Using SimGrid to Evaluate the Impact of AMPI Load Balancing In a Geophysics HPC Application

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

⋆: UFRGS GPPD/Inf, Porto Alegre, Brazil
†: CNRS/Inria POLARIS, Grenoble, France

Urbana, April 2016
14th Charm++ Workshop

Outline

Context

Parallel HPC applications are often written with MPI, which is based on a regular SPMD programming model.

- Many of these applications are iterative and such paradigm is suited to balanced applications;
- Unbalanced applications:
  - May resort to static load balancing techniques (at application level ☹)
  - Or not... (the load imbalance comes from the nature of the input data, evolve over time and space. e.g., Ondes3D)

    Handling this at the application level is just a nightmare.

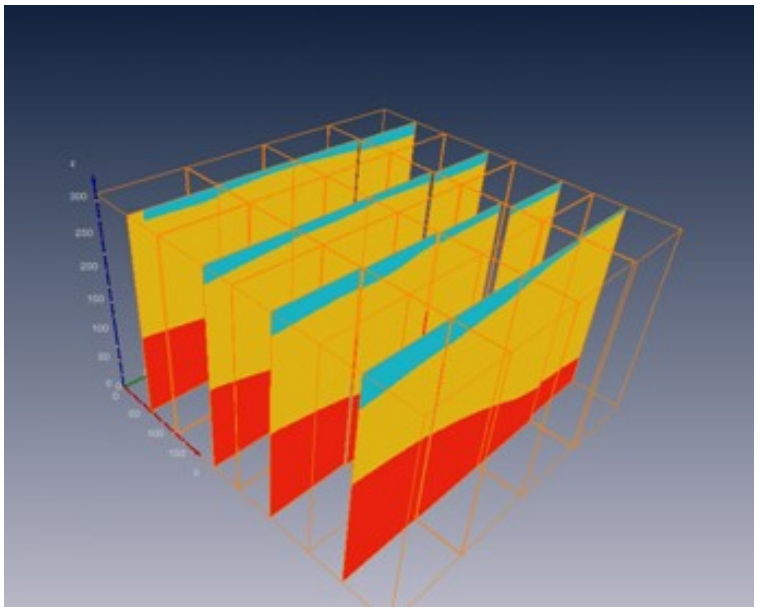
A possible approach is to use over-decomposition and dynamic process-level load-balancing as proposed with AMPI/CHARM++

Ondes3D, a Seismic Wave Propagation Simulator

- Developed by BRGM [Aochi et al. 2013];
- Used to predict the consequences of future earthquakes.

Many sources of load imbalance:

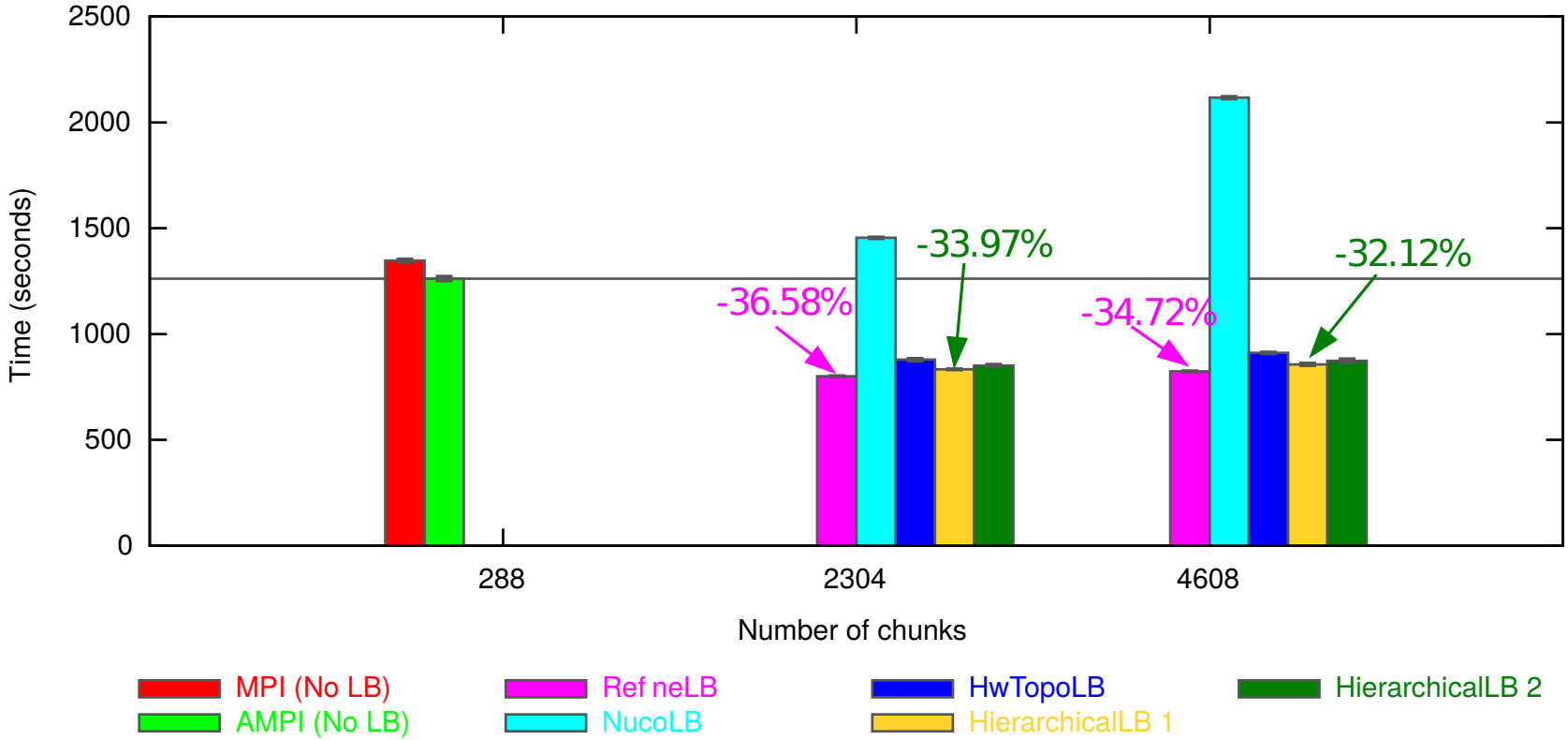
- Absorbing boundary conditions (tasks at the borders perform more computation)
- Variation in the constitution laws of different geological layers (different equations);



- Propagation of the shockwave in space and time;

Mesh partitioning techniques and quasi-static load balancing algorithm are thus ineffective.

AMPI can be quite effective



500 time-steps Average execution times

Based on Mw 6.6, 2007 *Niigata Chuetsu-Oki, Japan*, earthquake [Aochi et.al ICCS 2013]

- Full problem (6000 time steps) $\rightsquigarrow$ 162 minutes on 32 nodes (Intel Hapertown processors)

Challenges

Finding the best load balancing parameters:

- Which Load Balancer is the most suited?
- How many iterations should be grouped together? (Migration Frequency)
- How many VPs? (Decomposition level) Load-balancing benefit vs. application communication overhead and LB overhead
- ...

Challenges

Finding the best load balancing parameters:

- Which Load Balancer is the most suited?
- How many iterations should be grouped together? (Migration Frequency)
- How many VPs? (Decomposition level) Load-balancing benefit vs. application communication overhead and LB overhead
- ...

And preparing for AMPI is **not free**:

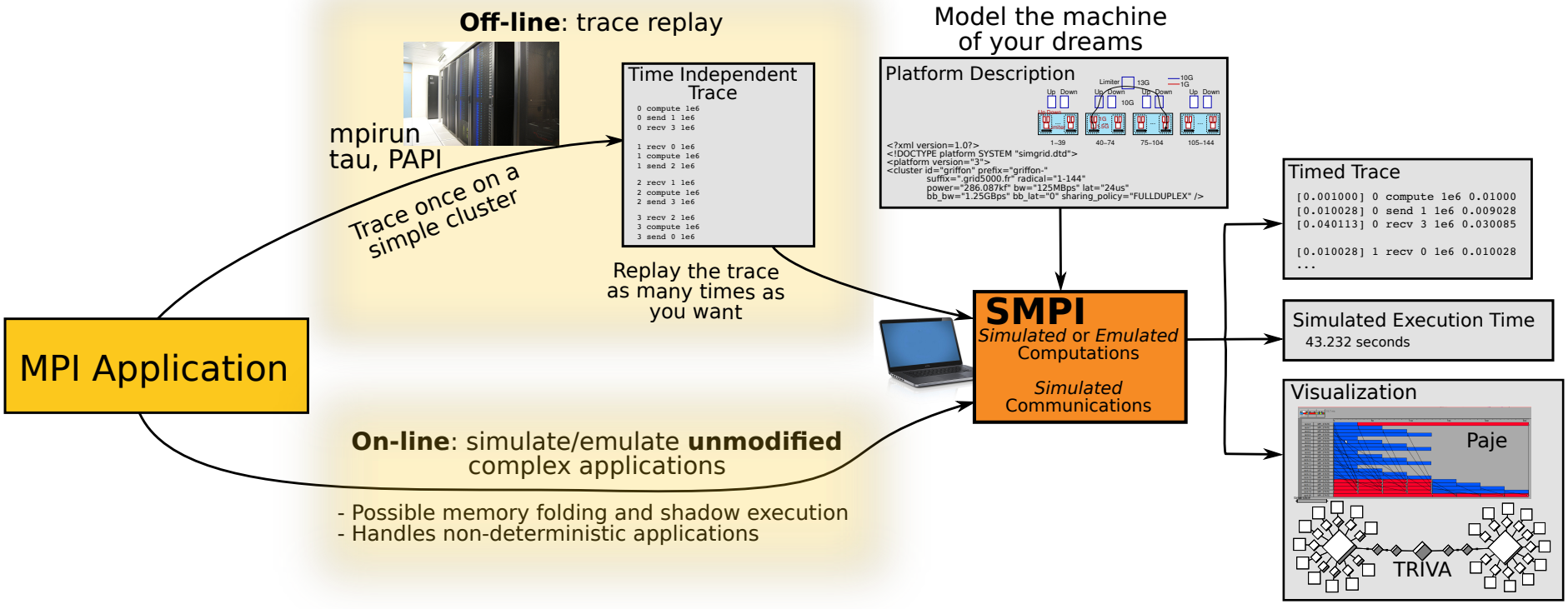
- Need to **write data serialization code**
- Engaging in such approach without knowing how much there is to gain can be deterring;

Goal

Propose a sound methodology for investigating performance improvement of irregular applications through over decomposition

Outline

SimGrid



- SimGrid: 15 years old collaboration between France, US, UK, Austria, ...
- Flow-level models that account for topology and contention
- SMPI: Supports both trace replay and direct emulation
- Embeds 100+ collective communication algorithms

Outline

Principle

Approach:

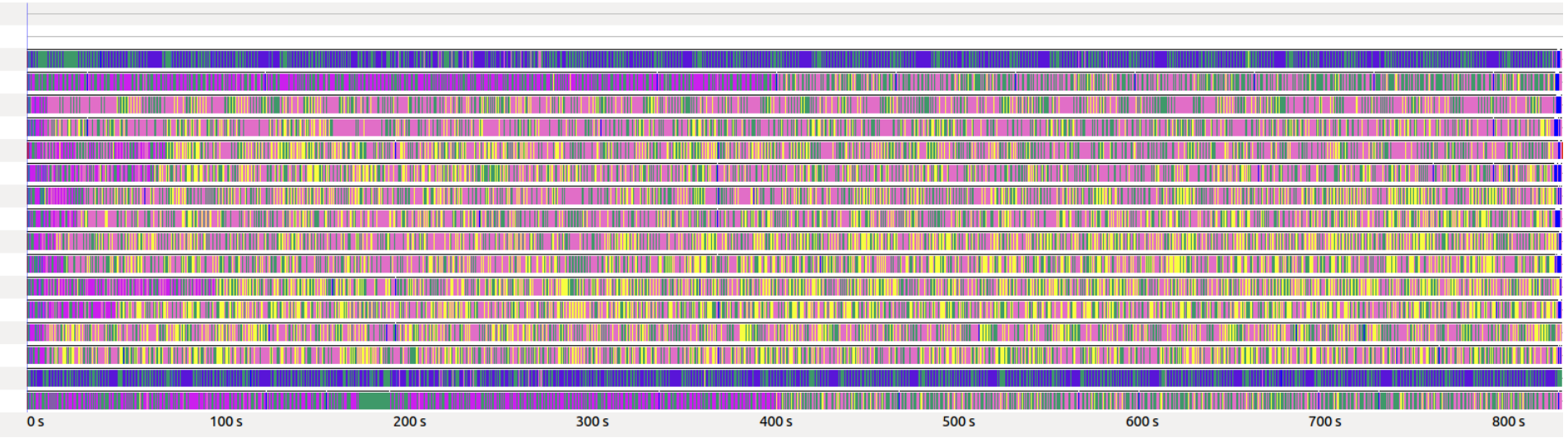
1. Implement various load-balancing algorithms in SMPI;
2. Capture a time independent trace (faithful application profile)
   - Two alternatives:
     - Standard tracing: parallel/fast ☺, requires more resources ☹
     - Emulation (`smpicc/smpirun`): requires a single host ☺, slow ☹;
   - Add a fake call to `MPI_Migrate` where needed
   - Track how much memory is used by each VP and use it as an upper bound of migration cost;
   - May take some time but does requires minimal modification / knowledge of the application;
3. Replay the trace as often as wished, playing with the different parameters (LB, frequency, topology, ...)

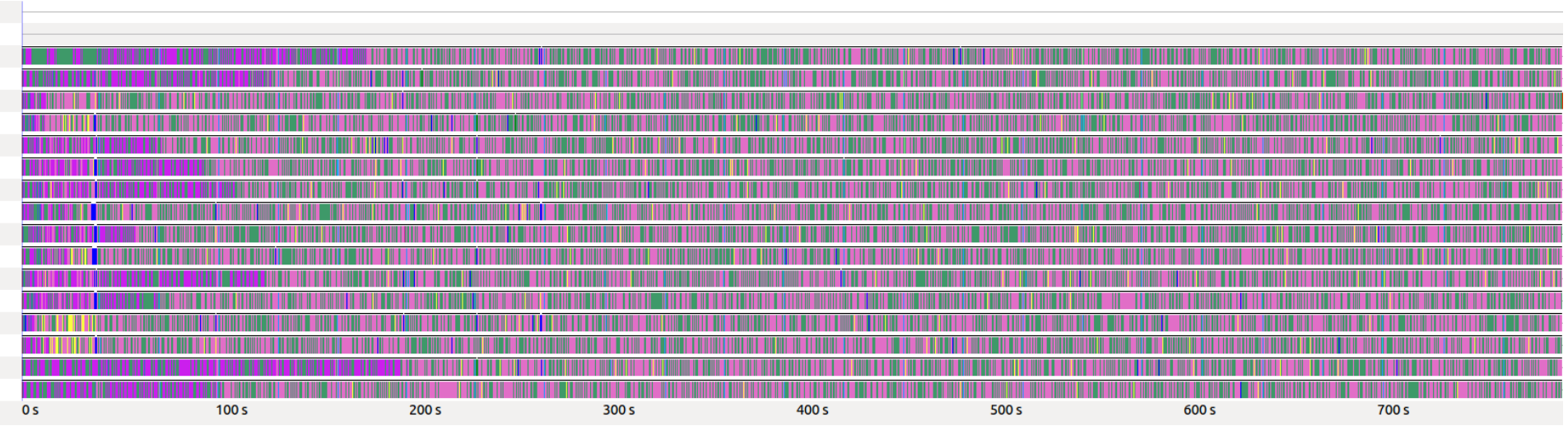
Principle

Key questions:

- How do we know whether our simulations are faithful?
- How do we understand where the mismatch comes from?
  - VP scheduling, LB implementation, trace capture, network, ...

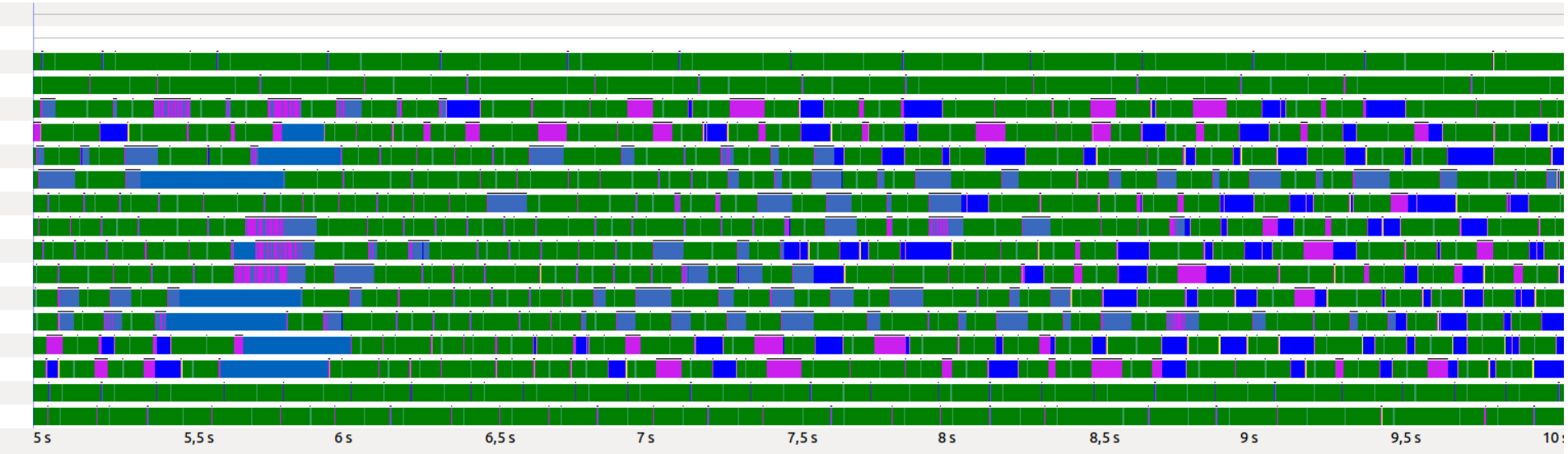
Evaluation Challenge

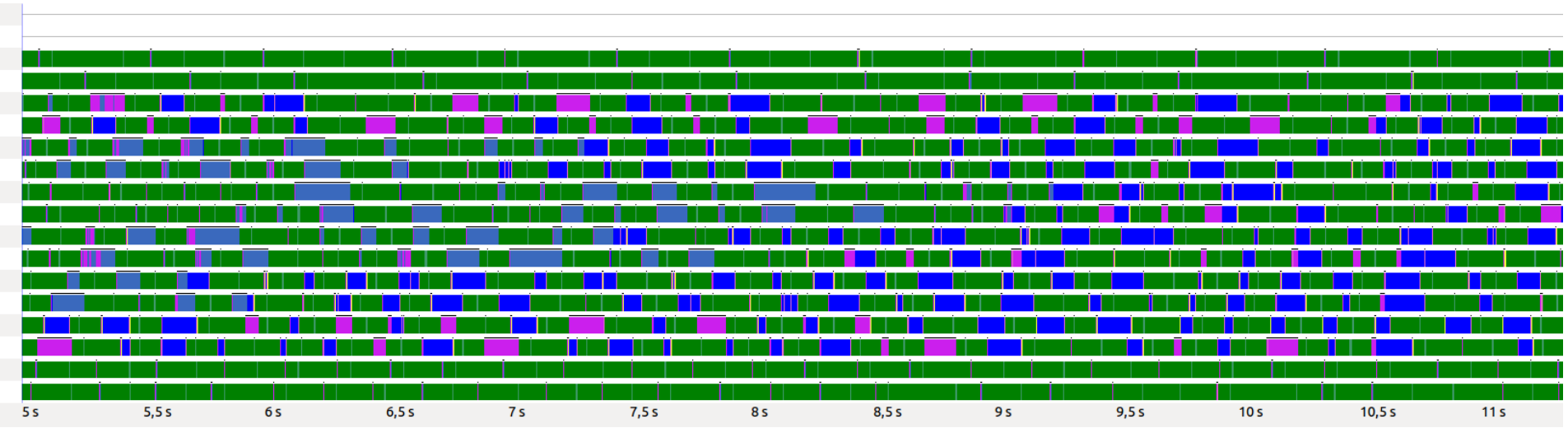




No LB vs. GreedyLB : Simple Gantt charts are not very informative

Evaluation Challenge





No LB vs. GreedyLB : Simple Gantt charts are not very informative

Outline

Description of the experiments

Scenarios Two different earthquake simulations:

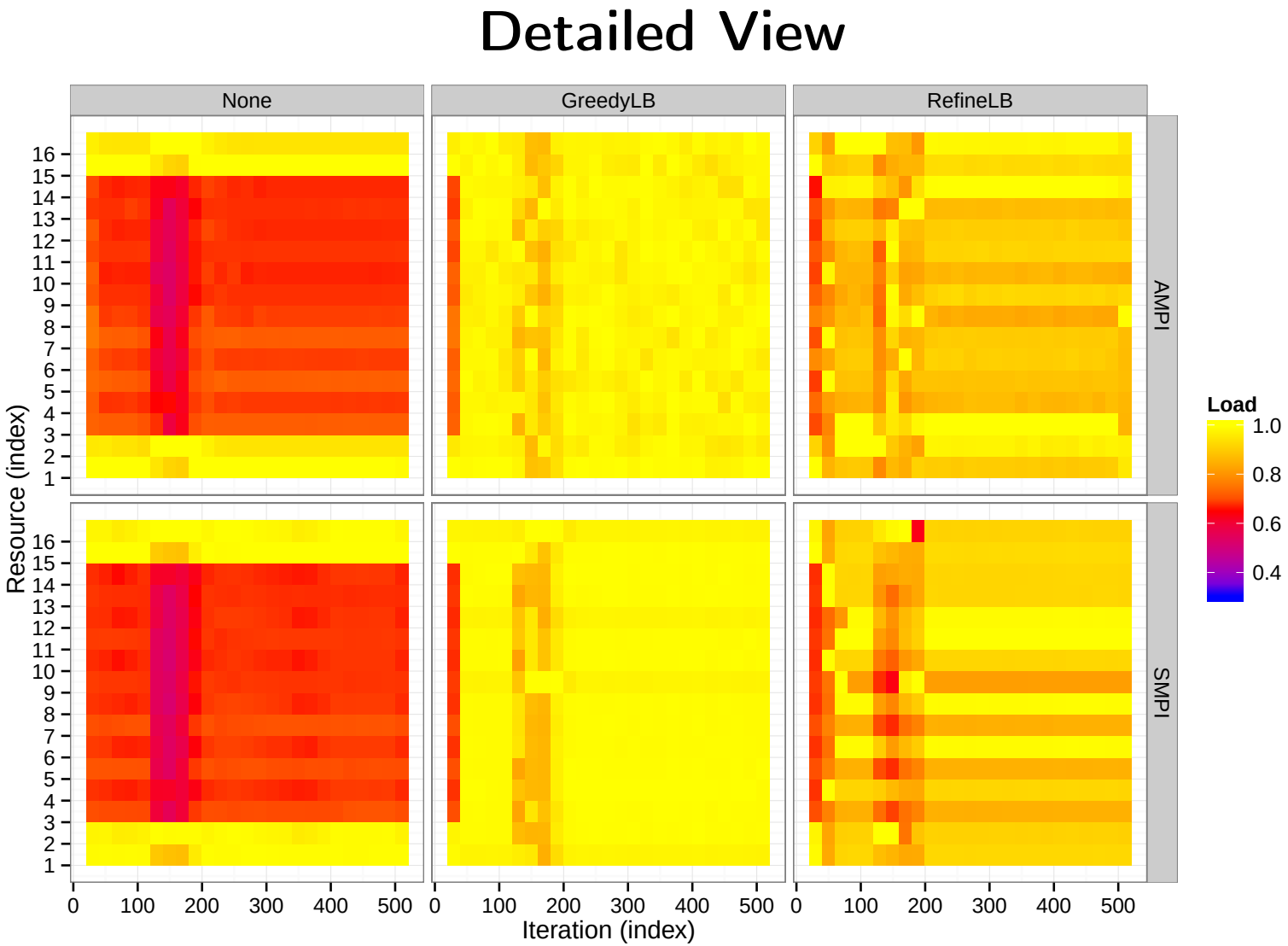
- Niigata-ken Chuetsu-Oki:
  - 2007, Mw 6.6, Japan
  - 500 time-steps; dimensions: 300x300x150
- Ligurian:
  - 1887, Mw 6.3, north-western Italy
  - 300 times-teps; dimensions: 500x350x130

Load balancers No load balancing vs. GreedyLB vs. RefineLB

Hardware Resources Parapluie cluster from Grid'5000

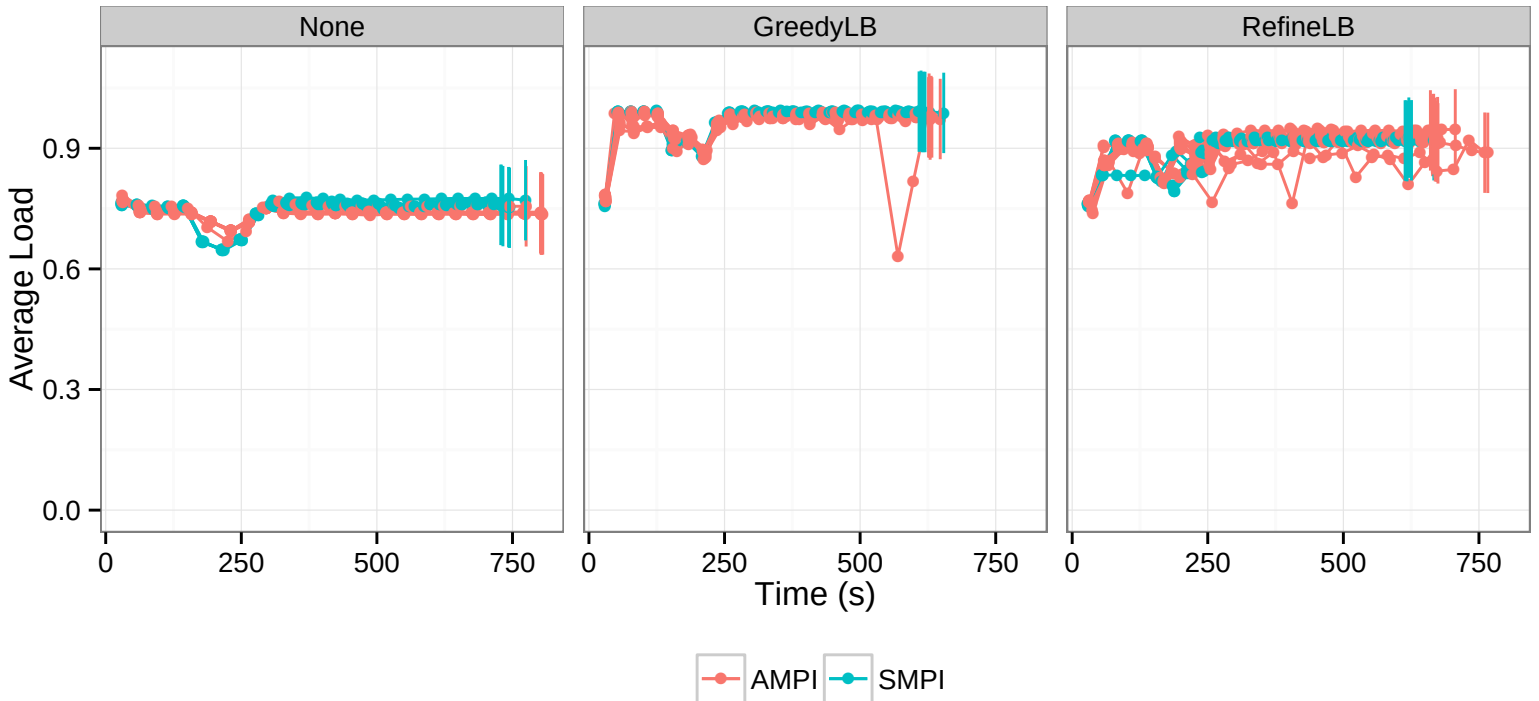
- 2 x AMD Opteron™ 6164 HE x 24 cores, 1.7GHz, Infiniband
- Plus my own laptop (Intel Core™ i7-4610M, 2 cores, 3GHz)

Chuetsu-Oki simulation - 64 VPs and 16 processes



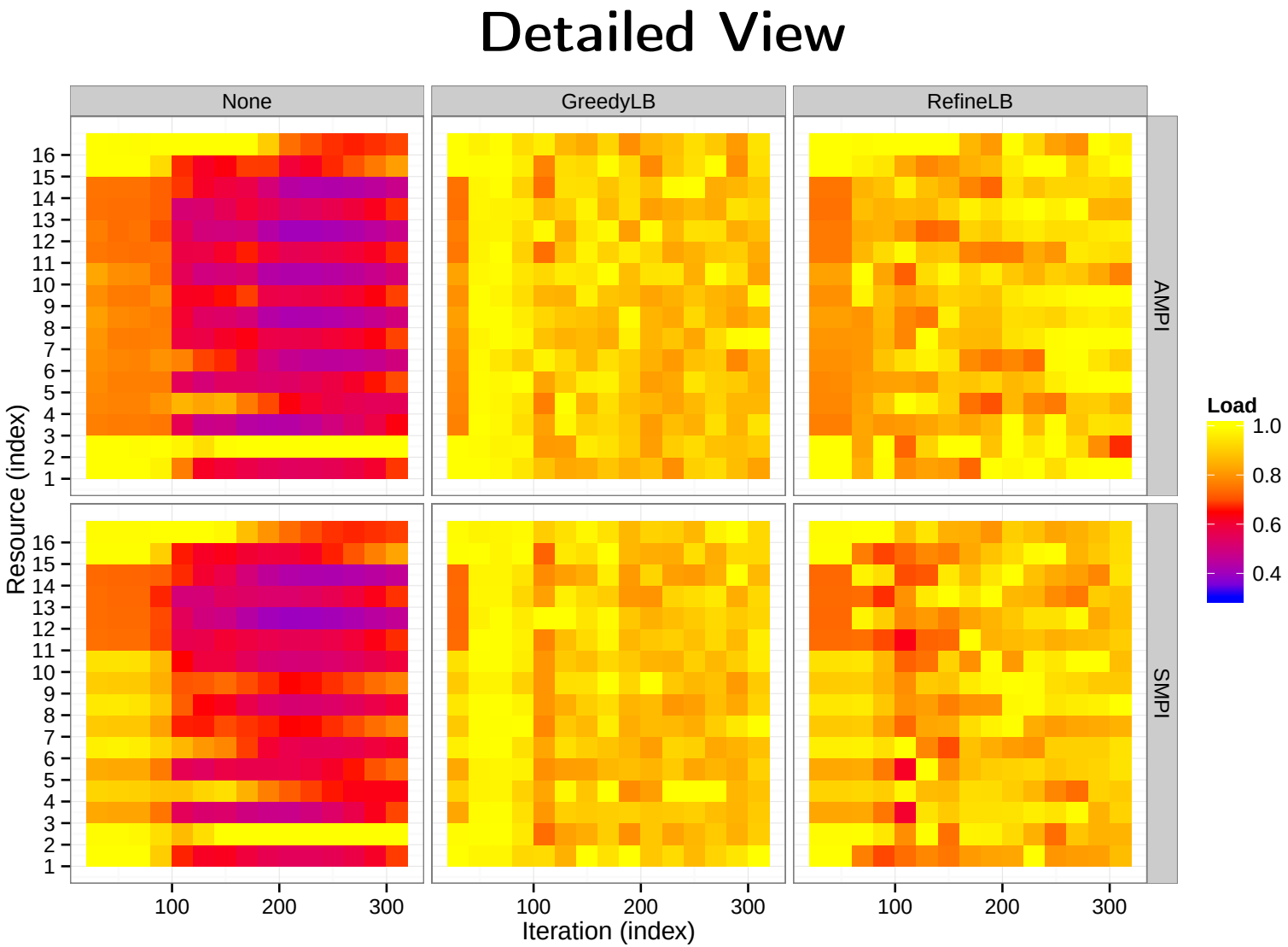
Chuetsu-Oki simulation - 64 VPs and 16 processes

Space Aggregated View 5-10 runs for each configuration

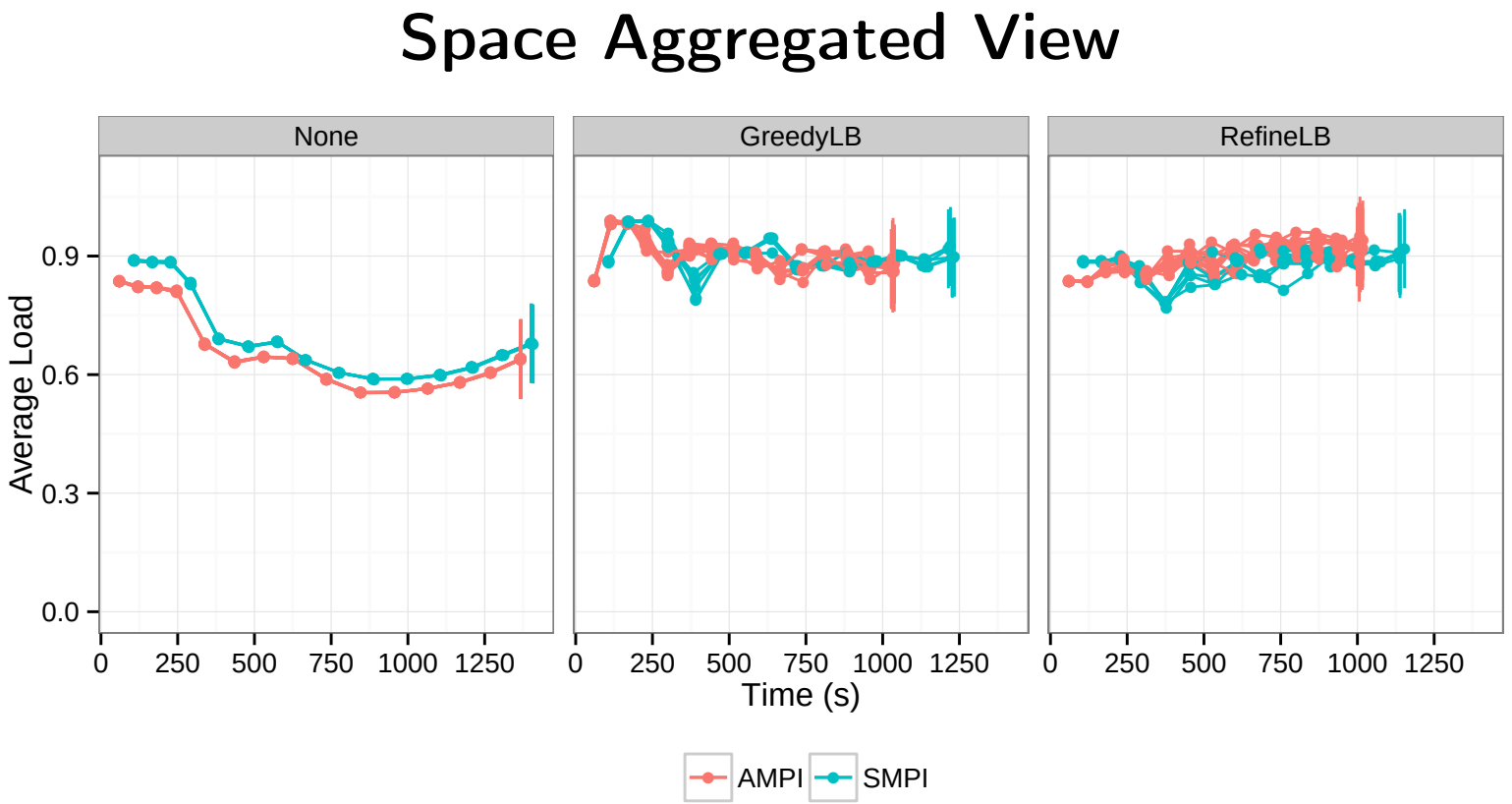


- The simulated load behaves very similar to real life
- GreedyLB is the best choice in both simulation and RL
- There is still some mismatch in terms of makespan

Ligurian simulation - 64 VPs and 16 processes



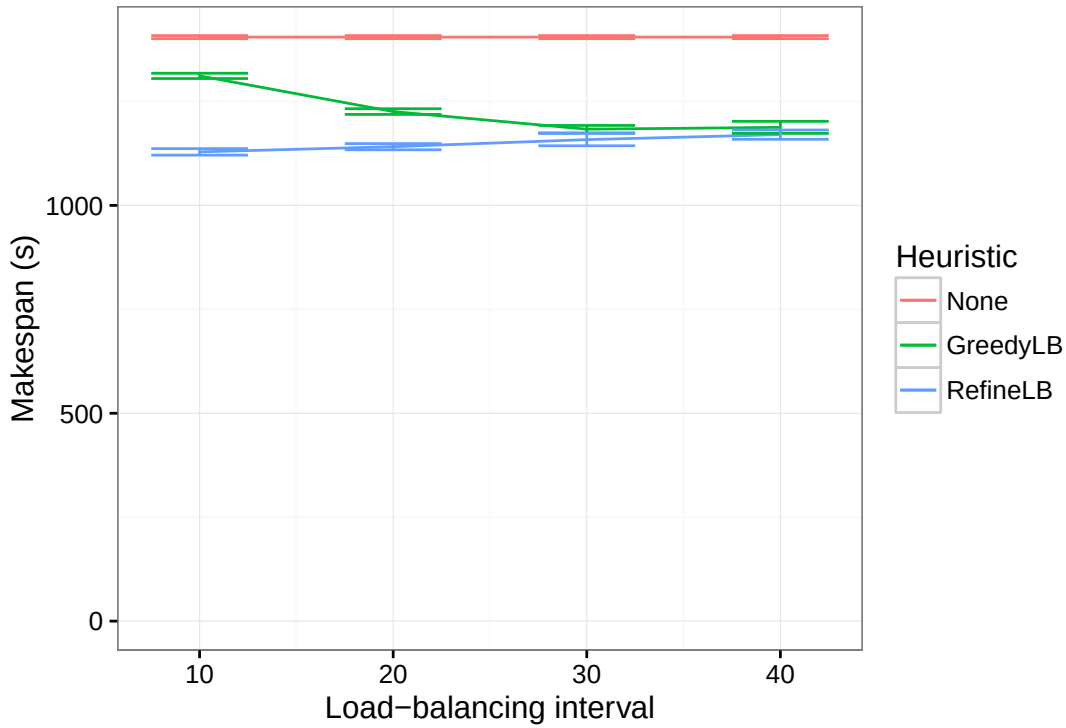
Ligurian simulation - 64 VPs and 16 processes



- Once again, the simulated an RL loads behave similarly
- RefineLB is the best choice in both simulation and RL
- Mismatch in the timings between simulation and RL

Impact of the LB frequency (simulation)

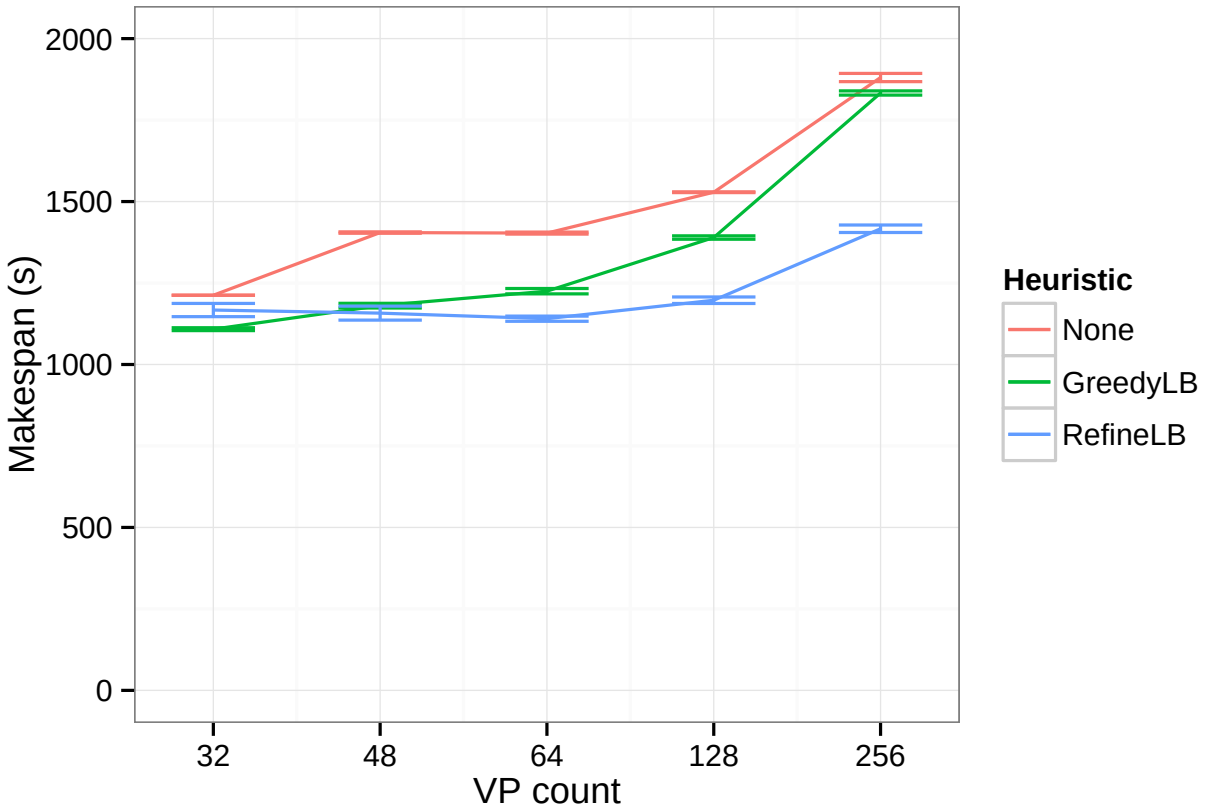
- Call `MPI_MIgrate` on every iteration
- Change the load balancing frequency in simulation



Use RefineLB every 10 or 20 iterations

- Trace capture time: $10(XP)x5$ hours $\approx$ 50 hours☹
- Simulation time: $10x3$(Heuristics)$x4$(Freq.)$x200$ sec $\approx$ 6h40m☺

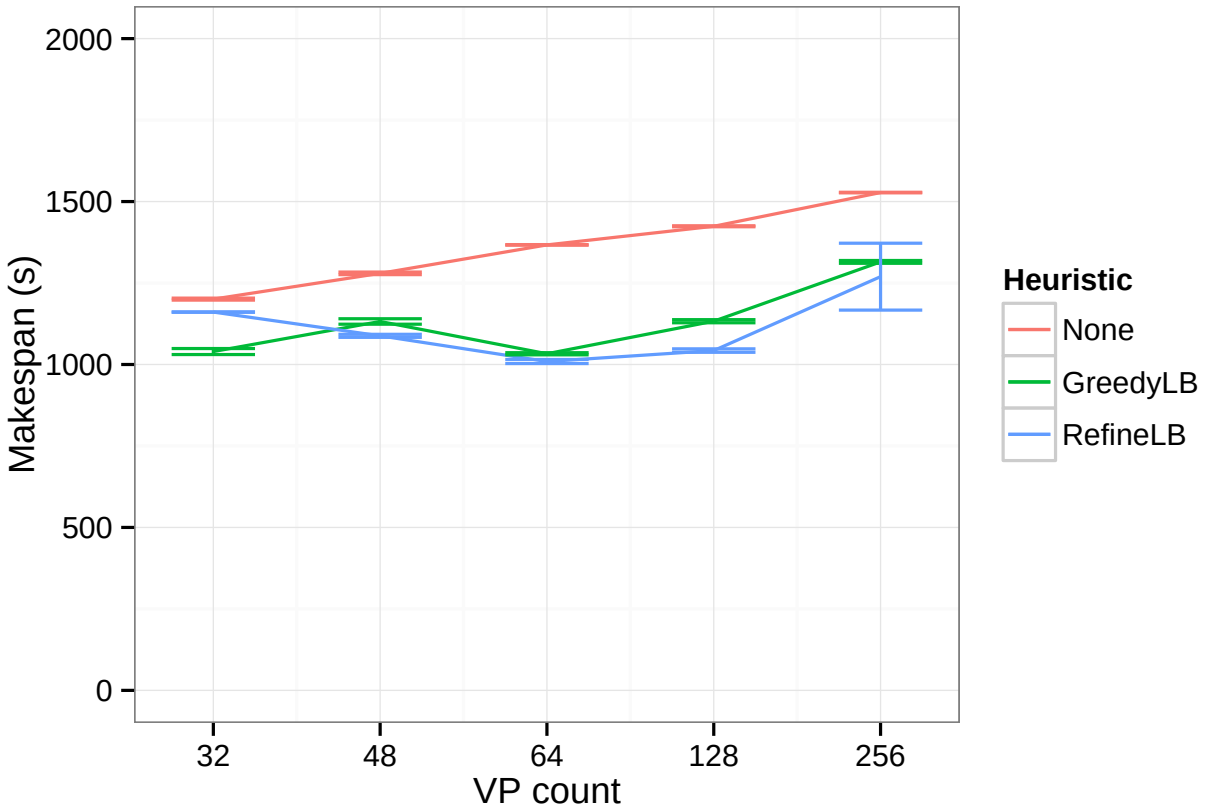
Impact of the decomposition level (in Simulation)



Use either GreedyLB with 32 VP or RefineLB with 64VP

- Trace capture time: $5(VP) \times 5(XP) \times 5$ hours ≈ 5 days ☹
- Simulation time: $5 \times 5 \times 3(\text{Heuristics}) \times 200$ sec ≈ 4.1 hours ☺

Impact of the decomposition level (real exp.)



Same conclusion... In only $\approx$ 29 hours but on a 16 node cluster.

Outline

Conclusion

- This is still ongoing work... Any comments are welcome!
- Simulation of over decomposition based dynamic load balancing
  - Good results in terms of load distribution;
  - Some inaccuracy in terms of total makespan.
- Visualize the evolution of resource usage:
  - quite useful to compare simulation with real life;
  - or to compare different load balancing heuristics.
- We need to devise some way to speed up trace collection:
  - Facilitate the analysis of different over-decomposition levels;
  - Is there some way to get similar input traces straight from Charm++/AMPI?

Acknowledgements

This research was partially supported by:

- CNPq: PhD Scholarship at the Post-Graduate Program in Computer Science (PPGC) at UFRGS
- 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 Fondation; financed by CAPES within the Ministry of Education of Brazil
- HPC4E: This research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E Project, grant agreement number 689772