UNIVERSIDAD NACIONAL DE INGENIERIA
Facultad de Ingeniería Industrial y de Sistemas

ARQUITECTURA NUMA LA TENDENCIA DE LA INDUSTRIA DEL HARDWARE Y UNA EXPERIENCIA DE LA EMPRESA DATA GENERAL

INFORME DE INGENIERÍA
PARA OBTAR EL TITULO PROFESIONAL DE
INGENIERO DE SISTEMAS

PRESENTADO POR :

CARLOS D. DURAND CHAHUD

Lima, 1998

Al invalorable apoyo de mis Padres y enseñanzas de mis maestros.

I N D I C E

RESUMEN EJECUTIVO

La "comoditización" de los servidores empresariales estará dirigida a las soluciones de Procesamiento de Bases de Datos, Sistemas de Soporte Decisional - DSS, Data Warehousing, aplicaciones OLTP (On-Line Transaction Processing), entre otras. Las cuales no sólo demandarán características especiales de performance, sino también la habilidad de poder escalar eficientemente hacia mayores aplicaciones e incrementos significativos en el número de usuarios.

Como consecuencia están apareciendo sistemas basados en tecnología de Multiprocesamiento Simétrico (SMP) con diseños de tarjetas de varios procesadores, su propia memoria RAM, memoria Cache de primer y segundo nivel y bus de I/O, utilizando una arquitectura de multiprocesamiento conocida como NUMA (Non Uniform Memory Access).

En ese sentido hoy en día existen tendencias que se notan claramente en la oferta tecnológica:

1. Mantener vigente la arquitectura SMP-UMA
2. Extender la vida de los Mainframes
3. Migrar hacia la arquitectura MPP
4. Evolucionar hacia la arquitectura NUMA

Por otro lado, el sistema operativo es esencial para ayudar a la arquitectura ccNUMA, para mantener eficiencia en configuraciones donde existan muchos procesadores involucrados, así como aislar a las aplicaciones de las complejidades de hardware que esta arquitectura involucra.

El sistema operativo debe tener: Soporte de multiprocesador, suficiente espacio de dirección para la identificación de los CPU´s y debe tener capacidad de trabajo multinodal para administrar la memoria virtual.

Así como las computadoras crecen y son cada vez más poderosas, para poder soportar aplicaciones con bases de datos cada vez más grandes, por lo tanto la necesidad de incrementar la memoria física, también llamada RAM o primaria, crece de modo proporcional.

Finalmente lo realmente importante en el juego tecnológico, desde el punto de vista del usuario, es como se puede mejorar la relación Costo/Rendimiento, de modo tal que el valor agregado de la tecnología sea significativo para su negocio.

Descriptores temáticos : Multiprocesamiento Simétrico, Non Uniform Memory Access, NUMA, Arquitectura de Computadores.

INTRODUCCION

En los próximos años, la "comoditizacion" de los servidores empresariales se incrementará orientada a las soluciones de misión critica: Procesamiento de Bases de Datos, Sistemas de Soporte Decisional, Data Warehousing, aplicaciones OLTP, entre otras. Estas no solo demandaran características especiales de performance, sino que la habilidad de poder escalar eficientemente hacia mayores aplicaciones e incrementos significativos en el numero de usuarios.

Respondiendo a esta necesidad están apareciendo, y lo harán cada vez mas, sistemas basados en tecnología de multiprocesamiento Simétrico (SMP) con diseños de tarjetas de varios procesadores, su propia memoria RAM, memoria Cache de primer y segundo nivel y bus de I/O, utilizando una arquitectura de multiprocesamiento conocida como NUMA (Non Uniform Memory Access).

Estas tendencias tecnológicas reflejadas en el mundo de los negocios, han obligado a los fabricantes ha enfrentar y sobrevivir con el cambio por siglos. Analizando los ciclos de cambios se ha podido observar que estos han sido casi constantes, las mejoras se dan en porcentajes anuales : 20% de mejoras en productividad y 20% en reducción de costos; por ejemplo, durante años las escuelas de negocios han venido desarrollando teorías muy complejas que puedan explicar estos ambientes y comportamientos económicos, pero al final sigue siendo algo que se entiende intuitivamente.

A pesar de esto la industria de los computadores a mantenido un comportamiento diferente, ya que esta influenciada por la tecnología de los semiconductores, y esta cambia de modo exponencial. Ya desde 1964, Gordon Moore, graficó lo que significaba el desarrollo tecnológico de los transistores, prediciendo de esta manera que la tendencia del cambio era exponencial y que cada dos años se duplicaría el rendimiento de los procesadores. Esta declaración dio lugar a lo que se conoce como la Ley de Moore que ha estado vigente durante las ultimas tres décadas y ha sido el signo distintivo para la industria de computadores.

Historia distinta sucede con el componente denominado Software, que a diferencia de la evolución exponencial que observamos en el lado del hardware, este ha evolucionado incrementalmente debido en gran parte a que es muy humano dependiente. Por ejemplo se puede diseñar un microprocesador de 5 millones de transistores con un equipo de gente relativamente pequeño, apoyado con herramientas sofisticadas de modo tal que se pruebe y funcione a la primera; en cambio, desarrollar un sistema operativo de unos 15 millones de líneas de código es mucho mas complejo y requerirá mayor mano de obra y tomara mucho mas tiempo hasta obtener una versión aceptable.

Poniendo esto en perspectiva y en el contexto de ciclo de vida de los productos, podemos entender del porque los ciclos de vida de los componentes de hardware son mas cortos en contraste con los de software que en muchos casos toman años.

Este informe pretende ratificar algunas de estas teorías y tratara de explicar como desde el lado de la oferta, vale decir los fabricantes de tecnología de hardware, se enfrenta a la evolución y competencia por el mercado de tecnología informática.

1. ENTORNO DE COMPETENCIA

Siendo este un mercado típico de competencia tecnológica, donde muchas veces la sobrevivencia de las empresas esta en juego, ya que la obsolescencia y preferencia del mercado no perdona a nadie, se hace imprescindible el adoptar posiciones estratégicas de mediano y largo plazo. En ese sentido hoy en día existen cuatro tendencias que se notan claramente en la oferta tecnológica para enfrentar los requerimientos de las soluciones denominadas de "misión critica":

1. Mantener vigente la arquitectura SMP- UMA
2. Extender la vida de los Mainframes
3. Migrar hacia la arquitectura MPP
4. Evolucionar hacia la arquitectura NUMA

Cada una de estas tendencias alternativas, no necesariamente mutuamente excluyentes, conllevan ciertas ventajas y desventajas. Revisemos las mas importantes que puedan aclararnos el entorno.

La arquitectura SMP-UMA conlleva la ventaja de aprovechar todas las aplicaciones y herramientas de software estándares vigentes en el mercado pero tiene como desventaja sus problemas de escalabilidad, en numero de procesadores limitando de esta manera su potencial crecimiento en rendimiento.

Los denominados "Mainframes" tienen la ventaja de su arquitectura robusta y capaz de soportar requerimientos muy grandes de rendimiento, pero su desventaja se hace notoria por el lado del software y las herramientas asociadas a él, que en todos los casos son propietarios y muy limitantes, comparadas a las opciones que existen actualmente para el mercado de sistemas abiertos.
La tercera alternativa la cual es la arquitectura MPP(Procesamiento Masivo Paralelo) no tiene tampoco una amplia gama de productos de software y aplicaciones que permitan aprovechar el rendimiento de esta arquitectura, obligando, en muchos casos, a tener que reescribir los programas y aplicaciones.

Finalmente la arquitectura NUMA trae consigo la ventaja de alta escalabilidad y aprovechamiento de las aplicaciones y herramientas de software disponibles en el mercado hoy en día, permitiendo de este modo "evolucionar" hacia plataformas más potentes protegiendo la inversión que se tenga en aplicaciones y herramientas de desarrollo. Teniendo todavía como desventaja su poca penetración en el mercado informático dado su reciente anuncio y desarrollo.

2. EXPERIENCIA DATA GENERAL

La motivación principal para desarrollar este informe se basa en la experiencia vivida como funcionario de DATA GENERAL Corp. Y el haber palpado el proceso de investigación, desarrollo y manufactura de los servidores basados en la arquitectura NUMA [23].

Quisiera destacar cuales son, a mi entender, los elementos claves que permitieron este desarrollo tecnológico:

- **La tradición de Data General como proveedor de sistemas medianos**. Desde la fundación de la Corporación en el año 1968, el enfoque fue hacia los sistemas medianos basados en arquitecturas innovadoras y de tecnologías de avanzada, estrategia que permitió crecer a la compañía y convertirse en uno de los proveedores tecnológicos mas importantes. Dentro de este contexto también afectado por los cambios tecnológicos y de demanda tuvo que hacer su propio proceso de reingenieria, de modo tal, que su oferta de productos y servicios cambiara radicalmente adecuándose de este modo a las tendencias y estándares tecnológicos del boom de los llamados Sistemas Abiertos así como del mercado.

- **La experiencia de muchos años de éxito proveyendo servidores UNIX en arquitectura de multiprocesamiento simétrico.**

Data General se ha caracterizado desde inicios de la década de los 90's por ser un proveedor de arquitectura abierta y tener como bandera al recientemente lanzado, a la arena comercial, sistema operativo UNIX; quien desde sus inicios marcó la estrategia de productos de la compañía.

- **Niveles de escalabilidad obtenidos en el numero de procesadores por servidor, llegando hasta 16 CPUs.**

  Siempre fue preocupación de Data General la creciente demanda de parte de los usuarios por una mayor capacidad de crecimiento y protección a la inversión en tecnología, siendo por esta razón una de sus características fundamentales la capacidad de escalamiento de los sistemas servidores que forman parte de su oferta de productos. La experiencia de haber desarrollado comercialmente el primer servidor basado en una arquitectura de tarjetas cuadraticas, llegando a escalar hasta 32 procesadores. Producto que puede considerarse como la aproximación más cercana a la arquitectura NUMA.

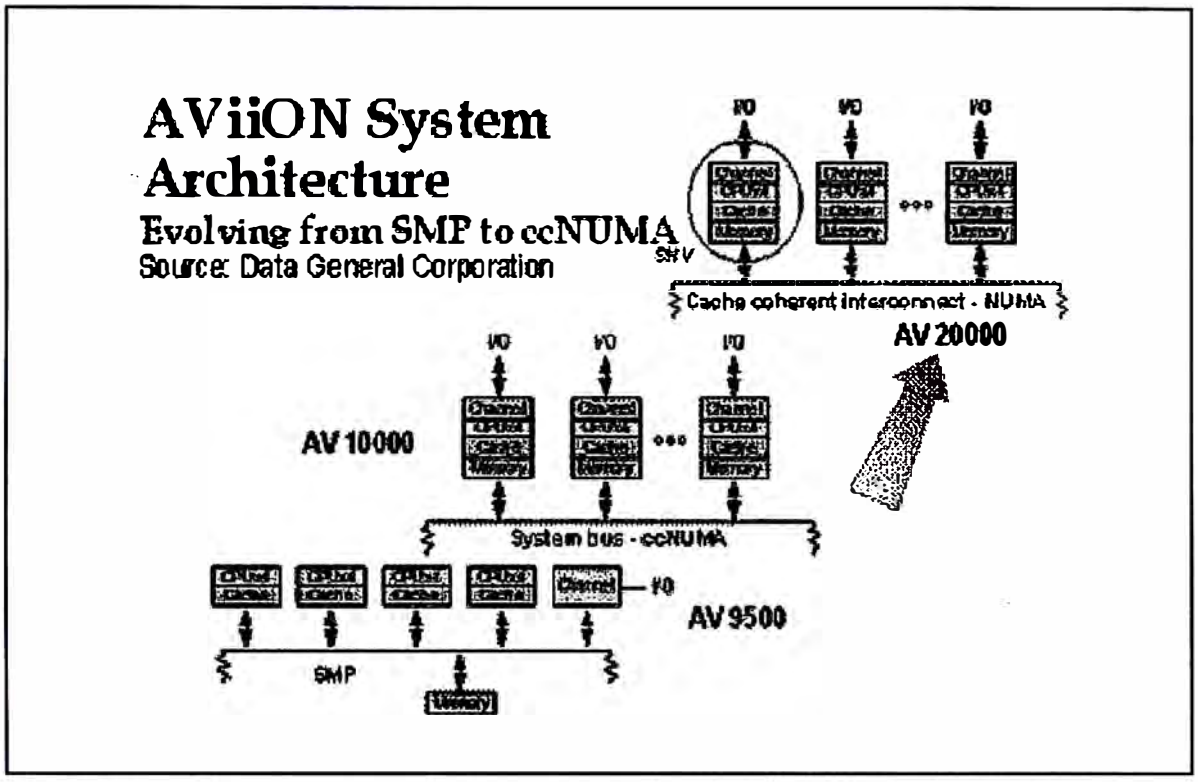
3 SITUACION ACTUAL Y ALTERNATIVAS A LA ARQUITECTURA SMP

Tal como ha venido evolucionando la arquitectura SMP, hoy en día la mayoría de sistemas multiprocesadores, continúan siendo muy adecuados para un amplio rango de aplicaciones. Pero eventualmente, esta arquitectura se está rindiendo debido a que la performance requerida por algunas aplicaciones no puede ser incrementada por simple añadidura de procesadores. Esto sucede cuando las configuraciones se encuentran entre los ocho (8) y dieciséis (16) procesadores, dependiendo en cada caso, de factores que van desde la calidad de la implementación SMP (CPU, Buses, el sistema operativo, etc.) hasta la aplicación misma.

Cuando se escala a grandes configuraciones SMP aparecen problemas tan simples como: Costo y Oportunidad de Mercado. Esto significa en pocas palabras, a mayor número de procesadores y velocidad de los mismos, mayor ancho de banda para su interconexión será necesaria; Por lo tanto, el proveer estos anchos de banda y sus interconexiones de baja latencia son muy costosos en el desarrollo y la implementación final de esta arquitectura. Finalmente esto no soluciona el problema medular, el cual es, que seguimos recargando un recurso compartido, el cual a la larga siempre será el cuello de botella.

Entre las opciones que algunos fabricantes están promoviendo se encuentra el CLUSTERING (Low Parallel Processing) y el MPP (Massive Paralleling Processing). Ambos tienen sus propias ventajas, pero lo común a ellos es una desventaja al problema central: Ambos, CLUSTERING y MPP requieren diseño y programación especiales, ya que rompen con el esquema de

"Aplicación Monolítica". Esto en la práctica significa que una aplicación que se ejecuta hoy en un ambiente SMP, requerirá rediseño y recodificación para poder ser migrado a alguna de las alternativas mencionadas [24].



4 NUMA : LA SIGUIENTE GENERACIÓN DE SMP

Existe una tercera alternativa basada en la arquitectura NUMA SMP, la cual permite romper la barrera en cuanto al número de procesadores por sistema, configurando de esta manera un nuevo modelo de escalabilidad de hasta varias decenas de procesadores.

¿Pero que es realmente NUMA?, ¿Cómo funciona?, ¿Qué ventajas tiene?, Son algunas de las preguntas que éste informe tratará de responder.

Cuando se diseña un sistema de multiprocesamiento, en lo concerniente a la interacción entre los procesadores y la memoria global compartida, existen sólo dos opciones:

- UMA (Uniform Memory Access) - Permite que todas las direcciones de memoria estén accesibles por igual a todos los procesadores.

- NUMA (Non Uniform Memory Access) - Permite que ciertas direcciones de memoria estén más accesibles a unos procesadores que a otros.

De esto podemos concluir que la mayoría de los sistemas SMP vigentes utilizan la arquitectura UMA (Ver gráfico Nº 1) y que el concepto del que trata este informe: NUMA, es una evolución de esta.

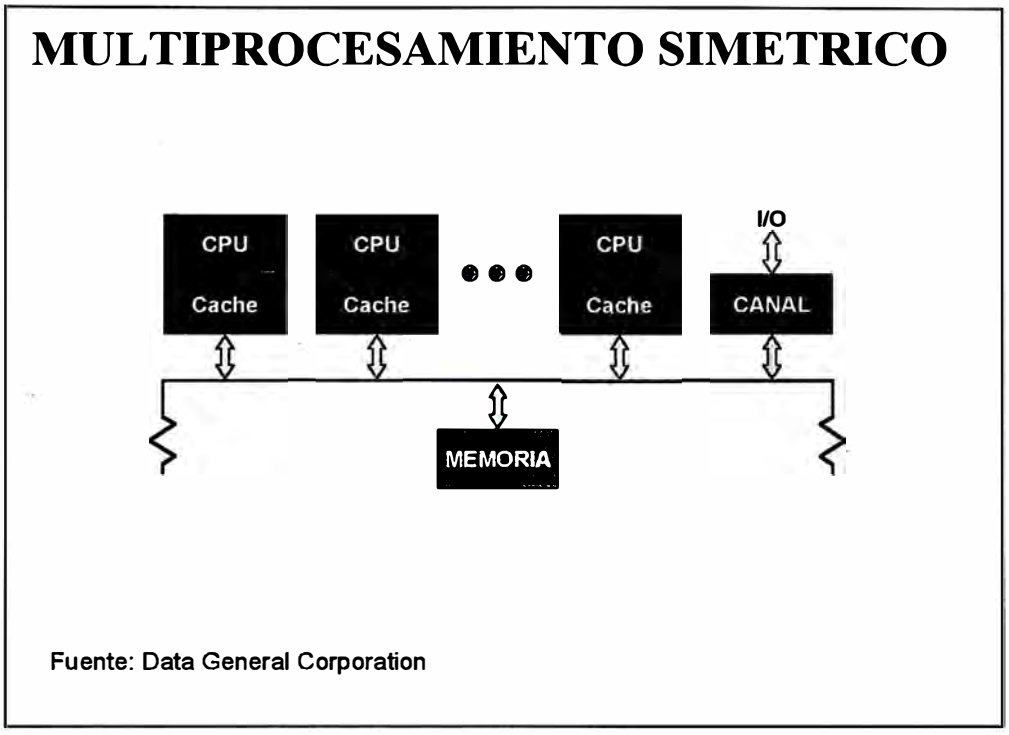


GRAFICO Nº 1. ARQUITECTURA SMP - UMA

En la arquitectura NUMA, al igual que en un sistema tradicional SMP, todos los procesadores comparten una memoria global, pero la diferencia está en que el tiempo de acceso a las direcciones de esta memoria dependerá de su ubicación así como cuál procesador está involucrado.

De acuerdo a lo que se muestra en el Gráfico Nº 2, cada "motherboard" o nodo contiene:

- Hasta cuatro (4) CPU's con su propia memoria Caché
- Memoria RAM
- Memoria Caché de segundo nivel (L2)
- Canales de I/O

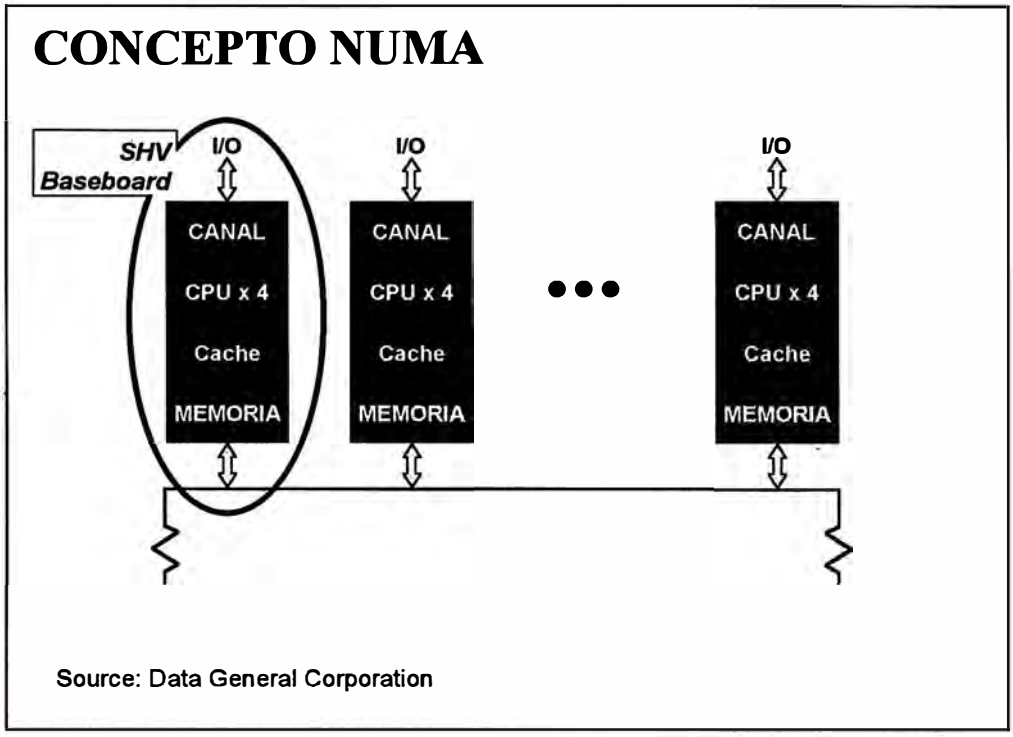


GRAFICO Nº 2 ARQUITECTURA SMP NUMA

En esta arquitectura, la memoria Caché (L2) tiene la misma velocidad del procesador por lo tanto es accesible con cada ciclo del procesador. Cuando la Data no se encuentra en la memoria Caché es leída desde la memoria RAM y copiada a memoria Caché (L2); si la data no se encuentra tampoco en la memoria RAM del nodo, será buscada en la memoria RAM de otro nodo vía un anillo de interconexión de alta velocidad, que a su vez tiene una memoria Caché de tercer nivel (L3) suficientemente grande como para mantener la coherencia a través de los nodos interconectados.

Aquí podemos observar un concepto novedoso, el de dos clases de memoria:

- La memoria cercana y
- La memoria lejana.

La primera es la memoria RAM asociada al nodo que requiere la data y la segunda es la asociada a otro nodo en el sistema.

5. ARQUITECTURA NUMA

En párrafos anteriores había definido a ccNUMA como una evolución de la arquitectura SMP con una implementación distribuida de memoria compartida, permitiendo que la interacción entre el procesador y la memoria sean transparentes al software de aplicación. Esto utilizando técnicas como la de memorias jerárquicas y administración inteligente de recursos para optimizar la carga de trabajo de los procesadores y mantener la idea de una sola memoria global.

Había indicado también que un sistema típico de nueva generación ccNUMA consistía de múltiples "motherboards cuadráticas" (cuatro procesadores), enlazadas mediante una tecnología de interconexión estandarizada denominada SCI (Scalable Coherent Interface). Este subsistema SCI es activado cuando la dirección de memoria de la data requerida no está ni en el caché de segundo nivel (L2) ni en memoria local "cercana"; es entonces que el SCI organiza los requerimientos de data en "paquetes" y los trasmite a los demás motherboards a través de anillos de alta velocidad, utilizando protocolos del tipo "snooping" hasta localizar las direcciones "lejanas" correctas. Luego la data, una vez ubicada, es enviada al procesador que la requirió y también es almacenada en la memoria caché de tercer nivel (L3) que reside en el "SCI bridge".

En general, el SCI emplea mensajería "send/receive" sincronizados para asegurar que los contenidos del Caché y la memoria RAM estén actualizados,

éste también mantiene directorios de estructuras tipo árbol sofisticadas para administrar la complejidad en las actualizaciones a memoria Caché y RAM; sin olvidarse de mencionar los mecanismos necesarios de protección y seguridad sobre la data.

. De esta manera ccNUMA y SCI trabajan en conjunto para mantener las instrucciones y la data en un solo "set", en el motherboard del procesador que los requiere, y lo más cerca posible a ese procesador. De esto se puede aseverar que la interconexión SCI colabora en esta arquitectura asegurándose que las transacciones remotas - entre "motherboards" - sean llevadas a cabo a muy altas velocidades, de por lo menos 1 Gbyte/seg. (Ver Grafico Nº 3)

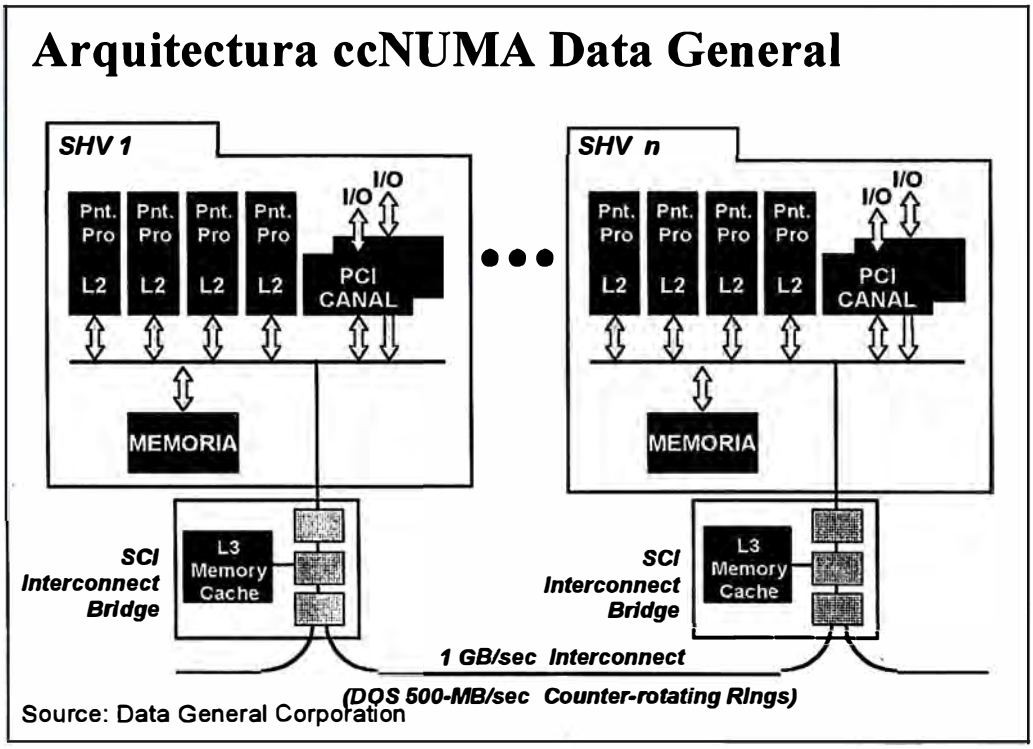


GRAFICO Nº 3. DETALLES DE LA ARQUITECTURA ccNUMA

ATRIBUTOS DE LOS SISTEMAS OPERATIVOS PARA TRABAJAR EN MODO NUMA

El sistema operativo es esencial para ayudar a la arquitectura ccNUMA, para mantener eficiencia en configuraciones donde existan muchos procesadores involucrados, así como aislar a las aplicaciones de las complejidades de hardware que esta arquitectura involucra.

Al nivel de estructura de datos, el sistema operativo debe tener: Soporte de multiprocesador, suficiente espacio de dirección para la identificación de los CPU's de modo que cada CPU tenga una única dirección, y debe tener capacidad de trabajo multi-nodal para administrar la memoria virtual del tipo "Page-Table".

El sistema operativo debe tener la capacidad de "Multi-threading", que es cuando varios procesos ("threads") son separados y ordenados para ejecución en diferentes procesadores. Adicionalmente el sistema operativo debe permitir un "locking" fino para los procesos, incluyendo los del Kernel, de tal modo que se logra una adecuada independencia de procesos.

Pero más allá de los atributos antes mencionados, que pocas versiones del UNIX los tienen y quizá sólo una o dos versiones todos juntos, es necesario que el sistema operativo optimice funciones importantes como son:

- **Process-affinity Scheduling** : Método para ordenar procesos relacionados, que compartan ciertos contenidos de memoria, de modo que sean ejecutados entre procesadores cercanos.

- **Memory-affinity Allocations** : Método para optimizar ubicaciones de memoria basándose en el ordenamiento de procesos previo de modo de

asegurar el nivel más alto posible de “hit rates” a la memoria caché de segundo nivel llamada L2.

Finalmente el sistema operativo debe permitir el ajuste o afinamiento de los algoritmos que administran los recursos basándose en una retroalimentación dinámica de la actividad del caché obtenidas vía hardware.

6. BENEFICIOS Y VENTAJAS DE LA ARQUITECTURA NUMA

De lo manifestado a lo largo del presente informe podemos anticipar que esta nueva tecnología NUMA no se debe considerar como un reemplazo mágico inmediato a las demás alternativas: Cluster, MPP e inclusive al SMP tradicional, ya que como toda tecnología deberá ir ganando su participación en el mercado informático por sus propios méritos, los cuales dependerán en gran medida de la atención, el talento y la ingeniería que los fabricantes le otorguen.

Dentro de este contexto creo importante destacar algunos de los atributos, beneficios y/o ventajas que esta arquitectura ofrece:

- Facilidad para su implementación: Como había indicado la arquitectura NUMA no es un reemplazo sino más bien una evolución de la conocida SMP, de modo tal que mientras los sistemas operativos y el Middleware estén específicamente adecuados y afinados para trabajar con ccNUMA se pueden estar usando las actuales herramientas y aplicaciones del modelo tradicional SMP.

- Compromiso y experiencia: Entre los proponentes de la arquitectura NUMA se encuentran fabricantes exitosos de la arquitectura SMP tales como: Convex, Data General, Sequent y Silicon Graphics. Estos fabricantes y diseñadores lograron implementaciones de SMP a niveles considerados

previamente como imposibles; por lo tanto creemos que sus conocimientos y experiencia son apropiados para obtener resultados importantes con NUMA.

- Costo-Performance: Como se ha dicho la arquitectura NUMA es fundamentalmente una aproximación SMP. La industria tiene ya una considerable experiencia e inversión en este modelo de programación y administración. NUMA preservando este modelo, permite que cualquier aplicación SMP existente pueda ser ejecutada sin modificación en el código en un sistema con arquitectura ccNUMA. Mientras algo de afinamiento será necesario para algún software, la gran mayoría de los esfuerzos recaen en los fabricantes para mejorar sus sistemas operativos o en los desarrolladores de las bases de datos que ya vienen trabajando muy cercanamente con los proveedores de equipos. El ahorro extraordinario que se logra con esto estoy seguro permitirá una aceptación casi inmediata de la arquitectura NUMA viéndola como una extensión de la ya conocida SMP.

- Aparición de SHV: Este anuncio de Intel no relacionado aún en ese momento con NUMA, hoy día viene a ser una de las piedras angulares en la estrategia que los fabricantes han considerado para sus sistemas NUMA. Por ejemplo Data General ya anunció y presentó lo que va a ser su nueva línea de servidores de alto nivel basándose en la tecnología SHV de Intel y la arquitectura ccNUMA, productos que estarán disponibles en sus primeros modelos para finales de 1997.

7. LA PREGUNTA DE LOS 64 BITS: ¿CUANTA MEMORIA ES SUFICIENTE?

Habiendo concluido con la visión evolutiva de los sistemas multiprocesadores, es importante para que éstos realicen un procesamiento efectivo de las bases de datos, destacar que requieren un razonable balance entre los recursos disponibles del sistema. Por ejemplo, así como las computadoras crecen y son cada vez más poderosas, a través de mayores velocidades del procesador y/o crecen en número de procesadores por sistema, para poder soportar aplicaciones con bases de datos cada vez más grandes, por lo tanto la necesidad de incrementar la memoria física, también llamada RAM o primaria, crece de modo proporcional.

Incrementos en capacidad de memoria primaria requiere también incrementos en sus equipos con memoria física cercana a esta cantidad, luego una pregunta importante a considerar sería: ¿Cuánta memoria es suficientemente adecuada para el procesamiento de las típicas bases de datos comerciales?

Existen dos rateos muy útiles para medir el balance de recursos: El rateo entre memoria y poder de procesador y el rateo entre memoria y rendimiento esperado del sistema, que pueden ser usados para obtener estimaciones sobre el dimensionamiento óptimo de la memoria en función de aplicaciones en base de datos. Los rateos de los sistemas actuales pueden ser calculados tomando como base los resultados publicados del Benchmark TPC y, por la aplicación de rateos típicos, las necesidades de memoria de sistemas futuros más grandes pueden ser fácilmente estimados.

8. REQUERIMIENTOS DE MEMORIA EN EL NUEVO MODELO

8.1 DETERMINANDO LOS REQUERIMIENTOS DE MEMORIA

Los Benchmarks estándares de la industria: TPC-C y TPC-D, fueron diseñados específicamente para representar necesidades típicas de procesamiento del tipo OLTP y DSS respectivamente. Ellos fueron definidos para requerir o reforzar grandes tamaños de bases de datos con a su vez mayores y más potentes sistemas computacionales. Que a su turno, requieren más memoria para poder soportar eficientemente esas aplicaciones. Por la misma naturaleza de estos benchmarks, es obvio que los proveedores que someten sus equipos a los mismos, usarán tanta memoria como sea necesaria en beneficio de la performance y/o hasta que sea sensible y económicamente conveniente. Por lo tanto, el rateo entre memoria física total (en GigaBytes) y potencia de procesador (en GigaHertz) provee una manera útil para determinar los requerimientos máximos de memoria para los sistemas de gran escala o superservidores.

8.2 NECESIDADES DE MEMORIA PARA UN AMBIENTE OLTP

A. MODELO DE USO DE MEMORIA TPC-C

Una gran variedad de experimentos se realizaron para investigar el impacto del tamaño de la memoria RAM en los resultados de performance obtenidos en el benchmark TPC-C. Uno de los factores de mayor peso, identificados con relación a la performance del TPC-C, es el

número de operaciones de I/O por cada orden nueva de transacción. En la FIGURA 1 se muestra, como, a tres diferentes tamaños de base de datos: 600, 900 y 1,200 "warehouses", el número de I/Os por cada nueva orden está directamente relacionada con el tamaño de memoria asignado al buffer de la Base de Datos. Esto sugiere que incrementando el tamaño de la memoria se debe mejorar la performance total del sistema.

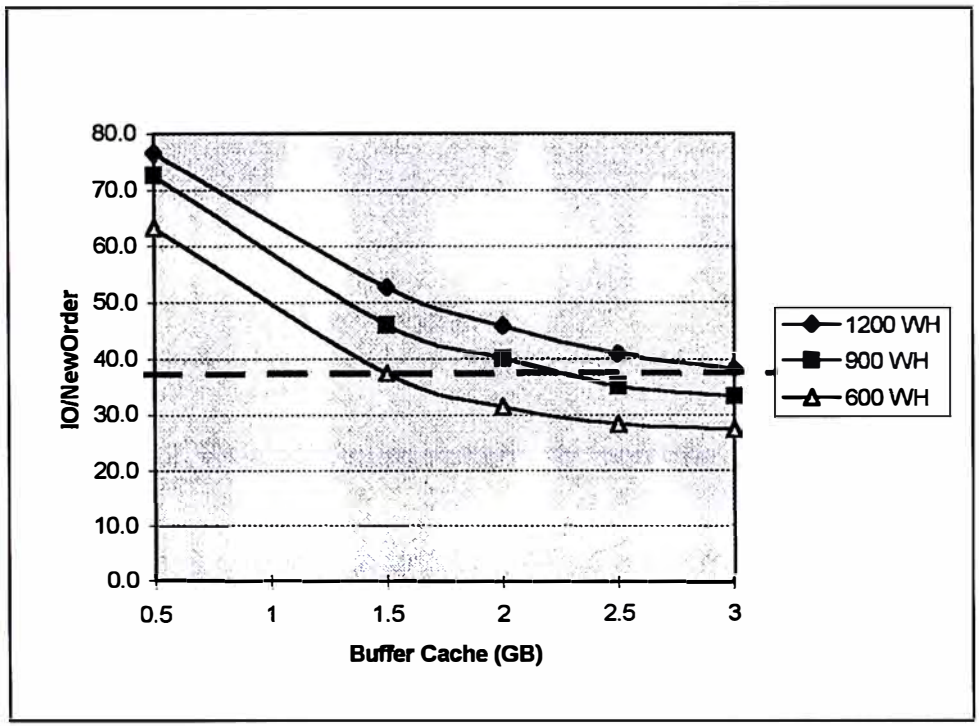


FIGURA 1: Relación entre tamaño de base de datos, memoria y I/Os

Al duplicar el tamaño de la base de datos desde 600 hasta 1,200 "warehouses" se requiere duplicar la memoria desde 1.5GB hasta 2GB, de este modo se mantiene constante el rateo de I/O. Por lo tanto, desde el momento en que una base de datos de 600 requiere por lo menos 3GB de memoria para minimizar el número de I/O, se espera que la otra base de datos de 1,200 requerirá de por lo menos 6GB. De acuerdo a las

reglas del TPC-C, una base de datos de 1,200 "warehouses" debe corresponder a la performance de un sistema de alrededor de 14,000 tpmC. Lo cual produciría un rateo GB/ktpmC de 0.43.

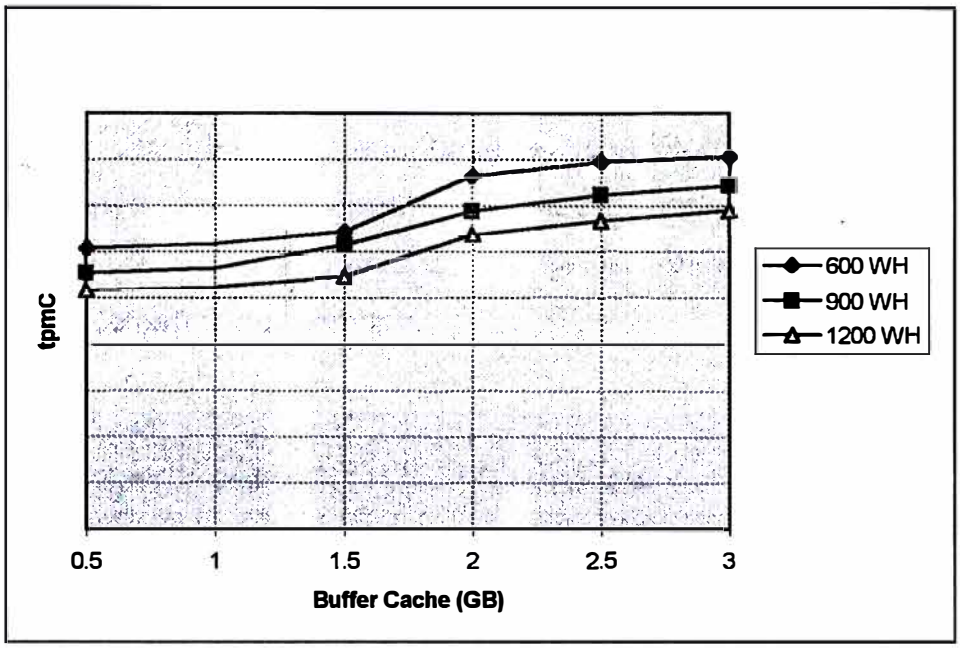


FIGURA 2: Relación entre tamaño de memoria y performance

Habiendo demostrado que un incremento en la memoria puede reducir el número de I/Os de manera significativa, luego su impacto en la performance fue evaluado. La Figura 2 muestra como la performance en el caso de los 600 "warehouses" puede incrementarse en cerca de 25% cuando la memoria se incrementa de 1.5GB a 3GB. Desde el momento en que los requerimientos de memoria son una función lineal del tamaño de la base de datos, se espera que para la base de datos de 1,200 "warehouses" ocurriría un beneficio similar en la performance si se incrementa la memoria de 3GB a 6GB.

B. NECESIDADES DE MEMORIA EN LOS TPC-C PUBLICADOS

Muchos benchmarks TPC-C publicados en el último año sirvieron de análisis para determinar donde la esperada relación entre memoria y performance mantenían coherencia en un ambiente competitivo. La Tabla 1 muestra que el esperado rateo de 0.43 GB/KtpmC está cerca de la media del rango obtenido: 0.21-0.56. En adición se observa, que los tamaños de memoria generalmente se incrementan con la mayor performance, ocasionalmente limitados por restricciones en la configuración física o la incapacidad del sistema operativo de soportar mayores tamaños de memoria.

Sistema	Tamaño de memoria (GB)	tpmC	GB/Ktpm C
IBM RS/6000 Power PC Server J40 c/s	2	5774	0.35
SIN RM600 Model 720 c/s	2	9524	0.25
HP 9000 Model K460 c/s	4	14739	0.27
Sun Ultra Enterprise 4000 c/s	5	15462	0.32
Sun Ultra Enterprise 4000 c/s	5	18438	0.27
Sun Ultra Enterprise 6000 c/s	5	23143	0.22
Digital AlphaServer 8400 5/35 c/s	6	14177	0.42
Digital AlphaServer 8400 5/350 c/s	8	14227	0.56
SGI Origin2000 Server c/s	13	25309	0.51

TABLA 1: TPC-C Necesidades de memoria[1]

[1] Información completa de los TPC-C se adjunta en el Anexo A.

C. NECESIDADES DE MEMORIA PARA UN AMBIENTE DSS

NECESIDADES DE MEMORIA DE LOS TPC-D PUBLICADOS

La Tabla 2 resume las necesidades de memoria para una muestra representativa de los benchmark TPC-D publicados en el último año, esto incluye sistemas de arquitectura SMP y MPP. El rateo GB/GHz fue escogido para analizar las necesidades de memoria en este benchmark por las siguientes razones:

- La velocidad de reloj de los procesadores se correlaciona muy de cerca con la performance del sistema en este benchmark TPC-D.
- No existe un tamaño de base de datos como requerimiento atado a la performance.
- Gran parte de la memoria no es usada como caché buffer de los blocks de la base de datos.

Nótese en la tabla 2 que ningún sistema necesita más de 4 GB/GHz para optimizar la performance o la relación precio/performance.

	Sistema	Memoria (GB)	CPUs	CPU MHz	Total MHz	GB/GHz
	Digital AlphaServer 8400 5/440	4	12	440	5,280	0.75
*	NonStop Tandem Himalaya K2000-16	4	16	200	3,200	1.25
*	Sun Ultra Enterprise 6000	12	24	250	6,000	2.00
*	Sun Starfire Ultra Enterprise 1000	16	64	250	16,000	1.00
	NCR 5100M 20 Node System	20	160	133	21,280	0.94
	HP 9000 EPS22 (12x4-way cluster)	24	48	180	8,640	2.78
	IBM RS 6000 SP Model 306	24	96	112	10,752	2.23
	IBM RS6000 SP	32	128	66	8,448	3.79
	Pyramid Reliant RM1000	48	96	250	24,000	2.00

* MPP

TABLA 2: TPC-D Necesidades de memoria[2]

[2] Información completa de los TPC-D se adjunta en el Anexo B.

LIMITACIONES PARA LA CONFIGURACION DE MEMORIA RAM

La Tabla 3 resume los límites de configuración de memoria para algunos sistemas grandes así como sus respectivos rateos de GB/GBz.

Sistema	Memoria (GB)	CPUs	CPU MHz	Total MHz	Max. GB/GHz
Data General AV5900	4	4	200	800	5.0
Data General AV20000-32CPUs	32	32	200	6,400	5.0
Digital AlphaServer 8400 (typical max. Used)	8	8	450	3,600	2.2
Digital AlphaServer 8400 (possible, but rare)	14	2	450	900	15.5
Sun Ultra Enterprise 5000	14	14	250	3,500	4.0
Sun Ultra Enterprise 6000	30	30	250	7,500	4.0
Sun Ultra Enterprise 10000	64	64	250	16,000	4.0

TABLA 3: Límites de Configuración de Memoria

D. LA ALTERNATIVA: CAPACIDAD DE CRECIMIENTO FUTURO

Si miramos a las tradicionales y actuales tecnologías y las contrastamos con las tendencias de arquitecturas de escalamiento modular basados en bloques o nodos, bajo dos opciones: MPP y NUMA, siendo esta última la más innovadora y comercial por su nivel de compatibilidad con las herramientas y aplicaciones del mundo SMP. Esta tecnología asegura que tanto la memoria y procesadores sean añadidos al sistema de modo sincronizado y sinérgico. Del mismo modo podemos notar que en comparación con las otras tecnologías, que tienen un número fijo de slots, se ven forzados a distribuirlos entre tarjetas de memoria y tarjetas

de procesador, de tal suerte que es necesario remover tarjetas de memoria si se quiere colocar más tarjetas de procesador y viceversa.

Por otro lado, si consideramos que la máxima memoria configurada para un sistema en el benchmark TPC es de 16GB (ver tabla 2), y desde que no todos los fabricantes son capaces de configurar mayor memoria que esta, resulta razonable y lógico concluir que más memoria no sería ni beneficioso ni adecuado en costo. Basandose en estadísticas, se sabe que la velocidad de los procesadores se duplica cada 18 meses, o lo que es lo mismo se cuadruplica a los tres años; por lo tanto, si las necesidades de memoria están estrechamente relacionadas al poder de procesamiento, luego, el tamaño máximo de la necesidad de memoria, para los próximos tres años, será de 64GB (16GB x 4).

Del mismo modo, mirando la información desde la perspectiva de comparar memoria y performance, una conclusión similar se puede alcanzar desde los datos de la tabla 1. El sistema de mayor performance obtiene 25,309 tpmC y el rateo GB/KtpmC es de 0.56. Para el peor escenario, combinando estos dos factores y luego extrapolando la performance del sistema para los próximos 3 años por un factor de cuatro veces, similar al caso anterior, el resultado del requerimiento de memoria más alto será:

25,309 x 4/1000 x 0.56 = 56.7 GB

Como se puede observar una cantidad razonablemente dentro del límite anterior.

E. COMO EL HARDWARE SOPORTA GRANDES CANTIDADES DE MEMORIA

En la actualidad existen tres diferentes tamaños de direccionamiento a memoria principal muy usados por la industria. La Tabla 4 muestra que tanta memoria puede ser direccionada por cada uno de ellos.

Capacidad de direccionamiento (Bits)	Máxima memoria (GB)
32	4
36	64
64	16,000,000,000

TABLA 4: Límites de la Memoria Física

De acuerdo a las proyecciones, los procesadores de 64bits estarán disponibles de modo masivo de acá a tres años, mientras tanto, como se ha visto anteriormente no existe una necesidad imperiosa o mandatoria por memoria física más allá de los 64 GB. Y como se observa en la TABLA 4 esta cantidad de memoria puede ser soportada con arquitecturas de 36 bits de direccionamiento, tal cual provee actualmente el más popular de los chips en el mercado: Intel Pentium Pro.

F. COMO EL SOFTWARE SOPORTA LA CAPACIDAD DE MEMORIA

La capacidad de soportar el tamaño máximo de memoria involucra, por el lado del software, no solo al Sistema Operativo sino también a las

herramientas de software (ej. La Base de Datos) que requieran acceso a más de 4GB de memoria compartida.

Sin entrar en detalles de como se usa la memoria en los sistemas OLTP y DSS, basta con decir que difieren en ciertos aspectos, y como resultado de esto tienen requerimientos de software también diferentes. Pero algo que es común a ambos es el requerimiento que el sistema operativo sea capaz de direccionar toda la memoria física disponible, usando por lo menos los 36 bits para poder contener valores únicos de las diferentes direcciones de memoria.

Muchas de las tareas de una aplicación DSS pueden hacer buen uso de grandes tamaños de memoria de uso privado, como es el caso por ejemplo, de memoria usada para un proceso SORT, y cualquier otro proceso en memoria temporal del tipo no compartida. De tal modo que a medida que estos procesos se incrementan, el total de memoria consumida fácilmente excederá los 4GB, sin que ninguno de estos procesos por si solo requiera más de esa cantidad. Este tipo de procesamiento requiere que no se hagan cambios ni a la aplicación ni a la Base de Datos.

Para las tareas de una aplicación OLT, es usualmente deseable que todos los procesos compartan un tamaño considerable de memoria del tipo caché buffer. En este caso, cada proceso va a necesitar accesar más de los 4GB de memoria que tradicionalmente es una limitación del software de base disponible para la plataforma basada en el procesador Pentium Pro. Con la finalidad de salvar esta barrera y conscientes de la importancia sobre el particular, proveedores de sistemas de alto rendimiento como Data General Corp., han mejorado sus Sistemas Operativos en aspectos como:

- Soporte a través del sistema operativo de la capacidad de direccionar por lo menos 36 bits y por ende hasta 64GB de memoria principal.

- Desarrollo de interfaces de Aplicación (API) que permitan a cualquier proceso tener acceso a los 64 GB de memoria física soportada actualmente por el Pentium Pro.

5 UNA EXPERIENCIA PRACTICA CASO SAINSBURY

Este es un caso práctico en donde la aplicación de la tecnología NUMA evidencio un beneficio importante para las operaciones comerciales de esta compañía inglesa.

Sainsbury es una Compañía dedicada a la comercialización de productos al menudeo mediante el esquema de supermercado y tienda por departamentos en el Reino Unido, siendo una de las compañías más grandes en ese mercado tan competitivo, donde una de las variables más importantes del marketing de consumo es la lealtad de los clientes. Con el objetivo de crear valor y ventaja competitiva, Sainsbury decidió implementar un programa que consistiría en lanzar una tarjeta de consumo frecuente, de modo tal que pueda almacenar apropiadamente en una base de datos estructurada, enormes cantidades de detalles sobre las compras de cada cliente durante un periodo de tiempo, teniendo en consideración que el universo de clientes frecuentes era de 9 millones y que estos realizaban el 90% de las transacciones, se requería por lo tanto una plataforma informática de características muy especiales y con una relación Precio/Rendimiento adecuada.

Este proyecto pretendía ser un tradicional proyecto de "Datawarehousing" en el cual se debía escoger entre las aplicaciones y bases de datos tradicionales para el caso y por lo tanto una plataforma de hardware acorde a esto: Un tradicional Mainframe o un "innovador" sistema de procesamiento paralelo. De este modo se inició una serie de pruebas y evaluaciones detalladas desde el

punto de vista técnico, con el objetivo de medir resultados en las diferentes plataformas de hardware que se consideraron como alternativas. Desde ese primer momento la arquitectura NUMA de Servidor Data General AV20000 comenzó a destacar por sobre las demás, entre las cuales podíamos encontrar: Hewlett-Packard, IBM, Pyramid y Sequent como finalistas.

Las pruebas se realizaron con un sistema de 16 procesadores y almacenamiento en discos externos en modalidad RAID de alrededor de un Terabyte, con el sistema operativo UNIX y la base de datos Oracle8. Como conclusión de estas pruebas el sistema AV20000 de arquitectura NUMA demostró tener el mejor resultado de Precio/Rendimiento.

Finalmente el proyecto se inicio con la instalación de dos servidores NUMA AV20000, uno con 32 procesadores y el otro con 16 procesadores, la base de datos Oracle8 con un tamaño inicial de un Terabyte y capacidad de crecer a tres Terabytes.

Hoy en día el sistema permite que la gerencia pueda a través de consultas complejas contra una base de datos masiva y luego de ejecutar formulas matemáticas, encontrar patrones de consumo y comportamiento de los clientes de modo individual, preparando de este modo las campañas de ventas y marketing mejor enfocadas preparando y hasta personalizando cada tienda en función de los hábitos y patrones de consumo de sus clientes. Esta solución permitió incorporar el control de productos por categorías en vez del típico "por producto" de modo que se empaquetaba un producto primario con otros complementarios, incrementando de este modo el volumen de ventas y la rotación de los diferentes productos.

A continuación se presenta un cuadro referencial comparativo en términos de Precio/Rendimiento de las plataformas evaluadas.

PLATAFORMA	SISTEMA NUMA DG AV20000	SERVIDOR RISC HP 9000 EPS22	SISTEMA MPP NCR WORLDMARK 5150 Server
# de procesadores	De 4 a 32	Hasta 64	Hasta 256
Capacidad de RAM	64GB	Mayor a 64GB	384GB
Almacenamiento	300 GB	300 GB	300GB
Sistema Operativo	UNIX	UNIX	UNIX
Base de Datos	Oracle	OPEN	TERADATA
Precio US$	2'710,643	8'029,048	12'269,156
TPC-D(300GB) Precio/Rendimiento	**$1,319**	**$1,982**	**$1,919**

Como se puede observar del cuadro resumen el menor costo por transacción que se puede obtener en un ambiente de DSS, según el benchmark TPC-D, pertenece al sistema de arquitectura NUMA, que demuestra en sus primeras pruebas responder de manera efectiva y respaldar las expectativas que esta novedosa arquitectura trae al mercado.

Queda claro que siendo la tecnología tan dinámica, los valores que estoy usando en el presente informe van a cambiar en el tiempo, pero lo que también es cierto la tendencia mostrada y cuya hipótesis he manifestado, para los avances tecnológicos, permitirán que las comparaciones mantengan una cierta proporcionalidad mientras esta o aquella tecnología siga vigente y en uso.

10. CONCLUSIONES Y RECOMENDACIONES

1. Quizá debiéramos comenzar por contestar la pregunta original sobre la necesidad de los 64 bits, respuesta que resultó ser más práctica que teórica y que podríamos asumir en que aún no ha llegado el momento para los 64 bits. Aunque algunos proveedores empiezan a ofrecer arquitecturas con esta caracteristica, un análisis, a la luz de lo explicado, muestra que estos ofrecimientos no crean automáticamente la capacidad de configurar y usar a la vez grandes cantidades de memoria. Veamos un par de ejemplos:

   - El servidor DEC AlphaServer 8400 tiene un número fijo de slots, los cuales limitan el tamaño de la memoria con que se pueda configurar el sistema. Tarjetas de CPU deben ser removidos para dar cabida a tarjetas de memoria y de ese modo ampliar su capacidad, esto obviamente atenta el balance deseable entre la cantidad de procesadores y la cantidad de la memoria RAM.

   - Los servidores SUN permiten expandir la memoria independiente o conjuntamente con el número de procesadores, pero por el lado del sistema operativo Solaris 2.5., según información obtenida en el respectivo web, este es incapaz de soportar los 64 bits limitando el acceso a no más de 4 GB de memoria compartida desde una aplicación.

2. Otra conclusión importante es que la tecnología en este aspecto no se reinventa un día si otro día no, sino por el contrario está en constante evolución, en ese sentido la arquitectura de los 64 bits es una proyección aceptable y con mucho sentido que deberá estar pragmáticamente implementada de modo masivo en tres años, mientras tanto la evolución nos facilita el contar hoy con alternativas que van más allá de los limitantes 32bits, insuficientes para las aplicaciones comerciales de hoy, tal cual es la capacidad de soportar 36 bits, hoy disponible en las plataformas basadas en la arquitectura Intel Pentium Pro y los sistemas operativos que a su turno también soporten esta misma capacidad.

Por lo tanto, el impacto más importante en la evolución de las tecnologías que he presentado se concentra en la llamada Arquitectura NUMA, la cual sí encuentra, hoy día, requerimientos y demandas importantes de parte de las aplicaciones, que obliguen a pensar en nuevas tecnologías o evoluciones de las existentes. Adicionalmente podemos observar como cada vez más los componentes de una determinada arquitectura van transformándola en una especie de "comoditie" donde el valor diferenciador casi ya no existe.

3. Luego el reto para los proveedores ya no es quien diseña el Chip más veloz, o el disco duro de mayor capacidad, la memoria de mayor densidad o finalmente el servidor más rápido; otros se especializaran y lo lograran en cada uno de esos rubros. Los verdaderos retos son:

   a) Cómo integrar los nuevos componentes, caso del SHV, dentro de su línea de productos de modo tal que se aproveche esta tecnología.

   b) Cómo construir sistemas grandes y escalables que puedan aprovechar estos componentes

c) Cómo migrar las aplicaciones y herramientas sin pasar por procesos traumáticos y manteniendo al menos la relación precio/rendimiento.

Finalmente solo el tiempo y el mercado dará su veredicto final permitiendo una vez mas que una tecnología de avanzada juegue un papel importante en la transición hacia los sistemas y tecnologías del tercer milenio.

ANEXOS

ANEXO A

RESULTADOS TPC-C

Empresa	Sistema	Throughput (tpmC)	Precio/perfo. ($per tpmC)	Costo Total	Availability
Digital	AlphaServer 8400 5/350 c/s	14227.25	$ 253	$ 3,586,486	01 June 1996
Digital	Digital AlphaServer 8400 5/350	14176.61	$ 199	$ 2,812,287	15 Dec. 1996
HP	HP 9000 Model K460 c/s	14739.03	$ 133	$ 1,949,237	30 June 1997
IBM	RS 6000 PowerPC Server J40 c/s	5774.07	$ 198	$ 1,143,858	15 Dec. 1996
SGI	Origin 2000 Server c/s	25309-2	$ 140	$ 3,519,012	29 Oct. 1997
SIN	RM600 Model 720 c/s	9524.47	$ 297	$ 2,833,151	01 June 1997
Sun	Ultra Enterprise 6000 c/s	18438.70	$ 140	$ 2,566,159	01 June 1997
Sun	Ultra Enterprise 4000 c/s	15461.87	$ 135	$ 2,077,487	01 Aug. 1997
Sun	Ultra Enterprise 6000 c/s	23143.65	$ 119	$ 2,734,009	31 Aug. 1997

ANEXO B

RESULTADOS TPC-D - 100 GB

Empresa	Sistema	Power (QppD)	(QthD)	Price/Perf ($per QphD)	Costo Total	Total N"CPU	Avail-ability
Digital	AlphaServer 8400 5/440	972.9	506.4	$ 1,920	$1,347,538	12	2/97
IBM	RS 6000 SP Model 306	755.3	384.4	$ 6,935	$3,736.428	96	3/97
NCR	S100M 20 Node System	1441.9	1150.4	$ 9,043	$11,645,965	160	11/96
Tandem	NonStop Himalaya K20000-16	392	214	$ 12,304	$3,563,442	16	8/16

RESULTADOS TPC-D - 300GB

Empresa	Sistema	Power (QppD)	(QthD)	Price/Perf ($per QphD)	Costo Total	Total N"CPU	Avail-ability
NCR	5100M 20 Node System	1441.9	1150.4	$ 9,043	$11,645,965	160	11/96
HP	9000 Model EPS22	3416.4	1673.7	$ 3,007	$ 7,191.171	48	10/97
IBM	RS600 SP	1360.5	916.3	$11,970	$13,364.002	128	3/96
Pyramid	Reliant RM1000	1169.2	749.3	$10,225	$ 9,570,337	96	2/97
Sun	Ultra Enterprise	1006.0	588.0	$ 3,447	$ 2,651,291	24	5/97
Sun	Starfire Ultra Enterprise 10000	1787.9	1122.3	$ 3,562	$ 5,045,645	64	5/97

ANEXO C

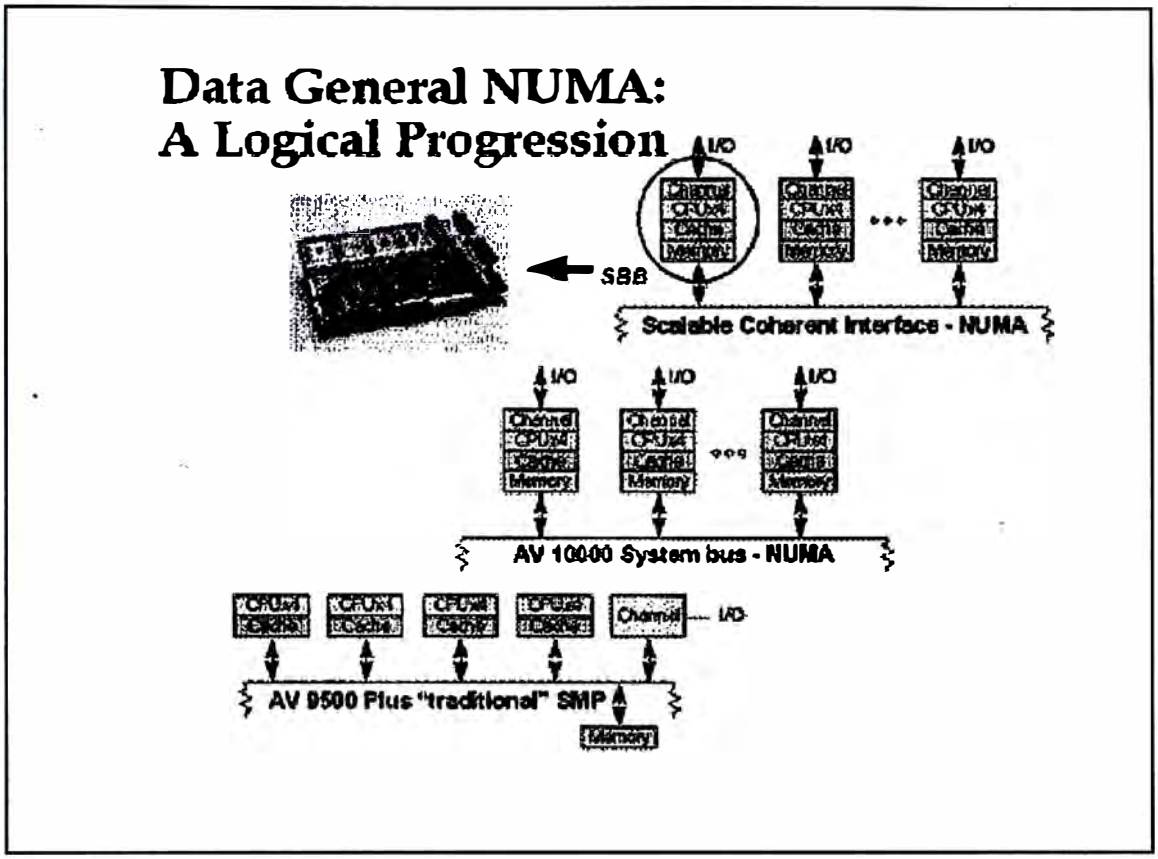
The NUMA Architecture

NUMA (Non-Uniform Memory Access) is a system topology that is the foundation for next-generation SMP (symmetric multiprocessing) systems. Based on "commodity" processing building blocks and physically distributed, but logically unified, coherent memory, NUMA extends the power and performance of SMP systems while preserving the shared memory programming model. As a result, today's SMP applications can run on NUMA systems without change, and new applications can be developed without extensive re-tooling.

Why NUMA?

Extending SMP

For Data General, the NUMA architecture represents a logical progression in the development of large-scale systems for an enterprise's mission-critical applications. Data General's AViiON UNIX-based product line was introduced in 1989 with built-in hardware and operating system support for SMP. This line of SMP systems, based on the Motorola 88K chip, culminated with the AV 9500 16-processor server.



Throughout this development effort, Data General continually improved DG/UX, our version of the UNIX operating system, to provide performance scalability as the number of processors increased. However, the ``big bus'' design that underlies SMP systems reaches a point of diminishing return on performance scalability as system traffic -- CPU, memory, and I/O – saturate the bus bandwidth. In addition, bus length limitations constrain the number of slots available for system expansion. This forces users to trade-off the number of CPUs, amount of memory, and I/O capacity when configuring systems.

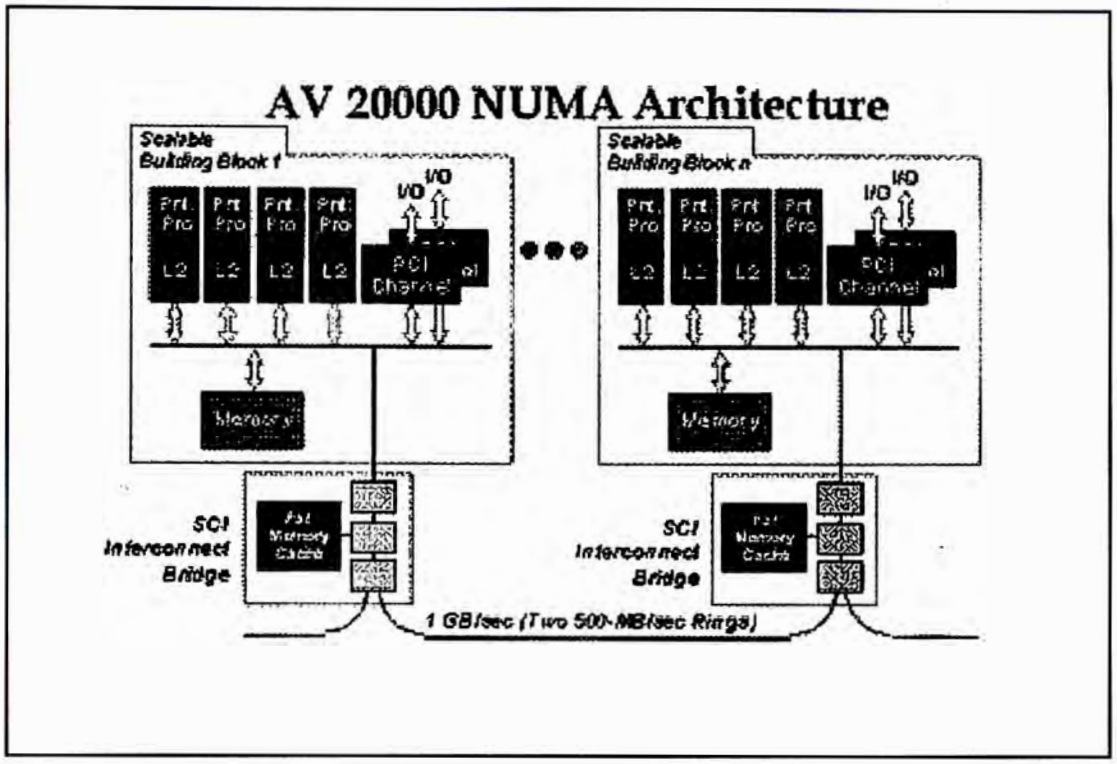
The NUMA architecture overcomes these limitations while maintaining SMP software compatibility through system-wide shared memory and management of all system resources by a single copy of the operating system. In addition, the NUMA architecture supports linear system expansion for I/O and memory as processing building blocks are added, and there is no single backplane bus.

First-Generation NUMA: AViiON AV 10000 Server

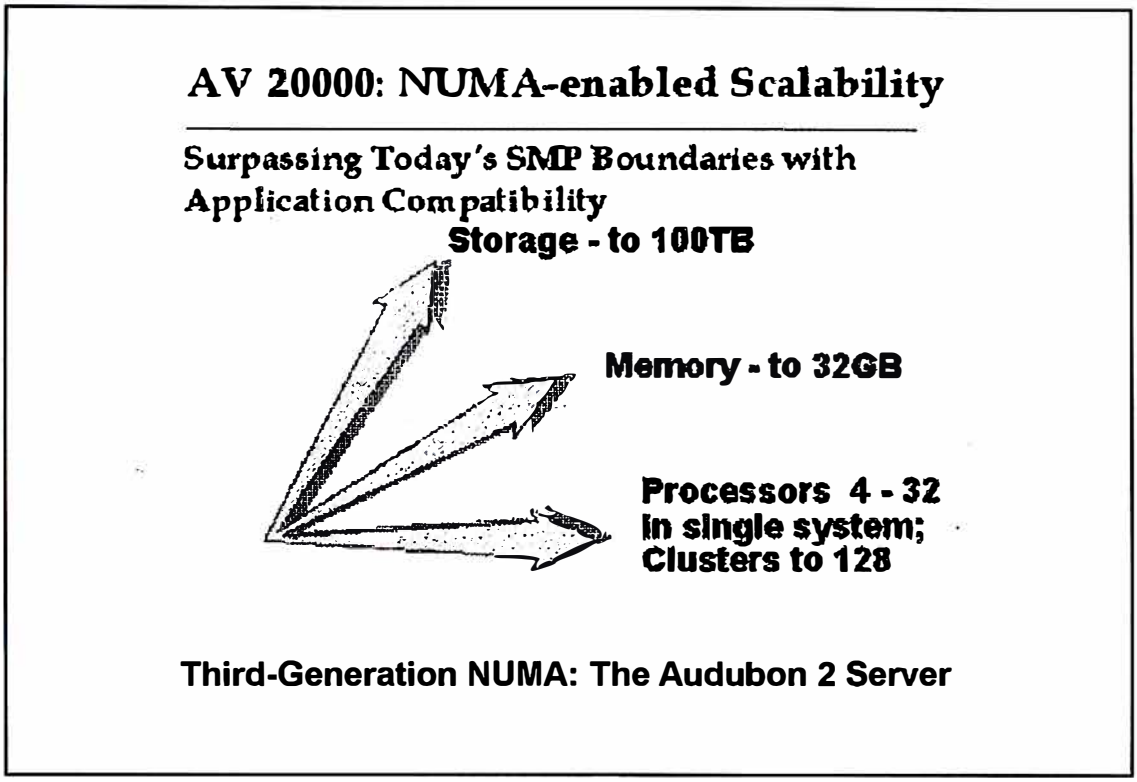
Data General first implemented the NUMA architecture in the AV 10000 server, based on Motorola 88K chips, which started shipping in 16- and 32-processor configurations in January 1996. The basic building block of the AV 10000 is a four-processor motherboard with its own memory and I/O. Building blocks are interconnected by dual system buses. These buses are also used to maintain memory coherency across building blocks. Further refinement of DG/UX ensured optimized performance through localizing algorithms which strive to ensure that CPU, memory, and I/O for a particular process will all reside in the same locale. In addition, Data General included a large third-level cache as part of this system design.

Second-Generation NUMA: AViiON AV 20000 Server

Data General leveraged the commodity economics advantages of the Intel Architecture in its second-generation NUMA implementation, the AV 20000 server, which began shipping in June 1997. Built from Intel Standard High Volume (SHV) server four-processor Scalable Building Blocks (SBB), the AV 20000 uses the IEEE/ANSI-standard Scalable Coherent Interface (SCI) interconnect's fast, 1-GB/sec link between blocks with dual counter-rotating rings for performance and availability. Each SBB also maintains a large third-level cache that works with the SCI logic to enforce cache coherence across the entire fabric of interconnected motherboards.



The AV 20000 is available with up to 32 200-MHz Intel Pentium Pro processors, optional 1-MB L2 cache, up to 32 GB of memory, and up to 100 Terabytes of CLARiiON Fibre Channel storage. In clustered configurations, the processor count grows to 128. Each SBB can have as much as 128 MB of far memory cache. This powerful system plus many N+1 features, M3D server management, AV/AlertSM diagnostics, component deconfiguration, and ``call home'' support combine to provide a highly available, highly scalable, and cost-effective platform for online transaction processing (OLTP), enterprise resource planning (ERP), and data warehousing applications.



Data General recently announced that its third-generation NUMA systems, code-named Audubon 2, will be available by the end of this year. These systems will be based on Intel's forthcoming Pentium II Xeon processors and will support up to 64 processors, up to 64 GB of memory, and up to 400 Terabytes of CLARiiON Fibre Channel storage in a single-system. The AV 20000 is easily upgradable to this next-generation processor by simply adding SBBs based on Pentium II Xeon processors.

ANEXO D

Implementation and Proformance of a CC-NUMA System

1. Introduction

"NUMA-Q" is a Cache Coherent Non-Uniform Memory Access (CC-NUMA) Multiprocessor designed and built by Sequent Computer Systems, Inc. for large-scale commercial computing. It represents the first major architectural change for Sequent since we first introduced commercial Symmetric Multiprocessor (SMP) systems 12 years ago. This change in architecture is the result of two major factors:

a. The increasing gap between microprocessor speeds and memory access latency in single shared-bus systems, and
b. The emergence of off-the-shelf small-scale SMP building blocks.

While the first factor forces an architectural change for manufacturers of mid-range and high-end computer systems, the second enables a cost-effective solution which leverages commodity components.

As stated above, the NUMA-Q design resulted from the realization that single large busses cannot connect large numbers of processors while still providing both the bandwidth capacity and memory access latency required by today's high speed microprocessors. Large bus-based systems may provide high bandwidth to a large number of processors, but their ability to support newer high speed processors is limited due to their latency characteristics. There are two factors which increase latency in big bus systems. First, with large numbers of processors, average latency is increased due to contention as well as design choices that favor throughput over latency. Second, bus clock frequencies are limited with the greater

number of memory controllers and I/O bus bridges required for a balanced system with many processors. It is very difficult to achieve high bus frequencies for these large systems, given the limits imposed by both propagation delay and electrical loading.

In order to overcome these limitations, NUMA-Q uses multiple 4 processor SMP Quads, each with a high bandwidth bus with low memory access latency. This Quad is based on the PENTIUM® PRO processor and the external bus it defines. Scalability to system sizes with more than 4 processors uses a second-level interconnect based on the Scalable Coherent Interface (SCI) specification [1]. This standard point-to-point interconnect breaks the length restriction of the single shared bus, since signals no longer need to propagate within a single clock cycle. Despite the increased latency for memory accesses that require transfers across both the first-level and second-level interconnects, the majority of memory references complete within the Quad where they originate, and average latency is still low. This approach combines high scalability with low latency, whereas shared-bus systems must trade off between these two attributes. A key enabler for the NUMA-Q architecture is the off-the-shelf 4-way SMP building block. The PENTIUM PRO processor and its accompanying chip set implement this building block, and we believe its availability to computer systems manufacturers will make NUMA-Q-like architectures a new standard. This observation is based on the impact that integrated 32 bit microprocessors had on the industry over the past 10 years.

In the early Eighties most computer companies designed their own instruction set and implemented their CPUs on Printed Circuit Boards (PCBs). Much of the effort and resources expended in developing a new system went into instruction set design and implementation of the CPU. As integrated 32 bit microprocessors became available several companies, including Sequent, decided that computer systems could be built more cost

effectively using these off-the-shelf components as the fundamental building block. The microprocessors did not quite match the performance of the board-level CPUs, but research in cache protocols provided relatively simple mechanisms for attaching several microprocessors to a common memory bus. Also, because the microprocessor-based CPU was substantially smaller than the PCB-based CPU, it was feasible to connect several CPUs to a common backplane. Building systems with more than one CPU provided a performance multiplier that overcame the single processor performance deficit of the microprocessors of the day. Over time, most computer companies that survived shifted over to the microprocessor-based SMP systems that are prevalent today. The economies of the microprocessor forced this shift.

We believe that the new SMP building blocks will cause a similar shift for computer systems manufacturers. The economies of these small scale SMP systems cannot be ignored in high-end computer design. New computer systems will emerge which incorporate these building blocks, and computer companies will shift their focus from large shared busses to point-to-point interconnects and protocols for communication between computing Building Blocks. Sequent is moving in this direction, and NUMA-Q is the first system to result from this trend.

2. NUMA-Q Architecture

2.1 Architectural Options

Given the basic building block of the four processor SMP system, there are four architectures to consider for commercial systems. Each of these architectures is addressed below.

2.1.1 Shared Nothing

Shared Nothing, or Massively Parallel Processing (MPP), architectures, such as the Intel Paragon® [2] and the Pyramid Reliant® 1000 [3], are attractive from a hardware developer's viewpoint because they present much simpler problems to solve. There is no hardware support for shared memory or cache coherency, so scalability to large numbers of processors is straightforward (hence the term "Massive"). These systems have been shown to provide high levels of performance for compute intensive applications with statically partitionable data and minimal requirements for node-to-node communication. However, most commercial applications are not well-suited to MPPs, as databases change in structure over time, and the overhead of re-partitioning the data to avoid hot spots is too great in a continuously available environment.

Furthermore, since commercial Database Management Systems (DBMSs) have not yet migrated to the shared nothing model, they still require frequent communication of small amounts of control information between processes. This communication model is not consistent with MPPs, which overcome a high communication latency with infrequent communication of very large data blocks. While DBMSs may eventually overcome the problem of data skew and succeed in a full migration to the shared nothing model, it is clear that for the time being commercially successful architectures must continue to provide a low latency shared-memory model.

2.1.2 Hierarchical Bus

Hierarchical Bus systems, such as the proposed Gigamax [4] from Encore, utilize multiple processors under an inclusive third level cache. Multiple third level caches communicate via a higher level bus and use a standard snooping protocol to maintain coherency among themselves.

Coherency traffic is filtered by the third level caches and passed down the local busses as necessary. These systems suffer from the same access latency and length pressures as the simple SMP systems. The problems are exacerbated by the form factor of the SMP building blocks which include I/O controllers and memory SIMMs, and therefore cannot be easily stacked on a backplane or midplane.

2.1.3 COMA

At first glance, a Cache Only Memory Architecture (COMA) based system similar to the Data Diffusion Machine [5] or KSR-1[6], appears inviting. It has the potential to work well with off-the-shelf or "shrink-wrapped" operating systems (OS's) that are unaware of the non-uniform access latency of the distributed memory. Building such a system is complicated by the fact that the physical address of a request can map to different blocks of DRAM over time, as the COMA mechanisms move blocks among Building Blocks. The implementation of this mapping can be done in one of two ways: either by modifying the memory controller or by modifying the virtual memory subsystem, as in Simple COMA [7]. With a shrink-wrapped OS the virtual memory subsystem cannot be modified, so a custom memory controller is required.

Evidence available to date still suggests that the leading shrink-wrapped OS's do not scale much beyond two to four processors [8]. Adding the development cost of a custom memory controller to the project to allow for hardware scalability in shrink-wrapped environments does not make sense given the software scalability limitations.

When one considers systems running modified versions of UNIX, it is apparent that NUMA aware page allocation policies, including page replication and migration, can provide in software much of the benefit of the hardware page replication and migration that COMA provides [9].

Furthermore, it should be noted that the advantage of COMA is perceived to be a large reduction in capacity misses at each Building Block when compared to a directory-based CC-NUMA system. However, when sufficient cache size is provided in combination with OS restructuring, capacity misses are virtually eliminated in the CC-NUMA system, and Building Block-to-Building Block traffic becomes dominated by communication (or coherency) misses. COMA systems offer no advantage over directory-based CC-NUMA systems for handling communication misses, as noted in [9] and [10]. This is discussed further in Sec. 4 below.

2.1.4 CC-NUMA

CC-NUMA systems, e.g. DASH[11], FLASH[12] and Alewife[13], provide a viable method of scaling using small single processor or SMP building blocks without the drawbacks of the other options. The familiar shared memory programming paradigm is maintained. Software need not understand the NUMA characteristics of the system, provided there is sufficient memory cache at the building block level. This need for cache can be offset to some degree using the OS restructuring techniques mentioned above. The interconnect for these systems can utilize point-to-point connections which do not have the physical length limitations of multidrop busses. Finally these systems can be built using the standard processor and memory controller chip sets, provided that the SMP bus of the building block provides a protocol that enables memory requests to be handled by a bus agent other than a processor or memory controller, and allows some form of out-of-order responses.

2.2. NUMA-Q Architecture

Figure 2.1 shows the high level block diagram of the NUMA-Q architecture. NUMA-Q is a collection of homogenous Building Blocks interconnected via a set of high speed point-to-point links. Each

Building Block is a complete 4 processor SMP machine, or Quad, containing processors, caches, memories and I/O busses. Every processor in every Quad has a common view of the system-wide memory and I/O address space. Within a Quad, cache coherence is maintained using a standard MESI snooping cache protocol. Each Quad contains a bridge board (called IQ-Link) that plugs into the local SMP bus. Included on the board is a 32 Mbyte, 4-way, remote cache which maintains copies of blocks fetched from remote memories. This board interfaces the caches and memory on the local SMP bus with caches and memories on remote SMP busses. It implements a directory-based cache protocol based on SCI to manage the coherency of the local and remote caches. In addition, it provides mechanisms for bridging interrupt messages to remote Quads and provides remote diagnostic access.

A high level block diagram of the IQ-Link board is shown in Fig. 2.2. Two sets of tags are provided close to the local SMP bus to participate in the local snooping protocol. The bus-side Local Directory contains two bits of state information for each block in the Local Quad Memory. These Directory State bits indicate whether each line is: HOME, meaning no other Remote Cache contains a copy; FRESH, meaning other Remote Caches contain a read-only copy; or GONE, meaning that other Remote Caches contain a writable copy and no valid copy is on the local Quad. They are implemented in SRAM so that snoops can be performed at bus speed. This bus-side Local Directory does not contain any Quad ID information, just state.

The bus-side Remote Tags provide snooping information for the lines in the Remote Cache, indicating whether a given bus request to remote memory can be satisfied locally from the cache. These are also implemented in SRAM and contain only state information. Bus accesses that hit in the Remote Cache can be satisfied with a latency similar to

that of the Local Quad Memory latency. Accesses to remote memories that miss the Remote Cache are passed to the Directory Controller.

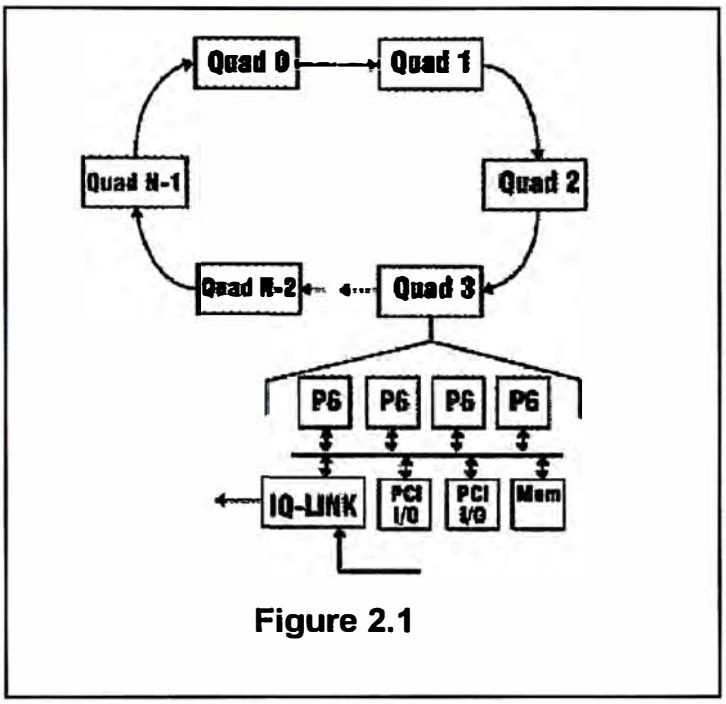


Figure 2.1

The Directory Controller maintains the network-side Local Quad Memory directory and the network-side Remote Tags used in the directory-based protocol. These two sets of tags parallel the bus-side Local Directory and the bus-side Remote Tags. Each block in the Local Quad Memory is represented in the network-side Local Directory by a 2 bit state field and a 6 bit list head pointer. For each block in the Remote Cache, the Directory Controller maintains a 7 bit state field, a 6 bit forward pointer, a 6 bit backward pointer and a 13 bit address tag. All of the network-side directory and tag information is maintained in SDRAM storage.

The Interconnect Controller provides the link and packet level interface to an SCI ring. Its role is to put SCI packets received from the Directory Controller on the ring, provide a bypass path for SCI packets targeted to

other Quads on the ring, and to strip those SCI packets from the ring that are targeted to its corresponding Directory Controller.

3. NUMA-Q Implementation

3.1 Quad

The initial implementation of the NUMA-Q Architecture uses Quads based on the Pentium® Pro microprocessor. Each Quad supports 4 microprocessors, up to 4 Gbytes of memory and 2 PCI busses. The local Quad bus is a 534 Mbytes/sec split transaction in-order bus. It provides limited facilities for the out-of-order responses required in a hierarchical system. The memory interface is implemented using the Intel Orion Memory Controller (OMC), and the PCI interface utilizes the Intel Orion PCI Bridge (OPB). The PCI bus is 32 bits wide, and has a maximum bandwidth rating of 133 Mbytes/sec.

3.2 Bridge

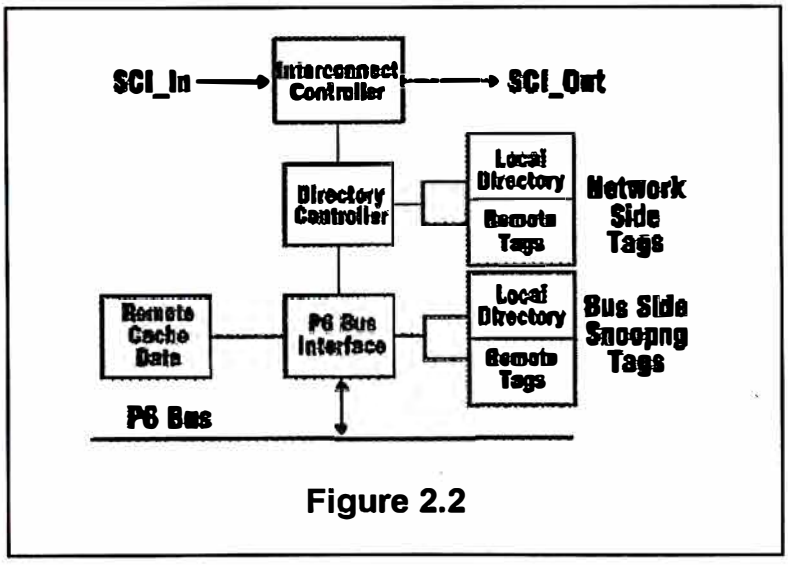


Figure 2.2

The IQ-Link Board provides the bridging function to the SCI interconnect. As shown in Fig. 3.1, it contains four major functional blocks: the Orion Bus Interface Controller (OBIC), the SCI Link Interface Controller (SCLIC), the DataPump®, and the RAM arrays.

3.2.1 Orion Bus Interface Controller (OBIC)

The OBIC is a 0.5 μ CMOS ASIC manufactured by LSI Logic. It contains 170,000 gates of logic and 8700 bits of embedded SRAM in a 550 pin TBGA package. It provides the interface to the PENTIUM PRO bus and manages the Remote Cache data arrays and bus snooping logic. The OBIC can handle up to 4 outstanding remote accesses and 2 incoming remote requests.

In the prototype implementation, the Remote Cache is 32 Mbytes and is organized as four-way set associative with 64 byte lines. As explained in Sec. 4.1 below, we believe that this size and organization is sufficient to virtually eliminate conflict and capacity misses for the database workloads we have studied.

However, it is possible that our analysis based on address traces collected on our current systems could lead us to incorrectly predict the size of Remote Cache needed to prevent capacity and conflict misses. We are not concerned in the case of overestimating the cache size requirement, as the relative cost of the RAMs is small when compared to the other components in a typical system configuration. In case we have underestimated the cache size requirement, we will use denser RAMs which enables us to provide 128 Mbytes of Remote Cache. We believe that the factor of 4 increase in cache capacity is sufficient to account for any errors in our analysis. We will perform benchmark NUMA-Q on prototype systems in order to determine the best Remote Cache size for the final product.

3.2.2 SCI Cache Link Interface Controller (SCLIC)

The SCLIC ASIC contains 140,000 gates of logic and 152,000 bits of embedded SRAM also in a 550 pin TBGA package. (See Fig. 3.2). Its primary function is to manage the directory-based cache coherence protocol, using one or more programmable protocol engines. We chose to implement a programmable protocol engine for several reasons.

- With the complexity of the SCI protocol we knew we couldn't generate a correct implementation in hardware on the first attempt.

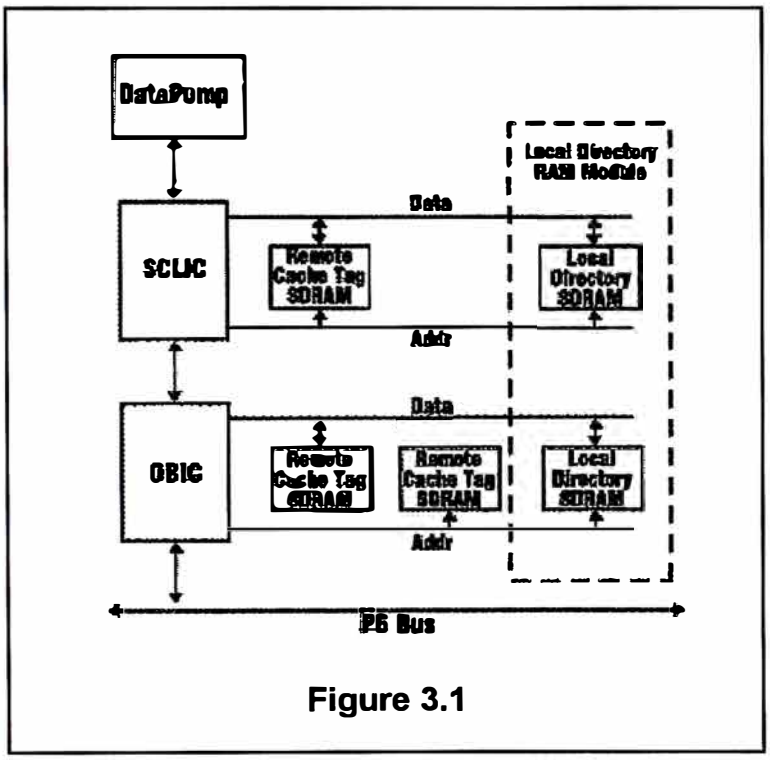


Figure 3.1

- A study by the Stanford FLASH team indicated that a programmable engine is not necessarily significantly slower than a hardwired implementation[14].

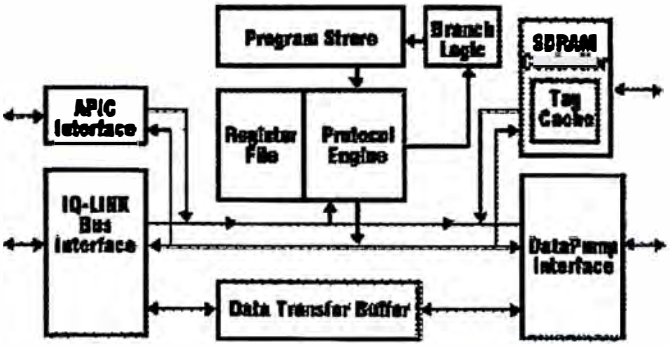


Figure 3.2

- A programmable engine would give us the flexibility to explore more efficient protocols after a correct SCI implementation was complete.

In the prototype implementation the protocol engine is a simple three stage pipelined processor with special logic for handling bit field operations. The Program Store RAM holds 16 KBytes of 64 bit instruction words. Each instruction provides a bit field extract, an ALU operation and a conditional branch. Conditional branches are executed in one cycle with no delay slot. Computed branches, used for decoding commands, require a branch delay slot. It is designed to support up to 12 independent tasks with fast hardware task switch. The register file contains global registers accessible to every task as well as task local registers accessible only from within a given task.

The Data Transfer Buffer holds the data portion of all packets entering or leaving the SCLIC. This allows the protocol engine to process only the header portion of packets. It can hold up to 8 data payloads of 64 bytes each.

The Tag Cache provides a small fully associative cache of the tag store for transactions in progress. When a task first reads the network-side cache tag or directory entry for the block it is working on the tag data is installed in the Tag Cache. Subsequent reads and writes to the tag data by the same or another task can occur within the Tag Cache without incurring the external SDRAM latency. When a task has completed an operation and is done modifying the cached information it can force a writeback operation to update the external SDRAM copy.

The PENTIUM PRO processors and OPBs exchange interrupt messages via the Advanced Programmable Interrupt Controller (APIC) bus. The APIC interface in the SCLIC provides an interrupt bridging function between Quads. It accepts interrupt messages from the local serial interrupt bus and provides them to the protocol engine so they can be sent as messages to remote Quads. It also accepts these incoming messages and transmits them on the local APIC bus.

3.2.3 DataPump®

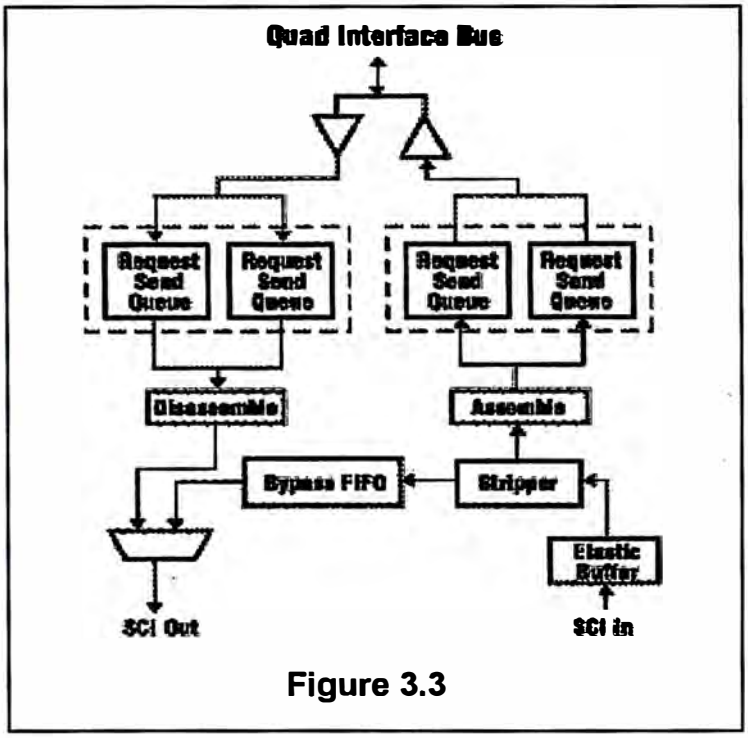


Figure 3.3

The DataPump is a GaAs chip developed by Vitesse Semiconductor Corporation. It provides the link and packet level protocol for an SCI network. Figure 3.3 shows a block diagram of the DataPump. The Elastic Buffer handles small frequency variations between the sender and receiver by deleting or inserting idle symbols between packets. The Bypass FIFO buffers incoming packets that arrive while a new packet is being inserted on the output link. The Stripper removes incoming packets whose destination ID matches this Quad ID and places them in one of the Receive Queues. An echo is placed on the output link in place of the removed packet. The Send Queues hold request and response packets while they are waiting to be inserted on the output link. The packets remain in the Send Queues until a positive echo is returned from the destination. If the destination does not have queue space to accept a packet, a negative echo is returned and the packet is re-transmitted using SCI's retry protocol. The initial implementation of the DataPump provides 4 entries in each Receive Queue, and 2 entries in each Send Queue.

The DataPump directly drives the 1 GByte/sec, 18 bit wide SCI link. Each 18 bit portion of an SCI packet is referred to as a symbol, and contains 16 bits of data, a flag bit and a clock. Using a 64 byte cache line size, SCI packets can be 8, 16, or 40 symbols long. At 500 MHz, the DataPump can route the first symbol of a packet to either the Assemble block or through the Bypass FIFO in 16 ns, and each additional symbol of the packet in 2 ns.

The OBIC, SCLIC and DataPump all provide performance counters to allow non-intrusive measurement of various statistics related to memory and bus utilization, protocol processing engine utilization, and cache miss rates.

3.2.4 Examples of Basic Operations

This section provides a brief description of how of a simple remote read cache miss is performed.

Every time a device on the PENTIUM PRO bus emits a request the OBIC executes a snoop cycle. If the address is within the range defined to be Local Quad Memory, the Local Directory is consulted. If the address is within memory space, but outside the Local Quad Memory space the Remote Cache Tags are consulted. In the case of a miss in the remote cache, the OBIC indicates on the bus that the reference will not be completed immediately. This allows subsequent independent requests on the bus to complete without waiting. The OBIC allocates a line in the Remote Cache and passes the request to the SCLIC. Further requests to the same line by other devices are backed-off until the first request is satisfied.

The request arriving at the SCLIC causes a task to be spawned in the protocol engine. For some requests a lookup is done to determine the current state of the requested block. For a simple read miss no such lookup is required. The Protocol Engine constructs a packet containing a CACHE_FRESH command, the address of the line, and the ID of the line's Home Quad. The Home Quad is determined by reading an internal table, indexed by high order address bits. The packet is then passed to the DataPump. The Protocol Engine then allocates a line in the internal Tag Cache, sets the state to Pending, and goes to sleep. While the task is asleep other tasks may execute to deal with other transactions.

When the response arrives from the Home Quad the task is awakened. The response packetÕs transaction ID determines which task to execute. The nature of the response determines what action is required. If a second request is required, the process is repeated.

When a response containing the data for the line arrives the data is passed to the OBIC, the cache state is updated, and the task completes.

When the OBIC receives the response data it will arbitrate for the bus and issue the response. As the response is placed on the bus the data is written into the remote cache and the cache state is updated accordingly.

3.3. Experiences with SCI

When we started the NUMA-Q project we decided to use an existing, or soon to exist, physical interconnect, rather than design our own packet routing chip because of resource limitations. The Vitesse DataPump chip provided the best combination of high-performance and commercial reality. The choice of this chip for the physical layer led us to use SCI directory cache protocols, even though we had started developing our own full-map protocol. Since the DataPump chip implements the appropriate portions of the SCI fairness and deadlock avoidance protocols we chose to leverage those by using the SCI cache protocols as well. Obviously, for the system sizes that we consider relevant, a full-map scheme is viable, and in fact would use less remote cache tag memory since no forward and back pointers would be required. After completing the design of the SCI protocol code we investigated the implementation of a full-map scheme and determined that the performance improvement would be less than 8% in our environment.

We chose to make several changes from the SCI protocol to simplify our implementation. We modified the packet format to send all command bits in the first header word and to send enough low order address bits to allow block data to be returned critical word first. We

store only 6 bit pointers which limits us to 64 Quads, or 256 processors. Given the sizes of systems sold in the commercial market, this is more than adequate.

The decision to use the DataPump, and thus SCI, for our interconnection scheme did not limit performance. Although other interconnection schemes may provide higher connectivity between Quads, there were no commercial options that matched the high bandwidth and low latency characteristics of the DataPump. As shown in Sec. 4.3 below, the contribution of the SCI ring to the average total remote access latency is always less than 25%, and the utilization of the SCI ring is always less than 30%. These performance characteristics met our requirements for a low-latency interconnect with sufficient bandwidth headroom for future growth.

One of the major shortcomings of the SCI protocol in this type of system is that a memory device cannot request that the head of a dirty list return a copy of a block. If a processor on the Home Quad requests a copy of the block, the protocol requires that the Home Quad join the sharing list to get a copy. Since the local directory state entry does not contain sufficient state storage to join a sharing list, additional storage is required for this function. One solution to this problem is to add the sufficient bits to every local directory entry so that it can participate in a sharing list. This approach requires 19 bits more than the 8 already required and was unacceptable. We maintain a pool of additional bits that are temporarily allocated when needed.

4. Performance

The NUMA-Q system is designed for the large-scale commercial computing marketplace. This market is dominated by On-Line Transaction Processing (OLTP) and Decision Support Systems (DSS) applications. In order to

predict performance for OLTP and DSS usage models, benchmark programs that represent each of these application areas were analyzed on current systems, and their behavior was characterized in the form of an "application profile" which was used to drive a simulation model.

The OLTP workloads used in construction of the application profiles included the Transaction Processing Performance Council (TPC) A, B, and C benchmarks [15]. The DSS workloads considered included parts of the TPC-D benchmark, as well as a variant of the Wisconsin Benchmark [16]. In this paper, we will report data for the TPC-B benchmark and Query 6 of the TPC-D benchmark. Although there are some fundamental differences between the TPC-B and TPC-C benchmark, the cache miss profiles are similar, and thus the TPC-B is representative of OLTP workloads in general. All DSS workloads studied were done in the context of parallel query decomposition, where multiple processors cooperate on the execution of a single query. While I/O bandwidth requirements for all the queries studied varied widely, their cache miss profiles were again very similar. We have selected TPC-D Query 6 as a representative query with an average I/O bandwidth requirement.

All of these programs were run on Sequent Symmetry 2000 and 5000 SMP systems [17,18], and detailed performance data was collected using both the performance counters embedded in the cache and bus control ASICs as well as logic analyzers connected to the system bus. Data was also collected via event counters in the operating system kernel. For each benchmark, data was collected during a "dominant phase" of execution where consistent steady-state behavior was observed. The execution phase is called "dominant" as it represents at least 80% of the total running time of the benchmark. In all cases, data collection was either non-intrusive or had no noticeable impact on the application.

4.1 Workload Profiles

The application profiles determined from system measurements establish rates for the occurrence of different events. Three of these rates are established on a per instruction basis: second-level cache misses, cache line invalidation requests, and I/O operations. Other events are established on a per cache miss or per I/O operation basis. For example, for each cache miss, there is a probability that the requested cache line can be found clean in Local Quad Memory, and that the processor is requesting that line so that it may modify it (i.e. the processor issues a write that misses the cache). Other rates that have been established in the profile include requested cache lines that are dirty in another cache, Remote Cache hits, the average number of copies of a cache line when an invalidation request is generated, etc.

The most important events for all the workloads are those which require a Quad-to-Quad transfer. To establish this rate, we started with a minimum Remote Cache/Local Quad Memory miss rate based on the measured second-level cache-to-cache transfer (or communication miss) rates. If a processor references a cache line that is modified in another second-level cache, it will miss the Remote Cache (or Local Quad Memory) as well unless that second processor is located on the same Quad. These communication misses establish a minimum miss rate for the Remote Cache and Local Quad Memory. The next task is to determine the additional misses that are compulsory as well as those caused by conflict and cache capacity.

In order to determine the additional misses caused by the other three effects listed above, we used our multiprocessor cache simulator on the address traces we collected to determine the overall miss rate for a 32 Mbyte cache with 64 byte lines. Negative cold start effects were largely offset by the trace collection technique, where traces are composed of misses from 1 Mbyte direct-mapped caches. Because the traces contain

a large number of communication misses, the simulator reported a rate very close to the minimum miss rate for very large caches. This led us to believe that capacity and conflict misses at the 32 Mbyte Remote Cache would be far outnumbered by communication misses. Further, examining the I/O rates compared to the instruction count and miss rates, it became clear that compulsory cache misses due to I/O would also be very low. These factors combined to lead us to use the minimum miss rate for the Remote Cache and Local Quad Memory based on the second-level cache-to-cache transfer rate.

Because the vast majority of Remote Cache misses (or all Remote Cache Misses in our simulations) are communication misses, the likelihood of finding this cache line at its home Quad in a large system is low, as it is equally likely to reside in the second-level cache of any other processor in the system. These Remote Cache misses cause "4 hop1" cache line fills to occur, where an SCI memory read message is first sent to the home Quad, followed by a response that indicates that a new cache read message should be sent to a third Quad. We assume that these cache lines that cause Remote Cache misses and Local Quad Memory misses are locks and DBMS kernel data structures which are constantly modified. Even though these cache lines are likely to all be mapped to memory that resides in a single Quad in a real system, we assume they are randomly distributed across all Quads. Even with this assumption, we still get the same average number of Remote Cache misses versus Local Quad Memory misses per Quad. This ratio is important, as we assume that Local Quad Memory misses are satisfied with a "2 hop" request.

Event	Rate	TPC-B	TPC-D Q6
L2 Cache Miss	per instruction	0.0223	0.0018
Hit to Local Quad Memory	per L2 Miss	50.4%	51.8%
Hit to Remote Cache	per L2 Miss	27.2%	27.8%
Local Cache-to-Cache Transfer	per L2 Miss	1.3%	1.6%
2 Hop Remote	per L2 Miss	3.1%	3.5%
4 Hop Remote	per L2 Miss	9.1%	10.9%
Local Hit/Remote Invalidate	per L2 Miss	6.8%	4.5%
Sharing List Length	per Write to Shared Line (in processors)	1.2	1.6
Invalidate Line	per instructions	0.00209	0.00018
Remote Invalidate	per invalidate	95%	108%

Table 4.1

Table 4.1 shows the number of cache misses per instruction for 512K 4-way set associative second-level caches and the location of the cache line requested for both of the workloads discussed in this paper. The table also shows the rate of line invalidation requests, and the ratio of remote invalidation messages to line invalidation requests. The rate of remote invalidation requests is derived from the sharing list length, which is the average number of processors possessing a shared block when an invalidation targets that block. Another factor determining the rate of remote invalidation is the rate of Read-with-Invalidate (or write miss) references to cache lines, and whether they are clean or modified in other caches. Write misses to lines cached locally on a Quad but requiring a remote invalidate message are shown in the "Local Hit/Remote Inval" line of the table. All of these rates were determined using performance counter data, address traces, and the communication miss analysis described above. The data in Table 4.1 is for 32 processor systems. Smaller systems will have fewer 4 hop remote

transfers as well as more local cache-to-cache transfers and 2 hop remote transfers as more communication misses are satisfied locally or at the home Quad.

Table 4.1 highlights some of the differences and similarities between the two workloads. While the cache miss rate for TPC-D Query 6 is much lower than that of the TPC-B, the distribution of locations for each requested cache line is similar. The TPC-B workload must process an order of magnitude more cache misses as well as an order of magnitude more invalidation messages per instruction. This reduces instruction throughput (MIPS) for the TPC-B workload, and reduces its scalability as well. The two workloads also differ greatly in the amount of I/O operations performed and I/O bandwidth consumed. This is not discussed here, as it is not responsible for the performance difference between the two workloads.

4.2 The Simulation Model

The idea behind the simulation model is simple. Instructions are generated for each processor at a rate equivalent to their core CPI, which is the number of clocks per instruction in a system with no second-level cache misses. Based on the probabilities in the workload profile, each instruction may generate a cache miss, an invalidation, and/or an I/O event. In the first two cases, these events may cause the processor to stall, which in the case of our simulation simply means that no more instructions can be generated until the event that caused the stall is complete2. As the simulator models in detail all the datapaths in the system and accounts for all contention, the latency for completing all events is determined accurately. The impact of this latency on performance is determined by measuring the instruction throughput rate, or MIPS rate, which is degraded from its potential maximum by cache misses and invalidation requests which cause processor stalls. The I/O

events are included simply to consume bandwidth and add contention to the model. Thus, there is an implicit assumption that the workloads are not I/O bound. This is indeed the case, and our simulation results are checked to ensure that the I/O rates keep pace with the MIPS rates.

The simulation model incorporates what are essentially behavioral models of all ASICs in the system. This models the interaction of the PENTIUM PRO bus, PCI bus bridge, memory controller, and components on the IQ-Link board on a cycle-by-cycle basis. All buffer sizes and protocols are modeled exactly, except for the mechanisms to ensure fairness on the SCI ring. This is not an important omission, as the SCI ring is lightly loaded in this system. The timing for data transfers between internal buffers in each ASIC as well as logic delays were determined through analysis of functional specifications, interviews with design engineers, and analysis of the results of functional simulation. This approach has provided a very detailed and accurate simulation model without the slowdown associated with true functional simulation.

An alternative to our simulation approach using workload profiles would be to use instruction or address traces to determine specifically which instructions cause misses and what specific cache lines are addressed. However, this approach is more difficult to code and more time consuming to execute. And, the added precision in modeling the occurrence of different events does not necessarily imply better accuracy in predicting performance. Our approach has been validated using our current Symmetry systems as a comparison to simulation models. The steady-state behavior of the workloads is consistent enough and dominant so our approach based on probabilities produces an accurate result.

A better approach to either trace-based simulation or our workload profile approach would be to use execution-based simulation. In this case, not only are the instructions which cause cache misses and the addresses they refer to represented, but also the interaction between the multiple processes is represented in the way it causes different code paths to be executed. For example, our approach cannot reveal a lock contention bottleneck as an increasing transaction rate is achieved. Such software bottlenecks can only be exposed by running the actual code. In these situations, the workload profile for the steady-state dominant phase for one performance level becomes invalid for another performance level. While we would prefer to uncover these potential bottlenecks in simulation, the difficulty of supporting execution-based simulation where a multiprocessor operating system must be booted and multiple Unix processes must be run with full IPC and I/O capabilities was too great for our schedule. The research community is making progress on the problem of execution-based simulation for multiprocessor OS's and DBMSs, and we plan to leverage that work in the future [20].

4.3 Results

In this section, we discuss the simulation results for an implementation with 133 MHz PENTIUM PRO processors each with 512K 4-way set associative caches, 66 MHz PENTIUM PRO bus and IQ-Link board components, a 32MB 4-way set associative Remote Cache, and a DataPump chip with an internal clock rate of 500 MHz3. Overall, the NUMA-Q system provides balanced performance in that there are no bandwidth limitations for any datapath in the system. As always, processor performance is limited by the latency to satisfy cache misses, but this latency is not inflated due to contention at a saturated datapath. However, a single protocol processing engine for the SCLIC does become saturated for the TPC-B workload, and the contention at this

resource does add to latency. The utilization of this resource and the techniques available to improve its performance are discussed after the bandwidth utilization and observed latency is addressed.

The utilization of the PENTIUM PRO data bus in each Quad in a NUMA-Q system is modest for both workloads, never exceeding 40% or about 170 Mbytes/s. The utilization of the SCI ring is even lighter, never exceeding 30%. The utilization of the datapaths between the ASICs on the IQ-Link board itself is also very modest. Of course, there are some interesting variations in the amount of bandwidth consumed on each of these datapaths depending on the number of Quads in the system. These variations are caused by interactions between an increased bandwidth requirement to accommodate data bus transfers on behalf of remote accesses and a decreased bandwidth requirement caused by the throttling effect of increased latency as the number of remote transfers increases. However, in all cases, none of the datapaths is saturated.

Given that the PENTIUM PRO bus and SCI ring are not saturated for either workload, we turn our attention to what is limiting instruction throughput for the system: latency. For each workload, we consider the latency for two cases which are essentially upper and lower bounds. The first, labeled "Baseline," shows the latency for an implementation where the SCLIC has a single directory protocol engine. The second case, labeled "Ideal," assumes a protocol engine where only 1 instruction (and thus 1 clock cycle) is required to process each 64 bit header word in an SCI packet. The motivation for this ideal model is that 1 clock cycle per header word is required as a minimum to copy this information through the engine, which must be done to enable protocol processing. The result is two to four clocks of engine processing per SCI packet, plus any delay to wait for the protocol engine if it is already

in use. The Data Transfer Buffer ensures that any data that must be copied to the OBIC or DataPump is ready after protocol processing in the ideal engine.

The ideal protocol processing engine should also pay a penalty for tag accesses, but we have eliminated that from the model. Tag access latency becomes the bottleneck in this case, as more clocks are required to read and write the tags than to copy the SCI headers. Our implementation uses SDRAM tags, as we can hide the access latency behind protocol processing. The ideal machine can not mask this latency, so we truly make it an ideal by giving it zero latency tags. This is not unreasonable, as a very large tag cache with a 1 cycle access time could show this behavior. The performance of the ideal model establishes a lower bound on protocol processor utilization, or occupancy, and thus latency for each workload.

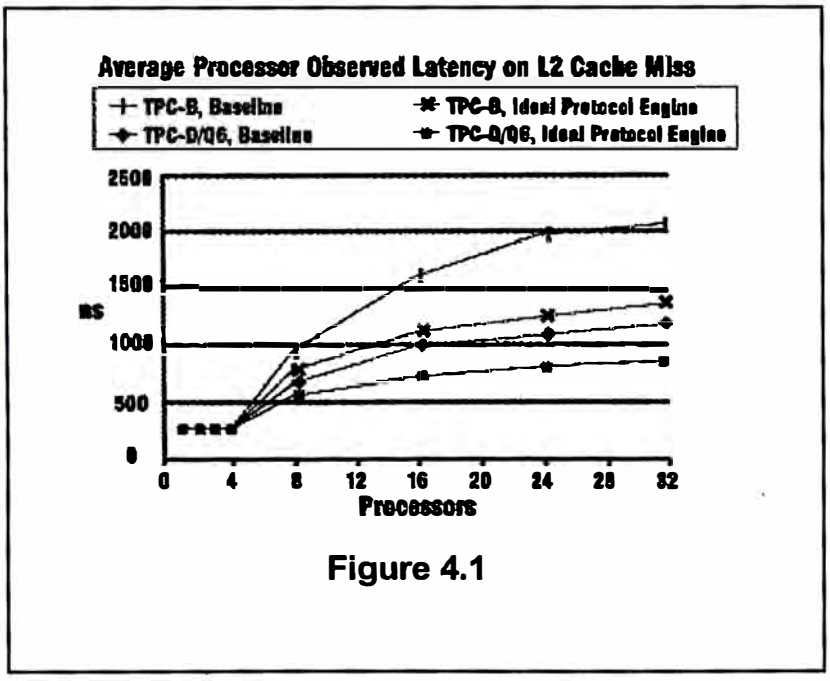


Figure 4.1

Figure 4.1 shows how the average latency to satisfy a second-level cache miss increases with system size as Quad-to-Quad transfers are

introduced. The TPC-D Query 6 workload has a much lower cache miss latency due to its lower cache miss rate and correspondingly lower Quad-to-Quad bandwidth requirement for both data and invalidations.

Although the cache miss service distributions are much different, the average L2 miss latencies we report are similar to those shown for the Stanford FLASH multiprocessor with 1 MB caches [14]. Even though the remote reference latencies for NUMA-Q are higher than reported for FLASH, the large Remote Cache reduces the average L2 miss service time for NUMA-Q. Our system also benefits from a faster link-level interconnect provided by the DataPump. NUMA-Q does not have lower remote latencies than FLASH due to the increased protocol processing burden placed on the SCLIC by four processors (instead of one) combined with its lower clock rate (66MHz versus 100MHz). Overall, it is interesting to note that a NUMA system with a large, fast Remote Cache, slower protocol processing, and a fast second-level interconnect can provide similar latencies to a system with no Remote Cache, faster protocol processing, and a slower second-level interconnect. This result, combined with the cost and packaging advantages of four processor Quads and off-the-shelf ASICs, further justifies our decision to use Quad processor building blocks in the NUMA-Q architecture.

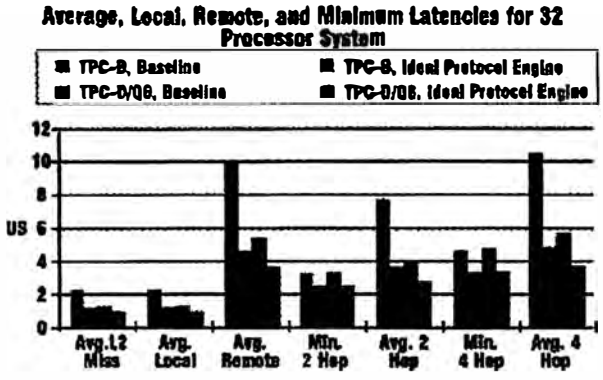


Figure 4.2

As stated above, NUMA-Q combines faster local access with slower remote transfers to determine average cache miss latency. While the average local latency can be seen to be about 200 to 300 ns in Fig. 4.1, the remote access latency is shown in Fig. 4.2. The figure shows that, for the TPC-B workload and Baseline implementation, the observed latencies for remote operations is much higher than that for the TPC-D Query 6 and the minimum values. This increase in latency is due to the higher protocol processing overhead required by the TPC-B workload. This latency is dramatically reduced in the Ideal case, as protocol processing overhead is reduced to a minimum.

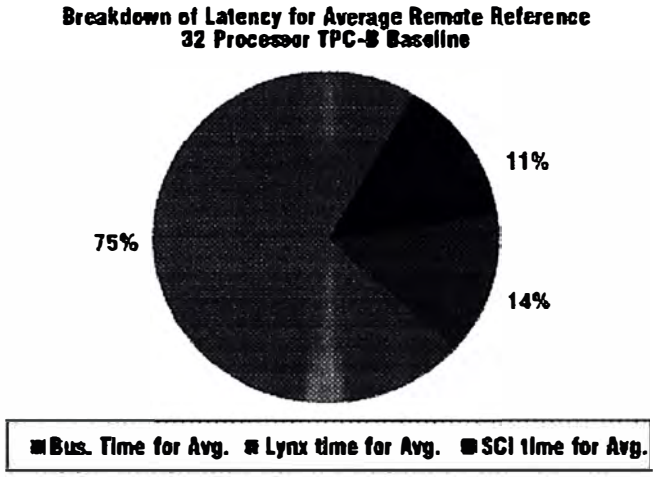


Figure 4.3

Figure 4.3 shows the breakdown of latency into different components for an average remote reference for a 32 processor TPC-B and Baseline implementation. As the figure clearly shows, protocol processing consumes the most time, followed by the time spent arbitrating for and transferring data across the PENTIUM PRO bus. Each of these categories is greater than the time spent sending and receiving SCI packets, which is done very quickly in the DataPump. This breakdown of

remote latency bodes well for the feasibility of building larger systems, as more SCI link-level components can be added to increase the number of Quads in the system with only a minimal impact on remote access latency.

Figure 4.4 shows how the relative latency breakdown for the 32 processor TPC-B changes with the Ideal implementation. The breakdown shows a shift in overhead from IQ-Link Board time to Bus and SCI time, but the IQ-Link Board component still accounts for almost one-half of the total latency. This is the result of the inherently high Quad-to-Quad communication requirement for the TPC-B workload.

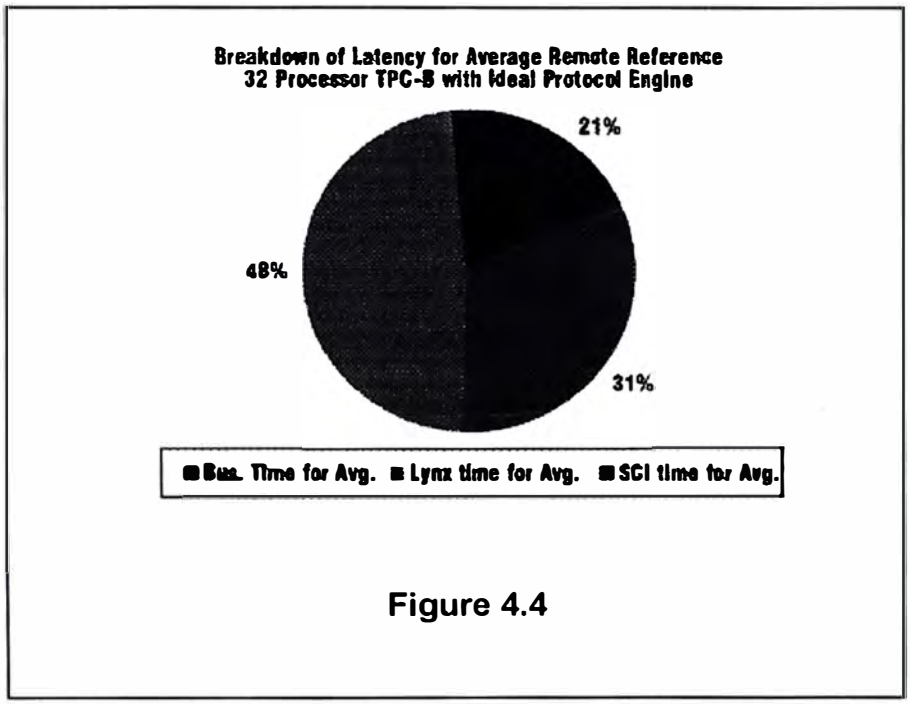


Figure 4.4

In order to decrease latency and increase performance for the TPC-B workload with the Baseline implementation, the occupancy of the protocol processing engine must be reduced. This conclusion is consistent with other studies [21]. We have identified several techniques for reducing this latency including: optimized microcode for the protocol processor; a new coherence protocol which enables

reduced instruction count microcode; hardware assist, where frequent instruction sequences are replaced by logic in the SCLIC, and finally dual sequencer engines, where two distinct protocol processing engines reside in the SCLIC similar to the S3.mp [22].

Results for a full-map coherence protocol were mentioned briefly in Sec. 3.3 above. We have also completed some simulations for the dual sequencer engine approach. The impact on latency is very dramatic for TPC-B, with average remote access latency being reduced by 30% and overall L2 miss latency being reduced by 25%. The latency reductions for the TPC-D Query 6 workload are 8% and 7% respectively. The modest reduction in latency for Query 6 reflects the fact that the performance of the Baseline implementation is close to the Ideal case for this workload.

Figure 4.5 shows the increase in MIPS throughput for both workloads and both SCLIC implementations as more processors are added to the system. The y-axis is stated as "Effective Processors," with each effective processor being equivalent to the MIPS rate achievable with a single processor. Figure 4.5 is essentially a plot of the scaling efficiency as processors are added to the system. Clearly, the TPC-B workload is much more sensitive to cache miss latency, as the Ideal SCLIC provides a 43% performance boost for 32 processor systems, as opposed to a 5% boost for TPC-D Query 6. A dual engine SCLIC recovers 46% of the 32 processor TPC-B performance gap between the Baseline and Ideal cases, boosting performance by 20%. Dual engines also enable the performance of the TPC-D Query 6 to achieve 97% of the Ideal SCLIC case.

Sequent's current bus-based systems achieve better efficiency in scaling the TPC-B workload, but this is caused by a low level of MIPS

performance in a single processor configuration. For Symmetry and other large bus-based systems, the ability to connect large numbers of processors to a single bus results in high memory access latency, and thus a performance burden for small configurations. For NUMA-Q, this performance burden is removed, as all memory references are local in systems with up to 4 processors. This improves low-end performance, but makes the task of scaling up performance with perfect efficiency much more difficult.

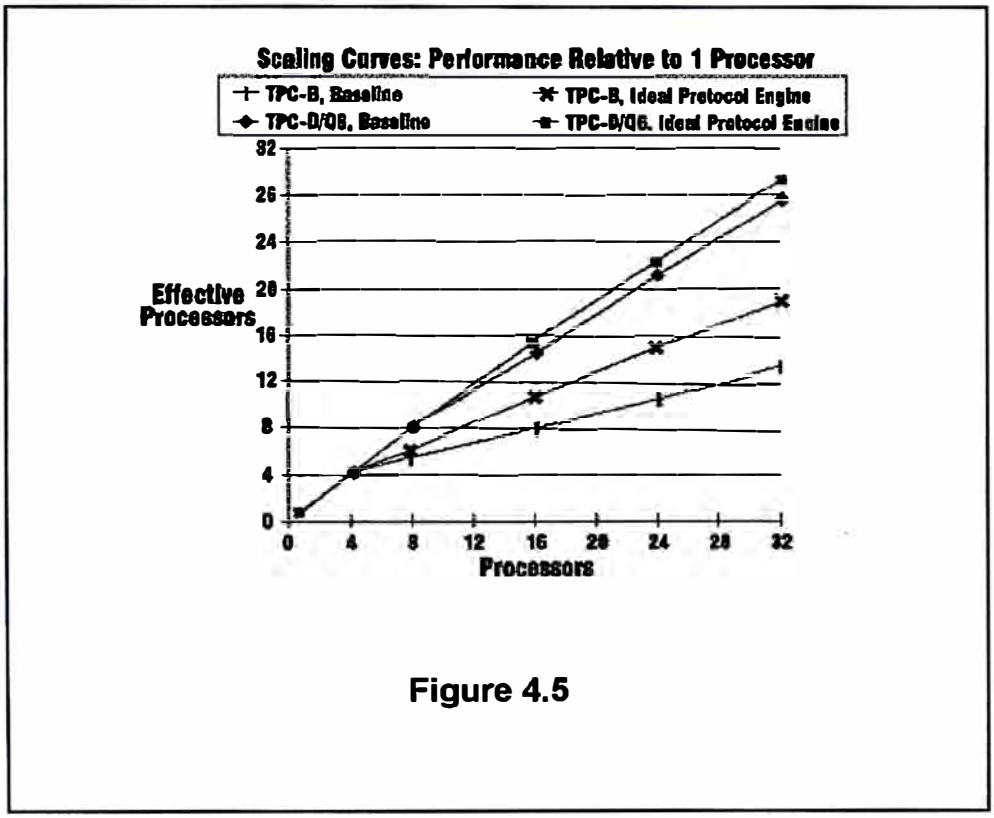


Figure 4.5

As Fig. 4.5 shows, the efficiency with which a workload can be scaled on the NUMA-Q system will depend on its cache miss rate and service profile. As development of new commercial DBMS releases continues, optimizations are routinely made to reduce software bottlenecks and eliminate unnecessary cache line sharing. These changes improve performance on shared-bus systems, but their impact may be more dramatic for CC-NUMA systems like NUMA-Q. Reducing the rate of sharing of modified cache lines will reduce the Quad-to-Quad transfer

requirement, which can reduce cache miss latency dramatically. Changes in the workload profile can improve performance to the point where it exceeds the results presented here for the "Ideal" case.

5. Status and Conclusion

NUMA-Q is a CC-NUMA system designed for the commercial marketplace. Because the architecture leverages SMP building blocks, it has enabled the use of commodity components. This has allowed us to keep costs under control, and focus on the issues of the second-level interconnect.

Performance analysis using simulation models and workload profiles from commercial database benchmarks has been used to assess hardware design tradeoffs. The results presented here demonstrate that the NUMA-Q hardware is capable of supporting high scalability of DBMS software.

NUMA-Q hardware is currently running in the lab. During 1996 we will run benchmark workloads to validate our simulation results. We expect customer shipments of systems to occur in 4Q96.

Acknowledgments

The authors would like to thank the other members of the design team including Bruce Gilbert, Rob Joersz, and Bob Safranek, for their contributions to the design and for providing some of the detailed information necessary to construct an accurate simulator. We would also like to thank Ruth Forester, Mary Meredith, Doug Delany, and the other members of the benchmark teams who were instrumental in obtaining performance data on Symmetry systems. In addition, we would like to acknowledge the insightful comments of Don DeSota, Ken Dove, Paul McKenney, and Roger Shelton, all of whom read preliminary versions of this paper. We would also like to thank our management team and marketing

colleagues, who agreed to let us publish this work. Finally, we would like to thank the anonymous reviewers for their feedback on the original version of this paper.

[1] Here we adopt the terminology of previous papers on CC-NUMA systems where a "hop" refers to a transmission of a single message from a source quad to its target destination quad. In our SCI ring configuration, each packet in a single hop transfer may pass through one or more DataPump chips in intermediate quads.

[2] Not all cache misses or invalidations will cause the processor to stall. The "dynamic execution" techniques of the Pentium Pro processor will prevent stalling in some cases depending on the instruction mix of the workload [19].

[3] Other speed and cache size combinations are possible.

BIBLIOGRAFIA

[1] IEEE Computer Society. IEEE Standard for Scalable Coherent Interface (SCI), IEEE Std 1596-1992, New York, New York, August, 1993.

[2] Intel Corporation. Intel Paragon ® Supercomputer Product Brochure. http://www.ssd.intel.com/paragon.html

[3] Pyramid Technology. Reliant ®1000. http://ra.pyramid.com/products/1.1.1.1.html

[4] A. W. Wilson, Jr. Hierarchical cache/bus architecture for shared memory multiprocessors. In Proceedings of the 14th International Symposium on Computer Architecture, pages 244-253, June 1987.

[5] E. Hagersten, A. Landin, and S. Haridi. DDM - A cache-only memory architecture. IEEE Computer, pages 44-54, vol. 25, no. 9, September 1992.

[6] S. Frank, H. Burkhardt III, and J. Rothnie. The KSR1: Bridging the gap between shared memory and MPPs. In Proceedings of the 38th IEEE Computer Society International Conference (Spring Compcon), pages 285 - 294, February 1993.

[7] A. Saulsbury and A. Nowatzyk. Implementing simple COMA on S3-MP. Presentation at The Fifth Workshop on Scalable Shared Memory Multiprocessors, Santa Margherita Ligure, Italy, June 1995. http://playground.sun.com:80/pub/S3.mp/simple-coma/isca-95/present.html

[8] B. Gallagher and M. Jonikas. SQL Server 6.0: Tough to top. PCWeek, page 83, vol. 12, no. 36, September 11, 1995.

[9] J. Chapin, S. A. Herrod, M. Rosenblum, and A. Gupta. Memory system performance of UNIX on CC-NUMA multiprocessors. In Proceedings of

the Joint International Conference on Measurement and Modeling of Computer Systems, pages 1-13, May 1995.

[10] P. Stenström, T. Joe, and A. Gupta. Comparative performance evaluation of cache-coherent NUMA and COMA architectures. In Proceedings of the 19th International Symposium on Computer Architecture, pages 80-91, May 1992.

[11] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, and J. Hennessy. The DASH prototype: Logic overhead and performance. IEEE Transactions of Parallel and Distributed Systems, pages 41-61, vol. 4, no. 1, January 1993.

[12] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy. The Stanford FLASH multiprocessor. In Proceedings of the 21st International Symposium on Computer Architecture, pages 302-313, April 1994.

[13] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B.-H. Lim, K. Mackenzie, and D. Yeung. The MIT Alewife machine: Architecture and performance. In Proceedings of the 22nd International Symposium on Computer Architecture, pages 2-13, June 1995.

[14] M. Heinrich, J. Kuskin, D. Ofelt, J. Heinlein, J. Baxter, J. P. Singh, R. Simoni, K. Gharachorloo, D. Nakahira, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy. The performance impact of flexibility in the Stanford FLASH multiprocessor. In Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI), pages 274-285, October 1994.

[15] Transaction Processing Performance Council. TPC Benchmark Specifications.
http://www.tpc.org/bench.descrip.html

[16] J. Gray, Editor. The Benchmark Handbook for Database and Transaction Processing Systems, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[17] T. Lovett and S. Thakkar. The Symmetry multiprocessor system. In Proceedings of the 1988 International Conference on Parallel Processing, pages 303-310, August 1988.

[18] Sequent Computer Systems, Inc. Symmetry 5000 Series. http://www.sequent.com/offerings/products/

[19] Intel Corporation. What is Dynamic Execution? http://www.intel.com/procs/P6 /dynexec.html

[20] M. Rosenblum, S. A. Herrod, E. Witchel, and A. Gupta. Complete Computer System Simulation: The SimOS Approach. IEEE Parallel and Distributed Technology, pages 34-43, vol. 3, no. 4, Winter 1995.

[21] C. Holt, M. Heinrich, J. P. Singh, E. Rothberg, and J. Hennessy. The performance effects of latency, occupancy and bandwidth in cache-coherent DSM multiprocessors. Presentation at The Fifth Workshop on Scalable Shared Memory Multiprocessors, Santa Margherita Ligure, Italy, June 1995.

[22] A. Nowatzyk, G. Aybay, M. Browne, E. Kelly, M. Parkin, B. Radke, and S. Vishin. The S3.mp scalable shared memory multiprocessor. In Proceedings of the 1995 International Conference on Parallel Processing, August 1995.

[23] Data General Corp.
The NUMA Architecture, 1998
Data General's NUMALiiNE Technology: The Foundation for the AV 20000 Server, 1998
http://www.dg.com/ numaliine/html/inside_the_new_computer_indust.html

[24] Sequent Computer Systems, Inc.
Implementation and Proformance of a CC-NUMA System, 1998
http://www.sequent.com/imp_wp1.html