IT'14

ŽABLJAK

XIX

naučno - stručni skup

INFORMACIONE TEHNOLOGIJE

SADAŠNJOST I BUDUĆNOST

Urednik
Božo Krstajić

IT'14

INFORMACIONE TEHNOLOGIJE

- SADAŠNJOST I BUDUĆNOST -

Urednik
Božo Krstajić

Zbornik radova sa XIX naučno - stručnog skupa
INFORMACIONE TEHNOLOGIJE - sadašnjost i budućnost
održanog na Žabljaku od 24. do 28. februara 2014. godine

Zbornik radova
INFORMACIONE TEHNOLOGIJE - sadašnjost i budućnost 2014

Glavni urednik
Prof.dr Božo Krstajić

Izdavač
Univerzitet Crne Gore
Elektrotehnički fakultet
Džordža Vašingtona bb., Podgorica
www.etf.ucg.ac.me

Tehnička obrada
Aleksandra Radulović
Centar Informacionog Sistema
Univerziteta Crne Gore

Tiraž
150

Podgorica 2014.

Organizator

Elektrotehnički fakultet Univerziteta Crne Gore

Skup su podržali:

* Ministarstvo za informaciono društvo i telekomunikacije
* Pošta Crne Gore
* Agencija za elektronske komunikacije i poštansku djelatnost
* Čikom

Programski odbor

Dr Novak Jauković, Elektrotehnički fakultet, Podgorica
Akademik Dr Ljubiša Stanković, CANU
Dr Zdravko Uskoković, Elektrotehnički fakultet, Podgorica
Dr Vujica Lazović, Ekonomski fakultet, Podgorica
Dr Branko Kovačević, Elektrotehnički fakultet, Beograd
Dr Milorad Božić, Elektrotehnički fakultet, Banja Luka
Dr Miroslav Bojović, Elektrotehnički fakultet, Beograd
Dr Zoran Jovanović, Elektrotehnički fakultet, Beograd
Dr Božidar Krstajić, Elektrotehnički fakultet, I. Sarajevo
Dr Milica Pejanović-Đurišić, Elektrotehnički fakultet, Podgorica
Dr Dejan Popović, Elektrotehnički fakultet, Beograd
Dr Božo Krstajić, Elektrotehnički fakultet, Podgorica
Dr Milovan Radulović, Elektrotehnički fakultet, Podgorica
Dr Budimir Lutovac, Elektrotehnički fakultet, Podgorica
Dr Igor Radusinović, Elektrotehnički fakultet, Podgorica
Dr Igor Đurović, Elektrotehnički fakultet, Podgorica
Dr Miloš Daković, Elektrotehnički fakultet, Podgorica
Dr Milutin Radonjić, Elektrotehnički fakultet, Podgorica
Dr Ramo Šendelj, Fakultet za Informacione Tehnologije, Podgorica
Dr Stevan Šćepanović, Prirodno-matematički fakultet, Podgorica
Dr Sašo Gelev, Elektrotehnički fakultet, Radoviš

Organizacioni odbor

Dr Božo Krstajić, Elektrotehnički fakultet, Podgorica
Dr Milovan Radulović, Elektrotehnički fakultet, Podgorica
Dr Ana Jovanović, Elektrotehnički fakultet, Podgorica
Dr Saša Mujović, Elektrotehnički fakultet, Podgorica
MSc Žarko Zečević, Elektrotehnički fakultet, Podgorica
Vladan Tabaš, dipl.ing., Čikom

Sekretarijat

Aleksandra Radulović, CIS Univerzitet Crne Gore

P R E D G O V O R

Poštovani učesnici i čitaoci,

Pred vama je zbornik radova koji su prezentovani na XIX Naučno-stručnom skupu "INFORMACIONE TEHNOLOGIJE – sadašnjost i budućnost" (IT'14) koji je uspješno održan od 24. do 28. februara 2014. godine na Žabljaku. U cilju sveobuhvatnog i multidisciplinarnog sagledavanja aktuelnosti u oblasti informaciono-komunikacionih tehnologija u Crnoj Gori i svijetu, Programski odbor je izvršio selekciju kvalitetnih radova čiji su rezultati bili prezentovani učesnicima Skupa, dostupni svim korisnicima Interneta preko web sajta www.it.ac.me I konačno vama putem ovog zbornika. Konstantno povećanje broja prijavljenih i prezentovanih radova, sve širi krug eminetnih naučnika uključenih u proces recenzije i izbor najboljih radova koji će se štampati u časopisu Elektrotehničkog fakulteta Univerziteta Crne Gore u Podgorici ("ETF Journal of Electrical Engineering") su jasna potvrda uvećanju kredibiliteta i kvaliteta konferencije.

U okviru Skupa je održana radionica „Razvoj mobilnih aplikacija", okrugli sto "IXP4ME – potreba i dileme" i prezentovani projekti: "Fostering innovation based research for e-Montenegro – Fore-Mont", "ECDL za digitalnu Crnu Goru", "Wireless Montenegro" i „Open Govrenment Data (OD) inicijative u Crnoj Gori".

U ime organizatora skupa se zahvaljujem svima koji su na bilo koji način učestovali u radu ovogodišnje konferencije i najavljujem posebno osmišljen i bogat program za naredni-jubilej XX IT.

Sve detalje o ovom, prošlim i narednom skupu možete naći na poznatoj adresi www.it.ac.me.

Prof. dr Božo Krstajić

SADRŽAJ

UPOTREBA GPU U SISTEMIMA ZA DETEKCIJU E-MAIL SPAM-A I IDS
USING GPU FOR QUERY OF EMAIL SPAM DETECTION SYSTEMS AND IDS

Aleksandar Sokolovski, *Neotel – Macedonia, R&D Department, Skopje, Macedonia*
Saso Gelev, *Faculty of Electrical engineering, University of Goce Delcev, Stip, Macedonia*

Sadržaj: *Cilj ovog istraživačkog rada je jedan od najvažnijih aspekata današnjice, a to je bezbednost i menađment jednog na najvažnijih servisa email i svih srodnih servisi. Ovim radom pokušavamo da iskoristime sve pogodnosti standardnih metoda baziranih na potpisu u kombinaciji so algoritmima za mrežnu detekciju napada. Primarni cilj je potvrditi dali će predloženo rešenje biti efektivna strategija za email spam. Glavni smer je odredite najbolju moguću integraciju standardnih metoda za detekciju spam-a sa logikom baziranu na potpisu u kombinaciji sa algoritmima za mrežnu detekciju napada. Glavni zadatak je efektivnije spravljanje sa email spam-om u poređenju sa prethodno ponuđenim rešenjima. Ovo ćemo postići testiranjem efikasnosti predloženog rešenja na NS'3 mrežnom simulatoru i drugih programskih alata.*

Abstract: *The scope of this research paper is one very important aspects nowadays, the security and management of one of the most important services the email and all of the alike online services today. This paper attempts to investigate the possible benefits of using standard signature-driven spam detection logic in combination with algorithm for network intrusion detection system (NIDS). The primary objective is to verify that proposed solution (standard signature-driven spam detection logic and NIDS) will be an effective strategy for dealing with email spam detection. The main aim is to determine best possible integration of standard signature-driven spam detection logic in combination with algorithm for NIDS, for creating more effective solution for dealing with email spam and priority mail compared to the previous solutions available. This will be achieved by testing the effectiveness of the solution compared to other solutions until today, by using network simulators NS-3 and other programming tools.*

1. INTRODUCTION

The email was one of the most important communication tools and it also is one of the most important communication tools nowadays. The email is also one of the most used services on the World Wide Web. According [1] to here are the general data about email:

- In 2012 144 billions emails were send per day worldwide
- In 2012 there were 2.2 billion email users worldwide.
- In 2012 4.3 billion email clients for sending email worldwide
- In 2012 and 69% of worldwide email was spam.
- In 2011 also 71% of email was email spam.

The importance of fast email delivery has change over the year and from being a reliable and fast delivery system, it has become an immediate delivery system. Many enterprises, companies and organization use various techniques and methods for dealing with spam:

- DNS whitelist and blacklists
- Load Sharing: Load is shared among several servers using Layer 4 switches or DNS round-robin.
- DNS Blacklisting: Identify whether the sender is going to send spam or not according to the sender's IP address with a help of DNS.
- Bayesian spam filtering
- Hybrid Filtering

Firewall is hardware or software based network security systems that monitors and controls inbound and outbound network traffic by analyzing the packets.

The first generation of firewalls only filter packets.

The second generation of firewalls are known as stafefull, they analyze packets like the first generation (Layer 3 – IP address) but the difference is they operate up to Layer 4 (Ports) of the ISO OSI network model.

The Third generation of firewall are known as application firewall and they work on Layer 7 (ISO OSI).

The Next Generation of firewalls are Intrusion Detection Systems (IDS) monitors network traffic for suspicious activity and alerts in case of intrusion. Intrusion Detection Systems that only detects intrusions are known as passive IDS, the IDS's that detects intrusions and act to prevent the intrusion are known as active IDS or Intrusion Prevention Systems. The IDS's can also be categories depending on the placement of the IDS (host or network) or depending on the way it detects intrusions (signature or anomaly based). Host Intrusion Detection Systems work on the side of the host and monitor inbound and outbound packets. Network Intrusion Detection Systems are monitoring the entire network (LANs or VLANs usually) all traffic inbound and outbound from any host within the network. Signature based IDS monitors' packets within entire network and compares the possible intrusions against a precompiled database of rules or signatures. Anomaly based IDS monitors' packets within entire network and compares the possible intrusions against an established baseline. The baseline detected “unusual” behavior within the packets and detected intrusions.

Many of the previously mentioned solutions for spam detection and IDS are great and in the beginning when the list is small and manageable work fast and efficient. When the lists becomes larger the processing becomes slower.

In Conclusion the main drawback in today’s email spam solutions, priority mail delivery and IDS solutions is the query processing. In our paper we present solution for the problem using GPU (Graphical processing Units) more specifically CUDA programming. In part 2 and 3 of this paper we present the GPU and CUDA. In part 4 of this paper we explain the DARPA dataset. In part 5 of this paper we present our solution.

2. GPU COMPUTING

GPU (Graphical Processing Unit) computing or the usage of graphical processors for all-purpose computing, began less than 10 years ago. The research in the beginning only allow programming via graphics language, which made in un-flexible. The NVIDIA's CUDA platform according to [2], provides massively multithreaded general-purpose architecture with up to 128 processor cores and thousands of threads. The platform is programmable in the programming language C and capable of hundreds of billions of floating-point operations each second and supports running on all current and older versions of NVIDIA GPUs, this includes the HPC oriented Tesla product line and Kepler core GPUs. The NVIIDIA GPU's are sold in millions worldwide which makes them a very good platform for accelerating high-performance computing (HPC) Applications. GPU computing speeds up the real-worlds science and engineering codes up to 10 or 100 times (depending on the GPU's hardware performance). It's domains of usage range from Computer Technology and MRI, computational chemistry, astrophysics to gene sequencing. Here are few examples of CUDA usage: Friedrich-Alexander- Universidad Erlangen-Nurnberg works on biomedical imaging to accelerate CT reconstruction, working on CUDA using the FDK algorithm. University of Illinois works on accelerating advanced MRI reconstruction techniques using CUDA [3]. The last mentioned work focuses on reconstruction for non-Cartesian scan paths, which reduces image-space error from 45% for conventional reconstruction to 12% but has been considered computationally infeasible in practice. This approach uses NVIDIA Quadro FX5600 GPU and works 13-times faster than on Intel Core 2 Extreme quad-core CPU. This means that the reconstruction times is under 2 minutes for 128 volume. This important types of speedup can change science. According to [4] in medicine this types of speedups can make a significant change in clinical practice, for example an analysis in a lab can be done in hours or minutes instead of days. In biomedical imaging the processes also speedups. Another science field where GPU CUDA speeds up the work is astrophysics (mapping of stars, image processing from telescopes like Hubble and other similar tasks). In short, the potential to greatly accelerate computational techniques opens exciting avenues for biomedical imaging research, astrophysics and other fields.

3. CUDA PROGRAMMING MODEL

The key strength of the GPU is its extremely parallel nature. The CUDA programming model enables developers to use the parallelism by using programming code written in C/C++ programming language. The code will run in thousands or millions of parallel invocations (depending on the GPU) or threads. We will present a simple example of Matrix Addition. For adding two N x N matrices on the CPU in C/C++, we would use double nested loop like in the next code:

```
void addMatrix(float *a, float *b,
float *c, int N)
{
int i, j, index;
for (i = 0; i < N; i++) {
for (j = 0; j < N; j++) {
index = i + j * N;
c[index]=a[index] + b[index];
}
void main()
{
..... addMatrix(a, b, c, N);
}
```

Example no.1

The same example written in CUDA will be like this. We write one C function, called a kernel function, to calculate one element in the matrix and invokes as many threads to run that function as the matrix has elements. In each thread the kernel runs with a predefined structure called “threadldx” indicating which of the many threads is running (example no.2):

```
_global_ void addMatrix(float *a,float *b, float *c, int N)
{
int i= threadIdx.x;
int j= threadIdx.y;
int index= i + j * N;
c[index]= a[index] + b[index];
}
void main()
{
dim3 blocksize(N, N),
addMatrix<<<1, blocksize>>>(a, b, c, N);
}
```

Example no. 2

In example no.2 the “_global_” declaration specifier indicates a kernel function that will run on the GPU, the “<<N, N>> syntax indicates that the “addMatrix()” function will be invoked across a group of threads run in parallel, this is called a thread block. Thread block may be one, two or three dimensional, which provides a natural way to invoke computation across the elements in domains like the following: matrix, fields and vectors.

CUDA makes critical improvements to the core components of running kernel function across many parallel threads: hierarchical thread blocks, shared memory and barrier synchronization. Next we explain in details.

Thread block can contain up to 512 thread if it is a NVIDIA Tesla architecture GPU, but kernels are invoked on a grid which consists of many thread block that are scheduled independently. In every thread in every block in a grid executes the kernel and then exits. Additional code will be added (example no.3):

```
_global_ void addMatrix(float *a,float *b, float *c, int N)
{
int i=blockIdx.x*blockDim.x+threadIdx.x;
int j=blockIdx.y*blockDim.y+threadIdx.y;
int index = i + j * N;
if ( i < N && j < N)
c[index]= a[index] + b[index];
}
void main()
{
dim3 dimBlock (16, 16);
dim3 dimGrid (N/dimBlk.x, N/dimBlk.y);
addMatrix<<<dimGrid, dimBlock>>>(a, b, c,N);
}
```

Example no.3

In the example above the size of the thread block which was chosen randomly is 256 (16 times 16). The created grid has enough block to have one thread per matrix element, which is same as the example before. All the threads in a sector block run in parallel. Whether multiple thread block can run one after another (in serial mode) or they can run in parallel, it all is dependable on GPU.

In the matrix threads can run independently, no threads needs to know the elements beings accessed by other threads, but when threads need to cooperate (share results the results of computations or memory fetches) in order to be more efficient. CUDA provides shared memory for this purpose. In the shared memory, the kernels can store data (variables or arrays) that is visible to other thread block. Simple example is calculating the sum value of the elements within an array, this can be archived when thread in a block places the element into an array in shared memory (adding the next and the next and so on). All the threads in a block do the same in parallel. They are calculating sum value of elements by cooperating using the shared memory. The shared memory is small because it is on a chip (16K in 2010 Models of NVIIDA GPU), but it is extremely fast, therefore it is perfect for this type of operations like calculating sum value of array elements.

When threads are operating in parallel on same memory it is important to have a mechanism that ensures that one thread will not attempt to read a result before another thread has ended writing it or changing it. CUDA provides the __syncthreads() intrinsic function for this purpose. "_syncthreads ()"acts as a barrier at which all threads in the block must wait before any are allowed to proceed. [2]

4. TEST DATA SETS

In 1998 and 1999 The Information Systems Technology Group of MIT Lincoln Laboratory with the support of the Defense Advanced Research Projects Agency and the Air Force Research Laboratory (all from the USA), had worked on a new innovative experiment in the field of intrusion detection systems. They had done a cutting edge experiment for the time, creating an Intrusion detection system that monitors the state of an active computer network, looking for some form of attack like denial of service, form of abuse like unauthorized usage, or rear and strange behavior like some forms of so called anomalous behavior.

The experiment was set in a real military base with real computers, but the attack were simulate (it was known what was attack what was a normal connection, this was used later on to evaluate the effectiveness).

The experiment in 1999 (1999 DARPA Intrusion Detection Evaluation Data Set [5]) was small improvement on the experiment done in 1998. In the 1999 the "simulated" attacks lasted 5 weeks, the first and third week was normal traffic, the second week Contained Labeled Attacks. The attacks were divided into five main categories: Denial of Service Attacks, User of Root Attacks, and Remote to Local Attacks, Probes and Data. The full list of attacks is presented on [REF-04]. Then the system was tested with random network packets (some attacks, some normal traffic), there were 201 instances of about 56 types of attacks distributed throughout these two weeks. At the time the main purpose of the experiment was creating the intrusion detection system, but the real "hided" value of this experiment was the 1998/1999 DARPA Intrusion Detection Evaluation Data Set.

The collected data from this are made available for all the researchers that needed a test data set for their intrusion detection system. This data set had made possible the creation of many future intrusion detection systems. Proof of the value of this data set is the number of publications using this data set in their research project. This is the reason why we intended to use the 1998/1999 DARPA Intrusion Detection Evaluation Data Set. [5]

The 1998/1999 DARPA Intrusion Detection Evaluation Data Set, is a very data set containing around 4-5 GB of data, our main purpose was testing our intrusion detection genetic algorithm, so in order to minimize the time for analyzing the data set and maximizing the testing type, we used the optimized versions of the DARPA data set, the KDD CUP 99 Data Set. KDD CUP 99 Data Set is compiled from the 1998 DARPA Intrusion Detection Evaluation Data Set. The database contains normal connections and 24 types of attacks, the types of attacks are presented on [5]. Evaluation on the KDD CUP 99 Data Set and Summary report with type of attacks and number of connections for each type of attacks is presented on the table in [6]. For this paper and our algorithm it is very important to present the KDD CUP 99 Data Set Schema properly and precisely, for this we will use the tables from the tasks for the KDD CUP 1999 [6].

5. ALGORITHM AND IMPLEMENTATION

In [7] we presented signature based IDS which can dynamically allocate the number of rules that are kept in the detection system based on the LAN complexity and number of packets in the network at the moment. The idea is to process and analyze the packet in the distributed agents IDS before the time of arrival at the destination computer and processing the packet by the host computer. This is done using complex mathematical mechanism for calculating the packet travel time. Using Least Recently Used algorithm to remove the not used rules are after the buffer of rules in the DB of the distributed agents is over its pre-defined limit. This makes the hosts in the LAN protected with intrusion detection and prevention for the more recent and frequent type of attacks, but leaves the hosts vulnerable to new attacks (that are not in the current buffer of rules in the DB of the distributed agents). The new attacks will be detected with deeper analyses at the IDS server side (after analyzing the whole "historical" packet in the LAN) and the rule for that attack will be added in the buffer of rules in the DB of the distributed agents. The main problems is that the first new type of attack or intrusion will be only detected but not prevented. In order to increase the efficiency we need to keep more rules in the buffer of rules in the DB of the distributed agents. This can be done by using CUDA programming. To explain the process of using CUDA in IDS analyses, we will use simple matrix (matrix no.1).

Matrix no.1: [1 4 5]; [1]; [9 7].

Firstly in order to be processed by GPU the matrix must have same number of elements in every row. In order to do that in the shared memory (of GPU) every thread adds the number 1 for every element in the row. Then we use the Max_Number() function, that way we calculate the most elements per row, and then the value is placed (MAX_NUMBER) in the shared memory in GPU. Afterward we process the matrix using thread for each row if there is a missing number of elements (compared to the compared to the MAX_NUMBER) we add "empty element" (the number -1 is ignored in processing). Then we get this matrix (matrix no.2). Adding the "empty element" must be done if we want be process the data with CUDA.

matrix no.2: [1 4 5]; [1 -1 -1]; [9 7 -1].

This in our simplistic example representing the buffer of rules in the IDS distributed agent. Then we and packet s from the network traffic (matrix no.3) in the shared memory, pre-process them (adding "empty element" if the number of elements is lower the MAX_NUMBER).

The last step is comparison of the DB IDS rules (matrix no.2) with the packets from the network traffic recorded (array [7, 9,-1]) placed in the shared memory of the GPU in order to be accessed by every thread at once. This speedup the comparison by 10 times at least.

6. RESULTS

The system for testing is the following: One PC: (Intel® Core™ i7-3770K with 8M Cache @ 3.90 GH; Mobile Intel® QM77 Express Chipset; 16 GB DDR3 @ 1600 MHz; HDD 500 GB @ 7200 RPM; GPU: NVIDIA Quadro K2000M - 384 CUDA CORES), RHEF Linux 6 with C/C++, NS-3 and Windows 8.1 with C# (VS 2010).

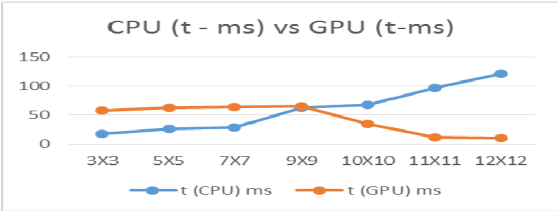

Figure No.1 CPU vs GPU compared (time – ms)

From figure no.1 we can clearly see that GPUs work much faster with larger data sets, compared to CPUs. Therefore they are ideal candidates for using them as main processing power for big data analytics (IDS, email spam).

7. CONCLUSION

NVIDIA GPUs provide massive computation potential in the field of computer science. Our proposal is a new type of using the GPU in the field of computer science more specifically in the field of system administration. Our Future work includes creating an interface for data exchange between CUDA and OPENGL in order to use Intel GPUs as well, also we will try to create dataset of mail for testing email spam (like DARPA DATA for IDS).

REFERENCES

[1] Internet 2012 in numbers - Royal Pingdom U.K., http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/
[2] NVidia CUDA platform, http://www.nvidia.com/
[3] W. Jeong, P. T. Fletcher, R. Tao, and R.T. Whitaker, "Interactive Visualization of Volumetric White Matter Connectivity in DT-MRI Using a Parallel-Hardware Hamilton-Jacobi Solver, 2007.
[4] F. Xu and K. Mueller. "Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware," Medicine and Biology, 2007.
[5] MIT Lincoln Labs DARPA DATA SET 1998/1999 http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/
[6] KDD Cup 1999 Data https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[7] Aleksandar Sokolovski, Saso Gelev ALGORITHM FOR DISTRIBUTED AGENT BASED NETWORK INTRUSION DETECTION SYSTEM (NIDS), CIIT 2011 Bitola, Macedonia.

INDEKS AUTORA

CIP - Каталогизација у публикацији
Национална библиотека Црне Горе, Цетиње

ISBN 978-86-85775-15-4
COBISS.CG-ID 24749840

ISBN 978-86-85775-15-4

9 788685 775154 >