Scalability of a pseudospectral DNS turbulence code with 2D domain decomposition on Power4+/Federation and Blue Gene systems

D. Pekurovsky\textsuperscript{1}, P.K. Yeung\textsuperscript{2}, D. Donzis\textsuperscript{2}, S. Kumar\textsuperscript{3}, W. Pfeiffer\textsuperscript{1}, G. Chukkapalli\textsuperscript{1}

\textsuperscript{1}San Diego Supercomputer Center
\textsuperscript{2}Georgia Institute of Technology
\textsuperscript{3}IBM

SP SciComp, Boulder CO, July 20, 2006
Turbulence: examples

The small scales are important.
**DNS code**

- Code written in Fortran 90 with MPI
- Time evolution: Runge Kutta 2\textsuperscript{nd} order
- Spatial derivative calculation: pseudospectral method
- Typically, FFTs are done in all 3 dimensions.
- Consider 3D FFT as compute-intensive kernel representative of performance characteristics of the full code
- Input is real, output is complex; or vice versa
**3D FFT**

Use ESSL library calls for 1D FFT on IBM, or FFTW on other systems (FFTW is about 3 times slower than ESSL on IBM)

Forward 3D FFT in serial. Start with real array \((N_x,N_y,N_z)\):

- **1D FFT in \(x\) for all \(y\) and \(z\)**
  - Input is real \((N_x,N_y,N_z)\).
  - Call SRCFT routine (real-to-complex), size \(N_x\), stride is 1
  - Output is complex \((N_x/2+1,N_y,N_z)\) – conjugate symmetry: \(F(k)=F^*(N-k)\)
  - Pack data as \((N_x/2,N_y,N_z)\) – since \(F(1)\) and \(F(N_x/2+1)\) are real numbers

- **1D FFT in \(y\) for all \(x\) and \(z\)**
  - Input is complex \((N_x/2,N_y,N_z)\)
  - Call SCFT (complex-to-complex), size \(N_y\), stride \(N_x/2\)
  - Output is complex \((N_x/2,N_y,N_z)\)

- **1D FFT in \(z\) for all \(x\) and \(y\)**
  - Input and output are complex \((N_x/2,N_y,N_z)\)
  - Call SCFT (complex-to-complex), size \(N_z\), stride \((N_xN_y)/2\)

**Inverse 3D FFT: do the same in reverse order.** Call SCFT, SCFT and SCRFT (complex-to-real).
3D FFT cont’d

Note: Alternatively, could transpose array in memory before calling FFTs, so that strides are always 1. In practice, with ESSL this doesn’t give an advantage (ESSL efficient even with strides > 1)

Stride 1: 28% peak Flops on Datastar
Stride 32: 25% peak
Stride 2048: 10% peak
Parallel version

- Parallel 3D FFT: so-called transpose strategy, as opposed to direct strategy. That is, make sure all data in direction of 1D transform resides in one processor’s memory. Parallelize over orthogonal dimension(s).
- Data decomposition: \( N^3 \) grid points over \( P \) processors
  - Originally 1D (slab) decomposition: divide one side of the cube over \( P \), assign \( N/P \) planes to each processor. Limitation: \( P \leq N \)
  - Currently 2D (pencil) decomposition: divide side of the cube (\( N^2 \)) over \( P \), assign \( N^2/P \) pencils (columns) to each processor.
Memory and compute power

• 2048³ on 2048 processors – 230 MB/proc. This problem fits on Datastar and Blue Gene. Extensive simulations under way.
• 4096³ on 2048 processors – 1840 MB/proc. This problem doesn’t fit on BG (256 MB/proc), and fits very tightly on Datastar.
  • Anyway, computational power of 2048 processors is not enough to solve problems in reasonable time. Scaling to higher counts is necessary, certainly more than 4096.

Therefore, using 2D decomposition is a necessity
(P > N)
1D Decomposition

1) Transform in X

2) Transform in Z

3) Transpose

4) Transform in Y
2D decomposition

1) Transform in X

2) Transpose X-Z

3) Transform in Z
2D Decomposition cont’d

4) Transpose Z-Y

5) Transform in Y
Communication

Global communication: traditionally, a serious challenge for scaling applications to large node counts.

- 1D decomposition: 1 all-to-all exchange involving P processors
- 2D decomposition: 2 all-to-all exchanges within \( p_1 \) groups of \( p_2 \) processors each \( (p_1 \times p_2 = P) \)
- Which is better? Most of the time 1D wins. But again: it can’t be scaled beyond \( P=N \).

Crucial parameter is bisection bandwidth
Alternative approaches attempted

- Overlap communication and computation
  - No advantage.
- Hybrid MPI/OpenMP
  - No advantage.
- Transpose in memory, call ESSL routines with stride 1
  - No advantage or worse
Platforms involved

- **Datastar: IBM Power4+ 1.5 GHz**
  - at SDSC, up to 2048 CPUs
  - 8 processors/node
  - Fat tree interconnect

- **Blue Gene: IBM PowerPC 700 MHz**
  - at SDSC, up to 2048 CPUs
  - at IBM’s T.J. Watson Lab in New York state, up to 32768 CPUs (2\textsuperscript{nd} in Top500 list)
  - 2 processors/node
  - 3D torus interconnect
Performance on IBM Blue Gene and Datastar

VN: Two processors per node
CO: One processor per node
A closer look at performance on BG

DNS 2048^3

- VN total
- CO total
- VN comm
- CO comm

N proc vs T x P

- 0 5000 10000 15000 20000 25000 30000 35000
- 0 20000 40000 60000 80000 100000 120000 140000

- 2048
- 4096
- 8192
- 16384
- 32768

San Diego Supercomputer Center
University of California, San Diego
Communication model for BG

- 3D Torus network: 3D mesh with wrap-around links. Each compute node has 6 links.
- Modeling communication is a challenge for this network and problem (mapping 2D processor geometry to 3D network topology). Try to do a reasonable estimate.
- Assume message sizes are large enough, consider only bandwidth, ignore overhead.
- Model CO (VN is similar)
Two subcommunicators: S1 and S2

\[ P = P_1 P_2 = P_x P_y P_z \]

Step 1: All-to-all exchange among \( P_1 \) processors within each group \( S_1 \)

Step 2: All-to-all exchange among \( P_2 \) processors within each group \( S_2 \)

By default tasks are assigned in a block fashion (although custom mapping schemes also available to the user and are interesting option)

- \( S_1 \)'s are rectangles in X-Y plane: \( P_1 = P_x \times (P_y / k) \)
- \( S_2 \)'s are \( k \) Z-columns: \( P_2 = k P_z \)

\( B = 175 \text{MB/s}, \) 1-link bidirectional bandwidth
Upper bound

- Assume dimensions are independent
- Find bottlenecks in each dimension, sum up the maximum time in each dimension
- Some links are idle some of the time

Take first step – communicator group is a plane $P_x \times (P_y/k)$

Assume torus (wraparound links) in x dimension, but not in y for $k > 1$

Bisection bandwidth across y lines is $b P_x$

Proceed in $P_x$, stages

The number of messages exchanged is $P_x, (P_y/2k)^2$ for each stage

Total time for y-dimension bottleneck is $t_y = (N_b/B) P_x, (P_y/2k)^2$

Now independently consider X direction, and derive

$$t_x = (N_b/B) (P_y/k) (P_x /2 )^2 (1/2)$$

Summing up, and using $N_b=(4N^3)/(P \ P_1)$, we get

$$T_1 = (N^3/P\ B) ((1/2)P_x+(1/k)P_y)$$
Upper bound cont’d

Now step 2: k Z-lines in each communicator, all lie in Y-Z planes
Again, assume staged implementation. First communicate along y.
Dimension size is k, bisection bandwidth is \( bP_x \) but it’s shared among \( P_1 \) groups. Do this \( P_z \) times.

Time is \( \frac{N_b}{b} \left( \frac{k}{2} \right)^2 \frac{P_1 P_z}{P_x} (1/2) = \left( \frac{N^3}{Pb} \right) \frac{P_y}{2} \)

Finally, exchange within Z-lines (k times)
\( T_z = \frac{N_b}{b} \left( \frac{P_z}{2} \right)^2 /2 \) (2 comes from torus links)
So \( T_2 = \left( \frac{N^3}{Pb} \right) (P_y + kP_z/2) \)

Summing up,
\[
T_{up} = T_1 + T_2 = \left( \frac{N^3}{P B} \right) \left( \frac{P_x}{2} + (1/2+1/k)P_y + P_z/2 \right)
\]

For the lower bound, assume all links busy, and only the maximum time counts. Obtain
\[
T_{lower} = \left( \frac{N^3}{P B} \right) \left[ \text{Max}(P_x/2, P_y/k) + 1/2 \text{ Max}(P_y, P_z) \right]
\]
Communication model for BG

![Graph showing communication model](image)
Summary

• 2D decomposition enables significantly increased scalability of the DNS turbulence code
• Achieve good scaling on both IBM SP4 and BG/L (up to 32k processors)
• Ready for next generation of machines
• DNS turbulence is one of the 3 Model Problems in the recent NSF Petascale RFP