Benchmark-guided HDF5 Application Tuning

The HDF Group
Ruth Aydt, Babak Behzad*, Quincey Koziol, Huong Luu*
LBL
Suren Byna, Joey Huchette*, Prabhat, Yushu Yao
With input from Mohamad Chaarawi (HDF),
Mark Howison (Brown University), and Mark Miller (LLNL).
* summer interns
Project Goal for Summer

Explore viability of benchmark-guided automatic tuning of I/O parameters in HDF5 applications.

That is, answer the question:

_Is auto-tuning a reasonable approach for achieving good HPC application I/O performance without hands-on effort by application developers?_
Approach

• Build prototype automated benchmark framework
• Use framework to run experiments
  • Multiple HPC systems
  • Multiple applications/benchmarks
• Explore
  • What I/O parameters really matter
  • Performance metrics needed for insights
  • Pruning of parameter search space
  • Sensitivity of parameter “sweet spots” to
    • application / problem size / # nodes / system / load …
  • Automating creation of application I/O kernel benchmark from full application run
Benchmark Framework
Library design

Goals:
1. No changes to application code
2. Adjust parameters without recompiling

Statically linked: GNU --wrap option
Dynamically linked: LD_PRELOAD environment variable

IO Application or Benchmark

H5AutoTuner

H5PerfCap

HDF5 Library (Unmodified)

1. Read I/O parameter config file
2. Set the I/O parameters
3. Call HDF5 function

1. Start to gather statistics for call
2. Call HDF5 function
3. Finalize & store statistics for call
Benchmark Framework

Run experiments, log results

Start

I/O Parameter Control File

Adjust

Application, I/O benchmark, Appl. I/O kernel

H5AutoTuner

H5PerfCapture

H5Evolve

Executable

HDF5 files

HPC System

Performance Statistics

Collection of Experimental Inputs & Results

Execute

Create Application I/O Kernel Benchmark

Application + H5Recorder

HPC System

HDF5 "playlist"

"Augmenter"

Appl. I/O kernel
H5AutoTuner

• Reads configuration file and adjusts I/O parameters without recompiling the application.

```xml
<?xml version="1.0" ?>
<Parameters>
  <High_Level_Io_Library>
    <alignment>
      2097120,1048560
    </alignment>
    <sieve_buf_size>
      536870912
    </sieve_buf_size>
  </High_Level_Io_Library>
  <Middleware_Layer>
    <cb_nodes>
      4
    </cb_nodes>
    <cb_buffer_size>
      4194304
    </cb_buffer_size>
  </Middleware_Layer>
  <Parallel_File_System>
    <striping_factor>
      16
    </striping_factor>
    <striping_unit>
      4194304
    </striping_unit>
  </Parallel_File_System>
</Parameters>
```
• Chosen I/O parameters to adjust

- **Stripe count** (Lustre): Number of OSTs over which a file is distributed
  - 4, 8, 16, 24, 32, 48, 64, 96, 128, 256, -1

- **Stripe size** (Lustre): Number of bytes written to an OST before cycling to the next.
  - 1, 2, 4, 8, 16, 32, 64, 128 MB

- **CB nodes** (MPI-IO): Maximum number of aggregators for collective buffering.
  - 4, 8, 16, 24, 32, 48, 64, 86, 128, 256

- **CB Buffer Size** (MPI-IO): Size of the intermediate buffer for collective I/O.
  - 1, 2, 4, 8, 16, 32, 64, 128 MB

- **Alignment** (HDF5): Minimum size of object aligned at N*Alignment address in HDF5 file.
  - 256, 512, 1024 KB

- … other I/O parameters can be added
H5PerfCapture

- Captures performance statistics of HDF5, MPI I/O and POSIX I/O function calls.
- Extended from Darshan library [1] to capture HDF5 statistics
  - Cumulative Metadata/Read/Write time over all processes
  - Total number of bytes of metadata/data read/written cumulated from all processes
  - Information about Fastest/Slowest rank (time taken and bytes transferred by this rank).
  - Timing counters for individual HDF5 functions
  - ...
- Currently supports chosen HDF5 functions but can be extended if necessary
<table>
<thead>
<tr>
<th>&lt;counter&gt;</th>
<th>&lt;value&gt;</th>
<th>&lt;name_suffix&gt;</th>
<th>&lt;mount pt&gt;</th>
<th>&lt;fs_type&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>17640208042566</td>
<td>CP_HDFMETA_TIME</td>
<td>28959.403169</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_READ_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_WRITE_TIME</td>
<td>29475.545817</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_META_BYTES_READ</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_META_BYTES_WRITTEN</td>
<td>8192.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_DATA_BYTES_READ</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_DATA_BYTES_WRITTEN</td>
<td>137438953472.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_MAX_READ_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_MAX_WRITE_TIME</td>
<td>7.512889</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5DWRITE_TIME</td>
<td>29475.478084</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5DREAD_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5AWRITE_TIME</td>
<td>0.06734</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5AREAD_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5ACLOSE_TIME</td>
<td>0.035534</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5ACREATE_TIME</td>
<td>0.482534</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5AOPEN_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5AOPEN_IDX_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5AOPEN_NAME_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5DCLOSE_TIME</td>
<td>0.126270</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5DCREATE_TIME</td>
<td>2.271976</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5DOPEN_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5FCLOSE_TIME</td>
<td>28591.803655</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5FCREATE_TIME</td>
<td>363.636622</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5FOPEN_TIME</td>
<td>0.000000</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5GCLOSE_TIME</td>
<td>0.067465</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5GCLOSE_TIME</td>
<td>0.447995</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_HDF_H5GOPEN_TIME</td>
<td>0.531118</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_FASTEST_RANK 73</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_SLOWEST_RANK 86</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_FASTEST_RANK_BYTES</td>
<td>268435472</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_SLOWEST_RANK_BYTES</td>
<td>268435472</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_F_FASTEST_RANK_TIME</td>
<td>114.117900</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
<tr>
<td>17640208042566</td>
<td>CP_F_SLOWEST_RANK_TIME</td>
<td>114.137108</td>
<td>..._dataset.h5part /scratch lustre</td>
<td></td>
</tr>
</tbody>
</table>
Benchmark Framework

Run experiments, log results

Start

I/O Parameter Control File

Adjust

Application, I/O benchmark, Appl. I/O kernel

Executable

H5AutoTuner

H5PerfCapture

H5Evolve

HDF5 files

HPC System

Performance Statistics

Collection of Experimental Inputs & Results

Create Application I/O Kernel Benchmark

Application + H5Recorder

Executable

HPC System

HDF5 "playlist"

"Augmenter"

Appl. I/O kernel
H5Evolve

- Built on Pyevolve[2] that uses Genetic Algorithms to narrow the huge space of possible configurations.
- Uses crossover and mutation functions to intelligently search for well-performing I/O parameter sets.
- Other approaches for pruning space could be plugged into framework.
H5Evolve

Evolution

Random selection of initial population

Stripe Count [10]
Stripes & CB Buffer Size [8]
CB nodes [12]
Alignment [6]

5760 possible configurations

Discrete set of possible values

Members of next generation (currently 12)

Population (currently 15)

Mutation

Crossover

Reproduction

Evaluate Fitness of each member based on runtime

Entire Population

Repeat for each generation (currently max of 40)

Elite members (currently 3)
Experiments Setup

• Systems:
  • **Hopper**: NERSC's Cray XE6 system, peak performance of 1.28 PFLOPS/sec, 153,216 compute cores
    ✓ Peak I/O Bandwidth: 35 GB/sec
  • **Ranger**: TACC’s Sun Constellation Linux Cluster, peak performance of 579 TFLOPS, 62,976 compute cores
    ✓ Peak I/O Bandwidth: 35 GB/sec
Experiments Setup

• Applications
  • **VPIC-IO**: I/O replay of Vector Particle-In-Cell (VPIC), a plasma physics code with 1D I/O.
  • **VORPAL-IO**: I/O replay of VORPAL, a particle accelerator code with structured I/O with different dimensions on each rank.
  • **GCRM-IO**: I/O benchmark for Global Cloud Resolving Model (GCRM), a global atmospheric model with structured I/O with the same dimensions on all ranks.
Preliminary Results

• Configurations used for results shown:
  - Ranger: 512 cores; 32 nodes;
  - Hopper: 512 cores; 32 nodes

• Amount of data written: (VPIC, GCRM, VORPAL)
  - Ranger: 128 GB, 163 GB, 229 GB
  - Hopper: 128 GB, 130 GB, 128 GB

• Default parameters

• NERSC recommended parameters: if file size > 100GB, use stripe_large (stripe count = 156, stripe size = 1 MB).
Preliminary Results

Genetic Evolution of VPIC's Running Time

- H5evolve_time(s)
- Default_time(s)
- NERSC_time(s)

Running Time (s)

GA experiment ID

8/9/13
Preliminary Results

Genetic Evolution of GCRM's Running Time

- HSolve_time(s)
- Default_time(s)
- NERSC_time(s)

Running Time(s)

GA experiment ID
Preliminary Results

Genetic Evolution of Vorpal's Running Time

Running Time(s) vs. GA experiment ID

- HSevolve_time(s)
- Default_time(s)
- NERSC_Time(s)
# Preliminary Results

## Ranger result

<table>
<thead>
<tr>
<th>Application</th>
<th>VPIC-IO (129 GB)</th>
<th>GCRM-IO (163 GB)</th>
<th>VORPAL-IO (229 GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Converged Sets of Tuned Values from H5Evolve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stripe Count</td>
<td>64</td>
<td>-1</td>
<td>64</td>
</tr>
<tr>
<td>Stripe Size &amp; CB_Buf_size</td>
<td>64 MB</td>
<td>4 MB</td>
<td>128 MB</td>
</tr>
<tr>
<td>CB Nodes</td>
<td>48</td>
<td>32</td>
<td>256</td>
</tr>
<tr>
<td>Alignment</td>
<td>128 KB</td>
<td>512 KB</td>
<td>1024 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Measured Runtime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One file per Processor</td>
<td>51.24</td>
</tr>
<tr>
<td>Default Parameters</td>
<td>462.51</td>
</tr>
<tr>
<td>NERSC Recommended Parameters</td>
<td>362.67</td>
</tr>
<tr>
<td>Converged Set</td>
<td>123.09</td>
</tr>
<tr>
<td>Speedup over default parameters</td>
<td>3.75x</td>
</tr>
<tr>
<td>Speedup over NERSC parameters</td>
<td>2.94x</td>
</tr>
</tbody>
</table>
Preliminary Results

• Hopper result

<table>
<thead>
<tr>
<th>Parameter</th>
<th>VPIC-IO</th>
<th>GCRM-IO</th>
<th>VORPAL-IO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stripe Count</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>Stripe Size</td>
<td>4 MB</td>
<td>32MB</td>
<td>32MB</td>
</tr>
<tr>
<td>CB Nodes</td>
<td>128</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>CB Buffer Size</td>
<td>4 MB</td>
<td>32 MB</td>
<td>32 MB</td>
</tr>
<tr>
<td>Alignment</td>
<td>1024 KB</td>
<td>512 KB</td>
<td>64 KB</td>
</tr>
</tbody>
</table>

### Description

<table>
<thead>
<tr>
<th>Description</th>
<th>Measured Runtime (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Parameters</td>
<td>377.76</td>
</tr>
<tr>
<td>Minimum Observed</td>
<td>47.74</td>
</tr>
<tr>
<td>Maximum Observed</td>
<td>806.33</td>
</tr>
<tr>
<td>Converged Set</td>
<td>55.00</td>
</tr>
<tr>
<td>Speedup</td>
<td>6.87x</td>
</tr>
</tbody>
</table>

8/9/13
Early conclusions

• Speedups of 1.3X – 6.8X were measured using auto-tuned I/O parameters compared to default values for three application-based I/O benchmarks on two HPC systems.

• Converged parameter sets differ across applications and systems.

• Auto-tuning can improve parallel I/O performance without hands-on optimization.
H5Recorder

- Record all HDF5 functions (along with their parameters) in an application
H5Recorder

- Record all HDF5 functions (along with their parameters) in an application
- Storing function id, name of the function, etc. is not hard.
  - How to store those arguments that are objects?
    - The value of the data is not important to us, since we just care about the I/O operations.
    - HDF5 objects have Encode/Decode functions:
      - H5Sencode/H5Sdecode (in Replayer)
      - H5Tencode/H5Tdecode (in Replayer)
      - H5Pencode/H5Pdecode (in Replayer)
H5Replayer

- Reissues the recorder HDF5 functions to recreate the I/O kernel of the application

```bash
bbehza2@cheshvan:~/HDF/Replayer-dev$ ./run_replayer.sh
My log file is /home/bbehza2/HDF/vpicbench_uni/bbehza2_vpicio_uni_dev/binarylog.0, PATH_MAX = 4096
func_id = 42, totalsize=24
  Going to call: H5Pcreate(150994952);
no map found for 167772177
  Inserted this to map: (167772177, -1)
func_id = 49, totalsize=1441
func_id = 42, totalsize=24
  Going to call: H5Pcreate(150994955);
  Inserted this to map: (167772178, 167772178)
func_id = 47, totalsize=261
func_id = 0, totalsize=1476
  Going to call: H5Fcreate(/sample_dataset.h5part, 2, 0, 167772179);
  Inserted this to map: (16777216, 16777216)
func_id = 7, totalsize=33
  Going to call H5Gopen2(16777216, /, 0);
  Inserted this to map: (33554432, 33554432)
func_id = 0, totalsize=1858476832
```
Future Works

• Finish H5Recorder + H5Replayer
  ▪ Therefore, we can take out I/O kernels of any scientific applications

• Use H5AutoTuner on these I/O kernels and find their tuned parameters
  ▪ Therefore, we can make sure that the I/O part of the scientific application is in its best shape

• Try to generalize set of good parameters to the I/O access patterns of applications
Questions/Comments

• Any Questions/Comments?

Thank you all very much for everything

We had a great summer here 😊
References
