A Brief Review of "A Bridging Model for Parallel Computation"

Yi Luo

1 Summary

Based on the inspiration of the von Neumann model , the author in this paper proposes the bulk-synchronous parallel (BSP) model as an effective bridge between software and hardware particularly for parallel computation. The BSP model is promised to insulate high-level language programming and underlying hardware designing from each other. As a result, software and hardware engineers are able to just focus on their own work.
The author firstly presents the composition and attributes of the BSP model. The BSP model consists of a number of components, a router and some synchronizing facilities. It makes computation independent from communication and adopts the barrier-style synchronization. The former strategy keeps the model from concerning too much about the network topology while the latter mechanism offers easy and straightforward synchronizing. Then the author discusses methods to measure the performance of the BSP model and extends parameters of describing a software from only the problem size to several BSP model parameters such as the number of processors, the communication throughout and the cost of barrier synchronization.
To support benefits of the BSP model, the author provides two aspects of arguments. First, high-level programs can be efficiently mapped to actual machines according to the BSP model. Second, the BSP model is possible to be implemented on diverse hardware architecture. The author finally concludes that the BSP model is a promising candidate bridging model for general parallel computation. [1]

2 Merits

As a very abstract model, the BSP model have three important advantages.
Above all, it provides potential automatic memory management for parallel computation. Similar to the compiler in sequential computation, the BSP model also manages memory yet in the more complicated parallel circumstance. It adopts heuristic strategies such as hashing for a better memory allocation. It also allows concurrent memory accesses by efficiently simulating various PRAM models. The automatic memory management benefits software programmers since they no longer need to concern memory allocation affairs. However, the model gives the programmer the selection on controlling memory management for better performance as well. Without automatic compiling, the direct implementations of many specific algorithms can achieve higher efficiency.
At last, to separate communication from computation is beneficial. On the one hand, computation strategies are determined independently from the topology situation, making the model compatible to diverse network forms. On the other hand, communication contents and protocols are irrelevant to the optimization of computation so that they turn to be simpler.

3 Drawbacks

However, I believe there is a very serious deficiency of the proposed BSP model. It seems that the periodicity parameter L, which indicates the time units between two supersteps, is not easy to decide. First, we have the relation $L \geq gh$, meaning L is selected based on all possible h-relations in the network. But the communication situation that influences h is always hard to predict. Second, the computation on

different processors might be completed in various time. Then L relies a lot on the granularity of parallelism. Therefore, both computation and communication conditions impact the selection of L, to choose a proper L is more like an empirical work. I think in a specific case, people should take account of various factors such as the goal, the platform, the network etc. to determine the eventual L.

4 Personal Thoughts

Based on Section 3, it is necessary to figure out reasonable ways to decide the granularity of parallelism. If the granularity is large (then L is large), the issue of uneven completion time of different processors becomes obvious. However, if the granularity is too small, the workload of communication increases significantly. I think it would be nice if we have a discussion on this trade-off.
Another interesting point to me in this paper is the mention that "sorting is one of the basic techniques known for simulating concurrent access" [1]. I'm curious on how sorting could solve concurrent memory accesses and how it would be optimized.

References

[1] Leslie Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8):103–111, August 1990.