Parallel Computer Architecture
Prof. Hemangee K. Kapoor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Week - 09
Lecture - 50

Lec 50: Sequent NUMA-Q Architecture

Hello everyone, we are doing module 6 on Directory Coherence case studies. This is lecture number 4 where we are going to start the Sequent NUMA-Q architecture and look at the different protocol and the correctness aspects. This is a flat cache based directory protocol. In this lecture, we are going to look at the architecture. So, what is the philosophy behind the Sequent NUMA_Q design? So essentially, it is an idea which is going to compose several off the shelf symmetric multiprocessors to construct a big parallel architecture and still be able to maintain coherence among the nodes. So, you have, for example, if you have a Pentium Pro processor, you can take several of them create a PCB board out of it and then several such nodes I am going to connect through some kind of a network and all these nodes which we have connected are going to run parallel applications.

So, these applications will share data. They will share data across memory modules. Where are these memory modules? They are with these individual nodes which are in itself multiprocessor nodes. Okay. So, all the memory modules within these small nodes will become the global memory.

You are going to access any location throughout this network and maintain coherence. Given that we have taken off the shelf processors, they are not aware of the type of network in which we are going to plug them in. They come with their own cache coherence protocol or if it is a uniprocessor node, it may not follow a protocol. So, given that I use a symmetric multiprocessor as a single node, it has its own coherence protocol, it could be a directory protocol or a snooping bus based protocol. Okay. So, I already have a node which is doing, for example, a snooping protocol.

Several such nodes I will club together and form a bigger system. Now, how are these different nodes in my system going to understand each other? Are they going to snoop each other? No, because they are independent implementations. They do not know where they are plugged into. Right. So, the snooping protocol within one node only understands itself. That is, the only their own system and they do not know what is outside.

So, we need something else, an external entity which will manage the coherence among all these nodes. So, how do we build such a system is an example of this Sequent NUMA-Q architecture. And for this, the idea is to use a distributed linked list data structure which suits good in this particular setup. So, we are going to have a distributed linked list or cache based directory coherence protocol at the outer level and inside every node, we are going to have a snooping protocol. Okay. So, we have this two level hierarchy.

To make matters simple, we are not going to look at how the snooping talks with the directory. Okay. So, snooping and directory interaction we will do in brief later at the end of this module. But right now, we are only going to concentrate on the cache based directory protocol and the directory protocol is going to see the node as a single entity and not a multiprocessor entity. All right. So, we are going to compose symmetric multiprocessors.

So, I am going every pink box here is a node which is itself a multiprocessor node. It is handling its own cache coherence protocol and all these nodes are connected using some network. In my current example, we are going to use a ring network. The concepts which will become clear or we, what we will understand in this topic is that I can compose logically disparate protocols. The snooping protocol is composed with a directory protocol.

Even the directory protocol could be memory based or cache based. So, in this case, it is a linked list cache based protocol and not a bit vector directory type protocol. So, you will get a feel of how I can compose different protocols. Then for this pink box which is the node, it itself is a multiprocessor node, but I said that for the directory it is a single processor or rather a logically single entity. So, who would represent this node to the outside directory protocol? Okay. So, we are going to use a concept of a cache which will provide the protocol abstraction and see how I can connect the node to the external system.

Every node as we discussed has its own existing cache coherence protocol and still it is usable to compose with heterogeneous nodes. So, I am not saying that each node itself should follow the same snooping protocol. I could have different nodes following different types of internal protocols, but as long as the external protocol is the same and your internal node gives a similar picture from every node to the whole global system, we are able to compose the protocols. Right. So, this idea is essentially kind of a scalable ready idea. So, my node which is a small pink box and my interface which I am going to construct, that is going to manage the disparate different nodes and connect to a global

network and manage a directory at a global level.

So, this is giving me the scalability aspect and hence the protocol which we are going to discuss is the SCI protocol for scalable coherent interface. This is an IEEE standard protocol. It is a very detailed and very intriguing protocol. We are not going to look at all the states, we are going to look at a subset of how this protocol works. Okay. So, zooming into further, so now this SCI ring which was there on this slide, I am going to elaborate further from a generic design to a more specific NUMA-Q architecture.

So, this is a processor coming out from the sequent systems and hence it is called a Sequent NUMA-Q. Okay. So, here you can see on the left hand side you have a ring network which is a unidirectional ring and there is the term quad written in every box there. So, that quad is your SMP node. So, we have the symmetric multiprocessor nodes which are the building blocks. We have several of them within each node.

So, this is the quad. If you can see this, this quad is, this picture here. It is made up of a quad bus. So, this is the bus and we have four processors, 1, 2, 3, 4 processors connected to the bus and we have a snooping protocol inside the quad. So, quad is implementing a snooping protocol.

These quads when they get connected over the ring network, they are going to maintain a directory protocol. The quads are connected over a unidirectional link of this 1 Gbps bandwidth and if you want even larger systems, normally we can connect 8 quads on the ring and if you want more you can have a bridge in between multiple rings. So, you have one ring with 8 quads, then you can have a bridge and then connect another ring with other 8 quads and construct larger systems. Okay. So, the quad bus which is this one is an internal bus to the SMP which is 532 Mbps link which connects the processors to a memory module. See there is a memory, there is I/O, everything.

So, this node in itself is a multiprocessor. So, there are 4 processors. In current example, we have the Pentium Pro processor. There are I/O links from third party connected, you have memory banks, you would have a disk and everything. So, in this example, the memory bank here is a 4 GB memory with every quad and multiple quads will have 4 GBs attached and they together will form a globally addressable big size main memory.

So, you have one memory size slice here, there will be such memory slices everywhere and that is a globally addressable memory to the complete system. Now, who does the magic of connecting this SMP to the ring? So that across the rings, each quad is able to see the memory and access the memory of another quad and still be able to share data.

Right. So, once we can access memory, we can share data. Each quad has got 4 processors, every processor has a cache and if I am having 8 quads, 8 * 4, these many caches are there in the system, but they are physically disjoint, but still they will be sharing the same memory blocks. Okay. So, that is the big picture which we have to handle.

And who does this magic for us? It is the IQ-Link board. So, this is the only customized block from the Sequent systems who would implement the directory protocol. So, what we have? We have readymade quads, but we implement this link which implements the directory protocol and are able to compose larger systems. So, what is this IQ- Link board looks like? Right. So, this IQ-Link, it is here connecting, it is at the boundary of this quad. So it connects, all the IQ-Links connect with each other forming the SCI ring and it is called SCI ring because it is implementing the SCI directory protocol.

So like looking inside the IQ-Link board, so this is the picture of the IQ-Link board. It has got three main components, one is the network interface, this is the SCI ring, this is the ring comes from here, this is unidirectional ring. So, I am connecting the network interface at the ring. Below this network interface is the quad. So, what is there in the quad? The processors are here.

So, this is your four processors and maybe other things, the cache and the memory. So, that is the quad bus. So below this is your SMP. Above this, we have the IQ-Link board. At the top, we have the network interface which connects the quad to the bus.

Now this network interface is again a readymade module from Data Pump. It provides a link and a packet level transport protocol. It implements the SCI standard. What is its job? Its job is to pull off packets from the network if the destination belongs to this quad. So when a packet comes on this network, the Data Pump will check whether this packet has a destination address for my quad.

If yes, it pulls it inside. If no, it simply forwards that packet further onto the network. Okay. So that is the job of the network interface. Below that, we have the directory controller. The directory controller is called the SCLIC because it is implementing the SCI protocol. This controller sits between the Data Pump and the OBIC.

Now what is OBIC? It is the Orion bus interface which is able to connect a bus with my protocol module. So we have a protocol implementation module. This module has to talk with the network interface as well as it should be able to speak with the bus. So for that, we put a additional interface called the OBIC. OBIC is a bus interface sitting

between the directory controller and the quad bus. Okay.

This directory controller is going to be managing all the interrupts. So, it should have access to the interrupt handling routine. Then it should have access to the directory tags because it is a directory controller. It needs the directory information. So directory information, then tags of the cache and so on.

So we are going to see more details of this. And overall, this is an SCI protocol implemented using programmable engines within the SCLIC box. Okay. Third module is the OBIC which is the Orion bus interface controller. This is an interface between the directory controller and the quad bus. And it, its main job is to manage the remote access or remote data cache.

So it connects the directory controller, the remote cache and the quad bus. Okay. So it manages the remote data cache. It manages bus snooping. And when you have snooping, the request response logic also it has to implement. So overall, if you look at this IQ-Link board, this big picture, what is it? Like the SCI ring, you have the quad.

The quad is linked using the IQ-Link board. So for the external world, if I am standing here, right, what is the IQ-Link board? It is the processor or the node for me. So IQ-Link board is serving as a processor for the external world. Right. It's a pseudo processor for the incoming network transactions. And for local data, it is acting as a memory controller. So if we are standing here, if I am standing here, what is the IQ-Link board for me? It is the memory controller because it is helping me to go out and access the memory modules across the other outside quad.

So suppose this is my memory module one, I can access memory module two, three, four through the IQ-Link board. Alright. So, IQ-Link board is a pseudo processor for external connections and it is a pseudo memory controller for internal processors. Okay. So that was about the modules. Now let us see how are we going to integrate the concept of a directory. When I say we will behave as a pseudo processor for the outside world.

So this IQ-Link board is going to act as a pseudo processor for the incoming transactions. So when I say pseudo processor, we need a processor, a cache and all the associated things. Okay. So we will understand that on this slide. Okay. So IQ-Link board is going to play the role of the hub chip in the SGI origin protocol.

If you know hub did everything there. Similarly, here IQ-Link is going to play this role in Sequent NUMA-Q. So it can generate interrupts between the quads. Then we have a remote cache. So there in, below the quad bus we have the SMP.

Each SMP node has its own cache. Now these caches are different. The ones which I am drawing in green color, these caches are the local caches of the Pentium Pro processors. Apart from this, you can see I have another cache here in the IQ-Link board which is the remote data cache. Okay. So this remote data cache is an extra cache. And what is this going to do? This is going to cache all the data which is coming from outside. Okay.

It is going to have 64 byte blocks, its size is 32 MB and a four way associative. So 32 MB capacity. The remote cache is visible to the SGI protocol. We will see more details of the remote cache in a while. But before that, let us see how the quads are connected.

In the picture I have just shown six quads, but normally we have eight quads on a ring. And the data pump drives this link at this particular bandwidth. It is following a strict request response protocol. A request comes, a response is sent. So when I have a strict request response, when a request comes to the IQ-Link board here, it has to send a response.

Remember the ring is unidirectional, so you cannot send response in the backward direction. You have to send response only in this clockwise direction. So when a packet comes to a quad and if the destination is this particular quad, it takes that packet in. If it can cater to the request, it has to send an acknowledgement. The acknowledgement is sent as a positive echo on the ring and when the sender receives this echo, it says that the transaction is complete.

If the quad cannot handle this request, then it sends a negative acknowledgement which is the NACK as a negative echo onto the ring. So when the sender data pump gets this NACK, it will then retry the transaction. Okay. So given that, this is the sender and suppose this is the destination. So source sends a packet. This packet goes, if destination takes it, it will send a positive echo.

If destination cannot handle this request, it sends a negative echo on the ring. Okay. Now coming back to the remote cache. So remote data cache. It is popularly called the remote access cache or the RAC. So this RAC is going to represent one node in my SCI protocol.

You have, you have this ring, SCI ring. To this I have connected a quad that is the physical connection. But in terms of the directed protocol, which is the cache and which is the memory we are talking about. Every quad has got four processors, they have their own caches. So am I going to talk to these four caches? No we are not. We are going to

have an additional cache which is the remote access cache and this cache here, is going to represent this particular node to the outside world.

So we can say that that is the level 3 cache because if you have two level caches in your SMP, this is the SMP, the L1, L2 and then we have an additional L3 cache. Now this L3 cache or the remote access cache has a directory associated with it. This directory is not a bit vector directory, but in the NUMA-Q architecture, it is a cache based linked list type of a directory structure. What is this cache going to store? It is going to store the locally and remotely allocated blocks. It is going to house all those blocks which are coming inside this quad from other nodes.

So if I am fetching a block from a remote home. Right. So if I am bringing data, there is a data here, if I bring the data into my quad, then that data will get stored in the remote access cache first. And then from here it will be sent to the processor which requested it. So once I have this, you need to maintain inclusion between the processor caches and the L3. So RAC and the processor caches inclusion has to be maintained. Because if L2 deletes this block, then you have to update it.

If L2 writes to this block, then RAC should be aware of it. If RAC deletes a block, then RAC has to also delete the block from the L2 cache. Okay. So this way having inclusion maintenance is critical to this particular protocol. And the SCI protocol is oblivious of how many processors are there in the system or the particular SMP node. This SMP node right now had 4 processors, I can connect any other one with 8, 12 processors. Still, the SCI protocol will be able to work and why? Because we have the RAC, because SCI only sees RAC, does not see how many processors or caches are there inside that SMP.

So understanding the concept further, in this picture we can see that there are 4 processors. Right. So every SMP node has got these 4 processors. Along with that, there is a bus definitely for the snooping protocol. And to this bus, we have connected, apart from other IO modules, we are connecting the remote access cache, which is integrated part of the IQ-Link board. Okay. So the IQ-Link board and the remote access cache together virtually plug into the bus through the OBIC.

So they are not directly connected to the bus, but the OBIC controller connects the RAC to the bus. Within the SMP node, these processors are following the snooping protocol. So P1, P2, P3 and P4, they will be doing a MESI cache coherent snooping protocol and this is normally the split transaction bus type of a design. So these processors P1, P2, P3, they will also be able to share data and in case the share data items for these processors comes from the 4 GB memory which is connected within this SMP, then they will do a snooping protocol of coherence and the RAC or the IQ-Link

board will not bother into it. Right. So the block is local, it is cached locally, so it is snooped and managed coherence locally.

If the data is coming from another quad to this quad, suppose there is an external quad through which I am bringing data and P2 wants to use it, then it has to come via the RAC. Okay. So overall, the RAC is going to represent the quad to the outside ring. So what is a quad? A quad is nothing but the RAC for the external world. It doesn't know how many processors are sitting.

From the quad, only the RAC is visible. P1 to P4, these processors are kept coherent with the RAC using snooping. So, in case an external block comes and sits here from outside, so this is a remote block. If I bring a remote block to this processor and suppose P2 and P3 use that particular block, then we are going to use snooping to keep these copies coherent. So RAC, P2 and P3, these three copies will be kept coherent using the snooping protocol. The directory protocol does not know how many processors are there within every node. Because the directory protocol only knows that I have one RAC and that is the data I want to manage.

The data from the RAC and among the processors is managed by the snooping protocol. Inclusion has to be maintained between the processors and the RAC and if RAC is deleting a block, it has to invalidate the block from P1 to P4. So, if it deletes this block, it will send an invalidation message as a snooping request on this and each processor will invalidate the data block. In case processors change the data value, that is I write to this, suppose we do a writing onto this, then the state of the block in RAC should reflect this, because RAC should know that P2 has changed the data, so that it can say that block A is in modified state in this particular node. Okay.

So that was the architecture aspect and the conceptual hierarchy. Coming to the directory structure now, so moving from the design to the protocol. So here, we are having a cache based directory protocol and in cache based we are looking at a distributed linked list. Now how is this linked list constructed? We have a quad, every quad has a memory. A block from the memory will be shared by the processors within the quad, then it is handled by the snooping protocol. If a block within the memory is accessed by a processor in another quad, it will go and sit in its remote access cache and we will follow the directory protocol. Okay.

So now we are going to look at the directory protocol. When a block is shared from a memory node by an external quad, we are simply going to maintain a pointer from this memory to that particular quad. We are not going to refer to processors but we are going to refer to a quad which is the RAC. So I am essentially saying my memory block from

this module is pointing to a RAC in quad number 3. Alright.

So this is the pointer. So pointers are going from a memory to a RAC within a quad. If there are more sharers in the system, then we have a linked list. Memory pointing to the first sharer, first sharer pointing to the second and so on. So sharers are sitting in the RAC. Each block in the RAC is pointing to the block in the next RAC forming the linked list structure. So we have, when I have a linked list, I have a head node and a tail node and some formalisms associated with it.

So all the sharers are in the form of a distributed linked list. Linked list has got a head node. So whole memory, which is the memory module, is pointing to the head node, which is nothing but a block inside RAC. Each block in RAC stores the data as well as the forward and backward pointers. So if this is the RAC and this is the block, it has the data, a pointer in this direction.

So we have storage for these pointers, so that I can form a doubly linked list. Okay. So home points to the first sharer, which can be the or rather it is called the head node and then we have further nodes, finally the tail node. So forward pointer, that is the pointer going towards the tail is called the forward pointer or a downstream pointer. The pointers in the reverse direction from the tail to the head are called backward pointers or upstream pointers. So these are some conventions we, or names which we have to remember. Home has to always point to the head node and in the protocol only the head is given permission to read and write.

In case a block has to be written, it has to be there at the head of the list. If that particular RAC or the quad is in the middle of the list, it is not permitted to write. If you want writing permission, you have to become head of the list. And other nodes can be read only accesses. So only the head has the writing permission, other nodes do not have the writing permission. In a linked list, we have the head node, the tail node and all the other nodes are called middle nodes.

What is the home going to store? The home is going to store the state of the block, whether it is clean, dirty, etc. plus the pointer to the head node. So home memory has the data, along with data it has the state of the block which is the directory state not the cache state. So what are the states I am talking about? We have two types, one is a directory state and one is a processor cache state. Processor caches are governed by the MESI protocol internally using the snooping logic and these are not related to the state in RAC.

So MESI protocol is done by the processors at the up to L1, L2 level, that is the four

processors follow the MESI protocol. At the directory state, we have three states. We have first state is called the home when I have no sharers of the block. Second is called shared which when there are more sharers and the third is equivalent to a dirty block.

So we have a clean block, a shared block and a dirty block. So using the terminology, a directory state in home that is the first one, what does this say that no remote caches have a copy of this block. Okay. So directory is maintaining information of the memory blocks in my local 4GB memory. The blocks in my memory can be used by the processors within my quad. So that is handled by the snooping protocol, nothing to do with directory.

When my blocks are taken outside my quad, then I need to maintain the directory information. In this context, the directory comes in three states. One is the home state, which says my data blocks are not taken outside my quad. They may be internally shared, they may be internally modified, but from the directory perspective, the block is at home. Okay.

So that is the home concept. The second state is called the fresh. Fresh means somebody is sharing my data block, but it is clean. So there are only read only sharers for this data block. And once I say, when I say shared, it is going outside my quad. The third is dirty state when my data is taken by an outside quad and modified.

So it is called a gone state. Okay. One is you are at home. The second is there are read only clean sharers. So we call it fresh. So fresh means sharers are present.

And the third is the block is dirty elsewhere. So I will say it has gone out from my quad. So that is the third one, which is a writable copy. And as you know, only the head node can be the writer in the system. Okay. Now these RAC blocks have got multiple states. So directory states are these three, which are the stable states. But conceptually, they will be when you implement the protocol, we have to go from one stable state to another.

There could be several intermediate states which are pending states. So overall, we have 29 stable states and many, many more pending states. So if you are interested, you can look up the IEEE standard for SCI. Maybe it is more upgraded one, but the initial one was published in 1993. It is a huge protocol with lots of connections and so on.

But we are not going to look at the complete protocol. We are only going to look at few scenarios, which will give us a feel of how a cache-based coherence protocol is implemented. So before we close this lecture, two points to note, that every stable state is

identified by two parts. We just don't use a single word. We are going to use two words.

The one, the first one is called the position in the linked list because I have a linked list. I have to say in which position I am and what is the state. So position plus state together tell me the state of the block. So either I am the head node or the tail node or I am the only node or I am in the middle. So this is the position. What is the position of the block? And the second part is what is the state? Is the block dirty? Is it clean? Is it a copy? Is it a read only? Is it pending? And so on.

There are several such. There are only a few of these. Okay. So with this we stop this lecture. In the next lecture we will look at the working of the protocol. Thank you so much.