Performance Characterization of a Molecular Dynamics Code on PC Clusters
Is there any easy parallelism in CHARMM?

Michela Taufer[1], Egon Perathoner[1,3], Andrea Cavalli[2], Amedeo Caflisch[2], Thomas Stricker[1].

[1] Department of Computer Science
ETH Zurich, Zentrum
CH-8092 Zurich, Switzerland
stricker,taufer@inf.ethz.ch

[2] Department of Biochemistry
University of Zurich
CH-8057 Zurich, Switzerland
caflisch@bioc.unizh.ch

[3] Università degli Studi di Padova,
Dipartimento di Elettronica ed Informatica,
Via Gradenigo 6/a, Padova, Italy

Abstract

The molecular dynamics code CHARMM is a popular research tool for computational biology. An increasing number of researchers are currently looking for affordable and adequate platforms to execute CHARMM or similar codes.

To address this need, we analyze the resource requirements of a CHARMM molecular dynamics simulation on PC clusters with a particle mesh Ewald (PME) treatment of long-range electrostatics, and investigate the scalability of the short-range interactions and PME separately. We look at the workload characterization and the performance gain of CHARMM with different network technologies and different software infrastructures and show that the performance depends more on the software infrastructures than on the hardware components. In the present study, powerful communication systems like Myrinet deliver performance that comes close to the MPP supercomputers of the past decade (e.g. Cray T3D), but improved scalability can also be achieved with better communication system software like SCore without the additional hardware cost.

The experimental method of workload characterization presented can be easily applied to other codes. The detailed performance figures of the breakdown of the calculation into computation, communication and synchronization allow to derive good estimates about the benefits of moving applications to novel computing platforms such as widely distributed computers (grid).

Keywords: *middleware, distributed computation, performance tuning, performance optimization, cluster of PCs, message passing, distributed molecular dynamics.*

1 Motivation

The most recent increase of activity in computational biology has created a big demand for affordable computing platforms that can run well established molecular dynamic codes, like CHARMM. Clusters of PCs can provide a tremendous compute power at a remarkably low cost once the application can deal with the characteristics that differ from traditional massively parallel multiprocessors (MPP). In most clusters currently used for CHARMM, the utilization of parallelism is limited to executing multiple CHARMM calculations at the same time (task parallelism) and little effort is made to speed up a single calculation. The goal of this paper is to quantify the benefits of data parallelism in CHARMM calculation executing on a variety of cluster platforms.
In general, we want to answer the question whether there is any easy parallelism in a typical CHARMM calculation. More precisely, we want to know which number of processors can be assigned to a single calculation given certain characteristics of a cluster platform until we reach the limits of scalability. Since we are interested in the more recent versions of CHARMM, we consider the performance of the classic computation together with the enhanced computation in particle mesh Ewald (PME) model incorporating part of the electrostatic force calculations in the frequency domain. It is to be determined whether the advanced computational method requires a better cluster than the classical method or if the scalability will be limited. In particular, we look more closely at the compute platform on which CHARMM could run. There are several choices of interconnects among the compute nodes of a cluster that come at different costs. What is their impact on the performance and the scalability of the code? There are several software systems that can be installed to support the parallel version of CHARMM. Are any of them more suitable than others?

The PCs commonly used as cluster nodes can often accommodate a second processor at a low cost. Is the use of dual-processor nodes worthwhile?

In our performance characterization study, we present a collection of performance measurements that can help to provide answers to these questions. In Section 2, we introduce some previous work dealing with performance issues of CHARMM and give a condensed overview of the physics used in CHARMM with and without PME model. We describe the experimental molecular structure and the experimental platform used for our study. In Section 3, we present our experimental design and characterize the computational steps of CHARMM in terms of their resource usage. In Section 4, we go through a systematic performance analysis of a CHARMM calculation on different cluster platforms, discussing the impact of the networks, the middleware and the processor configuration of cluster nodes. In Section 5, we summarize our results and discuss the extrapolation of performance estimates beyond traditional cluster platforms.

2 CHARMM (Chemistry at HARvard Macromolecular Mechanics)

CHARMM is a program for simulating biologically relevant macromolecules (proteins, DNA, RNA) and complexes thereof [2]. It allows to investigate the structure and dynamics of large molecules (solute) in the condensed phase (solvent or crystal). CHARMM can be used to calculate free energy differences upon mutations or ligand binding [12]. Moreover, it has been used to simulate the reversible folding of structured peptides [5, 7] and determine folding free energy surfaces [1, 6]. It uses classical mechanical methods to investigate potential energy surfaces derived from experimental and ”ab initio” quantum chemical calculations [14]. Furthermore, mixed quantum mechanical/classical systems can be defined to investigate chemical processes such as enzyme catalysis. One of the most common application of CHARMM is molecular dynamics (MD), in which the Newton equation of motion is discretized and solved by an integration procedure (Verlet algorithm). The force on the atoms is the negative gradient of the CHARMM potential energy [14].

2.1 Particle Mesh Ewald (PME)

Current molecular dynamics simulations sample between 1 and 1000 nanoseconds (depending on the size of the solute the treatment of solvent and the computer) while large conformational transitions of biological relevance (e.g. protein folding) take place in microseconds to minutes. Hence, it is essential to evaluate the energy and its gradient (force) in an efficient way. On the other hand, because DNA (or RNA) and protein molecules have many formal charges, it is very important to use an accurate treatment of long-range electrostatic interactions. Therefore, the main problem is to find the right balance between the accuracy of the energy function (systematic error) and the length of sampling (statistical error). One accurate and rather efficient way of treating the long-range electrostatics is the particle mesh Ewald (PME) approach which scales as Nlog(N), with N denoting the system size. The total electrostatic energy is split into local interactions which are computed explicitely and long-range interactions which are approximated by a discrete convolution on an interpolating grid, using the 3-dimensional fast Fourier transform (FFT) to efficiently perform the convolution. In the PME method implemented in CHARMM the electrostatic energy is split into direct and reciprocal Ewald sum [4]. The latter is the solution of the Poisson's equation in periodic boundary conditions, with Gaussian charge densities as sources.

2.2 The Molecular System

The molecular system used in our simulations is myoglobin, a 153-residue single domain protein of structural class α (i.e. only α-helical secondary structure), a carbonmonoxide molecule, 337 water molecules and a sulfate ion for a total of 3552 atoms. This is an interesting medium-size molecular system of biological relevance. It has been chosen for this study which focuses on networks and the communication software infrastructure. Furthermore, myoglobin represents an archetypal single-domain protein. Hence, its choice is also justified by the fact that most of the experimental studies on protein folding focus on single domain proteins of about 50-150 amino acids and the need of comparing the simulation results with experimental data. The CHARMM input files were downloaded from [3]. The long-range electrostatics effects are approximated by the PME method. The FFT grid points for the charge mesh are 80 x 36 x 48.

2.3 Previous Work

In previous migrations from vector supercomputer to massively parallel multiprocessors (MPP), the code showed a remarkable scalability as long as sufficient networking performance was available on the machine [8, 16, 9]. However, migrating CHARMM to clusters of commodity PCs has been a mixed success and migration to widely distributed computing on the Internet (Grid) remains a particular challenge [15]. The increased use of the particle mesh Ewald (PME) method in frequency domain does affect the computation/communication balance and renders parallelization more difficult.

In a preliminary technical report [17], we have studied the overall organization of the CHARMM communication and its impact on the performance. Based on these first series of experiments alone, it can not be concluded if overlap of the computation and the communication is beneficial or detrimental to performance and scalability of CHARMM on a particular platform. Decoupling computation, synchronization and data transfer resulted in better performance for certain compiled parallel programs on the Cray T3D and other machines [21].

2.4 The Experimental Platform

The performance tests are conducted on a well interconnected cluster of dual processor Pentium III nodes running at 1 GHz [20]. In particular, our cluster is interconnected by different high speed interconnect technologies, such as the Gigabit Ethernet (a typical LAN technology) and Myrinet [1] (a typical SAN technology). Linux has been used as operating system for our experiments with CHARMM.
Our 16 node system is fully dedicated to performance studies and development of communication system software. Therefore, there is no overhead on the measurements due to a timesharing environment or other workloads. In the tests, every timing has been repeated several times and the results are checked for low variability and a good reproducibility. After a stable testing environment is reached, we conduct our study with a reduced amount of molecular dynamics (MD) simulation steps.

3 Workload Characterization of CHARMM

A proper understanding of the performance of CHARMM requires a proper and systematic workload characterization of the code. The systematic experimental design presented in this section will help to sort out the relevant and irrelevant effects of the numerous parameters that determine the performance of CHARMM on distributed platforms.

3.1 Experimental Design

Previous work reports simply about a successful implementation effort, about a new algorithm incorporated or a new target machine. For our study, we specified the goal of investigating a broader variety of software and hardware platforms that are systematically chosen and varied. We conducted the study according to a proper experimental design in which we deliberately specify the computation we are running according to the criteria of a computer scientist and systems architect. The measured performance of the system are considered to be *response variables*, the variables which affect the response variables are considered to be *factors* and the values of the factors are called *levels* [11]. With this systematic approach, we attempt to gather the maximum information with the minimum number of experiments. At the same time, we determine the factors that have a significant effect on the response variables and quantify their effect. Since we are interested in quantifying the impact of PME as a variant of the algorithm in the application and the impact of the different software infrastructures, we look at classic and PME enhanced CHARMM computations. We particularly focus on factors which are directly or indirectly related to the network configuration. For additional factors like the processor type, the clock frequency or the compiler we refer the interested reader to a more technical report that deals with a similar topic [17].

[1]Our Myrinet Installation comprises a 16 line switch and older M2F-PCI32C network interfaces with a lanai on board processor.

Figure 1 displays the factor space that we have considered during the experimental design. Our factors are of three

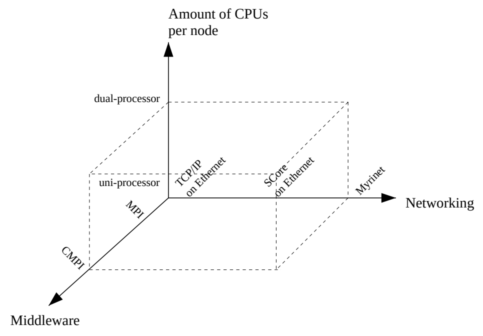


Figure 1. Factor space with its several levels.

types: (1) factors related to the network technology and the system software (i.e. *Networking*), (2) factors related to the communication libraries (i.e. *Middleware*) and (3) factors related to the node architecture (i.e. *amount of CPUs per node*). The factors related to the PC Cluster architecture on which CHARMM executes are called platform factors. We include different hardware infrastructure and different software infrastructures as platform factors. As expected, there are many interactions among the different communication libraries, the communication network and the node configuration (i.e. the number of CPUs per node) in the cluster.
In terms of networking hardware, we consider two physical interconnect technologies: Gigabit Ethernet and Myrinet. We bundle them with different driver software into a single platform factor. Both technologies are Gigabit/s interconnects, whereas Gigabit Ethernet is widely used as LAN (local area network), while Myrinet is a typical SAN (system area network) technology. A comparison to the highly common Fast (100 MBit/s) Ethernet interconnects is found in an earlier technical report [17]. Surprisingly, the Fast Ethernet has almost the same performance characteristics and the same interactions as Gigabit Ethernet. The interesting part of the communication system is the software infrastructure which is closely tied to the underlying hardware. For the communication library, we take three different communication infrastructures into account. All of them offer a standard MPI API (Application Programming Interface) [19] but rely on different communication mechanisms:

- Standard implementation of MPICH using the TCP/IP protocol over Gigabit Ethernet,
- Specialized SCore library [2] using its own protocol for reliable communication with high bandwidth and low latency communication over Gigabit Ethernet [10],
- Vendor specific MPICH-GM library using the advanced coprocessor on the Myrinet network interface

[2]SCore stands for System CORE. At first, SCore was the name of parallel operating system, but as the time went by it has become cluster system software.

card and the specific properties of the Myrinet interconnect (large packets, link level error and flow control).

The middleware factor comprises the kind of communication routines used by the application code. We have two different implementations of CHARMM:

- The standard implementation uses raw MPI calls in which point-to-point blocking communication routines are used and the necessary global synchronization operations are done by standard MPI barriers.

- An alternative implementation relies on a middleware layer called CMPI or CHARMM MPI that relies heavily on nonblocking communication using split send/receive calls to MPI as communication primitives. For best possible portability, all remaining synchronization operations are implemented by repeated exchanges of empty messages (or one byte) among nearest neighbor-processes.

In our past studies, we investigated similar middleware packages [22] that claim to facilitate the implementation of distributed computation, while pretending to preserve good portability through high level abstraction. According to previous experience with such middleware, the firm grip on the communication system can be lost and the resulting performance can easily become unstable and highly sensitive to platform factors.
Finally, as a third factor, multiple processors on a node can increase the compute performance at a remarkable low cost, especially if standard motherboards are prepared to accommodate a second processor. Therefore, we look at two cluster configurations with single and dual CPUs per nodes (1 GHz). In the dual-processor case, two processor share one memory and one communication system in a symmetric shared memory multiprocessor configuration. Our experimental CoPs cluster permits experiment with up to 16 nodes and a total of 32 processors.
Although we gathered all data of a full factorial design [11] (we did benchmark CHARMM for all 12 cases with factors at all levels), we limit the discussion of our result to a fractional factorial design centered around the focal point of a most common cluster configuration based on MPICH using TCP/IP over Gigabit Ethernet, with uni-processor nodes. For the study, we look at one factor at a time moving along the axes of the three dimensional space of test cases, as shown in Figure 1.

3.2 Characterization of the Important Computational Steps

The calculation of the energy in general and the non-bonded energy interaction over all pairs of atoms in particular is the most resource demanding and consuming time phase of a MD simulation [17]. In CHARMM without PME model (electrostatic interactions shifted to zero at 10 Å), the energy calculation comprises execution of multiple copies of the same routines without any communication. Only at the end of this computation, an all-to-all collective communication gathers the results from different processors. In the rest of the paper, we refer to this part of the calculation as the *classic energy calculation*. The variation of CHARMM based on the PME model, adds some additional computations in frequency domain to the classic energy calculation. The transformation from the time domain into the frequency domain and backwards requires two three dimensional FFTs. In the parallel version, such a FFT adds a communication step with an all-to-all personalized communication pattern. The global FFTs and the calculation of the additional energy and force contribution (electrostatic interactions between partial charges separated by more than 10 Å) is accounted for in the *PME energy calculation*.
Figure 2 shows the structure of the energy calculation without PME model composed of the classic energy calculation routine and the structure of the energy calculation with PME model composed of two different components: the classic energy calculation, which is independent from the PME model, and the PME energy calculation. For each of the several cases considered during the workload characterization, we measure the time of the two components *classic calculation* and *pme calculation* for ten MD simulation steps. Figure 3 shows the wall clock time of the clas-

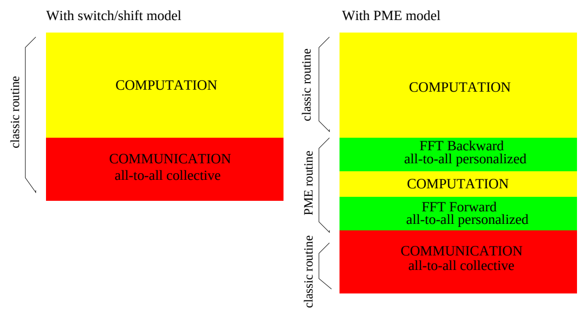


Figure 2. Structure of the energy calculation routines in CHARMM without and with PME model.

sic energy calculation and the PME energy calculation for the reference case chosen (CHARMM running on TCP/IP, MPI middleware and uni-processor cluster). In the sequential version of CHARMM (one processor), the PME time is slightly less than half of the total calculation time. In the parallel version, the PME time is almost two thirds of the total calculation time and for two processors, the execution time of the PME calculation is actually larger than for a one processor. This adversely affects scalability. In search of the cause of those scalability problems, we broke down the execution time further into computation, communication and synchronization for both types of energy calculations. We label their computation time *classic comp* and *pme comp*, their communication time *classic comm* and *pme comm* and their synchronization time *classic sync* and *pme sync*. In this study, we split general communication overhead into time spent for data transfer, which we label communication time and time spent for control transfer and

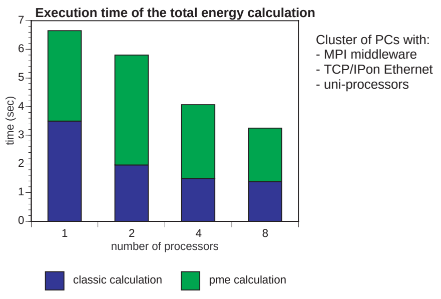
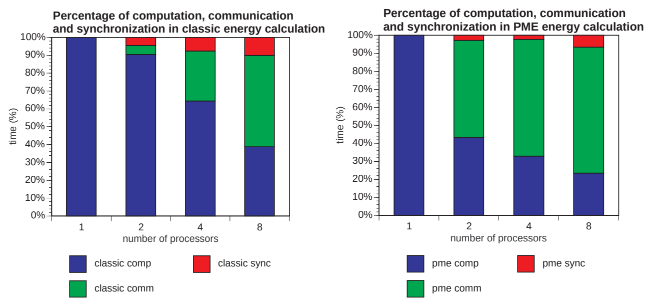


Figure 3. Wall clock time of the total energy calculation for the reference case chosen in our experimental design (CHARMM running on TCP/IP Ethernet, MPI middleware and uni-processor cluster). We split the total calculation time of the energy into two components: the classic energy calculation and the PME energy calculation.

coherency maintenance, which we label as synchronization time.

Figure 4.a displays the percentages of the communication, computation and synchronization times for the classic energy calculation, while Figure 4.b for the PME energy calculation. As expected, there is only computation in the one processor case. In the parallel case of the classic energy calculation, the overheads for communication and synchronization are less than 10% for two processors increasing to over 60% for eight processor. In the parallel case of the PME energy calculation, the overheads for communication and synchronization are climbing even more rapidly: from slightly more than 50% for two processors to over 75% for eight processors. This explains the inefficiency of the parallel version on the most common cluster configuration (TCP/IP, Ethernet, MPI).



(a) (b)

Figure 4. Percentage of computation, communication and synchronization in the classic energy calculation (a) and in the PME energy calculation (b) for the reference case (CHARMM running on TCP/IP Ethernet, MPI middleware and uni-processor cluster).

4 Performance Analysis of CHARMM

In this section, we investigate the effect of each factor reported in our experimental design (the network, the middleware, and the number of CPUs per host), starting from the reference case (focal point) in Figure 1 and moving to the extreme end one factor at a time.

4.1 Performance Impact of Network Technology

In our study, we consider different communication protocols on different network infrastructures with different cost and features. First, we look at the basic communication protocols of the Internet, TCP/IP on Gigabit Ethernet, which provides high bandwidth but does not optimize latency and overheads for small messages. Using the general Internet infrastructure with the TCP/IP protocol-stack is very popular in cluster computing since this software is readily available and free for every PC purchased. As an alternative, we look at the SCore communication infrastructure that works directly on top of the raw Ethernet and delivers the same bandwidth for large transfers, but has significantly lower latency and smaller per-packet-overheads. Finally, the vendor specific system area network (SAN) Myrinet is taken into account. Myrinet can deliver a communication performance that comes close to the massively parallel supercomputer of the past decade (e.g. the Cray T3D in 1993) [13]. Together with its software infrastructure using a communication coprocessor, the MPICH-GM for Myrinet delivers high bandwidth, low latency and small per-packet-overheads. The more powerful communication system systems come at a significant additional cost that can exceed 50% of the machine cost for Myrinet.

Figure 5 compares the wall clock times in seconds measured for the total energy calculation with TCP/IP on Gigabit Ethernet, Score on Gigabit Ethernet and Myrinet (all other factors are kept at the same level). For all cases, we consider the time of the classic energy calculation (*classic calculation*) and the time of the PME energy calculation (*pme calculation*) for CHARMM on a uni-processor cluster and MPI middleware. The chart shows a better scalability for the networks with low latency and smaller overheads. This obser-

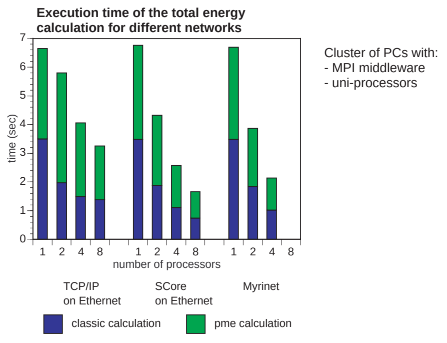
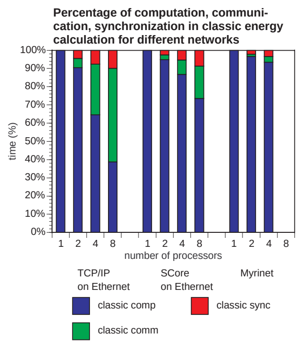
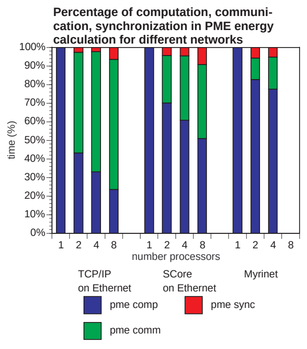


Figure 5. Wall clock time of the total energy calculation for the different networks considered in out study: TCP/IP on Gigabit Ethernet, SCore on Gigabit Ethernet and Myrinet.

vation is confirmed in Figure 6.a and Figure 6.b in which the relative amount of computation, communication and synchronization for the classic energy and PME energy are given in detail for the three networks. The pictures demonstrate clearly that the increased cost of communication and the loss of performance is strongly correlated to the amount of latency and overhead. The cost of synchronization alone remains within reasonable limits and is similar for all three networks considered. The performance of barriers and synchronization is determined mostly by the latency. The big difference arrises from the cost of the communication operations (i.e. the data transfers) providing solid evidence that differences in performance are due to different overheads in the communication infrastructure including the system software.



(a)



(b)

Figure 6. Percentage of computation, communication and synchronization in the classic energy calculation (a) and in the PME energy calculation (b) for TCP/IP on Gigabit Ethernet, SCore on Gigabit Ethernet and Myrinet.

Figure 7 shows the average communication speed and its variability per node (in MByte/sec) of CHARMM running on MPI middleware and uni-processor cluster. The vertical lines show the maximal and minimum communication rates measured. The figure reveals a low communication rate of TCP/IP on Gigabit Ethernet and a large variability of the speeds for this communication protocol. SCore provides stable and higher communication rate on Gigabit Ethernet. The measurements confirm the high communication rate of Myrinet. In particular, the high variability of MPI transfers over TCP/IP starts abruptly with four processors and get worse with eight processors. The interaction between TCP/IP flow control and message passing communication on clusters is likely to result in highly uneven and unstable performance. This suspicion is confirmed by the low variability of SCore communication that uses the same network (Gigabit Ethernet) but does not rely on the TCP/IP flow control algorithm. The exceptionally powerful backplane of the switch used in our 16 node CoPs cluster excludes switch contention as a cause for the bottlenecks. This is confirmed by constant throughput of the software infrastructure based on SCore.

Many Beowulf clusters are not yet equipped with Gigabit Ethernet and built with Fast Ethernet links connected to more or less powerful switches and hubs. The comparison of CHARMM running on a Fast Ethernet (lower bandwidth) vs. Gigabit Ethernet (higher bandwidth) is handled in previous work and is properly described in [17]. Gigabit Ethernet did not perform much better than Fast Ethernet and the strong dependency on the communication protocols and the software infrastructure is consistent across the different communication speeds.

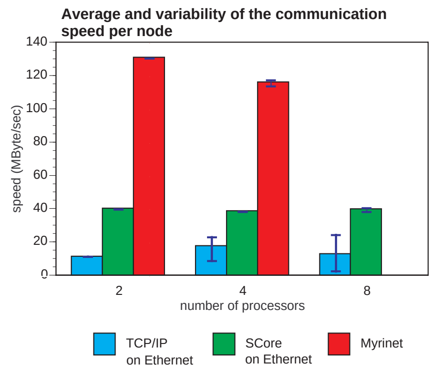
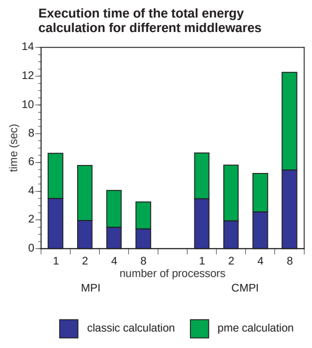


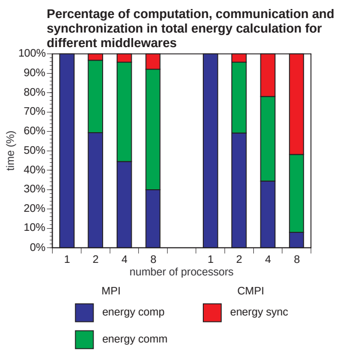
Figure 7. Average and variability of the communication speed per node in MBytes/sec for CHARMM running on MPI middleware and uni-processor cluster.

4.2 Performance Impact of Middleware

Many parallel and distributed scientific codes use a layered software system to facilitate the programming of interprocess communication and to reach better portability. We call these extensions of the bare bone message passing libraries *communication middleware*. For a better understanding of the effect of this middleware on the performance of CHARMM, we compare the reference case of CHARMM with standard MPI calls to a version using CMPI (CHARMM MPI) middleware. Again the other factors are not varied.



(a)



(b)

Figure 8. Wall clock time (a) and percentage of computation, communication and synchronization (b) in the total energy calculation for MPI middleware and CMPI middleware.

Figure 8.a displays the execution time of the classic energy calculation and the PME energy calculation for the two communication styles (i.e. with and without the use of

CMPI). With the increase of the number of slaves from four to eight, both parts of the execution time, i.e. the time for the classic energy calculation and the time for the PME energy calculation, are increasing instead of falling when CMPI is used. Since both parts show the same trends, we can look at the cumulative times for a more detailed explanation.

Figure 8.b shows the breakdown of the total energy calculation into the computation time (*energy comp*), communication time (*energy comm*) the synchronization time (*energy sync*). This chart explains the sudden slow down with a total loss of scalability in the synchronization operations that are performed in the CMPI middleware. A closer examination of the source code of CMPI reveals that a single synchronization call is built upon repeated send and receive calls transmitting a single byte with the neighbor-nodes and this operation is repeated $p-1$ times for p processors. This way of synchronization is inefficient for network protocols with per-packet-overheads such as TCP/IP on Ethernet.

4.3 Performance Impact of Multiprocessor Nodes

As a last consideration, we look at the effect of the number of processors used in each node of the cluster. Most motherboards of PCs provide a slot for a second processor as an cost effective mean to increase computation performance. Since the performance impact of the additional processor strongly depends on the network technology, we must look at two different places in the factors space. We consider the variation from the focal point (CHARMM with MPI over TCP/IP on Gigabit Ethernet) and, in parallel, the variation from the case of CHARMM over Myrinet from uni-processor to dual-processor nodes. Figure 9.a shows the wall clock time of the classic energy calculation and PME energy calculation for CHARMM on Gigabit Ethernet, while Figure 9.b displays the wall clock time of the classic energy and PME energy calculations on Myrinet.

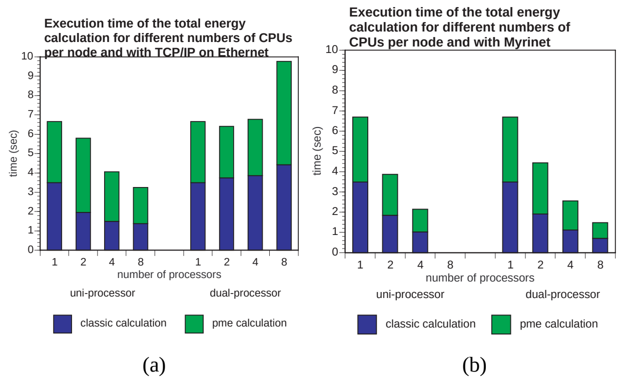


Figure 9. Wall clock time of the classic energy and PME energy on uni-processor and dual-processor clusters. CHARMM runs on TCP/IP and Gigabit Ethernet (a) and on Myrinet (b).

Figure 9.a reveals a scalability problem for CHARMM on dual-processor clusters when the network infrastructure is MPI over TCP/IP on Ethernet. In Figure 9.a, both the classic energy time (*classic calculation*) and the PME energy time (*pme calculation*) does not decrease but increases with the number of nodes in the dual processor case. Therefore, the use of dual processor nodes does adversely affect the scalability of CHARMM. This is not the case for network technologies such as SCore and Myrinet, as seen in Figure 9.b.

A possible cause for the loss of performance could be in the inefficient handling of the communication by the two processors that share same node with just one communication system. The results indicate that for CHARMM on a TCP/IP layer, the communication protocol is not able to route the incoming communication to the proper destination processor. In many operating systems, only one processor can handle interrupts of the network interface and this interrupt-handling may become a bottleneck [18]. On the other hand, Myrinet and SCore use a shared memory driver which is able to handle the communication in a more effective way.

5 Conclusion

The systematic and detailed performance characterization presented in this paper takes a closer look at the computation, communication and synchronization requirements of a typical CHARMM calculation on a cluster of PCs. We differentiate between a classic CHARMM calculation, which takes place entirely in the time domain, and an advanced CHARMM calculation, which adds some particle mesh Ewald (PME) computation in the frequency domain. We show that the PME method increases the dependency on the better networks and better computation infrastructures.

We study the scalability of the code on different network technologies (Gigabit Ethernet, Myrinet) and on different software infrastructures, including a conventional TCP/IP based MPICH library and SCore, a specialized MPICH library for Ethernet and Myrinet. For classic CHARMM calculation, the scalability to larger systems remains dissatisfactory as long as we rely entirely on the standard communication infrastructure of MPI over TCP/IP on Ethernet. Only with improved communication system software, like e.g. the SCore communication library, we can achieve lower communication overheads and good scalability for larger problems and larger clusters.

The amount of parallelism in CHARMM should suffice to run efficient parallel calculations on clusters with up to the 32 to 64 processors, which is the size of clusters that is most commonly installed. For more advanced calculations using the particle mesh Ewald method, good scalability is limited to parallel calculations spanning a reasonable fraction (e.g. a quarter) of such a cluster. For more parallelism, a low overhead, high speed interconnect like e.g. Myrinet must be included to achieve good scalability with PME. Since most research groups have multiple CHARMM calculations that could run in parallel, the cost of this additional network must be evaluated carefully. However, running a single CHARMM calculation faster provides a much shorter turn-around increasing research productivity.

We have further noted a strong dependency on the programming style and middleware used for the interprocess communication within CHARMM. The combination of weaker networks and general purpose protocols (e.g. TCP/IP on Ethernet) with a inefficient programming style (in portable CHARMM MPI) or multiprocessor nodes can heavily penalize communication performance and scalability even if sufficient network bandwidth is is available to both processors. A typical warning sign of an instable cluster configuration is a large variation of the communication throughput numbers. From the highly different communication overheads in Ethernet, SCore and Myrinet, we learn that optimizing the communication code with proper programming skills for software systems will add a significant amount of scalability to CHARMM at no extra hardware cost. Unfortunately, such optimized communication software is fairly costly to write and hard to maintain.

We believe that the precise and detailed timings given in this study contribute to a better insight into how well CHARMM will execute on different flavors of PC clusters and on novel computing platforms like widely distributed computing on the global computational grid.

Acknowledgments

We would like to express our thanks to all the people who helped us during this work. We thank Christian Kurmann and Felix Rauch who helped with our many questions about SCORE. We are very grateful to Roger Karrer for reading carefully through several drafts of our work. We are also very grateful to Urs Haberthuer of the University of Zurich for his help and his chemistry advice.

References

[1] E. M. Boczko and C. L. Brooks III. First-Principles Calculation of the Folding Free Energy of a Three-He. *Science*, 269:393–396, 1995.

[2] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[3] Pittsburgh Supercomputing Center. CHARMM examples for TCS. http://www.psc.edu/general/software/packages/charmm.

[4] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh Ewald method. *J. Chem. Phys*, 103:8577–8593, 1995.

[5] P. Ferrara, J. Apostolakis, and A. Caflisch. Thermodynamics and kinetics of folding of two model peptides investigated by molecular dynamics simulations. *J. Phys. Chem. B*, 104:5000–5010, 2000.

[6] P. Ferrara and A. Caflisch. Folding simulations of a three-stranded antiparallel β-sheet peptide. *Proc. Natl. Acad. Sci. USA*, 97:10780–10785, 2000.

[7] P. Ferrara and A. Caflisch. Native topology or specific interactions: What is more important for peptide folding? *J. Mol. Biol.*, 306:837–850, 2001.

[8] CLRC: Council for the Central Laboratory of the Research Councils. Application Performance on High-End and Commodity-Class Computers. Technical Report, collaborative computational project, Computational Science and Engineering Department, Council for the Central Laboratory of the Research Councils, Daresbury and Appleton-Rutherford Labs, Oxon and Cheshir, United Kindom. http://www.dl.ac.uk/CCP/CCP1/docs/beowulf.pdf, 2001.

[9] Y.-S. Hwang, R. Das, J. Saltz, B. Brooks, and M. Hodoscek. Parallelizing Molecular Dynamics Programs for Distributed Memory Machines: An Application of the CHAOS Runtime Support Library . Technical report, University of Maryland, Department of Computer Science and UMIACS Technical Reports CS-TR-3374, UMIACS-TR-94-125.

[10] Y. Ishikawa, H. Tezuka, A. Hori, S. Sumimoto, T. Takahashi, F. O'Carroll, and Harada H. RWC PC Cluster II and SCore Cluster System Software – High Performance Linux Cluster. In *Proceedings of the 5th Annual Linux Expo*, pages 55–62, 1999.

[11] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley Professional Computing, 1996.

[12] X. Kong and C.L. Brooks. lambda-dynamics: A new approach to free energy calculations. *J. Chem. Phys*, 105:2414–2423, 1996.

[13] Ch. Kurmann and T. Stricker. A Comparison of Three Gigabit Technologies: SCI, Myrinet and SGI/Cray T3D. In *Scalable Coherent Interface*, pages 39–68, 1999.

[14] A.D. MacKerell Jr. and et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[15] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, A. Grimshaw, and C. Brooks III. Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion. In *Proceeding of the HPDC Conference*, San Francisco, CA, USA, Aug 7-9 2001.

[16] CHARMM Benchmarking Page. National Institute of Chemistry in Ljubljana, Slovenia. http://www.kihp6.ki.si/parallel/summary.html.

[17] E. Perathoner. Performance Driver Migration and Optimization of a Common Molecular Dynamics Code (CHARMM) on Different Cluster Platforms. Technical report, ETH Zurich, Apr 2001.

[18] F. Rauch. Personal communication. ETH, Zurich, 2001.

[19] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI - The complete Reference*. MIT Press, 1997.

[20] T. Stricker, C. Kurmann, F. Rauch, and M. Taufer. CoPs: Cluster of PCs - Project investigates architectural and operating system support for parallel and distributed computing on PC clusters interconnected with a Gigabit/sec network. http://www.cs.inf.ethz.ch/stricker/CoPs/.

[21] T. Stricker, J. Stichnoth, D. O'Hallaron, S. Hinrichs, and T. Gross. Decoupling Synchronization and Data Transfer in Message Passing Systems of Parallel Computers. In *Proc. Intl. Conf. on Supercomputing*, pages 1–10, Barcelona, July 1995. ACM.

[22] M. Taufer and T. Stricker. Accurate Performance Evaluation, Modelling and Prediction of a Message Passing Simulation Code based on Middleware. In *Proceeding of the SC98 Conference*, Orlando, FL, USA, Nov 7-13 1998.