Exercise Introduction

Before attempting the exercises in this document please ensure that you have read and understood the key topics covered in tutorial.

Contents

Task 1: IO Benchmarking (10 min)

As benchmarking could help to understand a system's performance, this task is designed to create a simple small-scale IO500 benchmarking test for estimating or learning how to run IO benchmarking using the IOR (throughput) and mdtest (metadata) tests from the IO500 suite. Understand how to interpret basic I/O performance results in an HPC context and gain hands-on experience with compiling and running a lightweight benchmark on a shared HPC resource. (Notes: When details contain typos mistake then understand it is not intensional and the reviewers has also failed to locate them.)

Task Structure:

1. Introduction to Key IO500 Tests
   - Goal: Understand the basic components of the IOR and mdtest benchmarks.
   - IOR: Measures the sequential read/write performance of the file system. Relevant for testing the bandwidth and throughput of HPC storage systems.
   - mdtest: Measures the metadata operations performance, such as file creation, file stat (status checks), and file deletion. Relevant for workloads that involve managing a large number of files or frequent file system operations.
2. Part 1: Accessing the HPC Resources and collecting the test results.
   - Step 1: Access the Cluster

     SSH into the SCC cluster:

     bash

     ```
     $ ssh username@cluster-address
     ```
     e.g., *ssh -i ssh_fileid username@login-mdc.hpc.gwdg.de*
   - Step 2: To run the IO benchmark test do the following:
     - Load the necessary modules for **MPI** and **GCC**:

       bash

```
$ module load mpi gcc
```

- Create a working directory for the benchmark tests:

  bash

  ```
  $ mkdir ./io500-small-exercise
  $ cd ./io500-small-exercise
  ```

- Step 3: Installing the Required Tools
  - Clone the IO500 repository from GitHub:

    bash

    ```
    $ git clone https://github.com/IO500/io500.git
    $ cd io500
    ```

  - Compile the IO500 benchmark:

    bash

    ```
    $ make
    ```

- Step 4: Running a Small-Scale IOR Test
  - Goal: Run a simple IOR test to measure read/write throughput on a small dataset.
  - Set up a scratch directory for the IOR test (can be on a shared parallel file system or local node storage):

    bash

    ```
    $ mkdir ./scratch/ior-small
    ```

  - Create an ior-small.ini configuration file to limit the dataset size:

    bash

    ```
    $ nano ior-small.ini
    ```

  - Inside ior-small.ini, configure a small IOR test (adjust the block size and transfer size for a quick run):

    ini

    ```
    $ [global]
    api = POSIX
    transferSize = 1m
    blockSize = 64m
    repetitions = 1
    ```

  - Run the IOR test using the following command:

    bash

    ```
    $ mpirun -np 4 ./ior -f ior-small.ini -o
    ./scratch/ior-small/ior-output
    ```

  - Analyze the IOR output, which will report the read/write bandwidth (in MB/s or GB/s).

- Step 5: Running a Small-Scale mdtest
  - Goal: Run a simple mdtest to evaluate metadata performance (file creation, stat, and deletion) on a small dataset.

- Create a directory for the mdtest run:

  bash

  ```
  $ mkdir ./scratch/mdtest-small
  ```

- Run mdtest with a reduced number of files (e.g., 1000 files) using this command:

  bash

  ```
  $ mpirun -np 4 ./mdtest -d ./scratch/mdtest-small -n 1000 -u -L
  ```

  * -n 1000 specifies the number of files to create.
  * -u ensures unique working directories for each process.
  * -L performs file stat operations.

- Once the test completes, the output will show the file creation rate, stat performance, and deletion performance.

3. Part 2: Analyzing the Results
   - IOR Results: Look at the reported write and read bandwidth. Discuss how these metrics reflect the file system's ability to handle sequential I/O operations.
   - mdtest Results: Examine the file creation rate (files/s), stat rate, and deletion rate. Discuss how file system metadata operations can impact performance in real HPC workloads (e.g., running simulations that generate many small files).
4. Part 3: Class Discussion
   - Goal: To think critically about the results and storage performance.
     - Why might the read performance differ from the write performance?
     - What are some factors that could affect metadata performance on an HPC system?
     - How would these results change if we increased the number of files or used a larger block size in IOR?
   - Explore potential optimizations (e.g., increasing the number of MPI processes, changing the block size, or utilizing a different file system).

Hints

- Example solution:
  - IOR: A small IOR test with a block size of 64MB and transfer size of 1MB should finish quickly with minimal load.
  - Example IOR output:
    * Write bandwidth: 200 MB/s
    * Read bandwidth: 180 MB/s
  - mdtest: A small mdtest with 1000 files can provide a quick evaluation of metadata performance.
  - Example mdtest output:
    * File creation rate: 30,000 files/s

    - File stat rate: 25,000 files/s
    - File deletion rate: 28,000 files/s

- Discussion Points:
  - You should notice that small-scale IOR runs are sufficient to demonstrate throughput bottlenecks, and mdtest offers insight into metadata handling efficiency.
  - You can also explore how increasing the number of files, adjusting block size, or running more processes would affect performance on the limited resources available.
- Further Reading/References:
  - IO500 Documentation (https://github.com/VI4IO/io-500-dev/blob/master/doc/README.md)
  - IOR Benchmark User Guide (https://media.readthedocs.org/pdf/ior/latest/ior.pdf)
  - mdtest User Guide (https://www.illko.cz/images/dokumenty/mdtest_manual_en.pdf)

Optional Task 2: Optional: Compute Benchmarking (10 min)

This is a difficult **additional** task which will support your understanding in the topic.

As benchmarking could help to understand a system's performance, this task is designed to create a simple benchmarking test program for estimating or measuring the compute performance of a system. For this, you could use your own choice of programming language and the guidelines for the tasks are as follows:

1. First, think about the pattern you would like to measure, e.g., computational time.
2. Define the measurement protocol and metrics: How to time the operations, granularity of the measurement. How do you increase confidence that the returned value reflects the true value in the system?
3. Think about the expected performance on your system and document it.
4. Implement the benchmark.
5. Run the benchmark in such a way that it meets your experiment's measurement protocol.
6. Compare the results with your observations and document them briefly.
7. Conclude the experiment in your measurement protocol. Compare your steps and methodology with the scientific method.

For example, you could use the sample code shared with you during the demo session. You could find them in the course website:

1. Use the factorial function as shown in there.
2. Take the readings of the factorials from 1 to 151.
3. Repeat previous step (2) for 5 times. i.e., take reading for five rounds with different timestamps.
4. Now, plot these timestamp records together for factorial 1, 50, 100 and 150 as highlighted in gray color in the table 1.
5. Show the readings of all the five timestamps for a single factorial with a single line graph.
6. Repeat previous step (5) for other three factorials as well.
7. Then plot these readings in a line graph chart. You will get a clear pattern for your system's performance.
    - You could generate the showing factorial range in x-axis, time range on y-axis and each line graphs

for each factorials for different timestamps with different legend names.

8. Illustrate your graph results and share with others.

Table 1: Sample for taking readings

Factorial	Timestamp1	Timestamp2	Timestamp3	Timestamp4	Timestamp5
1.	0.00	0.00	0.00	0.00	0.00
2.	0.02	0.01	0.00	0.01	0.01
..	..	..	..	..	..
..	..	..	..	..	..
50.	0.05	0.06	0.05	0.07	0.05
..	..	..	..	..	..
..	..	..	..	..	..
100.	0.10	0.09	0.10	0.11	0.10
..	..	..	..	..	..
..	..	..	..	..	..
150.	0.15	0.17	0.13	0.16	0.15
..	..	..	..	..	..
Total	**151**				

Hints

- You could create the graph plot as shown in demo session.

Portfolio (directory: `1/submission-folder`)

`1/submission-folder/findings.txt` You could share your findings during the session for the feedback.

Hints

- You could create the graph plot as permitted by your time in demo session.