Simulating a Multicore Supercomputer

Steven A. Guccione
Cmpware, Inc.
Austin, TX (USA)
Steven.Guccione@cmpware.com

Abstract - A new breed of high performance computing machines is emerging which seeks to make full use of the performance gains available from single chip multiprocessor or multicore devices. These systems use commercially available devices, and in some cases, define and fabricate their own high performance multicore devices. These devices are in turn used as processing nodes in parallel supercomputers. This paper describes real world experiences involving the design and simulation of such a multicore device and of the supercomputer system using these devices. Particular emphasis is placed on early architectural simulation and lessons learned.

I. INTRODUCTION

Large scale high performance computing, or *supercomputing*, has gradually transitioned from a very small number of powerful centralized processors to a very large number of less powerful distributed processing nodes. Recently, a new trend has emerged in the supercomputing field: *multicore multiprocessor supercomputers*. Because modern multiprocessors, including supercomputers, are overwhelmingly based on commercial desktop microprocessors, the trend toward multicore microprocessors in the commercial arena has naturally extend into the supercomputer field.

While the use of new multicore desktop multiprocessors in parallel supercomputers in inevitable, these are most often viewed as simple replacements for traditional uniprocessor devices. This view is largely due of the low level of parallelism and lack of sophisticated tools for these devices. Currently, dual core devices have gone mainstream and four (quad) core devices are currently on the horizon [1][[2][3][4]. Perhaps related to this low level of core parallelism, there has been a lack of new tools and techniques to fully exploit these multicore microprocessors.

Related to these multicore devices, a new breed of multiprocessor machines is emerging which seeks to make full use of the potential parallelism and performance gains from multicore devices. These systems are using higher performance commercial multicore devices, or in some instances, defining and fabricating their own high performance multicore devices for use in parallel supercomputers.

These devices diverge from the evolutionary approach to multicore found in desktop microprocessors. Instead of a relatively small number of powerful cores sharing a die, these new devices are characterized by larger numbers of cores, with each core often being a non-traditional type of processing unit. Such devices offer unprecedented levels of performance from a general-purpose programmable computation platform. Because of the high level of performance, these devices are increasingly finding their way into large scale parallel supercomputers, either as the primary processing node or as an adjunct coprocessor connected to the main processing nodes.

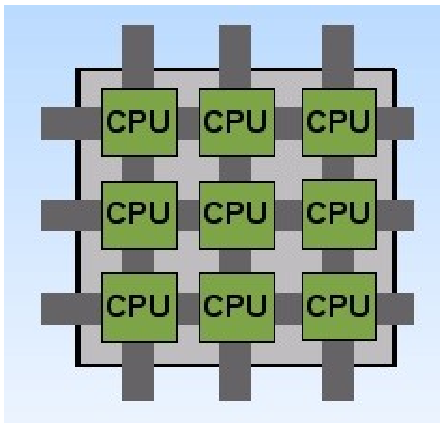


Figure 1: A multicore multiprocessor.

This paper describes real world experiences involving the design the simulation of such a high performance multicore device and of the supercomputer system using these devices. Specifically, this paper is concerned with the early simulation of this architecture and the impact of simulation on the design effort.

While simulation of such large scale systems has not typically been undertaken, tools such as the *Cmpware CMP-DK* [11] permit such systems to be modeled and analyzed with relatively little effort and at reasonably fast simulation rates. This the *Cmpware CMP-DK* also provides a built-in display environment for these models which allows accurate device and system level information to be gathered and analyzed, often interactively.

This combination of fast and flexible simulation combined with rich system data displays permitted early analysis of the device and system architecture which would not have been possible otherwise. This led to the discovery of substantial architectural deficiencies for a popular class of problems that necessitated a redesign of the system. The ability to identify such system deficiencies in such detail at such an early stage in development, before any device or system construction had begun has proven to be extremely valuable.

II. MULTICORE DEVICES

While many of the cluster-style multiprocessors have begun to use the new generation of multicore desktop microprocessors, this is mostly an inevitable side effect of the migration to multicore devices by microprocessor manufacturers. Since these platforms have committed to whatever high performance solution is made available by the large desktop microprocessor manufacturers, the adoption of multicore microprocessors is not necessarily one of choice. In fact, it will be interesting to see over time how these new architectures are used in large scale systems. The multiprocessor aspects of these devices will require special software support to be used effectively. Currently the level of software support for these multicore devices is relatively low.

While desktop microprocessor manufacturers such as Intel, Sun and Advanced Micro Devices are gradually edging into multicore architectures, several smaller companies are building more aggressively parallel multicore devices with dozens or even hundreds of processing units of a single device.

The multicore approach taken by established companies like Sun, Advanced Micro Devices and Intel are necessarily evolutionary. There is a massive installed base of software that must run, unmodified, on whatever the next-generation microprocessor offering from these companies happens to be. Essentially running existing operating systems such as Microsoft Windows and its applications is the only requirement of these devices.

Unfortunately, these desktop multicore devices provide a clear break with the decades long tradition of uniprocessor solutions. In the past, new transistor budgets were used to add various new architectural features to accelerate performance or perhaps extend instruction sets. But every new generation CPU devices made use of these newly available transistors to improve performance. Over time, the performance gains from increasingly large numbers of transistors resulted in diminishing gains in performance. But so long as transistors were cheap and performance crucial, this was the road map all manufacturers followed

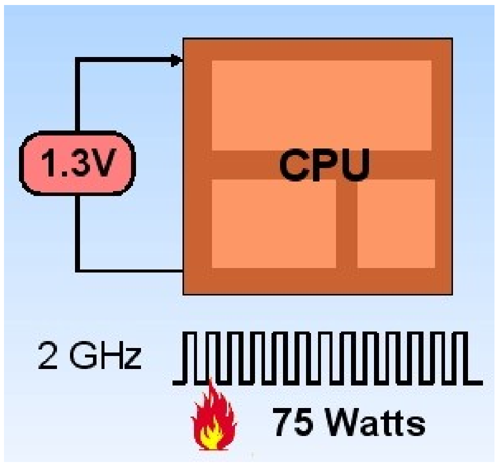


Figure 2: Power for a traditional uniprocessor CPU.

The new multicore microprocessors, however, break the existing programming model for these architectures. For the first time since the original x86 processor, for instance, applications will not directly benefit from the doubling of the number of transistors in this new generation of Intel and AMD devices. All of the existing software is written for a single core and cannot directly take advantage of the second core without being recompiled, and perhaps re-designed.

This significant break with the past was not done altogether willingly. Extending uniprocessor performance was abandoned simply because it was no longer feasible. The primarily problem was power consumption and the associated heat generation.

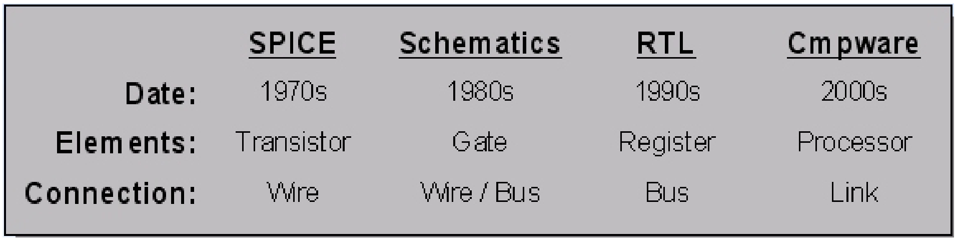
III. POWER CONSUMPTION

With modern unicore microprocessors approaching, and even surpassing, 100 Watts, change was inevitable. This amount of heat cannot be comfortably be dissipated in traditional air-cooled workstations and personal computers. To make matters worse, the trends in the underlying semiconductor technology had this number growing non-linearly.

The main goal of multicore devices was to provide increased performance without dramatically increasing power consumption. This all hinges on the relationship between clock speed and device voltage. To increase the clock speed, a non-linear increase in voltage is required. This, in turn, produces a non-linear increase in power consumption. Since it was no longer feasible to

TABLE 1

Simulation level over time.



	SPICE	Schematics	RTL	Cmpware
Date:	1970s	1980s	1990s	2000s
Elements:	Transistor	Gate	Register	Processor
Connection:	Wire	Wire / Bus	Bus	Link

use the extra transistor budgets to help increase the clock rate, they were used to create multiple cores, thus increasing the performance by increasing the parallelism. As figures 2 and 3 indicate, a dramatic decrease in power consumption can be achieved for identical levels of raw performance using a multicore approach.

Intel, Advanced Micro Devices and Sun took their existing designs and replicated them as multiple cores. This was a simple approach and provided some level of software compatibility. Code that ran on the previous generation of uniprocessor devices could still run on a multicore device -- it would just do so using a single core, or less than half of the device. Performance gains were no longer automatic.

IV. THE NEW PROGRAMMING MODEL

A group of smaller companies have recognized that once the basic uniprocessor programming model had been broken, a re-evaluation of existing architectures may be in order. All of the millions of transistors used over the previous decades in caches, buffers, and other assorted hardware used to increase uniprocessor performance were open to re-evaluation. The basic question now would be: are those transistors be better used by increasing the number of cores? While the large established processor manufacturers could not easily make this leap, several smaller companies have recently begun to manufacture devices containing dozens or even hundreds of simple processor cores.

The levels of raw performance of these new aggressively multicore devices turn out to be very high. Because hundreds of cores can be packed into a single device, orders of magnitude increases in raw performance are possible. Of course, the primary issue with these devices is the software. Software must be written to take advantage of these parallel computation resources.

These devices are particularly attractive to designers of supercomputers primarily because of their high level of performance. And since supercomputing software already tends to address issues of parallel execution, the programming model for these emerging devices is already familiar to the audience. For these reasons, these devices are rapidly finding adoption in the newer large scale multiprocessors [6][7][8]. Companies such as ClearSpeed as well as the Cell architecture from the IBM / Sony / Toshiba consortium are early adopters of the multicore trend, but more recent announcements by emerging architectures promise to push this level of parallelism even further.

In general, none of these new devices have provided substantial support for the multiprocessor design process. Using these devices in large numbers tends to be significantly more difficult than existing approaches using standard desktop microprocessors. While the raw performance is higher, it will tend to depend more on the structure of the software. And when multiple devices are used, issues concerning communication balance across devices becomes even more important.

While the new large scale multicore devices are gaining in popularity with supercomputer designers, most were not built with large scale multiprocessing in mind. For this reason, exploration of the use of a custom multicore device designed explicitly to support large scale multiprocessor systems has been investigated.

This paper is concerned with an ongoing project to design such a machine. Because the design in still in progress and has not been formally announced as of the writing of this paper, it will not describe specific details of the architecture, but will concentrate primarily on the simulation of the architecture during early development.

While somewhat unusual for a large, high-performance system, the ability to simulate the system and run application software led to important insights into the functionality of the system. Finally, because the system has not been announced, it will be referred to (for brevity) in this article as Multicore Multiprocessor Supercomputer, or MMS-1, for convenience.

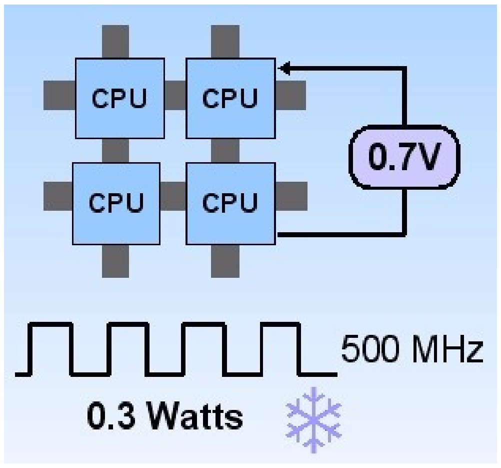


Figure 3: Power for a multicore CPU.

V. THE MMS-1 DEVICE AND SYSTEM DESIGN

The MMS-1 design is targeted specifically at matrix computations. Because these are popular representations for many supercomputer algorithms, dedicating hardware and networking resources to this class of problems is viewed as a way to provide significant performance gains. In addition, the use of multicore devices promises to increase the system performance dramatically while keeping power consumption relatively low. Because power consumption is becoming the limiting factor in supercomputer design, with power consumption in large systems measured in mega Watts, lower power consumption becomes very attractive. Reducing power consumption can reduce system cost due to reduced cooling needs, while lower operating costs and increasing reliability.

The general architecture of the MMS-1 is defined to be a 2D array of processors with floating point capability. These processors are interconnected by a 2D toroidal mesh network. While there are many other significant details of the architecture, discussion of these features is not necessary to the discussion of the simulation experiences and will be left to other publications.
One choice in the definition of the architecture has a direct impact on the simulation model. The array and device sizes of the MMS-1 are not defined as fixed values, but are left as parameters. Clearly, the number of cores per device is dependent on the implementation technology, and the number of devices per system depends on the configuration. The goal was to define a system that physically and logically could accommodate a large range of matrix sizes.

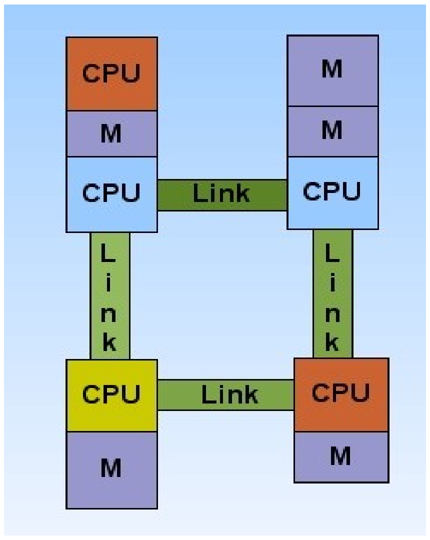


Figure 4: The Cmpware CMP-DK multiprocessor model.

In particular, an FPGA prototype of some small number of processing elements would also be implemented, primarily to aid in the hardware verification process, although such FPGA implementations could conceivably also be used as the computation platform in some systems.

While it is typical to perform simulations of hardware, system level simulation has only recently become available. And, primarily due to the processing power required, simulation of large scale multiprocessors is considered somewhat unusual. Initially, simulation of the multicore device using the *Cmpware CMP-DK* toolkit was investigated. It was quickly realized that because of the parameterized nature of the architecture, simulating the multicore array on a larger scale would also be identical to simulating the entire multiprocessor system.

VI. MODELING WITH THE CMPWARE CMP-DK

The *Cmpware CMP-DK* is a simulation and software development environment specifically for multicore devices. The *Cmpware CMP-DK* is typically used to construct a simulation model of the multiprocessor early in development. In general, these models consist of an array of traditional microprocessor cores connected by some interconnection network fabric.

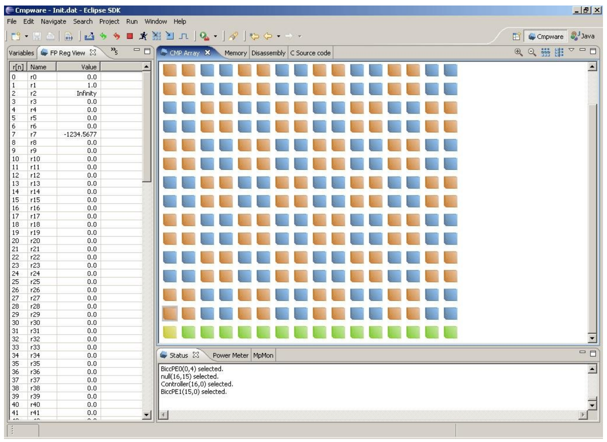
The *Cmpware CMP-DK* address the problems of simulating such systems by operating at a higher level than most traditional simulation tools. While today's circuit design tools operate at the gate or register level, such granularity is both difficult to model and slow to execute. The *Cmpware* environment takes simulation to a higher level and defines the basic computational element as a '*processor*'. This processor is typically a standard microprocessor core, and the *Cmpware* environment comes with several standard models, including *MIPS-32* or *Sparc-8* and *NIOS II*. While standard models are useful, custom processor models can be easily constructed, typically in just a few hours. And while support for traditional microprocessors is provided, a 'processor' can be anything which has inputs and outputs and a clock. This permits 'hardwired' cores to be modeled and mixed with other cores in a multiprocessor.

As part of the processor model, memory can be defined to be any size and may also exist in multiple banks. Non-contiguous memory banks and shared memory in various configurations is also supported.

While 'processors' are the basic computational element, 'links' are used to connect these processors. A link is defined as a communication channel with one source and one or more destinations. These channels can be of any bit width and may have handshaking control or be 'open loop' as in hardware pipelines. Links may also have state and may implement a shared register or even a FIFO. Any communication mechanism fitting this high level model can be simulated in the *Cmpware CMP-DK*.

Because the core of the MMS-1 was relatively simple, the model for the core was created in significantly less than one day. And because the torus network model is one of the standard network model options in the *Cmpware CMP-DK*, there was no work to do initially modeling the network. It should be mentioned that the existing network model consists of just over a dozen lines of highly repetitive code (implementing the north, south, east and west links are similar) and the processor model for the MMS-1, with its limited instruction set, was approximately 150 lines of code.

This model was imported into the *Cmpware CMP-DK* Eclipse-based user interface. This provides a multiprocessor debugger-like view into the architecture, supplying various views of state information at both the device and source code level. This interface comes for 'free' after developing the model. Rather than constructing displays and integrating them into an IDE as a separate effort, the *Cmpware CMP-DK* takes information embedded in the multiprocessor model and uses it used to dynamically drive the interface, providing a reliable and consistent view of the architecture, even as it changes from version to version.



r[n]	Name	Value
0	r0	0.0
1	r1	1.0
2	r2	Infinity
3	r3	0.0
4	r4	0.0
5	r5	0.0
6	r6	0.0
7	r7	-1234.5677
8	r8	0.0
9	r9	0.0
10	r10	0.0
11	r11	0.0
12	r12	0.0
13	r13	0.0
14	r14	0.0
15	r15	0.0
16	r16	0.0
17	r17	0.0
18	r18	0.0
19	r19	0.0
20	r20	0.0
21	r21	0.0
22	r22	0.0
23	r23	0.0
24	r24	0.0
25	r25	0.0
26	r26	0.0
27	r27	0.0
28	r28	0.0
29	r29	0.0
30	r30	0.0
31	r31	0.0
32	r32	0.0
33	r33	0.0
34	r34	0.0
35	r35	0.0
36	r36	0.0
37	r37	0.0
38	r38	0.0
39	r39	0.0
40	r40	0.0
41	r41	0.0

```
BiccPE0(0,4) selected.
null(16,15) selected.
Controller(16,0) selected.
BiccPE1(15,0) selected.
```

Figure 5: The *Cmpware CMP-DK* Integrated Development Environment (IDE) and the MMS-1 model.

Since this was a completely new architecture, even the most basic software development tools had to be constructed. An experimental tool from *Cmpware* called *AutoModel* was used to produce a simple stand-alone assembler and disassembler from the architectural description in the model. This was used to construct early test programs, which were loaded into the multiprocessor model and executed.

The *AutoModel* data-driven model and tools approach is somewhat slower than the hand-crafted models by a factor of approximately 5x, but the ability to rapidly have tools and simulation models which were completely in sync and guaranteed to be free of certain common types of programming errors proved to be very useful. It permitted very rapid cycling of design changes without requiring communication, specification and implementation of supporting tools to program the architecture. This permitted a much wider range of experimentation than would have otherwise been possible.

VIII. THE MMS-1 SIMULATION EXPERIENCE

While the SIMD control structure was still under design, the model implemented a small local memory on each processor and loaded each of these memories with identical code. While the model was clearly MIMD in this case, loading identical code to each processing node and having the processors execute the same instructions in lock-step effectively emulated a SIMD machine. This approach was crucial to pressing forward with software development and system test, even as the system details were still under active design.

Performance of the simulation of such a large system is also a concern. While the simulation of complex microprocessors can be extremely slow, the relatively simple cores used by the MMS-1 and other similar multicore devices generally eliminate these simulator performance issues. Processors typically execute at between one and two million cycles per second. This lets relatively complex code execute for relatively long runs and relatively large multicore systems. For the MMS-1, a 16 x 16 array (256 processors) was most often used for experimentation, but this was easily modified. The array size was controlled by a single parameter which can be changed in the IDE and would generate a new model on on the order of one second, somewhat longer for very large arrays.

It should be mentioned that much of the choice for the 16 x 16 array for testing and experimentation was not due to the performance of the simulation model. Performance was still interactive with this size array and larger sizes were sometimes used. The choice was primarily done because larger size arrays added little if any valuable information to the investigations being performed and often only served only to complicate the modeling process.

While the *Cmpware* simulation models operate at a higher level than other simulation environments, they are still very much tied to the underlying architecture and realistically models the system at the hardware level. One result of this is that the mechanism for loading code and data quickly came under scrutiny. One alternative was to use the existing communication network to perform this operation. Other techniques to directly address processors and their internal state, including the use of JTAG-style debug loading of data was discussed.

While these details were being addressed, the model was quickly fitted with a 'controller' node, which emulated dedicated hardware used to 'poke' data into processor memory and registers. While this specialized node was only done to move software development further along, it also had an impact on architectural discussions.

Finally, one relatively small point concerning the *Cmpware* displays emerged. While the *Cmpware* toolkit was essentially integer-based, the MMS-1 was using a large number of floating point registers. The default *Cmpware* displays would show these values as integers, or at best as hexadecimal numbers. While that was acceptable, particular at the early stages of design, it quickly became tedious to convert the data formats to and from floating point.

Fortunately, the *Cmpware CMP-DK* supports custom displays. In less than 20 lines of code, a new register display was created and integrated into the the IDE. This displayed all of the floating point registers in a standard floating point format and immediately enhanced the productivity of the programmers. This additional display took much less than one day to implement. While this display was relatively simple and closely resembled existing displays, new displays of any complexity can be integrated into the *Cmpware CMP-DK*. Additionally, these externally created displays are 'pluggable' and can easily be bundled and deployed. In fact, this display was deployed on an internal web server and downloaded and integrated by software developers as an Eclipse software upgrade, without any intervention from the developers of the display.

Perhaps the most significant event of the MMS-1 simulation was the uncovering of a significant imbalance between the system processing capability and the input and output requirements. While such balance is important, it is typically highly application dependent. In the case of the MMS-1, it was discovered that certain matrix operations would lead to significant data starvation and a low utilization of the processor resources.

While such problems are common in high performance computing, they are also difficult to analyze and discover, no matter how obvious they sometimes seem in hindsight. It was only the ability to simulate the architecture at the software / application level that uncovered this significant architectural problem. Several alternatives increasing the I/O bandwidth and rearranging the computational resources are currently under investigation. Unfortunately, some of the potential solutions involve architecture details not yet disclosed at the time of the authorship of this paper.

IX. CONCLUSIONS

This paper has discussed the simulation of the MMS-1 multicore multiprocessor supercomputer using the *Cmpware CMP-DK* toolkit. While originally aimed at the simulation and software development for multicore devices, it was quickly realized that a complete multiprocessor system could be adequately simulated. While such large scale systems are typically not simulated at the application / software level, this experience was extremely valuable in guiding the design, and in one case even uncovered a significant architectural deficiency. Perhaps what made the simulation experience so effective was that models were able to be developed and modified quickly and that a complete integrated development environment was made available. This architectural analysis and software development environment was especially useful because it was available so early in the design process.

REFERENCES

[1] “The Future of Microprocessors”, David Patterson, U. California at Berkeley, June 2001. http://www.cs.berkeley.edu/~pattrsn/talks/NAE.ppt

[2] Chip Multiprocessing Resources: http://www.princeton.edu/~jdonald/research/cmp/

[3] “Intel Demonstrates Breakthrough Processor Design”, http://www.intel.com/pressroom/archive/releases/20010828comp.htm, August 28, 2001.

[4] “AMD Announces Technology Milestone With Its Multiple-Core Strategy”, http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~86455,00.html, June 14, 2004.

[5] ”Sun Drives Multithreaded Processor Innovation with New UltraSPARC IV+”, http://www.sun.com/smi/Press/sunflash/2004-10/sunflash.20041005.2.html, October 5, 2004.

[6] Aaron Ricadela, "In Depth: Supercomputers Get A Speed Boost From Specialized Chips", *Information Week*, http://www.informationweek.com/story/showArticle.jhtml?articleID=190400264, July 17, 2006.

[7] "ClearSpeed Teams with IBM to Deliver Hybrid Clusters", *HPC Wire*, http://www.hpcwire.com/hpc/707056.html, June 27, 2006.

[8] John Markoff, "I.B.M. to Build Supercomputer Powered by Video Game Chips", *New York Times*, http://www.nytimes.com/2006/09/07/technology/07compute.html, Sept 7, 2006.

[9] W. Daniel Hillis. The Connection Machine. The MIT Press, Cambridge, MA, 1985.

[10] “Eclipse Platform Technical Overview”, http://www.eclipse.org/whitepapers/eclipse-overview.pdf, February, 2003.

[11] Cmpware, Inc. http://www.cmpware.com/, 2006.