Future IT TREND (s)

Whats stopping us from leaping forward? Hurdles to Moors Law!

Name: Senthoorkumaran Punniamoorthy

ABSTRACT

Analysis of the hurdles in front of us before we can fully reap the benefits from the advances in micro-processor technology. Its an analysis of where the technology can take us in the future "provided" we jump the hurdles.

Table of Contents

Executive Summary

After a briefly introducing the definitions of the terms used, the research looks at what hurdles, the paradigm shift to multi core processors, brings to the table. Then in order to over come the hurdles, this report looks at, what possible near term and future solutions available to the industry as a whole. Four such possible solutions are explored and in conclusion the report analyses which direction future might take. At the end, the report concludes with a section addressing the managerial issues faced by today's managers, due to this paradigm shift and equips them with enough information, to make the right decision about future software development in their own organization.

Definitions

Moore's Law

"The complexity for minimum costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase… That means by 1975, the number of components per integrated circuit for minimal cost will be 65,000."

"I believe such a large circuit can be built on a single wafer"

~Gordon Moore, Electronic Magazine, April 19,1965

In simpler term it is to say, "*Computer chips get twice as fast every year or two*" (Blankenhorn D, 2002, p1)

Frequency (MHz) Vs Processor Throughput

Processor throughput, also how much real work a processor can get done in a given duration has become synonymous with clock speed. People compare the clock speed of two different processors and come to wrong understanding that the one with higher clock speed has more throughput. This perception was created by Intel marketing machine.

Frequency of a microprocessor is indeed a defining attribute of a microprocessor, however it is not the only defining attribute. Instruction set, execution capabilities, number of cores, threads per core and memory interfaces (bus speed) also define capabilities of a microprocessor. A 500-MHz non-superscalar microprocessor could be easily over taken by 200-MHz four-issue, superscalar design. (Balch M, 2003, p171)

Processor Core and Threads Per Core

In the early days of processor enhancement the designers then concentrated on three forms of improvements to produce the next generation microprocessor.

1. Frequency
2. Instruction Set (e.g. Pentium MMX)
3. Level 1/Level 2 Cache

However the microprocessor industry specially the desktop industry hit a barrier when it tried to cross the 3GHz clock speed. After a long struggle when Intel finally broke the 3GHz in 2002, one aspect of processor design was clear, that making the processor faster by only increasing the clock speed in the future will not be the right approach. This led the microprocessor companies to start borrowing techniques from other disciplines like super computer and server-based processor.

Now the industry is moving forward with multiple cores (two processing dies in single chip) and multiple threads per core. For example Sun Micro Systems Rock Processor that has 16 Cores and each core capable of running 2 threads parallel, enabling greater parallelism. (Sun Microsystems Press Release, 2007)

Hurdles for Moore's Law

Single Core

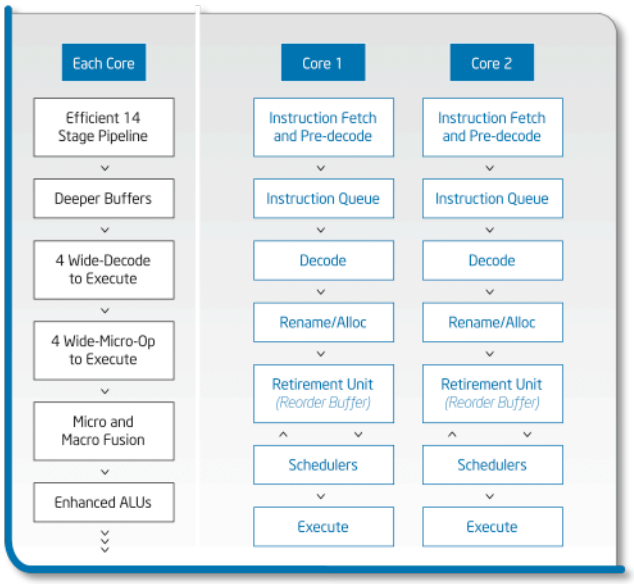
Single Core CPUs had a tremendous success in the past decade. A 58% year to year growth according to a research paper published by Chalmers University, Switzerland (Ekman M et el, 2004). However after breaking the 3GHz barrier the landscape of desktop computing changed where further increase in clock speed was very difficult and the performance gains were marginal. Three factors were considered the hurdles and they are

- ***Total dissipated power***: When the number of transistors packed in a die was doubling (from 180nm, 90nm to now 45nm) the power requirement of the processor is very high and it also generates heat.
- ***Wire Delay***: Due to resistance (Om) of the wire and other technical details the speed at which signal travel globally is reduced when the number of interconnects increases.
- ***"The memory wall"***: Increase in memory speeds doesn't match the processor speeds. The DRAM speeds have been up by 10% each year where the processor speeds have gone up by almost 50% each year.

If the numbers of transistors are doubled in a single core CPU, the throughput increase we gain in return is about 40%. However keep on doubling the transistors, due to the reasons outlined above, is not physically possible and thus the era of Single Core Processor has come to an end. (Johan D. G, 2005)

Multi Core

Welcome to a new era. In order to overcome the problems of Single core architectures, the industry shifted towards multi-core architecture, borrowing techniques from Server processor building technology, mainly being adapted by Sun, IBM and other server processor vendors.



From White Paper by Wechsler O, Intel.com

As depicted in the diagram, a multi-core architecture is equivalent to having two or more processors in one die. This is a scalable solution that will dominate the next generation of processors.

Software Bottleneck - Chocking Multi-Core

It's great, that processor industry has found a way to keep the Moore's law alive for foreseeable future. However if we the consumers are to realize the throughput gains of multi-core processor, the software world has to translate these gains into tangible improved user experience.

As much as the hardware vendors are excited about multi-core/multi-threaded architecture of processors, the software vendors are not so excited about the development since it significantly increases the resources (human/time/money) needed to scale the applications to match the hardware development.

Operating System Perspective

Today's desktop operating systems (Windows and Mac) can run different application in different cores enabling multi tasking quite possible and smooth, however single threaded application performance increase is yet a problem since the OSs are not able to utilize multiple cores to run a single threaded application.

Application Perspective

95% of desktop applications on popular Windows and Mac OS that have been written for decades are single threaded.

Core	Processor Speed Increase	Single-Threaded Application Increase
Single Core	X Speed To 2X Speed	Y Speed To 2Y Speed
Multi Core	One Core to Two Core	Remains in Y Speed

Table showing Single Threaded Application performance in various configurations.

Core	Processor Speed Increase	Multi-Threaded Application Increase
Single Core	X Speed To 2X Speed	Y Speed To 2Y Speed
Multi Core	One Core to Two Core	Less than 2Y Speed

Table showing Multi Threaded Application performance in various configurations.

Exceptions are in the form of few games which required multi threading in early days so that the Graphical Processing Unit (GPU) intensive process can be done in separate process than actual game logic so that gamers experience a smooth game play.

In the end the truth of the matter is all software venders who wrote desktop applications and enjoyed performance boost to their application when the processor speed doubled every 18 months with absolutely no modification to their software binary, are no longer enjoying that "free lunch", with the introduction of multi-core.

Software Venders Perspective – Free Lunch is over

A Software Architect at Microsoft Herb Shutter puts it like this “The biggest sea change in software development since the OO revolution is knocking at the door, and its name is Concurrency" (Shutter H, 2005). This quote is from an article of which had a title “Free Lunch is over”. What does Shutter means by Free Lunch?

In the single core era if an application performed at speed X when the next processor came out at higher speed without any effort from the software vender the application would run faster on the new processor thus the term Free Lunch.

With Multi Core, the free lunch was essentially over for desktop software venders who were developing Single threaded application, since OS is not capable of running a single threaded application on multiple cores. Venders now have to put additional developers, time and money to make the application multi threaded, and thus the term “Free Lunch is over”.

Princeton university researchers say, “at best, the common single-threaded applications will see no benefit from these future architectures. At worst, the will experience heavy slowdowns as the aggressiveness of each core is scaled back to make room for large number of cores”. (Vachharajani N, et el,2005)

Future Trends – Where will the Future take us

Possible Solution 1: Solution in OS Space:

I think it is highly unlikely! I have been following the OS technology trend from Sun and IBM, with Solaris/AIX/Linux and they are not able to introduce parallelism into a single threaded application utilizing multiple cores.

There are techniques available to increase performance but these techniques too are not suitable beyond 8-16 cores, and if the Moore's law keeps up we will have that in next 4 years time. (Vachharajani N, et el, 2005) In my view this effectively cuts any incentive there for OS vendors to implement these techniques because they will not be able to recover the return on such investments.

Possible Solution 2: Solution in Virtual Machine/Framework Space

I am a Java programmer, compared to many language its relatively new language and its 13 years old. It solves a unique problem of enabling a program written in one language to be able to run in multiple software/hardware platforms without the need of cross compiling the application on those platforms. This is achieved through a virtual machine, which sits in between the OS and Application as a translator. (Joshua. E, 1999)

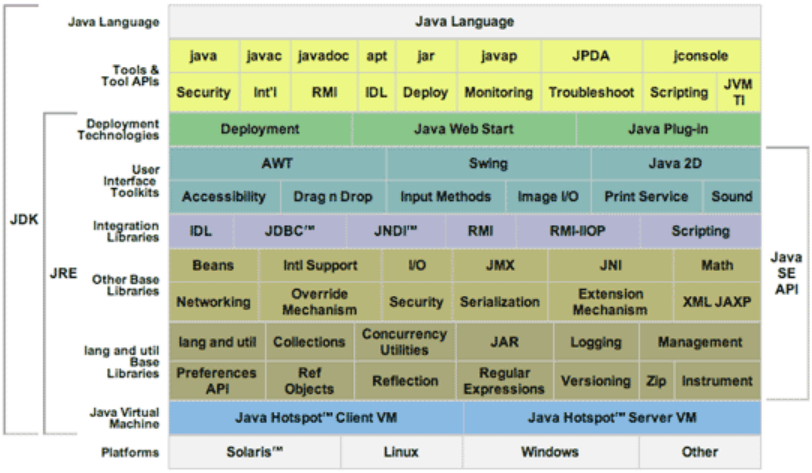


Figure a - Java Platform Stack (java.sun.com)

As the diagram shows, a virtual machine sits in between the platform and the language stack and does the necessary translation.

In my view there are two possible directions this virtual machine based architecture can evolve.

1. Existing VMs like Java and .Net becomes multi threading capable. The libraries and APIs accompanying these platforms will be made multi threaded. A programmer writes to these API and parallelism is built into the API.
2. New bread of VMs are born, these VMs will be capable of running the software across multiple cores when the software is written using the library provided by the VM.

In my view, the existing VM's becoming multi-core enabled has lot of promising future. Moreover this will enable a subclass of software vendors to enjoy free lunch. For example if a firm did write Java based desktop software, when Java

VM becomes multi-core enabled, the firm's software without any binary modification will be able to enjoy parallelism.

Possible Solution 3: Programming Language Space:

This is a reality in the works now. Leading the revolution is a language named Erlang developed at Ericson Computer Science Laboratory. Erlang supports applications with very large concurrent processes. (Armstrong J, et el, 1993).

Erlang is not a general-purpose tool. Currently it doesn't have mature API library. The available library for text processing and regular expression matching is rather slow on single core, but given multiple cores it will raze through, beating other current day languages. (Bray T, Co Inventor of XML, 2007).

Although this language is not prime time ready to be a general-purpose solution for the problem of concurrency, it acts as a reference implementation. It shows there is hope in progressing towards this direction in future.

Possible Solution 4: Network is the Computer, Web 2.0:

The growing trend now is conversion of the desktop applications to web based applications. This trend is called Web 2.0. This is made possible by number of recent development in browser technology and the main contributor is AJAX (Asynchronous JavaScript and XML) (Monica L.M, 2005)

In my view one prime example of desktop applications converting to web application is Google Docs. It has word processing, spreadsheet and presentation applications.

Frameworks like Map Reduce, introduced by Google brings enormous amount of parallelism to the server-based applications without incurring the cost of massive high-end large-scale processors, but with multiple low cost CPUs (Jeffrey D & Sanjay G, 2004). With such tools, it is easier to build and distribute desktop class applications over the web.

Conclusion

Multi-Core technology is a disruptive technology. It is a technological innovation, which uses a disruptive strategy rather than revolutionary strategy. It is a disruptive technology since it is going to change how the software is developed, ground up.

Until the multi-core architecture was introduced software world was getting a free ride. When processor speed increased, software speed increased. With the introduction of multi-core world of software development needs a radical change, to translate the processor gain to improved user experience.

Software development is complicated; adding concurrency to the equation is going to complicate the matter by order of magnitude.

Therefore I feel the direction of future is going to be in one of three forms.

1. A VM implementation with in built concurrency, where programmers can code without worrying about concurrency, and virtual machine will, under the hood, parallelize given application.

2. More and more applications converting to web base, in order to exploit already available concurrency in web servers/application servers/databases, so that vendors don't have to reinvent the wheel.

3. A special class of applications will be rewritten in the future to natively use multi threads to exploit the multiple cores. Examples will be Graphics editing application, Games, Scientific applications, Mathematical Simulations and so on. These will be rewritten in exciting languages like C using threads, or new bread of language will evolve with built in concurrency.

Interestingly the revolution happening in the mobile computing arena might influence which of the three approaches get more traction in order to jump the Moore's law hurdle. Already mobile Safari, the mobile browser from apple is said to be full blown browser on a mobile platform. A full-blown browser on a mobile platform coupled with AJAX based web technologies can deliver desktop class applications to mobile devices. This might have an heavy influence in future software development direction in my opinion.

Where we will end up in next 10 years? Web based application or Virtual Machine powered desktop applications? Only time will tell.

Managerial Issues

In your company, if there is a proposal to develop a new desktop application to automate some task/any task, as a manager with a long term view, what language/ architecture/ model would you select for the new software development, so that the software developed will stand the test of time. Current day managers probably don't realize how big of a decision this is, but this is going to be a million dollar (some software's do cost millions to develop and maintain) questions most manager are going to face.

At present it would make a lot of sense to go for a web based software, with AJAX and other technology involving. If Google can develop mapping application on the web, it is proof enough that such desktop class application can be written over web technologies. This path makes more sense until such time there is some framework enabling concurrency without programmers worrying at the code level rather than it enables at VM level.

If the application in question is real time, mission critical, and cannot work with the latency inherent for web application, the next best option for managers is to develop a multi threaded application, in one of the current day language. However managers must realize that this is one of the most expensive options since hiring talent who are capable of writing code, which is complex enough to handle concurrency, is going to be expensive.

One-third possibility is actually to wait, wait few years before one of the concurrency inbuilt virtual machine comes along to build the software. Although it's not a wise decision to wait in the competitive world to fulfill a need, it is best to do something and get it right rather than fail undertaking a risky project.

But at any cost managers should not encourage single threaded application development for any software component that is going to be enhanced and built upon on the long run.

References

1. Dana Blankenhorn, The Blankenhorn Effect: How to Put Moore's Law to Work for You, 2002, Published by Trafford Publishing

2. Mark Balch, Complete Digital Design: A Comprehensive Guide to Digital Electronics, Advance Microprocessor concepts, 2003, Published by McGraw-Hill Professional

3. Press Release, Sun Microelectronics Hits Key Milestone in High-End UltraSPARC Development, 2007, Retrieved on 31 May 2007, Retrieved from http://www.sun.com/aboutsun/pr/2007-05/sunflash.20070502.1.xml

4. Abdelsalam A. Helal, Bert Haskell, Jeffery L. Carter, Richard, Any time, anywhere computing: Mobile Computing Concepts and Technology, 1999, Published by Springer

5. Magnus Ekman, Fredrik Warg, Jim Nilsson, An In-Depth Look at Computer Performance Growth, 2004, Published at CHALMERS UNIVERSITY OF TECHNOLOGY, Department of Computer Engineering, Retrieved on 29 May 2008, Retrieved from http://www.ce.chalmers.se/~warg/papers/performancegrowth_tr-2004-9.pdf

6. OOfri Wechsler, Inside Intel®Core™ Microarchitecture Setting NewStandards for Energy-Efficient Performance, Intel Corporation, 2005, White Paper Published by Intel.

7. Herb Sutter, The Free Lunch Is Over A Fundamental Turn Toward Concurrency in Software, 2005, Microsoft Architect, Published by Dr. Dobb's Journal, Retrived on 26 May 2008, Retrieved by http://www.gotw.ca/publications/concurrency-ddj.htm

8. Johan De Gelas, The Quest for More Processing Power, Part One: "Is the single core CPU doomed?, 2008, Retrieved on 28 May 2008, Retrieved from http://www.anandtech.com/cpuchipsets/showdoc.aspx?i=2343

9. Paul Farhi, CNN Hits The Wall for the Election, 2008, Washington Post, Retrieved on 28 May 2008, Retrieved from http://www.washingtonpost.com/wp-dyn/content/article/2008/02/04/AR2008020402796.html

10. Neil Vachharajani , Matthew Iyer , Chinmay Ashok,Manish Vachharajani , David I. August, and Daniel Connors, Chip Multi-Processor Scalability for Single-Threaded Applications, No Year, Publsihed at University of Colorado.

11. Unattributed Article, Understanding the Java Platform Architecture, Sun Microsystem Publication, No Date, Retrieved on 26 May 2008, Retrieved from http://www.sun.com/software/opensource/java/intro_java_tech.jsp

12. Tim Bray, Erlang Blues, 2007, Co-Inventor of XML, Retrived on 28 May 2008, Retrieved from http://www.tbray.org/ongoing/When/200x/2007/09/22/Erlang

13. Martin LaMonica, Ajax spurs Web rebirth for desktop apps, 2005, Published by CNET, Retrieved on 27 May 2008, Retrieved from http://business2-cnet.com.com/AJAX+spurs+Web+rebirth+for+desktop+apps/2100-1012_3-5977268.html

14. Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Google Fellow Engineer, Retrieved on 29 May 2008, Retrieved from http://labs.google.com/papers/mapreduce.html

15. Joshua Engel, Programming for the Java Virtual Machine By Joshua Engel, 1999, Published by Addison-Wesley