ADM: una Arquitectura Distribuida Multitarea.

Autor: Alejandro Daniel Martinez.

Nota: este trabajo forma parte de la tesis de graduación del autor que fue supervisada por el Ing. Guillermo A. Jaquenod.

"Nowdays, the multi-microprocessor architectures are used in computer fields like robotics and artificial intelligence."

1.- Consideraciones Generales:

Con el ràpido incremento observado en los ùltimos años de la potencia de còmputo y disponibilidad de memoria en circuitos integrados, y la reducciòn de costo y tamaño consecuente, la relaciòn precio/performance de los sistemas distribuidos que utilizan microprocesadores en configuraciones fuertemente acopladas se ha tornado interesante para aplicaciones donde se necesite gran capacidad de procesamiento, con posibilidad de incremento de la misma sin rediseñar todo el sistema.

Pueden considerarse diversos tipos de arquitecturas multi-microprocesador, que se diferencian basicamente en su estructura de interconexiòn(topologia, velocidad), organizaciòn de ejecuciòn (SIMD, pipeline, MIMD) y en su objetivo global (Special purpose, General purpose).

Una propiedad indispensable en el diseño de estos sistemas es la MODULARIDAD: La modularidad lògica hace posible la cooperaciòn eficiente entre tareas ejecutandose en diferentes procesadores, por lo que estas màquinas modulares pueden ser consideradas como una evoluciòn desde las arquitecturas single-processor/multiple-task hacia las multiple-processor/multiple-task. Ademas, este esquema permite configurar arquitecturas optimizadas para aplicaciones especificas, a bajo costo.

2.- Objetivos del proyecto:

El objetivo basico de este proyecto es producir un conjunto integrado de hardware bàsico y elementos de software que soporten eficientemente un ambiente de multiprocesamiento.

En la configuraciòn intervienen elementos estandar como plaquetas de CPU, memoria y I/O mas mòdulos especiales que implementan las estructuras de interconexiòn: un rol clave en la perfomance del sistema es asignado a la red de comunicaciòn, responsable del intercambio de informaciòn entre procesadores y su protocolo de acceso.

El sistema debe incorporar un software basico (KERNEL) multitarea, que soporte primitivas de comunicaciòn y sincronizacion entre tareas concurrentes.

Es deseable que la estructura general sea organizada de modo tal que el usuario solo vea un pool de tareas ejecutandose sobre un pool de procesadores: de esta manera el incremento en el nùmero de elementos de procesamiento será visto por el usuario

como un incremento en la velocidad virtual de procesamiento, donde las tareas deben ser asignadas con un criterio de localidad para su eficiente ejecución.

3.- Descripción del hardware:

En los sistemas en los cuales varios procesadores comparten alguna porción de su espacio de direccionamiento la estructura de conexión es crítica en el diseño e impone restricciones mas o menos fuertes a todos los mdulos. En este caso la propuesta es utilizar una estructura estandar High-Throughput tal como es la especificación VME (IEEE P1014/d1.0), compuesta por cuatro barras paralelas (data transfer bus -DTB-, priority interrupt bus, DTB arbitration bus, y utility bus), con protocolos de acceso al medio compartido que permiten gran flexibilidad en el diseño de los módulos constituyentes.

El objetivo principal es permitir la configuración de un sistema donde la performance esté principalmente limitada por los dispositivos empleados y no por el sistema de interfase.

La arquitectura considerada presenta N procesadores independientes, cada uno con su memoria local y periféricos locales, y un banco de memoria compartida con el hardware de arbitraje y asignación correspondiente.

Cada procesador del sistema tiene su propio reloj de tiempo real (TIMER), un conjunto privilegiado de instrucciones (estado supervisor), y un mecanismo de interrupciones, asi como una instrucción de "test and set" indivisible que permite la implementación de semáforos en un ambiente de multiprocesamiento.

4.- Descripción del Software:

El diseño e implementación de sistemas de tiempo real en arquitecturas con procesadores múltiples y memoria compartida puede encararse a traves de un conjunto de procesos secuenciales concurrentes que se comunican y sincronizan entre si con grandes ventajas de flexibilidad, tiempo de desarrollo, puesta a punto y verificabilidad. En estos procesos el efecto de compartir recursos se manifiesta como un retardo virtual despues de cada E/S y como una reducción de la velocidad efectiva del procesador por un factor determinado. Una propiedad interesante en el caso de procesadores múltiples es que la estructura de comunicación y sincronización entre los procesos no depende de la asignación de las mismas a determinado procesador, pudiendo esta última ser modificada en funcion de criterios de optimización de comportamiento (localidad de las comunicaciones y uso de los recursos) sin cambiar el software ya diseñado.

La concurrencia, comunicación y sincronización es soportada por un KERNEL que debe proveer, al menos las siguientes funciones:

a. mecanismos para la creación y eliminación de procesos.
b. administración de procesadores, memoria y dispositivos para los procesos.

c. herramientas para que los procesos puedan sincronizar sus acciones.
d. primitivas de comunicación entre procesos que sean independientes de la asignación de los procesos a los procesadores.

La propuesta de implementación es un Macro-KERNEL distribuido compuesto por un núcleo mínimo en cada procesador (KERNEL basico) que implemente la concurrencia de procesos asignados al mismo (y probablemente residentes en su memoria local) por time-slicing y provea servicios de comunicación por intercambio de m nsajes no solo entre los procesos locales sino tambien con otros residentes en distintos procesadores.

La idea de utilizar intercambio de mensajes y no semáforos o construcciones de mas alto nivel como monitores se basa en que se trata de un modelo facil de entender y utilizar, y que ademas permite su extensión a ambientes mas distribuidos o con menor grado de acoplamiento: por ejemplo, si uno de los procesadores del "cluster" puede acceder a una red local, el intercambio de mensajes puede ser extendido al ambito de esta sin modificaciones para los procesos usuarios.

Es importante considerar que, dado que el manejo de memoria es realizado por el KERNEL mediante los servicios de pedido y liberacion de buffers, los espacios de direccionamiento de los procesos deben ser disjuntos, a fin de evitar efectos no deseados e interferencias.

Un KERNEL basico consta fundamentalmente de tres modulos:

1.- Interrupt-Handler (IH): se encarga del tratamiento de las interrupciones.

2.- Dispatcher: se encarga de decidir a qué proceso de los que estan listos para ejecución les asigna el procesador. Existe gran variedad de algoritmos de despacho, en este caso se propone un esquema de time-slicing con prioridades fijas.

3.- Syscall: modulo de provisión de servicios de creación y eliminación de procesos, y comunicación y sincronización entre ellos.

Es interesante destacar como el problema de sincronización es resuelto a traves de diferentes herramientas en las distintas capas del sistema:

- a nivel de hardware debe arbitrarse el uso del bus compartido mediante protocolos microprogramados.
- a nivel del KERNEL la exclusión mutua se obtiene:

> Para el empleo de la memoria local, gracias a la caracteristica ininterrumpible del nivel supervisor en el que se ejecuta dicho KERNEL.
> Para la memoria global, mediante semáforos implementados por software.

- a nivel de los procesos no existen recursos compartidos explicitamente (los espacios de direccionamiento de los procesos son disjuntos) y la sincronización se realiza a traves de intercambio de mensajes.