Featured Story Corner

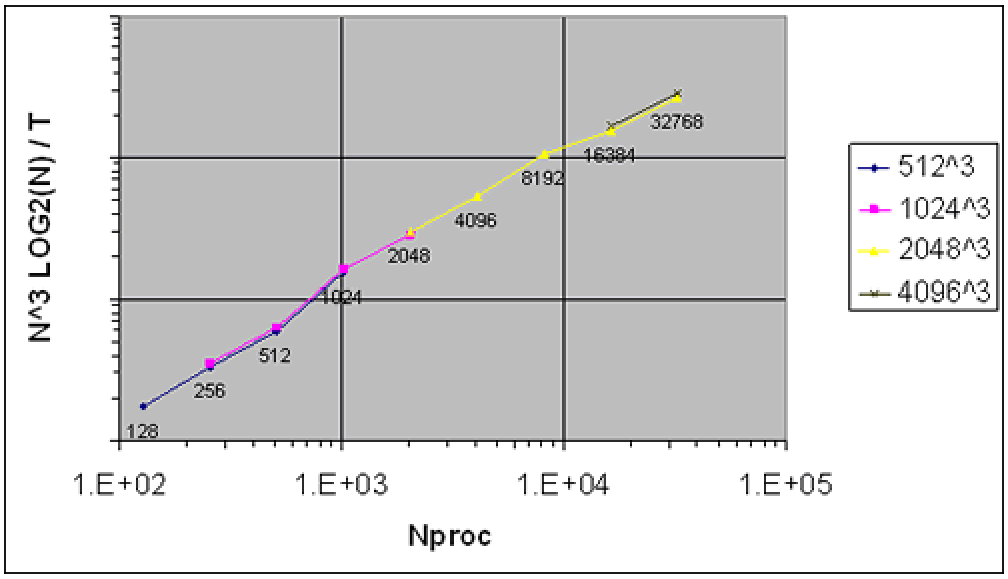
Highly Scalable Parallel Three-dimensional Fast Fourier Transforms

—Dmitry Pekurovsky

Fast Fourier Transforms (FFT) is a very commonly used numerical technique in computational physics, engineering, chemistry, geosciences, and other areas of high performance computing. Since this is a mature subject of research, there are many excellent FFT libraries available. Some of those libraries are optimized for a particular architecture while others are generic; some are proprietary, while others, such as theFastest Fourier Transform in the West (FFTW) are open source. FFT libraries typically provide several types of one dimensional FFT transforms, and some libraries provide two and three dimensional (3D) transform capability. However, not all of the libraries provide parallel implementations of FFTs. Those that do provide the parallel capability are subject to an important constraint that we will discuss below, leading to limited scalability. The subject of this article is parallelizing 3D FFT in a highly scalable fashion.

Consider a parallel implementation of a 3D FFT of a 3D array of complex numbers. Somehow, this array needs to be decomposed among P processors. One easy way to do it is to divide the volume of data into slabs, each consisting of one or several planes. If each processor gets such a slab to hold in its memory, this scheme is called*1D* or *slab decomposition*. It is most common for existing FFT libraries to use this scheme (for example, PESSL, which is the IBM's proprietary scientific library). The slab decomposition suffers from an important limitation: the number of processors cannot be more than the number of planes, that it the largest linear size of the data volume. While it may have been enough so far, we are close to the threshold when this limitation begins to substantially limit the computational power available to address large-scale simulations of today. For example, according to Georgia Tech Engineering Professor P.K.Yeung, currently, state-of-the-art simulations have been performed on 2048^3 grid size, with the only exception being researchers working on the Earth Simulator who reported using a 4096^3 grid. (Prof. Yeung is our partner in one of SDSC's recent Strategic Application Collaborations (SAC) projects concentrating on DNS studies of turbulence; see April 2006 Thread feature article.) Prof. Yeung has been using 1024 to 2048 of Datastar's (SDSC's IBM Power4 machine) processors to simulate 2048^3 grid and would like to go to 4096^3 in the near future when an adequate computational platform becomes available. Migrating from a 2048^3 to a 4096^3 problem size requires more processors than 4096, since the memory requirement grows by a factor of 8 and the problem's complexity increases by a factor of 16 when the linear problem sizes doubles. Therefore, the slab decomposition will not do the job. Since in the next several years we are likely to see a rapid increase in the number of processors in high performance compute architectures, more and more computational scientists will find it necessary to scale to the highest number of processors possible in order to stay on the cutting edge of their field. Therefore, the slab decomposition, while being the simplest parallelization strategy, is becoming less adequate for the imminent needs of many researchers.

An alternative approach was used to overcome the scalability limitation in the SAC project on DNS turbulence mentioned above. Here the data volume is divided into columns, which is called *2D* or *pencil decomposition*. This approach allows using up to N^2 processors (if N is the linear volume dimension). Using the 2D decomposition, a kernel was created to address the 3D FFT needs of the DNS turbulence application. The kernel was optimized for parallel and single-CPU performance. After the kernel worked to everyone's satisfaction, it was incorporated into the DNS application by changing the data decomposition scheme from the original 1D to 2D, which required a moderate amount of effort. Now the application is equipped to scale up to N^2 processing elements. It was recently tested on a Blue Gene machine at IBM's Watson Lab, showing good scaling up to 32,768 processors with a 4096^3 problem size. The scaling plot on Blue Gene is shown below (se Fig. 1). The execution speed (number of steps per second of execution), normalized by the problem size, is plotted on the Y-axis. We observe that scaling is less than perfect (note: the scale is logarithmic on both axes), but quite good, up to 32768 processors. With slab decomposition, the 4096^3 simulation would not even be possible since the minimum number of Blue Gene processors required to fit the problem in memory is 16384. The 2048^3 problem would fit on 2048 processors, but the maximum performance would be limited. For details about this project, please see the archived scaling analysis presentation (PDF, 2.2 MB).



The execution speed (number of steps per second of execution), normalized by the problem size, is plotted on the Y-axis.

In summary, 2D decomposition increases scaling capability of codes making intensive use of 3D FFT. Implementing 2D decomposition requires a somewhat more elaborate communication module than 1D decomposition. As an outcome of the SAC project mentioned above, we created a standalone 3D FFT module, which can be incorporated into other applications relying on 3D FFT usage.This module, dubbed P3DFFT, provides 3D FFT utilities in an optimized fashion and hopefully will be of benefit to our users who are in the process of gearing their software for petascale computing. See the SDSC User Services Applications section for a downloadable beta version and user guide.

We would appreciate hearing from users about the degree of interest in such a utility, as well as any feedback about using the beta version, for example bugs and suggested features. Please contact Dmitry Pekurovsky with your comments.