Research towards Future Supercomputer for Everybody

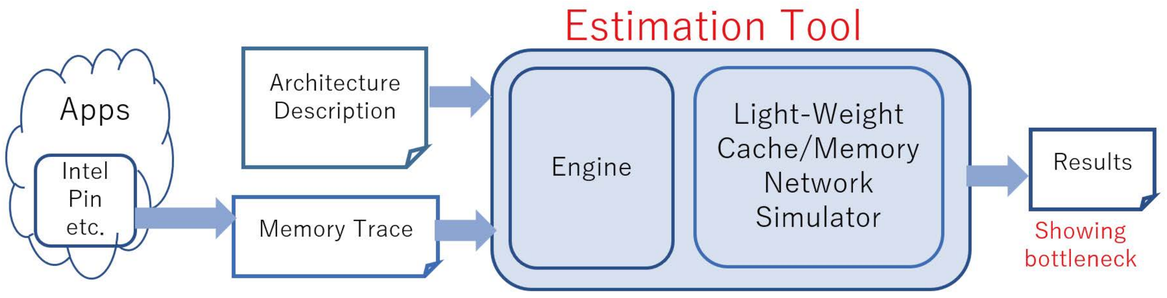
Exploring Next-Gen Architecture with Complex Memory Hierarchy

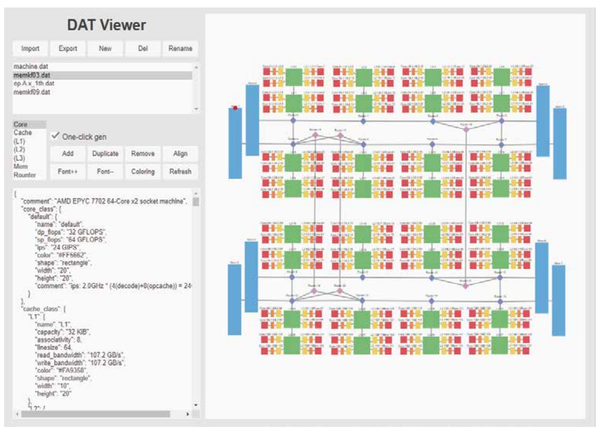
Supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

To achieve higher performance and larger memory capacity on future architectures, we need to explore next-gen memory hierarchy. It is important to understand application performance, not only synthetic benchmark, on variety of memory hierarchy. For this purpose, we are developing an estimation tool of memory performance of future architecture, whose features are:

Light weight: it takes history of the entire application execution as input, which can be too heavy for cycle-wise simulators.

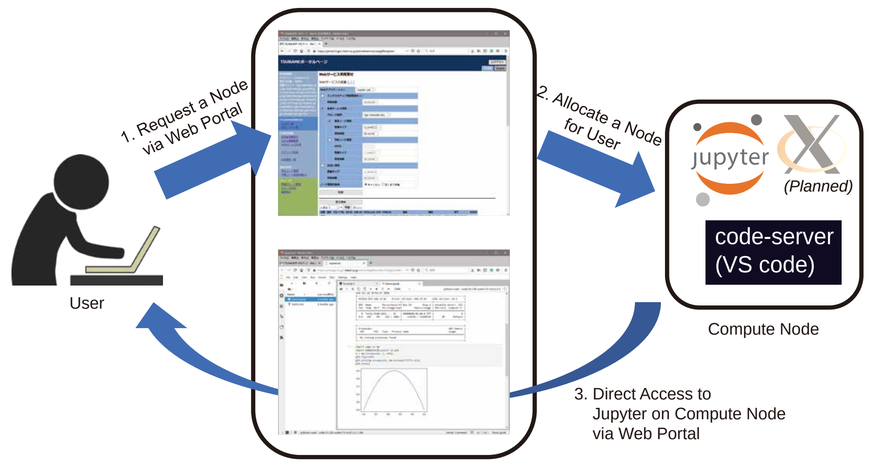
To support complex structure: it considers placement of many cores and topology including chip-lets and memory chips.





GUI tool to describe the architecture to be estimated.
The user describes the memory hierarchy as a graph that consists of memory objects, caches and CPU cores.

Web-based Interactive Access to Supercomputer Nodes



Supercomputers are traditionally designed to execute large and non-interactive jobs with a batch scheduler. This style does not match the users who want to interact with compute nodes: debugging, visualization, and education of novice users in the classroom.

In TSUBAME3.0 supercomputer, we introduced two new features to satisfy such demands:

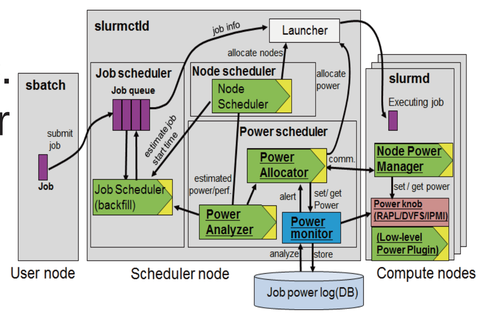
Interactive use only nodes: We spare four nodes as dedicated nodes for shared interactive use. Users can run interactive jobs without waiting for job execution, even if the compute nodes are filled with batch jobs. Performance might not be optimal as the nodes are shared with multiple jobs, but still acceptable for such interactive usage.

Web-based access to Applications on compute nodes: Novice users can use command-line shell and Python console running on high-performance TSUBAME nodes without any complicated knowledge of Linux, such as SSH key-pair authentication. Currently, Jupyter Lab and code-server (web-based clone of VS code) are supported, and web-based VNC server is planned.

Power Management Framework for Petascale Supercomputers

Power consumption is expected to be a first class design constraint for developing petascale supercomputers. To make effective use of limited power budget, hardware over-provisioning is proposed. We are creating a power-aware resource manager.

Scheduler overview:



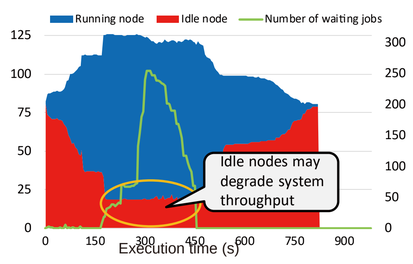
Job Scheduler for Power-Constrained HPC Systems

Objective: Providing a common resource management tool for hardware-overprovisioned HPC systems
- Maximize throughput of a system within a given power budget
- Provide extendibility and flexibility (plug-in based interface)

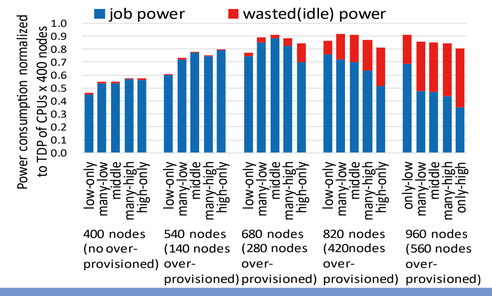
Plug-in based interfaces and functionalities:
Power Analyzer plug-in estimates job characteristics
Power Allocator plug-in controls power-cap dynamically
Node Power Manager plug-in provides a way to manage the power consumption within compute nodes
Power monitor provides continuous monitoring of power consumption of every compute node

Experiment: Measurement of power consumption on HA8000 HPC System (965nodes)

Runtime power consumption



Power utilization

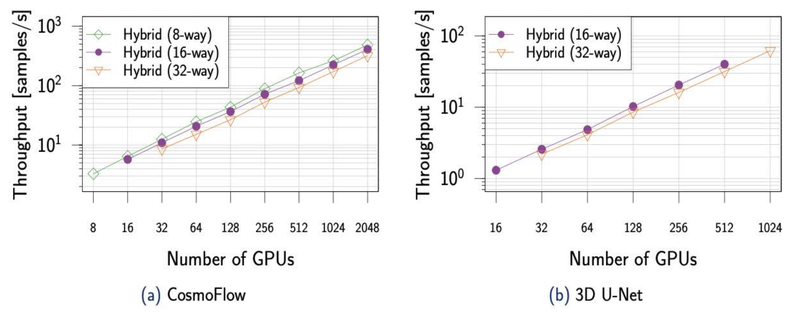


JST CREST Project on Scalable Deep Learning

This work was supported by JST CREST Grant Number JPMJCR19F5, Japan.

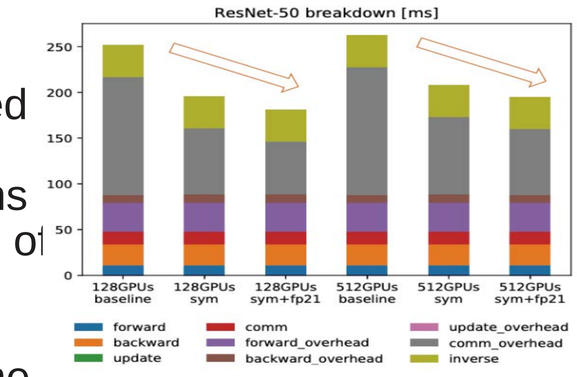
The Case for Strong Scaling in Deep Learning: Training Large 3D CNNs with Hybrid Parallelism

We present a new capability for spatially partitioning the training of 3D convolutional neural networks. We study the impact of data size and present comprehensive performance and scaling results of two different huge 3D CNNs, the CosmoFlow network and 3D U-Net, including training a single model with up to 2048 V100 GPUs. Furthermore, we demonstrate an order-of- magnitude improvement in the prediction quality of the CosmoFlow network.



A Systematic Performance Analysis of Second-order Optimization using K-FAC

We perform second order optimization on deep neural networks using Kronecker factored approximate curvature (K-FAC). Various performance optimizations were performed including the use of fp16 gradients and fp21 Fisher matrices, hierarchical ring AllReduce communication, and the use of stale Fisher matrices. We achieved 2 minutes training of ResNet-50 / ImageNet-1K with 2048 GPUs by Distributed K-FAC.



https://www.gsic.titech.ac.jp/sc21