Title: ???

Abstract:

In the best of cases, server I/O comprises a complex dance between an application and the underlying network across a well-defined interface.  Until now, the common practice has been for coders to take the I/O interfaces as a given and to code to them at some cost in terms of application performance. This BoF brings application developers together with the open source networking community to discuss whether application I/O can be improved through a framework of interfaces allowing developers to achieve significant improvements in I/O efficiency with minimal effort.  The outcome will provide direction to drive forward the development of open source I/O software.

Long Description:

For many classes of applications, I/O performance turns out to be a key determinant of application performance. Historically, when accessing I/O services, the common practice among application developers has been to code to one of a number of well-known interfaces, taking those interfaces as a given and coding to them using well-established conventions. The cost of doing so in many cases has been suboptimal application performance.

The goal for this BoF is to engage application developers in a conversation with network developers and providers to explore proposals for improving application performance through more streamlined access to I/O services.

Historically, the architecture of network protocols began with a “platform-centric” view. That is to say, network protocol designers have tended to focus on the transport of bytes of data across the wire from one server platform to the other, leaving it as a responsibility of the server platform to deliver those bytes to the application using one of these well-known interfaces. In some sense, the design of the I/O interface was driven by the characteristics of the underlying network, but not necessarily by the needs of the application.

The emergence of RDMA in the early 2000s marked the beginning of a different approach, one that began with examining the entire I/O path from one application to another, including the I/O interface, the network stack and the underlying network itself. In the case of RDMA, its protocol designers began their work by focusing first on the delivery of messages to an application. This approach yielded the now-familiar RDMA and SEND/RECEIVE protocols that are the basis for modern RDMA networks, the result of which has been an architecture capable of delivering industry-leading performance and efficiency. The OpenFabrics Alliance, in turn, has taken a leadership role in defining the entire I/O infrastructure, including APIs, to allow applications to take direct advantage of these improvements.

This BoF takes the next logical step in the improvement of application performance by starting a discussion between application developers and network designers leading to the development of an I/O framework that will give their applications access to a broad range of I/O services, including traditional RDMA operations, using an interface architecture that is familiar to the application developer. The BoF will discuss proposals first suggested within the OpenFabrics Alliance to define a broad I/O framework. This framework will allow a network provider to describe the services it provides and will allow an application developer to specify which of those services he wishes to use for his particular application via a set of well-defined interfaces or APIs.

The results of this discussion will be captured either as Powerpoint slides or as written text and will be actively discussed within the framework of the OpenFabrics Alliance. In the best traditions of open source software development, we hope that application developers will join the ongoing discussion in the OFA as full partners to help drive I/O development forward.

in application development has been to take the I/O interfaces as a given and to code to them, at some cost to the application in terms of performance.

In the best of cases, server I/O has always consisted of a complex dance between the Application, the Server Platform and the underlying Network. Historically, this dance has placed at least some burden on applications to adapt to I/O interfaces and protocols provided by the network. This BoF brings application developers together with providers of open source networking technology to look at I/O from the application perspective. By focusing on I/O interfaces in this way it should be possible to evolve I/O technology toward higher performance and broader market reach. The outcome will provide direction to drive the OpenFabrics Alliance going forward.

In the best of cases, server I/O has always consisted of a complex dance between the Application, the Server Platform and the underlying Network. Historically, this dance has placed at least some burden on applications to adapt to I/O interfaces and protocols provided by the network. This BoF brings application developers together with providers of open source networking technology to look at I/O from the application perspective. By focusing on I/O interfaces in this way it should be possible to evolve I/O technology toward higher performance and broader market reach. The outcome will provide direction to drive the OpenFabrics Alliance going forward.

The next logical step is to a

In other words, the interfaces used by applications to access I/O services have in some part been driven from the bottom up based on the protocols supported by the network.

In the best of cases, server I/O has always consisted of a complex dance between the Application, the Server Platform and the underlying Network. Historically, this dance has placed at least some burden on applications to adapt to I/O interfaces and protocols provided by the network. This BoF brings application developers together with providers of open source networking technology to look at I/O from the application perspective. By focusing on I/O interfaces in this way it should be possible to morph I/O technology toward higher performance. The outcome will provide direction to drive the OpenFabrics Alliance going forward.

In the best of cases, server I/O has always consisted of a complex dance between the Application, the Server Platform and the underlying Network. Historically, this dance has placed at least some burden on applications to adapt to I/O interfaces and protocols provided by the underlying network. This BoF brings application developers together with providers of networking technology with the ultimate goal of improving application performance by looking at I/O from the perspective of the application. By focusing on I/O interfaces in this way it should be possible to morph open source I/O technology toward higher performance.

This BoF is designed to facilitate a much closer working relationship between server application writers and the developers and providers of networking technology. The ultimate goal is to improve application performance by defining “application-centric” I/O interfaces. These interfaces, in turn, can be used to drive the development of the network stack and the underlying I/O protocols.

Server I/O when viewed at the application layer has always consisted of a complex dance conducted between the Server Application, the Server Platform (including the OS), and the underlying Interconnect. This has historically required This BoF is designed to facilitate a much closer working relationship between the users of Server Applications and the developers and providers of networking technology.

Historically, network architecture has begun with a “platform-centric” view. That is to say, network and protocol designers tended to focus on the delivery of bytes of data to the server platform. Once delivered to the platform, it becomes the server’s responsibility to ensure delivery of the bytes to the user application.

The emergence of RDMA technologies signified the beginning of a different approach to network architecture, beginning with an “application-centric” view. In other words, network and protocol designers begin their work by focusing first on the delivery of messages to an application. This approach yielded the now-familiar constructs that are the basis for modern RDMA technology.

Title - InfiniBand and RDMA applied to Next Generation Systems?

Abstract

The notion of direct communication between remote applications is well-established in the industry. Such “direct” communications are facilitated by a networking technology known as Remote Direct Memory Access, or RDMA. The best known and most widely available implementation of RDMA is the InfiniBand Architecture.

The emergence of the InfiniBand Architecture in 2000 signaled a change in the way that applications communicate, both to exchange small messages and to share large data sets. By eliminating the OS as a central character in the network path, the exchange of messages and data became significantly easier and faster while simultaneously reducing the burden on compute and network resources.

The InfiniBand Architecture has served the industry well for nearly 14 years, but there have been changes both in the underlying compute model for servers and clusters and in the underlying interconnect technology. For example, the scale of large clusters has changed dramatically since the introduction of InfiniBand, so what was once considered a truly scalable architecture may now be limited in its scalability. Another example is the emergence of “co-processing” and the emergence of new processor architectures now being deployed in server applications. These changes and others like them should cause us to re-think how RDMA is implemented. For example are there opportunities to:

* Improve the scalability of the RDMA architecture allowing it to meet the needs of emerging exascale systems?
* Improve performance, both at large and small clusters?
* Improve on its support for both small and large message transfers or information exchanges?
* Reduce its already low impact on the end node by, for example, reducing the memory footprint required to implement InfiniBand?
* Improve InfiniBand’s alignment with modern and emerging compute models, such as virtualized servers, massive multicores and the trend toward so-called dense servers (sometimes called microservers)?
* And last but not least, to better align the architecture with emerging memory models, such as PGAS languages?

The goal for this BoF is to discover the areas where it makes the most sense in terms of return on investment to consider improvements to the tried-and-true RDMA architecture.