Toward Energy-Efficiency Optimization of Pktgen-DPDK for Green Network Testbeds

Guo Li[1], Dafang Zhang[1,*], Yanbiao Li[1], Keqin Li[1,2]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2] Department of Computer Science, State University of New York, New Paltz NY 12561, USA
* The corresponding author, email: dfzhang@hnu.edu.cn

Abstract: The packet generator (pktgen) is a fundamental module of the majority of software testers used to benchmark network protocols and functions. The high performance of the pktgen is an important feature of Future Internet Testbeds, and DPDK is a network packet accelerated platform, so we can use DPDK to improve performance. Meanwhile, green computing is advocated for in the future of the internet. Most existing efforts have contributed to improving either performance or accuracy. We, however, shifted the focus to energy-efficiency. We find that high performance comes at the cost of high energy consumption. Therefore, we started from a widely used high performance schema, deeply studying the multi-core platform, especially in terms of parallelism, core allocation, and frequency controlling. On this basis, we proposed an AFfinity-oriented Fine-grained CONtrolling (AFFCON) mechanism in order to improve energy efficiency with desirable performance. As clearly demonstrated through a series of evaluative experiments, our proposal can reduce CPU power consumption by up to 11% while maintaining throughput at the line rate.

Keywords: CPU affinity; DPDK; energy-efficient; NUMA; packet generator

I. INTRODUCTION

The high-performance packet generator (pktgen) is a fundamental component of network testing tools. Emerging techniques, such as software-defined networking/network function virtualization (SDN / NFV) and cloud computing pose high demands on performance and flexibility of software testers. In large-scale testbeds, especially when SDN/NFV is deployed, different logical testers may physically run on the same device. Despite limited resources and bandwidth, accumulated energy consumption may become the next restriction. Additionally, energy-efficiency is evolving to be a hot topic in this area, especially in the area of green computing.

In recent years, new software and hardware techniques have brought about a trustworthy packet generator with adequately high performance using off-the-shelf hardware [1]. However, most of the focus has been on improving performance and accuracy [2], [3], [4], [5], [6], and few take power consumption into account. In the large scale of testbeds, however, power consumption becomes a constraint.

The testbeds of the future network can be used to validate innovative applications and

Received: May 2, 2017
Revised: Aug. 12, 2017
Editor: Tao Huang

The authors have investigated the effect of CPU Affinity on NUMA architecture server and packet generator. The AFFCON solution is proposed in this paper.

protocols, as some high intensity tests require a greater power supply. Meanwhile, the increase of cooling power will consume the limited total system power supply, leading to a restriction in improving the performance test.

Therefore, the trade-off between high-performance and energy-efficiency is worth discussion. A small reduction in power usage in critical devices such as a CPU will save global energy resources, including energy related to cooling systems.

The evolution of a multi-core CPU has had a profound effect on packet generators regarding both hardware and software. First, the original network card is driven by interruptions, and then develops into a combination of interruptions and polling. The multi-core CPU now makes pure efficient polling more feasible. Second, the memory access controller is moving closer to the architecture of a CPU in a multi-core non-uniform memory access (NUMA) server. Third, new software packet generators—such as the pktgen-DPDK—take full advantage of the multi-core and a huge page memory with the evolution of associated hardware.

There has been a considerable amount of related work conducted on packet generators. Wu [2] proposed a high-performance network measurement platform utilizing multi-core processors based on DPDK. Refs. [3], [7] presented a novel method for the precise control of inter-packet gaps in software based on DPDK. This method was even better than the hardware generators on rate control. Xu [8] proposed a flexibility and cheapness packet of an I/O engine, which was based on the libraries of DPDK while also possessing a high-performance for compute-intensive applications. In [4], some typical packet generators were compared as a means to measure the performance of high speed networks. Refs. [4], [9], [13] expressed that the trade-off between throughput and latency were of concern. Ref. [10] offered a flexible and fine-grained packet traffic statistics on the Open vSwitch (OVS) architecture. Refs. [5], [11], [12] discussed the high-speed packet forwarding on multi-processors. Li [14], [15] proposed a dynamic power management in order to improve multi-core server performance and reduce energy consumption in clouds and data centers.

Overall, further improvement is required between high-performance and energy-efficiency on the platform of network packet generators.

The DPDK official documents encourage using an affinity technique, because CPU affinity can improve performance. We find that, in using a CPU affinity, not only is performance improved, but energy consumption can also be saved. It is analogous to the concept that a manual transmission is more fuel-efficient than an automatic transmission in vehicles. Therefore, we use a fine-grained CPU Affinity mode based on pktgen-DPDK in an effort to improve performance and optimize energy consumption.

The goals are to utilize the feature of DPDK and to propose a scheme of arranging CPU cores in order to improve performance and save energy consumption, in comparison to the arrangement of the operating system scheme. The benefits of using affinity are the improvement of throughput, reduction of latency, and decreased energy consumption.

CPU affinity will lead, however, to a more complicated management overhead. In order to solve the problem, we aim to exactly arrange CPU cores to every port by hand instead of using the system arrangement. Therefore, our algorithm is a manual management scheme to CPU cores.

The experimental results show that the optimization between energy-efficiency and high-performance is achievable. The key contributions of our work are summarized as follows:

• Characterized the CPU Affinity of a multi-core NUMA server in the hardware.

• Deployed some representative packet generators in order to demonstrate and compare their performance and power consumption.

• Proposed an AFFCON solution, which includes the four components of CPU Affinity, NUMA Affinity, DVFS Affinity and Rate

Affinity. AFFCON can reduce the CPU power consumption, while keeping the line rate and decreasing latency and jitter.

The remainder of this paper is organized as follows: Section II describes software packet generators; Section III, analyzes CPU Affinity and proposes an energy-efficient solution; Section IV presents experimental results; and Section V summarizes the conclusions of the work.

II. Packet Generator

2.1 Software packet generator

Software packet generators can be divided into different categories with various features. We provide a presentation of the possible classifications along the time sequence.

First, the packet generators are based on a system socket within the OS user layer, such as Netperf and Iperf. To perform a Netperf or Iperf test, the user must establish a connection between the server and client, only then can the program sends packets over the OS sockets interface.

Second, directly generating packets within the OS kernel layer through the system network drivers. A representative is the Linux packet generator, it is as short as the Pktgen-Linux [16].

Third is to send the packets in the OS user layer with their own network drivers. For instance, Netmap [17], [18] is a framework for the fast packet I/O.

Fourth, the packet generators use their own network card drivers with huge page memory. For example, pktgen-DPDK can act as a transmitter or receiver at line rate within the minimum number of CPU cycles. MoonGen [3], [7] can saturate a line rate connection and it is also built on the DPDK.

All in all, the new generation of DPDK-based packet generators possess powerful functions as well as a higher performance.

2.2 DPDK

The DPDK consists of a set of libraries and network drivers for fast packet processing. There are four advantages that make the DPDK powerful. Those of which include huge memory, ring buffer, multi-core framework and poll-mode drivers.

The huge page memory can decrease cache miss, and prevent swapped memory. The ring buffer is a producer-consumer model, and can speed up the efficiency of memory access in the high-speed network. The multi-core framework is also feasible for CPU Affinity. The poll-mode is beneficial for high-performance, and can also pin tasks to the CPU cores. In summary, the features of the DPDK creates a higher performance compared with other platforms.

III. CPU Affinity Power Optimization

3.1 Server platform architecture

3.1.1 Traditional architecture

Symmetric Multi-processing (SMP) is a traditional server platform architecture, the multiple processors are all scheduled with equal access to the memory bank. Sometimes, a SMP is referred to as Uniform Memory Architecture (UMA).

SMP architecture works satisfactorily for a single physical CPU. However, with the development of the CPU cores, modern CPUs operate considerably faster than the memory. Subsequently, the Northbridge encounters obvious performance problems and bottlenecks in its response time, as all memory traffic is routed through this bridge.

3.1.2 NUMA architecture

In order to improve the performance bottleneck of the SMP/UMA architecture, Non Uniform Memory Access (NUMA) architectures is proposed. NUMA is logically derived from UMA architecture. Under the NUMA architecture, the memory access strategy is distributed. Each processor has a ‘local’ bank of memory, with each processor accessing its own local memory much faster.

3.2 CPU affinity

CPU Affinity, or CPU pinning is the ability to bind the program processes to the specified CPU cores, in multi-core or multi-processor systems. As affinity can be explicitly assigned to utilize a certain set of CPU cores, it can take advantage of its capabilities that the reduced cache misses. Affinity can be used to ensure that a process is always using assigned cores, while at the same time utilizing the best performance of the CPU cores.

There are already some benefits of the CPU affinity.

Firstly, it optimizes the CPU cache performance. If a thread changes to another processor, the cache data may get out of sync, and the result would be a costly cache switch. A CPU affinity protects against this potential problem, and improves the performance of the cache throughput. Secondly, the CPU affinity is useful for real-time or time-sensitive applications. As such, the CPU affinity improves latency.

Generally, an OS server has automatic scheduling algorithms that retrieve and set the CPU affinity of a processor. Some circumstances will cause a thread to change to another processor if the thread is in the process of a relatively heavy-load. Nonetheless, the switch between processors may introduce a cache miss problem, leading to a temporary reduction in performance. It is irrelevant for most tasks, except for tasks such as processor-intensive tasks.

To provide a similar example, an automatic transmission (AT) adds an automatic gears shift device that is operated by a vehicle Electronic Control Unit (ECU). A manual transmission (MT) has a feature whereby a gear is manually selected. Under some complicated road conditions, an AT shifts a gear, which is conducted by the ECU. However, the gear changes may be unnecessary. Just like an OS, an ECU may not cover all situations. A MT can avoid such an undesired gear ratio shift, because gear changes are not automatic. The gear pinning mode is similar with the CPU pinning (CPU Affinity) mode. In regards to our CPU and under some circumstances, the server with the NUMA architecture may require a CPU affinity operation instead of the OS auto scheduling. Considering the fuel-efficiency found in manual modes, we wonder whether the analogous power-efficient function may be useful in selectable CPU affinity modes.

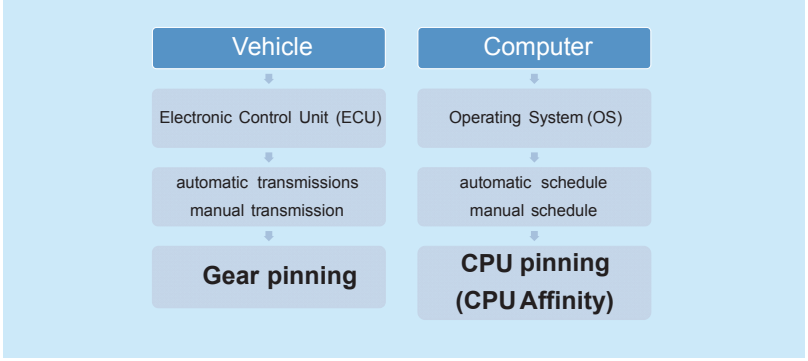


Fig. 1. *Gear pinning and CPU pinning.*

3.3 Power optimization algorithm based on cpu affinity

In multi-core NUMA server architecture, logical core indexes are not arranged in order within the physical CPU. Instead, indexes are arranged alternately, as shown in Table I. Note that table I is generated by professional DPDK tools "cpu_layout.py," and the indexes of cores suddenly jump from 5 to 8 because the system set the jump indexes separately from the two physical CPUs.

Taking our NUMA architecture server with two E5-2650 v4 CPUs as an example, a physical E5-2650 v4 CPU has twelve physical cores. The OS maps two logical cores for every physical core using Hyper-Threading, with the total number of logical cores is 48. The logical cores 0 and 1 are distributed to the two NUMA nodes or two physical CPU's, with this interwoven arrangement considerate of the CPU load balance. However, on a dedicated packet generators platform, the trade-off between high-performance and energy-efficiency is a primary concern, so we can use a CPU Affinity and NUMA Affinity to optimize performance and improve efficiencies.

In a NUMA system, DPDK developers ad-

vocate accessing local memory since remote memory access is slower. CPU Affinity can pin the CPU cores to the local memory.

A CPU Affinity uses a binary mask to set. A binary mask is a series of n bits, where each bit individually stands for a logical CPU core. The number 1 denotes an allowed binary mask, with 0 indicating it is a binary mask that is not allowed. Whenever the OS attempts to migrate a processor from one logical CPU core to another, it first checks to see if the destination core's bit is set in an allowed binary mask. For example, the CPU cores mask '0x1f' maps to logical cores [0,1,2,3,4], '0x155' maps to [0,2,4,6,8], and '0x2aa' maps to [1,3,5,7,9].

Dynamic Voltage and Frequency Scaling (DVFS) is a basic power management technique used in computer architecture. Many modern devices such as CPU and GPU allow frequency regulation to be controlled through software. CPUfreq is a linux tool that makes decisions to increase or decrease CPU running frequencies, in order to enhance performance or save energy. There are five governor modes provided by CPUfreq: 'Performance', 'Powersave', 'Ondemand', 'Userspace', and 'Conservative'. The default governor mode is 'Performance', which means it always runs at maximum frequency. 'Powersave' represents running at a constant minimum frequency. and 'Userspace' is a policy of a user-defined frequency. 'Ondemand' refers to quickly scaling up the CPU frequency once there is a high load, while 'Conservative' increases the frequency at a slow pace.

The governor mode of DVFS can be set to every logical core, thus, we can use this as an additional DVFS Affinity, to regulate the trade-off between high-performance and energy-efficiency. We propose an AFFCON solution, which includes four components of CPU Affinity, NUMA Affinity, DVFS Affinity and Rate Affinity, as shown in figure2. The algorithm is shown in Alg.1.

- NUMA affinity. We checked the slot of a network interface card, and then judged which NUMA node (CPU 0 or CPU 1) was affinity.
- DVFS affinity. We set the regulating governor mode to 'Performance' in the affinity cores, and 'Powersave' in the non-affinity cores.
- CPU affinity. We set a binary mask to control the CPU Affinity in Pktgen-DPDK. Through the manual operation of the CPU Affinity, unnecessary context switches which operated by the OS automatically can be avoided.
- Rate affinity. This is a dynamic adjustment process, combining the throughput of the port and the DVFS affinity of the related CPU core, performance and energy consumption can reach a balance.

Our AFFCON solution guarantees the performance of working cores and reduces the power of idle cores through the four affinity methods from hardware setting and algorithm adjusting.

Table I. *The layout of two E5-2650 v4 CPU cores.*

	Socket 0 (CPU 0)	Socket 1 (CPU 1)
Core 0	[0, 24]	[1, 25]
Core 1	[2, 26]	[3, 27]
Core 2	[4, 28]	[5, 29]
Core 3	[6, 30]	[7, 31]
Core 4	[8, 32]	[9, 33]
Core 5	[10, 34]	[11, 35]
--	--	--
Core 8	[12, 36]	[13, 37]
Core 9	[14, 38]	[15, 39]
Core 10	[16, 40]	[17, 41]
Core 11	[18, 42]	[19, 43]
Core 12	[20, 44]	[21, 45]
Core 13	[22, 46]	[23, 47]

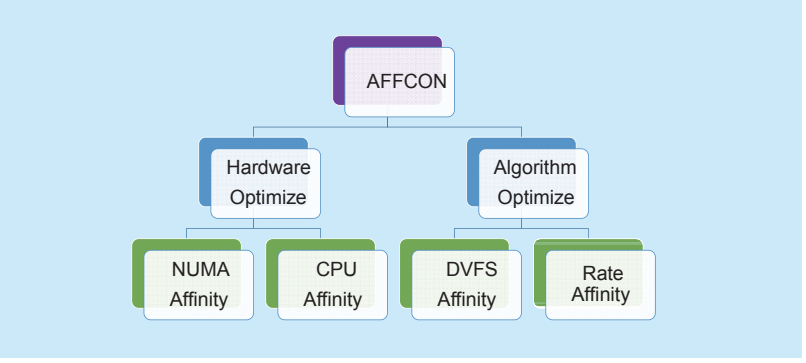


Fig. 2. *The AFFCON framework.*

Algorithm 1. AFFCON.

Require: NUMA structure, Packet generator command
Ensure: Performance and power value
1: **if** Network card slot is in NUMA 0 **then**
2: prepare working cores on NUMA 0
3: **else**
4: prepare working cores on NUMA 1
5: **end if**
6: //set core isolation to the prepared CPU cores
7: //map working cores to ports
8: **for** each core in CPU.cores **do**
9: set DVFS Affinity
10: **if** core is in prepared cores **then**
11: set to performance mode
12: **else**
13: set to powersave mode
14: **end if**
15: **end for**
16: **for** each port in Netwrok.ports **do**
17: set Rate Affinity
18: **if** rate is high **then**
19: set the cores mapped to the port in working status
20: **else**
21: set the cores mapped to the port in idle status
22: **end if**
23: **end for**

Table II. *The specifications of our experimental platform.*

Specification	Server A	Server B
Model	Dell PowerEdge T630	Dell PowerEdge T620
CPU	2 * Intel Xeon E5-2650v4@2.20GHz	2 * Intel Xeon E5-2603@1.80GHz
Thermal Design Power	2 * E5-2650 v4 @ 105W	2 * E5-2603 @ 80W
Socket number	2	2
Physical cores per socket	12	6
Logical cores	48	24
Memory	32 Gb DDR4 @ 2400MHz	64 Gb DDR3 @ 1333MHz
NIC	Intel X710-DA4 4*10GbE (SPF+)	Intel X710-DA4 4*10GbE (SPF+)
Operating system	Ubuntu 16.04 server	Ubuntu 16.04 server
Architecture	x86 64	x86 64
DPDK version	16.04	16.04

IV. Experiments and Evaluation

The specifications of our experimental platform is shown in Table II.

Note that the total number of logical cores is double of physical cores by Hyper Threading, according to Equation (1),

$$Logical\ cores = sockets * CPU\ cores * 2. \quad (1)$$

We adopted a direct connection by using four optical fibers, each port has an ability of 10 giga bits per second (Gbps) throughput. As such, the total throughput is 40Gbps.

First of all, we conducted a series of experiments on a single port. The packet generator platform is server A. The throughput comparison of various packet generators is shown at figure3(a).

When the size of packet is 64 bytes, the throughput cannot reach 10Gbps. This is because the number of generating packets has an upper limit of 14.88 million packets per second (Mpps) due to frame gap and the minimum packet structure. According to Equation (2),

$$Throughput = pps * (8 * pkt\_size). \quad (2)$$

the theory throughput of 64 bytes packet is 7.6Gbps. When the size of the packet is 1024 bytes, the packet generators of Netmap, Pktgen-DPDK and MoonGen are close to the theoretical line rate. Furthermore, in the three high-performance packet generators, Pktgen-DPDK has the lowest power consumption, as shown in figure3(b).

Next, we conducted CPU Affinity tests with Pktgen-DPDK, with the results of tests shown in figure4. On the horizontal axis, [0,1,2,3,4] stands for using logical cores sequentially. Our network card is in the NUMA node of CPU 0, so [1,3,5,7,9] is CPU Affinity but without the NUMA Affinity. It is obvious that [0,2,4,6,8], which represents the CPU Affinity and the NUMA Affinity, has a lower power consumption and a higher comprehensive performance in throughput, latency and jitter.

Then, we used the AFFCON solution including DVFS Affinity, and added more CPU cores. When we started the Pktgen-DPDK and used logical cores 0-8, there were tips that 'detected 48 logical cores', but 'nothing to do on logical cores 9-47'. This situation is a good opportunity to adopt the DVFS Affinity, as we set the working cores 0-8 in the governor mode of 'Performance', and the idling cores

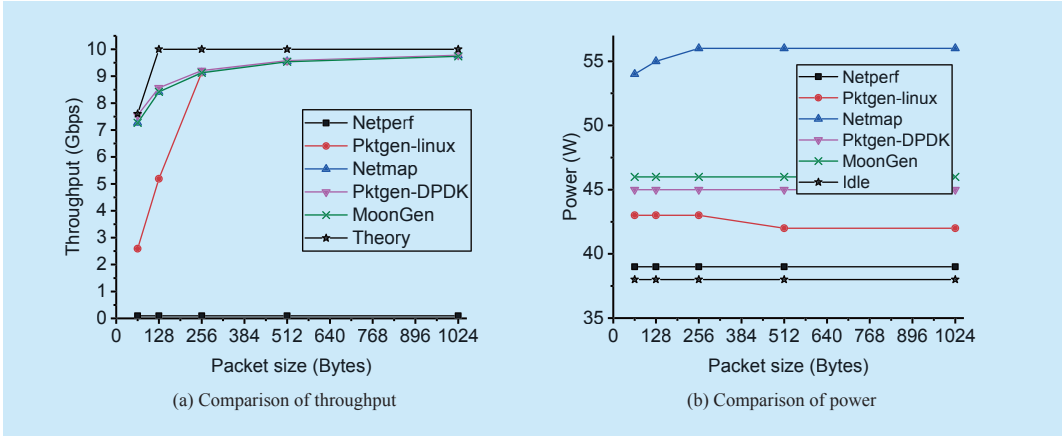


Fig. 3. *Comparison tests.*

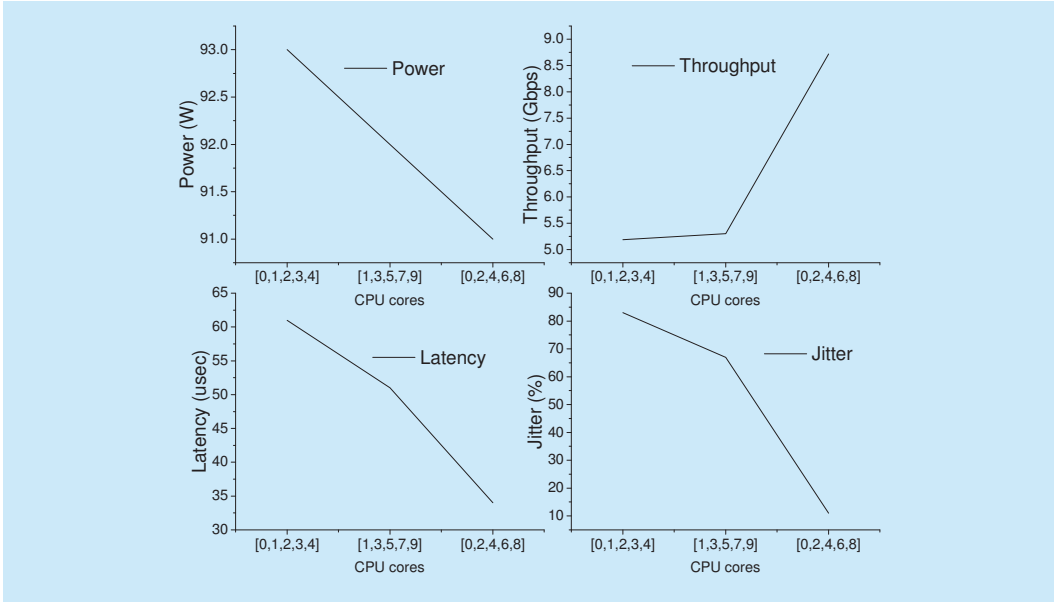


Fig. 4. *CPU affinity tests.*

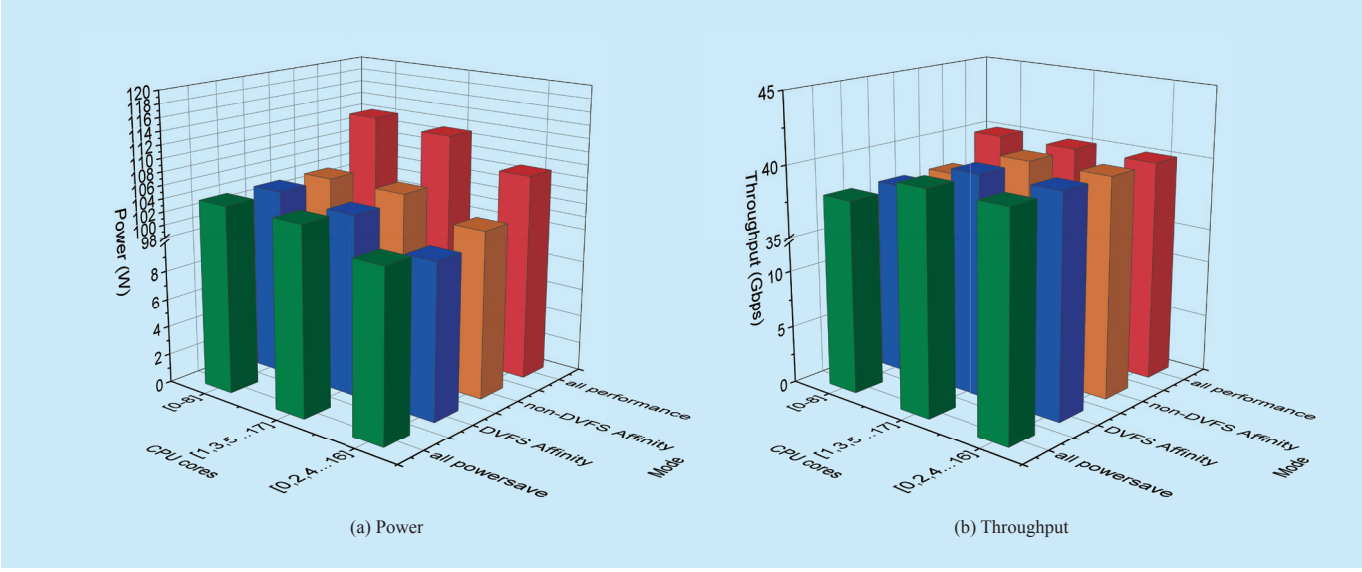
9-47 in 'Powersave' mode. The throughput and power tests are shown in figure5(a) and figure5(b).

Afterwards, we set server B as a packet generator with rate controlling, server A as a forwarding platform. In a real network, the speed rate of packet traffic is dynamic, such as busy time and leisure time. However, the DPDK PMD mode will keep cores in a high speed working status. Hence, we adopt a fine-grained controlling mechanism in every port on the forwarding platform, as shown in figure6. According to the traffic rate of every port, we set the cores which mapped to the port, in a dynamic working status. The results are shown in figure7, our algorithm is more power-efficient.

If we utilize a traditional default solution, then all the CPU cores are in performance mode and cores [0-8] are set in affinity. Meanwhile, the CPU working power is 112 W and the throughput is the line rate. When using the AFFCON solution, the CPU working power decreased to 99 W and the throughput is still the line rate. At the same time, the latency drops from 61 to 34 micro-seconds (usec).

V. Conclusion

We have investigated the effect of CPU Affinity on NUMA architecture server and packet generator, regarding the improvement of per-



(a) Power (b) Throughput

Fig. 5. *Test with AFFCON.*

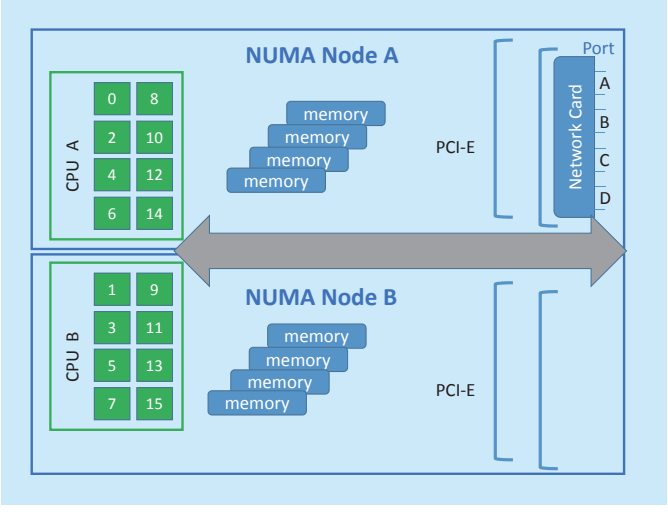


Fig. 6. *Set fine-grained CPU affinity to every port.*

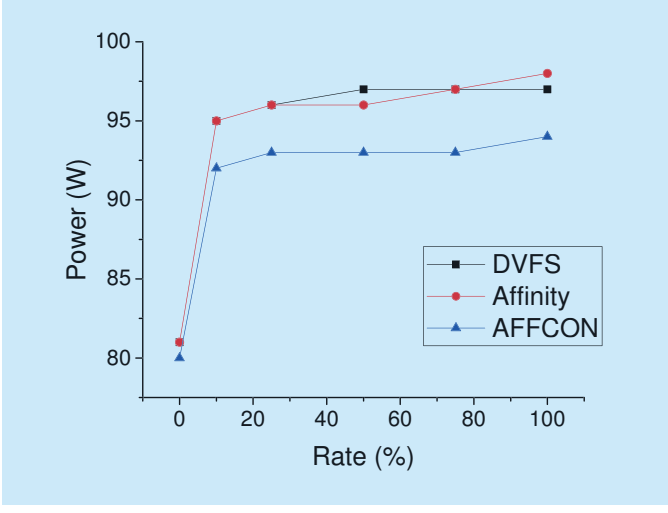


Fig. 7. *Power with packet sending rate.*

formance and reduction of energy consumption. We propose the AFFCON solution, which includes the four components of CPU Affinity, NUMA Affinity, DVFS Affinity and Rate Affinity. The AFFCON turns on the manual operation of the CPU, with the characteristics of the multi-core NUMA server architecture. We select logical cores pinned in the same NUMA node with a hard affinity mode.

The experimental results show that the AFFCON can provide savings in CPU energy resources by up to 11% and reduce latency up to a maximum of 44%, while the throughput maintains a high-performance line rate. In future network testbeds, network slice will share CPU cores to different test applications. Our CPU affinity-oriented fine-grained controlling scheme has an ability to adjust the cores' working status dynamically, which is beneficial for performance improvement and energy preservation.

Acknowledgement

This work was supported by the National Science Foundation of China (No. 61472130, Research on Graphic Processing Units-based High-performance Packet Processing), and the China Postdoctoral Science Foundation fund-

ed project (No. 61702174)

References

[1] J. Nilsson, "Cost-effective packet generation for performance evaluation of network equipment," *Umeå universitet*, 2016, pp. 1-8.

[2] X. Wu, P. Li, Y. Ran, et al. "Network measurement for 100Gbps links using multicore processors," *Future Generation Computer Systems*, 2016, pp. 1-10.

[3] S. Gallenmüller, P. Emmerich, D. Raumer, et al. "Moongen: Software packet generation for 10 Gbit and beyond," *Proc. USENIX NSDI*, 2015, pp. 1.

[4] D. Turull, P. Sjödin, R. Olsson. "Pktgen: Measuring performance on high speed networks," *Computer Communications*, vol. 82, no. 5, 2016, pp. 39-48.

[5] S. Laki, D. Horpácsi, P. Vörös, et al. "High speed packet forwarding compiled from protocol independent data plane specifications," *Proc. ACM SIGCOMM*, 2016, pp. 629-630.

[6] K. Xie, X. Wang, J. Wen, J. Cao, "Cooperative Routing with Relay Assignment in Multi-radio Multihop Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, 2016, pp. 859-872.

[7] P. Emmerich, S. Gallenmüller, D. Raumer, et al. "Moongen: a scriptable high-speed packet generator," *Proc. ACM IMC*, 2015, pp. 275-287.

[8] Z. Xu. "Packetusher: A dpdk-based packet i/o engine for commodity pc," *Proc. IEEE ICCC*, 2016, pp.1-5.

[9] S. Gallenmuller, P Emmerich, F. Wohlfart. "Comparison of frameworks for high-performance packet io," *Proc. IEEE ANCS*, 2015, pp. 29-38.

[10] A. Wang, Y. Guo, F. Hao, et al. "Umon: Flexible and fine grained traffic monitoring in open vswitch," *Proc. ACM CoNEXT*, 2015, pp. 15.

[11] B. Hirschman, P. Mehta, K. Ramia, et al. "High-performance evolved packet core signaling and bearer processing on general-purpose processors," *IEEE Network*, vol. 29, no. 3, 2015, pp. 6-14.

[12] L. Csikor, M. Szalay, B. Sonkoly, et al. "Nfpa: Network function performance analyser," *IEEE NFV-SDN*, 2015, pp. 15-17.

[13] K. Xie, J. Cao, X. Wang, J. Wen. "Optimal Resource Allocation for Reliable and Energy Efficient Cooperative Communications," *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, 2013, pp. 4994-5007.

[14] K. Li. "Power and performance management for parallel computations in clouds and data centers," *Journal of Computer and System Sciences*, vol. 82, no. 2, 2016, pp.174-190.

[15] K. Li. "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, 2016, pp.122-137.

[16] R. Olsson. "Pktgen the linux packet generator," *In Proceedings of the Linux Symposium*, Ottawa, Canada, vol. 2, 2005, pp. 11-24.

[17] L. Rizzo. "Netmap: a novel framework for fast packet i/o," *Proc. USENIX Security*, 2012, pp. 101-112.

[18] H. Redžović, M. Vesović, A. Smiljanić, et al, "Energy-efficient network processing based on netmap framework," *Electronics Letters*, vol.53, no. 6, 2017, pp. 407-409.

Biographies

Guo Li, PhD student of Hunan University. His research interests include DPDK, GPU, and social network. Email: liguo@hnu.edu.cn

Dafang Zhang, professor and PhD supervisor in Hunan University. His main research interests include dependable systems/networks, network security and software engineering. Email: dfzhang@hnu.edu.cn

Yanbiao Li, post-doctoral of Hunan University. His main research interests include IP lookup, Software Defined Network and Named Data Networking. Email: lybmath_cs@hnu.edu.cn

Keqin Li, distinguished professor in the State University of New York. His research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, mobile computing, service computing, Internet of things and cyber-physical systems. He is currently or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, the IEEE Transactions on Sustainable Computing. He is a fellow of the IEEE. Email: lik@newpaltz.edu