Poster: A Method for Designing High-speed Software NDN Routers

Kosuke Taniguchi
Osaka University
k-taniguchi@ist.osaka-u.ac.jp

Junji Takemasa
Osaka University
j-takemasa@ist.osaka-u.ac.jp

Yuki Koizumi
Osaka University
ykoizumi@ist.osaka-u.ac.jp

Toru Hasegawa
Osaka University
t-hasegawa@ist.osaka-u.ac.jp

ABSTRACT

To build high-speed NDN software routers, issues of time-consuming and memory-consuming functions of Named Data Networking (NDN) routers, i.e., caching and name-based forwarding, should be resolved. This paper highlights how caching and name-based forwarding consume computation time (CPU cycles) and memory (I/O bandwidth) in commercial off-the-shelf (COTS) computers. We implement a prototype of building blocks of an NDN software router and empirically measure the consumption of CPU cycles and dynamic random-access memory (DRAM) channels. Based on the empirical measurements, we reveal several implications for designing high-speed NDN software routers on current COTS computers.

CCS Concepts

•**Networks** → **Routers;** *Network design principles;* Packet-switching networks;

Keywords

Named Data Networking, Caching, Forwarding, Router Architecture

1. INTRODUCTION

Designing a high-speed NDN software router, which is built on a hardware platform based on a COTS computer, is a hot research topic. Although many existing studies focus on fast computation of caching and name-based forwarding [4], they do not consider how their proposed caching and name-based forwarding consume I/O bandwidth, such as DRAM channels, of COTS computers. The other study focuses on efficient exploitation of large memory resources, such as solid state drives (SSDs) [2]. However, it does not consider how the approach sacrifices the computation capabilities for caching and name-based forwarding.

ICN'16 September 26-28, 2016, Kyoto, Japan
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4467-8/16/09.
DOI: http://dx.doi.org/10.1145/2984356.2985234

This paper develops a method for designing high-speed NDN software routers, which is based on bottlenecks analyses of a NDN software router built on a COTS computer. The contributions of the paper are two-fold: i) As far as we know, this is the first study that sheds light on how caching and name-based forwarding consume I/O bandwidth of COTS computers and computation time, i.e., CPU cycles. Our method implies that such detailed analyses are indispensable to high-speed NDN software routers. ii) Through the implementation of a building blocks of the state-of-the-art NDN software router and empirical measurements based on the implementation, we provide several implications for designing high-speed NDN software routers on currently available COTS computers.

2. BOTTLENECK ANALYSIS

2.1 Reference NDN Software Router

A reference hardware platform of an NDN software router is illustrated in Fig. 1. The platform is based on the latest CPUs and chipsets [1]. Each CPU has multiple CPU cores, a memory controller and a PCI express (PCIe) 3.0 controller on its die. The memory controller has up to four DDR4 DRAM channels. PCIe devices, such as SSDs and network interface cards (NICs) are connected via the PCIe controller. A reference software platform is illustrated in Fig. 2. The platform is the minimum NDN router, which supports only caching and name-based forwarding.

2.2 Bottleneck Analysis

To achieve high-speed caching and name-based forwarding on a COTS computer, we have to analyze two notable bottlenecks, i.e., computation time and I/O bandwidth.

The packet forwarding rate, λ packet/s, is defined as the number of Interest and Data packet pairs forwarded per second. When the packet forwarding rate is λ, the consumed CPU cycles are derived as λC, where C is the average CPU cycles consumed to process one pair of Interest and Data packets and it is derived using the model in [3]. If assuming that the load for processing packets is equally assigned to N CPU cores operating at the frequency of F Hz, the maximum packet forwarding rate, $\lambda_{\max}$, must satisfy $\lambda_{\max} \le NF/C$. In the same way, the average number of bytes read from and written to DRAMs for processing one pair of Interest and Data packets is expressed as M and it can be modeled similarly to the CPU cycles. The maximum packet forwarding rate $\lambda_{\max}$ satisfy $\lambda_{\max} \le B/M$, where B is the bandwidth of DRAM channels, and

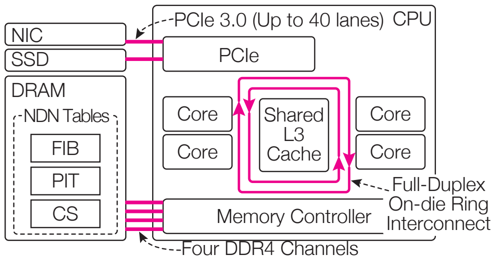
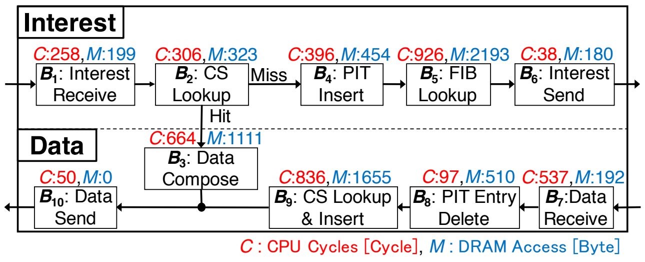


Figure 1: Hardware block diagram of an NDN software router



Interest
C:258, M:199 B1: Interest Receive
C:306, M:323 B2: CS Lookup
Miss
C:396, M:454 B4: PIT Insert
C:926, M:2193 B5: FIB Lookup
C:38, M:180 B6: Interest Send
Hit
Data
C:664, M:1111 B3: Data Compose
C:50, M:0 B10: Data Send
C:836, M:1655 B9: CS Lookup & Insert
C:97, M:510 B8: PIT Entry Delete
C:537, M:192 B7: Data Receive
C : CPU Cycles [Cycle], M : DRAM Access [Byte]

Figure 2: Software block diagram and the consumption of CPU cycles and I/O bandwidth

thereby $\lambda_{\max}$ satisfy

$$\lambda_{\max} \leq \min\left(NF/C, B/M\right). \tag{1}$$

3. PRELIMINARY EXPERIMENT

We implement a prototype of building blocks of an NDN software router designed by So et al. [4], which router achieves about 9×10^6 packets/s forwarding rate, and empirically measure how the NDN software router consumes the CPU cycles and I/O bandwidth to process a packet. We use a computer having one Xeon E5-2623 v3 CPU (3.00 GHz $\times$ 4 CPU cores) and four DDR4 16 GB DRAMs as a hardware platform of the NDN software router. The evaluation is accomplished using the prototype blocks with two steps: I) Measuring the CPU cycles and I/O bandwidth consumed to process a packet and II) estimating the maximum packet forwarding rate bounded by CPU computation time and I/O bandwidth.

I) CPU Cycles and I/O Bandwidth Consumption: CPU cycles and I/O bandwidth consumed by each block for processing one packet are measured using the time-stamp counter and the performance monitoring unit on the CPU, respectively. The CPU cycles and the DRAM channel bandwidth consumed by each block are shown in Fig. 2.

II) Maximum Throughput: Based on the measured results, we estimate the maximum packet forwarding rate bounded by either computation time or DRAM channel bandwidth. We focus on two computers: Computer #1 has one CPU (2.2 GHz $\times$ 22 CPU cores), which has the highest number of CPU cores among Intel Xeon E5 CPUs, and computer #2 has one CPU (2.6 GHz $\times$ 4 CPU cores) [1]. The maximum bandwidth of DRAM channels on both computers scales up to 76.8 GByte/s [1]. During this analysis, we assume 30% CS hit rate. The average CPU cycles and DRAM bandwidth consumed for processing a pair of Interest and Data packets are $C = 2794.2$ cycles and $M = 4484.1$ Bytes, respectively. The maximum packet forwarding rate bounded by DRAM bandwidth of both computers is $B/M = 17.1 \times 10^6$ packets/s. In contrast, the maximum packet forwarding rates bounded by CPU cycles, i.e., NF/C, for computer #1 and #2 are calculated to be $NF/C = 17.3 \times 10^6$ and 3.7×10^6 packet/s, respectively. In summary, the computation time is the stricter constraint to determine the maximum packet forwarding rate, i.e., $NF/C \ll B/M$, except for the case that the number of CPU cores is high, i.e., $NF/C \approx B/M$.

4. IMPLICATIONS

Through the bottleneck analysis and the measurements, we obtain the following implications for designing high-speed NDN software routers on currently available computers:

I) I/O Design: In the case that the number of CPU cores is low, i.e., $NF/C \ll B/M$, CPU cores should be used for caching and name-based forwarding to maximize the packet forwarding rate. A router cannot afford to assign CPU cores to other tasks than caching and name-based forwarding. For instance, the router implementation [2] focuses on exploitation of SSDs as a CS. It uses several CPU cores to compensate the latency to access SSDs. From the perspective of the forwarding rate, the approach sacrifices the computation capabilities for caching and name-based forwarding. In contrast, in the case that the number of CPU cores is high, i.e., $NF/C \geq B/M$, the implementation, where some of the tables is placed on SSDs to distribute accesses to the tables to DRAM channels and PCIe lanes, is a reasonable solution.

II) Computation Design: The computation time includes waiting time to fetch data from DRAMs, and hence data structures and algorithms should be not only fast but also highly optimized so that the number of data fetches from DRAMs is minimized. In the same way, mutual exclusion for any shared data causes I/O waits, which are also included in the computation time. Thus, a FIB, PIT and CS, must be implemented with lock-free data structures to fully exploit computing capabilities of multi-core CPUs. The simplest way to realize lock-free data structures is to prepare exclusive NDN tables, a FIB, PIT, and CS, for each thread. Each thread processes Interest and Data packets in a part of the whole name space so that it does not access the tables of the other threads.

5. ACKNOWLEDGMENTS

This work has been supported by the ICN2020 Project (Advancing ICN towards real-world deployment through research, innovative applications, and global scale experimentation), a research project supported jointly by the European Commission under its HORIZON 2020 (Grant Agreement No. 723014) and the National Institute of Information and Communications Technology (NICT) in Japan (Contract No. 184).

6. REFERENCES

[1] Intel Corporation. Intel® Xeon® processor E5 v4 family. http://ark.intel.com/products/family/91287.

[2] R. B. Mansilha, L. Saino, M. P. Barcellos, M. Gallo, E. Leonardi, D. Perino, and D. Rossi. Hierarchical content stores in high-speed ICN routers: Emulation and prototype implementation. In *Proceedings of ACM ICN*, pages 59–68, Sept. 2015.

[3] K. Ohsugi, J. Takemasa, Y. Koizumi, T. Hasegawa, and I. Psaras. Power consumption model of NDN-based multicore software router based on detailed protocol analysis. *IEEE Journal on Selected Areas in Communications*, 34:1631–1644, May 2016.

[4] W. So, A. Narayanan, and D. Oran. Named data networking on a router: Fast and DoS-resistant forwarding with hash tables. In *Proceedings of ACM/IEEE ANCS*, pages 215–226, Oct. 2013.