Performance Tuning of Deep Learning Framework Chainer on the K computer.

Akiyoshi KURODA*
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
kro@riken.jp

Kiyoshi KUMAHATA*
RIKEN Center for Computational Science
Kobe, Hyogo, Japan

Syuichi CHIBA†
Platform Software Business Unit, Fujitsu Limited.
Numazu, Shizuoka, Japan

Katsutoshi TAKASHINA†
Technical Computing Solutions Unit, Fujitsu Limited.
Kobe, Hyogo, Japan

Kazuo MINAMI*
RIKEN Center for Computational Science
Kobe, Hyogo, Japan

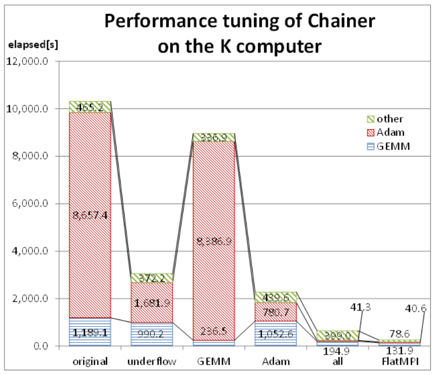

Figure 1: Elapsed time on the all tuning step.

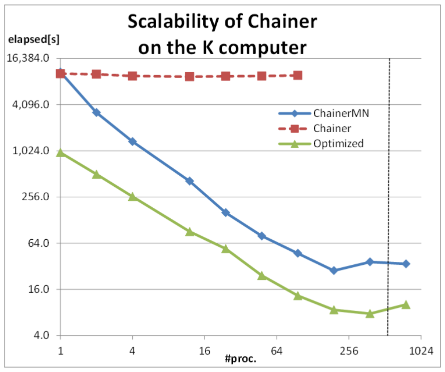

Figure 2: Elapsed time of the scaling result.

CCS CONCEPTS

• **Theory of computation → Theory and algorithms for application domains**; *Machine learning theory*; Boosting; • **General and reference** → *General conference proceedings*.

KEYWORDS

The Supercomputer Fugaku, Deep Learning, Chainer, Performance Optimization, The K Computer

Recently GPUs has become a popular platform for executing deep learning (DL) workloads. We revisit the idea of doing DL on CPUs, especially massively parallel CPU clusters (supercomputers). In anticipation of deployment of the Supercomputer Fugaku with much more DL capable CPUs, we investigate which optimizations can be already done using the K computer, current leadership computing facility and predecessor to the Supercomputer Fugaku. We use Chainer as a deep learning framework of choice. Chainer expresses the hierarchical structure of deep learning using Python, and all calculations can be realized using numpy without special libraries. Many of the cost was the calculation of the square root and the arithmetic when the filter was updated and activation functions.

These operations are not optimized when calculated using numpy and are particularly slow on the K computer. By replacing the kernel with software pipelining and SIMD optimization by Fortran library, the kernel elapsed time was improved to 1/11.08 and total elapsed time was improved to 1/4.54[fig.1]. Moreover, by optimizing floating point underflow exception when building Python, total elapsed time was improved to 1/3.39. Generally gemm convolution cost is high in the DL calculations, By replacing the SSL2 gemm library called by Python with the thread-parallel version, section elapsed time was improved to 1/5.03, the total elapsed time was improved to 1/1.15, and the performance efficiency of gemm convolution was improved about 70.05% [1]. Python control part has no thread scalability. By dividing the Python procedure by data process parallelization using ChainerMN [fig.2], the total elapsed time was improved to 1/2.24. As a result of these optimizations, the overall speed ratio was 36.4 times, and the efficiency reached 35.9%.

There are some limitations on the use of Chainer on the K computer. It is necessary to prepare the learning data beforehand and to stage-in the data to an appropriate storage system. Moreover, since Python is in the shared storage, it takes time to load the library. However, I believe that we will be able to use the supercomputer Fugaku for deep learning sufficiently as well as GPU.

REFERENCES

[1] Akiyoshi Kuroda, Kiyoshi Kumahata, Syuichi Chiba, Katsutoshi Takashina, and Kazuo Minami. 2019. Performance Tuning of Deep Learning Framework Chainer on the K computer. *ISC2019* Research Poster, PR (2019), 28.