Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Scalable Massively Parallel I/O to Task-Local Files

22. May 2009 | Wolfgang Frings, Jülich Supercomputing Centre

ScicomP15, Barcelona

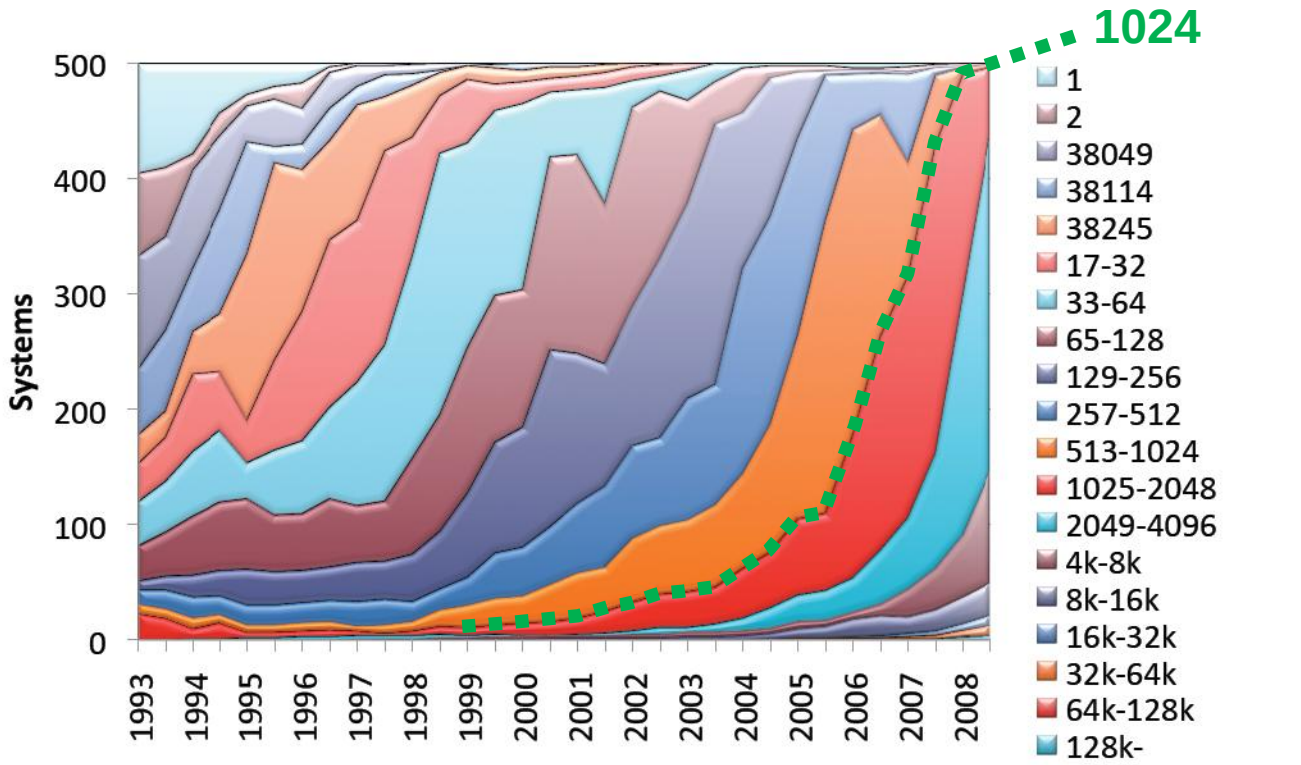
Increasing Importance of Scaling

- Number of Processors share for TOP 500 Nov 2008

NProc	Count	Share	∑Rmax	Share	∑NProc
<= 1024	4	0.8%	61 TF	0.4%	3,072
1025-2048	61	12.2%	923 TF	5.4%	113,906
2049-4096	290	58.0%	5,228 TF	30.9%	888,384
4097-8192	96	19.2%	2,860 TF	16.9%	550,150
> 8192	49	9.8%	7,855 TF	46.4%	1,561,411
Total	500	100%	16,927 TF	100%	3,116,923

- **Average** system size: **6234 cores**
- 4 smallest systems: 128, 960, 960, 1024

Increasing Importance of Scaling II



Jülich Supercomputing Centre (May 2009)

HPC-FF
Bull NovaScale R422-E2
8,640 cores

Juropa
Sun Blade 6048 system
17,664 cores

Jugene
72 rack IBM BlueGene/P
294,912 cores

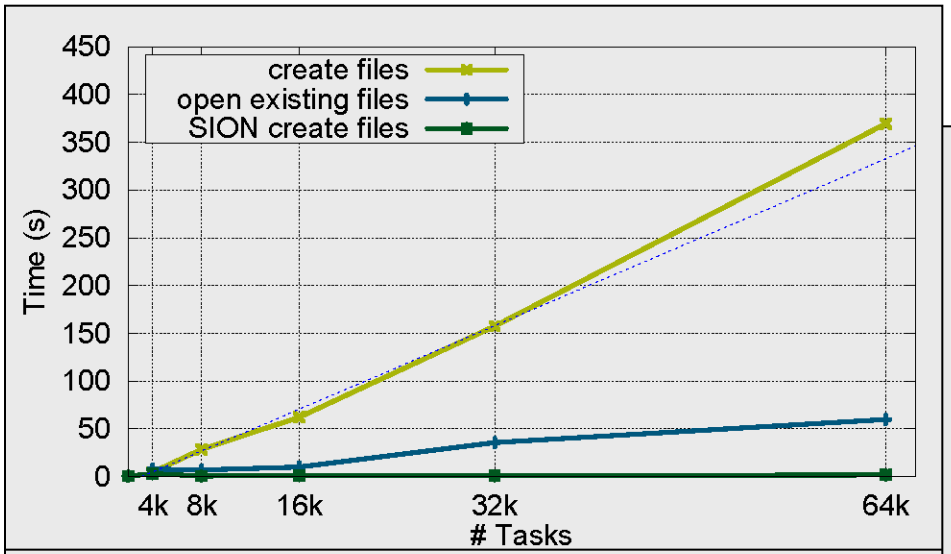
Motivation

- Many applications **write one or more files per MPI rank**, e.g.
  - Application checkpointing and restart files
  - Performance measurement tools
- Typical **issues on massively parallel systems**
  - Simultaneous file creation
  - Single-file parallel write
    - Filesystem Block Alignment
    - Local data size & file structure

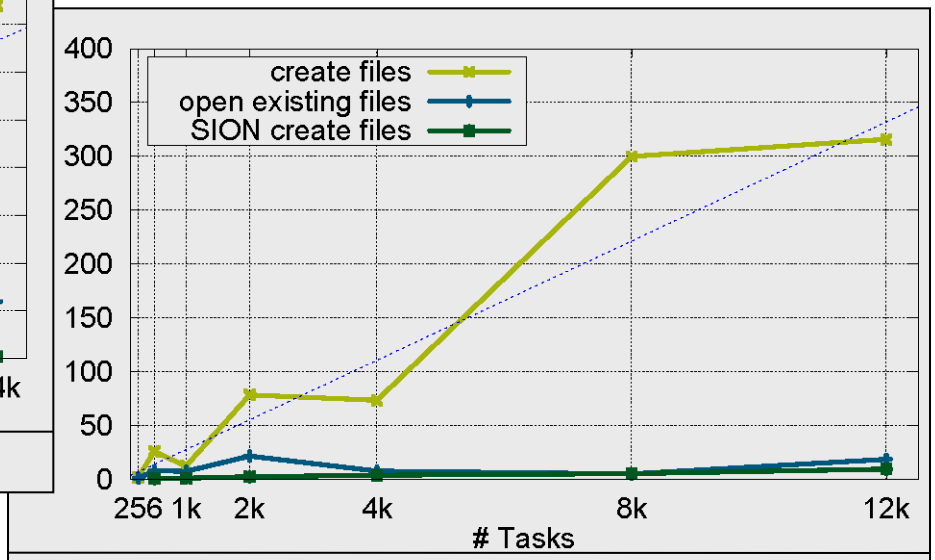
- **Our solution:** Library *SIONlib*
  - Scalable Massively Parallel I/O to Task-Local Files

Issue1: Simultaneous File Creation

- **Metadata contention** if creating thousands of files simultaneously in one directory (64k files ⇨ ~6min)
- If the creation problem could be solved
  ⇨ Handling of 64k files remains as a problem



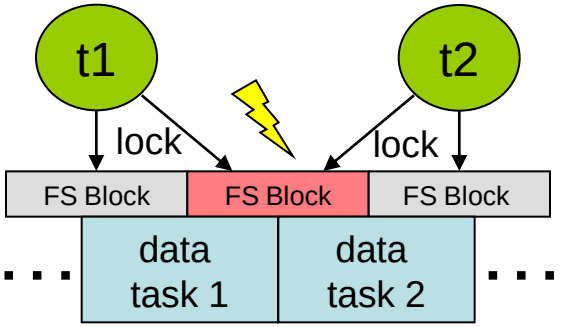
Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)



Jaguar (Oakridge, Cray XT4, Lustre, fs:scr72b)

Issue 2: Filesystem Block Alignment

- Contention problem if writing in parallel to direct access file:
  - ⇨ More than one task access one file system block at a the same time
- Similar to “false sharing” (cache line access)
- Could be prevented by:
  - Only one tasks write the data to fs block (e.g. MPI-I/O)
  - Align tasks related data to file system block (SIONlib)

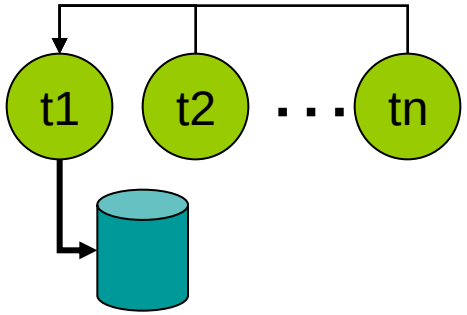


#tasks	data size	blksize	write bandwidth
32768	256 GB	aligned	5381 MB/s
32768	256 GB	not aligned	2125 MB/s

Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)

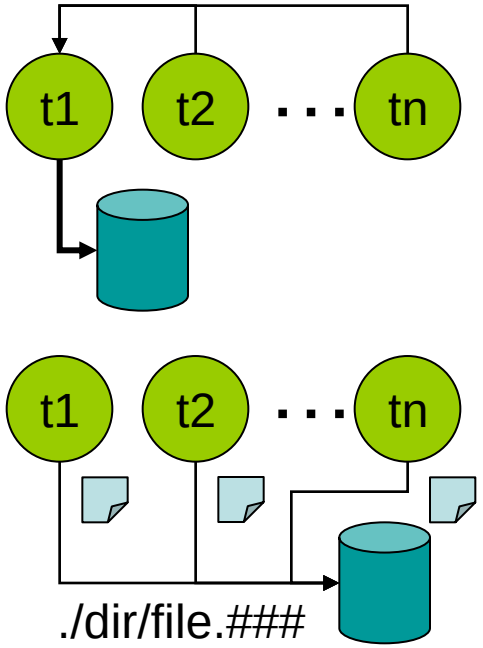
Application-based Checkpointing on Massively Parallel Systems

- **Single-file sequential write**
  - One writer, serialized I/O, bandwidth limited to node bandwidth
  - Memory and message buffer handling



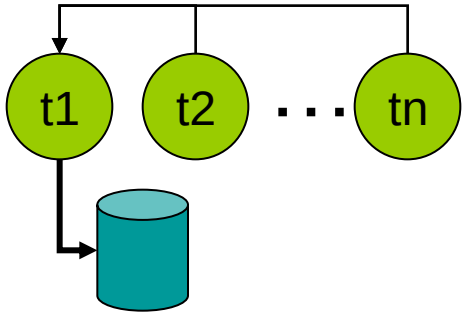
Application-based Checkpointing on Massively Parallel Systems

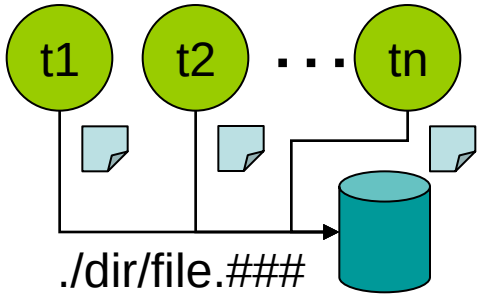
- **Single-file sequential write**
  - One writer, serialized I/O, bandwidth limited to node bandwidth
  - Memory and message buffer handling
- **Multiple-file parallel write**
  - Effective for saving task-local data
  - Limitation: time for file creation and file handling

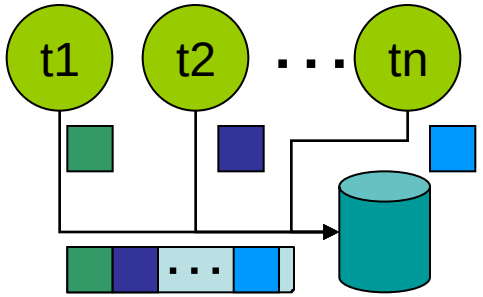


Application-based Checkpointing on Massively Parallel Systems

- **Single-file sequential write**
  - One writer, serialized I/O, bandwidth limited to node bandwidth
  - Memory and message buffer handling
- **Multiple-file parallel write**
  - Effective for saving task-local data
  - Limitation: time for file creation and file handling
- **Single-file parallel write**
  - Optimized I/O needed → library support
  - Metadata handling → library support
  - High-level libraries: MPI-IO, netCDF, HDF5, …
  - Binary stream data: SIONlib







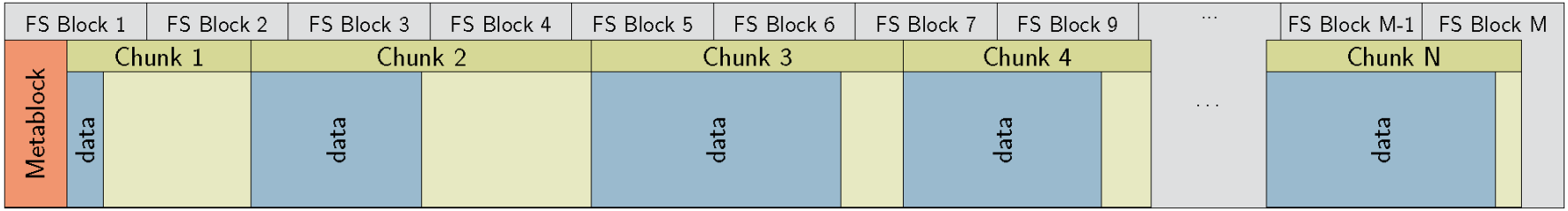
Comparison to Other Approaches

- **MPI-IO**
  - Requires to use MPI interface
  - Requires to use MPI data types to describe data
    - ⇨ (Potentially many) complex source code changes
    - ⇨ Especially if app uses own self-contained binary format
  - Tied to MPI

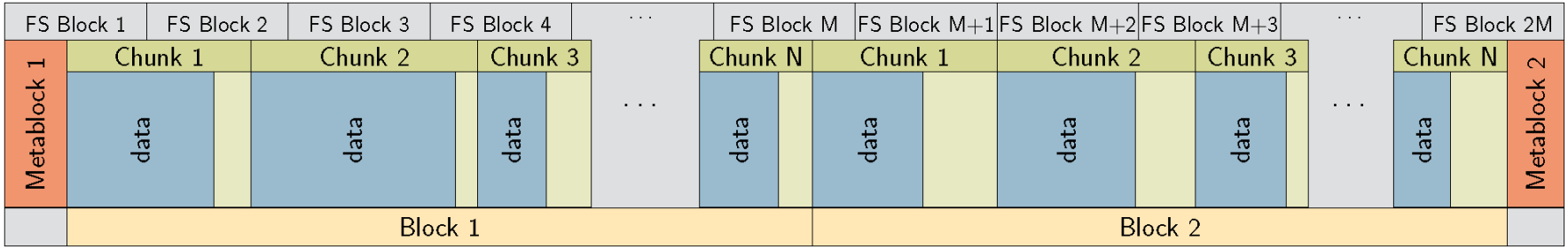
- **HDF5, NetCDF, …**
  - Requires to use library interface ⇨ many code changes
  - More useful for structured scientific data

Single-file Parallel Write: local data size & file structure

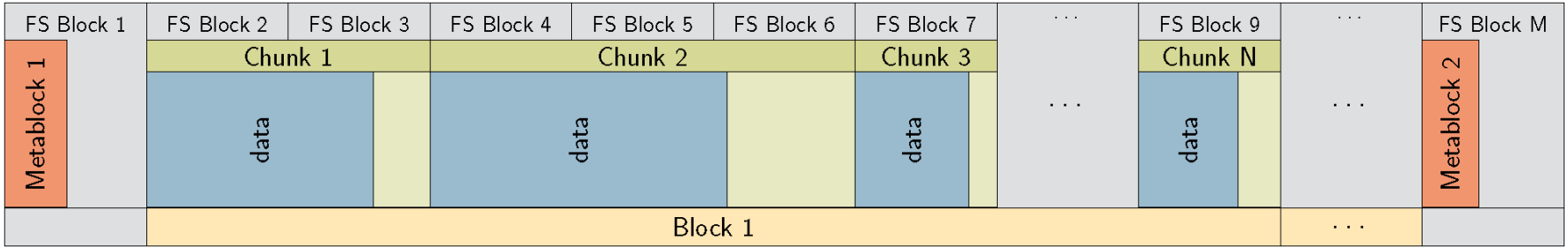
- **Limit1:** Maximum size of task-local data is known in advance



- **Limit2:** Maximum amount of data written on 1 piece is known in advance



- **Solution:** Chunks are aligned with file system block boundaries (SIONlib)



SIONlib: Writing API

- Parallel

```
/*-- open collective --*/
sid=sion_paropen_mpi( ... ,&chunksize,
                      gcom, &lcom, &fileptr, ...);

/*-- write non-collective --*/
sion_ensure_free_space(sid,nbytes);
fwrite(data,1,nbytes,fileptr);
/*-- or --*/
sion_fwrite(data,1,nbytes,sid);


/*-- close collective --*/
sion_parclose_mpi(sid)
```

- Serial

```
sid=sion_open( ...,&chunksizes,&fileptr);

sion_seek(sid,rank,chunk,pos);
sion_ensure_free_space(sid,nbytes);
fwrite(...,fileptr);

sion_close(sid);
```

SIONlib: Reading API

- Parallel

```
/*-- open collective --*/
sid=sion_paropen_mpi( ... ,&chunksize,
                      gcom, &lcom, &fileptr, ...);

/*-- read non-collective --*/
if (!sion_feof(sid)) {
btoread=sion_bytes_avail_in_chunk(sid);
bread=fread(localbuffer,1,btoread,fileptr);
/*-- or /
sion_fread(localbuffer,1,nbytes,sid);
}
/*-- close collective --*/
sion_parclose_mpi(sid);
```

- Serial

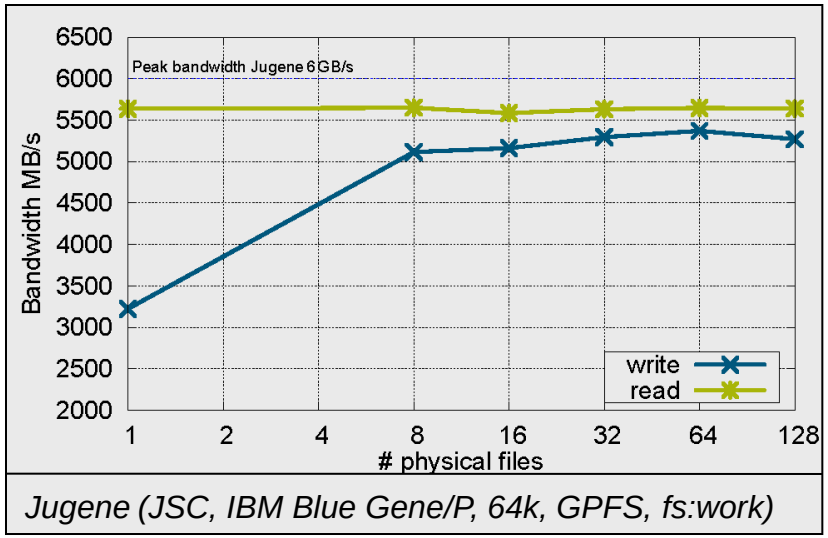
```
sid=sion_open( ...,&chunksizes,&fileptr);

sion_seek(sid,rank,chunk,pos);
sion_ensure_free_space(sid,nbytes);
fwrite(...,fileptr);

sion_close(sid);
```

SIONlib: Command Line Utilities

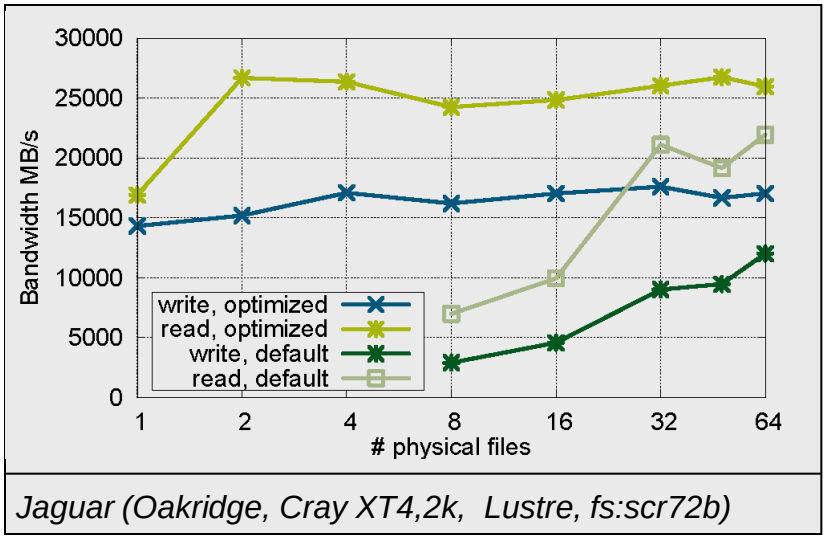
- `siondump`
  - Dumps multifile metadata to stdout
- `sionsplit`
  - Extracts all or only distinct logical files
  - Creates corresponding physical files
- `siondefrag`
  - Creates new multifile from existing one
    - Combines multiple chunks per rank
    - Removes gaps (un-used file system blocks)

SIONlib: Single or Multiple Physical File

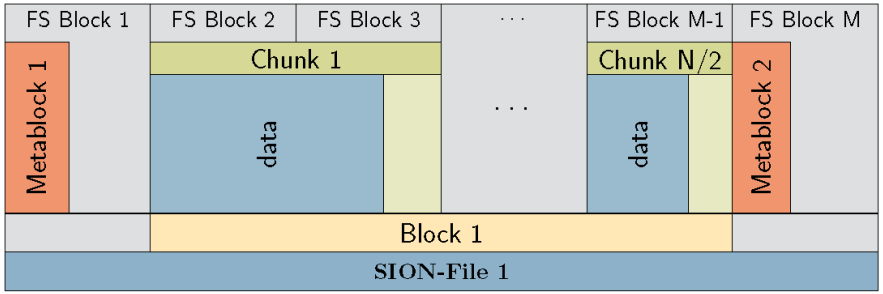
- Using **multiple physical files**, if underlying hardware or software can get advantage from parallelism or file size is limited
- Can be specified in `sion_par_open`

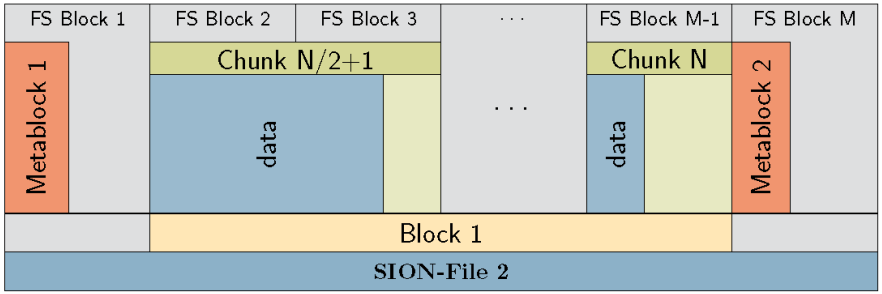


Jugene (JSC, IBM Blue Gene/P, 64k, GPFS, fs:work)

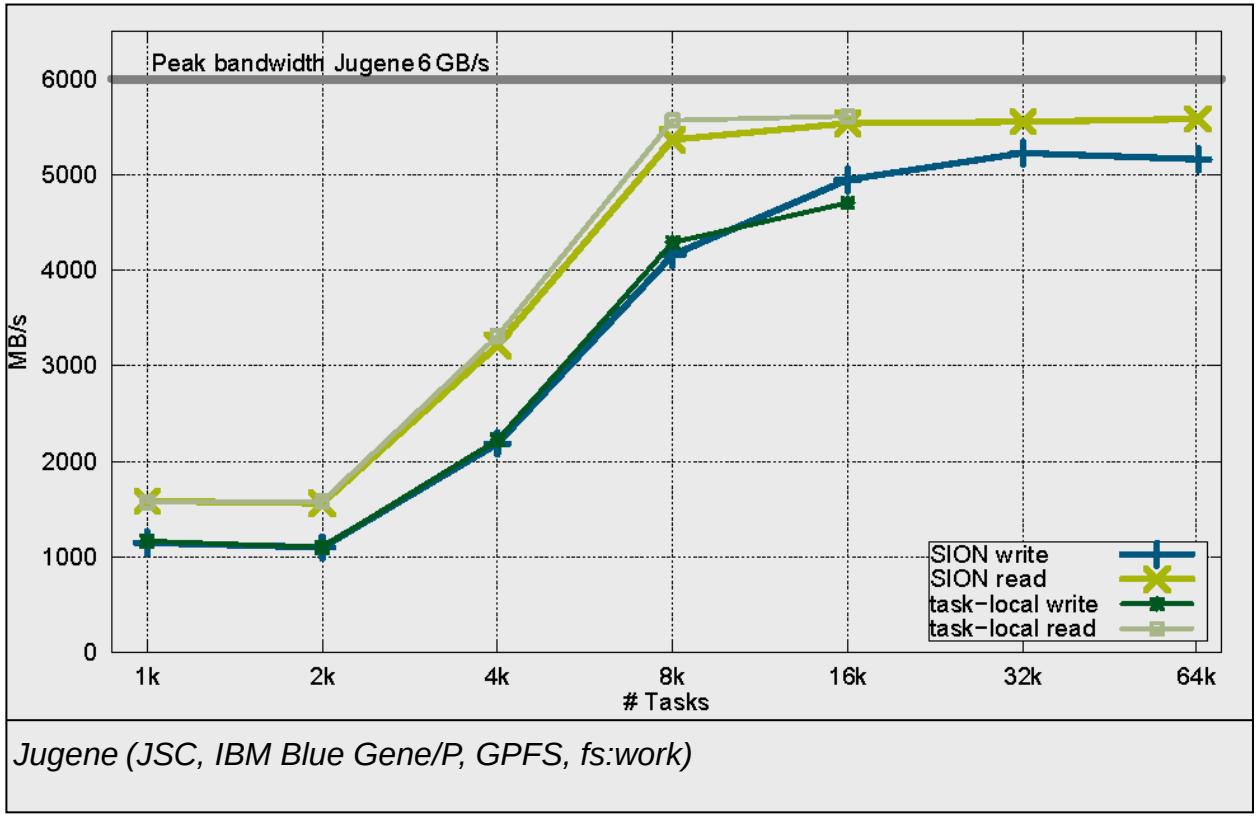


Jaguar (Oakridge, Cray XT4,2k, Lustre, fs:scr72b)





SIONlib: Bandwidth Comparison



Jugene (JSC, IBM Blue Gene/P, GPFS, fs:work)

- Task-local files compared to SIONlib (32 files, 1-2 TB data)
- No bandwidth degradation

Conclusion

- Fast parallel support is necessary for writing and reading application based checkpointing files on massively parallel system!
- Problems are the file creation time for task-local files, block alignment and meta data handling (file structure)
- SIONlib
  - == “very simple application-level file system”
  - POSIX-I/O compatible sequential and parallel API
    - ⇨Requires minimal source code changes
  - Command line utilities
  - Fits many typically usage scenarios
  - See: `www.fz-juelich.de/jsc/sionlib/`