Lattice QCD Performance

Computing Constraints

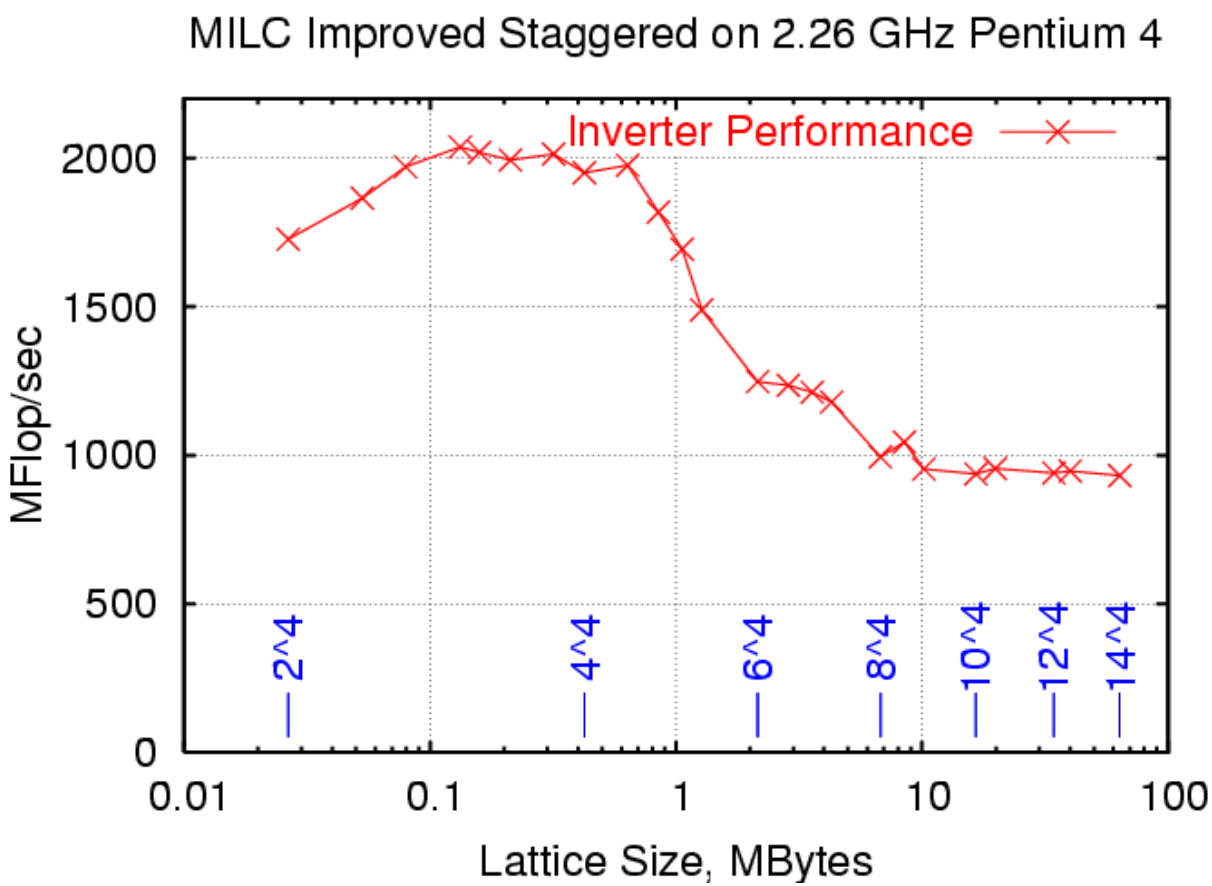
- The dominant computation is the repeated inversion of the Dirac operator
  - Equivalent to inverting large 4-D and 5-D sparse matrices
  - Conjugate gradient method is used
  - The current generation of calculations requires on the order of Tflop/s-yrs to produce the intermediate results (“vacuum gauge configurations”) that are used for further analysis
  - ~ 50% of Flops are spent on configuration generation, and ~ 50% on analysis using those configurations

Near Term Requirements

- Lattice QCD codes typically sustain 30% of the Tflop/s reported by the Top500 Linpack benchmark.
- Planned configuration generation campaigns in the next few years:

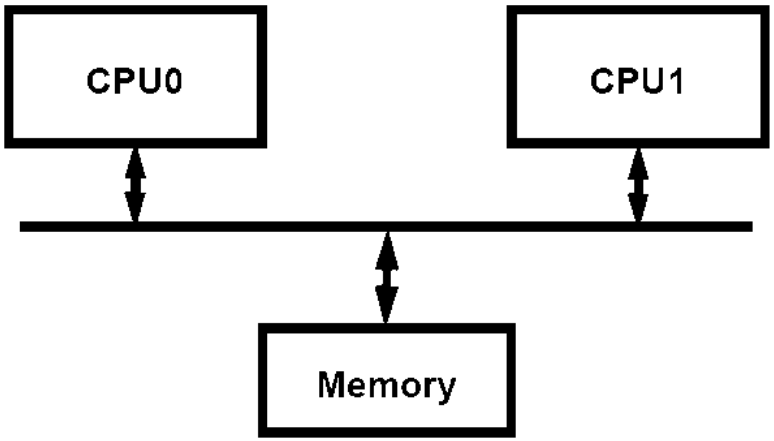
QCD Action	Lattice Spacing (Fermi)	m_l/m_s	Lattice Dimensions	Lattice Size (GB)	TF-Years
ASQTAD	0.060	0.10	$60^3 \times 144$	9.0	2.0
ASQTAD	0.045	0.20	$56^3 \times 192$	9.7	1.9
ASQTAD	0.045	0.10	$80^3 \times 192$	28.3	13.7
ASQTAD	0.060	0.05	$84^3 \times 144$	24.6	23.2
DWF	0.094	0.27	$32^3 \times 64$	0.6	1.2
DWF	0.094	0.19	$48^3 \times 64$	2.0	7.8
DWF	0.094	0.11	$48^3 \times 64$	2.0	25.2
CLOVER	0.100	0.22	$32^3 \times 128$	1.2	0.8
CLOVER	0.100	0.15	$40^3 \times 128$	2.4	4.1
CLOVER	0.080	0.18	$40^3 \times 128$	2.4	4.5
CLOVER	0.080	0.15	$48^3 \times 128$	4.1	22

Generic Single Node Performance

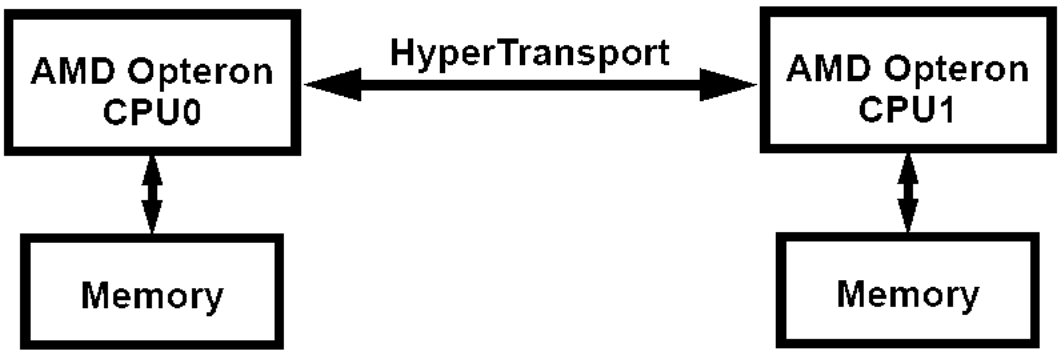


- Graph shows performance of the conjugate gradient Dirac operator inverter
- Cache size = 512 KB
- Floating point capabilities of the CPU limits in-cache performance
- Memory bus limits performance out-of-cache
- We care about 10^4 and larger

Memory Architectures

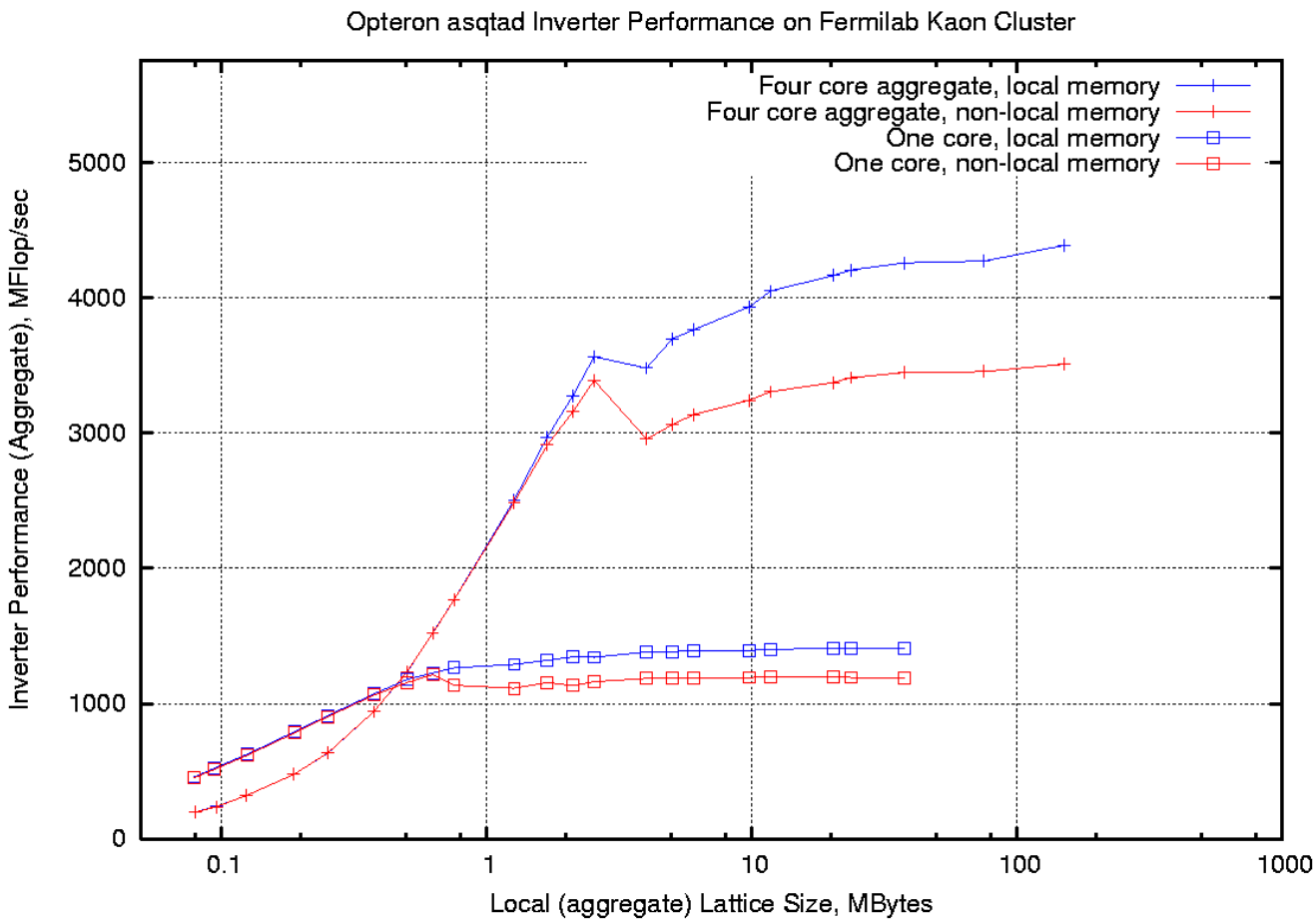


Intel Xeon SMP Architecture



AMD Opteron SMP Architecture

NUMA Effects



- Plots show performance of (bottom lines) single code instances and (top lines) aggregate performace of 4 MPI processes
- Non-local memory used (via numactl)
- Local memory used (via numactl)
- On NUMA clusters local memory must be used *and* processes must be locked to cores

Performance: Memory Bandwidth

- How to measure:
  - STREAMS “Copy” benchmark is correlated with performance
  - On multi-core machines, we use an OpenMP version of STREAMS to thread the inner loops
  - Very important: aggressive optimizations improve STREAMS numbers but are not relevant to LQCD code
    - Biggest performance gain comes from writing around cache (non-temporal writes)
    - Unfortunately these writes require memory alignments that are not compatible with the sizes of SU(3) data structures

Performance: STREAMS

CPU	Machine	Memory Type	Single Core	All cores (#)
2.93 GHz Pentium dual	Single socket	DDR2	3104 MB/sec	3085 MB/sec (2 cores)
2.66 GHz Xeon dual core	Dual socket	FB-DIMM	2712	5043 (4 cores)
2.4GHz Xeon Quad	Quad Socket	FB-DIMM	2732	8194 (16 cores)
2.0 GHz AMD Dual	Dual Socket	DDR	2368	5426 (4 cores)
2.6 GHz AMD Dual	Dual Socket	DDR2	2590	6693 (4 cores)
1.9 GHz Barcelona	Dual Socket	DDR2	2725	8123 (8) cores
1.9 GHz Barcelona	Dual Socket	DDR2	3236	10667 (8 cores)
2.1 GHz Barcelona	Dual Socket	DDR2	3900	13056 (8 cores)

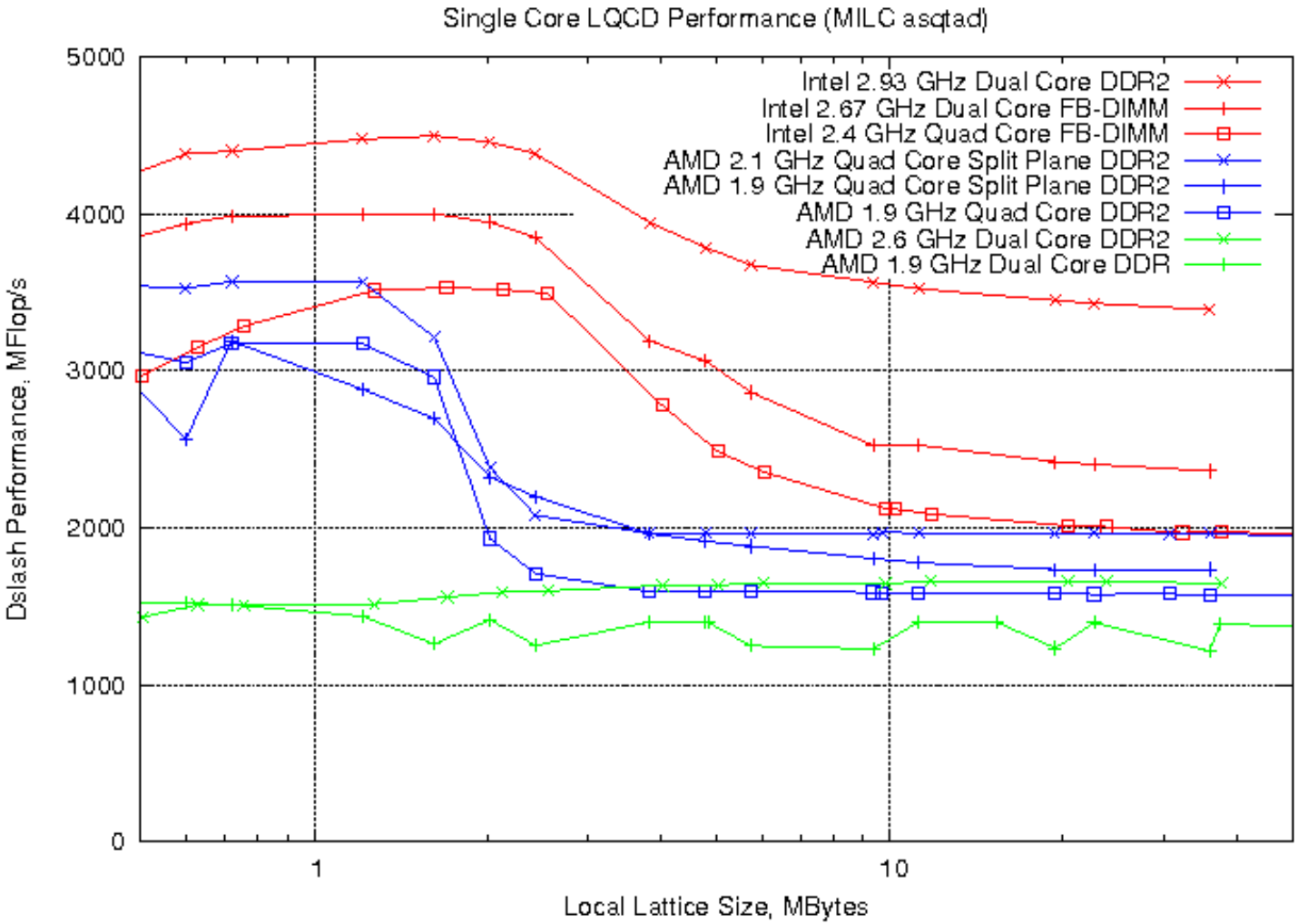
- All Intel cpus shown are “Core” microarchitecture
- AMD Barcelona – split-power plane (clock memory and processor separately) increases bandwidth (indicated in blue)

Performance: Single Node LQCD

CPU	Machine	Memory Type	Single Core	All cores (#)
2.93 GHz Pentium dual	Single socket	DDR2	3367 MFlops	3637 MFlops (2 cores)
2.66 GHz Xeon dual core	Dual socket	FB-DIMM	2363	4745 (4 cores)
2.4GHz Xeon Quad	Quad Socket	FB-DIMM	1800	6872 (16 cores)
2.0 GHz AMD Dual	Dual Socket	DDR	1387	4415 (4 cores)
2.6 GHz AMD Dual	Dual Socket	DDR2	1650	4807 (4 cores)
1.9 GHz Barcelona	Dual Socket	DDR2	1624	5556 (8) cores
1.9 GHz Barcelona	Dual Socket	DDR2	1792	7490 (8 cores)
2.1 GHz Barcelona	Dual Socket	DDR2	1965	8291 (8 cores)

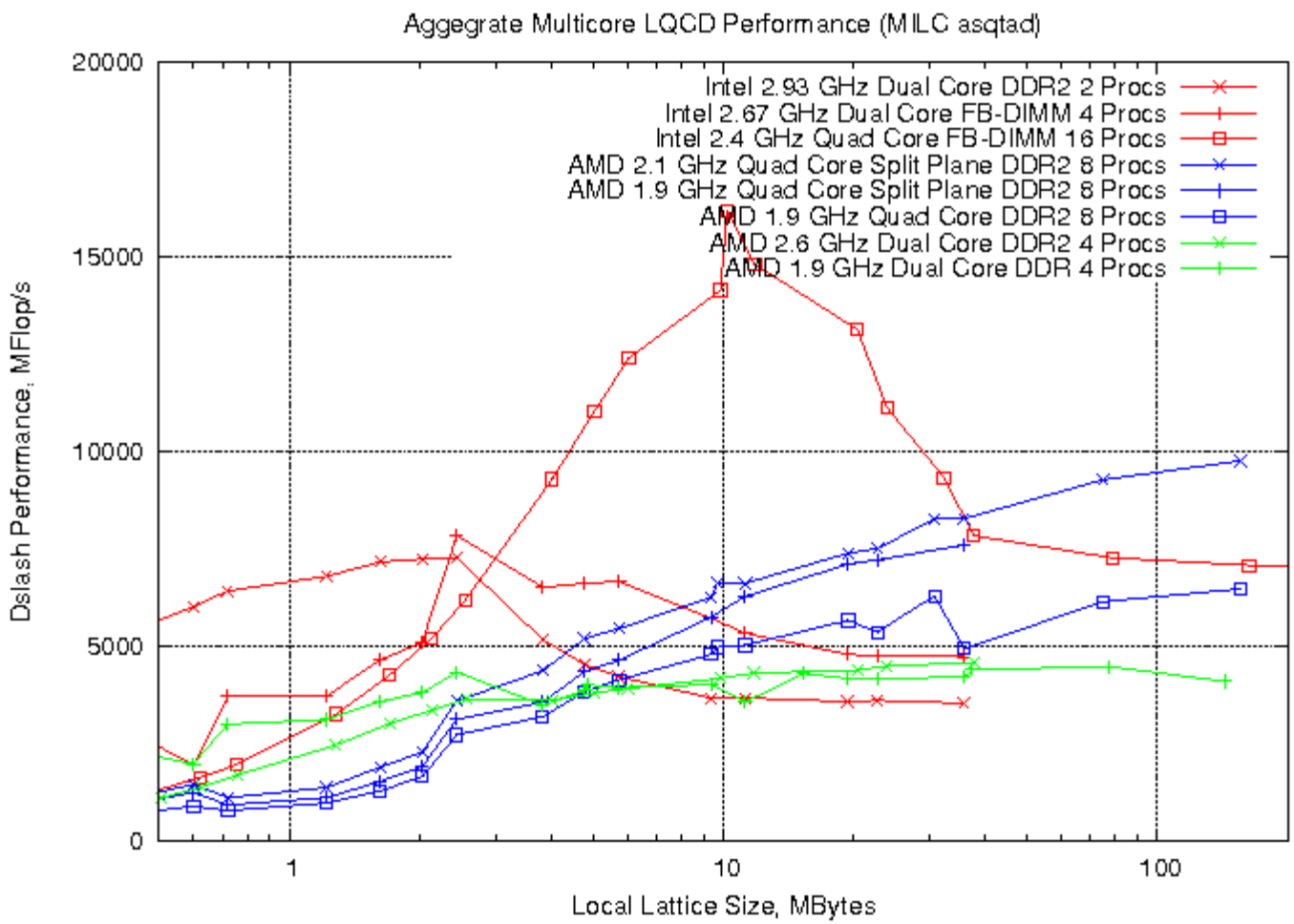
- Data are performance in *dslash* inverter using 14^4 lattice in each core
- Shared memory MPICH used for message passing

Performance: Single Node, Single Core LQCD



- Plots show performance of single code instance
- Intel Dual and Quad "Core" architecture
- AMD Quad (Barcelona)
- AMD Dual Socket-940 and Socket-F

Performance: Single Node, Single Core LQCD



- Plots show aggregate performance of one MPI process on each core
- In region of interest (> 50 Mbytes) the AMD quad-core systems had best price/performance