CRAY
THE SUPERCOMPUTER COMPANY

Meeting the Demands of Unstructured Data with an e**X**treme **M**ulti**T**hreaded Machine

Introducing the Cray XMT™ Supercomputer

Introduction

Every organization has a wealth of information in unstructured form. While the number can vary by industry, research shows that over 80 percent of data is unstructured – and at least one third of that data holds potentially vital business insight.

The challenge is that few have found a way to access and use this valuable data.

Unstructured information is generated continuously – through everything from emails to memos, user groups, chats, reports, letters, surveys, white papers, marketing material, research, presentations and Web pages. Market research firm International Data Corporation (IDC) reports that more than two billion new Web pages have been created since 1995 with an additional 200 million new pages added every month. The Web is only one example. Each day an organization's employees, customers, suppliers and competitors all add to this vast information resource.

Recently, the U.S. and U.K. governments have started making government databases available on the Internet, believing the data itself is an opportunity for business growth. However, this trove of data is "raw" or unstructured and accessing its invaluable contextual information requires capturing it first. The difficulty comes in bringing an unstructured mass of information into the structured world.

Structured data conforms to rules, always behaving in an orderly manner. Represented by a clearly defined foundation of rows, columns, labels and tags, structured data lends itself to repeatable analysis and deduction. Unstructured data, on the other hand, has no readily discernable format or anatomy and can be found just about anywhere – in transcribed conversations, purchase orders, design specifications.

Compounding the problem, this unstructured data is often intertwined with highly definable data, with little consistency in how both types of data interact or depend on one another. Consequently, the tools and techniques that successfully transform structured data into business intelligence simply don't work with unstructured data.

While the focus on unstructured data usually revolves around text-based information, a wealth of knowledge resides in non-textual formats (i.e., pixels and sounds) such as phone conversations, voice mails and photographs. However, this type of information is also more difficult to work with than static text, and in many cases, only extremely customized software can transform it into a useful format for business analysis and corporate intelligence.

Despite the challenges to harnessing unstructured data, consider the problems you could solve with a more complete view of your business and the industry and environment from which it operates. Think about the insight you could gain from pattern matching, scenario development, behavioral prediction or anomaly identification using all your data. From marketing analytics to fraud detection to anti-terrorism efforts to power grid situational awareness, the number of applications for this knowledge discovery continues to grow. Moreover, if you could gain this knowledge in real time, it could have an even greater impact on your organization's performance.

The world of information is not going to become less complex. The more we communicate with one another, the more unstructured data becomes fused with structured data. Of course, some industries will have more unstructured data than others and the amount of unstructured data will even vary within an organization's individual business segments and functional units.

In all the variance, the one certainty is that **organizations need to better process and understand their universe of unstructured data** – from discovery to sourcing, transformation, storage, analytics and dashboard business intelligence solutions. Unstructured data and non-rationalized text cannot be hastily placed into a structured world and expected to retain its initial meaning or usefulness.

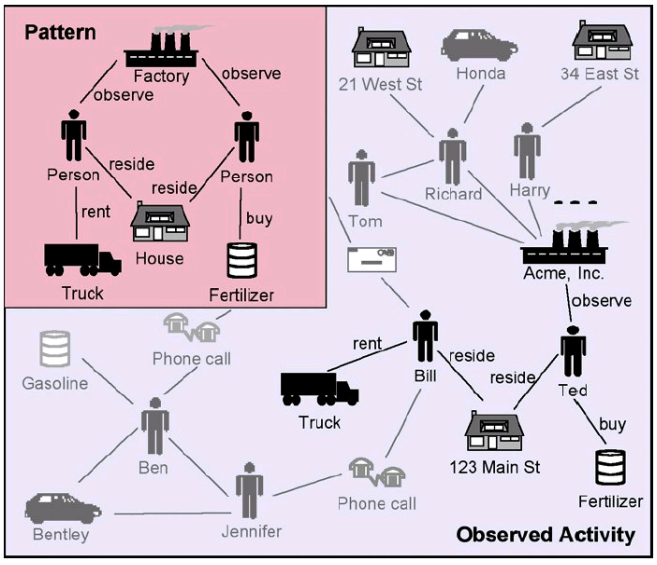
To be effective, unstructured information must undergo both transformation and integration processes that extend beyond the norm. The challenge is threefold:

1. **Data management.** Unstructured data stores continue to grow, and even more so when including disparate sources in order to have a richer store of information from which to mine intelligence. Management of this data is a large-scale problem on its own.
2. **Extracting vital intelligence.** How to transform, manipulate and analyze data in such a way as to extract useful intelligence.
3. **Extracting intelligence efficiently.** How to transform, manipulate and analyze data in or near real time (in order to extract insight at its most timely).

Solving these challenges requires a sophisticated technical architecture as well as resources experienced in bridging the gap between structured and unstructured domains and textual analytics.

Understanding the Technical Challenges

The real computational challenges stem from the algorithms ideal for managing and analyzing unstructured data. In order to perform well, these algorithms need thousands of computational threads interacting with a large globally shared memory with fine-grain synchronization. Examples of these types of algorithms exist in large graph analytics problems in intelligence, protein folding and bioinformatics, semantic databases, social networks analysis, computational geometry problems in graphics, scene recognition and tracking or any other application with irregular, practically random, remote memory access patterns.



One specific example of a graph analytic problem is searching for a pattern of activity in graphs containing different types of data. (Figure source: T. Coffman, S. Greenblatt, S. Marcus, "Graph-based technologies for intelligence analysis," CACM, 47 (3, March 2004): pp 45-47) Traversing this graph and looking for connections causes every

memory reference to be global and random since every reference leads to new connections located in different sections of memory. These problem types cannot be solved on distributed memory systems because the latency of the remote memory references keeps it from scaling to a reasonable number of processors.

Commercial computer systems leveraging commodity hardware (Intel® or AMD® microprocessors and Gigabit Ethernet or InfiniBand switches) are adequate for cache-friendly algorithms but not for irregularly structured problems. For example, a typical microprocessor needs a few instruction execution cycles to get data from its local cache. If the data is in an L2 or L3 cache, it takes tens of cycles to fetch. If the requested data is in local memory, it can take over a hundred instruction cycles to fetch. If the algorithm needs data from another processor's local memory, then by the time the message is sent and received, over a thousand instruction cycles may have passed. When the data is not available, the processor has to stall or spin until the data becomes available. Commercial computer systems can only make efficient use of their processors when the algorithm allows the data to be cached locally and operated on many times before being flushed.

Two related hardware issues when dealing with irregularly structured problems are:

1. **Interconnect networks**. The network needs to have low overhead for sending/receiving small messages and be scalable to a large number of nodes.
2. **Locking memory access**. The globally shared memory needs to be able to do fetch-and-add type of operations on every word of memory.

Most architectures distribute memory across many nodes. Accommodating this distribution requires an interconnect network that communicates requests between nodes, usually via messages passed between these nodes. This first issue has to do with the efficiency of the interconnect in performing small transfers. Many interconnects depend on data being moved in large blocks for bandwidth efficiency, making small transfers inefficient. Therefore, interconnect performance can impede shared memory operation.

The second issue is even more difficult. Once memory is distributed the accesses need to be controlled, so the system needs a hardware mechanism that can lock memory accesses. Without special hardware to create and manage locks, the locks must be managed by software. Software lock management in a distributed environment is complicated and difficult to get to perform correctly and/or quickly.

Another drawback to commercial systems is memory size limitations. Large memory systems are difficult to engineer and expensive, making it architecturally and economically much easier to limit a server size to a few hundred gigabytes of memory. Providing larger memories is possible by clustering, but the interconnect network between server nodes are geared for higher level protocols like TCP/IP or MPI. The transfers are targeted at large blocks of memory, but the kind of transfers useful for these problems are small – usually a word. Thus, the problem for unstructured data analysis is in creating a large shared memory with the same access characteristics as a local memory attached to a processor.

Finally, multithreading requires a software stack tuned to manage massive multithreading. The compiler must recognize the threading possibilities in a code and compile threaded requests. When the application is executed, the runtime support must be able to create threads with very low overhead and switch between them without large context switch times. Threading has seen some recent development, but massive multithreading and a complete software stack support are still rare.

The Solution: The Cray XMT™ supercomputer

The Cray XMT supercomputer is a massive multithreading engine specifically designed to tackle the problems associated with managing and analyzing large-scale irregular data structures and random reference patterns.

Among its features, the Cray XMT system has a global shared memory – the product of the nodes and interconnect network that offer single-word access from any processor to any word in the system with the ability to hide the latency. The distributed access is guarded by hardware-based memory semantics, so data can be managed and controlled across a scaled system.

Cray also provides a wealth of software that manages the hardware, and provides the programming environment to make a user code capable of taking advantage of the multithreading capability.

Cray XMT architecture

The Cray XMT system's multithreaded architecture is based on the Cray Threadstorm™ processor – a custom-designed device incorporating 128 parallel threads. Commodity processors can only issue a few outstanding memory requests before they have to stall to operate until the data arrives. This lag means a commodity processor in a parallel system can spend most of its time idle, waiting for fetched data. The Cray XMT processor is designed to support over a thousand outstanding memory requests. Therefore, any thread running on the Cray XMT system may also stall, but other threads are likely ready to issue instructions. This means the Cray Threadstorm processor stays busy – issuing memory requests to local and remote memory to maximize data communication throughput. A Cray XMT system can have 512 Threadstorm processors with independent threads all simultaneously issuing memory requests across the network. This performance also means the Cray XMT system excels in irregular search and graph applications.

Latency tolerance implies that a global address space is reasonable to use. If their application has sufficient parallelism, programmers can ignore the cost of remote references. Shared memory calls for efficient synchronization. Distributed memory architectures like a commodity cluster pass data from one processor to another via a message-passing protocol. The sending and receiving of a message provides an implicit synchronization point for the two or more processors involved. For shared memory architecture like the Cray XMT supercomputer, all processors can access the shared memory, eliminating the need for message passing. Thus, we must provide some other form of synchronization between threads executing in parallel.

The Cray XMT architecture provides an efficient, ubiquitous synchronization mechanism in the hardware. It comes in the form of two extra bits on every 64-bit word, called the "full/empty" bits. The

instruction set includes instructions such as load-if-full-and-leave-empty and store-and-leave-full (as well as normal loads and stores). Thus, there is a programmer productivity byproduct of multithreading: assuming abundant parallelism, shared memory and low-overhead synchronization is the simplest parallel programming model we have. We have seen Cray XMT programs that are 40 times shorter than their equivalents written for a distributed memory system using MPI message sends and receives. The Cray XMT system provides a large, globally shared memory by using a high performance interconnect designed for massive scale and small transfer sizes. The efficiency of the Cray XMT network is often 50 percent higher than the commodity networks at scale. The interface to the network is performed in hardware by the processor and network, so that the performance of a memory request is as fast as the absolute speed of the network. This allows the Cray XMT system to manage and process large-scale data stores efficiently.

The Cray XMT supercomputer uses a C/C++ compiler that has been modified to analyze loop-level programming structures and automatically parallelize them. A very advanced design, the Cray XMT compiler is as sophisticated and powerful in terms of its parallelization and optimization capabilities as any other in the computer industry. The compiler can also create load instructions far out into the future and launch them. This capability keeps the processor fully utilized, executing a result every clock cycle.

The runtime environment supports the application processes and the underlying threads across the Cray XMT processors. It creates threads through lightweight primitives supported in hardware. Threads can be created and stopped as needed with extremely low overhead. Since the runtime is aware of the system state, the support can be refined as the application runs to utilize current available resources. The programming environment also contains productivity tools like a parallel debugger and performance analysis tools.

Conclusion

The growth of unstructured data and the challenges in deriving value from it are a concern – and an opportunity. While the algorithms that sift through data in search, discovery and graph applications can run on commodity clusters, they are a poor match for current hardware and software architectures. Given the continual growth in unstructured data, mining it will only become more challenging.

Cray has been architecting a multithreaded system to address these technical problems for 20 years. With the Cray XMT supercomputer, Cray brings a complete hardware and software solution – delivering the performance you need to harness your enterprise's data and influence results in real time.