Streaming Programming Systems at Exascale

Stefano Markidis, Ivy Bo Peng and Erwin Laure
KTH Royal Institute of Technology

The emerging research area of data-intensive applications and big data analytics within the HPC community promotes the streaming computational paradigm with growing relevance. With the exascale era approaching, it is important to understand which streaming programming system is the most effective for the development of streaming applications on exascale supercomputers.

In this white paper, we analyze the uptake of the streaming computational paradigm on exascale supercomputers. We argue that MPI is likely to be the dominant programming system at exascale and the development of streaming programming systems requires an MPI streaming model. The support in MPI for a streaming model helps promote the adoption of the streaming computational paradigm in the HPC community. We discuss the current state of MPI streaming model and propose a stream benchmark to investigate the performance of streaming programming systems and HPC systems.

Streaming Programming Systems at Exascale. Exascale supercomputers will deliver 10^{18} floating-point operations per second (FLOPS) in the High-Performance Linpack (HPL) benchmark that solves a dense linear system. Despite the growing importance of big data problems in HPC community, the hardware and software for exascale supercomputers are still designed for compute-intense applications, such as HPL. Challenges remain in achieving maximum performance for data-centric streaming computing using hardware and software that are mainly designed for compute-intensive tasks.

Existing popular streaming software frameworks, such as Apache Spark and Flink, are designed for maximizing productivity and programmability. Such frameworks are intended for deployment in loosely coupled computing environment, e.g. cloud systems. On the other hand, MPI has been designed for strongly coupled systems with high performance interconnection networks, e.g. supercomputers. The main strengths of MPI have been portability and performance. As a matter of fact, the first exascale application will be an MPI-based HPL application. For this reason, MPI will be the first programming system that breaks the exascale barrier. Only very recently, the HPC community started investigating the possibility of including streaming model in MPI. An initial MPI library to support streaming computing paradigms has been developed as proof-of-concept. The use of such library enables streaming computing in traditional HPC applications.

We note that the streaming framework built on the top of MPI can achieve high productivity. The application developer can then decide whether to use MPI directly or higher-level frameworks for streaming computing based on MPI streaming model.

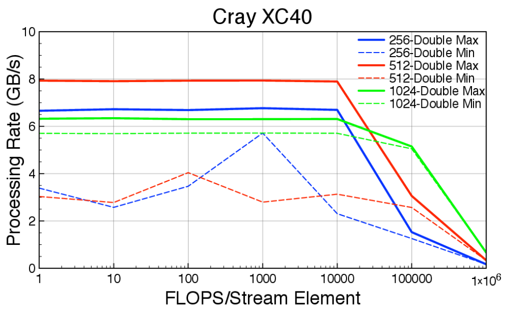
Uptake of Streaming Models in HPC. The vast majority of HPC applications uses MPI as the main parallel programming system. For this reason, it is likely that an HPC application developer that is interested in using streaming computing will investigate the support in MPI. By supporting a streaming model in MPI, the interoperability of the existing HPC application with new MPI streaming processes is guaranteed.

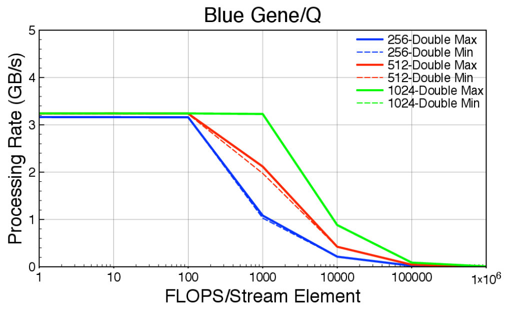
MPI promoted a widespread use among community and created a very large user-base through the presence of MPI Forum and a standardization process. Support in MPI for streaming model would help boost the use of streaming computing paradigm within the HPC user community.

High Performance Implementation of Streaming Models in MPI. A high performance implementation of MPI streaming model requires a native MPI implementation. The long standardization process will likely take more than five years for the MPI streaming model to be added to the MPI standard. The new streaming functions would be a major addition to the MPI standard, making this work comparable to the work done for one-sided communication in the second main release of MPI standard.

Meanwhile, it is possible to have high quality and high performance implementations of MPI streaming library, built on the top of MPI. Support for active messages and active access to data could lead to the development of such high performance streaming libraries.

The Need of Benchmark. There is a need for a benchmark to investigate the performance of streaming programming systems and HPC systems. We propose to adopt a benchmark in similar spirit as the widely used STREAM benchmark. The original STREAM benchmark calculates the sustainable memory bandwidth using four computational kernels: *copy(), scale(), sum()* and *triad()*. We suggest that a benchmark with the same four kernel operations can be used to calculate the amount of data being processed in time (processing rate) by using a given streaming programming system on a given platform. The Figure on the right shows an example of using such a benchmark with *scale()* kernel to determine the performance of a MPI streaming library on 64 cores on Cray XC40 and Blue Gene/Q architectures.


Cray XC40
Processing Rate (GB/s)
FLOPS/Stream Element
256-Double Max
256-Double Min
512-Double Max
512-Double Min
1024-Double Max
1024-Double Min


Blue Gene/Q
Processing Rate (GB/s)
FLOPS/Stream Element
256-Double Max
256-Double Min
512-Double Max
512-Double Min
1024-Double Max
1024-Double Min