CSC6740 BSP vs. MPI

Rob Gillen

Introduction

Designing algorithms to effectively leverage parallel hardware platforms is a challenge that has existed for over 30 years. Designing these algorithms such that they are generalized for a broad range of parallel systems is an even greater challenge. Today's multi-core systems, hybrid architectures, and customized interconnects makes this challenge even greater. Concurrent with the advent of parallel systems has been the desire to model these systems such that algorithms could be clearly expressed in a generic form with respect to generalized platforms. This short paper briefly introduces two approaches that are legends in the field and then offers a simple comparison and summary comments.

BSP Model

Initially introduced by Leslie Valiant in his 1990 paper titled *A Bridging Model for Parallel Computation*, the bulk-synchronous parallel (BSP) model has exterted a significant influence over the way that researchers since have considered parallel computing. BSP attempts to provide a strong mathematical framework for considering the various aspects of parallel systems while not over-complicating the approach. There are three primary properties of the model: the number of components that each are a processor and/or a memory store, the router which is used to deliver messages between components, and the ability to synchronize all components after some determined units of time. This last property is utilized to support the notion of supersteps - a collection of operations that the algorithm designer assigns to the various nodes and that each node can complete without needing to communicate with other nodes. Every so many time units (referred to as L), the system checks to ensure that all components have finished their activities and sent any messages to the router. If so, the next superstep begins. If not, another L time units pass and the check repeats. At the beginning of this next superstep, all communications that were staged at the end of the prior are now available. This process of ensuring all processors are finished before moving to the next step is a form of global barrier synchronization. Valiant introduces a handful of variables to support his model: p - the number of processors, g - the bandwidth between processors, and L - the periodicity parameter (the number of time units per block). Valiant's ultimate goal (based on this paper) is for BSP to become the standard model for expressing parallel algorithims and hardware in a fashion similar to the von Neumann model is for serial problems.

MPI "Model"

The Message Passing Interface (MPI) is not a parallel computing model in the same way as BSP is, but can, in some ways, be thought of as such. Formally, MPI is a standardized means of communicating between various processors (virtual, cores, or otherwise). Specifically, it is a specification that can (and has been) implemented in a number of different languages and platforms, each with their own unqique flavor (e.g. `mpich`, `openmpi`, and others). MPI defines boundaries of communication (`MPI_WORLD`, groups, etc.) and provides constructs for messages passing in a wide variety of fashions wihtin those communication boundaries such as point-to-point and broadcast. Additionally, the MPI specification defines a number of fixed operations (reduction operators) that are so common in parallel algorithm design that they warrant being codified so as to avoid the need for each programmer to implement their own version of them (e.g. `MPI_SUM`, `MPI_PROD`, `MPI_BXOR`). MPI

defines a number of programmer-niceties such as providing each instance of the process it's rank ID, as well as the total number of processors in the MPI world - these values are helpful for dynamic data partitioning (e.g. process 2 will handle one half, one third, one fourth of the total problem as appropriate).

Comparison

Comparing BSP and MPI is a little like comparing apples and oranges - one is a language-agnositic model (way of thinking of the problem) and the other is a specification for the implementation of libraries supporting the development of parallel codes. That aside, there are some points of difference that can be evaluated.

The single largest difference is the way that messages are passed between processors. In the BSP scenario, all messages are staged in the router but none of them are delivered (available for the recipient to use) until the next superstep (following a global barrier sync, after some whole-multiple of L time units). In MPI, while contstructs exist for global synchronization, they are generally avoided. Rather the `MPI_Send` and `MPI_Receive` operations can occur at any time and do not require synchronization beyond the two communicating nodes and the delivery assurances requested. This would allow `P1` to send his results to `P0` and begin working on the next set of work without having to wait for `P3` to finish his first set of work. In the BSP paper, there is a suggestion that sub-global barriers (communication groups) may be possible, but no formal definition or model is provided for such.

BSP, on the other hand, provides a solid theoretical foundation on which to consider the algorithim being designed. It provides a means of modeling the processors, memory, and communication costs that aren't expressely modeled in MPI. Using this model, the algorithim designer could mathematically compare various implementations and express the level of parallelism available. Reviweing the updated version of the BSP model in Valiant's 2010 paper establishes the approach of using the model to calculate which hardware components are necessary in which quantities to avoid/address bottlenecks in the algorithm. Such a calculation/comparision is not available natively in MPI.

Conclusion

In this paper we briefly considered the bulk-synchronous parallel (BSP) model and the message passing interface (MPI), both stalwarks of the parallel computing world. While they can be compared for their similarities in thinking about parallel algorithims, the author sees them in a more complementary light. Specifically if the BSP were extended to formally support non-global communications (ad-hoc point-to-point), it would seem to be a strong predecessor for modeling an algorithm that could then be implemented using the standardized MPI libraries.