# TIMEMORY ECP TUTORIAL

April 26

WHEN: April 26, 2021 @ 12:00 pm – 3:00 pm Repeats ET

CONTACT: Jonathan Madsen
Email

**Software monitoring**

Have you ever written a multi-level logging abstraction for your project? Created an error checking system? Written a high-level timer + label abstraction? Have you then added additional abstractions for logging data values and/or recording the memory usage? Did you add or want to add support for exporting these labels to external profilers like VTune, Nsight, TAU, etc.? Do you need to support flushing this data intermittently? If your answer to any of these questions is yes, this is the right tutorial for you.

Logging, error-checking, high-level timekeeping abstractions are a staple in HPC applications. As projects grow in complexity and users, the developers often end up having to provide these abstractions because these capabilities are generally viewed as necessary for debugging, validation, and ensuring optimal performance. Timemory aims to simplify monitoring the state and performance of your application so that relevant debugging, logging, and performance data can be trivially enabled or disabled in a consistent and portable manner.

**Why timemory?**

Timemory is designed as a toolkit for implementing profiling, debugging, and logging solutions as well as providing a holistic profiling solution. If you would like to keep all your current abstractions and only want type-safe handles for invoking groups of them in bulk, timemory can provide that; if you would like to simplify aggregating the data from different MPI/UPC++ ranks, timemory can provide that; if you only want to add support for exporting to JSON/XML/etc., timemory can provide that; if you want to create a new command-line tool which combines different measurements, timemory can provide the components to easily do that; if you want a holistic solution that you can easily extend or restrict, timemory can provide that.

**What is timemory?**

Timemory is a multi-purpose C++ toolkit and suite of C/C++/Fortran/Python tools for performance analysis, optimization studies, logging, and debugging. The primary objective of timemory is to create a universal instrumentation framework which streamlines building software monitoring interfaces and tools by coupling the inversion of control programming principle with C++ template metaprogramming. The original intention of the toolkit design was specific to performance analysis, however, it was later realized that the design allowed debugging and logging abstractions to co-exist seamlessly with the performance analysis abstractions.
The design allows developers to construct production quality implementations which couple application-specific software monitoring requirements with third-party tools and libraries. In order to help ensure this objective is fully realized, timemory provides a number of pre-built implementations of a generic C/C++/Fortran library interface, compiler instrumentation, dynamic instrumentation, various popular frameworks such as MPI, OpenMP, NCCL, and Kokkos, Python bindings, and an extended analogue of the UNIX time command-line tool.

**Does HPC need another profiling tool?**

No. HPC has a surplus of performance analysis tools and APIs: VTune, NSight, TAU, Caliper, Score-P, Callgrind, LIKWID, Arm-MAP, CrayPAT, OpenSpeedShop, ittnotify, NVTX, PAPI, CUPTI, MPI-P, MPI-T, OMPT, gperftools, ROC-profiler, ROC-tracer, and innumerable application-specific abstractions which perform anything from basic timekeeping and memory usage to implementations and callbacks for the aforementioned APIs. We designed timemory as a way to easily integrate and maintain the exact set of measurements/tools/features you want to support with an interface best suited for your application.

**Contents of the Tutorial**

This is a preliminary outline of the tutorial. The tutorial is divided into two days. The first day will cover the front-end tools for C/C++/Fortran/CUDA/Python. The second day will cover how to use the C++ toolkit. The interactive tutorials will be held on Mondays: 9:00 AM – 12:00 PM PT (12:00 PM – 3:00 PM ET).

Day 1: Tools and Library (04/19/2021)

Introduction to timemory

- Motivation
- Design philosophy and nomenclature
- Installation

Command-line Tools

- timemory-avail — information tool
- timem — UNIX time + more
- timemory-run — dynamic instrumentation and binary re-writing
- timemory-plotter — matplotlib plotting of results
- timemory-roofline — generate the roofline

Library API

- Compiler instrumentation
- Extern C interface

Python API

- Decorators and context-managers
- Iterating over results in-situ

Python Command-Line Tools

- timemory-python-profiler — python function profiler
- timemory-python-trace — python line-by-line tracing
- timemory-line-profiler — classic line-profiler tool extended to collect different metrics

Visualizing and Analyzing Results

- Converting timemory data to pandas dataframes via Hatchet
- Manipulating dataframes
- Visualizing in Jupyter notebooks

Day 2: C++ and Python Toolkit (04/26/2021)

Python

- Using Individual Components to build your own tools

C++

- Creating a new component
- Using a custom component for timemory-run
- Designing a customized profiling API for your project
- Designing a customized debugging/logging interface for your project
  - Wrapping externally defined functions
  - Creating profiling/debugging libraries for your project
  - Insert measurements/logging/error-checking around C/C++ function calls
  - Auditing incoming arguments and return values
- Replacing externally defined functions
  - Experiment with mixed-precision without modifying original source code

**How to Attend**

- The lecture series is available to everyone.
- No-cost registration is necessary, meeting password will be sent to registrants.
- For the exercises, timemory can be installed locally or registrants may use a provided docker image.

**Presenters**

- Jonathan Madsen
- Laurie Stephey
- Muazz Gul Awan
- Rahulkumar Gayatri

Tutorial Material
Recording – Day 1
Recording – Day 2