

The Multicore-aware Data Transfer Middleware (MDTM) Project

MDTM Research Team

Wenji Wu (PI); **P. DeMar**, L. Zhang (FNAL)
Dantong Yu (Co-PI), T. Li, Y. Ren, S. Jin (BNL)

CHEP 2015

April 14, 2015

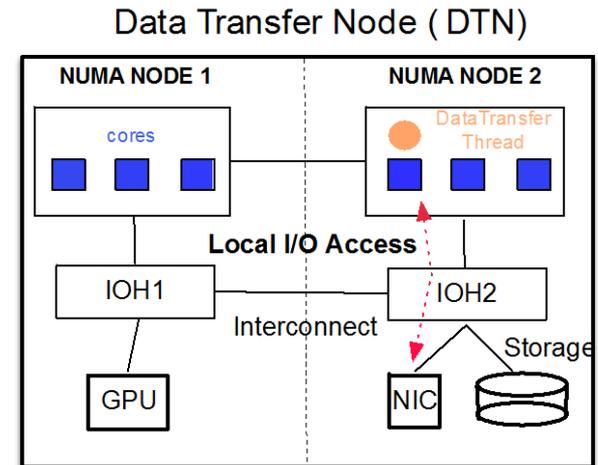
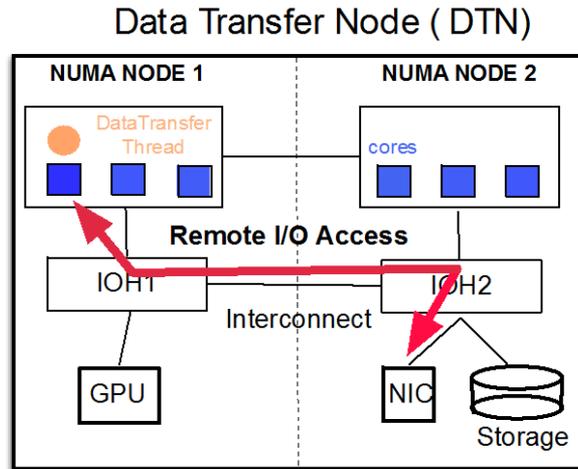
Problem Space

Multicore/manycore has become the norm for high-performance computing.

Existing data movement tools are still limited by major inefficiencies when run on multicore systems

These inefficiencies will ultimately result in performance bottlenecks on end systems. Such bottlenecks also impede the effective use of advanced high-bandwidth networks.

A simple inefficiency case ...



Scheduling without I/O locality

Scheduling with I/O locality

General-purpose OSES have only limited support for I/O locality!

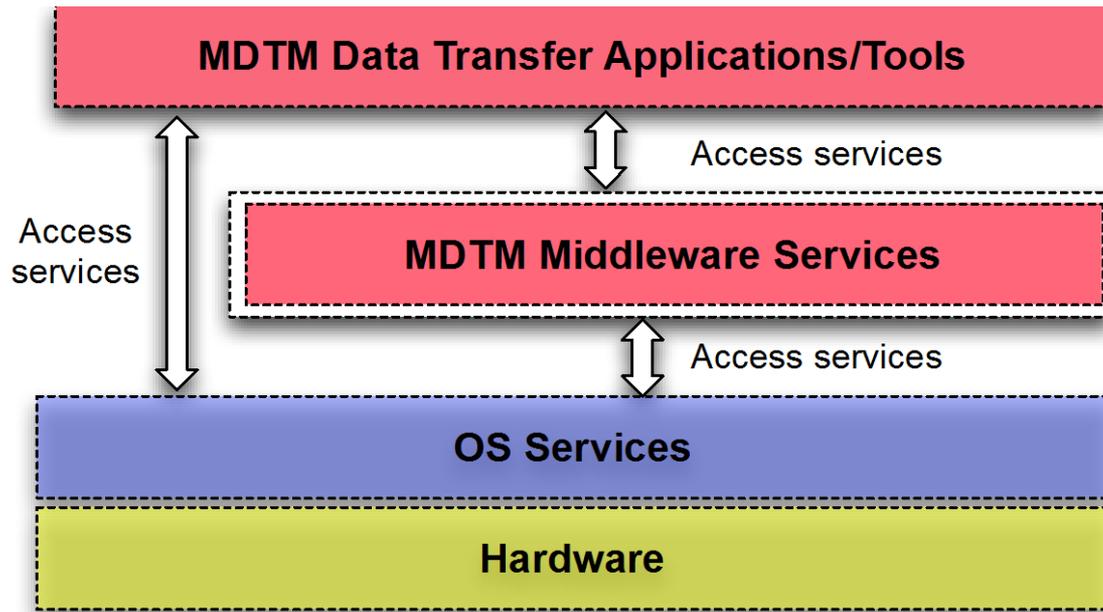
How can we improve?

Our solution

- **The Multicore-aware Data Transfer Middleware (MDTM) Project**
 - Collaborative effort by Fermilab and Brookhaven National Laboratory
 - Funded by DOE's Office of Advanced Scientific Computing Research (ASCR)
 - A three-year research project

MDTM aims to accelerate data movement toolkits on multicore systems

MDTM Architecture



MDTM data transfer application (BNL)

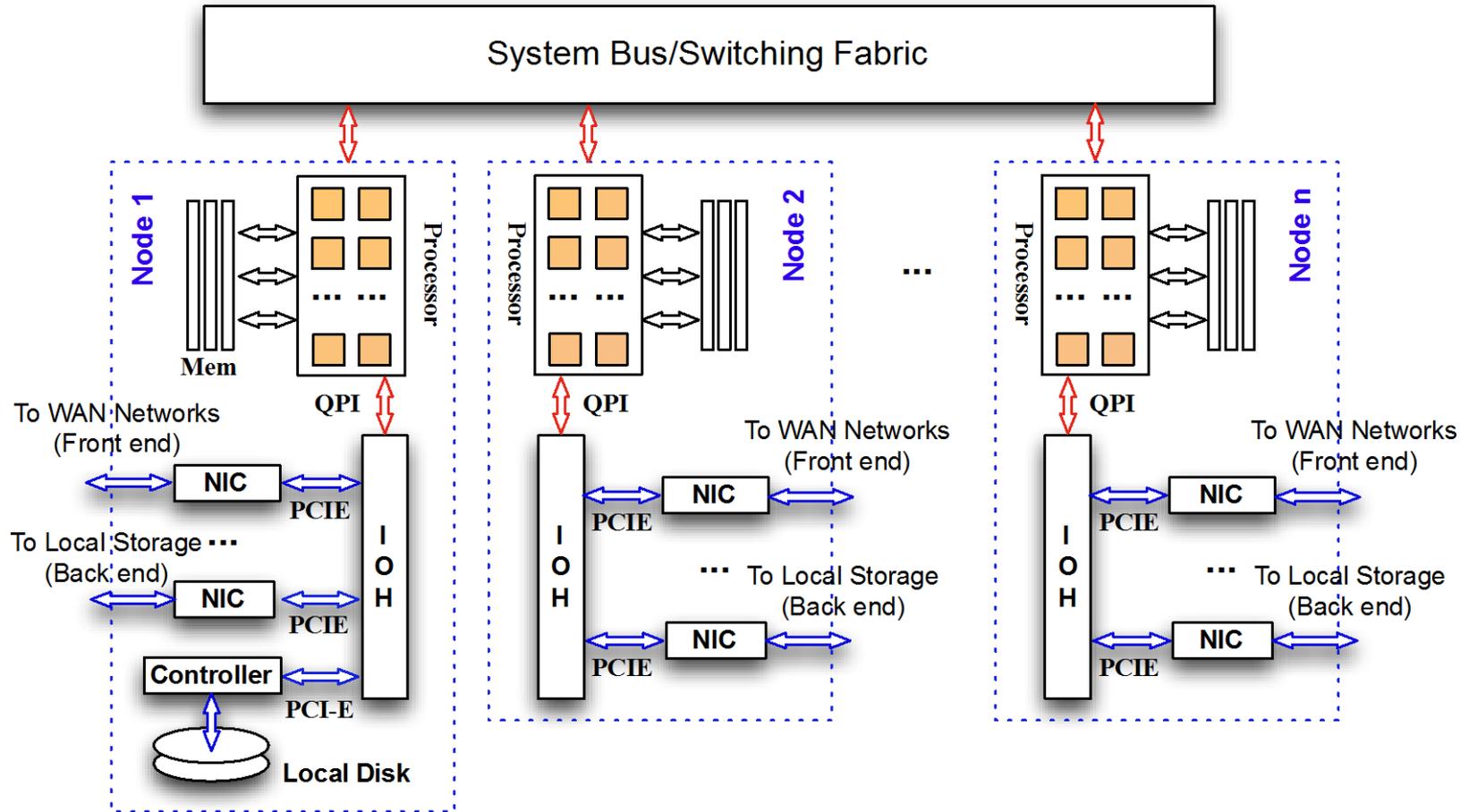
- Adopts an I/O-centric architecture that uses dedicated threads to perform network and disk I/O operations

MDTM middleware services (FNAL)

- Harness multicore parallelism to scale data movement toolkits on host systems

MDTM Hardware Platform

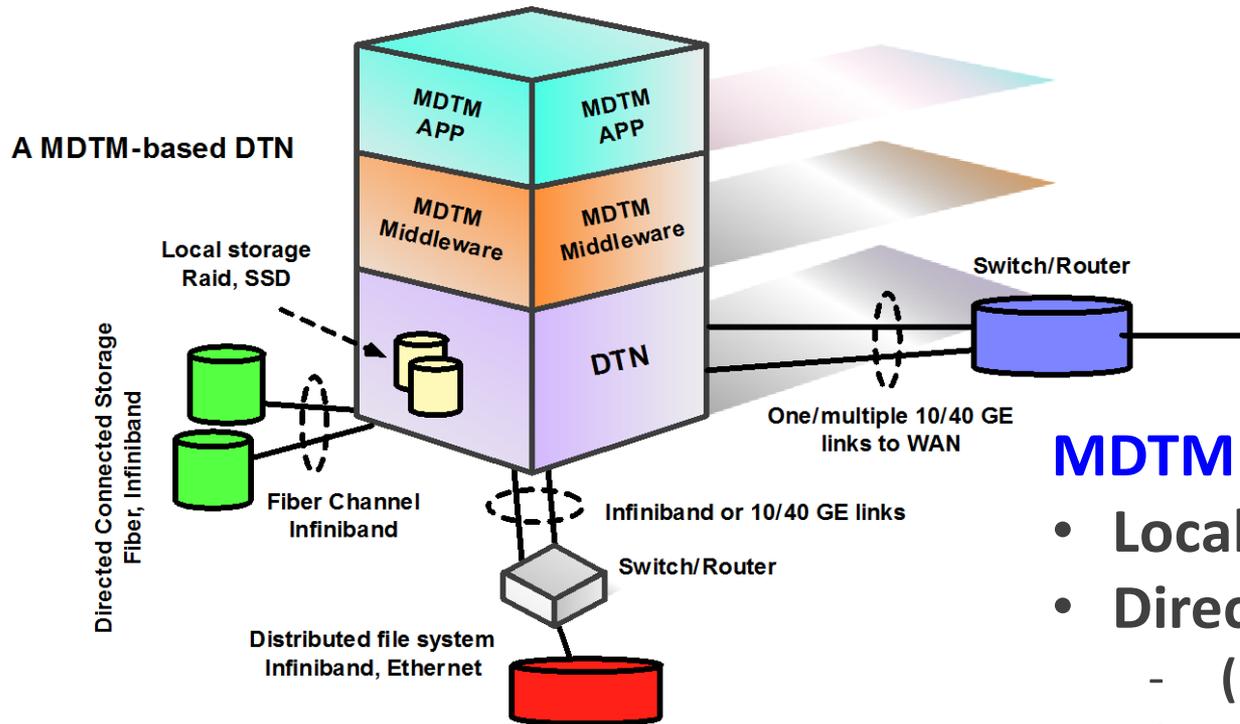
MDTM targeted for Data Transfer Nodes (DTN)



Each DTN features one or multiple NUMA nodes.

Each NUMA node features one or more processors of multiple cores.

MDTM-based DTN Storage & Networking Architecture



MDTM Storage Architecture

- Local storage (Raid, SSD)
- Directed connected storage
 - (FC, IB)
- Distributed file system
 - (IB, 10/40 GE)

MDTM Networking Architecture

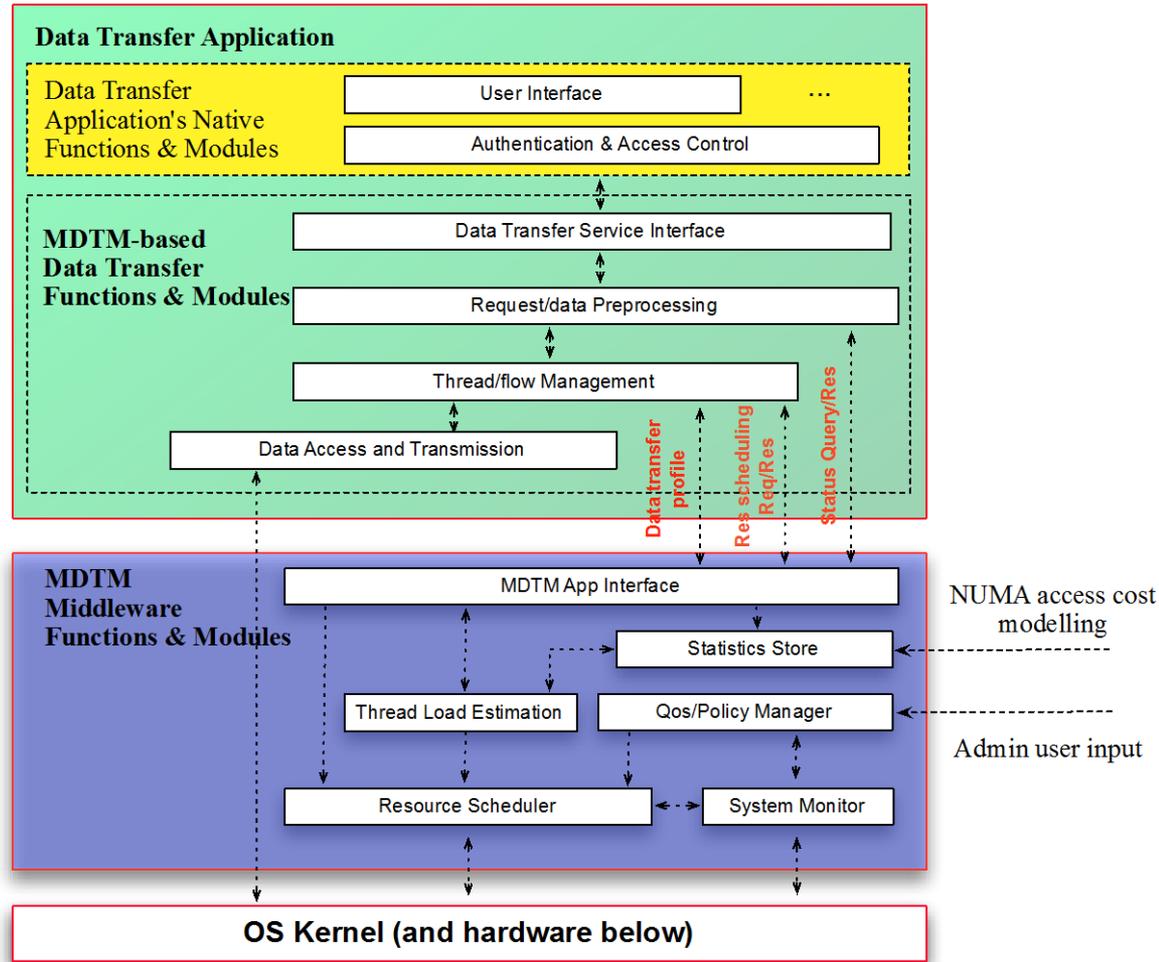
- One or more 10/40GE NICs for WAN
- One or multiple LAN links for storage access
 - Via 10/40 GE NICs, IB adaptors, FC adaptors

MDTM Software

MDTM Logical Functions & Modules

I/O-Centric architecture
Parallel data transfer
Data layout preprocessing

Data flow-centric scheduling
NUMA-awareness scheduling
I/O locality optimization
Maximizing parallelism



How does MDTM works?

An MDTM application spawns three types of threads

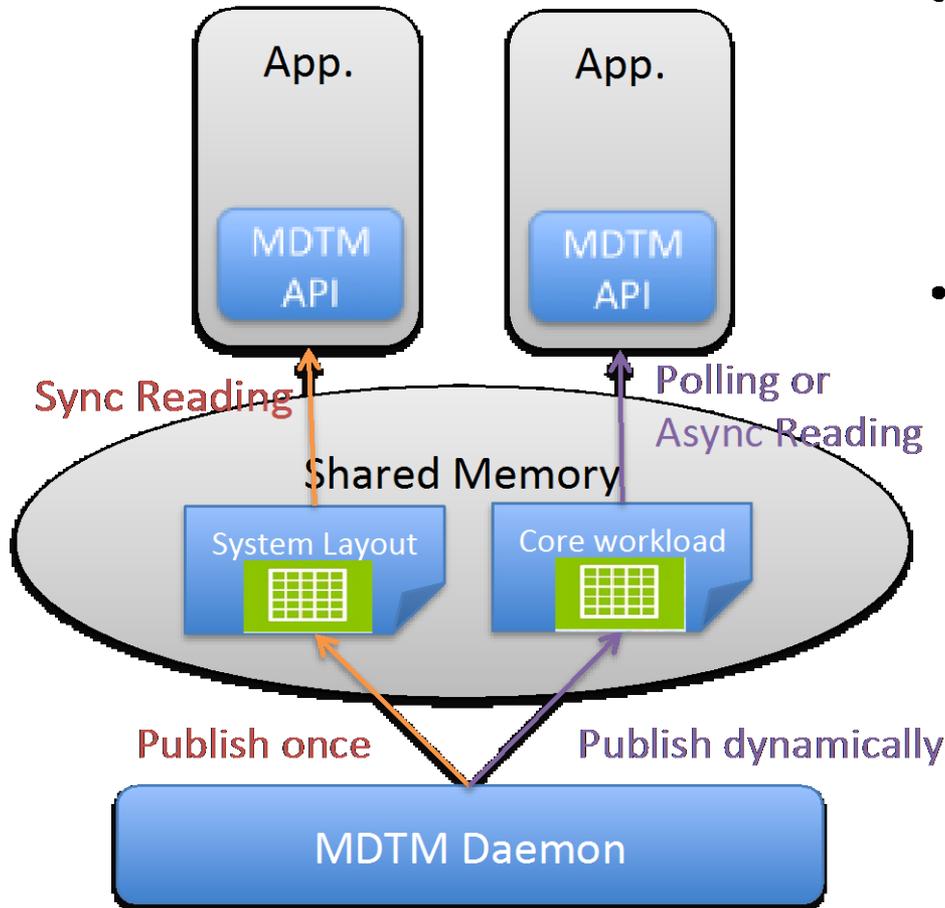
1. Dedicated network I/O threads to send/receive data
2. Dedicated disk/storage I/O threads to read/write disks/storages
3. Management threads for user requests & management functions

The application accesses MDTM middleware services explicitly via APIs

MDTM middleware daemon is launched, which supports two types of services

1. Query service allows MDTM APP to access system configuration and status
2. Scheduling service assigns system resources based on requirements of data transfer applications

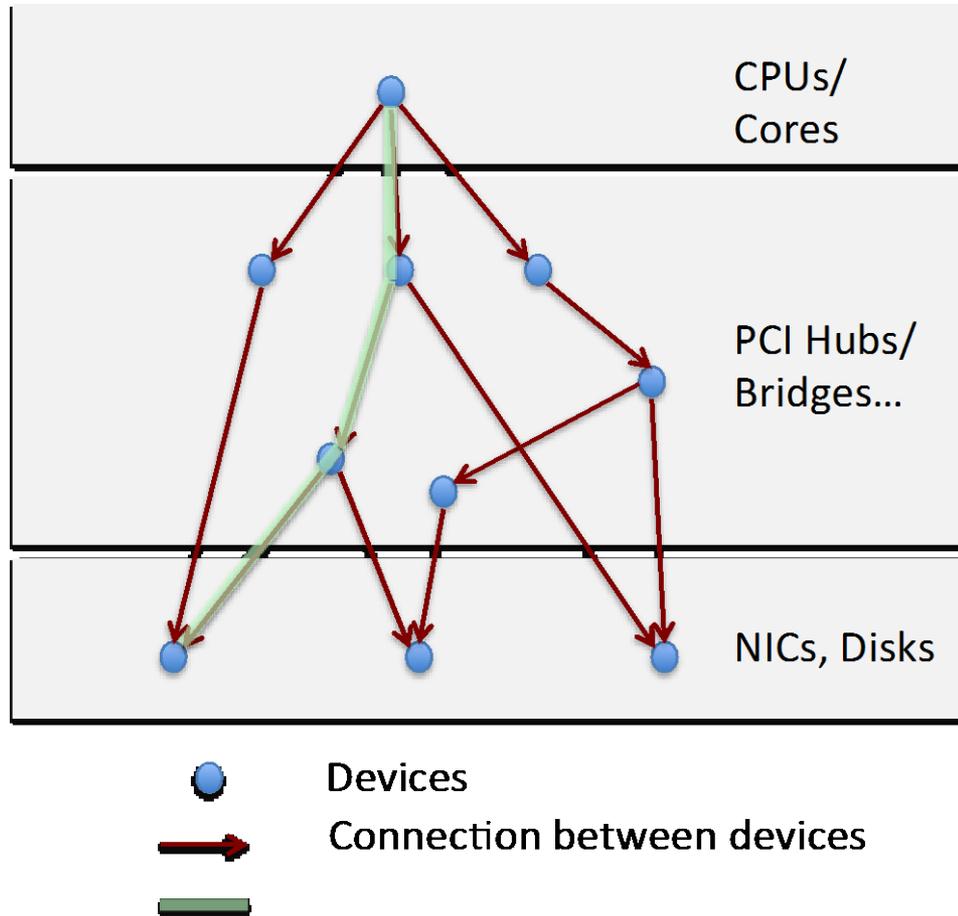
How does MDTM work? (Interaction)



- The data can be **static** like *System Layout*, which is published once and the APIs can retrieve it by calling the synchronous read function.
- The data can be **dynamic** like *Core Workload*, which is published periodically. Our implementation provide two ways to handling this case: **polling** and **async reading**:
 - Polling: use synchronous read function many times in case of data changes.
 - Asynchronous Reading: register the event of data change; upon event occur, calling callback function to invoke a read.

MDTM IPC Design

How does MDTM work? (Middleware)



- Each connection associated with a cost value which reflects scheduling factors like distance, traffic throughput and etc.
- Each node contains a cost table to its neighbors
- Applying Dijkstra's Algorithm to find the lowest cost path from CPU node to the NIC/Disk node in question
- pick up the core associated to the lowest cost path
- Pros and Cons
more extensive system picture;
scalable; dynamic; more complicated data structure

MDTM Middleware Scheduling

MDTM App Year-1/2 R&D Areas

- Capacity-based resource pre-allocation and management
- Support for both pipelining and striping modes
- Request preprocessing
 - Task grouping for affinity binding
 - Task sorting for I/O Optimization
 - Improve performance on different storage media
- Progress report for data transfer jobs

MDTM Middleware Year-1/2 R&D Areas

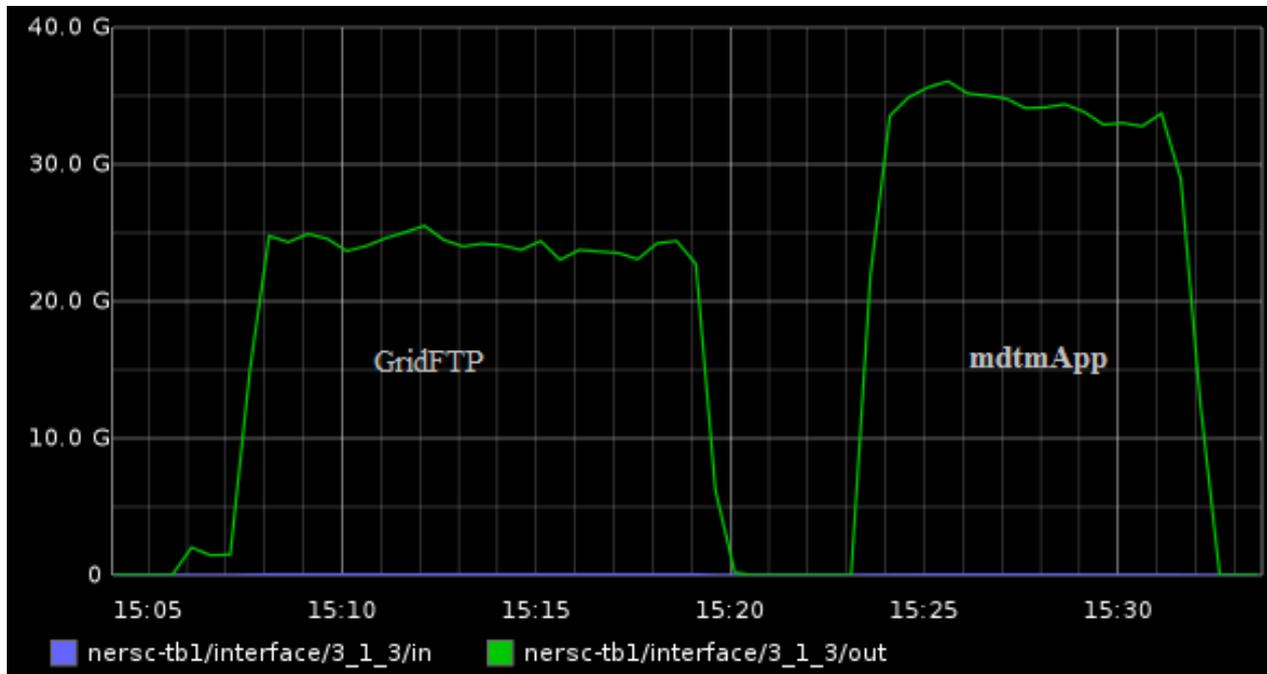
- Multicore system profiling
- Topology-based resource scheduling
- Interrupt affinity for network I/O
- Web-based monitoring and management

MDTM Current Status & Summary

- In year two of a three year R&D project
 - Long list of enhancements targeted in year three
- Alpha version of integrated (app & Middleware) prototype
- Beta version targeted for availability in August

MDTM Early Test Evaluation

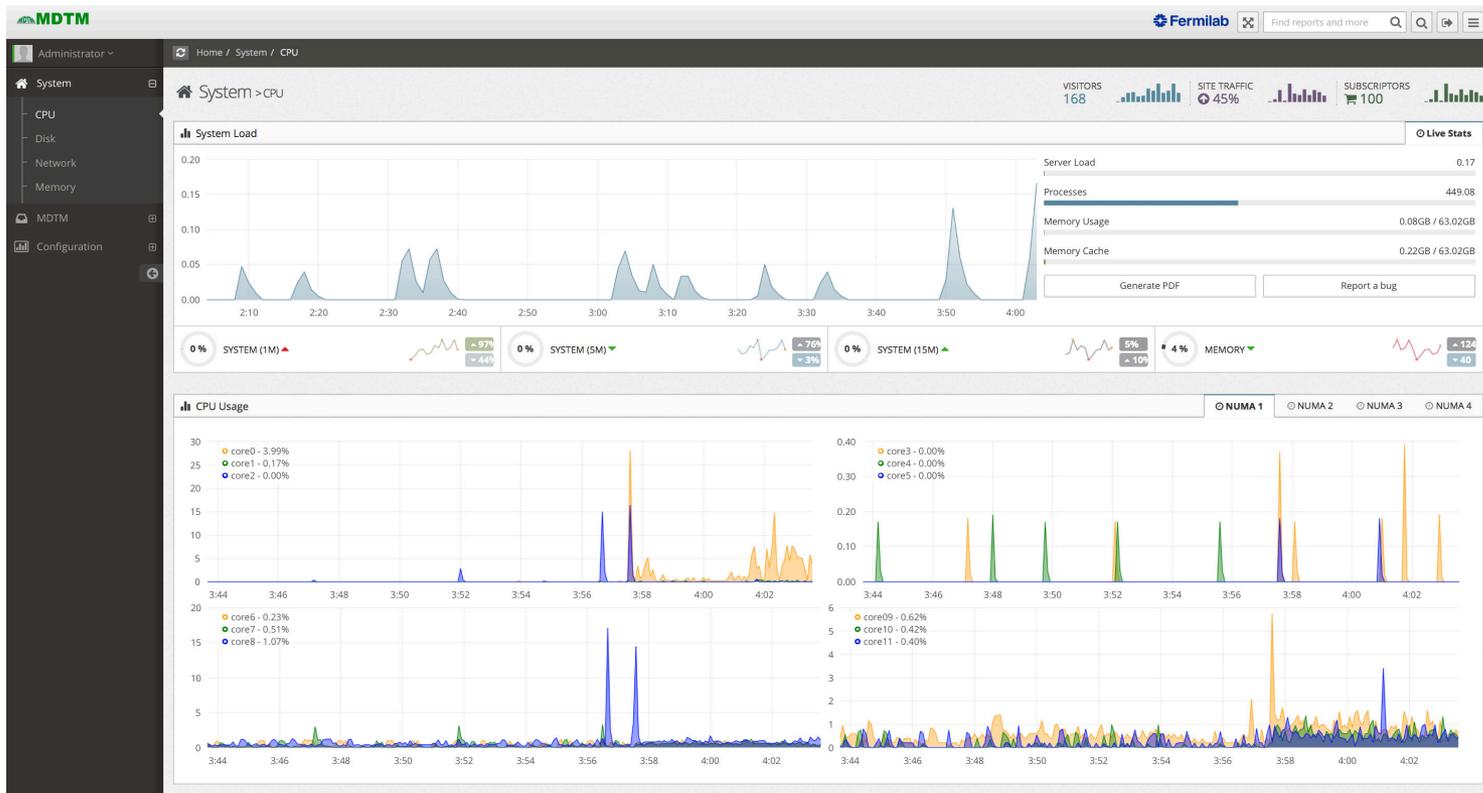
- Wide-area network links, end-to-end tests
- Performance comparison with GridFTP, bandwidth captured at ESnet's edge router



Parallel large file transfers(16 streams, 2TB, 8MB blocks), from SSDs to /dev/null, with 40Gbps links and 50Gbps aggregate disk bandwidth

MDTM Web-Based Monitor & Management

- MDTM monitor and management enable global access and management of the MDTM servers.

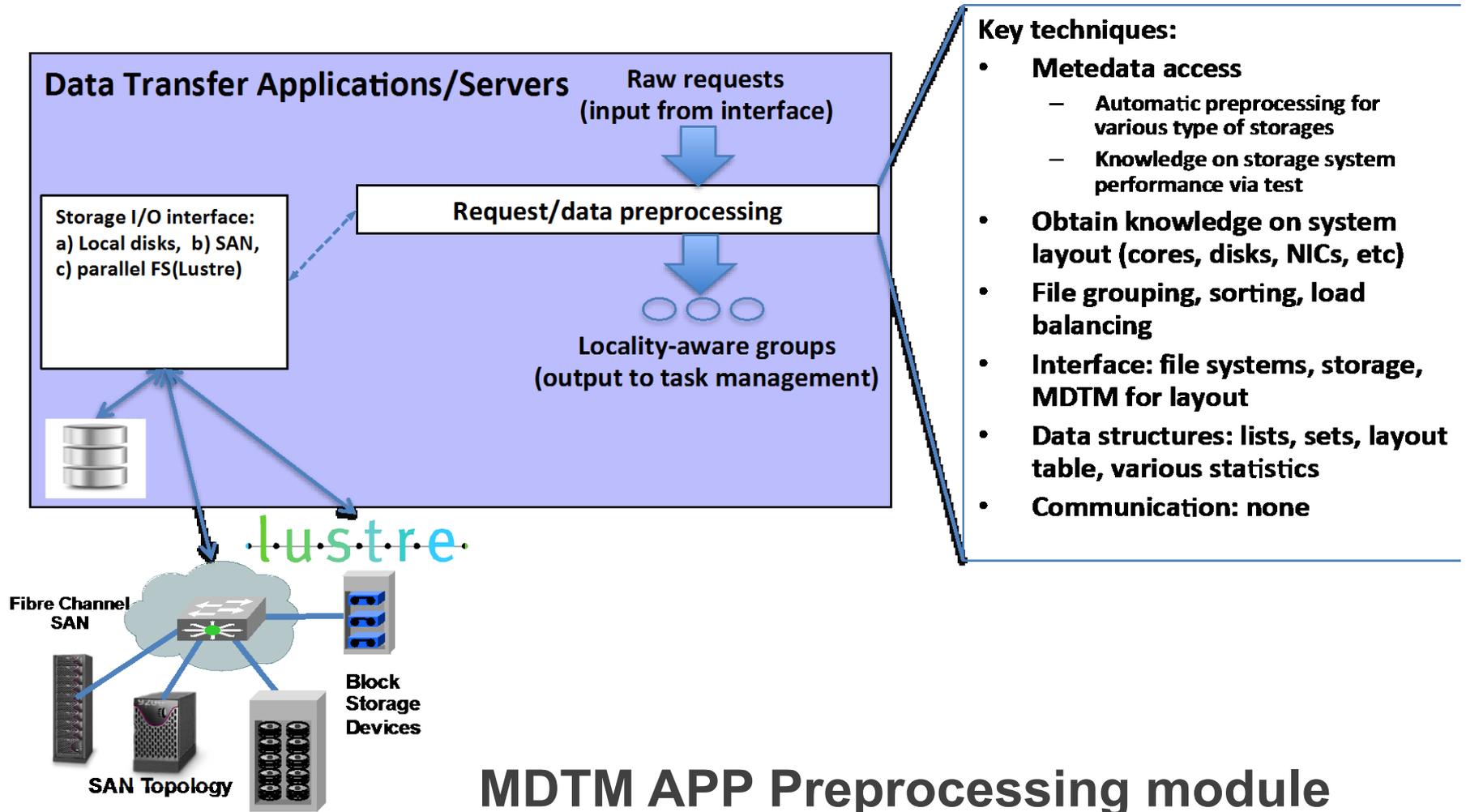


Extra Slides

MDTM Year-2 Research Areas

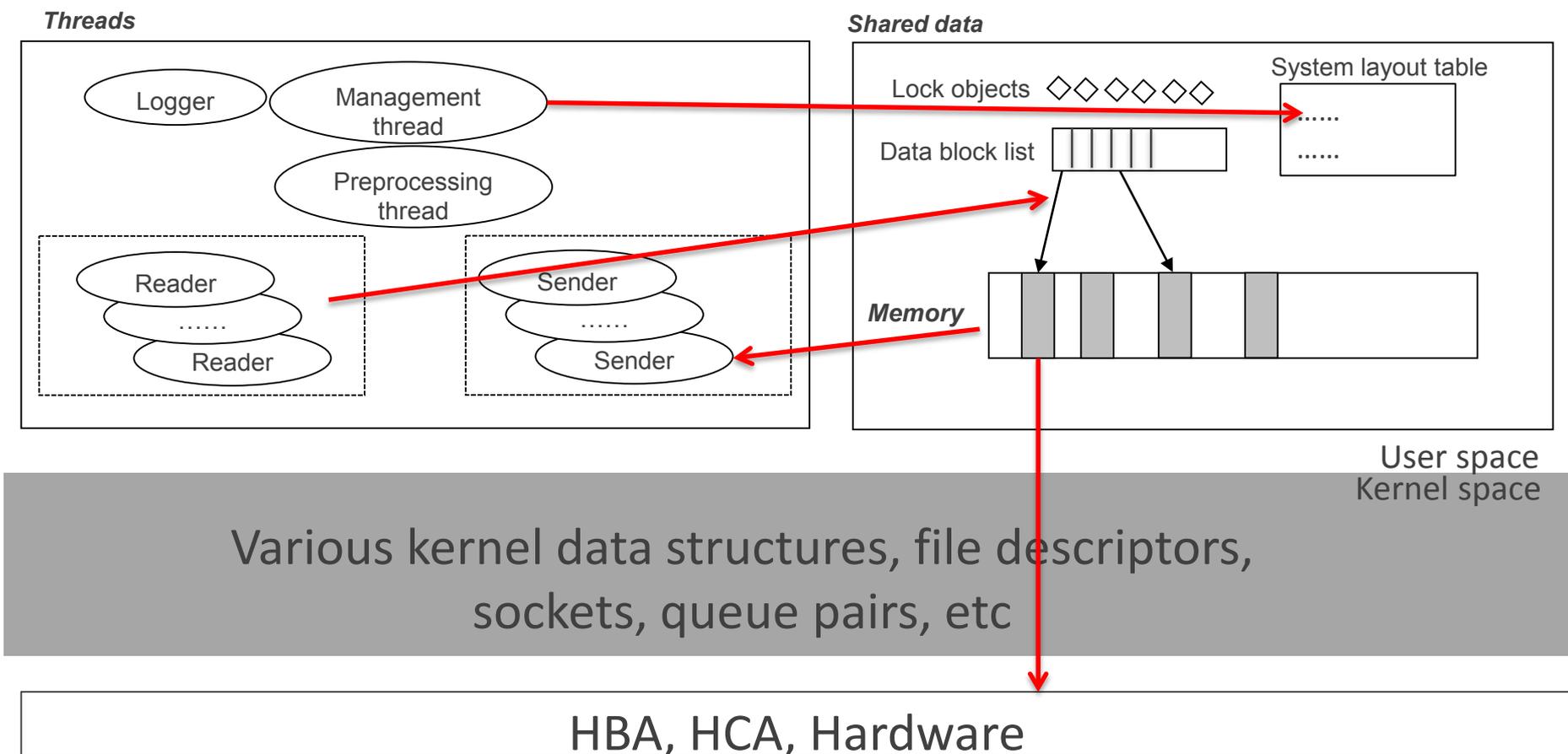
Requirements	Middleware (FNAL)	Applications (BNL)
Performance	<ul style="list-style-type: none">• Support NUMA-aware buffer pools• Automatic memory hotspots detection• System profiling• MDTM Scheduler• Disk/network performance optimization	<ul style="list-style-type: none">• Efficient thread/buffer management• High-performance event-driven architecture• Asynchronous I/Os to hide data access latency• Improved I/O scheduling• Improved request preprocessing
Reliability	<ul style="list-style-type: none">• Reliable, robust middleware services	<ul style="list-style-type: none">• High-performance, reliable data transfer protocol• Data transfer integrity check
Deployment	Security (Policy Enforcement) GSI, OpenID/FederatedID	<ul style="list-style-type: none">• Enhanced security features• Improved SSH• Automatic configuration file
User Interface	<ul style="list-style-type: none">• Automated config file generation• Web-based monitoring & management	<ul style="list-style-type: none">• Timely data transfer progress reporting

How does MDTM work? (MDTMApp)

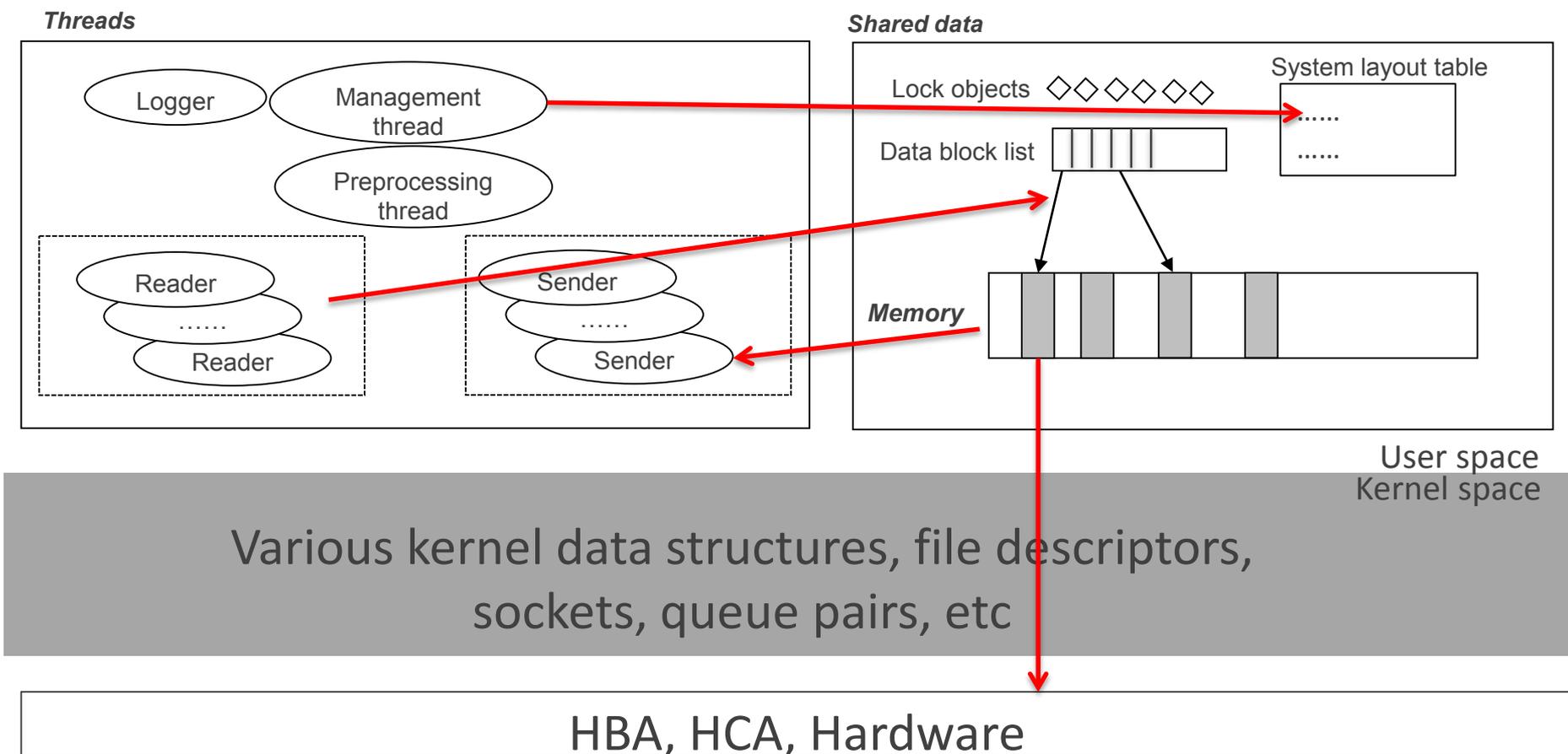


MDTM APP Preprocessing module

Data Access/Transmission Logic (Application memory layout)



Data Access/Transmission Logic (Application memory layout)



Key Techniques Used in MDTM App

- Meta Data Access
 - Automatic Preprocessing for various types of storages
 - Knowledge on storage system performance test
- Awareness of System Layout (cores, disks, NIC,etc)
- File Regrouping, File Sorting, Load Balance
- Interfaces: file systems, storage, MDTM for layout
- Software Design and Data Structures: Object-oriented, lists and sets, layout table, various statistics