Commodity Cluster Usage for Lattice Field Theory

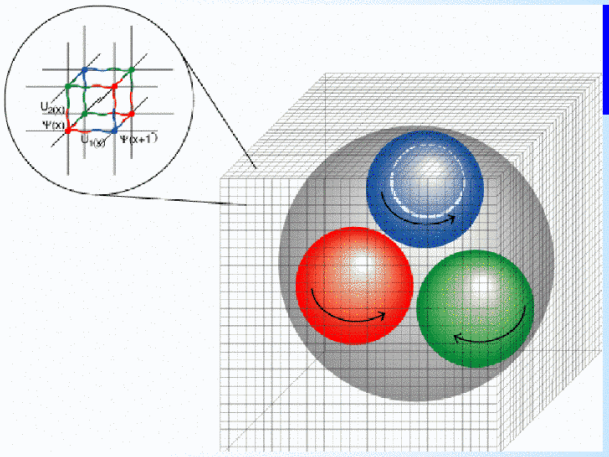
Robert Edwards
Jefferson Lab

Collaborators:
Chip Watson, Walt Akers, Jie Chen, Ying Chen,
Balint Joo, Andrew Pochinsky

Lattice QCD – extremely uniform

Dirac operator:

$$D\psi(x) = \sum_{\mu} \left(\partial_{\mu} + igA_{\mu}(x)\right)\psi(x)$$



- Periodic or very simple boundary conditions
- SPMD: Identical sublattices per processor

Lattice Operator:

$$D\psi(x) = \frac{1}{2a}\sum_{\mu}\left[U(x)\psi(x+\hat{\mu}) - U^{\dagger}(x-\hat{\mu})\psi(x-\hat{\mu})\right]$$

Clusters and HPC

- The motivation to use clusters is two-fold:
  - Assemble a large computational resource
  - Achieve teraflops performance without spending $10+M
- Relevant facts:
  - Moore's Law – processor price performance ~ 60% / year
  - High volume market
    - Drives component cost low
    - Newer components every few months
    - Allows increased capability each year at constant investment
  - Home video gaming
    - CPU vector extensions - 5 Gflops sustained (single precision) on a 2 GHz Pentium 4
    - Scaling to a cluster is the challenge!
  - Cluster interconnects maturing – ever larger clusters from semi-commodity parts

Challenges to Cluster Computing

- Clusters (as compared to large SMPs) face certain architectural challenges:
  - Distributing work among many processors requires communications (no shared memory)
  - Communications is slow compared to memory R/W speed (both bandwidth and latency)
- The importance of these constraints is a strong function of the application, and of how the application is coded (strategy).

Understanding the Requirements: LQCD Behavior

- Regular grid - 4D or possibly 5D problem
- Periodic boundary conditions – torus/mesh
- Time consumer – Dirac operators. Consider Wilson-like.
  - SU(3) multiplication onto spin – good mem. reuse
  - Algorithm splits into forward, backward portions; latency tolerance: 80% overlap possible

$$D\psi(x) = \frac{1}{2a}\sum_{\mu}\left[U(x)\psi(x+\hat{\mu}) - U^{\dagger}(x-\hat{\mu})\psi(x-\hat{\mu})\right]$$

  - For CG, frequency of global operations (barriers) per flop is fairly low in 4D – even smaller in 5D

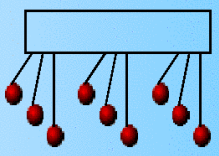
Cluster Architectures

- Switched
  - Ethernet: modest bandwidth, high latency, low cost
  - Myrinet: better bandwidth, lower latency, semi-commodity = moderate cost
  - Infiniband: good bandwidth, lower latency, semi-commodity = moderate to high cost
  - Quadrics: good bandwidth, low latency, high cost
- Mesh
  - Eliminates the cost of the switch
  - High aggregate bandwidth through multiple links
  - Still suffers from ethernets higher latency

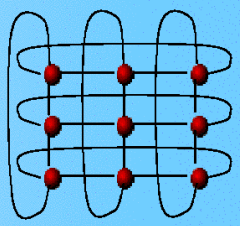
Motivation

Different network architectures suit different applications.

- Switched network:
  - General parallel computing platform
  - Any-to-any communication paths



- Multi-dimensional Mesh Connections (torus):
  - Good platform for nearest neighbor communications.
  - Potentially higher total bandwidth per node

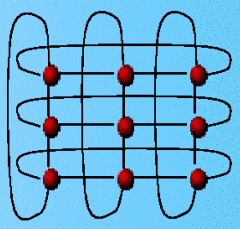


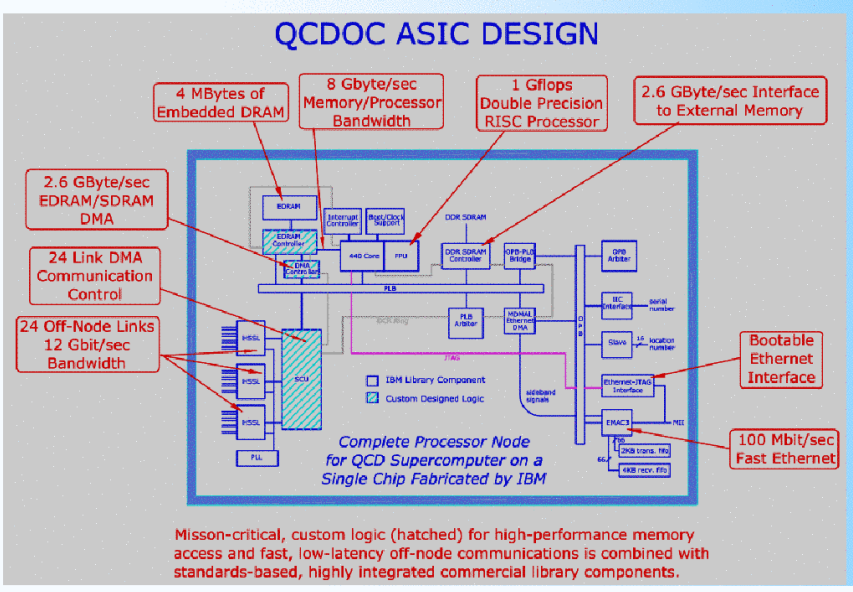
- Lattice QCD requires primarily nearest neighbor and some global communication.

LQCD and Mesh Machines

Lattice QCD has a history of using mesh architectures

- QCDSP (DSP based, w/ custom I/O chip)
- APE (Italian design)
- QCDOC (QCD On a Chip) – balanced design, but frozen technology





- Each node takes a 4D sub lattice, a portion of the problem.
- Lattice must be a multiple of the number of nodes in each direction.

Designing a Cluster

- The overriding metric is science per dollar, or $/Mflops
  - Peak is irrelevant.
  - Percent of peak is irrelevant.
  - Petaflops are needed (eventually), so cost matters!
- Intel Pentium 4 is a good building block
  - Map SU(3) matrix ops onto the SSE registers and instructions
  - P4 achieves > 2.5 flops / Hz for problems resident in L2 cache.
  - Memory bandwidth is a severe constraint
    - Memory Front-Side Bus (FSB) major bottleneck
    - Dual Xeons barely outperform single Xeon for non cache resident problems (NUMA???)
  - I/O (communications) is also a strong constraint =>
  - Until very recently - Xeon chips and chipsets (PCI-X) required: PCI-32 too slow
- IBM cluster nodes, Itaniums and Apple G5 too expensive.

SciDAC Prototype Clusters

The SciDAC project is funding a sequence of cluster prototypes which allow us to track industry developments and trends, while also deploying critical compute resources.

Myrinet + Pentium 4

- 48 dual 2.0 GHz P4 at FNAL (Spring 2002)
- 128 single 2.0 GHz P4 at JLab (Summer 2002)
- 128 dual 2.4 GHz P4 at FNAL (Fall 2002)

Gigabit Ethernet Mesh + Pentium 4

- 256 (8x8x4) single 2.66 GHz P4 at JLab (2003)
- 384 (8x6x4) single 2.80 GHz P4 at JLab (2004)

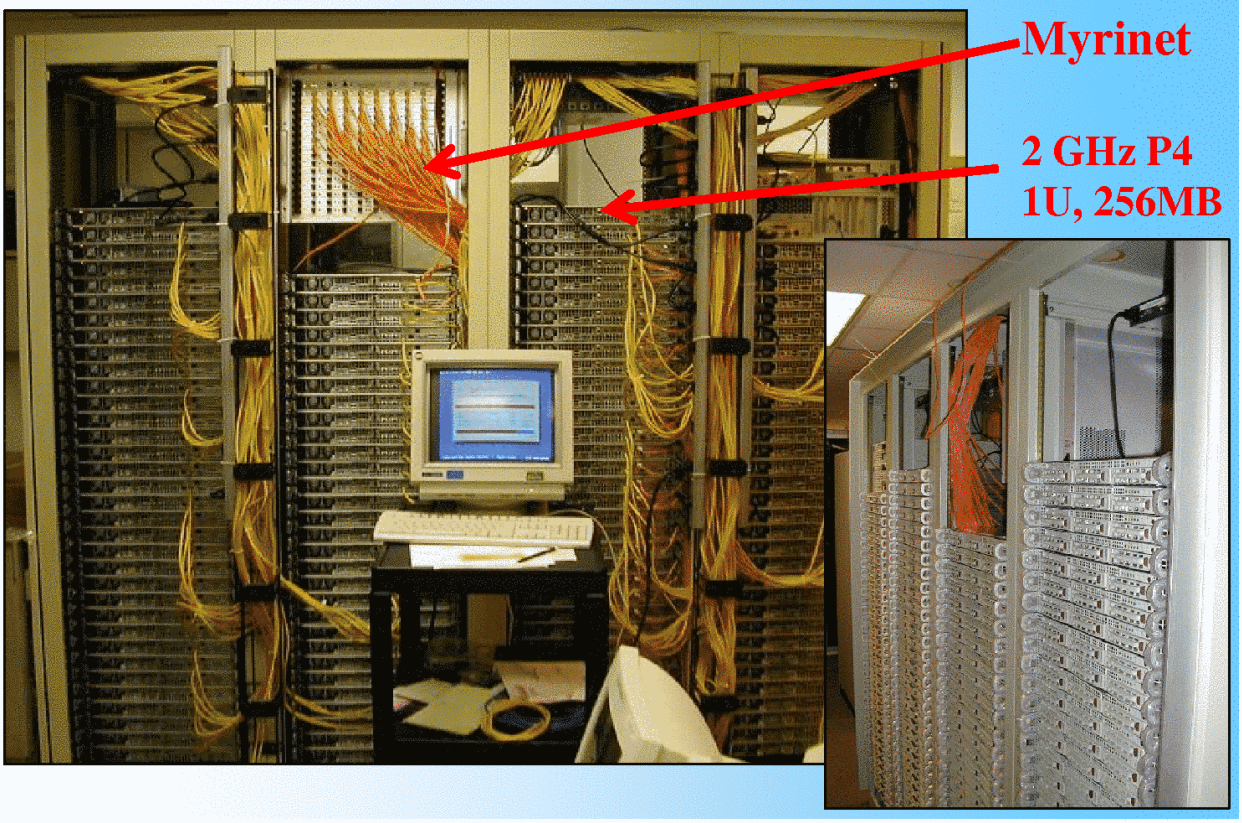
Infiniband + Pentium4

- 256 dual XXX GHz P4 at FNAL (Summer 2005)

Additional Technology Evaluations at FNAL

- Itanium 2, AMD Opteron, IBM PowerPC 970
- Infiniband

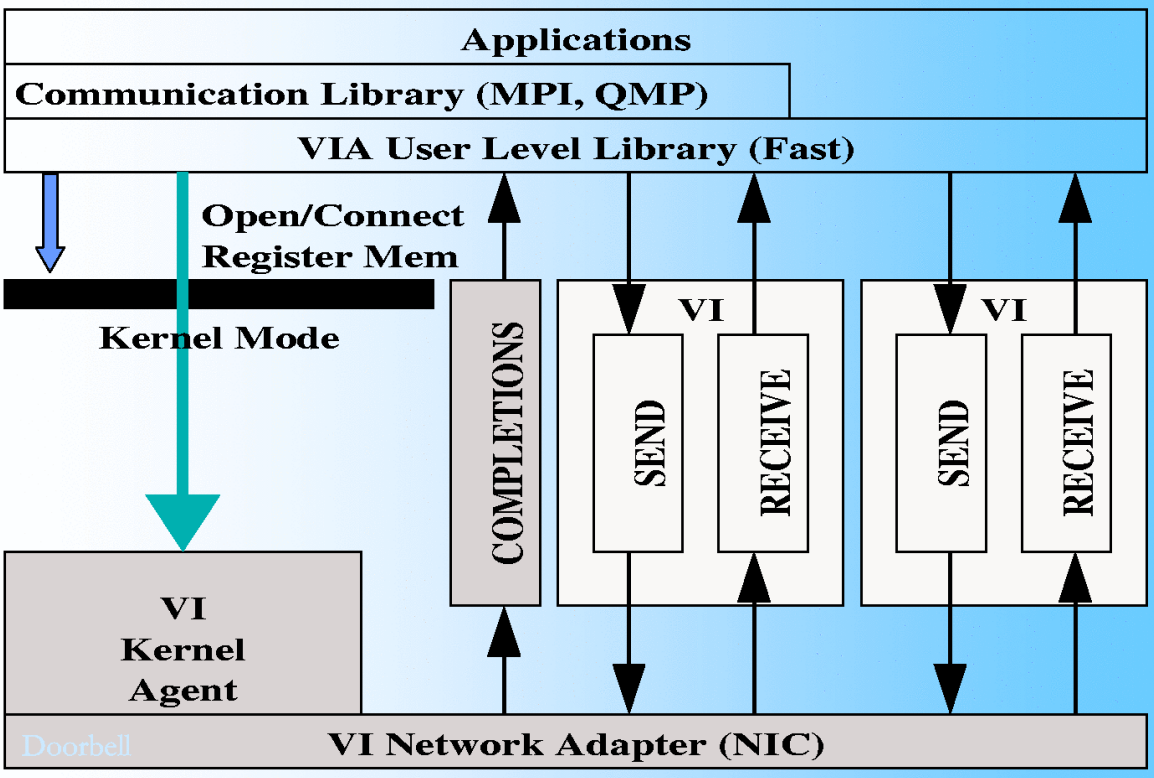
128 Node Cluster @ JLab



What about GigE?

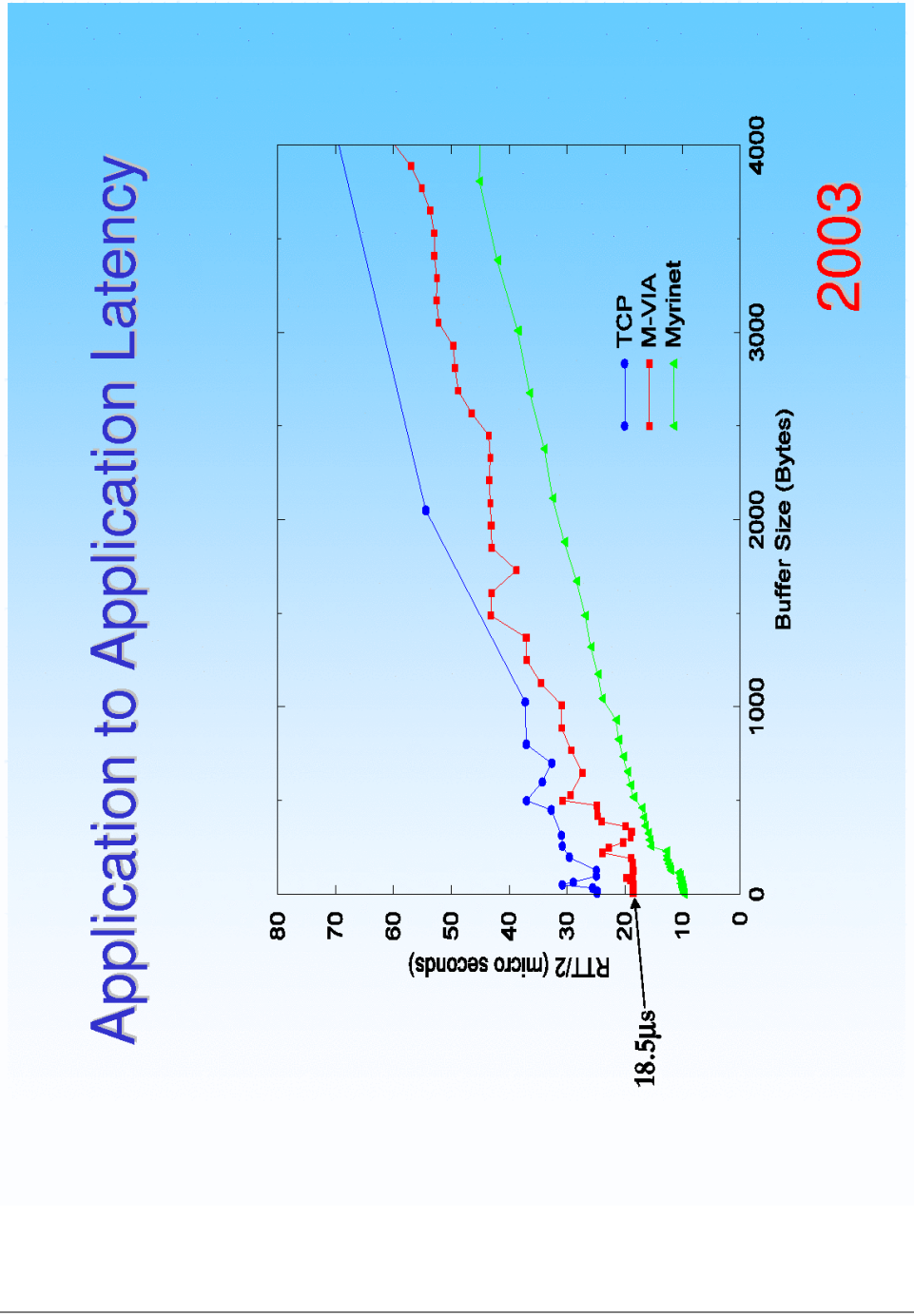
- Cost:
  - Myrinet: $1400 per node (switch + NIC) up to 256, $1800 for 1024 (2003). $1200/node (2005)
  - Infiniband: $1200 (switch + NIC) up to 256 (2005)
  - GigE: need multiple cards – avoid collapsing dimensions
  - GigE mesh: <$500 per node (3 dual gigE cards, 1 per dimension).
- Need efficient user space code:
  - TCP/IP consumes too much of the CPU

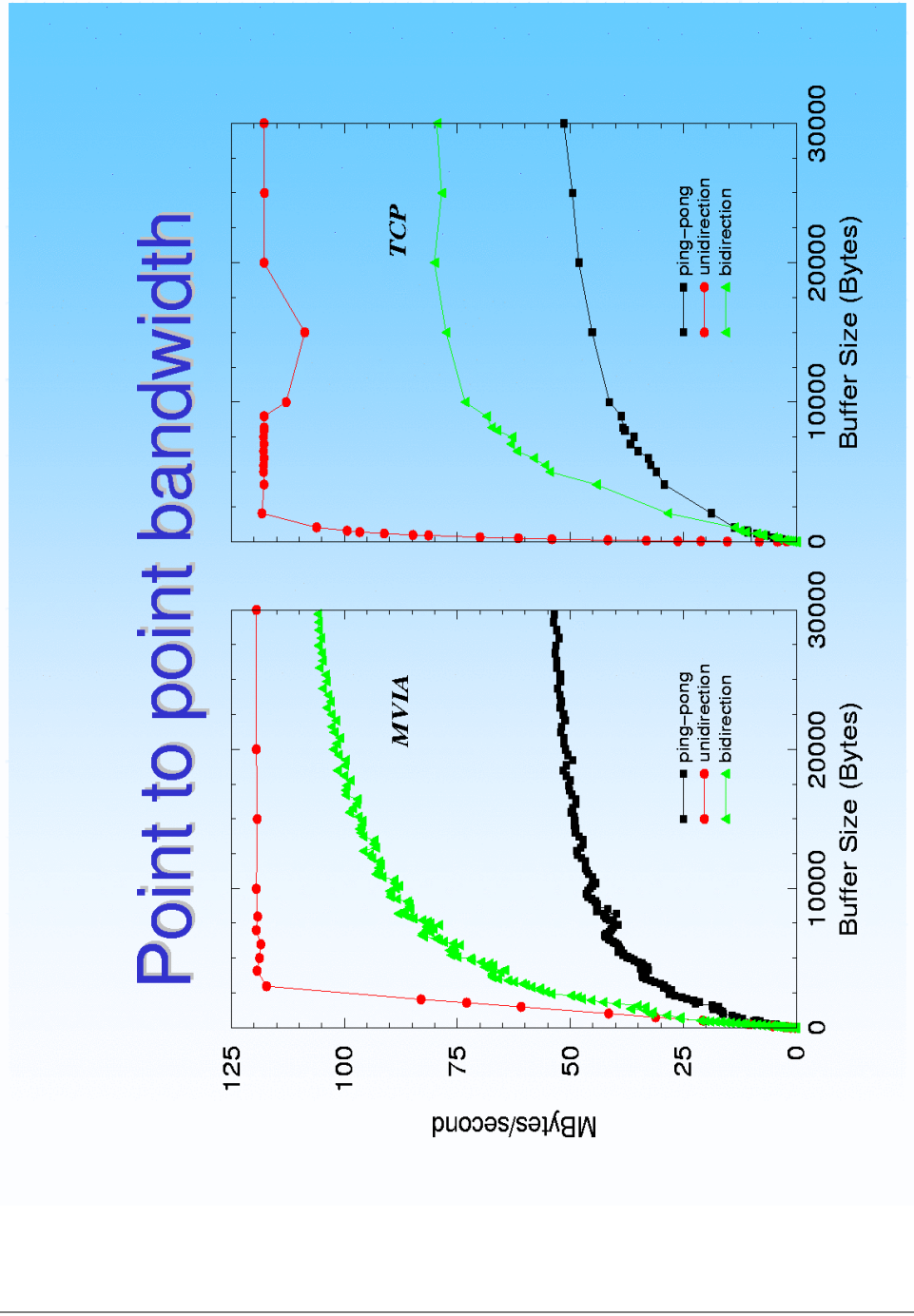
VIA Architecture

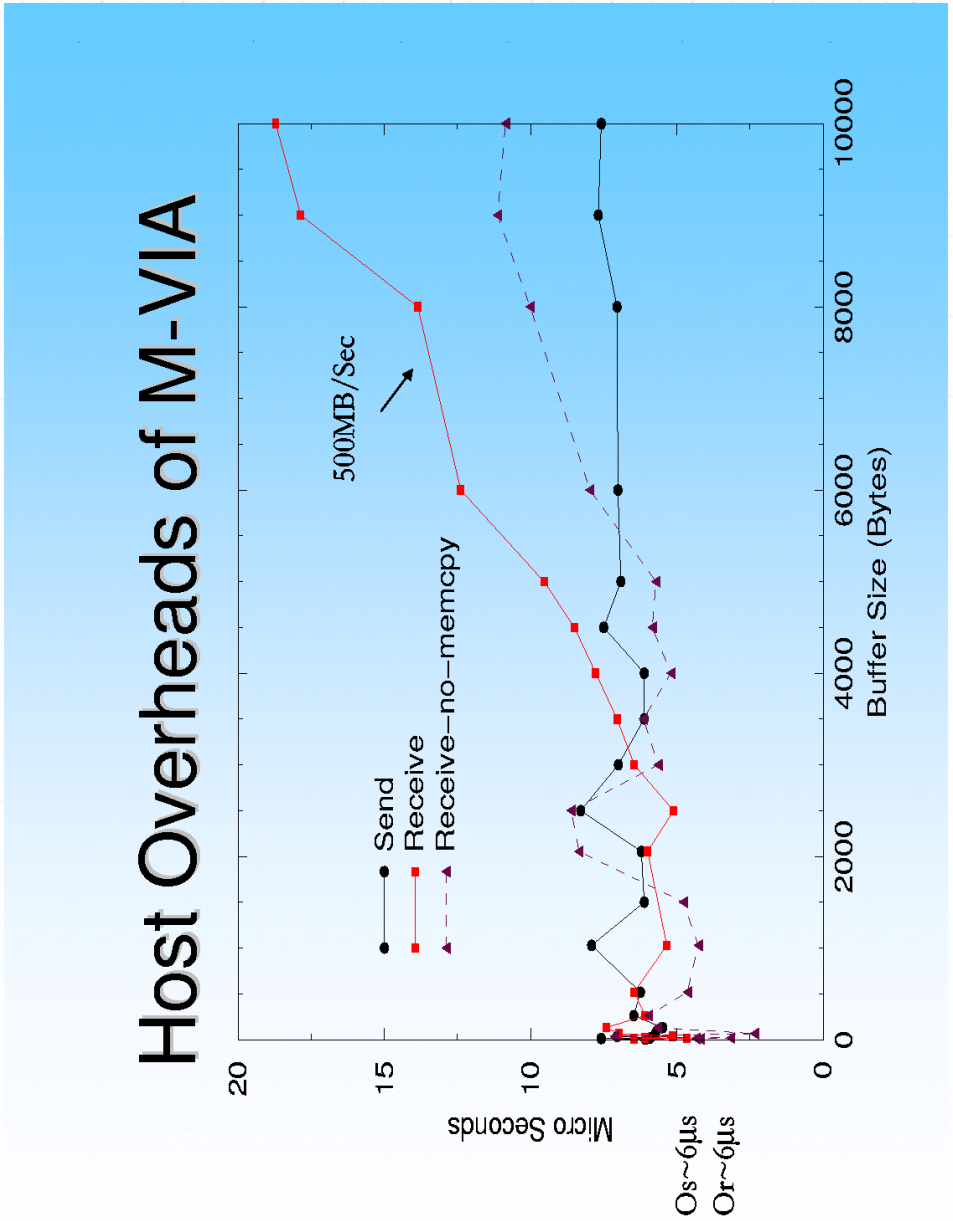


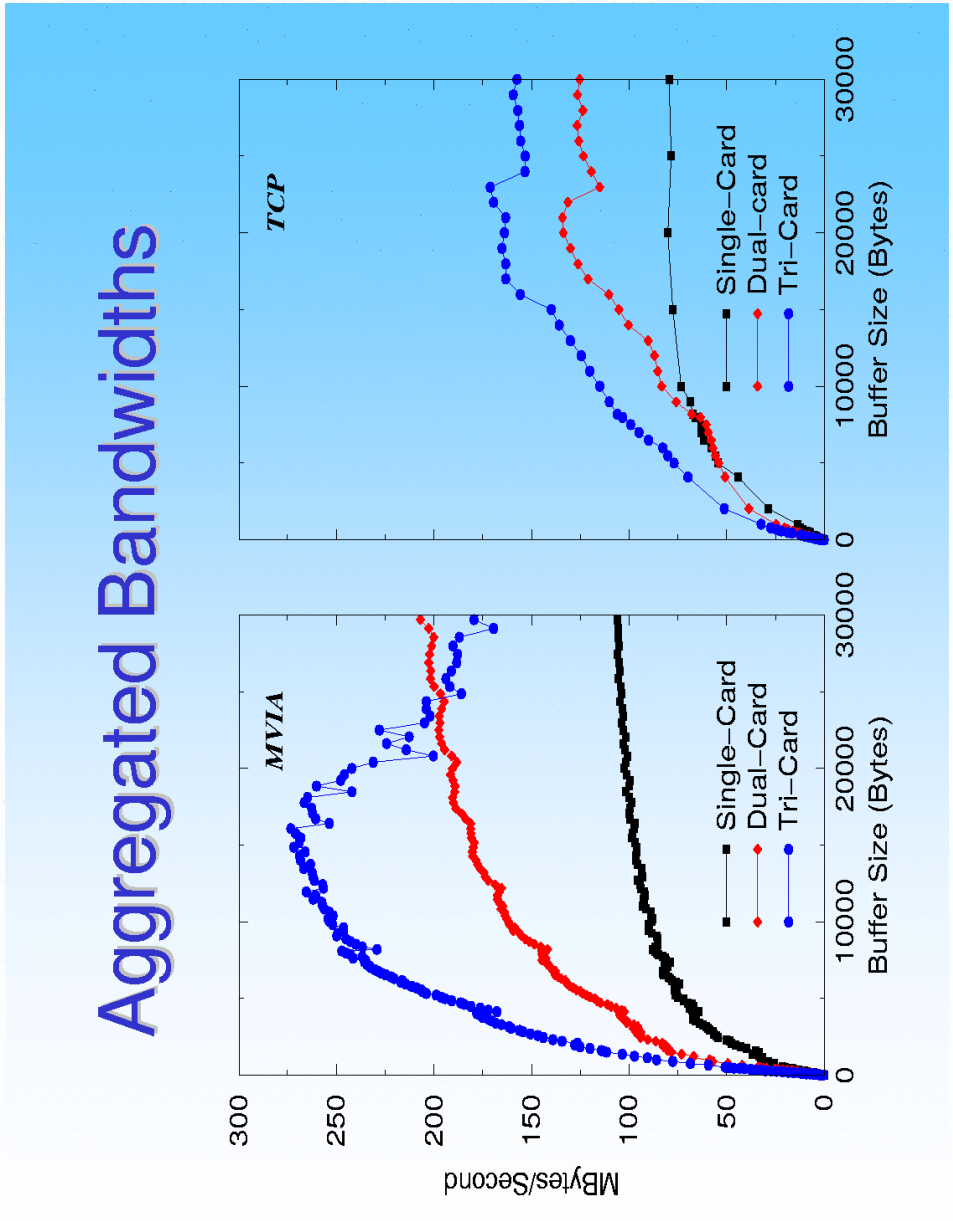
Integrating New Device with M-VIA

- Writing a Gigabit Ethernet driver for M-VIA (LBL) under Linux:
  - M-VIA version 1.2 with kernel version 2.4.
  - Intel e-1000 driver version 4.4.19 and version 5.x with effective MTU being 4096 Bytes.
  - M-VIA handles either reliable or unreliable data delivery and RDMA.
  - 5 driver macros, which handle DMA data gathering, initiation of data transmission, differentiation of VIA packet from IP packet and so on, need to be implemented for a new device.


Application to Application Latency
2003
RTT/2 (micro seconds)
Buffer Size (Bytes)
TCP
M-VIA
Myrinet
18.5µs
0
10
20
30
40
50
60
70
80
0
1000
2000
3000
4000


Point to point bandwidth
MBytes/second
TCP
MVIA
ping-pong
unidirection
bidirection
Buffer Size (Bytes)
0
25
50
75
100
125
0
10000
20000
30000


Host Overheads of M-VIA
Micro Seconds
Buffer Size (Bytes)
500MB/Sec
Send
Receive
Receive−no−memcpy
Os~6µs
Or~6µs
0
5
10
15
20
0
2000
4000
6000
8000
10000


Aggregated Bandwidths
MVIA
TCP
MBytes/Second
Buffer Size (Bytes)
Single−Card
Dual−Card
Tri−Card
0
50
100
150
200
250
300
0
10000
20000
30000

Message Passing Layer

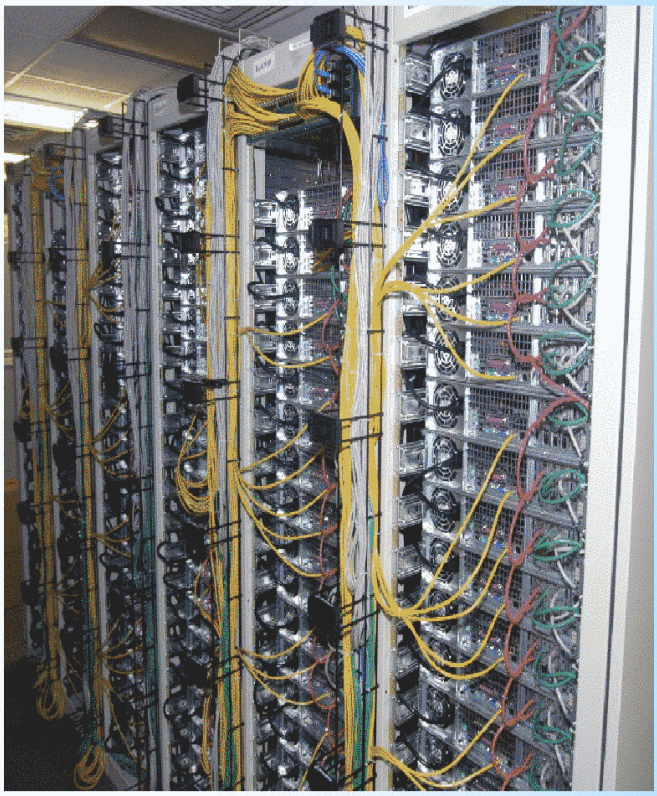
- Why not use MPI?
  - No MPI implementation on VIA with mesh connections
- QMP (QCD Message Passing)
  - Channel oriented
  - Synchronized remote memory write
  - Efficient global communications on mesh (torus) connections
  - Designed to be efficiently implemented on QCDOC custom machine

Current Status

- 256 node of 4x8x8 3-D torus – Dec. 2003
  - 2.66 GHz Xeon with 533 MHz front-side-bus
  - $2050/node (=$1600 unit + $450 gigE)
- 384 node of 4x6x8 3-D torus – Oct. 2004
  - 2.8 GHz Xeon with 800 MHz front-side-bus
  - $1860/node (gigE ~ $400)
- QMP implementation on M-VIA with Intel pro 1000 and 1000 MT.

	Bandwidth	Latency	Cost
Myrinet/GM (2004)	260 MB/s (*480MB/s*)	7.5 μs	300 + 900
GigaE/VIA (2003)	440 MB/s	18.5 μs	75 x 6
GigaE/VIA (2004)	450 MB/s	12.5 μs	70 x 6

red=X, green=Y, grey=Z



Future Directions

- Kernel level global operations
  - Push intermediate global operations (e.g. global sum) into kernel space (avoid user space).
  - Preliminary coding showed effective latency improvement from 18.5 μs to 12.5 μs.
  - Nearer cards – faster ASIC and lower latency
- Remove extra VIA receive side copy
  - Allow ethernet receive buffers to hold data until receiving host is ready (posts receive); currently data is copied to temporary user space buffer to prevent VIA from treating the lack of a receive buffer as an error

Cache trends

Commodity Clusters allow us to take advantage of the latest developments in processor design, memory sub-systems, and interconnect technology

- CPU's accelerate at ~ 60% / year (Moore's Law)
- Memory speed generally advances less rapidly and with fewer discrete steps, ~ 40% / year

=>Performance ratio of in-cache to out-of-cache is growing

- Implications:

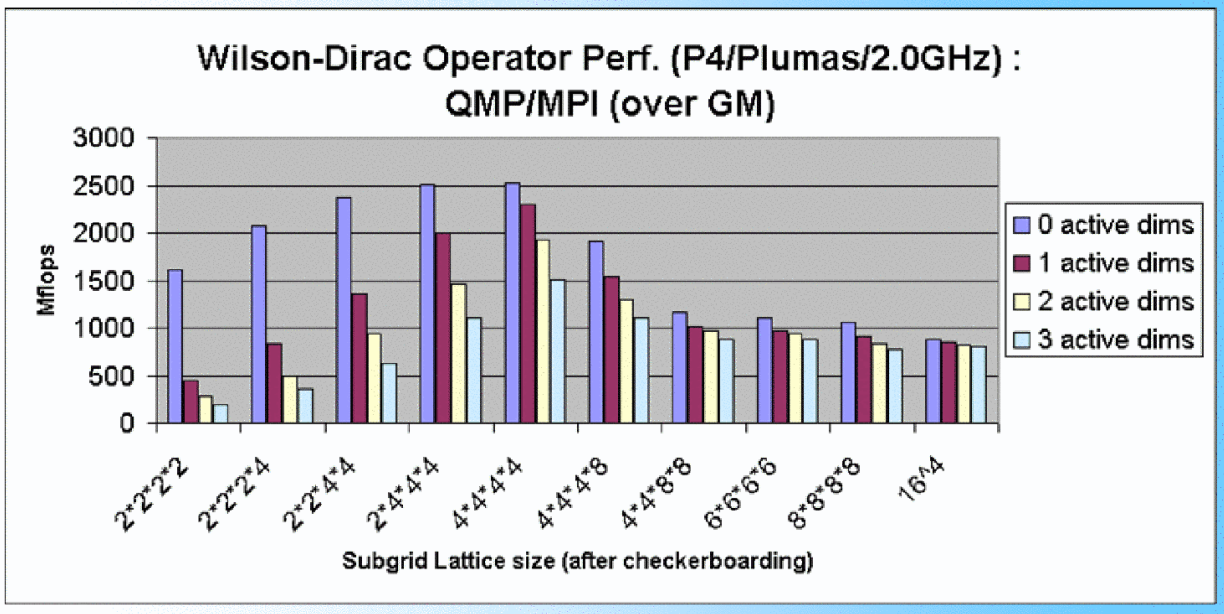
  Want to run as many applications in cache as possible (2x - 4x gain)

  => a large cluster used for single application

  => very high message rates (> 10 kHz !)

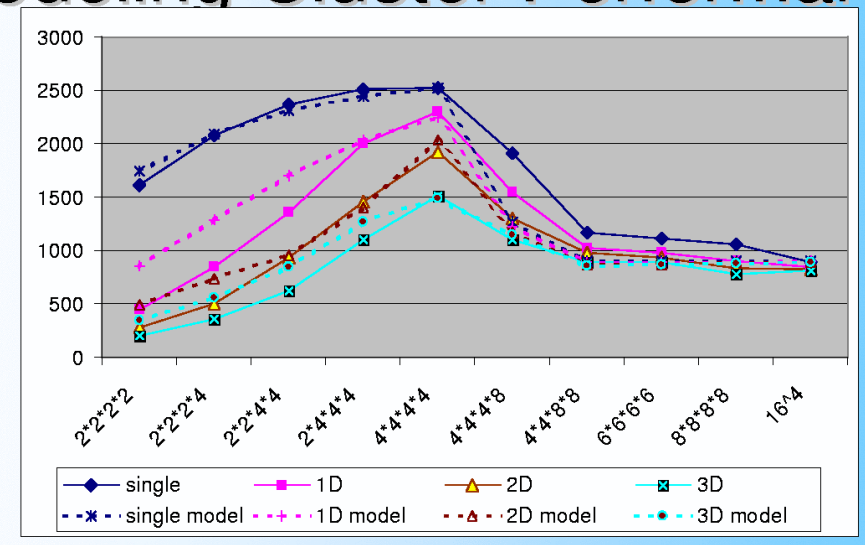
- Interconnects track external bus speeds, and server class motherboards will support processor evolutions for the next 2-3 years (multiple PCI-X busses)

Studying Cache Performance



Communications limited ← → Memory bandwidth limited

Modeling Cluster Performance



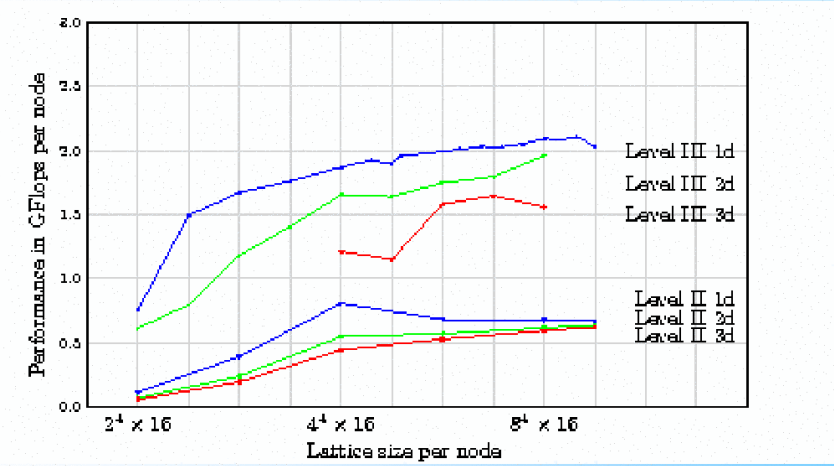
- Model includes CPU in- and out-of-cache single node performance, PCI and link bandwidth, latency, etc.
- Moderately simple model predicts cluster performance pretty well.

New Algorithms

- Optimizations in 4D fermions
  - Vectorizing of SU(3) indices awkward
  - Can strip-mine over sites – vectorize over 4 sites
  - Fortran indexing: Psi(4,Re/im, #Colors,#Spins,Sites/4)
  - Compiler vectors primitives (gcc, Intel): Vector a = b * c;
- 5D fermions:
  - Vectorize over 5D index
  - Fortran indexing: Psi(4,Re/im, #Colors,#Spins,Sites,N5/4)

Domain-Wall

- 256-node, 3d gigE mesh
  - P4, 2.66Ghz, 533Mhz FSB (memory bus) - 384 GFlops sustained
  - SciDAC DWF inverter (A. Pochinsky) – minimizes mem bus usage



- On current 384-node (800MHz FSB) – 1700MFlops/node - 650 GFlops sustained, $1.10/MFlop

Near Term Plans

- Processor/system improvements:
  - Multi-core / multi-processor (dual in 18 months)
  - Memory bus – 1066Mhz today, 1600Mhz in 18 mnths
  - Intel/AMD/IBM – cross-bar/NUMA memory design ??
  - PCI bus – more *"lanes"* - 4X: 1 GByte/s, 12X in 18 months (3GB/s)
- Network improvements:
  - Gigabit: Faster (cheaper) ASICs < 10 mu-secs latency
  - Infiniband: in 2005 new ASIC (single port) ~ $800/node
    - More *"lanes"* – 4X today (1Gbyte/s) , 12X in 18 months
- Consumer P4 series – 1 PCI-X bus + Infiniband
- Teraflop-scale systems in 2006 / 2007 – 512nodes

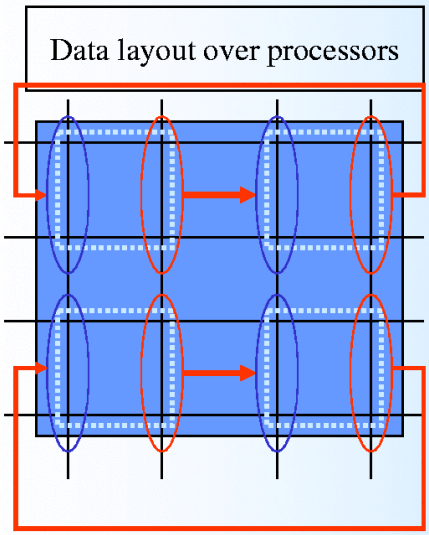
Reliability – Failure Modes

- 128-node Myrinet cluster (since 2002)
  - Power supply/myrinet card/disk ~ 1/month
- 256-node 3d gigE cluster (since 2003)
  - gigE ~ 1/6-months
  - Power supply/disk ~ 1/(2-3 months)
- 384-node 3d gigE cluster (since Oct. 04)
  - Tail of burn-in
  - gigE ~ 1/(2 months)
  - Power supply/disk ~ 1/(20 days)
- Software (PBS) bigger headache
  - Move to single machine world-view

- Software Infrastructure Goals: *Create a unified software environment that will enable the US lattice community to achieve very high efficiency on diverse multi-terascale hardware.*

TASKS:	LIBRARIES:
I. QCD Data Parallel API	➔ QDP
II. Optimize Message Passing	➔ QMP
III. Optimize QCD Linear Algebra	➔ QLA
IV. I/O, Data Files and Data Grid	➔ QIO
V. Opt. Physics Codes	➔CPS/MILC/Croma/etc.
VI. Execution Environment	➔ unify BNL/FNAL/JLab

Overlapping communications and computations



- C(x)=A(x) * shift(B, + mu):
  - Send face forward non-blocking to neighboring node.
  - Receive face into pre-allocated buffer.
  - Meanwhile do A*B on interior sites.
  - "Wait" on receive to perform A*B on the face.
- Lazy Evaluation (C style):

  Shift(tmp, B, + mu);

  Mult(C, A, tmp);

SciDAC Software Structure

Wilson Op, DWF Inv for P4; Wilson and Stag. Op for QCDOC

Level 3

Optimised Dirac Operators, Inverters

Level 2

Exists in C, C++, scalar and MPP using QMP

QDP (QCD Data Parallel)

Lattice Wide Operations, Data shifts

QIO

XML I/O LIME

QLA (QCD Linear Algebra)

Level 1

QMP (QCD Message Passing)

Exists, implemented in MPI, GM, gigE and QCDOC

QMP Simple Example

```
char buf[size];
QMP_msgmem_t mm;
QMP_msghandle_t mh;

mm = QMP_declare_msgmem(buf,size);
mh = QMP_declare_send_relative(mm,+x);
QMP_start(mh);
   // Do computations
QMP_wait(mh);
```

Multiple calls

Receiving node coordinates with the same steps except

```
mh = QMP_declare_receive_from(mm,-x);
```

Data Parallel QDP/C,C++ API

- Hides architecture and layout
- Operates on lattice fields across sites
- Linear algebra tailored for QCD
- Shifts and permutation maps across sites
- Reductions
- Subsets
- Entry/exit – attach to existing codes

Data-parallel Operations

- *Unary and binary:*

-a; a-b; …

- *Unary functions:*

adj(a), cos(a), sin(a), …

- *Random numbers:*

// platform independent

random(a), gaussian(a)

- *Comparisons (booleans)*

a <= b, …

- *Broadcasts:*

a = 0, …

- *Reductions:*

sum(a), …

- Fields have various types (indices): ***Tensor Product***

Lattice: A(x), Color: $U^{ij}(x)$, Spin: $\Gamma_{\alpha\beta}$, $\psi^{i}_{\alpha}(x)$, $Q^{ij}_{\alpha\beta}(x)$

QDP Expressions

- **Can create expressions**

$$c^{i}_{\alpha}(x) = U^{ij}_{\mu}(x)\, b^{i}_{\alpha}(x+\mu) + 2\, d^{i}_{\alpha}(x) \qquad \forall x \in \text{Even}$$

- **QDP/C++ code**

```
multi1d<LatticeColorMatrix>  u(Nd);
LatticeDiracFermion  b, c, d;
int mu;
c[even] = u[mu] * shift(b,mu) + 2 * d;
```

- **PETE: Portable Expression Template Engine**
  - Temporaries eliminated, expressions optimised

```
LatticeFermion   psi, p, r;
Real   c, cp, a, d;       Subset s;
for(int k = 1; k <= MaxCG; ++k)
{
   //  c  =  | r[k-1] |**2
   c = cp;

   //  a[k] := | r[k-1] |**2 / <M p[k], Mp[k] >
   //  Mp = M(u) * p
   M(mp, p, PLUS);  // Dslash

   //  d = | mp | ** 2
   d = norm2(mp, s);

   a = c / d;

   //  Psi[k] += a[k] p[k]
   psi[s] += a * p;

   //  r[k] -= a[k] M^dag.M.p[k] ;
   M(mmp, mp, MINUS);
   r[s] -= a * mmp;

   cp = norm2(r, s);
   if ( cp  <=  rsd_sq ) return;

   //  b[k+1] := |r[k]|**2 / |r[k-1]|**2
   b = cp / c;

   //  p[k+1] := r[k] + b[k+1] p[k]
   p[s] = r + b*p;
}
```

Norm squares

VAXPY operations

Performance Test Case - Wilson Conjugate Gradient

- In C++ significant room for perf. degradation
- Performance limitations in Lin. Alg. Ops (VAXPY) and norms
- Optimization:
  - Funcs return container holding function type and operands
  - At "=", replace expression with optimized code by template *specialization*
- Performance:
  - QDP overhead ~ 1% peak
  - Wilson: QCDOC 283Mflops/node @350 MHz, 4^4/node

Compute Density and Power

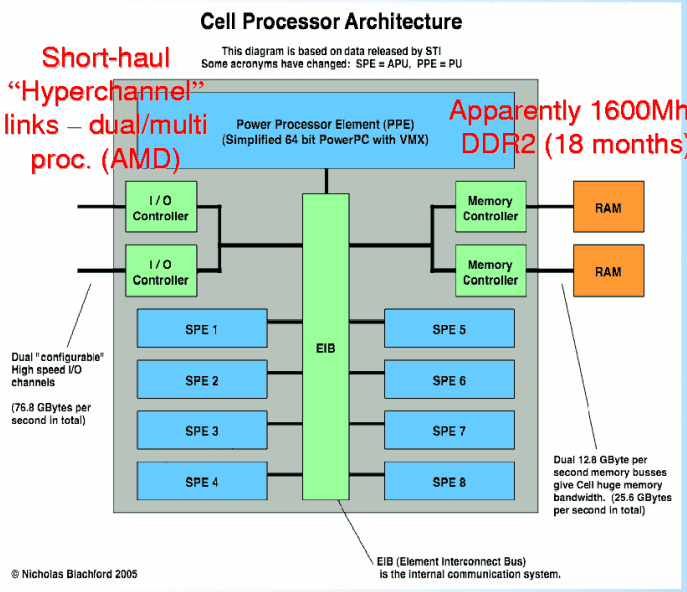
Comparison taken from talk by Dan Tanqueray (Cray)

	Cray X1	NEC SX-6	IBM p690 & SP switch	GigE - 384 node	QCDOC 420Mhz
Peak GFlops/proc	12.8	8.0	5.2	11.2	0.84
# Proc = 800 GFlops PEAK	64	100	156	72	1024
# Proc = 800 GFlops SUSTAINED	256	200	624	384	2048
Cabinet	4	26	20	16	2
Floor Space	128 ft^2	338 ft^2	256 ft^2	76 ft^2	24 ft^2
Compute Density	6 GF/ft^2	1 GF/ft^2	2.3 GF/ft^2	10 GF/ft^2	33 GF/ft^2
Compute Efficiency	3 GF/kw	4 GF/kw	1.3 GF/kw	11 GF/kw	48 GF/kw
Price-perf $/Mflop	???	???	???	1.10	1.10

- JLab: 384-node, 3 year life. A/C+elec. cost 15% machine price
- Only 5%/yr cost in elec. for cluster compared to 1%/yr QCDOC
- Power not a concern

18 Month Time frame - Cells

- Latest buzz – IBM "Cells" - ***Sony Playstation 3***
  - Super-Scalar processor
  - Vector co-processors - private memory – no cache coherency
  - Vector units initiated by loads issued from scalar processor
- Best Guess – only ***ever*** meant for Games with Virtual Reality



- I/O links very fast
- Bottleneck is memory in Cell and Clusters
- Same memory as clusters in 18 mth ~ 50% sustained bandwidth
- Double prec. 2.5 GFlop/s cluster nodes in 18 mths
- Optimistic: 10 GFlop/s Cell
- Cell needs 4 Infiniband 12X cards, cluster needs 1
- At best, a ***free*** Cell computer beats market by 2X in price-perf.

What is driving the ***Market***

- Crystal-ball (Wall-Street Journal)
- Business:
  - High performance: speech recognition/translation, web services
  - Networking: video conferencing
- Home:
  - High performance: games, virtual reality
  - Networking: entertainment/video, virtual reality
- Communications (cell phones – common to both):
  - High performance: not required – use back-end to save power
  - Networking: video
- Implications for us?
  - Technology (fab plants) in place for big iron
  - Something resembling clusters around for a while

For More Information

- U.S. Lattice QCD Home Page:
  http://www.usqcd.org/
- The JLab Lattice Portal
  http://lqcd.jlab.org/
- High Performance Computing at JLab
  http://www.jlab.org/hpc/