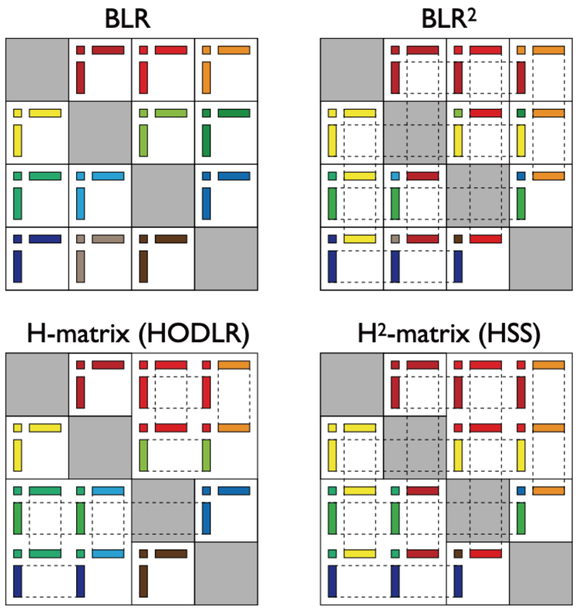
Applications on TSUBAME3.0
Fast Algorithms and Large Scale CFD

Fast Algorithms for HPC and Deep Learning

Structured Low-Rank Matrices

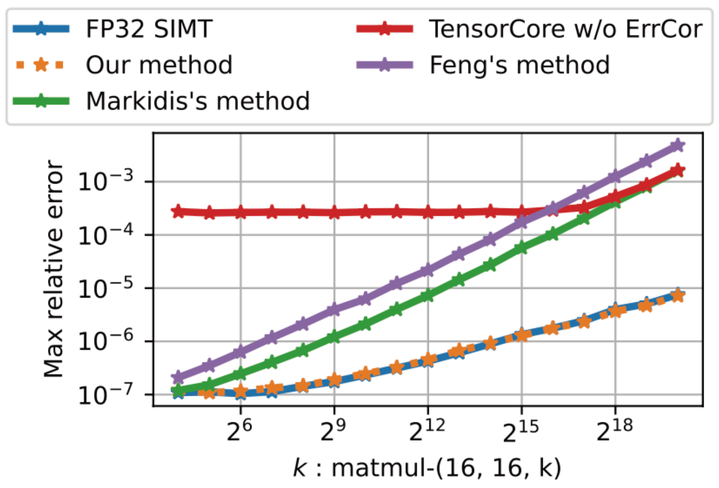
Presentation Thu 4:00pm SC22 Room C140-142



Structured low-rank matrices can reduce the complexity of dense matrix multiplication and factorization from $O(N^3)$ to $O(N)$. There are many variants of structured low-rank matrices, and we are investigating the trade-offs between computational work, parallelism, communication, load imbalance to achieve the best performance. We have developed an O(N) factorization method that has no dependency on trailing submatrices. We have a technical paper on Thursday that describes this approach.

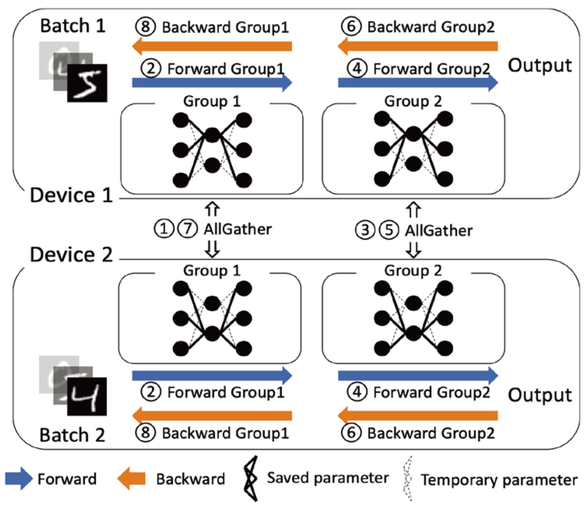
Error Correction for TensorCores

Poster Available SC22 C1-2-3



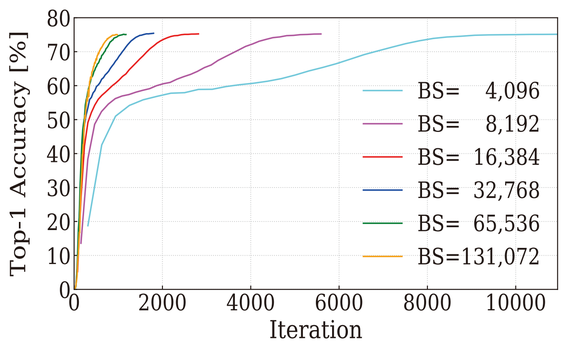
TensorCores multiply two 16bit matrices and accumulate into a 32 bit matrix. This conversion to 16bit results in a significant loss in precision. The use of an auxiliary 16bit matrix to store the mantissa loss is known to partially correct this error. However, the round-to-zero in TensorCore accumulation is another source of error that has not been accounted for in previous work. We develop a novel method that also corrects for this error, which allows us to exactly match an GEMM in single precision, while exploiting the compute capability of TensorCores.

Memory Efficient Deep Learning



The size of deep neural networks continues to grow rapidly, where models like GPT-3 and Megatron-LM exceed the memory capacity of a single GPU. The trend in computer architecture where the arithmetic throughput is growing faster than memory capacity, suggests that memory consumption is a critical issue in deep learning. We reduce the memory usage of deep neural networks by scattering, recomputing, and offloading the model parameters and activations.

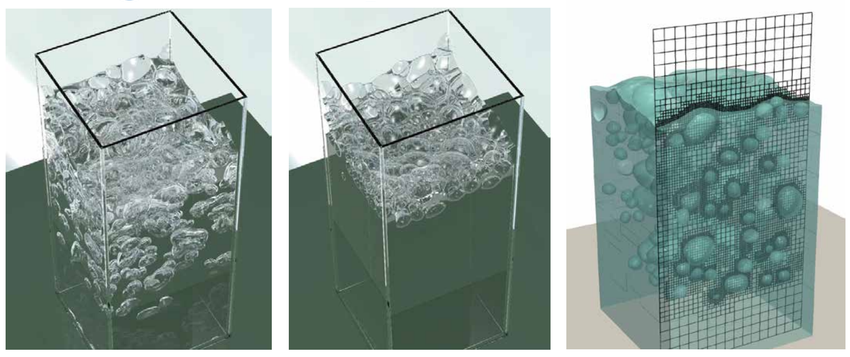
Training ImageNet in 2 minutes on 2048 GPUs



Data-parallel training of deep neural networks suffers from the large-batch problem, where the generalization gap increases as the mini-batch size increases proportional to the number of GPUs. Using the Kronecker Factorization mention on the right, we are able to use a second order optimization method with minimum overhead, which allows us to retain the convergence even for extremely large batch sizes. We trained ImageNet in 2 minutes on 2048 GPUs, using a batch size of 131,072.

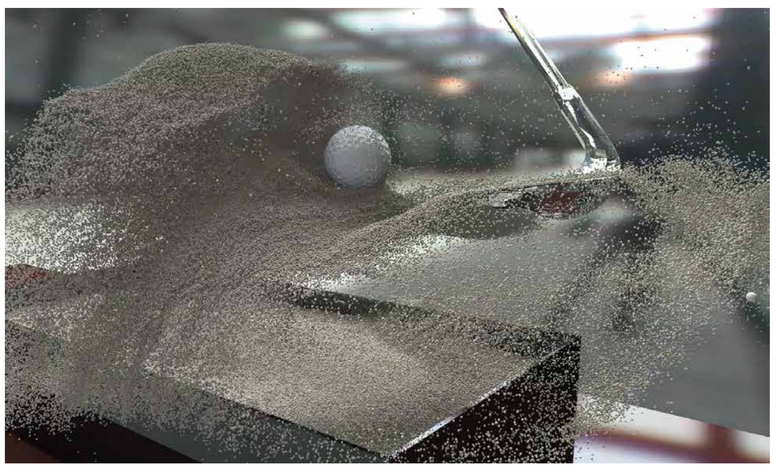
Large-scale Mesh-based and Particle-based Simulations

A Large-scale Foam Simulation using a Multi-phase Field LBM and AMR



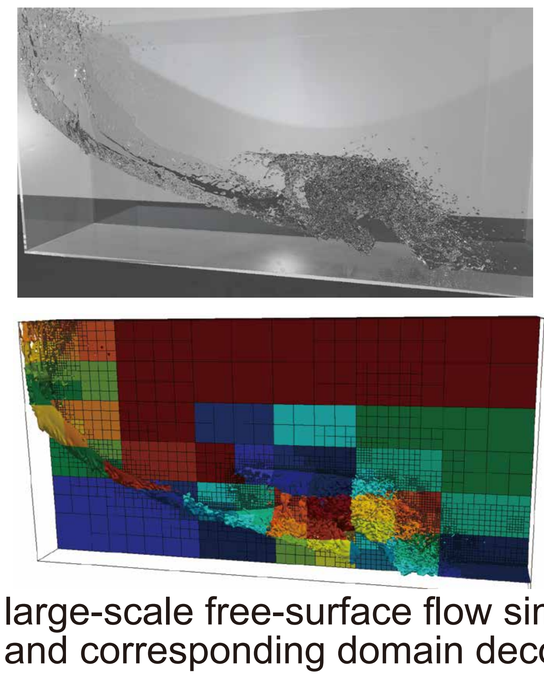
Foam formation with a stable thin liquid film is very difficult to simulate using conventional methods due to the limitation of mesh resolution that can be used. We addressed this issue by using a Multi-phase Field Lattice Boltzmann Method and Adaptive Mesh Refinement. Using the Multi-Phase Field LBM, we can prevent a “numerical coalesce” phenomenon that leads to bubble break-up. Adaptive Mesh Refinement has been introduced so that the thin liquid film can be simulated efficiently. Herein, we demonstrate a simulation of foam formed from 200 air bubbles using our proposed method.

Large-scale Granular Simulation



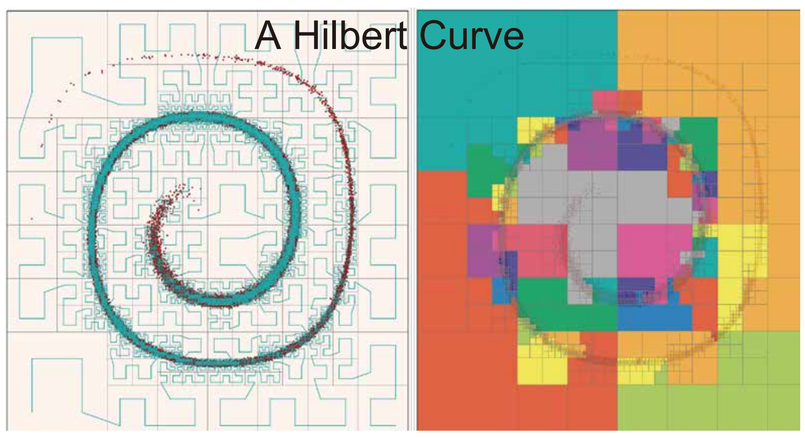
Discrete element method (DEM) is often used to simulate granular dynamics and its simple algorithm with the contact interaction is suitable for GPU computing. However so many particles are included and the particle distributions are changing in time and space. A dynamic domain decomposition has to be introduced for multiple-node computing. In a bunker shot, the sand wedge does not hit to the golf ball directly and transferring the force through the sand to the ball in order to reduce the impact. In this simulation, 16.7 millions of DEM particles are used to represent the dynamics of the sands with 256 GPUs.

AMR for Multi-phase Flows



Simulations for multi-phase flows require high-resolution grids to capture phenomena at the interface. By using the adaptive mesh refinement (AMR) method, which dynamically adapts high-resolution grids to interfaces, computational cost and memory usage are reduced. The spatial distribution of a computational load change in time; therefore, dynamic domain partitioning using a space-filling curve is introduced for multi-GPU computing to assign an equal number of grid points to each GPU. The figures show the large-scale free-surface flow simulations for the dam-breaking process and corresponding domain decomposition for 64 GPUs.

Dynamic Load Balancing using A Space-filling Curve



For large-scale particle-based simulation and Adaptive Mesh Refinement (AMR), it is a critical issue to achieve computational load balance and equal memory usage on multiple compute nodes. A domain partitioning in terms of a space-filling curve(SFC) is one of promising candidates and it is recognized that a 1-dimensional mapping of 3-dimensional space by cutting the equal length. Due to low cost of SFC domain partitioning, it is suitable for frequent re-partitioning in the simulations of unsteady phenomena.

https://www.gsic.titech.ac.jp/sc22