FUJITSU

shaping tomorrow with you

Current and Future Challenges of the Tofu Interconnect for Emerging Applications

Yuichiro Ajima
Senior Architect
Next Generation Technical Computing Unit
Fujitsu Limited

FUJITSU

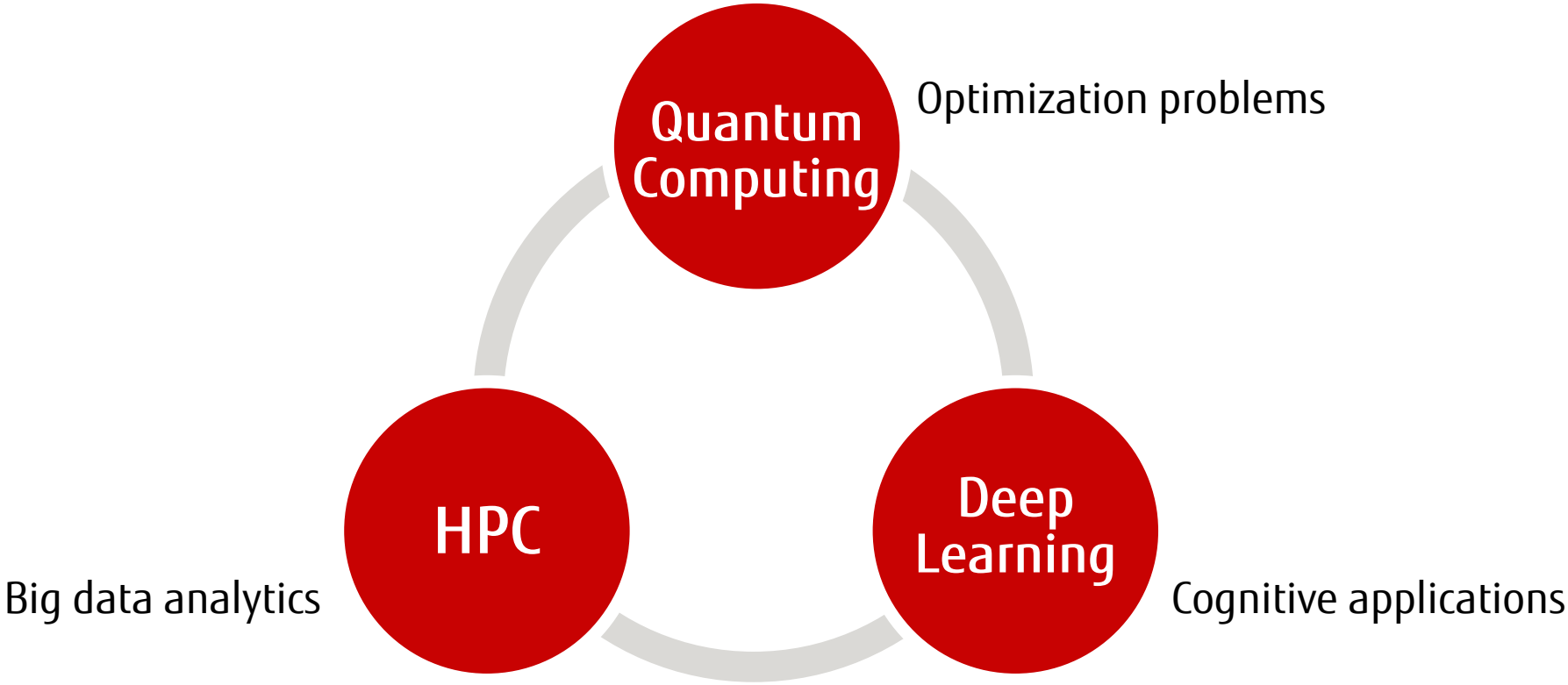
Index

- Fujitsu's strategy for emerging applications
- Recent projects conducted by JST-CREST teams
  - Graph500 benchmark (Prof. Fujisawa's team)
  - Inchworm (Prof. Nanri's team)
- Fujitsu's Deep Learning Unit
- Emerging technologies and future challenges

Emerging Applications

FUJITSU

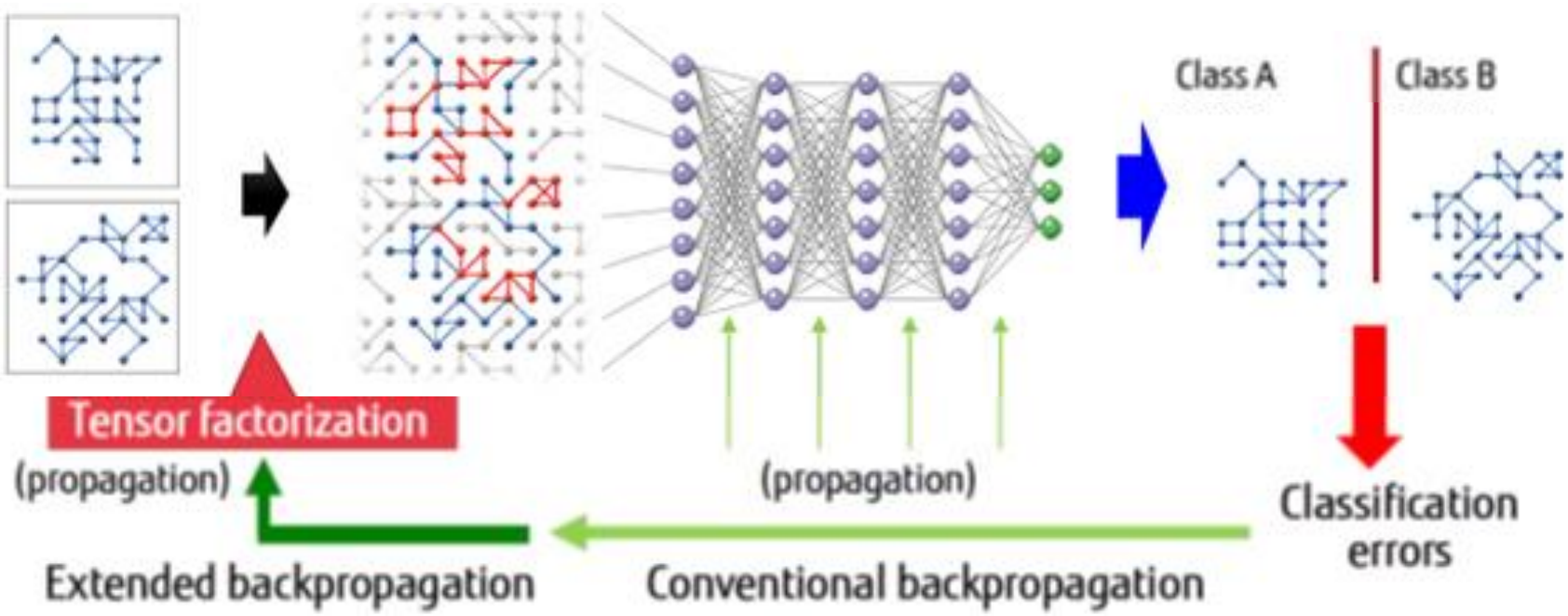
- Three core technologies in Fujitsu's vision
  - HPC, deep learning and quantum computing
- Three domains of emerging applications
  - Big data analytics, cognitive applications and optimization problems



Multi-Domain Application

FUJITSU

- Example: Fujitsu's 'Deep Tensor'
- A loose combination of graph analytics and neural network
  - Graph analytics: sparse matrix processing
  - Convolutional neural network: dense matrix processing

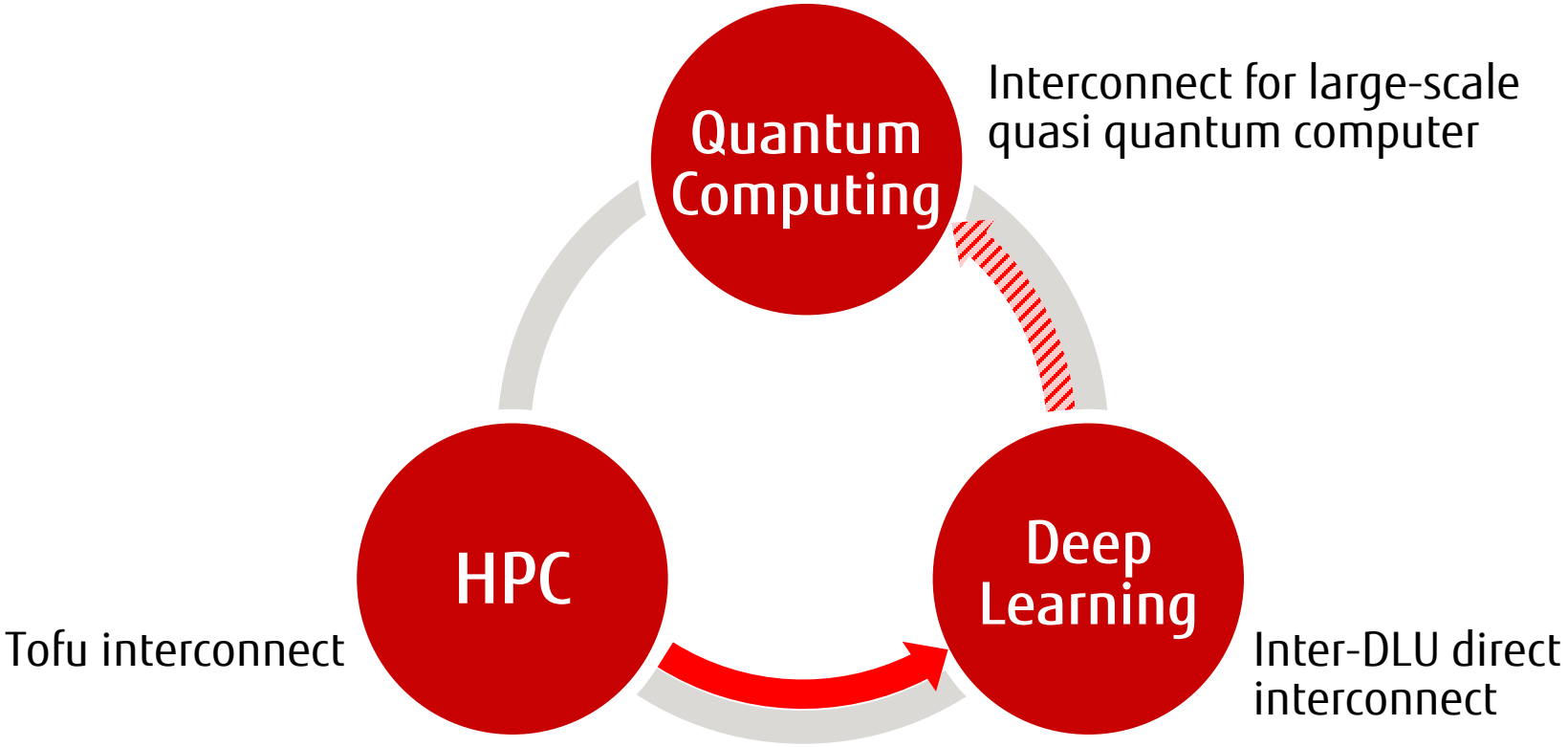


"Fujitsu Technology to Elicit New Insights from Graph Data that Expresses Ties between People and Things", http://www.fujitsu.com/global/about/resources/news/press-releases/2016/1020-01.html

Interconnect Development Strategy

FUJITSU

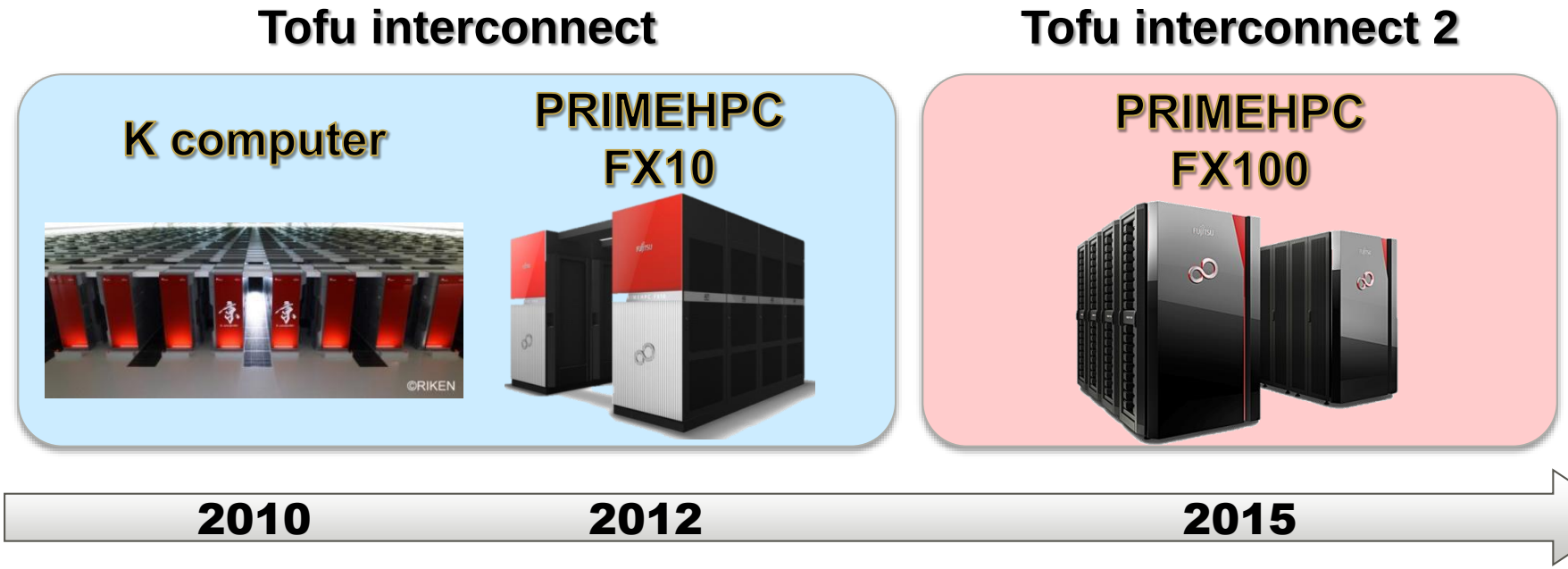
- Develop highly scalable domain-specific machines
  - For higher performance beyond the end of Moore's Law
- Fujitsu has developed Tofu interconnect as HPC interconnect
- High scalability technology will propagate to other machines



Tofu Interconnect

FUJITSU

- Highly scalable HPC interconnect
  - Tofu interconnect: developed for the K computer
  - Tofu interconnect 2: developed for PRIMEHPC FX100



- Physical 6D mesh/torus and virtual 1/2/3D torus network
- Fujitsu MPI provides topology-aware collective communication

Recent Work on Big Data Analytics

FUJITSU

- Two Recent projects conducted by JST-CREST teams
- Optimization of Graph500 benchmark (Prof. Fujisawa's team)
  - Parallel breadth-first search of huge graph
  - Topology-aware optimization of communication
- Parallelization of Inchworm (Prof. Nanri's team)
  - The first phase of de novo transcriptome assembly
  - Construct and traverse a dictionary of *k*-mer (DNA substring of length k)
  - Uses distributed data structure APIs for productivity

FUJITSU

Index

- Fujitsu's strategy for emerging applications
- Recent projects conducted by JST-CREST teams
  - Graph500 benchmark (Prof. Fujisawa's team)
  - Inchworm (Prof. Nanri's team)
- Fujitsu's Deep Learning Unit
- Emerging technologies and future challenges

Graph500 Benchmark

FUJITSU

- Parallel breadth-first search of a huge graph
  - Calculate an array of parent vertices

- Hyper-sparse adjacency matrix

$$A = \begin{pmatrix} A_{1,1} & \cdots\cdots & A_{1,2^S} \\ \vdots & \ddots\ddots & \vdots \\ A_{2^S,1} & \cdots\cdots & A_{2^S,\,2^S} \end{pmatrix}$$

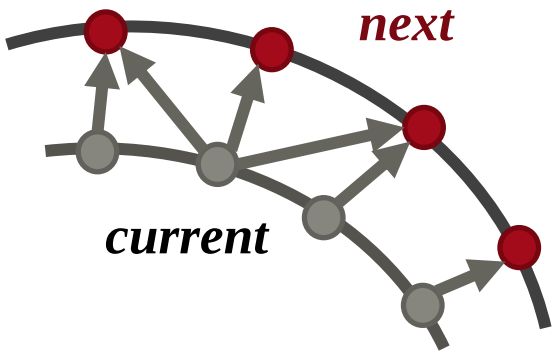
  - Size of adjacency matrix = $2^S \times 2^S$ (S = problem scale)
  - Number of edges = 16×2^S
  - As problem scale increases, adjacency matrix becomes sparse

- Two key points of optimization
  - Compression of the adjacency matrix for computation
  - Distribution of the adjacency matrix for communication

Direction-Optimizing Search [Beamer, SC12]

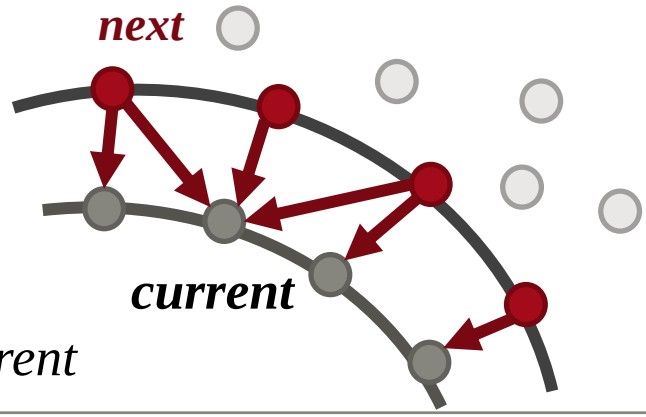
FUJITSU

- Target of Graph500 is a scale-free network
  - It includes high-degree vertices
  - If the currently visited vertices include high-degree vertices:
    - The number of edges to traverse in the next step will increase significantly
    - Reversing search direction may reduce the number of edges to traverse



- Top-down direction
  - Search next vertices directly
  - From the currently visited vertices

$$next = (A \times current) \,\&\, \neg\, visited$$



- Bottom-up direction
  - Search all unvisited vertices
  - For the adjacent vertices of the currently visited vertices

$$next = (A \,\&\, [\, \neg\, visited \mid \neg\, visited \mid \ldots\,])^{\mathrm{T}} \times current$$

2D Partitioning [Yoo, SC05]

FUJITSU

- Partitioning of the adjacency matrix in two-dimensions
  - Distribution of edges according to both source and destination vertices
- Each searching step requires:
  - All-gather communication for search calculation in one dimension
  - All-to-all communication for update in the other dimension
- Good scalability
  - Each node uses only a part of the vectors, such as *current* or *next*
  - If the problem scale is large, the vectors will be too large for memory
- High performance
  - Collective communication groups are small
  - Update communication can be overlapped by computation when the search direction does not change in the next step

2D Partitioning on the K computer

FUJITSU

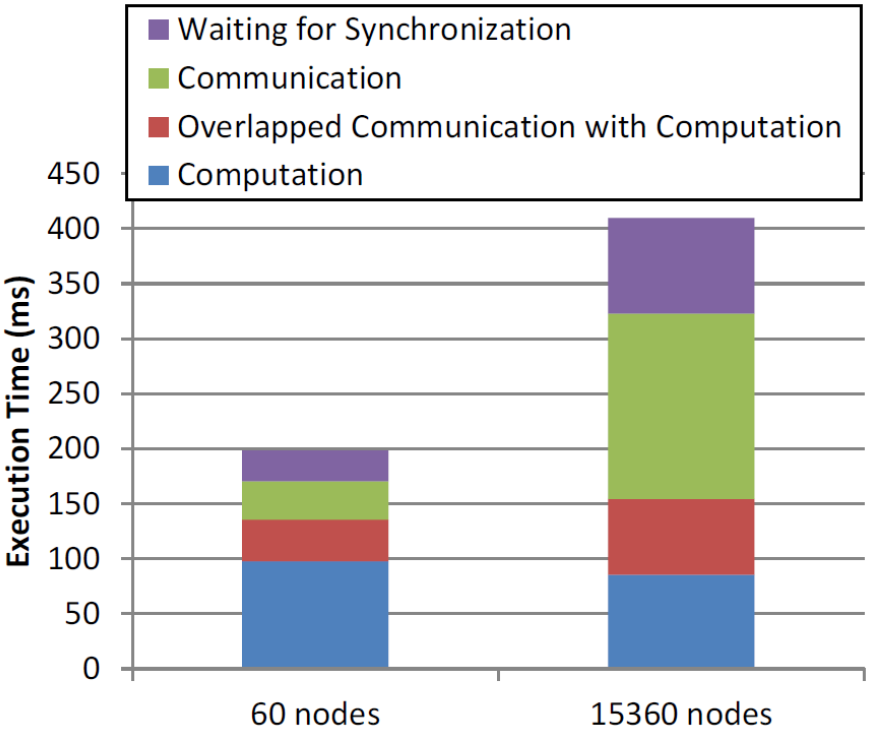
- The adjacency matrix was divided into 288×288
  - K computer: 82,944 nodes = $24 \times 18 \times 16 \times 2 \times 3 \times 2$
  - $18 \times 16 = 288$ and $24 \times 2 \times 3 \times 2 = 288$
- 288 row and 288 column communicators
  - Each node participated in one row and one column communicator
  - Each communicator consisted of all nodes in a 6D rectangular submesh, so that Fujitsu MPI used topology-aware algorithm
    - All-gather: three-phase quad rings algorithm
    - All-to-all: uniformly overlaid symmetrical pattern algorithm
- Communication was overlapped with computation if the number of vertices to be updated was small
  - Otherwise, an MPI collective operation was called
  - Different implementation was required for each dimension and direction

Repository: github.com/suzumura/graph500

Graph500 Performance on the K computer

FUJITSU

- History and breakdown [Ueno, IEEE Big Data 2016]
  - 17,977 GTEPS using 2D partitioning (June 2014)
  - 19,585 GTEPS using hybrid top-down/bottom-up (Nov. 2014)
  - 38,621 GTEPS by improved sparse matrix computation (July 2015)
    - Compute-efficient compression, vertex reordering for access locality and improved inter-thread load balance
  - The ratio of computation time was relatively small
  - Communication time became a major bottleneck in large scale
- High-throughput interconnect is important for graph BFS



FUJITSU

Index

- Fujitsu's strategy for emerging applications
- Recent projects conducted by JST-CREST teams
  - Graph500 benchmark (Prof. Fujisawa's team)
  - Inchworm (Prof. Nanri's team)
- Fujitsu's Deep Learning Unit
- Emerging technologies and future challenges

Trinity

FUJITSU

- A major Extraction de novo transcript assembly software
  - of RNA sequences without reference DNA
  - Consist of Inchworm, Chrysalis and Butterfly

- Usually run on a large memory machine
  - Inchworm is the most memory-consuming process

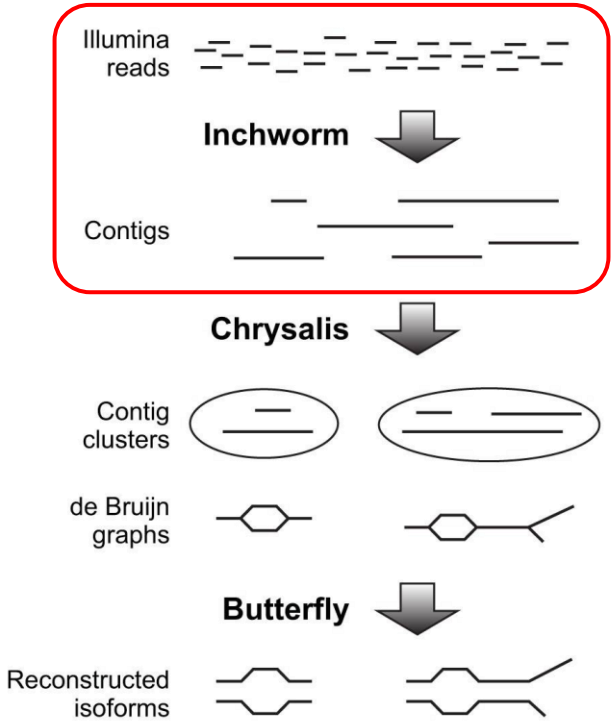
- Input: short reads of Next-Generation Sequencing (NGS)
  - NGS is a high-throughput DNA/RNA sequencing platform
  - Large number of short reads that are around 100 base pairs
  - Short reads overlap each other
  - Reads contains errors or unusable data at a certain ratio

Inchworm

FUJITSU

- Overview of Trinity's processing
  - Extracts 'contigs' (overlapped series of *k*-mers) from short reads
  - Constructs partial de Bruijn graphs from contigs
  - Finds Eulerian path for each disconnected subgraph
- Inchworm
  - Pre-process for de Bruijn graph construction
    - Counts the number of occurrences of each *k*-mer
    - Constructs 'contigs' with high-frequency *k*-mers
- Parallelization of Inchworm
  - Distributed data structure APIs
  - ACP communication library
    - Supports: Tofu, InfiniBand and UDP

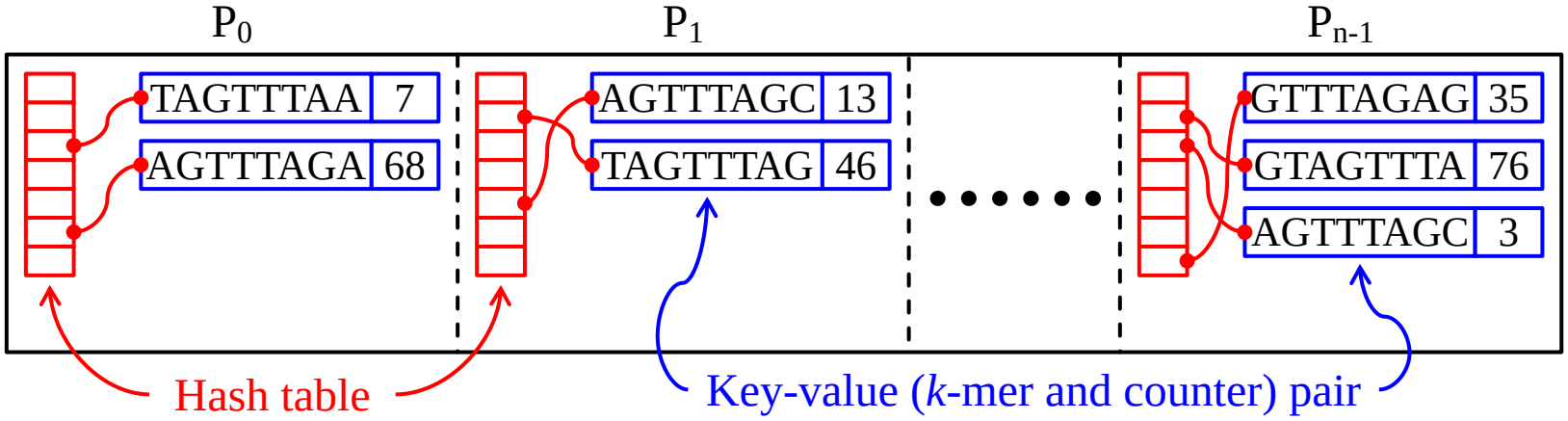
  Repository: github.com/project-ace/ACP



Parallel Construction of *k*-mer Dictionary

FUJITSU

- Using map data structure
  1. Each process creates a map that can be accessed from any process
  2. Each process reads input in parallel and inserts *k*-mers into the maps



- The process of inserting *k*-mer involves three steps
  1. Hashing of the *k*-mer to determine in which process's map to store it
  2. Inserting the *k*-mer and value 1 into the map
  3. Atomically incrementing the existing value if step 2 fails
  - There is additional overhead due to the separate steps 2 and 3

Multiset data structure

FUJITSU

- We added ‘multiset’ data structure to ACP
  - Multiset is an extended data structure of the ‘set’
  - Set: insertion fails if the key already exists
  - Multiset: insertion increments the counter of the key and does not fail
- Multiset simplifies *k*-mer insertion process
  1. The *k*-mer is hashed to determine in which process’s multiset to store it
  2. The *k*-mer is inserted into the multiset
- We also extended multiset API for the next splicing process
  - Retrieve: obtains the counter value of the specified key
  - Remove-all: deletes the specified key regardless of its counter value

Parallel *k*-mer Splicing

FUJITSU

- Each process selects a *k*-mer as a starting point
- Processing of *k*-mer splicing for each direction
  1. Retrieves counter values of four, ATGC, adjacent *k*-mer candidates
  2. Ends the splicing process if no adjacent *k*-mer is found in step 1
  3. Selects the *k*-mer with the highest counter value
  4. Splices the selected *k*-mer to the contig
  5. Deletes the selected *k*-mer from the *k*-mer dictionary
  - This process is the same as the sequential version, except that the *k*-mer dictionary is distributed
- Future research
  - Retrieve four adjacent *k*-mer candidates through a single API call
    - Exclude base pairs at both ends of *k*-mer in process selection
  - Increase probability of storing adjacent *k*-mers in the same process
    - Predict input and adjust the hash function of process selection

FUJITSU

Index

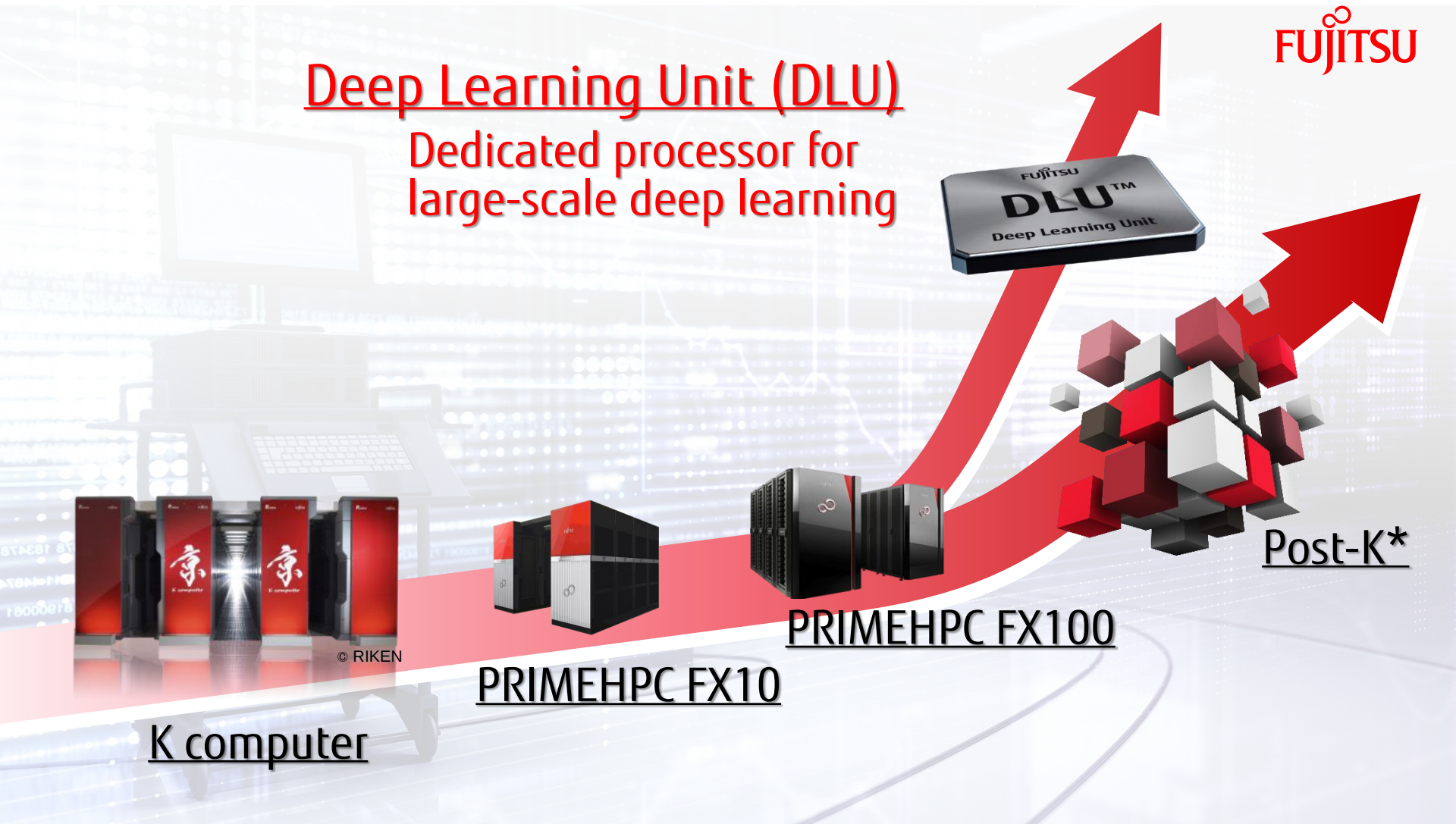
- Fujitsu's strategy for emerging applications
- Recent projects conducted by JST-CREST teams
  - Graph500 benchmark (Prof. Fujisawa's team)
  - Inchworm (Prof. Nanri's team)
- Fujitsu's Deep Learning Unit
- Emerging technologies and future challenges

Deep Learning Unit

FUJITSU

Deep Learning Unit (DLU)

Dedicated processor for large-scale deep learning

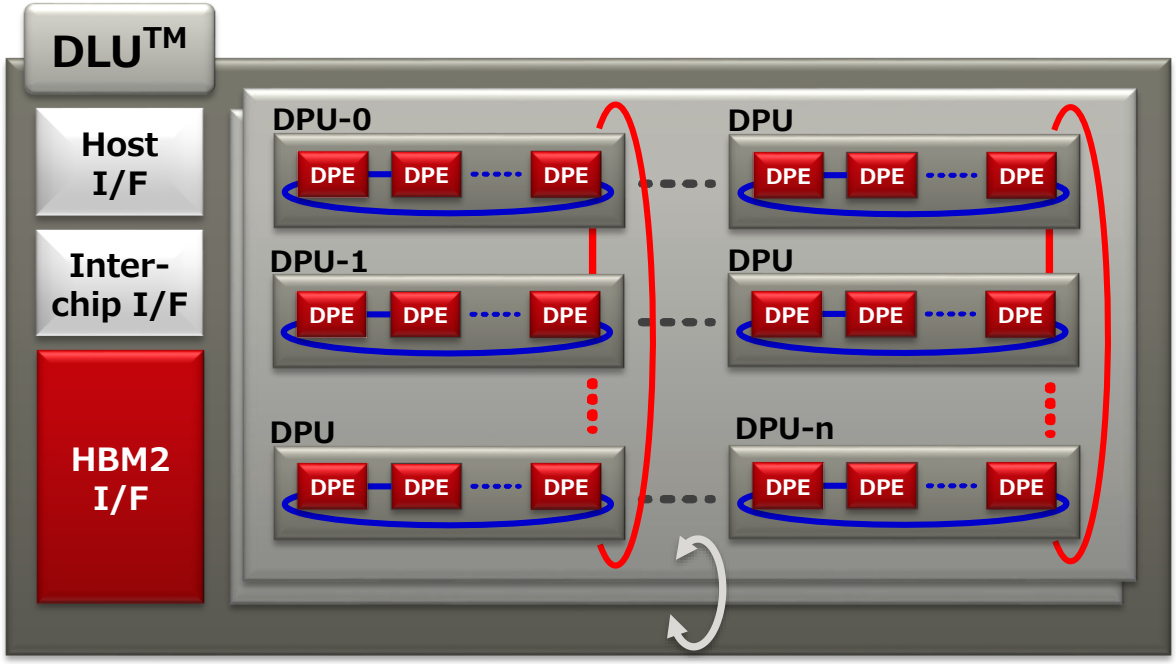


* RIKEN and Fujitsu are currently developing the post-K computer

DLU™ Architecture

FUJITSU

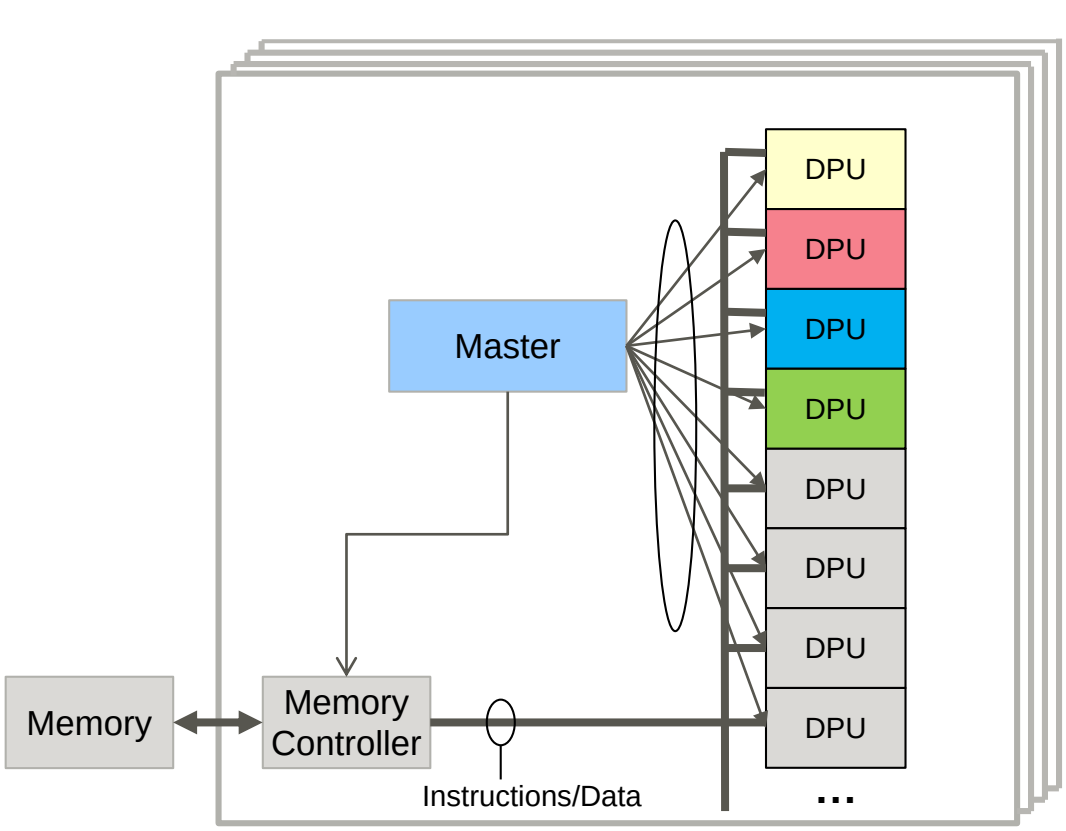
- New ISA developed for deep learning
- Highly power-efficient with simple microarchitecture
- High bandwidth memory of HBM2
- Large-scale network of inter-DLU direct connection

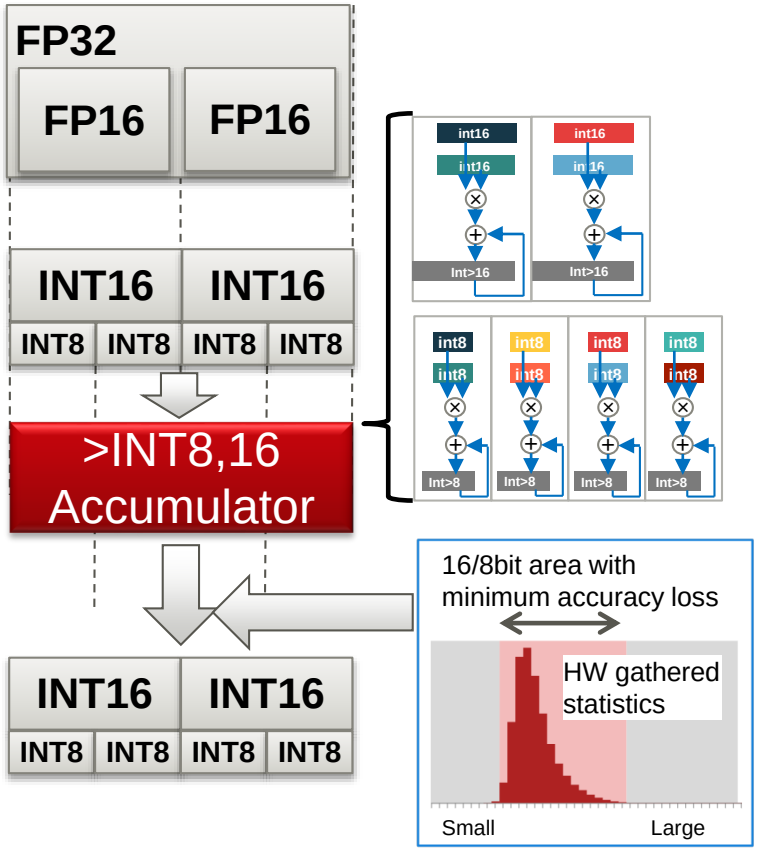


DPU: Deep learning Processing Unit
DPE: Deep learning Processing Element

DLU™ Architecture (Cont.)

- Heterogeneous core
  - Master core handles Memory access and controls DPU
- FP32, FP16 and Deep Learning Integer
  - INT16 and INT8 with automatic scaling



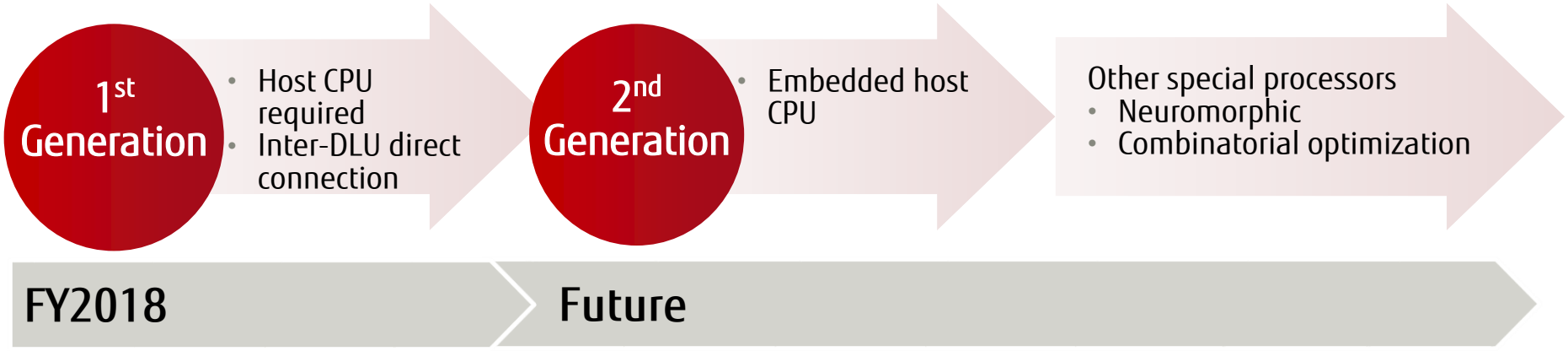


DLU Roadmap

FUJITSU

- The 2020s will be the era of domain-specific machines
- DLU™ is Fujitsu's first processor for this era

- Multiple generations of DLUs are included in the roadmap
  - Continuous processor development strategy, similar to Fujitsu's strategy in HPC, UNIX and Mainframe areas



* Subject to change without notice

FUJITSU

Index

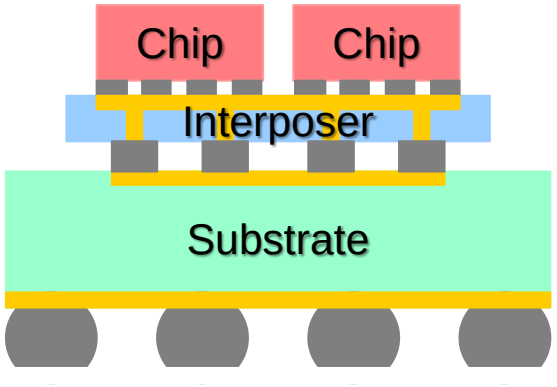
- Fujitsu's strategy for emerging applications
- Recent projects conducted by JST-CREST teams
  - Graph500 benchmark (Prof. Fujisawa's team)
  - Inchworm (Prof. Nanri's team)
- Fujitsu's Deep Learning Unit
- Emerging technologies and future challenges

Emerging Packaging Technology

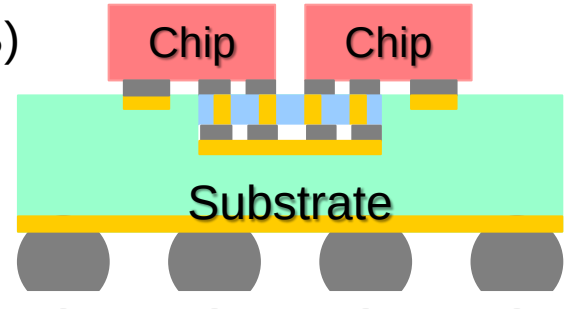
FUJITSU

- Packaging technologies impact node architecture
  - Off-package interconnect also needs to adapt to new node architectures

- Recent multi-chip package technology
  - High density chip-to-chip interconnection
  - Si-IP with TSVs
    - TSMC Chip-on-Wafer-on-Substrate (CoWoS)
    - NVIDIA Tesla P100 and V100



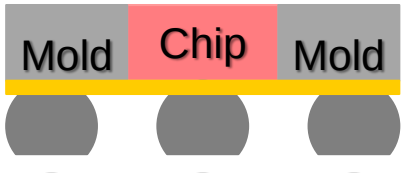
  - Small bridge for chip-to-chip interconnection
    - Intel Embedded Multi-die Interconnect Bridge (EMIB)
    - Altera Stratix 10

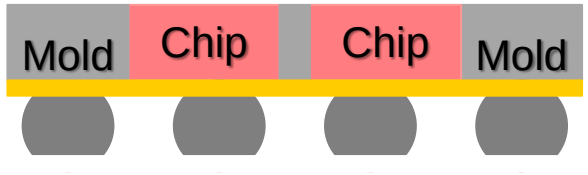


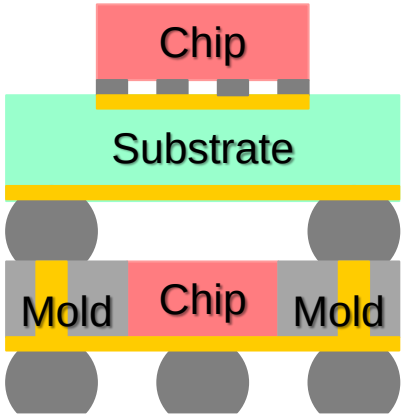
Thin Package without Substrate

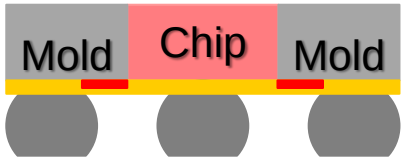
FUJITSU

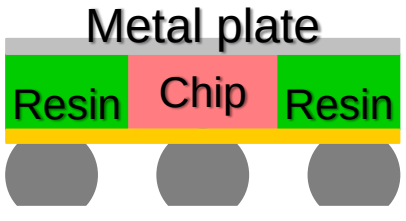
- Fan-out wafer-level packaging (FOWLP)
  - Redistribution layer is supported by mold
  - Better signal integrity and low power loss
- Variations of FOWLP
  - Multi-chip FOWLP
    - High density chip-to-chip interconnection
  - FOWLP package-on-package
    - TSMC InFO-PoP
    - Apple A10
  - Inductor integrated FOWLP
    - Further reduction in power loss
- Fan-out panel-level packaging (FOPLP)
  - Low-cost solution







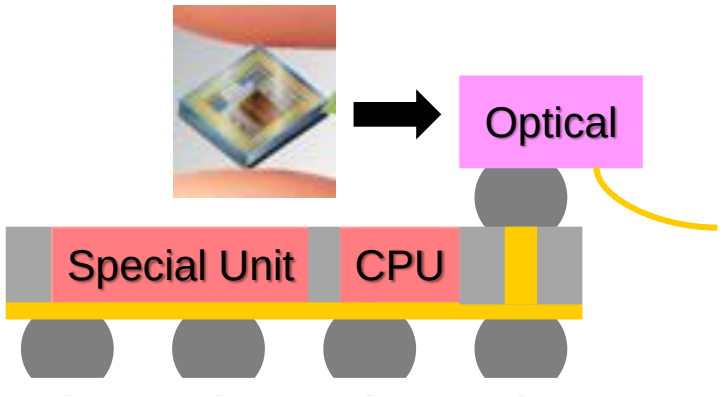




Future System and Interconnect

FUJITSU

- Future node architecture
  - Substrateless packaging technology
  - Heterogeneous integration
  - Optical transceiver module on package



- Trade-offs related to package size
  - Package size of FOWLP is generally smaller than that of Si-IP
  - Smaller package size improves yield rate
  - Smaller node requires high scalability and high parallel efficiency

- Interconnect design for future domain-specific machines
  - Different package technology, scalability, media, speed and cost
  - Some systems will incorporate dedicated interconnect design
  - For other systems, interconnect should be designed as a configurable IP

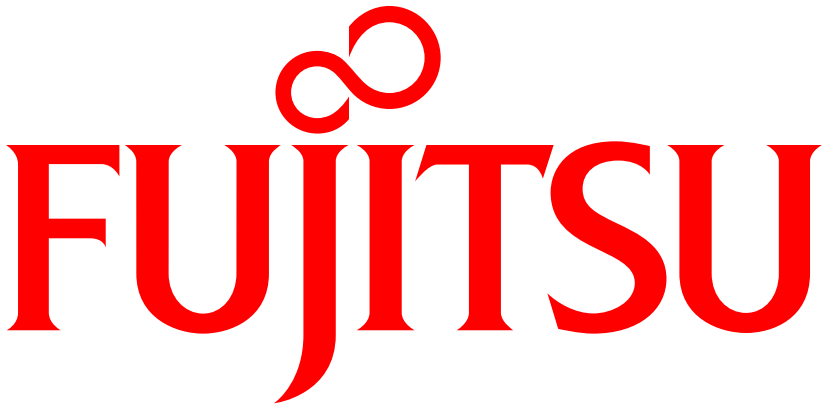
FUJITSU

Summary

Summary

FUJITSU

- Three domains of emerging applications
  - Big data analytics, cognitive applications and optimization problems
  - Fujitsu will develop highly scalable domain-specific machines
- Recent projects conducted by JST-CREST teams
  - Optimization of Graph500 benchmark by Prof. Fujisawa's team
  - Parallelization of Inchworm by Prof. Nanri's team
- Fujitsu's Deep Learning Unit
  - Large-scale network of inter-DLU direct connection
- Future challenges
  - Higher scalability for the trade-off between yield rate and package size
  - Not only dedicated design but also configurable IP is required



shaping tomorrow with you