Abstract. PVM uses round-robin as its default policy for process allocation to processors. The main drawbacks of this policy are the fact that PVM ignores load variations among different nodes and also the inability of PVM to distinguish between machines of different speeds. To redress this deficiency a Resource Manager (RM) is implemented which replaces round-robin with a scalable and adaptive algorithm for resource sharing [11] providing a High Performance Computing Cluster (HPCC). In this paper an implementation of a Resource Manager is proposed. The RM can be transparently plugged into PVM to offer improved performance to its users. The design of a resource manager to extend PVM is described. A prototype implementation in PVM is then measured to illustrate the utility of the approach. Finally, performance results favorably comparing the enhanced version to the original PVM are presented ...

1 Introduction

High capacity clusters of workstations are becoming an appealing vehicle for parallel computing. A parallel system may be viewed as a collection of services. There are workstations which seek services and those available which provide services. Much of the computing power is frequently idle. Resource sharing in complex network systems aims at achieving maximal system performance by utilizing the available system resources efficiently.

The PVM (Parallel Virtual Machine) system, used worldwide, enables a collection of heterogeneous computers connected by dissimilar networks to be used as a coherent and flexible concurrent computational resource. The concept of PVM pluggability is introduced in [7]. It describes a Resource Manager (RM) which can be transparently plugged into PVM offering improved process allocation. The default policy employed by PVM for resource allocation is round-robin. The main drawbacks of this policy are the fact that PVM ignores the load variations among the different nodes and also PVM is unable to distinguish between machines of different speeds. To redress this deficiency a tool that manipulates and administers system resources must replace round-robin allocation. Such a resource management system is described in this paper. It provides a replacement for the round-robin policy with a scalable and adaptive algorithm for resource sharing [11].
Section 2 of this paper describes PVM and its process control. In section 3 adaptive resource sharing algorithms are briefly described. Section 4 presents the PVM Resource Management (RM) system. In Section 5 an implementation of a RM which replaces the default round-robin assignment is described followed by prototype implementation measurements. Finally, conclusions derived from this study and directions recommended for future research are given in section 6.

2 PVM

PVM is composed of two parts - the library of PVM interface routines, called "pvmlib", and the support software system. The latter is executed on all the computers, that make up the virtual machine, called "daemon" - pvmd. These pvmds are interconnected with each other by a network. Each daemon is responsible for all the application component processes executing on its host. There is a master daemon which controls the physical configuration and acts as a name server. Otherwise, the control of the virtual machine is completely distributed. Process control is addressed in the following paragraphs.

PVM process control includes the policies and means by which PVM manages the assignment of tasks (processes in the PVM system) to processors and controls their execution. The computational resources may be accessed by tasks using the following policies: default (transparent) policy, architecture dependent policy, machine specific or a policy defined by the user to substitute the default (round-robin) PVM process control.

In the case of default/transparent, when a task initiation request pvm_spawn() is invoked, the local daemon determines a candidate pool of target nodes. The next node is then selected from this pool in a round-robin manner. The main drawbacks of such policy are the fact that PVM ignores the load variations among the different nodes and also PVM is incapable of distinguishing between machines of different speeds. Adaptive resource sharing algorithms respond to state changes. Such algorithms are briefly discussed below.

3 Adaptive Resource Sharing

The problem of resource sharing was extensively studied by DS (Distributed Systems) and DAI (Distributed Artificial Intelligence) researchers, particularly in relation to the load-sharing problem in such systems. Matching algorithms efficiently couple together nodes sharing a common interest. The performance of location policies with different complexity levels of load sharing algorithms was compared in [6]. Three location policies were studied: random policy (which is not adaptive), threshold policy, and shortest policy.

The random policy yields a dramatic performance improvement. Still, a lot of excessive overhead is required for the remote execution attempts, many of which may prove to be fruitless. Threshold policy probes up to a small number of nodes before actually sending a task for remote execution at the first one probed which is of mutual interest. In this manner, the amount of data, carried