Kernel Level Checkpoint Restart Functionality for SGI Altix systems

Poznań Supercomputing and Networking Center

Radosław Januszewski
radoslaw.januszewski@man.poznan.pl

SGI UG 2005, 13-16 June 2005

Introduction

- **Kernel level**
  - most transparent solution – require no changes in checkpointed application
- **User level (linkable library)**
  - require to recompile users applications
- **Application level**
  - require user to write all essential functionality

Target platform

All the development and tests efforts were performed on the Altix 3300 platform.

CPU: IA64 (4 x Itanium2)

OS: SGI ProPack v2.2, 2.3, 3.0 for Linux

ProPack 4.0

Kernel: 2.4.20-sgi220r3

2.4.21-sgi305r1

2.6.xx

Main features (1/5)

Support for multi processes programs

A user job can consist of many processes. Each process is allowed to possess any number of „children”.

Support for System V IPC

- semaphores
- message queues
- shared memory

Support for threads

Only for kernel 2.6

Main features (2/5)

Support for interactive programs

User applications can use terminals.

Support for signals

All the actions and resources connected with signals handling are stored/restored.

Support for programs linked statically

All statically linked libraries are stored/restored properly.

Support for programs linked dynamically

All memory segments mapped from dynamically linked libraries are stored/restored properly.

Main features (3/5)

Support for open files and open directories

During the restoring stage, all previously opened files and directories are reopened.

Support for mapped files

Memory segments that were explicitly mapped by the *mmap()* are properly stored/restored.

Support for *mprotect()* settings

All memory access control flags are stored/restored.

Support selected pseudo devices

/dev/null, /dev/zero, /dev/random.

Main features (4/5)

Support for STDIN / STDOUT / STDERR

Connections with standard input, output and error streams are stored/restored.

Emulation of „zombie" processes

If during checkpoint stage there are some „zombie" processes that constitute checkpointed job, after restoring stage the appearance of those „zombie" processes is emulated.

Support for program arguments

Vector of program arguments associated with each process is stored/restored.

Support for environment variables

Vector of environment variables associated with each process is stored/restored.

Main features (5/5)

Working directory

After the recovery stage, the working directory path is the same as during the checkpoint stage.

Settings made by *umask()* function

Umask settings are stored/restored.

Settings made by *nice()* function

Scheduling priority levels of processes are stored/resotred.

Settings made by *setrlimit()*

Resource limits set by means of *setrlimit()* are stored / restored..

Accounting data

They are stored and restored, but their sense can be a little violated.

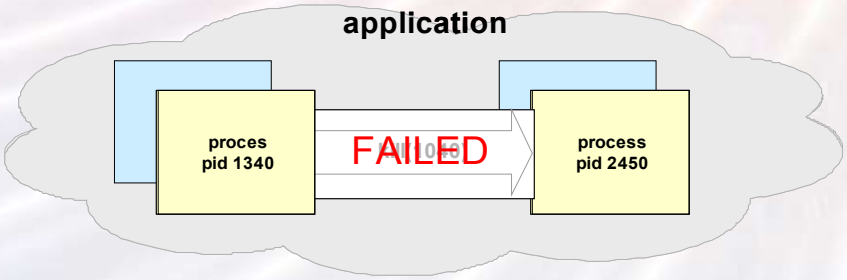
Problem

Even if the checkpoint image is created there is **no guarantee** of proper restart.

Unique resource identifiers

- Linux is not designed to support checkpointing
- After restart some resource identifiers may be busy/taken
- Problems with migration applications from smaller to larger computation nodes
- Operations on IPC resources are based on identifiers held outside kernel structures.

Basic scenario



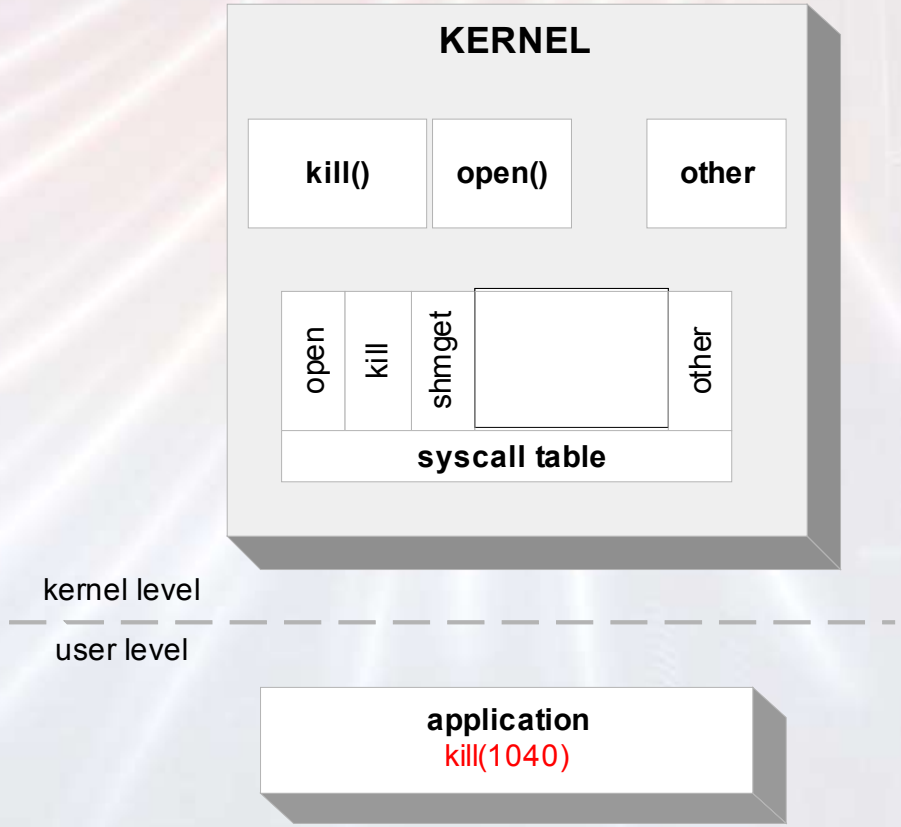
Solution

Resource identifier virtualisation

Resource voirtualisation - idea

Normal system call execution path

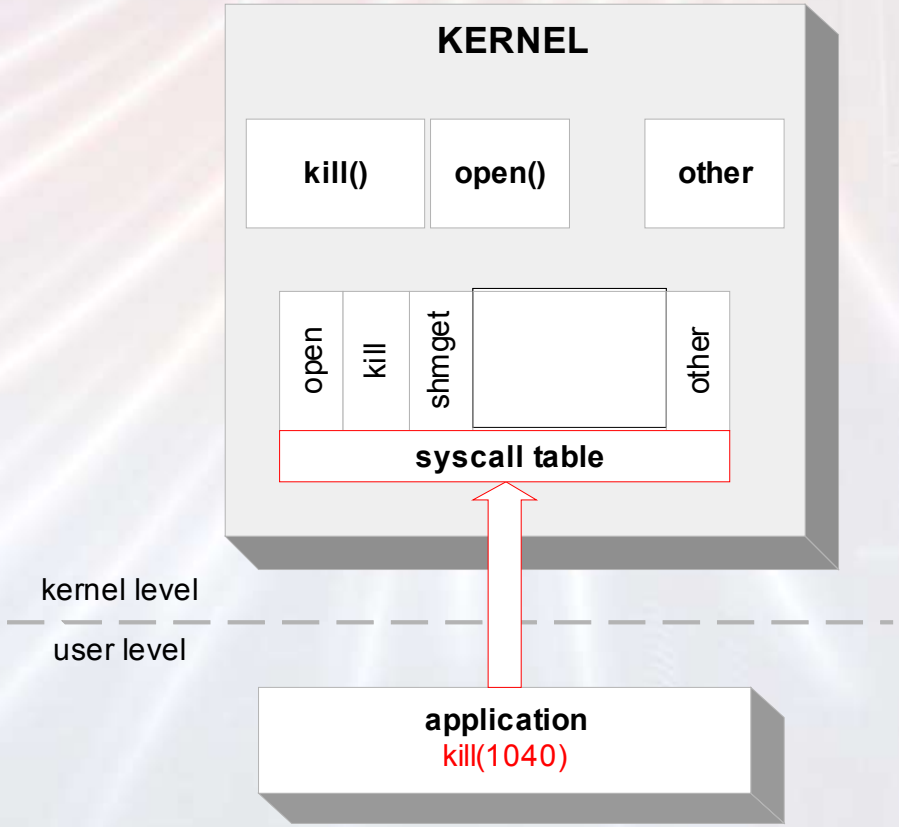
- **software interrupt**
- interrupt number translation
- execution of system call
- return result to application



Resource voirtualisation - idea

Normal system call execution path

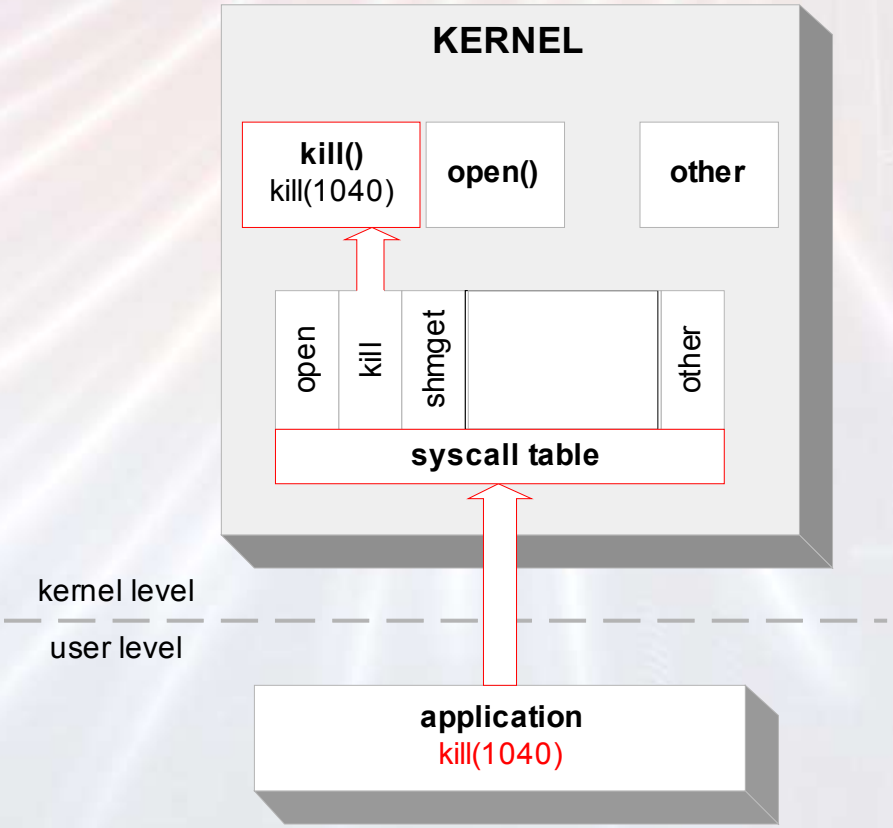
- software interrupt
- **interrupt number translation**
- execution of system call
- return result to application



Resource voirtualisation - idea

Normal system call execution path

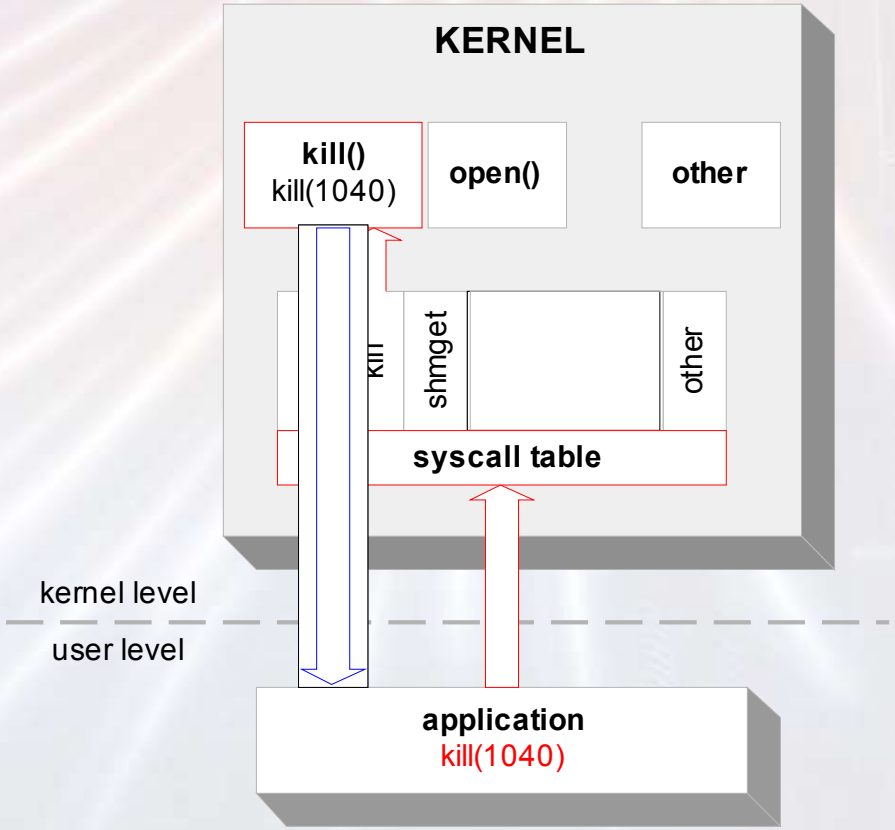
- software interrupt
- interrupt number translation
- **execution of system call**
- return result to application



Resource voirtualisation - idea

Normal system execution path

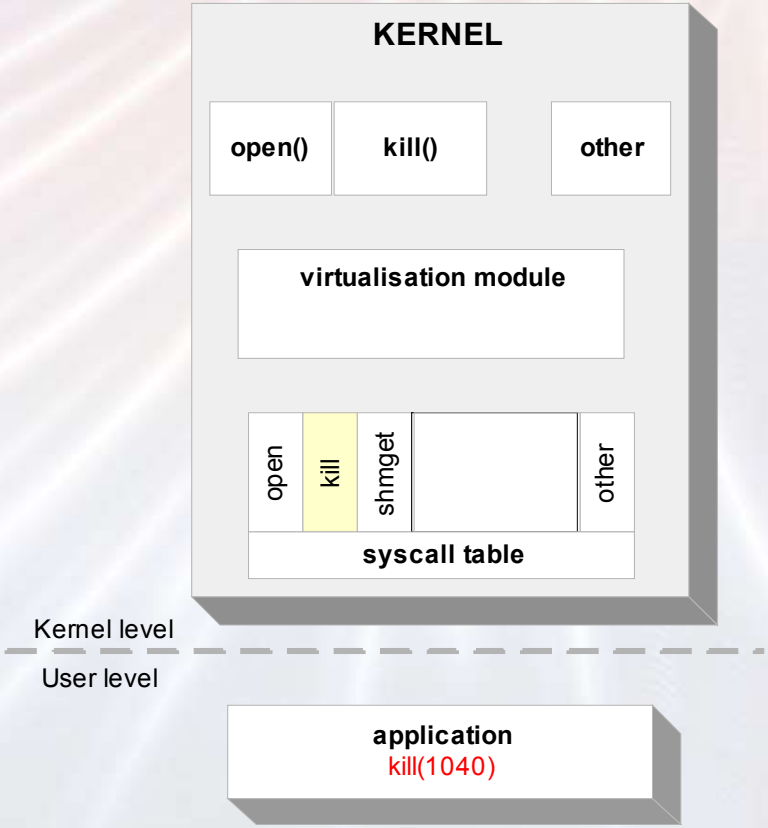
- software interrupt
- interrupt number translation
- execution of system call
- **return result to application**



Resource voirtualisation - idea

System call execution path with virtualisation

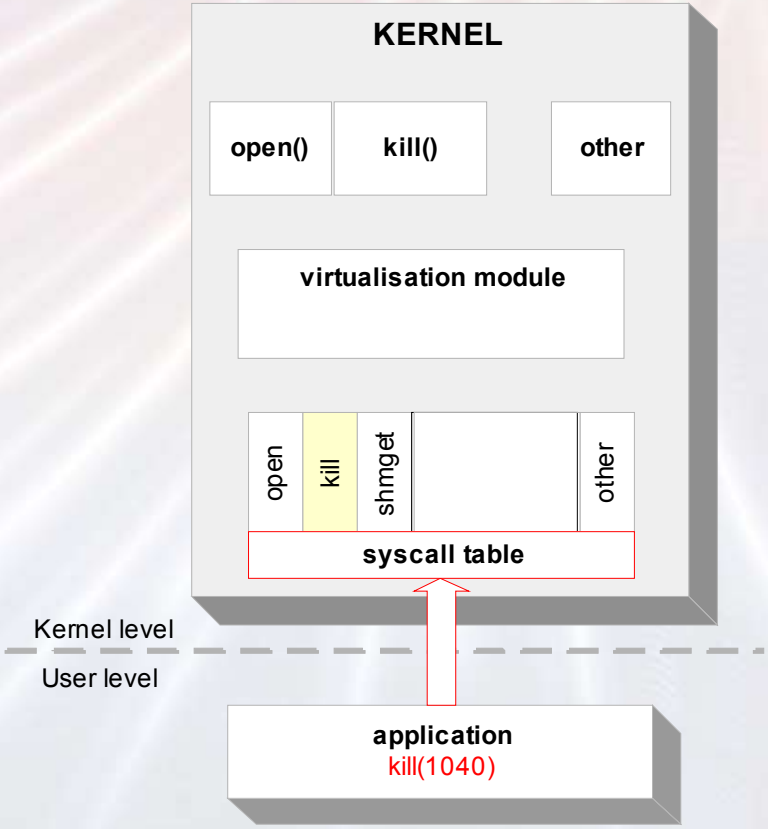
- **software interrupt**
- interrupt number translation
- parameter translation
- execution of system call
- result value translation
- Return result to application



Resource voirtualisation - idea

System call execution path with virtualisation

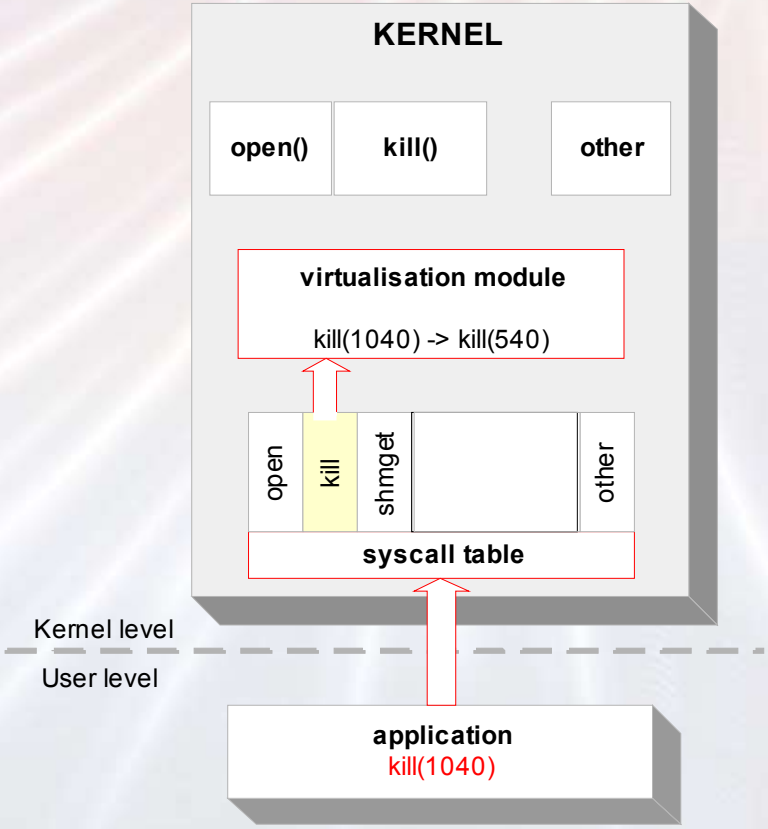
- software interrupt
- **interrupt number translation**
- parameter translation
- execution of system call
- result value translation
- return result to application



Resource voirtualisation - idea

System call execution path with virtualisation

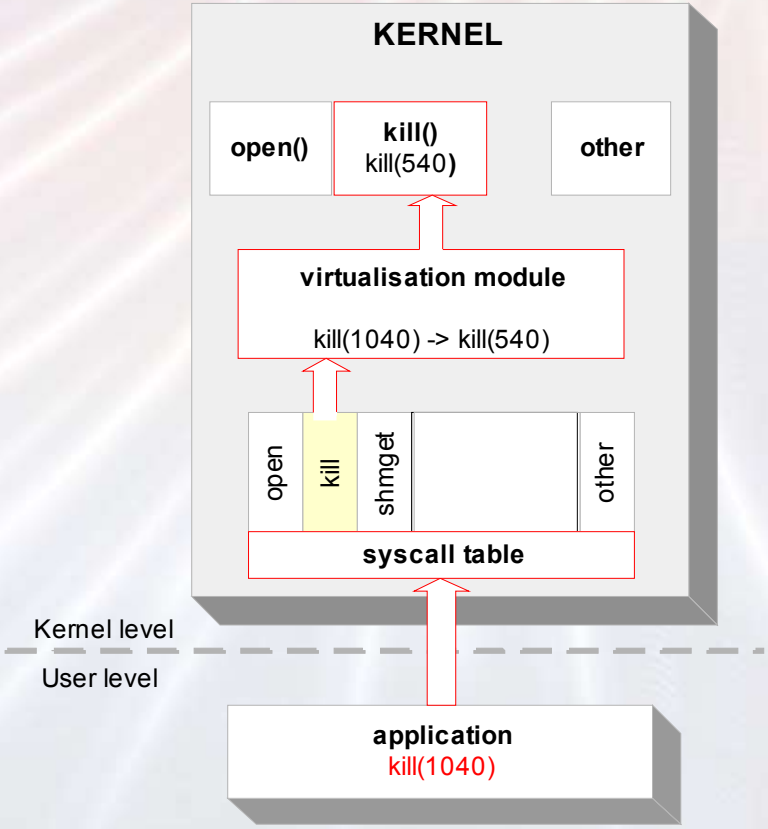
- software interrupt
- interrupt number translation
- **parameter translation**
- execution of system call
- result value translation
- return result to application



Resource voirtualisation - idea

System call execution path with virtualisation

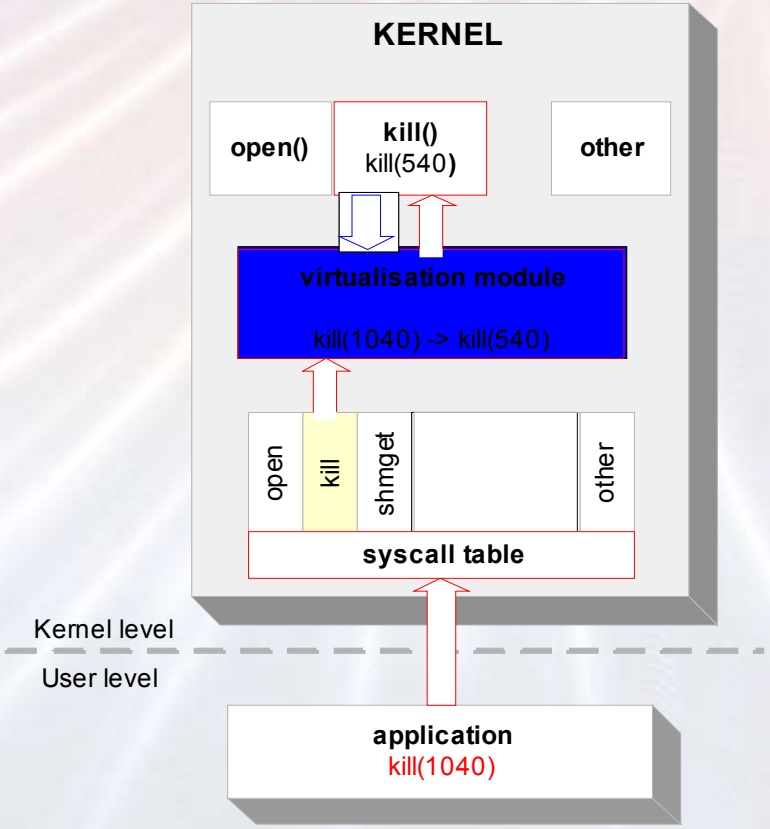
- software interrupt
- interrupt number translation
- parameter translation
- **execution of system call**
- result value translation
- return result to application



Resource voirtualisation - idea

System call execution path with virtualisation

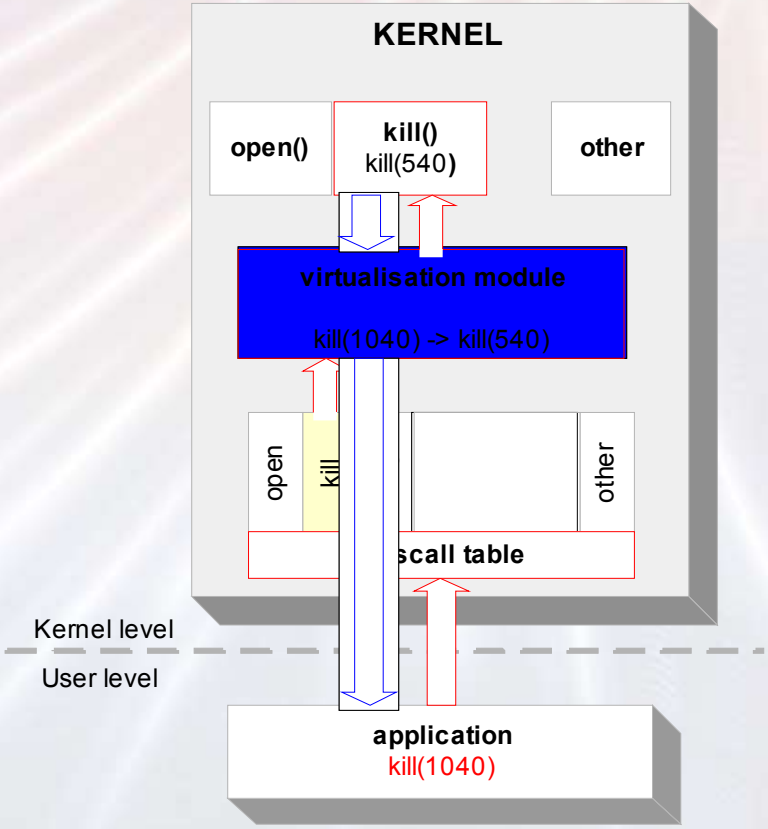
- software interrupt
- interrupt number translation
- parameter translation
- execution of system call
- **result value translation**
- return result to application



Resource virtualisation - idea

System call execution path with virtualisation

- software interrupt
- interrupt number translation
- parameter translation
- execution of system call
- result value translation
- **return result to application**



Virtualized resources

PID

After recovering, processes are deceived to possess the same PID as during the checkpoint stage. PID's virtualization allows to restore groups of processes and relationships between them (job tree).

PGID

Process group ID. After recovering, processes are deceived that PGID has not changed.

System V IPC's keys

Even if after the recovery stage, the previously used keys are occupied, it seems to the recovered program that it still uses the original key values.

System V IPC's identifiers

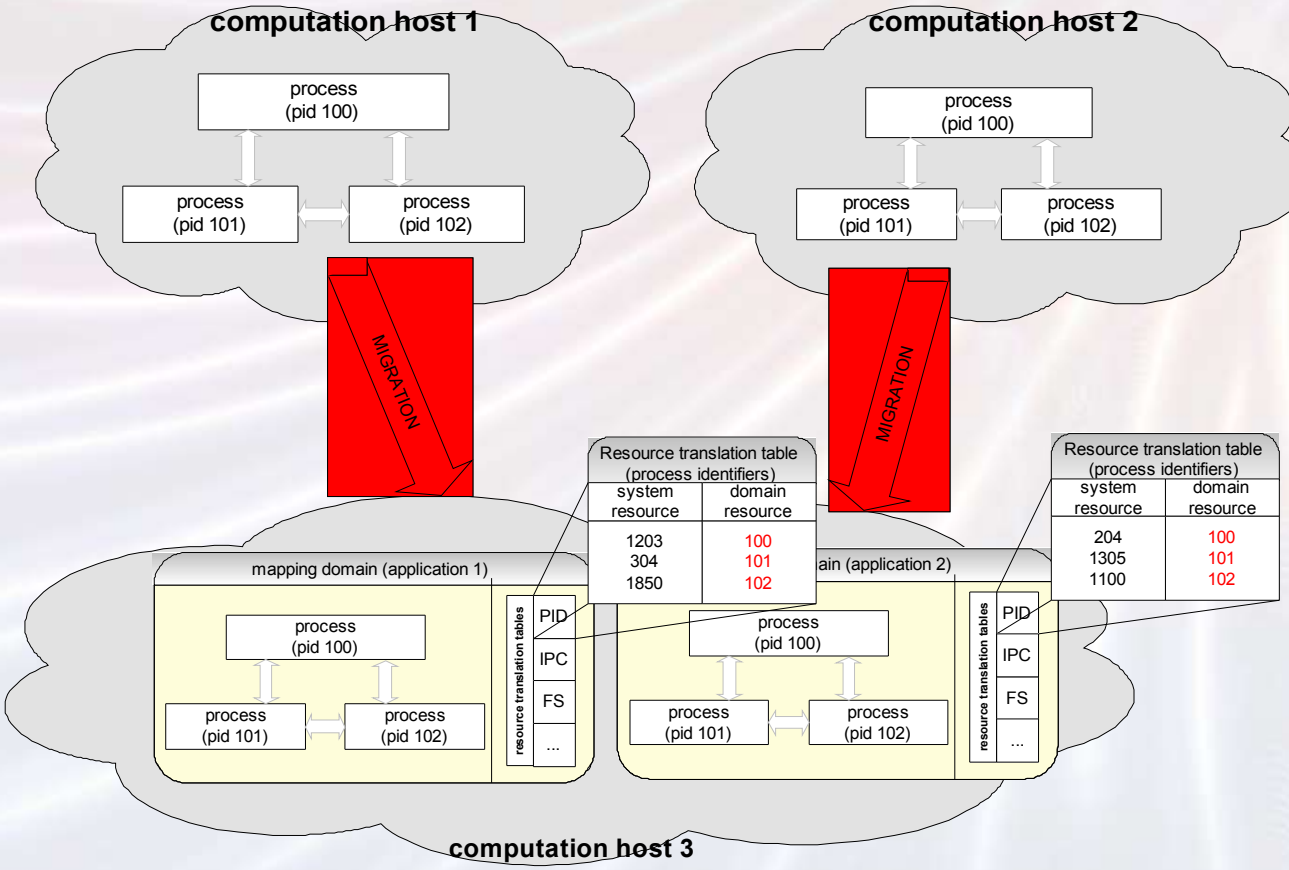
After the recovery stage, the identifiers values are changed, but the user programs are deceived that nothing has changed.

Resource virtualisation

Mapping domains

Every restarted application may be placed/encapsulated in a mapping domain which provides the application with its own set of identifiers.

Any process or thread created within a mapping domain will be also included into that mapping domain.


computation host 1
process (pid 100)
process (pid 101)
process (pid 102)
MIGRATION
computation host 2
process (pid 100)
process (pid 101)
process (pid 102)
MIGRATION
Resource translation table (process identifiers)
system resource
domain resource
1203
304
1850
100
101
102
Resource translation table (process identifiers)
system resource
domain resource
204
1305
1100
100
101
102
mapping domain (application 1)
process (pid 100)
process (pid 101)
process (pid 102)
resource translation tables
PID
IPC
FS
...
ain (application 2)
process (pid 100)
process (pid 101)
process (pid 102)
resource translation tables
PID
IPC
FS
...
computation host 3

Checkpoint-restart package content (1/2)

Kernel Level Elements

„ckpt_mod" module – Provides access to resources and information that are needed during the checkpoint and restart activity. If there was no virtualization in our package, it would be the only module.

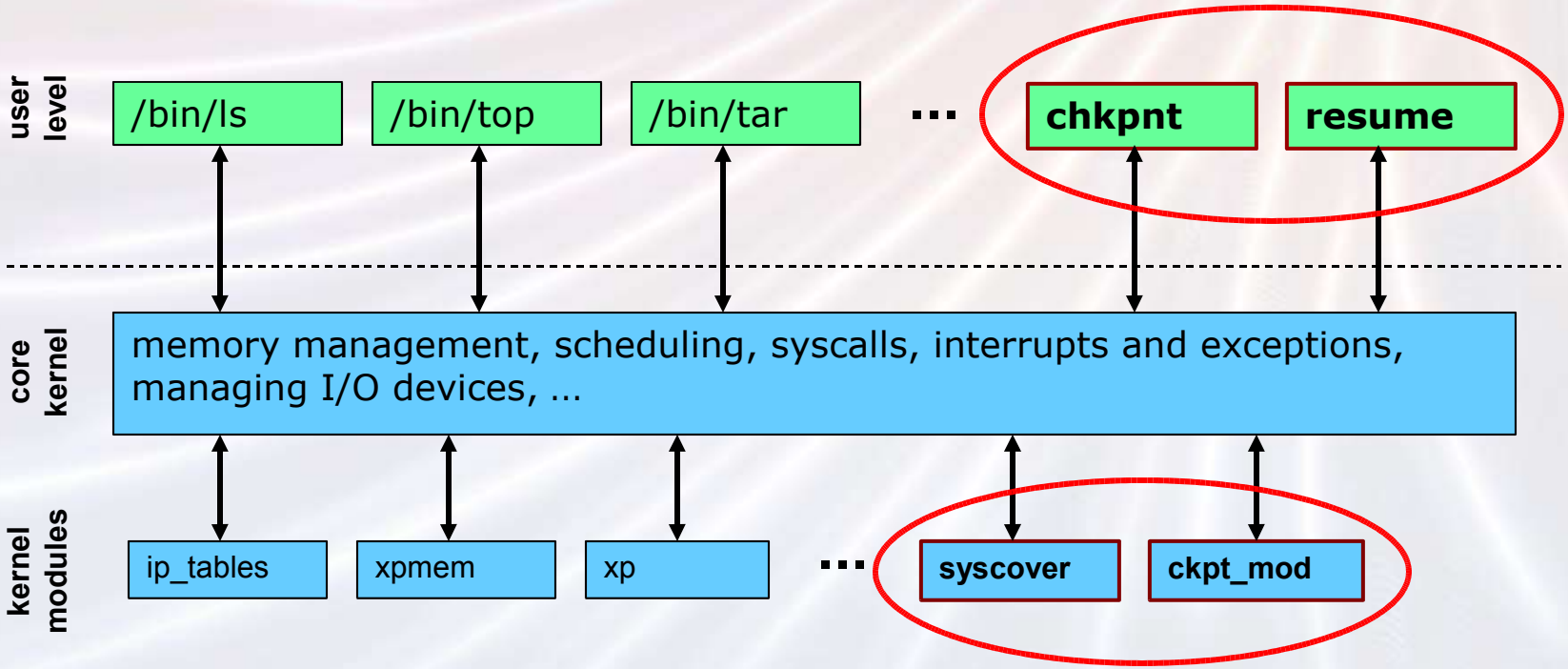
„syscover" module – Implements virtualization of identifiers of resources. Additionally it provides the mechanism for the „zombie" processes emulation.

User Level Elements

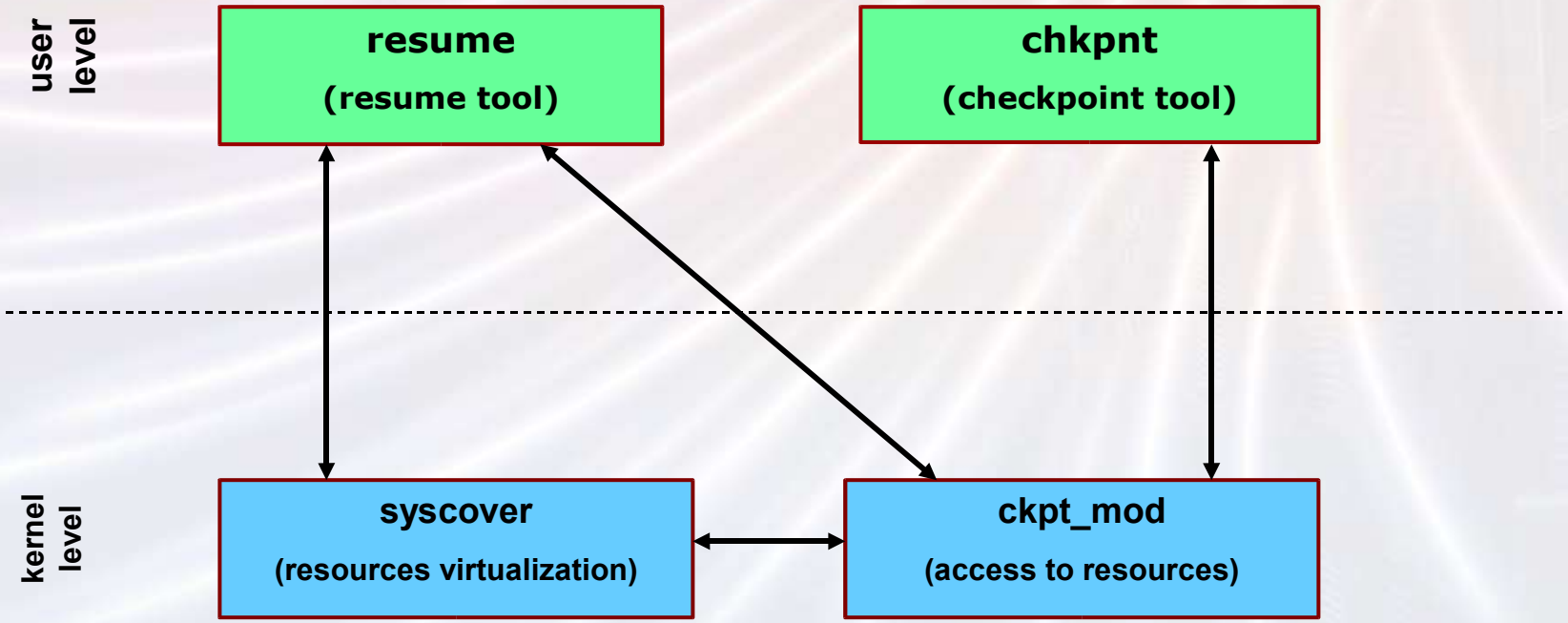
„chkpnt" tool –Tool which is executed by end user in order to do the checkpoint.

„resume" tool – Tool which is executed by user in order to restart a program.

Checkpoint-restart package content (2/2)

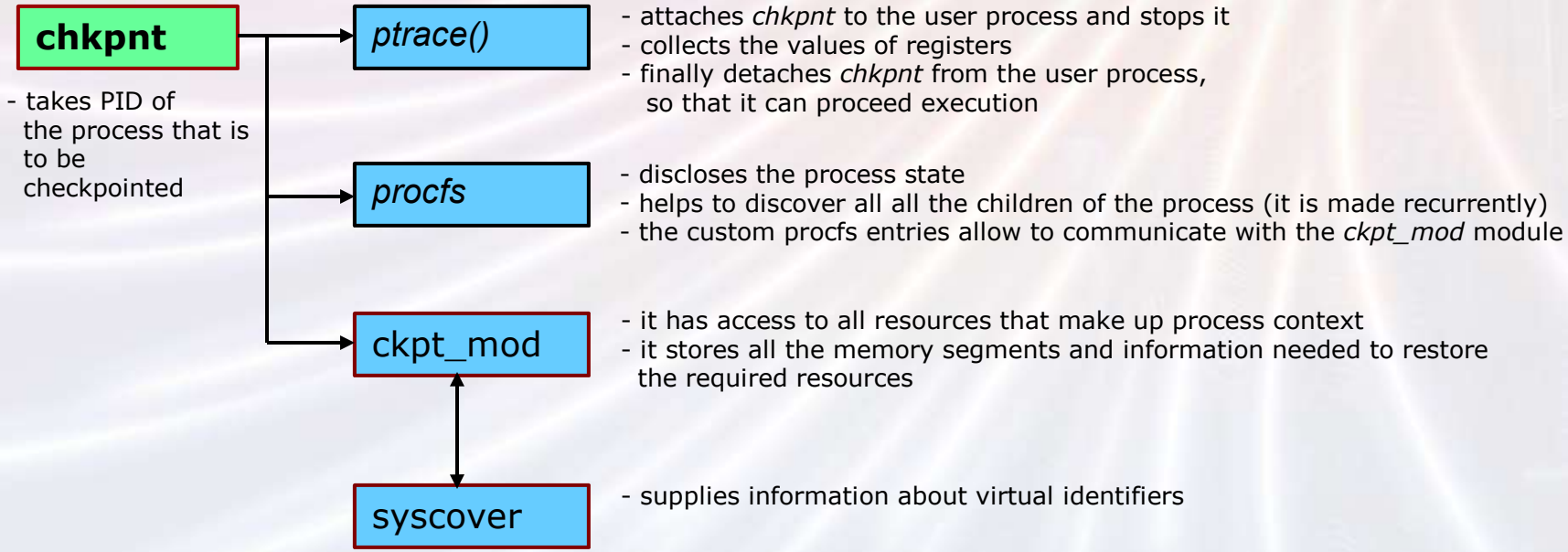


Dependencies between package elements



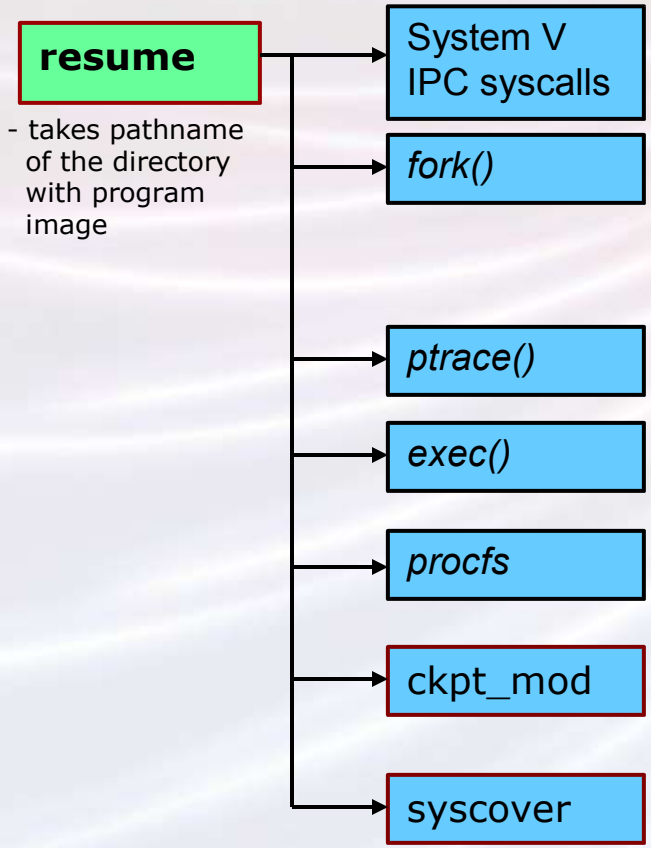
Storing processes

The main checkpoint labour is made by the *chkpnt* tool and the *ckpt_mod* module.



Restoring processes

The main restart labour is made by *resume* tool and *ckpt_mod* module.



- common syscalls are used to recreate System V IPC objects

- the *resume* tool calls *fork()* syscall to create the process that correspond to the process that is being restored
- to rebuild all the relationships between the processes, each newly created process calls *fork()* for each child that is to be recovered

barrier

- it is used to attach to each process that has been created in the previous step
- it is used to restore the registers of each process

- using original process binaries, it restores the .text segment

- the custom procfs entries are used as the interface to the *ckpt_mod* and *syscover* modules

- restores all memory segments (besides the .text)

- creates a new *mapping domain*, initializes the *mapping tables* associated with the virtualized identifiers
- initializes the structures needed for emulation of the „zombie" processes

To improve legibility some details, such as restoring signals actions, open files, working directory, etc. have been omitted.

psncC/R - Usage

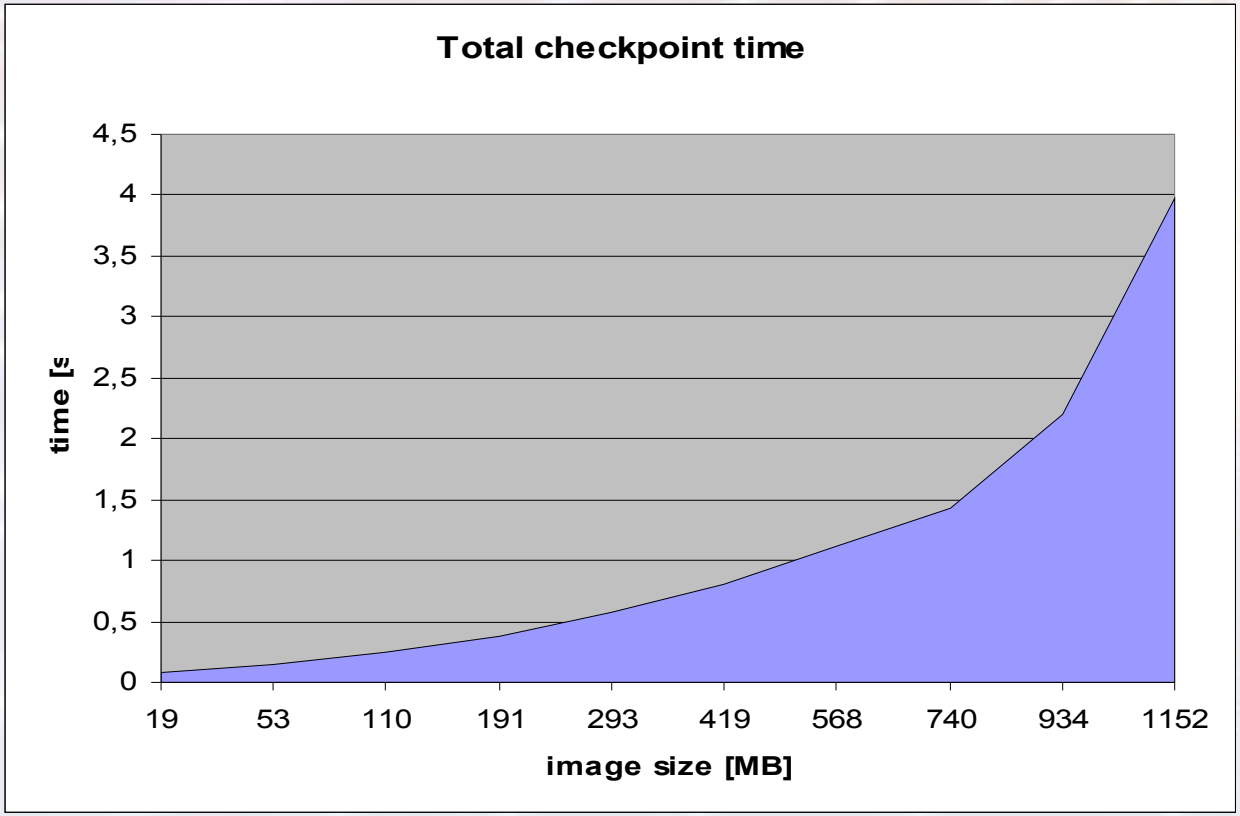
- To make checkpoint

  chkpnt –p <pid>

- To restore checkpoint

  resume –od <image_directory>

checkpoint performance



Summary

Capability	ppC/R
Changes in end user's code required.	no
Recompilation with additional libraries required.	no
Root privilages required in order to install checkpointing funcionality in system	yes
working directory recovery	yes
environment variables recovery	yes
command line arguments recovery	yes
signals (including pending signals, signal mask, signal disposition)	yes
simple devices like /dev/zero, /dev/null and simple procfs files	yes
reopening open directories	yes
reopening open files	yes
end user's processes can be linked statically and dynamically	yes
POSIX IPC	in plans
Threads	soon
support for zombie processes	yes
support for multi-process aplications	yes
PIDs and PGIDs virtualization	yes
System V IPC objects recovery	yes
System V IPC objects identifiers and keys virtualization.	yes

Summary

- The C/R mechanism for Altix systems is now available!
- You may try it without any changes in your applications.
- No changes in your system configuration required (kernel modules are loaded dynamicly)
- It can be downloaded from http://checkpointing.psnc.pl

THE END