Research Project - differential execution analysis in large scale traces

Research Project - differential execution analysis in large scale traces

Identifying performance bottlenecks in a parallel application running on a supercompute is tedious, especially because it requires analyzing the behaviour of various software components, as bottlenecks may have several causes and symptoms. For example, a load imbalance may cause long MPI waiting times, or contention on disk may degrade the performance of I/O operations. Detecting a performance problem means investigating the execution of an application and applying several performance analysis techniques. To do so, one can use a tracing tool to collect information describing the behaviour of the application. At the end of the execution, a trace file in a specific format is available to the application user, which can be used to conduct a complete post-mortem investigation. Several challenges emerge from the generation and use of traces. Tracing applications may alter the performance of the application, and can create thousands of heavy trace files, especially at a large scale. Most importantly, the post-mortem analysis needs to load these thousands of trace files in memory, and process them. This quickly becomes impractical for large scale applications, as memory gets exhausted and the number of opened files exceeds the system capacity.

As part of her PhD, Catherine Guelque designed PALLAS [1], a generic trace format tailored for conducting various post-mortem performance analysis of traces describing large executions of HPC applications. During the execution of the application, EZTrace [2] intercepts function calls and uses PALLAS to store events and detect their repetitions on-the-fly. When storing the trace to disk, PALLAS groups the data from similar events or groups of events together in order to later speed up trace reading. The PALLAS format allows faster trace analysis compared to other evaluated trace formats. Overall, the PALLAS trace format allows an interactive analysis of a trace that is required when a user investigates a performance problem.

Objective

The goal of this research project is to explore differential execution analysis in Pallas trace.

Pallas traces are structured by detecting sequences of events that repeat in each application thread. As a result, a series of events such as:

```
A B C A B C D E ABC  ...
```

will be structured as

```
2xS1 D E S1
S1: A B C
```

It is thus possible to compare all the occurences of S1 (which is typically a function execution) is order to measure the duration of the function.

In a parallel application, each thread will generate its own trace. Depending on the order of function calls in the different threads, similar events and sequences may get different identifiers. For example, thread T1 may generate:

```
2xS1 D E S1
S1: A B C
A: Enter MPI_Send
B: execute mpi_send(dest=17, tag=24, len=1024)
C: Leave MPI_Send
```

while T2 may generate

```
2xS2 A B S2
S1: C D E
C: Enter MPI_Send
D: execute mpi_send(dest=15, tag=24, len=1024)
E: Leave MPI_Send
```

We observe that T1 and T2 have the same behavior (except of the destination of MPI messages), but event identifiers differ.

The first step towards comparing thread traces is to build common identifiers for events and sequences, and to find the sequences that are similar. While finding sequences that are exactly the same is easy, finding sequences that are *more or less* the same is trickier (is event B in T1 the same as event D in T2 ?).

Next, we plan to investigate differential performance analysis by comparing the duration of similar sequences that where executed in different threads. The goal is to identify outliers that could lead to identifying a performance problem.

Differential performance analysis could also be used for comparing two different traces that where obtained in two different context (eg one trace with good performance, and one trace with bad performance) in order to identify the cause of the bad performance.

References

[1] Pallas

- **Pallas: a generic trace format for large HPC trace analysis.** Catherine Guelque, Valentin Honoré, Philippe Swartvagher, Gaël Thomas, François Trahay. In *IPDPS 2025*.
- Git repository
- Documentation

[2] EZTrace

- **EZTrace: a generic framework for performance analysis.** François Trahay, François Rue, Mathieu Faverge, Yutaka Ishikawa, Raymond Namyst and Jack Dongarra. In *CCGrid 2011*.
- Git repository
- Turorial
- Documentation