TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.4, April 2014, pp. 2950 ~ 2960
DOI: http://dx.doi.org/10.11591/telkomnika.v12i4.4811

Design of Multi-core Processor Software with Pipelining Strategy

Jian Zhang
Zhejiang Business College, Hangzhou, 310053
E-mail: 425145773@qq.com

Abstract

At present, in the field of application of numerous IT, multi-core processors are widely used, the hardware development is much faster than the speed of software development. Multi-core processor software is using the design concept of single-core processor software, so this software is not high-performance and the performance processor is not fully played. This paper firstly introduces the traditional software design method based on multiple cores, and then points out two problems existing in it.To improve its efficiency, I propose a method called pipeline which can improve processor utilization and the throughput of a processor, at the same time, reduce the cost of the system. This method has been verified in the network packet processing.

Keywords: *multi-core, software, pipelining, software architecture, spinlock, CAVIUM*

1. Introduction

Multi-core processor is named for chip multiprocessors (also known as chip multi-processor, CMP), or a single chip multiprocessor. Since 1996 the United States Stanford University first proposed chip multiprocessor (CMP) thought and the prototype multi-core structure, IBM launched the first commercial multi-core processor POWER4 in 2001, and then it is widely used by Intel and AMD in 2005, nowadays it becomes the market mainstream. So it seems multiprocessor has experienced the development more than ten years. In this process, many fields of application have applied multi-core processor, these fields contain the multimedia computing, embedded devices, personal computers, commercial services and high performance computer and multi-core technology, in the mean time the related research is developing rapidly.

With the growing use of multi-core chips, software design concepts began to change. Because application mode has been changed to parallel in multi-core environment, the traditional view of the design would encounter problems. Using traditional sequential control and loop control mode cannot play a multi-core chip performance. Especially in the field of data communications, the performance is particularly sensitive. Now most of the research is in the software field, there is not a set of complete method in the field of data communication, resulting in not fully-used multi-core processor rate. In the speed sensitive scenario, increasing the hardware investment is the only method to achieve the requirement. In fact, it increases the economic cost of the product and development cycle.

The paper research the traditional software design method for multi core, and then points out two problems existing in it. To improve its efficiency,, I propose a method called pipeline, which can optimize operation efficiency. According the test results, it proves that the method can improve the processor utilization, and can improve the processor throughput, reduces the cost of the system, this method has been preliminary verified in the network message processing.

2. Supporting for Multi-Core Design with the Traditional Method and Its Existing Problems

2.1. Traditional Design Method

The traditional software design method mainly refers to the structured development method. The basic point is top-down design, stepwise refinement, modular design, and

Received September 6, 2013; Revised November 11, 2013; Accepted November 23, 2013

structured programming. The key point is that solving a complex problem in stages, each stage of processing problems are in control, and it is easy to understand and handle.

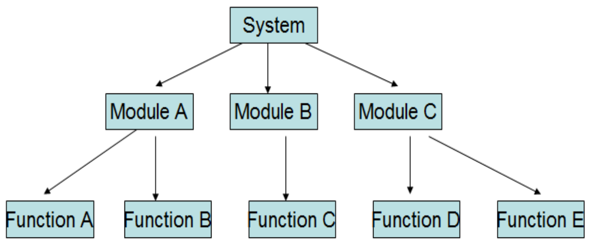

Figure 1. Raditional Design Method

The top-down method is splitting the complex problem into small one, summarizing the key points and important points, and finding out the difficulties. Then describing the problem qualitative and quantitatively. The key points is decomposing, the methods of realizing are the functions and modules. Stepwise refinement is that abstract the real world problems into logical space or the solution space. After the abstraction of complex problems into simpler problems, finally get the simple programming problem in solution domain. The modular design is that dividing the program into several modules, and each module do a sub function, the module together constitutes an organic whole part, and can complete the specified function. The purpose of the structured programming is to reduce the complexity of software, making the software design, debugging and maintenance operation simple. The structured programming also includes the using of the standard library function, using meaningful variable and meaningful function, coupling between modules is clear, making the large program into small modules which is easy to write and test etc.

2.2. Supporting for Multi-core Design with the Traditional Method

Structured design, among the most traditional design methods, is a mainstream method. It is suitable for the overall software system design. The basic principle is to divide system into modules which contain input, output and processing function for calling used modules.

The principle supporting the multi-core processor design with traditional design method is relatively simple. The method is that running the same software in each CORE, the executed software in the CORE is same as single-core software. Moreover, with this method it needs some mechanism to deal with the problems in the multi-core software. Each CORE of multi-core processor is fair, it has the same chance to operate the external input (discussed in this article, this message is the external input), after getting the external input and operating some steps, multiple CORE may enter the critical section which is a shared resource that only one process can enter into, if multi-core enters the section at the same time, the consistence is broken. In the single-core process, lock is used to protect the critical section, being similar with the single-core processor, spin-lock is used to protect the critical section in multi-core. As described in Figure 1, each core of the multi-core processors perform the same processing, processing steps includes from Step1 to Step4. Where Step3 is the critical region, only one CORE can get the right to do the operation at the same time, it needs spin lock to protect Step3, when CORE1 is entering into Step3, it needs to acquire the lock firstly. At this time, if CORE2 also needs to handle Step3, it cannot acquire the spin lock as CORE1 have not release it. So it needs to wait for CORE1 to release the spin lock, after the spin lock is released by CORE1, CORE2 could get the spin lock, and then process continuously.

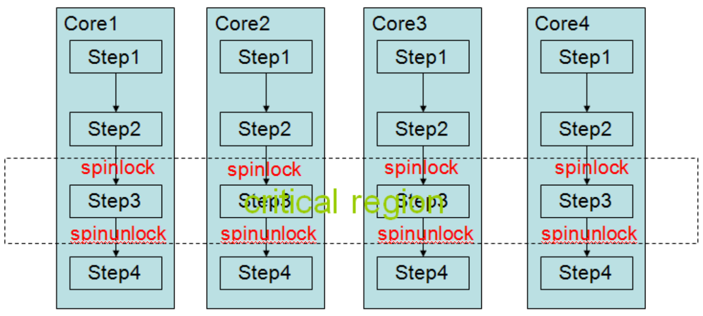

Figure 2. Design Multi-core Program with Traditional Method

3.3. The Problem in the Traditional Multi-core Software Design Method

The traditional design method is easy to understand, and currently it is also widely used in multi-core routers and multi-core network controller. With the development of process and the intensive application, the disadvantage of the traditional design method is gradually exposed. It is analyzed that two problems are existing in the traditional design method. I will add some background knowledge to demonstrate these problems simply. In general, when the multi-core process is applied in the field of data communication, the core would be divided into two classes, one class is responsible for the control function (like operation administration and maintenance function), and the other class is for the data processing. The control part COREs are generally running general-purpose operating systems, like Linux and VxWorks, and the number of COREs on each class can be configured with the configuration file. In addition, the data processing COREs designed by itself run business service and it always runs the simple operation system. We cannot make big changes on general purpose operating and so then do some optimization for multi-core processor, therefore we make the analysis of the problems on the data service COREs.

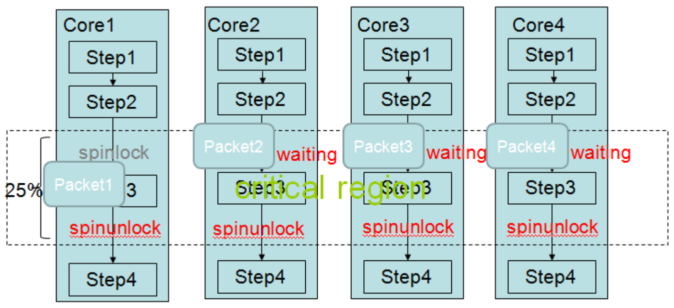

Figure 3. The Existing Problem in Traditional Algorithm

Problem a: the throughput of the system does not linear grow with the increasing CORE number.

In the development of this product, we find that the throughput of the system is not achieving the requirement of system, so the direct solution is to optimize the design of our code and improve the relevant algorithm. However, this improvement may have some limitations. Then, reducing the number of the core used by linux, and using more cores to deal with business logic is another method, as a result of unexpected performance. For example, if 4 cores can process 1M packets per second, 8 cores are not able to process 2M packets per second with a very large gap. Hence the only method is to upgrade the hardware, the truth known by embedded software engineers is that cost for upgrading the hardware is much more than other methods, this cost includes the time cost and the economic cost. Let us continue to find the answer of this question: why the performance is increased linear? Refer to Figure 2, step3 is the critical region, assuming that it occupies 25% processing time of the total, and the entire processing cycle is 100ms assumedly, so this step requires 25ms. The consumption of processing first core, second core, third core and fourth core is reported to be 100ms, 125ms, 150ms and 175ms respectively. As the data showed, the average time of processing a packet is 137.5ms, with the core number growth, the average processing time will increase accordingly. If configured with 8 cores, the average processing time will reach 187.5ms. A packet needs 50ms more than configuration with 4 cores, the processing time increases about 30%. It is more costing.

Problem b: packet delay is increased.

As the average processing time increases, the average delay time is added in the mean time. In the previous instance, the delay for each packet through the processor would increase by 30% in the time-sensitive application, which is not acceptable. In this scenario, message delay is increased, so the message ordering between packets become uncertain, and the packet jitter becomes larger, if the timeout is larger than a certain value, the message in application layer would be lead to discard, resulting in retransmission, increasing the network burden. In some real-time applications, if the control packet delay is too large, it will lead network slow convergence, if the important data packet delay is too large, the user experience would be poor as the delay is increased with 30%. For example, downloading a map is 60 seconds originally, now it is increased to 78 seconds.

3. Propose a New Method

3.1. Pipeline Algorithm Principle and Characteristics

First example is processor pipeline technology. We are quite familiar with the processor pipeline technology, in order to achieve instruction pipelining, we must divide the whole operation instructions into several sub function. Roughly we could decompose the whole instruction into the instruction fetch stage, decoding stage and implementation stage. The whole operation stage of an instruction is decomposed into smaller stages, so that multiple instruction could run parallel in time, called time overlap.

Another example, we use network video player which is popular currently as an example. Online video player program broadcast video generally consists of 4 tasks: Task 1 is to download data from the network to the hard disc, task 2 is to read data from the hard disk, task 3 is decoded data; task 4 is played with peripheral equipment. You can easily find that the task 1's output data is task 2's input data, the task 2's output data is task 3's input data, the task 3's output data is task 4's input data. The share structure of continuous data is very similar to a microprocessor in a pipelined architecture. If each task can be implemented as a thread, it is in connection with the pipeline architecture, through the thread parallel to accelerate the program, then the pipelining algorithm in this paper is introduced.

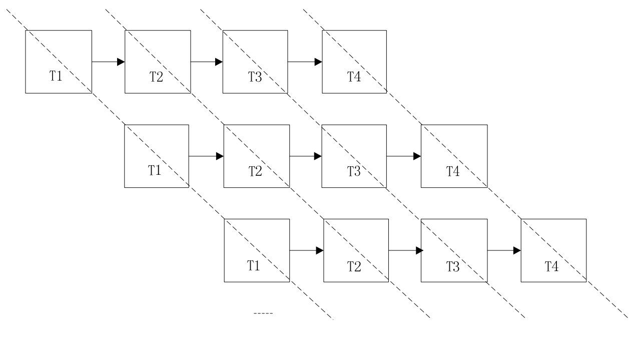

Figure 4. Pipelining Example

Assume that the thread T1 and T2 and T3 and T4 execution time are respectively t1and t2 and t3 and t4. If pipelining algorithm isn't used, then the total execution time is tnp=t1+t2+t3+t4, if pipelining algorithm is used, then the total execution time is tp= max{t1, t2, t3, t4}, as tnp would small than tp, so the pipelining algorithm can accelerate the speed, and the throughput is improved.

The basic principle of pipelining algorithm is a complex process is decomposed into several sub processes, each process can be run with other sub process at the same time, the function is decomposed, it runs in order in space, temporally overlapping parallel. It has the following characteristics:

a) a number of repeated running tasks.
b) each task can be decomposed into multiple sub tasks.
c) many functional components are used.
d) different sub tasks do different tasks.
e) the plurality of functional components running in parallel.

We are quite familiar with the processor pipeline technology, in order to achieve instruction pipelining, we must divide the whole operation instructions into several sub function. Roughly we could decompose the whole instruction into the instruction fetch stage, decoding stage and implementation stage. The whole operation stage of an instruction is decomposed into smaller stages, so that multiple instruction could run parallel in time, called time overlap.

Considering the above design carefully, we found that Core1 is in the operation of Step3, other cores may be pending on the lock, in fact it is idle, and it does not doing any business, but you cannot remove the spin lock out. In the actual system, the scenario is more complex, so there may involve more critical areas.

3.2. Basic Idea

The processor performance would be more decreased, are there any methods to get rid of these critical areas? This problem also exists in Linux, because of this too complex operating system function, a lot of methods are considered. The principal is to make infinitely narrow critical section improve performance. There is no universal method to fix such problems, reducing the impact of the critical region is the only way. In our application, the bottleneck of this performance is in data processing plane, the characteristics in data processing is that the logical function is not very complicated. According to this characteristic, this paper proposes a new method that using pipeline to make the code design, it can reduce the influence of the critical zone, then improves the performance of the system.

Only one core can enter into the critical section, so we just use one core to handle the code of critical section. Refer to the design idea of RISC instructions, each core operate only one step, after processing the previous step it posts the packets into the next core. That means each CORE is just doing one thing and does not need to wait for each other.

3.3. Detailed Design

The pipelining design method has been used in the design of RISC commands, the process is as below. Core1 only runs step1 program, core2 only runs step2 program, after core1 processed the packet, it forwards the packet to core2 directly, so each core only runs one module codes. Critical section existed in the traditional design, the step3 code only run in core3, so core3 does not need to add lock to do the protection. Compared with the traditional design method, the cost happens at the time of packets entering into the pipeline and flushing pipeline without waiting time. It costs small, when the time is at the begin of sending packets and the last few packets processing.

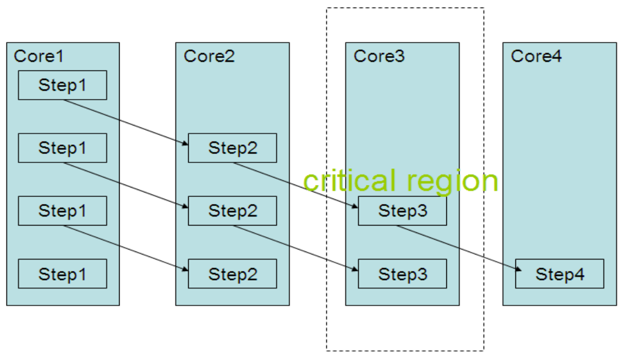

Figure 5. Design with Pipelining

In the high-speed processing, this overhead could be ignored. Also, some problems are considered in the code design. These problems are communication between cores, modules allocation, deployment and supporting for the debugging.

1. The communication between multiple cores

Now the multi-core processors would provide a communication mechanism between cores. Because multiple cores are in a chip, the communication speed is very fast. Take the CAVIUM processor as an example (other processors are similar). POW is one module of CAVIUM, which provides queuing function, de-schedule and submit functions. The process is showed as below, the cores dealing with the module are divided into the same core group, after the previous cores process the packets, they change the core group property in the packets, and then submit the packets to POW. Another cores with the core group could receive the packets to process further, step by step, when the last module finishes processing the data packet, it then decides whether to send the packet out or drop it. Of course, each module can determine whether the message is discarded, it depends on the module function.

When the previous cores process the packet, it directly copies the packets to the next cores, the data size would be large with more time. As multiple cores are in one chip, all of the memory space can be accessed by all cores, so pointer of data packets could be used to pass the data, or just forward the packet header. Certainly, when a packet enters into the first core, summary information for the packet could be generated, and then the summary information could be forwarded between multiple cores, the later cores process the packets accordingly.

2. Modules allocation and deployment

The principle for module allocation depends on the processing time of each module and the size of critical section. Try to get better performance, the best solution is to keep the processing time of each core with the same size. The bright reader may find the problem, if there are 4 modules in the program and the processor has 32 cores, only 4 cores are used, the other 28 cores are wasted. The method mentioned below could solve the problem.

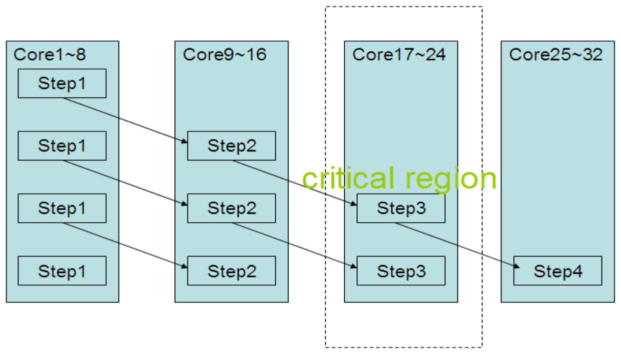

Figure 6. Mapping between Cores and Modules

Multiple cores could be configured in the same core group whose properties are same, and they just process one module with same opportunity to process the packets. Certainly, the module may contain the critical section, spin lock could be used to protect the section. As the cores are less to run the section, the cost is less. There is another kind of method, which divides the modules into sub-modules, let one core process one sub-module. It could remove the critical section. Such method could be used, it relies on the requirements of performance, the usage of scenarios and the results of test, it is balanced in the application.

3. Debugging module

No matter what kind of design method, program debugging functions need to be supported. Basic debugging method includes GDB and LOG system. Compared with the traditional design method, the modules of pipelining design method are deployed to each CORE, the complexity of the modules on each cores are decreased, so that the probability of problem will be greatly reduced. GDB debugging function is the same as traditional method, transplantation of GDB Server module into every core. About the system log, the mapping between module and core could be designed, each CORE can print the module debugging information and core information into memory, shared memory mechanism could be provided by the system, in the control core, diagnostic command is provided to print this diagnostic information, observing operation running information and the tracking of packets. In general, the cost for problems debugging will greatly reduced.

4. Networking Packet Processing Based on the Pipelining Design Algorithm

4.1. Networking Packet Processing

Network packet processing is generally divided into Ingress and Egress directions, two directions also can be divided into the following several module, data parser module, data processing module, data action and strategy processing module, data packet flow control and measurement and statistics module. As shown below:

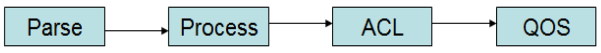

Figure 7. Networking Packet Processing

The data packet parser module is used to pick from a variety of sources of data stream to obtain various specified field, called the match fields, for example: packet source MAC address, destination MAC address, source IP field, before the message processing, packet parsing is needed for identification and classification.

The data processing module have different processing procedure in the different protocols and services, such as fragmentation and reassembly of IP message, message caching, message routing lookup etc.

Data action and strategies module is that using the strategy table and the matching rules to do packet matching, when a matching rule is found, the action is in the rule. Supported actions include: forwarding, discarded, redirection, mirror, change the message priority and so on, there is a great difference between the different products.

Traffic quality control and statistics module is mainly in order to guarantee the quality of service, let the important message pass through with high priority, an important message mainly includes protocol control message and important service message, statistics is mainly generated some statistical report, it offers dynamic real-time observation for treatment.

4.2. Networking Packet Processing with Traditional Design Algorithm

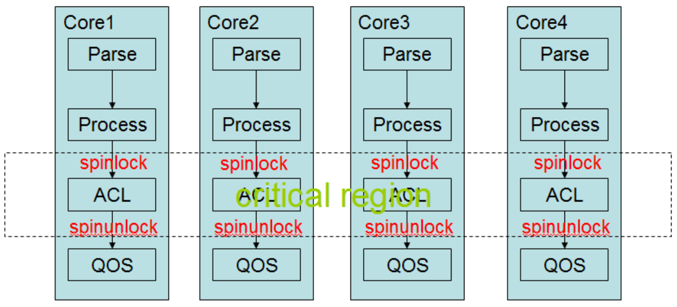

Figure 8. The Traditional Design Algorithm

Design the networking packet processing is as above, every modules runs on the same core, and every call runs all modules, when packet is gotten via interface, it enters into parse, process, ACL and QOS stage in the same core. So maybe the ACL process stage need to be protected, so spinlock is used in it, yet the operation will be pended on the stage, so the performance is impacted.

4.3. Networking Packet Processing with the Pipelining Design Algorithm

Compared with the traditional method, the pipelining design algorithm for networking process is as above, parse, process, ACL and QOS module are run on different cores, every core only do one module function, so no waiting in ACL stage. Pipeline processing is to reduce coupling, it reduces the influence of the spin-lock. According to the business module above, and the real module that needs spin-lock, these modules are put in the same cores to reduce waiting time, so that each core only responsible for their own modules, as shown above, the data packet in the first core is parsed, then the packets are sent to the next core to do packet processing, and so on, the core after continue to do action and strategy, final core do the flow control and statistical processing.

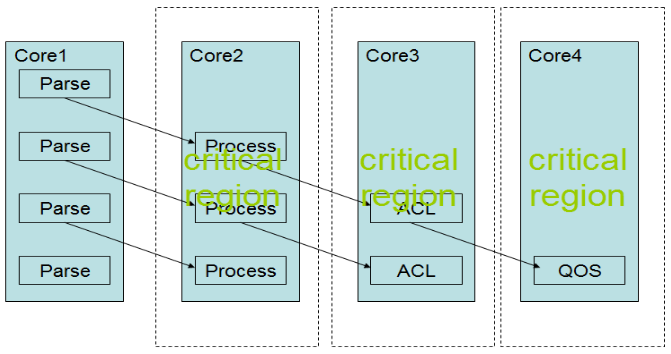

Figure 9. The Pipelining Design Algorithm

5. Testing Results and Analysis

In this project researched in the paper, two OCTEON processors are used, one is 8-core processor, and another is 32-core processor, the traditional design method is used at the beginning, then the pipeline design method is used to improve the performance, the following testing report is comparison and analysis of the two methods. The testing contains packet delay and system throughput. The basic function of the test is the same as the traditional processing method, it isn't includes in below testing. Below picture is test bed.

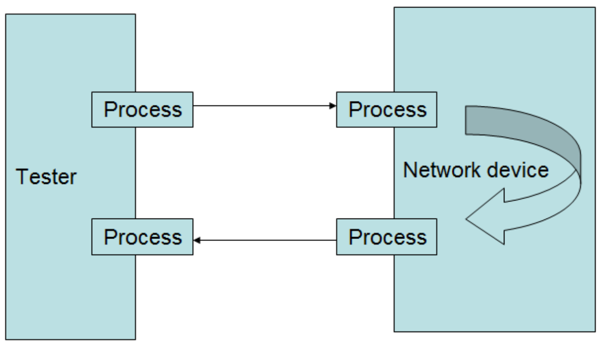

Figure 10. Test Bed

5.1. The Throughput Testing One

8-core processor is used in above testing, the CORE configuration in the testing is that 4 cores is deployment of Linux operating system, which run control plane program, and the other 4 cores is configured to run data processing programs, testing data is run on the data plane. The horizontal axis is each packet size in bytes per seconds; the vertical axis is the number of packets. Testing data covering the packet size, the range is 64 to 1500 bytes. Observing the above figure, the conclusion can be concluded, with the increasing of packet size, packets per second for each method will decrease, this could be understand, if the packet size is larger, it will affect the operation of memory and other external devices, this is normal. In addition, comparing two kinds of design methods, we found that the overall performance can be improved about 30% with pipelining method.

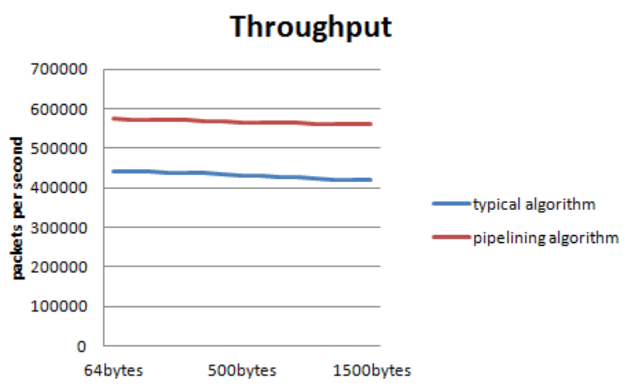

Figure 11. The Throughput Testing One

5.2. The Throughput Testing Two

8-core processor is used in above testing, the CORE configuration in the testing is that 2 cores is deployment of Linux operating system, which run control plane program, and the other 6 cores is configured to run data processing programs, testing data is run on the data plane. The horizontal axis is each packet size in bytes per seconds; the vertical axis is the number of packets. Testing data covering the packet size, the range is 64 to 1500 bytes. Observing the above figure, the conclusion can be concluded, with the increasing of packet size, packets per second for each method will decrease, this could be understand, if the packet size is larger, it will affect the operation of memory and other external devices, this is normal. In addition, comparing two kinds of design methods, we found that the overall performance can be improved about 30% with pipelining method.

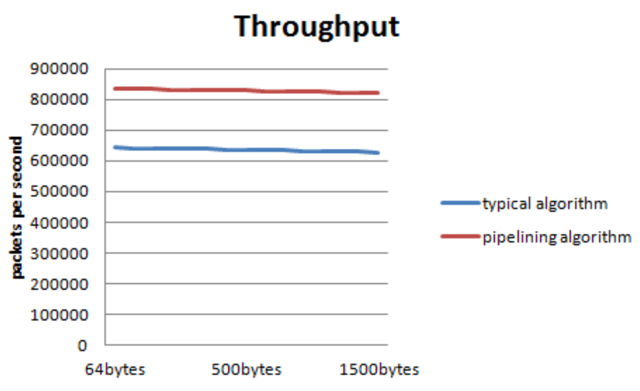

Figure 12. The Throughput Testing Two

5.3. The Throughput Testing Three

32-core processor is used in above testing, the CORE configuration in the testing is that 16 cores is deployment of Linux operating system, which run control plane program, and the other 16 cores is configured to run data processing programs, testing data is run on the data plane. The horizontal axis is each packet size in bytes per second, the vertical axis is the number of packets. Testing data covers the packet size, the range is from 64 to 1500 bytes. Observing the above figure, it is concluded that as the increasing packet size, packets per second for each method will decrease. This could be understandable, if the packet size is larger, it will affect the operation of memory and other external devices normally. In addition, comparing two kinds of design methods, we found that the overall performance can be improved about 30% with pipelining method.

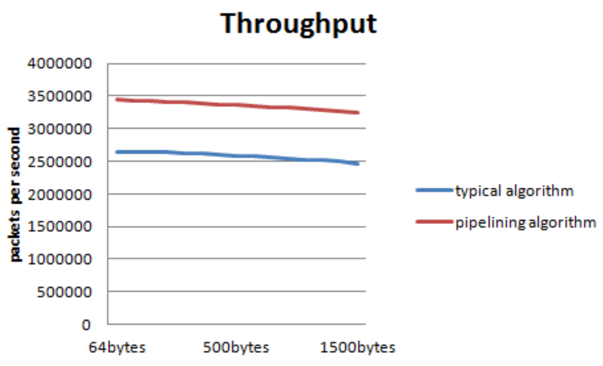

Figure 13. The Throughput Testing Three

5.4. The Throughput Testing Four

The above configuration is the result of testing with 16 cores, we try to modify the configuration with 24 cores, let us look at the results of this test.

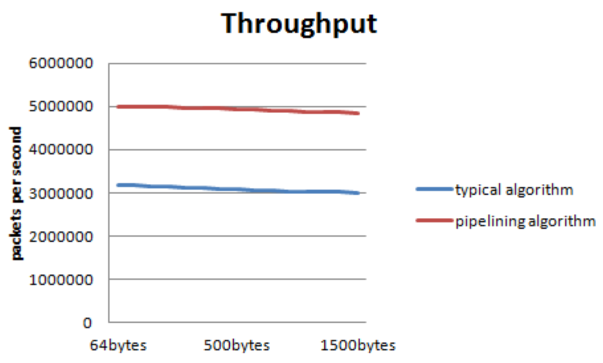

Figure 14. The Throughput Testing Four

According to result of the test, the pipelining design method has excellent performance with 24 CORE configurations. At the same time, as the test results showed, you can probably estimate the different performance between the different configurations. By comparison, with the pipelining design method, performance with 24 cores configuration is about 1.5 times than the performance with 16 cores configuration, probably increased 0.5 times, the performance is increased linear. With the traditional design method, the performance with 24 CORE configurations increased, but the amount of increase is far less than 0.5 times, it cannot achieve the linear increase. At this point, it can prove the pipelining design method being better than the traditional design method.

5.5. Packet Delay Testing

In this application, although packet delay is not very sensitive, it supplies very good test results for us to analyze.

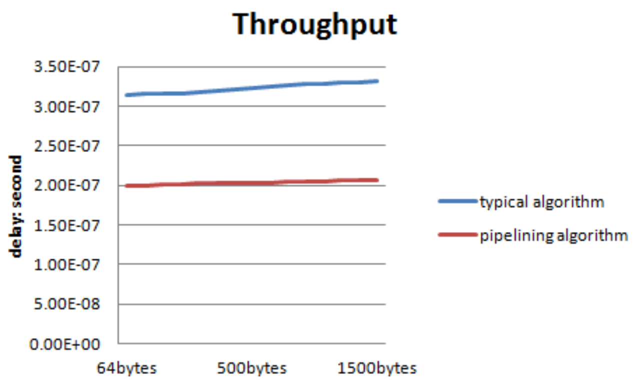

Figure 15. Packet Delay Testing

Similarly, the 24 cores for data plane is configured. In fact, according to the test of previous two cases, the delay time value can probably be calculated. Refer to the test results, the value in testing is consistent with in calculating, pipeline design method has improvement of the packet delay, reducing about 50% delay time. It is a great upgrade that the delay time reduces from 3.00E-07 seconds in traditional method to 2.00E-07 seconds with new algorithm. Therefore, this application which is quite sensitive to the delay should be very suitable for this algorithm.

6. Conclusion

Through the test, the algorithm can improve system performance; and it also flexible from 8 cores to 32 cores; many applications requiring additional hardware investment in the original design could improve the performance by improving the software algorithm. Not only reducing the economic cost of system, but also accelerating the product launch time, enhancing market competitiveness, it has the very big value.

References

[1] Raasch RS. *The impact of Resource Partitioning on SMT Processors.* Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques. New Orleans，USA：IEEE Press. 2003.
[2] Chaudhry S, Caprioli P, Yip S, et al. High-performance Throughtput Computing, *IEEE Micro*. 2005; 25(3):8-15.
[3] Performance of Multithreaded Chip Multiprocessors and Implications [EB OL], http://www.cs.sfu.ca/~fedorova/papers/usenix-fedorova.pdf.2011.
[4] Harchol-Balter M，Downey AB. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*. 1997; 15(3): 253-285.
[5] Yamasaki N, Magaki I. *Itou Prioritized SMT Architecture with IPC Control Method for Rea1．Time Processing.* Proceedings of the 13th IEEE Real Time and Embedded Techn ology and Applications Symposium. Washington DC. USA. IEEE Press. 2007.
[6] Fedorova A, Small C, Nussbaum D. *Chip Multithreading Systems Need a New Operating System Scheduler.* Proceedings of the 11th ACM SIGOPS European Workshop. Leuven. Belgiurm. ACM Press. 2004.
[7] Spracklen L, Abraham SG. *Chip Multithreading: Opportunities and Challenges*. Proceedings of the 1 lthInternational Conference on High-performance Computer Architecture. San Francisco, USA [S.n.]. 2005.
[8] F Sun, Srivaths Ravi, Anand Raghunathan, Nirai K Jha. A synthesis methodology for hybrid custom instruetion and co-processor generation for extensible processors. *IEEE Transactions on CAD of Integrated Circuits and Systems*. 2007; 26(11): 2035-2045.
[9] Seng Lin Shee, Andrea Erdos, Sri Parameswaran. *Heterogeneous multiprocessor implementations for JPEG: a ease study.* CODES+ISSS. 2006: 217-222.
[10] Seng Lin Shee, Sri Parameswamn. *Design Methodology for Pipelined Heterogenous Multiprocessor System.* DAC. 2007: 811-816.