Accelerating Quantum Chemistry Research using GPUs – Two Electron Integrals in GAMESS

Andrey Asadchev
Jacob Felder
Veerendra Allada
Dr. Mark S Gordon
Dr. Theresa Windus
Dr. Brett Bode

GPU Technology Conference , NVIDIA , San Jose, 2009

Outline

- Computational Quantum Chemistry
- General Atomic Molecular Electronic Structure Systems -GAMESS
- Electron Repulsion Integral (ERI) Problem
- Our Approach
  - CUDA Implementation
  - Optimizations
  - Automatically generated code
- Performance Results
- Future Goals
- Questions & Discussion

Computational Quantum Chemistry

- Use computational methods to solve the electronic structure and properties of molecules.
- Finds utility in the design of new drugs and materials
- Underlying theory is based on Quantum Mechanics –Schrodinger wave equation
- Properties calculated
  - Energies
  - Electronic charge distribution
  - Dipole moments, vibrational frequencies.
- Methods employed
  - Ab initio Methods (Solve from first principles)
  - Density Functional Theory (DFT)
  - Semi-empirical methods
  - Molecular Mechanics (MM)

General Atomic Molecular Electronic Structure System (GAMESS)

- Ab initio molecular quantum chemistry software
- USDOE "SciDAC Basic Energy Sciences" (BES) application
- Serial and parallel versions for several methods
- In brief, GAMESS can compute
  - Self Consistent Field (SCF) wave functions - RHF, ROHF, UHF, GVB, and MCSCF using the Hartree-Fock method
  - Correlation corrections to SCF using configuration interaction (CI), second order perturbation theory, and coupled cluster theories (CC)
  - Density Functional Theory approximations

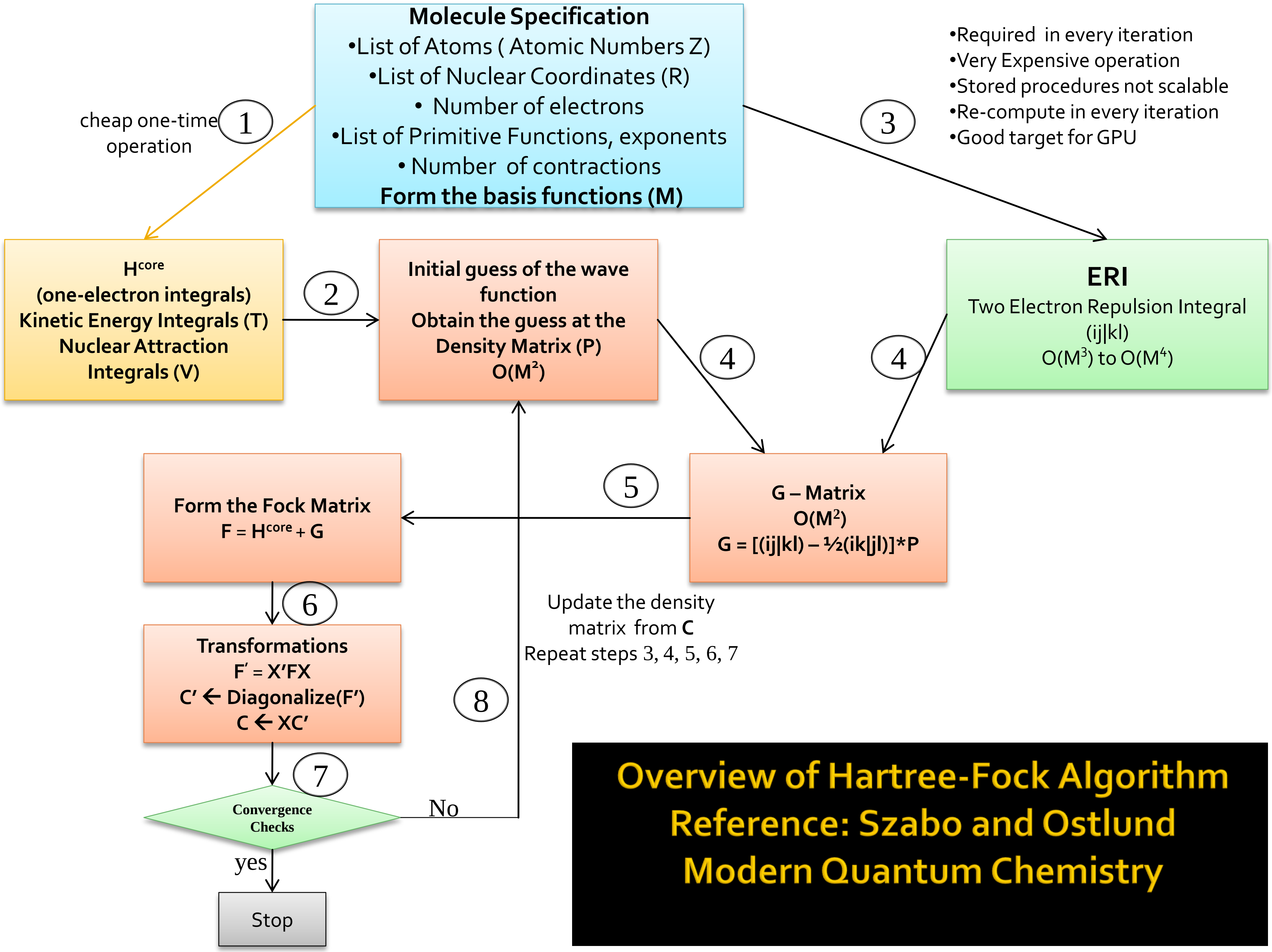
Reference:"Advances in electronic structure theory: GAMESS a decade later" M.S.Gordon, M.W.Schmidt pp. 1167-1189, in "Theory and Applications of Computational Chemistry: the first forty years" C.E.Dykstra, G.Frenking, K.S.Kim, G.E.Scuseria (editors), Elsevier, Amsterdam, 2005.

Background

- Molecules are made of atoms and atoms have electrons
- Electrons live in shells – s, p, d, f, g, h
- Shells are made of sub-shells – all have the same angular momentum (L)
- Shells are represented using the mathematical functions
  - Gaussian functions are taken as standard primitive functions (S.F. Boys)
  - $\varphi(r) = x^{a_x} y^{a_y} z^{a_z} exp(-\alpha r^2)$
  - x, y, z – Cartesian center
  - a_x, a_y, a_z – Angular momenta components; $L = a_x + a_y + a_z$
  - α is the exponent
  - Shells with low angular momentum are typically contracted
    - $\phi_a(r) = \sum_k^K D_{ka} \varphi_k(r)$
    - K is the contraction coefficient. D_k's are the contraction coefficients

Overview of Hartree-Fock Algorithm

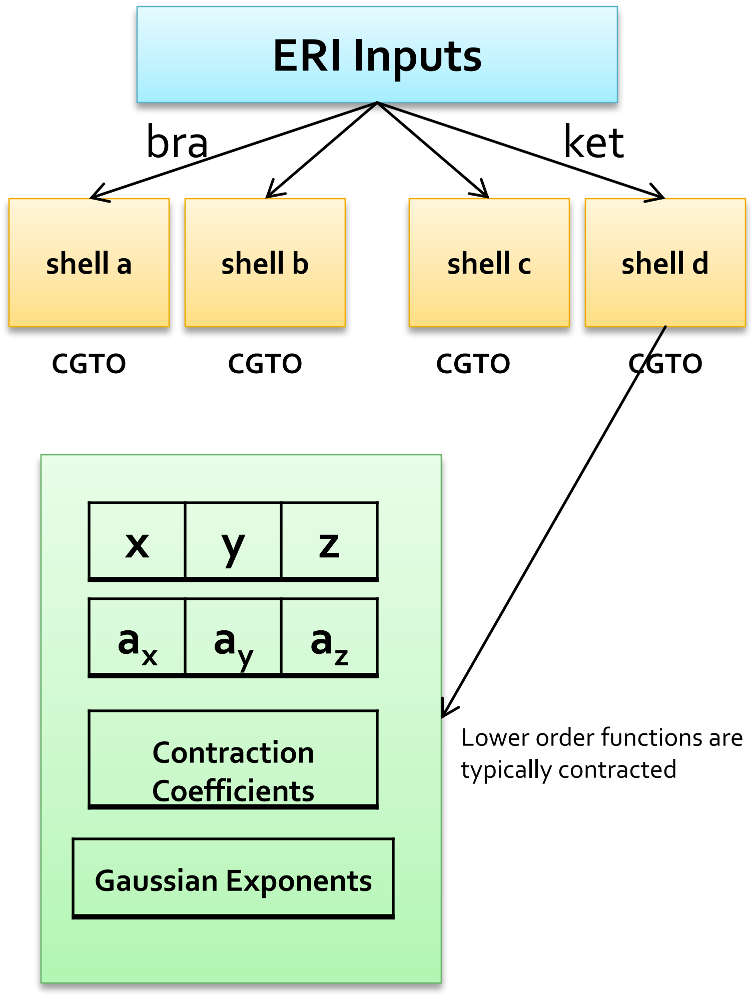
Reference: Szabo and Ostlund Modern Quantum Chemistry



Molecule Specification
•List of Atoms (Atomic Numbers Z)
•List of Nuclear Coordinates (R)
• Number of electrons
•List of Primitive Functions, exponents
• Number of contractions
Form the basis functions (M)
cheap one-time operation
1
•Required in every iteration
•Very Expensive operation
•Stored procedures not scalable
•Re-compute in every iteration
•Good target for GPU
3
H^core
(one-electron integrals)
Kinetic Energy Integrals (T)
Nuclear Attraction Integrals (V)
2
Initial guess of the wave function
Obtain the guess at the Density Matrix (P)
O(M^2)
4
4
ERI
Two Electron Repulsion Integral
(ij|kl)
O(M^3) to O(M^4)
Form the Fock Matrix
F = H^core + G
5
G – Matrix
O(M^2)
G = [(ij|kl) – ½(ik|jl)]*P
6
Transformations
F′ = X′FX
C′ ← Diagonalize(F′)
C ← XC′
Update the density matrix from C
Repeat steps 3, 4, 5, 6, 7
8
7
Convergence Checks
No
yes
Stop

Focus
Electronic Repulsion Integral (ERI) Problem

- Four-center two-electron repulsion integral
  - $(ab|cd) = \iint \varphi_a(1)\varphi_b(1)\frac{1}{r_{12}}\varphi_c(2)\varphi_d(2)$
- Major computational step in both *Ab initio* and DFT methods
- Complexity is O(M3)-O(M4), M is the number of basis functions (Gaussian functions are standard)
- Rys Quadrature – proposed by Dupius, Rys, King (DRK)
  - Numerical Gaussian quadrature based on a set of orthogonal Rys polynomials
  - Numerically stable, low memory foot print
  - Amenable for GPUs and architectures with smaller caches



Relevant to computations

Computation of Electron Repulsion Integrals Using the Rys Quadrature Method - Dupuis, Rys, King

- Two electron integral is expressed as $(ij|kl) = \sum_{m=0}^{L} C_m F_m(X)$ where , $F_m(X) = \int_0^1 t^{2m} \exp(-Xt^2)dt$ and $L = L_a + L_b + L_c + L_d$
- X depends on exponents, centers and is independent of angular momenta

$$X = \rho(r_A - r_B)^2 \qquad \rho = AB/(A+B)$$
$$r_A = (\alpha_i r_i + \alpha_j r_j)/A \qquad A = \alpha_i + \alpha_j$$
$$r_B = (\alpha_k r_k + \alpha_l r_l)/B \qquad B = \alpha_k + \alpha_l$$

- $(ij|kl) = \int_0^1 \exp(-Xt^2) P_L(t)dt$, where P$_L$(t) is polynomial of degree L in t2. Evaluated using N-point quadrature and hence $(ij|kl) = \sum_{\omega=1}^{N} W_\omega P_L(t_\omega)$ where $N = L/2+1$
- Using separation of variables, P$_L$(t) which is integral over $dr_1 dr_2$, can be written as a product of three (2-D) integrals over $dx_1 dx_2$, $dr_1 dr_2$, $dz_1 dz_2$
- $(ij|kl) = 2(\rho/\pi)^{1/2} \sum_\omega I_x(t_\omega) I_y(t_\omega) I_z(t_\omega) W_\omega$ and $I_{q(=x,y,z)}(N, 0{:}L_a, 0{:}L_b, 0{:}L_c, 0{:}L_d)$
- Ix, Iy, Iz are computed using recurrence and transfer relations

Rys Quadrature Algorithm

```
Rys Quadrature Algorithm
for all $l$ do
    for all $k$ do
        for all $j$ do
            for all $i$ do
                $I(i,j,k,l) = \sum_{\omega} I_x(\omega, i_x, j_x, k_x, l_x) I_y(\omega, i_y, j_y, k_y, l_y) I_z(\omega, i_z, j_z, k_z, l_z)$
            end for
        end for
    end for
end for
```

- Summation over the roots over all the intermediate 2-D integrals
- floating point operations = $3 * N * \binom{L_a + 1}{2} \binom{L_b + 1}{2} \binom{L_c + 1}{2} \binom{L_d + 1}{2}$
- Recurrence, transfer and roots have predictable memory access patterns, fewer flops. Quadrature step is the main focus here.

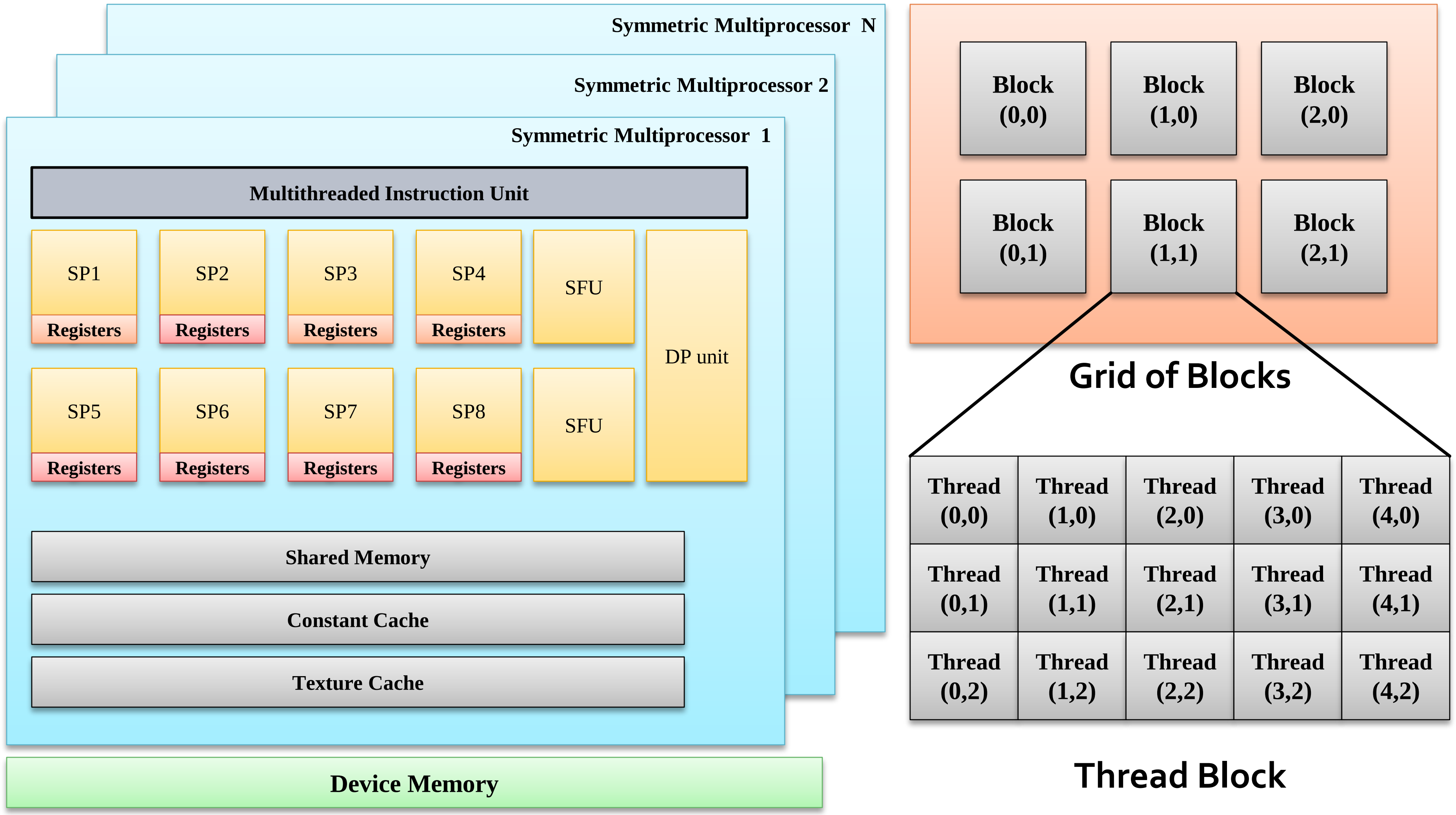
Rys Quadrature Algorithm

- Example: (dd|dd) ERI block
  - $L_a = L_b = L_c = L_d = 2$
  - Number of roots, N = 5
  - ERI size = 6^4 = 1296 elements
  - Intermediate 2-D integrals Ix, Iy, Iz size:$3^4*5 = 245$

Possible Optimizations

- ERI computations are memory bound, hence optimize memory accesses
- Intermediate 2-D integrals are reused multiple times to construct different ERI elements.
- Generate the different combinations automatically

CUDA – Compute Unified Device Architecture (Birds eye-view)



SM – Streaming Multiprocessor
SP – Scalar Processor Core
SFU – Special Functional Unit
DP – Double Precision Unit

CUDA Rys Quadrature Implementation considerations

- Since 2-D integrals are reused multiple times, load them into shared memory
  - However, shared memory access , synchronization limited to thread block boundaries
  - ERI block should be mapped onto a single thread block
  - Is it possible to map all the ERI elements to individual threads in a block ?
  - The answer depends on the ERI block under consideration
- For a (dd|dd) ERI block, ERI size = 6^4 = 1296 elements
  - Maximum of 512 or 768 threads per block
  - Map *i*, *j*, *k* indices corresponding to the three shells of the block to unique threads and iterate over the *l* index
  - Thread blocks are three dimensional, the mapping of *i*, *j*, *k* is natural
- For (ff|ff) ERI block, ERI size = 10^4 = 1000 elements
  - Map *i*, *j* indices corresponding to the first two shells of the block to unique threads and iterate over the *l* index

i,j,k mapping – algorithm, optimizations

CUDA Rys quadrature: i, j, k mapping

```
# map threads to ERI elements
I = threadIdx.x, j = threadIdx.y, k = threadIdx.z

# arrays LX, LY, LZ map functions  to exponents
(ix, iy, iz) ← (LX[i], LY[i], LZ[i])
(jx, jy, jz) ← (LX[j], LY[j], LZ[j])
(kx, ky, kz)← (LX[k], LY[k], LZ[k])

for all l do
    syncthreads
    ## load the 2-D  integrals to shmem
    $I_{x, shmem}$ ← Ix(:,:,:,LX[l])
    $I_{y, shmem}$ ←Iy(:,:,:,LX[l])
    $I_{z, shmem}$ ← Iz(:,:,:,LX[l])
    syncthreads

    I(i, j, k, l) ← $\sum_N I_{x,shmem} I_{y,shmem} I_{z,shmem}$
end for
```

Further optimizations

- (dd|dd) case
- $I_{\{x,y,z\},shmem} = 5(3^3) = 135$ elements per 2-D block
- Across iterations, some of the elements in shared memory can be reused

d-shell

$d_x^2, d_y^2, d_z^2, d_{xy}, d_{xz}, d_{yz}$ → 18 loads

$d_y^2, d_z^2, d_{yz}, d_{xy}, d_{xz}, d_x^2$ → 13 loads

I_x	0*	0	0	1*	1	2*
I_y	2*	0*	1*	1	0*	0
I_z	0*	2*	1*	0*	1*	0*

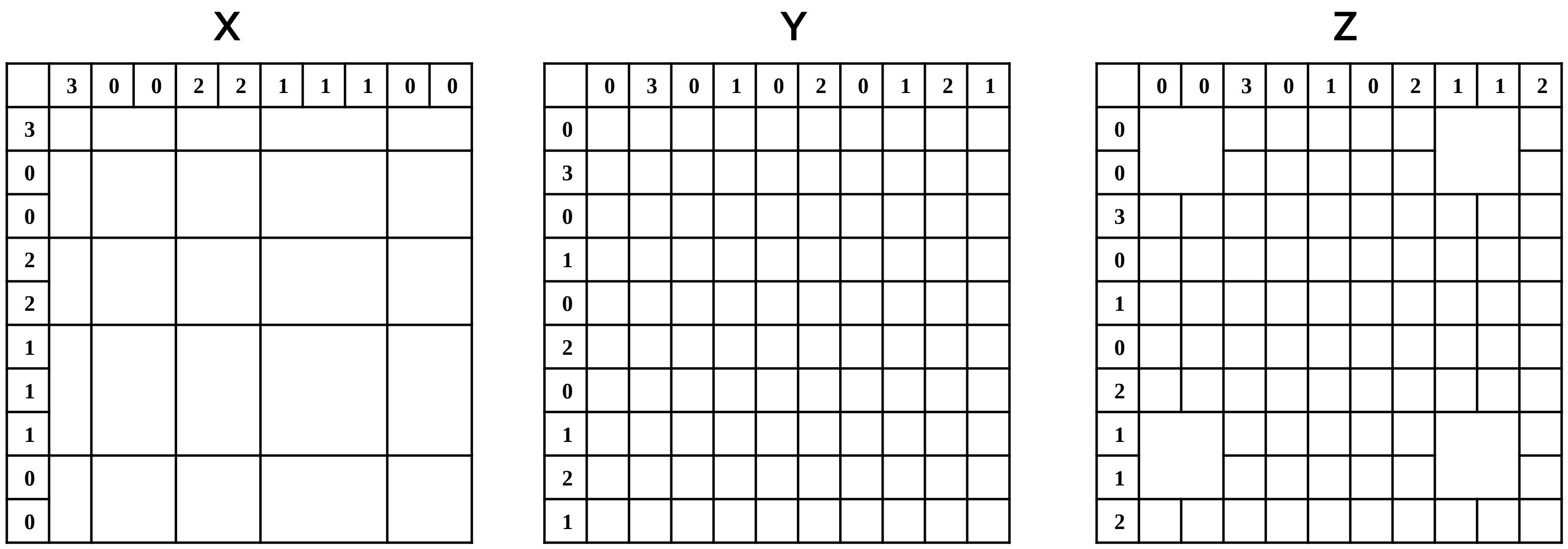
i,j mapping – algorithm, optimizations

CUDA Rys quadrature: i, j mapping

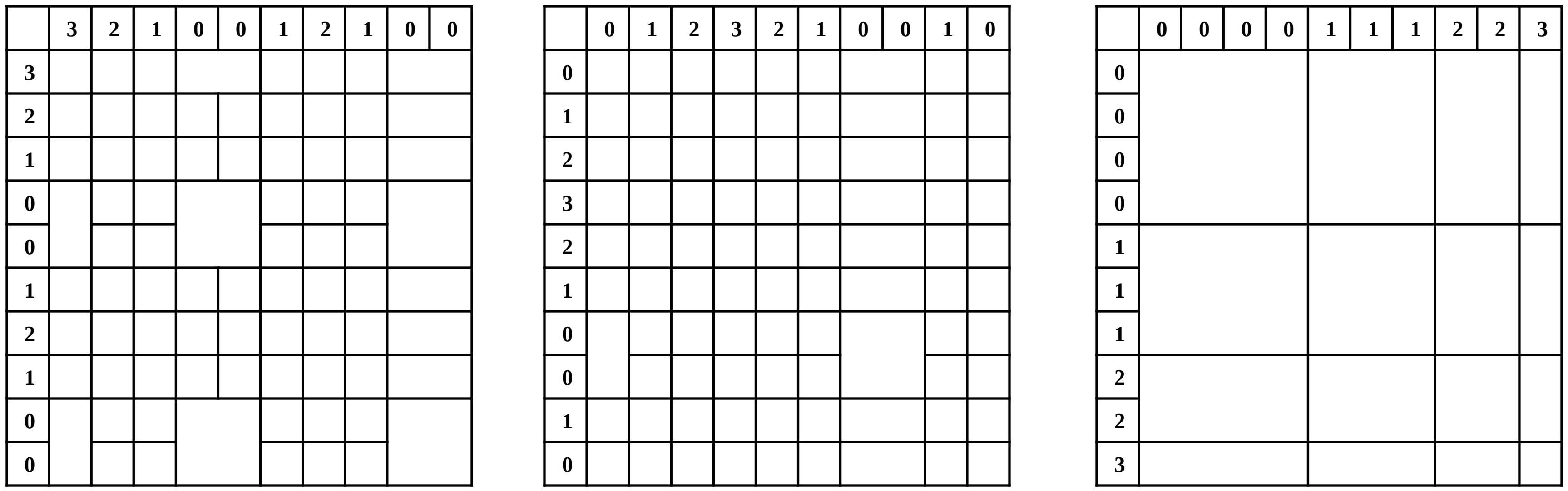
```
# map threads to ERI elements
I = threadIdx.x, j = threadIdx.y

# arrays LX, LY, LZ map functions  to exponents
(ix, iy, iz) ← (LX[i], LY[i], LZ[i])
(jx, jy, jz) ← (LX[j], LY[j], LZ[j])
for all $kl_{z\text{-}block}$ do
    syncthreads
    $I_{z,\ shmem}$ ← Iz(:,:,LZ[k],LZ[l])
    ## load 2-D  integrals to shmem
    for all $kl_{xy}$   $kl_{z\text{-}block}$ do
    syncthreads
    $I_{x,\ shmem}$ ← Ix(:, :, LX[k], LX[l])
    $I_{y,\ shmem}$ ← Iy(:, :, LY[k], LX[l])
    syncthreads
    I(i, j, k, l) ← $\sum_N I_{x,shmem} I_{y,shmem} I_{z,shmem}$
    end for
end for
```

Further optimizations

- (ff|ff) case
- $I_{\{x,y,z\},shmem} = 7(4^2) = 112$ elements per 2-D block
- 10 functions in the f-shell
- Reorder them (next slide)



$f_{x^3}, f_{y^3}, f_{z^3}, f_{x^2y}, f_{x^2z}, f_{xy^2}, f_{xz^2}, f_{xyz}, f_{y^2z}, f_{yz^2}$



$f_{x^3}, f_{x^2y}, f_{xy^2} f_{y^3}, f_{y^2z}, f_{xyz}, f_{x^2z}, f_{xz^2}, f_{yz^2}, f_{z^3}$

Automatic Code Generation based on Templates Python – Cheetah

Warning: Template engine but not C++ templates

- Number of registers per thread, shared memory per thread block limits the thread blocks that can be assigned per SM
- Loops implemented directly result in high register usage
- Explicitly unroll the loops. How ? Manually it’s tedious and error-prone
- Use a common template and generate all the cases
- Python based Cheetah template engine is used- reuse existing Python utilities and program support modules easily.

Performance Results – Evaluated using the GeForce GTX 275

ERI	blocks	flop count	GFLOPS$_{SP}$[3]		GFLOPS$_{DP}$[4]	
			map[5]i jk	map[5]i j	map[5]i jk	map[5]i j
(gg|gg)	2000	2733750000	n/a	45.23	n/a	22.55
(gg|f f)	4000	2160000000	n/a	34.42	n/a	15.32
(f f|gg)	4000	2160000000	n/a	30.91	n/a	14.11
(gg|dd)	10000	1701000000	n/a	43.08	n/a	21.05
(gg|pp)	40000	1458000000	n/a	**36.53**	n/a	**17.08**
(pp|gg)	40000	1458000000	**34.23**	6.93	**18.20**	5.38
(f f|f f)	10000	2100000000	n/a	40.43	n/a	20.11
(f f|dd)	20000	1296000000	n/a	**37.54**	n/a	**18.29**
(dd|f f)	20000	1296000000	**37.69**	23.32	**16.53**	15.04
(f f|pp)	80000	1080000000	27.43	**31.46**	15.23	**17.05**
(pp|f f)	80000	1080000000	**32.23**	6.21	**17.45**	4.84
(dd|dd)	60000	1166400000	**31.10**	20.17	**16.38**	13.67

- ERIs with odd number of roots have maximum performance over the even roots
  - Odd roots - (gg|gg), (gg|dd), and (ff|ff) cases
  - Even roots – (ff|gg), (gg|ff), and (dd|gg)
- The difference is as high as 25%
- Difference in the single and double precision is roughly a factor of two
- Larger *ijk* mapping perform better than the *ij* mappings

GTX 275 and Tesla Performance Comparison

ERI	blocks	flop count	GFLOPS$_{SP}$[3]		GFLOPS$_{DP}$[4]	
			GTX 275	Tesla	GTX 275	Tesla
$(gg\|gg)$	2000	2733750000	45.23	55.97	22.55	27.34
$(gg\|ff)$	4000	2160000000	34.42	42.07	15.32	18.67
$(ff\|gg)$	4000	2160000000	30.91	37.70	14.11	17.19
$(gg\|dd)$	10000	1701000000	43.08	53.39	21.05	25.34
$(dd\|gg)$	10000	1701000000	23.63	24.03	16.35	29.88
$(gg\|pp)$	40000	1458000000	36.53	45.15	17.08	20.65
$(pp\|gg)$	40000	1458000000	34.23	42.42	18.20	22.09
$(ff\|ff)$	10000	2100000000	40.43	50.19	20.11	24.46
$(ff\|dd)$	20000	1296000000	37.54	46.15	18.29	22.44
$(dd\|ff)$	20000	1296000000	37.69	45.71	16.53	19.71
$(ff\|pp)$	80000	1080000000	31.46	39.38	17.05	20.10
$(pp\|ff)$	80000	1080000000	32.23	40.33	17.45	21.46
$(dd\|dd)$	60000	1166400000	31.10	38.74	16.38	19.78

Inferences

- Performance depends on the ERI class under evaluation and hence also on the mapping (***i,j,k*** vs. ***i,j***)
- Difference between single and double precision performance is roughly a factor of two
- Difference between the GTX and Tesla T is roughly 30% (consistent with the clock speeds)
- In terms of register and shared memory usage both are identical

Conclusions

- Rysq quadrature implementation performance results are comparable or better than DGEMV BLAS routines.
- Some more improvements are possible by caching (texture, constant) and also by more aggressive memory reuse possibly at the expense of re-computation
- Very easy to generate the possible ERI shell combinations using a single template
- Explicit unrolling can be controlled at different levels such as shells, roots to test for performance improvements
- Being developed as a standalone library and application agnostic

On going work

- ERIs are 4-dimensional, hence it is very expensive to transfer them to the host memory after computation.
- Fock matrix is 2-dimensional. So, consume the ERI’s as they are formed to build the Fock matrix
- Handle the contracted ERI’s
- Mixed precision support
- A complete working SCF algorithm

References

1) Rys, J.; Dupuis, M.; King, H. *J. Comput. Phys.* **1976**, *21*, 144.
2) Boys, S.F. *Proc. R. Soc* **1950**, *200*, 542.
3) Rys, J.; Dupuis, M.; King, H. *J. Comput. Chem.* **1983**, *4*, 154–157.
4) Gordon, M. S.; Schmidt, M. W. Advances in electronic structure theory: GAMESS a decade later. In *Theory and Applications of Computational Chemistry: the first forty years*; Dykstra, C. E.; Frenking, G.; Kim, K. S.; Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005.
5) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222–231.
6) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 1004–1015.
7) Yasuda, K. *Journal of Computational Chemistry* **2008,** *29*, 334-342.

Acknowledgements & Contacts

US Department of Energy

Department of Defense - DURIP Grant

Ames Laboratory, Iowa State University

Air Force Office of Scientific Research

National Science Foundation - Petascale Applications grant

NVIDIA Corporation

Professor Todd Martinez and his group

asadchev@gmail.com
jfelder@iastate.edu
allada.v@gmail.com
mark@ si.msg.chem.iastate.edu
theresa@fi.ameslab.gov
brett@si.msg.chem.iastate.edu

Questions and Discussion

???