"Evolutionary Support for Revolutionary Programming Models and Runtime Systems"

Evolutionary Support for Revolutionary Programming Model

Runtime Systems

Abstract:

Various parallel programming models and runtime systems have been developed over the years to revolutionize the programming capabilities on large-scale parallel platforms. Each model comes with its own unique capabilities and advantages. So what is holding back applications from moving to these new models? One of the primary disadvantages of these "revolutionary" programming models is that, well, they are "revolutionary". That is, applications have to move as a whole to the new model. There is no piece-wise migration or transition path for applications to slowly move to newer models. The runtime systems of one model, for example, is unaware of the runtime systems of other models and therefore may not interoperate well if used together. As a result, combining multiple programming models at present is often not possible or may lead to deadlocks, unpredictable behavior, or suboptimal resource usage and performance loss.

Arguably, the only way to allow applications to start using these new "revolutionary" models is make them more "evolutionary". That is, their runtime systems should understand each other to allow an application to move some parts to newer models while retaining code that used older models. For example, applications written with MPI may need to use some threading model for exploiting the shared memory within a node more effectively; multimodule applications written in Unified Parallel C (UPC) or Coarray Fortran (CAF) may need to use math libraries written in MPI, such as PETSc, that have had hundreds of programmer-years of development invested in them; and applications written with Global Arrays may need to use load-balancing tools written in Charm++. The need to combine multiple models has also become evident in recent years when trying to scale applications in nuclear physics (GFMC), chemistry (NWChem), climate, nuclear reactor design, biology, and other multiphysics applications.

In this talk, I will describe some of the challenges associated with runtime systems where assumptions made so far with respect to the programming models they support are no longer true as we move to

exascale. The talk will focus on where current runtime systems are, what they are missing, and what we as a community have to focus on to get this working in the exascale timeframe.