東北大学

Two-Year Experiences with SX-ACE and Design Space Exploration of the Next Generation Vector System

Hiroaki Kobayashi
Deputy Directer for the HPC Strategy of Cybersicence Center
Professor of Graduate School of Information Sciences
Chair of Computer and Mathematics Sciences Department

Tohoku University
koba@tohoku.ac.jp

26th WSSP
October 10–11, 2017

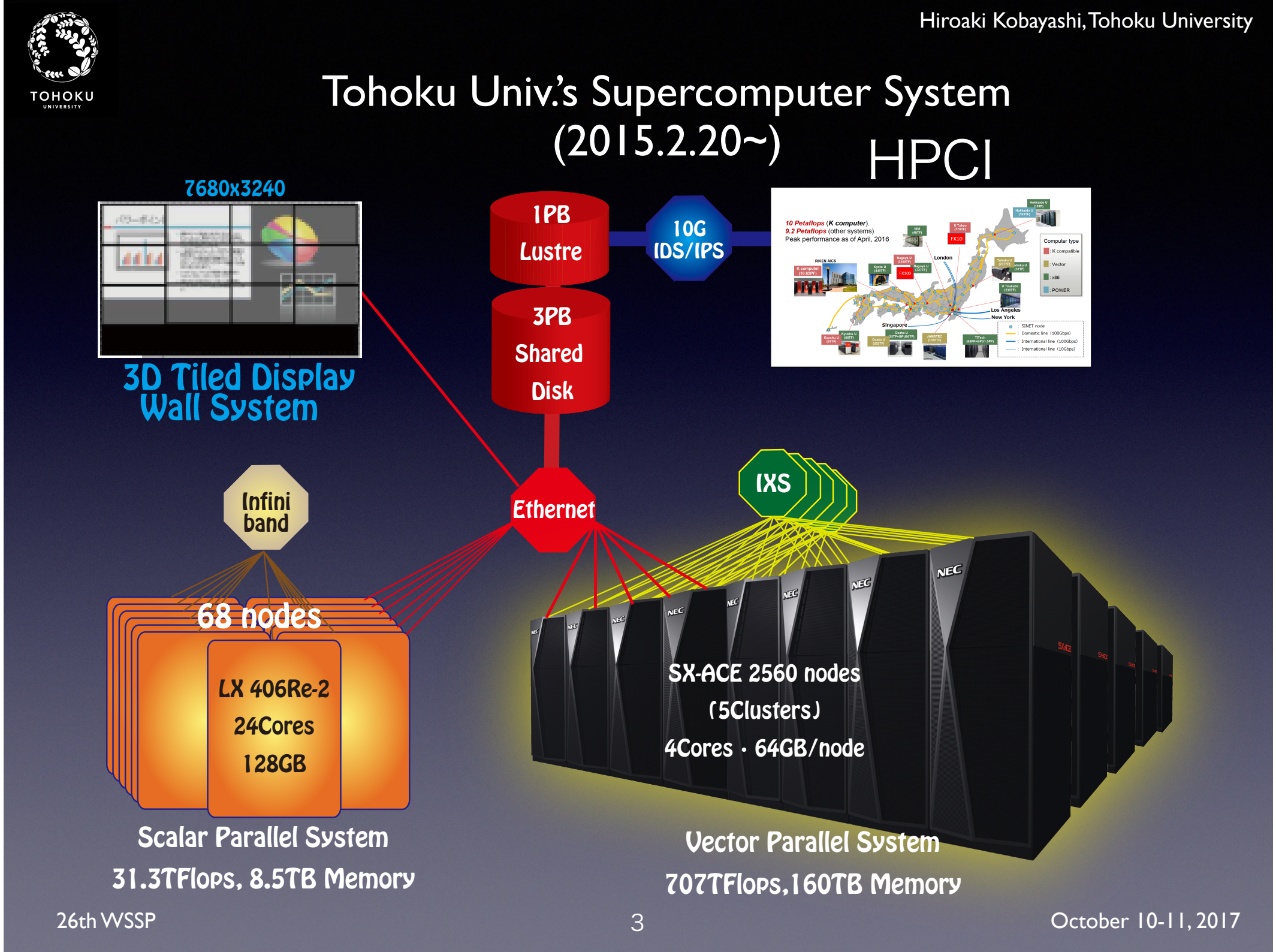
TOHOKU UNIVERSITY

Today's Agenda

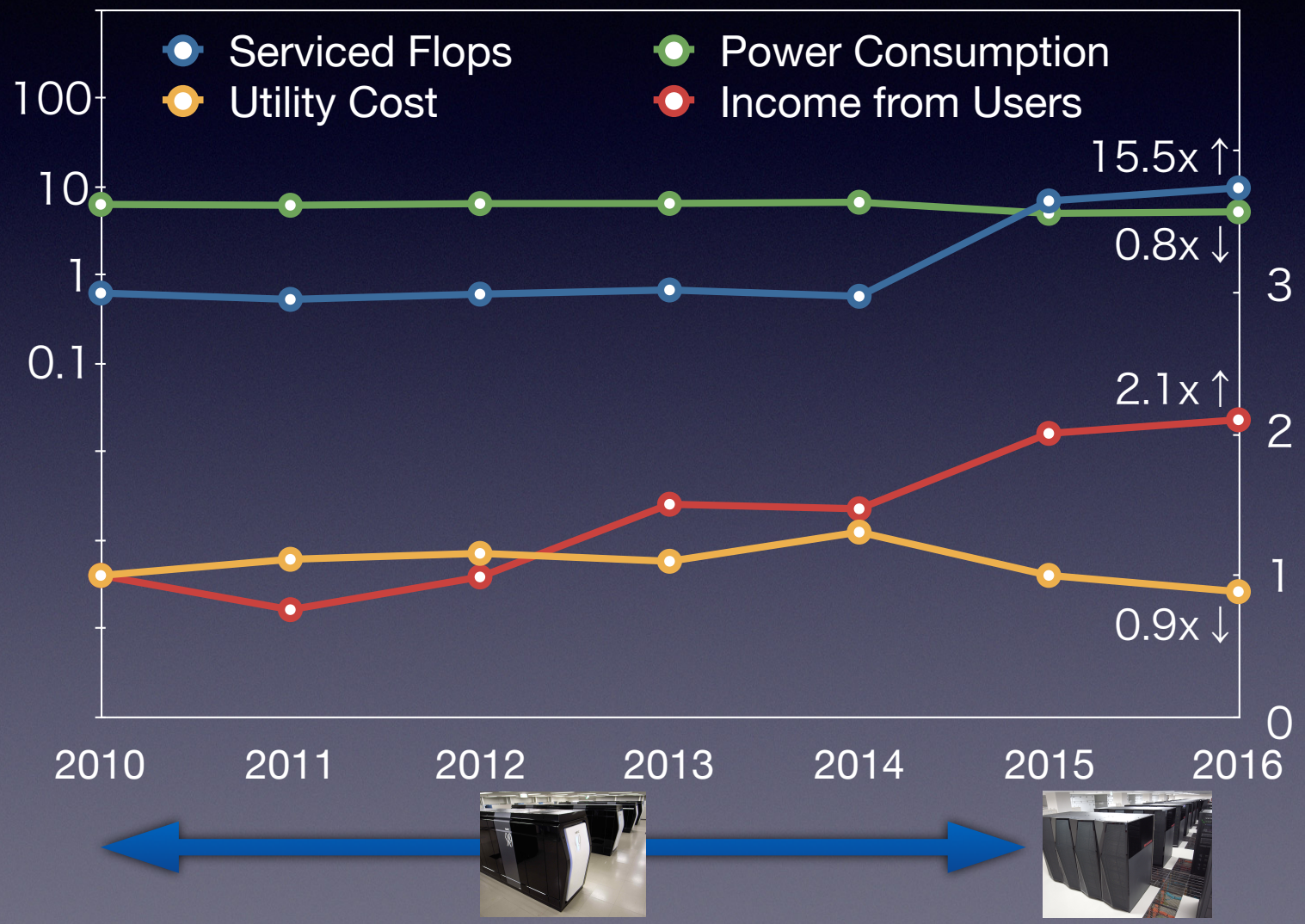
- ★ A two-year experience with SX-ACE since 2015
  - ✓ Lessons Learned from SX-ACE Operations snd Applications Developments
- ★ Some thoughts for design and development of the next generation vector system
  - ✓ Design Constraints
  - ✓ Domain-Specific Architecture Design in the Era of End-of-Moore's Law

TOHOKU UNIVERSITY

Tohoku Univ.'s Supercomputer System (2015.2.20~)

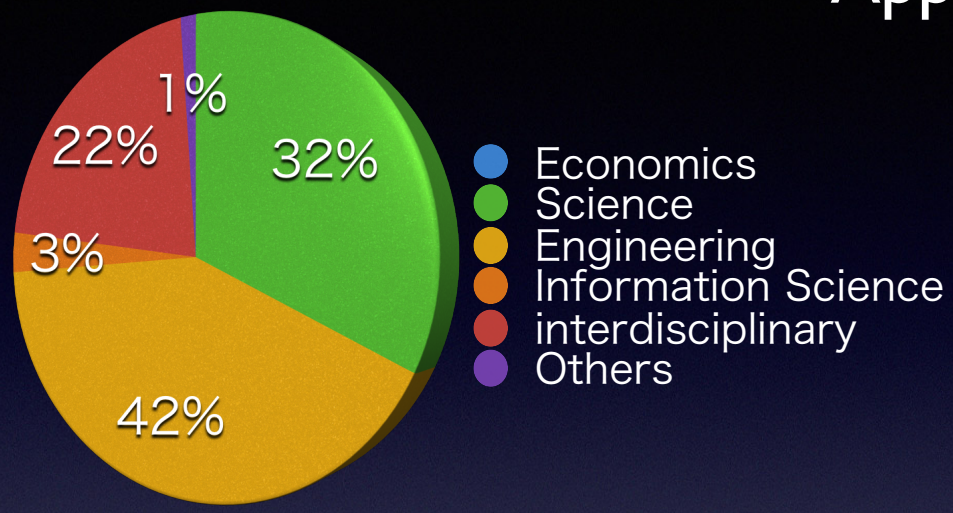


Operation Statistics of SX-ACE (Normalized by SX-9 Data)



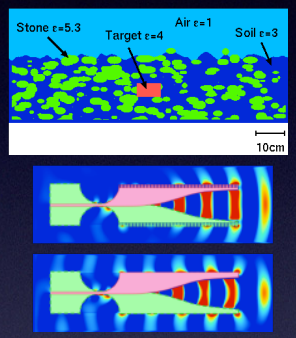
TOHOKU UNIVERSITY

Applications

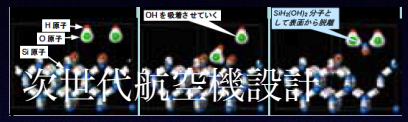


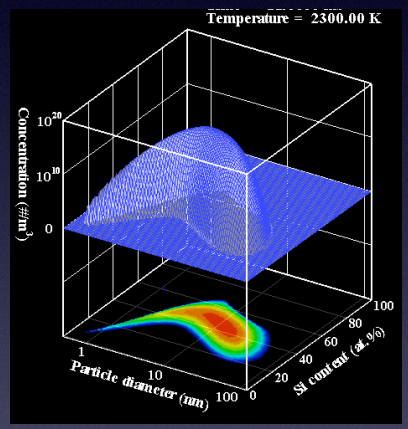
Heat Shock Analysis

Electromagnet Analysis

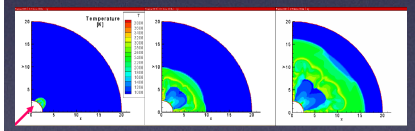


Material

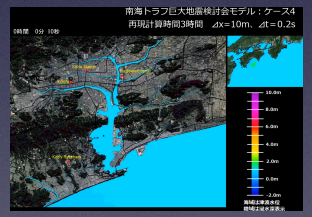


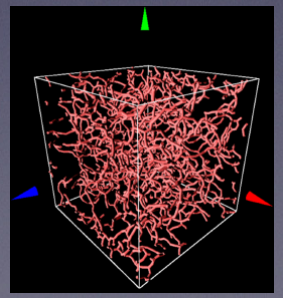


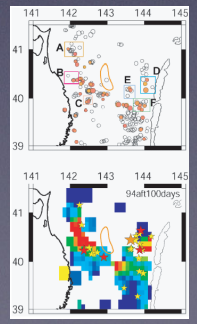
Combustion



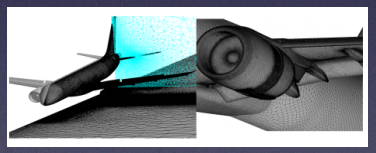
Earthquake&Tsunami



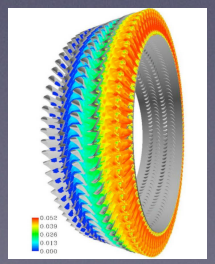




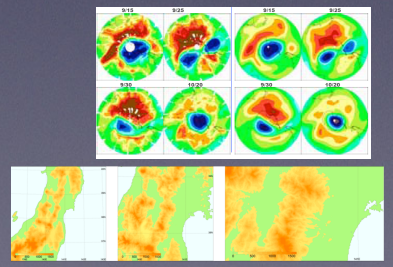
Airplane Design and Analysis



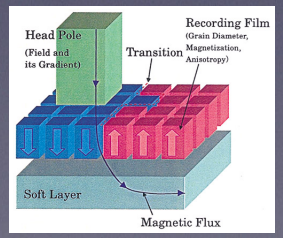
Turbine

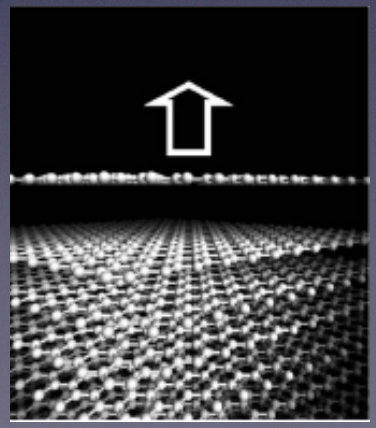


Weather/Atmosphere

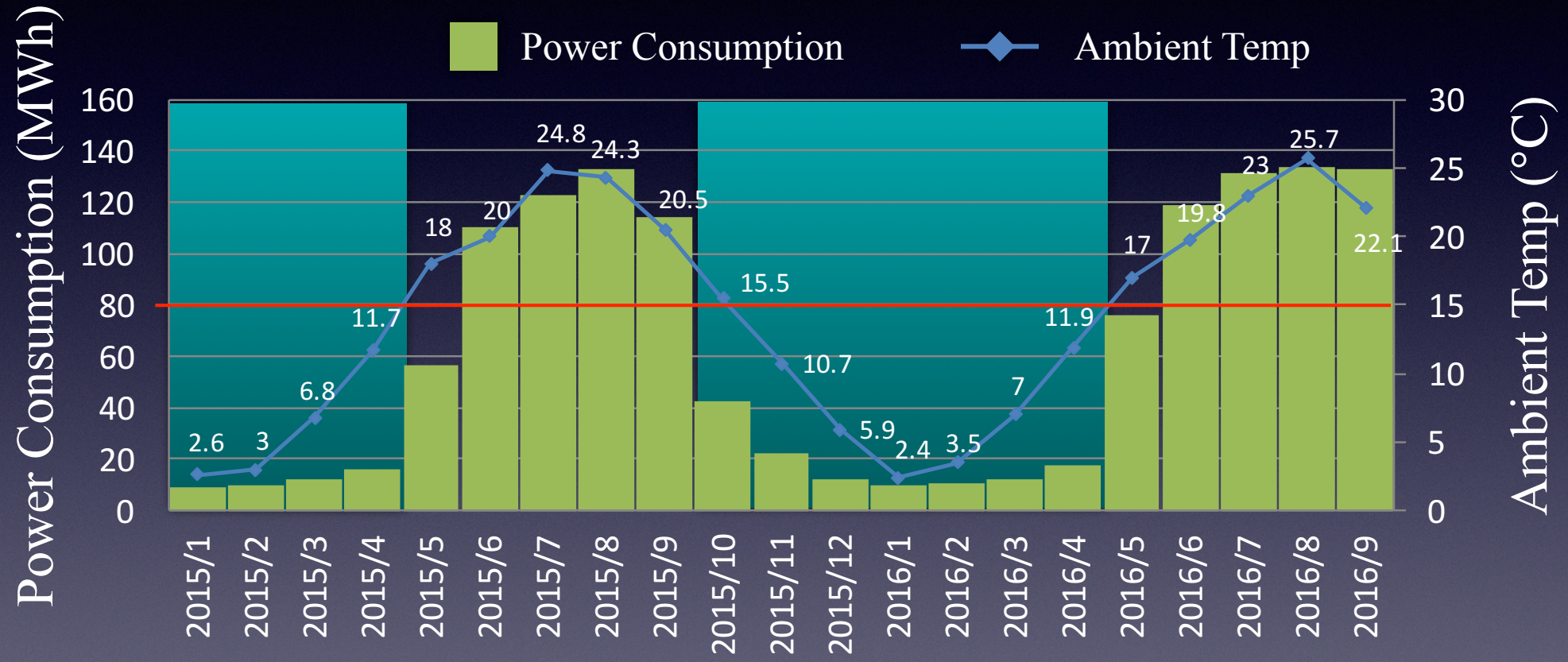


Perpendicular Recording Device

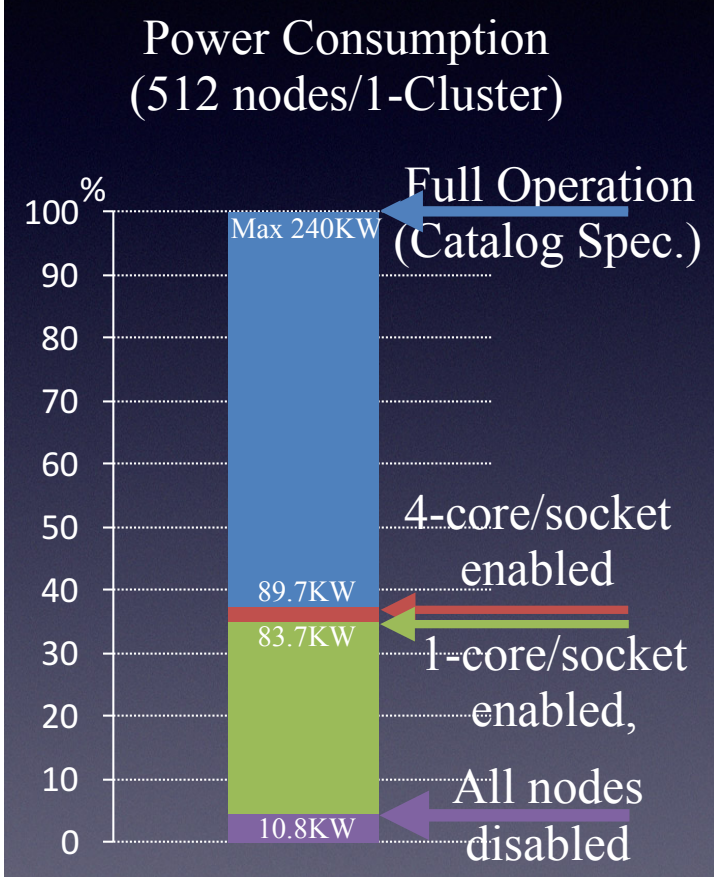


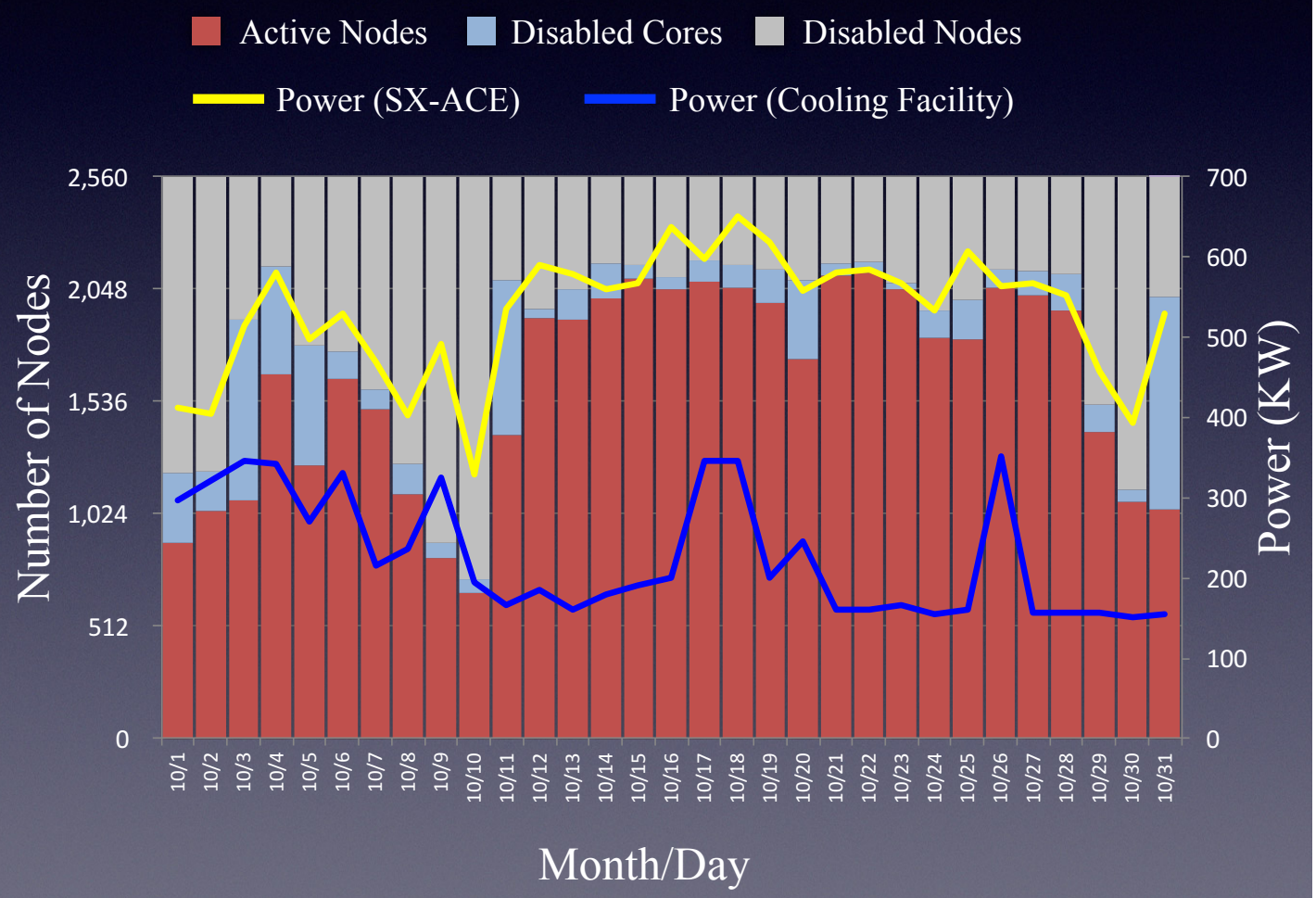


Power Consumption of the Cooling System
Effect of Fresh-Air Cooling



Node-Core Activity: Effect of Automatic Core-Node Activation/Deactivation Control

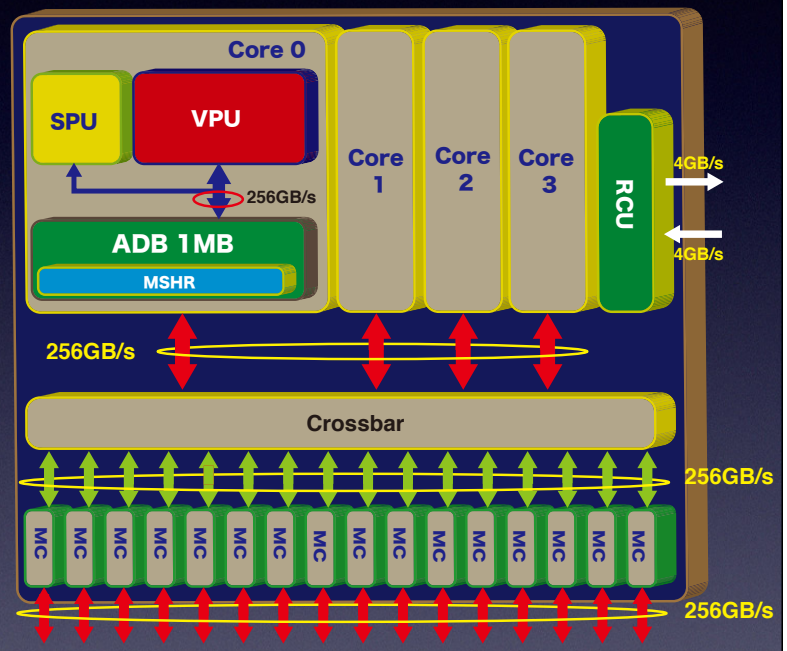




Features of the SX-ACE Vector Processor

- 4 Core Configuration, each with High-Performance Vector-Processing Unit and Scalar Processing Unit
  - 272Gflop/s of VPU + 4Gflop/s of SPU per socket
    - 68Gflop/s + 1Gflop/s per core
  - 1MB private ADB per core (4MB per socket)
    - Software-controlled on-chip memory for vector load/store
    - 4x compared with SX-9
    - 4-way set-associative
    - MSHR with 512 entries (address+data)
    - 256GB/s to/from Vec. Reg.
      - 4B/F for Multiply-Add operations
  - 256 GB/s memory bandwidth, Shared with 4 cores
    - 1B/F in 4-core Multiply-Add operations
      ~ 4B/F in 1-core Multiply-Add operations
    - 128 memory banks per socket
    - 128B-Block access to the memory (16B of SX-9)

SX-ACE Processor Architecture



Source: NEC

- Other improvement and new mechanisms to enhance vector processing capability, especially for efficient handling of short vectors operations and indirect memory accesses
  - Dynamically and Statically reordering execution of vector gather/scatter operations
  - Advanced data forwarding in vector pipes chaining
  - Shorter memory latency than SX-9

TOHOKU UNIVERSITY

Dynamically and Statically Reordering Execution of Vector Gather/Scatter Operations

- Vectorization
  - NODEP: Load & Store operations of each element of a list-vector are executed in parallel.
    - Vectorization Possible
- Giving Priority to Vector Gather(Loads) over Scatter(Vector) Stores
  - GTHREORDER
    - Compiler statically reschedules vector gather/scatter operations to give the priority to Loads over Stores
  - NOCONFLICT
    - Assign a flag that there are no data hazard between loads and stores to access different arrays, and dynamic reordering becomes possible by hardware to give the priority to VLD and VGT (gather) over VST and VSC (scatter)
  - VOVERTAKE
    - Assign a flag that there is no data hazard between loads and stores within the same array, and dynamic reordering become possible by hardware to give the priority to VLD and VGT over VST and VSC

Rescheduling vector gather/scatter operations by Compiler

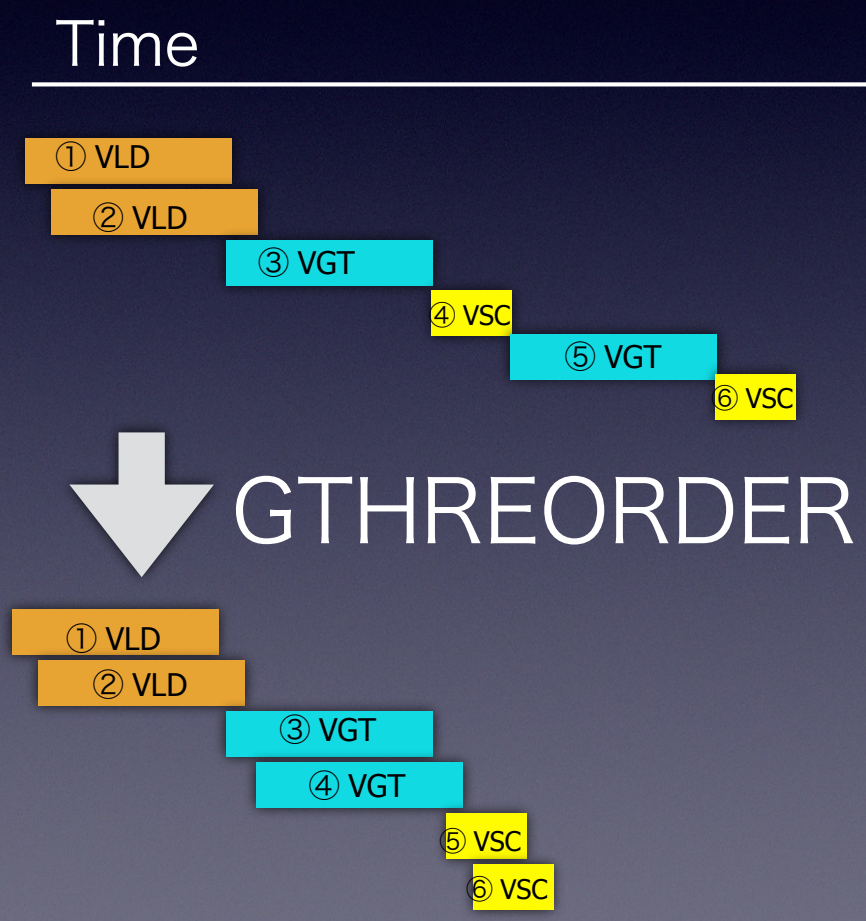
```
!CDIR NODEP
!CDIR GTHREORDER
DO I = 1, N
 A(IND(I)) = A(IND(I)) + X(I)
 A(IND(I)+1) = A(IND(I)+1) + X(I)
ENDDO
```

【NODEP Only】

① VLD	IND(I)
② VLD	X(I)
③ VGT	A(IND(I))
④ VSC	A(IND(I))
⑤ VGT	A(IND(I)+1)
⑥ VSC	A(IND(I)+1)

【NODEP and GTHREORDER】

① VLD	IND(I)
② VLD	X(I)
③ VGT	A(IND(I))
④ VGT	A(IND(I)+1)
⑤ VSC	A(IND(I))
⑥ VSC	A(IND(I)+1)



Out of Order execution for vector load&gather/ store&scatter operations across iterations by hardware

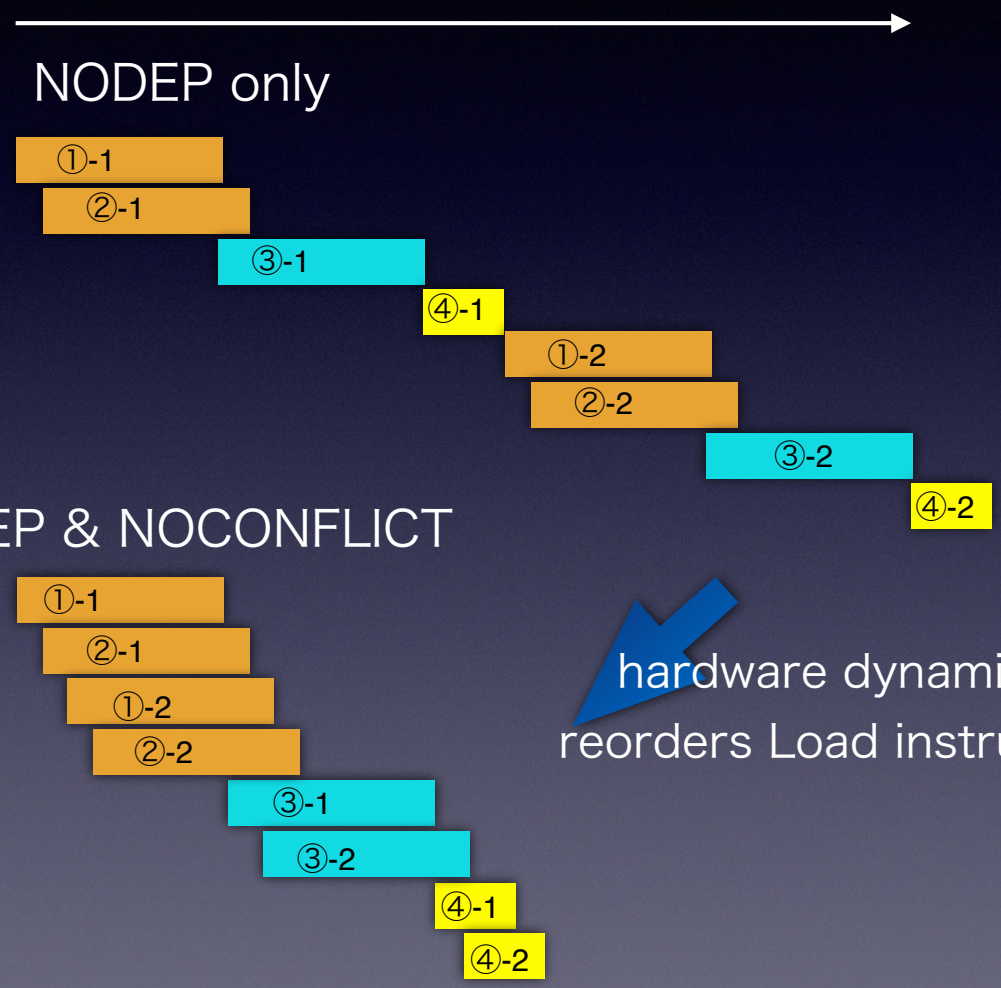
```
!CDIR NODEP
!CDIR NOCONFLICT
DO I = 1, 512
 A(IND(I)) = Y(IND(I)) + X(I)
ENDDO
```

I=1~256

①-1	VLD	IND(I)
② -1	VLD	X(I)
③-1	VGT	Y(IND(I))
④-1	VSC	A(IND(I))

I=257~512

①-2	VLD	IND(I)
② -2	VLD	X(I)
③-2	VGT	Y(IND(I))
④-2	VSC	A(IND(I))



hardware dynamically reorders Load instructions

Out of Order execution for vector load&gather/ store&scatter operations across iterations by hardware

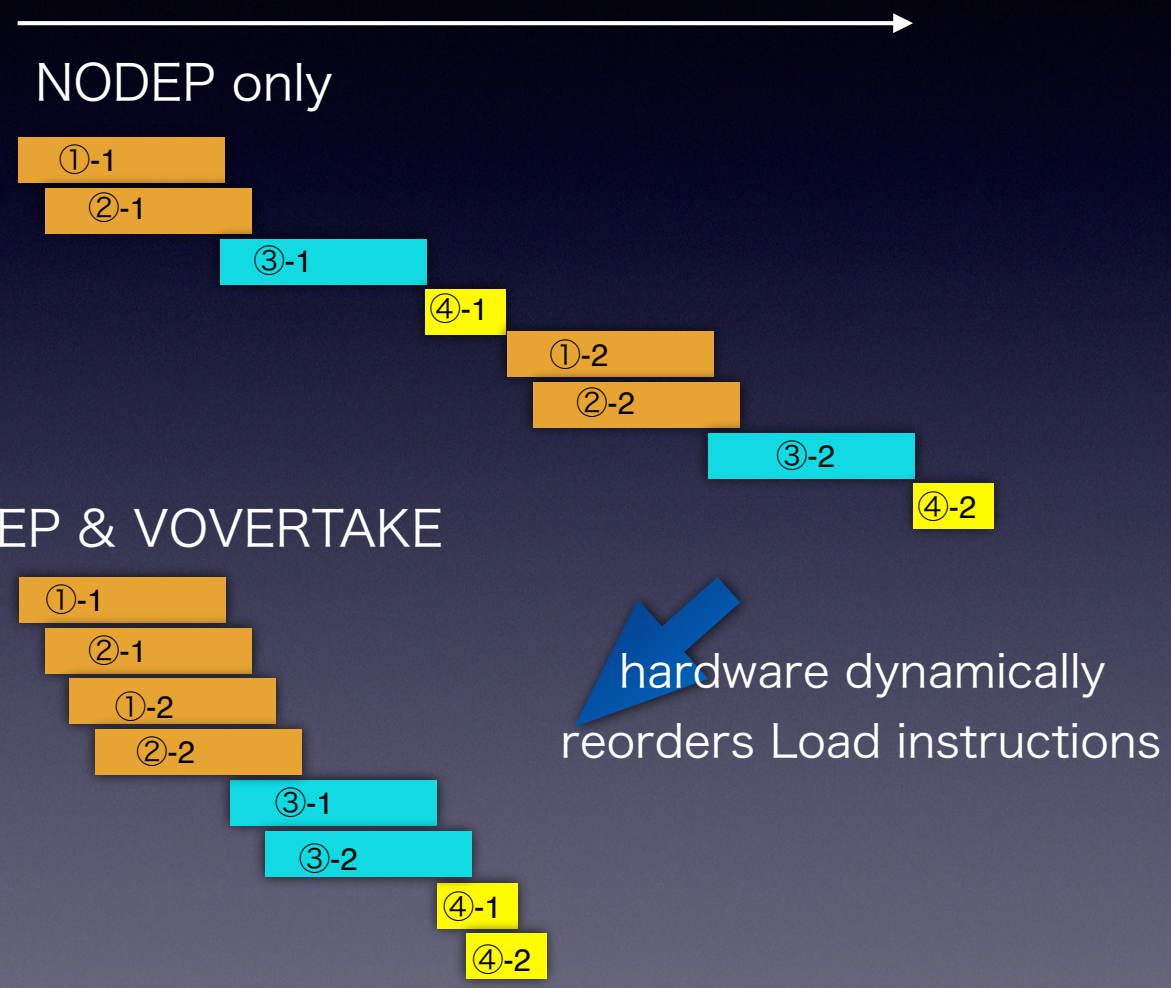
```
!CDIR NODEP
!CDIR VOVERTAKE(A), VOB
DO I = 1, 512
 A(IND(I)) = A(IND(I)) + X(I)
ENDDO
```

I=1~256

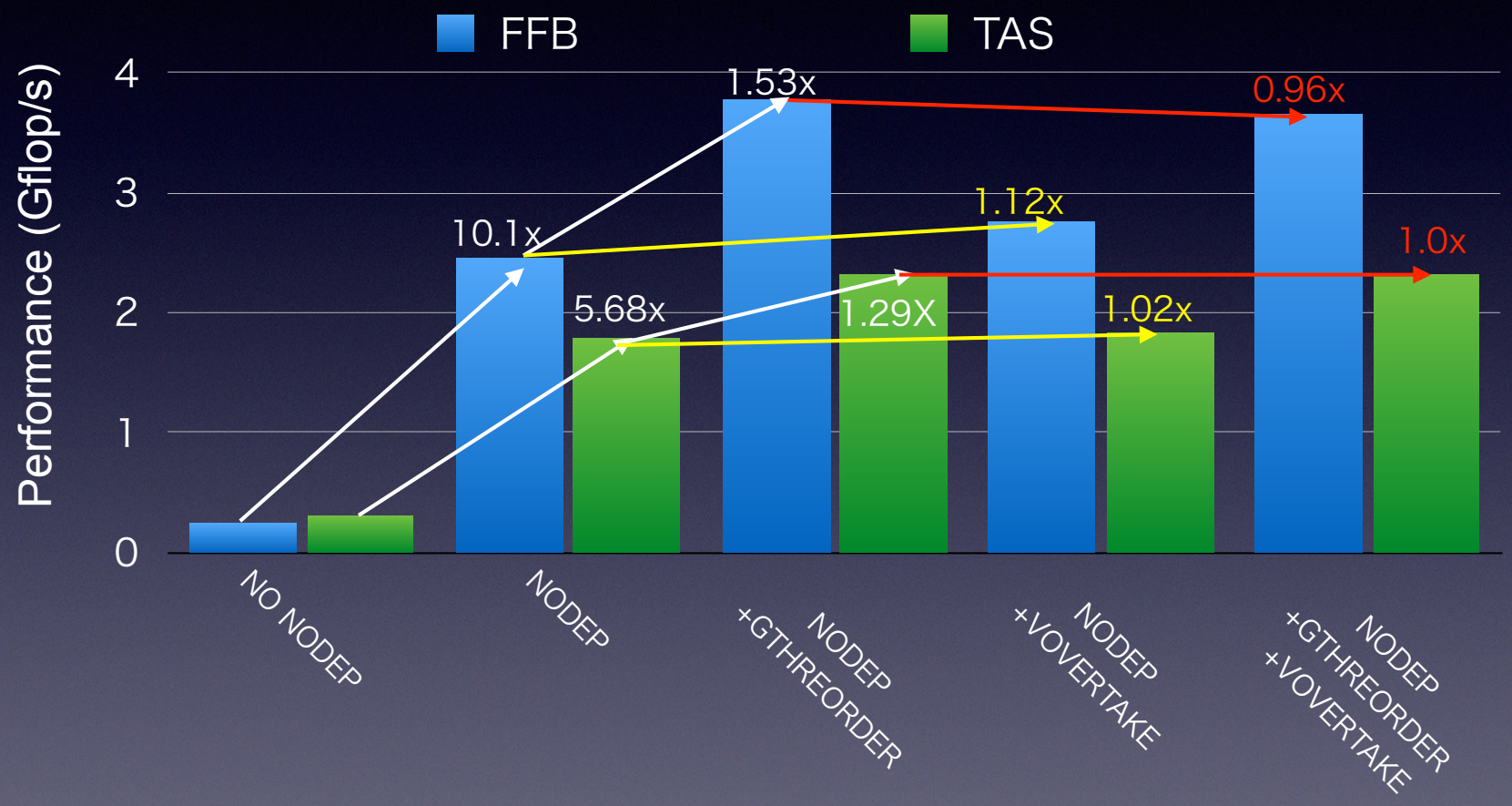
①-1	VLD	IND(I)
② -1	VLD	X(I)
③-1	VGT	A(IND(I))
④-1	VSC	A(IND(I))

I=257~512

①-2	VLD	IND(I)
② -2	VLD	X(I)
③-2	VGT	A(IND(I))
④-2	VSC	A(IND(I))



Effects of Giving the Priority to LOADs over STOREs

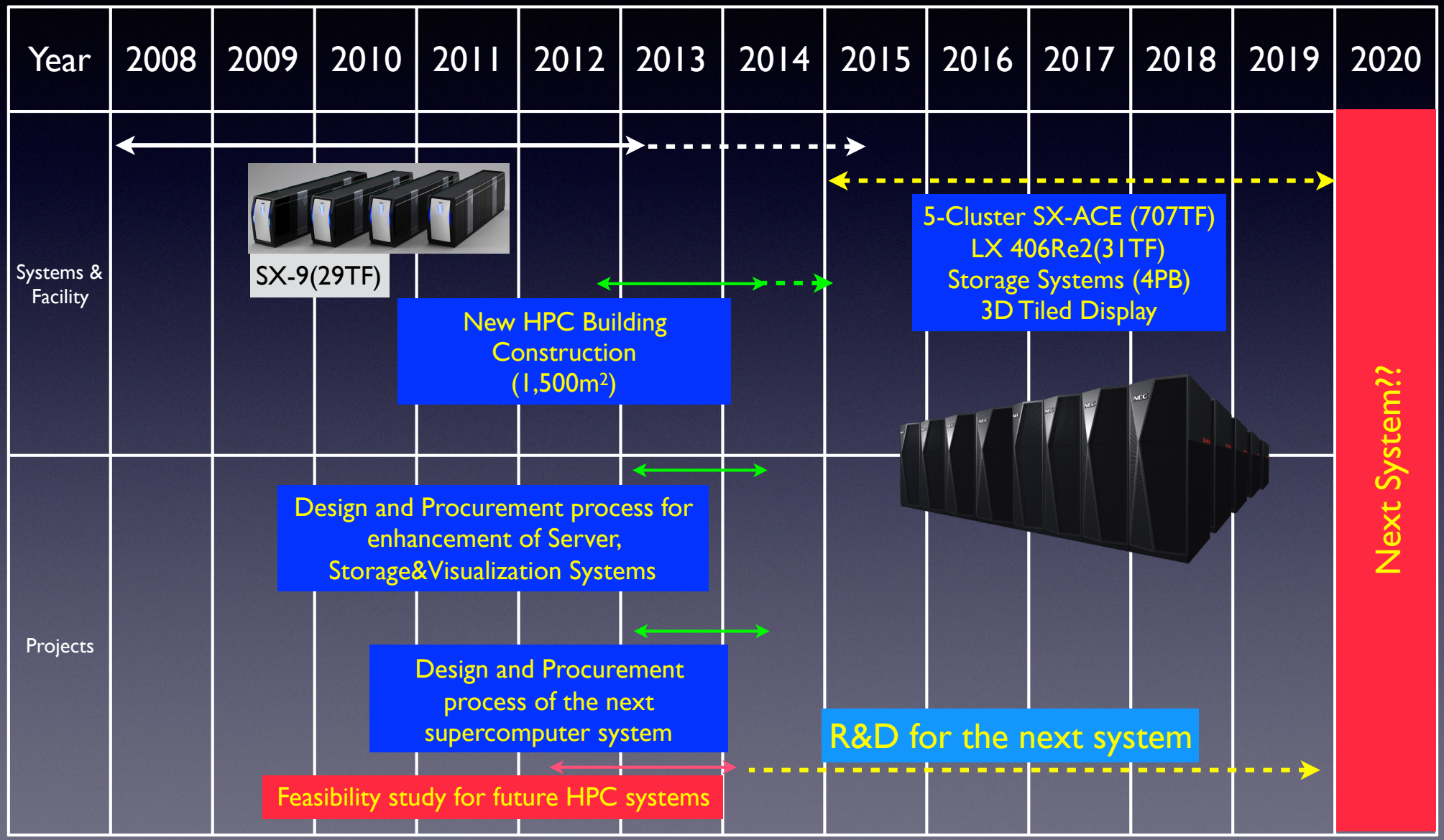


東北大学

Future Vector Systems R&D*

*This work is partially conducted with NEC, but the contents do not reflect any future products of NEC

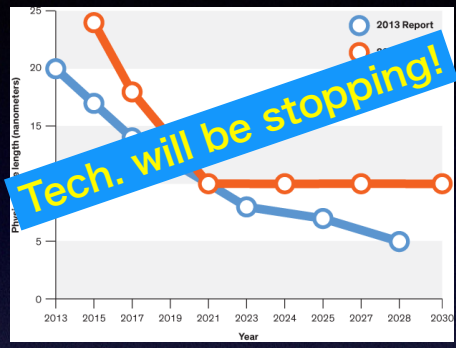
Timeline of the Cyberscience Center HPC System Development and R&D For the Future

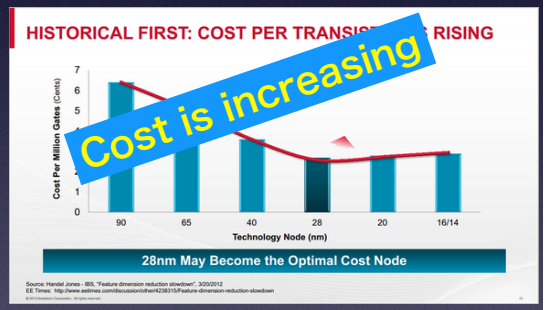


TOHOKU UNIVERSITY

Scaling may be End, but Silicon is not End! And Use it Smart and Effective!

- ✓ We are facing the end of Moore's low due to the physical limitations, and the transistor cost is hard to reduce, however
- ✓ Silicon is still fundamental constructing material for computing platforms such as plastic, steel and concrete for automobiles, buildings and home appliances.



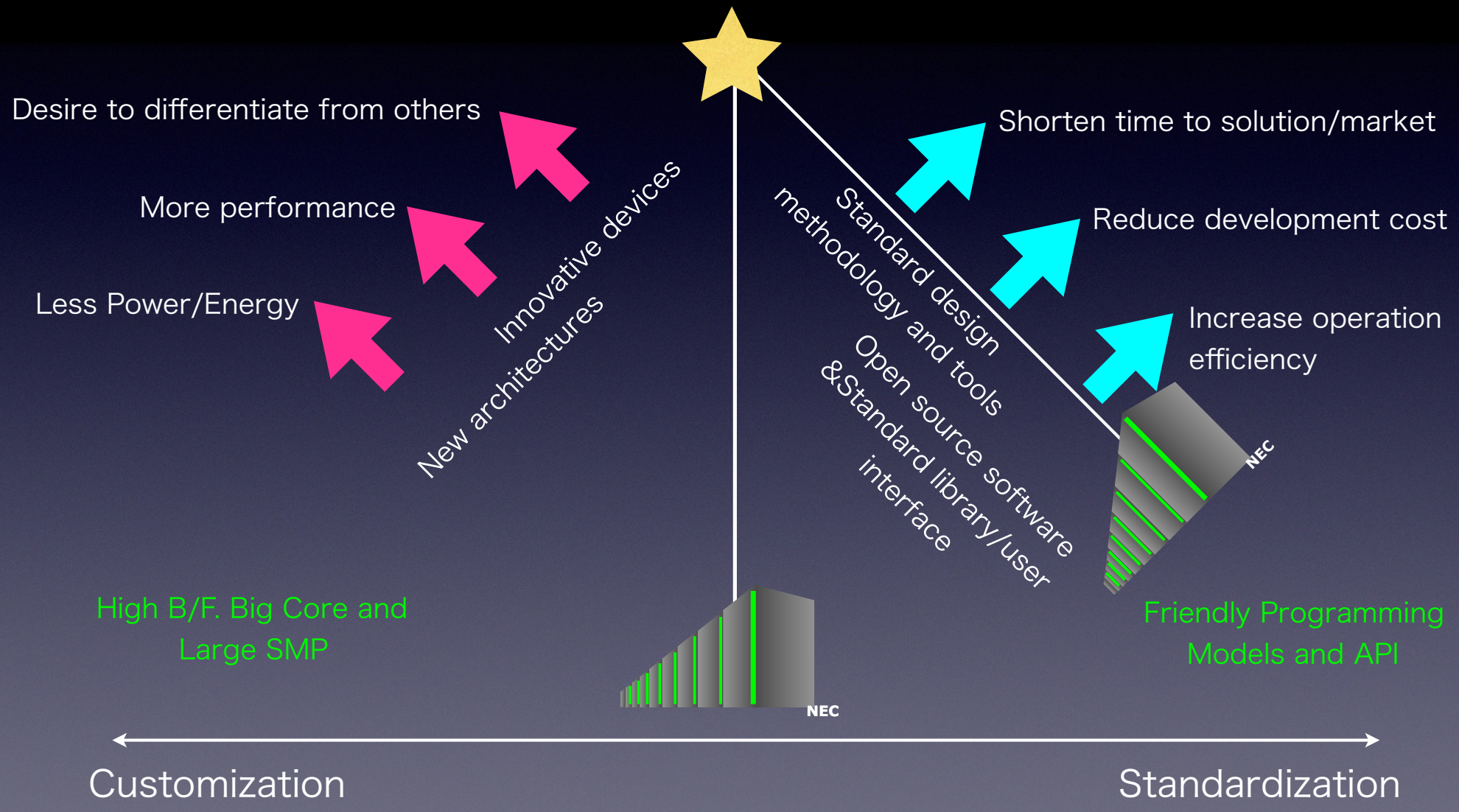


★So, we have to become much more smart for design of Future HEC systems.

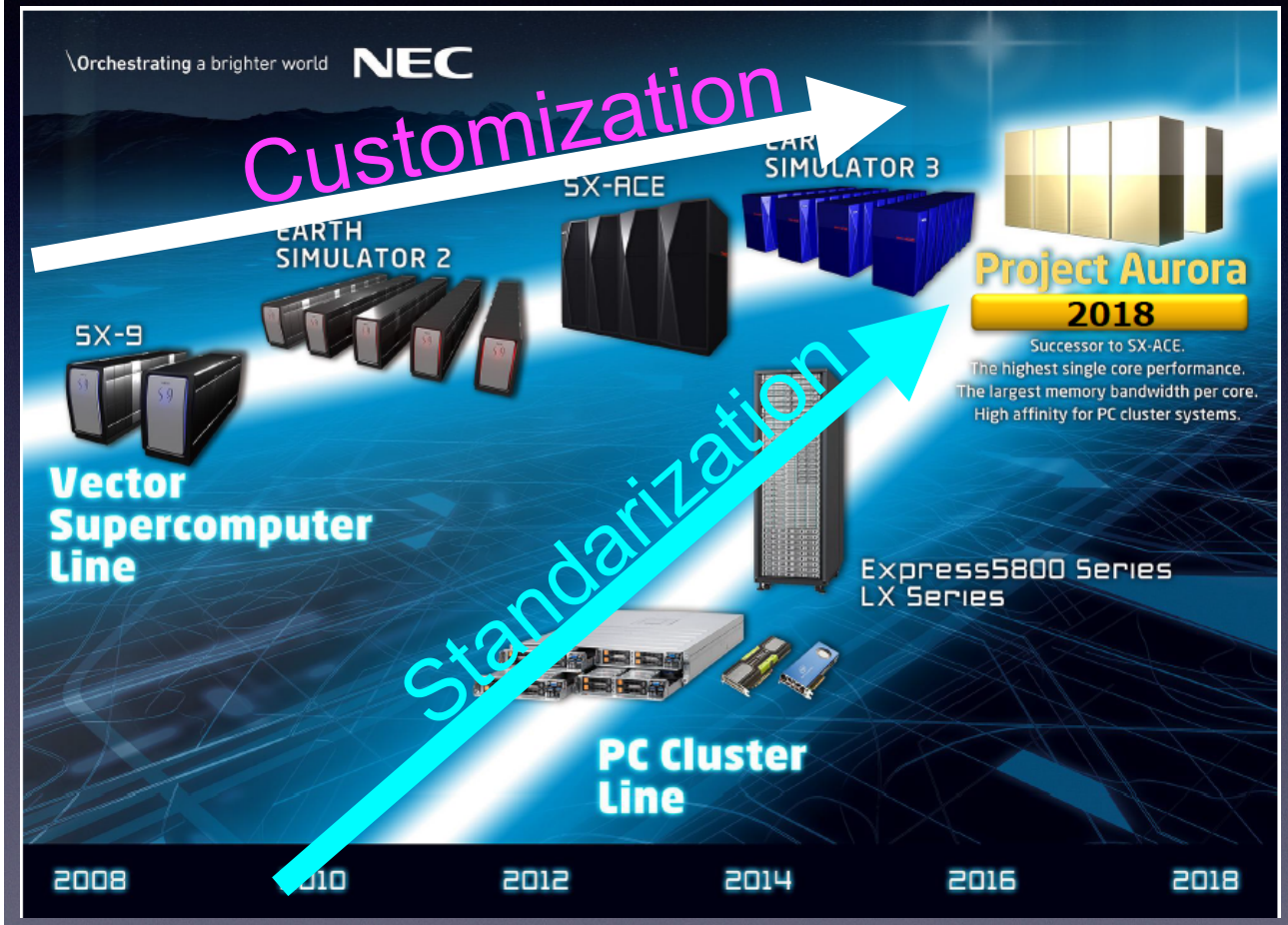
It's time to focus on Domain-Specific Architectures for computation-intensive, memory-intensive, I/O intensive, low-precision computing… etc applications to improve silicon/power efficiency!

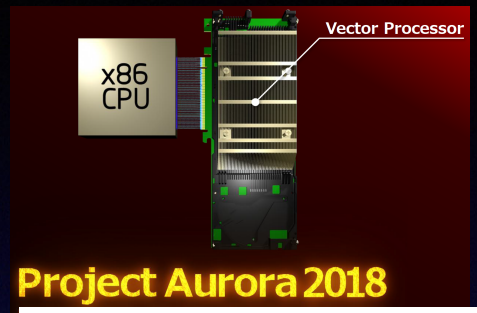
Slide at 16th WSSP in 2012

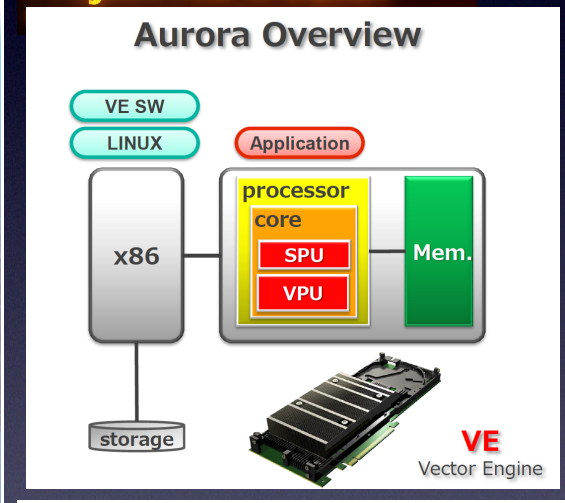
So What to Do
~NNGV(?) Design Pendulum~

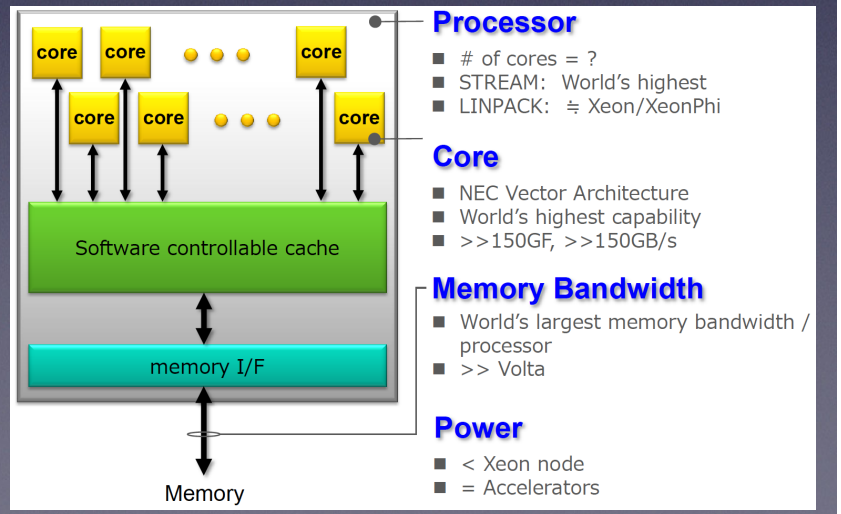


Aurora: NEC's Next Vector Product







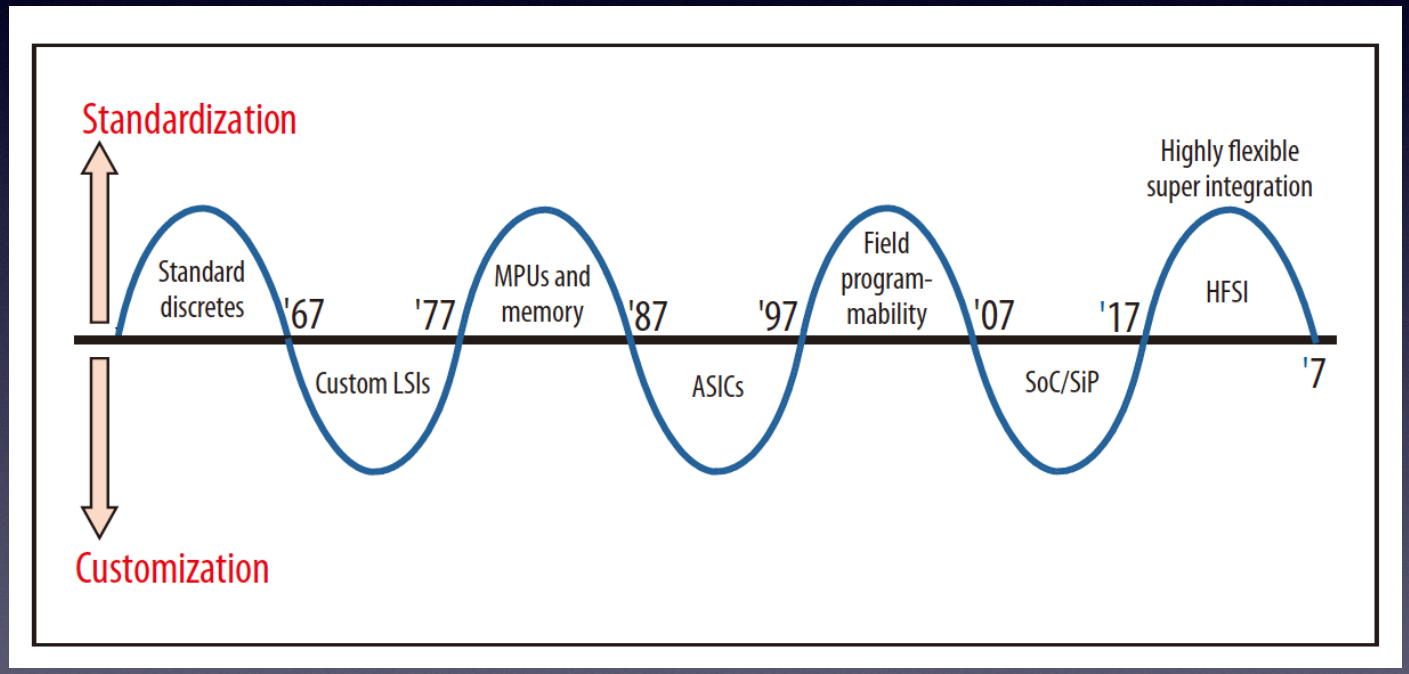


Source: NEC

Now "Makimoto"s Wave" also hits the HPC community.

Makimoto's Wave

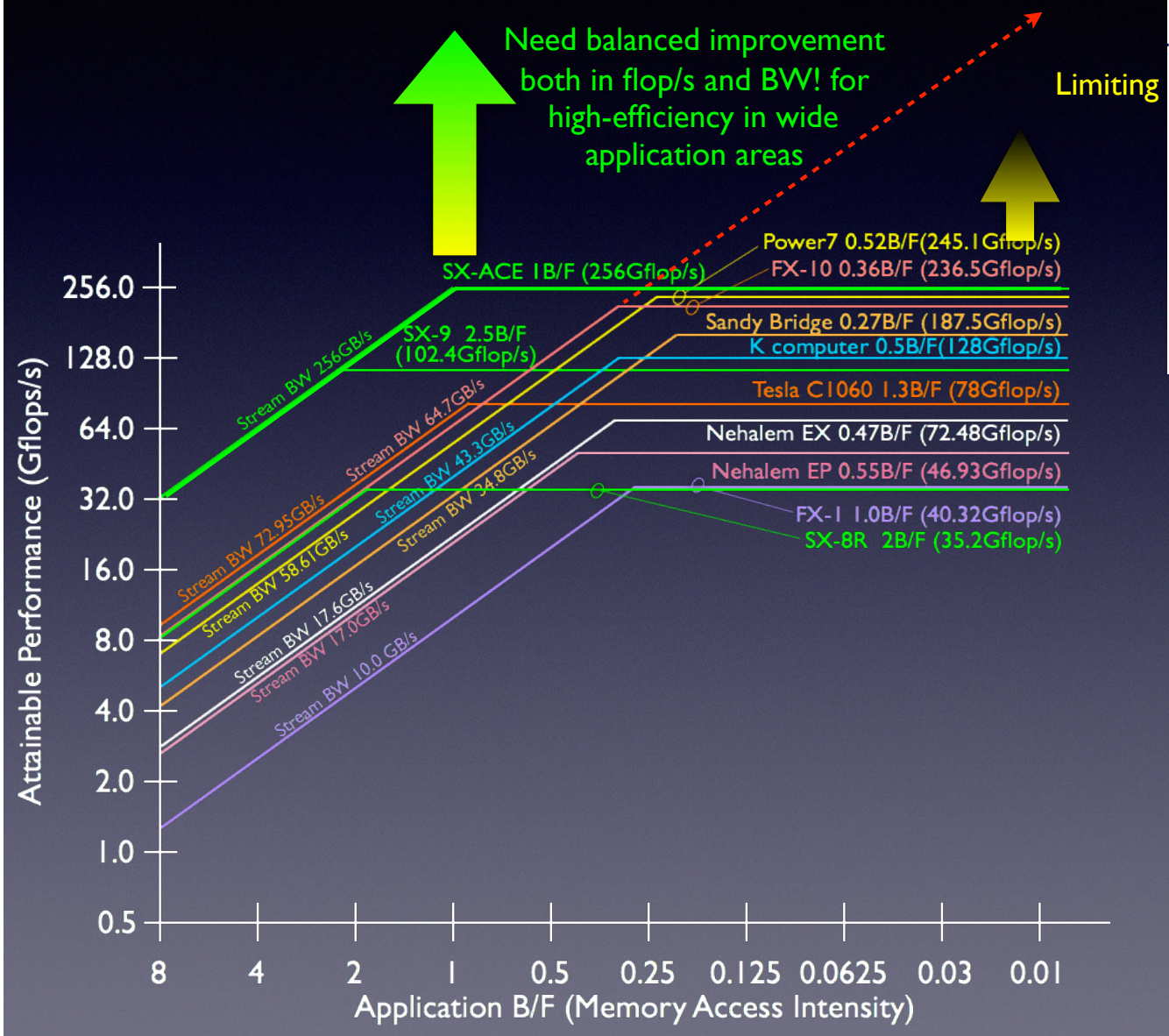
The cyclical nature of the semiconductor industry alternates between standardization and customization

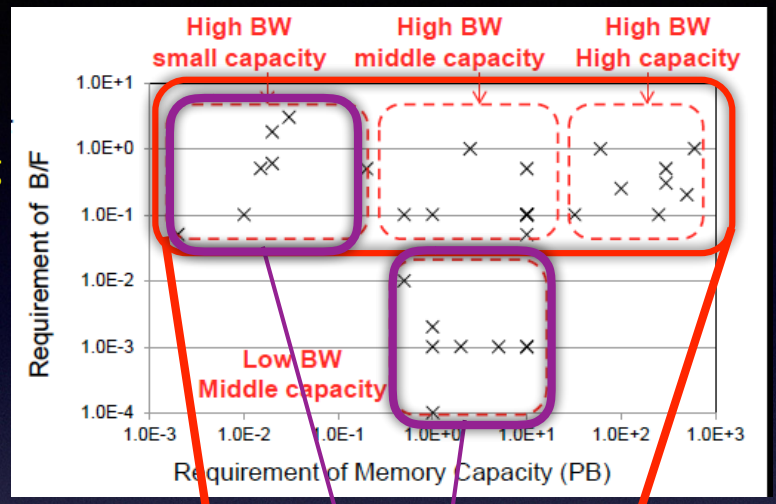


IEEE Computer Dec. 2013

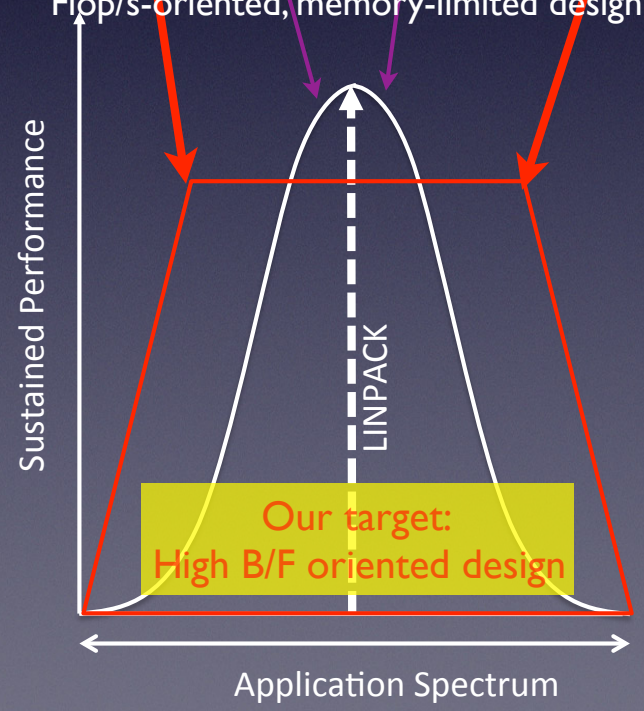
★ "one fits all" no longer make sense! means, shifting from General Purpose to Special Purpose

Domain Specific Balanced Architecture Design Approach: Not Peak Performance, Turn Memory-BW into Sustained Performance!

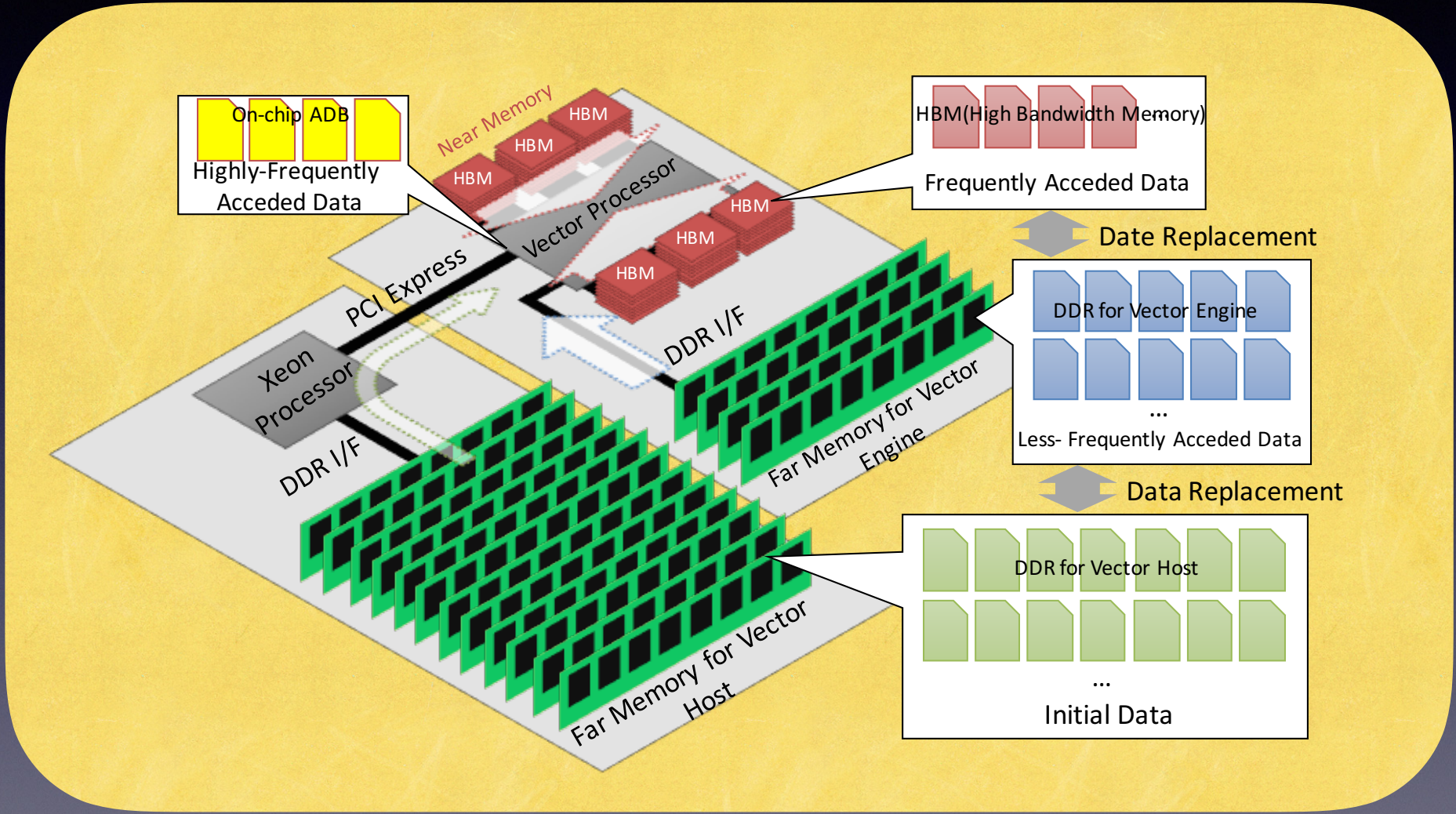




Flop/s-oriented, memory-limited design



One Feasible Solution to Future Vector Computing



TOHOKU UNIVERSITY

Design Space Exploration of the Next Generation Vector Architecture

- ★ Improving Flop/s-Memory Balance for Vector Operations
  - Effects of FMAs instead of traditional SX vector pipes
- ★ On-chip-Memory Capacity
  - Capacity becomes a help to recover the limited memory bandwidth?
- ★ Heterogeneous Memory Subsystem Design for Bandwidth-Capacity Tradeoff
  - Stacked High-Bandwidth near-memory and DDR-based Far-memory Combined.
  - A stacked high-bandwidth memory subsystem becomes a backup of ADB?

Design Space Exploration of the Next Generation Vector Architecture

★Effects of FMA s instead of using traditional vector pipes

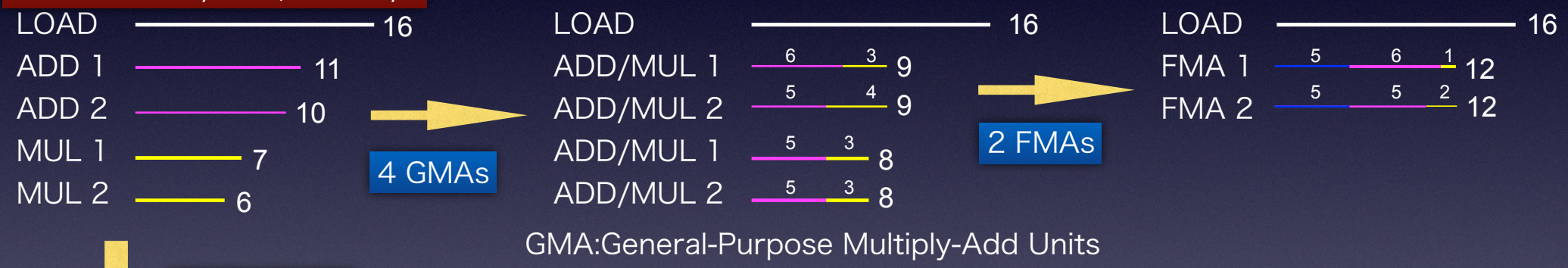
◉Operations of the Himeno benchmark kernel

LOADs	ADDs	MULs
16	21	13

FMAzation

LOADs	FMAs	ADDs	MULs
16	10	11	3

SX-ACE:64GF/core, 256GB/s



GMA:General-Purpose Multiply-Add Units

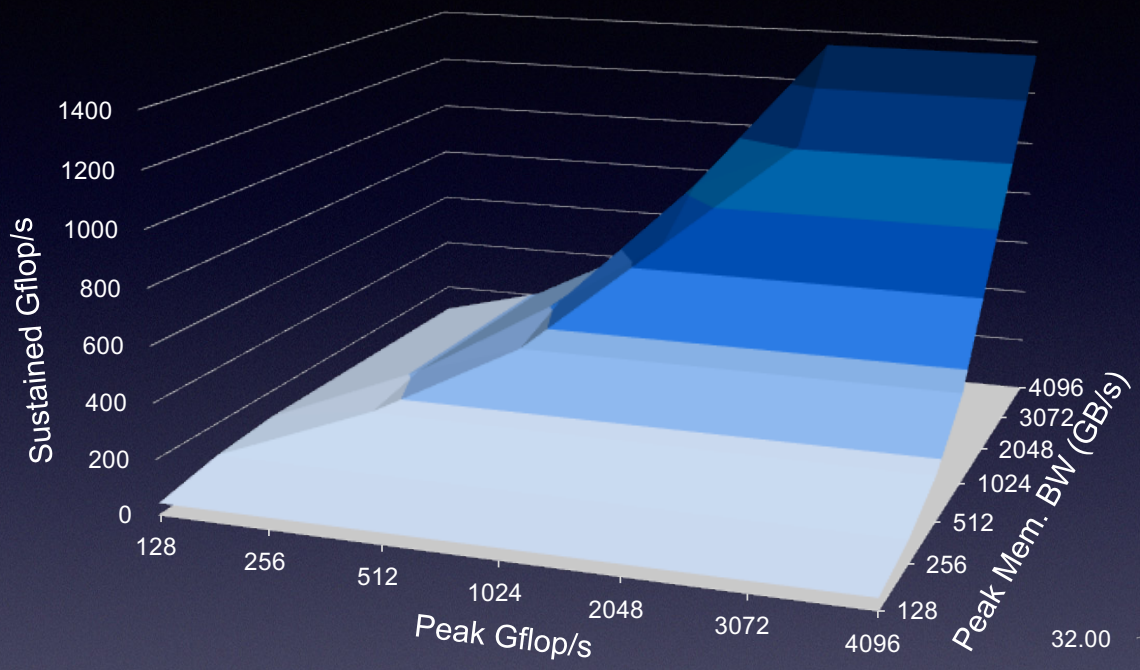
2x Mem. BW

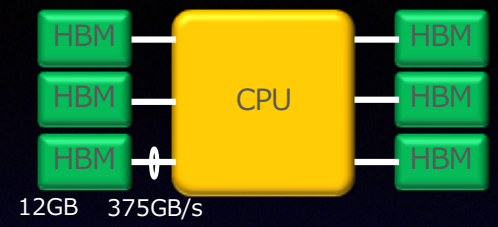
64GF/core, 512GB/s

96GF/core, 512GB/s

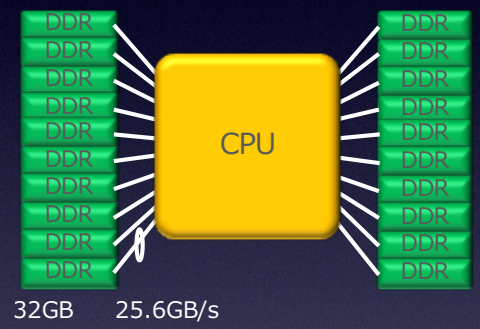
Design Space Exploration of the Next Generation Vector Architecture

★ Himeno Benchmark

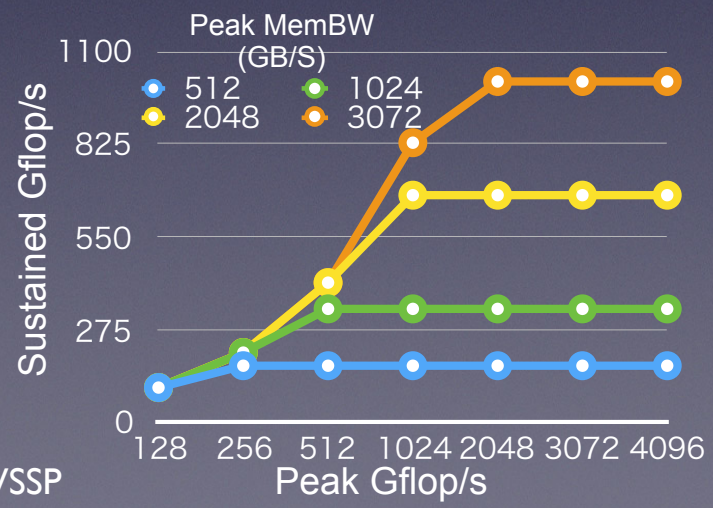


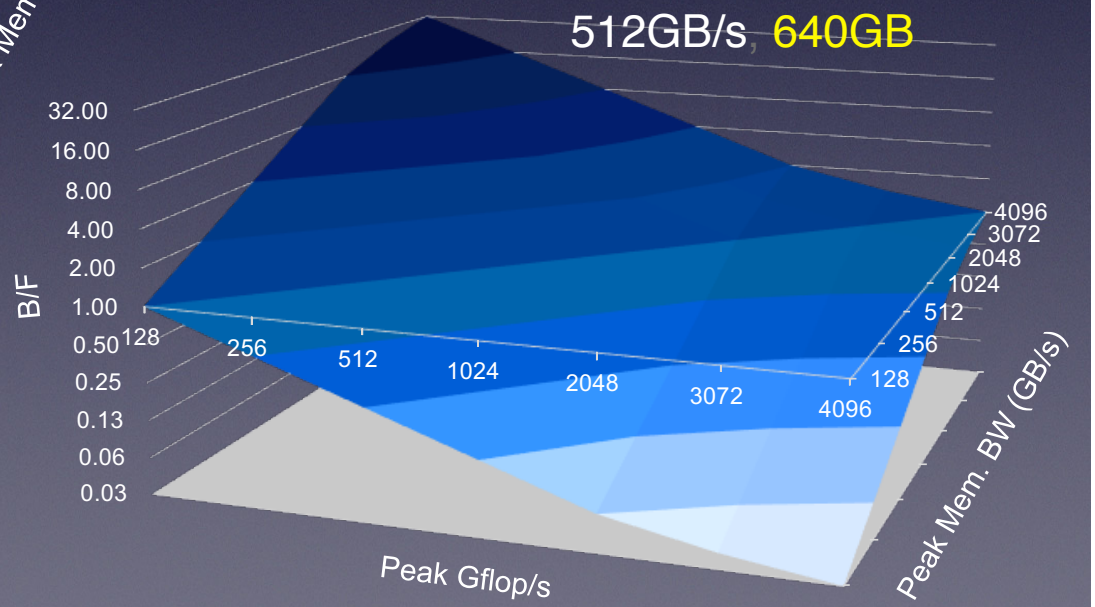


2250GB/s, 72GB

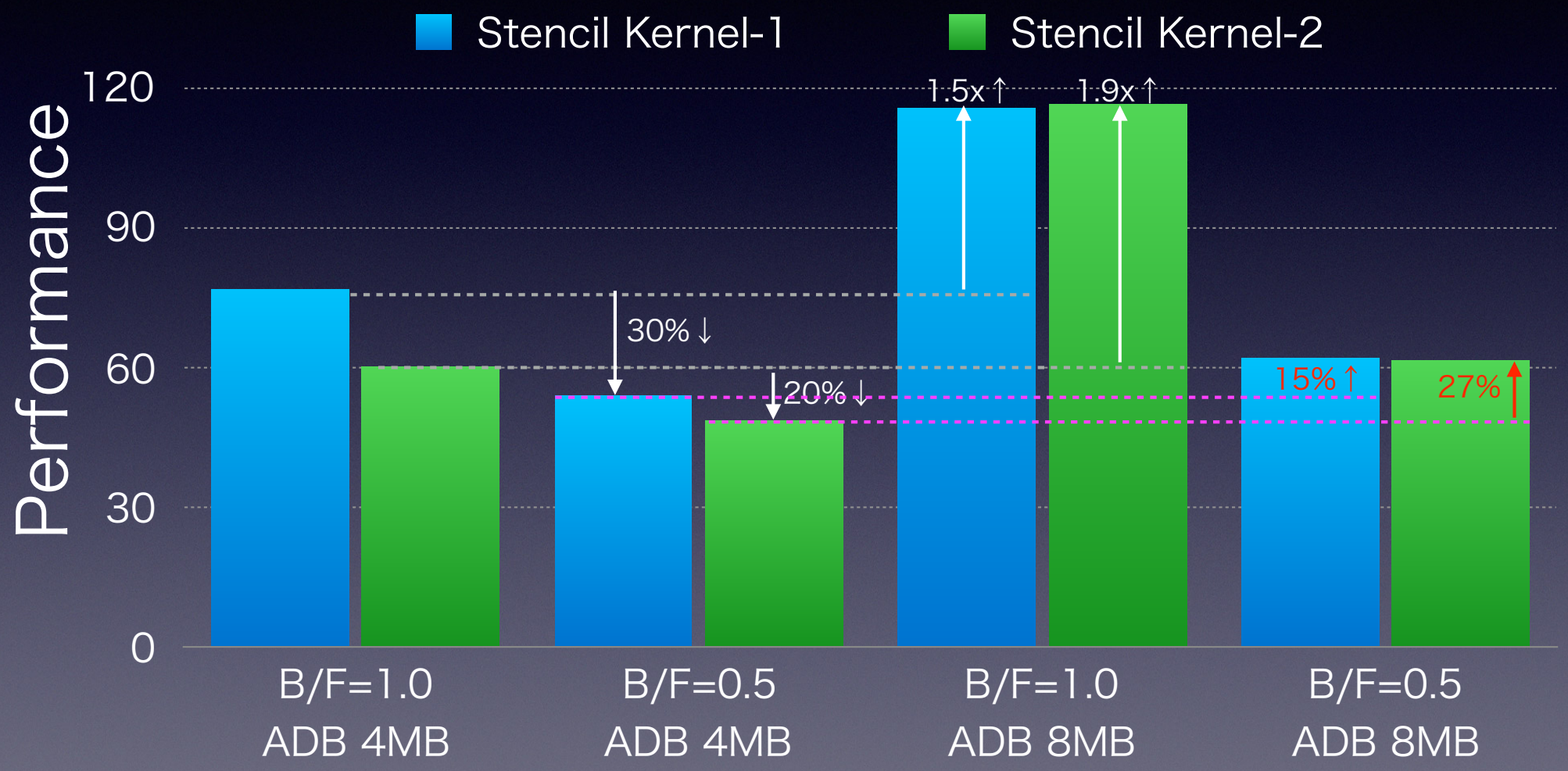


512GB/s, 640GB





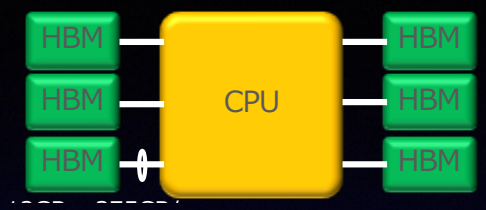
Increasing ADB Capacity Becomes a Help to Recover the Limited Memory Bandwidth?



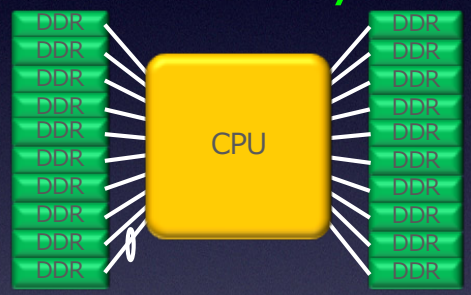
Memory Subsystem Design Choice

- ★ HBM: Bandwidth-Oriented
  - ✓ Stacked high-bandwidth memory subsystem
- ★ DDR: Capacity-Oriented
  - ✓ DDR-based memory subsystem
- ★ HMA: Bandwidth-Capacity Heterogeneous
  - ✓ Stacked High-Bandwidth near-memory and DDR-based Far-memory Combined for tradeoff between BW and capacity.

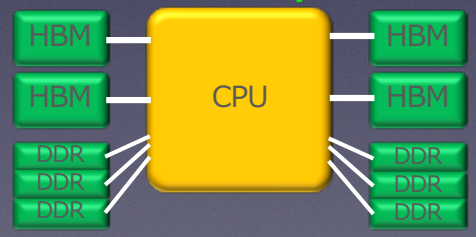
HBM only



DDR only

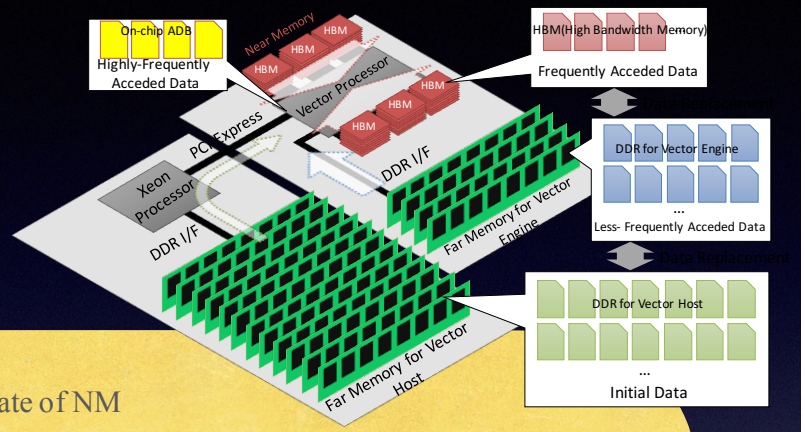


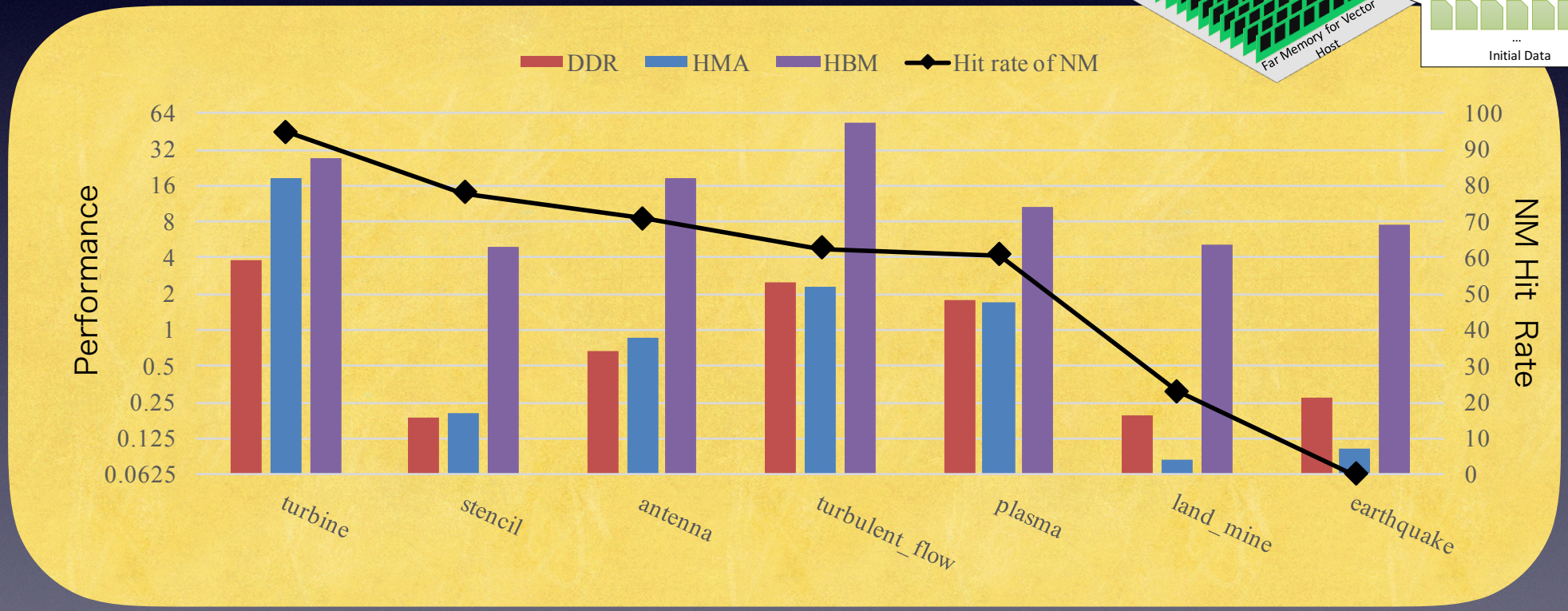
HMA: Hybrid



How can we effectively use these under the consideration of applications characteristics!?

Memory Management is Not So Easy!





Summary

★ SX-ACE shows high sustained performance compared with SX-9 and other modern HEC systems

- ✓ achieved the same single core performance in practical applications even with 60% of peak performance of SX-9
- ✓ No1. computing-efficiency and power-efficiency in the HPCG Benchmark ranking
- ✓ Pave the way to a new social infrastructure for homeland safety in Japan

★ Well balanced HEC systems regarding memory performance is the key to success for realizing high productivity and power&silicon efficiency in science and engineering simulations

- ✓ Consider domain-specific architectures with co-design with individual applications domains
  - One-fits-all does not make sense any more!
- ✓ Think different with Smart Force from Brute Force in HPC design
  - Computing Quality is the first, for power-efficient HPC , not Quantity for productive HPC!

Source: Toyota