



Leveraging Multi-core Processors Through Parallel Programming

Executive Summary

Swift and inexorable advances in hardware have historically challenged software developers to maximize system capabilities and meet users' soaring expectations.

This is especially true in microprocessor technology, where the state of the art has quickly moved from dual-, tri-, quad-, hexa-, octo-core chips to units with tens or even hundreds of cores. The good news is that processors are going to continue to become more powerful. The flip side is that, at least in the short term, growth will come mostly in directions that do not take most current applications along for their customary free ride.

This white paper offers an outlook for parallel software development and focuses in particular on application development for shared memory multi-core systems using an Ateji® PX¹ open source preprocessor for Java.²

Ateji® PX provides a smooth transition path from today's sequential programming languages to future parallel languages, likely to be exceptionally different and require new thinking and techniques for designing programs. Code written in Ateji® PX is both compatible with today's languages and ready for tomorrow's hardware.

Why Parallel Computing?

In 1965, Intel co-founder Gordon Moore observed that the number of transistors available to semiconductor manufacturers would double approximately every 18 to 24 months. Today's new and faster multi-core and chip multi-threading processor designs are making that level of scalability more easily attainable and affordable.

Parallel Thinking

Flynn's taxonomy⁴ is a specific classification of parallel computer architectures that is based on the number of concurrent instruction (single or multiple) and data streams (single or multiple) available in the architecture (see Figure 1).

The first dimension is the number of instruction streams that particular computer architecture may be capable of processing at a single point in time. The second dimension is the number of data streams that can be processed at a single point in time. In this way, any given computing system can be described in terms of how instructions and data are processed.

What This Means for Developers

The traditional approach to programming (sequential programming) will not efficiently take advantage of multi-core systems. Sequential programming proved beneficial when computers

Flynn's Rules

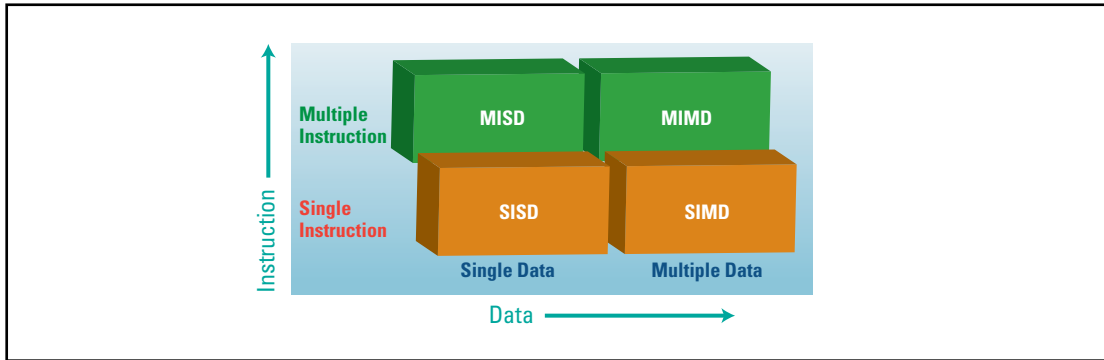


Figure 1

Architectural Comparison

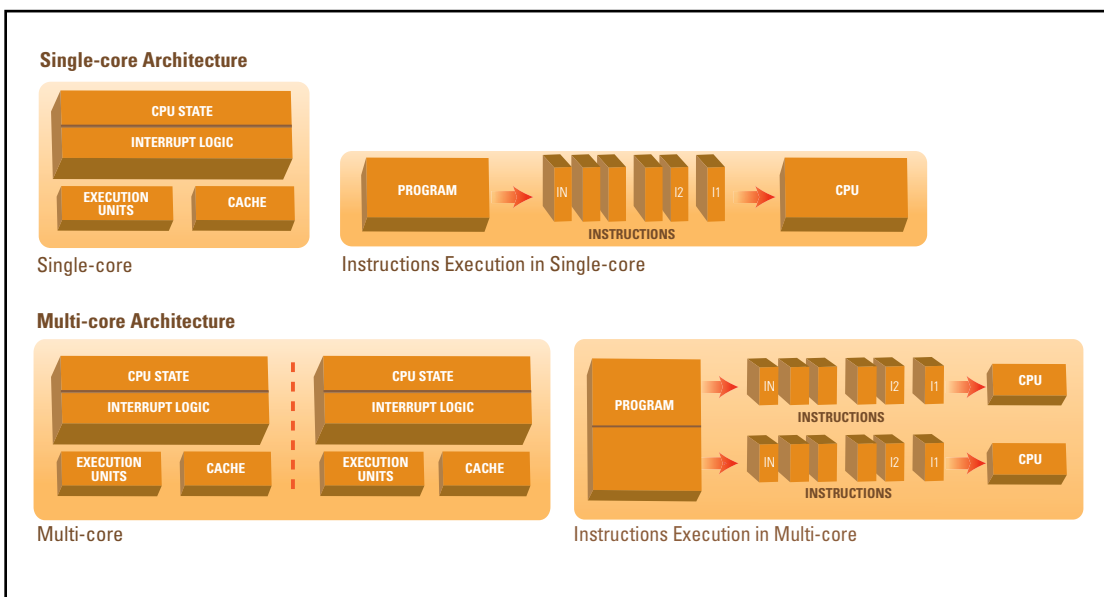


Figure 2

had single-core architectures, but with multi-core systems the approach is inefficient. To fully exploit the capability of multi-core machines, developers need to redesign applications so that each microprocessor can treat code instructions as multiple threads of execution. One way to resolve this is by utilizing parallel programming.

What Is Parallel Programming?

Parallel Programming⁵ is a form of computation in which program instructions are divided among multiple processors (cores, computers) in combination to solve a single problem, thus running

a program in less time. The single-core and multi-core architectures, along with the instructions executions, are highlighted above.

Designing and developing parallel programs has typically been a very manual process. The programmer is usually responsible for both identifying and actually implementing parallelism.

Parallel programming presents many pitfalls – race conditions are the most common and difficult multi-threaded programming problem. Other potential issues include mutual exclusion and deadlock. Overhead due to thread synchro-

Multi-core Assessment Sheet

Software Details																
Eclipse SDK	Helios (Version: 3.6.2)															
Java Virtual Machine	JVM version 1.6 or higher															
Ateji PX	Ateji PX™ (Version: 1.2.0.201106241352)															
SQL Server 2005	Microsoft SQL Server Enterprise Edition (Version: 9.00.5000.00)															
Hardware Details																
Type	Desktop		Laptop													
Operating System	Windows XP Professional SP3		Windows 7 Professional													
CPU Name	Intel Core 2 Duo E6550		Intel Core i5 520M													
Specification	Intel® Core™ 2 Duo CPU E6550 @2.33GHz		Intel® Core™ i5 CPU M520 @2.40GHz													
Cores	2		2													
Threads	2		4													
Core Speed	1990 - 2400 MHz (approx.)		1197 - 1330 MHz (approx.)													
Cache	L1 Data(2X32KBytes) 8 -Way, L1 Inst(2X32KBytes) 8 -Way, Level2 (4096 KBytes) 16 -Way		L1 Data(2X32KBytes) 8 -Way, L1 Inst(2X32KBytes) 4 -Way, Level2 (2X256 KBytes) 8 -Way, Level2 (3 MBytes) 12 -Way													
Memory	Type: DDR2 Channel #: Dual Size: 2048 Mbytes DC Mode: Symmetric		Type: DDR3, Channels #: Dual Size : 4096 MBytes, DC Mode: Symmetric													
Inference																
Type	Desktop		Laptop													
Latency (in Sec)	Sequential	Parallel	Sequential	Parallel												
Test Run1	358.31	214.12	675.13	208.69												
Test Run2	333.13	222.63	657.12	212.99												
Test Run3	336.97	215.99	685.61	205.07												
Average	342.8	217.58	672.62	208.91												
Sequential Vs. Parallel Program	<table border="1"> <caption>Desktop Latency Comparison</caption> <thead> <tr> <th>Program Type</th> <th>Average Latency (Sec)</th> </tr> </thead> <tbody> <tr> <td>Sequential</td> <td>342.80</td> </tr> <tr> <td>Parallel</td> <td>217.58</td> </tr> </tbody> </table>		Program Type	Average Latency (Sec)	Sequential	342.80	Parallel	217.58	<table border="1"> <caption>Laptop Latency Comparison</caption> <thead> <tr> <th>Program Type</th> <th>Average Latency (Sec)</th> </tr> </thead> <tbody> <tr> <td>Sequential</td> <td>672.62</td> </tr> <tr> <td>Parallel</td> <td>208.91</td> </tr> </tbody> </table>		Program Type	Average Latency (Sec)	Sequential	672.62	Parallel	208.91
Program Type	Average Latency (Sec)															
Sequential	342.80															
Parallel	217.58															
Program Type	Average Latency (Sec)															
Sequential	672.62															
Parallel	208.91															
Percentage of Improvement	36.52%		68.87%													

Figure 3

nization and load balancing can severely impact run-time performance and can be very hard to fix. Thus, very often manually developing parallel codes is a time-consuming, complex, error-prone and iterative process.⁶

How to Improve Time to Market

There is a well-known ancient Chinese saying: "A craftsman first sharpens his tools before laboring on his work." This is very true of

software development, particularly in the area of complex parallel applications.

For many years, various tools have been available to assist programmers in converting serial programs into parallel programs, thereby overcoming key parallel programming pitfalls such as race conditions, deadlock, etc. Ateji® PX is among the more promising tools used to parallelize a serial program.

Why Ateji® PX for Parallel Programming?

Ateji® PX for Java™ introduces parallelism at the language level, extending the sequential base language with a small number of parallel primitives. It is compatible at the source level and byte-code level with all Java™ and JVM-based programs and libraries. This makes parallel programming simple and intuitive, easy to learn, efficient, provably correct and compatible with existing code; it is suitable for parallelizing legacy applications and leaves more time to concentrate on performance improvements. We assayed the Ateji® PX for Java™ preprocessor against multi-core machines. What follows are our key findings.

We successfully incorporated the Ateji® PX for a CPU-intensive application program used by a client to generate approximately 4000 MSDS (Material Safety Data Sheets) from RTF to PDF format for various countries (see Figure 3).

The Business Benefits

- **Increased Company Value:** Improved performance and maximized profits. Well-designed parallel programs result in greater value for the business as well as for the customers.
- **Improved Productivity and Insight:** Improved application efficiency and productivity; thus, improved TTM (time to market).
- **Reduce Costs and Increased Flexibility:** Parallel programming provides the greatest flexibility possible since the application is

tailored to customer specifications and to the way customers do business. Importantly, parallel programs can be more easily altered as business changes require. We have implemented a solution for a large multinational manufacturing conglomerate, which powered them to accommodate multiple business processes efficiently.

- **Higher Return on Investment:** Implementing parallel programs would, over time, reduce development cycles and improve CPU utilization by making use of all the available cores in servers, thus enhancing the capacity to house more applications on specific machines.

Conclusion

The processing speed that can be obtained using parallel programming depends on the number of processors available, and also on the size of the problem and the way in which it can be broken into constituent parts. A program organized according to independence and divide-and-conquer principles is more easily run either in parallel or sequentially, according to resources available.

Given that multi-core processors/parallel computing and most computing platforms available support multiple instructions, multiple data (MIMD), it makes sense to leverage parallel programming to its fullest on shared-memory machines, massively parallel super computers, clusters, and even across a utility computing grid.

Footnotes

- ¹ <http://www.ateji.com/px/1.0/javadoc/>
- ² <http://www.oracle.com/us/technologies/java/index.html>
- ³ "Ten Laws Of The Modern World" http://www.forbes.com/2005/04/19/cz_rk_0419karlgaard.html
- ⁴ <http://www.phy.ornl.gov/csep/ca/node11.html>
- ⁵ http://en.wikipedia.org/wiki/Parallel_computing
- ⁶ "The Problems with Threads", Edward A. Lee, Professor, Chair of EE, Associate Chair of EECS, EECS Department, University of California at Berkeley, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.pdf>

Resources

http://en.wikipedia.org/wiki/Intel_Core

<http://www.vogella.de/articles/JavaConcurrency/article.html#concurrency>

About the Authors

Giri Muthusamy is a Cognizant Associate focused on Java/J2EE projects. His interests revolve around the design of enterprise applications and RDBMS concepts. Giri holds a bachelor's degree in Mechanical Engineering and is also a Microsoft SQL Server certified professional. He can be reached at Giri.Muthusamy@cognizant.com.

Rahul Ramalingam is a Cognizant Technology Specialist focused on Java/J2EE high-level system design and application development and building enterprise software applications. He has an engineering degree in computer science and holds various software certifications including IBM Rational solution designer – OOAD and TOGAF 8 Certified Enterprise Architecture Practitioner. Rahul can be reached at Rahul.Ramalingam@cognizant.com.

About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 118,000 employees as of June 30, 2011, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.



World Headquarters

500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277
Email: inquiry@cognizant.com

European Headquarters

Haymarket House
28-29 Haymarket
London SW1Y 4SP UK
Phone: +44 (0) 20 7321 4888
Fax: +44 (0) 20 7321 4890
Email: infouk@cognizant.com

India Operations Headquarters

#5/535, Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060
Email: inquiryindia@cognizant.com