

BGW Consortium Day on July 19

1. Project Title: DNS of Channel Turbulent Flow at High Reynolds Number
2. Project Investigator(s): Jin XU, Argonne National Lab.
3. Introduction:

Turbulence remains one of the most challenging problems in Physics. The complexity of solving turbulence in theory is beyond people's ability now. The fast development of supercomputing have made it possible to explore the turbulent flow through Direct Numerical Simulation (DNS).

Since Kim, Moin and Moser published their paper on DNS of turbulent flow in a channel in 1987, DNS has become an important tool for investigating the dynamics of wall-bounded, turbulent shear flows. The advantages of using DNS for turbulence research are: (1) no turbulence closure models are required, and the full Navier-Stokes equations can be solved without approximation; (2) much more detailed and accurate information of flow quantities (which is quite difficult and expensive to measure through experiment) can be obtained. Since the number of grid points needed to resolve the flow in DNS is proportional to Re^3 , this approach is presently limited to low Reynolds number. As supercomputers become faster, simulations at higher Reynolds number will be feasible. This has great impact on the fundamental development of turbulence modeling, including sophisticated industrial applications.

4. The Fourier Spectral Element code

Turb3d is a C/C++ code for direct numerical simulation (DNS) in turbulent channel flow. It has been developed and optimized over last several years. Our project targets at high Reynolds channel turbulence DNS. We use Fourier Spectral Element method for the discretion, and it is similar to Kim, Moin and Moser's method. The domain is a plane channel with periodic boundary conditions in stream-wise and span-wise directions. The flow field can be expanded by Fourier and Polynomial bases. Using this method, we can choose the position of elements boundary, so that we can control the number of points in the near wall region. We also parallelized the code by decompositions. We used high-order time splitting method (proposed by Karniadakis, Israeli, Orszag [1991]) to do the time integration. In order to eliminate the aliasing error generated in nonlinear step, we perform the nonlinear step in physical space using "3/2 rule". This means the mesh was expanded by 3/2 times larger in both stream-wise and span-wise directions. After evaluating the nonlinear terms, we transform it back to the normal mesh. The code uses standard high performance numerical libraries, such as FFTW, BLAS, LAPACK, etc.

5. Objectives:

First we want to test the scalability of Turb3d code on Blue Gene machine, using up to 32,768 processors. Second, we are targeting at Reynolds number ($Re_\tau > 1000$), and try to identify how many processors and time it required for this highest Reynolds number so far achieved in the world. As Blue Gene is the fastest machine, it is very promising to conduct this research.

6. Results:

(1). One rack system:

NPX	NPZ	NL	PR	VIS	TOT (seconds)
1	128	19.15	7.65	10.18	36.98
2	64	19.68	7.3	9.21	36.19
4	32	20.38	7.02	8.68	36.08
8	16	19.48	6.87	8.4	34.75
16	8	20.13	6.84	8.32	35.29
32	4	19.82	6.69	8.2	34.71
64	2	18.48	6.56	8.12	33.16
128	1	17.21	6.47	8.12	31.8

Table 1 Mesh $1024 \times 513 \times 1024$ on one rack system

On one rack using 2048 processors, we tested $1024 \times 513 \times 1024$ mesh with 16 elements, and 33 points in each element. Timing is shown in Table 1. This mesh is sufficient for DNS of $Re_\tau = 1000$ flow. As we can see from the table, the speed increases as we increase the processor number in x direction and decrease the processor number in z direction. Another parameter setting with 8 elements and 65 points in each element is also possible on one rack system. But the speed is about 20% slower than 16 elements.

(2). 2 ~ 16 racks system mesh

Several different meshes and racks have been tested, and we achieved good scalability.

a. Running same number of processors in vn mode is usually about 40% faster than in co mode. From case A and B, we see that running 2048 processor in co mode is about 38.6% times faster than using vn mode. Case H and I show that there is about 50% speedup by using vn mode instead of co mode.

Case	Mesh	Racks	Processors	Elements	Pts/Elet	Time
A	1024×513×1024	1	2048	16	33	35
B	1024×513×1024	2	2048	16	33	25.9
C	1024×513×1024	4	8192	16	33	11.5
D	1024×1025×1024	4	8192	32	33	24.92
E	1280×1025×1280	4	8192	32	33	47.04
F	1536×769×1536	4	8192	32	33	19.6
G	2048×513×2048	4	8192	16	33	46.5
H	2048×513×2048	8	16384	16	33	26.1
I	2048×513×2048	16	16384	16	33	17.0
J	2048×513×2048	16	32768	16	33	14.8

Table 2. On 2~16 rack systems

b. Since Fourier series have been used in x and z directions, we found that good scalability can be achieved by increasing the mesh points in these two directions. As can be seen in case C and G, the speed is 4 times slower when the mesh size increases by 4 times. The parallel efficiency is about 99%.

c. Since spectral element method is used in wall normal direction, when we increase the mesh points in this direction, we need to increase element numbers or mesh points in each element. Increasing mesh points in each element will lead to increasing of preconditioned matrices. The memory and complexity does not increase linearly, so the scaling is not comparable when we increase the total mesh points in y direction. As can be seen in case A and G, the mesh increases 4 times and processor number also increases by 4 times, but the speed is about 33% slower. The parallel efficiency is about 75%

d. Running on 8 or 16 racks system is very time consuming, and it is easy to have one processor failure randomly. Jim Sexton suggested running in safe mode. The safe mode can automatically save the data before the machine crash. We tested the code running under safe mode for case I, and it takes 20.9 seconds/step. This gives about 19% slowness in production runs, which is acceptable.

e. On 16 racks, we found the speed for different element is not so large as on one rack system. This is probably due the increase of mesh points in x and z direction. It may reduce the influence of other parameters. We found that there is only 10% different by doubling element numbers.

(3). I/O

DNS needs fast speed and large memory. Usually we need to do I/O for 10~100 GBs. I have tested several different methods for I/O.

a. It is possible to input 2048 files directly into one directory, however, it takes about 15 minutes in ASCII format for 15.7GBs. To output the same amount of data takes similar or longer time.

- b. To input 15.7 GBs data from 16 directories in binary format, it takes 2 minutes.
- c. To output 7.4 GBs data to 8 directories, with 128 processors doing I/O takes about 358 second for double precision number in binary and 215 seconds for float.
- d. To output 13 GBs data to 16 directories, with 16 processors doing I/O takes about 687.0 seconds. With 128 processors doing I/O, it only takes 465.s seconds.
- e. There are several issues remain to be tested. Due to time limit, I/O on 16 racks has not been done so far.

(4) Other problems

- a. We found that same code compiled on different Blue Gene machines had different results. Later we identified that this problem was due to an array was failed to be initialized to zero which caused the instability of code on BGW, although it did not demonstrate any problem on BGL/ANL. After fixing this problem, we obtained exactly the same results by running on BGL/ANL and BGW system.
- b. There exist some problems to compile C code using blrts_xlC compiler on BGW, and it seems to have some conflict with the libraries on BGW. I tried g++ compiler instead, and it works on BGL/ANL, but still have some problems on BGW.

7. Summary:

We have achieved good scalability on 2~16 racks BGW system. The code Turb3d has been successfully tested on 16 racks BGW system, and fast speed has been demonstrated. Based on present benchmark results on BGW, we are confident to use 16 racks on BGW for high Reynolds number DNS. After this BGW day test, several critical problems have been identified and successfully solved. The code is fully in production stage by now. In the future, we will perform more tests to optimize parallel I/O of ~100 GBs data. We also found several aspects needed to be improved in the future, and it may dramatically increase the speed of the code, and decrease the memory requirement in near future.