Optimizing Mechanisms for Latency Tolerance in Remote Memory Access Communication on Clusters

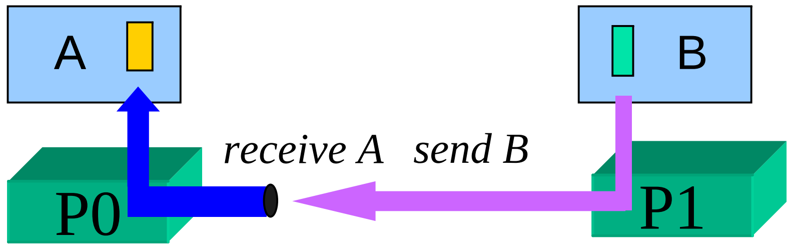
Jarek Nieplocha
Vinod Tipparaju
Manoj Krishnan
Pacific Northwest National Laboratory

Gopalakrishnan Santhanaraman
Dhabaleswar Panda
Ohio State University

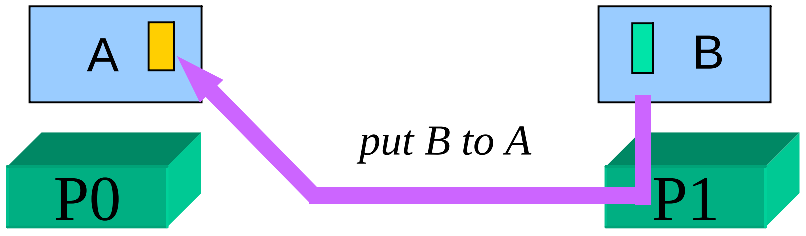
Overview

- Background & Motivation
- Implementation of Mechanisms for Latency Tolerance
  - Nonblocking operations
  - Aggregation of small messages
- Performance Studies
  - Microbenchmarks
  - Application kernels
- Conclusions

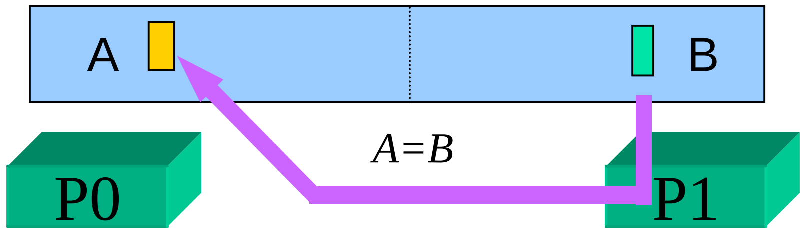
Remote Memory Access and other Communication Models



message passing



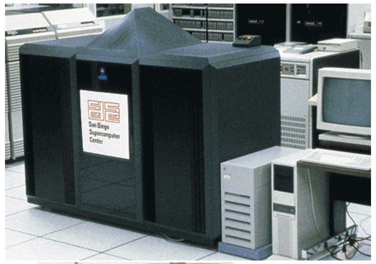
remote memory access (RMA)

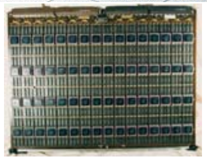


shared memory load/stores

Motivation: Processor-Network Speed Gap

1990

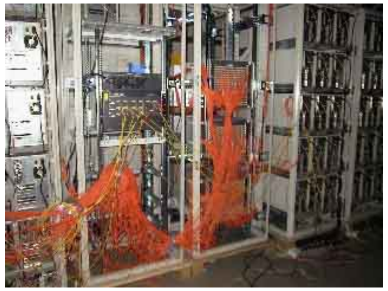




64-CPU board

NCUBE/2
2MFLOP/s CPU
100µs latency
(35 µs AM layer)
2.5MB/s B/W

Today



Itanium-2/Myrinet cluster
6GFLOP/s CPU
10 µs latency, 230MB/s (MPI)

Cray X1
12.8GFLOP/s CPU
9 µs latency 10GB/s (MPI)

Improvement over 12 years

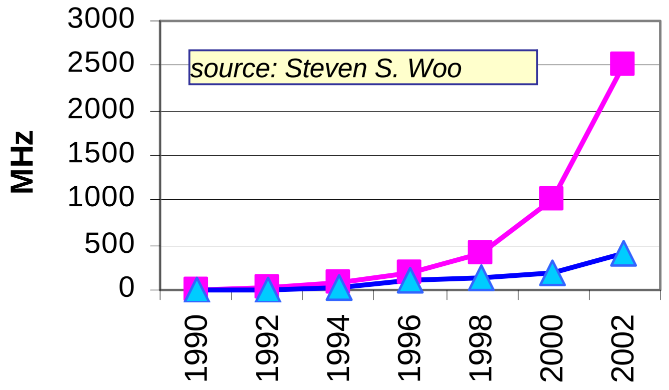
- CPU speed 10^4
- B/W 10^{2-4}
- Latency 10^1

$\Rightarrow$

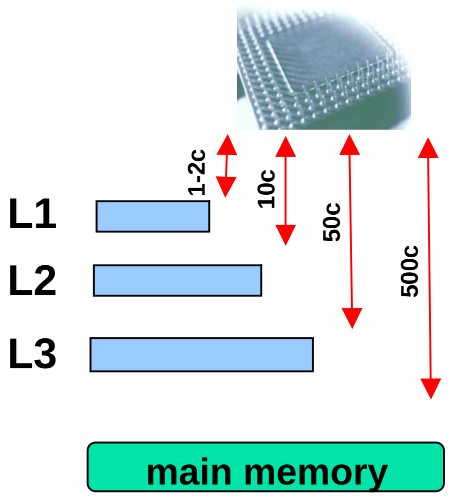
What it means for efficient communication?

- Application level
  - increase problem granularity to scale
  - avoid small messages
  - explore algorithmic opportunities for latency hiding
- Comm. system level
  - effective nonblocking communication
  - message aggregation

Motivation: Processor-Memory Speed Gap



W. Wulf S. McKee, “Hitting the memory wall: Implications of the obvious”, ACM Computer Architecture News, 1995



What it implies for efficient communication?

- Avoid processor touching data unnecessarily e.g.:
  - copying between user and special (DMA) buffers
  - data packing (sparse, strided)
  - format conversions

- Instead, we want
  - zero-copy communication
  - offload all/most data processing to the network adapter
    - including noncontiguous data
    - simplicity of RMA model attractive

Aggregate Remote Memory Copy Interface (ARMCI)

- Remote Memory Access Functionality
  - *put*, *get*, *accumulate* (also handles noncontiguous data transfers)
  - nonblocking communication interfaces
  - atomic *read-modify-write*, *mutexes* and *locks*
  - *memory allocation*
  - *fence* operations for managing memory consistency
- Characteristics
  - simpler & less synchronous model of RMA than MPI-2 1-sided
  - widespread portability (uses native protocols)
  - works with MPI
- Run-time component of the DoE Programming Models project
  - Co-Array Fortran compiler (Rice)
  - portable SHMEM (Ames lab)
  - Global Arrays library (PNNL)
  - Applications e.g., MPQC (Sandia)

Memory Management Challenge

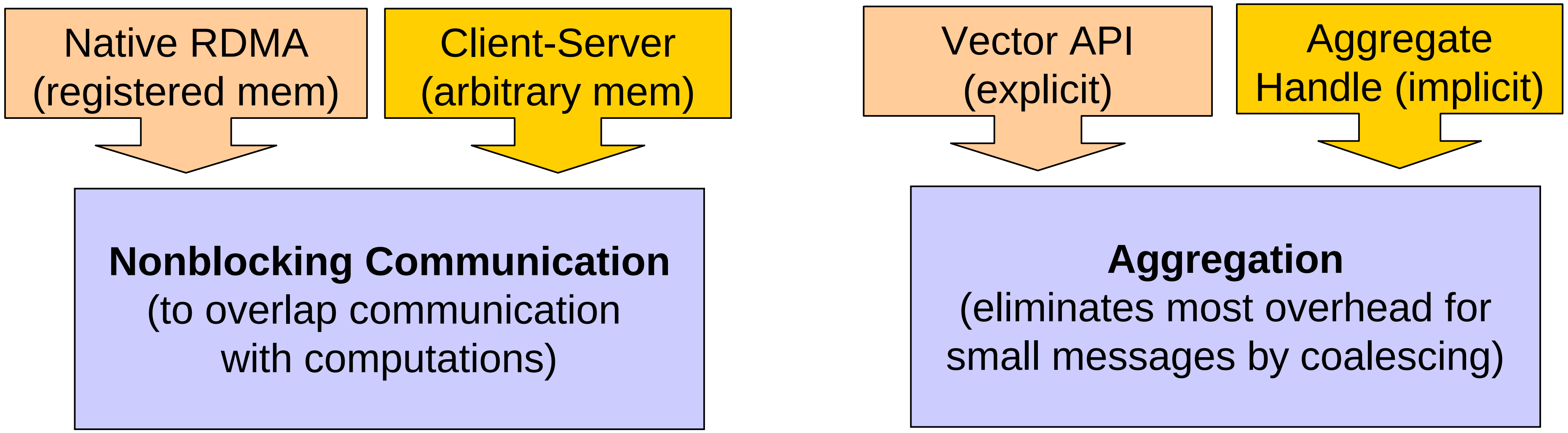
- User Communication Buffers
  - On systems with virtual memory, type of memory used for application communication buffers is critical to performance
    - *Good memory* (registered/pinned) that network adapter can communicate to/from directly (Myrinet, Infiniband, VIA, Hitachi SR8K)
    - *Good memory* on SMP nodes is shared memory (shmget/mmap)
  - Communication libraries designed to handle worst case scenario
    - on demand memory registration or data copying
    - this does not apply to Cray X1 and Quadrics (Elan-3,Elan-4)

- Our approach to memory management
  - Try to give user “good memory” suitable for fast communication
    - ARMCI_Malloc (registered + shared memory)
    - ARMCI_Malloc_local (registered)
  - If not, handle normal memory with some performance degradation
    - Want to provide effective nonblocking operations for both!

Two Mechanisms for Latency Tolerance

We implement portably two well known ideas:

- nonblocking coms: *Strumpen & Casavant 1994, Culler et al 1993*
- aggregation: *Pham & Albrecht 1999*



Native RDMA (registered mem)
Client-Server (arbitrary mem)
Vector API (explicit)
Aggregate Handle (implicit)
Nonblocking Communication (to overlap communication with computations)
Aggregation (eliminates most overhead for small messages by coalescing)

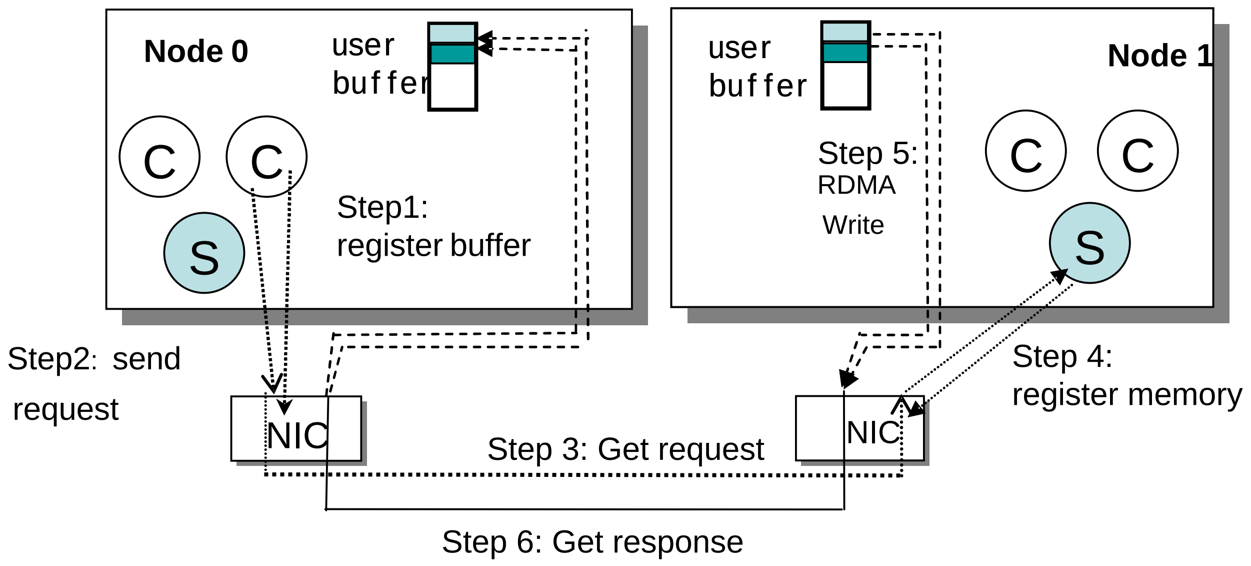
Details in the Paper:
descriptor and buffer management layers

Design Approach for Nonblocking Operations

- Aim for widespread portability
  - Implemented over GM, VIA, IBA, LAPI
    - Offer RMA (mostly in limited scope)
    - Papers on blocking RMA in CAC'02 and Cluster Computing 6, 2003
- Use multi-protocols to deliver best performance possible
  - switching between shared memory, zero-copy, buffered with and w/o pipelining
  - exploit and augment native vendor protocols
    - e.g., some not do well within the SMP node or miss noncontiguous data interfaces
- Manage multi-protocols transparently to the user
  - Track registered memory buffers
  - Request handle stores info on protocol used
- Pending requests
  - Either *active* or *completed* state
  - Older requests that use buffered protocols buffers might get internally completed in order to initiate a new request
  - A nonblocking call does not return w/o initiating network communication
    - Rather to be queued for future processing
    - Tries to provide effective overlapping

Example of Multiprotocols

- Get operation using dynamic memory registration
  - Server thread
  - RDMA put (write)
- Applicability
  - User memory not registered
    - For small messages it is less competitive than buffered protocols (registration cost)
  - Memory registered but network does not offer RDMA Read (GM 1.X, cLAN VIA)

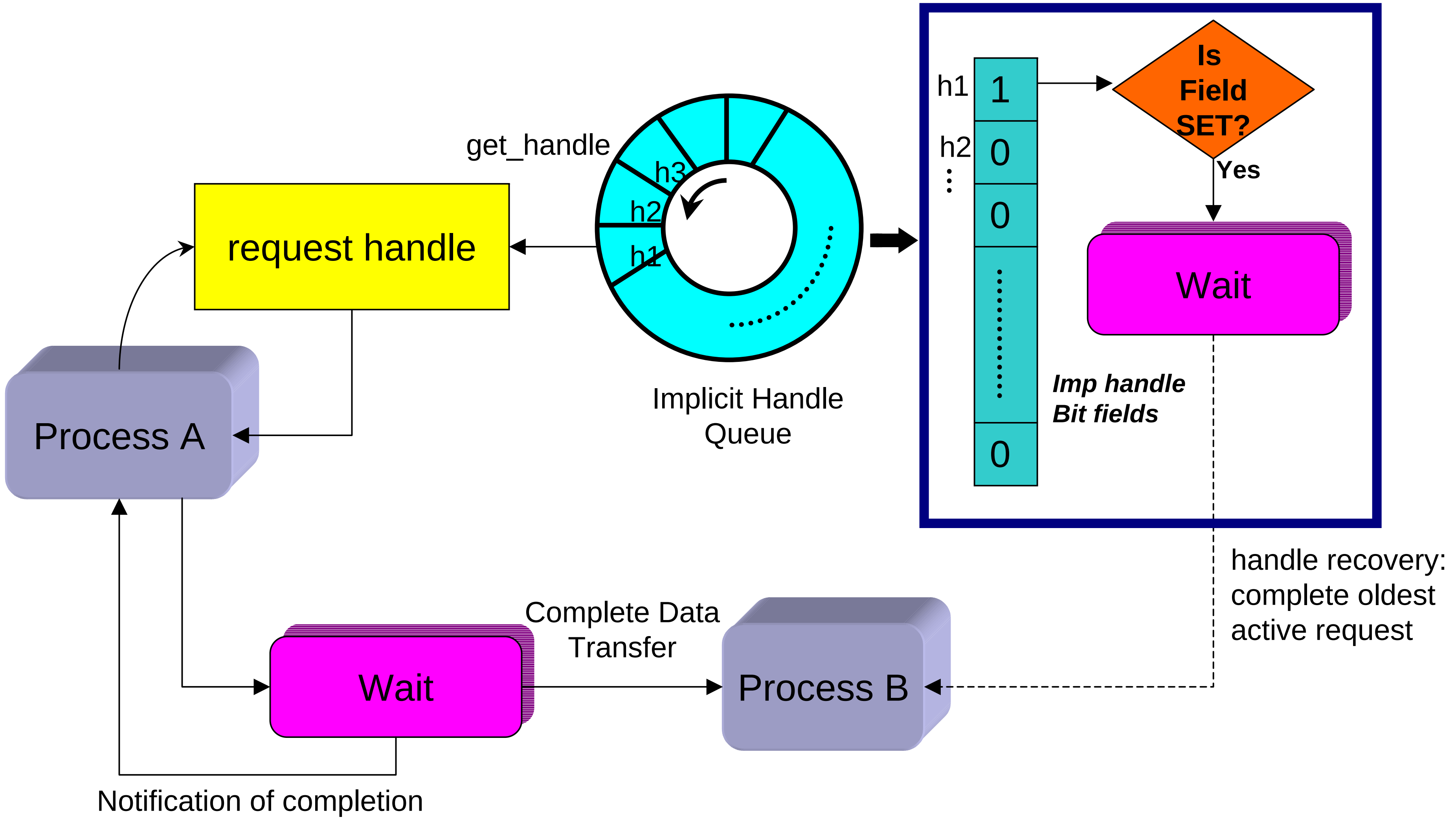


Request Handles

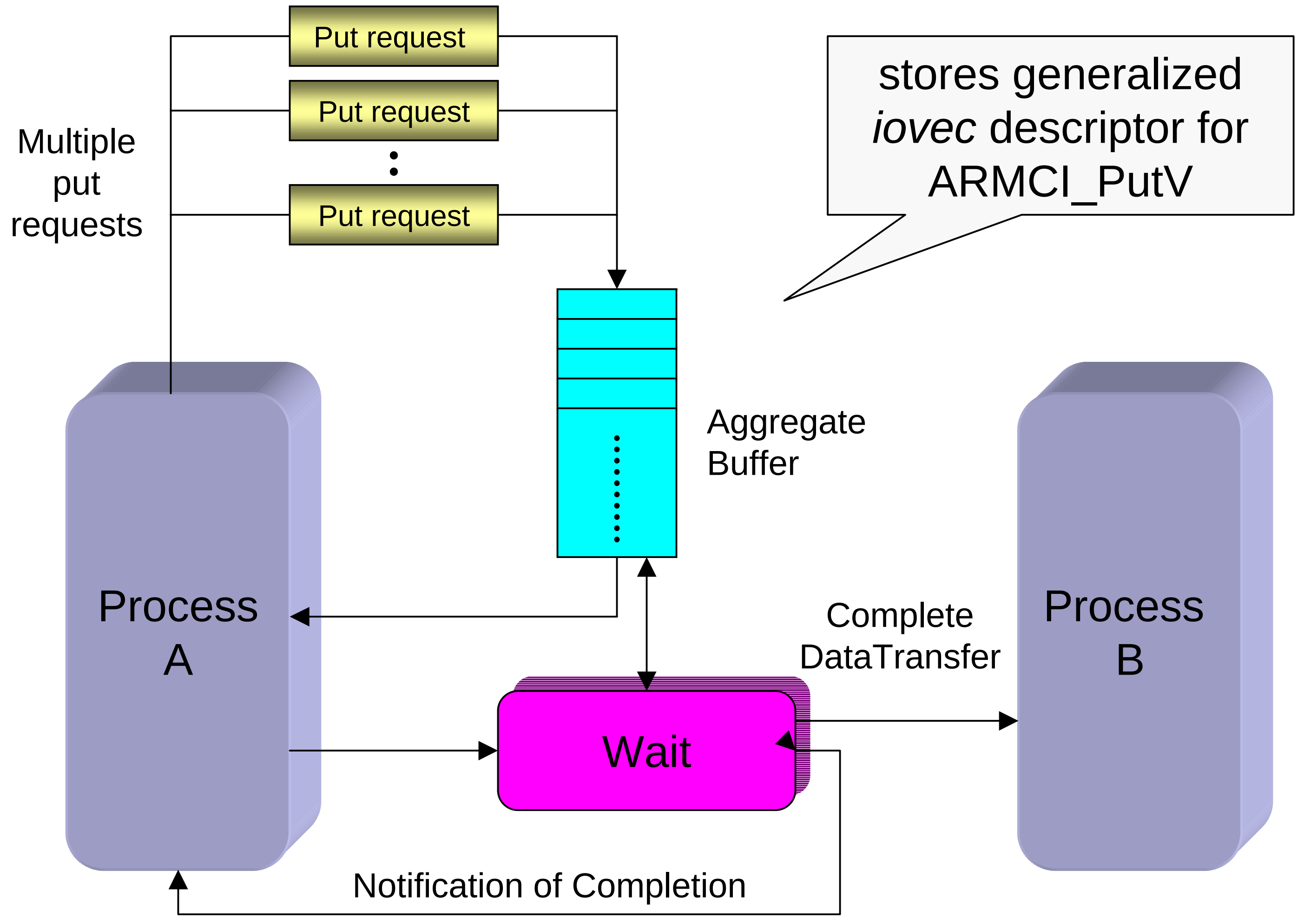
int **ARMCI_NbPut**(void* src, void *dst, int bytes,int proc, armci_hdl_t* handle)

- explicit handle
  - Used to wait for completion
  - Opaque data structure
    - RDMA descriptors
    - Buffer(s) info
    - other
  - User initialized
  - Can be used to aggregate messages
    - ARMCI_SET_AGGEGATE macro
- implicit handle
  - `handle=NULL`
  - convenience to the user
    - No need to manage handles

Implicit Handle Non-blocking Transfer



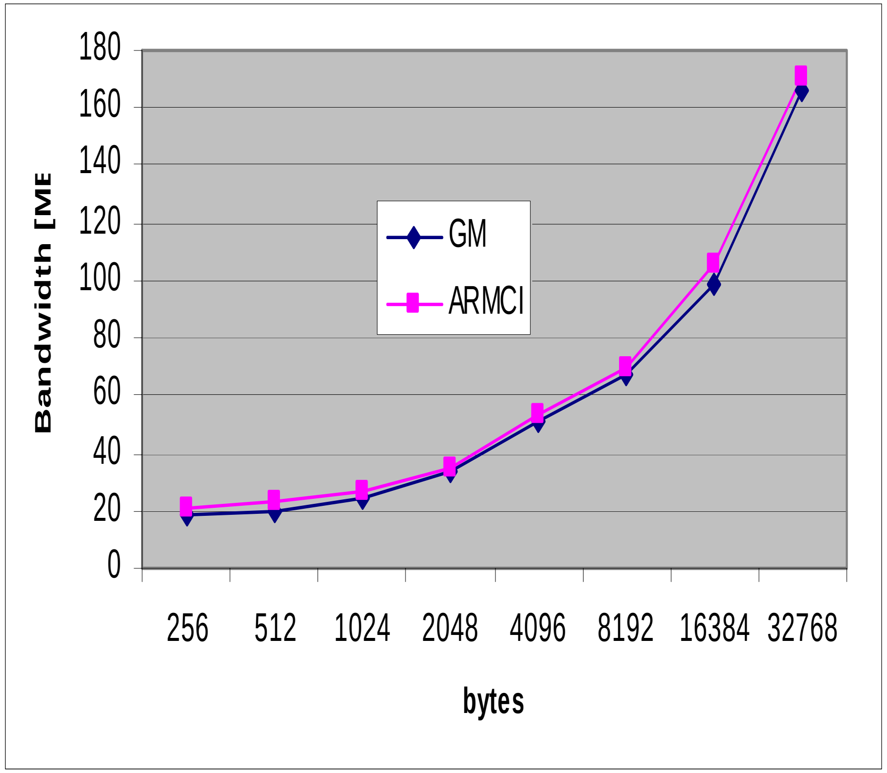
Message Aggregation: Put Example



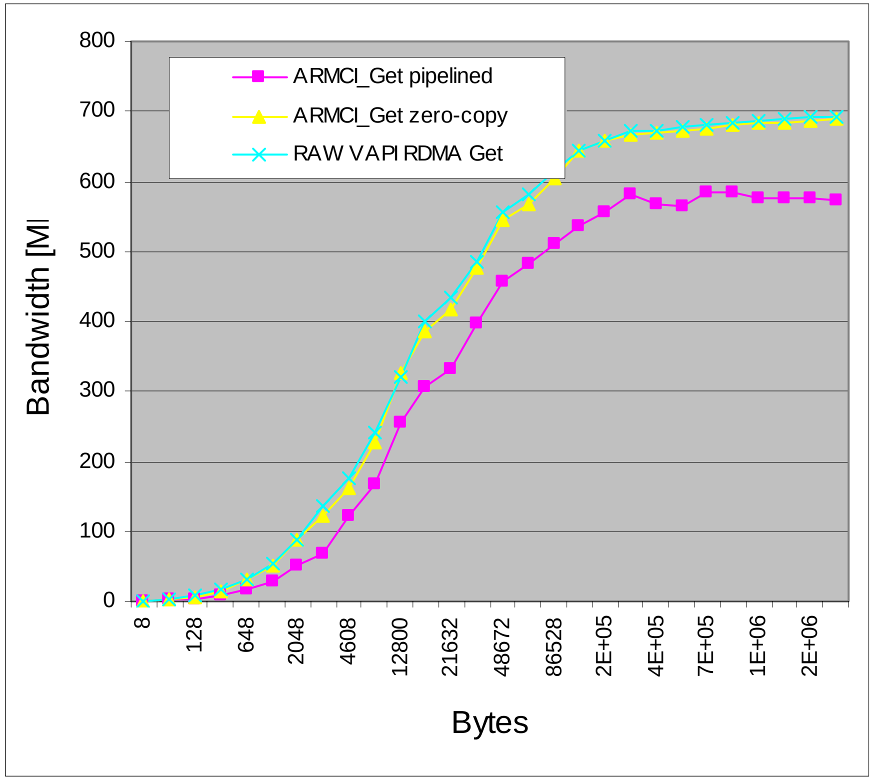
Experimental Results

- Platforms
  - Primary: dual CPU 2.4GHz P4 with Myrinet 2000
  - Secondary: Dual 1GHz Itanium-2 with Mallanox 4X IBA HCA
- Microbenchmarks
  - ARMCI overhead over native RDMA calls
    - Registered memory
    - nonregistered memory (pipelined protocols, see CAC'02 paper)
  - Potential for latency hiding in nonblocking operations
- Application Kernel Results
  - NAS CG & MG
  - Dense Matrix multiplication
  - Sparse Matrix-Vector multiplication

GET Overhead over Native RDMA Read



Myrinet/Pentium-4 GM 1.6

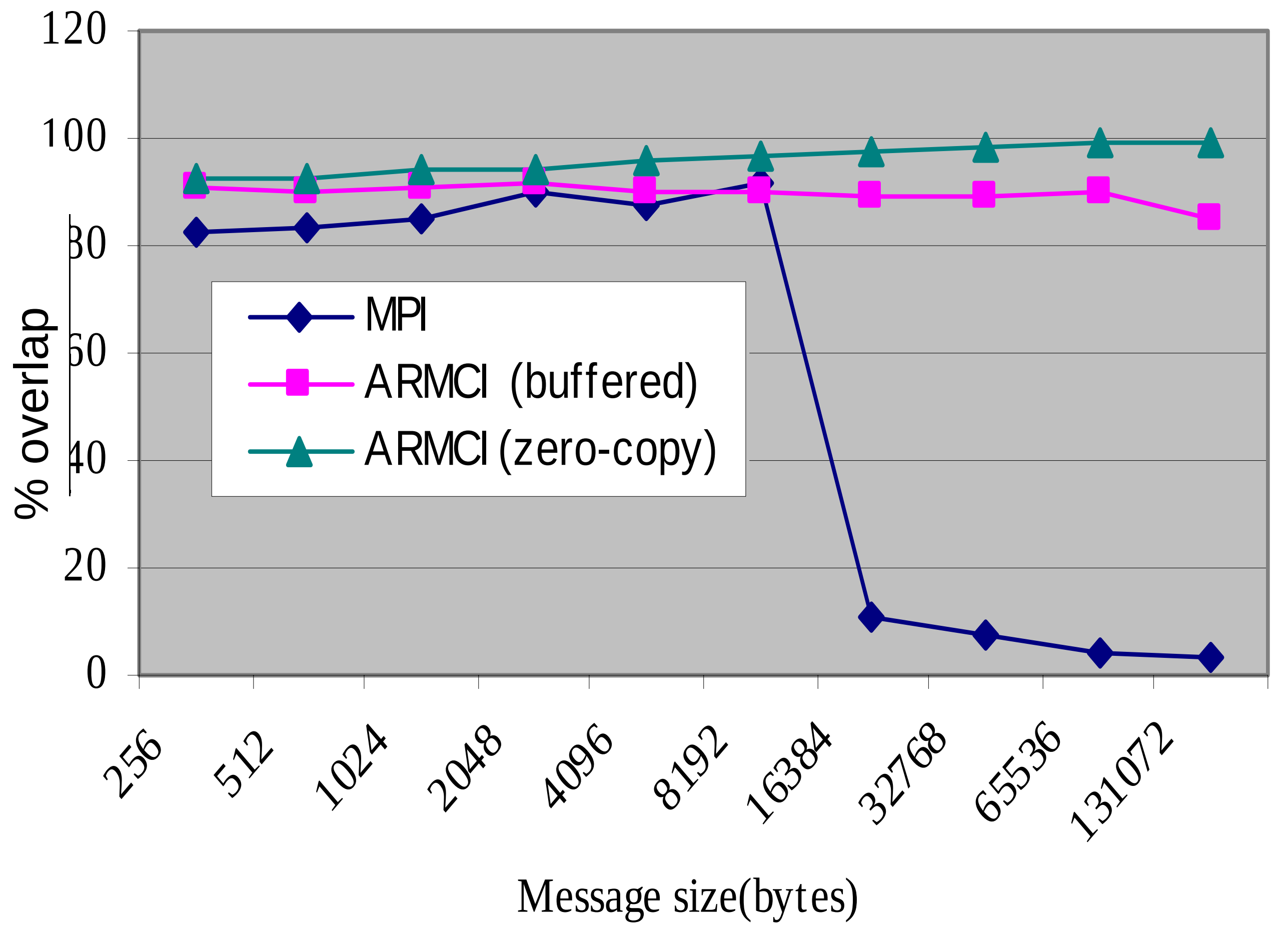


Infiniband/Itanium-2

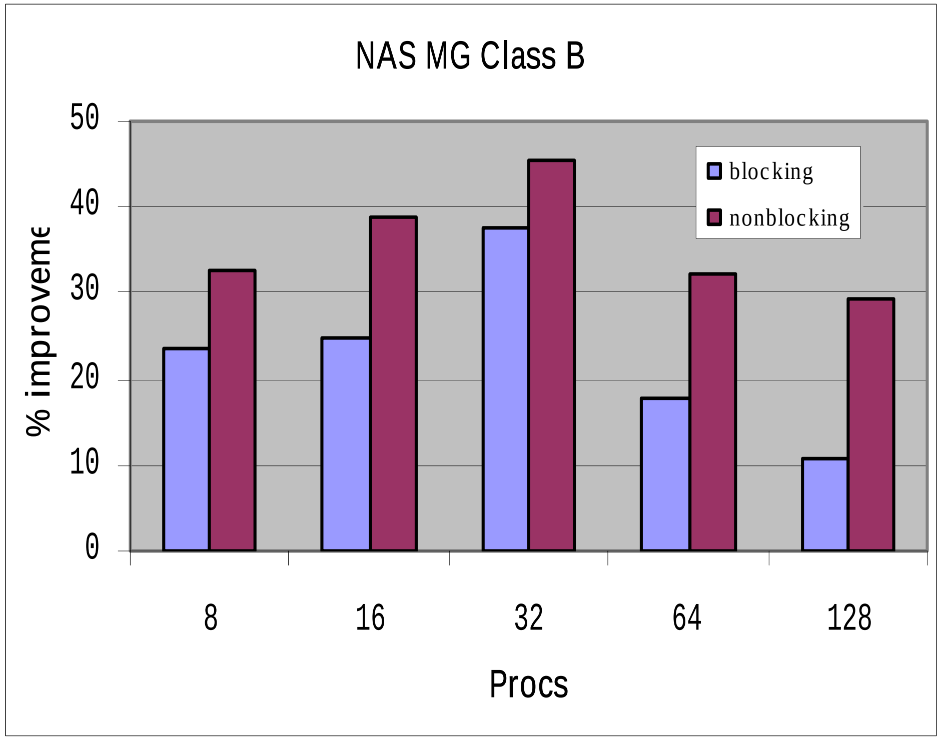
Overlapping Communication with Computations

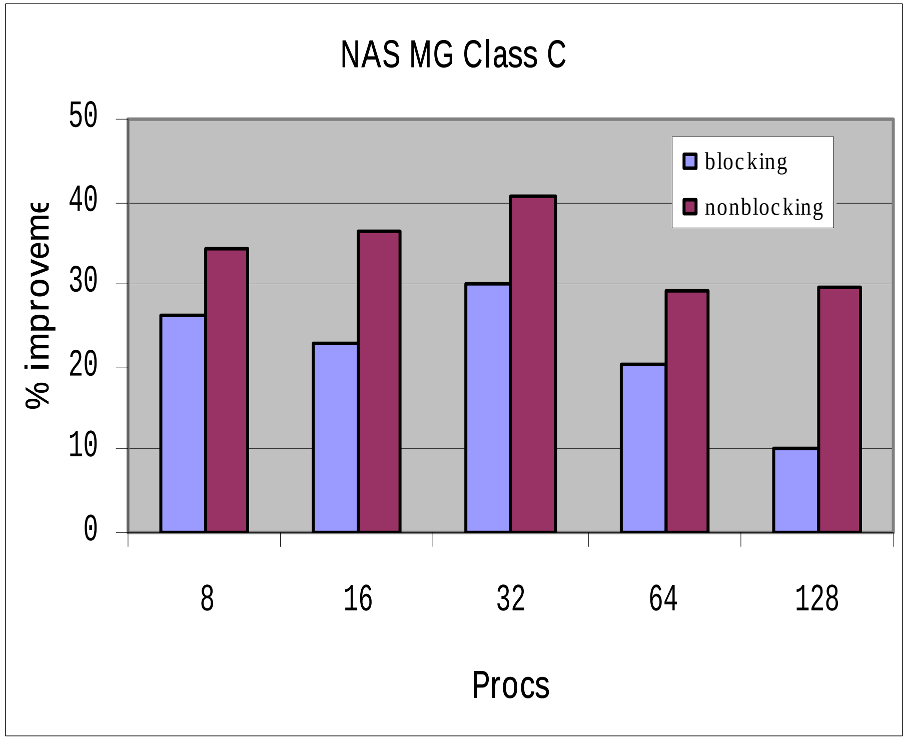
Benchmark

- •Issues a nonblocking call
- •inserts an increasing amount of computation
- •calls wait



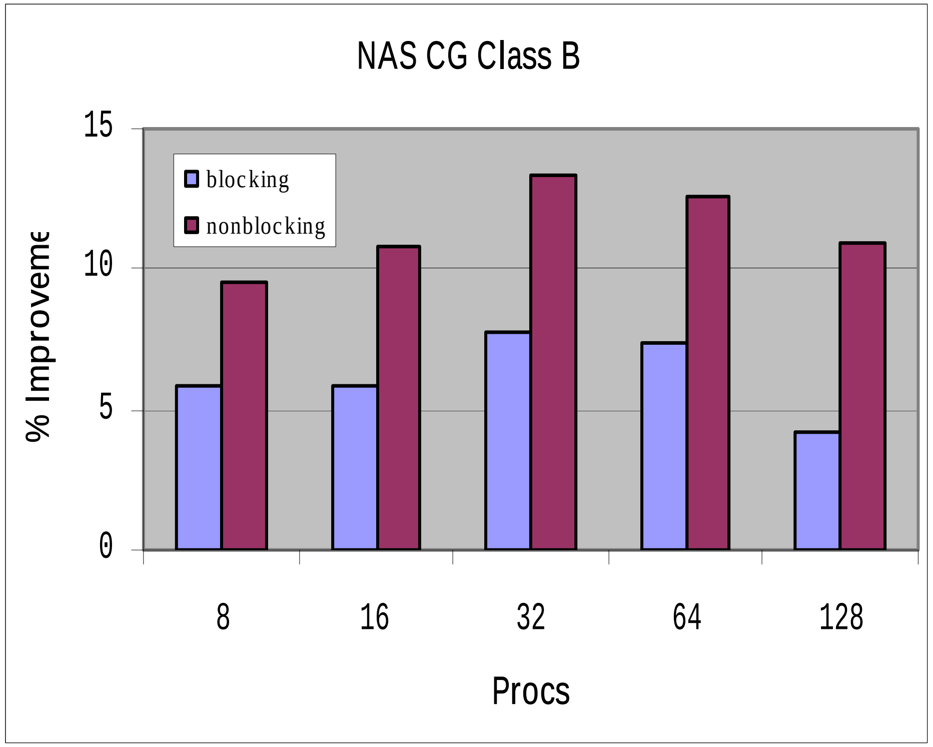
Improvement over MPI: NAS MG

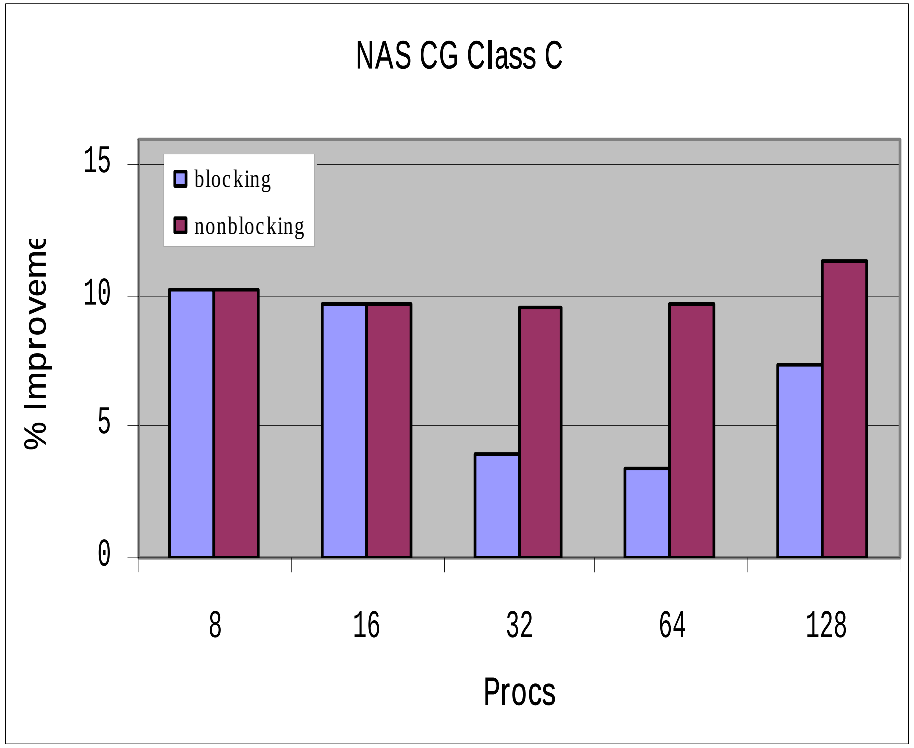




Linux, Pentium-4, Myrinet

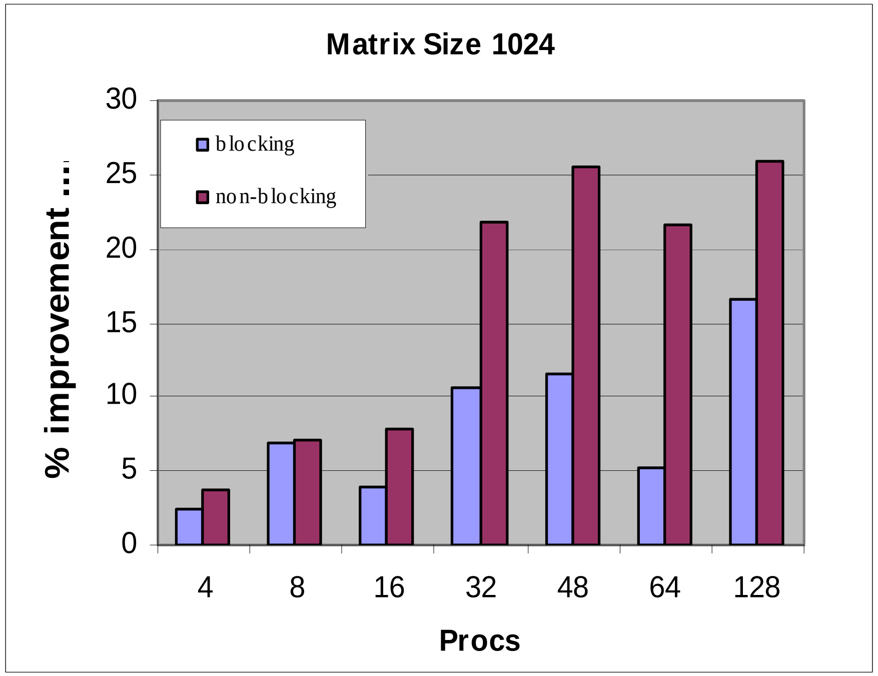
Improvement over MPI: NAS CG

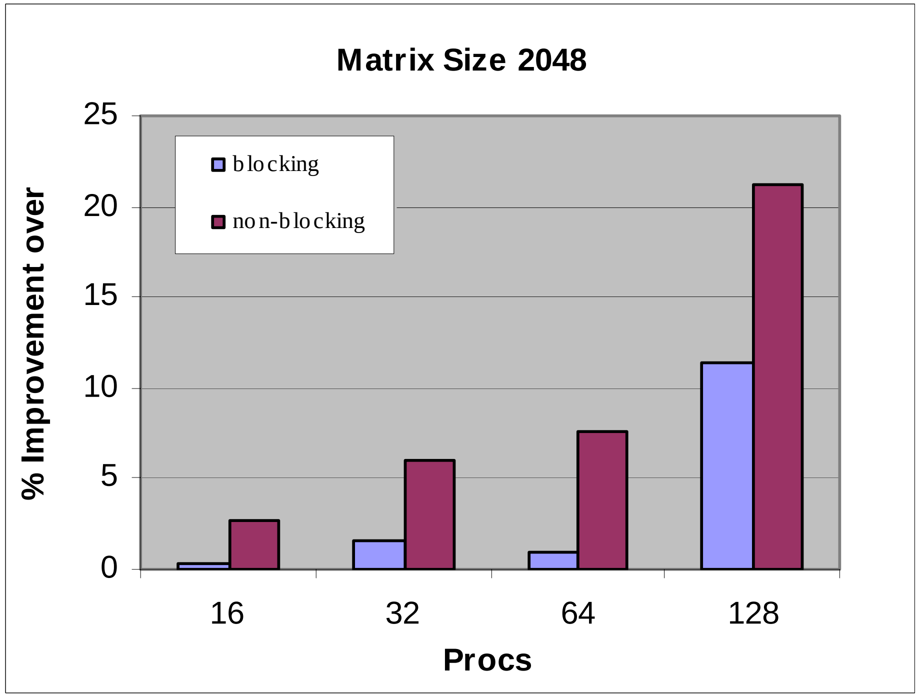




Linux, Pentium-4, Myrinet

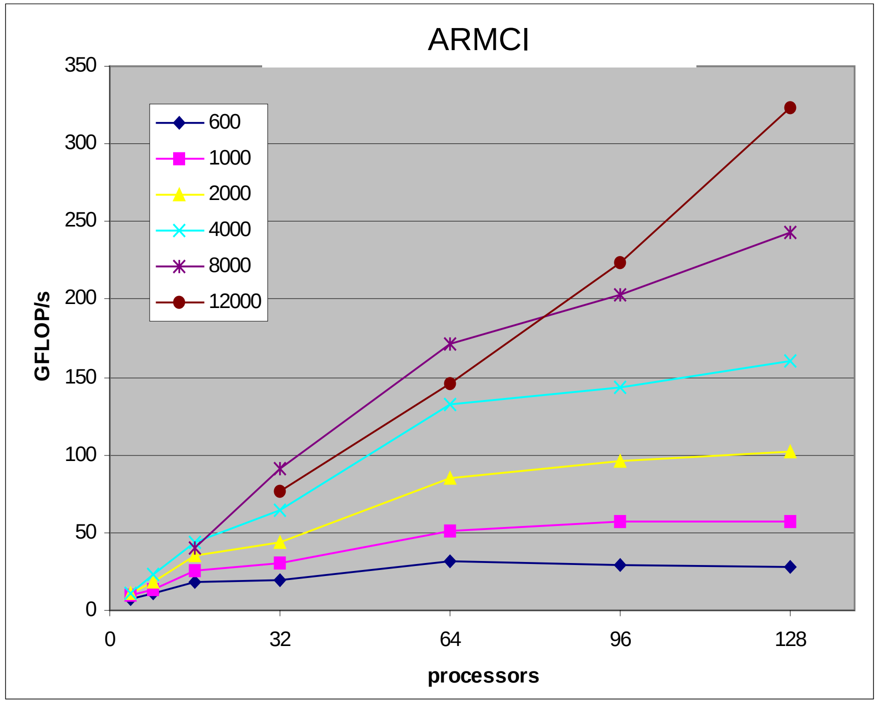
Matrix Multiplication
SUMMA (MPI) and ARMCI

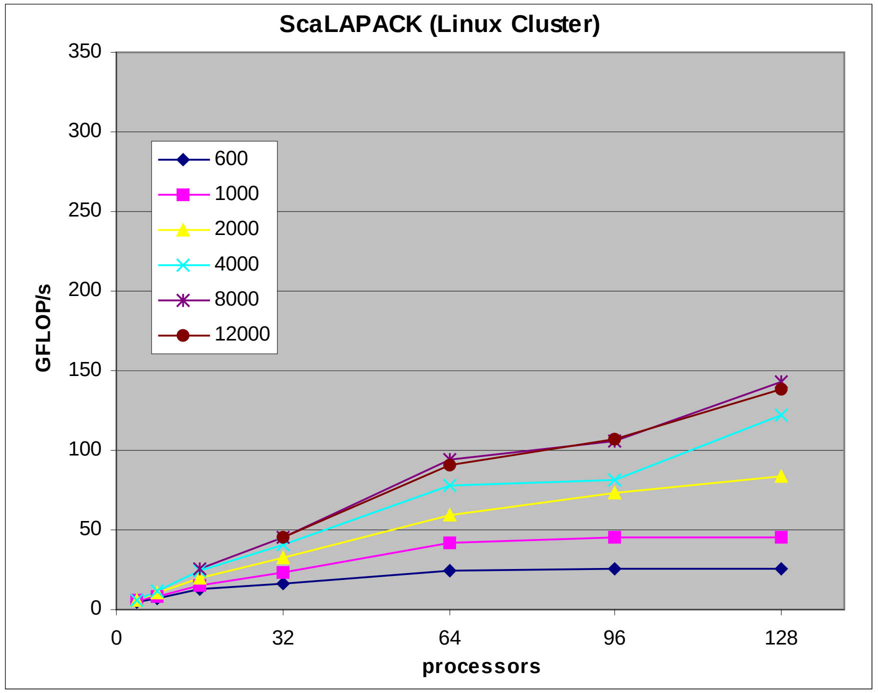




Linux, Pentium-4, Myrinet

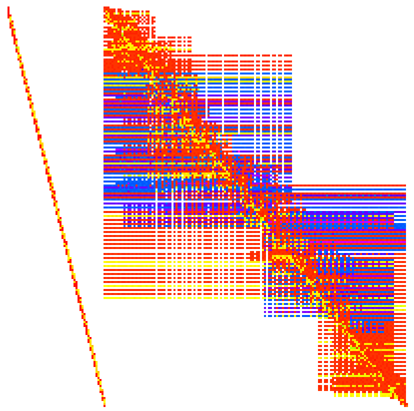
Aggregate Performance ARMCI and Scalapack/PBLAS pdgemm





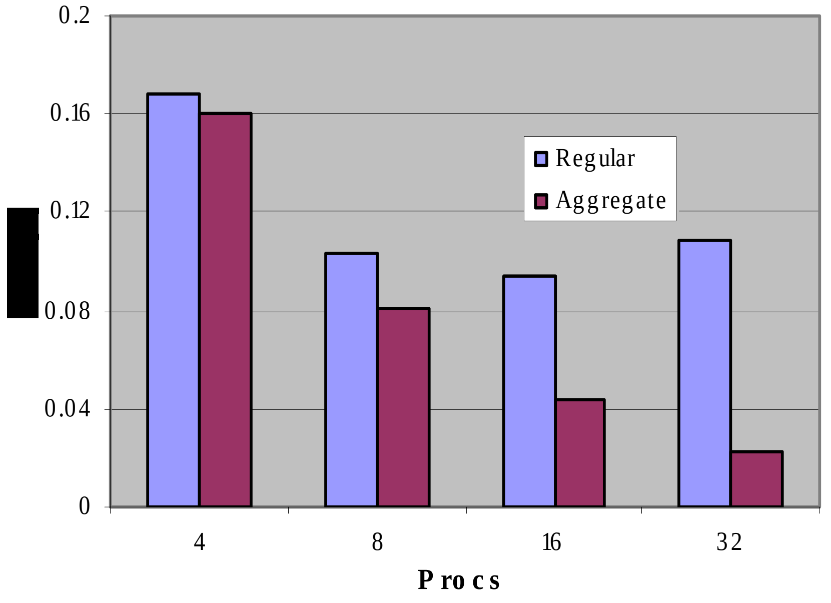
12/02/2003

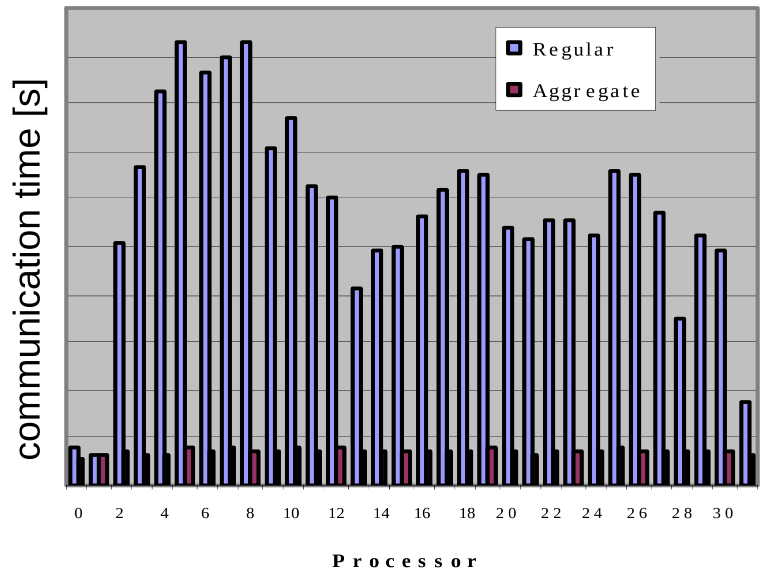
Sparse Matrix Vector Multiplication
message aggregation



Harwell-Boeing Sparse Matrix collection
– a finite element problem
– N= 41092 Non-zero elements: 1683902 (~.1%)

“aggregate handle” nonblocking ARMCI Get





Summary and Conclusions

- Mechanisms for latency tolerance increasingly important
  - Processor-Memory and Processor-Network performance gaps growing

- Paper focused on aggregation and nonblocking operations in context of RMA model
  - Efficient implementation on clusters
  - Handles underlying protocols transparently to the user through consistent and portable (GM, VIA, IBA, LAPI) interface
    - Zero-copy and buffered
    - Contiguous and noncontiguous data
  - Nonblocking operations provide real overlap
  - Effective implementation of small message aggregation

- Benchmarks show that these mechanisms help RMA model to deliver competitive performance to the send-receive model on clusters

Backup

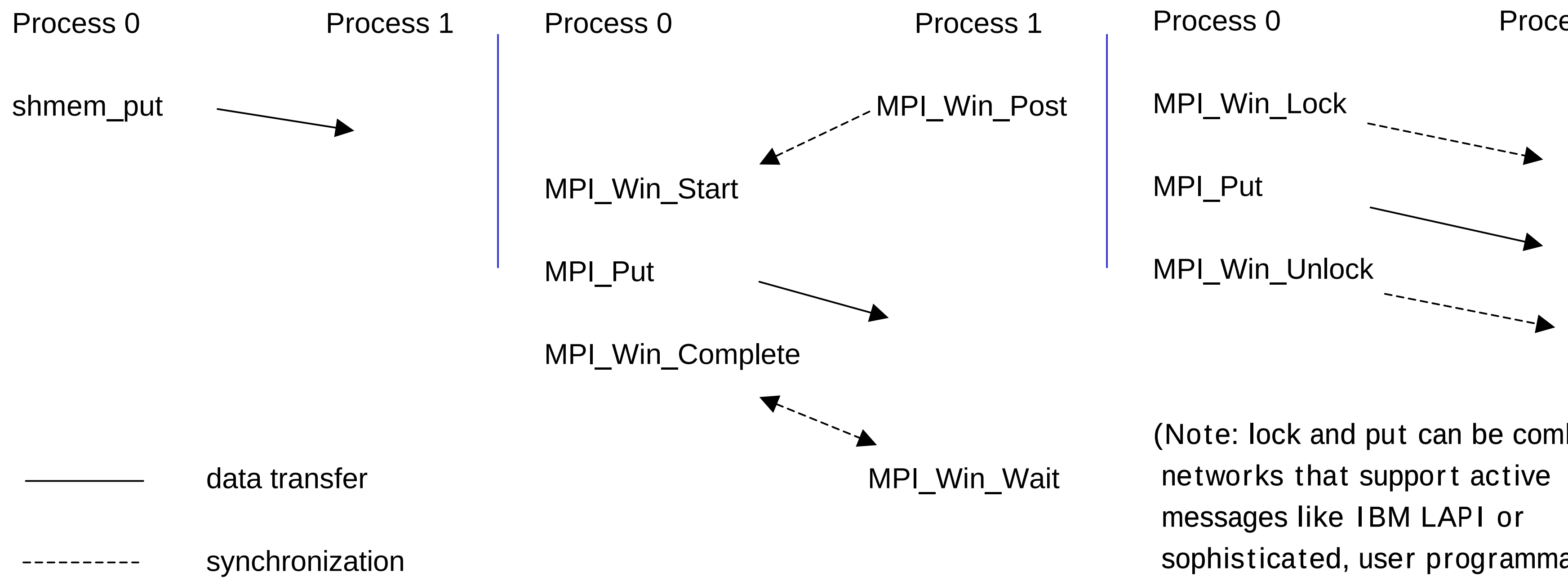
MPI-2 and other RMA models

Cray SHMEM

(IBM LAPI, GM, Elan, IBA similar)

MPI-2 1-Sided "active target"

MPI-2 1-Sided "passive target"



(Note: lock and put can be combined in networks that support active messages like IBM LAPI or sophisticated, user programmable adapters like Quadrics)

- MPI-2 1-sided is more synchronous than native RMA protocols
- Other RMA models decouple synchronization from data transfer