THE UNIVERSITY OF TOKYO

Speculative aspects of high-speed processor design

Kei Hiraki The University of Tokyo

Department of Creative Informatics

Graduate School of Information Science and Technology

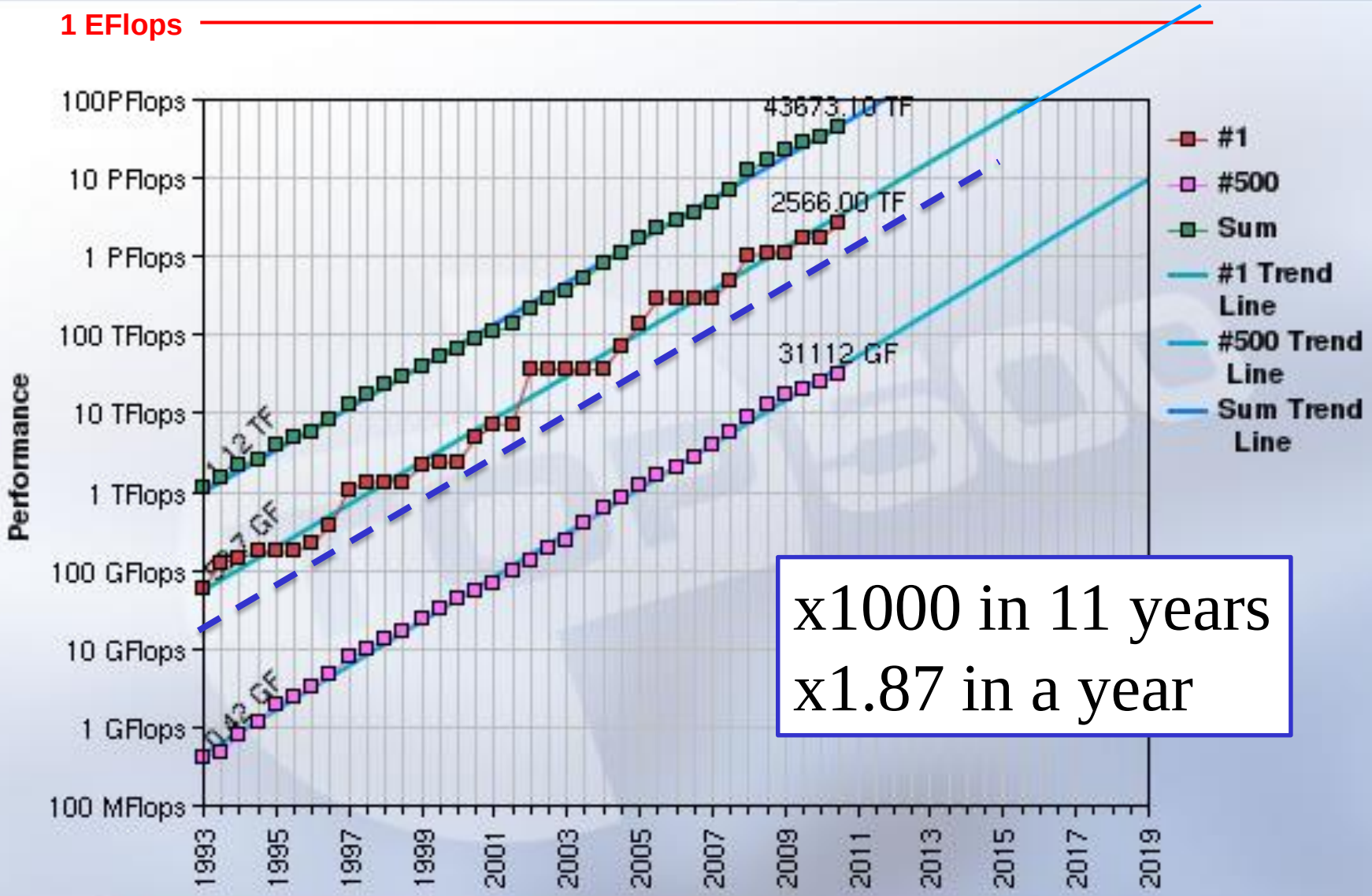
The University of Tokyo

Our goal

- Highest total system speed
  - Ex. TOP500 speed,
  - Application speed of supercomputers
- Highest processor chip performance
  - Ex. SPEC CPU rate
  - NAS parallel benchmarks
- **Highest single core performance**
  - **SPEC CPU int,**
  - **SPEC CPU fp**
  - **Dhrystone**

Single core performance is the starting point

Projected Performance Development



x1000 in 11 years
x1.87 in a year

Single Core Performance

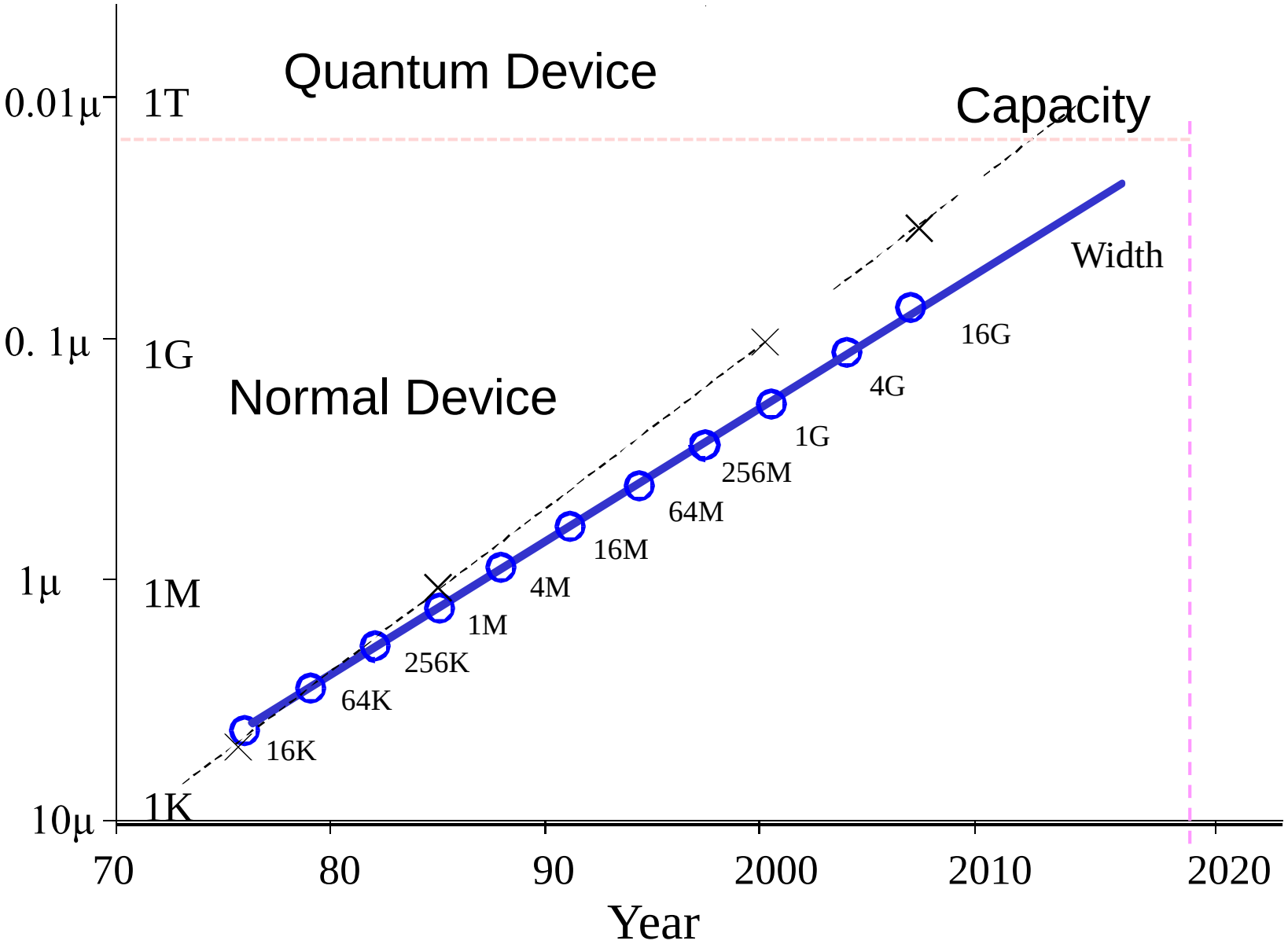
- Base for all the performance improvement
- Various speed-up methods
  - Faster clock frequency
    - New device --- Si, GaAs, HEMT, JJ device, Optical devices
    - Smaller device --- Semiconductor device width
      - Intel 4004 10,000 nm
      - Intel Corei7 3xxxx 22 nm

Clock speed is now saturating

–Power consumption

–Device density

Device Technology for Memries



Prediction byITRS

	2010	2012	2014	2016	2018	2020
Metal 1 ½ pitch (nm)	45	32	24	18.9	15	11.9
Vt (V)	0.289 EPbluk	0.291 EPbulk	0.221 UTB FD	0.202 MG	0.207 MG	0.219 MG
Vdd (V)	0.97	0.9	0.84	0.78	0.73	0.68
Power Density (W/mm2)	0.5	0.6	0.7	0.8	0.9	1
Pin count Max	4900	5300	5900	6500	7200	7900
Performance On-chip (GHz)	5.88	6.82	7.91	9.18	10.65	12.36
Performance Chip-to-Board (Gb/s)	10	14	17	30	40	50

∗ $I_{sd,leak}$ = 100 uA/um

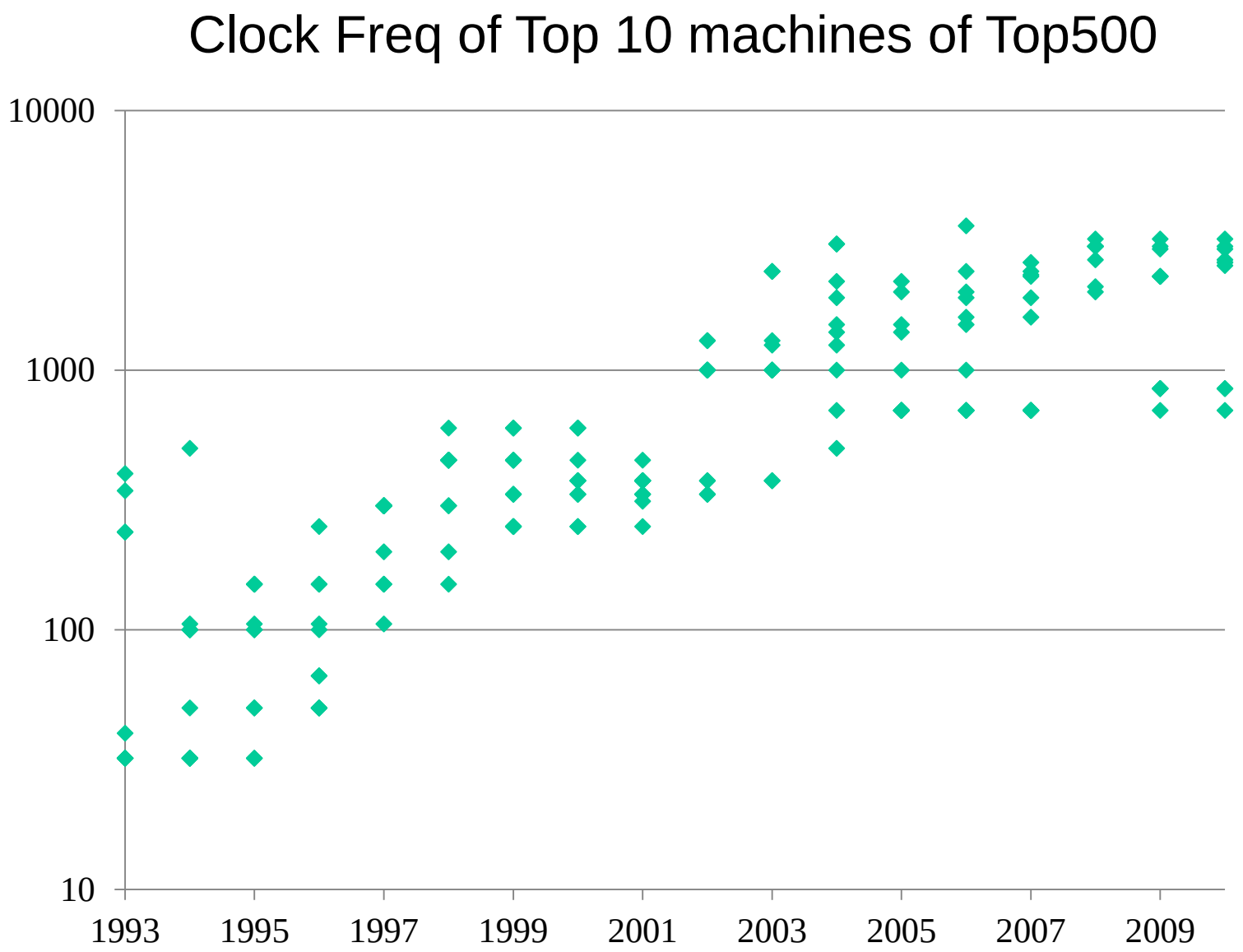
ITRS 2009 Process, Integration, Design and System / Assembly and Packaging

Power wall --- limitation of clock speed

- 100x faster clock from 1993 to 2003
  - Progress of CMOS technology

- No improvement from 2003 to Now
  - 150W/chip power limit
  - More transistor / area size
  - Faster clock requires higher threshold voltage
    - High-speed 1.2V
    - Low power 0.8V 40nm CMOS

Clock speed limit



Historical view of processor performance

- Performance measurements of historical and latest processors（100 systems）
  - Intel 386, 486, Pentium, Pen Pro, ……. Corei7, ATOM,Itanium II
  - AMD Opteron
  - SPARC　Weitek 8701、microSPARCII, UltraSparc I, II, III
  - Alpha　　21064, 21164, 21264(EV67)
  - MIPS　　R4000,　R5000, R12000
  - Power　　Power5, PowerPC750, 970
  - ARM　　Tegra, iMX515,
  - HP　　HP-PA 7100
  - Motorola　MC68328, MC68VZ328

※ Repair and maintenance are the biggest problems

1989 SHARP X68000 PRO HD
SONY NWS-1460
Apple Macintosh IIci
1991 Sun SparcStation IPX
NEC PC-9801DA
1992 NEC PC-9801RA
Fujitsu FM TOWNS II HR
SGI IRIS Indigo R4000
1993 EPSON PRO-486
NEC PC-9821As2
NEC PC-9801BS2
1994 HP 9000 712/80
Sun SPARCstation 5/85
Sun SPARCstation 5/110
1995 Apple PowerMac 7100/80
1996 Advantech PCA-6144V
NEC PC-9821V13
SGI O2
Sun Ultra2 2200
DEC AlphaStation 255/300
DEC AlphaStation 500/400
1997 PalmPilot Professional
1998 Sun Ultra5
Sun Ultra60 2360
Symbol SPT 1500
1999 SGI VWS 320
Intergraph TDZ 2000 GX1
Sun Ultra60 1450
Compaq XP1000
API UP2000
2000 Apple PowerBook G3(Pismo)
SGI Octane2
2001 Shuttle FV25
Apple PowerMac G4 (Digital Audio)
Sun Fire 3800
2002 Cobalt Qube 3 Plus
Sun Blade 2000
Tyan Tiger MPX
Palm m130
2003 Apple PowerMac G4 (FW800)
Apple PowerMac G5 (7,2)
Palm Zire 71
2004 VIA EPIA-ML
IBM p5 570
Apple PowerBook G4
Intel SR870BH2
HP Integrity rx5670
Sun Fire V40z
2005 HP ProLiant DL145 G2
Leadtek Winfast K8N
2006 Sony Playstation 3
ASUS P5LD2 SE
2007 Toshiba Dynabook CX/47E
XFX nForce 780i
SH-2007
2008 QNAP TS-409
DELL Inspiron 910
NEC SX-9 4P @CfCA
J&W MINIX-780G-SP128M
Convey HC-1
2009 Buffalo Kuro-box/T4
SHARP PC-Z1
DELL PowerEdge R410
ASUS P7P55D LE
2010 Intel S5520HCR
Fujitsu Lifebook MH380/1A
Toshiba Dynabook AZ
ThinkPad X201s
2011 ASRock P67 Extreme6

Old and New systems to be measured

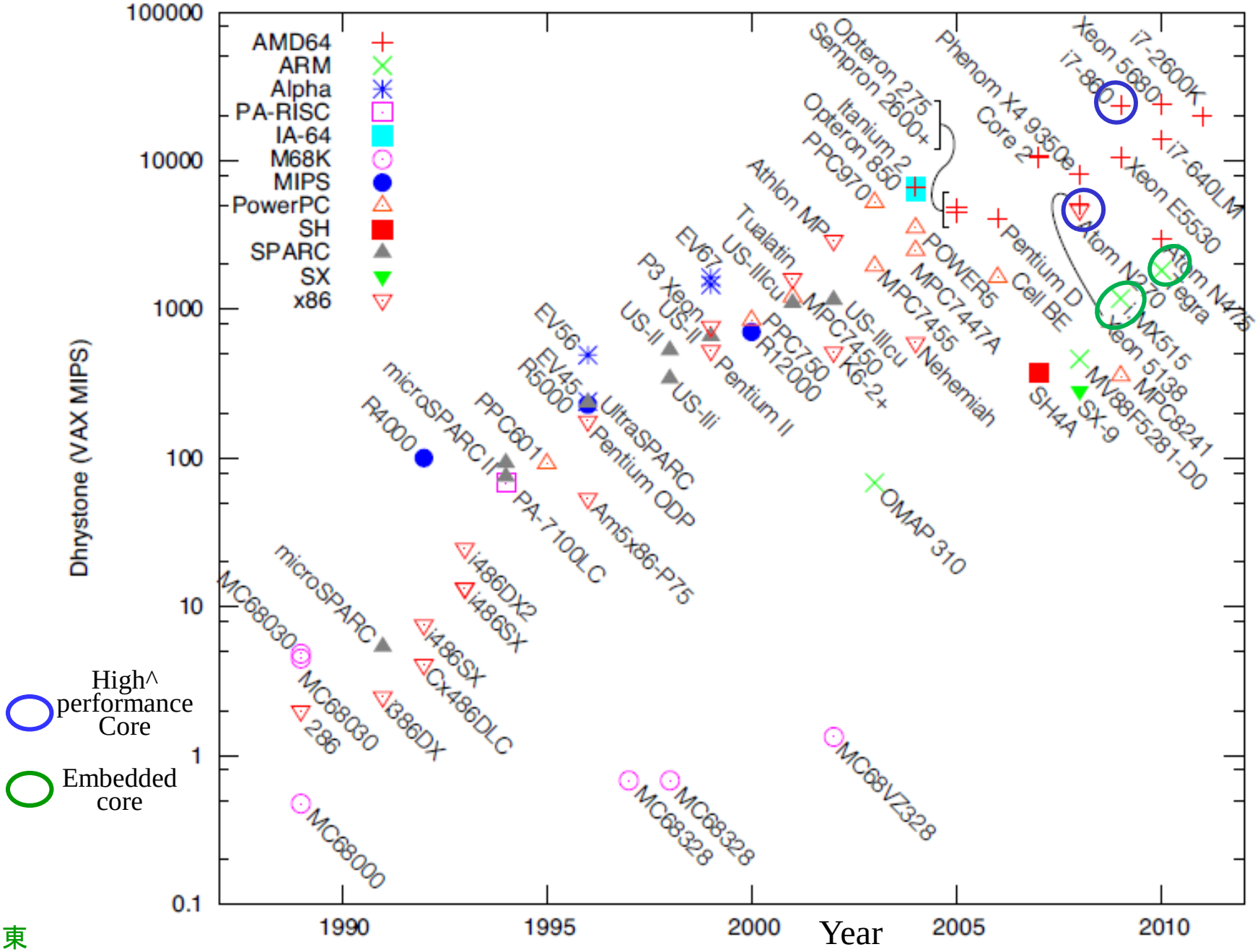






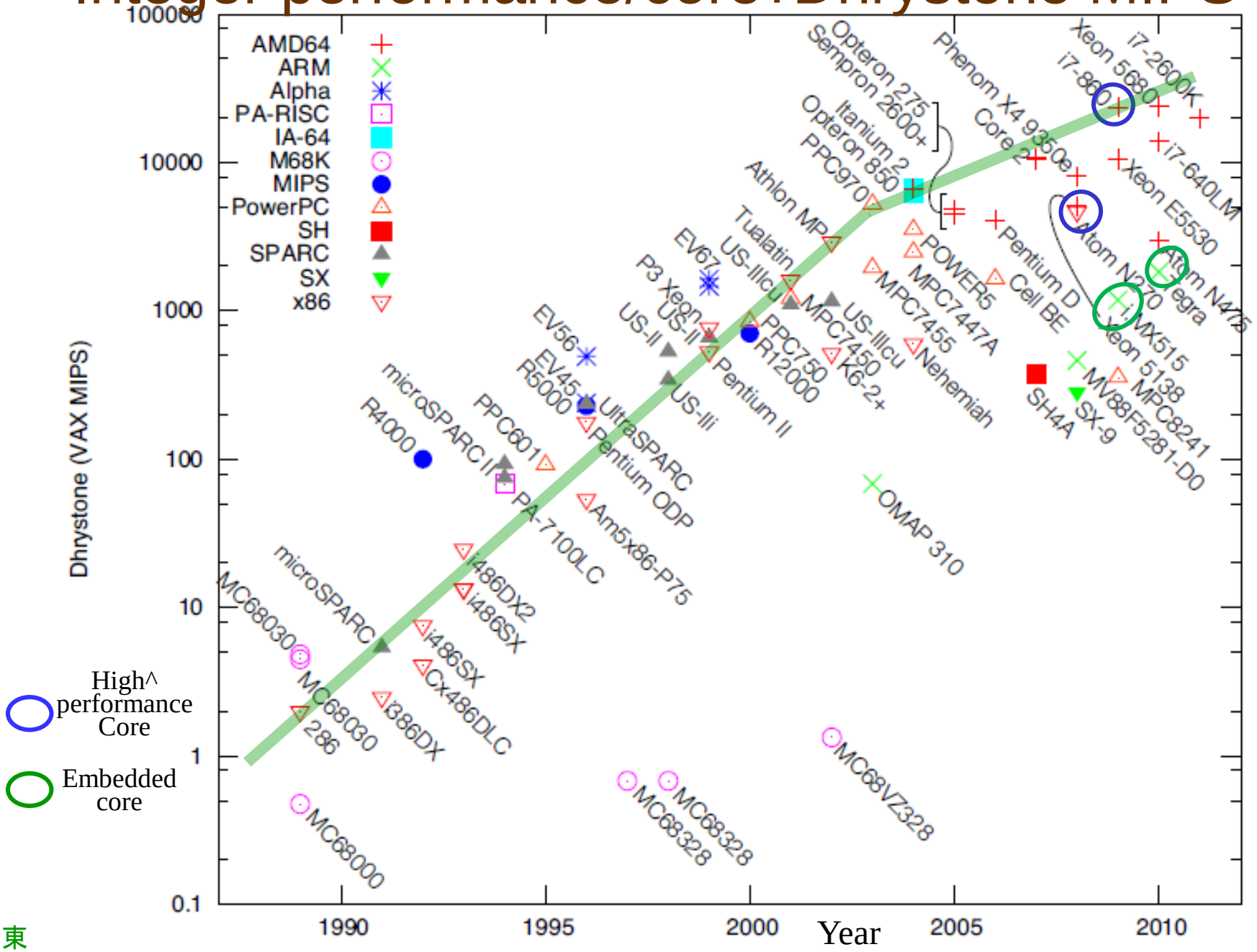



Integer performance/core：Dhrystone MIPS



東 敬

Integer performance/core：Dhrystone MIPS

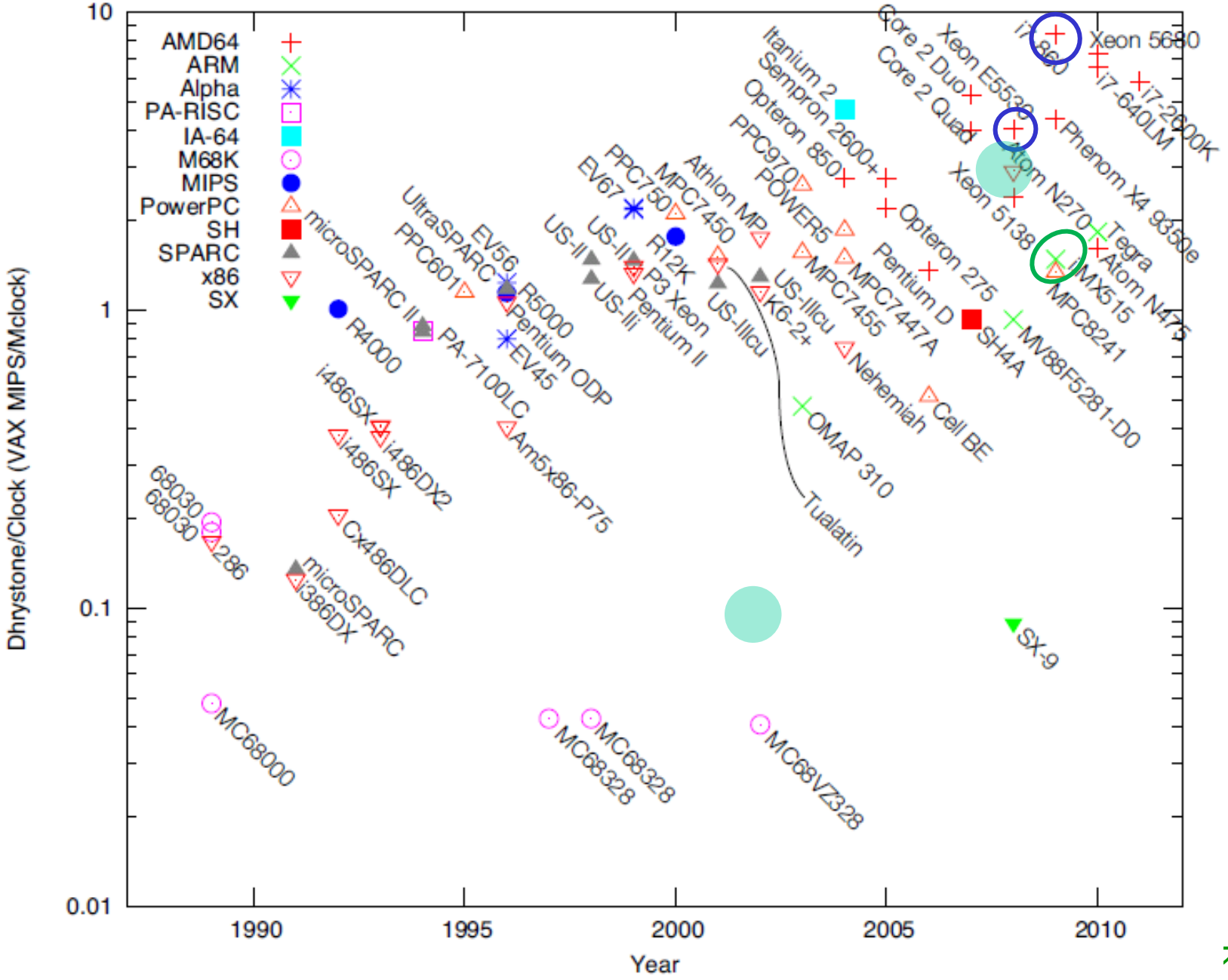


東 敬

Observation 1

- Performance rapidly increase till 2003
  - Faster clock frequency (up to 4 GHz)
  - Pipelined architecture design
  - Cache memory
  - Superscalar architecture
- Performance still increase from 2003
  - Constant clock freq.
  - Wider superscalar architecture
  - Deep branch prediction
  - Prefetching
  - ……
  - ……

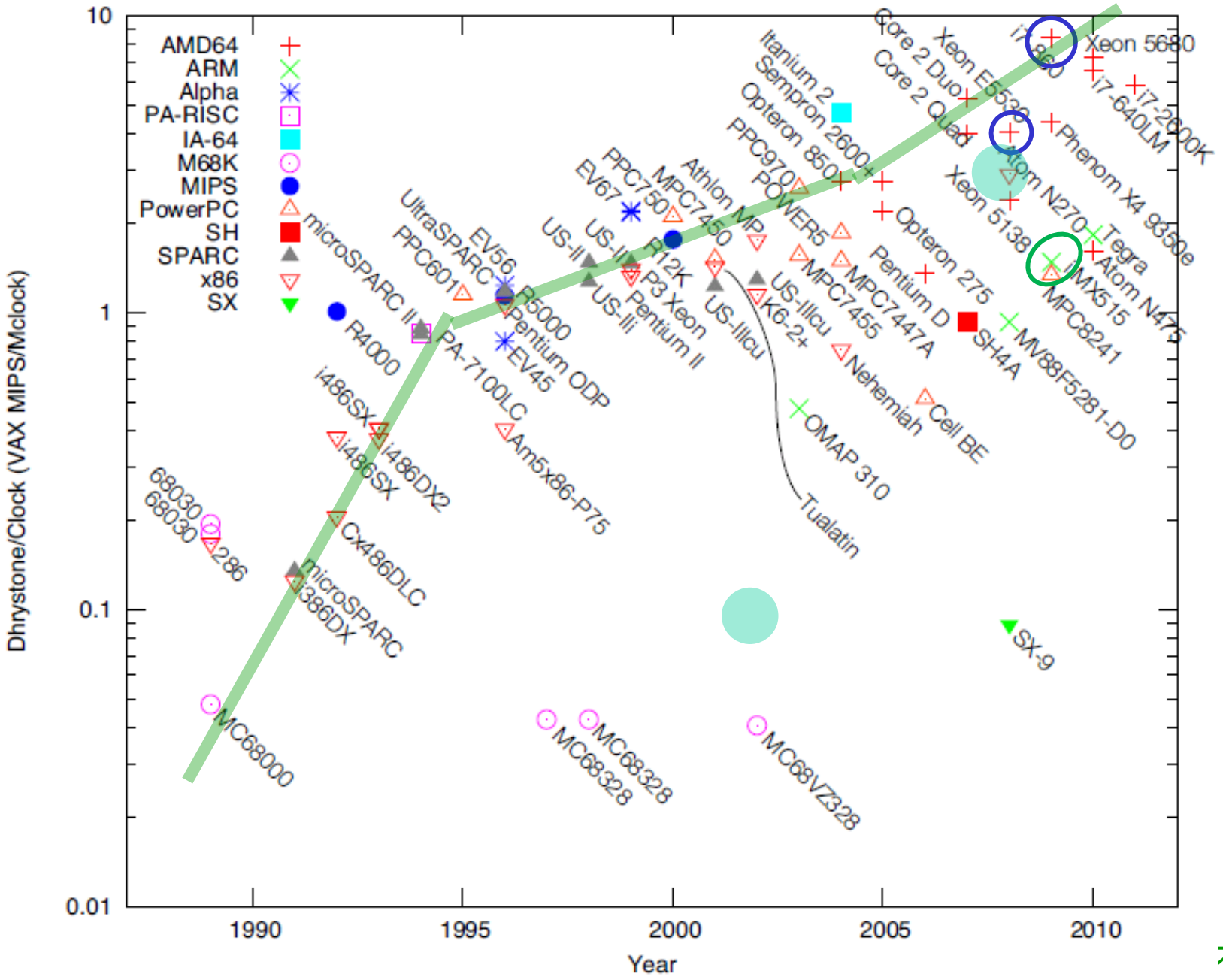
Performance / clock



東京

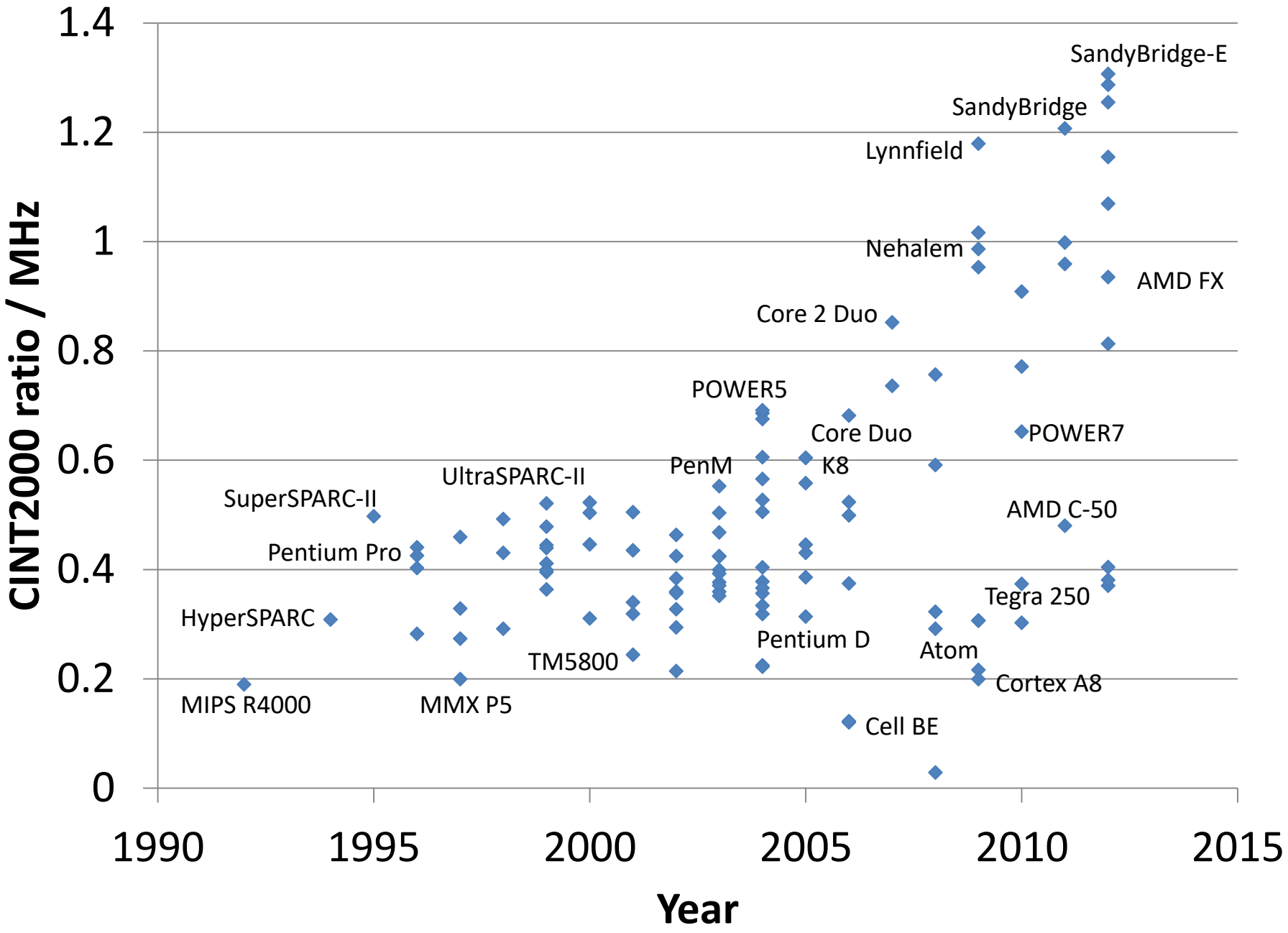
木　敬

Performance / clock



東京

木　敬


CINT2000 ratio / MHz
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1990
1995
2000
2005
2010
2015
Year
SandyBridge-E
SandyBridge
Lynnfield
Nehalem
AMD FX
Core 2 Duo
POWER5
Core Duo
POWER7
PenM
K8
UltraSPARC-II
SuperSPARC-II
AMD C-50
Pentium Pro
Tegra 250
HyperSPARC
Pentium D
Atom
TM5800
Cortex A8
MIPS R4000
MMX P5
Cell BE

High-speed features of a processor 1

1. Pipeline design
   - Sequential execution ～10 cycles/instruction
     - Old computers, Intel 8080
   - Basic pipelined execution 2～4 cycles/instructions
     - CDC6600, Manchester MU5, MIPS and many more
   - Out of order execution 1～ cycle/instruction
     - IBM 360/91, Alpha 21264, most of today's processors
   - SuperScalar execution ～0.5 cycle/instruction
     - Intel i960CA, Alpha21064, most of today's processors
   - VLIW ～0.3 cycle/instruction
     - Multiflow, Intel Itanium

- Out of order, SuperScalar should be used with branch prediction

High-speed features of a processor 2

2. Branch Prediction
   - Branch Target Buffer — Short and local history
     - Manchester MU5, processors before 1995
   - Two level branch prediction — history and pattern table
     - Intel Pentium III, and many more processos
   - Gshare and Gselect — Use of global history
     - AMD, Pentium M, Core2,
   - Perceptron predictor — Machine learning
     - AMD
   - ITTAGE — Cascaded history table

Practical use of speculative execution

High-speed features of a processor 3

3. Prefetch (hardware prefetch)
   - Memory address prediction for future accesses
   - Access throttling for optimal memory utilization

– Sequential Prefetcher	Next block
– Stride Prefetcher	Finding stride from history
– Global History Buffer	Use of Global history
– **Address Map Matching**	**Current State Of The Art**

Effective speculative execution

Practical use of global history

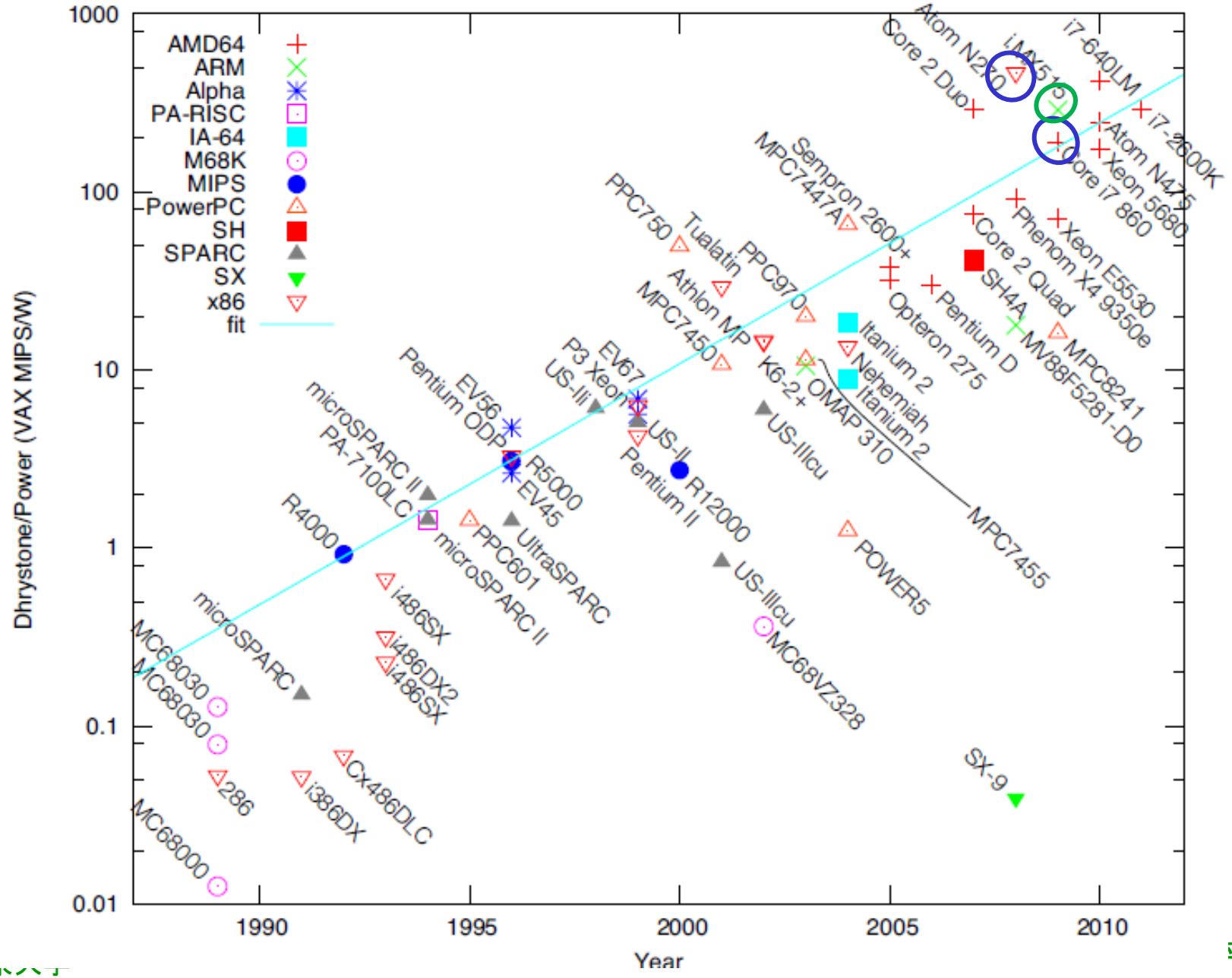
Other High-speed features

4. Cache memory, hierarchical cache memory
5. Cache replacement algorithm
   - Non-LRU algorithms to eliminate dead blocks
5. DRAM Memory access scheduling
6. Network on Chip (NoC) scheduling

7. Accelerator (floating point accelerator)

Power and hardware budget

MIPS / core of a processor core



Dataflow execution and Speculative execution

- Dataflow execution
  - Controlled by availability of data/control
  - Ideal for parallel execution
  - Difficulty in latency reduction
  - I am a dataflow Guru (Developed still largest dataflow machine)

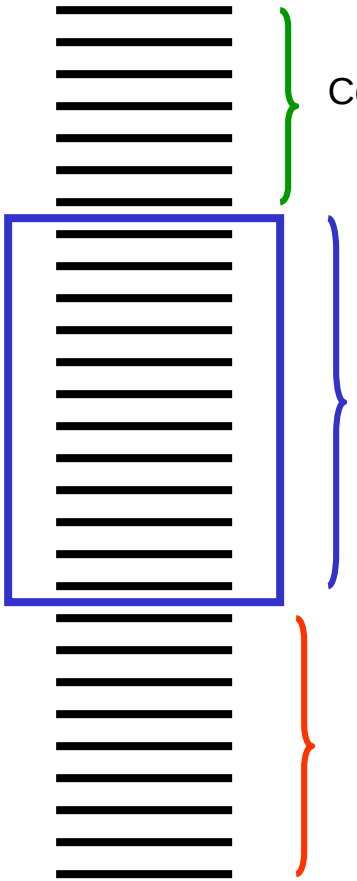
- Speculative execution
  - Independent from data dependency
  - Accuracy of prediction is the key
  - Today's speed-up of processors is mainly based on speculation

Methods for speculation (prediction)

- Local history
  - Past behavior of the target instruction
    - Outcome of the branch instruction
    - Accessed address of the load/store instruction
    - Prediction based on the patterns of local history
- Global history
  - Past behavior of instructions other than the target
    - Other branch instruction
    - Accessed address of other load/store instructions
    - For accurate prediction from the first iteration of the loop

Ideal form of Dynamic Scheduling

Executed instruction → Dynamic instruction sequence



Completed instructions

Instruction Window

Executing instructions and instructions that is wating

Instructions before execution

When a new instruction enters to the Instruction Window

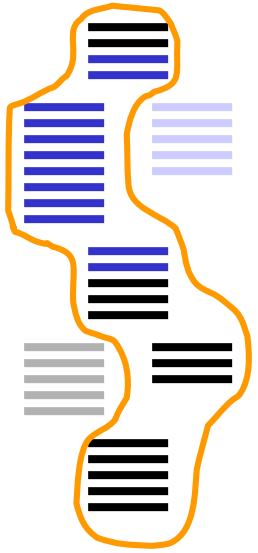
- Read registers and memory if the value on it is not reserved by instruction within the instruction window
- Operands produced by instructions within instruction window are passed directly from the instruction

Inside the instruction window、

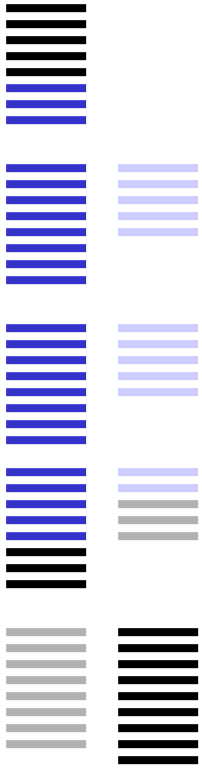
- Execution order is decided by data-dependency （Execution starts when all the input operand are ready）
- If CPU has infinite resources (ALU memory etc.) Dynamic Scheduling gives shortest execution time
-

Utilization of ILP (Instruction Level Parallelism)

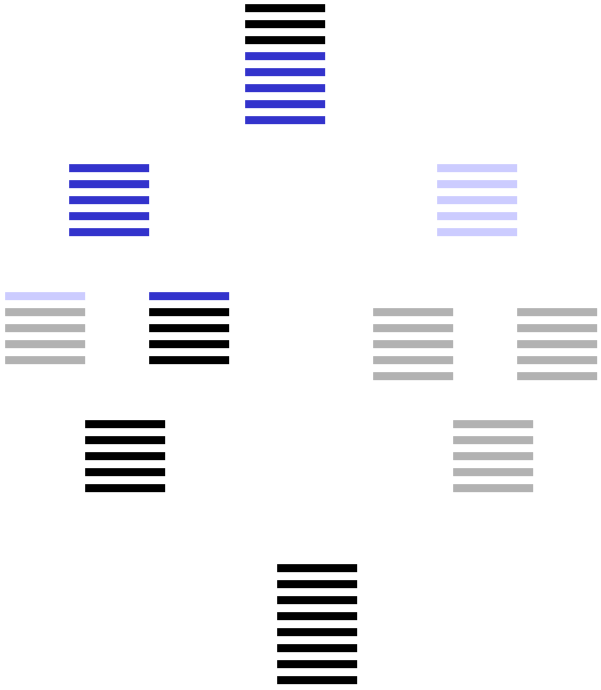
Conditional branch inside Instruction Window



if then else



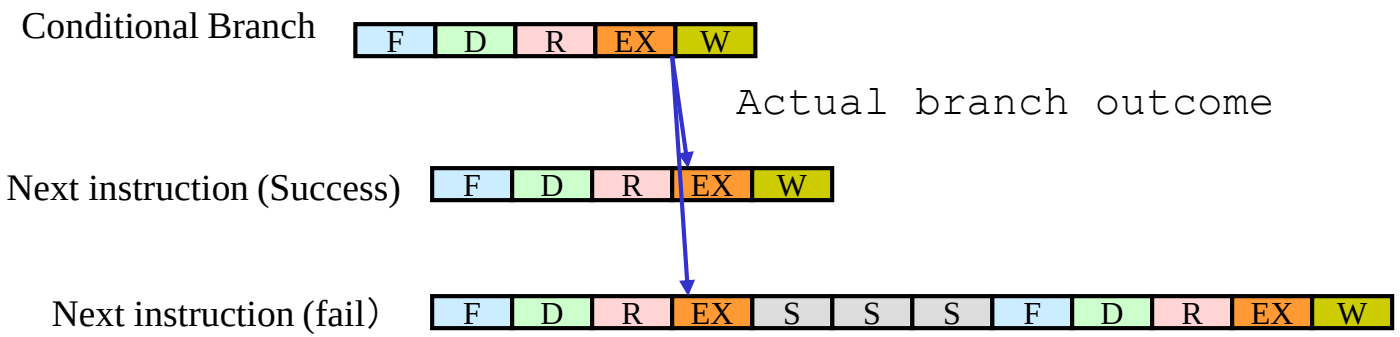
Loop constructs



Nested conditional branches

Example:Branch Prediction

- Speculative execution of instructions after the conditional branch
  - Static branch prediction
    - Compiler decides the major direction of the branch instruction
      - Alpha21064: based on the direction of branch
        （forward ⇒ Not taken、backward ⇒ taken）
      - SPARC etc.: Conditional Branch has a hint bit



Limitation of static branch prediction

- Speculative execution of conditional branch
  - Large penalty when speculation fails
    - Cancelation of speculative execution
    - Keeping memory consistency by speculative execution
  - Problem: High miss rate of prediction
    - Loop exiting branch : Loop constructs
      - 1 failure per one loop construct　　1／n Misprediction
    - Conditional branch in loop body
      - Difficult to predict statically ⇒ Profile based branch prediction

- About 80% successful prediction

History of Dynamic branch prediction

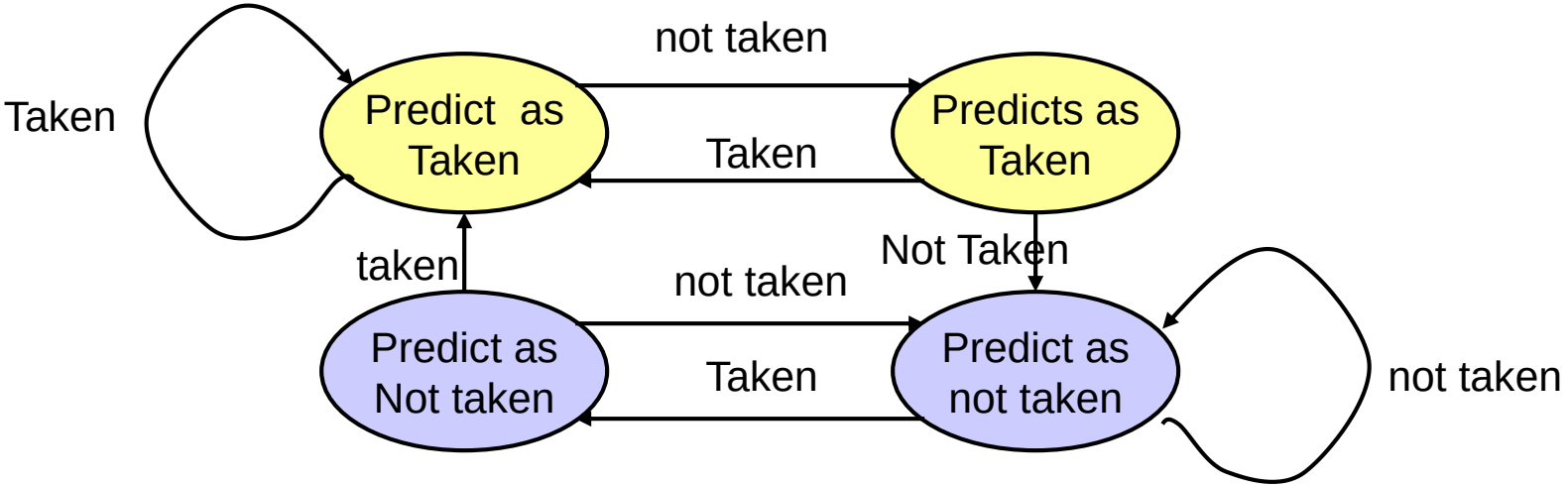
- Branch prediction table and its extension
  - Manchester University, I-Unit of MU-5 computer
    - Taylor, L. A. “Instruction Accessing in High Speed Computers,” MS thesis, University of Manchester, 1969.
  - Branch Target Buffer（Manchester Univ.）
    - Ibbett, R. N., “The MU5 instruction pipeline,” The Computer Journal, Vol. 15, No. 1, pp. 42-50, 1972.
  - 2-level branch prediction (used in Pentium III)
    - Yeh, T.-Y. and Patt, Y.N., “Two-Level Adaptive Branch Prediction”, Proc. 24th Int. Symposium and workshop on Microarchitecture, pp. 51-61, 1991.
  - **Gshare** (Used in DEC Alpha)
    - McFarling, “Combining Branch Predictors, “WRL Technical Note TN-36, June 1993.

Branch prediction table（BPT or BTB）
(Branch Prediction Table, Branch Target Buffer)

- BPT Address of branch, past branch direction (counter)

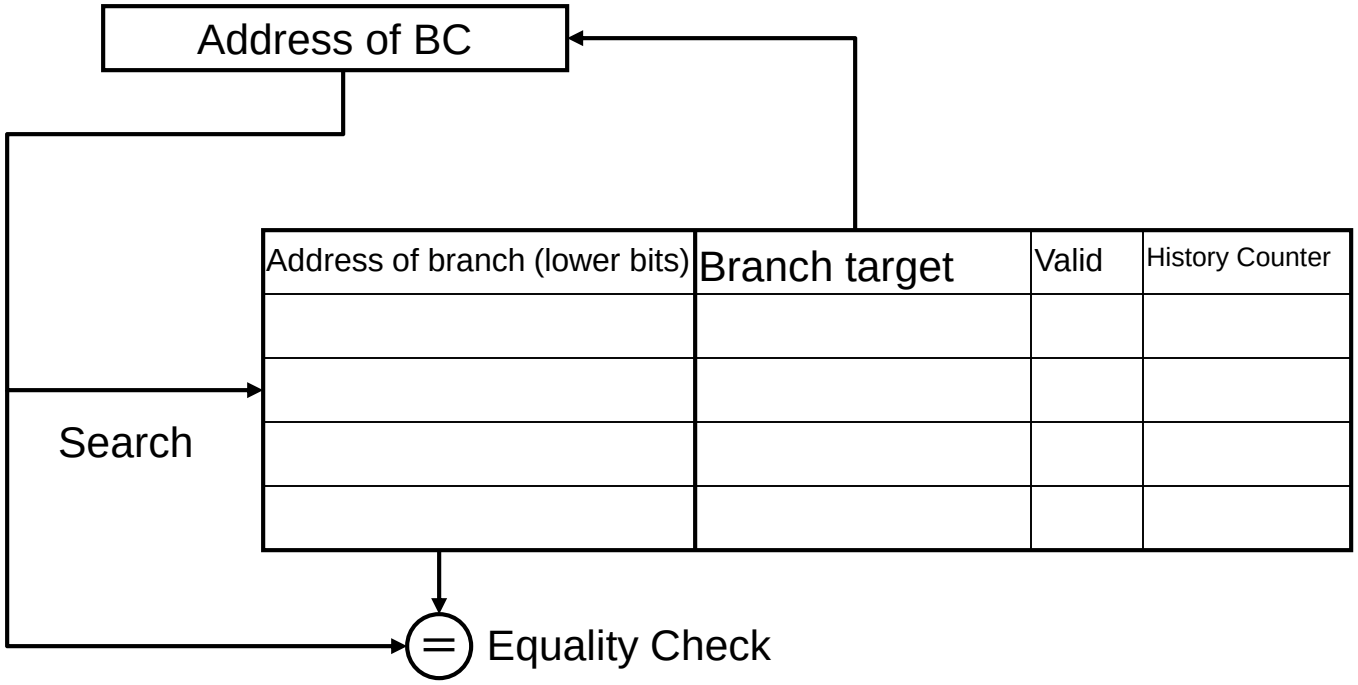
Instruction address	Valid	History Counter

- 1 bit prediction ⇒ High (x2) misprediction rate
- 2 bit prediction ⇒ Saturating counter (Bimodal prediction)



Branch Target Buffer

- Branch Target Buffer (BTB)
  - Table to get predicted branch target address
    ⇒ Zero branch prediction penalty



Further improvement of dynamic branch prediction

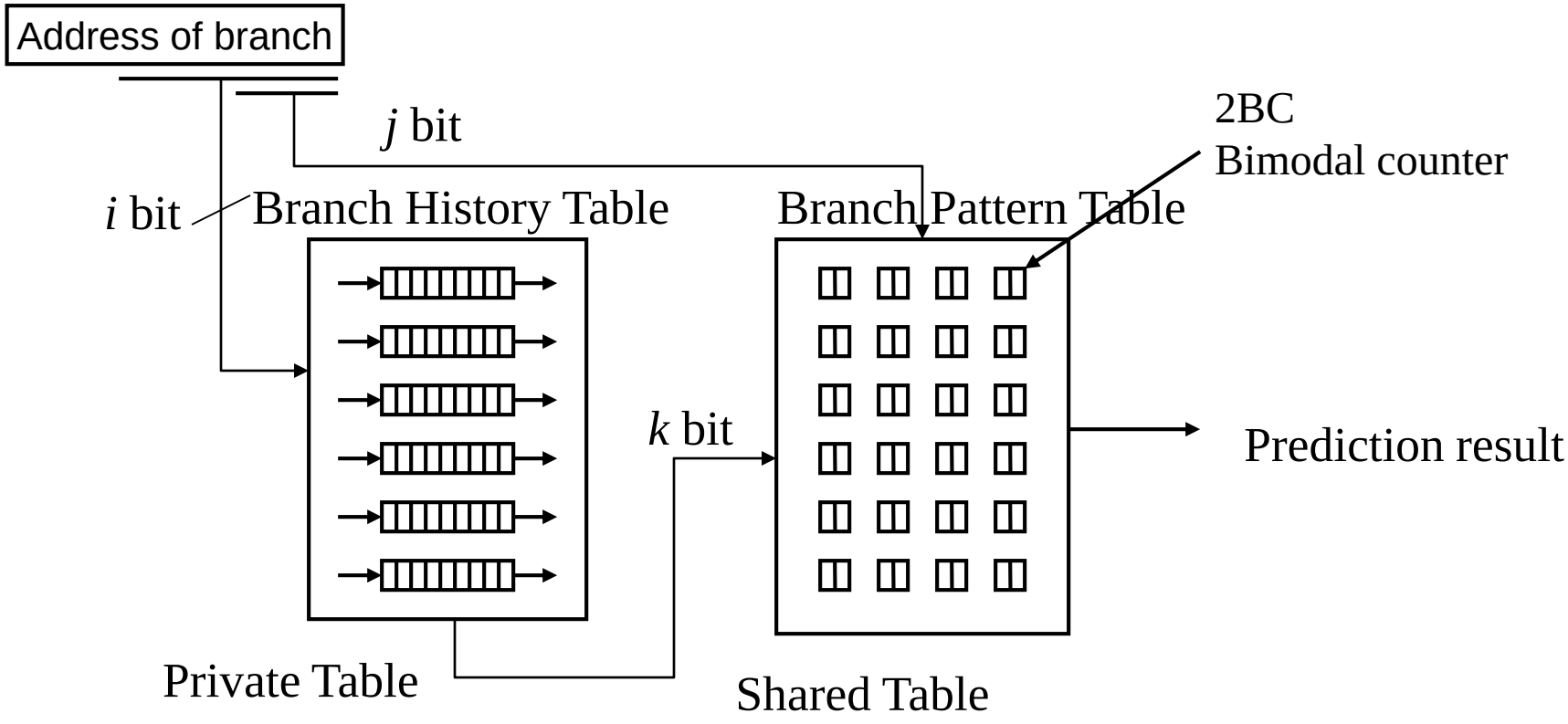
- 2 Level branch predictor (Bimodal)
  - Based on patterns of local history table
  - about 90% successful prediction
    - small loop size
    - nested loops

- 90% ⇒ about 0.35 clock penalty

2-level branch prediction

- Prediction by branch history and branch patterns

Example： PAs（Yeh and Patt, 1992）　　Intel PentiumIII



Prediction miss rate　7%　（SPECint92）

Function of 2-level branch prediction

- Detection of frequent patter of branch

- Effective to short loop (loop count < Local History Length

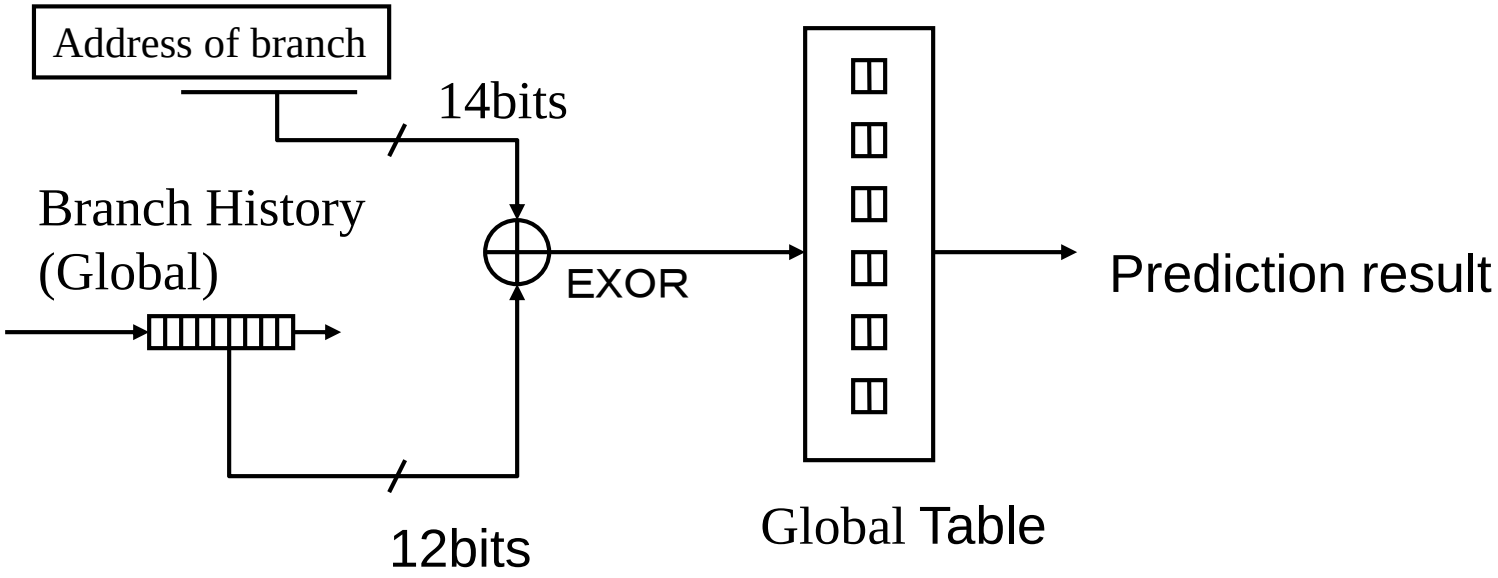
Example：Double loop (N=4)

Branch direction	TTT**N**TTT**N**	
2bc	TTTTTTTT	⇒ Miss rate 25%
PAs	TTTNTTTN	⇒ Miss rate0%

Gshare

- Inex ofPHT ＝ address EXOR history

  Ultra-SPARC3 (*j*:*k*) = (12:14)



Miss rate is about 6%

Use of Global history

```
for(i=0, I < N, i++)
   {loop body 1}
.......
.......
.......
for(i=0, I < N, i++)
   {loop body 2}
.......
.......
.......
for(i=0, I < N, i++)
   {loop body 2}
```

Hybrid branch prediction

- Combination of Local History prediction and global history prediction
- Reliability counter for each predictor
- DEC Alpha21264
- Advanced branch predictor
  - Perceptron hybrid predictor --- A kind of neural network
  - TAGE, GEHL ---- Advanced (more complex) hybrid
  - FTL ---- Path base predictor (Our predictor)

Computer Architecture Competition

Our history of competition

2012	Memory access scheduling	Winner
2011	Branch prediction	2nd place
2010	Cache replacement algorithm	2nd place
2009	Prefetching	Winner
2008	Branch prediction	2nd place

Rest of this talk

1. AMPM prefetcher

   Best prefetcher today

2. DRAM memory access scheduling

   One of best memory access schedulers

High Performance Memory Access Scheduling Using Compute-Phase Prediction and Writeback-Refresh Overlap

Yasuo Ishii, Kouhei Hosokawa, Mary Inaba, Kei Hiraki

Design Goal: High Performance Scheduler

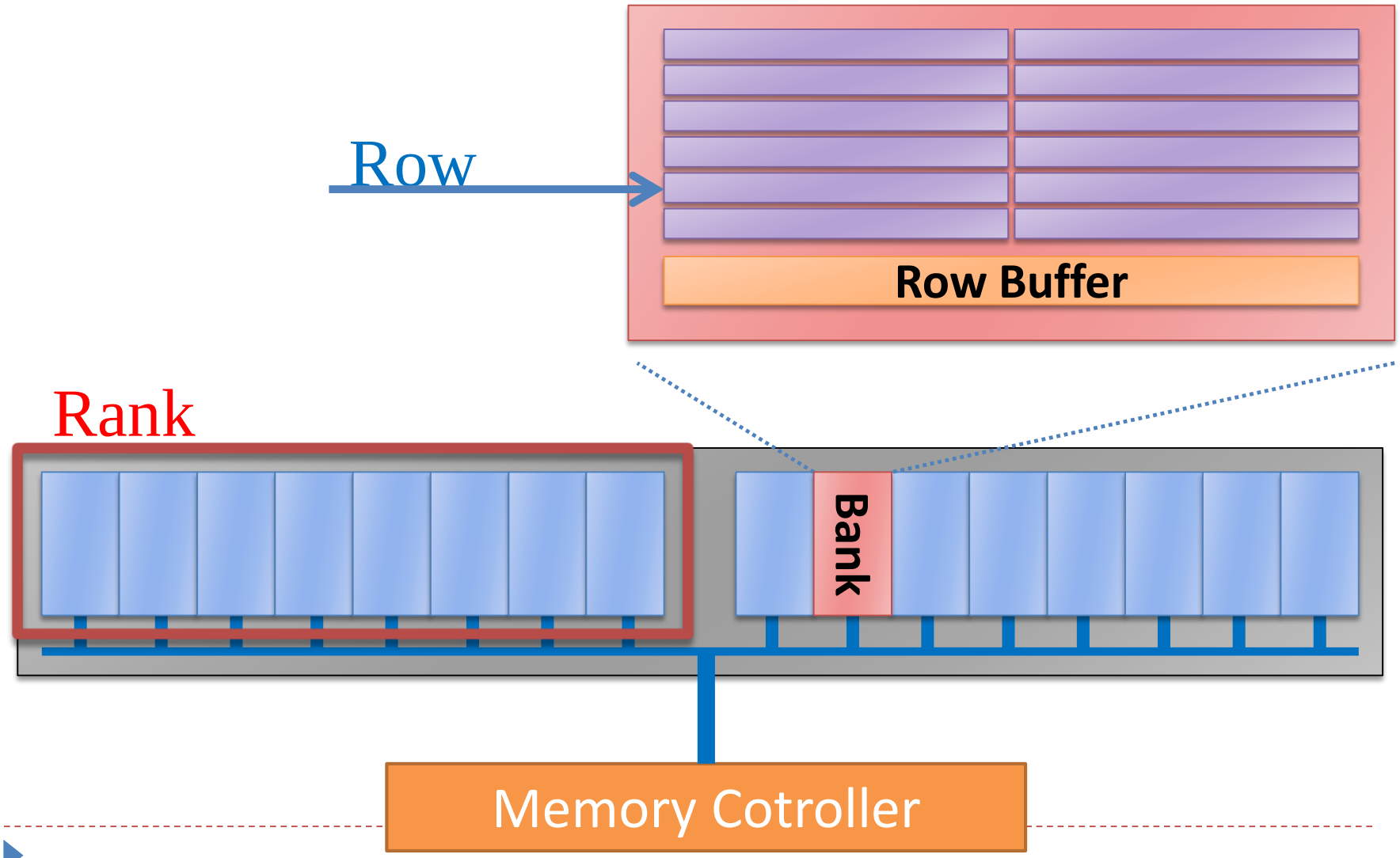
- Three Evaluation Metrics
  - Execution Time (Performance)
  - Energy-Delay Product
  - Performance-Fairness Product
- We found several trade-offs among these metrics
  - The best execution time (performance) configuration does not show the best energy-delay product

DRAM memory scheduling

- DRAM: Standard memory device for computers
  - High Density
  - Low cost

- Recent DDR3 memory has strong constraints on Row acess
  - **Row buffer access timing constraint due to power consumption**

Structure of DDR DRAM (1channel)

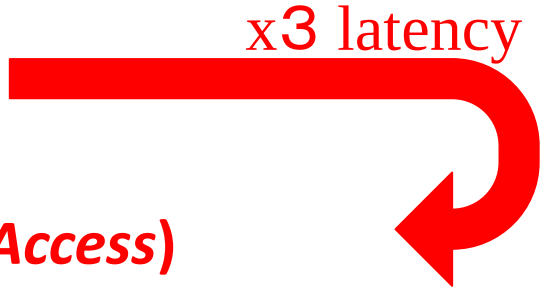


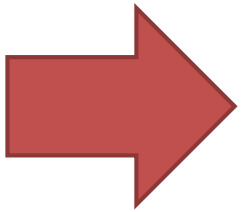
DRAM scheduling for a single core (single thread) processor

- Overhead of switching Row buffer contents
  - ① Latency
    - Row Hit access

      (Row Access)
    - Row Conflict access

      (Row Close) → ***(Row Open)*** → ***(Row Access)***
  - ② Power consumption
    - Re-write to DRAM cells (read modify write)

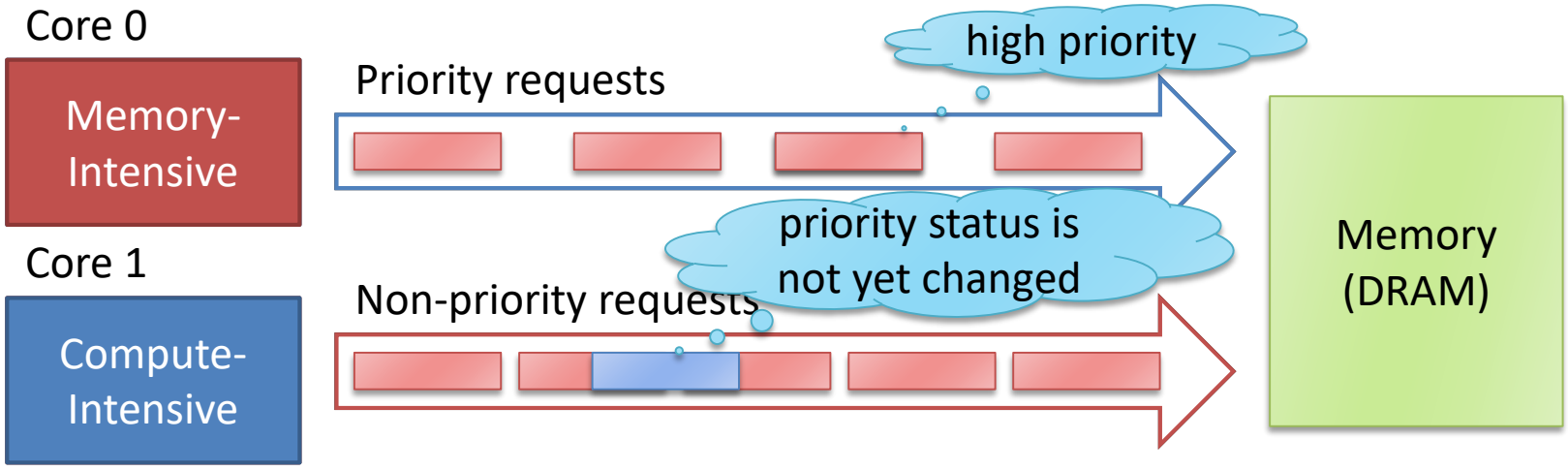




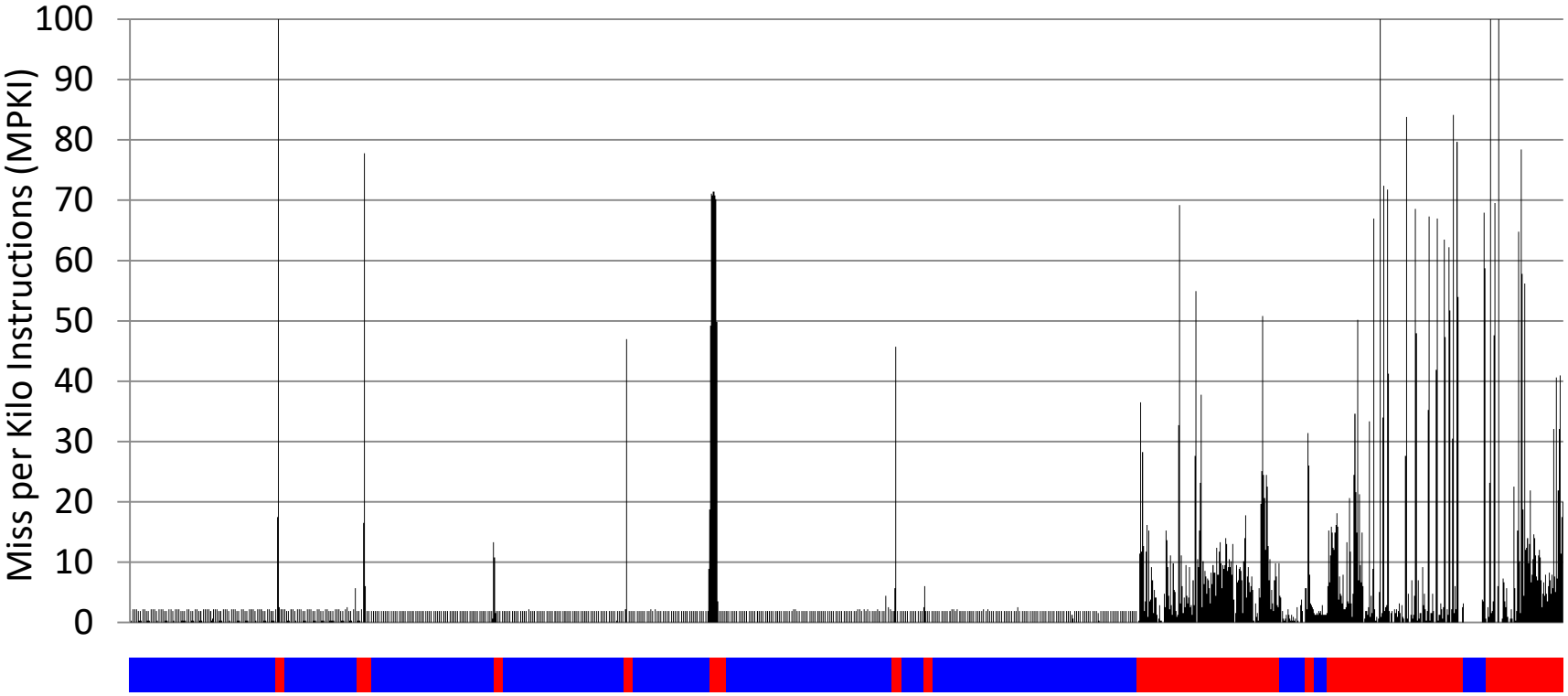
Improvement ofRow Hit ratio is important for a single thread

Thread-priority Control

- Thread-priority control is beneficial for multi-core chips
  - Network Fair Queuing[Nesbit+ 2006], Atlas[Kim+ 2010], Thread Cluster Memory Scheduling[Kim+ 2010]
  - Typically, policies are updated periodically (Each epoch contains millions of cycles in TCM)

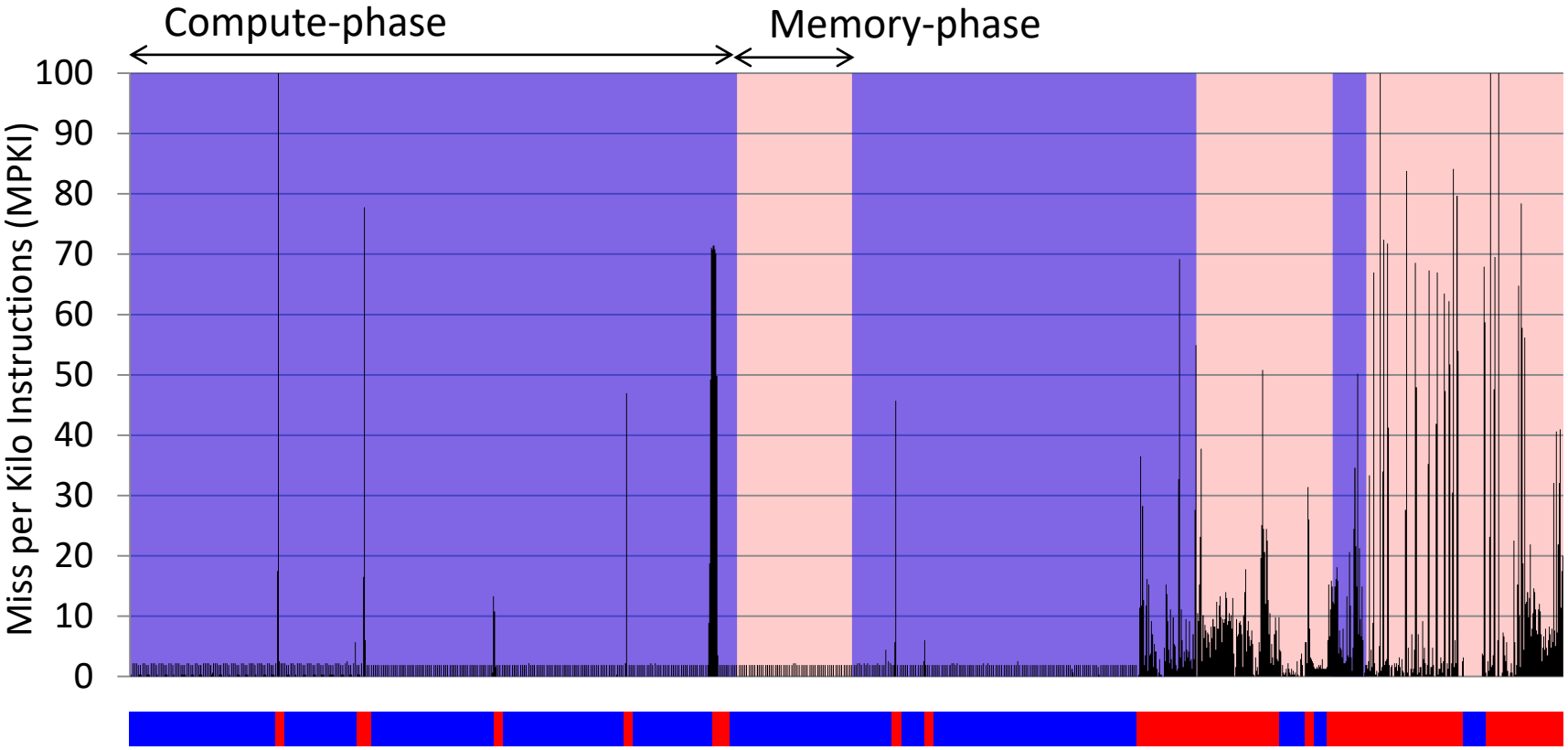


Example: Memory Traffic of Blackscholes



- One application contains both memory-intensive phases and compute-intensive phases

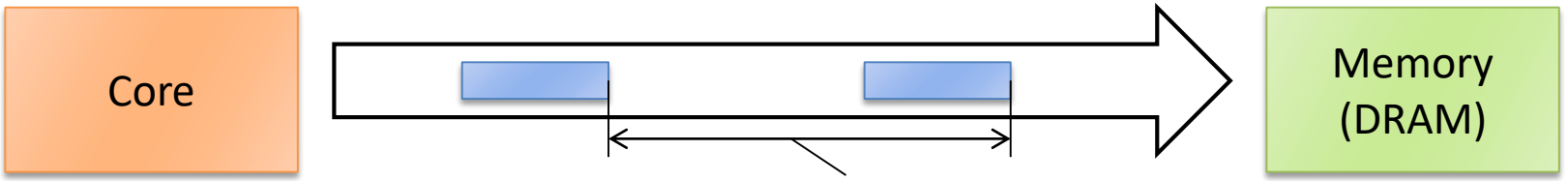
Phase-prediction result of TCM

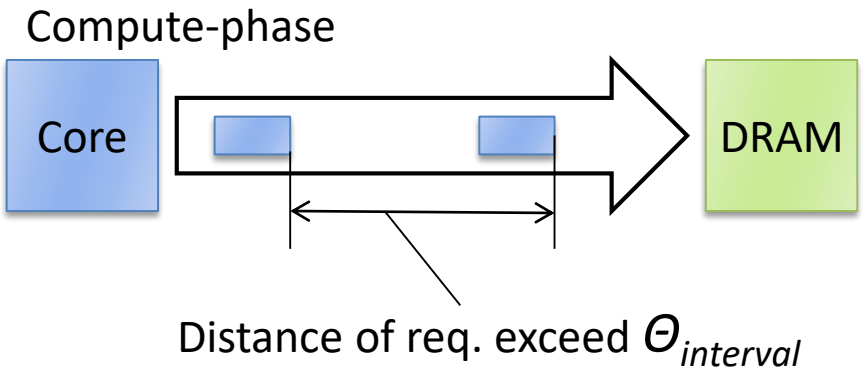


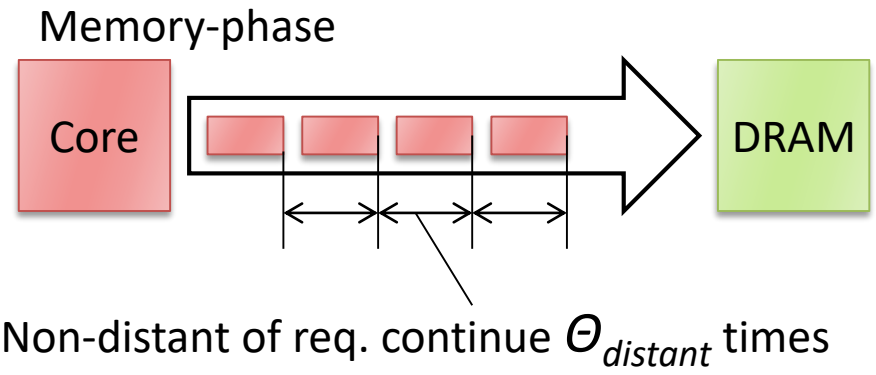
- We think this inaccurate classification is caused by the conventional periodically updating prediction strategy

Contribution 1: Compute-Phase Prediction

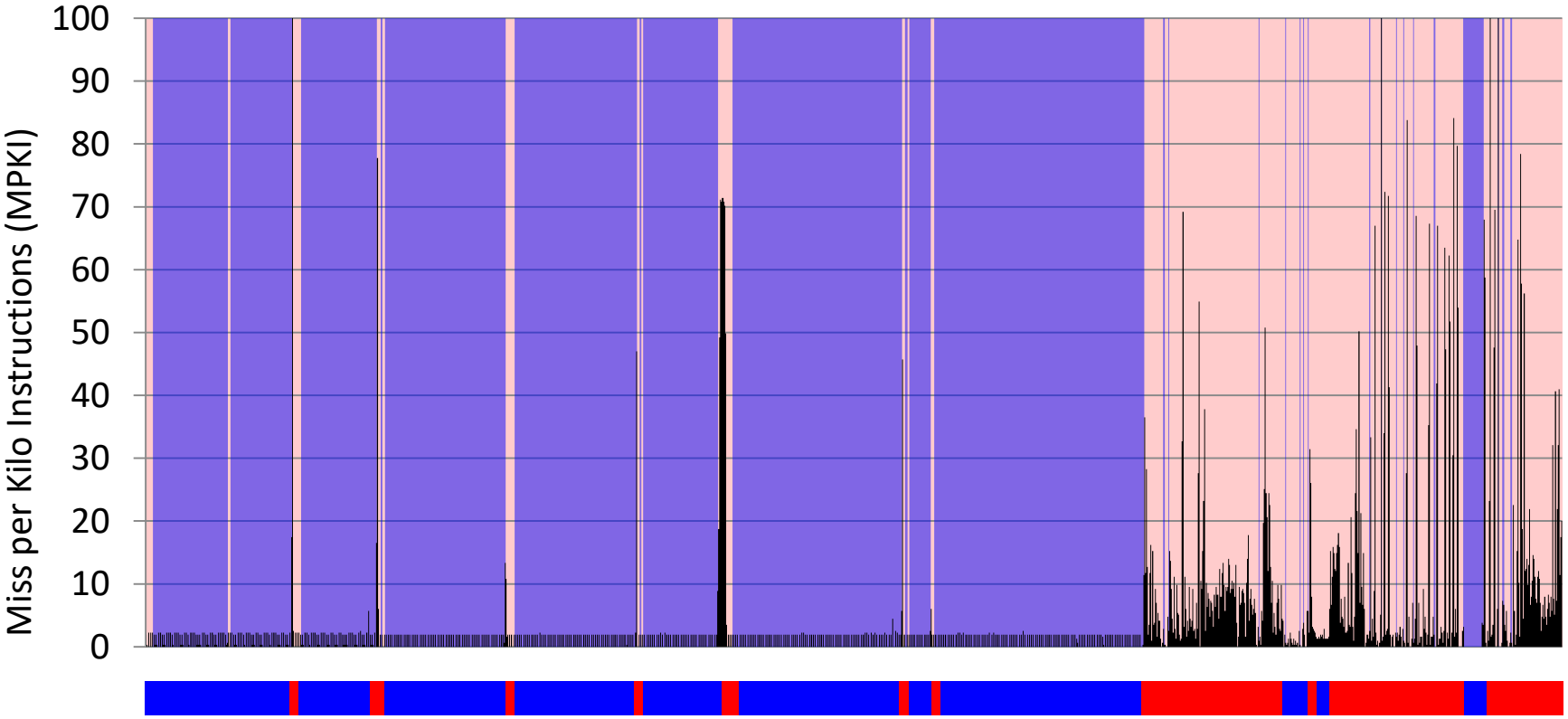
- "Distance-based phase prediction" to realize fine-grain thread priority control scheme







Phase-prediction of Compute-Phase Prediction

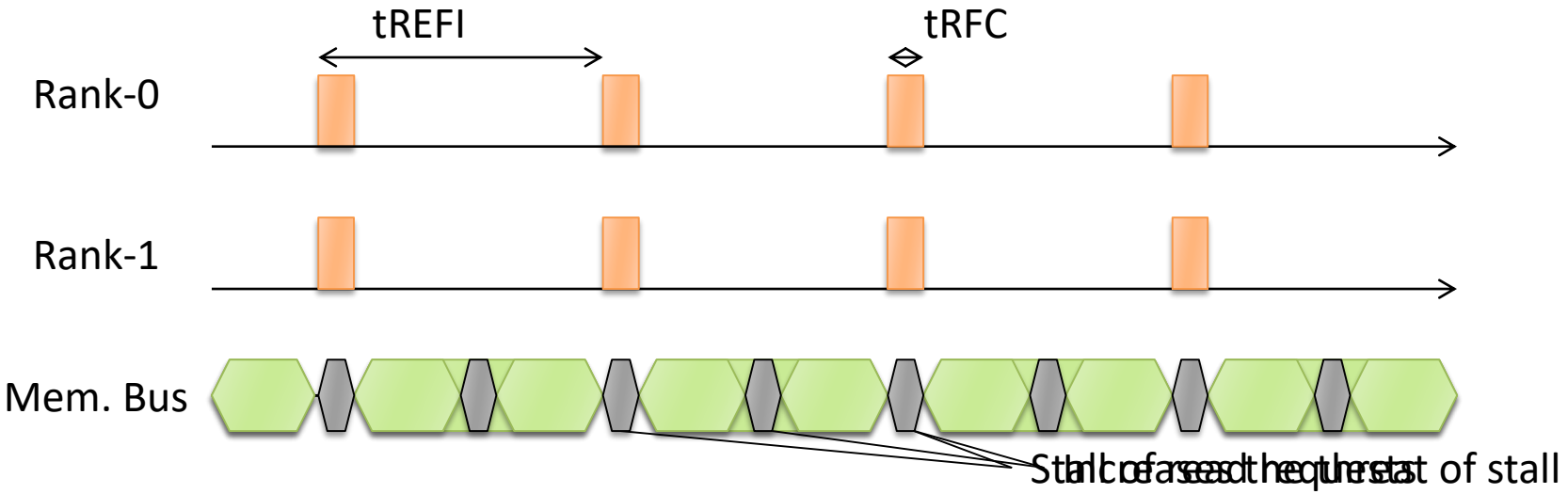


- Prediction result nearly matches the optimal classification
  - Improves fairness and system throughput

Outline

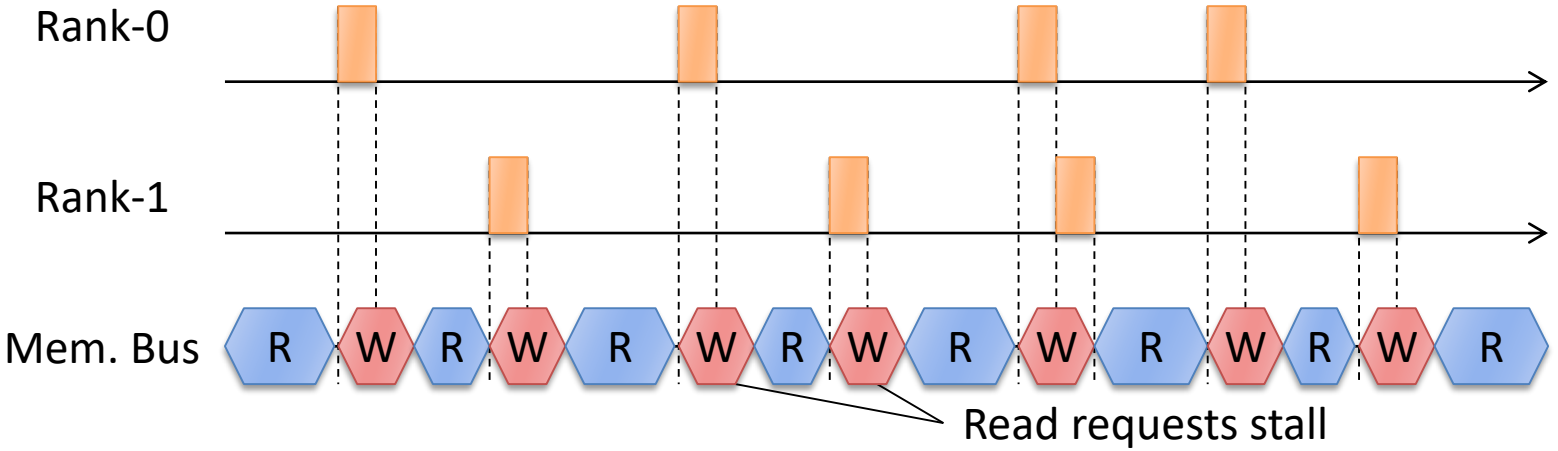
- Proposals
  - Compute-Phase Prediction
    - Thread-priority control technique for multi-core processor
  - Writeback-Refresh Overlap
    - Mitigates refresh penalty on multi-rank memory system
- Optimizations
  - MLP-aware priority control
  - Memory bus reservation
  - Activate throttling

DRAM refreshing penalty



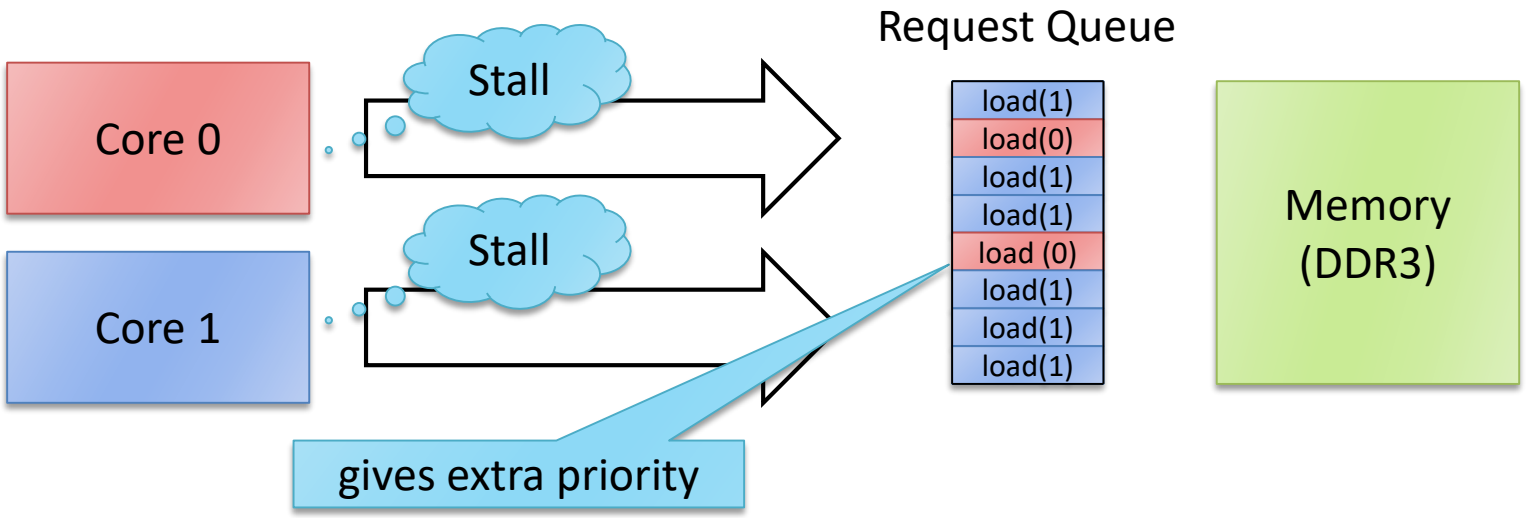
- DRAM refreshing increases the stall time of read requests
  - Stall of read requests increases the execution time
  - Shifting refresh timing cannot reduce the stall time
    - This increases the threat of stall time for read requests

Contribution 2: Writeback-Refresh Overlap



- Typically, modern controllers divide read phases and write phases to reduce bus turnaround penalties
- Overlaps refresh command with the write phase
  - Avoid to increasing the stall time of read requests

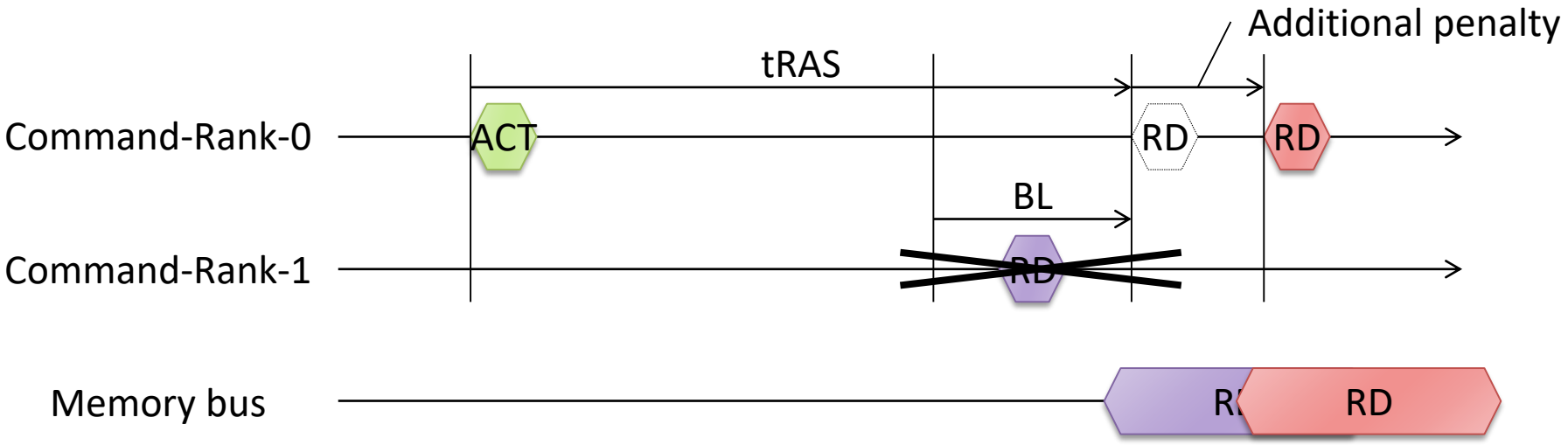
Optimization 1: MLP-Aware Priority Control



- Prioritizes Low-MLP requests to reduce the stall time.
  - This priority is higher than the priority control of compute-phase predictions
  - Minimalist [Kaseridis+ 2011] also uses MLP-aware scheduling

Optimization 2: Memory Bus Reservation

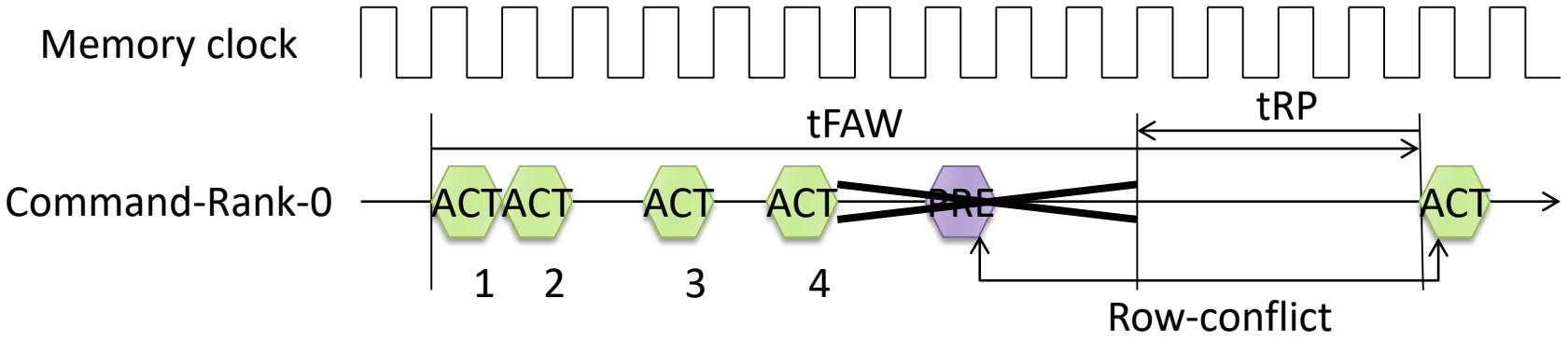
- Reserves HW resources to reduce the latency of critical read requests
  - Data bus for read and write (considering tRTR/tWTR penalty)



- This method improves the system throughput and fairness

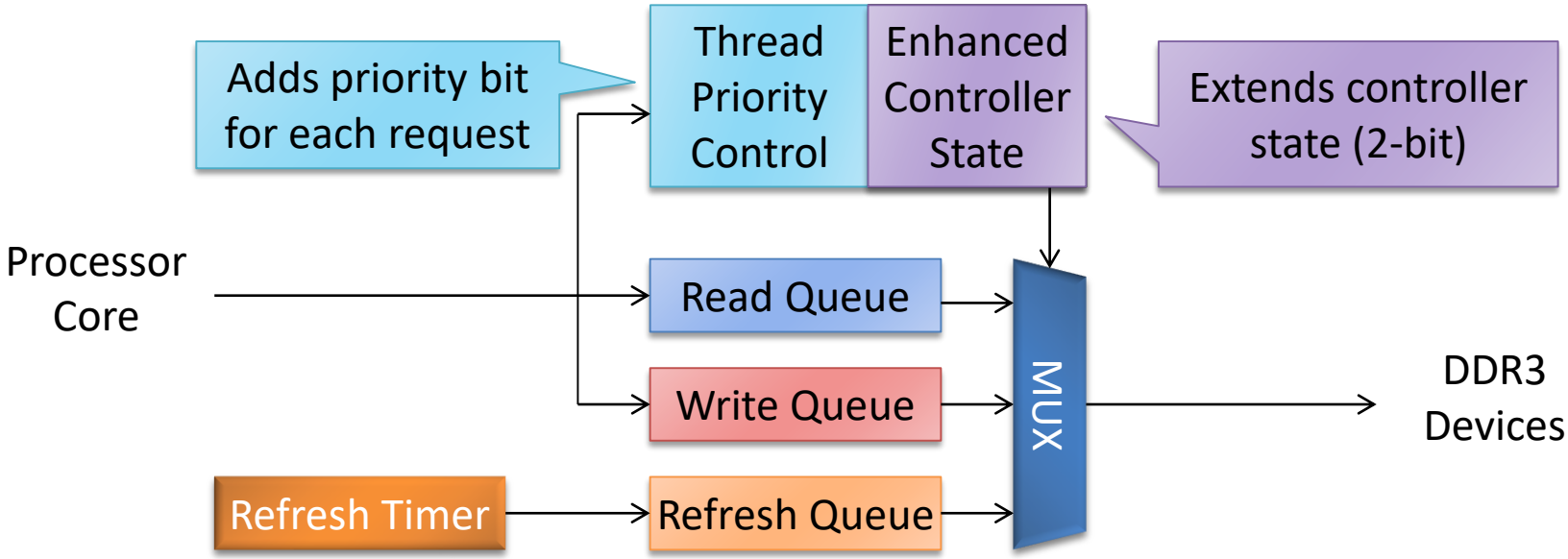
Optimization 3: Activate Throttling

- Controls precharge / activation based on tFAW tracking
  - Too early precharge command does not contribute to the latency reduction of following activate command



- Activate throttling increases the chance of row-hit access

Implementation: Optimized Memory Controller

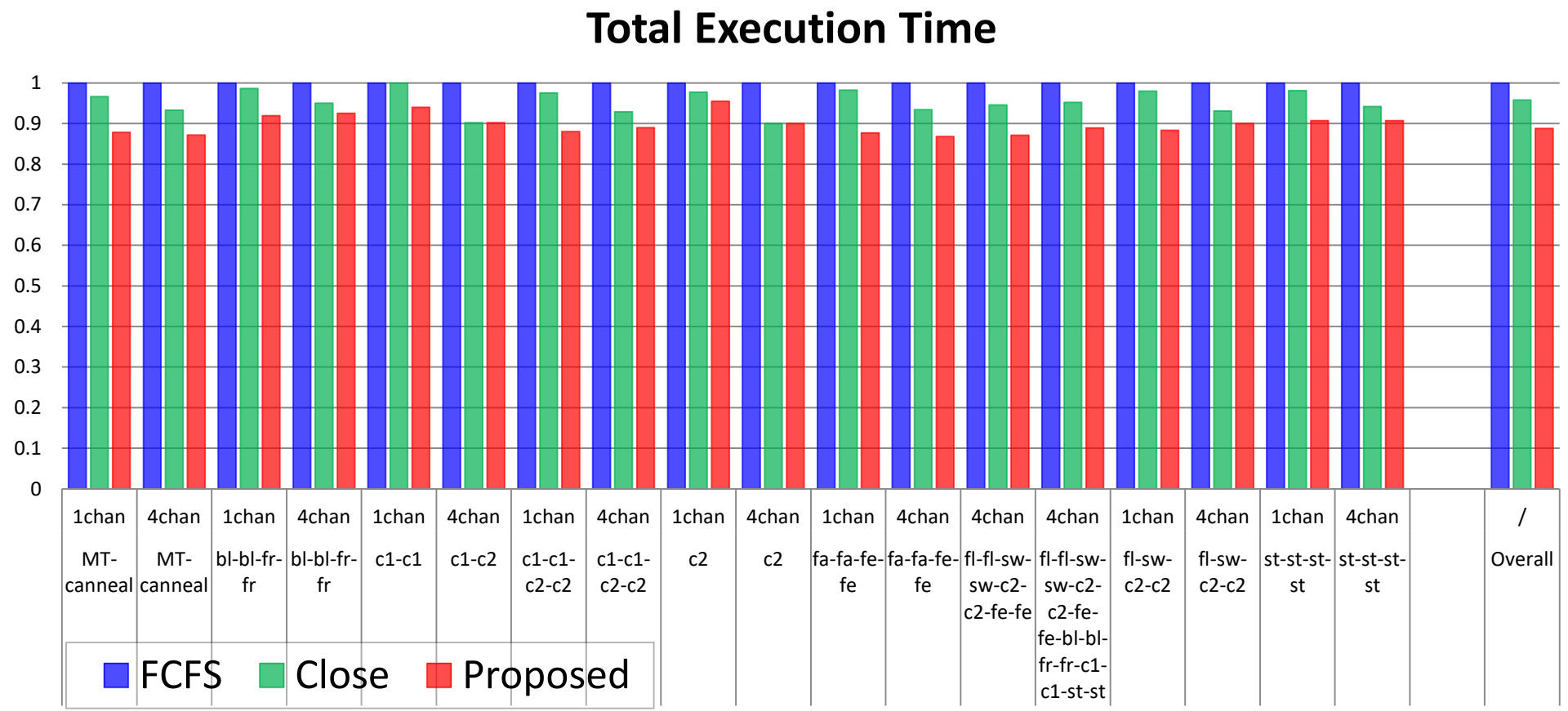


- The optimized controller does not require large HW cost
  - We mainly extend thread-priority control and controller state through our new scheduling technique

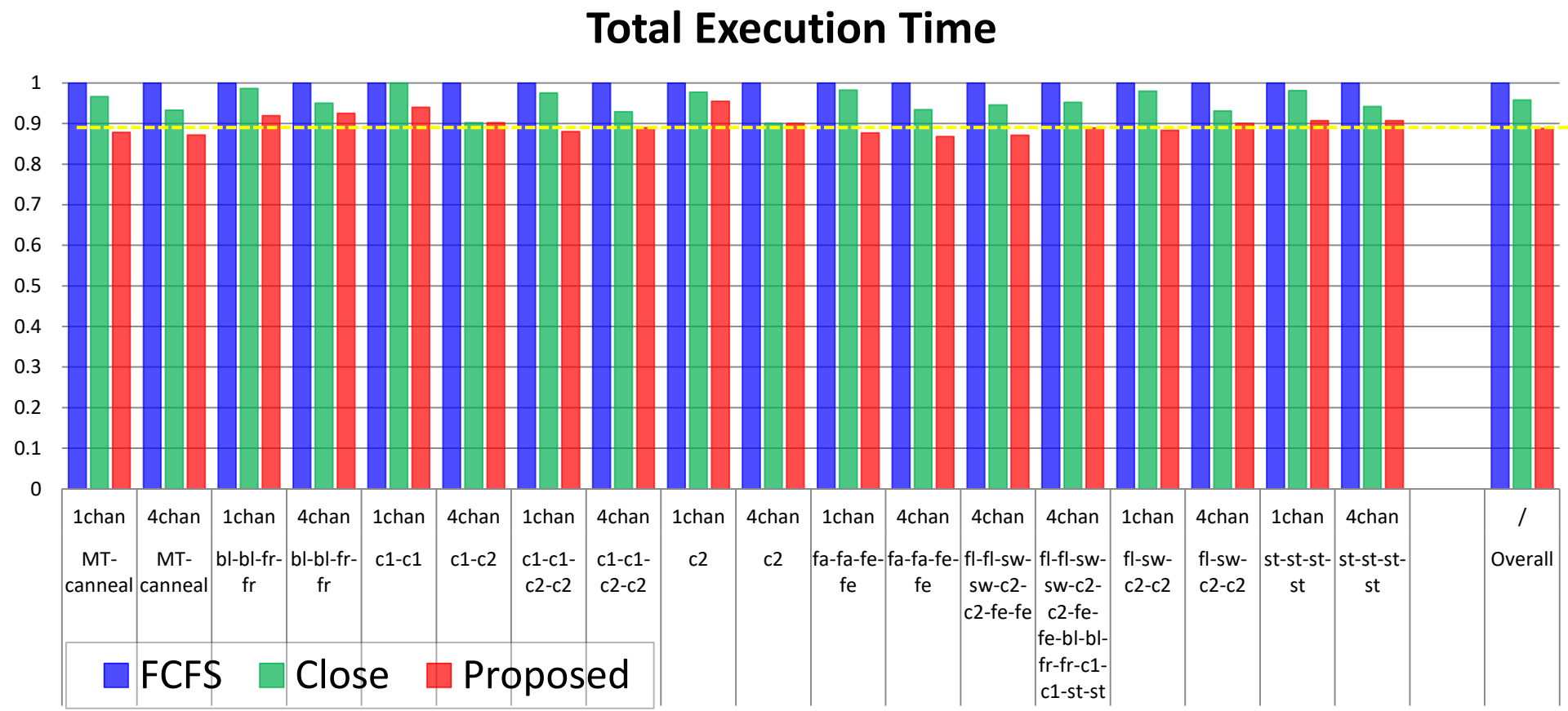
Implementation: Hardware Cost

- Per-channel resource (341.25B)
  - Compute-Phase Prediction (258B)
  - Writeback-Refresh Overlap (2-bit)
  - Other features (83B)
- Per-request resource (3-bit)
  - Priority bit, Row-hit bit, Timeout flag bit
- Overall Hardware Cost: 2649B

Evaluation Results



Evaluation Results



Op

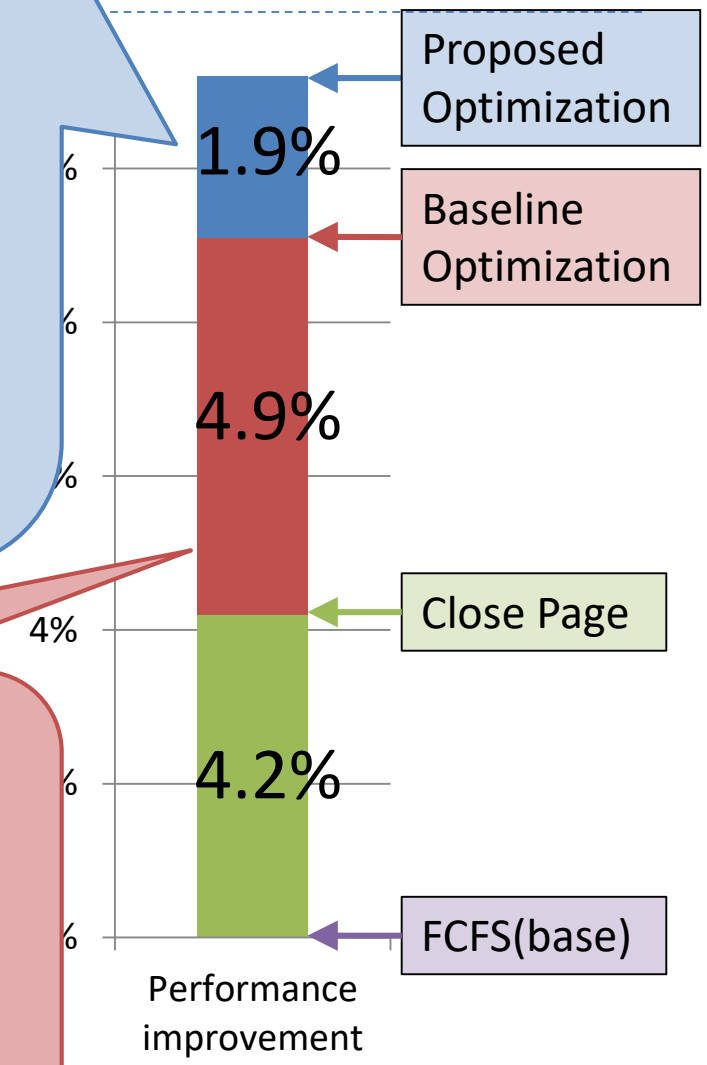
- 11
  fr
  -
  -
  -

Proposals
- Compute-phase prediction
- Writeback-refresh overlap

Optimizations
- MLP-aware priority control
- Memory bus reservation
- Activate throttling

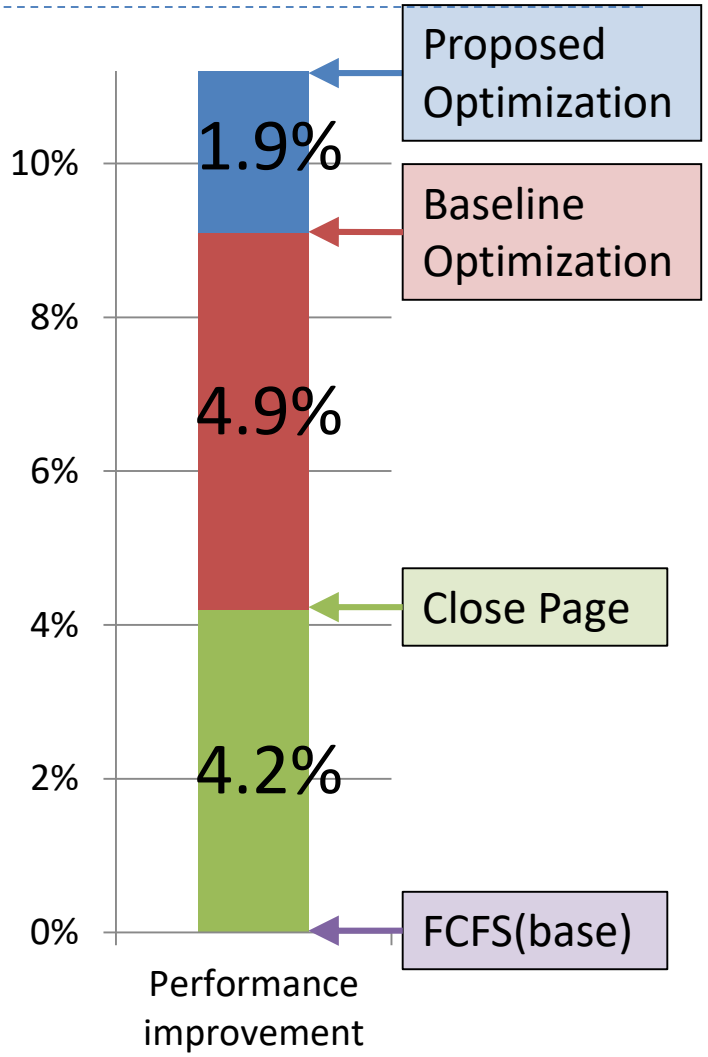
- Baseline optimization accomplish

- Timeout Detection
- Write Queue Spill Prevention
- Auto-Precharge
- Max Activate-Number Restriction



Optimization Breakdown

- 11.2% Performance improvement from FCFS consists of
  - Close Page Policy: 4.2%
  - Baseline Optimization: 4.9%
  - Proposal Optimization: 1.9%
- Baseline optimization accomplishes a 9.1% improvement



Summary of memory access scheduling

- High Performance Memory Access Scheduling
  - Proposals
    - Novel thread-priority control method: Compute-phase prediction
    - Cost-effective refreshing method: Writeback-refresh overlap
  - Optimization strategies
    - MLP-aware priority control, Memory bus reservation, Activate Throttling, Aggressive precharge, force refresh, timeout handling

- The optimized scheduler reduces exec time by 11.2%
  - Several trade-offs between performance and EDP
  - Aggregating the various optimization strategies is most important for the DRAM system efficiency

Access Map Pattern Matching Prefetch: Optimization Friendly Method

Yasuo Ishii[1], Mary Inaba[2], and Kei Hiraki[2]

Background

- Speed gap between processor and memory has been increased
- To hide long memory latency, many techniques have been proposed.
  - ▶Importance of HW data prefetch has been increased
- Many HW prefetchers have been proposed

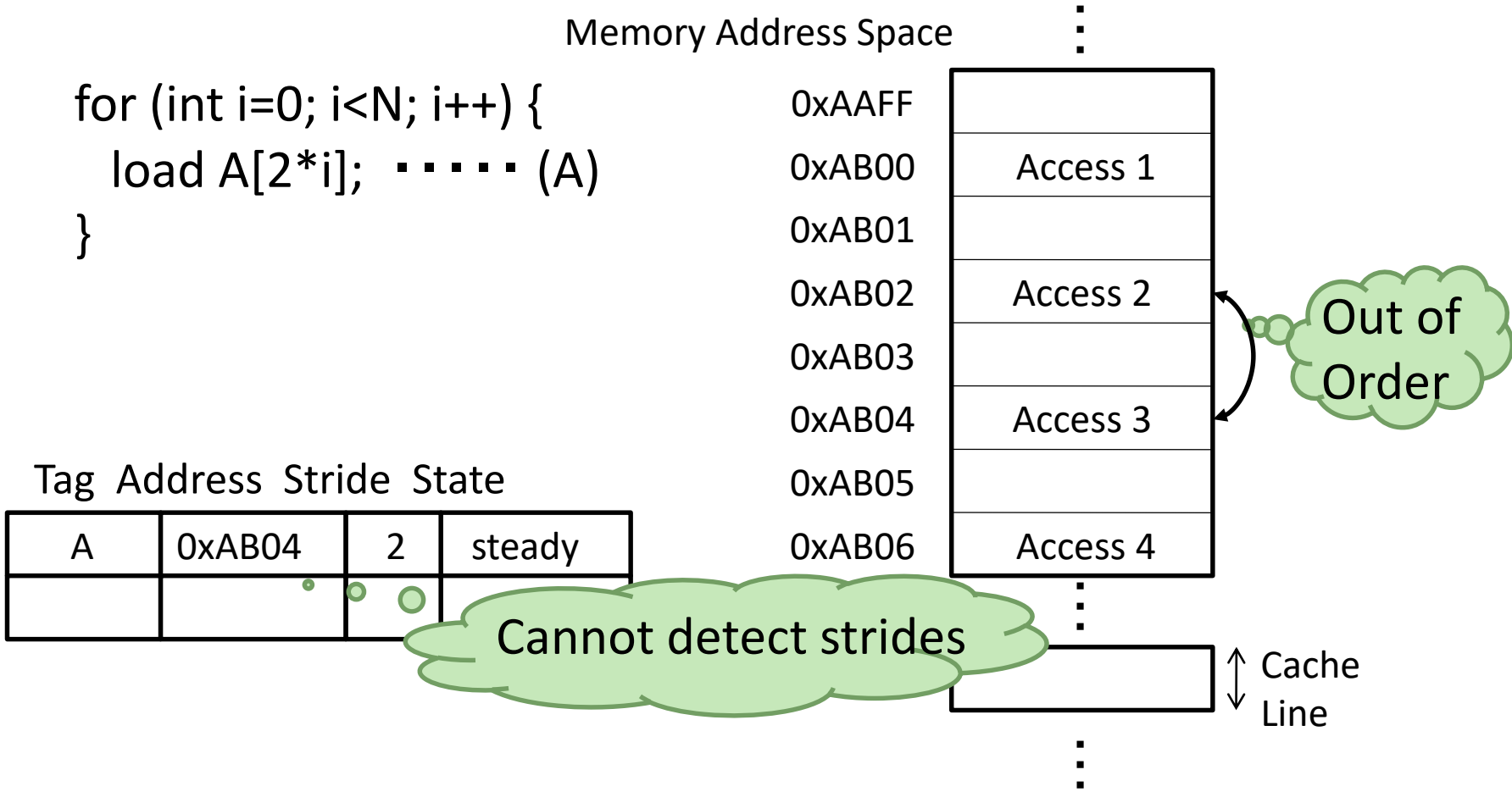
Conventional Methods

- Prefetchers uses
  1. Instruction Address
  2. Memory Access Order
  3. Memory Address

- Optimizations scrambles information
  - Out-of-Order memory access
  - Loop unrolling

Limitation of Stride Prefetch [Chen+95]
Out-of-Order Memory Access

```
for (int i=0; i<N; i++) {
  load A[2*i];  ・・・・・ (A)
}
```



Weakness of Conventional Methods

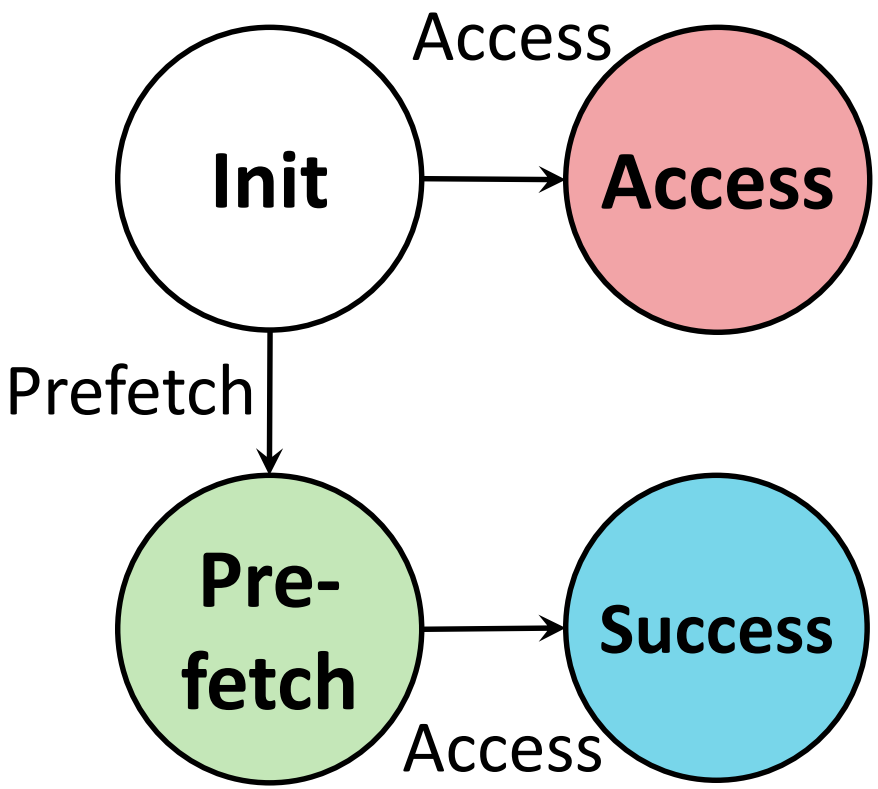
- Out-of-Order Memory Access
  - ▶ Scrambles memory access order
  - ▶ Prefetcher cannot detect address correlations
- Loop-Unrolling
  - ▶ Requires additional table entry
  - ▶ Each entry trained slowly

⇨ Optimization friendly prefetcher is required

Access Map Pattern Matching

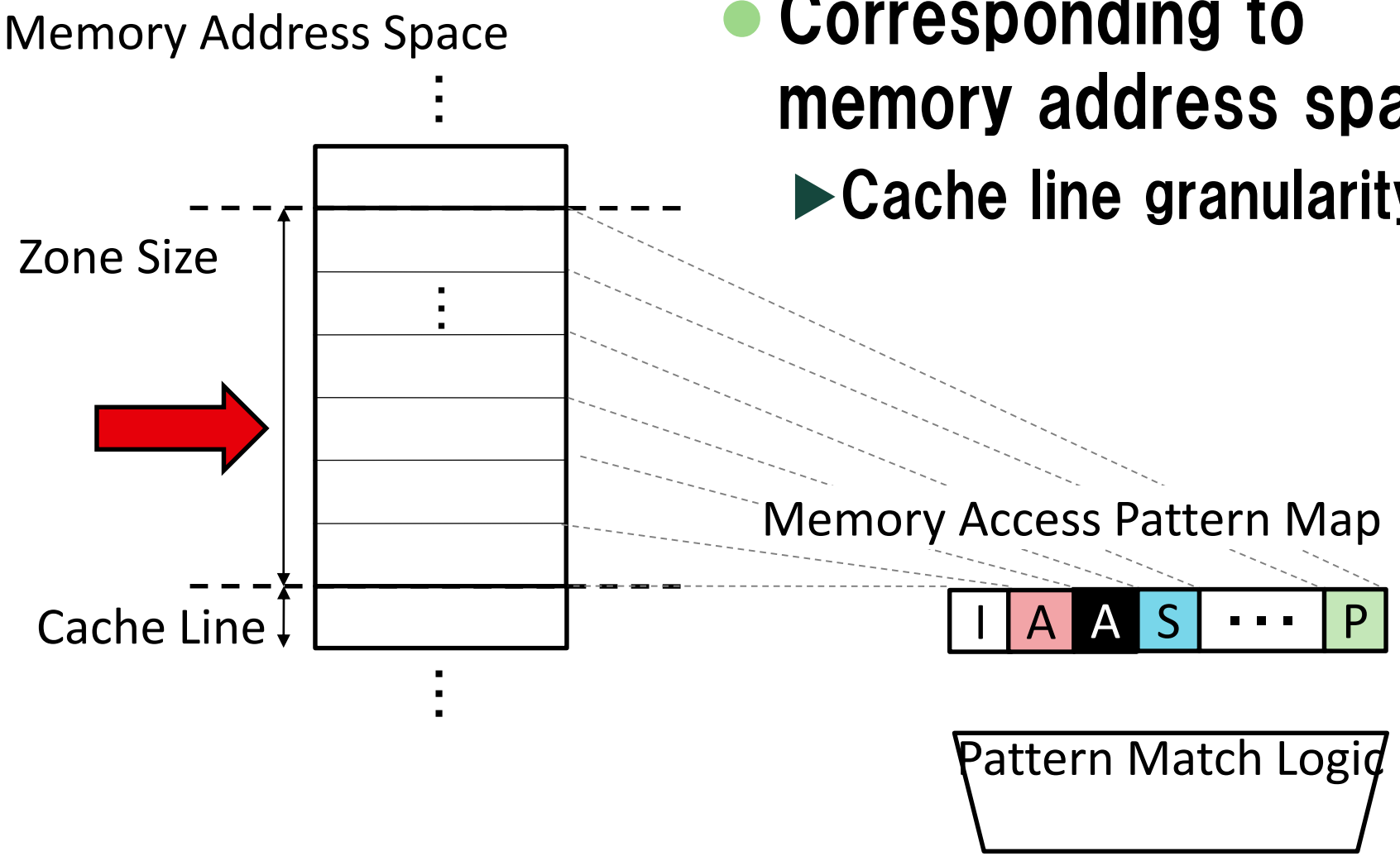
- Pattern Matching
  - ▶Order Free Prefetching
  - ▶Optimization Friendly Prefetch
- Access Map
  - ▶Map-base history
  - ▶2-bit state map
    - ◆Each state is attached to cache block

State Diagram for Each Cache Block



- Init
  - ▶Initialized state
- Access
  - ▶Already accessed
- Prefetch
  - ▶Issued Pref. Requests
- Success
  - ▶Accessed Pref. Data

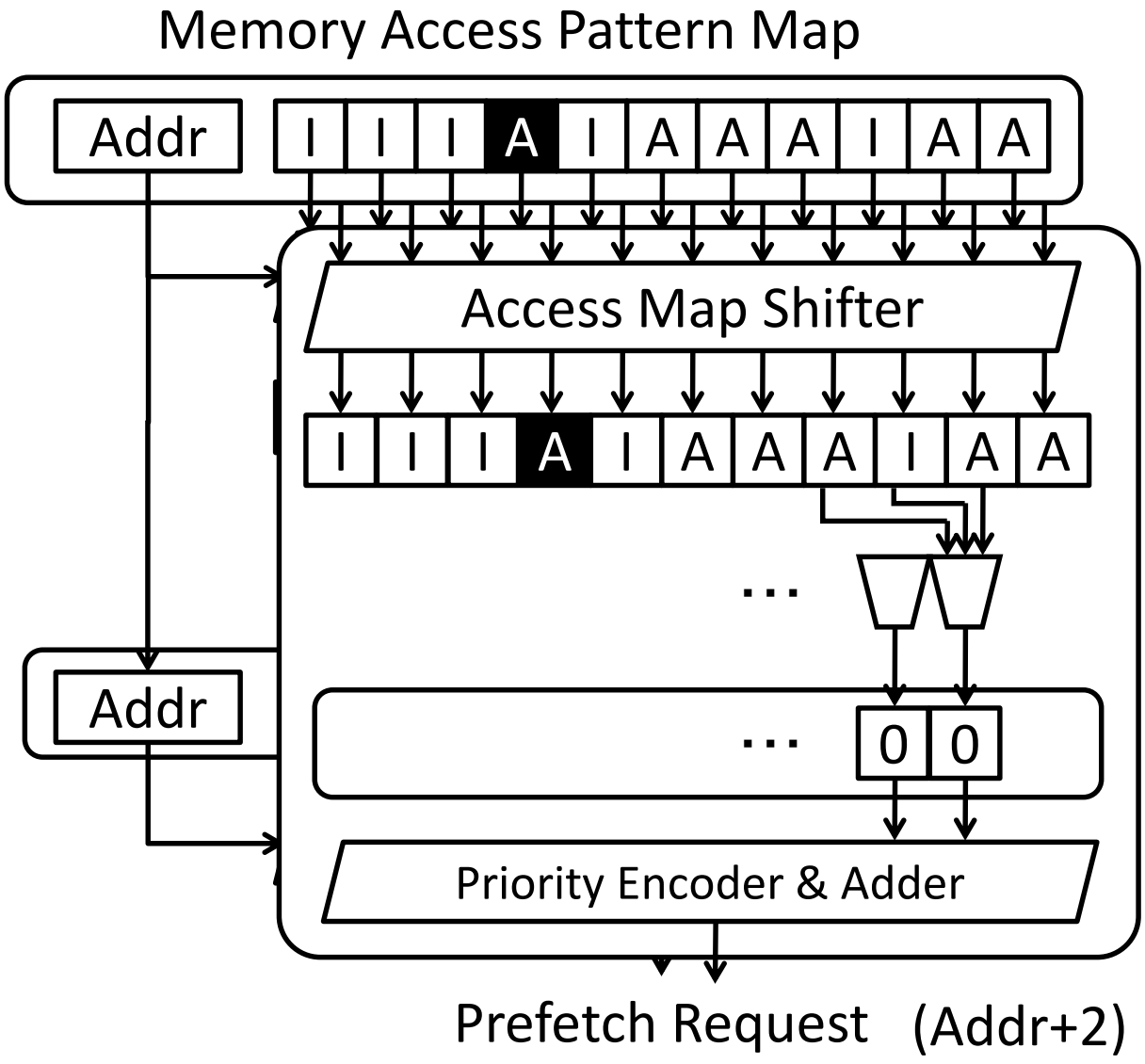
Memory Access Pattern Map



- Corresponding to memory address space
  - ▶Cache line granularity

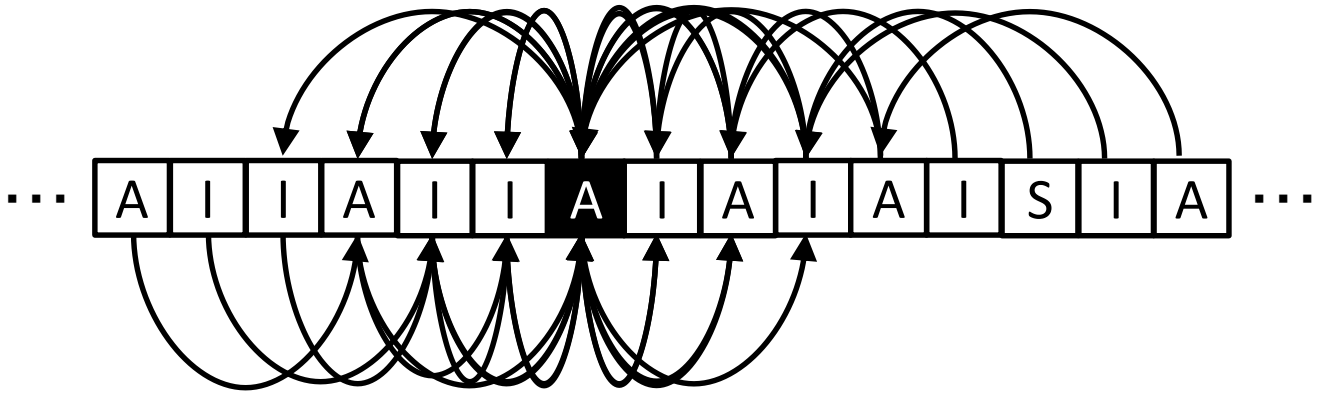
Pattern Matching Logic

- Access Map Shifter
- Pattern Detector
- Pipeline Register
- Prefetch Selector



Parallel Pattern Matching

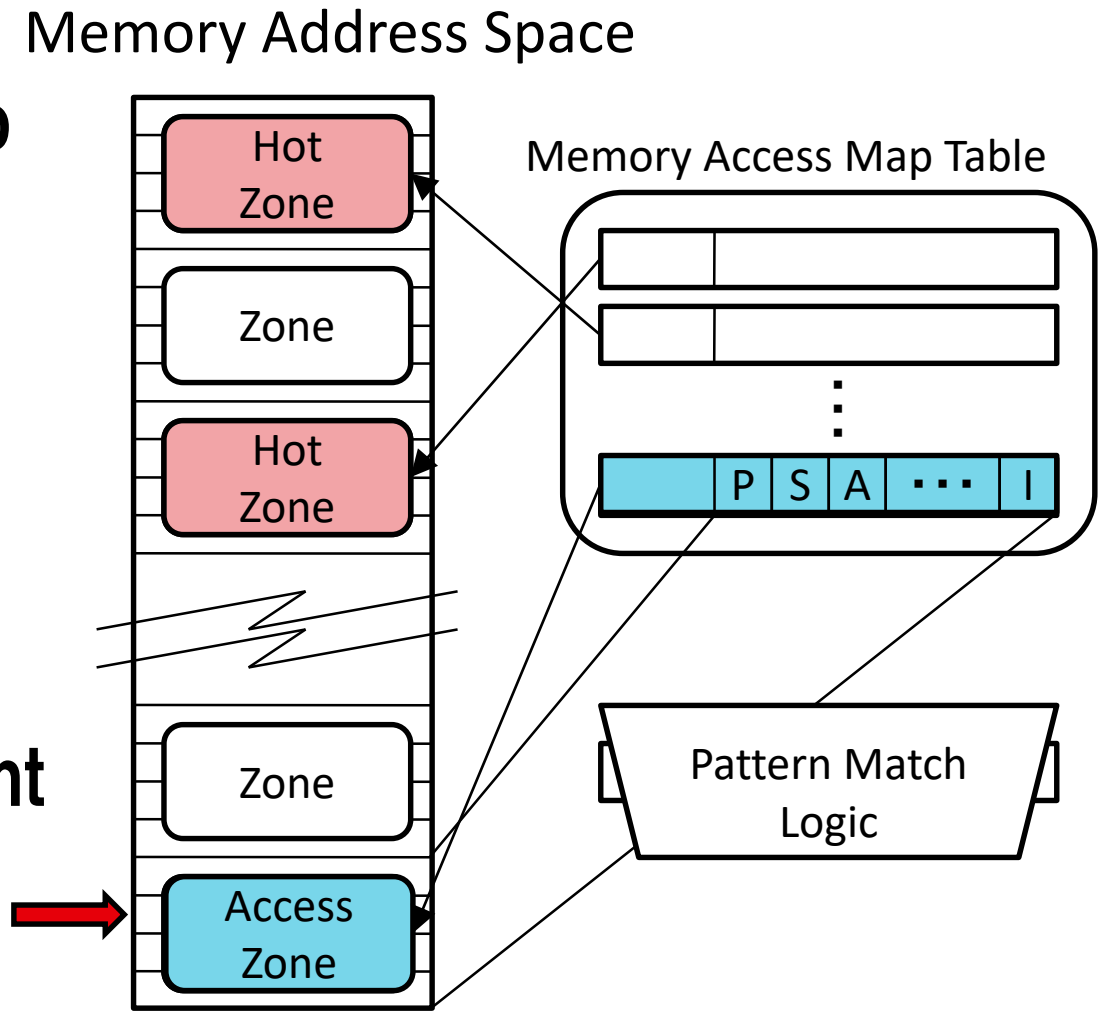
- Detects patterns from memory access map
  - ▶Detects address correlations in parallel
  - ▶Searches candidates effectively



Memory Access Pattern Map

AMPM Prefetch

- Memory address space divides into zone
- Detects hot zone
- Memory Access Map Table
  - ▶LRU replacement
- Pattern Matching



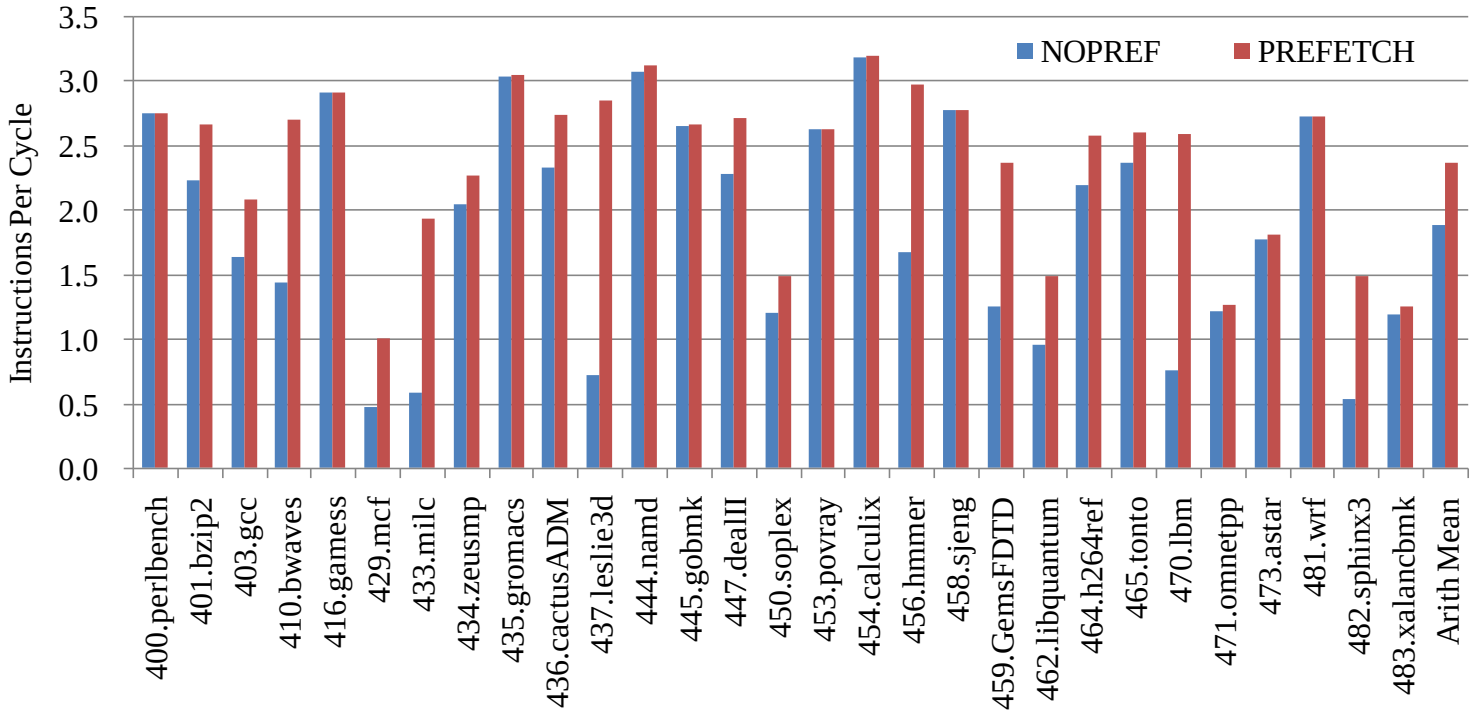
Features of AMPM Prefetcher

- Pattern Matching Base Prefetching
  - ▶Map base history
  - ▶Optimization friendly prefetching
- Parallel pattern matching
  - ▶Searches candidates effectively
  - ▶Complexity-effective implementation

Methodology

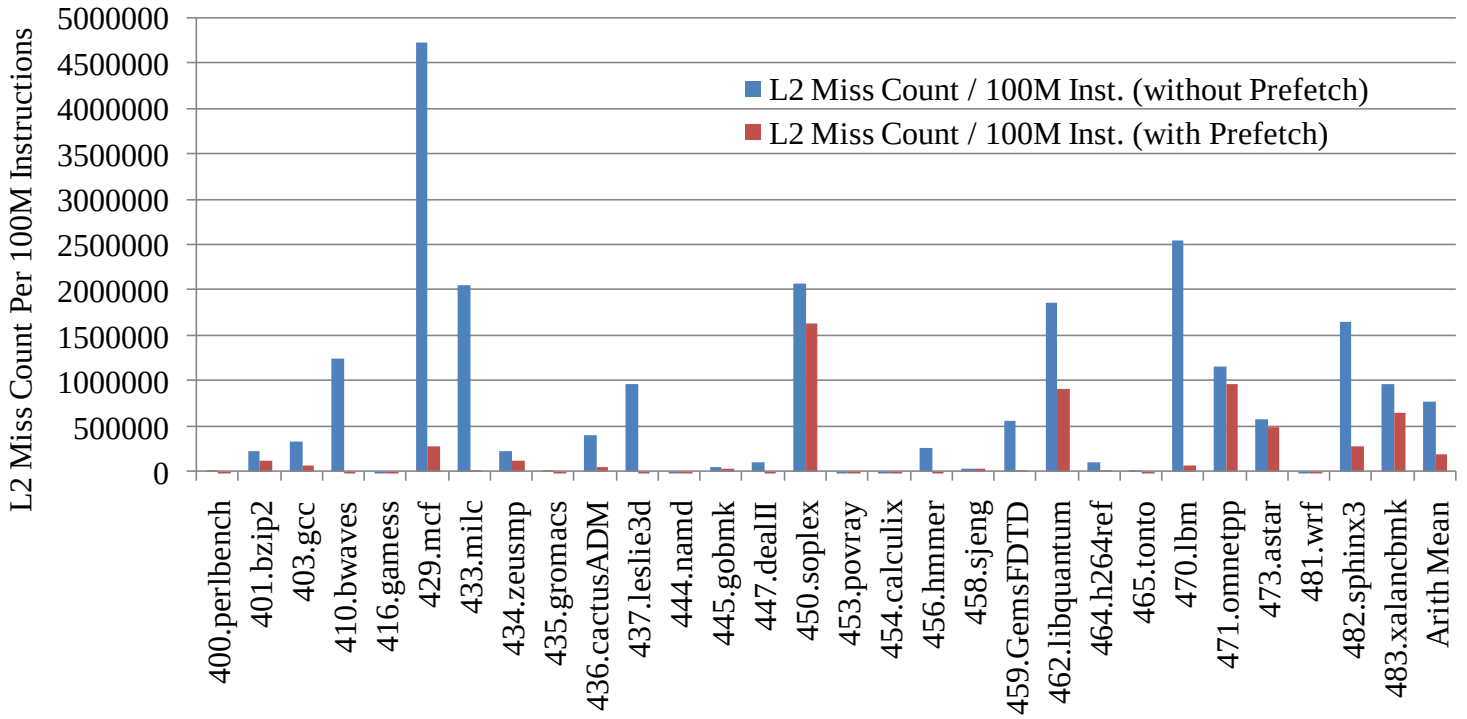
- Simulation Environment
  - ▶DPC Framework
  - ▶Skips first 4000M instructions and evaluate following 100M instructions
- Benchmark
  - ▶SPEC CPU2006 benchmark suite
  - ▶Compile Option: “-O3 -fomit-frame-pointer -funroll-all-loops”

IPC Measurement



- Improves performance by 53%
- Improves performance in all benchmarks

L2 Cache Miss Count

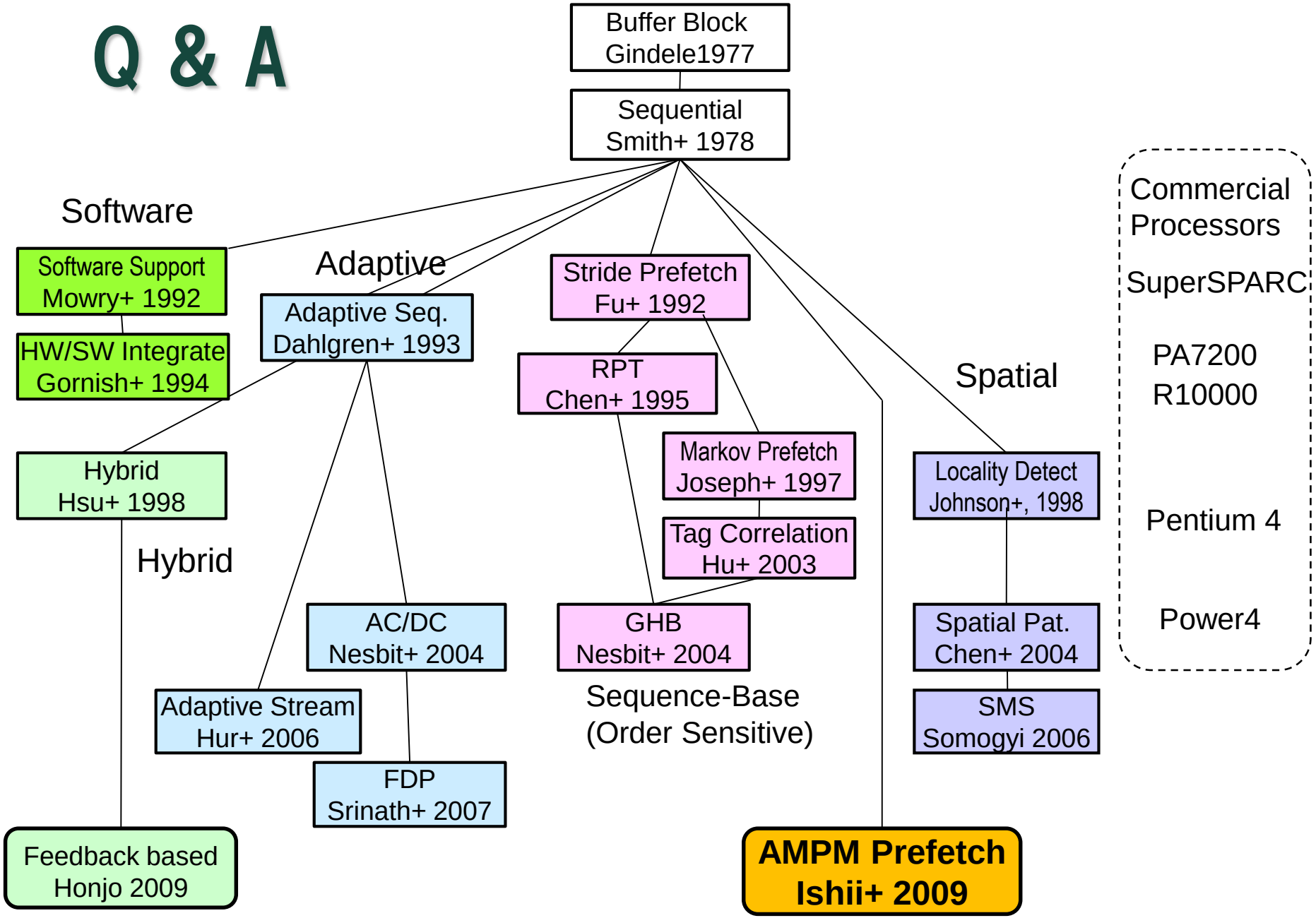


- Reduces L2 Cache Miss by 76%

Summary of prefetching

- Access Map Pattern Matching Prefetch
  - ▶Order-Free Prefetch
    - Optimization friendly prefetching
  - ▶Parallel Pattern Matching
    - Complexity-effective implementation
- Optimized AMPM realizes good performance
  - ▶Improves IPC by 53%
  - ▶Reduces L2 cache miss by 76%

Q & A


Buffer Block
Gindele1977
Sequential
Smith+ 1978
Software
Software Support
Mowry+ 1992
HW/SW Integrate
Gornish+ 1994
Adaptive
Adaptive Seq.
Dahlgren+ 1993
Stride Prefetch
Fu+ 1992
RPT
Chen+ 1995
Markov Prefetch
Joseph+ 1997
Tag Correlation
Hu+ 2003
Spatial
Locality Detect
Johnson+, 1998
Hybrid
Hsu+ 1998
Hybrid
AC/DC
Nesbit+ 2004
Adaptive Stream
Hur+ 2006
FDP
Srinath+ 2007
GHB
Nesbit+ 2004
Sequence-Base
(Order Sensitive)
Spatial Pat.
Chen+ 2004
SMS
Somogyi 2006
Feedback based
Honjo 2009
AMPM Prefetch
Ishii+ 2009
Commercial
Processors
SuperSPARC
PA7200
R10000
Pentium 4
Power4

Summary

1. Speculation is the most important tool to speed-up a single core processor

2. Our target in the next 10 years is more than 20 instructions / cycle

3. Next target would be

   Prediction of NoC data injection

   Prefetching for gather/scatter operation

   Practical value prediction

Questions