OpenSPARC: How Giving Away CMT Chip Hardware Implementations Creates Value for Sun Microsystems

By
Julie Mitchell

Bachelor of Science in Mechanical and Aerospace Engineering
Cornell University, 2000

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

February 2009

Signature of Author_____
Julie Mitchell
System Design and Management Program
Feb 2009

Certified by_____
Michael Davies, Thesis Supervisor
Senior Lecturer, Sloan School of Management

Accepted by_____
Pat Hale
Director, System Design and Management Program

OpenSPARC: How Giving Away CMT Chip Hardware Implementations Creates Value for Sun Microsystems

by
Julie Mitchell

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering and Management

Abstract

This thesis uses systems thinking and system dynamics modeling to explore how open source communities such as OpenSPARC can lead to enhancement of the performance of Sun's multithreaded systems and thereby increase its market share by increasing its share of the CMT ecosystem, and the share that CMT systems have within the overall computer server business ecosystem. This study explores Sun's motivation behind its investment in the OpenSPARC community, and explains how OpenSPARC creates value for Sun. The insight into Sun's value creation and capture strategy gained from this study can be generalized and applied to similar companies who are entering a new market where the ecosystem is not yet fully developed. The companies who benefit most from this study are ones which are strategically positioned to disclose relevant knowledge of a critical component within the ecosystem that enables its development without thereby compromising the full potential for value capture within the market.

The key findings of this study include:

a) Market adoption of multitcore and multithreaded servers is dependent on the rewriting of software applications with parallelization in order to boost the performance of multicore and multithreaded systems.
b) The overhaul of these software applications will take the coordination and involvement of all the major players in the computer industry.
c) The specific business structure of Sun allows for open sourcing the components of its systems while still gaining revenue on the sale of the system as a whole.
d) Factors attracting open source developers to write software for a particular platform include a developer's belief that his program will be successful in the market.
e) The information leakage to competitors from open sourcing Sun's multithreaded processor implementation does not diminish Sun's value capture of the market.
f) The benefits that open source communities can have on market adoption of multicore and multithreaded servers, provided that the disclosure of the chip hardware implementation actually improves the technical performance and economics of the application software.

This thesis will explore the reasons that Sun believes the open source community can be a catalyst for the wide-spread adoption of multithreaded processors and multithreaded software required simultaneously by all the major players in the computer industry.

Thesis Supervisor: Michael Davies
Senior Lecturer, Sloan School of Management

<This page has been intentionally left blank>

Acknowledgements

Most importantly, I'd like to thank my husband Paul Mitchell Jr. for his never ending support, encouragement, and guidance throughout this thesis and my completion of my Master's degree program. This paper would not have been possible without his advice and insight.

I'd like to thank my thesis advisor, Michael Davies, for his insightful advice and continuous coaching on the content and direction of this paper. He provided invaluable insight into areas of importance that may have otherwise been left unexplored.

Next, I'd like to thank and acknowledge the OpenSPARC technical experts at Sun who provided input, data and guidance along this journey, specifically Ricky Hetherington, David Weaver, Shrenik Mehta and Catherine Ahlschlager. Their input helped to provide much of the background and rich technical content throughout the paper.

I'd also like to thank my management and mentors at Sun who have helped me reach my goal of completing this degree program, specifically Sue Graham Johnston, Tony Devito, Angus McAlpine, Joan Cullinane and John Baxter. They provided the tremendous support and encouragement for me, both personally and professionally, which was greatly needed in order to achieve this accomplishment. For this I will always be grateful.

I'd like to thank the SDM Program directors, Pat Hale and Jack Grace for their guidance throughout the pursuit of this Master's degree program. They have helped make this experience more rewarding then I could have ever expected.

Lastly I'd like to thank my family, and new found friends and colleagues in the SDM cohort who continue to provide a rich environment of learning and discovery which challenges me daily to grow open my mind to new perspectives and possibilities.

<This page has been intentionally left blank>

Table of Contents

List of Figures

Chapter 1: Introduction

For over half a century, advancements in computer hardware have been following a well-known trajectory, the so-called Moore's Law. Moore's Law states that the number of transistors on a microchip will double every two years starting in 1959. This law, in turn, has become synonymous with the belief that system performance will double every two years as a result of processor speed increases. After 50 years of holding accurate, Moore's law is now at risk as we are approaching the physical limits of silicon and as other elements of the overall system become the bottleneck and thereby constrain overall system performance. As a result, the computer industry has embarked on a new path to increase the number of single compute cores on a chip and develop multithreaded software to increase the speed of computing at the system level. All major chip design manufacturers today, including Intel, Advanced Micro Devices and Sun Microsystems produce multicore, multithreaded chips ranging from 2 cores up to 8 core processors.

Chip multithreading (CMT) refers to the ability of a computer processor to process multiple software threads simultaneously. A thread is defined as a sequence of executing instructions that can run independently of other threads. A CMT processor can be designed to implement this capability using a variety of methods, which primarily include having multiple cores on a single chip and multiple threads on a single core, where a core is defined as central processing unit and the chip is defined as the integrated circuit that can contain one or more central processing units. In contrast, the traditional architecture of computer processors has been single cores with only single thread computing capability. This single-core, single-thread traditional architecture requires the CPU to sit idle while waiting for required data to return from memory for the next calculation. Critically, fetching data from memory takes time because memory access is slow relative to processor performance, so as a result, single-core single-threaded processors are stalled as much as 75 percent of the time while they wait for memory to fetch data. Thus, "memory latency", defined as the time between initiating a request for data until it is received and can be processed, has become the key bottleneck, the single most important limiting factor in improving single-core chip performance. The graph below illustrates

the gap that has been growing between the speed of the CPU and the speed of Memory (DRAM) over time:

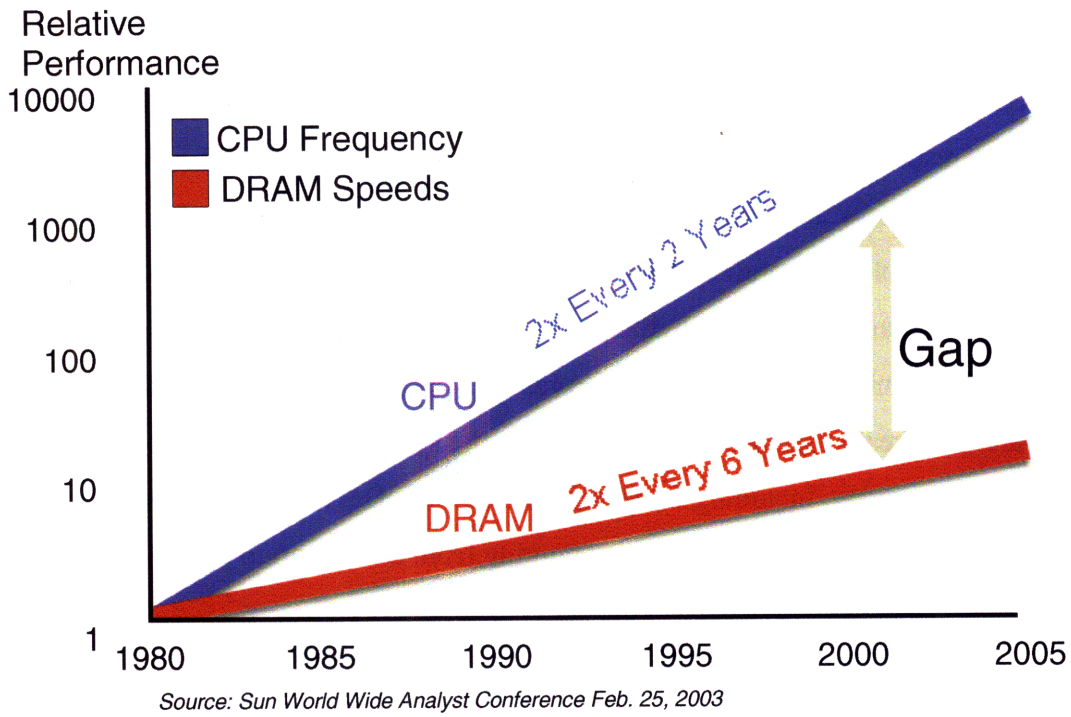


Figure 1: Memory speed growth over time verse CPU clock speed growth

Due to the multithread processing capability of CMT processors, when one thread is stalled waiting for memory, the core starts processing the next thread, thereby eliminating most of the wait time from the overall processing time. As a result, a chip multithreaded architecture greatly increases the overall efficiency of the system. The following picture illustrates the breakdown of the total processing time between computing time and memory fetch time in a single threaded computer processor.

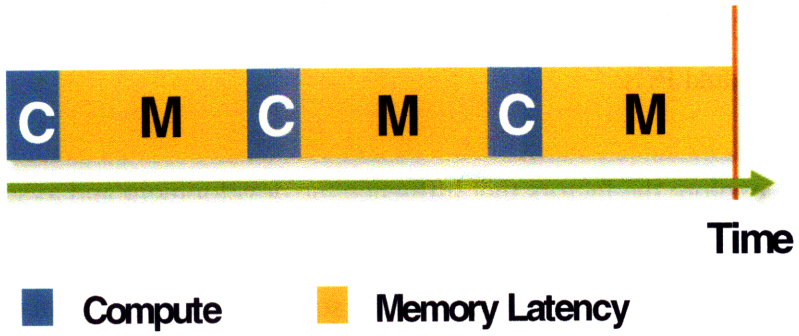


Figure 2: Single Threaded Computer Processor

In contrast, a multithreaded processor is able to execute multiple threads in parallel or to stack single threads on top of each other in order to maximize compute cycles of one thread while the other threads are idle waiting for memory fetches, per the picture below:

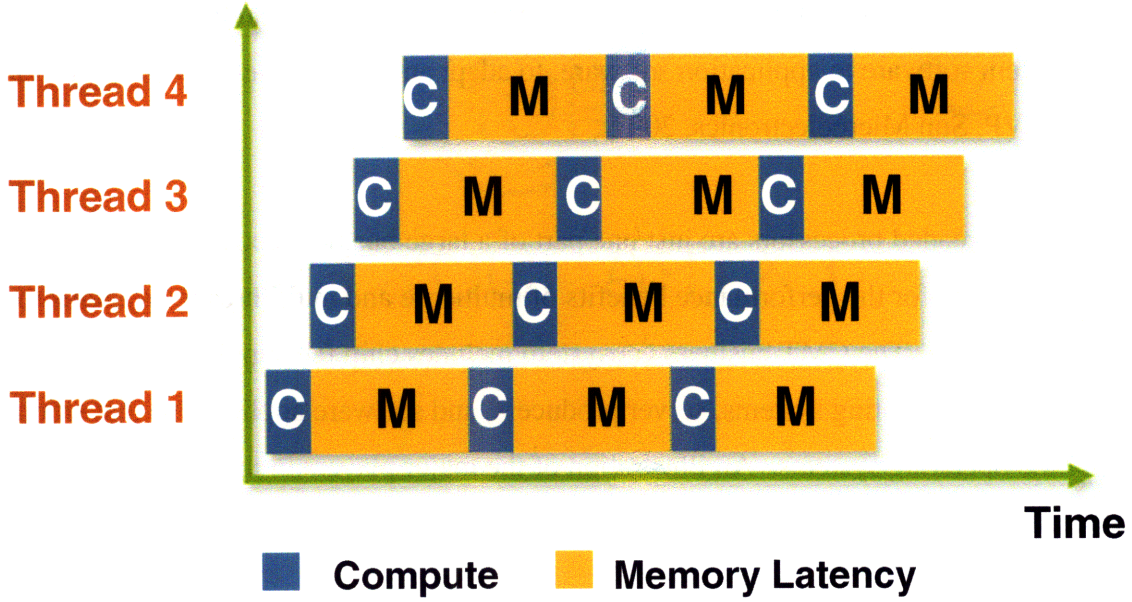


Figure 3: Multithreaded CPU Processor

The higher processor utilization that can be obtained this way is particularly important for servers because they are a shared resource, while individual PC's may not benefit as much from parallelization since they spend a lot of time idle, waiting for individual user

input. Thus, the systems that are focused on and referred to throughout this study are servers, not individual PCs.

This thesis will explore the factors that influence the development and availability of parallel software applications, and determine the related key assumptions that have been reflected in Sun Microsystems' decision to open source two of its system components, the operating system (OpenSolaris) and the multithreaded chip hardware implementation (OpenSPARC).

1.1 The CMT Ecosystem

"We strongly believe this multicore, multithreaded direction is the way to go, both for efficiency in computing and efficiency in power consumption, but this can not be done alone by hardware processor vendors. It requires the software community, all the way from system software to application software, to adapt and participate." –Dr. David Yen, former EVP, Sun Microelectronics, 2007

Chip multithreaded processors are just one part of a large business ecosystem that needs to exist in order for the performance benefits of multicore and multithreaded systems to be fully realized. The CMT business ecosystem includes chip fab suppliers, chip design manufacturer, operating systems, server producers, and software applications that sit on top of the operating systems. An illustration of the major players in the CMT Ecosystem can be seen in Figure 3 below.

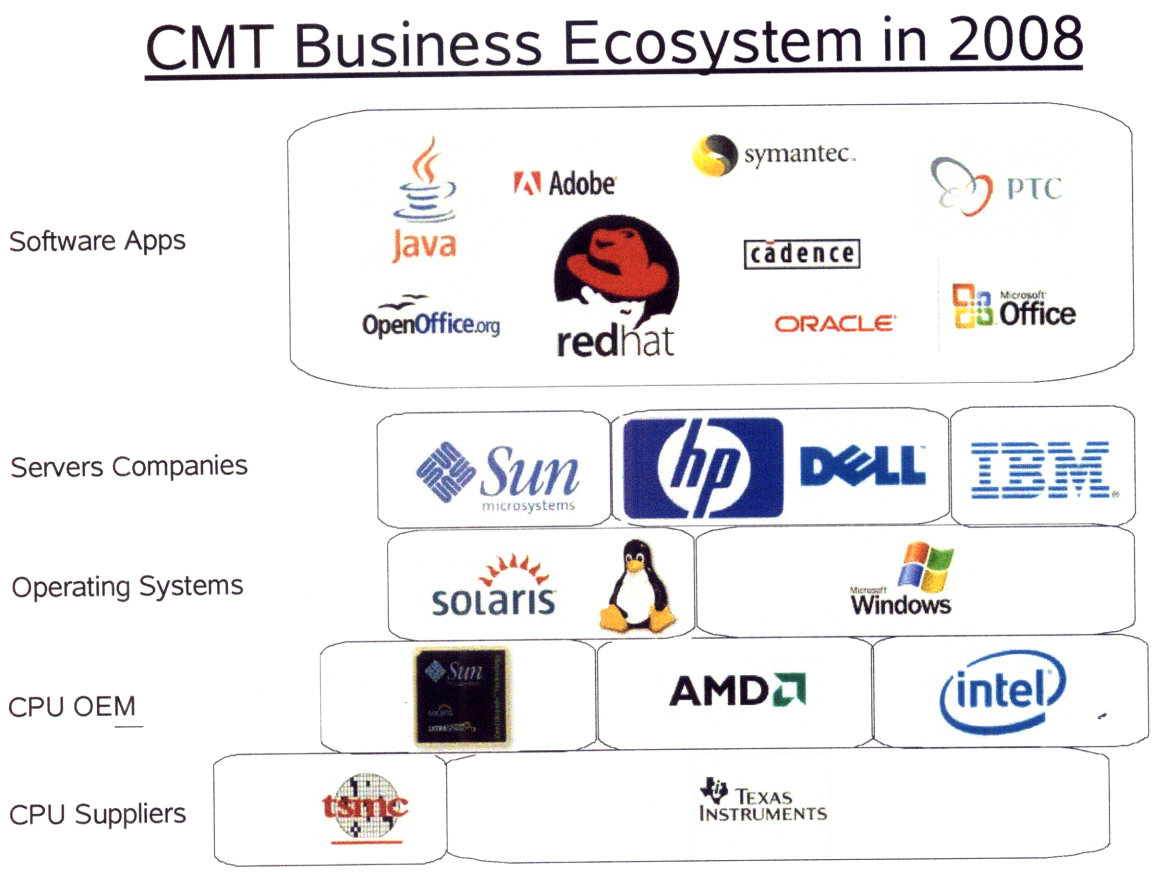


Figure 4: CMT Ecosystem 2008

A coordinated investment and development by all the players in the ecosystem must take place in order for CMT to become the dominant next generation processor design. The CPU suppliers need to have the infrastructure and capability to build and test CPU wafer fabs with multiple threads and cores. CPU OEM designers need to have the skills and knowledge needed to design multithreaded chips. The operating system developers need to be able to write operating system software that supports multithreaded environments. Lastly and most importantly, the software applications need to be written to take full advantage of the new chip architecture. Without parallel software applications available to utilize the improved performance of multithreaded chip architectures, the performance potential of the multithreaded systems will not be realized. According to Ashlee Vance at The Register, "Software companies – Microsoft included – will need to rewrite their applications *en masse* to take advantage of these new chips that use multiple low-power

cores instead of just increasing the speed of a single core chip to boost overall performance."[2] In order for CMT processors to be fully adopted by the industry, the entire ecosystem must invest time and resources to make the shift to the new architecture, from the manufacturing processes to the software implementation. This required paradigm shift is at the heart of the effort behind developing OpenSPARC. This thesis will test the assumption by Sun Microsystems that the most effective way to get other members of the ecosystem to adopt CMT, and invest in the required supporting technologies, is make information about its chip multithreaded hardware implementation widely and freely available.

1.2 What is OpenSPARC?

"By making the source for this design available for a larger community to review and learn from, we expect that ideas around the multi-thread concepts can be explored more freely and openly, and that truly beneficial innovations can be achieved." -Opensparc.net

In 2005, Sun released the Niagara T1 64 bit CMT processor to market. This 90-nanometer T1 chip features 32 threads, eight cores with four threads each. In 2007, Sun released the T2 processor, a 65-nanometer chip with a total of 64 threads, eight cores with eight threads each. Both of these multicore and multithreaded processors quickly became the fastest commodity processors available on the market at their time of release, holding two world-record single-chip SPEC CPU results.

In 2006, Sun launched OpenSPARC, which is a web-based community where visitors and members are given access to free downloads of the Sun Niagara T1 and T2 64 bit CMT microprocessor design implementation. Within the community, members will find the CPU specifications, including detailed functional descriptions of the OpenSPARC T1 and now T2 Processor components, a description of the architecture of each of the processor components, detailed block diagrams, and a signal list for each component. Chip designers and verification engineers will also find downloads available for Verilog RTL, a verification environment, diagnostics tests, scripts and tools needed to simulate the design and to do synthesis of the design, open source tools needed to simulate the

design, and scripts and documentation to help with FPGA implementation of parts of OpenSPARC T1 and T2 design.

In addition to the software downloads available at opensparc.net, a student or developer can sign up to receive a free Xilinx FPGA board which dramatically simplifies the learning curve needed to get started in creating next generation applications tuned specifically for multithreaded parallel computing environments. The purpose of the Xilinx board is to allow a developer or a student to physically input their program into a simulated 4-core CMT OpenSPARC chip by using the inherent parallel capabilities of an FPGA, which consists of the three primary components; registers, function generators, and the FPGA clock. Each function generator contains a set of logic gates, and can be programmed to execute in parallel with each other, synchronous with the clock cycles. The Xilinx board thus acts as a reference design that allows a scaled-down version of the OpenSPARC T1 processor to run on Xilinx Virtex 5 FPGAs.

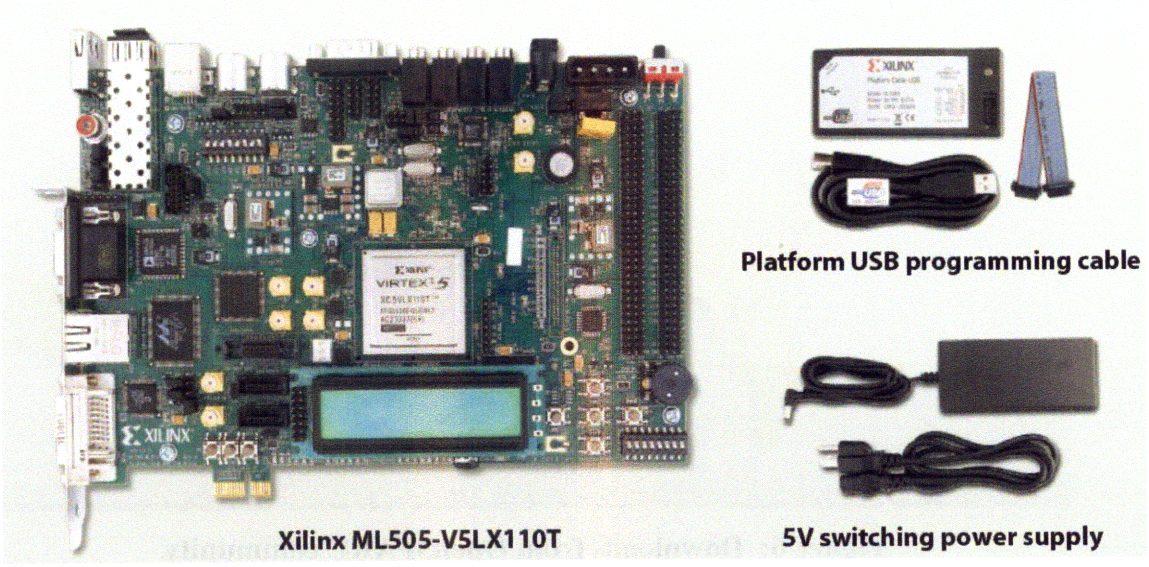


Figure 5: Xilinx ML505 FPGA board with the large Virtex 5 - V5LX110T FPGA

1.3 Who participates in the OpenSPARC Community?

The OpenSPARC community involves participants from over 130 countries worldwide. The most active members are the Center's of Excellence in academia, which include the University of California Santa Cruz, University of Texas, University of Michigan, University of Illinois, Carnegie Mellon University, Stanford University, University of Otago (Dunedin, New Zealand), and Peking University (China). Sun has also undertaken a collaboration agreement with the Ministry of Education for the People's Republic of China. In the figure below, the global participation of the community is demonstrated by the more then 10,245 hardware downloads and more then 7,294 software downloads from over 130 countries worldwide.

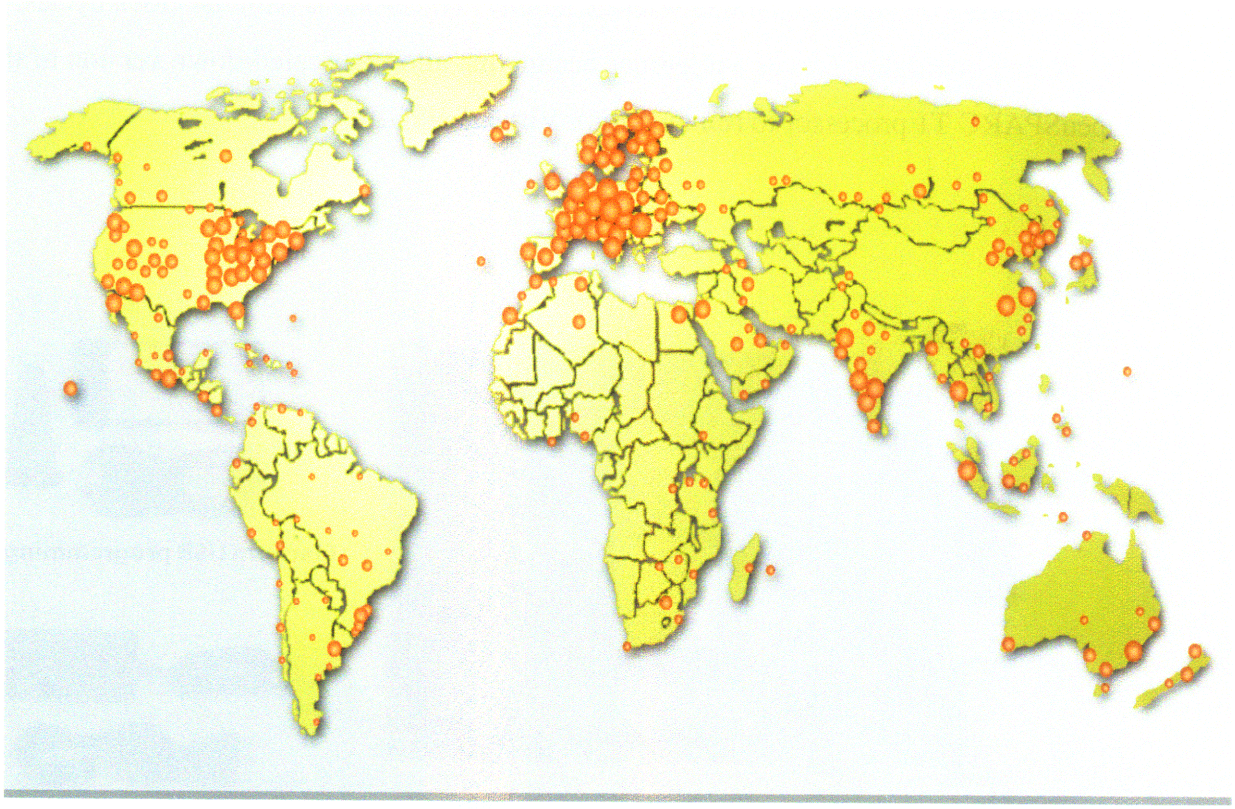


Figure 6: Downloads from OpenSPARC community

Chapter 2: Hypothesis

This study models the business ecosystem of multithreaded processors and parallel software in order to understand the impacts of open sourcing the hardware chip

implementation of Sun's T1 multithreaded processors on the ecosystem. By open sourcing the hardware implementation of the most advanced CMT processor available on the market today, Sun has demonstrated its belief that the open source approach will foster a community around innovation in multithreaded chip design. The industry has recognized that the software application community is dramatically lagging behind hardware in developing applications optimized for a multithreaded processor design. This delay in the availability of parallel software applications is negatively affecting multithreaded server performance since the utilization of multithreaded processors depends greatly on the parallelism of the software used to run the system. By open sourcing its chip hardware implementation and parallel software simulation tools, Sun has expressed its intention to encourage innovation and rapid growth in parallel software development.

Sun's use of the open source community in this way also reflects its underlying assumption that releasing detailed knowledge about the hardware design of its CMT processors will either directly or indirectly make writing parallel software applications easier for the developer community. The strength of the relationship between having detailed hardware knowledge of the processor and the ability to write software applications optimized for that processor has a significant impact on whether Sun will be successful in its mission to use the OpenSPARC community to help stimulate the development of parallel software for multithreaded processor architectures.

This thesis uses system dynamic modeling to test the mental models and assumptions about risks and rewards that Sun Microsystems used to make the decision to open source its multithreaded chip hardware implementation.

The key assumptions that are tested in this thesis using a system dynamics modeling approach include:

1. Performance of multithreaded systems is the primary factor driving their adoption in the marketplace.

2. The key to improving multithreaded system performance is increasing the degree of software parallelism available to utilize the multithreaded architecture of the processors.
3. Using the open source communities is a more efficient way to develop the supporting software needed for performance improvements of multithreaded systems, based on the underlying assumption that releasing detailed knowledge about the chip hardware implementation will help software application developers to write parallel code optimized for multithreaded chip architecture. Developing software internally or waiting for third party software companies to invest their resources in parallel software development would not have been an equal alternative to using the open source communities for innovation and development.
4. Information leakage to competitors from open sourcing key components of Sun's multithreaded systems does not outweigh the benefits Sun receives from using the open source communities to increase the performance of Sun's multithreaded systems. This information leakage is acceptable to Sun because its revenue is earned from its selling an integrated system to the market, and it can afford to open source the components, specifically the CPU and operating system, in order to improve the overall system performance and sales in the market.

2.1 The Use of System Dynamics

This thesis uses system dynamics modeling in order to explore the mental models and understanding about the multithreaded business ecosystem that drove Sun's decision to release its multithreaded T1 chip hardware implementation to the open source market. The system dynamics model assists in understanding the feedback loops and relationships between Sun, the open source communities, Sun's competitors, third party software developers and the CMT market. The model can be used to determine if the losses that Sun will incur due to this action will be negligible compared to the gains it will see from the feedback loops back from the open source communities by analyzing the relative

strengths of the dynamic loops between Sun, its competitors and the open source communities.

2.1.1 What is System Dynamics?

System dynamics and systems thinking is a methodology for understanding the behavior of complex dynamic systems. This methodology assumes a system of causal relationships between the components or agents within the system, and a presence of circular feedback loops within the causal relationships that interact over time and govern the long-term behavior of the system. Components of a system dynamics model include causal loop diagrams, feedback loops, accumulation of flows into stocks, and time delays[25].

2.1.2 Why was this method chosen?

System Dynamics was the method chosen to model the behavior of the system since we are studying a system that has dynamic complexity consisting of feedback loops and long time delays. Complexity is created when the decisions of one agent affect the other agents in the ecosystem. The responses of one agent to the other are not immediate, and can have significant time delays. System dynamics is a tool that helps us understand the influence of one agent's actions on the system over long periods of time. The next section outlines the principles that guided the development of the system dynamics model in this thesis.

2.1.3 Principles for Successful Use of System Dynamics:

Effective system dynamics modeling must comply to the following principles in order to facilitate the gain of insight into the behavior of the system:

1) Develop a model to solve a particular problem, not to model the system. Focus the purpose of the model on the problem you are trying to solve.
2) Modeling should be integrated into a project from the beginning. The modeling process should focus on the diagnosing the structure of the system.

3) Modeling works best as an iterative process. Modeling is a process of discovery. The goal is to reach new understanding of how the problem arises and then use that understanding to design high leverage policies for improvement.
4) A broad model boundary is more important than a great deal of detail. Models must strike a balance between a useful, operational representation of the structures and policy levers available to the clients while capturing the feedbacks generally unaccounted for in their mental models[25].

2.2 Other Methods Used

The system dynamics models were created through an iterative process using data collected from literature reviews as well as interviews with experts in the industry. Two methodologies were used to collect the data required to complete the model. The first method is through interviews of various CMT community members. These members include CMT experts that reside at Sun Microsystems, industry and academic contributors to the CMT community. The second method includes an in depth literature review on open source models, their risks and benefits, and the revenue streams employed by open source software companies. The data collected through the literature review and interview process has been feed into the system dynamics modeling process in order to create the relationships contained within the models and to derive understanding about the assumptions and mental models of the structure of the system.

Chapter 3: Literature Review and data collection

3.1 Multithreaded Systems Require Parallel Software for Performance

"As with its open-source software plans, the OpenSparc project is a bid for relevance first and revenue later." - Stephen Shankland, 2007, news.com

System performance of multithreaded servers is very dependent on parallelization of the software applications that drive the system. Because of this inter-dependency, the CMT

ecosystem must evolve through collaboration between hardware and software companies that produce the different layers of the system hardware and software in order to maximize system performance and value to the customers. Sun Microsystems has recognized that, despite the fact it had the most advanced multithreaded chip hardware available to the market, the technology would be irrelevant without the support of the surrounding software ecosystem needed to utilize its capability. Sun is looking towards open source communities such as OpenSolaris and OpenSPARC in order to grow the necessary software ecosystem required to accelerate the growth and adoption of CMT processors. According to Francis Allen, IBM Fellow and 2006 Turning Award Recipient, "the problem of parallelism has been around since multiprogramming systems. Hardware has aggressively added more parallelism to its capabilities, but software hasn't kept pace.", 2008, Grace Hopper Conference.

3.2 Impact of Parallelism on System Performance

From ideas presented in the previous section of this study, we realize that in order for the true value of multithreaded systems to be realized by the market, the systems must demonstrate higher performance capability then the traditional single core, single threaded systems available currently in the market. Through data collected by Glenn Fawcett of Sun Microsystems, we can gain some insight about what performance can be expected from Sun's T5240 server, which is 2-socketed system with Sun's T2 CMT CPU's with 8 threads each, running parallelized software applications. This section will outline two insightful experiments run on the T5240 that validate the claim that improved system performance will be achieved on multithreaded systems as a result of parallel software. The experiments look at two different cases for software functionality:

1) Executing Oracle commands
2) Installing and compiling open source software into a system's software stack

.

In the first study[23] conducted in May, 2008, Fawcett runs specific Oracle commands on a Sun server, T5240, which is a two socket system with a total of 16 CPU cores. The specific command ran was "Create as Select", and by using built-in parallelism features

within Oracle software, Fawcett was able to rerun this command while increasing the parallelism of the software from 1 to 16 cores. The time the system takes to execute the command dropped by 25 minutes between running it with no parallelism to running it on 8 threads. Hence, the command achieved nearly a 10x speedup by using the parallelism features already built into the Oracle software. A graph of Fawcett's results can be seen below:

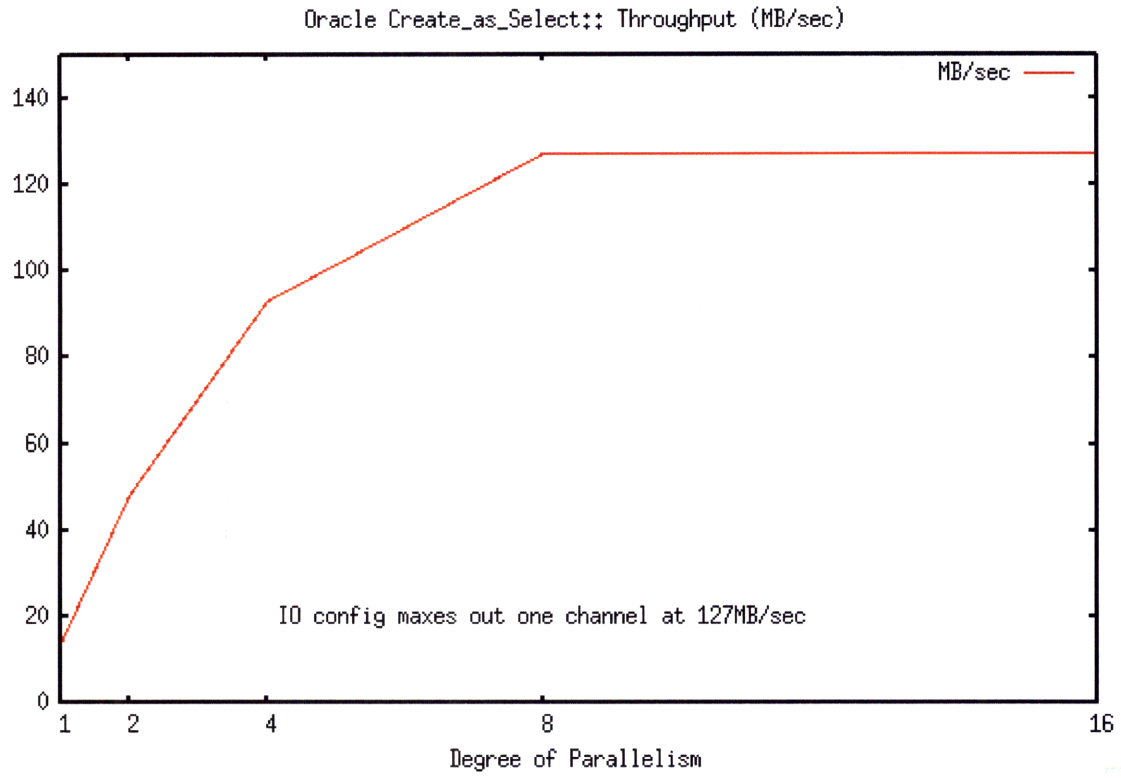


Figure 7: Fawcett's results on improving system performance using parallelism

To further emphasize the understanding that parallelism greatly improves system performance, Fawcett conducted another experiment[24] using Sun's T5240 servers, which was focused on the configuring and compiling of 5 popular open source software packages:

- httpd-2.2.6
- mysql-5.1.22-rc
- perl-5.10.0
- postgresql-8.2.4

- ruby-1.8.6

Fawcett conduced his experiment by compiling these programs using the "gmake –j" command which allows the level of parallelism to be specified for each of the packages. The installation time of these packages were then measured. We can see from Fawcett's results below, that the fastest compile times were seen when 16 or higher threads were utilized.

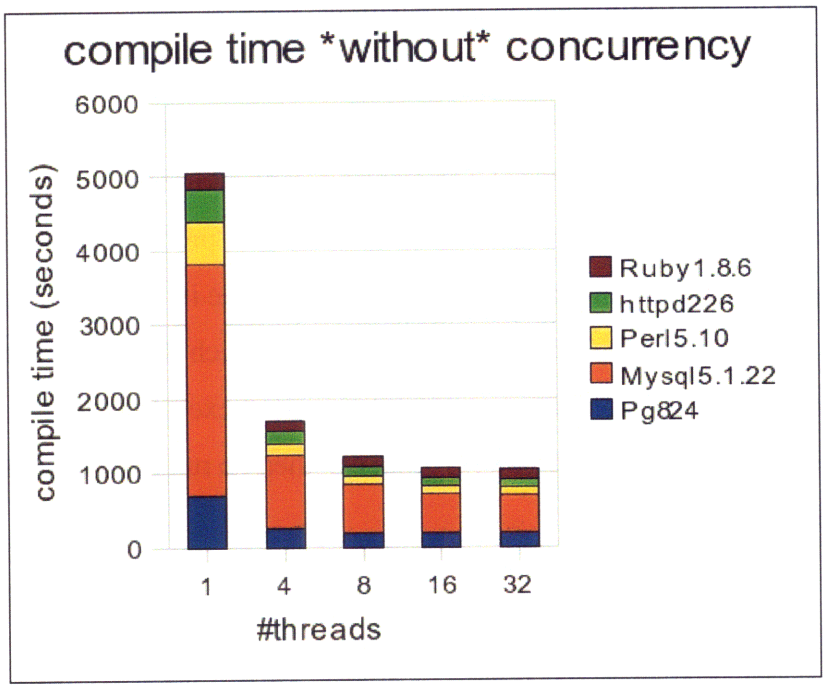


Figure 8: Fawcett's results on improving compile time using parallelism

3.3 Parallel Programming is Difficult

"Sun recognizes that this multicore, multithreaded approach isn't easy for some necessary partners in the software industry to digest. Trying to encourage their support is another reason Sun opted for the OpenSparc project" - Stephen Shankland, 2007, news.com

In order to understand the mind-shift required to encourage developers to write parallel code verse the traditional serial code, we have collected data from literature[22] on what the software community thinks about parallel processing. This data provides insight into the perceptions that must be addressed within the development community in order to

encourage more developers to write parallel software. In a book entitled Software Pipelines and SOA: Releasing the Power of Multi-Core Processing published on Dec 26, 2008, author Cory Isaacson conducts a statistical analysis on data collected from web documents which cross reference the following terms: multicore, multithreaded, parallel processing, and parallel programming with the associated terms: complex, hard, important, knowledge, and useful.

The following four graphs illustrate the associations that exist between the perception of the community on Parallel Processing, Parallel Programming, Multicore, and Multithread. *Parallel Processing* is defined as the simultaneous use of more then one CPU to execute a program, and *Parallel Programming* is defined as writing software code that is parallelized. The term *Multicore* is defined as a processor that combines two or more independent CPU's into a single package of a single integrated circuit, and lastly *Multithreaded* is a term that is shorthand for systems in which there are multiple "threads" of execution of a computer program; a thread of execution generates a "child process" from the original "parent process" that can then be run independently of the original process.

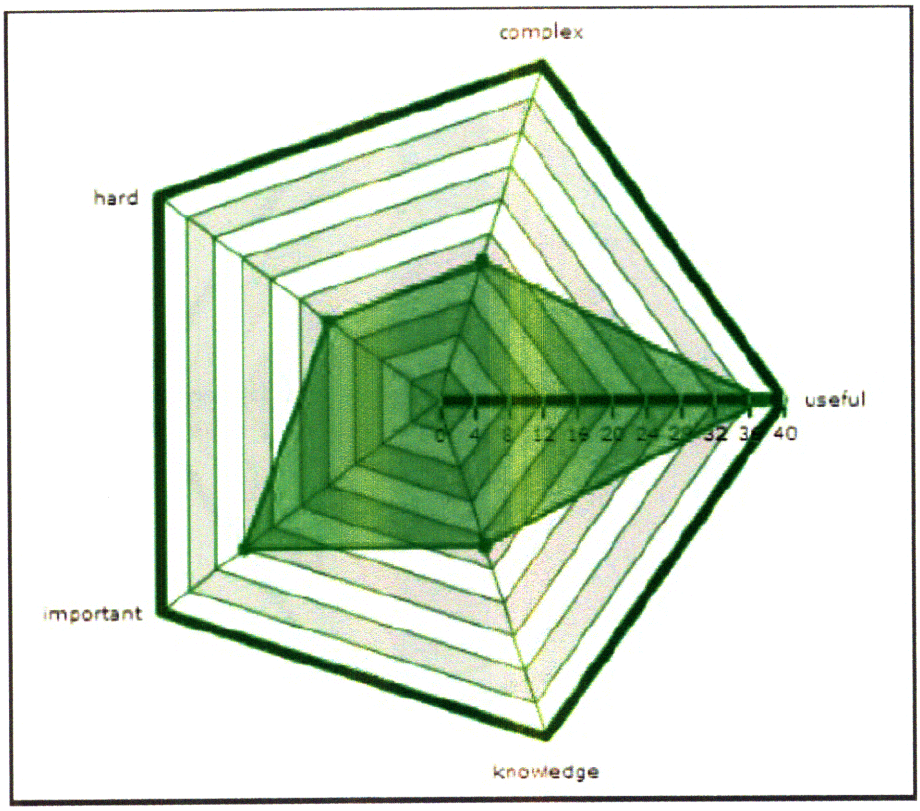


Figure 9: Perception of Parallel Processing (Isaacson, 2008)

The graph above shows that the community believes *Parallel Processing* is extremely useful and important.

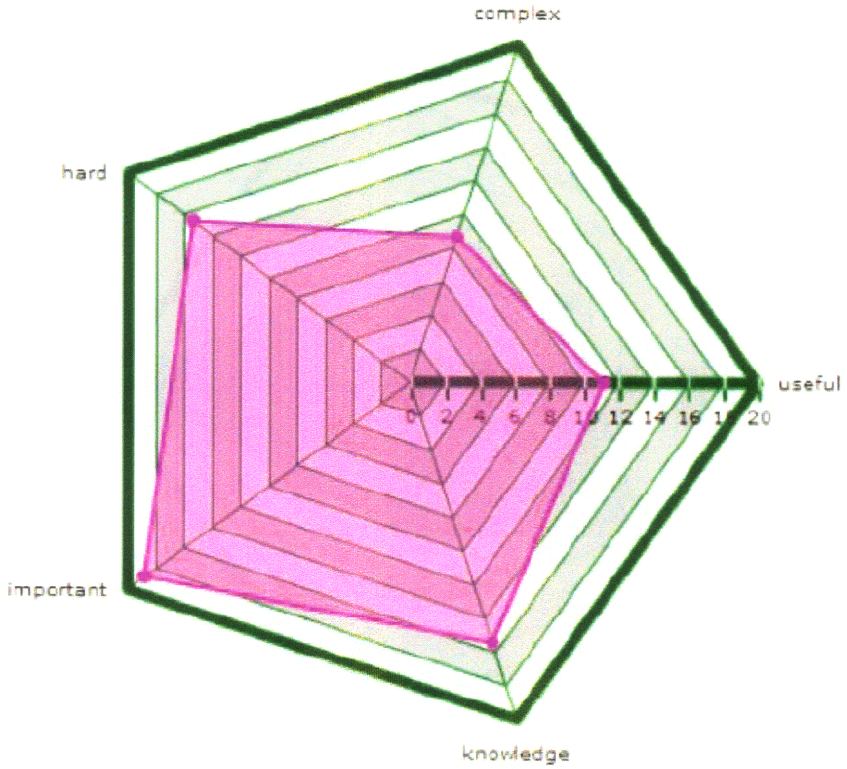


Figure 10: Perception of Parallel Programming (Isaacson, 2008)

However in contrast to Figure 9, the perception of *Parallel Programming*, shown in Figure 10, is that it is very important, but very hard, requires a lot of knowledge, and is not very useful. The following two graphs for *Multicore* and *Multithread* reveal interesting insights into the difference in perception amongst these various terms of parallel processing and programming and multicore and multithread. The perception of these two terms, *Multicore* and *Multithreaded* among the software community can be seen in the two graphs below.

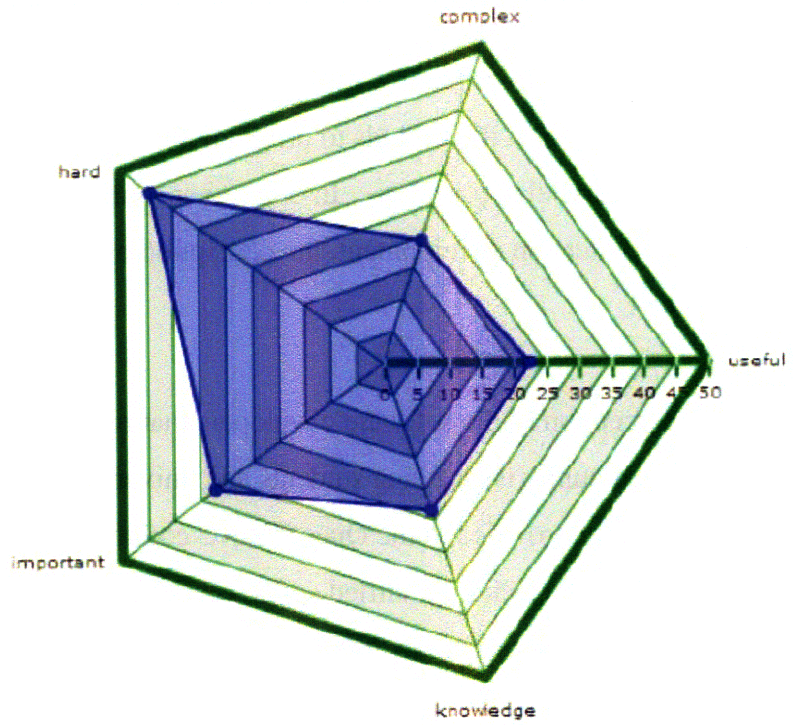


Figure 11: Perception of Multicore (Isaacson, 2008)

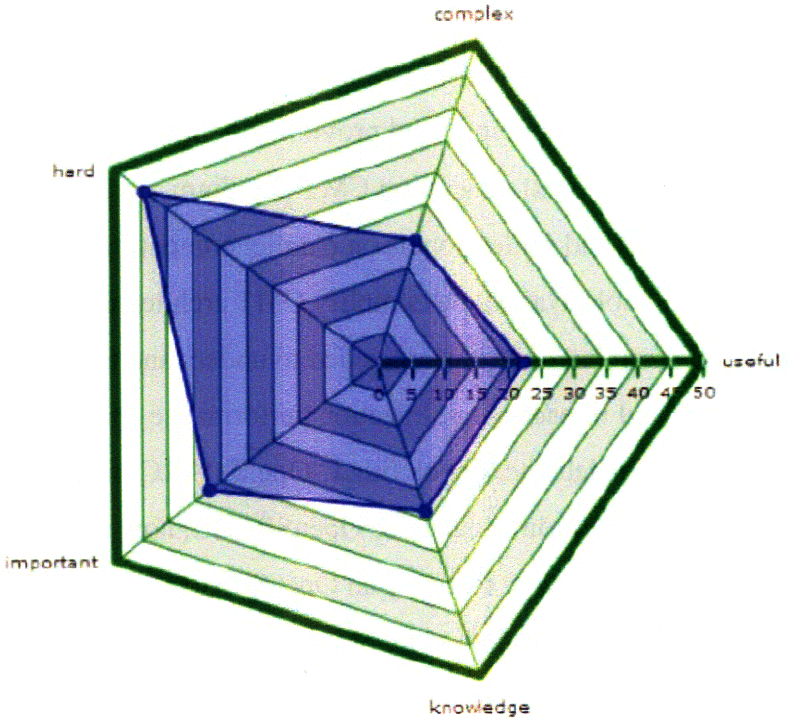


Figure 12: Perception of Multithread (Isaacson, 2008)

The perception of *Multicore* and *Multithread* both illustrate the strongest association that they are both "hard". The conclusions that we can draw from this data is that the largest barrier to getting developers to shift their minds to program in parallel is the perceived difficulty factor in writing parallel applications. In order to encourage more developers to join parallel software application development, the perception of the degree of difficulty must be reduced.

In order to change the perception of the software developers that parallel programming is hard, the OpenSPARC community is targeting students and universities as a primary target audience for the content provided in the OpenSPARC community. The perception of difficulty might stem from the lack of qualified software engineers available in the workforce. In an IEEE article[26] published in 2000, a panel of four experts state that "One of the most serious issues facing the software engineering education community is the lack of qualified tenure-track (full-time) faculty to teach software engineering courses, particularly at the undergraduate level. Similarly, one of the most serious issues facing the software industry is the lack of qualified junior and senior software engineers."

Sun Microsystems has recognized that there are few experts available in the industry who are capable of developing the technology and providing the innovation needed to move the computer industry into the next generation CMT paradigm. In an effort to contribute to the education of students and professionals on parallel computing, the OpenSPARC project has made several tools available to simplify CMT architecture so that college students and developers can learn the basics needed to understand the technology. These tools include compilers and threading optimization performance analysis. Sun has also provided a free and downloadable text book titled "OpenSPARC Internals"[27] which includes chapters on how to customize and use OpenSPARC, how to set up a simulation and emulation environment and how to verify an OpenSPARC design. This textbook has been utilized by several universities including Stanford University, Tsignhua University, and Carnegie Mellon University and is intended to supplement coursework for students in the area of parallel program development as well as next generation chip architecture curriculums.

3.4 Parallel Programming Tools available in OpenSPARC

According to a study[18] conducted by Funk, Basili, Hochstein, and Kepner on parallel software development, a comparison of a parallel application's performance to that of a baseline serial application, and a comparison of the effort required to develop the parallel application to the effort required for the serial version can be characterized through classroom studies in which students write two versions of the same software application using a serial architecture and a parallel architecture. Funk, Basili, Hochstein, and Kepner use the data from their classroom experiments to create a metric of relative development time productivity (RDTP) which is defined as the ratio between the improved performance of the application as a result of the parallel structure over a traditional serial structure (labeled as "speed up") and the increased effort required by the software developers in order to write the program using a parallel architecture over a serial architecture (labeled as "relative effort"). This ratio can be seen in the figure below:

$$\psi_{\text{relative}} = \frac{\text{speedup}}{\text{relative effort}}$$

Figure 13: Characterization of RDTP from Funk, Basili,Hochstein and Kepner

The RDTP metric characterizes the performance improvement of an application written in a parallel architecture verse a traditional serial architecture. Funk, Basili, Hochstein, and Kepner conclude that for their classroom experiments, the OpenMP software development toolkit provides the best improvement in parallel program speedup with only a minor increase in effort involved in writing the parallel version of the application. According to the study comparing the following toolkits: MPI, OpenMP and Matlab's StarP; the authors concluded that students were most productive in developing the parallel software application when using OpenMP. More importantly, the relative effort of development required using OpenMP is lower or equal to serial development, and the

achieved speed-up is much greater, as is seen in the following charts which show RDTP of 6 or greater using OpenMP.

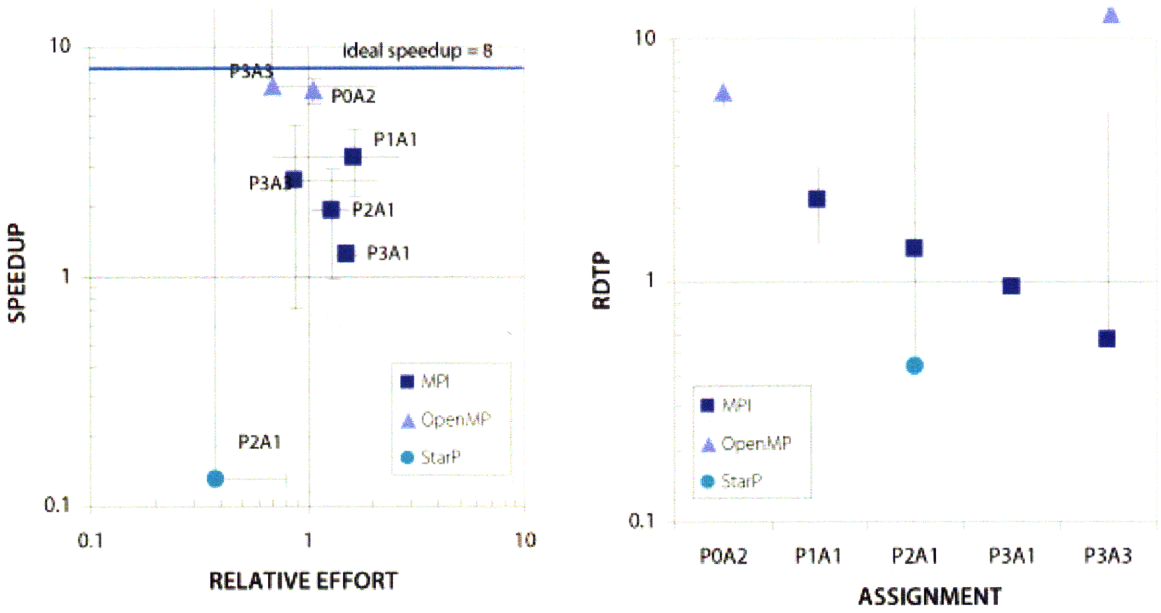


Figure 14: Results of RDTP metrics using different parallel software development tools

The results of these experiments are relevant to understanding the expected causal relationship between the developer tools that Sun is offering as part of the OpenSPARC community and increased parallel software application development from the open source community. As part of the tools Sun makes available on OpenSPARC community, Sun provides a free downloadable package of the Sun Studio toolkit, which utilizes the OpenMP application programming interface to assist the developer in writing code for parallel applications. According to Nawal Copty of Sun Microsystems, "OpenMP is an Application Programming Interface (API) that can be used to explicitly specify multi-threaded shared-memory parallelism in C, C++, and Fortran programs. The OpenMP API is composed of three components:

- Compiler directives,
- Runtime library routines, and
- Environment variables

The Sun Studio application development suite is a comprehensive, integrated set of compilers and tools for the development and deployment of applications on Sun platforms. The Sun Studio compilers support the OpenMP Specification Version 2.5. In addition, the Sun Studio tools support writing, debugging, and analyzing the performance of OpenMP applications"[18]. Thus, Sun's inclusion of OpenMP support in the Sun Studio compilers available in the OpenSPARC community is specifically intended to reduce the effort required by the developer community in the creation of parallel software development for the OpenSolaris operating system. In addition to OpenMP, Sun supplies the following tools that enable easier parallel programming development as part of the Cool Tools offering in the OpenSPARC community:

- Sun Studio 12 Compilers and Tools (development environment for the Solaris operating system)
- GCC for SPARC® Systems (C/C++ compiler)
- BIT (Binary Improvement Tool (BIT) works directly with SPARC binaries to instrument, optimize, and analyze them for performance or code coverage.)
- Faban (benchmark development and management knowledge)
- SPOT (Simple Performance Optimization Tool to measure the performance of an application.
- Sun Application Porting Assistant (a static source code analysis and code scanning tool that identifies incompatible APIs between Linux and Solaris platform and helps to simplify and speed up migration project estimation and engineering.)
- Solaris Grid Compiler (a locally or remotely hosted compilation resource management system for the Solaris Operating System.)

Thus, the purpose of the investment Sun has made in offering the Cool Tools toolkit as part of the OpenSPARC community is to reduce the required efforts of the developers to write parallel applications for the Solaris operating system. Sun believes that providing tools that reduce the difficultly in parallel software application development will encourage developers to write more parallelized applications.

3.5 SPARC Chip Architecture is an IEEE Open Standard

Sun's SPARC (from Scalable Processor Architecture) is a RISC microprocessor instruction set architecture. In 1989, Sun created SPARC International, a non-profit organization in charge of promoting the SPARC architecture, overseeing and guiding the SPARC evolution, and providing conformance testing. There have been three major revisions of the SPARC architecture released by SPARC International: SPARC V8; IEEE Std. 1754; and SPARC V9.

Sun has always believed that fostering competition using SPARC processors would benefit Sun's business. In an article written in 1998, Stephen Shankland states that "On the one hand, there is a limited market for SPARC chips, and a cloned chip can eat into Sun's market share. On the other, "Having that competitor could be a good thing for Sun," Diefendorff said, because "it appears to purchasers that there's more mass or weight or industry interest in SPARC."

"Having more muscle behind Sun's Unix platform could help the company get more applications written for the SPARC architecture and therefore be in a stronger position fighting off Microsoft's operating system threat", he said.[13]

Since the introduction of SPARC International, there has been several OEM's who have licensed and produced SPARC processors. Fujitsu is the most notable OEM that has adopted SPARC architecture for its server products. Since 1995 Fujitsu has designed SPARC V9-compliant processors under the SPARC64 brand.

Although the instruction set architecture for the UltraSPARC T1 processor was already available to the market as an open standard, the architecture specifications are not sufficient enough to be able to implement a multi-threaded processor design. According to David Weaver at Sun Microsystems, 2008, Sun open sourced the detailed processor implementation in order "to enable students at universities to take an existing multi-threaded design and understand it *by dissection*, learn by modifying it, and turbo-boost their research by giving them a "base" platform on which they could build and test their

research ideas. It's particularly useful that they can run an existing CMT processor design on an FPGA, including a full software stack (hypervisor, operating system, applications). Otherwise, it's a huge amount of work for them to re-invent the wheel (design a full processor, write or modify an operating system, etc) just to have a *starting* point for their research. Sun Microsystems even teamed up with two outside companies, Xilinx and Digilent, to provide an FPGA board and a canned OpenSPARC derivative design, ready to be immediately downloaded to the board, boot Solaris or Linux, and execute applications! Sun also wanted to make it possible for commercial companies to license a complete design, modify that design and produce interesting derivative processors for specialized applications (networking, crypto, HPC, you-name-it)."

3.6 The Power of Open Source Communities

"Free as in Freedom, not as in Beer" – Richard Stallman

In the 1980's the phrase "free software" was replaced by the term "open source" in order to eliminate the misinterpretation that free software only represented software which was free of cost. Instead, the phrase "free software" was intended to signify the freedom of information about its source code to allow a customer to dissect, alter, manipulate, and repackage the code according to the individual customer's needs.

According to the Open Source Initiative, we can define "free" or "open source" software as such if it complies with the following requirements:

1. Any other party is allowed to give away or sell the software as a part of an aggregate software distribution without having to provide royalties to the open source community.
2. The source code must be zero cost to download over the internet.
3. Modifications are allowed to be distributed under the same terms as the original software license. Modifications may be required to carry a different name or version number of the original.

4. Any person or group is allowed to utilize the program or source code, for any use or purpose.
5. No additional license is required by anyone using a redistributed version of the software.
6. The program is not restricted to a particular software distribution or redistribution. The program can be extracted from a particular distribution and the same rights are available as those granted with the original software distribution.
7. The open source program can be distributed along with licensed software, and does not require that the licensed software comply with the open source definition.[28]

Open source software companies have developed a mutually beneficial relationship with the open source communities that develop their software. Open source adopters who are concerned with service, support, uptime and timely resolution of product bugs are often willing to pay for support packages for the open source software. The revenue generated from these paying customers to the open source company is used to fund engineering resources that in turn contribute back to the feature sets and bug-fixes of the open source code. In addition, the open source developers contribute their own enhancements and bug fixes which drive up customer adoption. The improved software serves to increase even more customer adoption, thus perpetuating the cycle. The open source company benefits from this model by obtaining additional revenue from the increased customer adoption, and the community benefits by receiving enhanced software that they put to their own use.

3.6.1 The Bee Keeper Analogy

A popular analogy that clearly explains the success of the open source participation model is 'The Beekeeper Analogy'[3]. The Beekeeper analogy draws a very close parallel between the elements of the bee farm ecosystem, i.e. the bees living in and producing for bee farm, the role of the beekeeper in growing and profiting from the bee farm and the role of the customers who buy only the packaged wax and bottled honey from the farm;

and the elements of an Open Source community, i.e. the developers, the Professional Open Source Software sponsoring company, and the customers who purchase the final packaged software solution from the Open Source community. An illustration of the parallels between the Beekeeper and the Open source community can be seen below:

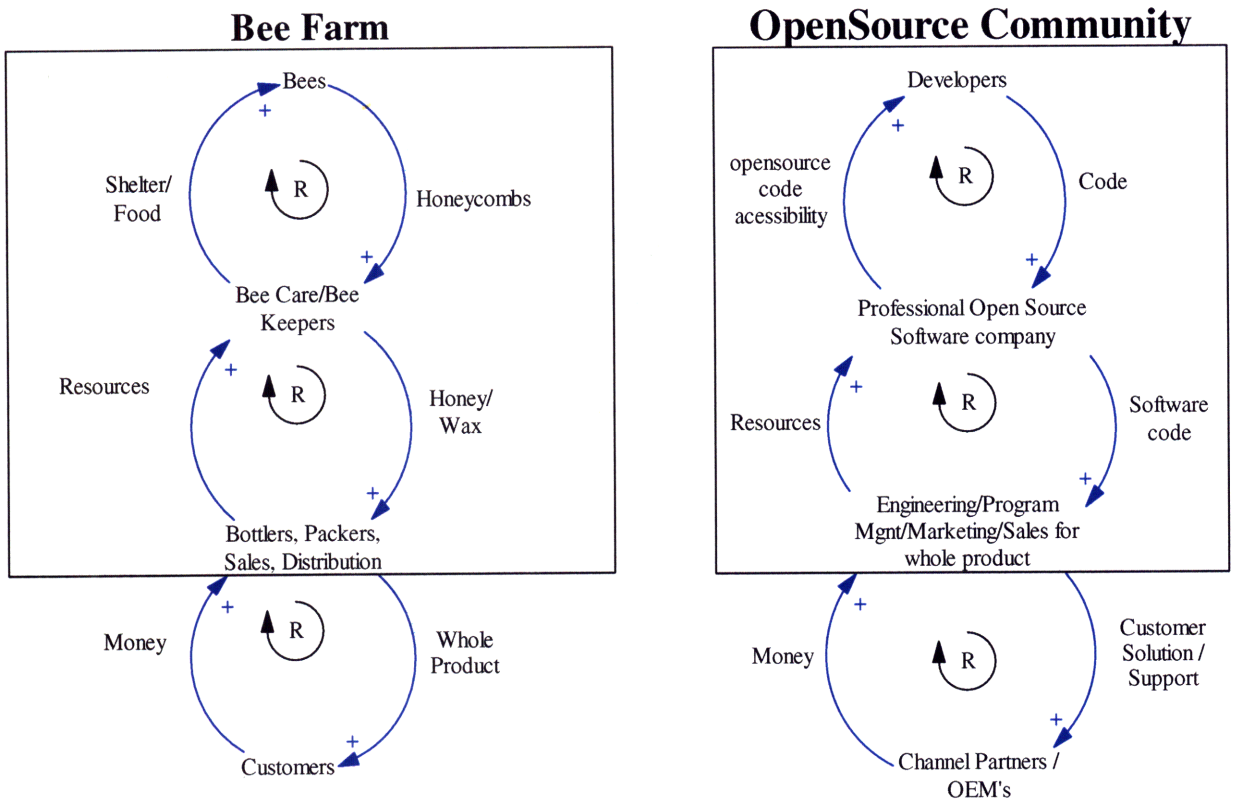


Figure 15: Bee Keeper Analogy to Open Source Communities

The important points to learn from Bee Keeper analogy on how to build a successful open source community are:

1) In the beekeeper model, the Bees must be kept happy in order for them to stay on the farm. They can leave the bee farm at any time since they are not held captive. The bees stay within the farm and keep producing honeycombs because they are getting as much benefit in the form of food and a hive as they are providing to farm in the form of

honeycombs. Like the bee's in the bee farm, the Open Source developers are not held captive to the Open source community, and can chose to depart the community at their own will. They continue to participate and contribute to the community when they are receiving the mutual benefit of receiving access to the open source code, and other tools and support that the Professional Open Source company is providing to the community.

2) In the beekeeper model, the bee keeper must create an environment that attracts and maintains his bees. A beekeeper must find a balance between taking too much honey and wax for him verse how much he leaves in the bees' hive. If he gets greedy and tries to force them too hard, they will flee the hive and he'll be left with no honey or wax. Like the beekeeper, the Professional Open Source Software (POSS) company, or project founder, must carefully chose what he takes from the community to sell as the revenue product and what he leaves within the community. The POSS Company must be a genuine supporter of the open source effort, and not push their requirements and force the community to participate solely on their behalf. This behavior would not be appreciated by the community and would result in loss of membership. A successful POSS and Open Source community results when the natural output of the community is a mutual benefit to the sponsoring company.

3) In the beekeeper model, the contribution from any one bee in particular is insignificant. It is the sum of the contributions of all the bees that make a significant difference in the amount of honey and wax that is produced by the farm. Like the bees, any one developer's contributions to the community are not likely to be significant. The amount of software that is developed by the community is dependent on a large quantity of developers contributing on a regular basis. This reality is one of the powerful motivations behind the open source model. Any one company can not hire enough software programmers to match the available number of developers available in the open source communities. This vast pool of developers is one of the major reasons that companies decide to release their products to the open source community.

4) In the beekeeper model, the customers are different from the bees, and you can not make a customer into a bee or vise-versa. A bee is an individual contributor that produces honey, and a customer is not interested in making honey, but in buying the whole product that the bee farm is selling. Likewise in the Open source model, the developers are individuals who contribute code to the community. While it is true that the developers can be a part of the organization that purchases the 'whole product' of the open source community, they are not the same entity.

3.6.2 The Proprietary Model

There are many differences between the Open Source and proprietary software development model, mostly based on where the resources are located that are completing the code development. We can see the key differences between these models in the illustration below:

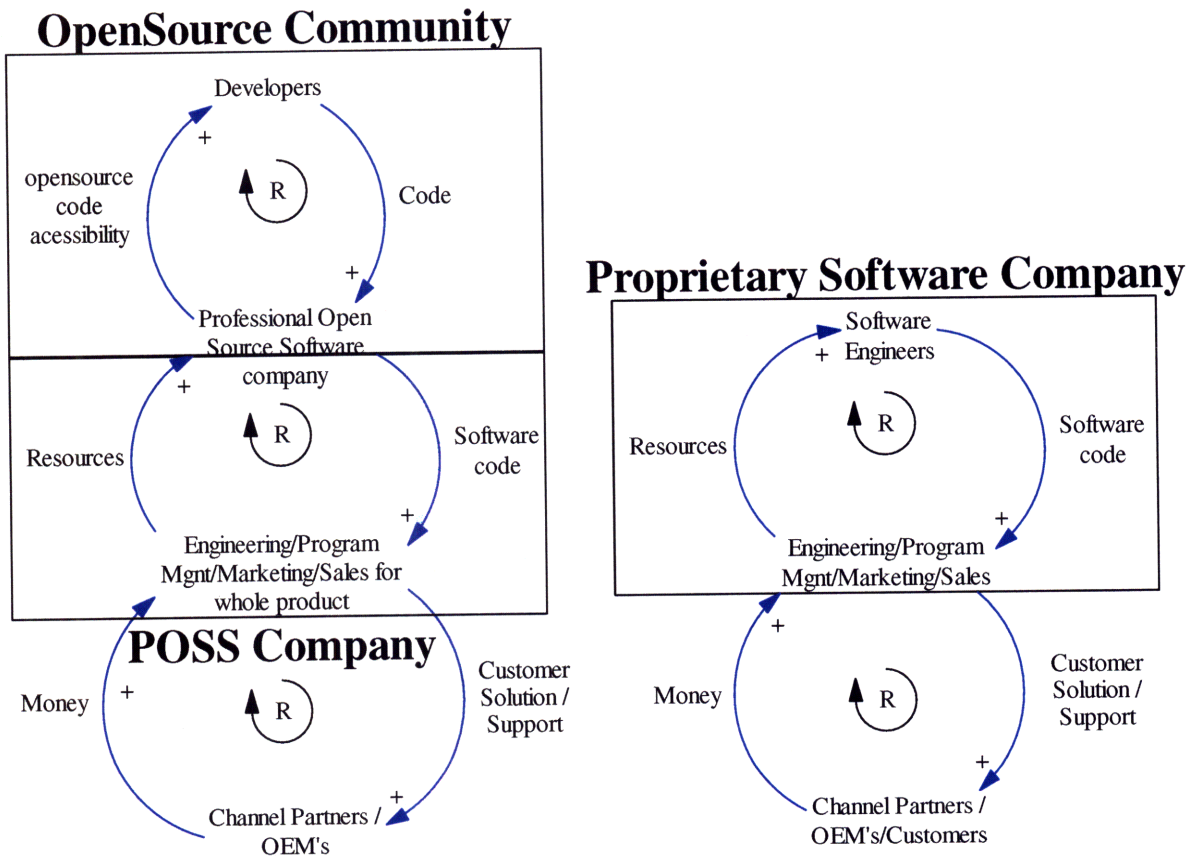


Figure 16: Comparison of Open Source Model to Proprietary Model

We can see from the comparison of the open source and proprietary software development model that the key differences include:

- In the open source model, there are two sources of code contribution, one from the internal resources of the Professional software company and the other is from the open source developers. In contrast, the proprietary company only has one source of code contribution, which is solely from its internal software development engineers.
- There is less opportunity for the customer to be involved in the development of proprietary software. In contrast, in the open source model, a portion of the developer pool may consist of software engineers at external companies which are customers of the final 'whole' product sold by the Professional Source Open Source Company. This opportunity for the customers to be involved in the development of the code base can lead to higher adoption of open source products as a result of greater value derived by the customer base.

3.6.3 Forces Driving Participation in the Open Source Developer Community

In order to understand in more detail the developers' decision to participate in an open source community, data was collected through a literature review on the dynamics that drive open source communities. The major areas of motivation discovered in this review include:

- The desire for social involvement
- Fame, recognition and appreciation
- The perceived value of contributions

3.6.3.1 Social Involvement

According to a study[20] by Yan Mao, Julita Vassileva, and Winfried Grassmann the importance of virtual communities in society are the following:

- Virtual communities help replace the relationships lost as more and more informal public spaces disappear from our real lives [25, 26].
- They allow people with similar interests to connect with each other and to get benefits from the presence and activities of other people in virtual communities.
- Virtual communities provide not just information, resources, and conversations which people can use and participate in, but they also provide a way to form social relationships that allow them to do things together with others in a new way.
- The positive emotion people experience when they belong to a virtual community becomes an intrinsic incentive for further participation in the community, which makes virtual communities self-sustained.

In summary, this study suggests that developers participate in virtual collaborative communities because they value the sense of social involvement the gain from their participation.

3.6.3.2 Fame, Recognition and Appreciation

Data collected through additional literature research shows that developers write code for the following reasons[29]:

1. Fame, recognition and appreciation that comes from contributing code
2. Believe their contributions will lead to future career opportunities
3. To feel community-social involvement
4. Because they love writing code

Because developers are motivated primarily by their own emotion and not from corporate charters and funding, we can determine that a critical driver in increasing their involvement in the open source community is their belief in the success of the community and their individual code's success in the market.

After a belief in the community's success is established with the pool of developers, a fundamental mind shift must take place in order to for the developers to consider writing

a parallel program verse a traditional serial-based software program. According to Craig Bailley in 2008[30], "Parallel computing is a significant mind shift for programmers. It's not taught (much) at universities, and it certainly isn't marketed as a 'sexy' side to development. The toolset to date has been almost non-existent, and it's no surprise there are hardly any applications written with parallelism in mind." This mind shift will be driven by the belief that parallelism is the future, and that there will be the fame, recognition and appreciation from the community that comes as a result of contributing parallel code.

3.6.3.3 Perceived Value of Contributions

In order to understand why the developers contribute to community, V. Diker published a system dynamics model looking at open source software communities[21]. Diker modeled the changes in developer population, contributions to the community, software product functionality, and motivations the developers have in joining and leaving the community. Diker's complete model was organized in seven sub-sections, and it involved more than 270 variables. Due to the large size of Diker's model, only a portion of Diker's open source system dynamics model is shown in the figure below:

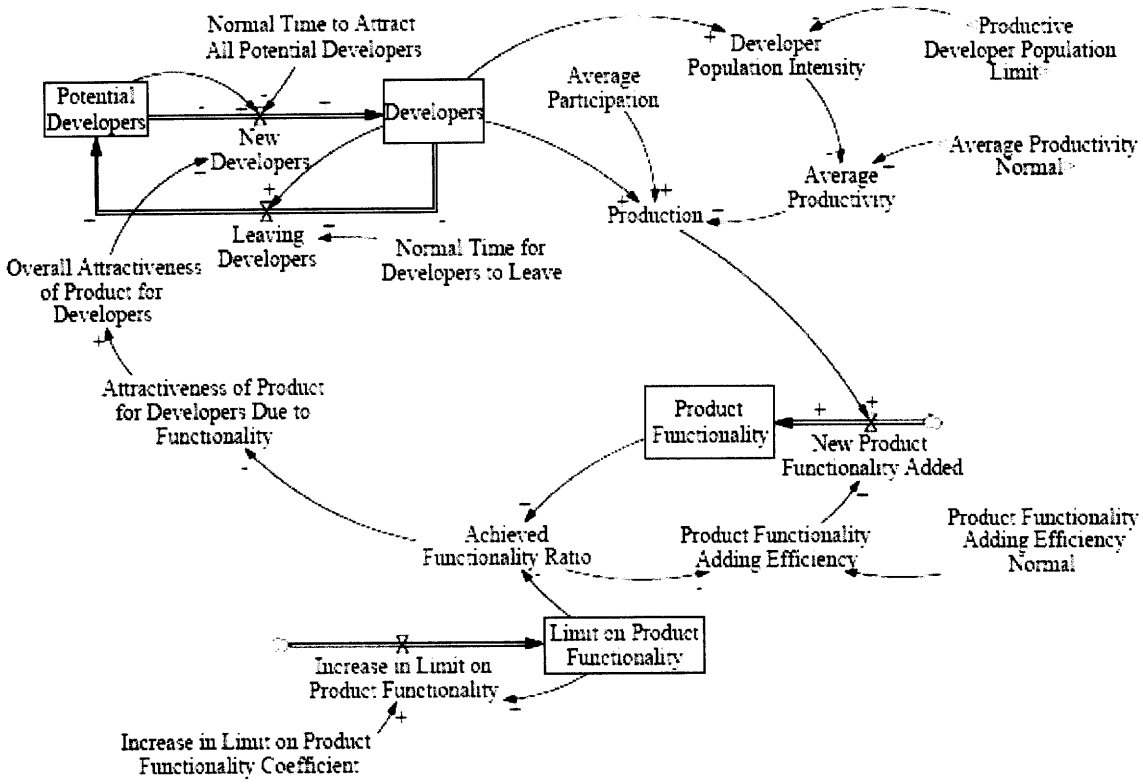


Figure 17: Developer participation in Open Source Community (Diker 2003)

The model section above demonstrates the developer population, code production, product functionality and developer motivations to join and leave the community. This model demonstrates how developer participation or "production" leads to new product functionality added to the software application. Diker's model demonstrates that as the software functionality reaches its upper limit, the developers are less attracted to contributing code to the community. According to Diker, "this relationship is based on Raymond's (2001) arguments that a developer joins an open source software project in order to "homestead" a certain portion of the software. By developing the "homesteaded" portion of the software the developer builds reputation and self-efficacy sentiments. When the achieved functionality approaches the functionality limit, the project does not offer many portions to homestead, and this decreases the attractiveness of the project. This also accelerates the rate of 'leaving developers'. As the project approaches the end, more developers leave the project, leaving a smaller number of developers for maintenance purposes."[21]

In order to validate these claims, we can look once again to Diker's study on the dynamics that exist within the open source collaboration community that govern the growth of the community. Diker has discovered a strong correlation showing how the size and the functionality of the collection influence the number of new users. Diker has found that "a larger and more functional (more useful) collection brings more new users, thus forming the main positive (reinforcing) loop."

In Diker's growth loop below, he illustrates the dynamics that exist when authors produce content that adds quality problems to the overall collection. He shows that the experienced authors review the collection, throwing away material that is incorrect and selecting some material for rework. Diker's growth model is centered on the dynamics that if too much material is discarded or sent for rework, it will decrease new authors motivation to contribute to the collection, thus decreasing the growth of the open source community. In the contrary, if new authors are able to contribute content that is accepted and utilized by the community with little rework, this will increase the ratio of users who become new authors. The growth dynamics identified by Diker can be applied to the open source communities for parallel software development to determine how multithreaded system companies can encourage more developers to contribute content to these communities.

In order for multithreaded systems companies such as Sun, IBM, Dell, etc that rely on 3rd party software developer communities to foster growth in these communities, they must help the communities to reduce the difficulty in writing parallel software code, as well as provide education and advertisement to the communities about how much new functionality is needed within the community and how successful their contributions are going to be in the market. Many companies may not appreciate how strongly the perception of need and promise of success correlates to the level of contribution within a community, and may ignore these steps under the assumption that they are "soft" requirements in order to foster a successful online community. However, data collected by Diker in his study reveals the strong dynamics between the emotions and perception of

the developers and the growth of the online community. As further emphasized in Diker's open online collaboration growth model below, the phenomenon of increased authors, or contributors, results from lower rejection rates of new code to the community, which leads to increased contributions by new authors, thus accelerating the growth of the community.

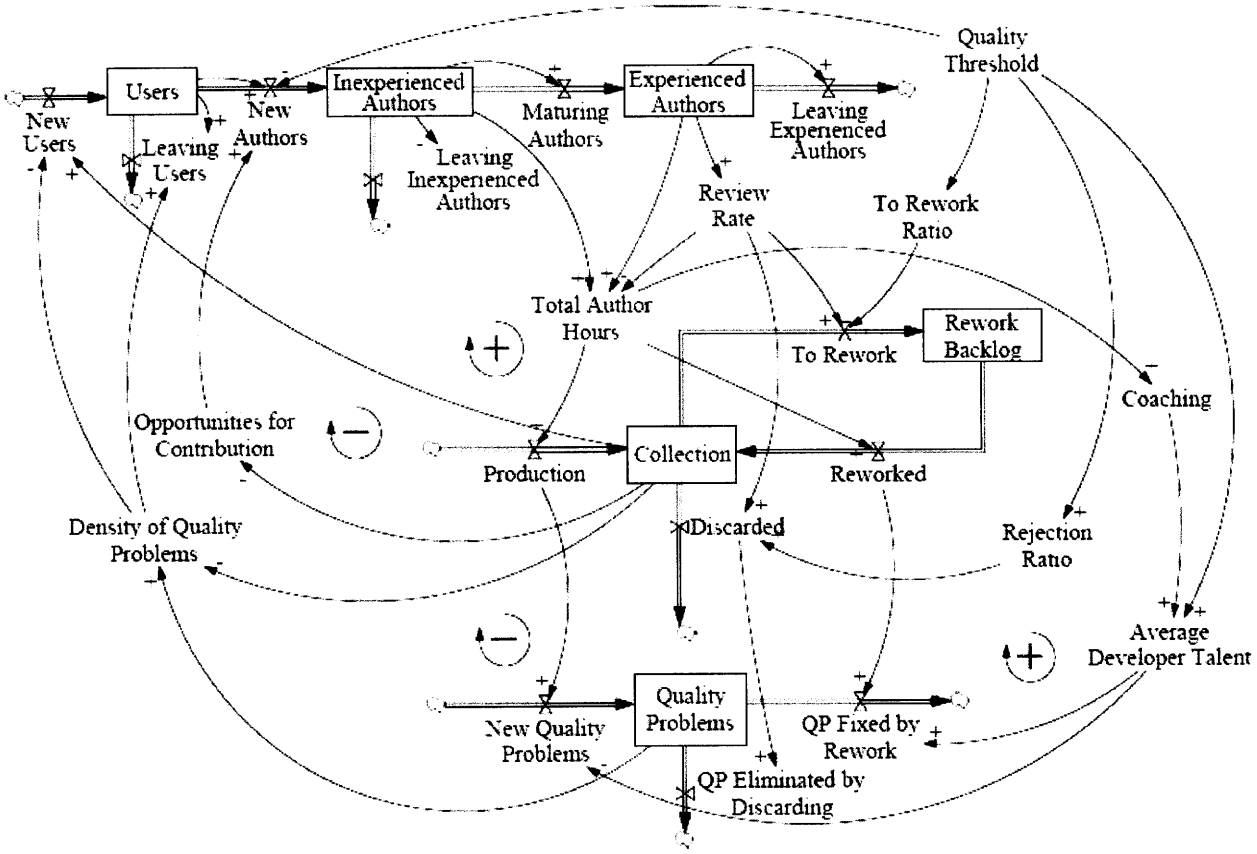


Figure 6. The Dynamic Feedback Framework for OOCCs.

Figure 18: Diker's model for growth in an open online collaboration community

3.6.3.4 Forces Driving Participation in OpenSPARC Community

We can take the lessons learned from the literature review of developer participation and growth in open online collaboration communities, and apply these theories to a simplified view of the forces driving participation in the OpenSPARC community.

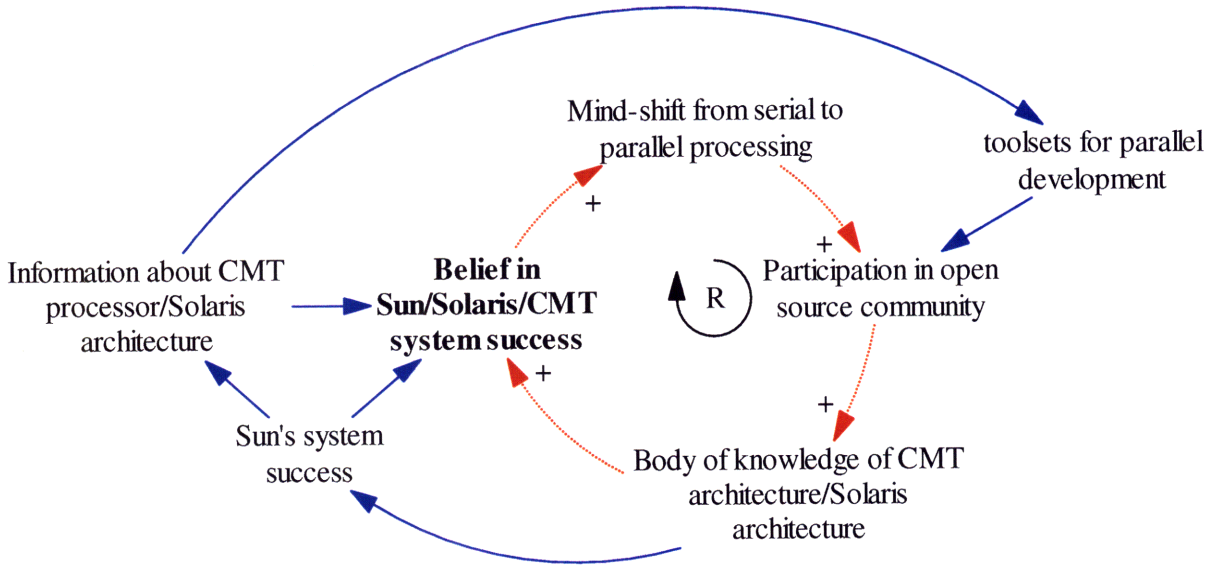


Figure 19: Forces driving participation in OpenSPARC Community

This simplified loop describes the forces that drive community participation in open source development. The first step of the reinforcing loop is the critical parameter of the belief in the success of Sun's CMT/Solaris systems in the industry. Developers are not likely to opt to write code for the Solaris/CMT system if they do not believe these systems will be successful the market. Developers have human emotion that drives them to want to see their code being used in successful software applications by many people; this is obviously a pre-requisite for fame and recognition as well, and their belief about the community will also be reflected in their belief about the opportunity for social participation.

The next step in the loop describes the participation of the developers in the open source community. Once the developers believe in the success and have shifted their thinking to developing parallel software, they will start to contribute code to the open source community. Their contributions are based on their skills and capability to produce the necessary software. These skills and capabilities come from the information that is published and available on CMT processors and Solaris architecture as well as the tool kits that are available to them. The easier this knowledge is for them to gain, the more

ease the developers should have in writing and contributing code to the community. This step relates back to the assumption Sun has made that knowledge about the hardware implementation of the processor will improve the ease of writing parallel software applications either directly or indirectly. Thus, Sun's desire to provide education and development support is one of the primary motivations behind the OpenSPARC community, which provides details and actual hardware models of the chips as well as open source software developer tools for parallel code development. Sun is working to eliminate any hardships the developers may face in their quest to contribute code, once they have developed the motivation to do so.

Lastly, the final step in the reinforcing loop is the body of knowledge that gets created as more and more developers join the community. As participation is increased, the developers learn from each other through the repositories of information that is created within the community. The tools the OpenSPARC community uses to distribute knowledge includes collaboration websites, email mailing lists for distribution of cognition, and wiki postings for problem identification and resolution, bug reports, source code trees for revision management, feature requests and news forums that is essential to the growth of the open source software development community serves as the growing body of knowledge that new developers can draw from. If this body of knowledge is effective in capturing the issues and driving effective resolution within the communities, it can serve as a mechanism to drive the developer's belief in the success of the community, which closes the reinforcing loop and perpetuates the growth cycle.

3.7 Why Sun is Uniquely Positioned for Open Source

Sun is uniquely positioned to benefit from open sourcing the main system components (CPU and Operating System) since it is positioned in the market as a true systems company, as opposed to a chip or software only company. In other words, Sun does not get revenue solely from CPU sales or software sales, but the sales of its entire systems. As Jonathan Schwartz, CEO Sun Microsystems, Inc. stated:

"[Sun is] predominantly a systems company — 75%-80% of all the revenue at Sun comes as a result of selling systems. And systems are the combination of silicon innovation, hardware engineering and packaging, as well as software design..." Jonathan Schwartz, CEO Sun Microsystems, 2008

The table below shows a comparison between Sun's product offerings to the marketplace and its major competitors, IBM, HP and Dell:

	Sun	IBM	HP	Dell
Systems	Workstations, Servers, Blade Servers, Storage, Software, Services, Solutions	Servers, Blade Servers, Storage, Workstations, Software, Services	Servers, Blade Servers, Storage, Solutions, Laptops, Desktops, Workstations,	Laptops, Desktops, Workstations, Servers, Rack Servers Storage, Printers, Switches, Services
Operating Systems	• Microsoft Windows • Linux **(Open Sourced)** • OpenSolaris **(Open Sourced)** • Solaris	• Windows • Linux **(Open Sourced)** • IBM i5/OS™ • AIX 5L™	• HP-UX • Linux **(Open Sourced)** • Microsoft Windows • OpenVMS	• Linux **(Open Sourced)** • Windows • OpenSolaris **(Open Sourced)**
CPU	• Intel x86 • AMD x86 • UltraSPARC **(Open Sourced)**	• Intel x86 • Intel IA-64 • AMD x86 • POWER	• Intel x86 • Intel IA-64 • AMD x86	• Intel x86 • AMD x86

Figure 20: Comparison of Sun's and Competitor's Product Offerings

We can see from Sun's product offering in the table above, that it sells systems with open source CPUs (UltraSPARC) and open source software (OpenSolaris), both of which benefit from having open source communities sponsored by Sun. In contrast, Sun's competitors offer systems only with open source software, specifically Linux, for which they neither sponsor nor control the open source community. Sun is able to sponsor the open source communities for its CPU and Operating system components since it owns their intellectual property. As we can see from the table, most of Sun's direct competitors capture market share using different approaches other than owning the IP of the full system, consisting of software, system design and processor. Sun's competitors still

generate revenue from their system sales, but chose to rely on 3rd party suppliers such as Intel, AMD and Microsoft to provide the critical system components such as the Operating Systems or the processor, the CPUs. The value-added differentiators that IBM, Dell and HP use to drive adoption of their systems include value-added services, logistics models, customer loyalty and brand recognition. Sun's specific position in the market will be explored in more detail in the results section.

3.7.1 Summary of Company Comparisons

"[Sun's] invention and innovation start from the silicon and rise up through the system and then move through Solaris and OpenSolaris up through the application tier. Companies that are trying to apply that after the fact on somebody else's microprocessor or somebody else's operating system end up, from where I sit, being less efficient." Jonathan Schwartz, CEO Sun Microsystems, Inc

According to Schwartz, Sun believes that it has a key competitive advantage in improving the performance of its systems since it contributes IP to the design of all of the major system components such as the CPU and the operating system. This is a critical insight; it's this belief that gives Sun Microsystems confidence that open sourcing its hardware design implementation through the OpenSPARC community to software developers will enable the development of software that works better on Sun's CMT hardware then on its competitors' hardware.

Sun believes it can achieve a better overall system performance than its competitors who rely on other 3rd party companies to achieve similar interoperability between the software and hardware components.

The core revenue stream of Sun's business, hardware systems sales, is positioned to benefit from the innovation coming from their partnership with the open source communities. Sun's competitors do not have similar access to this innovation channel. While HP, IBM and Dell offer some systems with open source operating systems pre-installed, they still rely on their partnerships with Intel, AMD and Microsoft to provide

them the innovation needed to differentiate their systems. Their business model is one which Jonathan Schwartz, CEO of Sun Microsystems, references as being less efficient since they have to integrate 3rd party operating systems and 3rd party software into one system where all components are required to work together seamlessly. Sun believes their competitive advantage is their ability to be a true systems company that controls the IP throughout all major components of the system and can optimize system performance starting from the silicon up to the software applications.

3.8 Risks and Benefits of the Open Source Model

3.8.1 Benefits of the Open Source Model

There are many benefits to using the open source software development model to generate open source code, beyond the economic benefit of using "free" developer resources. There is evidence in the software industry that open source software has the following benefits over proprietary closed-source software:

1. BETTER PRODUCT: Code is contributed from other experts and developers in the industry, not just the experts within the proprietary company. This external code contribution results in a more robust product, with higher security and faster bug resolution.
2. LOWER SALES COST: Sales and Marketing fees are less for open source companies than for traditional proprietary software companies. Because open source software has a large contributor base who may become customers in the future, the OSS companies do not need to spend as much on marketing their software[15]. According to John Roberts, chief executive of SugarCRM, "We allow organizations to download the software to do their own tests -- they can look at the source code, prior to getting into a dialogue with a sales person. We don't have much of a sales force," Roberts said.

3. WIDER MARKET: The open source community is capable of supporting deployments in more countries then a proprietary company is due to the localization and language support typically contributed by the community.

3.8.2 Risks of the Open Source Model

The benefits of the partnership with Open Source communities also come with some risk. The list below identifies the risks that companies take on by choosing to open source their intellectual property:

1. REVENUE RESTRICTIONS: Companies who package open source software as their business model must in most cases rely on enhancements or product complements in order to generate revenue from the open source software, depending on the license agreements.
2. COMPETITIVE INFORMATION EXPOSED: Competitors also have access to the innovations happening in the open source community. For example, Intel and AMD have access to innovations happening in the OpenSPARC community, which may to some extent strengthen their own multi-core CPUs architectures. The extent to which competitor's CPU architectures' are strengthened depends on the degree of similarity and co-specialization of the competitive information. While at first glance this competitive intelligence might appear to be a threat, we can see that Sun also benefits from Intel's and AMD's product improvements because, like its competitors, Sun also sells systems based on Intel and AMD processors. Lastly, the competitor's ability to exploit the innovations happening in the open source communities also depends on its access to complementary assets.
3. CODE LEAKAGE: Losing revenue on propriety software due to code leakage. Since companies often lend their internal development teams to contribute to both proprietary and open source development projects, sometimes the developers do not remove the proprietary code blocks before contributing their work back to the open source community. This leakage allows access of the proprietary code out to the community which can then be re-used in competitor's products.

4. RISK OF PATENT INFRINGMENT: Code that is contributed and included in an open source distribution sometimes violates proprietary patents, due to the large number of individuals contributing to the code base. This violation often leads to stop ship requirements placed on products until the violation is resolved. This infringement is usually inadvertent, and can be effectively mitigated or avoided through code reviews.

Traditionally, companies have not open sourced their software for fear of losing the right to their intellectual property. For many proprietary software companies, their business revenue model depends on the sales of licenses to their software. However, with the advent of open source development communities and professional open source software, many companies have restructured their business models to gain revenue on the sale of services and support of their distribution of open source software, rather then the sale of the license for the content of the software itself. The details around this restructuring to exploit open source communities will be explored in more detail in subsequent sections of this paper.

In summary, many software and system companies have discovered that the benefits that result from releasing their software to the open source communities greatly outweigh the risk of loss of intellectual property. These companies must be specifically positioned to either generate revenue from services, support, or other complements of the open sourced software; or systems companies that open source their software but generate revenue on the sales of their systems and services. Sun is an example of such a company that is positioned to benefit greatly from the efforts of open source communities. According to Ian Murdock, Sun's Vice President of Developer and Community marketing, "The money comes from services and support. We understand the most important thing we can be doing is getting our technologies into the hands of as many developers as possible - removing all barriers to adoption and all barrier to entry... you don't have to pay Sun anything. When you are scaling, when you need help when you need to establish a relationship with the vendor to grow and scale, that's when we make our money."[11]

Chapter 4: Results and Analysis

Using the data gathered through the interview process and the literature review, a system dynamics model was created to understand the relationship between Sun, the open source communities, and the CMT ecosystem.

4.1 System Dynamics Model of the CMT Ecosystem

The following system dynamics model displays the complex relationship between Sun's open source communities, namely OpenSPARC and OpenSolaris, and how they affect the overall market adoption of CMT.

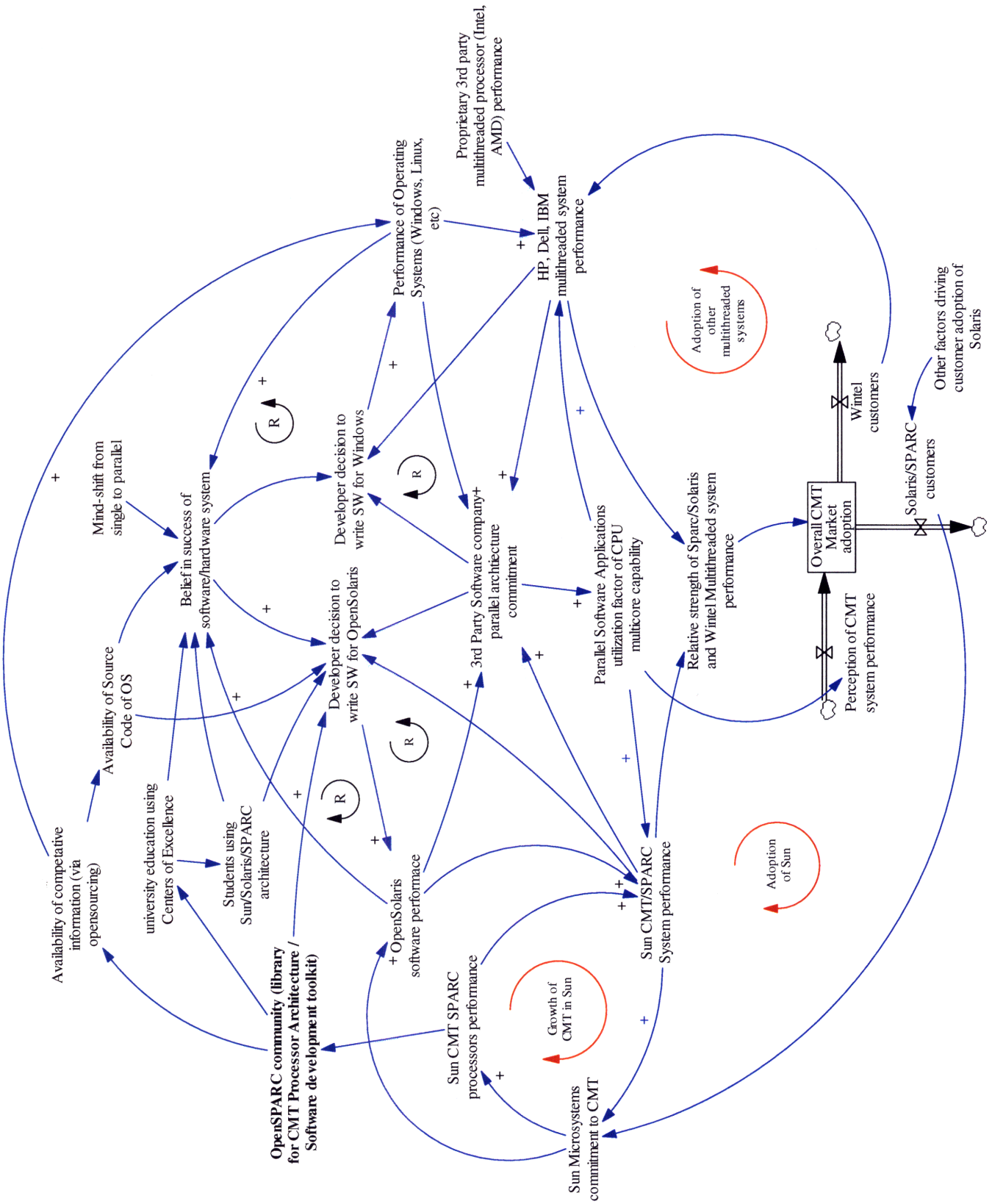


Figure 21: CMT Ecosystem with OpenSPARC

In the following sections, the system dynamics model will be broken down into smaller building blocks and analyzed to explain the causal relationships that were used to create each loop.

4.1.1 Sun's SPARC system development:

The following building block of the system dynamics model was created to understand Sun's system development strategy on the components that contribute to the performance of their multicore systems.

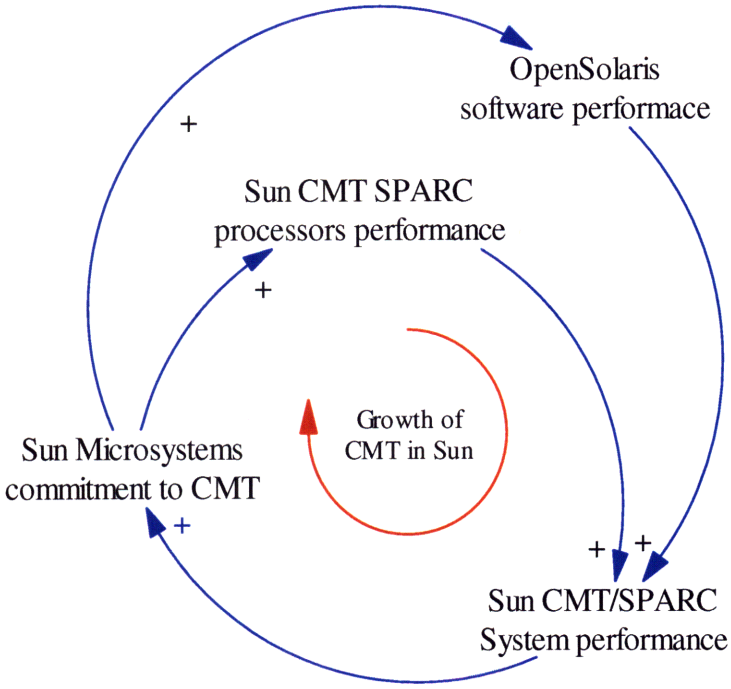


Figure 22: Loop showing Sun's system development

The key insights gained from this structure include:

- Sun is a systems company; it produces chips and software for the consumption in its servers. Sun does not sell either its chips or operating system to the market as individual components
- Sun only makes money on the sale of complete systems to the market.

In contrast, Sun's major competitors use 3rd party operating systems and 3rd party processors as the major components of their system, as seen in the next model segment below:

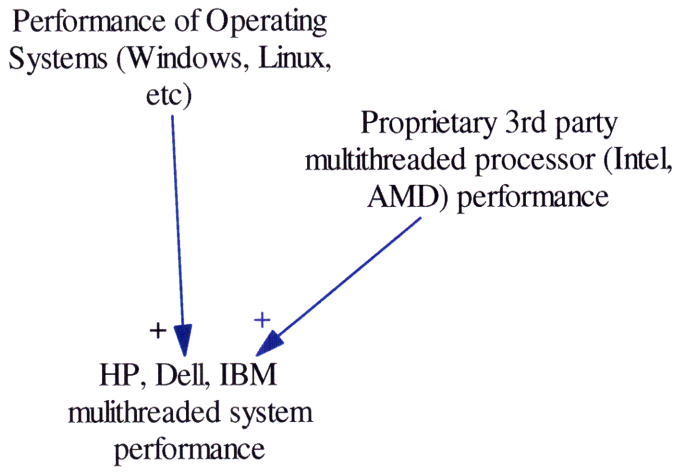


Figure 23: Loop showing Competitor's system development

Another way to represent the difference between Sun and its competitors such as IBM, HP and Dell can be seen in the block diagram below. This diagram shows a simplified view of the components that comprise the server solutions. We can see that Sun's strategy is to open source the major system components, Solaris (Operating System) and UltraSPARC (CPU), in the open source communities. Since Sun sells the whole system into the revenue market, it is able to open source the components without a loss of revenue. However in contrast, Sun's competitors' strategy is to utilize 3rd party components which they do not control. Thus, Sun's competitors do not have the same ability to open source their components. In addition, the component companies that produce the system components, for example Microsoft, which produces the operating system Windows, and Intel/AMD, that produces x86 CPUs, do not have the same motivation to open source their components. These companies make their revenue selling these individual components to the system producers, therefore releasing these products to the open source community would result in significant revenue loss to these companies.

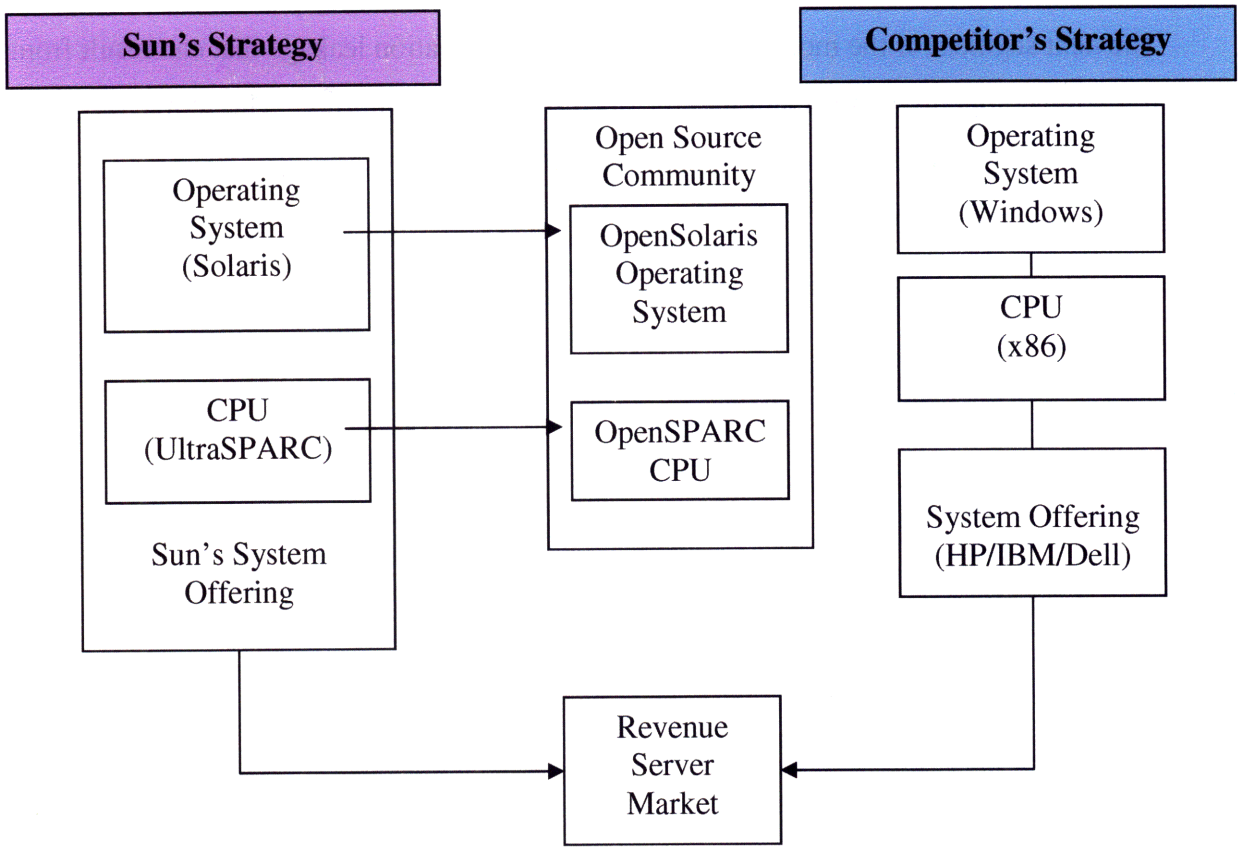


Figure 24: Block Diagram representing Sun's Strategy verse Competition

The key take-aways from this diagram include:

- Sun makes its revenue from the sale of its systems. It can afford to "give away" the components since it sells the complete system. Its competitors, however, do not control the components, and they can not afford to "give away" the components because they will lose their main revenue stream.
- Sun's competitors are not positioned to benefit from the open source innovation of the operating system and CPU since they generate their revenue solely on the sale of these components to the market.

4.1.2 Information leakage as a result of open sourcing Sun's components

The next segment of the model shows the risk of information leakage that can result from releasing competitive information to the open source communities. The source of information leakage can be seen in the dotted link below.

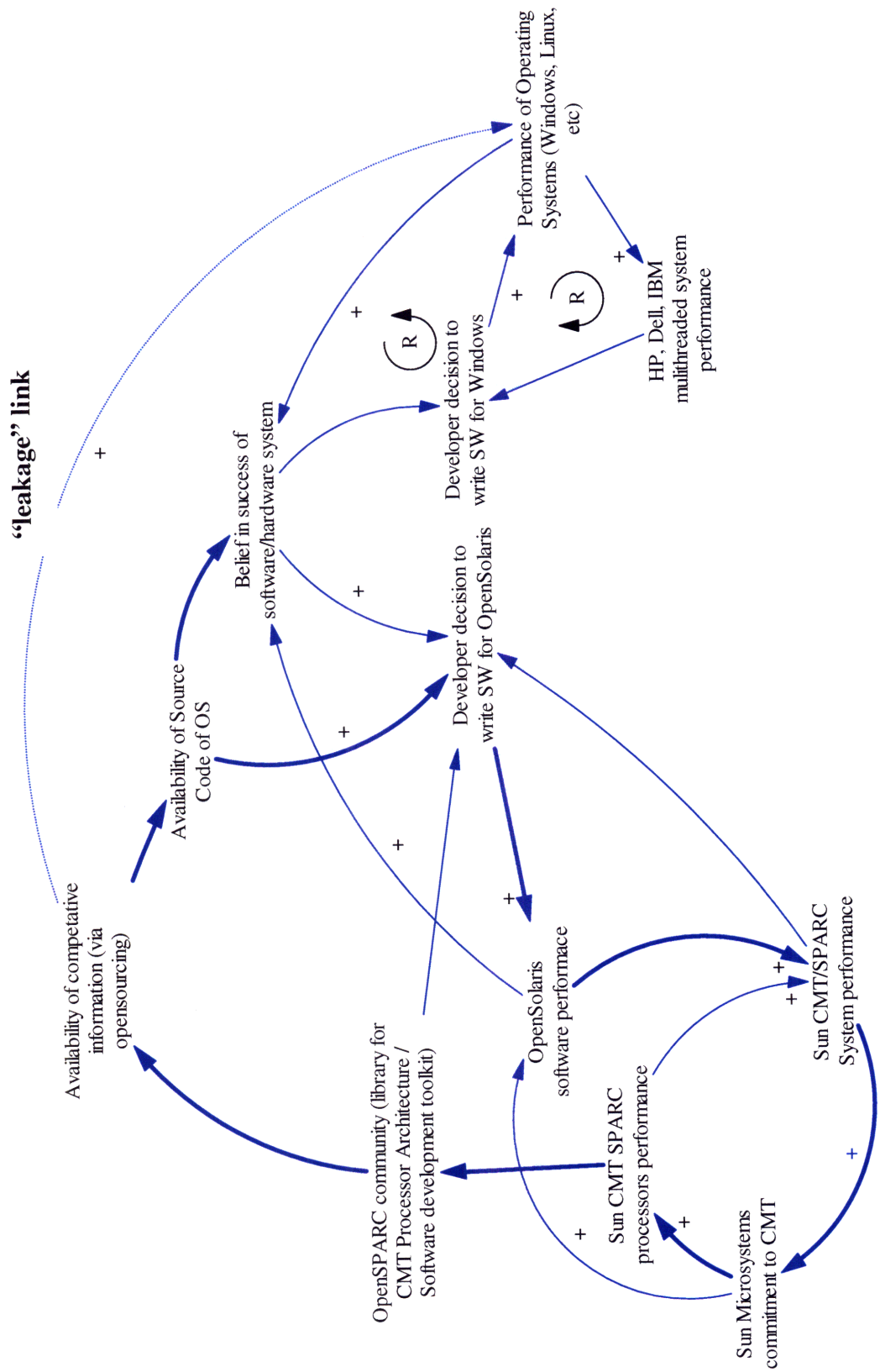
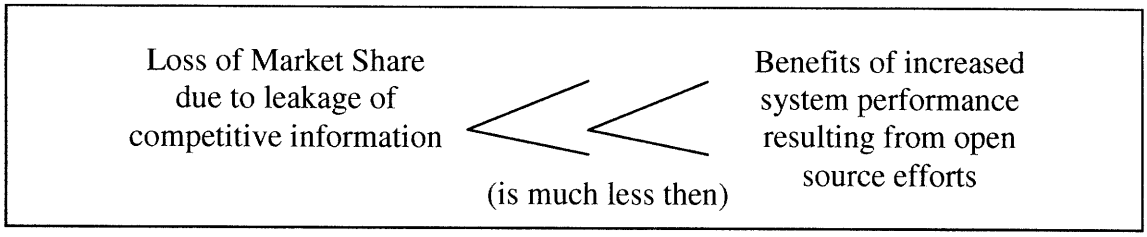


Figure 25: Diagram of Risk of Information Leakage as a result of open sourcing

The key insights from this model segment include:

- Sun's decision to open source means that it has made some "trade secrets" or competitive advantage information available for its competitors to access.
- There is an implicit belief at Sun that the positive impact of open sourcing its software code on the developers' desire to write software for OpenSolaris is much stronger than the negative impact of the resulting performance improvement on Sun's competitors' operating systems. This relative strength is depicted in the model by the different line weights between the links between Sun, the Open Source communities, the Open Solaris Software Performance, and the dashed link that depicts the information leakage to Sun's competitors.
- Following the previous point on the systems strategy of Sun, we can conclude the advantages it gains from the increase in performance that results from developers writing parallel software for OpenSolaris is greater then the losses it experiences from the increase in performance of its competition.

Another way to represent the insight gained from this model can be shown in the equation below:



4.1.3 Impact of OpenSPARC on Developers' Decision

The following loop segment illustrates the impact that OpenSPARC has on the decision of developers to write parallel software programs for the OpenSolaris operating system. The strong causal links are identified by heavy weighted lines.

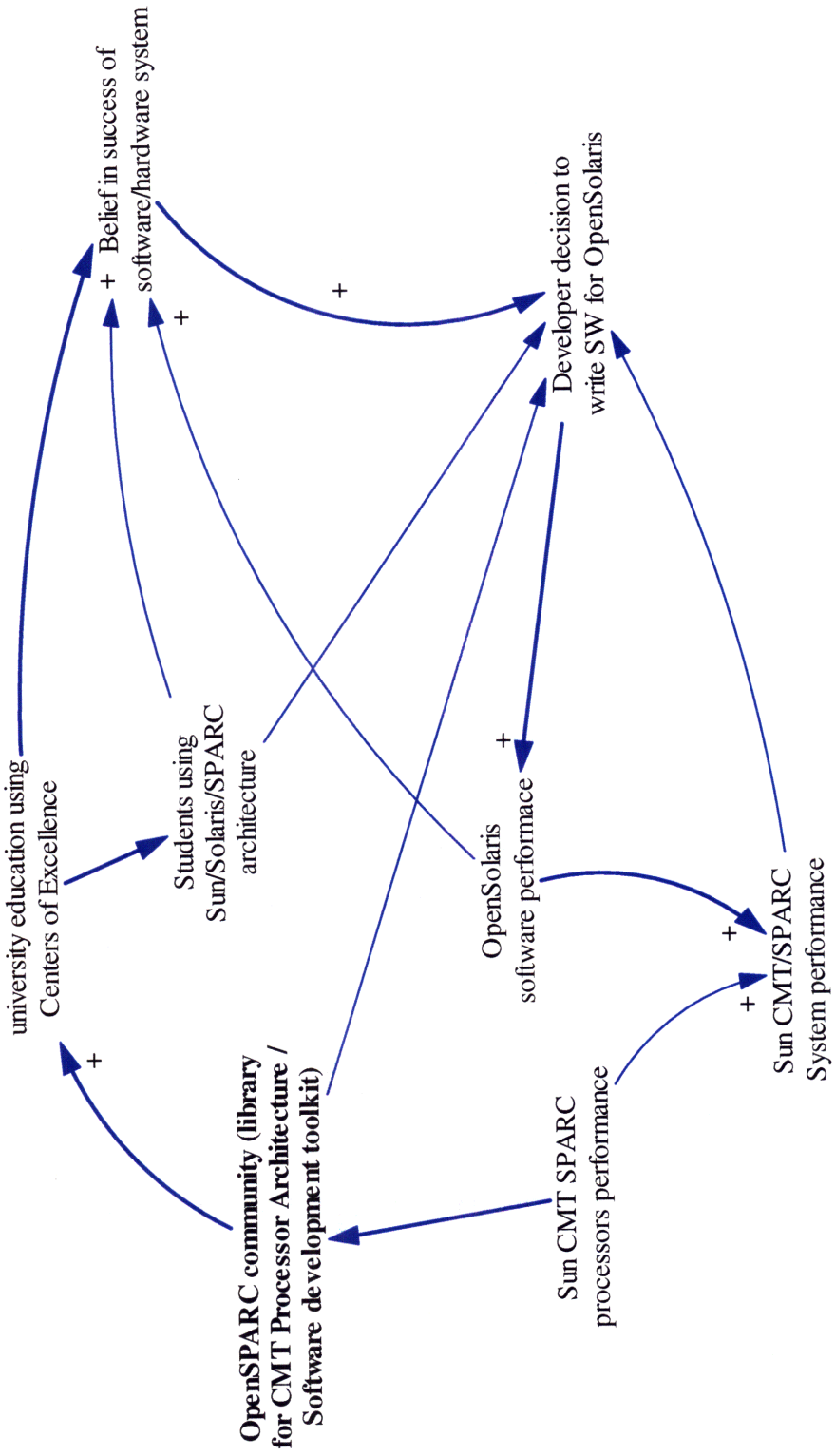


Figure 26: Diagram of Impact of OpenSPARC on Developer's decisions

The causal relationships identified by the heavy weighted lines outline the growth loop of Sun's CMT server performance. This growth starts with Sun's investment of OpenSPARC, which provides education and toolkits to universities, which in turn introduces and familiarizes students to Sun's software and system architecture, which then help drive their belief in Sun's success in the market, which drives their decision to develop software for Sun's platforms when they become part of the developer community, which improves Sun's system performance and thus leads to higher customer satisfaction and increase in Sun's market share.

The key insights from this model segment include:

- Sun is laying a foundation for the future developers of tomorrow to choose OpenSolaris/SPARC as their development platform by partnering closely with Universities to create OpenSPARC Centers of Excellence.
- OpenSPARC serves as another reinforcing loop for the open source software efforts that Sun has already started through the Open Solaris community
- OpenSPARC helps to introduce students to Sun's software and hardware early in their career, thus reinforcing the confidence that students have in Sun's success in the market.

4.3.1.1 Third Party Parallel Software Decision Tree

Another way to represent the selection process of development platforms by 3rd party software developers can be seen in the decision tree below. One of the primary purposes of OpenSolaris and OpenSPARC communities is to influence the factors that drive the developers to follow the heavy weighted decision line, and write parallel software for OpenSolaris operating system. The factors that affect the developers' key decision points (labeled A and B) are outlined following the diagram:

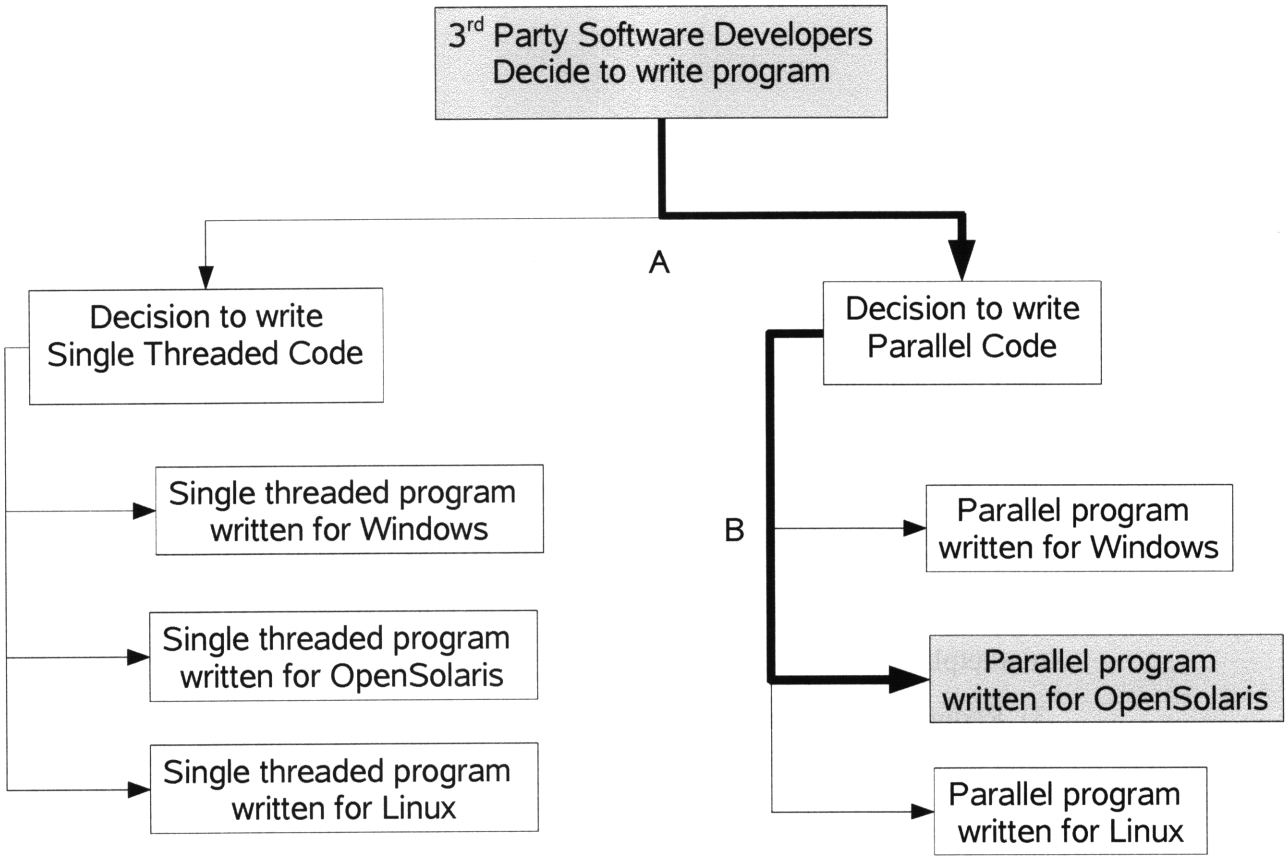


Figure 27: Simplified Decision Tree for Software Development

A) Factors that affect the decision point between Parallel and Single Threaded Software Development (decision point "A"):

1) Developer's access to the hardware needed for code development and validation. For example, multicore x86 or CMT SPARC desktop/ server is needed for parallel code validation. However for single threaded code development, a developer could use either a single core or multicore x86 or SPARC desktop/ server.

2) The developer's freedom to select the software platform he desires. This means he is independent of contracts and corporate agendas that force him to develop software for a particular platform.

3) Developer's skill to write code in parallel as well as single threaded. In other words, the relative effort required for the developer to write software in a parallel structure must be equal to or less then the effort required to write the code in single threaded format.

B) Factors that affect the decision point between Windows/Solaris/Linux development platform (decision point "B"):

1) Developer's access to hardware needed for code development and validation. For example, multicore x86 hardware is required for parallel Windows code development, and CMT SPARC hardware is required for parallel OpenSolaris code development.

2) A Developer's belief that his program will be used in the market. This factor refers back to the motivations that drive developers to contribute to open source communities, which largely stem from a belief of success of the program in the marketplace.

3) Required Operating environment needed for program. For example, if the program is going to be run largely in a desktop environment, the developer may be biased to develop his program on an x86 platform, since SPARC workstations are less prevalent in the market. However, if the program is intended to run in a server environment where both hardware types are prevalent, the developer would have to take into consideration his beliefs on the relative performance and success of the different server hardware types in the market (x86 verse Solaris/SPARC).

4) Size of install base for the operating system type (ie. network effects). This factor helps drive the developers' belief of the success of their program in the market. A basic assumption here is the larger the install base of a particular hardware will lead to a larger success of the program since there are more customers able to adopt the program on their systems.

4.1.4 Effect of 3rd party software on System Performance:

The following segment of the model depicts the effect that the 3rd party software companies have on the performance utilization factor of the multicore systems.

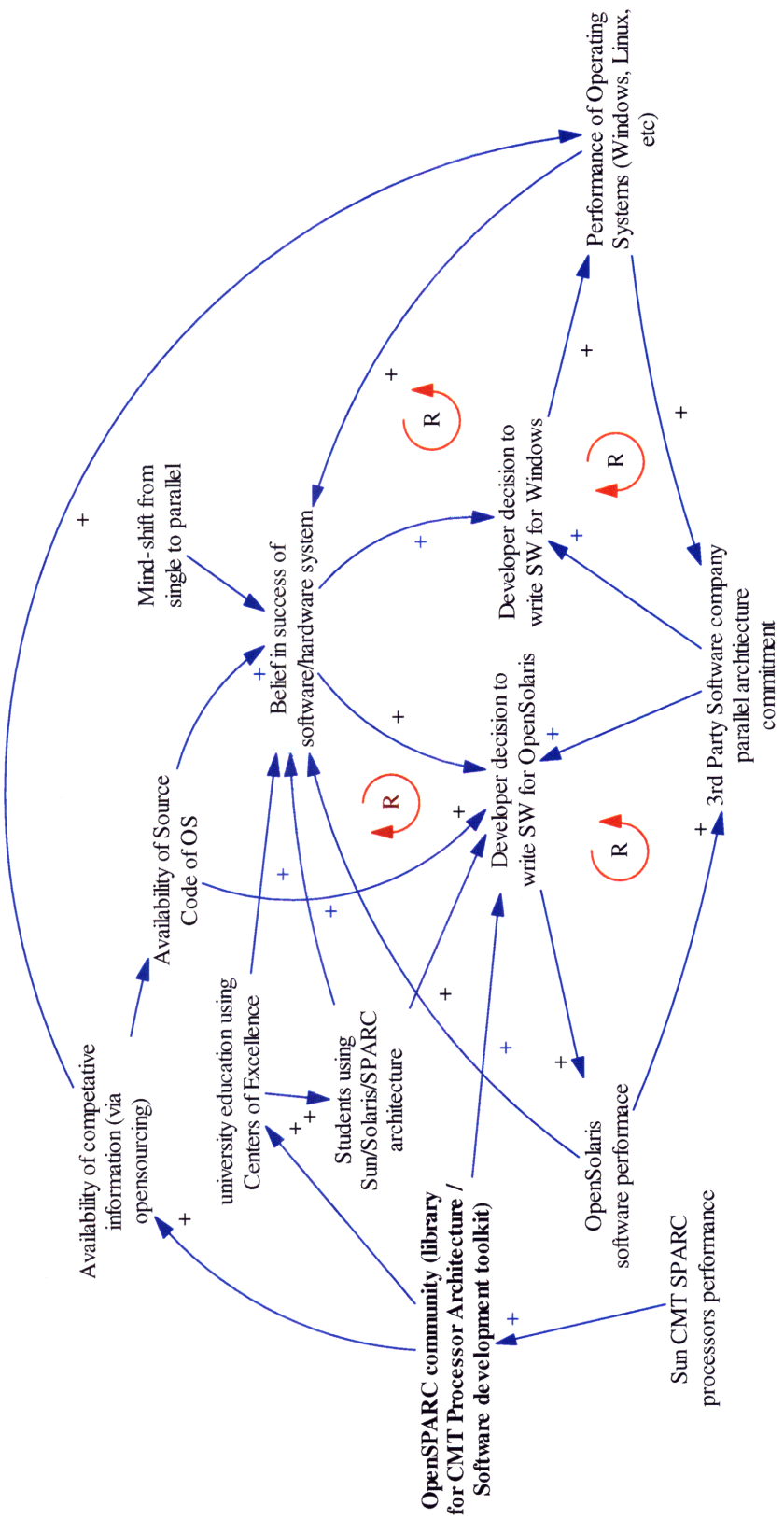


Figure 28: Diagram of effect of 3rd party software developers on OS performance

The key insights from this model segment include:

- There is a reinforcing loop between the belief of the success in the hardware/software system, the developers' desire to write software for this system, and the resulting performance of that system. As the performance increases, the belief of success increases, and thus the decisions to write software for that platform increases. This loop illustrates the critical importance of demonstrating system performance in order to attract more developers. There is a key feedback loop that illustrates the power of the developer communities involvement:

  As improved system performance is demonstrated, more developers' are likely to be attracted to writing code for that platform, which in turn drives the system performance higher. This reinforcing loop can have a powerful impact on the success of a system in the marketplace.

- There is a second reinforcing loop between the performance of the operating system and the 3rd party software company's commitment to parallel architecture. These companies are influenced by the relative performance between the OpenSolaris operating system and the performance of its competitors. As 3rd party software companies gain commitment to parallel architecture, they in turn effect the decision of developers to write parallel software programs for either OpenSolaris or Windows.

4.1.5 Performance effect on CMT market adoption:

The next segment of the model depicts the effect of performance on the market selection and overall market adoption:

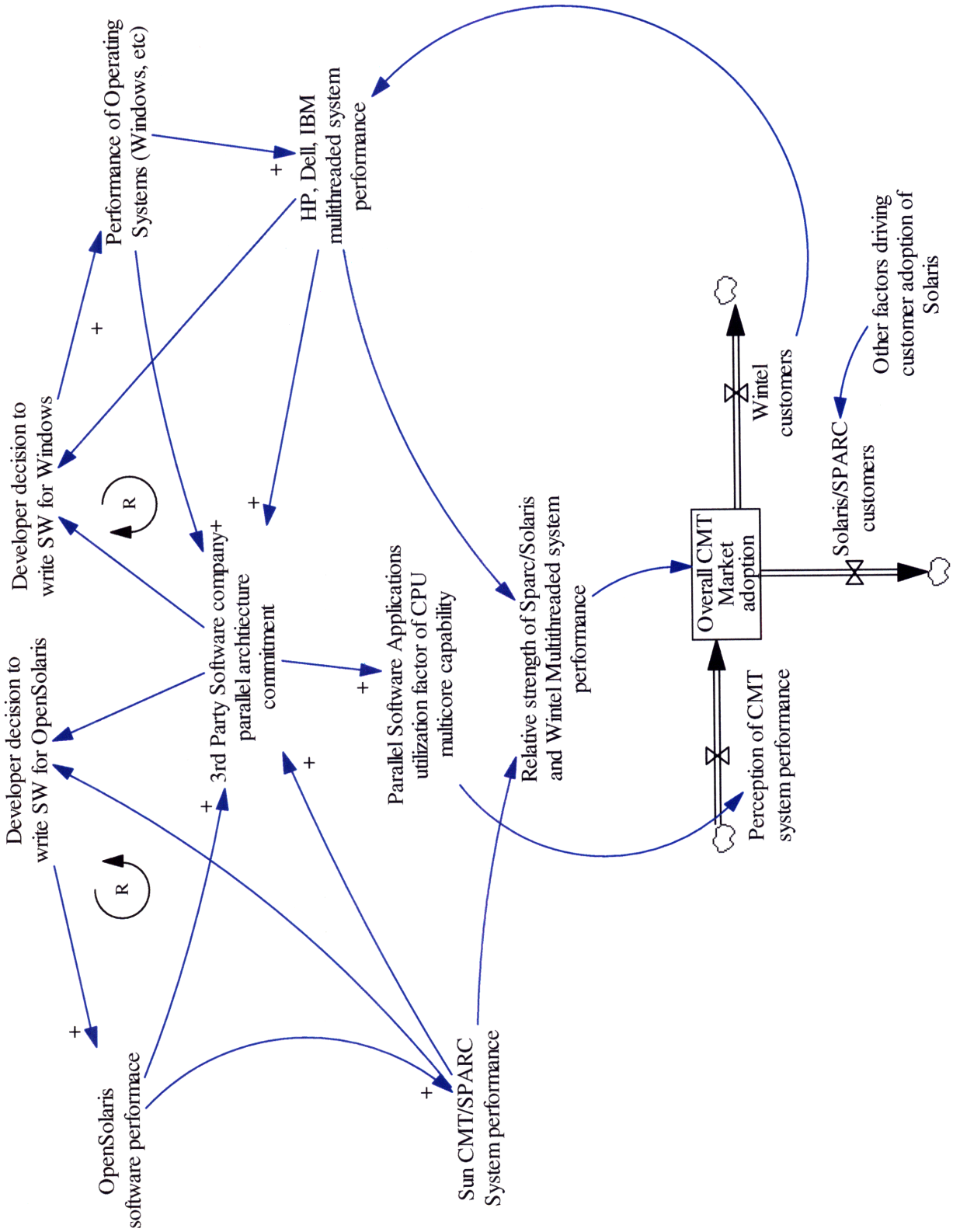


Figure 29: Diagram of Relative Strength of System Performance on Customer Adoption

Key Insights from this loop include:

- A key assumption in this model is that performance is the key parameter in the decision process that customers will use to purchase either Sun or Windows / Intel systems.[1] In other words:

  A customers' decision to purchase a Sun Solaris/CMT SPARC system or a multi-core Windows/x86 system is based mainly on the perceived performance difference between the two systems.

- The increased perception of the performance improvements of a multithreaded system over a traditional single threaded system will drive the overall CMT market adoption to grow.
- The increased performance of multithreaded systems will influence the developers' decision to write more parallel programs.

Another way to represent the relationship between system performance, the developers decision to write parallel programs for either OpenSolaris or Windows, and the overall CMT Market adoption can be seen in the uses tree for each of these variables below:

[1] Please note, the term Wintel is used for Windows and Intel based systems, although it also includes AMD based systems as well

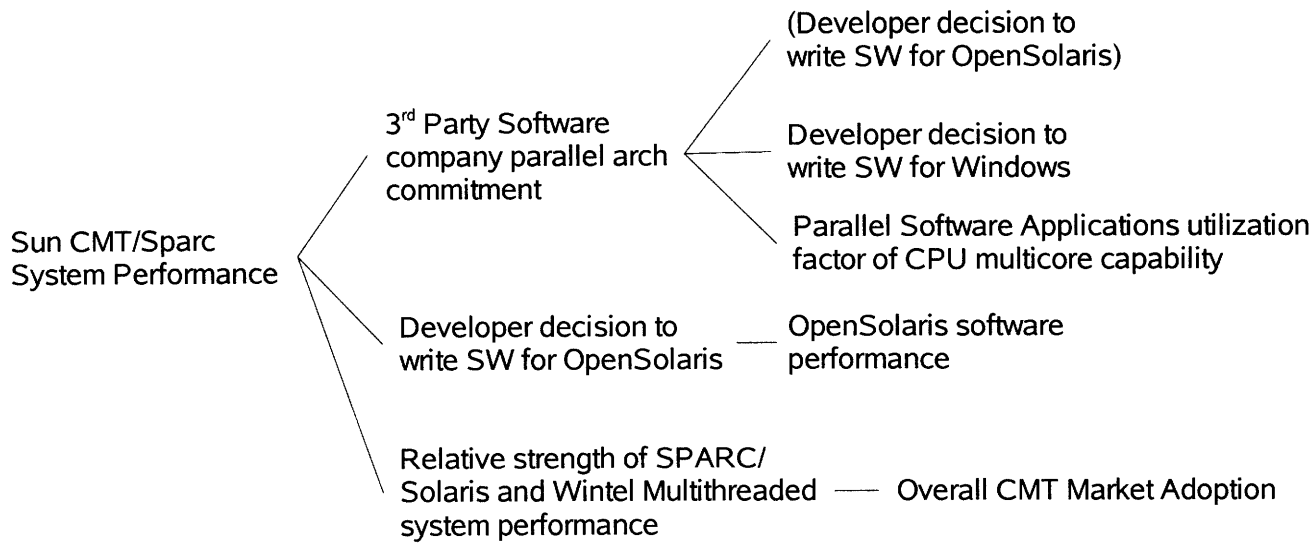


Figure 30: Uses Tree of Sun CMT/SPARC System Performance

Similarly, we can see a similar relationship between Sun's competitors' performance and their influence on developers to write software for the competition's operating system:

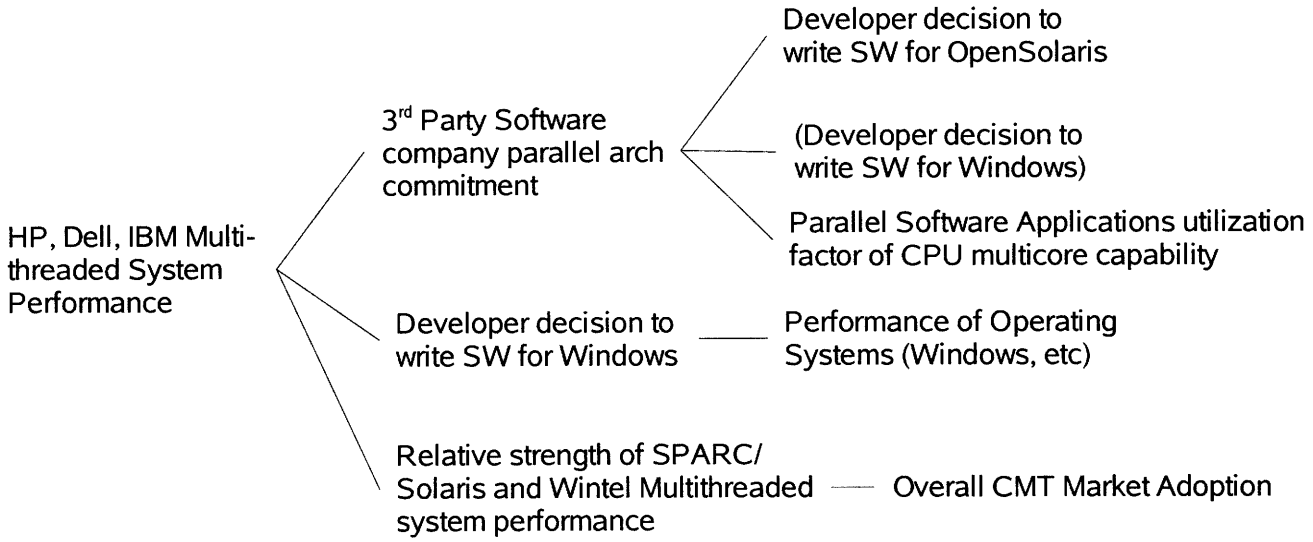


Figure 31: Uses Tree of Sun's competitors' System Performance

4.2 Other Industry Efforts to Improve Parallel Software Availability

The OpenSPARC effort is not the only effort available in the industry today to help educate students. Stanford University has also realized the need to educate students in parallel programming. In response to this need, Stanford has opened the Pervasive Parallelism Laboratory, with the help of major industry players including Sun, AMD, NVIDIA, IBM, Intel, and HP. "The Pervasive Parallelism Laboratory is hoping to produce hardware and software that will enable software developers to expose the parallelism within their programs and exploit the parallel processing capabilities of the hardware they use. This is a difficult task at the best of times, and this difficulty has been amplified by a relative lack of education among software developers."[10] Microsoft and Intel have chosen to use an alternative approach to using the open source communities to boost their multithreading software capability. Per a recent article in the Info World dated in March, 2008, Microsoft and Intel are spending over $20 million dollars in order fund their own research into new ways to program software for multicore processors.

Many other companies in the computer industry are also helping to reduce the manpower required to write parallel software by creating open sourced tools that ease the creation of code. As discussed in previous sections, OpenMP is an open source application programming interface that supports parallel software application development. OpenMP Architecture Review Board is a non-profit organization that owns the OpenMP brand, oversees the specification, and produces and approves new versions. Also, Pthreads is an application programming interface for creating and manipulating threads. Pthreads is a POSIX standard for threads, and is overseen by POSIX (Portable Operating System Interface), a family of standards specified by IEEE.

Recently, Intel released its own open source software called Intel Threading Building Blocks, available for free download. "Intel® Threading Building Blocks is a runtime-based parallel programming model for C++ code that uses threads. It consists of a template-based runtime library to help you harness the latent performance of multicore processors." Intel® Threading Building Blocks is intended to assist programmers in

writing scalable applications by specifying tasks instead of threads, emphasizing data parallel programming, and taking advantage of concurrent collections and parallel algorithms.

4.3 Model of Mature Open source effort: OpenOffice.org

OpenSPARC is only one among several open source communities that Sun has initiated. Sun has proven that it can be successful at sponsoring open source communities. Over the last 8 years, Sun has invested its resources into developing the successful community of OpenOffice.org for an open source office suite. Below is a system dynamic model illustrating how the OpenOffice.org community competes with Microsoft Office, and in turn creates competition and customer choice which provides benefit to the overall customer base for office suites.

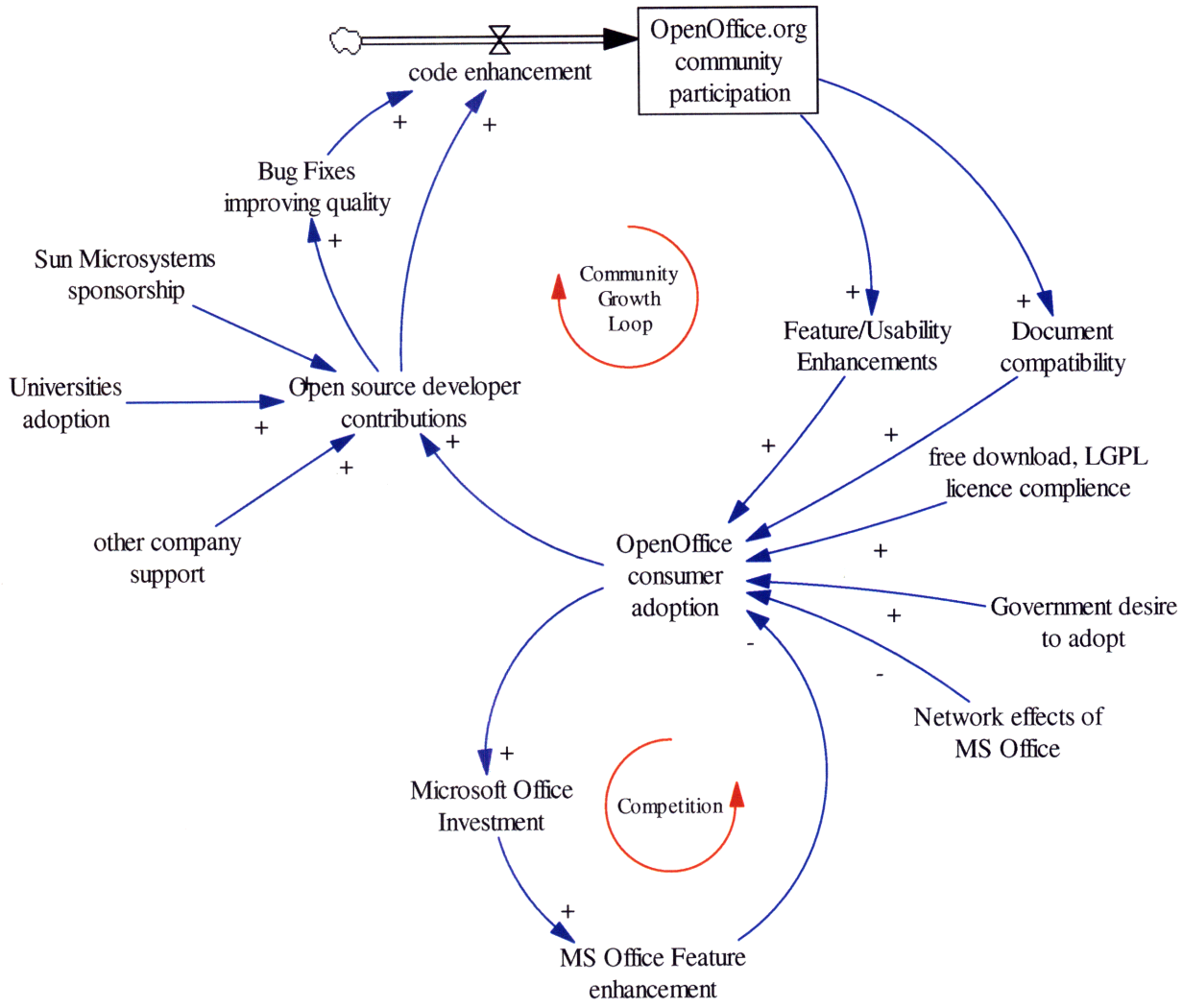


Figure 32: Mature Open Source Community Model of OpenOffice.org

In the OpenOffice.org model, there is a large install base of customers who have already standardized on using Microsoft Office. OpenOffice.org is a threat to this adoption since it is a free alternative which is competitive on feature set, cost and security. Since there is no return on revenue loop directly to Sun on the adoption of OpenOffice.org, Sun primarily invests in OpenOffice.org in order to gain customer-base brand recognition, communication and trust. Sun has proven the success of using an open source development model to create widely adopted software which competes heavily with the most market-dominating software available today, Microsoft Office. In 2008, InfoWorld's 2008 Bossies recognized the top free and open products for business, IT, and

personal productivity. According to InfoWorld, "Some of our picks were easy. For office productivity suite, what else but OpenOffice.org?".

While Sun Microsystems is the primary contributor to OpenOffice.org, other contributors include Novel, RedHat, RedFlag CH2000, IBM, Google and 450,000 individuals from all over the world. The power of the open source developer community is illustrated in this model of a mature and successful open source community.

Chapter 5: Conclusions

This section summarizes the assumptions and mental models that were used to create the CMT system dynamics model. These are the same major assumptions and mental models that drove Sun Microsystems to open source its multicore, multithread UltraSPARC T1 and T2 chip hardware implementation.

5.1 Key Assumptions Contained within the CMT System Dynamics Model

By looking at each of the building blocks within the overall CMT system dynamics model, we can pull out the key assumptions embedded within the full system model:

A) Sun Microsystems is a systems company, and as such is specifically positioned to capitalize on the open sourcing of the main system components, specifically the Operating System (OpenSolaris) and the CPU (OpenSPARC). Most of Sun's major competitors are not positioned to sell servers to the marketplace that contain self-owned intellectual property at the component level. Instead, Sun's competitors use mainly 3rd party IP in their system components, specially the Operating System (Microsoft Windows or Linux) and CPU (Intel x86 or AMD x86).

B) The key to unlocking multicore and multithreaded system performance lies in the parallelization of software applications. The level of parallelization of the software applications directly affects the utilization of the CPU multicore and multithreaded

capability. Without parallelized software, the same traditional performance measurements govern multicore systems as single core and single threaded systems. Since multicore and multithreaded systems are designed to run simultaneous but slower, lower power single compute cycles, multithreaded systems running single-threaded applications will not demonstrate any performance improvements over the traditional single-threaded servers. Thus, the parallelized software applications are a critical ingredient needed to unlock the performance enhancements of multicore and multithreaded servers over traditional single core, single threaded servers.

C) By open sourcing the key components (operating system and CPU) of Sun's systems, developers will be more likely to choose to develop programs for Sun servers over the competition. The decision variables that Sun has tipped in its favor by open sourcing these components include:

- A higher exposure to Sun's technology to students early in their career, which leads to a familiarity with Sun and a higher perception of Sun's success in the marketplace.
- Easier access to developers of operating system source code and hardware simulators needed for software code development and validation.
- Reduced effort to develop parallel software applications by increased educational resources and easy access to tool kits for parallel program development
- Increased perception of Sun's success in the market place.

D) The loss of competitive information Sun risks as a result of open sourcing its operating system and CPU hardware implementation is outweighed by the benefit gained from the innovation developed in the open source communities which contribute to the improvement of Sun's overall system performance.

E) The tipping point in CMT market adoption is caused by proven performance improvements of multicore and multithreaded servers over traditional single threaded servers. Sun believes that open sourcing the operating system and CPU hardware

implementation of its multithreaded systems will be the small investment needed to ignite an epidemic of market adoption of multithreaded systems.

Chapter 6: Future Work

Creating a fully functional and well calibrated system dynamics model of a large, complex and diverse business ecosystem such as the complete computer server marketplace, including hardware, operating systems and application software is a major undertaking, which requires extensive data collection and analysis at the level typically expected in a PhD thesis. The scope of this study was specifically focused on the initial creation of the system dynamics model of the CMT ecosystem in order to explore and document the assumptions and mental models used in the decision to open source Sun's operating system and CMT chip hardware implementation. The following section outlines the future research that should be conducted in order to determine if the mental models and assumptions about the structure of the CMT ecosystem represent reality at a quantitative level.

6.1.1 CMT System Dynamics Model Completion

The following areas of data collection should be conducted in order to fully complete the system dynamics model:

1. Fully completing the CMT system dynamics models will require adding units to each variable, initial values, and specifying the equations. An important step in validating the structure of a system dynamics model is achieving units that balance as a result of the specified equations. This step will often correct any inaccurate mental models used to create the system dynamics model.

2. Quantitative validation of the CMT system dynamics model by comparing the output of CMT market adoption to current state of the market. By validating the output of the model, you will validate Sun's assumptions about the structure of the model.

3. Utilizing the CMT system dynamics model to predict how long will it take parallel software to catch up to the performance capability of the hardware. This time study will lead to an accurate prediction of the tipping point in which the market will fully adopt multithreaded, multicore processors.

6.1.2 Using the model to Make Quantitative Predictions

From a simplified quantitative iteration of the CMT system dynamic model created during this study, we can formulate a basic prediction of the growth of the CMT market based on the increased availability of parallel software applications over time. We can see from the graph below that because initially there are so many software applications in need of parallelization, there is a rapid increase in the number of software programs that are parallelized. However, as the number of programs that need parallelization starts to decrease, the growth rate of parallel software available in the market will start to slow, but always remains greater then zero. We can also see from the graph below that the CMT market will start to grow slowly at first, but as the availability of parallel software applications increases, the market will reach a tipping point, and adoption will start to accelerate. By completing the quantitative version of the CMT system dynamics model, we should be able to answer questions such as; will this tipping point happen in the next five years? Or will it take ten years to achieve?

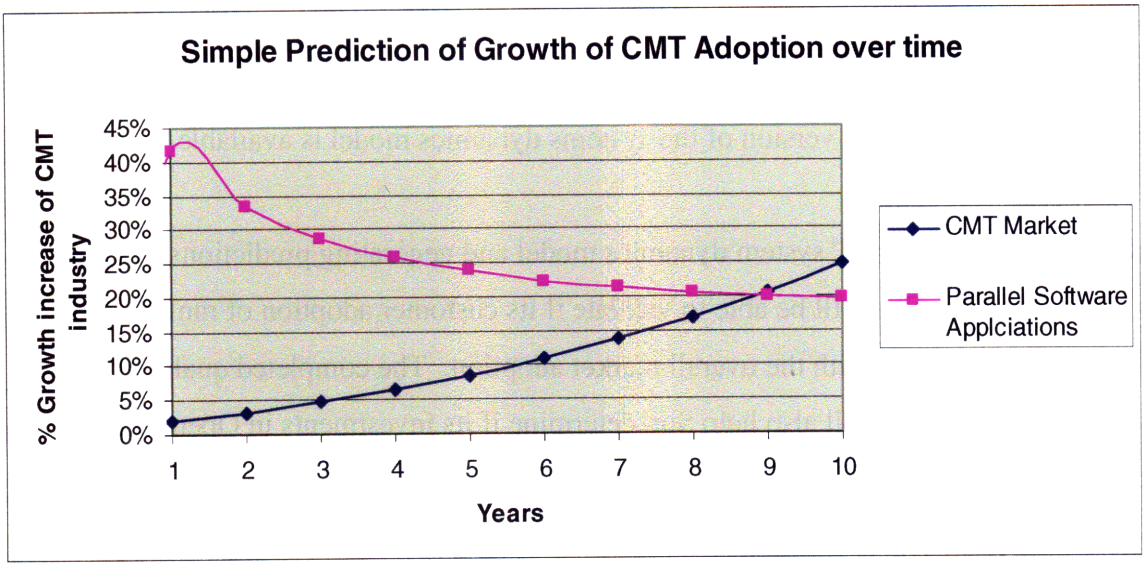


Figure 33: % Increase over time of CMT customers as a result of parallel software availability

From Sun's Q1FY09 Quarterly Earnings Announcement, we can see that indeed Sun is starting to see the return on investment of its CMT servers into the market place.

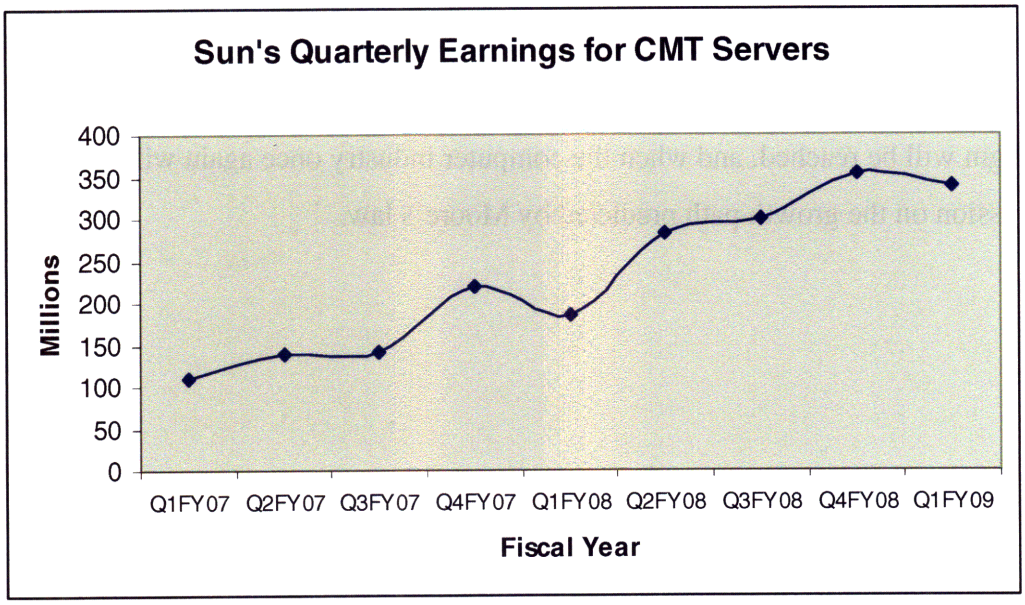


Figure 34: Sun's Quarterly CMT Server earnings (Q1FY07 – Q1FY09)

Will Sun see an increase in the adoption of its CMT servers once the market tipping point is reached? These questions are some examples of the powerful predictions that can be made once a qualitative version of the systems dynamics model is available.

By completing the CMT system dynamics model and generating predictions of CMT market adoption, Sun will be able to validate if its customer adoption of multithreaded servers is keeping up with the overall market adoption. The completed qualitative version of the model will also help Sun determine if its investments in OpenSolaris and OpenSPARC have generated the effects that we have outlined in this study.

Throughout this study, we have learned that the software community is dramatically lagging behind in developing applications optimized for a multicore and multithreaded processor design. The paradigm shift to parallelization will require the entire computer industry to contribute to the multicore and multithreaded epidemic in order to accelerate the availability of parallel software applications. This study has created the system dynamics model that explains the structure of the CMT business ecosystem, and the relationship between the open source communities and market adoption. The next steps of this study include completing a quantitative analysis of the CMT system dynamics model to help the industry predict the time delays expected before the parallelization paradigm will be reached, and when the computer industry once again will resume its progression on the growth path predicted by Moore's law.

Bibliography:

1. Alexandra Fedorova, Margo Seltzer, Christopher Small, and Daniel Nussbaum, Performance of Multithreaded Chip Multiprocessors And Implications For Operating System Design, Harvard University, Sun Microsystems, http://www.eecs.harvard.edu/~margo/papers/usenix05/paper.pdf

2. Vance, Ashlee; The Register "Intel begs for multi-threaded software aid", Servers, 22nd August 2006. http://www.theregister.co.uk/2006/08/22/intel_suite_help/

3. James Dixon, The Beekeeper Model; Pentaho Community, May 10, 2007. http://wiki.pentaho.com/display/BEEKEEPER/3.+The+Beekeeper+Model

4. Thibodeau, Patrick, "Q&A: Schwartz says financial meltdown plays into Sun's hands" ComputerWorld. October 13, 2008. http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyName=Default&articleId=9117060&taxonomyId=0&pageNumber=2

5. Mark Tolliver, "Open Source risks and Responsibilities." OS News. September 13, 2007. http://www.osnews.com/story/18610/Open_Source_Risks_and_Responsibilities/

6. Bruce Perens, The Open Source Definition, October, 2008. http://ldp.dvo.ru/LDP/LGNET/issue26/perens.html,

7. Elizabeth Montalbano, "Microsoft, Intel to fund multicore research at UC Berkeley." Info World. March 17, 2008 http://www.infoworld.com/article/08/03/17/Microsoft-Intel-to-fund-multi-core-research-at-UC-Berkeley_1.html?t=sendEmail.jsp

8. Mark Pacifico, Mike Merrill, CMSC 411 Project, A General Overview of Parallel Processing, Fall 1998 http://www.cs.umd.edu/class/fall2001/cmsc411/projects/parallel2/history.html

9. Sun Microsystems, Q1FY09 Quarterly Results Release, October 30, 2008 http://www.sun.com/aboutsun/investor/earnings_releases/call_slides.jsp

10. Bright, Peter, "Industry, Stanford hope to fix what ails parallel processing". Ars Technica. May 1, 2008. http://arstechnica.com/news.ars/post/20080501-industry-stanford-hope-to-fix-what-ails-parallel-processing.html

11. Clarke, Gavin. "Sun juggles love of code with need for cash, Don't attack us... attack them!" The Registar, May 6, 2008 http://www.theregister.co.uk/2008/05/06/sun_open_source/

12. Shankland, Stephen. "Sun, allies broaden open-source chip push." News.Com. May 15, 2007. http://news.cnet.com/Sun-allies-broaded-open-source-chip-push/2100-7344_3-6183562.html

13. Shankland, Stephen. "Fujitsu designs Sparc chip." CNET News. Published on November 5, 1998. http://news.cnet.com/2100-1001-217537.html&dd.ne.tx.fs5.1105

14. Rothschild,Michael; Bionomics: Economy As Ecosystem, H. Holt; 1st edition, 1990

15. Marson, Ingrid. "Open source firms claim business model benefits." ZDNet UK. October 4, 2006. http://www.builderau.com.au/news/soa/Open-source-firms-claim-business-model-benefits/0,339028227,339246880,00.htm#

16. Hinkle, Mark. "Microsoft ISV Benefits from Open Source Development Model" Socialized Software. May 5, 2008. http://socializedsoftware.com/2008/05/05/microsoft-isv-benefits-from-open-source-development-model/

17. Courtney, Philip E. "Parallel computing cart still chasing the horse - includes related article on growth of IBM S/390 Developers' Association" Industry Trend or Event Software Magazine. Nov, 1995.
http://findarticles.com/p/articles/mi_m0SMG/is_/ai_17908884

18.. Funk, A., Basili, V., Hochstein, L., Kepner, J. "Analysis of Parallel Software Development using the Relative Development Time Productivity Metric," *CTWatch Quarterly*, Volume 2, Number 4A, November 2006
http://www.ctwatch.org/quarterly/articles/2006/11/analysis-of-parallel-software-development-using-the-relative-development-time-productivity-metric/

19. Copty, Nawal. "OpenMP Support in Sun Studio Compilers and Tools." Sun Developer Network. December 13, 2005.
http://developers.sun.com/solaris/articles/studio_openmp.html

20. A System Dynamics Approach to Study Virtual Communities, Yan Mao, Julita Vassileva, Winfried Grassmann, IEEE, 0-7695-2755-8/07, 2007

21. Diker, Vedat G, A Dynamic Feedback Framework for Studying Growth Policies in Open Online Collaboration Communities, College of Information Studies - University of Maryland, Proceedings of the Tenth Americas Conference on Information Systems, New York, New York. August 2004

22. Isaacson, Cory. Software Pipelines and SOA: Releasing the Power of Multi-Core Processing , Addison Wesley Professional, December 26, 2008

23. Fawcett,Glenn, "Optimizing Oracle DSS operations with CMT based servers." Glenn Fawcett's Weblog. May 29, 2008.
http://blogs.sun.com/glennf/entry/optimizing_oracle_dss_operations_with

24. Fawcett,Glenn, "Throughput computing series: System Concurrency and Parallelism". Glenn Fawcett's Weblog. Jan 16, 2008. http://blogs.sun.com/glennf/entry/throughput_computing_series_system_parallelism1, accessed on Nov 29, 2008

25. Sterman, John D; Business Dynamics, Systems Thinking and Modeling for a Complex World, Irwin McGraw-Hill, Boston, 2000

26. Mead, Nancy R.; Saiedian, Hossein; Rube, Gunther; Bagert, Donald J; Panel: Shortages of Qualified Software Engineering Faculty and Practitioners: Challenges in Breaking the Cycle; ICSE 2000 Limerick Ireland; Copyright ACM 2000 1-581 13-206-9/00/6

27. Weaver, David, Sun Microsystems, OpenSPARC Internals; Published by Sun Microsystems, Inc, October 2008; ISBN 978-0-557-02974-8

28. Coar, Ken. "The Open Source Definition". Open Source Initiative, July 7, 2006. http://www.opensource.org/docs/osd

29. Peters, Stormy. "Why do open source software developers write code for free?", Open Source. May 16, 2007 http://www.openlogic.com/blogs/2007/05/why-do-open-source-software-developers-write-code-for-free/

30. Craig Bailley, "Craig Bailley on Microsoft", Parallel Computing, October 25, 2008, http://www.craigbailey.net/live/post/2008/10/25/Parallel-Computing.aspx