Large Scale Software Building with CMake in ATLAS

J. Elmsheuser, A. Krasznahorkay, E. Obreshkov, A. Undrus
for the ATLAS Collaboration
Brookhaven National Laboratory, USA; CERN, Switzerland; University of Texas, Arlington, USA

ATLAS Software Projects

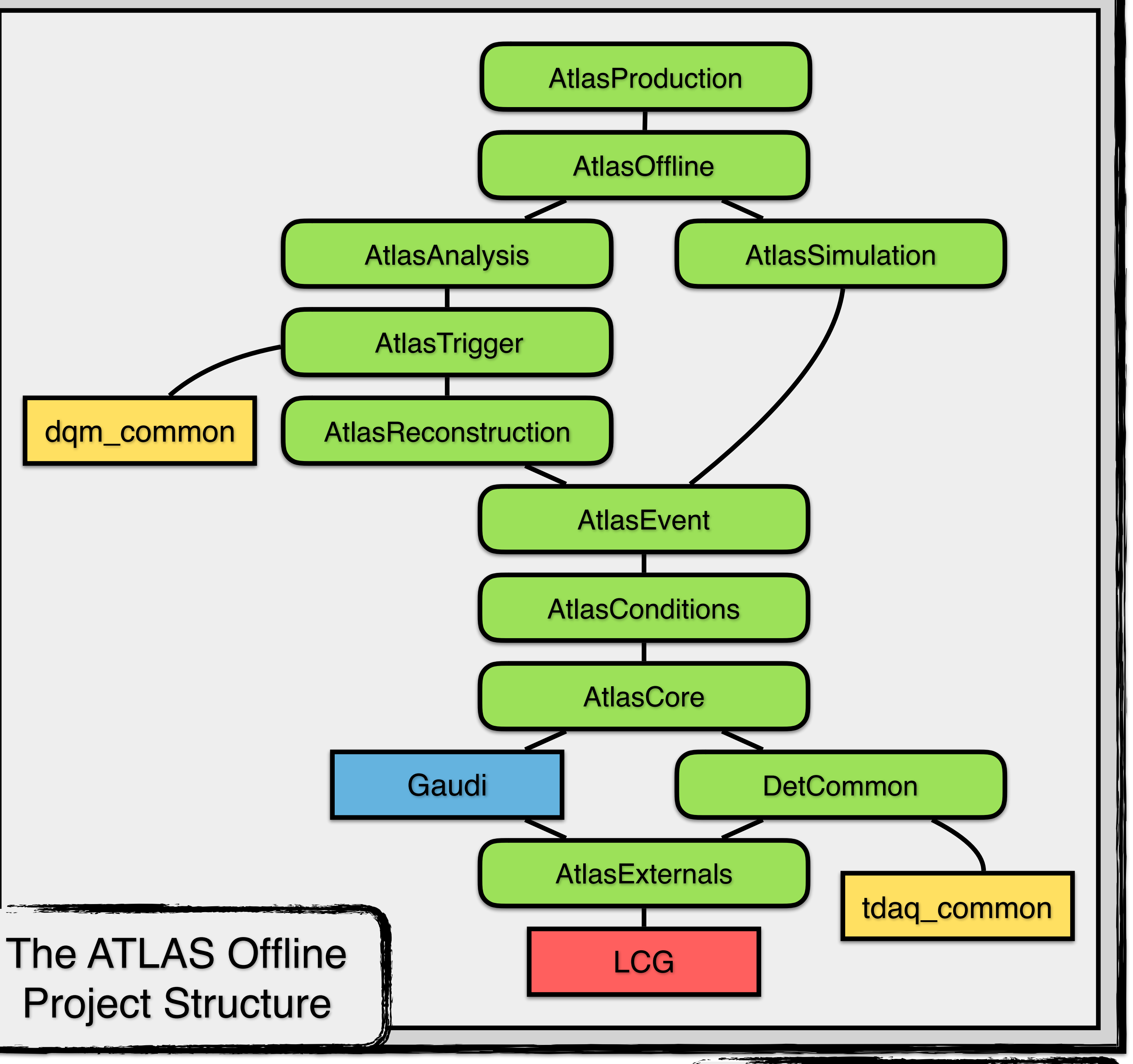
ATLAS organises its software into "projects", built from "packages". The smallest (analysis) project is composed of ~140 packages, while the full offline project is ~2200 packages strong. Some packages (like Event Data Model definitions) are used in multiple projects at the same time.

The build system needs to be able to:

- Discover the dependencies between packages, building them in the correct order
- Allow a package to be moved from one project to another one
- Allow a patch release to be built on top of an existing release, containing updated versions of just a few packages
- Be robust against build errors. Even with code errors present, the build and RPM creation needs to progress as far as it possibly can.
- Allow the users to test changes to packages on top of a software release/nightly

ATLAS switched to using CMake (https://cmake.org) as its offline software build tool in 2016.

- Replacing CMT (http://www.cmtsite.net)
- Converting the CMT configuration of >2200 packages into CMake configurations, using a custom Python script translating the configuration



The ATLAS Offline Project Structure

Package Configuration

Packages in ATLAS can have multiple roles:

- Hold the source for libraries and executables, building those libraries and executables
  - Most packages depend on source code from other packages for their build
- Hold scripts and other files necessary for the runtime environment, installing these into the right place during the build

Each one comes with a single CMakeLists.txt file, describing how that one package should behave.

Packages need to depend on each other, while not knowing where a dependent package is coming from.

- The upstream package could either be in a base project, or in the current one

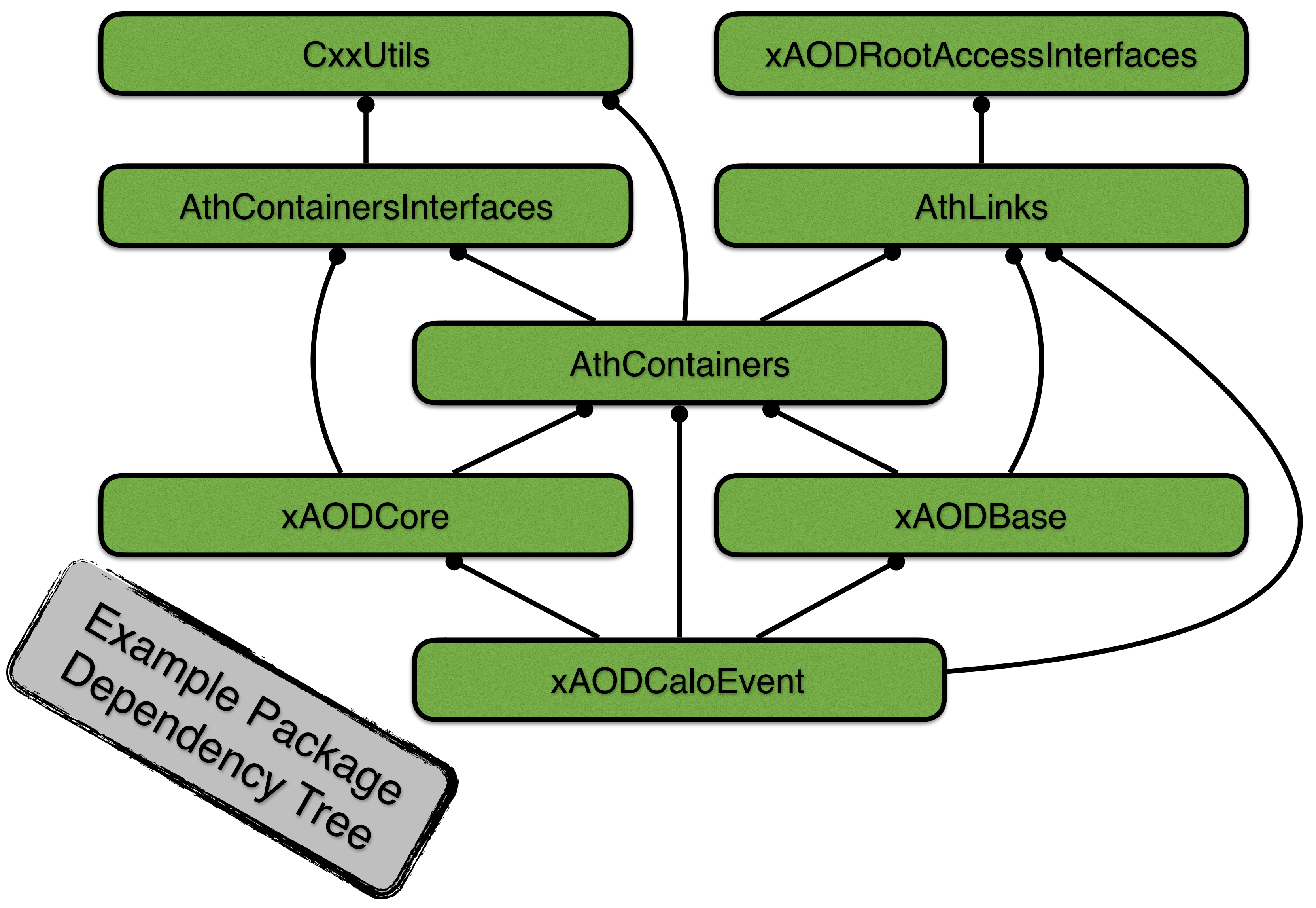
Example Package Configuration

```
atlas_subdir( HelperLibrary )
atlas_depends_on_subdirs(
   PUBLIC
   Control/AthenaKernel
   PRIVATE
   Control/CxxUtils )

find_package( ROOT COMPONENTS Core Hist )
find_package( Boost COMPONENTS regex )

atlas_add_library( HelperLibrary HelperLibrary/*.h src/*.cxx
   PUBLIC_HEADERS HelperLibrary
   INCLUDE_DIRS ${ROOT_INCLUDE_DIRS}
   PRIVATE_INCLUDE_DIRS ${Boost_INCLUDE_DIRS}
   LINK_LIBRARIES ${ROOT_LIBRARIES} AthenaKernel
   PRIVATE_LINK_LIBRARIES ${Boost_LIBRARIES} CxxUtils )
atlas_add_executable( HelperApp util/HelperApp.cxx
   LINK_LIBRARIES HelperLibrary )
atlas_add_test( HelperLibrary_test test/HelperLibrary_test.cxx
   LINK_LIBRARIES HelperLibrary )

atlas_install_python_modules( python/*.py )
```



Example Package Dependency Tree

Project Configuration

Projects need to describe:

- What external project(s) (LCG release, T/DAQ release), if any, they depend on
- What ATLAS project(s) they depend on
- How to set up the runtime environment for the project
- How to build a (RPM) package out of the project

Described by the CMakeLists.txt file at the root of the project's source tree.

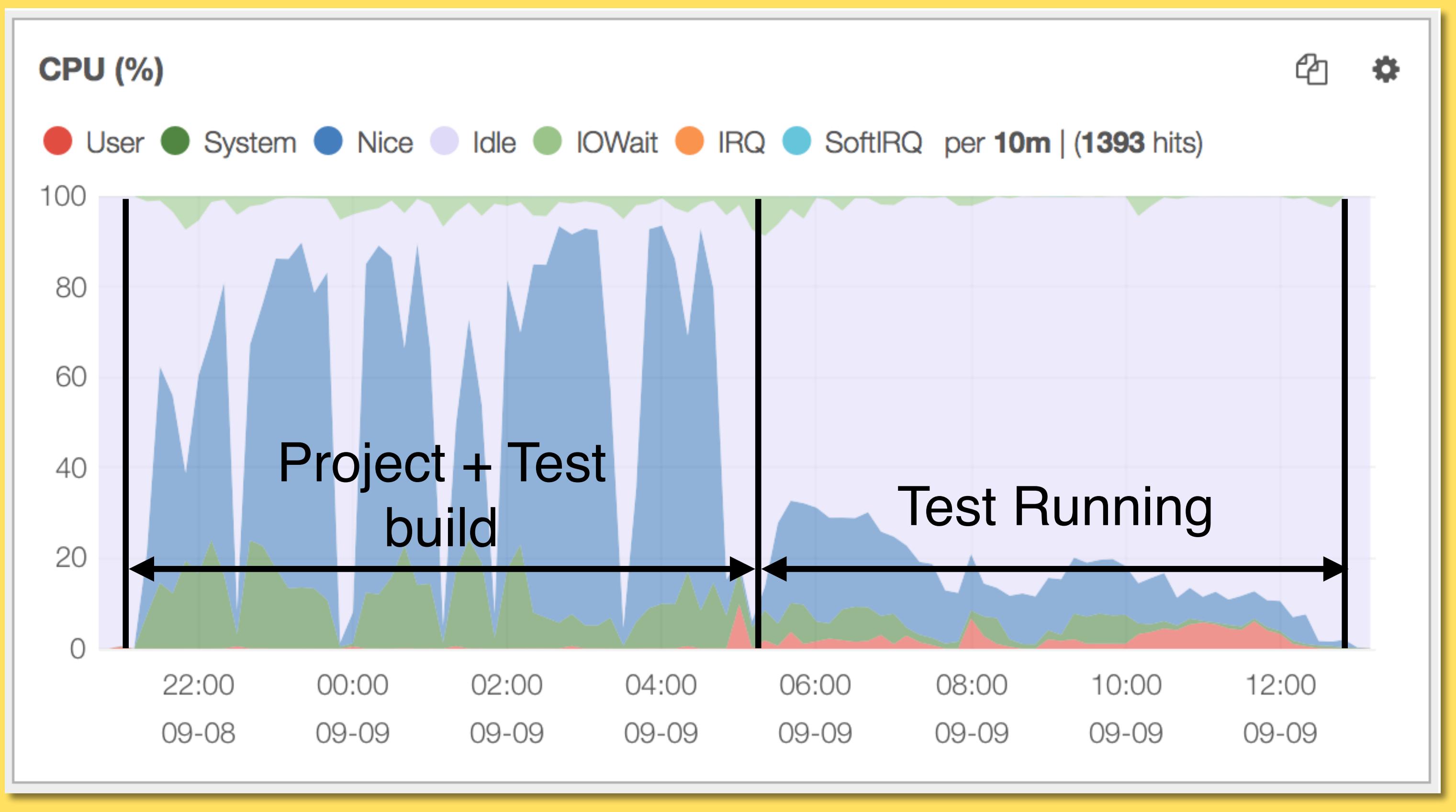
Simplified AtlasReconstruction Configuration

```
find_package( AtlasEvent )

atlas_ctest_setup()

atlas_project( AtlasReconstruction 21.0.5
   USE AtlasEvent 21.0.5 )

lcg_generate_env( SH_FILE ${CMAKE_BINARY_DIR}/${ATLAS_PLATFORM}/env_setup.sh )
atlas_cpack_setup()
```



Performance

Switching to CMake **reduced the build time over CMT by ~40%**. RPM packages are created as part of the nightly build. Delivering a software release is just copying the RPMs from a nightly server to the release YUM (http://yum.baseurl.org) server.

- Installing a full offline release on CVMFS from RPM using a modified version of YUM (https://gitlab.cern.ch/rhauser/ayum) takes ~1 hour.

There are things still to do:

- Reduce the number of projects. To avoid the installation downtime between project builds.
- Parallelise installation and RPM building
- Only build unit test code/executables after the general build step.
- Some further improvements for using Continuous Integration in the builds