Evaluation and tuning of the performance of the dynamic load balancing of a legacy geophysics code using simulation

Rafael Keller Tesser[⋆], Lucas Mello Schnorr[⋆], Arnaud Legrand[†], Fabrice Dupros[‡], Philippe O. A. Navaux[⋆]

⋆: UFRGS PPGC/Inf, Porto Alegre, Brazil
{rktesser,schnorr,navaux}@inf.ufrgs.br

†: Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, Grenoble, France
arnaud.legrand@imag.fr

‡: BRGM, Orleans, France
f.dupros@brgm.fr

Porto Alegre, September 4th 2017, WSPPD

Outline

Dynamic load balancing for legacy parallel codes

- Unpredictable performance issues in legacy parallel codes
  - Inherent to the scientific model
  - Caused by hardware-software effects
  - Hard to solve by re-factoring
  - Example: Dynamic load imbalance

- Possible solution: runtime level load-balancing
  - Domain over-decomposition and process migration (*Charm++*)
  - MPI $\rightarrow$ Adaptive MPI (AMPI): MPI over Charm++
    - Few application modifications
    - Can migrate MPI tasks (*ranks*)
    - Each MPI rank is a *Virtual Processor* (VP)
    - Load balancer heuristics based on runtime information

Simulation based evaluation methodology

Problem: High cost to tune several load balancing parameters

- LB heuristics; LB frequency; level of over-decomposition; number of resources
- Platform specific: big/long allocations (production system)
- Time consuming: test parameter permutations; statistics

Simulation based evaluation methodology

Problem: High cost to tune several load balancing parameters

- LB heuristics; LB frequency; level of over-decomposition; number of resources
- Platform specific: big/long allocations (production system)
- Time consuming: test parameter permutations; statistics

Proposal: A simulation-based methodology to evaluate the potential benefits of adaptive MPI runtimes to legacy codes.

Simulation based evaluation methodology

Problem: High cost to tune several load balancing parameters

- LB heuristics; LB frequency; level of over-decomposition; number of resources
- Platform specific: big/long allocations (production system)
- Time consuming: test parameter permutations; statistics

Proposal: A simulation-based methodology to evaluate the potential benefits of adaptive MPI runtimes to legacy codes.

Goal: Speeding up the evaluation of over-decomposition coupled with dynamic load-balancing with little application modification.

Simulation based evaluation methodology

Problem: High cost to tune several load balancing parameters

- LB heuristics; LB frequency; level of over-decomposition; number of resources
- Platform specific: big/long allocations (production system)
- Time consuming: test parameter permutations; statistics

Proposal: A simulation-based methodology to evaluate the potential benefits of adaptive MPI runtimes to legacy codes.

Goal: Speeding up the evaluation of over-decomposition coupled with dynamic load-balancing with little application modification.

Relies on SimGrid: MPI Emulation (SMPI) and Trace Replay

Outline

Ondes3D: A typical MPI iterative code

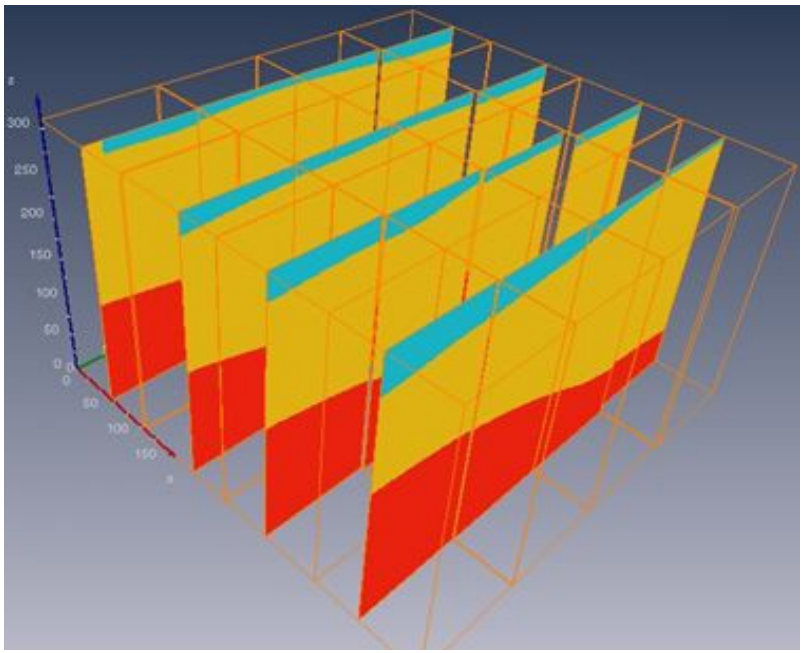


Figure: 3D rock medium; 4×4 domain decomposition; each process calculates a cuboid.

```
for (ts = 0; ts < N; ts++){
  Intermediates();

  Stress();
  //Intertwined Asynchronous
  //Neighborhood Communication

  Velocity();
  //Intertwined Asynchronous
  //Neighborhood Communication

}
```

Figure: The three main computation kernels.

- A typical, seemingly regular, MPI iterative code;
- Regular 2D decomposition of 3D space;
- Main loop:
  - 3 main computation kernels: Stress, Intermediates, Velocity;
  - 2 asynchronous communication phases interleaved with computation.

Ondes3D: Spatial and temporal load imbalance

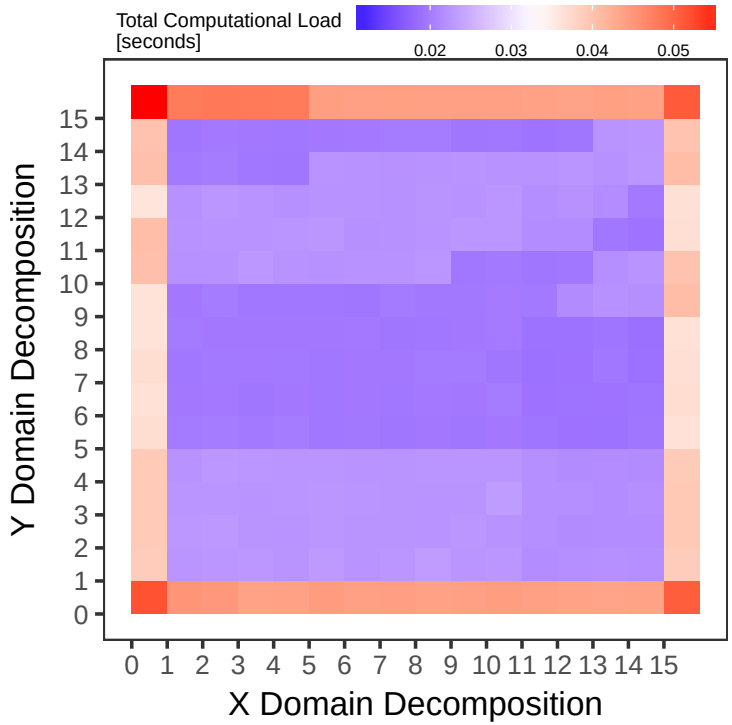


Figure: Spatial imbalance for the first iteration represented by a color gradient for each rank in a 16×16 grid (256 processes)

Spatial load imbalance (left):

- Borders: *Absorbing boundary condition* (CPML)
- Smaller imbalances: dependent on the *rock multi-layer* configuration of the *input* (six layers for this scenario)

Ondes3D: Spatial and temporal load imbalance

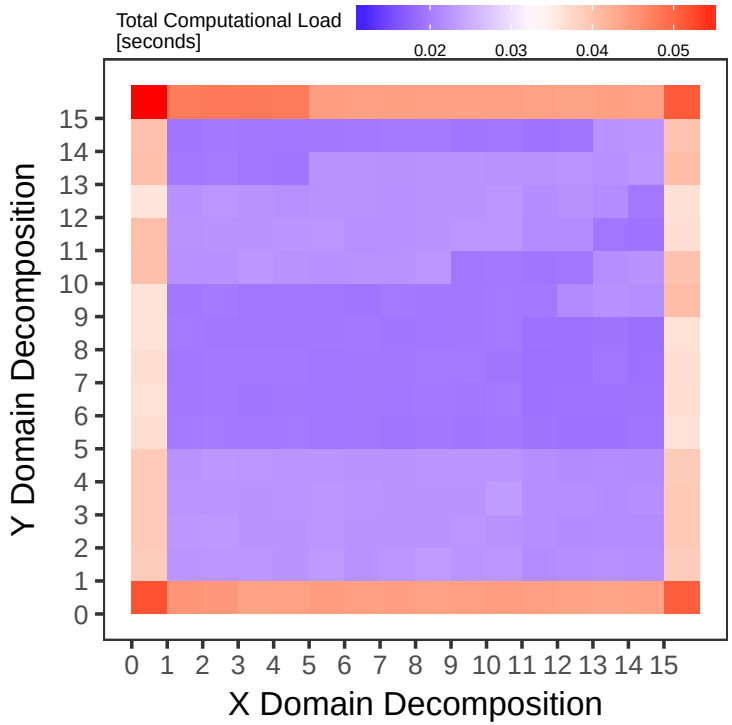


Figure: Spatial imbalance for the first iteration represented by a color gradient for each rank in a 16×16 grid (256 processes)

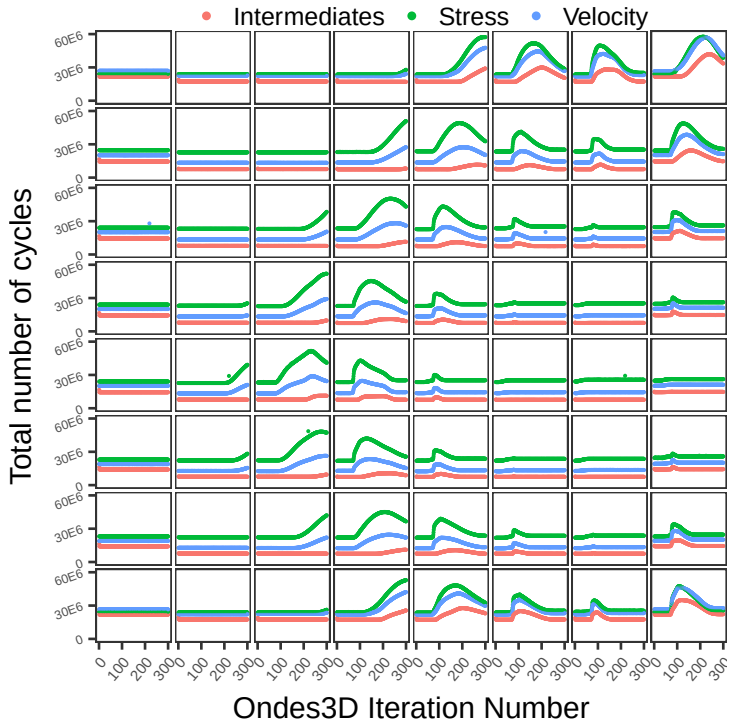


Figure: Total number of cycles (from PAPI) for each kernel along iterations for each rank in a 8×8 grid (64 processes).

Temporal (dynamic) load imbalance (right):

- Variation in the number of cycles per iteration
- Nothing in code the indicates such temporal irregularity

Ondes3D: Spatial and temporal load imbalance

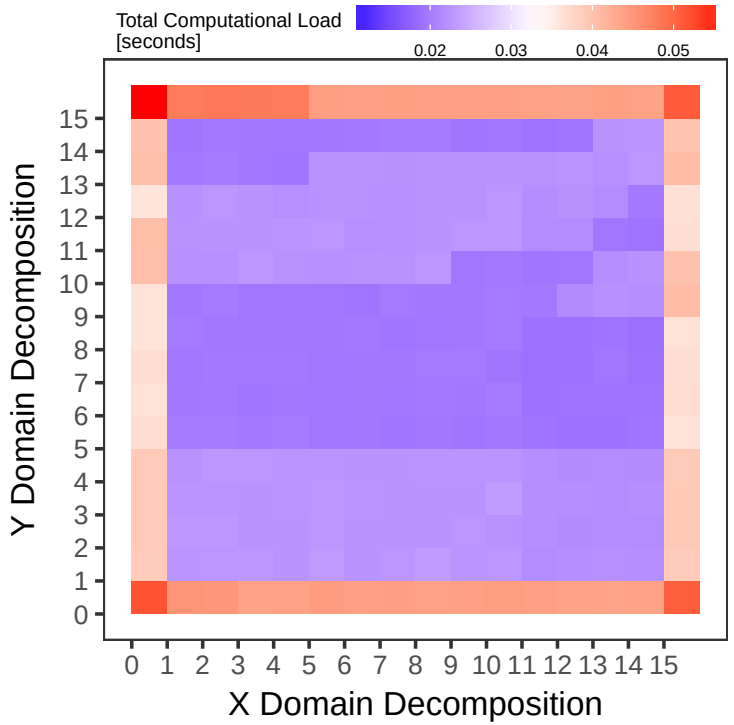


Figure: Spatial imbalance for the first iteration represented by a color gradient for each rank in a 16×16 grid (256 processes)

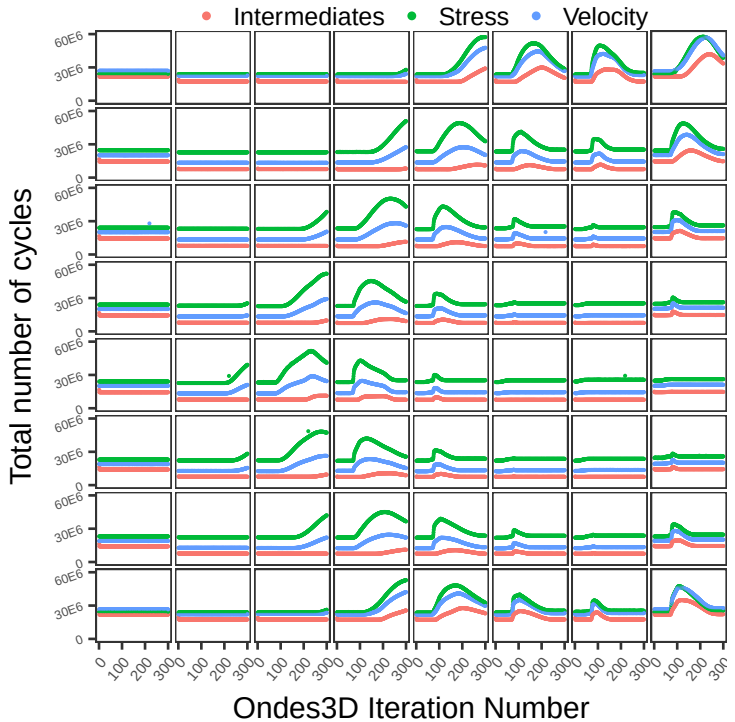
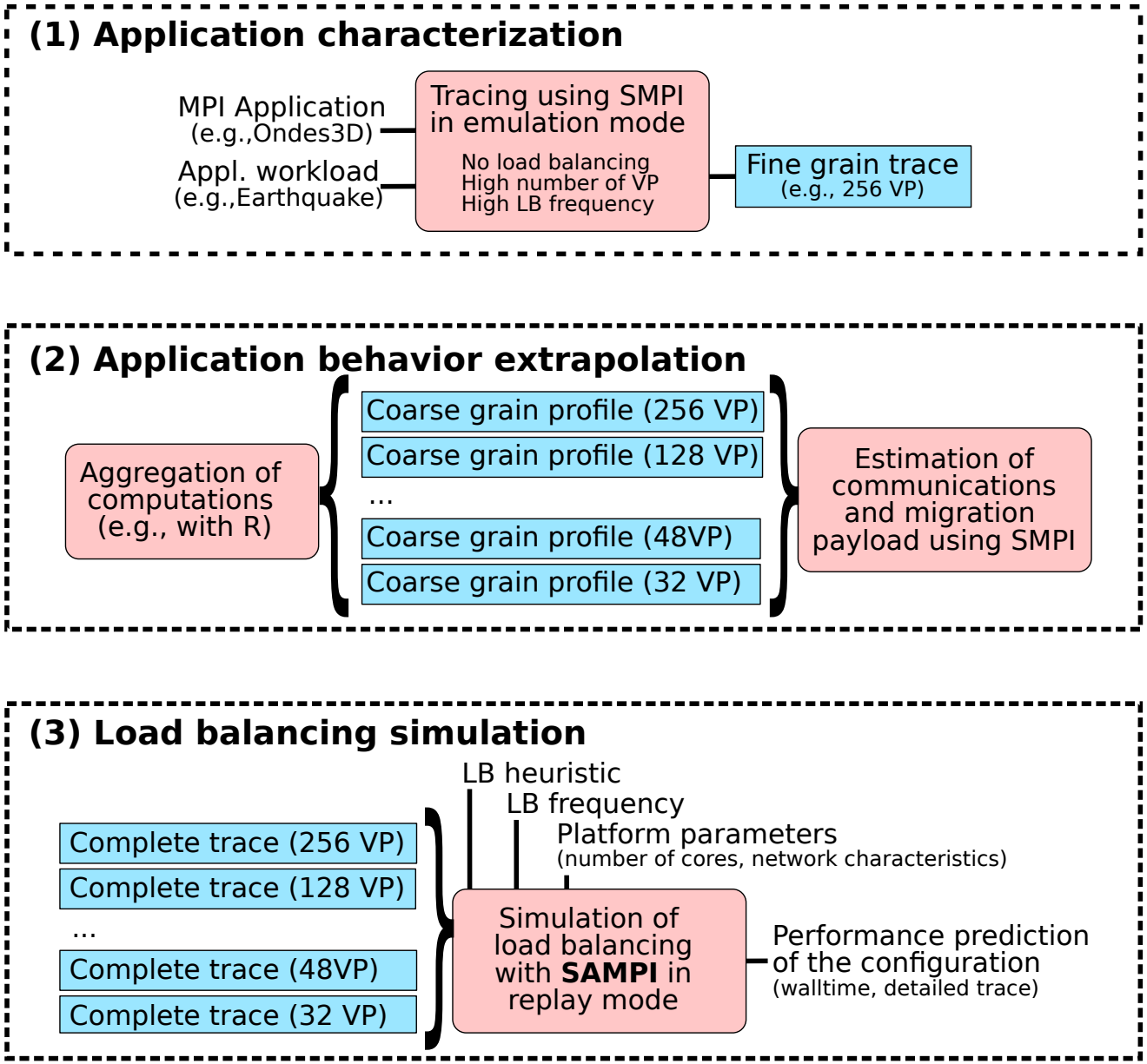


Figure: Total number of cycles (from PAPI) for each kernel along iterations for each rank in a 8×8 grid (64 processes).

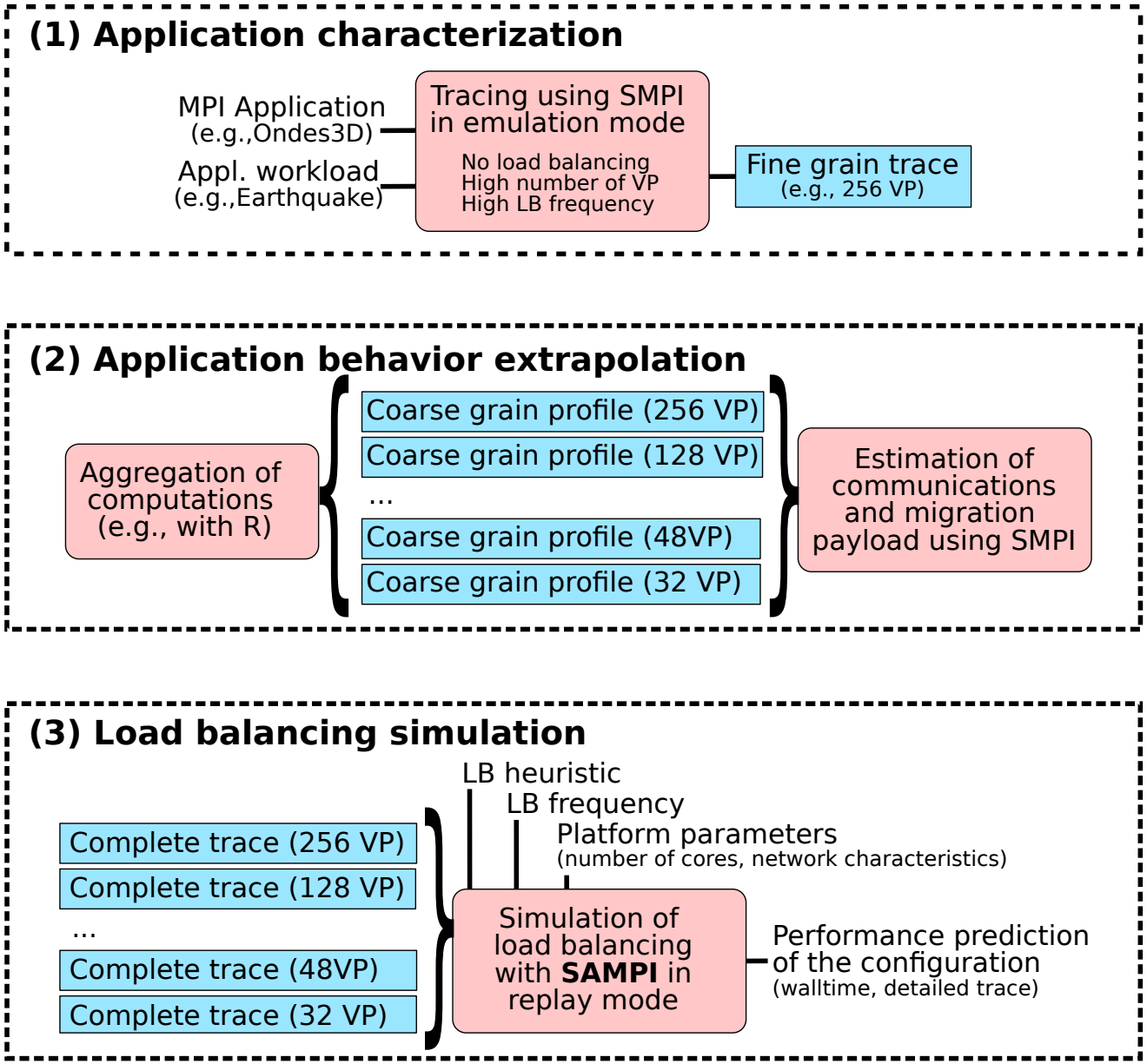
- Ondes3D was ported to AMPI in a previous work (≈24% faster)
- Later execution on a 288 core cluster: ≈38% faster

Outline

Simulation based evaluation approach

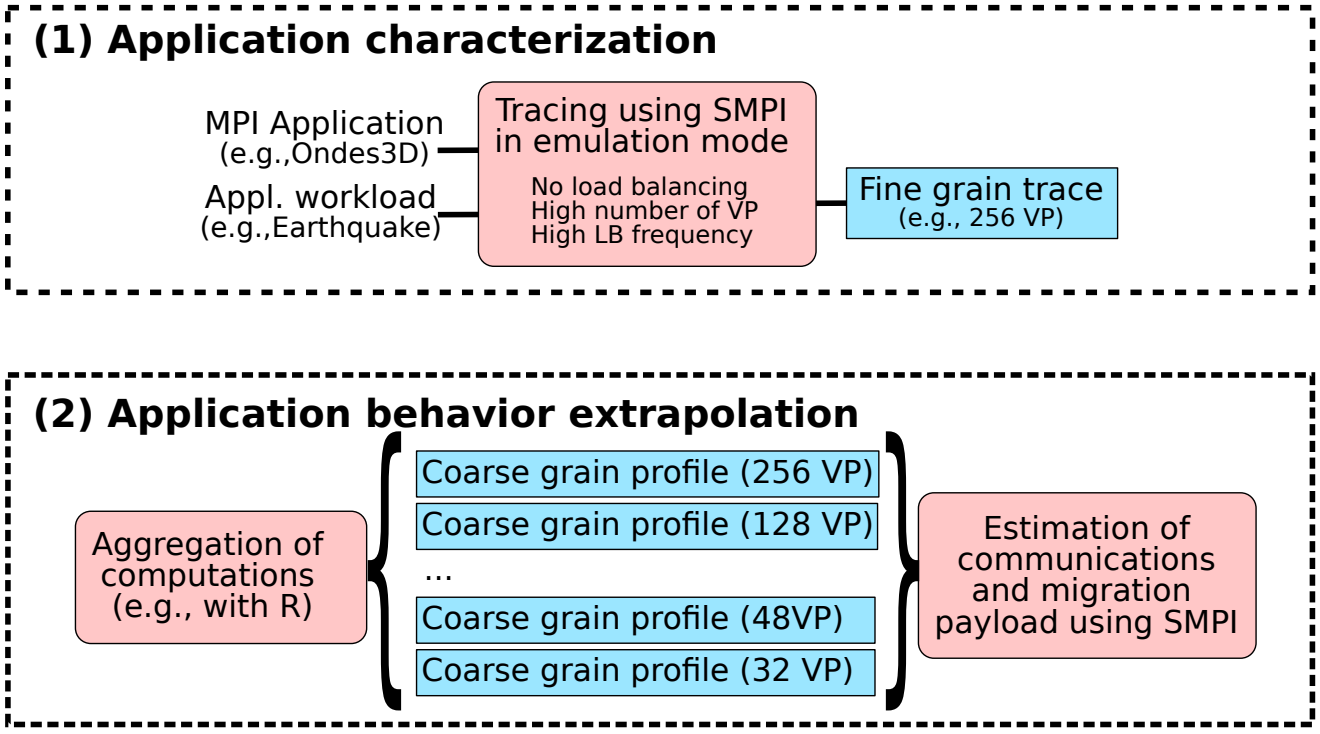


Simulation based evaluation approach



(1) Executed (*emulated*) once on a single host, per input data set.

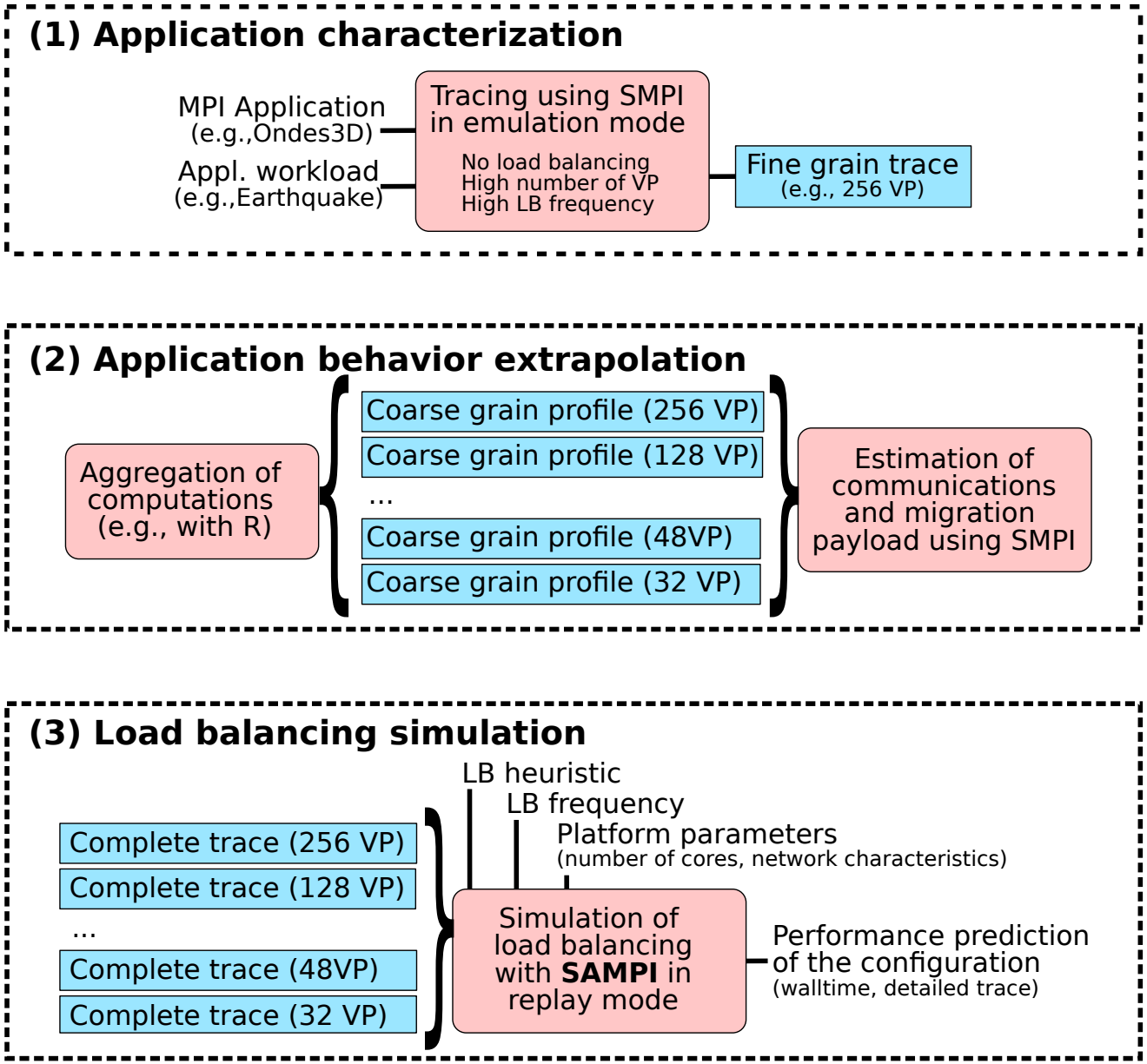
Simulation based evaluation approach



(1) Executed (*emulated*) once on a single host, per input data set.

(2) Fast simulation of multiple levels of over-decomposition from a single fine grain trace.

Simulation based evaluation approach

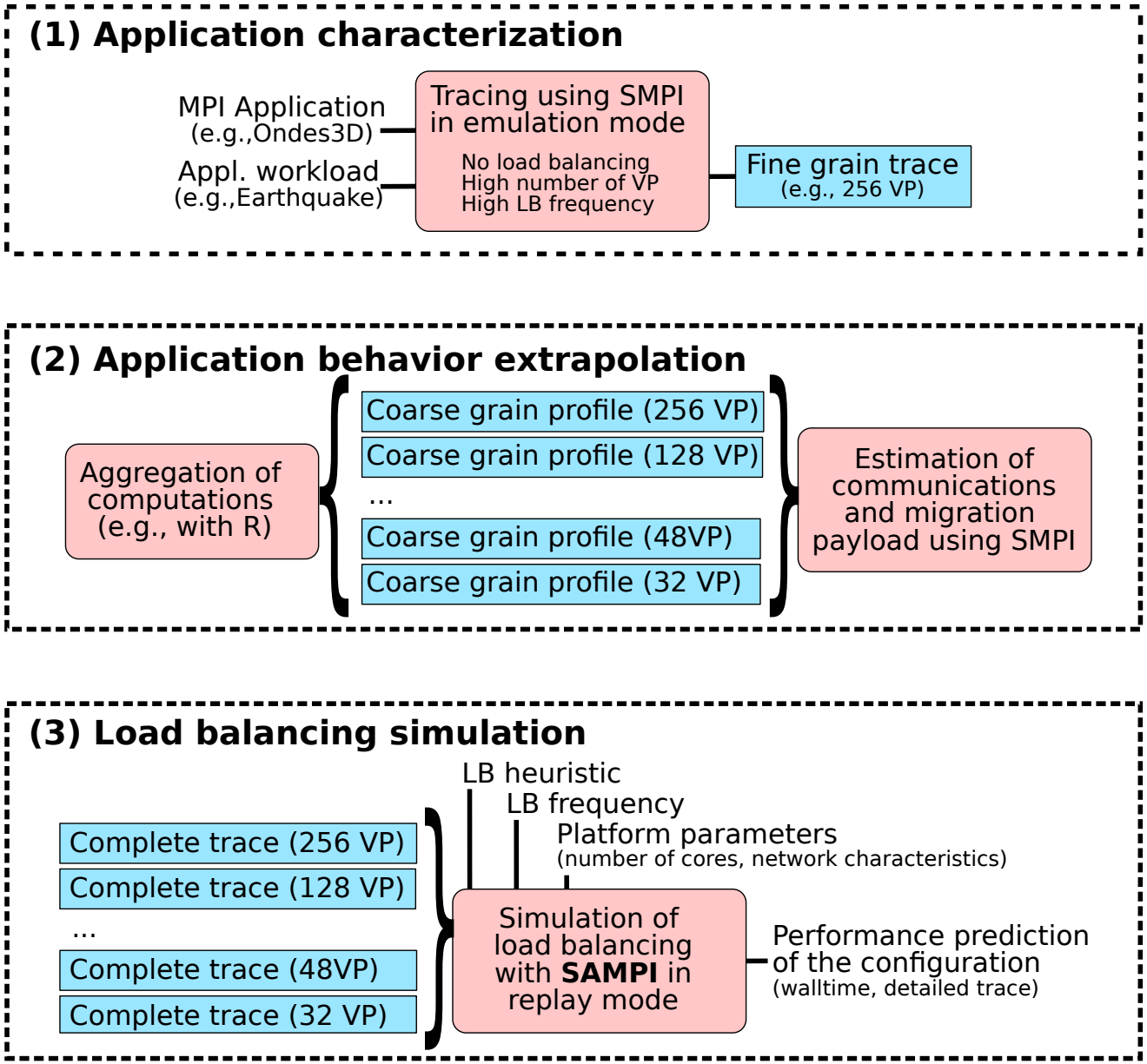


(1) Executed (*emulated*) once on a single host, per input data set.

(2) Fast simulation of multiple levels of over-decomposition from a single fine grain trace.

(3) Fast simulation of multiple load balancing parameter combinations and execution platforms from a single trace.

Simulation based evaluation approach



Result

Fast simulation of any number of load balancing configurations (heuristic, frequency and level of over-decomposition) on different platforms from a single full emulation trace, using a single host.

SAMPI: Simulated Adaptive MPI

- Implements the simulation of AMPI-like load balancing
  - Using SimGrid's MPI emulation and trace-replay
- Inclusion of two adapted Charm++ load balancers:
  - GreedyLB and RefineLB
- Requires very small modification to the MPI application
  - Call `MPI_Migrate`: indicate when the process is ready to migrate

Outline

Validation: SAMPI (simulation) vs AMPI

Two real earthquake scenarios:

Validation: SAMPI (simulation) vs AMPI

Two real earthquake scenarios:

- Niigata-ken Chuetsu-Oki:
  - 2007, Mw 6.6, Japan
  - 500 time-steps
  - dimensions: 300x300x150

Validation: SAMPI (simulation) vs AMPI

Two real earthquake scenarios:

- Niigata-ken Chuetsu-Oki:
  - 2007, Mw 6.6, Japan
  - 500 time-steps
  - dimensions: 300x300x150

- Ligurian:
  - 1887, Mw 6.3, north-western Italy
  - 300 times-steps
  - dimensions: 500x350x130

Validation: SAMPI (simulation) vs AMPI

Load balancers: No load balancing vs. GreedyLB vs. RefineLB

Level of over-decomposition: 64 VP mapped to 16 processes (16x4)

Frequency of load balancing: Every 20 time-steps

Hardware Resources: *Parapluie* cluster from Grid'5000

- 2 x AMD Opteron™ 6164 HE x 12 cores, 1.7GHz, Infiniband
- Plus my own laptop (Intel Core™ i7-4610M, 2 cores, 3GHz)

Chuetsu-Oki earthquake: 16x4 over-decomposition level

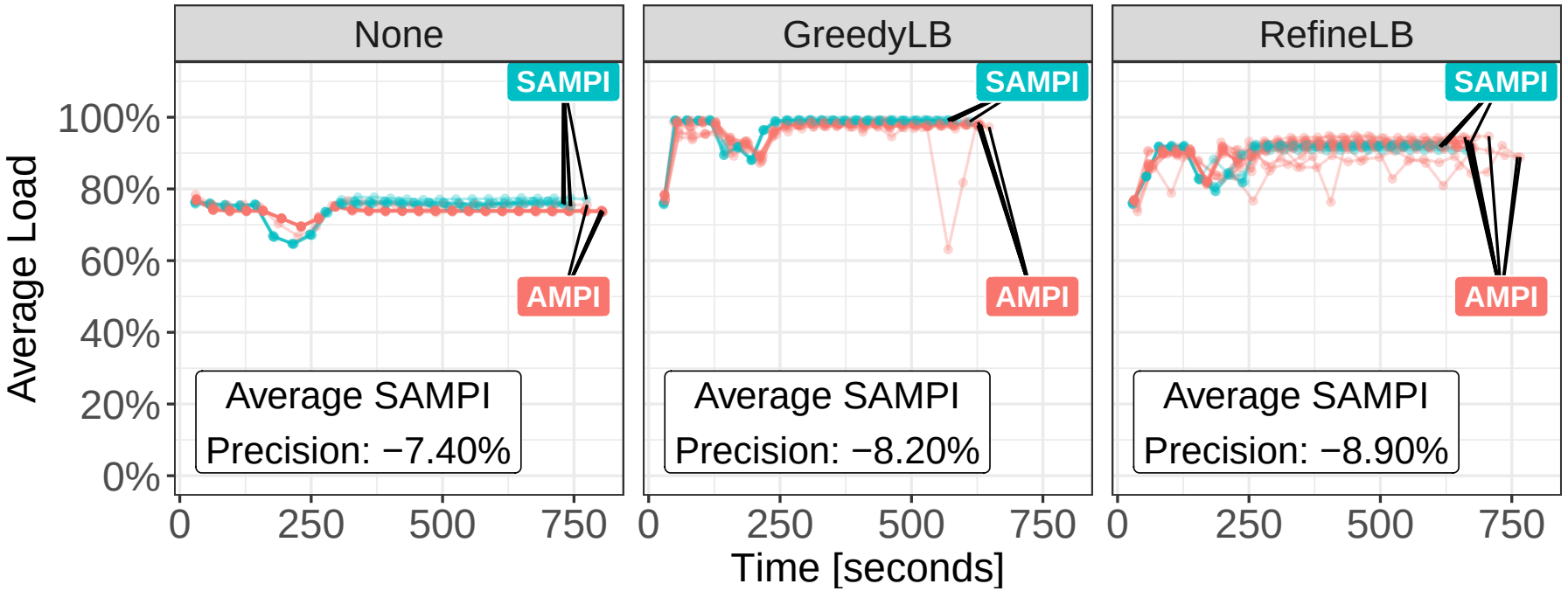


Figure: Average load and makespan.

- The simulated load behaves very similar to real life
- GreedyLB is the best choice in both simulation and real-life.
- Simulation is slightly optimistic in terms of average total makespan

Ligurian earthquake: 16x4 over-decomposition level

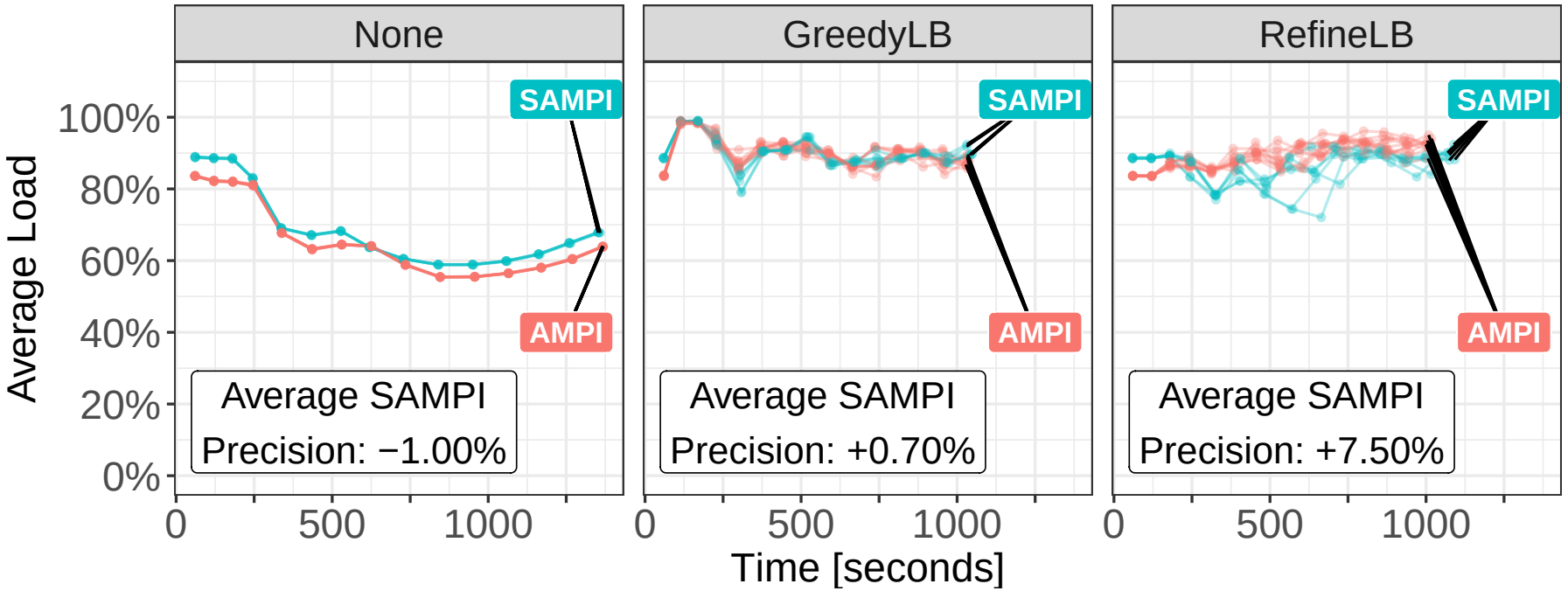


Figure: Average load and makespan.

- Once again, the simulated and real-life loads behave similarly
- RefineLB is the best choice in both simulation and real-life
- For RefineLB, the predicted average makespan is slight pessimistic

Tuning: frequency and over-decomposition level

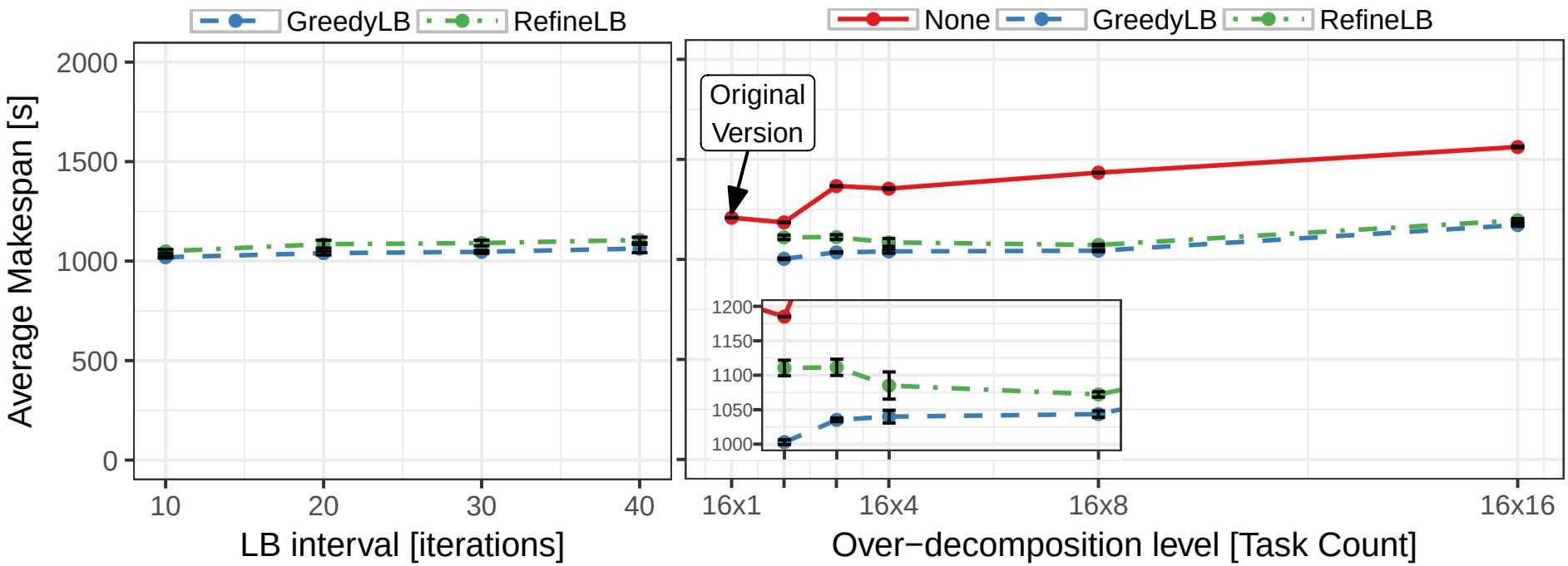


Figure: Simulated makespan predictions for the **Ligurian** earthquake simulation with (left) four load balancing intervals (in number of iterations) and (right) with six over-decomposition levels (1, 2, 3, 4, 8, and 16) on 16 cores (LB every 20 iterations).

- Load balancing frequency: none or little influence on average makespan
- Over-decomposition level:
  - Best for RefineLB $\rightarrow$ 16×8 (13% gain)
  - Best for GreedyLB $\rightarrow$ 16×2 (19% gain)

Outline

Conclusion and Future Work

- A simulation based approach:
  - performance evaluation and tuning of dynamic load balancing
  - applied to iterative MPI applications
  - Low cost in terms of time and resource requirements

- Contributions:
  1. Analysis of load balance in Ondes3D
     - dynamic imbalance even when there is no indication in the code
  2. A validated simulator (SAMPI)
  3. SAMPI prediction of the influence of load balancing parameters

Conclusion and Future Work

- The structure of Ondes3D is very typical among MPI applications
  - Therefore, we believe that the usefulness of our approach is not limited to Ondes3D

- Future work:
  - Explore the use spatial aggregation and trace-extrapolation to further accelerate the simulations

Conclusion and Future Work

- The structure of Ondes3D is very typical among MPI applications
  - Therefore, we believe that the usefulness of our approach is not limited to Ondes3D

- Future work:
  - Explore the use spatial aggregation and trace-extrapolation to further accelerate the simulations

Paper published in Euro-Par 2017

"Using Simulation to Evaluate and Tune the Performance of Dynamic Load Balancing of an Over-decomposed Geophysics Application"

Acknowledgments

This research was partially supported by:

- HPC4E: This research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E Project, grant agreement number 689772.
- FAPERGS: This work is partially supported by Green-Cloud project, funded by the Foundation for Research Support of Rio Grande do Sul (FAPERGS).
- LICIA: This research is in the context of the LICIA, a CNRS Équipe Associé joint laboratory between the Institute of Informatics of UFRGS and the Laboratoire d'Informatique de Grenoble (LIG).
- HPC-GA: This work was partially supported by the HPC-GA project which has received funding from the European Community's Seventh Framework Programme (FP7-PEOPLE) under grant agreement number 295217.
- CNPq: PhD Scholarship at the Post-Graduate Program in Computer Science (PPGC) at UFRGS, funded by the Brazil's National Council for Scientific Research (CNPq).
- CAPES-COFECUB: Part of this work was conducted during a sandwich doctorate scholarship at the Laboratoire d'Informatique de Grenoble, supported by the International Cooperation Program CAPES/COFECUB Foundation; financed by CAPES within the Ministry of Education of Brazil (project number 764-13).
- Grid'5000: Some experiments were carried out at the Grid'5000 platform (https://www.grid5000.fr), with support from Inria, CNRS, RENATER and several other organizations.

Evaluation and tuning of the performance of the dynamic load balancing of a legacy geophysics code using simulation

Rafael Keller Tesser[⋆], Lucas Mello Schnorr[⋆], Arnaud Legrand[†], Fabrice Dupros[‡], Philippe O. A. Navaux[⋆]

⋆: UFRGS PPGC/Inf, Porto Alegre, Brazil

{rktesser,schnorr,navaux}@inf.ufrgs.br

†: Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, Grenoble, France

arnaud.legrand@imag.fr

‡: BRGM, Orleans, France

f.dupros@brgm.fr

Porto Alegre, September 4th 2017, WSPPD

Chuetsu-Oki earthquake: 16x4 over-decomposition level

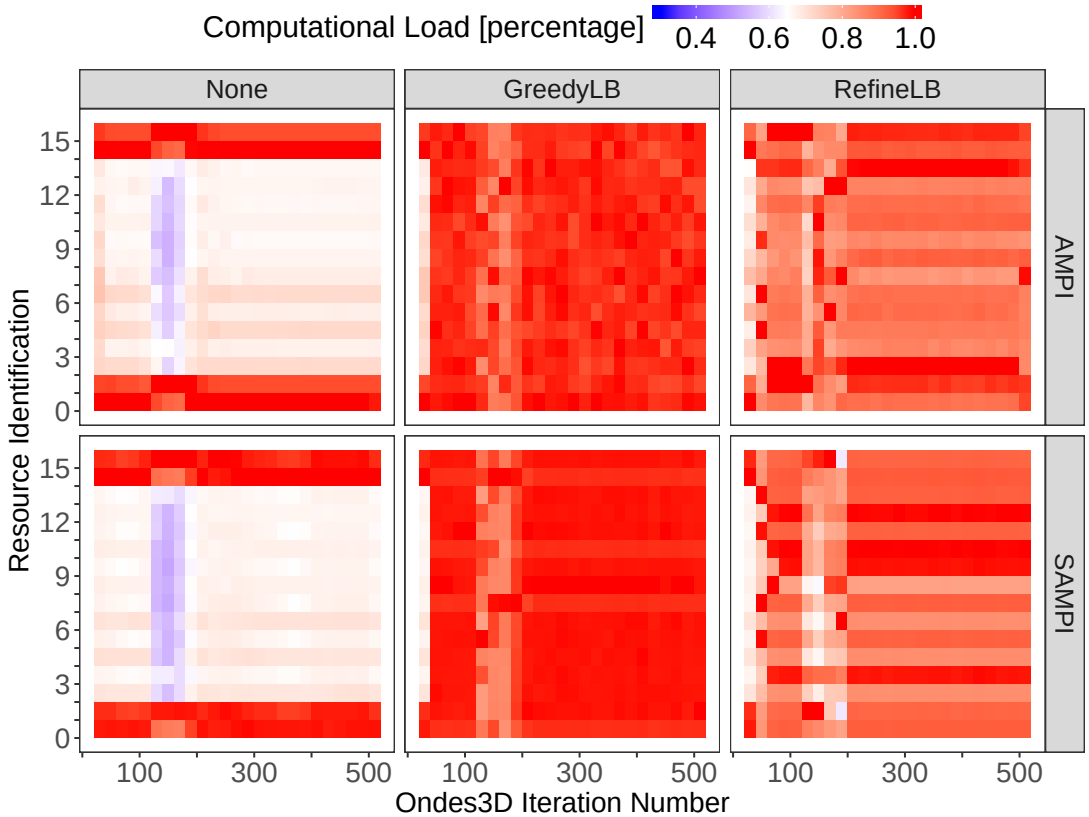


Figure: Per-process Computational Load

- We can clearly see the imbalance when running without an LB
- GreedyLB seems to have the best load distribution both simulation and RL
- But this chart shows only one execution...

Ligurian earthquake: 16x4 over-decomposition level

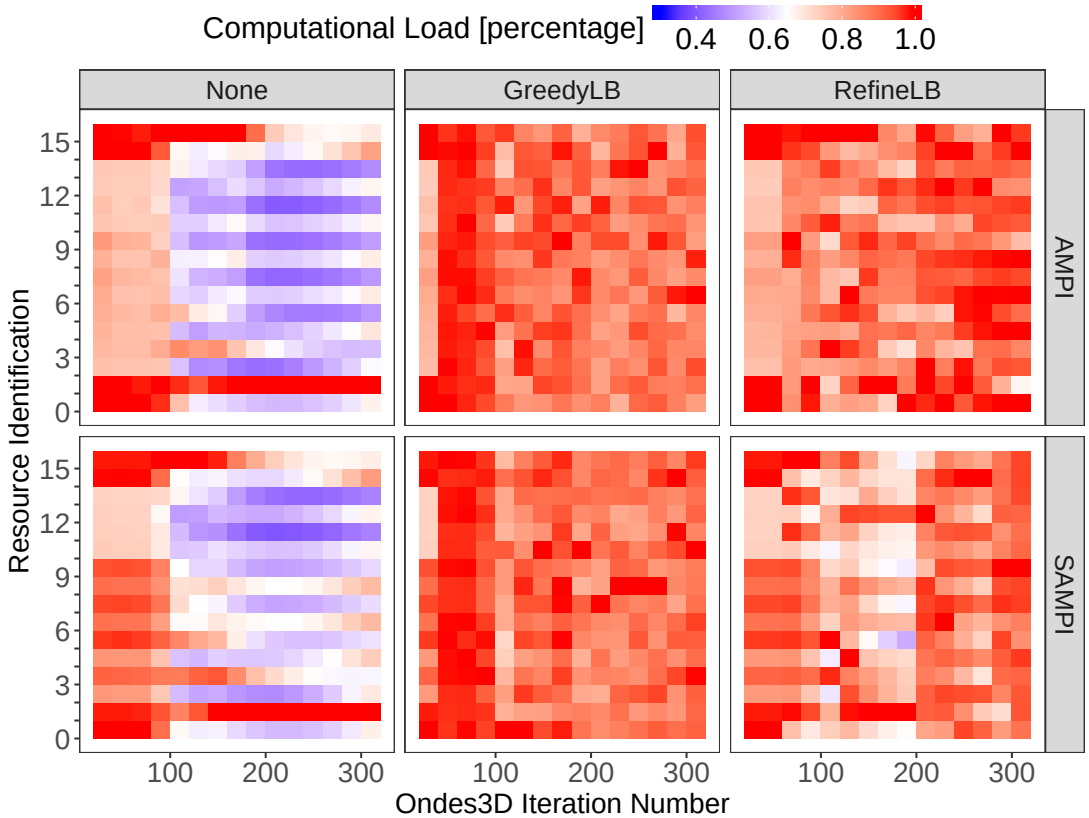


Figure: Per-process computational load.

- The Ligurian simulation has more dynamic imbalance than Chuetsu-Oki
- From this chart, it seems that GreedyLB has the best load distribution