ITM Code Camp June 2010

Approach on Parallel I/O

A. Galonska, D. Reiser, W. Frings, P. Gibbon, F. Wolf,
D. Borodin

Outline

- Motivation

- Concepts
  - Serial I/O in parallel codes
  - Parallel I/O concept
  - SIONLib
- Comparison parallel ↔ serial I/O @ ATTEMPT
  - Configuration
  - Comparison serial ↔ parallel I/O
- Conclusion & Outlook

Motivation

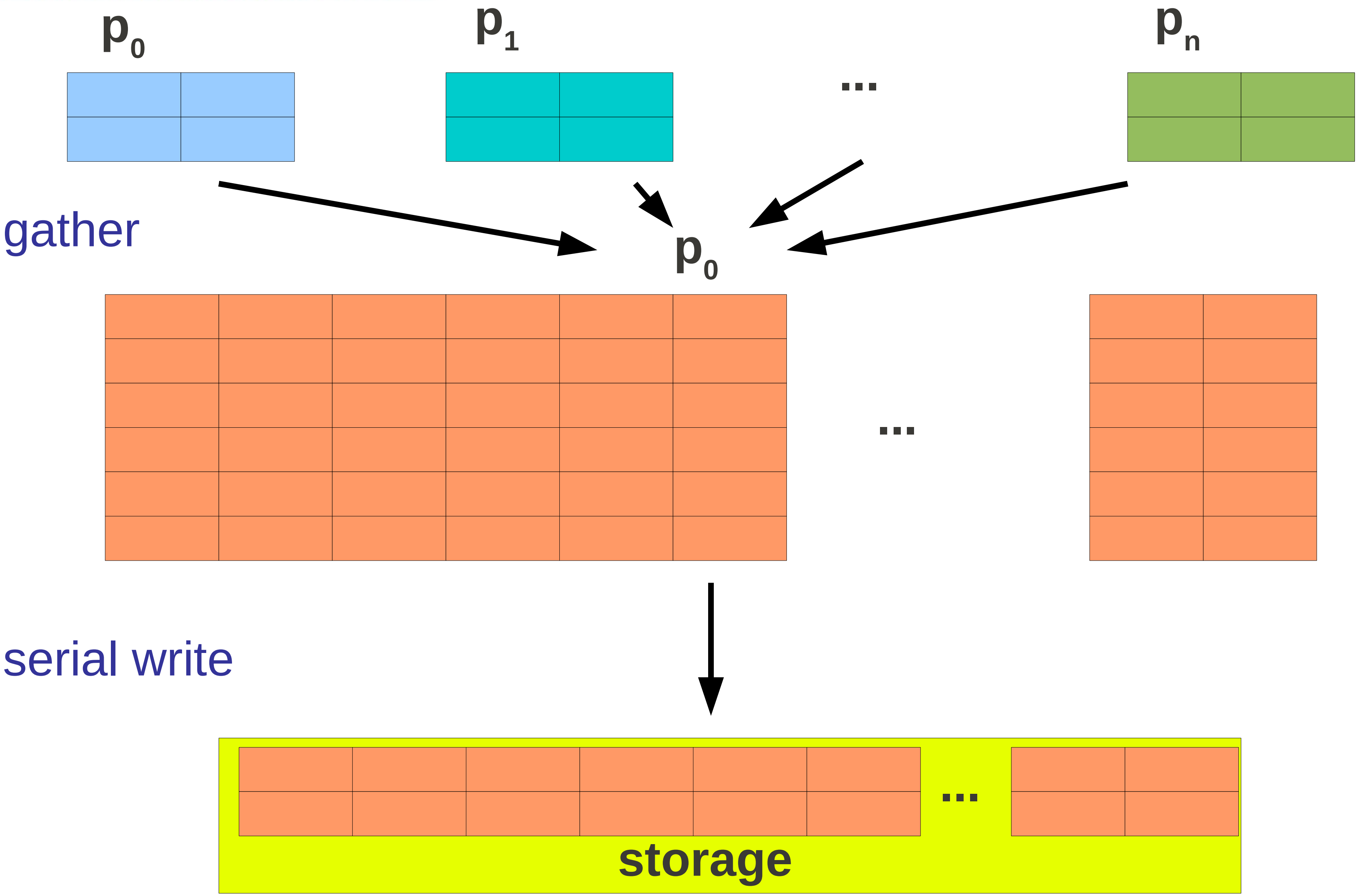
- Already unified data access for ITM codes (CPOs)

- Fusion related serial and parallel codes

- Workflow system KEPLER allows code coupling
  - Iterations (need data as precise as possible)
  - Serial data exchange of 1 to ND datasets
    - → Also for parallel codes!

→ **Parallel codes could achieve better performance when using parallel I/O**

Concepts – serial I/O



p_0 p_1 ... p_n

gather

p_0

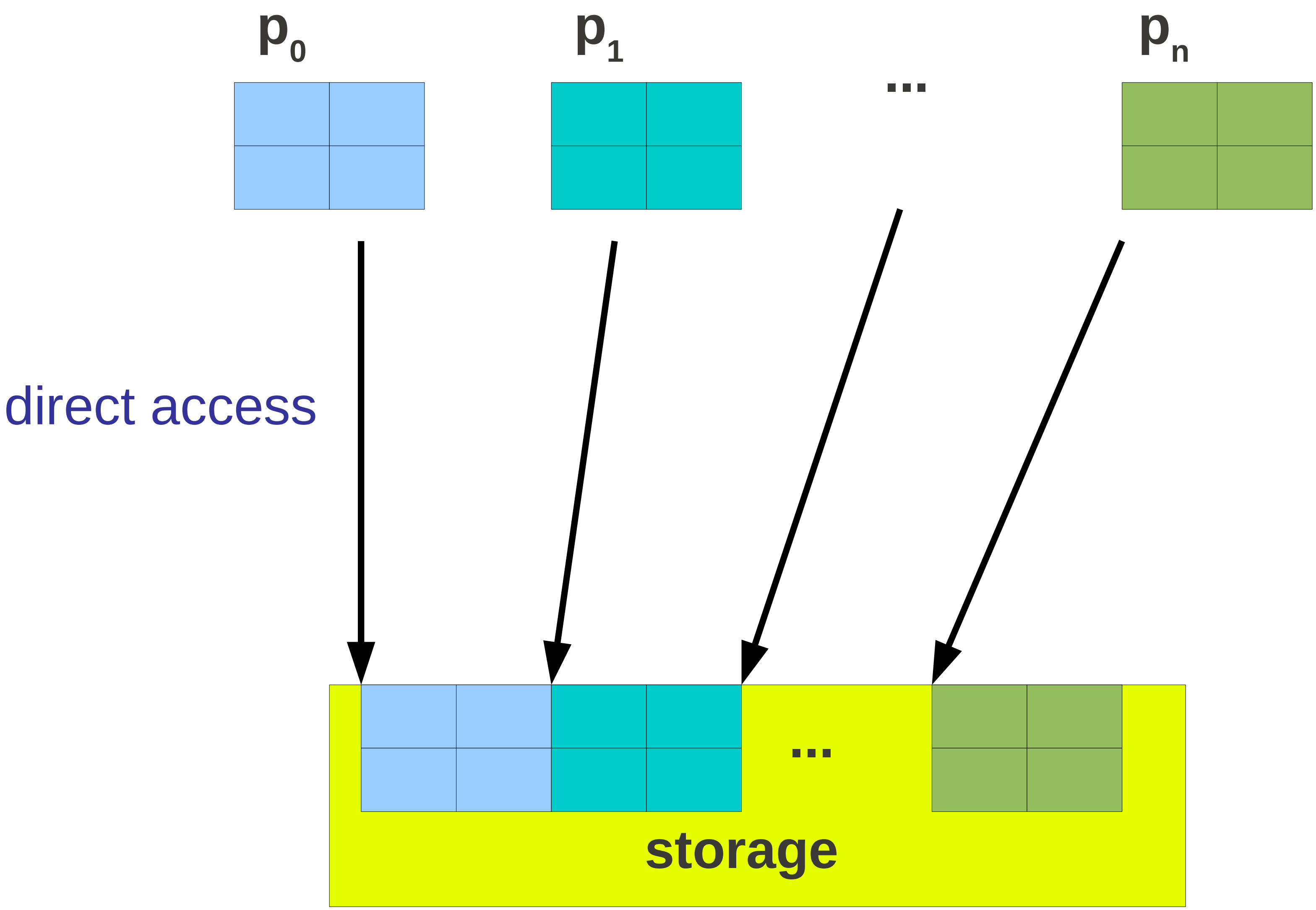
...

serial write

...

storage

Parallel I/O concept



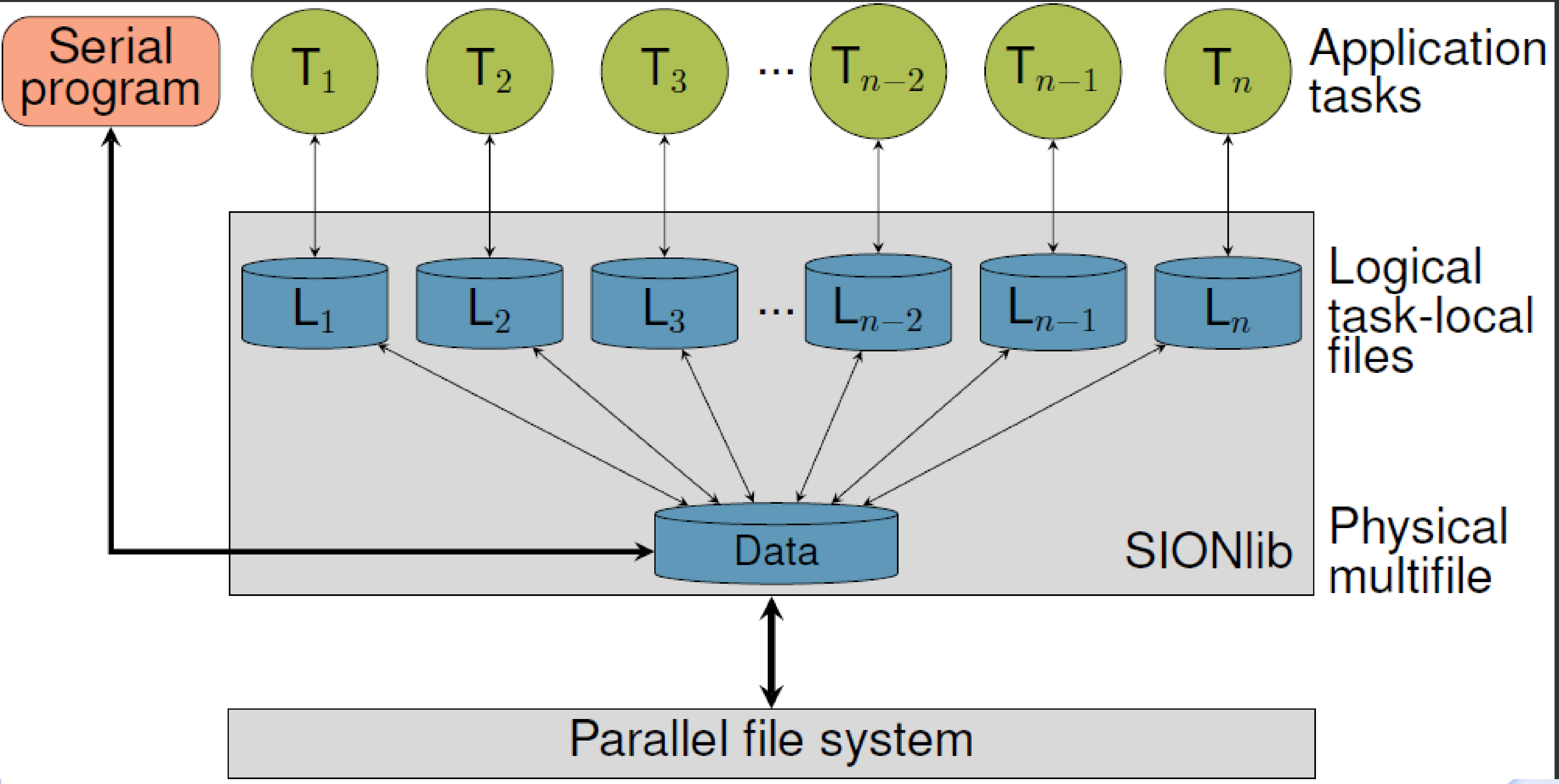
Drawbacks of serial I/O

- Huge data transfer from/to master process (gather/scatter)
  - Probably in many chunks → memory issues
- Serial write of huge data amount through ONE process
  - ➔ All other processes (probably) idle
  - ➔ Less efficient
- No explicit use of parallel filesystems
- Limits scalability (serial fraction!)
- Restart files very expensive

SIONLib (JSC/FZJ)

- No library forseen for parallel I/O trough UAL
- Provides parallel I/O to “multifile”
  - → Task local I/O expensive due to creation of too many files
  - → File system restrictions
- Aligns output to file system blocks
  - → No deadlocks
- Access similar to POSIX I/O
- Similar to ADIOS but more simple
- Supports serial access to data
- Used by wide range of scientific codes
- Approved on JUROPA/HPC-FF, JUGENE & JAGUAR
- Will be extended to handle object related data
  - → Can be used in low level interface of UAL (?)

SIONLib
Scalable I/O Library



ATTEMPT

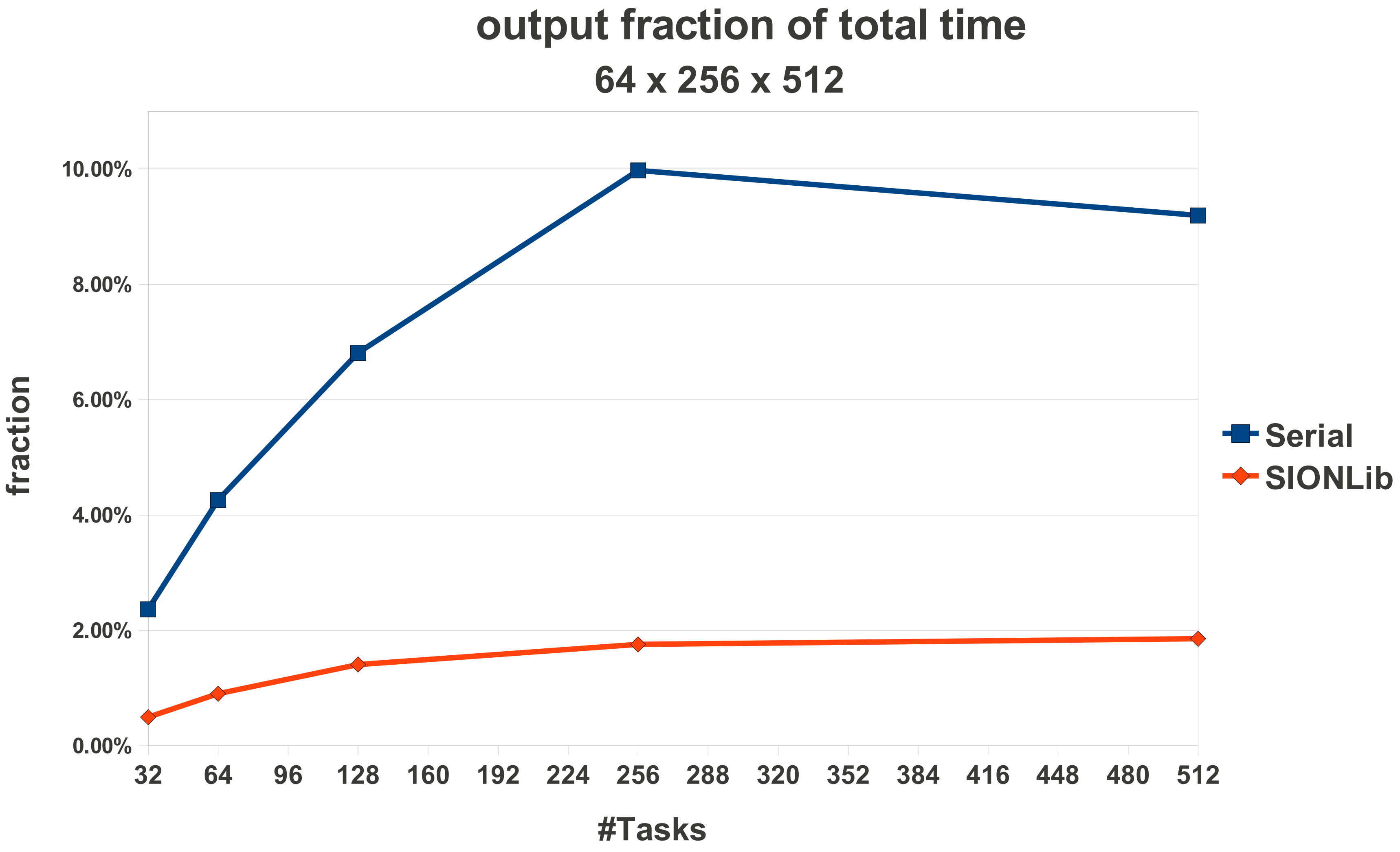
- ITM turbulence code
- Finite differences method (explicit)
  - Velocities, densities, potentials, etc...
- 3D Mesh
- MPI parallelisation → 3D domain decomposition
- Phase VI → documented KEPLER Actor

Configuration

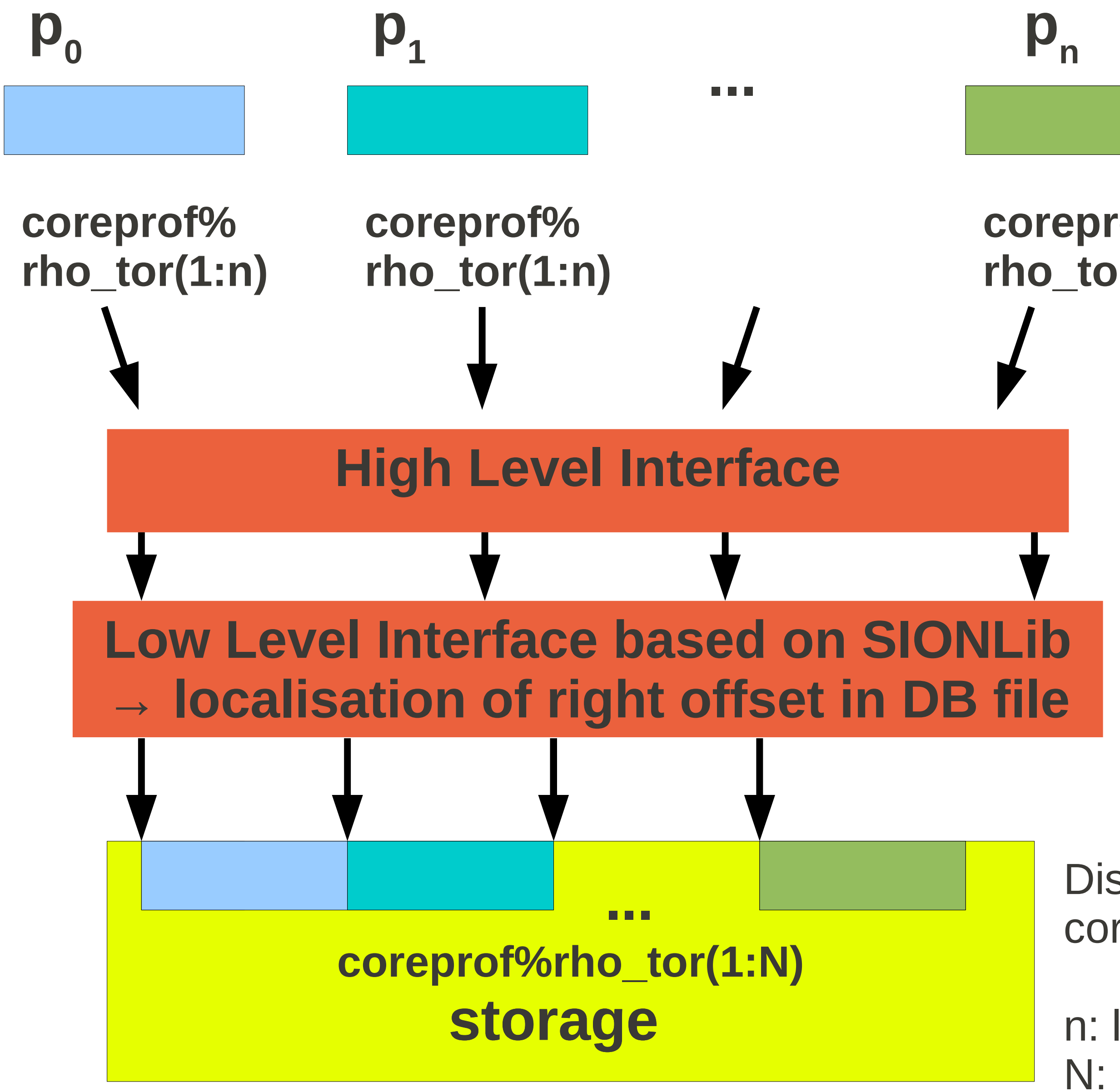
- 1000 timesteps
- Binary output of 3D mesh
- Basic conditions
  - System used: JUROPA/HPC-FF
  - 32 – 512 processes (4 – 64 nodes)
  - Mesh: 64 x 256 x 512 = ~ 8.4 mio. cells
  - Standard testbed configuration

1. **Data gathered and written by master process (serial output)**
2. **Data written directly by all tasks to multifile (SIONLib)**

I/O Comparison



CPO Approach: Example



Distributed
coreprof%rho_tor(1:N)

n: local dimension
N: global dimension

Conclusion

- Serial output inefficient

- Fraction of I/O compared to total time very high
  - ➔ Serial fraction
  - ➔ Limits scalability of code
  - ➔ Not applicable to use restart files (expensive)

- SIONLib already developed for parallel I/O
  - ➔ Needs database extension/converter
  - ➔ Can be used as low level interface (?)
  - ➔ Probably no need to change the high level interface

- Users: No need to gather distributed data anymore, local construction of CPOs