We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

3

Software Development for Parallel and Multi-Core Processing

Kenn R. Luecke
The Boeing Company
USA

1. Introduction

The embedded software industry wants microprocessors with increased computing functionality that maintains or reduces space, weight, and power (SWaP). Single core processors were the key embedded industry solution between 1980 and 2000 when large performance increases were being achieved on a yearly basis and were fulfilling the prophecy of Moore's Law. Moore's Law states that "the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years."[1] With the increased transistors, came microprocessors with greater computing throughput while space, weight and power were decreasing. However, this 'free lunch' did not last forever.[2] The additional power required for greater performance improvements became too great starting in 2000. Hence, single core microprocessors are no longer an optimal solution. Although, distributed and parallel programming solutions provide greater throughput, these solutions unfortunately increase SWaP. The most likely solution is multi-core processors which have been introduced into the embedded processor markets. Most microprocessor manufacturers have converted from developing single core processors to multi-core processors. With this conversion, the prophecy of Moore's Law is still being achieved. See Figure 1 and notice how the single core processors are not keeping pace with the multi-core processors. Multi-core processors increase throughput while maintaining or reducing SWaP for embedded environments which make them a good hardware solution for the aerospace industry. Intel, in particular, has estimated that by 2011, 95% of all the processors it ships will contain a multi-core design. However, the software market shows less optimism with multi-core processors. For instance, only 40% of software vendors thought their tools would be ready for multi-core processing by 2011. The reasons for software engineering's lack of excitement with multi-core processors include the following drawbacks:

- Lack and immaturity of multi-core specific development and debug software tools.
- Lack of multi-core processor standards.
- Lack and immaturity of multi-core enabled system software.

[1] http://en.wikipedia.org/wiki/Moore's_law
[2] Sutter, H., (March, 2005). "The free lunch is over. A fundamental turn toward concurrency in software," *Dr. Dobb's Journal, Volume 30, Number 3*.

- Lack of parallel programming experience by the software community.
- Lack of parallel programming models to support these multi-core processors.
- An abundance of differentiated multi-core processors from multiple suppliers. Greater differentiation with inexperience can be problematic for software developers converting applications for multi-core processors.

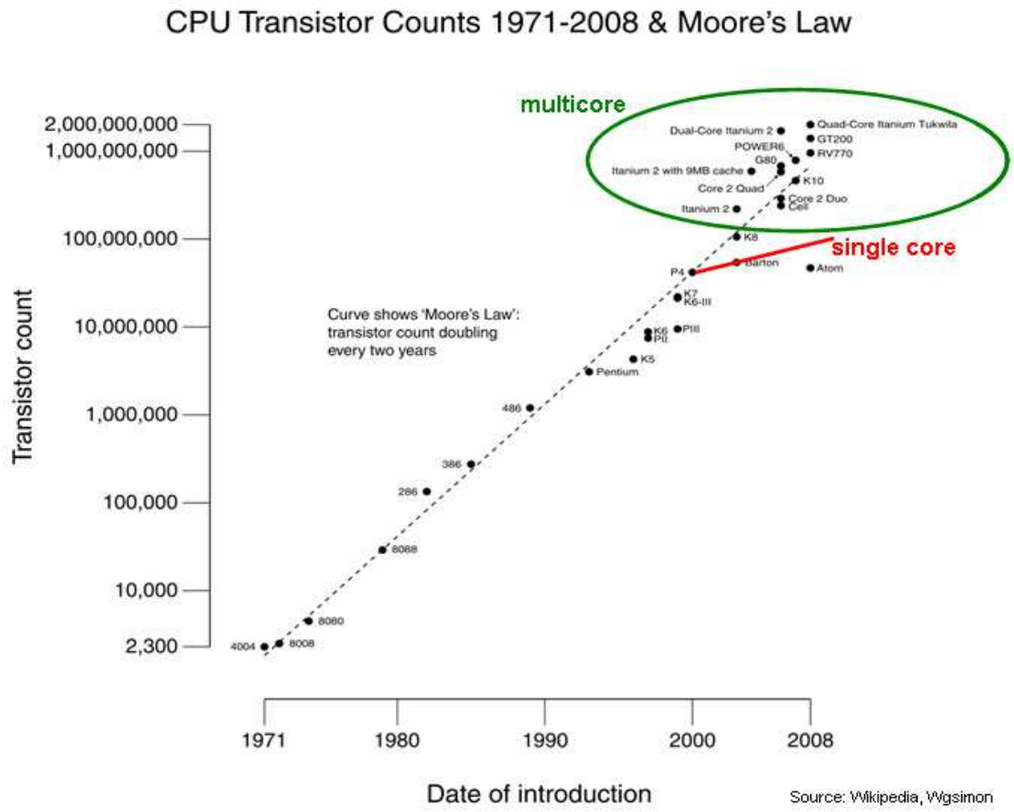


Fig. 1. Processor Transistor Counts and Moore's Law[3].

These problems led Chuck Moore, a Senior Fellow at AMD, to state "To make effective use of Multi-core hardware today, you need a PhD in computer science."[4] Therefore, multi-core software development has fallen behind multi-core hardware development. This chapter will provide information on the current best technologies, tools, methodologies, programming languages, models, and frameworks for software development on multi-core processors. Where different software development options exist, comparisons and recommendations will be provided to the reader.

2. Multicore definition

Previous multiprocessing, as opposed to multi-core processing, solutions, such as parallel and distributed programming, involved two or more processors, which doubled, tripled, or

[3] Fittes, Dale, (October 30, 2009) Using Multicore Processors in Embedded Systems – Part 1, *EE Times*.
[4] Moore, Chuck, (May 12, 2008) "Solving the Multi-core Programming Problem", *Dr. Dobbs Journal*.

even quadrupled the board space, weight, and power (SWaP) consumed and heat generated by the processing system. These solutions could comprise large networks leading to data latencies between processing components. However, multi-core processors place multiple processing cores on a single chip to increase processing power without noticeably increasing the system's SWaP and heat dissipation. Also, with multiple cores on a single chip the data latencies of distributed programming are mostly negated. With multi-core processing, the computer industry continues pushing the performance/power envelope through parallel processing rather than increasing the processor clock speed. For the most part, serial computing has been the standard software development model, with multiple cores on a processor, now parallel computing is emerging as the new standard and very few programmers are well versed in parallel computing. A multi-core processor, in general, appears similar to the dual core and quad core processors displayed in Figure 2. In both cases, each core has an associated L1 cache while the L2 cache is shared between all the cores. For systems with L1, L2, and L3 cache, normally the L3 cache is shared between all cores, each core has its own segregated L1 cache, and the L2 cache may be shared between cores or segregated L2 caches will be devoted to each core.

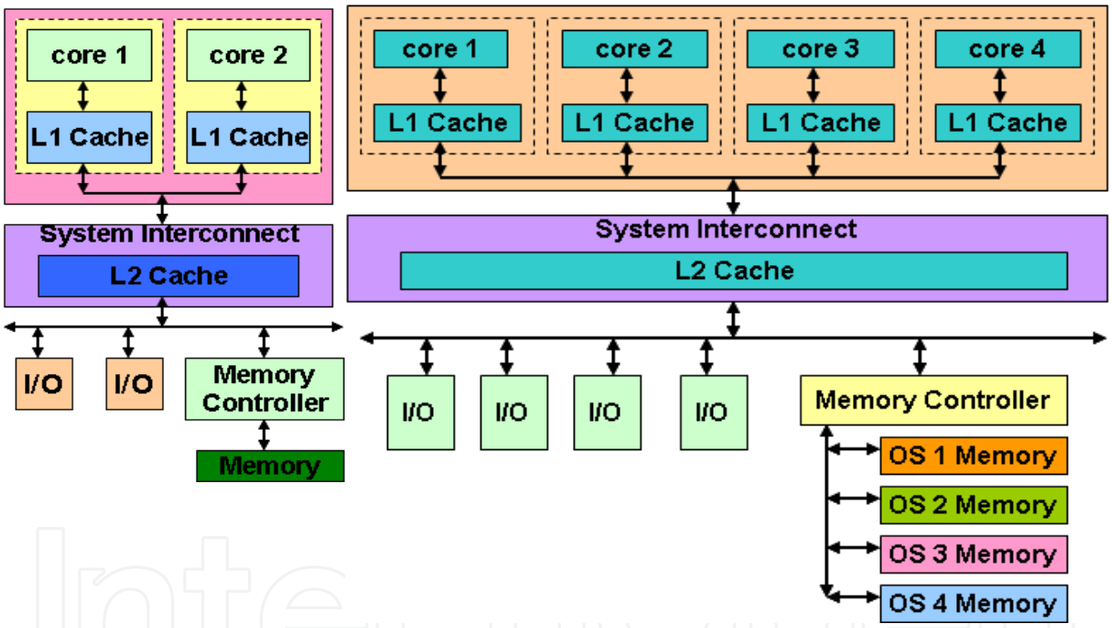


Fig. 2. Example Dual Core (left) and Quad Core (right) Multi-core Processors.

3. Multiprocessing models and frameworks

Traditionally, there were two multiprocessing models: Asymmetric Multi-Processing (AMP) and Symmetric Multiprocessing (SMP). For highly integrated processing, AMP designs incorporate several cores on a chip with each processor using its own L1 cache, and all processors share a common global memory. The AMP model can incorporate either heterogeneous cores executing different operating systems (OS) or homogeneous cores executing the same OS. With heterogeneous cores, the AMP architecture looks like a Digital Signal Processing (DSP) architecture. In AMP designs, application tasks are sent to the system's separate processors. These processors are collocated on the same board, but each is

a separate computing system with its own OS and memory partition within the common global memory. See Figure 3.

The advantages of the AMP multiprocessing model include:

- The operating systems, tasks, and peripheral usage can be dedicated to a single core. Hence, it offers the easiest and quickest path for porting legacy code from single core designs to multi-core designs. Therefore, it is the easier multiprocessing model for serial computing software engineers to start with.

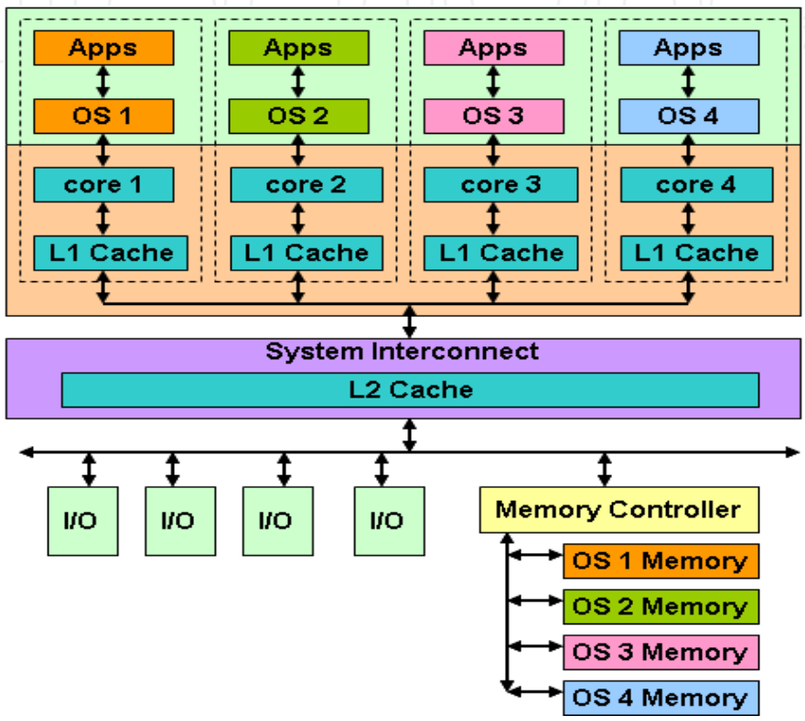


Fig. 3. Traditional AMP Model.

- Migrating existing (non-SMP) OSs to the model is relatively simple and usually offers superior node-to-node communication compared to a distributed architecture.
- AMP also allows software developers to directly control each core and how the cores work with standard debugging tools and methodologies. AMP supports the sharing of large global memories asymmetrically between cores.
- AMP provides software developers with greater control over efficiency and determinism.
- AMP allows engineers to embed loosely coupled applications from multiple processors to a single processor with multiple cores.

The disadvantages of the AMP multiprocessing model include:

- For tightly coupled applications, AMP approaches work best when the developers need no more than two cores while developing a solution. As more cores are added, the AMP multiprocessing model becomes exponentially more difficult especially for tightly coupled applications executing on all cores.

- AMP can result in underutilized processor cores. For example, if one core becomes busy, applications running on that core cannot easily migrate, to an underutilized core. Although dynamic migration is possible, it involves complex check pointing of the application's state which can result in service interruption while the application is stopped on one core and restarted on a different core. This migration may be impossible if the cores use different OSs.
- None of the OSs owns the entire application system. The application designer must manage the complex tasks of handling shared hardware resources. The complexity of these tasks increases significantly as more cores are added. As a result, AMP is ill-suited for tightly coupled applications integrating more than two cores.
- Memory latency and bandwidth can be affected by other nodes.
- The AMP multiprocessing model does not permit system tracing tools to gather operating statistics for the multi-core chip as a whole since the OSs are distributed on each core. Instead, application developers gather this information separately from each core and then combine the results for analysis purposes. This is only a concern for systems where the applications on the individual cores are tightly coupled.
- Cache "thrashing" may occur in some applications.

In SMP architectures, each node may have two or more processors using homogeneous cores, but not heterogeneous cores, while the multiple processors share the global memory. In addition, the processors may also have both local and shared cache, and the cache is coherent between all processors and memory. See Figure 4. SMP executes only one copy of an OS on all of the chip's cores or a subset of the chip's cores. Since the OS has insight into all system elements, it can transparently and automatically allocate shared resources on all cores. It can also execute any application on any core. Hence, "SMP was designed so you can mimic single-processor designs in a distributed computing environment," said Enea's Michael Christofferson[5]. The OS provides dynamic memory allocation, allowing all cores to draw on the full pool of available memory, without a performance penalty. The OS may use simple POSIX primitives for applications running on different cores to communicate with each other. POSIX primitives offer higher performance and simpler synchronization than the AMP system networking protocols.

Other SMP multiprocessing model advantages include:

- A large global memory and better performance per watt is due to using fewer memory controllers. Instead of splitting memory between multiple central processing units (CPU), SMP's large global memory is accessible to all processor cores. Data intensive applications, such as image processing and data acquisition systems, often prefer large global memories that can be accessed at data rates up to 100s of Megabytes/second (Mbytes/sec).
- SMP also provides simpler node-to-node communication, and SMP applications can be programmed to be independent of node count. SMP especially lends itself to newer multi-core processor designs.
- Systems based on SMP, have the OS perform load-balancing for the tasks between all cores.

[5] Morgan, Lisa L., (December 15, 2006), Making the Move to Multicore, *SD Times*.

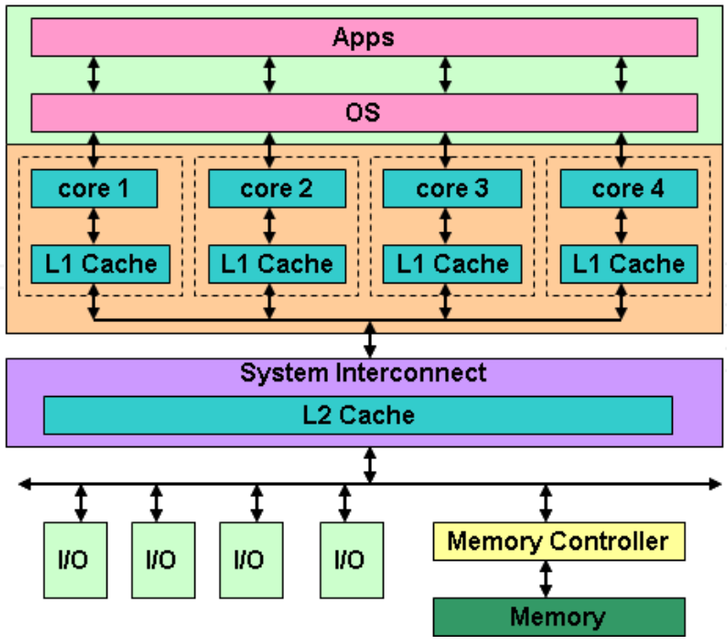


Fig. 4. Traditional SMP Multiprocessing Model Example.

- One copy of an OS can control all tasks performed on all cores, dynamically allocating tasks or threads to the underutilized core to achieve maximum system utilization.
- The SMP multiprocessing model permits system tracing tools to gather operating statistics for the multi-core chip as a whole, providing developers insights into optimizing and debugging applications. The tracing tools can track thread migration between cores, scheduling events, and other information useful for maximizing core utilization.
- An SMP approach is best for a larger number of cores and for developers who have time to adequately develop a long term solution that may eventually add more cores.
- SMP, versus AMP, is usually the preferred choice for applications implementing dynamic scheduling.

The disadvantages of the SMP multiprocessing model include:

- The memory latency and bandwidth of a given node can be affected by other nodes, and cache "thrashing" may occur in some applications.
- Legacy applications ported to an SMP environment generally require a redesign of the software. Legacy applications with poor synchronization among threads may work incorrectly in the SMP concurrent environment. Therefore, an SMP approach is better for software developers with parallel computing experience.
- Enea's Christofferson said that in many designs there are components within an operating system that may have hidden requirements that may not be running at the same time as another thread. To avoid the problem, Christofferson recommended that designers consider all OS and application threads to make sure there are no concurrency problems.[6]

[6] Morgan, Lisa L., (December 15, 2006), Making the Move to Multicore, *SD Times*.

- When moving legacy architectures from single core processing to multi-core processing, the major issue is concurrency. In a single operating environment, running multiple threads is a priority, so two threads with different priority levels can execute in parallel when they are distributed to different cores.
- SMP systems exhibit non-determinism. Hence any computing solutions that require determinism may need to stay away from an SMP model.

After listing the advantages and disadvantages of both AMP and SMP, a comparison between both multiprocessing models on several important programming concepts would be beneficial. See Table 1. With most programming concepts, the support that AMP provides is diametrically different from the support provided by SMP. However, as a software architect or developer of a system being ported to a multi-core processor, you want AMP support for some programming concepts and SMP support for other programming concepts. It was for this very reason that RTOS suppliers began to provide CPU affinity with their SMP support. What has become more prevalent in the past several years is developing hybrid models that combine some AMP support with some SMP support based on the system needs of the computing solution being developed. Two of the more popular hybrid models include:

- Combined AMP/SMP Model which executes both processing models on one processor. For example, for a quad-core processor, two cores will be executing an AMP model while the remaining two cores will be executing a SMP model. See Figure 5. In this hybrid model, there is no cross pollination between the models running on any of the cores. One benefit of this model is that architects can implement tasks that achieve better performance on AMP such as task parallelism on the AMP cores and tasks that achieve better performance on SMP such as data parallelism on the SMP cores, resulting in an overall system performance than an AMP or SMP only system.
- Supervised AMP Model which includes a layer of software executing between the OSs and the cores. The supervisor's primarily benefit is additional communication software that allows for improved system communication between the OSs running on the different cores. The benefits of this include:
  - Improving scalability for additional cores.
  - Providing better system debugging between processes on different cores.
  - Enabling reboot of individual cores on your system.[7]

Hence, Supervised AMP model has improved system debugging capabilities over a system implementing a traditional AMP model. See Figure 6.

Several embedded software frameworks have been developed for multi-core processors, but more are needed for improved software development and performance. The frameworks discussed in the rest of this chapter are a sampling of available frameworks. The mention of each framework is not intended as a recommendation. The Georgia Institute of Technology, through the Software and Algorithms for Running on Multi-core (SWARM) program, developed a framework consisting of portable open source parallel library of basic primitives that exploit multi-core processors. This framework allows developers to implement efficient parallel algorithms for operations like combinatorial sorting and

[7] Wlad, Joe. (2011), Freescale Multi-Core Forum, Freescale Multi-core Forum, St. Louis, September, 2011.

Programming Concept	AMP	SMP
Seamless resource sharing	No	Yes
Scalable beyond dual core	No/Complicated for tightly coupled apps	Yes
Mixed OS environment (ex: VxWorks & Linux)	Yes	No
Dedicated processor by function (CPU affinity)	Yes	Yes/No. CPU affinity is not supported in traditional SMP models, but most RTOS suppliers provide CPU affinity for their SMP models.
Inter-core messaging	Slower (application)	Fast (OS primitives)
Thread synchronization between cores	No/Complicated	Yes
Dynamic load balancing	No	Yes
System-wide debug and optimization	No/Complicated for tightly coupled apps	Yes
Migrating Legacy Apps/New App Development	Best at Migrating Legacy Apps. Good choice for New App Development.	Best for New App Development
Data/Task Parallelism	Task preferred	Data preferred
Engineer experienced in Serial Computing Only	Better choice than SMP	More difficult for a novice parallel computing developer

Table 1. AMP and SMP Model Comparisons.

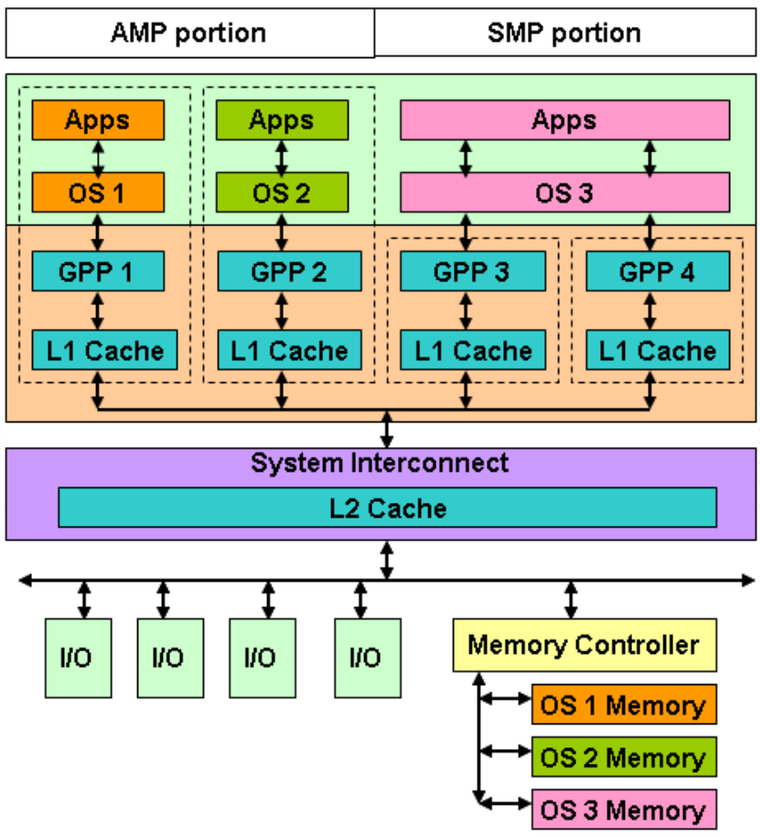


Fig. 5. Combined AMP/SMP Model.

selection algorithms. The University of California-Berkeley (UCB) and University of Illinois at Urbana-Champaign (U of I) are concentrating on software frameworks for multi-core processors. Both universities were partially funded by a $10 million grant from Intel and Microsoft. In particular, UCB is concentrating on developing frameworks and a composing language to assist programmers in creating and coordinating parallel programming models. Meanwhile, U of I is exploring new frameworks for extracting parallelism from serial code and developing software component building blocks required for parallel programming frameworks. Several software and hardware companies including AMD, IBM, Hewlett-Packard, Intel, and NVidia are funding Stanford University's Pervasive Parallelism Lab to investigate new parallel programming models including improved synchronization techniques between the cores on a multi-core processor.

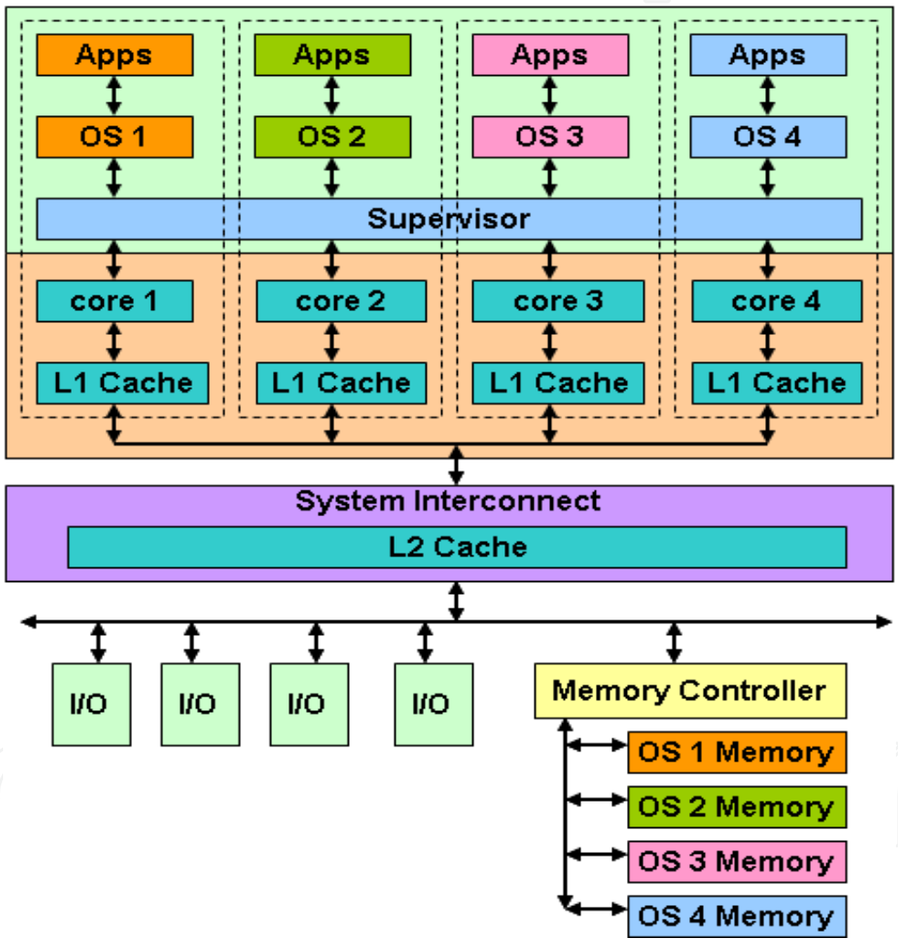


Fig. 6. Supervised AMP Model.

Of software suppliers, Microsoft has included multi-core support for its .NET framework. The .NET framework contains the Task Parallel Library (TPL) with software to support task parallelism and the Parallel Language Integrated Query (PLINQ) to support data parallelism. PLINQ provides the ability to parallelize data queries. Meanwhile, TPL provides parallelized versions of the C# for and foreach loops and partitions work for parallel processing and spawning, executing, and terminating threads that execute in

parallel. Intel markets its Threaded Building Blocks (TBB) which are used for parallelizing serial software. TBB is a C++ template library which treats a program's operations as tasks. The tasks are assigned to individual cores by TBB's run-time executables. The operations or tasks then execute in a sequence where synchronization bugs between the threads or tasks have been minimized or removed.

OpenMP provides a framework for parallelizing serial software written in C, C++, or Fortran using roughly fifty pre-processor declarations. In this model, one core acts as the master assigning tasks/work to the other cores. Using this framework, the developer writes a control software for the master core and complementary software for the tasks that the other cores perform. The MCF library of functions manages concurrent processes and distributes data among the cores. The biggest problem with MCF is that it only supports IBM's Cell processor.

4. Software development and debug tools

Software tools have and continue to be one of the biggest challenges for software developers working with multi-core processors. In general, the author finds most tools to be narrowly focused on just a single hardware vendor's products, a single processor, or a single programming language. These tools often provide results of limited value, or require greater manual labor than what is expected. For example, often times the output from one tool cannot simply be routed as input into another tool. These tasks may require a good deal of manual reformatting or manipulation prior to inputting the data into the next tool. Some tool vendors repackaged their multi-processor software tools with a few modifications to handle inter-core processing as tools for multi-core software development. However, the good news is a few software development and debug tools have entered the market that are mature, are focused on products from multiple vendors, and provide a good deal of automation to free up the developer for more pertinent, non-repetitive tasks. The rest of this section will discuss a few software development tools for multi-core processing. Most of the information below comes from the tool vendors themselves, tool investigations that the author has performed, or demonstrations that the author has witnessed.

Clean C overcomes some single core to multi-core conversion problems. IMEC has developed the Clean C utility as an Eclipse plug-in which automatically converts C code from a single core processor to a multi-core processor. However, Clean C has 29 programming rules that must be manually applied to the code base prior to using the utility. Once the C code base conforms to all 29 programming rules, the Clean C utility can be executed on the software with few updates for an optimized multi-core application. If the Clean C utility is applied without implementing the 29 programming rules to the C code base, the result will likely be non-operational. The Clean C utility can only be applied to C language software code bases. Clean C does not properly convert C++ based applications. The author has not tested this product.[8]

Intel's Parallel Studio is a C/C++ multi-core tool suite that integrates with Microsoft's Visual Studio 2005, 2008, and 2010 Integrated Development Environments (IDE). Parallel Studio is comprised of:

[8] http://www.imec.be/CleanC

- Intel Parallel Advisor which models an application and analyzes the best locations to implement parallelism within your application.
- Intel Parallel Composer which contains a C++ compiler, performance libraries, software thread building blocks, and parallel debugging extensions for improved performance on multi-core processors.
- Intel Parallel Inspector which automatically identifies memory and threading errors for the software developer.
- Intel Parallel Amplifier which analyzes processing hot spots, concurrency, and lock and waits with the goal of improving performance and scalability.[9]

Cilk++, Cilk, and jCilk, developed by Supertech Research Group who sold their product to Intel, assists developers with converting from single core to multi-core software systems. First, it implements its own three command standard for multi-core development. This standard allows developers to insert these commands in their existing code for spawning and synchronizing tasks, rather than restructuring their code base. Second, these products contain a number of debugging and run-time analysis tools to assist developers with optimizing their applications in a multi-core environment. Cilk++, Cilk, and jCilk apply to applications written in C++, C, and Java, respectively. Some of the Cilk components have been embedded in Intel's Parallel Studio tool. The author has witnessed a demonstration of Cilk++.[10]

The objective of Critical Blue's Prism tool is to provide analysis and an exploration and verification environment for embedded software development using multi-core architectures. A software developer could use this tool to assist in converting an application from a single core microprocessor to a multi-core microprocessor. It implements a Graphical User Interface (GUI) to assist with a developer's multi-threaded software development. The GUI provides multiple views for a look 'under the covers'. It provides detailed analysis of your application. The tool works for many processor chips including x86, PowerPC (PPC), Microprocessor without Interlocked Pipeline Stages (MIPS), and Advanced Reduced Instruction Set Computer (RISC) Machine (ARM). The author has tested this product and found it to be one of the better tools for moving an application from a single core to multi-core processor.[11]

Poly Core Software provides a multi-core software communications framework. The Poly Core software tool suite consists of:

- Poly-Mapper which is a Graphics User Interface (GUI) tool that allows developers to map software communications across multiple cores using XML commands.
- Poly-Generator converts the Poly-Mapper XML commands to C source code files.
- Poly-Messenger contains a software communications library for distributing processing on multiple cores.
- Poly-Inspector allows developers to inspect and analyze applications for communication 'hot spots'. A 'hot spot' occurs where a single or more cores have an increased amount of processing activity and while other processing cores are idle.

[9] http://en.wikipedia.org/wiki/Intel_Parallel_Studio
[10] http://supertech.csail.mit.edu/cilk
[11] http://www.criticalblue.com

The Poly Core Software suite of tools is used for generating only C code files that can be ported to multiple OSs. The code files execute on most processor chips such as x86, PPC, ARM, and MIPS.[12]

Transparent Inter Process Communication (TIPC) is an Open Source implementation that allows software designers to create applications that can communicate quickly and reliably with other applications within the computing core cluster. The TIPC protocol originated with Ericsson and has been deployed in their products for several years. TIPC is available for Linux, Solaris, and VxWorks OSs. Most applications using TIPC are written in C/C++ languages and support is available for Perl and Python.[13]

VLX by Red Bend Software is a real-time hypervisor that assists developers with migrating embedded systems from single core to multi-core processors. Their tool allows developers to run applications using a mixture of traditional Real-time Operating Systems (RTOS) along with Linux and Windows OSs concurrently on a shared hardware platform. Virtual Logix claims VLX maintains determinism and the same high performance that a RTOS provides. VLX has been certified to Common Criteria (CC) Evaluation Assurance Level (EAL) 5. VLX executes on ARM, Texas Instruments (TI), PPC, and x86 microprocessors. The author has witnessed a demonstration of VLX.[14]

Simics is primarily a virtualization emulation tool used by software developers to develop, test, and debug embedded software that will eventually execute on multi-core processors or in a simulated environment. Simics is produced by Intel's Wind River subsidiary. Simics can emulate many multi-core chip manufacturer's processors. However, Simics specializes in its support for Freescale Semiconductor processors. Simics claims to provide additional visibility into your system to improve overall debugging performance. Simics models hardware systems using software running on ordinary workstation computers for an improved development and debugging experience for software engineers. Simics allows developers greater control by varying the number and speed of the cores and injecting actual faults into the system. The author has witnessed a demonstration of Simics.[15]

QEMU is an open source virtualization emulation tool used by software developers to develop, test, and debug embedded software that will eventually execute on multi-core processors or in a simulated environment. It provides solutions for x86, ARM, MIPS, PPC, Scalable Processor Architecture (SPARC), and several other microprocessor families. A developer can simulate multiple communication channels by creating multiple QEMU instances. The author is currently working on a team using QEMU for its virtualization efforts.[16]

TRANGO Virtual Processors, a subsidiary of VMware, uses an Eclipse based IDE to provide secure virtual processes for software engineers to migrate legacy single core processors to multi-core platforms. TRANGO virtual processors assist with migration to multi-core by first instantiating multiple virtual processor units on a single core. Next, the developer populates each virtual processor unit with its own OS and application(s). Then, the

[12] http://www.polycoresoftware.com
[13] http://tipc.sourceforge.net
[14] http://www.redbend.com
[15] http://www.windriver.com/products/simics
[16] http://wiki.qeu.org/Main_Page

developers move the OSs and applications onto a physical multi-core hardware system. TRANGO recommends mapping one TRANGO hypervisor to each core. TRANGO hypervisors also support the SMP multiprocessing model and RTOSs. The author has not tested this product.[17]

Sysgo markets their PikeOS which is a paravirtualization RTOS based on a separation microkernel. It uses an Eclipse based IDE. This RTOS has been certified to DO-178B Level B. Sysgo claims that the PikeOS implements a Multiple Independent Levels of Security (MILS) architecture and that it is completing formal code verification for a CC EAL 7 certification during the Summer 2011. In 2009, Sysgo began marketing the PikeOS in North America. The author has not tested this product.[18]

Most of the major embedded RTOS suppliers including QNX, Wind River, Lynux Works, Green Hills, and DDC-I also support software development for multi-core processors. However, they do not offer identical support. Most of the suppliers also provide their own hypervisor that works with their own line of products. The author has tested Wind River's VxWorks OS with multi-core support and has witnessed demonstrations of QNX and Lynux Works RTOSs with multi-core support. When analyzing the RTOS's multi-core support, pay attention to the product's performance profiling tools which allow the developer to examine more closely what is happening 'under the hood'. Understand which software languages each RTOS supports and whether real-time support is provided for each language. Wind River, Green Hills, and Lynux Works also market their own real-time hypervisors.

This section has discussed several tools for developing software aimed at a multi-core processor. Very few tools are direct competitors with another tool. Currently most tools are attempting to solve one small piece of the software developer's task in writing software for a multi-core environment. When choosing software development tools for multi-core processors, keep in mind that most tools are still immature, are usually programming language specific, processor specific, and/or vendor specific. Make sure you have a thorough understanding of the application you are developing or migrating and the development needs are for the application. Ask very detailed, pointed questions of the tool vendors to make sure you understand what their tool can and cannot perform at the time of purchase or use.

5. Virtualization

Virtualization technology can be used to create several virtual machines to run on a single virtual machine. Virtualization technology allows multiple OSs to run on a single processor. Processors with multiple cores could easily simulate one virtual machine on each core of the physical processor or machine. Virtualization technology was first introduced in the 1960s with IBM mainframe computers with many benefits. First, virtualization allowed many users to concurrently use the same mainframe platform where each user had their own virtual machine and where each virtual machine can execute a different OS resulting in increased productivity from the expensive IBM mainframes. Second, the technology allowed legacy applications to run on future mainframe hardware designs. Third, the virtualization

[17] http://en.wikipedia.org/wiki/Trango_Virtual_Processors
[18] http://www.sysgo.com/products/pikeos-rtos-technology

technology all resided in a thin layer of software that executed on top of the hardware or an underlying OS.

With the introduction of the Personal Computer (PC), interest in virtualization technology died. However, with PCs and processors becoming more and more powerful in the last ten years, there is a resurgence in the technology for this computing equipment. The benefits of virtualization achieved with mainframe computers are the same for single core and multi-core processors. Virtualization products can be found for both real-time and non-real time embedded systems.

There are three main types of virtualization as shown in Figure 7. They are:

- Full Virtualization, which is the most flexible approach, can usually support any OS type. Most processor manufacturers have been adding full virtualization support for their processors. This approach allows any legacy or new OS designed for the processor to run virtualized. This approach can be implemented with a host OS executing between the hypervisor and the hardware, but it is not necessary. This approach can also be implemented with special virtualization technology built into the processor. Since this approach does not require any modifications to the OS, it is expected to eventually be the preferred virtualization type.
- Para Virtualization which can only support OSs that have been modified to run in their virtual machine. In this approach the OS is modified so that it would use the virtualized layer of software's interface to communicate between the guest OSs and the virtualized layer of software. Para virtualization is usually built into the host OS and then allows multiple guest OSs to execute in virtual machines. This approach executes faster at run-time than the full virtualization approach.
- Container Virtualization can only support OSs that have been modified to run in their virtual machine like a Para Virtualization approach, but here there is no attempt to virtualize the entire processor. Instead most of the OS components are reused between the container based OSs. Container virtualization implements a host OS and guest OSs for sharing the host code with one restriction. The guest OSs must be the same as the host OS.

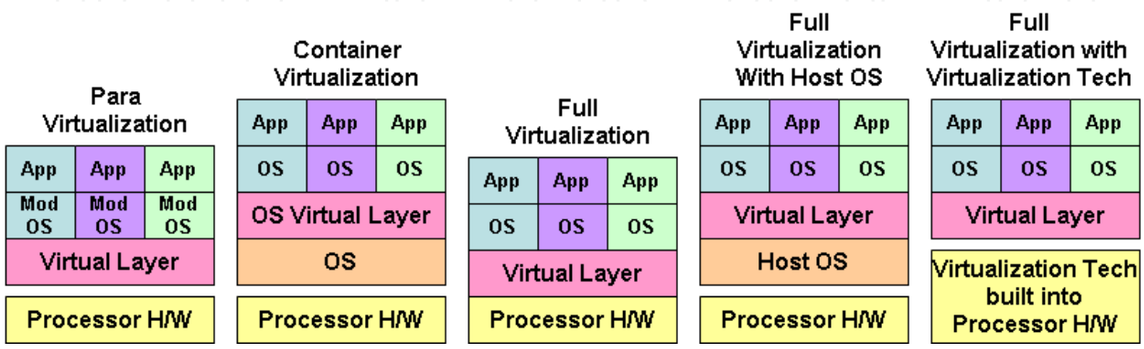


Fig. 7. Virtualization Technology Types.

Virtualization technology led to the development of Multiple Levels of Security (MLS). An MLS embedded system is a trusted system at a high robustness level that securely handles processing data at more than one classification level. An MLS system is similar to virtualization technology whereby a processor is divided into several virtual machines or partitions. The difference is that in a MLS system the partitions are based on security levels. For example, one partition may be unclassified, a second partition may be Secret, while a third partition may be classified Top Secret.

6. Software programming languages

Software language support for multi-core processors generally falls into two categories. New languages designed with parallelism from the beginning or extensions to current popular software languages. Most language extensions are focused on single Fortran and C/C++ standards. Some language extensions include:

- OpenMP Fortran which is an extension to Fortran 95. Basically, it implements OpenMP compiler directives, library functions, and environment variables for the Fortran language.[19]
- Co-array Fortran which is an extension to Fortran 95 and the 2008 Fortran standards. Co-array Fortran syntax is architecture independent and can be used in shared memory and distributed memory machines and on clustered machines. Co-array Fortran can be applied to a greater range of machine architectures than OpenMP Fortran, hence a Subset Co-Array Fortran has been generated which can be translated into OpenMP as part of the compilation process.[20]
- High Performance Fortran (HPF) is an extension to Fortran 90. HPF uses a data parallel model of computation to spread the work of a single array computation over multiple processors. Many users and vendors who initially used HPF have migrated to OpenMP Fortran or Co-array Fortran.[21]
- OpenMP C/C++ contains compiler directives, library functions, and environment variables that assist developers with managing parallel programs coded in the C/C++ languages. The directives extend the sequential C/C++ programming languages with parallel constructs.[22]
- Parallel Unified C, also known as Unified Parallel C, is an extension of the C programming language designed for computing on large-scale parallel machines. Parallel Unified C extends ISO C 99 with the following constructs:
  - An explicitly parallel execution model
  - A shared address space
  - Synchronization primitives and a memory consistency model
  - Memory management primitives[23]
- pC++ is a language extension to C++ that contains parallel constructs for C++ applications on high performance computers. pC++ allows programmers to develop distributed data structures with parallel execution semantics.[24]

[19] http://en.wikipedia.org/wiki/OpenMP
[20] http://www.co-array.org
[21] http://hpff.rice.edu
[22] http://en.wikipedia.org/wiki/OpenMP
[23] http://upc.lbl.gov

Some new languages designed with parallelism from the start include:

- Erlang is a concurrent/functional programming language with dynamic typing and strict evaluation. It supports hot swapping so code can be modified without stopping a system. It is used primarily in the telecom industry.[25]
- Fortress is an open source language that is being targeted for the multi-core and supercomputing software communities. The current Fortress prototype runs on top of a standard Java Virtual Machine (JVM). Fortress supports both task and data parallelism. The runtime implicitly farms out computations to the available processor cores using a fine-grained threading model. Basically, the designers implemented parallelism into the language at every possible location. Sun's Fortress language was originally funded by the Defense Advanced Research Projects Agency (DARPA) High Productivity Computing System (HPCS) program.[26]
- Z-level Programming Language (ZPL) is a portable, high-performance parallel programming language for science and engineering computations. It is an array programming language that uses implicit parallelism and can execute on both sequential and parallel computers.[27]
- Chapel is an open source language that is expected to support a multi-threaded parallel programming model. It is expected to support data parallelism, task parallelism, and nested parallelism. Chapel is expected to support object-oriented concepts, generic programming features, and code reuse. This language is being developed by Cray, Inc. Some Chapel concepts come from HPF and ZPL. Cray's Chapel language was originally funded by DARPA's HPCS program.[28]
- Haskell is a purely functional programming language that engineers from Galois are embracing that is richly statically typed. Functional programming languages lack side effects. These languages handle structures as values. Functional languages reduce code count. Functional programming languages like Haskell require a paradigm shift from both object oriented and modular programming languages. Parallel evaluation strategies and nested data parallelism are built into the language.[29]

Most of the above languages have been developed within the past six years. Erlang is the exception to this.

7. Multi-core processing standards

One of the goals of the Multi-Core Association has been developing standards for multi-core processors. The Multi-core Association is an industry consortium whose members include embedded software and hardware companies such as Intel, Freescale Semiconductor, Nokia Siemens Networks, QNX, Texas Instruments, and Wind River Systems. The Multi-core Association's goal is to support the multi-core ecosystem which includes vendors of development tools, debuggers, processors, operating systems, compilers, and simulators

[24] http://www.extreme.indiana.edu/sage
[25] http://www.erlang.org
[26] http://en.wikipedia.org/wiki/Fortress_(programming_language)
[27] http://en.wikipedia.org/wiki/ZPL_(programming_language)
[28] http://chapel.cray.com
[29] http://www.haskell.org/haskellwiki/Haskell

along with application and system developers. The Multi-core Association has either completed, started work, or has plans to develop the following standards:

- The Multi-core Communications Application Programmer Interface (MCAPI) is a high-performance, low latency communications and synchronization Application Programmer Interface (API) for closely distributed cores and processors in embedded systems. MCAPI is expected to support streaming communications that are fast and efficient and are similar to the sockets used for networking applications. The MCAPI is expected to support "socket like" stream-based API which would benefit multi-core devices. The MCAPI has the goal to support just the specific needs of embedded systems such as tighter memory constraints, high system throughput, and tighter task execution time constraints.
- Multi-core Resource Management API (MRAPI) provides a standardized API for the management, synchronization, and scheduling of processing resources. The MRAPI will support features for state management, context management, scheduling, and basic resource synchronization. The RAPI has the goal to support existing operating systems and the CAPI, Multicore Task Management API (MTAPI), and Debug API.
- Multi-core Programming Practices (MPP) provides a "best practices" guide for C/C++ developers to write "multi-core ready" software. The goals for this standard is to assist software developers in developing portable multi-core code which can be targeted at multiple platforms, reducing bugs due to multi-core related issues, and reduce the learning curve for multi-core software development.
- Multi-core Virtualization will provide users of embedded virtualization solutions with improved interoperability of applications and middleware between different virtualization vendors through the properties in its standard.
- MTAPI will provide a standardized API for dynamic scheduling and managing software tasks, including task creation and deletion for a large variety of architectures. The MTAPI goal is to support existing operating systems and the MCAPI, MRAPI, and Debug API.
- Debug API will enhance multi-core development systems with development tools to address problems in communication and interpretation of debug tools and on-chip debug components. This work includes:
  - Identifying and mapping multi-core debugging high level requirements to specific requirements for underlying infrastructures
  - Extending and standardizing current debug interfaces for multi-core debugging needs
  - Standardizing debugging and Joint Test Action Group (JTAG) interface connections.

The purpose of these APIs is to make the source code portable and reusable so that software multi-core architectures can be processor independent. The expectation is that the standards should complement one another. See Figure 8.

So far, the MCAPI and MRAPI standard APIs have been released. The MPP standard is expected to be released in later 2011 or early 2012. The scheduled release dates for the Multi-core Virtualization, and Debug standards have all passed without the standards being released. These standards are developed by the Multi-core Association's member organizations. Most of these organizations are companies with their own deadlines for

shipping software and hardware tools and products to market. Hence, their main priorities are satisfying their customers with their products and services. So the Multi-core Association's processing standards development is progressing at a slower rate than originally anticipated.

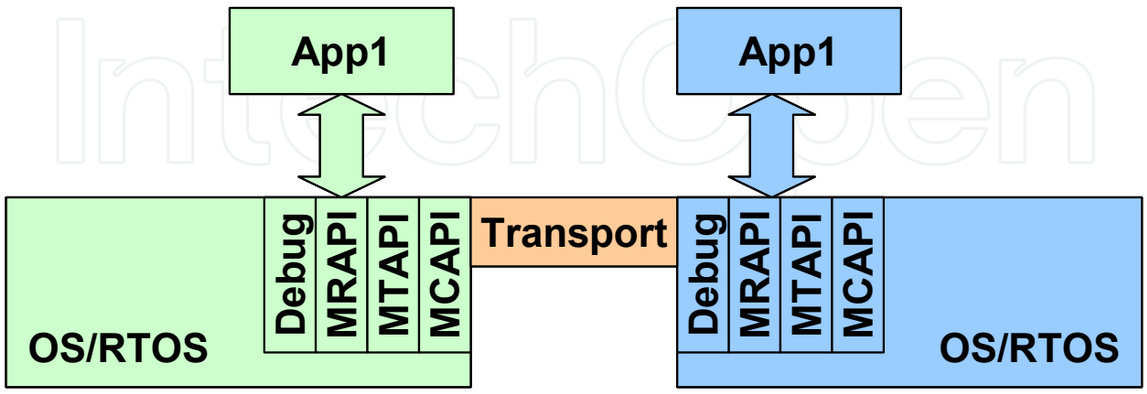


Fig. 8. MCAPI, MRAPI, MTAPI, and Debug Implementation View for Multi-core Devices.

OpenMP is a specification for a set of compiler directives, Runtime Library Routines, and environment variables that can be used to specify multithreaded, shared memory parallelism in Fortran and C/C++ programs. The OpenMP specification is being developed by the OpenMP Architecture Review Board (ARB). The OpenMP Version 3.0 Specification has been released to the public and addresses many multi-core processor needs. OpenMP is a portable, scalable model that provides shared memory parallel programmers a flexible API for developing parallel applications for multiple platforms. OpenMP contains a set of compiler directives and library routines for parallel application programmers. Typically OpenMP is used to parallelize looping constructs among multiple threads. OpenMP has the following advantages[30]:

- Provides both coarse-grained and fine-grained parallelism.
- When updating a serial application to run in a multi-core parallel environment, the original code set will most likely not require to be modified when parallelized with OpenMP pragma compiler directives.
- When executing a parallelized application in a serial environment, the OpenMP directives can be treated as comments.
- Data decomposition and layout are handled automatically by pragma directives.

OpenMP has the following disadvantages:

- Cannot be used on Graphics Processing Units (GPU).
- Scalability is limited. Easier to work with on small software applications of less than 1000 lines than large applications with several hundreds of thousands of lines of code.
- Can introduce synchronization bugs and race conditions without providing any assistance in removing these bugs.

[30] http://en.wikipedia.org/wiki/OpenMP

- Requires a compiler that supports OpenMP.
- Possible to accidently write false sharing code. False sharing occurs when multiple threads on different cores write to a shared cache line but not at the same location. Since the memory is changing, each core must update its copy of its cache resulting in much greater memory transfers than in a serial application with a single thread.

Other standards that are focused on issues pertaining to multi-core processors include:

- Mobile Industry Processor Interface (MIPI) is addressing a range of debug interface efforts for multi-core devices. However, its specifications are focused on mobile devices and not multi-core processors in general.[31]
- Message Passing Interface (MPI) is an API specification that allows multiple computers to communicate with each other. It is often used for parallel programs running on computer clusters and supercomputers, where accessing non-local memory can be expensive.[32]
- System C is a standard that allows engineers to design a system that spans both hardware and software. It contains a set of C++ classes and macros. It is often used for system simulations, modeling, and functional verification involving parallel processes. Multiple software suppliers support the System C standard.[33]

8. Software community parallel programming experience

The vast majority of software developers are experienced in serial software development. Few software engineers are experienced in parallel software development. First, training for software engineers has traditionally been focused on serial development efforts. Very few universities and colleges offer undergraduate courses aimed at parallel software development. The author has sponsored a short parallel and multi-core programming course at Boeing. One Boeing engineer with a PhD in Computer Science from a major university remarked that he planned to take the course since they had one course in parallel software development at his university and that he just did not understand the concepts. If PhDs from major universities are having problems with parallel software development, clearly software engineers with Bachelor degrees will also have problems. Second, the author and several Boeing teammates have investigated universities and colleges throughout the United States for course offerings in parallel software development. Unfortunately, we did not find many universities nor colleges offering any courses. Some of the better educational opportunities that were investigated include the University of Illinois at Urbana-Champaign, University California-Berkeley, Stanford University, MIT, and Washington University at St. Louis. Both the University of Illinois at Urbana-Champaign and University California-Berkeley offer Summer school courses in parallel software development. The author has found many recommendations for both universities' courses. Stanford University is offering training through its Pervasive Parallelism Lab. Also, the MIT professors at Supertech Research Group who developed the Cilk applications have been offering classes on parallelism topics that use the Cilk tool. There are some professional

[31] http://www.mipi.org
[32] http://www.mcs.anl.gov/research/projects/mpi
[33] http://www.systemc.org/home

training organizations such as ProTech which will provide training in parallel and multi-core software development. In conclusion, the availability for training in parallel software development has been and continues to be very slim.

One of the major challenges in migrating serial software to a parallel environment is ensuring that your system's functionality is still correct after spreading the functionality across several cores all executing simultaneously. In parallelizing your application there are several concurrency issues that a software developer needs to watch for:

- Dead lock: Occurs when two or more threads or processes are both waiting for the other to release a resource.
- Live lock: Similar to dead lock where tasks require a resource held by another thread or process, but the state of the waiting threads or processes are changing in regards to other tasks. A live lock example are when the Three Stooges are each trying to get through a doorway and they get stuck.
- False Sharing: Occurs when two or more cores access different data in a shared cache line. If one core writes to the cache line, the caching protocol may force the second core to reload the cache line even though the data has not changed.

A second major challenge for software developers is to analyze their software for data dependencies based on the execution of threads for the entire system. A data dependency occurs when two data references read from or write to the same variable, whether it is stored in memory or in a register. If a data dependency is detected, the software developer shall either reorder the statements or modify the thread execution on different cores. Look at the statements below which are executed in order from instruction 1 to instruction 5 and determine where the dependencies exist:

1. variable1 = 3;
2. variable2 = 5;
3. variable3 = variable2;
4. variable4 = variable1 + variable2;
5. variable1 = -8;

There are data dependencies between instructions 2 and 3 and between instructions 4 and 5. This means that if you switch instructions 2 and 3 and instructions 4 and 5, respectively, the application results will be different. If a software developer switches instructions 1 and 2, the application results will be the same. Hence, a data dependency does not exist between instructions 1 and 2. There are several data dependency types. First are true dependencies which exist when an instruction is dependent on the previous instruction, such as:

1. variable1 = 2;
2. variable2 = variable1;

True dependencies occur where a variable is defined in one statement and then used in the following statement. This is also known as "Write after Read" and these statements are not safe to be reordered. Second are anti-dependencies which exist when an instruction requires a value that is later updated, such as:

1. variable1 = variable2;
2. variable2 = 5.0;

Anti-dependencies occur where a variable is read prior to the variable later being reset to a different value. This is also known as "Read after Write" and these statements are not safe to be reordered. Third are input dependencies which exist where a variable is read prior to being read a second time, such as:

1. variable1 = variable2;
2. variable3 = variable2;

Input dependencies occur where a variable is read twice in a row. This is also known as "Read after Read" and these statements are safe to be reordered. Fourth are output dependencies also known as false dependencies. These dependencies exist where a variable is written to prior to being written to a second time, such as:

1. variable1 = 0.0;
2. variable1 = 3.0;

Output dependencies occur where a variable is written twice in a row. This is also known as "Write after Write" and these statements are not safe to be reordered. Fifth are control dependencies which exist when the output of an instruction was referenced in a previous decision block. An example of this is displayed below where variable2 is set in statement 3, but was referenced in a decision block in statement 1:

1. if (variable1 == variable2)
2. variable1 = variable1 + variable2;
3. variable2 = variable2 + variable1;

A control dependency does not exist between instructions 1 and 3. However, a control dependency may exist between instructions 1 and 2 if instruction 1 can be executed prior to instruction 2 or if the output of instruction 1 determines if instruction 2 will be executed. The control dependency displayed may exhibit "Write after Read" and instructions 1 and 2 may not be safe to reorder.[34]

With the challenges listed above, there are several solutions that software developers can use.

- First, there are software locks that can be placed around code that may lead to deadlock, live lock, or data dependency conditions. A software developer would place the lock start prior to a block of problematic code and the lock end after the block of problematic code. See Figure 9 where the synchronized command is used to place locks around the moveBox and updateBox functions. In using software locks, software developers can use them during writes to memory or a register or during reads from memory or registers that may have been updated. Software locks should not be used when invoking methods on other objects. The advantage of software locks is that it increases safety by guaranteeing that only the block of problematic code is functioning with other threads or processes are halted. The disadvantage of software locks is that all other threads and processes are halted during this execution, thus slowing the system execution to a serial environment.
- A second solution is to make your code immutable. A software developer accomplishes this by replacing public class variables and global variables with private class variables

[34] http://en.wikipedia.org/wiki/Data_dependency

and local variables and passing class variables into functions via the function call. The advantage of designing immutable software is that it eliminates data dependencies while increasing their parallel execution of your software. The disadvantage is that it may require significant modifications to your software.

- A third option is software confinement. Here the software developer confines all processing to execute on a single thread or single core. When this practice is followed, your resulting software is more loosely coupled and is a good software architecture strategy. The advantage of software confinement is that it eliminates data dependencies while increasing their parallel execution of your software. The disadvantage is that it may require significant modifications if your software was originally designed without confinement as a goal. This may also include ordering your system's code blocks so that any code with potential data dependency problems execute at different times on different threads or cores.

```
Thread on core 1:
Void synchronized moveBox(){
   x += getUpdatedX();
   . . . .
   y += getUpdateY();
   . . . .
}

        Thread on core 2:
        Void synchronized updateBox(int x, int y){
           y_ = y;
           x_ = x;
        }
```

Fig. 9. Lock Example.

- A fourth option is decomposing large blocks of code where data dependencies and dead locks take place into smaller blocks and place the locks around the smaller blocks of software. The advantage of this approach is that the software will be safer without spending significant time re-architecting your system. The disadvantage of this approach still involves halting all other threads and processes while the problematic code is still functioning.

As we saw in this section there are a number of issues such as dead locks, live locks, and data dependency situations that may cause applications to ineffectively run when they are parallelized. The good news is that there are several options for software developers to implement to correct these problems. While some options can be quickly implemented like software locks, they degrade overall system performance, while other options like software confinement and immutable software improve software performance, they can take many developer hours to correctly implement.

9. Differentiated multi-core processors

On the positive side, the differentiated multi-core processors have provided greater options for software developers. In the past, a large system would consist of several processors with different single core processing units. Some of the processors would have GPUs for display processing while other processors would have CPUs to perform the actual non-display processing. Now, multi-core processors are coming into vogue with multiple CPU cores and multiple GPU cores for both non-display and display processing, respectively. Hence, with

these multiple heterogeneous and homogeneous hardware multi-core processors, developers have greater and better choices for developing new large scale software systems.

The increase in differentiated multi-core processors has its share of problems on the software side. As we have seen with several frameworks and tools mentioned earlier, often times there is only support for certain processors or processor families. Of course, the microprocessor vendor is attempting to tie the software development to their own multi-core processor which can cause several problems. First, while most multi-core microprocessor vendors have developed some software tools, no vendor has developed a complete suite of tools to assist the software developer with requirements, architecture, code, and test. Second, with the hardware vendors entering this market, the software tool vendors' market share is reduced. They may decide against providing a new tool or supporting a particular multi-core microprocessor's chipset if the vendor themselves is already providing the support. Third, with so many software developers not trained nor experienced with developing parallel software, the addition of many differentiated multi-core processors increases the learning curve for developers. The software developer may be working on different multi-core processors at the same time. Hence, in this case the role of differentiated multi-core processors has probably slowed, rather than enhanced, their adoption by the computing industry.

10. Conclusion

By reviewing some of the key software development issues for multi-core processors, including:

- Immaturity of software tools
- Lack of standards
- Inexperience of current software developers
- Lack of software models and frameworks
- Lack of System software like libraries
- Differentiated processors with minimal support

Current software development for multi-core processors is at an immature level when compared to both software development for single core processors and hardware development for multi-core processors. Therefore, this chapter has provided details to support Chuck Moore's statement that "To make effective use of Multi-core hardware today, you need a PhD in computer science."[35] Even though the statement is a few years old, it still applies as of the writing of this chapter. There is still much research to be performed for improved parallel processing models and frameworks. Both Microsoft and Intel have spent millions in this research along with several small startup companies. The biggest question continues to be how to identify promising solutions along with attracting the research dollars to fund the work to develop the solutions. More attention needs to be paid towards standards development which should naturally improve over time. The biggest concern is the education and training of software professionals. Currently, some 'best practices' documents are being developed for beginner multi-core software developers. The biggest challenge is for the universities, colleges, and other training organizations to educate new and experienced software developers. While analyzing the improvements over the past several years, many more breakthroughs are still needed before the software industry can receive the full benefit from upgrading to multi-core processors.

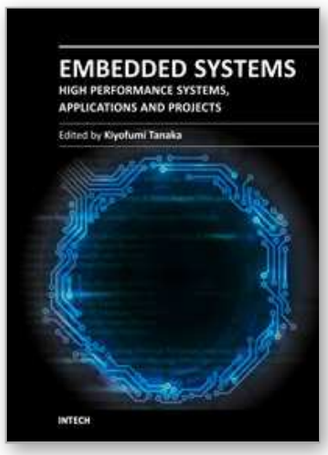
[35] Moore, Chuck, (May 12, 2008) "Solving the Multi-core Programming Problem", *Dr. Dobbs Journal*.

11. Acknowledgments

Kenn Luecke would like to thank Andrea Egan, Shawn Rahmani, and Wayne Mitchell for assisting him with his initial research into software development for multi-core processors for the Boeing Company. He would also like to thank fellow Boeing engineers David Cacchia, Tom Dickens, Jon Hotra , David Sharp, Don Turner, Homa Ziai-Cook, and Heidi Ziegler for their assistance with multi-core related issues.

12. References

[1] http://en.wikipedia.org/wiki/Moore's_law
[2] Sutter, H., (March, 2005). "The free lunch is over. A fundamental turn toward concurrency in software," *Dr. Dobb's Journal, Volume 30, Number 3*.
[3] Fittes, Dale, (October 30, 2009) Using Multicore Processors in Embedded Systems – Part 1, *EE Times*.
[4] Moore, Chuck, (May 12, 2008) "Solving the Multi-core Programming Problem", *Dr. Dobbs Journal*.
[5] Morgan, Lisa L., (December 15, 2006), Making the Move to Multicore, *SD Times*.
[6] Morgan, Lisa L., (December 15, 2006), Making the Move to Multicore, *SD Times*.
[7] Wlad, Joe. (2011), Freescale Multi-Core Forum, Freescale Multi-core Forum, St. Louis, September, 2011.
[8] http://www.imec.be/CleanC
[9] http://en.wikipedia.org/wiki/Intel_Parallel_Studio
[10] http://supertech.csail.mit.edu/cilk
[11] http://www.criticalblue.com
[12] http://www.polycoresoftware.com
[13] http://tipc.sourceforge.net
[14] http://www.redbend.com
[15] http://www.windriver.com/products/simics
[16] http://wiki.qeu.org/Main_Page
[17] http://en.wikipedia.org/wiki/Trano_Virtual_Processors
[18] http://www.sysgo.com/products/pikeos-rtos-technology
[19] http://en.wikipedia.org/wiki/OpenMP
[20] http://www.co-array.org
[21] http://hpff.rice.edu
[22] http://en.wikipedia.org/wiki/OpenMP
[23] http://upc.lbl.gov
[24] http://www.extreme.indiana.edu/sage
[25] http://www.erlang.org
[26] http://en.wikipedia.org/wiki/Fortress_(programming_language)
[27] http://en.wikipedia.org/wiki/ZPL_(programming_language)
[28] http://chapel.cray.com
[29] http://www.haskell.org/haskellwiki/Haskell
[30] http://en.wikipedia.org/wiki/OpenMP
[31] http://www.mipi.org
[32] http://www.mcs.anl.gov/research/projects/mpi
[33] http://www.systemc.org/home
[34] http://en.wikipedia.org/wiki/Data_dependency
[35] Moore, Chuck, (May 12, 2008) "Solving the Multi-core Programming Problem", *Dr. Dobbs Journal*.



Embedded Systems - High Performance Systems, Applications and Projects

Edited by Dr. Kiyofumi Tanaka

ISBN 978-953-51-0350-9
Hard cover, 278 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

Nowadays, embedded systems - computer systems that are embedded in various kinds of devices and play an important role of specific control functions, have permeated various scenes of industry. Therefore, we can hardly discuss our life or society from now onwards without referring to embedded systems. For wide-ranging embedded systems to continue their growth, a number of high-quality fundamental and applied researches are indispensable. This book contains 13 excellent chapters and addresses a wide spectrum of research topics of embedded systems, including parallel computing, communication architecture, application-specific systems, and embedded systems projects. Embedded systems can be made only after fusing miscellaneous technologies together. Various technologies condensed in this book as well as in the complementary book "Embedded Systems - Theory and Design Methodology", will be helpful to researchers and engineers around the world.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen