Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 154 (2016) 199 – 206

Procedia Engineering

www.elsevier.com/locate/procedia

12th International Conference on Hydroinformatics, HIC 2016

On the Efficiency of Executing Hydro-environmental Models on Cloud

Fearghal O'Donncha[a*], Emanuele Ragnoli[a], Srikumar Venugopal[a], Scott C. James[b], Kostas Katrinis[a]

[a]*IBM Research – Ireland, Damastown Ind. Park. Dublin 15, Ireland*
Baylor University, Department of Geosciences and Mechanical Engineering, One Bear Place #97354, Waco, TX

Abstract

Optimizing high-performance computing applications requires understanding of both the application and its parallelization approach, the system software stack and the target architecture. Traditionally, performance tuning of parallel applications involves consideration of the underlying machine architecture, including floating point performance, memory hierarchies and bandwidth, interconnect architecture, data placement – among others. The shift to the utility computing model through cloud has created tempting economies of scale across IT and domains, not leaving HPC as an exception as a candidate beneficiary. Nevertheless, the infrastructure abstraction and multi-tenancy inherent to cloud offerings poses great challenges to HPC workloads, requiring a dedicated study of applicability of cloud computing as a viable time-to-solution and efficiency platform. In this paper, we present the evaluation of a widely used hydro-environmental code, EFDC, on a cloud platform. Specifically, we evaluate the target parallel application on Linux containers managed by Docker. Unlike virtualization- based solutions that have been widely used for HPC cloud explorations, containers are more fit-for-purpose, sporting among others native execution and lightweight resource consumption. Many-core capability is provided by the OpenMP library in a hybrid configuration with MPI for cross-node data movement, and we explore the combination of these in the target setup. For the MPI part, the work flow is implemented as a data-parallel execution model, with all processing elements performing the same computation, on different sub-domains with thread-level, fine-grain parallelism provided by OpenMP. Optimizing performance requires consideration of the overheads introduced by the OpenMP paradigm such as thread initialization and synchronization. Features of the application make it an ideal test case for deployment on modern cloud architectures, including that it: 1) is legacy code written in Fortran 77, 2) has an implicit solver requiring non-local communication that poses a challenge to traditional partitioning methods, communication optimization and scaling and, 3) is a legacy code across academia, research organizations, governmental agencies, and consulting firms. These technical and practical considerations make this study a representative assessment of migrating

* Corresponding author. Tel.: +353 01 826924.
E-mail address: feardonn@ie.ibm.com

Peer-review under responsibility of the organizing committee of HIC 2016
doi:10.1016/j.proeng.2016.07.447

legacy codes from traditional HPC systems to the cloud. We finally discuss challenges that stem from the containerized nature of the platform; the latter forms another novel contribution of this paper.

Peer-review under responsibility of the organizing committee of HIC 2016

Keywords: Cloud; HPC; Numerical modelling; Containers; Hydrodynamic model

1. Main text

Resolving three-dimensional flows, transport and biogeochemical processes in surface water systems is a computationally intensive challenge. In particular, the drive to model more realistic and detailed scenarios, introduces increased computational demands of numerical solutions, due primarily to finer grid resolution and the simulation of a greater number of passive and active tracers. Parallel computing allows faster execution and the ability to perform larger, more detailed simulations than is possible with serial code. Traditionally, this high-performance computing is done on cluster systems featuring a hierarchical hardware design. Shared memory nodes with several multi-core CPUs are connected through network infrastructure. Parallel programming must combine distributed-memory parallelization on the node interconnect with shared-memory parallelization inside each node.

The benefits of cloud computing are well-known—efficiency, flexibility, high utilization—but have been difficult to achieve in technical and high-performance computing. The extraordinary performance demands that engineering, scientific, analytics, research workloads place upon IT infrastructure have historically meant that these workloads were not suitable for cloud deployments, especially those that use virtualization technology. Practical benefits arising from cloud deployments centre on availability, scalability, greater transportability offered by virtualization and a greater array of software choice. Traditional HPC users, however, have different features requirements, including [1]:

- Close to the "metal" – Many man-years have been invested in optimising HPC libraries and applications to work closely with the hardware, thus requiring specific OS drivers and hardware support.
- User space communication – HPC user applications often need to bypass the OS kernel and communicate directly with remote user processes.
- Tuned hardware – HPC hardware is often selected on the basis of communication, memory, and processor speed for a given application.

This paper investigates the performance of a widely used hydro-environmental code, Environmental Fluid Dynamics Code (EFDC), in a cloud environment. Many-core capability is provided by the MPI and OpenMP libraries in a hybrid configuration, and we explore the combination of these in the target setup. For the MPI part, the work flow is implemented as a data-parallel execution model, with all processing elements performing the same computation, but on different sub-domains. The domain decomposition strategy introduces additional considerations around load balancing of the application. Periodic synchronisation of each sub-domain at the end of each timestep means that the time to solution is constrained to that of the slowest sub-domain and hence requires intelligent distribution of each sub-domain to ensure approximately equal load on each processor. Fine-grain parallelism using openMP requires a much more invasive programming approach. Computationally intensive sections of the code are parallelized using compiler directives inserted at appropriate positions within the code to distribute workload across different threads. While explicit load balancing is not a factor for fine-grain parallelism, efficient performance requires consideration of the overheads introduced by openMP and the relevant computational cost to ensure that these overheads do not become punitive.

2. Methodology

EFDC is a public-domain, open-source, modelling package for simulating three-dimensional flow, transport, and biogeochemical processes in surface-water systems. The model is specifically designed to simulate estuaries and subestuarine components (tributaries, marshes, wet and dry littoral margins), and has been applied to a wide range of