Performance of Multicore Systems on Parallel Datamining Services

Xiaohong Qiu
xqiu@indiana.edu
Research Computing UITS
Indiana University Bloomington

Geoffrey C. Fox, Huapeng Yuan, Seung-Hee Bae
gcf@indiana.edu yuanh@indiana.edu sebae@indiana.edu
Community Grids Laboratory
Indiana University Bloomington

George Chrysanthakopoulos, Henrik Frystyk Nielsen
georgioc@microsoft.com henrikn@microsoft.com
Microsoft Research
Redmond WA

Abstract

Multicore systems are of growing importance and 64-128 cores can be expected in a few years. We expect datamining to be an important application class of general importance and are developing such scalable parallel algorithms for managed code (C#) on Windows. We present a performance analysis that compares MPI and a new messaging runtime library CCR (Concurrency and Coordination Runtime) with Windows and Linux and using both threads and processes. We investigate effects of cache lines and memory bandwidth and fluctuations of run times of loosely synchronized threads. We give results on message latency and bandwidth for two processor multicore systems based on AMD and Intel architectures with a total of four and eight cores. Generating up to a million messages per second on a single PC, we find on an Intel dual quadcore system, latencies from 5µs in basic asynchronous threading to 20 µs for a full MPI_SENDRECV exchange with all threads (one per core) sending and receiving 2 messages at a traditional MPI style loosely synchronous rendezvous. We compare our C# results with C using MPICH2 and Nemesis and Java with both mpiJava and MPJ Express. We are packaging our core algorithms not as traditional libraries but as services and use DSS (Decentralized System Services built on CCR) to compose workflows or mashups for complete applications. We show initial results from GIS and Cheminformatics clustering problems. Our results suggest that the service composition model and Windows/C# thread programming model will be a flexible parallel programming environment for important commodity applications. C.

1. Introduction