Buffered Coscheduled MPI: A New Approach in the System Software Design for Large-Scale Parallel Computers

Juan Fernández[1,2], Eitan Frachtenberg[1], Fabrizio Petrini[1]

[1]CCS-3 Modeling, Algorithms and Informatics
Computer and Computational Sciences (CCS) Division
Los Alamos National Laboratory, NM 87545, USA
URL: http://www.c3.lanl.gov

[2]Grupo de Arquitectura y Computación Paralelas (GACOP)
Dpto. Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30071 Murcia, SPAIN
URL: http://www.ditec.um.es

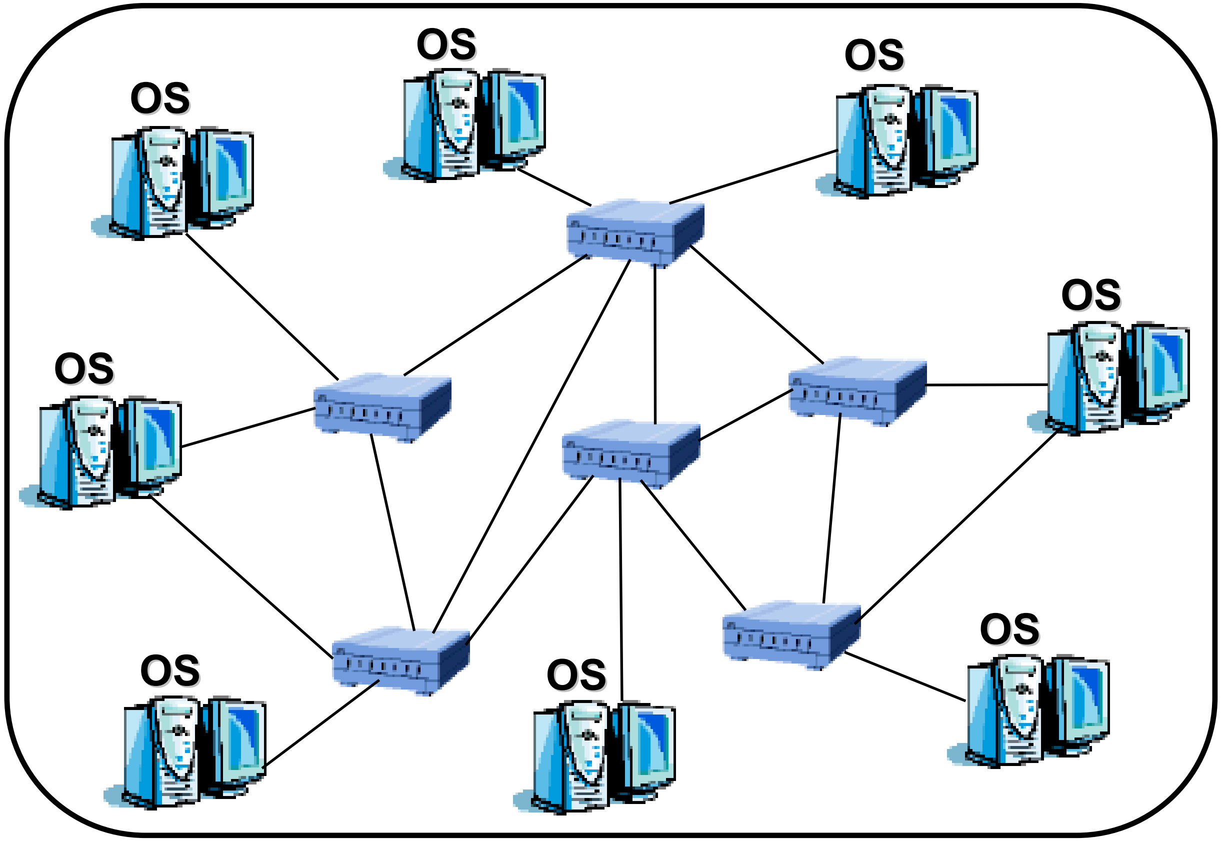
CCS

GACOP

email:{juanf,eitanf,fabrizio}@lanl.gov

IEEE/ACM Conference on SuperComputing - November 15-21, 2003 - Phoenix (AZ)

Motivation



Hardware / OSs are glued together by System Software:

Resource Management

Communications

Parallel Development and Debugging Tools

Parallel File System

Fault Tolerance

System software is a key factor to maximize usability, performance and scalability on large-scale systems!!!

CCS

GACOP

Motivation

- **System software complexity due to multiple factors:**
  - Extremely complex global state
    - Thousands of processes, threads, open files, pending messages, etc.
  - Non-deterministic behavior
    - Inherent to computing systems ⇨ OS process scheduling
    - Induced by parallel applications ⇨ MPI_ANY_SOURCE
  - Local OSs lack global awareness of parallel applications
    - Interferences with fine-grain synchronization operations ⇨ Non-scalable collective communication primitives.

COMPUTER & COMPUTATIONAL SCIENCES

CCS

GACOP

Motivation

- **System software complexity due to multiple factors:**
  - Independent **design** of different components
    - Redundancy of functionality ⇨ Communication protocols
    - Missing functionality ⇨ QoS user-level traffic / system-level traffic
  - User-level applications rely on system software
    - System software performance/scalability impacts user-application performance/scalability

CCS

GACOP

Goals

Target

- Simplifying design and implementation of the system software for large-scale computers
- Simplicity, performance, scalability

Approach

- Built atop a basic set of three primitives
- Global synchronization/scheduling

Vision

- SIMD system running MIMD applications (variable granularity in the order of hundreds of μs)

CCS

GACOP

Outline

- Motivation and Goals
- Introduction
- Core Primitives
- Design and Implementation
- Performance Evaluation
- Concluding remarks

CCS

GACOP

Introduction

- **In this paper we make our point by implementing the communication layer: BCS MPI**
  - Built atop the three core primitives we proposed as the basics for all system software components (STORM is a resource manager implemented atop the same primitives)
  - SIMD-like behavior: communications are synchronized and scheduled every few hundreds of microseconds
  - Implemented almost entirely in the Network Interface Card
  - Sacrifices neither performance nor scalability
  - Paves the way to provide improved system utilization, traffic segregation, deterministic replay of user applications and system-level fault tolerance

CCS

GACOP

Outline

- Motivation and Goals
- Introduction
- Core Primitives
- Design and Implementation
- Performance Evaluation
- Concluding remarks

CCS

GACOP

BCS Core Primitives

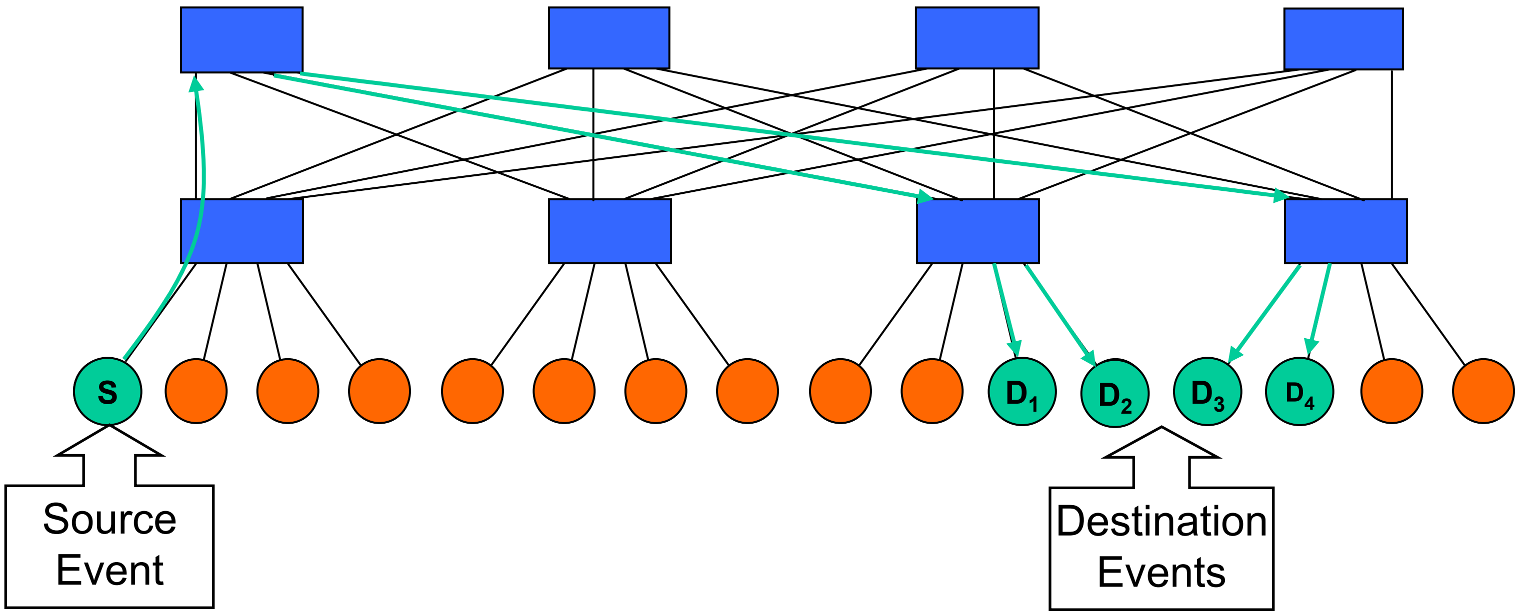
- **System software built atop three primitives**
  - Xfer-And-Signal
    - Transfer block of data to a set of nodes
    - Optionally signal local/remote event upon completion
  - Compare-And-Write
    - Compare global variable on a set of nodes
    - Optionally write global variable on the same set of nodes
  - Test-Event
    - Poll local event

CCS

GACOP

BCS Core Primitives

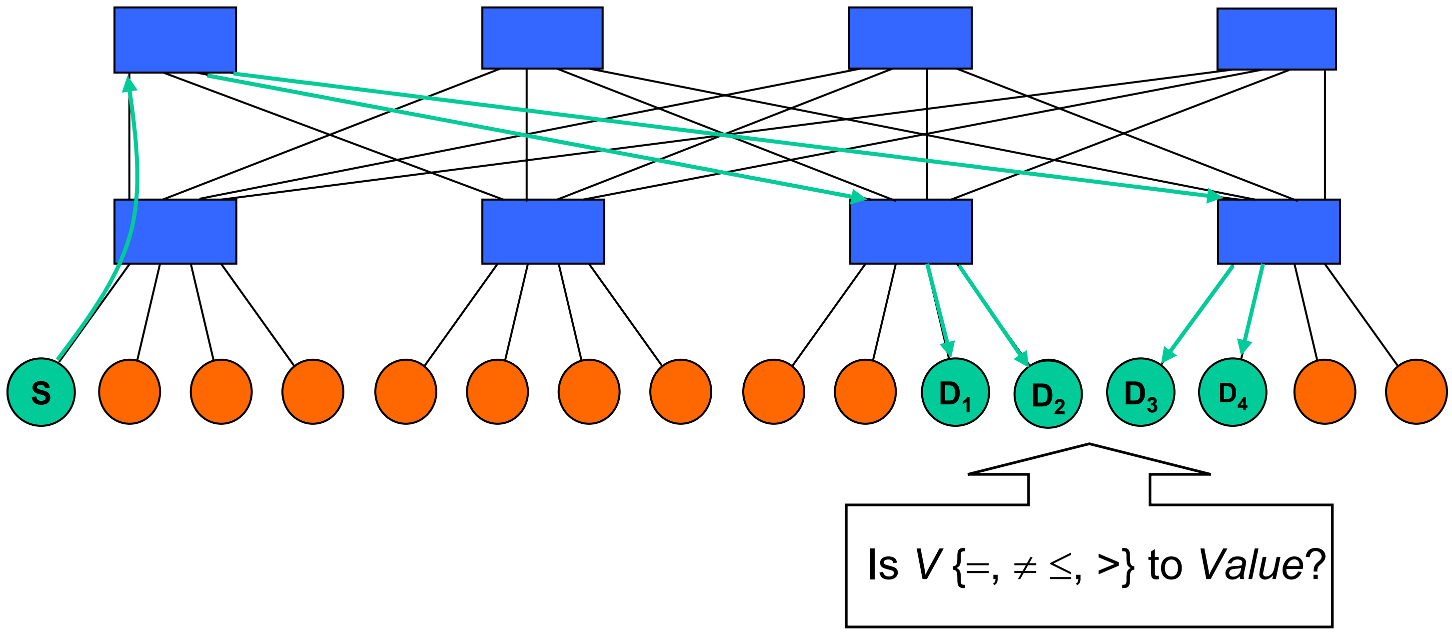
- **System software built atop three primitives**
  - Xfer-And-Signal (QsNet):
    - Node S transfers block of data to nodes D_1, D_2, D_3 and D_4
    - Events triggered at source and destinations



CCS
GACOP

BCS Core Primitives

- **System software built atop three primitives**
  - Compare-And-Write (QsNet):
    - Node S compares variable V on nodes D_1, D_2, D_3 and D_4

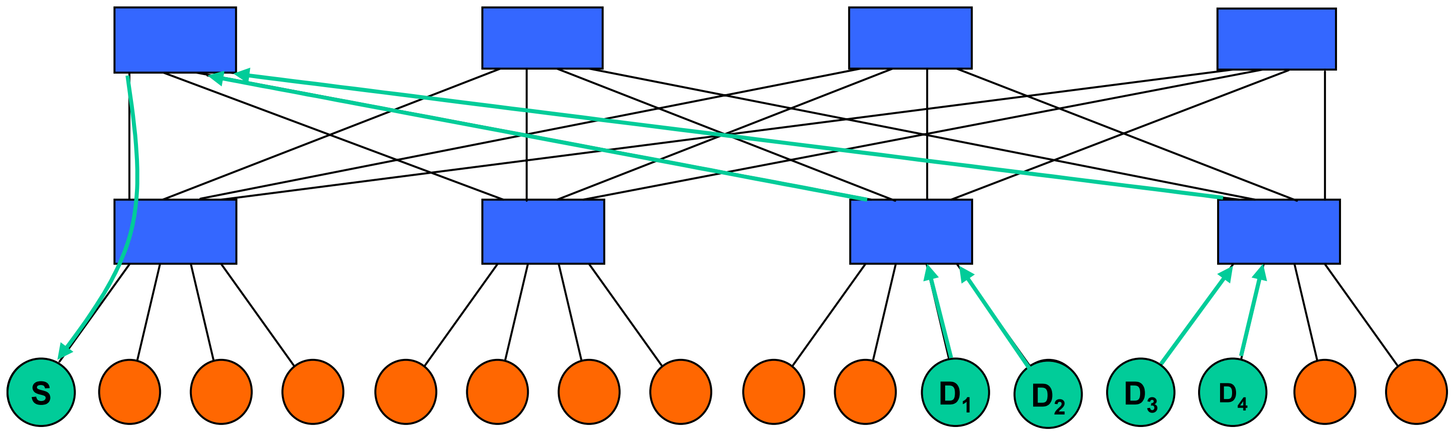


CCS

GACOP

BCS Core Primitives

- **System software built atop three primitives**
  - Compare-And-Write (QsNet):
    - Node S compares variable V on nodes D_1, D_2, D_3 and D_4
    - Partial results are combined in the switches



CCS

GACOP

Outline

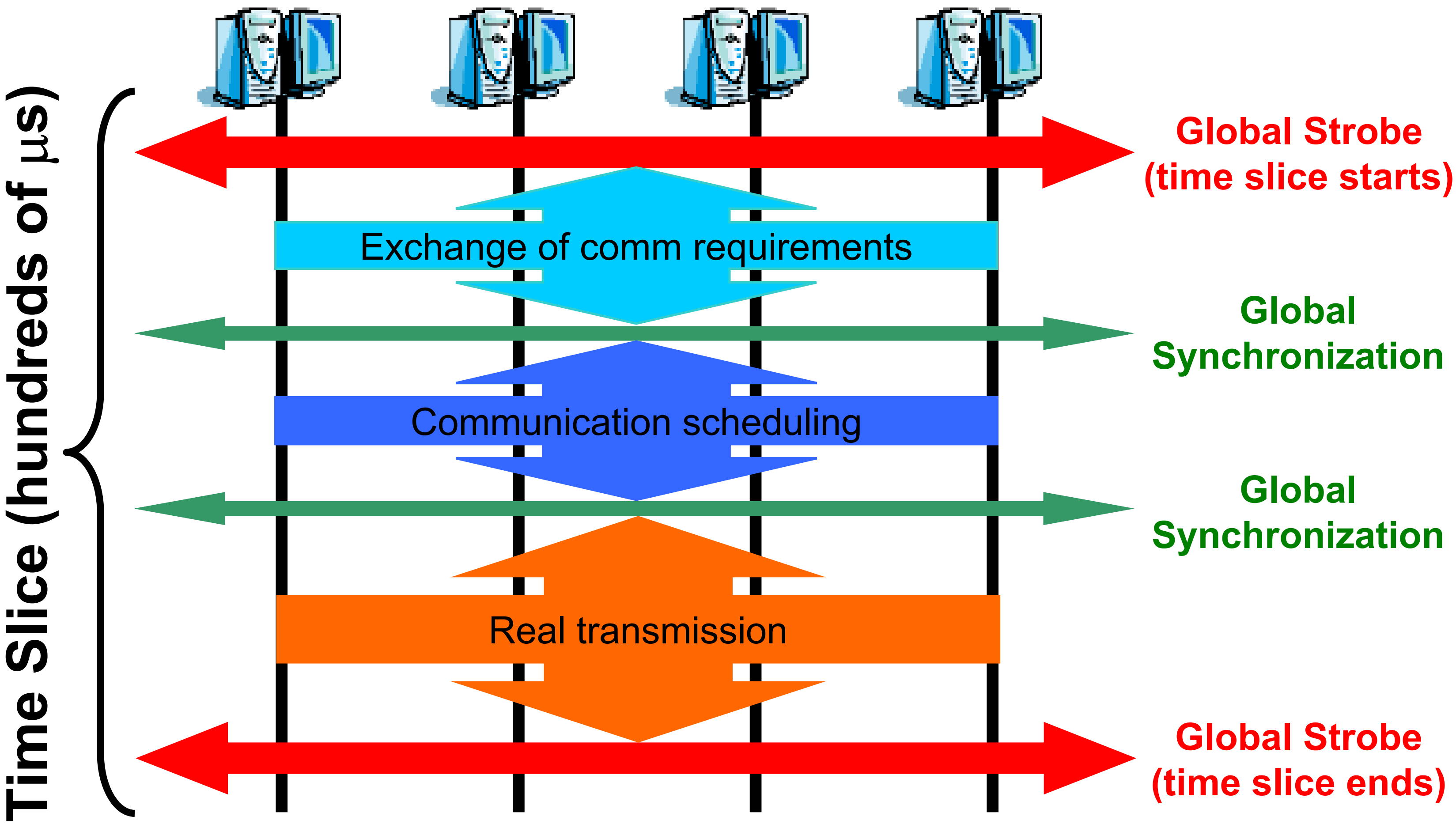
- Motivation and Goals
- Introduction
- Core Primitives
- Design and Implementation
- Performance Evaluation
- Concluding remarks

CCS

GACOP

Design and Implementation

- Real-time communication scheduling



CCS

GACOP

Design and Implementation

- **Global synchronization**
  - Strobe sent at regular intervals (time slices)
    - Compare-And-Write + Xfer-And-Signal (Master)
    - Test-Event (Slaves)
  - All system activities are tightly coupled
  - Global information is required to schedule resources, global synchronization facilitates the task but it is not enough
- **Global Scheduling**
  - Exchange of communication requirements
    - Xfer-And-Signal + Test-Event
  - Communication scheduling
  - Real transmission
    - Xfer-And-Signal + Test-Event

COMPUTER & COMPUTATIONAL SCIENCES
CCS
GACOP

Design and Implementation

- **Implementation in the *Network Interface Card***
  - Application processes interact with NIC threads
    - MPI primitive ⇨ Descriptor posted to the NIC
    - Communications are buffered
  - Cooperative threads running in the NIC
    - Synchronize
    - Partial exchange of control information
    - Schedule communications
    - Perform real transmissions and reduce computations
  - Computation/communication completely overlapped
    - Incoming messages do not generate interrupts
    - User processes do not need to poll for communication completion

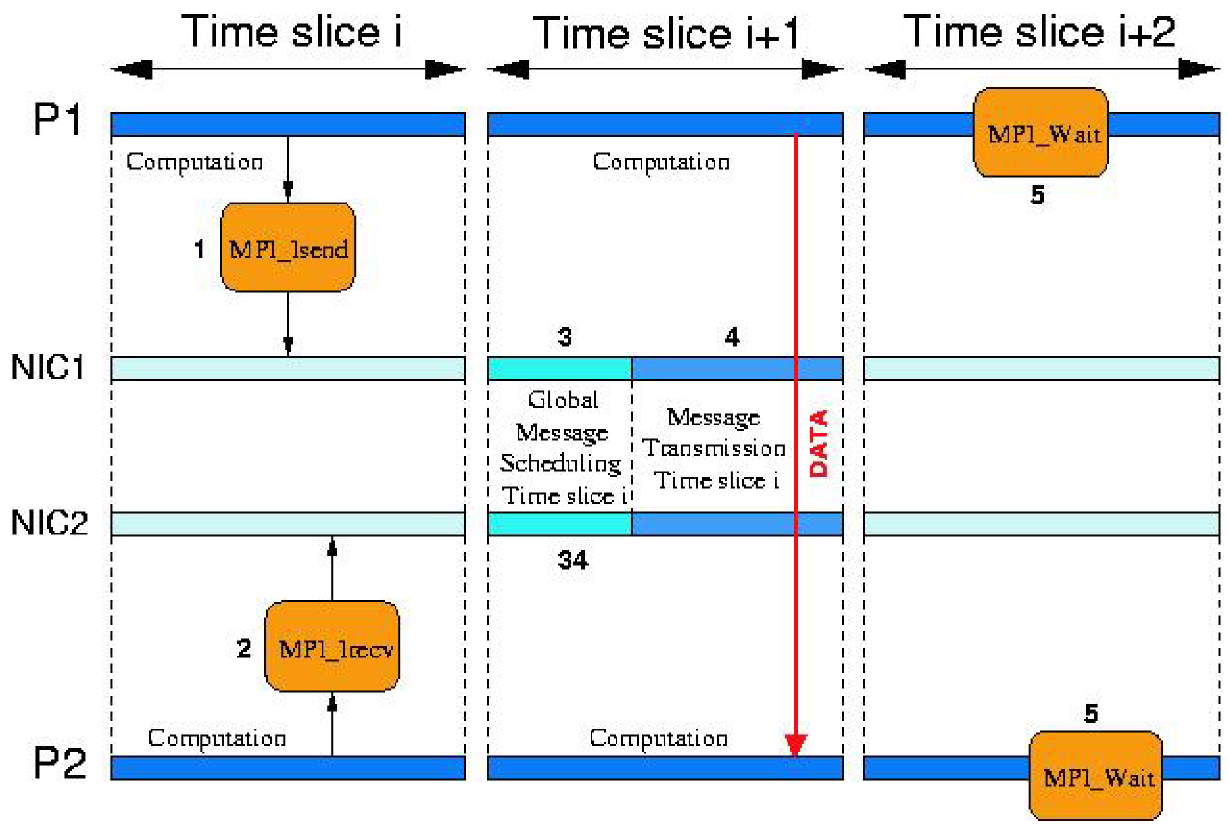
COMPUTER & COMPUTATIONAL SCIENCES

CCS

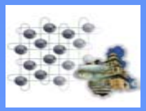
GACOP

Design and Implementation

- Non-blocking primitives: MPI_Isend/Irecv



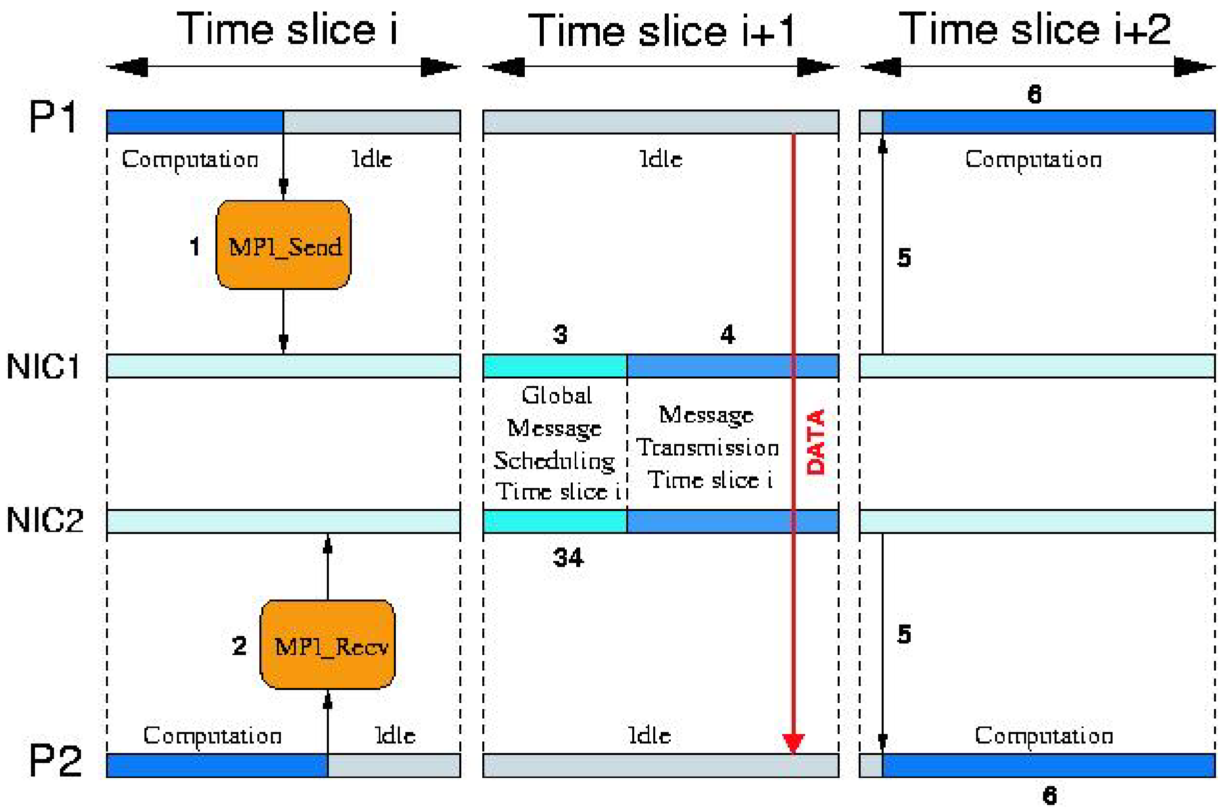
CCS



GACOP

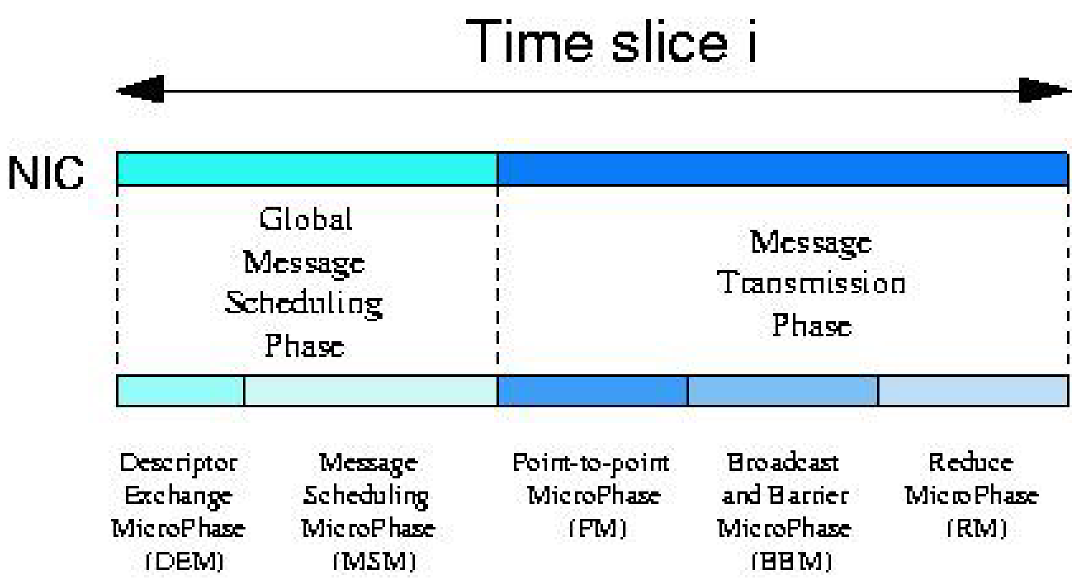
Design and Implementation

- **Blocking primitives: MPI_Send/Recv**



Design and Implementation

- **Global Synchronization/Scheduling Protocol**
  - Global Message Scheduling Phase
    - Microphases: Descriptor Exchange + Message Scheduling
  - Message Transmission Phase:
    - Microphases: Point-to-point, Barrier and Broadcast, Reduce



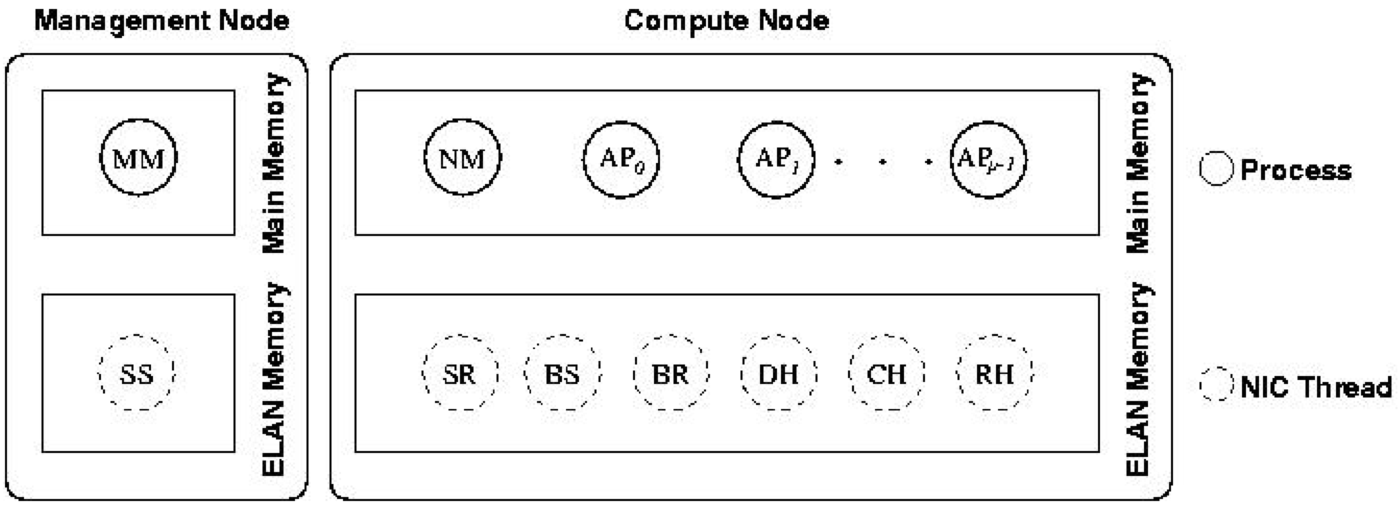
CCS

GACOP

Design and Implementation

Processes and Threads

- Synchronization: Strobe Sender + Strobe Receiver
- Scheduling: Buffer Sender, Buffer Receiver
- Transmission: DMA, Collective and Reduce Helpers



CCS

GACOP

Design and Implementation

- **Simplicity**
  - Hierarchical design atop a single kernel module which implements the three core primitives
- **Determinism**
  - Synchronization facilitates resource scheduling
  - Resource scheduling enforces reproducibility
- **Performance / Scalability**
  - Hardware-supported primitives
- **BCS MPI implementation**
  - QsNet hardware-supported multicast / events

CCS

GACOP

Outline

- Motivation and Goals
- Introduction
- Core Primitives
- Design and Implementation
- Performance Evaluation
- Concluding remarks

CCS

GACOP

Performance Evaluation

BCS MPI vs. Quadrics MPI

- Experimental Setup
  - *crescendo* cluster at LANL/CCS-3
    - 32 Dell 1550 compute nodes (two 1GHz P-III processors)
    - Quadrics QM-400 Elan3 NIC (1 RAIL)
    - RH 7.3 + Quadrics mods + qsnetlibs v1.5.0-0
    - Intel C/Fortran Compiler v5.0.1 (-O3)
  - *accelerando* cluster at LANL/CCS-3
    - 32 HP Server rx2600 (two 1GHz Itanium-II processors)
    - Two Quadrics QM-400 Elan3 NICs (2 RAILs)
    - RH 7.2 + Quadrics mods + qsnetlibs v1.5.0-0
    - Intel C/Fortran Compiler v7.1.17 (-O3)

CCS

GACOP

Performance Evaluation

BCS MPI vs. Quadrics MPI

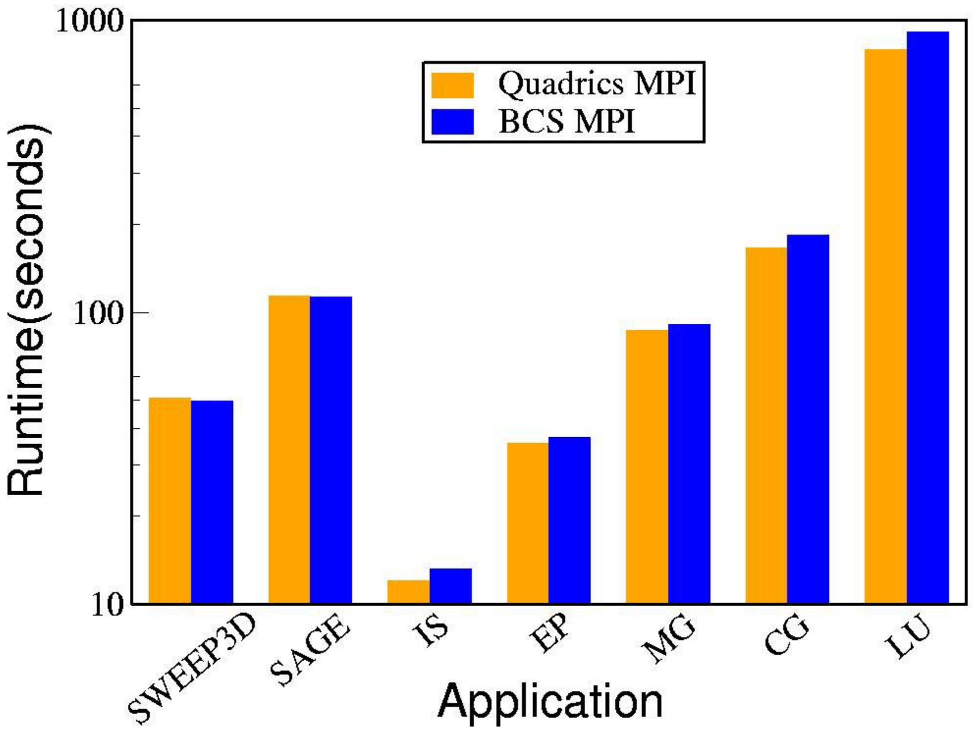
- Experimental Setup
  - **User-level** implementation of BCS MPI
    - Scheduling parameters
      - 500µs communication scheduling time slice (1 rail)
      - 250µs communication scheduling time slice (2 rails)
  - Benchmarks and Applications
    - NPB (IS,EP,MG,CG,LU) - Class C
    - SWEEP3D - 50x50x50
    - SAGE - timing.input

CCS

GACOP

Performance Evaluation

- Benchmarks and Applications (IA32)



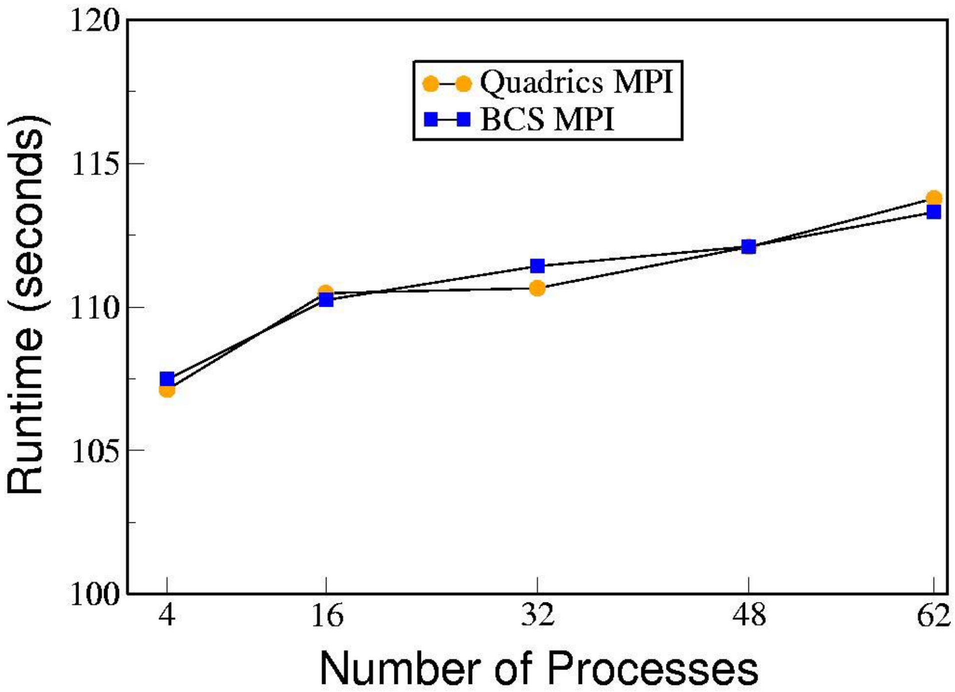
Application	Slowdown
IS (32PEs)	10.40%
EP (49PEs)	5.35%
MG (32PEs)	4.37%
CG (32PEs)	10.83%
LU (32PEs)	15.04%
SWEEP3D (49PEs)	-2.23%
SAGE (62PEs)	-0.42%

CCS

GACOP

Performance Evaluation

- **SAGE - timing.input (IA32)**



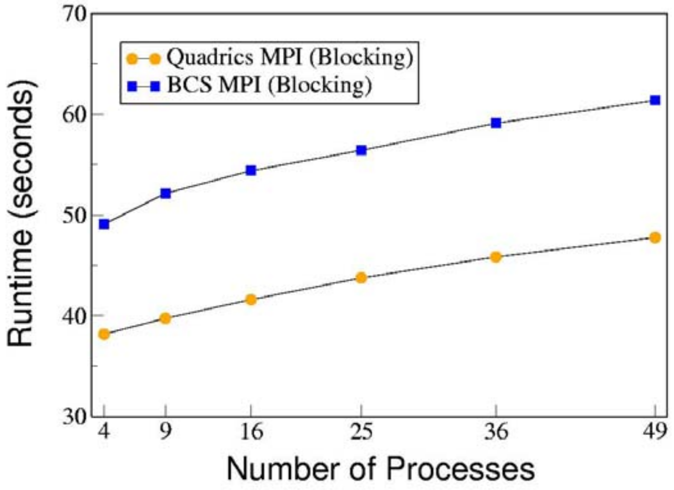
0.5% SPEEDUP

CCS

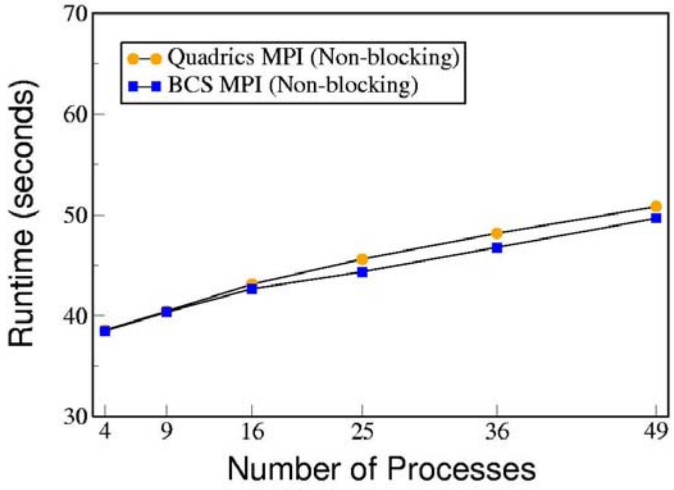
GACOP

Performance Evaluation

- **Blocking vs Non-blocking SWEEP3D (IA32)**
  - MPI_Send/Recv ⇨ MPI_Isend/Irecv + MPI_Waitall



30% SLOWDOWN



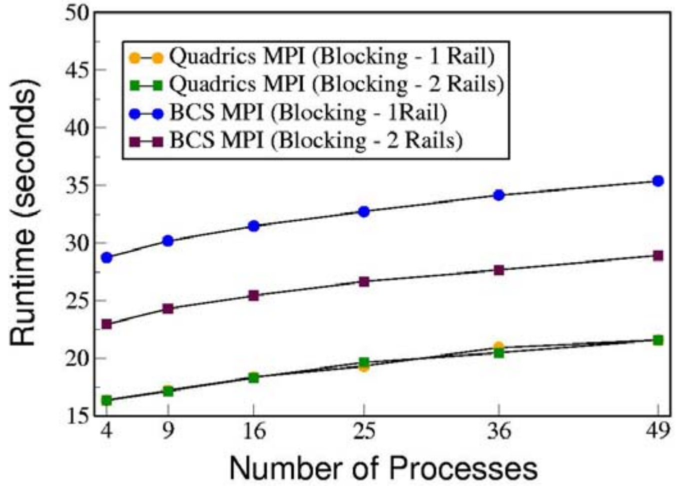
2% SPEEDUP

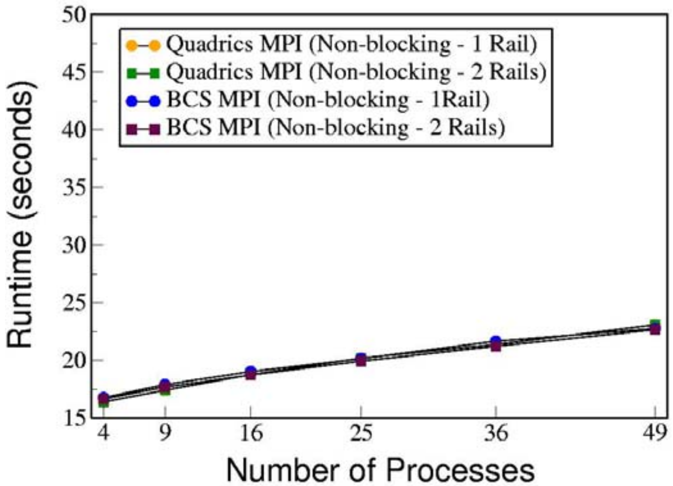
CCS

GACOP

Performance Evaluation

- **Blocking vs Non-blocking SWEEP3D (IA64)**
  - MPI_Send/Recv ⇨ MPI_Isend/Irecv + MPI_Waitall





64% SLOWDOWN 1 RAIL

34% SLOWDOWN 2 RAILS

1% SPEEDUP 1 RAIL

2% SPEEDUP 2 RAILS

CCS

GACOP

Performance Evaluation

BCS MPI:
Similar Performance with a Simplified Design

CCS

GACOP

Concluding Remarks

- **New approach to the design of system software**
  - Kernel module which implements the basics of most system software components (STORM) $\Rightarrow$ Simplicity
  - Hardware-supported primitives $\Rightarrow$ Performance / Scalability
- **Prototype implementation of BCS MPI**
  - Global synchronization of all system activities
  - Global scheduling of communications
  - Implemented almost entirely in the NIC
  - Global state optimization vs latency/bandwidth optimization
- **Promising preliminary results with real apps**
  - BCS MPI can compete with a production-level MPI

CCS

GACOP

Future Work

- **Kernel-level implementation of BCS-MPI**
  - User-level solution is already working with negligible performance degradation
- **Improved system utilization**
  - Scheduling multiple jobs
- **QoS for different types of traffic**
  - Scheduling messages may provide traffic segregation
- **Deterministic replay of MPI programs**
  - Ordered resource scheduling may enforce reproducibility
- **Transparent fault tolerance**
  - BCS MPI simplifies the state of the machine

CCS

GACOP

Buffered Coscheduled MPI: A New Approach in the System Software Design for Large-Scale Parallel Computers

Juan Fernández[1,2], Eitan Frachtenberg[1], Fabrizio Petrini[1]

[1]CCS-3 Modeling, Algorithms and Informatics
Computer and Computational Sciences (CCS) Division
Los Alamos National Laboratory, NM 87545, USA
URL: http://www.c3.lanl.gov

[2]Grupo de Arquitectura y Computación Paralelas (GACOP)
Dpto. Ingeniería y Tecnología de Computadores
Universidad de Murcia, 30071 Murcia, SPAIN
URL: http://www.ditec.um.es

email:{juanf,eitanf,fabrizio}@lanl.gov

CCS

GACOP