# Revisiting High-Speed Forwarding: Cases for State-full forwarding and Learned Index based Forwarding

1st International Workshop on Theory and Practice of Programmable Forwarding

June 28, 2021

Toru Hasegawa
Osaka Univierity

大阪大学
OSAKA UNIVERSITY

# INTRODUCTION

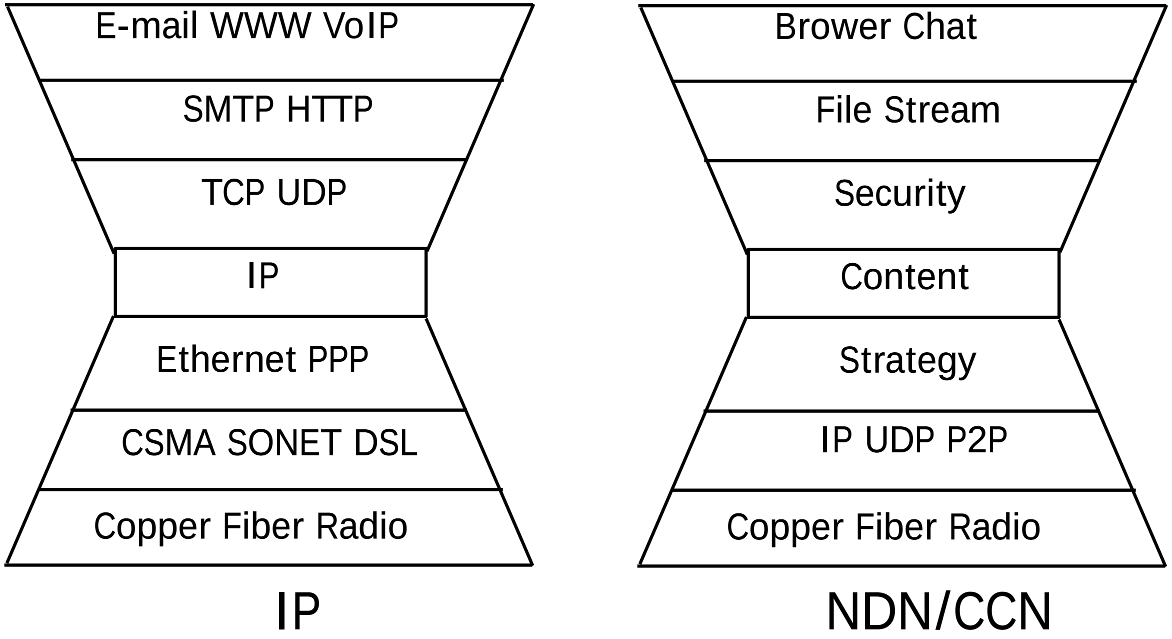
# Introduction

- High speed packet forwarding is an important research issue for the new network architecture: Named Data Networking (NDN)
- NDN is different from IPv4 in terms of
  - Hierarchical human readable names like URL
  - A huge number of name prefixes is not recorded at SRAM devices
  - Request-response style communications require routers to record states called Pending Interest Table (PIT) entries
  - In-network caching is supported by routers
- Toward 100 Gbps and 10 Tbps NDN packet forwarding
  - Survey of the studies
  - Achievements at Hasegawa Lab., Osaka University

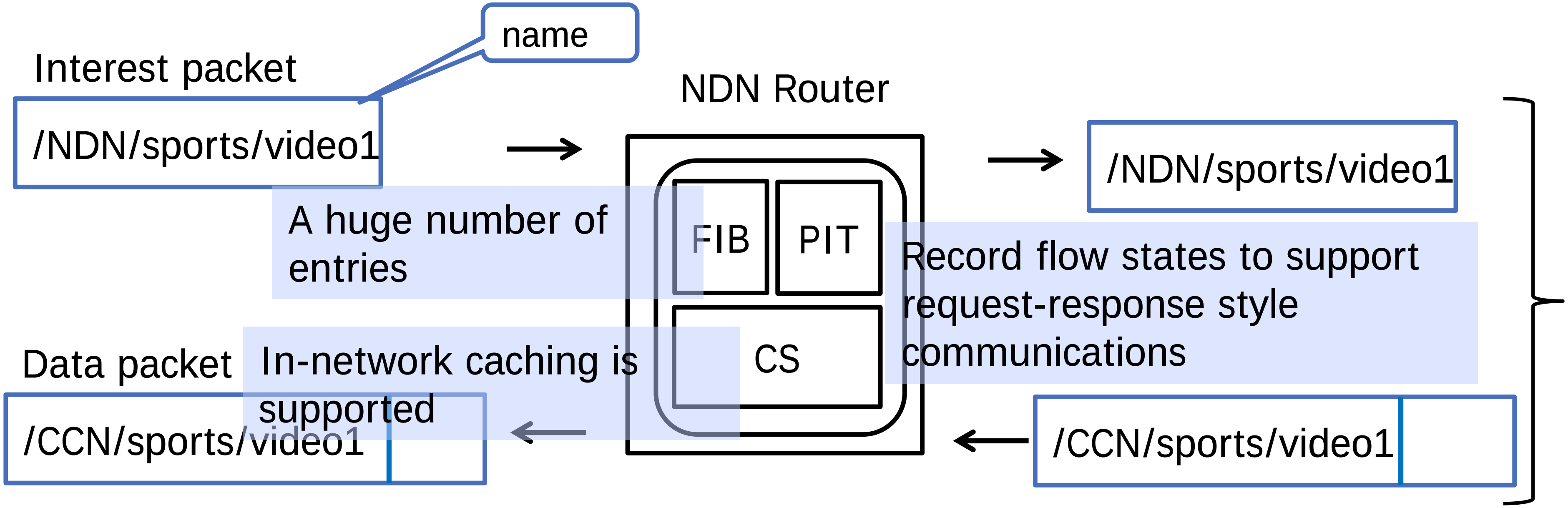
# NAMED DATA NETWORKING

# Named Data Networking (NDN) [1]

- Name-based Routing/Forwarding
  - Content name rather than the host address
    - Network Name ≒ Application Name
    - Each chunk (packet) has a name
  - Hierarchical human-readable names like URL
    - /jp/ac/osaka-u/ist/www-hasegawa



# Interest/Data Packet Forwarding



name

Interest packet

/NDN/sports/video1

NDN Router

FIB

PIT

CS

/NDN/sports/video1

A huge number of entries

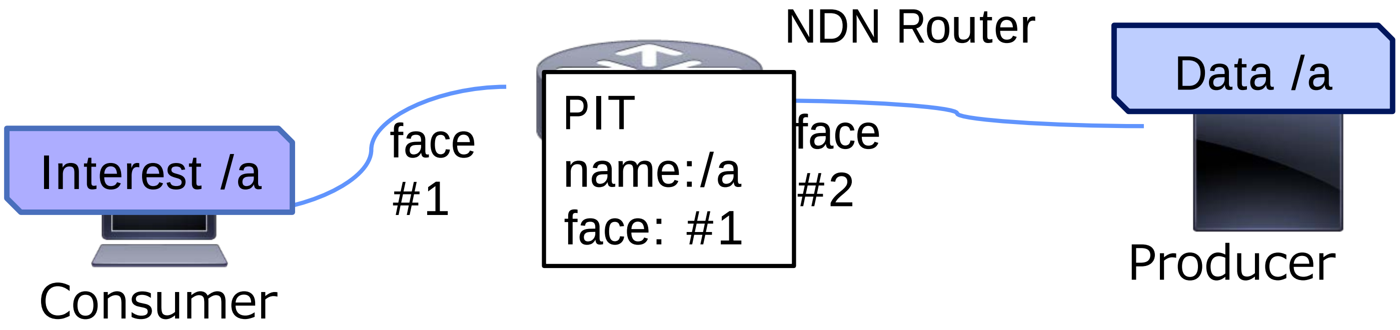
Record flow states to support request-response style communications

Data packet

In-network caching is supported

/CCN/sports/video1

/CCN/sports/video1



NDN Router

Interest /a

face #1

PIT
name:/a
face: #1

face #2

Data /a

Consumer

Producer

# NDN vs IP Packet Forwarding

LPM

- Huge number of name prefixes/names
  - Name prefixes : $2.1*10^8$ >> IP prefixes: $7*10^5$
- Variable size names/name prefixes
  - NDN: Average component length and number, 28 bytes [2]
    - Average component length: 7 bytes
    - Average component number: 4
  - IP: 32 bits/128 bits, fixed length

Flow state integrity

Load balancing

- Stateful forwarding:
  - Flow states should be maintained to support request-response style communication (Interest and Data packets)[3]
  - Huge number of flows' states: More than $10^6$ PIT entries at the core router [4]
- Packet-level in-network caching

# Reality Check [5]

- A large FIB is not stored at fast memory devices like SRAM devices
- 10 Tbps and 100 Gbps are feasible targets of forwarding speeds for hardware and software NDN routers, respectively

15M packet/s * 1K bytes = 120 Gbps
1G packet/s * 1K bytes = 8 T bps

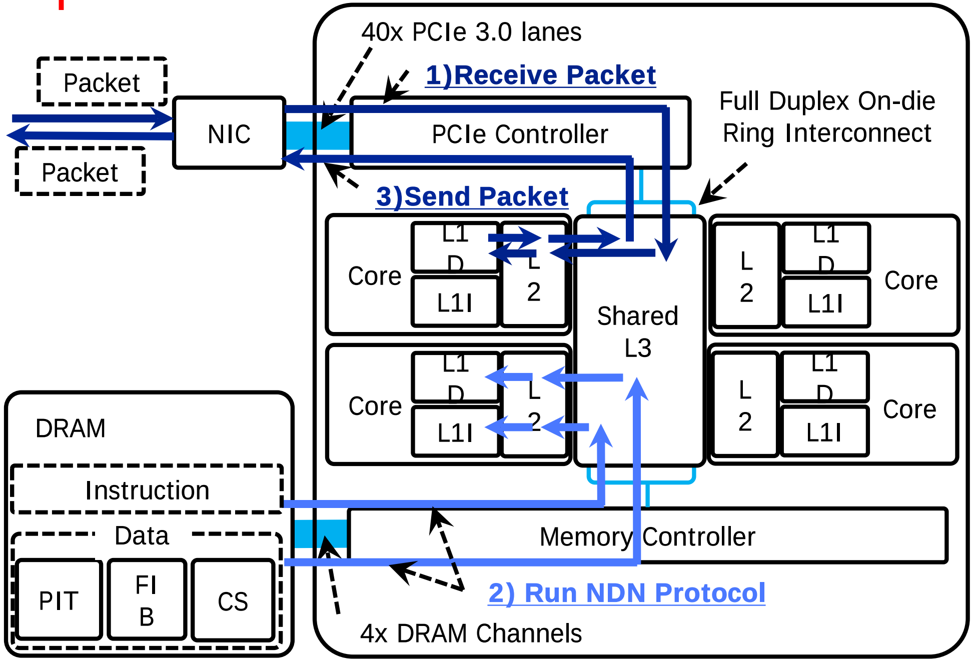
| | Objective | FIB | PIT | CS |
|---|---|---|---|---|
| Edge | **20M** prefixes<br>**15M** packet/s (Interest)<br>0.417Mpacket/s (Data Caching) | On-Chip SAM (200M bits)<br>Bloom Filter Off-Chip (140M bytes)<br>**Hash table** | RLDRAM (70M bits)<br>**Hash table** | DRAM(6G byte)<br>**Index table**<br>SSD (1T bytes)<br>**Packet store** |
| Core | **250M** prefixes<br>**1G** packet/s (Interest)<br>120Mpacket/s (Data Caching) | On-Chip SRAM (4G bits)<br>Bloom Filter Off-Chip (1.5G bytes)<br>**Hash table** | SRAM(266M bits)<br>**Hash table at each line card** | RLDRAM(266M bits):<br>**Index table**<br>DRAM (10G bytes) |

Cited from [5]

# TOWARD 100 GBPS FORWARDING: SOFTWARE NDN ROUTERS

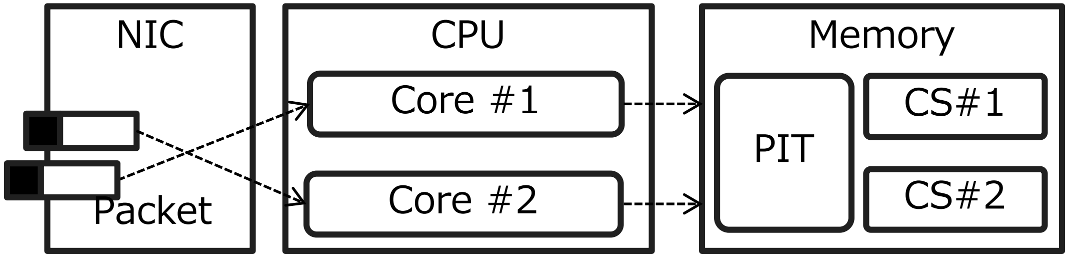
# Software Router :Toward 100 Gbps forwarding

- Goal: 100 Gbps packet forwarding at commodity PC servers with network interface cards (NICs)[2,6-11]
- System model:
  - PC Server: multi-core CPU device and NIC with DPDK
- Design rational
  - Exploits parallelism of multiple CPU cores
  - DPDK eliminates context switches
  - NIC's Receive Side Scaling (RSS) so that packets are equally dispatched to cores



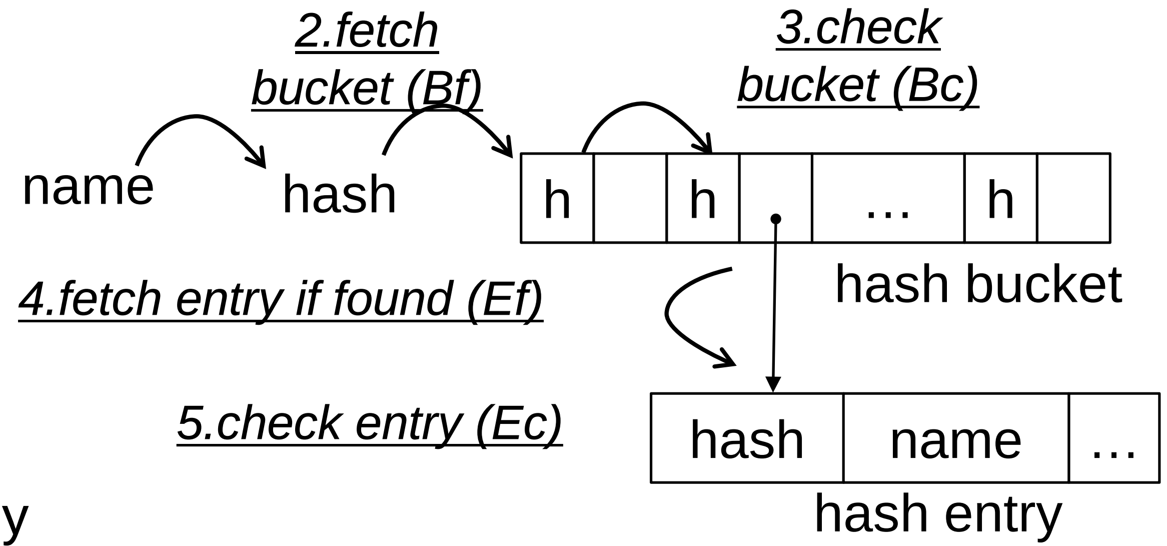
# Approaches to LPM and Load Balancingl [6]

- Hash table-based FIB is selected to compensate for late DRAM access latency
  - The whole FIB cannot be recorded at high speed memory devices like SRAM
- Sharding according to font end cache[12] to mitigate load imbalance due to heavy hitters
  - Sharding should be used: Interest/Data packets of the same name should be handled by the same CPU core in order to avoid using mutual exclusion of the same PIT entry
  - Even if sharding is used, packets are unequally assigned CPU cores due to heavy hitters



# LPM (Longest Prefix Matching) :Hash Table-based FIB [2,6]

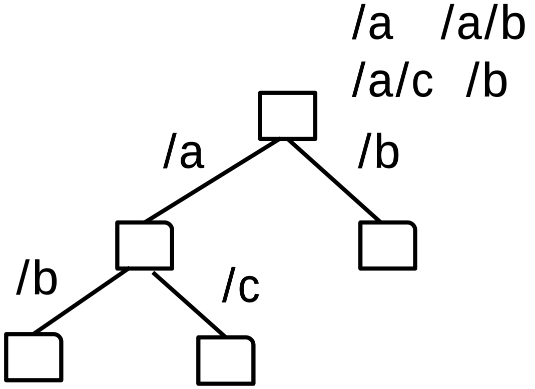
- Hash tables
  - Chained hash table



- Hash table lookup
  - 1) Fetching a hash bucket by using a hash value of a key
  - 2) Searching a pointer to a hash entry by traversing the bucket
  - 3) Fetching the hash entry if the bucket has a pointer to the entry
- LPM: Hash table lookup is repeatedly by the match
  - name /a/b/c/d
    hash(/a/b/c/d)
    hash(/a/b/c)
    …

# LPM: Bloom Filter, Trie-base

- Bloom filter-based FIB[7,8]
  - 1st: Decide the matched prefix length using the bloom filter
  - 2nd :Exact of matching of name with the decided length using the hash table = similar to exact matching of the hash table
- Trie FIB
  - Binary Patricia tree is the most compact among bit, character and component-based trie [9]
  - Tree search



Component-based Patricia

# Dependent DRAM accesses for LPM [6]

- Assumption: Some part of FIB is recorded at DRAM, and thus hiding DRAM accesses by prereftch is important
- Dependent DRAM accesses number (defined by [6])
  - Dependent data pieces should be fetched in sequence from DRM
    - Access latency of fetching the dependent piece cannot be hidden by prefetch
  - Dependent accesses should be as small as possible to hide DRAM access latency

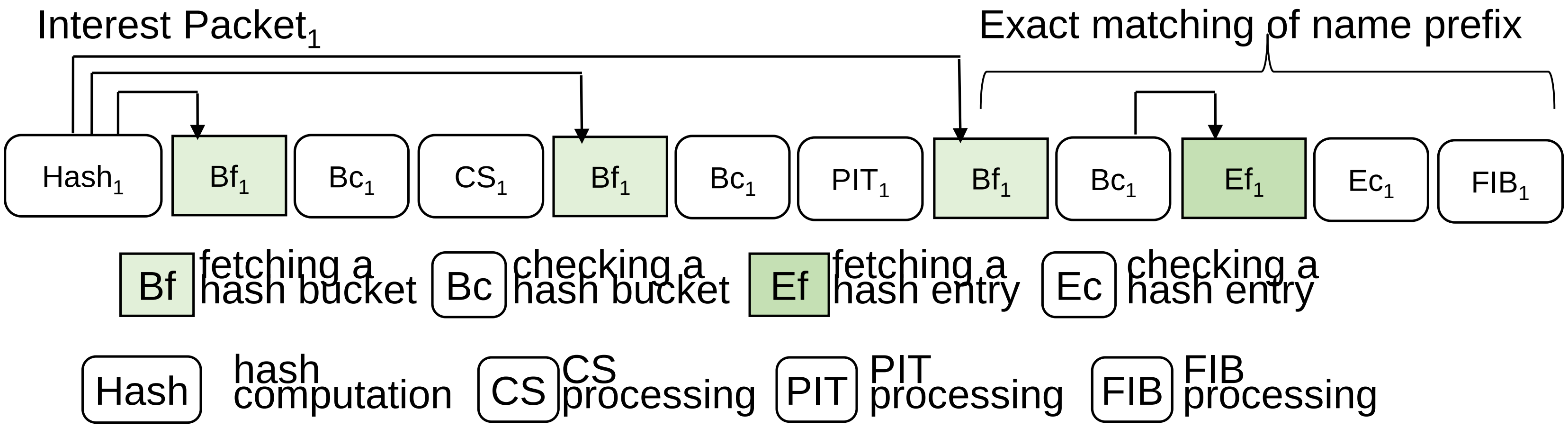
Two data pieces are dependent

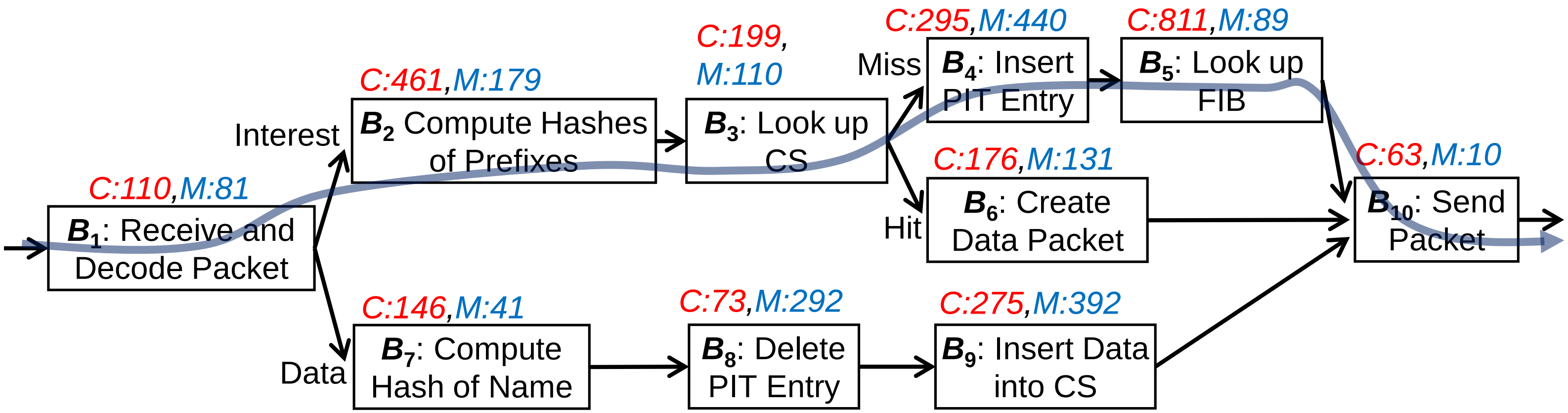
- LPM at hash-table FIB requires the smallest dependent DRAM access number:
  - Hash table-based FIB[2,6] : 2
  - Bloom filter-based FIB[7] : 2
  - Trie FIB[8] : 7 (average)

We choose hash table-based FIB

# Naïve Packet Processing : Waiting time due to hash bucket and entry fetches

- Waiting time to fetch hash buckets and hash entries must be hidden

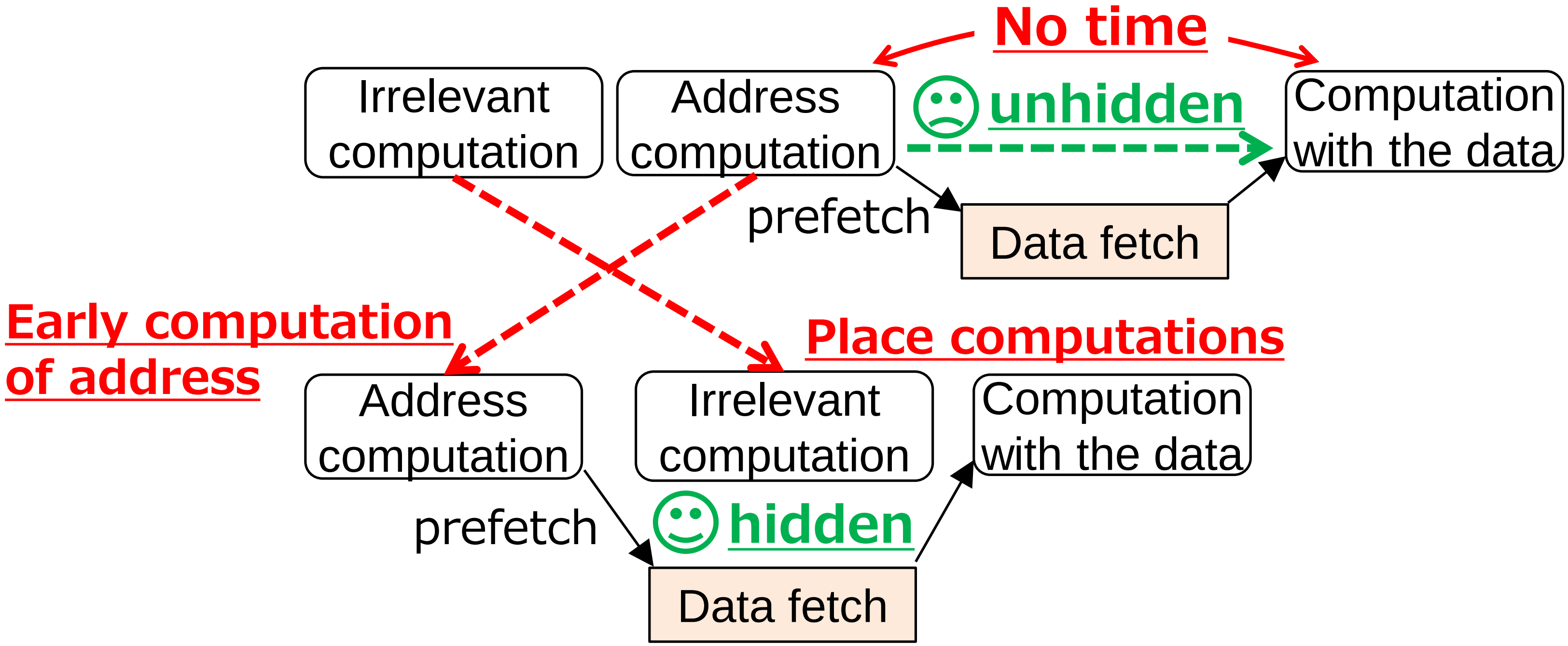




C : CPU Cycles [cycles], M : DRAM Access [bytes]

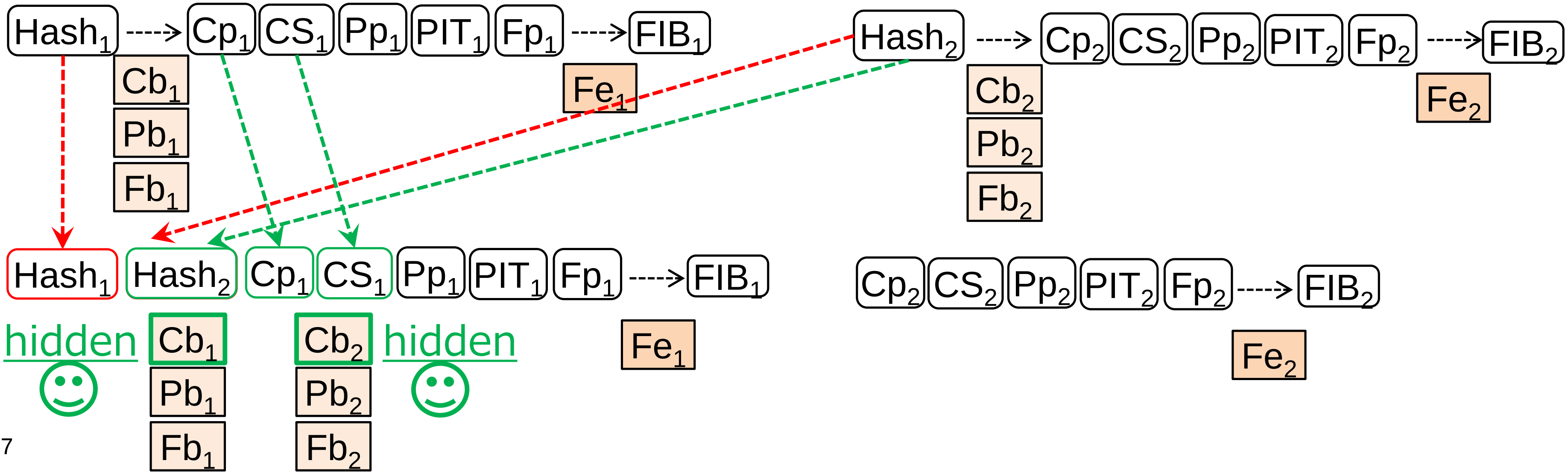
# Hiding DRAM Access Latency with Prefetch

- Issue: Unhidden data fetches in the naïve processing
  - There is no time between computations of addresses to data to be fetched and its uses
- Design guidelines of packet processing
  - Early computations of addresses to data to be fetched
  - Put some computations irrelevant to the fetched data between the address computation and uses of the fetched data



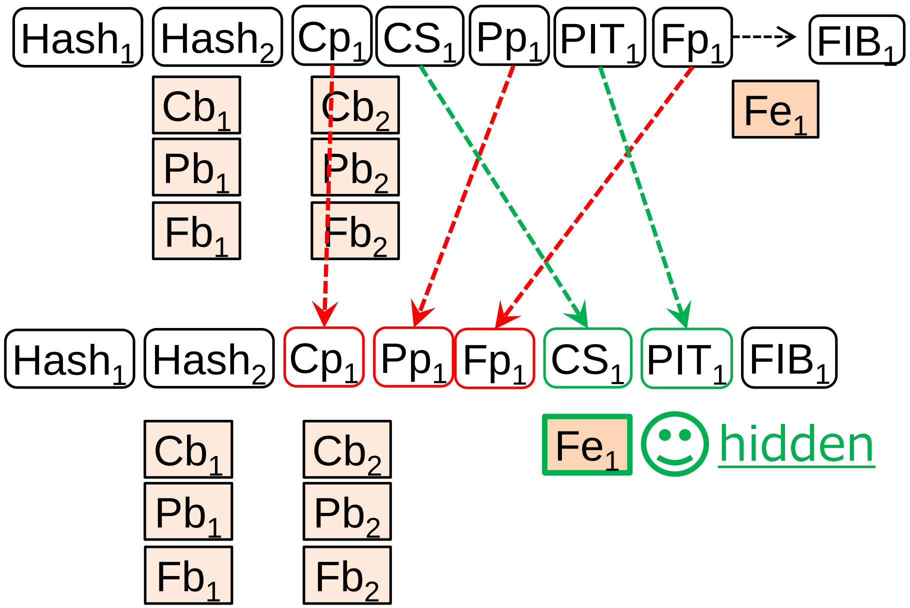
# 1) Pre-Hash Computation

- Reordering hash computations of consecutive 2 packets
  - Compute hashes for the packet 1 and 2 in advance
  - Addresses to Cb1 and Cb2 are early computed
- Put computations irrelevant to hash buckets of CS just after hash computations
  - Cb1: hidden by hash computation of packet 2
  - Cb2: hidden by computations of packet 1



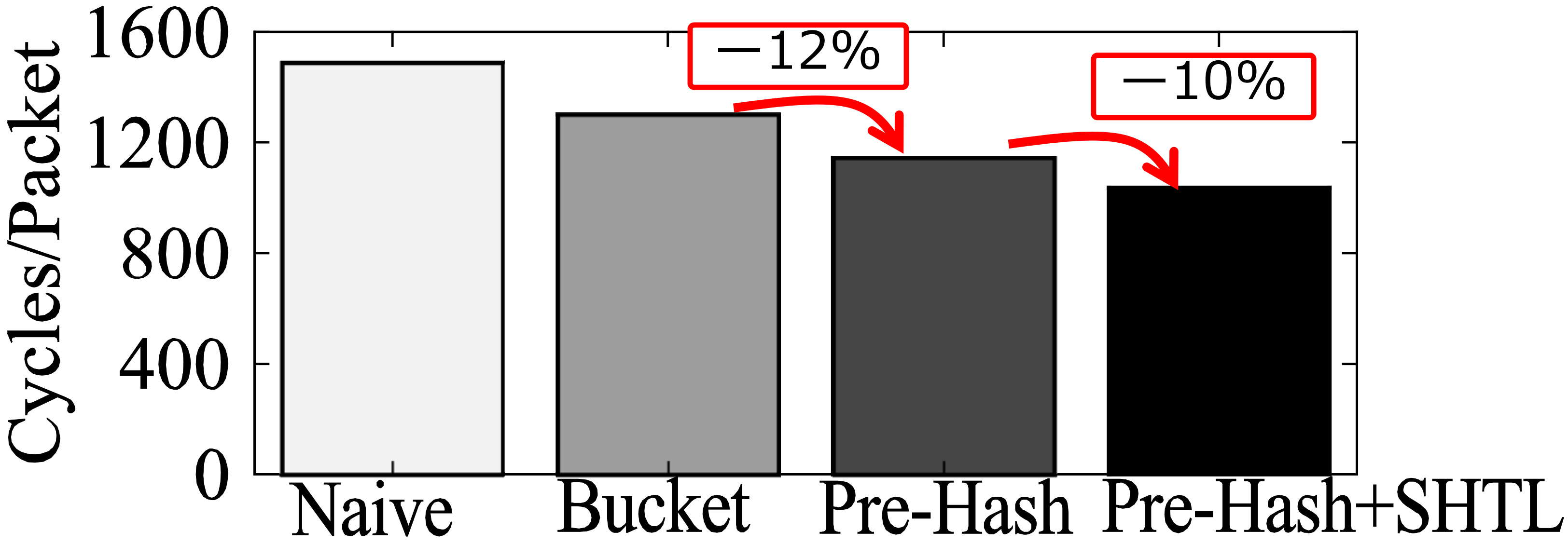
# 2) Speculative Hash Table Lookup (SHTL)

- Reordering tasks composing multiple hash table lookups of one packet
  - Speculatively search pointers to entries of all the tables
    - Address to Fe1 is early computed
  - Put computations of CS, PIT just after finding the pointer
  - Fe1: hidden by computations of CS and PIT



# Forwarding Speed

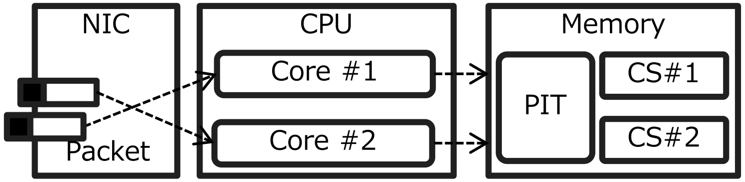
- Pre-Hash and SHTL hide DRAM access latency of data fetches
- 43.48 MPPS (about 36 Gbps) packet forwarding on a single computer
  - Xeon E5-2699 v4 (2.20GHz*22 CPU cores), 8*16 G bytes DRAM, 2*40 G Ether

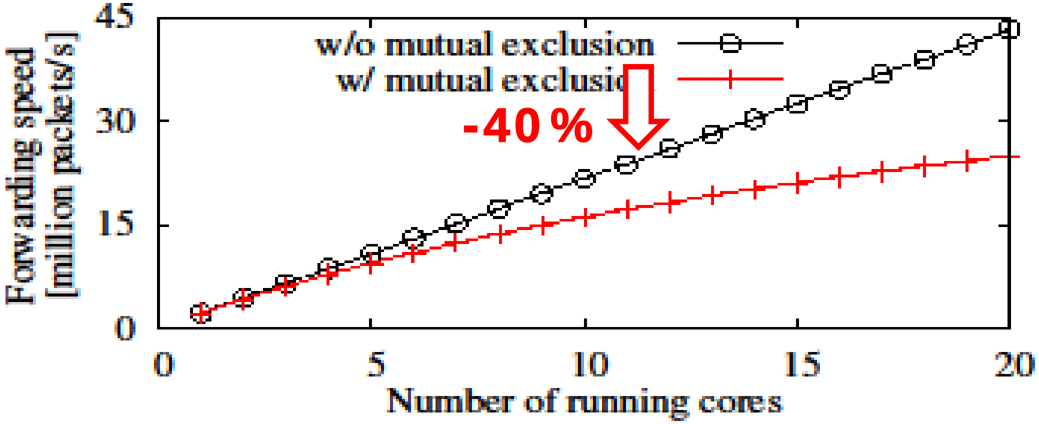


CPU cycles for processing one packet (Single-Threaded)

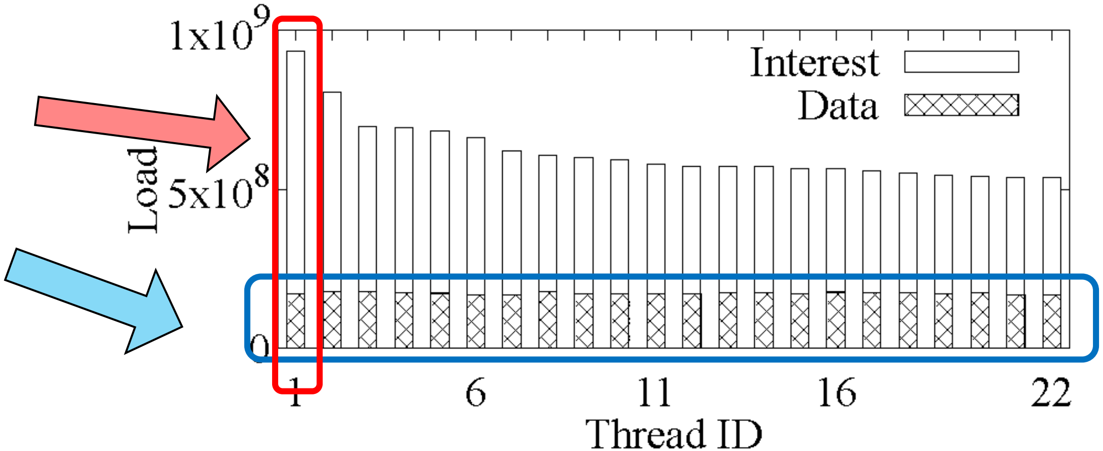
# How NIC Dispatches Incoming Packets to CPU Cores

- Random dispatching: NIC randomly chooses CPU cores
  - Mutual exclusion degrades forwarding speed



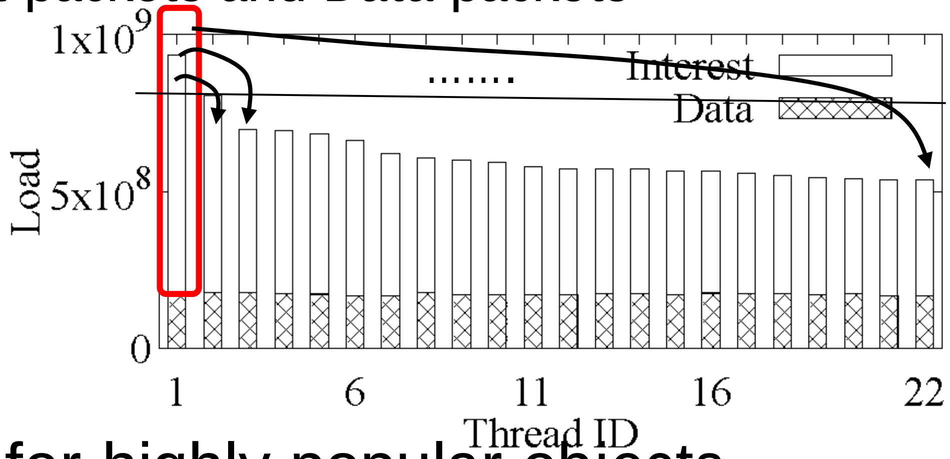


- Sharding[13]: Heavy hitters (Highly popular objects) cause load imbalance
  - Interest packets are not equally dispatched, whereas Data packets are equally assigned



# Design Rational of Equal Load Balancing [3]

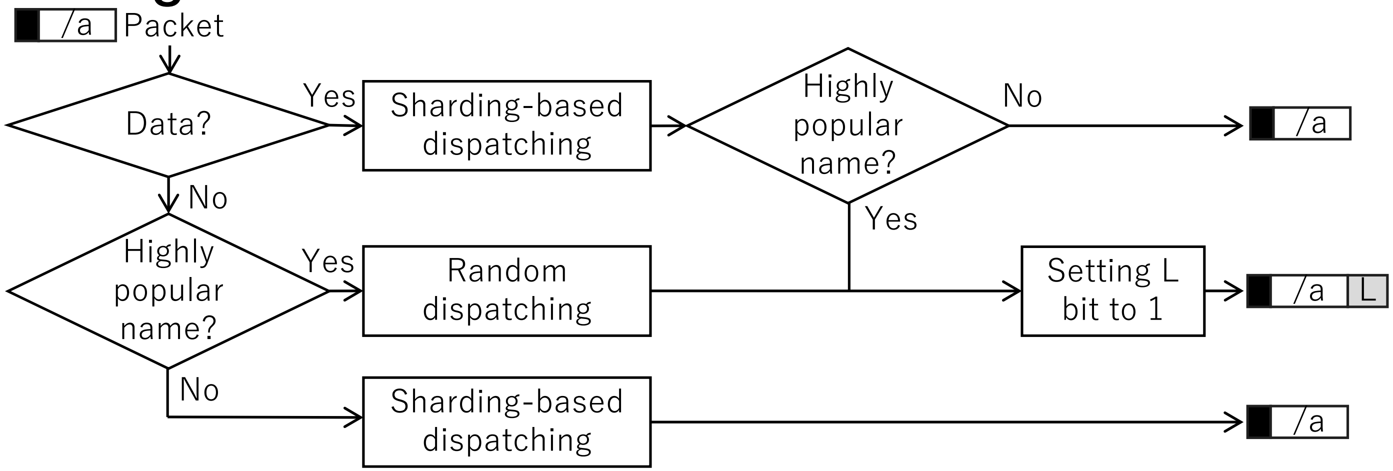
- Equal Dispatching
  - Radom dispatching of Interest packets of highly popular objects: CPU cores are randomly selected
  - Shading of the other Interest packets and Data packets
    - Interest and Data packets of unpopular objects are dispatched to the same CPU cores: No need of mutual exclusion of PIT entries



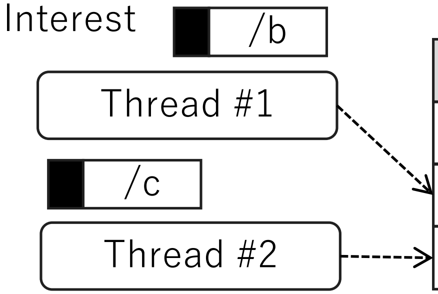
- Avoiding mutual exclusion for highly popular objects
  - Most Interest packets of highly popular objects hit the corresponding Data packets in the CS, thus PIT is not accessed
  - The remaining issue is mutual exclusion of CS entries for highly popular objects

# Algorithms of NIC and PIT

- Packet dispatching: NIC



- Fine-grained pessimistic locking: PIT
  - CAS instruction (for mutual exclusion) is rarely issued because most Interest packets of highly popular do not access the PIT



A binary lock variable

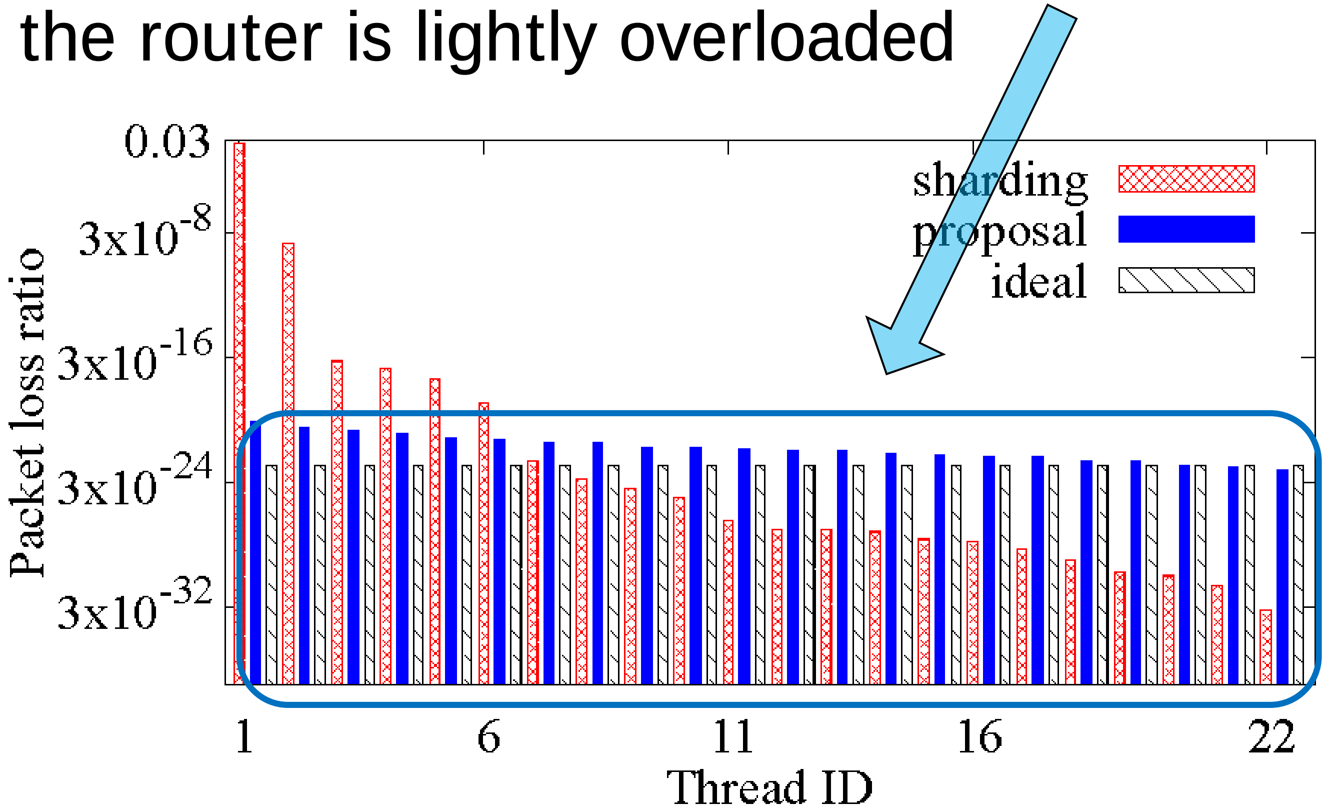
| Name | Lock | Faces |
|---|---|---|
| /a | 1 | #1 |
| /b | 0 | |
| /c | 0 | |

# Algorithm of CS: FIFO Queue with Optimistic Locking [13]

- Setup
  - CS: Look up table and FIFO queues
  - **Version variables** are prepared for entries of the look up table
- Optimistic locking provides mutual exclusion for Data packet replacement
- When a Data packet is received,
  - If the version variable is an odd value, the CPU core writes the CS entry to insert or to evict a Data packet
  - If the version variable is an even value, the CPU core does not access the CS entry, then retries reading the version variable

# Evaluation by Simulation

- Conditions: 22 CPU cores, CS hit rate 0.35
- Proposed dispatching method achieves equal load balancing of CPU cores
  - Packet loss rates of CPU cores are similar even if the router is lightly overloaded



# Summary of Software NDN routers

- Most research issues are solved for 100 Gbps packet forwarding, in the case of large Data packets (1-8K bytes)
  - Full-fledged Implementation [11]
    - Leveraging state-of-art techniques
      - DPDK: multi-threading, ring-buffer, bypass kernel
      - 2-stage LPM [2]
      - FIB replication on NUMA sockets
- Open source
  - Cefore : NICT https://cefore.net/
  - NFD (NDN Forwarding Demon) : UCLA https://named-data.net/doc/NFD/current/
  - CICN: .io https://wiki.fd.io/view/Cicn
  - CCN-lite: https://github.com/cn-uofbasel/ccn-lite

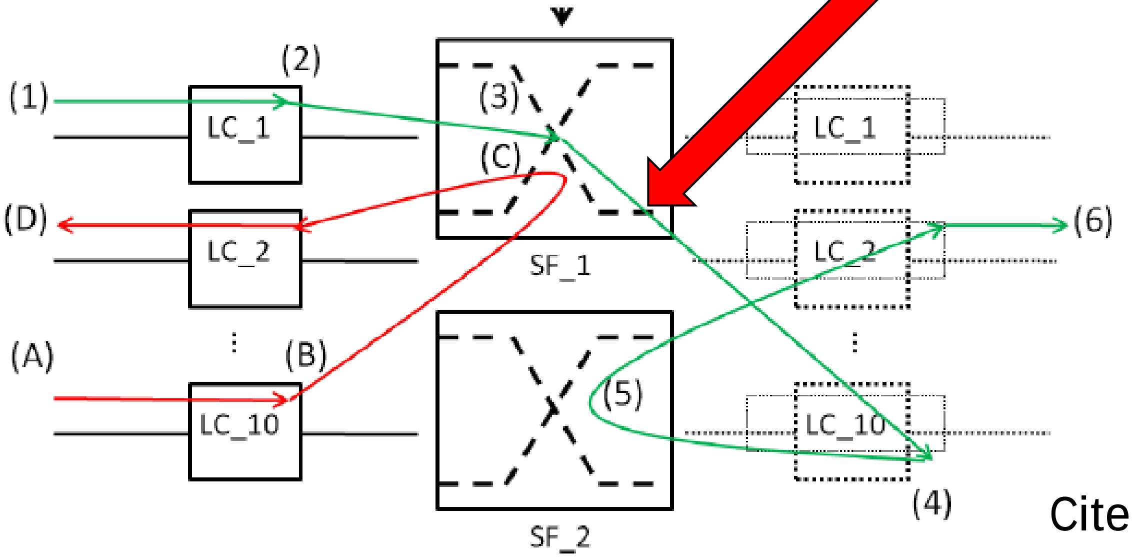
# TOWARD 10 TBPS FORWARDING: HARDWARE NDN ROUTERS

# Toward 10 Tbps NDN Packet Forwarding

- Research for toward 10 Tbps NDN packet forwarding is still in the early stage
- Approaches for hardware NDN routers
  - Naïve backplane switch and interface cards
    - Caesar [15]: One of the first hardware routers
  - P4 programmable switch
    - Programmable ASIC and interfaces
    - NDN.p4 [16]: A prototype NDN implementation on P4 programmable switches
  - P4 programmable switch and a few servers
    - On going at Osaka University
      - Learned index-base FIB as a candidate for high speed LPM
- High speed per-packet caching is not addressed, so far

# Caesar [15] : One of the first hardware routers

- FIB is separated to line cards (LCs) so that FIB is stored at SRAM devices of them
- Simple Sharding: $LC_i$ records prefixes of which hash value of the first name component is i (hash routing according to the first component)
- But, LC of the FIB entry is not always the outgoing face
  - Packets traverse the backplane switch twice



Cited from [14]

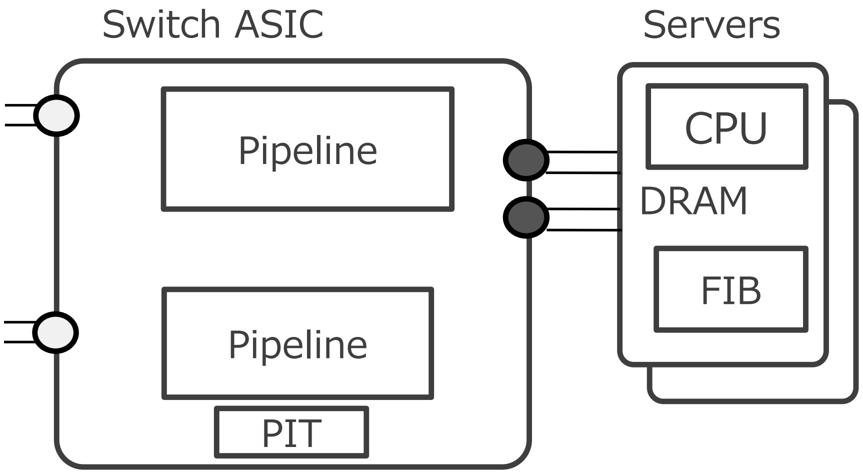
# NDN Router on P4 Programmable Switch Is in The Early State

- NDN.p4 [16]:Prototype on BMv2
  - Do not support large numbers of FIB and PIT enries

Cited from [15]

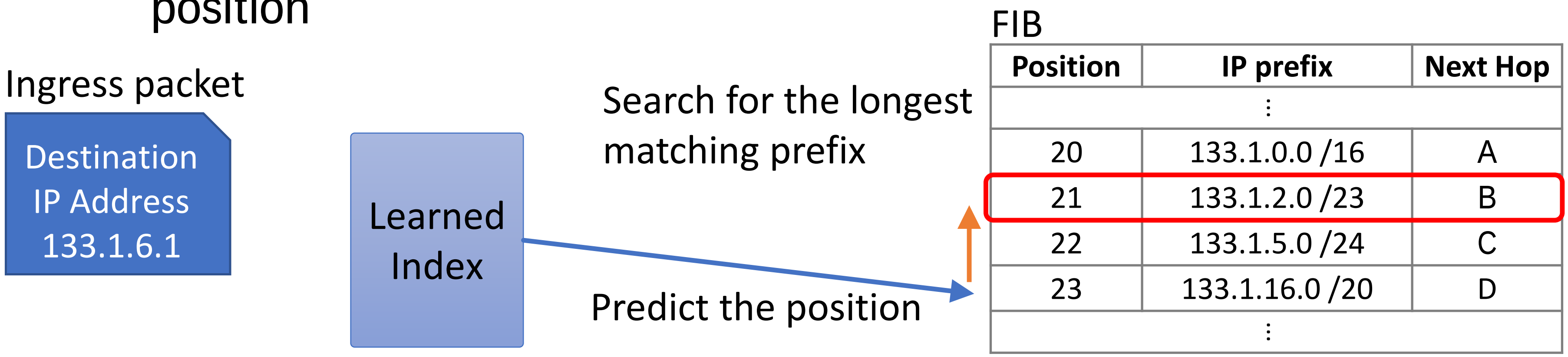
# How to Store a Large number of FIB Entries

- Compact FIB Data Structures which can be stored at TCAM and SRAM Devices of P4 ASIC
  - Multi-bit Trie:
    - The look up depth is larger than the number of stages of a single pipe line
  - Learned Index [17]
    - Float point calculations are required
- Offloading FIB to another device
  - DRAM device at the controller
  - DRAM device at a server



# LPM Using Learned Index [17]

- Learned index:Learning key-position relationships using machine learning (e.g., neural network)
- FIB based on learned index (learned FIB) for IP LPM
  - Sort IP prefixes in ascending order and store them in an array
  - Learn the relation between each IP prefix and its position with a neural network
  - Predict the position of the destination IP address of an ingress packet with the neural network
  - Search for the longest matching prefix around the predicted position


Ingress packet
Destination IP Address 133.1.6.1
Learned Index
Search for the longest matching prefix
Predict the position
FIB

| Position | IP prefix | Next Hop |
|---|---|---|
| | ⋮ | |
| 20 | 133.1.0.0 /16 | A |
| 21 | 133.1.2.0 /23 | B |
| 22 | 133.1.5.0 /24 | C |
| 23 | 133.1.16.0 /20 | D |
| | ⋮ | |

# Computation Time and Memory Size

- Computation time: The number of CPU cycles required to find the next hop for one IP address
  - LC-Trie FIB: 112 CPU cycles
  - Learned FIB: 115 CPU cycles
- Memory size
  - LC-Trie FIB: 2.504 Mbytes
    - vertices: 534,093 (32 bits/vertices)
    - bits for the skip bit validation: 6,772 bytes
      - LC-Trie FIB must maintain a bit sequence for validating skipped bits
    - next-hop information: 360,879 bytes
  - Learned FIB: 1.173 Mbytes
    - Keys in FIB entries: 229,454 (4 byte/entry)
    - weights in model: 6,400 (4 byte/weight)
    - next-hop information: 360,879 bytes

# Summary

- 100 Gbps NDN packet forwarding is almost solved
  - High speed LPM for name prefixes
  - Sharding based on front end cache
- Remaining issues for toward 10 Tbps NDN packet forwarding
  - Approaches of hardware routers
  - Research direction for accommodating large FIB

# REFERENCES

# References

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, “Networking Named Content,” Proceeding of ACM CoNEXT 2009, pp.1-12, December 2009.

[2] W, So, a. Narayanan, D. Orran, “Named data networking on a router: fast and DoS-resilient forwarding with hash tables,” in Proceedings of ACM/IEEE ACNS, pp.21-226, 2013.

[3] J. Takemasa, Y. Koizumi and T. Haseagawa, “Load Balancing for Stateful Forwarding by Mitigating Heavy Hitters: A Case for Multi-Threaded NDN Software Routers,” IEEE Accesss, vol. 8, pp. 155071-155085, 2020.

[4] G. Carofiglio, M. Gallo, L. Muscariello and D. Perino, “Pending Interest Table Sizing in Named Data Networking,” in Proceedings of ACM ICN 2015, pp. 49–58, Sept. 2015.

[5] D. Perino and M. Varvello, “A Reality Check for Content Centric Networking,” in Proceedings of ACM ICN 2011, pp.44-49, Aug. 2011.

[6] J. Takemasa, Y. Koizumi and T. Hasegawa, “Data Prefetch for Fast NDN Software Routers based on Hash Table-based Forwarding Tables,” Computer Networks, Volume 173, 22 May 2020.

[7] H. Dai, J. Lu, Y. Wang and B. Liu, “BFAST: Unified and scalable index for NDN forwarding architecture,” in Proceedings of IEEE Infocom, pp.2290-2298, May, 2015.

# References

[8] Y. wang, T. Pan, Z. Mi, H. Dai and X. Guo, T. Zhang, B. Liu and Q. Dong, “NameFilter: Achieving fast name look up with low memory cost via applying two-stage bloom filters,” in Proceedings of IEEE Infocom Mini Conference, pp. 95-99, May. 2013.

[9] T. Song, H. Yuan, P. Crowley and B. Zhang, “Scalable name-based packet forwarding: from millions to billions,” in Proceedings of ACM ICN, pp.19-28, Sept. 2015.

[10] D. Kirchner, R. Ferdous, R. Lo Cigno, L. Maccari, M. Gallo, D. Perino, and L. Saino, “Augustus: a CCN router for programmable networks,” in Proceedings of ICN, pp. 31–39, 2016.

[11] J. Shi, D. Pesaventoa and L. Benmohamed, “NDN-DPDK: NDN Forwarding at 100 Gbps on Commodity Hardware,” in Proceedings of ACM ICN, pp.30-40, Sept. 2020.

[12] B. Fan, H. Lim, D. G. Anderson and M. Kaminsky, “Small cache, big effort: Provable load balancing for randomly portioned cluster services,” in Proceedings of 2nd ACM Symp. Cloud Comput., pp.1-12, 2013.

[13] L. Saino, I. Psaras, and G. Pavlou, ``Understanding sharded caching systems,'' in Proc. 35th Annu. IEEE Int. Conf. Comput. Commun., Apr. 2016, pp. 1-9.

[14] Hyeontaek Lim et al., “MICA: A holistic approach to fast in-memory key-value storage”, in Proceedings od USENIX NSDI 14, pp.429-444, April 2014.

[15] D. Perino, M. Varvello, L. Linguaglossa, R. Laufer, R. Boislaigue, “Caesar: A content router for highspeed forwarding on content names,” In Proceedings of ACM ANCS 2014, oo.137-148, Feb. 2014.

# References

[16] S. Signorello, R. State, J. François and O. Festor, "NDN.p4: Programming information-centric data-planes," 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 384-389, doi: 10.1109/NETSOFT.2016.7502472.

[17] Shunsuke Higuchi, Junji Takemasa, Yuki Koizumi, Atsushi Tagami and Toru Hasegawa, “Feasibility of Longest Prefix Matching using Learned Index Structures,” ACM SIGMETRICS Performance Evaluation Review, Volume 48, Issue 4, pp 45–48, March 2021.