ANL/ALCF/ESP-13/6

Accurate Numerical Simulations Of Chemical Phenomena Involved in Energy Production and Storage with MADNESS and MPQC (Fast Linear Algebra Libraries in MADNESS: a Numerical Framework for Quantum Chemistry on Petascale Platforms)

ALCF-2 Early Science Program Technical Report

Argonne Leadership Computing Facility

ANL/ALCF/ESP-13/6

Accurate Numerical Simulations Of Chemical Phenomena Involved in Energy Production and Storage with MADNESS and MPQC
(Fast Linear Algebra Libraries in MADNESS: a Numerical Framework for Quantum Chemistry on Petascale Platforms)

ALCF-2 Early Science Program Technical Report

prepared by
Álvaro Vázquez-Mayagoitia[1], Jeff R. Hammond[1]
[1]Argonne Leadership Computing Facility, Argonne National Laboratory

May 13, 2013

Fast Linear Algebra Libraries in MADNESS: a Numerical Framework for Quantum Chemistry on Petascale Platforms

Álvaro Vázquez-Mayagoitia and Jeff R. Hammond

May 13, 2013

Abstract

In order to solve the electronic structure of large molecular systems on petascale computers using MADNESS, a numerical tool kit, are required fast and accurate implementations for linear algebra. MADNESS uses multiresolution analysis and low separation rank which translates high dimensional functions in tensor products using Legendre polynomial. The multiple tensor products make to the singular value decomposition and matrix multiplication the most intense operations in MADNESS. This work discusses the interfacing of Eigen3 as a C++ substitute of LAPACK and introduces Elemental for the diagonalization of large matrices. Furthermore, the present paper shows the performance these libraries on Blue Gene/ Q.

1 Introduction

Quantum chemistry has influenced many fields in science by revealing the structure of materials and molecules at atomic level, its most notable achievement is to predict accurately chemical and physical properties. The study of materials at atomic level is a complicated task, even very sophisticated experiments are challenged to reproduce observations at such level. In practice, to obtain values from quantum chemistry methods with meaningful precision requires a huge computational effort. Quantum mechanical methods as the so-called Density Functional Theory (DFT), that approaches the electronic

exchange and correlation, in average reproduces atomization energies within an error of 0.2 eV. Although the availability of large modern computers, DFT currently might compute molecular systems with only few thousands of electrons. [1]

On the other hand, along the last decades the theoretical chemistry has changed drastically in part due to the new developments and approximations to apply the quantum chemistry, and in part to the increment of calculation power. Thus, the unprecedented availability of petascale computers for fundamental research has required new software development according with the last architecture capacities, overall it has bee required software that exploits the technology of multi-cores and multi-processors using large blocks of distributed memory, and new codes that can be reusable. Thus, the synergy of modern quantum chemistry and supercomputing demands the production of new generations of codes able to utilize efficiently large computer systems, and desirably, be able to evolve at the same pace as the novel technologies emerge.

The software Multiresolution Adaptive Numerical Environment for Scientific Simulation (MADNESS) [2] is a general numerical framework for massive parallel computations. MADNESS was designed to reduce the programming effort offering a set of high level tools to solve many dimension integral-differential equations and maximize the science productivity by letting to the programmer to be focused in her or his application instead writing complex low level instructions. MADNESS has been successfully used for several applications in nuclear physics, chemistry, atomic physics, among other areas. MADNESS uses a multiresolution analysis (MRA) that relies on the low separation rank (LSR) representation for functions and operators which lead a generalization of one spatial dimension to higher dimensions and yields algorithms that are too costly for practical applications. The current implementation of MADNESS might operate with a large variety of kernels and boundary conditions. For quantum chemistry MADNESS has an implementation to solve the electronic structure problem with the methods DFT, [3, 4, 5] Hartree-Fock [6, 7] and MP2.[8] This software discretizes the orbital functions within an orthogonal basis sets constructed with Legendre polynomia which in conjunction with the LSR of the local potentials leads to solve linearly the electronic structure at a given arbitrary precision. [9]

This report will summarize part of the effort made to port MADNESS and make it efficient to the IBM Blue Gene/Q technology. The text will discuss briefly the most computational costly parts of the MADNESS calcu-

lations related with linear algebra operations and the interface to external algebra libraries in order to speedup the code. This text will avoid any deep discussion of the formulation of the approaches used and/or the hybrid parallel model found MADNESS, nevertheless we will address to the readers for further details in the correspondent references.

2 Methods

2.1 Numerical Methods Used

The MRA allows to represent a d-dimensional space in d-dimensional boxes, each box with a basis set formed as a of tensor product of Legendre polynomials. The LSR representation of a $3D$ function is written as follows

$$\phi(x,y,z) \approx \sum_{k_1,k_2,k_3}^{r} s_{k_1,k_2,k_3}\varphi_{k_1}(x)\varphi_{k_2}(y)\varphi_{k3}(z)$$

where $\varphi_i(i)$ is a set of orthogonal and polynomial functions and the coefficients s_{k_1,k_2,k_3} are scalar and are adjusted in an adaptive separation rank r to archive a threshold for the accuracy of ϵ that is given by the difference

$$||\phi(x,y,z) - \sum_{k_1,k_2,k_3}^{r} s_{k_1,k_2,k_3}\varphi_{k_1}(x)\varphi_{k_2}(y)\varphi_{k3}(z)||_2 \leq \epsilon,$$

some elements of the rank r may need a refinement to reach faster the required accuracy. This technique is similar in speed as Fast Fourier Transformation used in spectral algorithms on uniform grids. In the practice, the LSR representation in wavelets give us to formulate our functions as a tensor product

$$F = (((S^T D)^T_{k_1} D)^T_{k_2} D)_{k_3},$$

the indexes k_1, k_2, k_3 run over the number of subspaces in the rank used to represent the original function. The matrices D and S are matrices with the filter coefficients and scalar coefficients of the wavelets. We can anticipate that small matrix-matrix operations are the most intense operations in our calculations.

In quantum chemistry, the key equation to obtain the wavefunction of a time-independent system, composed by electrons and ions, is the Schrödinger

equation $\hat{H}\Psi = E\Psi$, where the electronic Hamiltonian $\hat{H}$ is the sum of the kinetic and potential operators $\hat{H} = \hat{T} + \hat{V}$. The numerical low-rank representation of the wavefunctions allows to solve the Schrödinger equation in an integral equation form as: [10, 11]

$$\Psi = -2 \cdot \hat{G}_{\mu}(V\Psi),$$

where $\mu^2 = (-2 * E)$, and E is the total energy of system. The wavefunction Ψ can be seen as the auxiliary Kohn-Sham wavefunction and can be applied for DFT. The integral operator $\hat{G}_{\mu}$ can be written as

$$(\hat{G}_{\mu} * f)(x) = \int \frac{e^{-\mu|x-x'|}}{4\pi|x-x'|} f(x')dx'.$$

Finally the problem is reduced to find a Ψ that minimizes the energy E, and this is solved iteratively since in this approach the energy is variational.

The application of the integral operator implies the convolution of the real-space functions in the LSR representation. The deconvolution operation is also computationally very demanding and requires the decomposition of the coefficients s_{k_1,k_2,k_3} of the tensor products in the low rank separation. decomposes s_{k_1,k_2,k_3} such as The decomposition of the coefficients s_{k_1,k_2,k_3} is the second most expensive operation and is performed using standard Single Value Decomposition algorithms (SVD). Thus the most intensive operations are applied to small matrices, this leads to implement very specific algorithms for those cases.

Additionally, the wavefunctions in the methods DFT and Hartree-Fock are chosen to be real wavefunctions. DFT and HF establishes a formalism of separable particle functions that permits to write the wavefunction of the system as a single Slater determinant $\Psi = \frac{1}{\sqrt{N!}}|\phi_1(r_1)...\phi_N(r_N)|$. In MADNESS, like in any other quantum chemistry code, to obtain the energies of each electronic state needs to solve a problem of eigenvalues, this problem is written as

$$(\widetilde{H} - \epsilon\widetilde{I})\widetilde{S} = 0.$$

The elements of matrix $\widetilde{H}$ are the integrals of the independent particle Hamiltonian $H_{i,j} = \int d\vec{r}\phi_i(\vec{r})^*\hat{h}\phi_j(\vec{r})$ and the elements of $\widetilde{S}$ are the values of the orbital overlap $S_{i,j} = \int d\vec{r}\phi_i(\vec{r})^*\phi_j(\vec{r})$. The dimension of this problem is the size of the basis set used, that is proportional to the number of electrons in systems. For large calculations this is a potential computational bottle

neck to solve the whole wavefunction. The orbital functions $\phi_i(\vec{r})$ initially are built as a linear expansion of Gaussian functions, and for multiresolution representation we chose a finite basis represented by wavelets.

In summary, MADNESS requires for high performance calculations 1) a fast small matrix-matrix multiplication algorithm, 2) a reliable and fast Single Value Decomposition and 3) a parallel eigen-solver for real Hermitian matrices. MADNESS uses by default the external linear algebra subroutines included in BLAS and LAPACK. In order to improve performance of the code, we substituted BLAS/LAPACK with libraries that are more efficient for the matrix dimensions used in MADNESS, which are generally smaller than those for which BLAS/LAPACK are optimized. In the follow sections we will discuss the results when MADNESS uses linear algebra libraries written in C++.

3 Transition from BG/P to BG/Q

The Blue Gene /Q (BG/Q)architecture is a totally new technology for scientific applications and its closest technology reference available is the previous generation Blue Gene /P (BG/P). When we compare the performance between the two generations BG/Q and BG/P in most of cases we experience a speedup of 3x-4x. In the Figure 1 is plotted the comparison between BG/Q and BG/P with same number of nodes and calculation a cluster of 5 water molecules. In a glance, for small calculations BG/Q is 4 times faster than a BG/P with few nodes.

For large molecular systems, where the number of operations are more intensive, BG/Q has a much better performance than BG/P; and this performance grows directly proportional with the size of the problem, see Figure 2.

4 Lineal Algebra Libraries

4.1 Eigen3

The singular value decomposition for small square matrices are one of the most intensive serial operations in MADNESS. We added the option to substitute LAPACK by the templated C++ library Eigen3 [12]. This library

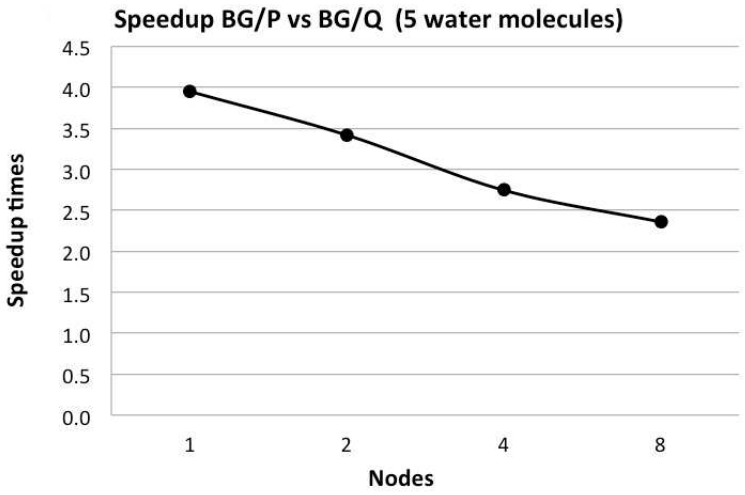

Figure 1: Comparison BG/P vs BG/Q with same number of nodes and same size of molecular system.

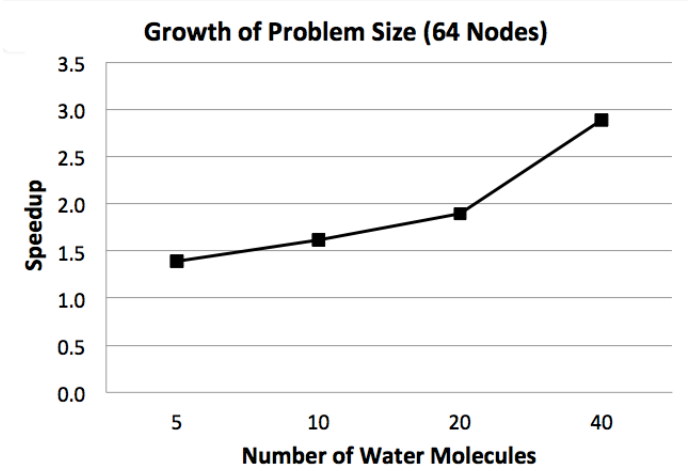

Figure 2: Performance comparison between BG/P and BG/Q when we growth the size of the molecular system computing with 64 nodes.

Program 4.1 Example of single value decomposition application using Eigen3.

```
typedef Matrix<double,Dynamic,Dynamic> MatrixX;
MatrixX U, V, M;
Matrix<double, Dynamic, 1> Sigms;
// init M

// get U and V from M=U.Sigms.V using JacobiSVD decomposition
JacobiSVD <MatrixX> svd(M, ComputeThinU | ComputeThinV);

U = svd.matrixV();
V = svd.matrixU();
Sigms = svd.singularValues();
```

substitutes most of the operations found in LAPACK, providing several algorithms to obtain the eigenvectors and eigenvalues of matrices. Each algorithm could be as fast as the precision required and the size of the matrices involved. Usually more precision means slower calculations. Eigen3 implements a special class to manipulate matrices and some operations for small matrices are hard coded. Initially, the decomposition in the LRS require small matrices operations Eigen was consider an excellent candidate to replace LAPACK in MADNESS.

An example of a small piece of code using Eigen3 to call a SVD calculation is shown in the Program 4.1. Notice that Eigen3 has its own class to represent matrices, `Matrix`. The initialization of matrices from Eigen3 with the matrices in MADNESS is made using pointers. MANDESS and Eigen3 matrix classes have in common objects to insert the directly the elements of the matrices.

In the Figure 3 we show the timings in the decomposition of the real matrix M as $M = U\Sigma V^T$ (where U and V^T are the right and left unitary matrices respectively and Σ is a vector of eigenvalues) with a size (20,20) and smaller using the libraries LAPACK and Eigen3. In this Figure we might notice that Eigen3 is very competitive when computes with small matrices; with sizes less than (16) is faster than LAPACK. Because the SVD procedure is called millions of times, even small increases in performance have a large impact on the overall runtime of the code.

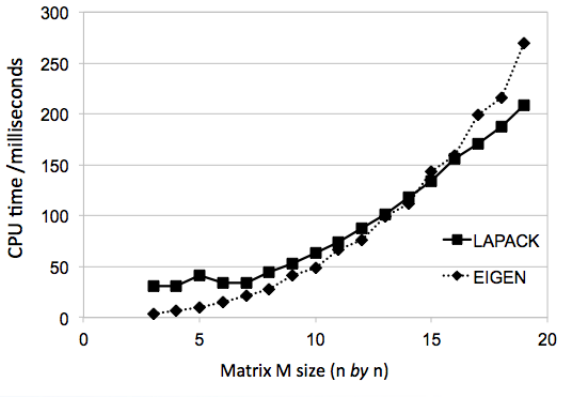

Figure 3: CPU time solving the SVD using Eigen3 and LAPACK in square matrices dimension 20 and smaller.

In spite of the positive results in the performance of Eigen3 for small matrices, its competitiveness is lost with larger matrices, as is shown in the Figure 4 where for medium size and large matrices LAPACK is approximately twice the speed of Eigen3.

4.2 Elemental

In MADNESS in order to obtain orbital energies from the Kohn-Sham or Hartree-Fock methods is necessary to solve a matrix problem of eigenvalues. The dimension of the matrix in our case is the number of occupied orbitals. We interfaced MADNESS matrices to Elemental to facilitate the operations of large matrices and vectors. This is particularly relevant to diagonalize the Hamiltonian matrix and also useful to project the initial Gaussian basis set functions into the polynomial basis set via LRS. For small molecular systems with hundreds of electrons the matrix operations made with LAPACK had negligible times, and single-node libraries where sufficient. Nevertheless, when MADNESS calculates large molecules, with thousands of orbitals, the use of a parallel eigen solver is mandatory, thus, we chose Elemental [13] because it is a modern, object-oriented C++ library that fits with the rest of MADNESS' design and because its performance has been shown to be excellent on Blue Gene systems, i.e. it is faster than ScaLAPACK (this result is not unique to Blue Gene, however).

Elemental maps the MPI processes used on 2D grid and distributes the

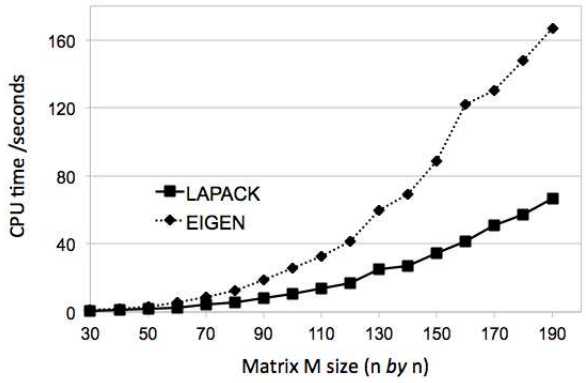

Figure 4: CPU time solving the SVD using Eigen3 and LAPACK in square matrices with dimension between 30 to 200.

matrices data in blocks along that grid. The mathematical operations of the matrices are performed in groups of elements in the grid, the time saved doing this operations are reflected in operation of reduction and collection. Elemental is based in the previous designs found in FLAME and PLAPACK, projects held in the University of Texas, Austin. The snippet in the Program 4.2 exemplify how we call the eigen solver in Elemental using C++. The value `blocksize` should be chosen taking in account the size of the matrix to manipulate and the number of processors. For large matrices in BG/Q (size >3600) we found convenient to set `blocksize=128`.

The Figure 5 shows the performance of Elemental obtaining the eigenvectors and eigenvalues of real matrices with dimension of 3200 to 9000. The speedup values in this plot refer to the timings obtained with 8 nodes of BG/Q. For very large problems, when one passes form 8 to 16 processors the speedup is bigger than the expected, 1.2x . A careful analysis of the plot proves that Elemental is hard to saturate even with 128 processors. The difference between LAPACK and Elemental is extremely large in a multiprocessor scheme, since the former is serial. serial code and the later a parallel one. To exemplify the difference between the too libraries to solve the eigen problem of a matrix with a size 3600 to LAPACK takes 753 seconds, while Elemenetal takes 68.2 seconds when using 16 nodes.

Program 4.2 Example of a solution of an eigenvalues problem kind AxBx using Elemental.

```
const int blocksize = 128; //set block size for the data distribution.
SetBlocksize(blocksize);
Grid GG(MPI_COMM_WORLD); //set grid processors within the MPI rank

// 'int n' is the size of the matrix
DistMatrix<T> B( n, n, GG ), A( n, n, GG );
DistMatrix<double> X( n, n, GG ); //eigenvectors
DistMatrix<double,VR,STAR> w( n, n, GG); //eigenvalues

// init matrices A and B

HermitianGenDefiniteEigType eigType = AXBX; //problem to solve Ax=wBx
UpperOrLower uplo = CharToUpperOrLower('U');
HermitianGenDefiniteEig( eigType, uplo, A, B, w, X ); //get w and X
```

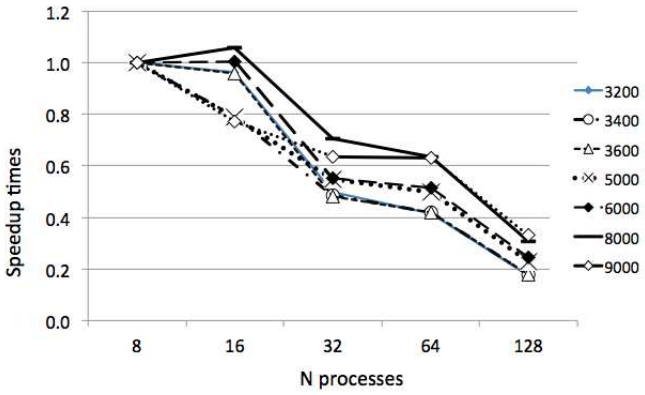

Figure 5: Parallel speedup of Elemental computing the eigenvalues and eigenvectors of matrices with different sizes in BG/Q.

5 Conclusions

MADNESS as a mathematical framework for scientific computation on petascale plataforms was optimized to speedup calculations replacing the most intense subroutines from standard linear algebra packages BLAS/LAPACK to Eigen3 and Elemental, which are more efficient and capable to exploit the particular characteristics of the BG/Q architecture. Eigen3 in BG/Q has better performance than LAPACK for small matrices, up to a size of 16. Nevertheless Eigen3 has a bad performance with bigger matrices. Interfacing of Elemental to MADNESS provides the capability to operate faster with large and distributed matrices, in particular MADNESS was benefited of the parallel eigensolver implemented in Elemental. In the near future we plan to use more features of Elemental to improve MADNESS.

References

[1] W. Koch and M. C. Holthausen, *A chemist's guide to density functional theory*, vol. 2. Wiley-Vch Weinheim, 2001.

[2] Robert J. Harrison, et al., "Multiresolution ADaptive NumErical Scientific Simulation (MADNESS)," 2010. http://code.google.com/p/m-a-d-n-e-s-s/.

[3] R. J. Harrison, G. I. Fann, Z. Gan, T. Yanai, S. Sugiki, A. Beste, and G. Beylkin, "Multiresolution computational chemistry," *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 243, 2005.

[4] R. J. Harrison, G. I. Fann, T. Yanai, and G. Beylkin, "Multiresolution quantum chemistry in multiwavelet bases," in *Computational Science—ICCS 2003*, pp. 103–110, Springer, 2003.

[5] R. J. Harrison, G. I. Fann, T. Yanai, Z. Gan, and G. Beylkin, "Multiresolution quantum chemistry: Basic theory and initial applications," *The Journal of Chemical Physics*, vol. 121, no. 23, pp. 11587–11598, 2004.

[6] T. Yanai, G. I. Fann, Z. Gan, R. J. Harrison, and G. Beylkin, "Multiresolution quantum chemistry in multiwavelet bases: Hartree–Fock exchange," *The Journal of Chemical Physics*, vol. 121, no. 14, pp. 6680–6688, 2004.

[7] F. A. Bischoff and E. F. Valeev, "Low-order tensor approximations for electronic wave functions: Hartree–fock method with guaranteed precision," *The Journal of Chemical Physics*, vol. 134, p. 104104, 2011.

[8] F. A. Bischoff, R. J. Harrison, and E. F. Valeev, "Computing many-body wave functions with guaranteed precision: The first-order møller-plesset wave function for the ground state of helium atom," *The Journal of Chemical Physics*, vol. 137, p. 104103, 2012.

[9] G. I. Fann, R. J. Harrison, G. Beylkin, J. Jia, R. Hartman-Baker, W. A. Shelton, and S. Sugiki, "MADNESS applied to density functional theory in chemistry and nuclear physics," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012018, 2007.

[10] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi, "Adaptive solution of partial differential equations in multiwavelet bases," *Journal of Computational Physics*, vol. 182, no. 1, pp. 149 – 190, 2002.

[11] M. Kalos, "Monte Carlo calculations of the ground state of three-and four-body nuclei," *Physical Review*, vol. 128, no. 4, p. 1791, 1962.

[12] "Eigen3." http://eigen.tuxfamily.org.

[13] "Elemental." https://code.google.com/p/elemental/.

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

www.anl.gov