Abstract

The typical batch queuing system schedules jobs for execution by a set of queue controls. The controls determine the queue from which jobs will be selected. Within each queue, jobs are typically selected in first-in, first-out (FIFO) order. This limits the set of scheduling policies available to a site.

The Portable Batch System (PBS) removes this limitation by providing an external scheduling module. This separate program has full knowledge of the available queued jobs, running jobs, and system resource usage. Sites are able to implement any policy expressible in one of several procedural languages. Policies may range from "best fit" to "fair share" to purely political. Scheduling decisions can be made over the full set of jobs regardless of queue or order. The scheduling policy can be changed to fit a wide variety of computing environments and scheduling goals. This is demonstrated by the use of PBS on an IBM SP-2 system at NASA Ames.

1: Introduction

Job scheduling consists of two activities. First is selection for execution of a job or jobs from the set of submitted jobs. Second is the allocation of memory and CPU resources among the set of jobs that have been selected for execution. The second activity is in the domain of the operating system kernel and will not be discussed in this paper. The first activity is typically part of a batch add-on subsystem and is the subject of this paper.

The typical batch subsystem is queue-based and schedules or selects jobs based on a set of queue controls. The nature and availability of these controls often limit the range of policies a site may implement. The scheduling policy at a given site may be a computer science related problem, such as determining the best fit of jobs in memory or aiming toward maximum CPU usage. Other sites however have policies that are driven by factors that are not related to computer science. These factors frequently include finance and even office politics. For example, priority may be given to jobs submitted by the department that owns the hardware, to a user who has "money" banked in his/her account, or to a researcher whose project is in favor with the management.

1. Computer Sciences Corporation, NASA Contract NAS 2-12961, Moffett Field, CA 94035-1000
To overcome policy limitations, the Numerical Aerodynamic Simulation (NAS) Facility at NASA Ames Research Center has developed a new batch system which provides an external job scheduler. This detachment of policy and implementation results in full freedom to express a site's job scheduling policy.

2: Queue Based Scheduling

A number of batch systems are available for open systems. The early ones only supported serial jobs and were commonly found on supercomputers. With the increasing interest of cluster computing, batch systems became more important and their capabilities have been expanded to include knowledge of parallel jobs.

It is interesting to examine several of these systems to see the common approach to scheduling and the weakness therein.

2.1: Early Batch Queuing

The first batch job queuing system which gained widespread use was the Network Queuing System (NQS) developed at NAS in the mid 1980s.[1] NQS supported multiple queues of several types. Pipe queues fed jobs into execution queues. Jobs in any given execution queue were placed into execution in a pure FIFO order. Limited controls were available to the systems operators and administrators. Queues could be turned on or off and the number of concurrent running jobs in each queue could be set. There was no provision to examine the level of system resource usage.

2.2: Current Batch Systems

Three newer batch systems are popular today. Each provides support for workstation clusters and improved scheduling controls. However, the controls still tend to be queue based.

2.2.1: Condor

Condor, a batch system designed to distribute serial jobs among workstations in a cluster, and developed at the University of Wisconsin, can be credited with starting the interest in cluster computing in 1987 and 1988.[2] [3]. Support for PVM parallel jobs has been added. While Condor does not provide queues as do most other batch systems, its concept of classes serves much the same purpose.

When a job is submitted to Condor, its requirements are sent to a central negotiator daemon. System information, including load average and current owner keyboard/mouse activity, is forward to a collector daemon. The negotiator attempts to match jobs with available hosts. This is accomplished by taking the job requirements, expressed as logical expressions, and the system information and applying a set of rules. If the rules match, the job is initiated.

As a general batch system, two limitations exist with Condor as a result of its design goal to fill empty cycles on workstations. The rule set is limited to determining if a job can run somewhere. It does not provide the "best" ordering. Also, Condor provides no usage limit enforcement.