e-Science Journal
Global Scientific Information and Computing Center
Vol. 10

TSUBAME ESJ.

The TSUBAME2.5 Evolution

Molecular Dynamics Simulation Accelerated by GPU for GPCR with a non-Ewald Algorithm

Large-scale Parallel Iterated Local Search Algorithm for Traveling Salesman Problem

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

The TSUBAME2.5 Evolution

Satoshi Matsuoka
Global Scientific Information and Computing Center (GSIC) Tokyo Institute of Technology

TSUBAME2.0, was designed, installed, and operated at the Global Scientific Information and Computing Center (GSIC), Tokyo Institute of Technology, in collaboration with our partner vendors, commissioned on Nov. 1st, 2010. In Sept. 2013, it was upgraded to TSUBAME2.5, by which the theoretical peak precision in double precision floating point (DFP) arithmetic was improved from 2.4 Petaflops to 5.7 Petaflops. More impressive is the improvement in single precision floating point (SFP), in which the boost was from 4.8 Petaflops to 17.1 Petaflops, becoming the fastest supercomputer in Japan under that metric. Despite the massive increase in performance, average power consumption was reduced by about 20%, while upwards software compatibility was fully preserved. TSUBAME2.5 will serve its role as a leading machine in the Japanese HPCI (High Performance Computing Infrastructure). This article will describe the upgrade of TSUBAME2.0 to 2.5, and technological directions towards 3.0 in late fiscal year 2015.

Introduction 1

The TSUBAME2.0 Supercomputer has been in active use since its deployment in 2010 for early 3 years as Japan's first petascale supercomputer at Tokyo Institute of Technology, and one of the leading machines in the Japanese HPCI (High Performance Computing Infrastructure). In September 2013, it was upgraded to TSUBAME2.5, by which the theoretical peak precision in double precision floating point (DFP) arithmetic was improved from 2.4 Petaflops to 5.7 Petaflops, single precision floating point (SFP), performance accordingly from 4.8 Petaflops to 17.1 Petaflops, becoming the fastest supercomputer in Japan under that metric. However, the upgrade was by no means effortless, and rather was met with various challenges. This article covers the upgrade from various angles, the rationale for its planning, the challenges met, how they were resolved, and the resulting performance gains.

Fig. 1 TSUBAME2.5 Commissioned on Sept. 10th, 2013

Fig. 2 Compute node of TSUBAME2.5 embodying the new NVIDIA Kepler K20X GPU

Overview of the Predecessor --- TSUBAME2.0 2

TSUBAME2.0 was developed by GSIC in collaboration with an industry consortium consisting of major HPC vendors such as NEC/HP/NVIDIA, and became operational on Nov. 1st, 2010. It was the first ever Japanese supercomputer to surpass a petaflop, and became No. 4 in the world on the Nov. 2010 edition of the Top500 List and No.2 on the Green 500 list, and moreover was designated the "Greenest Supercomputer in the World" award as it was the only large-scale production supercomputer high on the latter list. Also, in Nov. 2011, it won the ACM Gordon Bell Award, which is the most esteemed award in supercomputing regarding the execution of a real large-scale application, sharing the honor with the Riken AICS K Computer. By the end of July 2013, there are approximately 10,000 registered users, of which 2,000 were bona-fide supercomputer users, and at any time 50-100 users were logged onto the machine. Aside from periodic maintenance every several months, and emergency situation imposed by the Tohoku earthquake in March 2011, TSUBAME2.0

has been in operation 24/7 throughout the year, helping to produce various important scientific results.

TSUBAME2.0 is serving as one of the primary resources for the nationwide supercomputing infrastructure, along with other major university supercomputing centers and the K computer at Riken AICS (Advanced Institute for Computational Sciences) and the Earth Simulator 2 at JAMSTEC ES Center, altogether forming a consortium of centers called HPCI (High Performance Computing Infrastructure). A part of the resource of TSUBAME2.0 is allocated through a national HPCI allocation process. Moreover, there are many industrial users that utilize TSUBAME2.0 under various industrial usage programs, ranging to over 100 companies to date.

TSUBAME2.0 was developed as an advanced supercomputer leading the technology fronts in many aspects, in collaboration with the top HPC companies of the world, such as NEC HP, NVIDIA and DDN. Some of the salient technical elements are as follows:

High Performance, High Bandwidth Compute Nodes:

The primary compute nodes of TSUBAME2.0 are called "thin nodes". There are 1408 thin nodes, that have been productized as HP Proliant SL390s G7, each consisting of 2 Intel Xeon multi-core CPUs (6 cores 2.93Ghz Westmere-EP), and 3 NVIDIA M2050 GPUs (448 CUDA cores, 515GigaFlops), with 54 or 96 GigaBytes of CPU memory, and 3 GigaBytes of fast DDR5 memory for each GPU, each with 150 GigaByte/s of bandwidth. The aggregated compute and memory bandwidth capability of the "thin nodes" are 1.6 TeraFlops and over 500 GigaByte/s, respectively. There are additional 40 nodes of "medium" and "fat" nodes that utilized standard servers and host 128-512 GigaBytes of memory.

Hierarchical Large-Scale Storage:

In order to meet a variety of diverse and massive I/O demands, TSUBAME2.0 storage is composed of three hierarchical elements, namely (1) node-local SSDs, (2) shared parallel file systems using HDDs, and (3) archival tape library.

(1) Each compute node is equipped with 120-240 GigaBytes of SSDs (solid state drives) configured as Raid-0. By utilizing SSDs for temporary file I/O of each node such as scratch I/O and checkpoints, we greatly reduce the I/O demands of the parallel file system below in the hierarchy. The I/O bandwidth of the SSDs are over 300 MegaByte/s, or over a 1/2 TeraByte/s for the entire machine.

(2) The shared parallel file system has 7.2 Petabytes of raw capacity comprised of nearly 4000 HDDs controlled by the DDN SFA 10000 storage controller and a farm of storage servers. The storage is divided into 6 partitions, one as a home directory and 5 as parallel filesystems, 3 being Lustre and 2 as GPFS parallel filesystems. The bandwidth of each partition is approximately 10 GigaByte/s, for over 50 GigaByte/s aggregated I/O bandwidth.

(3) Finally, as a backup and archival storage, there are 8 PetaBytes (compressed) tapes managed by the SL8500 tape system. The GPFS filesystem region works automatically with SL8500 to implement a Hierarchical File System using IBM Tivoli software.

Full Bisection Infiniband Network:

Over 1400 compute nodes, the storage nodes and their servers are interconnected with QDR Infiniband network. Each node has two rails (links) of Infiniband, each with 40 Gigabit/s of bandwidth, totaling 80 Gigabit/s of injection bandwidth into the network. The actual measured bandwidth is approximately 7.5 GigaByte/s, and less than 2 microsecond latency. The entire fabric consists of 12 core switches with 324 ports each, and 179 edge switches with 36 ports each, comprising a full bisection fat-tree network, mutually connected with advanced silicon photonics optical network with 3500 optical fibers totaling 100 kilometers in length. The bisection bandwidth of the entire network is 220 Terabit/s, which is exceeds the combined average of all the global Internet traffic in 2012.

Upgrade Plans from TSUBAME2.0 to TSUBAME2.5 3

TSUBAME2.0 saw extremely high utilization rate thanks to its advanced features, but recently it was reaching capacity limit. In particular, in the busy seasons of the latter half of the fiscal year, the utilization rate easily sustained 90 % , and at extreme times it reached 100 % ; so despite being number two fastest supercomputer in Japan, the overall capacity was becoming obviously insufficient. Also, the overall international competitiveness of the TSUBAME2.0 was obviously degrading steadily. Since supercomputer performance progresses by a factor of 1,000 over every 10 years, such decline was imminent, but nonetheless TSUBAME2.0 global ranking on the Top 500 had degraded being 3rd in Japan and outside the top 20 in the world.

Since TSUBAME 2.0 was planned to have a four

year lifetime, such degradation would not have been a major problem if TSUBAME3.0 could be deployed in November 2014. This interval was planned assuming that processor vendors would make generational progress every two years; as such, TSUBAME3.0 was planned to employ processors two generations forward. However, due to the slowdown of advances in semiconductor processing, in discussion with the processor vendors it became apparent that such two-year interval would be somewhat extended with minor time-slips, and it was turning out that two-generation process advance in four years was becoming infeasible.

Moreover, the Tohoku earthquake that occurred on March 3, 2011 rejuvenated the general public awareness for disaster prevention; as such, there are now much stronger emphasis for supercomputers to be utilized towards matters of high social interest, such as disaster prevention, environment, medical applications, and advanced manufacturing. TSUBAME2.0 had already hosted numerous applications of such categories, e.g., various seismic applications on TSUBAME2.0 had contributed significantly to the creation of a national hazard map. However, as described earlier in busy times the utilization of TSUBAME2.0 approaching 100 % had prevented timely allocations of resources for such applications of urgent needs; thus, not only capacity increase was deemed important, but also, prioritized resource scheduling for such applications was also required.

Finally, TSUBAME2.0 as being one of the leading HPCI resources, such application of significant social demands need to be accommodated smoothly across the supercomputers in the overall HPCI. HPCI as an infrastructure already hosts a nationwide production infrastructure such as unified HPCI account and its associated authentication and authorization services, but the most important is common and unified nationwide archival storage. In fiscal year 2012 such storage system was deployed at two locations, the East one being at the supercomputer center of the University of Tokyo, and the West one being co-located with the K-Computer at Riken-AICS in Kobe, totaling to approximately 22 petabytes of nationwide shared storage. It then became clear that TSUBAME2 would also require near-line storage that would coordinate with those two centers in order to alleviate burst remote I/O traffic through our national academic network backbone SINET 4.

Given the above status quo, we as GISC planned to extend the operational lifetime of TSUBAME2.0 by at least one year, and to plan for a partial upgrade utilizing the excess funds due to contractual extensions, but at the same time proposed the full-system upgrade to the Informatics Division of MEXT (The Japanese Ministry of Education, Culture, Sports, Science and Technology), and commenced the official acquisition process for supercomputers as required by law.

During the acquisition process, a new regime came into office of the Japanese government in the latter half of 2012. The Prime Minister instituted a large-scale supplementary fiscal budget plan as economic stimulus and also to accelerate the recovery from the Tohoku Earthquake, and the HPCI infrastructure became one of the subjects of the budget allocation. We in turn re-submitted the earlier proposal to enhance the coverage of applications of high societal needs, and subsequently was accepted, and allowed us to commence with full upgrade from TSUBAME2.0 to 2.5. Although there were a few different proposals, in the end NEC along with NVIDIA and HPC won the upgrade bid on July 12, 2013, and immediately commenced the upgrade, for TSUBAME2.5 to become operational on September 10, 2013.

Technical Details of the TSUBAME2.5 Upgrade 4

By replacing the NVIDIA Fermi M2050 GPU entirely with the latest generation NVIDIA Kepler K20X GPU, TSUBAME2.5 peak performance was boosted to 5.76 PetaFlops DFP and 17.1 PetaFlops SFP respectively; also, the peak memory bandwidth became 1.16 PetaByte/s, and approximately 0.8 PetaByte/s measured. Figure 3 shows the overview of the node upgrade, and Table 1 is the comparison of the specification for TSUBAME2.0 and 2.5. However, such upgrade of GPUs was not automatic, and many possibilities had to be considered, as well as technical challenges be met and resolved.

Although thanks to the supplementary budget substantial upgrade was possible, adding new nodes was quickly ruled out, due to the limitations in power and space. Other upgrade paths were also considered, but the upgrade of the accelerators was judged to be the most technically viable. However, since TSUBAME2.0 was an production machine with 24/7 operational responsibilities with thousands of users, various technical problems arose:

TSUBAME 2.0 ⟶ 2.5 Thin Node Upgrade

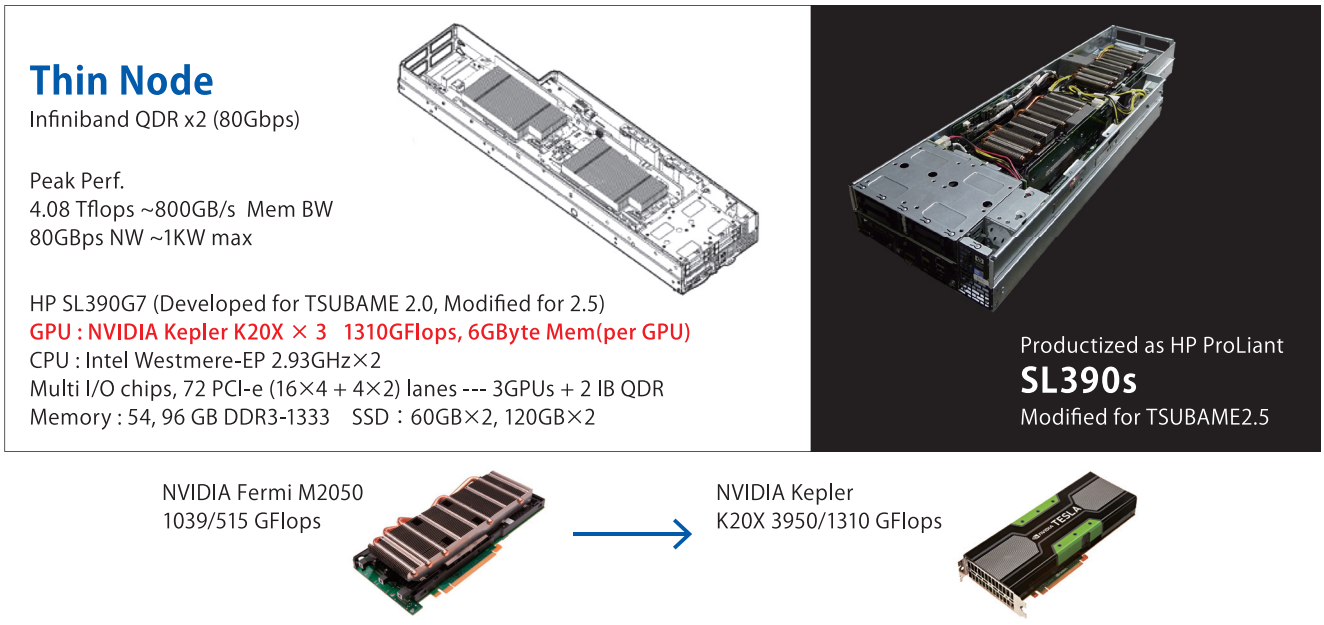

Fig. 3 TSUBAME Compute "Thin" Node Upgrade

(1) Which many-core accelerator to upgrade to? Not only performance increase important, can we maintain upward software compatibility with the existing software stack and the applications? Since upgrade was initially not publically planned or budgeted for TSUBAME2.0, users might not accept drastic changes to the system in its mid-life.

(2) Will the new accelerator work on the TSUBAME2, especially on its thin nodes? Will it have hardware compatibility? Will there be power increase, and if so, will the power and cooling system for the node, rack, as well as the entire machine be able to tolerate the increase? Upgrading a large supercomputer is very different from upgrading your personal PC, where the change might be an instant parts-swap; rather, for large machines long-duration operational capabilities and reliability over many years with 24/7 high load are essential, and as such the machine must be shown to operate properly at that level even with the parts upgrade.

(3) Will it be efficient, in particular, will new bottlenecks manifest themselves due to performance increase? The candidate accelerators all exhibited 2 to 3 times speedup, and thus PCI-e, Infiniband network, as well as storage I/O could potentially become bottlenecks, negating the effect of the upgrade.

(4) Related to above, some of the candidate accelerators such as the K20X, offered higher boost in SFP (Single-Precision Floating Point) compared to M2050, as the ratio of SFP:DFP would increase from 2:1 on M2050 and Westmere CPUs, to 3:1 or 4:1. The question is, however will such boost performance further, or is there no return on the increase? In particular, Intel Xeon CPU also sports the same 2:1 factor, while CPUs dedicated to supercomputers often have 1:1 ratio. We have shown on TSUBAME2.0 that multiple applications benefit significantly from this boost, by computing in single only or mixed single/double precision, however it was not obvious whether higher ratio would benefit the real applications.

(5) Would upgrade be possible minimally affecting the operations, especially for the users? There are 4224 M2050 GPUs in TSUBAME2.0 in its 1408 thin nodes, and upgrading each one involves stopping of the node, exchanging the GPU, and stress testing the nodes for some duration, such that it would take weeks to conduct the replacement. Moreover, if we maintain TSUBAME2 in operation during the upgrade, there will be rather random mixture of old and new nodes. As such it was an operational challenge to conduct the upgrade essentially in the background without compromising the user perception of the system, e.g., avoiding extended and/or unscheduled down time, given these constraints.

	TSUBAME2.0	TSUBAME2.5
Thin Node x 1408 Units		
Node Machine	HP Proliant SL390s	← No Change
CPU	Intel Xeon X5670 (6core 2.93GHz, Westmere) x 2	← No Change
GPU	NVIDIA Tesla M2050 x 3 ● 448 CUDA cores (Fermi) ➢ SFP 1.03TFlops ➢ DFP 0.515TFlops ● 3GiB GDDR5 memory ● 150GB Peak, ~90GB/s STREAM Memory BW	NVIDIA Tesla K20X x 3 ● 2688 CUDA cores (Kepler) ➢ SFP 3.95TFlops ➢ DFP 1.31TFlops ● 6GiB GDDR5 memory ● 250GB Peak, ~180GB/s STREAM Memory BW
Node Performance (incl. CPU Turbo boost)	● SFP 3.40TFlops ● DFP 1.70TFlops ● ~500GB Peak, ~300GB/s STREAM Memory BW	● SFP 12.2TFlops ● DFP 4.08TFlops ● ~800GB Peak, ~570GB/s STREAM Memory BW
TOTAL System		
Total System Performance	● SFP 4.80PFlops ● DFP 2.40PFlops ● Peak ~0.70PB/s, STREAM ~0.440PB/s Memory BW	● SFP 17.1PFlops (x3.6) ● DFP 5.76PFlops (x2.4) ● Peak ~1.16PB/s, STREAM ~0.804PB/s Memory BW (x1.8)

Table 1 TSUBAME2.0 and TSUBAME2.5 Thin Node Specifications

Such technical problems were resolved with careful planning and engineering, then reflected onto the TSUBAME2.5 acquisition specifications. By all means the partner vendors that won the bid contributed significantly in the resolution. Although we cannot go into every detail due to the lack of space, we outline the specifics of the problems, and how they were resolved.

(1) **Which Many-Core Accelerator to Upgrade to:**

The only practical many-core processor that could be usable for large-scale, general purpose supercomputers in the TSUBAME2.0's development and acquisition timeframe in 2008-2010 were NVIDIA (GP)GPUs, and both TSUBAME1.2 and 2.0 were equipped with the Tesla variant which was dedicated to HPC. In practice, during the 3 years of operations, the 4224 GPUs performed reliability as many-core processors sans some minor glitches, and recent utilization rate has been 30~50% and improving steadily, as well as allowed for whole-system stable execution of GPU-based petascale grand-challenge problems. Due to the successes of NVIDIA, other processor companies followed suite, such as AMD FireStream, and in particular Intel Xeon Phi which facilitated x86 instruction set compatibility and could execute most program with simple re-compile. However, upon various tests we conducted, with the first generation Xeon Phi, it took considerable tuning effort to match the performance of applications running on the Fermi GPUs, and difficult to match those of Kepler. As a result, we partitioned the upgrade into two systems, one requiring direct compatibility and immediate higher performance of existing applications on Fermi GPU, and the other requiring only software compatibility with either GPU or CPU multithreaded code, possibly accommodating Xeon Phi as well as other many-core processors. In the end, both system became K20x upgrades as already noted as a result of the public bidding process.

(2) **Will the new accelerator work on the TSUBAME2.5 thin nodes?**

According to the specification supplied by the vendors, power consumption figures for the candidate accelerators such as K20x and Xeon Phi were 235-300 Watts TDP, significantly exceeding that of M2050 which had 225W TDP; as a result, the thin nodes had to be modified to strengthen its power and cooling of the accelerator bay. Moreover, the thin node that was developed for TSUBAME2.0 by HP, namely SL390s G7, did not support the new power and cooling interface protocol that was revised for Kepler GPUs, and as a result, mere plugging in of the new GPUs would be totally inoperative. By all means this could be resolved by engineering the server control plane software to recognize the new protocols, but would require significant collaborations between NVIDIA and HP-- a non-trivial effort. Fortunately, after long discussions and negotiations on all sides, custom modifications were engineered for SL390 to fully accommodate the Kepler GPUs, with appropriate vendor certification for long-duration operations.

(3) **Will it be efficient, in particular, will other parts of the system such as PCI-e and Infiniband network become new bottlenecks?**

The key to the design of a supercomputer architecture utilizing many-core processors is that, compared to using standard multi-core CPUs, both compute and memory bandwidth are several-fold greater, and thus would require significant improvement in the underlying intra-node I/O switches, in the node-to-node interconnect, and likewise in the storage I/O. TSUBAME2.0 was specifically designed to accommodate the massive bandwidth of 3 GPUs hosted by the SL390 nodes, with multi I/O hubs and multi-rail QDR Infiniband network, as well as intra-node SSDs. However, with 2-3 times boost in both compute and memory bandwidth capabilities, it quickly became an issue whether the node I/O capabilities were sufficient. For example, in the initial node-wise Linpack measurements, the SL390 generation Intel Westmere Xeon + Tylersberg IOH combination exhibited sufficient bandwidth for M2050; however, for Kepler K20x, in comparison to the newer

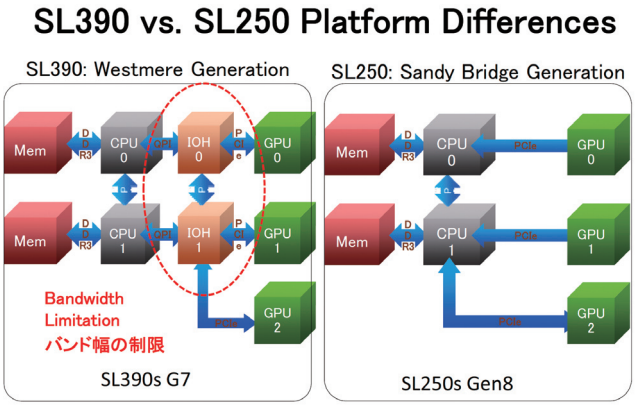

Fig. 4 Comparing HP SL390 and SL250 I/O Architecture

generation Intel Sandy Bridge Xeon with CPU-integrated and improved IOH as facilitated in the HP SL250 (successor to SL390), the PCI-e bandwidth proved to be insufficient, and as a result, we could only attain half the performance in Linpack by comparison (Figure 4).

Similar problem also manifested itself in the network; for TSUBAME2.0 the dual-rail QDR Infiniband exhibited approximately 7 Gigabyte/s of node-to-node bandwidth, sufficient for TSUBAME2.0, but is significantly inferior to the new dual rail FDR Infiniband which would have achieved 12~13 Gigabyte/s of bandwidth to have matched the speed of Kepler.

In order to resolve the problem, we conducted R&D on various fronts. For Linpack, we developed as well as tested and employed alternative algorithms that were more oblivious to relative lack of node I/O bandwidth. For the network itself, we conducted and continued our research effort on alternative and more efficient routing algorithms that would achieve higher network utilizations, as well as bandwidth-reducing algorithms on the compute side to cope with the reduced bandwidth. Altogether, we were largely able to recover the performance lost due to I/O bandwidth degradation.

Such efforts are not only specific to TSUBAME2.5, but we also believe that for future systems the bandwidth balance may become even worsen; as such our research efforts will continue to pay-off for future systems leading towards TSUBAME3.0 and exascale.

(4) **Is Single Precision FP Enhanced Acceleration Useful ?**

As mentioned many HPC applications utilize DFP arithmetic and not SFP, one of the primary reasons being that, in classic supercomputers DFP and SFP compute performances were the same, and with sufficient memory bandwidth for both in classical vector supercomputers, SFP provided very little performance advantage except for storage space, and application programmers used DFP "by default" to "play it safe" in numerical precision. However, due to the decreasing memory bandwidth relative to compute, and "boosting" of SFP in modern processors in order to accommodate multimedia applications that exhibit high data locality, the use of SFP arithmetic now provides significant potential performance advantages for both compute bound and memory bound codes, and various research are ongoing to utilize SFP, either alone or as "mixed precision" where SFP will dominate but DFP will be occasionally used to enhance precision. Many applications today can make good use of SFP, such as seismic wave propagation, tsunami simulation, as well as climate/weather prediction. Most such applications with explicit PDE solvers could directly utilize SFP. Also, various manufacturing applications such as electro-magnetic analysis, computational fluid dynamics for automotive and aerospace, as well as molecular dynamics for pharmaceutical drug design, are all subject to SFP or SFP-dominated mixed-precision. Such applications have been demonstrated to obtain significant speedup on TSUBAME2.0. The question is, what if we had higher peak beyond the 2:1 ratio. We know that in general there are several classes of algorithms and applications that can benefit, such as the (gravitational) N-Body problems, or Fast Multipole Methods (FMM), and various apps based on dense linear algebra; we will continue our efforts to widen the applicability of such boost.

(5) **Can the upgrade be performed without affecting the users ?**

Since the Tohoku Earthquake, TSUBAME2.0 has been operating in "peak-shift" mode during the hottest summer months from July until September, in which we turn off some of the nodes automatically to reduce the daytime power consumption as requested by the government. Such a scheduling mechanism had been developed as part of the "Green Supercomputer" project sponsored by MEXT. By coinciding our 2.5 upgrade with this mechanism, we were able to round-robin through the nodes by capturing the ones that are being turned off, and upgrading them during the day and re-commissioning them after the upgrade. This plus utilizing the mandatory power down of the machine due to campus-wide electrical maintenance,

we were able to prceed with the upgrade for almost two months without being "noticed" by the users. Since the old and new accelerators co-existed randomly, we devised an operational procedure by which the users could ask purely for old- or new GPUs if such purity would be required, and reflect them in batch scheduling. Altogether, the entire upgrade was largely complete by the end of August, two weeks ahead of schedule.

Thus, the upgrade from TSUBAME2.0 to TSUBAME2.5 was achieved, with significant boost in performance, compensating for the lack of capacity and allowing the extension of TSUBAME2 operation by at least one year, to adjust the timing of TSUBAME3.0 deployment. Current GPU-based benchmarks are demonstrating that the speedup is largely as expected despite the technical concerns described above, with 2~3 times improvements in performance metric. More concretely, Table 2 are some the major full-system metrics that have been observed:

Application	TSUBAME2.0 Performance	TSUBAME2.5 Performance	Boost Ratio
Top500/Linpack 4131 GPUs (PFlops)	1.192	2.843	2.39
Green500/Linpack 4131 GPUs (GFlops/W)	0.958	3.069	3.20
Semi-Definite Programming Nonlinear Optimization 4080 GPUs (PFlops)	1.019	1.713	1.68
Gordon Bell Dendrite Stencil 3968 GPUs (PFlops)	2.000	3.444	1.72
LBM LES Whole City Airflow 3968 GPUs (PFlops)	0.592	1.142	1.93
Amber 12 pmemd 4 nodes 8 GPUs (nsec/day)	3.44	11.39	3.31
GHOSTM Genome Homology Search 1 GPU (Sec)	19361	10785	1.80
MEGADOC Protein Docking 1 node 3GPUs (vs. 1CPU core)	37.11	83.49	2.25

Table2 TSUBAME2.0 to TSUBAME2.5 Performance Boost

We observe that Green500 metric in particular has seen considerable improvement. This is largely due to the improved power efficiency of the Kepler GPU despite the significant performance boost, plus that other parts of TSUBAME2.0 is more power efficient under lower thermal load. On the other hand, we are observing the network becoming a bottleneck despite our efforts as described above, and we hope that our ongoing research will alleviate some of the limits to further improve performance. By all means we will conduct further comparative benchmarks, and will publish the results publically.

Final Words --- Towards TSUBAME3.0 5

As mentioned earlier, upgrade to TSUBAME2.5 resulted in factor 2~3 boost in performance, which allows the extension of TSUBAME2's operational life by 1~1.5 years by supercomputer yearly performance improvement standards. Currently, we are actively researching and developing various technologies for TSUBAME3.0 as well as its doing its overall design. However, the detailed schedule does get affected by changes in the plans of the processor vendors. Currently, TSUBAME3.0 is slated to be coming into production by the end of the Japanese fiscal year 2015 (end of March 2016 calendar year), which will mean that TSUBAME2.5 lifetime will be more than 2.5 years, and TSUBAME2 overall would be nearly 5.5 years, despite initially planned with a 4 year lifespan. The upgrade performance boost and the lifetime extension strongly correlate, but such a long operational lifetime is largely due to the solid design and the flexibility subject to upgrades, a property not actually seen often even for cluster-based supercomputers.

The peak performance of TSUBAME3.0 is projected to be approximately 25-30 Petaflops, while being about the same size and power consumption as TSUBAME2.0. Such tremendous boost in power performance ratio will only be possible with various advances including more advanced and efficient cooling than TSUABME2.0/2.5. Moreover, in order to cope with massive data in the big data era, TSUBAME3.0's I/O as well as resiliency will be greatly enhanced. The details of various technologies as well as the TSUBAME3.0 design will be published in the forthcoming issues of this journal and elsewhere.

Molecular Dynamics Simulation Accelerated by GPU for GPCR with a non-Ewald Algorithm

Tadaaki Mashimo*† Takayuki Kochi* Yoshifumi Fukunishi** Narutoshi Kamiya***
Yu Takano*** Ikuo Fukuda*** Haruki Nakamura***

*The Japan Biological Informatics Consortium (JBIC)
** Molecular Profiling Research Center for Drug Discovery (molprof), National Institute of Advanced Industrial Science and Technology (AIST),
***Institute for Protein Research, Osaka University

Structural analysis of a protein as a drug target is now essential in pharmaceutical science, where molecular dynamics (MD) simulation of a protein is still time consuming. A major bottleneck in MD simulation is calculation of electrostatic interactions between atom pairs. We recently developed the zero-dipole summation (ZD) algorithm that can make MD simulation rapidly and precisely. We implemented the ZD algorithm into our MD software code, myPresto/Psygene, with the space decomposition algorithm, and we also developed its GPU version, myPresto/Psygene-G, for the TSUBAME supercomputer system. We examined the simulation quality and performance of myPresto/Psygene-G with the MD simulations for three protein systems. The GPU version was 30 times faster than the original CPU version, and it was applied to a G-protein coupled receptor (GPCR), indicating that the simulations are useful to understand the dynamics of GPCR.

Introduction 1

Since drug development is highly risky and expensive, computer-aided drug discovery is now essential to reduce the cost and time. Nearly 50 % of drug molecules bind G-protein coupled receptors (GPCRs) as their target proteins, which are known to be very flexible. Thus, computer simulation to reveal the dynamic features of the complex of a drug and its target GPCR is one of the major issues in pharmaceutical science. The GPCR protein family consists of several hundreds GPCRs, and the three-dimensional structures of several GPCRs have been revealed recently. It is now known that GPCRs show specific structural changes, called as the induce fit, when they bind their ligands and drugs, and that such structural changes depend on the ligand functions. To observe the structural changes like the induced fit, molecular dynamics (MD) simulation is a useful approach. In general, the all-atom MD simulations of GPCRs are time consuming, when lots of membrane and solvent molecules are included in a realistic manner. Thus, acceleration of MD simulation is a key technology for drug discovery.

One of the most time-consuming processes in MD simulation is calculation of electrostatic interactions between atom pairs, because of its long-ranged nature. We have so far developed the zero-dipole summation (ZD) method, which drastically reduces the amount of computations for calculations of the Coulombic electrostatic interactions keeping the accuracy in several different kinds of MD simulations [1-5]. Here, we developed a program for GPU-accelerated MD simulation, myPresto/Psygene-G, using the ZD method for TSUBAME supercomputer system. In the ZD method, the long-range Coulombic interaction is cut off within a short range as long as 12 Å, and the effect of the long-range electrostatic interactions are compensated by the image charges that are located on the cut-off sphere, so as to neutralize the monopole (charge) and dipole moment in the cut-off sphere. The ZD method is useful for parallel computation by reducing the numbers of long-range interactions, which would increase communications between the nodes.

In the current study, we applied the myPresto/Psygene-G to the complex structure of GPCR with its ligands, and we analyzed the dynamic features of the complexes.

Zero-dipole summation method 2

The evaluation of the Coulombic interactions does not allow the simple cutoff truncation, from the viewpoint of the accuracy and the stability of the MD simulation. For their suitable evaluation, we aimed at constructing a method to fulfill the following requirements: (i) high accuracy and low computational cost; (ii) freedom from artifacts; and (iii) ease of the implementation, which enhances the availability for use in high-performance parallel computational architectures.

For this purpose, we switch from the conventional view such that the value of the pair potential function is decreasing with increasing the distance r_{ij} between two particles, i and j. Instead, we take into account the electrostatic feature, i.e., the individual charge q_i and a certain structure. In the vivo environment, many molecules and ions crowd over individual particles. Each positively or negatively

*† Current position: IMSBIO (Information and Mathematical Science and Bioinformatics) Co. Ltd.

charged particle assembles in such a way that the electrostatic interactions cancel each other well, unless very high energy phenomena occur. Thus, actual interactions in biological systems are essentially screened, as compared with the bare Coulombic form $1/r$. These considerations provide a positive motivation for employing the cut-off based methods.

Based on these considerations, we have developed a novel idea, the ZD method. This method prevents the nonzero-charge and nonzero-dipole states artificially generated by a simple cutoff truncation, but the resulting energy formula is nevertheless represented by a pair-wise summation form. The method is due to the following two strategies: (A: *ideal*) For each particle i, the summation with respect to the all particles is replaced by the one with respect to the neutralized subset M_i, whose existence is assumed; (B: *specific*) Pair potential function is redefined in order that we can handle the summation defined in (A) using a simple pair-wise-sum form.

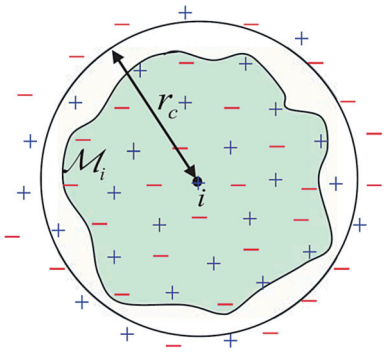

Fig. 1 The current ZD method conceptually deals with the particles only in the shaded region, which schematically represent the zero-dipole subset M_i.

We then have the total electrostatic energy in the following form [1,2]

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j(\neq i)\\ r_{ij}<r_c}} q_i q_j \left[u\left(r_{ij}\right)-u\left(r_c\right)\right]-\left[\frac{u\left(r_c\right)}{2}+\frac{\alpha}{\sqrt{\pi}}\right]\sum_i q_i^2,$$

$$u(r)=\frac{\operatorname{erfc}(\alpha r)}{r}+\left[\frac{\operatorname{erfc}(\alpha r_c)}{2r_c}+\frac{\alpha}{\sqrt{\pi}}\exp\left(-\alpha^2 r_c^2\right)\right]\frac{r^2}{r_c^2}$$

Here, $\operatorname{erfc}(\alpha)$ is the complementary error function of αr, with α being a damping parameter. For general molecular system, some modifications are required [2]. The resulting energy formula is represented by a simple pair-wise function sum along with a constant term, enabling the simple implementation and facile applications to high-performance computation. The ZD method does not assume the exact periodic boundary condition as used in the lattice sum method, which often causes artifacts in an application to an inherently non-periodic system [3]. In addition, the ZD method conserves the total energy and the center of mass of the physical system in the MD simulation, for which these conservations are not trivial in the particle mesh Ewald method and the fast multipole method even if they are good at the energy accuracy. The accuracy of the ZD method has been examined in several systems [1,2,4,5] and high efficiencies were confirmed. For example, in the GPCR system [4], which will be also discussed later, the energy accuracy of the ZD method was about 0.04 % at 12Å cutoff length. Thus, the reaming task is to attain its efficient parallelization, which is the main theme of this article.

MD simulation with the space decomposition algorithm 3

TSUBAME supercomputer is composed of a distributed memory system of a few hundred of GPU accelerators. Thus, the MD program for TSUBAME supercomputer should combine the implementation of CPUs and accelerators, which execute processes in parallel while being accelerated by an accelerator and exchanging the data through high-speed network communication.

myPresto/Psygene-G is an MPI/GPU-combined parallel program with an NVIDIA GPU as its accelerator, which was developed for massively parallel computers. This program divides many atoms that construct a system in a coordinate space. The MD simulation of the atoms belonging to the subspaces is assigned to the components (MPI processes) of the parallel computer. Among the assigned MD calculations, the pair-wise interactions of the non-bonded terms are computed on GPUs. Along with the data transfer required for the migration of atoms that extend across subspaces and the control of temperature and pressure, the mutual transfer of the atomic data between neighboring subspaces, which is necessary for calculations of the pair-wise interaction, uses the MPI communication. The parallel execution of myPresto/Psygene-G with more than a hundred CPUs accelerated by GPU nodes could allow it to handle a system consisting of over a few million atoms (Fig 2).

In the ZD method, the long-range electrostatic interaction is cut off within a short range, and so the most of the

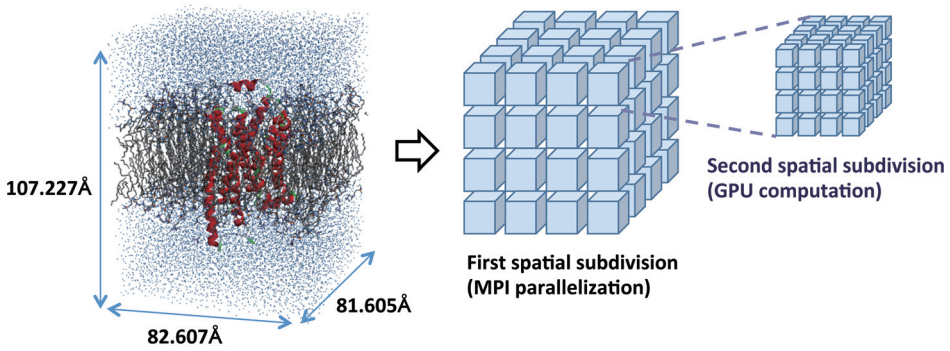

Fig.2 Schematic representation of the space decomposition in myPresto/Psygene-G

computations are performed within each cell. The ZD method is useful for parallel computation by reducing the communication among the cells, thus, the ZD method was applied in myPresto/Psygene-G.

Computation in each cell 4

In the MD simulation, most of the inter-atomic energy potentials are pair-wise interactions. For computing those interactions, which consist of the electrostatic and the van der Waals interactions, the forces between the atom pairs are individually calculated. Thus, the calculation time is almost proportional to the square of the number of atoms. The space decomposition method reduces the number of calculations by performing pair-wise interaction calculations for only the atom pairs included in the relevant and neighboring decomposed regions.

In myPresto/Psygene-G, the components of the decomposed system are called cells. The cells are obtained by decomposition of the cuboid system along three axes by an arbitrary number. Each cell manages its respective region boundaries, constituent atom quantity, and data of each constituent atom. Then, the entire system is formed by integrating all of the cells. Each cell is the smallest unit of the calculated regions, handled by a CPU core of the parallel computer. One CPU core as an MPI process is assigned to one cell.

In General, one step in the MD calculation is divided into calculations of force and integral of the equations of motion, where the force calculations are further divided into calculations of bonded terms and non-bonded terms. The calculation of non-bonded terms is further partitioned in three ways: calculations of the forces due to the van der Waals interactions, the short range Coulombic electrostatic interactions, and the long range one. Whereas the former two force terms are directly computed by pair-wise interaction calculations, and the last long range electrostatic forces are calculated by the ZD method.

Communication between processes 5

For calculations of the pair-wise interactions performed in each cell, it is necessary to calculate the interactions between the atoms in the relevant cell and the atoms in its neighboring cells. Thus, in the three-dimensional space decomposition, we need to send and receive the atom information in a cell to its surrounding 26 cells.

In myPresto/Psygene-G, the communication between the processors is conducted for exchanging atom information between the neighboring cells, as well as for actualizing the migration of atoms that straddle the cells. This communication is performed using the Message Passing Interface (MPI) communication. Here, we applied the asynchronous one-to-one communication, in order to reduce the communication overhead.

Protein system (No of atoms including solvent and membrane)	No of cells	1 step execution time (ms)	
		GPU	CPU
EGFR kinase (38,452)	1	73.28	3631.44
	8	10.57	456.24
β_2-adrenergic receptor (56,120)	1	128.05	6151.17
	8	17.21	773.60
Aquaporin-4 (104,414)	1	229.57	14708.82
	8	26.94	1827.38
	27	17.13	368.77
Dynein (1,004,846)	1	2352.85	248148.26
	27	96.89	9581.55
	64	51.27	2664.80

Table1 myPresto/Psygene-G benchmark results

Application to GPCR 6

There are mainly two types of GPCR-targeting drugs: agonists and inverse agonists. Partial agonist has the aspects of both agonists and inverse agonists. In the current study, we studied the structures of the agonist bound and the inverse agonist bound β_2 adrenergic receptors.

In biological system, GPCR is located in the membrane and the system consists of approximately 56,000 atoms. The dynamics of GPCR is slow process and the MD simulation should be performed at least several-tens nsec to observe the structural change of GPCR.

GPCRs are mainly composed of seven trans-membrane helices embedded in the membrane (Fig. 3). In Fig 3, the up-side and down-side are the extra-cellar and intra-cellar regions. Ligands (agonist, inverse agonist and partial agonist) bound to the almost middle of the trans-membrane helices. The structural change due to the induce fit is small around the bound ligand, but the structural change in the intra-cellar part is large enough for signal transduction.

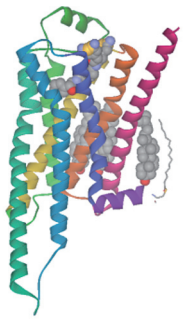

Fig.3
Cartoon graphics of GPCR (β_2-adrenargic receptor). Each helix is shown in different color.
The membrane and solvent molecules are not shown.

We applied myPresto/Psygene-G to the GPCR system, in which one GPCR, its bound ligand, membrane, and solvent atoms were included (total about 56,000 atoms), and we performed 20-50 nsec simulations to analyze the structural change of the GPCR induced by the agonists and the inverse agonists. MD simulation by myPresto/Psygene-G on TSUBAME supercomputer system was 30 times faster than that by the usual CPU version.

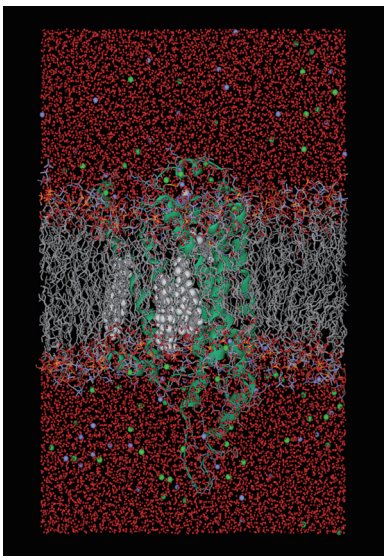

Fig.4 Molecular dynamics simulation of β_2-adrenargic receptor system. Final coordinate (after 20 nsec) at room temperature.

The ligand (agonist or inverse agonist) was placed into the ligand-binding site of the GPCR by docking calculation, and then the protein-ligand complex structure was embedded into the membrane. Finally, solvent water and counter ions (Na^+/Cl^-) were added to complete the system. We performed the MD simulation at room temperature with 1 atm pressure. After relaxing the system, we started observation of the system (Fig 4).

While we performed many MD simulations with various drugs, we showed the MD simulation results for carazolol (an inverse agonist) and formoterol (an agonist) with β_2-adrenargic receptor.

Figs 5 and 6 show the root-mean square deviations (RMSD) of the seven trans-membrane helix structures along the simulation time. The structural change induced by the inverse agonist (Fig 5) was larger than that by the agonist (Fig 6). These results show that the binding of small ligands could induce the larger structure change of the whole GPCR.

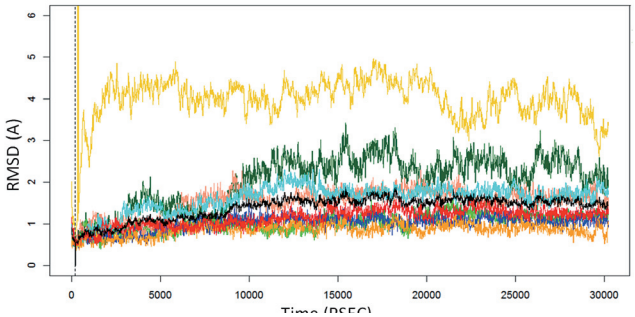

Fig.5 RMSD of seven trans-membrane helix (TMH1-TMH7) structures obtained by the MD simulation of β_2-adrenargic receptor with an inverse-agonist Carazolol. Light yellow: RMSD of all atoms of GPCR including loops. Dark green: TMH1, Green: TMH2, Blue: TMH3, Dark orange: THM4, Cyan: TMH5, Yellow: TMH6, Red: TMH7, Black: all atoms of TMH1-TMH7.

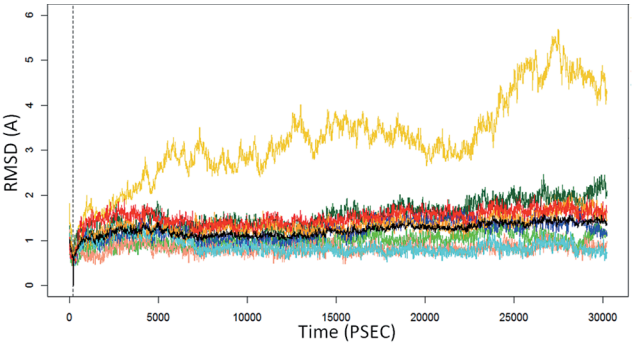

Fig.6 RMSD of seven trans-membrane helix (TMH1-TMH7) structures obtained by the MD simulation of β_2-adrenargic receptor with an agonist Formoterol. Light yellow: RMSD of all atoms of GPCR including loops. Dark green: TMH1, Green: TMH2, Blue: TMH3, Dark orange: THM4, Cyan: TMH5, Yellow: TMH6, Red: TMH7, Black: all atoms of TMH1-TMH7.

Conclusion 7

We developed an MD simulation software myPresto/Psygene-G for TSUBAME supercomputer system and the ZD method realized the rapid and precise calculations of electrostatic interactions. In this software, non-bonded pair-wise interactions were calculated on multi GPUs and the software was 30 times faster than the CPU version for conventional PC clusters. The myPresto/Psygene-G showed a scalable acceleration by the space-decomposition method, according to the number of computer nodes.

myPresto/Psygene-G was applied to the simulations of GPCR systems, which includes the GPCR molecule, its bound ligand (agonist or inverse agonist), membrane, and solvent molecules. We observed that the agonist-bound GPCR showed the different dynamics from the inverse agonist bound GPCR. Our study showed that the long-time large-scale MD simulations of drug-target proteins should be important to understand the mechanism of the drug effect and drug design.

Acknowledgements

These computations were executed as the TSUBAME Grand Challenge Program in 2012 and we thank to have an opportunity to use the TSUBAME resources. This work was supported by grants from the New Energy and Industrial Technology Development Organization of Japan (NEDO).

References

[1] I. Fukuda, Y. Yonezawa, and H. Nakamura: Molecular Dynamics Scheme for Precise Estimation of Electrostatic Interaction via Zero-Dipole Summation Principle, J. Chem. Phys., Vol. 134, 164107 (2011)

[2] I. Fukuda, N. Kamiya, Y. Yonezawa, and H. Nakamura: Simple and Accurate Scheme to Compute Electrostatic Interaction: Zero-dipole Summation Technique for Molecular System and Application to Bulk Water, J. Chem. Phys., Vol. 137, 054314 (2012)

[3] I. Fukuda and H. Nakamura: Non-Ewald methods: Theory and Applications to Molecular Systems, Biophys. Rev., Vol. 4, pp.161-170 (2012)

[4] N. Kamiya, I. Fukuda, and H. Nakamura: Application of Zero-dipole summation method to molecular dynamics simulations of a membrane protein system, Chemical Physics Letters, Vols. 568–569, pp.26–32 (2013)

[5] T. Arakawa, N. Kamiya, H. Nakamura, and I. Fukuda: Molecular Dynamics Simulations of Double-Stranded DNA in an Explicit Solvent Model with the Zero-Dipole Summation Method, PLoS One, Vol. 8, e76606 (2013)

Large-scale Parallel Iterated Local Search Algorithm for Traveling Salesman Problem

Kamil Rocki Reiji Suda
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

The importance of high performance parallel algorithms for tackling difficult combinatorial optimization problems cannot be understated. With the advent of computer systems equipped with millions of heterogeneous compute cores, it is especially important to develop algorithms which can be well suited for future machines. Currently, the most successful methods of solving such problems are meta-heuristic algorithms providing an approximate solution. One of the most successful and general algorithms is called Iterated Local Search (ILS), where the solution is gradually refined and its quality depends on the time available. We demonstrate that this can also depend on available parallelism, i.e. number of cores without losing the generality and requiring problem-specific knowledge, thus keeping the same assumptions as the original ILS. In this article, we are showing the Parallel Iterated Local Search (Parallel ILS) algorithm, a very efficient method of performing distributed combinatorial optimization. Due to its simplicity and abstraction it can be applied to any problem that can be solved using traditional ILS method requiring only slight modification of the sequential code. Our experimental results based on Traveling Salesman Problem (TSP) solving indicate that this algorithm is also more efficient than careful and time-consuming local search parallelization. We achieve over 90x speedup compared to sequential algorithm using our Parallel ILS method with MPI inter-node communication scheme on TSUBAME 2.0 supercomputer using 256 nodes.

Introduction 1

Combinatorial problems are present in many areas of computer science and other fields in which computational methods are applied, such as artificial intelligence, operations research or bioinformatics. The best known examples of such problems include optimal scheduling, finding models of propositional formulae (SAT), graph traversal (Traveling Salesman Problem - TSP[4]) or Quadratic Assignment problem (QAP). These problems typically involve finding groupings, orderings, or assignments of a discrete set of objects which satisfy certain conditions or constraints. Those where solutions are encoded with discrete variables contain a class of problems called Combinatorial Optimization (CO) problems. Therefore a solution is an object composed of a finite, or possibly countably infinite, set of integer numbers, a subset, a permutation, or a graph structure. For most combinatorial optimization problems, the space of potential solutions for a given problem instance is exponential in the size of that instance[1]. As a result of the practical importance of CO problems, many algorithms to approach them have been developed. These algorithms can be categorized as either complete (exact) or heuristic (approximate) algorithms. Complete algorithms are guaranteed to find for every finite size instance of a CO problem an optimal solution in bounded time [1][2][3]. Within the approximate algorithms we can distinguish between constructive methods and local search methods. Constructive algorithms generate solutions from scratch by adding - to an initially empty partial solution - components, until a solution is complete. Local search algorithms start from some initial solution and iteratively try to refine the current solution by a better one in an appropriately defined neighborhood of the current solution[3][5]. However, Local search methods can get stuck in a local minimum, where the solution is unsatisfactory and improvement not possible. Therefore they are not suited for searching the whole state-space as the number of such local searches would be enormous. However, they can quickly converge to a local minimum, which means that a large number of local minima can be explored. In order to improve this process, a new type of algorithms called metaheuristics[7][8] has emerged. This class of methods tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. It includes algorithms such as Ant Colony Optimization[11] (ACO), Genetic Algorithms[10] (GA), Iterated Local Search[6] (ILS), Simulated Annealing[9] (SA), or Tabu Search[7] (TS). This paper focuses on the ILS algorithm (Figs. 1 and 2), it is a simple but powerful metaheuristic algorithm [12][13][16][17].

Iterated Local Search 2

Iterated Local Search (ILS) is a Stochastic Local Search (SLS) method that generates a sequence of solutions generated by an embedded heuristic, leading to far better results than if one were to use repeated random trials of that heuristic[5]. The implicit assumption is that of a clustered distribution of local minima. When minimizing a function, determining good local minima is easier when starting from a local minimum with a low value than when starting from a random point. The iterative algorithm is based on building a sequence of locally optimal solutions by: 1. perturbing the current local minimum; 2.

applying local search after starting from the modified solution. The perturbation strength has to be sufficient to lead the trajectory to a different attraction basin leading to a different local optimum. However it cannot be to strong as it would lead to a random restart strategy. There are many strategies of choosing the right perturbation technique as well as there are many local search algorithms for every problem. Our algorithm uses a local search method called 2-opt exchange.

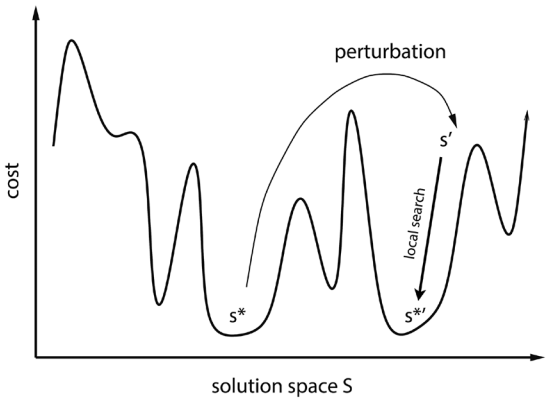

Fig.1 Iterated Local Search – solution space

```
1: procedure ITERATED LOCAL SEARCH
2:     s0 := GenerateInitialSolution()
3:     s* := 2optLocalSearch (s0)
4:     while (termination condition not met)
5:         s' := Perturbation (s*)
6:         s*' := 2optLocalSearch (s')
7:         s* := AcceptanceCriterion (s*, s*')
8:     end while
9: end procedure
```

Fig.2 Iterated Local Search – algorithm

2.1 2-opt Local Search

The 2-opt algorithm basically removes two edges from the tour, and reconnects the two paths created. This is often referred to as a 2-opt move. There is only one way to reconnect the two paths so that the tour remains valid (Fig. 3). It improves tour by reconnecting and reversing order of sub-tour. The procedure is repeated until no further improvement can be done. It is good for finding a local, but it is not guaranteed to find the best possible solution (the global optimum) out of all plausible solutions. A method allowing escaping from local optima has to be provided, which usually means that a local solution needs to be worsened is some way, keeping it within a certain search-space.

distance(B,F) + distance(G,D) > distance(B,D) + distance(G,F)

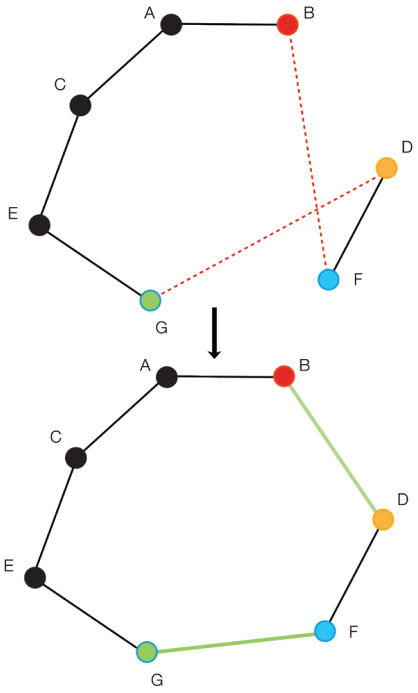

Fig.3 2-opt exchange

Parallel Iterated Local Search 3

It is important to mention, that in the previously presented Multi-start Local Search algorithm does not have strict synchronization point, therefore the threads or processes can run on different machines at different speeds or even using different algorithms. This is the key point of the modified algorithm that is using active communication: Parallel Iterated Local Search (or Multi-start Local Search with Communication). The sequential code remains basically unchanged except for the thread encapsulation, added memory synchronization (critical section) as well as storing and reading the best global solution. Due to the required communication between the threads - both the cost and the solution itself have to be shared - the implementation differs depending on the system. The whole algorithm is explained in Listing 1.

```
Algorithm :  Multi-start Local Search with Communication
 1: procedure  Parallel Iterated Local Search
 2:     S* < - ∞                          //Best Known Solution
 3:     s^n_0 <- GenerateInitialSolution ()  //Random or Heuristic
 4:     s^n* <- ParallelLocalSearch (s^n_0)
            //All processes
 5:     while termination condition not met do
 6:         if Cost(s^n*') > Cost (S*) then
 7:             s^n* <- S*
            //Read the Best Global Solution
 8:         end if
 9:         s^n* <- Perturbation (s^n*)
10:         s^n*' <- ParallelLocalSearch (s^n*)
    //Intra-thread parallelism (SIMD)
11:         if Cost(s^n*') < Cost (S*) then
12:             S* <- s^n*'      //Update the Best Global Solution - Critical Section
13:         end if
14:         if AcceptanceCriterion (Cost(s^n*')) == TRUE then
15:             break
16:         end if
17:     end while
18: end procedure
```

Listing 1 Multi-start Local Search with Comminication Pseudo-code

First, each process starts with a random solution which stands for a different point in the state-space. Further, the first, initial descend is being performed. Afterwards, each process executes the search-perturb ILS cycle until at least one of them has an acceptable solution. During this procedure, once a better solution is found by one of the processes, it is propaged to other ones. Lines 7 and 12 show the steps where the data is being exchanged. This can be done using shared memory or MPI in case of distributed processes.

3.1 Shared Memory Implementation

A single node comprising homogenous or heterogeneous, but shared memory can use global variables and threads to exchange the data (Fig. 4). This method is the simplest and probably the most effective way of implementing the algorithm.

3.2 Distributed Memory Implementation

When we consider multiple processes instead of multiple threads, the code becomes simpler on one hand as it does not have to be contained in a function that runs as a thread. On the other hand, it requires inter-process communication which makes it more complicated and typically slower. We have successfully implemented and tested the algorithm using MPI (Message Passing Interface). The fastest implementation uses shared memory based communication within a node to minimize the data turnaround and only one of the threads may be responsible for the inter-process communication (Fig. 5).

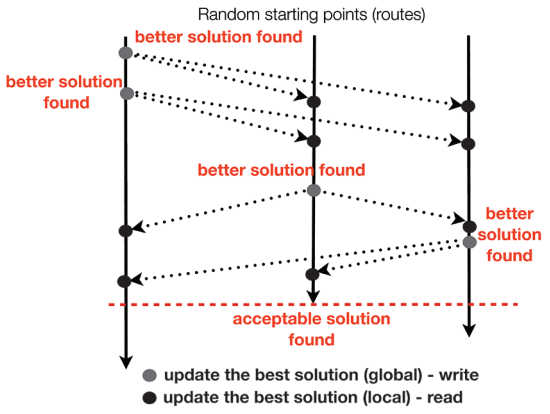

Fig.4 Shared-memory Parallelization Scheme

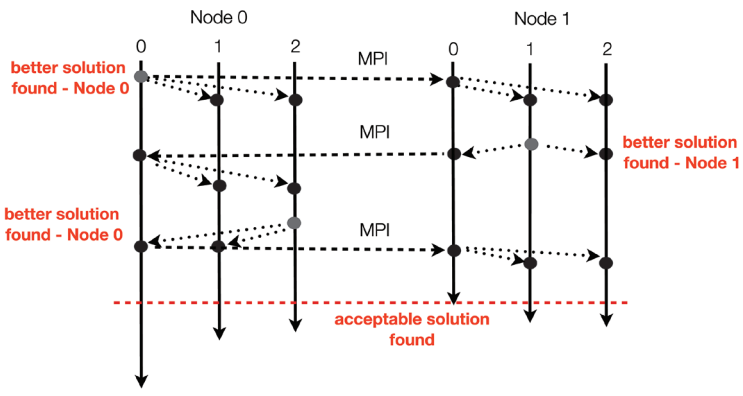

Fig.5 Distributed-memory Parallelization Scheme

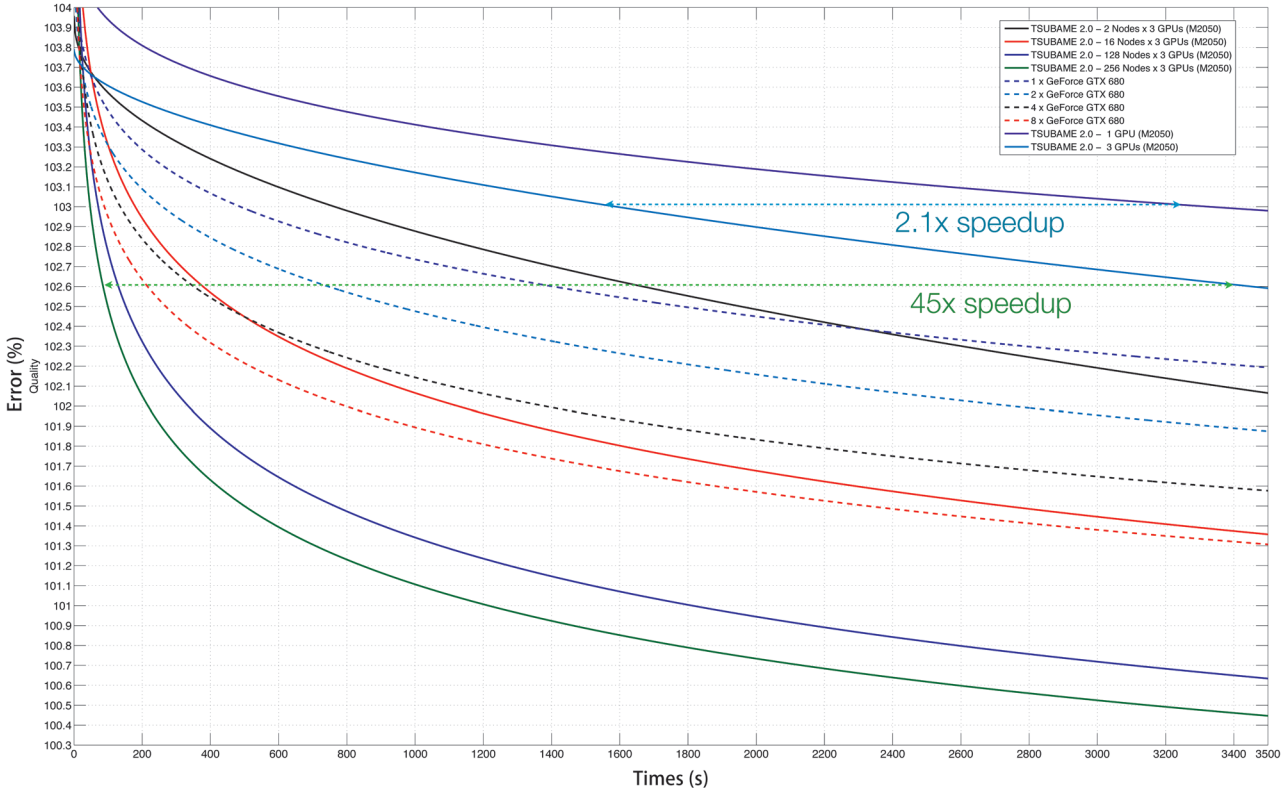

Fig.6 Results of Parallel ILS using TSUBAME 2.0

Results 4

We tested the shared version of our algorithm on a GeForce GTX 680 GPU. When we tested the distributed version, we used up to 256 nodes of TSUBAME 2.0 with three NVIDIA Tesla M2050s per node. The performances of the computations are plotted in Fig. 6. In this figure, time is plotted on the x-axis and the quality of the solution on the y-axis. The lower the result, the better the quality. As expected, the solution gradually improves over time with the ILS algorithm. Two distinct results can be compared by measuring the time needed to reach an equally good solution. In this way we performed the speedup measurement. We observed that the algorithm runs approximately 90 times faster when all 768 GPUs are utilized. In our opinion, the imperfect scalability of the algorithm is due to the communication between the nodes as well as time which is required to copy the data to and from the GPU.

Summary 5

In this article we have presented a high-performance Multi-GPU implementation of 2-opt local search in Traveling Salesman Problem. However, the main contribution of our work is the Parallel Iterated Local Search algorithm which can be used with other local search methods. It can be used to solve arbitrarily big problem instances using distributed GPU systems such as TSUBAME 2.0. We believe that it is a good base for more sophisticated approaches. Our algorithm solves the problem in a brute-force way, but due to the very high parallelism, the overall speed allows to tackle large TSP instances. The results show that the time needed to perform a single search operation can be decreased up to 90 times compared to sequential GPU implementation using a single TSUBAME 2.0 node. We believe that the algorithm presents strong-scaling features, but it is limited by network and CPU-GPU communication.

Acknowledgements

This work was supported by Core Research of Evolutional Science and Technology (CREST) project of Japan Science and Technology Agency (JST) and Grant-in-Aid for Scientific Research of MEXT Japan.

References

[1] Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2007)

[2] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, Chichester (1985)

[3] Johnson, D. and McGeoch, L.: The Traveling Salesman Problem: A Case Study in Local Optimization. Local Search in Combinatorial Optimization, by E. Aarts and J. Lenstra (Eds.), pp. 215-310. London: John Wiley and Sons, 1997.

[4] Garey,M.R.andJohnson,D.S.ComputersandIntractability:A Guide to the Theory of NP-Completeness. San Francisco: W.H. Freeman, 1979.

[5] Croes G. A.;A Method for Solving Traveling-Salesman Problems, Operations Research November/December 1958 6:791- 812;

[6] M. A. O'Neil, D. Tamir, and M. Burtscher.: A Parallel GPU Version of the Traveling Salesman Problem. 2011 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 348-353. July 2011.

[7] Dorigo, M. and Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53-66. April 1997.

[8] Fujimoto, N. and Tsutsui, S.: A Highly-Parallel TSP Solver for a GPU Computing Platform. Lecture Notes in Computer Science, Vol. 6046, pp. 264-271. 2011.

[9] Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library. ORSA Journal on Computing, Vol. 3, No. 4, pp. 376-384. Fall 1991.

[10] Rego,C.and Glover,F.: Local Searchand Metaheuristics. The Traveling Salesman Problem and its Variations, by G. Gutin and A.P. Punnen (Eds.), pp. 309-368. Dordrecht: Kluwer Academic Publishers, 2002.

[11] Lourenco, H. R. Martin, O. C. Stutzle, T.: Iterated Local Search, International series in operations research and management science, 2003, ISSU 57, pages 321-354

[12] Karp, R. Reducibility among combinatorial problems: In Complexity of Computer Computations. Plenum Press, pp. 85- 103. New York, 1972

[13] Tsai, H.; Yang, J. Kao, C. Solving traveling salesman problems by combining global and local search mechanisms, Pro- ceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Vol.2, pp. 1290-1295.

[14] Pepper J.; Golden, B. Wasil, E. Solving the travelling sales- man problem with annealing-based heuristics: a computational study. IEEE Transactions on Man and Cybernetics Systems, Part A, Vol. 32, No.1, pp. 72-77, 2002

[15] NVIDIA CUDA Programming Guide http://docs.nvidia.com/cuda/index.html

[16] Helsgaun, K.; An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research, 2000, vol 126, pages 106-130

[17] Nilsson, Ch.; Heuristics for the Traveling Salesman Problem, Linkoping University, pages 1-6

● TSUBAME e-Science Journal vol.10
Published 11/18/2013 by GSIC, Tokyo Institute of Technology ©
ISSN 2185-6028
Design & Layout: Kick and Punch
Editor: TSUBAME e-Science Journal - Editorial room
Takayuki AOKI, Thirapong PIPATPONGSA,
Toshio WATANABE, Atsushi SASAKI, Eri Nakagawa
Address: 2-12-1-E2-6 O-okayama, Meguro-ku, Tokyo 152-8550
Tel: +81-3-5734-2085 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