Yet More On Tasters

This page contains supplementary material for part two of *Performance Assurance for IT Systems*. These observations are typically short, in-line pieces that are considered to be useful additions to the existing technology tasters. It is obviously assumed that you have access to a copy of the book to understand the context of the additional material.

Observations

The specific taster in the book to which an observation relates is given in the heading unless it is a more general topic.

Contents:

Intel / AMD – confusion between 32-bit, "partial-64 bit", and 64-bit
IP-Based Storage: iSCSI
IP-Based Storage: FCIP and iFCP
Infiniband-based SANs
Dual Core Processor Chips
IO Interconnect Technologies

Intel / AMD – Confusion between 32-bit, "partial 64-bit", and 64-bit systems (CPU and Operating System Basics Tasters)

Intel's approach was "relatively" straightforward: it had Xeon and Pentium 4 in the 32-bit arena; and Itanium in the 64-bit arena. AMD came along and confused matters by announcing AMD64 chips that will run both in both 32-bit and 64-bit modes. Naturally, the majority of the AMD "marketing speak" emphasises the 64-bit capability; I will call this mode "partial 64-bit" for the moment. Intel was forced to respond, bringing out EM64T, its own partial 64-bit offering. This is something that they had been working on in secret, frequently denying its existence. This is hardly surprising; as it was a defensive measure which they no doubt fervently hoped would never see the light of day, given the investment that has been put into Itanium.

The following paragraphs attempt to alleviate some of the confusion between the various offerings.

Existing Plain Vanilla 32-bit. Chips employ an IA-32-compatible architecture. While there are various IA-32 technical shortcomings in hindsight, the most obvious constraint to users is 32-bit memory addressing which limits the size of the system address space to 4GB. In Windows Operating Systems the default memory allocation is split: 2GB for user mode and 2GB for kernel mode. The user allocation can be increased to 3GB at the expense of a reduced kernel. This arrangement may be suitable for some applications, e.g. Microsoft Terminal Server that can consume significant memory when supporting large numbers of users, but not others. The fact that large amounts of physical memory can now be configured on relatively small servers, e.g. 12GB on a dual CPU system is feasible, can make the use of 32-bit a constraining factor.

Memory Extenders. Intel introduced Page Addressing Extensions (PAE) to its 32-bit chips in an attempt to alleviate the memory constraint. Data (and only data) can be stored between the 4GB line and 64GB line. However, it must be brought below the 4GB line before it can be accessed. This approach can be thought of as a three tier memory management system: memory (0-4GB); memory-based backing store (4-64GB); and disk-based backing store. PAE is useful for software with very large data caches, e.g. DBMSs.

Partial 64-bit. The first thing to say is that AMD64 and EM64T chips can operate in standard 32-bit mode (otherwise known as "legacy" mode) running on a 32-bit OS. Compatibility mode is next; it runs 32-bit applications on a system with a 64-bit OS and 64-bit drivers. The advantage of this mode is that each process has its own 4GB limit, as opposed to a system-wide 4GB limit. Finally, there is 64-bit mode (actually termed long mode by AMD and IA32e by Intel). 64-bit mode makes use of 64-bit registers and ALUs. With respect to memory addressing, it uses 40 bits, which can provide access to 1TB of memory. Apart from a 64-bit OS and drivers, this mode requires 64-bit applications that have been compiled to make use of the new features. Note that, despite the memory enhancements and selective use of 64-bit, Partial 64-bit is still essentially an IA32-compatible architecture.

64-bit (IA-64). This is pure 64-bit, engineered from the ground upwards. Intel started work in this area 11 years ago to produce servers that would be competitive with products from the main Unix vendors. It employs a totally new architecture, termed IA-64.

What does it all mean? In particular, is Partial 64-bit an alternative to IA-64? Here is my current twopenneth:

- Despite any technical attractions, Partial 64-bit is fundamentally a commercial fight between Intel and AMD. AMD64 has forced Intel to veer away from the grand march towards IA-64, if only temporarily, while it fights AMD on this middle ground. If Intel loses this fight then IA-64 may be in some jeopardy

- Partial 64-bit provides a transition path towards full 64-bit for those customers who require it

- Compatibility and 64-bit modes can undoubtedly help to relieve memory constraints for non-DB software that suffer from such problems

- Encryption algorithms will benefit from 64-bit registers and ALUs

- No benefits for floating point, as these registers are already 80 or 128 bits on existing 32-bit systems

- Although general performance improvements are claimed over 32-bit, I have not seen any figures yet to be able to comment. A number of techniques, mainly increases in clock speed, have allowed IA-32 architectures to keep abreast with the performance of IA-64 systems. This parity is helped by the fact that IA-64 is still relatively immature, and some of the promised benefits, e.g. parallel processing, have yet to be realised

- multi-processor performance (4 CPUs upwards) has always been something of a problem on 32-bit architectures, despite gradual improvements. Many of the issues relate to general multi-processor system design, chipsets and OSs rather than to basic CPU design. The handling of large amounts of physical memory may be a particular problem on Partial 64-bit multiprocessors, depending on the design. Certainly above 4 CPUs I feel more comfortable with IA-64 systems.

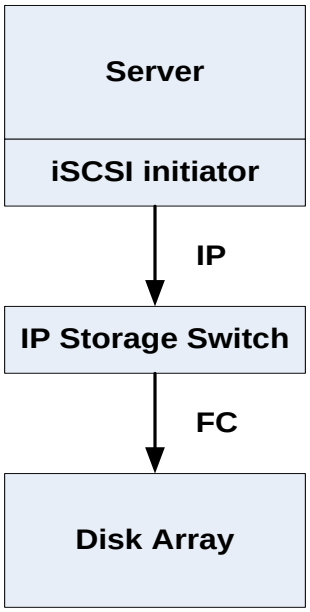
IP-Based Storage: iSCSI (Hard Disk: Fibre Channel, SAN and NAS taster)

Traditional disk subsystems have direct connections with the servers that access them (albeit sometimes via SAN fabric). The entire system from host to disk target is usually SCSI-based, or in more recent times Fibre Channel-based. While NAS subsequently provided a mechanism to connect an entire disk subsystem to a LAN, the disks themselves are still directly connected to the server(s) within the NAS itself. iSCSI now allows the connection between the server and the disk target to be done via the LAN. As shown in the diagram below the iSCSI protocol provides a mechanism to stretch the components of physical disk access over an IP network in a roughly similar fashion to the way that the software components are stretched in NFS or CIFS.

In iSCSI, the “client” is an iSCSI initiator, which is provided by the majority of operating system vendors. The iSCSI commands, encapsulated in TCP/IP, pass over an IP network to the disk target where the commands are processed. The “marketing speak” for IP SANs tends to concentrate on comparisons with Fibre Channel-based SANs. However, the waters can quickly become muddied:

- The disk target may be an array with network port(s) to support IP but the hardware may well be “wall-to-wall” fibre channel. Some traditional storage vendors who had been watching IP SAN from the sidelines have joined the fray by offering two versions of a given disk array, an iSCSI variant in addition to the original Fibre Channel version. There does not appear to be a great deal of difference between the two, apart from the host ports and the support for the iSCSI protocol.

- Another approach is to use IP storage switches. As indicated in the following diagram, they are modified LAN switches that act as bridges; IP through standard ethernet ports on one side and fibre channel on the other side



- It is easier for existing NAS vendors to support IP SANs. As their host ports are already network ports, they simply have to support the iSCSI protocol.

Key Performance observations include:

- As stated above, iSCSI uses TCP/IP. If a server generates significant disk traffic there may be a significant CPU overhead (remember from the Network Basics taster that TCP/IP is a fairly fat protocol). It follows that the performance of the server could be adversely affected. Some vendors offer a special network card that offloads the TCP/IP overhead (and in some instances the iSCSIi overheads as well). These cards are obviously more expensive than simple network cards, though they are still cheaper than fibre channel cards

- Similar to any relatively immature protocol, iSCSI can be quite "chatty", i.e. multiple exchanges may be required between the source and the disk target to complete a single disk access. One paper that I have read discussed the use of a software-based cache on the server which was claimed to reduce protocol traffic by 3-4 fold

- The addition of disk traffic encapsulated in TCP/IP will increase the overall network traffic (possibly significantly). If the network design and implementation does not cater for the disk traffic the overall LAN performance may be adversely affected. Similarly, a single network connection from a server that is used for both normal network traffic and disk traffic may become a bottleneck if it is not correctly sized

- A common outcome when using NAS or IP SAN is that there is invariably less bandwidth between the server and the disk than is the case in a more traditional disk subsystem. Any decrease in bandwidth, coupled with the TCP/IP overheads, is likely to result in increased latency, i.e. longer disk response times

- iSCSI may be eminently suitable for small to low-end medium systems, or for those systems where disk performance is not crucial and hence higher latencies can be accommodated (e.g. email systems, off-peak batch tasks such as overnight backup, or possibly asynchronous remote mirroring to a Disaster Recovery site). However, it is very questionable if it is suitable for medium to large systems or systems where any disk performance degradation is not acceptable (e.g. OLTP systems)

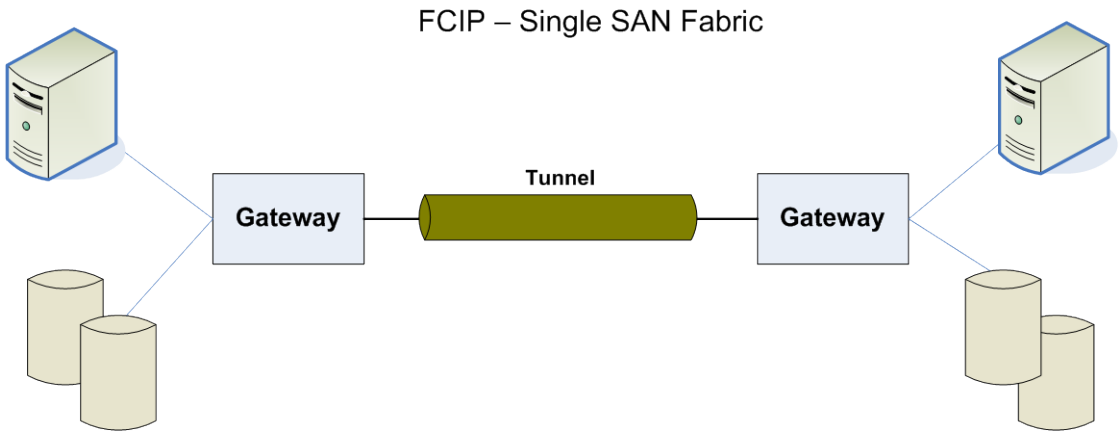
- Think carefully about using iSCSI for existing NFS or CIFS systems. Here you have already stretched the front-end (NFS client separated from the server) and hence increased disk latencies, and now you are looking at stretching the back-end (NFS or CIFS server to disk arrays).

The amount of marketing hype on this subject has increased markedly over the past 12 months. Much of it revolves around the message that running more stuff under the IP that we know and love must be a good thing, e.g. our network people do not need to be retrained and it will be significantly cheaper. I find this all a bit simplistic and slightly economical with the truth. Although Fibre Channel may still be expensive (relatively), I tend to go by the old maxim which indicates that ultimately you get what you pay for.

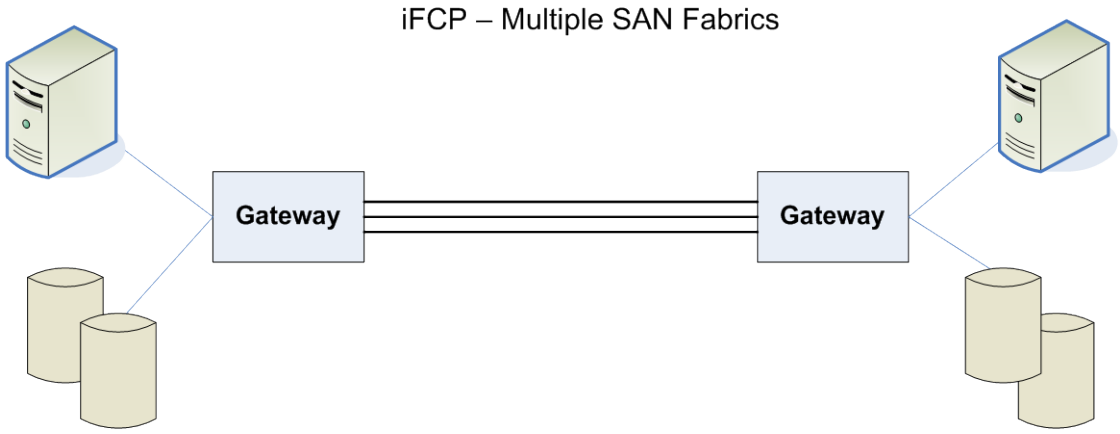
IP-Based Storage: FCIP and iFCP (Hard Disk: Fibre Channel, SAN and NAS taster)

FCIP and iFCP are mechanisms for running fibre channel commands over IP. Whereas iSCSI is very visible in the sense that it is involved in the whole physical disk I/O from its issue on a server through to the disk target (or IP SAN switch if the target is fibre channel), FCIP and iFCP are relatively invisible, in that they are part of the plumbing. They are implemented in gateways, sometimes called switches or more typically SAN routers by some vendors, allowing fibre-channel traffic to flow between “islands” of fibre-channel equipment over an IP network.

FCIP is the simplest protocol for a vendor to implement, as it simply encapsulates entire FC (Fibre Channel) frames within TCP, a technique that is often referred to as tunnelling. This effectively merges multiple SAN islands into a single SAN fabric, as shown in the diagram below. It can be said that FCIP operates at the fabric level. The easiest implementation is to work with a single tunnel, which in essence corresponds to a single TCP session.



iFCP operates at the device level rather than the fabric level. It maps each FC address (server, device or whatever) to a separate IP address and each FC session to a separate TCP session. In essence, iFCP replaces the FC transport layer with an IP network, e.g. Ethernet, but retains the information in the upper layer by mapping it to TCP. As Fibre Channel devices use FC Generic Services, IP-equivalent protocols are required for iFCP, the most obvious example being iSNS which provides storage name services. It is important to note that FC sessions are terminated at the gateway device, shipped over the network by IP to the target gateway where corresponding FC sessions are constructed to carry traffic to the target device. It seems reasonable to describe iFCP as a more native protocol than FCIP. If FCIP extends a single fabric then iFCP can be said to enable communication between multiple fabrics.



It should be noted that there is much semi-religious debate surrounding the use of FCIP and iFCP, mostly generated by the marketing people of the vendors who offer one or the other. The debate is partly fuelled by the fact that implementations of the protocols currently differ from vendor to vendor. Arguments may equally occur between vendors pushing iFCP, in addition to the more obvious FCIP versus iFCP battles. Differences in the implementation of the protocols bring the possible disadvantage of locking you into a single vendor.

Arguably, the main advantages of FCIP and iFCP are that they provide a cheaper solution than wall-to-wall fibre channel and they bring the concept of wide area storage networks closer.

With respect to performance, many of the observations on iSCSI also apply here. Briefly,

- TCP is a fat protocol that will inevitably result in longer latencies than vanilla-flavoured fibre channel. This may be an issue (e.g. in larger OLTP systems) or it may not (e.g. in messaging systems)

- Congestion, due to inadequate network bandwidth or spikes of access activity, will exacerbate latencies, as TCP takes its usual avoiding action by reducing the rate of flows across the network. This may be unacceptable for disk performance

- Congestion may more of an issue in FCIP, particularly in those implementations where there is a single tunnel, as the multiple FC flows which have been merged into that tunnel will all be affected, whereas not all sessions may be affected in iFCP.

FCIP and iFCP may be well suited to those storage infrastructures that only have to support small to medium disk access rates, for off-peak activities such as overnight back-up, or for those systems where asynchronous IO is satisfactory (e.g. remote mirroring). They are unlikely to be suitable for larger infrastructures, particularly those with high access rates.

Beware of vendors peddling IP storage, replete with the warning that fibre channel technology is dead. I am not sure whether they are talking about all parts of the technology or just the fabric. Needless to say, they are probably network vendors. Take what they say with a very large pinch of salt.

Finally, having whinged previously about the quality of technical information that is put out by some Technology Associations and Forums, I have to say that I found some very useful documents on the SNIA site; viz. *iFCP - A Technical Overview* and a slide show entitled *IP Storage* by Dave Dale of Network Appliance.

Infiniband-based SANs (or even Infiniband-based IP)

The recent taster on cluster interconnect technologies and the various pieces on IP-based storage have resulted in a question from one reader on the possible use of Infiniband to provide the fabric for SANs, as promoted by the Infiniband Trade Association.

In essence, the Infiniband architecture provides a conduit for other protocols to run over the medium; or to use the current lingua franca these protocols are “tunnelled” through Infiniband. For example, IPoIB supports the tunnelling of IP. With respect to storage, the SCSI RDMA Protocol (SRP) runs over Infiniband. Similar to iSCSI in

the IP world, SRP requires an initiator at the client end (i.e. on the server), as well as the necessary protocol support at the device target end.

The promoters of Infiniband preach the message of a single cluster-wide IO fabric, obviating the need for Fibre Channel fabric. The marketing spiel talks about the simple need for a single interface card (HCA) on each node in the cluster to support both inter-node and storage traffic. However, the target disk subsystem(s) also need HCA(s) and the necessary support for SRP. While Mellanox, one of the major Infiniband vendors, provide a modest Infiniband-based storage system that supports up to 1TB of SATA disk, the major storage vendors have been loath to follow suit. A more popular approach has consisted of the use of Infiniband-to-Fibre Channel gateway boxes from companies such as TopSpin (recently acquired by Cisco) and Voltaire (part-funded by Hitachi). One side of the gateway plugs into the Infiniband fabric while there are storage ports on the other side, viz. Fibre Channel and SCSI. IBM is currently selling a BladeCentre, consisting of up to 14 blade servers that can form a cluster via a TopSpin-based Infiniband fabric; TopSpin gateways are supported to allow storage and IP extensions. Currently available gateways are limited, typically supporting only two Fibre Channel ports. While this may not be an issue on small systems, it is likely that multiple gateways will be necessary on any reasonable-sized system.

From a performance perspective, the plus points are the inherent high throughput and low latency of Infiniband, while the possible disadvantages are the overheads of SRP or the gateway, which obviously depend on the implementation. Careful sizing is required if there are significant amounts of both inter-server and storage traffic at a given node. Apart from basic throughput requirements, the performance of the server IO infrastructure on each node may need to be assessed, e.g. the capacity of a PCI bus.

As mentioned, Infiniband also supports IP via IPoIB. This can muddy the waters in the storage area, as iSCSI, the IP-based storage protocol, can be run over Infiniband using IPoIB. Despite the availability of increased bandwidth it is still likely to suffer from the relatively poor performance of TCP. Some improvements can be obtained by using the RDMA assist for iSCSI. Here, the control aspects of the protocol continue to be sent by TCP but the actual data uses RDMA to bypass it. The effect is to maximise the advantages of the increased bandwidth and low latency, although the benefits are more likely to be noticeable when larger storage block sizes are used.

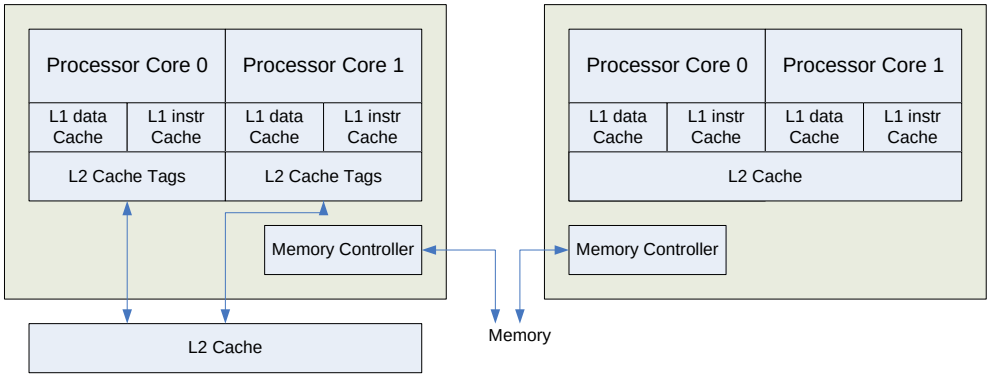
The bottom line at the time of writing is that, while all vendors are keen on the concept of a unified fabric that will support all system IO, there is no general market acceptance of Infiniband, or indeed of any other technology. Infiniband may triumph eventually but we will have to wait and see. I will pen some words on other IO technologies shortly.

Dual Core Processor Chips (CPU Basics and Multiprocessors – Shared Memory Tasters)

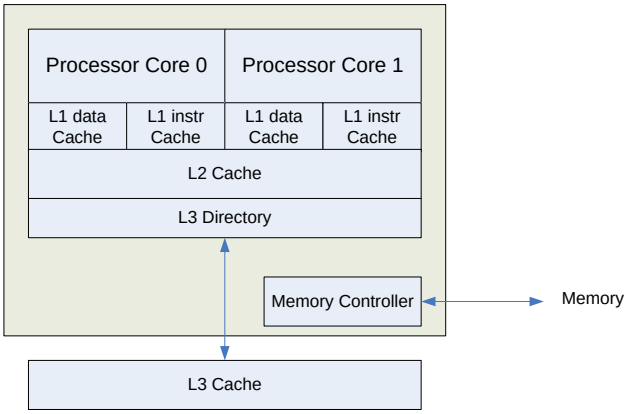
Dual core CPU chips (two CPUs on a single die) have been gradually appearing in server-based systems, including: IBM who were early in the field with Power 4, and

more recently with the Power 5; Sun's UltraSPARC IV appeared in 2004; AMD's announcement came early in 2005; while Intel's initial offerings in the IA32 space were announced at the back end of 2005, their hand probably being forced by AMD. HP is also in this arena with the PA-8800 and PA-8900. However, they are making few noises about them, as the overall plot (at least from a commercial perspective) seems to be that Itanium will replace PA-RISC chips in their server systems, as soon as is practical. With respect to Itanium, Intel's dual core Montecito chip is currently slated for 2007 although HP already has its own version of dual core Itanium, a temporary measure until Montecito appears.

Dual core designs vary, as shown in the examples on the diagram below. The basic set up (a) incorporates two processor cores, each with its own L1 caches; the memory controller is shared by the cores; and the L2 cache is off-chip. An improvement (b) is similar except that the L2 cache is on-chip. A further refinement (c) also has the L2 cache on-chip but introduces a larger L3 cache which is held off-chip.



(a) L2 Cache Off-chip **(b) L2 Cache On-chip**



(c) L2 Cache On-chip plus L3 Cache Off-chip

When comparing the performance of systems with single and dual core processor chips, there are two key areas: cache coherence and general cache performance; plus memory performance. Minimally, there needs to be as much cache, main memory and system bus bandwidth per CPU on a dual core-based system as there is on the equivalent single core system. Some people claim that the amount of main memory accesses will be reduced on a dual core system, on the basis that the data required is more likely to be in the processor cache, possibly having been previously fetched by the other CPU on the chip. This seems highly speculative: yes, it can be the case on

certain workloads that exhibit high levels of non-volatile data / instruction sharing between threads; but the more likely case, certainly in the commercial world, is that applications exhibit high levels of memory volatility, particularly in the middle and database tiers, and therefore the likelihood of cache hits (finding the data already in the cache) is probably low.

Let us assume, for the sake of argument, that we have two systems which are largely identical, except for the use of single or dual core processor chips. In particular, each system has the same amount of main memory, L1 and L2 cache and system bus bandwidth per CPU. In a dual CPU system, the single core variant will need to use a standard SMP cache coherency protocol to ensure that data integrity is maintained across the two CPUs. However, the dual core variant only requires one processor chip to house the two CPUs, obviating the need for cross chip cache coherency. All things being equal (and I know that they are often not) the dual core variant should perform at least as well as the single core variant in this dual CPU system, possibly better. In reality, the amount of cache per core, particularly L2, is reduced on many current implementations and performance will suffer. Some early implementations of design type (a) and (b) lose as much as 25% speed per core, i.e. a dual core will perform at 1.5 times the speed of the equivalent single core; better designs may lose 10-20%; while design type (c) may only lose 5% (or less) per core.

On a 4 CPU system, or above, the dual core system must also resort to using a cache coherency protocol and will lose the advantage that it had on the dual CPU system. Based on a limited sample, a dual core system with 4 CPUs (based on design type (b) in the diagram), running a typical commercial workload appears to under-perform the equivalent single core system by 10-20%. This appears to be an intuitive result. A system based on design type (c) in the diagram should in theory be closer to the performance of a single core system although I have not seen any numbers to validate this speculation. Naturally, there may well be other factors to take into consideration when comparing performance, viz. any price difference between the respective single and dual core systems, plus any performance improvements that may have been made in other areas of the system, e.g. the underlying server infrastructure (chipset), which may help to offset the reduction in performance.

In addition to the provision of dual core processor chips, some vendors are also introducing support for SMT (Simultaneous Multi-Threading), which allows multiple threads (typically two at the current time) to execute concurrently on a single CPU. This obviously muddies the waters further when attempting any comparisons, as the purpose of SMT is to increase the throughput of each CPU. See the taster on this subject for further information.

Multi-core roadmaps generally focus on squeezing more cores onto a single chip. Manufacturers of embedded systems are already producing products in this area. For example, in the commercial computing world Sun have plans to introduce a chip in 2006, currently codenamed Niagara, with eight cores and support for 4 threads per core. The cores will be simplified, in comparison with a typical modern complex CPU that attempts to get as much performance as possible by employing all the tricks of the trade, e.g. speculative memory fetches, out of order processing, *et cetera*. Rather, the approach in Niagara will simply be to switch to another core / thread if any sort of delay is encountered. The emphasis will be on keeping as many cores / threads as

busy as possible. Niagara will be deployed in a single processor chip system, i.e. there will be no support for traditional SMP. It seems likely that this type of system will be deployed in a blade farm, arguably slanted at web servers / streaming media servers.

IO Interconnect Technologies (Server Infrastructure)

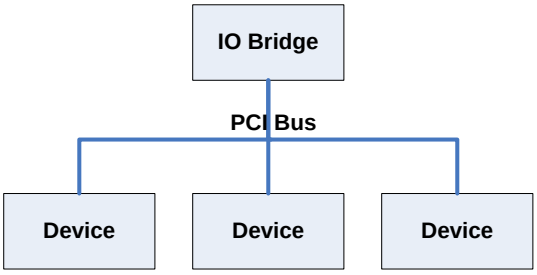
PCI has been the major IO interconnect technology for close on a decade. Although it started life in the PC world it quickly spread to servers. Despite its universal popularity, PCI has a number of disadvantages which are becoming more noticeable as the years slip by:

- It employs a **shared bus technology** (as shown in diagram (a) below). This means that only one device can use the bus at a time, which in turn means that a bus arbitration protocol is required to obtain use of the bus

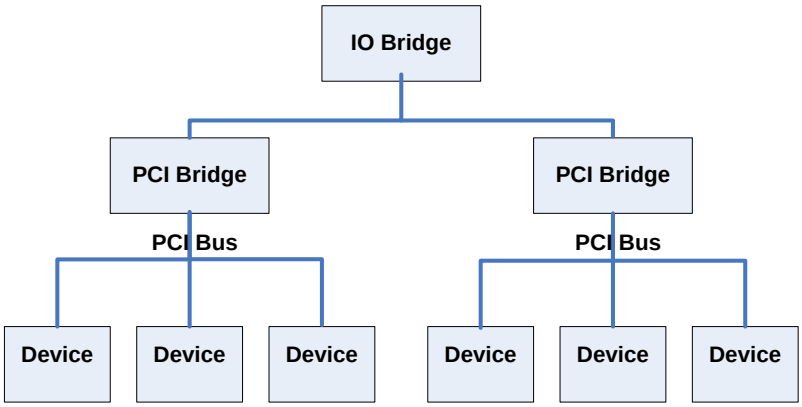
- **Electrical noise** can limit the number of devices on a bus. This is handled by a hierarchical approach where multiple buses are accessed via bridges (see diagram (b) below)

- **Bandwidth / speed.** PCI started out with 32 bit buses running at 33MHz (equivalent to 132 MB/sec). Refinements have led to: 64 bit at 33MHz (264 MB/sec); 64 bit at 66 MHz (512 MB/sec). It is typically the slowest part of a server, which is becoming an issue with the advent of higher speed networks and cluster interconnects

- It uses a straightforward **load-store, flat memory-based communications model**, which is a constraint, particularly when compared with a routed, packet-based approach.



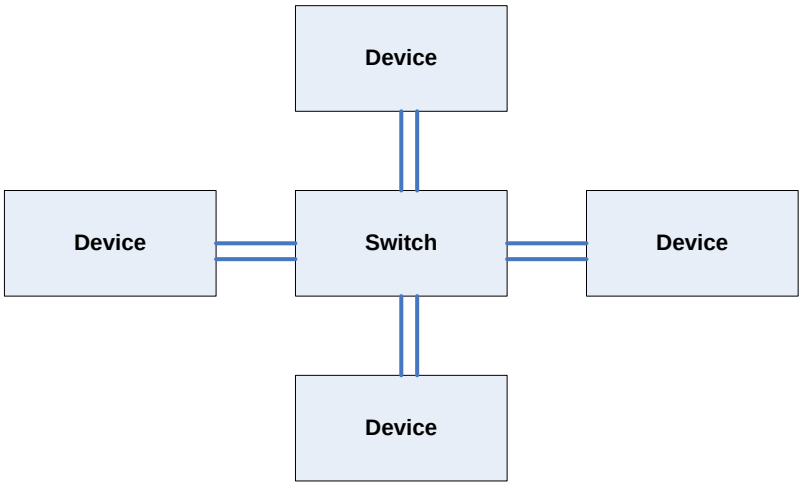
(a) Simple PCI Bus



(b) Hierarchical PCI

A short term solution, particularly with respect to the bandwidth / speed issue, is PCI-X. It started with a top end of 133MHz but versions are now available at 266 MHz and 533 MHz (plus 1066 MHz is being talked about). The arrival of 266 MHz saw the introduction of double data rate (transferring data on both the rising and falling edges of a clock cycle). The theoretical speeds of 266 MHz and 533 MHz working at double rate are 2 GB/sec and 4 GB/sec respectively. The other problems remain and issues like electrical noise can be exacerbated by the increase in traffic.

Work has been going on in the industry to replace the shared bus approach with a high performance switch-based, point-to-point technology, as shown in the diagram below.



Switch-based IO Interconnect with 2x links

PCI-SIG has plumped for Intel's 3GIO technology, which is now known as PCI Express. Here a device is connected to the switch via a serial link. The basic link (called 1x) transmits at 2.5Gbits/sec in each direction (full duplex is supported). Multiple links (also called lanes) can be bonded together to form a channel to support higher bandwidth. The PCI Express specification talks about 2x, 4x, 8x, 16x and 32x links (a 32x link supports a theoretical 16GB/sec). Support for PCI is part and parcel of PCI Express. "Bridges" can translate PCI Express packets to PCI signals and vice versa. These bridges can be on a motherboard or on a card. The switch itself can be part of a server chipset. For example, the Intel 900 series chipset incorporates the switch on the southbridge; it includes "bridge" logic which will allow both PCI and PCI Express devices to be supported. Advanced Switch is a new feature of PCI Express which will attempt to challenge RapidIO in the embedded system market.

Although PCI Express is arguably in pole position to take over from PCI as the de facto industry standard, there are other serial switch-based technologies that are in the frame:

- **HyperTransport** developed by AMD. This is already used as a CPU interconnect on AMD and MIPS processors. It employs a low latency parallel point-point technology with varying width of bits per link – 2,4, 8, 16 or 32. A 32 bit wide path can support a theoretical 22.8GB/sec. It is used as an IO bus in Apple's G5 PowerMac. PCI is supported. A feature called Direct Path is being introduced that will attempt to challenge RapidIO in the telecoms arena

- **RapidIO** developed by Motorola. The focus of this low latency technology has been on embedded systems, particularly in the telecoms market, e.g. DSP farms. There are two variants: Parallel RapidIO has 8 and 16 bit paths, operating at up to 6 GB/sec; while SerialRapidIO has 1 and 4 lanes, operating at up to 1.8GB/sec

- **Infiniband.** This technology is arguably trailing at the moment in the IO interconnect space. See the *Cluster Interconnect Technologies* taster for background information.

Note that it is not valid to simply compare the bandwidth of a single link across the technologies. The bandwidth of the overall system, along with the associated latencies, is the real criterion. In fact, it is probable that the performance of the switch will be the ultimate differentiator.

The high speed serial IO interconnect is a marketplace which it will pay to monitor, as the various parties jockey for market share. At the current time PCI Express seems to be the favourite in the general purpose server arena with RapidIO in the embedded design market. HyperTransport, along with Infiniband, appear to be trying to be all things to all men.