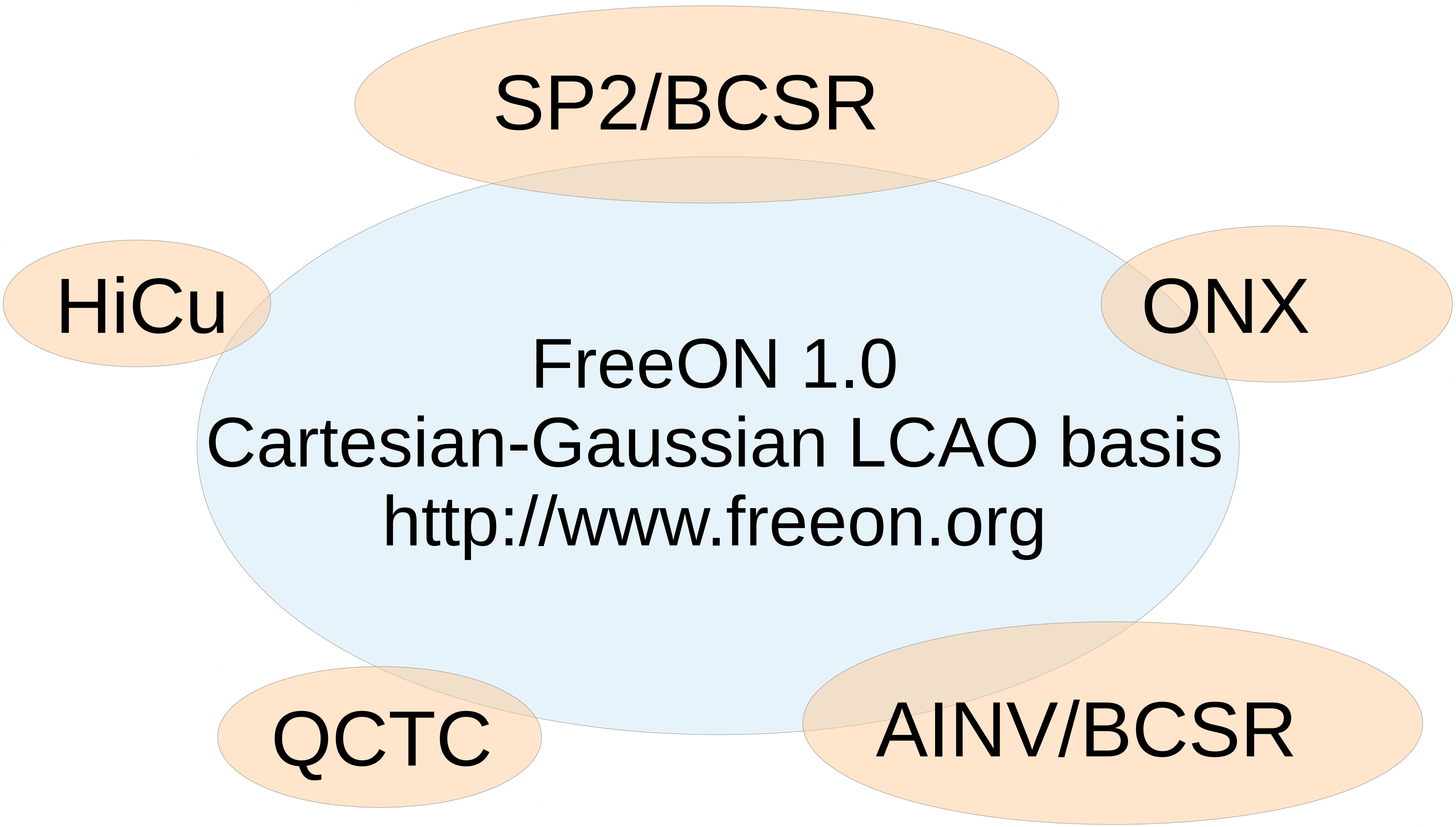
Solvers for O(N) Electronic Structure in the Strong Scaling Limit with Charm++

Nicolas Bock
Matt Challacombe
Theoretical Division
Los Alamos National Laboratory

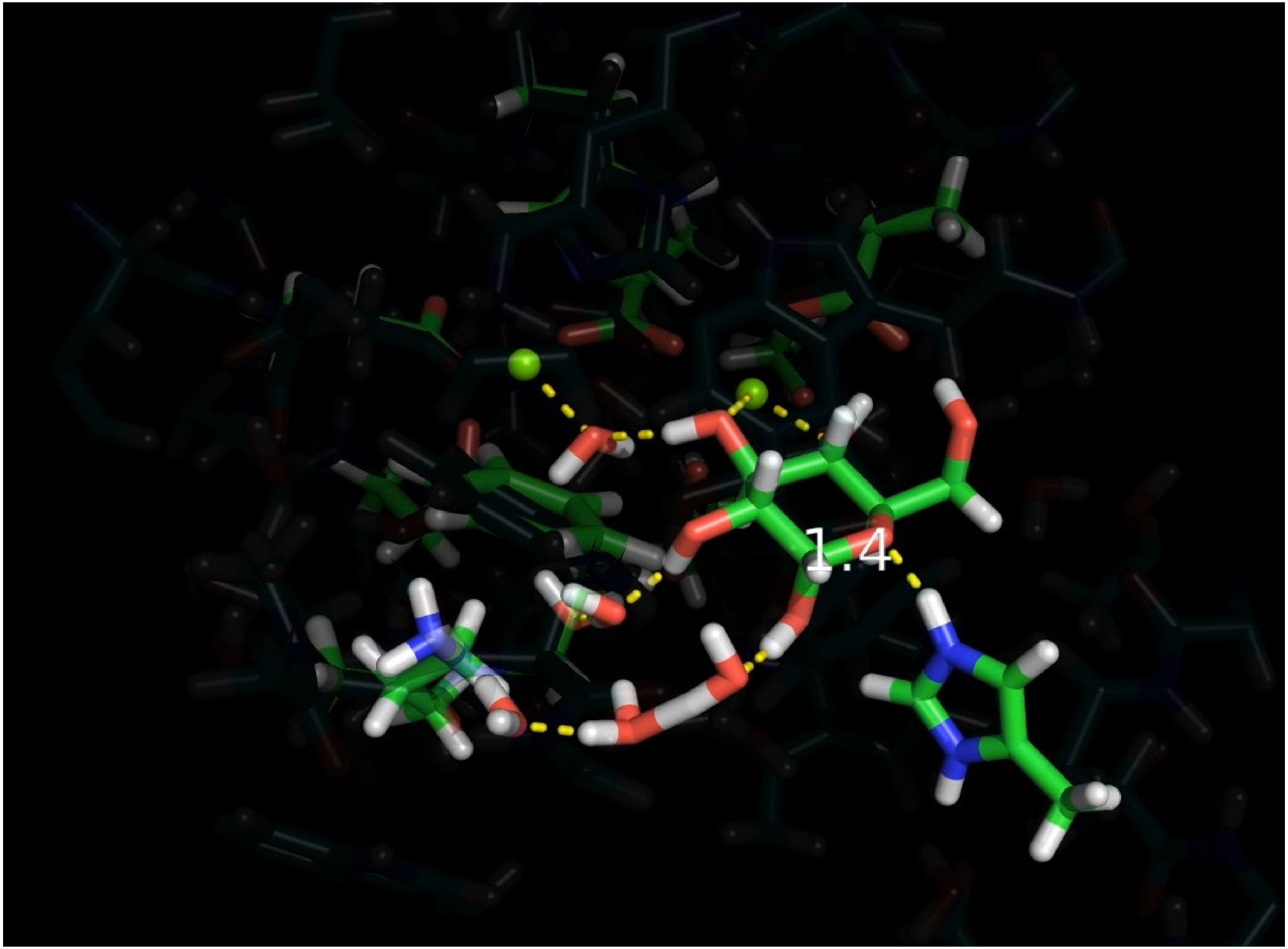
Laxmikant V. Kalé
Parallel Programming Laboratory
University of Illinois at Urbana-Champaign

12th Annual Workshop on Charm++ and its Applications
29th April 2014

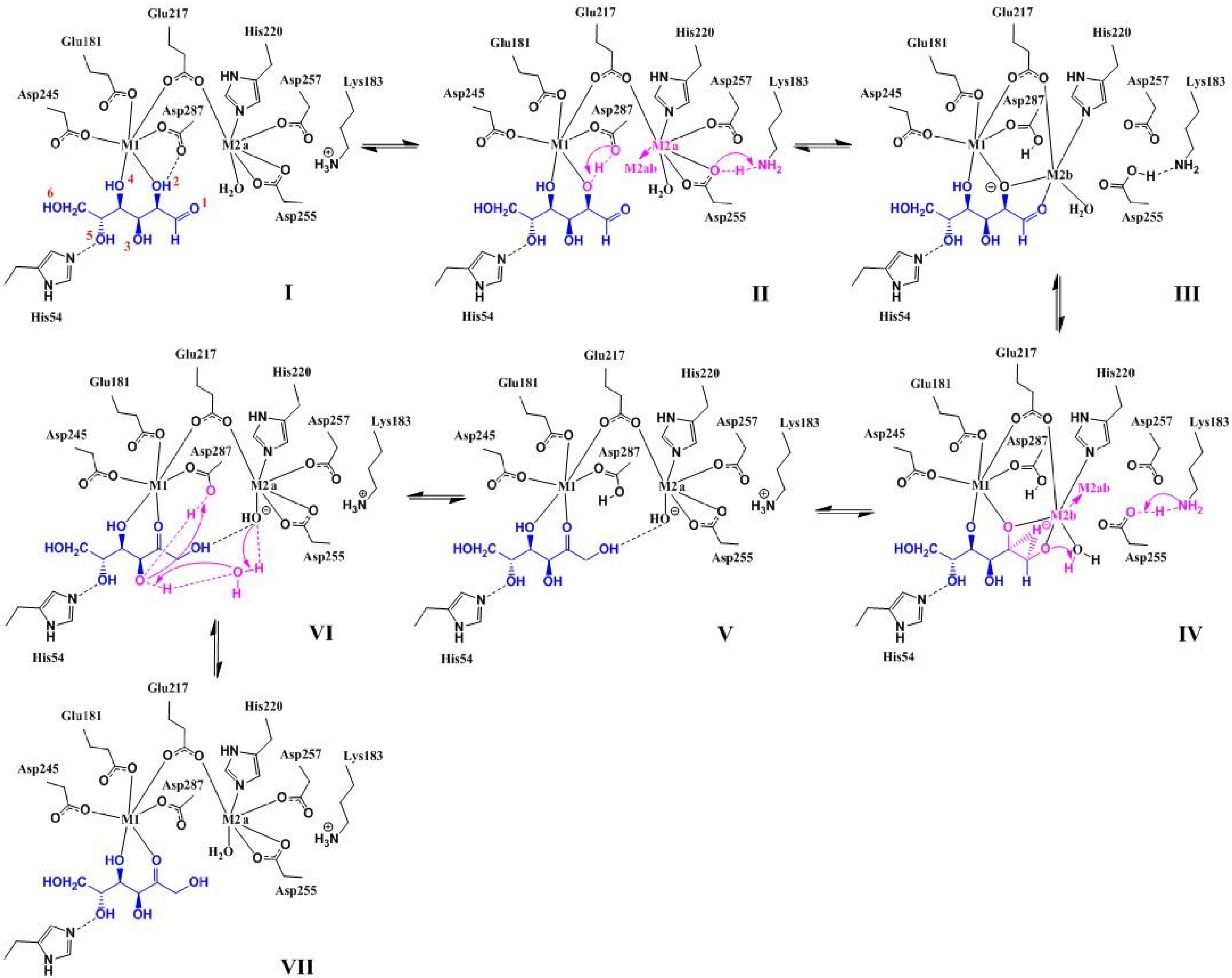
FreeON - O(N) Electronic Structure



Xylose Isomerase in FreeON



Xylose Isomerase in FreeON



FreeON – O(N) Electronic Structure

In a Nutshell, FreeON...

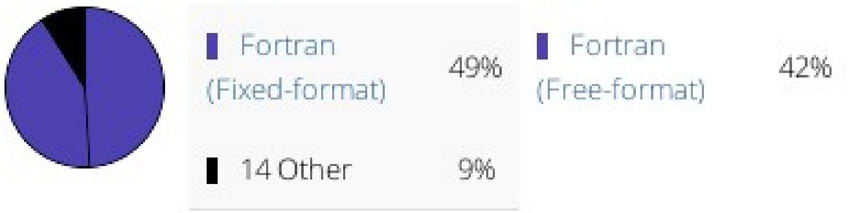
... has had 5,029 commits made by 12 contributors
representing 992,564 lines of code

... is mostly written in Fortran (Fixed-format)
with a low number of source code comments

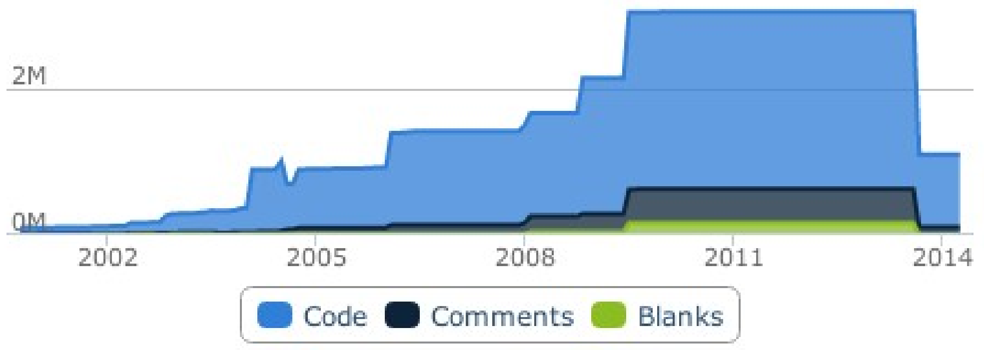
... has a well established, mature codebase
maintained by by one developer
with increasing Y-O-Y commits

... took an estimated 281 years of effort (COCOMO model)
starting with its first commit in October, 2000
ending with its most recent commit about 1 month ago

Languages

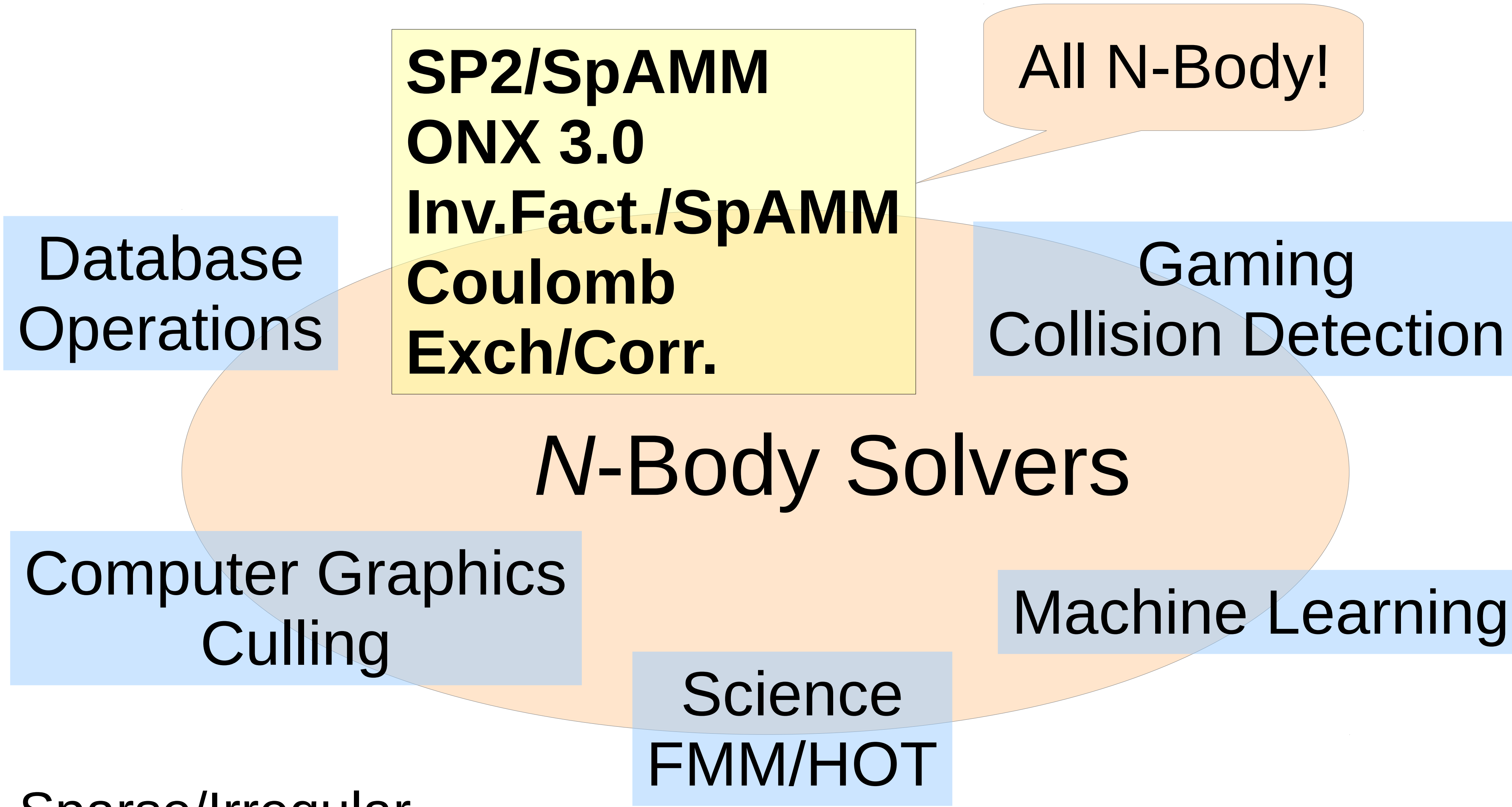


Lines of Code



Ohloh code analysis: http://www.ohloh.net/p/freeon

Unified Solver Framework



Sparse/Irregular

- Linear scaling complexity, O(N)
- **With scalable parallelism, increasing core count yields proportional capability gains**

N-Body for Electronic Structure

- Generalize **range query** → **metric query + ...**
- All 5 solvers as N-Body
- Unified programming model
- Unified data structures
- **Task-parallel** decomposition
- Clean separation between **solver** and **runtime**
- **Concise** solver code

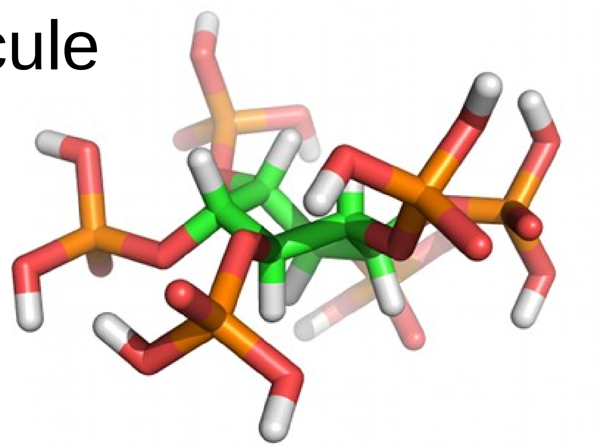
SpAMM

Sparse Approximate Matrix Multiply (SpAMM) for matrices with decay

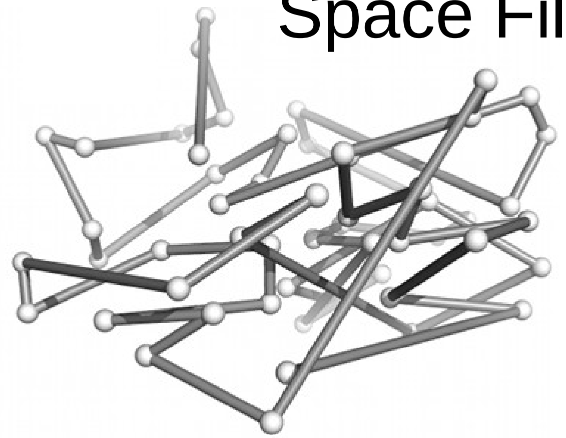
- Occlusion based on metric query
- Linear scaling electronic structure (FreeON)
- General alternative to incomplete matrix algebra (“sparsification”)
- N-Body learning
- On the fly dropping of product contributions can lead to better accuracy than GEMM, and O(N) execution time for matrices with decay.

SpAMM

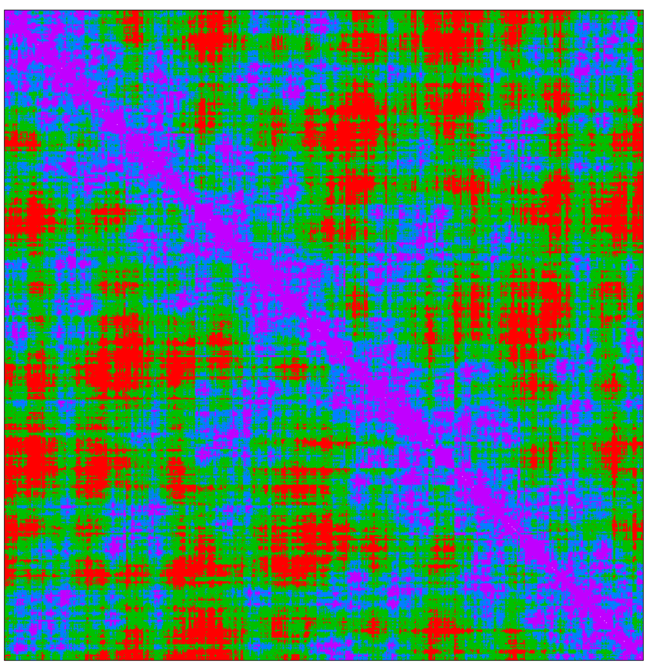
Molecule



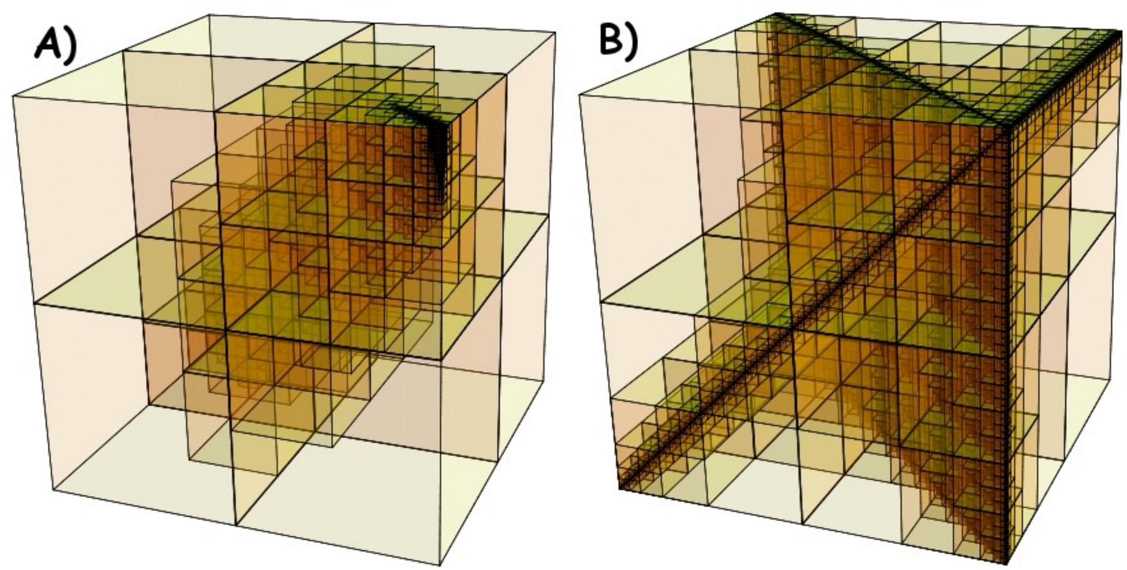
Space Filling Curve



Matrix/Ouadtree



Convolution/Octree

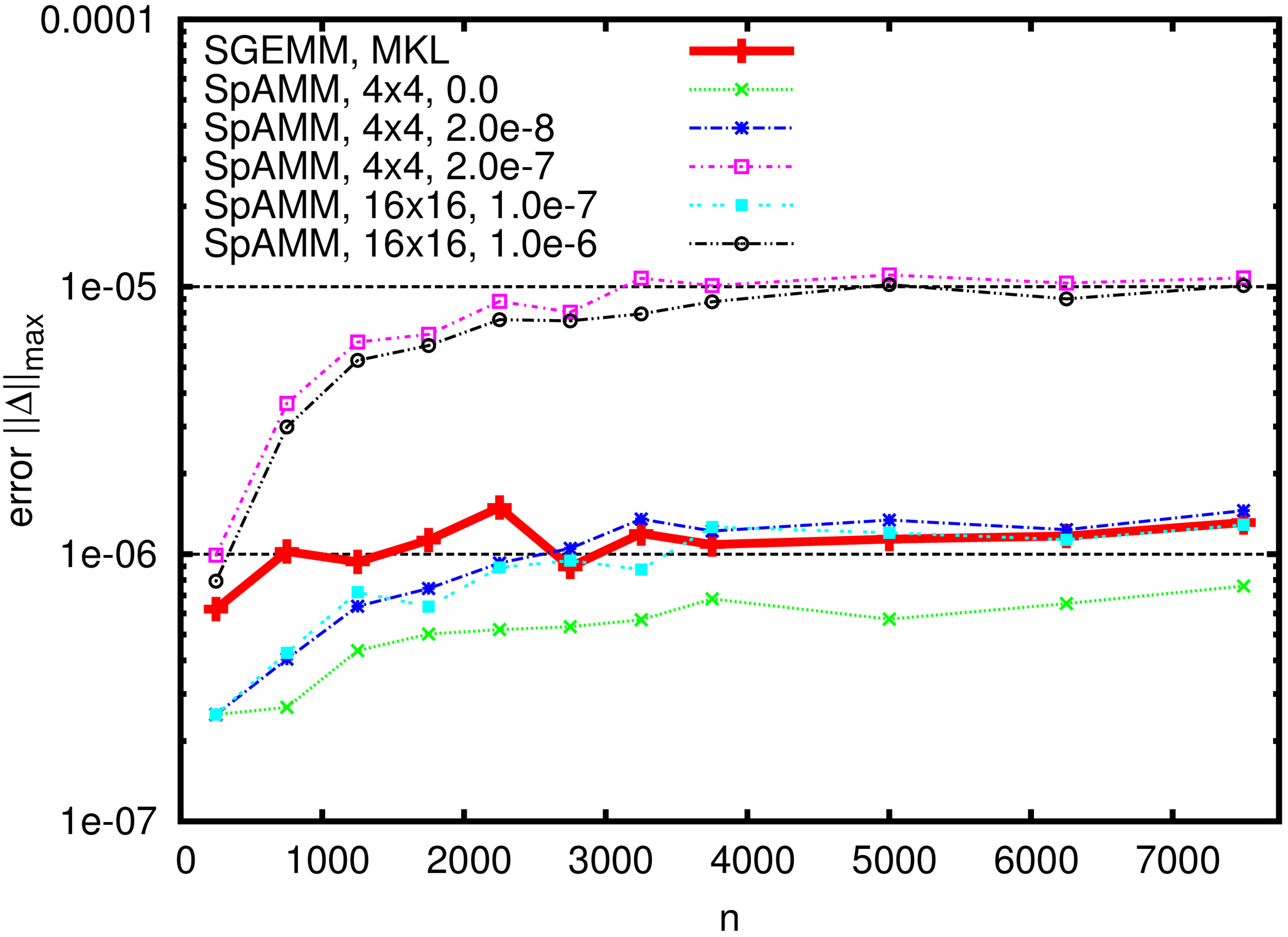


A) Exponential decay, B) Algebraic decay

SpAMM – Task-Parallel

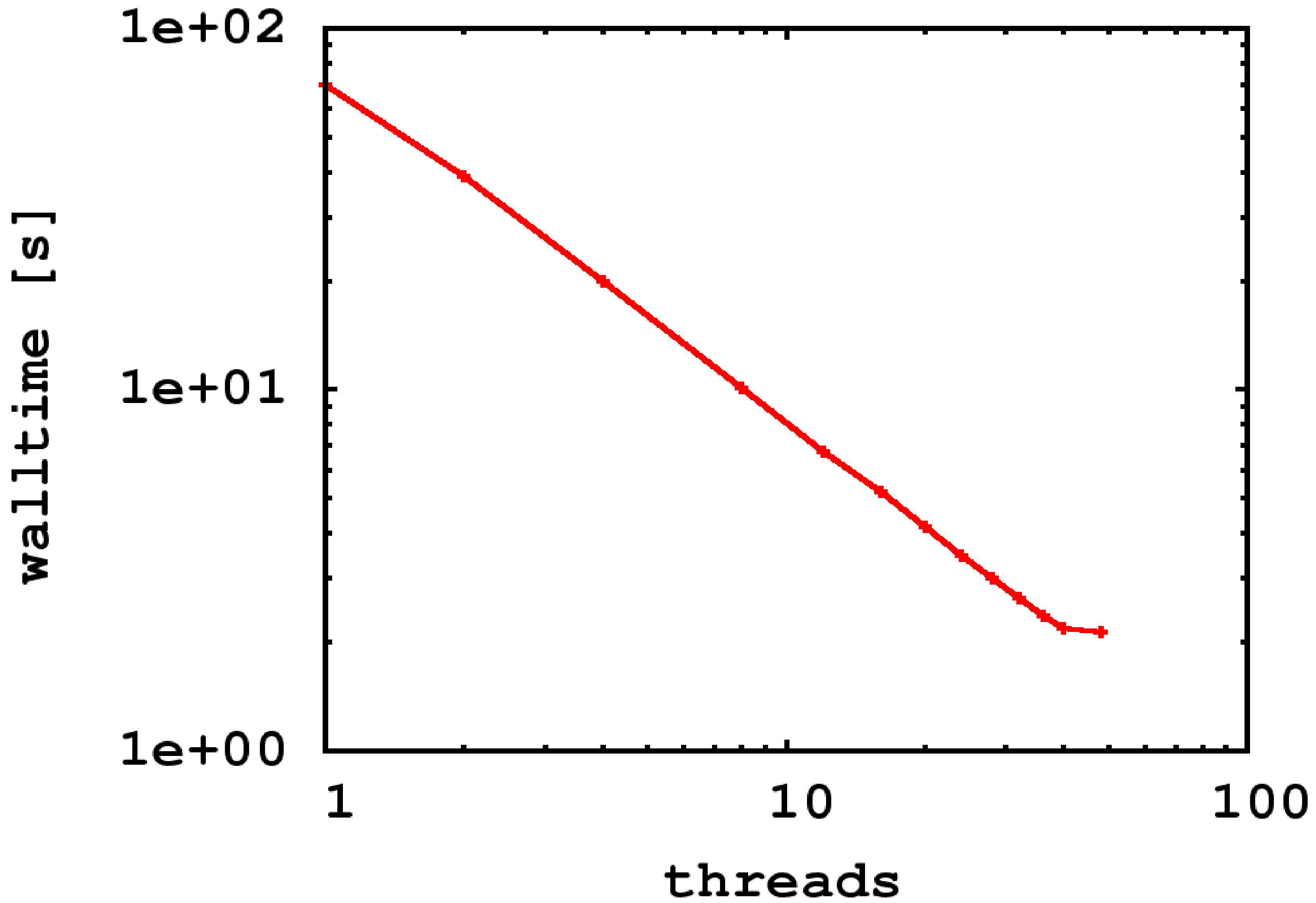
- Linked list on top tiers → recursive execution
- Task parallelism with OpenMP at top
- Linear quadtree on bottom tiers
  - Hashtables/Linear index
  - Kernel for efficient submatrix multiplication
- High performance serial execution at bottom
- Dropping is applied all the way down to 4x4
- Non-contiguous, dynamic allocation
- Or, contiguous allocation and position independent data structure

SpAMM – Error

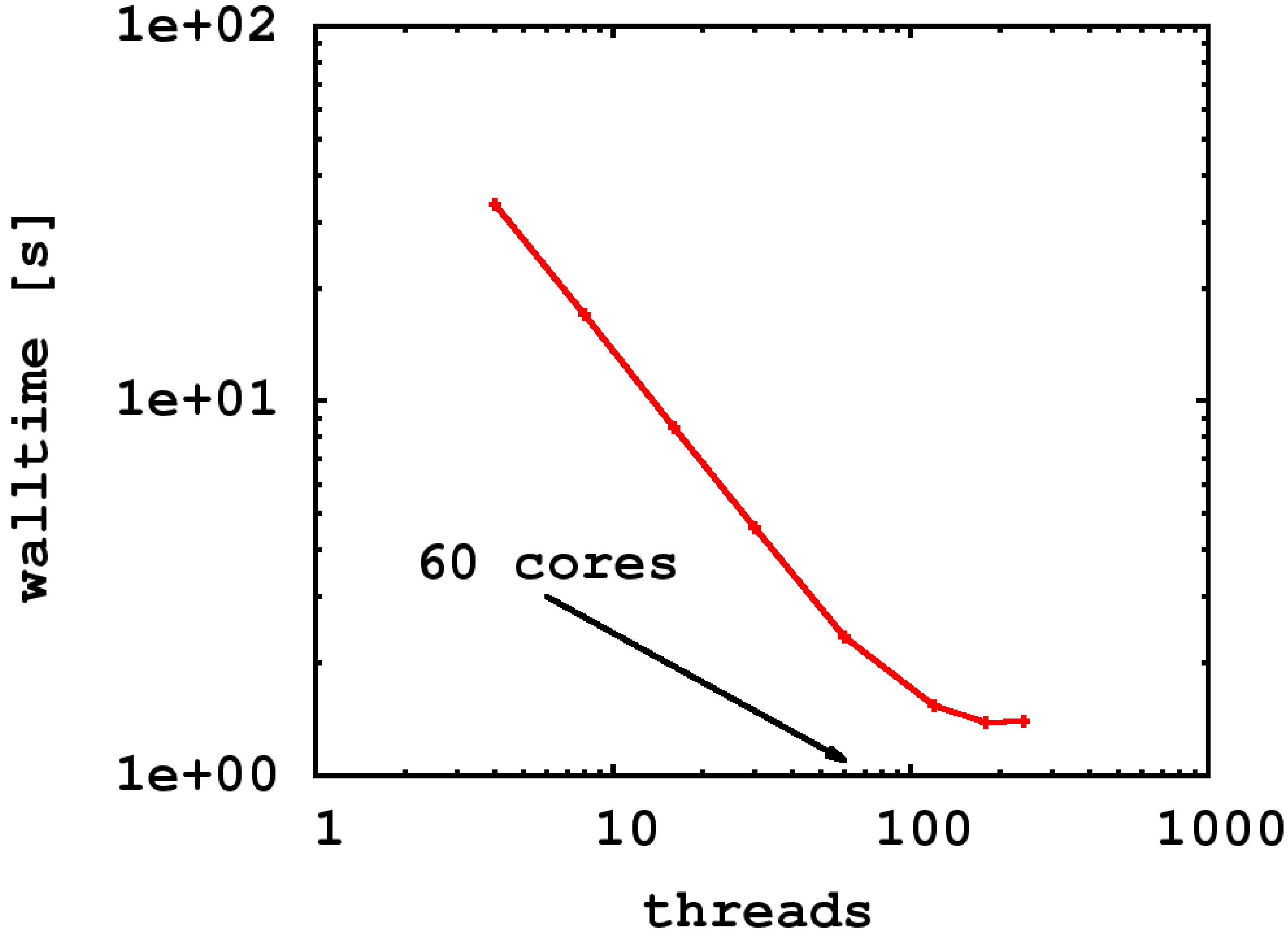


0.0001
1e-05
1e-06
1e-07
error $||\Delta||_{max}$
0
1000
2000
3000
4000
5000
6000
7000
n
SGEMM, MKL
SpAMM, 4x4, 0.0
SpAMM, 4x4, 2.0e-8
SpAMM, 4x4, 2.0e-7
SpAMM, 16x16, 1.0e-7
SpAMM, 16x16, 1.0e-6

SpAMM – Parallel Efficiency on Magny Cours



SpAMM – Parallel Efficiency on Xeon Phi



SpAMM - OpenMP

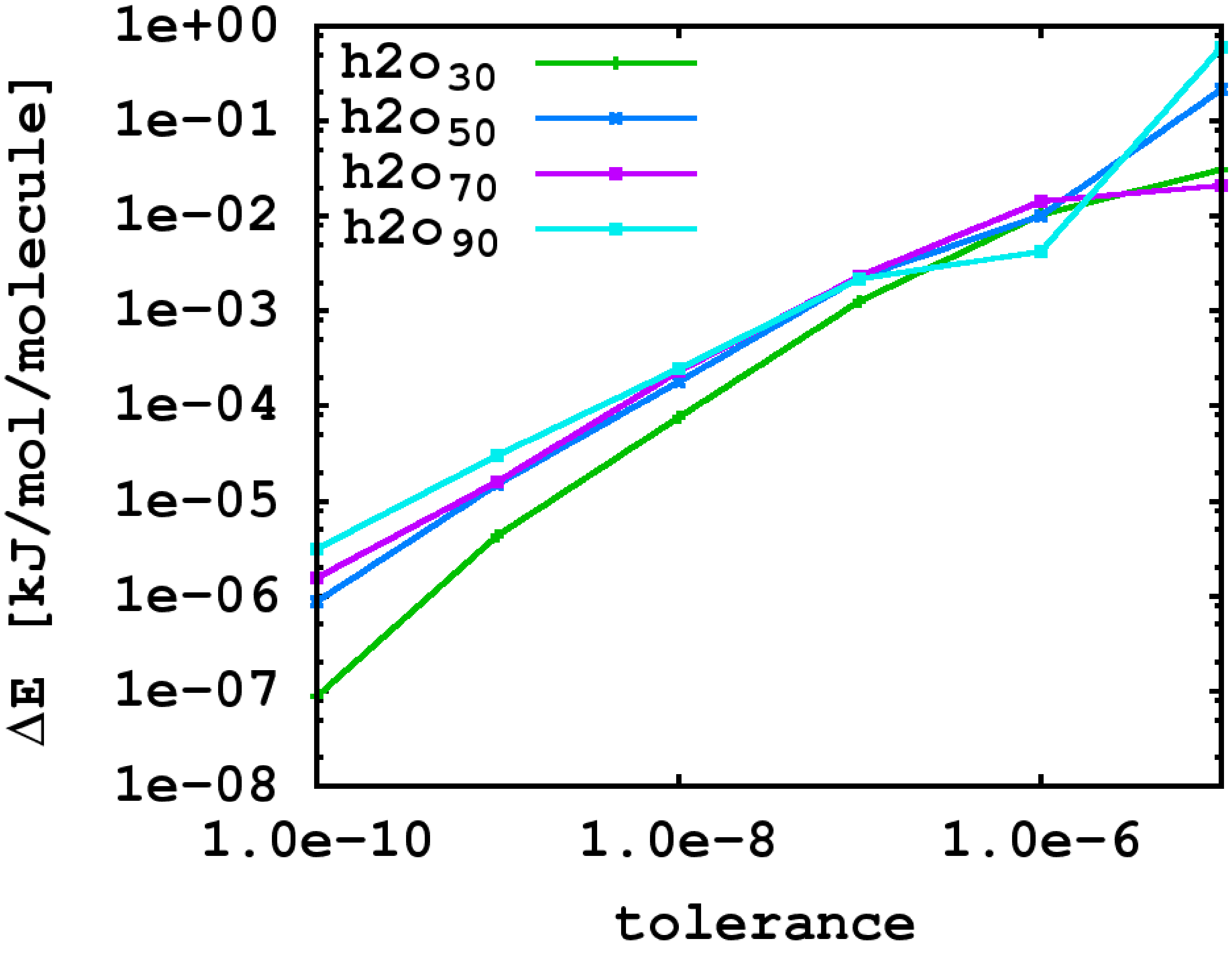
```
function MULTIPLY(τ, tier, A, B, C)
    if tier < depth then
        for all {i, j, k | C_ij ← A_ik B_kj} do
            if ‖A_ik‖ ‖B_kj‖ > τ then                          ▷ Eq. 1
                Create untied OpenMP task
                MULTIPLY(τ, tier+1, A_ik, B_kj, C_ij)
            end if
        end for
        OpenMP taskwait
    else
        Acquire OpenMP lock on C
        C ← C + A × B                    ▷ Dense product, e.g. BLAS
        Release OpenMP lock on C
    end if
end function
```

SpAMM - Charm++

- Quadtree linked list → 2D chare array per tier
- Recursive multiply → 3D chare array per tier
- GreedyComm LB after each multiply

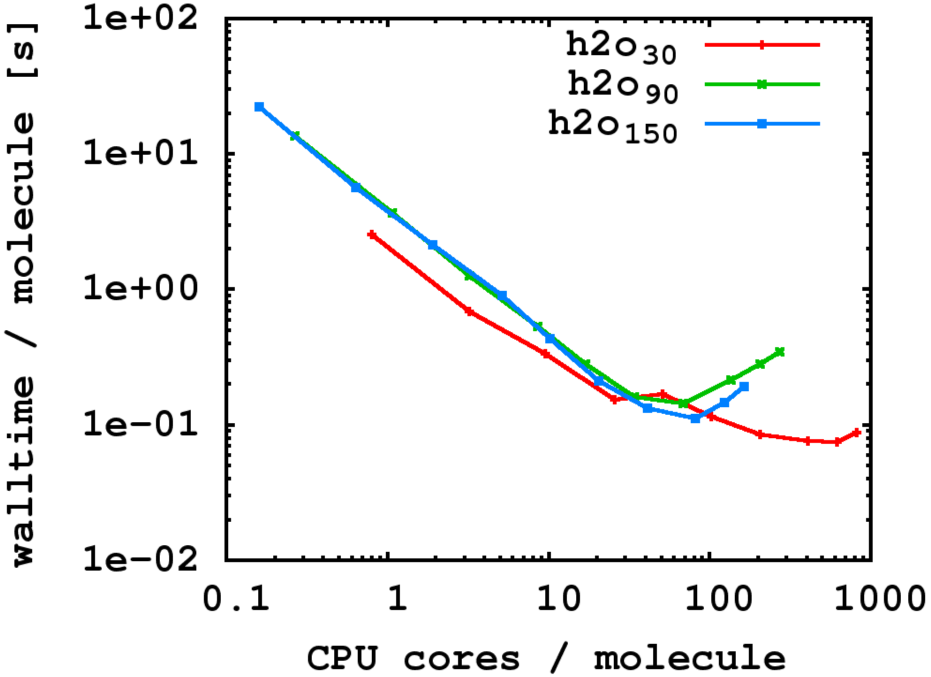
function MULTIPLY(τ, A, B, C)
    for $t \geq 0 \wedge t < d$ **do**
        convolution[t].prune()
    end for
    convolution[d].multiply()
    convolution[d].store()
end function

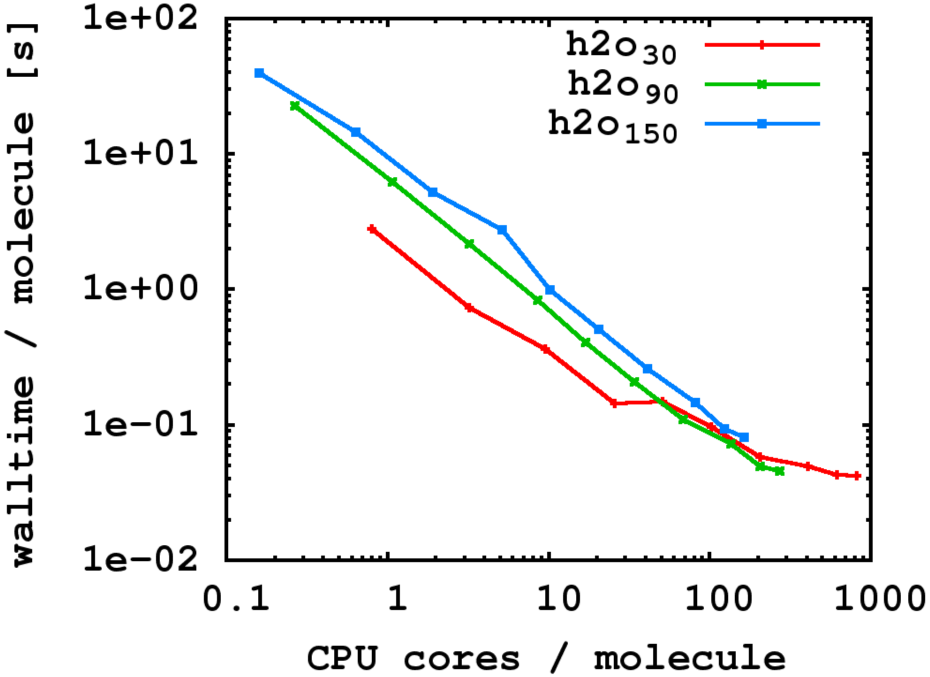
SP2/SpAMM - Charm++



SP2/SpAMM - Charm++

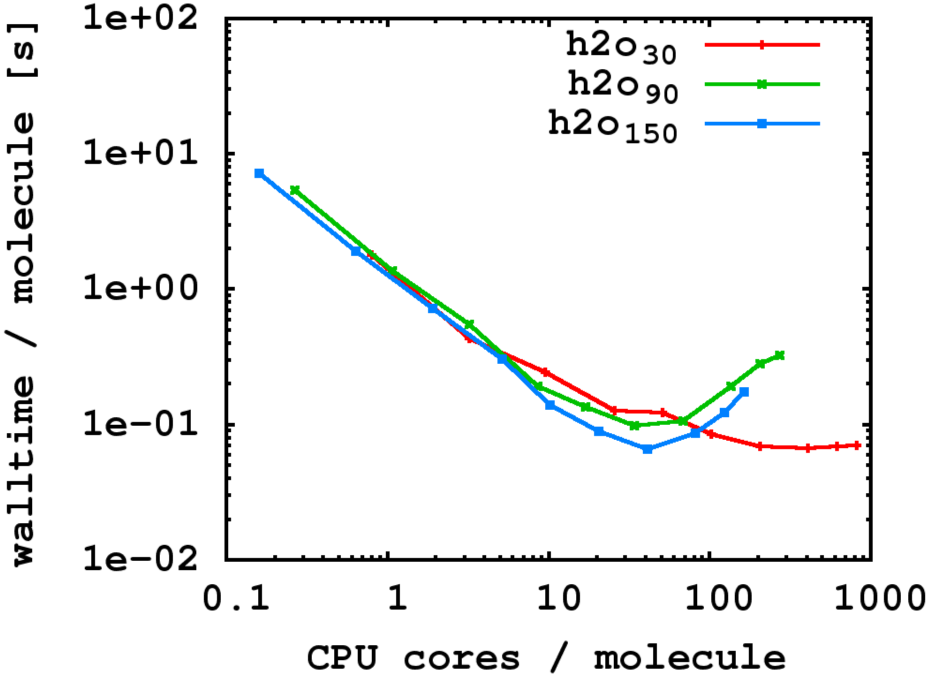
B3LYP/6-31G**: tolerance = 10^{-10}

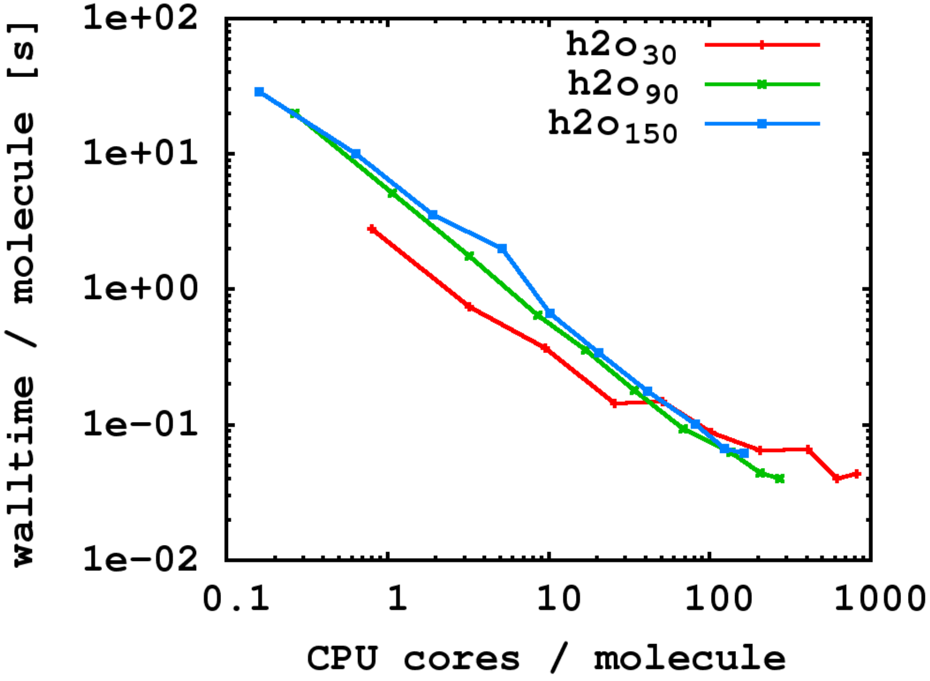




SP2/SpAMM - Charm++

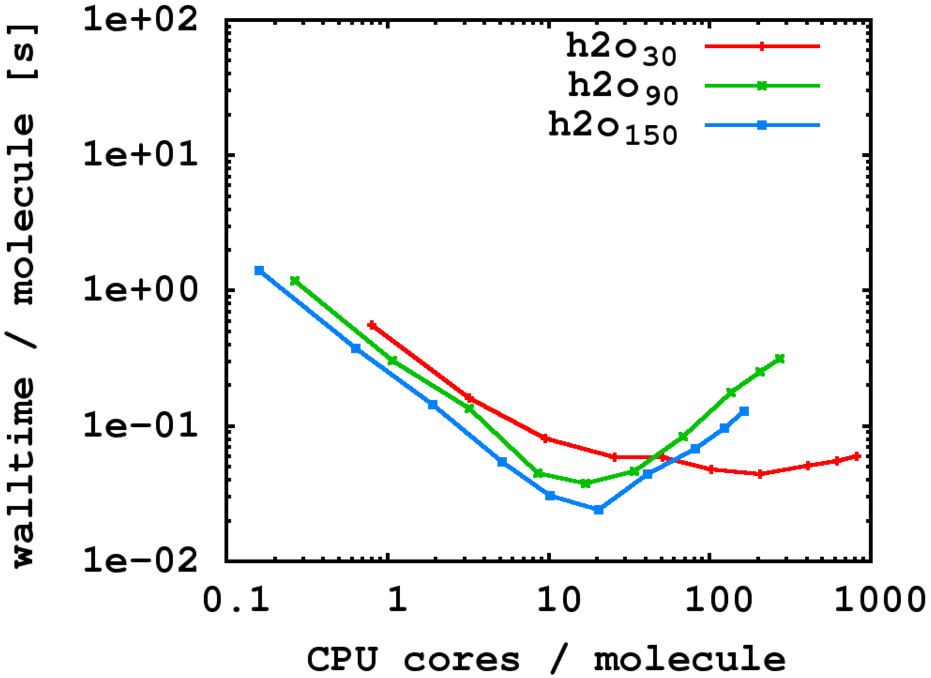
B3LYP/6-31G**: tolerance = 10^{-8}

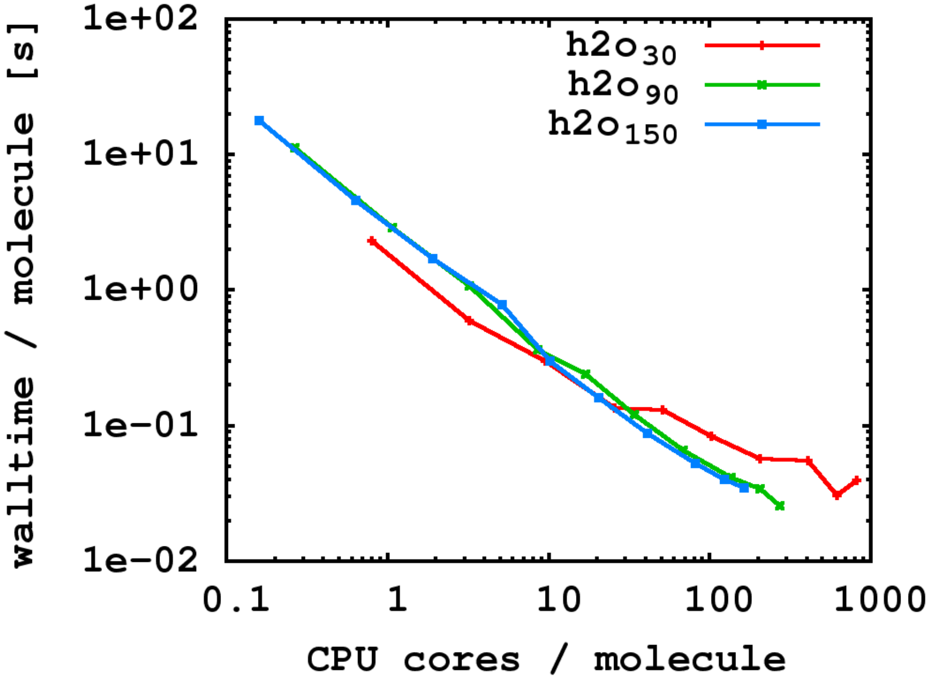




SP2/SpAMM - Charm++

B3LYP/6-31G**: tolerance = 10^{-6}





Conclusions

- Novel unified solver approach based on N-Body
- First time demonstration of O(N) electronic structure solver in strong scaling limit
  - Parallel scaling to almost 1000 (!) cores / atom
  - The competition: 1 molecule or atom / core
- Closer alignment of programming models?
  - Singleton chares for N-Body?
  - Express same recursive task-parallel approach?
- **Holistic load balancing across solver collective?**