Parallel Task-local I/O

25. April 2012, JICS/GRS Workshop on Large-scale Computer Simulation

Wolfgang Frings

w.frings@fz-juelich.de

Jülich Supercomputing Centre

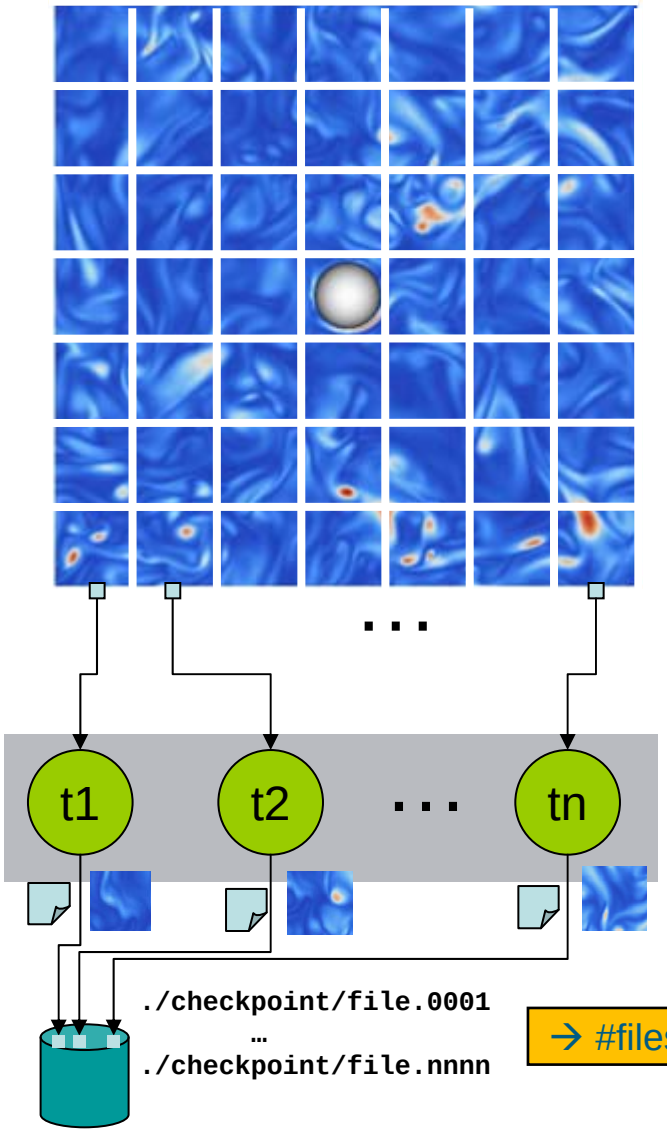
Felix Wolf

f.wolf@grs-sim.de

German Research School for Simulation Sciences

Mitglied der Helmholtz-Gemeinschaft

Parallel Task-local I/O

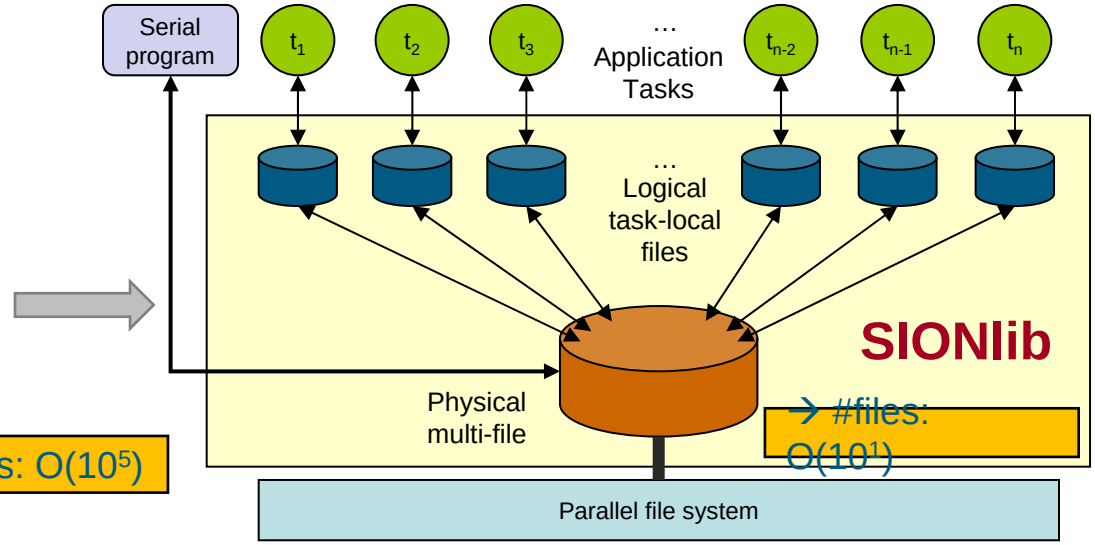


Usage Examples:

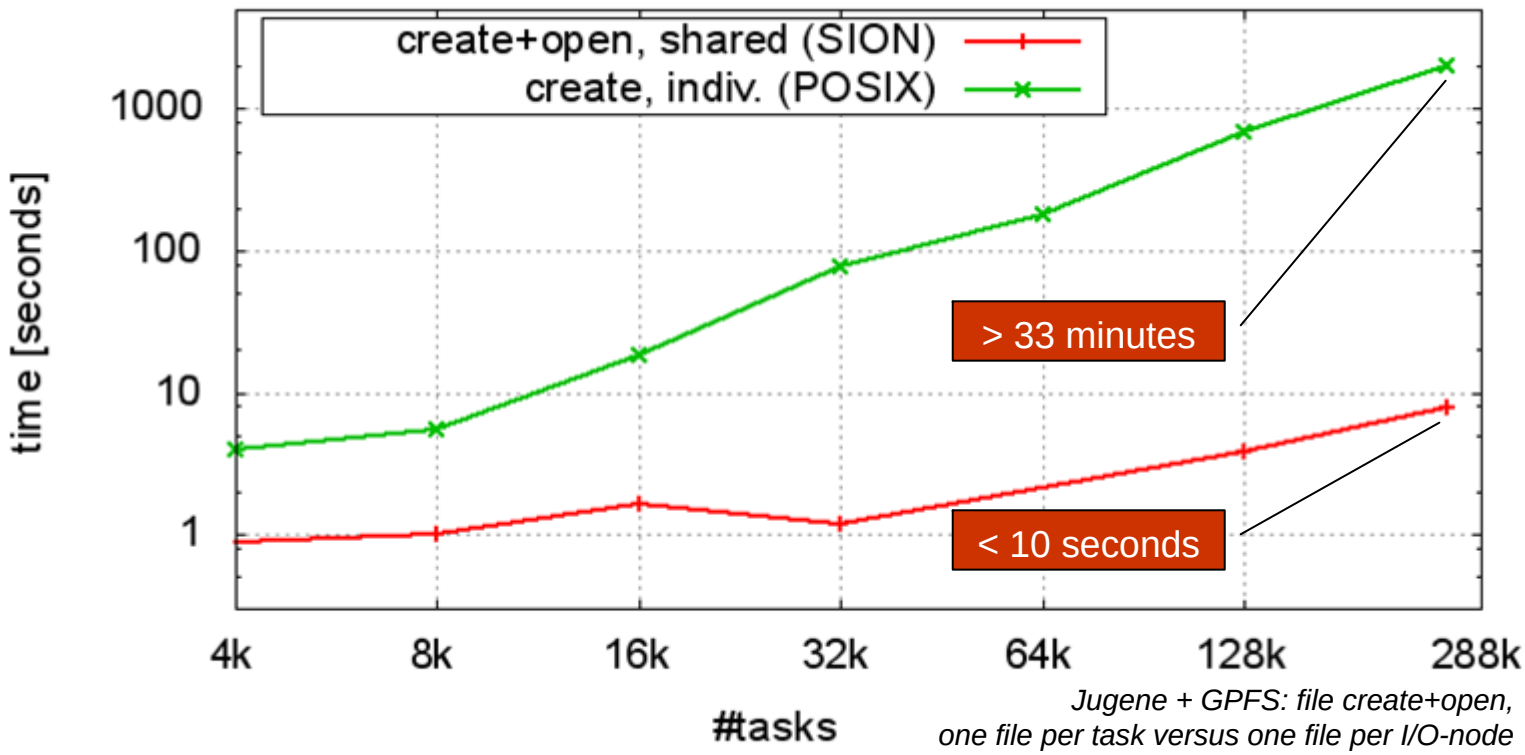
- Check-point files, restart files
- Result files, post-processing

Data types:

- Simulation data (domain-decomposition)
- Performance data (e.g. trace data of parallel performance tools)



The Showstopper for Task-local I/O: Parallel Creation of Individual Files



Jugene + GPFS: file create+open, one file per task versus one file per I/O-node

- Contention at node doing directory updates (directory meta-node)
- Pre-created files or own directory per task may help performance, but does not simplify file handling
- Complicates file management (e.g. archive)

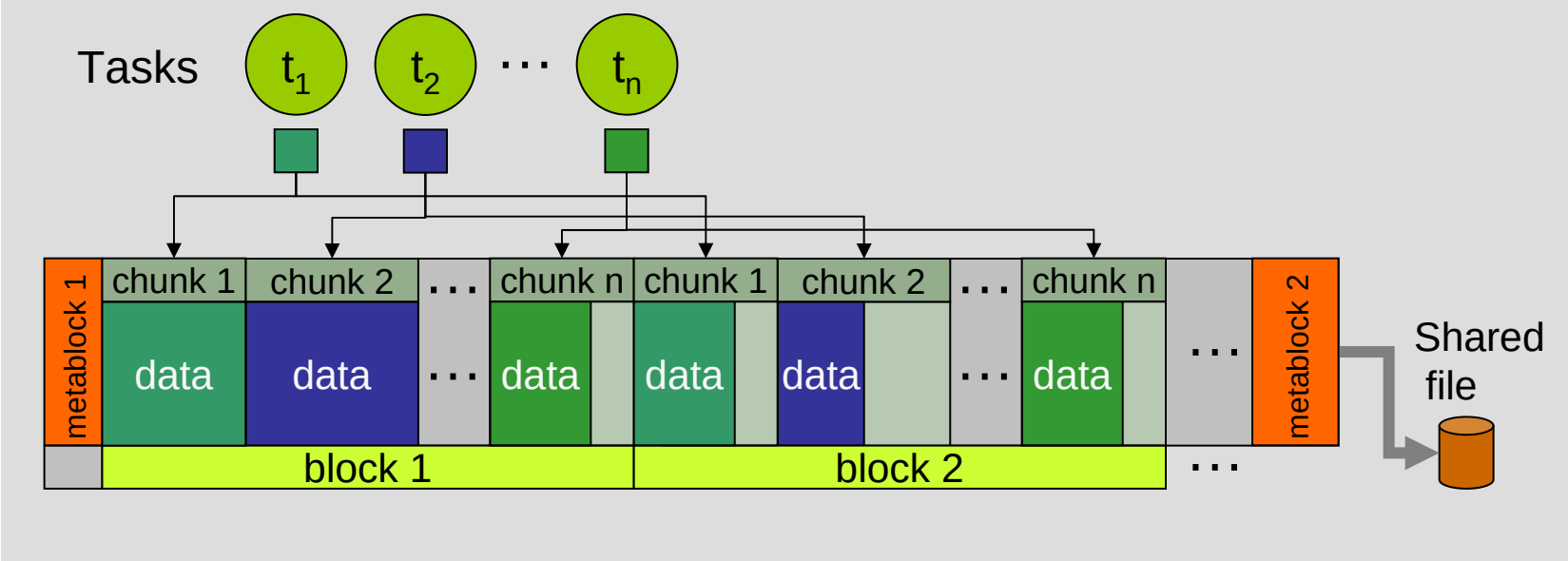
→ **shared files are mandatory**

Overview

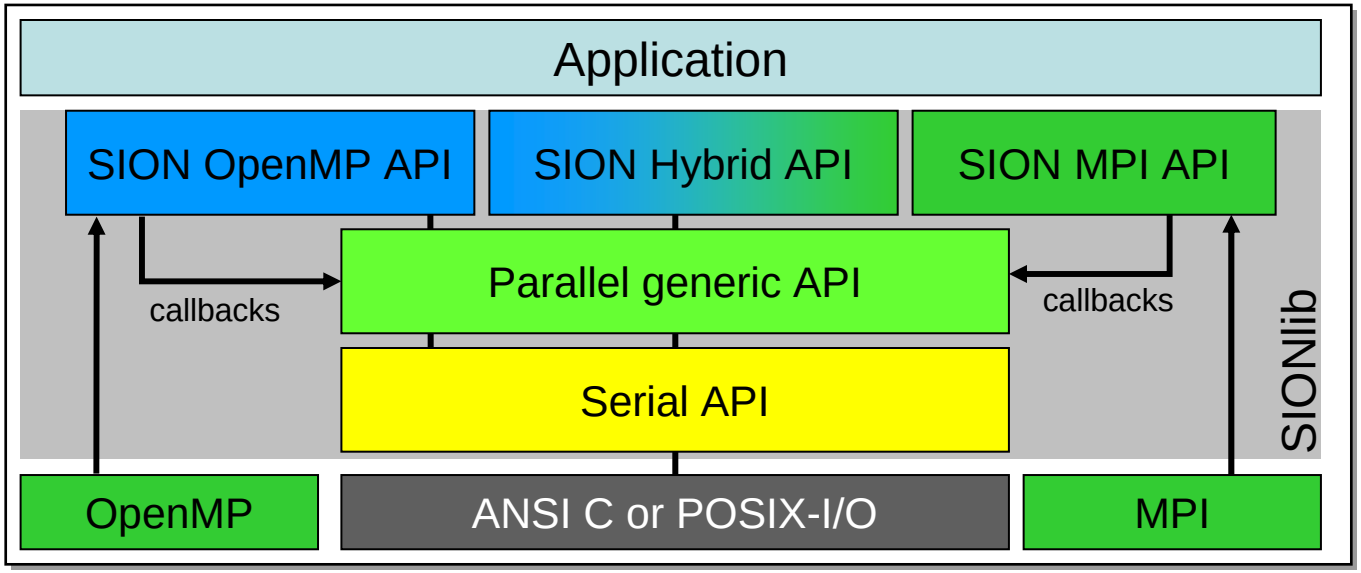
- Introduction
  - Parallel task-local I/O
  - Parallel file creation
- SIONlib
  - Introduction and features
- Scalability
  - Meta-data walls
  - Shared resources
  - SIONlib & task mapping
- Conclusion

SIONlib: Overview & File Format

- Self-describing container format for task-local binary data
- Meta data handling (offset and data size, big/little endian, …)
- Multiple chunks per task
- Automatic alignment to file system blocks
- Transparent support of **multiple physical files**
- File coalescing: support for collective I/O



SIONlib: Architecture & Example



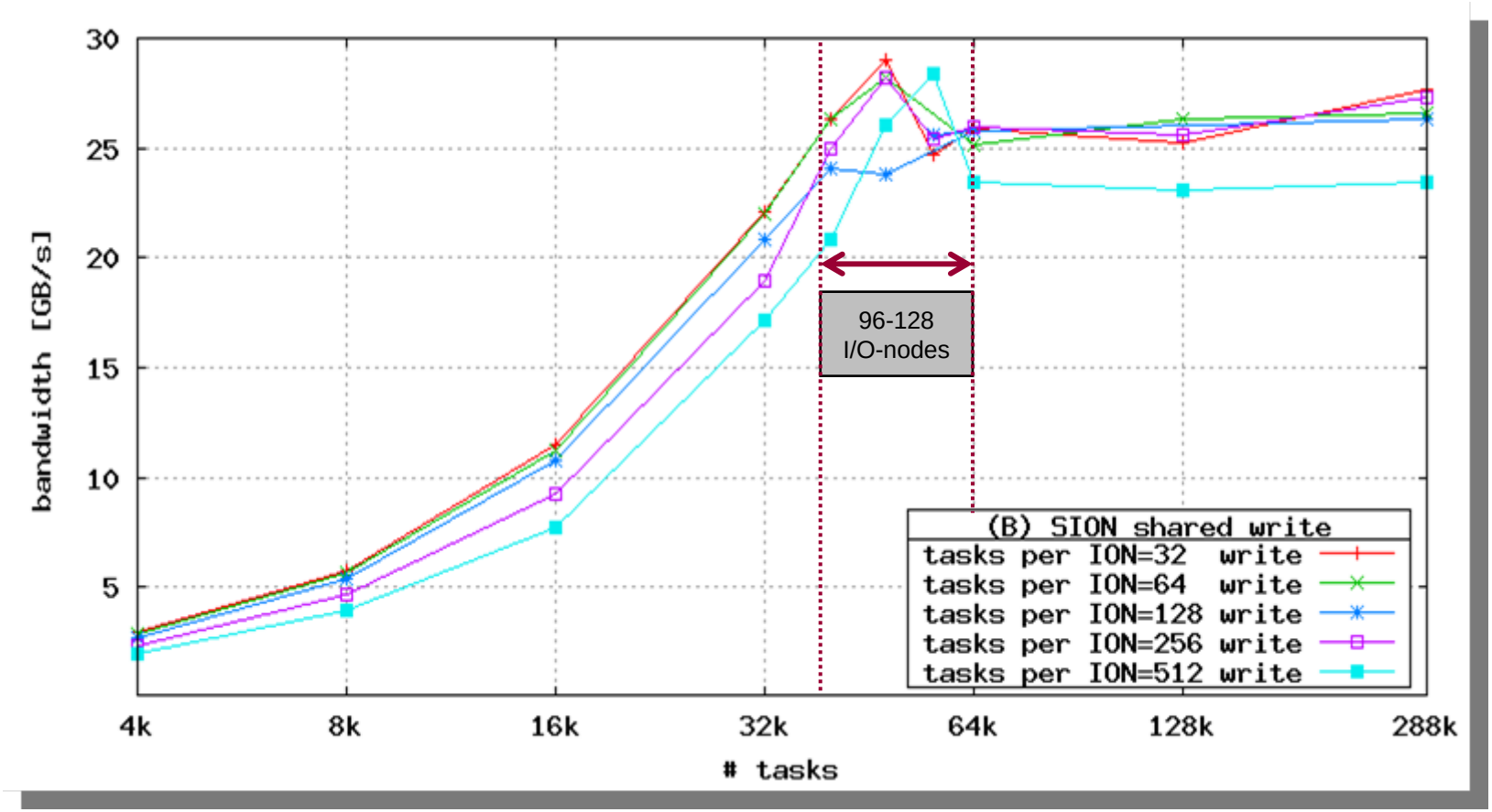
- Extension of ANSI C-API
- C and Fortran bindings, implementation language C
- Current version: 1.3p5
- Open source license: http://www.fz-juelich.de/jsc/sionlib

```
/* fopen() → */
sid=sion_paropen_mpi( filename , "bw",
                      &numfiles, &chunksize,
                      gcom, &lcom, &fileptr, ...);

/* fwrite(bindata,1,nbytes, fileptr) → */
sion_fwrite(bindata,1,nbytes, sid);

/* fclose() → */
sion_parclose_mpi(sid)
```

SIONlib: Scaling on Full-System (BG/P)



JUGENE: Total bandwidth (write) scaling, one file per I/O-node (ION), varying the number of tasks doing the I/O

SIONlib: Applications

- **Applications**

  DUNE-ISTL: Multigrid solver, Univ. Heidelberg (restart)

  ITM: Fusion-community (restart)

  LBM: Fluid flow and mass transport, Univ. Marburg (restart)

  MP2C: Massively Parallel Multi-Particle Collision Dynamics, (restart, output files)

  OSIRIS: Fully-explicit particle-in-cell code (restart)

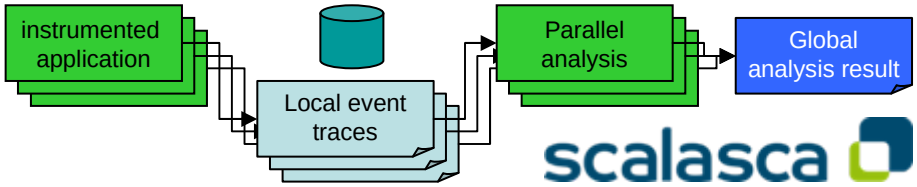
  PSC: SIONlib I/O-module for particle-in-cell code

  PEPC: Pretty Efficient Parallel Coulomb Solver (restart, output)

  Profasi: The Protein folding and aggregation simulator (I/O layer for high frequent I/O of small chunks)

- **Tools**

  Scalasca: Performance Analysis

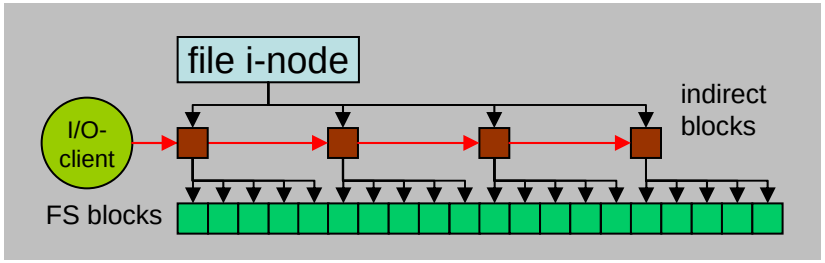


  LINKtest: Large scale MPI P2P bandwidth benchmark, communication matrix, Salsa-viewer

Touching the Limit again: Number of Tasks per Shared File (I)

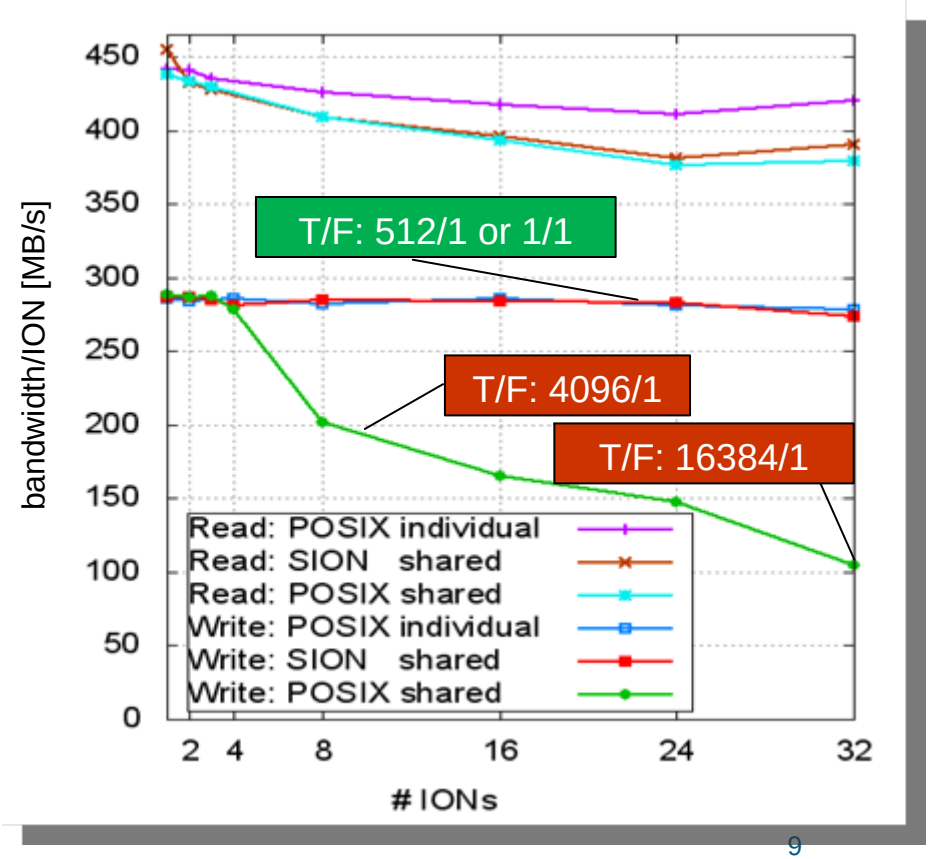
Meta-data wall on file level

- File meta-data management
- Locking



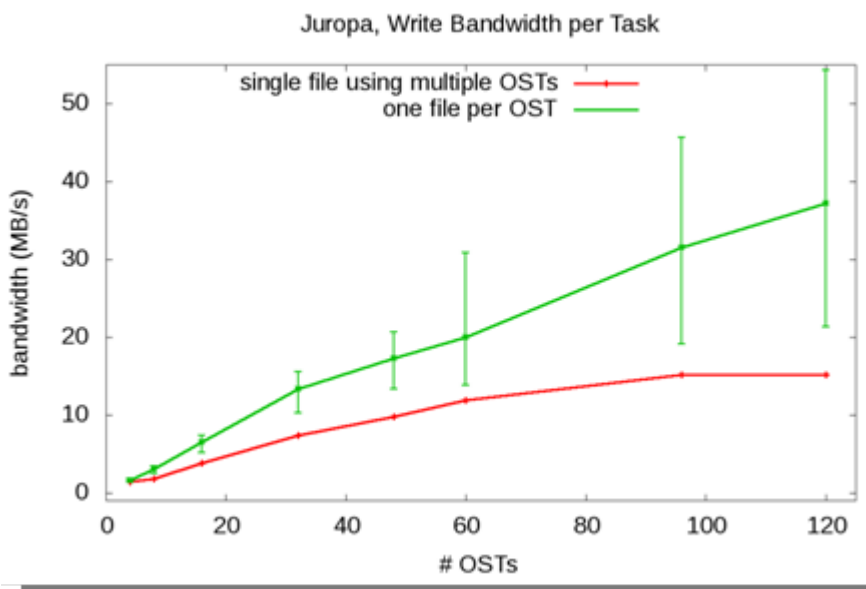
Example Blue Gene/P

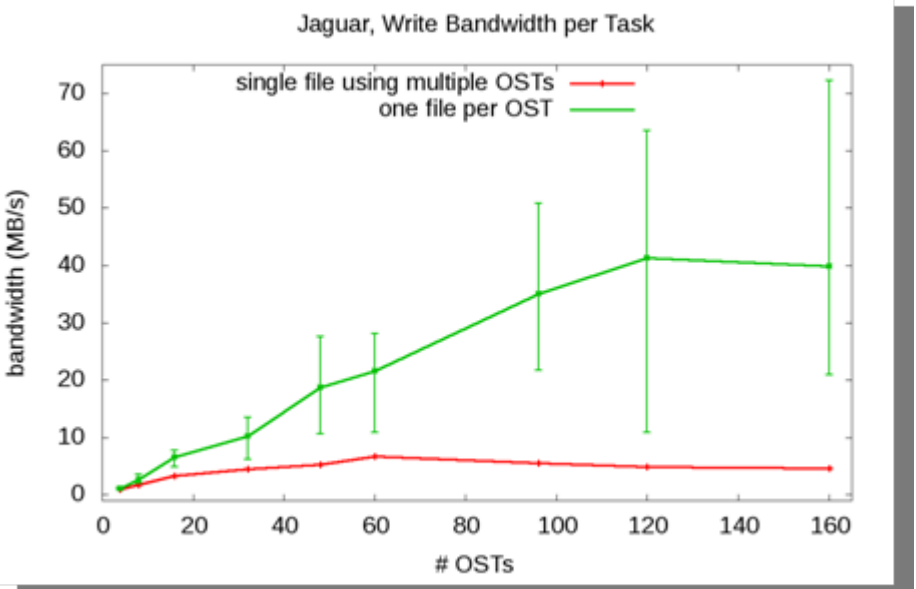
- Jugene (72 racks)
- I/O forwarding nodes (ION)
- GPFS client on ION
- Solution:
  - → tasks : files ratio ~ const
  - → SIONlib:
    one file per ION
    implicit task-to-file mapping



Touching the Limit again: Number of Tasks per Shared File (II)

- Example Lustre
  - Shared file distributed over multiple I/O servers (OST, Object Storage Target)
  - Meta-data synchronization between OSTs during write
  - Solution: no distribution over OSTs (stripe count → 1)
  - SIONlib: One file per OST, pre-create mapping



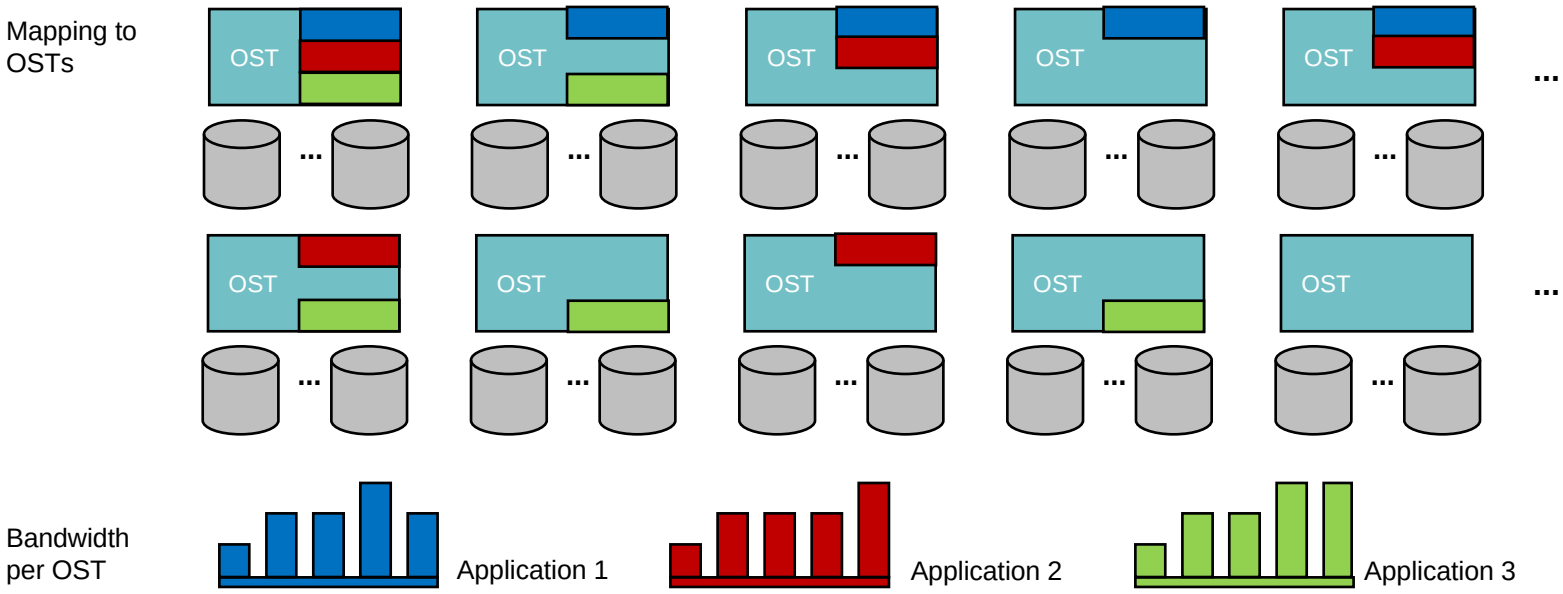


Challenges at Large Scale: File System as a Shared Resource

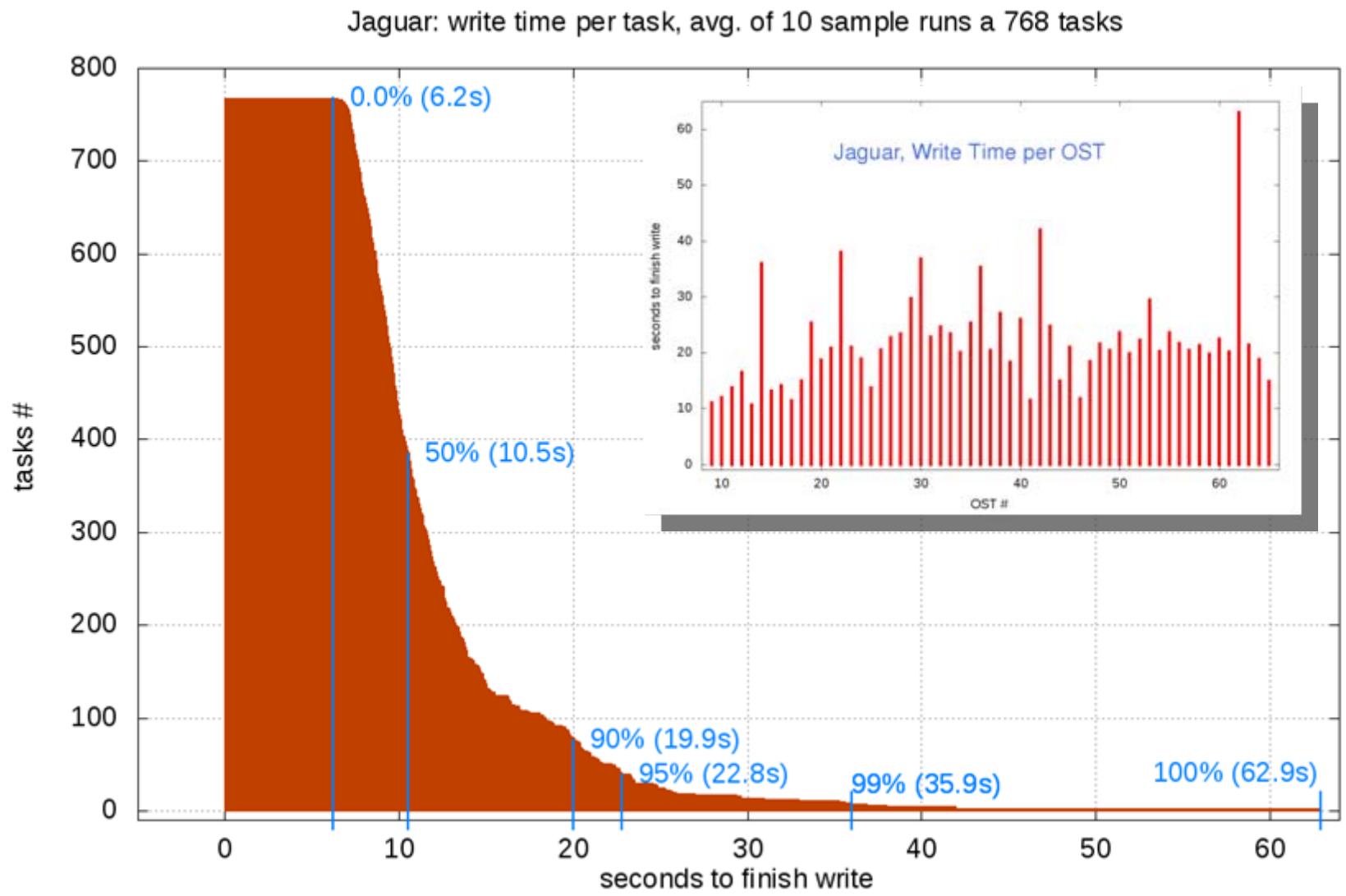
- I/O of one application: typically parallel, synchronous and uniform
- File system is used by several applications at a time

→ different load on different I/O servers (OSTs)

→ load imbalance in parallel I/O of one application

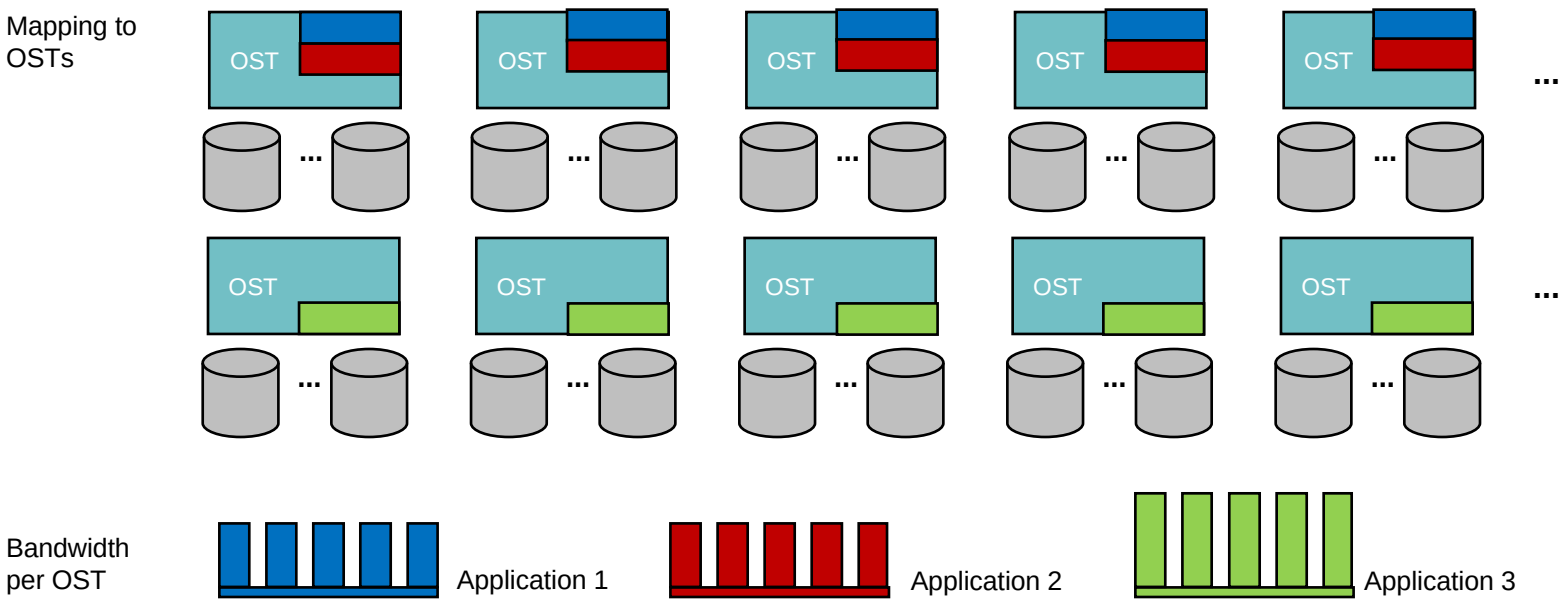


Challenges at Large Scale: File System as a Shared Resource



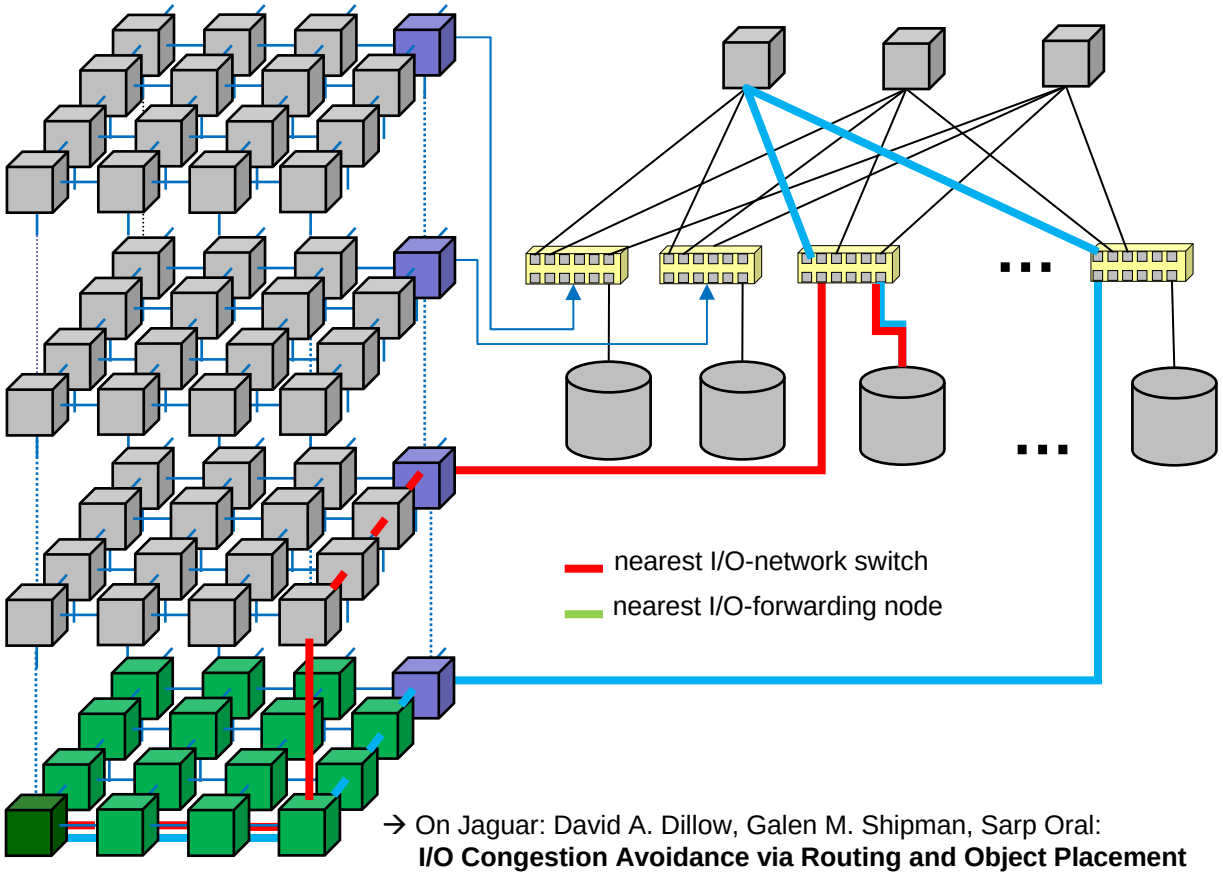
Challenges at Large Scale: File System as a Shared Resource

- Optimization:
  - → Dedicate I/O servers to applications
  - → Uniform mapping: applications – I/O server
  - Application and I/O topology → task mapping



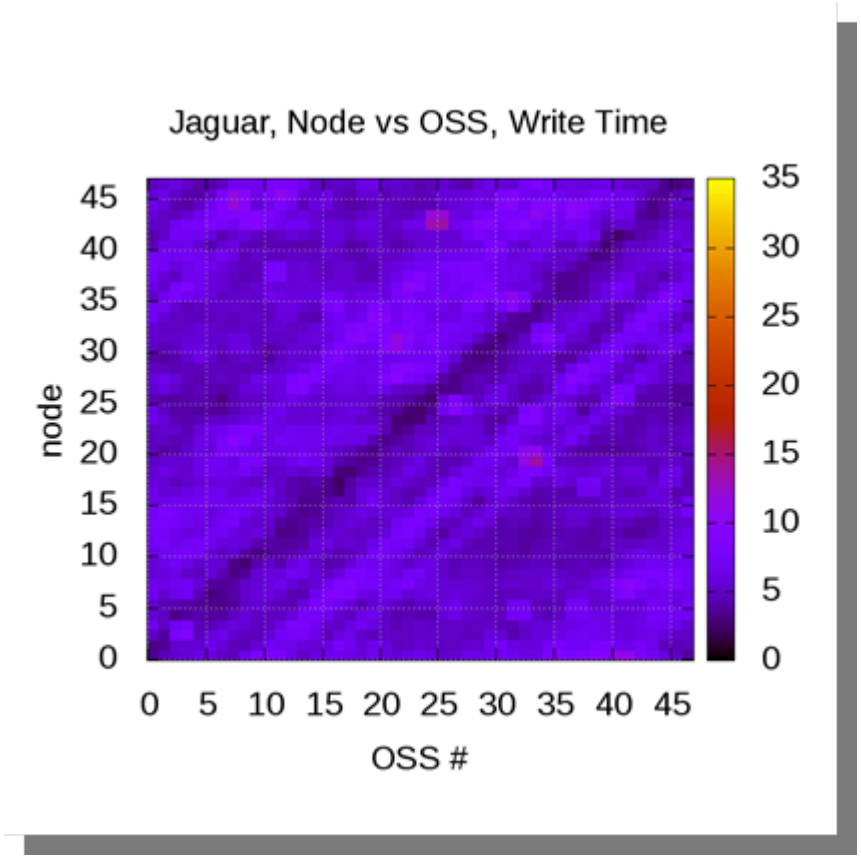
Challenges at Large Scale: (I/O) Network as a Shared Resource

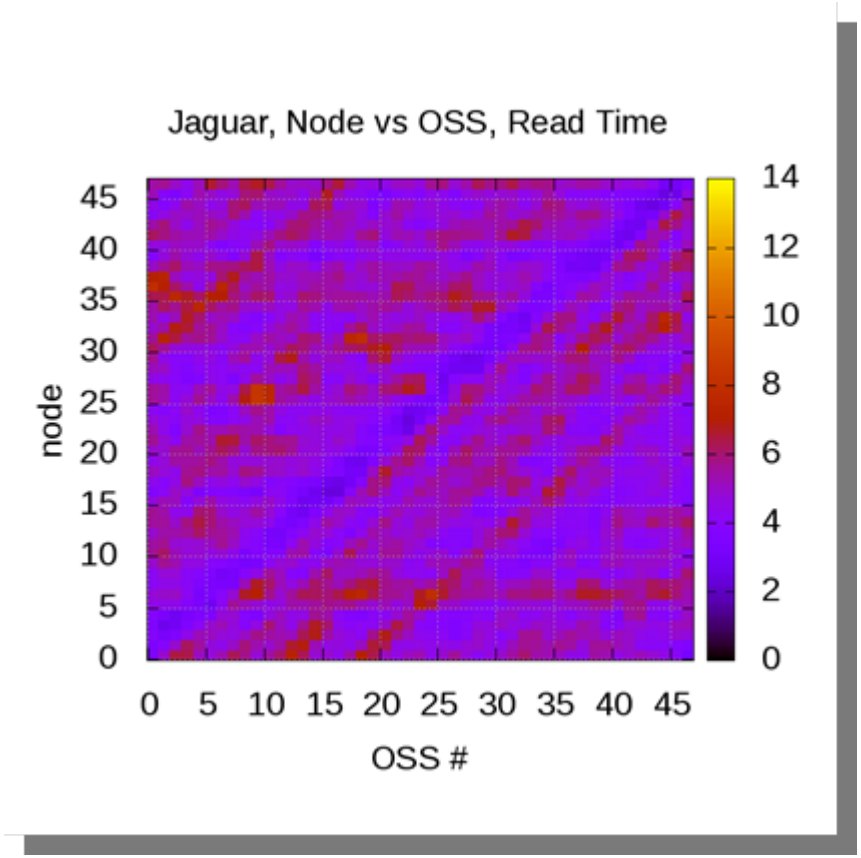
- Two network topologies (Torus + Fat Tree)
- Contention and cross traffic caused by multiple I/O streams
- Solution
  - "I/O zoning"
  - Topology-aware task mapping
- SIONlib
  - Task-to-file mapping
  - multi-physical files



→ On Jaguar: David A. Dillow, Galen M. Shipman, Sarp Oral: **I/O Congestion Avoidance via Routing and Object Placement**

Challenges at Large Scale: (I/O) Network as a Shared Resource

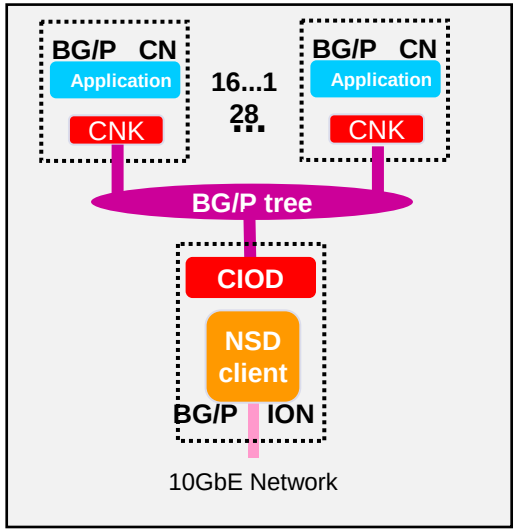




JAGUAR: 768 tasks writing/reading 90 MB each (average of 3 runs)

SIONlib: Task-Mapping

- Example: Blue Gene/P
  - One file for all tasks connected to same I/O node (Pset)
  - BG/P runtime provides Pset information: MPI communicator via MPIX-function
  - Additional parameter to SION open function: local MPI communicator



- Example: Lustre file system
  - One file per I/O server (OST)
  - Lustre parameters: stripe count, size and offset
    → Accessible by application
  - Lustre forwarding and routing (LNET)
    → Accessible on system level

Conclusion

- Parallel task-local I/O
  - Checkpoints, restarts
- Challenges at large scale
  - Management of large number of files
  - Meta-data walls
  - Usage of shared resources
- SIONlib
  - Provides shared file access for task-local parallel I/O
  - Multi-file support to break the meta-data walls
  - Supports flexible mapping of tasks to I/O component
- Approaches for Exa-scale
  - Application and topology-aware task-to-file mapping
  - “I/O-zoning” on software level (library)