An Infrastructure for Network Development. Proof of Concept: Fast UDP

Edgar A. León
Computer Science Department
University of New Mexico

Michal Ostrowski
IBM T. J. Watson Research
(Dated: March 1, 2005)

Scientific applications demand a tremendous amount of computational capabilities. In the last few years, computational power provided by clusters of workstations and SMPs has become popular as a cost-effective alternative to supercomputers. Nodes in these systems suffer from a variety of performance and scalability problems that may affect the applications running on them. Two of these problems are: (1) Host network processing scales poorly with respect to processor, bus and link bandwidths (this effect has become more evident as network speeds continue to increase); (2) Host overhead, due to communication processing, significantly reduces the processor availability to perform application's work.

In recent years, network vendors have created network interface controllers (NIC) which can be programmed and provide a significant amount of memory and computational resources. These controllers allow overlap of computation and communication by processing communication tasks on the NIC and computational tasks on the host processor(s).

Although processing on the NIC is beneficial to application's performance in some cases, it is not clear what the capabilities of these smart NICs should be and how they integrate with the rest of the system. The interactions between the Application, the Operating System (OS) and the Network are not well defined and should be studied further. These interactions are a key component to application scalability and performance.

To investigate these issues, we have created an infrastructure for network development. It is based on a functional model of a NIC which can run arbitrary functionality. As a proof of concept, we have used this infrastructure to implement a simple communication mechanism, *Fast UDP*, that improves application performance by carefully applying three network optimizations. Thus, showing that this infrastructure allow us to investigate network optimizations and NIC architectures that match current and future application's demands.

In the first part of this work we describe the details of our network infrastructure. This infrastructure has been implemented in the *Mambo* architecture simulator. Although this system is not open source, we have created a *shim layer*, that allows the creation and dynamic loading of smart network devices without access to Mambo source. Thus, our infrastructure can be used by any institution with access to Mambo binaries. An important component of the infrastructure is defining the functional abstraction the NIC provides to upper layers of software running on the NIC. We have defined and implemented a NIC API. By explicitly defining this API, NIC independent code can be developed and potentially ran in any NIC that implements this interface. Our NIC model also exposes a number of registers that can be memory mapped to allow communication to it.

The second part focuses on describing Fast UDP. This system is composed by code running on the NIC and code running in the Operating System on the host. The operating system we have used is K42, an open source, high-performance OS. Fast UDP is a communication mechanism with UDP matching semantics. In fact, the UDP stack in the host's kernel is unmodified. When a UDP packet arrives from the network, the NIC matches the packet using its destination port, and if a user has posted a received for that port, it will be delivered to the user buffer. The NIC *splinters* the payload from the header, moving the payload directly into application space while control information (header) goes to the kernel. Thus, the kernel remains aware of network processing, but application's data processing bypasses the kernel.

Using a simple UDP application we obtain a 5% performance gain even when the application spends 80% of its time computing. This performance improvement increases as the application's communication time increases.

In conclusion, we have created a network infrastructure that allow us to:

1. Better understand recent and future network architectures to fully take advantage of their capabilities;

2. Make a case for optimizations that improve application performance and provide arguments for those ones who do not.

3. Better understand the interactions between the Operating System, Applications and smart NICs to avoid bottlenecks in the data path from the network all the way to the application.

As a proof of concept, we have applied 3 network optimizations techniques to improve application performance: *matching on the NIC*, *NIC offloading* and *splintering of control and data*. We obtained significant performance improvements even for a computation-bound application.