TSUBAME Grand Challenge program
Adopted projects on TSUBAME3.0

TSUBAME Grand Chalenge Summary

This program is only chance to use all nodes of TSUBAME3.0 exclusively, because TSUBAME3.0 is shared by thousands of users. There are two categories:

Category A The large scale application aims high peak-performance. All of TSUBAME3.0 nodes are available.

Category B The large scale application aims scientifically meaningful results.

Table Number of Adopted Projects in the TSUBAME Grand-Challenge Program

	2018	2017	2016	2015	2014	2013	2012	2011	Total
A	1	2	2	3	3	1	4	7	23
B	2	1	1	4	4	2	0	2	16
Total	3	3	3	7	7	3	4	9	39

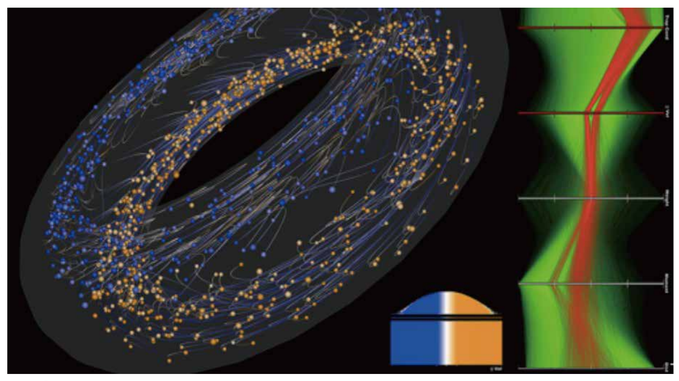
We started this program since FY2011, and keep on carrying out twice in each year.
Under this program, we have adopted total 39 fruitful projects, some of which were awarded such as AMC Gordon Bell Prize in SC11.

Projects in Fall 2017

Deep Learning for Fusion Simulation Using the Fusion Recurrent Neural Network

FRNN (Fusion Recurrent Neural Net) is the deep learning framework developed for fusion energy applications. It is implementing a distributed data-parallel approach to train deep neural networks (in particular, stacked LSTMs). With this approach, a replica of the model is kept on each worker, processing different mini-batches of the training dataset in parallel. The model parameters from each worker are collected using MPI, and synchronized via parameter averaging with learning rate adjusted after each epoch to improve convergence. This produces a global set of parameters, which are then broadcasted to each model replica. The stochastic gradient descent (SGD) method is used for large-scale optimization with parallelization via mini-batch training to reduce communication costs. In FRNN, we integrate Keras (TensorFlow and Theano backends) and MPI to enable training across multiple GPU nodes using high-speed interconnects.

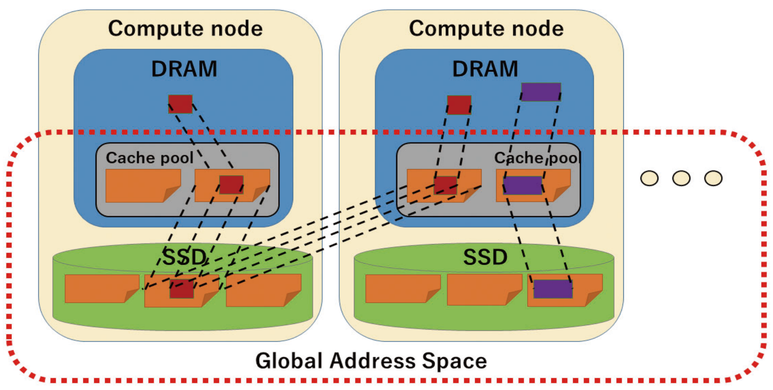
The starting phase of this proposed 2017 TSUBAME 3.0 Grand Challenge project will be to initiate scaling studies on the powerful TSUBAME 3.0 supercomputer featuring thousands of Pascal P-100 GPU' s will begin with the goal of replicating the excellent FRNN scaling observed on "Titan" as well as Pascal P-100 systems at NVIDIA and Princeton University. We look forward to exploring challenges associated with engaging the full capability of TSUBAME 3.0. In particular, since the FRNN software uses the steepest gradient descent (SGD) with mini-batch training to reduce communication costs, it will be very interesting to observe the behavior of this DL software with scaling studies on TSUBAME 3.0 to examine if the convergence rate saturates/decreases with increasing mini-batch size to thousands of P-100 GPU' s.



Fusion Simulation Using the Fusion Recurrent Neural Network

Realizing Extreme Large-Scale Matrix Computations with a Memory Management Library Utilizing Fast Flash SSDs

The objective of our project is to achieve extremely high-speed and large-scale matrix computation on TSUBAME3.0. Here we focus on large scale Cholesky decomposition, which is an important kernel in solving semi-definite programming (SDP) problem. In order to support even larger matrices than aggregated host memory capacity of available nodes, we harness large capacity of NVMe SSDs, which is 8 times larger than DRAM capacity in TSUBAME3.0. This should be achieved with less development effort; thus we have developed a global address space runtime library, named vGASNet. On vGASNet, the aggregated capacity of SSDs distributed among compute nodes are virtually visible as a single address space. For high-performance computing, caching mechanism using DRAM is necessary. Moreover, for scalable data movement, it supports "cooperative caching mechanism" to avoid bottleneck in access congestion. On top of vGASNet, we also have implemented the parallel Cholesky decomposition algorithm. While it is based on data-driven execution and tile-based dynamic scheduling to suppress global synchronization, the implementation is relatively simple and clean owing to the global address space view. Inside each task, we use CUBLAS and MAGMA library in order to harness high performance of NVIDIA P100 GPUs. Taking this co-design approach, we aim to achieve peta-scale matrix computation.



Global Address Space Model of vGASNet

mSMS: A New DSM System for HPC

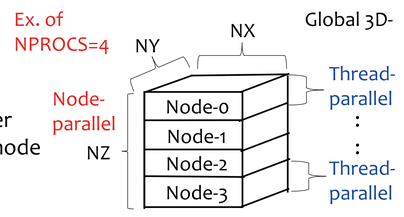
Hiroko Midorikawa (Seikei University) midori@st.seikei.ac.jp

Key Features

mSMS realizes a highly productive parallel programming environment.

- Seamless virtual shared memory which is fully accessible by all threads in a cluster.
- C-compatible address pointer-based programming for shared data is available for multi-node parallel processing.
- Global shared data are transparently distributed over nodes in a cluster by C-based programming (MpC). *Directive-based API (SMS-Mint)* is also available in conjunction with OpenMP and OpenACC for multi-core & multi-node computing.
- *Remote data preload API* is available for sub-array-data prefetch before the computing of the sub-array to achieve higher performance in data-parallel comp.

```
shared double A[NZ][NY][NX]::[NPROCS][1][1] (0, NPROCS);  // Global 3D- Arrays, A and B, distributed
shared double B[NZ][NY][NX]::[NPROCS][1][1] (0, NPROCS);  //   in z-dimension to multiple nodes in a cluster
main()
{   double (*src)[NY][NX];  double (*dst)[NY][NX];  double (*tmp)[NY][NX];    // pointers to a global 3D-array
    sms_startup(&argc, &argv);    // mSMS is started up
                  :       // A & B arrays Initialization
    src = A;   dst = B;   // set pointers to A and B arrays
    for (t = 0; t < NT; t++) {      // time step loop
     #pragma Mint parallel for  //  Node parallel processing in a cluster
     #pragma omp parallel for  //  Thread parallel  processing in one node
      for (z =1; z < NZ-1; z++)
       for (y = 1; y < NY-1;  y++)
        for (x = 1; x < NX-1; x++) {        // 7-point Stencil Calculations
          dst[z][y][x]=0.4*src[z][y][x]+
            0.1*(src[z-1][y][x]+src[z+1][y][x]+src[z][y-1][x]+src[z][y+1][x]+src[z][y][x-1]+src[z][y][x+1]);
        }
     sms_sync_drop();  // Execution &  memory consistency Sync. (Each node discards or reflects cached pages)
     tmp = dst;    dst = src;     src = tmp;       //  switch  A and B arrays  ( swap src & dst pointers )
   } // time step loop end
   sms_shutdown();    // mSMS is finalized
}
```

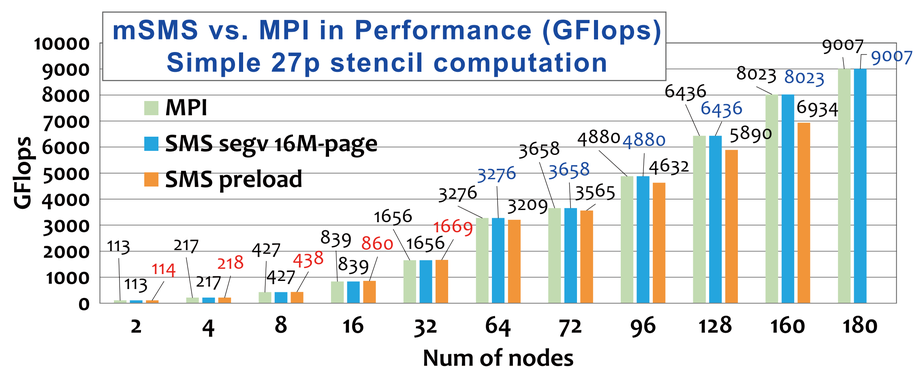


The mSMS program for 7-point stencil comp.

- High performance: mSMS incorporates dedicated sms-system-threads for efficient communication to achieve the comparable performance of MPI programs.

Results

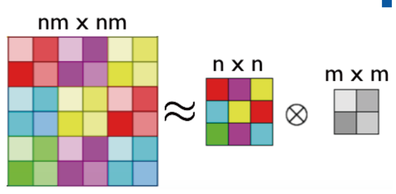
- *mSMS realized 30TiB-address-space and 23TiB-global-arrays **by using 180 nodes (9900 threads)** in Tsubame3.0.*
- *It achieved MPI comparable perf.*

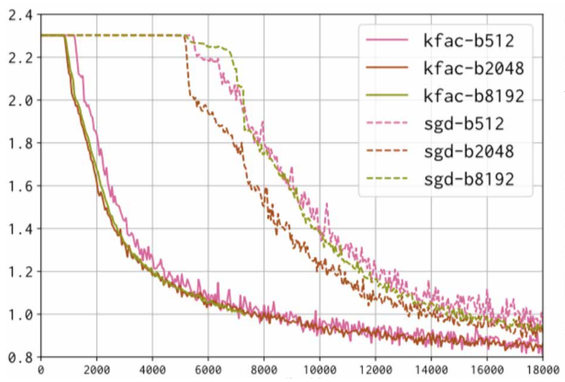


Training ImageNet on 2048 GPUs

Rio Yokota, Kazuki Osawa, Hiroki Naganuma, Hiroyuki Otomo, Mohamed Wahib, Alexandr Drozd, Yosuke Oyama, Yohei Tsuji, Keita Yashima, Hitoshi Sato (Tokyo Institute of Technology)

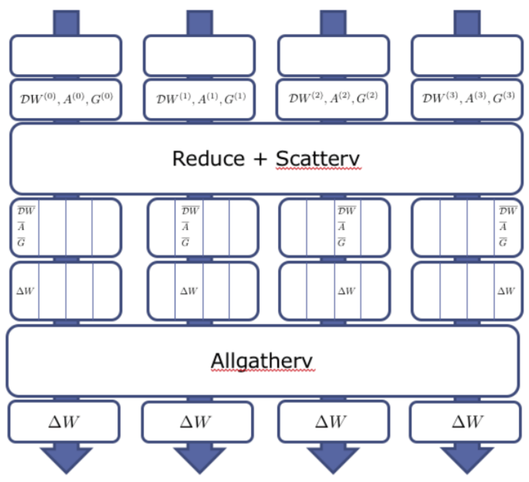
Second Order Optimization for Deep Learning





As deep neural networks increase in size the amount of data and time to train them become prohibitively large to handle on a single compute node. Distributed deep learning on thousands of GPUs forces the batch stochastic descent methods to operate in a regime where the increasing batch size starts to have detrimental effect on the convergence and generalization. We investigate the possibility of using second order optimization methods with proper regularization as an alternative to conventional stochastic gradient decent methods.

Hybrid data and model parallelism



Second order optimization methods require the communication of the Hessian matrix. The present method reduces the communication significantly through Kronecker factorization and the use of reduce-scatter and allgather collectives to swtich between data parallel and model parallel execution. The optimal implementation of allreduce operations has the same communication pattern, so the present method simply inserts the computation of the Hessian between the two collective operations.