Abstract

With the advent of powerful workstations and improved network technology, using a network of workstations (NOW) as a cost-effective platform for parallel processing is gaining a great deal of momentum. In addition, tools like PVM (Parallel Virtual Machine) enable us to utilize a heterogeneous NOW effectively. In this work we look at several issues that plague the running of parallel programs on a NOW and work out strategies to tackle them and harness such a pool in a more meaningful way. The thesis presents several new strategies and results that are of use for better load balancing, scheduling and fault-tolerance of parallel programs. It also looks at the details of tools built for automatically parallelizing sequential programs for the purpose of running them on NOW. Our work can be roughly divided as follows:

Building a parallelizing compiler PACWON for automatically parallelizing sequential C-programs to parallel programs embedded with PVM calls that can run on a NOW. In this work we have used a new representation of symbolic regular section descriptors for finding the data access regions of coarse grain blocks like a nest of loops, function calls etc. We have devised simple tests that can be carried out symbolically to find whether such data descriptors overlap. We have used the Monte-Carlo method for approximating the cost of data dependence between two coarse-grain blocks. We have also built a parallelizing compiler for FORTRAN 77 that generates data-parallel code. Both these tools are complete by themselves in the sense that they generate code that can be run on the workstation farm.

In the domain of scheduling we have proposed the use of a multiobjective search strategy keeping in mind the smallness of task graphs due to exploiting parallelism at a coarse level in a NOW setup. The multiobjetive search strategy is used for producing a set of good schedules instead of a single one. Such schedules can be used for better adaptation to fluctuating machine and system requirements. As a fallout of multiobjective search strategy, we have devised a heuristic algorithm that finds the maximum processor requirement for optimally scheduling a layered task graph with cloning. This algorithm can estimate the maximum processor requirement based on just the task graph topology and without the knowledge of the edge and node weights.

We have proposed the use of alternative schedules for parallel programs to adapt to faults or varying machine requirements in a unified manner. Such a strategy can be built in a parallel program such that transparent reconfiguration takes place as and when required during the running of a program.

We have also developed a load adaptation strategy for data-parallel programs based on data replication. In such a strategy a lot of unncessary data movements can be avoided at run-time while carrying out load balancing for data-parallel programs by replicating data. We have also analyzed cases where such data replication can be useful.

In short, we have built front-ends for parallelizing sequential programs to parallel ones. In addition, we have built strategies for load balancing and fault tolerance which can be embedded in the code generated by such tools for better run-time behaviour of parallel programs. Also, one can apply our scheduling techniques for scheduling task graphs more efficiently.