Performance Implications of Cloud Sourcing Scientific Applications

Robert L. Cloud Michael J. Hannon Suman Silwal
R. Steven Wingo

December 10, 2011

1 Abstract

High-Performance scientific computing applications simulate natural phenomena on a large scale using mathematical models. Historically, these applications have been primarily targeted towards tightly controlled computing environments composed of massive quantities of commodity hardware components bound together with high performance interconnects (Beowulf type clusters). The most powerful of these clusters are extremely expensive, and access to their computational time is rare and valuable, but they enable the solution of previously intractable problems.

Recently, a new environment for scientific computing has emerged with the advancement of the Internet economy, which requires large server farms that often have excess storage and computational capacity. Organizations, such as Amazon.com, have created tools that enable one to leverage the spare capacity of these resources for, e.g., scientific endeavors. This development, known as cloud computing, enables more democratic access to high performance computational resources; investigators now do not have the burden of purchasing a cluster for their research, but rather can pay for compute instances on demand.

Our research analyzes the performance gradients of the cloud environments relative to the traditional cluster through the means of two scientific applications. First, we use a well-known discretization of Laplace's equation and measure the time required for it to converge with known boundary conditions on both the cluster and in the cloud. Furthermore, we will attempt to implement an inverse modeling application with experimental data and use this as an example of practically using cloud based resources to further scientific research.

2 Introduction

In this paper we investigate the characteristics of High Performance Computing (HPC) in both the traditional cluster/supercomputing environment and an abstracted computing environment, known as Cloud Computing. The latter approach lowers the economic barrier of access to HPC facilities by offering researchers access to the on-demand computational capacity. We study these environments and compare their relative performance by creating implementations of a classical mathematical model, Laplaces Equation. We also plan to apply both techniques to an inverse modeling application with experimental data from a pressure-pulse decay experiment used to determine geologic properties of porous samples.

2.1 High Performance Computing

Supercomputers are at the forefront of computational capability and lead the development of storage, interconnects and next generation processors. Over the past 60 years, supercomputers have increased dramatically in computational capability through Moores Law, generally stating that computing performance doubles every 18 months. However, because of the laws of physics and namely energy densities, Moores Law has changed in its mechanism; increased transistor densities are devoted to increasing the number of computational cores rather than scaling up clock frequencies. This leads to greater challenges for the programmer, as parallel, scalable algorithms must be developed, but the promises of supercomputing are great. With each new breakthrough in performance, previously intractable problems are now solvable.

HPC uses supercomputers to solve problems which are not possible on consumer-grade work stations. Although now we are seeing HPC applied in business contexts such as the IBM Watson Supercomputer used in applied healthcare, typically HPC has been used for solving numerically-intensive scientific problems by discretizing mathematical models of physical phenomena. As the capabilities of HPC are growing, opportunities for low cost, on-demand, scalable computers are appearing using servers made available as virtualized computing resources, marshaled together and offered as Cloud Computing.

2.2 Cloud Computing

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network, typically the Internet. It can be used for computationally intensive applications to process data or perform calculations in a fraction of the time needed for a workstation. While comparisons to the capabilities of parallel clusters for scientific problems still need to be performed, Cloud Computing has the obvious advantage of being accessible to anyone connected to the Internet, not just those who can afford access to a supercomputer. For the current study, the Amazon Elastic Compute Cloud, provided by Amazon.com, will be used. This service allows access to spot instances for Cloud Computing on a pay-as-you-use basis.

2.3 Laplace's Equation

Laplace's Equation is a fundamental tool in learning numerical analysis and scientific programming. While simple, it exhibits the common characteristics of much more complex equations, and its implementation can be used as a measurement of performance between computational environments. While having an analytical solution, it is more commonly solved with numerical finite-difference schemes. There are two popular ways to solve the problem: Jacobi and Gauss-Seidel iteration. Since it is better-suited for parallel processing, we have chosen Jacobi iteration for this investigation. The discretized form of Laplaces equation translates easily to a parallel scheme, where a number of rows are distributed to each parallel process. One can scale up this simple decomposition to create a problem as computationally expensive as needed by increasing the size of the domain and manipulating the number of processes or computational cores allocated to the problem. After validating the program a known solution, we can add further complexity by repeatedly creating semi-random, varying boundary conditions, and converging to a particular solution for each case. For each convergence, a data set is produced modeling the distribution of the independent variables at a steady state. In this way, the program taxes the cluster in a manner controllable by the programmer.

2.4 Inverse Modeling of Pressure-Pulse Decay of a Low-Permeability Porous Media

Modeling the physics of fluid flow through low-permeability porous materials is vital for many applications, including predicting transport of CO2 in underground reservoirs. However, determining properties of these media such as their absolute permeability or porosity often involves a long, painstaking measurement of the extremely small flow rates through their tortuous and heterogeneous pore paths. The length of time for such experiments using small cylindrical samples can often be on the order of days to weeks. Given the multiple independent variables (i.e. pressure, temperature, or confining stress) that may have an effect on these flow properties, a faster experimental method is very desirable. Several experimental procedures are considered below, each with a potential reduction in experimental time by an order of magnitude or more as compared to conventional methods. The results from such experimental procedures can provide invaluable data for predicting CO2 storage capacities for multiple geologic formations.

Conventional experimental techniques involve allowing fully-developed flow across a sample to reach a steady-state flow rate. However, with low-permeability materials, a fluid will take a very long time to reach a steady state. In order to overcome this limitation, a pressure-pulse-decay technique uses a mathematical model to measure the transient flow behavior rather than waiting for steady-state. Using experimental data and this mathematical model, the desired geological properties can be determined with a calculation-intensive inverse modeling procedure.

References

[1] Amal Alghamdi, Aron Ahmadia, David I Ketcheson, Matthew G Knepley, and Kyle T Mandli. PetClaw : A Scalable Parallel Nonlinear Wave Propagation Solver for Python. In *Proceedings of HPC*, number 1, 2011.

[2] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The Best of Both Worlds. *Computing in Science & Engineering*, 13(2):31–39, March 2011.

[3] Paul Brebner. Performance and cost assessment of cloud services. *Service-Oriented Computing*, 2011.

[4] Xing Cai, H.P. Langtangen, and Halvard Moe. On the performance of the Python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1):31–56, 2005.

[5] L Dalcin, R Paz, M Storti, and J Delia. MPI for Python: Performance improvements and MPI-2 extensions. *Journal of Parallel and Distributed Computing*, 68(5):655–662, May 2008.

[6] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. MPI for Python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, September 2005.

[7] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo a. Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, April 2011.

[8] Stefan Finsterle and Peter Persoff. Determining permeability of tight rock samples using inverse modeling. *Water Resources Research*, 33(8):1803, 1997.

[9] B. Granger. Python an ecosystem for scientific computing. *CiSE 2011 Special Python Issue*, page 5, 2010.

[10] S.A. Haskett, S.E.; Narahara, G.M.; Holditch. Simultaneous determination of permeability and porosity in low-permeability cores. *SPE Formation Evaluation*, (September 1988):651 – 658, 1988.

[11] S.E. Hsieh, P.A.; Tracy, J.V.; Neuzil, C.E.; Bredehoeft, J.D.; Silliman. Transient laboratory method for determining the hydraulic properties of tight rocks - 1. Theory Hsieh, P A; Tracy, J V; Neuzil, C E Int J Rock Mech Min Sci, V18, N3, June 1981, P245252. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, 18(5):89–89, October 1981.

[12] Andreas Kloeckner. Scripting GPUs with PyOpenCL. In *Scipy 2010*, 2010.

[13] Stefano Masini and Paolo Bientinesi. High-Performance Parallel Computations using Python as High-Level Language. In *EuroPar 2010*, 2010.

[14] X. Ning. *The measurement of matrix and fracture properties in naturally fractured low permeability cores using a pressure pulse method*. PhD thesis, 1992.

[15] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, number c, pages 427–436. IEEE, 2009.

[16] Dag Sverre Seljebotn. Fast numerical computations with Cython. In *Proceedings of the 8th Python in Science Conference (SciPy 2009)*, number SciPy, pages 15–22, 2009.

[17] José Unpingco. User Friendly High Productivity Computational Workflows Using the VISION/HPC Prototype. In *2008 DoD HPCMP Users Group Conference*, pages 387–390. Ieee, 2008.

[18] I.M. Wilbers, H.P. Langtangen, and Å smund Ø degård. Using Cython to Speed up Numerical Python Programs. In H. I. Andersson, editor, *Proceedings of MekIT*, volume 9, pages 495–512, 2009.