Introduction to Parallel IO

PPCES 2019

Joachim Protze / Marc-André Hermanns

IT Center / JARA-HPC

(with content used with permission from FZJ course
“Parallel I/O and Portable Data Formats” by Sebastian Lührs)

Motivation

- **Bottlenecks of serial I/O**
  - → Bandwidth is limited by performance of a single data path
- **I/O to local disk lacks shared view**
- **HPC file systems are parallel**
  - → Applications should leverage this

Filesystems on Claix 2018

Env	Path	Type	Bandwidth	Use case	Backup
$HOME	/home/*<username>*	CIFS/NFS	10 GB/s	Source code, configuration files	yes
$WORK	/work/*<username>*	CIFS/NFS	3 GB/s	Output files, working data	no
$HPCWORK	/hpcwork/*<username>*	LUSTRE	150 GB/s	I/O intensive, Large files	no

- **Use appropriate file system for your data**
  - → Potentially stage data between different file systems for your jobs
- **NB: Per-node bandwidth on $HPCWORK is 12 GB/s (omnipath)**
  - → Use at least 13 nodes to achieve full bandwidth

I/O Strategy Overview

- **Master process performing I/O**
  - → Easy to implement
  - → Serializes I/O
- **Task-local files**
  - → Easy to implement
  - → No synchronization issues
  - → Can severely overload meta-data servers and tools
  - → Physical file structure may not reflect logical file structure
- **Shared files**
  - → Potential process synchronization and false sharing
  - → Easier to keep logical and physical file structure congruent

Common Pitfalls

Frequent Flushing

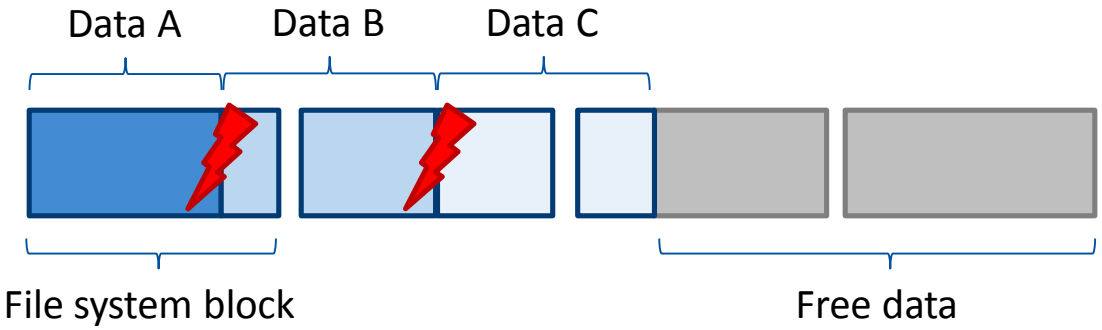
- **File systems read and write data per file system block**
  - → Common size on modern file systems are 4MB
- **Writing data**
  1. Read file system block from disk
  2. Modify block in memory
  3. Write block to disk
- **Frequent flushing forces the same block to be read and written multiple times**

Serialization of meta-data modification

- **Creating files or directories modifies parent directory meta data**
- **Concurrent accesses need to be serialized**
  - → Multiple processes creating files in a single directory
- **Meta-data server may get flooded with requests**
  - → Completion takes long
  - → Serialization may induce severe imbalance

False Sharing

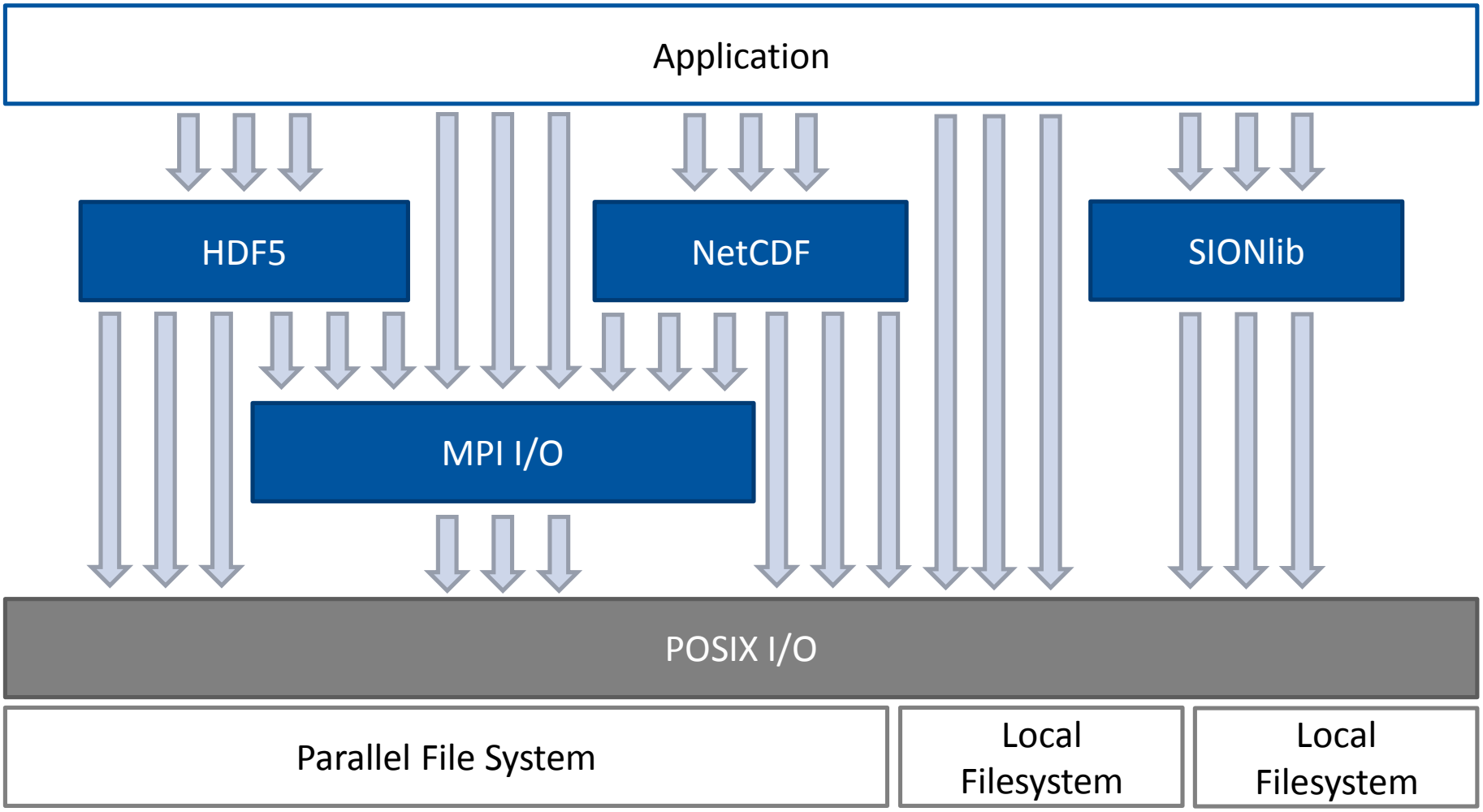
- **Each file comprises an exclusive collection of file system blocks**
- **Multiple processes may access data in the same file system block**
  - → Coordination of accesses needed
  - → Serialization of write accesses
- **Often difficult to optimize logical file structure for file system**
  - → File may be used on different file systems



Data Portability

- **Different hardware architectures may use different bit-representation**
  - → Big-endian vs. Little-endian
- **Manage endianness of data if needed**
  - → Save as meta data to file
  - → Enable on-the-fly conversion
- **Use a portable data format**
  - → NetCDF (https://www.unidata.ucar.edu/software/netcdf/)
  - → HDF 5 (https://www.hdfgroup.org/)
  - → Explore file formats used in your community

Software View



Conclusion

- **Impact of I/O on scalability if often overlooked**
- **Implementing a parallel I/O strategy**
  - → Depends on application and data distribution
  - → Can be complex
- **Enable data portability with portable data formats (NetCDF/HDF5)**
  - → Check which data format is common for your community
- **Enable fast checkpointing with SIONlib**
  - → Fast task-local I/O to shared files
- **Want to know more?**
  - → Forschungszentrum Jülich / PATC Course:

    “Parallel I/O and Portable Data Formats”

Thank you for your attention!