TUTORIAL, HINTS AND GOOD PRACTICES ON

PROFILING TOOLS

WHAT IS AND WHY WE SHOULD USE PROFILING TOOLS?

- In a nutshell, code profiling tools perform *dynamic analysis* to gathers richness information about the program, allowing developers to optimize their codes
    - **Execution time**
        - **Program's complexity**
        - **Identify bottlenecks**
        - **Number of calls and time spent in each instruction**
    - **Memory usage**
        - **Space complexity**
        - **Overhead with allocations**
        - **Cache efficiency**
    - **Help to prevent bugs**

A POSSIBLE OUTLINE FOR THE TUTORIAL (SHOULD BE ADAPTED TO MATCH THE ATTENDEES' NEEDS)

1. Introduction to Profiling tools:
   - Motivation with examples to demonstrate the importance and benefits of profiling a program.
   - Introduction to the main concepts and information that can be obtained
2. Present two of the most well known profiling tools:
   - *Valgrind* - C/C++
   - *profile* and *cProfile* - Python
3. Good practices to execute and interprete a profiling tool
4. Profiling for distributed system and parallel programming
5. Final remarks