Communications on Distributed Architectures

Arnaud LEGRAND, CR CNRS, LIG/INRIA/Mescal
Jean-Louis ROCH, MCF ENSIMAG, LIG/INRIA/Moais

Vincent DANJEAN, MCF UJF, LIG/INRIA/Moais
Derick KONDO, CR INRIA, LIG/INRIA/Mescal
Jean-François MÉHAUT, PR UJF, LIG/INRIA/Mescal
Bruno RAFFIN, CR INRIA, LIG/INRIA/Moais
Alexandre TERMIER, MCF UJF, LIG/Hadas

October, 13th 2008

Goals of this lecture

Understand how communication libraries can efficiently use high speed networks

Understand the limitation of such libraries

Outlines

High Speed Networks

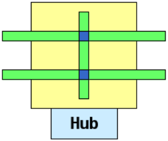
High Speed Networks are used in clusters

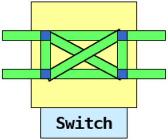
- low distance
- very interesting performance
  - low latency: about 1 μs
  - high bandwidth: about 10 Gb/s
- specific light protocols
  - static routing of messages
  - no required packet fragmentation
  - sometimes, no packet required

Myrinet, Quadrics, SCI, ...

(Fast|Giga)-Ethernet

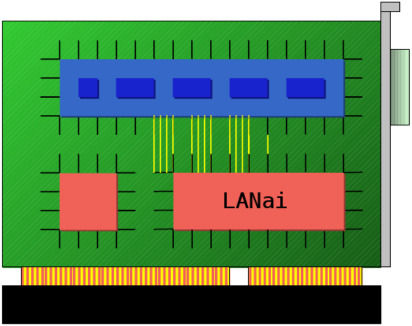
- Interconnect:
  - Hub or switch
- Wires:
  - Copper or optical fiber
- Latency:
  - about 10 μs
- Bandwidth:
  - From 100 Mb/s to 10 Gb/s
- Remark:
  - compatible with traditional Ethernet





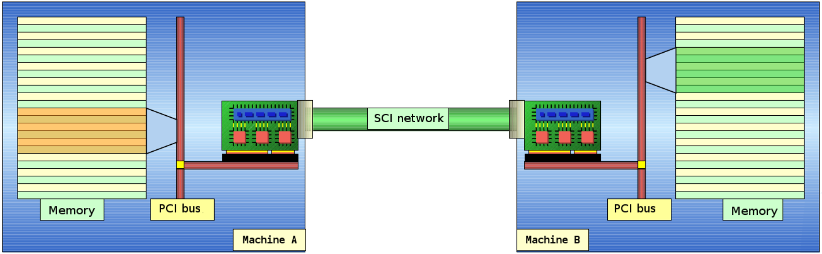
Myrinet

- Myricom corporate
- Interconnect:
  - Switch
- PCI card with:
  - a processor: LANai
  - SRAM memory: about 4 MB
- Latency:
  - about 1 or 2 μs
- Bandwidth:
  - 10 Gb/s
- Remark:
  - static, wormhole routing



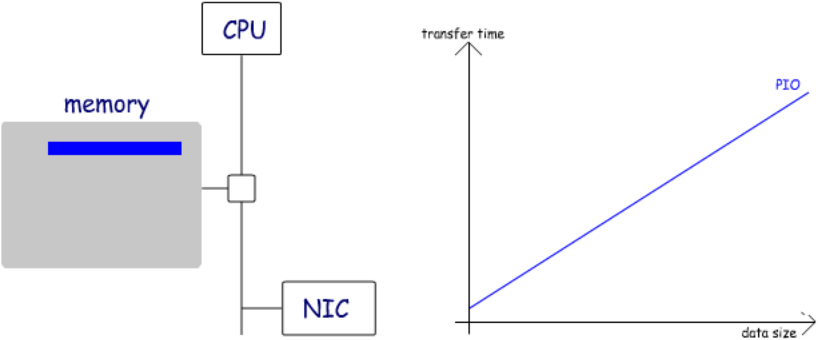
SCI

- Scalable Coherent Interface
  - IEEE norm (1993)
  - Dolphin corporate
- Uses remote memory access:
  - Address space remotely mapped



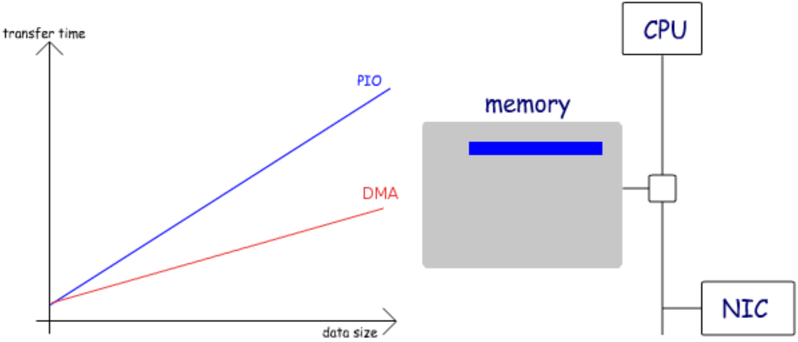
Outlines

Interacting with the network card: PIO mode



Programmed Input/Output

Interacting with the network card: DMA mode



Direct Memory Access

Zero-copy communications

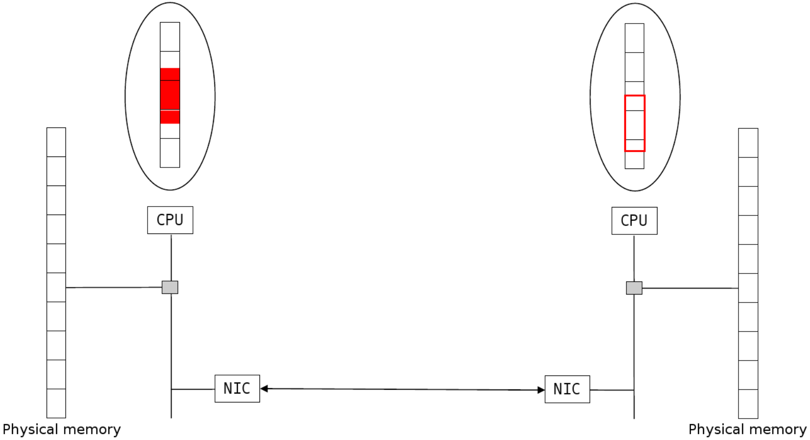
Goals

- Reduce the communication time
  - Copy time cannot be neglected
    - but it can be partially recovered with pipelining
- Reduce the processor use
  - currently, `memcpy` are executed by processor instructions

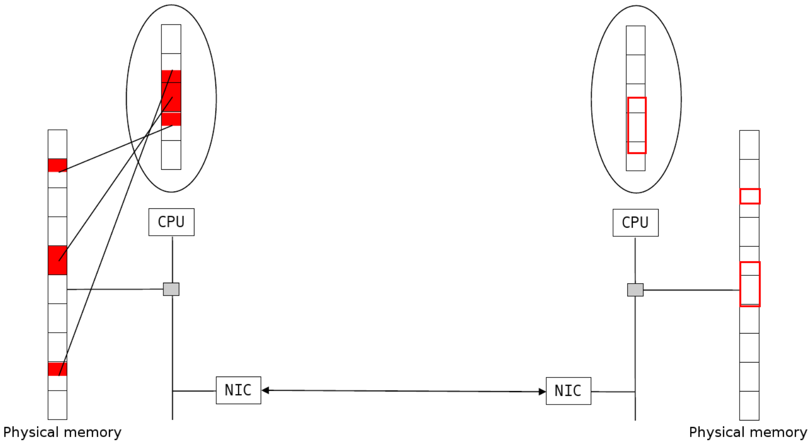
Idea

The network card directly read/write data from/to the application memory

Zero-copy communications



Zero-copy communications



Zero-copy communications for emission

PIO mode transfers

- No problem for zero-copy

DMA mode transfers

- Non contiguous data in physical memory
- Headers added in the protocol
  - linked DMA
  - limits on the number of non contiguous segments

Zero-copy communications for reception

A network card cannot “freeze” the received message on the physical media

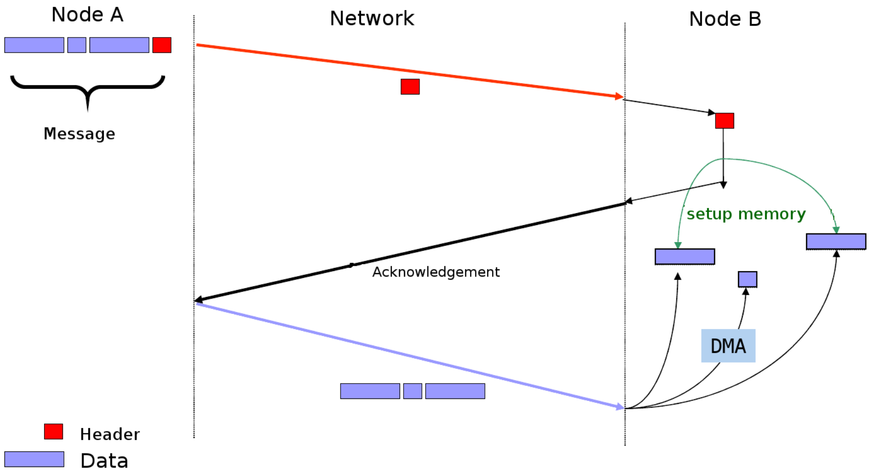
If the receiver posted a “recv” operation before the message arrives

- zero-copy OK if the card can filter received messages
- else, zero-copy allowed with bounded-sized messages with optimistic heuristics

If the receiver is not ready

- A handshake protocol must be setup for big messages
- Small messages can be stored in an internal buffer

Using a Handshake Protocol



A few more considerations

The receiving side plays an important role

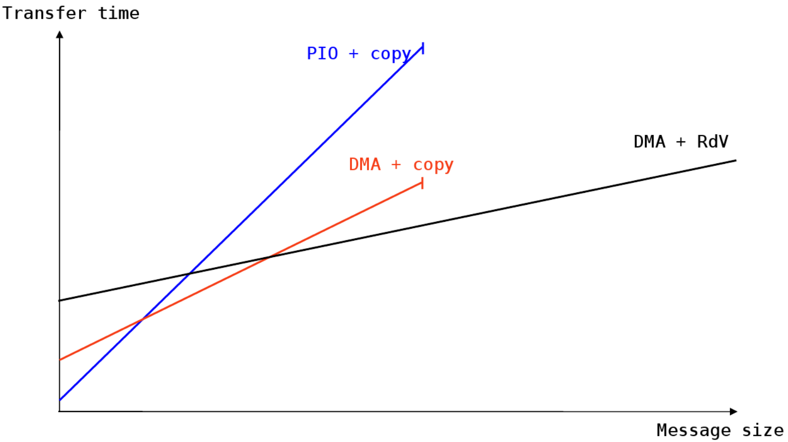
- Flow-control is mandatory
- Zero-copy transfers
  - the sender has to ensure that the receiver is ready
  - a handshake (REQ+ACK) can be used

Communications in user-space introduce some difficulties

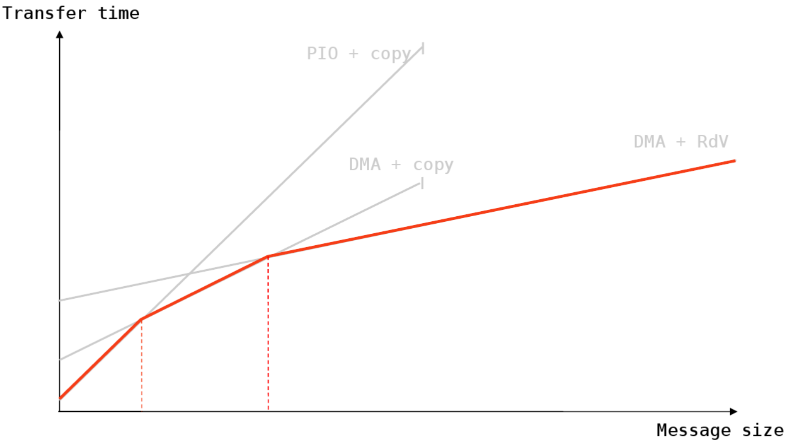
- Direct access to the NIC
  - most technologies impose “pinned” memory pages

Network drivers have limitations

Communication Protocol Selection

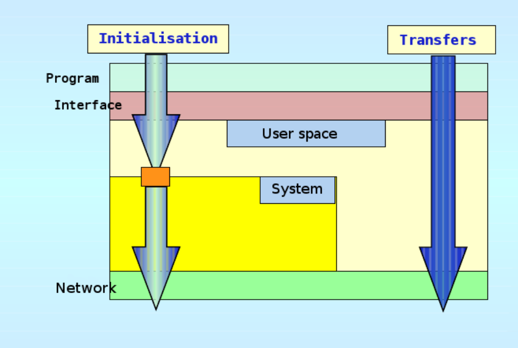


Communication Protocol Selection



Operating System Bypass

- Initialization
  - traditional system calls
  - only at session beginning
- Transfers
  - direct from user space
  - no system call
  - “less” interrupts
- Humm… And what about security ?

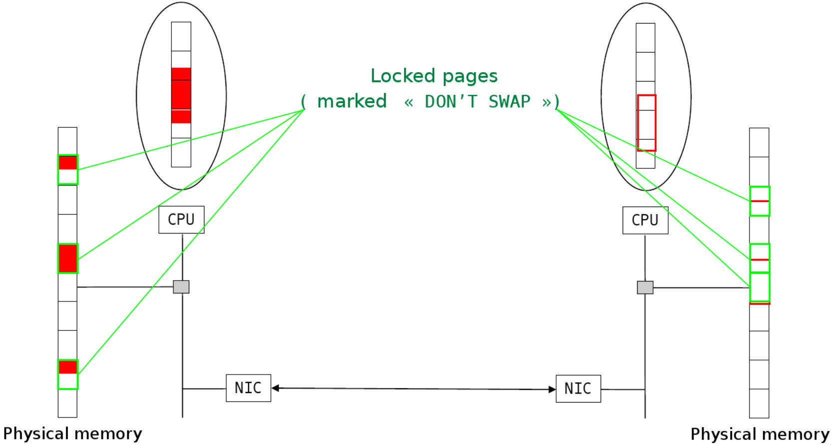


OS-bypass + zero-copy

Problem

- Zero-copy mechanism uses DMA that requires physical addresses
- Mapping between virtual and physical address is only known by:
  - the processor (MMU)
  - the OS (pages table)
- We need that
  - the library knows this mapping
  - this mapping is not modified during the communication
    - ex: swap decided by the OS, copy-on-write, etc.
- No way to ensure this in user space !

OS-bypass + zero-copy



OS-bypass + zero-copy

First solution

- Pages “recorded” in the kernel to avoid swapping
- Management of a cache for virtual/physical addresses mapping
  - in user space or on the network card
- Diversion of system calls that can modify the address space

Second solution

- Management of a cache for virtual/physical addresses mapping on the network card
- OS patch so that the network card is “advertised” when a modification occurs
- Solution chosen by MX/Myrinet and Elan/Quadrics

Direct consequences

- Latency measure can vary whether the memory region used
  - Some pages are “recorded” within the network card
- Ideal case are ping-pong exchanges
  - The same pages are reused hundred of times
- Worst case are applications using lots of different data regions. . .

Outlines

BIP/Myrinet

- Basic Interface for Parallelism
  - L. Prylli and B. Tourancheau
- Dedicated to Myrinet networks
- Characteristics
  - Asynchronous communication
  - No error detection
  - No flow control
    - Small messages are copied into a fixed buffer at reception
    - Big messages are lost if the receiver is not ready

MX/Myrinet

- Myrinet eXpress
  - Official driver from Myricom
- Very simplistic interface to allow easy implementation of MPI
  - Flow control
  - Reliable communications
  - Non contiguous messages
  - Multiplexing

SiSCI/SCI

- Driver for SCI cards
- Programming model
  - Remote memory access
    - Explicit: RDMA
    - Implicit: memory projections
- Performance
  - Explicit use of some operation required:
    - memory “flush”
    - `SCI_memcpy`
    - RDMA

VIA

- Virtual Interface Architecture
- A new standard ?
  - Lots of industrials
    - Microsoft, Intel, Compaq, etc.
- Characteristics
  - Virtual interfaces objects
    - Queues of descriptors (for sending and receiving)
  - Explicit memory recording
  - Remote reads/writes
    - RDMA

Summary

Very specific programming interfaces

- dedicated to specific technologies (but VIA)
- different programming models
- quasi no portability

It is not reasonable to program a scientific application directly with such programming interfaces

Outlines

Message Passing Interface

Characteristics

- Interface (not implementation)
- Different implementations
  - MPICH
  - LAM-MPI
  - OpenMPI
  - and all closed-source MPI dedicated to specific hardware
- MPI 2.0 begins to appear

Several Ways to Exchange Messages with MPI

`MPI_Send` (standard)

- At the end of the call, data can be reused immediately

`MPI_Bsend` (buffered)

- The message is locally copied if it cannot be send immediately

`MPI_Rsend` (ready)

- The sender “promises” that the receiver is ready

`MPI_Ssend` (synchronous)

- At the end of the call, the reception started (similar to a synchronization barrier)

Non Blocking Primitives

MPI_Isend / MPI_Irecv (immediate)

```
MPI_request r;

MPI_Isend(..., data, len, ..., &r)

// Calculus that does not modify
'data'
MPI_wait(&r, ...);
```

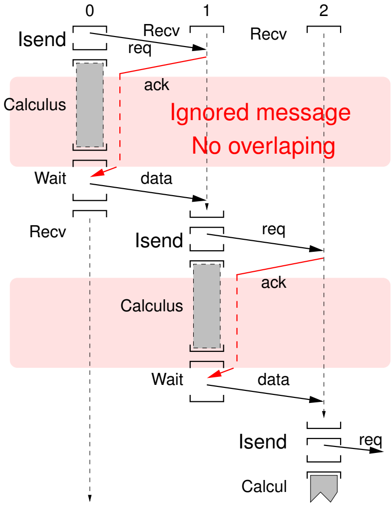
These primitives must be used as much as possible

About MPI Implementations

- MPI is available on nearly all existing networks and protocols!
    - Ethernet, Myrinet, SCI, Quadrics, Infiniband, IP, shared memory, etc.
- MPI implementation are really efficient
    - low latency (hard), large bandwidth (easy)
    - optimized version from hardware manufacturers (IBM, SGI)
    - implementations can be based on low-level interfaces
        - MPICH/Myrinet, MPICH/Quadrics

BUT these “good performance” are often measured with ping-pong programs. . .

Communicating while Computing



Token circulation while computing on 4 nodes

```
if (mynode!=0)
  MPI_Recv();

req=MPI_Isend(next);
Work(); /* about 1s */
MPI_Wait(req);

if (mynode==0)
  MPI_Recv();
```

Communicating while Computing

Problem

- The process does other things when the ACK occurs

Solutions

- Using threads within MPI (MPICH/Madeleine)
- Implementing part of the protocol in the network card (MPICH/GM)
- Using remote memory reads

Token circulation while computing on 4 nodes

```
if (mynode!=0)
  MPI_Recv();

req=MPI_Isend(next);
Work(); /* about 1s */
MPI_Wait(req);

if (mynode==0)
  MPI_Recv();
```

- expected time: ~ 1 s
- observed time: ~ 4 s

Communicating while Computing

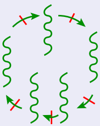
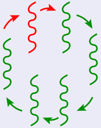
Low-level libraries sometimes prefer using the processor in order to guaranty low latencies

- Depending on the message size
  - PIO for small messages
  - Pipelined copies with DMA for medium messages
  - Zero-copy + DMA for large messages
- Example: limit medium/large is set to 32 KB for MX
  - sending messages from 0 to 32 KB cannot overlap computations

Independent Communication Progression

Using threads and scrutations

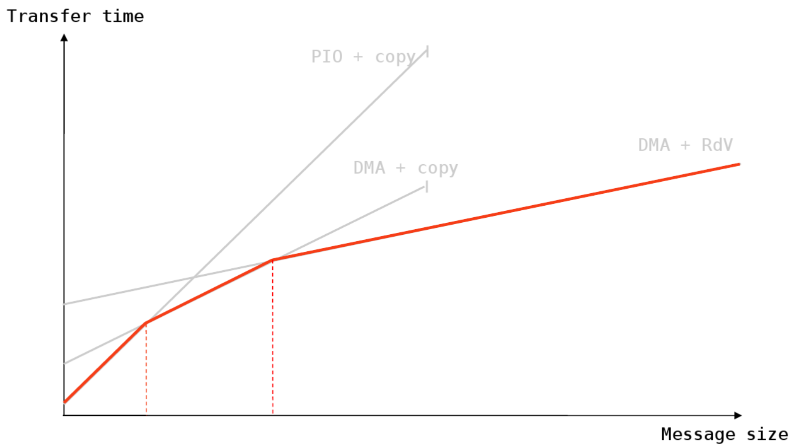
- difficult to implement
- some threads library support can help to get guarantee frequency for scrutation
  - independent with respect to the number of threads in the application


instead of


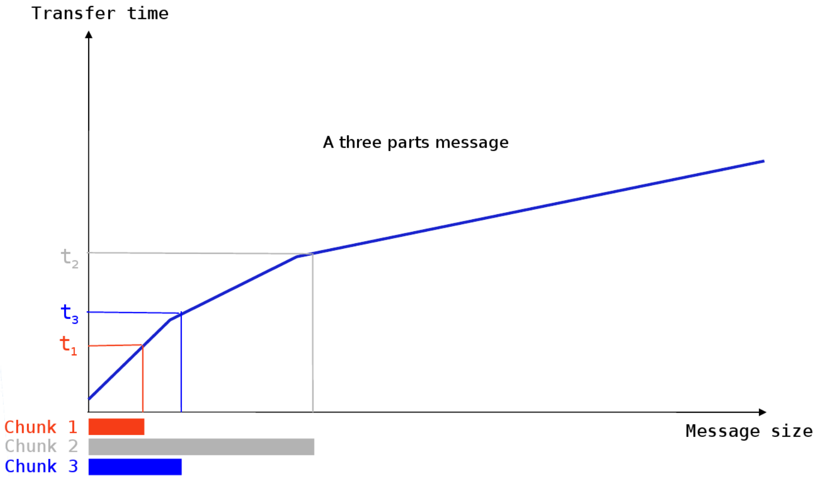
Using specialized firmware on network cards

- Require a processor on the network card
- Myrinet, Quadrics

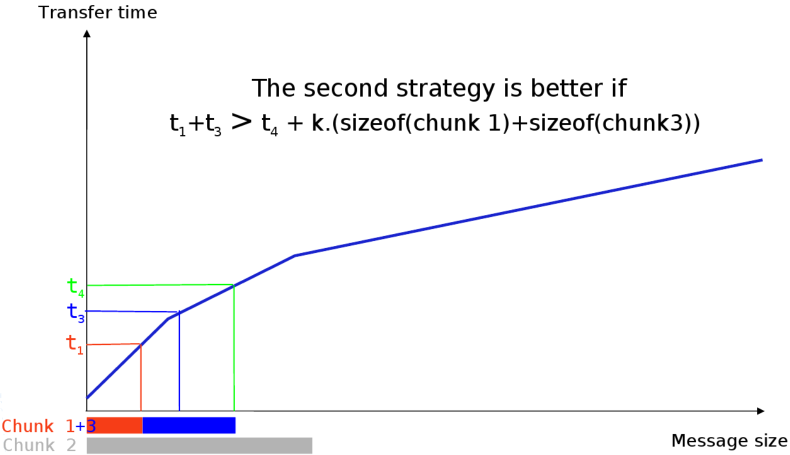
Choosing the Optimal Strategy



Choosing the Optimal Strategy



Choosing the Optimal Strategy



Choosing the Optimal Strategy

It depends on

- The underlying network with driver performance
  - latency
  - PIO and DMA performance
  - Gather/Scatter feature
  - Remote DMA feature
  - etc.
- Multiple network cards ?

But also on

- memory copy performance
- I/O bus performance

Efficient AND portable is not easy

Outlines

Key points

Using high-speed networks require using lots of optimization techniques

These optimizations, often mono-criteria, are deep inside communication libraries

It can be needed to use low-level interfaces to keep an absolute control over communications

The future

Better cooperation between languages and communication libraries

Better management or hierarchical configurations

Distribution of network work on large multiprocessors architectures