UNIVERSITY OF Southampton

OpenMP in ONETEP

Karl Wilkinson and Nicholas Hine

Motivation

The number of cores that could be used in a ONETEP calculation was subject to restrictions:

Issue 1: Number of MPI processes may sometimes be less than the number of cores available on a node.

- Memory usage per core means that it is not always possible to use every core..

Issue 2: Number of MPI processes must always be less than the number of atoms

- Distribution of workload.

MPI and OpenMP

Message Passing Interface (MPI):

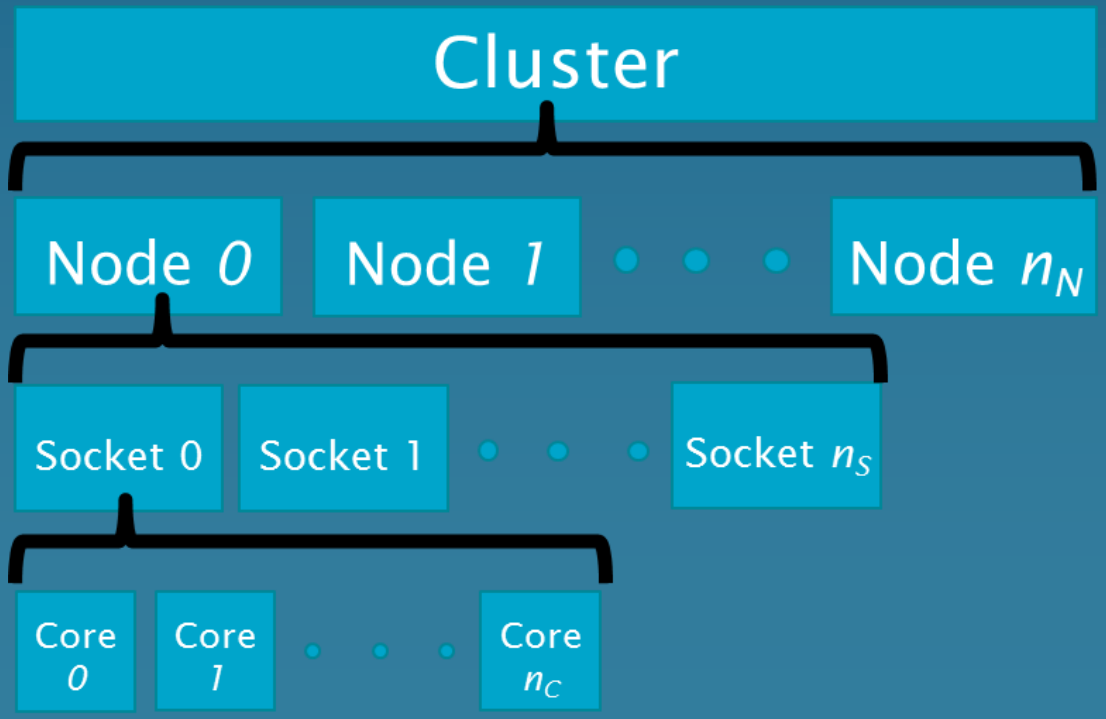
Each MPI "process" has its own memory space – communication of data between processes is controlled explicitly.

OpenMP:

Shared memory model, each thread can access the same memory.

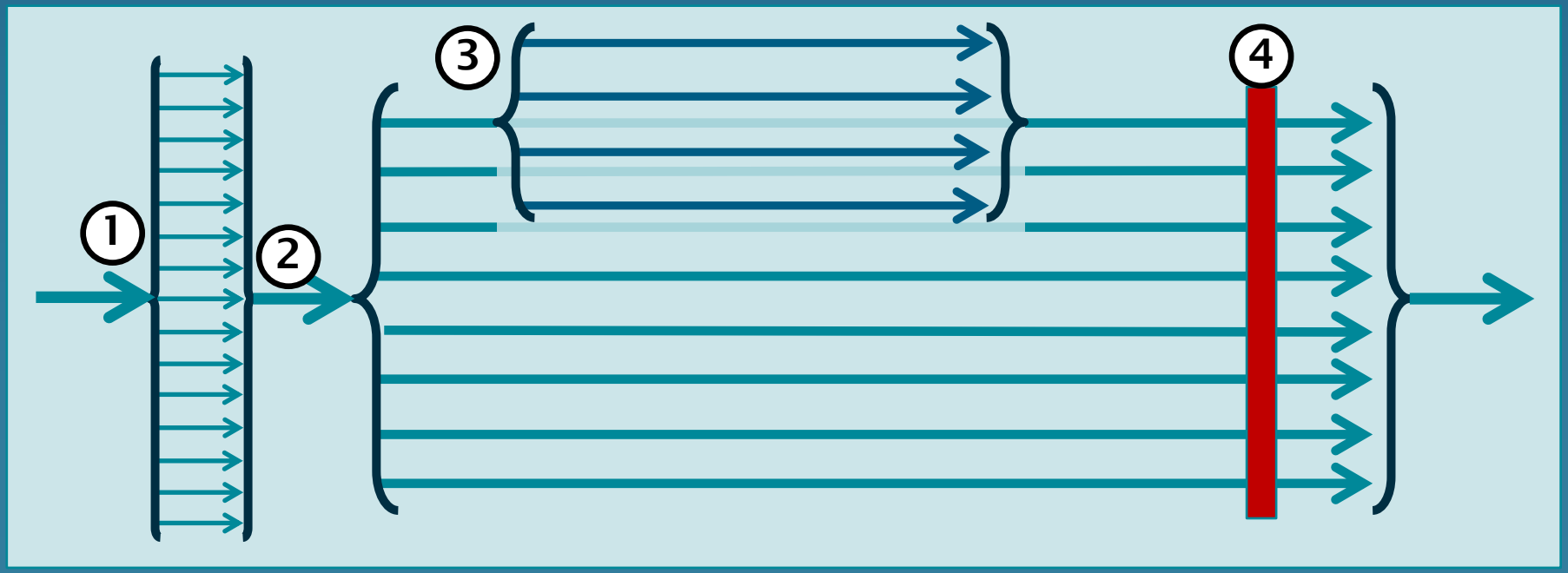
Hybrid MPI/OpenMP:

Reflects hardware – MPI to distribute work across nodes and OpenMP to distribute work between cores on a node.



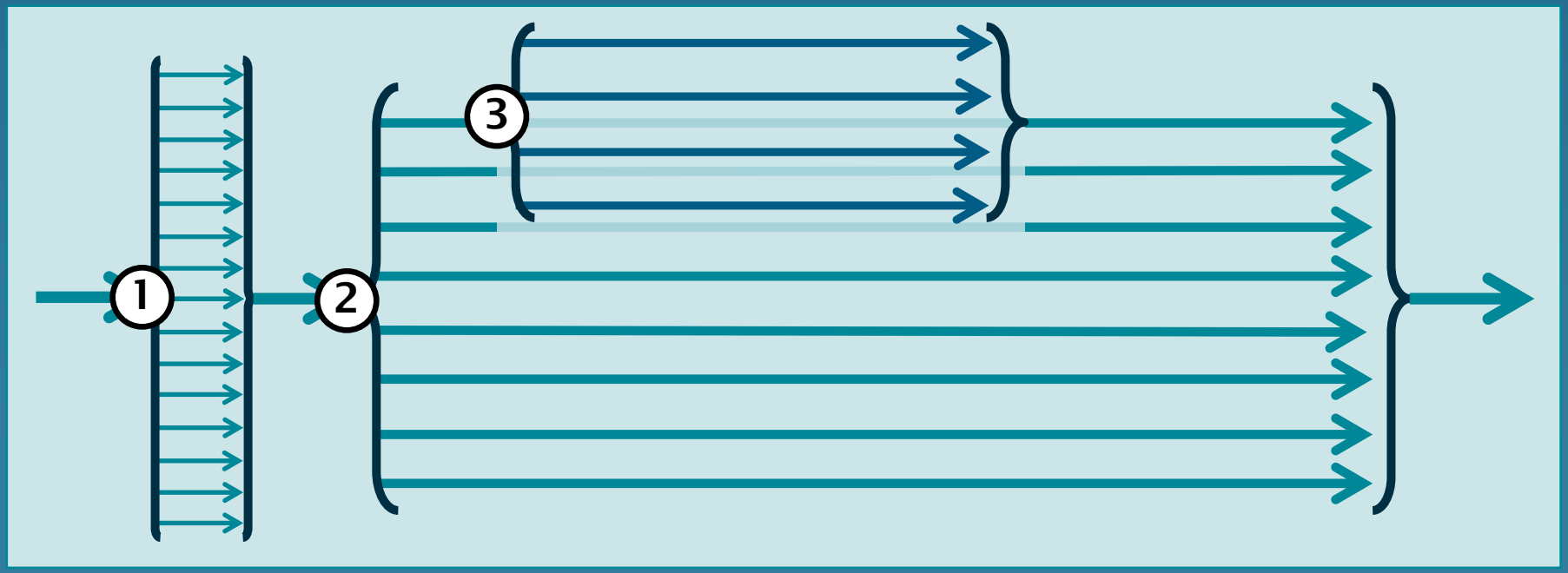
OpenMP Paradigms in ONETEP

1. Thread Creation !$OMP PARALLEL DO/WORKSHARE
2. Thread Merging !$OMP END PARALLEL
3. Thread Branching
4. Thread Blocking !$OMP CRITICAL

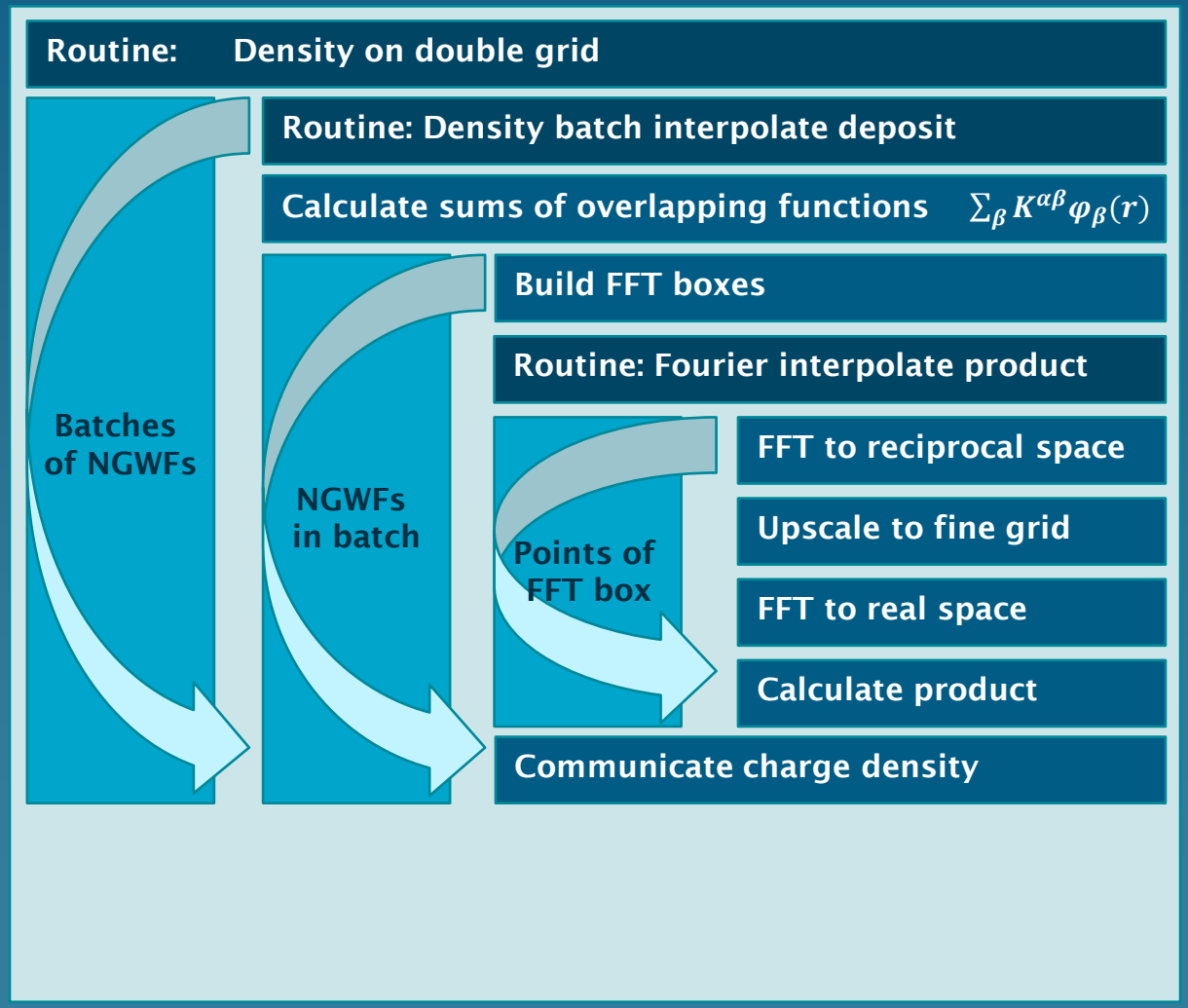


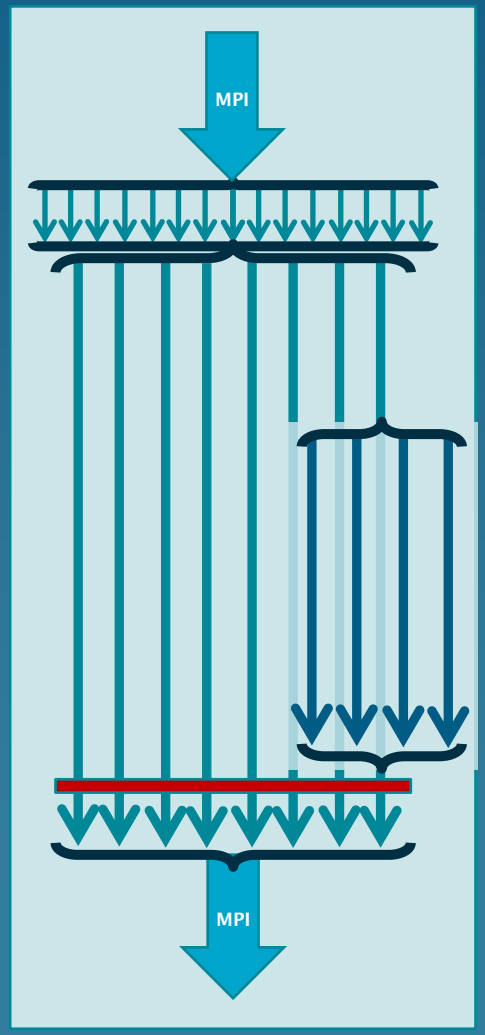
OpenMP Targets in ONETEP

1. General parallel operations – Sparse linear algebra/Simulation cell FFTs.
2. Operations on batches of FFT boxes – Multiple threads, each operates on one FFT box of a batch.
3. *Operations on FFT boxes – Multiple threads operate on an FFT box.*



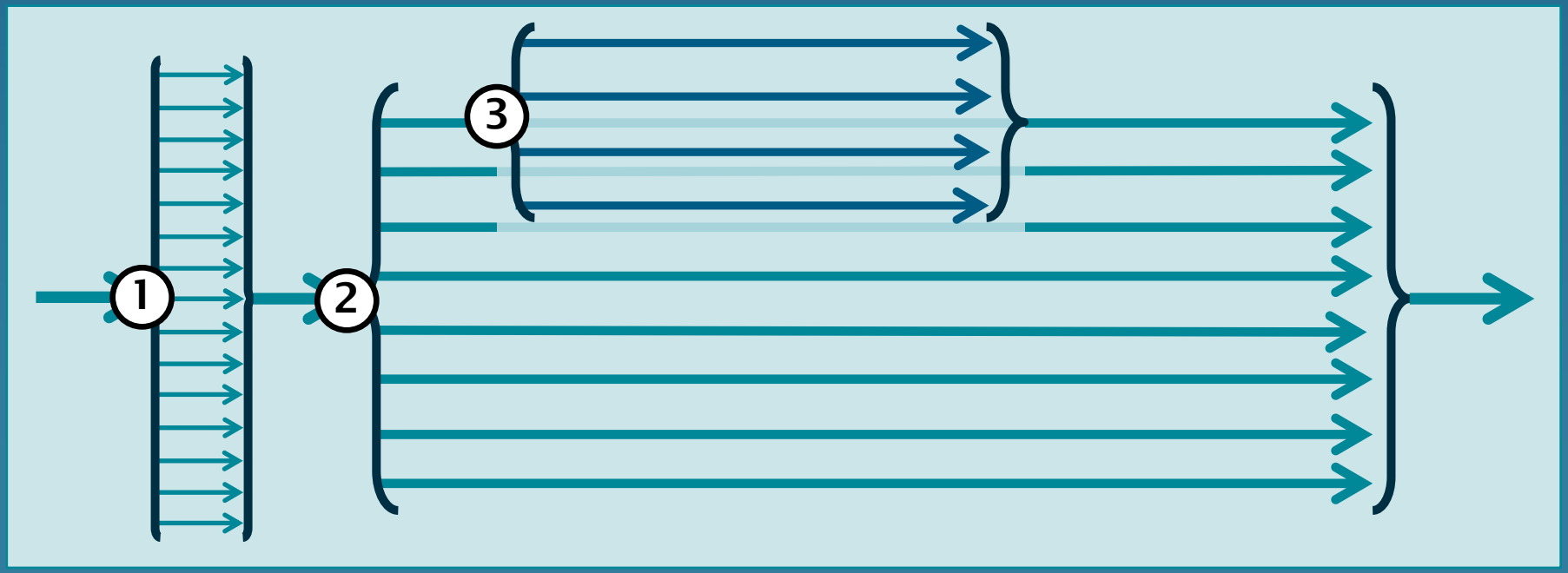
Charge Density Example





OpenMP Options in ONETEP

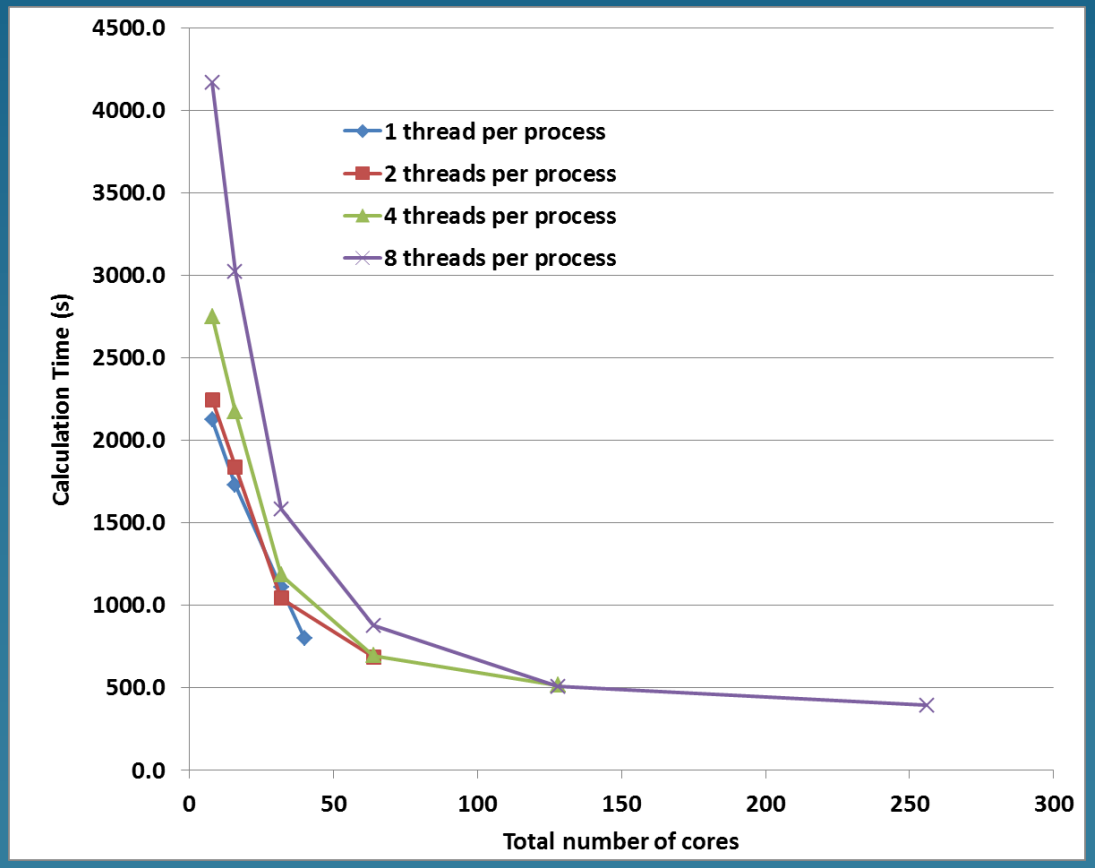
1. General parallel operations: threadsmax and threadspercellfft
2. Operations on FFT box batches: threadsnumfftboxes/fftboxbatchsize
3. *Operations on FFT boxes* *threadsperfftbox*



Performance: Small systems

64 atom DNA base pair – 8 to 0.25 atoms per core

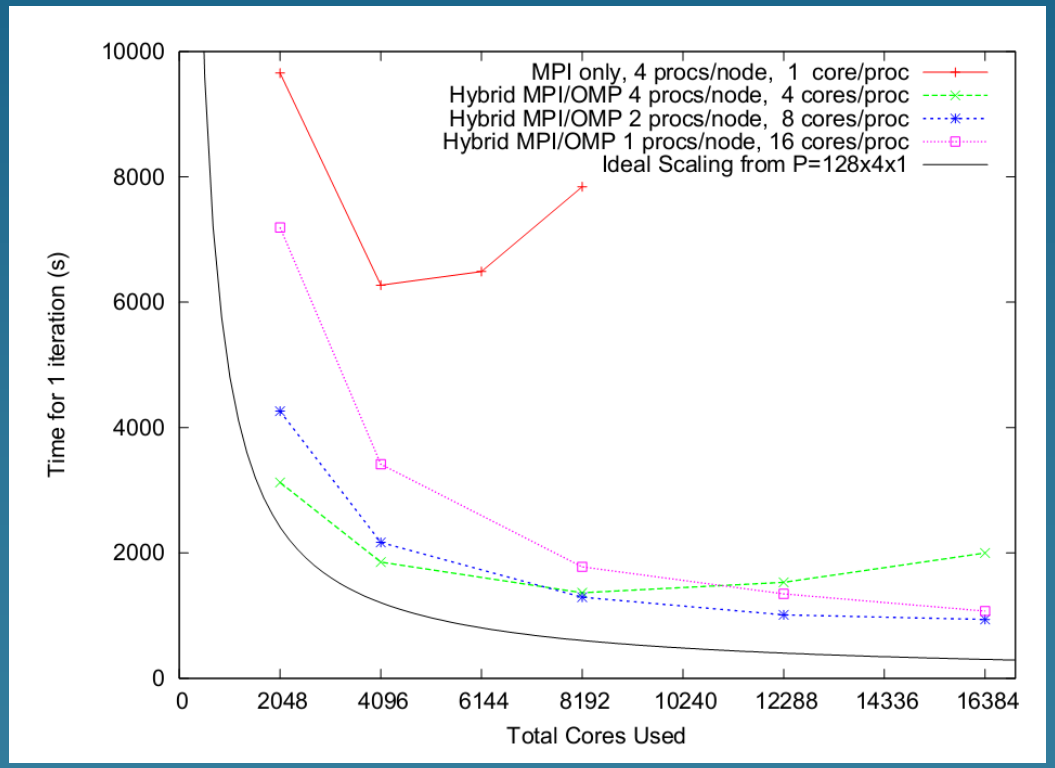
- Limited to 40 cores if using pure MPI (799s)
- 393s with 256 cores (32 MPI processes, 8 threads per process)
- Lower threads per core better for lower number of cores (communication)
- Can reach 512 cores (0.125 atoms per core) but no performance gain over 256 cores



Performance: Large Systems

13,969 atom beta amyloid fibril – 6.8 to 0.9 atoms per core

- Pure MPI can only be executed on up to 8192 cores – antiscales above 4096 cores
- Lower atoms per core = lower threads per process
- Other tests have indicated that OpenMP may cause antiscaling at high atoms per core



Summary

Eases restrictions on number of atoms per core

Significantly increases the scale of ONETEP calculations

Reduces time to solution for a given problem but requires more cores --
- May be more efficient to run jobs in parallel, each on fewer cores.

- Still work to do:
  - Antiscaling at high numbers of atom per thread (serial communications?)
  - Performance analysis of different areas of the code (optimum number of threads per process)
  - Performance analysis on wider range of machines
  - "Data parallel" code