

CSP: A System-Level Architecture for Scalable Communication Services

Greg Regnier

Intel Enterprise Architecture Lab, Intel Corp.

Index words: communications, System Area Network, VI Architecture, network processor, servers

ABSTRACT

The boundary between the communication equipment at the network edge and the servers within the data center is blurring. Appliance vendors are flooding the market with new capabilities, while router vendors are scrambling to add these services to their traditional transport services. The result of this competition is a set of ad-hoc technologies and capabilities that provide services at the network edge. This paper describes the Comm Services Platform (CSP), a system-level architecture for this new communication services tier of the data center. CSP describes a set of architectural components to provide scalable communication services built from standard building blocks. These building blocks consist of emerging server, I/O, and network technologies. The building blocks of CSP include a System Area Network (SAN), the Virtual Interface (VI) Architecture [1], programmable network processors, and standard high-density servers.

INTRODUCTION

The three-tier data center model is well known. The first tier consists of front-end servers that provide Web, messaging, and various other services to clients on a network. The middle tier handles transaction processing and generally implements the data center business logic. The back-end consists of databases that hold persistent data. We see a fourth tier emerging in this model between the network and the front-end server farm: the communication services tier. These services operate on network traffic at and above the network layer, with well known examples such as load balancing, security (firewall and SSL), caching, and others. These services increase the responsiveness and throughput of the data center by offloading communication related tasks from the front-end server farm and by distributing the client request load.

The Comm Services Platform (CSP) describes a system-level architecture for a scalable, high-performance communication services tier based on emerging server and network technologies that are, and will become, standard building blocks. These technologies include a System Area Network (SAN) and the Virtual Interface (VI) Architecture for high-performance message-passing, programmable network processors, and standard high-density servers. CSP includes a new core service, the SAN Proxy Service, which offloads and decouples TCP/IP processing from the front-end server farm.

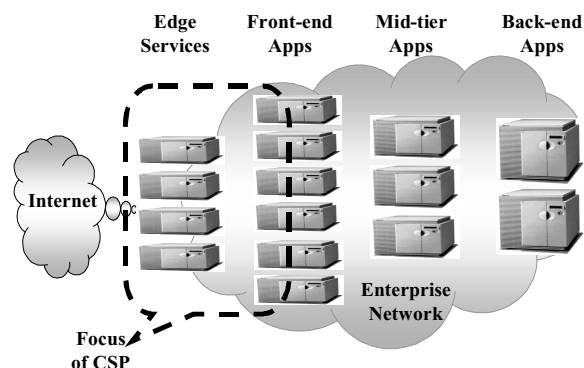


Figure 1: The N-Tier Data Center

CSP OVERVIEW

The main goal of the Comm Services Platform (CSP) research project was to show that scalable communication services can be built based on standard building blocks. CSP describes these building blocks and describes their role in the overall system.

This paper outlines the CSP architecture, then focuses on its benefits for the communication-intensive, front-end applications of a typical data center. It then discusses the

remaining challenges in order to reap even greater benefits from emerging server and communication technologies.

CSP Architecture

The CSP architecture consists of multiple functional elements, or nodes, interconnected by a Virtual Interface (VI) Architecture-enabled System Area Network (SAN). These elements can be enumerated to enable the construction of CSP systems with varying levels of functionality, scaling, and performance. The decomposition of the CSP into a functional pipeline of building blocks allows scaling of the network, proxy, and application nodes independently. Figure 2 illustrates the CSP system architecture. The functional elements of the CSP system architecture are described as follows:

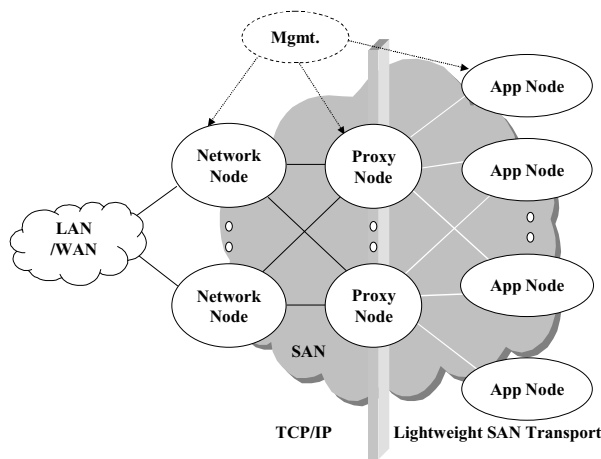


Figure 2: The CSP architecture

The SAN provides the connectivity between elements of the CSP. CSP nodes attach to the SAN using the capabilities described by the VI Architecture specification [1]. These VI capabilities allow for more efficient communication between the nodes of the system than is possible using existing Local Area Network (LAN) technologies [6]. Our performance tests use the VIPL API [2] to allow the applications to bypass many operating system overheads and access the SAN hardware directly. The InfiniBand* architecture [3] is the most prominent example of an emerging SAN technology based on VI capabilities.

Network Nodes provide the interface between the client IP network and the SAN. They perform the first level of processing in the functional pipeline of the CSP system by processing IP packets, encapsulating them within SAN messages, and forwarding them for further processing. Examples of Network Node packet processing include Layer-3 packet forwarding and Layer-4 load balancing and flow classification. Network

nodes are optimized for fast packet processing and designed to operate at the full line-rate of the client network attachment. These requirements are best met with programmable network processors.

Proxy Nodes perform the next level of processing in the CSP pipeline. Proxy nodes terminate client TCP/IP sessions and communicate with the front-end server farm over a lightweight SAN transport. Proxy nodes based on Intel® Architecture processors additionally provide the compute power to perform various higher-level network services such as http proxy services, Web content transformations, security, and content-based distribution of Web transactions.

Application Nodes form the front-end server farm and host well-known applications, such as Web, e-mail, or directory services. Application nodes are built from standard high-density server hardware and run standard operating systems and applications. Application nodes within CSP use the sockets API over a lightweight SAN transport for efficient client communications by bypassing the kernel-resident network stack.

The **Management** function of CSP provides dynamic resource discovery and configuration services, along with network policy management. As a central location for resource and policy information, the management function is assumed to be configurable in a redundant manner.

Interfaces

Figure 3 shows the interfaces of the CSP. The network node bridges the IP-based client network and the SAN of the data center by tunneling TCP/IP packets through SAN messages. It can also perform load balancing between multiple proxy nodes at Layer 3 (IP) and/or Layer 4 (TCP).

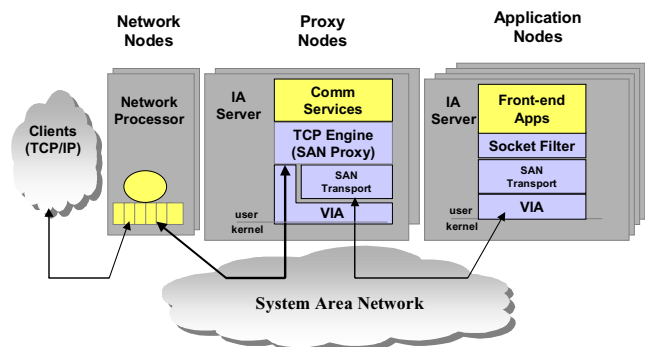


Figure 3: CSP interfaces

At the proxy nodes, the SAN tunnel allows the SAN proxy service to transfer SAN messages between itself and the network node directly from the user level using the VIPL programming interface. This allows much greater communication efficiency by bypassing the operating system overheads associated with kernel-based network stacks. It terminates the TCP/IP sessions at the proxy node and converts them to a higher level SAN transport protocol that requires less control traffic; state management is thus more efficient for the front-end application nodes. Figure 4 shows the flow of packets for an HTTP/1.0 transaction on the CSP with TCP/IP termination and protocol translation at the proxy node, illustrating the simplified control traffic at the application node.

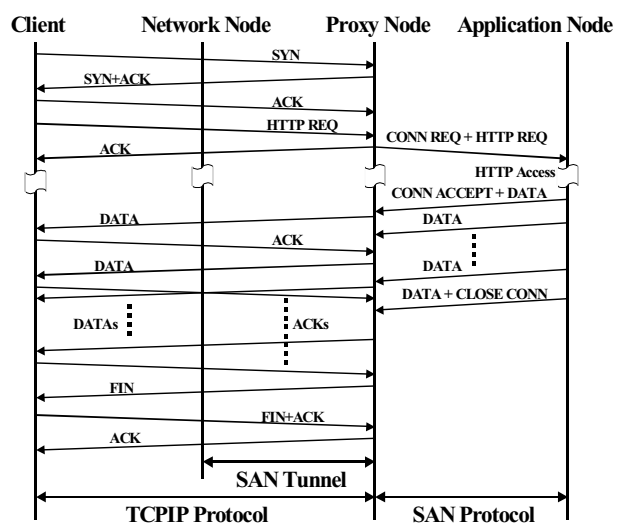


Figure 4: Example HTTP Transaction on CSP

The proxy node is the logical place for higher level communication services to exist since the SAN proxy service has each packet at the user level, and can easily pass the packets on to higher level services efficiently. Examples of these higher level services would be caching, security, and transcoding.

Given the TCP/IP termination at the proxy nodes, the application nodes can now process client requests over a more efficient communication stack. A lightweight SAN transport protocol built on top of VIPL provides basic end-to-end flow control between the proxy nodes and the application nodes. Another thin layer of software, the “socket filter” maps the legacy sockets interface to the SAN transport. This stack runs at the user level in the performance critical path.

CSP PROTOTYPE

A combination of system prototyping and simulation was used for evaluation of the Comm Services Platform (CSP) performance, scalability, and for architectural validation. Prototyping with off-the-shelf hardware and software components was focused on dealing with the real-world problems of implementing a CSP system with existing technology.

For the network node, we used an Intel[®] IXP-1200 [4] network processor platform. We developed the microcode that tunnels client Local Area Network (LAN) traffic into the System Area Network (SAN). The prototype also performed simple Layer-4 load balancing to distribute the traffic to multiple proxy nodes.

For the application and proxy nodes of the CSP prototype, we used standard rack-mounted servers, each with a single 800 Megahertz Pentium[®] III processor and a 64-bit, 33 Megahertz PCI IO bus. These servers ran the Linux^{*} Version 2.2 kernel.

For the SAN network interface, we used a proprietary SAN (cLAN^{*}) from Giganet Inc. that natively implements the Virtual Interface (VI) Architecture with a maximum hardware transfer rate of 1.25 Gigabits per second full duplex. For the LAN measurements, we used off-the-shelf Gigabit Ethernet PCI network adapters.

CSP APPLICATION RESULTS

Given that a major focus of the Comm Services Platform (CSP) is to provide a more efficient communication infrastructure for front-end server applications, we focus here on the performance analysis of the application node. A more complete analysis of the complete CSP system is described in our paper to be presented at the 2001 Usenix Symposium on Internet Technology and Systems [5].

Software overhead in the operating system network stacks, specifically sockets and TCP/IP, has been identified as a major bottleneck in server communications performance. Even after factoring in increases in server CPU speeds, and incrementally offloading portions of the stack onto network interface hardware, communication demands are outstripping the ability of servers to perform the required protocol processing. To overcome these inefficiencies, in particular for the communication-intensive front-end applications, CSP exploits the capabilities of a Virtual Interface (VI)-enabled System Area Network (SAN) fabric as the system interconnect.

* Other brands and names may be claimed as the property of others.

As an initial experiment, we ran a simulated Web transaction performance test. This transaction performance test consisted of iteratively sending a simulated HTTP *get* request from a client, and sending reply messages of varying sizes from the server. The transaction rate and the server host CPU cycles spent per transaction were measured. The SAN version of the test used the VIPL programming interface directly over the native SAN hardware. The LAN version of the test used the sockets API over TCP/IP running over the Gigabit Ethernet LAN hardware. The results of this experiment are shown in Figure 5.

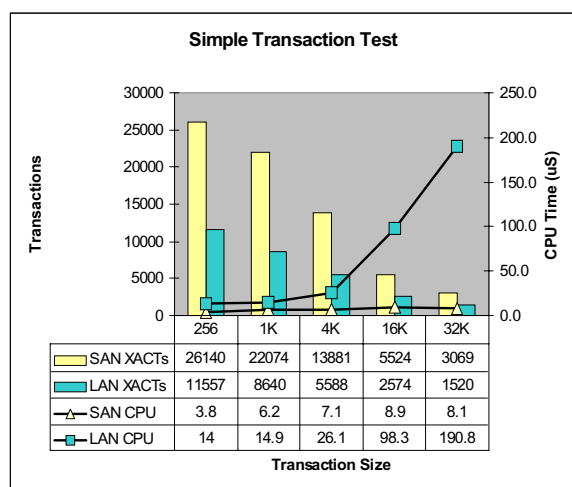


Figure 5: Transaction test results

These results illustrate that, for a given reply size, the SAN version of the test was able to execute more than double the number of transactions per second than a comparable Local Area Network (LAN) using TCP/IP. Furthermore, the number of CPU cycles spent per transaction for the SAN was significantly lower than for TCP/IP.

The transaction performance test shows the potential of VI and SAN for increasing transaction performance and efficiency. But the SAN version of the test does not take into account the legacy application interfaces required in order to run existing network applications, in particular the sockets API. Thus, we ran the next experiment using the CSP prototype with the full CSP stack as described in Figure 3. This included a lightweight SAN transport between the proxy and application node, as well as the socket filter that maps the SAN interfaces to the sockets API so that the Apache Web server application can be run unchanged. We then ran Apache over the SAN-based stack and the standard TCP/IP (LAN) stack. Figure 6 compares the results of this test.

As Figure 6 shows, the number of transactions per second that can be completed by the Apache Web server over the

SAN stack is significantly higher than the number that can be completed over the LAN TCP/IP stack. Also, the SAN stack generally reduces the amount of CPU time used per transaction. Yet, the efficiency is considerably lower than that of the simple transaction performance test, where the transactions were not constrained by the legacy sockets API.

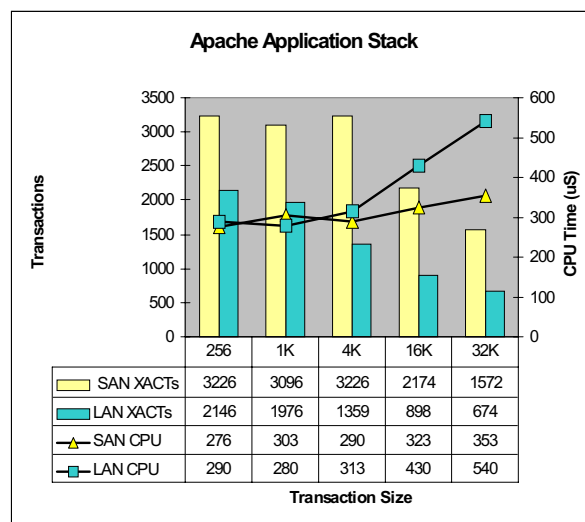


Figure 6: Apache Web server results

DISCUSSION

The Comm Services Platform (CSP) architecture, including the use of the proxy node to terminate TCP/IP sessions, allows the use of System Area Network (SAN) technology at the front-end of the data center. The principle behind the use of SAN is to spend more time processing client requests and doing useful work, as opposed to processing network protocols. The CSP prototype results show that this can be achieved to a large extent.

In the simple transaction test, the delta between the efficiency of the SAN versus the Local Area Network (LAN) is considerable. When the results are averaged, the SAN version of this simple test can process 2.3 times the number of transactions using 22% of the processing time per transaction. When the CSP application stack is completed, including the Apache Web server, the SAN transport, and the socket filter, the SAN version can process about twice the number of transactions using 87% of the processing time per transaction.

These results show a reasonable return on investment, but also show that when the solution stack is completed in order to accommodate legacy applications, the gain in CPU efficiency is reduced.

When developing the software to map the sockets interface to VIPL, there are two main factors that cause this reduction in efficiency.

The first performance-limiting factor is the *select* API that is used for synchronization in legacy applications. When mapping the sockets API to VIPL in our prototype implementation, use of the *select* API forces a kernel transition in order to check for kernel-level I/O activity. Generally, the *select* API causes significant CPU overhead in the common case because it searches the specified list of file descriptors for the presence of I/O completions on each of the selected client sessions. Conversely, the native form of synchronization for VI/SAN is the completion queue, which allows the application to poll or wait for I/O completions directly from user space without causing a kernel transition. If there is a message ready for processing, which is the common case during times of heavy communication, blocking calls and kernel transitions can be avoided altogether. The completion queue mechanism also avoids the searching of completions because completions are aggregated across a number of channels and are returned in the order of occurrence.

Another significant performance factor is the method used to map client sessions to SAN channels. SAN technologies based on the Virtual Interface (VI) Architecture are predominantly connection oriented. Each connection forms a bi-directional channel that can guarantee reliability and allows the hardware to directly multiplex and de-multiplex message traffic on behalf of the application. In our prototype implementation, we chose to map many client sessions to one SAN channel, the alternative being a one-to-one mapping. We did this because, to date, the number of channels implemented in existing SAN hardware is relatively small. This choice caused us to multiplex incoming request traffic in software. We used a thread/dispatch model where a single thread waits for client requests and dispatches them to other threads. This model forces a context switch for each request, thus causing significant overhead.

CONCLUSION

The Comm Services Platform (CSP) describes a distributed system architecture for enabling scalable Internet services. It shows that this can be achieved through the integration standard building blocks including programmable network processors, dense servers, and Virtual Interface (VI) Architecture-based System Area Networks (SANs). As a preeminent supplier of building blocks to the Internet economy, it is important for Intel to have a good understanding of how these building blocks are combined to build complete systems.

In particular, the CSP shows significant benefit to front-end applications. In our basic prototype, we have shown that we can improve the performance of HTTP processing by more than two times. Continued evolution of the server hardware platforms, communication APIs, and front-end server applications could result in additional performance and efficiency gains.

There are additional, second-order benefits that arise from the CSP architecture. To date, SAN technologies have largely been deployed at the back-end of the data center in order to construct high-performance parallel databases. Once external client communication is efficiently channeled into a SAN at the front-end, the mid-tier, back-end servers and storage within the data center can be joined by a seamless SAN environment. All of the tiers in the data center could then take advantage of additional high-performance communication mechanisms such as distributed objects, Remote Procedure Calls (RPC), and native VIPL in order to achieve very high-performance distributed computing and storage environments.

ACKNOWLEDGMENTS

The Comm Services Platform (CSP) project was a team effort conducted in Intel's Enterprise Architecture Lab with major contributions from Annie Foong, Gary McAlpine, Dave Minturn, Hemal Shah, and Rajesh Sankaran. Thanks to Justin Rattner for providing the initial support for the project.

REFERENCES

- [1] Virtual Interface Architecture Specification, <http://www.viarch.org>.
- [2] VI Architecture Implementation Guide, <http://developer.intel.com/design/servers/vi/>.
- [3] InfiniBandSM Trade Association, <http://www.infinibandta.org>.
- [4] Intel[©] Internet Exchange Architecture, <http://developer.intel.com/design/IXA>.
- [5] H. Shah, D. Minturn, A. Foong, G. McAlpine, R. Madukkarumukumana, and G. Regnier, "CSP: A Novel System Architecture for Scalable Internet and Communication Services," 3rd Usenix Symposium on Internet Technologies and Systems, March 2001. <http://www.usenix.org/events/usits01/>.
- [6] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, et. al., "The Virtual Interface Architecture," IEEE Micro, March/April 1998, pp. 66-76.

AUTHOR'S BIOGRAPHY

Greg Regnier is a Principal Engineer and manager of the I/O and Cluster Architecture group in Intel's Enterprise Architecture Lab. Greg's experience and interests include high-performance message passing, networking, and distributed computing. Greg has a BSCS ('79) from Saint Cloud State University in Minnesota. His e-mail address is greg.j.regnier@intel.com

Copyright © Intel Corporation 2001. This publication was downloaded from <http://developer.intel.com/>.

Legal notices at <http://developer.intel.com/sites/developer/tradmarx.htm>.