What Every Administrator Should Know Before Trying To Set Up a Share Hierarchy in the UDB

Kathlean C. Zinnel, Cray Research, Inc., 655-F Lone Oak Drive, Eagan, Minnesota 55121

ABSTRACT: *Ninety percent of reported Fair Share Scheduler problems are actually caused by unclear Share hierarchy definitions in the User Data Base. Fair Share Scheduler behavior is determined by information contained in the "lnode" table. Information passed to the kernel when creating lnodes comes from the User Data Base. The "shrtree" command was created to "visualize" what lnode chains will look like after instantiation so that administrators can verify if political goals are clearly defined. Several planning stages facilitate Share hierarchy implementation. Guidelines are presented for deciding if share-by-user-id or share-by-account-id is the correct choice for your site.*

INTRODUCTION

The UNICOS Fair Share Scheduler (FSS) prevents users, or groups of users, from stealing more CPU cycles than they are entitled to consume when there is no idle time. Turning on FSS for the first time is not as simple as setting a flag. Political goals must be expressed through User Data Base (UDB) information. Inappropriate UDB structure could result in lnode chains that cause pathological FSS behavior. Having worked with a number of sites recently to get them up and running with FSS in share-by-account-id mode, this papers offers suggestions for making the transition smooth and painless; both for system administrators and their user base.

UNICOS code for Fair Share exists in many different places. Run-time data needed for CPU allocation decisions reside in kernel structures called the lnode table (lnode signifies limit node) and the share constants (sh_consts) structure. Each process is attached to a particular lnode chain which determines the CPU allocation policy applied to the process. CPU ticks are the currency used to determine which lnodes are deserving, or not, of having the CPU resource allocated to their attached processes in the near future. Lnode configuration is determined from UDB information. With a share-by-*uid* mode of operation, a minimum lnode chain consists of the Root lnode pointing to a resource group lnode which points to a user-id lnode. Under share-by-account-id mode of operation, a minimum lnode chain consists of the Root lnode pointing to a resource group lnode which points to a shareholder lnode. In both modes, processes are attached to the terminal (user-id or shareholder) lnodes only. Under share-by-account-id mode of operation, processes from different user-ids can be attached to the same shareholder lnode.

Of the approximately 100 fields in each UDB record, only 8 are concerned with FSS hierarchy definition: *uid*, *shflags*, *resgrp*, *shares*, *acids[]*, *shusage*, *shcharge*, and *shextime*. The user-id (*uid*) field is the basic identifier of each UDB record and must be unique. Administrators define entitlement through resource group (*resgrp*), account-id (*acids[]*), and share (*shares*) fields. System administrators should be aware that user-ids, resource groups, and account-ids all share the same namespace. At each level, the *shares* value determines how to allocate parent resource group entitlement among sub-groups. Long-term usage history for individual users, and groups of users, is also kept in the *shusage*, *shcharge*, and *shextime* UDB fields and preserved across system startup. These fields are used to initialize the l_usage and l_charge fields when an lnode is created. L_charge is the long term accumulated costs and always increases unless the *shcharge* field in the UDB is cleared by an administrator. L_usage is the decaying accumulated costs and can increase or decrease based on the usage decay rate set with shradmin. When an lnode is created, l_usage is created from *shusage* after first decaying this value according to the last logout time stamp in *shextime*. A UDB permbits flag (PERMBITS_ACCTID) allows the holder to newacct to any account-id which is useful when testing share-by-account-id.

Translating Political Goals Into UDB Entries-Stage I

- Write down political goals first
- Circulate among user base to set expectations
- Match resource group names to political unit
- Each resource group subdivides next level
- Categorize every UDB entry as to affiliation

Upon starting the process of configuring the UDB for FSS, it is very important to write down political goals. Figure A depicts a Share hierarchy on a machine that is primarily used by developers. To ensure that developers work takes precedence over other work, the top level "Users" resource group has been

subdivided into "SoftDev", "CCN", and "Mktg"resource groups with 60%, 20%, and 20% of the machine, respectively. Once machine entitlement goals have been expressed in words, time must be taken to circulate the hierarchy design for review by the various political units. A participatory review process helps to set correct user expectations for what system behavior will be after FSS is turned on.

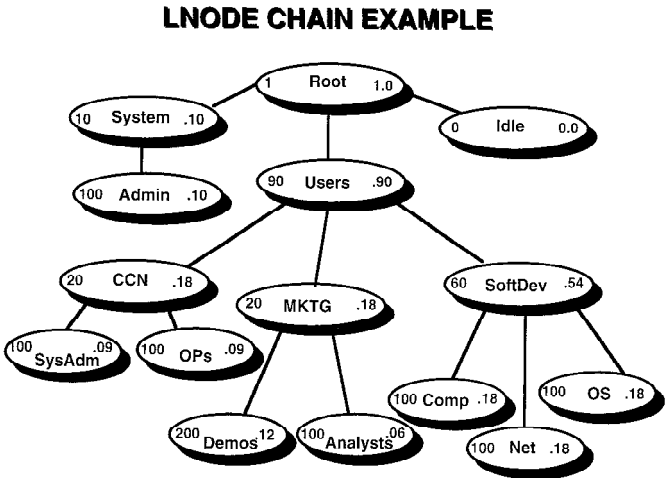

Figure A

When selecting names for the resource group and shareholder (another way of referring to account-id) UDB entries, think about how they will look in the shrmon or shrview display. Groups of users are represented by resource group and acid UDB entries. Names which convey political affiliation are useful because everyone will be looking at these displays initially to verify what is happening. Categorize every UDB entry as to political affiliation on paper first; this eases the process of translating these relations into UDB values for resource group and acid list.

The relative machine share, i.e. entitlement, given to a particular lnode in a chain is a function of the number of shares (UDB field *shares*) given to that particular lnode divided by the sum of all the shares of all the lnodes who have the same parent lnode, i.e. resource group. This group share ratio is then multiplied by the share ratio of the parent lnode to calculate entitlement. Common practice for assigning *shares* seems to be permitting the political unit represented by a particular resource group to choose the numbers used for *shares* for the next lower level of resource groups or acids. Another hint borne of experience is to design a balanced Share hierarchy such that the depth of all possible lnode chains for users are within one level of each other. This keeps the variation among the relative machine share (entitlement) for each terminal lnode within the same order of magnitude.

Translating Political Goals Into UDB Entries - Stage II

- Decide whether to base lnode chain by uid or acid
- Select uid range for resource groups and shareholders
- Make copy of UDB files in local directory
- Use -p option of udbgen & udbsee to prepare directives

Structure of the UDB for FSS really depends upon which mode of operation, share-by-uid or share-by-account-id, is selected. Under share-by-account-id, a single user may have processes running under different lnodes. Again using Figure A, assume user xyz does system admin work for CCN occasionally, runs demos sometimes for Marketing, but mostly does OS development. When user xyz logs in, his/her default acid (*acids[0]*) will be used to create shareholder lnode OS which is under resource group SoftDev. Any work submitted will run under this lnode chain, unless user xyz newacct's to SysAdm or Demos which are also in user xyz's acid list in the UDB. Work submitted after a newacct to SysAdm runs under the CCN lnode chain and has a lower machine share entitlement than OS. Work submitted after a newacct to Demos runs under the Mktg lnode chain and has a higher machine share entitlement than SysAdm. The qsub "-A" option can also be used to specify under which shareholder (account-id) lnode a batch job should run.

With share-by-uid, only one user runs under the terminal lnode in a chain and the newacct command does not change lnodes. In this mode it is the *resgrp* field in the user UDB entry that is used when a user logs in, rather than the default acid list entry (*acids[0]*), for creating an lnode chain. Under share-by-uid operation, a misbehaving user can be controlled by setting the users *shares* UDB field to zero with shrdist without impacting other users. Share-by-uid mode is appropriate when the desired level of control is an individual user. Share-by-account-id mode is best suited when control by project is preferred.

UDB entries for resource groups and shareholders (acids) share the same *uid* namespace as the entries for users who actually log in. There must be a unique *uid* number allocated for each resource group and shareholder as well as for each user. When setting up a share-by-account-id hierarchy, there are often acids used by accounting that are the same number as the *uid* of a user login entry. The solution to this problem as shown by Example 1, is to start the resource group and shareholder uid numbering sequence at a high value, in this case 8000, but not so high as to exceed sysconf (_SC_UID_MAX). Mapping all existing acid and resource group numbers to the new numbering sequence as 8000 + old-number solves the *uid* namespace problem but often requires some changes to accounting reports.

All necessary input to udbgen for adding Share hierarchy definitions to the UDB can be prepared without modifying the live system. This must be done as root and with sysadm privilege if running security. First copy /etc/udb, /etc/udb.public,

Lv	Name	ID	System Share	Group Share	System Usage	Status	Flags
0	_ROOT_	0	100.0%	100.0%	100.0%	G	40000
1	Bkgrnd	8311	5.0%	5.0%	0.0%	G	40000
2	Batts	5098	5.0%	100.0%	0.0%	A	1000000
1	CCN	8354	20.0%	20.0%	6.3%	G	40000
2	Serv	8001	5.0%	25.0%	2.6%	A	1000000
2	SysAdm	8373	10.0%	50.0%	0.3%	A	1000000
2	Syssup	8388	5.0%	25.0%	3.4%	A	1000000
1	Mktg	8381	10.0%	10.0%	59.8%	G	40000
2	Country	8359	0.2%	2.3%	0.2%	A	1000000
2	Cust_a	8361	9.0%	90.1%	0.0%	A	1000000
2	Cust_b	8362	0.1%	0.9%	0.0%	A	1000000
2	Demos	8367	0.2%	2.3%	0.1%	A	1000000
2	Intl	8374	0.2%	2.3%	0.0%	A	1000000
2	TechOps	8390	0.2%	2.3%	0.0%	A	1000000
1	SoftDev	8386	45.0%	45.0%	23.7%	G	40000
2	Cust_c	8363	3.5%	7.8%	0.0%	A	1000000
2	Cust_d	8364	35.3%	78.4%	0.0%	A	1000000
2	Netdev	8383	0.9%	2.0%	0.1%	A	1000000
2	Userint	8394	0.9%	2.0%	0.0%	A	1000000
2	Users	8395	0.9%	2.0%	6.1%	A	1000000
2	Xydev	8397	3.5%	7.8%	17.1%	A	1000000
1	System	8389	20.0%	20.0%	10.2%	G	40000
2	Admin	8306	15.0%	75.0%	0.3%	A	1000000
2	Ce	8355	5.0%	25.0%	1.7%	A	1000000
1	Unknown	8393	0.0%	0.0%	0.0%	GlZs	40000
2	unknown	12	0.0%	0.0%	0.0%	AlNclZs	1000000

Example 1: Shrtree display of SHAREBYACCT Hierarchy.

/etc/acid, and /etc/group to a private directory. (If UNICOS 8.3, also copy the /etc/udb_2 directory and its contents.) Prepare an initial udbgen input directives file by executing "udbsee -p. -a -fupdate,shflags,resgrp,uid,shares,acids > local". By changing the *resgrp* and *acids[]* fields in the local file and using "udbgen -p. local", the local UDB will be modified. (Initial udbgen directives for new entries should begin with create instead of update.) The *shflags* field is 0 for a user, 40000 for a resource group, and 1000000 for a shareholder (i.e.acid) entry.

The "shrtree -p. -F ..." command was created in order to verify whether the desired political structure has been imposed on the local UDB. Examples 2. & 3. show first the shrtree output from analyzing the UDB on machine "hot" and second the use of "shrtree -f" and a grep to find the UDB entry causing a really bad error. Errors flagged by shrtree can be corrected by modifying the local file of udbgen directives and repeating the process until shrtree reports the local UDB is free from errors. Some manual editing of /etc/acid may be required to eliminate spurious account-id definitions.

Translating Political Goals Into UDB Entries-Stage III

- Apply to real UDB and turn on NOSCHED
- Watch lnode creation with shrview/shrmon
- Coordinate new account processing
- Make modifications until satisfied

When ready to observe live lnode creation, apply the prepared list of udbgen directives in the local file to the real UDB by executing "udbgen local" without the "-p.". Using "shradmin -F ...", set Share scheduling flags including NOSCHED, to desired configuration. For example, setting SHAREBYACCT, ADJGROUPS, and NOSCHED requires executing "shradmin -F 032". Although it is possible to turn FSS on on-the-fly, shutdown to single user state is recommended. After turning on FSS with "shradmin -F ...", the shrview and shrmon commands provide the capability for monitoring lnode creation. There will be some user processes remaining attached to the Root lnode for a while after turning on FSS. When jobs that were running under the Root lnode when FSS was turned on complete, processes should be attached to terminal lnodes only. While testing FSS, the shrdaemon should also be brought up to ensure harvesting lnodes. At normal system startup, shrdaemon and shradmin entries in the /etc/config/daemons file configure FSS automatically.

```
DISPLAY OF SHARE TREE
---------------------
UDB path:       DEFAULT
Analyzed:       By UID
Format:         Groups only
Maxgroups:      4
Node:           ALL
Group Count:    8
Account Count:  1
User Count:     1656
Warnings:       37
Errors:         2

Warning Count:  1    (Nc)   Group has no references
Warning Count:  12   (Rl)   User referenced ROOT directly
Warning Count:  23   (Zs)   User has zero shares
Warning Count:  1    (Zs)   Group has zero shares
Error Count:    1    (Ng)   User referenced missing group
Error Count:    1    (Ng)   Group does not exist but is used
```

Lv	Name	ID	System Share	Group Share	System Usage	Status	Flags
0	_ROOT_	0	100.0%	100.0%	100.0%	G	40000
1	_ERROR_	8363	0.0%	0.0%	0.0%	A|Ng|Zs	0
1	System	8389	0.8%	0.8%	0.0%	G	40000
1	Users	8395	7.6%	7.6%	63.3%	G	40000
2	CCN	8354	0.2%	3.0%	3.7%	G	40000
2	Mktg	8381	0.2%	3.0%	26.1%	G	40000
3	Demos	8367	0.0%	1.3%	0.0%	G	40000
2	SoftDev	8386	0.2%	3.0%	27.4%	G	40000
2	Unknown	8393	0.0%	0.0%	0.0%	G|Nc	40000

Example 2: Shrtree display of SHARE-BY-UID Hierarchy.

Procedures must be put in place to maintain the FSS hierarchy in the UDB. Personnel responsible for setting up new accounts must be educated as to the rules for assigning users to resource groups and initializing acid lists. A frequent situation that results from the *resgrp* UDB field being left zero when a new user account is created is that the new user logs in and becomes entitled to 50% of the machine. Needless to say the new user is happy but other users experience a marked drop in responsiveness. Example 4. shows how to find these "Root linked" users with the shrtree command. When making modifications to correct Share hierarchy definition problems in the UDB, it is important to remember that processes created under an erroneous lnode chain must fully exit the system and the lnode harvested before a new lnode chain can be created that shows the effect of the UDB changes.

Selecting FSS Parameters To Effect Policy-Crucial Shradmin Parameters for Tuning

After removing the NOSCHED flag with shradmin, FSS will begin to affect process priorities. It is reasonable to expect a period of tuning experiments to select an appropriate combination of shradmin parameters before achieving desired FSS behavior. Of the approximately 20 shradmin parameters, there are 4 that should NOT be left set at their default settings (Table 1). Most important is the "-K" parameter which controls how long "usage" will be remembered. At the default setting of 60 seconds, there exists no usage history for FSS to consider when adjusting process priority. At a minimum, the "-K" parameter should be set to 1 hour (3600 seconds) for normal FSS operation. Based on information from sites running FSS, typical "-K"

Table 1: Shradmin Parameters Requiring Adjustment

Parameter	Description	Value	Crucial
(-K)	Usage decay rate (seconds)	3600.0	MINIMUM
(-U)	Max usage	1.0e+14	> 8 CPUs
(-Y)	Minimum group share	0.75	ADJGROUPS
(-G)	Max groups	4	_ROOT_+3

```
# ~kcz/cmd/shrtree/shrtree -f |grep Ng
Error Count:  1    (Ng) User referenced missing group
Error Count:  1    (Ng) Group does not exist but is used

1        _ERROR_          8363      0.0%      0.0%      0.0%      AlNg|Zs      0
2          n5827          27870     0.0%      100.0%    0.1%      UlNg         0

User Entry n5827 Refers To Non-existent Resource Group

# udbsee -v n5827 |grep 8363
      resgrp        :8363: # uid
```

Example 3: Using Shrtree to Find Really Bad Errors

parameter values seem to be of two flavors; either 6-8 hours or 3-5 days.

If running FSS with the LIMSHARE flag set, consideration must be given to the shradmin "-U" parameter which specifies a "ceiling" or "cap" on the amount of usage that will be considered when calculating process share priority. On machines with 8 or more CPU's or sites that charge for other things besides CPU ticks (I/O, system calls, and especially memory), maximum usage (-U) has to be set higher so that heavy users do not get a "free ride". The "shrview -ds" command can be used periodically to detect "clipping" of maximum usage. The "Usage:" line of this display shows the highwater usage for that sample period next to the value of the "-U" parameter. If the highwater usage is close to the maximum usage, the shradmin "-U" parameter should be increased.

If running FSS with the ADJGROUPS flag set, consideration must be given to the "-Y" parameter which specifies at what point FSS becomes concerned that a resource group is receiving less than its entitlement. If very close adherence to the machine share specified by the resource group lnode is desired, "-Y" must be set close to 1.0; for example 0.90. The default setting of 0.75 says that FSS will not become concerned about adjusting lnodes unless a group is getting less than 75% of its entitlement. Last, but not least, the "-G" parameter specifies the depth of the Share hierarchy; the default setting of 4 only allows 3 levels below "Root". Most sites recently modifying their UDB to support FSS have been ambitious in defining their Share hierarchy; more than one site designed a Share hierarchy requiring 7 levels.

Conclusion

Based on recent experience, the process of designing and implementing a Share hierarchy in the UDB to support the use of FSS in a complex diverse user community is best approached in stages. Political scheduling goals must be clearly defined before modifying the UDB. A new Share command (shrtree) can be used to evaluate UDB Share hierarchy definitions and point out errors before actually running "live". By following the process described in this paper, the transition to using FSS for managing different priority work can be made with relative ease.

```
# ~kcz/cmd/shrtree/shrtree -f |grep Rl
Warning Count: 12    (Rl) User referenced ROOT directly

1    root          0        8.4%    8.4%    0.0%    UlRl    0
1    jwhite        4401     7.6%    7.6%    0.0%    UlRl    0
1    n5828         27871    7.6%    7.6%    0.0%    UlRl    0
1    n5829         27874    7.6%    7.6%    0.0%    UlRl    0
1    n5830         27875    7.6%    7.6%    0.0%    UlRl    0
1    n5831         27876    7.6%    7.6%    0.0%    UlRl    0
1    n5832         27877    7.6%    7.6%    0.0%    UlRl    0
1    n5833         27878    7.6%    7.6%    0.0%    UlRl    0
1    n5834         27879    7.6%    7.6%    0.0%    UlRl    0
1    n5853         27913    7.6%    7.6%    0.0%    UlRl    0
1    n5854         27914    7.6%    7.6%    0.0%    UlRl    0
1    n5855         27915    7.6%    7.6%    0.0%    UlRl    0
                                                    Resource Group 0
```

Example 4: Using Shrtree to Find User Entries With