

HOMME and POPperf High Performance Applications: Optimizations for Scale



WHITE PAPER

Abstract

The High Order Method Modeling Environment (HOMME) and the modified version of The Parallel Ocean Program (POPperf) are two important applications for atmospheric and weather research. With an emphasis on efficiency, portability, maintainability and most importantly, scalability, HOMME and POPperf have been successfully deployed over the years on a wide variety of high-performance systems, such as Cray and Blue-Gene. With the increased adoption of HPC commodity clusters based on the high-speed InfiniBand network, understanding HOMME and POPperf scalability and optimization options available for clusters at scale are crucial for utilizing the capability of InfiniBand-based systems to serve as cost effective high-performance and highly scalable solutions. Our results identify HOMME and POPperf scaling capabilities on one of the world's largest InfiniBand networks and demonstrate the critical elements for optimizing these applications at scale.

Introduction

The increasing research on the Earth's atmosphere and the growing ability to perform correct predictions require a highly scalable and accurate simulation of the atmospheric dynamics. Precise numerical schemes are essential to ensure high-fidelity simulations capable of capturing the convective dynamics in the atmosphere and their contribution to the global hydrological cycle. Scalable performance is necessary to exploit the massively parallel Petascale systems that will dominate high-performance computing (HPC) for the foreseeable future.

HOMME, The High-Order Method Modeling Environment [1], developed by the Computational and Information Systems Laboratory at the National Center for Atmospheric Research (NCAR) in partnership with the Computational Science Center at the University of Colorado at Boulder (CU), is a framework to investigate the utilization of high-order element-based methods to build scalable, accurate, and conservative atmospheric general circulation models (AGCMs). The primary objective of the HOMME project is to provide the atmospheric science community a framework for building the next generation of AGCMs based on high-order numerical methods that efficiently scale to

hundreds-of-thousands of processors, achieve scientifically useful integration rates and can easily be coupled to community physics packages.

POP, The Parallel Ocean Program [2] is an ocean circulation model derived from earlier models of Bryan, Cox, Semtner and Chervin in which depth is used as the vertical coordinate. The model solves the three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations. A wide variety of physical parameterizations and other features are available in the model and are described in detail in a reference manual distributed with the code. Because POP is a public code, many improvements to its physical parameterizations have resulted from external collaborations with other ocean modeling groups and such development is very much a community effort. Although POP was originally developed for the Connection Machine, it was designed from the start for portability by isolating all routines involving communication into a small set of modules which can be modified for specific architectures. Currently, versions of these routines exist for MPI and SHMEM communication libraries and also for serial execution. POP is the ocean component of the Community Climate System Model and has been used extensively at Los Alamos National Lab in ocean-only mode for eddy-resolving simulations of the global ocean and for ocean-ice coupled simulations with the CICE model. POP is freely available to the community (under a copyright agreement). In our testing we have used a modified version of POP – POPperf. POPperf [8] is a modified version of the standard POP 2.0 code that contains a number of modifications which improve scalability on large processor counts. They include: a re-writing of the conjugate gradient solver to use a 1D data structure, the addition of a space-filling curve partitioning technique and low memory binary parallel I/O functionality.

HOMME and POP simulations are typically carried out on large scale HPC systems. Among the HPC systems, HPC clusters have become the preferred solution as they provide an efficient performance compute solution based on industry standard hardware connected by a high speed network. The main benefits of clusters are affordability, flexibility and availability. A cluster uses the aggregated

power of compute server nodes to form a high-performance solution for parallel applications. When more compute power is needed, it can be simply achieved by adding more server nodes to the cluster.

The way HPC clusters are architected (i.e. multi-core, multi-processor-based HPC servers with high-speed interconnects) has a great influence on the overall application performance and productivity. In order to meet the demand of more powerful HPC servers, more execution cores (e.g. dual, quad-core) are being integrated into each processor and more processors are being tightly connected.

The cluster interconnect is crucial to deliver efficiency and scalability for the applications as it needs to handle the networking requirements of each CPU core without imposing additional networking overhead. In a multi-core multi-socket HPC server based cluster, the driving factors of performance and scalability for HOMME and POP have shifted from the frequency and cache size per core to the memory and interconnect throughput per core. The memory bottleneck can be relaxed by using interconnects that support Direct Memory Access (DMA), Remote DMA [4] and zero-copy transactions.

The scalability of high-performance commodity clusters

While HPC clusters architecture are being used for the majority of the HPC systems today (for example, more than 80% of the TOP500 [3] supercomputer list is listed as clusters), very large scale systems tend to use proprietary interconnect solutions such as Cray XT4 and XT5, IBM Blue-Gene and SGI NUMA. Historically, proprietary interconnects were necessary to provide the low latency, high bandwidth necessary to enable scaling to tens and hundreds of thousands of cores. However commodity based InfiniBand [5] clusters, which also provides a low latency and high bandwidth interconnect in addition to adaptive routing, congestion avoidance, MPI collective offloading ("CORE-Direct" [6]) can now deliver scalability on par with leading proprietary networks.

In order to assess the scalability of InfiniBand-based clusters, we have utilized the JUROPA systems for various application testing and optimizations. JUROPA [7] (the Jülich Research on Petaflop Architectures) is being used pan-European-wide by more than 200 research groups to run their data-intensive applications. JUROPA is based on a cluster configuration of Sun Blade servers, Intel Nehalem processors, Mellanox 40Gb/s InfiniBand and Cluster Operation Software ParaStation from ParTec

Cluster Competence Center GmbH. The system was jointly developed by experts of the Jülich Supercomputing Centre and implemented with the partner companies Bull, Sun, Intel, Mellanox and ParTec. It consists of 2,208 compute nodes with a total computing power of 207 Teraflops and was sponsored by the Helmholtz Community. The system was ranked number 10 on the June list of the TOP500 supercomputers.

In our testing we have not used the advanced capabilities as listed above, but mainly the latency, throughput and the CPU offloads capabilities of the InfiniBand network. We believe that with the advanced capabilities of the adaptive routing, congestion control and CORE-Direct, the scalability and the efficiency of InfiniBand clusters will certainly be improved. CORE-Direct is aimed to eliminate noise and jitter in large scale systems as it offloads the communications from the MPI to the network (collective operations), but it requires Mellanox's ConnectX-2 adapters, while JUROPA system includes the previous generation of ConnectX adapters.

ParTec ParaStation-5 MPI Critical Parameters

When planning to optimize MPI-based applications at scale, one needs to review and identify the critical MPI parameters that will greatly influence the application's performance or run time. For ParaStation MPI, the critical parameters are PSP_ONDEMAND, PSP_OPENIB_SENDQ_SIZE and PSP_OPENIB_RECVQ_SIZE.

PSP_ONDEMAND – each MPI connection needs a certain amount of memory by default. The more processes being used the more memory will be required for the MPI connections. As a rule of thumb each MPI connection utilizes roughly 0.5 MByte of memory. This is on top of the memory needed for the application itself. With ParaStation MPI, all possible MPI connections between all application's processes will be established at the beginning of the application's run within the MPI_Init() function. If the memory requirements of the application plus the memory allocation for the MPI connections will exceed the given memory capacity of the compute node, the application run will fail. In this case, the memory footprint of the MPI library has to be reduced significantly, and this is being done by setting the PSP_ONDEMAND to be true. As a result, MPI connections will be established per need, and not at the maximum level at the beginning of the application run. The default setting is not to use this behavior since otherwise connections being required to be established late during the application's run might fail due to insufficient amount of memory available. PSP_OPENIB_SENDQ_SIZE and PSP_OPENIB_RECVQ_SIZE

– these parameters enable another option for reducing the memory footprint/allocation by changing the default send/receive queue size. By default those parameters are set to 16 (for both), which means that each established connection will actually use about 0.55 MB of memory. Changing PSP_OPENIB_SENDQ_SIZE and PSP_OPENIB_RECVQ_SIZE to be 8 reduces the amount of memory allocated to each MPI connection. It can help to improve the performance in some cases, but fine tuning is needed in order to find the best buffer size per each application run.

POPperf Application at Scale

For POPperf we have explored three different scenarios. Scenario 1 is with the default PSP_ONDEMAND=0 setting, which means that all MPI connections are established at the beginning of the application run resulting in significant memory usage for larger jobs. In scenario 2 we have kept the default PSP_ONDEMAND=0 setting, but reduced the memory footprint by setting both PSP_OPENIB_SENDQ_SIZE=8 and PSP_OPENIB_RECVQ_SIZE=8. This setting reduces the memory allocated for each MPI connection from 0.55 MByte to 0.3 MByte. In the 3rd scenario we have enabled the dynamic establishment of MPI connections – and therefore the memory allocation – by setting PSP_ONDEMAND=1. At the same time this reduces the amount of memory allocated for the MPI connection during the application run time since only few connections are actually used.

POPperf Performance Results

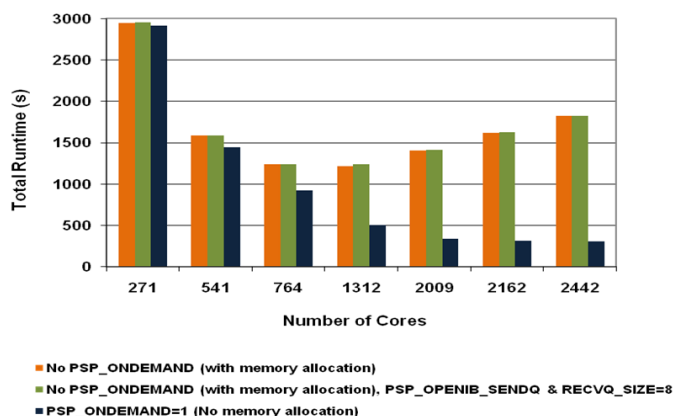


Figure 1 – POPperf performance results

The results are shown in Figure 1. For runs with less than 256 cores, PSP_ONDEMAND can be kept in the default setting, but beyond 256 cores, due to the large memory

allocation required for POPperf, dynamic establishment of the MPI connection is a must in order to achieve high performance and scalability.

By reducing the number of MPI connections, the time required for data received from MPI_ANY_SOURCE can be reduced as well, which affects the overall application performance as well.

Figure 2 shows the scalability results up to 2009 cores run. The scalability is being calculated by taking the performance per core from the 271 cores test, and comparing it to the performance per core in the larger core count tests. Setting PSP_ONDEMAND=1 enables achieving high scalability of nearly 100% from 541 cores to 2162 cores and 95% at 2442 cores. In fact, in some cases the scalability was greater than linear scalability. PSP_OPENIB_SENDQ_SIZE and PSP_OPENIB_RECVQ_SIZE parameters had a negligible effect on the achieved performance.

POPperf Scalability Results

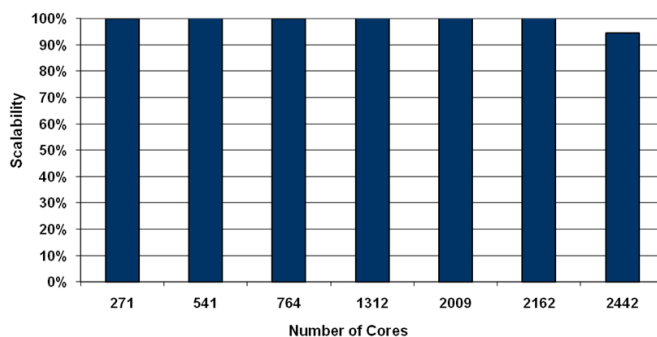


Figure 2 – POPperf scalability results

HOMME Application at Scale

For HOMME we have explored the same three scenarios. Scenario 1 is with the default PSP_ONDEMAND=0, PSP_OPENIB_SENDQ_SIZE=16 and PSP_OPENIB_RECVQ_SIZE=16 setting, which means that the memory will be allocated for each MPI connection at the beginning of the application run (as all MPI connections will be established at that time) and each MPI connection will consume 0.55 MByte. In scenario 2, we kept the default PSP_ONDEMAND=0 setting, but reduced the memory footprint by setting both PSP_OPENIB_SENDQ_SIZE=8 and PSP_OPENIB_RECVQ_SIZE=8. This setting reduces the memory allocated for each MPI connection at the beginning of the application run from 0.55 MByte to 0.3 MByte. In the 3rd scenario we enabled dynamic connections establishment by setting PSP_ONDEMAND=1. This enabled the memory to be dynamically allocated for

the MPI connection during the application run time and can lead to a significantly reduced memory footprint. The results are shown in Figure 3. Unlike POPperf, HOMME does not consume large memory in the tested core range (up to 3456 cores) and therefore the JUROPA system can accommodate the MPI connection memory allocation at the beginning of the application run, and avoid the dynamic allocation. Therefore using the default setting provides the best performance (up to 3% better than the other options).

HOMME Performance Results
(Standard.nl, ndays=12)

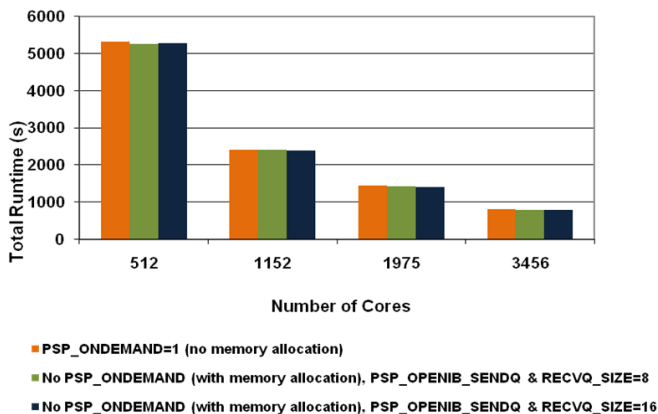


Figure 3 – HOMME performance results

With increasing system size, we noticed that beyond 8000 cores, the system memory cannot accommodate the allocation of the memory for all possible MPI connections and the performance was reduced. As such, we enabled the PSP_ONDEMAND=1 to set a dynamic allocation of the memory footprint for the MPI needs. The performance and scalability results are shown in Figures 4 and 5.

HOMME Performance Results
(Standard.nl, ndays=12)

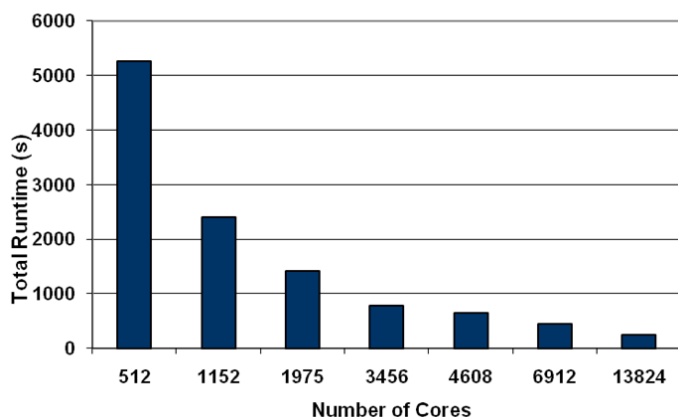


Figure 4 – HOMME performance results at scale

HOMME Scalability Results
(Standard.nl, ndays=12)

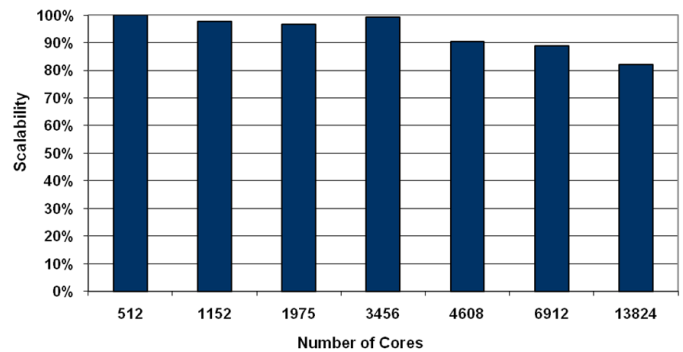


Figure 5 – HOMME scalability results

The results shown on Figures 4 and 5 were achieved by using the default MPI setting, i.e. PSP_ONDEMAND=0, PSP_OPENIB_SENDQ_SIZE=16 and PSP_OPENIB_RECVQ_SIZE=16 for the application runs up to 8000 cores, and with PSP_ONDEMAND=1 for the application runs beyond 8000 cores for optimum performance. Up to 3456 cores, the scalability demonstrated was above 90%, and at a system size of 13824 cores, scalability of more than 82% was demonstrated.

Summary

Numerical prediction models are critical tools for investigating the world we are living in, and to predict future changes. HOMME and POPperf are designed to provide accurate and sophisticated atmospheric and ocean models. For efficient simulations, HOMME and POPperf require high-performance computing systems. Commodity clusters have become very important for high-performance computing due to the price for performance, flexibility and scalability they can deliver. In this paper we analyzed HOMME and POPperf performance on an InfiniBand HPC cluster in order to identify best practices for researchers interested in maximizing the application performance, to determine the capability of InfiniBand-based clusters to scale for tens of thousands of cores, and to identify potential issues for scaling to hundreds of thousands of cores. The results of this study demonstrated that simplicity and value do not come at the expense of performance or scalability.

The results of our study show that an InfiniBand-based cluster is capable of providing an efficient, scalable high-performance system for applications at scale. We have found no issues that might prevent commodity clusters based on InfiniBand to be able to scale even beyond the tested environment, to hundreds of thousands of

cores. We did identify that each application might require different settings on the software interface or the message interface in order to achieve the high scalability and performance. We have concluded that InfiniBand-based system can provide an alternative to the more expensive, high-end, proprietary HPC solutions.

Due to the given hardware configuration of the JUROPA system, we were unable to include the new InfiniBand MPI collectives offload technology (CORE-Direct), which can definitely increase both the performance and scalability of the new InfiniBand-based systems, by eliminating the noise and jitter effect that is related to the MPI collectives operations and cause de-synchronization at large scale. Future work can add the CORE-Direct capability into application profiling at scale and determine the effect.

References

- [1] HOMME - <http://www.image.ucar.edu/CMG/Software/HOMME/>
- [2] POP - <http://www.cisl.ucar.edu/css/benchmarks/POPperf/>
- [3] TOP500 - <http://www.top500.org>
- [4] Gilad Shainer; "Why Compromise?", HPC Wire, October 2006
- [5] InfiniBand Trade Association – <http://www.infinibandta.org>
- [6] Richard L. Graham, Steve Poole, Pavel Shamis, Gil Bloch, Noam Bloch, Hillel Chapman, Michael Kagan, Ariel Shahar, Ishai Rabinovitz, Gilad Shainer; "ConnectX-2 InfiniBand Management Queues: First investigation of the new support for network offloaded collective operations", was send for publication, 2010
- [7] JUROPA - <http://www.fz-juelich.de/jsc/juropa>
- [8] J. M. Dennis, Inverse Space-filling Curve Partitioning of a Global Ocean Model , IEEE International Parallel & Distributed Processing Symposium, 26-30 March 2007, pp 1-10

Authors

Gilad Shainer¹, Tong Liu¹, Ben Mayer², John Dennis², Bastian Tweddell³, Norbert Eicker³, Thomas Lippert³, Axel Koehler⁴, Jens Hauke⁵, Hugo R. Falter⁵

(¹ Mellanox Technologies, ² National Center for Atmospheric Research, ³ Jülich Supercomputing Centre, ⁴ Sun Microsystems, ⁵ ParTec Cluster Competence Center)



350 Oakmead Pkwy, Sunnyvale, CA 94085
 Tel: 408-970-3400 • Fax: 408-970-3403
www.hpcadvisorycouncil.com