Analysis and Optimization of Parallel Features on Simulation of Self-similar Network Traffic

Sai Sui[1,a], Jing Du[2,b], Yancang Chen[3,c], Pei Wei[4,d]

[1,2,3,4] Luoyang Electronic Equipment Test Center, Luoyang, 471000, China

[a]email: suisai@foxmail.com, [b]email:jdstarry@aliyun.com,

[c]email: yancangchen@gmail.com, [d]email:weipei_uestc@163.com

Keywords: Self-Similar; Traffic Simulation; Parallel

Abstract. Cyberspace security is an important aspect of national security strategy, and cyberspace experiment needs to construct a realistic test environment. In addition to hardware, software and basic service, the network traffic simulation is also an important aspect of test environment construction. This paper focuses on the requirements of constructing a large-scale network test environment, studies the key elements of network traffic simulation, and the parallelism characteristics are analyzed and optimized.

Introduction

In recent years, the development of Internet particular the mobile Internet has experienced an explosive growth, the network size expands rapidly, while the network has generated a larger amount of traffic. Cyberspace has almost surrounded everyone, which is very important for both consumer and provider. The rapid innovation and growth of the network application put forward a higher requirements of transmission and carrying capacity of network. To ensure the performance of network, and make sure the processing capability of network equipment is in a balanced state, in addition to increasing the carrying capacity of bandwidth, another important aspect is the optimization of network management.

Optimization of network management, performance testing of network devices and analysis of network load detecting are in line with the need to build a network test environment with real network features. In addition to hardware, software, and basic services, building and operating of a network test environment also require more of traffic simulation with the actual network characteristics. This article focuses on the requirement of building a large-scale network testing environment, and the key elements of network traffic simulation are researched, while the parallelism characteristics are analyzed and optimized.

Network Traffic Model

With the increasing size of the network and network applications continue to expand, researchers has got a lot of models by studying network traffic characteristics, which can describe features of network traffic, such as renewal model, Markov model, fluid model, linear stochastic model and self-similar traffic model. Paxson and his partner discover that network traffic exhibits self-similar characteristics by observing WAN(wide area network) and applications. Traditional telephone traffic and common packet traffic models contain Markov model and fluid model etc. In contrast, the autocorrelation function of self-similar network traffic model has special damping characteristics, that can describe a very wide burst nature within a large range of time. In such many models, the self-similar traffic model is the main researched model due to that can truly reflect the observed macroscopic properties of some modern network traffic, such as long-related and self-similar properties.

A. *Self-Similar Network Traffic Model*

Self-similar refers to the consistency between local structure and the overall structure with both space and time. Its physical meaning specific to the computer network traffic, refers to the statistical characteristics of similar nature in different time scales.

Self-similar process is divided into continuous-time and discrete-time self-similar process[1]. Continuous-time self-similar process is defined as follows:

When a> 0, the random process X(t) and $a^{-H}X(at)$ with the same distribution, can be expressed as:$\{X(t_1), X(t_2), \ldots, X(t_n)\} \sim \{a^{-H}X(at_1), a^{-H}X(at_2), \ldots, a^{-H}X(at_n)\}$

Among which, the statistical self-similarity should have the following features:

Mean: $\mu = E[X(t)] = a^{-H}E[X(at)]$

Variance: $\mu = E[X(t)] = a^{-H}E[X(at)]$

Autocorrelation function: $R(k, T_i) = a^{-2H}R(at, a\tau)$

Discrete time self-similar process is defined as follows:

Suppose a time series $X = \{X_n, n \in Z^+\}$, represent a trace of traffic in fixed time granularity. Define: $X^{(m)} = \{X_n^{(m)}, n \in Z^+\}$ is a m order process of polymerization,

There $X_n^{(m)} = \frac{1}{m}\sum_{i=nm-(m-1)}^{nm} X_i$

There are two self-similar processes of discrete-time, certainty self-similar process, and gradual self-similar process.

If the variance of process X: $Var\left[X^{(m)}\right] = \frac{Var[X]}{m^\beta}$, $\beta = 2(1-H), 0 < \beta < 1, m \in Z^+$

Autocorrelation function: $R\left(k, X^{(m)}\right) = R(k, X)$,

Which called deterministic self-similar process.

If the variance of process X: $Var\left[X^{(m)}\right] = \frac{Var[X]}{m^\beta}$

Autocorrelation function: $m \to \infty$, $R\left(k, X^{(m)}\right) \to R(k, X)$, this self-similar process called progressive self-similar process.

Wherein, H represents self-similarity parameters, the value between (0.5, 1] represents a self-similar characteristics.

Self-similar flow model has a variety of different forms, including fractional brown motion model, fractional Gaussian noise model, wavelets model, ON/OFF processes model, Poisson-Zeta processes models[2]. Wherein, ON/OFF process model is very popular, because it is simple and practical, easy to control and can better reflect the characteristics of self-similarity.

B. *ON/OFF Traffic Models*

Consider N independent data sources $X_i(t)$, i ϵ [1,N], each data source is an renewal reward process, and with an independently and identically distributed ON/OFF cycle. $X_i(t)$ alternately generate a value of 0 or 1, corresponding ON or OFF state, such non-overlapping time intervals are called the ON period and the OFF period. $X_i(t) = 1$ means that send a packet at time t, therefore, an ON period can be considered to form a packet column. And $X_i(t) = 0$ does not transmit any data, it is idle. At time t, N such data packets overlap and can be expressed as: $S_N(t) = \sum_{i=1}^{N} X_i(t)$, the synthesis flow shown in Figure 1.

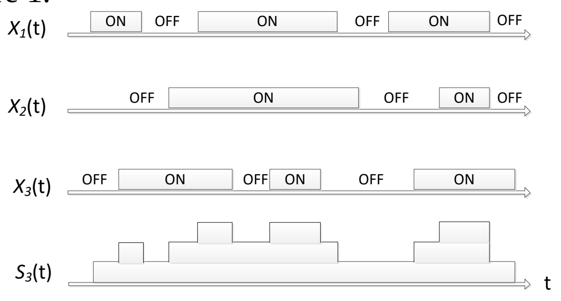

Fig. 1. N = 3 ON/OFF data sources superimpose flow

Let T is renormalization time factor, and the process of packet overlapping in [0, Tt] is as follows:

$Y_N(Tt) = \int_0^{Tt}\left[\sum_{i=1}^{N} X_i(u)\right]du$

For sufficiently large N and T, the statistical properties of data packet overlap process

$Y_N(Tt), t \geq 0$, which is depended on the distribution of ON/OFF cycles. Studies show that, N independent and identically distributed ON/OFF source overlapped to form a self-similar traffic, and the self-similarity index H is only determined by the shape of the data source parameters [3].

Parallel Programming Model

Parallel is a broad concept, depending on the implementation of different levels, that can be divided into several levels. As shown in Figure 2, the most micro level is single-core instruction level parallelism (ILP), in which a single processor execution unit can execute multiple instructions at the same time. The next higher level is multi-core parallel, that in a chip integrated of multi-cores to achieve thread-level parallelism (TLP). The next higher level is multi-processor parallel, that multi-processors is installed on a circuit board to the thread-level and process-level parallelism. Finally, networks can be used to achieve large-scale clusters or distributed parallel, each node is an independent computer, by which can achieve a more massive parallel computing.

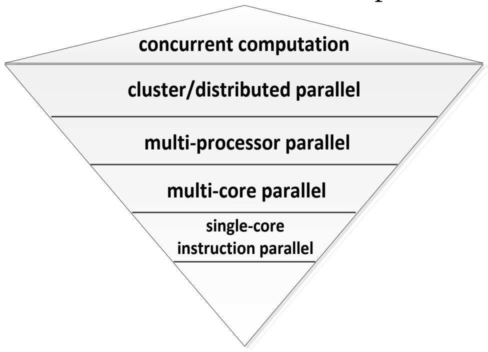

Fig. 2. parallel hierarchical graph

A. *Analysis of Parallel Characteristics*

The network traffic model based on the ON/OFF process can be more self-similar with more nodes and longer simulation time. The main parameters of the ON/OFF model are packet size, simulation time, and number of data sources. From the software implementation, the three main parameters are defined as packet size (PS), inter departure time (IDT) and simulation node N. If the simulation time is very long, the range of simulation time t is very large, which take up more storage space, and the equivalent parameter IDT (t_i-t_{i-1}) can reduce the range of parameters, and that is very effective to reduce the amount of storage space. Because the network traffic simulation program has many simulation nodes and can run in the distributed environment, it can realize the distributed parallel simulation. In addition, when the simulation time is relatively long, the calculation of PS and IDT parameters is very large, the instruction level parallel optimization is adopted to improve the program performance. The instruction-level parallel optimization design of large-scale parameter calculation is carried out based on the GPU platform.

B. *GPU Architecture and CUDA Programming Model*

GPU uses large amount of cores with SIMD(single Instruction Multiple Data) parallel processing[4], which is easy to implement parallel execution of multiple threads or tasks that can fully exploit the parallelism of hardware resources for calculation program of large-scale networks traffic models.

NIVIDIA launches the general parallel computing architecture CUDA that the CPU is master controller, and GPU is data processor as well as co-processor, and CPU and GPU collaborative to complete a task. CPU handles strong logic and serial computing-related work, and GPU mainly handles work with highly parallel data processing among threads[5-8].

CUDA can run and manage multi-threads through the three level to manage these threads. A certain number of threads constitute thread block, and a certain number of thread blocks organized into one or two dimensional thread blocks Grid. The execute process of CUDA program is shown in Figure 3.

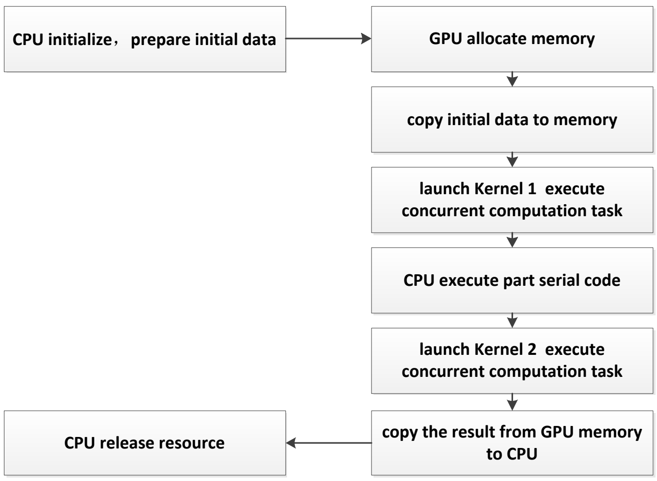
CPU initialize，prepare initial data
GPU allocate memory
copy initial data to memory
launch Kernel 1 execute concurrent computation task
CPU execute part serial code
launch Kernel 2 execute concurrent computation task
copy the result from GPU memory to CPU
CPU release resource

Fig. 3. Execute process of CUDA program

Program Design and Test

For large-scale network traffic simulation, we extract key parameters to IDT(inter departure time) and PS(packet size) based on ON/OFF traffic model, and considering the actual effect of the network users combining with group behavior modeling methods. We experiment and test the acceleration performance of parallel computing program through the generation of normal distributed random parameters.

Normal parameters can be generated by a variety of methods: generating by uniform distributed random number, by Box-Muller method, by a sinusoidal graphics and so on. Here are analyzed and tested for common Box-Muller method.

A. *Parametric Analysis*

Box-Muller method uses two groups of uniform distributed random numbers U1, U2 in (0,1], to generate a standard normal random number, and then calculating to produce a normal distribution with any mean and variance. The procedures are as follows:

(1) Generate uniform distributed random numbers in (0,1];

(2) Using the Box-Muller method to generate standard normal random numbers;

(3) Calculating to produce a normal distribution with any mean and variance.

B. *Test Results*

We have tested the performance of the simulation program on the NVIDIA NVS540 GPU platform. The platform has two multiprocessors (MP), each multiprocessor has 48 CUDA cores and 1536 physical threads, and thus a single instruction can operate up to 3072 (1536 × 2) physical threads, which can achieve instruction-level parallel optimization of single-instruction multiple data(SIMD).

When the size of the normal parameters for the calculation is n, the consuming time of program calculating is t, statistical data is shown in Figure 4, Figure 5, Figure 6 and Figure 7:

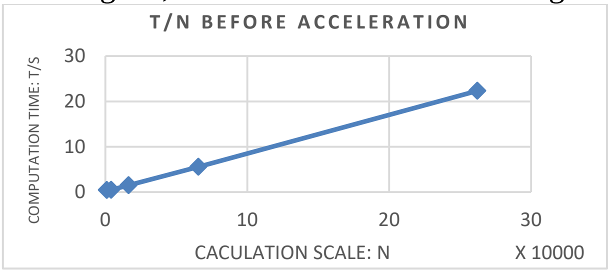

Fig. 4. Figure of t/n before acceleration

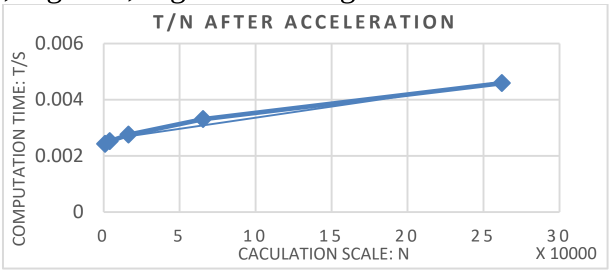

Fig. 5. Figure of t/n after accelerating

As shown in Figure 4, with the increase of the calculation scale n, computation time t is gradually increased. Overall, the computation time is linearly increasing with the expansion of computing scale, and when the calculate size increased to a certain size (such as 10^4 or more), the calculation time has reached the second level. As shown in Figure 5, with the increase of the calculation scale n, computation time t is gradually increased. Overall, the computation time is linearly increasing with the expansion of computing scale, but the increase trend is relatively more slowly, and when the calculate size increased to a considerable size (such as 10^5 or more), the calculation time is still the order of milliseconds.

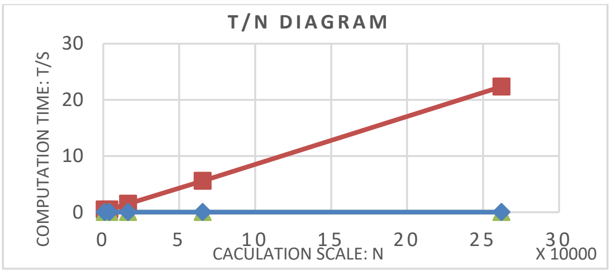

Fig. 6. T/N diagram

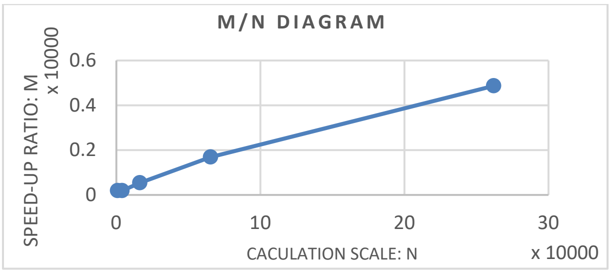

Fig. 7. M/N diagram

In comparison, the computation time of program after GPU accelerated increases relatively slow and the consumed time of calculating is less of three orders of magnitude, and the acceleration performance is very obvious, as shown in the figure 6, Figure 7. Therefore, when the computing scale is large enough, the computing performance of program after parallel optimization with GPU can be greatly improved.

Conclusion

As the network traffic in large-scale network environment shows the self-similar characteristics, study of self-similar network traffic simulation technology is an important support of building large-scale network test environment. Due to the many distributed simulation nodes and long simulation time, the large-scale network traffic simulation shows strong parallel characteristics. At present, more and more network terminals integrated GPU equipment, carrying out the parallel optimization and design of network traffic simulation based on GPU architecture and CUDA programming model is very practically significance and feasible. Experimental tests have shown that the computing performance of network traffic simulation program is able to achieve a great improvement.

Acknowledgement

In this paper, the research was sponsored by the National Nature Science Foundation of China. (Grant No.61303061)

References

[1] Liu Yan. Several key technologies of network traffic control, Doctoral dissertation of Fudan University, 2004.

[2] Chen Yufeng, Dong Yabo, Lu Dongming. Advances in network traffic simulation, Computer Engineering and Design, 2009.

[3] Wu Zemin, Zheng Shaoren. One kind of empirical self-similar flow simulation algorithms, System Simulation Academic Journal, 2002, 14(1):41-43.

[4] Ding Peng, Chen Lixue Gong Jie, Research of GPU general computing[J]. Computer and Modernization, 2010 (1):12-10.

[5] G. Michael, L. G. Scott, N. John, et al. Parallel Computing Experience with CUDA[J]. IEEE Computer Society, 2008 (1):0272-1732.

[6] Owens J D, Houston M, Luebke D, et al. GPU computing [J]. Proceedings of the IEEE, 2008,96(5):879-899.

[7] Blythe D. Rise of the Graphics Processor[J]. Proceedings of the IEEE, 2008,96(5):761-778

[8] Ogawa S, Aoki T. GPU computing for 2-dimensional incompressible-flow simulation based on multigrid method[C]. Transactions of the Japan Society for Computational Engineering and Science, 2009.