NAME: George Michelogiannakis
AFFILIATION: Lawrence Berkeley National Laboratory
EMAIL: mihelog@lbl.gov
EXPERTISE: Networking, computer architecture, HPC

KEY DOE NETWORK/TRANSPORT CHALLENGE

Due to the rapid increase in parallelism necessary to meet performance goals combined with cost and manufacturing constraints, networks can easily become the bottleneck for a large class of applications important to HPC. The current state of the art emphasizes abstraction and hiding complexity from the software and programmer, at the expense of missed optimization opportunities. For instance, applications and optimization methods such as communication avoiding algorithms operate agnostically from the underlying network architecture and without knowledge of which communication events are expensive. Thus they cannot prioritize optimizing certain communications instead of others. Also, task and data placement methods have no means to reconfigure the network for better placement interoperability, and have little knowledge of the network's dynamic state such as congestion. Collective (e.g., multicast) communication is a common weak point, despite modern networks providing hardware support. This is partly because those mechanisms do not usually map adequately to MPI runtime collectives, and because the network has no time to prepare because it has no knowledge of impending collectives. As a final example, because the configuration and dynamic state of the network is unknown, the software is unable to accurately predict the expected cost and performance of a given communication event. *The strict and simplistic abstraction layers in current machines and APIs hide away numerous optimization and interoperability opportunities.*

DOE RESEARCH CHALLENGE DIRECTION

We propose the community to extend SDN functionality to intra-machine HPC networks, and expose the expanded functionality and information that this brings to the software layer via an easy and intuitive API. Through this API, we will provide an efficient way to expose the network configuration and state. Therefore, applications will be made aware of static information such as the distance between two endpoints, as well as dynamic information such as latency and power consumed for each message as it is being sent. In addition, we can make prediction models that take network configuration and dynamic state into account to improve task placement (e.g., avoid congested areas), as well as better guide application optimizations. For instance, programmers can now optimize to reduce *expensive* communication, instead of all communication that is done today. Via breaking down existing abstractions, applications will also be able to *warn* the network in advance of impending significant communication (e.g., collectives) in order to allow for network reconfiguration to match the expected communication pattern. This also opens up further opportunities such as passing to the network software-level knowledge, such as packet criticality that is a measure of a packet's projected impact to application performance if it is delayed.

We propose to adopt the necessary SDN principles in HPC because SDNs can efficiently provide the functionality and flexibility to solve the above problems and provide the necessary functionality the API requires. This direction requires an overhaul of current HPC networking. An added benefit of this is better integration with wide-area networks, which are often SDNs. Therefore, we can extend service guarantees that currently reside only in the inter-machine SDN through the machine boundary.