

SISCI API LIBRARY

Dolphin Interconnect Solutions
Roy Nordstrøm

Copyright 2014 All rights reserved.

Agenda

1. SISCI API Library
2. PIO model - Programmed Input/Output
3. DMA model
4. Remote interrupts
5. Error handling

connect better

Copyright 2014 All rights reserved.

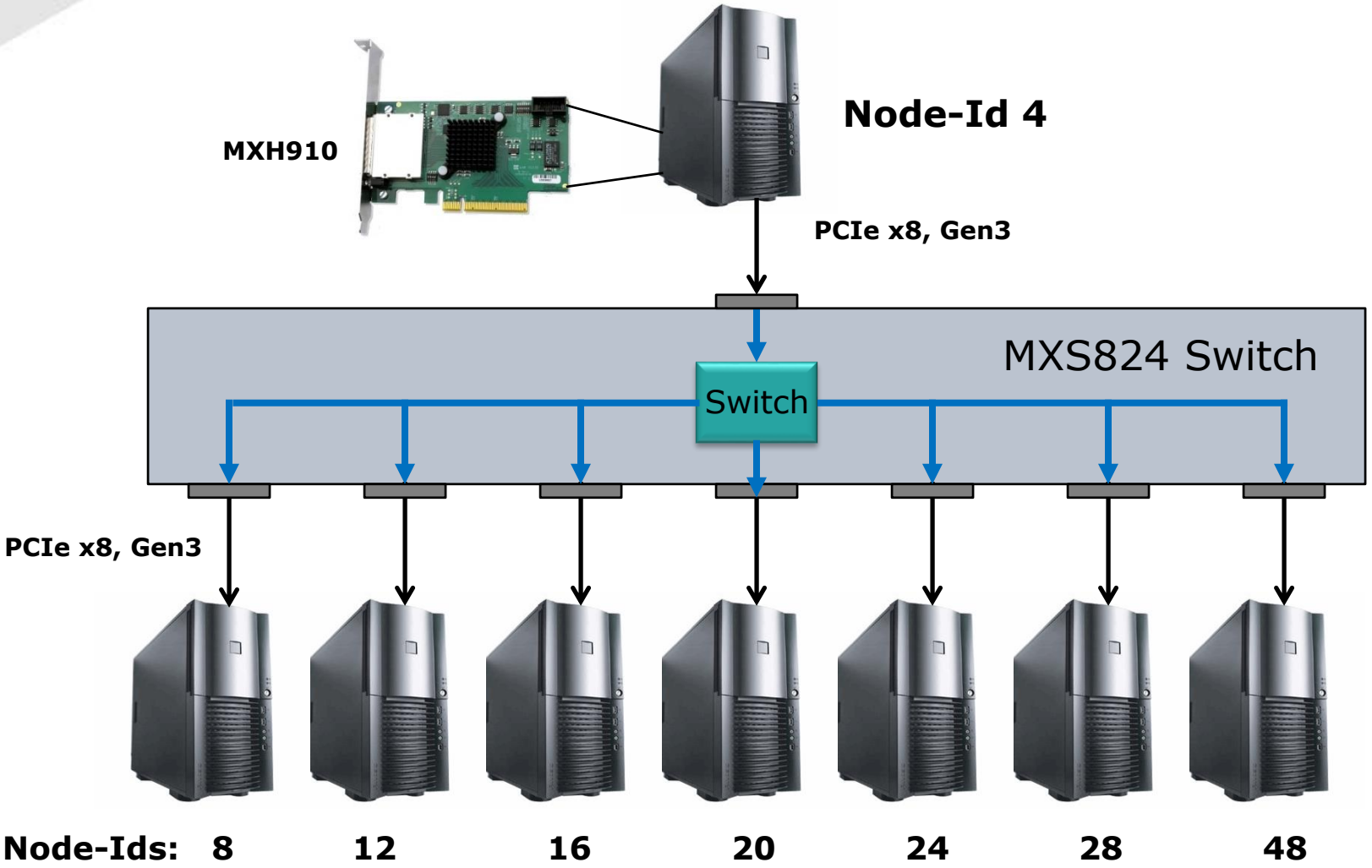
PCIe Link Speed

	Raw bitrate per lane	Raw x8 Bandwidth	Raw x16 Bandwidth	x16 payload bandwidth
PCIe 1.0	2.5 GT/s	2 GB/s	4 GB/s	~ 3 GB/s
PCIe 2.0	5 GT/s	4 GB/s	8 GB/s	~ 6 GB/s
PCIe 3.0	8 GT/s	8 GB/s	16 GB/s	~ 14 GB/s
PCIe 4.0	16 GT/s	16 GB/s	32 GB/s	~ 28 GB/s
PCIe 5.0	32 GT/s	32 GB/s	64 GB/s	~ 57 GB/s
PCIe 6.0	64 GT/s	64 GB/s	128 GB/s	~ 114 GB/s

connect better

Copyright 2014 All rights reserved.

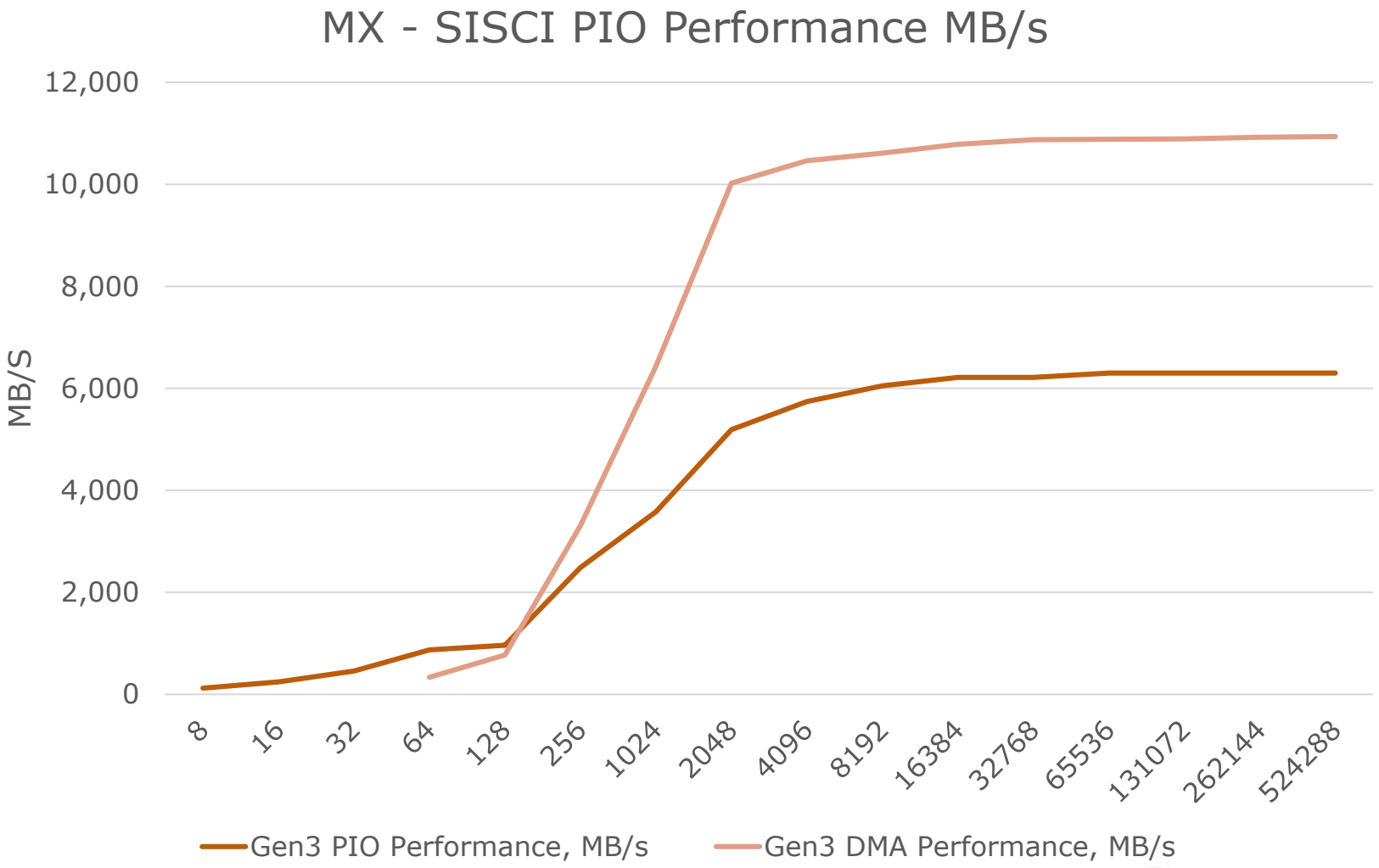
Dolphin Cluster - Node-Id Assignment



connect better

Copyright 2014 All rights reserved.

SISCI - Performance results for MXH910



connect better

Copyright 2014 All rights reserved.

SISCI – Performance test application

- **scibench2 –rn 4 –client**
- **scibench2 –rn 8 –server**
- Function: sciMemCopy_OS_COPY_Prefetch (5)
- ---
- Segment Size: Average Send Latency: Throughput:
- ---
- 4 0.07 us 61.01 MBytes/s
- 8 0.07 us 122.38 MBytes/s
- 16 0.07 us 244.85 MBytes/s
- 32 0.07 us 467.00 MBytes/s
- 64 0.07 us 870.77 MBytes/s
- 128 0.13 us 962.52 MBytes/s
- 256 0.10 us 2485.39 MBytes/s
- 512 0.14 us 3579.03 MBytes/s
- 1024 0.26 us 3921.95 MBytes/s
- 2048 0.39 us 5190.43 MBytes/s
- 4096 0.71 us 5740.19 MBytes/s
- 8192 1.35 us 6049.09 MBytes/s
- 16384 2.64 us 6115.79 MBytes/s
- 32768 5.54 us 6114.19 MBytes/s
- 65536 13.01 us 6136.38 MBytes/s
- Node 4 triggering interrupt
- The remote segment is unmapped

connect better

Copyright 2014 All rights reserved.

SISCI – Latency test application

- **scipp –rn 4 –client**
- **scipp –rn 8 -server**

- Ping Pong data transfer:

size	retries	latency (usec)	latency/2 (usec)
0	164	1.207	0.604
4	268	1.126	0.563
8	379	1.116	0.558
16	425	1.115	0.558
32	427	1.115	0.558
64	429	1.117	0.559
128	432	1.138	0.569
256	442	1.177	0.589
512	472	1.296	0.648
1024	496	1.403	0.701
2048	573	1.727	0.864
4096	719	2.364	1.182
8192	1019	3.670	1.835

connect better

Copyright 2014 All rights reserved.

SISCI – DMA test application – Gen3/x16

- **dma_bench –rn 4 –client**
- **dma_bench –rn 8 -server**

Message size	Total size	Vector length	Transfer time	Latency per message	Bandwidth
64	16384	256	49.16 us	0.19 us	333.26 MBytes/s
128	32768	256	42.52 us	0.17 us	770.63 MBytes/s
256	65536	256	42.22 us	0.16 us	1552.32 MBytes/s
512	131072	256	39.60 us	0.15 us	3309.65 MBytes/s
1024	262144	256	40.75 us	0.16 us	6433.14 MBytes/s
2048	524288	256	52.30 us	0.20 us	10024.24 MBytes/s
4096	524288	128	50.10 us	0.39 us	10464.83 MBytes/s
8192	524288	64	49.40 us	0.77 us	10613.76 MBytes/s
16384	524288	32	48.60 us	1.52 us	10786.93 MBytes/s
32768	524288	16	48.21 us	3.01 us	10875.31 MBytes/s
65536	524288	8	48.17 us	6.02 us	10883.44 MBytes/s
131072	524288	4	48.13 us	12.03 us	10892.94 MBytes/s
262144	524288	2	47.99 us	23.99 us	10925.17 MBytes/s
524288	524288	1	47.93 us	47.93 us	10938.39 MBytes/s

connect better

Copyright 2014 All rights reserved.

SISCI – DMA test application – Gen4/x16

- **dma_bench –rn 4 –client**
- **dma_bench –rn 8 -server**

Message size	Transfer time	Bandwidth
64	2.64 us	24.25 MBytes/s
128	2.60 us	49.25 MBytes/s
256	2.59 us	98.69 MBytes/s
512	2.60 us	196.70 MBytes/s
1024	2.62 us	390.54 MBytes/s
2048	2.67 us	768.19 MBytes/s
4096	2.82 us	1450.42 MBytes/s
8192	2.92 us	2804.52 MBytes/s
16384	3.17 us	5171.72 MBytes/s
32768	3.94 us	8318.86 MBytes/s
65536	5.11 us	12822.54 MBytes/s
131072	7.36 us	17815.96 MBytes/s
262144	11.98 us	21881.80 MBytes/s
524288	21.41 us	24493.72 MBytes/s
1048576	39.95 us	26247.21 MBytes/s
2097152	77.80 us	26954.64 MBytes/s

connect better

Copyright 2014 All rights reserved.

SISCI – DMA test application – Gen5/x16

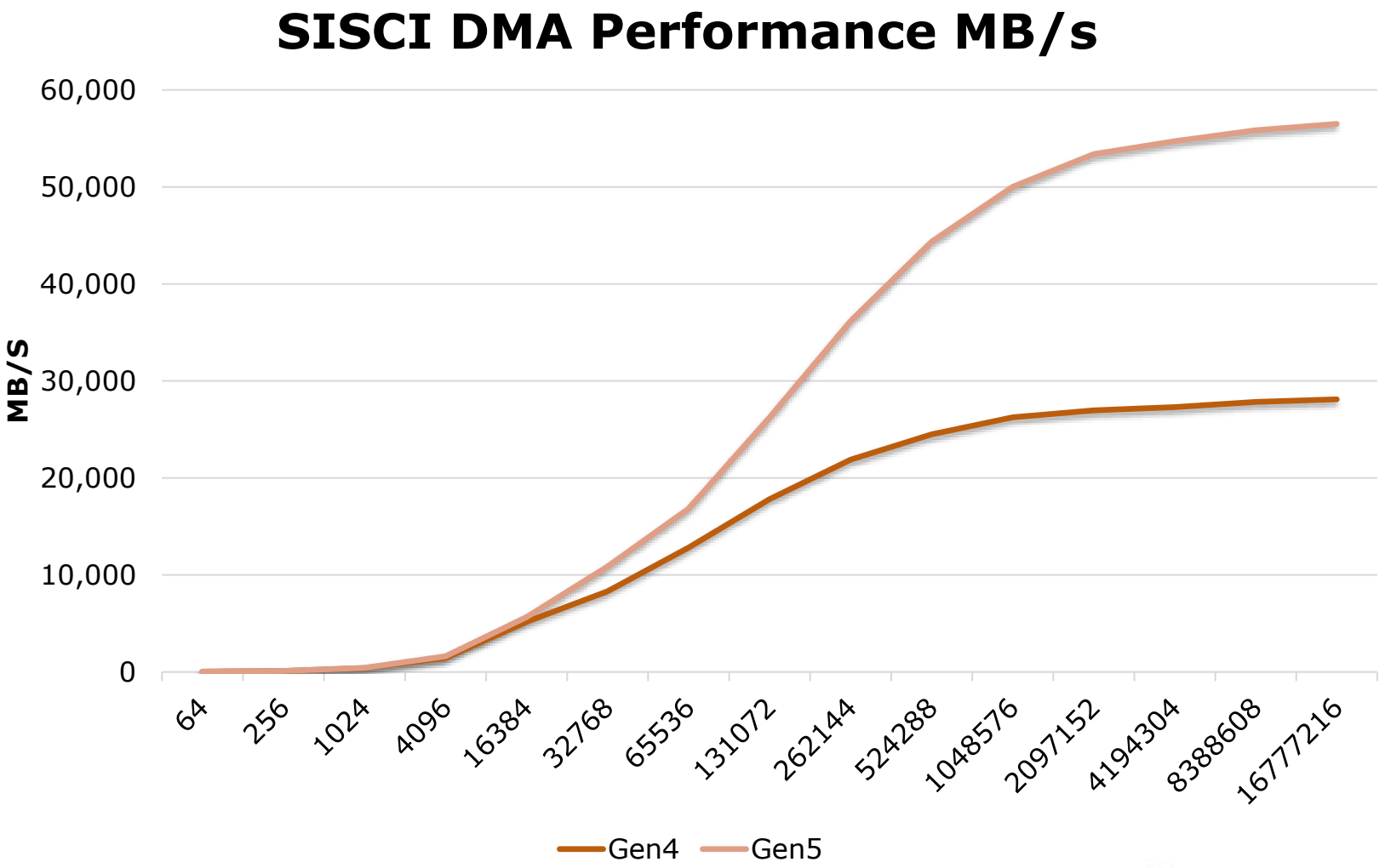
- **dma_bench –rn 4 –client**
- **dma_bench –rn 8 -server**

Message size	Transfer time	Bandwidth
64	2.45 us	26.17 MBytes/s
128	2.44 us	52.46 MBytes/s
256	2.44 us	104.92 MBytes/s
512	2.44 us	209.81 MBytes/s
1024	2.45 us	418.51 MBytes/s
2048	2.46 us	831.84 MBytes/s
4096	2.52 us	1623.01 MBytes/s
8192	2.64 us	3107.15 MBytes/s
16384	2.89 us	5660.00 MBytes/s
32768	3.02 us	10862.20 MBytes/s
65536	3.89 us	16834.32 MBytes/s
131072	5.00 us	26236.96 MBytes/s
262144	7.25 us	36178.25 MBytes/s
524288	11.82 us	44356.01 MBytes/s
1048576	20.97 us	50014.83 MBytes/s
2097152	39.29 us	53381.39 MBytes/s
4194304	76.64 us	54723.99 MBytes/s
8388608	150.21 us	55845.27 MBytes/s
16777216	296.91 us	56505.42 MBytes/s

connect better

Copyright 2014 All rights reserved.

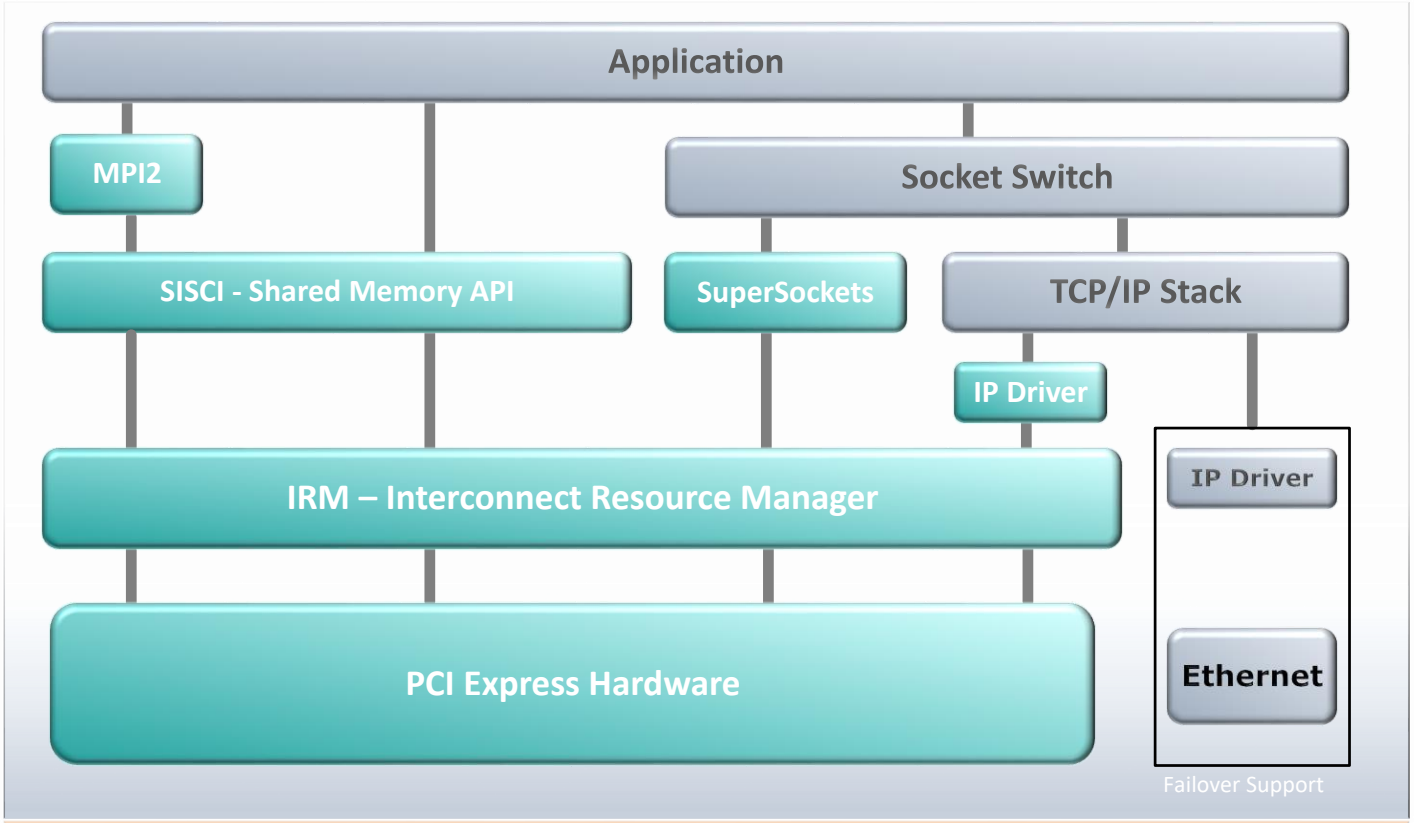
SISCI - Performance results



connect better

Copyright 2014 All rights reserved.

Dolphin Software Stack



SISCI Shared Memory API
SuperSockets Berkeley Sockets API
Optimized TCP / IP Driver

Network Management
IRM – Interconnect Resource Manager

Dolphin Software Components

Standard Software Components

connect better

Copyright 2014 All rights reserved.

SISCI API

connect better

Copyright 2014 All rights reserved.

SISCI API

- **SISCI** –
  Software **I**nfrastructure for **S**hared-Memory **C**luster **I**nterconnects

- Application Programming Interface (API)

- Shared Memory Programming Model

- User space access to basic **NTB** (Non-Transparent Bridge*) operations*

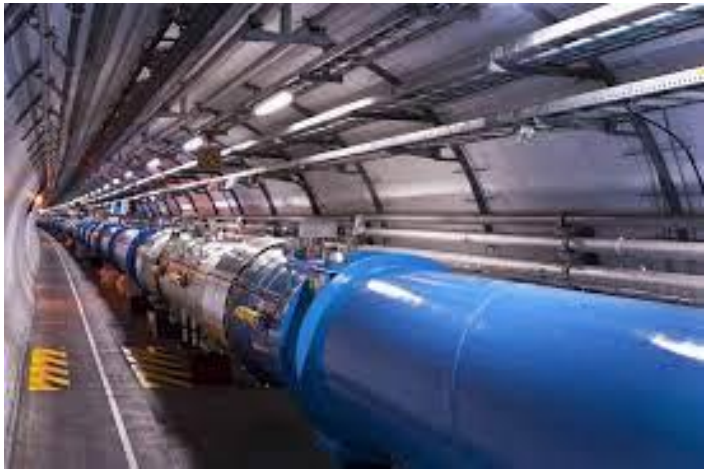
  - High bandwidth
  - Low latency
  - Memory Mapped Remote Access
  - DMA Transfers
  - Remote Interrupts
  - Callbacks

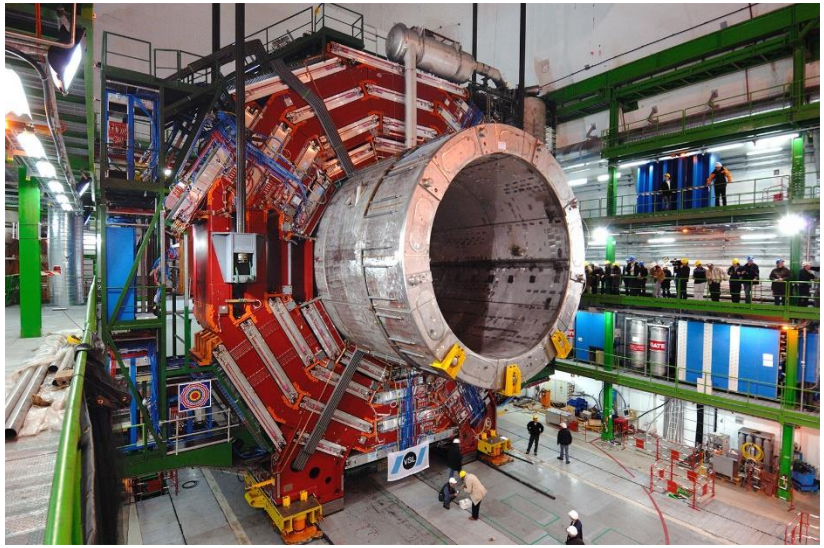
connect better

Copyright 2014 All rights reserved.

SISCI API

- Developed in a European research project in collaboration with CERN (1997 – 1999)
- Particle accelerator (27-kilometre ring) where the particles collide together at close to the speed of light





connect better

Copyright 2014 All rights reserved.

SISCI API widely used different areas

- Military
- Medical (MR and Ultrasound)
- Stock market, Financial market
- Autonomous driving (Self driving cars)
- PCIe extension (Transparent)
- Clustering with GPUs and NVMe
- NVMe over PCIe fabrics, Data mirroring
- Reflective memory systems
- Simulation

connect better

Copyright 2014 All rights reserved.

SISCI API

- SISCI API provides a powerful interface to migrate embedded applications to a Dolphin Express network.
- Simplified Cluster Programming
- Cross Platform / Cross Operating systems
  - Big-endian (BE) and little-endian (LE) machines can be mixed
- Cross Operating systems
  - Linux
  - Windows
  - RTX
  - VxWorks
  - QNX

connect better

Copyright 2014 All rights reserved.

SISCI API HANDLES

connect better

Copyright 2014 All rights reserved.

SISCI API – Handles – SISCI Types

- Remote shared memory, DMA transfers and remote interrupts, require the use of logical entities like devices, memory segments and DMA queues
- Each of these entities is characterized by a set of properties that should be managed as an unique object in order to avoid inconsistencies
- To hide the details of the internal representation and management of such properties to an API user, a number of handles / descriptors have been defined and made opaque

connect better

Copyright 2014 All rights reserved.

SISCI API – Handles - SISCI Types

- sci_desc_t
  - A SISCI virtual device, which is a communication channel to the SISCI driver. It is initialized by **SCIOpen()**.
- sci_local_segment_t
  - A local memory segment handle. It is initialized by **SCICreateSegment()**
- sci_remote_segment_t
  - It represents a segment residing on a remote node. It is initialized by **SCIConnectSegment()** and **SCIConnectSCISpace()**

connect better

Copyright 2014 All rights reserved.

SISCI API – Handles - SISCI Types

- sci_map_t
  - A memory segment mapped in the process' address space. It is initialized by **SCIMapRemoteSegment()** and the function **SCIMapLocalSegment().**
- sci_sequence_t
  - It represents a sequence of operations involving error handling with remote nodes. It is used to check if errors have occurred during data transfer. The handle is initialized by **SCICreateMapSequence()**

connect better

Copyright 2014 All rights reserved.

SISCI API – Handles - SISCI Types

- sci_dma_queue_t
  - A chain of specifications of data transfers to be performed using DMA. It is initialized by **SCICreateDMAQueue()**.
- sci_local_interrupt_t
  - An instance of interrupts that an application has made available to remote nodes. It is initialized when the interrupt is created by calling the function **SCICreateInterrupt()**.
- sci_remote_interrupt_t
  - An interrupt that can be trigged on a remote nodes. It is initialized when the interrupt is created by **SCIConnectInterrupt()**.

connect better

Copyright 2014 All rights reserved.

ERROR CODES

connect better

Copyright 2014 All rights reserved.

ERROR CODES

- SISCI API functions returns an error code as an output parameter to indicate if the execution succeeded or failed
  - **SCIInitialize(……., &error);**
- **SCI_ERR_OK** is returned when no errors occurred during the function call.
- The error codes are collected in an enumeration type called ***sci_error_t***
  - *sci_error_t error;*
- The error codes are specified in the sisci_error.h file

connect better

Copyright 2014 All rights reserved.

FLAG OPTIONS

connect better

Copyright 2014 All rights reserved.

FLAG OPTIONS

- Most SISCI API function have a flag option parameter
  - **SCI_FLAG_** …
  - The flag options are specified in sisci_api.h file
- The default option for the flag parameter is 0
  - **SCI_NO_FLAGS**
    - The flag is commonly used, but not defined in the SISCI API
    - #define SCI_NO_FLAGS 0
  - **SCIMemCpy**(sequence, src, remMap, remoteOffset, size, **SCI_FLAG_ERROR_CHECK**, &error);

connect better

Copyright 2014 All rights reserved.

FLAG OPTIONS

```
#define NO_FLAGS  0

SCIOpen(sci_desc_t *sd,
        unsigned int flags,
        sci_error_t *error);

SCIOpen(&sd, NO_FLAGS, &error);
```

connect better

Copyright 2014 All rights reserved.

SISCI API FUNCTIONS

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIInitialize()

- **SCIInitialize()**
  - Initialize the SISCI Library
  - Fetch the CPU type, hostbridge, adapter type. Select the optimized copy function for a system
  - Driver version checking
  - Allocates internal resources
  - Must be called **only once** in the application program and before any other SISCI API functions
  - If the SISCI library and the driver versions are not consistent, the function will return **SCI_ERR_INCONSISTENT_VERSIONS**

connect better

Copyright 2014 All rights reserved.

SISCI API - SCITerminate()

- **SCITerminate()**
  - Before an application is terminated, all allocated resources should be removed
  - De-allocates resources that was created by the **SCIInitialize()**
  - Should be the last call in the application
  - Should be called only once in the application

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIOpen()

- **SCIOpen()** creates a SISCI API handle (virtual device)
- Each segment must be associated with a handle
- Up to 32 segments can be created using the same handle
- If the **SCIInitialize()** is not called before **SCIOpen()**, the function will return **SCI_ERR_NOT_INITIALIZED**

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIClose()

- **SCIClose()**
  - Closes the virtual device
  - The virtual device becomes invalid and should not be used
  - If some resources is not deallocated, the SISCI driver will do the neccessary cleanup at program exit

connect better

Copyright 2014 All rights reserved.

SISCI API – Initialization example

```
sci_error_t error;
sci_desc_t   sd;
SCIInitialize(NO_FLAGS,&error);
if (error != SCI_ERR_OK) {
   /* Initialization error */
   return error;
}
SCIOpen(&sd,NO_FLAGS,&error);
if (error != SCI_ERR_OK) {
   /* Error */
   return error;
}
/* Use the SISCI API */

SCIClose(sd, NO_FLAGS, &error);
SCITerminate();
```

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIProbeNode()

- **SCIProbeNode(..., nodeId)**
  - The function check if the remote node is reachable on the cluster
  - The function is useful to check if all nodes on the cluster is initialized and reachable
  - Possible error codes
    - **SCI_ERR_NO_LINK_ACCESS**
    - **SCI_ERR_NO_REMOTE_LINK_ACCESS**

connect better

Copyright 2014 All rights reserved.

Dolphin

PIO MODEL

connect better

Copyright 2014 All rights reserved.

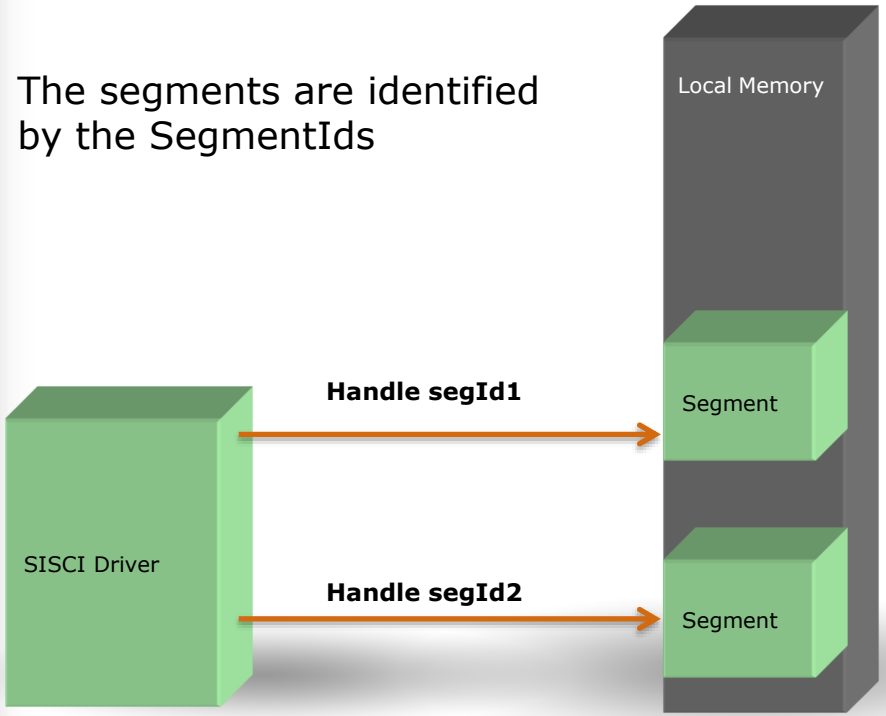
SISCI API - PIO Model

- What is PIO (**Programmed Input/Output**)?
  - **Programmed input/output** (PIO) is a method of transferring data between the CPU and a peripheral device, such as a network adapter or storage devices. For **programmed I/O**, the software that is running on the CPU uses instructions to perform data transfers to or from an **I/O** device.
  - In such a case, it is the **CPU** that actively reads from or writes to remote memory using **load/store** operations
  - This approach is known as **Programmed I/O (PIO)**
  - Once the mapping is created, the driver is not involved in the data transfer

connect better

Copyright 2014 All rights reserved.

SISCI – SCICreateSegment()



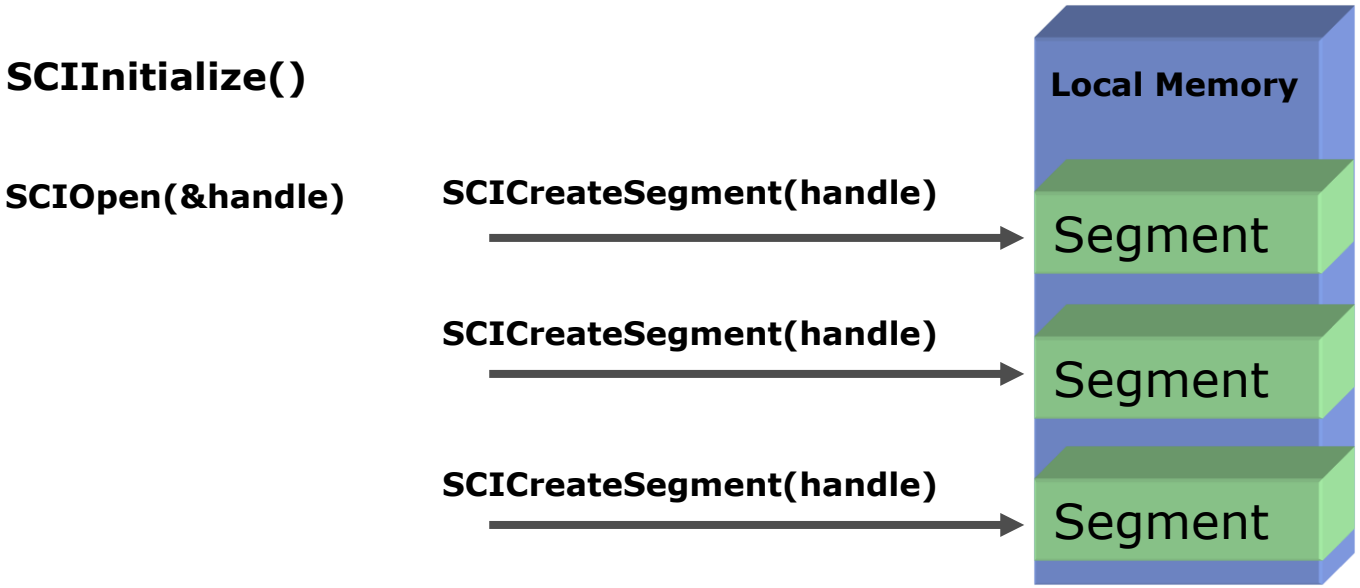
- Segment Allocation
  - Allocation of a segment on a local host
- Contiguous memory
  - Allocate contiguous memory
- Segment-Id
  - The segmentId for each segment must be unique on the local machine
- Identifying local segments
  - NodeId, segId
  - If segmentId already exist, the **SCICreateSegment()** will return **SCI_ERR_BUSY**

connect better

Copyright 2014 All rights reserved.

SISCI – SCICreateSegment()

- Creating multiple segments



connect better

Copyright 2014 All rights reserved.

SISCI API - SCIRemoveSegment()

- **SCIRemoveSegment()**
  - This function will de-allocate the resources used by a local segment

connect better

Copyright 2014 All rights reserved.

SISCI API - Creating Segment-Ids

- A segment-id for a segment must be unique on the local machine (32 bit)
- A segment is identified by segmentId and nodeId
- Local and remote nodeId can be used to create a segmentId
- Use group number as segmentId (part of the pre-code)
- One possible way to create a segment-Id:

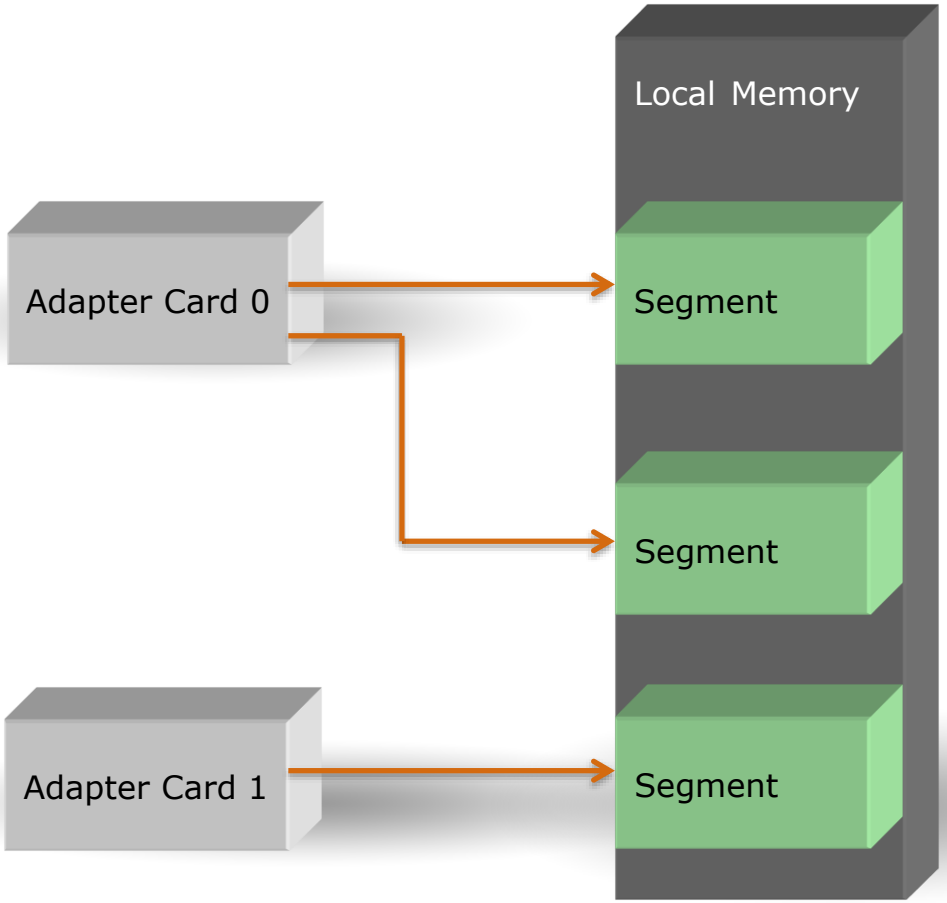
```
localSegmentId     = (localNodeId << 16) | remoteNodeId  << 8 | (group_number << 4) | key_offset;

remoteSegmentId = (remoteNodeId << 16) | localNodeId << 8  | (group_number << 4) | key_offset;
```

connect better

Copyright 2014 All rights reserved.

SISCI - Multi-card support



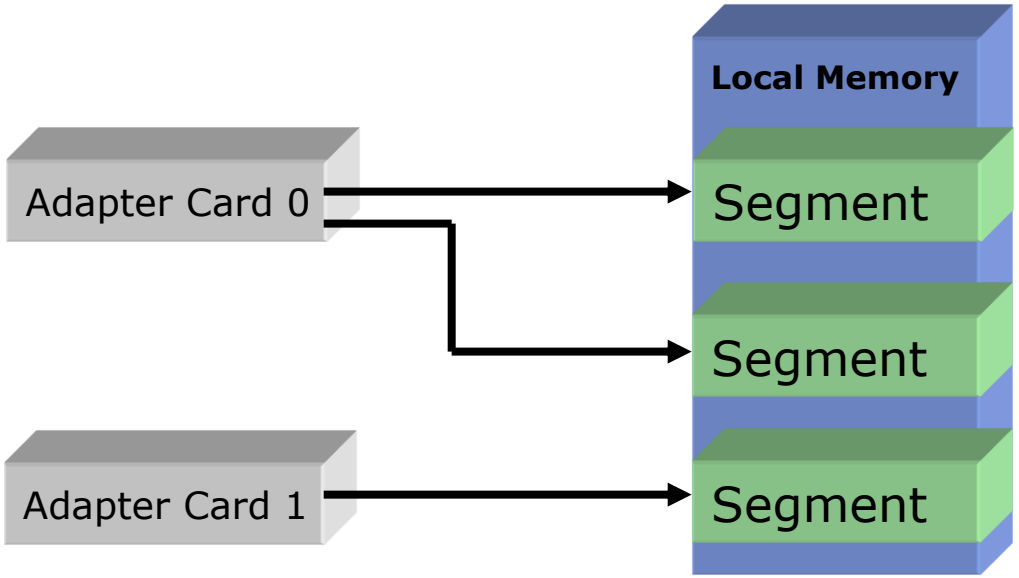
- Multi-card support
  - One machine can support several adapter cards
- Multiple memory segments
  - Multiple memory segments can connect to each card

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIPrepareSegment()

- One host can have several adapter cards.
- The function **SCI PrepareSegment()** prepares the segment to be accessible by the selected Dolphin adapter

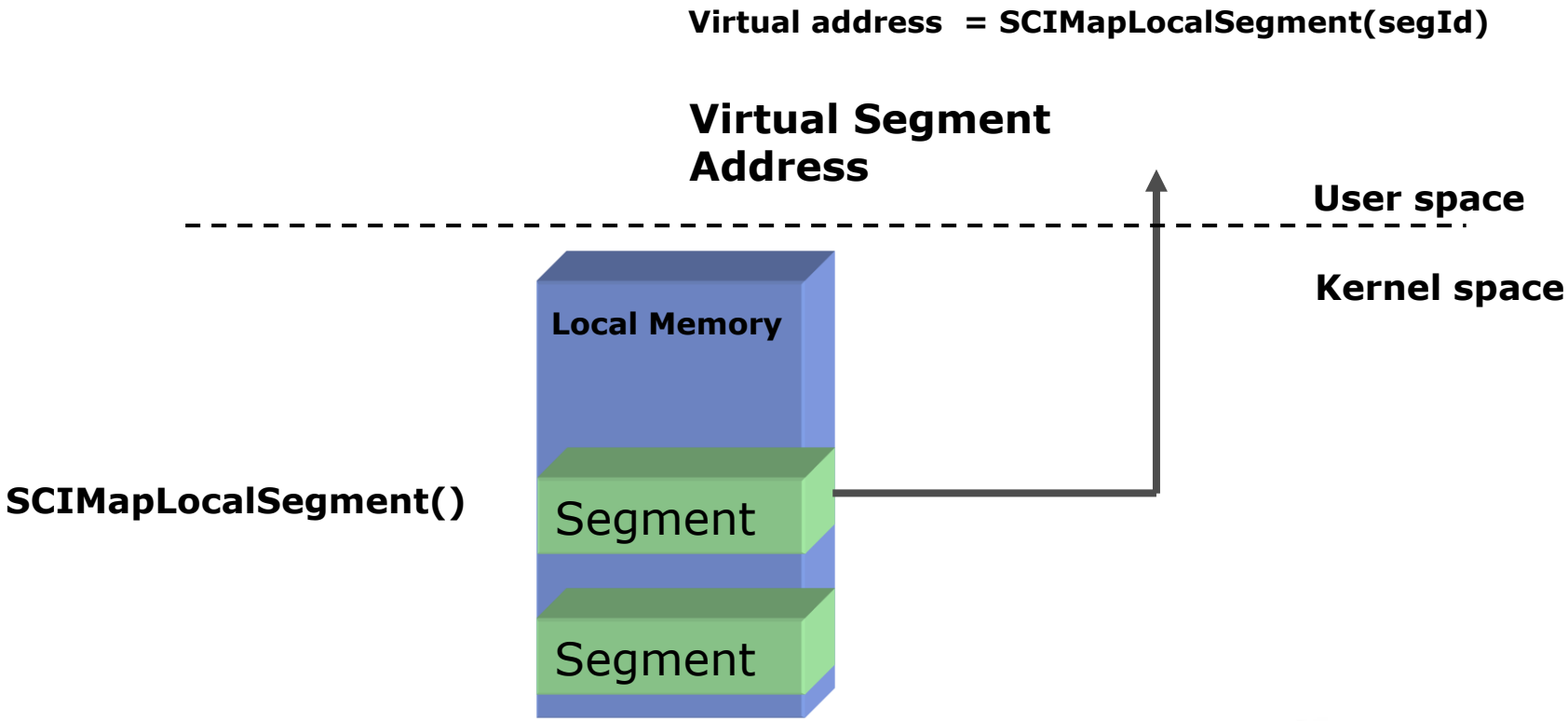


connect better

Copyright 2014 All rights reserved.

SISCI API - SCIMapLocalSegment()

- **SCIMapLocalSegment()** maps the local segment into the application's virtual address space

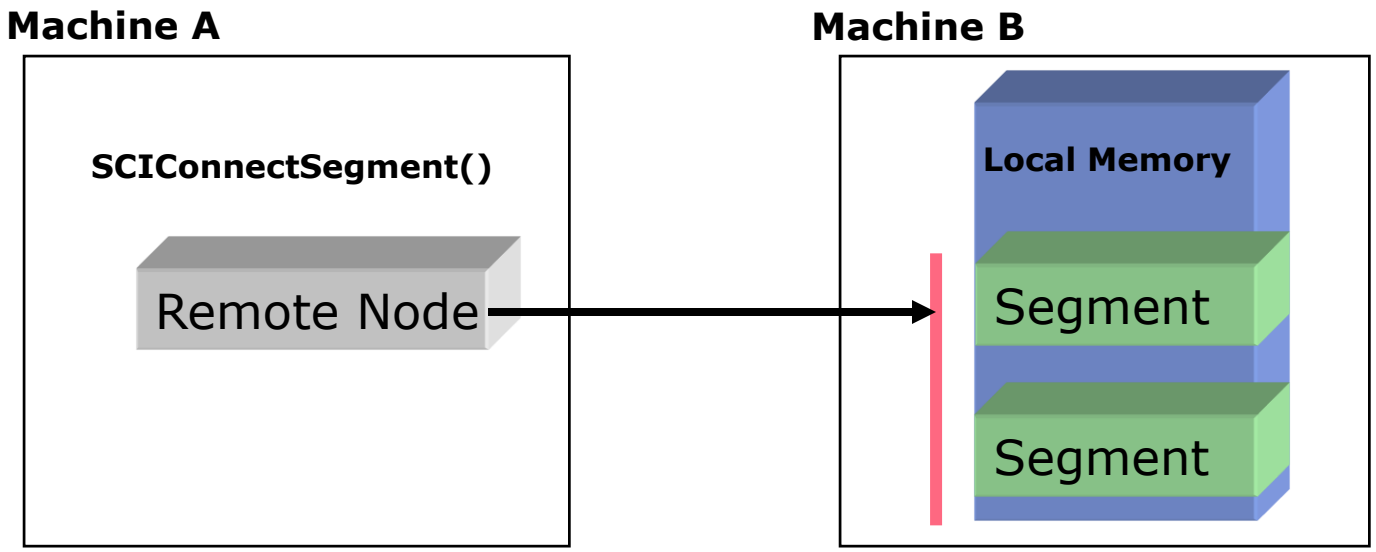


connect better

Copyright 2014 All rights reserved.

SISCI API - SCISetSegmentAvailable()

- The function **SCISetSegmentAvailable()** makes a local segment visible to the remote nodes
- The local segment is available to allow remote connections

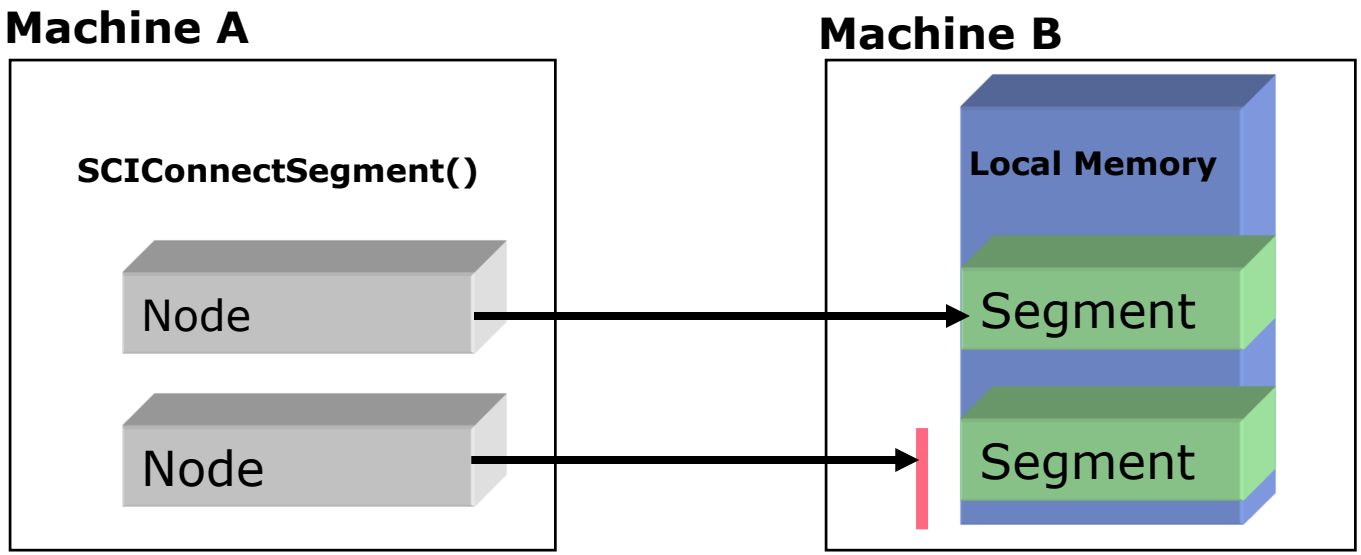


connect better

Copyright 2014 All rights reserved.

SISCI API - SCISetSegmentUnavailable()

- No new connections will be accepted on that segment
- The call to **SCISetSegmentUnavailable()** doesn't affect existing remote connections
- SCISetSegmentUnavailable()

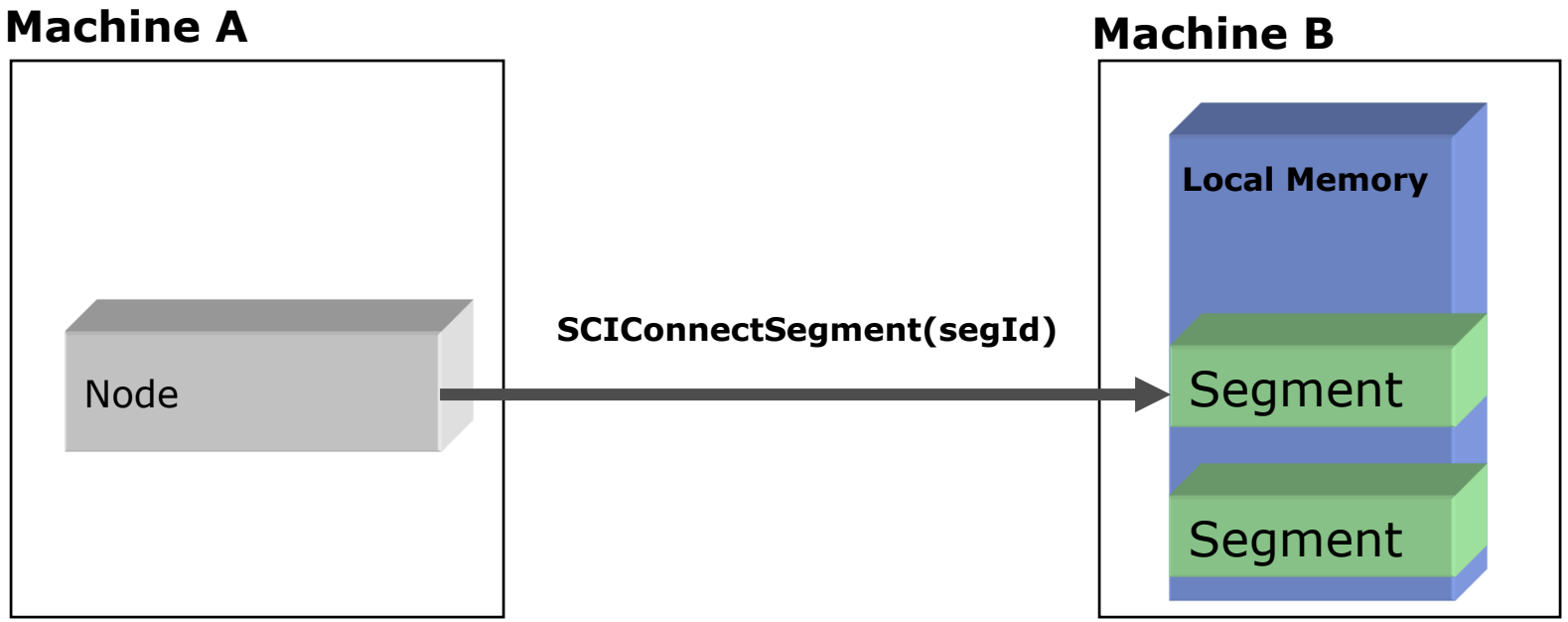


connect better

Copyright 2014 All rights reserved.

SISCI API - SCIConnectSegment()

- **SCIConnectSegment()** connects to a segment on a remote node
- Creates and initializes a handle for the connected segment



connect better

Copyright 2014 All rights reserved.

SISCI API - SCIConnectSegment()

- The function **SCIConnectSegment()** must be called in a loop
- The status of the remote segment is not known
  - The segment is not created
  - The remote node is still booting
  - The driver is not yet loaded

```
do {
        SCIConnectSegment(&error);
         /* Sleep before next connection attempt */
        sleep(1);
} while (error != SCI_ERR_OK) ;
```

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIDisconnectSegment()

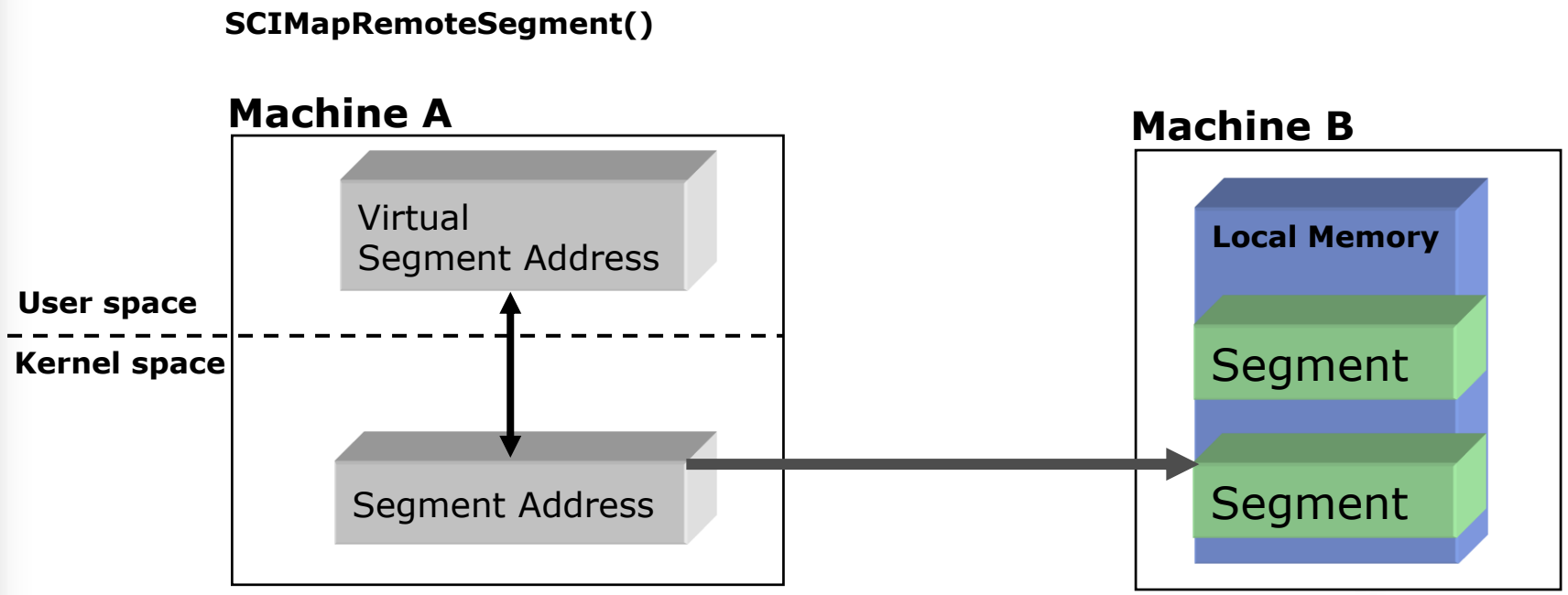
- **SCIDisconnectSegment()**
  - The function disconnects from a remote segment
  - If the segment was connected using **SCIConnectSegment(),** the execution of **SCIDisconnectSegment()** also generates an **SCI_CB_DISCONNECT** event directed to the application that created the segment.
  - If the Segment is still mapped, the function will return **SCI_ERR_BUSY**

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIMapRemoteSegment()

- **SCIMapRemoteSegment()** maps a *remote segment's memory* into user space and returns a pointer to the beginning of the mapped segment



connect better

Copyright 2014 All rights reserved.

SISCI API - SCIMapRemoteSegment()

- It is possible to map only a part of the segment by varying the the *size* and *offset* parameters, with the constraint that the sum of the size and offset does not go beyond the end of the segment

- Once a memory segment is available, i.e. you have a handle to either local or remote segment resources, you can access the segment in different ways:
  - Map the segment into the address space of your process and then access it as normal memory operations - e.g. via pointer operations or **SCIMemCpy()**
  - Use the Dolphin adapter DMA engine to move data (RDMA)

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIUnmapSegment()

- **SCIUnmapSegment()**
  - Unmaps the segment from the program's address space (user space) that was mapped either with **SCIMapLocalSegment()** or **SCIMapRemoteSegment()**
  - Destroys the corresponding handle
  - Error return value **SCI_ERR_BUSY**
    - the segment is in use

connect better

Copyright 2014 All rights reserved.

DATA TRANSFER

connect better

Copyright 2014 All rights reserved.

SISCI API - Data Transfer

- The virtual segment address can be used for data transfers
  - Use the address directly doing CPU load/store operations
  - SCIMemCpy()
  - DMA
- If the function succeeds, the return value is a pointer to the beginning of the mapped segment
- The address can be used directly to transfer data
  - *remoteAddress = data;
- Note that the address pointer is declared as ***volatile*** to prevent the compiler from doing wrong optimization of the code

```
volatile *unsigned int remoteMapAddr;
remoteMapAddr = SCIMapRemoteSegment();

for (i=0; i < numberOfStores; i++) {
        remoteMapAddr[i] = i;
}
```

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIMemCpy()

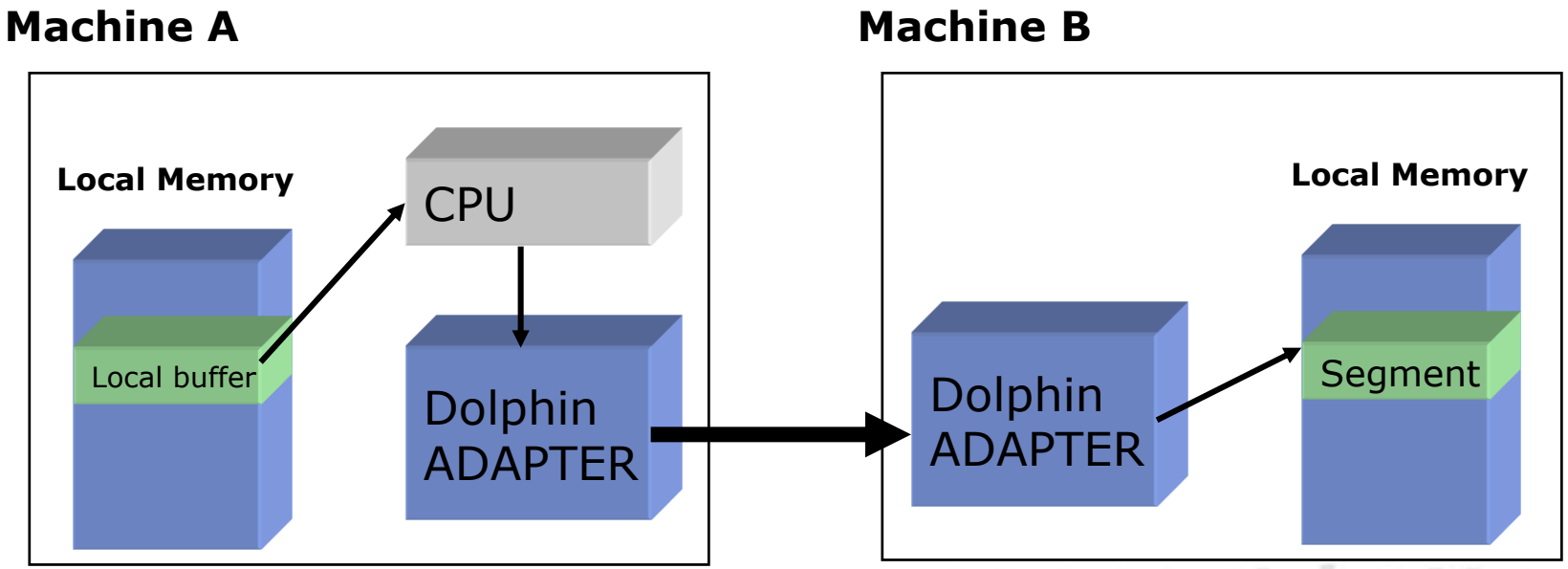
- **SCIMemCpy()**
  - PIO data transfer
- The function is optimized for the CPU and various hostbridges
- Transfers specified size of data
- Flag option to enable error checking.
- MMX and SIMD registers to optimize the data transfers

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIMemCpy()

- Optimized data transfer
- The low-level copy function is selected in **SCIInitialize()**

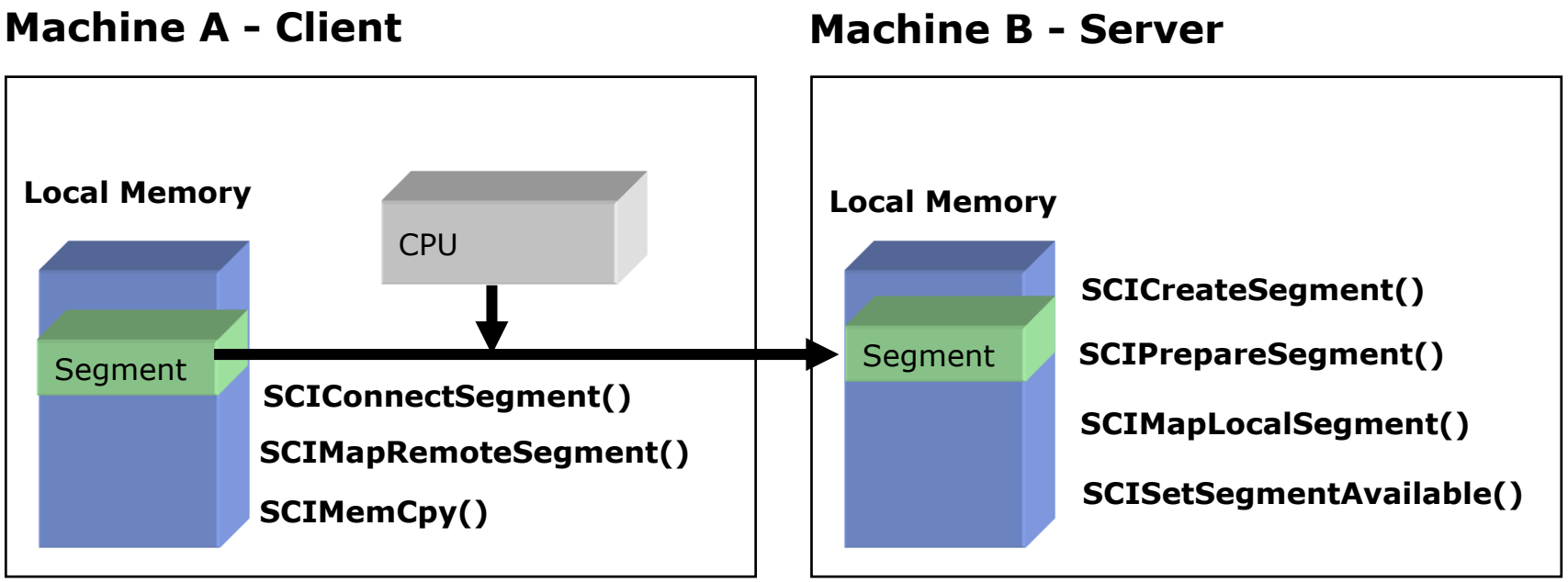


connect better

Copyright 2014 All rights reserved.

SISCI API - PIO Model summary

- PIO model on client and server node



connect better

Copyright 2014 All rights reserved.

PIO – Example - Server

```
/* Create a segmentId */
remoteSegmentId = (remoteNodeId << 16) | localNodeId | keyOffset;

/* Create local segment */
    SCICreateSegment(sd, &localSegment, localSegmentId, segmentSize, NO_CALLBACK, NULL,
      NO_FLAGS, &error);
/* Prepare the segment */
    SCIPrepareSegment(localSegment, localAdapterNo, NO_FLAGS, &error);

/* Set the segment available */
    SCISetSegmentAvailable(localSegment, localAdapterNo, NO_FLAGS, &error);

/* Map local segment to user space */
   localMapAddr =  SCIMapLocalSegment(localSegment, &localMap, offset,segmentSize, NULL, NO_FLAGS, &error);

SCIWaitForInterrupt(localInterrupt, SCI_INFINITE_TIMEOUT, NO_FLAGS, &error);

buffer = (unsigned int *)localMapAddr;
/* Get the data */
   for (i=0; i< 20; i++) {
       printf("%d ",buffer[i]);
   };
```

connect better

Copyright 2014 All rights reserved.

PIO – Example - Client

```
/* Create a segmentId */
remoteSegmentId = (remoteNodeId << 16) | localNodeId | keyOffset;
do {
        SCIConnectSegment(sd, &remSegment, remoteNodeId, remoteSegmentId, localAdapterNo,
                                NO_CALLBACK,NULL, SCI_INFINITE_TIMEOUT, NO_FLAGS, &error);
 } while (error != SCI_ERR_OK);
/*
dst = SCIMapRemoteSegment(remSegment, &remMap, offset, segmentSize, NULL, NO_FLAGS, &error);

/* Copy the data */
SCIMemCpy(sequence, src, remMap, remoteOffset, size, SCI_FLAG_ERROR_CHECK, &error);

/* Send an interrupt to notify the remote node that the data has been sent */
SCITriggerInterrupt(remoteInterrupt, NO_FLAGS, &error);

/* Alternative memcopy approach, but not very efficient */
 for (j=0;j<nostores;j++) {
        dst[j] = src[j];
}
```

connect better

Copyright 2014 All rights reserved.

REMOTE INTERRUPTS

connect better

Copyright 2014 All rights reserved.

SISCI API – SCICreateInterrupt()

- **SCICreateInterrupt()**
  - Creates an interrupt resource and makes it available for remote nodes
  - Initialize a handle for the interrupt
  - An interrupt is associated by the driver with a unique number (interruptId)
  - If the flag **SCI_FLAG_FIXED_INTNO** is specified, the function use the number passed by the caller

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIConnectInterrupt()

- **SCIConnectInterrupt()**
  - Connects the caller to an interrupt resource available on a remote node
  - The function creates and initializes a descriptor for the connected interrupt
  - Since the status of the remote interrupt is not known (i.e, not created) the **SCIConnectInterrupt()** must be called in a loop

```
do {
        SCIConnectInterrupt(...., &error);
        sleep(1);
while (error != SCI_ERR_OK) ;
```

connect better

Copyright 2014 All rights reserved.

SISCI API – SCITriggerInterrupt()

- **SCITriggerInterrupt()**
  - The function triggers an interrupt on a remote node
  - The remote node gets notified

connect better

Copyright 2014 All rights reserved.

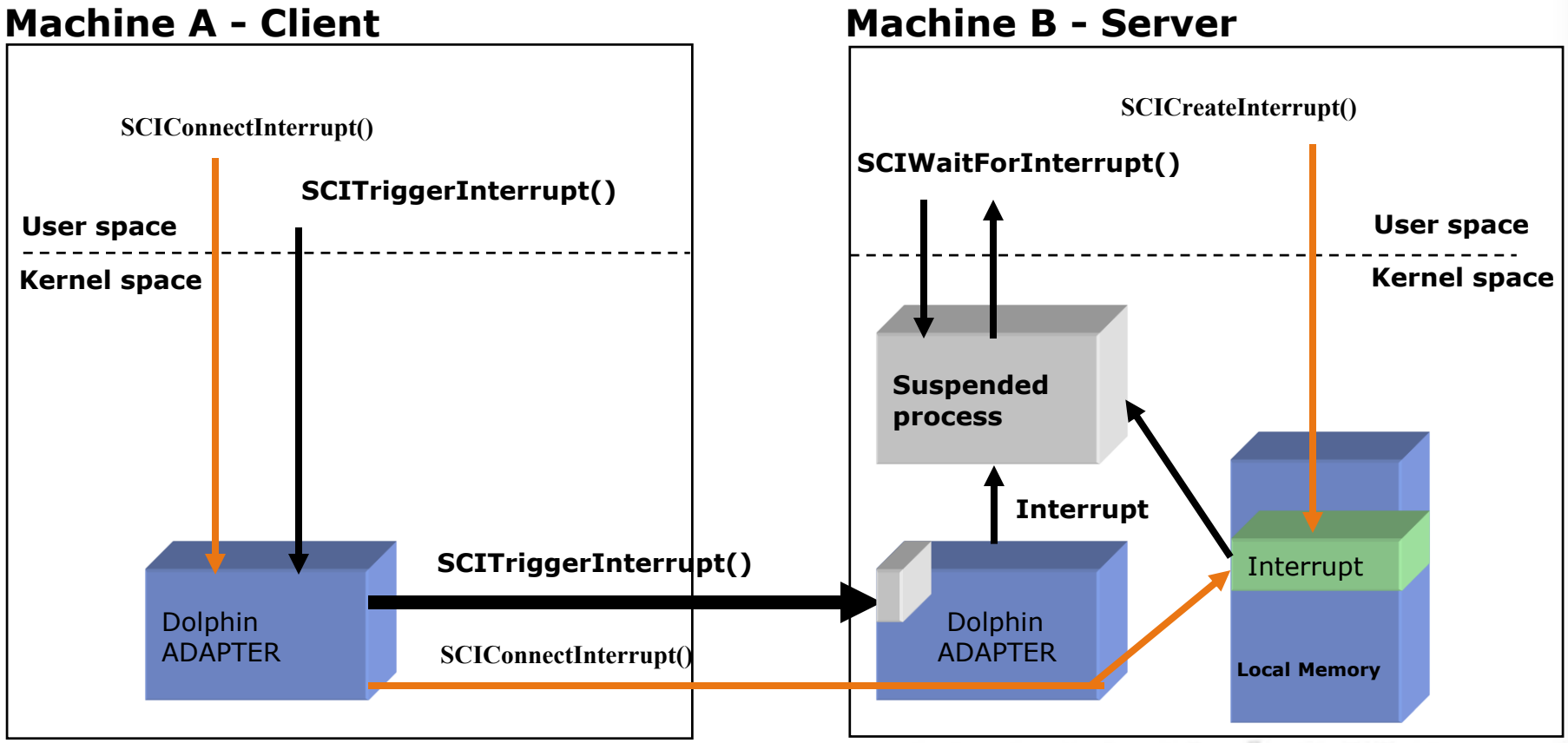
SISCI API – SCIWaitForInterrupt()

- **SCIWaitForInterrupt()**
  - This function blocks a program until an interrupt is received
  - If the flag option **SCI_INIFINITE_TIMEOUT** is specified, the function waits until the interrupt has completed
  - If a timeout value is specified, the function gives up when the timeout expires (not implemented)

connect better

Copyright 2014 All rights reserved.

SISCI API – INTERRUPT MODEL



connect better

Copyright 2014 All rights reserved.

SISCI API – Interrupt example - Server

```
interruptNo = USER_INTERRUPT;
 /* Create an interrupt */
 SCICreateInterrupt(sd, &localInterrupt, localAdapterNo, &interruptNo,
                            NO_CALLBACK, NULL, SCI_FLAG_FIXED_INTNO, &error);

 /* Wait for an interrupt */
 SCIWaitForInterrupt(localInterrupt, SCI_INFINITE_TIMEOUT,NO_FLAGS,&error);
 if (error != SCI_ERR_OK) {
    printf("\n");
    fprintf(stderr,"SCIWaitForInterrupt failed - Error code 0x%x\n",error);
    return error;
 }

 printf("\nNode %u received interrupt (%u)\n",
       localNodeId, interruptNo);
```

connect better

Copyright 2014 All rights reserved.

SISCI API – Interrupt example - Client

```
interruptNo = USER_INTERRUPT;
/* Now connect to the other sides interrupt flag */
do {
 SCIConnectInterrupt(sd, &remoteInterrupt, remoteNodeId, localAdapterNo,
                                 interruptNo, SCI_INFINITE_TIMEOUT,NO_FLAGS, &error);
} while (error != SCI_ERR_OK);

SCITriggerInterrupt(remoteInterrupt, NO_FLAGS, &error);
if (error != SCI_ERR_OK) {
    fprintf(stderr,"SCITriggerInterrupt failed - Error code 0x%x\n",error);
    return error;
} else {
    printf("Node %u triggered interrupt\n",localNodeId);
}
```

connect better

Copyright 2014 All rights reserved.

REMOTE INTERRUPTS WITH DATA

connect better

Copyright 2014 All rights reserved.

SISCI API – Remote Data Interrupts

- **SCITriggerDataInterrupt()**
  - Triggers a remote interrupt
  - Similar to the SCI Interrupt functions, but with data
  - **Maximum data transfer is 100 bytes**.

```
SCITriggerDataInterrupt(
        sci_remote_data_interrupt_t interrupt,
        void                        *data,
        unsigned int                 length,
        unsigned int                 flags,
        sci_error_t                 *error);
```

connect better

Copyright 2014 All rights reserved.

SISCI API – Remote Data Interrupts

- SCICreateDataInterrupt()
- SCIRemoveDataInterrupt()
- SCIConnectDataInterrupt()
- SCIDisconnectDataInterrupt()
- SCITriggerDataInterrupt()
- SCIWaitForDataInterrupt()

connect better

Copyright 2014 All rights reserved.

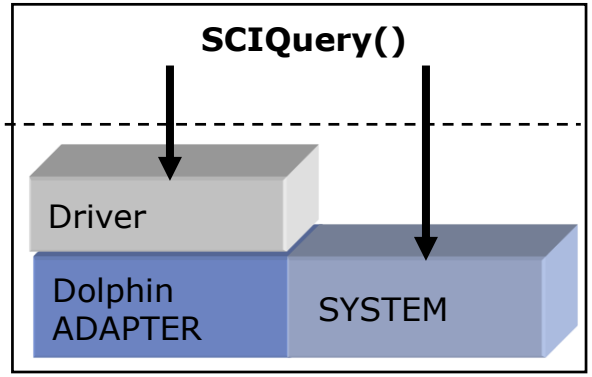
SCI QUERY

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIQuery()

- **SCIQuery()**
  - Provides information about the underlying Dolphin Express system
  - The main group defines its own data structure to be used as input and output to **SCIQuery()**
  - The queries consist of the main COMMAND and a subcommand
    - SCI_Q_ADAPTER
      - SCI_Q_ADAPTER_SERIAL_NUMBER
      - SCI_Q_ADAPTER_NODEID
      - SCI_Q_ADAPTER_LINK_WIDTH
      - SCI_Q_ADAPTER_LINK_SPEED
      - SCI_Q_ADAPTER_LINK_OPERATIONAL



  - Definitions of the queries are defined in sisci_api.h file

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIQuery() Example

```
sci_error_t GetLocalNodeId(unsigned int localAdapterNo,
                           unsigned int *localNodeId)
{
   sci_query_adapter_t queryAdapter;
   sci_error_t  error;
   unsigned int nodeId;

   queryAdapter.subcommand = SCI_Q_ADAPTER_NODEID;
   queryAdapter.localAdapterNo = localAdapterNo;
   queryAdapter.data = &nodeId;

   SCIQuery(SCI_Q_ADAPTER, &queryAdapter, NO_FLAGS, &error);

   *localNodeId = nodeId;
   return error;
}
```

connect better

Copyright 2014 All rights reserved.

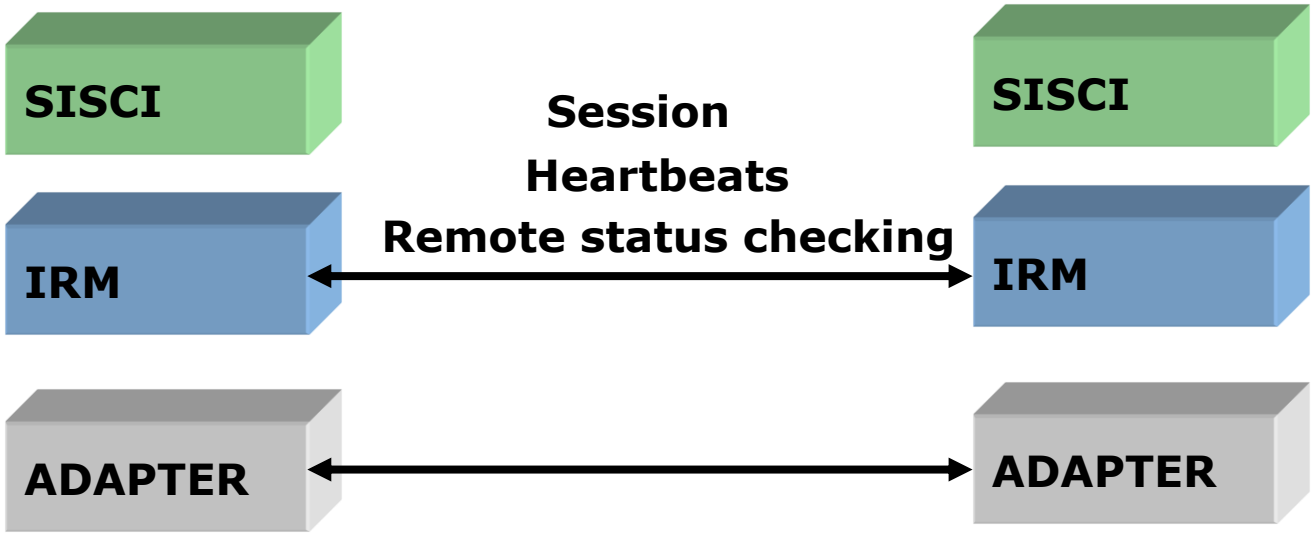
ERROR HANDLING

connect better

Copyright 2014 All rights reserved.

SISCI API – Session

- A session is established between the nodes that communicates
  - A heartbeat mechanism ensures that the remote node is alive
- Periodically checking the status of the remote node
- The error handling mechanism checks the session status, the interrupt status register and the cable status.



connect better

Copyright 2014 All rights reserved.

SISCI API – Error Handling

- **SCIStartSequence()** and **SCICheckSequence()**
- The hardware protocols guarantees that the data are delivered successfully when error handling mechanism is used and the sequence check returns SCI_ERR_OK
- Guaranteed correct data delivery from the function call **SCIStartSequence()** and **SCICheckSequence()**
- The **SCICheckSequence()** flushes the write buffers from the CPU and wait for the outstanding requests
- The error handling rate is system and application dependent
- Some overhead is added to the data transfer

connect better

Copyright 2014 All rights reserved.

SISCI API – SCICreateMapSequence

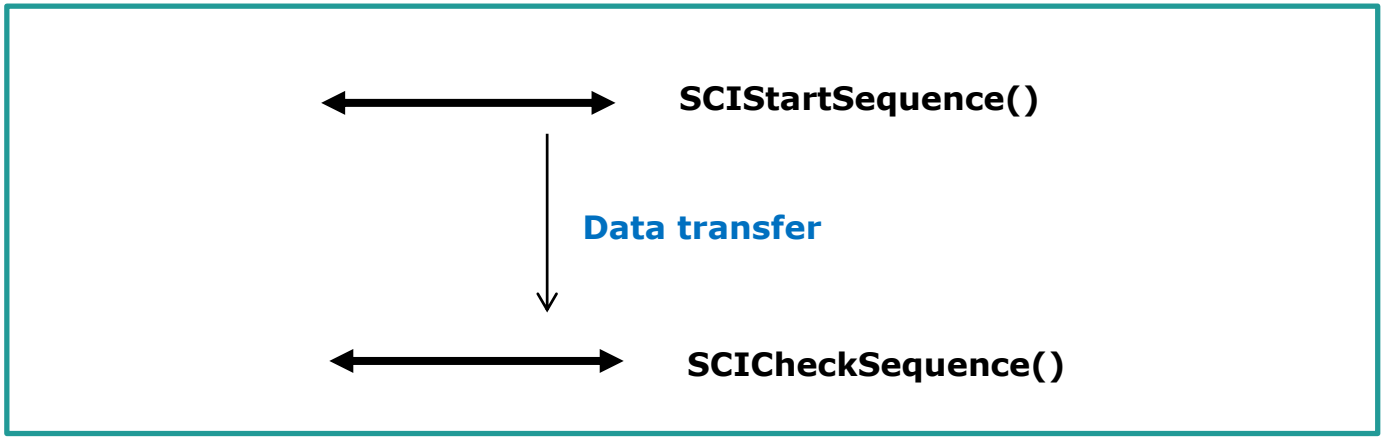
- **SCICreateMapSequence(remoteMap,&sequence, ….)**
  - This function creates and initializes a new sequence descriptor to check for transmission errors
  - Creates a sequence associated with a remote_map_t

connect better

Copyright 2014 All rights reserved.

SISCI API – Error Handling

- Guaratees the delivery of data between **SCIStartSequence()** and **SCICheckSequence()**
- The size of the data transfer size depends on the application



/ Start the data error checking */*
*sequenceStatus = **SCIStartSequence**(sequence,....);*

<DATA TRANSFER>

*sequenceStatus = **SCICheckSequence**(sequence,....);*

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIStartSequence

- **SCIStartSequence()**
  - The function performs the preliminary check of the error flags on the network adapter before starting a sequence of read and write operations on the mapped segment
  - Subsequent checks are done calling **SCICheckSequence()**
  - If the return value is **SCI_SEQ_PENDING** there is a pending error and the program is required to call SCIStartSequence until it succeeds before doing other transfer operations on the segment

connect better

Copyright 2014 All rights reserved.

SISCI API – SCICheckSequence

- **SCICheckSequence()**
  - The function checks if any error has occurred in the data transfer controlled by a sequence since the last check
  - The function can be invoked several times in a row without calling **SCIStartSequence()**
  - By default **SCICheckSequence()** also flushes the CPU write buffers and wait for all outstanding transactions to complete.
    - SCI_FLAG_NO_FLUSH
    - SCI_FLAG_NO_STORE_BARRIER
  - **SCICheckSequence**(SCI_FLAG_NO_FLUSH | SCI_FLAG_NO_STORE_BARRIER)

connect better

Copyright 2014 All rights reserved.

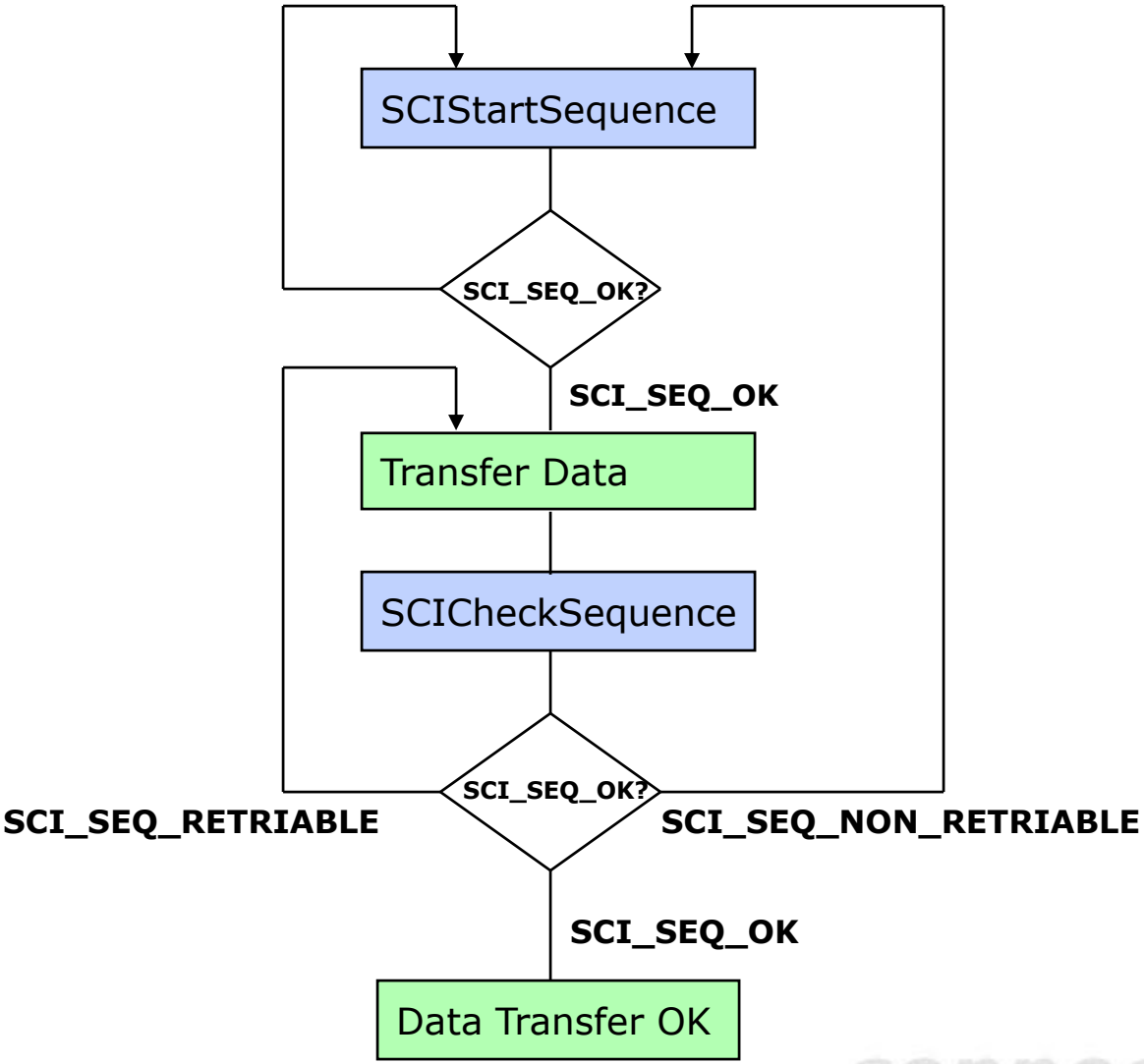
SISCI API – SCIStartSequence() / CheckSequence()

- The status from the sequence functions can return four possible values:
- **SCI_SEQ_OK**
  - The transfer was successful
- **SCI_SEQ_RETRIABLE**
  - The transfer failed due to non-fatal error but can be immediately retried (e.g. The system is busy because of heavy traffic)
- **SCI_SEQ_NON_RETRIABLE**
  - The transfer failed due to a fatal error (e.g. cable unplugged) and can be retried only after a successful call to SCIStartSequence()
- **SCI_SEQ_PENDING**
  - The transfer failed, but the driver hasn't been able to determine the severity of the error (fatal or non-fatal). SCIStartSequence() must be called until it succeeds

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIStartSequence() / CheckSequence()



connect better

Copyright 2014 All rights reserved.

SISCI API – SCIStartSequence() / CheckSequence()

```
{/* Start the data error checking */
do {
        do
              sequenceStatus = SCIStartSequence(sequence,....);
         } while (sequenceStatus != SCI_SEQ_OK);

        /* The connection is OK */
        <Do the data transfer>

        sequenceStatus = SCICheckSequence(sequence,....);
        if (sequenceStatus == SCI_SEQ_NON_RETRIABLE) {
                    <error handling>
                    break;
        }
} while (sequenceStatus != SCI_SEQ_OK);
/* Successful data transfer */
```

connect better

Copyright 2014 All rights reserved.

SCI FLUSH

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIFlush() in PIO mode

- **SCIFlush()**
  - **SCIFlush()** flushes the data from internal CPU buffers
  - Flushes the data from the CPU buffer, cache, IO-system and from the adaper card
    - CPU flush
  - If flag option **SCI_FLAG_FLUSH_CPU_BUFFERS_ONLY** is specified, only the the CPU buffer/cache is flushed

connect better

Copyright 2014 All rights reserved.

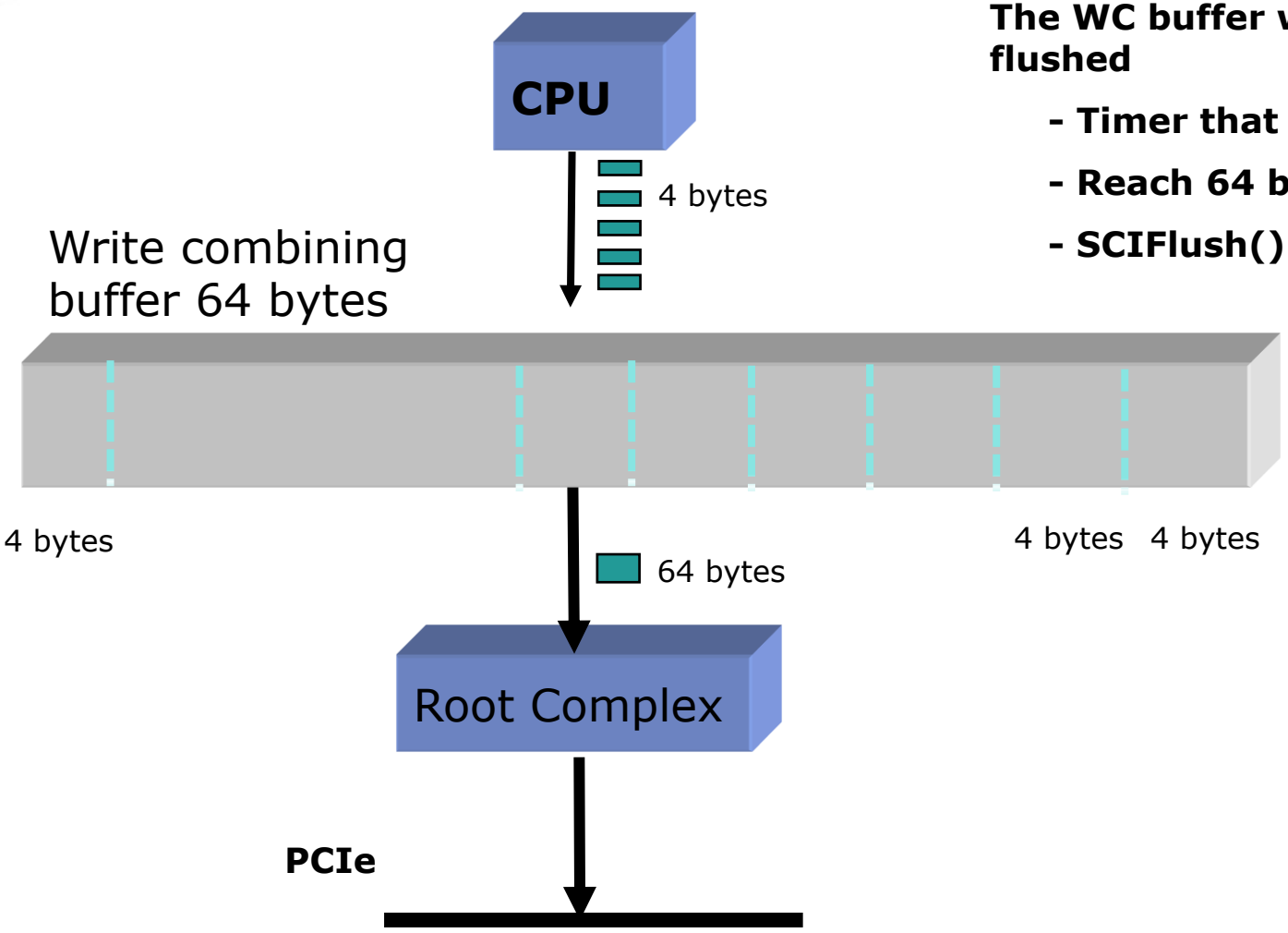
SISCI API – Write combining

- **Write combining** (**WC**) is a computer technique for allowing data to be combined and temporarily stored in a buffer — the **write combine buffer** (WCB) — to be released together later in burst mode instead of writing (immediately) as single bits or small chunks.
- **SCIFlush()** function flushes the write combining buffers
- Write combining mechanism is not supported on ARM systems

connect better

Copyright 2014 All rights reserved.

SCIFlush() - Write Combining



The WC buffer will be flushed

- **Timer that expired**
- **Reach 64 bytes**
- **SCIFlush() function**

connect better

Copyright 2014 All rights reserved.

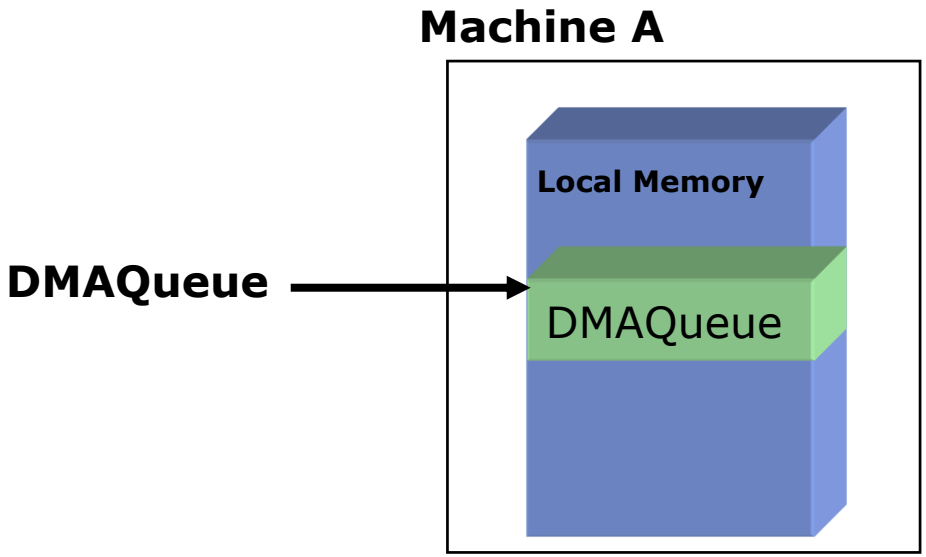
DMA MODEL

connect better

Copyright 2014 All rights reserved.

SISCI API - SCICreateDMAQueue()

- **SCICreateDMAQueue()**
  - Allocates resources for a queue of DMA transfers
  - Creates, initializes and returns a handle for the new DMA queue



connect better

Copyright 2014 All rights reserved.

SISCI API - SCIStartDmaTransfer()

- **SCIStartDmaTransfer()**
-
  - **SCIStartDmaTransfer()** starts the execution of the DMA queue and creates a control block in the memory for the DMA transfer
    - The control block contains the source and destination addresses and transfer size
  - Either the source or the destination of the transfer must be a local segment
  - By default the transfer operation is PUSH, i.e. from the local segment to the remote segment
    - In the opposite direction, the flag **SCI_FLAG_DMA_READ** has to be specified

connect better

Copyright 2014 All rights reserved.

- **SCIStartDmaTransferVec()**
  - Vectorized DMA
    - Vectors of control blocks
    - Vectors of DMA descriptors is sent as a parameter to the function
  - The function starts the execution of the DMA queue (vectors)
  - Creates a control block in the memory for each DMA transfer

connect better

Copyright 2014 All rights reserved.

- **SCIStartDmaTransferVec()**
  - The control blocks contains the source and destination addresses and the transfer size
  - Either the source or the destination of the transfer must be a local segment
  - By default the transfer operation is PUSH, i.e. from the local segment to the remote segment
    - In the opposite direction, the flag **SCI_FLAG_DMA_READ** has to be specified.

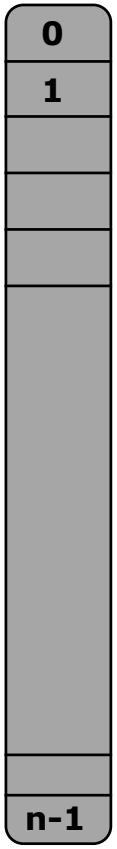
connect better

Copyright 2014 All rights reserved.

SISCI API - SCIStartDmaTransferVec()

- **SCIStartDmaTransferVec(...,dma_vec, vec_len,..)**

Dma_vec



Vec[0] = *src, *dst, size

Vec[1] = *src, *dst, size

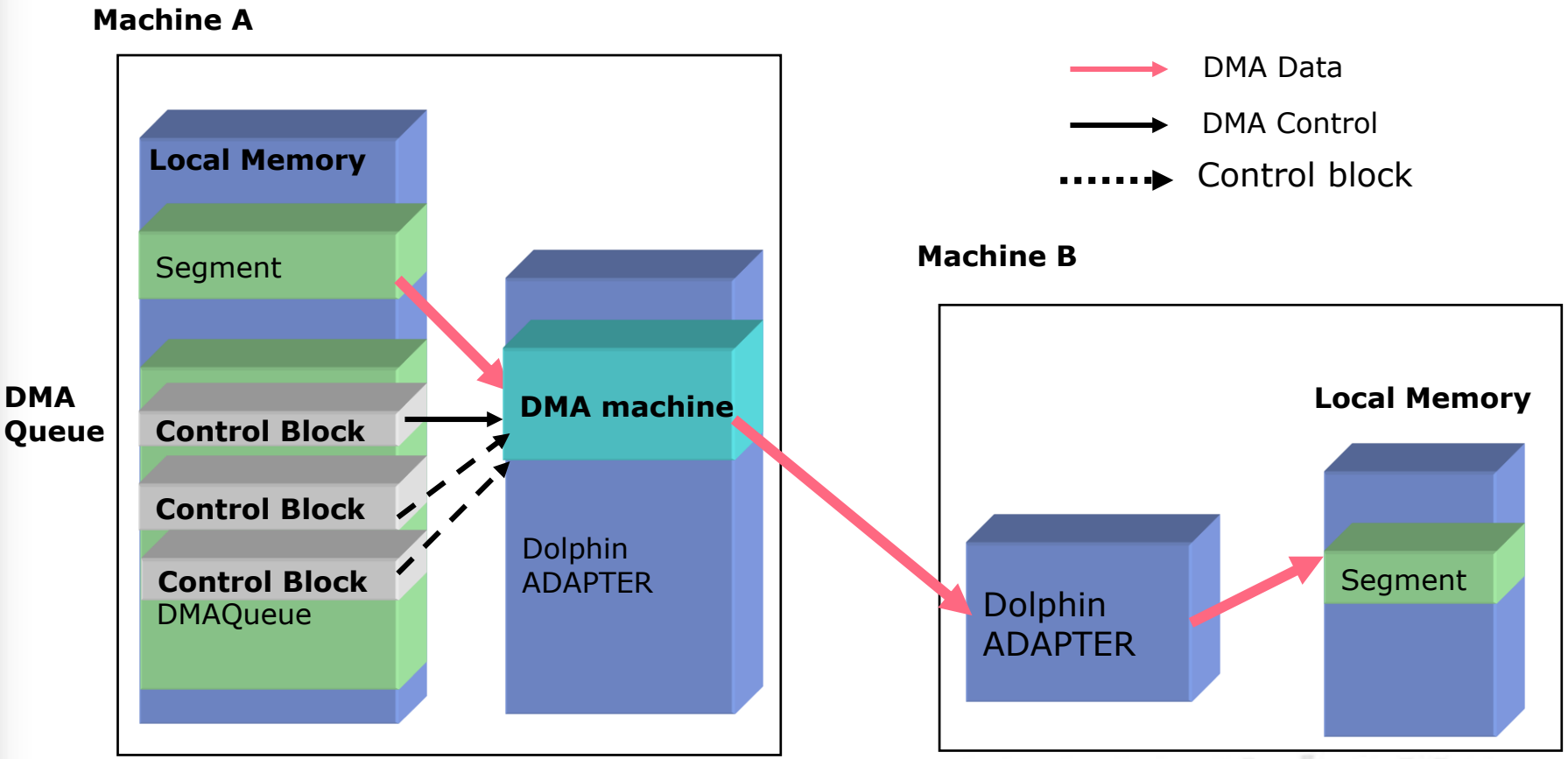
Vec[n-1] = *src, *dst, size

- **Max vector length = 256**

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIStartDmaTransferVec()

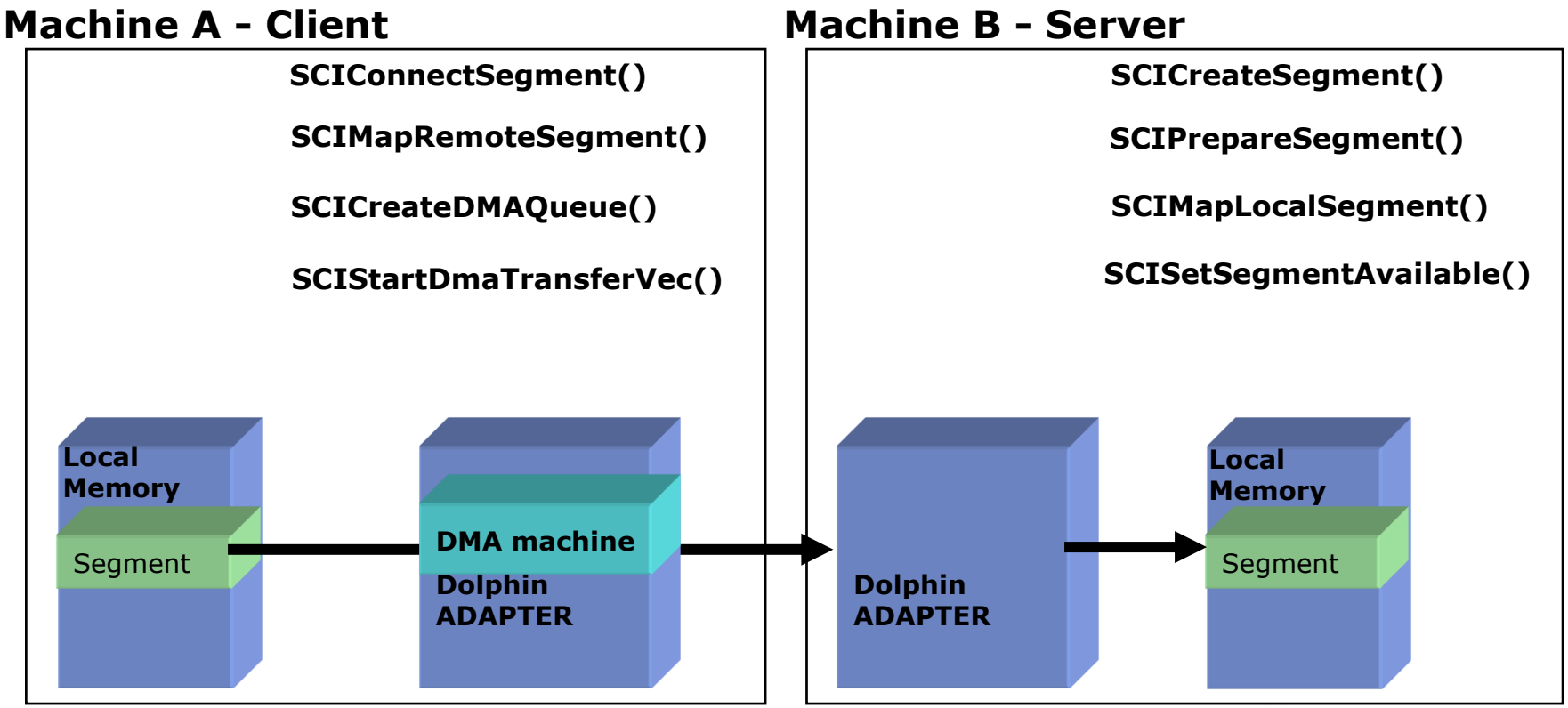


connect better

Copyright 2014 All rights reserved.

SISCI API - DMA Model

- DMA call sequence on client and server node



connect better

Copyright 2014 All rights reserved.

SISCI API - SCIRemoveDMAQueue()

- **SCIRemoveDMAQueue()**
  - De-allocates resources of the DMA queue which was allocated by **SCICreateDMAQueue()**
  - This function can be called only if the DMA state is either in:
    - IDLE (initial state)
    - DONE, ERROR or ABORTED (final state)

connect better

Copyright 2014 All rights reserved.

SISCI API – Comparison PIO vs DMA

	PIO	DMA	Comments
Min latency	0,54 us	-	4 byte transfers
Latency	0,54 us	6,74 us	64 byte transfers
Max performance x86	6300 MB/s	11000 MB/s	DMA Size > 64 k
Max performance Xavier	1500 MB/s	6000 MB/s	DMA Size > 64 k No write combining
CPU usage	High	Low	

connect better

Copyright 2014 All rights reserved.

SISCI API – How to write optimized applications?

- Use only PIO **write** operations
  - Store
  - DMA push / pull
- Use **SCIMemCpy()** for optimized PIO transfers
- Aligned **src** and **dst** buffers
- Use error checking only when required
  - **SCIStartSequence()/SCICheckSequence()**
- Minimize the use of remote interrupts
  - Context switching on the destination

connect better

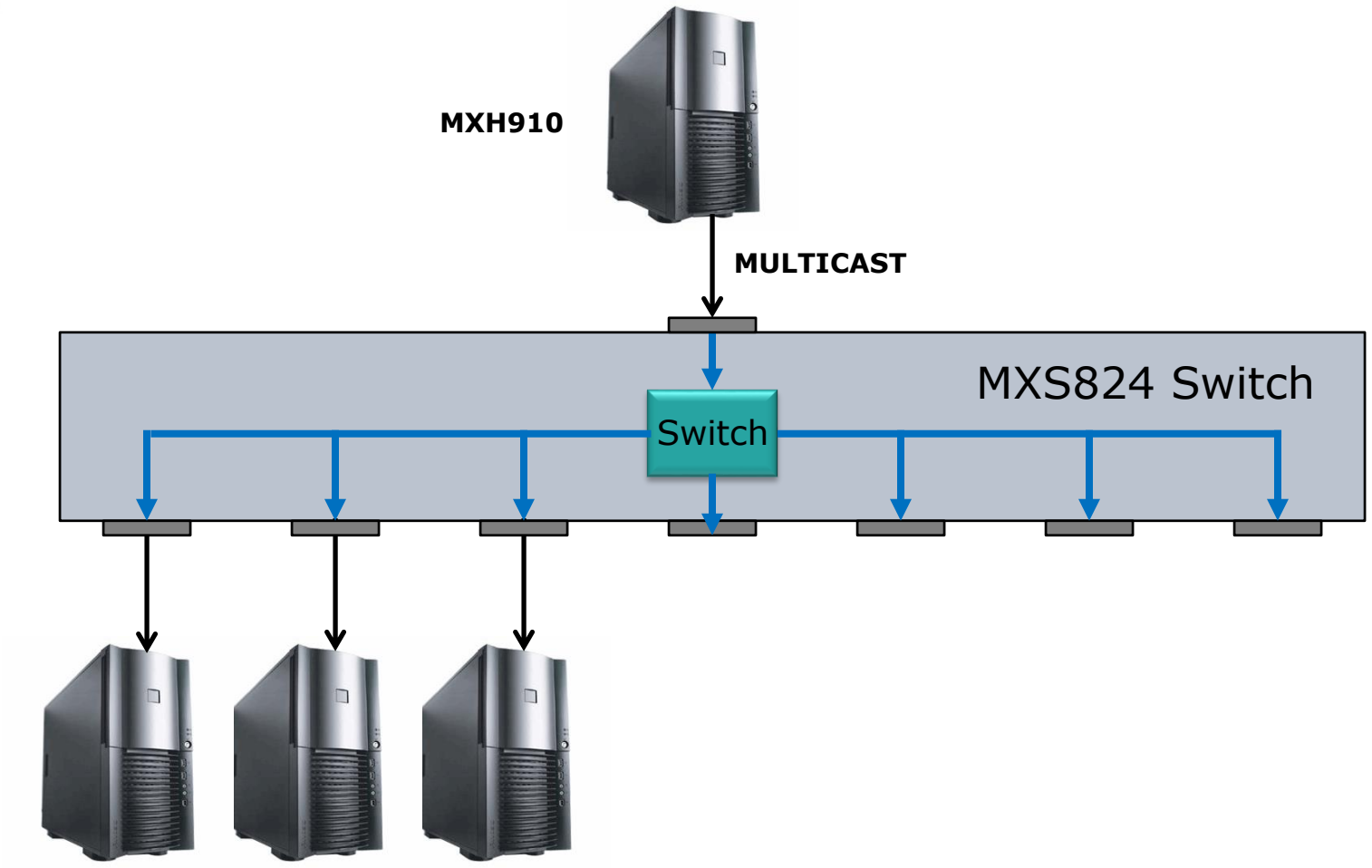
Copyright 2014 All rights reserved.

MULTICAST

connect better

Copyright 2014 All rights reserved.

Dolphin Cluster - Multicast



connect better

Copyright 2014 All rights reserved.

SISCI API - Multicast

- **Multicast**
  - Also called
    - **Reflective memory**
    - **Broadcast**
- **The data are multicasted by the chip in the MXS824 switch**
- Reflective programs client node sends data to the receivers
  - Sender: ./reflective_bench –client –nodes 2
  - Receiver 1: ./reflective_bench –server –rank 1
  - Receiver 2: ./reflective_bench –server –rank 2
- Use the flag SISCI API flag: **SCI_FLAG_BROADCAST**
- Support for Multicast:
  - **SCICreateSegment(…, SCI_FLAG_BROADCAST);**
  - **SCIConnectSegment (…, SCI_FLAG_BROADCAST);**
  - **remoteNodeId = DIS_BROADCAST_NODEID_GROUP_ALL;**

connect better

Copyright 2014 All rights reserved.

CALLBACKS

connect better

Copyright 2014 All rights reserved.

SISCI API – Callbacks

- Callbacks is used as an alternative method for the SCIWaitFor...() functions
- Callback won't block the application
- The SISCI API supports segment, DMA and interrupt callbacks
- A callback function and a callback parameter must be specified
- **SCI_FLAG_USE_CALLBACK** must be specified in the appropriate function calls

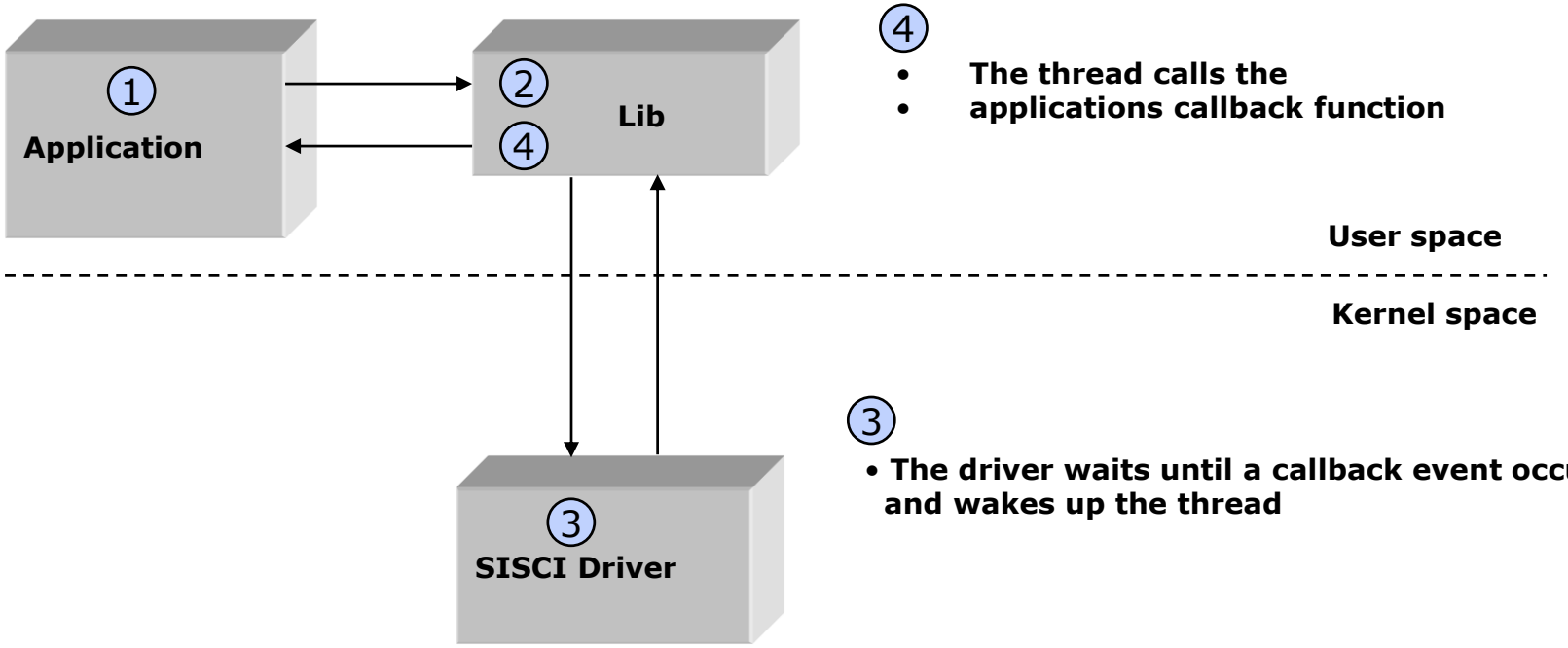
connect better

Copyright 2014 All rights reserved.

SISCI API – Callback implementation

1. An application does a call to the function with CALLBACK specified.

2.
- A thread is created in the library
- The library thread calls waitFor...()
- The application thread returns from the function and can continue the execution



4.
- The thread calls the
- applications callback function

3.
- The driver waits until a callback event occurs and wakes up the thread

connect better

Copyright 2014 All rights reserved.

SISCI API – DMA Callback

- The DMA callback is trigged when the DMA has finished

```
SCIStartDmaTransferVec(sci_dma_queue_t       dq,
                       sci_local_segment_t   localSegment,
                       sci_remote_segment_t remoteSegment,
                       unsigned int          vecLength,
                       sci_dma_vec_t        *sciDmaVec,
                       sci_cb_dma_t          callback,
                       void                  *callbackArg,
                       unsigned int          flags,
                       sci_error_t           *error);

SCIStartDmaTransferVec(dmaQueue,
                       localSegment,
                       remoteSegment,
                       vectorLength,
                       sci_dma_vec,
                       callback_func,          ←
                       args,                   ←
                       SCI_FLAG_USE_CALLBACK,  ←
                       &error);
```

connect better

Copyright 2014 All rights reserved.

SISCI API – Local Segment Callback

- **SCICreateSegment(CALLBACK)**
- A local segment callback event is issued every time the segment state is changed
  - Connects or disconnects to the local segment
  - Link status change (operational/not operational)
  - A connection is lost on a remote node
- Callback reasons:
  - SCI_CB_CONNECT
  - SCI_CB_DISCONNECT
  - SCI_CB_OPERATIONAL
  - SCI_CB_NOT_OPERATIONAL
  - SCI_CB_LOST

connect better

Copyright 2014 All rights reserved.

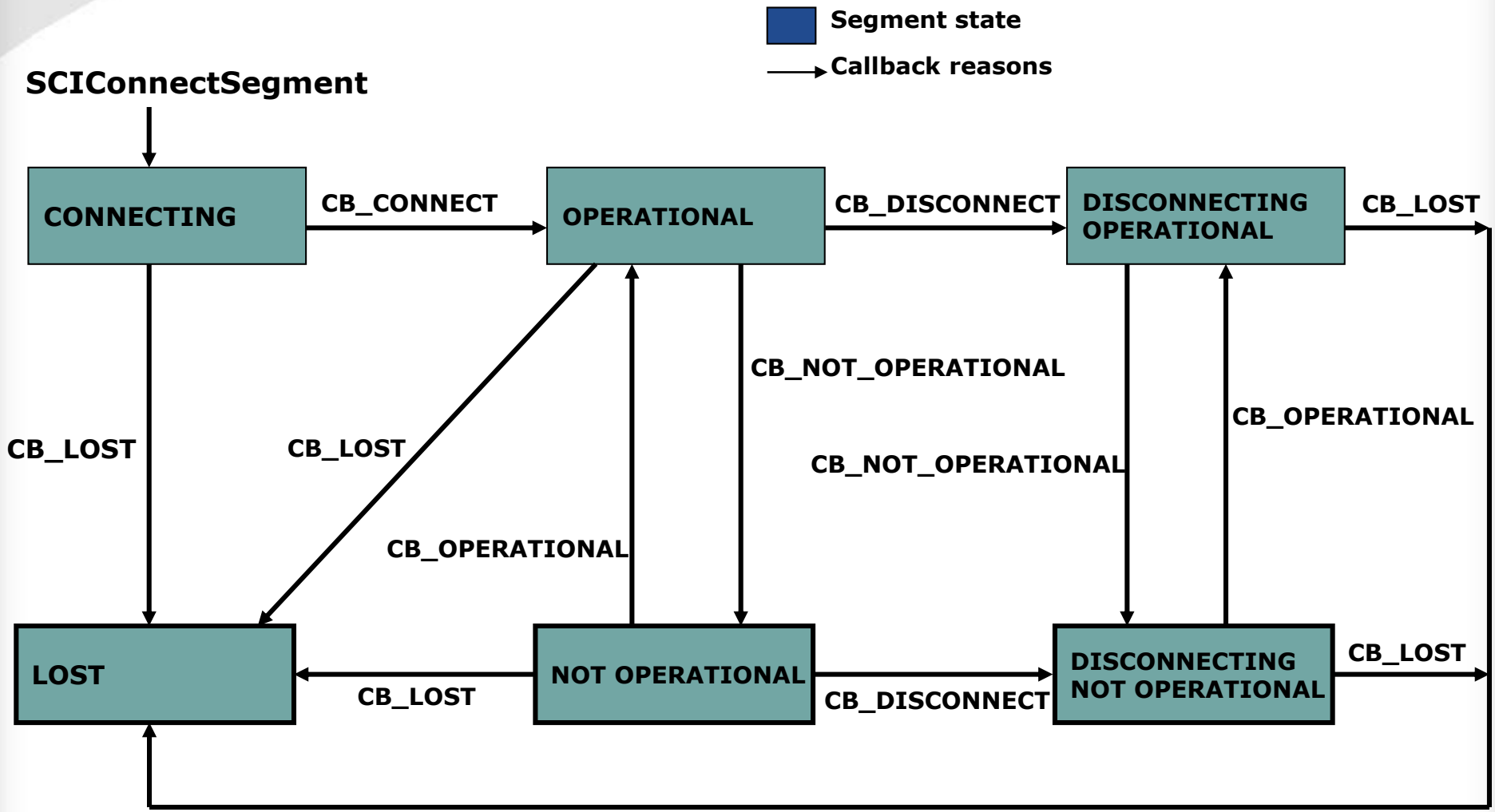
SISCI API – Remote Segment Callback

- **SCIConnectSegment(CALLBACK)**
- A remote segment callback event is issued every time the segment state has changed
  - Link status change (operational/not operational)
  - The connection is lost to the remote node
- If a CB_DISCONNECT callback is issued, it's a request from the remote node to disconnect
  - It's a request not a demand

connect better

Copyright 2014 All rights reserved.

State diagram for remote segment callback



connect better

Copyright 2014 All rights reserved.

SISCI API – Interrupt callback

- **SCICreateInterrupt(CALLBACK)**
- An interrupt callback is issued every time an interupt from a remote node is seen on the interrupt handle

connect better

Copyright 2014 All rights reserved.

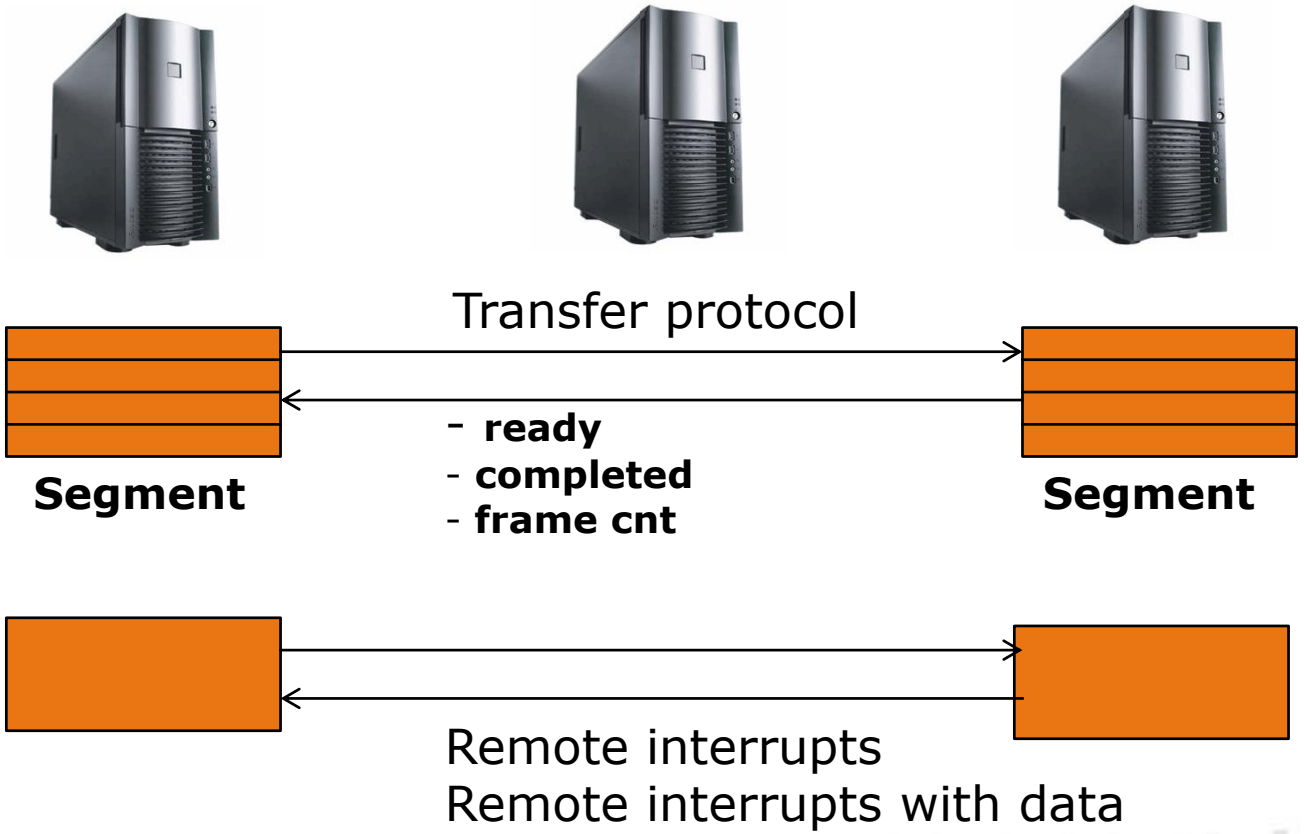
SYNCHRONIZATION

connect better

Copyright 2014 All rights reserved.

SISCI API – Protocol synchronization

- *Synchronization between machines is often required*



connect better

Copyright 2014 All rights reserved.

SISCI API – Protocol synchronization

- Separate segment for synchronization only
  - Make a synchronization protocol
- Remote interrupts
- A combination of a protocol segment and remote interrupts

connect better

Copyright 2014 All rights reserved.

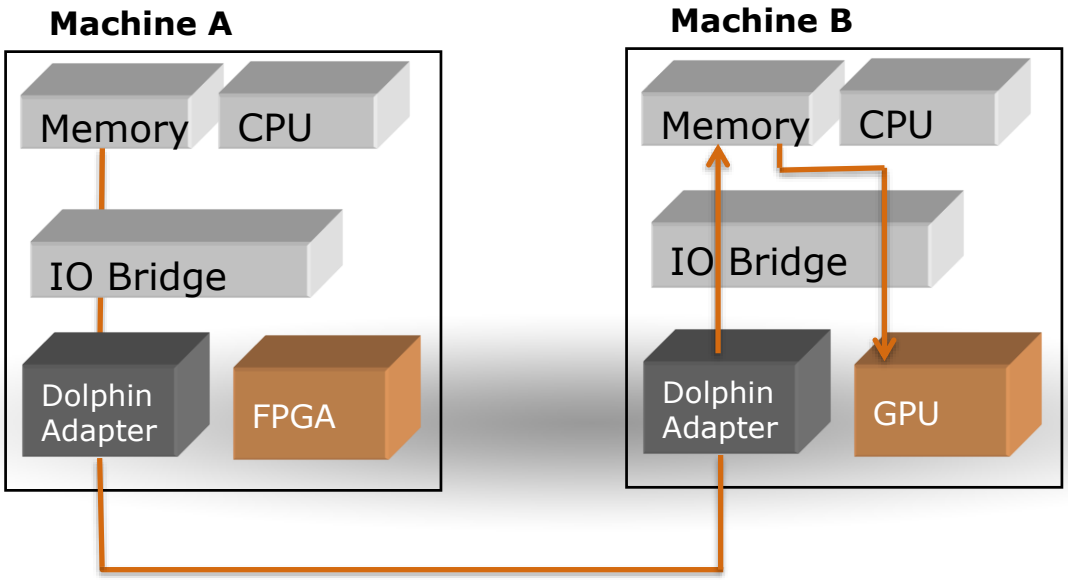
DIRECT TRANSFERS

connect better

Copyright 2014 All rights reserved.

SISCI API – Remote P2P

- Remote access to PCIe devices



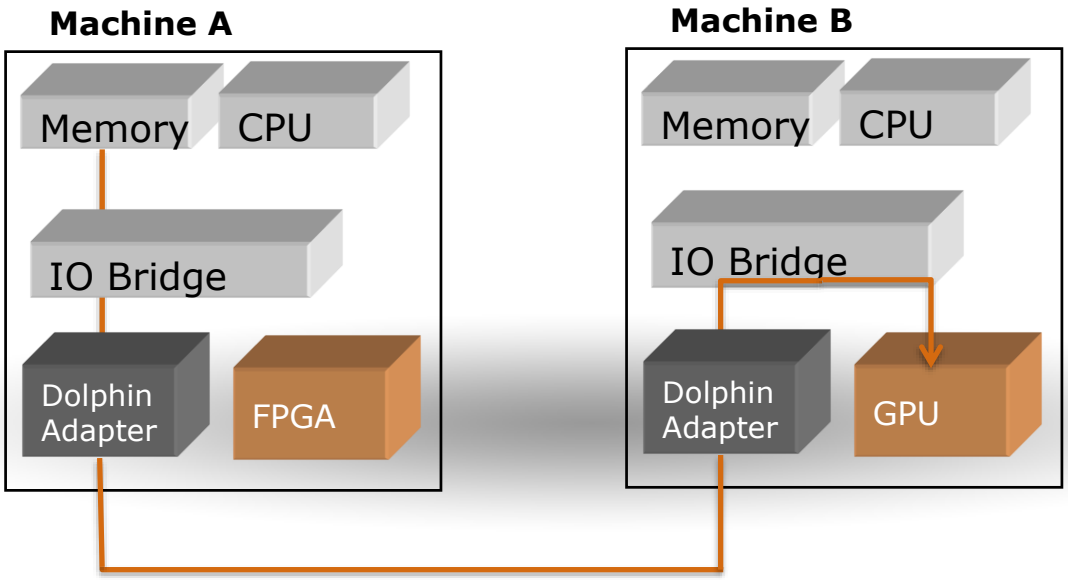
- CPU access to remote GPU
- Not possible to read direct from a GPU on a Tegra machine

connect better

Copyright 2014 All rights reserved.

SISCI API – Remote P2P

- Direct remote access to PCIe devices



- CPU access directly into remote GPU

connect better

Copyright 2014 All rights reserved.

SISCI API – SCIAttachPhysicalMemory()

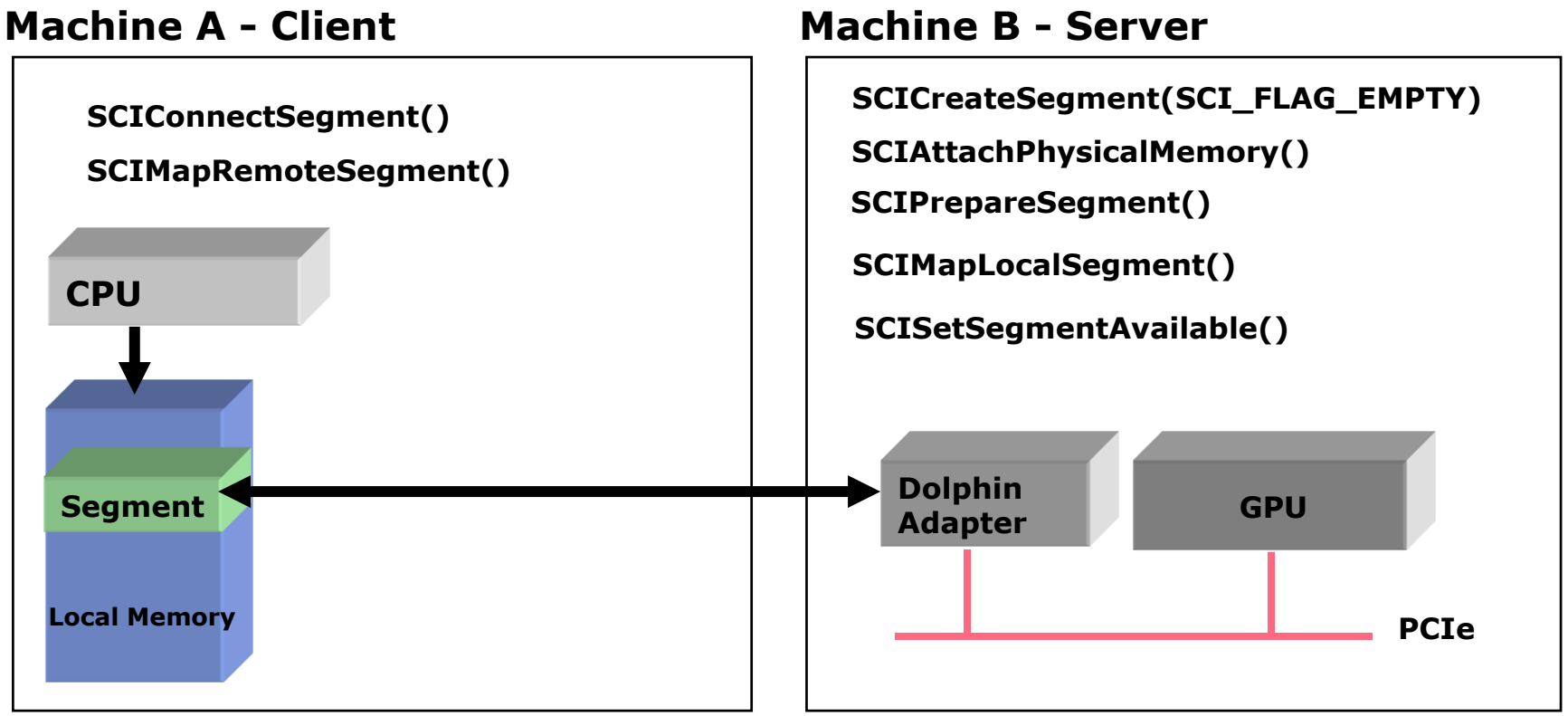
- **SCIAttachPhysicalMemory()**
- Enable the possibility to set up a transfer into other devices than main memory
- In normal case, the transfer is between the local segment allocated in local memory and the remote node
- This function enables attachment of physical memory regions where the Physical PCIe bus address (and mapped CPU address) is already known
- The function will attach the physical memory to the SISCI segment which later can be connected and mapped as a regular SISCI segment
- The mechanism can can attach and transfer data directly to/from an extern PCIe board through the Dolphin adapter
  - Memory boards
  - Preallocated main memory
  - IO device e.g. GPUs

connect better

Copyright 2014 All rights reserved.

SISCI API - SCIAttachPhysicalMemory

- **SCICreateSegment()** with flag **SCI_FLAG_EMPTY** must have been called in advance



connect better

Copyright 2014 All rights reserved.

SISCI API – SCIAttachPhysicalMemory()

```
SCIAttachPhysicalMemory(sci_ioaddr_t          ioaddress,
                        void                  *virt_addr,
                        unsigned int          busNo, (reserved)
                        size_t                size,
                        sci_local_segment_t   segment,
                        unsigned int          flags,
                        sci_error_t           *error);
```

sci_ioaddr_t ioaddress : PCIe address to the device (virt_address can also be used)

void * virt_addr: This is the (mapped) virtual address that the application has to use to access the device. This means that the device has to be mapped in advance by the devices own driver.

busNo: The bus number on the local PCIe

```
SCIAttachPhysicalMemory(0,
                        (void *)cuda_buffer_ptr,
                        0,
                        segmentSize,
                        localSegment,
                        SCI_FLAG_CUDA_BUFFER,
                        &error);
```

connect better

Copyright 2014 All rights reserved.

EXAMPLE PROGRAMS

connect better

Copyright 2014 All rights reserved.

SISCI API – Example programs

- Simple example applications are available to demonstrate the SISCI API interface
  - Located in the **/opt/DIS/src/** directory
- Test and benchmark application programs are located in the **/opt/DIS/bin** directory
  - Testing of the system
  - Benchmarking
- Available as source code and binaries

connect better

Copyright 2014 All rights reserved.

SUMMARY

connect better

Copyright 2014 All rights reserved.

dis_diag utility

- [test_machine]# dis_diag
- Number of configured local adapters found: 1
- Adapter 0 > Type : MXH910
- NodeId : 4
- Serial number : MXH910-AA-003347
- MXH chipId : 0x4028
- MXH chip revision : 0x1
- EEPROM version NTB mode : 003
- Card revision : AA
- Topology type : Switch

- ********** IXH ADAPTER 0 LINK 0 STATE **********

- Link 0 uptime : 677597 seconds
- Link 0 state : ENABLED
- Link 0 state : x8, Gen3 (8 GT/s)
- Link 0 cable inserted : 1
- Link 0 port active : 1

connect better

Copyright 2014 All rights reserved.

dis_diag utility

- Link 0 uptime : 677597 seconds
- Link 0 state : ENABLED
- Link 0 state : x8, Gen2 (5 GT/s)
- Link 0 cable inserted : 1
- Link 0 port active : 1
- ********** MXH ADAPTER 0, PARTNER INFORMATION FOR LINK 0 **********
- Partner board type : MXS824
- Partner switch no : TOP, Port 4
- Partner number of ports : 12

- ********** TOPOLOGY SEEN FROM ADAPTER 0 **********
- Adapters found: 4
- ----- List of all nodes found:
- **Nodes detected: 0004 0008 0012 0016**

- ********** SESSION STATUS FROM ADAPTER 0 **********
- **Node 4: Session valid**
- **Node 8: Session valid**
- **Node 12: Session valid**
- **Node 16: Session valid**
- ---
- dis_diag discovered 0 note(s).
- dis_diag discovered 0 warning(s).
- dis_diag discovered 0 error(s).
- TEST RESULT: *PASSED*

connect better

Copyright 2014 All rights reserved.

Hints

- Run tests and benchmarks
- Step by step approach
- PIO Reads are slow
  - Use **Write only** approach if possible
- Limit the use of remote interrupts
- Use **SCIFlush()** in the end of a block transfer
  - SCIMemCpy() uses build in SCIFlush()
- Use DMA for large data transfers

connect better

Copyright 2014 All rights reserved.

SISCI API – Example programs

Example programs: **/opt/DIS/src/examples**

/opt/DIS/src/demo

- shmem.c
- dmacb.c
- dmavec.c
- interrupt.c
- intcb.c
- queryseg.c
- reflective_write.c

- Benchmark applications: **/opt/DIS/src/bin**
  - Scibench2 (PIO benchmark)
  - Scipp (PIO Latency)
  - dma_bench (DMA benchmark)
  - intr_bench (INTERRUPT benchmark)
- Header files: **/opt/DIS/include**
  - sisci_api.h
  - sisci_error.h

connect better

Copyright 2014 All rights reserved.

Documentation and Support

- SISCI Documentation:
  - User Guide
  - API specification
  - https://docs.dolphinics.com/SISCI_v2/
  - http://docs.dolphinics.com/
  - **http://docs.dolphinics.com/sisci/guides/**

- In case of questions regarding the SISCI API:

simula-support@dolphinics.no

connect better

Copyright 2014 All rights reserved.