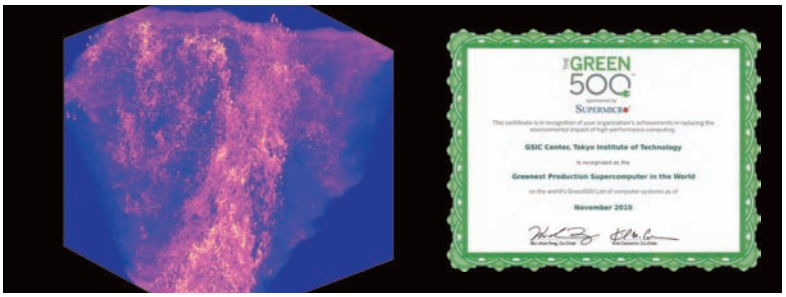
e-Science Journal
Global Scientific Information and Computing Center
Vol. 3

TSUBAME ESJ.



TSUBAME 2.0 Begins

The long road from TSUBAME1.0 to 2.0 (Part Two)

GPU Computing for Interstellar Atomic Hydrogen Turbulence

- 11.5 TFlops with 120 GPU on TSUBAME 1.2 -

Fast Fourier Transform using GPU

TSUBAME 2.0 Begins

The long road from TSUBAME1.0 to 2.0 (Part Two)

Satoshi Matsuoka*
* Global Scientific Information and Computing Center

TSUBAME2.0 is Japan's first muti-petascale supercomputer with multitudes of innovative in architectural and software features such as extensive use of GPUs, highly scalable and high-bandwidth node and network design, as well as massive utilization of advanced I/O technologies such as SSD. TSUBAME2.0 had gone into production operation as of early Nov., 2010; the part two of the article will cover the benchmarking activities conducted just prior to that event, characterizing the overall performance of the machine in terms of dense compute-bound applications, power consumption, and high-bandwidth applications . TSUBAME2.0 became the fourth fastest supercomputer in the world on the Top500, as well as being awarded the "Greenest Production Supercomputer in the World" award on the Green 500 in Nov. 2011, and at the sametime setting world record on bandwidth intensive ASUCA weather code, demonstrating its innovative performance and design.

1 The First Cries of a Newborn: TSUBAME2.0

The first challenges for the newborn TSUBAME2.0, as its installation was finishing in Oct., 2010, were large-scale benchmarks that would use the entire machine, such as Linpack. Such all-machine benchmarks in the early days of supercomputer inception are fairly commonplace and important for the following technical reasons:

(1) The number of components embodied in a large supercomputer such as TSUBAME2.0 is more than several thousands of factors greater than a standard PC. Even if we count merely the number of sockets of compute elements, namely the GPUs and CPUs, TSUBAME embodies over 7000, whereas a standard PCs only would have one or two. Memory on PCs would usually be a few gigabytes, whereas TSUBAME2.0 embodies nearly 100 Terabytes, several tens of thousands greater. For such numerous components to perform under prolonged stress is one of the most important factors in attaining stable and reliable operation, as the failure rates are roughly proportional to the number of components in the system; that is to say, a PC which only fails once in the years would fail over 3 times a day if enlarged to the size of TSUBAME2.0.

(2) At the same time, it is very important to confirm that expected design performance is being met in reality is especially important for supercomputers, as slightest deviation could profoundly deteriorate overall performance of the system. It is quite common in a supercomputer for a component which seemingly function but not performing up to its specs become the critical performance bottleneck. For example, the Infiniband network employed in TSUBAME has several speed specs, and the system automatically detunes itself to the lowest mutual common denominator between the end points, so that communication can be established. However, when some glitch occurs that compromises a connection, and due to some reason the established communication is of much lower specs, it would be more difficult to detect such anomalies. Automated means of detecting and compensating for such performance anomalies are strongly required.

(3) In addition, large-scale supercomputers require constant monitoring by tens to hundreds of thousands of sensors, as well as proactive means for allocating resources to its numerous simultaneous users with extremely large-scale requests---users may submit tens of thousands of jobs at the same time, for example. If any portion of the job allocation algorithm embodies $O(n^2)$ behavior, then scaling of the machine would be catastrophic---a 100 times scaling would manifest in 10,000-fold increase in overheads.

By all means, it is important not just for operations, but also from academic Computer Science perspective, to determine how much we would be on target with the designed performance, and/or what the unexpected overhead would be. The computational science users would only benefit from the exercise, as various factors including performance but also reliability and usability at scale would greatly affect their actual usage.

With such a set of objectives, a series of large-scale, whole-machine benchmarks were conducted just before the operational commencement of TSUBAME2.0 on November 1, 2010, mostly throughout late October just after the machine was born:

(a) Linpack---the benchmark employed by the famous Top500[1] supercomputer performance ranking. Basically, it computes the LU-decomposition of a very large dense matrix. For an n-by-n matrix, the computational complexity can be given as $2/3\,n^3 + O(n^2)$. This means that, the larger we can make the problem, up to the point where the entire matrix fits within the memory of the supercomputer, the more efficient the

computation becomes due to the communication and other costs becoming relatively minimized. As a result, top-level supercomputers employ extremely large matrixes namely n=several million, and subjects the CPUs/GPUs to the ultimately high workload for a long duration. The solution algorithm is delicate with no margin of error---even a single numerical error for its total $O(10^{19\sim20})$ operations would result in an error in the residual check, which would nullify the entire result. On the other hand, since the communication complexity being relatively low being $O(n^2)$, a reasonable supercomputer network would incur less than 10% network overhead. An extremely slow network such as the Gigabit Ethernet would end up having the network cost being dominant. Finally, memory bandwidth requirements are also fairly low.

Although there is no "official" software for LU-decomposition, most supercomputers use the HPL(High Performance Linpack)[2] which was designed for large parallel machines. However, on TSUBAME2.0, we employed a set of heavily customized versions of HPL for heterogeneous GPU-CPU computation. One was the *Heterogeneous Linpack* that we had been developing on Linux over many years at the Matsuoka Laboratory, and other was the Windows HPC Linpack which was the result of our collaboration with and developed by Microsoft, and naturally running on Windows HPC. The two programs took a very different approaches to cope with GPU/CPU heterogeneity, and it was quite interesting to determine which would be advantageous for the future advancement of GPU computing.

Also, we conducted precise measurement of power consumption of TSUBAME for being ranked high on the Green 500[4], as the most important criteria of TSUBAME2.0 was to become ultimate green supercomputer of the time.

(b) GPU Version of ASUCA---ASCA is the next generation weather forecast code for extremely large machines, being developed by Japan's Meteorological Agency. In collaboration with the Agency, a group at Professor Aoki's laboratory succeeded in full porting of ASUCA on a multi-GPU heterogeneous supercomputing environment [5][6]. The principal computational kernel of ASCA is a finite difference transport code, requiring extremely high memory and network bandwidth, quite contrasting to Linpack. Previously, such high-bandwidth application had been perceived to be best served by custom-design, high-end vector supercomputers; so it was deemed important that such code would perform extremely well on TSUBAME2.0, whose chief compute element, GPU, embodies extremely high bandwidth as a modern-day vector processor Indeed we expected ASUCA on GPUs to achieve world's top-level performance on TSUBAME2.0, as the theoretical memory bandwidth of TSUBAME2.0 is approximately six times greater than the Earth Simulator, and ASUCA on GPU was demonstrating to be quite efficient and scalable on our preliminary tests on TSUBAME 1.2. Such tremendous performance was expected to allow real-time weather prediction at unprecedented resolution and precision.

(c) Also, a set of benchmarks were performed as a part of acceptance test of TSUBAME2.0.

2 TSUBAME2.0 Linpack ---"The Greenest Production Supercomputer in the World"

The whole-machine benchmarks were commenced in mid-October 2010, immediately after the initial deployment tests of TSUABME2.0 were completed. We first commenced the Linpack efforts spearheaded by the two teams running two different heterogeneous Linpack programs as described above in (a). Due to the unprecedented load imposed on the machine not possible on initial tests, we found a number of minor problems as expected, and resolved the issues one by one to attain stability. The Linux team at Tokyo Tech. and the Windows HPC team sent from Microsoft took turns running their respective benchmarks. Both were very closely matched, but in the end the Linux team's heterogeneous edged the latter (Figure 1). It is important to note however that, under slightly different conditions it would have been be quite possible that the results could have been the opposite.

As a result, TSUBAME2.0 recorded 1.192 Petaflops, achieving approximately 52% of the theoretical peak performance. This is lower than the typical 70-90% achieved by Linpack on CPU-based supercomputers. However, it is not technically correct in simply assuming that GPU-based machines are inherently less efficient compared to CPU-based ones. For our particular case, the lower efficiency is due to combination of the following performance degradation factors:

1. Firstly, the current NVIDIA Fermi GPU as employed in TUBAME2.0 embodies a set of design bottlenecks that are not fundamental to GPU computing but rather results of particular design decisions. Although sufficient for graphics as well as for high-bandwidth applications where the GPUs are being used in the similar manner as traditional vector processors, for dense matrix multiply (Level 3 BLAS) which is the principal kernel of Linpack, we only achieve 70-75% efficiency. This is substantially

```
================================================================================
- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
     ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR15R2R16     2490368  1024    59    69            8639.84              1.192e+06
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0008911 ..... PASSED
================================================================================
Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------
End of Tests.
================================================================================
```

Figure 1 TSUBAME2.0 Linpack Execution Output.
Here we see that the Linpack run involved the matrix of n= 2.5 million squared elements , and the run was completed in 2.4 hours, resulting in 1.192 Petaflops which ranked TSUABME2.0 to be fourth fastest in the world on the Nov. 2011 edition of the Top500. Notice that the residual computation is within the proper error bounds which is a required property of the run.

lower than the efficiency achieved by CPUs that exhibit more than 90% efficiency. However, with architectural as well as algorithmic improvements we expect future GPUs to match CPU efficiency in this regard, if not greatly exceed it.

2. For our Heterogeneous Linpack on Linux, CPUs are not utilized for BLAS kernel computation; however, the Top500 results mandate us to incorporate the CPU peak performance in determination of the theoretical peak performance of the machine. For TSUBAME2.0, the two CPUs on each compute node (Intel Xeon Westmere 2.93Ghz) constitute approximately 8% of the peak performance, which we lose when we compute measured versus theoretical performance ratio. By all means we could conceive an algorithm that does utilize the CPU, and in fact we did so for TSUBAME1.2[3], but that particular version proved to be less efficient due to various issues such as load balancing.
3. Our Heterogeneous Linpack effectively utilizes GPUs as a matrix multiply engine, where we send the sub-matrices of the matrix which is stored in CPU memory in stream pipelined fashion, perform the multiplication, and stream the matrix back. For normal applications where we typically transfer data in bulk to the CPU, dense computing algorithms such as matrix multiply where the compute overhead is $O(n^3)$ as opposed to the transfer overhead being $O(n^2)$, enlarging the matrix size n for pragmatic applications would hide most of the transfer latency. However, for HPL the sub-matrix size is rather small, with n being 100~1000, resulting in non-negligible transfer overhead.

The combination of all the factors above results in approximately 30% overhead. With advances in the GPU/CPU architecture as well as algorithms to utilize them efficienctly we believe that the overhead could be effectively eliminated. By all means they are subject of future research.

On the 36th edition of the Top 500 which was announced during IEEE/ACM Supercomputing held in New Orleans, USA during November, 2011, TSUBAME2.0 was ranked to be number 4 in the world; this was higher than TSUBAME1.2's initial appearance in June 2006, which had been 7th in the world. The exhibited 1.192 Petaflops was six times greater than the second ranking machine in Japan. Moreover, on the Green500, which ranks supercomputers based on their power efficiency, the average power consumption of 1243.80KW during the Top500 run resulted in 958.35 Flops/W, which ranked TSUBAME2.0 to be second in the world on its initial November announcement. More importantly, TSUBAME2.0 was recognized to be the "Greenest Production Supercomputer in the World" (Figure 2), as other top machines on the Green500 were largely prototypes in nature.

The high rankings of TSUABME2.0 on both lists simultaneously have an important technological significance. According to the current rules, it is difficult for a machine to be high on both lists; in practice, the top supercomputers of the Top500 are extremely large-scale production supercomputers, whereas the top ranks of the Green500 are smaller-scale prototypes and/or special-purpose machines (Figure 3). Only TSUBAME2.0 is ranked within world's top five on both lists. Here are why such difficulty exists:

(1) Since the top supercomputers on the Top500 are large scale, general-purpose production machines worth 10s to 100s of millions of dollars; as a result they typically embody numerous elements that are necessary for production runs but will be detrimental to power efficiency. For example, such machines incorporate 100s of terabytes of memory which is necessary for practical high bandwidth/memory applications, but does not contribute much to increasing the performance of Linpack. On large machines DRAM power consumption could be as much as 20-30% of the entire machine. On the other hand, to shoot for power efficiency on Linpack the best strategy would be to have rather small memory, but this would limit the scope of the machine to a very small number of specialized (typically compute intensive) applications.

(2) The top ranks of Top500 often is of extreme logistical importance for computing centers and even countries, and as a result, all other factors could become sacrificed just to go up a notch in the rankings, including power efficiency in the Linpack algorithm and settings. On the other hand, if one would shoot for top ranks of the Green 500, going down in rankings on the Top500 would not matter---all that matters would be that the machine being on the Top500. Such difference of objectives are difficult to coexist especially at the tops of each list.

(3) The Top500 runs of Linpack involve matrix size of n=a few million, with tens of thousands of processor cores. Unfortunately, by the nature of the algorithm Linpack is inhererently more efficient on smaller machines. For example, one typically sees 5-10% drop in performance just by going multi-node, with increasing overhead with large to machine size [3].

Despite such disadvantageous, TSUBAME2.0 being ranked highly on both lists was the reason for the award in Figure 2. By all means this was not achieved by simple employment of GPUs, as other GPU machines did not achieve similar results. Rather this was the result of years of basic research on low power, high performance computing at Tokyo Tech. GSIC, including the JST-CREST Ultra Low Power HPC (ULP-HPC) project.

Figure 2 November 2011 Green 500 Special Award for "Greenest Production Supercomputer in the World"

TSUBAME2.0 World Rankings
(Nov. 2010 Announcement Top500 and Green500)

The Top 500
Absolute Performance

#1	2.56 PetaFlops	China Defense Univ. Dawning Tianhe 1-A (11)
#2	1.76 Petaflops	US ORNL Cray XT5 Jaguar (81)
#3	1.27 PetaFlops	China Shenzen SC Nebulae (13)
#4	**1.19 PetaFlops**	**Japan Tokyo Tech. HP/NEC TSUBAME2.0 (2)**
#5	1.054 PetaFlops	US LLBL Cray XE6 Hopper (30)
#33 (#2 Japan)	0.191 Petaflops	JAEA Fujitsu (95)

(Green500 rank)

The Green 500
Performance/Power Efficiency

#1	1684.20	US IBM Research BG/Q Prototype (116)
#2	**958.35**	**Japan Tokyo Tech/HP/NEC TSUBAME 2.0 (4)**
#3	933.06	US NCSA Hybrid Cluster Prototype (403)
#4	828.67	Japan Riken "K" Supercomputer Prototype (170)
#5-7	773.38	Germany Julich etc. IBM QPACE SFB TR (207-209)
#2+	1448.03	Japan NAO Grape-DR Prototype (383) (Added in Dec.)

(Top500 rank)

Figure 3
The top ranking machine of November 2010 Top500 and Green500 and their corresponding rankings on the other list. (At the very end of 2010 a revised list which ranks Japan NAO's Grape-DR to be ranked 2+ on the Top500 at 1448.03 and Top500 being 383; this did NOT change the status of TSUBAME2.0 being the most power efficient production supercomputer in the world.)

Memory and Network Benchmarking in a Real Application
--- GPU Porting of ASUCA Weather Code and its World Record Performance

3

Although Linpack/Top500 is a significant metric for supercomputer performance measurement, for numerous applications that are largely memory or network bandwidth bound the Top500 numbers are not effective metrics. That is to say, for many important simulation applications such as computational fluid dynamics, structural simulations, and even modern apps such as Internet page rankings, how much effective bandwidth is achieved governs the overall performance, not how much Flops, and as a result, the baseline availability of theoretical peak bandwidth as well as the ease at which major fraction of the peak bandwidth could be achieved, become the dominant performance factor. Unfortunately, in recent supercomputer architectural trends, the amount of available bandwidth relative to the machine size, as well as its ratio to compute, is on constant decrease due to various physical limitations. Vector supercomputers of the past, such as the Cray X series and the NEC SXes were built precisely for this purpose in mind, i.e., increase the memory bandwidth during the days where achieving high compute flops was technologically difficult, resulting in high computational efficiency as a result.

Such glory days are over, and in fact we must really re-think our notion of efficiency for modern machines in terms of efficiency over the dominant performance bottlenecks, in this case the memory/network bandwidth, but not compute. So, the true "efficiency" metric is how much the application is utilizing the memory/network bandwidth in the system relative to the theoretical peak available, and has no correlations to the peak FLOPS of the machine.

In fact, in this regard we must point out that high computational efficiency that was apparently achieved in such classic vector machines were rather artificial and misleading, the result of technological trend of the times, and is not an effective metric in the modern times. That is to say, for such vector machines that under-provisioned the computing resources do not properly exploit the available opportunity presented by dense problems and the available locality (e.g., due to the lack of cache memory), and as a result, its efficiency might seem artificially high, but the resulting absolute performance being relatively low in relevance to the size/cost of the machine.

Exactly the same argument applies to GPUs versus CPUs, but often the same mistake mistakes are made. As had been mentioned in part one of this article, GPUs exhibit extremely high memory bandwidth per socket compared to CPUs, but at the same time, also embody much higher (and effectively overprovision) compute as well. In fact, on TSUBAME2.0, the per-socket peak compute capability of each GPU is approximately 7 times that of CPU, but at the same time, measured effective memory bandwidth is also 6-7 times per socket. So if CPU-based implementation on TSUBAME2.0 would be obtaining 5% of peak, then it is likely that a GPU-based implementation would be obtaining similar computational efficiency, while being 6-7 times faster with the same number of sockets.

This in effect allows us to largely conjecture that TSUBAME2.0 would be comparable to x86 CPU-based machines of similar peak performance, or those with socket counts that are 7 times greater, if no artificial bottlenecks are imposed as was the case for Linpack. That is to say TSUBAME2.0 with 4200 GPU and 2800 CPU sockets would be roughly equivalent to a 30,000 socket / 200,000 core x86 CPU-based supercomputer, which would largely equal the size of ORNL Jaguar. In fact we can expect that Jaguar would have advantage in compute but might not perform as well on high bandwidth code, as the older-generation x86 it employs is much less efficient in memory but similar in FLOPs utilization.

The ASUCA benchmark was important in this regard to determine if such performance estimations would hold for high-bandwidth applications. In particular, since finite difference solvers for transport codes are known to be mostly pure memory-bandwidth limited, the issues are whether (1) GPUs would be able to efficiently utilize the available memory bandwidth, (2) whether we could achieve 6-7 times speedup per socket as discussed above, and finally (3) how the performance would compare as a whole machine to Jaguar in executing the same or at least very similar weather code.

For details of the ASUCA code itself the readers are referred to [5]; the results achieved largely confirmed our conjectures (1)-(3) above. The GPU version of ASUCA on TSUBAME2.0 scaled up to 3990 GPUs[6] almost linearly in weak scaling (the problem size proportionally increasing relative to the machine size), and achieved 145 TeraFlops in single precision, and 76.1 TeraFlops in double precision (Figure 4). The per-socket performance is approximately 6 times that of CPUs, confirming our conjecture in real, production-level code.

Moreover, the previous world record holder was the WRF code, a similar weather code to ASUCA, on Jaguar at approximately 50 TeraFlops (double precision); so ASUCA was even faster, by a factor of 10 on a socket-to-socket basis, we might attribute this to less efficient memory bus of the older-generation AMD processors in Jaguar, but since the applications are different, this result should be taken as preliminary, and more rigorous benchmarking should be done using the same applications under a controlled environment.

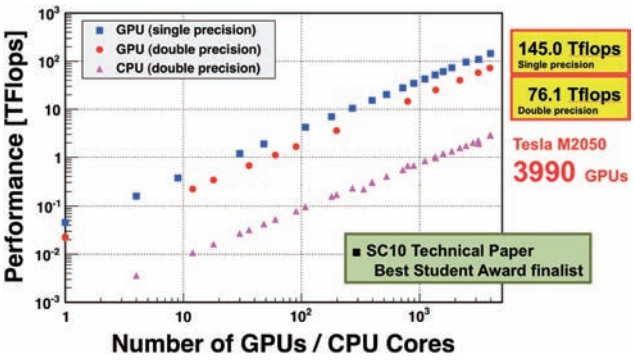


Figure 4 ASUCA Benchmarking on TSUBAME2.0 (1). Notice that performance scales almost linearly to 3990 GPUs.

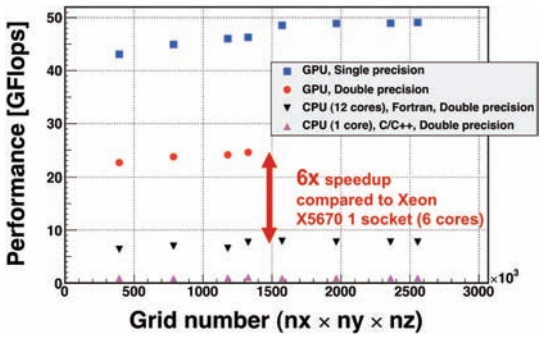


Figure 5 ASUCA Benchmarking on TSUBAME2.0. The per-socket performance difference is approximately factor of six, confirming the relative difference in peak achievable memory bandwidth.

Final Words : Let More Petascale Applications Begin! 4

There were several other benchmarks that were run on TSUBAME2.0, including I/O, to confirm that the machine was running correctly, after which the production operation began in early November, 2010 as planned. At the time of this writing numerous users are using TSUBAME2.0 daily imposing high loads. Large-scale applications that utilize 5000 CPU cores or 1200 GPUs (that are maximally allocatable under normal operations) are fairly commonplace. Applications that utilize other aspects of the system for large-scale data processing, such as the SSD and the LUSTRE parallel filesystem, are also substantially increasing in number. Combined, the performance and scaling leap from TSUBAME1.2 is very apparent, and users seem to be enjoying the massive capability and relatively forgiving characteristics of TSUBAME2.0

However, to best utilize the capability of TSUBAME2.0, one must utilize its most advanced features. For example, in order to best utilize the available memory bandwidth the use of GPUs are a must, but not only this involves programming in extension of C namely the CUDA language, but also involves fairly sophisticated hybrid programming involving MPI and CUDA in CPUs and GPUs. Also GPU performance is more sensitive to algorithms, programming methods, and data layouts, mandating more careful tuning process to optimize the performance. This also applies to other aspects of the system. Fortunately for TSUBAME2.0 networking is less troublesome as the network is configured as a full-bisection network, allowing the compute nodes to be much less sensitive to its placement, but for I/O effective hybrid use of SSDs and LUSTRE is sometimes required.

From a broader perspective, supercomputers composed from CPUs that consists of large many-core / multithread / vector processor, combined with a small number of low latency scalar cores in proximity, interconnected with multi-rail high-bandwidth network and I/O capabilities would be the next trend in large scale supercomputer as a continuum to TSUBAME2.0 architectural trend. By all means in the not so distant future the two types of cores would be bonded on the same die, sharing memory. Such would alleviate many of the complexities as currently being experienced in hybrid architectures such as TSUBAME2.0 today. This would allow tremendous number of applications scale to be petaflop-class, be it compute or memory bound, as we have seen for Linpack and ASUCA, but with much less effort. In effect, in the most advanced large-scale GPU applications we are observing

the future, where petascale performance is being achieved with substantial effort, but will be quite the norm, allowing high-end science and engineering to progress much faster than in the past. We will continue to strive in TSUBAME2.0 to demonstrate such possibilities, and prepare for TSUBAME3.0 to be designed and become operational in 2014, with possibly yet another great leap in performance.

References

[1] The Top500 Supercomputing Sites http://www.top500.org/
[2] A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary. "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers v.2.0", http://www.netlib.org/benchmark/hpl/
[3] Endo, T.; Nukada, A.; Matsuoka, S.; Maruyama, N. "Linpack Evaluation on a Supercomputer with Heterogeneous Accelerators", Proc. IEEE Parallel & Distributed Processing (IPDPS) 2010, the IEEE Press, Apr. 2010, pp.1-8.
[4] The Green 500 http://www.green500.org/
[5] Takashi Shimokawabe and Takayuki Aoki. "Multi-GPU Computing for Next Generation Weather Forecasting", The TSUBAME E-Scinece Jounal, vol. 2, GSIC Center, Tokyo Institute of Technology, Nov. 2010, pp.11-15.
[6] T. Shimokawabe, T. Aoki, C. Muroi, J. Ishida, K. Kawano, T. Endo, A. Nukada, N. Maruyama, and S. Matsuoka. "An 80-Fold Speedup, 15.0 TFlops, Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code", Proc. 2010 ACM/IEEE Supercomputing, The ACM Press, Nov. 2010, pp.1-11.

GPU Computing for Interstellar Atomic Hydrogen Turbulence

— 11.5 TFlops with 120 GPU on TSUBAME 1.2 —

Takayuki Muranushi *
*The Hakubi Center, Kyoto University

We have performed three-dimensional hydrodynamic simulations to study the thermal instability of the interstellar medium. The instability is powered by phase-transition of the atomic hydrogen gas and is one of the sources for turbulence in the interstellar atomic hydrogen gas. We have also performed clump-detection and spectra analyses, as well as three-dimensional visualizations. The analyses revealed multi-scale turbulent cascade connecting the supersonic compressive turbulence at the large scale to the well-known Kolmogorov turbulence. Such finding was enabled by the massive computational power of the GPU cluster supercomputers.

Introduction 1

My favorite word to describe our profession is "armchair astronaut." Human beings have sent astronauts to the Moon and unmanned probes to the end of the Solar System. But to visit the neighboring stars and collect the evidences will remain a very hard job for next generations. To overcome the limitations and understand the mystery of the universe, we have to collect the observational evidences sitting on the Earth, and carefully construct the reasoning based on the scientific laws. Computer simulations are powerful tools for us. They help us understand how the rich structures are formed out of the simple laws, and how the structures will look like to our limited eyes. The beautiful visualizations of today's advanced simulations make us feel like travelling the universe while sitting on armchairs.

What we have studied using TSUBAME is one of the processes that will determine the mass of the stars. Stars are born from fragmentation and condensation of the interstellar gas in the galaxy, which is as thin as one Hydrogen atom per cubic centimeter. If the typical stars are ten times lighter, the nuclear fusion won't ignite and our galaxy will remain dark. If they are ten times heavier, they will much more rapidly burn Hydrogen than they do today and our galaxy will be filled with black holes. The mechanism that sets the adequate initial masses for the stars --- is the interstellar turbulence driven by the thermal instability.

Interstellar hydrogen has two stable phases determined by the balance of various heating and cooling processes such as star irradiation and molecular line emission. Triggered by supernova shock waves, the interstellar medium makes phase transits from the warm, low density phase to numerous clumps of the cold, high density phase. It takes several more stories until stars and planetary systems are formed from these clumps. There are lots of detective works left to do in the universe.

On the JHPCN program 2

First, we would like to mention the JHPCN Program (Joint Usage / Research Center for Interdisciplinary Large-Scale Information Infrastructures) through which we gained access to the machine located in Tokyo Institute of Technology. JHPCN is a network consisting of eight university-owned supercomputer facilities in Japan. We've been using TSUBAME since 2009 through this program.

We applied to the JHPCN program as the joint research between Takayuki Muranushi at The Hakubi Center, Kyoto University and Tsuyoshi Hamada at Nagasaki Advanced Computing Center, Nagasaki University. Hamada leads the development and operation of GPU cluster DEGIMA (DEstination of GPU Intensive Machines). Also there is a small GPU cluster TenGU (Tenmon GPU Cluster) in Kyoto University. It was useful to have access to these different types of computers for different stages of the code development. Joint research brought helpful communications with the researchers of varying fields for code development and research. Also we had many technical, administrative and clerical supports from Tokyo Institute of Technology and JHPCN staffs.

Simulating the Interstellar Medium 3

We have prepared several themes at the beginning of the JHPCN program. In this section we will describe one of the themes, "Numerical Simulation and Analysis of The Two-Phased Hydrogen Gas Turbulence in Molecular Cloud Forming Region." As introduced in the first chapter, this study contributes to the

GPU Computing for Interstellar Atomic Hydrogen Turbulence

— 11.5 TFlops with 120 GPU on TSUBAME 1.2—

study of the interstellar medium, one of the various themes in astrophysics.

In the galaxy compressive and anisotropic turbulences of the interstellar medium are observed. Considering the radiative equilibrium, the atomic interstellar hydrogen gas has bistable equations of state. It is much different from that of the ideal gas and has two stable phases, differing hundred times in density (Field et al, 1969[1].) However, observed interstellar turbulence is described by the homogeneous, isotropic and incompressive Kolmogorov turbulence spectra (power law index of the velocity field α_v =11/3.) Still, some observations report significant deviations from the Kolmogorov turbulence (α_v=3.87 ±0.11) (Chepurnov et.al, 2010[2].) Understanding this turbulence has been one of the grand challenges for interstellar physics, requiring simulations of the resolutions of the order 1000^3. The use of GPU computer lets us simulate the turbulence for time much longer than before keeping the high resolutions, and draw out more detailed information on the turbulent statistics.

The basic equations to be simulated are the following Navier-Stokes equations coupled with the heating and cooling formulae.

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \mathbf{u}) = 0$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \bullet (\rho \mathbf{u}\tilde{\mathbf{u}}) + \nabla P = 0$$

$$\frac{\partial E}{\partial t} + \nabla \bullet ((E+P)\mathbf{u}) = \Gamma(\rho,T) - \Lambda(\rho,T)$$

For the heating term Γ and the cooling term Λ, numerous heating and cooling processes contribute (e.g. Koyama & Inutsuka, 2000[3].) Based on that, we use the following fitting formula proposed by Inoue & Inutsuka (2008)[4]:

$$\Gamma(\rho,T) = 6.802\times10^{-2}\rho$$

$$\Lambda(\rho,T) = \rho^2\left[3.871\times10^{5}\exp\left(\frac{-19.25}{T+0.1626}\right) + 4.25\times10^{-2}\sqrt{T}\exp\left(\frac{-1.496\times10^{-2}}{T}\right)\right]$$

Quantities are nondimensionalized based on the following three scales typical to the interstellar phenomena:

$$\text{Density}: \quad \rho_w = 1.211\times10^{-21} \quad [\mathrm{kg\,m^{-3}}]$$

$$\text{Pressure}: \quad p_w = 4.832\times10^{-8} \quad [\mathrm{Pa}]$$

$$\text{Length}: \quad l_w = 1\,\mathrm{parsec} = 3.086\times10^{16} \quad [\mathrm{m}]$$

This means that the sound speed of the gas is about 10km per second, as fast as the speed of a rocket escaping the Earth. Still, it takes about one hundred thousand years to cross a parsec (approximately 3.62 light years) with this sound speed. The simulation region (Figure 1) was 20 parsec each side, and we have simulated eight sound-crossing times to have the enough data to study the statistics of the turbulence. The total simulation corresponds to whopping sixteen million physical years. It is, still just an instance in the cosmological scales of time.

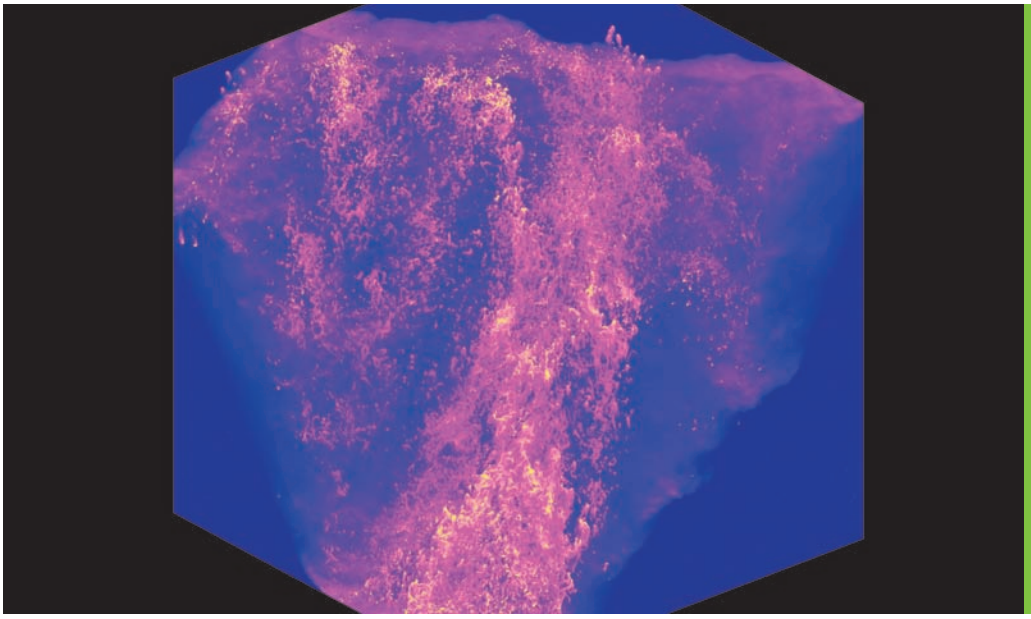


Figure 1
A visualization of the 1440^3 simulation performed on TSUBAME GPU Cluster at Tokyo Institute of Technology.

Our code can be classified like "MPI GPU Full-Godunov 2nd order MUSCL 3-dimensional uniform mesh Navier-Stokes equations solver". Godunov method is proposed by a Russian Mathematitian Godunov(1959) [5], and its higher-order extensions are established by efforts of e.g. van Fig. The idea of Godunov's method is to apply the analytic solutions of the Riemann problems (hydrodynamic problem starting with two constant initial conditions separated by a wall) to each mesh boundary. This is necessary treatment to sharply resolve the shock fronts. The second order version costs about 3000 floating operations per mesh update. Here, GPU's high computation capability comes in handy.

Of course this code can handle multiple GPUs via MPI framework. We have implemented checkponiting capabilities, to overcome the accidental abort of the computing devices and also overcome the time-limit applied to the job-queues on supercomputers. We have also implemented irreversibly compressed data outputs for movie visualizations.

We have carried out most of our code development and tuning on DEGIMA. On DEGIMA about 800 NVIDIA GT200 GPU chips are available, and 576 of them was connected via InfiniBand.It has 514.9Tflops single precision peak performance, and can handle 1769'4720 threads at most. Total amount of the video memory was 460GB, total bandwidth was 64.454TB/s. On DEGIMA we have recorded 40.91Tflops sustained performance on 1280^3 resolution simulations.

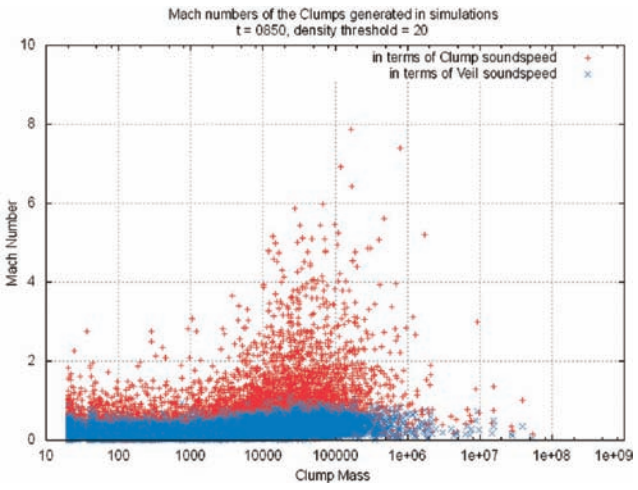


Figure 2
The Clump Detection Analysis.
The horizontal axis is detected clump mass.
The vertical axis is the clump speed relative to its surroundings (veil) in Mach numbers. We have found that clump motion is supersonic in terms of its own sound speed, but is subsonic in terms of veil sound speed.

Next, we have carried out long-run cariculations using GPU supercomputer TSUBAME. We have reserved TSUBAME three times: July 6-9, July 20-23, August 20-24. The hpc1tes2 queue we have used consisted of 120 NVIDIA GT200 GPU chips and had 124.2Tflops single precision peak performance. The maximum number of threads available was 122'880. Total amount of the video memory was 480GB and its bandwidth 1.224TB/s. It costed 132'000 yen. Thus we obtained 1440^3 resolution simulated data for about 8 sound crossing times, sufficient for the statistical analyses of the turbulence. Also we obtained about 200 snapshot data for the movie. The sustained performance was 11.5Tflops.

Analyses and Visualization 4

Because we have achieved so high resolutions, the analyses and visualization of the data became challenging tasks and we had to develop new tools for them. For example, each of the snapshot data was about 60GB, which is larger than the memory capacity of typical personal computer. So we have invented algorithms that consume memory only proportional to the 2-dimensional cross section of the data, and make minimum number of access and only sequential access to the disk, while performing the desired tasks.

For example, in clump detection analysis we need to detect high density regions of the gas (Clump) and its surrounding media (Veil), calculate their physical properties such as density, sound speed, relative motion, and perform statistical analysis on them. A naïve implementation for connected component finding algorithm in three-dimensional grid will require random access on the 3D grid. We have developed a new algorithm that performs the above analysis within only (constant)×(two-dimensional cross section) memory and twice sequential access of the whole data. The algorithm enabled clump detection analysis in practical time on PCs while keeping the main data on the disk.

The analysis revealed that the Cold Neutral Medium (CNM) motion was actually faster than CNM sound speed. However, each CNM clump is surrounded by unstable medium. The CNM motion was slower than the sound speed of the surrounding medium. This suggests that incompressive, Kolmogorov aspect of the turbulence can be explained by the CNM motion subsonic in the unstable media.

To study the turbulence, we have performed spectra analyses of the 1440^3 simulation by the following algorithm. First, create two-dimensional projection for each of the three axes while scanning the snapshot once. Next, perform two-dimensional Fourier analysis of the velocity field. Finally, reconstruct the turbulent spectra using the same method used in reconstructing the astronomical observation. This algorithm shortened the time required in Fourier analyses to practical level and also meaningful in comparing the simulated data with observations.

The spectra of the simulated turbulence showed the three-stage structure that consists of supersonic turbulence regime, Kolmogorov-like turbulence regime, and numerical dissipation regime (Figure 3). Overall fit of the density and velocity power spectra showed significantly agreement with that of Chepurnov et.al. 2010's (α_v=3.87±0.11) (α_ε=3.0±0.1.) Supersonic turbulence regime has softer, and Kolmogorov regime has harder velocity power spectrum. These findings indicated that non-Kolmogorov turbulence observed like in Chepurnov et.al. may be explained as superposition of the supersonic and Kolmogorov turbulences.

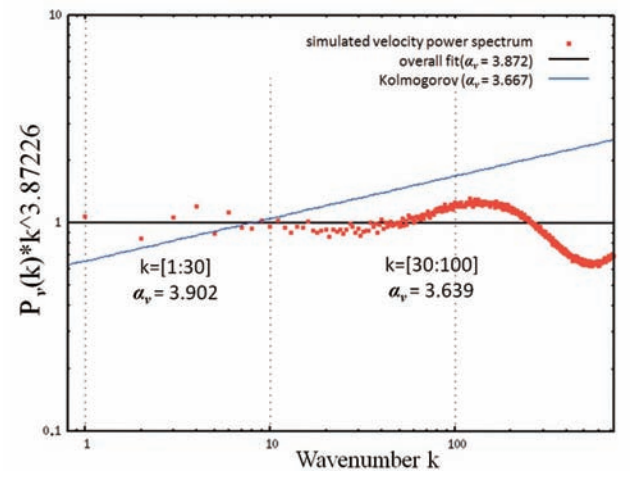


Figure 3 The result of velocity spectrum analysis.

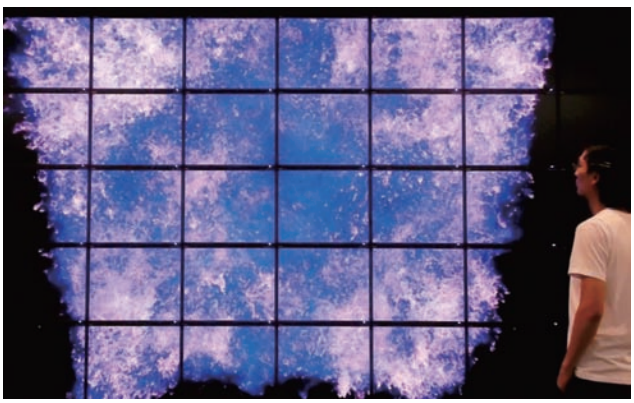


Figure 4 A visualization on 40-side tiled display in collaboration with Oyamada Lab. at Kyoto University.

To visualize the simulation as movies, we have implemented an renderer that can create perspective or parallel projection plots of various physical quantities with only one or several times of sequential access to the data. We can clearly see in the movie the violent displacement of the shock front and episodic penetration of streaming flows (Figure 1.)

Also in collaboration with Oyamada Lab. in Kyoto University, specialists of large-scale data visualization, we have made large and high-resolution visualization using the 40-side tiled display (Figure 4). We have also created a movie of our simulation on TSUBAME with Prof. Aoki's help for use in advertisement of TSUBAME.

We are continuing the research on this theme, changing conditions and performing more analyses, to draw out more physical knowledge.

From Hydrodynamics to Magnetohydrodynamics 5

Based on the experiences earned in the development and operation of the hydrodynamics code on multi-GPUs, we begun development of magnetohydrodynamic (MHD) codes from August. MHD is one of the basic equations to describe behavior of ionized gas, plasma. Magnetic field takes the essential roles in many active astrophysical phenomena including accretion disks, jet activities, the Sun, and the Earth magnetosphere. MHD is also used to control plasma e.g. in nuclear fusion reactors studies. Use of GPGPUs in magnetohydrodynamics will accelerate simulations, and make more high-resolution simulations possible in those fields of science. It will also help in engineering connected to our daily lives such as solar activity prediction and design of spacecrafts.

Current version of the code is second-order in space and time. We have adopted HLLD scheme by Miyoshi & Kusano 2005[7] for MHD Riemann solvers. As we develop the MHD code we have performed various tests, and confirmed that our code reproduces the known test results (Figure 5).

From the development experiences of the multi-GPU hydrodynamics codes, we have learned that simple C-like coding style will require too many repetitions of lines and

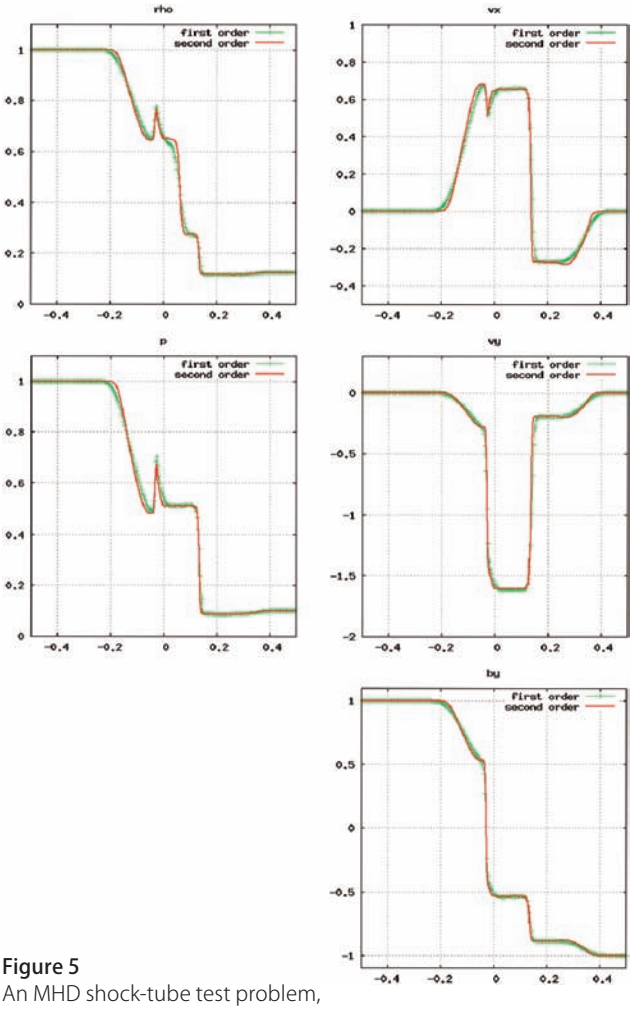


Figure 5
An MHD shock-tube test problem, corresponding to Miyoshi & Kusano 2005[7] Fig. 8. Equations of MHD is integrated upto t = 0.1 with initial conditions (ρ, p, v_x, v_y, v_z, B_y, B_z) = (1, 1, 0, 0, 0, 1, 0) on the left, (ρ, p, v_x, v_y, v_z, B_y, B_z) = (0.125, 0.1, 0, 0, 0, -1, 0) on the right, and B_x = 0.75. Each of the figure above contains 256 meshes.

inacceptable. CUDA has C++ capabilities; but we have also learned limitations of abstractions in C++ style. So we have designed Cprb (https://github.com/nushio3/cprb,) a simple code generator for C-like programming languages. Cprb is written in Haskell and one can meta-program C++ codes in ruby style. We are writing current version of MHD code in Cprb; it is very fluent. We hope that Cprb contribute a step in search for new schemes of parallel programming.

I would definitely like to solve the mysterious discharge phenomena in protoplanetary disks. Say, it's study of space thunderstorms that come after studying space clouds. I have in fact proposed a discharge mechanism powered by collisional charging of ice particles in protoplanetary disks (Muranushi, 2010 [8]) --- the mechanism similar to lightning on Earth. Also Inutsuka-Sano (2005) [9] proposed a discharge mechanism arising from elementary processes of Resistive MHD. The paper above treated lightning in a one-zone model, but recently Okuzumi has predicted three-dimensional instability arising from the mechanism. To study how these phenomena will be active and saturate, we need three-dimensional Resistive MHD simulations, possibly coupled with dust component and chemistry. I would continue to develop codes, making use of the advanced computing techniques, to meet such goals.

Acknowledgments

Takayuki Muranushi's research is supported by The Hakubi Center of Kyoto University. We are deeply grateful to Tsuyoshi Hamada, Takayuki Aoki and many others for useful advices on coding. We are also grateful to people working for JHPCN (Joint Usage / Research Center for Interdisciplinary Large-Scale Information Infrastructures) Program through which this work was made possible.

References

[1] Field G. B., Goldsmith D. W., Habing H. J., (1969), Astrophysical Journal Letters, 155, L149+
[2] Chepurnov A., Lazarian A., (2010), Astrophysical Journal, 710, 853
[3] Koyama, H., & Inutsuka, S. (2000), Astrophysical Journal, 532, 980
[4] Inoue, T. & Inutsuka, S. (2008), Astrophysical Journal, 687:303-310
[5] Godunov, S. K. (1959), Matematicheskii Sbornik, 47, 271-306
[6] van Leer, B. (1979) Journal of Computational Physics 32, 101-136
[7] Miyoshi & Kusano, Journal of Computational Physics 208 (2005) 315–344
[8] Muranushi, T. (2010), Monthly Notices of the Royal Astronomical Society. 401, 2641–2664
[9] Inutsuka & Sano, The Astrophysical Journal, Volume 628, Issue 2, pp. L155-L158 (2005)

Fast Fourier Transform using GPU

Akira Nukada*
*Global Scientific Information and Computing Center, Tokyo Institute of Technology

Fast Fourier Transform (FFT) is one of the most important computations used in various fields from multimedia to large-scale simulation.
Therefore, speed-up of the FFT computation benefits many people.
TSUBAME 2.0 compute nodes equips GPUs which accelerates
many kinds of computations. In this column, latest status of GPU FFT is presented.

1 Introduction

GPU has very high floating point operation performance as well as high memory bandwidth, which is a large advantage for memoryintensive computations. Indeed, many real applications in high-performance computing areas are categorized as memory-intensive computations.

2 Fast Fourier Transform

The Discrete Fourier Transform (DFT) is used in many fields of science and engineering. This is a transformation between physical or time space and frequency space. DFT is used not only in large-scale simulations such as Molecular Dynamics, but also in commodity multimedia applications like audio and video encoding/decoding.

DFT calculates N outputs $Y(k)$ from N input values $X(k)$ as described below:

$$Y(k) = \sum_{j=0}^{N-1} X(j) e^{-2\pi i j k / N}$$

$X(k)$ and $Y(k)$ are floating-point complex numbers. The computation of all outputs requires $O(N^2)$ floating-point operations. However, this computation includes many redundant operations when N is L multiplied by M. To clarify it, the formula is transformed as follows.

$$X'(k_1 + k_2 M) = \sum_{j_2=0}^{L-1} X(k_1 + j_2 M) e^{-2\pi i j_2 k_2 / L}$$

$$X''(k_2 + k_1 L) = X'(k_1 + k_2 M) e^{-2\pi i k_1 k_2 / N}$$

$$Y(k_2 + k_1 L) = \sum_{j_1=0}^{M-1} X''(k_2 + j_1 L) e^{-2\pi i j_1 k_1 / M}$$

The first line and third line correspond to L-point DFT and M-point DFT, respectively. The second line is called multiplication by twiddle factors. Thus, an N-point DFT can be divided into L M-point DFTs and M L-point DFTs and some additional operations. This is the base of Fast Fourier Transform (FFT) algorithm to calculate DFT efficiently. This transformation can be applied recursively. If N is a highly composed number, the number of floating-point operations can be reduced to $O(N \log N)$. In this case, the computation of DFT is now memory-bound.

Actually, FFT is well known as one of the most memory-intensive computation in typical benchmark applications. Each compute nodes of TSUBAME 2.0 has two Intel Xeon X5670 Processors (Westmere-EP 2.93GHz, six cores), which provides 32GB/s with triple-channel DDR3-1333 memory. However, this bandwidth is not sufficient for the computation of FFT.

TSUBAME 2.0 compute node has three GPUs (NVIDIA Tesla M2050) in addition to the CPUs. The memory bandwidth of the GPUs is about 150GB/s which is more than four times as much as that of CPUs. The GPU is a main computation resource of TSUBAME 2.0.

GPGPU 3

One of the latest trends in HPC is generalpurpose computation using GPU (GPGPU). Current GPUs have sufficient programmability to implement not only graphics operations but also more generic computations efficiently. Thus, now GPU is recognized as a powerful, inexpensive, low-power computation resource.

In 2006, one of the major GPU vendor NVIDIA introduced CUDA, which is a new GPU architecture and programming tools for GPGPU. Since it is much easier than prior conditions, many of compute intensive applications especially in HPC area were ported to GPU using CUDA and achieved drastic speed-ups.

In general, graphics operations do not require high accuracy. Therefore, GPU supported only single-precision floating-point operations. For this reason, porting applications to GPU is always involved with accuracy issues. But latest GPUs support double-precision floatingpoint operations which enables compatible computations with the latest CPUs.

The advantage of GPU computing is not only the high performance floating-point operations. GPU has also high memory bandwidth to transfer the data required in the computation. The memory bandwidth of the Tesla M2050 GPUs is about 150GB/s, which corresponds to more than four times as much as CPU. This will be a strong advantage for FFT computations.

Performance of FFT using CPU or GPU 4

Figure 1 shows the performance of FFT using a CPU or a GPU on TSUBAME 2.0 compute nodes. Performance is calculated in GFLOPS, where the number of floating-point operations required in computing an N-point FFT is assumed to be $5N \log N$.

We used Intel MKL Library 10.2.5 and FFTW 3.1.2 as FFT libraries using CPU. Both of them are installed on TSUBAME 2.0 system by default. To measure the performance, all of the six cores of a CPU are used by multi threading. For FFT library for GPUs, we used NVIDIA's CUFFT library 3.1 and our NukadaFFT library 1.0. The CUFFT library is provided as a part of the CUDA toolkit packages as well as BLAS library and so on.

Performance of FFTW library is almost the same as that of MKL library. Of course, their implementation is quite different from each other. However, the achieved performance is basically limited by the memory access.

On the other hand, the performance of NukadaFFT library is much higher than CUFFT library.
This means, the FFT routines in double precision is not well optimized yet. Actually, its performance in single precision is very high. Of course, the performance using GPU is much higher than that using CPU.

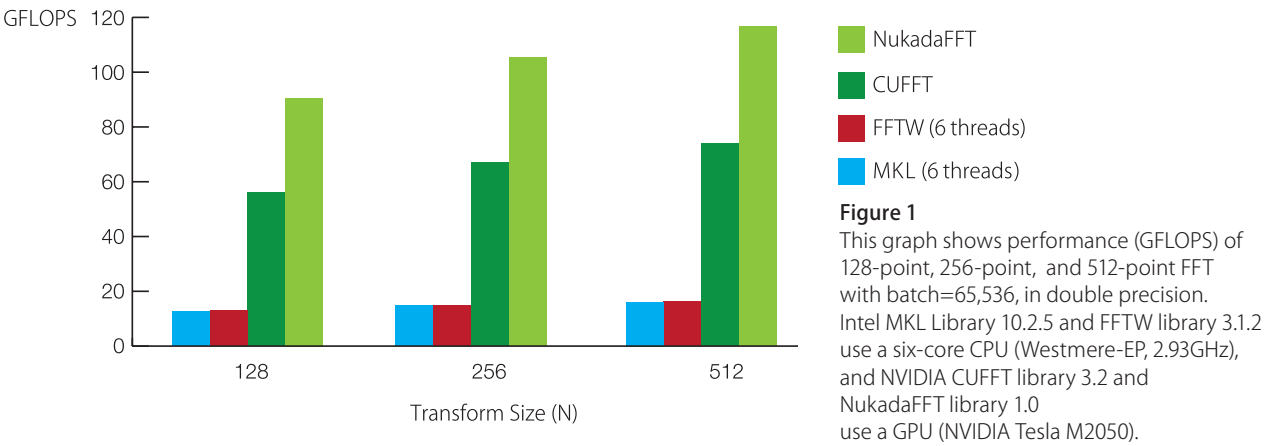


Figure 1
This graph shows performance (GFLOPS) of 128-point, 256-point, and 512-point FFT with batch=65,536, in double precision. Intel MKL Library 10.2.5 and FFTW library 3.1.2 use a six-core CPU (Westmere-EP, 2.93GHz), and NVIDIA CUFFT library 3.2 and NukadaFFT library 1.0 use a GPU (NVIDIA Tesla M2050).

In the case of TSUBAME 2.0 compute nodes, Tesla M2050 GPU has about 4.6 times theoretical peak memory bandwidth than CPU. However, the speed-up by using GPU reaches about seven times in FFT computation. This is because there is a large gap between theoretical peak and achieved memory bandwidth. The memory bandwidth of the GPU is about 150GB/s, however the achieved memory bandwidth in FFT is only 82GB/s. Since the memory of the Tesla GPU is protected by ECC, additional data transfer of ECC code consumes a part of the memory bandwidth. Writes to the GPU memory is slower than reads from the GPU memory. In case of FFT computation, about 50% of memory access is write access. This degrades the data transfer efficiency. This is common in the case of CPU. Simple memory copy operation can achieve slightly less than 20GB/s, and the data transfer ratio in FFT computation is less than 12GB/s.

Auto-tuning 5

There are many kinds of GPU products. The number of CUDA-capable GPU models already exceeds a hundred including GeForce series, Quadro series, Tesla series, ION series and mobile GPUs. Table 1 shows specificaions of some major GPU models we used. Although the performance of GPUs rapidly increase year by year, they vary in number of cores, performance, memory bandwidth, etc.

GeForce series are designed for gaming and desktop use, and have high memory bandwidth. On the other hand, Tesla series are designed for high performance computing. Their doubleprecision performance is much higher than GeForce series. The memory capacity is also larger, and protected by ECC. Needless to say, the Tesla series is selected for TSUBAME 2.0 compute nodes.

There can be many kinds of GPU implementation to compute FFTs. To achieve high performance, we have to implement programs for each transform size.

GPU model is another large factor. When we were developing the essential part of the NukadaFFT library, neither Tesla M2050 nor GeForce GTX 480 existed. The development was done on older-generation GPUs such as GeForce GTX 280. Now the library shows competitive performance even on next generation GPUs. This is because of the auto-tuning features of the library.

The auto-tuning strategy used in this library is exhaustive. The library generates many kinds of FFT routines, executes them, and selects the best one. We employ three tuning parameters as follows.

Factorization The key of the FFT algorithm to reduce the number of floating-point operations is the factorization of the transform size. Factorizing to prime numbers does not always show good results. In case of GPU implementation, it is better that the gap between the largest and smallest factors is small. Otherwise, the SIMDization for GPU wastes many computation resources.

Number of threads It is important to exploit the high memory bandwidth of GPUs. It depends not only on the access patterns but also on the number of threads running simultaneously. We need to choose the best number of threads.

Shared memory access pattern CPU implementation of FFT uses cache memory to store temporal data which will be used again immediately. GPU implementation uses onchip shared memory to exchange the data between threads. The shared memory can be accessed simultaneously by multiple threads since it consists of multiple banks. Therefore, we have to select the access pattern carefully to avoid a bank conflict. To do this, the library inserts padding automatically in specific patterns.

Table 1 The specifications of major CUDA-capable GPUs.

	GeForce 8800 GTX	GeForce GTX 280	GeForce GTX 480	Tesla M2050
Released	Nov. 2006	Jun. 2008	Apr. 2010	Jul. 2010
Number of Cores	128	240	480	448
Memory Capacity	768MB	1024MB	1536MB	3072MB
Double-Precision	N/A	77.8GFLOPS	168.1GFLOPS	515.0GFLOPS
Memory Bandwidth	86.4GB/s	141.7GB/s	177.4GB/s	150.2GB/s

Although the time spent for the auto-tuning essentially depends on the transform size, GPUs, CPUs, and so on, we observed it completes within a minute in most cases. Generally, applications repeatedly calls FFT functions with the same transform sizes. In such a case, the length of the auto-tuning time is acceptable enough. By default, the results of the autotuning procedure, that is, the best parameters selected will be stored in the database file. Therefore, applications start computation using them immediately them, except on the first time.

It is not true that GPU applications always require auto-tuning features to achieve high performance. In many cases, the GPU model used for the GPU application are limited and the same kernel can achieve good performance for all of them. Auto-tuning mechanism is important especially for libraries which will be publically available because these libraries should work on varying GPU models with acceptable performance.

Summary 6

We introduced the latest status of FFT computation using GPUs. In case of the CPUs and GPUs on TSUBAME 2.0 compute nodes, the use of GPU resulted in about 7 times speedup due to the high memory bandwidth of the GPUs. This is not only for FFT but also many kinds of memory-intensive computations like CFDs. Needless to say, GPUs also accelerate many compute-intensive applications. In the near future, we hope many new applications will be ported to GPUs.

NukadaFFT library is still updated frequently. The latest version is available at the following
URL. http://matsu-www.is.titech.ac.jp/~nukada/nufft/

Acknowledgments

This work is partially supported by Core Research of Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST) "ULP-HPC: Ultra Low-Power, High-Performance Computing via Modeling and Optimization of Next Generation HPC Technologies", by Microsoft Technical Computing Initiative "HPC-GPGPU: LargeScale Commodity Accelerated Clusters and its Application to Advanced Structural Proteomics", by NVIDIA CUDA Center of Excellence Program, and by MEXT Grant-in-Aid for YoungScientists (A) 22680002.

References

[1] James W. Cooley and John W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Comput. Vol. 19, pp. 297-301, 1965.

[2] Charles Van Loan, "Computational frameworks for the fast Fourier transform", SIAM Press, Philadelphia, PA, 1992.

[3] Matteo Frigo and Steven G. Johnson, "The Design and Implementation of FFTW3", Proceedings of the IEEE, Vol. 93, No. 2, pp. 216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

[4] Akira Nukada, Yasuhiko Ogata, Toshio Endo, and Satoshi Matsuoka, "Bandwidth Intensive 3-D FFT kernel for GPUs using CUDA", In Proceedings of the ACM/IEEE conference on Supercomputing (SC08), Austin, IEEE Press, Pages 1- 11, November 2008.

[5] Akira Nukada and Satoshi Matsuoka, "Auto-Tuning 3-D FFT Library for CUDA GPUs", In Proceedings of the ACM/IEEE conference on Supercomputing (SC09), Portland, ACM, Page 1-10, November 2009.

● **TSUBAME e-Science Journal No.3**

Published 02/25/2011 by GSIC, Tokyo Institute of Technology ©
ISSN 2185-6028
Design & Layout: Kick and Punch
Editor: TSUBAME e-Science Journal - Editorial room
Takayuki AOKI, Toshio WATANABE, Masakazu SEKIJIMA,
Thirapong PIPATPONGSA, Fumiko MIYAMA
Address: 2-12-1 i7-3 O-okayama, Meguro-ku, Tokyo 152-8550
Tel: +81-3-5734-2087 Fax: +81-3-5734-3198
E-mail: tsubame_j@sim.gsic.titech.ac.jp
URL: http://www.gsic.titech.ac.jp/

TSUBAME

International Research Collaboration

The high performance of supercomputer TSUBAME has been extended to the international arena. We promote international research collaborations using TSUBAME between researchers of Tokyo Institute of Technology and overseas research institutions as well as research groups worldwide.

Recent research collaborations using TSUBAME

1. Simulation of Tsunamis Generated by Earthquakes using Parallel Computing Technique
2. Numerical Simulation of Energy Conversion with MHD Plasma-fluid Flow
3. GPU computing for Computational Fluid Dynamics

Application Guidance

Candidates to initiate research collaborations are expected to conclude MOU (Memorandum of Understanding) with the partner organizations/ departments. Committee reviews the "Agreement for Collaboration" for joint research to ensure that the proposed research meet academic qualifications and contributions to international society. Overseas users must observe rules and regulations on using TSUBAME. User fees are paid by Tokyo Tech's researcher as part of research collaboration. The results of joint research are expected to be released for academic publication.

Inquiry

Please see the following website for more details.
http://www.gsic.titech.ac.jp/en/InternationalCollaboration

GSIC
Global Scientific Information and Computing Center