Designing Efficient FTP Mechanisms for High Performance Data-Transfer over InfiniBand

Ping Lai, Hari Subramoni, Sundeep Narravula, Amith Mamidala and Dhabaleswar. K.Panda

Computer Science and Engineering Department
The Ohio State University, USA

Outline

- Introduction & Motivation
- Designing Zero-copy FTP Mechanism
- Experimental Results
- Conclusions & Future Work

Introduction

- Increasing demands in high ending computing leads to the deployment of compute and storage nodes in global scale
- Bulk data transfer within and across clusters is important
  - Data-sets distribution, content replication, remote site backup
- FTP is the most popular mechanism
  - E.g GridFTP in WAN

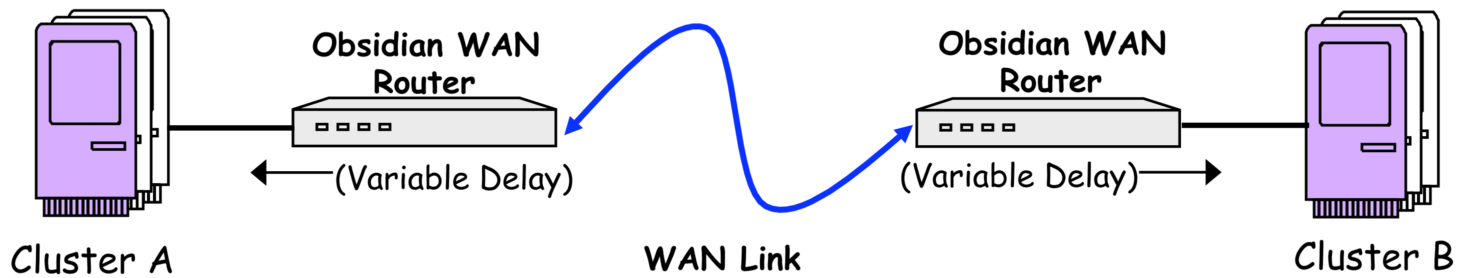
Introduction (cont.)

- System Area Network (SAN) gains momentum
  - InfiniBand, 10Gigabit Ethernet/iWARP etc.
  - High bandwidth, low latency
  - Other advanced features: zero-copy communication, RDMA operations
- IB WAN routers are introduced to extend IB capabilities beyond a cluster
- Zero-copy communications are possible in WAN
  - Provides new scope for designing FTP mechanisms !

InfiniBand

- Open Industry Standard based
- High Performance
  - High Bandwidth (~ 40Gbps)
  - Low Latencies (~1 us)
- Multiple Transport modes
  - Including RC, UD
- Two communication semantics
  - Channel semantics: send/recv
  - Memory semantics: RDMA operations
- WAN capabilities!!
  - Obsidian Longbow routers
  - Bay Microsystem products

InfiniBand WAN



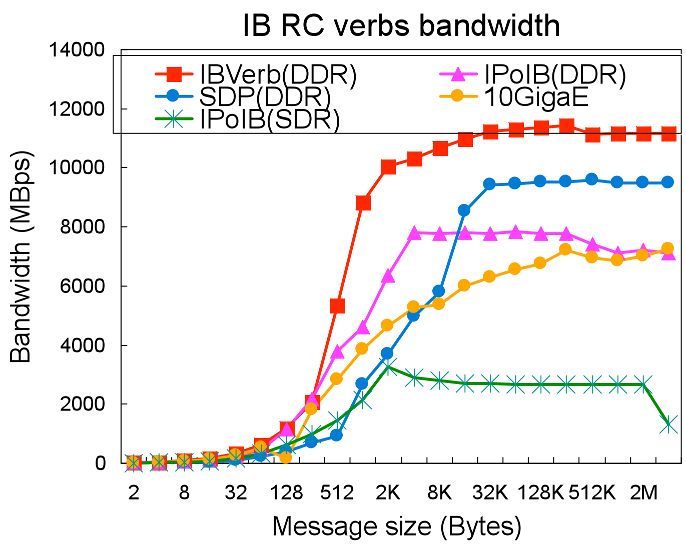
- Point-to-point inter-cluster links
- SDR data rate
- Varying delay emulates the WAN distance

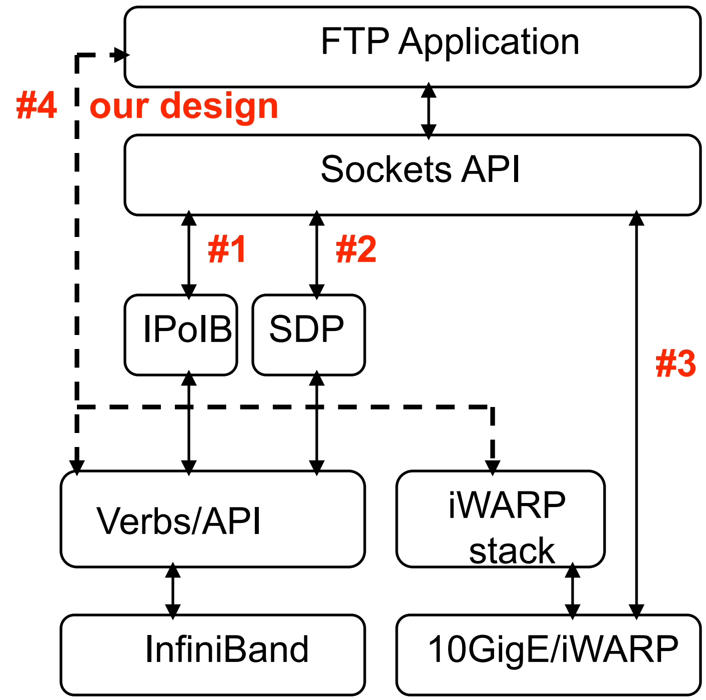
Delay (us)	Distance Emulated(km)
0	0
10	2
100	20
1000	200
10000	2000

Links emulate each *km* of WAN link length with an increase of 5 *us* to each packet latency

Implement FTP in IB LAN & WAN

- Directly use the existing sockets based FTP implementations
  - Scheme 1, 2, 3
  - All lose the native IB benefits

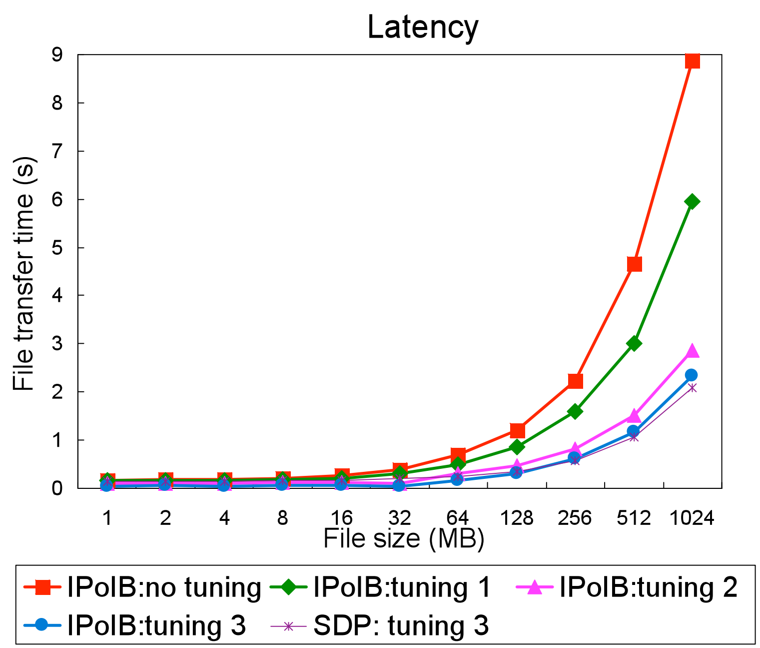


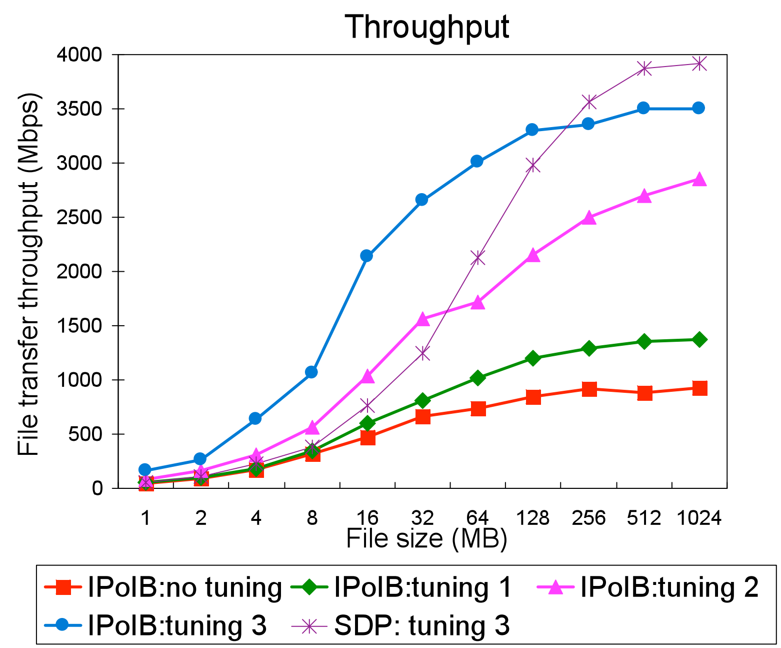


- Need to design native IB based mechanisms (scheme 4)
  - Efficient data transfer by making use of native IB benefits

More Motivation

- Example: GridFTP cannot achieve good performance in IB scenario
  – Through IPoIB or SDP





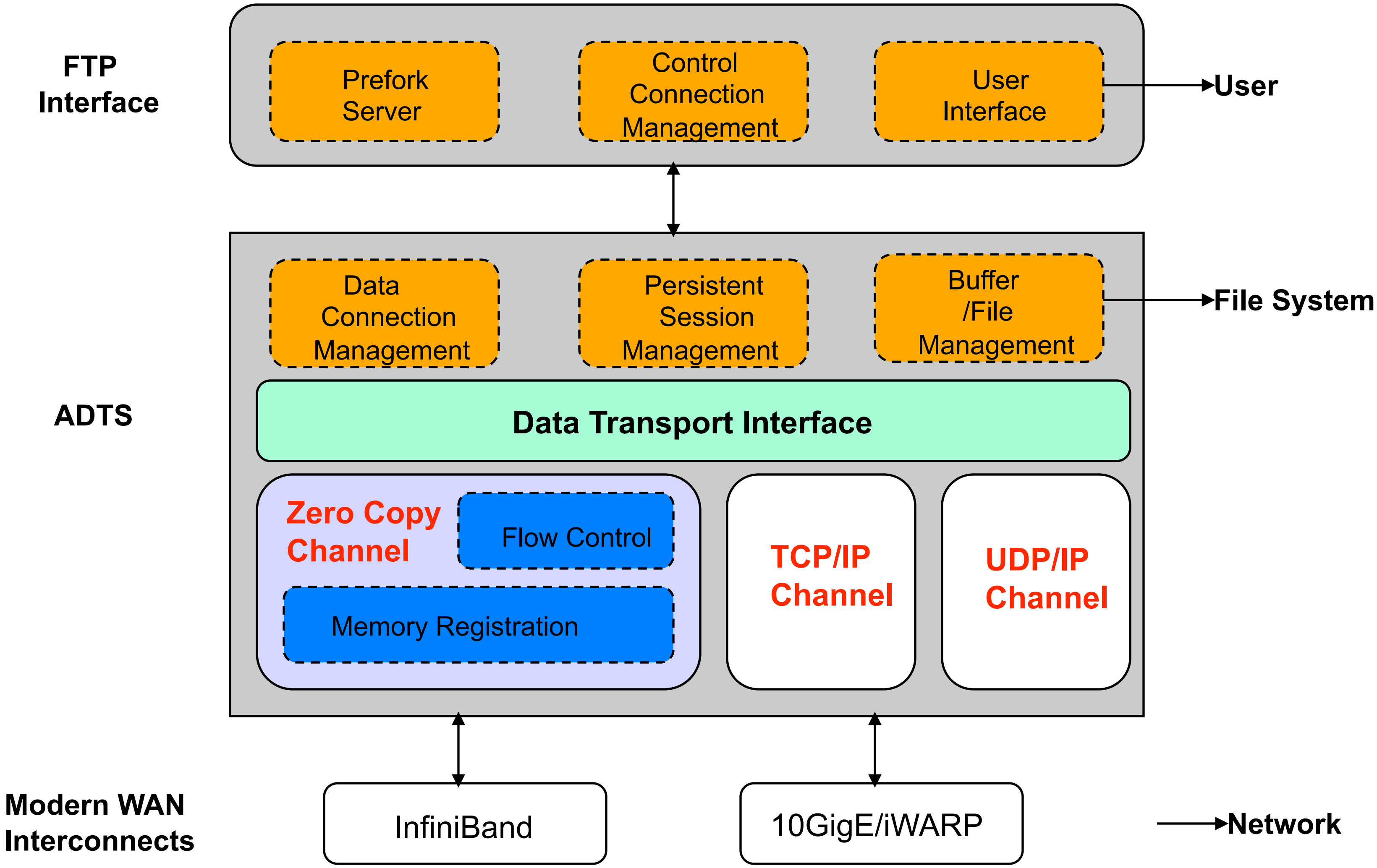
Tuning 1: increase MTU
Tuning 2: use parallel streams + Tuning 1
Tuning 3: adjust TCP buffer size & block size + Tuning2

Low-level IB benefits are not fully translated into FTP performance !

Outline

- Introduction & Motivation
- Designing Zero-copy FTP Mechanism
- Experimental Results
- Conclusions & Future Work

FTP-ADTS Architecture



Advanced Data Transfer Service (ADTS)

- Support various transport
  - TCP/IP channel, UDP/IP channel, Zero-copy channel
  - Dynamically adapted on a per client connection basis
- Data connection management
  - Initiate connection to remoter peer based on particular channel
- Persistent session management
  - Will be discussed in detailed design
- Buffer/File management
  - Will be discussed in detailed design

Zero-copy Channel Design

- Two alternatives
  - Memory semantics using RDMA
  - Channel semantics using send/recv

	Zero-copy	Latency	Flow control	Completion notification	Use RC/UD	Buffer info exchange
RDMA	Yes	Lower (may not seen in WAN)	Explicit	Explicit	Only RC	Needed
send/recv	yes	Also low	Easy	Implicit	Both	No need

Send/Recv based Design

- Buffer management
  - Buffers need to be registered and pinned in memory
  - Keep a small set of pre-allocated buffer
  - More buffer is allocated and registered on demand; unregistered and released after completion
- Flow control
  - Sender must be ensured that the receiver has available buffer
  - Receiver side flow control by using Shared Receive Queue (SRQ)
  - Fall back to explicit flow control to throttle the sender as needed

Additional Design Enhancements

- Memory registration cache
  - Registration cost is high
  - Do not perform de-registration for frequently used buffer
  - Not work for the situation that each file is transferred on different data connections!
- Persistent sessions
  - Keep data connection and the associated buffer alive during multiple files transfer
- Pipelined data transfer
  - Designed with two threads
    - Network thread: handle network related work
    - Disk thread: handle reads/writes from/to the disk
  - Data transfers are packetized and pipelined

FTP-ADTS Design

- Utilize zero-copy ADTS layer
- User interface
  - Handle user interaction
- Control connection management
  - Socket based control connection
  - Relay control info: FTP commands, errors
  - Negotiate active/passive mode and transport support
- Prefork server
  - Main FTP server daemon forks multiple processes for different clients
  - Maintain a small pool of pre-forked processes

Outline

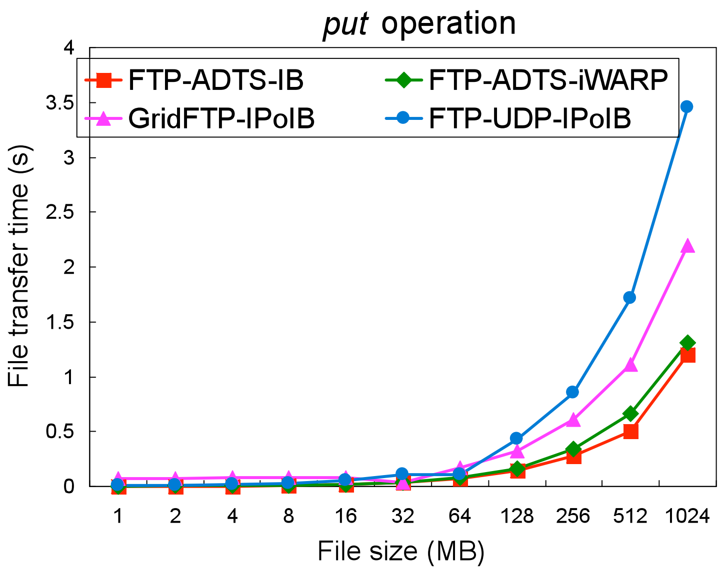
- Introduction & Motivation
- Designing Zero-copy FTP Mechanism
- Experimental Results
- Conclusions & Future Work

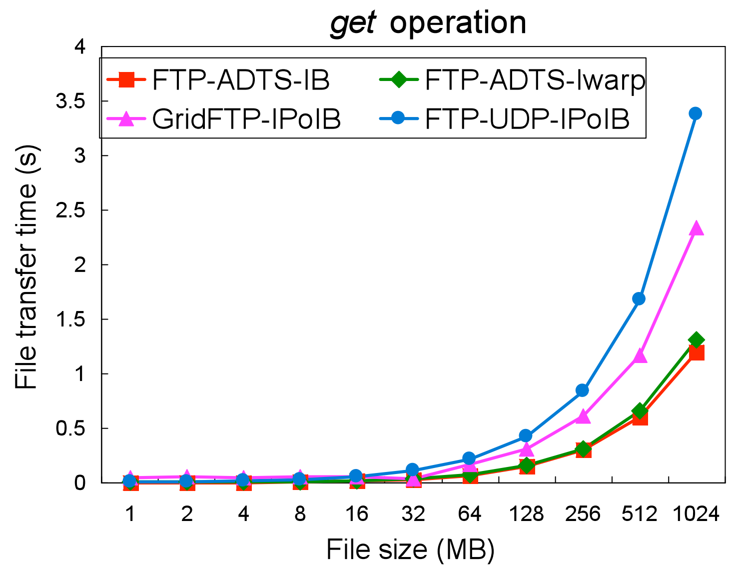
Experimental Setup

- Testbed
  - Dual quad-core Xeon processors, 6 GB memory
  - Linux kernel 2.6.9.34
  - Use InfiniBand (IB) DDR ConnectX HCAs with OFED 1.3
  - Use Chelsio T3b 10 Gigabit Ethernet/iWARP adapters
  - Nodes are divided into cluster A and cluster B that are connected with Obsidian routers
- Experiment design
  - GridFTP and FTP-UDP: base line reference
  - Tune TCP window size and MTU size for best performance

NETWORK-BASED COMPUTING LABORATORY

Performance in IB LAN

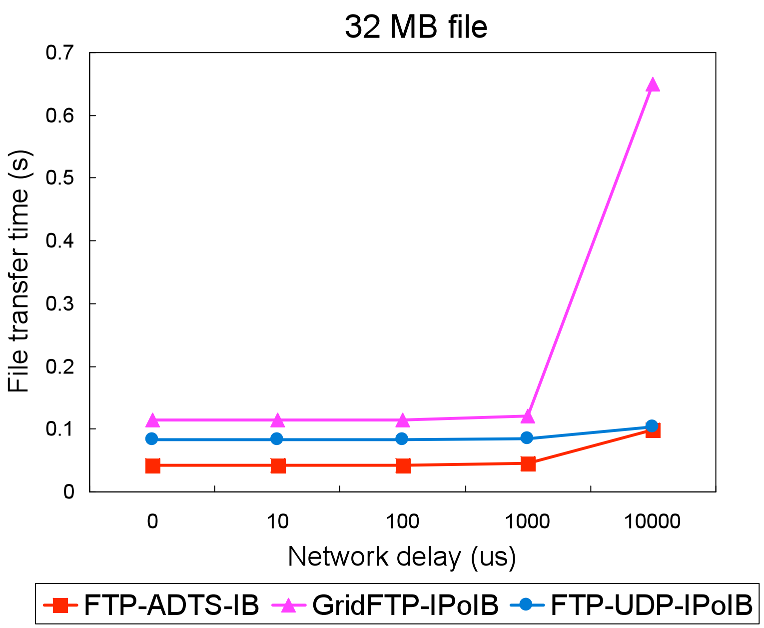


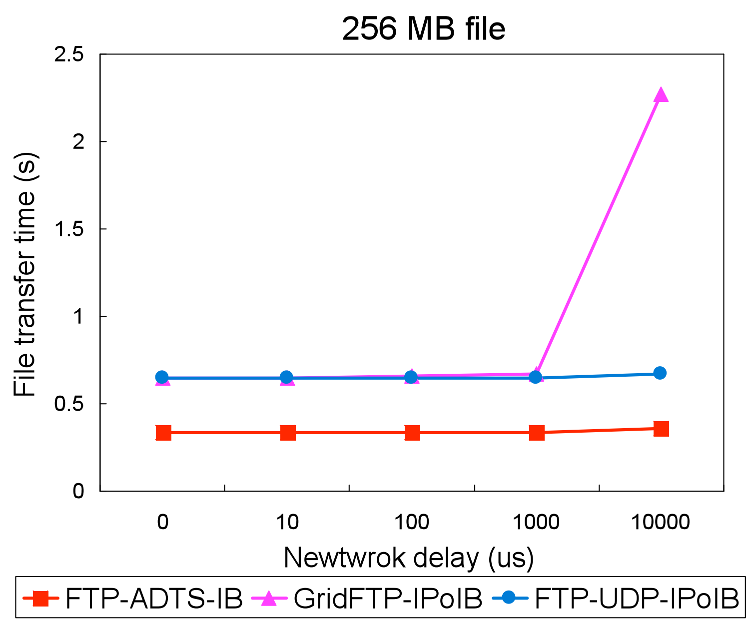


- FTP-ADTS improves performance by up to 95%
- Zero-copy operations has lower latency thatn IPoIB based operations

OHIO STATE

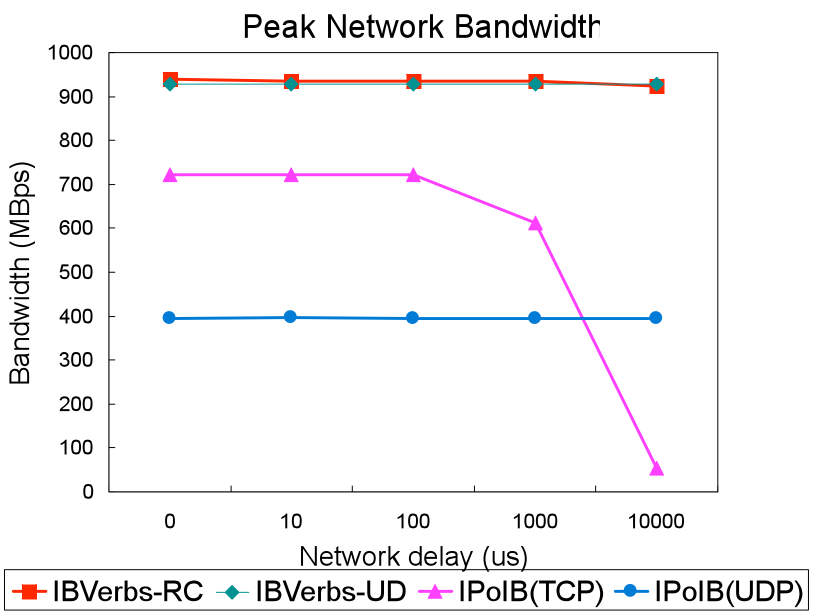
Performance in IB WAN

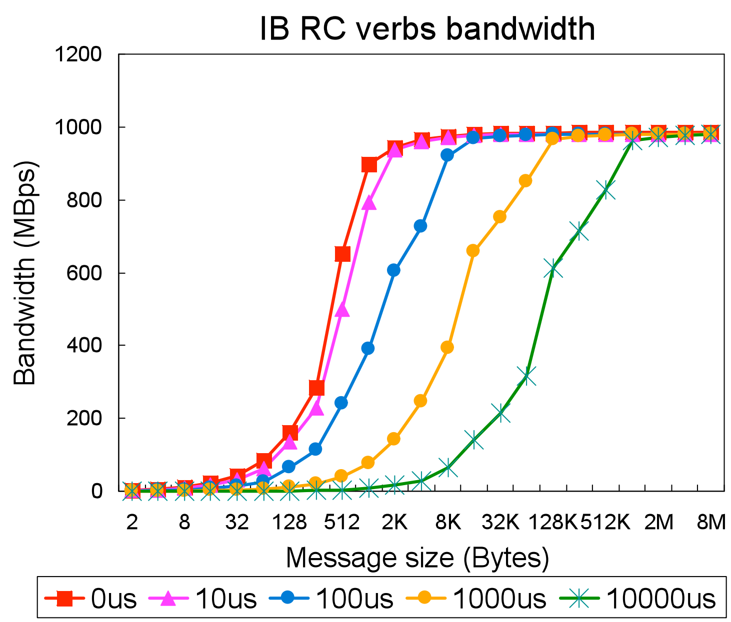




- File transfer time for *get* operation
- FTP-ADTS sustains good performance for large WAN delays
- IPoIB (GridFTP) has degradation due to flow control, RTT, MTU etc.
- FTP-UDP has the benefits of UDP over WAN

In-depth Analysis



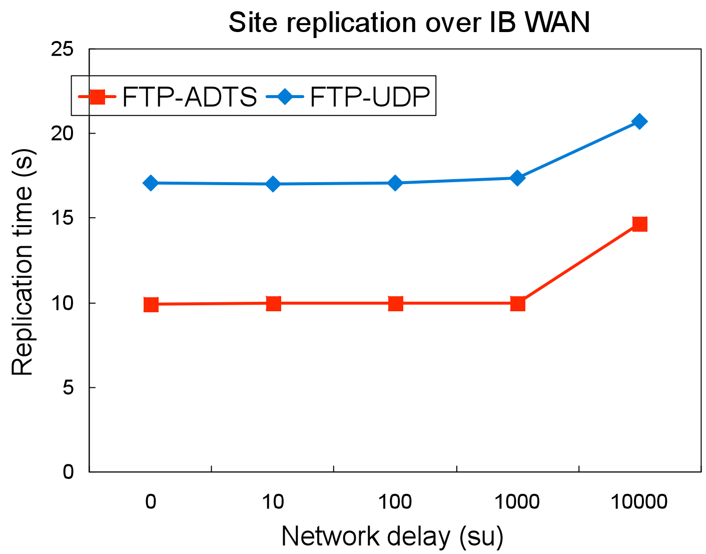


- IB verbs have stable highest bandwidth as delay increases
- The trends are consistent with the FTP performance over WAN

- Large messages can sustains the bandwidth with increasing network delays
- We use very large packet size (e.g. 1M) in FTP-ADTS

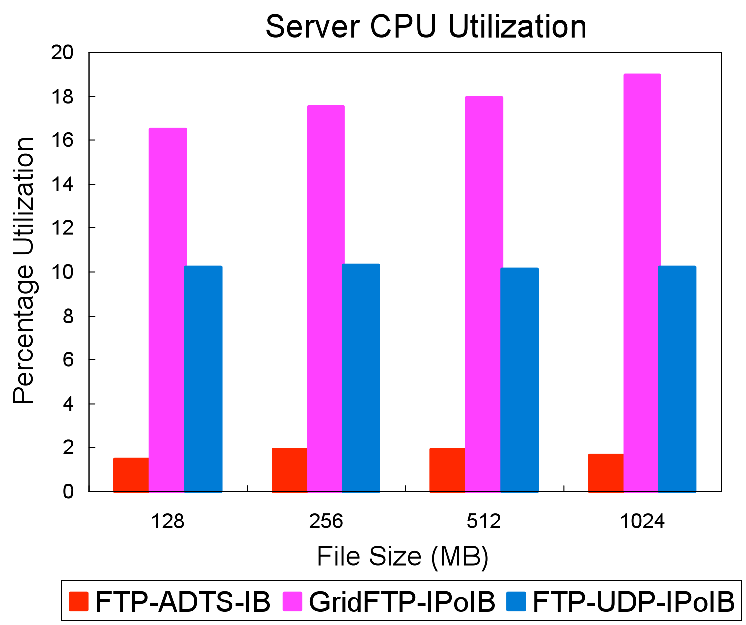
Multiple Files Transfer Time

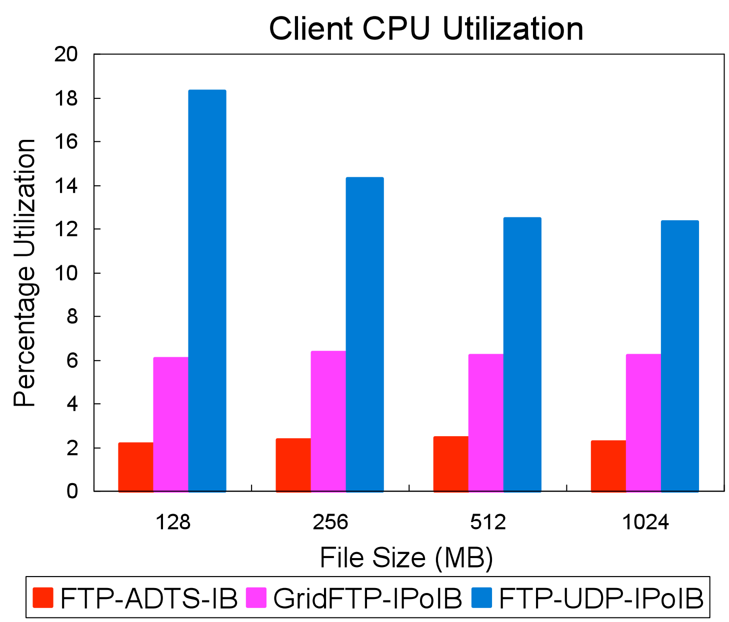
- Use a zipf file trace with an average file size of 66 MB
- Replicate this trace from one node in cluster A to another node in cluster B



- FTP-ADTS speeds up the replication by up to 65%
- Performance degradation at large network delay due to a lot of small sized files in zipf trace

CPU Utilization

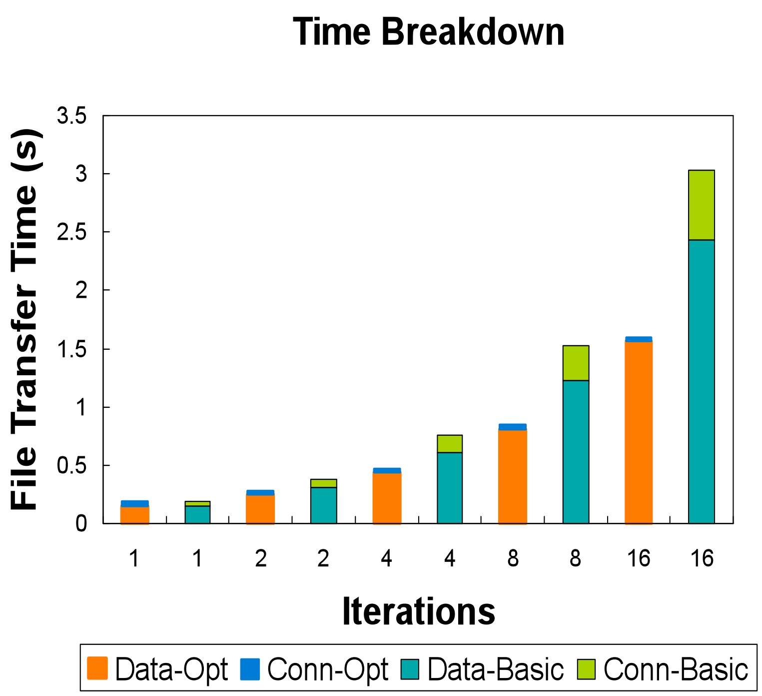




- CPU utilization for *put* operation
- FTP-ADTS has lowest CPU utilization on both server and client because of the zero-copy
- GridFTP has low CPU utilization on client due to the use of *sendfile* call; this cannot be applied to UDP

NETWORK-BASED COMPUTING LABORATORY

Benefits of Design Enhancements



- File transfer time is split into connection time and data transfer time
- Design enhancements for data communication improve the performance up to 55%
- Persistent session enhancement reduces the connection set up cost

OHIO STATE

Outline

- Introduction & Motivation
- Designing Zero-copy FTP Mechanism
- Experimental Results
- Conclusions & Future Work

Conclusions & Future Work

- Design a portable communication layer ADTS with optimizations including memory registration cache, persistent data sessions and pipelined data transfer
- Propose and design a novel FTP library (FTP-ADTS)
  - Efficient file transfer by using the zero-copy operations of modern interconnects
- FTP-ADTS achieves significantly better performance (by up to 95% improvement) at much lower CPU utilization in both IB LAN and WAN scenarios
- Future work
  - Study the performance of the new FTP mechanisms in data-center or file system applications
  - Explore other communication middleware and the impact of modern WAN technologies

Thank you

{laipi, subromon, narravul, mamidala, panda}
@cse.ohio-state.edu

NBC-LAB

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/