# h5perf\_serial, a Serial File System Benchmarking Tool

The HDF Group

April, 2009

HDF5 users have reported the need to perform serial benchmarking on systems without an MPI environment. The parallel benchmarking tool, h5perf, cannot be used for this purpose since the code is dependent on MPI functions and names. In addition, desired features like the use of extendable datasets and file drivers are not available in h5perf. These considerations call for the development of a new benchmarking tool, h5perf\_serial.

**Requirements**

Although h5perf\_serial is still under development, the following initial requirements have been implemented:

1. Use of POSIX I/O calls
   1. Write an entire file using a single I/O operation.
   2. Write a file using several I/O operations.
2. Use of HDF5 I/O calls
   1. Write an entire dataset using a single I/O operation.
   2. Write a dataset using several I/O operations with hyperslabs.
   3. Select contiguous and chunked storage.
   4. Select fixed or extendable dimensions sizes for the test dataset.
   5. Select file drivers.
3. Support for datasets and buffers with multiple dimensions.

Most of the design and options are taken from h5perf, e.g. the datatype of each array element is char. The options and parameters of the tool, dataset organization examples, and API features are described next.

**Options and Parameters**

-A api\_list

Specifies which APIs to test. api\_list is a comma-separated list with the following valid values: hdf5, posix. (Default: All APIs)

-e dataset-dimension-size-list

Specifies the sizes of the dataset dimensions in a comma-separated list. The dataset dimensionality is inferred from the list size. For example, a 3D dataset of dimensions 20 × 30 × 40 can be specified by -e 20,30,40 (Default: 1D dataset of 16M, i.e. –e 16M)

-x buffer-dimension-size-list

Specifies the sizes of the transfer buffer dimensions in a comma-separated list. The buffer dimensionality is inferred from the list size. For instance, a 3D buffer of dimensions 2 × 3 × 4 can be specified by -x 2,3,4 (Default: 1D buffer of 256K, i.e. –x 256K)

-r dimension-access-order-list

Specifies the dimension access order in a comma-separated list. h5perf\_serial starts accessing the dataset at the cardinal origin, then it traverses the dataset contiguously in the order specified. For example, -r 2,3,1 will cause the tool to traverse first the dataset dimension 2, then the dimension 3, and finally, the dimension 1. (Default: -r 1)

-c chunk-dimension-size-list

Creates HDF5 datasets in chunked layout, and specifies the sizes of the chunks dimensions in a comma-separated list. The chunk dimensionality is inferred from the list size. For instance, a 3D chunk of dimensions 2 × 3 × 4 can be specified by

-c 2,3,4 (Default: Off).

-t

Use an HDF5 dataset with extendable dimensions. (Default: Off, i.e., fixed dimensions).

-v file-driver

Specifies which file driver to test with HDF5. Valid values include: sec2, stdio, core, split, multi, family, and direct. (Default: sec2)

-i iterations

Sets the number of iterations to perform. (Default: 1)

-w

Performs only write tests, not read tests. (Default: Read and write tests)

**Data Organization Examples**

An execution of the following command

h5perf\_serial -e 16,16 -x 2,4 -r 2,1

defines a dataset of 16×16 bytes, a transfer buffer of 2×4 bytes, and dimension access order 2,1. Figure 1 shows the state of the dataset after seven write operations.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Figure 1 Data pattern for access order 2,1**

A different buffer size and access order can be specified. The following command

h5perf\_serial -e 16,16 -x 4,2 -r 1,2

defines a dataset of 16×16 bytes, a transfer buffer of 4×2 bytes, and dimension access order 1,2. Figure 2 shows the state of the dataset after seven write operations.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Figure 2 Data pattern for access order 1,2**

h5perf\_serial will check that the size of each dataset dimension is a multiple of the size of the same dimension in the transfer buffer. Also, the dimensionality of the dataset, transfer buffer, and chunks must be the same.

**APIs Features**

The available APIs for testing are POSIX and HDF5. In both cases, h5perf\_serial can write the dataset through one or several I/O operations by setting the appropriate sizes for the dataset and transfer buffer, e.g. a dataset can be written in a single operation by using a transfer buffer that matches the dataset dimensions.

The HDF5 API allows the selection of chunked storage using the option -c. When chunked storage is selected, h5perf\_serial offers the added option –t, which extends the dataset while it is being written. During each I/O access the dataset is extended minimally, if needed, to support the writing of the transfer buffer. The extension process continues until the dataset has achieved its final size. Following this scheme, the datasets on Figures 1 and 2 would have been extended four times assuming that the initial dataset size was that of the transfer buffer.

An additional feature of h5perf\_serial under HDF5 is the possibility to select a particular file driver to test the variation in I/O performance under different file access implementations.

**Tool Output**

The resulting output is similar to that of h5perf; throughput statistics are displayed by API and write/read operations. The use of multiple iterations provides information for maximum, average, and minimum throughput values.

./h5perf\_serial -e 4K,4K -x 512,256 -c 4K,256 -i 3

HDF5 Library: Version 1.9.39

==== Parameters ====

IO API=posix hdf5

Number of iterations=3

Dataset size=4KB 4KB

Transfer buffer size=512 256

Dimension access order=1 2

HDF5 data storage method=Chunked

HDF5 chunk size=4KB 256

HDF5 dataset dimensions=Fixed

HDF5 file driver=sec2

Env HDF5\_PREFIX=not set

==== End of Parameters ====

Transfer Buffer Size (bytes): 131072

File Size(MB): 16.00

IO API = POSIX

Write (3 iteration(s)):

Maximum Throughput: 52.78 MB/s

Average Throughput: 52.03 MB/s

Minimum Throughput: 50.68 MB/s

Write Open-Close (3 iteration(s)):

Maximum Throughput: 35.53 MB/s

Average Throughput: 34.05 MB/s

Minimum Throughput: 31.52 MB/s

Read (3 iteration(s)):

Maximum Throughput: 62.50 MB/s

Average Throughput: 62.24 MB/s

Minimum Throughput: 61.87 MB/s

Read Open-Close (3 iteration(s)):

Maximum Throughput: 62.44 MB/s

Average Throughput: 62.17 MB/s

Minimum Throughput: 61.80 MB/s

IO API = HDF5

Write (3 iteration(s)):

Maximum Throughput: 533.30 MB/s

Average Throughput: 523.39 MB/s

Minimum Throughput: 505.82 MB/s

Write Open-Close (3 iteration(s)):

Maximum Throughput: 89.24 MB/s

Average Throughput: 59.46 MB/s

Minimum Throughput: 35.76 MB/s

Read (3 iteration(s)):

Maximum Throughput: 607.49 MB/s

Average Throughput: 605.97 MB/s

Minimum Throughput: 603.66 MB/s

Read Open-Close (3 iteration(s)):

Maximum Throughput: 585.72 MB/s

Average Throughput: 581.98 MB/s

Minimum Throughput: 577.45 MB/s