Influence of Computational Parameters and Hardware Configurations on Energy Consumption in OpenRadioss Simulations

Denergium, More Flops per Watt!

2025-06-04

Abstract

This study investigates the impact of computational parameters (number of MPI processes, cores per process, CPU frequency, power capping) and hardware configurations on the energy efficiency of **OpenRadioss**, an open-source finite element solver for structural mechanics and crash simulations. Using **EnergyScopium**, we measured CPU energy consumption, efficiency metrics, and runtime during a vehicle crash simulation (Neon 1M finite elements) across two HPC clusters: **Curta** (MPI/Cores parameter tuning) and **Grid5000** (CPU frequency scaling and power capping experiments).

Introduction

Finite element analysis (FEA) is a cornerstone of engineering simulations, enabling predictive modeling of structural integrity, crash dynamics, and material behavior. However, large-scale simulations in tools like **OpenRadioss** demand significant computational resources, leading to high energy consumption. As global HPC workloads grow, optimizing energy efficiency without compromising simulation fidelity becomes critical for both cost and environmental sustainability.

This study focuses on **OpenRadioss** (https://www.openradioss.com), an open-source descendant of the Radioss solver, widely used for crash and impact simulations. By analyzing the influence of computational parameters (MPI process count, thread distribution, CPU frequency scaling and power capping) and hardware configurations, we aim to:
1. Quantify energy consumption trade-offs under varying parameter combinations.
2. Identify efficient configurations balancing runtime and energy usage.
3. Provide actionable insights for energy-aware HPC scheduling.

Using **EnergyScopium**, we extend traditional performance metrics to include energy efficiency, enabling holistic optimization of simulation workflows.

Experimental Setup

In this section, we present the computational resources used for our experiments. Our setup includes a diverse set of high-performance computing (HPC) clusters, each providing different hardware configurations and capabilities suited for our workload.

We had access to Curta, a computing cluster attached to the Mésocentre de Calcul Intensif Aquitain (MCIA), offering a local infrastructure for initial testing and benchmarking. Additionally, we leveraged Grid’5000, a large-scale and flexible testbed for research in distributed and parallel computing.

The combination of these resources allowed us to conduct a comprehensive evaluation, benefiting from different architectures, scheduling policies, and execution environments. The following subsections provide further details on the specific hardware configurations of each machine.

Hardware and Software Configurations

Curta

Hardware configuration

Component	Specifications
Processors	2x Intel Xeon Gold SKL-6130 (16 cores each)
RAM	92 GB

Software

Component	Version
OS	RockyLinux 8.60
OpenRadioss	v20250422
MPI	OpenMPI 4.10.2
Compiler	GCC 11.20.0

Grid5000 Cluster (France)

Hardware

Component	Specifications
CPUs	2x Intel Xeon Gold 5320 (26 cores each)
RAM	384 GB

Software

Component	Version
OS	Linux Debian 5.10.234-1
OpenRadioss	v20250422
MPI	OpenMPI 4.10.5
Compiler	GCC 11.30

Application: OpenRadioss Simulation Parameters

OpenRadioss is the open-source version of Altair’s industry-proven Radioss explicit dynamics solver. It inherits all core capabilities of Radioss, including high-fidelity crash-test, impact and explosion simulations with support for complex contact and nonlinear material ehavior, while offering full access to its source code for itnegration, extension, and community-driven development. Input decks in both the native Radioss “Block” format and the LS-DYNA keyword format are supported, enabling seamless use of existing preprocessing tools like Gmsh ans HyperMesh, and results can be post-processed with ParaView or other VTK-capable platforms. Backed by Altair’s commitment to fostering an active global user community, OpenRadioss combines enterprise-grade performance with the openness and adaptability demanded by modern scientific research and engineering workflows.

Simulation Model

OpenRadioss provides a ready-to-run Chrysler Neon benchmark model, a full-body car mesh of around 1M elements, that lets you perform high-fidelity frontal crash

simulations on commodity hardware. This Neon setup exercises the solver’s contact, large-deformation and material nonlinearity capabilities, reproducing realistic crash-test scenarios without needing proprietary LS-DYNA decks. Thanks to OpenRadioss’ explicit dynamics core, you can launch a Neon car crash job on a single workstation or scale it to HPC cluster, then visualize detailed strain, stress and energy fields in ParaView. The Neon benchmark thus serves both as a performance test (up to hundreds of thousands of elements per second) and as a validation case for automotive CAE workflows.

Parameter Space

Parameter	Tested Values	Cluster
Cores/Processes ratio	[2/16, 4/8, 8/4, 16/2, 32/1]	Curta
CPU Frequency Scaling	[100%, 85%, 70%, 50%, 25%]	Grid5000
Power Capping	[100%, 85%, 70%, 50%, 25%]	Grid5000

Energy Consumption Measurement

Denergium (www.denergium.com) is an Inria startup, founded in 2023. Denergium improves the efficiency of supercomputing. Our customers are major players in industry, academia, government agencies, BFSI and Pharma, for whom simulation and AI are major challenges. Thanks to our innovation, based on an understanding of the energy involved in computations, we can provide software solutions and expertise to improve energy management and, ultimately, increase the number of computations.

EnergyScopium is a software suite developed by Denergium. EnergyScopium is available in 2 versions: EnergyScopium Smart Decision and EnergyScopium profiler.

EnergyScopium Smart Decision gives HPC infrastructure managers qualitative feedback on the energy efficiency of all jobs and servers. As a result, they can improve infrastructure efficiency, and ultimately perform more computations.

EnergyScopium also provides users and developers with different levels of information:

- an efficiency label
- energy profile data
- insights into efficiency quality

EnergyScopium profiler is dedicated to users and developers who want to compare, improve and measure their applications from an energy angle. We use EnergyScopium profiler to collect the data in this study.

EnergyScopium provides information on the entire energy chain

Energyscopium precisely measures the energy consumed by each physical unit and calculates the efficiency—such as the ratio of CPU energy usage to its Thermal Design Power (TDP). These raw measurements are then refined through estimations that integrate the overall node consumption. The PUE (Power Usage Effectiveness) is then taken into account to calculate consumption as close as possible to reality. Thanks to our knowledge of the energy mix of the place where the calculation was made, we can propose a consumption equivalent in gCO2 for the calculation.

Experimental Protocol

In our experimental protocol, we first explored the parallel execution configuration by varying the number of MPI processes and, for each MPI rank, the number of OpenMP threads. We conducted this study both on a single node, adjusting the balance between inter-process communication and intra-process threading, and across multiple nodes to asses how scaling affects performance and energy efficiency. Once the optimal MPI +

OpenMP combinations were identified, we focused on single-node runs to investigate hardware-level power management: we swept through different CPU clock frequency settings and applied varying power caps via RAPL to abserve their impact on runtime, power draw, and energy consumption. Together, these experiments allow us to disentangle the effects of software-level parallelism and hardware-lvel power controls on overall application behavior.

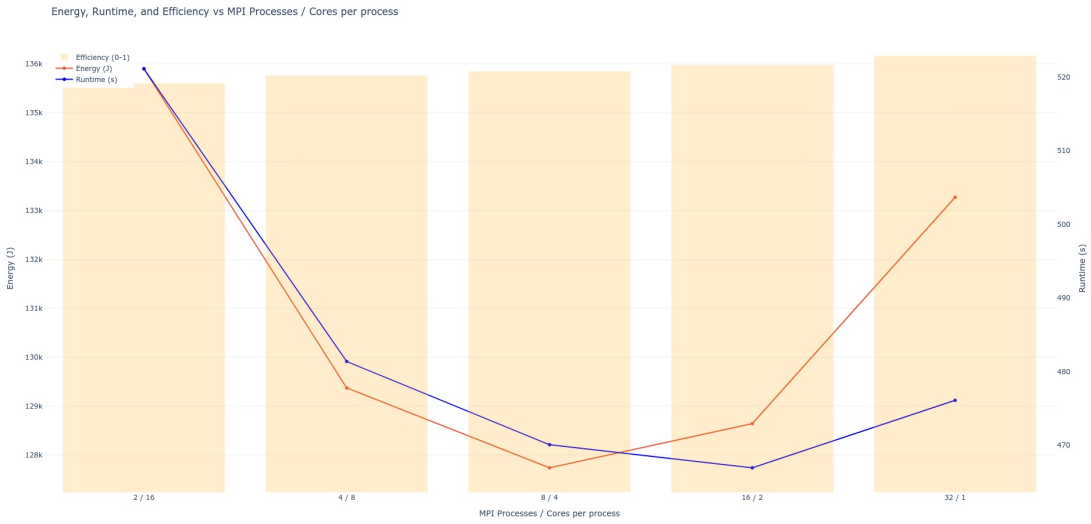
Cluster	Core/Processes	Power Capping	Frequency	Total Configurations
Curta	5	-	-	5
Grid5000	-	5	5	10

Number of MPI processes and OpenMP threads are inversely linked. For each configuration, 5 simulation runs will be executed to ensure statistical robustness.

Results

This section presents the experimental results obtained frome the three different computing platforms, each focusing on a specific aspect of our study. We begin the analysis conducted on Curta, which investigates performance and energy efficiency in a single-node environment. The second subsection explores the effects of frequency scaling and powercapping using the Grid5000 infrastructure. Each study highlights trade-offs between runtime, energy consumption and overall efficiency under varying execution conditions.

Curta



This plot shows how energy, runtime and energy efficiency of the hardware evolve with varying MPI process configurations. As the number of processes increases (and core per process decreases), both energy consumption and runtime drop initially, reaching optimal values around 8 processes with 4 cores each. Beyond this point, both metrics rise again, most likely due to overhead and contention.

Efficiency, represented by the shaded bars, keep growing to peak with the 32/1 configuration. As a function of energy, runtime and power, this behavior is linked to the lines as the energy decreases slower than the runtime and inversely increases faster.

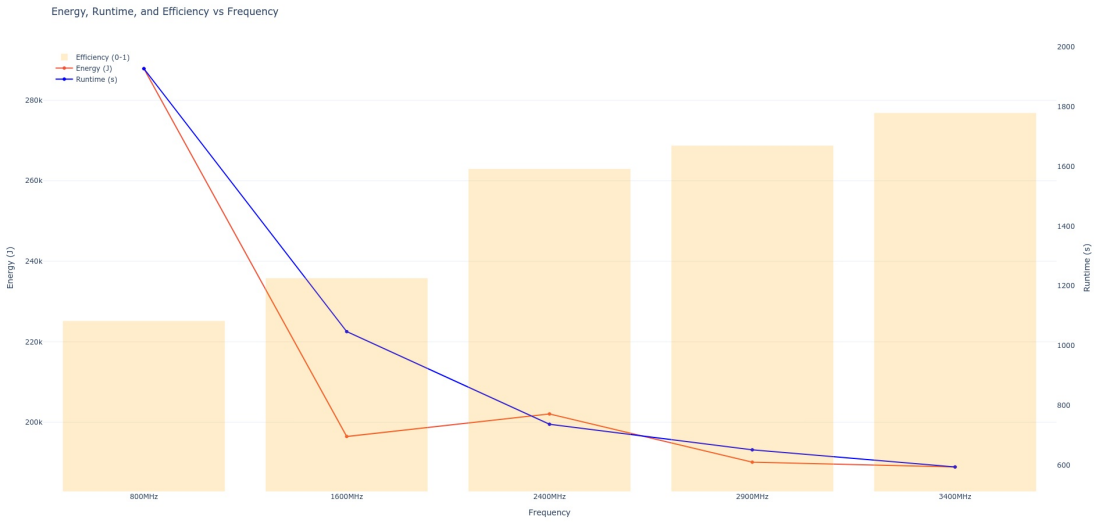
The 8/4 or 16/2 setups offer the best balance between performance and energy, highlighting the tradeoff between underutilization of the machine and parallel

overhead.

Grid5000

The notion of efficiency in this subsection should be interpreted cautiously. Since the the clock frequencies were manually capped, we are not operating under natural scaling conditions. As a result, the computed efficiency values do not fully reflect the system’s true performance potential or energy behavior under typical usage. The nominal case here with 3400MHz clock frequency and 185W powercap achieves jobs with an average runtime of 593 seconds and consumes 184kJ.

Frequency Scaling



This plot illustrates how frequency scaling affects energy consumption, runtime and CPU energy efficiency. At the lowest frequency (800MHz, ~25% of the CPU capacity), both energy usage and runtime are high, with poor efficiency. As frequency increases to 1600MHz (~50%), energy consumtpion drops significantly, reaching its minimum, while runtime also improved.

Beyond 1600MHz, runtime continues to decrease, but energy make slight rebound at 2400MHz (~70%) before decreasing again, to reach the lowest value for the maximal setup, 3400MHz (100%). Efficiency increases continuously from the minimal setup to the maximal.

No significant tradeoff hilighted here, the most energy efficient come with the lowest energy consumption and execution time at full capacity of the machine.

Power Capping

This ploy shows the impact of power cap on energy consumption, runtime and CPU energy efficiency. At the lowest cap (46W, ~25% of the CPU capacity), both energy and runtime are extremly high, leading to poor efficiency. Increasing the cap to 93W (~50%) shrply reduces both energy and runtime.

Energy consumption reach its lowest point at 130W (~70%), with runtime keeping to slightly decrease. Beyond this mark, runtime looks almost flat with unsignificant gains while energy consumption starts to increase. Efficiency almost double between 800MHz and 1600MHz, then increases to more than 80% reaching a plateau.

The 130W is the optimal cap, delivering fast execution with minimal energy consumption.

Discussion and Conclusion

Key Findings

- **MPI/Thread Balance**: A possible tradeoff between energy and runtime.
- **Frequency Scaling**: No favorable tradeoff possible using frequency scaling.
- **Power Capping**: A sweep spot setup with around 10% energy save and similar runtime.

Future Work

- Conduct more single-node experiments with a focus on initial overhead and runtime consistency.
- Explore finer granularity in power cap adjustments to refine the energy-performance balance.
- Extend evaluation to multi-node configurations to assess scalability.

Acknowledgments

This work was supported by Denergium and Altair.

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.