Ghost Process: a Sound Basis to Implement Process Duplication, Migration and Checkpoint/Restart in Linux Clusters

Geoffroy Vallée (INRIA / ORNL / EDF), Renaud Lottiaux (INRIA), David Margery (INRIA), Christine Morin (INRIA), Jean-Yves Berthou (EDF)

ISPDC 2005, July 2005
Context

- Application execution on clusters
  - Efficient use of resources
  - Needs mechanisms for global process management
    (e.g. process placement, migration, checkpoint/restart)
- Several systems provide such mechanisms
  - Checkpoint/restart: BLCR, EPCKPT, ...etc.
  - Process migration: OpenMosix, OpenSSI, ...etc.
Global Process Management - Approach

- All mechanisms have a common concept: process virtualization

- But no system provides a single mechanisms to implement mechanisms of global process management
Ghost Processes

• Goals
  • Extract a process for a local system
  • Create of image allowing to execute the process independently to its location
  • Provide a set of interfaces to ease the implementation of new mechanisms for global process management

• Two main issues
  • What data is to extract from the system to have an image of the process?
  • How do we deal with this data set to create a new create a new mechanism for global process management?
Process Virtualization – Data Extraction

- Define the set of information to extract a process
  - memory
  - disk
  - registers

- Kernel implementation
  - memory/disk information is directly available
  - registers: only available during specific states of the kernel
Extraction of Registers Values

• Constraints
  • be able to notify the process extraction at any time
  • independent mechanisms
  • lightweight modification

• Similar to the signal treatment

• Kernel hooks to create a new state where value of registers are available (about 10 lines of assembler)
Mechanisms for Global Process Management

• Some interfaces to ease the use of ghost processes
  • export/import interface
    – export = process virtualization (using a resource plug-in)
    – import = process creation from an image (using a resource plug-in)
  • interface to plug resource plug-ins

/* Export a process */
int export_process(ghost, process);
/* Import a process */
int import_process(ghost);
/* Plug a resource object to a ghost process */
int plug_resource_object(class_id, ghost);
Process Virtualization & Resources

• Many resources can be used for global process management
  • Network for process migration
  • Files/memory for process checkpoint/restart
• How to specify which resource to use?
  • Resource plug-in for Ghost Processes
Resource Plug-Ins

- Allows to associate a resource to a ghost process
- A plug-in is identified by a unique identifier (predefined)
  - send/receive through the network
  - read/write in memory
  - read/write on disk
- To create a new resource plug-in, system programmers have to extend the ghost process mechanism
Resource Plug-In – Example

• File plug-in

/* Initialization function */
file* file_open(pathname);
/* Read function */
int file_read(file, destination, size);
/* Write function */
int file_write(file, source, size);
/* Finalization function */
int file_close(file);
/* Function to plug the file access method in an instance of a ghost process */
int associate_file_to_ghost(file, ghost);
Implementation of a Process Checkpoint Mechanism on Disk

- Simple algorithm: extraction of the process and storage on the local disk

```c
ghost_t disk_checkpoint (pid) {
    file = file_open (pathname);
    process = find_task_by_pid (pid);
    ghost = create_new_ghost ();
    plug_resource_interface (DISK_WRITE, ghost);
    associate_file_to_ghost (file, ghost);
    export_process (ghost, process);
}

void disk_restart (disk_checkpt_id) {
    if (process_state == running) then
        destroy_process ();
        ghost = create_new_ghost ();
        plug_resource_interface (DISK_READ, ghost);
        file = find_file_checkpoint (disk_checkpt_id);
        associate_file_to_ghost (file, ghost);
        import_process (ghost);
    }
```
Experiments

• Results for
  • process migration (using the network)
  • process checkpoint/restart
    – using local disks
    – using local memory
  • presentation of process checkpoint/restart on disk results only

• PIII, 1GHz, 512MB of memory, 100Mps Ethernet network
• Evaluation of the ghost process cost when used to implement a mechanism for global process management
Experiments - Size of Ghost Processes

- Application: Modified Gram-Schmidt
  - Produce from a set of vectors an orthonormal basis of the space generated by these vectors

<table>
<thead>
<tr>
<th>Matrix Size</th>
<th>Ghost Size (Kbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500x500</td>
<td>4,229</td>
</tr>
<tr>
<td>750x750</td>
<td>9,354</td>
</tr>
<tr>
<td>1000x1000</td>
<td>12,429</td>
</tr>
<tr>
<td>1250x1250</td>
<td>20,629</td>
</tr>
<tr>
<td>1500x1500</td>
<td>24,729</td>
</tr>
<tr>
<td>1750x1750</td>
<td>28,833</td>
</tr>
<tr>
<td>2000x2000</td>
<td>32,933</td>
</tr>
</tbody>
</table>
Checkpoint on Disk - Results

[Bar chart showing time in milliseconds for different resource access times]
Restart from Disk - Results

• Results with cold cache

![Bar chart showing restart and importation times for different data sizes](chart.png)
Conclusion

- Ghost processes ease implementation of mechanisms for global process management
  - full service of process export/import
  - resource plug-ins

- Ghost processes ease maintenance
  - improving efficiency of ghost processes => improve efficiency of all mechanisms for global process management
Conclusion (2)

• Prototype
  • included in Kerrighed/SSI-OSCAR
    – Kerrighed 1.0.2 (http://www.kerrighed.org/)
    – SSI-OSCAR 3.0 (http://ssi-oscar.irisa.fr/)
  • used to implement process migration, duplication, checkpoint/restart

• Future work
  • Adapt to new resources: e.g. new network technologies like Myrinet
  • Study a common kernel patch for all systems providing mechanisms for global process management