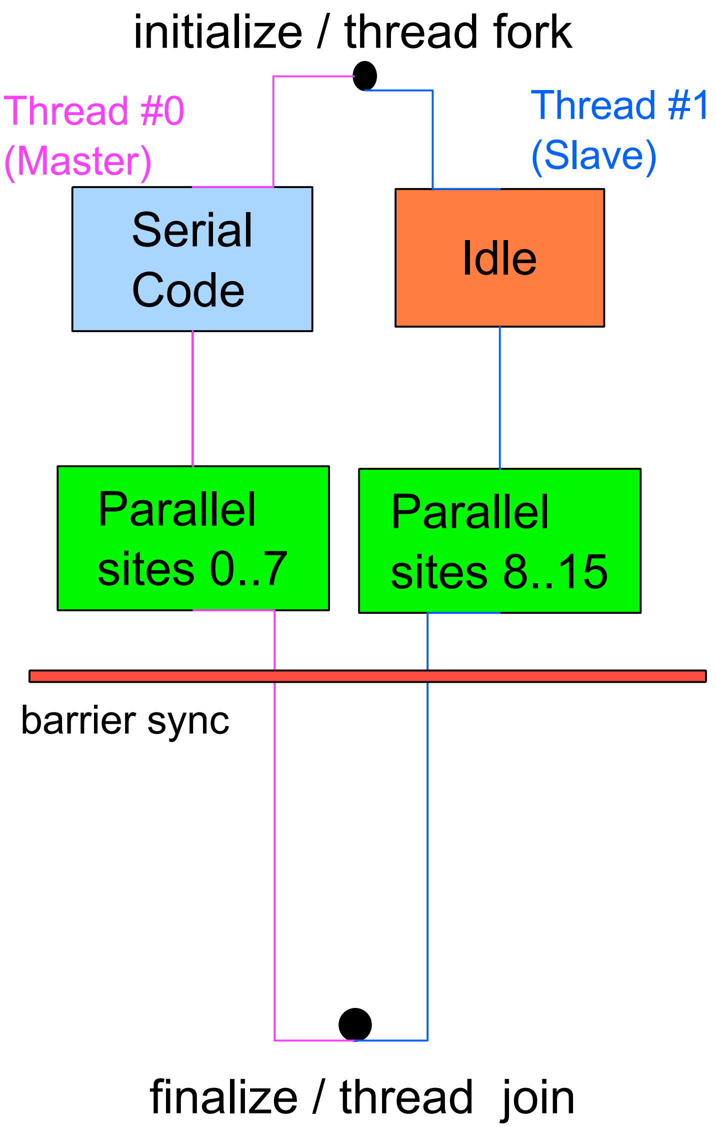
QMT – QCD Multi Threading

- First steps
  - Step 1: General Evaluation
    - OpenMP vs. Explicit Thread library (Chen)
      - Explicit thread library can do better than OpenMP
      - OpenMP performance is compiler dependent
        - » Intel compiler does much better than GCC
  - Step 2: Simple Threading API: QMT
    - based on older smp_lib (A. Pochinsky)
    - use pthreads and investigate barrier synchronisation algorithms
  - Step 3: Evaluate usefulness of QMT in SSE-Dslash
  - Step 4: Tweak QMT... Go back to Step 3 until done.

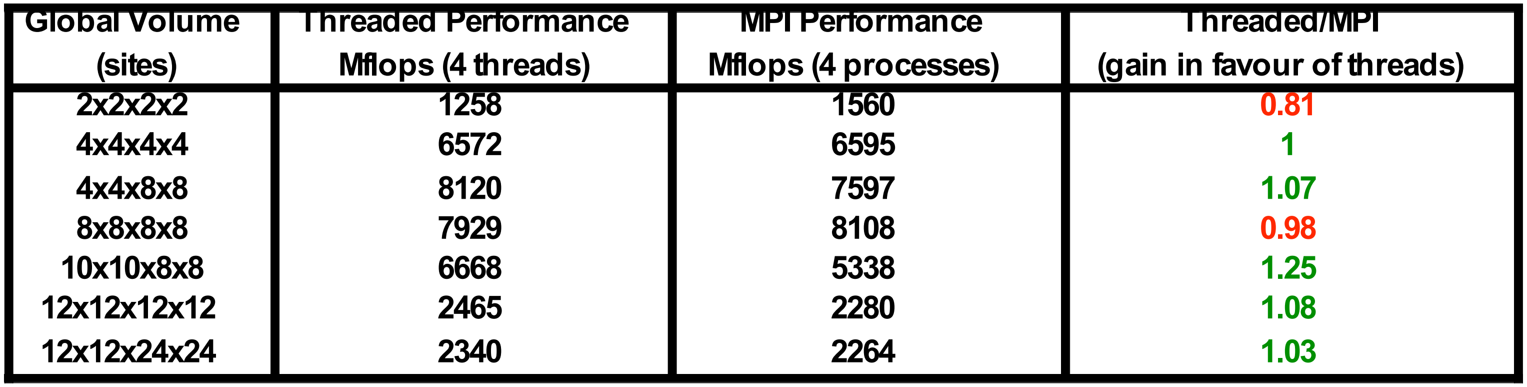
QMT – Basic Threading Model

- 1 Master Thread & several slave threads spawned when calling qmt_init()
- Node- Serial part of code runs in master thread – while slaves sit idle.
- Node-Parallel parts of code run in master and slave threads
  - Data parallel: All threads execute same function on different data.
  - Data blocks described in terms of first & last site of block.
- Slave threads destroyed by calling qmt_finalize();



Dslash

- Implemented (re-enabled) threading in SSE Dslash
- Tested on Dual Socket, Dual Core (4 cores in total) Opteron, 64 bit linux.
- Compare 4 threads in 1 MPI process vs 4 MPI processes communicating through memory.



Global Volume (sites)	Threaded Performance Mflops (4 threads)	MPI Performance Mflops (4 processes)	Threaded/MPI (gain in favour of threads)
2x2x2x2	1258	1560	0.81
4x4x4x4	6572	6595	1
4x4x8x8	8120	7597	1.07
8x8x8x8	7929	8108	0.98
10x10x8x8	6668	5338	1.25
12x12x12x12	2465	2280	1.08
12x12x24x24	2340	2264	1.03

- On the whole threading seems to help some
- But not a lot... Can we do better?

Future Improvements

- Increase access to local vs remote memory
  - eg: interleave memory allocation between processors (libnuma
- If there are leftover cores, but memory bandwidth is exhausted – use core for something else (comms coprocessor, heater etc)
  - need to tweak API.
- Improvements likely to be architecture specific, depending on things such as
  - systems libraries and facilities (eg: libnuma)
  - actual node architecture
    - hardware memory strategies (number of controllers, available bandwidth), shared caches & coherency etc.
- Grand Unified Threading Interface will be challenging...

Chroma on BG/L with BAGEL Dslash 1.4.6

- BU BG/L & MIT BG/L – all regressions pass, some 1024 core tests fail at MIT - following up on this to determine cause of problems.
- Dslash Performance (BU BG/L)
  - single node, single core, Vol=4x4x8x8
    - Double Prec: 1328 Mflops/core (47% of peak)
    - Sloppy (single internal) Prec: 1521 Mflops/core (54% of peak)
  - 512 node, 1024 core, Local Vol =4x4x8x8, CPU Grid=8x8x8x2
    - Double Prec: 696 Mflops/core (24.8% of peak)
    - Sloppy Prec: 869 Mflops/core (31.1% of peak)
- Clover Inversion – in (R)HMC, 512 nodes, 1024 cores, vol=16x16x16x64, subgrid=**8x2x2x8**, cpu grid=2x8x8x8, Sloppy Prec, (BU BG/L)
  - Chroma Level 2 CG: 312 Mflops/core (11% of peak)
  - Chroma Level 2 Multi Shift CG (9 poles): 294 Mflops/core (10.5%)
- Need to try native QMP or QMP-MPI-2-1-7, track problem on MIT machine convert QDP_BLAS for double hummer if not done already.