Mixed Mode computation in CASINO

Lucian Anton[1,*], Dario Alfe[2,†]

[1] *Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK*

[2] *Department of Earth Sciences, and Department of Physics and Astronomy, University College London, Gower Street, London,WC1E 6BT, United Kingdom*

May 10, 2010

Abstract

CASINO is a quantum Monte Carlo code that solves the many particle Schroedinger equation with the help of configurations of random walkers. This method is suitable for parallel computation because it has a very good computation/communication ratio. The standard parallel algorithm increases the computation speed by distributing equally the configurations among the available processors. For a computation with P processing elements the computation time for N_c configurations is proportional with $N_c t_c/P$, where t_c is the average time taken for one configuration step. On petascale computers one can have more processing elements than configurations and besides that for models with more that 1000 electrons t_c increases significantly. We present a mixed mode implementation of CASINO that takes advantage of the architectures with large numbers of multicore processors to improve computation speed by using multiple OpenMP threads for the computation of each configuration step.

Keywords: Quantum Monte Carlo, OpenMP, mixed mode parallelism.

1 Introduction

CASINO [1] is a quantum Monte Carlo code suitable for computation in quantum many body models with a large number of particles as it uses algorithms that scale algebraically with the system size. In a previous paper [2] we gave a brief description of the algorithms and general code architecture, extended documentation is available in Refs [3, 4].

CASINO can be used efficiently on parallel computers because it can run the computation on sets of independent configurations distributed over the available number of processing elements. Therefore, for a model with N_e electrons the computation time is approximated by

$$T_{CPU} \approx t_c \frac{N_c}{P} \ , \tag{1}$$

$$t_c \approx N_e^{\alpha} \ , \tag{2}$$

where N_c is the average number of configurations, P is the number of MPI tasks used, t_c is the time needed per configuration with $\alpha = 2$ for the models studied in this paper.

Nonetheless the current parallel algorithm used by CASINO cannot use optimally the top performance parallel computer systems available today for the following reasons:

- the current parallel algorithm can use at most N_c processing elements which for very large parallel systems could be significantly less that the total processing capacity,
- the computation of each configuration step is serial, hence t_c may increases significantly for models with large N_e ,
- the previous problem is amplified by the fact that the clock rate of the multicore modern processors is decreasing slightly and in the computational throughput per processing el-

*email: lucian.anton@nag.co.uk

†email: d.alfe@ucl.ac.uk

ement might decrease further due to technological constrains (the so called power wall, see e.g. Ref [5]).

In this paper we describe the second level parallelism (SLP) addition to CASINO algorithm which corrects the deficiencies listed before by offering the possibility to compute in parallel the configuration's time steps with OpenMP threads.

The paper is organised as follow. In Section 2 we present the guiding ideas of the SLP implementation and the performance analysis of the implemented algorithm on three models. In Section 3 we present a detailed analysis of SLP scaling breakdown which was observed for a particular matrix operation. General conclusions are presented in Section 4.

2 Second Level Parallelism

The results presented in this paper are obtained for models described by the following trial wavefunction

$$\Psi(\alpha, \mathbf{R}, \mathbf{R}_I) = e^{J(\alpha,\mathbf{R},\mathbf{R}_I)} D_\uparrow(\mathbf{r}_1, \ldots, \mathbf{r}_{N_\uparrow}) \times D_\downarrow(\mathbf{r}_{N_\downarrow+1\ldots,\mathbf{r}_{N_e}}) \ , \quad (3)$$

where $D_{\uparrow,\downarrow}$ are the Slater determinants of the electrons with spin up($\uparrow$) or down($\downarrow$) while the Jastrow function $J(\alpha, \mathbf{R}, \mathbf{R}_I)$ adds the electron-electron and electron-ion correlation whose parameters are determined by a variational Monte Carlo computation [4].

The SLP is implemented using OpenMP threads for the set of loops whose number of iterations scales with the number of electrons or ions. Since in the computation of a configuration time step the scaling with the number of electrons or ions appears only through the length of these loops it follows that the fraction of computation done in SLP increases with the number of electrons or ions of the model.

From the analysis of the source code the following sectors were identified to scale with the number of particles and consequently computed in SLP:

- one particle orbitals (OPO), which computes the one particle orbital value from a grid of B-spline coefficients,
- Jastrow (JAS), which computes the value of the Jastrow factor,
- Ewald summation (EWA) needed for the computation of the electrostatic energy,
- evaluation of the matrix $\bar{D}$ used for the fast computation of the Slater determinant,
- electron-electron relative distances R_{ee}.

The percentage of computing time spent in each selected sector by the original algorithm is shown in Fig 1 where one can see also that the percentage of the total time spend in these sectors increases with the number of electrons.

The Figures 2, 3 and 4 show the scaling of the computation time function of number of threads for the SLP sectors. In this study we have used three similar models containing 1024, 1536 and 2400 electrons. The time is normalised to the time taken by the respective sector in the original code for each compiler. The data were collected for executable produced by PGI 10.0.0 and GNU 4.4.2 compilers[1] on the quadcore AMD Barcelona processor [6] which is used in the phase 2a of HECToR [8].

The results show that the sectors with the simplest code structure, "EWA" and "R_{ee} " have a good scaling for all models. A baffling feature observed in data are several large variation (about 20%) of the computation time between the original code and the SLP code with one thread. These overhead times change significantly also with the compiler although the original source code was not changed beside the introduction of the OpenMP directives and a few local scalar variables. A constant negative feature is the loss scaling for the $\bar{D}$ sector for models with 1536 and 2400 electrons which occurs for both compilers when the number of threads is doubled from 2 to 4. This problem is analysed further in Sec 3.

The SLP aggregate performance gain in a realistic computation was measured using an equilibrated population of configurations with a 100 MPI tasks for 10 DMC block steps and 1,2, 4 SLP threads. Fig 5 shows that for the model with 1024 electrons the speed up factors are close to 1.5 and 2 for SLP with 2 and 4 OpenMP threads respectively.

3 On the scaling loss in "$\bar{D}$" sector

The sector " $\bar{D}$" performs the following operations on the cofactor matrix $\bar{D}$ for a given electron index i [9]

[1] Although the Pathscale compiler generates the fastest CASINO code, see http://www.hector.ac.ukcse/reports/compilers.php, the version installed on HECToR fails to compile with the OpenMP flag modules that contain THREADPRIVATE directives for module variables.

$$\bar{D}_{kj} = \begin{cases} \bar{D}_{kj}/q_{ii} & \text{if } j = i \\ \bar{D}_{kj} - \frac{q_{ji}}{q_{ii}}\bar{D}_{ki} & \text{if } j \neq i \end{cases} \quad (4)$$

$$q_{ji} = \sum_{l=1}^{N_e} \bar{D}_{lj}\phi_l(\mathbf{r}_i) \ , \quad (5)$$

where $\phi_j(\mathbf{r}_i)$ are one particle orbitals used to compute the Slater determinants.

The calculation described by Eq 4 was implemented in a small test program for a detailed measurement of the computation time scaling with respect to the number of threads for various sizes of matrix $\bar{D}$. Fig 6 (left) shows that the computation time scales with the number of threads up to a matrix size close to 700. A matrix with this linear size needs approximately 4 MB of memory to store its elements (in double precision) which is also the amount of available cache memory ($4L_2 + L_3$) on the Barcelona processor [6]. This indicates as a possible cause for the scaling breakdown the lack of memory bandwidth to move data concurrently between the cache memory of all four cores and the main memory.

A hardware performance counter analysis confirms this hypothesis. We have found that the values of hardware counters for the cache misses and translation look aside buffer do not change significantly with the matrix size over the whole range of the study, instead the stalled instructions counts increases with more than 20% for matrices with linear size larger than 700.

The test program was also run on a Magny-Cours processor [7], which will be used in phase 2b of HECToR. The scaling breakdown occurs for larger matrices but still in agreement with the cache fill argument, see Fig 6 (right).

4 Conclusion

We have studied the performance increase obtained by adding the SLP to CASINO for computations on models with a large number of electrons. The numerical intensive loops whose lengths increase with the number of particles were identified, implemented in parallel with OpenMP threads, and their performance measured over a range of models.

This study shows that using the SLP in CASINO the computation speed can be improved with a factor of ≈ 1.5 or ≈ 2 if 2 or 4 threads are used respectively in the SLP for a model with 1024 electrons.

On the negative side we have found that although the fractions of operations done in parallel with SLP grows with the number of electrons the computation time does not scale better, as one would expect from the Amdahl's law, because the computation speed of certain SLP sectors become memory bound if the number of electrons in the model increases beyond a certain limit. Also it was found that significant overheads (up to $\pm 20\%$) are introduced by the OpenMP directives in some sectors.

The analysis done in Sec 3 predicts that a good speed up can be achieved on Magny-Cours processors for models containing up to approximately 2000 electrons. Nevertheless, the results of this study and the current hardware trends point to the necessity consider new algorithms for some numerical intensive sectors of CASINO if SLP is to be used for models with even a larger number of electrons.

Acknowledgements

The authors acknowledge HECToR - a Research Councils UK High End Computing Service. LA thanks Kevin Roy for useful discussions.

References

[1] `http://www.tcm.phy.cam.ac.uk/~mdt26/casino2.html`

[2] L. Anton, D. Alfè R. Q. Hood and D. Tanqueray, CUG 2009 proceedings, `http://www.cug.org/5-publications/proceedings_attendee_lists/CUG09CD/`.

[3] Richard Needs, Mike Towler, Pablo Lopéz Ríos, CASINO User's Guide (Theory of Condensed Matter Group, Cavendish Laboratory, Cambridge, UK, 2008).

[4] M. D. Towler, Phys. Stat. Sol. (B) **243**, No. 11, 2573 (2006).

[5] `www.darpa.mil/Docs/ExaScale_Study_Initial.pdf`

[6] The AMD Barcelona quadcore processor has the following technical specifications: 2.3 GHz clock rate, 8 GB of RAM, 64KB L1 cache, 512 KB L2 cache, 2 MB L3 cache(shared), peak performance close to 40 Gflops in double precision.

[7] The AMD Magny-Cours 12 core processor has the following technical specifications: 2.1 GHz clock rate, 16 GB of RAM, 64KB L1 cache, 512 KB L2 cache, 6 MB L3 cache(shared).

[8] `http://www.hector.ac.uk`

[9] S. Fahy, X. W. Wang and Steven G. Louie, Phys. Rev. B **42** 3503 (1990).

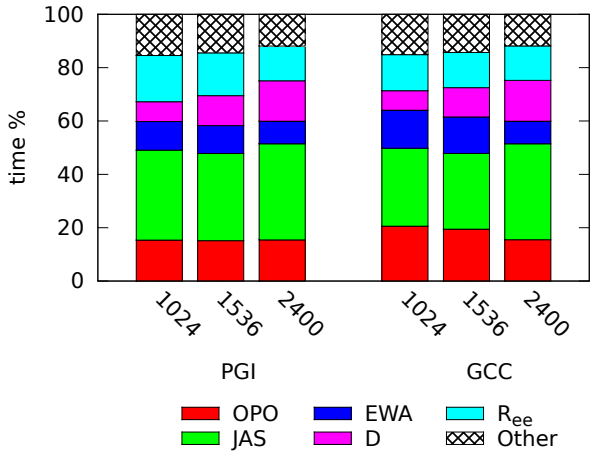

Figure 1: Percentage of time spent by the initial algorithm in the computational sectors described in text for models with 1024, 1536 and 2400 electrons.

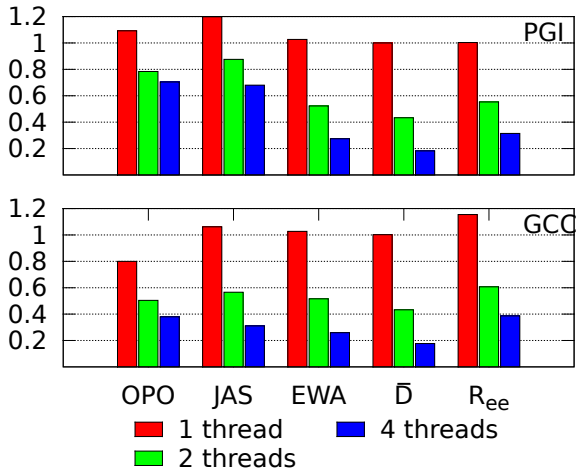

Figure 2: The computation time function of the number of threads for each SLP sector for a model with 1024 electrons. Time is normalised to the time taken by an identical run of an executable compiled without OpenMP support.

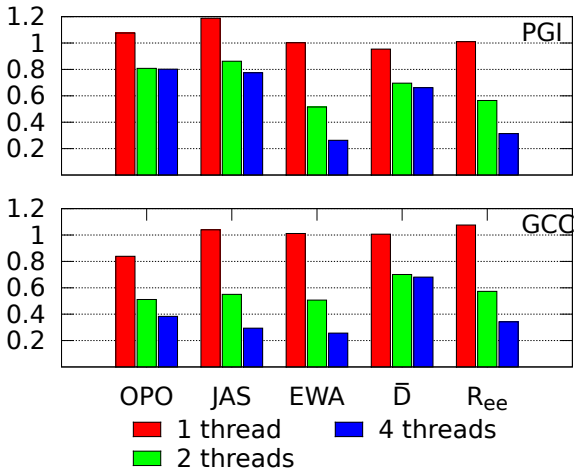

Figure 3: As in Fig 2 for a model with 1536 electrons.

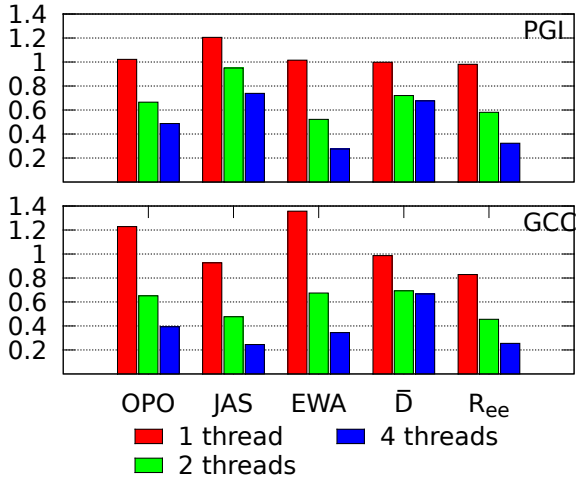

Figure 4: As in Fig 2 for a model with 2400 electrons.

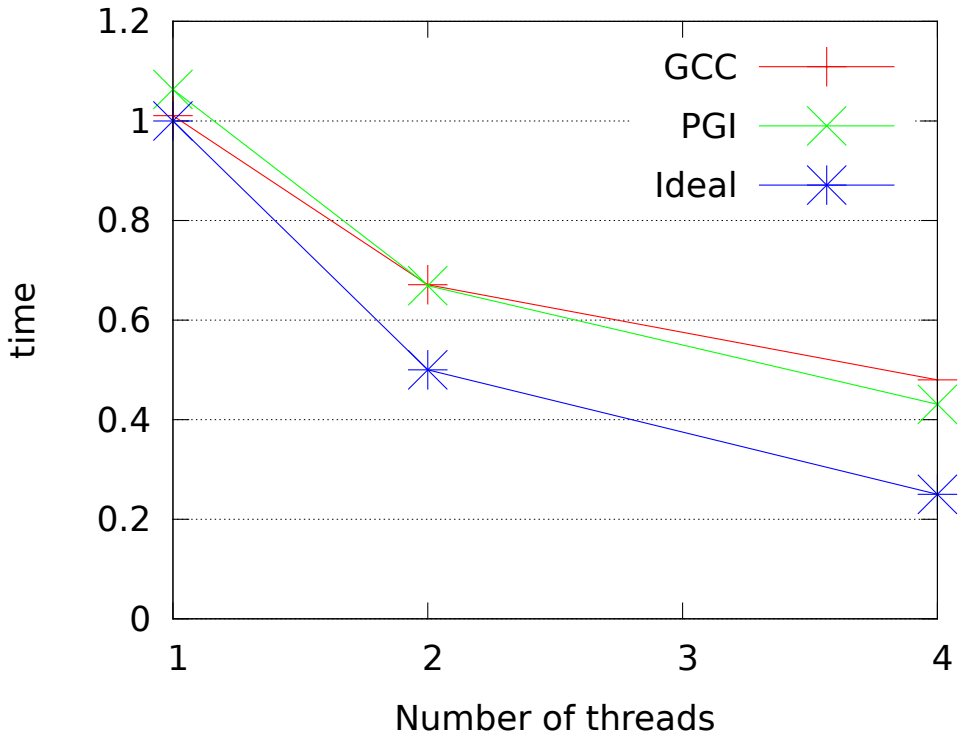

Figure 5: The computation time with SLP support using 1, 2, 4 OpenMP threads, 1024 electrons model. The time is normalised to the computation time of the same run done by an executable compiled without OpenMP (PGI compiler).

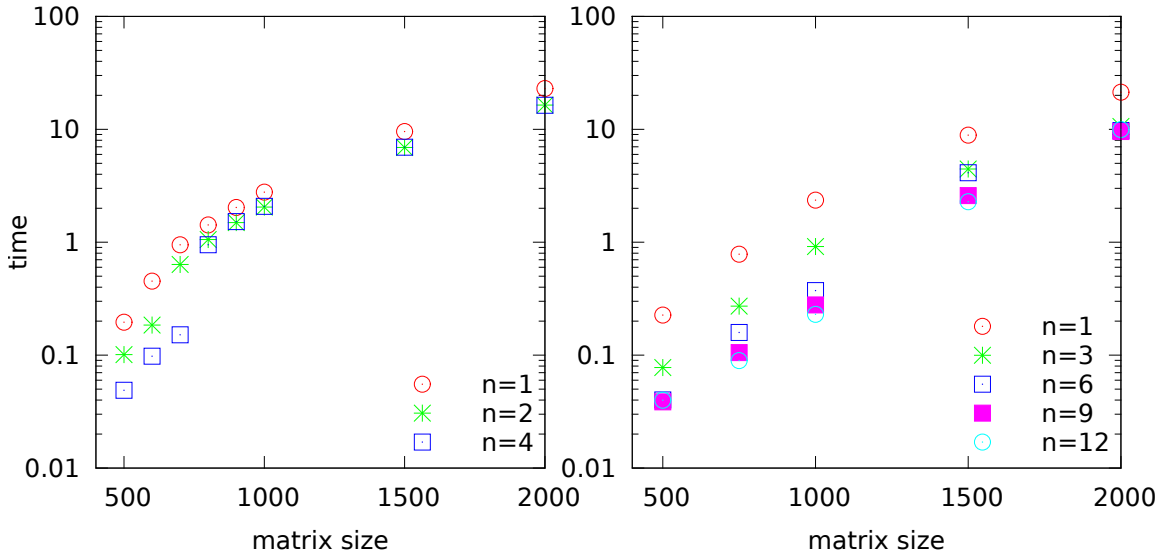

Figure 6: Scaling of computation time for Eq 4 done in parallel with OpenMP on the quadcore processor with $n = 1, 2, 4$ threads, left panel, and for the twelve-core with $n = 1, 3, 6, 9, 12$ threads, right panel.