High Performance Computing

# Assignment 3

## Theory Questions

1. Profiling is program analysis to check how an application is doing, its performance and identify areas that need optimizations. The classification into the different types of profilers is done based on how they collect their information during the execution of a program.

**Event Profiling**

Java, .NET, Python, Ruby are Even-based profilers. These programming languages use an interface or profiling agent like a Profiling API during runtime for trapping events like calls, class-load, unload, thread enter and leave

Event Profiling captures a subset of event from an application during the execution of the program and collects data like memory usage.

**Advantage** :

* It can be performed with minimal overhead on the application making it suitable for high performance applications.
* It ca identify bottlenecks during program, how much space a program is taking, where it is going wrong
* It can be suitable for HPC applications since it can be used with low overhead

**Disadvantage**:

* Since it considers a subset of the program, it does not capture all events in the application therefore it can miss some important parts of the program during execution
* Therefore when using it for HPC, basically a heterogenous network, it can be disadvantageous as it only takes a subset of event, but in HPC multiple critical events are running in parallel which might make lead to missing on some important data

**Instrumentation Profiling**

In Instrumentation Profiling, we add instruction to the program itself to collect required information during execution.

**Advantage**:

* Instrumentation allows us to determine the level of control and the amount of time resolution that is available to the profilers
* We can get detailed information and all information we like in a comprehensive manner

**Disadvantage**:

* This can affect the program performance
* In high performance application, it can case a lot of unnecessary and additional overhead that is unsuitable in a already ‘busy’ environment
* Instrumentation profiling can be time consuming
* Instrumentation profiling can be difficult to implement in a complex HPC environment

**Statistical Profiling**

When sampling profiling is done, the results are not exact but a statistical approximation. The actual value is usually n times the sampling period.

**Advantage**:

* Statistical Profiling allows in identifying the performance data which are statistically accurate
* Helpful in identifying trends and patterns in the data

**Disadvantage:**

* Time consuming as it takes time to analyse the data after collection of data
* In heterogenous environments, it can be difficult since there are multiple applications, machines, too much going on to be able to collect all data accurately and analyze it before producing the result
* This means, there will also be unnecessary overhead

1. Some obvious parameters to consider and investigate when evaluating the performance of a file system and storage configuration is:

* Latency: trying to get the data form the storage
  + Read/Write/IO operations: Time it takes when trying to open, read or create and write in a file of a very large size. How many operations can be performed pr second
* Throughput: Perform benchmarking to evaluate the amount of data that can be read or written from/into a file typically for a huge size of data
* For a large network with multiple nodes, checking if nodes are able to communicate with each other at any given time over a network with consistency and least possible latency/interruptions/loss of data
* Check the bottleneck links for the file system to see how quickly it gets saturated and increase the latency
* What level of storage the file systems are stored in? What I mean by this is let’s say data that is constantly fetched is stored in cache to that it is immediately available when it is requested

ZFS - robust data integrity features and support for advanced storage features such as snapshots and replication. Works well on large disks and joining multiple disks

BTRFS is known for its support for transparent compression and snapshotting. Built to deal with fault tolerance, management and data protection in storage systems

ReFS is designed specifically for use with Microsoft's Windows operating system, and it includes features such as built-in support for advanced storage technologies such as Storage Spaces Direct over multiple remote users on a network connection

## Programming Questions

1. **Use VampirTrace (or score-p if it doesn’t want to behave for you) or Google profiler to profile one of your applications either in this course or from another (note that since these are fairly simple programs it won’t show you much, but it’s something to say you are familiar with profiling). The point of this is just that you’ve seen how to profile an application, not that the results are particularly meaningful**

Running Lab4Array.cpp with valgrind

Text

Description automatically generated

Text

Description automatically generated

**--sigill-diagnostics=<yes|no> [default: yes]**

Enable/disable printing of illegal instruction diagnostics. Enabled by default, but

defaults to disabled when --quiet is given. The default can always be explicitly

overridden by giving this option.

When enabled, a warning message will be printed, along with some diagnostics, whenever

an instruction is encountered that Valgrind cannot decode or translate, before the

program is given a SIGILL signal. Often an illegal instruction indicates a bug in the

program or missing support for the particular instruction in Valgrind. But some

programs do deliberately try to execute an instruction that might be missing and trap

the SIGILL signal to detect processor features. Using this flag makes it possible to

avoid the diagnostic output that you would otherwise get in such cases

Text

Description automatically generated

Text

Description automatically generated

**--log-file=<filename>**

Specifies that Valgrind should send all of its messages to the specified file. If the

file name is empty, it causes an abort. There are three special format specifiers that

can be used in the file name.

Text

Description automatically generated

1. **Write a simple benchmark application to test the performance of a relatively large (>100MB) file copy, and a simple test of repeatedly reading and writing from several dozen small files (< 1MB each). This is related of course to theory question 2, and feel free to improve on the experiment or visualise the results. (You can do this in python easily enough). Note: You can do this in windows or Linux or Native Mac OSX**

Language: Python | Doing benchmarking using default using ‘timeit’

For big 100MB file

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

For multiple small files (1mb) eachText

Description automatically generated

Text

Description automatically generated

**CPROFILE** – I did this on the multiple small file, and let me tell you, of course it took time because it was getting all details and all. And I did not have the guts to try on 100MB. But I did, it’s the next set of screenshots.

Text

Description automatically generated

Text

Description automatically generated

SO, I did try profiling for 100MB and as you can see it has been running for minutes! And I got scared of my laptop blowing up so I stopped but there:

Text

Description automatically generated