JÜLICH
FORSCHUNGSZENTRUM

sionlib

Scalable I/O library for native parallel access to binary files

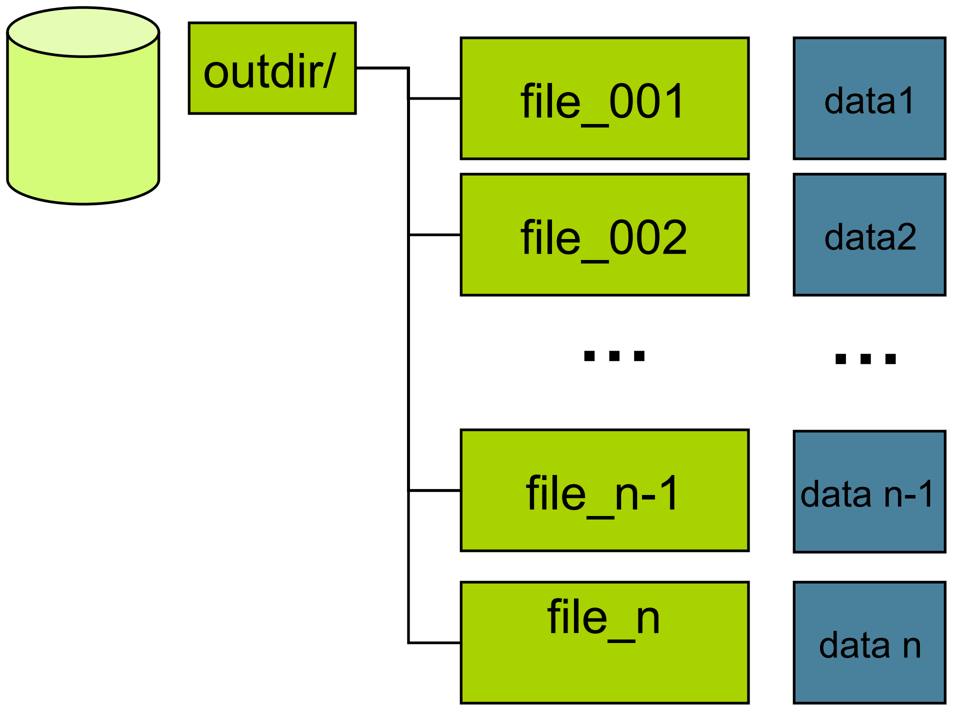
July 30, 2008 | Wolfgang Frings

sionlib: Overview

- **S**calable **I/O** library for **n**ative parallel access to binary files
- Parallel access to one or more direct access files from thousands of tasks
- Designed for handling of binary parallel program I/O, e.g.
  - writing scratch/restart files from each task
  - trace files from performance analysis tools (Scalasca)
- simplified file handling
  → *only one large file instead of thousands small files*
- optimized I/O
  → *alignment to file system blocks*
- minimal source code changes
  → *using of standard filte pointer (FILE* fp)*
- supports intermediate flushes if task writes more data than expected

Typical use case: parallel I/O to separate files

```
MPI_Init() /*  n tasks */
   …
   fileptr=fopen(file_###)
   …
   fwrite(buffer,fileptr)
   …
   fclose(fileptr)
   …
MPI_Finalize()
```

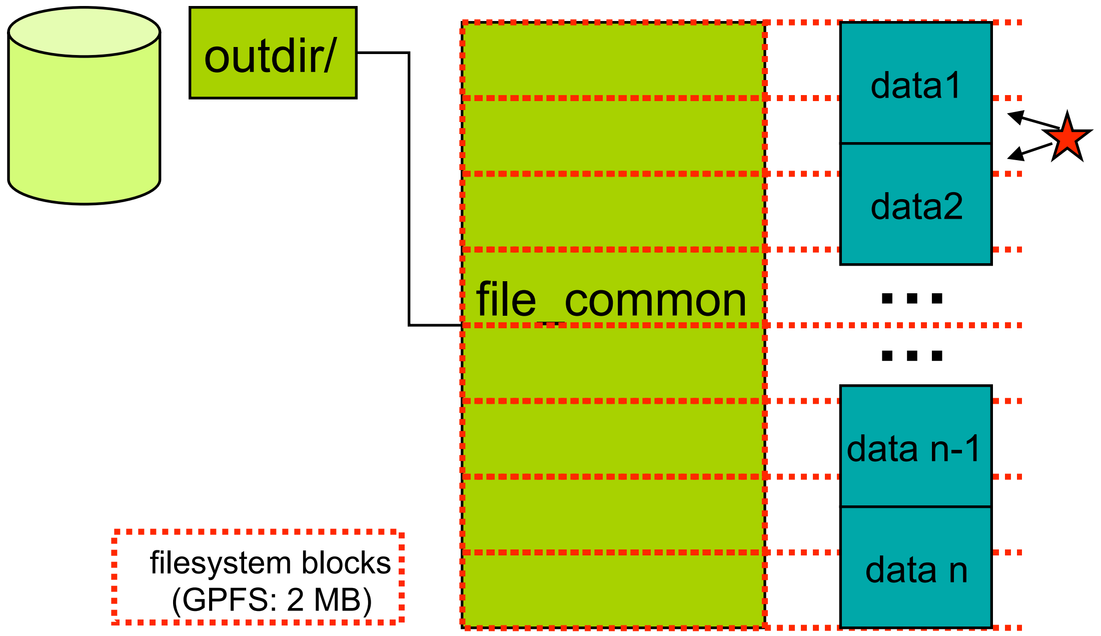


Problem 1: file handling (Backup, HSM) ← number of files

Problem 2: slow create & open of files ← Lock on outdir (serialization)

Example: native parallel direct access

```
MPI_Init() /*  n tasks */
   …
   fileptr=fopen(file_common)
   …
   fseek(mypos)
   fwrite(buffer,fileptr)
   …
   fclose(fileptr)
   …
MPI_Finalize()
```



Initial Problem solved: fast open, only one file

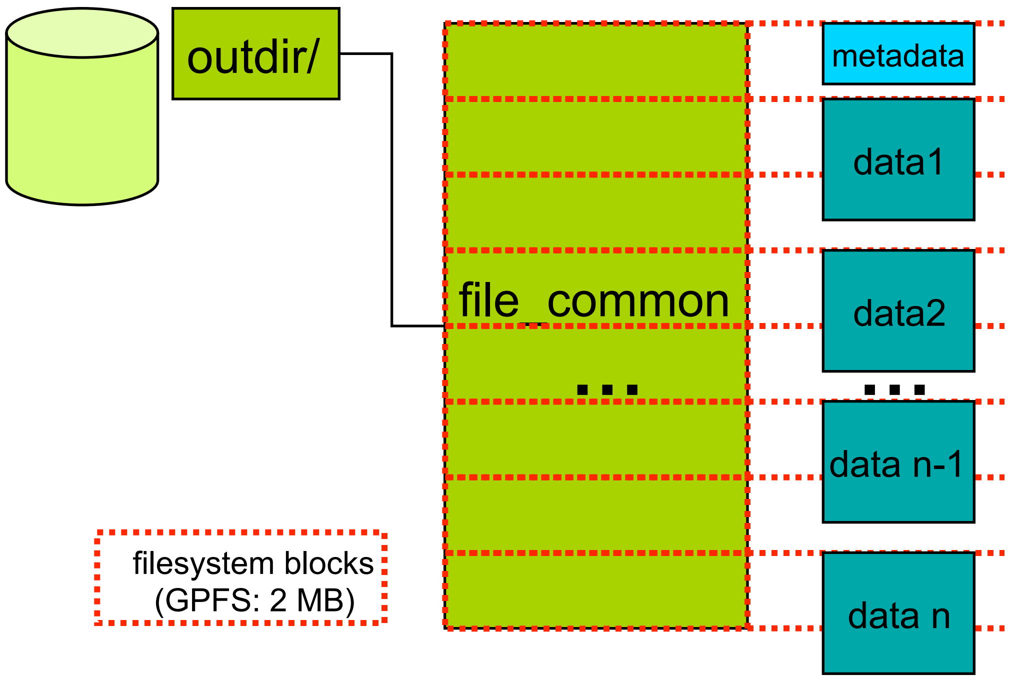
New Problem 1: meta data handling, start positions and length not stored

New Problem 2: filesystem locks on blocks, overlapping parallel access to blocks ★

Restriction: specification of data block size before writing data (for fseek)

sionlib: access with sionlib

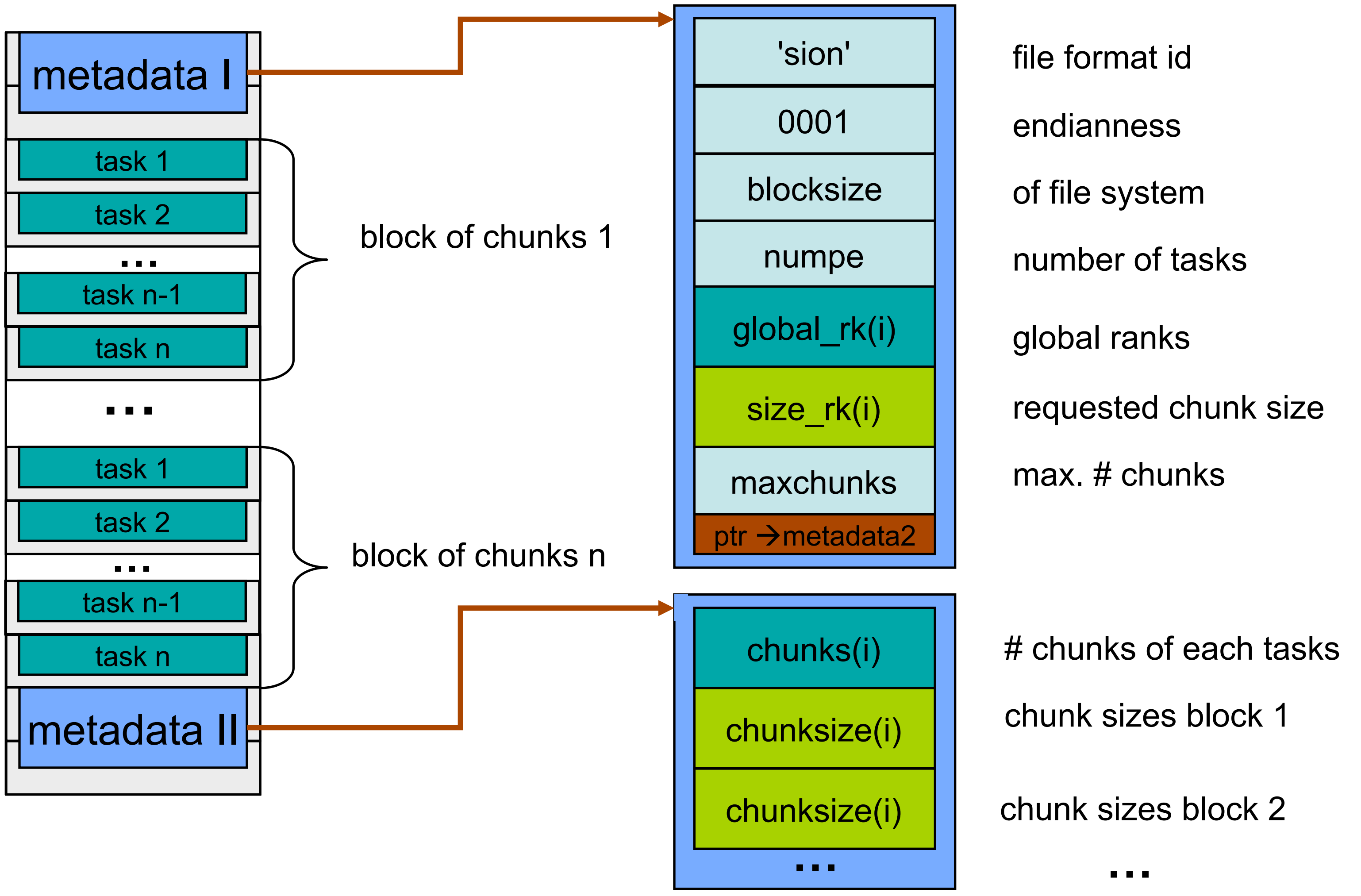
```
MPI_Init() /*  n tasks */
  …
 sid=sion_paropen_mpi(fname,
                blocksize, …, &fileptr)
 …
 sion_ensure_free_space(sid, bsize)
 fwrite(buffer,fileptr)
 …
 sion_ensure_free_space(sid, bsize)
 fwrite(buffer,fileptr)
 …
 sion_parclose(sid)
 …
MPI_Finalize()
```



Problems solved: simple file handling, fast open and fast I/O (fs block alignment)

Restriction: specification of data block size before writing data (for fseek …)
→ SION-internal flushes if writing more data, sion functions move filepointer to correct position

sionlib: internal file format



sionlib Usage: parallel read/write

parallel write

```
sid=sion_paropen_mpi( ... ,chunksize, comm, &fileptr, ...)      # collective
  loop: {
          sion_ensure_free_space(sid,nbytes);                     # non-collective
          fwrite(data,1,nbytes,fileptr)
        }
  sion_parclose_mpi(sid, comm);                                   # collective
```

parallel read

```
sid=sion_paropen_mpi( ... ,&chunksize, comm, &fileptr, ...)     # collective
  while((!sion_feof(sid))) {                                      # non-collective
        btoread=sion_bytes_avail_in_block(sid);                   # non-collective
        bread=fread(localbuffer,1,btoread,fileptr);
  }
  sion_parclose_mpi(sid, comm);                                   # collective
```

sionlib usage: serial read/write

serial write

```
sid=sion_open( ...,chunksize, &fileptr)
rank_loop: {
       sion_seek(sid,rank,SION_CURRENT_BLK,SION_CURRENT_POS);
       sion_ensure_free_space(sid,nbytes);
       fwrite(...,fileptr)
  }
  sion_close(id);
```

serial read

```
 sid=sion_open( ...,chunksize, &fileptr)
 sion_get_locations(sid,&size,&blocks,…,&sion_chunks,&sion_chunksizes);
  loop: {
     sion_seek(sid,rank,blknr,pos);
     fread(...,fileptr)
  }
  sion_close(id);
```

sionlib: comand line tools

siondump [-a] <sionfile>

- prints on stdout all information from the first meta data block , with -a also all chunk sizes from the second meta data block

sionsplit [-d digits] <sionfile> <prefix>

- extracts task related files from a sion file
- a file will be generated for each task with a filename starting with <prefix>
- the task number will be appended to the <prefix>

siondefrag [-q blksize] [-s chunksize] <sionfile> <new_sionfile>

- generates a new sion file from an existing sion file
- the new file will have only one chunk per task which contains the data of all chunks of this task in the old sion file
- generates with “–q 1” a compact sion file without gaps

sionlib: Measurement on 16 rack Blue Gene/P

- BG/P connected to file server with 128 x 10 GiE
  GPFS file system bandwidth: ~ 6GB/s
- Parallel test: (file server in production)
  - writing and reading 2 TB data, 32 MB from each task
  - 65536 MPI-tasks, 128 I/O-nodes
  - parallel open of one SION file → < 1s
  - overall write bandwidth → 3.7 GB/s
    550s for writing 2 TB
  - overall read bandwidth → 5.4 GB/s
    380s for reading 2 TB