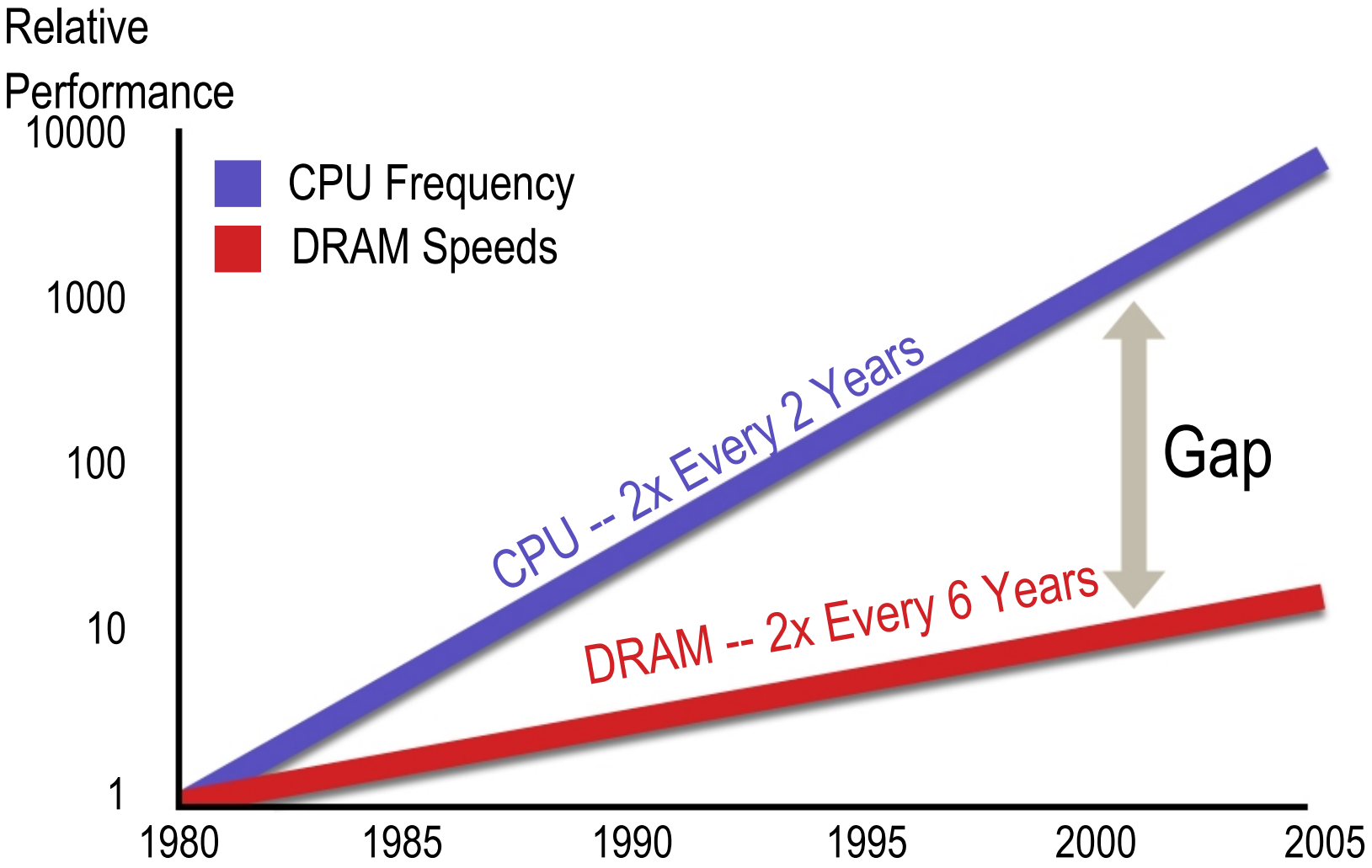
Hardware and Software solutions for scaling highly threaded processors

- Denis Sheahan
- Distinguished Engineer
- Sun Microsystems Inc.

Agenda

- Chip Multi-threaded concepts
- Lessons learned from 6 years of CMT
- Aim of Victoria Falls
- Hardware scaling
- Scaling the OS
- Virtualization and Consolidation
- Scaling Applications
- Conclusions

Memory Bottleneck

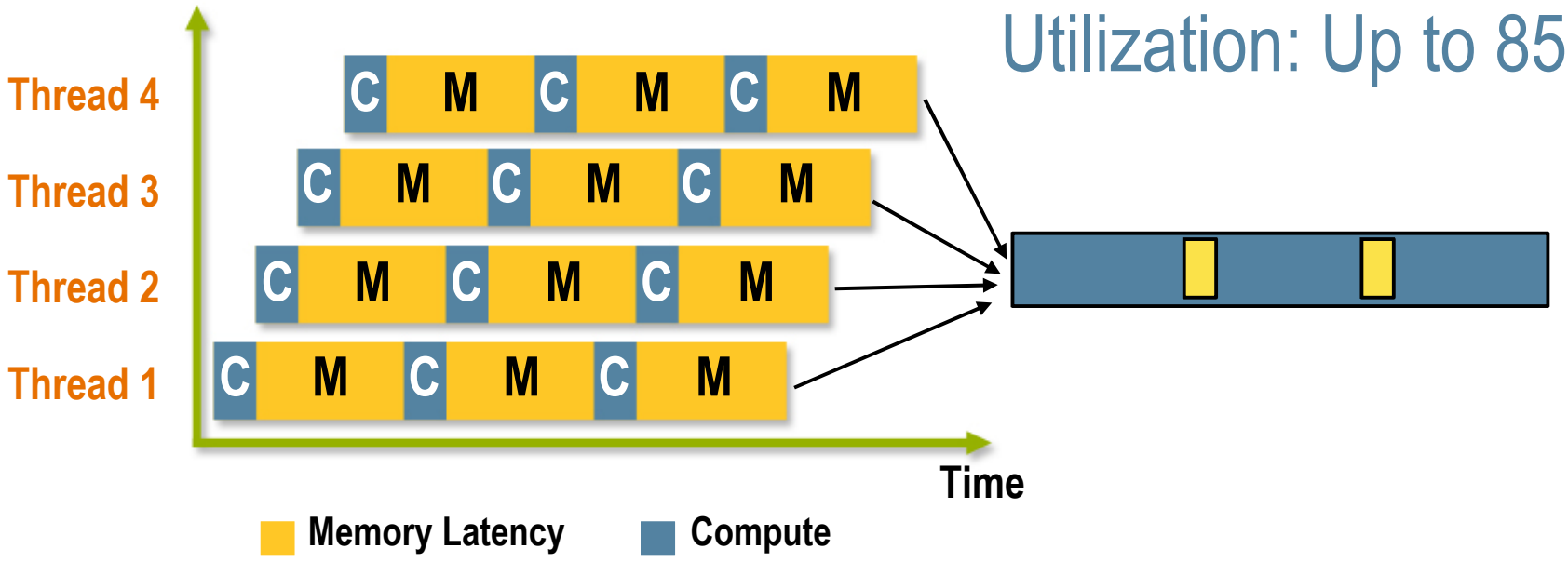


Source: Sun World Wide Analyst Conference Feb. 25, 2003

CMT Implementation

Four threads share a single pipeline
Every cpu cycle an instruction from a different thread is executed

Niagara Processor
Shared Pipeline
Utilization: Up to 85%



Lessons from CMT

- **Multiple cores are here to stay.**
  - Already quad core, 8 core on most roadmaps
  - Big impact on commercial applications,
  - Big problem for desktop and laptop software
- **The pursuit of frequency is over.**
  - Apart from one or two vendors most have dropped their 4GHz and 5GHz plans
  - Very high frequency usually means higher power
  - Higher frequency usually adds complexity and lengthens time to market

Lessons learned from CMT

- **Power and cooling have become extremely important to most customers**
- Ecoresponsibility and carbon footprint are the new standards for datacenters
- Many datacenters are at capacity
- Acceleration in the development of lower power systems
- Also accelerated Virtualization technology.
- Customer trading performance versus power

Lessons learned from CMT

- **Many throughput workloads benefit more from higher thread count than higher frequency**
  - CMT has proven this time and again with public benchmarks
- **Some code is single threaded and we need to work to make them parallel**
  - Not as big an issue in the HPC space but even here can have periods such as results aggregation and report generation that are serial

Big Question

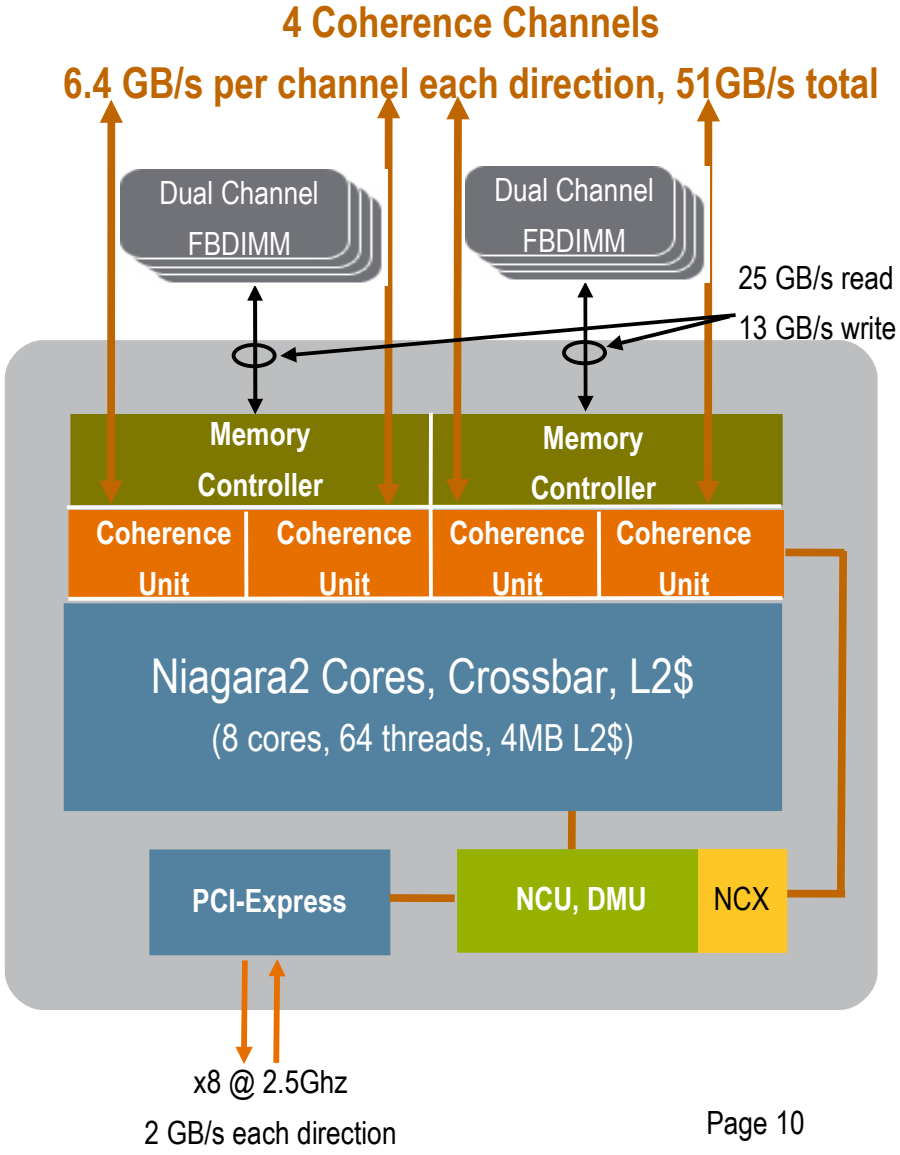
How do you make CMT Scale to 32 cores and 256 threads

Aim of Victoria Falls

- Create an SMP version of CMT to extend the highly threaded Niagara design
- Use T2 as the basis for these systems
- Minimal modifications to T2 for shorter time to market
- Create two-way and four-way systems without the need for a traditional SMP backplane
- Avoid any hardware bottlenecks to scaling
- High throughput low latency interconnect
- Scale memory bandwidth with processors
- Scale I/O bandwidth with processors
- Hardware features to enable software scaling

Victoria Falls Processor

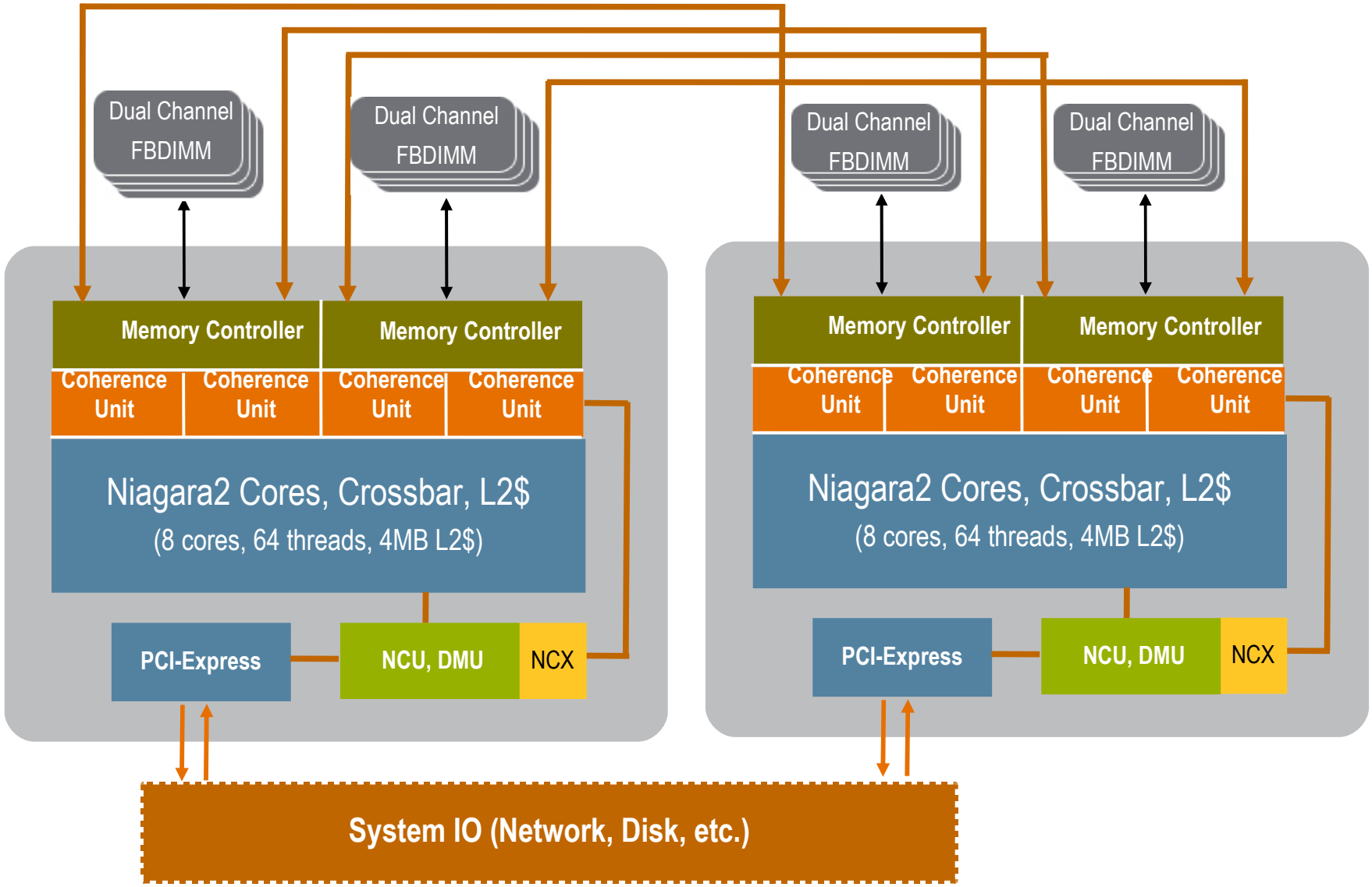
- VF is a derivative. Re-use UltraSPARC T2 cores, crossbar, L2$, PCI-Express I/F
- Remove 10GE network interface
- Replace two memory channels with four coherence channels
- Increase memory link speed to 4.8Gbps (was 4.0Gbps)
- Added relaxed DMA ordering to PCI-Ex interface
- All I/O on VF via the x8 PCI-E link per chip, scale I/O with processors
- Similar packaging as US T2



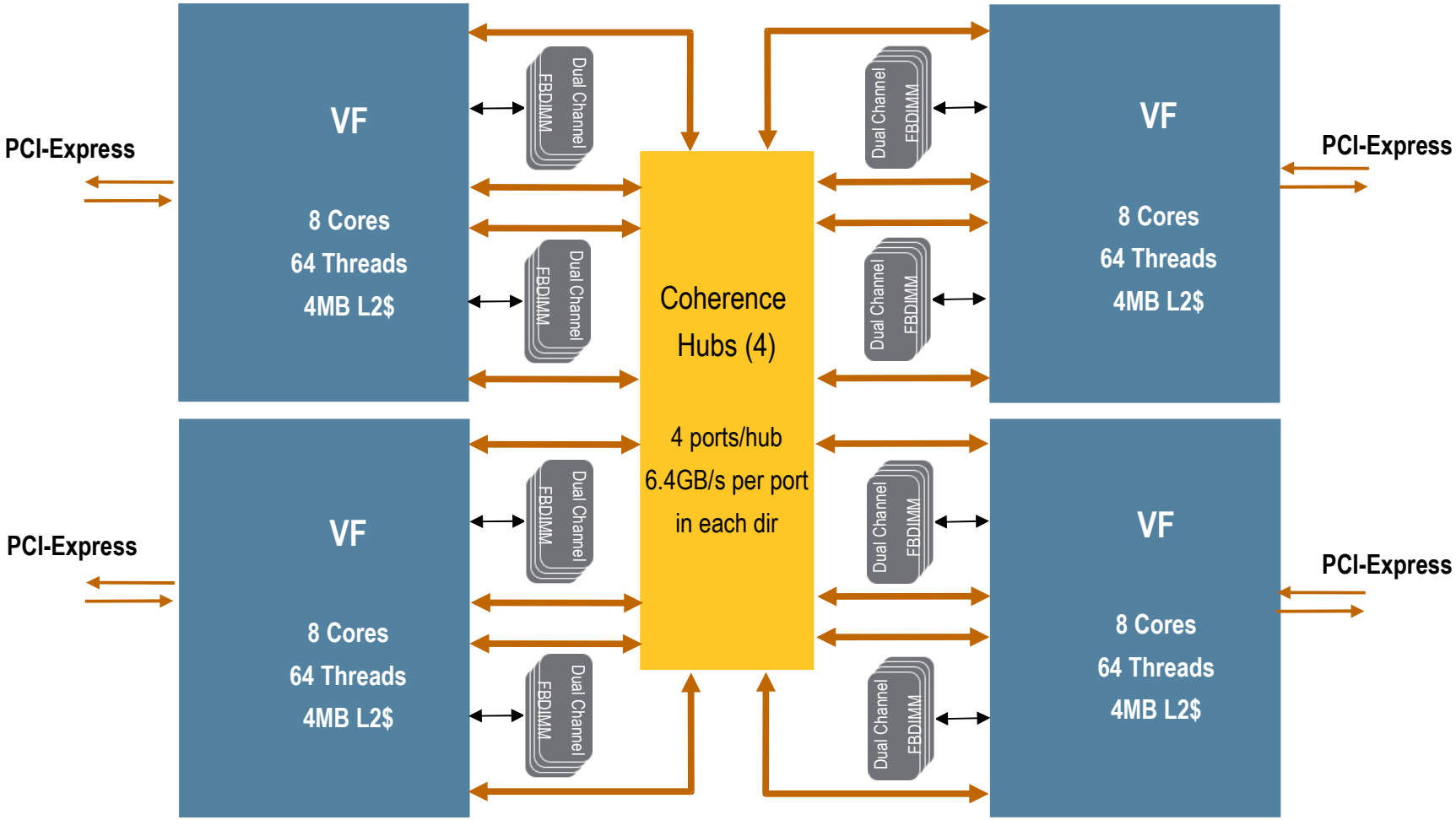
Hardware Scaling - Interconnect

- Interconnect is key to scaling to 32 cores and 256 threads
- Decision not to use a traditional bus given the memory bandwidth requirements
- Instead we used half the FBDIMM channels for memory and the other half to create 4 bidirectional links for interconnect
- Physical link of interconnect same as FBDIMM
- Reuse the FBDIMM pins for the interconnect
- Result is a high speed SERDES link
- Provides high bandwidth and a low latency interconnect

VF 2-Socket System



VF 4-Socket System



Hardware Scaling - Memory

- Two elements are key to scaling on highly threaded servers
- Memory bandwidth. Hundreds of threads require a very wide pipe to memory
- Local vs. Remote memory latency. Systems that are highly NUMA can show inverse scaling if memory is placed badly
- Each VF processor has its own local memory connected delivering high memory bandwidth
  - 21GB/sec Read and 10GB/sec Write
  - Memory bandwidth scales with the number of processors
- Remote memory latency has a 76ns penalty for access, about 1.5x latency to local memory This makes VF systems NUMA although not highly so.

Software Scaling - scaling the OS

- There are a set of common issues when scaling the OS on a Multi-core highly threaded system
- Scaling single threaded kernel services
- Scaling contended mutexes
- Optimising memory placement
- Optimal scheduling across cores
- Scaling I/O
  - Local vs. remote
  - Delivering interrupts
- Virtualization

Single threaded kernel services

- Prime example of a single threaded kernel service is when Solaris performs accounting and book keeping activities every clock tick.
- As the number of CPUs increases, the tick accounting loop gets larger. On a 4 way VF system there are 256 threads to check
- Tick accounting is a single threaded activity and on a busy system with many CPUs, the loop can often take more than a tick to process if the locks it needs to acquire are busy. This causes clock drift and other unpleasantness.
- Solution is to involve multiple cpus in the Tick scheduling. Cpus are collected in groups and one is chosen to schedule all their ticks
- Algorithm is used to spread the tick accounting evenly across active cpus.

Scaling contended mutexes

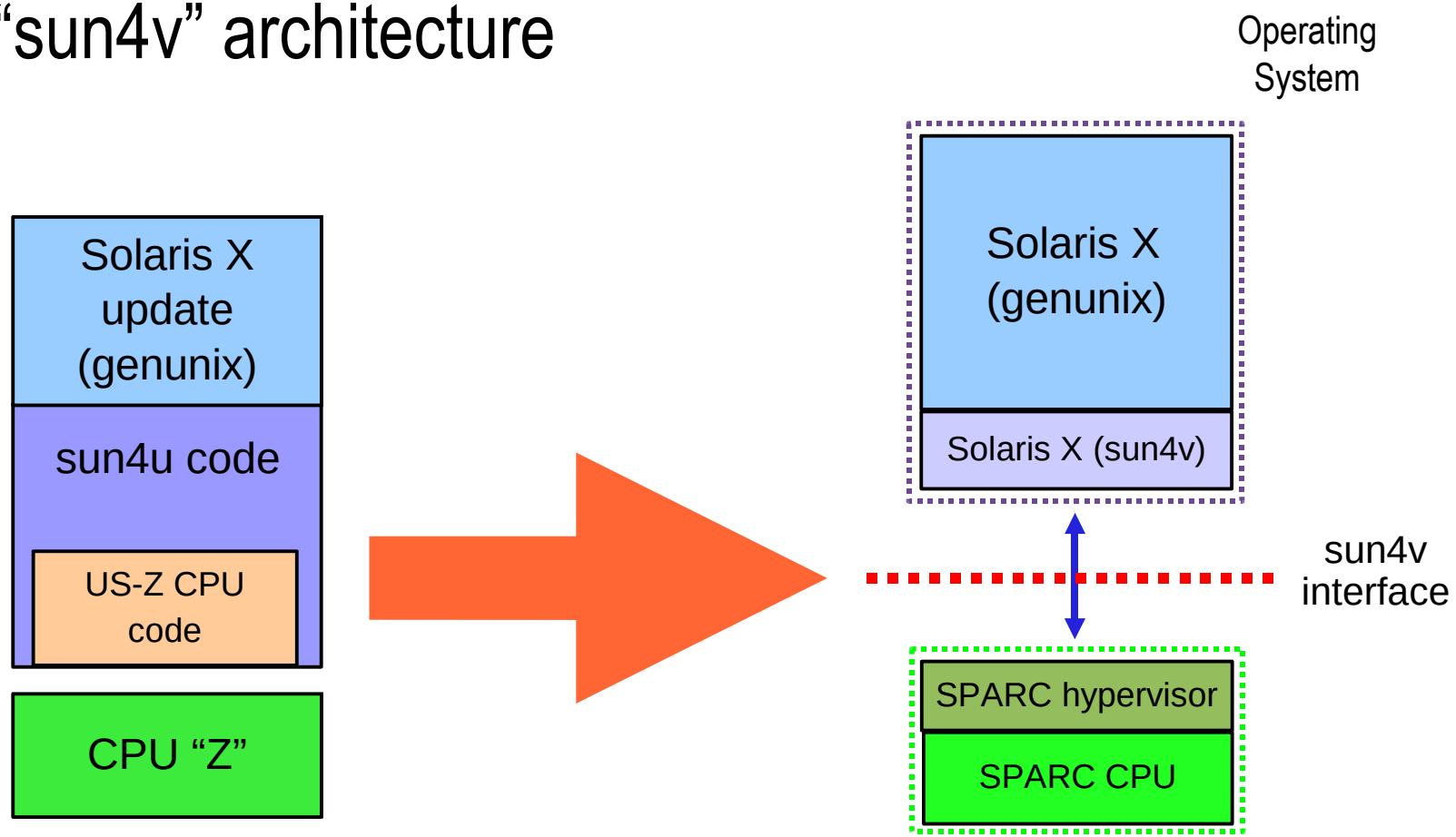
- mutexes in Solaris are optimised for the non contended case. Contended mutexes are considered rare
- With up to 256 active threads in a single OS instance, calls such as *atomic_add_32()* on contended mutexes can become much hotter.
- Solaris solution is an Exponential backoff algorithm. This has the advantage of little overhead in the non-contended case and performance gains in the contended case

NUMA aware OS

- As we have seen VF systems are the first NUMA CMT servers
- A NUMA aware OS ultimately helps in scaling an application as it reduces dramatically the interconnect traffic between chips in the system
- Solaris has been NUMA aware since Solaris 9
- Processes are assigned to a local *lgroup* where hot areas of their address space such as heap and stack are allocated and where they will be scheduled
- The effect is that there is an increase in accesses to lower latency local memory

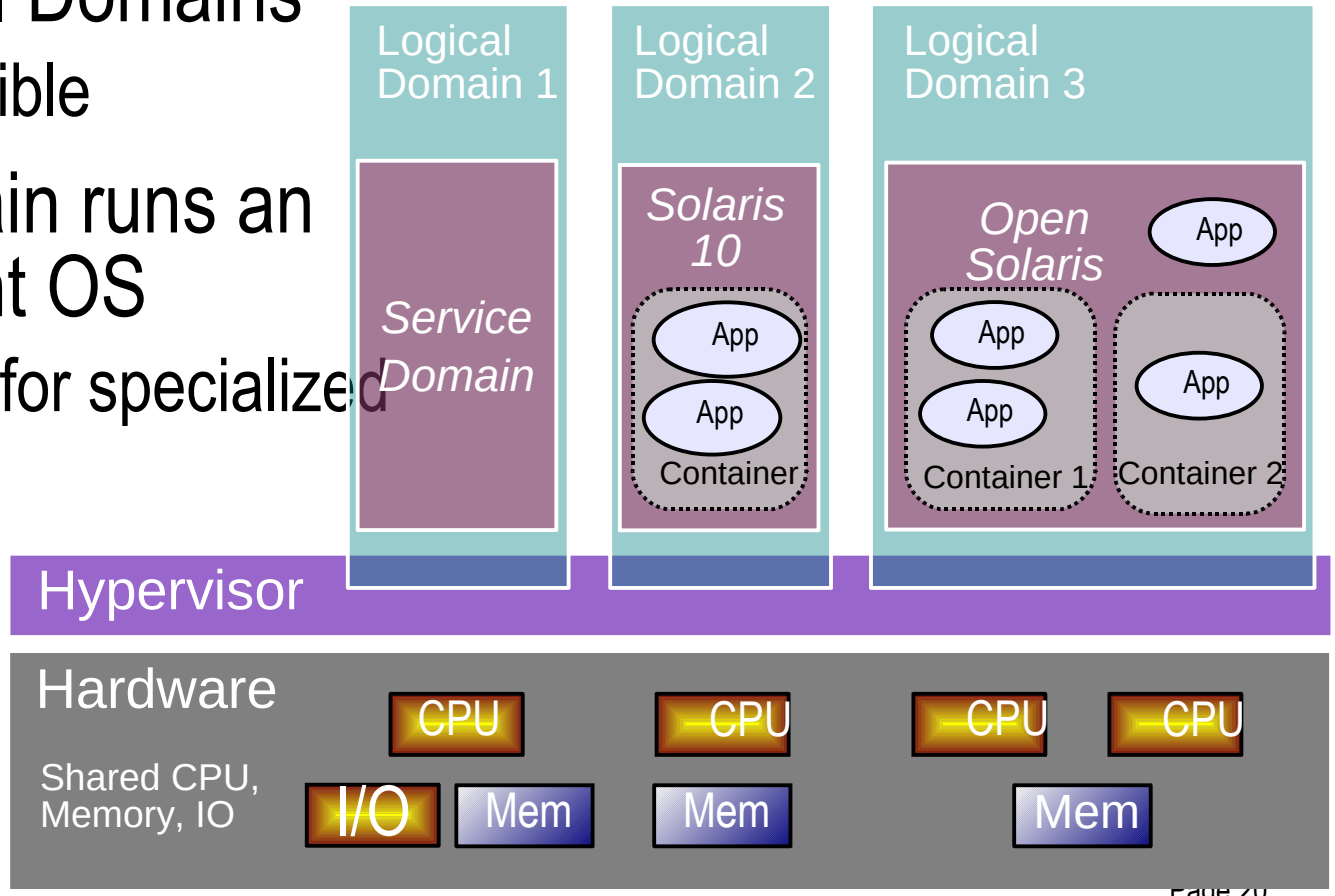
The Hypervisor: Virtualization for CMT Platforms

- "sun4v" architecture

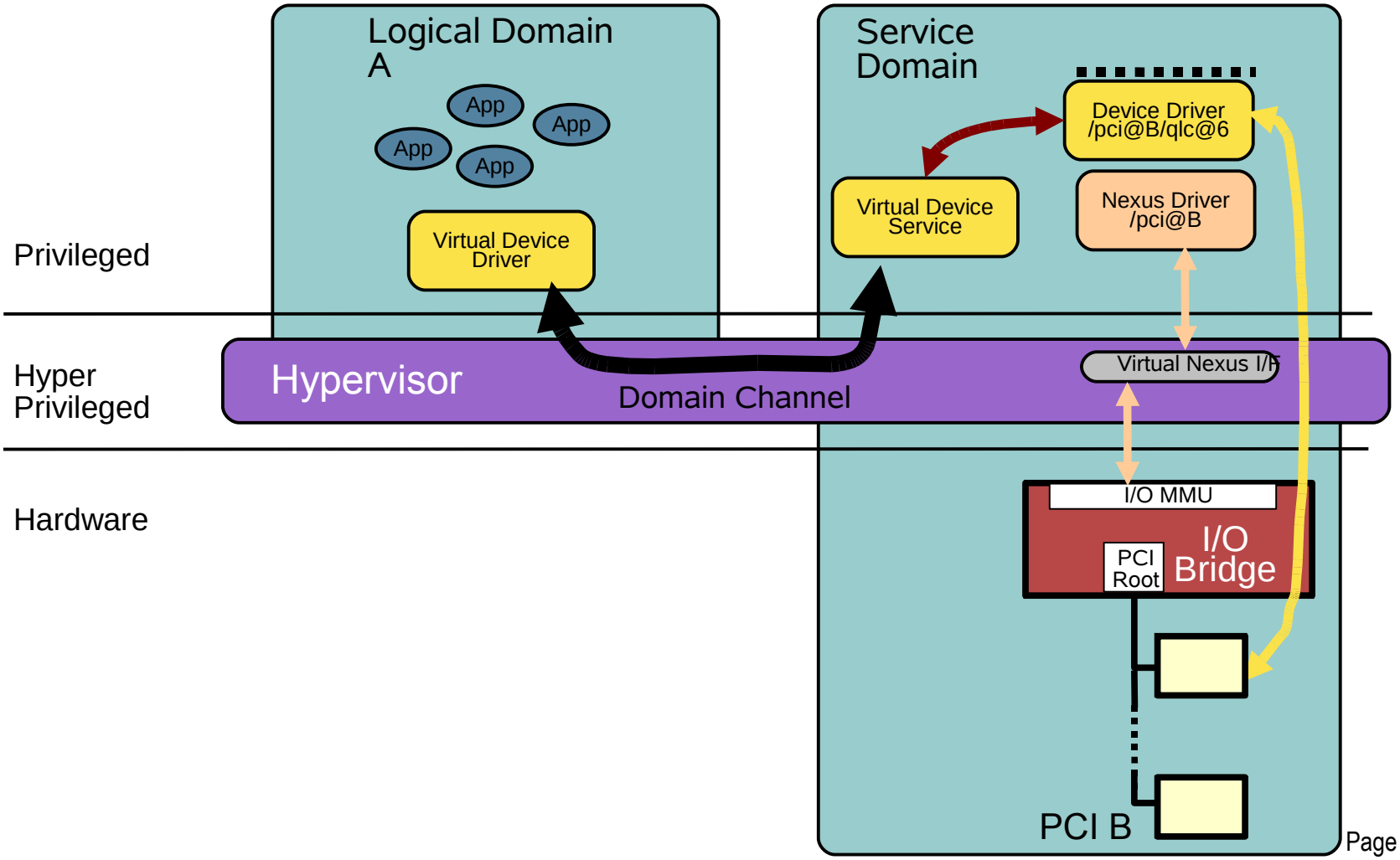


The Sun4v/HV/LDOMs Model

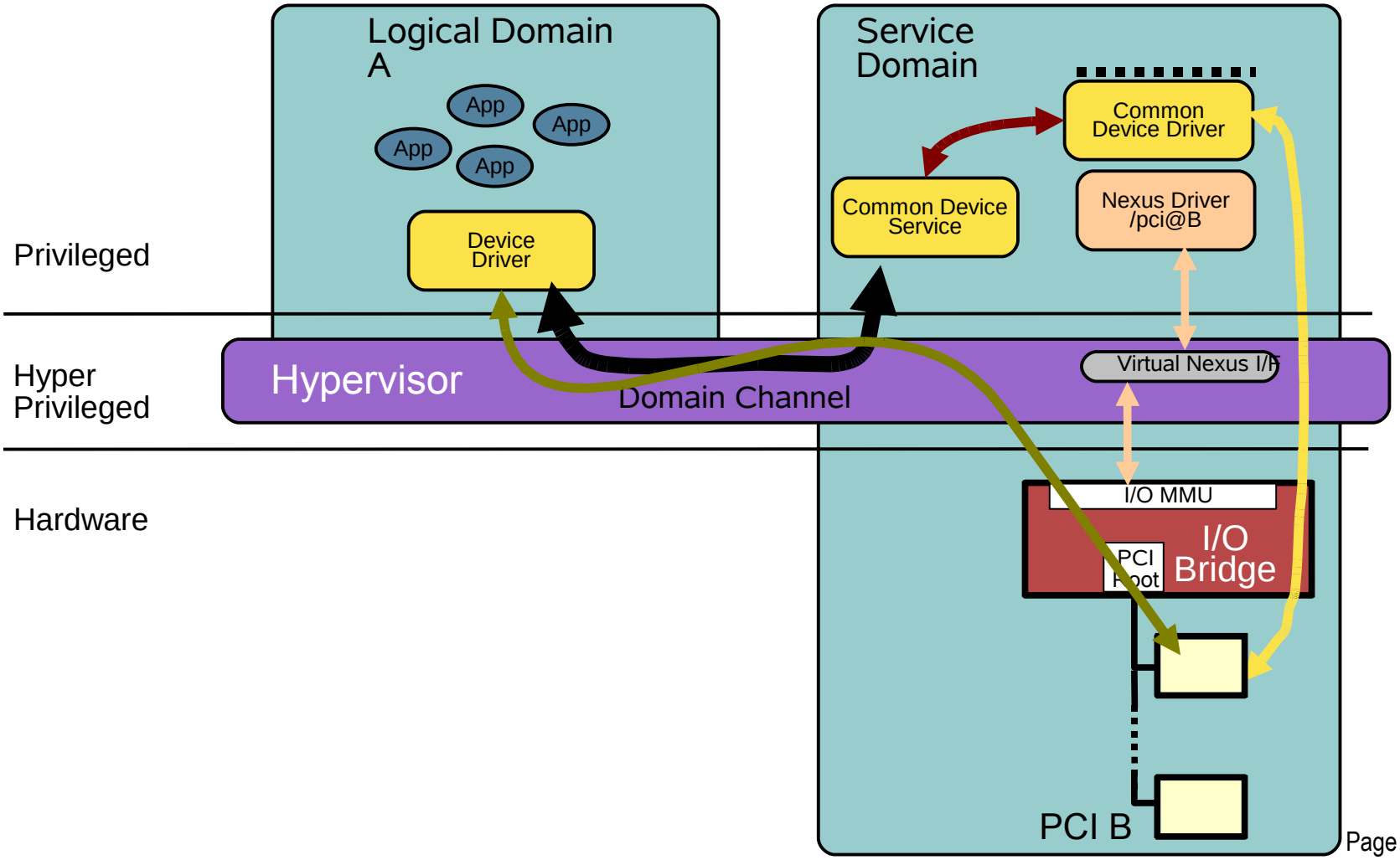
- Replace HW domains with Logical Domains
  - Highly flexible
- Each Domain runs an independent OS
  - Capability for specialized domains



Virtualized I/O



The future - Hybrid I/O



Scaling Applications on many cores

- Scaling and throughput are the most important goals for performance on highly threaded systems
- High Utilisation of the pipelines is key to performance.
- Need many software threads or processes to utilise the hardware threads.
- Threads have to be actually working
- Multiple instances of an application may be necessary to achieve scaling
- A single thread can become the bottleneck for the application
- Spinning in a tight loop waiting for a hot lock is not good for scaling on CMT

Utilization and Corestat

- In order to determine the utilization we use low level hardware counters to count the number of instructions per second per pipeline – both integer and floating point
- We have written a tool called corestat which collects data from the low level performance counters and aggregates it to present utilization percentages
- Available from *http://cooltools.sunsource.net/corestat/*
- Corestat reports :
  - Individual integer pipeline utilization (Default mode)
  - Floating Point Unit utilization -g flag)

Locking issues

- The most common reason an application fails to scale on CMT is hot locks
- We have found this especially true when migrating from 2-4 way systems where access to hot locks and structures is effectively serialized
- If a system has 16 cores and 128 threads, then there can be up to 128 instruction streams executing in parallel. Hot locks which will tend to reside in the L2 cache and can become extremely contended.
- When developing locking code we need to use a low-impact, long-latency opcode to add delay in the busy-wait loop. The low-impact opcode frees up cycles so that other threads sharing the core get more useful work done

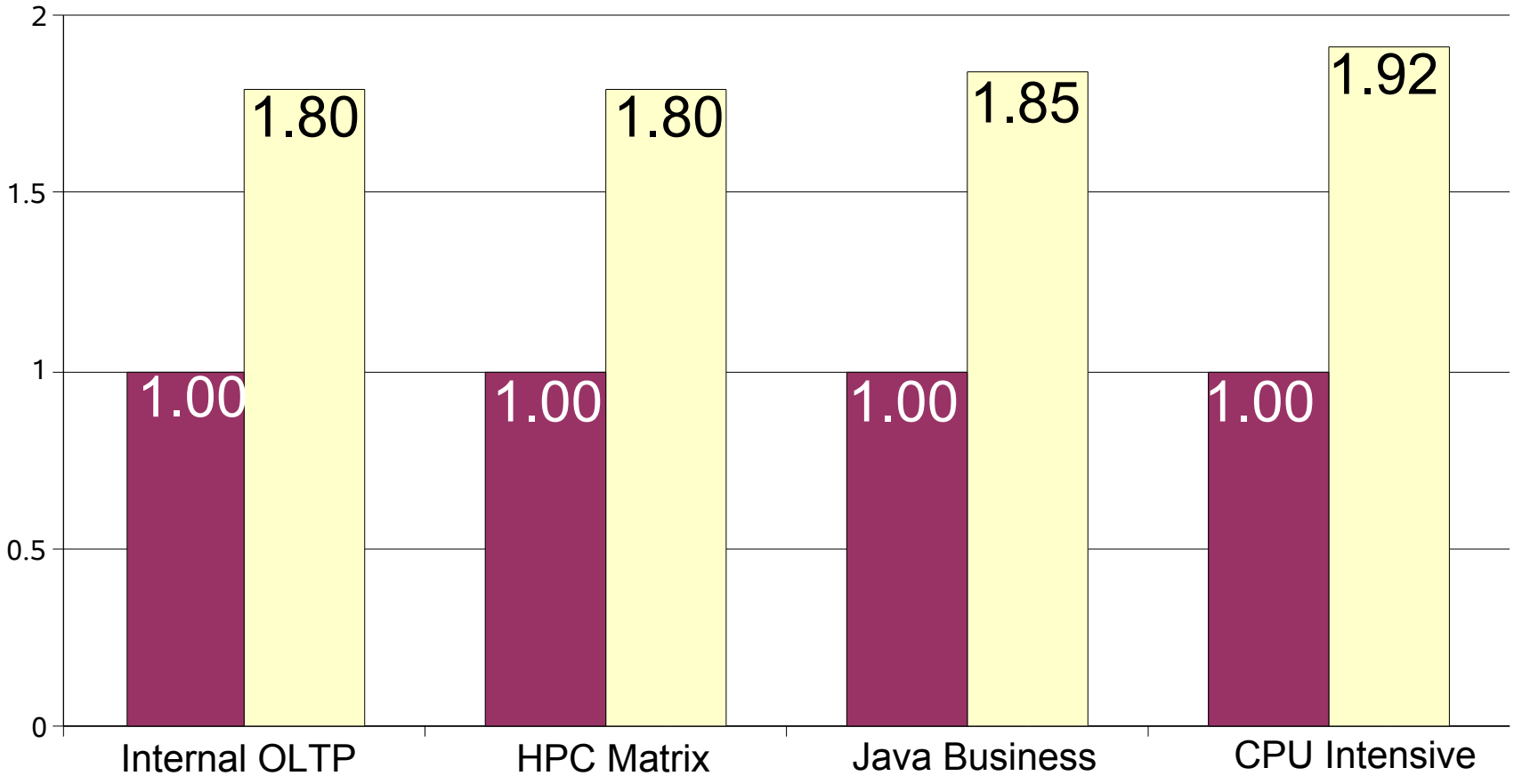
Proven Applications Areas for CMT

- Web-servers: Apache, SunOne, Lighttpd
- Web Proxy: Sun Web Proxy
- J2SE Appservers: BEA WLS, Websphere, SunOne, Oracle
- Database Servers: Oracle, Sybase, mySQL, Postgres, DB2
- Mail Servers: Sendmail, Domino, Brightmail, Openwave
- ERP: Siebel, Peoplesoft Oracle E-Business
- Security

Proven Applications Areas for CMT

- JES Stack: Directory, Portal, Access Manager
- NFS Servers
- DNS: BIND
- E-Learning: Blackboard
- Capacity Planing: BMC, HP, Tivoli
- Net Backup:
- Search: Lucene
- Development: Clearcase, compile servers
- Streaming Video

VF Performance Scaling



- Single-chip 1.4Ghz Victoria Falls, 8 Cores, 64 Threads, Solaris
- Dual-chip 1.4Ghz Victoria Falls, 16 Cores, 128 Threads, Solaris

VF Availability

- Victoria Falls 2-socket Niagara systems Available today
- Victoria Falls 2-socket Niagara blades 2HCY08
- Victoria Falls 4-socket Niagara systems 1HCY08

Conclusions

- Multiple cores are here to stay and the pursuit of frequency is over
- Power and cooling are the new driving factors in the datacenter. This requires much higher utilization through scaling, consolidation and virtualization
- A high bandwidth, low latency interconnect is essential to achieve good scalability on multi-core systems
- Scalable memory bandwidth and physical I/O is also required
- An OS running on a highly threaded processor must also be scalable and in many cases NUMA aware. It must also be virtualizable
- Servers with many cores and hardware threads present a different set of scaling issues to applications