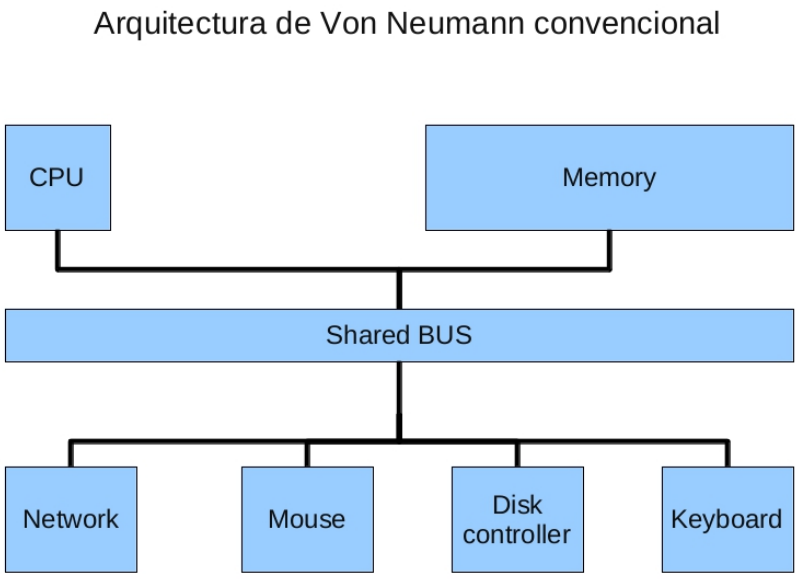
Que és y para que sirve el NUMA

Autor: Marcelo Soares
http://communities.vmware.com/people/marcelo.soares/blog
http://vmsoares.wordpress.com

Introducción

La arquitectura de computadoras mas conocida es la que llamamos de "Von Neumann", donde todos los componentes de IO, procesadores y memória comparten un mismo BUS, o sea, toda la información que necesita ser tratada por el procesador siempre tiene que pasar por un único medio de comunicación. El imagen abajo muestra un ejemplo bastante simple y directo de esta arquitectura:

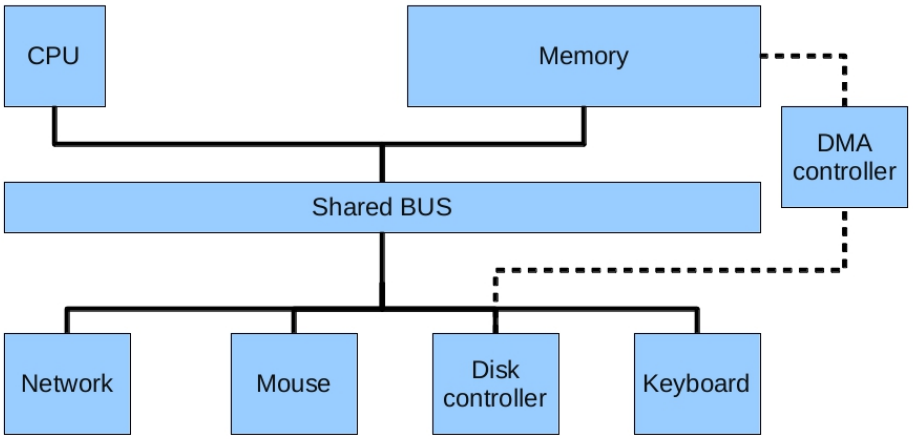


Actualmente, muchas alternativas son implementadas para evitar el sharing de este recurso, pero al final casi siempre es necesário utilizar un BUS compartido para comunicación.

En ese contexto, las nuevas computadoras con muchos procesadores (no olvidamos que ya es comun ver servidores con 32, 64 y 128 cores, principalmente ejecutando algun hypervisor de virtualización) tienen un problema de necesitar compartir una sola via para acceso a la memória, lo que puede generar latencias de tiempo para que una CPU específica pueda acceder al área de memória necesário.

Un ejemplo de mejora en este architectura es el DMA (Direct Memory Access) para dispositivos de IO específicos, que reciben un acceso directo a la RAM, como en el comparativo abajo:

Ejemplo: Optimización DMA para disco



El DMA permite que un disco o placa de red (u otro dispositivo) acceda la memória directamente si tener que pasar por la CPU ni por el BUS compartido entre todos. Es un canal exclusivo, lo cual es más rápido y ahorra tiempos de CPU. De hecho, todas las computadoras de la actualidad trabajan con DMA para disco,y si eso llega a desabilitarse, se puede percibir una gran lentitud en el sistema.

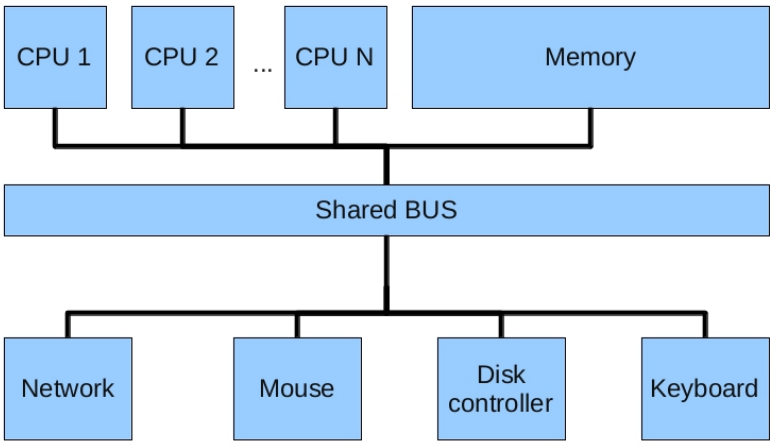
UMA

UMA es el estilo comun en lo cual vários procesadores acceden a la memória de un equipo, y quiere decir "Uniform Memory Access", o Acceso a la Memória por un medio comun (en este caso, el BUS compartido).

Con el crescimiento de la cantidad de procesadores que un único servidor puede tener, el BUS puede no ser suficiente para proveer la mejor velocidad de acceso de los procesadores a la memória, quedando sobrecargado y generando latencias.

La figura abajo muestra un resumen de como es la arquitectura SMP (con multiples procesadores):

UMA – Multiples procesadores



En esta architectura, cuanto más CPUs y dispositivos estna en el sistema, mas lento y costoso va a ser la comunicación entre ellos. Este es un buen ejemplo de que no basta agregar RAM y CPU para que un sistema se vuelva más rápido – de hecho, esto puede mas daños que beneficios.

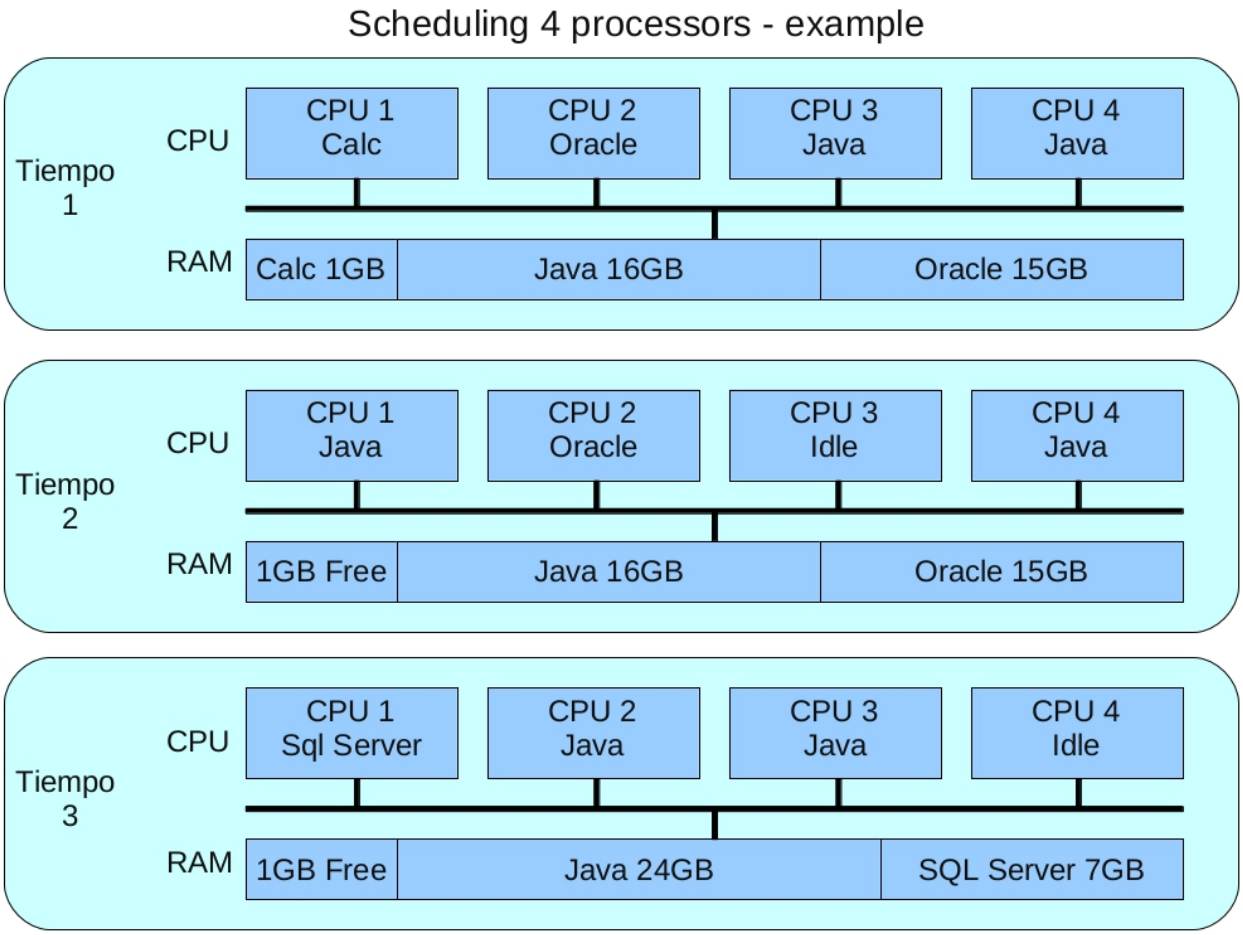
NUMA

NUMA quiere decir “Non-Uniform Memory Access”, y se puede traducir como Acceso a Memória de modo No Uniforme. En entornos con multiples procesadores, puede ser visualizado como una modularización y división de la memória y los procesadores en "conjuntos".

La ventaja de agregar muchos procesadores y mucha memória en una sola computadora es el costo – se ahorra en motherboard, recursos de alimentación, enfriamento, almacenamiento interno, chassis, etc. Actualmente, hay servidores que soportan un gran cantidad de RAM (terabytes) y tienen 2, 4, 8 y más sockets para colocación de procesadores, que actualmente son, como mínimo, quadcore.

Los sistemas operativos tienden a aprovechar de la mejor manera posible un procesador, de modo que varios procesos que se ejecutan son asignados a procesadores distintos para que un buen balanceo de carga entre ellos ocurra, y ninguno se quede sobrecargado. A esta operación llamamos de *scheduling* (o agendamiento).

Abajo muestro un ejemplo de scheduling de 4 procesadores y la utilización de memória:



En este ejemplo, existen 3 puntos de tiempo (no me voy a detener en saber si son segundos, milisegundos, o lo que sea). En el punto de tiempo 1, existen 4 procesos ejecutando (Calculadora, Oracle y 2 procesos de Java, muy probablemente en hyperthreading). La memória tiene 32GB distribuídos entre los procesos.

El punto de tiempo 2 no tiene el Calc, y si un procesador libre. Fijense de que los procesos cambian de procesador, lo que es normal durante su vida en un OS. Como un proceso no está mas activo (el Calc) se libró 1Gb de ram que ocupaba, y un CPU está idle.

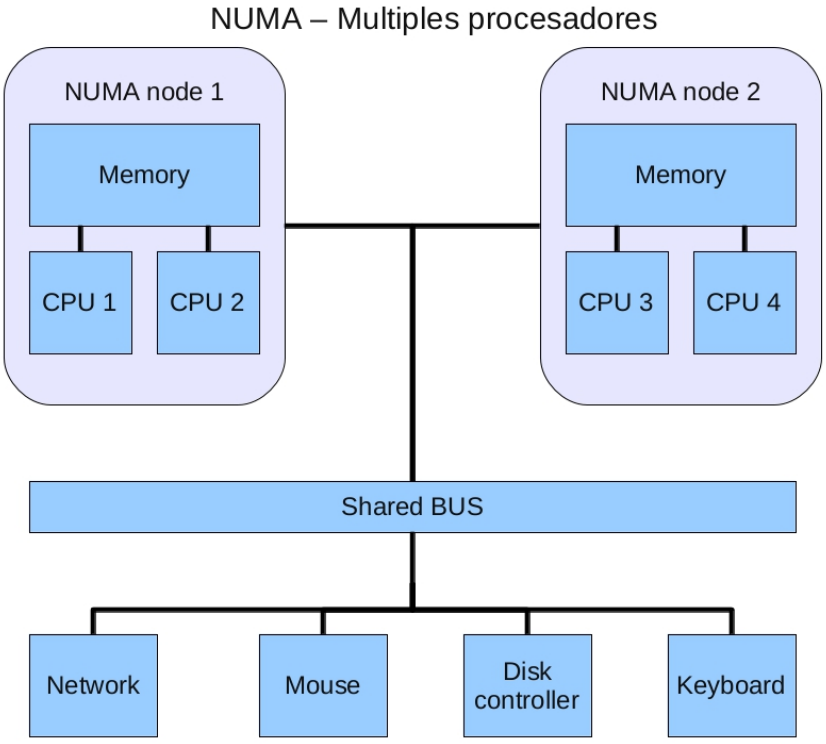
En el punto de tiempo 3 se puede ver que el Oracle no está mas en ejecución, y si ahora un Sql Server. El Java sigue su funcionamiento.

Estes 3 ejemplos están funcionando sobre una arquitectura UMA, o sea, todos los procesadores pueden acceder toda la memória, pero solo hay 1 ruta. Si bien puee

ejecutar cosas en paralelo, nunca van a poder leer o escribir en la memória todos juntos, y esa es la desventaja del UMA. Por menor que sea el tiempo, una fracción de segundos perdida por la CPU esperando que se desocupe el BUS para comunicarse con la RAM genera una latencia perceptible para las aplicaciones.

El NUMA agrega otra perspectiva acerca del acceso de memória. Cada grupo de procesadores físicos tienen acceso a solamente una parte de la memória física, por lo cual se distribuye en vários canales el acceso a RAM. La desventaja es que para cada procesador hay menos memória disponible para acceso directo, pero la velocidad de acceso a RAM es más alta.

En NUMA en hardware crea nodos, o sea, conjuntos de procesadores con sus memórias privadas. Así la disputa por el BUS diminuye, posibilitando un acceso mucho más rápido a la RAM:



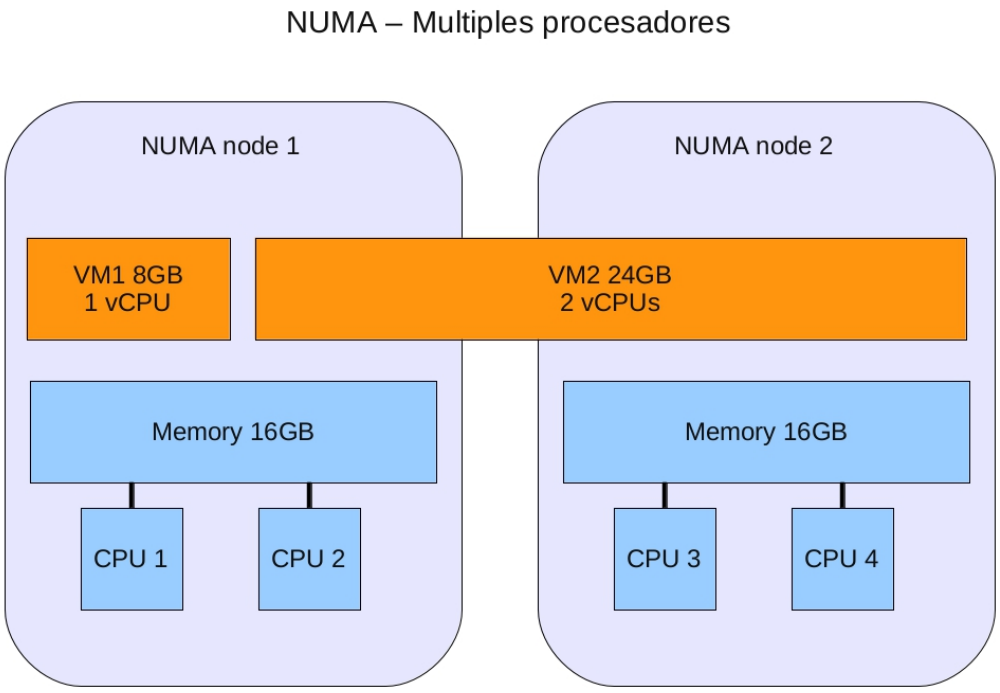
El único problema del NUMA es en el caso de que un proceso específico necesite más RAM do que existe en el nodo donde se está ejecutando. En condiciones normales, es muy raro que un único proceso necesite más que eso, pero puede pasar. En este caso, la memória de otro nodo deberña ser utilizada, lo que generará una lentitud, ya que la comunicación deberá pasar por el BUS compartido y también por el BUS interno del otro nodo utilizado.

Como dicho anteriormente, un proceso puede nunca llegar a necesitar de más RAM que un nodo, pero la virtualización cambió esto definitivamente.

NUMA y la Virtualización

Los hypervisors ya hace bastante tiempo tienen el conocimiento del NUMA y laburan para que, en el caso de que esté activado en la BIOS, cada VM este contenida en un solo nodo, migrando las mismas en caso de necesidad de un nodo a otro. El VMware ESX 2.5 ya tenía esta capacidad, y hasta hoy lo hace automaticamente, sin que el administrador tenga que especificar nada manualmente.

Con las nuevas versiones de hypervisors, actualmente es comun tener lo que llamamos de "VM Monstruos" (así llamadas en el VMworld 2011 y representadas fisicamente como un gracioso muñeco caminante en los pasillos), con mucha vRAM y muchos vCPUs. Eso puede hacer con que VMs tengan tamaños grandes lo suficiente para generar lentitud en sistemas UMA, o en NUMA, generar el hecho de que una única VM pueda estar compartida entre 2 o más nodos de CPU/RAM, com el ejemplo abajo:



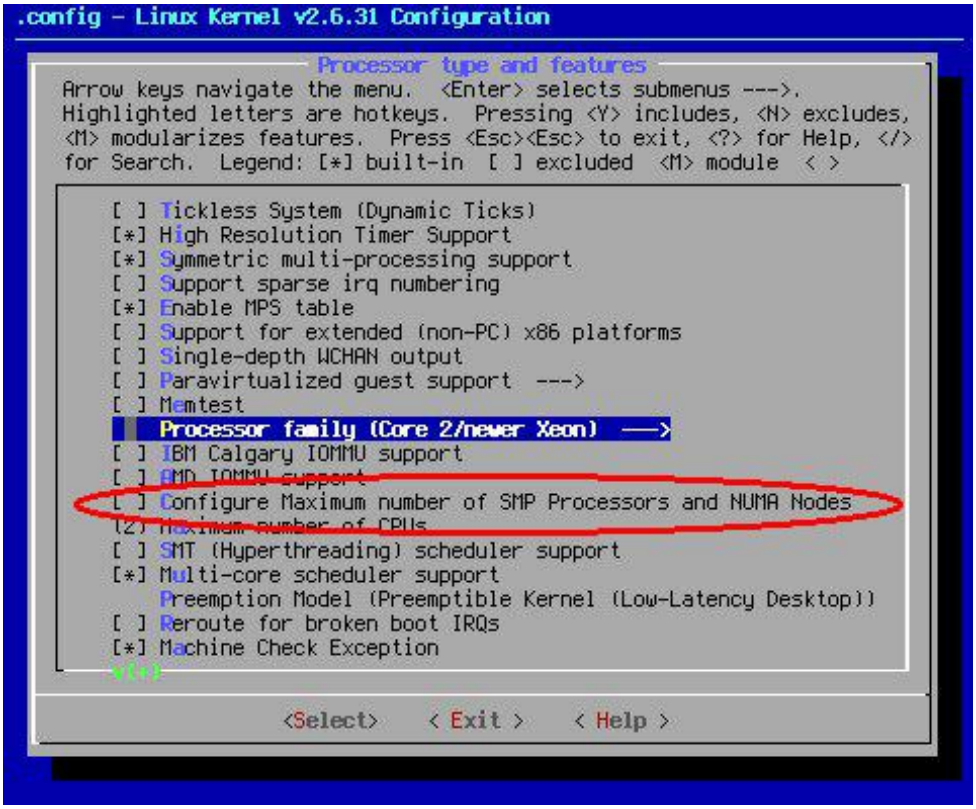
La VM1 está contenida en un nodo y no tiene mas vCPUs ni vRAM que el nodo 1. La VM2 sí tiene más memória que un nodo, por lo cual una parte de su memória tendrá una grande latencia de acceso si es necesário por un procesador de un nodo a la cual no pertenece.

Usamos un ejemplo utilizando el imagen anterior: la VM2 es puesta a funcionar en

el CPU 3 y 4 (tiene 2 vCPUs). Si el procesador 3, a pedido del Guest OS de la VM, necesita acceder a una página de memória que está en el Nodo 1, eso va tardar mucho, generando lentitud para la VM. Igualmente, si es puesta a funcionar en los procesadores 1 y 4, puede que la lentitud sea más grande, ya que el OS puede pedir al procesador 1 lo que esté en el nodo 2, y al 4 lo que esté en el nodo 1. Eso es lo peor que puede pasar para la latencia de memória.

Para solucionar eso, se debe elegir entre no utilizar NUMA en el hardware físico (quedandose así una arquitectura UMA, pero al costo de una lentitud promedio en todas las VMs) o habilitar el **vNUMA** en las VMs que tengan mucha vRAM y vCPUs. De este modo, la información de que procesador tiene acceso a que áreas de memória es repasado al Guest OS, lo cual va trabajar como se estuviera en el hardware físico, evitando al máximo hacer con que los procesos que estén en el CPU1 pidan memória del nodo 2, por ejemplo, en nuestra figura.

Los sistemas operativos que pueden tener este tipo de funcionalidad son los Windows desde la versión 2000 (consultar la documentación para cada versión), y también el Linux/UNIX. En el Linux, por ejemplo, está soportado desde el kernel 2.4 para máquinas de 64 bits, habilitando la opción mostrada abajo al compilar el kernel:



En Windows, la opción ya viene preconfigurada, y el kernel ya lo detecta en la BIOS. Este tipo de sistema operativo llamamos de NUMA-aware.

vNUMA

La opción del vNUMA solo está disponible en el VMware vSphere 5, versión en la cual fue introducido el soporte a VMs con 32 vCPUs y 1TB de vRAM, siendo casi imprescindible dicha funcionalidad.

VMware provee mucha información acerca del vNUMA. Consultá la documentación de Resource Management: http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-resource-management-guide.pdf y también las mejores prácticas: http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf.

Un ejemplo práctico de como se puede configurar aplicaciones específicas para utilizar NUMA:

http://msdn.microsoft.com/en-us/library/ms345357(v=sql.105).aspx

Marcelo Soares
vExpert 2012 – VCP 310/410/510