QCD Performance on Blue Gene/L

Experiences with the Blue Gene/L in Jülich

Blue Gene at NIC/ZAM in Jülich

Overview:

- BGL System
- Compute Chip
- Double Hummer
- Network/
  MPI Issues
- Dirac Matrix
  Performance

IBM Blue Gene/L System Buildup

System
64 Racks, 64x32x32
65.536 chips

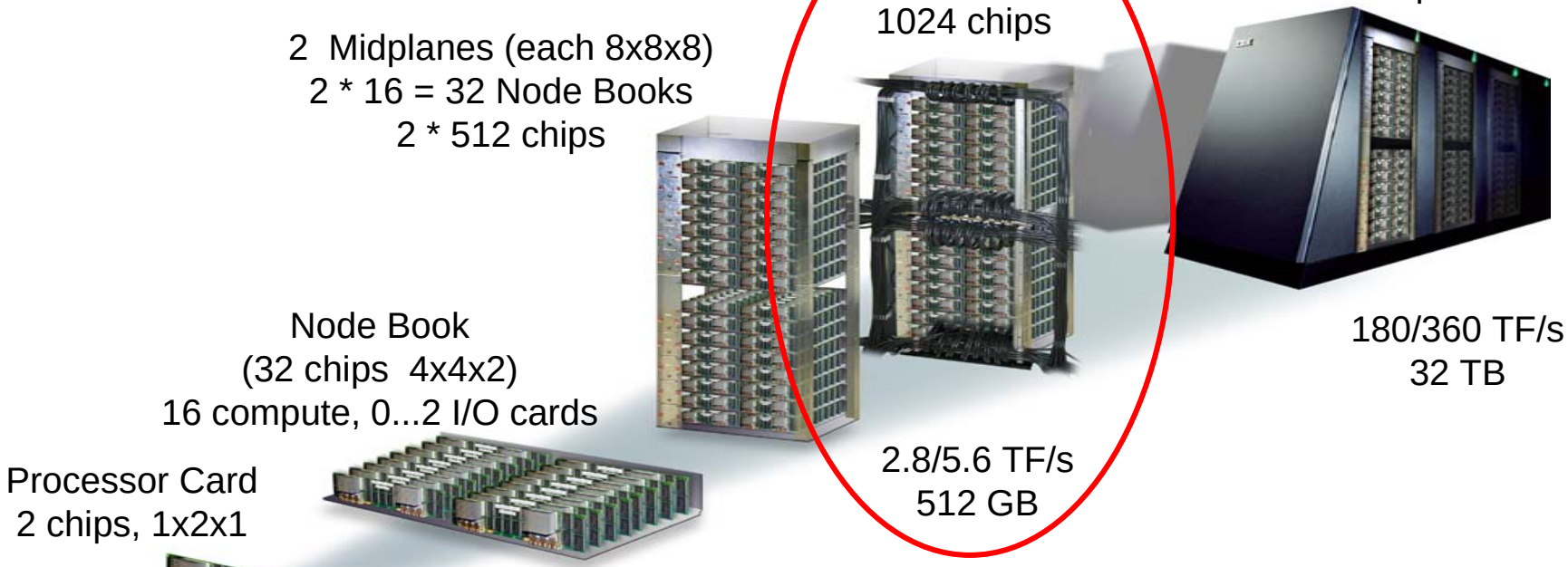
Rack
cabled 16x8x8
1024 chips

2 Midplanes (each 8x8x8)
2 * 16 = 32 Node Books
2 * 512 chips

Node Book
(32 chips 4x4x2)
16 compute, 0...2 I/O cards

Processor Card
2 chips, 1x2x1

Node
Dual Processor



180/360 TF/s
32 TB

2.8/5.6 TF/s
512 GB

90/180 GF/s
16 GB

5.6/11.2 GF/s
1.0 GB

Blue Gene/L Compute Chip

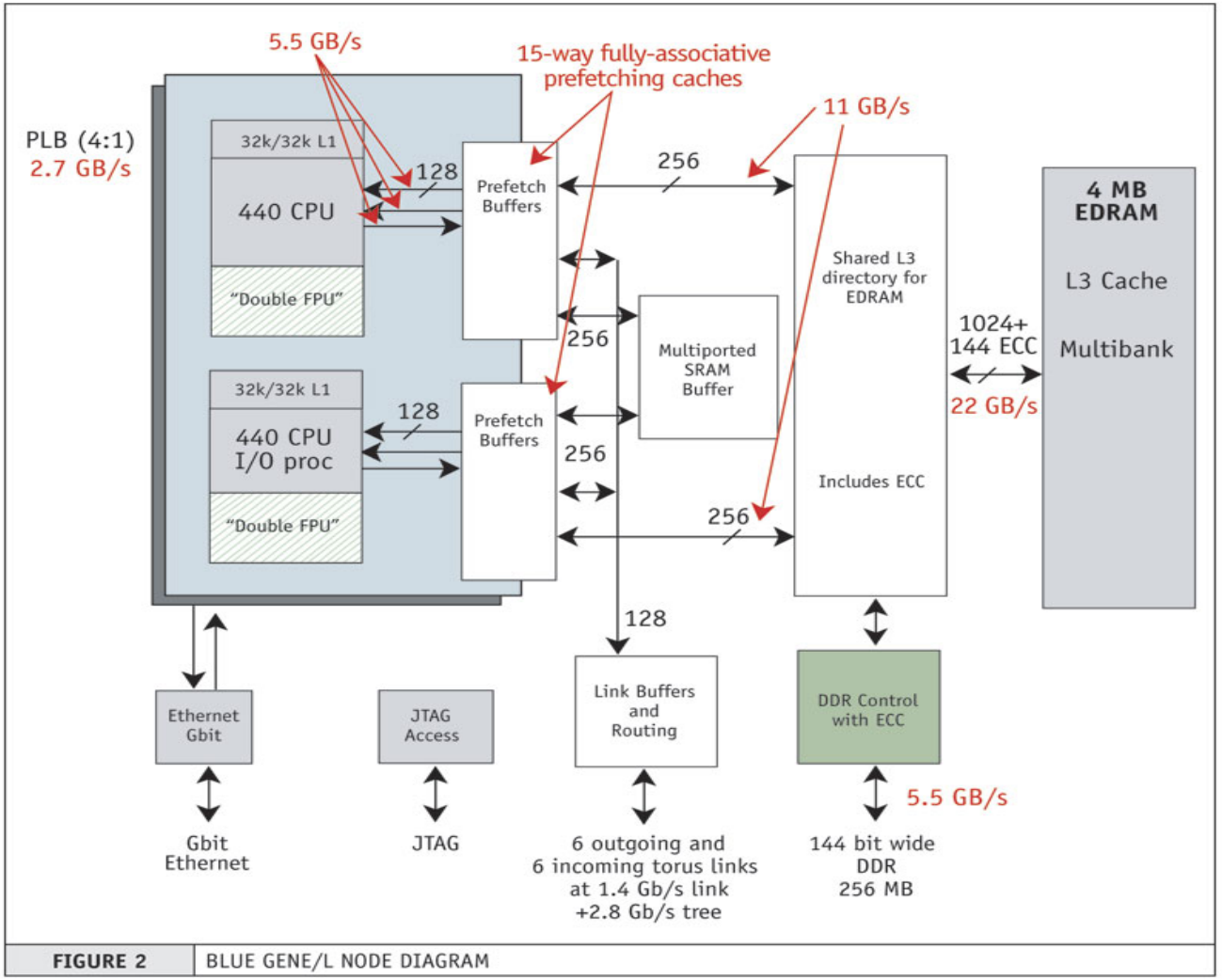


FIGURE 2 BLUE GENE/L NODE DIAGRAM

Virtual Node Mode vs. Co-processor Mode

Co-processor Mode:

- CPU0 does all computations
- CPU1 does all communications
- Communication may overlap with communication
- Peak performance is 5.6/2 = 2.8 GF
- #MPI-tasks = #Chips

(CP:18.7%;VN:16.6%)

Virtual Node Mode:

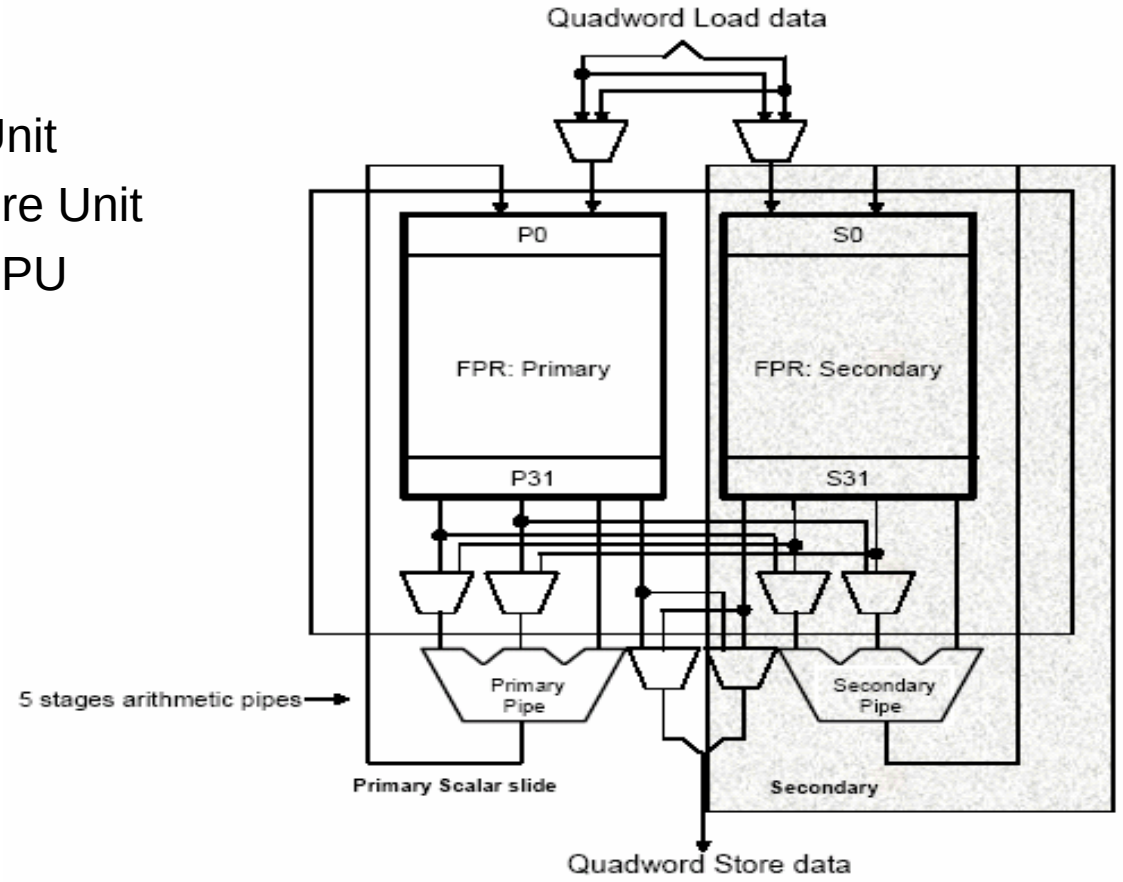
- CPU0, CPU1 run independent “virtual tasks”
- Each does own computation and communication
- CPUs communicate via cache/memory
- Communication and computation cannot overlap
- Peak performance is 5.6 GF
- #MPI-tasks = #CPUs

Blue Gene/L Compute Chip Performance Data

(VN MODE)	L1	L2	L3	Memory
Size/proc.	32KB data 32KB instr.	2KB	2MB	256MB
Latency (Clocks)	3	11	28/36/40 (h/m/m,b)	86
Bandwidth, random(B/clk)	NA	NA	1.8/1.2 (h/m)	0.5
Bandwidth, seq. (B/clk)	16.0	5.3	5.3	3.4
Line width	32	128	128	

Double Hummer FPU

- CPU contains
- One Integer Unit
- One Load/Store Unit
- One special FPU



Double Hummer FPU

- Blue Gene Compute Chip contains 2 FPUs with 32 registers each
- Registers can be accessed from both FPUs
- Instruction set contains special instructions to implement complex algebra

Mnemonic	Primary Ap=	Secondary As=
fpadd A,B,C	Bp+Cp	Bs+Cs
fxpmul A,B,C	Bp*Cp	Bp*Cs
fxcxnpma A,B,C,D	-(Bs*Cs-Dp)	Bs*Cp+Ds
fxcpmadd A,B,C,D	Bp*Cp+Dp	Bp*Cs+Ds

Double Hummer FPU

- Complex number multiply requires 2 instructions
  (1 cycle each, 5 cycle pipeline latency)

  fxpmul A, B, C
  …
  fxcxnpma E, B, C, A

- SU(3) matrix vector multiply requires 2*9 instructions

  … fxpmul … fxcxnpma … (fxcpmadd … fxcxnpma …)

  Matrix multiply has theoretical performance of 91.7% peak.

Using the Double Hummer

- The IBM XLc, XLC, XLf compilers can generate Double Hummer code
- However #pragma's and alignment statements have to be inserted
- Compiler usually generates poorly performing code (slower than w/o Double Hummer)
- IBM compilers allow usage of "intrinsic functions"
- Intrinsic functions implement an abstract assembly language
- Scheduling of instructions is done by compiler
- Thus routines can be easily rewritten using intrinsic functions
- Routines written using intrinsic functions are usually very efficient

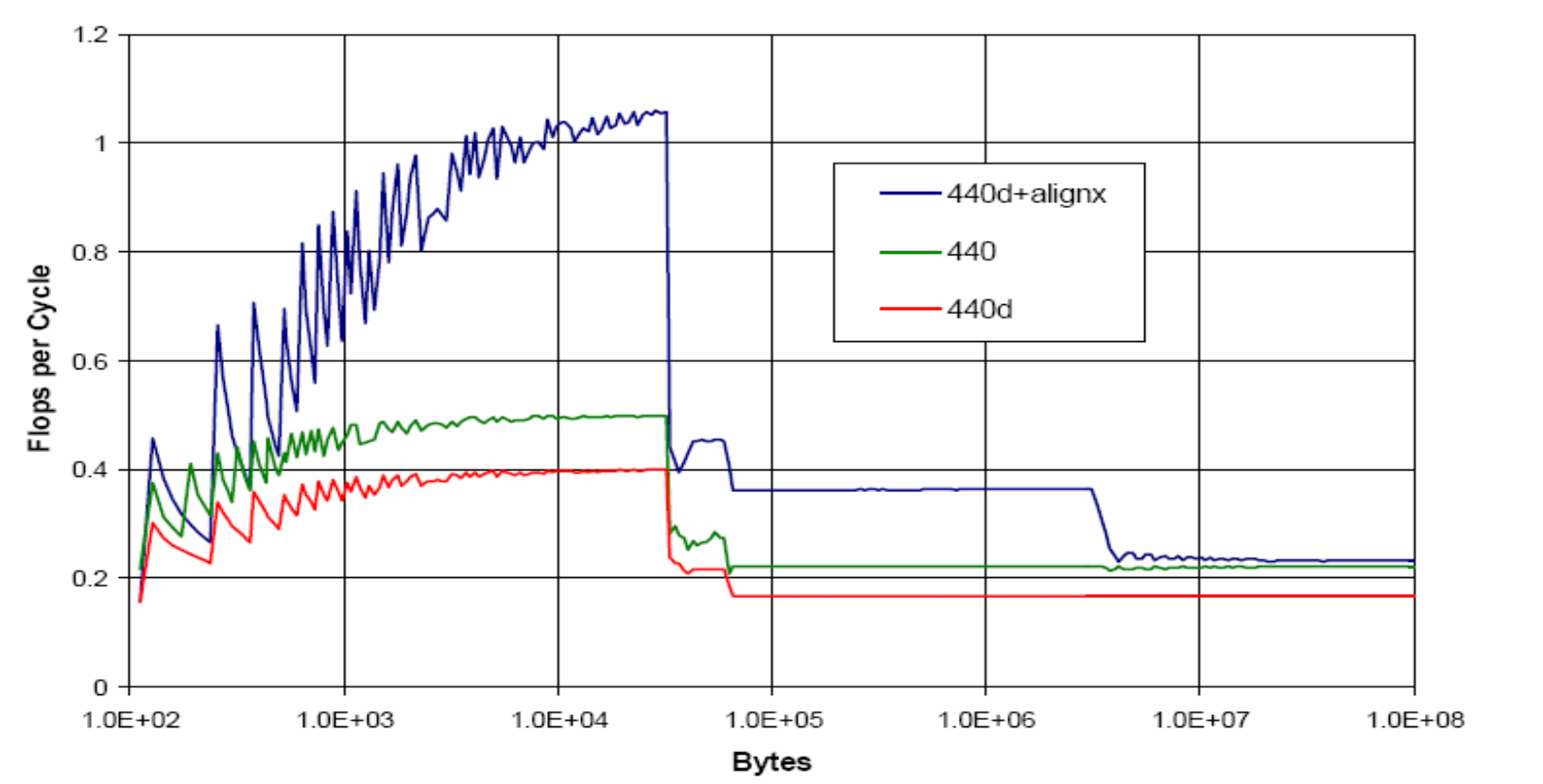
Using the Double Hummer

```
void spinor_caxpy(double a[2], double *x,
      double *y){
#pragma disjoint (*x, *y)
   __alignx(16, x);
   __alignx(16, y);
   int i;

#pragma unroll(12)
   for(i=0; i<vol*12; i++){
      y[2*i   ] += a[0] * x[2*i] – a[1] * x[2*i+1];
      y[2*i+1] += a[1] * x[2*i] + a[0] * x[2*i+1];
   }
}
```

440	3,6% (2,6%)
440d	3.0% (2,2%; 2,7%; 2,8%)
Intrin.	35,9% (11,7%)

```
void spinor_caxpy(double a[2], double *x,
      double *y){
#pragma disjoint (*x, *y)
   __alignx(16, x);
   __alignx(16, y);
   int i;
  double _Complex x00,…,y00,…,t00,…;
  double _Complex alpha;
  alpha=__cmplx(a[0], a[1]);

#pragma unroll(12)
   for(i=0; i<vol; i++){
      x00=__lfpd(&x[12*i]);…
      y00=__lfpd(&y[12*i]);…
      t00=__fxcpmadd(y00,x00,__creal(alpha));…
      t13=__fxcxnpma(t00,x00, __cimag(alpha));…
      __stfpd(&y[12*i]);…
   }
}
```

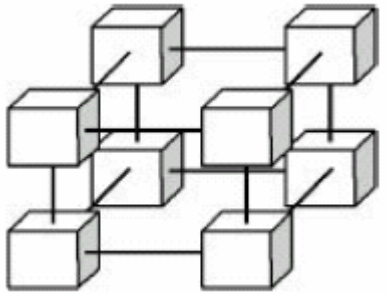
daxpy Performance

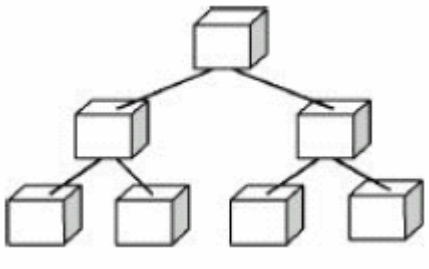


Blue Gene Network

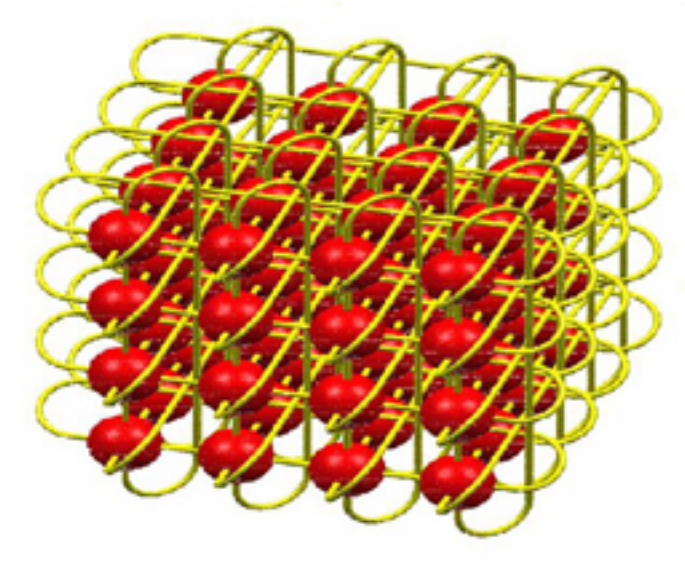
Blue Gene has 5 independent networks:

- Torus Network for point to point communication
- Tree Network for collective operations
- Tree Network for barriers and interrupts
- Gigabit Ethernet for file i/o
- Control Network



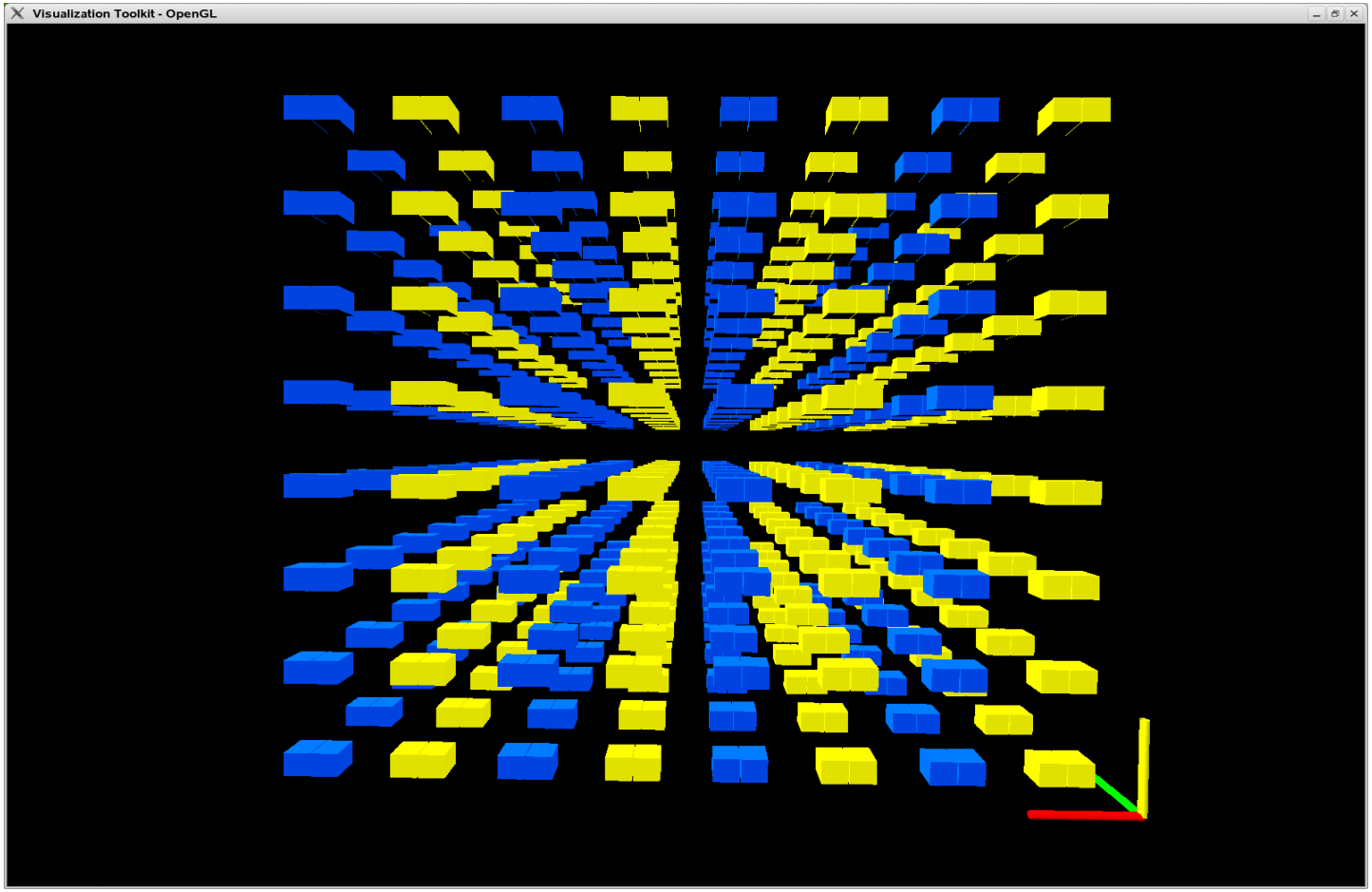


MPI Issues

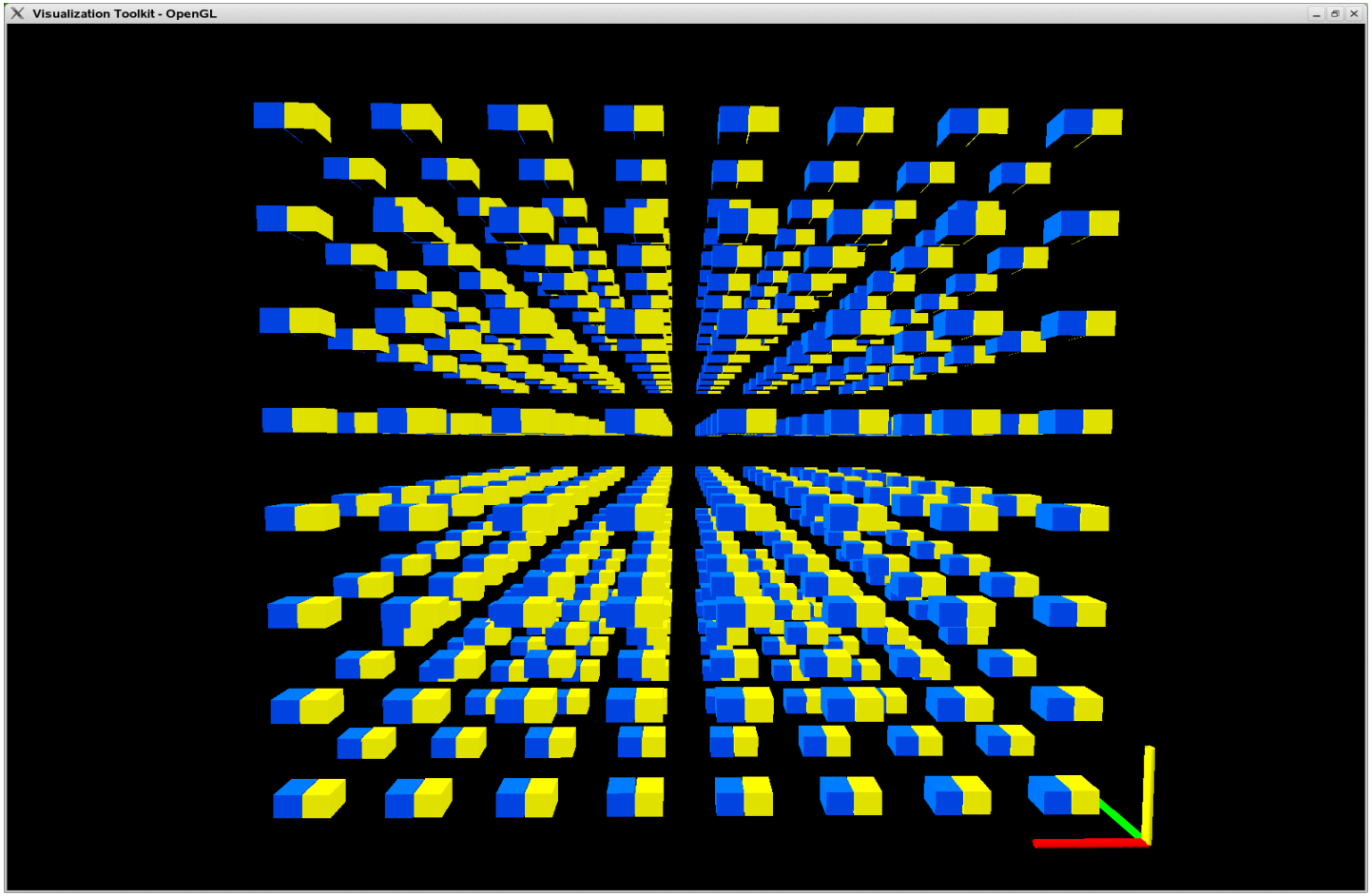


- Main Network is a 3dim Torus/Mesh with NN connections
- Optimal MPI performance is achieved when communicator matches hardware layout
- Suboptimal communicators result in unnecessary hops
- Mapping can be influenced via environment variables or a mapping file
- Communication can be set to be optimized for nearest neighbour communication via environment variable
- Various other environment variables exist that i.e. control collective ops.

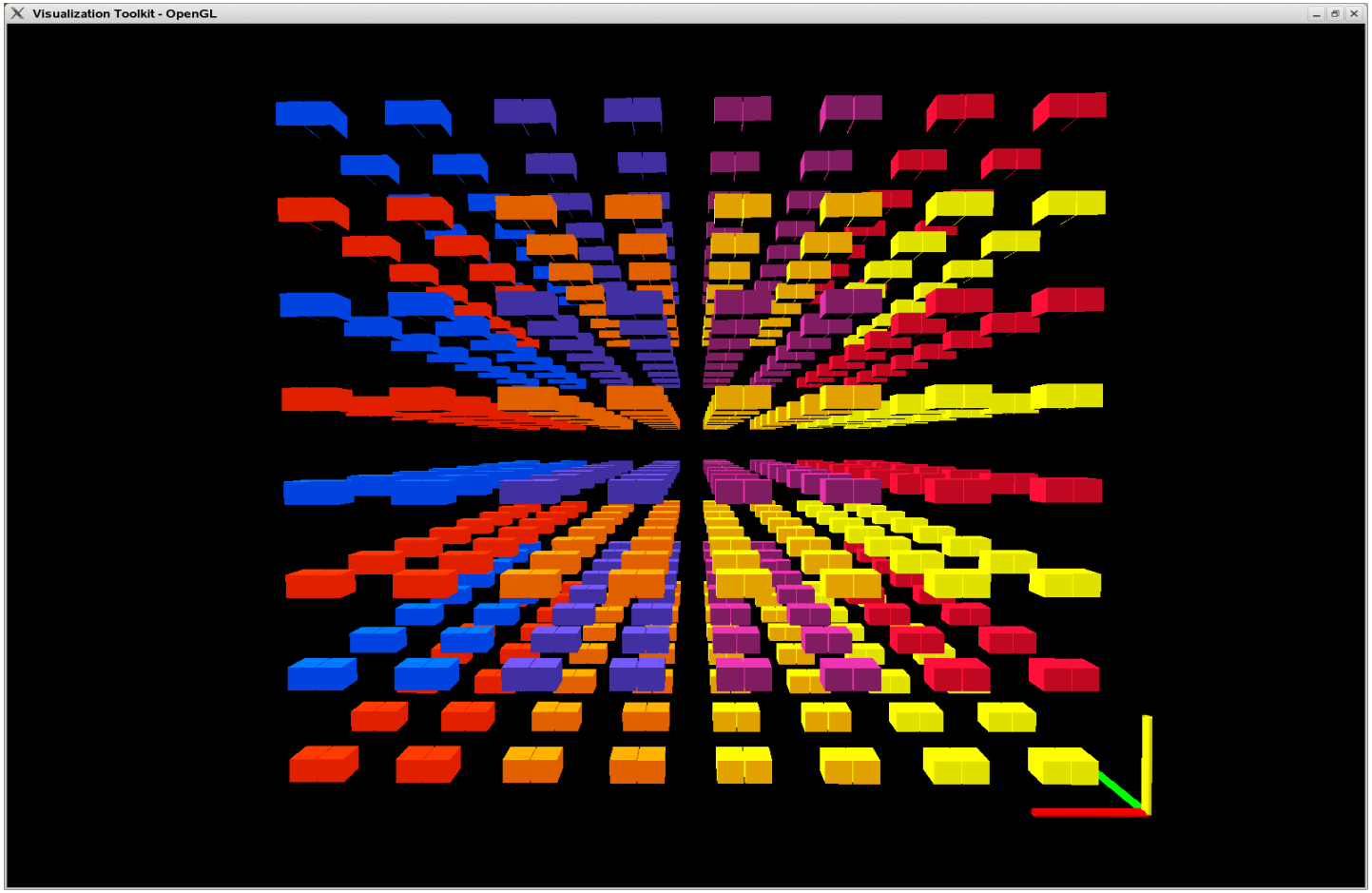
Suboptimal Communicator T-Dir



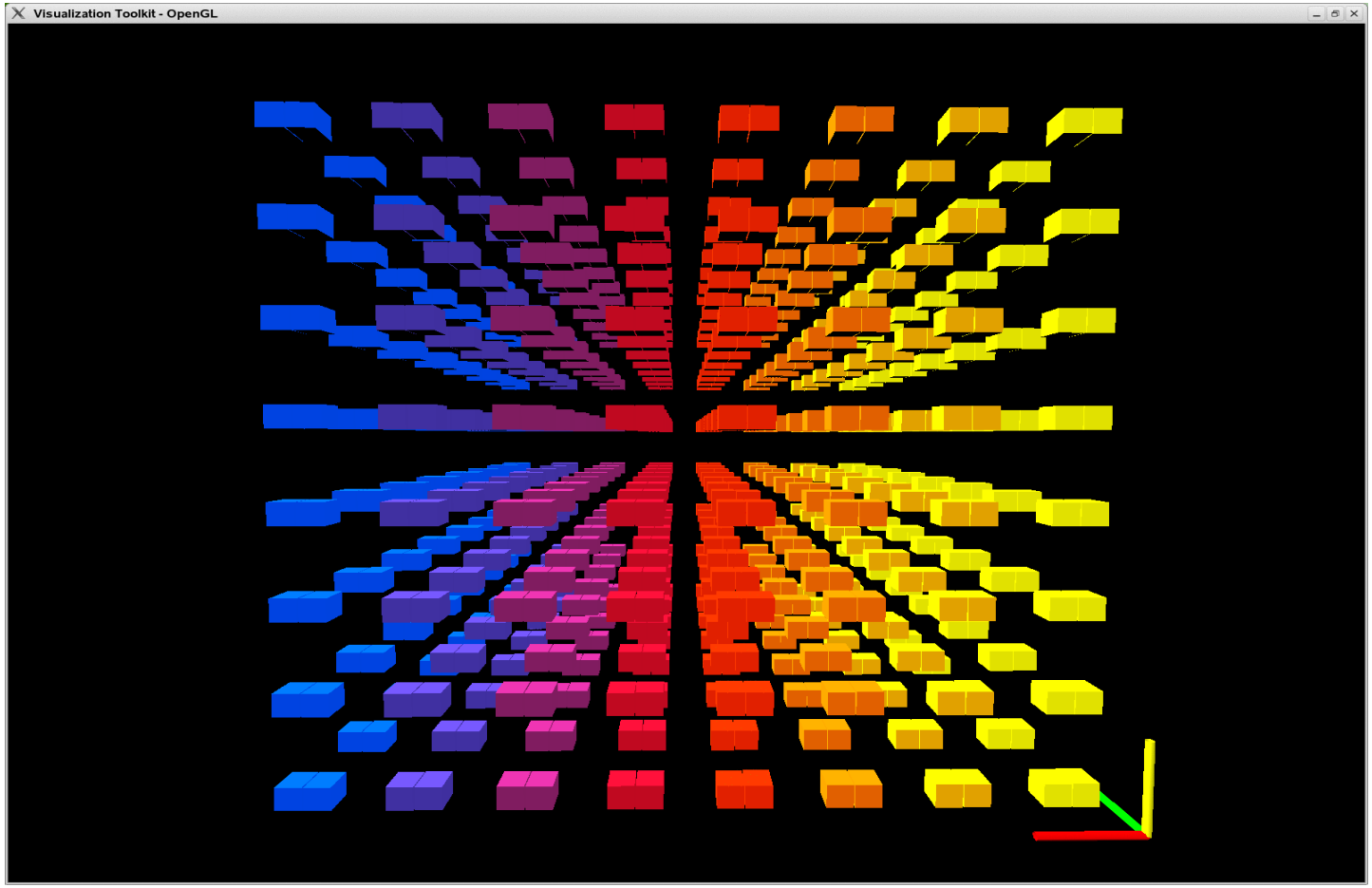
Optimal Communicator T-Dir



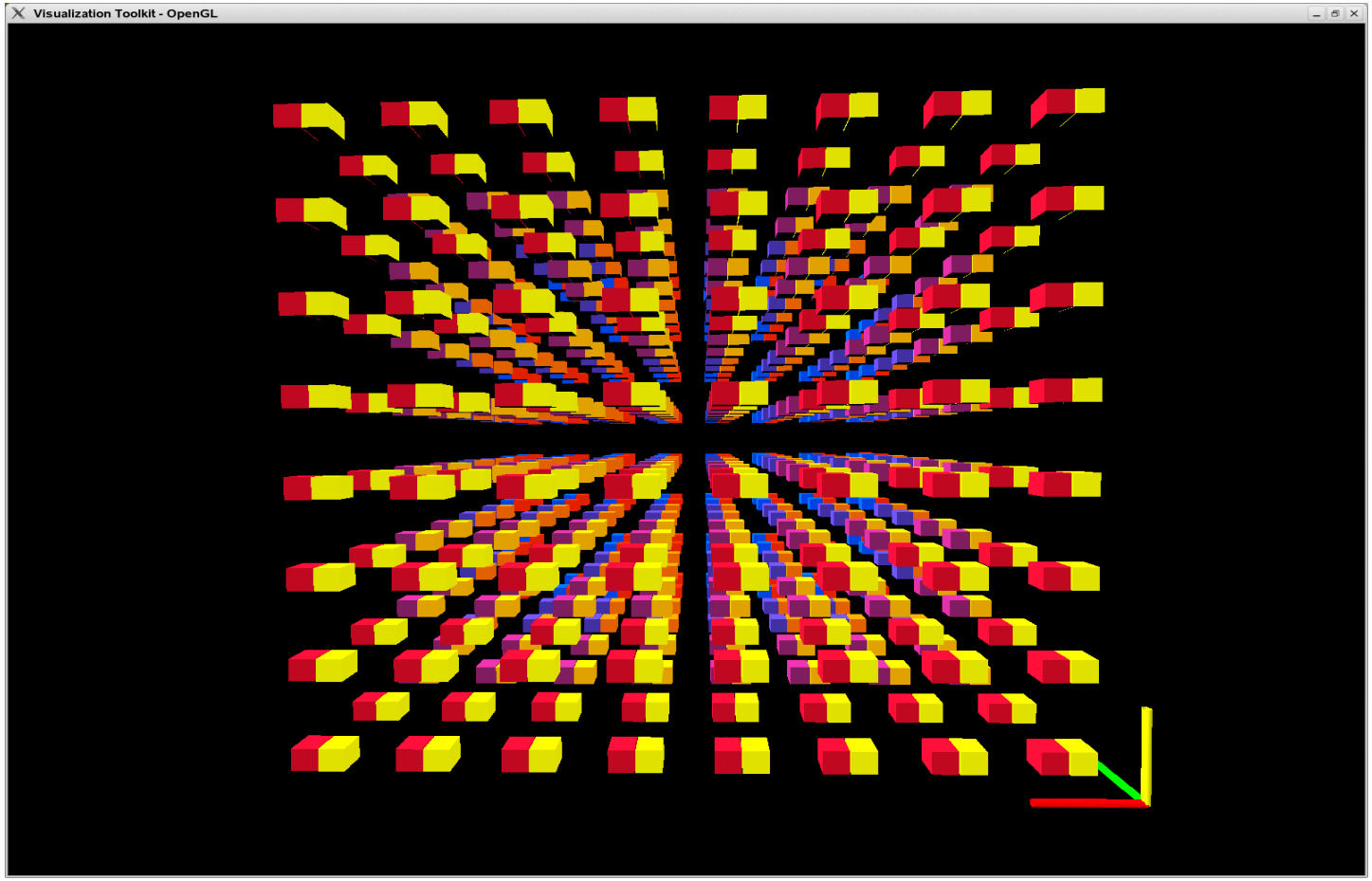
Suboptimal Communicator Z-Dir



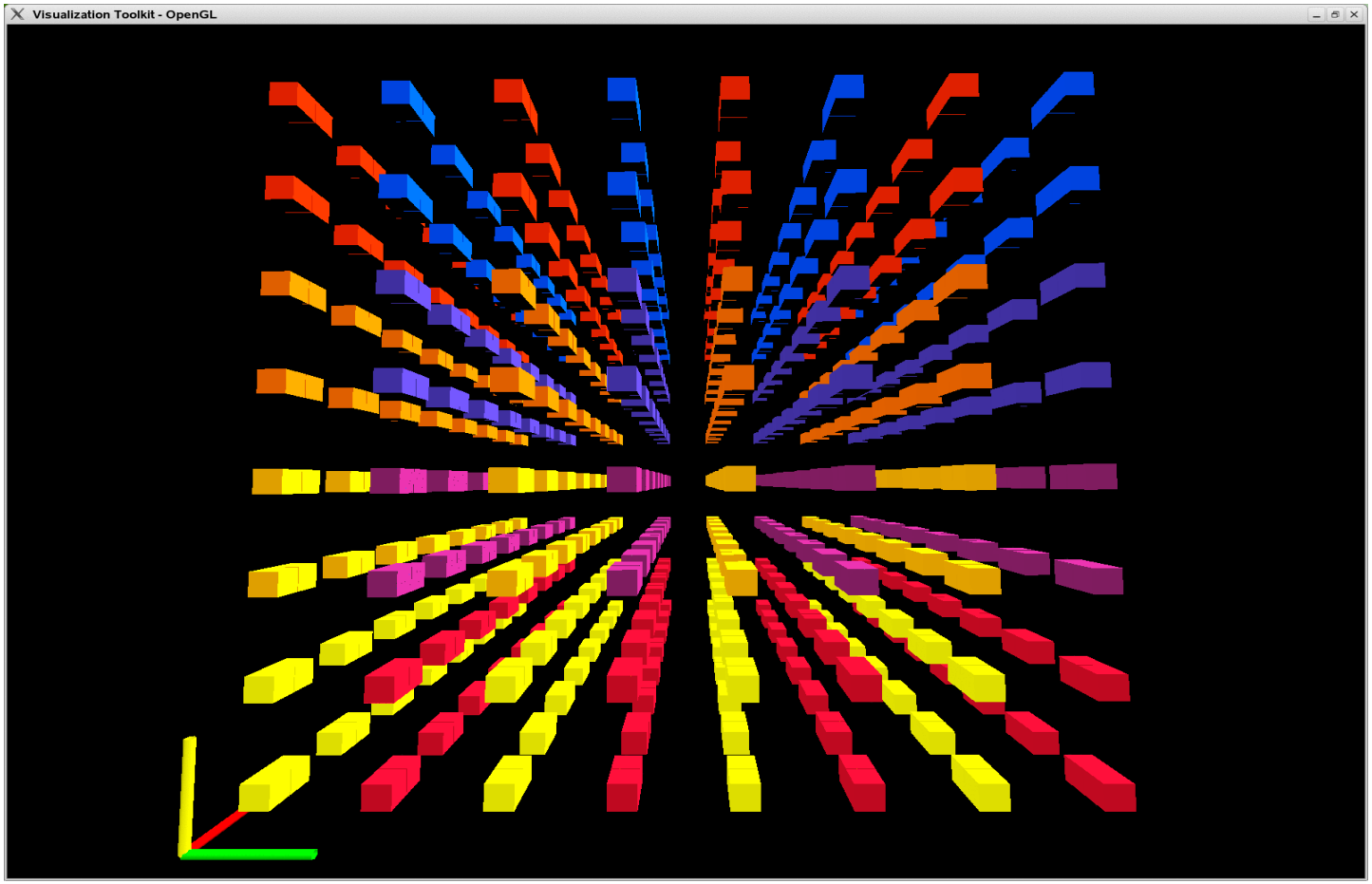
Optimal Communicator Z-Dir



Suboptimal Communicator X-Dir



Suboptimal Communicator Y-Dir



MPI Wilson Matrix

- Code written using intrinsics
- Uses project/reconstruct algorithm
- Communicates twospinors
- Communication via MPI
- Communicator matches hardware layout
- Communication switched to be optimized for nearest neighbour communication

32*8^3, 64 CPUs	**24 GFs** 13.1%
32*8^3, 256 CPUs	**74 GFs** 10.3%
32*16^3, 64 CPUs	**19 GFs** 10.6%
32*16^3, 256 CPUs	**118 GFs** 16.5%
32*16^3, 1024 CPUs	**336 GFs** 12.0%

IBM Dslash_eo

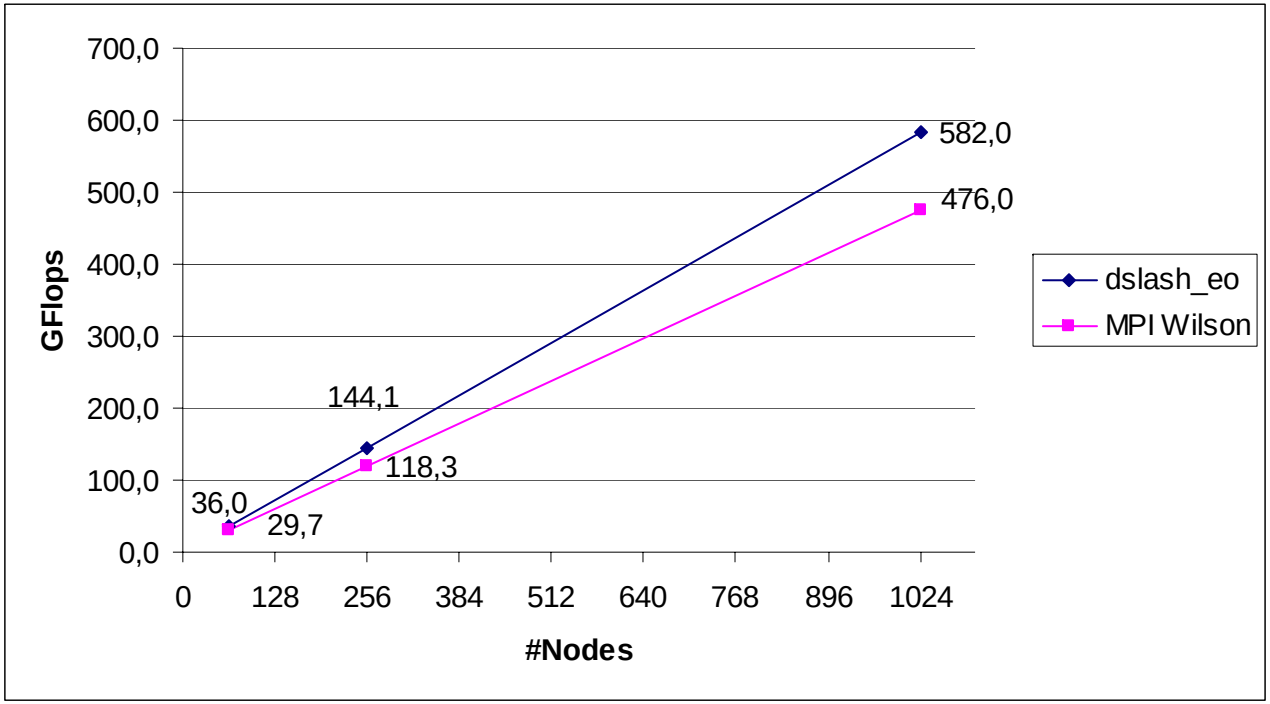
- Code written using g++ inline asm
- Uses project/reconstruct algorithm
- Communicates twospinors
- Communication via low-level APIs
- Parallelization scheme matches hardware layout
- Only nearest neighbour communication

32*8^3, 64 CPUs	**30 GFs** 16.5%(12.1%)
32*8^3, 256 CPUs	**105 GFs** 14.7%(NA)
32*16^3, 64 CPUs	**27 GFs** 14,9%(NA)
32*16^3, 256 CPUs	**116 GFs** 16.2%(12.1%)
32*16^3, 1024 CPUs	**430 GFs** 15.0% (11.0%)

Weak scaling

Dslash_eo:
Local lattice size:
16*4^3
(eo)

MPI Wilson:
Local lattice size:
8*4^3



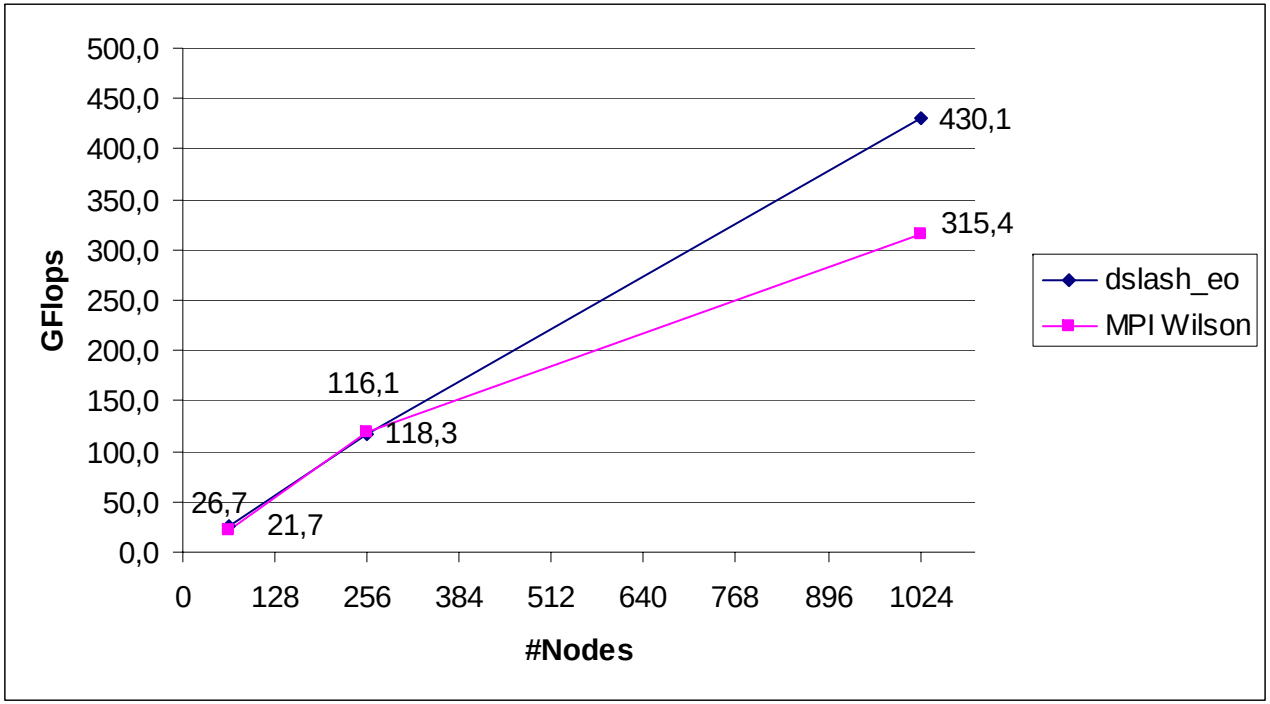
Dslash_eo: 20.0%

MPI Wilson: 16.6%

Strong scaling

Dslash_eo:
Global lattice size:
32*16^3
(eo)

MPI Wilson:
Global lattice size:
32*16^3



Dslash_eo:
14.9% -16.2%

MPI Wilson:
11.0%-16.5%

Conclusions

- Using intrinsics it is easily possible to tune the code
- Acceptable performance can be achieved even with MPI
- Including low level communication APIs the performance of the Wilson kernel can exceed 20% of Peak, expected final result (without e/o preconditioning) is 25% of Peak
- Blue Gene/L is a promising environment for QCD simulations

Thank you for your attention!