Kernel SMP - PSO

Liptak, Leandro - leandroliptak@gmail.com

July 14, 2011

Introducción

El objetivo del presente trabajo era extender el *kernel* desarrollado durante la cursada de la materia *Programación de Sistemas Operativos* (PSO) dotándolo de la capacidad para operar y sacar provecho en un entorno de multiprocesamiento simétrico (*Symmetric Multiprocessing*, SMP), más precisamente en un entorno de los provistos por procesadores *Intel* o compatibles.

Una de las principales características de la configuración y puesta en marcha de los diversos procesadores en estos esquemas antes mencionados es que resulta dependiente de las diversas arquitecturas. Es decir, cada arquitectura puede requerir implementaciones distintas para la inicialización de los procesadores o la comunicación entre los mismos. Las arquitecturas que requieren distinción son: *Intel 486DX2*, *Intel P6* e *Intel P4*. Dichas arquitecturas están ligadas a diferentes modelos de APIC (*Advanced Programable Interrupt Controller*), de aquí las consideraciones.

Lineamientos generales

Para convertir nuestro *kernel* en SMP (abusando de aquí en más de dicha expresión) lo primero y más fundamental es reunir la información necesaria acerca de los diversos procesadores en el sistema. Una vez que dichos procesadores se han identificado y se han creado las estructuras de datos pertinentes, es necesario inicializarlos para que se vuelvan operables, pues por defecto dichos procesadores (exceptuando el procesador de arranque, o *Bootstrap Processor*, BSP) se encuentran suspendidos. Finalmente, es necesario modificar los mecanismos de planificación de procesos (es decir, el *scheduler* del sistema) para que saque provecho de los diversos procesadores y los administre eficientemente como recurso. Sin embargo en el transcurso de lograr muchas de estas metas problemas inherentes surgirán como ser, por ejemplo, la sincronización en estos esquemas multiprocesador.

Implementación

Identificación del sistema

Parseo de la tabla MP basica y extendida, etc. Codigo del kernel de Linux. Estructuras de datos creadas.

Inicialización de los procesadores

Sequencia de arranque, INIT IPI, SIPI segun modelo 486DX2 o P4, APIC o xAPIC. *Warm reset*, *code vector*, *stacks*.

APIC

Identificación, esquema de uso y facilidades implementadas.

Spinlocks

Spinlocks, ventajas y porqué son necesarios.

Scheduling

Multiples colas, organizacion.

Balanceo de carga

Balanceo de carga en `fork()`, `run()` (?)

Conclusiones

Trabajo futuro