HPC.NRW

aiXcelerate 2021: Part I – File I/O

HPC.NRW Competence Network

Parallel I/O Schemes

Marc-André Hermanns

Significant parts taken from the guest contribution to the PDP lecture
by Wolfgang Frings of Jülich Supercomputing Centre, Forschungszentrum Jülich

I/O in applications

HPC.NRW

– Serial POSIX I/O

```c
#include <stdio.h>
#include <fcntl.h>
double data=3.1415;
int fd = open("foo.dat", O_RDWR | O_CREAT);
int sz = write(fd,&data,sizeof(double));  /* or */
int sz = read( fd,&data,sizeof(double));
close(fd);
```

C

– Serial ANSI I/O

```c
#include <stdio.h>
#include <fcntl.h>
double data=3.1415;
FILE* fp = fopen("foo.dat", "r+");
sz = fwrite(&data,sizeof(double),1,fp);  /* or */
sz = fread( &data,sizeof(double),1,fp);
fclose(fd);
```

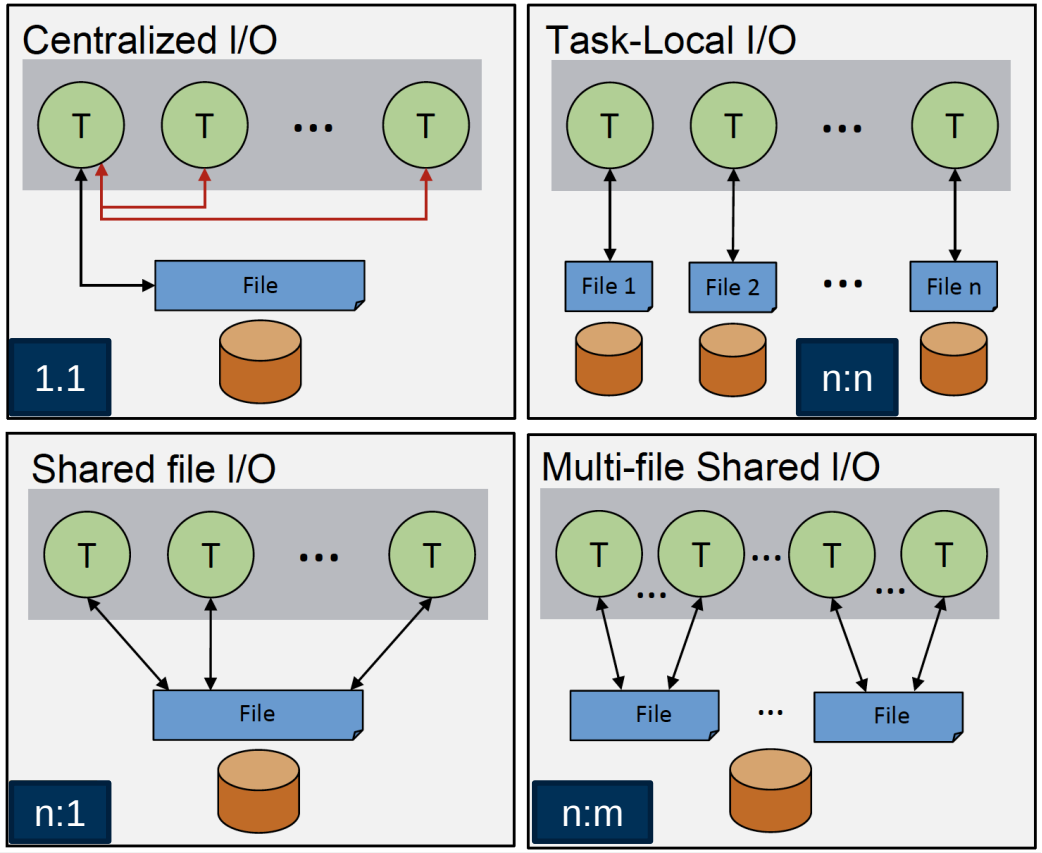
C

– Output: binary data stream (remember endianess)

Parallel I/O Schemes

HPC.NRW

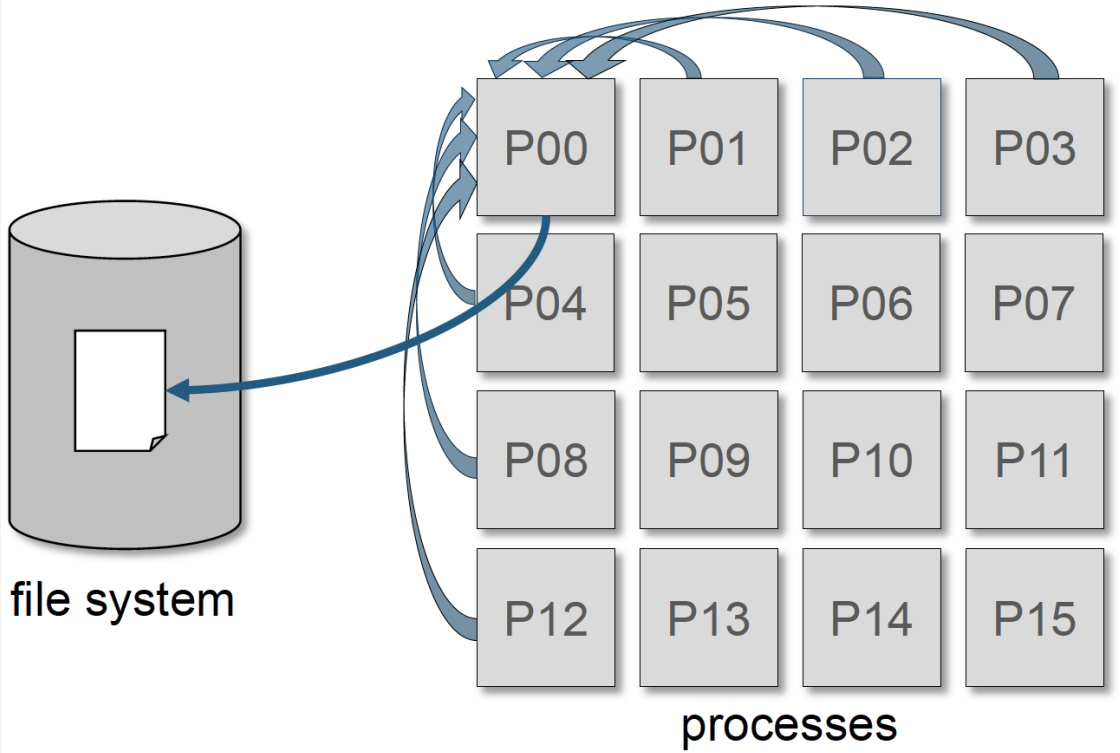
- Parallel I/O is defined by I/O operations that write or read shared data simultaneously from multiple processes of a parallel application
  - a: number of processes
    b: number of files



Centralized I/O

HPC.NRW

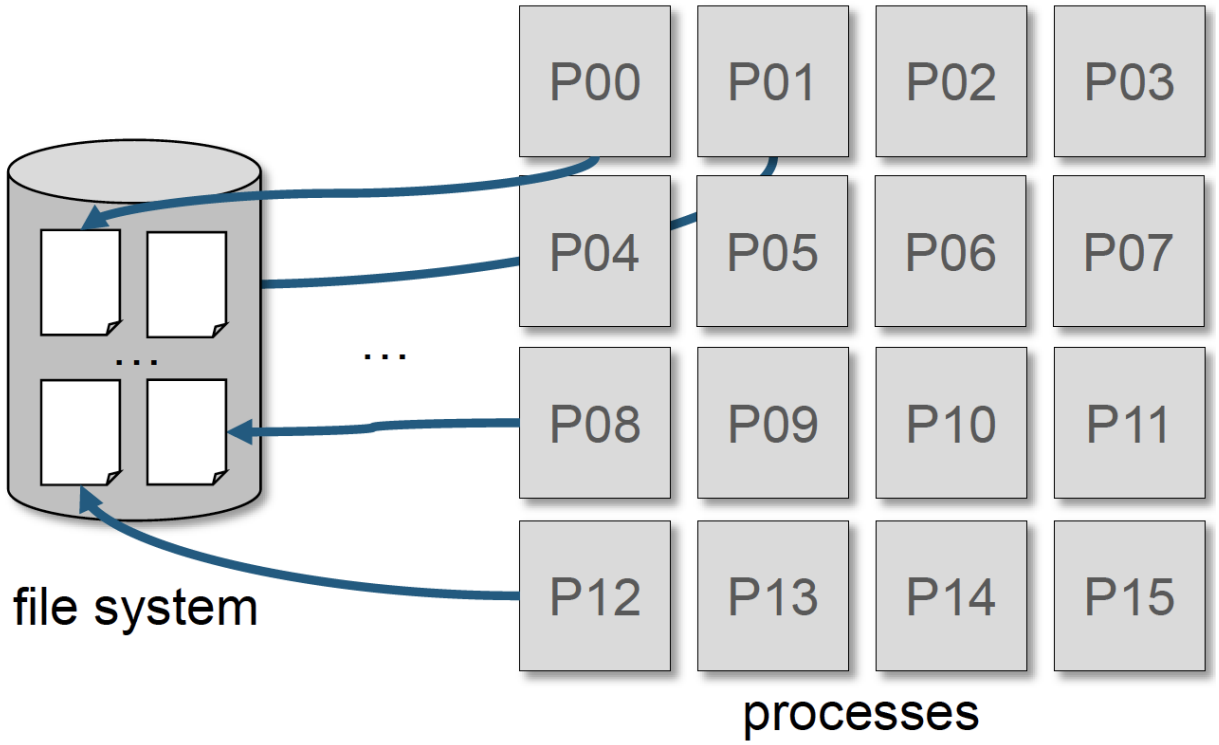
- One task performs I/O
- Pro:
  - Simple to implement
- Con:
  - I/O bandwidth is limited to the rate of this single process
  - Additional communication might be necessary
  - Other processes might idle (wasting resoures during I/O)



Task-local I/O

HPC.NRW

- One file per process
- Pro:
  - Simple to implement
  - No coordination or consistency mech. necessary
- Con:
  - Number of files quickly becomes unmanageable
  - Files often need to be merged in order to create a canonical dataset
  - Serialization of meta-data modification (see below)



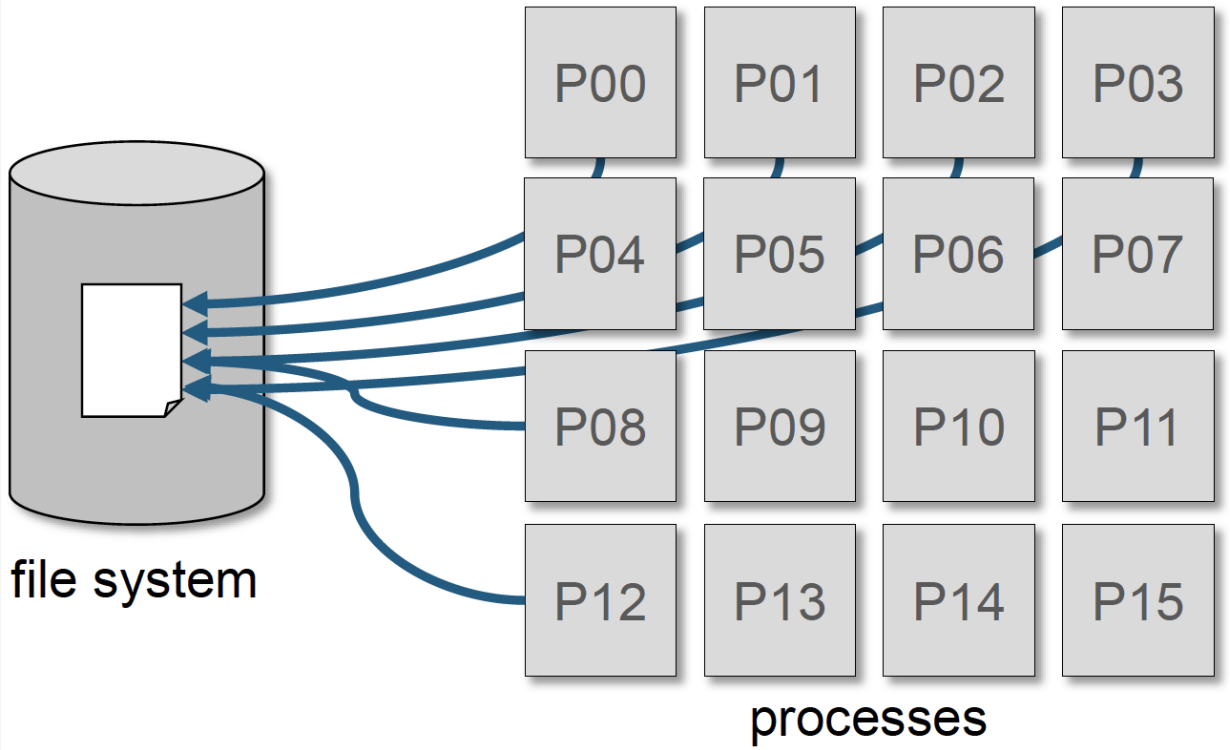
Shared Files

HPC.NRW

- Shared file(s)

- Pro:
  - Number of files is independent of no. of processes
  - Canonical dataset

- Con:
  - Uncoordinated client requests might induce time penalties
  - File layout might induce false sharing of file system blocks (see below)

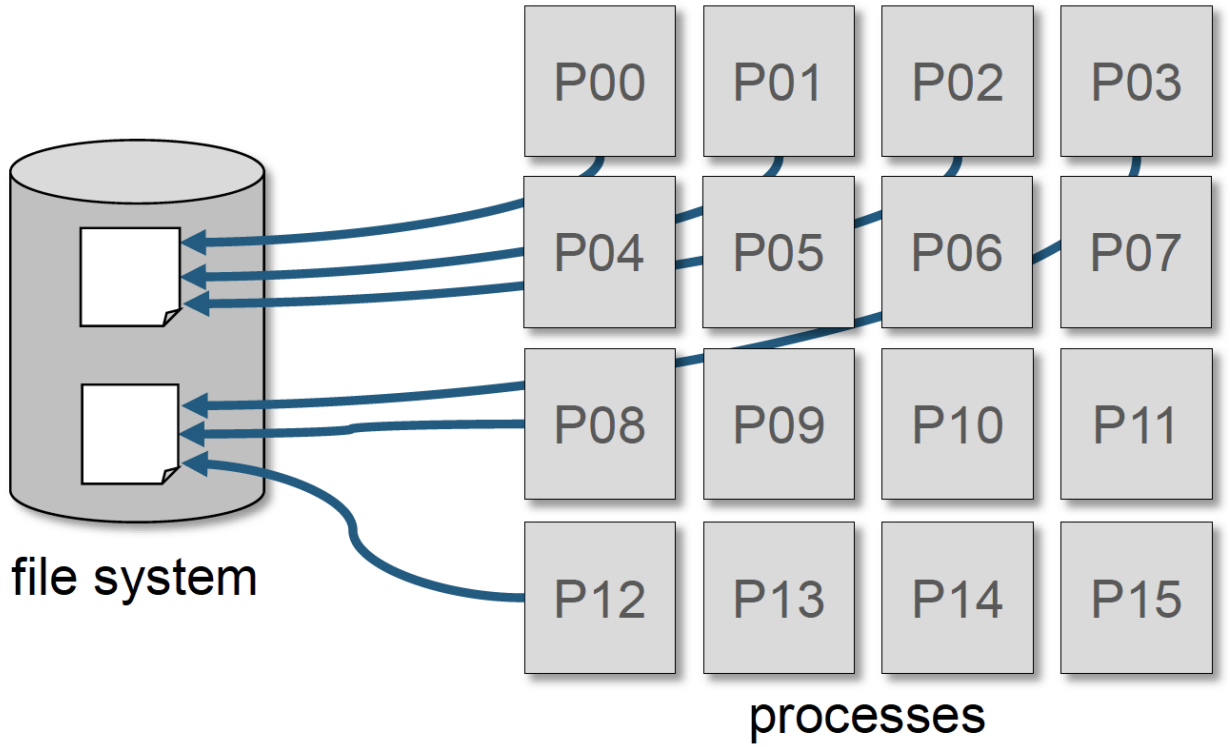


Multi-shared files

HPC.NRW

- Multi-shared files

- Pro:
  - Number of files is independent of no. of processes
  - Canonical dataset
  - Optimization according to locality or infrastructure possible

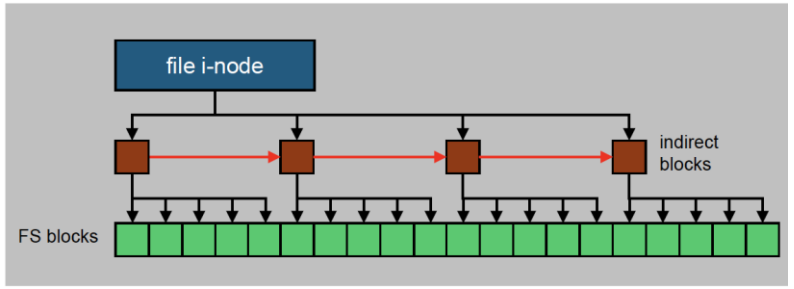


- Con:
  - Uncoordinated client requests might induce time penalties
  - File layout might induce false sharing of file system blocks (see below)

File system performance

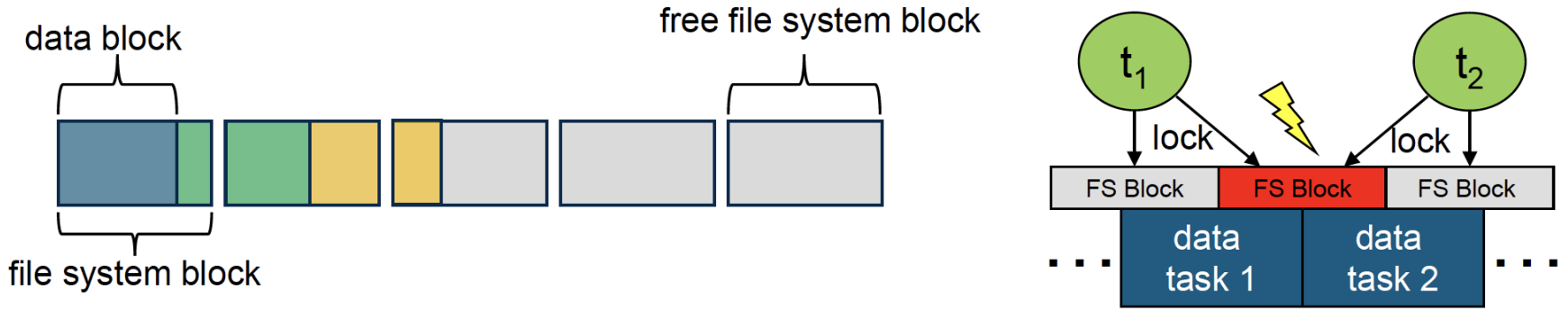
HPC.NRW

- File system blocks
  - Modern file systems in HPC have large file system blocks (e.g. 4MB)
  - A flush on a file handle forces the file system to perform all pending write operations
  - If application writes in small data, the same file system block has to be read and written multiple times
  - Performance degradation due to the inability to combine several write calls



- Solutions
  - Combine small I/O operations to one large operation accessing consecutive range of bytes in file
  - Stay inside node-local cache in memory, or any other buffer

False sharing of blocks

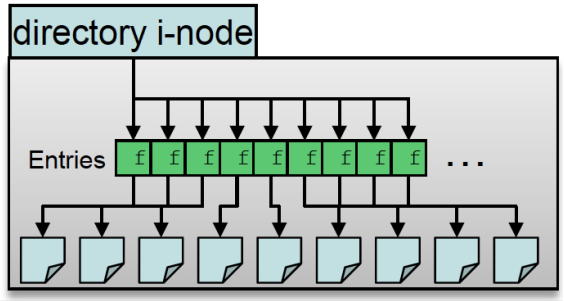


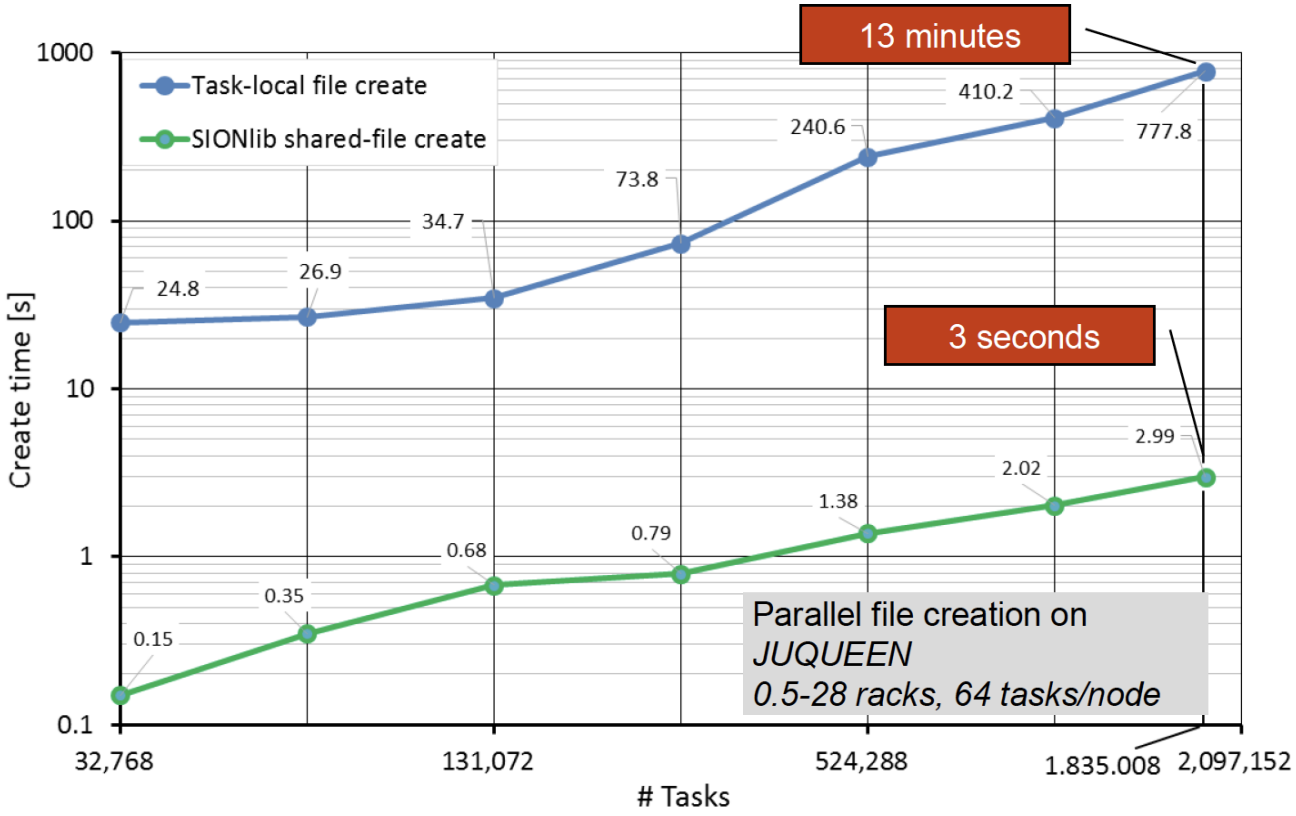
- Data blocks of individual processes do not fill up a complete file system block
- Several processes share a file system block
- Exclusive access (e.g. write) must be serialized
- The more processes have to synchronize, the more waiting time will propagate through the parallel application

Serialization of meta-data modification

HPC.NRW

– Example: creating files in the same directory





The creation of 1.835.005 files costs **99296 core hours** on JUQUEEN!

Overview of IO Libraries

HPC.NRW

Feature	POSIX	MPI-IO	SIONlib	HDF5	NetCDF
Individual interface	✓	✓	✓	✓	✓
Collective interface		✓	✓	✓	✓
Prevent false-sharing			✓		
File format portability		(✓)	(✓)	✓	✓
Non-contiguous access		✓			
Multi-dimensional Cartesian access		✓		✓	✓