Experiences with Lattice QCD on the Juelich BG/Q

S.Krieg, K.K.Szabo

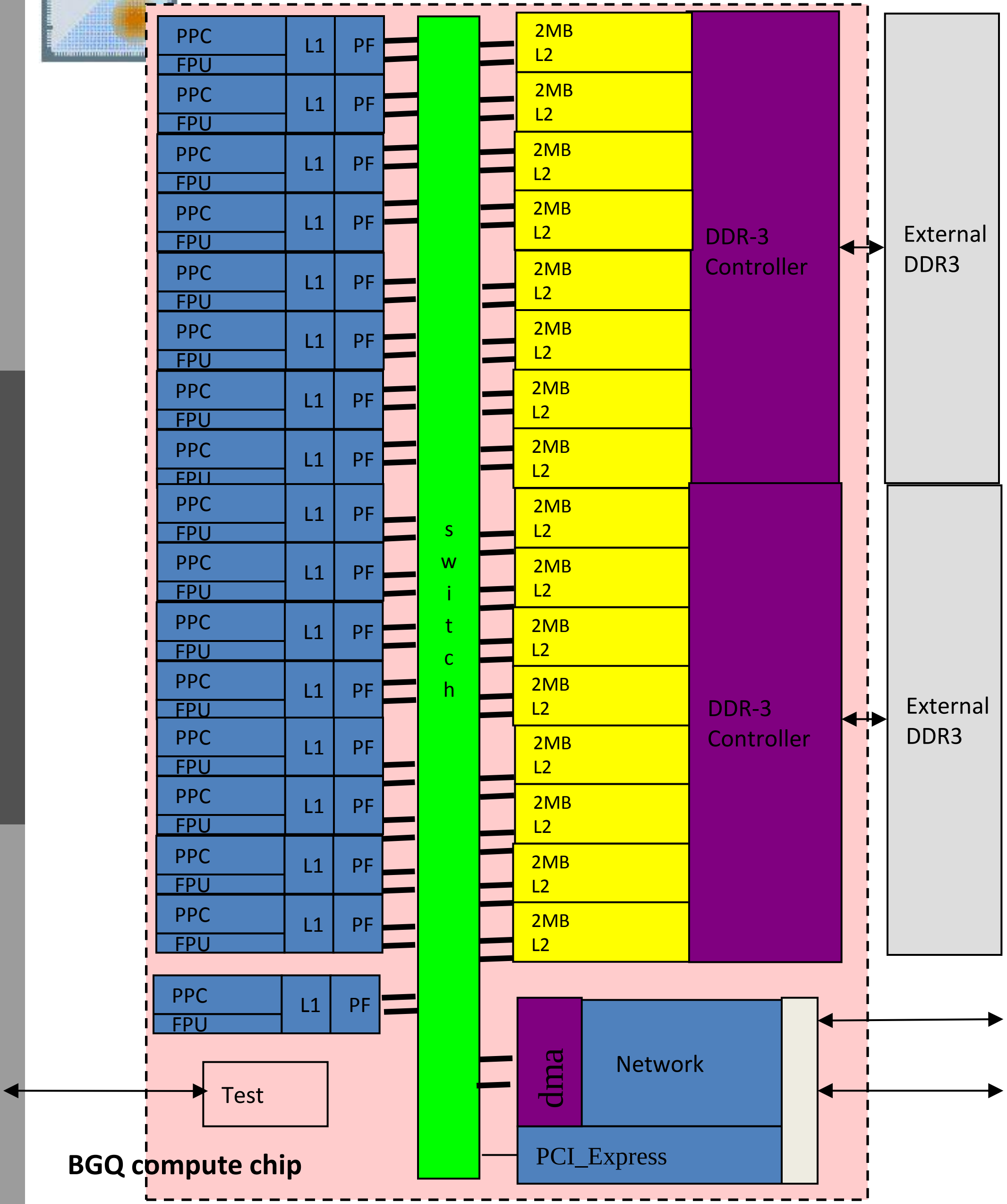
12. Juli 2011 | Theo Mustermann

(my) Machine

Specs:

- 28 Racks
- 7x2x2Racks
  = **28**x8x8x*8*x2 nodes
  = 28,672 nodes
  = 458,752 cores
  = 1,835,008 HW tds.
- 5.9 Pflop
- Top500 #7
- Public: PRACE, GCS, NIC local use (JARA)
- Production: Jan. 2013 (8->16->24->28)

BGQ Chip architecture



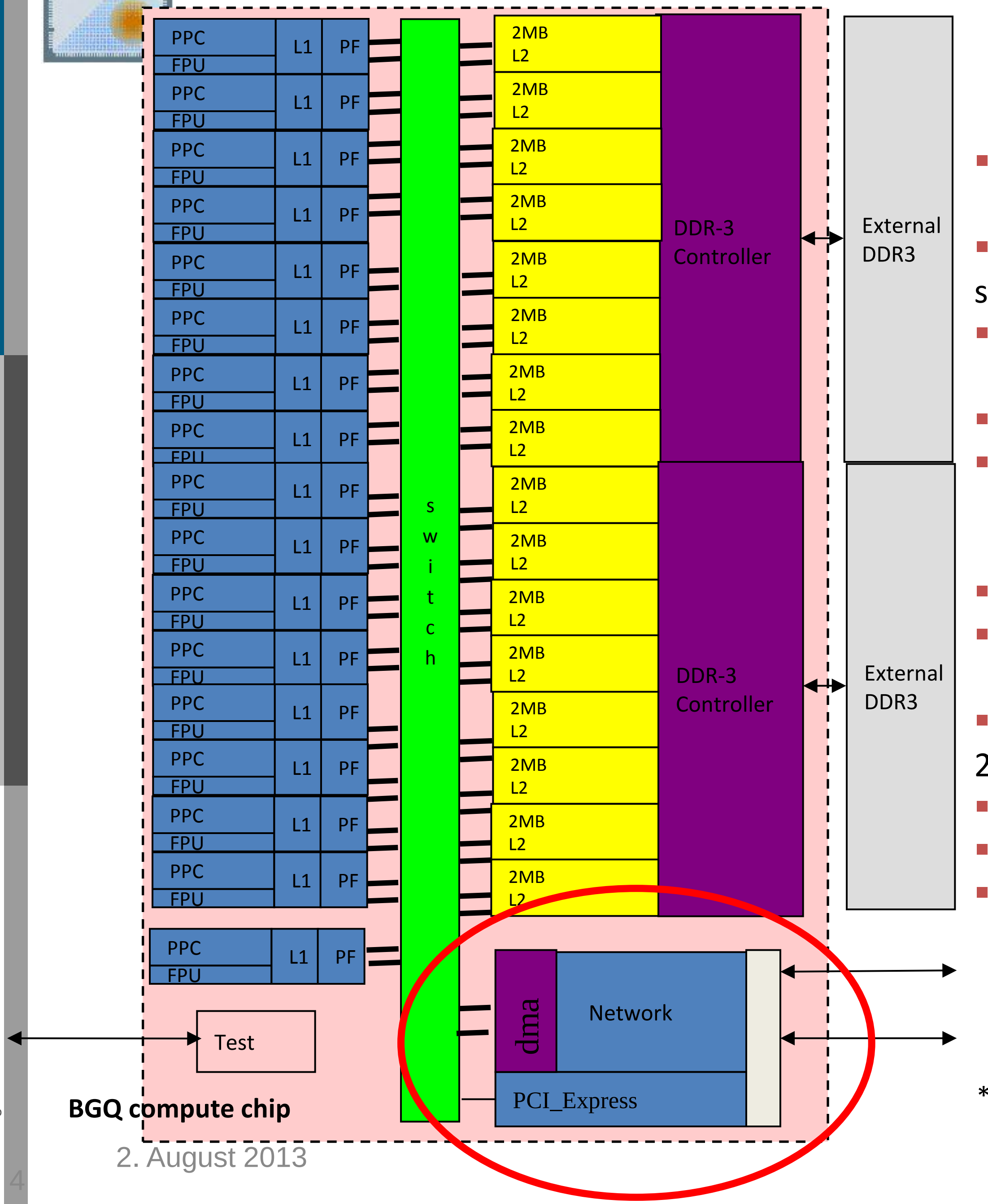
- 16+1 core SMP
  Each core 4 way hardware threaded
- Transactional memory and thread level speculation
- Quad float point unit on each core
  204.8 GF peak node
- Frequency target of (1.6) GHz
- 563 GB/s bisection bandwidth to shared L2
  (BGL at LLNL has 700 GB/s system bisection)
- 32 MB shared L2 cache
- 42.6 GB/s DDR3 bandwidth
  (2 channels each with chip kill protection)
- 10 intrarack interprocessor links each at 2.0GB/s
- 1 I/O link at 2.0 GB/s
- 4-8 GB memory/node
- ~30 Watts chip power

2 GB/s I/O link (to I/O subsystem)

10*2GB/s Intrarack (5-D torus)

** chip I/O shares function with PCI_Express

BGQ Chip architecture



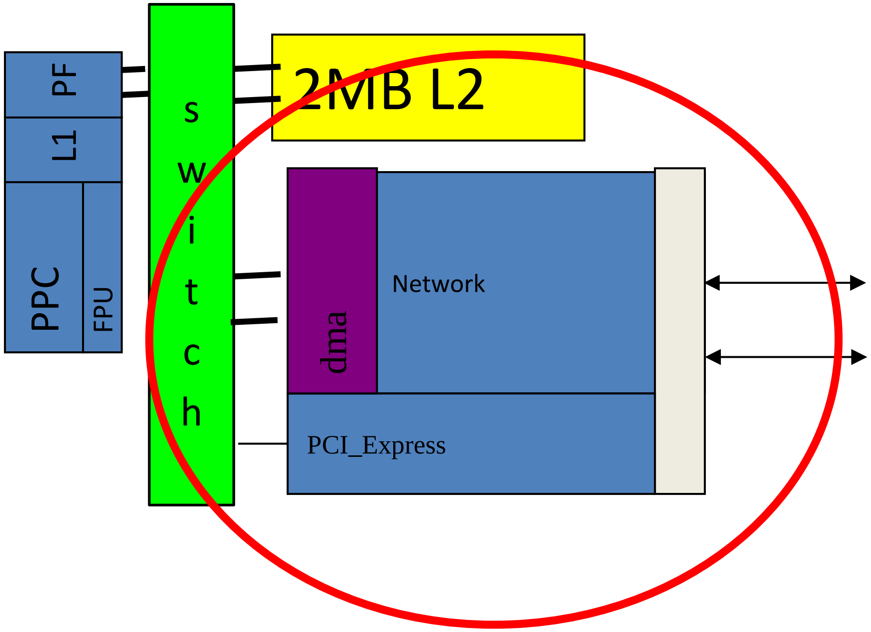
- 16+1 core SMP
  - Each core 4 way hardware threaded
- Transactional memory and thread level speculation
- Quad float point unit on each core
  - 204.8 GF peak node
- Frequency target of (1.6) GHz
- 563 GB/s bisection bandwidth to shared L2
  - (BGL at LLNL has 700 GB/s system bisection)
- 32 MB shared L2 cache
- 42.6 GB/s DDR3 bandwidth
  - (2 channels each with chip kill protection)
- 10 intrarack interprocessor links each at 2.0GB/s
- 1 I/O link at 2.0 GB/s
- 4-8 GB memory/node
- ~30 Watts chip power

2 GB/s I/O link (to I/O subsystem)

10*2GB/s Intrarack (5-D torus)

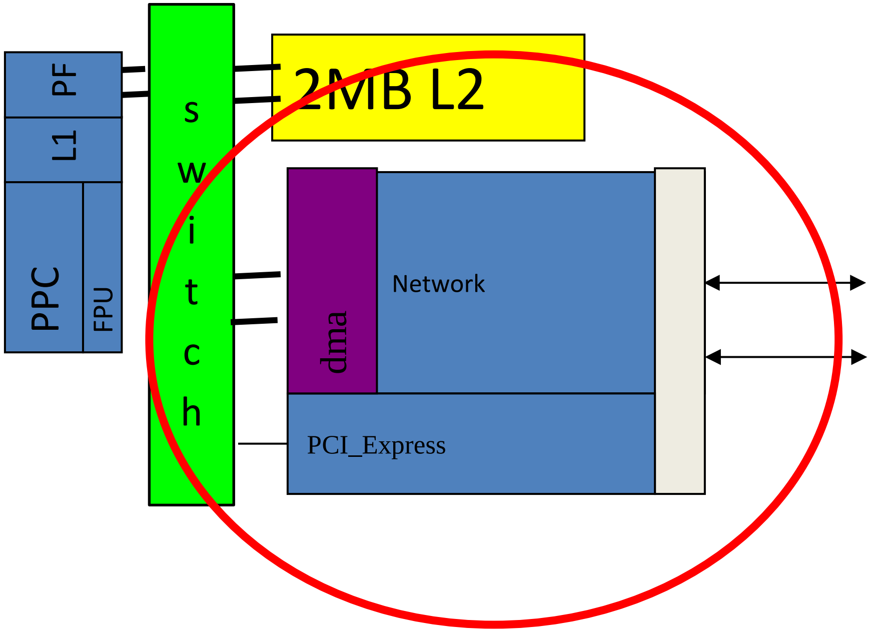
** chip I/O shares function with PCI_Express

Messaging unit (mu)

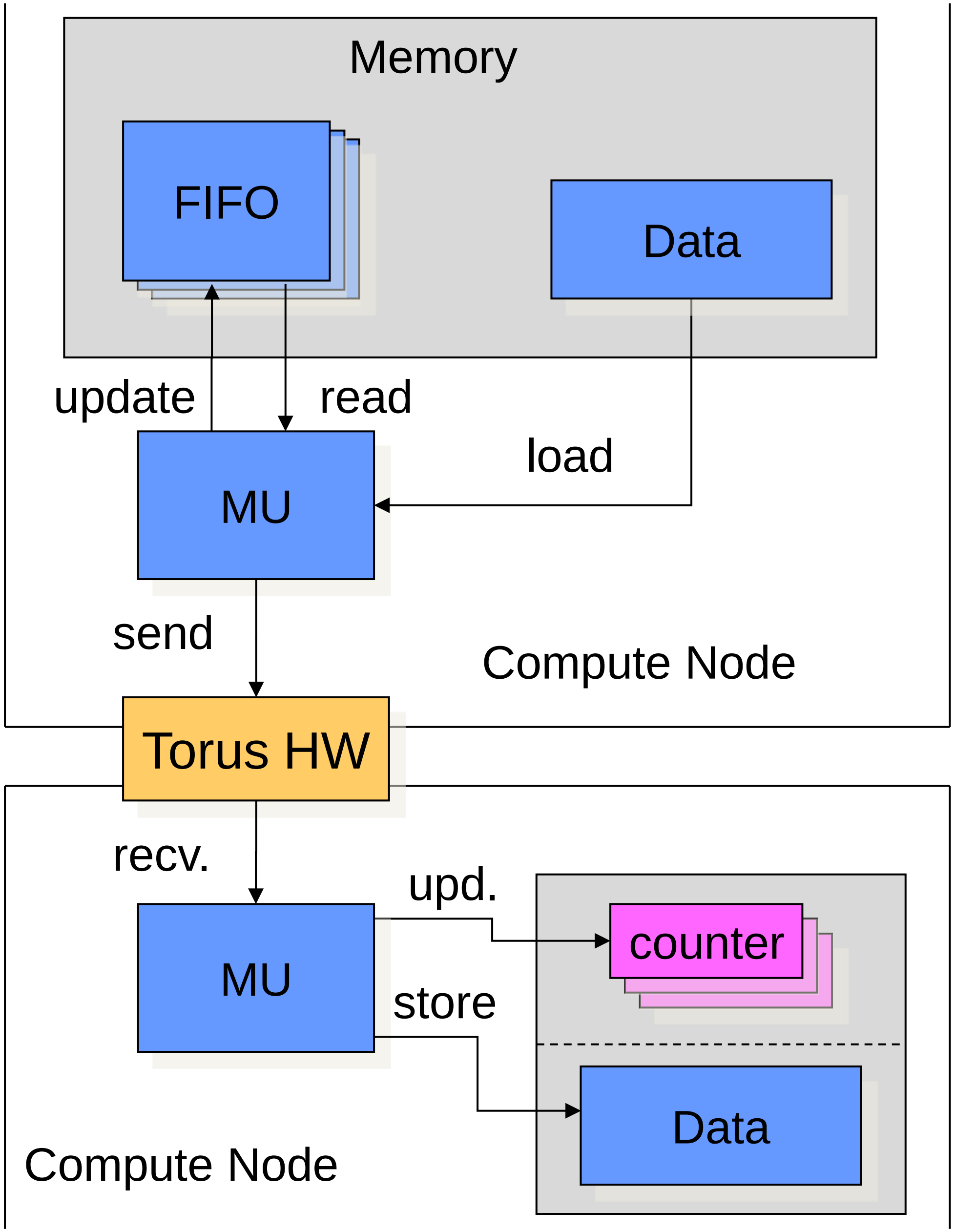


- Direct access to L2
- Direct access to network HW
- Fully user programmable
- Performs PtP and collective communications
- Runs independent of cores:
  - Sends data
  - Receives data and stores to memory subsystem
- Shared resource for all 17 cores

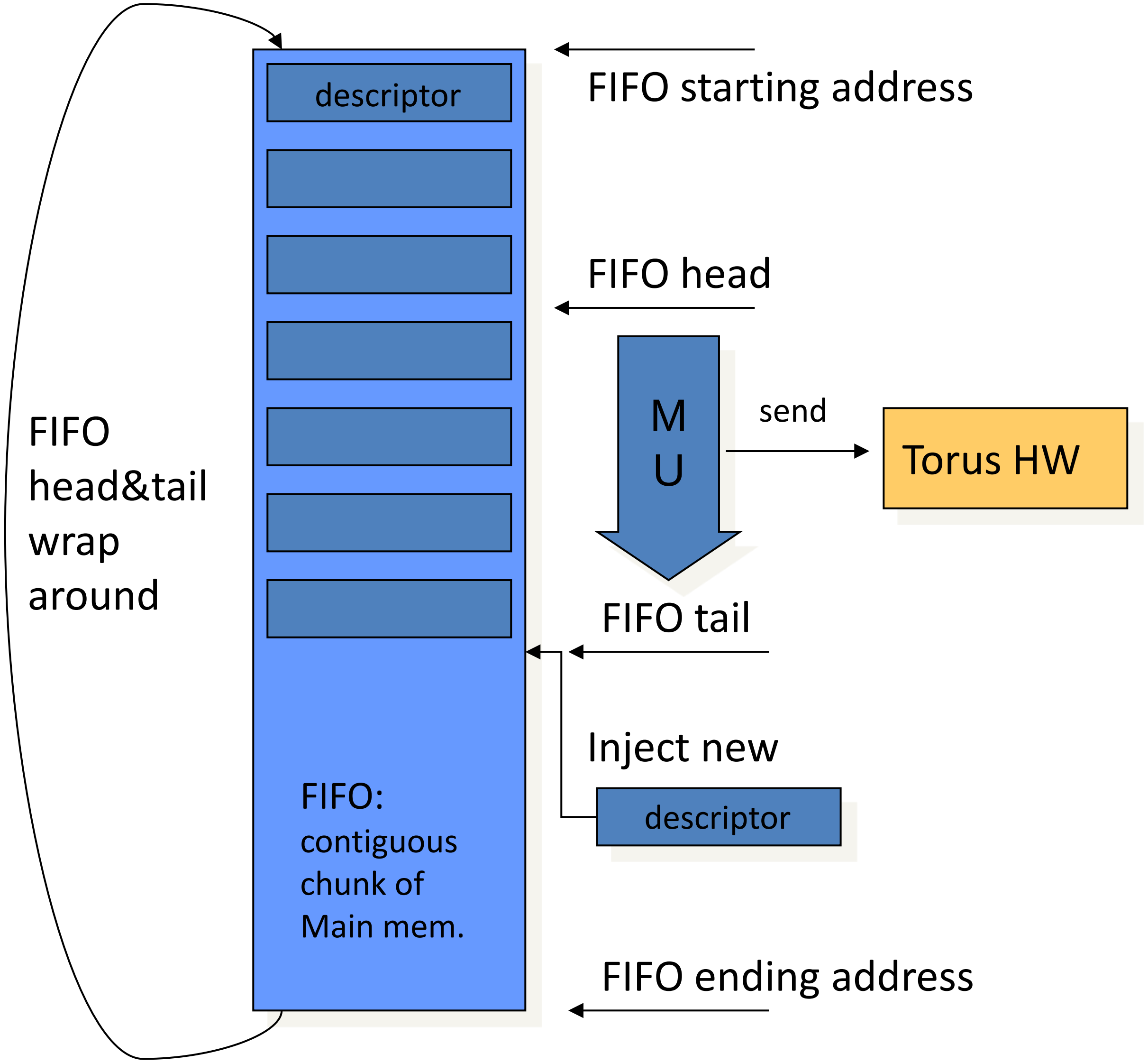
Messaging unit (mu)



- 17 groups
  - 4 subgroups
    - 8 inj fifos
    - 4 rec fifos
    - 8 base address table (bat) entries
- "unlimited" counters:
  - implemented in L2 using atomics
  - bat entry required


Memory
FIFO
Data
update
read
load
MU
send
Compute Node
Torus HW
recv.
upd.
MU
counter
store
Data
Compute Node

"direct put"


FIFO starting address
descriptor
FIFO head
M
U
send
Torus HW
FIFO head&tail wrap around
FIFO tail
Inject new
descriptor
FIFO: contiguous chunk of Main mem.
FIFO ending address

PtP Communication

For a “direct-put“ using SPI one could proceed as follows:

- Allocate 1 inj. FIFO, 1 rec. counter
- Create "mem-region" for data to be sent
- (destination node) Create "mem-region" for reception window
- (destination node) Set bat entry to point to rec. window
- (destination node) Set rec counter to #bytes to be received

Synchronize (could be superfluous)

For each continuous chunk of data, inject 1 descriptor

- Calculate the address offset relative to rec. base (need # bat on rec)
- Give the individual message size

"Wait"

- Make sure all data has been injected before modifying the buffer!
- (destination node) poll reception counter for comm completion

Mitglied der Helmholtz-Gemeinschaft

LQCD code

- Vanilla (C/MPI), Cuda, BG/Q (extensive use of macros)
- Threaded (pthreads, master/workers)
- Parallelization strategy for BG/Q:
  - 16/4 (AxBxCxDxEx16)(x4)
  - Use shmem window to communicate between processes
  - Synchronize threads/processes with A2 barrier (shmem)
  - Node layout: wrap 2 dimensions into a 4d torus
- Implementation strategy:
  - Use permutes

→ Always stay in 2nd level cache

Communication

- SPI based
- Code interface requires flexible communication using tags/communicators
- Standard comms are made persistent and are freed in times of need
- Restarts check if the tag/communicator has been force freed, otherwise performs a simple restart
- Works in SMP (1/64) or multiple process modes (eg. 16/4)
- Global comms proceed via SMP window (shm_open/mmap)
- PtP comms contain no global ops (i.e. no global sycs)

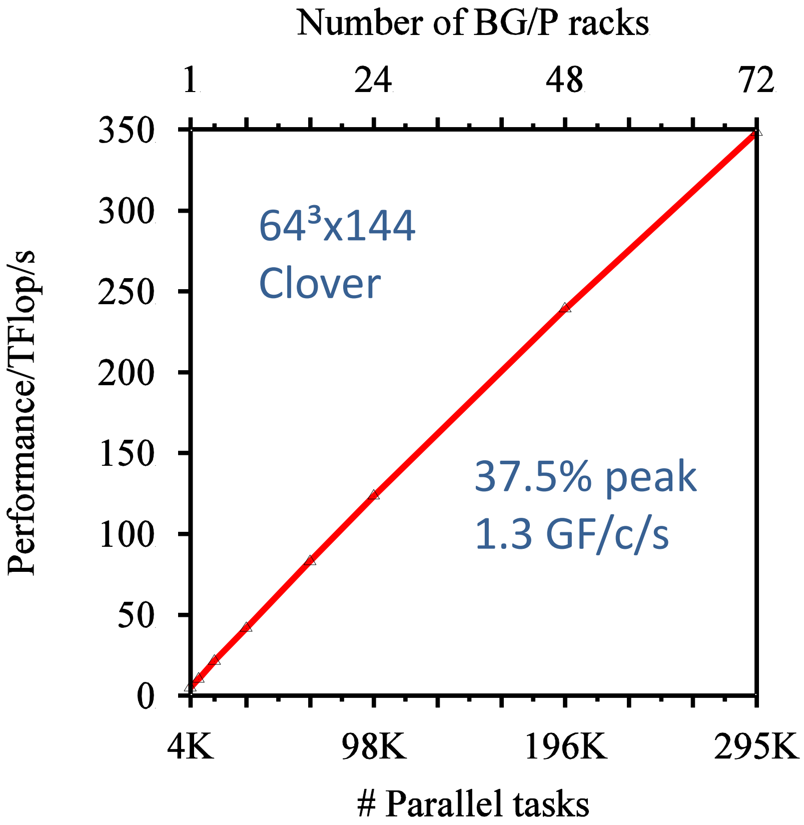
Dslash

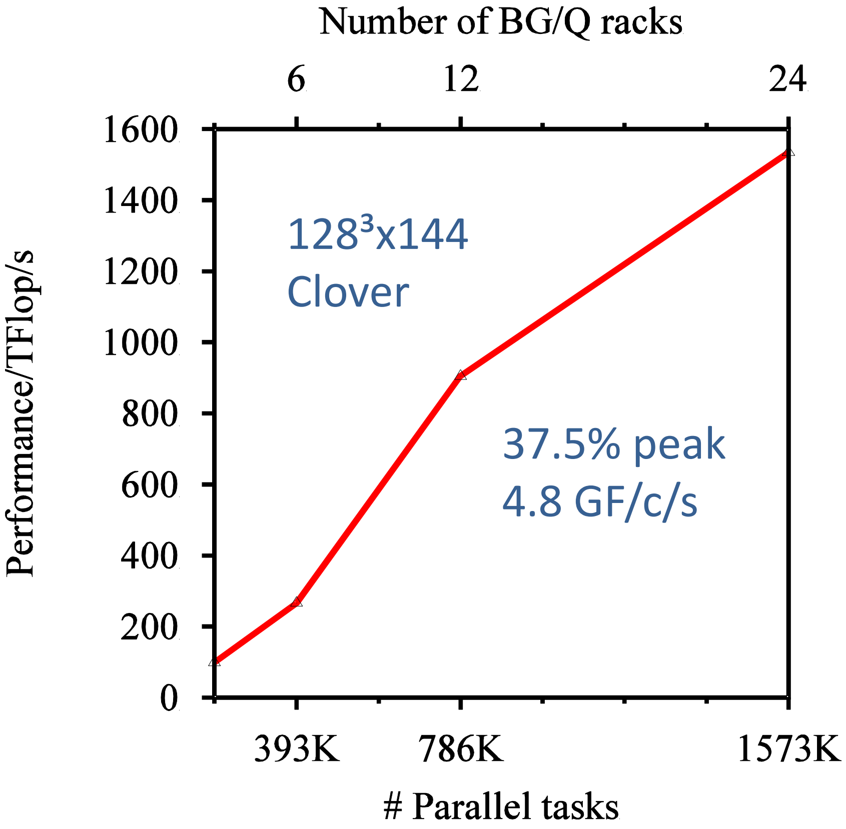
- ‘classical’ volume layout
- Standard 2spin approach
- Use permutes and ld2 to rearrange data:
  - Gauge field required by both 2spin components
  - Locate 2spins in registers
  - Load gauge field with ld2
  - Caveat: ld2 has extra latency (approx. 2 x ld)
- Clover term integrated in last merge
- Communication persistent by default

2spin dslash

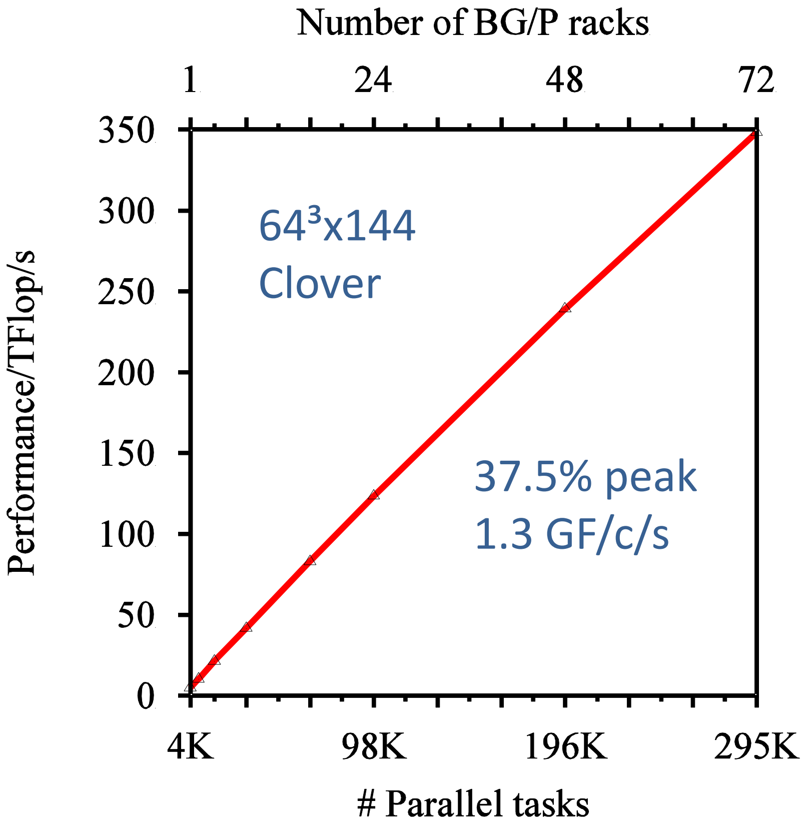
1. (scalar) Spin project forward
2. (comm) Start communication forward
3. (scalar) Spin project backward and SU(3) multiply
4. (comm) Start communication backward; Wait forward
5. (scalar) SU(3) multiply fwd. and sum up
6. (comm) Wait backward
7. (scalar) Add backward

Performance



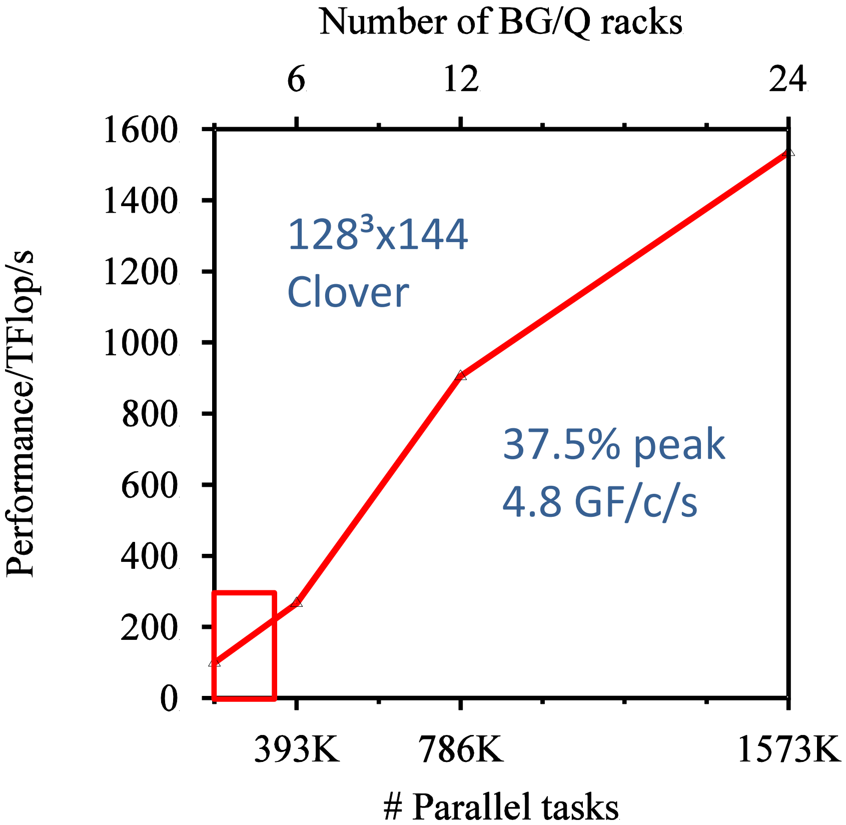


Performance



64³x144
Clover

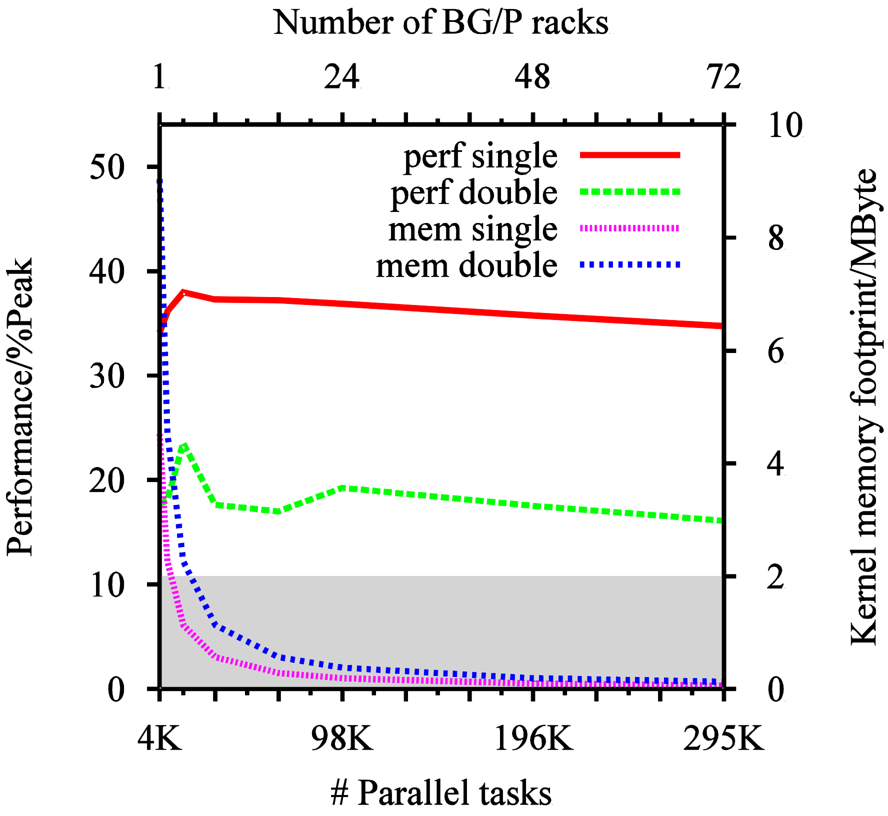
37.5% peak
1.3 GF/c/s

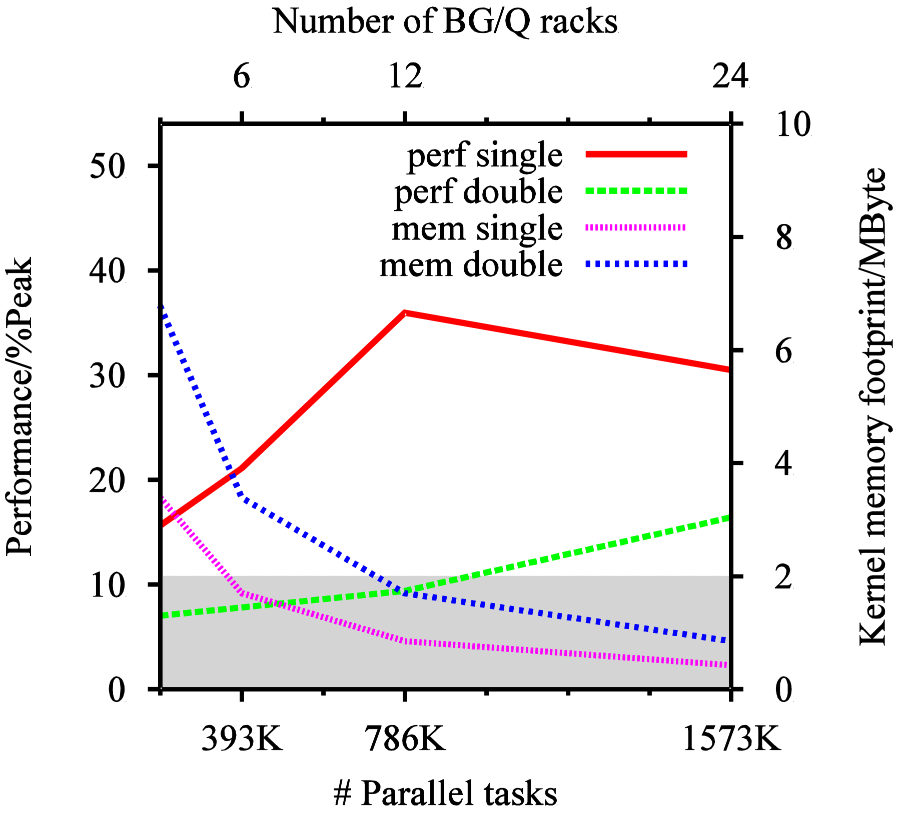


128³x144
Clover

37.5% peak
4.8 GF/c/s

Performance (BG/P)





Solver performance (production)

- Multishift CG: 3.2 Gflop/core/s (3x6x6x6 loc. lat.)
  (CG more efficient, BiCGstab similar)
- Multilevel method (Frommer et al. 1303.1377, 1307.6101)
  - 2 level implemented so far
  - Smoother: 3.4 Gflop/core/s (12x6x6x3 loc. lat., 48^3x96)
  - Restriction/interpolation: 3.6/6.6 Gflop/s
  - Coarse grid operator: 1.2/2.2 Gflop/s (2x2x1x1 loc. lat.)
  - Total 1.9 Gflop/s
  - Setup: 2.4 Gflop/s

Conclusions

- Efficiency on BG/Q appears to much more peaked when compared to BG/P
- Peak value remains unchanged
- Solver performance noticeably lower than for dslash
- Continuing tendency towards more complex solvers (e.g. MG)

→ Tuning becomes more difficult

→ So far, performance results of complex solvers do not match those of ordinary ones