To appear in Proceedings of the 2nd International Workshop on Grid Computing, November 2001, Denver, Colorado
Lecture Notes in Computer Science, Springer Verlag Press

A Methodology for Account Management in Grid Computing Environments

Thomas J. Hacker[1], Brian D. Athey[2]

[1] University of Michigan, Center for Parallel Computing,
2281 Bonisteel Blvd., Ann Arbor, MI USA 48109

[2] University of Michigan, Cell & Developmental Biology,
1071 Beal Ave., Ann Arbor, MI USA 48109
{hacker,bleu}@umich.edu

Abstract. A national infrastructure of Grid computing environments will provide access for a large pool of users to a large number of distributed computing resources. Providing access for the complete pool of potential users would put an unacceptably large administrative burden on sites that participate in the Grid. Current approaches to solve this problem require an account for each user at a site, or maps all users into one account. This paper proposes an alternative approach to account allocation that provides the benefits of persistent accounts while minimizing the administrative burden on Grid resource providers. A technique for calculating the upper bound on the number of jobs and users offered to the system from the Grid that is based on historical use is presented. Finally, application of this approach to the National Institutes of Health Visible Human Project is described.

1 Introduction and Motivation

When a national scale Grid computing environment [1] becomes fully operational, potentially tens of thousands of users will be actively using resources made available to them by Grid resource providers. To access these resources, each of these users will require some form of a local account. However, requiring each participating site on the Grid to accurately maintain tens of thousands of accounts is neither desirable, nor feasible.

Many sites will likely have a finite set of users that regularly visit the site for resources. If a Grid user utilizes resources at a site infrequently, or only once, the overhead associated with creating, maintaining and deleting an account for the infrequent user would quickly overwhelm systems staff and provide a strong disincentive for resource sites to participate in the Grid.

To address this problem, a mechanism for binding Grid users to "template" accounts for a finite period of time in a local environment is proposed that provides a mechanism for local resource administrators to provide relatively instantaneous access to local resources, persistence for frequent users, and enables close tracking and accounting of Grid user resource utilization. To predict an upper bound on the

number of individual jobs and unique users that will utilize the system, a technique based on historical job logs is presented in the last section of this paper.

The temporary account binding mechanism replaces the fundamental systems paradigm of a strong binding between a real user and an account on a system with the model of a temporary binding between an account and a real user. Authentication and authorization that traditionally has been performed on the local system is replaced with distributed authentication and authorization systems, such as Kerberos, X.509[2], and Akenti[15]. To enforce user accountability, local record keeping can be used to track the actions and resource utilization of users. To preserve local administrative control over accounts, the concept of a "binding pool" is introduced, which represents an administrative domain for mapping Grid users to local accounts in a fashion similar to a NIS domain or Kerberos realm.

1.1 Intergrid Account Protocol

Account management protocols are an instance of *intergrid protocols* [17]. The establishment of a common protocol between virtual organizations to create and delete associations between Grid users, Grid computing environments (GCE), and local resource pools is essential for several reasons. First, in the absence of a common account management protocol, each GCE operated by resource providers requires individual attention to add, delete, and manage users. The amount of effort required to manage each GCE operated by a local site increases as additional GCEs are introduced into the mix of resources and services provided by a site. These efforts represent hidden costs that are rarely perceived by end-users, but are all too well known to site administrators. Thus, minimizing the number of accounts required at a local site can have a substantial financial impact. Second, if an account must be established *a priori* for each user that *might* use the site, the local resource site must maintain a large number of "unoccupied" accounts. This requirement introduces several problems. Each account requires a minimum allotment of resources, such as backed-up disk space for file storage, authentication and authorization entities (the Kerberos principals database, for example), and the purchase of additional software licenses. Additionally, a large number of dormant accounts present an additional security risk to the system. Finally, the aggregate loss of resources (i.e. disk space) due to the required minimum allocations for dormant accounts prevents the commitment of these resources to active users. The third motivation for a dynamic account binding mechanism is the growing reliance on global authentication and authorization facilities, with the corresponding decrease in the usefulness of traditional "local" authentication and authorization mechanisms. X.509 has become a global standard for authentication, and is the basis for several projects that are using X.509 for fine-grained authentication and authorization [2, 19, 21].

2 Current Situation and Related Work

The problem of maintaining accounts across a large number of hosts in a local administrative domain has been addressed over the years with a number of solutions.

Project Athena solved the problem by using Kerberos for verifying user identity, and Hesiod for maintaining password files. This central control mechanism worked well on a limited number of platforms and administrative domains. As the class of desktop hosts that required centralized authentication and authorization mechanisms increased, existing solutions were extended to cover the new systems. Kerberos authentication was added to Apple Macintosh [5] and Microsoft Windows platforms. Other authentication and authorization systems were created that addressed the problem well in one host class, but poorly in others. Examples of these include Novell and DCE.

All of these account management schemes were designed to manage a unique match of user to account (binding pool) for a well-defined administrative domain across a variety of platforms. However, when attempts are made to utilize resources across administrative domains or when the number of users exceeds 100,000, the solutions fail to scale elegantly.

Condor solves the account allocation problem by using one UID ("nobody") to execute jobs for users that do not have an account in a Condor flock [3]. The PUNCH system uses a similar scheme, where all users are represented by logical user accounts within a single physical account [4] with the ability to use a set of dynamic account bindings for system calls.

There are disadvantages to these approaches. If there are multiple jobs on the same system from different users, with all of the users assigned to one UID, it is difficult for the system to distinguish between those users since they all share the same UID (i.e. sharing scratch disk space and accountability). Moreover, with a shared UID, persistence features of the system (i.e. shared memory) are essentially unavailable to users.

The scheme described in this paper is fundamentally different than the PUNCH approach in that it creates a 1:1 account binding between a user and an account that is identical to a normal local account. There are several advantages to this scheme over the single UID approach. First, this solution is a compromise between requiring one firmly bound account for every user that may use the system versus provisioning only one UID for all users. Additionally, account templates provides a form of persistence to frequent users, but also allows new users to use the system without excessive account management overhead. This approach allows site administrators to predict and minimize the effects of Grid usage on systems they administer. Thus, the account template approach described in this paper provides a measured form of persistence, identity, and accountability.

Attempts are now being made in the Grid computing milieu to address the cross-domain authentication and authorization problems. Globus[6, 18], for example, is utilizing X.509 based authentication mechanisms to successfully deploy a computational job across a set of supercomputer systems by statically mapping an X.509 identity to a local account. The process used by Legion [20] to bind users to Legion accounts is identical to the process used in UNIX where an administrator must create an identity and password for a new user. The difference between Legion and the protocol described in this paper is that Legion requires *a priori* creation of an account for an individual user for Legion, whereas this protocol does not. Legion will thus encounter the same problems with dormant accounts that have been described.

3 Account Templates

Using account templates for user account allocation takes advantage of the potential "locality" of the Grid user's utilization pattern to support thousands of Grid users while simultaneously providing site administrators an analytical tool for provisioning their systems to limit and meter utilization of their systems by users from the Grid.

The basic idea is to create a pool of "template" accounts that have no permanent association (or "binding") with a user, and to create temporary persistent bindings between template accounts and Grid users. Each template account uses the regular account attributes that are normal for the host system, but the user information refers to a pseudo user, not a real user. For example, on a UNIX host the template account may use a pseudo anonymous user account that only permits X.509 based secure shell (ssh) logins, with proxy process creation by a root level process after authentication.

Authorization and authentication for distributed services, such as file systems, computational resources, and remote instruments is provided by distributed systems such as Akenti [15] and Keynote [19].

3.1 Binding Pools and Account Template States

To provide for integration with existing systems, there must be an administrative domain to facilitate the use of services (i.e. NFS) on systems that were not designed to operate with temporary account bindings. The set of assignments of template accounts to Grid users represents a pool of bindings (a "binding pool"). A binding pool represents a 1:1 match of a user to a template account within a finite administrative domain. To associate a user with one unique account on a host, a set of hosts should subscribe to one and only one binding pool. For example, a Linux cluster consisting of 64 hosts would subscribe to one binding pool, which may also be subscribed to by any other collection of hosts. It may be the case that user to account template assignments are identical across binding pools, but no assumptions should be made, since the binding pools are intentionally separate.

Firmly bound accounts exist in one of two states: active/available, and unavailable. A firmly bound account is active/available when the account is available for a properly authenticated and authorized user. The firmly bound account in unavailable when the account is still on the system, but has been disabled. Due to the more complex nature of temporary account bindings, additional states are necessary. Each template account in a set of template accounts will be in one of the following states and transitions described in figure 1.

4 System Issues with Temporary Binding

Most operating systems have been designed with an implicit assumption that firm binding between a user and an account is inviolate. When this assumption is invalidated, several systems issues must be addressed. These issues include

persistence, multi-role/multi-site usage, accounting, and usage metering. This section will examine these issues.

NEW	New local account that has never had any Grid user association.
ACTIVE	Local account with an active online association to a Grid user.
QUIET	Local account with no active online association to a Grid user.
UNAVAILABLE	Local account not available for any associations to a Grid user.
SCRATCH	Assignment of local account to a Grid user has been broken.
ERROR	An error has occurred in the template, and is unavailable for use.

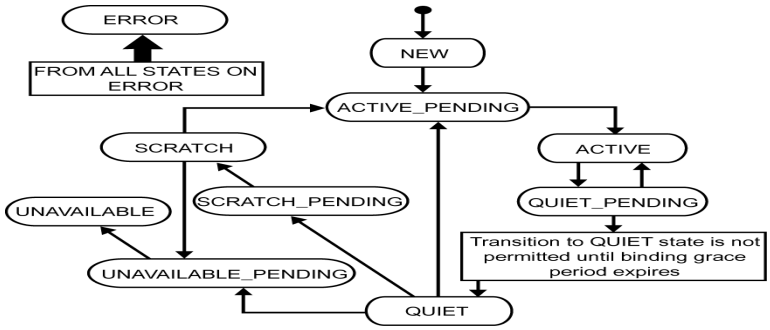

Fig. 1. Account Template States

On UNIX systems, persistent artifacts may be left upon the completion of a process. These artifacts represent persistent state that the operating system maintains on behalf of the user. Examples of this include files held in the file system, shared memory segments, semaphores and message queues. Any account template solution that creates a temporary binding between users and accounts must also have mechanisms to manage these persistent objects. To address these persistence problems, a "grace period" timer is utilized in the state transition from QUIET_PENDING and QUIET states. When a Grid process completes, the binding is not considered breakable until the grace period is expired. This gives the user some guarantee of binding persistence, and allows them to run several consecutive jobs using the bound account template without worrying about losing persistent information that is an artifact of their computation.

If a Grid user is consuming resources in a "usage role" on behalf of a sponsored project, the user's resource consumption should be charged against that project, not another project. To provide for participation in several projects on multiple sites, a mechanism to link resource usage from a "real" account to a project account is necessary. This mechanism is a Resource Consumption Account that sits between "real" user home site accounts and account templates on remote Grid sites. The only access a user can have to Grid resources is through the Resource Consumption Account. Research is currently underway [8] to define mechanisms to successfully deal with this situation.

UNIX based systems use a unique UID as the basis for authorization, and as a key for storing and retrieving resource usage information. With the introduction of high

level authorization mechanisms, new extensions to UNIX will be required to map artifacts such as X.509 certificates to the local version of authentication. Host based accounting systems will also need to be extended to store a Grid wide unique identifier, such as an X.500 distinguished name (DN) along with the usual accounting information. In practice, authentication and authorization on hosts for users with template accounts will only work if there is a production quality high-level authorization and authentication system in place at the resource site. Usage metering and accounting will still be done locally on the basis of the template account UID. Work is under way in the Grid community on these mechanisms [8, 7]

5 Determining the Peak Number of Account Templates

For practical considerations, determining an upper bound on the number of template accounts required to satisfy a stream of utilization requests for the Grid is very desirable. Maintaining an arbitrarily large pool of template accounts to satisfy 100% of offered utilization requests would compel a Grid resource provider to incur a large fixed overhead cost, since each template account requires a certain amount of static resources. To calculate a reasonable upper bound on the number of template accounts necessary to satisfy a stream of job requests, we can use the historical resource utilization of the system as a guide for prediction. Given the historical job arrival rate, job time in the system, and probability of a user arriving at the system based upon aggregate use, it is possible to determine an upper bound on the number of template accounts that will provide a predictable grade of service (G.O.S.) to the offered job stream. Consider the following: For a series of weeks from week 1 to a given week c, let N_i is the number of jobs successfully executed during $week_i$, U_i be the complete list of users (with repeats) that submitted a successful job during the week, and S_i represent the average time spent in the system for all the jobs that ran that week.

During the time period of a week, there will be a period during which the number of jobs in the system will reach a maximum. This peak usage is analogous to the "busy hour" in telephone systems in which the maximum is the daily peak reached in the number of outside telephone lines used to service outgoing telephone calls in a private telephone system [13]. Let the maximum number of jobs in the system during the peak period be represented by $J_i max$.

5.1 Characterizing the Job Stream

To model the characteristics of the offered job stream, some underlying observations of the job stream must be established. Jobs in the job stream are initiated by user actions. Paxson [10] determined that user initiated connections (such as Telnet and FTP) demonstrate exponentially distributed interarrivals and independent arrival events, and could be successfully modeled with a Poisson distribution. Since the job stream is also user initiated, it would be reasonable to hypothesize that the job stream could also be modeled with a Poisson distribution. To verify this hypothesis, the interarrival times between requests in the job stream were analyzed. The interarrival times demonstrated an exponential distribution, and the submission of jobs was

assumed to be independent, since each represents a unique job submission event by a user. Based upon these characteristics, and examination of the data, it was determined that the arrival rate of the job stream follows a Poisson distribution. The median of the arrival rate can be calculated from the historical job information, or can be estimated based upon some upper limit of execution in the system, such as a queue time limit.
For a given $week_i$, the distribution of arrival rates into the system follows a Poisson distribution with median λ_i. During the "busy period" of peak utilization of the system, the arrival rate of jobs is much higher than the median arrival rate λ_i. Accurately characterizing the arrival rate during the busy period is complex [13,14], but the arrival rate during the busy period can be approximated by taking advantage of the fact that the arrival rate follows a Poisson distribution. If we use an approximation of the median plus two standard deviations, we should be able to generate a value for the arrival rate that is larger than approximately 98% of the values in the Poisson distribution. Thus, we assume that (given that η_i is N_i normalized to units of hours):

$$\lambda_{i\max} = \lambda_i + 2s = \lambda_i + 2\sqrt{\lambda_i} \equiv \eta_i + 2\sqrt{\eta_i} \quad \textbf{(1)}$$

The time spent in the system for all jobs was analyzed and determined to be exponentially distributed (this will be useful later). The average time in spent in the system S_i is the arithmetic mean of the time spent in the system for all the jobs in $week_i$, and thus the holding time $1/\mu_i$ is equivalent to S_i.

5.2 Calculating the Peak Number of Jobs

Using Little's Law, the number of simultaneous active sessions during $week_i$ is

$$E_i = \frac{\lambda_{i\max}}{\mu_i} \quad \textbf{(2)}$$

Where E_i represents the maximum number of session active during the busy hour of the week.

The job stream offered to the system demonstrates the following characteristics: there are (potentially) an infinite number of sources; job requests arrive at random; job requests are serviced in order of arrival; refused requests are "lost"; and time in the system is exponentially distributed. Given that the system has these characteristics, the Erlang-B distribution can be used to predict the probability that an account binding request will be blocked. Moreover, for a desired blocking grade of service (GOS), the minimum number of account templates required can easily be calculated [12,13,14]. The algorithm developed by [12] can be used to calculate the number of template accounts required to satisfy a desired GOS given E_i.

The historical job logs the for the IBM SP-2 systems at the University of Michigan Center for Parallel Computing and the Advanced Computing Center for Engineering & Science (ACCES) system at the University of Texas at Austin were analyzed to measure the ability of this method to successfully predict the maximum number of account templates required to satisfy an offered job stream. The University of

Michigan logs contain 7,294 jobs over a period of 44 weeks. The University of Texas logs contain 3,160 jobs over a period of 27 weeks. Jobs that were in the system less than 10 minutes were filtered out of the job stream, to remove jobs that did not actually execute in the system. From the logs, the average arrival rate for all of the weeks was calculated using the following equations:

$$\lambda = \frac{\sum_{c} \lambda_i}{c}, \quad \lambda_{\max} = \lambda + 2\sqrt{\lambda} \tag{3}$$

Table 1 contains the results of the calculations from (3), along with the calculated average time in the system over all the weeks ($1/\mu$). Using the equation (2), E can be calculated. If we then calculate the peak number of sessions with ½% GOS using the Erlang-B Loss Formula [9, p. 273], $E_{\max}$ can finally be determined.

Table 1. Calculated Values for U-M and U-T.

Site	λ *(users/hr)*	$\lambda_{\max}$ *(users/hr)*	$1/\mu$ *(hours)*	E *(sessions)*	$E_{\max}$ *sessions*
U-M	0.6706	2.308	18.86	43.53	59
U-T	0.5245	1.973	11.588	22.86	36

To measure the ability of this method to utilize information from preceding weeks to predict the peak number of jobs and unique users for a week, the logs from both U-M and U-T were used to successively predict these values. This approach is called the *moving averages approach* in [13] and is used for forecasting demand in telecommunications circuits. After the first few weeks, the technique was fairly successful in predicting appropriate values to satisfy all of the requests for that week measured from the historical logs. Figure 2 shows that the peak number of jobs calculated compared with the actual peak number of jobs measured per week. The

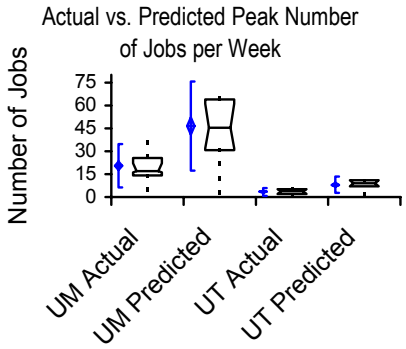

Fig. 2. Peak vs. Predicted Number of Jobs per Week

Table 2. Statistics for Figure 4

Statistic	*Median*	*IQR*	*95% CI of Median*
U-M Actual	17	11.750	16.000
U-M Predicted	45.5	33.000	32.000
U-T Actual	4	3.000	2.000
U-T Predicted	9	4.000	7.000

predicted peak value of 59 exceeds all of the measured values, and should be able to provide a ½% GOS over all weeks. The ability to calculate the predicted peak number of jobs based upon the average arrival rate and time in the system is useful for allowing system managers to provision the system for a known peak number of jobs.

5.3 Adding a Grace Period to the Binding

The results presented in the previous section is for the scenario in which jobs arrive, are temporarily bound to an account template, and the binding between the user and the template account lasts only as long as the job is in the system. In practice, however, immediately breaking the binding is undesirable for the user and for the system. If a user frequently returns to the system, the aggregate overhead of setting up and tearing down account bindings would be excessive. From the user's point of view, it would be desirable to have a period of time (a "grace period") after the completion of the jobs to be able to collect or analyze the output of the execution. If we introduce a grace period of any significant length to the system, however, some changes must be made to the predictive model to take into account the effects extending the time the account binding is in the system. If we introduce a uniform

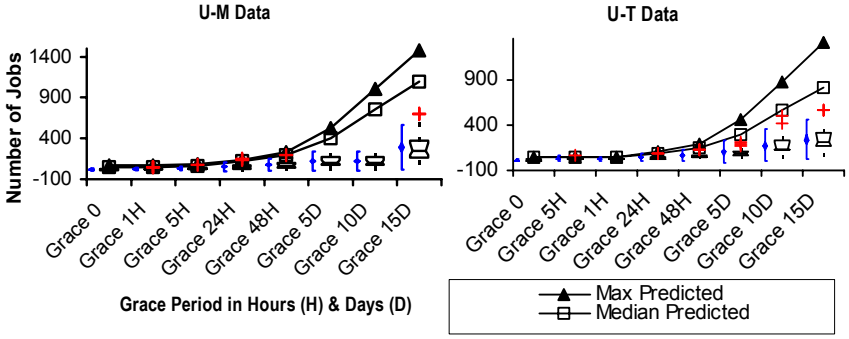

Fig. 3.1- 3.2: Predicted Peak Number of Jobs vs. Actual Peak Number of Jobs

grace period G that is selected by the systems manager, the average holding time for each job $1/\mu$ will be extended to $(1/\mu) + G$, and the corresponding weekly $1/\mu_i$ and aggregate $1/\mu$ will also be extended by G. The arrival rate of the offered jobs to the system λ_i would remain unchanged, and the peak predicted number of jobs would also be scaled appropriately for G. To verify this result, the University of Michigan and University of Texas logs were analyzed to determine the peak number of jobs in the system over all the weeks in the logs for a given grace period G. Figures 3.1 and 3.2 shows the predicted vs. actual peak number of jobs for a set of grace periods from 0 to 15 days for University of Michigan and University of Texas. This analysis demonstrates that the model remains valid when an additional grace period is introduced.

5.4 Predicting the Number of Unique Users in the Job Stream

At the busy period during the week, if the actual peak number of jobs is *M*, there are *M* jobs owned by *X* unique individual users in the system, assuming that the jobs during the busy period start at time t_0 and completes on or after time t_1. For the worst case, one could assume that each job is assigned to a unique individual user, and thus would require *M* template accounts. In practice, however, the number of jobs attributable to a user is the product of the probability of a job originating from that user in the job stream and *M*. This is due to the theorem that a Poisson process can be partitioned into a set of independent Poisson processes [9, p. 74]. To apply this theorem, we must determine the probability of the user being the originator of a job in the job stream from the historical logs.

Let p_i = Probability ($user_i$) = (Number of jobs for $user_i$ / Total Number of Jobs). Thus,

$$\lambda_{avg} = p_1\lambda_{avg} + p_2\lambda_{avg} + \cdots + p_k\lambda_{avg} \quad \textbf{(4)}$$

$$\lambda_{\max} = p_1\lambda_{\max} + p_2\lambda_{\max} + \cdots + p_k\lambda_{\max} \quad \textbf{(5)}$$

where k is the number of users in the historical logs. Thus,

$$E_{\max} = \frac{1}{\mu}\left[p_1\lambda_{\max} + p_2\lambda_{\max} + \cdots p_k\lambda_{\max}\right] \quad \textbf{(6)}$$

$$E_{u1} = \frac{(p_1\lambda_{\max})}{\mu}, E_{u2} = \frac{(p_2\lambda_{\max})}{\mu}, \cdots, E_{uk} = \frac{(p_k\lambda_{\max})}{\mu} \quad \textbf{(7)}$$

The predicted number of unique users at the busy period will be the number of terms in (7) that have a values greater than 1.

$$\|U\| \text{ where } U = \{E_{ui} : E_{ui} > 1\} \quad \textbf{(8)}$$

We can then add the grace period G described in the previous section to (7):

$$E_{\max} = \left(\frac{1}{\mu} + G\right)\left[p_1\lambda_{\max} + p_2\lambda_{\max} + \cdots + p_k\lambda_{\max}\right] \quad \textbf{(9)}$$

The period of time window used in the historical job log to calculate the probability for each user that is used to calculate E_{max} must be at least 2 ($1/\mu + G$) to capture all possible jobs in the window. For the most accurate results p_i should be based upon an analysis of the job log as far back into the past as is practical. Figures 4.1 and 4.2 shows the predicted peak number of individual users per week vs. the actual peak number of individual users for U-M and U-T. It can be demonstrated that as the grace period G approaches infinity, the maximum number of users calculated from the probability vector approaches the number of individual users that utilized the system. This corresponds to the firm binding case, where over a very long period of time, the number of users that use the system matches the number of users contained within the

password file on the system. With the addition of the probability of individual users using the system, system managers can then confidently provision the system for a known peak number of users based upon the historical use information of the system.

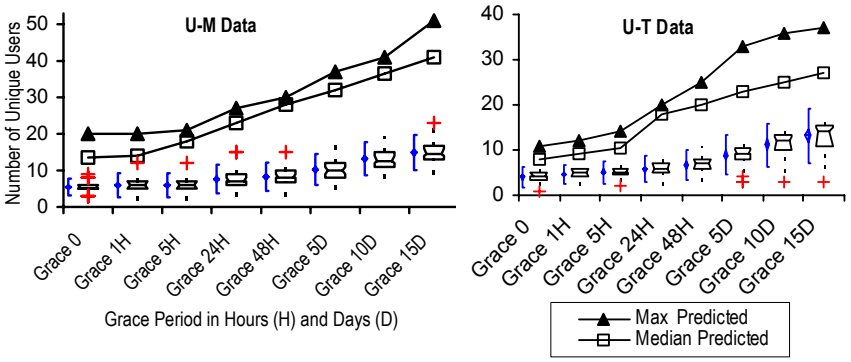

Fig. 4.1-4.2 Predicted Peak Number of Users vs. Actual Peak Number of Users.

5.5 Application: Determine Grace Period and Number of Template Accounts

Now that we can accurately predict an upper bound on the number of template accounts and individual users a Grid resource site would service given the historical logs, GOS, and grace period, we now want to be able to easily calculate the number of template accounts a site would need to support a GOS and grace period selected by the site administrator. The process to do this is as follows: first, calculate average holding time $1/\mu$ by calculating the arithmetic mean of the time in hours in the system for all jobs over a certain filter threshold (to filter out unsuccessful jobs); second, calculate arrival rate λ by dividing the total number of filtered jobs in the log by the number of hours the log covers; third, calculate the user probability vector **p** by taking the inverse of the number of times a job comes from a particular user (for example, if user X has 10 jobs, Prob(X) = 1/10). Now, calculate λ_{max} and E using equations (2) and (3).

If you have a package to calculate the Erlang Loss Formula, such as Qsim for Excel [11], or a Java applet [12], you can calculate E_{max}, and the peak number of users by the number of elements of the vector $\mathbf{u} = \mathbf{p}E_{max}$ that are greater than one. This will give you the peak number of users that will utilize the system with a known GOS.

6 Conclusion: Application and Future Work

In this paper, an alternative to existing account allocation techniques was presented that addressed the problems of identity, persistence and accountability present in these systems. The Visible Human Project (VHP) at the University of Michigan sponsored by the National Institutes of Health [16] (NO1-LM-0-3511) plans to serve visible human

datasets to learning systems at medical training facilities. The allocation approach described in this paper is critical for supporting thousands of VHP users. For future work, investigation of the effects on this model of self-similar arrival processes described by Paxson [10] from automated processes should be done. Another area for future work is a thorough investigation of the effects of temporary account bindings on Operating Systems design, and investigation of what changes would be required on a UNIX system to support temporary account bindings. Another area of work is the prediction of peak job and users for jointly scheduled production systems, such as the Berkeley Millennium system, Globus, and the U-M/U-T NPACI Grid.

References

1. I. Foster, C. Kesselman (editors), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kauffman Publishers, 1999.
2. R. Housley, W. Ford, W. Polk, D. RFC 2459 “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, January 1999.
3. D. H. J Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne, "A Worldwide Flock of Condors: Load Sharing among Workstation Clusters", Journal on Future Generations of Computer Systems Volume 12, 1996.
4. N. Kapadia and J. Fortes, “PUNCH: An Architecture for Web-Enabled Wide-Area Network-Computing”, Cluster Computing: The Journal of Networks, Software Tools and Applications, September 1999.
5. B. Doster, J. Rees, “Third-Party “Authentication for the Institutional File System”, University of Michigan CITI Technical Report 92-1.
6. I. Foster, C. Kesselman, C, “Globus: A Metacomputing Infrastructure Toolkit”, International Journal of Supercomputing Applications, 11(2): 115-128, 1997.
7. T. Hacker, W. Thigpen, “Distributed Accounting on the Grid”, Grid Forum Working Draft.
8. R. Buyya, D. Abramson, J. Giddy, “An Economy Grid Architecture for Service-Oriented Grid Computing”, 10th IEEE International Heterogeneous Computing Workshop,April 2001.
9. R. Wolff, Stochastic Modeling and the Theory of Queues. Prentice Hall, 1989.
10. V. Paxson, S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, pp.226-244, June 1995.
11. 3 Point Technologies, Inc. Qsim Modeling Functions for Excel.
12. S. Qiao, L. Qiao, “A Robust and Efficient Algorithm for Evaluating Erlang B Formula”, TR CAS98-03, Department of Computing and Software, McMaster University, 1998.
13. J. Green, The Irwin Handbook of Telecommunications Management. Irwin Professional Publishing, 1996.
14. Intel Support Document #8150:A Traffic Engineering Model for LAN Video Conferencing.
15. M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, Essiari, “A. Certificate-based Access Control for Widely Distributed Resource”,. Proceedings of the Eighth Usenix Security Symposium, Aug. `99.
16. M. J. Ackerman, “The Visible Human Project,” J. Biocomm., vol. 18, p 14, 1991.
17. I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations.” (to be published in Intl. J. Supercomputer Applications, 2001).
18. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. “A National-Scale Authentication Infrastructure.” IEEE Computer, 33(12):60-66, 2000.
19. M. Blaze, J. Feigenbaum, A. Keromytis. “KeyNote Trust-Management System”, RFC 2704
20. Legion 1.7 System Administration Manual. The Legion Group, Department of Computer Science, University of Virginia. pp 32-33.
21. W. Doster, M. Watts, D. Hyde. “The KX.509 Protocol”, CITI Technical Report 012, February 2001.