CS 6400 Service Considerations

Nicole Gerard, Electricité de France, Direction des Etudes et Recherches, 1, av. du Général de Gaulle, 92141 Clamart Cedex, France

ABSTRACT: *Electricité de France (EDF) was the first CS6400 site in Europe. The paper briefly describes the new functions undertaken by EDF, why the CS6400 was chosen to satisfy these, and the hardware and software configuration of the system. The developments required to provide management control are described together with the way these have been implemented. In particular, we focus on cpu and memory usage control, along with cpu partitioning.*

Introduction

In 1993, the *Direction des Etudes et Recherches* (DER) of the French Power Company, *Electricité de France*, took the decision to reduce all of its data processing costs, and in particular, the hidden costs of over-centralized data processing. We will explain this decision, and the method chosen to reach this goal.

CS6400 - Why ?

The Data Processing Centre in 1993

The Data Processing Centre included several different technical environments:

- Bull Dps7 for the internal administration of the DER,
- IBM ES9000 for management applications and technical codes,
- Cray YMP for scientific computing,
- General UNIX servers for DNS, MAIL, NEWS, and Internet access.

The clients of the Data Processing Centre consisted, on one hand, of DER researchers (1200 workstations), and on the other hand, of some applications from other EDF Divisions.

The study conducted by the DER in 1993

This study conducted into UNIX environments, helped stress some important facts:

- The full cost of the UNIX workstations and departmental servers (hardware and software purchases, maintenance, premises, administration, etc.) connected to the YMPs is almost equivalent to the total cost of the supercomputers themselves,
- The power of those workstations is probably under use,
- A significant proportion of small programs running on the Crays do not use the vectorial possibilities of the machine.

Therefore, we conclude that the DER had the possibility to redistribute the workload to make a better use of the available power on the site.

The strategic plan

To redistribute the workload, we followed a five-points plan of action:

- Supplying a powerful centralized UNIX server,
- Running the smaller scalar codes of the supercomputers on this low cost server, hence reducing the workload of the Y-MPs,
- Porting the Bull/Gcos7 applications to UNIX. Since this porting was based on a client/server architecture, it helped standardize the overall computing environment (the users did not have UNIX workstations by the time),
- Encouraging the IBM/MVS users to take the same route as Bull, for similar reasons,
- Attracting workstation users by offering them better resources and services:
  - ➤ Backup management,
  - ➤ Large disk storage capacity, and eventually migration and archiving tools,
  - ➤ Large processing capacity, and greater memory resources than those available on any workstation,
  - ➤ Very large catalogue of self-service products, thanks to centralized licence management,
  - ➤ 24 hours a day, 7 days a week operation.

The choice of the CS6400

The first point of the strategic plan was the choice of a UNIX server which could host both management and scientific applications.

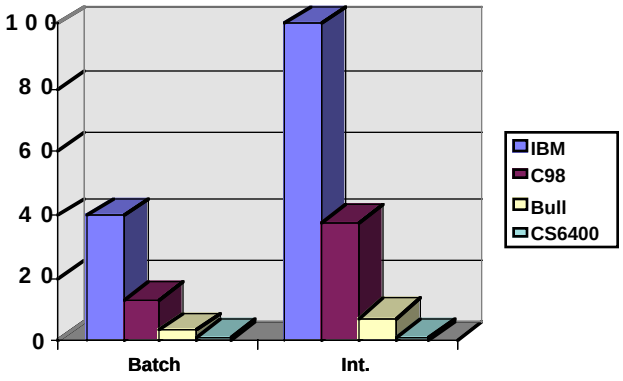
The CS6400 has been chosen for the following reasons:

- Very good performance as an ORACLE server,
- The possibility to run moderately parallel codes,
- 60% of the DER's workstations are SUNs running SunOS or Solaris. This avoids several compatibility issues,
- A much lower price than conventional mainframes.

The attraction of cheap processing

Based on the principle that "you cannot trap flies with vinegar", the Data Processing Centre offers assistance in code porting for users who want it as well as attractive pricing on the CS6400.

For instance, the graph below shows that for 50 seconds of CPU time used up on a weekday, the customers pay $100 under TSO on IBM, $38 for the interactive mode on the C98, and ... $1.30 on the CS6400.



Prices in $ for daytime processing
(batch and interactive)

Configuration

Hardware

- 8 System boards,
- 32 Viking processors (60 MHz),
- 4 Gbytes Memory,
- 220 Gbytes Disks

  38 Elite 3 disks (~110Gb)
  12 Elite 9 disks (~108 Gb),
- Ethernet and FDDI network interfaces.

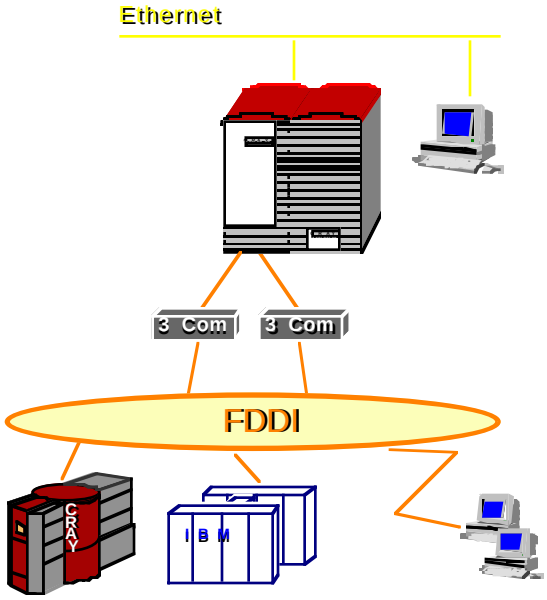
Network

The CS6400 is connected to

- the SSP through a private Ethernet link,
- the FDDI network of the Data Processing Centre's mainframes with two 3Com Hubs (Redundant Single-Attach Station).

Software

The operating system is Solaris 2.3 (the upgrade to Solaris 2.4 is planned for November 1995).



Network Configuration

The *Online: DiskSuite 2.0.1* product is used for mirroring `/usr`, `/var`, and `/opt` among others, for concatenation, and for its *Hot Spare* facilities. For `root`, a disk backup is taken every evening.

Moreover, the CS6400 offers a wide range of products:

- Administration, monitoring, and performances:
  - **NQE**, from CraySoft,
  - **NetWorker**, from Legato,
  - **PerfStat**, from Instrumental Inc,
  - **DB Control**, from Alcane,
  - **Tivoli Works**, from Tivoli,
  - **Operation Center**, from HP.
- Databases and software engineering workshops:
  - **Oracle**, from Oracle Corp.,
  - **TeamWork**, from Cadre Technologies.
- Compilers and scientific libraries:
  - **F77**, **C**, and **C++**, from SunSoft,
  - **F90**, from CraySoft,
  - **Libsci**, from Craysoft, **IMSL**, from Visual Numerics,
  - **PVM** (Public Domain),
  - **SparcWorks**, from SunSoft.
- Graphic development support tools:
  - **Motif**, and **SDK**, from SunSoft,
  - **OPGS**, from G5G,
  - **XRT**, from KL Group Inc.
- Engineering tools:
  - **SAS**, from SAS Institute,
  - **AVS**, from AVS Inc.,
  - **I-DEAS**, from SDRC,
  - **INSIGHT**, from Cray Research Inc,
  - ...

Limits and Partitioning - Why ?

In a system running both strategic business applications and research programs, it is necessary to protect each others by:

- partitioning processors (Cray),
- managin batch job (NQE),
- limiting CPU time and memory space usage in interactive mode.

The next sections describe the technical implementation of the limits and the partitions, along with the constraints emerging from this setup.

Limits

To limit the usage of some resources for a process and its children, we can modify the following parameters:

cputime	Maximum CPU seconds per process,
filesize	Largest single file allowed,
datasize	Maximum data size (including stack) for the process,
stacksize	Maximum stack size for the process,
coredumpsize	Maximum size of a core dump file,
descriptors	Maximum number of file descriptors,
memorysize	Maximum size of virtual memory.

For each resource, there is a **hard** and a **soft** limit. The rules for setting these values are as follow:

- Any user can set the soft limit to a lower value than the hard one,
- Any user can lower a hard limit,
- Only the super-user can increase a hard limit.

Examples: Limiting processing time to 1 hour:

	sh & **ksh**	**csh**
hard	ulimit -tH 3600	limit -h cputime 1:0
soft	ulimit -tS 3600	limit cputime 1:0

Notice that these limits can be viewed and changed through the `getrlimit(2)` and `setrlimit(2)` system calls respectively.

The requirements of the DER

Those requirements are the following:

- Allocating memory and CPU default limits to all users,
- Allowing the possibility to handle exceptions easily,
- Only limiting **cputime**, **datasize**, **stacksize**, and **memorysize** resources.

Developments

The main idea was to develop the **`cshlim`**, **`shlim`**, and **`kshlim`** C programs – for the **`csh`**, **`sh`**, and **`ksh`** *shells* respectively – which, after the desired limits have been set, execute the user's requested *shell*. These programs use the `setrlimit(2)` and `execve(2)` system calls, and replace the standard *shells* in the last field of `/etc/passwd`.

Example :

```
Tim:x:101:100:/home/Tim:/etc/opt/edf/kshlim
```

The parameter setting file **`limit.cf`** handles exceptions.

On execution, the new *shell* searches for the user's login name in the **`limit.cf`** file. If it finds it, it allocates the values found in the file. Otherwise, it allocates the C programs default values (128 Mbytes of memory, and 1 hour of CPU time).

`limit.cf` *file description :*

```
login_name   cpu_limit   memory_limit
```

• login_name	As it exists in `/etc/passwd`
• cpu_limit	CPU time limit, or if 0 : program default value if -1 : unlimited
• memory_limit	Memory space limit, or if 0 : program default value if -1 : unlimited

Some applications require special limits. These are stored in the **`appli.cf`** file.

Upon creation of a user's account, if he/she works on an application defined in this file, the limits of the application are automatically transferred to the `limit.cf` file.

`appli.cf` *file description :*

```
appli_name   cpu_limit   memory_limit
```

• appli_name	Application name
• cpu_limit	CPU time limit, or if 0 : program default value if -1 : unlimited
• memory_limit	Memory space limit, or if 0 : program default value if -1 : unlimited

Effects on other files

The names of the new *shells* (`/etc/opt/edf/cshlim`, `/etc/opt/edf/shlim`, and `/etc/opt/edf/kshlim`) must be defined in the **`/etc/shells`** file to enable the **`ftp`** access, and in **`/etc/profile`**, in order to display **`/etc/motd`** and execute the **`quota`** and **`mail`** commands.

Limits & cron

For authorized users, **`cron`** can execute commands at periodic dates and times.

Example of a user's cron file:

```
5 0 * * * /tmp/Tim_cron
```

The authorization of users is handled by the `cron.allow` and `cron.deny` files, located in the `/etc/cron.d` directory.

The reasons for restricted availability

By default, **`cron`** submits commands with the ***Bourne shell***. The resources limits normally allocated to the user by the "limited" *shells* are not therefore taken into account.

Example :

- Tim displays its hardware limits using the `ulimit`. command (3600 seconds cpu time, 128 Kbytes of memory)

```
Tim@dragon> ulimit -aH
time(seconds)          3600
file(blocks)             unlimited
data(kbytes)           131072
stack(kbytes)          131072
coredump(blocks)       unlimited
nofiles(descriptors)   64
vmemory(kbytes)        131072
```

- Then he submits this command via the **`cron`**, with a **`Tim_cron file`**,

```
Tim@dragon> more /tmp/Tim_cron
ulimit -aH
```

- He creates the cron parameter file.

```
Tim@dragon> crontab -l
39 * * * * /tmp/Tim_cron 1>/tmp/Tim_cron.o 2>&1
```

- After submission, Tim verifies the results:

```
Tim@dragon> more /tmp/Tim_cron.o
time(seconds)          unlimited
file(blocks)           unlimited
data(kbytes)           2097148
stack(kbytes)          8192
coredump(blocks)       unlimited
nofiles(descriptors)   64
memory(kbytes)         unlimited
```

On executing `/tmp/Tim_cron`, the interactive limits are not set !!

Actually, to use the "limited" *shell*, the user must modify his/her cron file, as in the following example:

```
Tim@dragon> crontab -l
46 * * * * /etc/opt/edf/kshlim   /tmp/Tim_cron 1>/tmp/Tim_cron.o 2>&1
Tim@dragon>more /tmp/Tim_cron.o
time(seconds)          3600
file(blocks)           unlimited
data(kbytes)           131072
stack(kbytes)          131072
coredump(blocks)       unlimited
nofiles(descriptors)   64
vmemory(kbytes)        131072
```

The rule to follow for users who are authorized to use the **`cron`**, is to specify one of the "limited" *shells*: `/etc/opt/edf/cshlim`, `/etc/opt/edf/kshlim`, or `/etc/opt/edf/shlim`.

We did not wish to develop additional interfaces to offer widespread access to this tool.

Constraints

- A user who normally uses **`cron`** mechanisms on his/her workstation cannot do so on the CS6400.
- The administrator must determine if the **`cron`** is essential to the user who requests it. He must make sure that the rule of good conduct is observed, and finally, he has to manage the **`cron.allow`** file.

Limits & at

Authorized users may submit an **`at`** command at dates and times of their choice, without passing through the intermediary of a parameter setting file.

The syntax of the `at` command is the following:

```
at [-csm] [-qqueue] time [date] [+ increment] < script
```

Examples :

```
at 0815am Jan 24  <  Tim_at
at now + 1 minute  <  Tim_at
```

The authorization of users is managed with the **`at.allow`** and **`at.deny`** files, located in the **`/etc/cron.d`** directory.

The command submitted by **`at`** is converted into a "job" stored in the directory **`/var/spool/cron/atjobs`** under the name ***`timestamp.q`***, where ***`timestamp`*** is the date in seconds (since 1st January 1970) at which the job must be submitted, and ***`q`*** is the name of the queue (***`a`*** by default).

In order to simplify, and only describe what has been installed, let us say that the **`/etc/cron.d/.proto`** file allows the administrator to customize **`at`** command by controlling the information written in the **`at`** ***`timestamp.q`*** job file.

The file **`/etc/cron.d/.proto`** contains the following lines as standard:

```
#ident"@(#).proto   1.3   89/12/12 SMI"   /* SVr4.0 1.2  */
cd $d(Replacement of $d with the current directory name)
umask $m(Replacement of $m with the current creation mask)
$<   (Reading from the standard input until EOF is reached)
```

*Example of conversion of the command **`/tmp/Tim_at`** to an **`at`** job :*

```
Tim@dragon> more /tmp/Tim_at
ulimit -aH
Tim@dragon> at now  + 3 minutes  < /tmp/Tim_at
warning: commands will be executed using /etc/opt/edf/kshlim
job 797505120.a at Mon Apr 10 11:12:00 1995
Root #cd /usr/spool/cron/atjobs
     more 797505120.a
: at job
: jobname: stdin
: notify by mail: no
export _; _='/usr/bin/at'
export PATH; PATH='/usr/bin:/usr/sbin:/usr/openwin/bin:
export OPENWINHOME; OPENWINHOME='/usr/openwin'
export LOGNAME; LOGNAME=Tim
```

```
export SHELL; SHELL='/etc/opt/edf/kshlim'
export PWD; PWD='/home/Tim'
export ...
$SHELL << '...the rest of this file is shell input'
#ident--"@(#).proto     1.3     89/12/12 SMI"   /* SVr4.0 1.2   */

cd /home/Tim
umask 27
ulimit -aH
```

Moreover, notice that:

- the commands submitted via **`at`** can be directed into 26 queues, named from **a** to **z**,
- a **`/etc/cron.d/.proto.q`** file (where ***q*** is the name of the queue) may be associated with each of these queues,.by default, the **`etc/cron.d/.proto`** file is used,
- it is possible to monitor the queued jobs with the **`/etc/cron.d/queuedefs`** file.

The **`at`** command is not suited to hande periodic jobs. However, one can develop a command which will reschedule itself. Thus, it can act as a substitute for **`cron`**.

Example :

```
Tim@dragon> more /tmp/Tim_at
ulimit -aH
echo "/tmp/Tim_at " | at 1900 thursday next week
```

The reasons for open availability

`At` submits each command with the *shell* specified in the $SHELL variable. If this variable is not set, `/bin/sh` is used. If the variable is set to **login shell**, the limits are taken into account.

Example :

```
Tim@dragon> at now  + 3 minutes  < /tmp/Tim_at
warning: commands will be executed using /etc/opt/edf/kshlim
job 797505120.a at Mon Apr 10 11:12:00 1995
Tim@dragon> mail
Your "at" job "/var/spool/cron/atjobs/797505120.a" produced the
following output:
     time(seconds)          3600
     file(blocks)           unlimited
     data(kbytes)           131072
     stack(kbytes)          131072
     coredump(block)        unlimited
     nofiles(descriptors)   64
     vmemory(kbytes)        131072
```

As with the **`cron`**, one may indicate in which *shell* he/she wishes to run commands. This is done with the **`-s`** (shell) or **`-c`** (cshell) options of the **`at`** command or by setting the $SHELL variable. Hence, the user's limits are not taken into account.

Example :

```
Tim@dragon> at -s now  + 3 minutes  < /tmp/Tim_at
warning: commands will be executed using /bin/sh
job 797505240.a at Mon Apr 10 11:14:00 1995
Tim@dragon> mail
Your "at" job "/var/spool/cron/atjobs/797505240.a" produced the
following output:
     time(seconds)          unlimited
     file(blocks)           unlimited
     data(kbytes)           131072
     stack(kbytes)          131072
     coredump(blocks)       unlimited
     nofiles(descriptors)   64
     vmemory(kbytes)        unlimited
```

The existence of **`/etc/cron.d/.proto`** allowed us to customize the generated jobs, and to control the *shell* used.

This file currently contains the following lines:

```
#ident"@(#).proto   1.3   89/12/12 SMI"   /* SVr4.0 1.2  */
/etc/opt/edf/proto
cd $d
umask $m
$<
```

Here, the script **`/etc/opt/edf/proto`** (developed by EDF) fetches the name of the submitter of the command. Then, with this name, it gets the name of the login *shell* in `/etc/passwd`, and compares this *shell* to the execution *shell*. If they are not similar, the job is aborted and a message is sent to the user.

`/etc/opt/edf/proto` *script:*

```
#!/bin/ksh
LANCE_SHELL=$(ps -f -p $PPID | tail -1 | awk '{print $8}')
if [ "$LOGNAME" ]
  then
        LOGIN_SHELL=$(/usr/bin/grep  "^$LOGNAME:"  /etc/passwd  |
/usr/bin/cut -f7 -d ":")
else
    /usr/bin/echo"La variable $LOGNAME n'a pas de valeur,
 veuillez vous identifier ?  "
       kill -9 $PPID
fi

if [ $LOGIN_SHELL != $LANCE_SHELL ]
  then
    /usr/bin/echo "Le shell d'exécution ne correspond pas au login shell
 utilisés dans la commande at"
    kill -9 $PPID
fi
```

Example :

```
Tim@dragon> at -s  now  + 1 minutes  < /tmp/Tim_at
warning: commands will be executed using /bin/sh
job 797505900.a at Mon Apr 10 11:25:00 1995
Tim@dragon> mail
…
Your "at" job "/var/spool/cron/atjobs/797505900.a"
produced the following output:
Le shell d'exécution ne correspond pas au login shell
Killed
```

Constraints

- This configuration imposes few constraints on the user, and no special request has been received yet.
- However, the administrator must manage the standard **`/etc/cron.d/.proto`** script (upgrade, reinstallation), and the home-made **`/etc/opt/edf/proto`** script.

Limits & NQE

By default, NQE submits a request using the user's login *shell*, so the limits are taken into account.

Example:

```
Tim@dragon> more /tmp/Tim_nqe
echo $SHELL
ulimit -aH
Tim@dragon> qsub /tmp/ Tim_nqe
nqs-181 qsub: INFO
  Request<5311.dragon>:
Submitted to queue<nqebatch> by Tim(107)>.
Tim@dragon> more Tim_nqe.o5311
    /etc/opt/edf/kshlim
    time(seconds)3600
    file(blocks)204800
    data(kbytes)131072
    stack(kbytes)131072
    coredump(blocks)1048576
    nofiles(descriptors)64
    memory(kbytes) 131072
```

In order to get the optimal resource setting with the different NQE queues (day, night, weekend), the user's login *shell* limits must be cleared, and the *shell* to be used must be included in the request itself. We achieve this with the following NQE directive:

```
#QSUB -s shell_name
```

Example:

```
Tim@dragon> more /tmp/Tim_nqe
#QSUB -s /bin/ksh
echo $SHELL
ulimit -aH
Tim@dragon> qsub /tmp/ Tim_nqe
nqs-181 qsub: INFO
  Request<5312.dragon>:
Submitted to queue<nqebatch> by Tim(107)>.
Tim@dragon> more Tim_nqe.o5312
    /bin/ksh
    time(seconds)10800
    file(blocks)204800
    data(kbytes)unlimited
    stack(kbytes)unlimited
    coredump(blocks)1048576
    nofiles(descriptors)64
    memory(kbytes) 524288
```

Constraints

The user must not omit the QSUB directive.

Partitioning

The Cray CS6400 offers the possibility to create logical "partitions" by regrouping processors.

From the point of view of the scheduling, each partition is managed separately.

The system administrator can assign a set of processes to a particular partition. Also, users can choose their own partitions assuming they have the necessary permissions.

The number of partitions is limited by the number of processors available on the machine.

Partition attributes

attribute B "can Borrow":

This attribute determines whether idle processors within a partition, can temporarily borrow processes from other partitions.

However, the tasks normally assigned to this partition take priority over those coming from other partitions.

attribute L "can Loan":

This attribute determines whether processors in the partition can loan processes to idle processors in other partitions.

attribute S "Sbus Interrupt Servicing":

This attribute defines whether the partition can service Sbus interrupts.

By default, the processors in the system partition can "borrow", "loan", and "serve", while those in other partitions cannot.

The attributes are set with the **`partn`** command.

Examples :

```
partn -s -a +S+L+B -p3#Circus
```

Partition access permissions

Each partition has a corresponding system file, named **`/devices/pseudo/partn@0:[0-63]`**, along with a symbolic link on **`/dev/partn/[0-63]`**.

The standard conventions of Unix permissions apply to these files. Thus, we can define access rights by assigning an owner and a group to the file, with the read, write and execute permissions for the owner, the group and the rest of the world.

We can give a meaningful name to a partition, by creating a symbolic link between this name and the file **`/devices/pseudo/partn@0:[0-63].`**

Implementation

Partition 0 is defined as the system partition. It has no name nor any special group name associated with it.

Partition 1 is available to all users (non-dedicated)

- name *p1-user*
- group *p1-user*

- rights read and write for everyone
- attributes can Loan, can Borrow, can Serve Sbus Interrupts

Currently, the **three other partitions** are dedicated to strategic applications.

The procedure adopted to create a dedicated partition is the following:

- Choose a partition name, and use it to create the **`/dev/partn/`*`px-name`*** symbolic link. To simplify, this name will be the same as the name of the unix group,
- Create the group ***px-name*** in **`/etc/group`**,
- Set the permissions (owner *root*; group *px-name*) and the attributes of the partition,
- Modify the partition creation script, which is located in **`/etc/rc2.d/S??partn`**, in order to create the partition at each start-up of the CS6400.

Example:

```
#!/bin/sh
echo "Creation of partition 3."
for cpu in 11 12 13
  do
     if [ "`psrinfo -s $cpu 2> /dev/null`" -eq 1 ]
        then
             partn -s -f -p3 $cpu
     fi
  done
echo " Modification of partition 3 attributes"
partn -s -a +S+L+B -p3
ln -s /devices/pseudo/partn@0:3 /dev/partn/p3-circ 2>/dev/null
chgrp p3-circ /devices/pseudo/partn@0:3
chmod g=rw  /devices/pseudo/partn@0:3
```

EDF developments

The goal is to place the user in the proper partition at login. Since his/her *shell* is in the right partition, all the child processes will be forked in the same partition, and the user will not have to take any special action.

The re-use of the programs developed for assigning limits seemed a straightforward solution. The system calls used are `partn_open(3)`, `process_assign_to_partn(3)`, and `partn_close(3).`

Using the parameter setting file **`limit.cf`**, we place each user into his/her partition.

On execution, the program (**`cshlim`**, **`shlim`**, **`kshlim`**) searches for the user's login name in **`limit.cf`**. If it finds it, it puts the user's *shell* in the requested partition. Otherwise, he/she goes into ***p1-user*** partition.

limit.cf *file description :*

login_name	cpu_limit	memory_limit	**partition_name**

- login_name As it exists in `/etc/passwd`,
- partition_name Partition in which the user has to work.

Using the same mecanisms as for limits management, the partition name (which can be *p1-user*) is entered in the `appli.cf` file.

Upon creation of a user's account, if he/she works on an application defined in **`appli.cf`** file, the name of the dedicated partition of the application is automatically copied into the `limit.cf` file setting the user's partition. The unix group corresponding to the partition will be assigned to the user as a secondary group.

appli.cf *file description :*

appli_name	cpu_limit	memory_limit	**partition_name**

- appli_name Application name,
- partition_name Partition corresponding to the application.

Constraints

This form of management requires that a single application corresponds to a user name. This constraint is not troublesome because the applications attached to a dedicated partition are independant.

Tuning

It is very difficult to determine whether the number of processors per partition is optimal, since there exists very few tuning tools and paper covering this issue !!!

Conclusion

Thanks to the attractive pricelist and the wealth of products and services, lots of applications have been transfered to the CS6400, while others are being ported.

In order to fully succeed in our project, it is important that Solaris becomes more reliable (we are in Solaris 2.3), that better tuning tools appear on the market, and that we find efficient migration and archiving tool.

Acknowledgements

I would like to thank my colleagues at EDF for their help, suggestions, and review of this paper.

Bibliography

[1] *SunOS™5.3 Reference Manual Section 1. User Commands* - SunSoft.

[2] *Solaris 2.3 CRAY Version R - CRAY SUPERSERVER* CRAY RESEARCH Superservers, Inc