Automatic Conversion of Non Series-Parallel DAGs to Series-Parallel DAGs

Sajindra Jayasena Sharad Ganesh

October 9, 2003

Acknowledgements: Prof. Charles E. Leiserson, Prof. Hsu Wen Jing for suggesting the idea.

Introduction

Most of the parallel computations and applications are expressed without any restriction in the synchronization structure. The analysis of such a parallel unstructured computational model is difficult. The SP (Series-parallel) DAG model used in Cilk has significant advantages in terms of the program structure, cost analysis, estimations and scheduling for a practical parallel application. In order to facilitate the analysis and cost computations, it would be useful to be able to convert a regular DAG to a series-parallel DAG.

What we propose

We propose to develop an efficient technique for the conversion of a non-series-parallel DAG to a SP DAG with the following objectives:

- Conversion preserves the properties of the DAG like correctness , precedence relations between the graph nodes.
- Analyze and come up with an upper bound on the conversion time. We would at least like to come up with a polynomial time conversion algorithm.
- Analyze the space bounds of the transformation.
- Analysis of the increase in the critical path, due to this conversion and possibly have an upper bound on this increase, due to this transformation.
- Develop an application to back the claims made for the conversion and evaluate its performance.

Background - (Literature Review)

At this level of understanding of the problem, we have read up and have a preliminary understanding on the starting point. We plan to go about achieving what was proposed in the following way:

1. Understand and analyze the different types and structures of non-series parallel graphs.
2. Delineate the conversion of the DAG's based on their complexities.

3. Develop a conversion algorithm, which primarily aims at preserving the properties of the original DAG nodes. For example: If node A precedes node B in the original DAG, the same should hold in the converted SP- DAG.
4. The nodes in the NSP DAG can be incrementally modeled either as a spawn node or a sync node in the SP DAG. We can consider branches of the NSP DAG and recursively convert them to a SP graph structure. In this process, there might be addition of dummy nodes in order to preserve dependencies, though we have not delved deeper into the consequences of the approach yet.
5. Our basic idea is to work on a divide and conquer principle, by converting the subgraphs of the NSP graph and working up to obtain the SP graph. (However, keeping the dependencies among the nodes intact) If we are able to isolate sub-graphs of computation, by ensuring that dependency is atleast preserved transitively, then we can come up with estimates of upper bound of the increase in the critical path of the resultant graph.

Preliminary work

We are currently understanding the different graph structures and properties, in order to be able to pilot an efficient algorithm. We are actually trying to understand whether it is possible to apply the concept of Thompson's construction of creating an NFA(Non-deterministic finite automata) from a regular expression. In the graph model, the task nodes can be regarded as a state, while the dependencies can be seen as transitions. Since, a DAG with n nodes can have at most (n^2) edges. Hence, we believe it is possible to atleast have a polynomial time transformation algorithm to achieve this.