Spack Buildcache Updates
HSF Packaging Group Meeting
Feb. 26, 2020

Patrick Gartung
Scientific Computing Division
Fermilab

What prompted this?

- Direct feedback on the Spack slack. I was identified as “the buildcache guy” and I had many discussions about:

- Spack buildcache mirror for E4S being set up at University of Oregon that is the results of CI build pipelines of common spack packages.

    - Browsable URL https://instinct.nic.uoregon.edu:8083/minio/e4s/build_cache/

    - Mirror URL https://instinct.nic.uoregon.edu:8083/e4s

    - Public key URL http://oaciss.uoregon.edu/e4s/e4s.pub

- Pipeline cache-only install for dependencies and then no-cache build of one package and adds it to mirror.

- Cache-only install calls binary_distribution:get_specs which downloads all spec.yaml files for all packages on the mirror.

- There are files for 5+ operating systems (OS) and architecture(arch) combinations and over 200 packages per each

- The process ends up spending more time downloading 1000’s of spec.yaml files than doing the unpacking relocation and build of the package. Caching helps after the first call to get_specs. Pipeline builds need done in a clean container because cache can become stale.

PR’s to mitigate this

- spack/spack#14467, spack/spack#14639: Only download spec.yaml’s for the current OS and arch in binary_distribution:get_specs()
    - two pull requests because I didn’t test the first one (oops)
- spack/spack#14659, spack/spack#14696: Further limit the number of downloaded specs using regex’s on links in index.html
    - two pull requests because I didn’t wait for a review
- spack/spack#14698, spack/spack#14714: Add method binary_distribution:get_spec() that takes the fully concretized spec from the install method and checks directly for the spec.yaml file instead of downloading all of the specs.
    - two pull requests because internal caching wasn’t handled correctly in the first PR.

Related and followup PR's

- spack/spack#14732: Restore the original behavior of buildcache list to allow constraints on the command line

- spack/spack#14929, spack/spack#15086: Replace direct call to patchelf call to function get_existing_elf_rpaths which handles exception codes from patchelf, e.g. fully statically linked elf binaries have no paths

- spack/spack#15003, spack/spack#15054: Calling spack's install_tree copies hard linked files. Use python tarfile function to create and extract from temporary file so hard links are handled properly, e.g. git package which makes 100+ hard links in libexec dir for git subcommands.

- All of these PR's made it into the v0.14 release of spack.

Pull requests in review

- spack/spack#13797: Install buildcaches into non-default directory layout. Need to support the case where buildcaches are created with the default directory layout but installing them into a spack config with a non-default directory layout. Came up in the context of spack dev where the layout is non-default.

```
install_path_scheme: “${ARCHITECTURE}/${COMPILERNAME}-${COMPILERVER}/${PACKAGE}-${VERSION}-${HASH}”

install_path_scheme: "${ARCHITECTURE}/${COMPILERNAME}-${COMPILERVER}/${PACKAGE}/${VERSION}-${HASH}"
```

- This had gone many reviews which resulted in three different implementations:
  - Load spec in new layout root, query the build environment for the new rpaths including the compiler. Use these to set the rpaths in the elf binary. This would work except some packages implement custom rpaths for finding libraries.
  - Search for the required libraries in the new rpaths and use the found paths to set the rpaths. This works for elf and mach-o binaries but there is not a one to one correspondence between the paths in the old install prefix and the new install prefix.
  - Map dependency prefixes in old directory layout to the corresponding package hash and save a dictionary of this in $prefix/.spack/binary_distribution. On installation use this dictionary and another dictionary mapping the dependency hashes to dependency prefixes in the new directory layout. This method works well and I can now relocate buildcaches created with relativized paths in the new directory layout. The initial implementation did not allow this.

Pull requests in review

- spack/spack#14719, spack/spack#15090: Address issue that buildcache create silently creates the build_cache directory in the current working directory when -d option is not used.

- spack/spack#15192: add install -o/--otherarch option to allow installing buildcaches from another OS on current OS, eg macOS on linux.

Conclusions

- Thanks to user feedback on buildcache usage I was able to implement a change that significantly improved the performance of finding and installing buildcaches

- Ongoing work in review and hopefully approved soon.