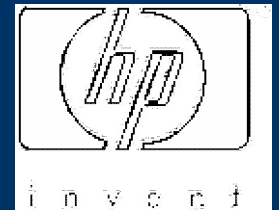


Technology for better business outcomes



# HP-UX 11i v3 Knowledge-on-Demand

- Objective: Support developers, deployment personnel and customers in achieving better business outcomes with HP-UX 11i
- What HP is providing: A series of technical on-demand training Webcasts
  - Focused on helping people who deploy HP-UX Integrity solutions increase performance through optimizing their installations for HP-UX 11i v3 on HP Integrity servers
  - Available at [www.hp.com/go/kod](http://www.hp.com/go/kod)

# HP-UX 11i v3 Knowledge-on-Demand webcasts – planned topics

- HP-UX 11i v3 operating system optimization
  - Dynamic nPartitions
  - HP-UX 11i v3 tunables
  - I/O optimization
  - System Management Homepage
  - Software Assistant
  - GlancePlus
  - Performance trouble-shooting on v3
- Optimizing high availability configurations
  - Serviceguard configuration and manageability
  - Configuring Serviceguard with Oracle RAC
  - Serviceguard delta training
- Optimizing virtualized configurations
  - Capacity advisor
  - Workload monitoring and management
  - System sizing with HP VM

Additional Webcasts to be published going forward!

Topics subject to change without notice.



# Related HP-UX 11i v3 resources

- All deployment resources
  - HP-UX 11i developers content  
[www.hp.com/go/hpuxdev](http://www.hp.com/go/hpuxdev)
  - HP-UX 11i v3 news, functionality, product download, and services resources  
[www.hp.com/go/hpux11i](http://www.hp.com/go/hpux11i)
  - HP Integrity server product information  
[www.hp.com/go/integrity](http://www.hp.com/go/integrity)

We hope you enjoy this  
Knowledge-on-Demand topic!

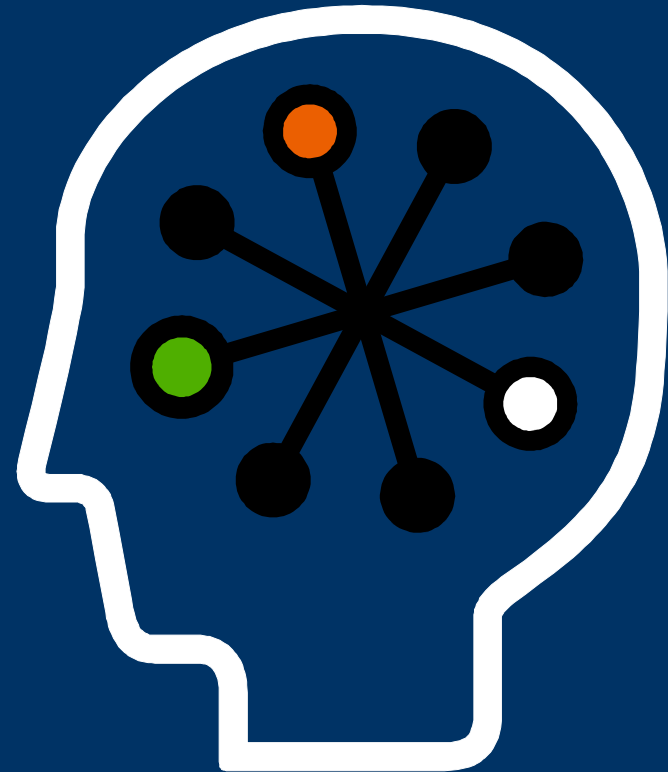
Thank you for taking time to learn about HP-UX 11i  
v3 and related technologies.

Please provide feedback on today's topic and/or  
future topics by using our online HP-UX 11i  
Knowledge-on-Demand Feedback form:

[www.hp.com/go/kodfeedback](http://www.hp.com/go/kodfeedback)

# SAP on HP-UX Itanium

Maximizing server performance



# Introducing the speaker

- Bob Wynne works in the performance section of the HP-UX kernel lab, EKSL. For the past 6 years his main responsibility has been optimizing SAP performance and publishing SAP benchmarks on HP-UX.
- Bob holds a PhD from the University of California, San Diego, and has worked on OS performance for over 20 years



# Scope of presentation

IS:

- Basic principles for optimizing performance of SAP on HPUX, particularly on large, cell-based systems.
- Based primarily on lab experiments, benchmarks

IS NOT:

- Tools for monitoring performance
- Techniques for sizing SAP applications
- Bottleneck analysis within SAP or HP-UX
- Database tuning

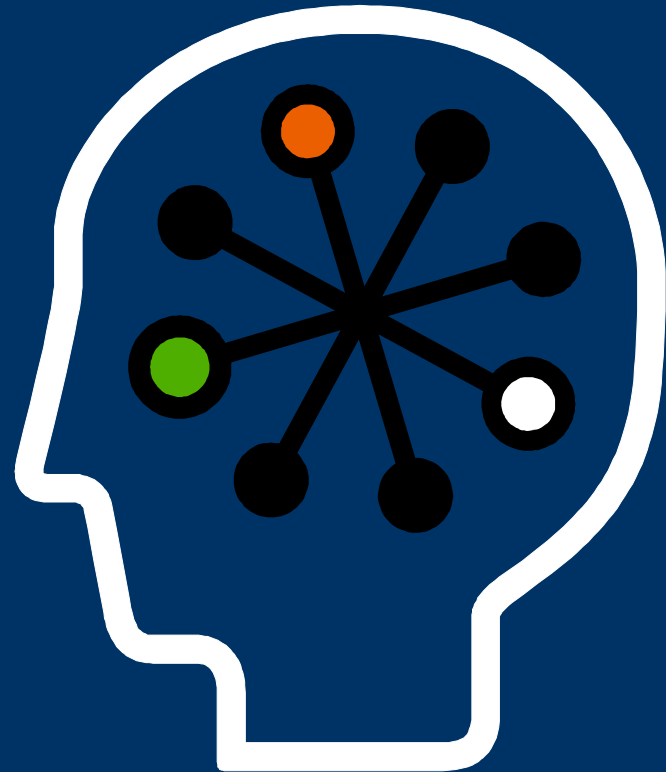
# Assumptions

- About the system to be optimized:
  - System already tuned to minimize disk I/O
  - Database properly tuned
  - Relatively high CPU utilization
- About the audience:
  - Current resource utilization is well-understood
  - Basic knowledge of SAP and HP-UX system administration and tuning
  - Basic understanding of computer architecture

# Topics for discussion

- Maximizing processor efficiency by reducing:
  - TLB misses
  - Cache misses
  - Memory latency
- Configuring and tuning SAP instances
- Tuning HP-UX for SAP
- LiveCache and other multi-threaded apps

# 3 keys to maximizing performance



# Minimize TLB misses

- Use large virtual pages for text and data
  - chatr +pi 16M +pd 64M disp+work

Page Size		ITLB Miss/Kinst	DTLB Miss/Kinst	% Cycles in handler	SAPS	RT	% Idle	Peak Memory Use
Instruction	Data							
4K	4K	0.974	2.27	7.44	8930	3.82	0	26.0G
16K	16K	0.142	0.635	3.68	10118	2.17	.8	26.5G
4M	16M	0.023	0.075	1.29	10777	1.42	.8	29.1G
64M	L (largest)	0.018	0.076	1.13	10788	1.40	2	33.5G

Test results for 2050 SD users on 8-core rx6600 with HT enabled, total of 36 SAP work processes plus Oracle 11g database with 4GB SGA

# Reducing cache misses

- Should have virtually no instruction cache misses running SAP
- Reduce dcache misses by
  - Balancing workload
  - Limiting and grouping of work processes
  - Using psets
    - psrset (1M)

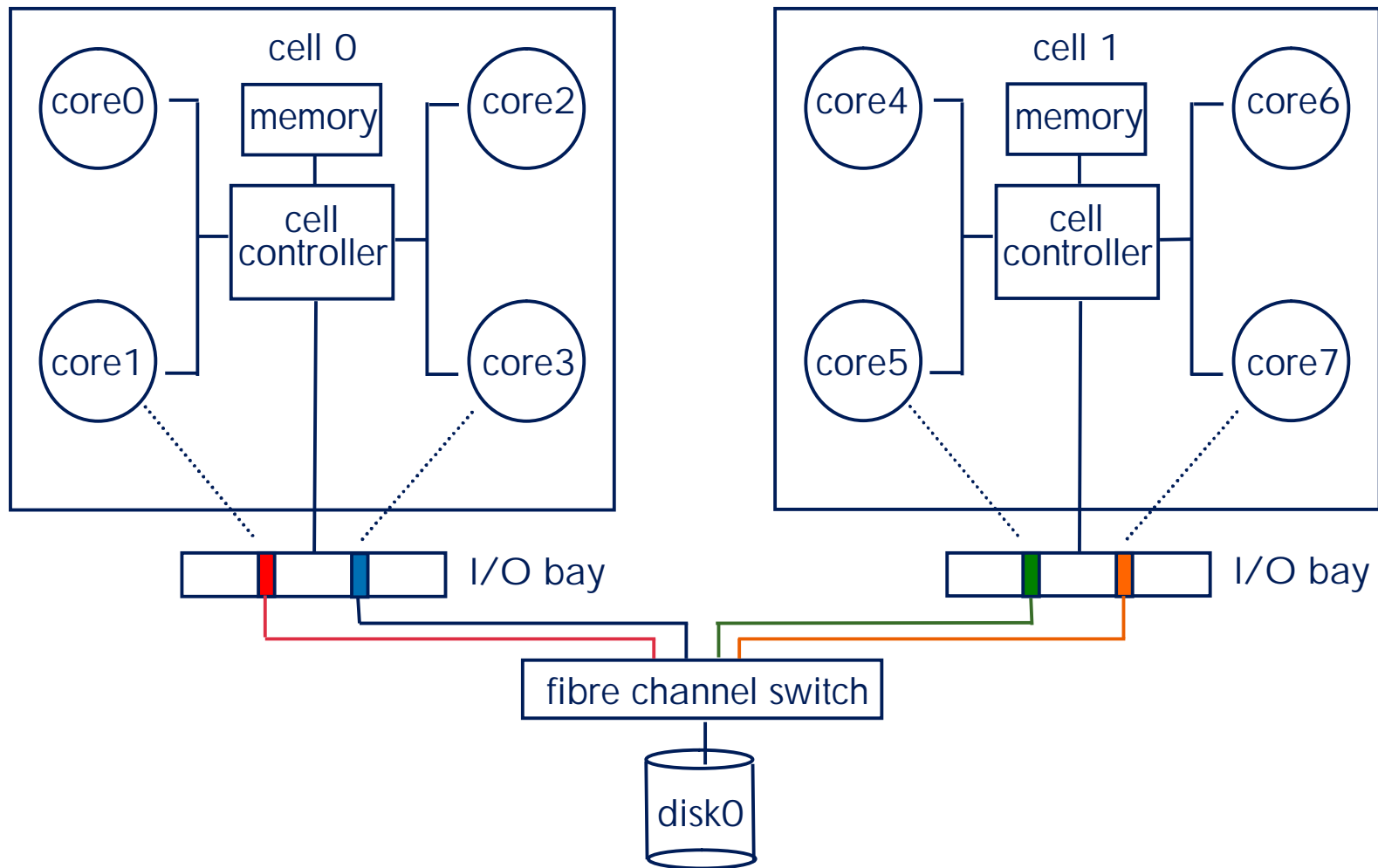
PSETS	I2 cache miss/Kinst	RT	CPU/DS (ms)	DB request time (ms)
YES	1.239	1.382	35.76	9.51
NO	1.419	1.702	36.57	11.46
delta	14.53%	23.15%	2.27%	20.50%

2050 SD Users on rx6600 w/HT enabled

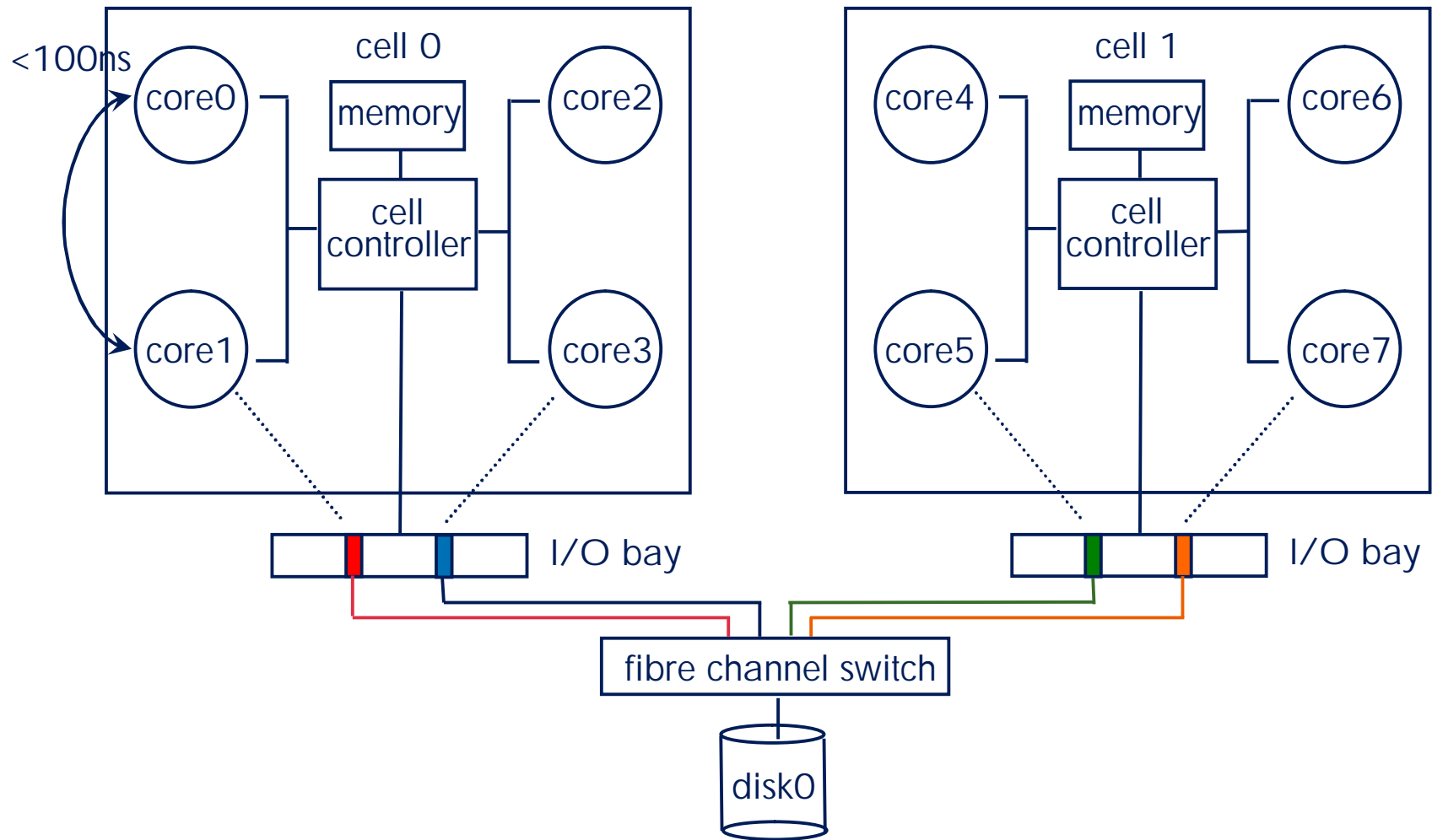
# Reducing memory latency

- Why is Cell Local Memory (CLM) important?
- Benchmark data
- Recommendations

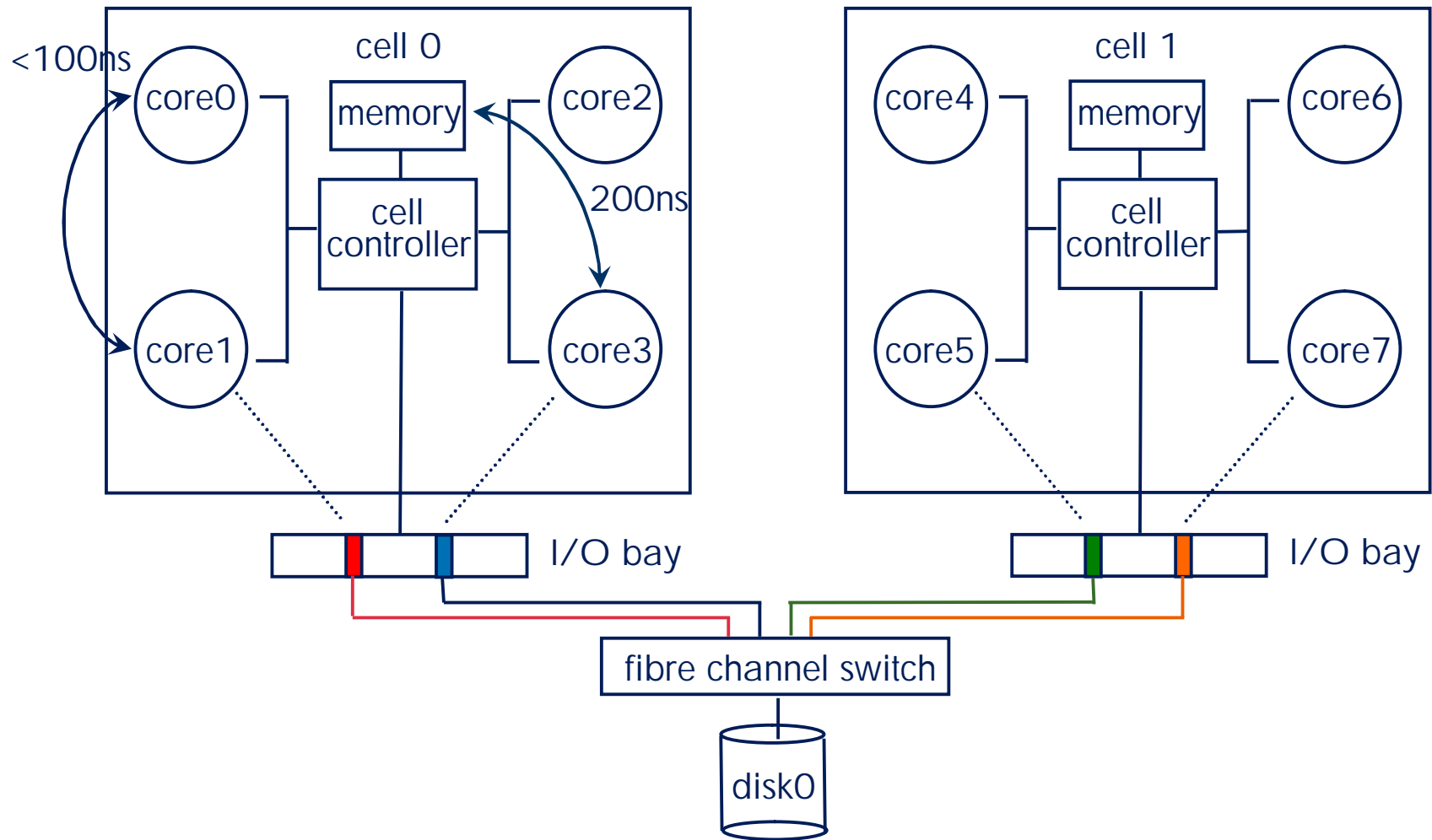
# Typical load latencies on multi-cell systems



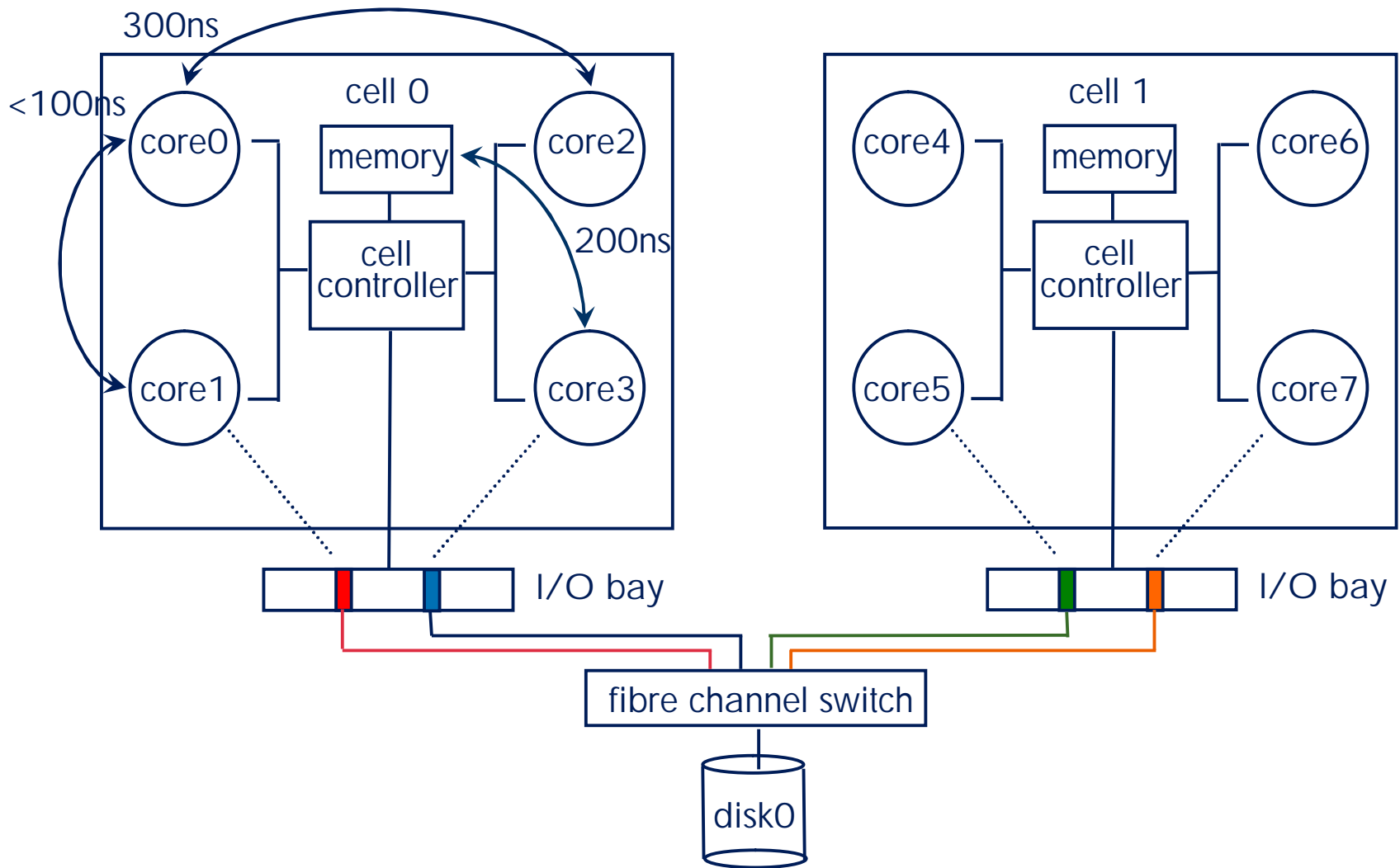
# Load latencies



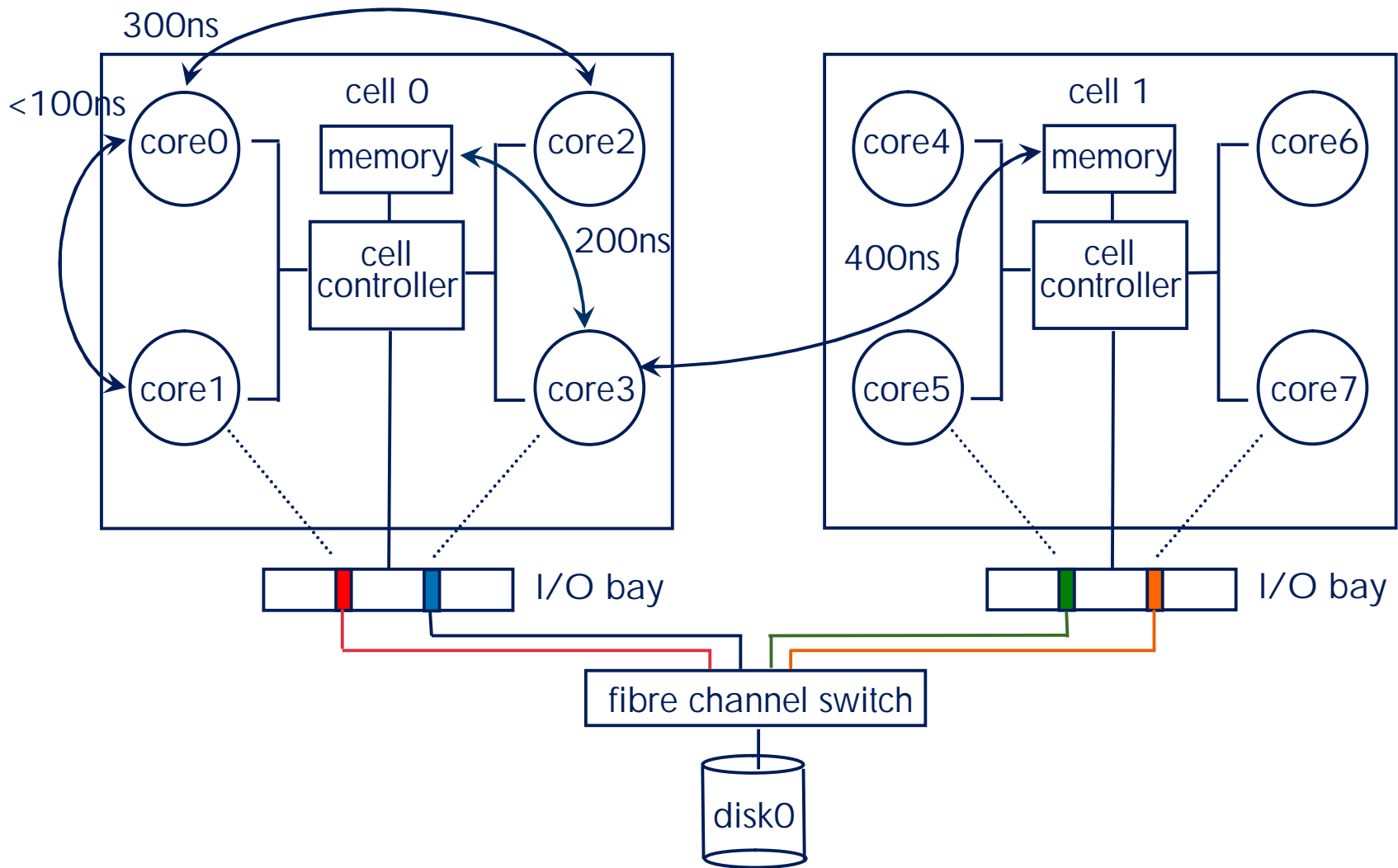
# Load latencies



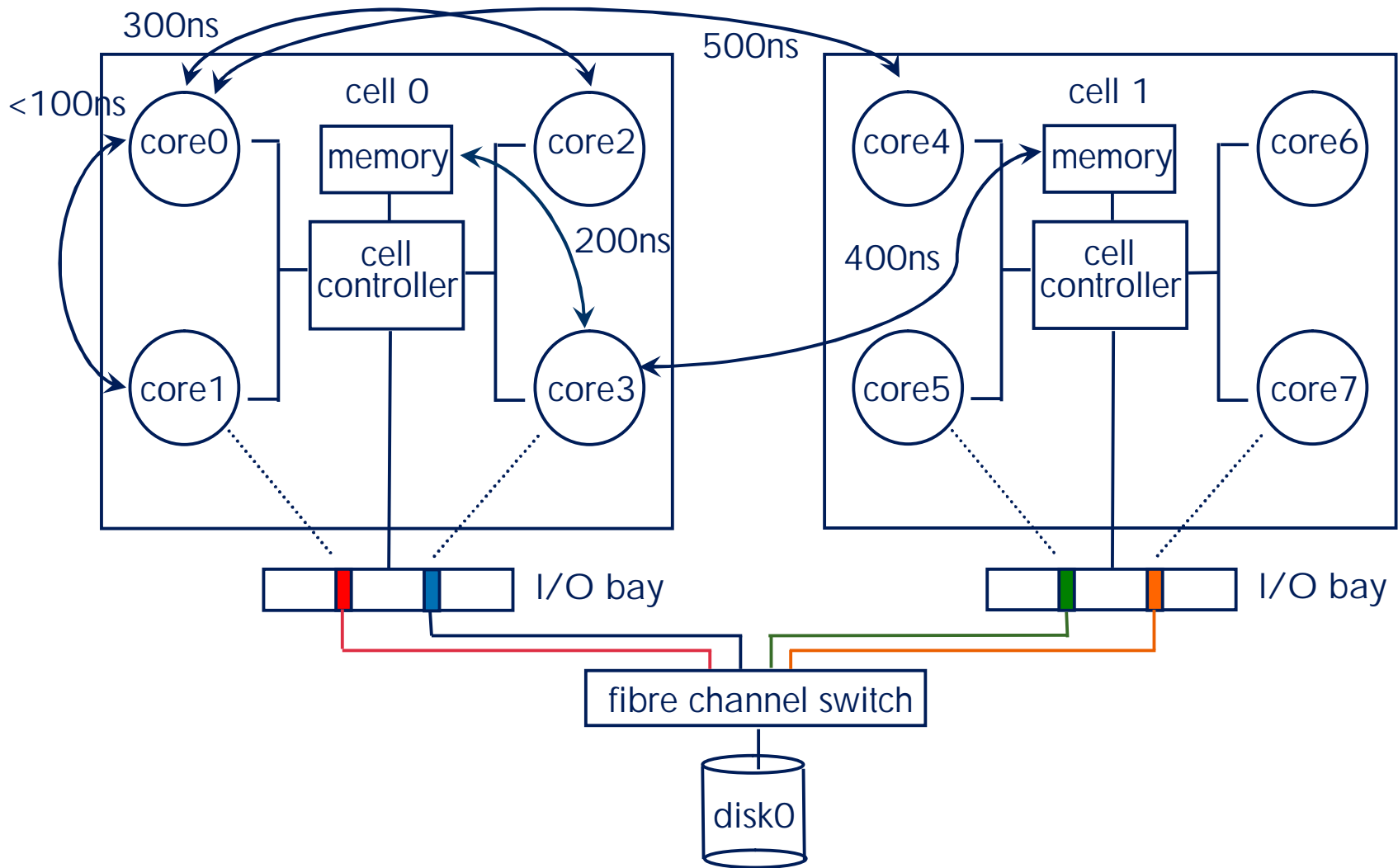
# Load latencies



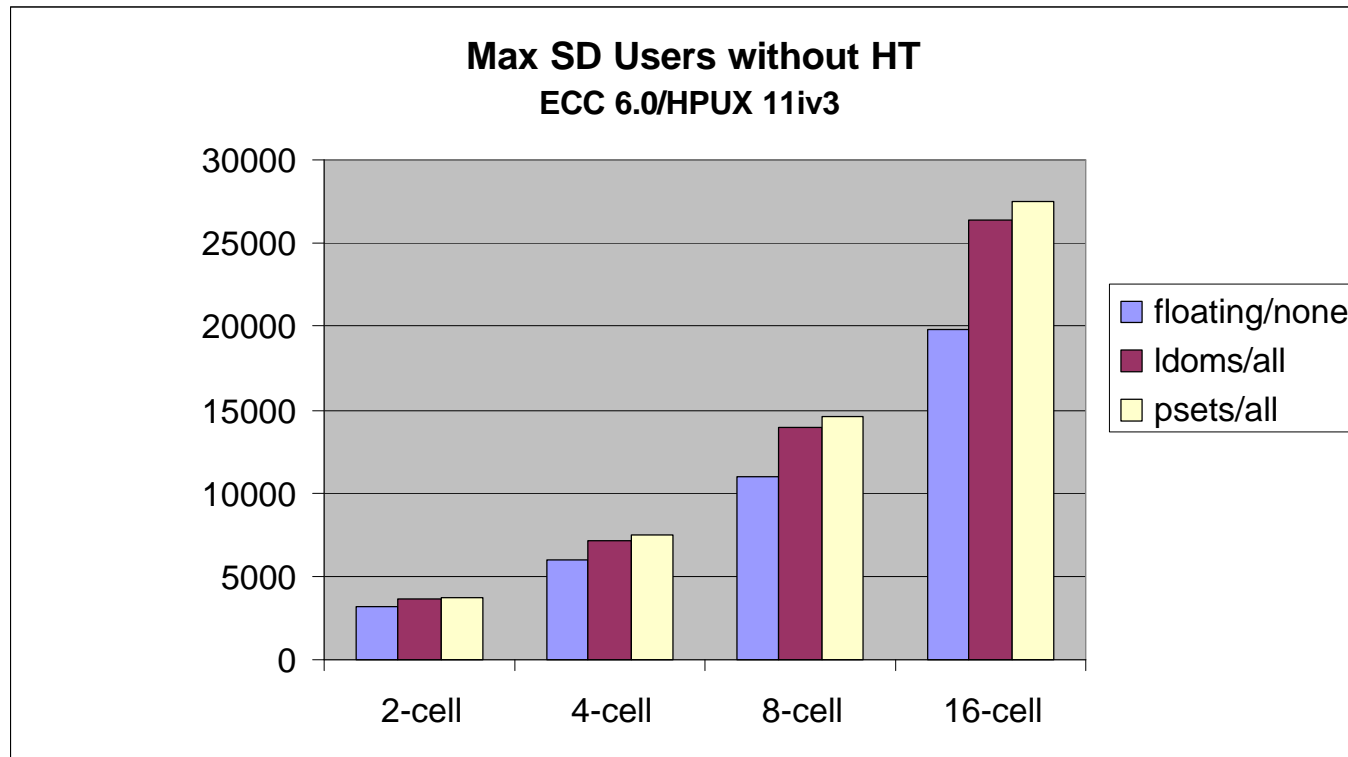
# Load latencies



# Load latencies

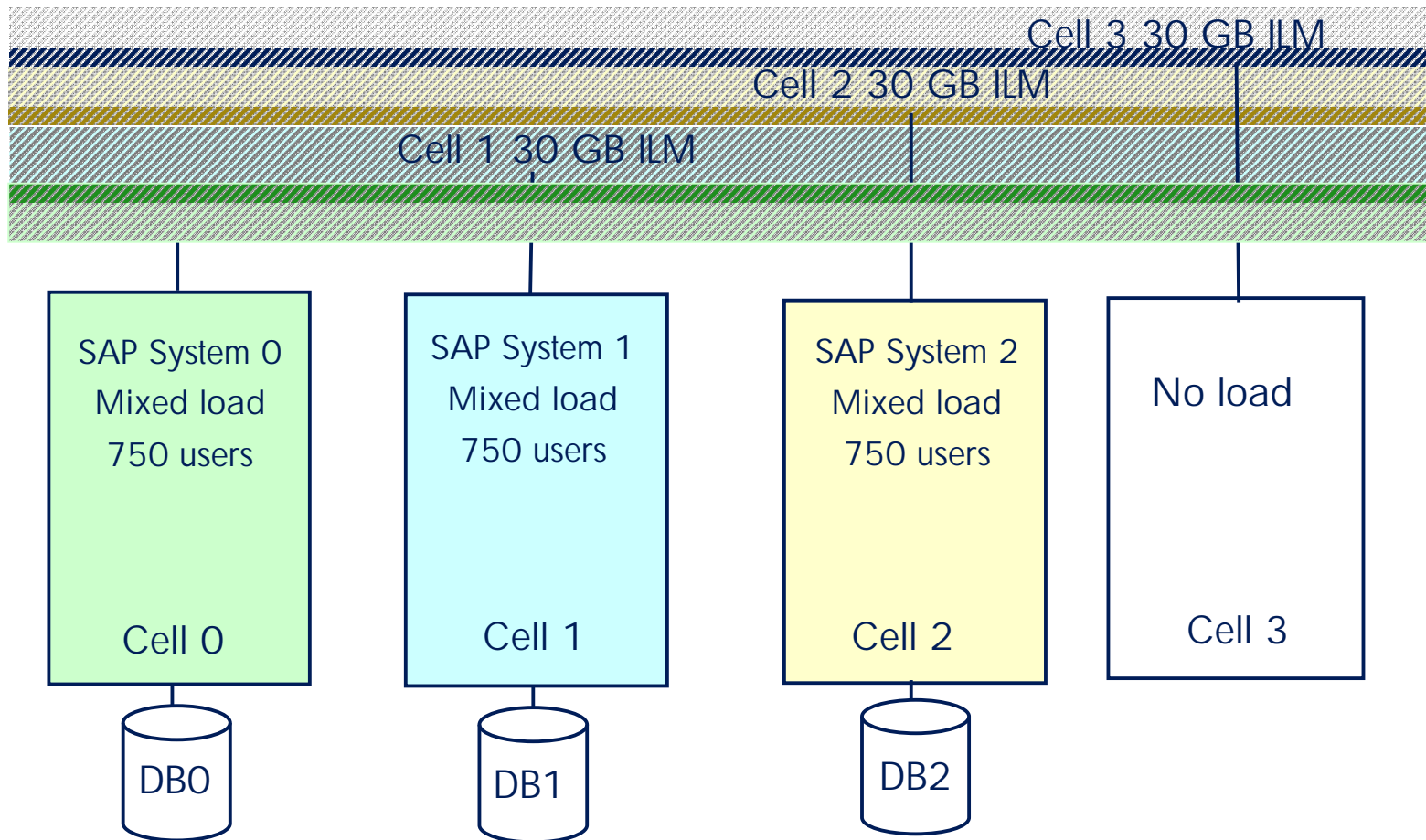


# Benefit of CLM on SD Benchmark



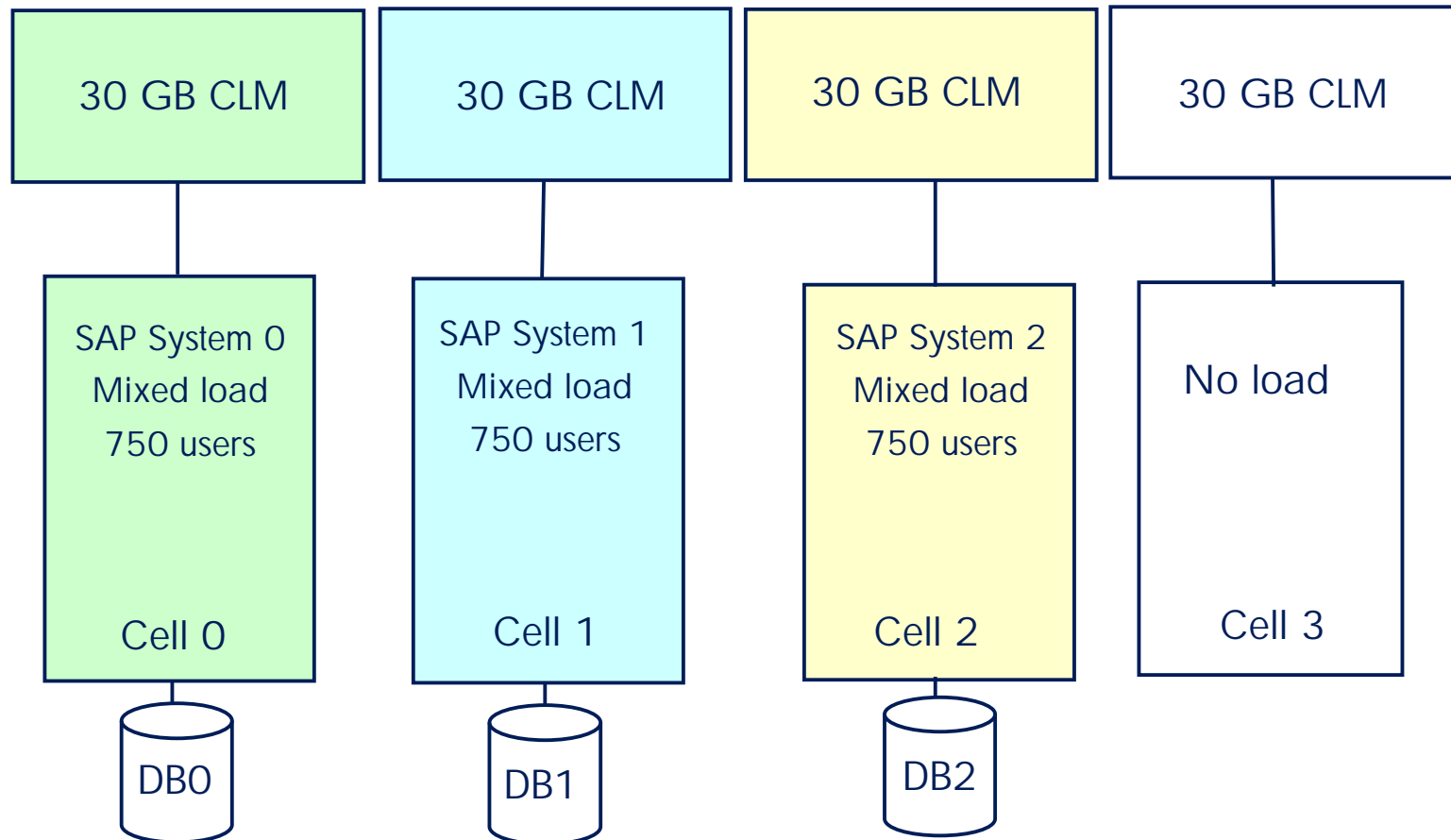
# vPars configuration one: 100% ILM

rx8640 120GB RAM



# vPars configuration two: 100% CLM

rx8640 120GB RAM

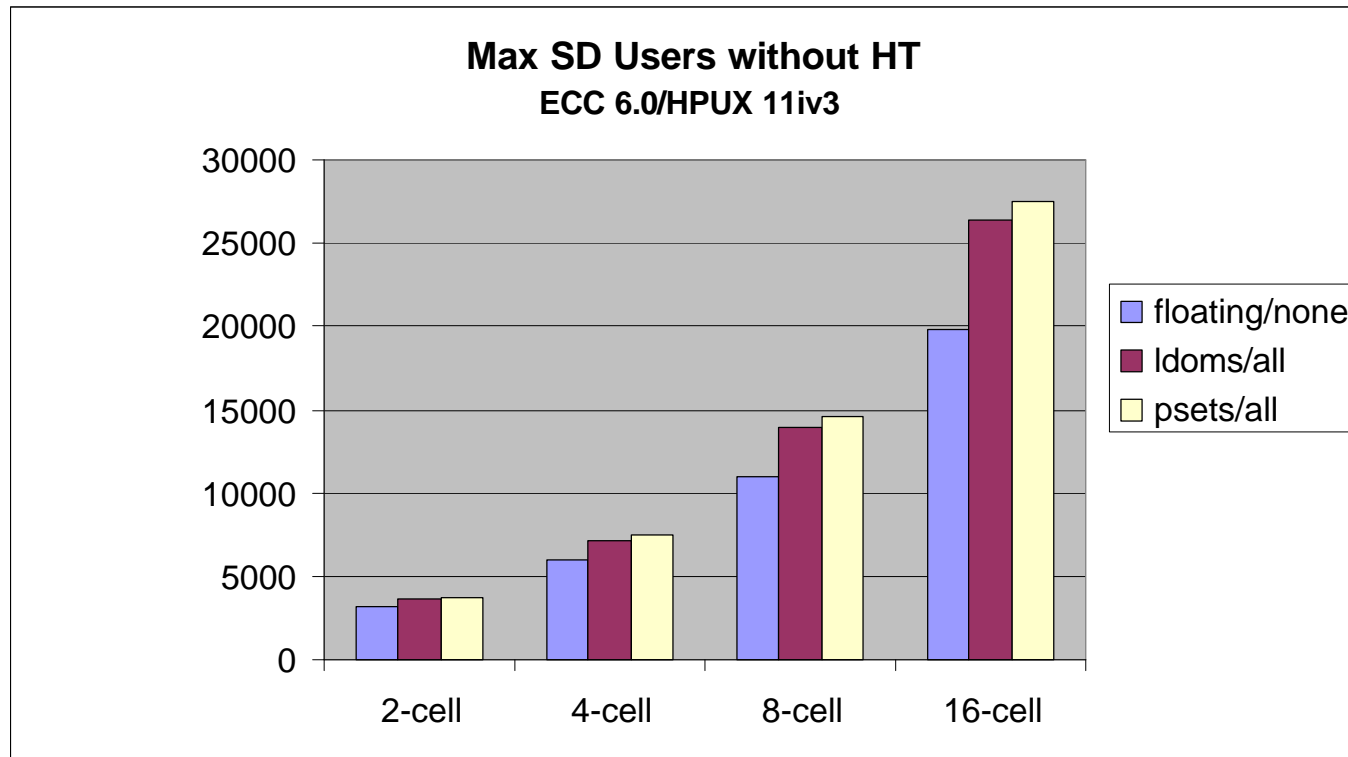


# SAP on vPars: CLM vs. ILM

Memory layout	Average Response time	DS/second	CPU/DS (ms)	CPU Utilization
100% ILM	1.15	67	89	94%
100% CLM	.27	73	72	72%

Results summary for 3 virtual partitions, each running 750 users, mixed SD, finance, MM, and PP workload on rx8640

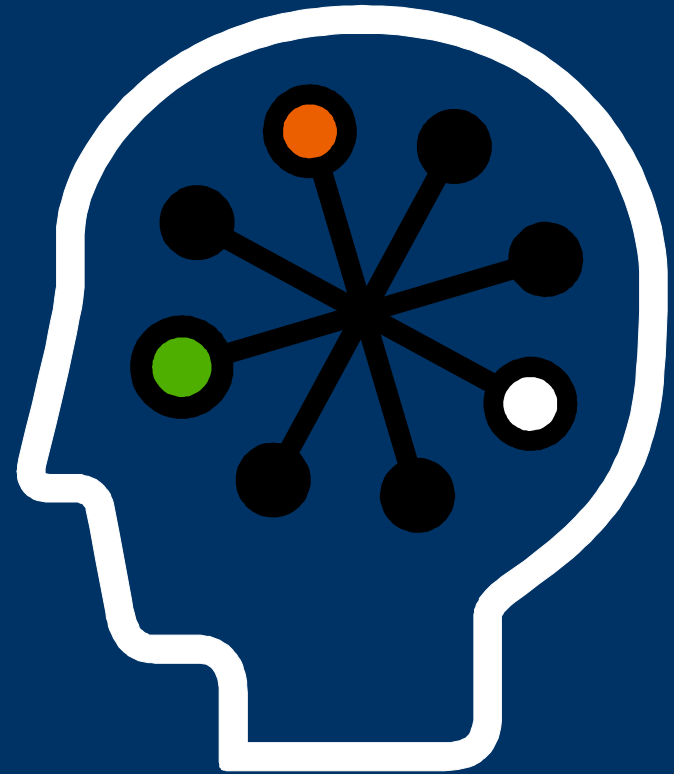
# Benefit of CLM on SD Benchmark



# CLM recommendations

- Always use some CLM on large systems
  - 10% CLM is enough for kernel data structures and process- private data
- Restrict SAP instances to cells by starting instances with `mpsched(1M)`
- Ideally, configure enough CLM for SAP shared memory (11iv3 and later)
- If database is on the same server, configure user threads or “shadow” processes to run in the same cell as corresponding SAP work processes

# Configuring SAP for performance



# Configuring SAP on a single server

- Segregate SAP from other applications using mpsched, psets, virtualization tools, etc.
- Configure sufficient instances
- Don't over-configure work processes

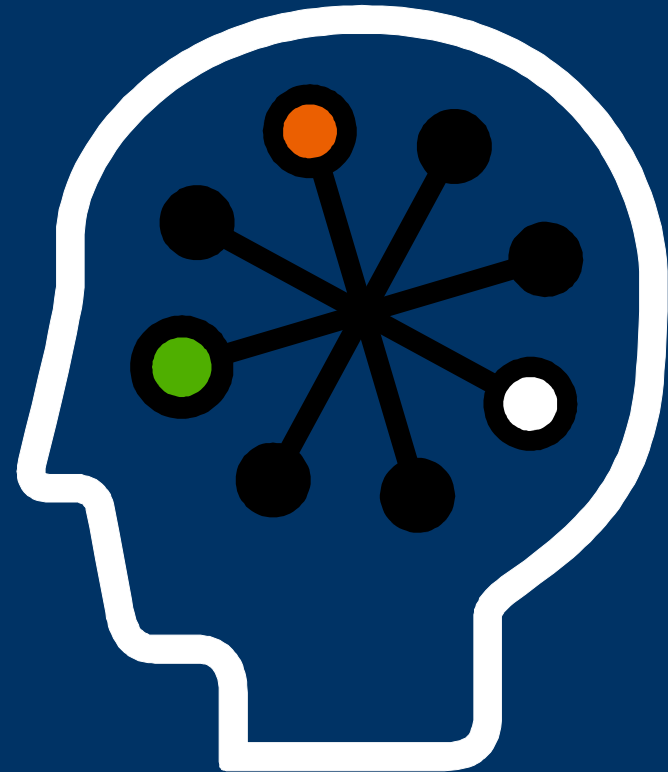
# How many SAP instances?

- Don't let dispatcher become a bottleneck. More smaller instances better than fewer larger.
- 1 instance for every 2 – 4 cores  
(1,500 – 2,500 SAPS)
- Keep it local! Don't span cells. Use mpsched(1)

# Configuring SAP instances

- How many work processes?
  - As few as possible!
  - Monitor CPU usage and dispatcher queue
- Keep updates on same cell as dialog WPs
- Profile parameters
  - abap/initrc\_degree = 0
  - stat/clock=sap\_clock

# Tuning HP-UX for performance



# HP-UX kernel settings

- Follow recommendations in sapnote 172747
- Don't over configure buffer cache
  - 3% of memory should be more than sufficient
- Configure adequate swap space
  - At least 2X memory
- If possible, use CLM for SAP shared memory:
  - `kctune numa_policy=1` (HP-UX 11iV3 only)

# LiveCache and other multi-threaded applications

- Use the latest version of HP-UX.
  - 11iv2 fixes many pthreads problems
  - 11iv3 and 0803 fusion contain additional improvements
- Set environment variables:
  - PTHREAD\_FORCE\_SCOPE\_SYSTEM=1
  - PERF\_ENABLE=1
- If on 11iv2, apply patches
  - PHCO\_36323 and PHKL\_36826

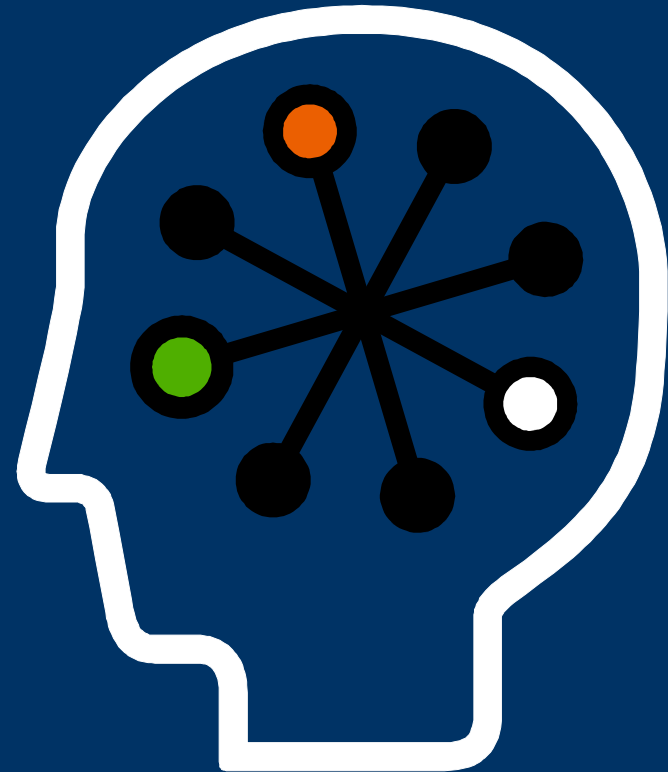
# Summary

- Understand your workload! Monitor performance and resource utilization.
- Minimize TLB misses with large pages
- Minimize cache misses by limiting migration
- Minimize memory latency through CLM
- More instances with fewer SAP work processes is preferable to fewer instances with more WPs

# Additional resources

- Schneider, Thomas, **SAP Performance Optimization Guide**, SAP Press, 2003, ISBN 1592290221
- Sauers, Robert F., Ruemmler, Chris P., Weygant, Peter S., **HP-UX 11i Tuning and Performance**, Prentice Hall, 2004, ISBN 0131433490
- Robert Wynne and Patricia Coon, "Using HP-UX features to enhance the performance and scalability of an HP Integrity Superdome SD64B server running SAP"  
<http://h71028.www7.hp.com/ERC/downloads/4AA1-3728ENW.pdf>

# Backup Slides



# Backup slide one: More on TLB and large pages

On Montecito processors, 256 L1 TLB entries (128 instruction + 128 data)  
4K, 8K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M, 1G, 4G page sizes supported

Can monitor TLB and cache efficiency with caliper.

For example, to collect system-wide cpu statistics for 2 minutes:

```
# caliper cpu -o calout -w -duration=12
```

Excerpt from file `calout` (collected during run with 4K page hints for disp+work):

## TLB Summary Statistics

Stats	-----Misses Per Second-----				-----Misses Per Kinst-----				%ITLB	%DTLB
	I1TLB Miss	I2TLB Miss	D1TLB Miss	D2TLB Miss	I1TLB Miss	I2TLB Miss	D1TLB Miss	D2TLB Miss	H/W update	H/W update
MEAN	3399768	984953	5958040	2254450	3.389	0.974	5.983	2.227	98.53%	74.23%
STDEV	167096	111431	431503	267378	0.202	0.075	0.692	0.193	0.82%	16.18%

Should have < .5% cpu cycles in TLB handler

D2TLB Miss rate < .1 per thousand instructions

# Backup slide two: More on reducing cache misses

Use `psrset(1M)` to create and manage processor sets. For example to create a processor set on CPUs 2 and 3 and then launch SAP instance D00 on this processor set:

As root:

```
# psrset -c 2 3
successfully created pset 1
successfully assigned processor 2 to pset 1
successfully assigned processor 3 to pset 1
```

As <sid>adm:

```
sp5019:sdbadm 4>/usr/sbin/psrset -e1 startsap r3 D00
```

Use `mpsched` to bind a process to a CPU. For example, to restrict the VXFS daemon to CPU 14:

```
# ps -ef | grep vxfsd
root  70  0 0 10:24:18 ?      0:28 vxfsd
```

```
# mpsched -c 14 -p 70
```

Pid 70: bound to processor 14 using the default process launch policy

## Tools:

caliper (dtlb, itlb miss rates)

pi (tlb miss rate, cpi)

vpsinfo (per process info on page sizes)

# Backup slide three: More on CLM

- The best latency is for a cache-to-cache copy between cores on the same front-side bus (around 85ns)
- The next lowest latency is a miss to memory on the same cell as the core requesting the memory (< 200 ns)
- A two hop memory miss is an additional 20% more expensive than one hop.
- Cache to cache misses for larger configurations can be even more costly (up to nearly 700 ns).
- Use the parmodify(1M) command to create cell local memory. Creation of CLM requires a reboot of the system.
- Command `mpsched -s` will display all locality domains and their corresponding CPUs.
- Start SAP instances using the mpsched command and PACKED policy. For example, to start instance D02 of SAP in cell (locality domain) one:
  - `mpsched -l 1 -P PACKED startsap r3 D02`
- To allocate all shared memory for SAP instances in CLM, change default NUMA policy using `kctune(1M)` before starting SAP:
  - `kctune -b no numa_policy=1`

# Backup slide four: CLM Experiment 2: SAP on vPars

- For this experiment, the rx8640 was divided into 4 separate vPars, each running HP-UX 11iV3. Each virtual partition was assigned 4 dual-core Intel® Itanium 2 9000 series processors (1.6GHz, 24MB), and 30 GB memory. The unicode version of SAP ECC 6.0 was installed on each system, using an Oracle 10g database. Hyper-threading was NOT enabled. Two instances of SAP were created on each partition.
- For the first test, the memory was interleaved across all cells. In the second test, each virtual partition was allocated 30GB of cell local memory.
- A mixed SD-FIN-MM-PP benchmark was run in parallel in the three of the 4 virtual partitions. As with the SD benchmark, there is a 10 second think time between dialog steps, so the response time and dialog steps per second figures are related, and DS/second is limited to 75 on each system with this load of 750 users.
- A good metric for performance comparison is the average dialog work process CPU usage per dialog step. This shows a close to 25% performance benefit for CLM vs. ILM. This is consistent with the 4-cell result on the Superdome system, where 7450 users could be supported with sub-2 second response time using CLM and psets, an increase of 25% above the maximum of 5960 users with ILM and default scheduling parameters.

# Backup slide five: More on configuring SAP instances

monitor dispatcher queues with dpmon or transaction sm51.

monitor memory usage with transaction st02

monitor response time with stad or st03

add work processes only when response time declines and dispatcher queues grow and there are idle cpu cycles and memory headroom. Otherwise, additional work processes only increase resource utilization and slow things down further!

parameter abap/initrc\_degree is described in sapnote 559874

# Backup slide six: More on configuring SAP instances

monitor dispatcher queues with dpmon or transaction sm51.

monitor memory usage with transaction st02

monitor response time with stad or st03

add work processes only when response time declines and dispatcher queues grow and there are idle cpu cycles and memory headroom. Otherwise, additional work processes only increase resource utilization and slow things down further!

parameter abap/initrc\_degree is described in sapnote 559874

# Backup slide seven: More on multithreaded applications

Except in very rare cases, MxN threads hurts performance. Explicitly setting scope to “system” saves significant kernel overhead.

pthreads cumulative patch PHCO\_36323 can greatly improve performance of conditional variables.

PHKL\_36826 greatly speeds up sleep/wakeup for all pthreads operations.

Recent LiveCache testing on 64-core PA superdome have shown that 11iv3 can support a 100X higher thread switch rate than earlier versions of HP-UX.

Technology for better business outcomes

