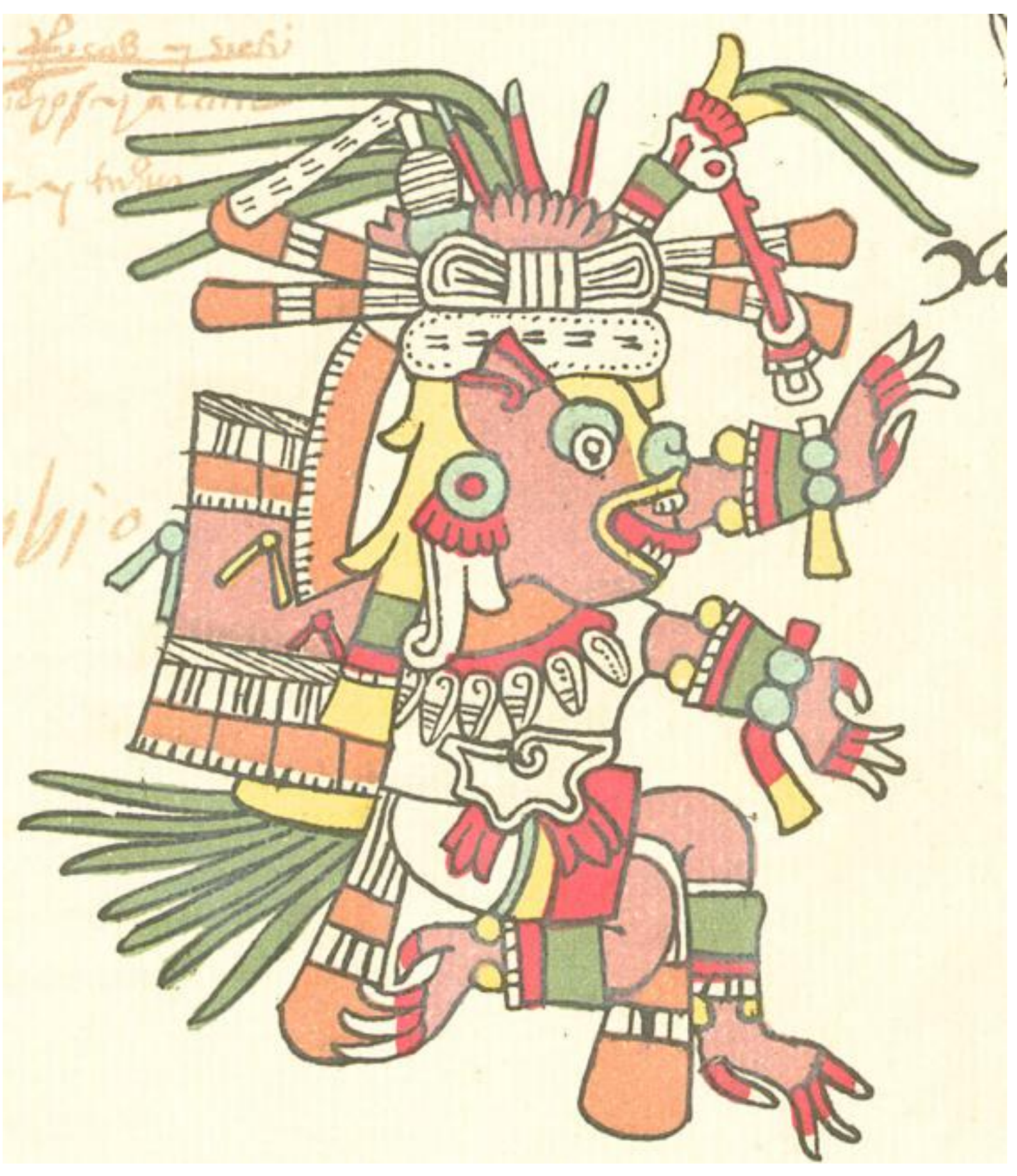
Data Visualization/Analysis and Performance Support In The XOLOTL Plasma Surface Interactions Model

Jay Jay Billings, Sophie Blondel, Crystal Jernigan, Jeremy S. Meredith, and Philip C. Roth
Oak Ridge National Laboratory
{billingsjj, blondels, jernigancl, jsmeredith, rothpc}@ornl.gov

Background and Motivation

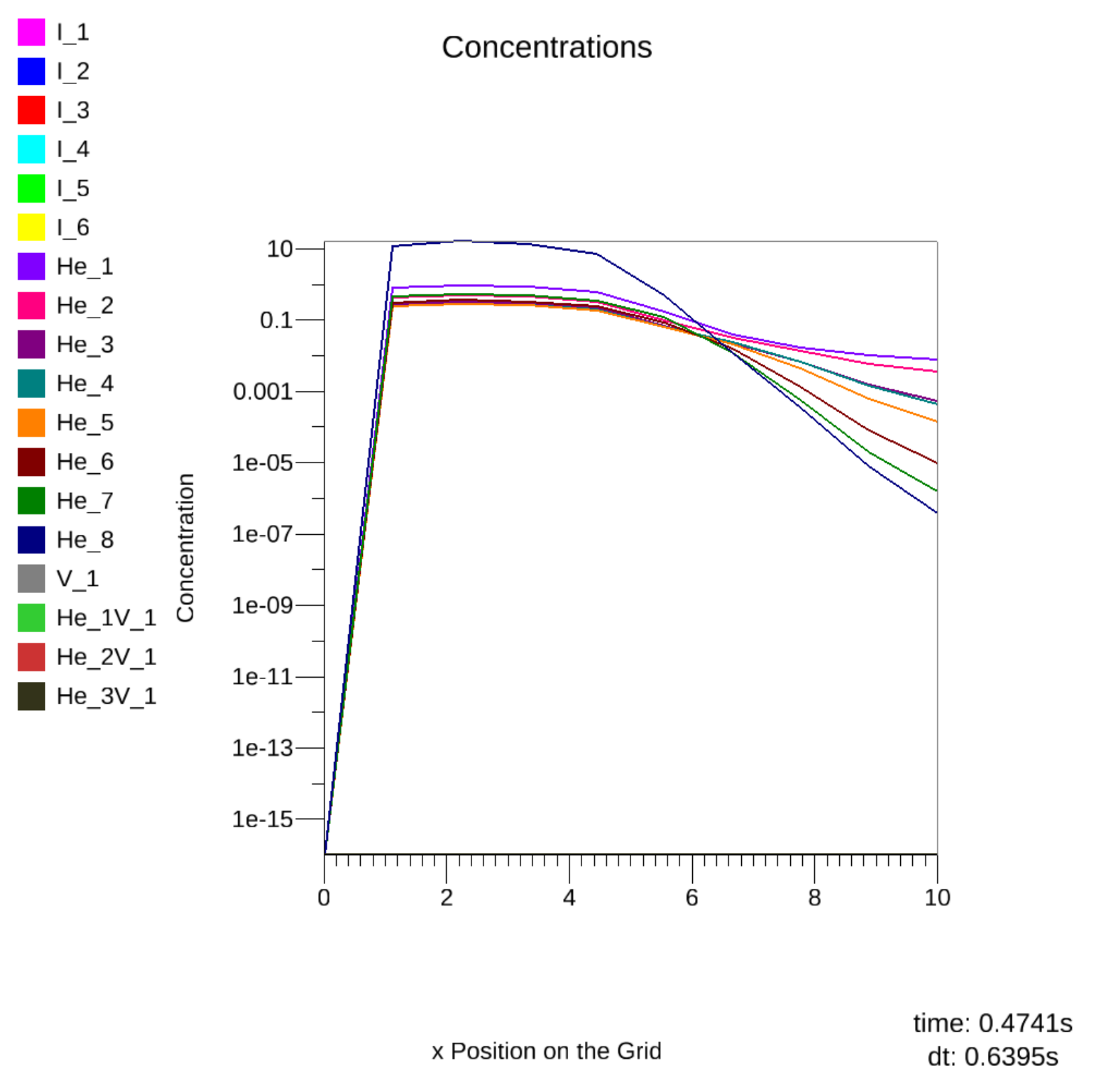
- XOLOTL model being developed as part of Plasma Surface Interactions SciDAC project
  - Intended for predictions of operating lifetime and performance of plasma facing components and bulk fusion materials
  - 3D, spatially discretized
  - Targeting range of computing systems from DOE Leadership Computing Facility (LCF) systems to traditional clusters to laptops and desktops
  - Targeting systems with traditional and heterogeneous architectures
- XOLOTL's rapidly changing code base creates challenges in understanding impact of code modifications on:
  - Quality of results
  - Performance and scalability
- Our response: build-in support for in-situ visualization/data analysis and lightweight performance data collection
  - Lightweight, "always on" data analysis and plotting for both performance and scientific data



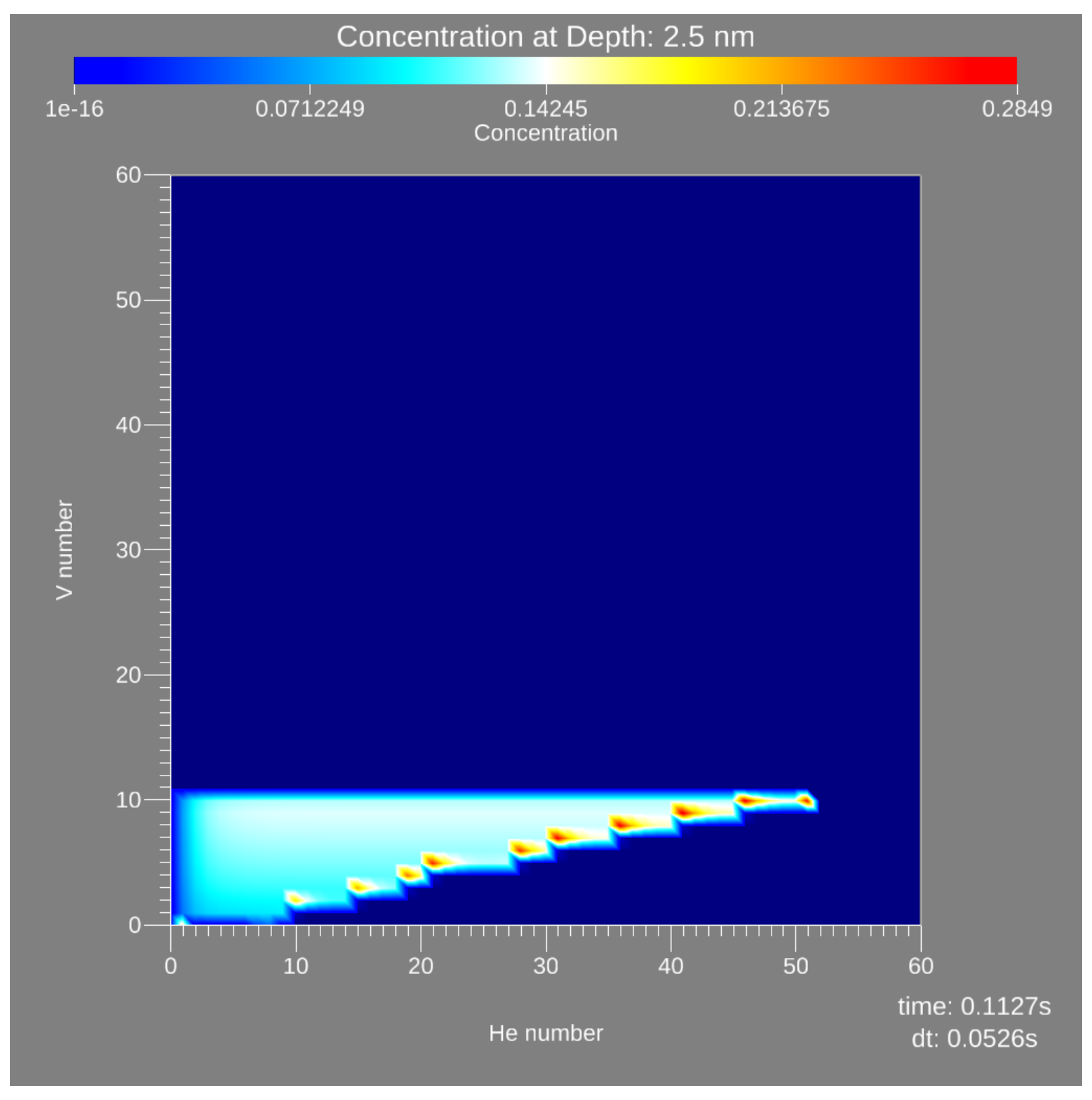
Data Visualization/Analysis

- C++ classes support collection and rendering of several types of scientific data:
  - Scatter plots show concentration along the grid
  - Series plots compare multiple species
  - Surface plots show range of cluster concentrations for each depth
- Uses Extreme-scale Analysis and Visualization Library (EAVL) for data analysis and rendering capabilities
  - lightweight, flexible, heterogeneous system support
- Controlled through command line switches
- Collaboration with **SDAV**, the SciDAC Institute for Scalable Data Management, Analysis, and Visualization

Example showing multiple species concentrations across grid points

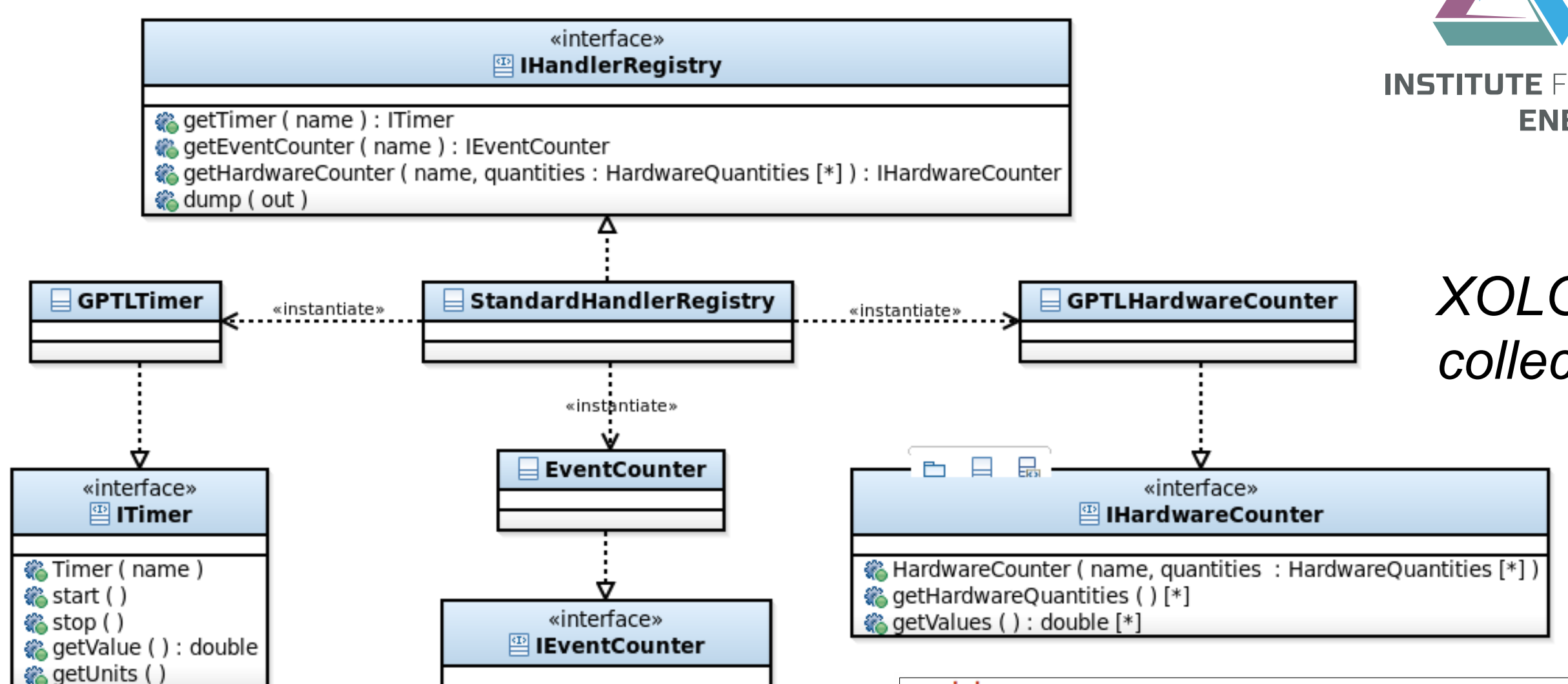


Example showing concentration for each cluster type at 2.5mm



Performance

- C++ classes support collection and analysis of several kinds of performance data:
  - Timers: measure wall-clock time required to execute a specific part of the code
  - Event counters: count the number of times an event of interest occurs
  - Hardware counters: provide access to processor hardware counters
- Uses General Purpose Timing Library (GPTL) with optional Performance Application Programming Interface (PAPI) integration
- Can be disabled with command line switch for lowest overhead
- Collaboration with **SUPER**, the SciDAC Institute for Sustained Performance, Energy and Resilience



«interface»
IHandlerRegistry
getTimer (name) : ITimer
getEventCounter (name) : IEventCounter
getHardwareCounter (name, quantities : HardwareQuantities [*]) : IHardwareCounter
dump (out)

GPTLTimer

«instantiate»

StandardHandlerRegistry

«instantiate»

GPTLHardwareCounter

«instantiate»

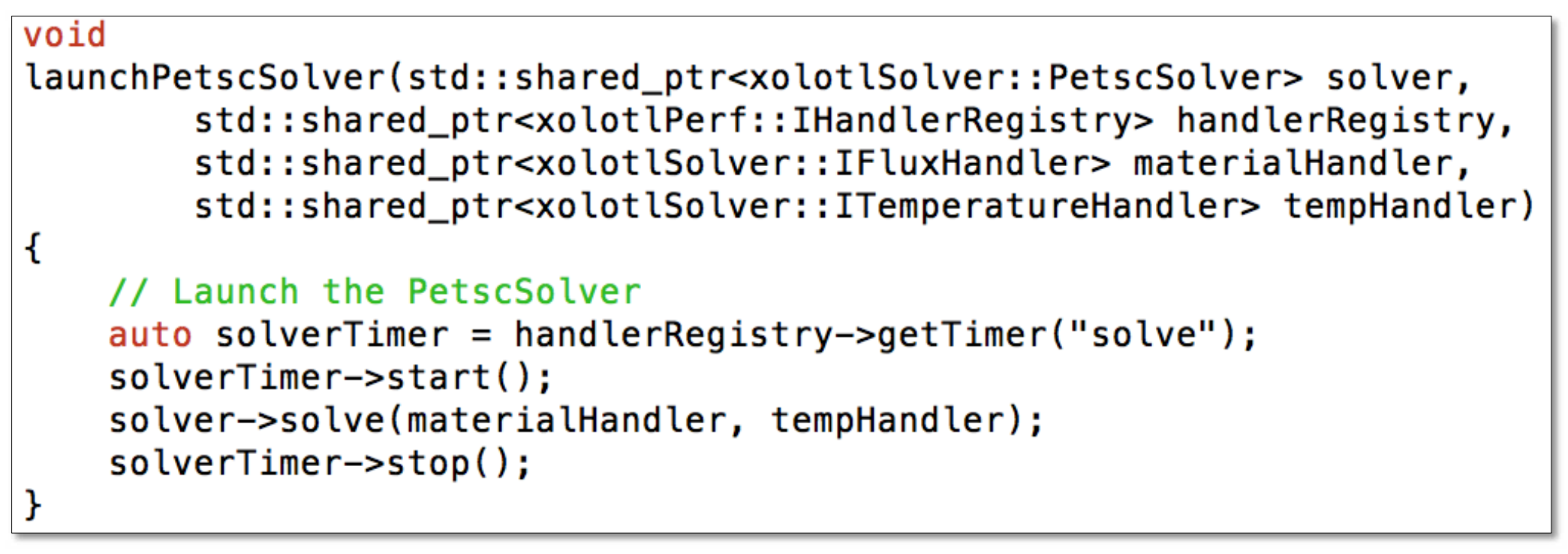
«interface»
ITimer
Timer (name)
start ( )
stop ( )
getValue ( ) : double
getUnits ( )

EventCounter

«interface»
IHardwareCounter
HardwareCounter (name, quantities : HardwareQuantities [*])
getHardwareQuantities ( ) [*]
getValues ( ) : double [*]

«interface»
IEventCounter
EventCounter (name)
getValue ( ) : unsigned long
increment ( )

XOLOTL performance data collection class model



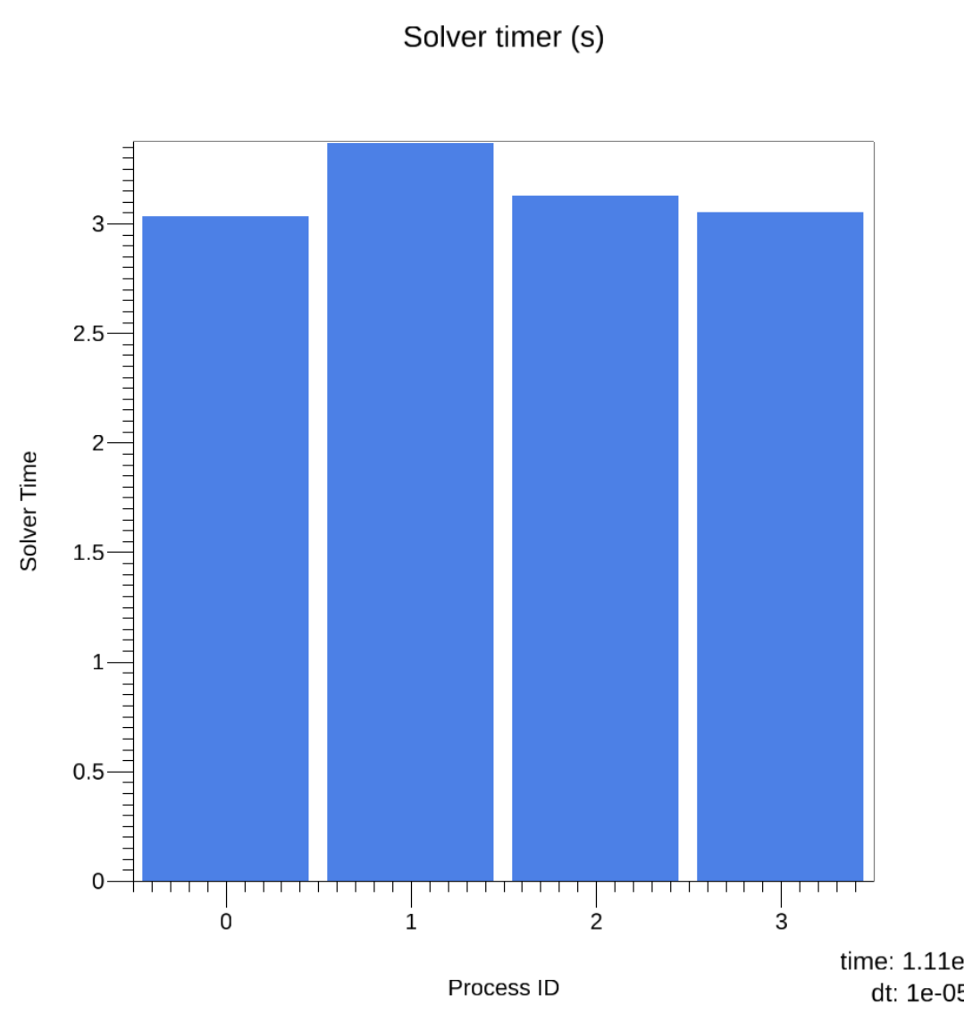
```
void
launchPetscSolver(std::shared_ptr<xolotlSolver::PetscSolver> solver,
        std::shared_ptr<xolotlPerf::IHandlerRegistry> handlerRegistry,
        std::shared_ptr<xolotlSolver::IFluxHandler> materialHandler,
        std::shared_ptr<xolotlSolver::ITemperatureHandler> tempHandler)
{
    // Launch the PetscSolver
    auto solverTimer = handlerRegistry->getTimer("solve");
    solverTimer->start();
    solver->solve(materialHandler, tempHandler);
    solverTimer->stop();
}
```

Example showing use of a Timer from XOLOTL's main.cpp

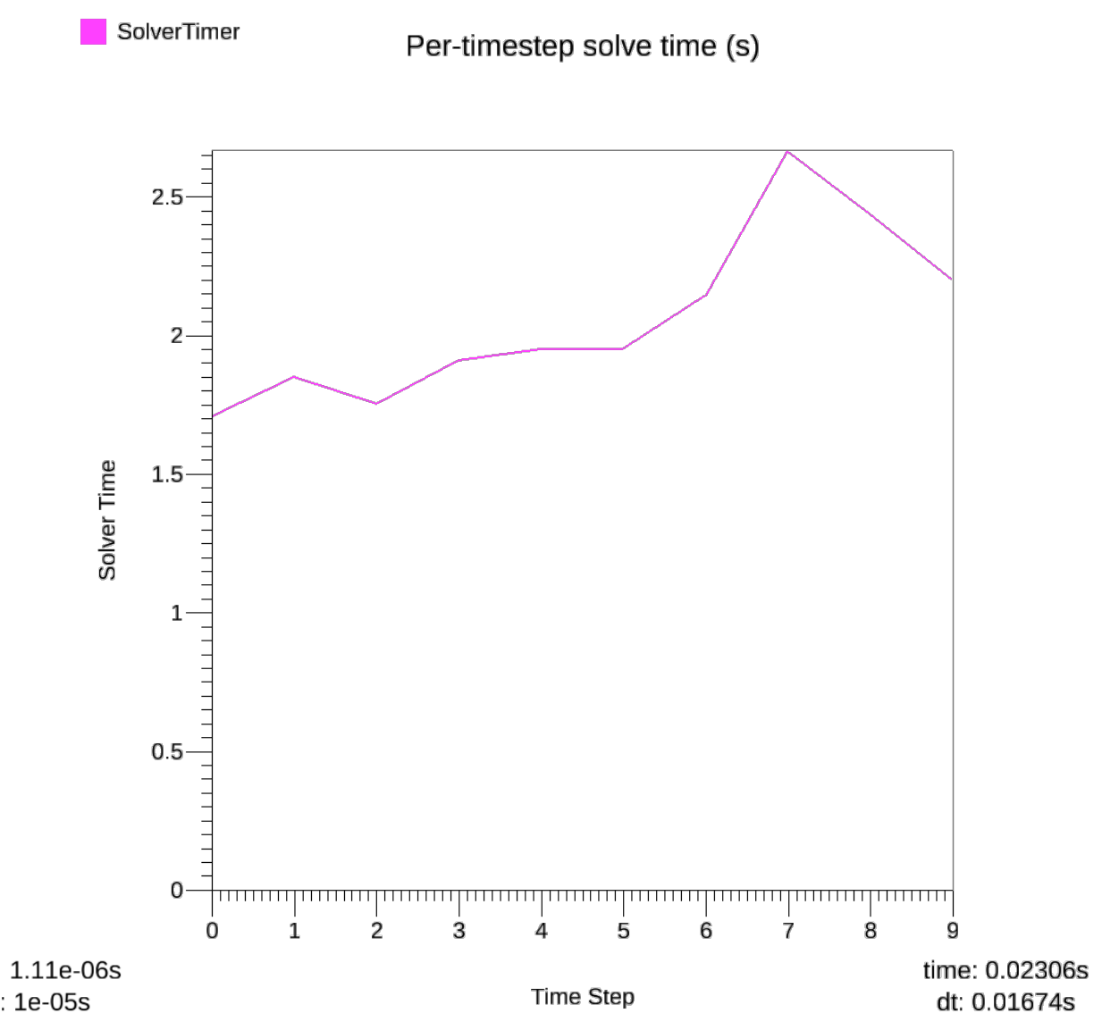
Integration

- Combining in situ performance analysis with scientific visualization and rendering capabilities
- Plot types showing key performance metrics over tasks and over time

Example showing cumulative time per MPI task



Example showing runtime for each timestep



Next Steps

- Data Visualization/Analysis
  - Additional in situ analysis, including temporal data
  - Detailed performance data visualization
- Performance
  - Deployment of a performance database to collect XOLOTL performance results over time
  - Determine regimes where XOLOTL is compute-bound and communication-bound

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.