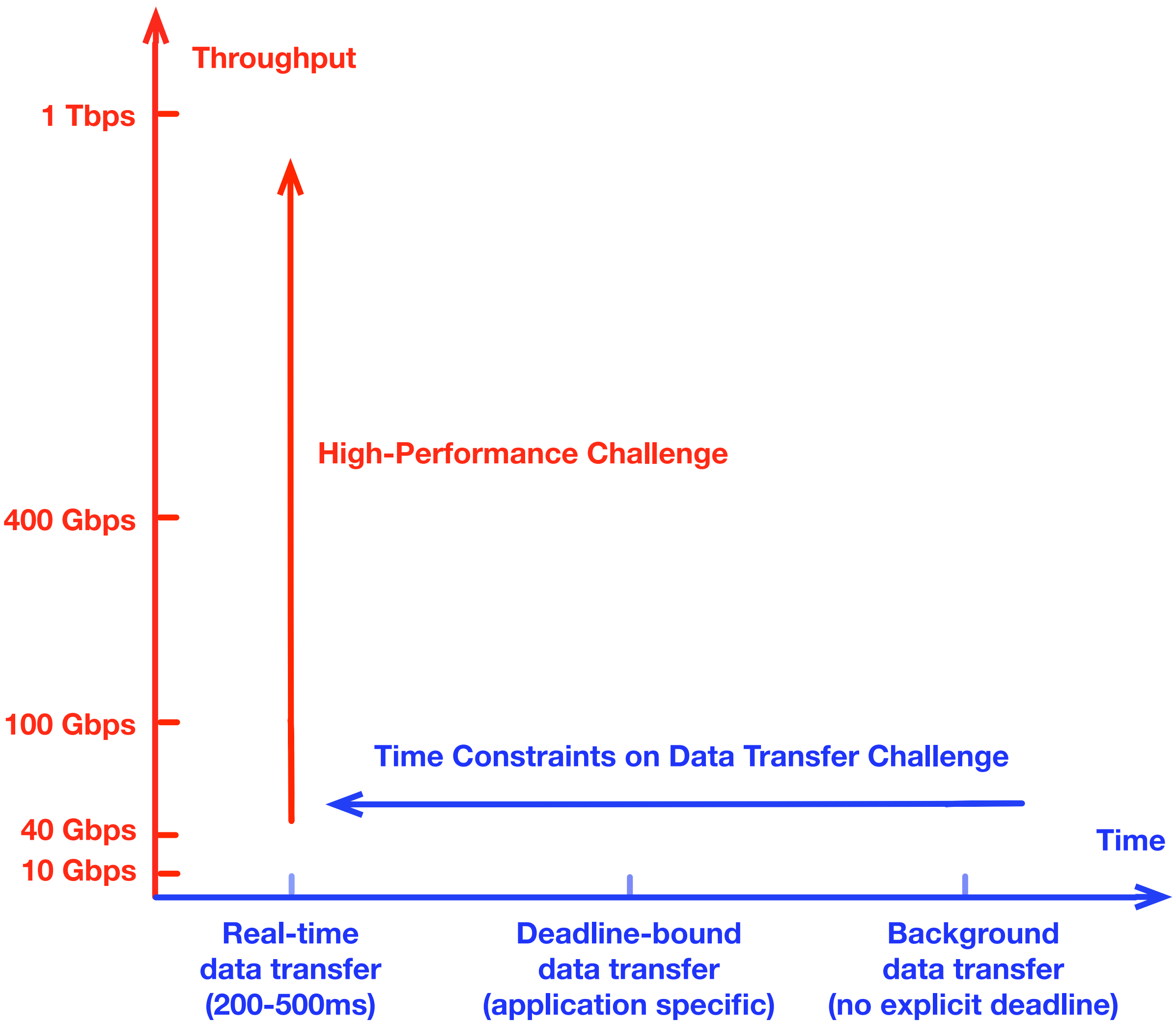
# mdtmFTP and Its Evaluation on ESNET SDN Testbed

Liang Zhang[*], Wenji Wu[*], Phil DeMar[*], Eric Pouyoul[+]

Fermilab[*], ESNET[+]

Funded by DOE/SC/ASCR Network Research  Program

# Big data transfer challenges



# Data transfer – state of the art

- Advanced data transfer tools and services developed
    - GridFTP, BBCP
    - PhEDEx, LIGO Data Replicator, Globus Online

- Numerous enhancements
    - Parallelism at all levels
        - Multi-stream parallelism
        - Multicore parallelism
        - Multi-path parallelism
    - Science DMZ architecture
    - Terabit networks

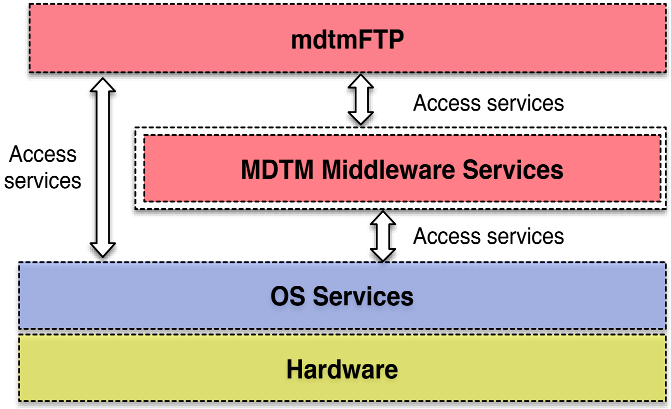
# Problems with existing data transfer tools

- Unable to fully exploit multicore hardware under default OS support, especially on NUMA systems

- Unable to effectively address the lots of small files (LOSF) problem
  - Either inefficient, or don’t scale well:
    - Pipelining
    - Concurrency
    - Tar-based solution

# mdtmFTP: a high-performance data transfer tool

- Pipelined I/O-centric design to streamline data transfer
- Multicore-aware data transfer middleware (MDTM) optimizes use of underlying multicore system
- Extremely efficient in transferring of Lots Of Small Files
- Various optimization mechanisms
  - Zero copy
  - Asynchronous I/O
  - Batch processing



A DOE/SC/ASCR-sponsored research project
Software is available at: http://mdtm.fnal.gov

# A pipelined I/O centric design - 1

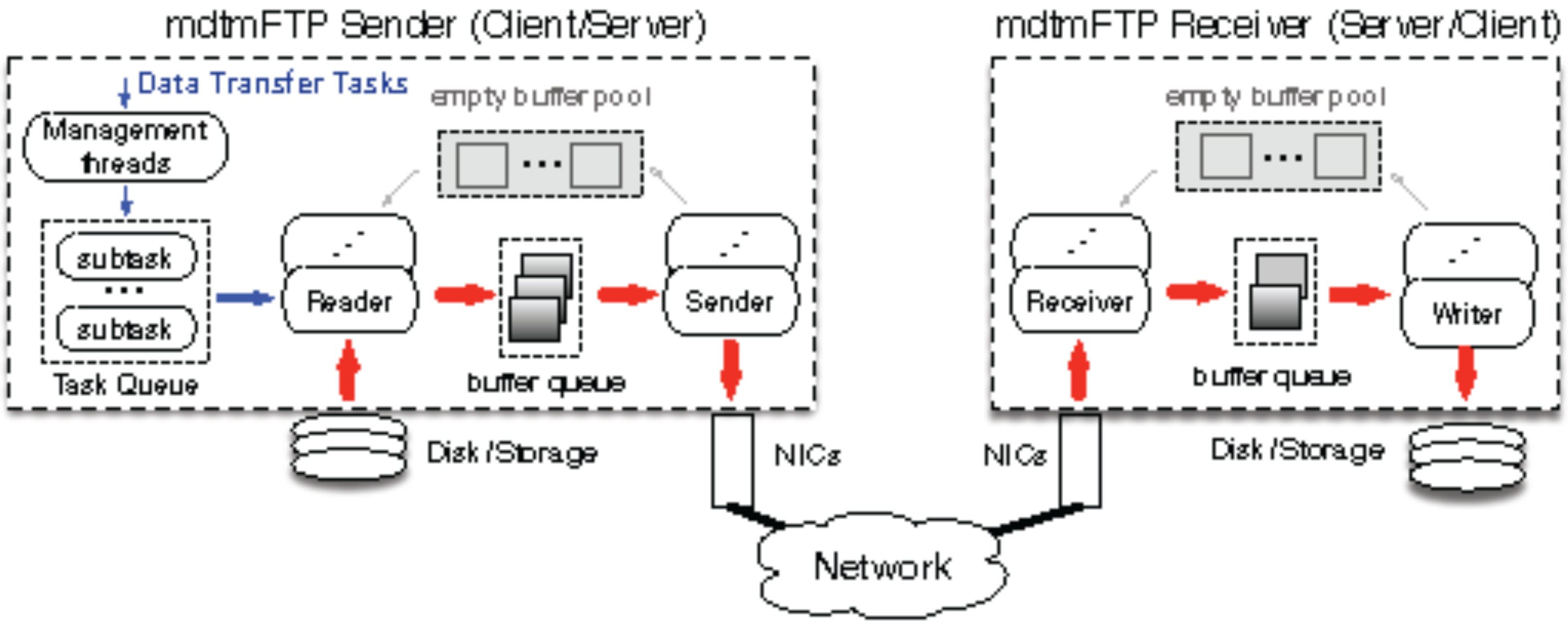
- Dedicated I/O threads to perform network & disk I/O operations in parallel
- MDTM middleware to schedule cores for I/O threads
  - Each I/O thread pinned to a core near the I/O device the thread uses
    - I/O locality
    - Core affinity for I/O operations
  - An I/O thread is typically dedicated with a single core
  - System zoning to avoid interference with other applications
    - MDTM-zone for mdtmFTP
    - Non-MDTM-zone for other applications

# A pipelined I/O centric design - 2

- Advanced data buffer mechanism to improve I/O performance
  - Pre-allocated data buffers to avoid costly memory allocation/deallocation in the critical data path of data transfer
  - Data buffers are pinned and locked to avoid memory swap and migration

# A pipelined I/O centric design - 3

- Data transfers are executed in a pipelined manner
- A data transfer task is split into multiple subtasks
- Subtasks are executed in parallel



# MDTM middleware – why?

- Default OSes cannot support data transfer tools on multicore systems well, especially NUMA systems:
  - Dynamic load balancing may degrade data transfer performance
    - Frequent thread migration
      - No core affinity for I/O operations
      - Inefficient use of cache
    - High-latency inter-node communication
  - Limited support for I/O locality
    - I/O throughputs can be significantly improved if I/O locality is available
  - Other applications' interferences
    - CPU, MEM, I/O

# What is MDTM middleware?

- A user-space resource scheduler
- Implemented as a system daemon
  - Periodically, collects, monitors, and caches information about the multicore system
    - Physical layout (e.g., NUMA topology)
    - Configurations
    - System loads
  - Upon requests, provide middleware services to mdtmFTP
    - Query service
    - Scheduling service

# MDTM middleware – key features

- Key Features
    - Computer system layout profiling
    - Real-time system status monitoring
        - CPU usage of each core
        - Memory load latency of each NUMA node
    - NUMA topology-based core scheduling
        - Support I/O locality
    - Support core affinity on I/Os
    - System zoning
        - MDTM Zone and Non-MDTM zone
    - Data buffer allocation and pinning capability
- MDTM middleware can be readily extended to support other types of applications
    - E.g., as a research tool to study advanced scheduling algorithms and policies on NUMA systems

# A large virtual file mechanism to address the LOSF problem

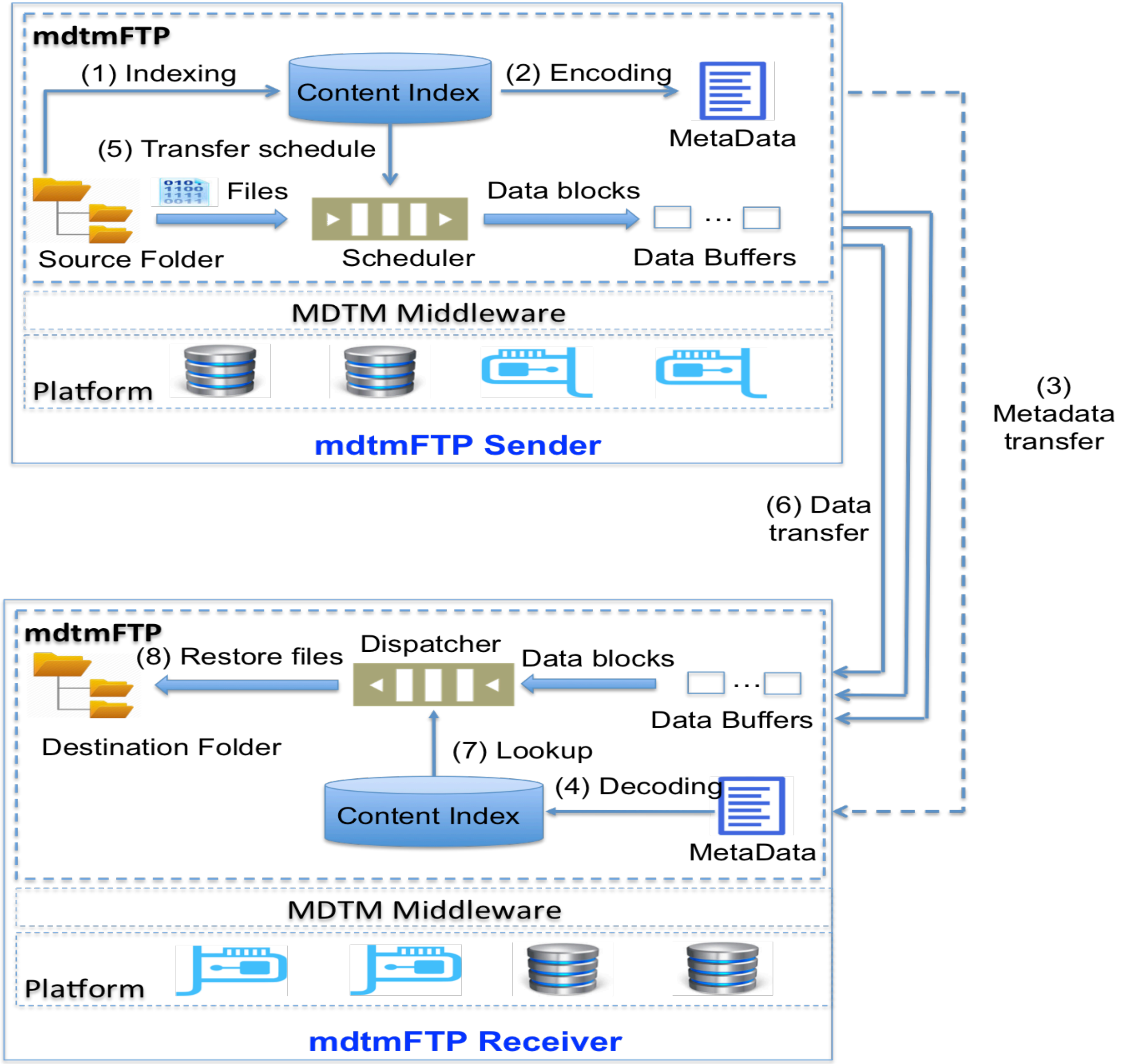
**Key idea:**

- Treat a dataset as a large “virtual file”.
- Each file in the dataset is treated as a file segment in the virtual file, and sequentially “added” to the virtual file.
- The virtual file is logically, instead of physically, created.
    - Different than Tar-based solutions
- The whole data set is transferred, continuously & seamlessly, as a single virtual file.
    - Different than GridFTP’s per-file-based mechanisms (e.g., pipelining, concurrency)

**Major advantages:**

- Avoid protocol processing on a per-file basis
- Allow for batch processing small files in the sender/receiver to optimize I/O performance

# Large virtual file transfer mechanism



# mdtmFTP evaluation @ ESnet testbed

- Test and evaluate mdtmFTP at WAN environment
- Test and evaluate mdtmFTP at high-performance DTN environment
- Compare mdtmFTP with other data transfer tools

# ESNET testbed - 1

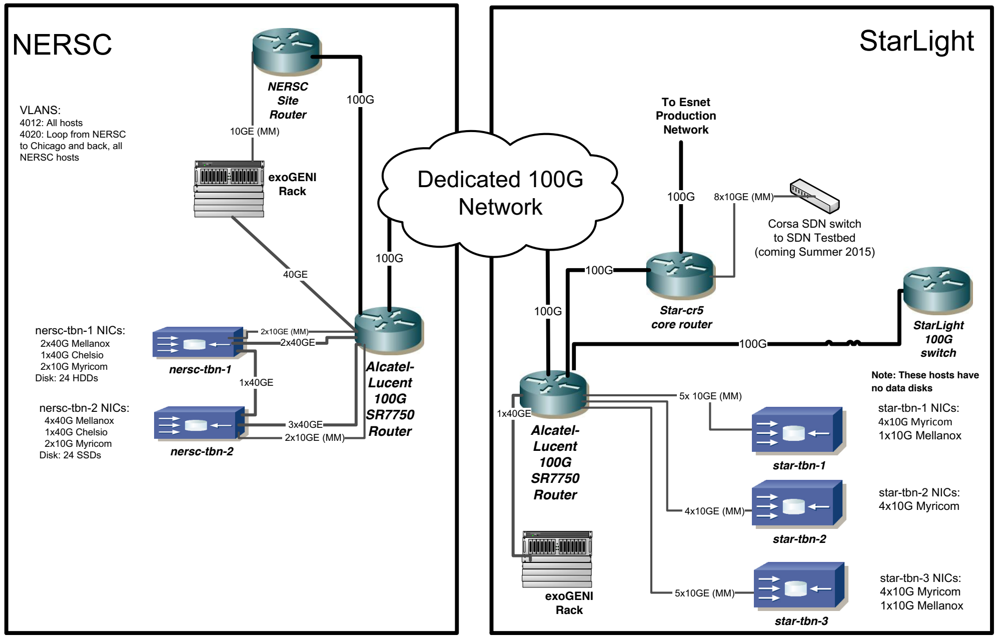
**nersc-tbn-1**

- 2xIntel HaswellXeon E5-2643 6 cores
- Motherboard: superMicro X10DRi (PCIe Gen3)
- 2x40G Mellanox NICs
- Support high performance I/O operation (Write)
    - An array of 24 HDDs

**nersc-tbn-2**

- 2xIntel HaswellXeon E5-2643 6 cores
- Motherboard: superMicro X10DRi (PCIe Gen3)
- 2x40G Mellanox NICs
- Support high performance I/O operation (Read)
    - An array of 12 SSDs

100G Component of Esnet SDN Testbed



VLANS:
4012: All hosts
4020: Loop from NERSC to Chicago and back, all NERSC hosts

nersc-tbn-1 NICs:
2x40G Mellanox
1x40G Chelsio
2x10G Myricom
Disk: 24 HDDs

nersc-tbn-2 NICs:
4x40G Mellanox
1x40G Chelsio
2x10G Myricom
Disk: 24 SSDs

Corsa SDN switch to SDN Testbed (coming Summer 2015)

Note: These hosts have no data disks

star-tbn-1 NICs:
4x10G Myricom
1x10G Mellanox

star-tbn-2 NICs:
4x10G Myricom

star-tbn-3 NICs:
4x10G Myricom
1x10G Mellanox

Data transfer:

- DTN “nersc-tbn-2” → “nersc-tbn-1”.
- 95ms RTT loop between nersc-tbn-1 and nersc-tbn-2.

Note: Thanks to ESNET Brian Tierney and Eric Pouyoul

# Evaluation methodology - 1

- Transfer data from nersc-tbn-2 to nersc-tbn-1
- Performance metric: Time-to-Completion
- Data transfer tool
  - mdtmFTP (developed by FNAL)
    - http://mdtm.fnal.gov
  - FDT (developed by CalTech)
    - http://monalisa.cern.ch/FDT/
  - BBCP (developed by SLAC)
    - https://www.slac.stanford.edu/~abh/bbcp/
  - GridFTP (developed by University of Chicago)
    - http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/

# Evaluation methodology - 2

- Transfer Mode
  - Client-Server data transfer
  - 3rd-Party data transfer
- Data Transfer Scenarios:
  - Large file transfer: Transferring a 100GB large file from nersc-tbn-2 to nersc-tbn-1.
  - Folder transfer 1: Transferring a folder that has 30 10G files from nersc-tbn-2 to nersc-tbn-1
  - Folder transfer 2: Transferring a Linux-3.18.21 folder from nersc-tbn-2 to nersc-tbn-1

# Evaluation Methodology - 3

- Data transfer tool configuration

| Data Transfer tools | # of Parallel Streams | Pipelining | Currency | TCP parameters |
|---|---|---|---|---|
| **FDT** | 4 | N/A | N/A | System configuration |
| **GridFTP** | 4 | -PP | -CC 8 | System configuration |
| **BBCP** | 4 | N/A | N/A | System configuration |
| **mdtmFTP** | 4 | N/A | 2 I/O threads | System configuration |

Note: when # of parallel streams > 4, data transfer performance has negligible changes

# Result – Client/Server

| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 74.18 | 79.89 | 91.18 | Poor performance |

**Larger file data transfer – 1 x 100G (Smaller is better)**

| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 192.19 | 217 | 320.17 | Poor performance |

**Folder data transfer – 30 x 10G (Smaller is better)**

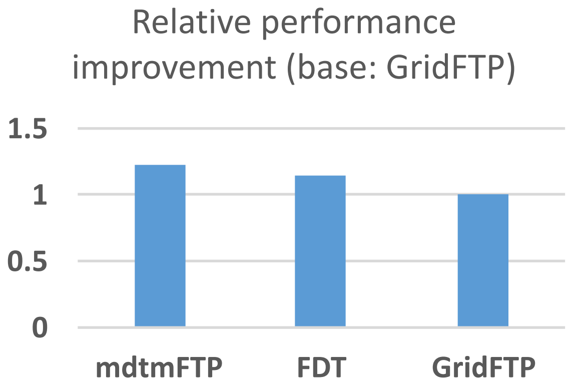
| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 10.51 | - | 1006.02 | Poor performance |

**Folder data transfer – Linux 3.12.21 (Smaller is better)**

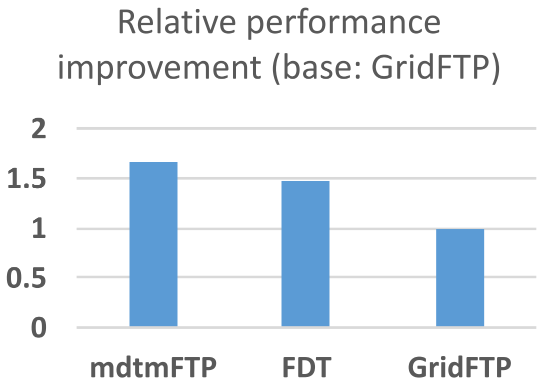
Note 1: "-" indicates inability to get transfer to work

Note 2: BBCP performance is very poor, we do not list its results here

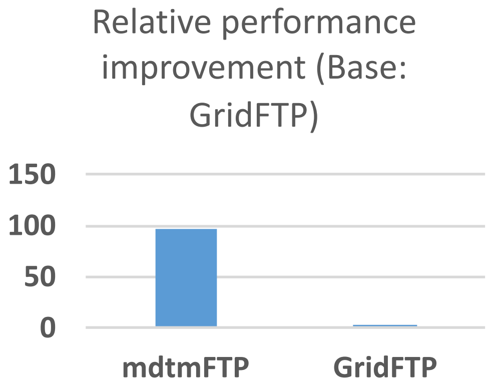
# Result – Client/Server



Large File Data Transfer (1x100G)



Folder Data Transfer (30x10G)



Folder Data Transfer (Linux 3.12.21)

$$\text{Relative performance improvement (base: GridFTP)} = \frac{\text{GridFTP's Time-to-Completion}}{\text{other tools' Time-to-Completion}}$$

Note: Larger is better

# Result – 3rd party data transfer

| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 34.976 | N/A | 106.84 | N/A |

**Larger file data transfer – 1 x 100G (Smaller is better)**

| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 95.61 | N/A | - | N/A |

**Folder data transfer – 30 x 10G (Smaller is better)**

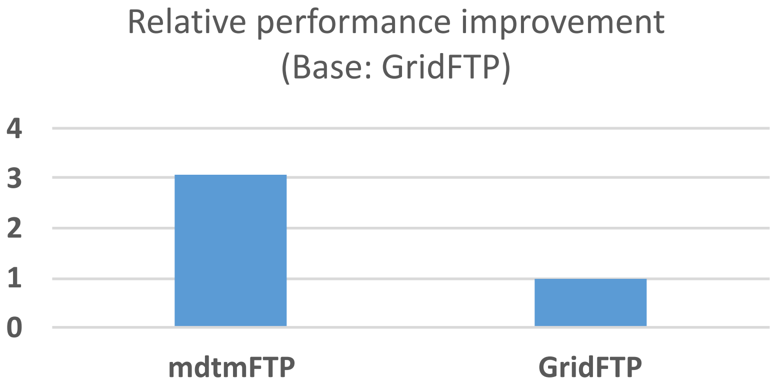
| | mdtmFTP | FDT | GridFTP | BBCP |
|---|---|---|---|---|
| Time-to-Completion (seconds) | 9.68 | N/A | - | N/A |

**Folder data transfer – Linux 3.12.21 (Smaller is better)**

Note 1: “-” indicates inability to get transfer to work

Note 2: : BBCP and FDT support 3rd party data transfer. But BBCP and FDT Couldn’t run 3rd party data transfer on ESNET testbed due to testbed limitation

# Result – 3rd party data transfer



Large File Data Transfer (1x100G)

$$\text{Relative performance improvement (base: GridFTP)} = \frac{\text{GridFTP's Time-to-Completion}}{\text{other tools' Time-to-Completion}}$$

Note: Larger is better

# Summary

- mdtmFTP is a high-performance data transfer tool
  - Pipelined I/O-centric design to streamline data transfer
  - Multicore-aware data transfer middleware (MDTM) optimizes use of underlying multicore system
  - Extremely efficient in transferring of Lots Of Small Files
- Evaluations show that mdtmFTP can achieve higher performance than existing data transfer tools.

# Acknowledgement

We would like to thank Brian Tierney who contributed to mdtmFTP evaluation @ ESnet testbed