1. General information

The PBS server does the checkpoints in several circumstances e.g. when user suspends the job by running the qhold command. The full list of events related to checkpoint and restart is presented in PBS Administrator Guide.
The PBS supports two ways of using the checkpointing – the first one mentioned as OS specific is not configurable and works only on IRIX and XXX. The other way is called "site specific checkpoint" and is fully configurable. All the required changes are entered in the MOM configuration file. Integration of some 3rd party checkpointing software is done by (defining) the three mom actions[1]:

- action checkpoint: this action is called at several occasions, e.g. when user wants the PBS to checkpoint the application periodically – inside script called during this action you have to execute appropriate commands to take application checkpoint. The PBS passes to the action script several parameters that should be used to store the image. Especially important is appropriate handling of the parameter that implies the output directory because the PBS manages the images according to its internal algorithms.
- action checkpoint_abort: this action is called when the PBS wants to store the application image (eg when qhold is issued). The parameters passed to this are exactly the same as in case of *action checkpoint* with the exception that the PBS expects the script to kill the application after the checkpoint is saved. After execution of this script the application exists only as entry in the PBS database
- action restart: this action is called when PBS wants the application to be restarted from previously saved image. The PBS passes several parameters such as original identifier of the job and directory with the image (assuming that the checkpoint script used in proper way the parameters passed to action script when the checkpoint image was created). There is a special parameter (*restart_transmogrify*) influences the assumptions on the restart script behavior.

By default all scripts called as a result of above mentioned actions are executed with superuser permissions, so you have to be aware of this and manage all the rights on your own to avoid possible security threats.

2. Scenarios:

2.1. Periodic checkpoint

When submitting the job to PBS user can specify that the job should be checkpointed in regular time intervals. This can be done by submitting the job in such way:

q*sub –c c=<time interval> <PBS script>*

Where *time interval* is amount of time (in minutes) that has to pass between next checkpoint will be taken. By default the newly created checkpoint will overwrite the last one.

[1] Action is just a bash script called by PBS when the related event occurs.

2.2. Manual checkpoint

There is no way of forcing the PBS to checkpoint the job, however you can "freeze" the job by issuing the *qhold* command. The application will be checkpointed (if the PBS is configured appropriately) and killed. You unfreeze the application using the *qrls* command.

2.3. Automatic checkpoint executed when PBS is shut down.

<<<<<<<<<<<<<<< TO BE TESTED >>>>>>>>>>>>>>>

3. Example configuration.

In this chapter we will present the example integration of the PBS with the PSNC/R checkpointer. The testbed was a small 4cpu Altix machine running SLES 9 SP3 and PBS version XXX

3.1. Installation of the checkpointer

You can download the checkpointing package from http://checkpointing.psnc.pl. The packages on this site are binaries of a checkpointing toolkit for Altix systems customized for different kernels and ProPack versions. The package is available without any fees – you have only to register for the purpose of internal statistics. The installation instruction is also available to be downloaded from the same site.

3.2. PBS installation

3.3. PBS configuration.

We will only describe actions required to make the checkpointer work with the PBS. The suggestions on how to configure the PBS you can find in the PBS Administrator Guide.
The PBS/checkpointer integration required writing three scripts (*pbs-mom-checkpoint.sh*, *pbs-mom-checkpoint-and-stop.sh* and *pbs_restart_test.sh*) called by the MOM daemon and appropriate configuration of the MOM.
The example mom configuration file (in our case it was the */var/spool/PBS/mom_priv/config* file)

```
$clienthost altix
$restrict_user_maxsysid 499
$restart_background true
$action checkpoint 0 !/usr/pbs/bin/pbs-mom-checkpoint.sh %globid %jobid %sid %ta
skid %path
#$checkpoint_path       /var/spool/pbs_checkpoint
$action restart 0 !/usr/pbs/bin/pbs_restart_test.sh %path %taskid
$restart_transmogrify true
$action checkpoint_abort 0 !/usr/pbs/bin/pbs-mom-checkpoint-and-stop.sh %globid
%jobid %sid %taskid %path
```

The example action checkpoint script (*/usr/pbs/bin/pbs-mom-checkpoint.sh*).

```
#!/bin/sh
exec_pid=`/bin/ps -ef | /usr/bin/grep $3 | /usr/bin/awk '{ if ($3 == PATTERN) pr
int $2; }' PATTERN=$3`
echo "exec_pid: " $exec_pid >> $LOG
path=$5
/usr/local/bin/chkpnt -v -p $exec_pid -od $path >> $LOG
EXIT_STATUS=$?
echo "chkpnt EXIT_STATUS: "$EXIT_STATUS >> $LOG
echo "chkpnt stored in: "$path >> $LOG
```

This script is quite simple – the first line obtains the process id of the application process tree root, which is passed as a parameter to the *chkpnt* command. The additional lines are providing debug information and are optional.
The action script checkpoint_abort looks almost exactly the same as in case of action script checkpoint. The only difference is the "-k" switch passed to the *chkpnt* command indicating that the application should be killed after the checkpoint is saved, so the action checkpoint_abort look like this:

```
/usr/local/bin/chkpnt -k -v -p $exec_pid -od $path >> $LOG 2<&1
```

The script used to restart applications (*pbs_restart_test.sh*)

```
#!/bin/sh
LOG=/var/log/pbs.checkpoing.log
echo "USER = "$USER >> $LOG
echo "tryning to restart "$1 >> /var/log/pbs.checkpoing.log
echo "LOGNAME = "$LOGNAME >> /var/log/pbs.checkpoing.log
COM="/usr/local/bin/resume -drp -ve -v -od "$1
echo "COMMAND = "$COM >> /var/log/pbs.checkpoing.log
/bin/chown -R $USER $1 >> $LOG 2>&1
/bin/su - $USER -c "$COM" >> $LOG 2>&1
```

The script is not very long but requires a few words of comment.
As mentioned earlier, by default all the *$action* scripts are executed with root permissions which results, in case of this checkpointer, in the applications being restarted as a root applications (from the system OS point of view, the user still sees the job as his own when looking at the queue status by running the *qstat* command). In our opinion, in the desired scenario the application should work with the original user's credentials , so we had to switch the user context just before the application is restarted. The *resume* command is normal user's command so the user has to have access to the images. This is not possible by default, because the PBS stores the images as root files (due to security reasons). We had to change the ownership of the image files just before the su is executed.

4. Remarks

Restarting the applications using the PSNC/R checkpointer causes spawning additional process required to maintain the connection between terminal and restarted application. For example if you run an example *matrix* application in the PBS, the process might look like this:

```
27973 ?        00:00:00 bash
27991 ?        00:00:00  \_ 195.altix.SC
27993 ?        00:00:16      \_ matrix
```

There is also a 195.altix.SC process – this is script created by the PBS and will be also included in the image.
After checkpointing and restarting the application will look as follows:

```
28017 ?        00:00:00 su
28018 ?        00:00:00  \_ resume
28036 ?        00:00:00     \_ 195.altix.SC
28037 ?        00:00:07        \_ matrix
```

As you can see there are two additional processes – the “su” process is needed in order to ensure that the restarted application will run with the privileges of the original application – by default the PBS restarts the application with root privileges so we have to switch to original user before the application is restarted. The second “resume” process is a process added by the checkpointer – its only purpose is maintain the coherency of the recovered process tree. If there would be no such process, the PBS would lost the connection with the restarted application.

The checkpointer is well aware of those processes and if you will call qhold again it will skip both the “su” and the “resume” processes – they will not be included in the process image.

In order to make the whole deployment work we had to change the access mode of the directory where the images will be stored. By default this directory is: */var/spool/PBS/checkpoint/* and the default rights are not allowing users to list the directory . You should change the rights by issuing the command

chmod +x /var/spool/PBS/checkpoint/