Parallelization and optimization of Hartree-Fock method in GAMESS-US quantum chemistry code

Vladimir Mironov
Lomonosov Moscow State University
Introduction

What is quantum chemistry (QC)?

• Subject of study: chemical substances which are treated as sets of electrons and nuclei
• Method of study: approximate solution of Schrodinger equation
• Results: chemical and physical properties

Why QC?

• A very high accuracy is required to get reasonable values for the most of chemical and physical properties of molecular systems.
• Allows to study chemical reactions
Introduction

Example of the study using QC: Ras-GAP catalysis of GTP hydrolysis
GAMESS(US) quantum chemistry package

• The General Atomic and Molecular Electronic Structure System.
• One of most widely recognized packages (more than 10 000 references in literature)
• Also one of the oldest: developed since early 1980-ies
  • Mostly written in Fortran 77, now switches to Fortran 90 (only for the new modules)
  • More than 1 000 000 lines of code
• Many QC methods implemented
• Open source license
• Gaussian basis sets – not a plane wave based code!
Challenges to Intel Xeon Phi

• Many parts of GAMESS code was written decades ago
  • Optimized for old architectures
  • Global data is used throughout all the code
  • GOTOs, EQUIVALENCE statements etc.

• Only MPI parallelization

• High memory demands of QC algorithms

• To-do list:
  • Code cleanup, use Fortran 90/95
  • OpenMP parallelization and vectorization
Goals of the project

- Make GAMESS working on Xeon Phi in native regime
- Develop an OpenMP version of Hartree-Fock energy routines
- Develop an OpenMP version of Hartree-Fock gradient routine
- Switch further to density functional theory (DFT)
- Tune fragmented molecular orbital (FMO) approach for work on Xeon Phi
Hartree-Fock method

• One of the first *ab initio* methods

• Basis for many other QC methods

• Theoretical computational complexity is $O(N^4)$: it requires calculation $N^4$ of two-electron integrals (also known as electron repulsion integrals, ERIs)

• ERI screening techniques decrease complexity down to $\sim O(N^{2.5})-O(N^3)$
Direct Hartree-Fock algorithm in GAMESS

- Initial guess of density and orbital energies ($\mathbf{D}$: $\epsilon_i$), overlap ($\mathbf{S}$), one-electron integrals ($h_{ij} = \langle \chi_i | \mathbf{h} | \chi_j \rangle$)

- Two-electron integrals calculation and Fock matrix construction

- Fock matrix diagonalization, update $\mathbf{D}$, $\epsilon_i$

- Is energy/density converged?
  - no
  - yes

- Properties calculation, output the results, etc.

>90% of total compute time

```
DO ish=1,NSH
  DO jsh=1,ish
    DO ksh=1,jsh
      DO lsh=1,ksh
        test = max(D,ish,jsh,ksh,lsh) * max(1,ish,jsh,ksh,lsh)
        IF (test.GT.cutoff) THEN
          (ish,jsh,ksh,lsh)
          (ish,ksh,jsh,lsh)
          (ish,lsh,jsh,ksh)
          CALL DIRFCK
        ENDIF
      ENDDO
    ENDDO
  ENDDO
ENDDO
ENDDO
```

NSH – number of shells, commonly NSH ≤ 1000

(ISH,JSH|KSH,LSH) – integral calculation over shells also contains fourfold nest of small loops

Calculate ERIs over shells if unique

Integral screening

Fock matrix update
Hatree-Fock MPI/OpenMP parallelization

- Rectangular iteration space instead of triangular: OpenMP load imbalance is very low (<1%)
  good LB not only for large but also for medium workloads
- MPI parallelization over top loop
  heavy MPI communication is only once in each HF iteration
- MPI dynamic load balance scheme – not so effective as OpenMP load balance but acceptable
- Density matrix is shared, Fock matrix is stored in thread-local memory for now...
- Currently rotated axis (up to D shells) and Rys quadrature ERI algorithms are supported. GAMESS ERIC electron-repulsion integral algorithm is upcoming
- Tested for R/U/RO-HF methods and for HF exchange in hybrid R/U/RO-DFT calculations (eq. B3LYP)

```c
#define MAXD 10

!$omp parallel
  do ish=NSH, 1
  mpi dynamic load balance
!$omp do schedule(dynamic,1) collapse(2)
  do jsh=1, ish
    do ksh=1, ish
      if (ish.eq.ksh) then
        lmax = jsh
      else
        lmax = ksh
      endif
      do lsh=1, lmax
        test = max(d,ish,jsh,ksh,lsh) * max(i,ish,jsh,ksh,lsh)
        if (test.gt.cutoff) then
          calculate (ish,jsh,ksh,lsh)
          call dirfck
        endif
      enddo
    enddo
  enddo
enddo
!$omp end do nowait
!$omp end parallel
mpi fock matrix reduction
```
Strong scaling: OpenMP/MPI

- $C_{60}$ molecule
- 6-31G basis set
Strong scaling: OpenMP

- $C_{60}$ molecule
- 6-31G basis set
Integral code optimization

• Specifics:
  • ERI compute path is strongly dependent on the properties of shells in quartet (angular momentum, degree of contraction).
  • Algorithms are multistep, some steps are memory bounded, some – compute bounded.
  • Most of calculation occurs in nests of short loops (< vector length).
• Gauss-Rys ERI code has been rewritten from scratch
Challenges:

- Short loops:
  depending on basis set and shell combination trip counts of bottom loops in nests are frequently smaller than optimal vector length
  Solution: if in critical part of code: manual unroll, collapse loop nest if possible, etc.

- Indirect memory access:
  sometimes indirect memory access is unavoidable
  Solution: change data layout and/or loop structure to minimize the impact of such operations on performance, cache data. If it is impossible – use vectorized gather/scatter.

- Hardly vectorizable routines (like numerical interpolation):
  a combination of small loops, complex logic, calls to external functions
  Solution: enhance logic of the routine to avoid as many computations as possible, further investigation of algorithm modification

- Various minor issues with vectorization
  compiler’s decisions are very conservative on vectorization stage
Example: 2D to 6D integral transformation

- GAMESS original: $3 \times 13 \times 7^4$ static array (>700 Kbytes) in common block
- Array elements access (in case $N_{\text{roots}} = 5$):

```plaintext
DO 51 K = 1, MAX
   MX = NX+KLX(K)
   MY = NY+KLY(K)
   MZ = NZ+KLZ(K)
   N = N1+KLGT(K)
   GHONDO(N) = GHONDO(N) + D1*DKL(K)*
                  ( XIN(MX )*YIN(MY )*ZIN(MZ )
                  * + XIN(MX+ 2401)*YIN(MY+ 2401)*ZIN(MZ+ 2401)
                  * + XIN(MX+ 4802)*YIN(MY+ 4802)*ZIN(MZ+ 4802)
                  * + XIN(MX+ 7203)*YIN(MY+ 7203)*ZIN(MZ+ 7203)
                  * + XIN(MX+ 9604)*YIN(MY+ 9604)*ZIN(MZ+ 9604))
```

Loops over primitive Gaussian quadruplets

Indirect access

Top-level

Second level
Example: 2D to 6D integral transformation

- GAMESS original: $3 \times 13 \times 7^4$ static array (>700 Kbytes) in common block

```
Example: 2D to 6D integral transformation

- GAMESS original: $3 \times 13 \times 7^4$ static array (>700 Kbytes) in common block

```

**Diagram Description:**
- **YINTS**
  - Root1
  - Root2
  - ...

- **ZINTS**
  - Root1
  - Root2
  - ...

- **XINTS**
  - Root1
  - Root2
  - ...

- **GHONDO**
  - From 1 to 2401 doubles
  - 19 Kbytes
  - 19 Kbytes

**Notes:**
- 09.06.2016
- IXPUG EMEA 2016 Ostrava Workshop
Example: 2D to 6D integral transformation

• How to improve performance: simple way
  • Reorder array indices:
    e.q. $XYZ(i,j,k,l,\text{roots},\text{xyz}) \rightarrow XYZ(\text{roots},i,j,k,l,\text{xyz})$
  • Accumulate sum in temporary variable to eliminate vector dependence in GHONDO array

• Current implementation
  • Lowest array dimension corresponds to both roots and primitive quadruplets – increased data vector length
  • Dynamical array sizes – improved cache utilization
  • Reduced multiplication scheme of Gauss-Rys quadrature implemented: indirect access simplified and regularized, intermediate products are cached
  • Combined with contraction step
Integral code optimization

Performance results on single Xeon Phi 7120D
single HF iteration, balanced thread affinity

High ($L_{\text{max}}=4$) angular momentum benchmark
($\text{Li}_6$ cluster, cc-pVQZ basis set)

speedup 2.4x
Integral code optimization

Performance results on single Xeon Phi 7120D
single HF iteration, balanced thread affinity

Low $l_{\text{max}}=1$ angular momentum benchmark
(fullerene molecule, DZV basis set)

speedup 1.6x
Integral code optimization

Performance results on single Xeon Phi 7120D
single HF iteration, balanced thread affinity

Organic molecule benchmark
(two pyrazine molecules
at 15 Å, \( L_{\text{max}} = 2, \)
cc-pVDZ basis set)

speedup 1.9x
Summary of the improvements

• OpenMP vs original MPI implementation
  now it is possible to utilize all cores
  on Xeon Phi MPI only implementation is extremely slow

• Performance improvement:
  Xeon ~2.0x, MIC 1.5-2.5x
  OpenMP scaling on Xeon Phi is much better (1.2-2.0x improvement)

• Xeon vs MIC:
  depends on the workload,
  for organic molecules with popular basis sets KNC 7120 ≈ Ivy Bridge Xeon

• ERI code:
  More effective memory access
  Modern algorithm with reduced flops count w/o penalty to parallelization
  Vectorization – only effective for high angular momentum workloads, but it would be improved in next version of code
What’s next?

Current problems:
• Vectorization is effective only for high angular momentum: need to increase data vector length
• MPI load balance is not always effective enough

Solution to the both problems: sort shells by their type at the very beginning of the HF procedure
• Calculation of ERIs over multiple shells at once
• Better screening techniques
• Vectorized Fock matrix update
• Vectorized subroutines for Rys roots and weights calculation
• MPI load balance will automatically improve
• Only small modifications to the current ERI and HF codes are required
More information about the project

• OpenMP version of the HF of code is available at the GAMESS developer repository, branch “openmp_mod”
  https://www.msg.chem.iastate.edu/review/
  registration is required

• New ERI code would be available after shell sorting update

• Paper on OpenMP code for HF:
Thank you for your attention!
ERI calculation algorithm (Gauss-Rys quadrature)

1. Get shell indices, load shell data
2. Compute 2-index coefficients
3. Compute 4-index coefficients
4. First recursion step ("VRR"): 4-i coefficients $\rightarrow [e,0|f,0]_{2D}$ integrals
5. 2d to 6d integral transformation: $[e,0|f,0]_{2D} \rightarrow [e,0|f,0]$
6. Contraction: $[e,0|f,0] \rightarrow (e,0|f,0)$
7. Second recursion step ("HRR"): $(e,0|f,0) \rightarrow (a,b|c,d)$
Integral code optimization

Thread affinity impact

Organic molecule benchmark (two pyrazine molecules at 15 Å, $L_{\text{max}}$=2, cc-pVDZ basis set)
Quantum chemistry specifics:

- Computational complexity of *ab initio* methods is $O(N^4)$ and above
  - High CPU and memory demands
- Large variety of methods
  - Very different data structures
  - Different computational bottlenecks
  - Not all of methods could be easily/efficiently parallelized on current hardware
  - Chains of methods
  - Almost no optimized QC libraries
  - Most popular approach – density functional theory
- Double (or even higher) precision is required for chemical accuracy