SIM-SITU: A Framework for the Faithful Simulation of *in-situ* Workflows

Valentin Honoré[1], Tu Mai Anh Do[2], Loïc Pottier[2], Rafael Ferreira da Silva[3], Ewa Deelman[2], Frédéric Suter[3]

1. IN2P3 Computing Center, CNRS, Villeurbanne, France
2. USC Information Sciences Institute, Marina del Rey, CA, USA
3. Oak Ridge National Lab, Oak Ridge, TN, USA

February 23, 2022

Table of Contents

Data-intensive numerical simulation

- 2 main phases:
  - large-scale MPI simulation
  - then, post-processing of simulation data

- **BUT** I/O bottleneck

Source: https://www.top500.org/	Cray Jaguar (Nov. 2012)	Fugaku (Nov. 2021)	
R_{max} (TFlop/s)	17,590	442,010	**x26**
Memory (PB)	10	250	**x25**
Network performance (GB/s)	57.6	100	**x2**

- We can generate more and more data  but data movement is VERY costly 

Here comes the *in situ* paradigm!

- **Intertwine** simulation and analytics
- Consume partial simulation results to periodically run the data analysis
- Take some resources from simulation to give it to analytics

Here comes the *in situ* paradigm!

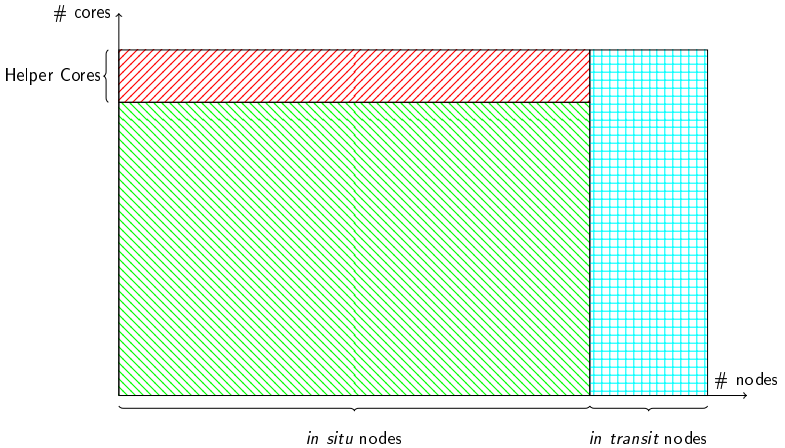


Figure: Resource allocation between *in situ* and *in transit* processing. Simulation cores are in green, *in situ* cores in red while *in transit* cores are in blue.

Here comes the *in situ* paradigm!

- **Intertwine** simulation and analytics
- Consume partial simulation results to periodically run the data analysis
- *in situ* workflows = 3 components
  - Simulation component
  - Data Analytics component (DA)
  - Data Transport Layer (DTL)
- Benefits
  1. Data locality
  2. Usage of idle resources from simulation
  3. Limitation of data movement

Motivation for designing SIM-SITU

- *in situ*: users have to decide
  1. **allocation**: how many resources to each component
  2. **mapping**: where & at which frequency to run the data analytics

- Perfomance modeling of *in situ* workflows [Aupy et al., 2019]
  1. Evaluation on synthetic applications (lack realism)
  2. Evaluation using *in situ* software
     - Need platform + software + application
     - Not designed for performance evaluation studies

- **Goal: a tool to evaluate scheduling & mapping strategies**
  - What amount of analysis can be done and at which frequency?
  - How many resources can be taken off from the execution of the simulation to execute the analysis?
  - Is it better to perform *in situ* or *in transit* analytics?

Motivation for designing SIM-SITU

- *in situ*: users have to decide
  1. **allocation**: how many resources to each component
  2. **mapping**: where & at which frequency to run the data analytics

- Perfomance modeling of *in situ* workflows [Aupy et al., 2019]
  1. Evaluation on synthetic applications (lack realism)
  2. Evaluation using *in situ* software
     - Need platform + software + application
     - Not designed for performance evaluation studies

- **Goal: a tool to evaluate scheduling & mapping strategies**

We propose a **faithful** and **scalable** simulation framework that models the behavior of all the components of *in situ* workflows in order to evaluate scheduling & mapping strategies.

Table of Contents

Sim-Situ relies on SimGrid

- SimGrid: open-source versatile framework
- Simulation of distributed applications on heterogeneous platform
- **Scalability** and **accuracy**
- Theoretically & experimentally evaluated and validated [Velho et al., 2013]

SIM-SITU relies on SimGrid

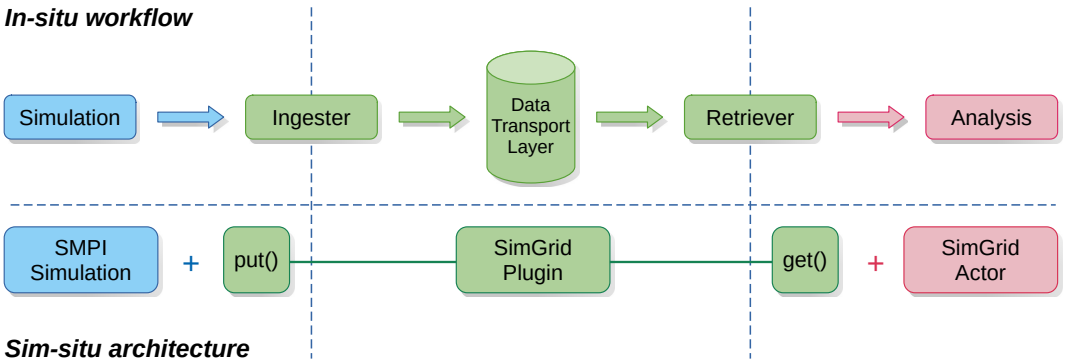


Figure: Software architectures of a generic *in situ* workflow (top) and the SIM-SITU framework (bottom).

Architecture of the 3 components

- Simulation component
  - Simulate unmodified simulation code
  - SMPI [Degomme et al., 2017] simulator: compile and run MPI programs
  - Extra command-line argument with platform description (network topology etc)
  - MPI ranks executed as threads in single process

- Analytics component
  - SimGrid actor(s) to cover wide range of data analytics components
  - Actor = simulated process. Consumes simulated resources ; performs simulated activities
  - **Strength**: out of MPI world, great flexibility

- DTL as a SimGrid plugin
  - Producer-Consumer synchronization mechanism (put/get)
  - Offers two different internal implementations of the message queue
    1. "queue": configurable size, instantaneous data exchanges & respect the flow dependencies
    2. **"mailbox"**: rendez-vous point between a sender and a receiver

SIM-SITU: for and cons

- Many advantages
  - Faithful simulation engine and flexible component design
  - Evaluation of wide range of allocation and mapping strategies
  - Not (too much) code-intrusive
  - Run on a single core!
- ... and some limitations!
  - Need some benchmarking of analytics component (sometimes hard)
  - Does not implement (yet) complex DTL (DataSpaces, Dimes) and DA
  - Hard to model platform interference

Molecular Dynamics (MD) *in situ* workflow

- ExaMiniMD proxy-application [Thompson and Trott, 2017, ExaMiniMD, 2021] *
- Representative application, compact & maleable code
- All-atom MD simulations: floating-point intensive pairwise atom-atom unbounded interactions
  - Simulation: domain decomposition + point-to-point MPI communications (asynchronous receives)
  - Analytics: computes temperature, potential & kinetic energy of the system

*Part of the Exascale Computing Project [Aaziz et al., 2019]

Experimental platform

- *dahu* cluster of the Grid'5000 experimental testbed.
- 32 nodes with x2 Intel Xeon Gold 6130 CPUs 16 cores, 192 GiB of memory.
- 10 Gb/s Ethernet network
- We use thorough platform calibration from [Cornebize, 2021] for SMPI model

Simulation component: characterization

- Simulate unmodified code (compilation with *smpicxx*)
- Usage of **sampling** for faster simulation
  - Replace a time-consuming loop/kernel by a delay
  - 150 samples with a standard deviation threshold of 0.002 in our experiments
- Two different types of sampling
  1. "Local": each rank determines its delay
  2. "Global": delay determined from samples executed by all MPI ranks
- ExaMiniMD setup: stride (DA frequency) = 50 iterations, 8000 iterations in total

Simulation component: characterization

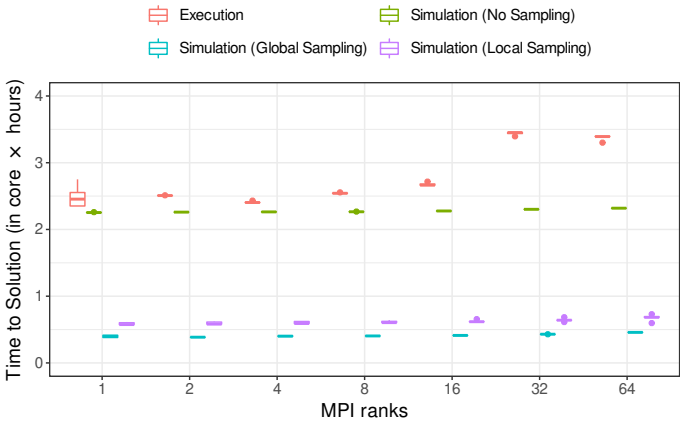


Figure: Time to solution for running or simulating (with or without kernel sampling) the execution of a 3D Lennard-Jones melt on a 70^3 region with ExaMiniMD.

Simulation component: accuracy

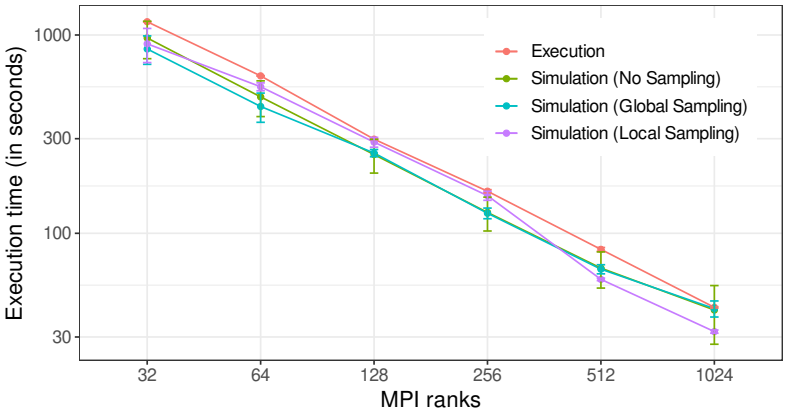


Figure: Accuracy of the simulation of 3D Lennard-Jones melt on a 70^3 region (with or without kernel sampling) when varying the number of MPI ranks.

Data Transport Layer

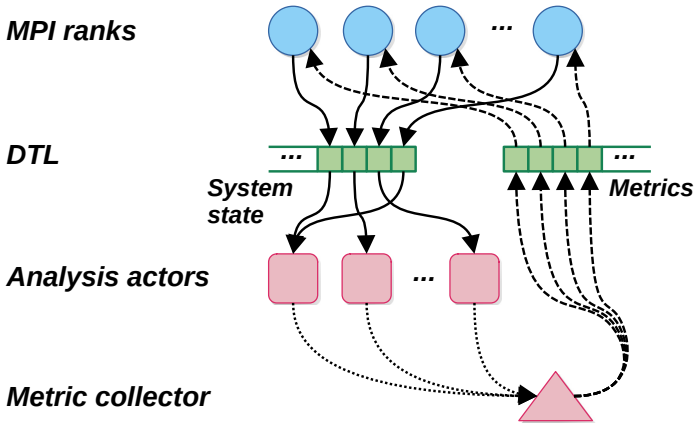


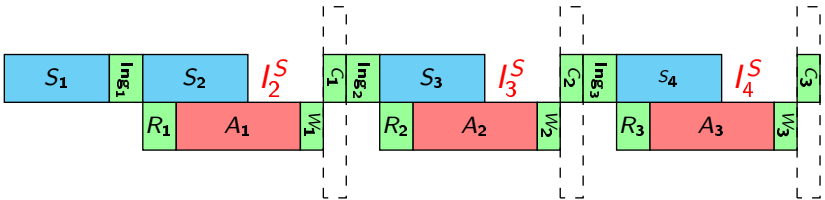
Figure: Data exchanges between the Simulation and Analytics components through the Data Transport Layer.

Analytics component

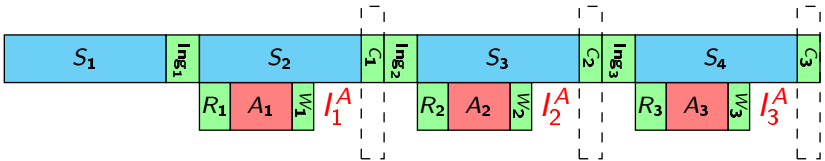
- Most versatile component of *in situ* workflows
- Configure the *in-situ* version of ExaMiniMD without having to recompile the application
- **Our choice**: extra command-line flag `--analysis`
- Six parameters for ExaMiniMD
  1. the **number of analytics actors** to spawn;
  2. a **hostfile** for the mapping of actors
  3. the **cost per particle of the analytics**
  4. a **computing scaling factor**
  5. the **size per particle** of the data transferred from the simulation to analytics component.
  6. a **data transfer scaling factor**

Table of Contents

Workflows with constraints



(a) Idle Simulation (IS) Scenario: $R_i + A_i > S_i + \text{Ing}_i$



(b) Idle Analytics (IA) scenario: $S_i + \text{Ing}_i > R_i + A_i$

Figure: Two execution scenarios for the ExaMiniMD *in-situ* workflow.

Efficiency of *in situ* workflows

- Total idle time I_*

$$I_* = I_*^S + I_*^A = \begin{cases} S_i + \text{lng}_i - (R_i + A_i) & \text{if } I_*^S = 0 \\ R_i + A_i - (S_i + \text{lng}_i) & \text{if } I_*^A = 0 \end{cases} = |S_i + \text{lng}_i - (R_i + A_i)|$$

- Makespan: sum of the makespan of each step $i \leq \rho$:

$$m = \sum_{i=1}^{\rho} m_i,$$

- η: efficiency

$$\eta = 1 - \frac{\rho \times I_*}{m} = 1 - \frac{|S_i + \text{lng}_i - (R_i + A_i)|}{\max(S_* + \text{lng}_*, R_* + A_*)}$$

Experimental setup

- $70 \times 70 \times 70$ region problem instance
- 8,000 iterations, *dahu* cluster
- DA component instantiation after benchmarkings
  1. compute cost per particle = 7.93*e*-7
  2. data size per particle = 100 Bytes
  3. basically, scaling factors of 1
- Allocation strategies from [Malakar et al., 2018]

Table: Considered simulation to analysis core allocation ratios.

R	# simulation cores	# analysis cores
1	16	16
3	24	8
7	28	4
15	30	2
31	31	1

Exp.#1 - Impact of Simulation to Analysis Core Allocation Ratio

- **Scenario:**
  - *in situ* processing (no *in transit*)
  - Simulation and DA = ExaMiniMD
- **Question:** For a given number of cores and application input, best core allocation ratio?
- User can act on two parameters
  - *stride* (frequency of data analytics)
  - *cost* of one analytics execution
- Example: 8000 iterations, 400 units of analytics
  - (stride, cost) = (20, 1), (200, 10), (500, 25), and (1000, 50)
- SIM-SITU: just play with the computing scaling factor parameter!

Exp.#1 - Impact of Simulation to Analysis Core Allocation Ratio

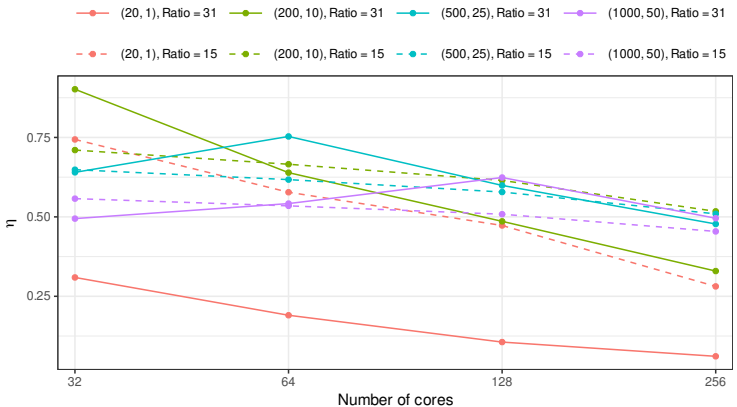


Figure: Efficiency of ExaMiniMD *in-situ* workflow in four (stride, analytics cost) configurations for two of the core allocation ratios.

Exp.#1 - Impact of Simulation to Analysis Core Allocation Ratio

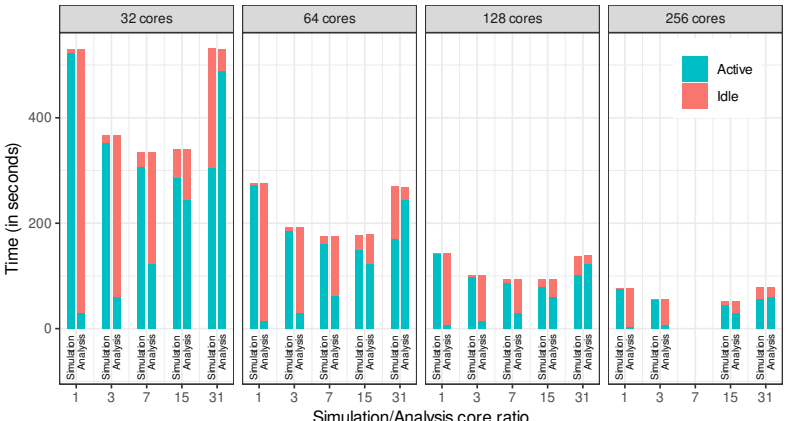


Figure: Evolution of the active and idle times of the simulation and analytics components when increasing the core allocation ratio and the total number of cores for the (1000, 50) scenario.

Exp.#2 - Impact of simulation to analysis data transfers

- **New scenario:**
  - Simulation = ExaMiniMD as before
  - New DA component = execution time increases with the resource scattering[†]
- 16 nodes: *in transit* (1 node, 32 cores) and *in situ* ($R = 15$) mapping strategies
- Volume of transferred data determined by user (variation from factor 1 to 1000)
- **Question:** Is it better to adopt an *in-situ* or an *in transit* strategy?
- 'Tipping point' of the maximal *in transit* data transfer

[†]See bonus slide 1 for scattering function

Exp.#2 - Impact of simulation to analysis data transfers

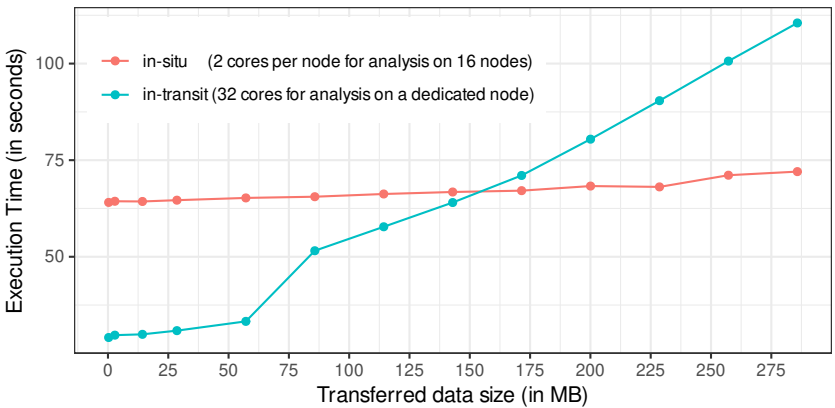


Figure: Evolution of the execution time of the application when the volume of data transfer is scaled up in the *in-situ* and *in transit* execution modes with 16 nodes and $R = 15$.

Table of Contents

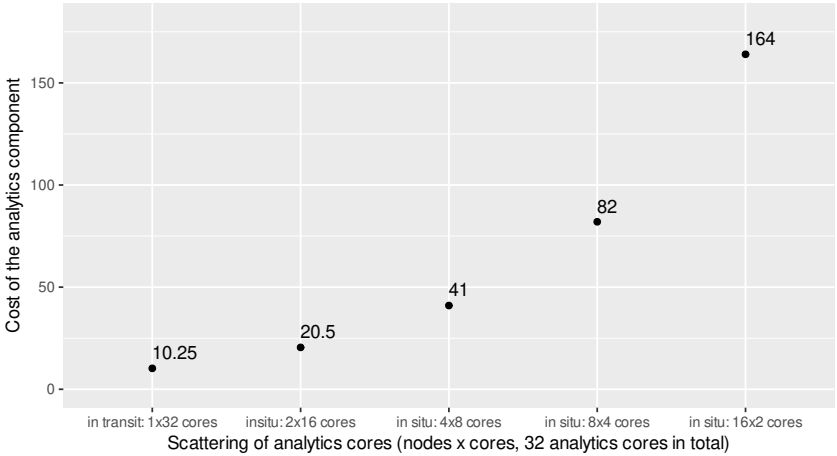
Perspectives & Future Work

- Evaluation tool for the performance of *in situ* workflow
- Generic simulation framework based on SimGrid toolkit
- Demonstration on MD case, helps finding tradeoffs and tipping points
- Companion report for complete reproducibility
- Future work
  - Investigate more allocation and mapping strategies (impact of network etc)
  - Extend capacities and realism of SIM-SITU(more complex DTL, parralel DA)
  - Use SIM-SITU for experiments difficult to do on supercomputers (online scheduling decisions for adaptive sampling process)

Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr)

Execution time increases with resource scattering



Analytics actor pseudo-code

Algorithm 1 Analytics actor

```
1: loop
2:     Get system state from the DTL
3:     if Poisoned value then
4:         if Last actor running then
5:             Send poisoned value to metric collector
6:         end if
7:         Return
8:     end if
9:     Compute analytics
10:    Send computed metrics to metric collector
11: end loop
```

Metric collector pseudo-code

Algorithm 2 Metric collector actor

```
1:  loop
2:      n_collected_values = 0
3:      repeat
4:          Get metrics from analytics actors
5:          if Poisoned value then
6:              Return
7:          end if
8:          Accumulate metrics
9:      until n_collected_values = n_ranks
10:     for all n_ranks do
11:         Put a copy of accumulated metrics into the DTL
12:     end for
13: end loop
```

Simulation process with SimGrid

- Code of the MPI program is executed as it is...
- but MPI ranks executed as threads in a single process (same address space)

- Each MPI call:
  - control is handed off to SMPI
  - network operations replaced by delays
  - delays computed by simulation models of SimGrid

- Code in between two MPI calls is benchmarked on the machine used to execute the simulation.

- SMPI simulates both communication and computation operations as delays.

References I


Aaziz, O., Vaughan, C., Cook, J., Cook, J., Kuehn, J., and Richards, D. (2019).
Fine-Grained Analysis of Communication Similarity between Real and Proxy Applications.
In *Proc. of the 10th IEEE International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, Denver, CO.


Aupy, G., Goglin, B., Honoré, V., and Raffin, B. (2019).
Modeling High-Throughput Applications for In Situ Analytics.
International Journal of High Performance Computing Applications, 33(6):1185–1200.


Cornebize, T. (2021).
High Performance Computing: towards better Performance Predictions and Experiments.
PhD thesis, Université Grenoble-Alpes, Grenoble, France.


Degomme, A., Legrand, A., Markomanolis, G., Quinson, M., Stillwell, M., and Suter, F. (2017).
Simulating MPI applications: the SMPI approach.
IEEE Transactions on Parallel and Distributed Systems, 18(8):2387–2400.

References II


ExaMiniMD (2021).
ExaMiniMD Proxy Application GitHub Repository.
[Online] https://github.com/ECP-copa/ExaMiniMD.


Malakar, P., Munson, T., Knight, C., Vishwanath, V., and Papka, M. E. (2018).
Topology-Aware Space-Shared Co-Analysis of Large-Scale Molecular Dynamics Simulations.
In *Proc. of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, Dallas, TX.


Thompson, A. and Trott, C. (2017).
A Brief Description of the Kokkos implementation of the SNAP potential in ExaMiniMD.
Technical Report 1409290, Office of Scientific and Technical Information.


Velho, P., Schnorr, L. M., Casanova, H., and Legrand, A. (2013).
On the Validity of Flow-Level Tcp Network Models for Grid and Cloud Simulations.
ACM TOMACS, 23(4).