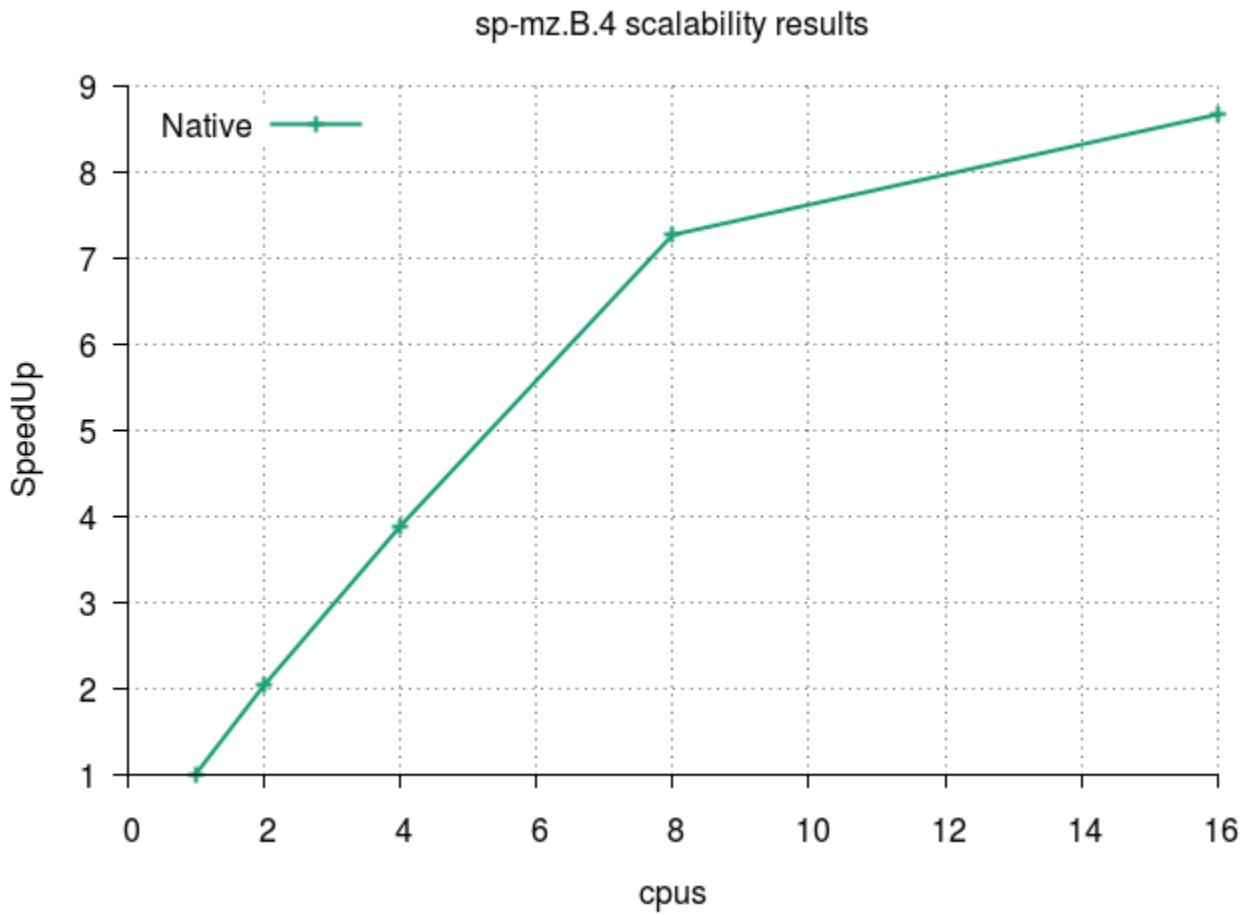
```
$> squeue -l
```

- Results on native_logs/
- Generate speedup graph:

```
$> ./generate_speedup_graph.bash
```

Step 3: Native run

- Initial speedup result:



Step 4: Burst trace generation

- To modify the configuration:

```
$> vim job_tracer.bash
```

- Contains:
  - APPNAME, ARGS, RANKS
  - MEMORY_MODE
  - Memory mode options
  - TSMPI_SIM_CONFIG
  - TSMPI_SIM_CORES

Step 4: Burst trace generation

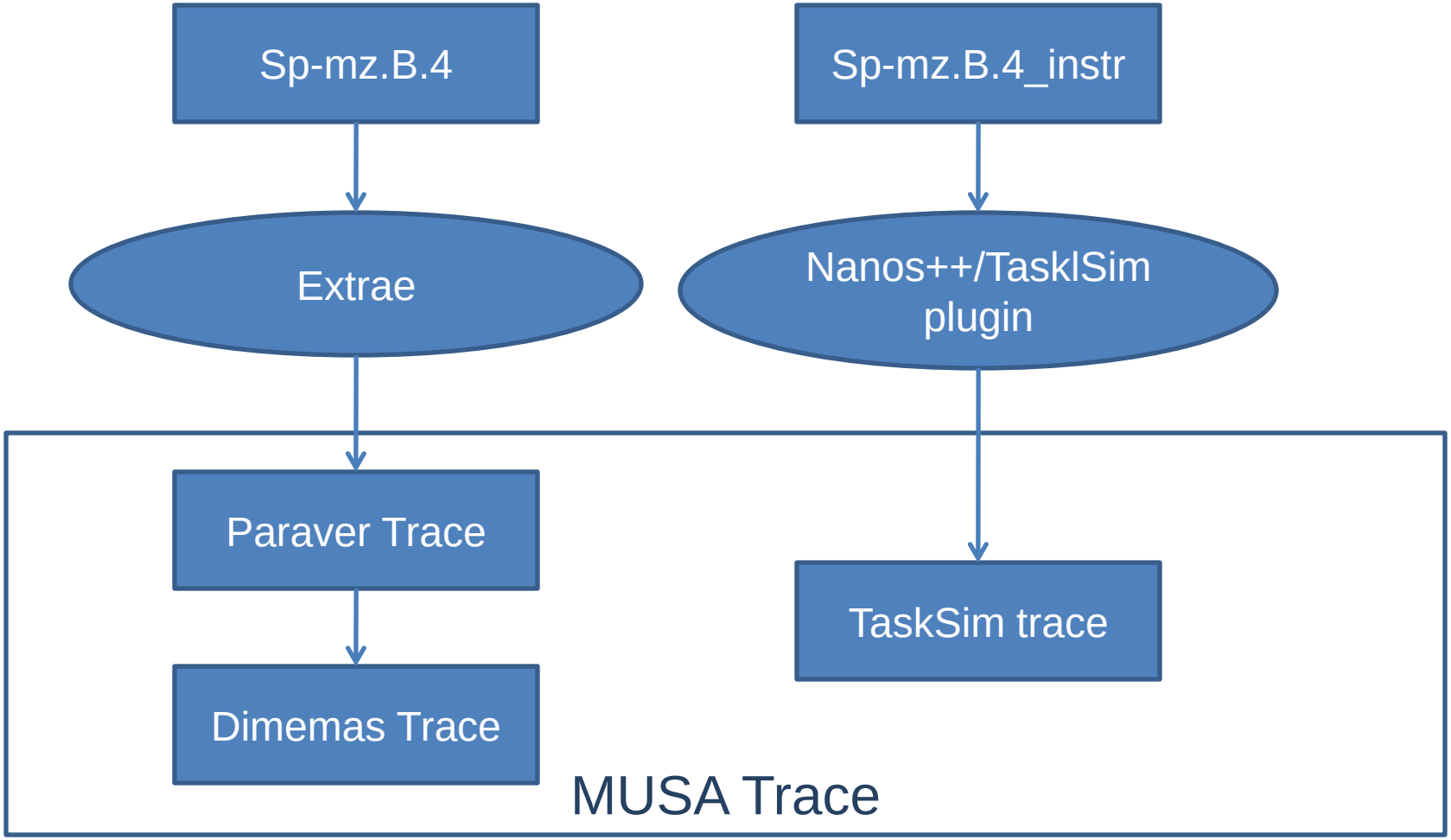
- To generate the Burst mode trace:

```
$> sbatch job_tracer.bash
```

- A Burst-mode trace contains:
  - User code and runtime phase durations.
  - Events for task creation, dependencies, waits, …
  - Separated by MPI Events

Step 4: Burst trace generation

- Both binaries are used!



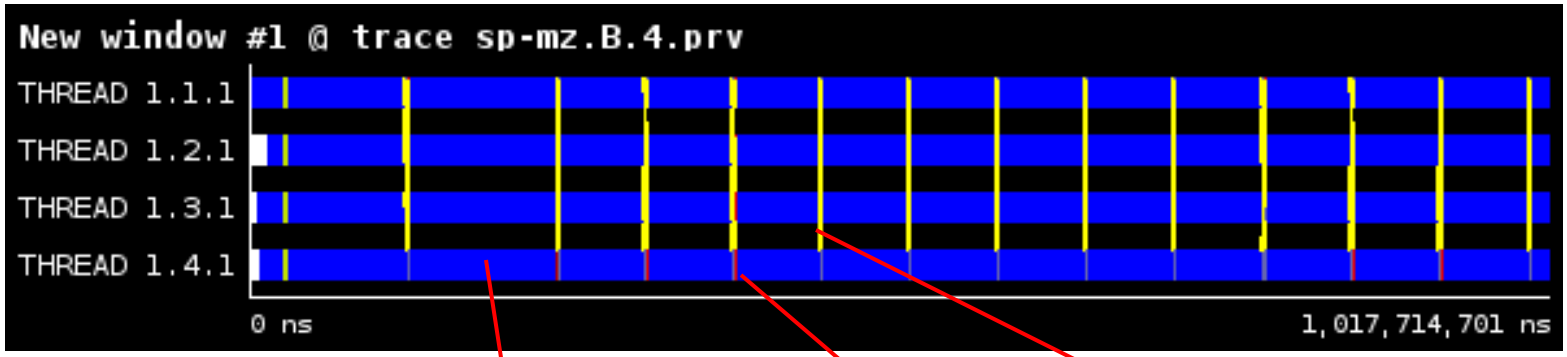
Step 4: Burst trace generation

- What the script has generated:
  - logs_musa_generation-${jobid}.[out|err]
  - TRACE_sp-mz.B.4_000004_BRST/
    - LOGS/
    - TRACE/
      - trace_prv/
      - trace_ts/
    - SIMULATION/
      - A1_PRESIM/
      - A2_INTEGRATION_PRESIM/

Real execution Paraver trace

- Open the Paraver trace:

```
$> module load paraver
$> cd TRACE_sp-mz.B.4_000004_BRST/TRACE/trace_prv
$> wxparaver trace_sp-mz.B.4.prv
```

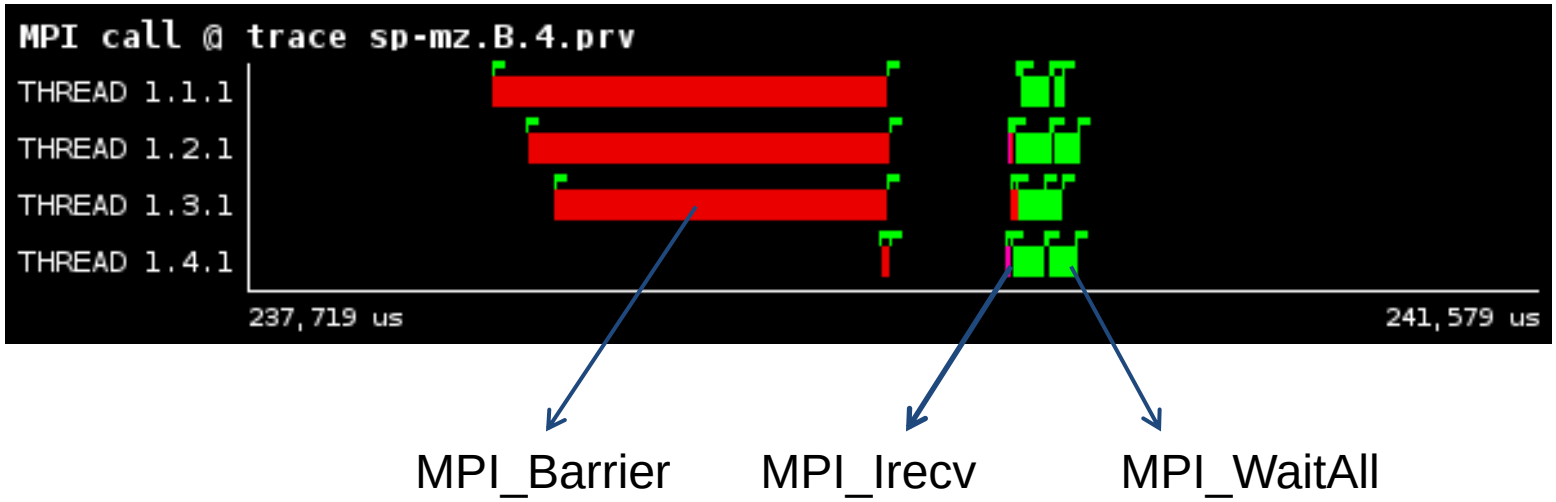


Computation phase MPI phase Communication

Real execution Paraver trace

- Load MPI Call view.

  (Menu->File->Load Configuration select MPI_call.cfg)

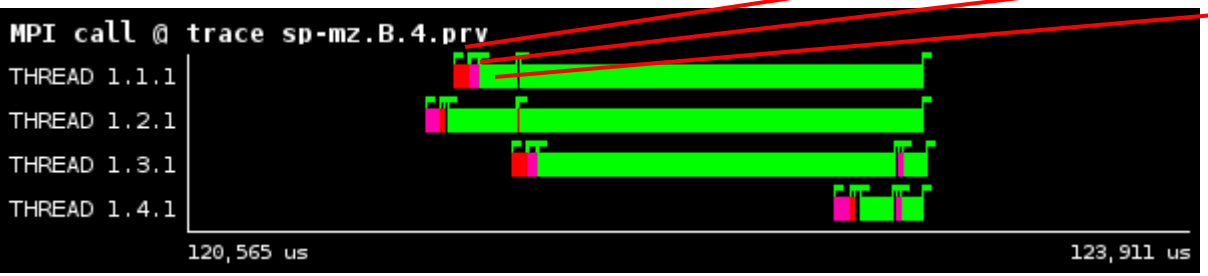


Real execution Paraver trace

Correlates with information on .mpiphases

```
$> cd TRACE_sp-mz.B.4_000004_BRST/TRACE/trace_ts
$> vim sp-mz.B.4_proc_000001.ts.mpiphases
```

```
0:1:0:0:43:INIT
1:18005000:0:0:1:MPI_Ini
2:18005001:0:0:8:MPI_Bca
3:18005002:0:0:8:MPI_Bca
4:18005003:0:0:8:MPI_Bca
5:18005004:0:0:8:MPI_Bca
6:18005005:0:0:8:MPI_Bca
7:18005006:0:0:8:MPI_Bca
8:18005007:0:0:8:MPI_Bca
9:18005008:0:0:8:MPI_Bca
10:18005009:1:0:8:MPI_Bc
11:18005010:0:0:5:MPI_Is
12:18005011:0:0:6:MPI_Ir
13:18005012:0:0:11:MPI_W
14:18005013:0:0:5:MPI_Is
15:18005014:0:0:6:MPI_Ir
```



Step 5: Burst mode simulation

- Go to the simulation folder:

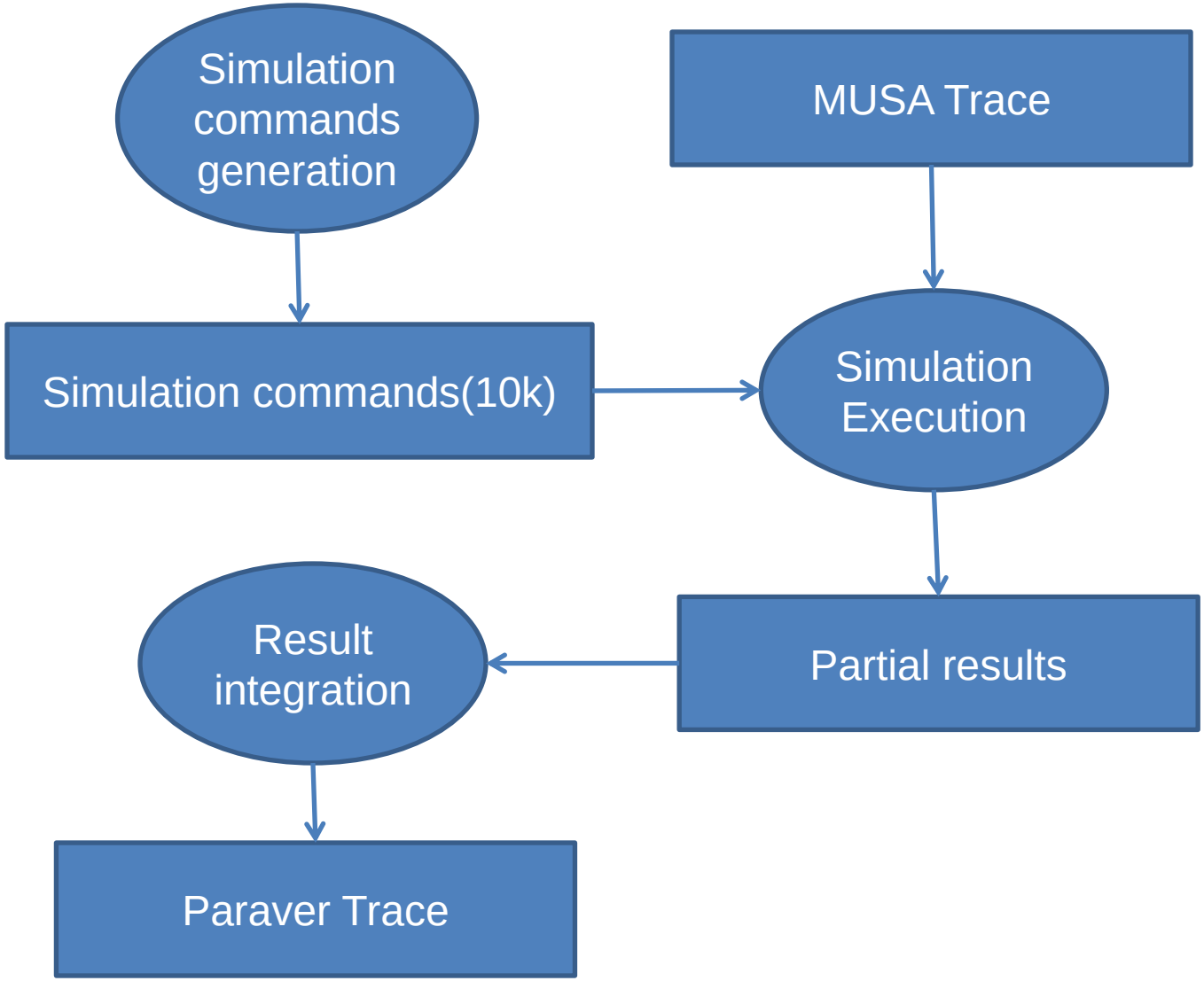
```
$> cd TRACE_sp-mz.B.4_000004_BRST/SIMULATION/A1_PRESIM
```

- Open any of the Hardware configuration files:

```
$> vim mn4_musa_000016_BRST.conf
```

- Contains sections for
  - CPU, Caches, Ram(ramulator)
  - Paraver simulation trace
  - McPAT additional data

Step 5: Burst mode simulation



Step 5: Burst mode simulation

- Generate the greasy command files:

```
$> ./generate_musa_presim.bash
```

- When finished submit all Burst mode simulations:

```
$> sbatch launch_all_musa_presims.bash
```

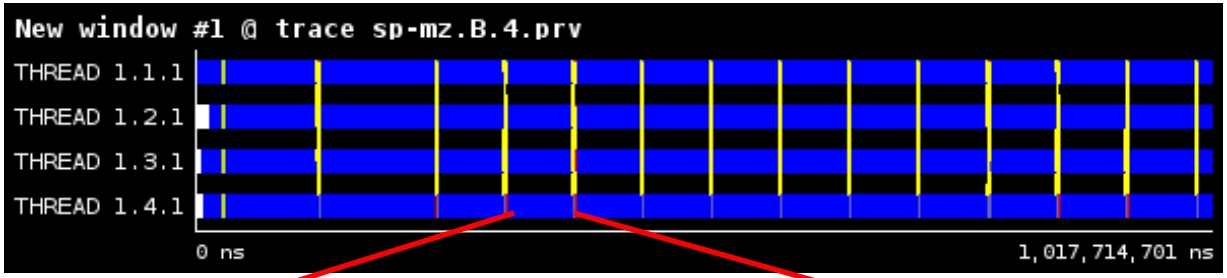
- If something fails:

```
$> ./cleanup.bash
```

Step 5: Burst mode simulation

What is actually happening:

- Running TaskSim Burst-mode simulations for each OpenMP phase between MPI events.
- In parallel



- Each computation phase is simulated like:

Step 5: Burst mode simulation

- Pre-simulation results:
  - musa_out_sp-mz.B.4_000001/

    mn4_musa_000001_BRST/

    mn4_musa_000001_BRST.dat
  - RANK:OPENMP_PHASE_ID:777:0:MEMORY_MODE?:DURATION

Step 6: Simulation integration and result analysis

- Integrating the results with Dimemas

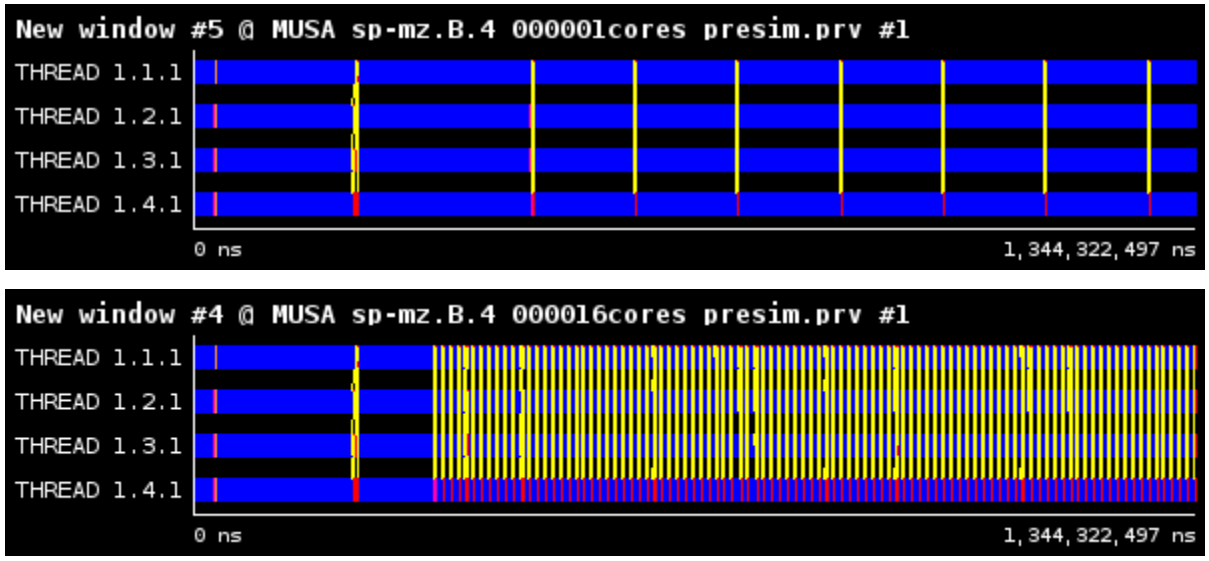
```
$> cd TRACE_sp-mz.B.4_000004_BRST/SIMULATION/
   A2_INTEGRATION_PRESIM
$> ./integrate_dimemas_simulations.bash
```

- This will:
  - Run Dimemas with the original extrae trace
  - Replace original phase duration with TaskSim Burst-mode results.
  - Generate Paraver traces for each configuration

Step 6: Simulation integration and result analysis

Open the Paraver trace:

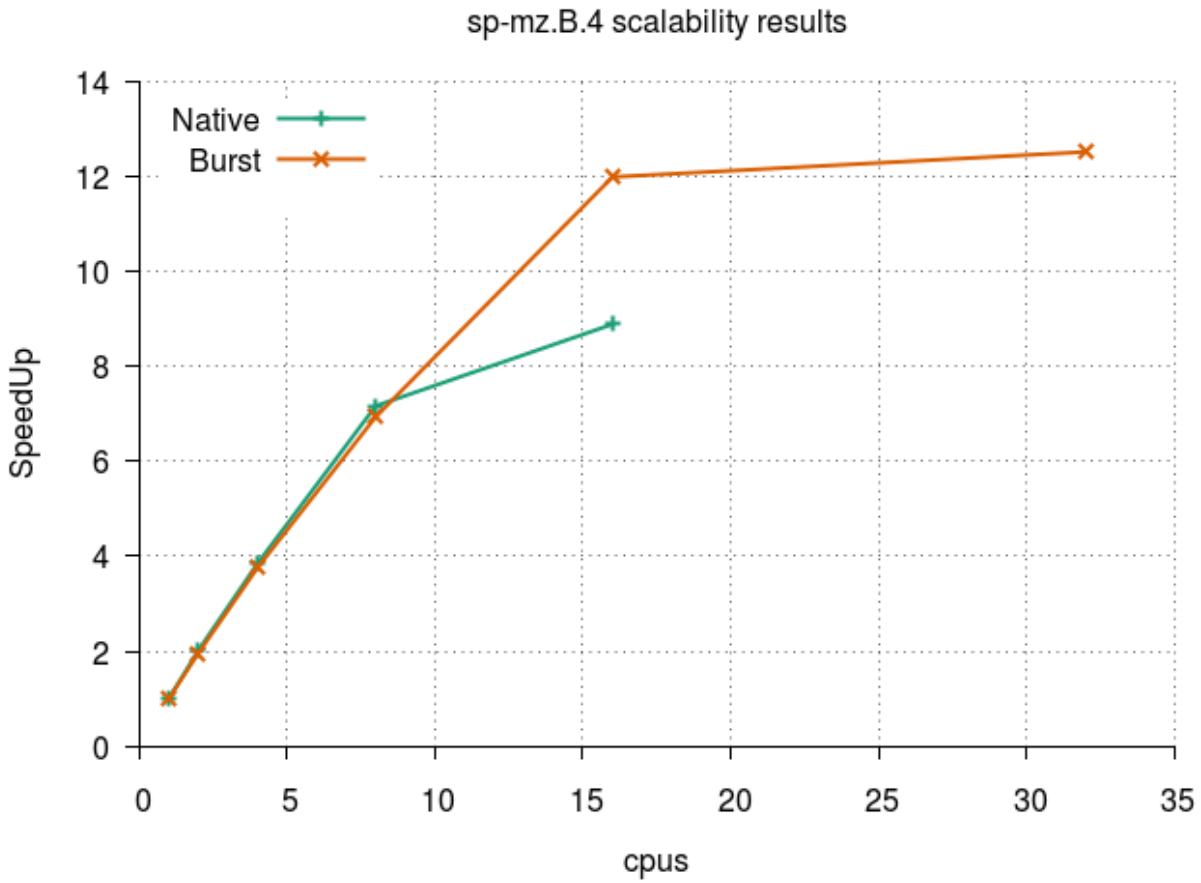
```
$> module load paraver
$> cd TRACE_sp-mz.B.4_000004_BRST/SIMULATION/
   A2_INTEGRATION_PRESIM/trace_SIMULATED
$> wxparaver MUSA_sp-mz.B.4_000016cores_mn4_musa_presim.prv
```



Step 6: Simulation integration and result analysis

- Generate the speedup graph:

```
$> ./generate_speedup_graph.bash
```



Questions?

Next session peek

- Please generate next session trace:

```
$> sbatch job_tracer_memory.bash
```