The Rise of the Commodity Vectors

Satoshi Matsuoka
Tokyo Institute of Technology
National Institute of Informatics / JST CREST

matsu@is.titech.ac.jp

Abstract: Commodity clusters revolutionized high performance computing in a fundamental way since its first inception, and now now dominate many of the world's premiere supercomputers on the Top500, growing to the scale of over 10,000 CPU cores and beyond. Still,"classical" specialized vector supercomputers still remain to be sold and facilitated, especially at high end of the market, largely due to the nature of some of the HPC workloads still requiring the computing power of vectors, in areas such as CFD, FEM with kernels such as FFT, characterized as mostly large-scale irregular sparse codes Finally, however, commoditization of vector computing is on the rise, lead by multimedia application requirements, and spurred many architectures to arise such as GPUs and the Cell processor. But various problems still remain by which we cannot claim with 100% confidence that commodity vectors are here to stay in the HPC space. Research and development has to be conducted at various degrees of intensity to utilize the new breed of commodity vector hardware to their fullest capabilities, just as various research were needed to harness the power and the scalability of commodity clusters. In the talk I will outline some of the details, and our recent research endeavors aimed at solving the various issues.

1. Introduction---the Rise of the Commodity Vectors

Commodity clusters revolutionized high performance computing in a fundamental way, Since its first inception (Wigraf) in 1994, it quickly spread and rode the rapid technology curve advances of commodity microprocessors, and now dominate many of the world's premiere supercomputers on the Top500 [1], growing to the scale of over 10,000 CPU cores and beyond.

The biggest cultural effect that were given rise by cluster computing was to bring HPC to the mainstream--- the building block components were those of commodity in nature, especially in the (albeit high-end) PC desktop & server ecosystem, and the architectural research on constructing a supercomputer now rested on how the components are pieced together using innovative software techniques. This is in stark contrast to the old days of customized supercomputers of the past---the Crays / NEC SXs / Fujitsu VPPs / ETA10 / … etc., that star-studded the supercomputing arena only a decade earlier.

Commoditization is very important from the industry ecosystem viewpoint, both for economy and sustainment of the infrastructure, as supercomputing now cannot be alienated from rest of the IT, both from a financial point of view as well as from a technical point of view. For example, software compatibility and longevity are very

important, from the software vendor's standpoint in market penetration and customer retention, and from the user's standpoint in availability/longevity, as well as continuum with the now complex software R&D environment he would be using every day with sophisticated GUIs instead of the arcane half-duplex TTY systems (or even punch cards in the good old days.)

Still, circa early 2008, one also notices that ``classical'' specialized vector supercomputers, and non-commodity CPUs that emulate them to some extent, still remain to be sold and facilitated, especially at high end of the market, by major supercomputing centers. These include the Cray Black Widow, the NEC SX-8/8R/9, and the high-end IBM Power servers. The existence is largely due to the nature of some of the HPC workloads still requiring the computing power of vectors, mostly those that require high bandwidth and amenable to vectorization. These include areas such as CFD, FEM with kernels such as FFT, characterized as mostly large-scale irregular sparse codes sometimes with serialization bottlenecks. One could say that, irregular sparse codes and their kernels are the last stronghold of vector supercomputing market, as performance parity could occur for those types of applications due to higher efficiency, and as a result, may lead to overall advantage for the classical vectors in the overall price, facility requirements, or application porting (for codes that had been optimized for vectors).

Finally, however, starting from a few years ago, commoditization of vector computing is on the rise, driven by the needs of multimedia, gaming, video, and other applications to offer richer UI experiences. Now, such capabilities are starting to be applied to non-graphical/media technical computing. This is synonymous to the first Beowulf cluster in 1994 where commodity desktop PC processors (Intel 486 CPUs at the time) were utilized to construct the first computing cluster. Despite that some of the earlier attempts especially. with GPUs had less than stellar performance, and/or the user having to deal with arcane programming model based on graphical pipelines, inherent advances to enrich the user experiences had lead to generalization of the overall architecture to be significantly resemble traditional many-core, multithreaded SIMD CPU in order to cope with very sophisticated rendering algorithms. As a result, recent hardware and software advances thereof are finally making the system to be tractable for a non-graphics layman in scientific computing, and the performance advantage potentials that are beginning to find substantive traction and impetus in the community. A recent IEEE article[2] gives an excellent overview of the history and the status quo of GPU computing, especially focusing on molecular dynamics that the one of the authors' group had been working on in the context of Folding@Home[3] and other apps. Similar can be said for non-graphical but multimedia-oriented CPUs such as the CELL Broadband Engine[4].

There are less commoditized but still fairly inexpensive and general-purpose SIMD-Vector accelerators on the market, such as the ClearSpeed Accelerator board[5], the ones in particular which are housed in Tokyo Tech.'s supercomputer TSUBAME, representing almost half of the peak computing power provided by our Global Scientific Information and Computing Center (GSIC). ClearSpeed has been the driving force of TSUBAME's ability to retain the top performing position as the Japan's No. 1 supercomputer on the Top500, due to several innovative works in heterogeneous Linpack algorithm that we had developed that allowed increase of performance over four consecutive rankings, the first of its kind in the world[6]. Still,

one could claim that ClearSpeed is situated in the realm of commodity space, as it employs standard interfaces to the PC (PCI-X and PCI-E), and as such could be added onto any commodity desktop PC---and its price range is on the high end of commodity platforms as well, in the low thousands of dollars as of 2008.

Some may also claim that we had gone the way before---certainly we remember the old days of Weitek vector add-ons for early RISC based MPPs such as the Thinking Machine's CM-5, Meiko CS-2, etc. There had also been dedicated accelerators that were more generic, such as Riken's Grape series such as the MD-Grape 2[7]. The differences being that, for the new breed of commodity vector accelerators above, both the hardware and the software are riding on the commodity ecosystem, whereas these older accelerators that were dedicated to particular hardware platforms were not. As a result, one could not, for example, easily "upgrade" the accelerator as the general CPU, and/or not being able to take advantage of a vast software infrastructures that were present, i.e., by the time the software would be customized to take advantage of acceleration, the hardware would be "caught up" with general-purpose CPUs such that it would effectively be rendered "obsolete". This emphasizes the importance of the acceleration being on the same performance growth curve as dictated by the Moore's law, but the only successful way of sustaining such a growth curve had been to ride on the commodity ecosystem.

2. Challenges in Commodity Vectors

So, it is our claim that commoditization of vectors are finally here, a phenomenon of such importance after numerous years of supercomputing such that it would propel HPC in the mainstream market--- over a decade of gradual commoditization of central pieces that effectively comprised supercomputers, starting from CPUs to memory systems, the operating systems and middleware, networks such as 10GbE and Infiniband, and finally onto vector acceleration. But as pointed out, albeit from rather graphics-oriented point of view in [2], various problems still remain by which we cannot claim with 100% confidence that commodity vectors are here to stay in the HPC space. In fact, we are not even settled on what would be the governing architecture, as various hardware chips differ considerably in the architectural organizations as of current, and their evolutions could even be more disparate, rather than being on the convergence paths. The associated software platforms are still nascent at best, a historical parallel to early days of older vector computing. In fact, some of the algorithms and the software artifacts could be applied straightforwardly, but those are fairly few in number. In many cases, research and development has to be conducted at various degrees of intensity to utilize the new breed of commodity vector hardware to their fullest capabilities, just as various research were needed to harness the power and the scalability of commodity clusters. In the talk I will outline some of the details, and our recent research endeavors aimed at solving the various issues, but here are some of the highlights:

1). *Combined Commodity Vector / CPU Algorithms and their Performance Models*---Commodity vectors such as GPUs are aimed mainly at stream processing, while CPUs are more general purpose and are more latency optimized, but still have significant stream processing power. Moreover, communication bandwidth between commodity vector components and the CPUs are currently

limited by the PCI-Express bus, whereby our measurements show that even for the most current generations being far inferior compared to naïve memory system performance (over 60GBytes/s for high-end GPU memory systems compared to 5GB/s for PCI-Express x16 Gen2). As such, devising an effective methodology as to how to divide the labor between the CPUs and commodity vectors in the system needs to be investigated. This is different from the classical vector days where algorithmic vectorization would be done gradually in a piecewise fashion in the performance critical portions of the code, whereby the merger between the scalar versus the vector processing came essentially for free.

Our recent work in this context mentioned earlier[6] comes up with a methodology and an algorithmic instantiation in Linpack whereby we virtualize the ClearSpeed accelerators as CPUs (Figure 1). This is possible when the workload can be described as divisible labor in which both CPUs and vectors could compute the same subspace in the workload, albeit with different performance characteristics. The improved scheme now allows other CPUs to be consolidated into the system, allowing us to achieve over 67 Teraflops for yet another performance-improvement Top500 submission for June, 2008.

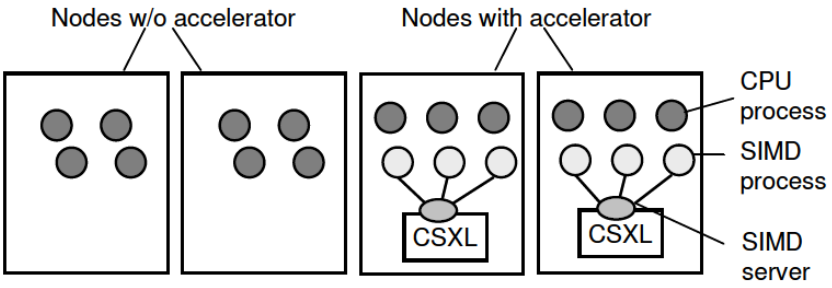

Figure 1: Overview of Virtualization of SIMD Vector Processing in our Heterogeneous Linpack Algorithm presented in [6]

Another one of our work [11] attempts to solve such disparity in hybrid FFT algorithm where we devise a higher-performing FFT algorithm compared to the vendor Cuda FFT library (Figure 1), and moreover, construct a performance model such that the combined heterogeneous outcome in performance can be accurately described according to how we divide the labor; here the performance model was shown to be accurate to be within 5%, and effective division could be derived with a simple search for the minimal point in the performance curve in the model.

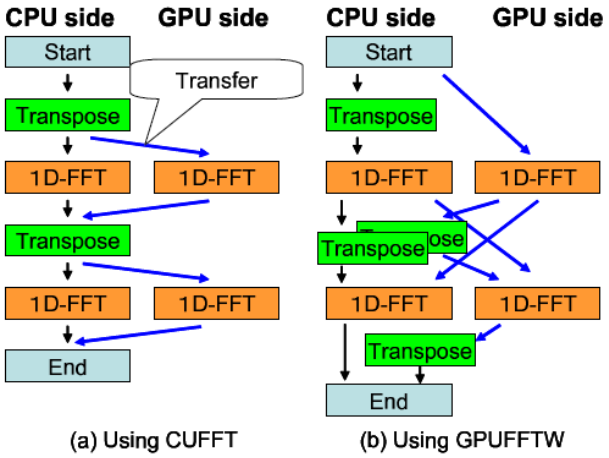

Figure 2: Overview of our hybrid GPUFFTW Algorithm

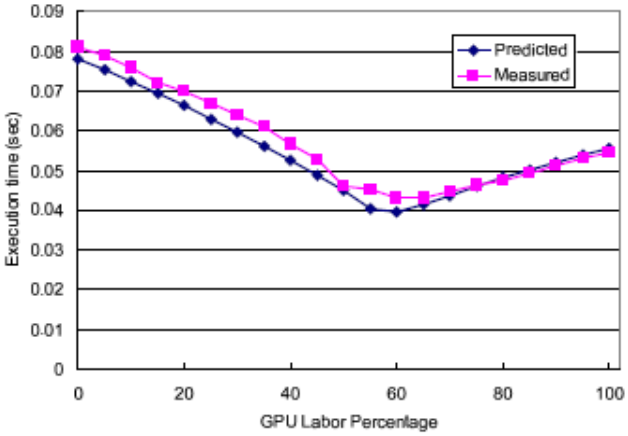

Figure 3: Performance comparison of our hybrid 1024-point 2-D FFT algorithm---predicated performance model instantiated by a 512-point sample run versus real measurement.

Still another work by our collaborator, Professor Takayuki Aoki's group approaches the problem in a completely different fashion---in solving the Poisson equation for a benchmark CFD problem (Himeno Benchmark), the inner loop of the algorithm principally conducts discretized stencil differentiation operations exploiting the computational power of multiple GPUs in a compute node, whereas the CPU "driver" routine is an outer-loop coordinator that effectively synchronizes the GPUs and conducts boundary region data exchange using OpenMP. As such, the CPU stays away from the core of the inner loop. The combined parallel performance using four Nvidia 8800 GTX Ultra cards in a node essentially "blew away" the previous known result for Himeno for a single

node by several factors, achieving nearly 100GigaFlops on a 3-U chassis, allowing them to win the Himeno Benchmark Contest[12] for 2008. Now Aoki's team is starting to apply the methodology to more realistic CFD and related problems.

2). *Effective usage of memory bandwidths*---Vector accelerators, especially GPUs are heavily geared towards stream processing, and thus sport substantial memory bandwidth. Although there is some commonality to traditional vector machines, there are vast differences in how the memory system is architected, affecting how memory latency is tolerated. Traditional vector machines involve bank-interleaved memory of substantial parallelism, up to 128 on the NEC SX series[8]. Long-latency memory operations such as stride access are tolerated by such parallelism in the memory system, which would be quite expensive to implement for various reasons. GPUs, on the other hand, cannot afford such an expensive memory system. Instead, it relies heavily on multithreading to tolerate overall memory latency---a modern day GPU can have hundreds to thousands of threads with outstanding memory operations in flight. However, maximum performance as afforded by the memory system can only be achieved when threads access the memory in an orderly fashion, enabling *memory coalescing*, which is fairly large granule (64 bytes in the case of Nvidia 8800 GTX/GTS) with strict access ordering constraints. Irregular memory accesses, such as stride access, are thus particularly poor-performing on GPUs, where latency can be hidden but bandwidth is still sacrificed quite considerably. (The Cell processor is somewhat intermediate in that, they offer lower bandwidth but our measurements have found the stride access capability to be more flexible compared to GPUs.)

The issue then, is how much the traditional vector-oriented algorithm developed over the years would be applicable (or not) to modern-day commodity vectors from a performance perspective. Although there are no general answers to the question yet, there are several works in porting representative numerical kernels effectively onto commodity vectors. BLAS kernel performance on Clearspeed is particularly both fast and efficient. [9] proposes mixed usage of GPUs and CPUSs in BLAS kernels, and [10] presents a work where Level-2 BLAS achieves considerable stability over varying problem sizes on CUDA GPUs. Our recent work lead by Akira Nukada et. al. studies efficient 3-D FFT algorithm on a Nvidia CUDA GPU, where we found that some of the classical vector-based algorithms are well-applicable, but in other cases various resource constraints, particularly in the registers, plus the stringent memory coalescing requirements call for substantive adaptation of such algorithms. The net results, however, are stellar: while a 16 CPU-core AMD Opteron (2.4 Ghz) node of TSUBAME achieves less than 20 Gigaflops in double precision for a 256^3 3-D FFT calculation, a single precision version on a Nvidia 8800 GTX achieves almost 80 Gigaflops (Figure. If this were achieved in double precision, it would almost be equivalent to a single node performance on a NEC SX-8 (16 vector CPUs). It is quite interesting that, with appropriate tuning, a single graphics card almost achieves the same performance as a purpose-built high-end vector supercomputer costing 1000 times as much, and at the same semiconductor process (90nm).

Figure 4: a 3U Compute Node with 4 Nvidia 8800GTS cards.

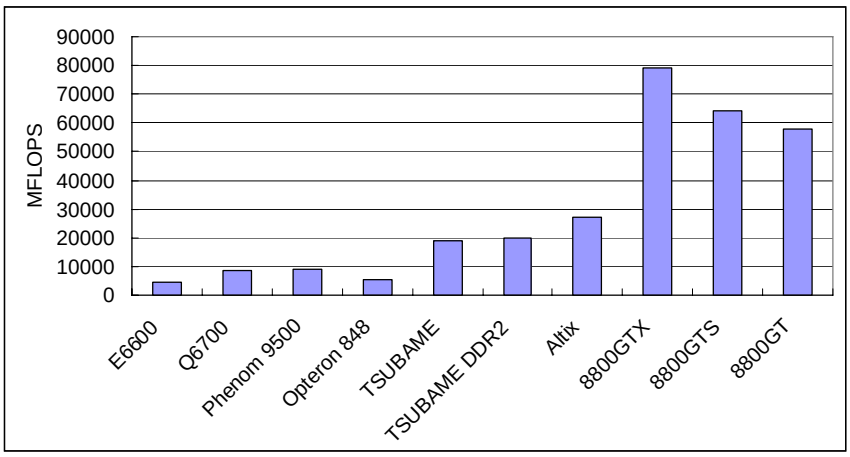

Figure 5: Performance of our 2563 3D-FFT Library (written by Akira Nukada) in comparison to standard CPUs. Here, TSUBAME is a 16-core 2.4 Ghz Opteron node (SunFire x4600), and TSUBAME DDR2 is the 2.6 Ghz node with PC2700 DDR2 memory. For standard CPUs single and double precision performances have been found to be largely equivalent.

3) *Coping with poor vector accelerator to CPU (and network) bandwdith*---We have

already mentioned that there is a vast disparity between memory bandwidths esp. of GPUs versus the achievable transfer speed on a PCI-Express bus. Although this could be alleviated somewhat by stronger integration of CPUs and vector accelerators as is the case for gaming consoles such as Sony PlayStation 3 or Microsoft Xbox 360, such disparity will likely persist at some scale due to substantial increase in point-to-point data transfer rates of directly-attached memory versus those that are not.

Given such fundamental restrictions in bandwidth, the current major workarounds are as follows 1) offload only non-bandwidth intensive application kernels to a vector accelerator 2) confine (almost) the entire application kernel to reside on the vector accelerator, or 3) attempt to hide the transfer latency by overlapping computation with data transfer. Of these, 1) is the approach used in dense problems, in which data transfer relative to computation is minimal. 2) is often used in many applications to date, including applications we have written that exploits our 3-D FFT above. In particular, we were able to realize a 3-D protein docking application whose main driver loop resides entirely on the card, and whose computation is dominated by 3-D convolution / 3-D FFT, achieving almost 50 Gigaflops on a Nvidia 8800 GTS, avoiding the major PCI-Express transfer bottleneck (Figure 6). However, this is not always applicable, and in fact could be difficult when the scalar bottleneck within the kernel becomes more dominant. 3) would be effective albeit in a smaller way, as the ability to overlap would depend on a good balance between computation and communication in the first place. Instead, the core solution to the problem from the software perspective would be to devise new algorithms such that 1)-3) or other methods would be applicable. Again, these may call for substantive research and development.

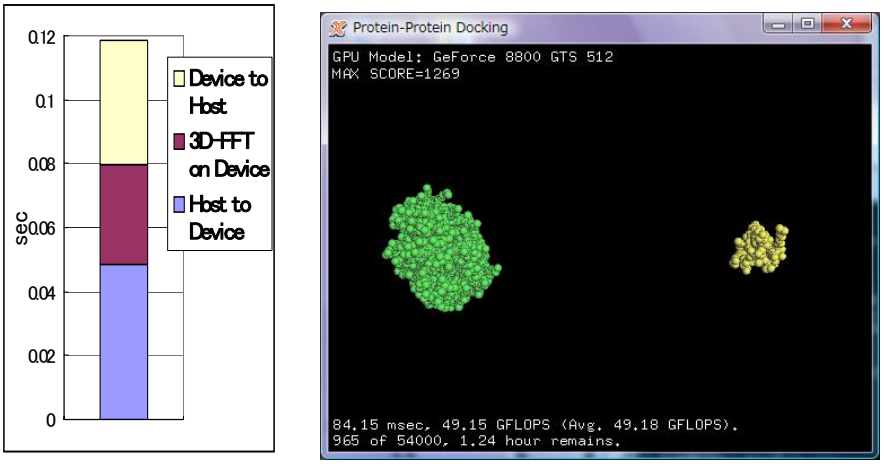

Figure 6: The graph on the left hand sides shows the overhead of PCI-E transfer of 256^3 3-D FFT (using PCI-E Gen 2 x16). The docking application on the right eliminates the transfer overhead by combining the entire application loop, including 3-D FFT, within the GPU card. The sample run using Nvidia 8800 GTS is exhibiting almost 50 Gigaflops per card, largely preserving the 3-D FFT performance.

Other difficulties include general programming issues, ranging from the higher-level language model that would exploit the SIMD/thread parallelism while realizing orderly access to memory, as well as the actual software infrastructural support in program development, such as compilers, debuggers, various libraries, etc. Another issue is general reliability including numerical accuracy and fault tolerance. Some of the problems resemble those for the early days of cluster computing, and still would require several years of research to have them 100% resolved, despite the accumulation of knowledge and experiences from the cluster days, due to a very different computational hardware underneath. Nevertheless, as was with clusters and classical vectors, such continued research and the resulting innovation, as well as user education, will likely allow commodity vectors to be used in an easy fashion

3. The Future of Commodity Vectors

As the commodity vectors progress, would there be convergence in the architectures, just was as the case for standard CPUs to x86, due to software platform issues, or instead, diverge and become more hybrid in order to exploit the various hardware advantages that are given rise due to the nature of the target application class? Would they still be subordinates to standard CPUs, where only specific workloads are offloaded, or would they become core compute engines for most HPC workloads, demoting the standard CPUs to subordinate status for essentially running errands? What would be the key technology advances that would sustain the tremendous performance growth, not only in hardware---3-D chip design, optical chip-chip interconnect, terabyte/s-bandwidth memory systems, etc.--- but also in software, to cope with the peaky and highly parallel nature of the architecture, where the approach could be leveraging much from classical vectors (i.e., rebirth), or will we be diverging considerably, due to the commodity nature of the hardware? These and other questions are still unanswered, but their exploitations would lead to promising rise of the performance and applicability of commodity vectors, allowing us to reach not only the petaflops, but even exaflops and beyond in the very near future.

Acknowledgements

This work is supported in part by Microsoft Technical Computing Initiative, and also in part by Japan Science and Technology Agency as a CREST research program entitled ``Ultra-Low-Power HPC''. The work represents numerous efforts by our commodity vector / low power computing team, including Toshio Endo, Akira Nukada, Naoya Maruyama, Yutoh Hosogaya, Yasuhiko Ogata, and other collaborators including Profs. Takayuki Aoki, Yutaka Akiyama, and Fumikazu Konishi at Tokyo Institute of Technology.

References

[1] HW Meuer, E Strohmaier, JJ Dongarra, HD Simon. "The Top500 Supercomputer Sites", http://www.top500.org, 1993.

[2] John Owens, Mike Houston, David Luebke, Simon Green, John Stone, and James Phillips. "GPU Computing", Proceesings of the IEEE, Vol. 96, No. 5, the IEEE Press, May, 2008, pp. 879-899.
[3] Michael Shirts and Vijay Pande. "Screen Savers of the World Unite!", *the Science Magazine*, the American Association for the Advancement of Science, Vol. 290, No. 2498, Dec. 2000. pp. 1903-1904. Also http://folding.stanford.edu/.
[4] T. Chen, R. Raghavan, J. N. Dale, and E. Iwata. "Cell Broadband Engine Architecture and its first implementation—A performance view", IBM Journal of Research and Development, Vol. 51, No. 5, Aug. 2007.
[5] Clearspeed Whitepaper: CSX Processor Architecture, http://www.clearspeed.com/ docs/resources/ClearSpeed_Architecture_Whitepaper_Feb07v2.pdf
[6] Toshio Endo and Satoshi Matsuoka. "Massive Supercomputing Coping with Heterogeneity of Modern Accelerators", in Proc. IEEE International Parallel and Distributed Processing Symposium, the IEEE CS Press (CD-ROM), April 2008.
[7] Ryutaro Susukita, Toshikazu Ebisuzaki et. al. " Hardware accelerator for molecular dynamics: MDGRAPE-2", *Computer Physics Communications*, Volume 155, Issue 2, October 2003, pp. 115-131.
[8] Yukio To, Toshiyuki Furui, Takshi Nishikawa, Masato Yamamoto, and Katsumi Inoue. YA " Development Concept of Supercomputer SX-8", NEC Technical Journal, Volume.58;No.4;2005, pp. 3-6 (In Japanese).
[9] S. Ohshima, K. Kise, T. Katagiri, and T. Yuba. "Parallel processing of matrix multiplication in a cpu and gpu heterogeneous environment!. In Proc. VECPAR'06 - Seventh International Meeting on High Performance Computing for Computational Science, 2006, pp. 305–318.
[10] N. Fujimoto. "Faster Matrix-Vector Multiplication on GeForce 8800GTX", In Proc. LSPP Workshop '08, IPDPS08 Proceedings (CD-ROM), the IEEE CS Press, April, 2008.
[11] Yasuhiko Ogata, Toshio Endo, Naoya Maruyama and Satoshi Matsuoka. "An Efficient, Model-Based CPU-GPU Heterogeneous FFT Library", In Proc. HCW '08: 17th International Heterogeneity in Computing Workshop, IPDPS08 Proceedings (CD-ROM), the IEEE CS Press, April, 2008.
[12] Ryutaro Himeno et. al. "The Himeno Benchmark Contest", Riken, http://accc.riken.jp/HPC/HimenoBMT/contest_e.html, 2002