Lustre & Application I/O

2008-04-16

WangDi

Agenda

- Scientific Application IO
- LUSTRE IO Tuning
- LUSTRE ADIO driver

Scientific Application IO

- HPC Software Stack

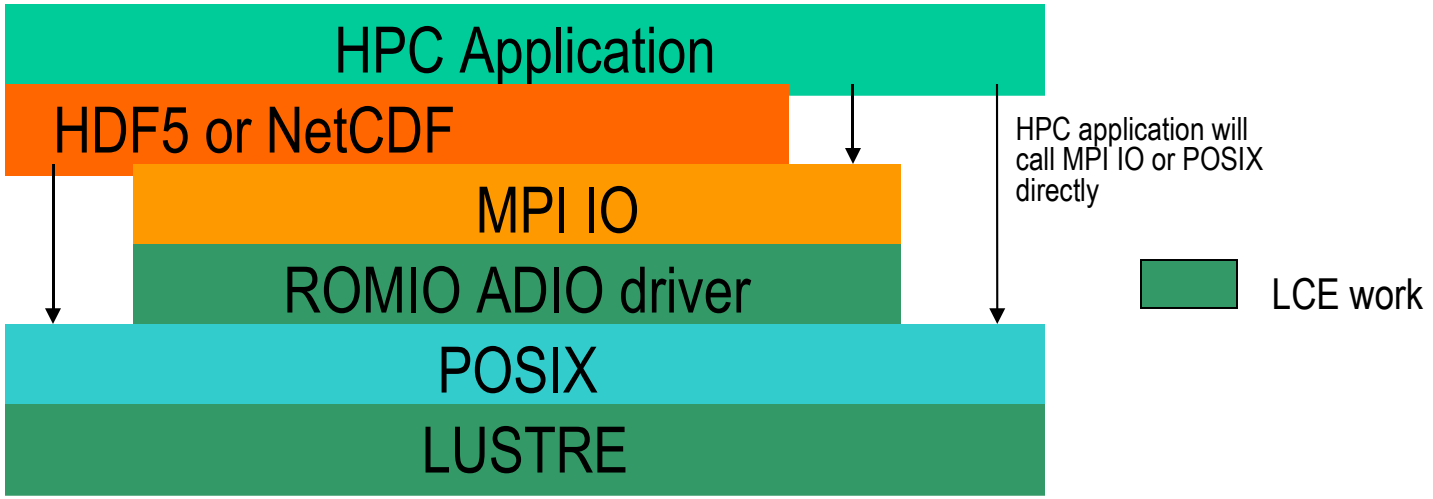


Figure1. HPC application software stack

Scientific Application IO

- Different IO
  - Required IO
    - Reading input data and Writing final results.
  - Checkpoint IO
    - Checkpoint IO is the data a program writes periodically in case of hardware and software failure, and the application can restart from the checkpoint.

Scientific Application IO

- IO pattern
  - Scientific applications usually manage multidimensional arrays, which are stored as one-dimensional arrays of bytes. Therefore, two array elements that are logically contiguous might not be stored in adjacent memory or file location.
  - Contiguous IO
    - Each client only accesses continuous part of the array.
  - Discontinuous IO
    - Each client accesses discontinuous part of the array.

Scientific Application IO

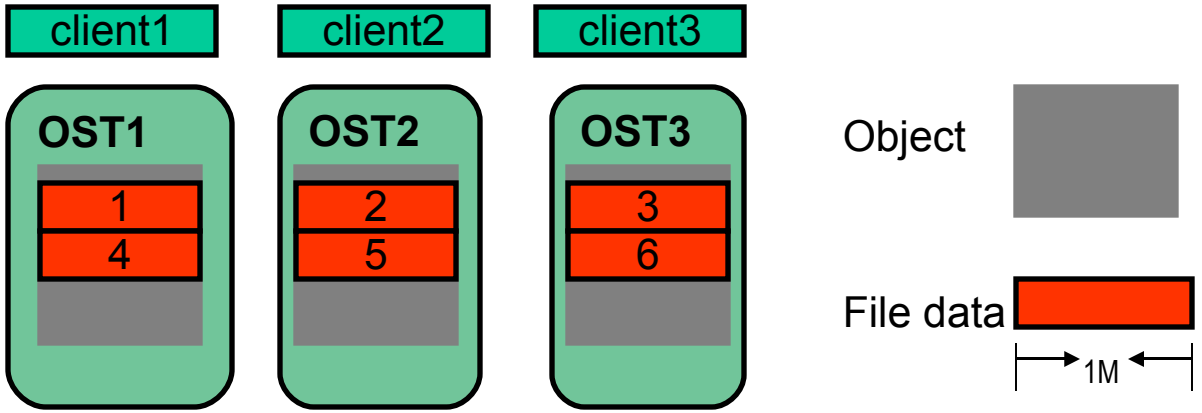
- Implement IO by scientific IO LIB (NetCDF, HDF5)
  - The IO lib manages multiple metadata and data in the same file by creating different data_sets and accessing the data_set by name.
  - The IO library usually has different I/O layers MPI IO or posix IO.
  - Some IO libraries support parallel IO eg. pNetCDF, HDF5.
- Most of the metadata operations of Scientific applications are open/create.

Lustre IO Tuning

- General Tuning
- Different IO behaviour
- HDF5 specification
- Examples for IO tuning

General IO Tuning

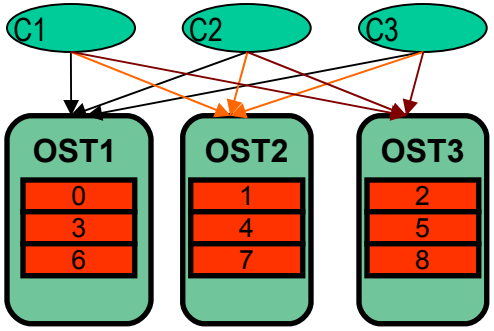
- General Tuning
  - > Lustre distribute data by stripe size and stripe count.



  - > System could achieve best IO performance
    - – Balanced OST load.
    - – Efficient RPC between clients and servers.

General Tuning

- Balanced OST load
  - Depend on the stripe size and stripe count.
  - A bad example
    - C1 write (0, 3M), C2 write (3M, 6M), C3 write (6M, 9M). IO size is 1M.
    - If we choose stripe_size 1M, stripe_count 3



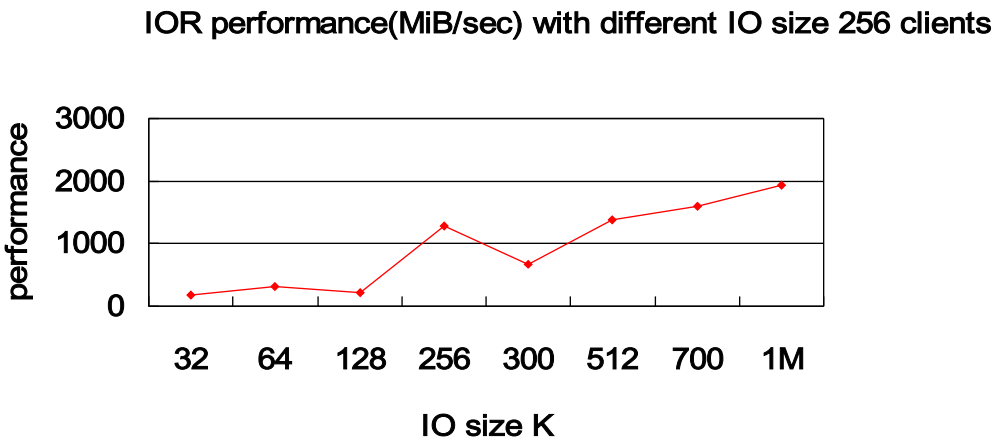
1. C1(0,1M) C2(3M,4M) C3(6M, 7M), all IO requests goes to OST1.
2. C1(0,1M) C2(3M,4M) C3(6M, 7M), all IO requests goes to OST2.
3. C1(0,1M) C2(3M,4M) C3(6M, 7M), all IO requests goes to OST3.

General Tuning

- Efficient RPC
  - Saturate Network and disk IO.
    - Currently, the RPC size is equal to IO size in the data servers (OST). Networks achieve very good throughput at much smaller packet sizes than disk system. So RPC size is largely depends on how much I/O size the disk system requires to get best performance. Current max RPC size is 1M.
    - Lustre client(CNL) could aggregate data itself and send efficient RPC(1M) to server, but which is affect by many factors, eg, amount dirty cache, lock and grant. So RPC may not efficient sometimes.
    - If the Application could provide 1M size data to lustre, which will help lustre client send efficient RPC.
  - Less RPC (stripe size aligned IO)
    - To make client access less OST in each I/O, and then improve the whole system parallelism.
    - More OST means more RPC, also more disk I/O.

General Tuning

- Different IO size comparison



- Right stripe_size and stripe_count
- Large write (1M)
- Aligned write
- Optimal number of writers

Different IO Behavior

- Different IO behaviour
  - POSIX
    - Call POSIX system call directly, no optimization.
  - Independent
    - Optimize the data pattern locally by data_sieving and stripe_size aligned.
    - But sometimes improper using of data_sieving(read-modify-write) cause unnecessary overhead. Data_sieving also includes flock in the process, which is expensive sometimes.
  - Collective
    - Optimize the data over multi-clients. Change interleave ,discontinuous and uneven IO load over multi clients into continuous and even IO load.
    - But there are also overheads for reorganizing the data over the clients.
      - Send/receive data over different node.

Different IO Behavior

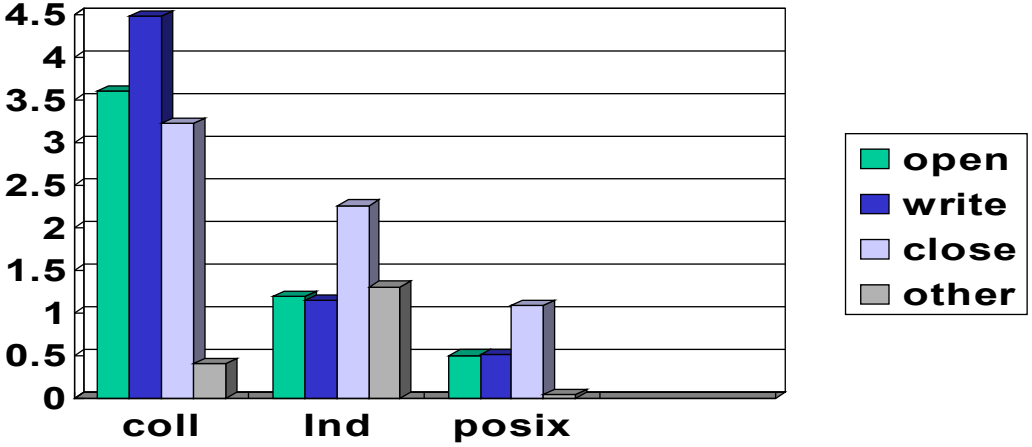
- Comparison
  - Overhead
    - POSIX: no overhead.
    - Independent: read-modify-write and flock.
      - read-modify-write and flock are expensive in Lustre.
      - Improper read-modify-write in MPI IO.
    - Collective: communication
  - IO pattern for different IO behavior
    - If each client write big(>=1M) and contiguous data, use posix IO.
    - If each client write discontinous data but non-interleave between the clients, use independent IO.
      - Disable read-modify-write and increase IO size by hints.
    - If each client write interleave data, use collective IO.
    - But It always worth to try different IO way, if you met performance problems.

HDF5

- HDF5 IO Library
  - HDF5 supports two complementary data models, a dataset and a group. The group is a collection of datasets, which can also contain other groups in a hierarchical structure. A HDF5 file can also contain attributes, containing a text name and a small collection of data. HDF5 also support different IO layers (POSIX, Independent, and collective).
  - Overhead
    - Writing Extra metadata block for each HDF5 file.

HDF5 Specification

- HDF5 supports different low-level IO.
  - Flash IO performance
    - Each client write about 320K continuous data.



driver	coll	Ind	Posix
Time(seconds)	11.7	5.85	2.13

Different layer performance with flash IO (256nodes)

HDF5 Specification

- Open
  - Open costs abnormal high time in Flash IO sometimes
    - 30%-40%time (1.3 seconds ---- 3.2 seconds)
    - Reason: In HDF5, when open existing file with (TRUNC flags), all the clients will call MPI_SET_File_size to truncate the file to zero, which occupies about 95% open time.

```
If (mpi_rank == 0) {
#ifdef H5_HAVE_MPI_GET_SIZE
     if (MPI_SUCCESS !=(mpi_code=MPI_File_get_size(fh, &size)))
          HMPI_GOTO_ERROR(NULL, "MPI_File_get_size failed", mpi_code)
#else
     if ((mpi_code=HDstat(name, &stat_buf))<0)
          HMPI_GOTO_ERROR(NULL, "stat failed", mpi_code)
     size = (MPI_Offset)(stat_buf.st_size);
 #endif
}
.............
if (size && (flags & H5F_ACC_TRUNC)) {
     if (MPI_SUCCESS != (mpi_code=MPI_File_set_size(fh, (MPI_Offset)0)))
          HMPI_GOTO_ERROR(NULL, "MPI_File_set_size failed", mpi_code)
..........
}
/* MPI_File_set_size costs a lot time about 90% time of open */
```

HDF5 Sepcification

- Open
  - Fixes
    - Flash IO, unlink the file, then open/create the file, then open time decrease from 3seconds to less than 0.1second. So Open truncate existing file should be avoided.
- Write
  - Improper read-modify-write
    - When choosing collective IO in HDF5, which will set discontinuous file view, and it triggers read-modify-write forcely, no matter whether the writing buffer is continuous or discontinuous, which impacts the writing performance a lot.

HDF5 Specification

- Close
  - HDF5 close include flush(HDF5_mpio_flush).
    - Which will cost about 40%-50% time.

```
herr_t
H5F_try_close(H5F_t *f)
{
 ……
      /*Flush at this point since the file will be closed */
      /* (Only try to flush the file if it was opened with write access) */
      if (f->intent & H5F_ACC_RDWR) {
            /*Flush and destroy all caches */
            if (H5F_flush(f, H5AC_dxpl_id, H5F_SCOPE_LOCAL, H5F_FLUSH_INVALIDATE |
H5F_FLUSH_CLOSING) < 0)
            HGOTO_ERROR(H5E_CACHE, H5E_CANTFLUSH, FAIL, “unable to flush cache”)
      }
}
```

Examples for IO Tuning

- Several Examples
  - POP
    - 42 I/O clients, each I/O client aggregate data from other computation clients. I/O size is about 60M.
      - Support Fortran binary IO (parallel), and NetCDF (non-parallel)
    - Optimization
      - Implement HDF5 parallel IO
      - Stripe_size for 60M IO
        - 60M IO size will hold too much client lock cache of multi-server on client, which will impact other clients access those server. So choose stripe_size to make each client access servers in parallel.

Examples for IO Tuning

- WRF mode
  - Produce a HDF5 file (about 8M)
    - Each client writes several K bytes(small I/O size) to the shared data_set.
  - Each client writes small and contiguous data segment
    - Lustre does not like this I/O pattern.
    - It is even worse for more clients.
  - Optimization
    - Optimize the WRF mode by the new Lustre ADIO driver.
    - Aggregate the data from multi-clients and write big I/O size.

Examples for IO Tuning

- Programming examples
  - > Fortran examples

```
if (dst_dist%proc(n) == my_task )
    ! Each block is a 4*4 real array return by get_block
    msg_buffer = get_block(n, n)
    write(id, rec=start_record+n) msg_buffer
endif
```



```
if (dst_dist%(proc(n) == my_task)
   msg_buffer = get_block(n,n)
   if (dst_dist%proc(n) < io_tasks ) ! It is io_process
      p_max = get_up(n)
      p_min = get_down(n)
      allocate(BIFFER(p_max- p_min, 4, 4))
      do p= p_min, p_max
         MPI_IRECEIVE(BUFFER(p-p_min, 4, 4),  4*4, mpi_real, n,
                                p, MPI_COMM_ALL, rcv_requests(p), ierr)
      end do
   else                                  ! Non io_process
         p_io = get_io_process(n)
         MPI_ISEND(msg_buffer, 4*4, mpi_real,  p_io, n,
                              MPI_COMM_ALL, snd_request, ierr);
   endif
MPI_WAIT(……)
```

Bad examples: **Each process only write 4*4*8 = 128 bytes.**

Gather data then writing with optimal IO process

Examples for IO Tuning

- HDF5 examples

```
if(FieldType .eq. WRF_LOGICAL) then
      allocate(BUFFER(di,x1:x2,y1:y2,z1:z2), STAT=stat)
      ! Loop to fill the buffer……

      call HDF5IOWRITE(DataHandle,Comm,DateStr,Length,  DomainStart, DomainEnd ,PatchStart,PatchEnd,MemoryOrder
      ,FieldType,XType,groupID,TimeIndex,DimRank,Var,BUFFER,Status)


! Inside call HDF5IOWRITE

      HDF5IOWRITE(….)

         CALL h5pset_dxpl_mpio_f(xfer_list, H5FD_MPIO_COLLECTIVE_F& ,hdf5err)


         CALL h5dwrite_f(dset_id,FieldType,XField,dimsfi,hdf5err, mem_space_id =dspace_id,file_space_id =fspace_id,
                        xfer_prp = xfer_list)
```

Examples for IO Tuning

- Tow problems in this process
  - > Each client write small amount of data.

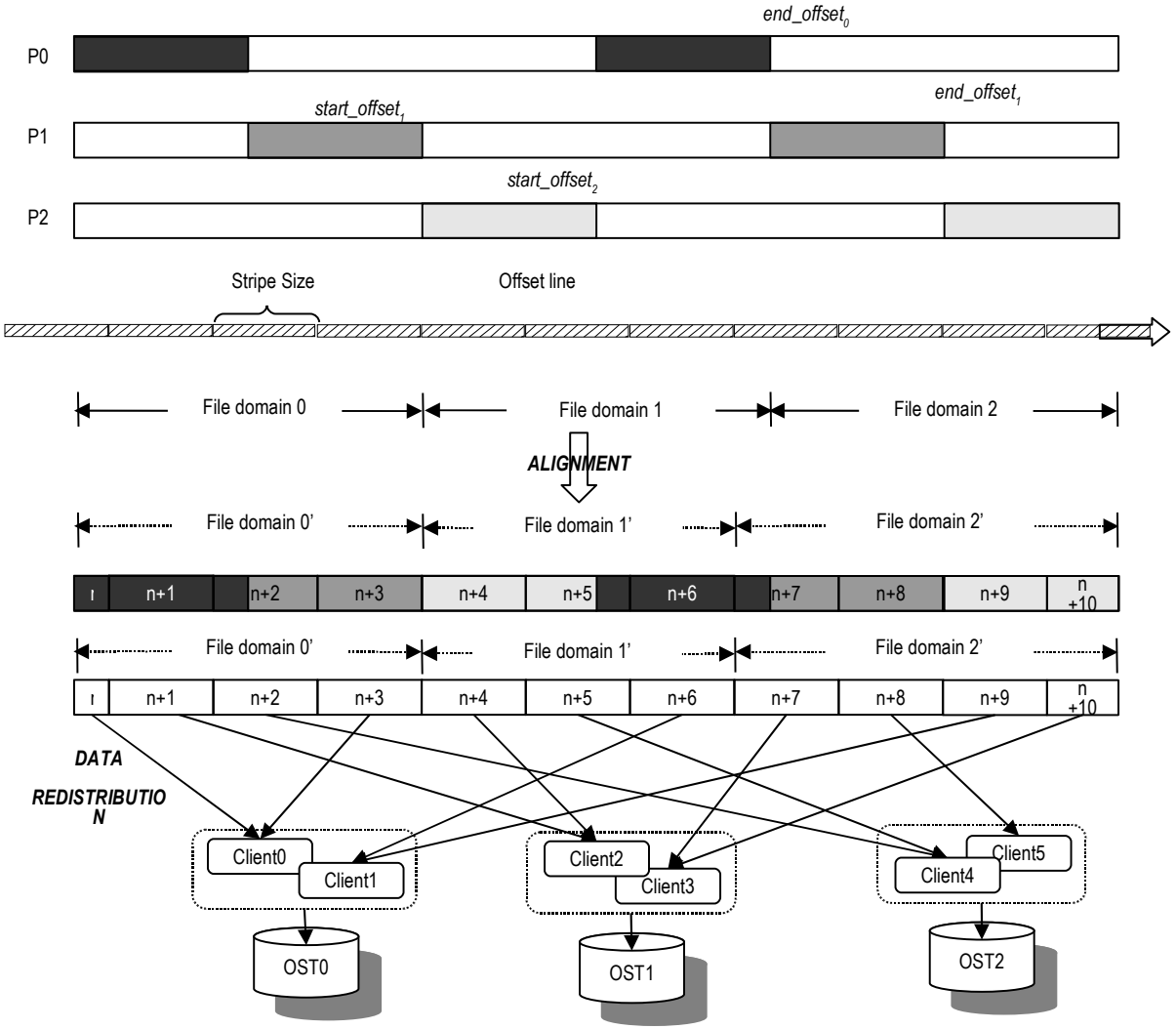
clients	4	8	16
Bytes from each client	3200	1600	800
Time consumed	2.69	6.72	9.37

    - – Gather data in WRF-mode
  - > Choose collective write for contiguous data will impose improper read-modify-write.
    - – Choose Independent or POSIX write driver here.

Lustre ADIO Driver

- **Collective Write**
  - Reorganize the data between the clients according to striping information.
    - Reorganize the data according to real data location on OST.
    - Choose IO clients to avoid unnecessary communication between clients.
    - Do stripe_size I/O
  - I/O patterns benefits from this driver.
    - Big size IO will be split to stripe_size IO.
    - For small size IO, the data will be aggregated and do big size IO.

Lustre ADIO Driver



LUSTRE ADIO Driver

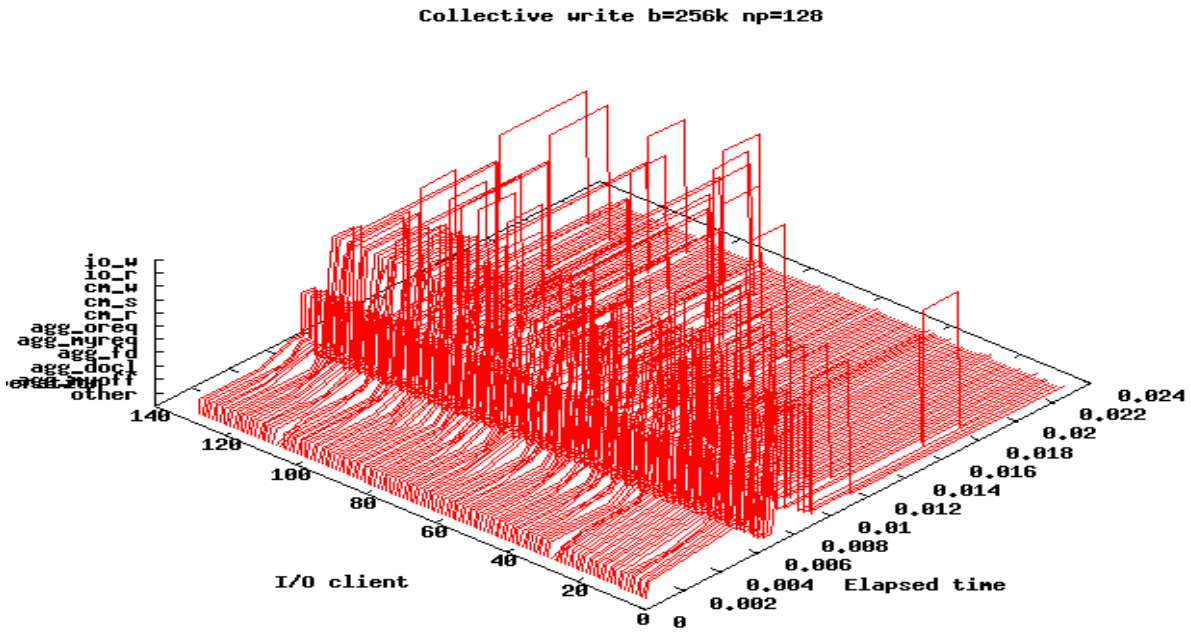
- Comparison

IO size	256 bytes	512 bytes	1024 bytes	2048 bytes
Old adio driver	0.074 sec	0.059 sec	0.026 sec	0.015 sec
New adio driver	0.002 sec	0.003 sec	0.003 sec	0.003 sec

IOR performance comparison (48 clients)

LUSTRE ADIO Driver

- Overhead
  - In the new ADIO driver, the time costs on communication (send/receive data between real IO clients and other clients) increases a lot when IO size increases, which is unexpected.



LUSTRE ADIO Driver

- The overhead occupies almost 80% for some IO nodes.
  - Communication between these nodes are also unbalanced.
- The reason is being investigating
  - Inefficient send/receive algorithm?
  - The bottleneck of JanguarCNL environment?
  - Open MPI ?

Thanks & Questions