HSF Packaging Group: Introduction

Graeme Stewart and Ben Morgan

2018-05-16

Follow on from micro-architectures discussion

- Ben put together some very useful links discussing micro-architecture builds and fat binaries
  - https://github.com/HSF/packaging/blob/master/istools/README.md
  - Please comment and make PRs for adding and improving the material
- The HSF proposal for naming build platforms is ‘promoted’ to a consultation document
  - See http://hepsoftwarefoundation.org/technical_notes.html
  - This means that we’re asking people to really take a look now, with the intention to finalise this note soon
  - There are a few missing things
    - The micro-architecture description piece (e.g., x86_86+avx2)
    - Compiler release numbering has evolved since the document was written (e.g., for gcc only the major release number matters now)

Guix Report*

*pronounced ‘gix’

- CERN Computing Seminar on GNU Guix on May 3
  - Unfortunately there are no slides, but you can watch the recording of the presentation
- In a nutshell....
  - Guix is very similar in architecture to nix
    - Each package is built and identified by a hash of its source files, built options and dependencies
      - Recursive with dependencies, change libfoo and all clients of libfoo change their hash too
      - Aim for extremely high reproducibility
    - Nix style runtime environment, including versioning of each modification to a user’s environment
      - Multiple flavours of environment supported
    - All builds live in the same filesystem path, /guix, has to be writable
      - User builds supported - daemon takes care of doing this
      - Can use binary build caches

Guix

- Main users seem to be on HPCs
  - Big shared filesystem, all users “close”
- Selling points
  - Reproducibility
    - Of builds and runtime
  - Delegation of control to users
  - Composability (build on top of something that’s there, or any piece of it)
- Main technical difference with Nix is the language used for describing the builds
  - Guix uses scheme (or should (I say (scheme)))
  - cf., Nix that has it’s own functional DSL
    - Guix developers claim this as an advantage as it’s easier to write extensions and integrate with other scheme code, e.g., their dependency visualisation code

Interesting...?

- It's another point in packaging space, so in principle, yes
  - But one that lives very close to Nix
  - Given our limited resources probably not something to prioritize (unless someone has a burning desire)
  - Might be more interesting if we were to decide that (gu|n)ix was the right space to be in
    - Then the question of scheme vs. Nix DSL would be more relevant
    - IMO, neither of these languages are very good fits to our community's skill set
      - Python and shell match better what we do
- Observations
  - Smaller package base than Nix
  - No OS X distribution (build from source)

CHEP

- CHEP talk from the group will be followed by two talks on Spack

Beautiful, simple and remote ROOT graphics and GUI	*Serguei Linev*
Sofia, Bulgaria	11:30 - 11:45
Exploring server/web-client event display for CMS	*Alja Mrak Tadel*
Sofia, Bulgaria	11:45 - 12:00
Software Packaging and Deployment in HEP	*Benjamin Morgan*
Sofia, Bulgaria	12:00 - 12:15
Spack-Based Packaging and Development for HEP Experiments	*Dr Christopher Green*
Sofia, Bulgaria	12:15 - 12:30
IceCube CVMFS Software and Spack	*David Schultz*
Sofia, Bulgaria	12:30 - 12:45
dCache as open-source project showcase for education	*Mr Tigran Mkrtchyan*
Sofia, Bulgaria	12:45 - 13:00

Today’s Meeting

1. Introduction
2. Test drive round table

- As ever, further contributions and updates are very welcome - just ask Ben and me for a slot
- In two weeks time there is the CERN SFT LIM workshop, so we’ll skip that meeting
    - Meet again on 13 June