



## IronPort™ AsyncOS™

### The technology powering Messaging Gateway Appliances™

**Scaling email delivery infrastructure.** As the use of email becomes ubiquitous among both businesses and consumers, corporations are finding that email is a highly effective way to communicate with customers and prospects. Sales and marketing are clamoring to deliver an ever-increasing volume of email. Customer service and operational functions also rely heavily on email to confirm transactions and interface with customers. This extraordinary growth in outbound email communications is taxing existing email delivery infrastructures to the breaking point.

IT managers responding to this need are faced with performance and manageability issues. Current solutions require complex hardware and software configuration, trial and error setup, and the inevitable need to deploy multiple servers and then deal with the subsequent management issues. These limitations spring in large part from the inability of existing technology to address the unique scalability problems faced by Internet messaging gateways: concurrency-based communications bottlenecks and the limitations of file-based queuing.

IronPort Systems' AsyncOS is a new software architecture engineered from the ground up to address concurrency-based communications bottlenecks and the limitations of file-based queuing.

# IronPort™ AsyncOS™

## **IRONPORT AsyncOS™**

Traditional operating systems were designed to accommodate a multitude of general-purpose tasks. Internet messaging gateways have unique requirements that are unlike those of other applications, rendering general-purpose operating systems ill-suited to meet these requirements. By developing an integrated hardware/software appliance, IronPort controls the operating system, application and hardware, delivering solutions that general-purpose operating systems cannot.

IronPort Systems' AsyncOS is a new software architecture engineered from the ground up to address concurrency-based communications bottlenecks and the limitations of file-based queuing. This paper discusses two AsyncOS components, a high-concurrency threading model that addresses allocation of system resources and AsyncFS™, an Asynchronous File System optimized for message queuing.

## **CONCURRENCY-BASED CONNECTION BOTTLENECKS**

Traditionally, the I/O bottlenecks in server software development have been disk subsystems, databases, or connections to local servers. With the advent of high volume communications over the Internet, these bottlenecks now pale in comparison to the bottlenecks caused by connections to remote servers and clients.

A messaging gateway deployed in a corporate marketing environment must communicate with hundreds of thousands of remote messaging gateways in order to successfully deliver messages to millions of customers. These remote gateways may be on a high-speed backbone (e.g., aol.com), on a T1 or T3 (e.g., cisco.com), on a DSL line or even on a modem (e.g., homes or small businesses). Even worse are the multitudes of undeliverable emails in a large campaign that require multiple connections to remote gateways, only to end up with a bounced message without realizing any delivery throughput!

If the originating gateway connected to these remote gateways one at a time, overall message delivery throughput would be limited by the Internet connectivity of the slowest remote gateways. While delivering to a slow remote gateway, the originating gateway could experience an I/O bottleneck of 28.8kbps or worse, even though it has a 100Mb connection to the Internet! The only way to overcome this bottleneck is through multiple concurrent connections. 28.8kbps connections to slow remote gateways are not a problem when connecting to thousands of them concurrently. Unfortunately, each connection to a remote gateway requires a separate thread or process in the OS, and traditional software platforms were not designed with very high thread concurrency in mind. With the exception of the specialized computers that operate the circuit-switched telephone networks, high concurrency was rarely, if ever, a software requirement prior to the advent of Internet communications.

## **SOLUTION: HIGH CONCURRENCY THREADING MODEL**

The AsyncOS high-concurrency threading model intelligently allocates system resources to overcome the bottlenecks of Internet communications. Since thread or process concurrency ultimately becomes the I/O bottleneck in a messaging gateway, IronPort AsyncOS technology includes a radical new threading model, as well as a new task scheduling model. Stackless Threads™ support tens of thousands of Internet connections and I/O-Driven Scheduling™ optimally schedules service for each connection.

On a traditional platform, each thread is allocated a fixed and dedicated memory stack. Because handling stack overflow errors is very difficult, and because stack overflows are a common source of security holes, this stack allocation must be generous, and this can quickly consume a large amount of RAM. For example, 1000 threads, each with a 1MB stack, would consume 1 GB of RAM! AsyncOS Stackless Threads are unique because they are allocated memory only as needed, with no dedicated stack. This efficient memory utilization strategy not only enables an order of magnitude increase in concurrency, but it leaves more RAM available for caching file system data and eliminates the risk of security holes and system crashes from stack overflows.

AsyncOS I/O-Driven Scheduling uses a similar approach of only allocating system resources as needed. On a traditional platform, a preemptive multitasking operating system hands control of the system to each task for a finite time slice called a quantum, e.g., 1/100th of a second. The scheduler cycles through each task, ensuring that no task is starved and allocating a pre-determined time slice of the CPU. The traditional OS approach to

## IronPort™ AsyncOS™

task scheduling is very inefficient when solving the problem of Internet message delivery, a problem that is typically not bound by CPU processing. Each time the processor switches to a new task it incurs the penalty of a context switch. For a highly concurrent application such as message delivery, the performance penalty of excessive context switching is substantial.

I/O-Driven Scheduling dramatically reduces this penalty by scheduling tasks around the availability of TCP connections for reading or writing. When a TCP connection becomes available, I/O-Driven Scheduling immediately grants system resources to the associated task until such time as the connection is no longer capable of I/O. In this way, AsyncOS applies system resources to tasks that can use them and does not needlessly rotate through multiple tasks. This allows AsyncOS to achieve throughput levels in excess of ten times greater than messaging applications based on traditional operating systems. In addition, since the thread switches always happen at the completion of an I/O read or write, threads are never left hanging in the middle of an operation and the memory management in each thread is simplified, further increasing efficiency.

### FILE-BASED QUEUING BOTTLENECKS

Traditional messaging applications face a second architectural bottleneck when each message is allocated a separate and unique file that must be written, read and deleted. When receiving hundreds of thousands of messages per hour the overhead associated with managing these files rapidly becomes prohibitive. Some messaging programs have attempted to work around this problem by using heavily nested directory trees, but most have achieved only limited success. As the message queue grows and the number of files being managed increases, performance can grind to a halt.

Furthermore, traditional file systems were optimized to allow low-latency random access to data and to rapidly resume global data access in the event of system failure. To achieve this objective, these file systems store inodes, maps of pointers to each block of data within a file, on disk. Thus modifying any file involves multiple disk accesses to update the data as well as multiple disk accesses to update the inodes.

### SOLUTION: ASYNCFSTM, THE ASYNCHRONOUS FILE SYSTEM

AsyncFS is a revolutionary file system architecture with unique data structures optimized for asynchronous message delivery. First, messages are stored in batches, not in individual files, dramatically reducing the number of basic read/write operations. Second, each queue data structure, which governs the order of message delivery, efficiently doubles as an inode that maps where messages live on disk. Furthermore, these "inode queues" are stored in RAM instead of on disk, further reducing the number of read/write operations. The Asynchronous File System yields the ultimate file system accomplishment for Internet messaging: fast read/write performance that is independent of queue size.

Safety is not sacrificed in delivering this performance. In the event of a system disruption or failure that removes the inode queues from RAM, AsyncFS reads message data from disk as it rebuilds the inode queues in RAM and resumes message delivery. The traditional file system requirement of rapid random access to data does not apply to Internet messaging and the overhead associated with this burden is eliminated with AsyncFS. AsyncFS delivers breakthrough performance, yet maintains complete data integrity in the event of system failure.



**IronPort Systems, Inc.**  
1100 Grundy Lane, Suite 100  
San Bruno, California 94066  
**tel** 650.989.6500 **fax** 650.989.6543  
**email** info@ironport.com  
www.ironport.com

#### THE IRONPORT STORY

IronPort Systems is focused on one goal: to revolutionize Internet Messaging. We are developing a family of appliances called Messaging Gateway Appliances™ that offer breakthrough performance, unprecedented ease of use, and reduced total cost of ownership. The IronPort team has done it before: our technology stems from experience at Hotmail, eGroups, ListBot, and Yahoo!