# Application Profiling with Score-P and Cube: Dr-Ing Bernd Mohr

Johansell Villalobos
*University of Costa Rica*
San José, Costa Rica
johansell.villalobos@ucr.ac.cr
0009-0002-3398-0714

***Abstract*—This lecture covers experimental performance analysis methods for optimizing parallel applications, focusing on a five-phase measurement cycle: instrumentation, measurement, analysis, presentation, and optimization. Probes inserted into the code facilitate data collection, with analysis informing optimization efforts. Performance triggers can be external or internal, and data collection methods include tracing (detailed event sequences) and profiling (summative but less dynamic). A combined approach is recommended for comprehensive analysis. The Score-P ecosystem, integrating tools like Vampir, Scalasca, and TAU, enables flexible measurement and supports multi-parallel paradigms, helping users visualize and interpret data for optimization across HPC systems.**

***Keywords*—High Performance Computing, Graphic Processing Units, High-energy Physics**

The lecture starts off with some background on parallel performance tools and how performance analysis is carried out. Generally there are two ways that performance analysis is conducted, analytically and experimentally. The lecture focuses on experimental performance analysis. The performance measurement cycle is composed of a series of steps which are:

- Instrumentation
- Measurement
- Analysis
- Presentation
- Optimization

In the instrumentation phase, extra code is inserted as probes into the application. Then relevant data is collected with those probes to then be analyzed. After analysis performance problems are clear and now these are presented to the user which needs to fix the problems in the optimization phase. This cycle is repeated until problems are fixed or a reasonable speed has been achieved.

In order to measure performance, one must accurately answer these two questions:

- When is performance triggered?
- How is performance data recorded?

In the context of this lecture, measurement triggers can be either external or internal. External triggers can be timer interrupts or hardware counter overflows while internal triggers are typically automatic or manual code instrumentation. Regarding data collection, one can have *traces* which are sequences of events over execution time, and *profiles* which imply summations over execution time.

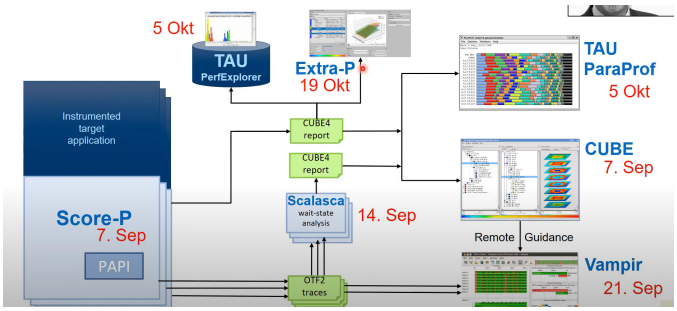


**FIGURE 1:** Score-P Ecosystem.

A benefit of tracing is that it is useful to reconstruct the dynamic behavior of the instrumented code. Given this fact, profiles can be calculated from this data. A critical downside is the size of these trace files, which can scale with the complexity of code and the number of instances or processes in which the code is executed on. Profiling on the other hand, is beneficial for longer running programs, yet due to its statistical nature variations in the dynamics of code get lost over time.

The conclusion at which experts in performance analysis arrive is that there is no single solution to performance analysis and a combination is needed. This is where the Score-P ecosystem comes into play. Score-P provides instrumentation for multi-process, thread-parallel and accelerator-based paradigms, with flexible measurement without recompilation of code. The Score-P ecosystem is built on top of Vampir, Scalasca, CUBE, TAU, Extra-P and TAUdb which trace and/or profile applications in their own unique way, see figure 1.

The lecture continues on to present some demos on how these tools are used. As a summary, performance profiling is a huge task which is composed of not only data acquisition, but of analysis, visualization and most importantly interpretation of the results of each of these phases. Whether using a highly scalable application in an HPC system or software in an embedded system, performance analysis can provide useful insights on how a particular application behaves and from this optimization can be implemented.