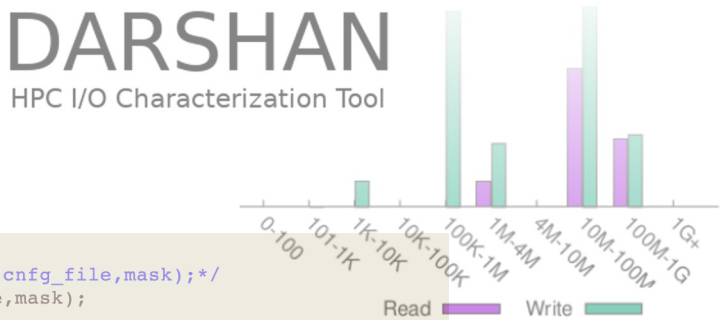
MPI-IO

Bottleneck I/O

- Know your bottleneck!
- How do we know it is I/O related
  - Use tools like darshan
  - Simple Time I/O routines

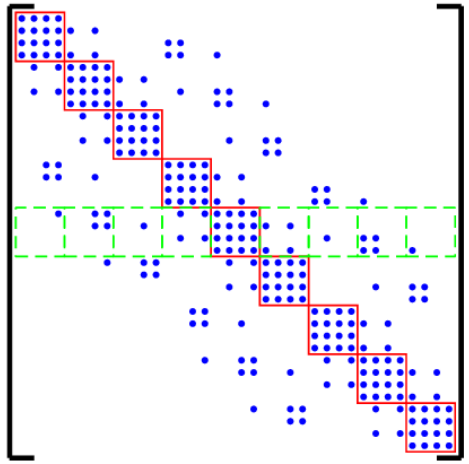



```
t_start=MPI_Wtime();
/*import_cnfg_mpi_io(cnfg_file,mask);*/
import_cnfg(cnfg_file,mask);
t_stop=MPI_Wtime();
if (my_rank==0)
  printf("Import took %.12e seconds\n",t_stop-t_start);
```

- Divide and conquer proves good stratgey in computing
  - Use MPI for large problems
  - Use “embarassing“ parallelization
- Same in I/O
  - Use MPI-IO for large jobs reading/writing one file
  - Let every task write its own file (=„ embarassing“ parallelization)
- There are pros and cons to both approaches
- Need to be careful though
  - Distribute files to be written over directories, do not create too small files ….

MPI-IO - Motivation

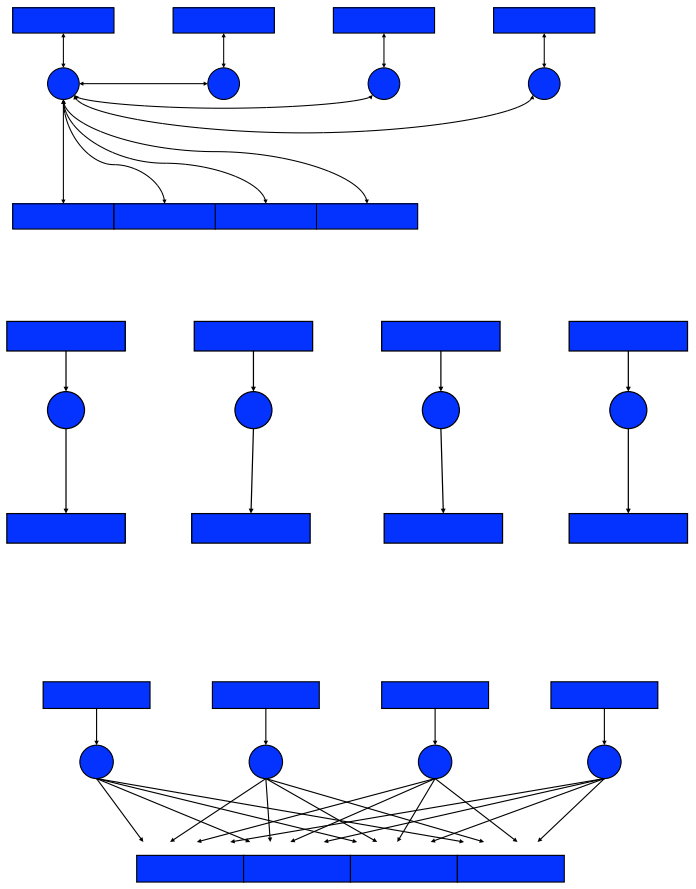
- I will try to cover MPI-IO using a real life example
- Example is openQCD, massive parallel computation
- Lattice QCD code
- Several thousand ranks compute
- Traditionally read/write of configs sequential
  - Our configs range from a few Gbyte to 90 (now 300 Gbyte)
  - Potential to save some idle computing time



MPI-IO

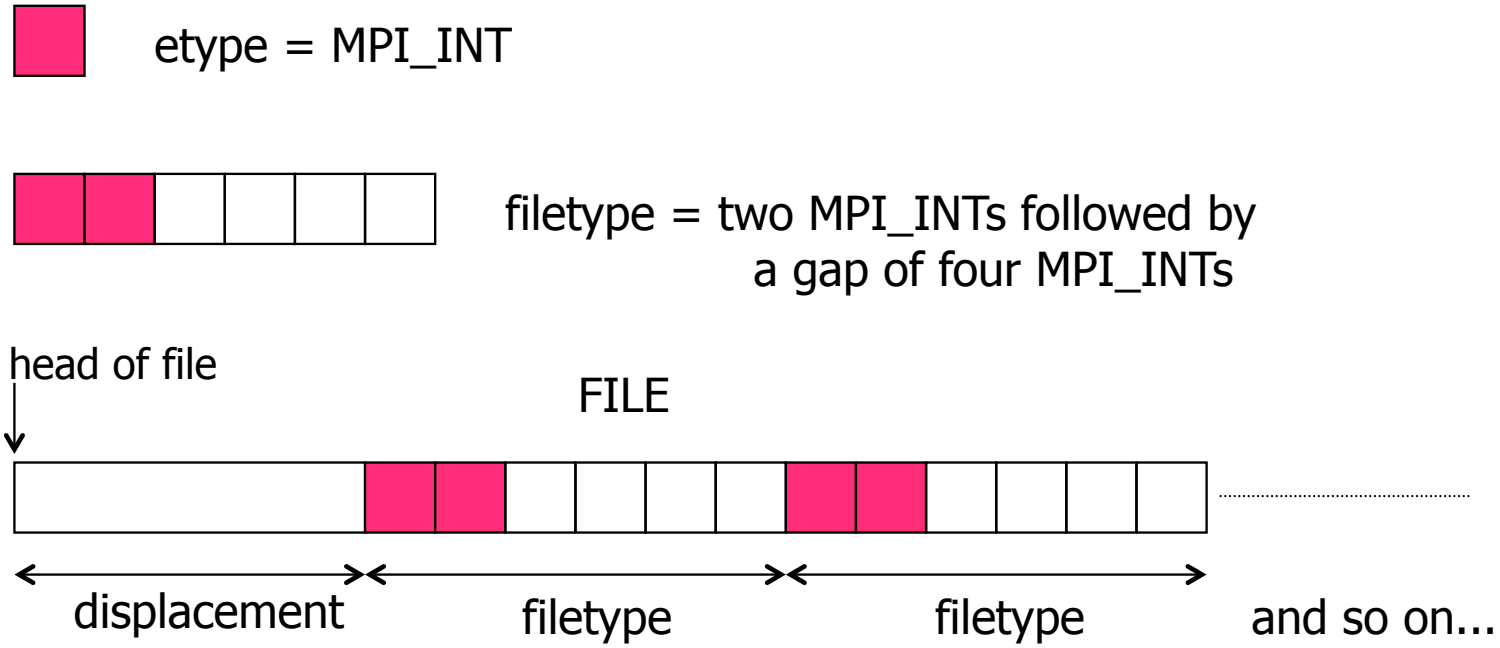
- I am not an MPI-IO expert!
- Google search for talks for 2 days
  search: mpi-io intro filetype:pdf
  - Order 10 talks (order 3 very useful) (pics stolen from first hit)
  - Played around for a week
- I'd say most important aspect is that of FileViews
- Each rank will be assigned a portion of the whole file = FileView
- MPI-IO manages data distribution

Types of I/O

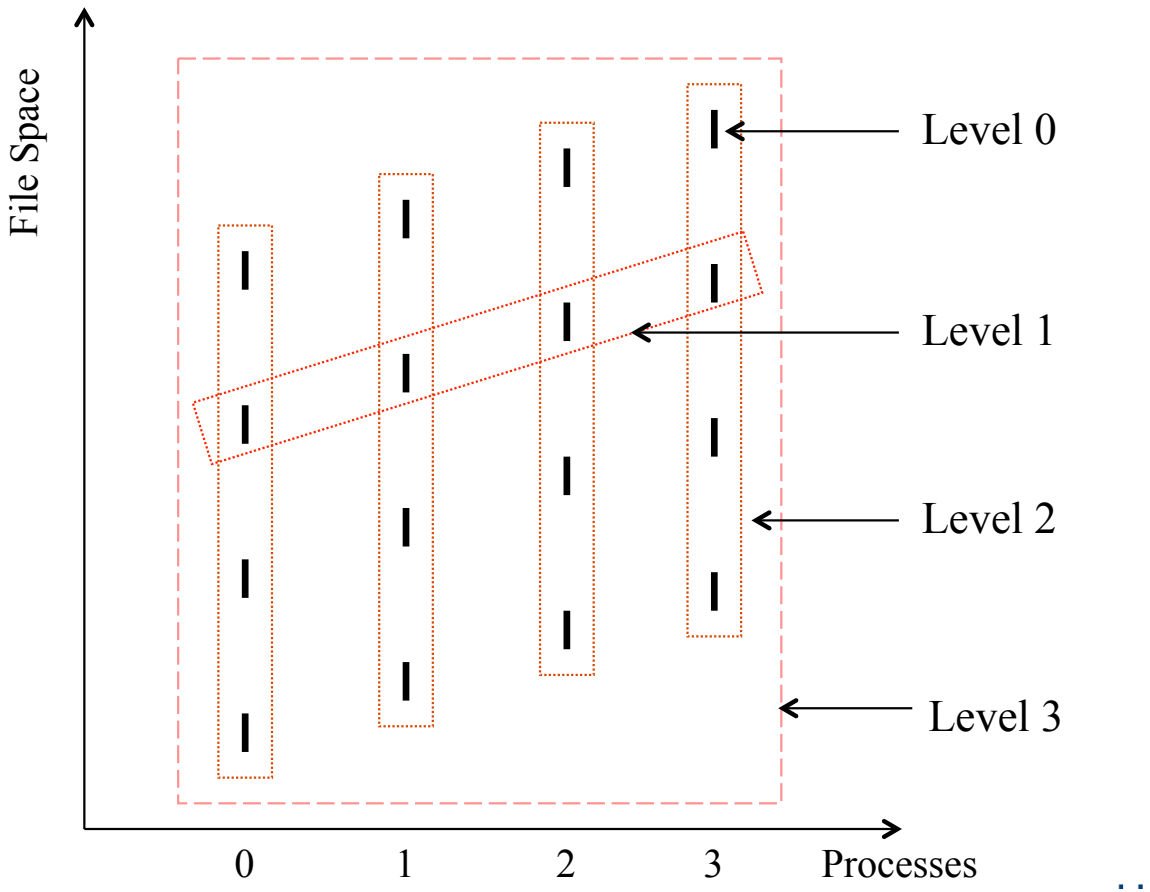


- Non-Paralell IO
  - Legacy
  - Not scaling well
- Parallel
  - Lots of small files
  - Combine afterwards
- MPI-IO:
  - Prallel
  - Potential overhead

File View


etype = MPI_INT
filetype = two MPI_INTs followed by
a gap of four MPI_INTs
head of file
FILE
displacement
filetype
filetype
and so on...

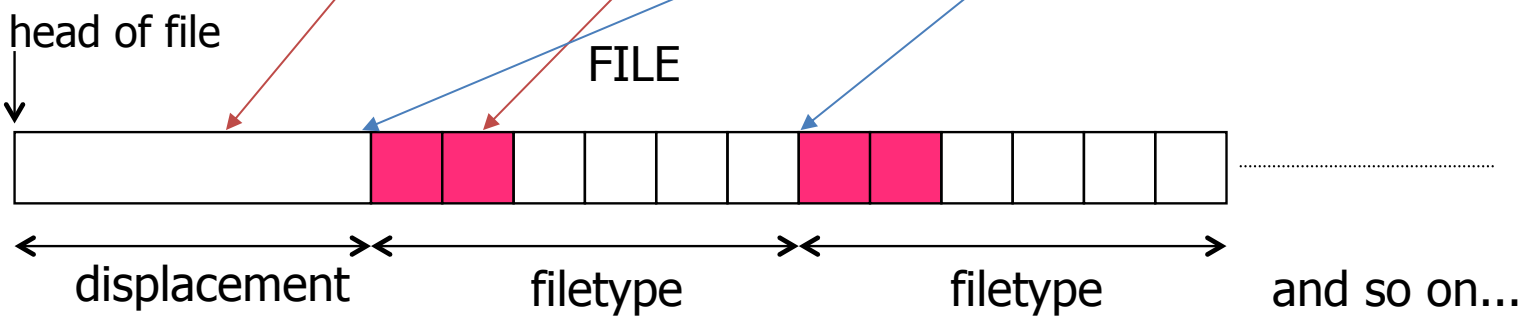
Ways to do I/O in parallel



Level 3

- Make derived datatype
- Maps non-contiguous access pattern to contiguous FileView

```
MPI_Type_create_hindexed_block(lsz[0]*lsz[1]*lsz[2],4*lsz[3]*18,roffsets,MPI_DOUBLE,&stype);
MPI_Type_commit(&stype);
MPI_File_open(MPI_COMM_WORLD,in, MPI_MODE_RDONLY, MPI_INFO_NULL, &fh);
MPI_File_set_view(fh,disp,MPI_DOUBLE,stype,"native",MPI_INFO_NULL);
MPI_File_read_all(fh,mudb,VOLUME*4*18,MPI_DOUBLE,&status);
MPI_File_close(&fh);
```



Performance Gains

- Some defaults are bad:
  - export OMPI_MCA_io_ompio_num_aggregators=8
- Filesystem Specific tuning necessary
- Machine: (type2 io openQCD-2.4) —> MPI-IO === Speedup

  Reading a 300 Gbyte file on 16384 Ranks

  Hawk : 786 sec —> 41 sec === 19 x

  SuperMUC-NG: 423 sec —> 12 sec === 35 x

SuperMUC reads with 25 Gbyte/s

HELMHOLTZ
Helmholtz-Institut Mainz