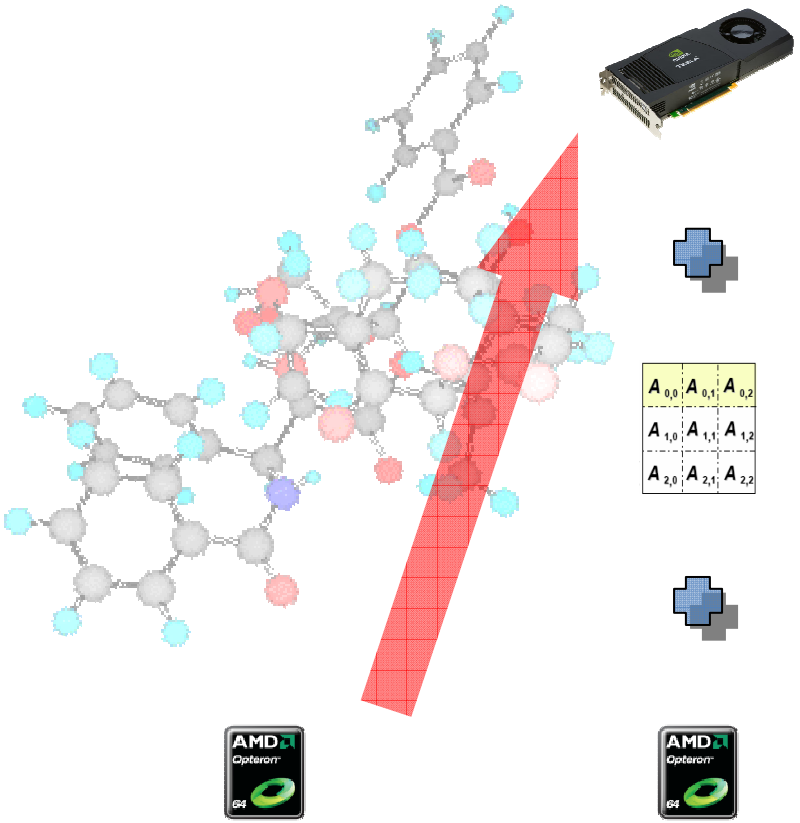
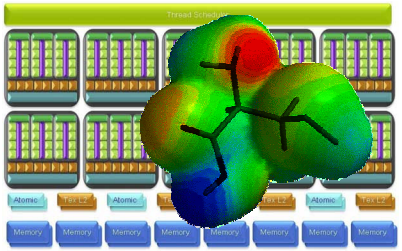
Quantum Chemistry on the GPU
Accelerating DFT Calculations

Zhengting Gan
Yihan Shao
Jing Kong
Q-Chem Inc.

Roberto Olivares-Amaya
Alan Aspuru-Guzik
Harvard University



Outline



Background of GPU computing and DFT

BLAS3 based DFT XC algorithm

Hybrid computing solution

The emerging trend in HPC

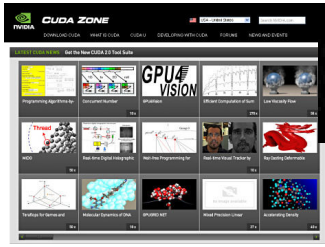
Q-CHEM
A Quantum Leap Into the Future of Chemistry

Roadrunner, IBM, 1.02PF
PowerXCell 8i +Opteron

Tsubame, Sun, 78TF
NVidia S1070 +
Opteron + Clearspeed

Hybrid architectures in supercomputing

Bring HPC mainstream

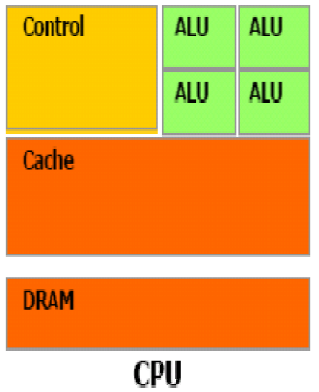


Growing number of accelerated applications

Hardware acceleration

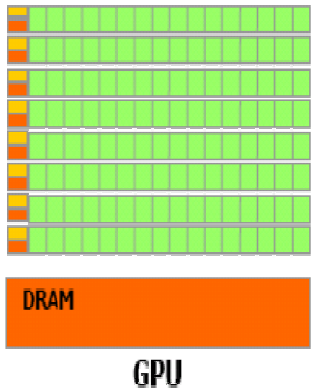
General computing CPU

- Branch prediction
- Out of order execution
- Data cache: L1,L2,L3
- SMT



Specialized hardware (GPU)

- Simple SIMD instructions
- Small but programmable cache
- Latency hiding (register reuse, multi-threading)



Remaining Issues III
Heterogeneous computing solution

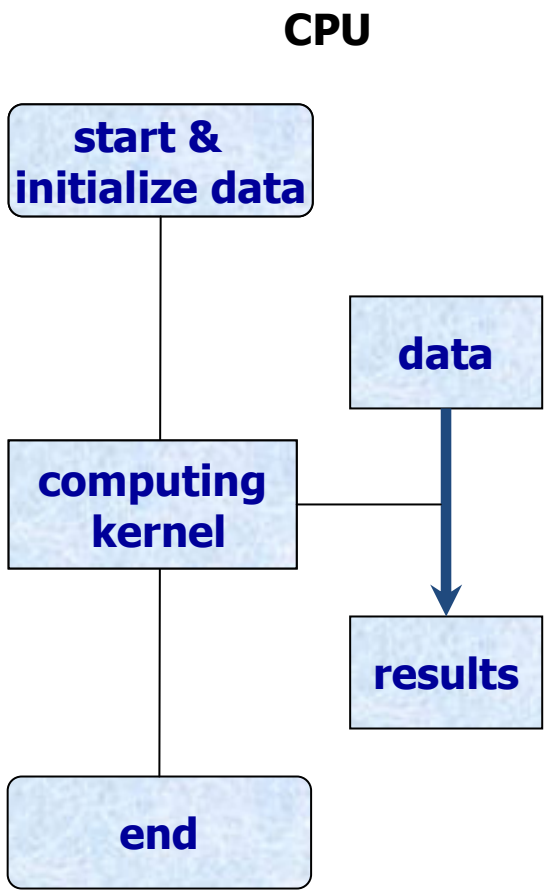


Fig. CPU computing workflow

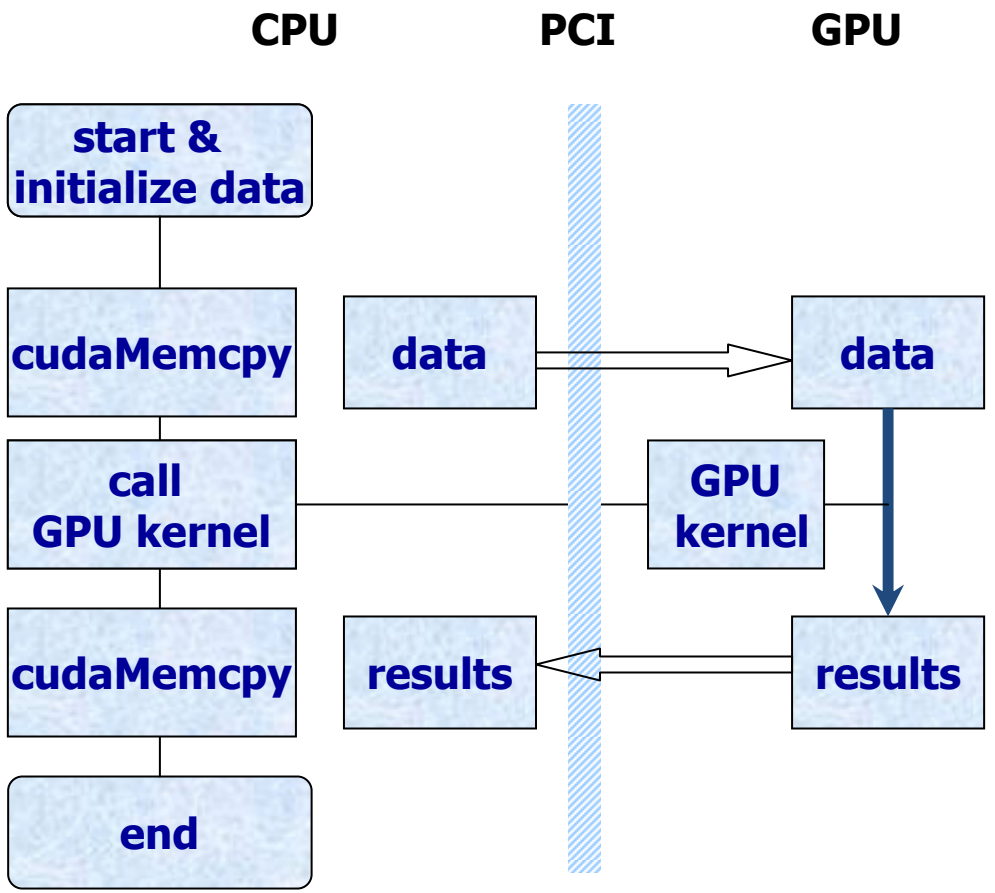


Fig. GPU computing workflow

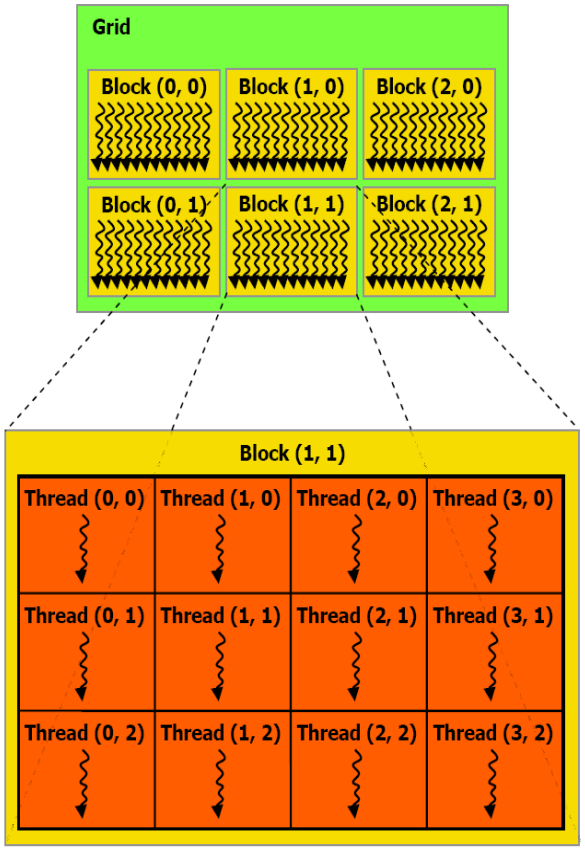
CUDA computing on Tesla

Hardware abstraction (C1060):

- 30 cores (Processor array)
- Each core does SIMD op.

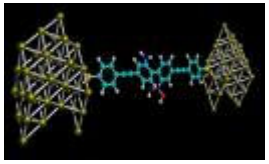
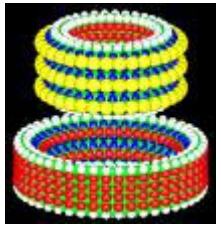
Software level abstraction:

- Create NBLOCK tasks (NBLOCK >24)
- Each task consists of BLOCK_SIZE threads (BLOCK_SIZE=<512, more == better)
- Task parallel is OK at block level
- Inside block, do DATA parallel (Vector)



About DFT

DFT is the most widely used quantum chemistry methods

Balance of accuracy and efficiency

The importance of time to solution

- **Geometry optimization**
- **Reaction path search**
- **QM/MM simulation**

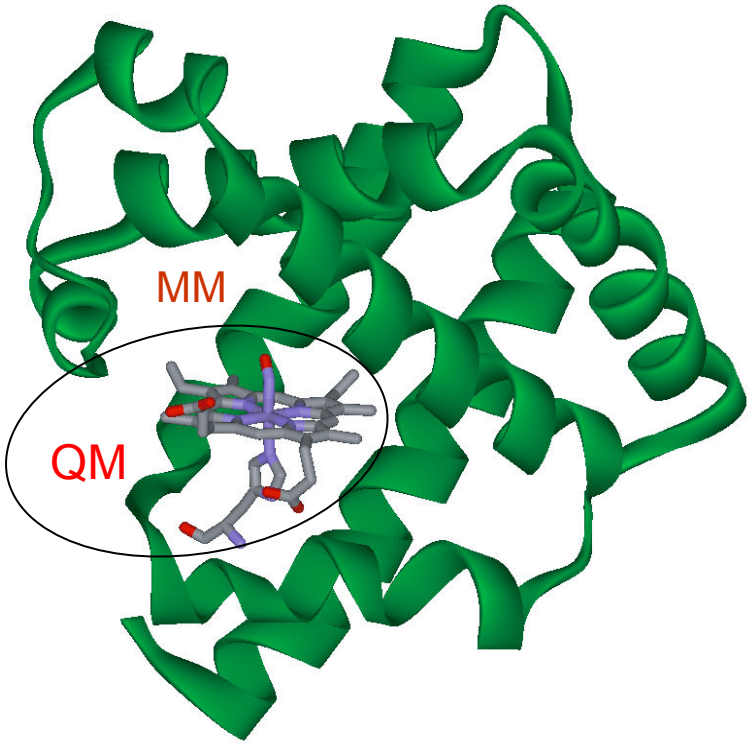
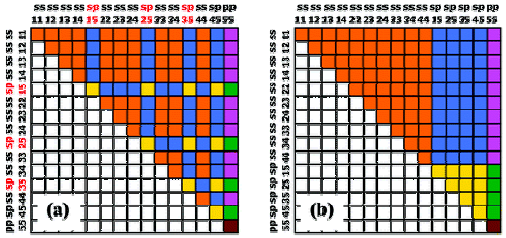


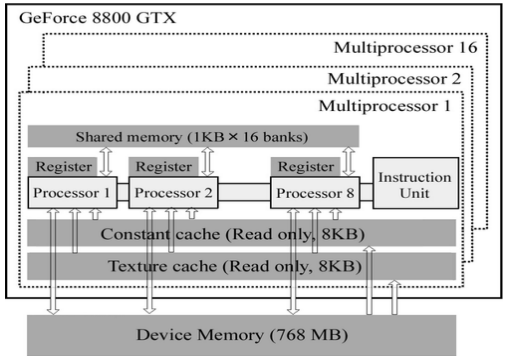
Fig. The active site (QM) of myoglobin (MM).

Existing works on DFT acceleration

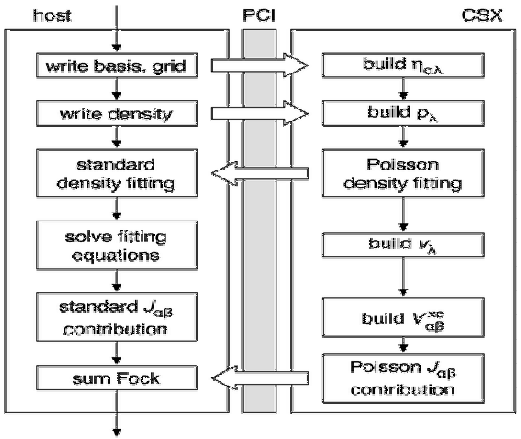
- **Two-electron integral evaluation using GPU.**
  Ivan S. Ufimtsev and Todd J. Martínez



- **XC Fock matrix building accelerated by GPU.**
  Koji Yasuda

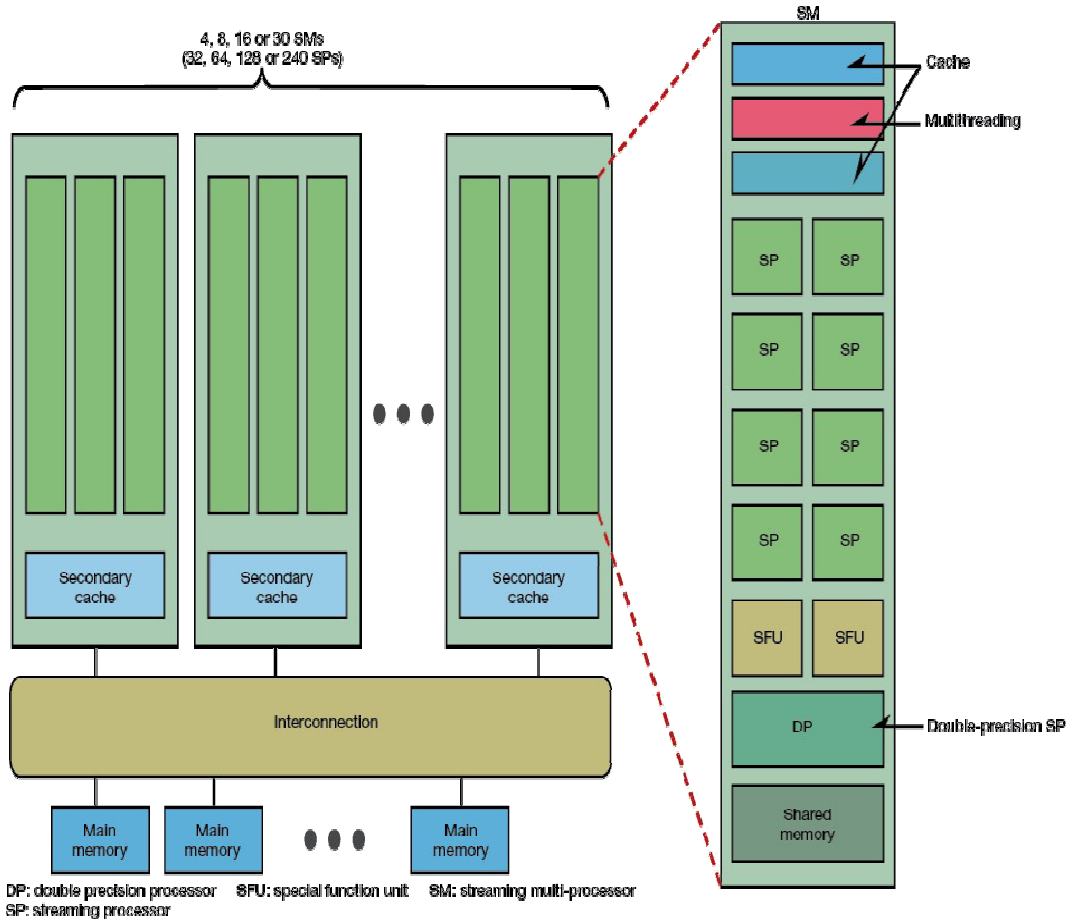


- **DFT using grid-based density-fitted Poisson accelerated by ClearSpeed board.**
  Philip Brown, Christopher Woods, Simon McIntosh-Smith and Frederick R. Manby



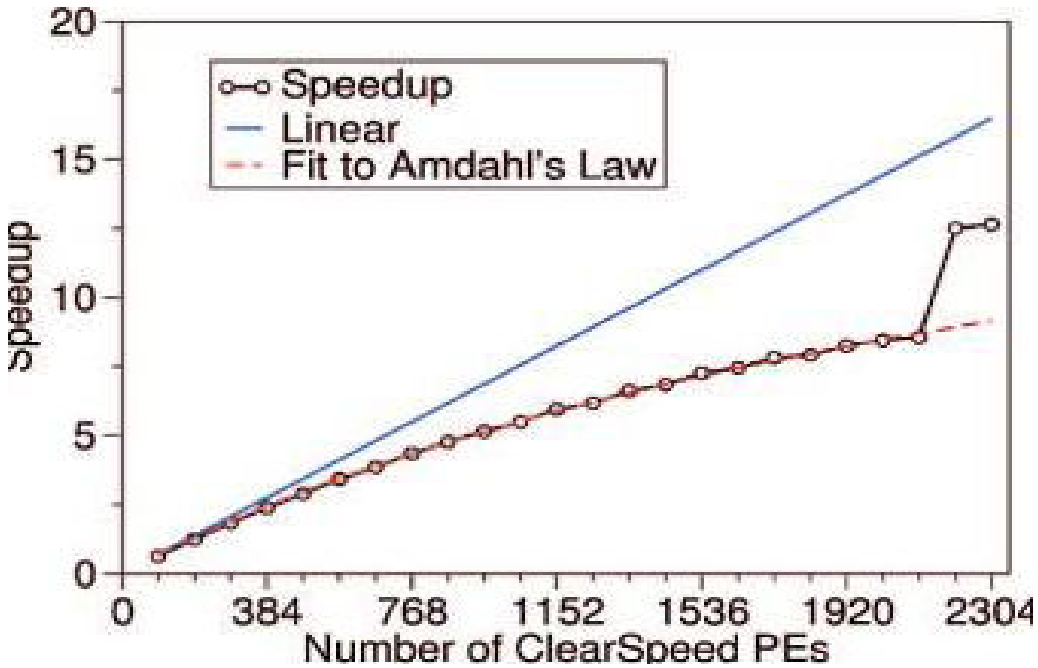
Remaining Issues I
Double precision implementation

- **Efficient implementation in double precision is needed**



Remaining Issues II
Basis set evaluation

- **Basis set computing on the grid remains a performance and scalability bottleneck**



"We can also observe a significant performance gain for the last two points, where enough memory becomes available to **store the AOs** on the grid, removing a significant portion of the work performed by the accelerators."
J. Chem. Theory Comput., 2008, 4 (10), 1620-1626

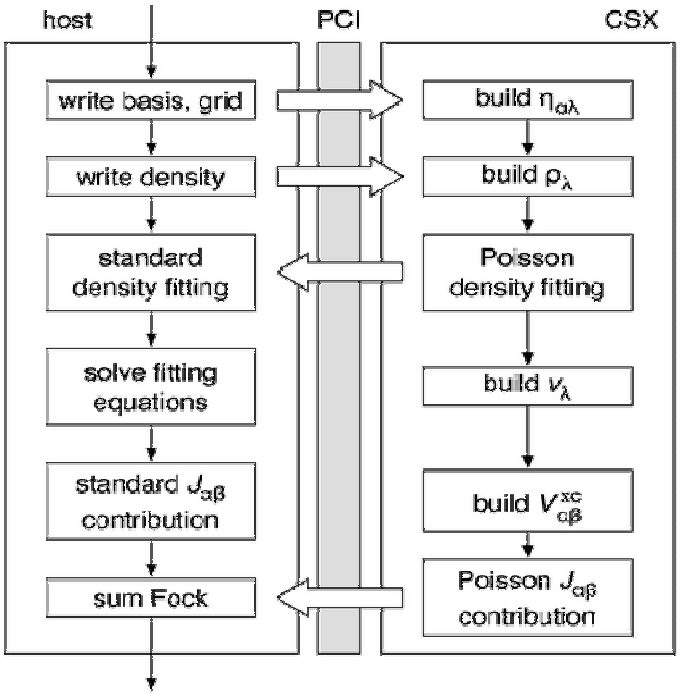
Remaining Issues III
Heterogeneous computing solution

Heterogeneous parallel computing strategy

CPU & GPU work
- **synchronously**
- **asynchronously**

Heterogeneous computing programming

Programming model for Multicore +
- **GPUs**
- **(Cell BE, ClearSpeed, FPGA)**



Control flow of accelerated GDFT implementation

DFT Algorithm Outline

Setup molecular grid (Atomic grid with becke weights)

Loop over grid batch

- **Determine significant shells**
- **Compute basis set values on the grid**
- **Determine significant shell pairs that overlap with the grid**
- **Compute functional variables**

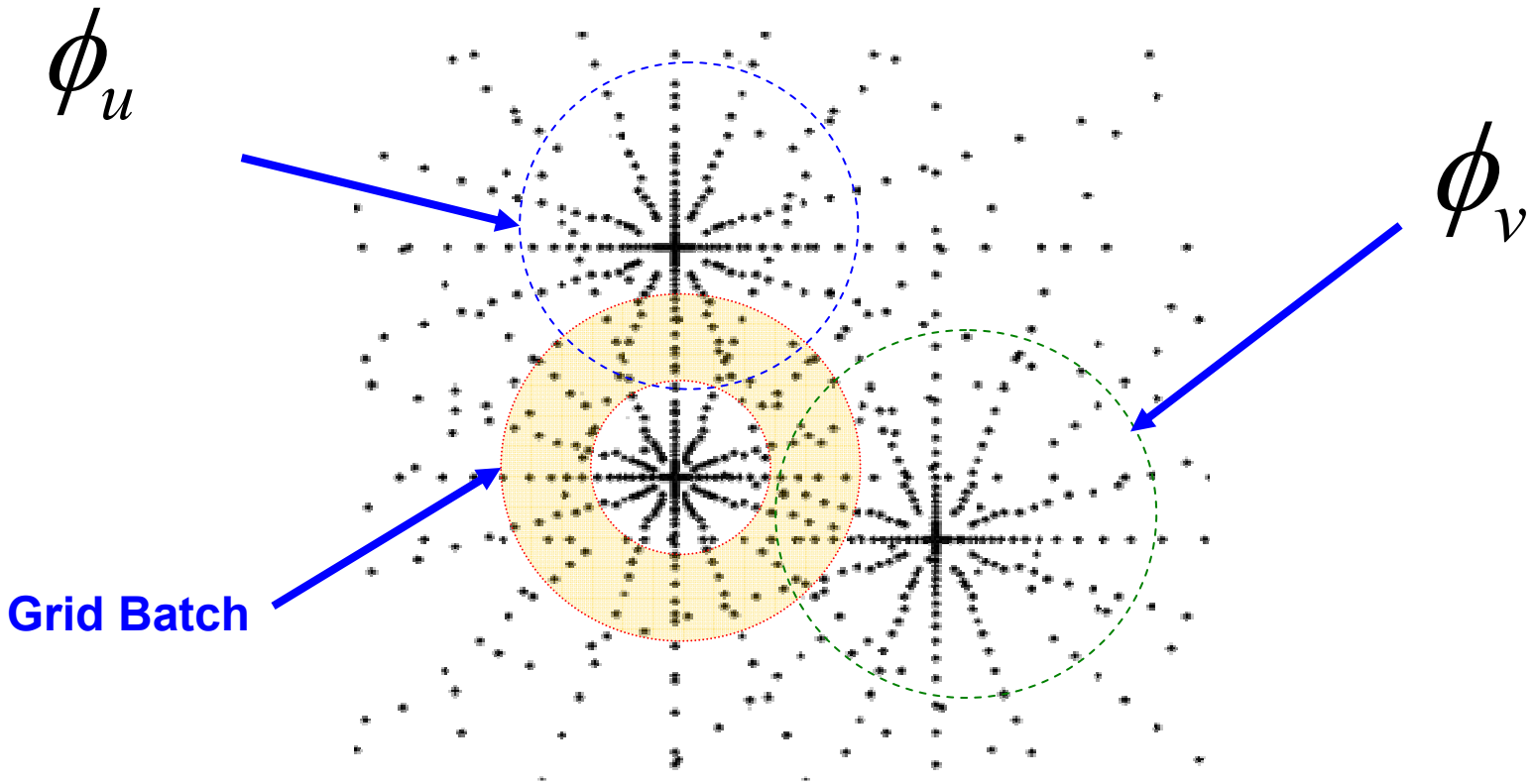
$$\rho(i) = \sum_{u,v} P_{uv}\phi_u(i)\phi_v(i)$$

- **Compute functional values and derivatives on the grid**
- **Build XC Fock matrix**

$$F_{uv}^{xc} = \sum_i \phi_u(i) v_{xc}(i) \phi_v(i)$$

DFT Algorithm
Shell pair screening

$$\rho(i) = \sum_{u,v} P_{uv} \phi_u(i) \phi_v(i)$$

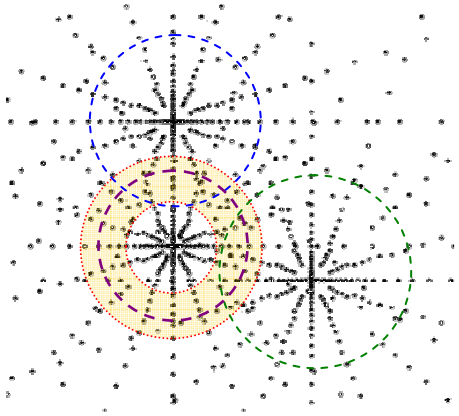


DFT Algorithm
BLAS3 vs. Screening

Shell Pair Screening

- eliminate unnecessary work. (20% less)
- BLAS1 like kernel $\rho(i) = \sum_{u,v} P_{uv}\phi_u(i)\phi_v(i)$
- overhead screening shell pairs

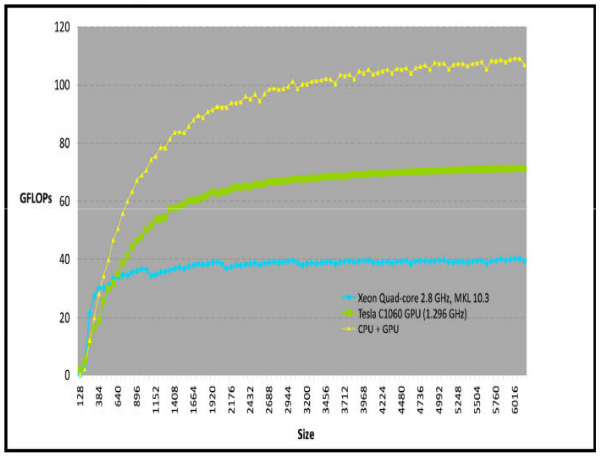
Fig. Grid, grid batch, significant shells and shell pairs.



BLAS3 based algorithm ✓

- DGEMM kernel (3-4X faster)
- gather/scatter vector operation
- portable performance on CPU and GPU

CUBLAS performance - DGEMM



Basis set evaluation Algorithm & difficulties

Cartesian basis functions:

$$\phi = x^l y^m z^n \sum c_\alpha e^{-\alpha r^2} \qquad L = l + m + n$$

L

$$\phi_x' = x^{l+1} y^m z^n (-2\sum \alpha c_\alpha e^{-\alpha r^2}) + l x^{l-1} y^m z^n \sum c_\alpha e^{-\alpha r^2}$$

L+1 L-1

Angular functions need to be kept in memory for efficiency

f type	*s,p,d,f,g*	***35*** *functions*
g type	*s,p,d,f,g,h*	***56*** *functions*

Basis set evaluation
Revised algorithm

Solution: **High momentum angular functions computed on the fly**

$$\phi = x^l y^m z^n \sum c_\alpha e^{-\alpha r^2} \qquad L = l + m + n$$

L

$$\phi_x' = x^l y^m z^n (-2x \sum \alpha c_\alpha e^{-\alpha r^2}) + l x^{l-1} y^m z^n \sum c_\alpha e^{-\alpha r^2}$$

L ← *x,y,z* — L-1

Angular functions to be kept in memory:

f type	*d*	***6*** *functions*
g type	*d*	***6*** *functions*

Basis set evaluation Cartesian to Pure

Cartesian to Pure function Transform

$$\phi_p(i) = \sum_c \phi_c(i)\ C2P(c,p)$$

Matrix Multiplication, DGEMM

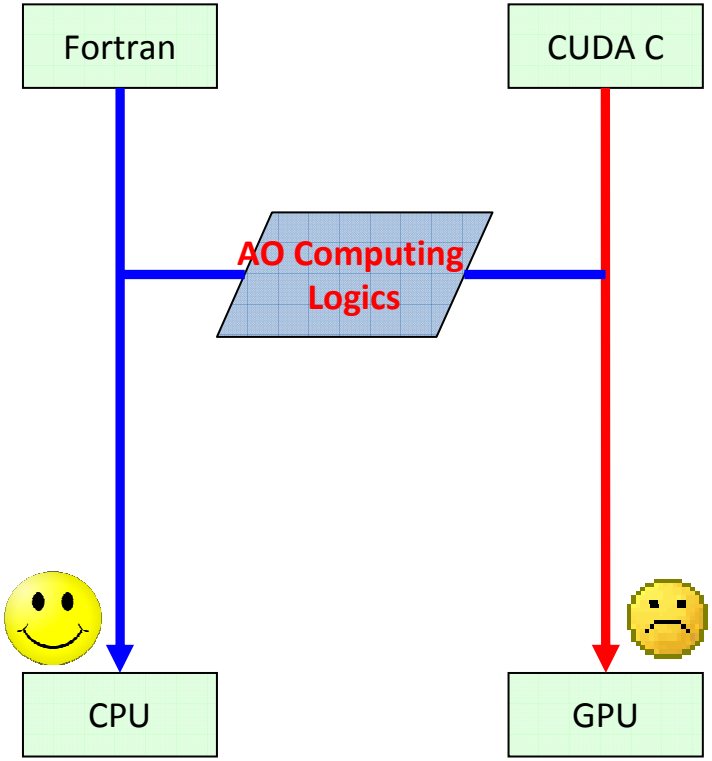
Use DGEMM or NOT?

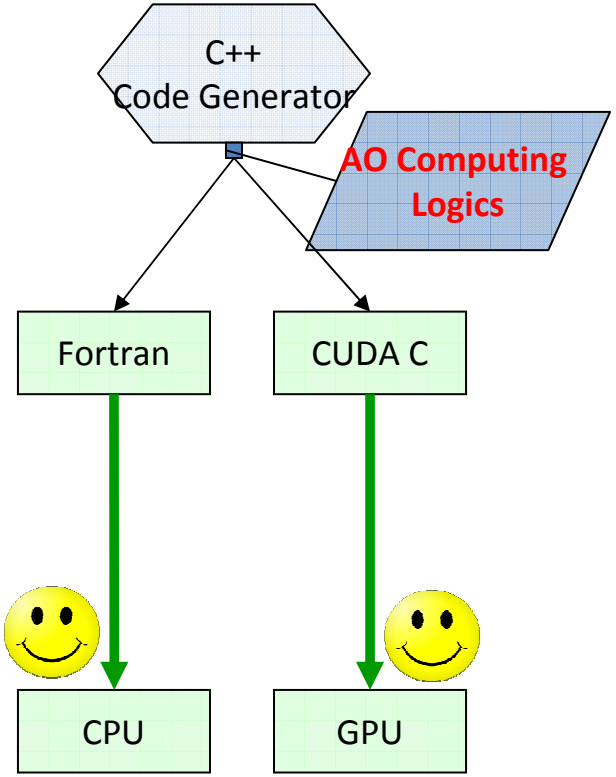
C2P transform matrix is very sparse

DGEMM is not effective for small matrices

Sparsity must be exploited!
But, how to compute the index efficiently on GPU?

Basis set evaluation
Coding AO logics





Problem:

-- Complicated logics for AO
-- Redundant index calcu. on GPU

Solution:

-- Move AO logics out of GPU
-- Computer generates optimized code

Hybrid computing solution TOOM model

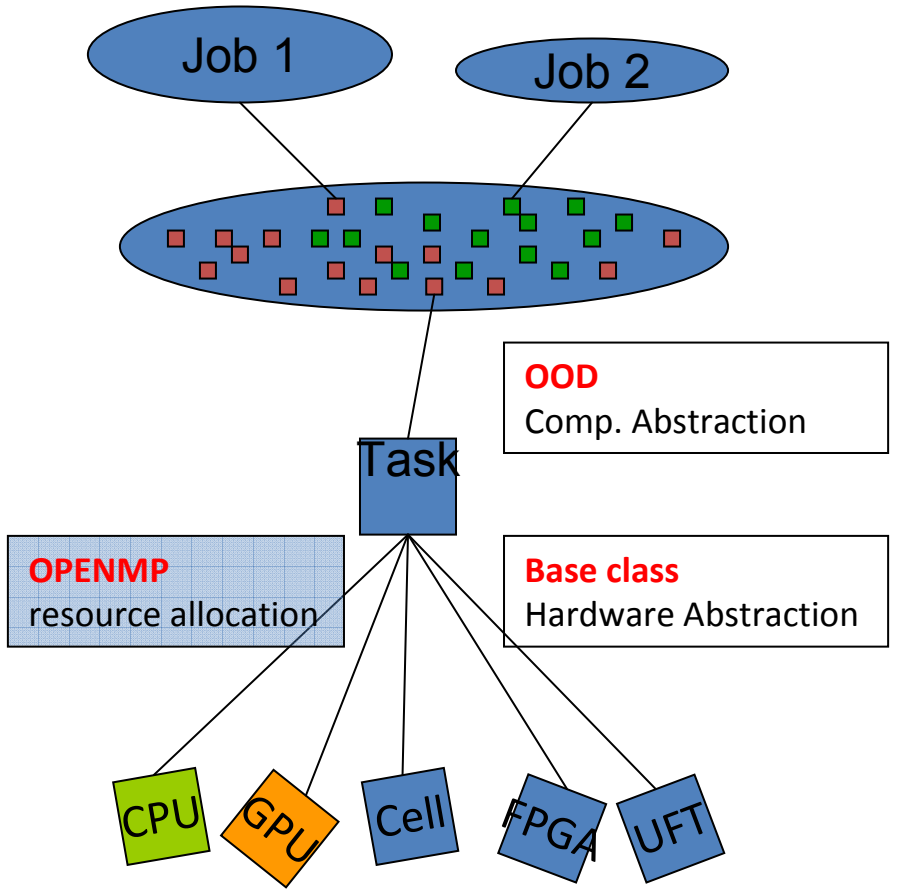
Task based programming

Object-oriented design

OPENMP for multithreading

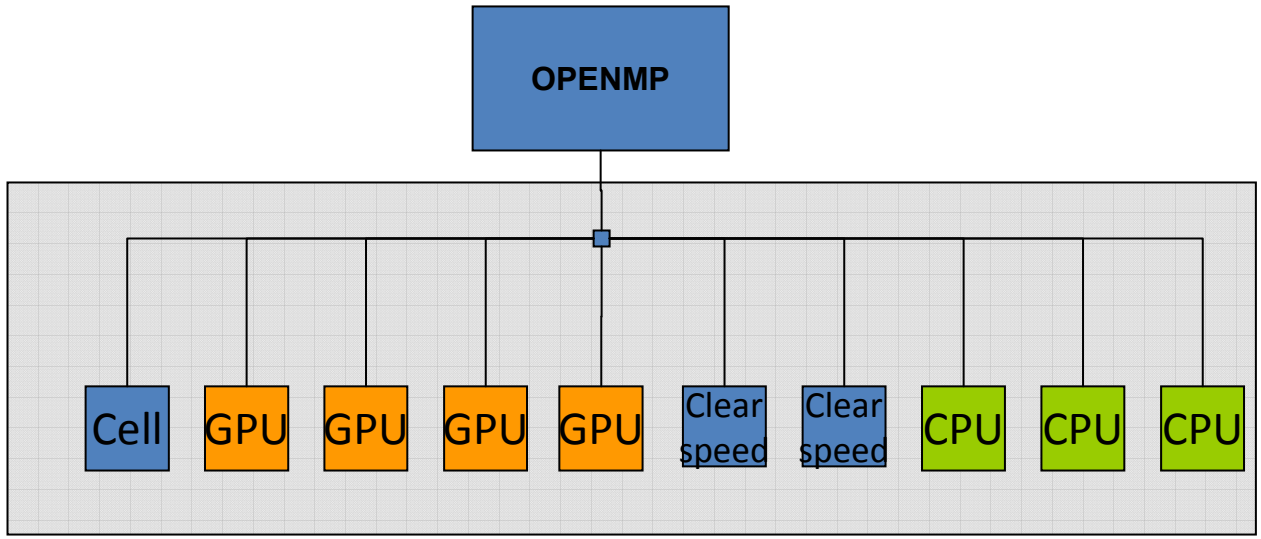
MPI for cross node comm.

Mature technologies
Correct integration is the key



Hybrid computing solution
Resource Management in TOOM

- **Hardware resource managed by OPENMP threads**
- **Threads are populated with objects representing hardwares**



Hybrid computing solution
OOD & Taks based computing

```
Class CPUXCTask: public XCTask
{
    // overloaded functions below
    alloc_memory(); //on CPU

    run_task(int id){ //base class
      calcu_bas(id);
      calcu_den(id);
      calcu_fnl(id);
      calcu_fxc(id);
    };

    calcu_bas(int id);
    calcu_den(int id);
    calcu_fxc(int id);
};
```

```
Class GPUXCTask: public XCTask
{
    // overloaded functions below
    alloc_memory(); //on GPU

    run_task(int id){ //base class
      calcu_bas(id);
      calcu_den(id);
      calcu_fnl(id);
      calcu_fxc(id);
    };

    calcu_bas(int id);  //on GPU
    calcu_den(int id);
    calcu_fxc(int id);
};
```

Hybrid computing solution TOOM code example

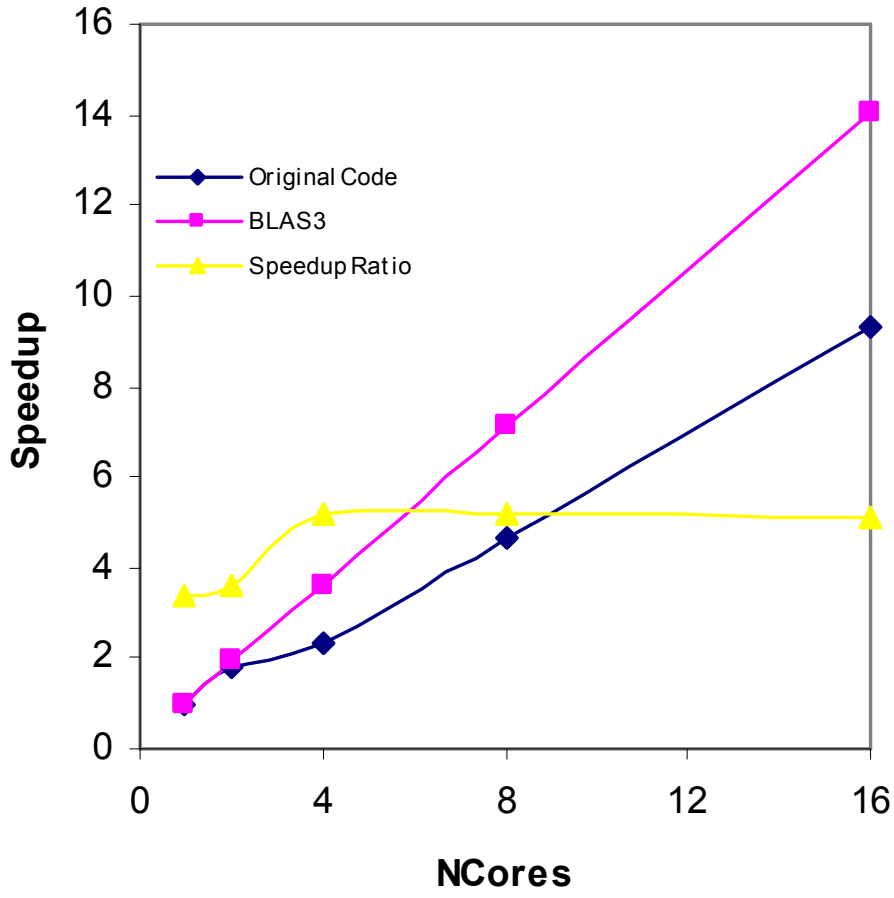
```
CalcuDFTXC
{
    #pragma omp parallel                    OPENMP
    do {
        XCTask* pTask;

        itask=getDLBNext();                 MPI        multiple Nodes

        int id=get_threadid();
        if ( id < NCards )
            pTask=new XCTaskGPU();                     multiple GPUs
        else
            pTask=new XCTaskCPU();                     multiple CPUs

        pTask->run_task(itask);
    } while (! end)
}
```

Performance Analysis
BLAS3 vs. Screening Algorithm



Linear speedup

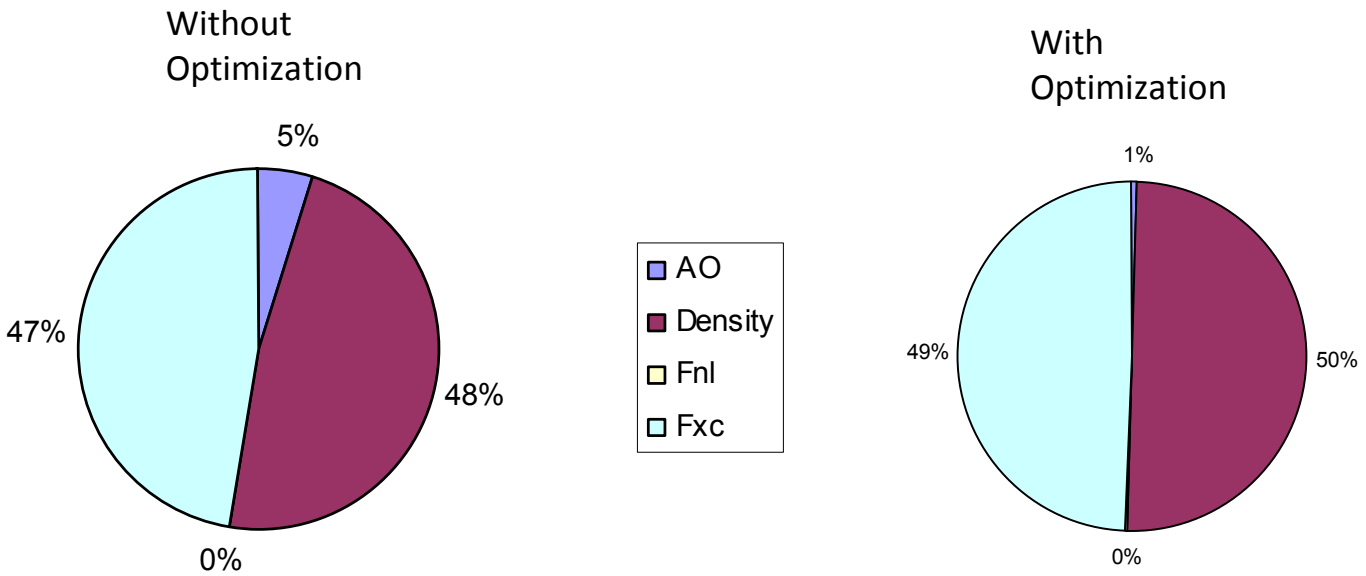
3X faster for serial job

Much improved scaling on single node

Performance comparison of BLAS3 and shell-pair sceenning algorithm on AZT calculations using aug-cc-pVTZ and SG-1 on Quadcore Opteron nodes

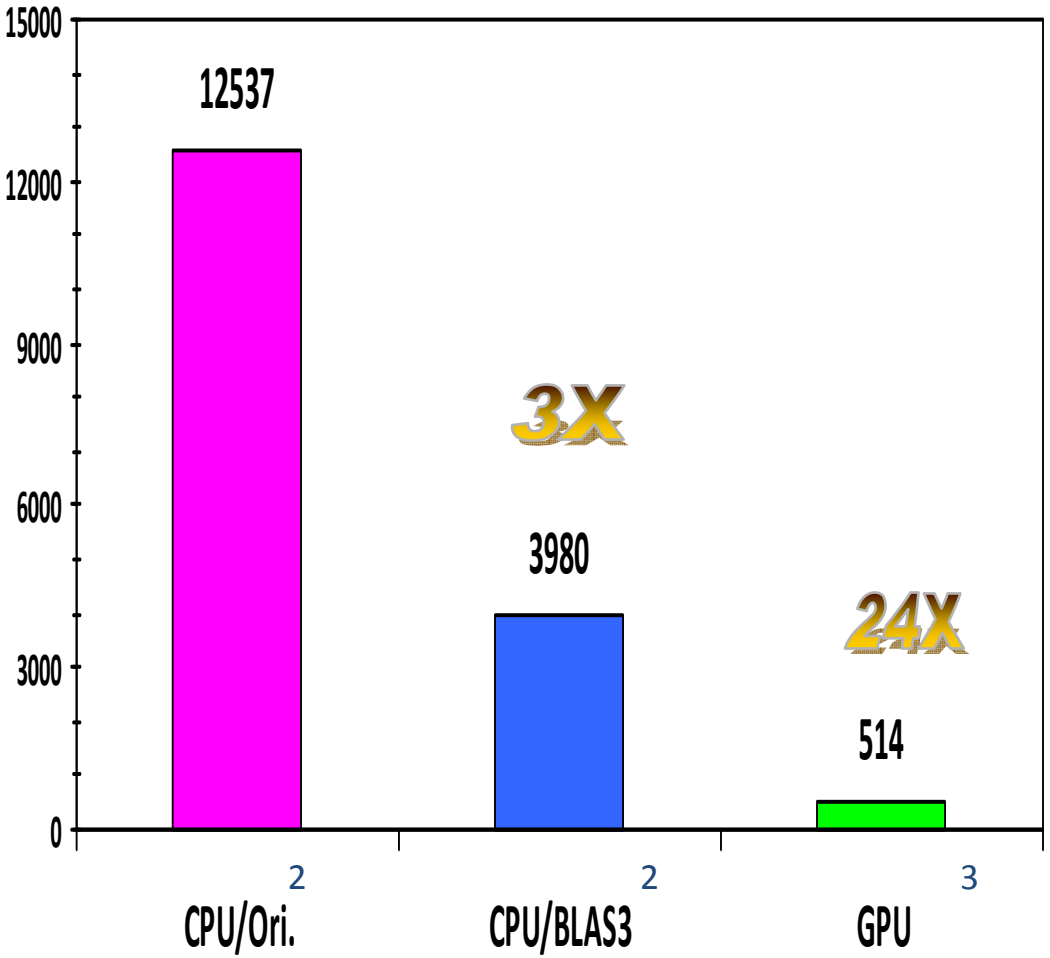
Performance Analysis
AO Optimization

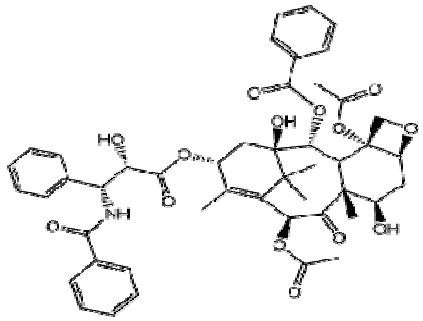
Percentage of time spent with and without AO optimization



Taxol calculation using aug-cc-pVTZ basis running on AMD Phenom II X4.

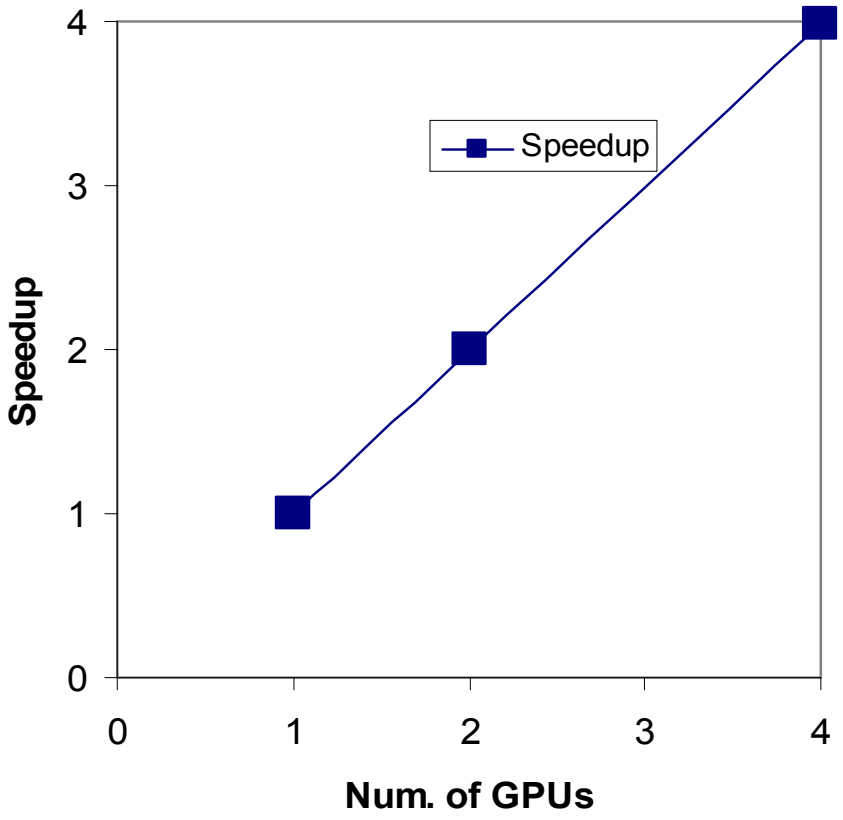
Performance Analysis
Acceleration on Single CPU & GPU

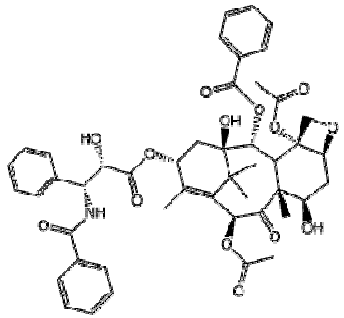




Molecule: Taxol(C47H51NO14)
Basis set: aug-cc-pVTZ
Number of basis functions: 4025
Grid: Lebedev grid (75,302)
XC: B-LYP
1. Timing is for building XC part of Fock matrix per SCF iteration.
2. Serial code using single core of AMD Phenom II X4 940 3.0GHz.
3. Using Tesla C1060 GPU

Performance Analysis
Scaling on Multiple GPUs





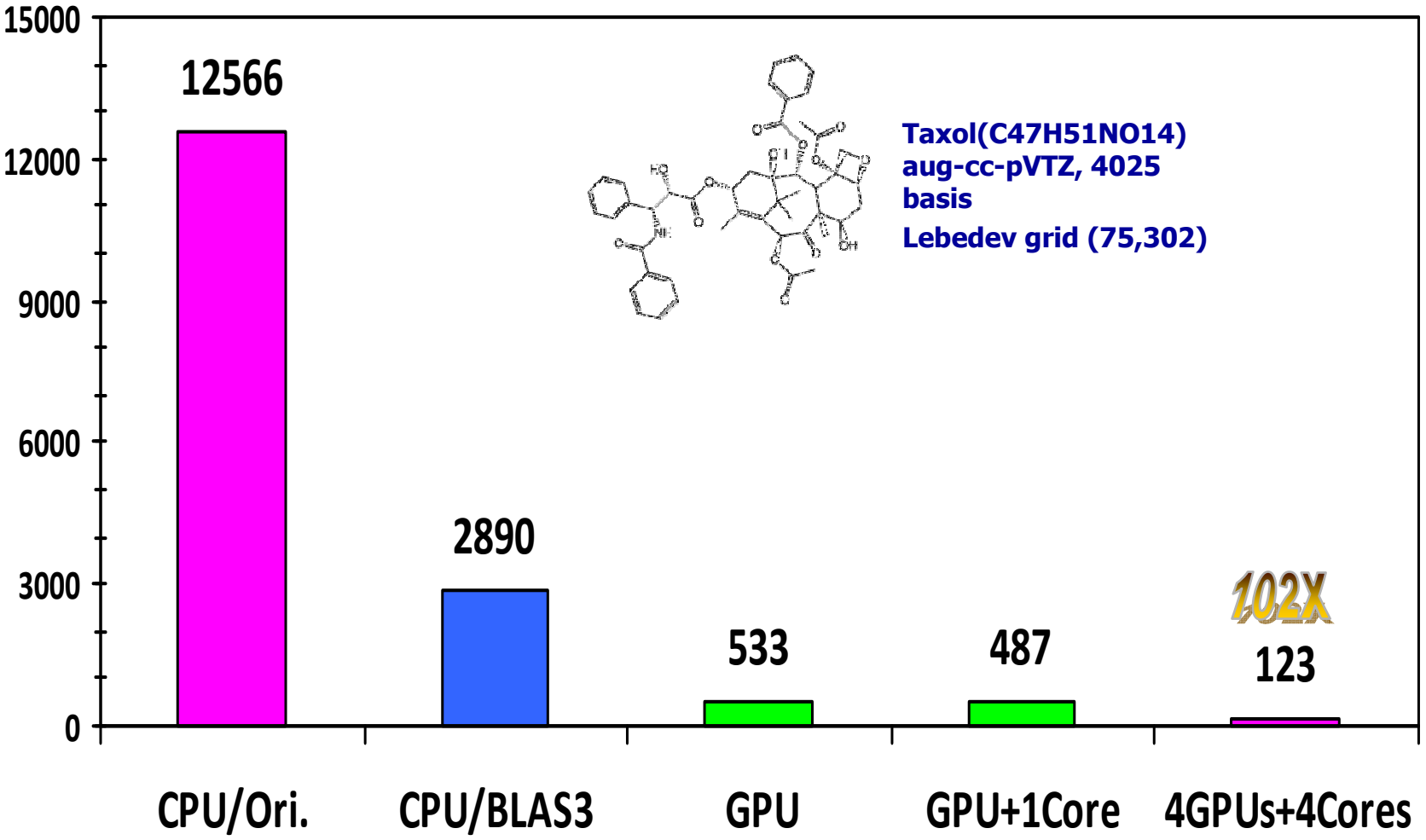
Taxol ($C_{47}H_{51}NO_{14}$) / aug-cc-pVTZ

4025 basis functions

Lebedev grid (75,302)

Benchmark calculation on QuadCore Xeon with NVidia S1070 card attached

Performance Analysis
Heterogeneous Computing

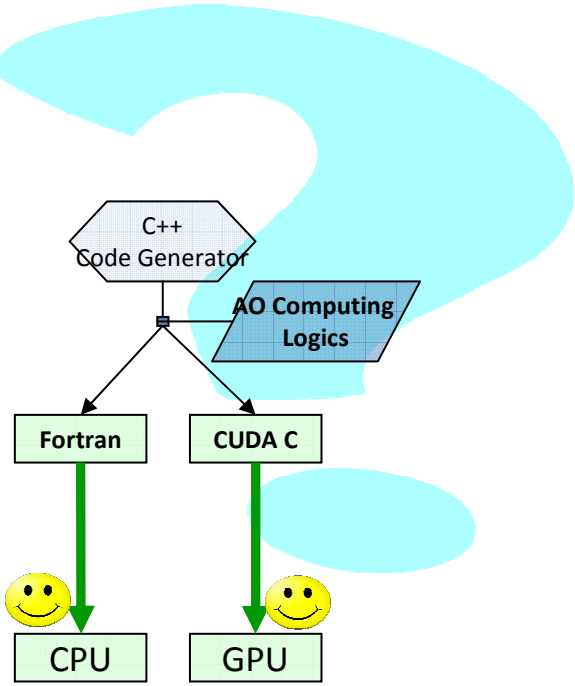


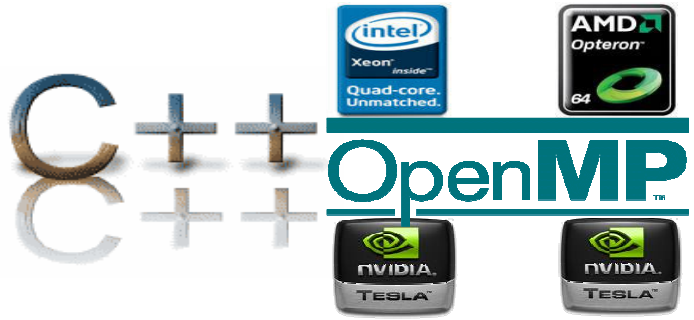
Calculation run on Dual socket Xeon E5462 2.8GHz with S1070 GPU

Summary & Questions

BLAS3 based algorithm

GPU friendly code in AO computing





"TOOM" hybrid computing solution