Spacehog | Evaluating the costs of dedicating resources to *in situ* analysis

Sandia National Laboratories
Exceptional service in the national interest

Rebecca Kreitinger
University of New Mexico

Scott Levy, Kurt B. Ferreira, Patrick Widener
Center for Computing Research, Sandia National Laboratories

INTRODUCTION

Scientific simulation codes generate enormous quantities of output data. Making sense of these enormous corpora of data requires domain scientists to use sophisticated analysis tools. However, the cost (in both time and energy) of exchanging data between the simulation and the analysis codes through a global filesystem is becoming prohibitive. As a result, workflows that use in situ analytics to process output data are becoming more common.

- In situ analytics require that resources be shared between the simulation and the analysis
- Two common types of sharing resources:
  - Time-sharing
    - Simulation and analysis run on the same set of resources
    - Each is granted intervals of exclusive access to the computational resources
  - Space-sharing
    - Divides the computational resources between the simulation and analysis
    - Both are given exclusive use of a fraction of the total resources
- Space-sharing can introduce the possibility for contention over these shared resources:
  - Memory (cache and DRAM)
  - Memory bandwidth
  - Network bandwidth
  - Filesystem bandwidth
- To evaluate application sensitivity to the use of shared resources, we used a set of microbenchmarks
- We examine how the application's performance is affected by space-sharing with each member of a set of analysis proxies-computational kernels that represent resource usage patterns that are representative of analytics that may be used with scientific simulation.

The contributions of this poster include:

- The description of a software library that facilitates the evaluation of the performance impact of space-sharing MPI applications with analysis tasks.
- An examination of how space-sharing MPI applications with different computational workloads may impact application time-to-solution.
- An analysis of how the characteristics of the execution environment affect the impact of space-shared in situ analytics on application performance

APPROACH

- The analytics tasks used in this study are modeled using a library called libspacehog.
- This library uses the MPI profiling interface to intercept all MPI calls transparently.
- This library sets aside the requested number of dedicated MPI processes for analytics proxies and returns the remainder of the compute resources for the application.
- The user can request a number of options for this library. These includes: 1.) the layout of the dedicated resources for the analytics proxy; 2.) the analytics proxy (or *hog task*) listed in table 1P; and 3.) parameters for each of these hogs, for example the volume data copied or processed.

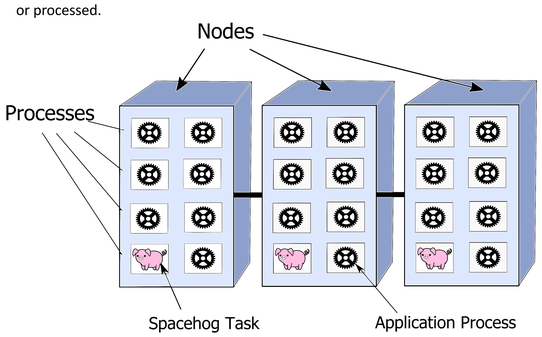


Figure 4: Visual representation of the libspacehog library's functionality.

EXPERIMENTAL METHODOLOGY

- We ran three sets of experiments, each set examined a different dimension of how simulation workloads might be space-shared with analysis codes.
- The results of these experiments are shown in Figures 1-3.
- We repeated these experiments for each of the five workloads described in Table 2P
- For all of our experiments, we ran these workloads on 84 application processes.
- We conducted our experiments on the two HPC systems, Volta and Chama, described in Table 3P
- Except where otherwise specified, the results presented were collected on Volta.
- Data collected for each experiment over five independent trials: height of bars = mean, whiskers = min, max.
- All of the results are normalized to the application time-to-solution when space-shared with the busy wait analytics task.

RESULTS

Computational Kernels: Microbenchmarks > Analysis Proxies [all figures]

- Microbenchmark tasks (e.g,. `allreduce`, `bursty copy`, `copy`) have the greatest impact on application performance
- Analysis proxy tasks (e.g. `FFT`, `pi`, `k-means`, `voronoi`) have a relatively minor impact on application performance

Process Density: more processes per node may degrade (HPCG) or improve (miniAMR) performance [Figures 1.1-1.5]

- Fewer processes per node + memory-sensitive application (e.g., HPCG) ⇒ better performance
- Fewer total nodes + space-sharing communication-sensitive applications (e.g., miniAMR) & communication-intensive benchmark (e.g. `allreduce`) ⇒ better performance

Applications: miniAMR and HPCG most impacted [all figures]

- miniAMR and HPCG experience the largest degradation in performance
- LAMMPS-lj, LAMMPS-crack, and miniFE experience relatively minor degradation in performance

Number of Spacehog Tasks: fewer spacehogs ⇒ better performance [Figures 2.1-2.5]

- Fewer spacehog processes almost universally improves performance

System Architecture: Volta ⇒ better network performance, Chama ⇒ better memory performance [Figures 3.1-3.5]

- Space-sharing memory-sensitive applications (e.g., HPCG) & memory-intensive benchmark (e.g., `copy`, `bursty copy`) ⇒ better performance on Chama
- Space-sharing communication-sensitive applications (e.g., miniAMR) & communication-intensive benchmark (e.g., `allreduce`) ⇒ better perform on Volta

Process Density

Experiments performed by varying the total number of processes per node

Figure 1.1

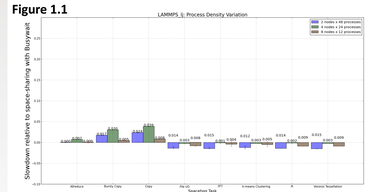


Figure 1.2

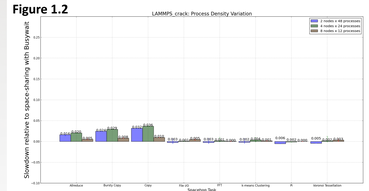


Figure 1.3

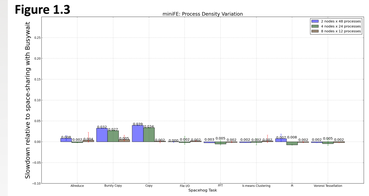


Figure 1.4

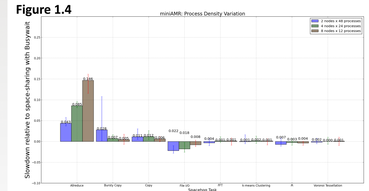
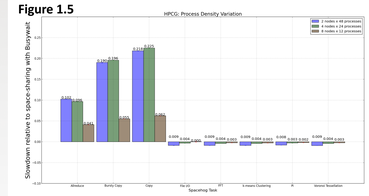


Figure 1.5



Number of Spacehog Tasks

Experiments performed by varying the total number of spacehog tasks

Figure 2.1

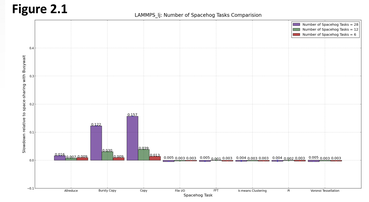


Figure 2.2

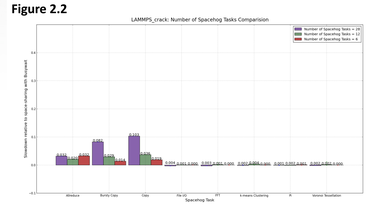


Figure 2.3

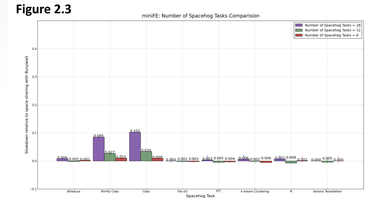


Figure 2.4

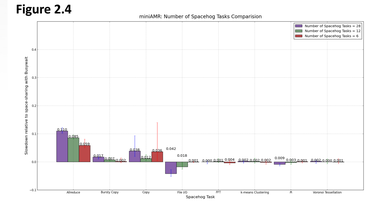
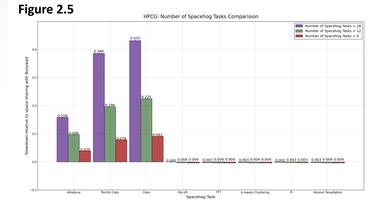


Figure 2.5



Architecture

Experiments performed by comparing results between two different systems: Volta (Cray XC30) & Chama (Infiniband cluster)

Figure 3.1

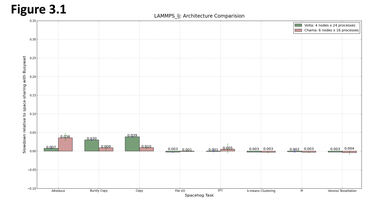


Figure 3.2

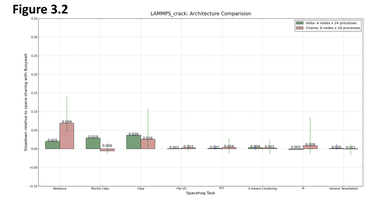


Figure 3.3

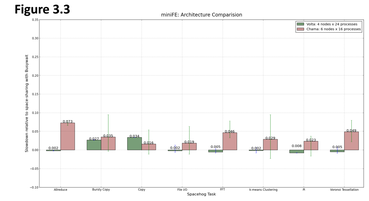


Figure 3.4

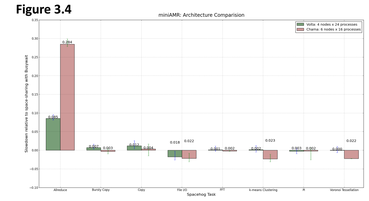


Figure 3.5

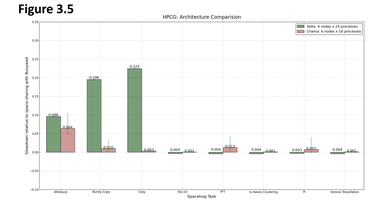


Table 1P: Descriptions of the analysis tasks used.

	Analysis Task	Description
Microbenchmark Tasks	Busy Wait	Just checks for simulation completion
	Allreduce	Performs a sequence of MPI_Allreduce operations.
	Copy	Continuously copies data from one memory allocation to another.
	Bursty Copy	Alternates intervals of sleep and intervals of copying memory.
	File I/O	Analysis tasks write random data to a POSIX file.
Proxy Tasks	FFT	Parallel computation of the fast Fourier transform (FFT) on an array of random data
	K-means Clustering	Parallel clustering of random data
	Pi	Monte Carlo computation of π
	Voronoi Tessellation	Parallel partition of a plane based on randomly selected points

Table 2P: Descriptions of the workloads used for evaluation.

Application	Description
LAMMPS	A classical molecular dynamics simulator from Sandia National Laboratories [1]. The data presented in this paper are from experiments that use the Lennard-Jones (LAMMPS-lj) and crack (LAMMPS-crack) potentials
miniFE	A proxy application that captures the key behaviors of unstructured implicit finite element codes [2].
miniAMR	A mini-application that captures the key behaviors of adaptive mesh refinement (AMR) codes [3].
HPCG	A benchmark that generates and solves a synthetic 3D sparse linear system using a local symmetric Gauss-Seidel preconditioned conjugate gradient method [4]

Table 3P: Descriptions of the two architectures used.

Volta

Cray XC30m
55 total nodes
2.4 GHz Intel Xeon E5-2695 (Ivy Bridge)
2 sockets x 12 cores
24 cores/node (Hyperthreading enabled)
64 GB DDR3 (1866 MHz) DRAM/node
Interconnect: Cray Aries DragonFly

Chama

Linux Cluster
1232 total nodes
2.6 GHz Intel Xeon E5-2670 (Sandy Bridge)
2 sockets x 8 cores
16 cores/node (Hyperthreading disabled)
64 GB DDR3 (1600 MHz) DRAM/node
Interconnect: Infiniband

References

[1] Steve Plimpton. 1995. Fast Parallel Algorithms For Short-Range Molecular Dynamics. Journal of Computational Physics 117, 1 (1995), 1–19.
[2] Michael A Heroux, Douglas W Doer er, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich. Improving performance via mini- applications. Sandia National Laboratories, Tech. Rep SAND2009-5574 (2009).
[3] Courtenay Vaughan, Richard Barrett, Enabling tractable exploration of the performance of adaptive mesh refinement, in 2015 IEEE International Conference on Cluster Computing, 2015, pp. 746–752.
[4] Sandia National Laboratories. HPCG High Performance Conjugate Gradient Benchmark. https://software.sandia.gov/hpcg/html/index.html.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.