Parallel I/O on JUQUEEN

3. February 2015
3rd JUQUEEN Porting and Tuning Workshop

Sebastian Lührs, Kay Thust
s.luehrs@fz-juelich.de, k.thust@fz-juelich.de
Jülich Supercomputing Centre
Overview

- Blue Gene/Q I/O Hardware
  - Overview, Cabling, I/O Services
- GPFS architecture and data path
- Pitfalls
- The Parallel I/O Software Stack
- Darshan
  - Overview, Usage, Interpretation
- SIONlib
- Parallel HDF5
- I/O Hints
Blue Gene/Q Packaging Hierarchy

- I/O-Nodes

JUQUEEN I/O Nodes: 248 (27x8 + 1x32)
Blue Gene/Q: I/O-node cabling (8 ION/Rack)
Blue Gene/Q: I/O Services

- Function shipping system calls to I/O-node
- Support NFS, GPFS, Lustre and PVFS2 filesystems
- Supports ratio of **8192:1** compute task to I/O-node
  - Only 1 I/O-Proxy per compute node
- Standard communications protocol
  - OFED verbs
  - Using Torus DMA hardware for performance
IBM General Parallel File System: Architecture and I/O Data Path on BG/Q

Application

BG/Q CN : CNK

BG/Q Network (dual 2GByte/s raw)

BG/Q ION : SYSIOD

10 GbE Network

GPFS NSD
Server

OS Adapter / Disk
Device Driver

Storage Subsystem

IO size
parallelism

large_region_size

pagepool (streams)
prefetch threads

transfer size

pagepool (disks)
NSD workers

hdisk dd
adapter dd

© IBM 2012
Pitfall 1: Frequent flushing on small blocks

- Modern file systems in HPC have large file system blocks
- A flush on a file handle forces the file system to perform all pending write operations
- If application writes in small data blocks the same file system block it has to be read and written multiple times
- Performance degradation due to the inability to combine several write calls
Pitfall 2: Parallel Creation of Individual Files

- Contention at node doing directory updates (directory meta-node)
- Pre-created files or own directory per task may help performance, but does not simplify file handling
- Complicates file management (e.g. archive) → shared files are mandatory
Pitfall 3: False sharing of file system blocks

- Parallel I/O to shared files (POSIX)

- Data blocks of individual processes do not fill up a complete file system block
- Several processes share a file system block
- Exclusive access (e.g. write) must be serialized
- The more processes have to synchronize the more waiting time will propagate
Pitfall 4: Number of Tasks per Shared File

- **Meta-data wall on file level**
  - File meta-data management
  - Locking

- **Example Blue Gene/P**
  - Jugene (72 racks)
  - I/O forwarding nodes (ION)
  - GPFS client on ION
  - Solution:
    - tasks : files ratio ~ const
    - SIONlib: one file per ION implicit task-to-file mapp

![Diagram showing file i-node, indirect blocks, and I/O client](image.png)

![Graph showing bandwidth vs. # IONs](image.png)

- T/F: 512/1 or 1/1
- T/F: 4096/1
- T/F: 16384/1
Pitfall 5: Portability

- Endianess (byte order) of binary data
- Example (32 bit):

  2.712.847.316

  =

  10100001 10110010 11000011 11010100

<table>
<thead>
<tr>
<th>Address</th>
<th>Little Endian</th>
<th>Big Endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>11010100</td>
<td>10100001</td>
</tr>
<tr>
<td>1001</td>
<td>11000011</td>
<td>10110010</td>
</tr>
<tr>
<td>1002</td>
<td>10110010</td>
<td>11000011</td>
</tr>
<tr>
<td>1003</td>
<td>10100001</td>
<td>11010100</td>
</tr>
</tbody>
</table>

- Conversion of files might be necessary and expensive
- Solution: Choosing a portable data format (HDF5, NetCDF)
The Parallel I/O Software Stack

Parallel application

- P-HDF5
- NetCDF-4
- PNetCDF
- ... (Various components)

MPI-I/O

POSIX I/O

Parallel file system

Shared file

Task-local files

SIONlib

Parallel application depends on the parallel file system and interfaces like POSIX I/O, MPI-I/O, and various libraries such as HDF5, NetCDF-4, and PNetCDF.
How to choose an I/O strategy?

- Performance considerations
  - Amount of data
  - Frequency of reading/writing
  - Scalability
- Portability
  - Different HPC architectures
  - Data exchange with others
  - Long-term storage
- E.g. use two formats and converters:
  - Internal: Write/read data “as-is”
    → Restart/checkpoint files
  - External: Write/read data in non-decomposed format
    (portable, system-independent, self-describing)
    → Workflows, Pre-, Postprocessing, Data exchange, …
Darshan: Overview

- Developed at ANL: [http://www.mcs.anl.gov/darshan](http://www.mcs.anl.gov/darshan)
- “HPC I/O Characterization Tool“ instruments I/O
  - Uses modified versions of I/O libraries during linking
    - No code changes needed only recompilation with wrapped compiler
  - Helps analysing I/O patterns and identifying bottlenecks
  - Analyses parallel I/O with MPI-IO and standard POSIX I/O
- Ready to use version available on JUQUEEN
Darshan: Usage example on JUQUEEN

- Load module
  - module load darshan
- Recompile
  - make MPICC=mpicc.darshan
- Tell runjob where to save the output (in submit script)
  - runjob ... --envs \
    DARSHAN_LOG_PATH=$HOME/darshanlogs ...
- Analyse output
  - darshan-job-summary.pl mylog.darshan.gz
  - evince mylog.pdf
Darshan: Interpret the summary

- Average and statistical information on I/O patterns
  - Relative time for I/O
  - Most common access sizes

- Additional metrics
  - File count
  - I/O size histogram
  - Timeline for read / write per task
  - ...

<table>
<thead>
<tr>
<th>Most Common Access Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>access size</td>
</tr>
<tr>
<td>4194304</td>
</tr>
</tbody>
</table>
SIONlib: Overview

- Shared file I/O with automatic file system block alignment
  - One single or few large files (e.g. one per ION)
    - Only open and close calls are collective
  - Support for file coalescing
    - For data sizes << file system block size

- [http://www.fz-juelich.de/jsc/sionlib](http://www.fz-juelich.de/jsc/sionlib)

```c
/* fopen() → */
sid=sion_paropen_mpi( filename, "bw", numfiles, chunksize, gcom, lcom, fileptr, ...);

/* fwrite(bindata,1,nbytes, fileptr) → */
sion_fwrite(bindata,1,nbytes, sid);

/* fclose() → */
sion_parclose_mpi(sid)
```
Parallel HDF5: Overview

- Self-describing file format
  - Hierarchical, filesystem-like data format
  - Support of additional metadata for datasets
- Portable
  - Builds on top of standard MPI-I/O or POSIX-I/O
  - Move between system architectures
    - Automatic conversion of data representation
  - Move between parallel/serial applications
    - e.g. 4096 MPI processes → 65,536 MPI processes
    - e.g. simulation → post-processing
  - Complex data selections possible for read or write
    - e.g. read only sub-array of dataset with stride

Slide provided by Jens Henrik Göbbert
Parallel HDF5: I/O Hints

- Align datasets to file system block size
  ```c
  H5Pset_alignment(...);
  ```

- Use hyperslabs
  ```c
  H5Sselect_hyperslab(...);
  ```

- Chunk datasets
  ```c
  H5Pset_cache(...);
  H5Pset_chunk_cache(...);
  ```

- Increase transfer buffer
  ```c
  H5Pset_buffer(...);
  ```

- Improve metadata handling
  ```c
  H5Pset_istore_k(...);
  H5Pset_meta_block_size(...);
  ```

slide provided by Jens Henrik Göbbert
I/O Hints

- Reduce locking for MPI-IO
  - export BGLOCKLESSMPIO_F_TYPE="0x47504653"

- ROMIO hints
  - echo " IBM_largeblock_io true" > romio.hints
  - echo " cb_buffer_size 33554432" >> romio.hints
  - export ROMIO_HINTS=romio.hints

- Note: don’t forget to add exports to runjob, via “--exp-env X”

- MPIX_Calls: splitting communicators per I/O-bridge

**FORTRAN:** MPIX_PSETSAME_COMM_CREATE(INTEGER pset_comm_same, INTEGER ierr)

**C:** #include <mpi.h>

int MPIX_Pset_same_comm_create( MPI_Comm *pset_comm_same )

sid=sion_paropen_mpi( filename , "bw“, &numfiles, &chunksize, gcom, &lcom, ...);