1 of 1

Livermore Computing

Production Control System, 3.0*

Product Description

Robert R. Wood

MASTER

yz

Livermore Computing Production Control System

Copyright © 1993 by the Regents of the University of California. All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means without the prior written consent of an authorized agent of the Regents of the University of California.

This document describes a software product in progress. This software is a product of the Lawrence Livermore National Laboratory, Livermore Computing and was performed under the auspices of the United States Department of Energy contract No. W-7405-Eng-48. Information contained herein is subject to change without notice and does not represent a commitment on the part of the Regents of the University of California.

Acknowledgments

Designing and implementing the PCS product has been challenging and rewarding. Solving the problems and issues involved has been an interesting experience and the team put together to develop requirements and to implement them has worked together very well. I sincerely thank the many people who have contributed large amounts of time and energy to the development of the PCS.

Bob Cooper, manager LC Client Services, laid out customer requirements rationally and kept discussions focused on the subject at hand. Barbara Atkinson, LC project manager, tirelessly made sure that "bases were covered."

Without substantial input from the user community, the PCS could not have been written to solve real problems experienced by coordinators. Doug Dickson, Nancy Alexander, Ken Giallanza and Lynn Groves, customer coordinators, attended meetings held once per week during a period that covered two years. At these meetings, they laid out customer requirements comprehensively and graciously showed the kind of flexibility needed in trading off desires against feasibility, efficiency, etc.

Don von Buskirk, project member, brought twenty seven years operating systems programming experience to the development of the Production Workload Scheduler. Phil Eckert, has many years experience in managing resources on production machines. He was and is a "sounding board" for many of the design ideas. (He has kept a few less-than-perfect ideas from seeing the "light of day!") Phil also wrote significant portions of the RAC system and injected humor at the nicest times--when it was badly needed.

Introduction

The purpose of this document is to introduce and describe the Livermore Computing Production Control System, commonly called the PCS. The intended audiences for this document are system administrators and resource managers of computer systems who intend to use the PCS.

Resource management is very difficult in typical UNIX and related systems. The weakness involves the allocation, control and delivery of the usage of computer resources. This is a result of the origins and subsequent development of UNIX which started as a system for small platforms in a collegial or departmental atmosphere without stringent or complex budgetary requirements. Accounting also is weak, being generally limited to reporting that a user used an amount of time on a process. A process accounting record is made only after the completion of a process and then only if the process does not terminate as a result of a system "panic".

Thus, resources can only be allocated to users. No cognizance can be taken of organizational structure or projects. Allocations can not be managed at these levels. Denying service to a user who has access to a machine is crude: administrators can beg users to log off, can "nice" undesirable processes or can disable a login.

Large computing centers frequently have thousands of users working on hundreds of projects. These users and the projects are funded by several organizations with varying ability or willingness to pay for the computer services provided. With only typical UNIX tools, the appropriate delivery of resources to the correct organizations, projects, tasks and users requires continual human intervention.

UNIX and related systems have become increasingly more reliable over the past few years. One consequence of this is that resource managers and accountants have been presented with an attendant problem of accurate accounting for resources used. Processes can now run for days -- even months -- without terminating. Thus, a run-away process or a malicious or uninformed user can use an organization's budgeted resource allocation many times over before an accounting record to that effect is written. If a process's accounting record is not written because of a panic, the computer center is faced with possibly significant financial loss.

PCS Mission Statement

The PCS project was begun in 1991 to address these UNIX shortcomings. The goals of the PCS are:

THE PCS PROVIDES THE BASIC DATA REPORTING MECHANISMS REQUIRED FOR PROJECT LEVEL ACCOUNTING SYSTEMS. THIS RAW DATA IS PROVIDED IN NEAR-REAL TIME.

THE PCS PROVIDES THE MEANS FOR CUSTOMERS OF UNIX BASED PRODUCTION SYSTEMS TO BE ALLOCATED RESOURCES ACCORDING TO AN ORGANIZATIONAL BUDGET. CUSTOMERS ARE THEN ABLE TO CONTROL THEIR USERS' ACCESS TO RESOURCES AND TO CONTROL THE RATE OF DELIVERY OF RESOURCES.

THE PCS PROVIDES THE MECHANISMS FOR THE AUTOMATED DELIVERY OF RESOURCES TO ALL PRODUCTION BEING MANAGED FOR CUSTOMERS BY THE SYSTEM.

THE PCS DOES MORE THAN MERELY PREVENT THE OVERUSE OF THE MACHINE WHERE NOT AUTHORIZED. IT ALSO PROACTIVELY DELIVERS RESOURCES TO ORGANIZATIONS THAT ARE BEHIND IN CONSUMPTION OF RESOURCES TO THE EXTENT POSSIBLE THROUGH THE USE OF THE UNDERLYING BATCH SYSTEM.

Overview

The purpose of the PCS is to allocate, control and assure the proper delivery of resources on production capacity computer systems. It is also an integral part in resource usage data collection, reporting and charging mechanisms. There are two major components of the PCS. They are the Resource Allocation & Control system (RAC) and the Production Workload Scheduler (PWS).

The PCS system requires the RAC subsystem to run, but may be configured to run without the PWS. The PWS may not, however, run unless the RAC system is also running.

Resource Allocation & Control system (RAC)

Generally, the RAC system is used to create and manage recharge accounts, to create and manage allocation pools and to create and manage user allocations within the allocation pools. A recharge account should not be confused with a user "login" account. So that the term "group" not continue to be overused, the PCS has borrowed another term to mean an allocation pool or group. This term is "bank".

As resources are consumed on the machine the RAC system associates those resources with the users who are consuming them. Then the resources are associated with a bank and a recharge account. The user and the bank are debited the resources used and an accounting report is made for the purpose of charging the account. All of this is done in near-real time. If the RAC system determines that a user, recharge account or bank has consumed as much of the resources as has been permitted, the RAC system takes steps to prohibit further resource consumption by that user, recharge account or bank.

recharge accounts

A recharge account, or simply account, is essentially a credit that represents an amount of usable resources (which may be unlimited). Users and banks may be permitted to charge an account for resources used. Some users, called account coordinators, may be permitted to manage the account. That is, account coordinators may grant and deny other users access to an account. Accounts are independent from each other; that is, accounts have no sub-accounts.

The primary purposes of accounts are 1) as a mechanism by which budgetary charges may be determined for organizations whose personnel use the computer and 2) a "one stop" limit on resource accessibility to users.

domains and partitions

Eventually, a site will be able to specify several heterogeneous hosts as a single domain to be managed by the PCS. Currently, a domain definition is restricted to contain only one host. Within a domain, the nodes that make up a host may be partitioned, but only if the underlying environment, in conjunction with the PCS accounting daemon, is capable of identifying, either directly or indirectly, the partition on which resources are used. The partitioned nodes must be homogeneous.

shifts

The resources available on the machine are partitioned into time periods that exactly cover a week (168 hours) without any overlap. These time periods are called shifts. Each site

to a bank, but may not create or destroy sub-banks. The authority of coordinators extend from the highest level bank at which they are named coordinator through out that bank's sub-tree. The same is true of designees.

user allocations

Users are permitted access to a part or all of bank's resources through a user allocation. As with banks, user allocations are made on a shift by shift basis and are automatically replenished at the beginning of each shift.

Production Workload Scheduler (PWS)

The Production Workload Scheduler schedules production on the machine. Production requirements are made known to the PWS in the form of batch requests. When the PWS is installed, users do not submit their requests directly to the batch system, but rather submit them to the PWS which then submits them to the batch system.

One important function of the PWS is to keep the machine busy without permitting it to be so busy that it begins thrashing. Interactive work on the machine is not scheduled by the PWS. However, the PWS does track the resource load presented by interactive usage and adjusts the amount of production to "load level" the machine.

At any point, there is a set of production requests being managed by the PWS. This set is called the production workload. Requests in this workload are prioritized according to rules and allocations laid out by system administrators and coordinators. High priority requests are permitted to run insofar as the machine is not overloaded.

When users submit a batch request, they must specify the bank from which resources are to be drawn and the account to be charged for the request's resources.

When the PWS determines that a production request should be run, it uses a "tree walk" algorithm to decide which production request to run. That is:

- a. When the PWS determines that a new request should be scheduled, a search begins at the root bank. That is, the root bank is chosen as the "current" bank.
- b. If one or more runnable requests is drawing from a user allocation to the current bank, one of them is scheduled and the algorithm is complete.
- c. A prioritized list of child banks of the current bank is constructed. (See bank prioritization below.)
- d. For each bank in the list continue at step b.
- e. Indicate that no request can be scheduled from the sub-tree.

In other words, a recursive search is made through the tree. At each level, sibling bank's sub-trees are searched in the order of the bank's relative priority. The tree walk is exemplified in Figure 2 on page 6. If the tree is represented with highest priority sibling banks pictured from left to right, then the path taken in the search for a job is as shown in the figure. Mathematically, this is called a preorder search. The search is stopped as soon as an executable job is found.

bank prioritization

The PWS uses a mechanism called adaptive scheduling to prioritize a set of sibling banks. Simply stated, to schedule a set of sibling banks adaptively is to schedule from the bank (or

Requirements

The PCS requires an underlying NQS batch processing system. The ability to checkpoint batch jobs is highly recommended where feasible, but not required.

System functionality required of the platform vendor is as follows:

The processes of a session (in the POSIX sense of that word) must be identifiable at the session level. For instance, when a user logs in, the login process (or telnetd process) should call the POSIX routine `setsid()`. Every process spawned from any child of that login (or telnetd) process should be considered to be a member of the same session (unless one of them calls `setsid()` again in which case a new session is created for that process sub-tree). The locality of processes in a session is not material to membership in the session. This is meant to address the issue of "parallel" processes executing in several nodes of a system. Each session existing simultaneously on a platform must have a unique identifier. This identifier must be independent of the node(s) on which the processes of the session are executing.

The resources used by processes on the platform must be collected by the underlying system at the session level. The most critical pieces of information are the session owner user id and the amount of user CPU, system CPU and user induced idle CPU time consumed by the processes of the session. User induced idle CPU time results when a CPU is dedicated to a user process which then sleeps. Other data are useful as well such as characters of I/O, paging (on virtual machines), swap space consumed, etc.

The resources used by the system's idle processes must be collected as "idle time" and the resources used by other system processes that are not part of any session must be collected as "system time."

There must be a way to periodically extract the data collected by the system. The periodicity of data extraction must be configurable. The manner of extraction (reading a kernel table, report by vendor supplied software, etc.) must be efficient.

Various session management functions are required. These functions take a session id and then apply the function to every process in the session. (Usually, these functions are written to allow several classes of process identifier sets such as process, process group leader, session, etc.) These functions are:

- Send any signal to all processes in the session (including SIGKILL).

- Set the nice value or priority of all processes in the session.

- Temporarily suspend (i.e., deny any CPU time to) all processes in the session. (This function must not be revocable by the user.)

- Revoke the action of the suspend function for all processes in the session.

The final requirement is an administrative requirement. The center that intends to run PCS efficiently must put strict limits on the interactive usage of the machines being controlled. For example, an interactive per process time limit of 10 CPU minutes and a per process memory limit of ~125MBytes on a Cray YMP 8/128 has been found sufficiently strict to permit the efficient functioning of PCS on that platform. At the same time, users should be encouraged to run their production via the batch system by removing time and memory limitations through the batch system to the extent possible. For instance, on the same YMP 8/128 mentioned above, a time limit of unlimited and a standard memory limit of ~656MBytes and exceptional memory limits as high as available memory has been found to be enough of a carrot to entice users away from using the machine interactively for production purposes.

under allocation

Under allocation implies no competition for time but does imply reservation of time for the parent bank. Assume the coordinator allocated 20 minutes each to banks A, B and C. This would imply that each bank would have available to it 20 minutes each of non-contested time, but that the remaining 40 minutes of bank P's time is (nominally) unavailable to any of them. The problem of under allocation is that time allocated can not be used because it has not been allocated. The Time Sharing among Banks feature described below helps to alleviate this problem.

exact allocation

Exact allocation implies no competition for time and implies no reservation of time for the parent bank. Assume the coordinator allocated 33 minutes and 20 seconds each to banks A, B and C. This would imply that each bank would have available $33\frac{1}{3}$ minutes each of non-contested time, and that all of bank P's time is available to its sub-banks. The problem with exact allocation is that if bank A does not use its time, that time is, nevertheless, not available to banks B or C. This is alleviated by the Time Sharing among Banks feature described below.

Time Sharing among Banks

Any bank can be marked as a "time sharing" bank. The scope of time sharing for a bank is its entire sub-tree except where prevented from sharing by the exclusion feature described below. The implication of sharing time is that all time available to a bank, but not allocated to a sub-bank is delivered to the sub-bank at a very low priority.

From the above examples, assume the under allocation scheme is used and that bank A has drawn its 20 minutes of allocated time. From that time on, until the beginning of the next shift, time is delivered to sessions drawing from bank A at low priority.

Also, assume the exact allocation is used and that bank A has drawn its $33\frac{1}{3}$ minutes. From that time on, time will be delivered to bank A at reduced priority. This is, in effect, taking time away from banks B and C. In other words, time sharing permits competition (over allocation) at reduced priority.

Exclusion from Time Sharing

A bank can be marked as being excluded from time sharing. Marking any bank A as being excluded from sharing time from any of its parent banks also excludes that bank's entire sub-tree from sharing time allocated to any of bank A's parent banks.

From the above examples, assume that bank A has been excluded from sharing time, that the under allocation scheme is used and that banks A and B have drawn their allocated 20 minutes. Then, until the beginning of the next shift, no resources can be delivered to sessions drawing from bank A but sessions drawing from bank B can continue to draw resources at low priority.

assures that users can not accelerate their rate of resource consumption above a limit determined to be acceptable by the site.

short production request management

Short production introduces a strain on the ability of the PWS to deliver resources as otherwise indicated by normal prioritization means. Therefore, the site that runs PCS can limit the impact of short production by setting the maximum number that can run simultaneously, the maximum number that one user can run simultaneously and the maximum time limit that a short production request can claim after being scheduled.

Addressing the Future

Several development needs have been identified for PCS. These enhancements will make the use of PCS by administrators more flexible and will encompass a larger environment (more platforms, for instance). Users will also find the PCS to be more of an aid in the handling of their production.

Cross Host Submission and Management of Production

Users should be able to submit their production requests on one machine and have it execute on another. The results should be available at the submitting host or the executing host as requested by the user. Users should be able to establish cross-host dependencies for their production.

Enhanced Dependent Execution of Production

PCS already supports a dependency type wherein a request is prohibited from running until a designated job terminates. New dependency types should be supported. Some of these are:

Don't run a request until and unless a designated request terminates with a specified exit status. (If a designated request terminates with an exit status other than that specified in the dependency, the dependent request is removed.)

Don't run a request until a designated request has begun executing.

Begin the execution of a set of production requests simultaneously.

Don't run a request until an "event" is posted by some other request. (This would require mechanisms by which "event definitions" are declared, the occurrence of the event is posted and event definition removed when no longer needed.)

Don't run a request until specified resources (files, etc.) are staged and available.

There may be other needed synchronization tools as well.

Centralized Management of the PCS

One problem with the current PCS in an environment with more than one host is that system administrators and coordinators must work with each PCS system independently. We need to reduce the management workload by managing the PCS for all hosts from a single platform.

Distributed Management of the PCS

Coordinators and system administrators should be able to manage the entire PCS system from any host rather than being required to manage it from a single platform (centralized management).

DATE

FILMED

2 / 18 / 94

END