GNI Provider Iovector Support for Libfabric

Author: Evan Harvey (New Mexico Institute of Mining and Technology)

Mentors: Sung-Eun Choi (Cray), Zachary Tiffany (Cray), James Swaro (Cray), and Howard Pritchard (Los Alamos National Laboratory)

What is Libfabric?

Libfabric is a new library that provides a network API which may span many networks. The 'fi_getinfo' function returns information about the available fabric services for reaching a specific node or service and is essential for portability. In order to achieve such a general API the designers took input from several middleware developers and vendors. Hardware providers change frequently to meet performance demands, as such, new network APIs are required; Libfabric provides a facade to the underlying network APIs so that application programmers may port, develop, and test code for new systems, using the Libfabric API, while the Libfabric developers program the operations needed to utilize the new networking hardware.

Motivation for Libfabric

As old HPC systems are retired and new systems are deployed, there is an ongoing effort within the HPC community to prepare for future systems that haven't even been implemented. Many existing systems use the user-level Generic Network Interface (uGNI), verbs, or other popular network APIs. Libfabric is a research project currently aimed at preparing for future systems by comparing performance of applications on existing systems with and without Libfabric as the underlying network API.

What is the GNI provider?

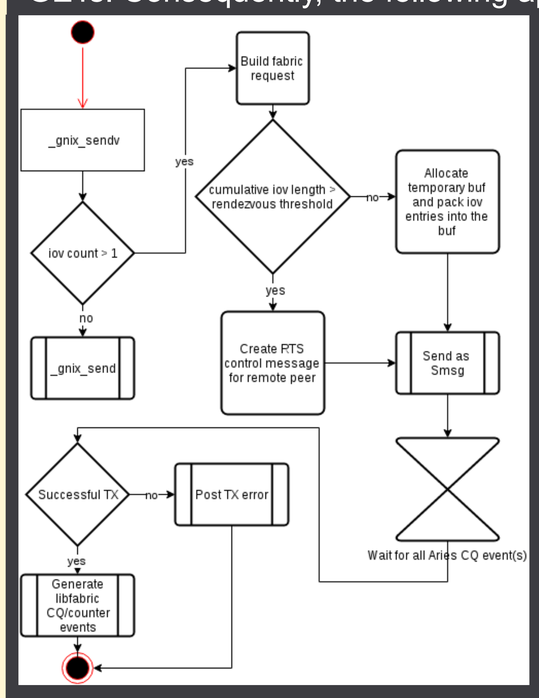
The Generic Network Interface (GNI) provider is one of Libfabric's eight hardware providers. The GNI provider uses the uGNI API to utilize the Aries NIC. The uGNI API is one of the few interfaces that does one sided operations and the Aries NIC has a theoretical maximum bandwidth ~10 GB/s for user data via Remote Direct Memory Access (RDMA), Fast Memory Access (FMA), or FMA Short Messaging (SMSG) interfaces. One notable feature of uGNI and the Aries NIC is that it can perform FMA PUTs/GETs without causing a context switch to kernel space.

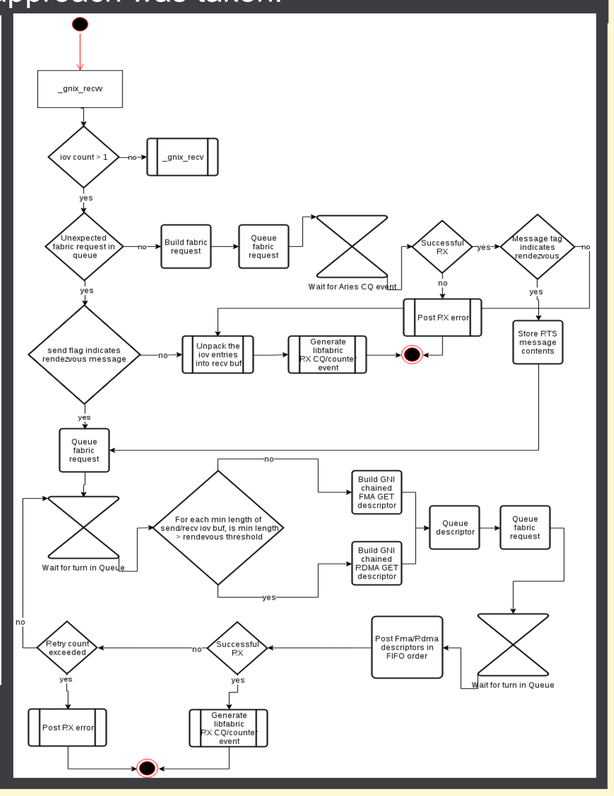
Motivation for this project

The next generation MPICH ch4 device targeting the Libfabric API is currently under development. MPICH sits on top of the Libfabric API and the ch4 netmod will be used by Intel, Cray and others to target next generation networks. Since any Libfabric provider may not fully support the Libfabric API on a given platform, there is a general active-message version of all the MPICH functions; MPICH active-message functions require “scatter-gather” lists to augment the provider's missing API functionality. Therefore, to enable testing of MPICH ch4 at scale using today's Aries-based systems, the GNI provider should support an iovector limit of at least 2.

Implementation of GNI provider Iovector support

uGNI performance results for data transfers using the FMA and RDMA routines show that chain lengths greater than one and individual messages 4K or less in size perform better using chained FMA GETs rather than RDMA GETs. Consequently, the following approach was taken.





Test environment

A Cray XC40 and XC30 were used for running experiments. The Cray XC40 ran CLE 6.1.0 UP02 and had the following specifications:

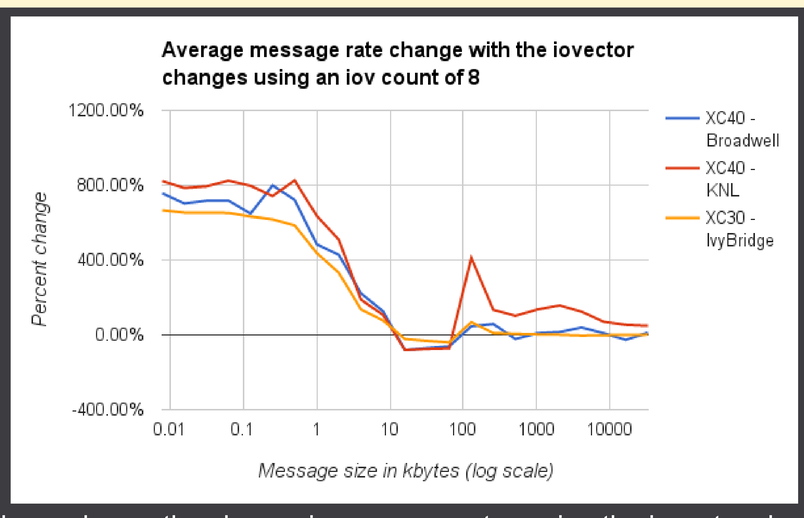
Type	Aries	Processors	Cores	Threads	Processor	Memory
HPDC	1.1	2	28	56	2.6 GHz Broadwell B0	128 GB DDR4-2400
KPDC	1.1	1	64	256	1.3 GHz Knights Landing B0	96 GB DDR4-2400

The XC30 ran CLE 5.2 UP04 and had the following specifications:

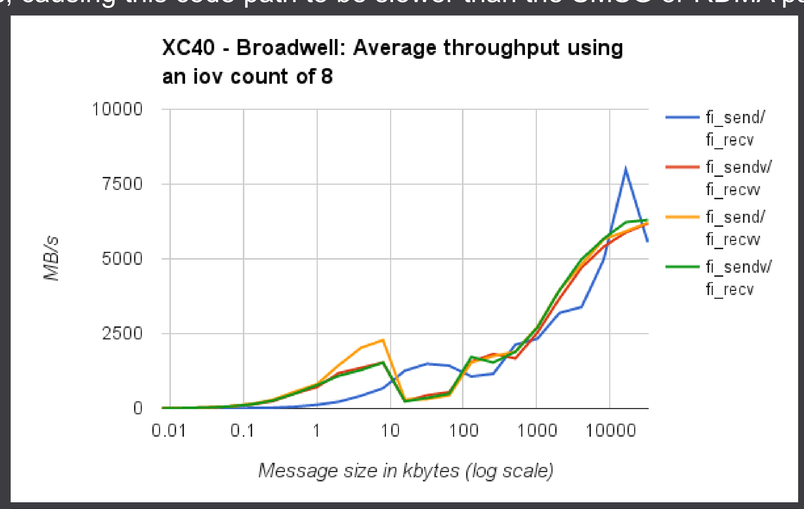
Test environment continued

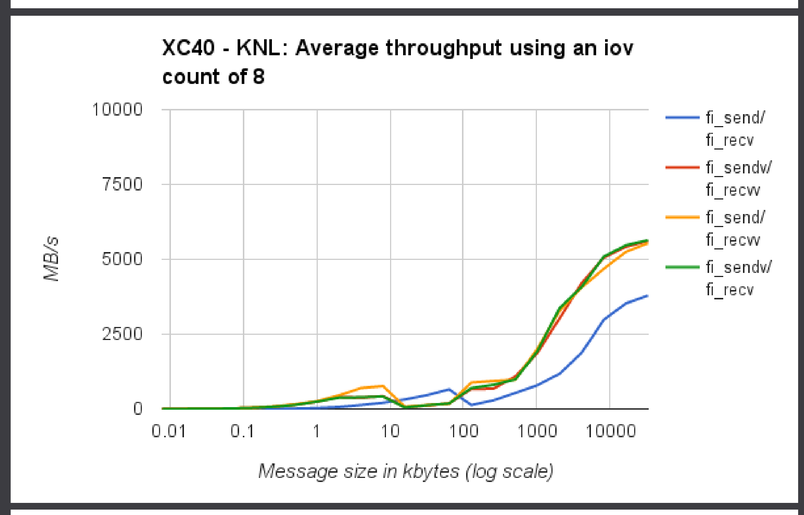
Type	Aries	Processors	Cores	Threads	Processor	Memory
QPDC	1.1	2	24	48	2.7 GHz Ivy Bridge C0	128 GB DDR3-1866

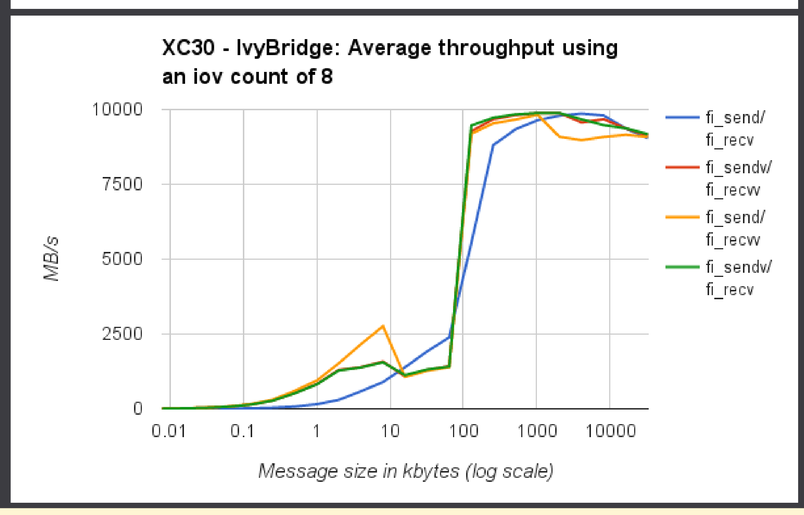
Eight samples from the tests were collected on two Broadwell nodes and two KNL nodes as well as ten samples from two Ivy Bridge nodes. Each test instance performed 10 warm-up iterations followed by 100 timed iterations to calculate the message rate and average throughput.



The graph above shows the change in message rates using the iovector changes instead of iteratively calling 'fi_send'/'fi_recv'. Also note that between X-axis values 14 and 16 the FMA mechanism is used and it dynamically allocates gni FMA descriptors using malloc, causing this code path to be slower than the SMSG or RDMA paths.







Conclusions

These results show that a portable network API implementation such as Libfabric does deliver good network performance and that good performance can be obtained from the 'fi_sendv' and 'fi_recvv' interfaces which do not map directly to a hardware feature. The significance of this API addition to the Libfabric GNI provider will not be realized until MPICH ch4 has been developed with the Libfabric API and tested on Aries-based systems. Future work includes improving the dynamic allocations for GNI provider's ”scatter-gather” SMSG and FMA mechanisms and profiling the 'fi_sendv'/'fi_recvv' API addition in order to improve the performance of the new code paths.

For more information contact evan.harvey@student.nmt.edu.