Parallel Monte Carlo simulations on an ARC–enabled computing grid

Jon K. Nilsen¹, Bjørn H. Samset²
¹Department of Physics, University of Oslo
²CICERO Center for International Climate and Environmental Research, Oslo
E-mail: j.k.nilsen@fys.uio.no, b.h.samset@cicero.uio.no

Abstract.

Grid computing opens new possibilities for running heavy Monte Carlo simulations of physical systems in parallel. The presentation gives an overview of GaMPI, a system for running an MPI-based random walker simulation on grid resources. Integrating the ARC middleware and the new storage system Chelonia with the Ganga grid job submission and control system, we show that MPI jobs can be run on a world–wide computing grid with good performance and promising scaling properties. Results for relatively communication-heavy Monte Carlo simulations run on multiple heterogeneous, ARC-enabled computing clusters in several countries are presented.

1. Introduction

Monte Carlo (MC) simulations of physical processes are usually too heavy to process in a reasonable time on a single CPU, or even a multi–core machine. The natural extension from a single machine is to run the simulations in parallel on a cluster of nodes, with one or more cores each, and have the processes communicate via the Message Passing Interface (MPI).

With the recent improvements in grid computing technology, especially in the areas of storage solutions and user–space job configuration tools, it is interesting to run massively parallel MC simulation on a large computing grid. The main obstacles to doing this have been the lack of efficient communication between parallel threads, which on a cluster would be handled via MPI, and also the complexity of configuring a grid–based MC solution.

In this paper we present the results of a diffusion Monte Carlo (DMC) simulation, running on NorduGrid resources, via the Advanced Resource Connector [1, 2] (ARC) grid middleware. Inter–process communication is handled through the new Chelonia grid storage system [3, 4], while the jobs are configured and submitted via the Ganga grid user tool [5, 6]. The DMC application and the MPI framework used for comparison is described in [7], while in Section 2 we describe the GaMPI framework. Next, in Section 3, we show three series of performance tests where we run various configurations of the DMC simulation on the grid, employing ARC–enabled computing clusters located in several countries before we come to some concluding remarks in Section 4.
Figure 1. The communication flow between Ganga (laptop), Chelonia (top cloud) and ARC (bottom cloud), and between the services internally in Chelonia. Communication between Ganga and the grid is carried out by exchanging files through the Chelonia cloud. Internally, Chelonia consists of sets of metadata services and storage nodes.

2. GaMPI description
When running simulations in parallel on a cluster, processes typically communicate using the message passing interface (MPI). While there exists MPI implementations for cross-site, inter-cluster connections [8, 9], they are either not readily available on production grids or requires direct login to the clusters to be utilized. GaMPI is a grid framework for running simulations in parallel on a production grid. While GaMPI is not a MPI implementation for grid, it does have certain similarities. As sketched in Figure 1, GaMPI is a framework using an ARC-enabled grid, Ganga and Chelonia. In analogy with a standard MPI system, ARC provides access to a distributed cluster with a large number of CPUs, Chelonia plays the role of the MPI protocol, allowing the parallel processes to communicate when needed and Ganga is the glue core process that sends and controls MPI jobs on the cluster.
In somewhat more detail, the workflow of GaMPI can be described as follows:

(i) A master process is running inside Ganga, which creates, submits and monitors the grid jobs.

(ii) The master splits the required number of walkers into blocks. The walker blocks are uploaded to Chelonia, together with a semaphore file indicating that the work can be started.

(iii) When the work can be started, the grid jobs try to download a random block of walkers. When all the walkers in the block are processed, the grid job uploads the walker block with a new prefix.

(iv) In DMC, the main variables of interest after a timestep are the number of walkers, which is dynamic, and the ground state energy after moving the walkers. As Chelonia supports arbitrary metadata for a file, these two variables are appended to the metadata of the walker block file and the master process only checks the metadata between every timestep.

(v) The master monitors the walker pool, and as soon as there are only moved walker blocks in the walker pool, the master removes the semaphore file.

(vi) The master then checks the metadata of the files to get the ground state energy and the number of walkers. Depending on these numbers, the master alters the metadata and renames the files to have the original prefix and uploads the semaphore file again.

(vii) After repeating steps (i) to (v) for the required number of time steps, the master uploads a file signaling the grid jobs to end their work and exit.

Figure 1 shows the communication flow between the components of GaMPI. Here, Ganga and the grid jobs communicate by exchanging files through Chelonia. Chelonia itself consists of a set of services which can broadly be divided into metadata services (MDS’s) and storage services (StS’s). The MDS’s are contacted for looking up files and file information, for adding new files, getting files and deleting files, i.e. for all metadata operations. The actual metadata is then stored in a database shared between the MDS’s. The StS’s handles the physical storing and maintenance of data. In the setup shown in Figure 1, each of the StS’s are linked to one set of MDS’s. In a production setup of Chelonia, each of the storage services would be linked to both sets of MDS’s so that if one MDS goes down, the StS can fail-over to the other set of MDS’s. However, with only two sets of MDS’s there is a great likelihood that both StS’s connect to the same set of MDS’s, thus generating a skew load-balance between the two sets of MDS’s.

3. DMC and performance results on the grid

This section presents performance results on running the DMC simulation described in [7] via the GaMPI framework described above. Three major test series were run, and are described in turn below. The first establishes a baseline result on an MPI cluster and then shows that the same performance can be achieved using our GaMPI setup on the grid. The second test studies the scaling of GaMPI performance with the number of grid jobs run in parallel, and the third studies the scaling to more complex simulations by increasing the number of random walkers.

3.1. From cluster to grid

For the first test series we ran three identical simulations, each with 30k walkers, 50 timesteps and 16 CPUs or grid jobs. The jobs were sent (a) to a single MPI–enabled cluster, (b) to the same cluster but using the GaMPI setup, and (c) to a set of three clusters in three different countries.

The results are presented in Table 1. Our performance metric is the average walltime per timestep that the simulations used, in addition to the maximum and minimum time taken by a single timestep. The maximum and minimum values indicate both variations in actual
### Table 1

| Case | Description                  | Minimum (s) | Average (s) | Maximum (s) |
|------|-------------------------------|-------------|-------------|-------------|
| a    | MPI, single cluster           | 735         | 790         | 838         |
| b    | GaMPI, single cluster         | 545         | 778         | 1191        |
| c    | GaMPI, three clusters         | 738         | 885         | 1157        |
| d    | 16 CPUs, 60k walkers          | 1388        | 1656        | 1993        |
| e    | 32 CPUs, 60k walkers          | 785         | 1073        | 1454        |
| f    | 60 CPUs, 60k walkers          | 692         | 802         | 1060        |
| g    | 16 CPUs, 30k walkers          | 738         | 885         | 1157        |
| h    | 32 CPUs, 60k walkers          | 785         | 1073        | 1454        |
| i    | 60 CPUs, 120k walkers         | 1245        | 1387        | 1705        |
| j    | 120 CPUs, 240k walkers (prelim) | 858         | 1063        | 1268        |

Table 1. Timings per iteration (time-step) for running 50 iteration of diffusion Monte Carlo using regular MPI on a single cluster, using GaMPI on the same cluster, and running GaMPI with grid jobs distributed between three clusters in three different countries. In cases a, b and c 16 CPUs were used in parallel and in cases d through j all grid jobs are distributed. Note that the timings do not take into account the number of walkers in each time-step. Values are for timesteps 30 through 50.

computing time due to a varying number of walkers per block, and in the load on the system used in each case. To capture the variation due to computing effects and not simply due to expected variations between the DMC timesteps, we allow the system to run for 30 timesteps before comparing. We then calculate the average and find max/min values for timesteps 30 through 50 only.

Table 1 shows that for our baseline test (a) on a normal MPI cluster, the average time per step was somewhat in excess of 10 minutes. Tests (b) and (c) show an approximately identical behaviour, with comparable average times but with larger variations between timesteps. This means that there is no net performance drop in going from a standard MPI system to a grid– and Chelonia–storage–based implementation, but that GaMPI is more vulnerable to e.g. the load on the clusters or the network connections. Test (c) proves that the simulations can also be run over a wide area network without a major performance hit. For DMC–style MC calculations, a very significant fraction of the process time is spent doing pure calculations, meaning that changing the interprocess communication protocol should not significantly impact the absolute performance. This is consistent with the results in this section.

#### 3.2. Scaling with number of CPUs

The second test series involved scaling up the number of CPUs, or in our case parallel grid jobs, used. The baseline, called case (d) in Table 1, is similar to case (c) of the first test, with 16 parallel processes on three computing clusters connected via ARC and Chelonia. However, to tax the systems we here increased the number of random walkers to 60k. Cases (e) and (f) then employed 32 and 60 parallel processes respectively.

Results are shown in Table 1, where we give average time per timestep (fourth column), as well as maximum and minimum times for a single step. The results are also illustrated in Figure 3 (left figure) for a more visual comparison.

While the variations between individual steps are still large, as expected on heterogeneous systems running steps with a varying number of walkers, there is a systematic decrease in average times with the number of processes. This comes from each process having to process fewer walkers per timestep. The total simulation time, \( T = t_{\text{average}} \cdot N_{\text{steps}} \), was reduced from 19 to 10 hours when going from 16 to 60 parallel processes.
3.3. Scaling the system size

For the third test we performed a scaling from small to large systems, in our case meaning changing the number of random walkers in the simulation. To keep our performance metric meaningful, we also varied the number of processes but kept the ratio $N_{\text{walkers}}/N_{\text{CPUs}}$ approximately constant. Both the test cases and results are presented in Table 1 and in Figure 3. Note that in our largest system (j), we have used a different setup for Chelonia than in the other cases. Connecting more than 60 clients to one single Chelonia instance leads to a performance drop, which is being addressed independently of the present study. One solution here is simply to split the walker pool into several partitions and have each partition be hosted by a separate Chelonia instance. The results from system (j) was obtained by using three partitions hosted by three separate Chelonia instances on one desktop machine, with 40 grid jobs connecting to each partition.

Figure 2 gives a graphic illustration of going from 30k (top row) to 120k (bottom row) walkers. The left plots show the gaussian initial state of the DMC system, the middle plots show the 25th timestep and the right plots show the walker positions after the 50th timestep. In all cases it is even visually clear that the increased number of walkers in the simulation will lead to a better determination of the ground state energy, which is the goal of the exercise. See ref. [10] for more details.

4. Conclusions

A diffusion Monte Carlo simulation has been successfully run on a worldwide computing grid, using a combination of the ARC grid middleware, Chelonia storage and the Ganga grid job control interface (GaMPI).

We have shown not only that it is technically feasible to run a large set of complex MC
simulations that require inter–process communication in this environment, but also that there is no appreciable performance loss associated with going from MPI communications to a grid–based storage, for a system that can be appropriately modularized. When scaling to larger numbers of parallel CPUs we see a smooth decrease in the time spent per timestep for a given simulation. Also we see no unexpected performance issues when scaling to more complex simulations, beyond what is expected from the increased need for inter-process communication.

The goal of this exercise was to run a DMC simulation on the grid, either going beyond the number of random walkers feasible on a 16 CPU cluster in a reasonable timeframe, or making each timestep of the simulation more efficient. We have shown that both can be done. Having proven the concept of running MPI–type simulations on a grid, it is our intention to go on to use the GaMPI setup for studies of more realistic physical systems.

References

[1] Ellert M, Gronager M, Konstantinov A, Konya B, Lindemann J, Livenson I, Nielsen J, Niinimaki M, Smirnova O and Waanaanen A 2007 Future Generation Computer Systems 23 219 – 240 ISSN 0167-739X URL http://www.sciencedirect.com/science/article/B6V06-4KGG1GK-1/2/f023d436acac58c2efba117f6c56c61
[2] Kleist J 2010 Presentation at CHEP2010
[3] Nilsen J K, Toor S, Nagy Z, Mohn B and Read A L 2010 Submitted to proceedings of CHEP2010
[4] Nagy Z, Nilsen J K, Toor S and Mohn B 2010 ACC CYFRONET AGH, Krakow Proceedings of the Cracow Grid Workshop 2009
[5] Moscicki J, Brochu F, Ebke J, Egede U, Elmsheuser J, Harrison K, Jones R, Lee H, Liko D, Maier A, Muraru A, Patrick G, Pajchel K, Reece W, Samset B, Slater M, Soroko A, Tan C, van der Ster D and Williams M 2009 Comp. Phys. Comm. 180 2303–2316 ISSN "0010-4655" URL "http://www.sciencedirect.com/science/article/B6JTB-4WJHB93-2/2/4d539f23d8c114b9b18d5447c7dc24943"
[6] Elmsheuser J 2010 Presentation at CHEP2010
[7] Nilsen J K, Cai X, Hyland B and Langtangen H P 2010 "Computational Science & Discovery" 3 015003 URL http://stacks.iop.org/1749-4699/3/i=1/a=015003
[8] Karohis N T, Toonen B and Foster I Mpich-g2: A grid-enabled implementation of the message passing interface
[9] Czarnul P 2008 PPAM’07: Proceedings of the 7th international conference on Parallel processing and applied mathematics (Berlin, Heidelberg: Springer-Verlag) pp 271–280 ISBN 3-540-68105-1, 978-3-540-68105-2
[10] Nilsen J K 2007 Comp. Phys. Comm. 177(10) 799–814