X1: Commodity Clusters with Real-Time Reflexes

Tudor Marian, Mahesh Balakrishnan, Hakim Weatherspoon, Ken Birman
Cornell University
Ithaca, NY-14853

1 Introduction

An increasingly important class of applications requires high-speed online processing of massive quantities of real-time information — examples include sensor network monitoring, deep packet inspection, financial calculators, online gaming and mission-critical command-and-control. Commodity clusters are used as compute backends for such applications; however, the traditional tiered architectures used in these clusters are too slow and bulky to enable immediate processing of incoming real-time traffic.

For example, imagine a backend datacenter for a sensor network that needs to trigger alarms whenever it observes correlated values across multiple inputs — perhaps to observe changes at a physical location that has different sensors trained on it. Such a datacenter would have to scan multiple gigabits per second of traffic to catch correlations across packets. The partitioning techniques used to scale conventional Internet services on commodity datacenters would require incoming updates to traverse multiple levels of intermediate nodes before they can be checked for patterns, incurring seconds of overhead before the datacenter can respond to the real-time inputs.

X1 is a framework for building high-speed packet processors that can sift through extremely high rates of transient data. Internally, X1 is a modified Linux kernel optimized for high-speed traffic inspection, using a range of techniques that allows an inexpensive commodity blade-server to process packets at line-speed. Externally, X1 exposes programming abstractions that allow developers to easily build packet processing applications that can scale to multiple gigabits per second of traffic. Accordingly, X1 enables the construction of scalable clusters for processing real-time data from inexpensive commodity blade-servers.

In the sensor network example, the developer would instruct X1 to buffer packets whose headers satisfy a certain constraint (e.g., from sensors A, B and F) and subsequently check if the content in these packets fit additional constraints (e.g., temperature field greater than 30 degrees). X1 decomposes such queries into a workflow of modules, each of which has an associated buffer and can potentially generate additional packets once it accumulates enough input packets in its buffer.

In addition to enabling real-time clustered applications, we envision X1 as an extremely sensitive first-response layer that intervenes between high-speed networks and traditional datacenters. The possible applications for such a layer are numerous: X1 can aggregate packets and reduce the amount of traffic that hits the slower software layers within a datacenter. Also, it can be used to design and implement custom wide-area application and protocol accelerators; in fact, we have already built a TCP/IP accelerator called Maelstrom [1] and a filesystem device called SMFS [2] using the X1 architecture. Other uses include intrusion detection and traffic monitoring, discarding unwanted traffic and alerting systems infrastructure before nodes within the datacenter are affected.

Crucially, all these different applications can run on a single platform of X1 blades, eliminating the device sprawl that plagues modern datacenters. The presence of a single unified platform for high-speed device implementation allows for easy installation and maintenance of perimeter functionality.

Currently, we have a fully functional X1 implementation that's able to process a Gigabit per second of data on a cheap (less that 600$) machine with a 3 GHz processor and a 1 Gbps NIC. As mentioned, we have working implementations of protocol and filesystem accelerators written within the framework. Our current efforts are focused on the exact abstractions to provide developers with, as well as load-balancing techniques to spread incoming traffic across racks of X1 blades.

References

[1] M. Balakrishnan, T. Marian, K. Birman, H. Weatherspoon, and E. Vollset. Maelstrom: Transparent error correction for lambda networks. In *NSDI 2008: Fifth Usenix Symposium on Networked Systems Design and Implementation (To Appear)*, 2008.

[2] H. Weatherspoon, L. Ganesh, T. Marian, M. Balakrishnan, and K. Birman. Smoke and mirrors: Mirroring files over high-speed long-distance links without performance loss. In *Submission*, 2008.