Autumn Meeting ‘99 of AK „Scientific Computing“

MPICH FOR SCI-CONNECTED CLUSTERS

Joachim Worringen

RWTH Scalable Computing Aachen

Lehrstuhl für Betriebssysteme

AGENDA

- Introduction, Related Work & Motivation
- Implementation
- Performance
- Work in Progress
- Summary

MESSAGE-PASSING

MPI: the de-facto standard for technical and scientific Message-Passing-Programming:

- Open Standard / Open Source implementations
- Reliable specification
- Easy to use, but extensive functionality if required
- Portable code
- Efficient implementation on every architecture
- Many libraries & tools available

MPI ON SAN-CLUSTERS

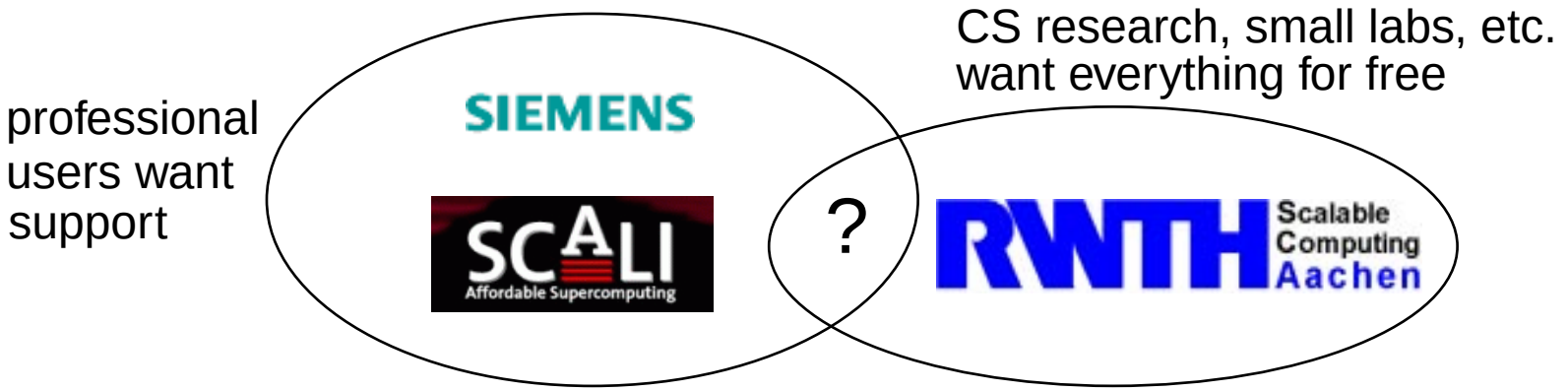
Available SAN-Interconnects with MPI:

- DEC MemoryChannel
  - Digital MPI (by DEC, commercial)
- Giganet
  - MPI/Pro (by MPI Software, commercial)
- Myrinet
  - MPI-FM (by CSAG at UCSD, **freely available**)
  - LAMP (by NEC / GMD, not publically available)
- SCI
  - ScaMPI (by Scali, commercial)
  - Sun MPI (by Sun, commercial)

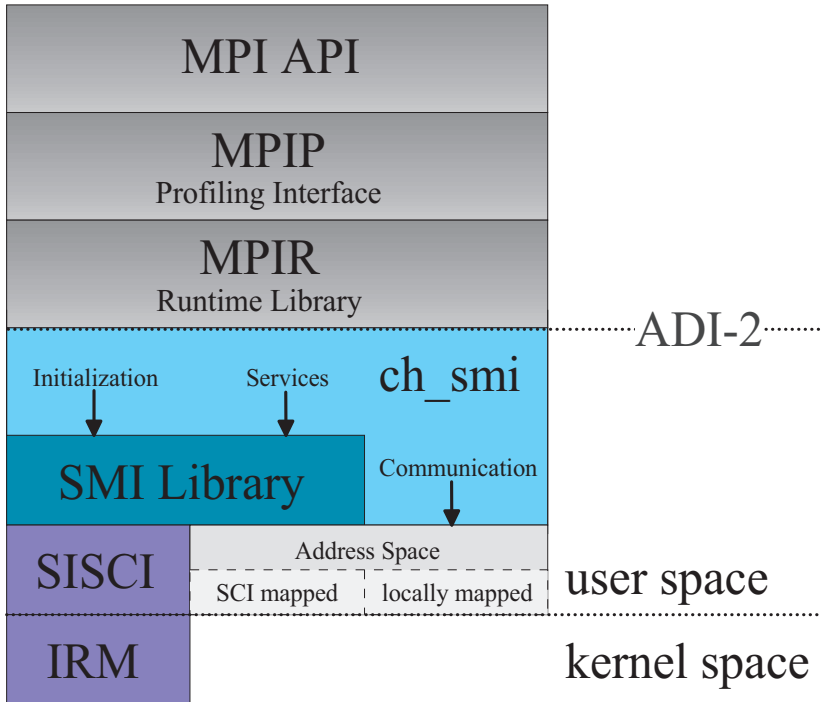
MOTIVATION FOR SCI-MPICH

Lack of free MPI-Implementations for SCI-Clusters:

- researchers, universities etc. prefer free software
- free software eases the decision for a SCI-Cluster
- availabitlity of source code prevents „sudden death“ of the development
- multiple available implementations for one platform stimulate the development



SCI-MPICH Design



- SMI-Library

  „Shared Memory Interface“ offers high-level services based on SISCI:

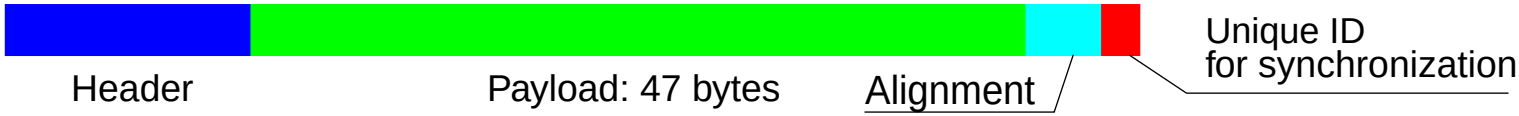
  - Startup, Memory Allocation & Management, Synchronization,

    Load/Store Barriers / Error Checking

MPICH MESSAGE PROTOCOLS

SHORT: Message contents inside a *Control Packet:*

- 64 byte write for each short message
- packet structure with implicit synchronization



EAGER: Message transfer without interaction of the receiver:

- static buffers at the receiver
- ringbuffer of buffer-pointers at the sender
- remote-write of the message followed by control packet

RENDEZ-VOUS: Transfer of arbitrary sized messages:

- dynamic buffer allocation by receiving process
- synchronization between sender & receiver before, during and after transfer
- write-read interleave of transfer buffer

MPI BENCHMARKS

Benchmark Environment:

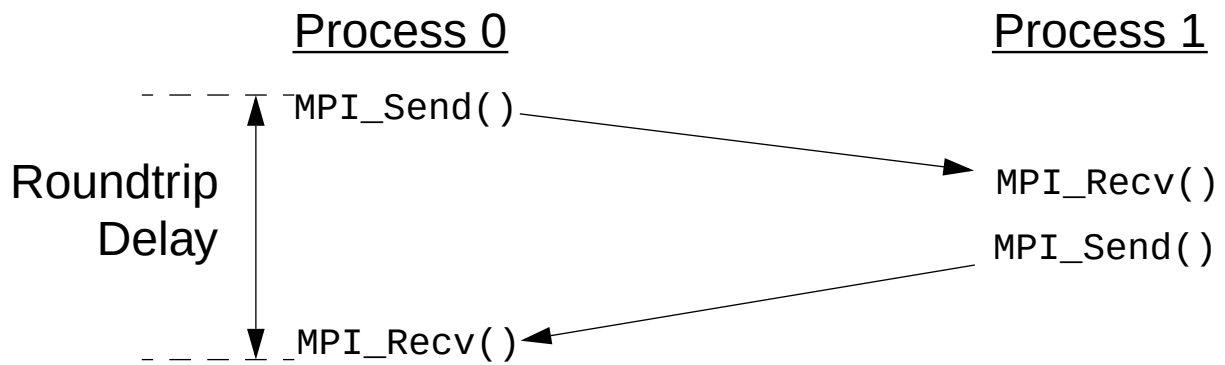
A: MPICH 1.1.2 on PentiumII @ 450 MHz, Solaris 7 with

- ch_smi: communication via SCI (**SCI-MPICH**)
- ch_p4: communication via TCP/IP on FastEthernet
- ch_shmem: communication via Shared Memory

B: ScaMPI 1.7 on PentiumII @ 400 MHz, Linux 2.2

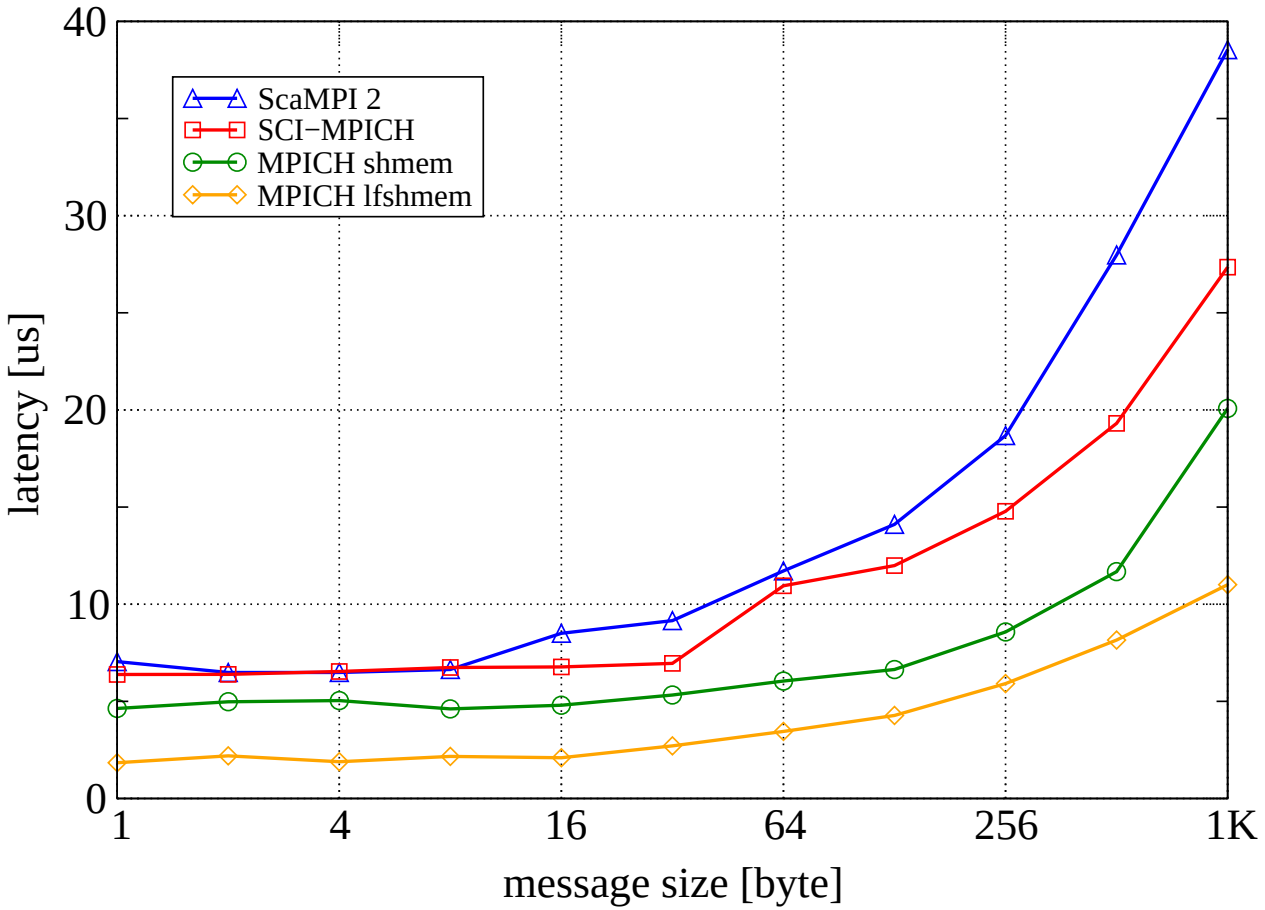
- hpc*L*ine at RWTH Aachen

Point-to-Point Performance:



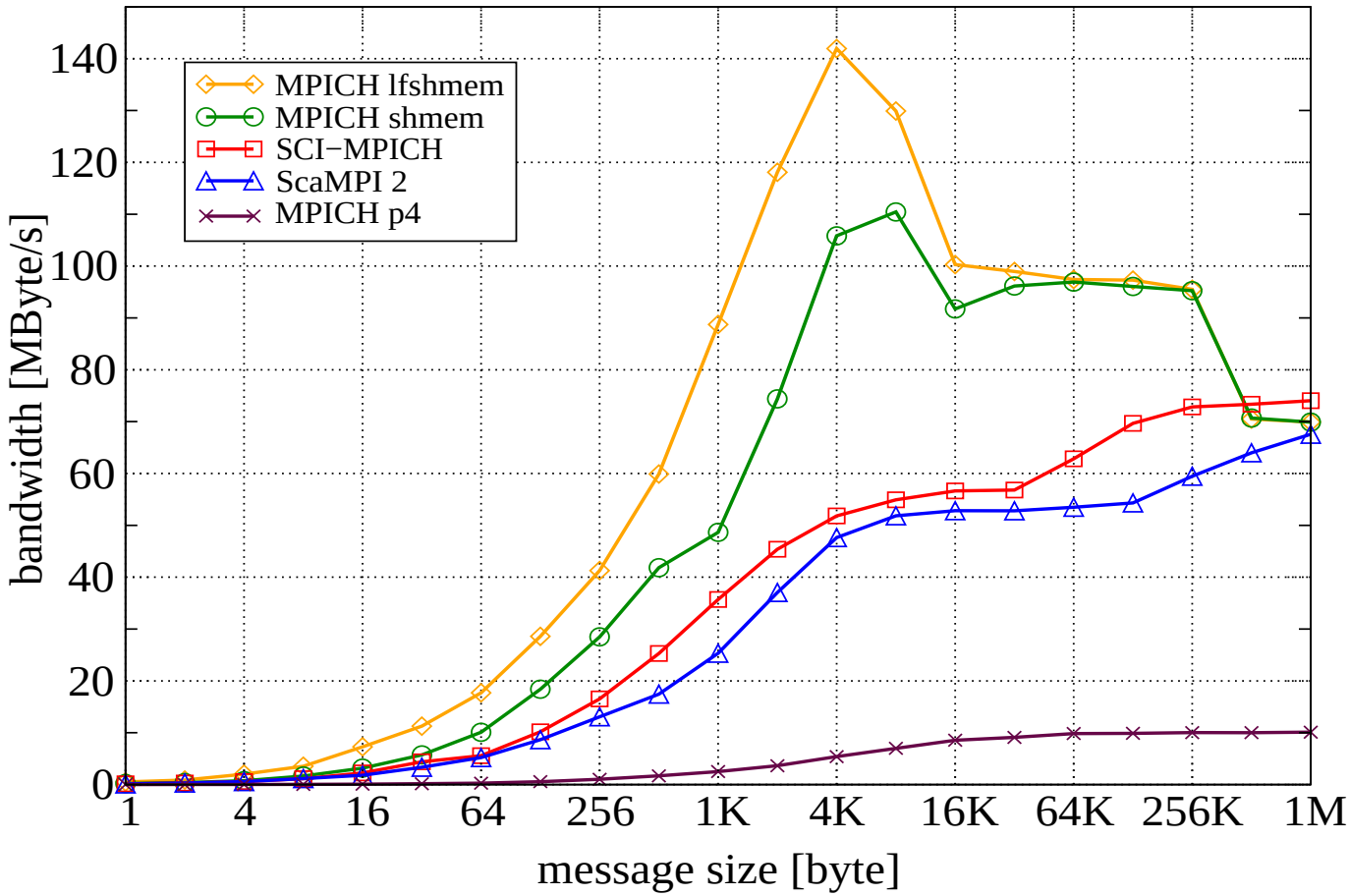
POINT-TO-POINT PERFORMANCE

Roundtrip/2-Latency for blocking send/receive:



POINT-TO-POINT PERFORMANCE

Bandwidth for blocking send/receive:



NAS Parallel Benchmarks

Standard Suite of Parallel Kernel/Application Benchmarks
(NASA Ames Research Center)

Fortran 77 / MPI codes for

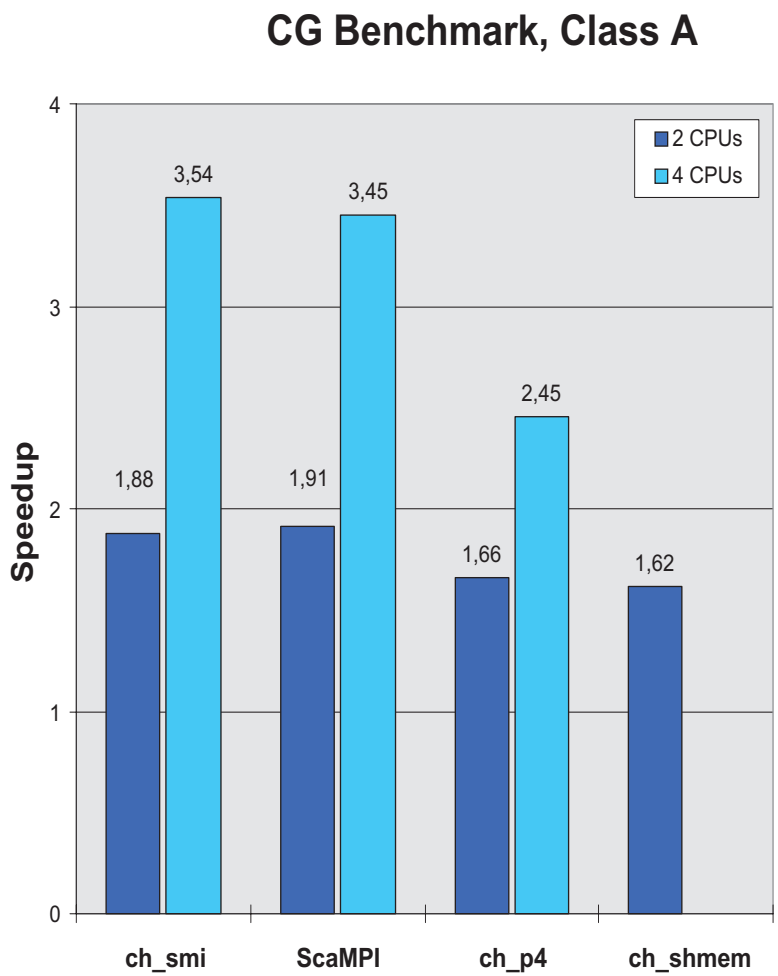
- Simulated CFD application (LU)
- 3D Fast Fourier Transformation (FT)
- Conjugate Gradient (CG)
- 3D Scalar Poisson by Multigrid (MG)
- others (Integer Sort, „Embarassing Parallel“, CFD Block Solvers etc.)

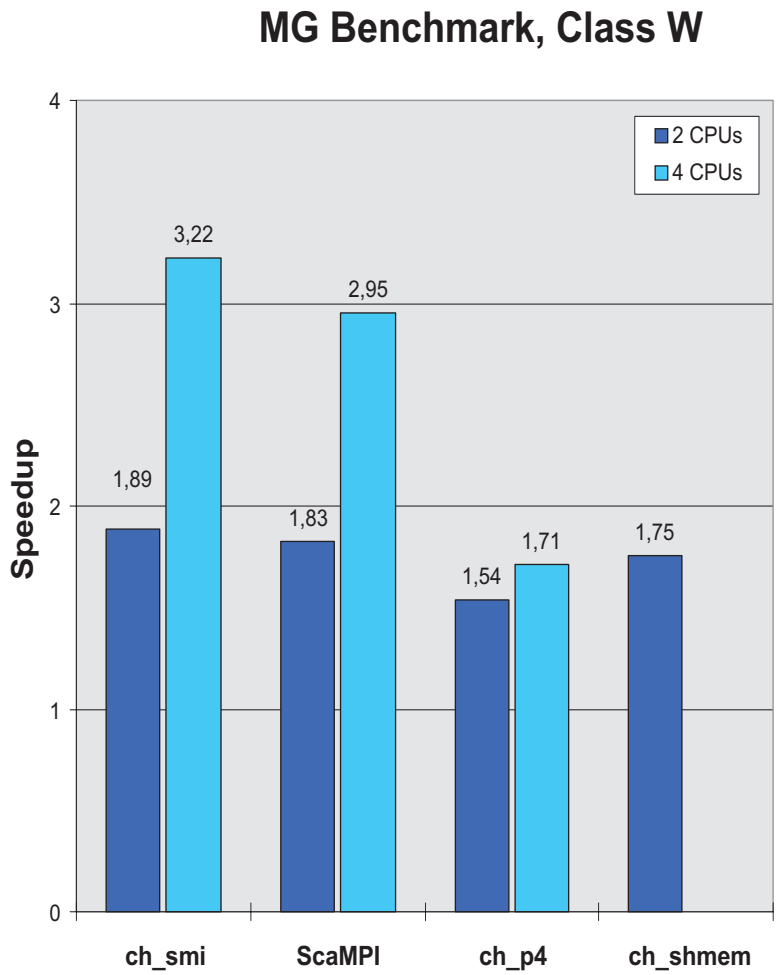
Comparison of **Speedup** for different MPICH communication devices / MPI implementations.

Compilers used with full optimization:

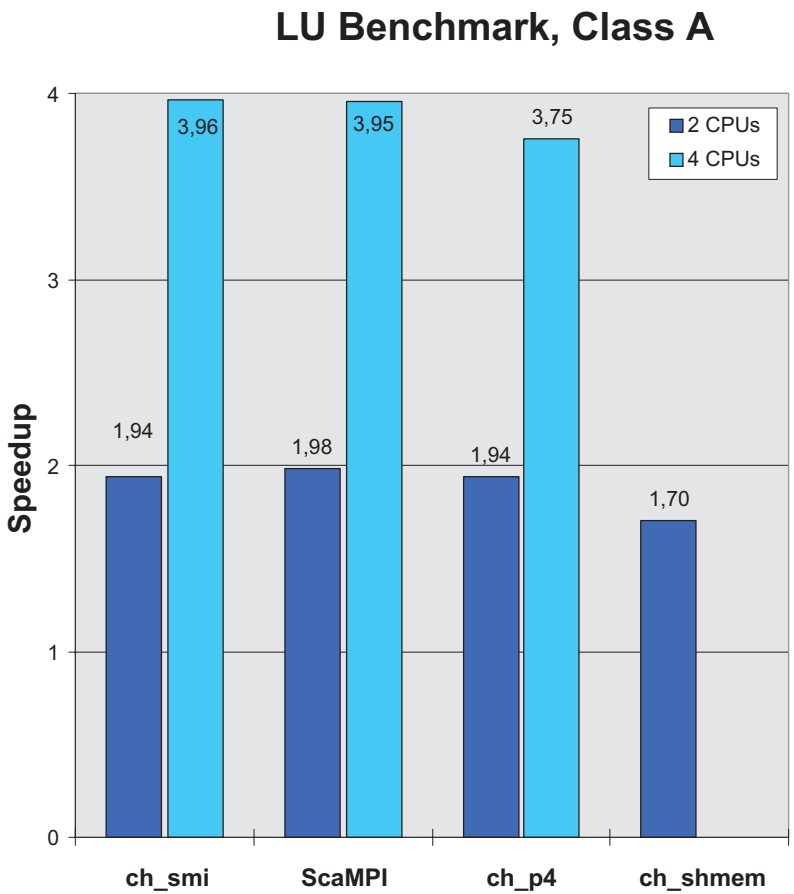
- Sun f77 5.0 for Solaris 7 (MPICH)
- Portland Group pgf90 3.0.4 for Linux (ScaMPI)

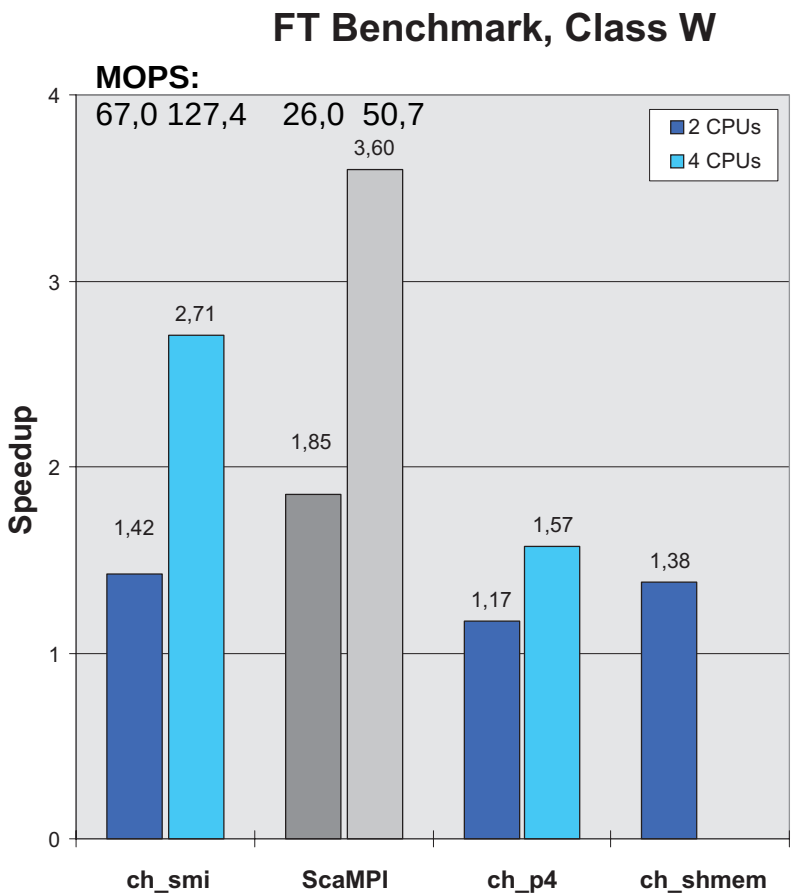
NPB: CG & MG


CG Benchmark, Class A
Speedup
2 CPUs
4 CPUs
0
1
2
3
4
1,88
3,54
1,91
3,45
1,66
2,45
1,62
ch_smi
ScaMPI
ch_p4
ch_shmem


MG Benchmark, Class W
Speedup
2 CPUs
4 CPUs
0
1
2
3
4
1,89
3,22
1,83
2,95
1,54
1,71
1,75
ch_smi
ScaMPI
ch_p4
ch_shmem

NPB: LU & FT


LU Benchmark, Class A
Speedup
0
1
2
3
4
2 CPUs
4 CPUs
1,94
3,96
1,98
3,95
1,94
3,75
1,70
ch_smi
ScaMPI
ch_p4
ch_shmem


FT Benchmark, Class W
MOPS:
67,0 127,4 26,0 50,7
Speedup
0
1
2
3
4
2 CPUs
4 CPUs
1,42
2,71
1,85
3,60
1,17
1,57
1,38
ch_smi
ScaMPI
ch_p4
ch_shmem

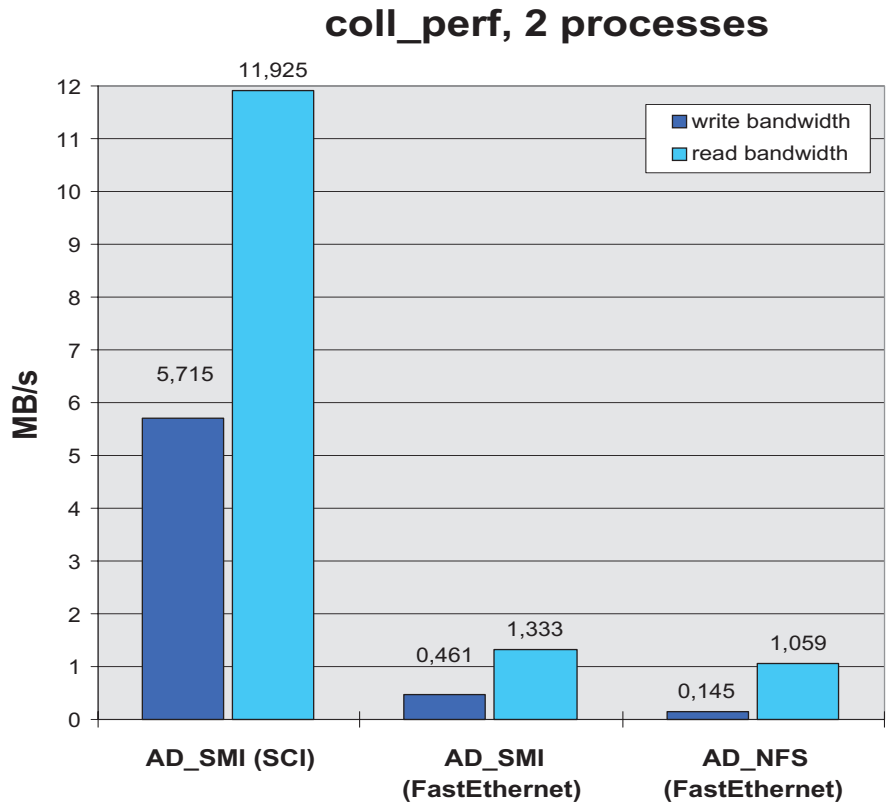
FURTHER DEVELOPMENT

- MPI-IO
  fast parallel I/O via SCI
- Adaptive Buffer Sizing
  optimize memory layout during execution
- Asynchronous Transfer
  utilize DMA, Threads & Interrupts for true asynchronous transfers
- Collective Operations
  replace Point-to-Point with shared-memory operations
- Cluster Manager
  Java-based, heterogenous execution environment

PARALLEL IO

Implementation of a SCI-based ADIO device for MPI-IO/ROMIO

- **prelimitary** results of the examples/io/coll_perf benchmark



UNIPROCESSOR VS. SMP

Platform: Windows NT on Dual-PentiumPro (200MHz, 256 MB RAM)

Testcase 1: NAS-Benchmark BT, Class W

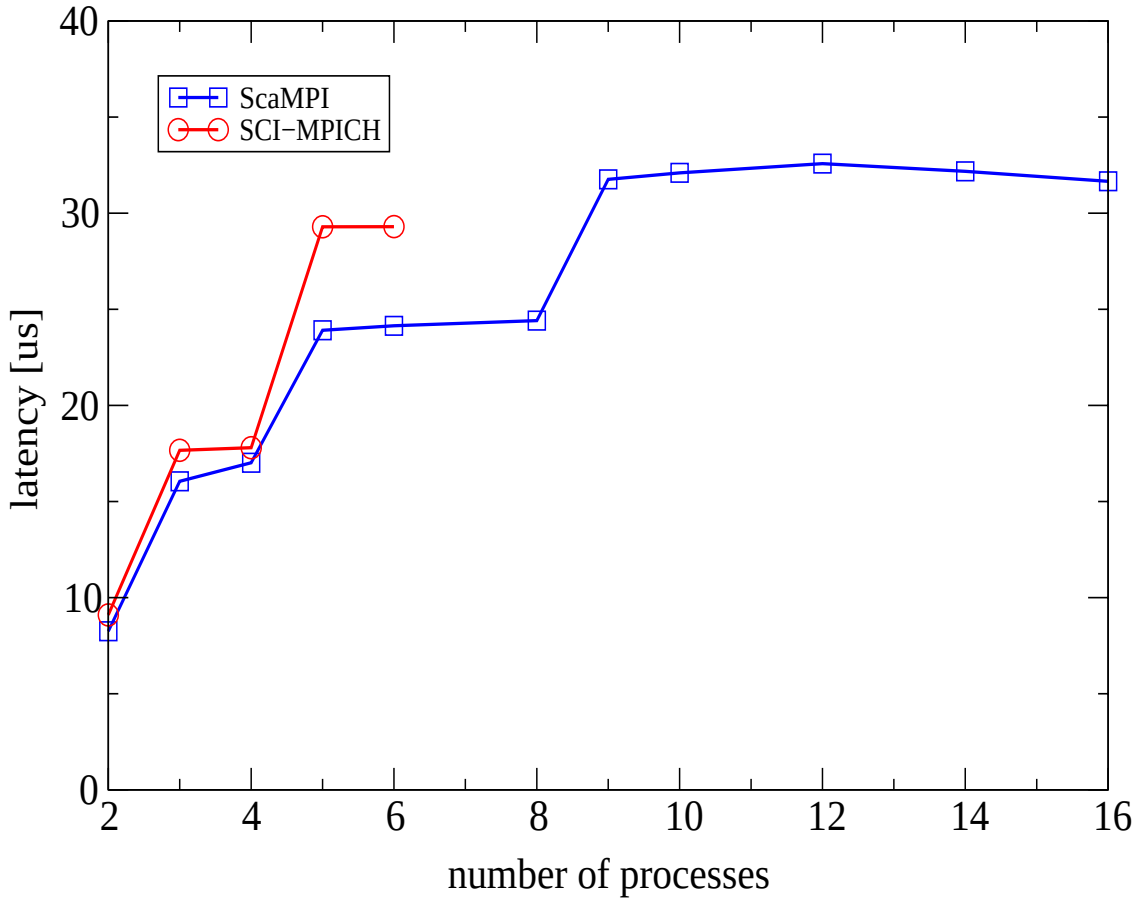
processes:	**1**	**4**	**9**
dedicated nodes	340,83 s	84,73 s (speedup **4,02**)	-
shared nodes	340,83 s	146,32 s (speedup **2,32**)	72,30 s (speedup **4,71**)

Testcase 2: NAS-Benchmark CG, Class A

processes:	**1**	**4**	**8**
dedicated nodes	127,08 s	32,27s (speedup **3,93**)	-
shared nodes	127,08 s	59,34 s (speedup **2,14**)	35,42 s (speedup **3,58**)

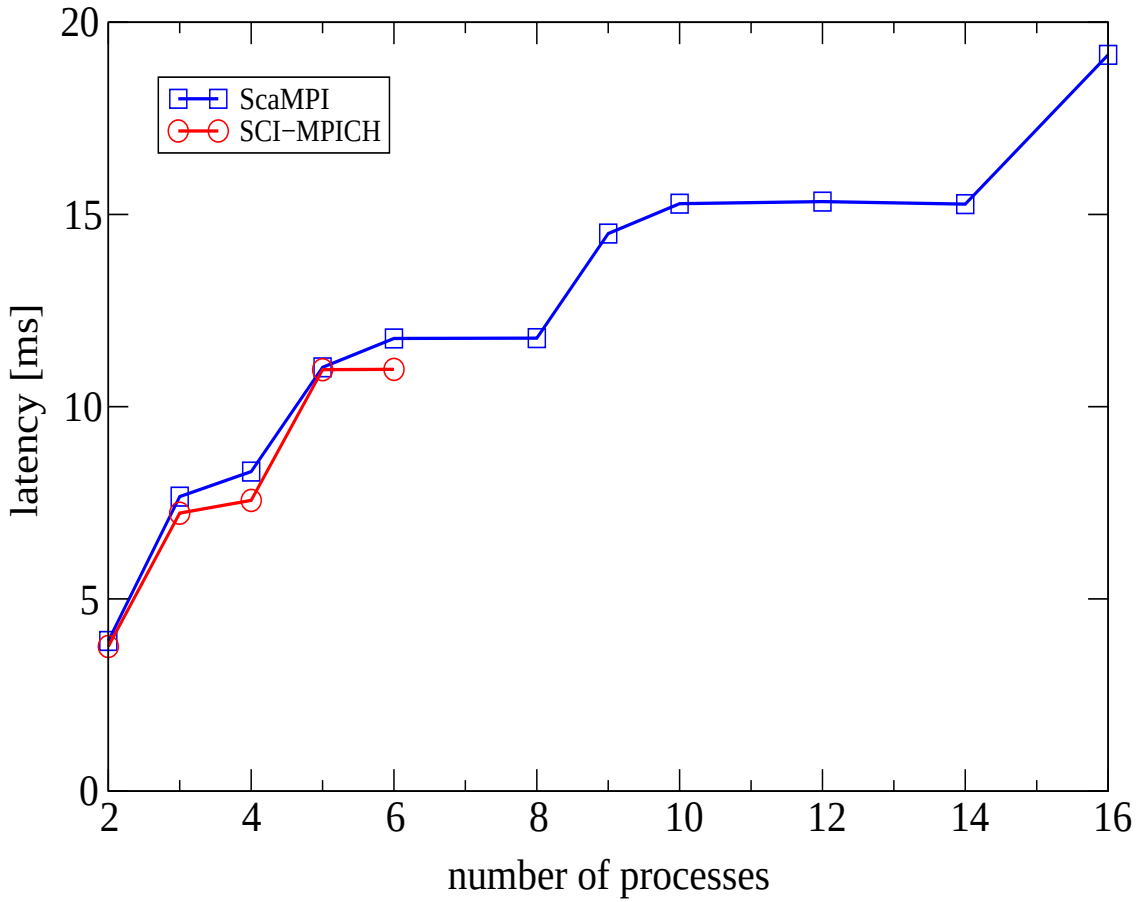
COLLECTIVE OPERATIONS

MPI_Bcast() for 8 Bytes



COLLECTIVE OPERATIONS

MPI_Bcast() for 256 kB



SUMMARY

SCI-MPICH delivers:

- High-performance, low-cost MPI platform for x86-based PC-Clusters with SCI interconnect
- Complete MPI 1.1 implementation (except MPI_Cancel)
- Available for Solaris x86, Linux and NT
- OpenSource for open development
- Significant performance gain against TCP/IP (88% better speedup for NAS Multigrid-Code)
- Similar performance to ScaMPI
- Many outstanding issues for further development
- Large scale testing to be done
- Homepage: `http://www.lfbs.rwth-aachen.de/~joachim/SCI-MPICH`