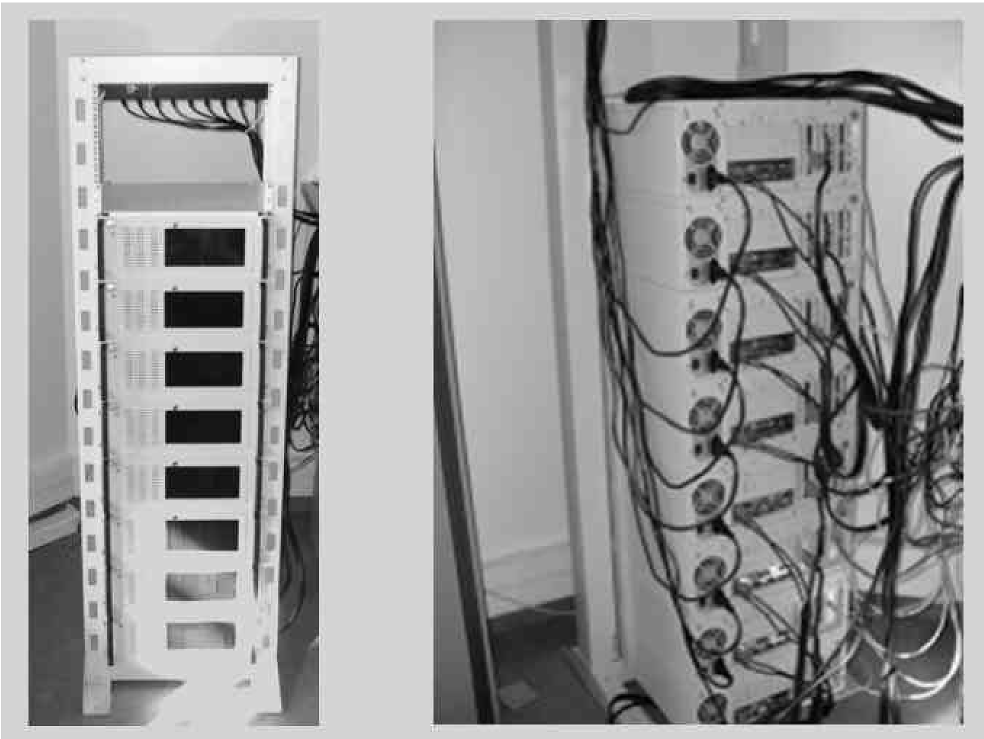
The influence of system calls and interrupts on the performances of a PC cluster using a Remote DMA communication primitive



Olivier Glück

Jean-Luc Lamotte

Alain Greiner

Univ. Paris 6, France

http://mpc.lip6.fr

Olivier.Gluck@lip6.fr

Outline

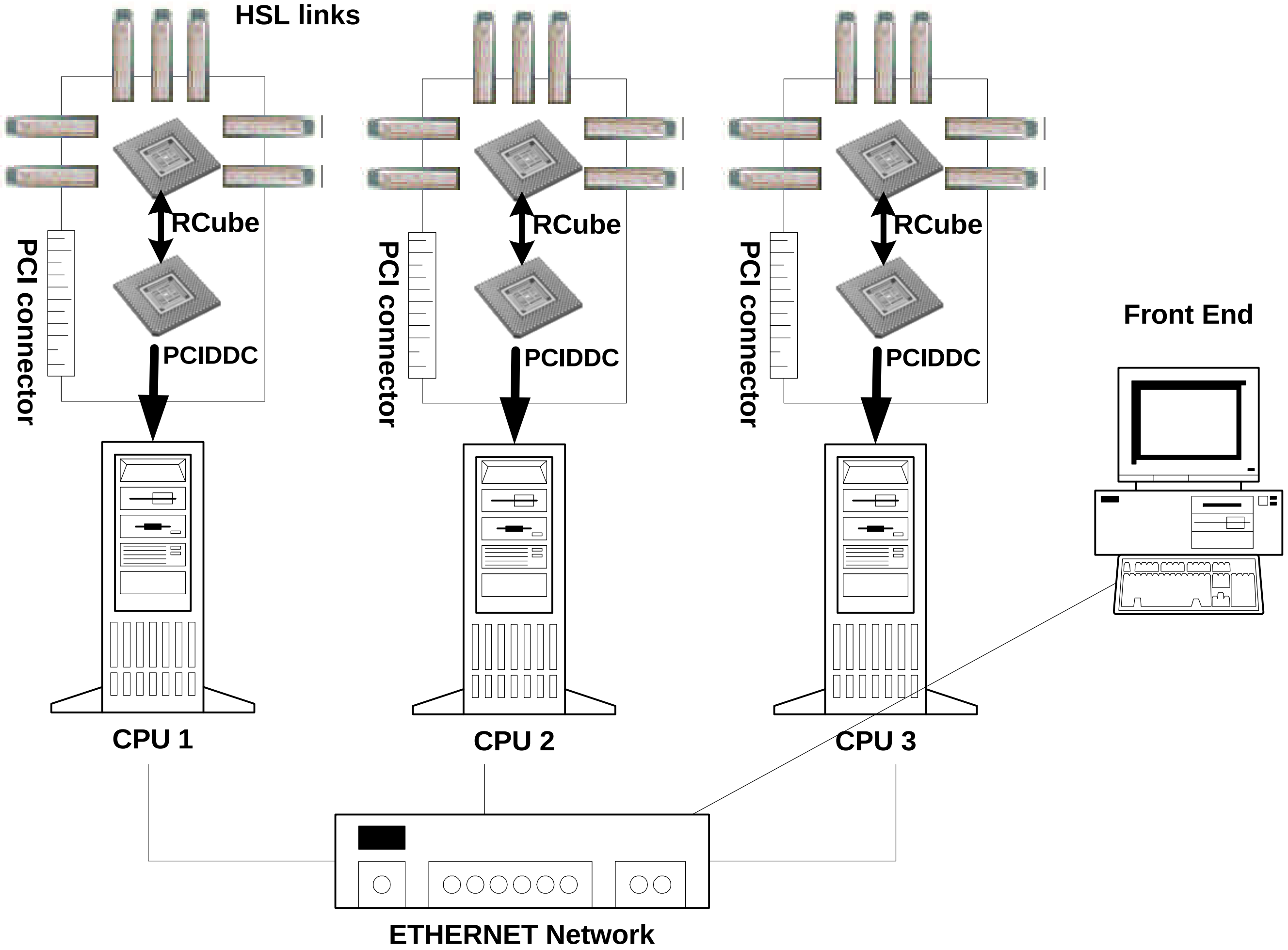
1. Introduction
2. The MPC parallel computer
3. MPI-MPC1: the first implementation of MPICH on MPC
4. MPI-MPC2: user-level communications
5. Comparison of both implementations
6. A realistic application
7. Conclusion

Introduction

- Very low cost and high performance parallel computer
- PC cluster using optimized interconnection network
- A PCI network board (FastHSL) developed at LIP6:
  - High speed communication network (HSL,1 Gbit/s)
  - RCUBE: router (8x8 crossbar, 8 HSL ports)
  - PCIDDC: PCI network controller (a specific communication protocol)
- Goal: supply efficient software layers

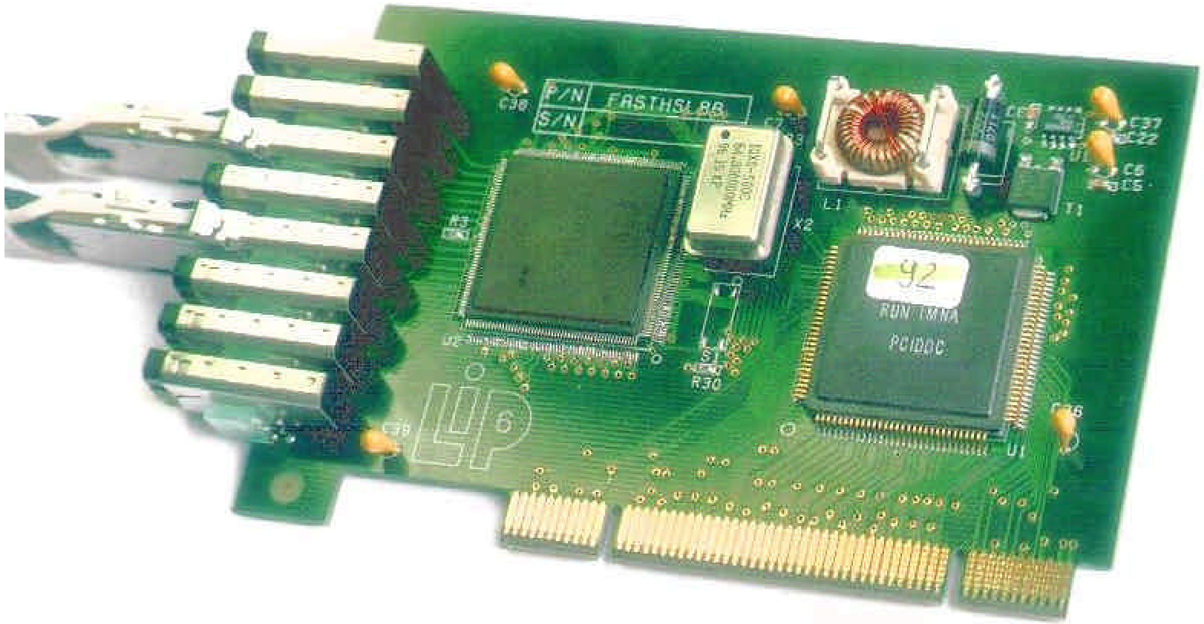
Þ **A specific high-performance implementation of MPICH**

The MPC computer architecture



Our MPC parallel computer

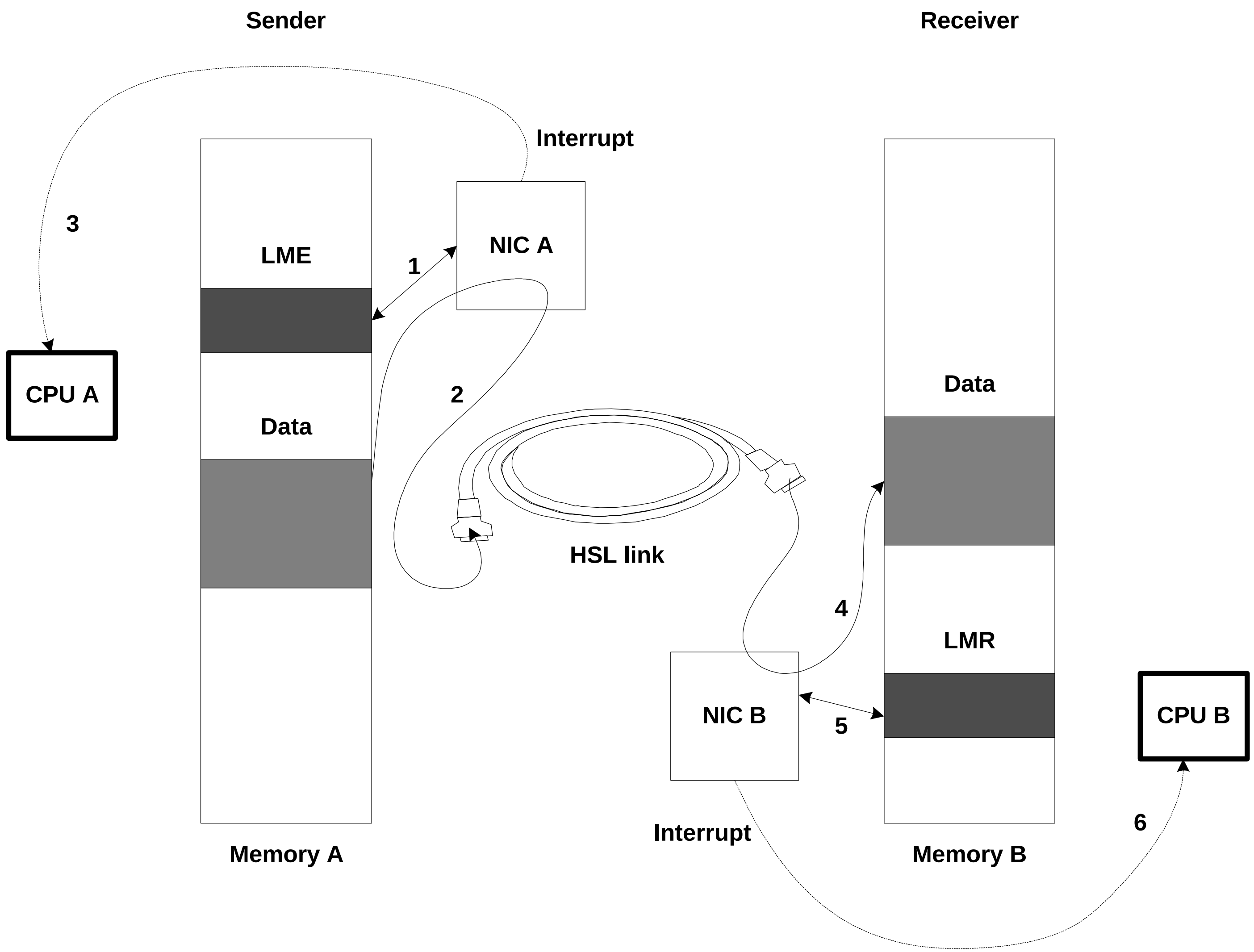
The FastHSL PCI board



Hardware performances:

- **latency: 2 µs**
- **Maximum throughput on the link: 1 Gbits/s**
- **Maximum useful throughput: 512 Mbits/s**

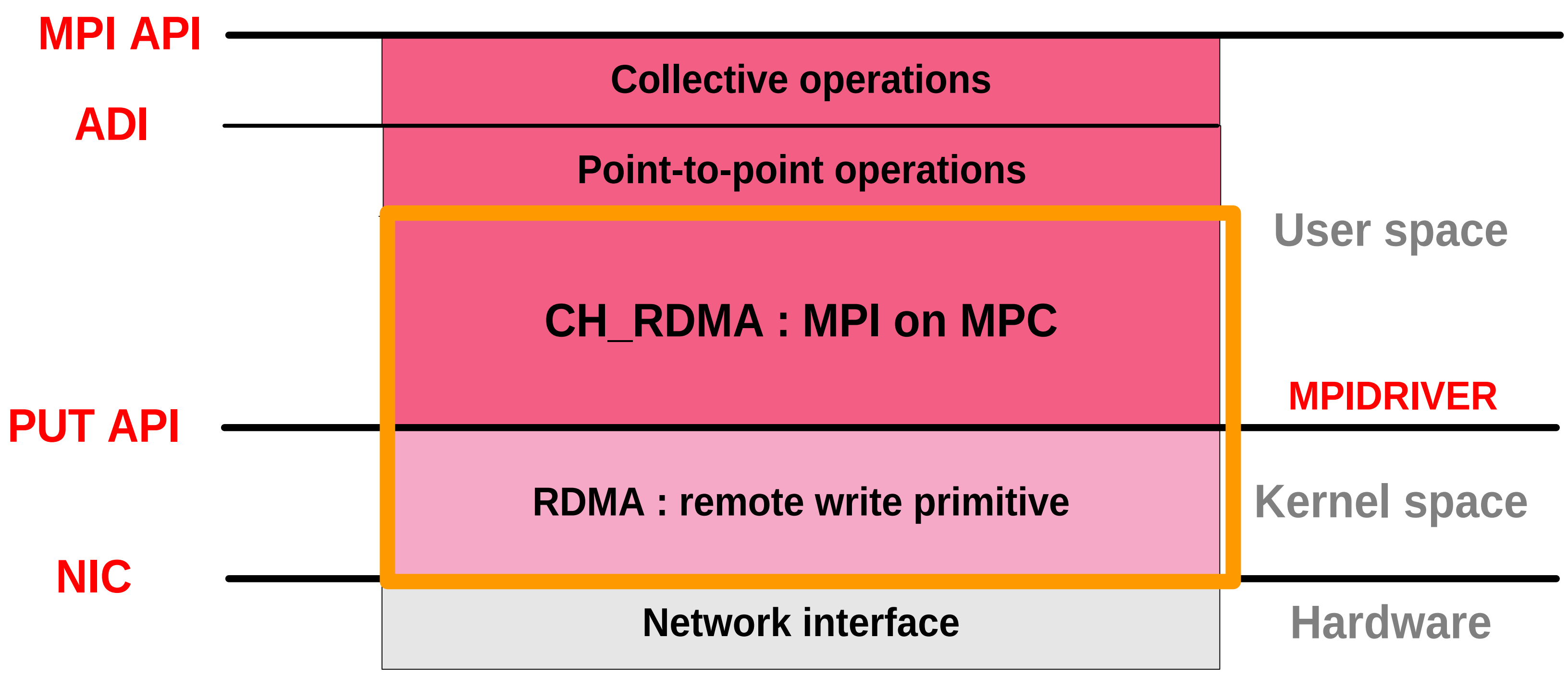
The remote write primitive (RDMA)



PUT: the lowest level software API

- Unix based layer: FreeBSD or Linux
- Provides a basic kernel API using the PCI-DDC remote write
- Implemented as a module
- Handles interrupts
- Zero-copy strategy using physical memory addresses
- Parameters of 1 PUT call:
  - remote node identifier,
  - local physical address,
  - remote physical address,
  - data length, …
- Performances:
  - 5 µs one-way latency
  - 494 Mbits/s

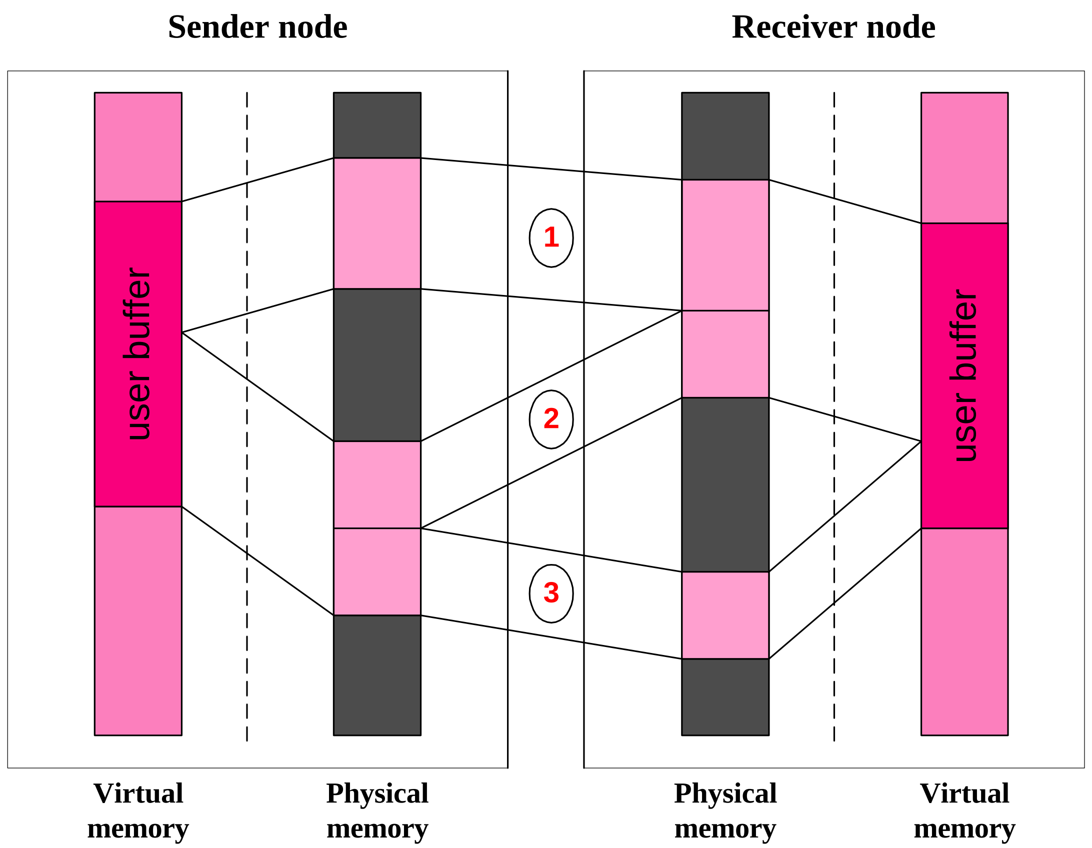
MPI-MPC1 architecture



MPICH on MPC: 2 main problems

Virtual/physical address translation?

Where to write data in remote physical memory?



MPICH requirements

Two kinds of messages:

- **CTRL messages: control information or limited size user-data**
- **DATA messages: user-data only**

Services to supply:

- **Transmission of CTRL messages**
- **Transmission of DATA messages**
- **Network event signaling**
- **Flow control for CTRL messages**

Þ **Optimal maximum size of CTRL messages?**

Þ **Match the Send/Receive semantic of MPICH to the remote write semantic**

MPI-MPC1 implementation (1)

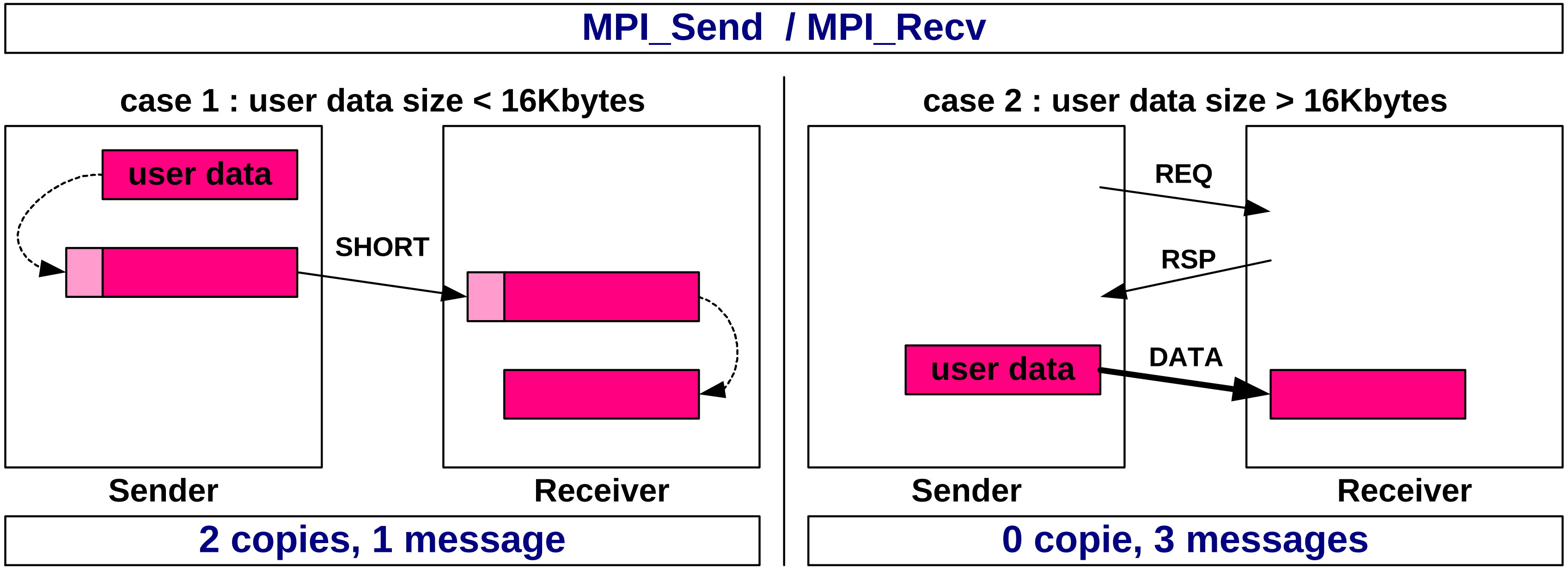
CTRL messages:

- **pre-allocated buffers, contiguous in physical memory, mapped in virtual process memory**
- **an intermediate copy on both sender & receiver**
- **4 types:**
  - **SHORT: user-data encapsulated in a CTRL message**
  - **REQ: request of DATA message transmission**
  - **RSP: reply to a request**
  - **CRDT: credits, used for flow control**

DATA messages:

- **zero-copy transfer mode**
- **rendezvous protocol using REQ & RSP messages**
- **physical memory description of remote user buffer in RSP**

MPI-MPC1 implementation (2)



MPI-MPC1 performances

Each call to the PUT layer = 1 system call

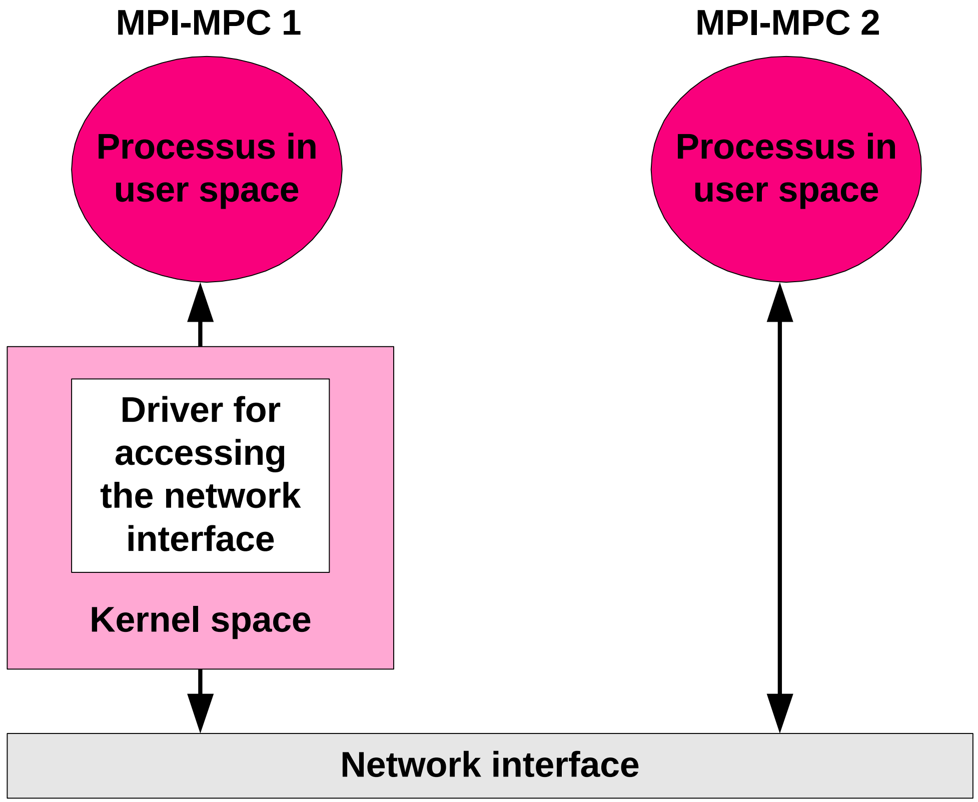
Network event signaling uses hardware interrupts

Performances of MPI-MPC1:

- **benchmark: MPI ping-pong**
- **platform: 2 MPC nodes with PII-350**
- **one-way latency: 26 µs**
- **throughput: 419 Mbits/s**

Þ **Avoid system calls and interrupts**

MPI-MPC1 & MPI-MPC2



- Post remote write orders in user mode
- Replace interrupts by a polling strategy

MPI-MPC2 implementation

Network interface registers are accessed in user mode

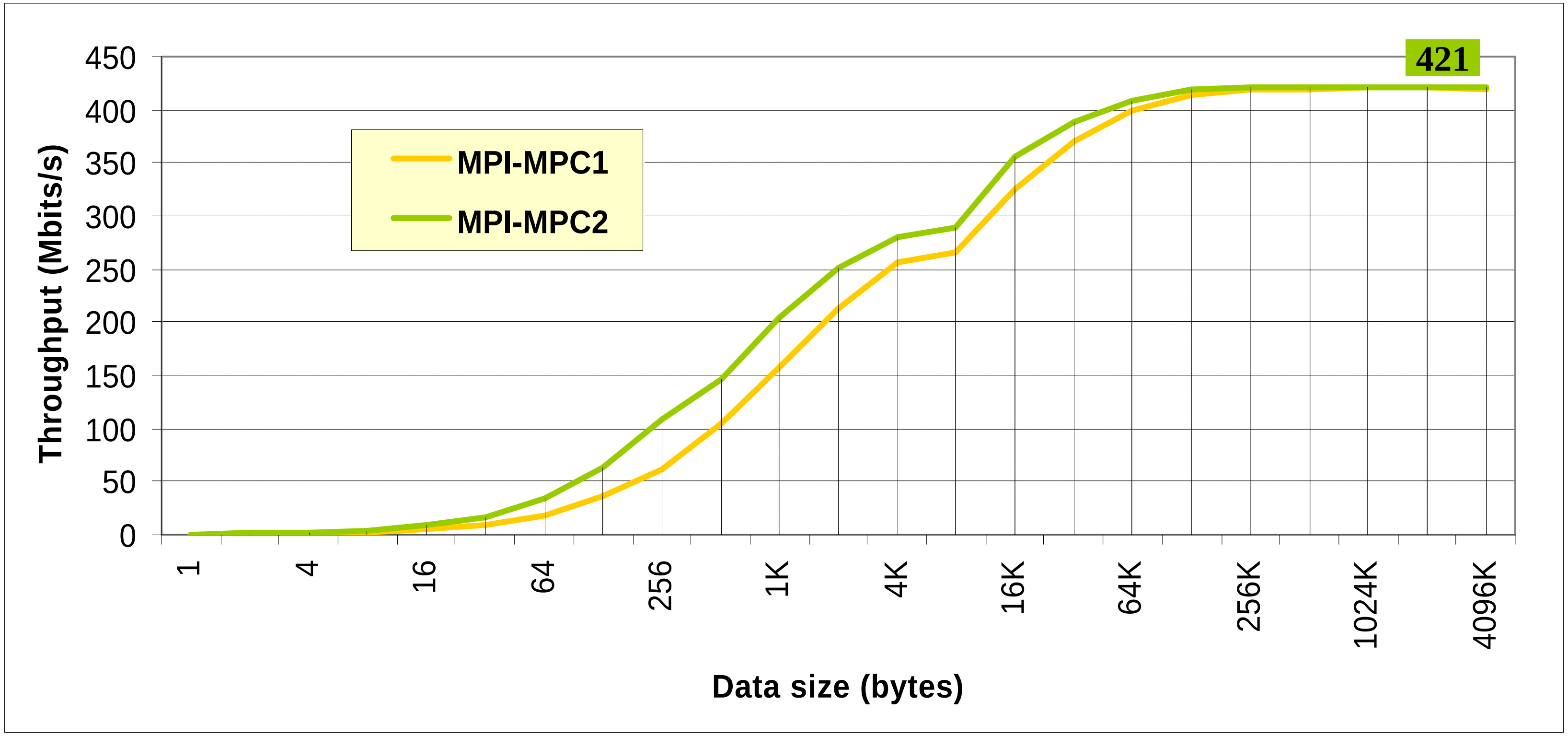
Exclusive access to shared network resources:

- **shared objects are kept in the kernel and mapped in user space at starting time**
- **atomic locks are provided to avoid possible competing accesses**

Efficient polling policy:

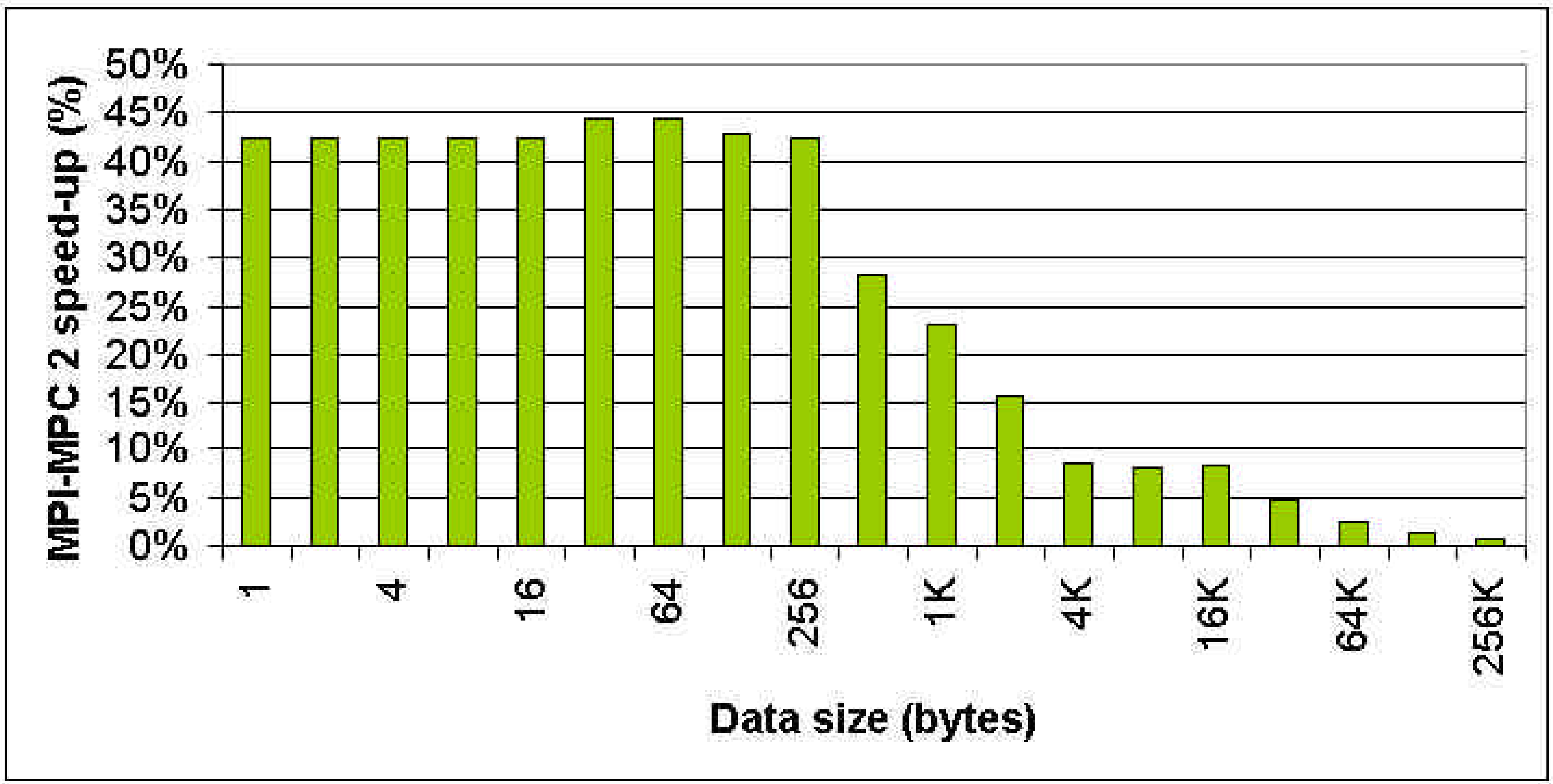
- **polling on the last modified entries of the LME/LMR lists**
- **all the completed communications are acknowledged at once**

MPI-MPC1&MPI-MPC2 performances



MPI-MPC implementation	One-way latency	Throughput
MPI-MPC1	**26 µs**	**419 Mbits/s**
MPI-MPC2	**15 µs**	**421 Mbits/s**

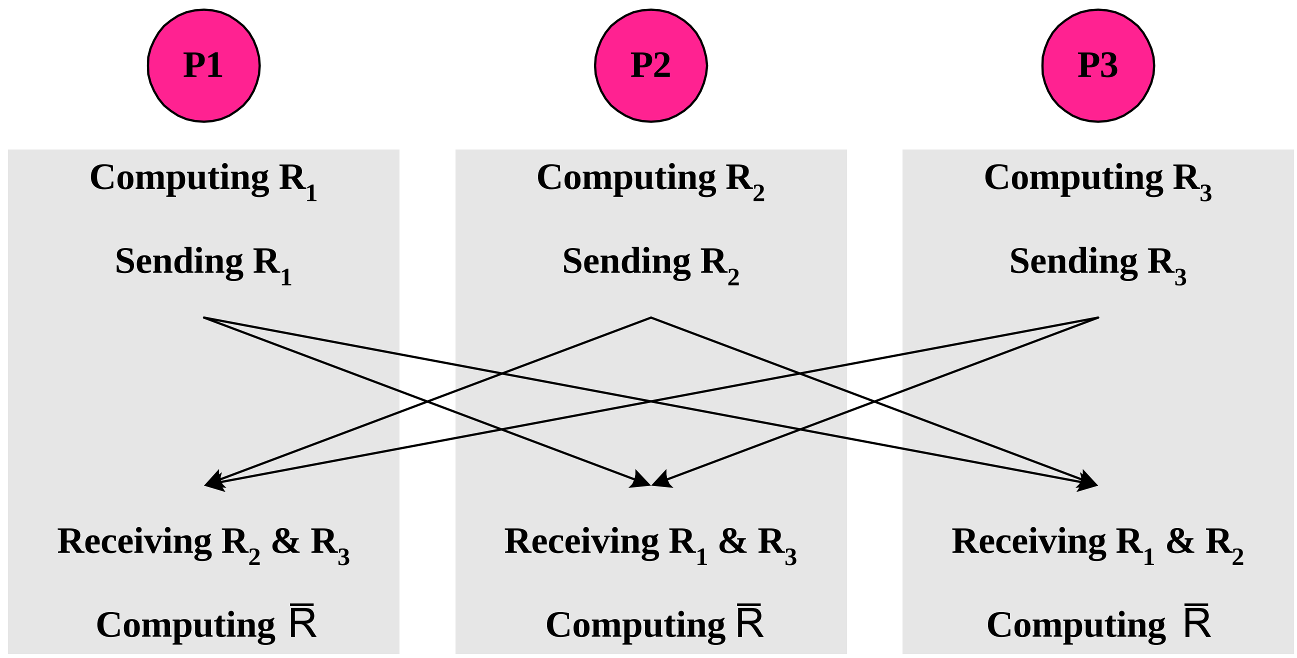
MPI-MPC2 latency speed-up



The CADNA software

CADNA: Control of Accuracy and Debugging for Numerical Applications

- **developed in the LIP6 laboratory**
- **control and estimate the round-off error propagation**



MPI-MPC performances with CADNA

Application: solving a linear system using Gauss method

- **without pivoting: no communication**
- **with pivoting: a lot of short communications**

System size	Number of exchanges	Communication time (sec.)		One exchange time (µs)	
		MPI-MPC1	MPI-MPC2	MPI-MPC1	MPI-MPC2
800	646682	51	31	79	48
1200	1450450	101	66	70	46
1600	2574140	191	128	74	50
2000	4018285	288	177	72	44
Mean value (µs)				**74**	**47**

Þ **MPI-MPC2 speed-up = 36%**

Conclusions & perspectives

- **2 implementations of MPICH on a remote write primitive**
- **MPI-MPC1:**
  - **system calls during communication phases**
  - **interrupts for network event signaling**
- **MPI-MPC2:**
  - **user-level communications**
  - **signaling by polling**
  - **latency speed-up greater than 40% for short messages**
- **What about maximum throughput?**
  - **Locking user buffers in memory and address translations are very expansive**
  - **MPI-MPC3 Þ avoid address translations by mapping the virtual process memory in a contiguous space of physical memory at application starting time**