Ubiquitous Multicore (UM) Methodology for Multimedia

[1]Ashley Chonka, [1]Wanlei Zhou, [1]Leanne Ngo, and [2]Yang Xiang
[1]School of Engineering and Information Technology, Deakin University, Melbourne
[2]School of Management and Information Systems, Centre for Intelligent and Networked Systems, Central Queensland University
[1]{ashley, wanlei,mln}@deakin.edu.au, [2]y.xiang@cqu.edu.au

Abstract

For more than a decade now, multimedia developers have usually "ride the waves", so to speak, with the coming of each generation of microprocessors, which allows their applications, designs and programs to usually running more proficiently, efficiently and effectively. This so-called 'free' ride seems to be coming to an end, with results of increases clock speeds, the widening of the gap in processor and memory performance, and the tradeoffs that are needed to meet the former two points, with the new multi-core systems. In this paper, we build upon our previous work within multi-core systems, by proposing a ubiquitous multi-core (UM) design. The goal of such a framework is help researchers to plan and implement their multimedia applications so they can take advantage of speed up computations of multi-core systems and allow real-time multimedia. As our experiments show, our UM system increases performance speeds at an average of 100%, with the average execution cost of 1.4ms, showing that multimedia can use multi-core resources efficiently and effectively.

Keywords: *Multi-core, Multimedia, Ubiquitous Multi-core Framework*

1. Introduction

Today's multimedia designers are demanding more computational speed so that they can meet the demand for better designed and implemented products. This demand by multimedia designers is not the only area in Computer Science requiring greater processing speed. The information industry, in general, is pushing forward towards providing more computer systems that contain multi-processors, networks, security, web services and many more for example. Multi-core systems can be defined as two or more core processors that are connected to a single CPU [1][2][3]. These core processors are incorporate into their design, which in turn share computer resources, like memory, L2 cache and front-side bus [4][5].

With the industry push towards multi-core systems, multi-media developers are confronted with complex questions that require answers, how to make use of these extra core processors? How will these extra cores help me in my development of multimedia applications? Will more core processors, make my applications run faster, slower or not effect it at all? With multi-core systems roughly doubling every 18 months, Intel are now bring out a 8 core system sometime in 2009 [19], which has basically ended serialized programming forever, and forces programmers to adopt parallelized code instead, so that they can make use of these multiple cores, otherwise performance and resource usage are going to suffer. Communication efficiency, for example, between cache resources will probably reduce as more and more cores are implemented, since programmer who serialize their code are not taken them into account while they programming, instead they will just leave it to the O/S to assign instead. If this is the case then cache fragmentation or outdate data within cache (stale cache) will tend to get worse [13].

In this paper, we build upon our previous framework on multi-core architectures [6][20], which we now call ubiquitous multi-core (UM) framework, due to the fact that we have found that our model can be applied to most computer systems using multi-core. Our framework was firstly applied our security system that we designed and implemented on a multi-core system[6] and furthered our results in [20], to our security applications so that we could take advantage of the improved performance that multi-core system offer, but we also found some solutions to issues that was confronting security applications. In [6] we separated out applications, by using affinity method within the C library, so that we assign our applications to their own core processors. We found that we had a +15% increase in performance, since we believe our programs process did not have to wait for the O/S to assign or free up the core processors. Furthering our results in [20], we conducted further analysis and a more through experimentation, which we showed an increase from 15% to 120%, with an average cost the system of 1.5ms. The main reason for such a discrepancy is [6] just had the programs assigned to the core process, with [20] we started to incorporate MPI into our programs, which increased processing speed. Other researchers have also found that this type of framework can be expanded into their areas of research like SPAM filtering [7], and we see that many other researchers in the future will end up using this type of model or something similar.

In papers [6][7] and [20], we list a number of advantages that researchers would see if they started to incorporate the UM framework to their research. In multimedia, researchers could see the following: different media applications running on isolated environments, different media application running in real-time with each other, multimedia activities could be monitored and visualized in real-time. Lastly, multimedia troubleshooting could be greatly enhanced. The rest of this paper is organized as follows. Section Two briefly covers the related work done in multi-core and the UM framework. The application of UM framework being applied to multimedia design is discussed in Section Three. Section Four presents the experiments and performance evaluation that were conducted. Lastly, Section Five covers the conclusion and future work

2. Related Work

In this section, we discuss briefly our past efforts using the UM framework, which was formerly called the bodyguard framework. We further discuss how other researchers have used multi-core systems for multimedia applications, and lastly we briefly cover how other researchers have used the UM framework.

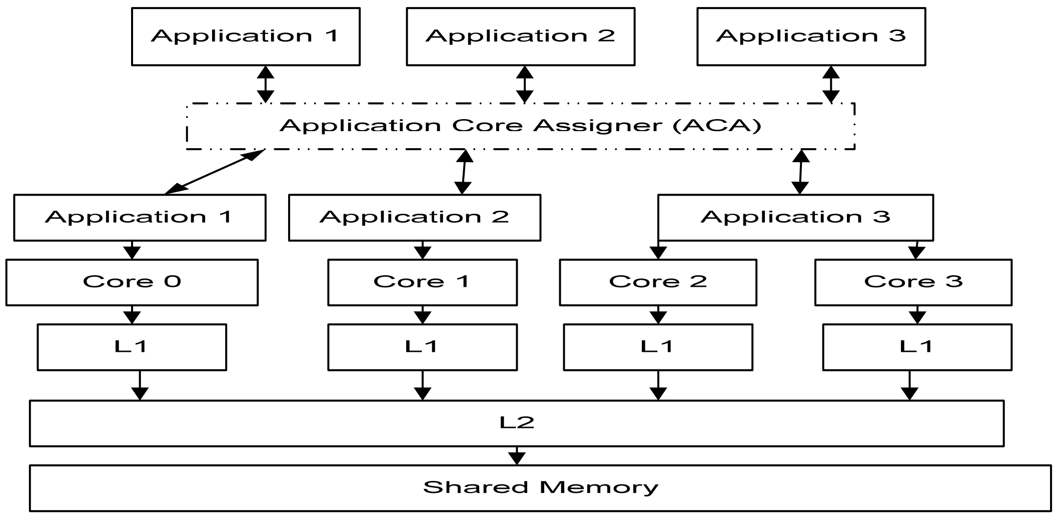


Figure 1. Ubiquitous Multi-core Framework

2.1 Ubiquitous Multi-core (UM) Framework

The Ubiquitous Multi-core Framework, formally known as the bodyguard framework, is built upon the hypothesis of divide-and-conquer [9]. Multi-media developers would have their applications divide and placed on separate core processors, as shown in Figure 1. By placing applications on their own core processors, a multi-media developer should see a speed-up of the applications, but also they will see that their computer resources are being fully utilized, and lastly, race conditions and deadlocking will be reduced. The separation of the core-processors within the UM Framework, is called application core assigner (ACA), which the multi-media user can assign themselves or allow the system to do it on their behalf. Once an application is assigned to a core, depending on the application program, a number of jobs or threads can then be executed on this core processor, see figure 2.

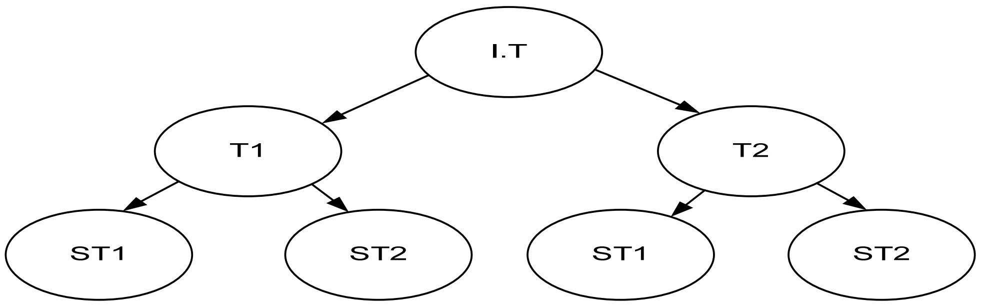


Figure 2. Example of Thread Processes on Core 0

2.2 Multi-core systems used for multimedia applications.

Multi-core systems have two or more processing cores integrated into a single chip [1][2][3]. In such a design, processing cores have their own private cache (L1) and a shared common cache (L2). The shared cache and main memory share the bandwidth between all the processing cores. Multimedia co-processor interface was developed by

[8], in which they used a multi-core system to offload task management jobs from MPU or DSP. From their evaluations conducted on JPEG files, Ou et al. achieved an overall performance increase of 57%, while they kept their overhead to 1.56% of the DSP core. The UM framework is very different from Ou et al., in which UM is more abstract, by applying applications (not separate sections of a file) to separate core processors.

2.3 UM Framework for a Security Application

In our first paper [6] on multi-core systems, we presented a multi-core defence framework called bodyguard. Using this framework, we developed a bodyguard called Farmer (named after the Kevin Costner character in the movie, bodyguard). The basic hypothesis of the bodyguard framework, was to separate all security processes from other processes (email, browser, etc), and assign them to a set of cores. The remaining cores within the system were assigned to the applications that require security. The bodyguard framework is made up of a Forward Bodyguard (FB) and Side Bodyguard (SB). For example, in our Farmer bodyguard, the SB is responsible for providing a fast decision on whether to filter out any attack traffic. From this paper, we then were able to see that this type framework could be applied to other areas, like multimedia and Spam filtering, which lead us to the development of the Ubiquitous Multi-core Framework.

Following on from paper [6], we update our research work by moving our bodyguard paradigm, into our new ubiquitous multi-core framework [20]. From this shift, we show a marked improvement from our previous result of 20% to 110% speedup performance with an average cost of 1.5ms. We also conducted a second series of experiments, which we trained up Neural Network, and tested it against actual DDoS attack traffic. From these experiments, we were able to achieve an average of 93.36%, of this attack traffic.

2.4 UM framework for Multi-classifier Spam Filtering

To follow up on [6] and [20], other researchers have started to implement this type of framework to update their research, and to see whether multi-core system can in some way improve their results. Islam et al [7] were able to applied the UM framework to a multi-classifier SPAM filter. What they found was that each classifier process, which was run in parallel with each other, greatly improved the performance of their multi-classifier architecture. It also reduced the false positives, and increased the accuracy. Further advantages were report by Islam et. al., which are as follows:

- Reduced computation burden of the overall mail server.
- Reduced memory storage, email messages are processed independently from other classifiers.
- When one of the classifiers becomes idle it will directly go into training mode, thereby optimizing resource usage.
- Is robust as the adaptive selection can still provide accurate email classification if one of the core fails.

3. Applied UM Framework to multimedia applications

In paper [14], a case study was conducted with the overall aim to help company staff members their IT security culture and awareness based on our IT Security Culture Transition

Model [15]. To accomplish this task a case study was conducted, in which a questionnaire was setup for staff members to answer. In conjunction as to answering the questions, the administrator was monitoring in real-time. Figure 3, displays an example of how to use the UM framework to develop and build our E-learning IT multimedia application

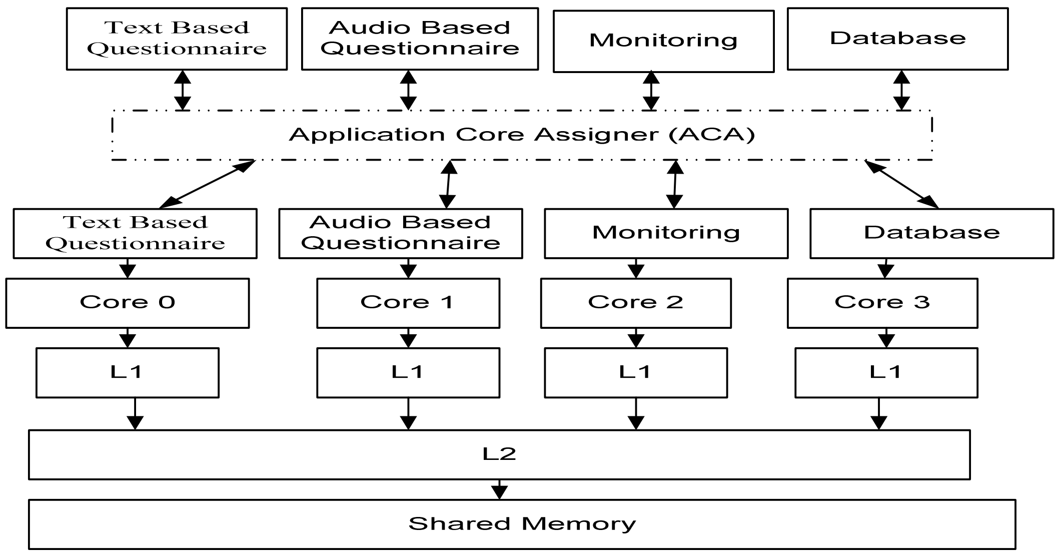


Figure 3. E-Learning IT Security Multimedia Application

3.1 E-Learning IT Security Multimedia Multicore Algorithm (ELITE MA)

From figure 3, we develop Figure 4 ELITE MA (E-Learning IT Security Multimedia Algorithm). In order to partition the application takes into consideration the following the developmental algorithm equations of [17][18]. Note: Our algorithm calculations are a little different from Fosters and Wilkinson et. al., since they deal with multiple processors on different machines, our algorithm equations are based on multi-core systems. The total communication time for the 4 partition applications in a multi-core environment is as follows:

$$t_{com1} = \frac{n((cp * tcp) - 1)}{(cp * tcp)} t_{md} \tag{1}$$

where t_{md} is the transmission time for a data message sent over broadcasting. Cp is each core processor being used, and tcp is the number of processor to be used. Computational time is represented by counting the number of computational steps, usually if all processors are being used then just one process computation is necessary: where n is the number of computation and cp is the number of cores.

$$t_{comp} = f(n, cp) * tcp \tag{2}$$

```
1. Ask User if they want to
assign partition manually or
automatically
2. Partition Application
3. Assign Partitions to core
processors.
4. Each Application
performs communications
and computations.
5. If application is finished
go back to 1.
6. If all partitions are
finished, notify user of the
performance (if required),
otherwise End
```

Figure 4. E-Learning IT Security multimedia Algorithm (ELITE MA)

Communication Time is depended upon the number of messages, size of the message, communication infrastructure (communication and network):

$$t_{com} = t_{beg\_startup} + w\, t_{md} \tag{3}$$

$t_{beg\_startup}$ is the message latency, which is the time it takes for a message to be sent with no data. The data messages sent via each partition is found in the formula:

$$t_{com2} = (\log(cp * tcp))t_{md} \tag{4}$$

For the total communication time is as follows:

$$t_{ttc} = t_{com1} + t_{com2} = \frac{n((cp * tcp) - 1)}{(cp * tcp)} t_{md} \tag{5}$$

$$T_{tc} = T_{ttc} + (\log(cp * tcp))t_{md} \tag{6}$$

The computation formula for the 4 partition applications at the end of the partition phase (7) is as follows:

$$t_{comp} = \frac{n}{(cp * tcp)} \tag{7}$$

This gives us the Overall Execution Time for the 4 partition applications in the following formula:

$$t_p = \left[\frac{n((cp*tcp)-1)}{(cp*tcp)} + \log(cp*tcp) \right] t_{md} \quad (8)$$

$$t_{p1} = t_p + \frac{n}{(cp*tcp)} + \log(cp*tcp) \quad (9)$$

The very best speedup we could expect, when the 4 partitioned applications have completed their computations, is as follows:

$$\frac{t_s}{t_p} = \frac{n-1}{((n/cp*tcp)((cp*tcp)-1) + \log(cp*tcp)t_{md}} \quad (10)$$
$$+n/(cp*tcp + \log(cp*tcp)$$

The actually speedup will be less than this due to partition phase; computation/communication (c/c) ratio is as follows:

$$\frac{t_{tcom}}{t_{comp}} = \frac{n/(cp*tcp) + \log(cp*tcp)}{((n/(cp*tcp))((cp*tcp)-1) + \log(cp*tcp))t_{md}} \quad (11)$$

For load balancing we use the Mandelbrot computation [16], in which if the maximum performance (mp) is reached for the processor, it will then search for another core processor to continue the work.

$$T_s \leq mp * m \quad (12)$$

To partition the application correctly we use three phases' communication, computation and communication.

Phase 1:

$$t_{comm1} = (p-1)(t_{stup} + t_d) \quad (13)$$

Phase 2:

$$t_{comp} \leq \frac{mp * n}{p-1} \quad (14)$$

Phase 3:

$$t_{comm2} = u(t_{stup} + vt_d) \quad (15)$$

In order to maintain the highest speedup and computation/communication ratio we use the Overall Execution Time (16), Speedup factor (17), C/C ration (18):

$$t_p \leq \frac{mp*n}{p-1} + (p-1)(t_{stup} + t_d) + k \quad (16)$$

$$\frac{t_s}{t_p} = \frac{mp*n}{\frac{mp*n}{p-1} + (p-1)(t_{stup} + t_d) + k} \quad (17)$$

$$\frac{mp*n}{(p-1)((p-1)(t_{stup} + t_d) + k)} \quad (18)$$

4. Performance Evaluation

We evaluate ELITE MA by simulating Figure 3, in which we assigned 4 applications on a multi-core system

4.1 ELITE MA Performance Analysis

To assess the performance of our multi-core system, we compared the two kernel benchmarks. The hardware on the multi-core system had Intel Core 2 Quad Q6600 2.4GHz Quad Core Processor, 2 GB of RAM and 2 300GB SATA hard-drives. The kernel under measurement was 2.6.22.14.72 fc6. To gather computational data, we included timers with our application, in order to record execution times. Communication time is depended upon the number of messages, the size of the message and the interconnection speed. We have decided to set the standard to 10ms, for each message sent by Text Based and Audio Based Questionnaire, followed by 20ms being applied to Database and Monitoring.

4.2. Simulation Setup

Benchmark factors

Once we have the execution times t_s, computational time t_{com}, and communication time t_{com}, we can establish what the speedup factor (Formula 19) and computation/communication ratio (formula 20) from a single core to multi-core system. The speedup is as follows:

$$\frac{t_s}{t_{cp}} = \frac{t_s}{t_{comp} + t_{com}} \quad (19)$$

Where t_s will stand for execution time on a single core processor (t_{cp}), this includes computation time and communication time.

$$\frac{t_{comp}}{t_{com}} \quad (20)$$

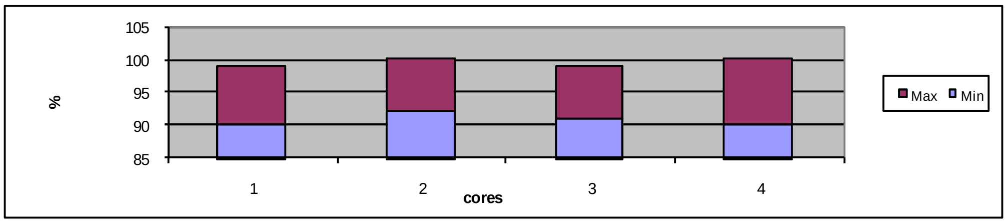


Figure 5. Min (90%)-Max(100%) CPU usage that was archived during our simulation

Table 1. Results of speedup and the costs, which show an average increase of 100% at the average cost of 1.4ms

	Core 1	Core 2	Core 3	Core 4
Exe Time	1.5ms	1.4ms	1.3ms	1.4ms
Comp Time	.5 ms	.9ms	.3ms	.9ms
Comm Time	1ms	.5ms	1ms	.5ms
Speed Ratio	100%	100%	100%	100%
C/C	0.5	1.8	0.3	1.8
Time Complex	7.5	7.5	7.5	7.5
Cost	1.5	1.4	1.3	1.4
Cost-Optimal	3.7	3.7	3.7	3.7

Apart from speedup and the Computation/Communications ratios, we also evaluate the ELITE MA algorithm, through the use of Time Complexity or “big-oh”, also referred to as “order of magnitude” [12]

$$\begin{gathered} f(x) = O(g(x)) \\ \left[0 \le f(x) \le cg(x)\right] \text{ for all } x \ge 0 \end{gathered} \tag{21}$$

Where f(x) and g(x) are functions of x, a positive constant, c, has to exist for all $x \ge x_0$ otherwise it is zero. To evaluate Time complexity, we use the total sum of computation and communication (11).

$$t_{cp} + t_{com} = (n / cp + 1) + (2t_{stup} + (n / cp + 1)t_{md} \tag{22}$$

Where n is the number of threads on each core processor. The last benchmark we will use is the cost and cost-optimal, which are as follows:
Cost = (execution time) * (total number of processor used)
Cost Optimal = time complexity * number of processor = (n log n)

Simulated Program

To measure and evaluate the performance, we wrote 4 simple programs to simulate the media applications, and assigned them to 4 cores within our multi-core system by using affinity methods. The multimedia functions are simulated, by the 4 programs just to demonstrate the model, though 4 actual multimedia programs are planned in the future.

4.2. Evaluation

Based on our evaluations, displayed in table 1 and figure 5, we see that a speed average of 100% was archived at the average cost of 1.4ms. This is achieved by separating out each application and allowing them to run on their own separate cores.

The 100% ratio, we think is a bit optimistic; thereby computation time and communication time do need to be tested in real-time, to give more accurate account. The time complexity results also show that the efficiency of our algorithm is at 7.5, which means that for 13 computational steps (estimated) we achieved 7.5 data items.

Our opinion of this result is that the more computations that are done the more data items were complete. For example, 15 computational steps will give us 8.5 data items.

One of the more interesting results displayed in Table 1, was the Computation/Communication Ratio, which showed that it was less then Time Complexity. This means that speedup or efficiency will not go beyond the figures we already have. Lastly, we see that the cost of running our program was below the cost-optimal, and at the same achieving an average of 95% CPU (see figure 5). This means that our model/program was quite cost efficient, which ran resource usage, such as the CPU, to it fullest optimization. We also note that due Time Complexity being higher then Computation/Communication ratio, it would not be worthwhile trying to send our costs up to reach the optimal, since we would gain no performance benefit

5. Conclusion and Future Work

In this paper, we introduced a ubiquitous multi-core framework which was generated from our previous work on multi-core. From this framework, we designed and built a multimedia multi-core system called ELITE MA.

The goal of such a system is to use the new multi-core machines that are coming out, in order to fully utilize the power of the multi-core system. We show with our experimental results that a speedup average of 100% with an average cost of 1.4ms, and a CPU efficiency of +90% for multimedia programs. In the future, we are plan to move our new multi-core system on to the enterprise grid system (a number of machines with 4 cores each), at Deakin University, and to improve upon our results.

Acknowledgments

This research was supported by the ARC Linkage grant (Project number LP0562156).

References

[1] Multi-Core from Intel – Products and Platforms. http: //www.intel.com/multi- core/products.htm, 2006.
[2] AMD, (2008), http://multicore.amd.com/en/Products/, 2006.
[3] Gorder, P.M, (2007), '*Multicore processors for science and engineering*', IEEE CS and the AIP, 1521-9615/07/,March/April 2007
[4] Calandrino, J.M, Anderson, J.H., and Baumberger, D. P, 2007, '*A Hybrid Real-Time Scheduling Approach for Large-Scale Multicore Platforms*'**,** 19th Euromicro Conference on Real-Time Systems (ECRTS'07), IEEE, 2007
[5] Bader, D.A, Kanade, V and Madduri, K, (2007), 'SWARM: A Parallel Programming Framework for Multicore Processors', Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International 26-30 March 2007 Page(s):1 – 8
[6] Chonka, A., Zhou, W., Knapp, K and Xiang, Y., (2008), "*Protecting Information Systems from DDoS Attack Using Multicore Methodology*", IEEE 8th International Conference on Computer and Information Technology, IEEE, 2008.
[7] Islam, R. M.D, Singh, J, Zhou, W., and Chonka, A., (2008) , "*Multi-Classifier Classification of Spam Email on a Multicore Architecture*", Proceedings of IFIP International Conference on Network and Parallel Computing, 2008
[8] Ou, S.H., Lin, T.J., Deng, X.S., Zhuo, Z.H., Liu, C.W., (2008), "*Multithreaded coprocessor interface for multi-core multimedia SoC*', Proceedings of the 2008 conference on Asia and South Pacific design automation, Seoul, Korea SESSION: University LSI design contest, Pages 115-116, ISBN:978-1-4244-1922-7, 2008
[9] JaJa, J. (1992), '*An Introduction to Parallel Algorithms*", Addison Wesley, Reading, MA
[10] Pierucci, L, and Del Re, E, (200), "An Interactive Multimedia Satellite Telemedicine Service," *IEEE MultiMedia*, vol. 07, no. 2, pp. 76-83, Apr-Jun, 2000

[11] Sicurello, F**,** (2001), “*Some aspects on telemedicine and health network*”, Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on Volume , Issue , 2001 Page(s):651 – 654

[12] Knuth, D.E, (1976), “Big Omicron, Big Omega and Big Theta”, SIGACT News (April-June), pp18-24

[13] Dongarra, J., Gannon, D., Fox, G., Kennedy, K. "The Impact of Multicore on Computational Science Software ," *CTWatch Quarterly*, Volume 3, Number 1, February 2007. http://www.ctwatch.org/quarterly/articles/2007/02/the-impact-of-multicore-on-computational-science-software/

[14] Ngo, L, Lanham E., Zhou, W., Warren, M., (2007), ‘Using E-learning to improve to Improve IT Security in a corporate environment: A Case Study’

[15] Ngo, L, (2008), ‘IT Security Culture Transition Process’ IGI Global encyclopedia, Encyclopedia of Information Ethics and Security, edited by Dr. Quigley

[16] Wilson, G.V, (1995), ‘*Practical Parallel Programming*’, MIT Press, Cambridge, MA

[17] Foster, I, (1994), “*Designing and Building Parallel Programs: concepts and tools for parallel software engineering*”, Addison-Wesley Publishing Company, (1994)

[18] Wilkson, B & Allen, M, (2005), “*Parallel Programming: Techniques and Applications using network workstations and parallel computers*”, Pearson Education, Pearson Prentice Hall, (2005)

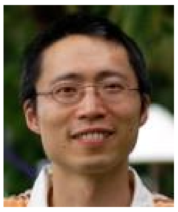
[19] Chiappetta, M, (2007), “8-Core Intel Xeon ‘V8’ Sneak Peek – Dual Quads”, Hot Hardware, http://hothardware.com/News/8Core_Intel_Xeon_V8_Test_Rig__Sneak_Peek/ , 25th April, 2007

[20] Chonka, A, Chong, S.K, Zhou,W, and Xiang, Y, (2008) “Multi-core Security Defense System (MSDS)”, *IEEE The Australasia Telecommunications Networks and Applications Conference*, IEEE, 2008

Authors

Ashley Chonka is a PhD candidate at Deakin University, since December of 2005, and will be completing his PhD June 2009. He has successfully published several peer-reviewed papers and is currently a research assistant at Deakin's first Honeypot Project, which is under the supervision of Professor Wanlei Zhou.

Wanlei Zhou received his PhD degree from The Australian National University, Canberra, Australia, in October 1991. He also received the DSc degree from Deakin University,Victoria, Australia in 2002. He is currently the Chair Professor of Information Technology and the Associate Dean (International), Faculty of Science and Technology, Deakin University, Melbourne, Australia. His research interests include distributed and parallel systems, network security, mobile computing, bioinformatics and e-learning. Professor Zhou has published more than 170 papers in refereed international journals and refereed international conferences proceedings. Since 1997 he has been involved in organizing more than 50 international conferences and has been invited as keynote speaker for a number of conferences.



Yang Xiang is currently with School of Management and Information Systems, Central Queensland University. His research interests include network and system security, and wireless systems. In particular, he is currently working in a research group developing active defense systems against large-scale network attacks and new Internet security countermeasures. He has served as PC Chair for the 11th IEEE International Conference on High Performance Computing and Communications (HPCC-09), 2008 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS 08), the 2008 IFIP International Workshop on Network and System Security, the 2008 IEEE International Workshop on Cyberspace Safety and Security, and the 2007 IFIP International Workshop on Network and System Security. He has been PC member for many international conferences such as IEEE ICC, IEEE GLOBECOM and IEEE ICPADS. He has served as or is serving as guest editor for ACM Transactions on Autonomous and Adaptive Systems, Journal of Network and Computer Applications, Concurrency and Computation: Practice and Experience, Security and Communication Networks, and the International Journal of Computer Systems Science and Engineering. He is on the editorial board of Journal of Network and Computer Applications.

Leanne Ngo is an I.T Security Lecturer and PhD candidate at Deakin University. She is the author of several peer-reviewed papers and her current research deals with I.T. security culture within organizations. Her other research interests are with network security, security in multi-core system and education of general computer and information security.