W911Nf-12-1-0604, 1st interim report: Accelerating development and maintenance of scientific Python distributions

Overview

This report covers the period 26 September 2012 to 1 May 2013 of contract W911Nf-12-1-0604 between Simula Innovation AS and ERDC-IRO. The work done in this period completes the milestones described in the original project proposal (Milestones 1 and 2), and we describe the work on these below. Since then, the contract has been amended with additional tasks for a third milestone. This work has not yet started.

During the period, the project PI, Dag Sverre Seljebotn, worked 300 hours on the project. Including all overhead and expenses the calculated cost to Simula Innovation AS for this part of the project is $42,000.

Milestones 1 and 2 describes a software component for accelerating development of scientific Python distributions, dubbed the Build Artifact Cache in the proposal. This software component has since been given the name Hashdist, which we will use in the following. Hashdist is in continuous development and licensed under a permissive open source license (3-clause BSD). The code and issue tracker for Hashdist is available at https://github.com/hashdist/hashdist. In addition, as a proof of concept, we have worked on the Python-HPCMP Python distribution, authored by Chris Kees. A modified version of this software distribution using Hashdist technology is available at https://github.com/hashdist/python-hpcmp2. The work on this modified version of Python-HPCMP has been done together with Ondřej Čertík from the Texas Advanced Computing Center and Chris Kees (ERDC-CM).

Hashdist is a very low-level piece of technology, providing exactly the features we feel is missing from existing software packaging/meta-build systems. Focus has been on providing a solid core for Python-HPCMP to build further on, and in this we have succeeded. However, we would like Hashdist to be used by many other similar projects and build a community around it. A problem here is that Hashdist is only made available as a low-level library API, and the lack of higher-level abstractions to the features has turned out to be a significant barrier to wider adoption. The goal of Milestone 3 is to develop an API/domain specific language for describing software builds, which will attach an end-user-facing side to the project and so help bootstrap a community around Hashdist.

Milestone 1

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 2013**	2. REPORT TYPE **1st Interim Report**	3. DATES COVERED **26-09-2012 to 01-05-2013**

4. TITLE AND SUBTITLE **Accelerating development and maintenance of scientific Python distributions**	5a. CONTRACT NUMBER **W911NF-12-1-0604**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Simula Innovation AS,POSTBOKS 134, 1325 Lysaker,Norway,**	8. PERFORMING ORGANIZATION REPORT NUMBER **; 1535-EN-01**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **Engineer Research & Development Center - International Research Office, ERDC-IRO, Unit 4507, APO, AE, 09421**	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **1535-EN-01**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**3**	

The requirements of Milestone 1 were:

- Development of a specification format to enable fully describing a build process, and use it to give every build a unique ID (through hashing)
- Given the specification, perform the described build in an isolated environment, producing a *build artifact,* and maintain a *build artifact cache* to prevent gratuitous rebuilds
- A mechanism for declaring which software on the host system, not built by Hashdist, should be made available in the isolated build environment
- The ability to link together multiple build artifacts into a *profile*, as the entry-point of the user to a coherent software stack built by Hashdist

These objectives were completed within the first third of the project, and present in a public demonstration release of Hashdist in December 2012. Some the details have since changed as we got experience with applying Hashdist to Python-HPCMP, but most of the code and design remains from this milestone.

Of particular note:

- For specifying the source code of the build to use, we natively support archives (tar.gz, tar.bz2, zip) and the *git* version control system (VCS). Support for other VCSes like Subversion is not present but should be an easy extension later. To lay the foundation for future caching and mirroring of source code across an organization, the source code is identified by a commit ID or an archive hash, *not* URLs, in the build specification itself. Thus changing the download URL does not trigger a rebuild.
- In line with the proposed Milestone, the sandbox was at this stage very simple, only affecting the environment variables (such as clearing $PATH and $LD_LIBRARY_PATH and rebuilding them).
- The mechanism for using software from the host system has only been tested on small scale in isolated test cases, and not yet in Python-HPCMP, which currently assumes that all software from the host system should be made available (in the $PATH environment variable). Work on this is currently under way, outside of the scope of this project. However, we are confident that the mechanism Hashdist exposes for this is robust.

Milestone 2

The required goal of Milestone 2 was to build a subset of python-hpcmp and resolve any issues discovered. At the present date, all packages included in python-hpcmp builds (on the Linux platform) in the new Hashdist edition except "daetik" (python-hpcmp2 Issue 25). We stress that the goal of the contract of the present report was to lay the foundation of this work and provide assistance, and that the main work was carried out by Ondřej Čertík. Numerous Hashdist issues were reported and had to be fixed by us during this process.

The end-result is a much better controlled build process than what was provided by the previous version of Python-HPCMP, and we feel our original assumptions have been validated: The

Hashdist approach to performing isolated builds and caching the resulting build artifacts

Milestone 2 also specified a list of optional features, marked "If time allows", to be worked on while Python-HPCMP development was taking place. The status on these are:

- Improved sandbox functionality: Hashdist currently has an optional "jail" (Linux only) which able to capture filesystem access to "illegal" files made by the build process, both for logging these incidents for debugging purposes or to hide their presence. However, this has so far only been used in small-scale testing and not during Python-HPCMP development.
- Backwards compatibility layer for build scripts: Fully implemented.
- Statistics about build artifact cache, garbage collection: Not implemented. However the important structure for doing garbage collection (the build artifact dependency graph) is available, so that the work to be done is primarily figuring out how to expose garbage collection to the user.