UPC Queues for Scalable Graph Traversals: Design and Evaluation on InfiniBand Clusters

Jithin Jose   Sreeram Potluri   Miao Luo   Sayantan Sur   Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University
{jose, potluri, luom, surs, panda}@cse.ohio-state.edu

1. Abstract

PGAS languages like UPC are growing in popularity because of their ability to provide shared memory programming model over distributed memory machines. While this abstraction provides better programmability, some of the applications require mutual exclusion when operating on shared data. Locks are a common way to achieve mutual exclusion in shared memory algorithms. However, they impose a huge performance penalty on distributed memory machines and have been shown to be one of the major scaling bottlenecks. Simplistic approaches to eliminate locks by replicating resources is inherently non-scalable due to high memory cost.

In this paper, we introduce a UPC library that provides the abstraction of Queues. Our UPC library is tightly integrated with underlying UPC Runtime and utilizes Active Messages. The implementation of Active Messages provides implicit mutual exclusion that we exploit to design Queues. We present the design and implementation of Queues in UPC and compare their performance with that of existing mechanisms for operating on shared data with mutual exclusion. We evaluate our approach by re-designing two popular graph benchmarks: Graph500 and Unbalanced Tree Search (UTS), using Queues in UPC. Experimental results indicate that queue-based implementation for Graph500 outperforms the replication-based implementation by around 44% and 30% for 512 and 1,024 UPC-threads, respectively. Performance improvements of queue-based version of Unbalanced Tree Search (UTS) benchmark over the current version are about 14% and 10% for similar scale runs, respectively. Our work is based on the Berkeley UPC Runtime and the Unified Communication Runtime (UCR) for UPC and MPI, developed at The Ohio State University.