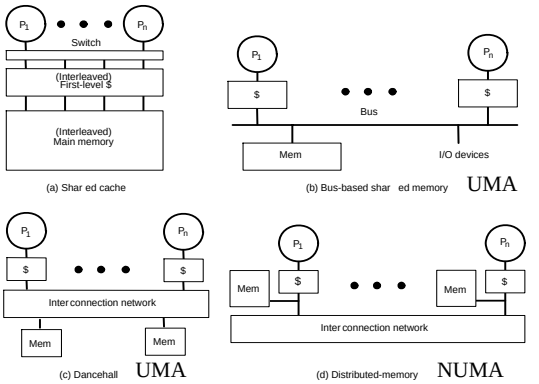
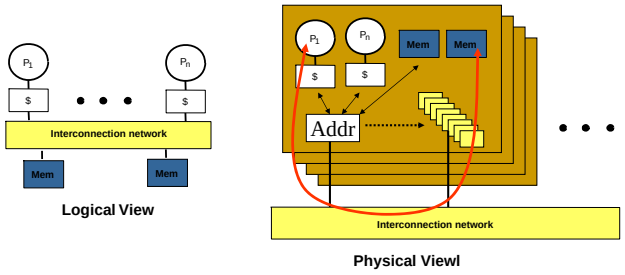
Sun's E6000 Server Family

Erik Hagersten
Uppsala University
Sweden

What Approach to Shared Memory

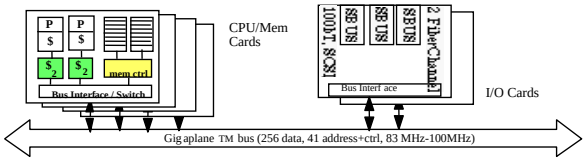


Looks like a NUMA but drives like a UMA



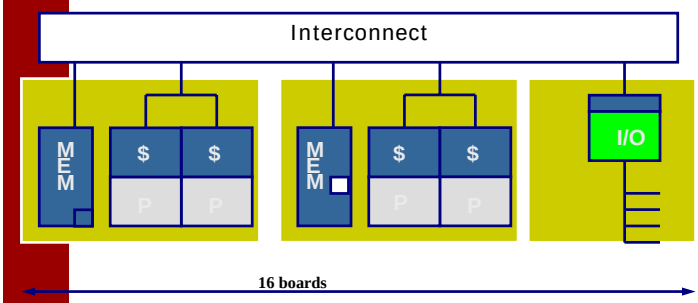
- Memory bandwidth scales with the processor count
- One "interconnect load" per (2xCPU + 2xMem)
- Optimize for the dancehall case (no memory shortcut)

SUN Enterprise Overview

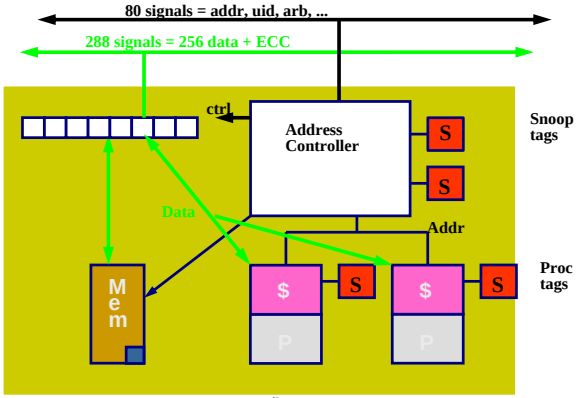


- 16 slots with with either CPUs or IO
- Up to 30 UltraSPARC processors (peak 9 GFLOPs)
- Gigaplane™ bus has peak bw 2.67 GB/s; up to 30GB memory
- 16 bus slots, for processing or I/O boards

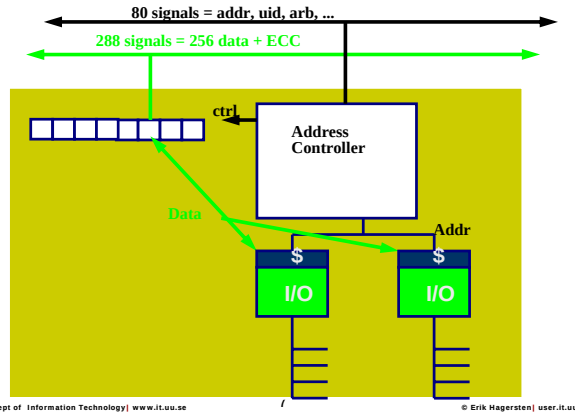
Enterprize Server E6000



An E6000 Proc Board

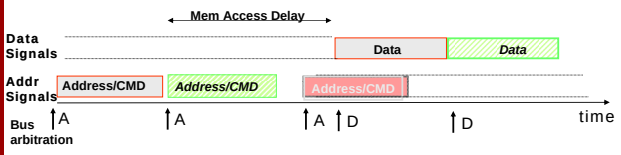


An I/O Board

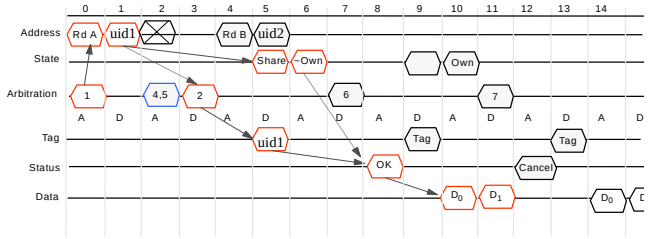


Split-Transaction Bus

- Split bus transaction into request and response sub-transactions
  - Separate arbitration for each phase
- Other transactions may intervene
  - Improves bandwidth dramatically
  - Response is matched to request
  - Buffering between bus and cache controllers



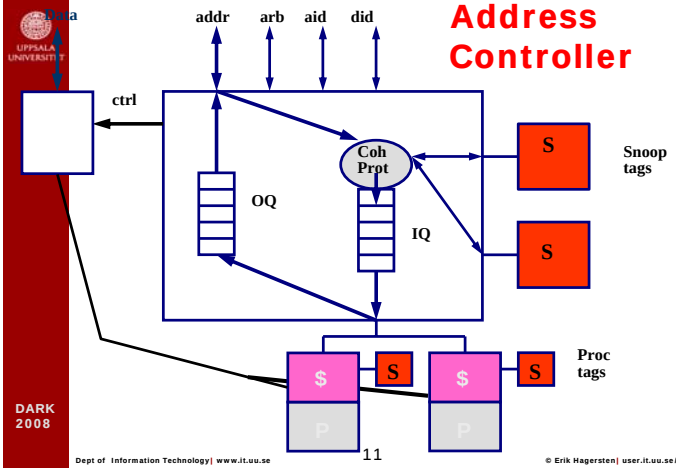
Gigaplane Bus Timing



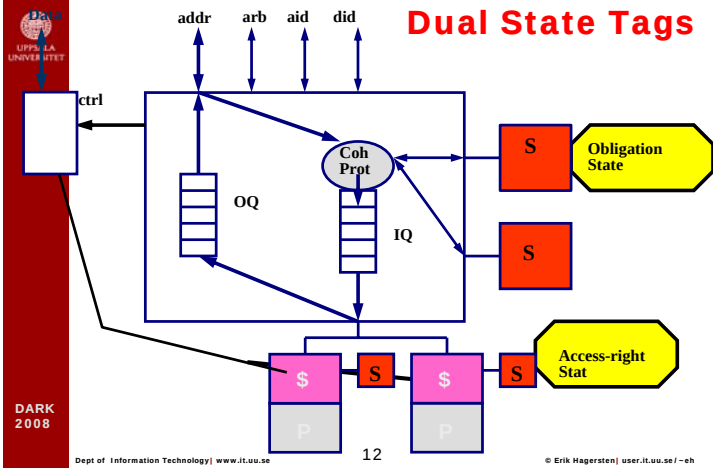
Electrical Characteristics of the Bus

- At most 16 electrical loads per signal
- 8 boards from each side (ex. 15 CPU+1 I/O)
- 20.5 inches "centerplane"
- Well controlled impedance
- ~350-400 signals
- Runs at 90/100 MHz

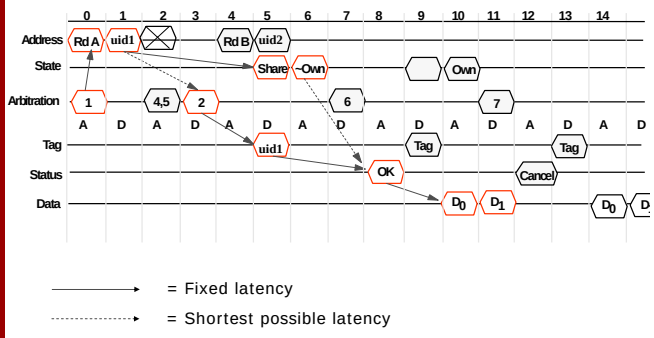
Address Controller



Dual State Tags



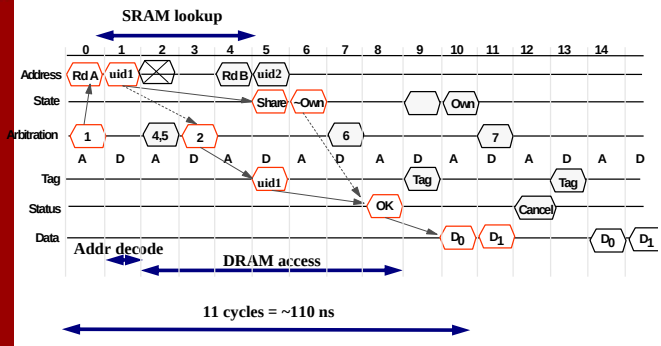
Timing of a single read trans
Board 1 reading from mem 2



= Fixed latency

= Shortest possible latency

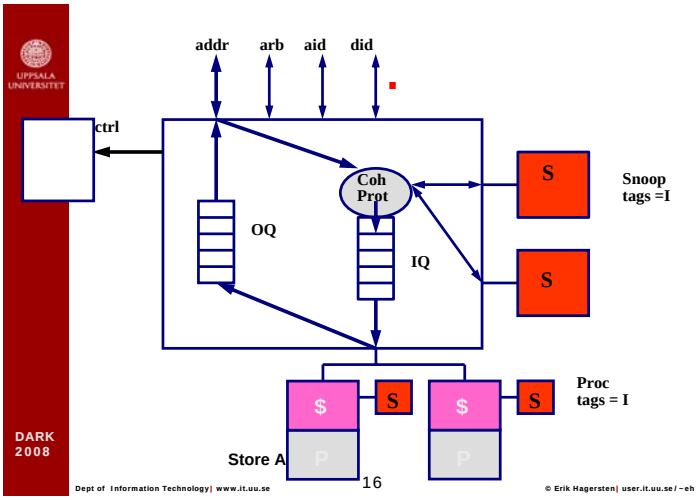
Protocol tuned for timing

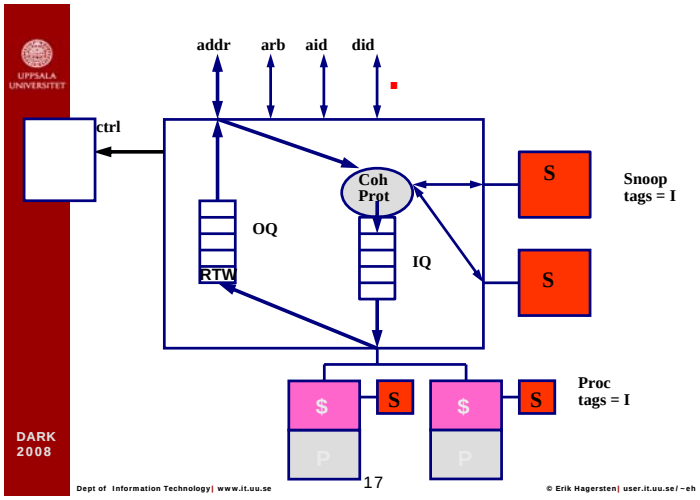


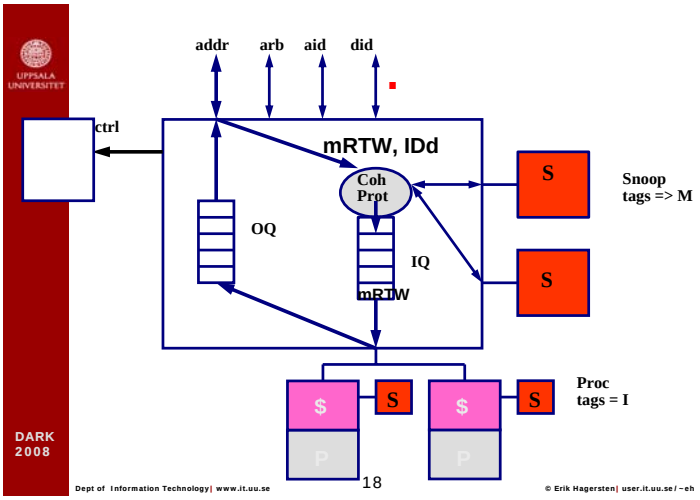
Foreign and own transactions queue in IQ
State Change on Address Packet

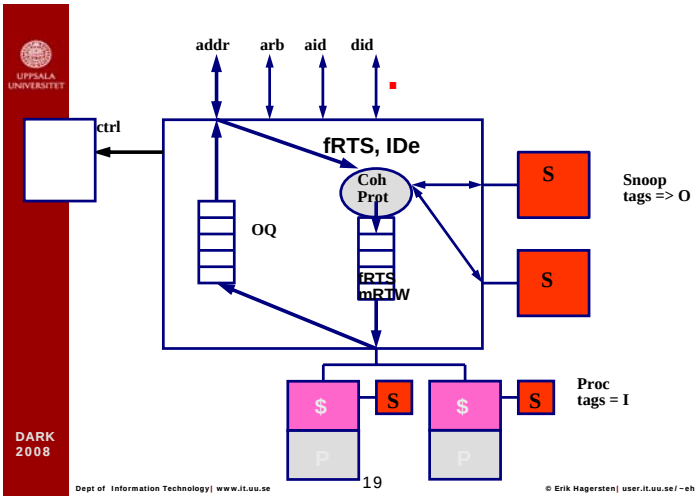
- Data "A" initially resides in CPU7's cache
- CPU1: Issues a store request to "A"
- CPU1: Read-To-Write req, ID=d, (i.e., "write request")
- CPU13: LD "A" -> Read-To-Shared req, ID=e
- CPU15: ST "A" -> RTW req , ID=f

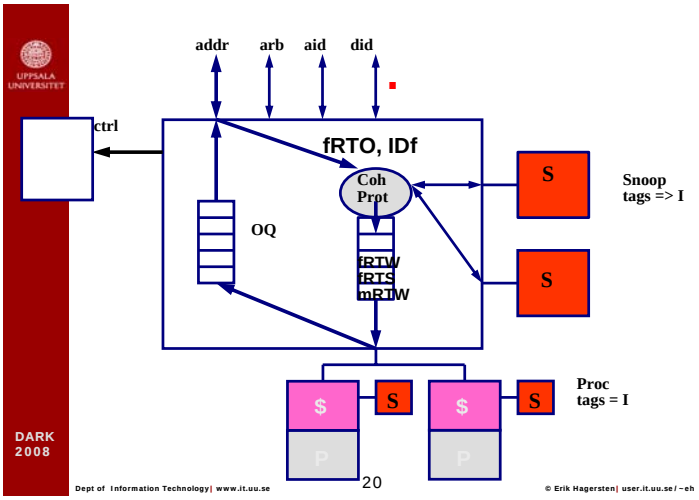
mRTO stored in IQ_{CPU1}
Own read IQtrans retired when data arrives
Later requests for A queued in IQ_{CPU1} behind mRTO
IQ_{CPU1} will eventually store: <$mRTW_{IDd}$, $fRTS_{IDe}$, $fRTW_{IDf}$>

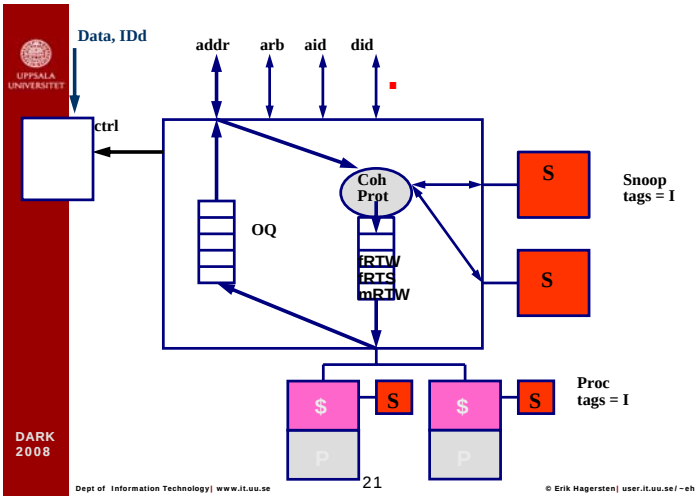


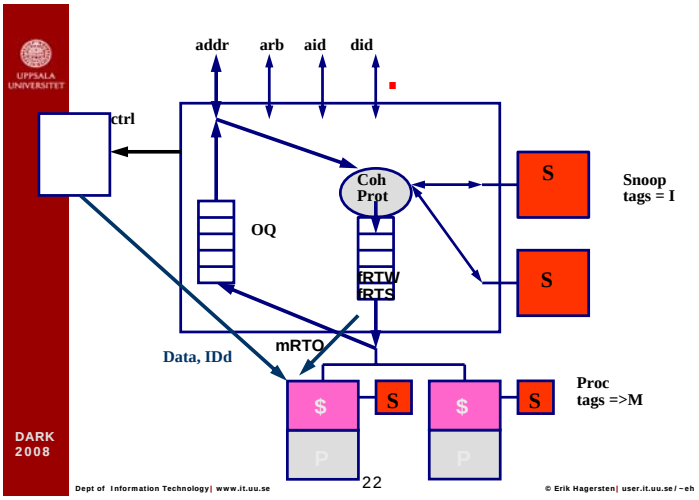


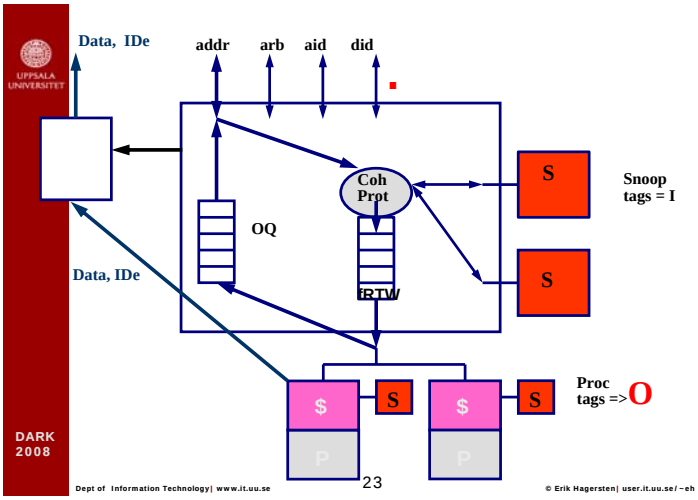


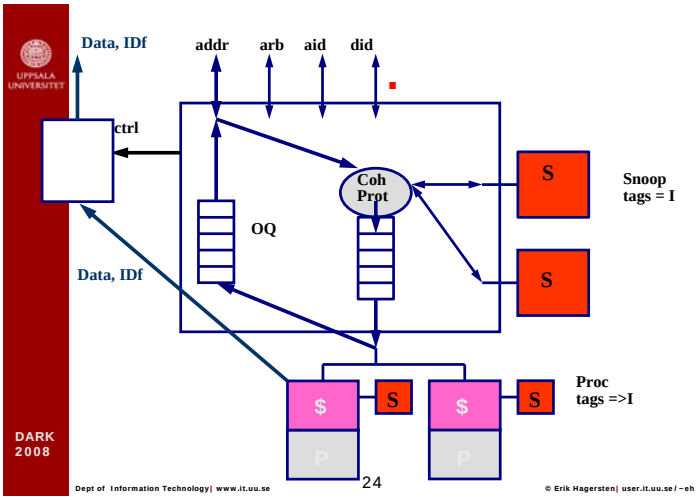

addr
arb
aid
did
ctrl
fRTS, IDe
Coh Prot
OQ
fRTS
mRTW
S
Snoop tags => O
$
P
Proc tags = I
DARK 2008
Dept of Information Technology| www.it.uu.se
19
© Erik Hagersten| user.it.uu.se/~eh


addr
arb
aid
did
ctrl
fRTO, IDf
Coh Prot
OQ
fRTW
fRTS
mRTW
S
Snoop tags => I
$
P
Proc tags = I
DARK 2008
Dept of Information Technology| www.it.uu.se
20
© Erik Hagersten| user.it.uu.se/~eh


Data, IDd
addr
arb
aid
did
ctrl
Coh Prot
OQ
fRTW
fRTS
mRTW
S
Snoop tags = I
$
P
Proc tags = I
DARK 2008
Dept of Information Technology| www.it.uu.se
21
© Erik Hagersten| user.it.uu.se/~eh


addr
arb
aid
did
ctrl
Coh Prot
OQ
fRTW
fRTS
mRTO
Data, IDd
S
Snoop tags = I
$
P
Proc tags =>M
DARK 2008
Dept of Information Technology| www.it.uu.se
22
© Erik Hagersten| user.it.uu.se/~eh


Data, IDe
addr
arb
aid
did
Coh Prot
OQ
fRTW
Data, IDe
S
Snoop tags = I
$
P
Proc tags => O
DARK 2008
Dept of Information Technology| www.it.uu.se
23
© Erik Hagersten| user.it.uu.se/~eh


Data, IDf
addr
arb
aid
did
ctrl
Coh Prot
OQ
Data, IDf
S
Snoop tags = I
$
P
Proc tags =>I
DARK 2008
Dept of Information Technology| www.it.uu.se
24
© Erik Hagersten| user.it.uu.se/~eh

A cascade of "write requests"

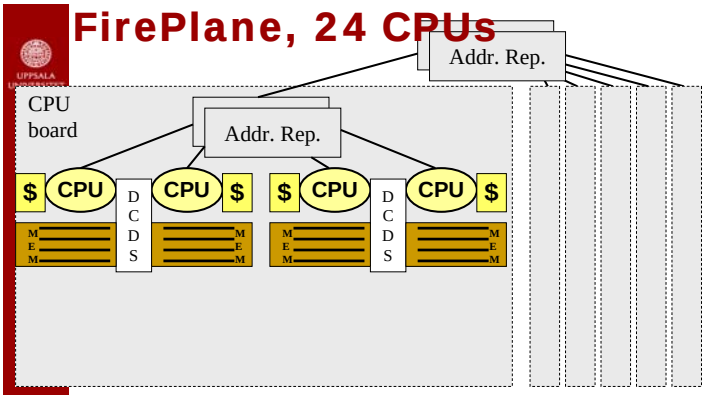
- **A initially resides in CPU7's cache**
- **CPU1: RTW, ID=a**
- **CPU2: RTW, ID=b**
- **…**
- **CPU5: RTW, ID=f**

CPU tags		Snoop tags
I	**IQ1 = <mRTW$_{IDa}$, fRTW$_{IDb}$>**	I
I	**IQ2 = <mRTW$_{IDb}$, fRTW$_{IDc}$>**	I
	…	
I	**IQ5 = <mRTW$_{IDf}$>**	M
	…	
S	**IQ7= < fRTW$_{IDa}$>**	I

Implementing Sun's SunFire 6800

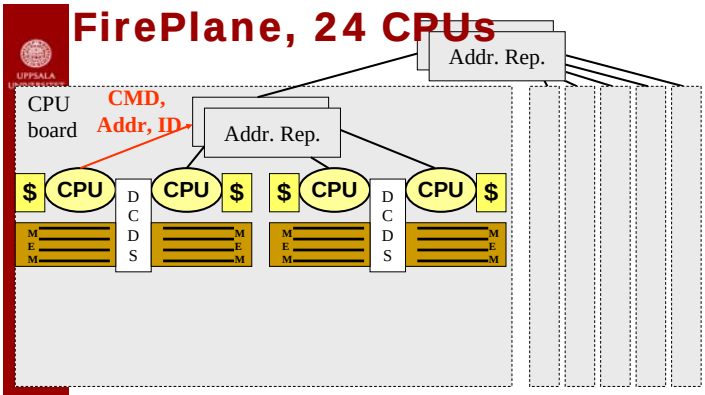
Erik Hagersten
Uppsala University
Sweden

FirePlane, 24 CPUs



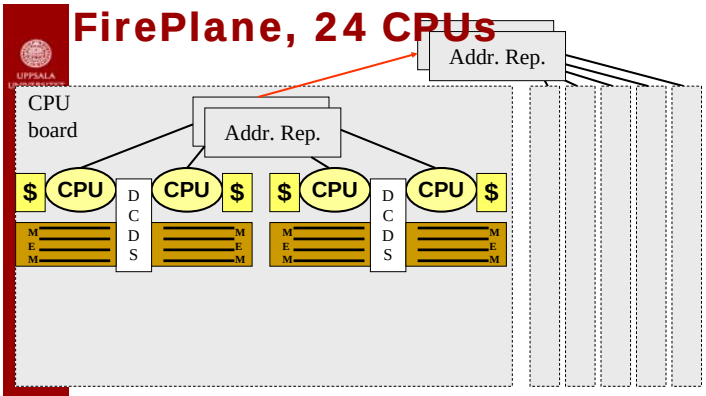
L2 cache = 8MB, snoop tags on-chip
CPU 1+GHz UltraSPARC III
Mem= 4+GB/CPU

FirePlane, 24 CPUs

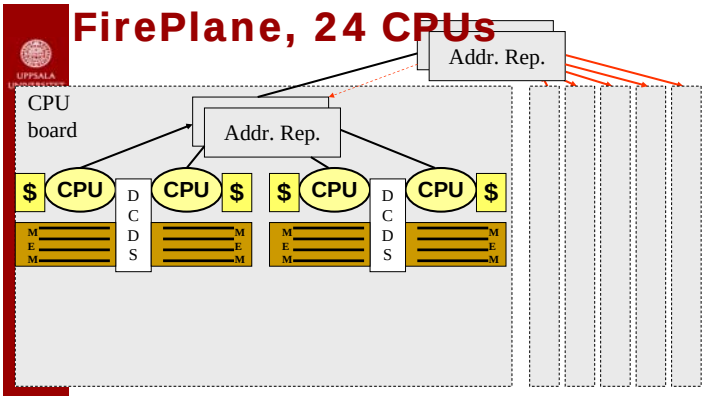


ID = <CPU#, Uid>

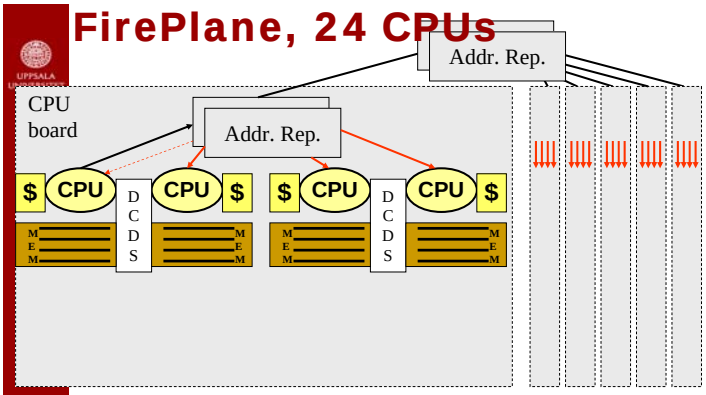
FirePlane, 24 CPUs



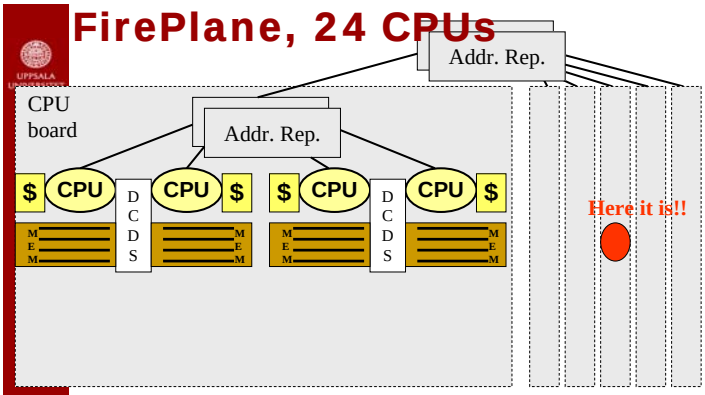
FirePlane, 24 CPUs



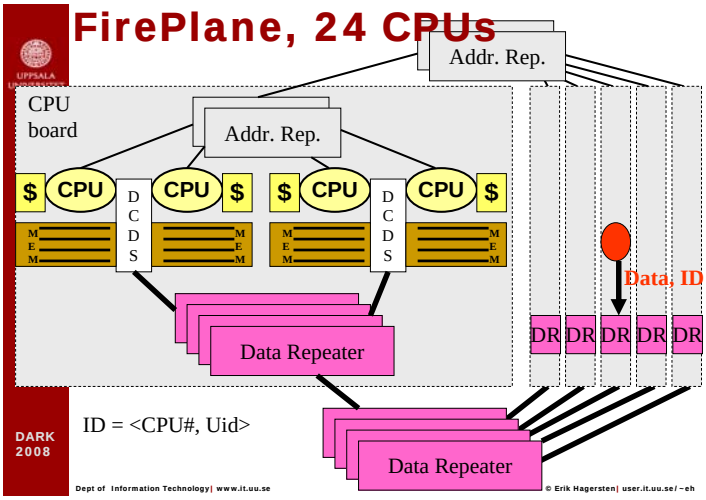
FirePlane, 24 CPUs



FirePlane, 24 CPUs



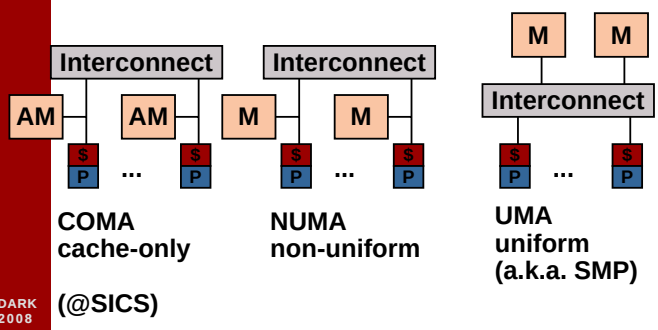
FirePlane, 24 CPUs



Scalable Shared-Memory Implementations

Erik Hagersten
Uppsala University
Sweden

Three options



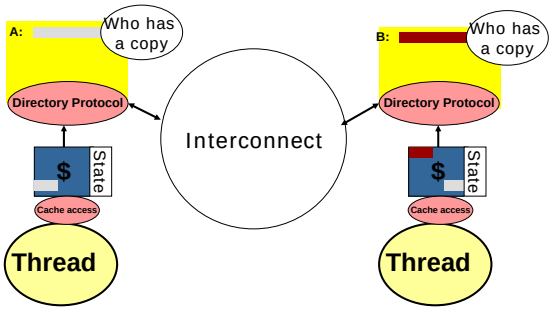
COMA
cache-only

NUMA
non-uniform

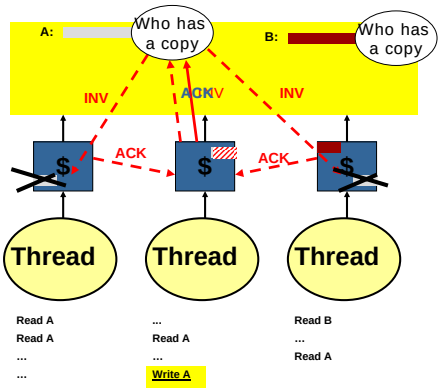
UMA
uniform
(a.k.a. SMP)

(@SICS)

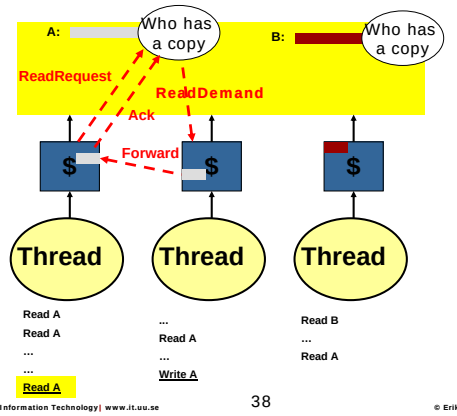
Directory-based snooping: NUMA. Per-cachline info in the home node



”Upgrade” in dir-based

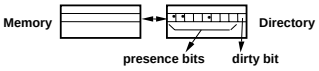


Cache-to-cache in dir-based



Fully mapped directory

”

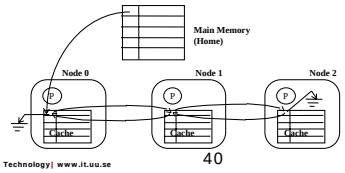


- *k* Nodes
- Each node is the ”home” for 1/*k* of the memory
- Dir entry per cacheline in home memory: *k* presence-bits + 1 dirty-bit
- Requests are first sent to the home node’s CA

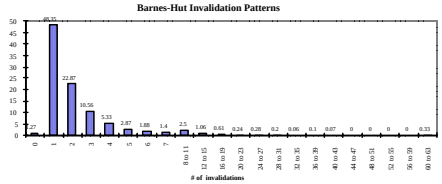
Reducing the Memory Overhead: SCI

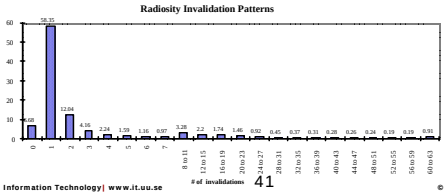
--- Scalable Coherence Interface (SCI)

- home only holds pointer to rest of the directory info [log(N) bits]
- distributed linked list of copies, weaves through caches
  - cache tag has pointer, points to next cache with a copy
- on read, add yourself to head of the list (comm. needed)
- on write, propagate chain of invalidations down the list
- on replacement: remove yourself from the list



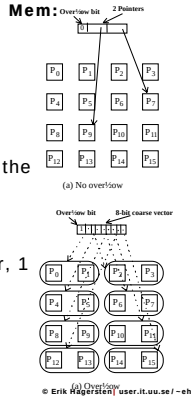
Cache Invalidation Patterns



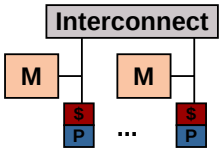


Overflow Schemes for Limited Pointers

- Broadcast (Dir$_i$B)
  - broadcast bit turned on upon overflow
  - bad for widely-shared invalidated data
- No-broadcast (Dir$_i$NB)
  - on overflow, new sharer replaces one of the old ones (invalidated)
  - bad for widely read data
- Coarse vector (Dir$_i$CV)
  - change representation to a coarse vector, 1 bit per k nodes
  - on a write, invalidate all nodes that a bit corresponds to



cc-NUMA issues



- Memory placement is key!
- Gotta' migrate data to where it's being used
- Gotta' have cache affinity
  - Long time between process switches in the OS
  - Reschedule processor on the CPU it ran last
- Origin 2000's migration always turned off☹

Sun's WildFire System

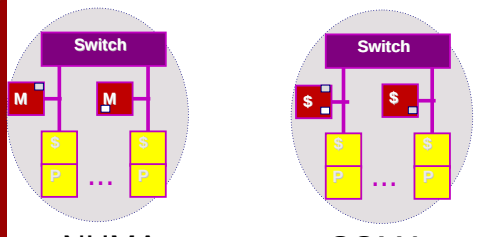
Erik Hagersten
Uppsala University
Sweden

Sun's WildFire System

- Runs unmodified SMP apps in a more scalable way than E6000
- Minot modifications to E6000 snooping required
- CPUs generate local address OR global address
- Global address --> no replication (NUMA)
- Coherent Memory Replication(~Simple COMA@ SICS)
- Hardware support for detecting migration/replication pages
- Directory cache + address translation cache backed by memory
- Deterministic directory implementation (easy to verify)

WildFire: One Solaris spanning four nodes

COMA: self-optimizing DSM

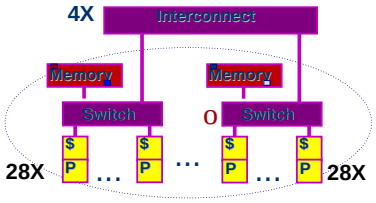


ccNUMA COMA

COMA:

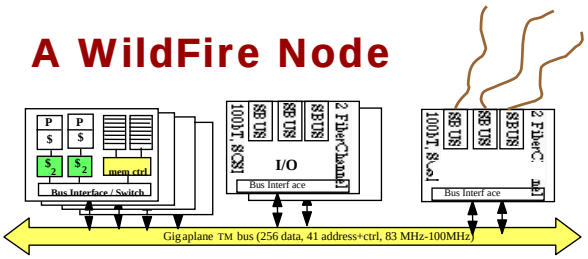
- **Self-optimizing architecture**
- **Problem at high memory pressure**
- **Complex hardware and coherence protocol**

Adaptive S-COMA of Large SMPs



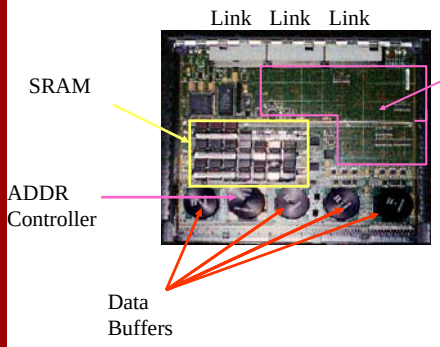
- **A page may have space allocated in many nodes**
- **HW maintains memory coherence per cache line**
- **Replication under SW control --> simple HW (S-COMA)**
- **Adaptive replication algorithm in OS (R-NUMA)**
- **Coherent Memory Replication (CMR)**
- **Hierarchical affinity scheduler (HAS)**
- **Few large nodes ->simple interconnect and coherence protocol**

A WildFire Node

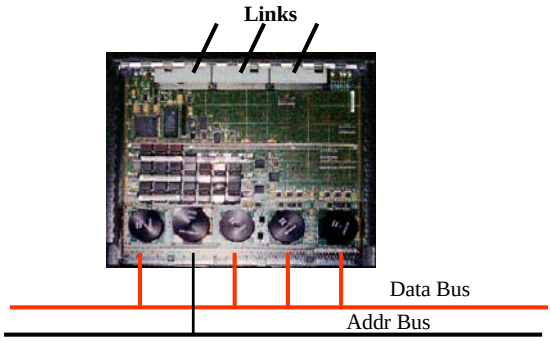


- 16 slots with either CPUs, IO or… WildFire extension board ➔
  - Up to **28** UltraSPARC processors
  - Gigaplane™ bus has peak bw 2.67 GB/s
  - Local access time of 330ns (lmbench)

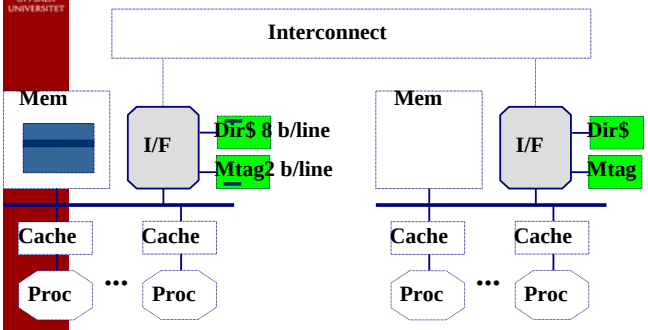
Sun WildFire Interface Board



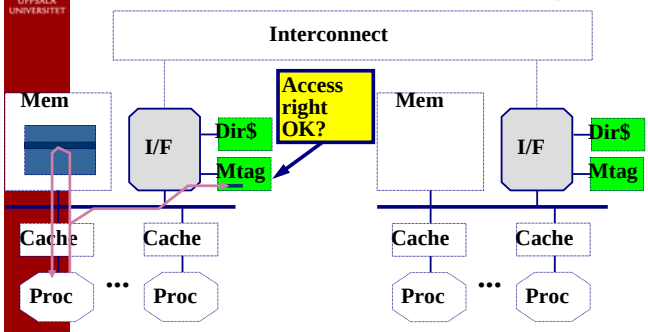
Sun WildFire Interface Board



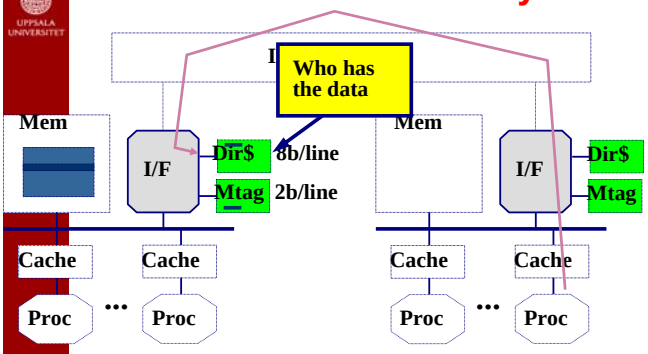
WildFire as a vanilla "NUMA'



NUMA -- local memory acces

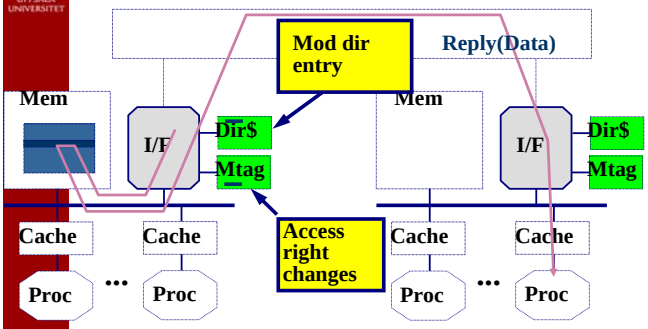


NUMA -- remote memory access

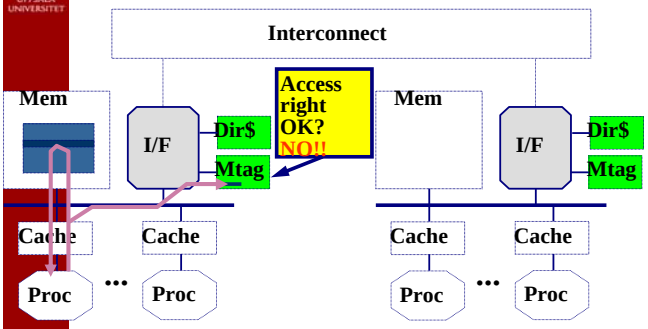


SRAM overhead = 10/512 = 2% (lower bound 2/512 = 0.4%)

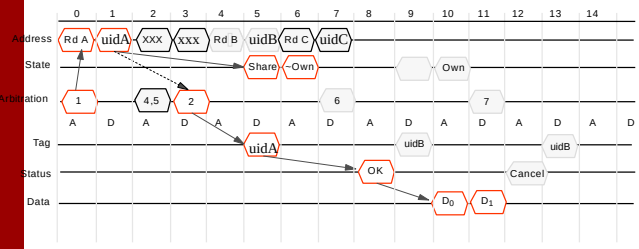
Global Cache Coherence Prot.



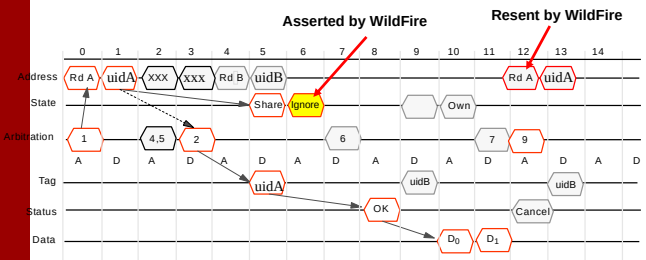
NUMA -- local memory access



Gigaplane Bus Timing

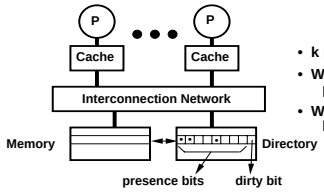


WildFire Bus Extensions



Ignore transaction squashes an ongoing transaction => not put in IQ
WildFire eventually reissues the same transaction
RTSF -- a new transaction sends data to CPU and memory

WildFire Directory -- only 4 nodes!!

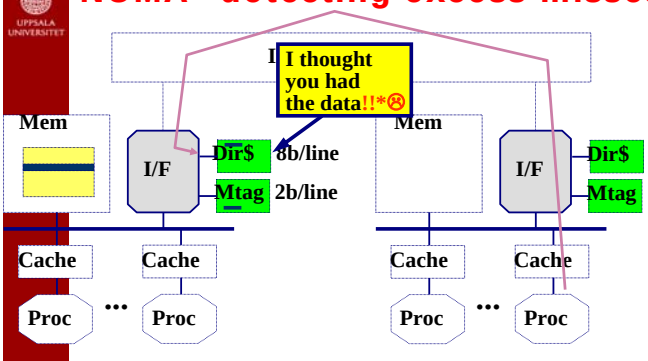


- k nodes (with one or more procs).
- With each cache-block in memory: k presence-bits, 1 dirty-bit
- With each cache-block in cache: 1 valid bit, and 1 dirty (owner) bit

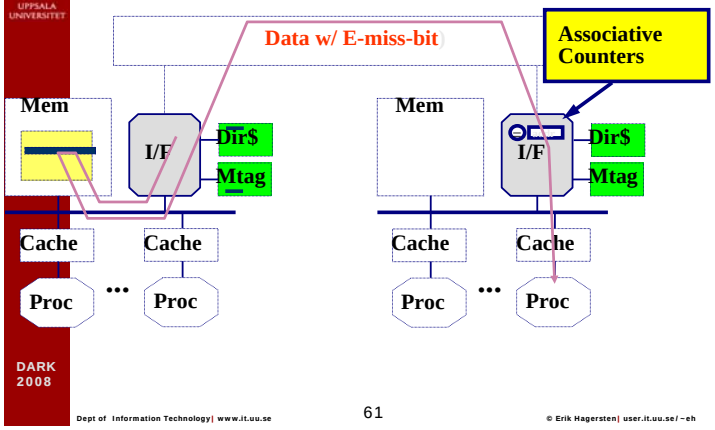
- **ReadRequestfrom main memory by processor i:**
  - **If dirty-bit OFF then { read from main memory; turn p[i] ON}**
  - **if dirty-bit ON then { recall line from dirty proc (cache state to shared); update memory; turn dirty-bit OFF; turn p[i] ON; supply recalled data to i;}**

....

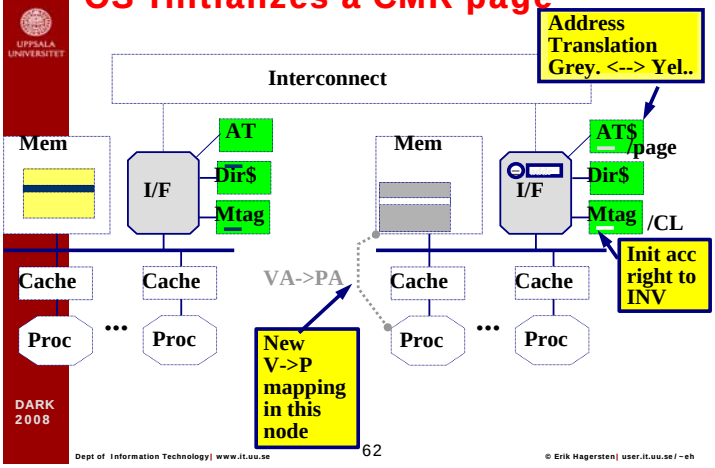
NUMA "detecting excess misses"



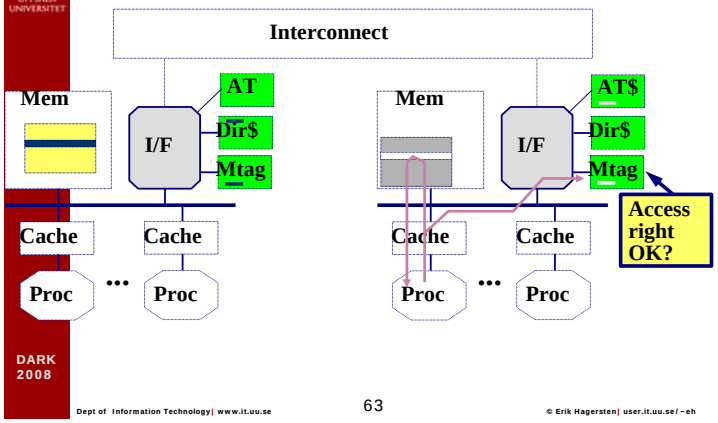
Detecting a page for replication



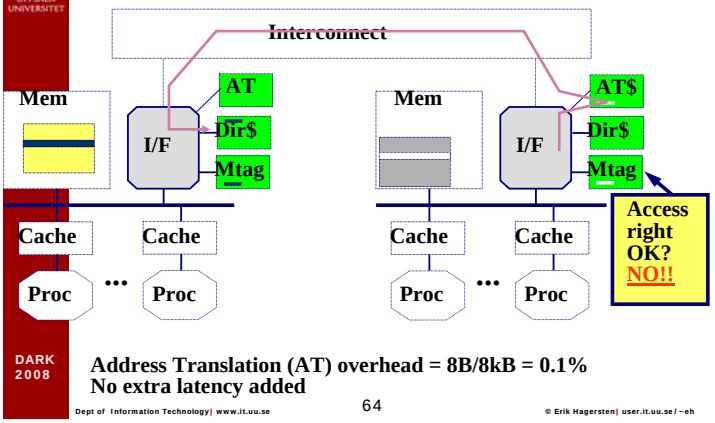
OS Initializes a CMR page



An access to a CMR page

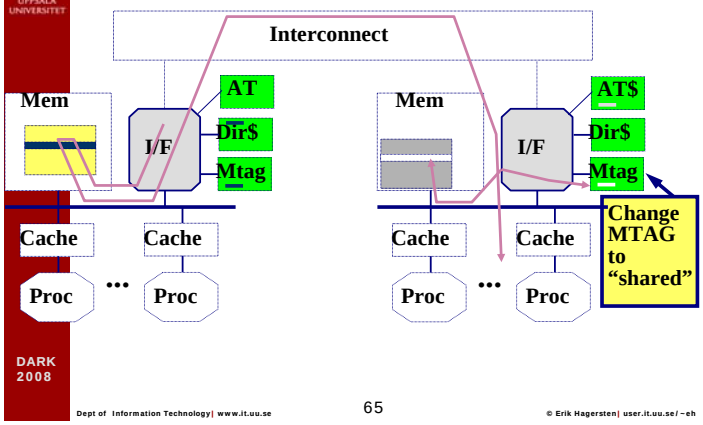


An access to a CMR page (miss)

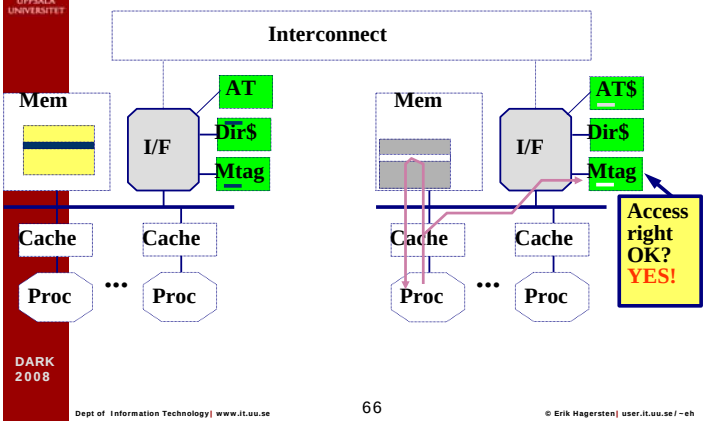


Address Translation (AT) overhead = 8B/8kB = 0.1%
No extra latency added

An access to a CMR page (miss)

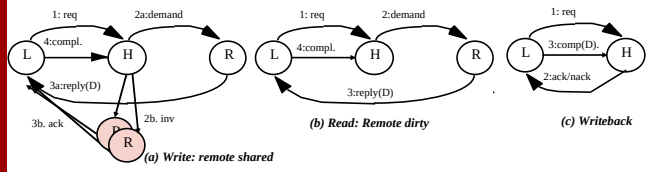


An access to a CMR page (hit)

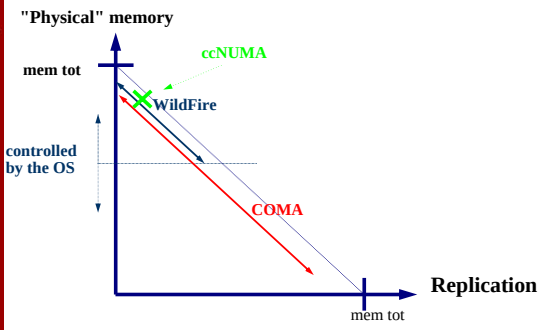


Deterministic Directory

- MOSI protocol, fully mapped directory (one bit/node)
- Directory blocking: one outstanding trans/cache line
- Directory blocks new requests until completion received
- The directory state and cache state always in agreement (except for silent replacement...)



Replication Issues Revisited



- Only "promising" pages are replicated
- OS dynamically limits the amount of replication
- Solaris CMR changes in the hat_layer (=port)

Advantages of Multiprocessor Nodes

Pros:

- amortization of fixed node costs over multiple processors
- can use commodity SMPs
- fewer nodes to keep track of in the directory
- much communication may stay within node (NUCA)
- can share "node caches" (WildFire: Coherent Memory Replication)

Cons:

- bandwidth shared among processors and interface
- bus may increases latency to local memory
- snoopy bus at remote node increases delays there too

Memory cost of replication

Example: Replicate 10% of data in all nodes

- 50 nodes, each with 2 CPUs
  ==> 490% overhead
- 4 nodes, each with 25 CPUs
  ==> **30% overhead**

Does migration/replication help?

NAS parallel Benchmark Study (Execution time in seconds)

[M. Bull, EPCC 2002]

Shallow

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	26	5.9	6.1	6.1
Repl	7.2	6.2	6.1	6.1

Unopt. HWopt. SWopt.

BT

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	960	610	620	600
Repl	590	580	580	580

FT

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	520	330	380	260
Repl	250	260	190	200

SP

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	1540	780	760	780
Repl	670	680	670	670

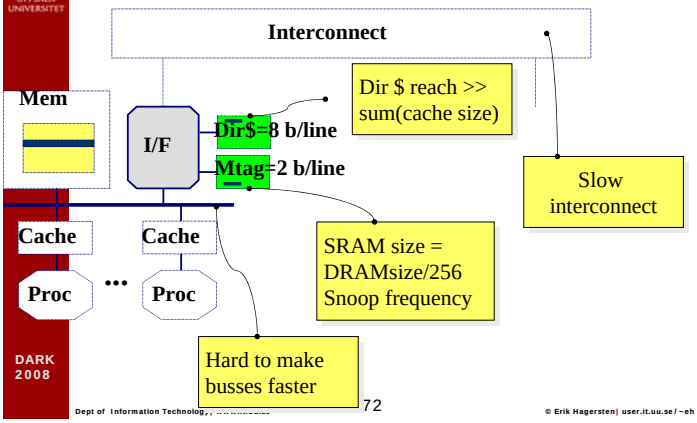
MG

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	230	230	240	230
Repl	220	220	220	220

CG

	No Initial Plac.		Initial Placement	
	No migr	Migr	NoMigr	Migr
No Repl	1060	700	940	700
Repl	300	280	300	290

WildFire's Technology Limits



Sun's SunFire 15k/25k

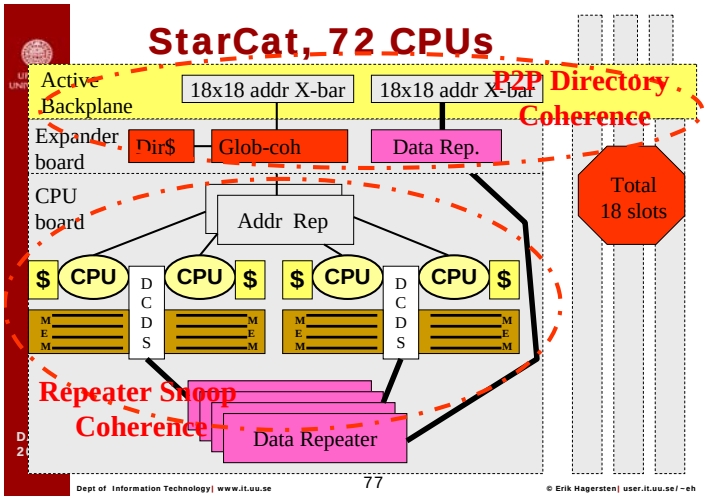
Erik Hagersten
Uppsala University
Sweden

StarCat
Sun Fire 15k/25k
(used at Lab2)

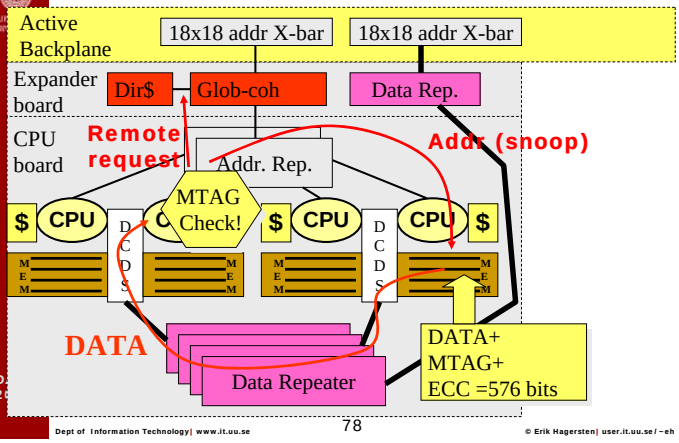
Front Side

Back Side

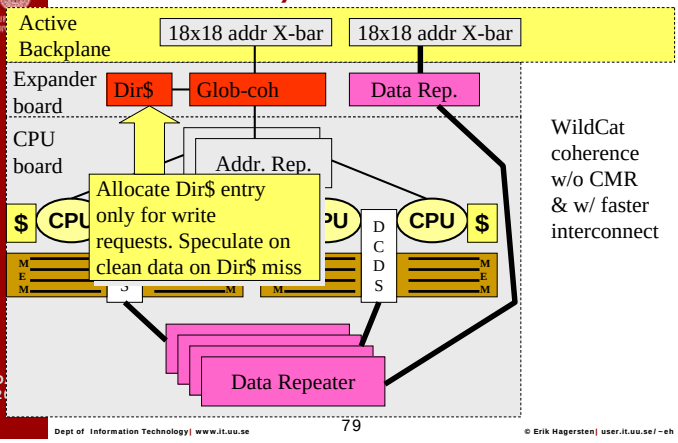
StarCat, 72 CPUs



StarCat Coherence Mechanism

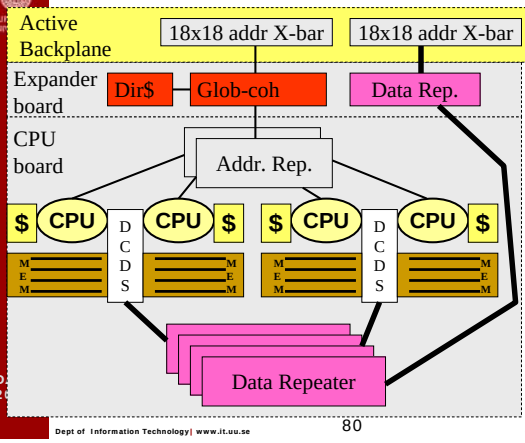


StarCat, 72 CPUs



WildCat coherence w/o CMR & w/ faster interconnect

StarCat Performance Data



Lat = 200-340ns
GBW=43GB/s
LBW=86GB/s

Up to 104 CPU (trading for I/O)