QCDOC: A highly scalable architecture for lattice QCD simulations

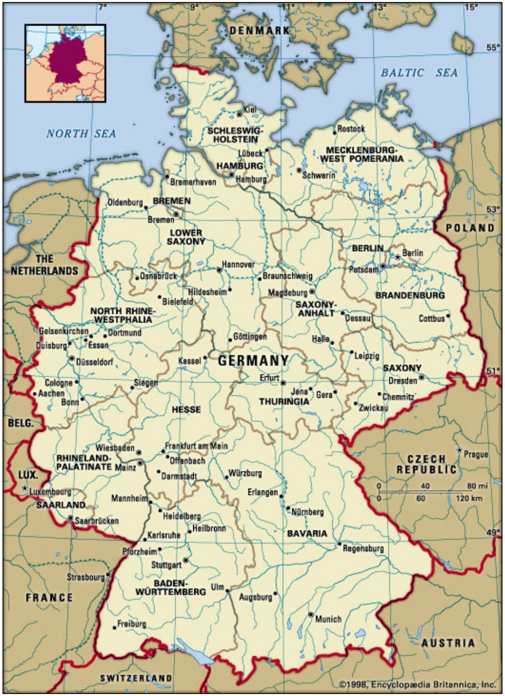
Tilo Wettig
University of Regensburg

Outline

1. Why custom-built machines?
2. QCDOC architecture
3. Performance figures
4. Summary and outlook

Thanks to IBM for $10k of conference sponsorship

KITP Santa Barbara, 30 March 2005



DENMARK
BALTIC SEA
NORTH SEA
POLAND
THE NETHERLANDS
BELG.
LUX.
FRANCE
SWITZERLAND
AUSTRIA
CZECH REPUBLIC
GERMANY
SCHLESWIG-HOLSTEIN
MECKLENBURG-WEST POMERANIA
HAMBURG
BREMEN
LOWER SAXONY
BERLIN
BRANDENBURG
SAXONY-ANHALT
NORTH RHINE-WESTPHALIA
SAXONY
THURINGIA
HESSE
RHINELAND-PALATINATE
SAARLAND
BADEN-WÜRTTEMBERG
BAVARIA
Kiel
Rostock
Lübeck
Hamburg
Schwerin
Bremerhaven
Oldenburg
Bremen
Berlin
Potsdam
Hannover
Osnabrück
Braunschweig
Hildesheim
Magdeburg
Bielefeld
Dessau
Cottbus
Gelsenkirchen
Dortmund
Duisburg
Essen
Göttingen
Halle
Leipzig
Düsseldorf
Kassel
Dresden
Erfurt
Cologne
Siegen
Jena
Gera
Chemnitz
Aachen
Bonn
Zwickau
Wiesbaden
Frankfurt am Main
Offenbach
Mainz
Prague
Darmstadt
Würzburg
Luxembourg
Erlangen
Mannheim
Nürnberg
Heidelberg
Saarbrücken
Heilbronn
Karlsruhe
Regensburg
Pforzheim
Stuttgart
Strasbourg
Ulm
Augsburg
Munich
Freiburg
7°
9°
11°
13°
15°
55°
53°
51°
49°
0 40 80 mi
0 60 120 km
©1998, Encyclopædia Britannica, Inc.

1. Why custom-built machines?

- dynamical fermions are expensive:

  ECFA scaling estimate

  cost (in Flops) to generate a single independent gauge field configuration (dynamical Wilson fermions, unimproved)

$$\text{Cost} \approx 1.7 \times 10^7 \, V^{4.55/4} \left(\frac{1}{a}\right)^{7.25} \left(\frac{1}{m_{ps}}\right)^{2.7}$$

  ...and we want $V \to \infty$, $a \to 0$, $m \to 0$

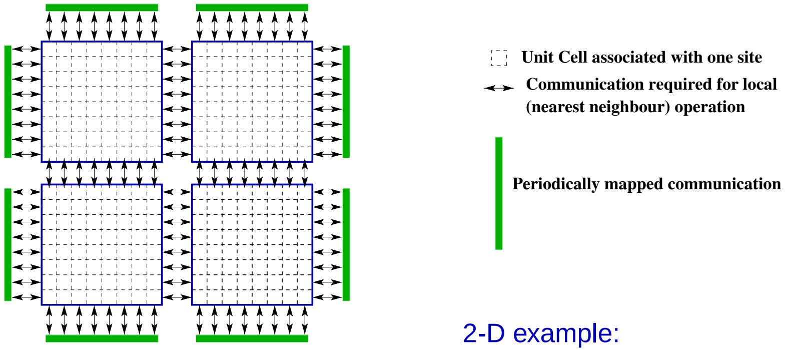
- staggered fermions somewhat cheaper
- chiral fermions (overlap, DWF) $\sim$100$\times$ more expensive

$\longrightarrow$ one of the “grand challenge problems” in HPC

Lines of attack

- better analytical understanding of the various limits
  - improvement program ($a \longrightarrow 0$)
  - effective chiral theories ($m \longrightarrow 0$)
  - finite-V calculations; ε-regime of QCD ($V \longrightarrow \infty$)
- better algorithms
- bigger and better (super-) computers

lattice QCD needs massively parallel machines (global volume distributed over many processors)


Unit Cell associated with one site
Communication required for local (nearest neighbour) operation
Periodically mapped communication

2-D example:
16×16 global volume
8×8 local volume
2×2 processor grid

- main numerical problem: inversion of fermion (Dirac) matrix, typically done using conjugate gradient algorithm
  $\longrightarrow$ need efficient matrix–vector routines and global sums

- on a parallel machine, communication between processors slows down the calculation; two main factors:
  1. communication bandwidth
  2. communication latency

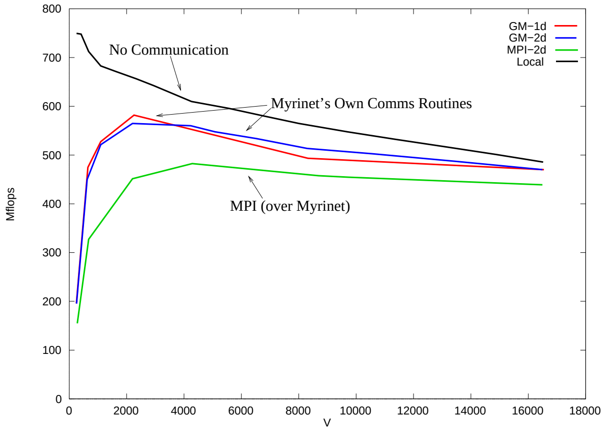
- for small local volumes (“hard scaling”):
  - surface-to-volume ratio large $\rightarrow$ high commun./comput. ratio
  - latency is dominating factor

- ideally, communication can be overlapped with computation: communication latency hiding

- measure of success: sustained vs. peak performance

- Scalability: keep physical problem size fixed and increase number of processing nodes — how does sustained performance scale? (It typically goes down drastically.)

Typical scaling behavior of clusters (measured by P. Boyle)



$\longleftarrow$ $\#$ of processors $\propto 1/V_{\text{local}}$

2-d grid has twice the bandwidth, but performance roughly the same
$\longrightarrow$ latency dominated

- PC clusters:
  - see Robert Edwards’ talk yesterday
  - cost-effective, easy to build
  - useful for wide range of applications and for experimentation
  - don’t scale well to very large numbers ($>$1000 or so) of processors

- commercial supercomputers:
  - very expensive
  - aimed at general-purpose market
  - typically switched clusters of SMP’s
    if not, networks are often underpowered or not scalable (crossbar)
    $\longrightarrow$ limited scalability
  - exceptions: BlueGene/L, Cray XT3, others?

- special-purpose machines can exploit the problem characteristics in hardware (extreme example: Grape)

  for lattice QCD:
  - predictable memory access $\rightarrow$ prefetch
  - mainly nearest-neighbor communications
  - communications predictable and repetitive

  $\longrightarrow$ better scalability

The QCDOC project

- successor to QCDSP (and earlier Columbia machines)
- collaboration of IBM Research (Yorktown Heights), Columbia University, RIKEN-BNL Research Center, UKQCD, SciDAC
- design based on an application-specific integrated circuit (ASIC)
- optimized for communications requirements of lattice QCD
- scalable to several 10,000 nodes with sustained performance of $\sim$50% even on very small local volumes ($V_{\text{local}} = 2^4$)
- best price/performance ratio in the industry
- low power consumption and heat dissipation
- standard software environment

another special-purpose lattice QCD machine (not discussed here):
apeNEXT by INFN/DESY/Paris-Sud
(successor to APE1, APE100, APEmille)

The QCDOC design team

Columbia: Norman Christ, Calin Cristian, Zhihua Dong, Changhoan Kim, Xiaodong Liao, Goufeng Liu, Bob Mawhinney, Azusa Yamaguchi

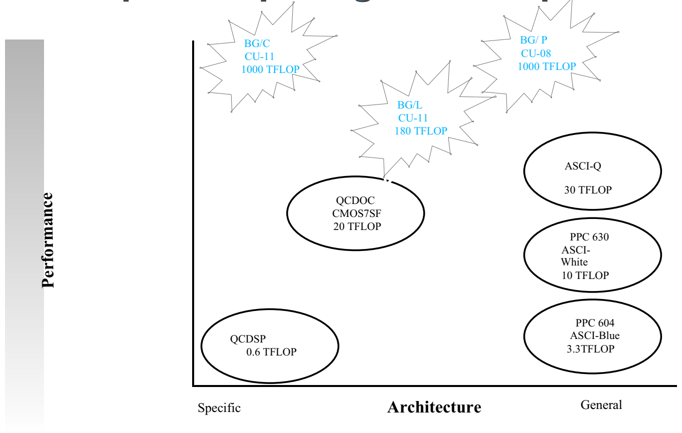
IBM: Dong Chen, Alan Gara, Minhua Liu, Ben Nathanson

RIKEN-BNL: Shigemi Ohta (KEK), Tilo Wettig (Yale $\rightarrow$ Regensburg)

UKQCD: Peter Boyle, Mike Clark, Bálint Joó

SciDAC: Chulwoo Jung, Kostya Petrov

Supercomputing Landscape



from http://www.research.ibm.com/bluegene

Current hardware status

- 14,720 nodes UKQCD machine at Edinburgh: running/testing
- 13,308 nodes RBRC machine at BNL: running/testing
- 14,140 nodes DOE machine at BNL: running/being assembled
- 2,432 nodes Columbia machine: running/testing
- 448 nodes Regensburg machine: just arrived

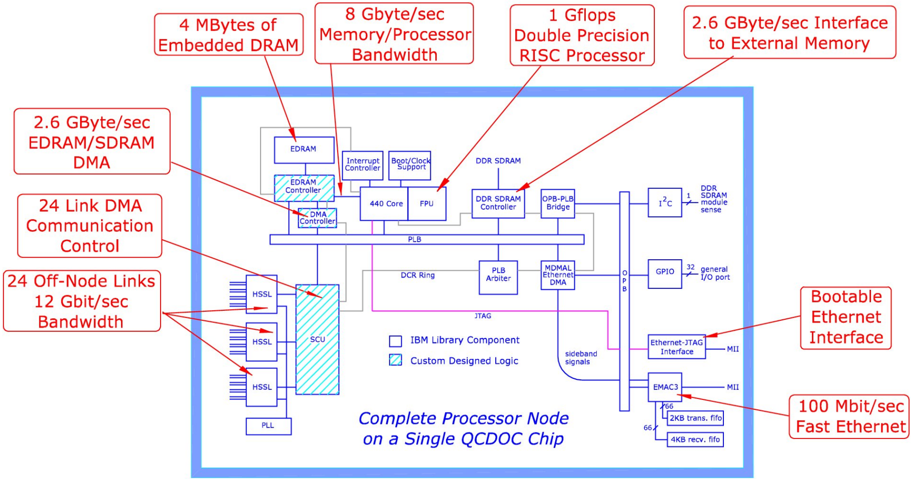
machines will be run 24x7 for five years
work horse for lattice QCD in the US and the UK

2. Architecture of the QCDOC supercomputer

- MIMD machine with distributed memory (in SPMD mode)
- system-on-a-chip design (QCDOC = QCD on a chip)
- QCDOC ASIC combines existing IBM components and QCD-specific, custom-designed logic:
  - 500 MHz (nominal) PowerPC 440 core with 64-bit, 1 GFlop/s FPU
  - 4 MB on-chip memory (embedded DRAM), accessed through custom-designed prefetching eDRAM controller (PEC)
  - nearest-neighbor serial communications unit (SCU) with aggregate bandwidth of 12 Gbit/s
- separate networks for physics communications and auxiliary tasks
- price/performance ratio $<$ 1 US-$ per sustained MFlop/s
- very low power consumption, no serious cooling issues

All this on $\sim$ (12 mm)^2 of silicon, consuming $\sim$ 5 W

The QCDOC ASIC



(0.18μ process, using low-alpha lead)

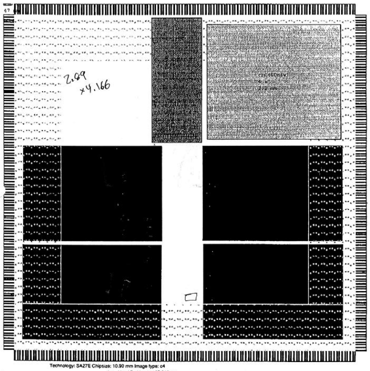
ASIC Floorplan

NOV 29 '99 20:57 FR TO 88623684 P.02

4M Flavour

32k x 288 x 4 = 36,864,000 bits

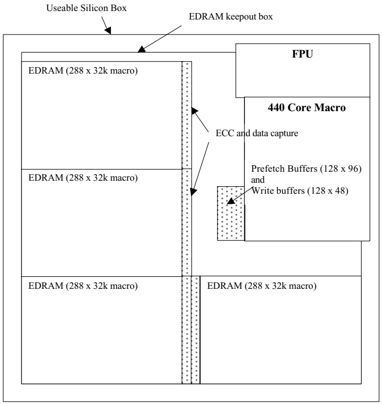
N x M x 4



Technology: SA27E Chipsize: 10.90 mm Image type: c4

Area Used By Macros: 67.916mm,

as a percentage of the total usable area: 64.768%

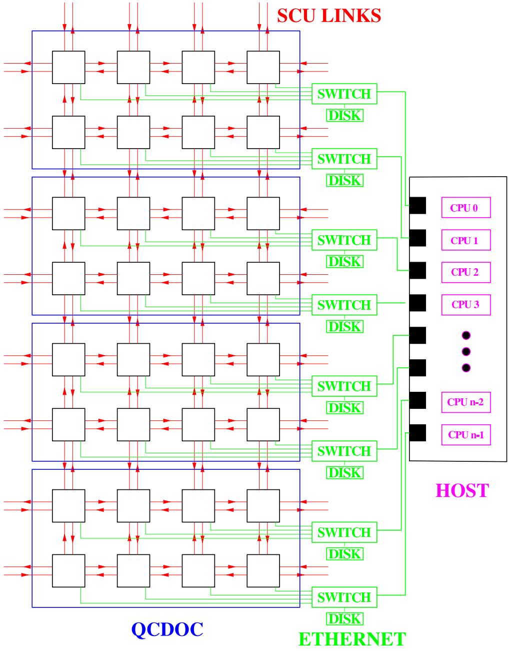


440 PowerPC processor core

- 32-bit implementation of IBM Book E architecture
- out-of-order dual-issue, superscalar, pipelined (7 stages), . . .
- 32kB instruction and 32kB data caches (64-way associative, partitionable, lockable)
- hardware memory protection through TLB
- 3 PLB master interfaces: instruction read, data read, data write
- complete control through JTAG interface
- connected to 64-bit, 1 GFlop/s IEEE floating point unit

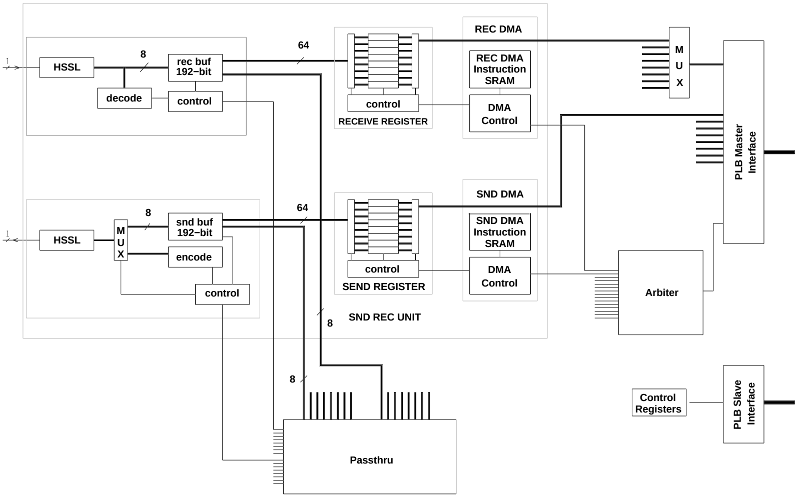
QCDOC networks

- fast “physics” network
  - 6-dimensional torus with nearest-neighbor connections (allows for machine partitioning in software)
  - LVDS serial links using IBM HSSL transceivers with 2×500 Mbit/s bandwidth per link (total bandwidth per node 12 Gbit/s)
  - custom-designed Serial Communications Unit (SCU)
    - runs all links concurrently
    - direct memory access
    - packets have 8-bit header and 64-bit data
    - error detection and correction (ECC on header, parity on data, hardware checksum counters)
    - low-latency passthrough mode for global operations
    - reconfigurable partition interrupts and barriers
  - communications performance $\sim$ memory performance
- global tree network for three independent interrupts
- Ethernet network (switched) for booting, I/O, control

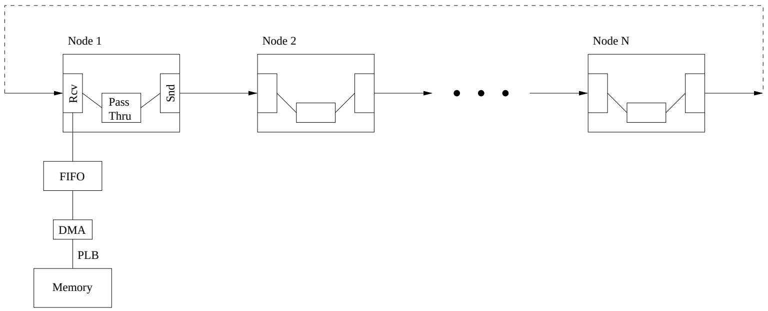


SCU LINKS
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
SWITCH
DISK
CPU 0
CPU 1
CPU 2
CPU 3
CPU n-2
CPU n-1
HOST
QCDOC
ETHERNET

Serial Communications Unit (SCU)

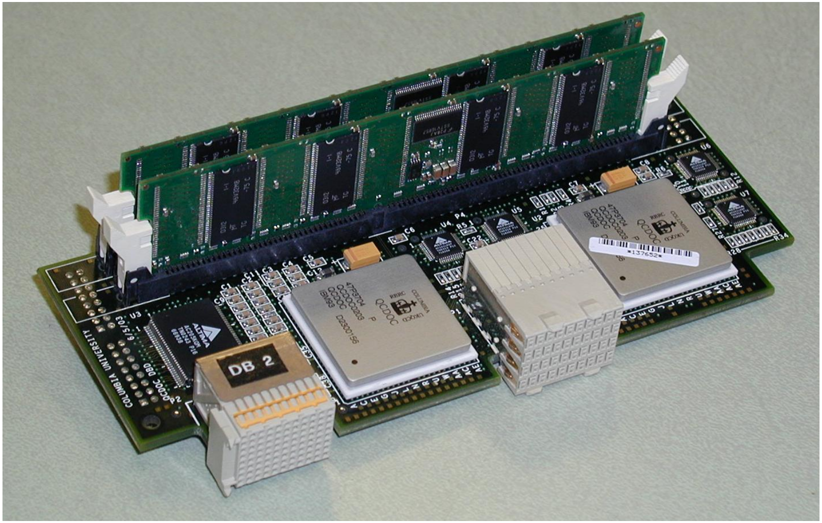


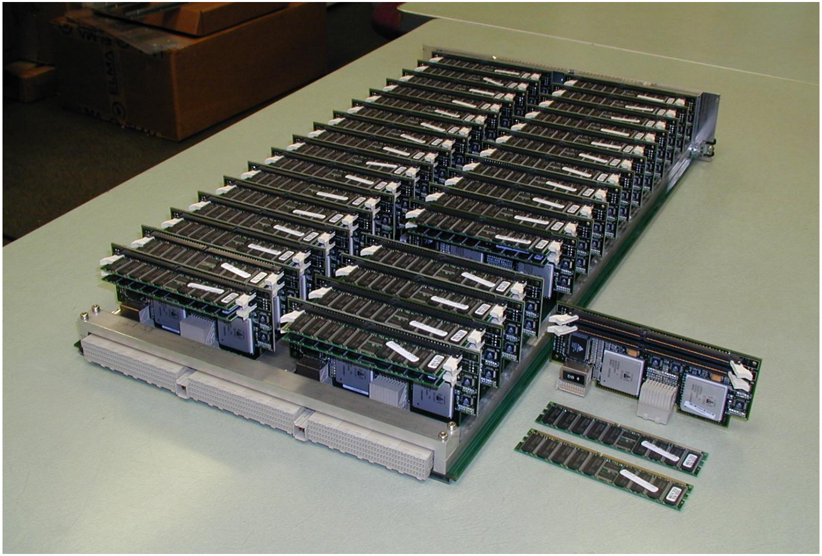
Communications latency hiding for global sums (1-d example)

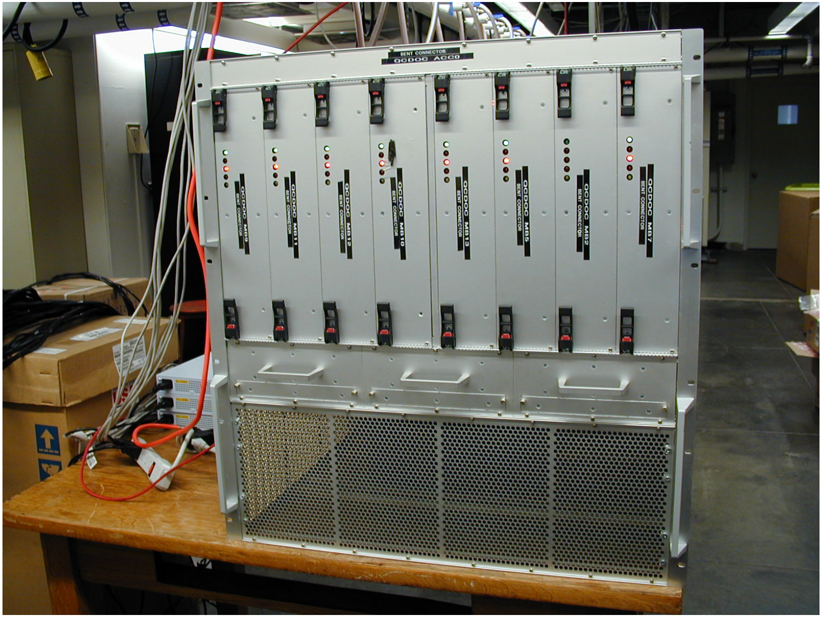


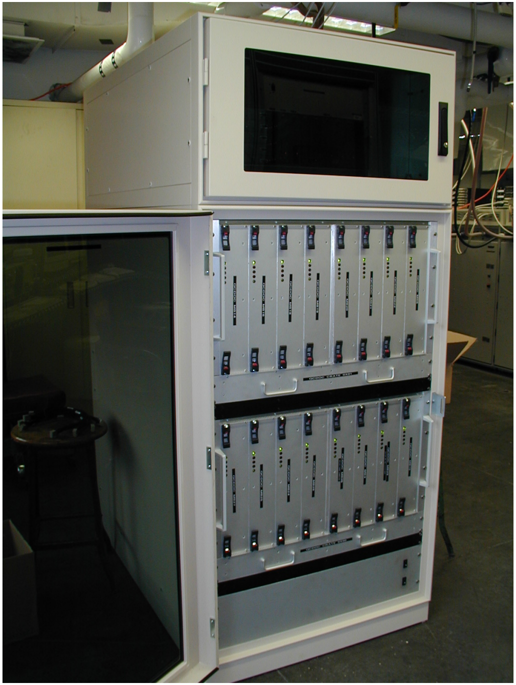
- receive unit gets number from neighbor in $-$ direction
  $\longrightarrow$ puts it in FIFO and simultaneously passes it on to neighbor in $+$ direction
- FIFO is flushed when PLB access is available
- after $N-1$ shifts, all numbers from this dimension are in memory
- add done by CPU

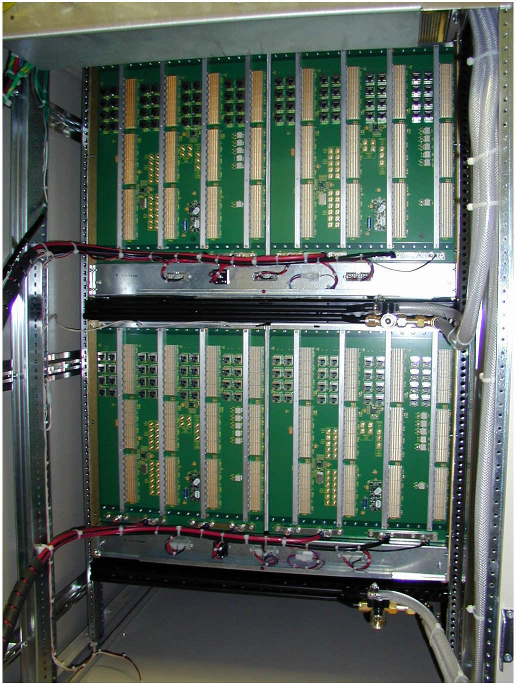
$\longrightarrow$ software and bus latency only paid once instead of N times
broadcast of final result not necessary

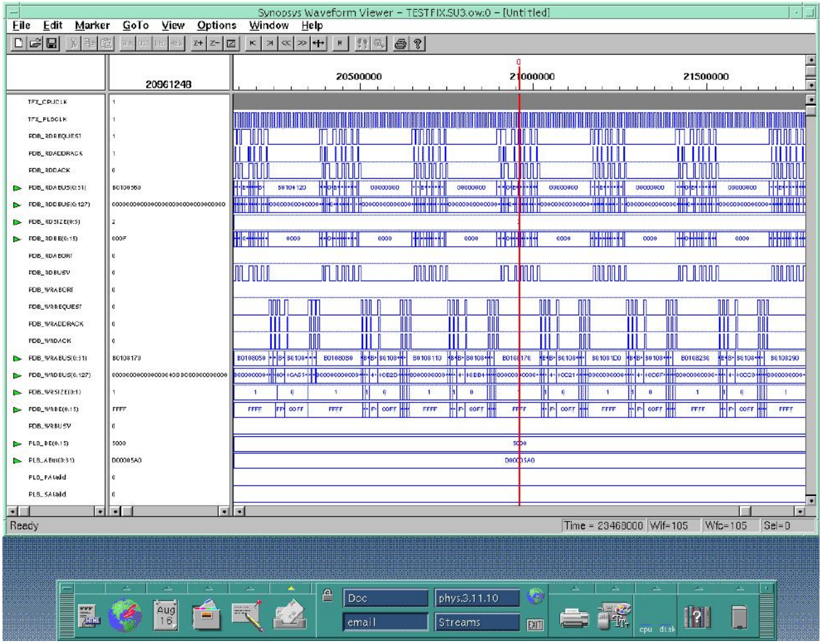

DB 2
COLUMBIA UNIVERSITY
QCDOC DBD 6/5/03
QCDOC
QCDOC
COLUMBIA
IBM93
ALTERA




BENT CONNECTOR
QCDOC ACC0
QCDOC MB9
BENT CONNECTOR
QCDOC MB11
BENT CONNECTOR
QCDOC MB12
BENT CONNECTOR
QCDOC MB10
BENT CONNECTOR
QCDOC MB13
BENT CONNECTOR
QCDOC MB5
BENT CONNECTOR
QCDOC MB2
BENT CONNECTOR
QCDOC MB7
BENT CONNECTOR


DC/DC CRATE 0X01

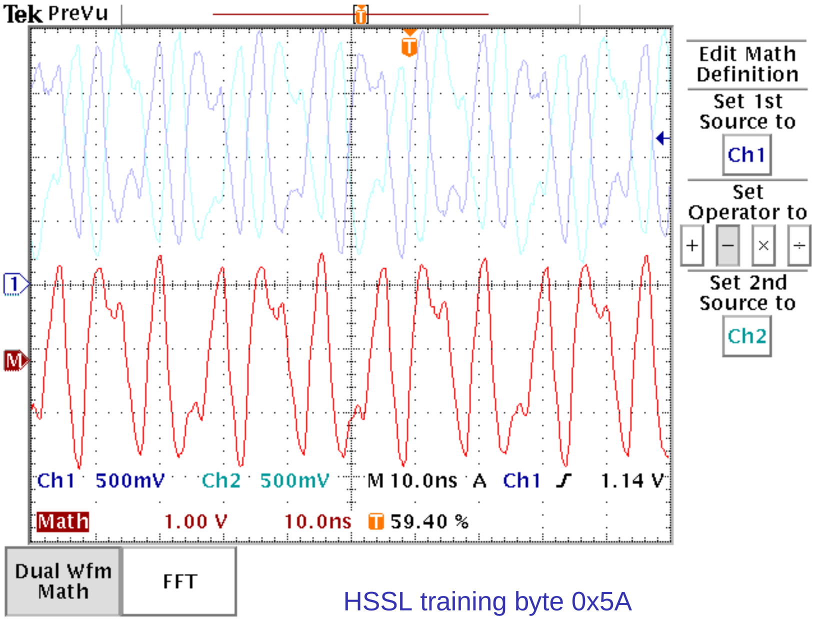




```
20961248

TFX_CPUCLK            1
TFX_PLBCLK            1
PDB_RDREQUEST         1
PDB_RDADDRACK         1
PDB_RDDACK            0
PDB_RDABUS(0:31)      80100560
PDB_RDDBUS(0:127)     00000000000000000000000000000000
PDB_RDSIZE(0:3)       2
PDB_RDBE(0:15)        000F
PDB_RDABORT           0
PDB_RDBUSY            0
PDB_WRABORT           0
PDB_WRREQUEST         0
PDB_WRADDRACK         0
PDB_WRDACK            0
PDB_WRABUS(0:31)      80108170
PDB_WRDBUS(0:127)     800000000000000040B8C8000000000
PDB_WRSIZE(0:1)       1
PDB_WRBE(0:15)        FFFF
PDB_WRBUSY            0
PLB_BE(0:15)          5000
PLB_ABus(0:31)        D00005A0
PLB_PAValid           0
PLB_SAValid           0
```

Ready   Time = 23468000   Wif=105   Wfc=105   Sel=0

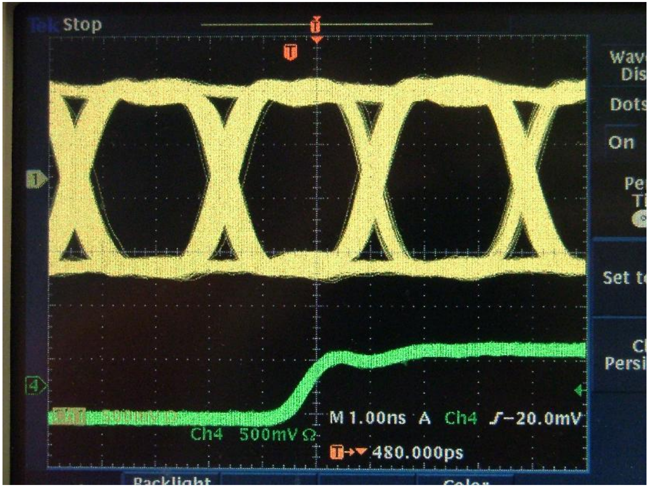


Ch1 500mV Ch2 500mV M 10.0ns A Ch1 ∫ 1.14 V

Math 1.00 V 10.0ns 59.40 %

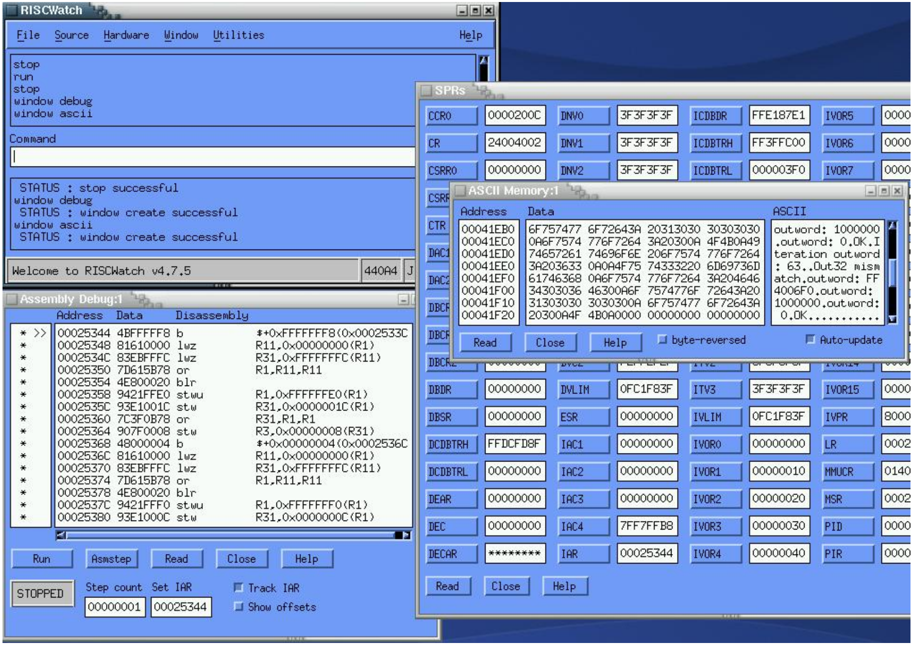
HSSL training byte 0x5A

Serial communications data eye



Software environment

- custom OS (QOS) written by Peter Boyle
  - qdaemon on host machine
    (csh-like user interface, program load, integrated diagnostics)
  - boot kernel download via Ethernet/JTAG to I/D cache in parallel
  - run kernel download via Ethernet
  - single application process $\rightarrow$ no scheduler-induced slowdown
  - user-space access to communications hardware $\rightarrow$ low latency
- standard compile chains: gcc/g++ and xlc/xlC
- QMP library for inter-node message passing (Chulwoo Jung)
  (SciDAC cross-platform library for lattice QCD)
- CPS, MILC, and Chroma/QDP++ lattice QCD codes ported
- Bagel automatic assembly generator (Peter Boyle)
- MPI port by BNL (Robert Bennett)
- SciDAC software: see Robert Edwards' talk yesterday
- RISCWatch for diagnostics



```
stop
run
stop
window debug
window ascii
```

Command

```
 STATUS : stop successful
window debug
 STATUS : window create successful
window ascii
 STATUS : window create successful
```

Welcome to RISCWatch v4.7.5 | 440A4

```
      Address  Data      Disassembly
* >>  00025344 4BFFFFF8 b           $+0xFFFFFFF8(0x0002533C
*     00025348 81610000 lwz         R11,0x00000000(R1)
*     0002534C 83EBFFFC lwz         R31,0xFFFFFFFC(R11)
*     00025350 7D615B78 or          R1,R11,R11
*     00025354 4E800020 blr
*     00025358 9421FFE0 stwu        R1,0xFFFFFFE0(R1)
*     0002535C 93E1001C stw         R31,0x0000001C(R1)
*     00025360 7C3F0B78 or          R31,R1,R1
*     00025364 907F0008 stw         R3,0x00000008(R31)
*     00025368 48000004 b           $+0x00000004(0x0002536C
*     0002536C 81610000 lwz         R11,0x00000000(R1)
*     00025370 83EBFFFC lwz         R31,0xFFFFFFFC(R11)
*     00025374 7D615B78 or          R1,R11,R11
*     00025378 4E800020 blr
*     0002537C 9421FFF0 stwu        R1,0xFFFFFFF0(R1)
*     00025380 93E1000C stw         R31,0x0000000C(R1)
```

STOPPED

Step count: 00000001 Set IAR: 00025344

CCR0	0000200C	DNV0	3F3F3F3F	ICDBDR	FFE187E1	IVOR5	0000
CR	24004002	DNV1	3F3F3F3F	ICDBTRH	FF3FFC00	IVOR6	0000
CSRR0	00000000	DNV2	3F3F3F3F	ICDBTRL	000003F0	IVOR7	0000
CSRR							
CTR							
DAC1							
DAC2							
DBCR							
DBCR							
DBCR2							
DBDR	00000000	DVLIM	0FC1F83F	ITV3	3F3F3F3F	IVOR15	0000
DBSR	00000000	ESR	00000000	IVLIM	0FC1F83F	IVPR	8000
DCDBTRH	FFDCFD8F	IAC1	00000000	IVOR0	00000000	LR	0002
DCDBTRL	00000000	IAC2	00000000	IVOR1	00000010	MMUCR	0140
DEAR	00000000	IAC3	00000000	IVOR2	00000020	MSR	0002
DEC	00000000	IAC4	7FF7FFB8	IVOR3	00000030	PID	0000
DECAR	********	IAR	00025344	IVOR4	00000040	PIR	0000

```
Address   Data                                  ASCII
00041EB0  6F757477 6F72643A 20313030 30303030   outword: 1000000
00041EC0  0A6F7574 776F7264 3A20300A 4F4B0A49   .outword: 0.OK.I
00041ED0  74657261 74696F6E 206F7574 776F7264   teration outword
00041EE0  3A203633 0A0A4F75 74333220 6D69736D   : 63..Out32 mism
00041EF0  61746368 0A6F7574 776F7264 3A204646   atch.outword: FF
00041F00  34303036 46300A6F 7574776F 72643A20   4006F0.outword:
00041F10  31303030 3030300A 6F757477 6F72643A   1000000.outword:
00041F20  20300A4F 4B0A0000 00000000 00000000    0.OK...........
```

3. Performance figures

Three relevant areas:

1. Memory system performance
2. Network performance
3. Application code performance

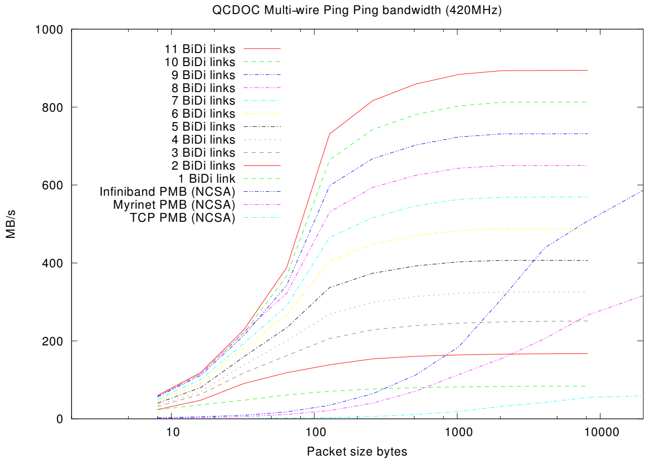
all of the following courtesy of Peter Boyle

Memory system performance

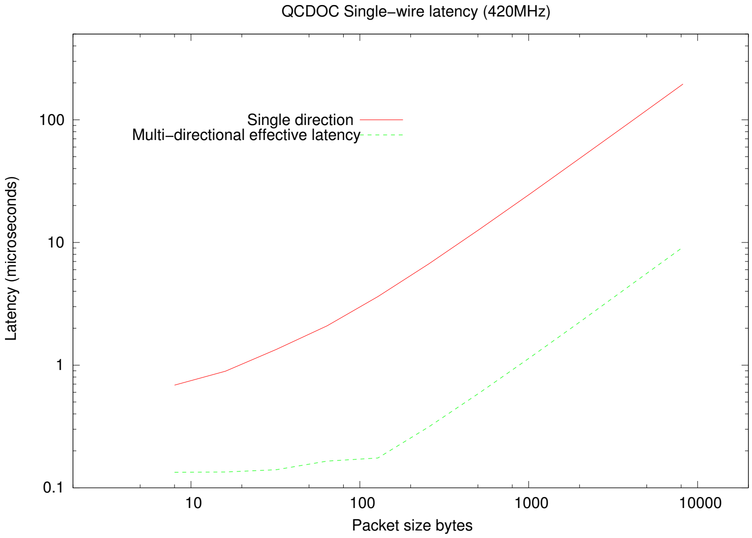
Streams benchmark (measures sustainable memory bandwidth)

Compiler/Options/Code	Comment	Memory	MB/s
xlc -O5 -qarch=440	vanilla source	Edram	747
gcc-3.4.1 -funroll-all-loops -fprefetch-loop-arrays -O6	vanilla source	Edram	747
gcc-3.4.1 -funroll-all-loops -fprefetch-loop-arrays -O6	__builtin_prefetch	Edram	1024
Assembly	auto-generated asm	Edram	1670
xlc -O5 -qarch=440	vanilla source	DDR	260
gcc-3.4.1 -funroll-all-loops -fprefetch-loop-arrays -O6	vanilla source	DDR	280
gcc-3.4.1 -funroll-all-loops -fprefetch-loop-arrays -O6	__builtin_prefetch	DDR	447
Assembly	auto-generated asm	DDR	606

Network performance



- multi-link bandwidth as good as CPU-memory bandwidth
- single-link ping pong obtains 50% of maximum bandwidth on 32-byte packets

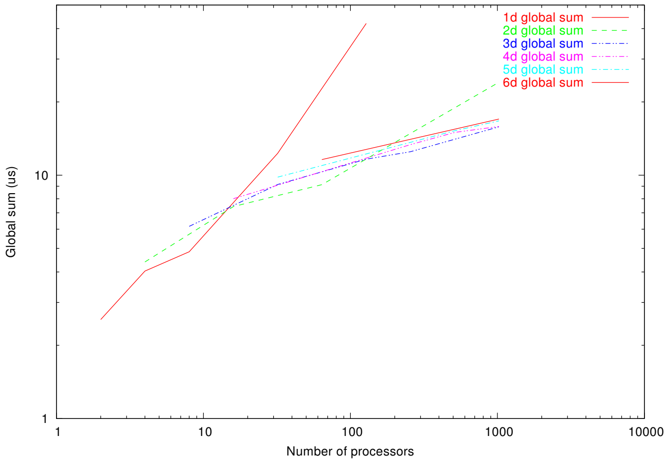

QCDOC Single−wire latency (420MHz)
Single direction
Multi−directional effective latency
Latency (microseconds)
100
10
1
0.1
10
100
1000
10000
Packet size bytes

Global reduction

- hardware-acceleration for “all-to-all” along an axis
- CPU performs arithmetic

$$\text{global sum} \rightarrow 300\text{ns} \times \frac{1}{2} D (N_{\text{proc}})^{1/D}$$

- 1024-node global sum in less than 16μs (12k nodes in 20-30μs)

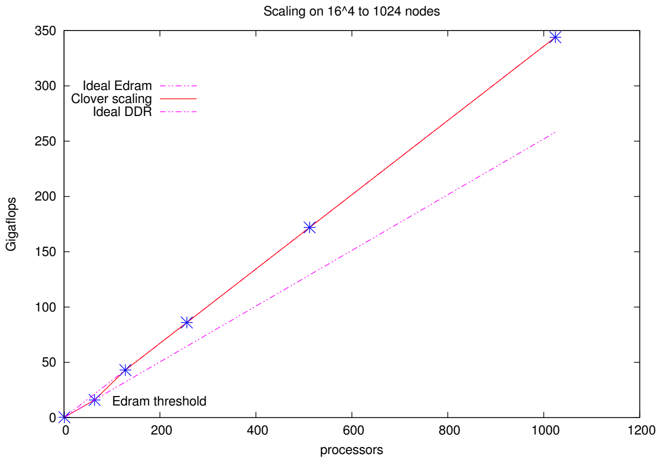


Application code performance

various discretizations, 4^4 local volume

Action	Nodes	Sparse matrix	CG performance
Wilson	512	44%	39%
Asqtad	128	42%	40%
DWF	512	46%	42%
Clover	512	54%	47%

Scalability



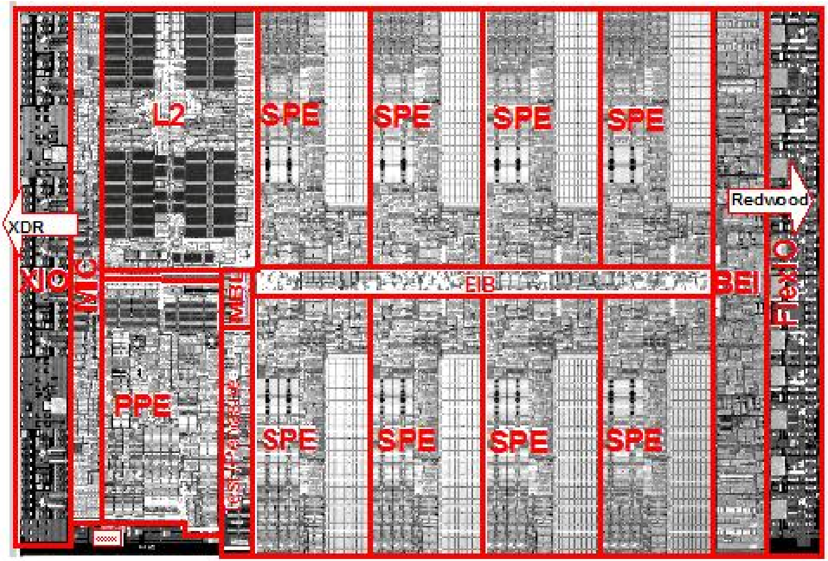
- 16^4 on 1024 nodes (equivalent to 32^4 on 16k nodes)
- expect 4 to 5 TFlop/s sustained on large machines

4. Summary and outlook

- QCDOC provides
  - exceptional scalability
    (due to network performance and hardware assist for global sums)
  - standard software environment
  - best value for the money:
    < 1 US-$ per sustained MFlop/s at 5 TFlop/s on a non-trivial problem
- low power and low cost allows large systems to be built
- hardware status (again)
  - current clock speed is 420 MHz (hope to improve this)
  - 14,720 nodes UKQCD machine at Edinburgh: running/testing
  - 13,308 nodes RBRC machine at BNL: running/testing
  - 14,140 nodes DOE machine at BNL: running/being assembled
  - 2,432 nodes Columbia machine: running/testing
  - 448 nodes Regensburg machine: just arrived

Future possibilities

- cannot beat IBM in general-purpose market (BG/P, BG/Q)
- clusters are catching up fast
- exploiting the special-purpose character can still give us an edge
- must aim at 10$\times$ in price/performance over IBM and/or clusters goal is \$0.01/MFlop/s or \$10 million/PFlop/s (sustained)
- currently exploring a number of possibilities
  - custom ASIC
  - commercial chip plus companion ASIC
  - standard protocols (Hypertransport, . . .)
  - . . .
- ASIC NRE costs exploding $\rightarrow$ high risk
- Cell looks interesting


L2
SPE
SPE
SPE
SPE
Redwood
XDR
XIO
MIC
MBL
EIB
BEI
FlexIO
PPE
SPE
SPE
SPE
SPE

- 300 engineers
- 0.09/0.065μ process
- 50$\sim$80 W at 4 GHz
- 1 (new) PowerPC CPU with 32 kB L1 caches (D/I)
- 8 FPU’s with 256 kB of private memory
- each FPU can do 4 FMADD’s per cycle
  $\rightarrow$ 256 GFlop/s at 4 GHz (single precision)
- double precision $\sim$10$\times$ slower
- 512 kB on-chip shared L2 cache
- 25 GB/s memory bandwidth (Rambus XDR)
- 76.8 GB/s (44.8 in, 32 out) I/O bandwidth (Rambus FlexIO)
- Can memory subsystem keep the FPU’s fed?
- Programming model?