Performance Evaluation of MM5 on Clusters With Modern Interconnects: Scalability and Impact

Ranjit Noronha and Dhabaleswar K. Panda

Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210

Abstract. Clusters have become a crucial technology for providing low-cost high performance computing to scientific applications like weather prediction. In addition, networks like Myrinet, InfiniBand and Quadrics have become popular as an interconnection technology for high performance clusters. The high-bandwidth, low-latency characteristics of these networks make them ideally suited to the demanding characteristics of large scale weather simulations. Additionally, these networks have features like efficient and scalable hardware broadcast, reduce and atomic operations. Some of the features have been integrated into the MPI stack for these networks, allowing the user to exploit them for improved performance. In this paper, we evaluate the communication characteristics of a popular weather simulation code MM5 using InfiniBand. We also investigate how special features of InfiniBand like scalable broadcast can benefit MM5 performance. For some workloads, we see that InfiniBand performs up to 34% better than other interconnects. It also performs better in general than other networks for all workloads.

Keywords: MM5, Myrinet, InfiniBand, Quadrics, System Area Networks, Clusters

1 Introduction

Clusters have been widely deployed for providing high-performance computing for scientific applications. The lower cost of clusters means that several thousand nodes may be deployed for running large scale applications. Achieving improved performance from applications on large scale clusters is a challenging endeavor. This is especially so, given the wide diversity of architectures and networking technologies. Understanding the characteristics and trade-offs in different cluster architectures is crucial for achieving the best performance from applications.

To exploit the benefits of parallel computers, several large scale applications have been parallelized using implementations of the popular MPI standard [1]. MPI provides an interface to the application, abstracting out details of the underlying architecture and network. Computationally demanding applications such as weather simulation, computational fluid dynamic codes and crash simulation codes have MPI parallelizations [2–4]. Implementation of MPI such as MPICH have been ported to a variety of architectures and networks. This allows the

* This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506; National Science Foundation's grants #CCR-0204429 and #CCR-0311542; and equipment donations from Intel and Mellanox

application to be run on a wide-range of platforms. The application may potentially exploit the characteristics of these architectures and networks for improved performance.

Myrinet [5], InfiniBand [6] and Quadrics [7] are some of the popular networks used in high performance computing. These networks offer low latency of a few microseconds and high-bandwidth communication. Additionally, they offer several features which may be exploited by the application or MPI layers. Networks like Myrinet and Quadrics have a programmable network interface card (NIC), which may be used to offload application or system level computation [8]. Scalable collectives may be used to enhance application performance. Exploiting these features is possible not only at the MPI level, but also by the application itself.

In this paper, we evaluate a widely used weather simulation code MM5. We attempt to study its communication characteristics. This is done by analyzing the MPI calls characteristics with increasing system size. Following that, we study the impact of varying different network parameters like latency and bandwidth on the performance of the MM5. The impact of special network features is also evaluated.

The rest of the paper is organized as follows. Section 2 gives some background on high performance networks, bus technologies and MM5. Following that in section 3, the scalability of different workloads is evaluated. In section 4, the performance of MM5 while varying various network parameters is evaluated. Some related work is discussed in section 5. Finally, in section 6, conclusions and future work is presented.

2 Background

In this section, we discuss some of the topics relating to networks and weather simulation models. In particular, in section 2.1, we first discuss the networking technology Myrinet, InfiniBand and Quadrics. Following that in section 2.2, we discuss the weather simulation model MM5.

2.1 Overview of Cluster-Networking Technologies

In the high performance computing domain, Myrinet, InfiniBand and Quadrics are three of the popular networking technologies. InfiniBand [6] uses a switched, channel-based interconnection fabric, which allows for higher bandwidth, more reliability and better QoS support. The Mellanox implementation of the InfiniBand Verbs API (VAPI) supports the basic send-receive model and the RDMA operations read and write. There is also support for atomic operations and multicast. MVAPICH [9] is an implementation of Argonne's MPICH [1] over InfiniBand. The design of MVAPICH is based on the InfiniBand RDMA primitives. MVAPICH delivers small message latency of 5.0μs and large message bandwidth of up to 900 MillionBytes/sec. MVAPICH is designed to take advantage of hardware based multicast in InfiniBand [10].

Myrinet [5] is another low latency, high-bandwidth network which uses cut-through switches. Myrinet E-cards [11] are programmable and allow up to two ports for maximum bandwidth. MPICH-GM is an implementation of MPICH over Myrinet delivering small message latency of up to 6.0 μs and large message bandwidth up to 500 MillionBytes/sec. Quadrics [7] is another high-speed network. The current generation of Quadrics is Elan 4. Quadrics has a programmable NIC which can be used to offload computation from the host. MPI/Elan4 is an implementation of MPICH over Quadrics QsNet II. MPI/Elan4 can

send small messages with a latency of 2.4 μs and large messages with a bandwidth of up to 900 MillionBytes/s. The latency and bandwidth of these different networks is shown in Table 1. A basic comparison of these networks in terms of micro-benchmarks is presented in [12].

Table 1. Latency and Bandwidth for some high performance networks

Network	Latency(μs)	Bandwidth (MegaBytes/sec)
Myrinet (MPICH-GM)	6.0	500
InfiniBand(MVAPICH)	5.0	890
Quadrics(MPI/Elan4)	2.4	900

2.2 Overview of MM5

MM5 [4] is a limited area, non-hydrostatic, terrain following sigma-coordinate model designed to simulate or predict mesoscale atmospheric circulation. This regional model may be used for prediction on domains ranging from several thousand miles to a few hundred miles or less. Domains are uniform rectangular three dimensional areas of the atmosphere. The atmospheric dynamics are non-hydroscopic and use finite-difference approximations. The model is supported by several pre- and post-processing programs, which are referred to collectively as the MM5 modeling system. The MM5 modeling system software is mostly written in Fortran, and has been developed at Penn State and NCAR as a community mesoscale model with contributions from users worldwide.

The distributed-memory version of MM5 [13](MM5-MPP) has been implemented using MPI message-passing provided by the parallel Runtime System Library (RSL) [14]. RSL is a run-time system and library to support parallelization of grid-based finite-difference weather models. RSL supports *mesh refinement*. *Mesh refinement* allows the original domain to be divided into smaller areas (which may be nested). By allowing these areas to be non-uniform, computation may be focused on areas of more active interest in the domain. This usually sacrifices resolution in some areas of the domain (which may not be of interest), but reduces the computational requirements. RSL communicates results between sub-domains as shown in Figure 1. In the next section, we discuss how different workloads scale with an increase in the number of nodes.

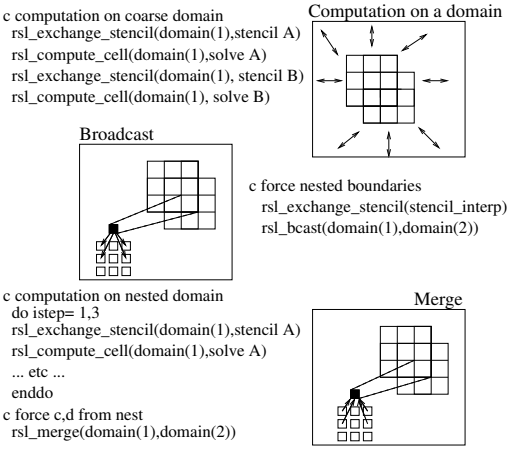

Fig. 1. Overall parallel driver for a MM5 timestep with nest interactions (courtesy J. Michalakes, et al. [13])

3 Communication Characteristics of Parallel MM5 (MM5-MPP)

In this section, we take a look at the communication patterns in the parallel version of MM5 (MM5-MPP) when using MPI over InfiniBand (MVAPICH). This was done to help us understand what parameters of a network would help us achieve better performance from this application. For example, if the application sends a lot of small messages, low network latency might help. If it sends large messages, it might be bandwidth sensitive. In addition, we would like to understand whether the application employs special operations like collectives. If it does, we would like to examine how efficient implementation of some collectives by certain networks might impact the performance of MM5-MPP. In-order to do this, we first evaluated how increasing the number of processors or system size, impacted the performance of the application. Also with increasing system size, we looked at how the distribution of MPI calls in the application changed. Finally, we also looked at how the message sizes to different MPI calls changed with increase in system size.

To evaluate these characteristics of MM5-MPP, we chose two different workloads and ran them on a 64-node, dual 2.4 GHz processor cluster with Mellanox MT23108 InfiniBand adapters and a MVAPICH 0.9.4 installation (cluster A). The first workload is the MM5 benchmark data set [15], which specifies a 3 hour run, TIMAX = 180, with an 81 second time-step (T3A). The second is the large-domain run (LDOM) which may be obtained from [16]. MM5-MPP allows the user to divide the workload among the different processors, so as to reduce the memory usage. This is achieved by specifying two parameters; namely number of processors in the North-South directions (PROCNS) and processors in the East-West directions (PROCEW) [13]. PROCNS and PROCEW were set so that PROCNS >= PROCEW and PROCNS x PROCEW = number of processors.

In Section 3.1, we look at the impact of increasing the number of processors on execution time. In Section 3.2, we look at the breakdown of time between application computation and communication layers. Following that in Section 3.3, the breakdown of time spent in the various calls in the MPI stack is presented. Finally in Sections 3.4 and 3.5, the distribution of MPI calls and message sizes with increasing system size is discussed.

3.1 Effect of system size

In this section, we observe the effect of an increasing number of processors on parallel execution time. The workloads T3A and LDOM were run on cluster A. Since each node has a dual processor, the total number of processors in the system is 128. This allows us to study the impact of system size up to 128 processors. The effect of increasing system size on execution time for T3A and LDOM is shown in Figure 2. It can be observed that with increasing system size for T3A, the execution time decreases up to 128 processors. For T3A, there is an approximate decrease in execution time of up to 37% when doubling the number of processors. Figure 2 also shows the scaling efficiency of the two workloads. It can be seen that for the workload T3A, the scaling efficiency starts at above 90% and then gradually decreases to a little over 65%.

For LDOM, there is a maximum decrease in execution time of 32% when doubling the number of processors. The scaling efficiency gradually decreases from 74% to approximately 49%, as shown in Figure 2. For LDOM, the benefits of an increasing system size plateaus after 64 processors. This can be attributed to the smaller problem size of LDOM compared to T3A. This leads to an increased load imbalance, which manifests itself as increased wait time. This effect is discussed in further detail in Section 3.2.

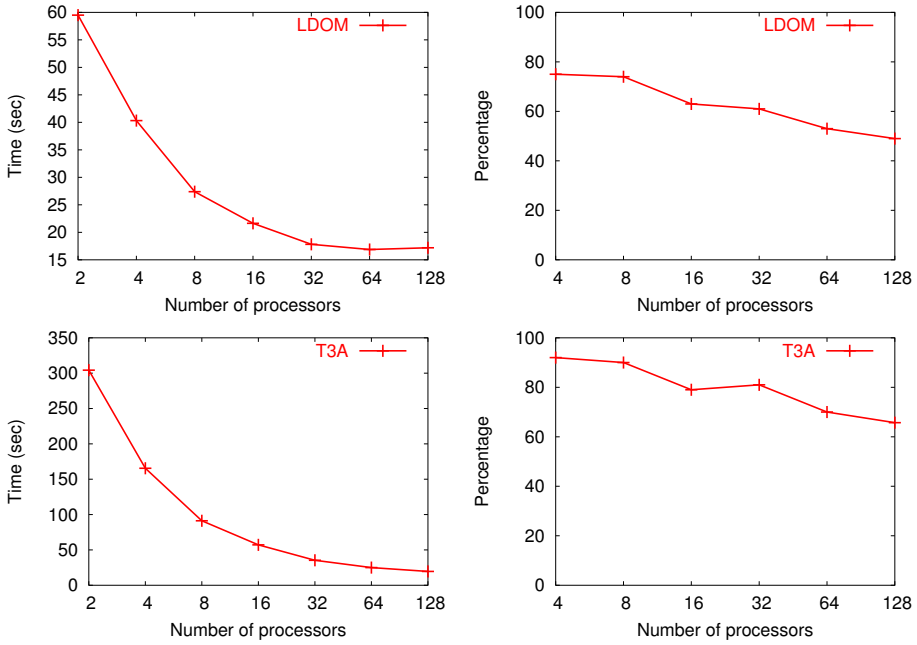

Fig. 2. MM5-MPP execution time with increasing system size of two different workloads (left) and scaling efficiency (right)

3.2 Overall application timing breakdown

We will now discuss the average per-process breakdown of execution time of the workloads LDOM and T3A. For improved scalability, it is better to spend the maximum amount of time in application level computation and as little time as possible in the communication libraries or in MPI calls. How much time is spent in the communication libraries is partly dependent on the design of the application as well as the communication library. If the application uses non-blocking MPI calls, this time can be minimized. Blocking calls on the other hand increase the amount of time spent in the communication libraries. The time spent in the communication library also depends partly on the nature of the progress function employed by the MPI stack.

The breakdown of timing was obtained using the lightweight profiling tool mpiP [17]. We find that for LDOM, the percentage of time spent in communication (time spent in MPI layers) increases from slightly less than 5% at two processors, to approximately 37% at 128 processors. For T3A, the percentage of communication increases from approximately 2% at two processors to 27% at 128 processors. This difference can be mainly attributed to the difference in sizes of the two workloads. LDOM is a smaller workload as compared to T3A. As a result, the computation datum assigned to each processor is smaller. This effect manifests itself as increased process skew. Overall, a large amount of time is spent in MPI layers particularly blocking MPI Receive calls. This issue will be discussed in more detail in Section 3.3.

3.3 MPI timing breakdown

In this section, we discuss the average per-processor distribution of time spent in different MPI calls for LDOM and T3A. Understanding the distribution of

time spent in different calls, gives us insight into which network might potentially enhance the performance of MM5-MPP. This is specially true in the case of efficient implementation of collective operations in some of the stacks such as MVAPICH [10]. The percentage of MPI time spent in different calls is shown in Figure 3. MM5-MPP largely uses the calls for blocking receive, blocking send, non-blocking receive, message wait, message broadcast and gather corresponding to MPI_Recv, MPI_Send, MPI_IRecv, MPI_Wait, MPI_Bcast and MPI_Gather respectively. Since the time spent in MPI_IRecv is not significant, it is not shown in the figure. For both datasets, the percentage of time spent in MPI_Bcast increases with increasing system size. For LDOM, time spent in MPI_Bcast increases from approximately 4% of total MPI time for a two processor run to approximately 30% at 128 processors. For T3A, MPI_Bcast time increases from 2% at two processors to about 20% at 128 processors. In Section 4.3, the impact of hardware broadcast on MM5 performance is evaluated.

A large percentage of time is spent in MPI_Recv for both LDOM and T3A and increases with increasing system size. At 128 processors for LDOM and T3A, the time spent is about 30% and 44% of communication time respectively. For both cases, time spent in MPI_Wait decreases with increase in system size. This decrease is more rapid in the case of T3A. Time spent in MPI_Recv and MPI_Wait can be correlated to the amount of application wait time. This is approximately 26% for LDOM and 21% for T3A. This would suggest that MM5-MPP would benefit from dynamic load balancing, currently not implemented in this version of MM5-MPP.

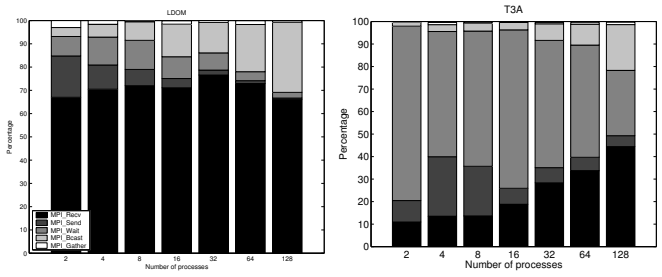

Fig. 3. Breakdown of time spent in different MPI functions for two different workloads

3.4 MPI call count distribution

In this section, we look at the average per-processor distribution of MPI calls in MM5-MPP. The distribution of MPI calls for LDOM and T3A with increasing system size is shown in Figure 4. As discussed in section 3.3, implementation of MM5-MPP makes calls to the MPI functions for blocking sends, blocking receives, non-blocking receives, broadcast and gather. These calls are MPI_Send, MPI_Recv, MPI_IRecv, MPI_Bcast and MPI_Gather respectively. Since the proportion of calls to MPI_IRecv is not significant, these calls are not shown in the graphs. For both workloads, the number of calls increases with increasing system size. For LDOM, MPI_Send has the highest count, while for T3A MPI_Bcast is the highest. For both cases, the number of calls to MPI_Send and MPI_Bcast increases ten-fold, when the system size is increased from two processors to 128 processors.

From this we observe that MM5-MPP largely uses blocking MPI calls. MM5-MPP might benefit from a design which uses more non-blocking calls. This might be possible through the modification of the *rsl_exch_stencil* and *rsl_merge_stencil*

calls in Figure 1 to use non-blocking calls. In this case, it might issue a non-blocking receive, to receive data from its adjacent neighbors. It might then continue computation on different sub-domains (assuming there is sufficient data available). Between computations, it might check if there is any additional data from its adjacent neighbors. If there is data available, it might use that to complete some computations rather than blocking. This would help us with overlap of computation and communication. This might also help reduce some of the application wait time discussed in section 3.3. We plan on investigating this in our future work.

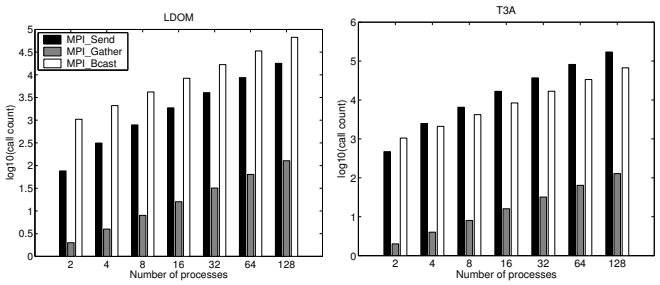

Fig. 4. Frequency of different MPI calls for the two different workloads

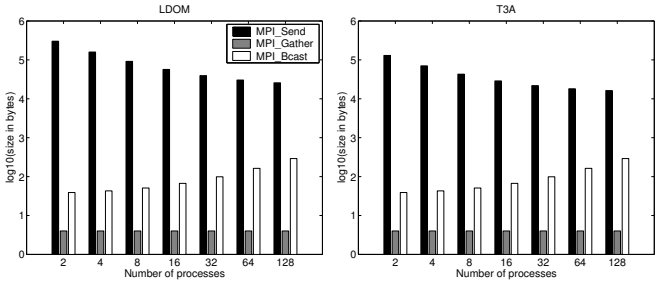

Fig. 5. Sizes of messages sent through different MPI calls in two different workloads

3.5 Message size distribution

As discussed in Section 3.4, MM5-MPP largely makes blocking MPI calls. In this section, we look at the average sizes of messages sent from these blocking calls namely MPI_Send, MPI_Recv, MPI_Bcast, and MPI_Gather. These results are shown in Figure 5. For both workloads LDOM and T3A, the size of the message passed to MPI_Send starts at between 129 KiloBytes and 300 KiloBytes at two processors and gradually decreases to about 40 KiloBytes at 128 processors. On the other hand, the size of messages passed to MPI_Bcast increases from about 50 bytes at two processors to approximately 300 bytes at 128 processors. It is possible that MM5 might benefit from InfiniBand hardware based multicast support integrated into MVAPICH. The impact of increase in unidirectional bandwidth on MM5-MPP performance is examined in section 4.1, while the impact of hardware multicast on MM5-MPP is examined in section 4.3. We will now examine the impact of different network parameters on the performance of MM5-MPP.

4 Impact of Network Technology

In this section, we look at how different network parameters affect the execution time of MM5-MPP. In particular, the impact of latency, bandwidth and hardware broadcast is examined. Experiments are conducted using the workloads LDOM and T3A described in Section 3. These workloads were run on cluster B (8-node, dual 3.0 GHz processor cluster with Myrinet E-cards, Quadrics Elan-4 and Mellanox MT23108 InfiniBand adapters). All experiments were run with 16 processes on eight nodes. In Section 4.1, the effect of network bandwidth on applications is examined. Following that, we look at the impact of network latency on MM5-MPP performance in 4.2.

Table 2. Explanation of notation used in this section.

Notation	Explanation
MPICHGM-1P	MPICHGM 1.2.6..14a using E-cards, with a single port activated (GM 2.0.21)
MPICHGM-2P	MPICHGM 1.2.6..14a using E-cards, with both ports activated (GM 2.1.21)
MVAPICH-1N	MVAPICH 0.9.5 with a single NIC per node
MVAPICH-HB	MVAPICH 0.9.5 with InfiniBand hardware broadcast enabled
MVAPICH-SB	MVAPICH 0.9.5 without InfiniBand hardware broadcast
MPI/Elan4	Quadrics MPI

4.1 Effect of Network Bandwidth

In this section, we examine the impact of bandwidth on the performance of MM5-MPP. This impact was measured using both different networking technologies, as well as multi-port support offered by different technologies. Myrinet E-cards [11] has two ports, each capable of up to 250 MegaBytes/sec for a total of up to 500 MegaBytes/sec. It is possible to activate either one or both ports on these cards. We use notation as explained in Table 4. For large messages, MVAPICH-1N delivers up to 900 MegaBytes/sec. MPI/ELAN4 delivers up to 900 MegaBytes/sec [9]. The two workloads were run on cluster B, described in Section 4. The execution time across different networks for LDOM and T3A at 16 processes on 8 nodes is shown in Figure 6. For LDOM, execution time is reduced by approximately 34% when MPICHGM-2P is replaced by MVAPICH-1N. The reduction in execution time may be attributed mainly to to the reduction in time spent in MPI_Bcast (24.2%), followed by the reduction in MPI_Recv (5%), along with small reductions in MPI_Wait, MPI_Gather and MPI_Send making up the remaining 5%. Note that hardware broadcast was not enabled for MVAPICH-1N. For T3A, on replacing MPICHGM-2P with MVAPICH-1N, there is a reduction in execution time of up to 12%. Most of this reduction comes from reduced time spent in MPI_Bcast.

4.2 Effect of Network Latency

We will now examine the effect of network latency on the performance of MM5-MPP. For the different network MPI stacks, we use notation similar to that in Table 4. On cluster B, the latency of a 0-byte message for MPI/Elan4 is approximately 2μs while for MVAPICH-1N it is 5μs. The bandwidth for large messages of these two networks is comparable as shown in Table 1. The execution time of

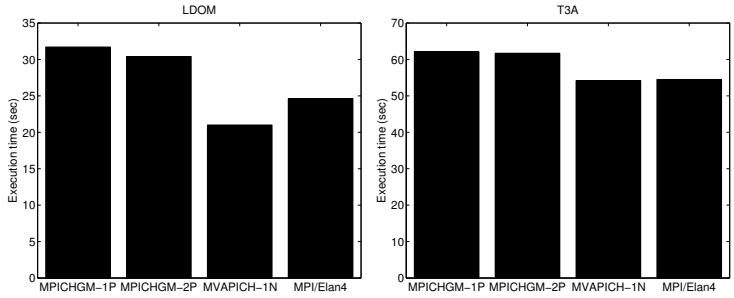

Fig. 6. MM5-MPP execution time with different networks

the two workloads LDOM and T3A at 16 processors, on eight nodes is shown in Figure 6. At 16 processors, for LDOM MVAPICH-1N performs better than MPI/Elan4 by approximately 20%. Most of this difference may be attributed to time spent in MPI_Recv and MPI_Wait. For T3A, there is very little difference in performance between MVAPICH-1N and MPI/Elan4.

4.3 Effect of Hardware Broadcast

In this section, we evaluate the impact of hardware based broadcast in InfiniBand on the performance of MM5-MPP. As discussed in section 3.3, and shown in Figure 3, a significant amount of time spent in the blocking call MPI_Bcast. At 16 processors for LDOM, approximately 10% of time is spent in MPI_Bcast. For T3A at 16 processors, approximately 5% of time is spent in MPI_Bcast. Also as discussed in Section 3.5, at 16 processors, the message size passed to MPI_Bcast by both T3A and LDOM is approximately 100 bytes. At this size, hardware based broadcast does better by up to 50% in terms of latency than the current software based point-to-point algorithm [10]. It seems likely that MM5-MPP could potentially benefit from InfiniBand hardware broadcast.

The workloads LDOM and T3A were evaluated with and without hardware broadcast referred to as MVAPICH-SB and MVAPICH-HB respectively on cluster B, as explained in Section 4. All runs were taken up to 16 processes on eight nodes. For LDOM there is a reduction in execution time of approximately 2.14%. For T3A, the reduction in execution time is approximately 5.1%.

5 Related Work

The parallel implementation of MM5, MM5-MPP, was described in [13]. Only basic scalability in terms of execution time is discussed here. The performance using different commodity cluster interconnects is not discussed in this paper. Also the impact of efficient collective operations in modern interconnects on application performance is not discussed. The evaluation of the MM5 benchmark T3A on various architectures is carried out in [15]. Only the basic scalability in terms of execution time with increasing number of processors is discussed here. The impact of various network features like multicast is not evaluated here. The performance and scalability of various networks is evaluated using micro-benchmarks and NAS parallel benchmarks in [18]. This study focuses on comparing Myrinet, Quadrics and InfiniBand. The relative performance of Myrinet, InfiniBand and Quadrics in terms of micro-benchmarks is evaluated in [12]. There is no application-level evaluation here.

6 Conclusions and Future Work

In this paper, we have looked at the scalability of the parallel distributed memory version of the popular weather simulation code MM5. We have also looked at the sensitivity of MM5 to network parameters like latency, bandwidth and efficient collectives like hardware broadcast in InfiniBand. MM5 uses messages sizes of the order of 100 to 300 KiloBytes for system sizes up to 16 processors. These sizes decrease with increase in system size. A considerable amount of time is spent in the collective call MPI_Bcast which increases with increasing system. We conclude that, at smaller system sizes, MM5 would benefit from increased bandwidth. Experimentation with InfiniBand shows a reduction in execution time up to 34% compared with Myrinet at 16 processors. For larger system sizes, the improved latency of hardware based broadcast might be more beneficial to the application. Experimentally on a 16 processor environment, we see an improvement of up to 5% in overall execution time when using InfiniBand based hardware broadcast. Additionally, MM5 spends substantial time waiting in the MPI calls MPI_Wait and MPI_Recv as system size increases. We would like to determine the impact of efficient communication progress functions available in stacks like Myrinet MX and Quadrics on the performance of MM5, for large scale systems. MM5 also packs and unpacks its own data structures. We would like to investigate the effect of efficient zero-copy datatypes on the performance of MM5.

References

1. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, (Argonne National Laboratory and Mississippi State University)
2. Fluent CFD. (http://www.fluent.com)
3. LSDYNA. (http://www.lstc.com)
4. G.A. Grell, J. Dudhia, and D.R. Stauffer: A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5). Tech. Rep. NCAR/TN-398+STR, National Center for Atmospheric Research, Boulder, Colarado (1994)
5. Boden, N.J., Cohen, D., et al.: Myrinet: A Gigabit-per-Second Local Area Network. IEEE Micro (1995) 29–35
6. Infiniband Trade Association. (www.infinibandta.org)
7. Quadrics Ltd. (www.quadrics.com)
8. R. Noronha and N. B. Abu-Ghazaleh: Using Programmable NICs for Time Warp Optimization. IPDPS (2002)
9. MPI over InfiniBand Project. (http://nowlab.cis.ohio-state.edu/projects/mpi-iba)
10. J. Liu, A. Mamidala and D.K. Panda: Fast and Scalable MPI-Level Broadcast using InfiniBand's Hardware Multicast Support. IPDPS (2004)
11. Myrinet E-cards. (http://www.myri.com/myrinet/PCIX/m3f2-pcixe.html)
12. J. Liu, B. Chandrasekaran, W. Yu, J. Wu, D. Buntinas, S. Kini, P. Wyckoff and D. K. Panda.: Micro-Benchmark Performance Comparison of High-Speed Cluster Interconnects. IEEE Micro. (2004)
13. J. Michalakes, T. Canfield, R. Nanjundiah and S. Hammond: Parallel Implementation, Validation and Performance of MM5. Sixth Workshop on the Use of Parallel Processors in Meteorology, European Center for Medium Range Weather Forecasting, Reading, U.K. (1994)
14. Michalakes, J.: A Runtime System Library for Parallel Finite Difference Models with Nesting. Technical Report ANL/MCS-TM-197 (1997)
15. Parallel MM5 benchmarks. http://www.mmm.ucar.edu/mm5/mpp/helpdesk/20040304a.html (2004)
16. MM5 Community Model. (http://www.mmm.ucar.edu/mm5/)
17. mpiP MPI Profiling Tool. (http://www.llnl.gov/CASC/mpip)
18. R. Brightwell, D. Doerfler and K.D. Underwood: A Comparison of 4X InfiniBand and Quadrics Elan-4 Technologies. IEEE Conference on Cluster Computing. (2004)