Parallel
Computing
and I/O

Scimon - Scientific Monitor for Automated Run Logging and Reproducibility

- Author: Daniel Bremer
- Type: Master's Thesis
- Date: 2021-06-09
- Reviewers: Jun.-Prof. Dr. Michael Kuhn, Prof. Dr. Thomas Ludwig
- Supervisors: Jakob Lüttgau, Jun.-Prof. Dr. Michael Kuhn

In many scientific fields, the ability to simulate on computers and precompute experiments became a huge part of research. In classical experiments, researchers often utilized notebooks to document their experiments and actions, but with the introduction computers as an experimental environment the problem sizes quickly increased, often making it hard to document with classic means. With data for the environment of such experiments not being available, reproducibility, a cornerstone for reviewable science, is impacted and lowered. This thesis provides means to increase reproducibility for computational science experiments automatically capturing workflows, as well as the system's environment. Goals are not only on-the-fly logging of steps that lead t results but also generation of artifacts that can help to reproduce runs. The proposed Scimon software suite approaches reproducibility from multiple vectors. Logging information on workflows is enabled by logging specific commands entered in a user's shell or in jobs submitted to clusters. Besides the logged commands, also all files used and generated in the runs are stored, allowing re-execution with the same inputs. To ensure the correct version of the executed applications, the Git version control system is utilized to get the user's code state. Considering Artifact Descriptions required for conferences, such as the Supercomputing Conference, workflows and input files do not make a sufficient description, as it also requires information on the system's environment and libraries and other external code used with the applications. Scimon allows to automatically read such libraries from the applications in the building process and, being compatible with Spack, large parts of the system environment c be captured quickly. All this is done mostly invisible and unobtrusive to the user, allowing normal workflows which can be evaluated at a later point in time. Evaluation with multiple use cases will show the broad range of applications for Scimon. Besid the ability to correctly reproduce software that is in active development and could have runs in uncommitted states, a broad rang of artifacts is captured. The ability to not only automatically extract dependencies, but to capture the environment of a system because of a tight integration with Spack generates extensive documentation, enabling a high level of reproducibility.