Quantum chemistry many-body methods on GPUs and multicore CPUs

Jeff Hammond[1] and Eugene DePrince[2]

Argonne National Laboratory
[1] Leadership Computing Facility (jhammond@anl.gov)
[2] Center for Nanoscale Materials (adeprince@anl.gov)

26 January 2011

Abstract

Quantum chemistry many-body methods (QMBM) solve the Schrödinger approximately, but at relatively high accuracy as compared to molecular dynamics (MD) and density-functional theory (DFT). While both MD and DFT have been implemented on GPGPUs by a number of groups, very little work has been done with QMBM on GPGPUs despite the excellent match between their regular and floating-point intensive algorithmic structure and the data-parallel capability of GPGPUs. In this talk we discuss the implementation of one variant of QMBM – the coupled-cluster method – on NVIDIA Tesla and Fermi processors. The performance is compared to an identically structured multicore CPU implementation as well as many existing software packages (NWChem, GAMESS, Molpro, PSI, Dalton and Aces II). The performance of our prototype implementation is 7 to 10 times faster than the fastest available CPU implementation (Molpro) and our mixed-precision algorithm achieves over 500 GF while converging to the same double precision accuracy achievable with a CPU code. Finally, we will compare our single-node CPU/GPU implementation to the massively-parallel implementation in NWChem, highlighting the challenges and opportunities for realizing petaflop QMBM using large GPU clusters.

Atomistic simulation in chemistry

1. classical molecular dynamics (MD) with empirical potentials
2. ab initio molecular dynamics based upon density-function theory (DFT)
3. quantum chemistry with wavefunctions e.g. perturbation theory (PT), Coupled-Cluster (CC) or Quantum Monte Carlo (QMC).

Classical molecular dynamics

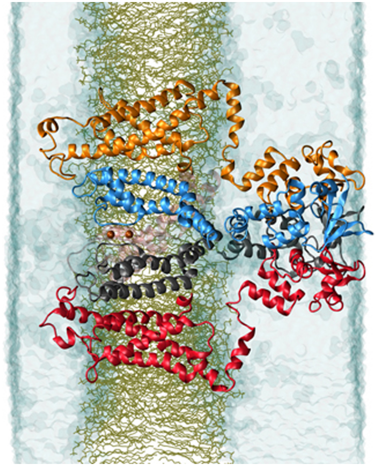


Image courtesy of Benoît Roux via ALCF.

- Solves Newton's equations of motion with empirical terms and classical electrostatics.
- Size: 100K-10M atoms
- Time: 1-10 ns/day
- Scaling: $\sim N_{atoms}$

Data from K. Schulten, et al. "Biomolecular modeling in the era of petascale computing." In D. Bader, ed., *Petascale Computing: Algorithms and Applications*.

Car-Parrinello molecular dynamics

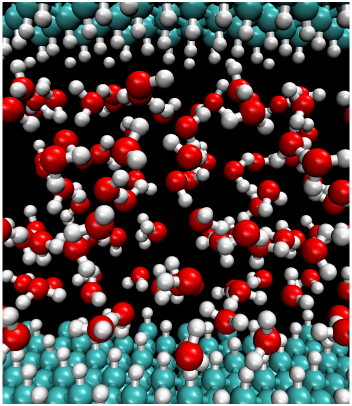
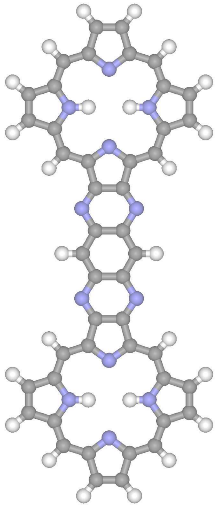


Image courtesy of Giulia Galli via ALCF.

- Forces obtained from solving an approximate single-particle Schrödinger equation; time-propagation via Lagrangian approach.
- Size: 100-1000 atoms
- Time: 0.01-1 ns/day
- Scaling: $\sim N_{el}^{x}$ (x=1-3)

F. Gygi, *IBM J. Res. Dev.* **52**, 137 (2008); E. J. Bylaska et al. *J. Phys.: Conf. Ser.* **180**, 012028 (2009).

Wavefunction theory



- MP2 is second-order PT and is accurate via magical cancellation of error.
- CC is infinite-order solution to many-body Schrödinger equation truncated via clusters.
- QMC is Monte Carlo integration applied to the Schrödinger equation.
- Size: 10-100 atoms, maybe 100-1000 atoms with MP2.
- Time: N/A
- Scaling: $\sim N_{bf}^{x}$ (x=2-7)

Image courtesy of Karol Kowalski and Niri Govind.

MD on GPUs (of many more)

All major MD packages have or will soon have a GPU implementation.

- J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten. "Accelerating molecular modeling applications with graphics processors." *J. Comp. Chem.*, 28 (16), 2618–2640, 2007.
- J. A. Anderson, C. D. Lorenz, and A. Travesset. "General purpose molecular dynamics simulations fully implemented on graphics processing units." *J. Comp. Phys.*, 227 (10), 5342–5359, 2008.
- P. Friedrichs, M. S.and Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande. "Accelerating molecular dynamic simulation on graphics processing units." *J. Comp. Chem.*, 30 (6), 864–872, 2009.
- R. Yokota, T. Hamada, J. P. Bardhan, M. G. Knepley, and L. A. Barba. "Biomolecular electrostatics simulation by an FMM-based BEM on 512 GPUs." *CoRR*, abs/1007.4591, 2010.

DFT on GPUs

- K. Yasuda, "Accelerating density functional calculations with graphics processing unit." *J. Chem. Theo. Comp.*, 4 (8), 1230–1236, 2008.
- L. Genovese, M. Ospici, T. Deutsch, J.-F. Mehaut, A. Neelov, and S. Goedecker. "Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures." *J. Chem. Phys.*, 131 (3), 034103, 2009.
- I. S. Ufimtsev and T. J. Martínez. "Quantum chemistry on graphical processing units. 2. Direct self-consistent-field (SCF) implementation." *J. Chem. Theo. Comp.*, 5 (4), 1004–1015, 2009; "Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics." *J. Chem. Theo. Comp.*, 5 (10), 2619–2628, 2009.
- C. J. Woods, P. Brown, and F. R. Manby. "Multicore parallelization of Kohn-Sham theory." *J. Chem. Theo. Comp.*, 5 (7), 1776–1784, 2009.
- P. Brown, C. J. Woods, S. McIntosh-Smith, and F. R. Manby. "A massively multicore parallelization of the Kohn-Sham energy gradients." *J. Comp. Chem.*, 31 (10), 2008–2013, 2010.
- R. Farmber, E. Bylaska, S. Baden and students. "(Car-Parrinello on GPGPUs)." Work in progress, 2011.

MP2 on GPUs

MP2 using the resolution-of-identity approximation involves a few large GEMMs.

- L. Vogt, R. Olivares-Amaya, S. Kermes, Y. Shao, C. Amador-Bedolla, and A. Aspuru-Guzik. "Accelerating resolution-of-the-identity second-order Møller-Plesset quantum chemistry calculations with graphical processing units." *J. Phys. Chem. A*, 112 (10), 2049–2057, 2008.
- R. Olivares-Amaya, M. A. Watson, R. G. Edgar, L. Vogt, Y. Shao, and A. Aspuru-Guzik. "Accelerating correlated quantum chemistry calculations using graphical processing units and a mixed precision matrix multiplication library." *J. Chem. Theo. Comp.*, 6 (1), 135–144, 2010.
- A. Koniges, R. Preissl, J. Kim, D. Eder, A. Fisher, N. Masters, V. Mlaker, S. Ethier, W. Wang, and M. Head-Gordon. "Application acceleration on current and future Cray platforms." In *CUG 2010*, Edinburgh, Scotland, May 2010.

QMC on GPUs

QMC is ridiculously parallel at the node-level hence implementation of the kernel implies the potential for scaling to many thousands of GPUs.

- A. G. Anderson, W. A. Goddard III, and P. Schröder. "Quantum Monte Carlo on graphical processing units." *Comp. Phys. Comm.*, 177 (3), 298–306, 2007.
- A. Gothandaraman, G. D. Peterson, G. Warren, R. J. Hinde, and R. J. Harrison. "FPGA acceleration of a quantum Monte Carlo application." *Par. Comp.*, 34 (4-5), 278–291, 2008.
- K. Esler, J. Kim, L. Shulenburger, and D. Ceperley. "Fully accelerating quantum Monte Carlo simulations of real materials on GPU clusters." *Comp. Sci. Eng.*, 99 (PrePrints), 2010.

CC on GPUs

Since we started this project, two groups have implemented non-iterative triples corrections to CCSD on GPUs. These procedures involve a few very large GEMMs and a reduction. At least two other groups are working on CC on GPUs but have not reported any results.

- M. Melicherčík, L. Demovič, and P. N. Michal Pitoňák. "Acceleration of CCSD(T) computations using technology of graphical processing unit." 2010.
- W. Ma, S. Krishnaoorthy, O. Villa, and K. Kowalski. "GPU-based implementations of the regularized CCSD(T) method: applications to strongly correlated systems." Submitted to *J. Chem. Theo. Comp.*, 2010.
- A. E. DePrince and J. .R. Hammond. "Coupled-cluster theory on graphics processing units I. The coupled-cluster doubles method." Submitted to *J. Chem. Theo. Comp.*, 2010.

Gaussian Integrals on GPUs

Quantum chemistry methods spend a lot of time generating matrix elements of operators in a Gaussian basis set. All published implementations are either closed-source (commercial) or the source is unpublished, otherwise we would be using these in our code.

- I. S. Ufimtsev and T. J. Martínez. "Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation." *J. Chem. Theo. Comp.*, 4 (2), 222–231, 2008.
- A. V. Titov, V. V. Kindratenko, I. S. Ufimtsev, and T. J. Martínez. "Generation of kernels for calculating electron repulsion integrals of high angular momentum functions on GPUs – preliminary results." In *Proc. SAAHPC 2010*, pages 1–3, 2010.
- A. Asadchev, V. Allada, J. Felder, B. M. Bode, M. S. Gordon, and T. L. Windus. "Uncontracted Rys quadrature implementation of up to G functions on graphical processing units." *J. Chem. Theo. Comp.*, 6 (3), 696–704, 2010.

Coupled-cluster theory

The coupled–cluster (CC) wavefunction ansatz is

$$|CC\rangle = e^{T}|HF\rangle$$

where $T = T_1 + T_2 + \cdots + T_n$.

T is an excitation operator which promotes n electrons from occupied orbitals to virtual orbitals in the Hartree-Fock Slater determinant.

Inserting $|CC\rangle$ into the Schödinger equation:

$$\hat{H}e^{T}|HF\rangle = E_{CC}e^{T}|HF\rangle \qquad \hat{H}|CC\rangle = E_{CC}|CC\rangle$$

Coupled-cluster theory

Projective solution of CC:

$$\begin{aligned} E_{CC} &= \langle HF|e^{-T} H e^{T}|HF\rangle \\ 0 &= \langle X|e^{-T} H e^{T}|HF\rangle \quad (X = S, D, \ldots) \end{aligned}$$

CCD is:

$$\begin{aligned} E_{CC} &= \langle HF|e^{-T_2} H e^{T_2}|HF\rangle \\ 0 &= \langle D|e^{-T_2} H e^{T_2}|HF\rangle \end{aligned}$$

CCSD is:

$$\begin{aligned} E_{CC} &= \langle HF|e^{-T_1-T_2} H e^{T_1+T_2}|HF\rangle \\ 0 &= \langle S|e^{-T_1-T_2} H e^{T_1+T_2}|HF\rangle \\ 0 &= \langle D|e^{-T_1-T_2} H e^{T_1+T_2}|HF\rangle \end{aligned}$$

Notation

$$\begin{aligned} H &= H_1 + H_2 \\ &= F + V \end{aligned}$$

F is the Fock matrix. CC only uses the diagonal in the canonical formulation.

V is the fluctuation operator and is composed of two-electron integrals as a 4D array.

V has 8-fold permutation symmetry in V_{pq}^{rs} and is divided into six blocks: V_{ij}^{kl}, V_{ij}^{ka}, V_{ia}^{jb}, V_{ij}^{ab}, V_{ia}^{bc}, V_{ab}^{cd}.

Indices $i, j, k, \ldots$ $(a, b, c, \ldots)$ run over the occupied (virtual) orbitals.

CCD Equations

$$R_{ij}^{ab} = V_{ij}^{ab} + P(ia,jb)\Big[T_{ij}^{ae} I_e^b - T_{im}^{ab} I_j^m + \frac{1}{2} V_{ef}^{ab} T_{ij}^{ef} + \frac{1}{2} T_{mn}^{ab} I_{ij}^{mn} - T_{mj}^{ae} I_{ie}^{mb} - I_{ie}^{ma} T_{mj}^{eb} + (2T_{mi}^{ea} - T_{im}^{ea}) I_{ej}^{mb} \Big]$$

$$\begin{aligned}
I_b^a &= (-2V_{eb}^{mn} + V_{be}^{mn}) T_{mn}^{ea} \\
I_j^i &= (2V_{ef}^{mi} - V_{ef}^{im}) T_{mj}^{ef} \\
I_{kl}^{ij} &= V_{kl}^{ij} + V_{ej}^{ij} T_{kl}^{ef} \\
I_{jb}^{ia} &= V_{jb}^{ia} - \frac{1}{2} V_{eb}^{im} T_{jm}^{ea} \\
I_{bj}^{ia} &= V_{bj}^{ia} + V_{be}^{im}(T_{mj}^{ea} - \frac{1}{2} T_{mj}^{ae}) - \frac{1}{2} V_{be}^{mi} T_{mj}^{ae}
\end{aligned}$$

Turning CC into GEMM

Some tensor contractions are trivially mapped to GEMM:

$$
\begin{aligned}
I^{ij}_{kl} &\mathrel{+}= V^{ij}_{ej} T^{ef}_{kl} \\
I^{(ij)}_{(kl)} &\mathrel{+}= V^{(ij)}_{(ej)} T^{(ef)}_{(kl)} \\
I^{b}_{a} &\mathrel{+}= V^{b}_{c} T^{c}_{a}
\end{aligned}
$$

Other contractions require reshaping to use BLAS:

$$
\begin{aligned}
I^{ia}_{bj} &\mathrel{+}= V^{im}_{be} T^{ea}_{mj} \\
I_{bj,ia} &\mathrel{+}= V_{be,im} T_{mj,ea} \\
J_{bi,ja} &\mathrel{+}= W_{bi,me} U_{me,ja} \\
J^{ja}_{bi} &\mathrel{+}= W^{me}_{bi} U^{ja}_{me} \\
J^{(ja)}_{(bi)} &\mathrel{+}= W^{(me)}_{(bi)} U^{(ja)}_{(me)} \\
J^{z}_{x} &\mathrel{+}= W^{y}_{x} U^{z}_{y}
\end{aligned}
$$

Alteratives to the BLAS Approach?

Vendor BLAS does GEMM 100-1000 times faster than three loops.

The permutations matter if one is not paying attention. NWChem can spend more than half the CPU time in permutations. This is partly due to data distribution across nodes. Serial codes spend less than 5% in permutations.

Permutations are N^4 mops just like GEMM but GEMM is pipelined whereas permutations are cache torture tests.

Eliminating BLAS means writing tensor contractions that can get more than half of peak. If you think your code generator can do this, please write me.

Computational cost of CCD in terms of GEMM

$o < 30$ is a small calculation.
$o > 70$ is a large calculation.

$v < 200$ is a small calculation.
$v > 600$ is a large calculation.

Biggest (o, v) calculations run with NWChem on supercomputers:

$(33, 1435)$ – octane
$(240, 1008)$ – buckyball
$(300+, 700+)$ – diporphyrin

	dimension		
count	M	N	K
1	o^2	o^2	v^2
1	o^2	v^2	v^2
1	o^2	v^2	o^2
6	ov	ov	ov
1	v	v	o^2v
1	v	o^2v	v
1	o	o	ov^2
1	o	ov^2	o

Computational cost of CCD in practice

Tiny calculation running on my 20 GF/s laptop.

$o = 15$ and $v = 90$

count	M	N	K	gigaflop	time (s)	gigaflop/s
6	1350	1350	1350	29.5	1.42	20.82
1	225	8100	8100	29.5	1.54	19.20
1	225	8100	225	0.8	0.08	9.98
1	225	225	8100	0.8	0.10	8.00
1	90	90	20250	0.3	0.05	6.46
1	90	20250	90	0.3	0.05	6.95
1	15	15	121500	0.05	0.01	5.51
1	15	121500	15	0.05	0.03	1.78

Computational cost of CCD in practice

One of the largest CC calculations ever performed.

$o = 240$ and $v = 1008$

count	M	N	K	petaflop
6	242K	242K	242K	169.90
1	58K	1016K	1016K	118.93
1	58K	1016K	58K	6.74
1	58K	58K	1016K	6.74
1	1008	1008	58M	0.12
1	1008	58M	1008	0.12
1	240	240	244M	0.03
1	240	244M	240	0.03

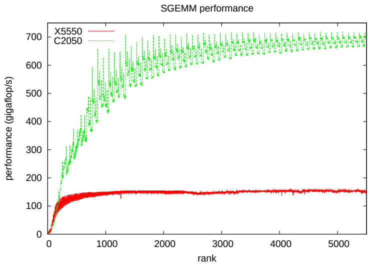
Hardware Details

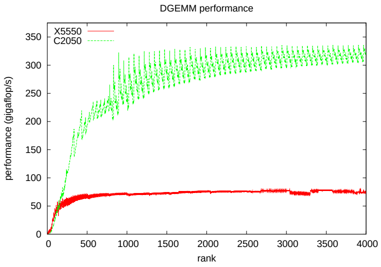
	CPU		GPU	
	X5550	2 X5550	C1060	C2050
processor speed (MHz)	2660	2660	1300	1150
memory bandwidth (GB/s)	32	64	102	144
memory speed (MHz)	1066	1066	800	1500
ECC available	yes	yes	no	yes
SP peak (GF)	85.1	170.2	933	1030
DP peak (GF)	42.6	83.2	78	515
power usage (W)	95	190	188	238

Note that power consumption is apples-to-oranges since CPU does not include DRAM, whereas GPU does.

Relative Performance of GEMM

GPU versus SMP CPU (8 threads):





Maximum:
CPU = 156.2 GF
GPU = 717.6 GF

Maximum:
CPU = 79.2 GF
GPU = 335.6 GF

We expect roughly 4-5 times speedup based upon this evaluation.

Pseudocode

```
Copy $F$ and $V$ to GPU (cudaMemcpy)
while $|\Delta T| > \epsilon_{\mathrm{thresh}}$ do
  if $V_{cd}^{ab}$ fits in GPU memory then
    Update $T$. (cublasDgemm)
  else
    for all tiles do
      Copy $V_{cd}^{ab}$ tile to GPU (cudaMemcpy)
      Contract $T_{ij}^{ef} V_{ef}^{ab}$ (cublasDgemm)
    end for
    Update $T$ with remaining terms. (cublasDgemm)
  end if
  Compute $|\Delta T|$ (cublasDnrm2)
end while
Copy amplitudes from GPU (cudaMemcpy)
$E \leftarrow T, V$
```

Comparison Codes

Package	Parallelism	Spin-free?	Partially direct?	Same equations?
Our code	SMP-BLAS	Yes	No	Yes
GAMESS	No	Yes	No	Yes
PSI3	No	Yes	No	Maybe
Molpro	GA	Yes	Yes	No
NWChem	GA	Yes	Yes	No
TCE	GA	Partially	No	No

GA parallelism means multiple processes. The ARMCI data-server requires a dedicated core, hence we run 7 processes across two X5550 sockets.

The performance of GAMESS and PSI3 does not improve significantly when multiple threads are used in BLAS.

Chemistry Details

Molecule	o	v
C_8H_{10}	21	63
$C_{10}H_8$	24	72
$C_{10}H_{12}$	26	78
$C_{12}H_{14}$	31	93
$C_{14}H_{10}$	33	99
$C_{14}H_{16}$	36	108
C_{20}	40	120
$C_{16}H_{18}$	41	123
$C_{18}H_{12}$	42	126
$C_{18}H_{20}$	46	138

- 6-31G basis set
- C_1 symmetry
- F and V from GAMESS via disk

Targeting integration with PSI3 to avoid I/O. May also compute blocks of V on-the-fly.

CCD Performance Results

Iteration time in seconds

		our DP code			X5550		
o	v	C2050	C1060	X5550	Molpro	TCE	GAMESS
21	63	0.3	0.8	1.3	2.3	5.1	6.2
24	72	0.5	1.5	2.5	4.8	10.6	12.7
26	78	0.8	2.5	3.5	7.1	16.2	19.7
31	93	2.0	7.1	10.0	17.6	42.0	57.7
33	99	2.7	10.2	13.9	29.9	59.5	78.5
36	108	4.5	16.7	21.6	41.5	90.2	129.3
40	120	8.8[a]	29.9	40.3	103.0	166.3	238.9
41	123	10.5[a]	35.9	50.2	83.3	190.8	279.5
42	126	12.7[b]	42.2[a]	50.3	111.8	218.4	329.4
46	138	20.1[b]	73.0[a]	86.6	157.4	372.1	555.5

[a] $oo \cdot vv \cdot vv$ term was tiled.

[b] Parts of V pushed every iteration (and [a]).

Numerical Precision versus Performance

Iteration time in seconds

	C1060		C2050		X5550	
molecule	SP	DP	SP	DP	SP	DP
C_8H_{10}	0.2	0.8	0.2	0.3	0.7	1.3
$C_{10}H_8$	0.4	1.5	0.2	0.5	1.3	2.5
$C_{10}H_{12}$	0.7	2.5	0.4	0.8	2.0	3.5
$C_{12}H_{14}$	1.8	7.1	1.0	2.0	5.6	10.0
$C_{14}H_{10}$	2.6	10.2	1.5	2.7	8.4	13.9
$C_{14}H_{16}$	4.1	16.7	2.4	4.5	12.1	21.6
C_{20}	6.7	29.9	4.1	8.8	22.3	40.3
$C_{16}H_{18}$	9.0	35.9	5.0	10.5	28.8	50.2
$C_{18}H_{12}$	10.1	42.2	5.6	12.7	29.4	50.3
$C_{18}H_{20}$	17.2	73.0	10.1	20.1	47.0	86.6

This the apples-to-apples CPU vs. GPU.

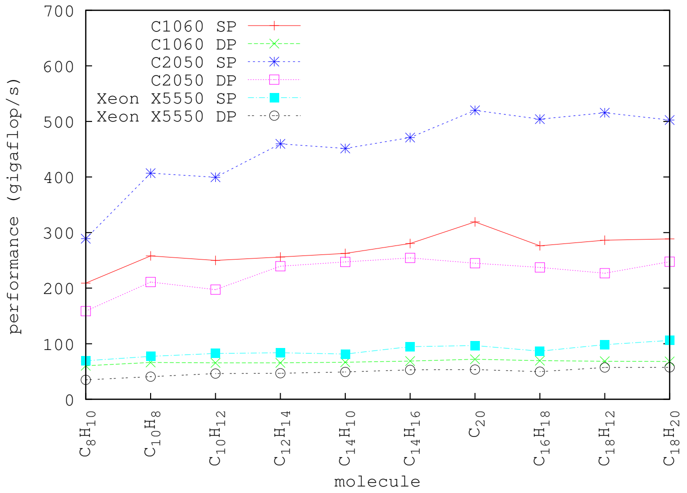
Numerical Precision versus Calculation Accuracy

We want at least 10^{-6} (μH) convergence in the residual.

	Error in μH	
molecule	SP	Mixed
C_8H_{10}	0.05	0.01
$C_{10}H_8$	-0.42	-0.04
$C_{10}H_{12}$	-0.13	-0.02
$C_{12}H_{14}$	-0.30	-0.04
$C_{14}H_{10}$	-3.74	-0.35
$C_{14}H_{16}$	-1.00	-0.16
C_{20}	-1.43	0.09
$C_{16}H_{18}$	-2.66	-0.44
$C_{18}H_{12}$	-15.03	-1.30
$C_{18}H_{20}$	-5.72	-0.91

Mixed means we converge in SP ($>$ 10 iter.), promote and turn the crank once in DP.

Performance Summary



Skeleton CCD on POWER7

$o = 46$ and $v = 138$ on the best CPU money can buy:

term	M	N	K	gigaflops	seconds	gigaflop/s
6 $ovovov$	6348	6348	6348	3069.7	10.78	284.8
$o^2v^2v^2$	2116	19044	19044	1534.8	5.64	272.2
$o^2v^2o^2$	2116	19044	2116	170.5	0.63	271.5
$o^2o^2v^2$	2116	2116	19044	170.5	0.62	275.1
vvo^2v	138	138	292008	11.1	0.27	40.8
$ooov^2$	46	46	876024	3.7	0.26	14.3
vo^2vv	138	292008	138	11.1	0.06	176.3
oov^2o	46	876024	46	3.7	0.04	83.8

CCD versus CCSD

Iteration time in seconds

		Molpro		TCE		GAMESS	
o	*v*	CCD	CCSD	CCD	CCSD	CCD	CCSD
21	63	2.3	2.4	5.1	8.4	6.2	7.2
24	72	4.8	5.1	10.6	16.8	12.7	15.3
26	78	7.1	7.2	16.2	25.2	19.7	23.6
31	93	17.6	19.0	42.0	64.4	57.7	65.1
33	99	29.9	31.0	59.5	90.7	78.5	92.9
36	108	41.5	43.1	90.2	129.2	129.3	163.7
40	120	103.0	102.0	166.3	233.9	238.9	277.5
41	123	83.3	84.1	190.8	267.9	279.5	345.8
42	126	111.8	116.2	218.4	304.5	329.4	380.0
46	138	157.4	161.4	372.1	512.0	555.5	641.3

Preliminary CPU+GPU CCSD

		Iteration time in seconds					
o	v	Hybrid	Molpro	NWChem	PSI3	TCE	GAMESS
21	63	0.72	2.4	3.6	7.9	8.4	7.2
24	72	1.28	5.1	8.2	17.9	16.8	15.3
26	78	1.88	7.2	11.3	23.6	25.2	23.6
31	93	4.28	19.0	29.4	54.2	64.4	65.1
33	99	6.44	31.0	49.1	61.4	90.7	92.9
36	108	9.43	43.1	65.0	103.4	129.2	163.7
40	120	14.98	102.0	175.7	162.6	233.9	277.5
41	123	18.87	84.1	117.5	192.4	267.9	345.8
42	126	19.82	116.2	178.6	216.4	304.5	380.0
46	138	31.34	161.4	216.3	306.9	512.0	641.3

Our hybrid DP code uses streams and overlaps computation.

Preliminary MPI+GPU CCD

		N_{GPU}		
o	v	1	7	speedup
21	63	0.157	0.070	2.24
24	72	0.248	0.105	2.36
26	78	0.408	0.150	2.72
31	93	1.017	0.309	3.29
33	99	1.506	0.467	3.22
36	108	2.431	0.660	3.68
40	120	4.140	1.118	3.70
41	123	4.951	1.302	3.80
42	126	5.594	1.447	3.87
46	138	10.121	2.362	4.28

Summary

- Implemented CCD on GPU and CPU using vendor BLAS. Implementation quality very similar.
- Compared our code to theirs *as directly as possible*.
- Apples-to-apples CPU vs. GPU is 4-5x (as predicted).
- Apples-to-oranges us versus them shows 7-10x.
  Our CPU code is 2x, so again 4-5x is from GPU.
- MPI scaling efficiency is far from perfect but improves with weak-scaling.

We've done due deligence for CCD, but for CCSD, we're targeting only time-to-solution on the same hardware. We will not apologize for other implementations poor design, lack of threading, etc. If our equations require more flops but run faster, so be it.

What did we learn?

- Put down the legacy code.
  Step away from the legacy code.
  Turn and run away from the legacy code.
- Do not write a CPU code that uses GPUs.
  Do write a GPU code.
  Consider backfilling small tasks on CPU.
- Do not try to compensate for bad vendor libraries.
  Complain and wait.
- Take the path of least resistance (e.g. CUBLAS) and understand the big picture.

To petascale and beyond?

Today:

- NWChem scales efficiently to 200K cores with process-based parallelism alone.
- Global Arrays leads to flat communication pattern across the machine.
- Dynamic load-balancing prevents locality optimization.

Tomorrow:

- Hybrid approach is already necessary on XE6 (24 cores), so moving to fat-node algorithms is important independent of GPUs.
- Need to apply inspector-executor approach to realize static load-balancing at runtime.
- Need Gaussian integral kernels on GPU to avoid memory storage/motion wall.

Acknowledgments

Argonne NATIONAL LABORATORY

Director's Fellowship (JRH)
Computational Fellowship (AED)
Breadboard cluster
Fusion cluster

NERSC

Dirac cluster

SAAHPC 2011

What: Symposium on Application Accelerators in HPC
When: July 19-21, 2011
Where: Knoxville, Tennessee

Application Sessions:
Computational chemistry on accelerators (Chair: Jeff Hammond, ALCF)
Lattice QCD (Chair: Steven Gottlieb, Indiana University, Bloomington)
Weather and climate modeling (Chair: John Michalakes, NREL)
Bioinformatics (Chair: TBD)

Submissions:
Short paper (up to 4 pages, for a poster presentation)
Long paper (up to 10 pages, for an oral presentation)

Deadlines:
Submissions due: May 6, 2011
Presentation acceptance notification: June 6, 2011
Final papers due: June 30, 2011