



Multithreaded Application  
Acceleration with  
**CHIP MULTITHREADING (CMT),  
MULTICORE/MULTITHREAD  
ULTRASPARC® PROCESSORS**

UltraSPARC Processor Applications  
White Paper  
August 2008

## Table of Contents

<b>Introduction</b> .....	<b>3</b>
Traditional single-threaded architectures have hit a wall .....	3
Multicore processors can still have memory latency issues .....	3
<b>Chip multithreading (CMT) — a breakthrough solution</b> .....	<b>4</b>
CMT enables more efficient use of microprocessor resources .....	4
<b>Boosting telco infrastructure performance with no added cost</b> .....	<b>6</b>
<b>Virtualization at the chip level</b> .....	<b>7</b>
Economy by simplification .....	7
Flexibility through threading .....	7
<b>Near-linear crypto acceleration</b> .....	<b>8</b>
Master and helper threads — microparallelism .....	8
<b>CMT accelerates string searching</b> .....	<b>10</b>
Fast, cost-effective virus protection .....	11
Eliminating the need for TCAMs .....	11
<b>Conclusion</b> .....	<b>12</b>
For more information .....	12
Disclosures .....	12

## Chapter 1

# Introduction

The proliferation of billions of devices — ranging from 3G cell phones to netbook computers — means consumers increasingly demand instantaneous access to information, whether the latest news story or a sports video clip. To meet this burgeoning demand, the processors that power modern networks must deliver huge improvements in performance and throughput.

## Traditional single-threaded architectures have hit a wall

Standard methods of increasing real-world application performance, such as higher clock speeds and pipeline branch prediction, have been yielding diminishing returns for the last few years. Higher-frequency processors waste a lot of power and generate so much heat that they need expensive HVAC systems to properly cool the systems in which they run.

In addition, a critical gating factor is memory latency/speed of data access. Memory speed has grown at a much slower rate than processor speeds, because memory suppliers have focused on increasing density and lowering cost.

## Multicore processors can still have memory latency issues

An early stage in the evolution of microprocessor design has been to group two or four conventional processor cores on a single physical die, creating a multicore processor. However, most current offerings simply replicate cores from existing single-threaded processor designs. This approach typically yields only slight improvements in aggregate performance, since it ignores key performance bottlenecks such as memory speed and hardware thread context switching. While multiple programs can be accommodated in parallel, the principal problem of cache misses and consequent stalling is not addressed.

## Chapter 2

## Chip multithreading (CMT) — a breakthrough solution

Sun Microsystems was the first company to recognize that the speed of data access from memory was the critical bottleneck, and has overcome this problem with the chip multithreading (CMT) architecture that is the basis of the UltraSPARC® T1 and T2 processors.

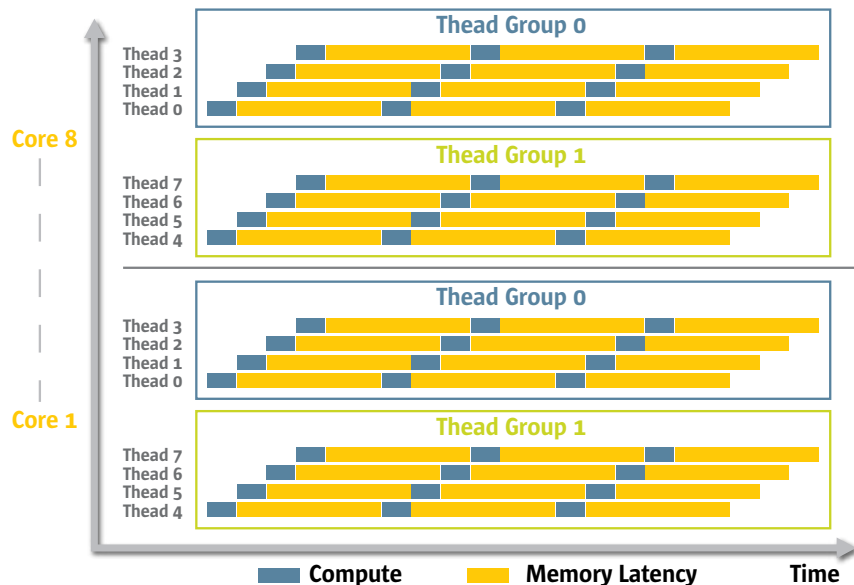


Figure 1. The UltraSPARC T2 processor uses two instruction pipelines per core, with four threads per pipeline, for a total of eight threads per core, and up to 64 threads per processor.

Using Sun's CMT architecture, applications are divided into active threads, and each processor core is designed to switch between up to eight threads on each clock cycle (UltraSPARC T2 processor). Even if a particular thread stalls while waiting for data to be available from memory, the core can switch immediately to another thread and the pipeline remains continuously active, doing useful work. The processor's entire execution pipeline is thus optimized to execute active threads as much as possible, rather than be held up by any particular thread waiting for data to be available from memory. The negative effect of memory latency is therefore masked and minimized.

### CMT enables more efficient use of microprocessor resources

The old prevailing philosophy of instruction-level parallelism (ILP) required complicated tactics, such as deep pipelines, large caches, speculative prefetches, and out-of-order execution. This resulted in complex, hot, and power-hungry processors.

By contrast, thread-level parallelism (TLP) enables the use of much simpler pipelines that focus on scaling with threads rather than frequency. These simpler pipelines can process a large number of simultaneous threads, rather than running a single thread as quickly as possible. This approach is more congruent with the profile of present-day applications, which typically need to address high simultaneous user or transaction counts. An added advantage of the high pipeline utilization is that an efficient processor consumes less energy and consequently runs much cooler.

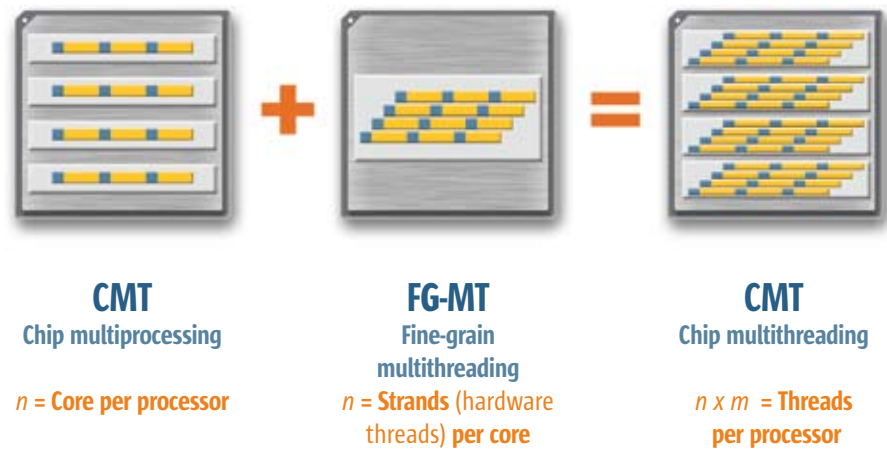


Figure 2. Chip multithreading combines chip multiprocessing and fine-grained hardware multithreading.

Fine-grained multithreading works in tandem with chip multiprocessing to result in a multiplier effect on application performance without a corresponding need for increased frequency and resultant heat generation. Since multiple instructions can execute at every clock cycle, the improvement in throughput and execution speed can be very high.

### Chapter 3

## Boosting telco infrastructure performance with no added cost

A good example is provided in a set of benchmarks run recently at Continuous Computing, a global provider of integrated services that helps telecommunication equipment manufacturers deploy next-generation networks. Among the company's key offerings is the Trillium software suite, a line of more than 60 standards-based telecommunication protocols. A key product among these is the Session Initiation Protocol (SIP), which forms the backbone for voice communications.

In May 2008, Continuous Computing ran the multicore Trillium SIP on a server using an eight-core UltraSPARC T2 multithreaded processor and achieved a breakthrough 6,000 calls per second, more than twice as many as any other processor that they had previously tested. Most importantly, the application effectively utilized all 64 available threads to scale performance via higher CPU utilization, versus the addition of more CPUs. This equates to a company needing less than half its previous hardware footprint to serve the same number of customers, thereby reducing infrastructure and power costs while improving density.

<b>Calls per Second (CPS)</b>	<b>Messages per Second (sent plus received)</b>	<b>TCP CPU Utilization (mpstat)</b>	<b>UDP CPU Utilization (mpstat)</b>
500	6,000	6	6
1,000	12,000	11	12
1,500	18,000	17	18
2,000	24,000	23	24
2,500	30,000	29	30
3,000	36,000	35	36
3,500	42,000	42	42
4,000	48,000	49	48
4,500	54,000	56	55
5,000	60,000	63	62
5,500	66,000	70	69
6,000	72,000	78	76

*Figure 3. Continuous Computing's benchmark of the Trillium SIP protocol on an UltraSPARC T2 processor-based Sun Netra™ T5220 server showed near-linear scalability as more threads were used, and processor utilization increased.<sup>1</sup>*

When an application is run as multiple threads as in the case of the Trillium SIP protocol discussed above, the effect is to scale almost linearly because of progressively higher utilization of the processor through more threads. Better performance is achieved through greater utilization efficiency of a single processor, as opposed to the cost and complexity of adding multiple processors.

1. Source: <http://www.ccpu.com/papers/multicoresip/>.

## Chapter 4

# Virtualization at the chip level

The model underlying the UltraSPARC T1 and T2 processor architectures is that of fine-grained interleaved threading, where a new thread (or process) is launched on each clock cycle. Where previously there was only one running thread monopolizing the processor, now multiple threads can share its apparatus and features. The effect of multiple threads is thus to virtualize the processor, where the processor essentially can be considered as a group of individual, bootable machines, each with memory and I/O carved out.

## Economy by simplification

A positive secondary outcome is that what was formerly a deep process pipeline that had to do branch prediction in hardware to be efficient is now replaced by very short pipelines where the effect of a branch miss is resolved in two clock ticks. The savings on prediction-oriented hardware enable the use of fewer transistors and storage, and thus less power consumption.

In current network devices, there might be separate chipsets designated for the control unit, for routing, and for media. This requires the knowledge of three different instruction sets — that of a microprocessor, a specialized network processor, and a digital signal processing chip — each made by a different vendor. Chip multi-threading enables individual threads in the same processor to be devoted to each of these functions, so the knowledge requirement for engineers can be reduced to one chip architecture.

## Flexibility through threading

The ability to devote specific threads to certain functions makes the processor very flexible. For instance, one could designate two threads to processing, two to the network interface, and the rest to important features such as firewall and encryption. CMT especially shines in improving the implementation of some of these latter security functions.

## Chapter 5

# Near-linear crypto acceleration

In this age of wireless and mobile computing, encryption of packets across networks is becoming increasingly important. It is critical for privacy of VPNs, secure VoIP, and cellular 3G networks. Encryption also is fundamental for the protection of intellectual property in media gateways, and for securing file systems and databases. The traditional challenge is that it adds a lot of computational overhead, and results in a throttling of packet throughput. An example of an important encryption tool for 3G mobile telephony is the KASUMI algorithm.<sup>2</sup>

Current software implementations of this algorithm on traditional processors use multiple lookup tables sized to all fit in their small Level 1 caches. This use of multiple small tables requires a significant number of arithmetic operations in order to access the various tables, manipulate the data, and recombine the results from them.

For an optimized CMT implementation, one can combine many small lookup tables into a few large ones that can instead reside in Level 2 cache. Lookup tables are constant and can be precomputed. The net effect is that the number of instructions to access the data from the lookup table can be greatly reduced. Normally the trade-off is that this would result in a greater number of memory stalls; but this plays to the strength of CMT, because a memory stall simply results in the firing off of a new thread. It consumes much less of a processor core's resources while it is stalled.

As a result, as the number of threads is increased, performance scales almost linearly. For the UltraSPARC T2 processor, per-core KASUMI performance is around eight times the performance of a single thread, and per-chip KASUMI performance is close to 64 times a single-strand performance.

## Master and helper threads — microparallelism

Another optimization technique that is well suited to CMT processors is microparallelism,<sup>3</sup> or the division of small chunks of work between multiple threads. These helper threads are assigned to helping master threads in the task of rapidly processing performance-limiting serial components. It is the key to eliminating single-threaded performance limitations.

2. KASUMI is a block cipher that forms the heart of the 3GPP confidentiality algorithm f8, and the 3GPP integrity algorithm f9.

3. "Multicore Processors and Microparallelism," by Lawrence Spracklen, 3 April 2008, <http://www.opensparc.net/publications/presentations/multicore-expo-2008-multicore-processors-and-microparallelism.html>.

In a CMT processor, the inter-thread latency is very low because the threads communicate through Level-2 cache. The time this takes is on the order of tens of clock ticks, as opposed to the hundreds of clock ticks that is characteristic of “bare-metal” latency, when communicating across processors. Furthermore, problems such as hot lock are overcome.

Microparallelism means that one can think of adding threading to operations that have traditionally been considered single-threaded, such as traversing a linked list, B-Copy, and string location. By judicious use of helper threads, applications can be greatly sped up by the use of CMT across a wide range of applications in areas such as genomics, high-performance computing, and Web searches.

## Chapter 6

# CMT accelerates string searching

One of the core functions needed for text identification algorithms in data repositories is real-time string searching. A popular procedure used in this space is the Aho-Corasick algorithm.<sup>4</sup> Recently, Sun compared the results of an Aho-Corasick string search using an UltraSPARC T2 processor-based system (Sun SPARC® Enterprise T5220 server) to the published results<sup>5</sup> of a similar search using an IBM Cell Broadband Engine (Cell/BE) DD3 Blade.<sup>6</sup>

To test the performance of their processor, IBM used a 4.4 MB variant of the King James Version of the Bible using a dictionary of the 20,000 most used words in the English language (average word length of 7.59 characters). To approximate the dictionary and Bible IBM used, Sun used a dictionary of 25,144 English words<sup>7</sup> (average word length: 8.22 characters) and a 4.6 MB variant of the King James Version of the Bible.<sup>8</sup>

The results of the test are summarized in the following table:

System	Processor	Throughput (Gb/sec)	Number of Processors	Number of Cores	Frequency (GHz)
Sun SPARC Enterprise T5220 server	UltraSPARC T2 processor	24.6	1	8	1.4
IBM Cell Broadband Engine DD3 Blade	IBM Cell Broadband Engine	3.8	2	16	3.2

*Figure 4. In a test of string searching, the UltraSPARC T2 processor outperformed the IBM Cell Broadband Engine in throughput and efficiency.*

Comparing the systems, the throughput of the single-processor, UltraSPARC T2-based Sun SPARC Enterprise T5220 server was 6.5 times greater than the IBM system, which used twice as many processors and cores, running at over twice the clock speed.

This also demonstrates the huge advantage that is obtained from rapid access to the large, shared Level 2 cache in the UltraSPARC T2 processor.

4. The Aho-Corasick algorithm is a string-searching algorithm. A kind of dictionary-matching algorithm, it locates elements of a finite set of strings (the "dictionary") within an input text.

5. See: IEEE Computer, Volume 41, Number 4, pp. 42–50, April 2008.

6. See "Disclosures" for details.

7. The OpenSolaris™ file [cvs.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/cmd/spell/list](http://cvs.opensolaris.org/source/xref/onnv/onnv-gate/usr/src/cmd/spell/list).

8. [www.patriot.net/users/bmcgin/kjv12.zip](http://www.patriot.net/users/bmcgin/kjv12.zip).

In addition to its throughput superiority, the UltraSPARC T2's multicore/multithreaded architecture was much easier to code for than the IBM Cell Broadband Engine. IBM's article cites numerous, elaborate optimizations that were required to achieve its results. By contrast, Sun implemented the Aho-Corasick algorithm using ANSI C and a simple compilation — no special optimizations of the algorithm were required to achieve the reported performance. So the UltraSPARC T2 processor not only delivered superior throughput, but did so while saving significant time and effort in software development.

### Fast, cost-effective virus protection

A practical application of this algorithm is in screening for computer viruses. Each virus has a certain signature — the “words” by which it can be identified. Virus recognition requires deep packet inspection, that is, a detailed examination of every byte of every packet received. So a real-time method of screening for viruses is necessary.

CMT offers a simple solution. One could build a dictionary starting with the particular sequences of bytes that characterize a virus, and add to the dictionary all such words for all relevant viruses. The dictionary can then be searched using the text of incoming packets for the presence of these words.

### Eliminating the need for TCAMs

Typically, virus signatures are stored in ternary content addressable memory (TCAM). This is a specialized type of memory that is designed for searching in a single lookup operation but which is very expensive to add on to a network device. By being able to scan and identify viruses in real time in software itself, a designer using the UltraSPARC T2 processor can avoid the additional cost and complexity of both the TCAM interface and the additional memory.

## Chapter 7

# Conclusion

As the preceding benchmarks show, the combination of multithreaded software running on a multicore/multithreaded, CMT processor results in a huge increase in application performance, with no additional hardware. The ability to designate the functions of different threads on the fly enables tremendous flexibility and customizability, as well as radical levels of consolidation. The increase in performance afforded by better processor utilization — rather than the addition of more processors — also reduces time to market, by simplifying design and development, and it enables smaller form factors. CMT technology-based devices can draw less power, while at the same time being compact and powerful enough to handle the most complex real-time networking tasks. Sun's CMT, multicore/multithreaded processor architecture is the best option for system designers building embedded network infrastructure.

*For additional information on how UltraSPARC processors  
can help with application throughput and more efficient utilization  
of compute resources, go to [sun.com/microelectronics](http://sun.com/microelectronics).*

### Disclosures

Pattern matching benchmark: Sun SPARC Enterprise T5220 server (1x 1.4 GHz UltraSPARC T2 processor, one chip, eight cores); the Solaris™ 10 Operating System (OS); Sun C 5.9, 21.7 GB/sec

IBM Cell Broadband Engine (Cell/BE) DD3 Blade (2x 3.2 GHz Cell Broadband Engine, two chips, 16 cores); Linux kernel v2.6.16; IBM CBE Software Development Kit v2.1, 3.8 GB/sec

IBM results were obtained from: Figure 7(d) of IEEE Computer, Volume 41, Number 4, pp. 42–50, April 2008. Sun benchmark results as of 8/01/2008.