Supporting Efficient Noncontiguous Access in PVFS over InfiniBand

Jiesheng Wu (Ohio State University)

Pete Wyckoff (Ohio Supercomputer Center)

Dhabaleswar K. Panda (Ohio State University)

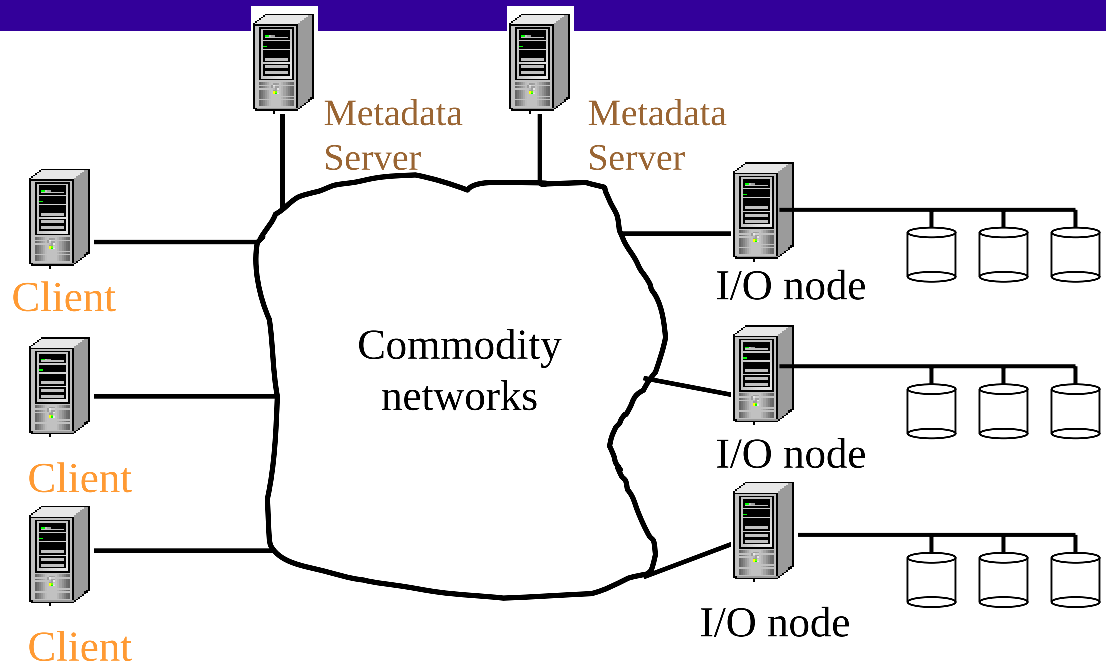
Outline

- Motivation
- Noncontiguous I/O Access in PVFS
- Network support for Noncontiguous I/O Access over InfiniBand
- Performance Evaluation
- Conclusions and Future Work

Parallel I/O in Commodity -Based Cluster Systems

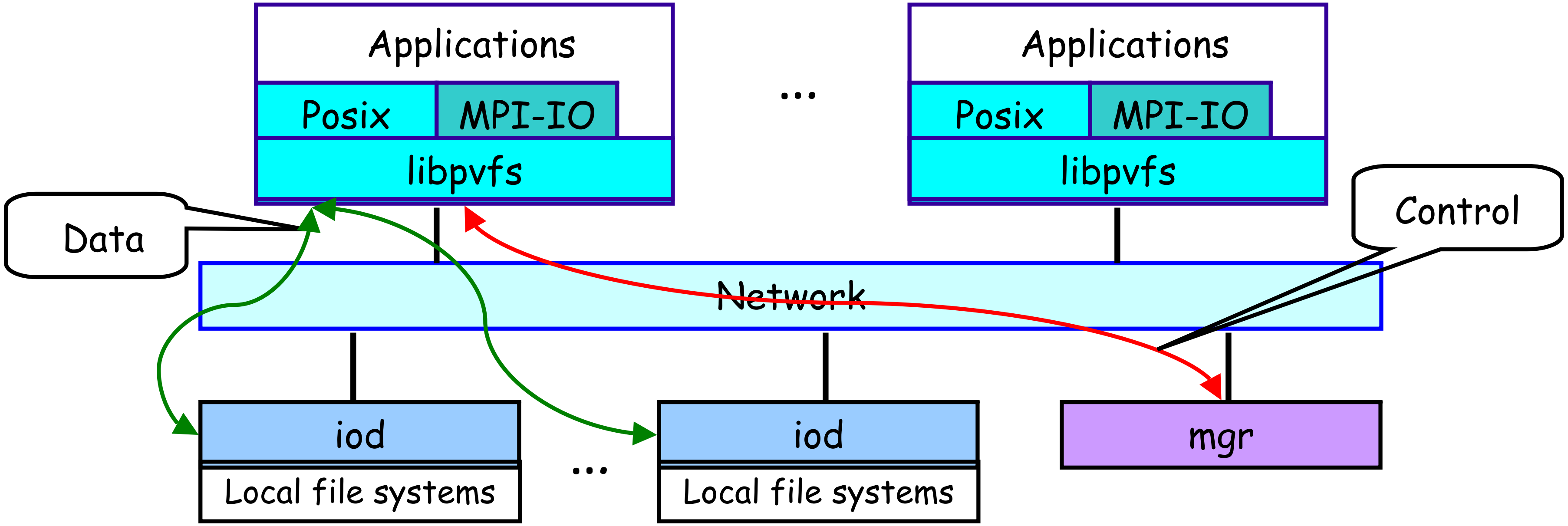
- Recognized as a problem for long time
- Requirements
  - High bandwidth concurrent write/read to the same file
  - Performance critical
  - Easy management
- Solutions
  - Parallel and cluster file systems
    - IBM GPFS, PVFS1, Lustre, many other cluster file systems
    - PVFS2 has been released recently
    - Easy file level management
  - Libraries
    - Too many. Check out http://www.cs.dartmouth.edu/pario/projects.html

Parallel I/O Architecture in Clusters



- Built upon commodity hardware and networks
  - Network storage systems
- Managed by parallel/cluster file systems
- Posix I/O and MPI-IO interfaces
- Potential for high performance, scalability, reliability, and manageability
  - But long way to go to achieve all these potentials

Parallel I/O in Clusters via PVFS



- PVFS: Parallel Virtual File System
  - Parallel: stripe/access data across multiple nodes
  - Virtual: exists only as a set of user-space daemons
  - File system: common file access methods (open, read/write, …)
- Designed by ANL and Clemson

Performance Issues (1)

- Low performance of network subsystems
  - Memory copying
  - Network access costs
  - Protocol overheads
- Solutions
  - User-level networking technologies
    - OS bypass with protection, protocol processing offload
  - Remote Direct Memory Access (RDMA)
    - Copy avoidance, transparency, reduced host intervention
  - InfiniBand
    - Offers these features
    - No change on production operating systems

Jiesheng Wu, Pete. Wyckoff, and D. K. Panda, PVFS over InfiniBand: Design and Performance Evaluation. In Proceedings of International Conference on Parallel Processing (ICPP 03). Oct. 2003.

Performance Issues (2)

- **Lack of integration and interaction among server application components and subsystems (file cache, file system, network subsystem,…)**
  - Redundant data copying among different components
  - Multiple buffering
  - Narrow interfaces
- Our efforts
  - Application-level cache management
    - Enable adaptation, application-specific optimization
    - Expose cache information to other components for better cooperation
  - Integration of cache management and communication buffer management
    - Eliminate copying and multiple buffering, increase the effective cache size
    - Reduce impact of memory registration and deregistration costs
  - Zero-copy I/O path from the disk to the network

  Jiesheng Wu, Pete Wyckoff, D. K. Panda, and Rob Ross Unifier: Unifying Cache Management and Communication Buffer Management for PVFS over InfiniBand. Submitted for publication.

Performance Issues (3)

- **Mismatch between the capability of file systems and application access characteristics**
  - Less than 10% of the peak I/O performance realized
  - Noncontiguous I/O access
  - Structured data access → a large number of noncontiguous small access
  - However, file systems optimized for large contiguous and sequential file accesses
- Solutions
  - Data sieving, Two-phase I/O, Disk-directed I/O, Server-directed I/O
  - PVFS list I/O

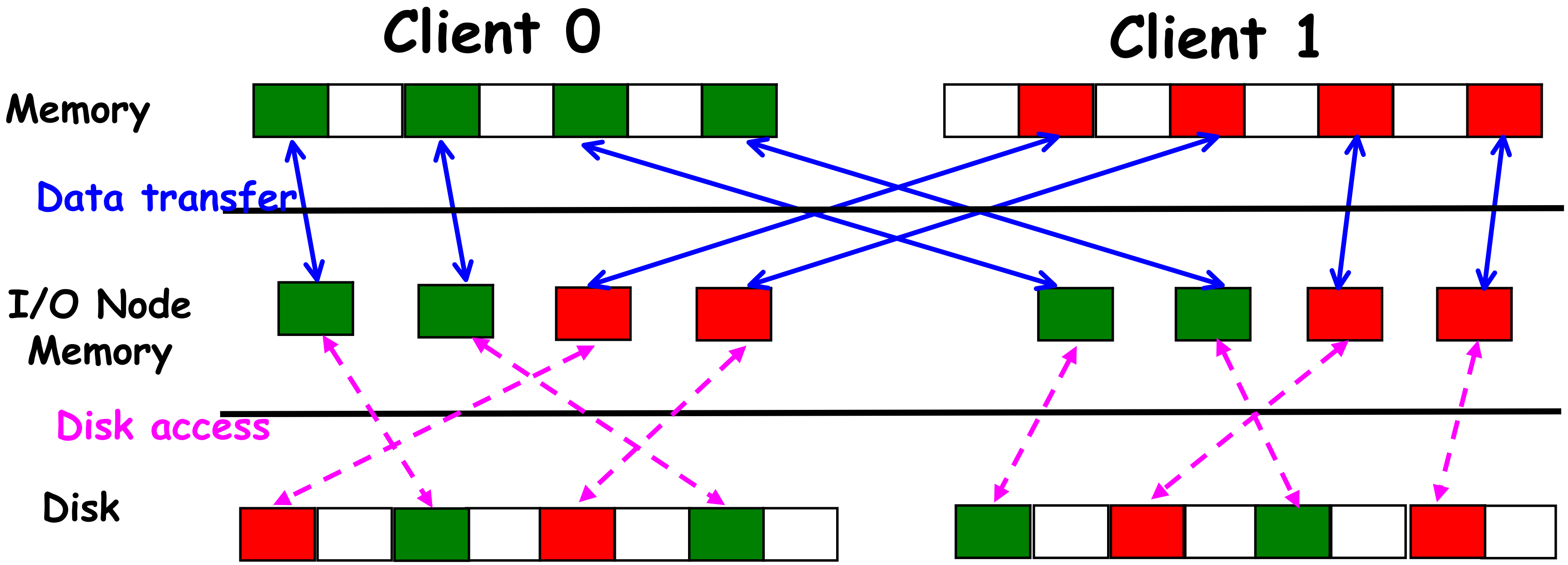
Our Objectives

- Two areas to improve noncontiguous access performance
  - Noncontiguous data transmission between the compute nodes and the I/O nodes
  - Noncontiguous file/storage access on the I/O nodes
- Key idea
  - Provide native support for noncontiguous data access from both the transport component and the file/storage component
- Our work in this paper
  - To leverage InfiniBand features to support efficient noncontiguous data transmission

Outline

- Motivation
- Noncontiguous I/O Access in PVFS
- Network support for Noncontiguous I/O Access over InfiniBand
- Performance Evaluation
- Conclusions and Future Work

A PVFS Noncontiguous I/O Access



- Non-contiguity, four combinations
  - Whether client memory is contiguous or not
  - Whether the file access is contiguous or not

PVFS List I/O

- A list of <offset, length> pairs used to represent the client buffers and the accessed regions in the file
- A single function used to transfer the whole request
  - Effectively reduce the number of request and reply message pairs
  - Increase data transfer efficiency
- Implementation based on TCP/IP
  - Take advantage of TCP/IP stream semantics
  - Noncontiguous data transmission is NOT considered as an issue

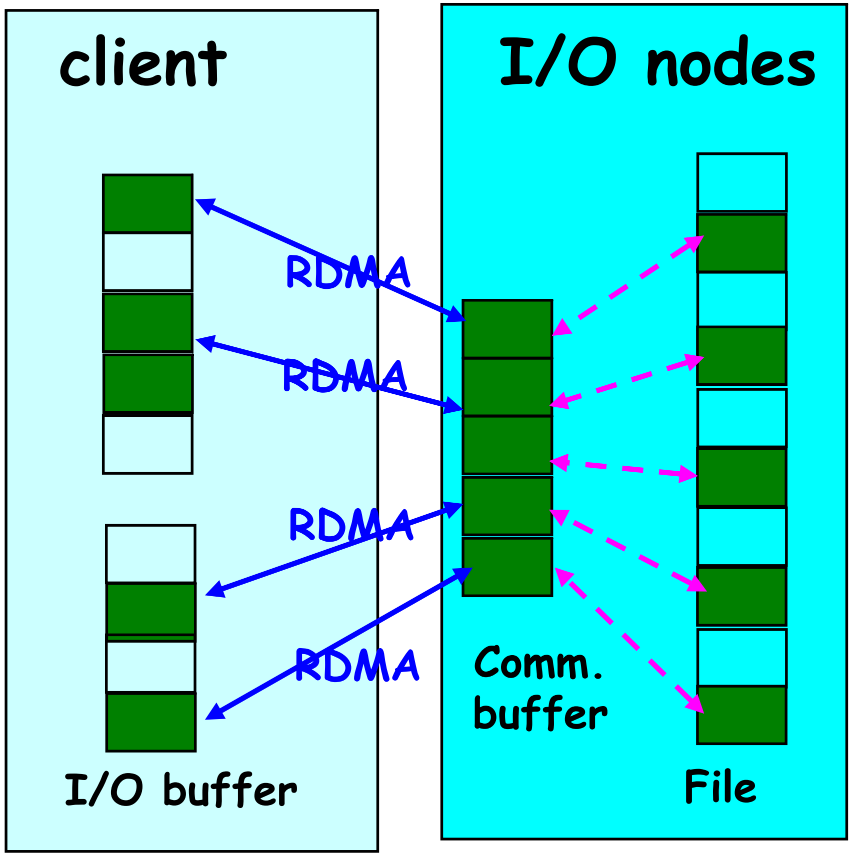
Outline

- Motivation
- Noncontiguous I/O Access in PVFS
- Network support for Noncontiguous I/O Access over InfiniBand
- Performance Evaluation
- Conclusions and Future Work

Noncontiguous Data Transmission

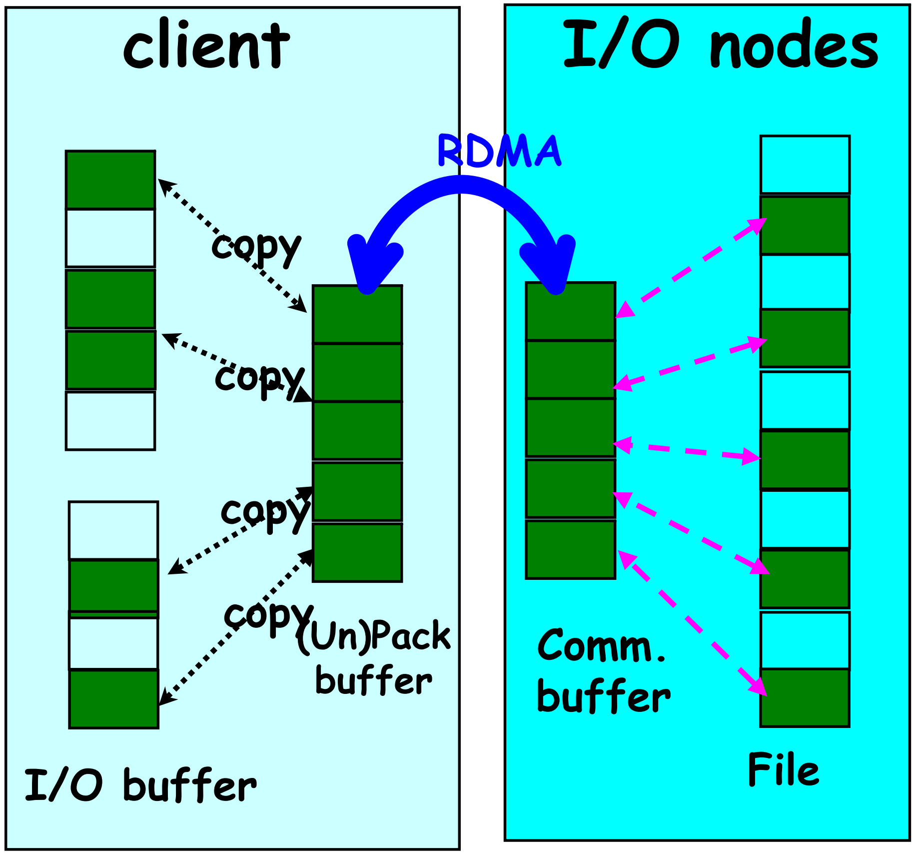
- Two often used approaches
  - Multiple messages
  - Pack/Unpack
- Multiple messages
  - Each operation sends only one contiguous piece
- Pack/UnPack
  - Pack noncontiguous data into a contiguous buffer
  - Receive data into a contiguous buffer, then unpack to user buffers
- RDMA operations can be used in both approaches

Multiple Messages with RDMA Operations



- Pros
  - No copy
  - Communication buffer on the I/O node can be noncontiguous
- Cons
  - Register I/O buffers
  - Unpredictable buffer alignment
  - The number of operations may be large

Pack/Unpack with RDMA Operations



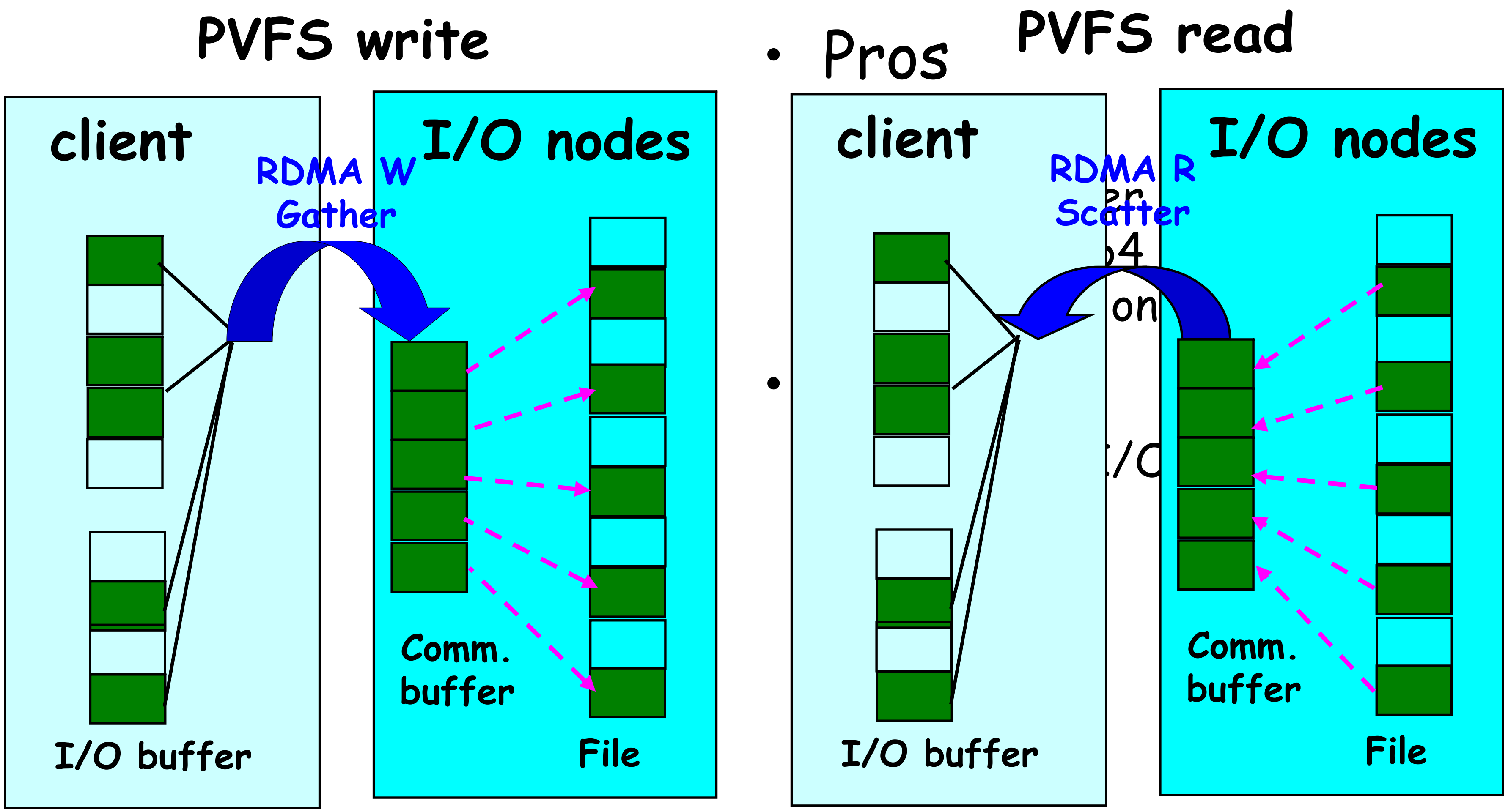
- Pros
  - Much less number of operations
  - Can use pre-registered pack buffer
  - Pack/unpack buffer may be well aligned
- Cons
  - Memory copy
  - Pack/unpack buffers consume memory

RDMA Write Gather/Read Scatter Approach

- Proposed to leverage InfiniBand features to support efficient noncontiguous data transmission
- RDMA Write Gather
  - Gather multiple data segments together and place them in a single communication buffer
  - For PVFS write
- RDMA Read Scatter
  - Scatter data into multiple buffers from a single communication buffer
  - For PVFS read
- Feasibility due to
  - Client side initiates RDMA operations
  - Communication buffer is always contiguous on the I/O node in our design

Details about RDMA Write Gather/Read Scatter Approach

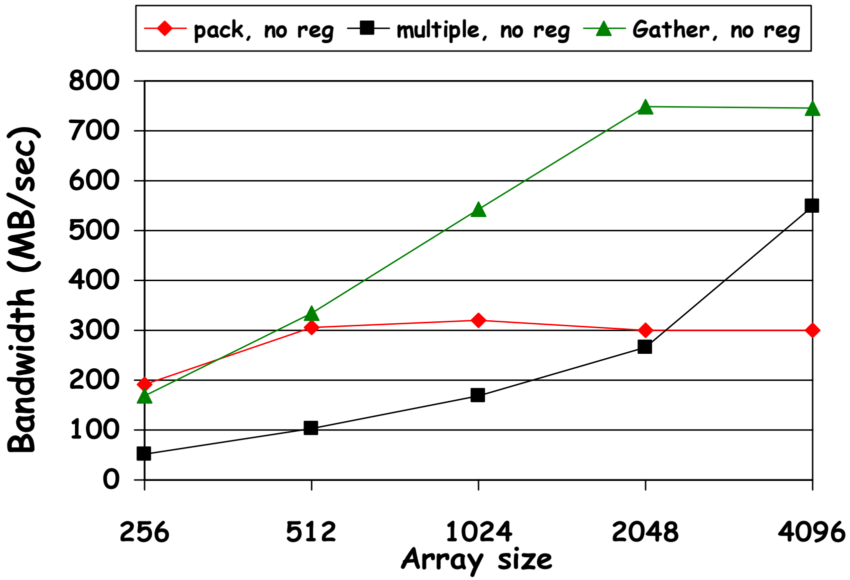
• Pros

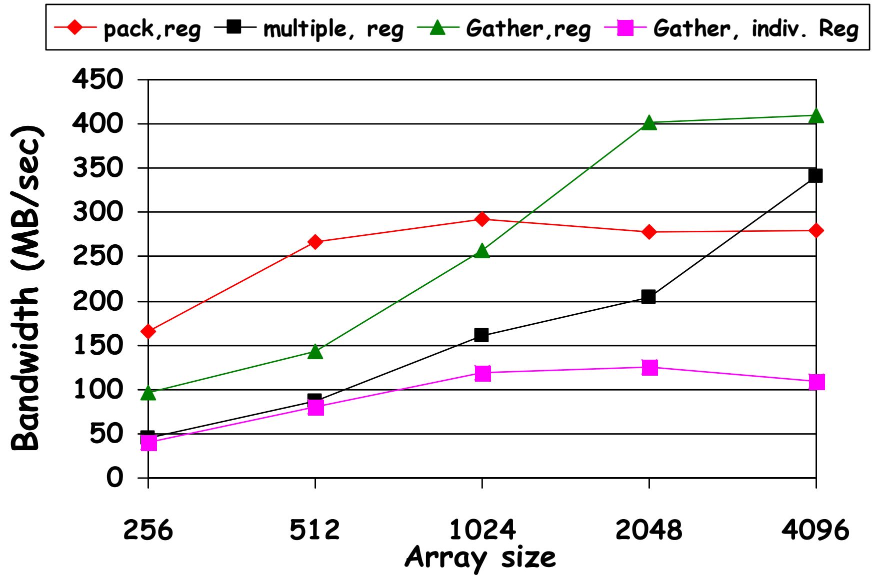


Tradeoffs in the Transfer Methods with RDMA Operations

- Copy or memory registration
- Number of communication operations
- Buffer alignment
- Application buffer access patterns

Making tradeoffs





- 2-D sub-array on 4 processes using a block distribution on both dimensions
- Memory copy is really costly
- If no registration, Gather is a clear win (similarly for Scatter)

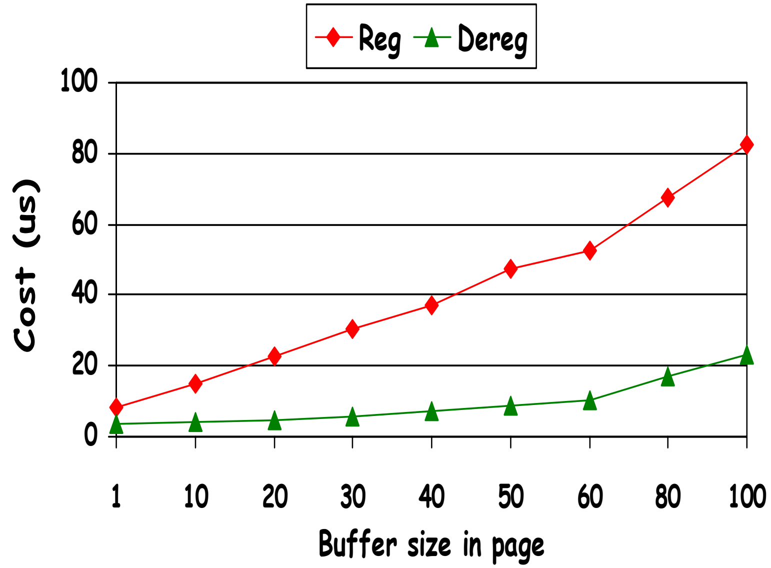
- With registration, pack is the best when the transfer is small, Gather is the best for the large transfer
- Multiple is always worse than Gather
- Gather with individual registration is the worst

Minimizing memory registration/deregistration overhead

- Vendors improve their SDK and hardware
  - Fast memory registration facilities in VAPI
  - It is still expensive
- Application-level solutions
  - Pin-Down cache, dealing with contiguous buffers
- However, new complication in registering a list of buffers
  - Large number of buffers
  - Holes between buffers
  - Need other solutions

Registration/deregistration on a List of Buffers

- Two simple approaches
  - Individual registration
  - Single registration
- Individual registration
  - Register each buffer one by one
  - The total per-operation overhead may be very high
  - Lead to registration cache thrashing
- Single registration
  - Register the whole region covering all buffers
  - The unused memory regions may be very large, the total per-page overhead may be very high
  - Some unused memory regions may not be allocated
- Our solution
  - Optimistic Group Registration (OGR)



T = a x p + b

b: per-operation overhead

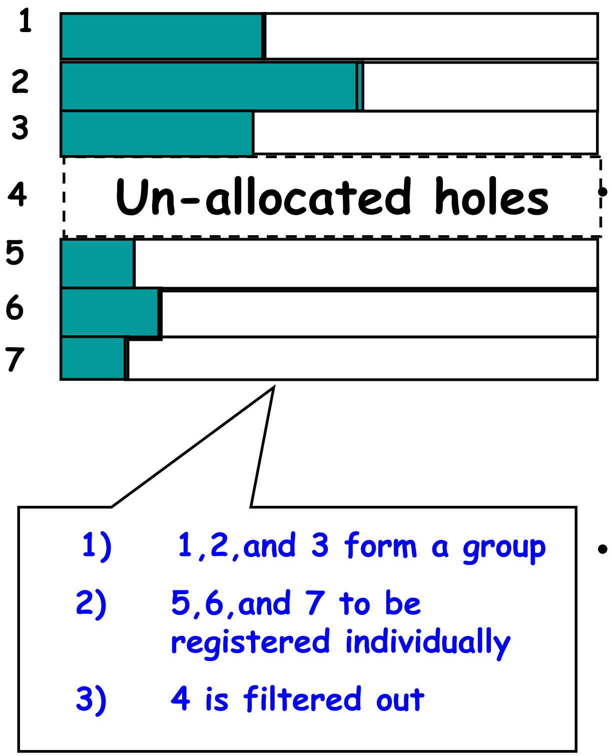
a: per-page overhead

p: buffer size in page

Reg: a = 0.77us, b = 7.42us

Dereg: a = 0.23us, b = 1.1us

Optimistic Group Registration (OGR)



- Optimistic Group Registration (OGR)
  - Make a tradeoff between the number of registration/deregistration operations and the total size of registered memory region
- Three steps
  - Group buffers into regions based on the cost model, avoiding truly large "holes"
  - Optimistically register each grouped region
  - If fails, filter out un-allocated "holes" or fall back on individual registration for the group region
- Benefits
  - Efficient in common cases where un-allocated holes are rare
  - Safe by virtue of filtering out un-allocated holes
  - Transparent to applications

Choosing an Appropriate Method

- Hybrid approach
  - For small transfer, Pack/Unpack
    - Memory copying costs are less than registration costs
  - For large transfer, RDMA Gather/Scatter
    - Avoid memory copying
    - OGR can achieve efficient registration
    - Pin-down cache can further reduce registration and deregistration costs
- No implementation of the Multiple approach in our work
  - The communication buffer is always contiguous in our design
  - The Multiple approach is always worse than the Gather/Scatter

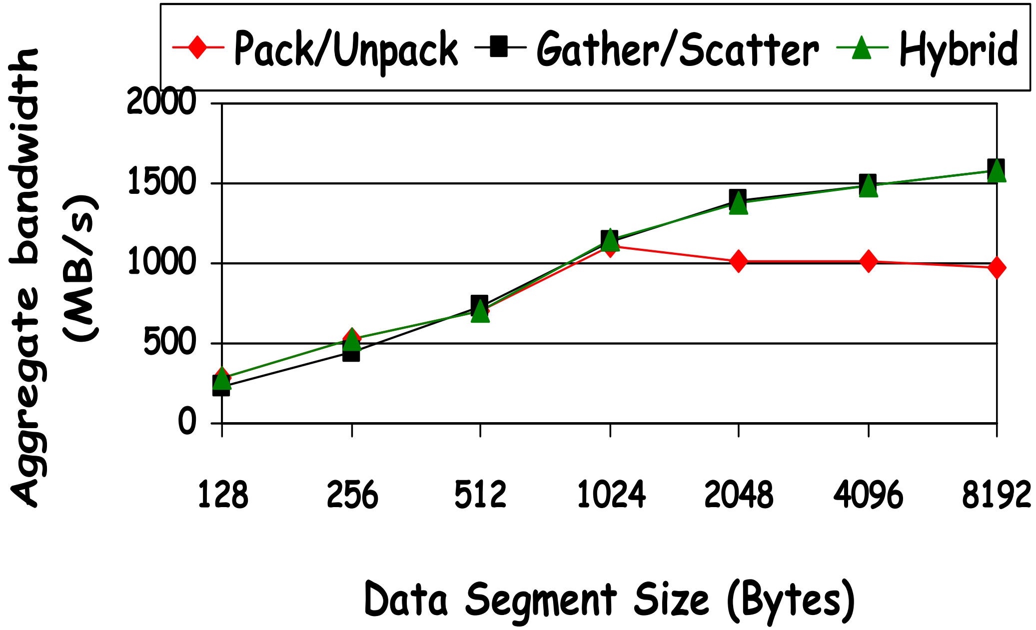
Outline

- Motivation
- Noncontiguous I/O Access in PVFS
- Network support for Noncontiguous I/O Access over InfiniBand
- Performance Evaluation
- Conclusions and Future Work

Performance Evaluation

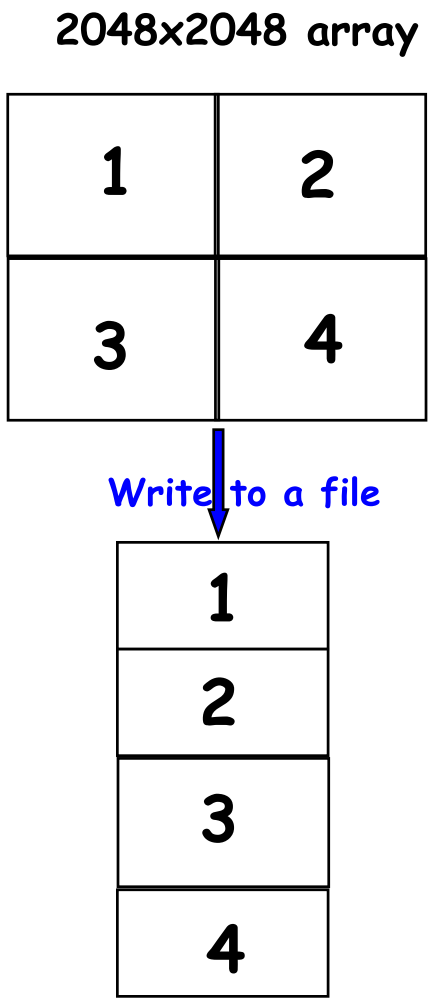
- Experimental Setup
  - A cluster with 8 nodes, SuperMicro P4DL6 motherboard, Intel Xeon 2.4 Ghz processor
  - 512kB L2 cache, FSB: 400 MHz, 512 Mbytes main memory
  - Mellanox InfiniHost MT23108 DualPort 4x HCA
  - InfiniScale MT43132 Eight 4x port switch
  - HCA SDK: thca-x86-0.1.2-rc12-build-001
  - Seagate ST340016A ATA 100 40GB disk
  - PVFS 1.5.6
- Tests
  - PVFS list I/O write/read
  - Effects of Optimistic group registration
  - Performance of NAS BTIO benchmark

PVFS List I/O read



- Cached read 128 segments, the segment size varies from 128 bytes to 8192 bytes
- Gather/Scatter and Hybrid can achieve a factor of 1.6 improvement
- Similar results for PVFS List I/O write

Optimistic Group Registration Performance

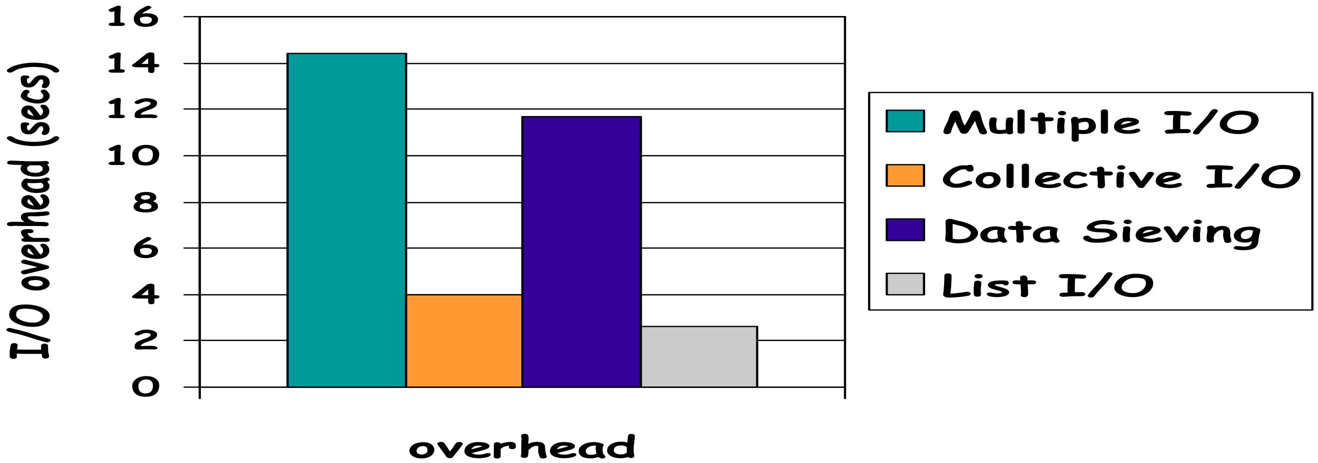


- Test info
  - Four servers, four clients , Sequential write
  - Case 1: no unallocated hole
  - Case 2: 10 unallocated holes

case	No sync (MB/s)	Sync (MB/s)	# reg	Overhead (us)
Case 1				
Ideal	1010	82	0	0
Indiv.	424	73	1024	5254
OGR	950	≈ 82	1	227
Case 2 (other numbers unchanged)				
OGR	879	≈ 82	11	496

- OGR achieves 2.1-2.3 times faster without sync, 1.1 times with sync

Performance of BTIO



- Class A, four nodes, about 429MBytes data read and write, respectively
- List I/O
  - Reduce requests to 1360 from 163840 in Multiple I/O, 82040 in Data Sieving
  - Same reduction in registration numbers due to OGR
  - Less network traffic compared to Collective I/O
- With list I/O: 1.6 and 5.6 times better in terms of I/O overhead

Conclusions

- Native support for noncontiguous data access is important
- RDMA Gather/Scatter can be leveraged to achieve efficient noncontiguous data transmission
- Registration on a list of buffers should be handled well
- We provide efficient PVFS List I/O over InfiniBand
- Benefits:
  - 20-60% improvement seen in a throughput micro-benchmark compared to other noncontiguous data transmission approaches in List I/O
  - Our List I/O implementation performs 1.6-5.6 times faster in terms of BTIO I/O overhead compared to other approaches in MPI-IO, including Multiple I/O, Collective I/O, and Data Sieving

Ongoing/Future Work

- Take advantage of advances in MPI Datatype to represent I/O requests
- Smart I/O server architecture
  - Integration and interaction among components
  - Take advantage of processing power and memory in the NIC card
    - NIC memory as cache for “hot” data

Further Information

http://nowlab.cis.ohio-state.edu/
http://nowlab.cis.ohio-state.edu/mpi-iba/