Mapping Applications on Irregular Allocations

Seonmyeong Bak, Nikhil Jain(Mentor), Laxmikant V. Kale(Advisor)

Abstract

- Mapping applications on clusters becomes more difficult as the number of nodes become larger
- Supercomputers assigns allocations with irregular shapes to users to maximize the utilization of resources
- Much more difficult to map applications on these irregular allocations
- We extended Rubik, a python based framework to map applications on irregular allocations with a few lines of python codes
  - Rubik was originally designed for regular allocations, so we added features to handle allocations with irregular structure and unavailable nodes and two mapping algorithms such as row-ordering and recursive splitting
- We evaluate our work with two widely used HPC applications on Blue Waters: MILC and Qbox
  - We reduce execution time by 32.5% in MILC and by 36.3% in Qbox, and communication time by 60% in MILC and 56% in Qbox

Rubik, a python framework for structured communication

- Rubik is a python based framework developed at LLNL for mapping applications with structured communication onto regular allocation[1]
- Rubik facilitates mapping of user application using a few lines of python code
  - Users can easily generate several different mappings for their applications
- Rubik supports many types of operations for better mapping of application grids onto network grid
  - Projecting Algorithm
    - Row ordering
      - Place MPI ranks in the order of each axis in an allocation
      - Suitable for allocations that are mostly regular
    - Recursive splitting
      - Split virtual network grid and an allocation into subcuboids
      - Suitable for irregular allocations

Limitation of Rubik for irregular allocations

- Rubik is designed for mapping applications onto regular and symmetrical allocations
  - However, in many cases, the shapes of the allocations are irregular as the Blue Waters Grid
  - Motivation of this work: how to enable use of Rubik on irregular allocations for its broader applicability

Experimental Results

- MILC
  - 32.5% improved in execution
  - 59.3% improved in communication
  - With many number of cores, MILC spent more time on communication
    - This work minimizes hops between cores so p2p operations are more improved than collective operations
- Qbox
  - 36.3% improved in execution
  - 59.3% improved in communication
  - Qbox doesn’t call p2p routines directly. Instead, it uses SCALAPACK for p2p communication between ranks
    - This random shape of the virtual network grid can increase hops between cores by inefficient placement of nodes

Optimizations

- The direction of splitting
  - Calculate bisection bandwidth in each splitting and choose the direction where the bisection BW is lowest
  - To maximize the bisection bandwidth between closest subcuboids
- The shape of the virtual network grid
  - Estimate the shape of the virtual network grid with the shape of the allocation
  - Factorize the number of tasks and use factors by this factorization for the estimation

References