System Resource Accounting in a Heterogeneous Distributed Computing Environment

B.K.Rana
SERC,IISc
Bangalore,India

M.Jacob
CSA,IISc
Bangalore,India

N.Balakrishnan
SERC,IISc
Bangalore,India

Abstract

The accounting tool, SAT, is designed to perform system resource accounting in a large, heterogeneous, distributed computing environment. This paper describes the design,implementation and operation of SAT. The tool is currently operational at Supercomputer Education And Research Centre (SERC) at Indian Institute Of Sceince. SERC is a large academic computing centre catering to the computing needs of the students and faculty , as well as to those of several paying users. SAT is equipped with both command line and graphical user interfaces. The software is developed using C-shell, Tcl, Expectk (Expect with TK), demonstrating the usability of these development languages. It is currently being used in an environment of a few hundred computers comprising SGI Iris, Indy, Indigo and Onyx machines, SUN Sparc20s, IBM RS6000s, a DEC VAX 8810, etc. Besides, the work brings out certain vagaries of connect time accounting in UNIX systems and the quality results of the software development process followed.

Introduction

Most operating systems provide accounting software for standalone machines with provisions to account CPU time, connect time and disk blocks used by a user. The systems are typically configured to generate a month end fiscal report of users and their system usages. These fiscal reports are of different formats on different machines. In a large heterogeneous computing environment, the collection of these fiscal reports from all machines and their combination into a single report for the chargeable users in the system is both time consuming and tedious. We have developed the SERC Accounting Tool or SAT to automate this process of heterogeneous system resource accounting. The tool employs the client-server computing paradigm. Specifically, it collects monthly fiscal files from different machines, prepares monthly bills for chargeable users and maintains information about chargeable users.

In a related work done at the University of Rocester [2], the emphasis was on writing scripts to generate monthly fiscal files for UNIX machines whose accounting subsystems are incomplete. This is not a problem with UNIX variants today. SAT is designed to operate with any operating system capable of performing system resource accounting. Certain design aspects of *[2]* are considered and the requirements are specified in the following section.

Software Requirements

The software is designed to meet the following requirements.

1. The package is intended to generate monthly bills for chargeable users.
2. The software should maintain the chargeable users' database, providing the system administrator with an interface to create, delete and modify information records about chargeable users.
3. The software should work with any number of computers running any operating systems - UNIX variants, VMS or any other operating system capable of accounting CPU time, disk usage, connect time and of preparing monthly fiscal files.
4. The software should be able to operate in a large environment in the presence of temporary and short time computer failures.
5. The entire process of accounting should be automated, not requiring direct intervention of a system administrator.
6. Easy operator interface, installation and configuration procedure.

7. Easy upgradability and configuration of tariff information for different machines' computing resources.

8. From the performance standpoint, the software should not generate heavy network traffic, use minimal disk space and I/O should be minimum.

9. Since operation is distributed across a large environment, it should be strictly non-instrusive and non-destructive.

10. On-line user help should be provided.

Software Architecture

With these requirements in mind, the software has been designed as shown in Figure 1. Figure 2 illustrates its hierachical design. Note that the package is structured as a client-server application. Each machine doing stand-alone accounting is a client. The server runs the centralized portion of the software, responsible for the merging of fiscal reports, bill preparation, user information maintainance and configuration. The communication between accounting clients and server could be provided through ftp, sockets, email, or other mechanisms. Instead, we employ a common file system, accessible throughout the environment in all machines, already in use for other system administration activities. Another major issue in the design is the database, since there is significant information management involved. Instead of using facilities like dbm, gdbm in UNIX we manage with simple text processing utilities like grep,awk,sed and hypertext file. Currently the later is used. Desiging in a database paradigm is an upgradability of the package. Implementaion issues are discussed later. Figure -1 depicts five accounting program modules, i.e. AP1-AP5, and eight files related to the software, i.e FISCF, SFISCF, hosts.config, UIF, CUF, CLOGF, CF and BF. These are next described briefly.

Modules of'software

1. AP1 This is the accounting subsystem of the computer's operating system. Today, most UNIX systems provide programs to generate daily and monthly system resource accounting, which normally run as cron jobs. Detailed information is available in the machine specific manuals. As far as SAT is concerned, the monthly fiscal file of the machine, FISCF, is of interest.

2. AP2 This is the client module of SAT. In the first step it checks whether fiscal report exists; if not it invokes AP1 to generate fiscal report. In the next step, it prepares the short fiscal file reading common user file CUF and FISCF. Then SFISCF is copied to the directory based on the machine category(DPMC in Figure-1). Thus, the actual disk space required by the software is minimized. This module runs in machine's cron job from the second day to fifth day of each month during off hours. If a machine is down for a short period (less than these four days), it does the job because of it's repeated run. Administrator may execute this using rsh in UNIX machines.

3. AP3 Generates common username file, CUF, out of the user information file, UIF. This should be run after a new chargeable user is created, an user is given a new user name or a chargeable user is deleted.

4. AP4 Combines all short fiscal files per machine category. If the machine category is not UNIX it converts the short fiscal file to unix format and then does the combination.

5. AP5 This is the billing module of the software. It reads all combined fiscal files and user information file and charge file, generates the bill for the month. After bill generation it updates the user information file. This module also contains a program to generate the summary of the bill and accounting status for the month (like how many machines have reported, machines suffering from malfunction and errors etc.).

Files of Software

Files and the formats of the files used are discussed here. More detailed informationuser may ftp the package.

User Information File or UIF

This file contains all information for chargeable users in hypertext format. Each user information record is set of simple text lines containing start of the record, office reference , year of creation and tariff type , name of the user, address , copy to , disk quota, list of user names in the lab , advance paid , cummulative charge , mode of payment.

Common User File or CUF

The file contains list of user names. The names are extracted from UIF. A sample file is given below.

FIGURE - 1 ANY MACHINE

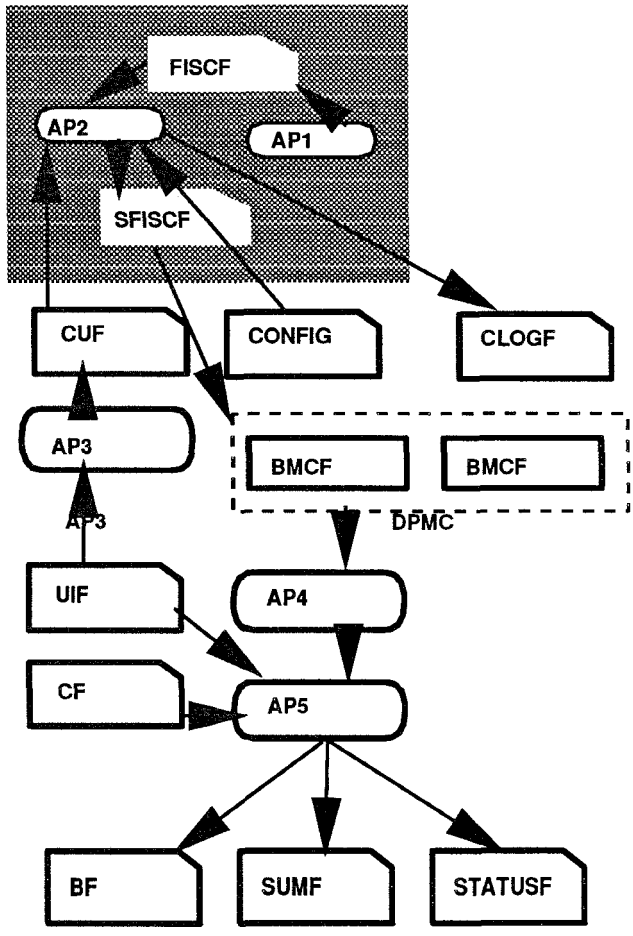
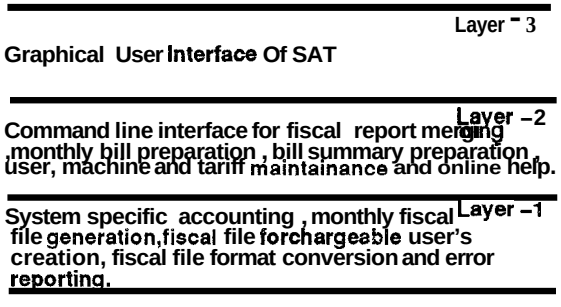


FIGURE –2



Common log file or CLOGF

Machine specific accounting events (report prepared successfully, error occured etc.) are logged into this file. These files are read, while preparing accounting status, as described in AP5. Administrator may read this file to know the status. One such file is presented below. The first line shows an error. Following three types errors are recorded.

1. Fiscal report not found.
2. Wrong configuration of software.
3. Installtion not OK.

Bill file or BFIEE

This is the final output of the software, containing bills for users. We are not strict about the format of the this file. Charges may be formatted anyway with extra office related informations to produce this. Code provides modifiability at this point.

Status file or STATUSF

This file contains a list of the machines that submitted SFISCF correctly and a list of the machines that were not able to submit SFISCF. During the first week of a month the administrator may generate and look into the file and if all machines submit SFISCF, go ahead to merge SFISCFs and generate bills.

Bill summary file or SUMF

This file contains the summary of bills generated for a month.

Hosts configuration file or hosts.config

This file contains the machine specific configuration details. It has got two sections. First section contains the information for each machine category as given below.

1. Location of the monthly fiscal file.
2. Name (with path) of the accounting script provided by operating system to generate the monthly fiscal report.
3. Name of the tariff category (see charge configuration file) for the machine.

Second section contains machine name , alias ,ip address and the tariff category for all machines in the environment in a tabular manner.

Fiscal file or FISCF

This is the monthly accounting report prepared by a machine's accounting subsystem. In an UNIX machine, on the first day of every month "monacct" program is run in cron job to create this file for the previous month. One can see the file format in any unix machine.

Short fiscal file or SFISCF

Short fiscal file contains accounting information for a machine in the same format as fiscal report, but it contains only for the chargeable user names listed in common username file or CUF. This is prepared by the accounting program module AP2, as discussed previously.

Charge file or CF

This file contains the charging information. It contains a table in hypertext format having cpu time , connect time and disk charges for machines.

User interface

As shown in Figure-2, the software is designed in three layers. It provides both command line and graphical user interfaces. A synopsis of command line interface calls are presented below. Figure-3 shows the main window view of the graphical user interface. The package has a graphical user interface based on tk4.0.

Implementation

Based on the requirements listed earlier, we found the following issues for implementation.

1. What language or script to use for implementing individual modules?
2. In interfacing an interactive command line interface with a graphical user interface, does any particular language seems more beneficial than others help?
3. Fast and modifiable programming.
4. How to interface non UNIX machines?
5. What are the environmental constraints for the software?

We found that tcl, tk, expectk are most suitable considering the points listed above. Expectk eliminates the problem of interfacing command line code and graphical user interface. Layer-2 of the software is written in tcl and Layer3 in expectk. Since the only non-UNIX machine is-a VAX 8810, we decided to copy the fiscal file manually and generate the short fiscal file using a tcl script. For any othe non UNIX machine this part of the software should be written. Since the client part of the software (AP2) runs as a cron job in the UNIX machines, C Shell is used here. The Layer -1, as shown in Figure -2, is written in C Shell. While writting AP2 in C-shell for varieties of UNIX machine the problem of non standard interface to UNIX utilities is faced. For example, for some machines, grep provides "-w" option to grep word, some machines restrict the size of hostname to eight, some machines provide "hostname" call to get hostname, whereas in others, one has to get it by "uname" call. It is found that since client side execution is done **as** a cron Job and it is time based e.g monthly accounting program should run on the first day, 5am in the morning, fiscf2sfiscf (AP2) should run on the second day through the fifth day of the month, *6* am in the morning and also the daily accounting program should run early in the morning. This requires synchronization of time in the environment.

Loophole in UNIX Accounting Design

The existing accounting concept in UNIX operating system is process based. As discussed [5], the accounting record for the process is created when the terminal for the process is opened. This poses a problem in charging users for connection time in present days machine having windowing systems. Accounting based on connection time is useful for graphics machines. Presently, connect time for a user is the sum of connect time for all windows opened in the terminal. The measurement of connect time can be made accurate by accounting the maximum of all connect times, in stead of summing up connect time for processes of an user.

Monthly billing procedure

1. Monthly accounting €or each machine is done by a system cron job on the first day of every month. SFISCFs are created by the fifth day of every month. After the fifth day, the administrator should prepare the status file. If all machines submit SFISCFs successfully then SFISCFs are combined and bill prepared. Otherwise SFISCFs for the failed machines are obtained using GetSfiscf. For non UNIX machines this should be done manually.
2. SFISCFs are combined and bill prepared. Afterwords, if bill file is edited manually, UIF should be updated accordingly.

Table - 1 : Command line interfaces

Interface Call	Description
MakeBill	Generates the bill for the previous month if combined fiscal files for the machines exist.
MergeRpt	Combines the short fiscal file per machine category.
GetSfiscf	Prepares the short fiscal file for a remote machine. This is used if the machine failed to generate short fiscal file bv cron iob.
PrepareStatus	Prepares status of accounting.
UIF2CUF	Generates common user file out of user information files.
PrepareSum	Prepares bill summary.
CleanOldFiles [-all] [-month]	Cleans accounting related files for the specified month and files older than this month. If -all specified it cleans bill files.
Help [-topic]	Provides accounting related help.

Table - 2 : Quality Matrix

No	Quality Element	Use Of tcl/tk	Design	UNIX env.	Remark
1	Maintainability	2	1	0	Apart from coding flexibilty because of use of tcl and tk.
2	Flexibility	2	1	0	partitioning GUI and CLI interfaces contributed significantly.
3	Testability	0	2	0	No comment
4	Correctness	1	2	0	Use of TCL/TK has not imposed any constraints on correctness.
5	Reusability	0	0	1	Few GUI modules are reusable and design concept is reusable ,particularly for small database design.
6	Integrity	2	0	0	No comment
7	Efficiency	0	2	1	No comment
8	Usability	0	2	1	No comment
9	Portability	2	1	0	Guided by TCL/TK portability.
10	Cycle time reduction	2	0	0	Contributed from process of development and software used.

Degree Of contributions from different sources
2 : Highest, 1: Average, 0 : No effect, -1 : Average reverse effect, -2: Extreme reverse effect.

3. Creation of a chargeable account in SAT (updating UIF and generating CUF) should be done after the bill is prepared. For example, if a new account is created on October 2nd, BF is created on October 10th, after 10th October the new account information may be added in UIF and CUF generated.

4. Similarly, deletion of a chargeable account should be done after bill is generated. For example, suppose that a user requests to delete his account on October 25th, the bill for October is prepared on November 10th. The account information is UIF may be deleted after November 10th.

5. Old accounting related files should be deleted. Bill file may be kept until accepted by customer.

Software quality

Development of the package using tcl, tk, expectk urned out to be an excellent exercise in quality software development process. **A** breif quality matrix is presented in Table -2. We observe that

1. The development process confirms to the class of Rapid Application Development.

2. The code is extremely maintainable.

3. The code is not large.

4. Portability has been achieved.

Upgradation

As discussed in the architecture and implementation section, the software is upgradable in following respects.

1 New machine category can be added by configuring hosts.config, charge file (CF) and creating a machine category directory under DPMC. AP2 module needs to be coded. For example, to include HP 7xx machine, one has to create HP700 directory under DPMC, hosts.config and charge file need to be configured as per given instructions in the file. If the provided fiscf2sfiscf (AP2) is not suitable for the machine, it must to be written for this category.

2. Efficient database management can be done using some standard database package.

Conclusion

Currently, the first error free version of the software is operational in Supercomputer Education and Research Center, Indian Institute of Science. The software conforms to all desired requirements. Due to the use of tcl, tk and expectk, the developement cycle is drastically reduced (hardly six staff months). The package demonstrates the usability of tcl, tk and expectk in many directions. The package is available for download at http://www.serc.iisc.ernet.in.

References

[1] Brent Welch, Practical Programming in TCL and TK, *Draft, January 13 , 1995.*

[2] John Simonson, System Resource Accounting on UNIX Systems, University of Rochester Computing Center,*LISA V - Sep. 30-Oct. 3, 1991 - San Diego, CA.*

[3] Don Libes, X Wrappers for Non-Graphic Interactive Programs, *Proceedings of Xhibition 94, San Jose, June 20-24, 1994.*

[4]Don Libes, Exploring Expect,*O' Reilly & Associates, Inc.*

[5] Samuel J. Letter, Marshall Kirk McKusick, Michael J. Karels, John S. Quarterman, The Design and Implementation of the **4.3** BSD UNIX operating system,***1986.***