Building the LLM model on the supercomputer Fugaku toward the AI for Science

Fujitsu Ltd.
Takahide Yoshikawa

History of our Supercomputer System

FUJITSU

Technologies based on inherited and accumulated from the past

World' s Fastest Prize
(Nov. 1993,TOP500)
Gordon Bell Prize
(1994, 1995,1996)

Vector

F230-75APU

First Supercomputer
in Japan (1977)

VP Series

Scalar

NWT*1

VPP500

VPP300/700

VPP5000

AP1000

AP3000

Cluster

PRIMEPOWER
HPC2500

PRIMERGY
RX200

HX600

PRIMEQUEST

SPARC
Enterprise

FX1

PRIMERGY
BX900

PRIMERGY
CX1000

PRIMEHPC FX10

PRIMEHPC FX100

K computer*2

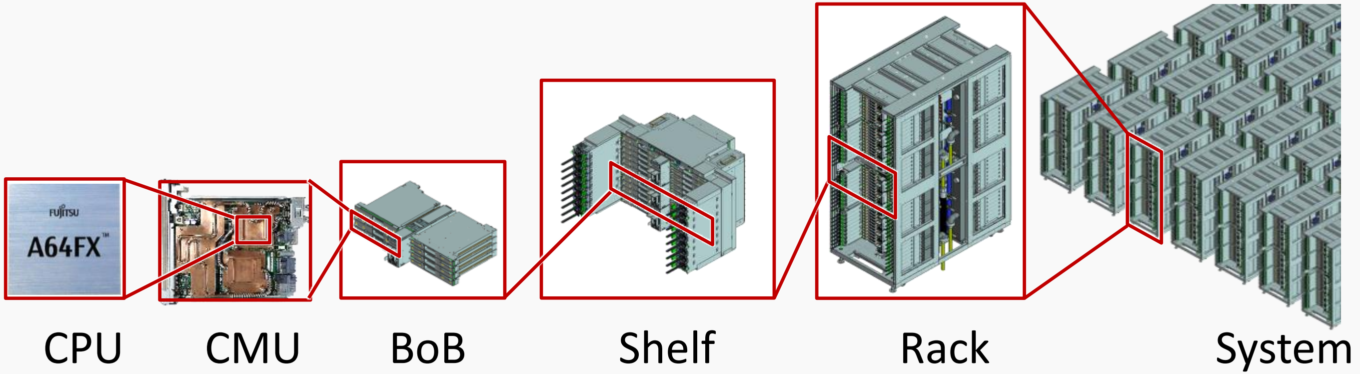
World' s Fastest Prize
(Jun/Nov, 2011,TOP500)
Gordon Bell Prize
(2011, 2012)

Fugaku*2

World' s Fastest Prize
(Jun/Nov 2020, Jun/Nov 2021, TOP500)

*1 NWT Numerical Wind Tunnel
Co-development with NAL
*2 Co-development with RIKEN

Fugaku System



Unit	# of CPUs	Description
CPU	1	Single socket node with HBM2 & Interconnect
CMU	2	CPU Memory Unit: 2x CPU
BoB	16	Bunch of Blades: 8x CMU
Shelf	48	3x BoB
Rack	384	8x Shelf (Front 4, Back 4)
System	158,976	432Racks×384CPU＝165,888≠158,976 396Racks are Full, 36Racks are Half(192CPUs)

Fugaku System Specification

Fugaku System	
# of Rack	432 Racks
# of Nodes	158,976 Nodes
Total Length of Interconnect Cable	≧900 km
Footprint	1,920 m^2

Peripherals		
CTRL, Management	PRIMERGY	≧140
Network	Ethernet Switch	512
	InfiniBand Switch	58

Applications of Fugaku

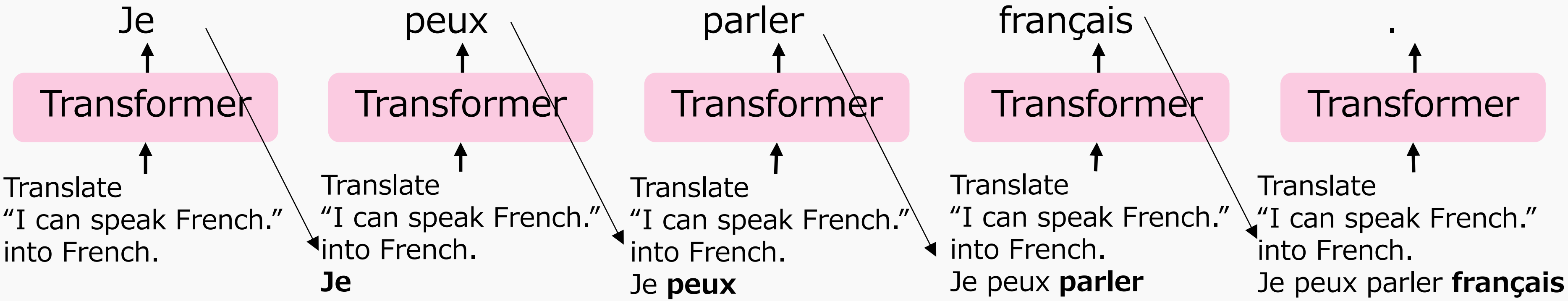
- High Performance Computing for Science
  - Computer Simulations:
    Fluid Dynamics, Molecular Dynamics,
    Electro Magnetic, Quantum Chemistry etc.
- AI/ML
  - Accelerating 13B parameter LLM on Fugaku
  - Training 400B Japanese tokens in 4 months using 13,824 nodes
  - Achieving an average score of 5.5 on the Japanese MT-Bench. This score is the highest among LLM for the Japanese language

Courtesy of RIKEN, Suzuki Motor Corp.

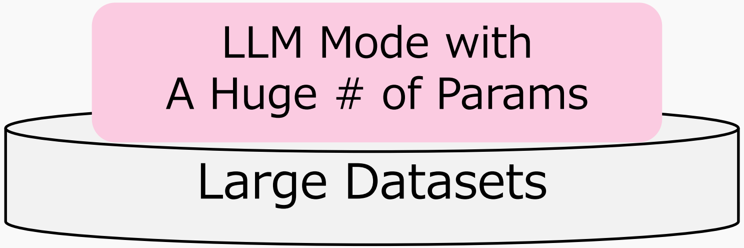
Courtesy of Kobe Univ, Honda R&D, RIKEN

Large Language Model (LLM)

- AI Model for natural language processing
  - Uses: Generating Text, Answering Questions, Summarizing Text, Translation etc.
  - Most of recent LLMs are **based on Transformer.**

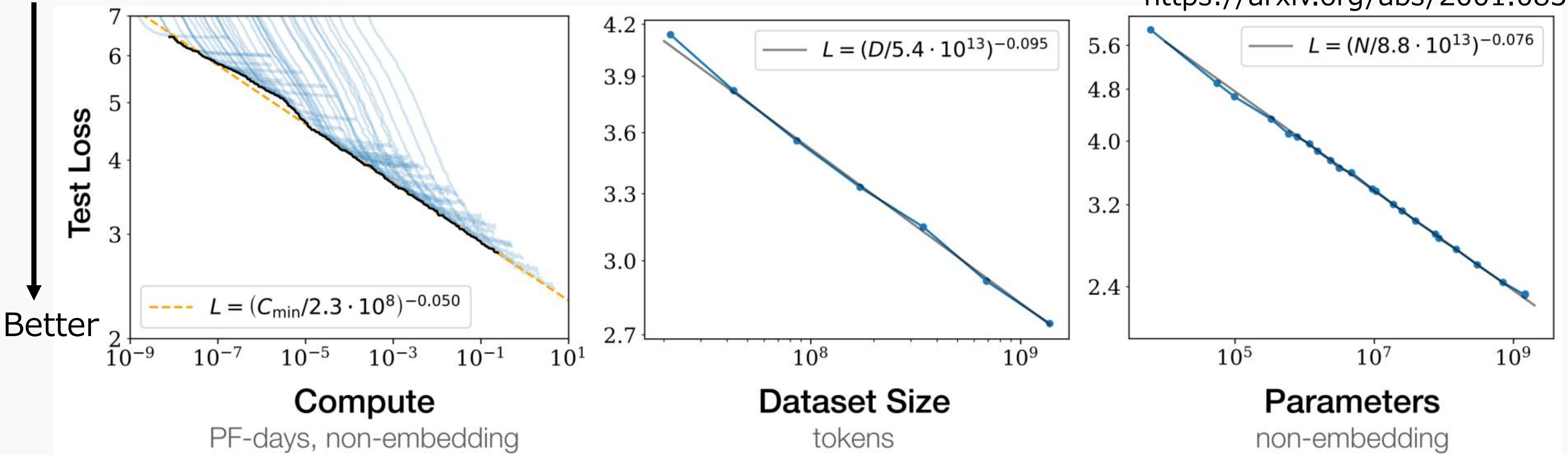


  - Models are trained by **large datasets.**
  - **A Huge # of parameters** are required.



Scaling Laws

https://arxiv.org/abs/2001.08361



- Claiming a power law relationship among
  Accuracy ∝ Computational Complexity, Amount of Data, Parameters
  - Creating **more accurate** models = **Scaling the factors** (Compute, Dataset, Params)
- Various high-accurate models with a huge # of params are proposed.
  - Requiring **enormous amount of computing resources**

How much resources are required?

FUJITSU

1. Huge amount of Computation

- Training of GPT-3 with 175B params
  - **1 month for A100x8,000**
- Training of GPT-4 with 1.8T params
  - More than 3 months for A100x20,000
  - **3.2 days for H100x100,000**

2. Huge amount of Memory

- Training of GPT-3 with 175B params
  - Storing params (FP16) requires **>350GB**
  - For training, requires 8 times larger memory = **Cannot fit into a single node**

Large-scale distributed training is essential.
(H100x100,000 needs 150MW=$124M/year)

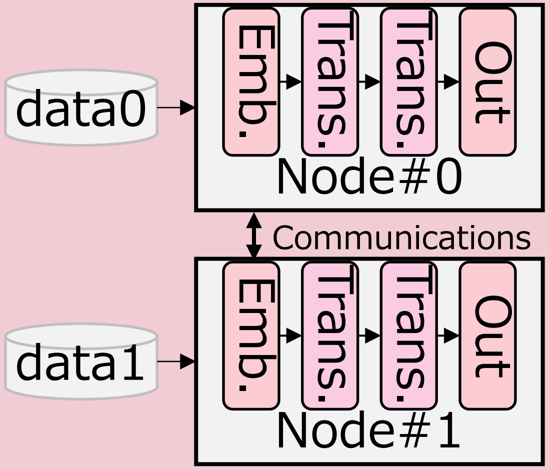
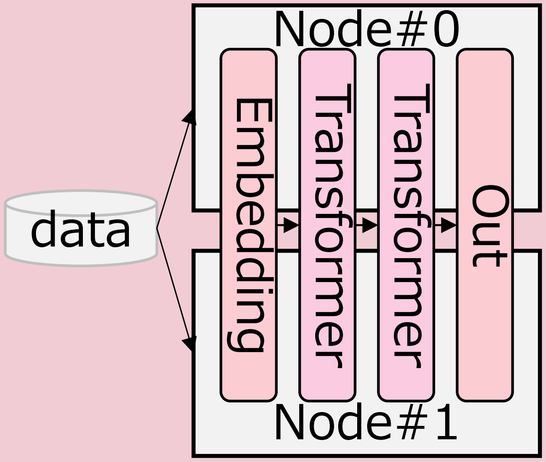
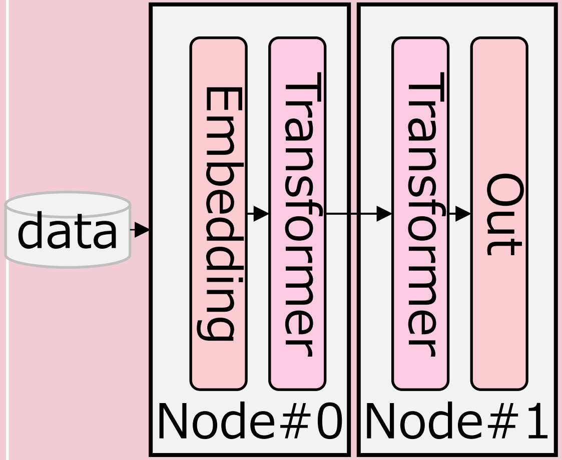
model		#params[B]	#tokens	#accelerators &training time	
GPT-3 [55]	May-2020	175	300B tokens	-	-
GShard [104]	Jun-2020	600	1T tokens	2048 TPU v3	4 d
Codex [105]	Jul-2021	12	100B tokens	-	-
ERNIE 3.0 [106]	Jul-2021	10	375B tokens	384 V100	-
Jurassic-1 [107]	Aug-2021	178	300B tokens	800 GPU	-
HyperCLOVA [108]	Sep-2021	82	300B tokens	1024 A100	13.4 d
FLAN [67]	Sep-2021	137	-	128 TPU v3	60 h
Yuan 1.0 [109]	Oct-2021	245	180B tokens	2128 GPU	-
Anthropic [110]	Dec-2021	52	400B tokens	-	-
WebGPT [81]	Dec-2021	175	-	-	-
Gopher [64]	Dec-2021	280	300B tokens	4096 TPU v3	920 h
ERNIE 3.0 Titan [111]	Dec-2021	260	-	-	-
GLaM [112]	Dec-2021	1200	280B tokens	1024 TPU v4	574 h
LaMDA [68]	Jan-2022	137	768B tokens	1024 TPU v3	57.7 d
MT-NLG [113]	Jan-2022	530	270B tokens	4480 80G A100	-
AlphaCode [114]	Feb-2022	41	967B tokens	-	-
InstructGPT [66]	Mar-2022	175	-	-	-
Chinchilla [34]	Mar-2022	70	1.4T tokens	-	-
PaLM [56]	Apr-2022	540	780B tokens	6144 TPU v4	-
AlexaTM [115]	Aug-2022	20	1.3T tokens	128 A100	120 d
Sparrow [116]	Sep-2022	70	-	64 TPU v3	-
WeLM [117]	Sep-2022	10	300B tokens	128 A100 40G	24 d
U-PaLM [118]	Oct-2022	540	-	512 TPU v4	5 d
Flan-PaLM [69]	Oct-2022	540	-	512 TPU v4	37 h
Flan-U-PaLM [69]	Oct-2022	540	-	-	-
GPT-4 [46]	Mar-2023	-	-	-	-
PanGu-Σ [119]	Mar-2023	1085	329B tokens	512 Ascend 910	100 d
PaLM2 [120]	May-2023	16	100B tokens	-	-

https://arxiv.org/abs/2303.18223

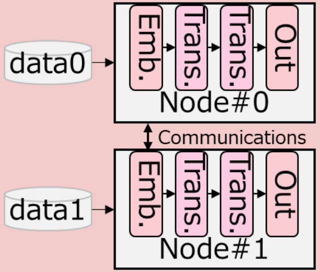
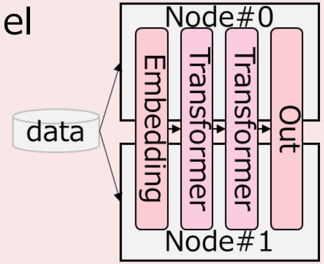
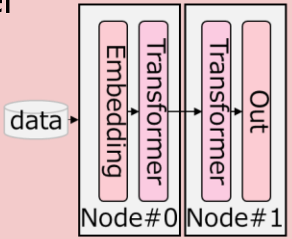
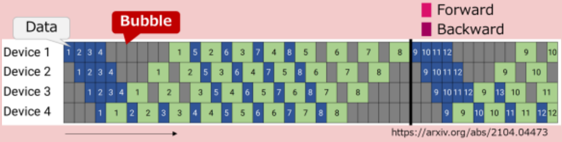
1. Characteristics of LLM
   (Large Language Model)
2. **Technology to Accelerate Training**
3. Accelerating Techniques for Fugaku LLM
4. Process and Achievements of Fugaku LLM
5. Summary

Three types of Distributed Training

FUJITSU

	Data Parallel	Model Parallel	
	Data Parallel	Tensor Parallel	Pipeline Parallel
			

Pros and Cons of the Distributed Types

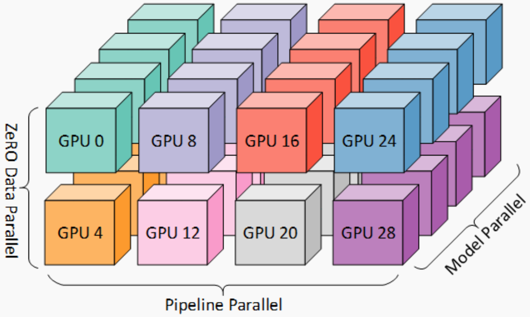
	Pros	Cons
Data Parallel 	Highly Scalable Easy to Distribute	**Model processing time does not become shorter** **Large Mini-Batch Problem** (If the split data volume is too large, the data becomes homogenized and thus learning accuracy decreases.)
Tensor Parallel 	**Model processing time becomes shorter** Memory usage = 1/N (N=# of nodes)	**Frequent Communications** Communication and computation cannot be overlapped.
Pipeline Parallel 	**Model processing time becomes shorter** Memory usage = 1/N (N=# of nodes)	**Pipeline bubble** 

3D Parallelism

- Combining 3 types of parallelism
  - Splitting the Model → Tensor + Pipeline Parallelism
  - Splitting the Dataset → Data Parallelism

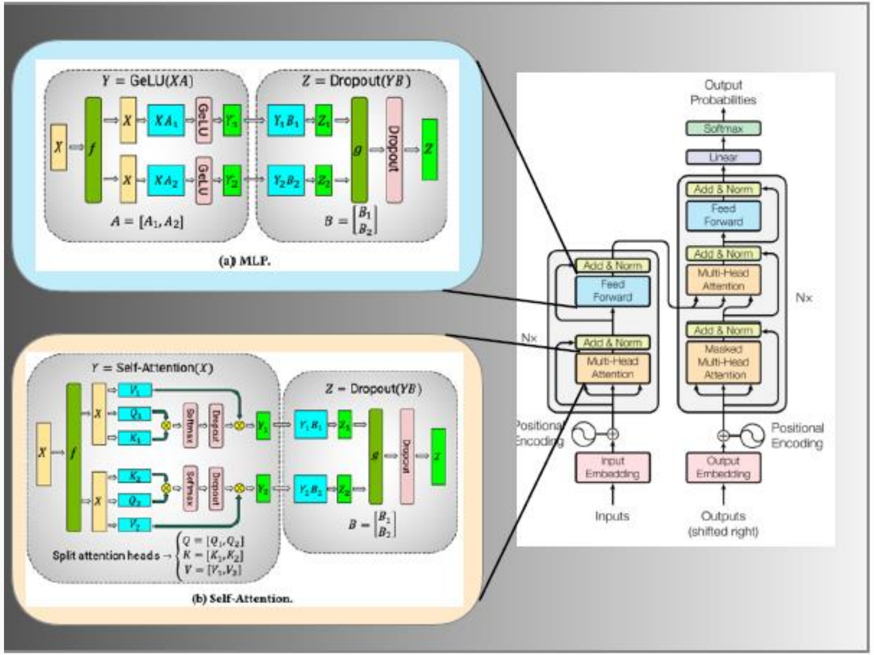
- Typical Combination of Parallelism on GPU clusters
  - Tensor: Keep **within a node** with multiple GPUs
  - Pipeline: Min # of GPU which has **enough memory to store the model**
  - Data: Make batch size as large as possible (**<4M Tokens**)

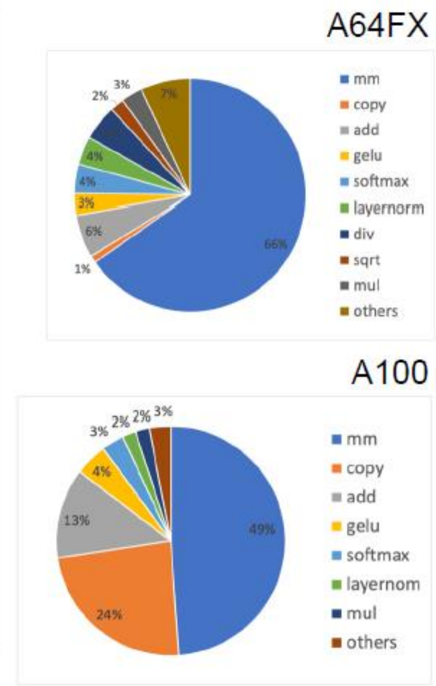
- Frameworks which support 3D parallelism
  - Megatron-LM + PyTorch
    - Megatron-LM https://github.com/NVIDIA/Megatron-LM
    - PyTorch https://pytorch.org/



Breakdown of GPT Computation Time

- Most of LLM **computation** includes **Dense Matrix Multiplication.**
  → 66% on A64FX (Fugaku), and 49% on A100



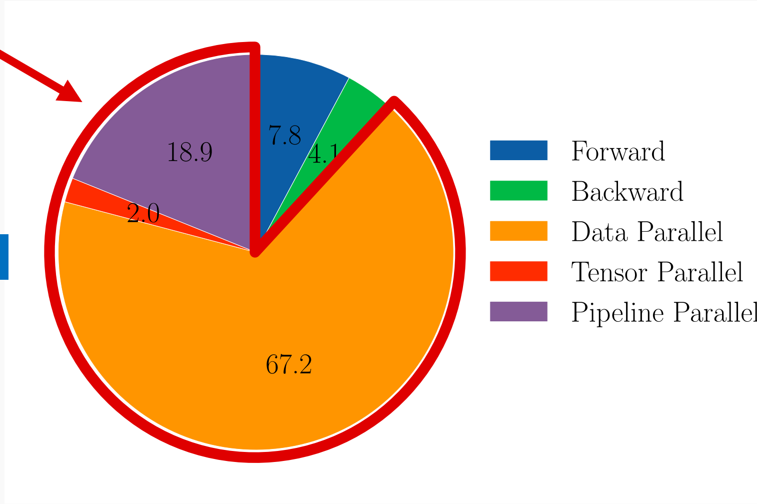


Rio Yokota and Shukai Nakamura,
https://hpcic-kkf.com/forum/2022/kkf_02/data/yokota_kkf2022-02_v2.pdf

Breakdown of GPT 3D Parallelism

- 90% of time is Parallel Processing
  - Reducing communication time leads to faster processing.
- For **Data Parallel** and **Tensor Parallel**
  Bottleneck = **All Reduce Comms.**
- For Pipeline Parallel
  Bottleneck = **Waiting Time (Bubble)**

→ Optimizations are implemented to reduce these bottlenecks.



Parallelization Ratio on Fugaku to train GTP-3 13B
Data Parallel=67.2%
Tensor Parallel = 2.0%
Pipeline Parallel = 18.9%
Data : Tensor : Pipeline ≒ 35:1:10

Optimizing Transformer on Fugaku

- The bottlenecks of Transformer:
  - **Dense Matrix Multiplications**
  - **Network Communications (All Reduce)**

Software Layers

Software Layers	
Transformer (GPT-x)	**Bottleneck Analysis** **→ Dense Matrix and Network Comms.**
Parallelization (Megatron-DeepSpeed)	**Communication Optimizations**
AI/ML Framework (PyTorch)	**Porting PyTorch to Fugaku**
Math Libraries	**Matrix Multiplications Optimizations**

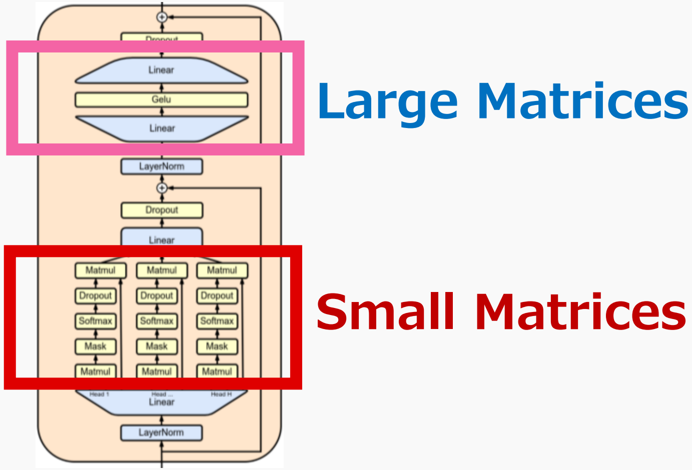
Accelerating Matrix Multiplications (1/2)

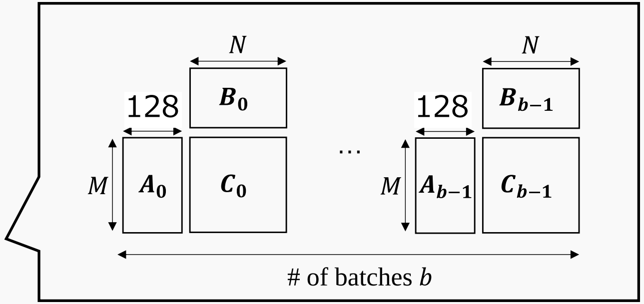
FUJITSU

- Most of LLM consists of two types of matrix multiplications
  - MLP: **Large Matrices**, which can be covered by BLAS
  - **Attention head**(<u>B</u>atch <u>M</u>atrix <u>M</u>ul.): Multiple **Small Matrix Mul.**

    ➡ These cannot be efficiently covered by BLAS.

Implemented optimized BMM to accelerate attention head.



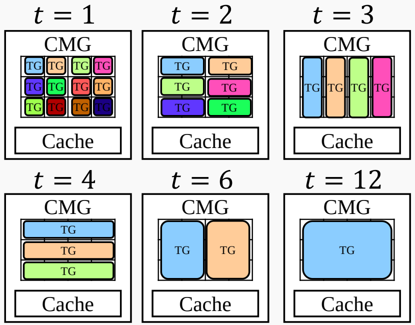


Implementation of Batch Matrix Multiplication for Large Language Model Training on A64FX CPUs, Hiroki Tokura et al., COOL Chips 27

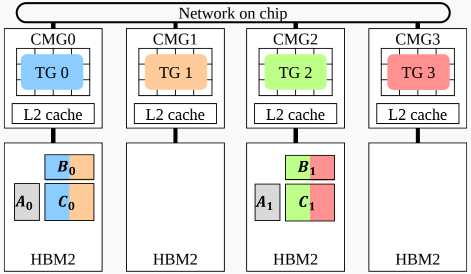
Accelerating Matrix Multiplications (2/2) FUJITSU

Implementation of Batch Matrix Multiplication for Large Language Model Training on A64FX CPUs, Hiroki Tokura et al., COOL Chips 27

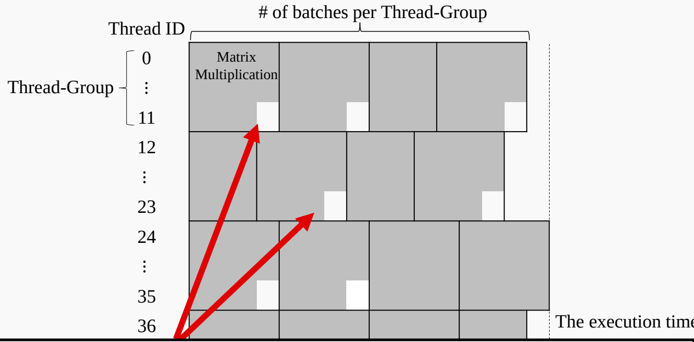
- A64FX has 48cores (4CMG).
- Partitioned a single matrix multiplication into multiple cores.
  - If unsuitably partitioned, **the waiting time to sync cores** becomes huge.
  - Once the LLM model is fixed, there are **only several patterns** of matrices. Thus, **the best splitting pattern can be determined beforehand**.



The patterns how to divide among 12 cores.



Example of how to partitioned to 2 cores (t=2).

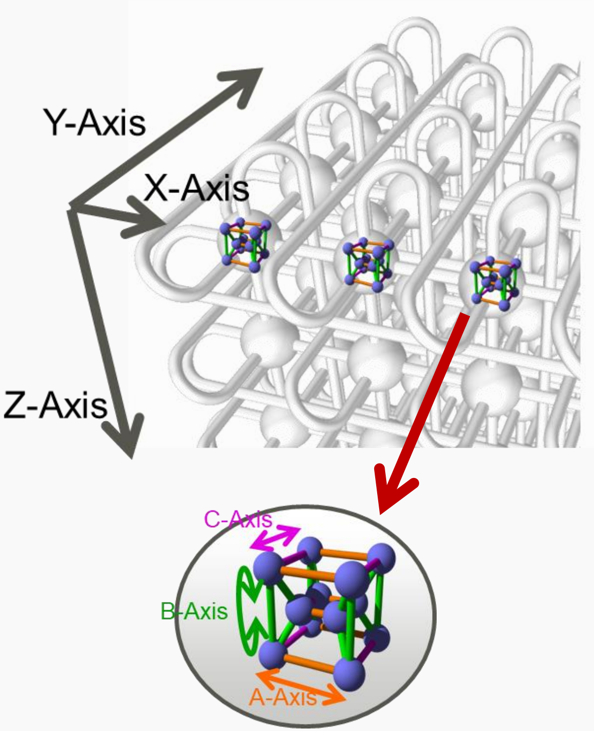


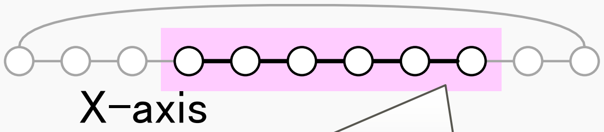
Waiting Time
If the size of matrices is not Nx of # of cores, **load imbalance** between cores occurs. Also, when handling **data that is too large for the cache**, it **results in waiting time**.

Accelerating All Reduce Comms. (1/2)

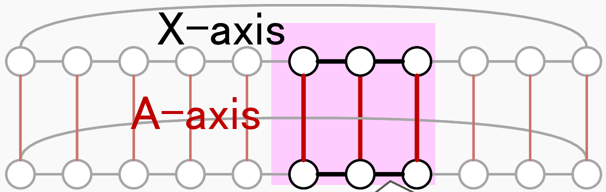
- Tofu Interconnect connects Fugaku Nodes.
  That is **6D Mesh-Torus** network (X,Y,Z,A,B,C).

- X and A, Y and B, Z and C are used in a couple.
  Even though it was partitioned,
  still the torus structure is retained.





The configuration is torus
but **the partition is mesh**

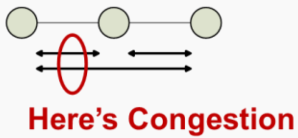


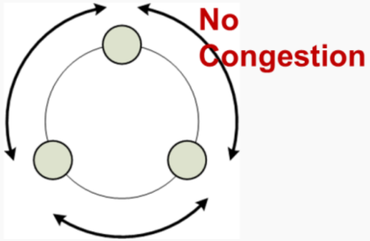
By using X and A axis,
the **mesh becomes torus**

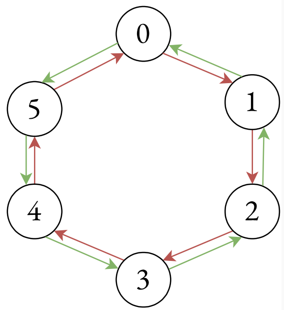
Accelerating All Reduce Comms. (2/2)

FUJITSU

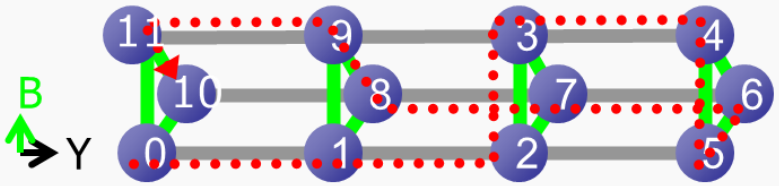
- If **collective comms**. are mapped onto mesh network, **congestions** will occur at the center.
- To prevent this, **bidirectional torus comms.** are applied.







- In order to secure such comm. paths using 2 axis (X and A, Y and B etc.), finding such **a non-overwrapped, unicursal path** is critical.
- This is achieved by using **a dedicated rank mapping algorithm**.



FUJITSU

1. Characteristics of LLM (Large Language Model)
2. Technology to Accelerate Training
3. Accelerating Techniques for Fugaku LLM
4. **Process and Achievements of Fugaku LLM**
5. Summary

Training LLM on Fugaku

FUJITSU

- Fugaku-LLM with 13B parameters
  - https://huggingface.co/Fugaku-LLM/Fugaku-LLM-13B
  - AI Model Architecture: GPT-2
    - Multiple hyperparameter patterns were publicly available.
  - # of layers: 40, Hidden size: 5184, **# of attention heads: 36**

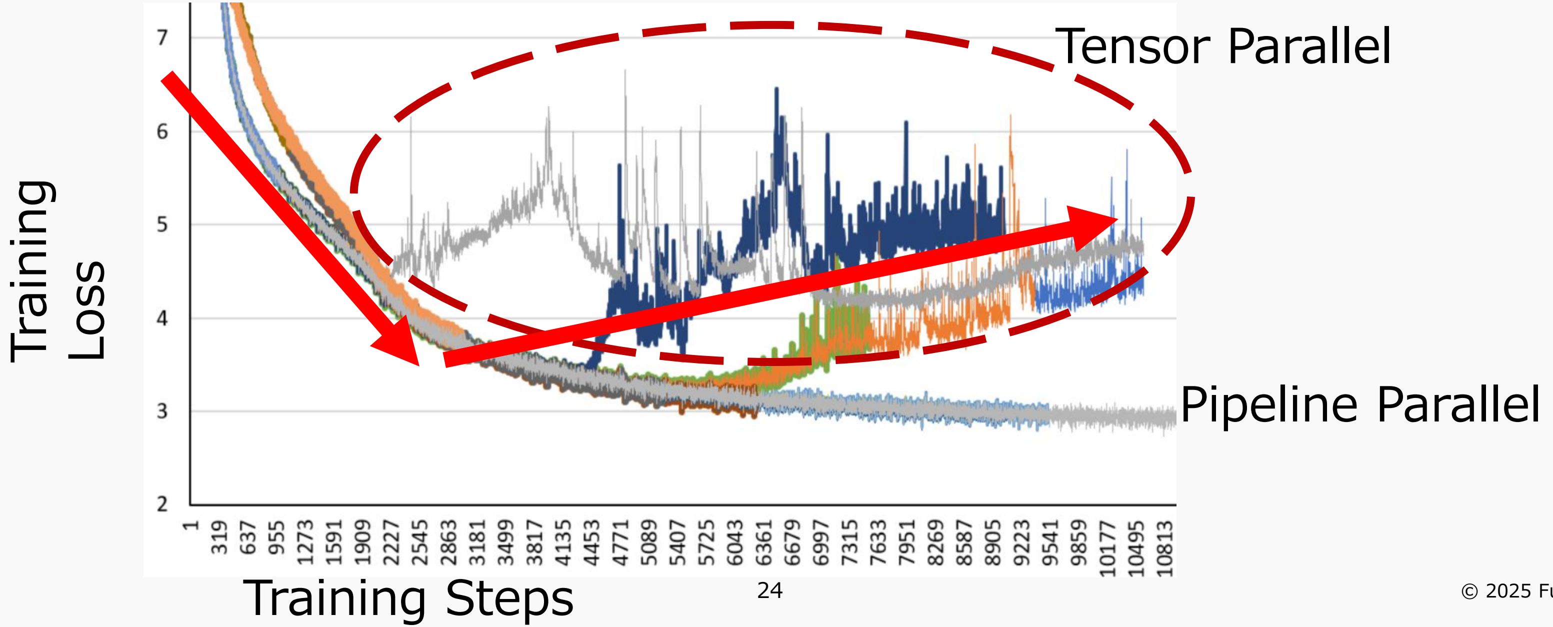
- Final AI Model Training Period: 4 months (From Dec. to Mar.)
  - Fine Tuning began in Feb.
- **# of Nodes: 13,824**

Data Parallel	288
Pipeline Parallel	8
Tensor Parallel	6

288(Data) x 8(Pipeline) x 6(Tensor) = 13,824 Nodes

Bug Found#1: Deterioration of LOSS in Tensor Parallel (1/2)

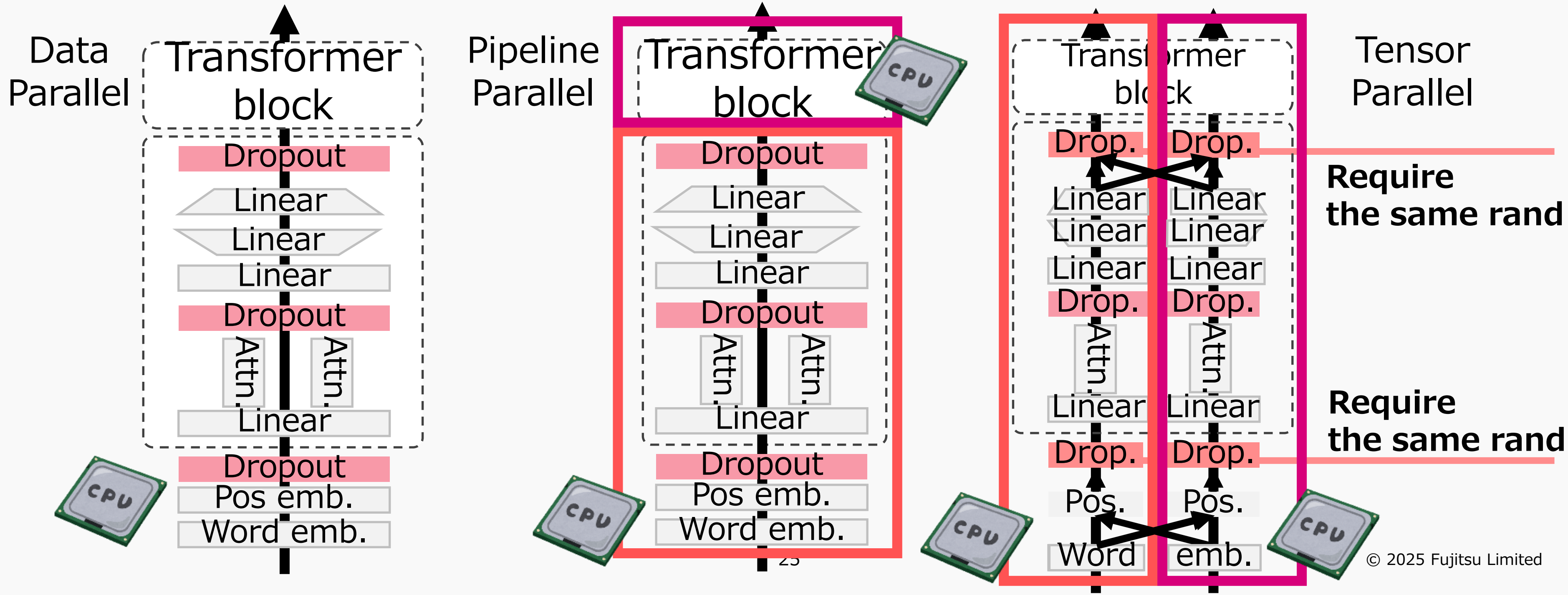
- When training with tensor parallelism, the loss worsens as the training steps progress.
  - At the beginning, both tensor's and pipeline's loss decrease.
  - Tensor's **begins to increase the loss** halfway through.



Bug Found#1: Deterioration of LOSS in Tensor Parallel (2/2)

- Cause: Random Num Generator's State Management among Nodes
  - Some **dropout layers are processed redundantly** in tensor Parallel.

➡ These redundant parts **require the same random number** sequence.

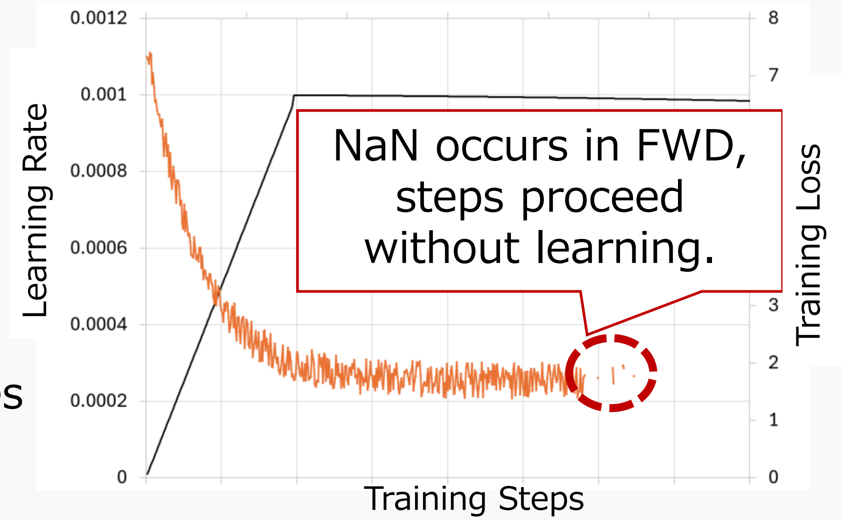


Bug Found#2: NaN occurs Frequently

- As training steps go, **NaN frequently appears** in loss.
- Cause: **Overflows in activation Function**
  - No inf guard was implemented in the optimization codes.

How to debug these?

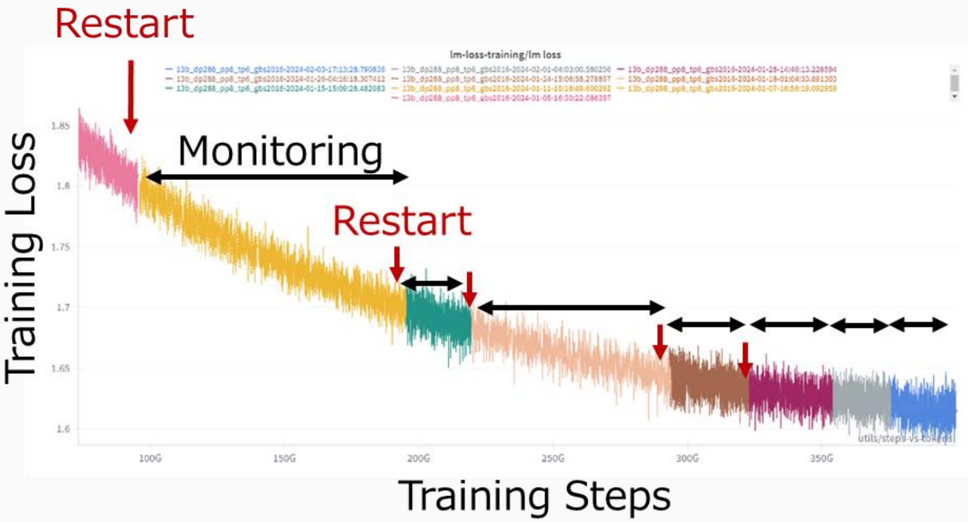
- Dumping data and states in the model
- Checking for equivalent points in multiple nodes
- Narrowing down using binary search



Pre-training of LLM on Fugaku

- To prevent training from stopping, **researchers must stay online**. (From Dec. to **Jan.**)
  - Even if **one node goes down**, training **restarts from the checkpoint**.
    ➡ Arranging a **24-hour shift**, and once stops, researchers restart the process.

- Why do we monitor?
  - Depending on the state of Fugaku and LLM, **prompt responses** are necessary.
    - Tuning Hyperparameters
    - Responding to crashed nodes
    - Adjusting # of nodes and rank mapping



Optimization Results: x6 Computation Speed and x3 Comm. Speed

Ported the DL framework Megatron-DeepSpeed to Fugaku and Accelerated the small matrix multiplications (BMM) 6x faster (110sec → 18sec), due to **the successful partitioning of the matrices** on multiple cores.

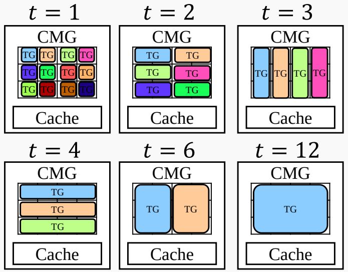
1693389241.318550480.fcc.pytorch.y.r1.13_for_a64fx.tar のプロファイル結果

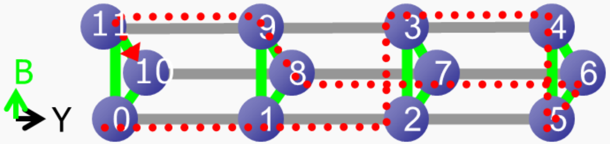
Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::bmm	18.07%	110.819s	18.08%	110.845s	24.055ms	4608
aten::bmm	18.17%	108.802s	18.17%	108.832s	23.618ms	4608
aten::bmm	18.53%	110.858s	18.53%	110.890s	24.065ms	4608
aten::bmm	19.15%	110.594s	19.16%	110.625s	24.007ms	4608
aten::bmm	18.33%	108.646s	18.34%	108.679s	23.585ms	4608

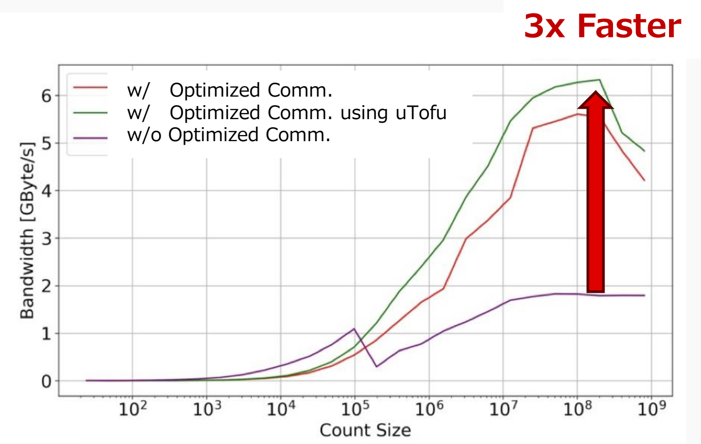
1701935794.711074240.fcc.pytorch.y.r1.13_for_a64fx.tar.gz のプロファイル結果

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::bmm	3.56%	18.273s	3.57%	18.302s	3.972ms	4608
aten::bmm	3.64%	18.394s	3.64%	18.423s	3.998ms	4608
aten::bmm	3.57%	18.154s	3.57%	18.185s	3.946ms	4608
aten::bmm	3.58%	17.959s	3.59%	17.990s	3.904ms	4608
aten::bmm	3.61%	18.341s	3.62%	18.373s	3.987ms	4608

Accelerated communication performance 3x faster by using **bidirectional torus communications**.







Achievements: LLM for the Japanese Language with 13B params

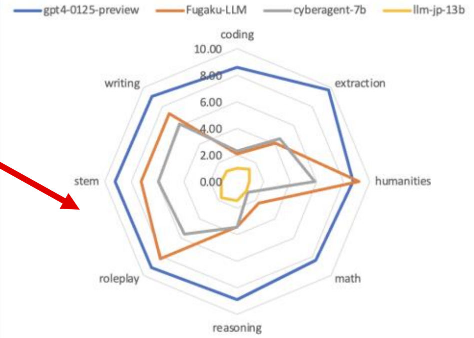
FUJITSU

Fugaku-LLM was trained on **380 billion tokens** using **13,824 nodes** of Fugaku, with about **60% of the training data being Japanese**, combined with English, mathematics, and code. (2 months for pre-training, 2 months for post training)

- Fugaku-LLM is trained from scratch using our own data, so the **entire learning process can be monitored**, which is superior in terms of transparency and safety.
- Fugaku-LLM is the best model among open ones that are developed in Japan.
- The **model shows a score of 9.18** in the humanities and social sciences tasks.
- The model will be able to perform natural dialogue based on keigo (honorific speech) and other features of the Japanese language.

The Score of Japanese MT-Bench, which is designed to evaluate Japanese LLMs.

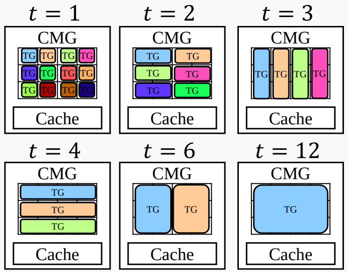
- Cyberagent (7B)
- Fugaku LLM (13B)
- GPT-4 (1.8T?)



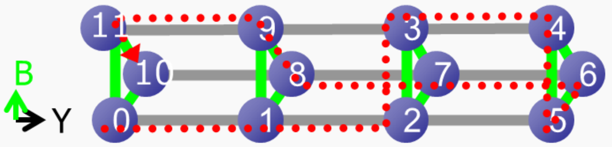
1. Characteristics of LLM (Large Language Model)
2. Technology to Accelerate Training
3. Accelerating Techniques for Fugaku LLM
4. Process and Achievements of Fugaku LLM
5. **Summary**

Summary

- Characteristics of LLM
  - **Scaling Laws** (Computation x Dataset x Params)= High Accuracy
- Distributed Training for LLM
  - Data Parallel, Tensor Parallel and Pipeline Parallel
    Combination of these = **3D Parallelism**



- Accelerating Techniques for Fugaku LLM
  - Small Matrix Mul.:
    Pre-analyze matrix shapes and optimally map on multiple cores → **6x faster**
  - All Reduce:
    Bidirectional Collective Comms. → **3x faster**



- Achievements of Fugaku LLM (13B Params)
  - Trained in 4 months using 13,824 nodes
  - Achieved a score of 9.18 in humanities category, which is higher than GPT-4

If you have any questions
please get in touch:

Takahide Yoshikawa
yoshikawa.takah@fujitsu.com