

Processor-Level Partitioning of Kernel and User Mode Functionality

Introduction

- We propose implementation of a tool that permits clear partitioning of User and Kernel functionality on Linux/SMP systems.
- We intend to show that this tool may lead to more effective utilization of available resources.
- This presentation shall outline our design objectives and rationale.

Hypothesis

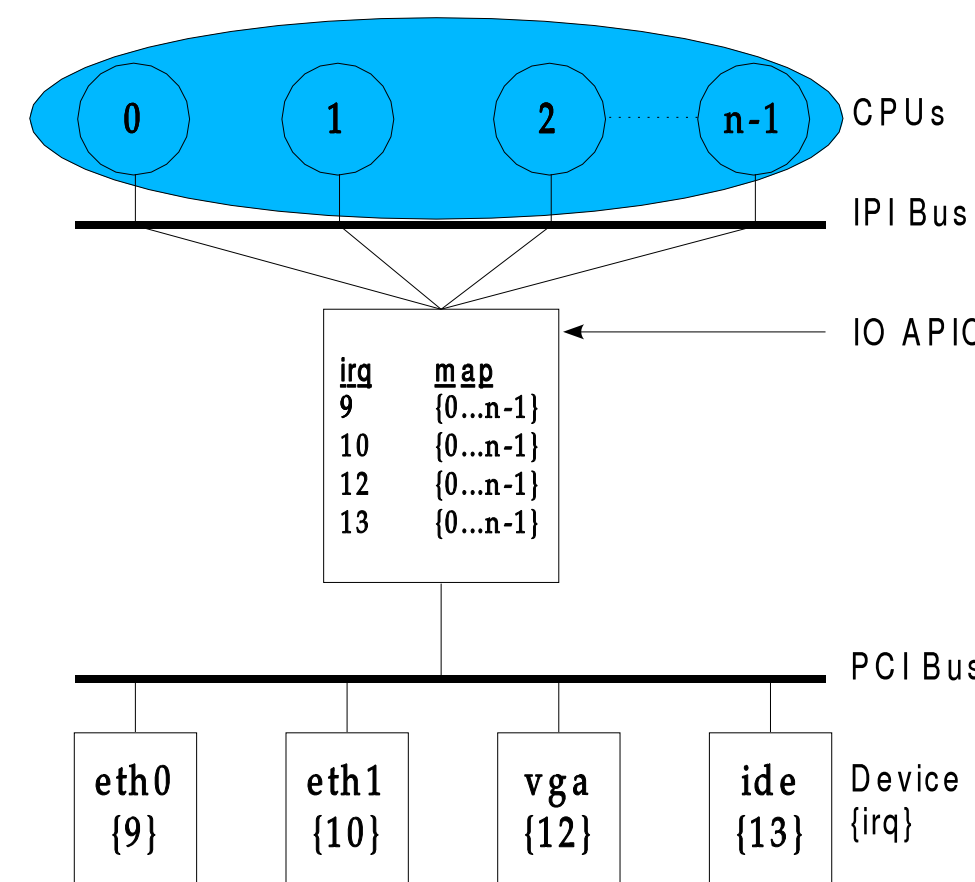
Processor-level partitioning of User and Kernel functionality will smooth execution times of parallel algorithms, enhance reproducibility of results, and may provide more effective utilization of system resources on large scale SMP systems. We believe that our tool will accomplish these goals by providing the following:

- Elimination of interrupt overhead on user tasks.
- Elimination of the arbitrary interleaving of user and kernel task execution.
- Giving preference to user tasks that do not require kernel services.

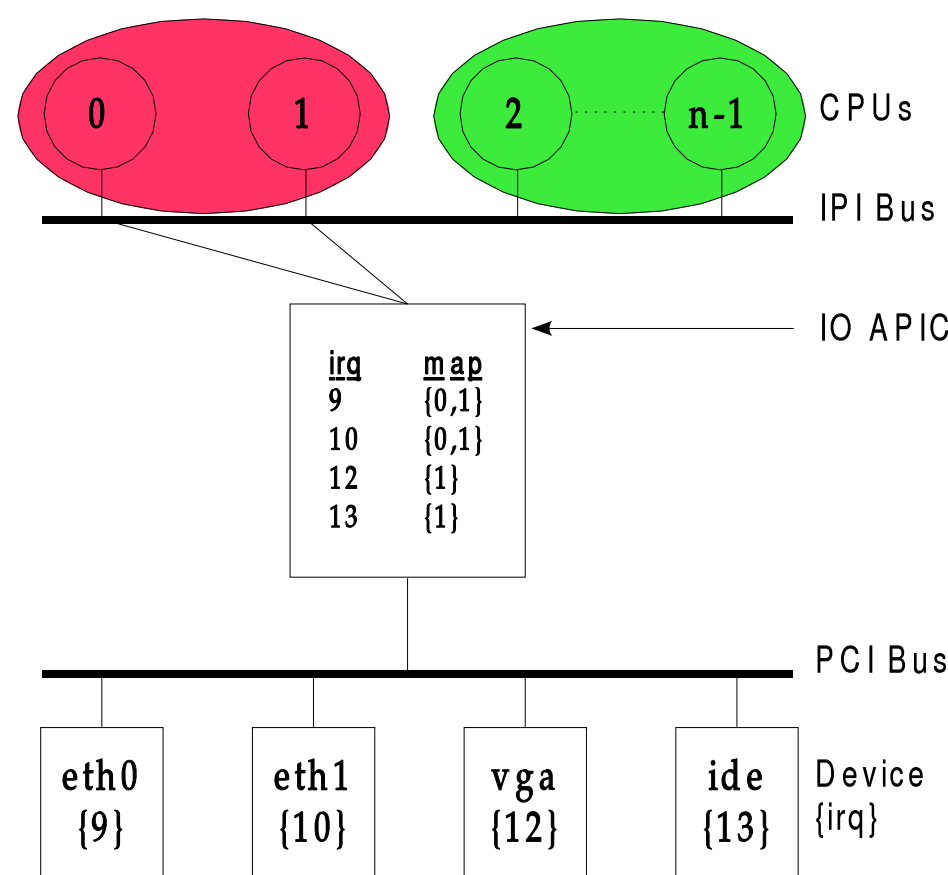
Interrupts

Interrupts are inherently asynchronous and may suspend task execution at any time and with unpredictable periodicity. Interrupt handling is done at the highest priority and therefore may not be preempted. Our goal is to eliminate the costs associated with interrupt handling on user tasks.

The diagram below illustrates the default configuration one would encounter on a x86 Linux/SMP system.



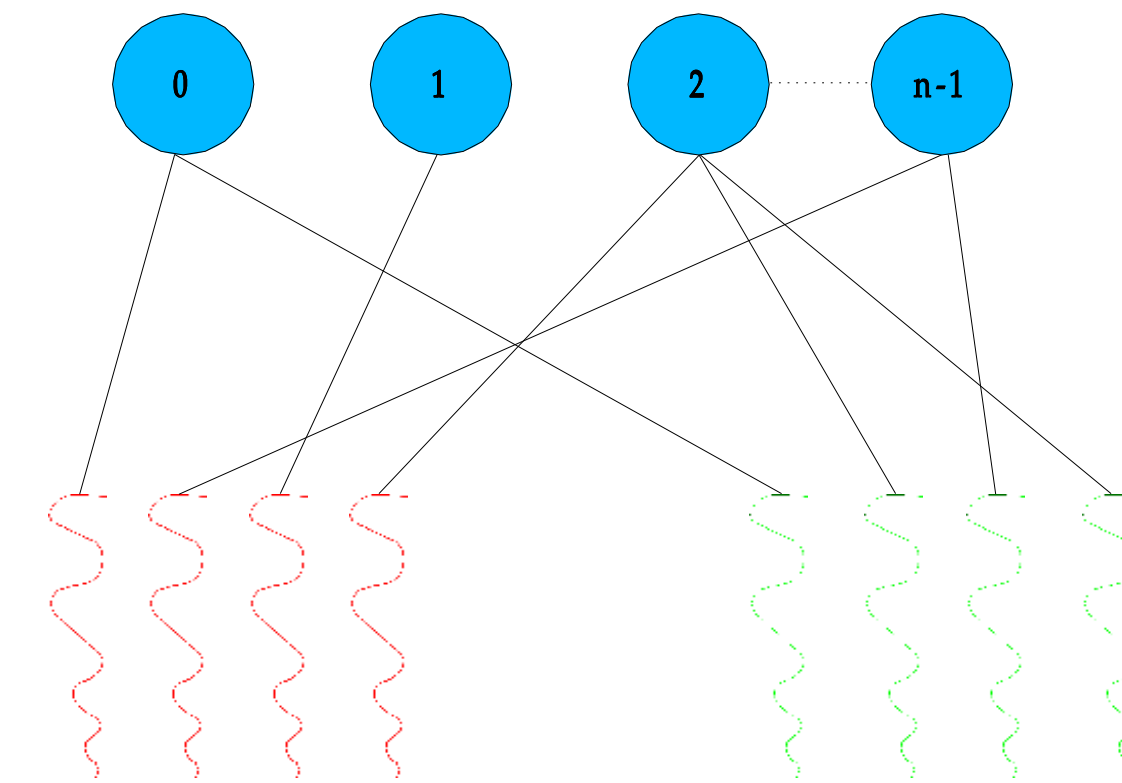
Restricting raising of interrupts to a subset of available processors eliminates the interrupt overhead on user tasks. The diagram below illustrates how this is accomplished.



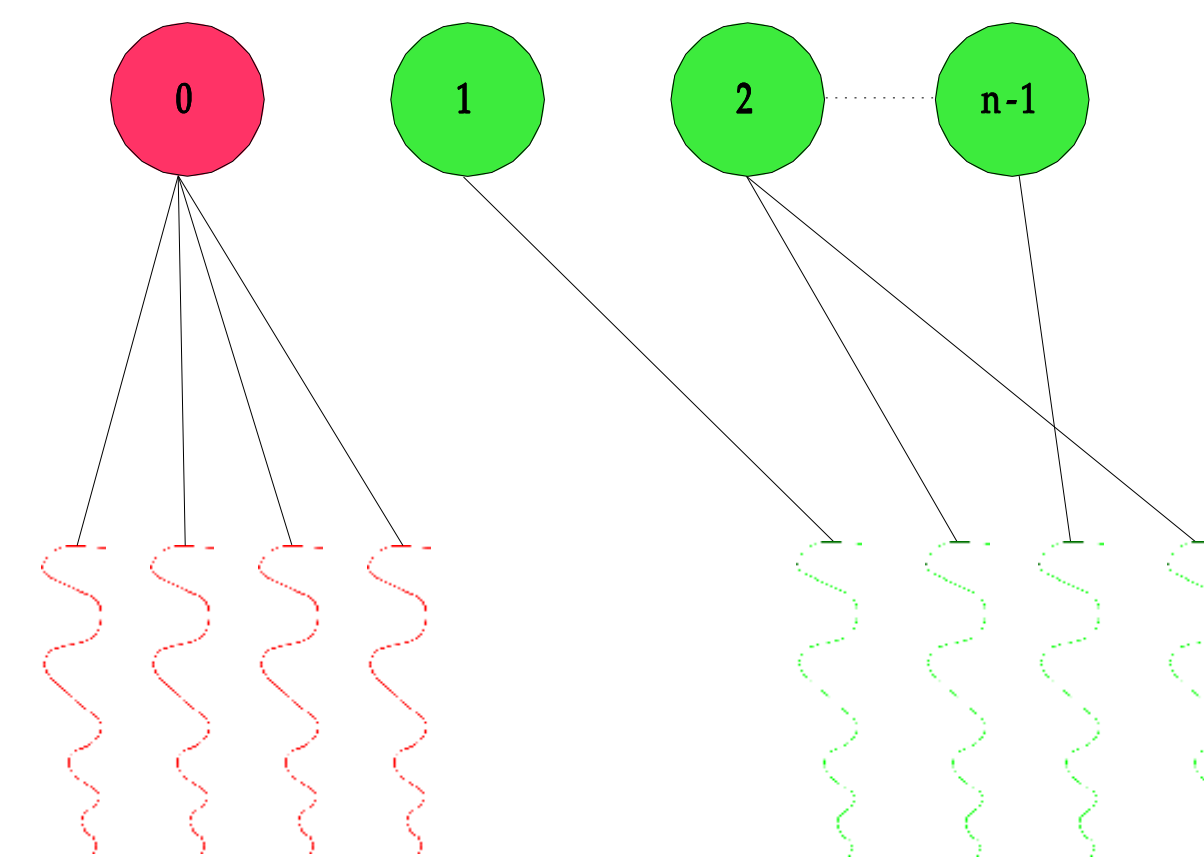
Processes

The arbitrary interleaving of kernel and user tasks may adversely affect the execution time of user tasks. We propose to partition the system in such a way that kernel tasks are only permitted to execute on a specified subset of processors. This shall eliminate the costs associated with execution of kernel tasks on user-level processes.

The diagram presented below illustrates the arbitrary interleaving of kernel and user tasks that one would encounter on a default Linux/SMP configuration.



Elimination of the arbitrary interleaving between user and kernel task execution is illustrated in the following diagram. This is accomplished by specifying a subset of processors for each kernel and user task execution.

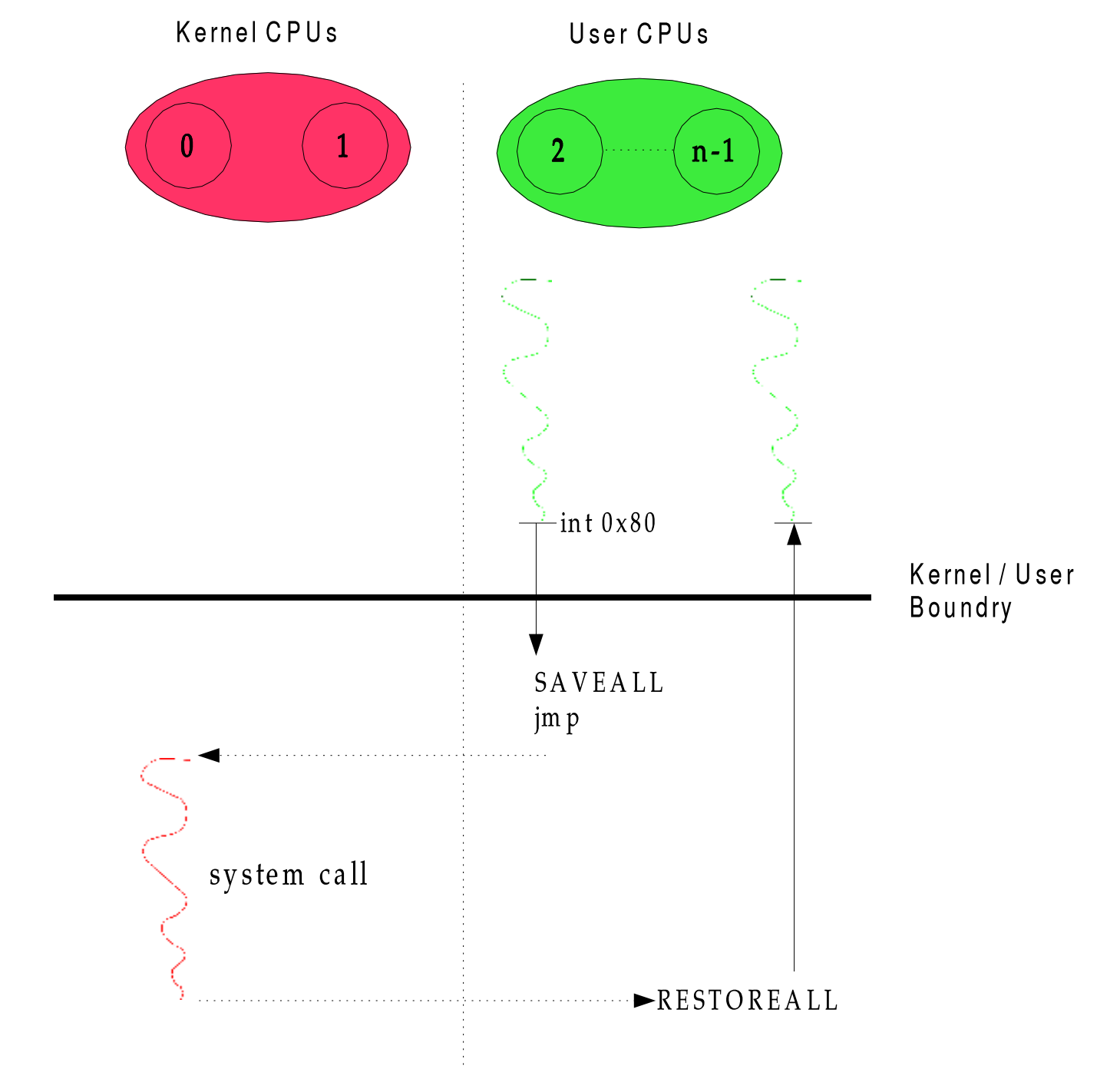


System Calls

Processes may request services from the kernel in the form of system calls. These requests require crossing of the user/kernel boundary and the execution of some kernel

code on behalf of the user-level task. We propose that such services should be rendered on processors dedicated to kernel task execution. This frees the processor for user tasks awaiting execution.

The diagram presented below illustrates how control is transferred between user and kernel processors during system call invocations.



Conclusion

We have proposed and illustrated the design of a tool that will allow for processor-level partitioning of kernel and user mode functionality for Linux/SMP systems.

We claim that SMP systems configured with this tool should provide for parallel algorithm execution that exhibits smoother runtimes and enhanced reproducibility of results.

As of now all the functionality described in this presentation has been implemented for the Linux 2.6 kernel. We are currently in the process of collecting and analyzing results that shall be presented in future works.