

A case for Application Level Acceleration

Introduction

There has been a lot of activity in the application level acceleration area lately. The concept of an application level accelerator is now gaining credence. For example, while talking specifically about XML and accelerators, Paul Horn, director of IBM Research, said recently (2006):

'What's going to happen now is that the general purpose ones (processors) aren't going to accelerate so fast, and the technologies and tools and software for integrating the accelerators have gotten better. So you're going to see a lot more special purpose stuff that you can just plug in simply.'

Intel's recent opening up of their front side bus (IDF spring'07) and Xilinx announcement of their acceleration module indicates their view of application acceleration.

'The Xilinx Accelerated Computing Platform (ACP) targets Intel Front Side Bus (FSB)-FPGA accelerator modules and features an FSB-capable VirtexTM-5 FPGA module as a plug-in to an Intel® Xeon® CPU socket.'

At first glance, why bother about application accelerators? Just throw more standard servers at any compute intensive problem. Questions arising include, what areas/problems (simulation, XML processing) might be suitable (what size niche?) and how much of a performance gain would justify using an accelerator.

Historical Precedent points to General Requirements

Standard processors solve most people's compute problems most of the time so, as a general rule, buy standard Intel servers (or AMD, or Sun...). A decade ago, the Dajeil team were looking at the performance bottlenecks in SSL processing. At the time, standard servers were managing of order 10 transactions/second at 100% capacity. The kernel of the problem was the public-key processing algorithm (RSA) that involved processing of large (1024/2048 bit) numbers - not suited to a 32/64 bit architecture. Our solution was to design an accelerator comprising a 2048 bit wide ALU which implemented the modular exponentiation function (and not much else), with 10 instances of this and a scheduler, slotted into an ASIC, thus delivering 10,000 transactions/second. Three orders of magnitude performance boost. This example illustrates how effective, special hardware can be in delivering application specific solutions. The tricky part was to make sure the standard libraries (Open SSL, Windows crypto API etc.) didn't mind. Today, (nearly) all servers/appliances that deal with any volume of SSL traffic use hardware accelerators.

Any reasonably sized niche where special processing is required is now a valid target for application level acceleration. If greater than an order of magnitude performance boost is achievable (without this Moore's law applied to standard CPUs will quickly outstrip any advantage) and the accelerated functions can be integrated with standard libraries, (special software, special API's etc. are a no, no.) then application acceleration is appropriate.

Web Services/SOA Applications - A Candidate for Acceleration.

So considering the Web Services/SOA area. These applications use XML and SOAP messages to communicate business information. The concentration of these messages

across the enterprise network and server infrastructure is increasingly exposing performance issues in the processing of XML and related security functions.

Tasks such as XML document transformation, parsing, message validation, classification, security, and routing are inherently CPU-intensive and place a significant load on ICT infrastructure. All of this places an increasingly unacceptable load on the general-purpose processors being used for these tasks.

XML is inefficient for a number of reasons, including the following:

- XML messages are typically 20 times larger than the equivalent messages coded in binary formats.
- Systems must receive, decrypt, validate, parse, transform, process, serialize, canonicalize, sign, encrypt, and transmit each message.

It was clear from the start that many of the tasks (such as XPath processing) were suited to implementation using highly parallel hardware architectures. As in most situations, knowing what we want to achieve is a fundamental ingredient to success! Instrumenting the application to discover and quantify bottlenecks is a good starting point.

Processing bottlenecks in a Web Services Gateway

An XML intensive application tested under load yielded the table of results below. This table gives an indication of the processing overhead for identified XML operations:

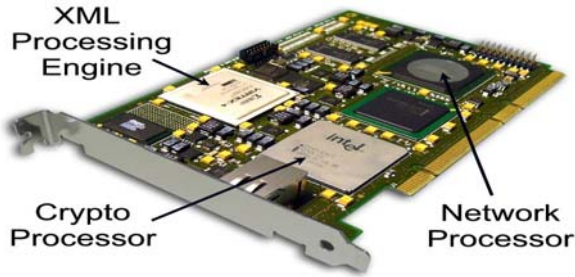
No	Activity	Throughput Level	Increase in Latency
1	Message received and sent with no processing	100%	0%
2	1 + Convert XML message to DOM format	59%	112%
3	2 + Convert DOM format to XML message	46%	178%
4	2 + Execute 2 XPath searches	22%	590%
5	4 + Signature Verification	11%	1,306%
6	3 + Sign message with 2048-bit key	8%	1,200%

The bottlenecks identified from the tests above and many other are:

- Parsing – Conversion from XML to internal software representation (DOM)
- Re-serialisation – Conversion from internal format (DOM) to XML
- XPath Searches – Extracting specific information from the XML message
- Digital signature processing – Generation and Validation
- Schema validation – Adherence of message to specific rule-set
- Transformations – Conversion of message format or content.
- Analysis of many Web Services/SOA products has yielded similar results.

A Web Services Accelerator

Dajeil has produced an accelerator to address the primary bottlenecks identified above. The solution has been delivered in PCI-X short card form factor. The main functional blocks are indicated in the photograph below.



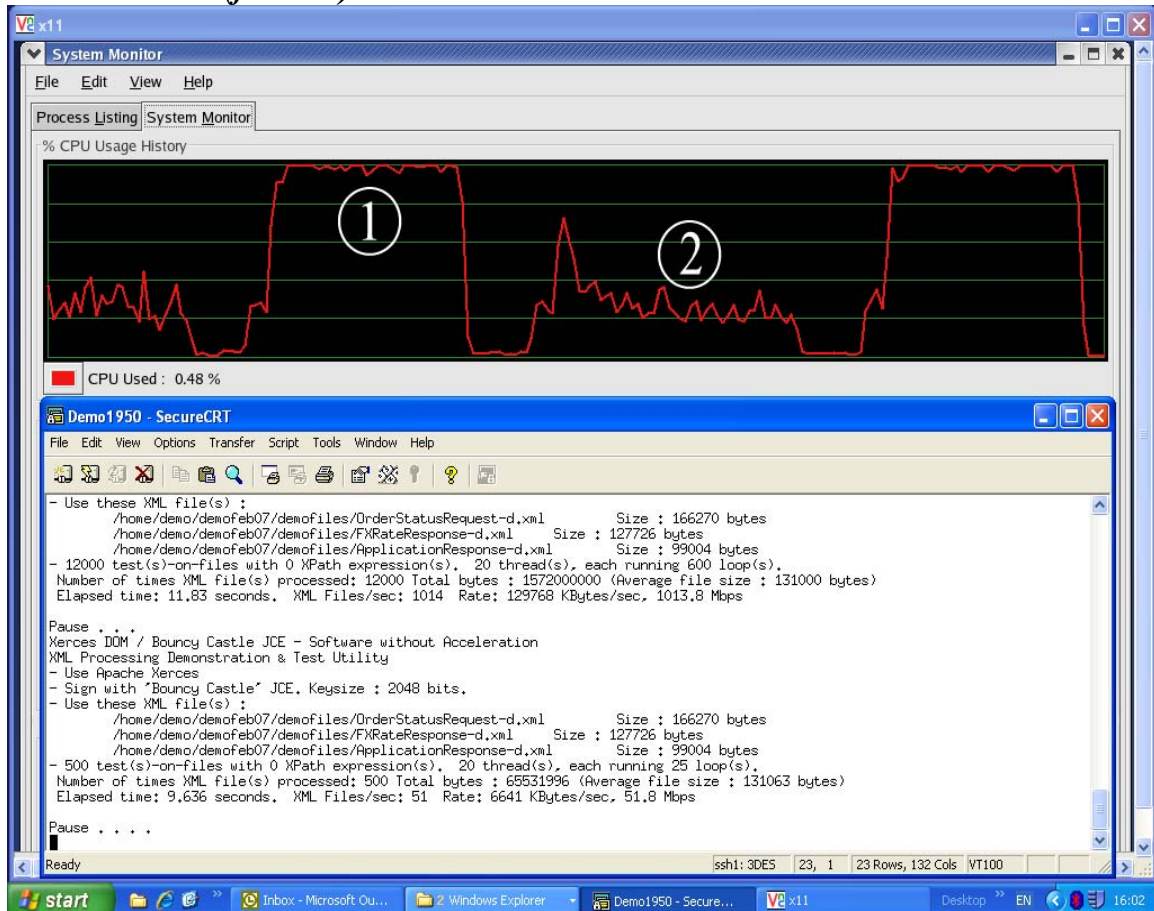
The solution is a balanced one, where the functions provided work in parallel at similar processing rates to provide both performance increase and server offload across all XML functions. The functions accelerated in hardware are XML parsing, XPath operations, and public-key processing (for SSL and WSS).

Performance testing - Parse, Navigate, XPath and Sign

The test bed used comprised a standard dual Xeon (dual core) server running Linux. A number of tests were conducted on a selection of publicly available XML files of different sizes (2KB to166KB). The first test consisted of a continuously repeating loop, that:

- (1) executed a java program that parsed, navigated and generated a digital signature on the set of XML documents (UTF-16). This program performed the XML operations in software using the Xerces, Xalan and Bouncy Castle libraries.
- (2) The same operations were then carried out using the Dajeil DH15K hardware accelerator and Dajeil's DOM API.

Test Screenshot (first test)

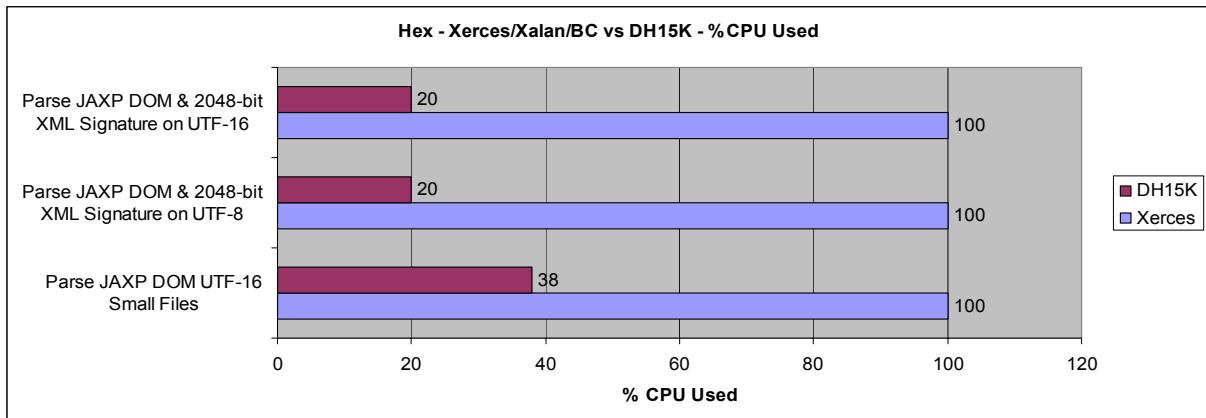
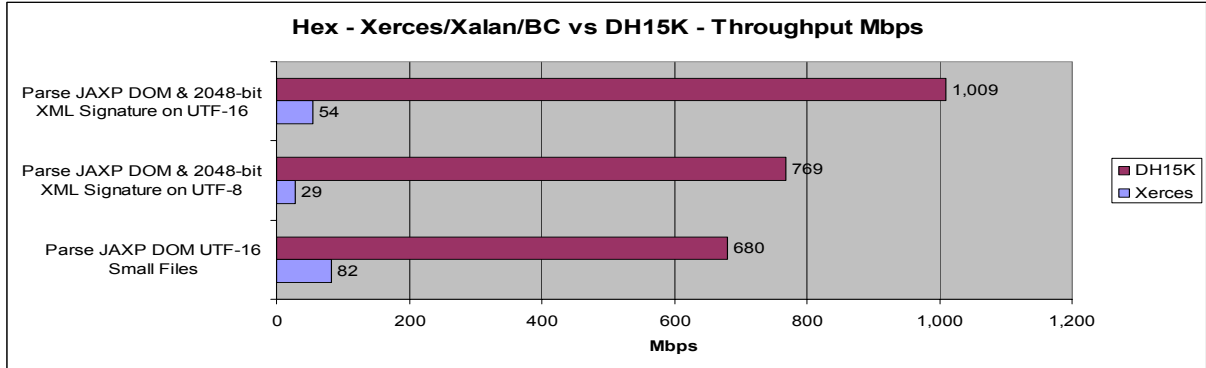


The following points are worth noting from the screenshot above:

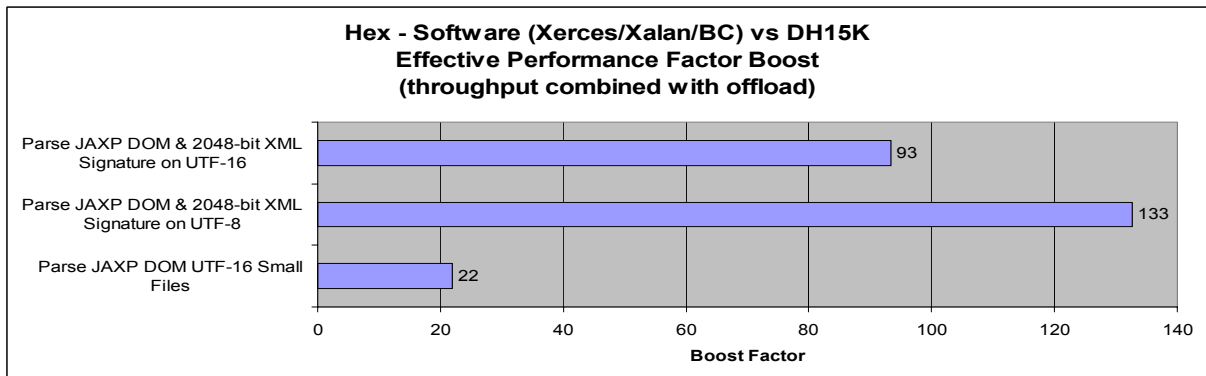
1. The CPU is 100% busy when the software is processing the XML at a throughput of approximately 50Mbps.
2. The CPU settles to about 20% busy using the DH15K, with throughput of 1Gbps.

The second test performed the same operations on a selection of UTF-8 files. The third test performs DOM parsing on a selection of small files. The results of all these tests are shown graphically below.

Test Results



When the results of the throughput increase are combined with the CPU offload we calculate the effective performance boost shown in the graph below.



It can be seen above that depending on the mix of public-key crypto, XPath and parsing, java applications benefit from performance increases in excess of 100 times over the standard Xerces libraries when using hardware acceleration. The final bar in the graphs above (i.e. Parse JAXP DOM) indicates the performance boost from the DH15K for parsing alone.

The overall performance increase is dependant on many factors and each application must be tested to obtain its performance boost.

Conclusions

In summary:

- Performance gains in excess of 100 times are possible using hardware acceleration in Web Services/SOA applications. Both throughput and CPU offload combine to achieve this performance boost.
- Proprietary interfaces will not be adopted by the major software vendors, as there are already widely adopted standard interfaces and libraries in use. Higher performance gains may be achieved using non-standard interfaces, but standard I/F support is essential for mainstream deployment of accelerators.
- A balanced solution, which removes a number of bottlenecks is important i.e. parsing, XPath and crypto etc. As each choke point to system performance is removed, another emerges, albeit at a higher performance point. Next generation accelerators (DH150K PCIe module) will deliver further improvements in overall system performance.

The potential gains from using XML hardware acceleration and its ease of deployment with standard APIs, support the now widely held belief that such specialized hardware is no longer optional in high performance XML servers and appliances.