

On the Use of General-Purpose Multi-Core Processors in Networking Devices

Patrick Crowley and Jon Turner
 {pcrowley|jon.turner}@wustl.edu
 Department of Computer Science & Engineering
 Washington University in St. Louis

With the arrival of multi-core general-purpose processor products in the last few years, and with the promise of more cores per chip in future products, it is natural to wonder how well these processors might work within programmable networking equipment, such as routers. After all, network processors (NPs) are also multi-core processors, augmented with networking-specific instructions, hardware-assists, and memories. While these specialized NP features might improve performance, they come at the cost of reduced generality and familiarity. There are good reasons to want general-purpose processors in networking equipment, such as improved programmer productivity, a larger installed base of software and developers, and an expectation of better application portability to future systems. Many equipment vendors would be happy to trade reasonable amounts of performance or efficiency to realize some of these benefits. (Although those vendors interested in maintaining closed, proprietary systems will likely never use, or admit to using, general-purpose processors in order to limit the threat of third-party development.) To illustrate some of the issues that influence the selection of a processor for use in a networking context, we can compare some recent general-purpose multi-core processors to a selection of network processors, namely Cisco's Silicon Packet Processor [EA05], Intel's IXP 2855 [AD02, II07], and Cavium's CN3860 and CN5860 [HU06,CA07a, CA07b]. We will consider some first-order characteristics of the processors themselves, such as available CPU cycles, memory and I/O bandwidth, and power efficiency, as well as more qualitative issues such as software productivity and performance evaluation.

Table 1. Characteristics of multi-core network and general-purpose processors.

Processor	Clock (MHz)	CPU Power (W)	Chipset Power (W)	Packet I/O (Gb/s)	Mem I/O (Gb/s)	Processor Cores	Core Issue Width	Peak BIPS	Peak BIPS/W
Cisco SPP	250	35	0	192	175	188	1	47	1.34
Intel IXP2855	1500	27	0	25	121.6	16	1	24	0.89
Cavium Octeon CN5860	1000	40	0	25	102.4	16	2	32	0.80
Cavium Octeon CN3860	600	30	0	25	102.4	16	2	19.2	0.64
Intel Quad-Core Xeon 5300, Intel 5000P chipset	2330	80	30	0	85.6	4	4	37.28	0.34
AMD Turion 64 X2 Dual-Core Mobile TL-56	1800	33	0	51.2	85.6	2	3	10.8	0.33
Intel Mobile Core 2 Duo, Intel 965e chipset	2400	35	28	0	68	2	4	19.2	0.30
Intel Dual-Core Xeon 5138, Intel 5000P chipset	2130	35	30	0	85.6	2	4	17.04	0.26
AMD Dual-Core Opteron 1218 HE	2600	65	0	192	85.6	2	3	15.6	0.24
Intel Dual-Core Xeon 7120, Intel E8501 Chipset	3000	96	32	0	51.2	2	4	24	0.19

Ten processors and a summary of their characteristics are shown in Table 1. In addition to the two network processors, we include three Intel Xeon processors, one Intel Mobile Core 2 Duo processor [IX07c], one AMD dual-

core Opteron 1218 HE [AM07a, CH03], and one AMD Turion dual-core mobile TL-56 [AM07b]. The Xeon 5000 series [IX07a, IX07c] is marketed for embedded processing applications; the Xeon 7000 family [IX07b] is intended for use in server and workstation platforms. The Intel general-purpose processors do not have on-die memory controllers, so the memory bridge component of the typical platform chipset is included in the description above, mostly for the purpose of accounting for power consumption.

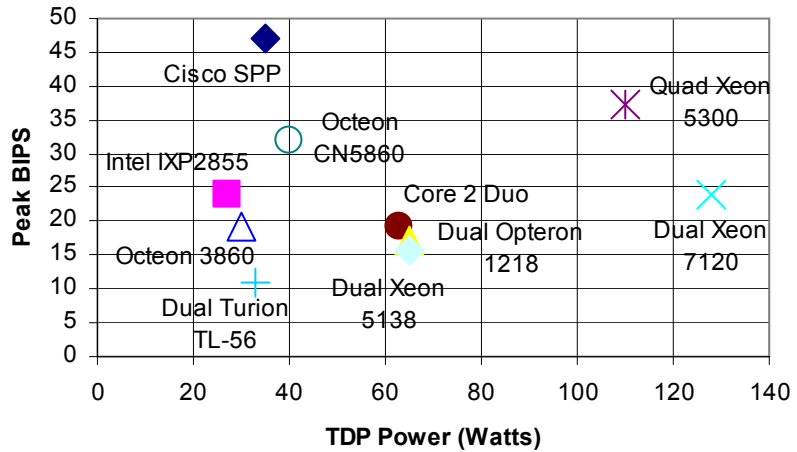


Figure 1. Peak instruction throughput, in billions of instructions per second (BIPS), versus thermal design power (TDP).

Peak instruction throughput, in units of billions of instructions per second (BIPS), can be calculated by taking the product of clock rate, processor core count, and processor core instruction issue/completion width. In comparison, the NPs have lower clock frequencies, more processor cores, and unit issue widths. In fact, these represent the traditional differences in processor organization between NPs and general-purpose processors. These general-purpose processors have 3- or 4-way multiple issue pipelines, thus when a program exhibits sufficient instruction-level parallelism, a processor core can execute three or four instructions from that program in the same cycle.

Two high-level observations can be made. First, there is no substantial difference in peak instruction throughput between these two groups of processors. On the high-end, for example, the SPP provides nearly 50 BIPS and the quad-core Xeon CPU features nearly 40 BIPS. The second observation concerns the fraction of the peak throughput that can be achieved in practice. The comparatively simple, single-issue pipelines in the Intel and Cisco NPs allow developers to structure their code in order to improve pipeline utilization, thus near-peak performance is achievable at the cost of programmer optimization. Multiple-issue pipelines either support concurrent instruction issue under limited circumstances, as is the case with the Cavium dual-issue cores, or include a number of sophisticated microarchitectural features and mechanisms, such as branch prediction, speculative execution, and out-of-order execution, which can improve performance for unstructured code by reordering instructions dynamically but render most programmer code optimization techniques ineffective. These pipelines maintain an instruction window of future instructions—obtained by looking forward into program execution by predicting likely control flow across branch and basic block boundaries—from which batches of data-independent instructions can be drawn for concurrent execution. Common wisdom, based mostly on sustained instruction completion rates seen in general-purpose programs, would predict throughput at or near 1 instruction per cycle for arbitrary code with an issue width of 4 (that is, a 25% sustained utilization). Generally speaking, we would expect narrower pipelines to have better utilization than wider ones. This has an impact on performance evaluation and optimization, as we will discuss shortly.

The consequences of high clock rates can be seen clearly by examining power consumption. Faster clock rates translate directly into increased power consumption, as can be seen in Table 1. Each of the general-purpose processors has at least double the power consumption seen in the NPs. When expressed in terms of peak BIPS per watt, the Cisco and Intel NPs can be seen to have significantly greater efficiency, 4x and 2.6x resp., as compared to the best general-purpose processor. The ratio between instruction throughput and thermal design power (i.e., the power consumption that a cooling solution must be designed for) is illustrated in Figure 1.

Another important difference can be seen in the overall amount of memory bandwidth available. The NPs each have in excess of 100 Gb/s of memory bandwidth, compared to a maximum of 86 Gb/s among the general-purpose processors. In a networking context, this corresponds to fewer external memory transactions available per packet. Both NPs and general-purpose processors make extensive use of on-chip memories to increase bandwidth; it can be noted that the general-purpose processors rely much more heavily on multi-level cache hierarchies to reduce memory latency and increase effective memory bandwidth. Indeed, data structures and algorithms can be designed with caches in mind, but the cache organizations are not part of the programmer’s interface—the instruction set

architecture—and are therefore free to vary between processors (although this has not happened often in the past). This, too, has an impact on performance evaluation and optimization.

Software productivity—including programmer productivity, existing software and developer bases, and software portability—is a great advantage of platforms based on general-purpose processors. In networking equipment, the majority of a system’s intellectual property, competitive advantage, and overall complexity is found in the software rather than the hardware. Thus, creating and maintaining that software efficiently is a primary concern. In low-performance network-edge applications, where line rates are below 1 Gb/s, vendors already provide solutions based on general-purpose platforms and open-source software [VY07]. In high-performance contexts, some aspects of software productivity might be difficult to realize. For example, it is difficult to implement high-performance, I/O-centric processing in user-level processes in traditional operating systems due to the performance overheads involved. Consequently, it is customary to implement such functionality in kernel-resident modules or, in some cases, device driver code. Kernel and device driver software development is low-level in nature, and very often requires substantial modifications across software and hardware versions.

Networking devices very often have performance requirements—e.g., to implement function X on minimum size packets at a specific line rate—which in turn place performance requirements on the software implementing the functionality. Simple processor cores, such as those found in the Intel and Cisco NPs, make it possible for programmers to understand the performance of their code and how it might vary at run time; this is true because most of the factors and mechanisms that determine performance are under programmer control. Sophisticated processors, with aggressive pipeline features and multi-level cache hierarchies, do not allow software developers to observe or control many of the performance-related features of the processor core. This strategy works well in server and workstation application contexts where little is known in advance about the execution context and where no strict performance requirements apply.

One particularly useful illustration of programmer control can be found in the asynchronous, non-blocking nature of load/store operations in the Intel IXP. In the IXP’s instruction set, memory operations have associated hardware signals which are asserted when an I/O operation has completed. These signals can be used to make control flow and thread scheduling decisions in subsequent instructions (i.e., they can serve as inputs to a class of branch instructions). For example, after issuing an SRAM load, a program can be structured: to block further execution until the load completes (which is the usual blocking operation found in general-purpose instruction sets), to continue executing other useful code until the requested load value is required for further program progress, or to put the current thread to sleep and allow another to execute until the hardware signal is asserted. These non-blocking memory operations allow programmers to precisely control the utilization of the processor’s execution resources, even in the presence of variable and lengthy memory operations. In general-purpose instruction sets, there is no way to structure a program to react dynamically to the performance of the memory system. With careful and explicit data and instruction layout, it may be possible to keep a sufficient number of useful instructions executing in the presence of a long-latency memory operation (i.e., to keep the processor core’s instruction window filled with instructions that can be executed while the memory operation completes), but any such optimizations rely on processor characteristics that are not part of the instruction set architecture and can be expected to differ between processor families and generations.

In conclusion, general-purpose multi-core processors offer substantially lower memory bandwidth and power efficiency, as compared to current NPs. While the current generation of multi-core general-purpose processors may be a poor fit for high-speed networking, there are good reasons to presume that future processors and chipsets will offer improved memory bandwidth and efficiency. However, due to their primary use in workstation and server applications, it is unlikely that future x86-based processor cores will be simplified or reorganized to enable greater programmer control of low-level processor mechanisms. Attempts to optimize program performance by exploiting non-architectural features, such as with cache-conscious data structures and carefully crafted instruction layout, are feasible but tend to negate much of the software productivity that motivate the use of general-purpose processors.

REFERENCES

- [AD02] Adiletta, M., et al. “The Next Generation of Intel IXP Network Processors,” Intel Technology Journal, vol. 6, no 3, pp. 6-18, Aug 2002.
- [AM07a] AMD, Inc., AMD Opteron Processor Product Data Sheet, http://www.amd.com/us-en/Processors/TechnicalResources/0,,30_182_739_9003,00.html

- [AM07b] AMD, Inc., AMD Turion 64 X2 Dual-Core Mobile TL-56 Product Data Sheet, http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_13909_13913,00.html.
- [CA07a] Cavium Networks. OCTEON Plus CN58XX Multi-Core MIPS64 Based SoC Processors. http://www.cavium.com/OCTEON-Plus_CN58XX.html
- [CA07b] Cavium Networks. OCTEON CN38XX/CN36XX Multi-Core MIPS64 Based SoC Processors, http://www.cavium.com/OCTEON_CN38XX_CN36XX.html
- [CH03] Chetana N. Keltcher, Kevin J. McGrath, Ardsheer Ahmed, Pat Conway, "The AMD Opteron Processor for Multiprocessor Servers," IEEE Micro, vol. 23, no. 2, pp. 66-76, Mar/Apr, 2003.
- [EA05] Eatherton, Will. "The Push of Network Processing to the Top of the Pyramid," ANCS 2005 keynote address, Oct 27, 2005. <http://www.cesr.ncsu.edu/ancs/slides/eathertonKeynote.pdf>.
- [HU06] Hussain, Muhammad. "Multi-Core Processors for Networking and Communication Equipment," ANCS 2006 keynote address, Dec 4, 2006. <http://www.cse.wustl.edu/ANCS/2006/OCTEON%20presentation%20for%20ANCS%202006.pdf>
- [IC07] Intel Corp., Mobile Intel Core 2 Duo and 965e chipset data sheets, <http://www.intel.com/products/processor/core2duo>, <http://www.intel.com/products/chipsets/p965>.
- [II07] Intel Corp., Intel IXP2855 Network Processor Product Brief. <http://www.intel.com/design/network/prodbrf/309430.htm>.
- [IX07a] Intel Corp., Intel Dual-Core Xeon 5138 and Intel 5000P chipset data sheets. <http://www.intel.com/design/intarch/dualcorexeon/5100/>, <http://www.intel.com/products/chipsets/5000p>.
- [IX07b] Intel Corp., Intel Dual-Core Xeon 7120 and Intel E8501 Chipset data sheets, <http://www.intel.com/products/processor/xeon7000>, <http://www.intel.com/products/chipsets/e8501/>.
- [IX07c] Intel Corp., Intel Quad-Core Xeon 5300 and Intel 5000P chipset data sheets, <http://www.intel.com/design/intarch/quadcorexeon/5300>, <http://www.intel.com/products/chipsets/5000p>.
- [VY07] Vyatta Systems. <http://www.vyatta.com/>.