NUMA　machines and directory cache mechanisms

AMANO,Hideharu

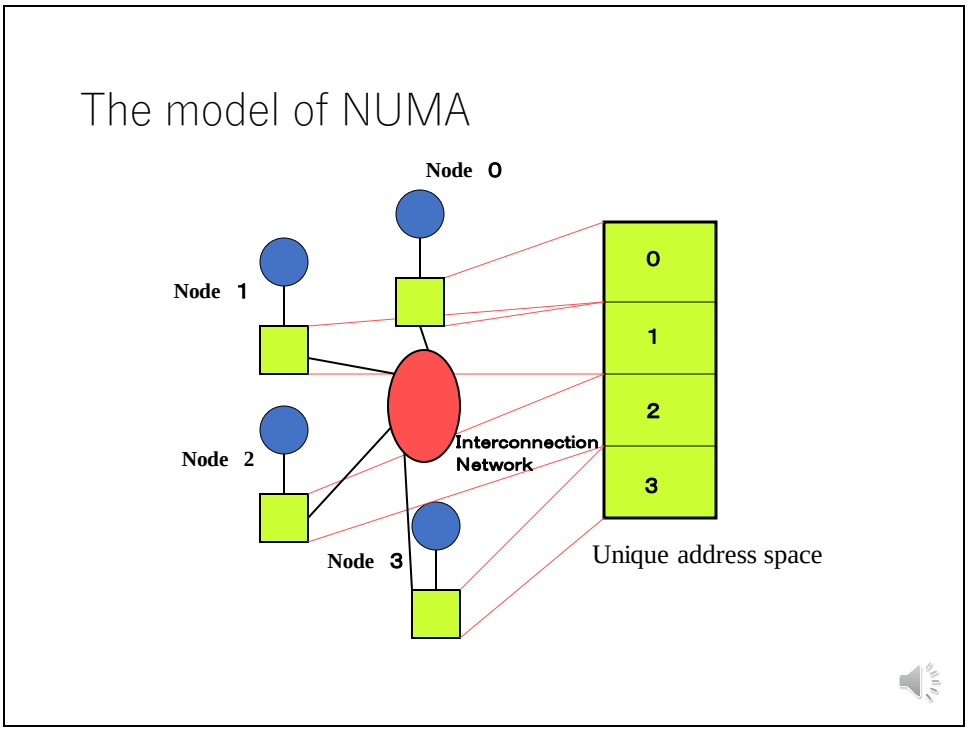
Textbook pp. 7 0 ～ 7 9

NUMA(Non-Uniform Memory Access model)

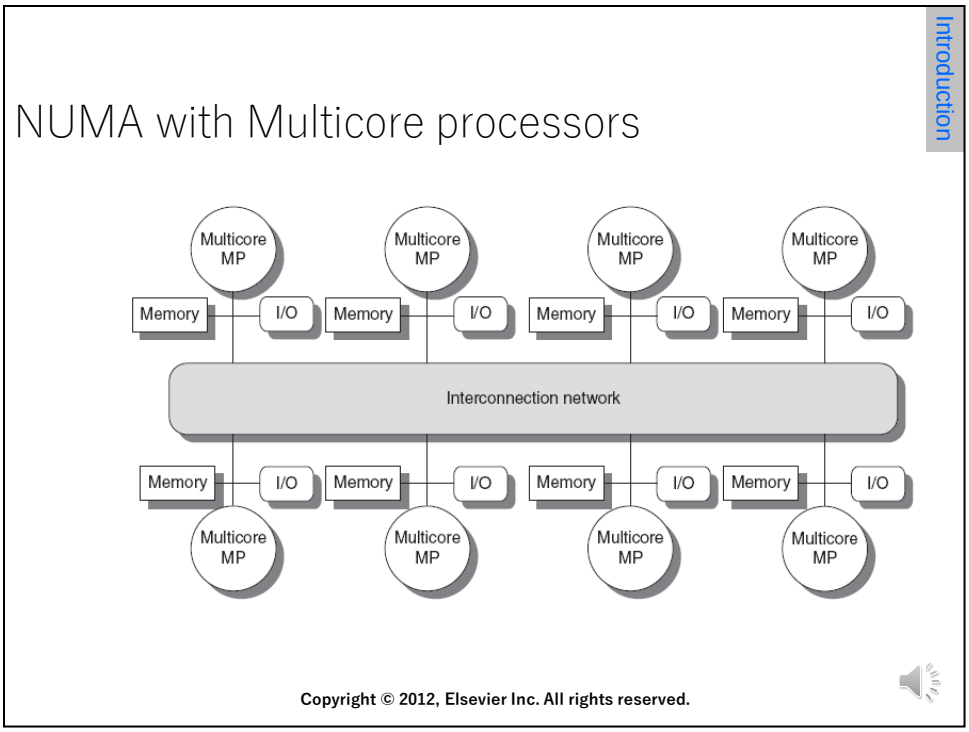
- Providing shared memory whose access latency and bandwidth are different by the address.
- Usually, its own memory module is easy to be accessed, but ones with other PUs are not.
- All shared memory modules are mapped into a unique logical address space, thus the program for UMA machines works without modification.
- Also called a machine with Distributed Shared Memory

  ⇔ A machine with Centralized Shared memory (UMA).

NUMA or non-uniform memory access model has shared memory but their access latency and bandwidth are different by the address. Usually, its own memory module is easy to be accessed but ones with others PUs are not. All shared memory modules are mapped into a unique logical address space, thus the program for UMA machines works without modification. They are called with distributed shared memory. It is an opposite concept of a machine with centralized shared memory or UMA.



This diagram shows the model of NUMA. Each memory module is assigned into the unique address space. If the node 0 accesses the address area 0, it accesses its own memory module. But, if it wants to access the other address, the request must be transferred through the interconnection network. Thus, the latency is stretched, and the bandwidth is limited.



In the recent servers, each node is a multicore, and its architecture is UMA which is introduced in the previous lesson.

Variation of NUMA

- Simple NUMA： cache coherence is not kept by the hardware（CM*,Cenju, T3D, RWC-1, Earth simulator）
- CC (Cache Coherent)-NUMA： providing coherent cache.（DASH, Alewife, Origin, SynfinityNUMA, NUMA-Q, Recent servers）
- COMA (Cache Only Memory Architecture) : No home memory（DDM,KSR-1）

NUMAs are classified into three categories. One is a simple NUMA. It can cache the memory attached to the other PUs, but the coherence is not kept. On the contrary, Cache Coherent NUMA provides the coherent cache. It must provide the hardware mechanism to keep the coherence, so it tends to be complicated. The last style is COMA. But the machines in this class is not used recently.

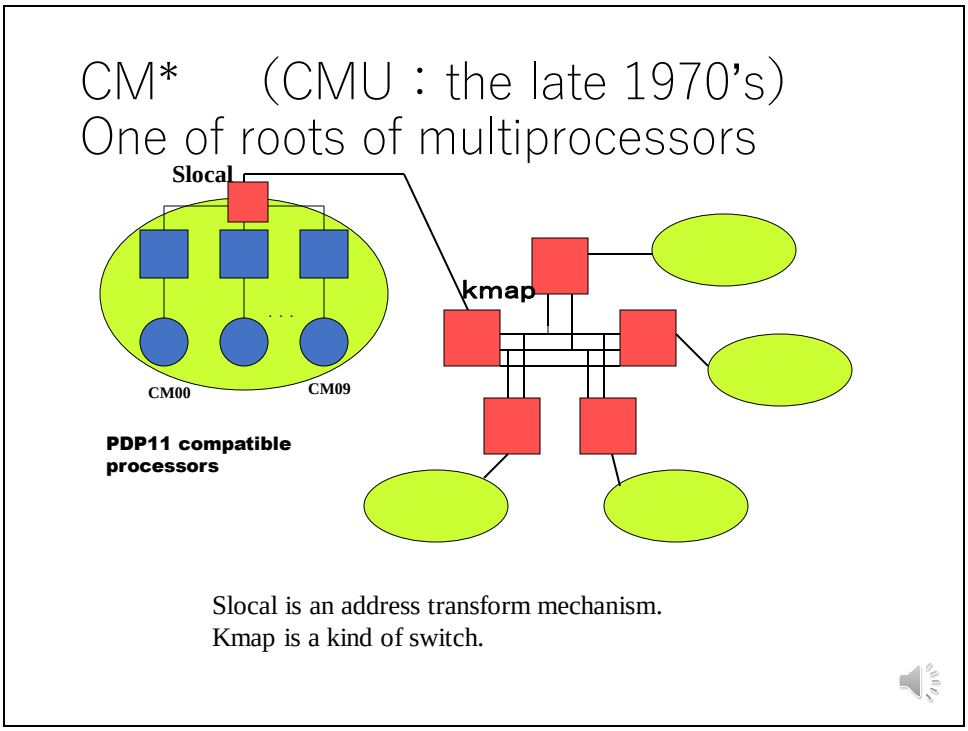
Glossary 1

- NUMA（Non-Uniform Memory Access model):
  メモリへのアクセスが均一ではないモデル（アーキテクチャ）、今回のメインテーマで別名Distributed Shared Memory machine：分散共有メモリマシンとも呼ばれる。この言葉の逆の意味はCentralized Memory: 集中共有メモリということになりUMAである
- Cache-Coherent NUMA:キャッシュの一貫性がハードウェアで保証されているNUMA　後で説明するようにプロトコルが面倒
- COMA(Cache Only Memory Architecture):キャッシュだけのメモリアーキテクチャという意味だがもちろんキャッシュだけで構成されているわけではなく、ホームメモリを決めないものをこのように呼ぶ

Simple NUMA

- A PU can access memory with other PUs/Clusters, but the cache coherence is not kept.
- Simple hardware
- Software cache support functions are sometimes provided.
- Suitable for connecting a lot of PUs:
  Supercomputers : Cenju, T3D, Earth simulator, IBM BlueGene, Roadrunner, K, Fugaku
- Why some supercomputers take the simple NUMA structure?
  - Easy programming for wide variety of applications
  →Powerful interconnection network

First of all, the simple NUMA is introduced. Some supercomputers use this style. It has some benefits. I will introduce some of them.

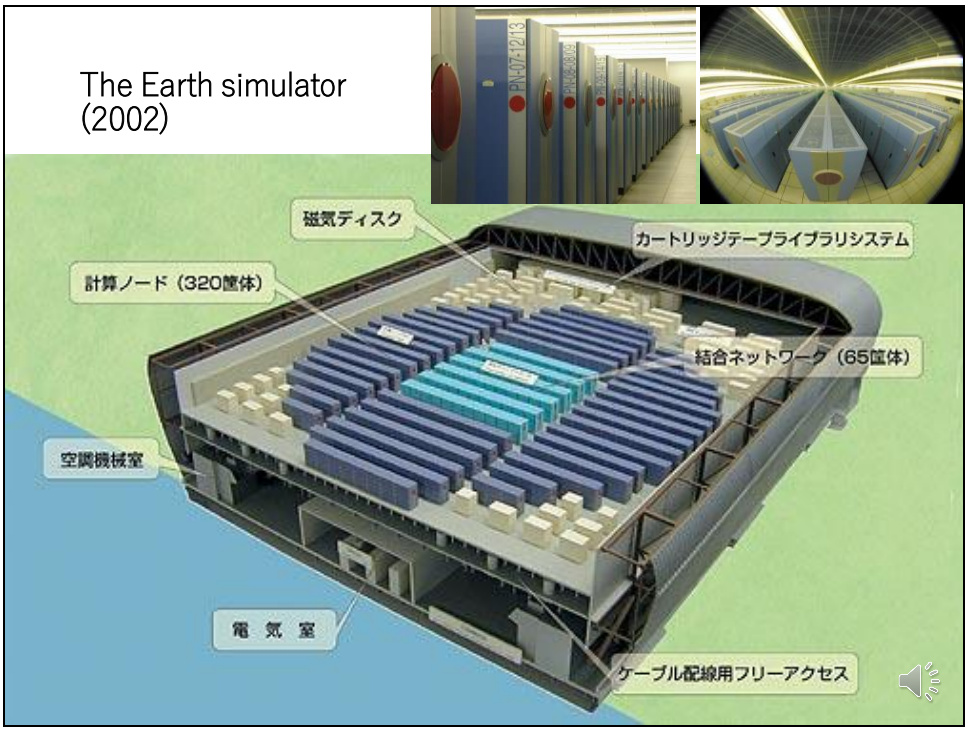


CM*, developed by CMU in the late 1970's, is a root of multiprocessors. They used PDP11 as a cluster, and provided an address transform mechanism called Slocal. The link from Slocal is connected with Kmap, a kind of switch. The memory in the other cluster can be accessed through the Kmap and Slocal.

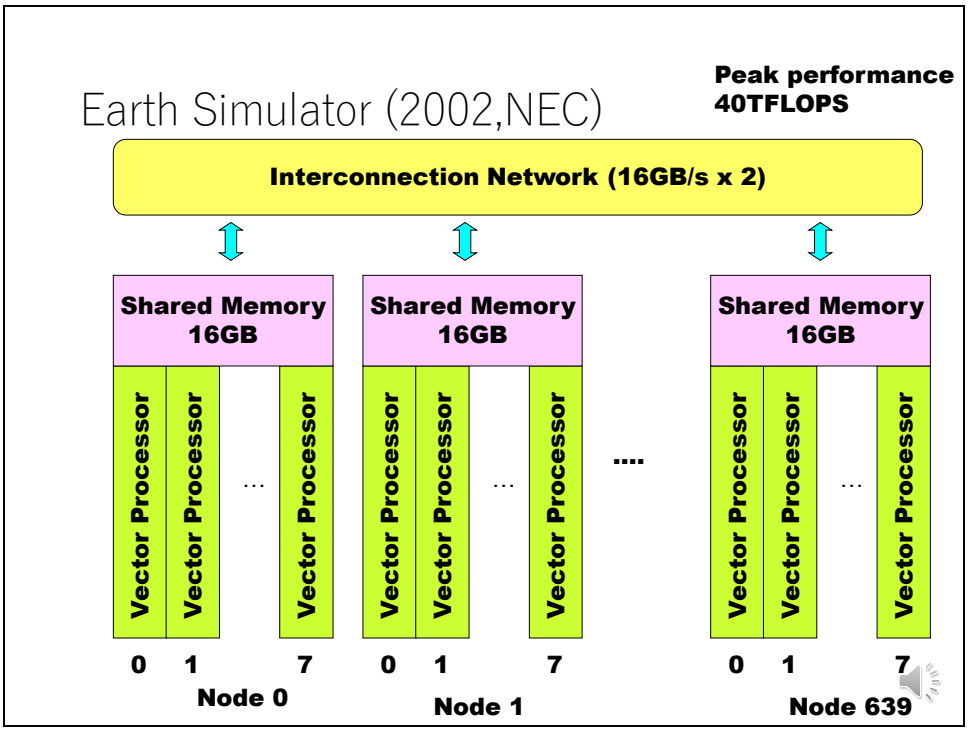
Cray's T3D: A simple NUMA supercomputer (1993)

- Using Alpha 21064

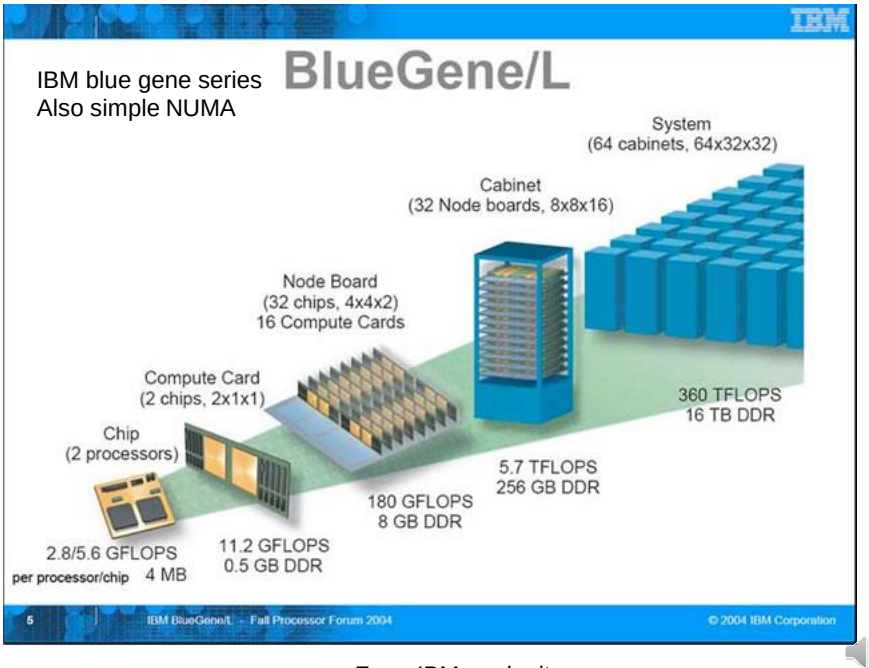
Supercomputers have used this style. Cray's Tera three D was a simple NUMA supercomputer.



The Earth Simulator got the top 1 in the world 2002. A lot of cabinets are placed on the big building like a gym. The deep blue ones are for computational nodes and light blue ones are for interconnection networks.

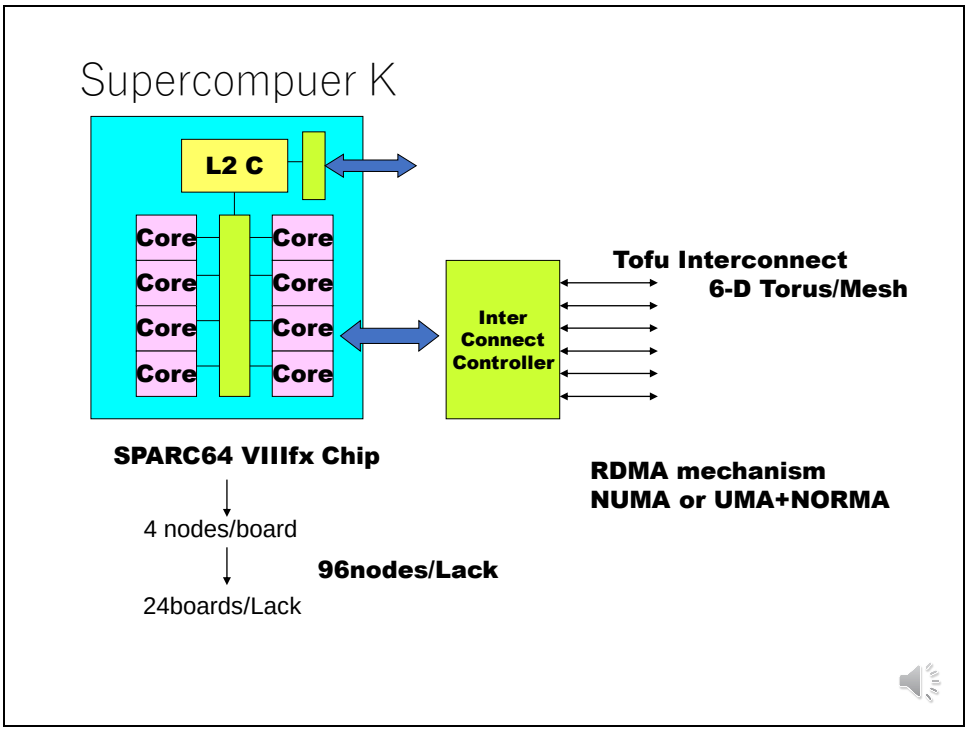


It forms a node with 8 vector processors, and connects 639 nodes with a large crossbar switch. Since the performance of the interconnection network was huge, it achieved an efficient performance close to the peak performance.

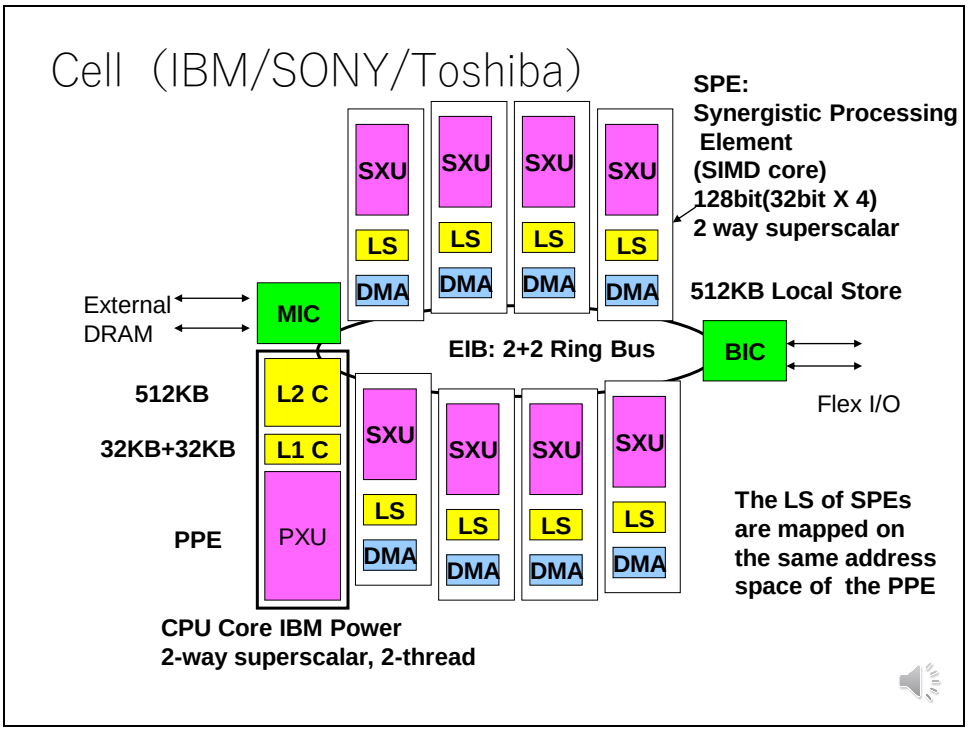


From IBM web site

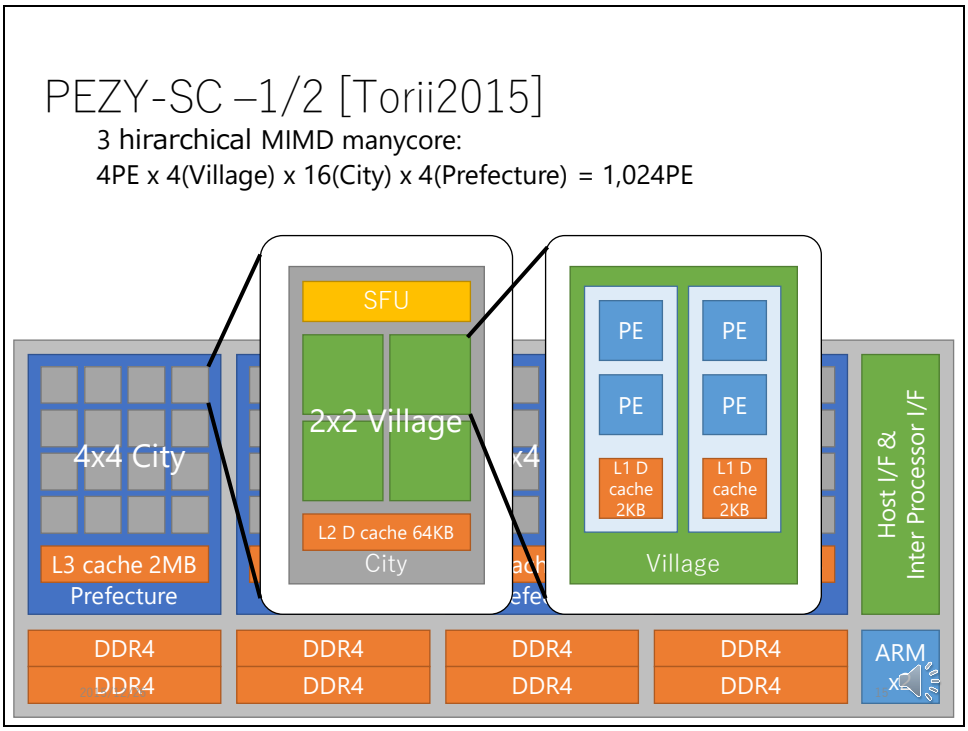
IBM blue gene series BlueGene/L, P and Q also used simple NUMA structure. They are connected with 3-D torus network instead of the crossbar of the earth simulator.



Japanese super computer K uses a simple NUMA structure. It provides Remote DMA mechanism to send the data from other nodes.



The IBM/SONY/Toshiba developed cell broadband engine for their game machine play station 3. It was also used as several supercomputers. In this architecture, all local memory modules attached to eight SPEs are mapped into the same address space of the host processor address space.



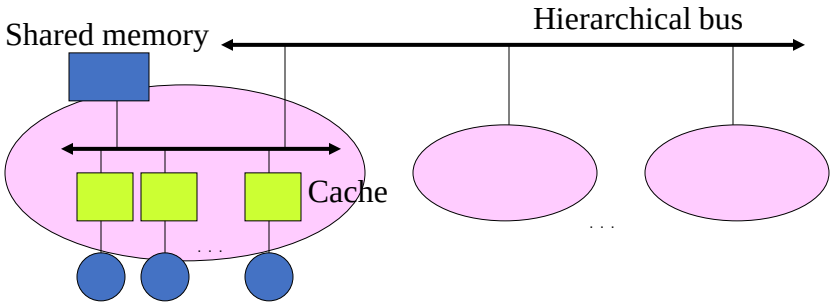
Pezy SC 1 and 2 adopted a hierarchical structure. 4x4 cities which share the L3 cache form a prefecture. A city consists of 2x2 villages which share L2 cache, and a village is built by 4 PEs. 2PEs share the L1 cache. It can cache the main memory but coherence is only kept in each hierarchy. Although it has an interesting memory architecture, the company head was arrested for the illegal acquition of national research fund and the project was terminated.

CC-NUMA

- Directory management mechanism is required for coherent cache.
- Early CC-NUMAs use hierarchical buses.
- Complicated hardwired logic
  - Stanford　DASH、MIT　Alewife、Origin、Sinfinity NUMA
- Dedicated management processor
  - Stanford　FLASH（MAGIC)、NUMA-Q(SCLIC)、JUMP-1(MBP-light)

Unlike simple NUMAs, CC-NUMAs provide a directory management mechanism for keeping the coherent cache. Early CC-NUMAs were an extension of the snoop cache and had hierarchical buses. But, later it was replaced to the directory management system with a point-to-point network. Some used complicated hardware logic, others used dedicated management processors.

Ultramax (Sequent Co.)
An early CC-NUMA

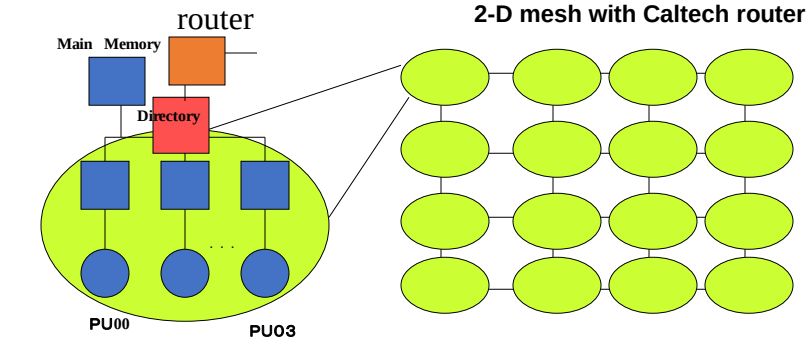


Hierarchical extension of bus connected multiprocessors

Hierarchical bus bottlenecks the system.

An early CC-NUMA used an extension of the snoop cache by introducing a hierarchical bus. Each cluster was a snoop cache connected multiprocessor, and the accesses for the other cluster uses hierarchical bus. With the similar protocol of the snoop cache also on the hierarchical bus, the cache coherence was kept. Apparently, this approach causes the traffic congestion of the hierarchical bus.

Stanford DASH
A root of recent CC-NUMAs

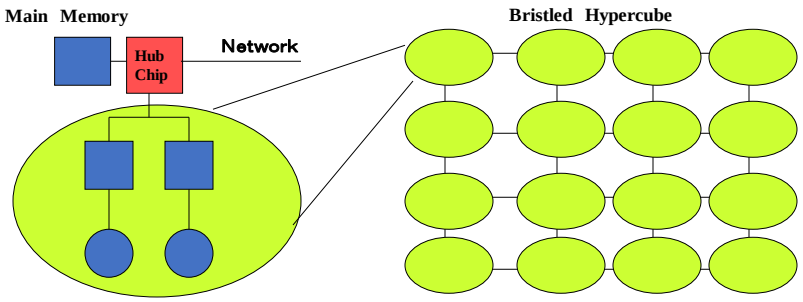


SGI Power Challenge

Directory Coherent control、Point-to-Point connection
Release Consistency

Stanford DASH introduced the directory coherent control mechanism, point-to-point interconnection and release consistency model which are used in the current servers. The cluster was SGI's Power Challenge workstation and they attached the directory mechanism and the router. The router was developed in the Caltech university, and a simple 2-dimensional mesh network was used.

SGI Origin



Main Memory is connected with Hub Chip directly.
1 Cluster consists of 2 PEs.

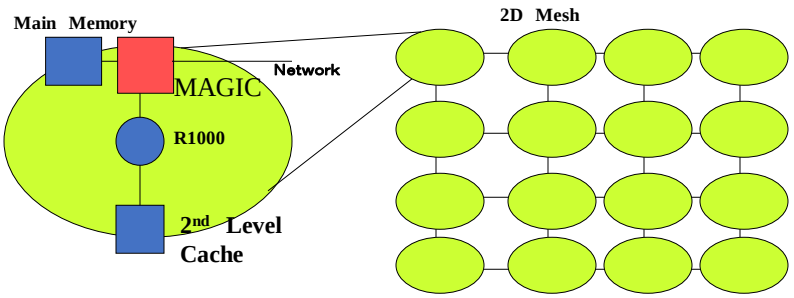
SGI Origin is a commercial version of the DASH. The number of Processors in a cluster was reduced because of the rapid performance improvement of a processor. Hub chip which manages the directory were used, and they formed a bristled hypercube.

SGI's CC-NUMA Origin3000(2000)

- Using R12000

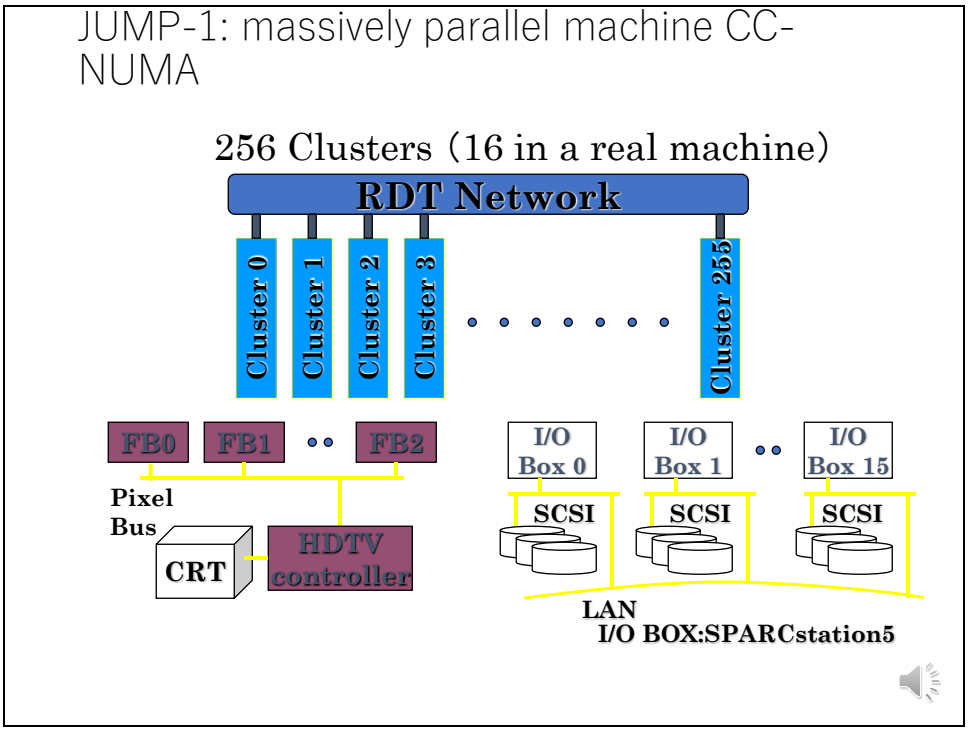
This machine was working in the ITC of this campus. We developed parallel programs on this machine.

Stanford FLASH

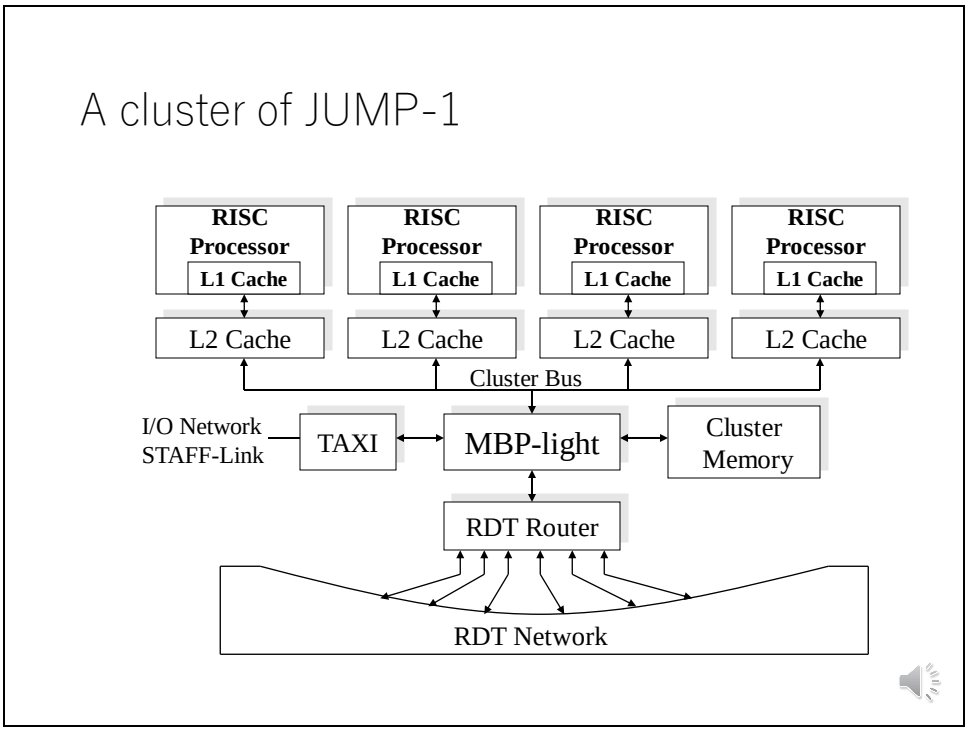


MAGIC is a dedicated processor for protocol control.

Since the hardware which controls the coherence became so complicated, the Stanford university developed a dedicated chip which controls the cache coherence with its software.



Jump-1, a CC-NUMA was developed by the Japanese national project by cooperation of seven Japanese Universities.



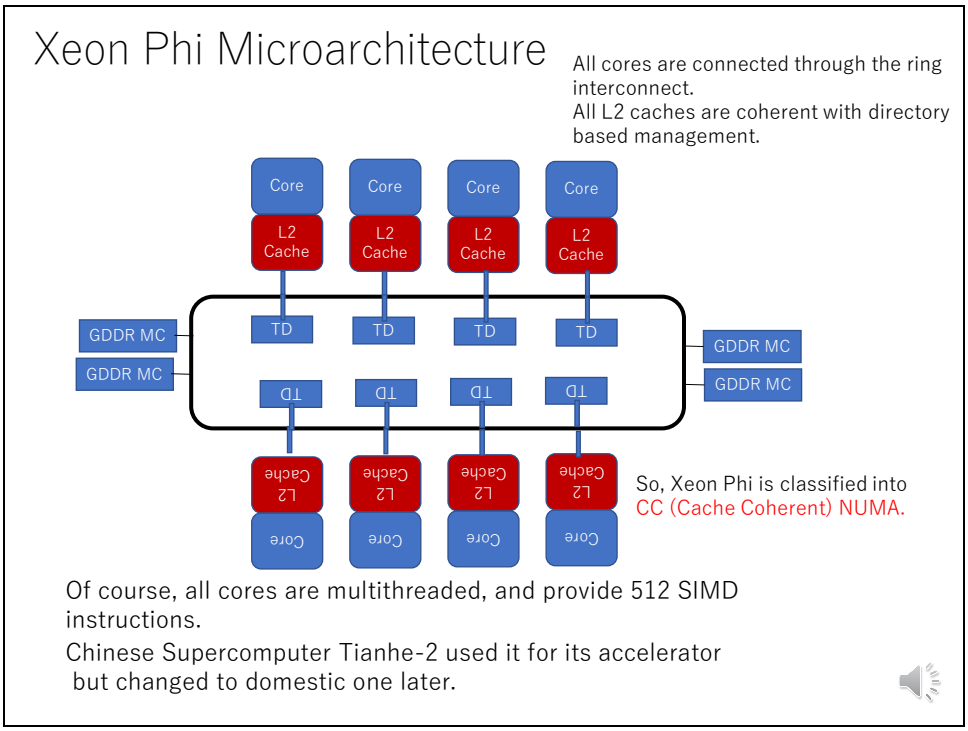
Like a Stanford project, it sed a dedicated processor called MBP light for the cache coherent control. 4 SPARC processors are used to develop a cluster, and a special interconnection network called RDT or recursive diagonal torus was used as an interconnection network.

JUMP-1 was developed with 7 universities

A system with 4 clusters
(Keio Univ.)

A system with 16 clusters
(Kyoto Univ.)

They are outlook of Jump-1. A first prototype with 4cluster and 16 processors were developed in Keio University. Later 16 clusters with 64 processors worked in Kyoto University.

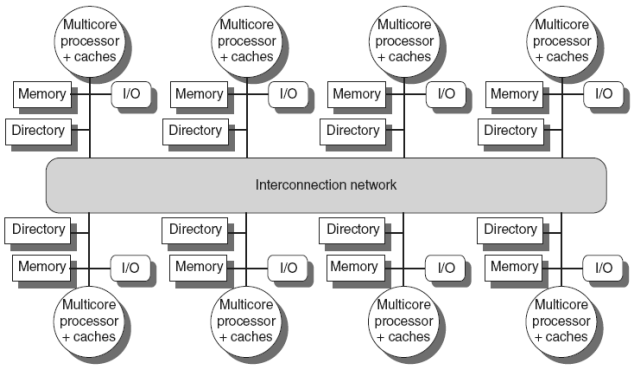


Xeon Phi microarchitecture is a CC-NUMA with directory control mechanism. It provides 8 cores each of which provide directory. L2 cache is kept coherent with this mechanism.

Multicore Based systems

- Implementing in shared L3 cache
  - Keep bit vector of size = # cores for each block in L3
  - Not scalable beyond shared L3

IBM Power 7
AMD Opteron 8430



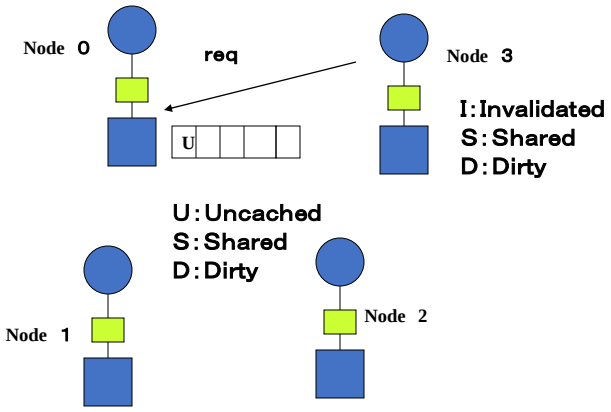
Some recent server used directory controlled CC-NUMA structure. Here directory is attached to each memory system.

Distributed cache management of CC-NUMA

- Cache directory is provided for the cache block of the home memory.
- The cache coherence is kept by messages between nodes.
- Invalidation type protocols are commonly used.
- The protocol itself is similar to those used in snoop cache, but everything must be managed with message transfers.

The directly is provided for each cache block of the home memory, and the cache coherence is kept by messages between nodes.

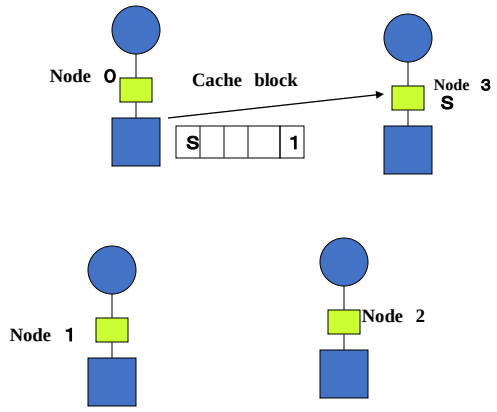
Cache coherent control (Node 3 reads)



Let me explain the cache coherent control using the directory. Each directory entry for the home memory has the state of the block and the bit map which shows who has the copy of the block. There are three states: U, S and D. At first, the state is U. Each cache directory has also its state. We assume three states: I, S, and D.

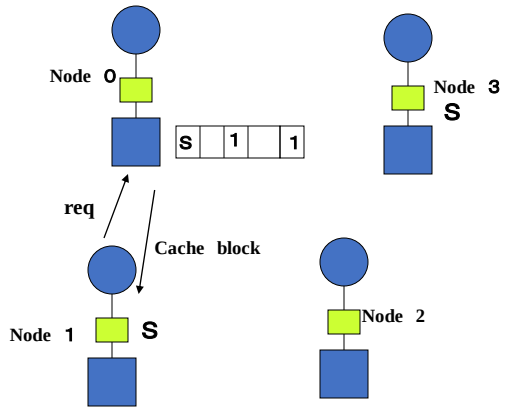
Here, let's assume Node 3 sends the request to the Node 0 home memory.

Cache coherent control (Node 3 reads)



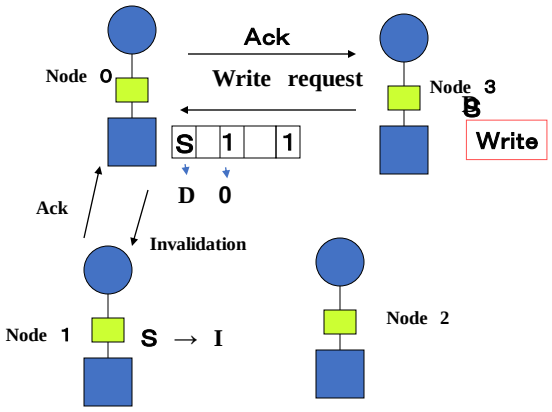
Node 0 replies it and sends the cache block to the requesting node 3. It also changes its state into S, and set 1 at the corresponding bit map. The state of the cache of the requesting node 3 turns its state into S.

Cache coherent control (Node 1 reads)

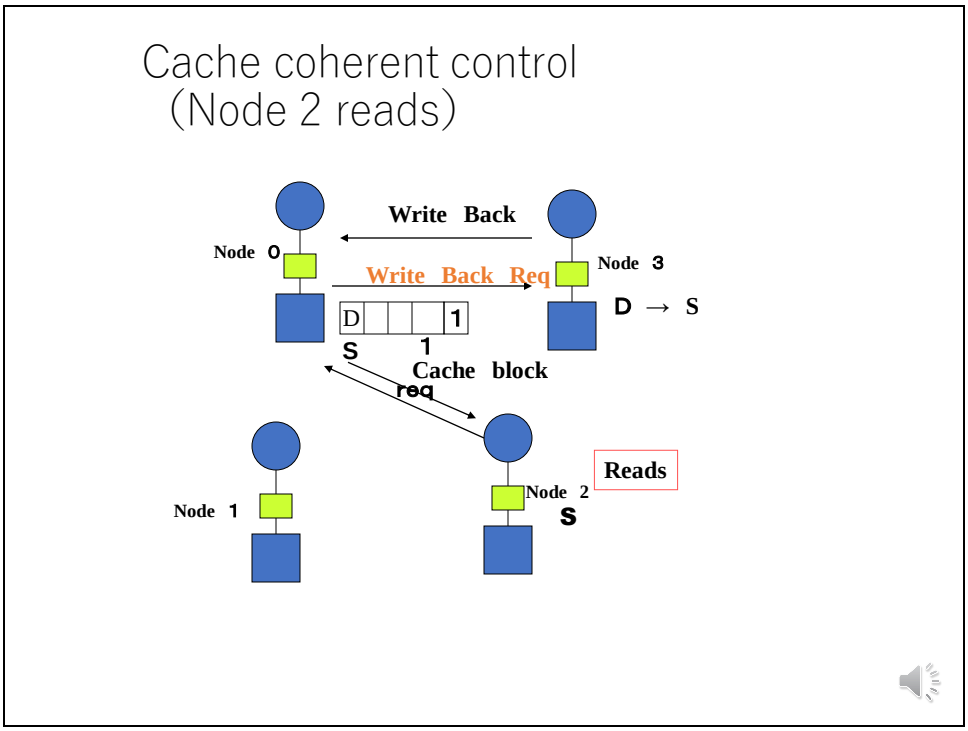


The similar thing happens when node 1 issues the request. The cache block is sent back from the node 0, and the corresponding bit is set.

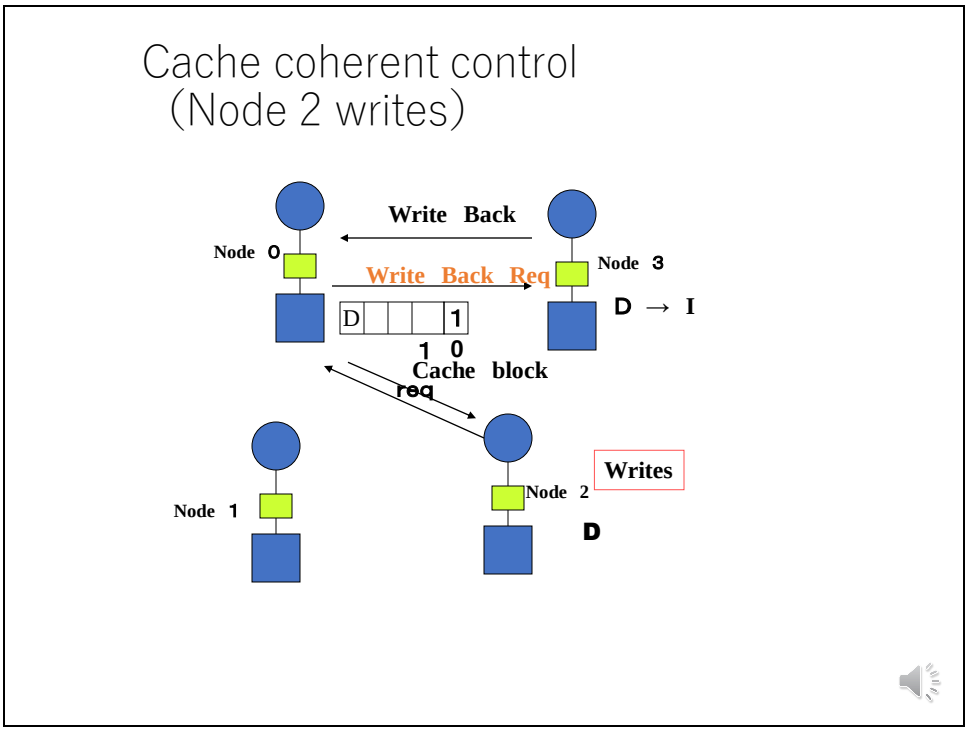
Cache coherent control (Node 3 writes)



When node 3 wants to write the data into the cache block, it sends the write request message to the home node 0. It checks the directory and knows that node 1 has the same block. So, node 0 sends the invalidation message to node 1. Node 1 invalids its cache block and sends back the acknowledge signal to node 0. Node 0 changes home memory state to D, and reset the bit corresponding to node 1. Then it sends the acknowledge message to node 3. After receiving it, node 3 changes its cache state into D. After that node 3 can read and write the block without sending any messages.



What happens when node 2 wants to the same memory address. It sends the home memory and the home node knows that it has been updated by the node 3 by checking the directory. So, node 0 sends the write back request message to node 3. Node 3 replies to send the updated cache block and changes its state to S. After writing back the data, node 0, the home node changes its sate into S and set the bit corresponding to Node 2. Then node 0 send the cache block to node 2. The cache state of node 2 becomes S.

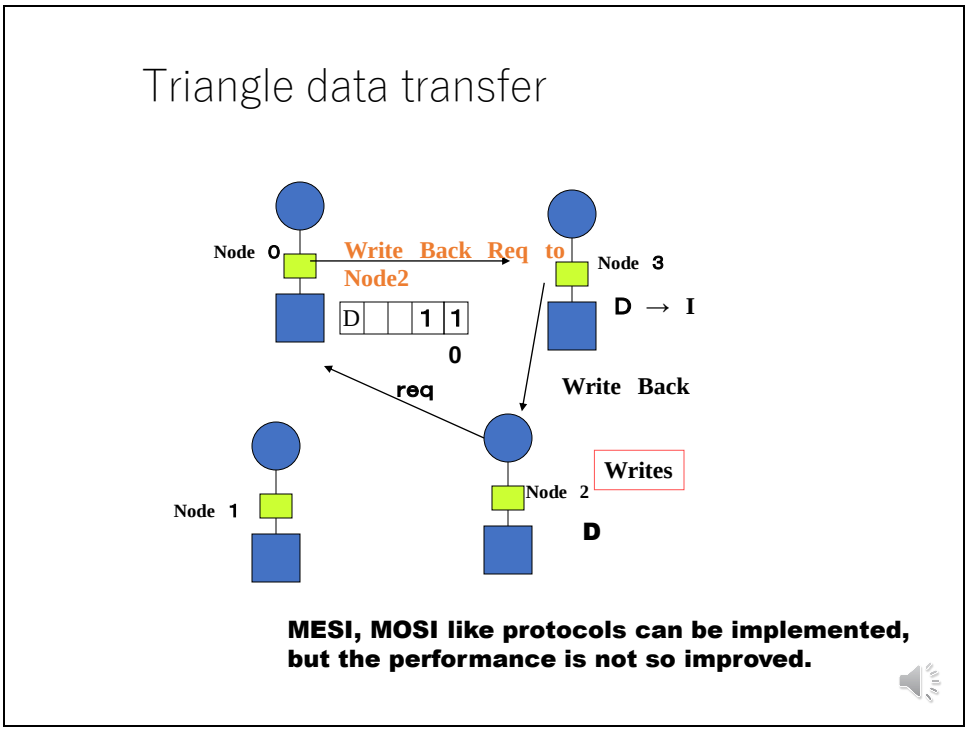


What happens node 2 sends the write request instead of the read request. Similar to the case of read request, node 3 writes back the cache block to the node 0 home memory. But the cache state becomes I. Node 0 changes its state into D and set the bit 2 instead of bit 3. After getting the cache block, node 2 changes its state into D.

Quiz

- Show the states of cache connected to each node and directory of home memory in CC-NUMA.
- The node memory in node 0 is accessed:
  - Node 1 reads
  - Node 2 reads
  - Node 1 writes
  - Node 2 writes

Here is a quiz.



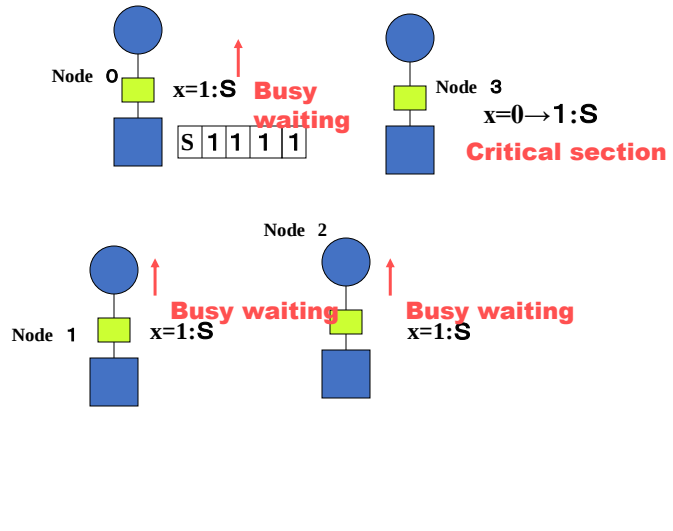
In order to improve the performance, node 3 sends the cache block directly to the requesting node 2 instead of the home node 0. The node 2 writes the data directly and changes state into D. It is somehow similar to that of the ownership of the snoop protocol. Techniques proposed for the snoop cache can be used, but the performance improvement is not so large.

Synchronization in CC-NUMA

- Simple atomic operations (eg. Test&set) increase traffic too much.
- Test and Test&set is effective, but not sufficient.
  - After sending an invalidation message, traffic is concentrated around the host node.
- Queue-based lock:
  - linked list for lock is formed using directory for cache management.
  - Only the node which can get a lock is informed.

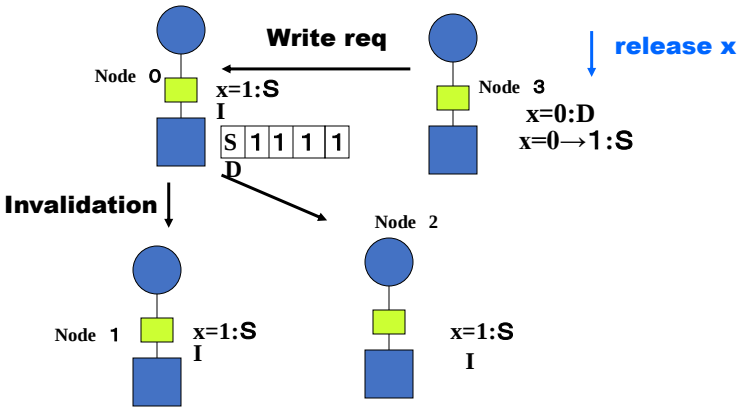
Simple atomic operations increase traffic too much. Test and Test&Set is effective, but not enough. So, Queue-based lock is proposed.

Traffic congestion caused by Test and Test&Set(x) (Node 3 executes the critical section)



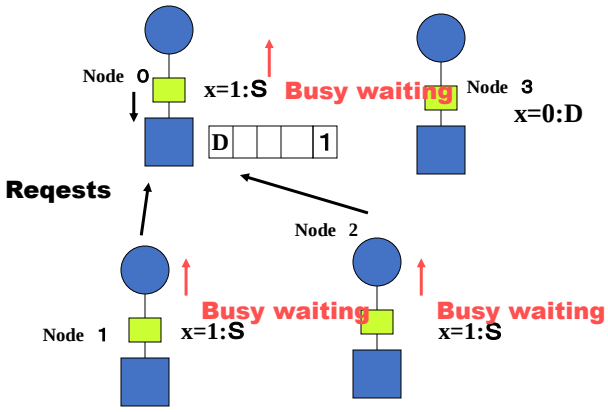
Assume that node 3 executes the critical section, and other nodes are waiting for the releasing the synchronization variable x. Thanks to test and test&Set, each node executes busy waiting without sending messages.

Traffic congestion caused by Test and Test&Set(x) (Node 3 finishes the critical section)

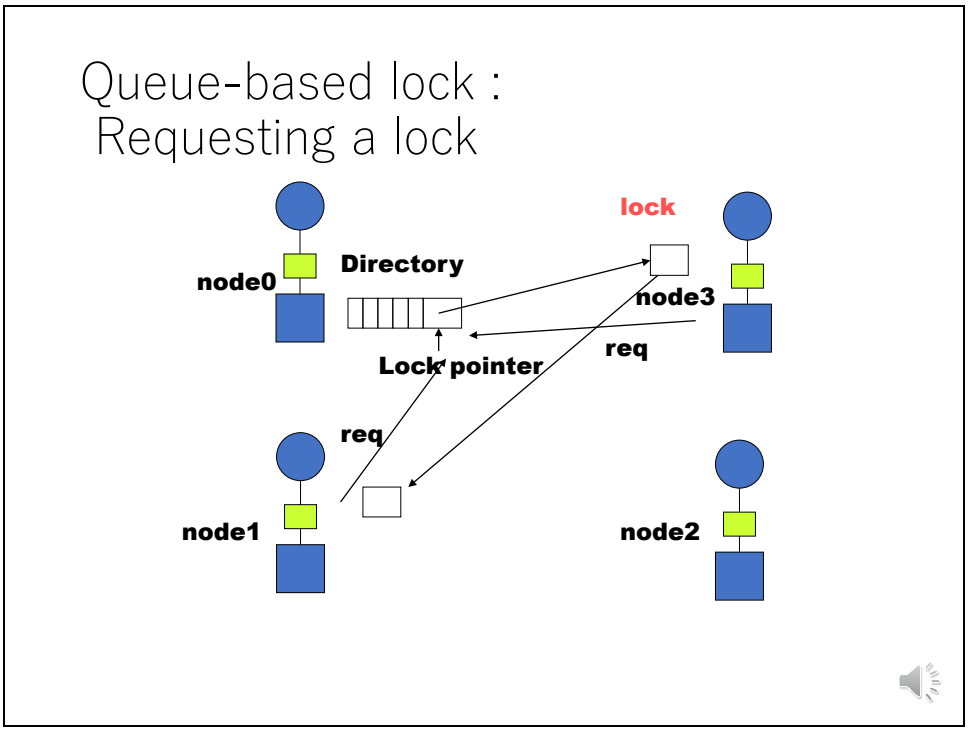


However, when the node 3 releases the critical section by writing x, it sends the write request to the home node. and the node 0 sends invalidation messages to all other nodes.

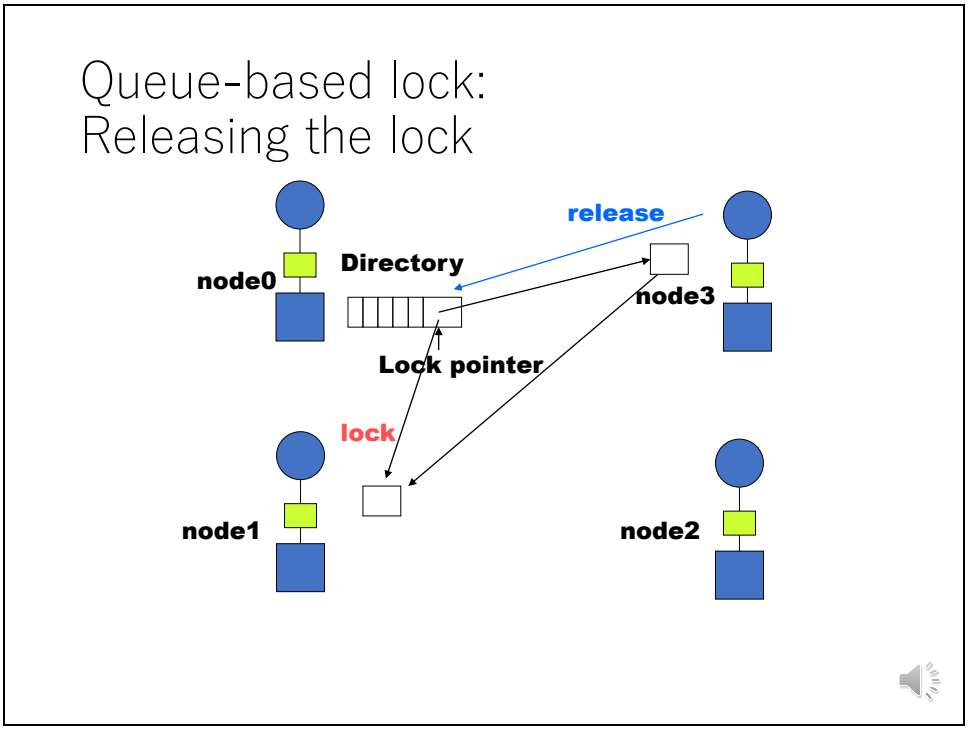
Traffic congestion caused by Test and Test&Set(x) (Waiting nodes issue the request)



After that, all nodes must reply the acknowledge messages, and requests again to get the synchronization variable x. All these operations require a lot of messages and causes the congestion around the home node.



Queue-based lock provides the pointer to each node. When synchronization request is issued from node3, a link to get the lock is made, and the pointer is stored. When other nodes make a request, they are linked to the list in order.



When node3 releases the synchronization variable, it changes the pointer so that it indicates the next node. So, lock, the right to access the critical section will move around the linked list.

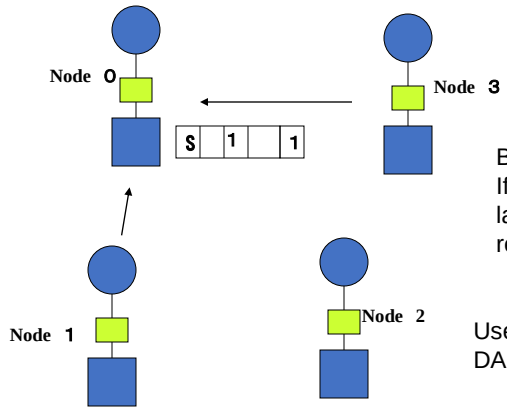
Directory structure

- Directory Methods
  - Full Map directory
  - Limited Pointer
  - Chained Directory
  - Hierarchical bit-map
- Recent CC-NUMAs with multicore nodes is small scale, and the simple full map directory is preferred.
  - The number of cores in a node is increasing rather than the number of nodes.

Directory at the home memory tends become large, because the total size of the memory is much larger than cache.

In order to reduce the memory requirement, various methods have been proposed.

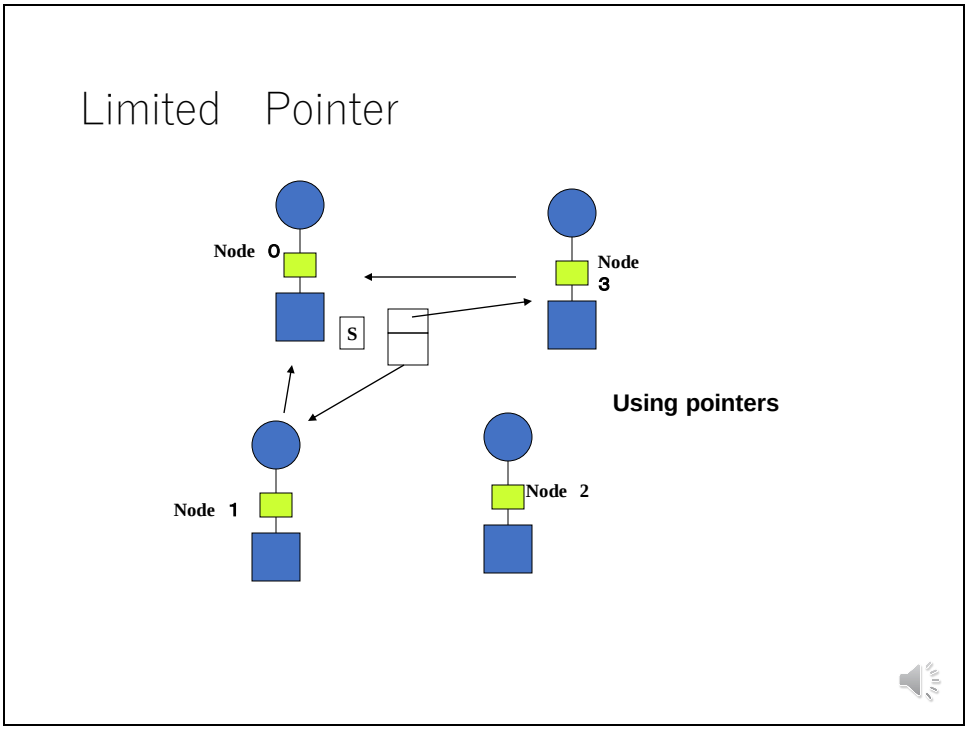
Full map directory



Bit = Nodes
If the size is large, a large memory is required.

Used in Stanford DASH

The basic method which I introduced is called the full map directly. In this method, the number of bits are the same ss the number of nodes.

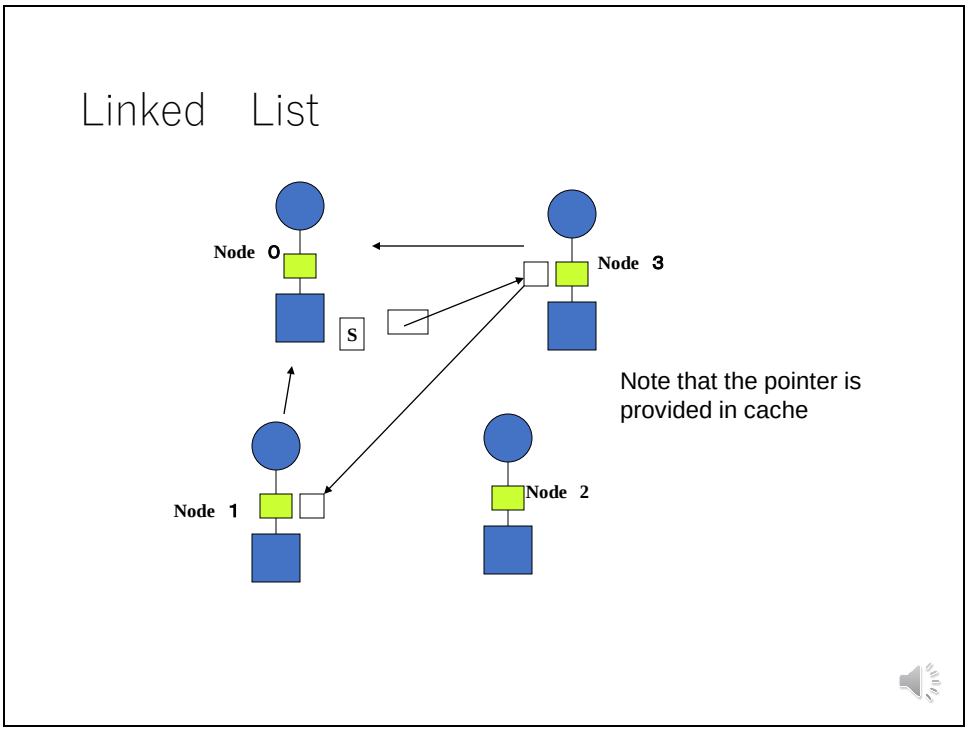


Instead of the bit-map, how about providing pointers. In this example, two pointers to provide to store the node number.

Limited Pointer

- Limited number of pointers are used.
  - A number of nodes which share the data is not so large (From profiling of parallel programs)
- If the number of nodes exceeds the pointers,
  - Invalidate (eviction)
  - Broadcast messages
  - Call the management software (LimitLess)
    - Used in MIT Alewife

The apparent problem is that the number of nodes which share the data is limited. But how can we do when the number of nodes exceeds the pointers. Some methods have been proposed. One is called eviction. It invalidates one of pointer. But it of course may cause the performance degradation when the evicted node sends the request again. Another method is to give up keeping the shared nodes. That is, invalidation messages are broadcasted to all nodes. This invalidation messages are just discarded if the node is not related, but it may cause the traffic congestion. The third method is to invoke the management software. It was used in MIT Alewife.

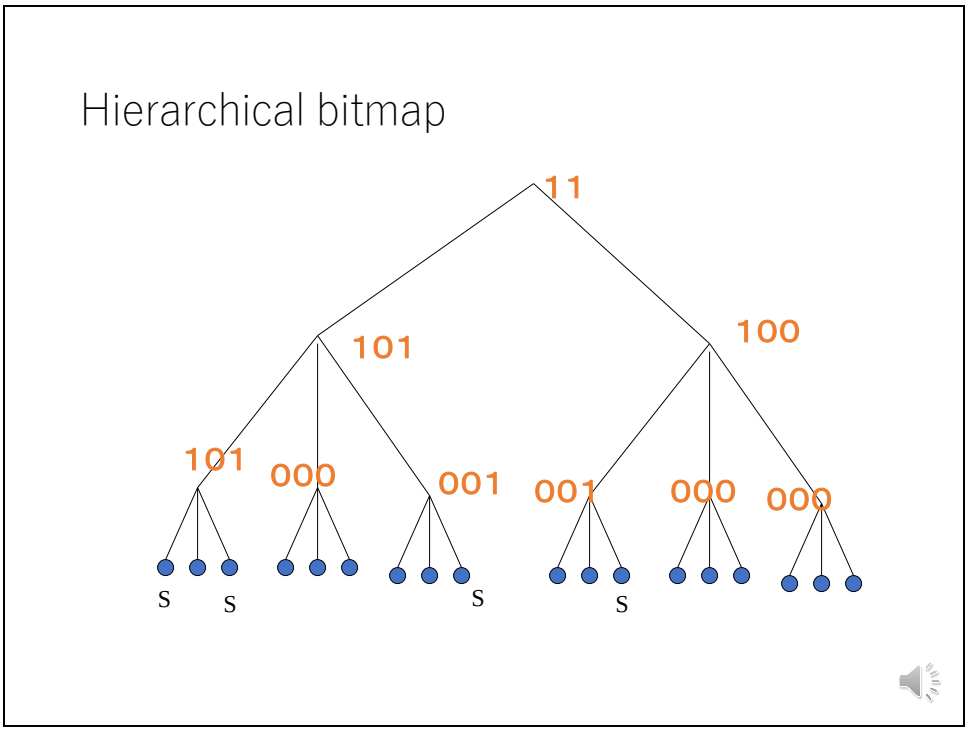


An alternative method is to make a linked list between nodes who want to share the block like the queue based lock.

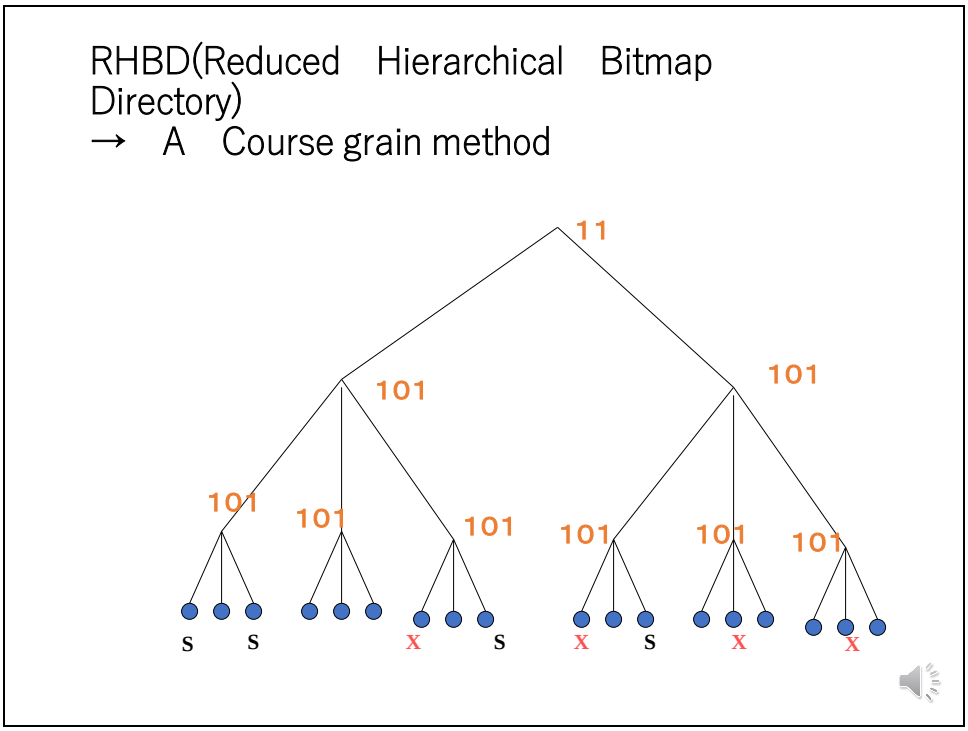
Linked List

- Pointers are provided in each cache.
- Small memory requirement
- The latency for pointer chain often becomes large.
- Improved method: tree structure
- SCI(Scalable Coherent Interface)

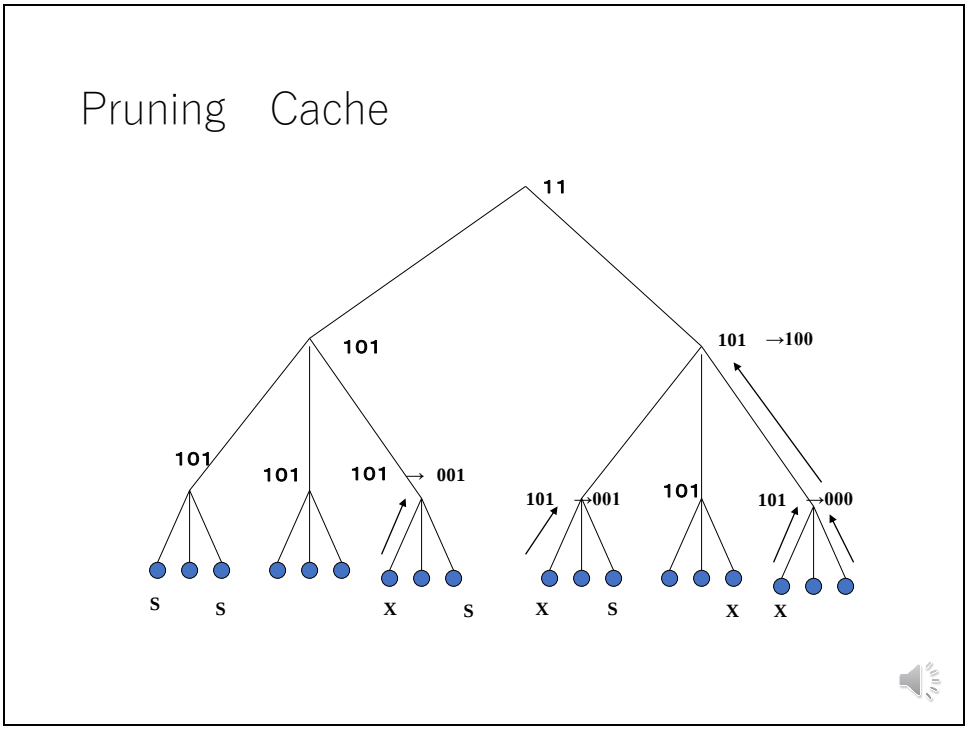
It requires relatively long time to manage and to trace the pointer chain. The improvement method to make the tree link structure was proposed. However, the benefit of method is small resource requirement. So, it is adopted in the standard protocol called SCI.



If the network has hierarchical structure, the directory can be held at the branch of the hierarchy. It was adopted in COMA machine explained later. The problem is that it requires more amount of memory than the simple bit-map method.



In order to reduce the required amount, a course grain method can be used. For example, we can use the same bitmap at the all branch of the same hierarchy.

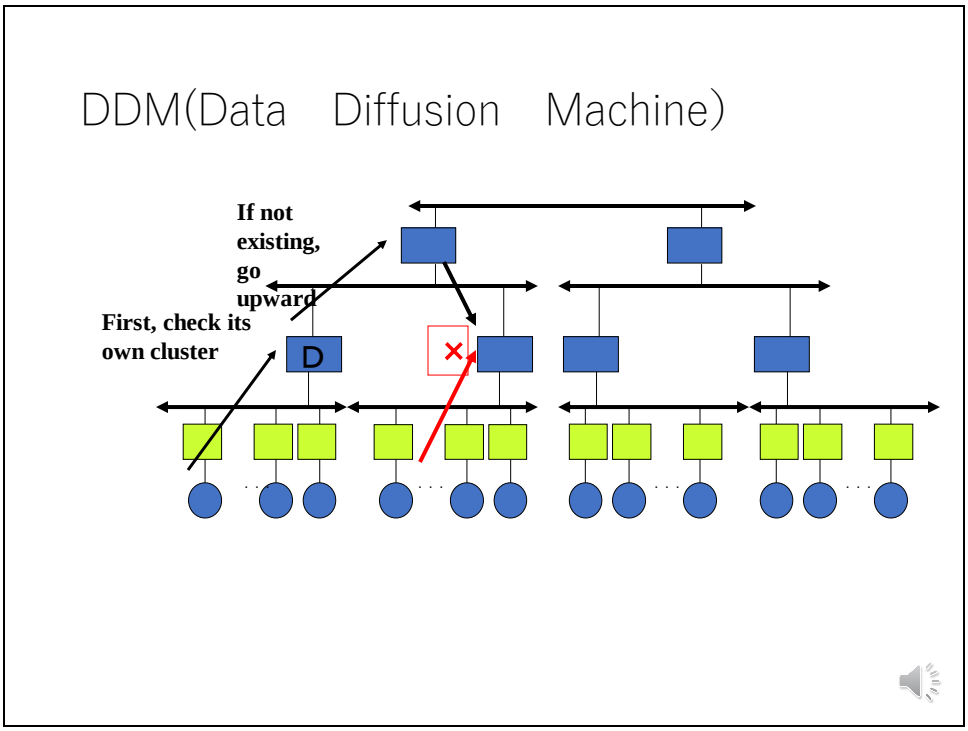


The problem is the method is increasing unnecessary invalidation message, to cope with the problem, we can introduce a kind of cache mechanism to the directory itself. This idea to introduce the cache can be used for the basic bit-map.

COMA(Cache Only Memory Machine)

- No home memory and every memory behaves like cache (Not actual cache)
- Cache block gathers to required clusters. Optimal data allocation can be done dynamically without special care.
- When miss-hit, the target block must be searched.
- DDM、KSR-1

OK. The last type of the NUMA is COMA. Of course it takes a large cost if there is only cache memory. It means that there is no home memory and every memory behaves like cache. The cache block gathers to required clusters. That is an optimal data allocation can be done automatically. But, when miss-hit happens, the target block must be searched.



The data diffusion machine uses hierarchical bitmap structure to manage the COMA system. If the block cannot be found to a hierarchy, the upper hierarchy is searched. It requires rather complicated handling if conflicts happen in the upper hierarchy.

Glossary 2

- Directly based cache protocol:ディレクトリを用いたキャッシュプロトコル、スヌープキャッシュではなく、ホームメモリ上のテーブル（ディレクトリ）を用いてキャッシュの一貫性を管理する方法
- Full map directory:ディレクトリ管理法の一つ。PEに対応するビットマップをもつ
- Limited Pointer:ディレクトリ管理法の一つ。限定された数のポインタを用いる。evictionは不足した場合、強制的に無効化する方法
- Linked-list:リンクドリスト、ポインタの連鎖構造による管理法、SCI(Scalable Coherent Interface)はこれを用いたディレクトリ管理の標準規格
- Queue-based lock:リンクドリストでロックの順番を管理する方法。NUMAの同期手法として一般的に用いられる。
- Hierarchical:階層的、今回はバス構造、ディレクトリ構造のところで出てくる。

Summary

- Simple NUMA is used for large scale supercomputers
- Recent servers use CC-NUMA structure in which each node is a multicore SMP.
  - Directory based cache coherence protocols are used between L3 caches.
  - This style has been a main stream of large scale servers.

Now, let's make a summary of today's lesson.

Exercise

- Show the states of cache connected to each node and directory of home memory in CC-NUMA.
- The node memory in node 0 is accessed:
  - Node 1 reads
  - Node 3 reads
  - Node 1 writes
  - Node 2 writes
  - Node 3 reads
  - Node 3 writes

This is today's exercise.