ANL/ALCF/ESP-13/13

Ab-initio Reaction Calculations for Carbon-12 (ESP Technical Report)

ALCF-2 Early Science Program Technical Report

Argonne Leadership Computing Facility

ANL/ALCF/ESP-13/13

Ab-initio Reaction Calculations for Carbon-12 (ESP Technical Report)

ALCF-2 Early Science Program Technical Report

prepared by
Alessandro Lovato[1] and Steven C. Pieper[2]
[1]Argonne Leadership Computing Facility, Argonne National Laboratory
[2]Physics, Argonne National Laboratory

May 7, 2013

ESP technical report

Alessandro Lovato

*ALCF and Physics division, Argonne National Laboratory**

Steven C. Pieper

Physics division, Argonne National Laboratory

(Dated: April 3, 2013)

* lovato@alcf.anl.gov

I. DESCRIPTION OF SCIENCE

The electroweak response is a fundamental ingredient to describe the neutrino - ^{12}Carbon scattering, recently measured by the MiniBooNE collaboration to calibrate the detector aimed at studying neutrino oscillations. As a first step towards its calculation, we have computed the sum rules for the electromagnetic response of ^{12}C. The cross section of the process

$$e +^{12} \mathrm{C} \to e' + X \,. \tag{1}$$

can be written in Born approximation as [1]

$$\frac{d^2\sigma}{d\Omega_{e'} dE_{e'}} = -\frac{\alpha^2}{q^4}\frac{E_{e'}}{E_e} L_{\mu\nu} W^{\mu\nu} \,, \tag{2}$$

where $\alpha \simeq 1/137$ is the fine structure constant, $d\Omega_{e'}$ is the differential solid angle specified by $\mathbf{k}_{e'}$ and $q = k_e - k_{e'}$ is the four momentum transfer of the process. The leptonic tensor $L_{\mu\nu}$ is fully determined by the measured kinematical variables of the electron, while all information on target structure, which is largely dictated by nuclear interactions, is enclosed in the hadronic tensor

$$W^{\mu\nu} = \sum_X \langle \Psi_0 | J^\mu | \Psi_X \rangle \langle \Psi_X | J^\nu | \Psi_0 \rangle \delta^{(4)}(p_0 + q - p_X) \,. \tag{3}$$

The sum over the final states includes an integral over $\mathbf{p}_X$, the spatial momentum of the final hadronic state, while p_0 is the initial four-momentum of the nucleus.

In the nonrelativistic approach, the hadronic tensor can be written in terms of the longitudinal and transverse response functions, with respect to the direction of the three-momentum transfer $\mathbf{q}$. For instance, taking $\mathbf{q}$ along the z-axis, the transverse response is defined by [2]

$$R_{xx+yy}(\mathbf{q},\omega) = \sum_X \delta(\omega + E_0 - E_X) \Big[\langle \Psi_0 | j^x(\mathbf{q},\omega) | \Psi_X \rangle \langle \Psi_X | j^x(\mathbf{q},\omega) | \Psi_0 \rangle + \langle \Psi_0 | j^y(\mathbf{q},\omega) | \Psi_X \rangle \langle \Psi_X | j^y(\mathbf{q},\omega) | \Psi_0 \rangle \Big] \tag{4}$$

while the longitudinal is given by

$$R_{00}(\mathbf{q},\omega) = \sum_X \delta(\omega + E_0 - E_X) \langle \Psi_0 | \rho(\mathbf{q},\omega) | \Psi_X \rangle \langle \Psi_X | \rho(\mathbf{q},\omega) | \Psi_0 \rangle \tag{5}$$

The *sum rules* are obtained integrating the response functions over the energy transfer and using the completeness relation of the states $|X\rangle$. For R_{xx+yy} and R_{00} one has

$$\begin{aligned} S_{xx+yy}(\mathbf{q}) &\equiv \int d\omega R_{xx+yy}(\mathbf{q},\omega) = \langle \Psi_0 | j^x(\mathbf{q},\omega_{el}) j^x(\mathbf{q},\omega_{el}) + j^y(\mathbf{q},\omega_{el}) j^y(\mathbf{q},\omega_{el}) | \Psi_0 \rangle \\ S_{00}(\mathbf{q}) &\equiv \int d\omega R_{00}(\mathbf{q},\omega) = \langle \Psi_0 | \rho(\mathbf{q},\omega_{el}) \rho(\mathbf{q},\omega_{el}) | \Psi_0 \rangle \,, \end{aligned} \tag{6}$$

where the energy transfer dependence of the current and density operators is determined at the the quasi-elastic peak: $\omega_{el} = \sqrt{|\mathbf{q}|^2 + m^2} - m$. Hence, the sum rules of the response can be evaluated by computing the expectation values of the electromagnetic currents and density on the ground state of ^{12}C.

II. NUMERICAL METHODS

The calculation of the sum rules requires the knowledge of the nuclear ground state wave-function of ^{12}C. Solving the many-body Schroedinger equation

$$\hat{H}\Psi_0(x_1 \ldots x_A) \ = E_0\Psi_0(x_1 \ldots x_A)\,, \tag{7}$$

where the generalized coordinate $x_i \equiv \{\mathbf{r}_i, s_i, t_i\}$ represents both the position and the spin-isospin variables of the i-th nucleon, is made particularly difficult by the complexity of the interaction. The nuclear potential is indeed spin-isospin dependent and contains strong tensor terms; thus Eq. (7) consists in $2^A\binom{A}{Z}$ complex coupled second order partial differential equations in $3A$ variables. For the actual case of ^{12}C, there are 270,336 coupled equations in 36 variables.

Standard methods for solving partial differential equations are not feasible in this context. Green Function Monte Carlo (GFMC) algorithms use projection techniques to enhance the true ground-state component of a starting trial wave function Ψ_T

$$\Psi_0(x_1 \ldots x_A) = \lim_{\tau\to\infty} e^{-(\hat{H}-E_0)\tau}\Psi_T(x_1 \ldots x_A)\,. \tag{8}$$

In the actual calculation, the imaginary time evaluation is done a sequence of imaginary time steps, each one consisting in a $3A$ dimensional integral, evaluated within the Monte Carlo approach.

In GFMC all the spin-isospin configurations are considered and the wave-function is a vector of $2^A\binom{A}{Z}$ complex numbers. For example the eight spin configurations of the ^{3}H nucleus are represented by [3]

$$|\Psi_{^3H}\rangle = \begin{pmatrix} a_{\uparrow\uparrow\uparrow} \\ a_{\uparrow\uparrow\downarrow} \\ a_{\uparrow\downarrow\uparrow} \\ a_{\uparrow\downarrow\downarrow} \\ a_{\downarrow\uparrow\uparrow} \\ a_{\downarrow\uparrow\downarrow} \\ a_{\downarrow\downarrow\uparrow} \\ a_{\downarrow\downarrow\downarrow} \end{pmatrix} \tag{9}$$

Each coefficient a_α, which is a function of the coordinates $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$, represents the amplitude of a given many-particle spin configuration; for instance

$$a_{\uparrow\uparrow\downarrow} = \langle\uparrow\uparrow\downarrow|\Psi_{^3H}\rangle\,. \tag{10}$$

The application of the spin matrix $\sigma_{12} \equiv \sum_i \sigma_1^i \sigma_2^i$ yields

$$\hat{\sigma}_{12}|\Psi_{^3H}\rangle = \begin{pmatrix} a_{\uparrow\uparrow\uparrow} \\ a_{\uparrow\uparrow\downarrow} \\ 2a_{\downarrow\uparrow\uparrow} - a_{\uparrow\downarrow\uparrow} \\ 2a_{\downarrow\uparrow\downarrow} - a_{\uparrow\downarrow\downarrow} \\ 2a_{\uparrow\downarrow\uparrow} - a_{\downarrow\uparrow\uparrow} \\ 2a_{\uparrow\downarrow\downarrow} - a_{\downarrow\uparrow\downarrow} \\ a_{\downarrow\downarrow\uparrow} \\ a_{\downarrow\downarrow\downarrow} \end{pmatrix} \tag{11}$$

The "new" wave function can be expressed in terms of the coefficients of the old one. Therefore, in order to reduce the computational complexity of the spin and isospin matrix multiplication, a specialized table-drive code is implemented.

III. BEFORE MIRA AND ON MIRA

The GFMC code needed to be deeply revised to better capitalize the resources of a leadership class computer like Intrepid (BQP) and Mira (BGQ).

The branching process of the GFMC algorithm involves replication and killing of the samples, the number of which can undergo large fluctuations. Therefore, to achieve an high

efficiency, the old version of the code did several Monte Carlo samples, say at least 10, per processor. However, a typical ^{12}C calculation involves around 15,000 samples while leadership class computers have many 10,000's of processors, making the old algorithm quite inefficient.

Fortunately, for nuclei as large as ^{10}B and ^{12}C, the calculation of the energy and of the response is complex enough to allow for splitting one sample over many processors. To this extent, the general purpose Automatic Dynamic Load Balancing (ADLB) library [4], was developed on Intrepid and implemented in the code.

Both the direct calculation of the response with the $\Psi_{1^+,T=1}$ state and the evaluation of the sum rules would have not been possible on Intrepid, due to the limited amount of RAM per node (2GB). On the other hand, Mira, with 16 GB of RAM per node enables us to perform such a large calculations.

We were pleased to find that the version of ADLB developed for Intrepid works very well on Mira; no modification was required. The conversion to Mira consisted primarily of timing the OpenMP (OMP) sections of the GFMC code to work well up to 64 threads and developing the new subroutines for the response and for the sum rules.

IV. THE CODE

The scheme of ADLB, illustrated in Fig. 1, shows that the nodes are organized in *servers* and *slaves*; in standard GFMC calculations approximately 3% of the nodes are ADLB servers. A shared work queue, managed by the servers, is accessed by the slaves that either *put* work units, denoted as "work packages" in it or *get* those work package out to work on them. Once a work package has been processed by a slave, a "response package" may be sent to the slave that put the work package in the queue.

ADLB is a general purpose library, which hides communication and memory management from the application, providing a simple programming interface. Besides the initialization and termination functions, the truly essential function calls of the ADLB application programmer interface (API) are the `ADLB_Put`, `ADLB_Reserve` and `ADLB_Get_reserved`. To better illustrate these three function calls, it is worth showing the explicit case of the sum rules subroutines.

The expectation value of Eq. (6) has to be evaluated for momentum transfer directed along x, y and z axis. In each of these cases, $\sim$ 20 values of the discretized momentum transfer magnitude are considered; hence for each configuration $\sim$ 60 independent expectation values

have to be computed. Since the evaluation of the sum rules of the ^{12}C for a single value of $\mathbf{q}$ takes of about 100 seconds (with 32 OMP threads), we decided to split the calculation in such a way that each ADLB slave calculates the sum rules for a single value of $\mathbf{q}$.

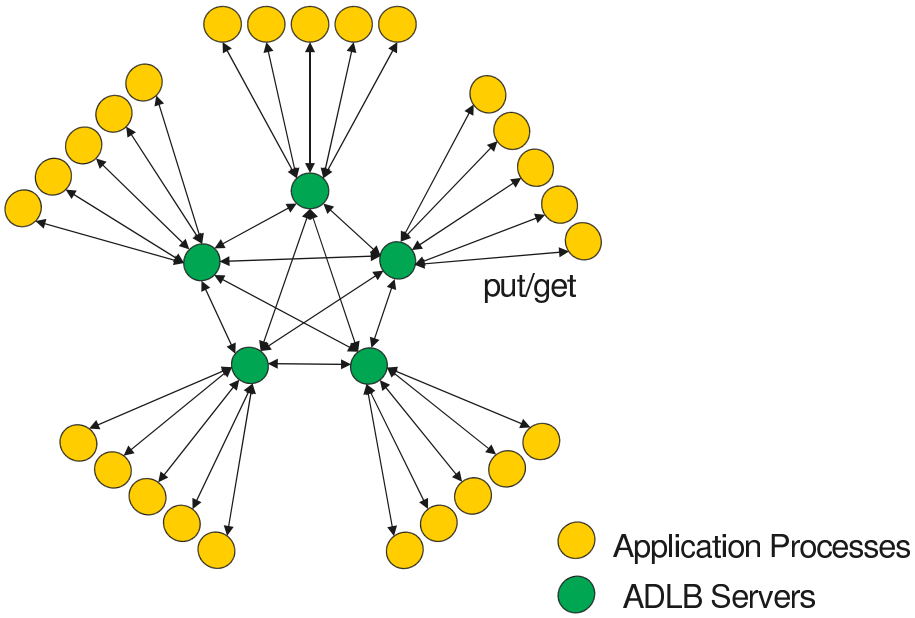

Figure 1. Automatic Dynamic Load Balancing work flow.

- `subroutine o_em_wk`

  Let us concentrate on a particular ADLB energy slave, managing a single configuration. It enters `o_em_wk` and immediately puts into the work pool the part of work package independent on $\mathbf{q}$

  ```
  call ADLB_Begin_batch_put (rwp%cfl,respon_wp_len_common,ierr)
  ```

  where `rwp%cfl` indicates the beginning of the work package, `respon_wp_len_common` denotes its size and `ierr` will get a return code.

  Afterwards, the $\mathbf{q}$ dependent parts of the work packages are placed in the work pool for each of the ~ 60 cases.

  ```
  call ADLB_PUT(rwp%qh,respon_wp_len_var,-1,myid, adlbwp_respon,i_prior,ierr)
  ```

The size of the **q** dependent part of the work package is specified by `rwp%qh,respon_wp_len_var`, while `myid` identifies the energy slave from which the work package originates.

As a matter of fact, the work packages can be processed either by the same ADLB energy slave which put them in the work pool or by another one. However, only the slave that sent the work package can retrieve the corresponding response package by means of the following call

```
call get_adlb_respon_work_ans ( ierr, node )
```

which is iterated until all the response packages have been collected.

- `get_adlb_respon_work_ans`

To retrieve a unit of work, the energy slave uses this subroutine to call `ADLB_reserve`

```
call ADLB_reserve ( (/ adlbwp_respon_ans, adlbwp_respon, -1 /),  &
  &   i_wrk_type, i_prior, i_handle, i_len, i_answer, ierr )
```

specifying that it is looking either for a work package or for a response package. If either one is present, ADLB will find it and send back a handle (`i_handle`) , a global identifier (`i_wrk_type`) along with the size of the reserved work unit (`i_len`) and the origin identifier (`i_answer`). The priority of the answer is set much larger than the priority of the work packages, so that they will be preferentially returned.

If an answer has been found by ADLB, it is retrieved by

```
call ADLB_GET_RESERVED_TIMED ( rap, i_handle, qtime, ierr )
```

where `rap` denotes the response answer package and the energy slave returns into the subroutine `o_em_wk`.

If a work package is instead found by `ADLB_reserve`, the energy slave processes it. It has to be remarked that other ADLB slaves than the energy ones can process a work package. For this purpose an `entry` appears in the subroutine

```
entry process_adlb_respon_work (ii_prior,ii_handle,ii_len,ii_answer,ierr)
```

Analogously to what happens for the answer package, `ADLB_GET_RESERVED_TIMED` is called

```
call ADLB_GET_RESERVED_TIMED ( rwp, i_handle, qtime, ierr )
```

with `rwp` appearing as a first argument instead of `rap`.

Using the retrieved work package, the actual calculation of the sum rule is performed for a single value of **q**

```
call o_em_wk_q(rwp%iptb,rwp%if2,rwp%actf,rwp%q,rwp%qh,rwp%weight,  &
      &  rwp%rpart0,rwp%cfl,cfdl,rwp%cfr,cfdr,rap%fxtt_q,rap%fxll_q, &
      &  iqq,iqh,.false.)
```

If the work package answer is addressed to a different ADLB slave from the one that made the computation, `myid .ne. i_answer`, then the answer needs to be put in the work pool

```
call ADLB_PUT ( rap, respon_ans_len, i_answer, myid, adlbwp_respon_ans,&
       &  i_prior+1000, ierr)
```

Otherwise, the answer package is not put in the pool and the energy slave returns in the subroutine `o_em_wk`.

- `master_get_work`

  The main program continuously calls the subroutine `master_get_work` to look for work packages. These can be of any type (except answers). The appropriate subroutine is called to process the work and then `master_get_work` is used on each slave, again.

V. TUNING THE CODE AND PERFORMANCE ON MIRA

The conversion of the GFMC code from Intrepid to Mira did not show particular difficulties. The ADLB performance turned out to be even better on Mira than on Intrepid without modifications. Moreover, OpenMP scales well with the number of threads.

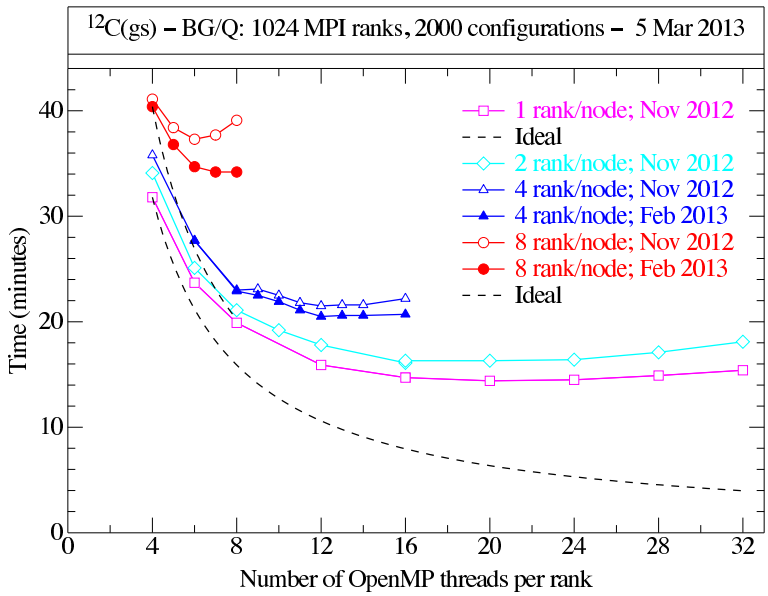

Figure 2. OMP strong scaling performance for ^{12}C GFMC calculation with 2048 configurations, 1024 MPI ranks: total wall time comparison.

A. OpenMP (OMP) strong scaling

First of all let us analyze the OMP scaling of the total wall time required to do a GFMC calculation for 2048 configurations of the ^{12}C ground state using 1024 MPI ranks, displayed in Figure 2. The different colours indicate the different number of ranks per node and hence the total number of nodes. As expected, the single rank per node case exhibits the best OMP scaling, as there are no threads associated with different ranks competing for memory on the same node. Due to the competition among the threads belonging to the same rank, the scaling saturates at about 20 threads, remaining fairly above the ideal case, represented by the dashed curve, where the wall time decreases as 1/(#OMP threads).

Since in every node there are at most 64 threads, keeping fixed the number of MPI ranks and increasing the number of threads results in a larger node usage. Thus, a more meaningful scaling test consists in studying the number of configurations processed by a single node with different combinations of ranks per node and threads per rank, keeping in mind that the product of these two quantities cannot exceed 64. The results of Fig. 3 show that, with the new driver installed in February 2013, the most efficient configuration is 8 ranks per node, 8 threads per rank. In this case a performance of 6.4 GFLOPS per node (about 3.1% of the

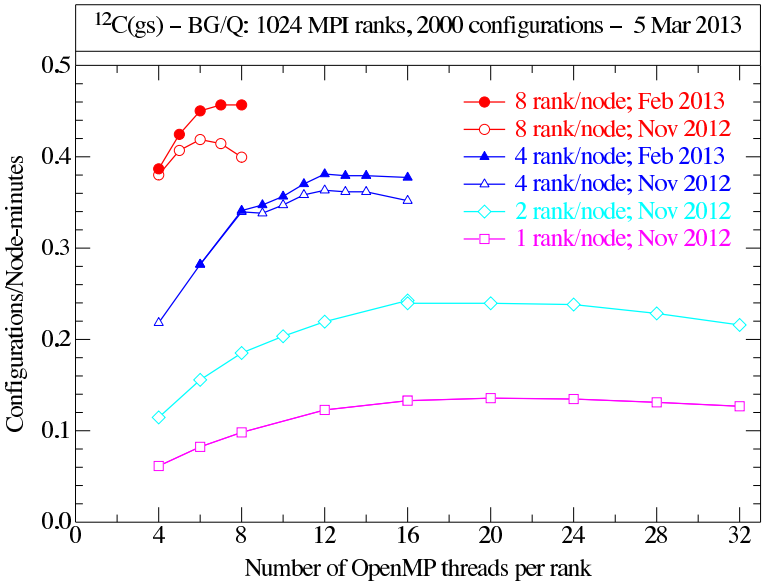

Figure 3. OMP strong scaling performance for ^{12}C GFMC calculation with 2048 configurations, 1024 MPI ranks: number of configurations per node per minute.

peak) is achieved. Is should be noticed that with the former version of the driver using more than 6 threads per rank resulted in worse performances. Finally, the limit of 8 ranks per node is dictated by memory requirements.

An analogous analysis, shown in Fig. 4 has been performed for the sum rules calculation with 32 MPI ranks. Due to the large size of the wave function derivative, not more than 1 rank per node can be used in the calculation; however if the derivatives are disregarded, 4 ranks per node can be used.

Because of large loops over the spin and isospin indices of the wave functions, OMP keeps improving up to 64 threads, although very slowly beyond 32 threads. However, as for the energy, while the minimum total wall time consumption is obtained with 1 rank per node and 64 threads per rank, the highest efficiency of about 12 GFLOPS per node is achieved with 4 ranks per node and 16 threads per rank.

B. ADLB weak scaling

The ADLB library was not significantly exercised in the results shown in the former section, as the number of MPI ranks was limited to 1024. By looking at Fig. 5, in which the total wall

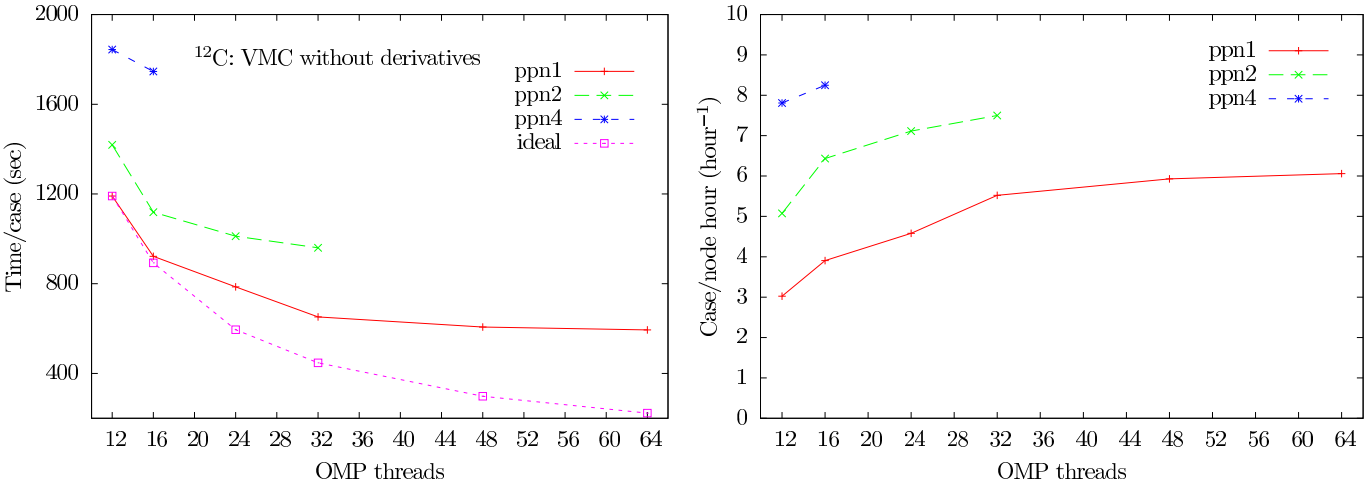

Figure 4. OMP strong scaling performance for sum rule calculation with 2 values of momentum transfer. The total wall time (left panel) and the case/node per hour (right panel) for 32 MPI ranks are shown.

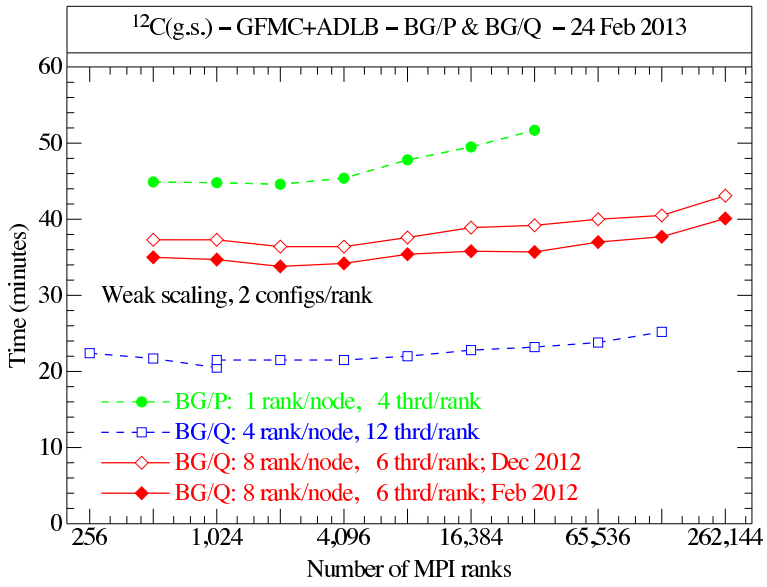

Figure 5. ADLB weak scaling performance for energy calculation with 2 configurations per rank: total walltime

time used for computing 2 configurations per MPI rank is plotted against the number of MPI ranks, it is possible to appreciate the improvements brought about by ADLB. A good scaling, fairly close to the ideal case of a straight and horizontal line, is shown up to 260,000 ranks, 524,688 cores, 1,572,864 threads. As for the OMP scaling, it is worth analyzing the scaling of

the configurations per node per minute, displayed in Fig. 5. Despite the smallest total wall time being consumed by using 4 ranks per node and 12 threads per rank, the most efficient configuration is the one with 8 ranks per node and 6 threads per rank.

Finally, it is interesting to notice that a Mira node is almost ten times faster than an Intrepid one.

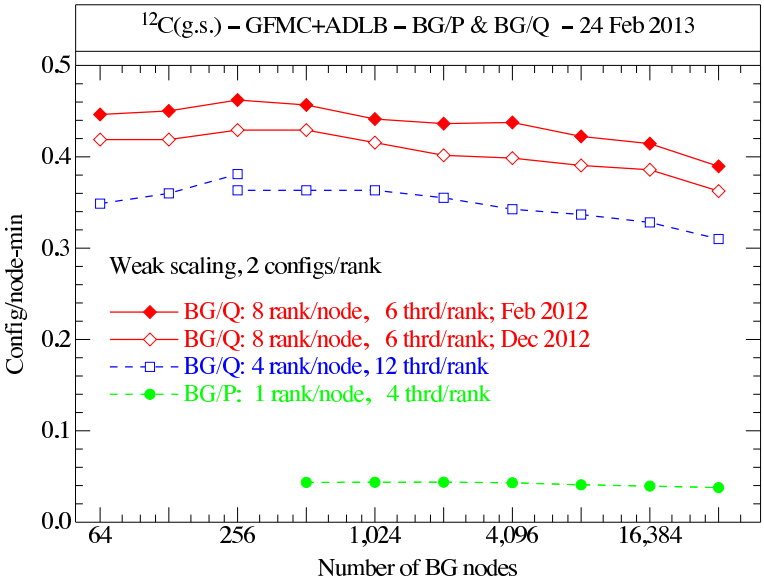

Figure 6. ADLB weak scaling performance for energy calculation with 2 configurations per rank: number of configurations per node per minute

VI. FIRST SUM RULE RESULTS

In Fig. 7 we show the preliminary results for the sum rule of the transverse electromagnetic response of ^{12}C, obtained neglecting the derivatives of the wave functions. It has been obtained by averaging over $\sim$ 1000 configuration for each of the 5 imaginary time values, $\tau = 0.8,\ 0.18,\ 0.28,\ 0.40,\ 0,48$, after the constrained path has been released.

The two-body currents have a prominent effect, relatively much larger than the difference between GFMC and VMC calculations, that at this level of accuracy, provide compatible results.

Many more configurations are needed for quantities defined in terms of differences between

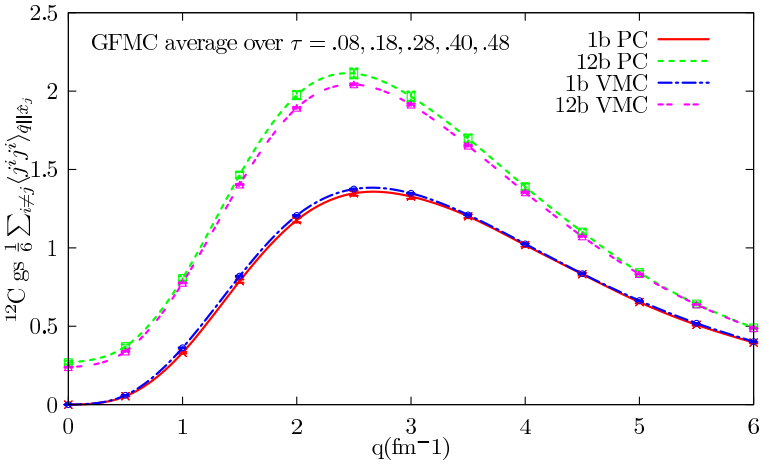

Figure 7. ^{12}C transverse sum rule for the electromagnetic transverse response.

large Monte Carlo estimates, such as the longitudinal Coulomb sum rule

$$S_L(\mathbf{q}) = \frac{1}{6[G_p^e(q^2)]^2}\Big[\langle\Psi_0|\rho(\mathbf{q},\omega_{el})\rho(\mathbf{q},\omega_{el})|\Psi_0\rangle - \langle\Psi_0|\rho(\mathbf{q},\omega_{el})|\Psi_0\rangle^2\Big] \tag{12}$$

where $G_p^e(q^2)$ is the proton electric form factor. We currently are in the production phase, and will soon have the necessary statistical significance, hopefully allowing us to predict the data of a recent Jefferson Lab experiment which is nearing publications.

This is not yet the end of the story: the subroutines for the sum rules of the weak response have been presently developed by the Los Alamos group and already tested in VMC calculations of small nuclei. We plan to implement them in the GFMC code and tune them for Mira in the very next months.

[1] Omar Benhar, Donal Day, and Ingo Sick. Inclusive quasielastic electron-nucleus scattering. *Rev. Mod. Phys.*, 80:189–224, Jan 2008.

[2] G. Shen, L.E. Marcucci, J. Carlson, S. Gandolfi, and R. Schiavilla. Inclusive neutrino scattering off deuteron from threshold to GeV energies. *Phys.Rev.*, C86:035503, 2012.

[3] Steven Pieper. Monte carlo calculations of nuclei. In Jesùs Navarro and Artur Polls, editors, *Microscopic Quantum Many-Body Theories and Their Applications*, volume 510 of *Lecture Notes in Physics*, pages 337–357. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0104530.

[4] Ralph M. Butler Ewing L. Lusk, Steven C. Pieper. More scalability, less pain:

A simple programming model and its implementation for extreme computing. http://www.cs.mtsu.edu/~rbutler/adlb/.

Argonne Leadership Computing Facility
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439

www.anl.gov