# Preliminary evaluation of Spack as a new backend for LCG releases

I. Razumov

LiM Meeting, 3 December 2019

## Introduction

### What is Spack?

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures, using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

# Introduction

## What is Spack?

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures, using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

## Why Spack?

- Is recommended by HSF Packaging group

[a]to build their software on top of existing LCG release
[b]by a small group inside ALICE

## Introduction

### What is Spack?

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures, using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

### Why Spack?

- Is recommended by HSF Packaging group
- Experiments (ATLAS and LHCb) has expressed interest in evaluating Spack during CHEP'19 conference held in Adelaide, South Australia, over Monday-Friday 4-8 November 2019.

[a]to build their software on top of existing LCG release
[b]by a small group inside ALICE

## Introduction

### What is Spack?

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures, using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

### Why Spack?

- Is recommended by HSF Packaging group
- Experiments (ATLAS and LHCb) has expressed interest in evaluating Spack during CHEP'19 conference held in Adelaide, South Australia, over Monday-Friday 4-8 November 2019.
- Already used by FCC[a], ALICE[b]

[a] to build their software on top of existing LCG release

[b] by a small group inside ALICE

## Evaluation

The Good:

- Spack has an easier syntax (Python), has recipe templates (e.g. for autoconf or python packages)
- A lot of “hacks” we have or want in lcgcmake (e.g. setting a fixed version of dependency, using git commit id in place of version, “layered” releases[1]) are already implemented in Spack
- Recipes for many packages are already available in Spack: of 387 “external” packages (i.e. everything but generators), Spack has recipes for about 279.
  - We are missing a bunch of Python packages (machine learning, Jupyter), all GRID packages, a few ‘go‘ modules
  - Migrating the recipes for them is not expected to be a difficult task, just time consuming

[1]a.k.a. chained installations

## Evaluation

The Not-so-Good:

- Spack can only build one package at a time (a fix for that is expected to arrive next February)

## Evaluation

The Not-so-Good:

- Spack can only build one package at a time (a fix for that is expected to arrive next February)
- In some cases (e.g. openblas, ROOT) the recipes differ significantly between spack and lcgcmake
  - We can, for now, work around this by overwriting the recipes for such packages with our own (spack allows that)
  - Will try and push the changes upstream.

## Evaluation

The Not-so-Good:

- Spack can only build one package at a time (a fix for that is expected to arrive next February)
- In some cases (e.g. openblas, ROOT) the recipes differ significantly between spack and lcgcmake
- The concretizer (a part of Spack that resolves versions and dependencies of packages) has a few problems :
    - In some cases (e.g., python packages like matplotlib) one needs to explicitly specify versions of dependencies, otherwise the concretizer will pick a default one, and will complain about incompatibility
    - "Virtual" packages (i.e. packages that have more than one implementation: blas, lapack, java) are often not resolved correctly — even when you explicitly set what package provides what in packages.yaml.
        - E.g., it tends to use just about anything but Netlib's implementation of LAPACK.
        - Had to edit recipes to fix that.

## Evaluation

The Not-so-Good:

- Spack can only build one package at a time (a fix for that is expected to arrive next February)
- In some cases (e.g. openblas, ROOT) the recipes differ significantly between spack and lcgcmake
- The concretizer (a part of Spack that resolves versions and dependencies of packages) has a few problems
- Like lcgcmake, Spack uses hashes to uniquely identify a “concretized” package. However, the hashing in Spack is a bit too aggressive: even adding a new version of package will change the hash. No ETA for a fix.

## Status and Plans

- As a first step, we will try to use Spack to build an equivalent of LCG_96b for one platform (x86_64-centos7-gcc8-opt) for the experiments to test.
- It will contain only the packages for which recipes are already present in Spack
- If the test is successful, we will start implementing missing recipes