CMake: Improving The Build

Attila Krasznahorkay

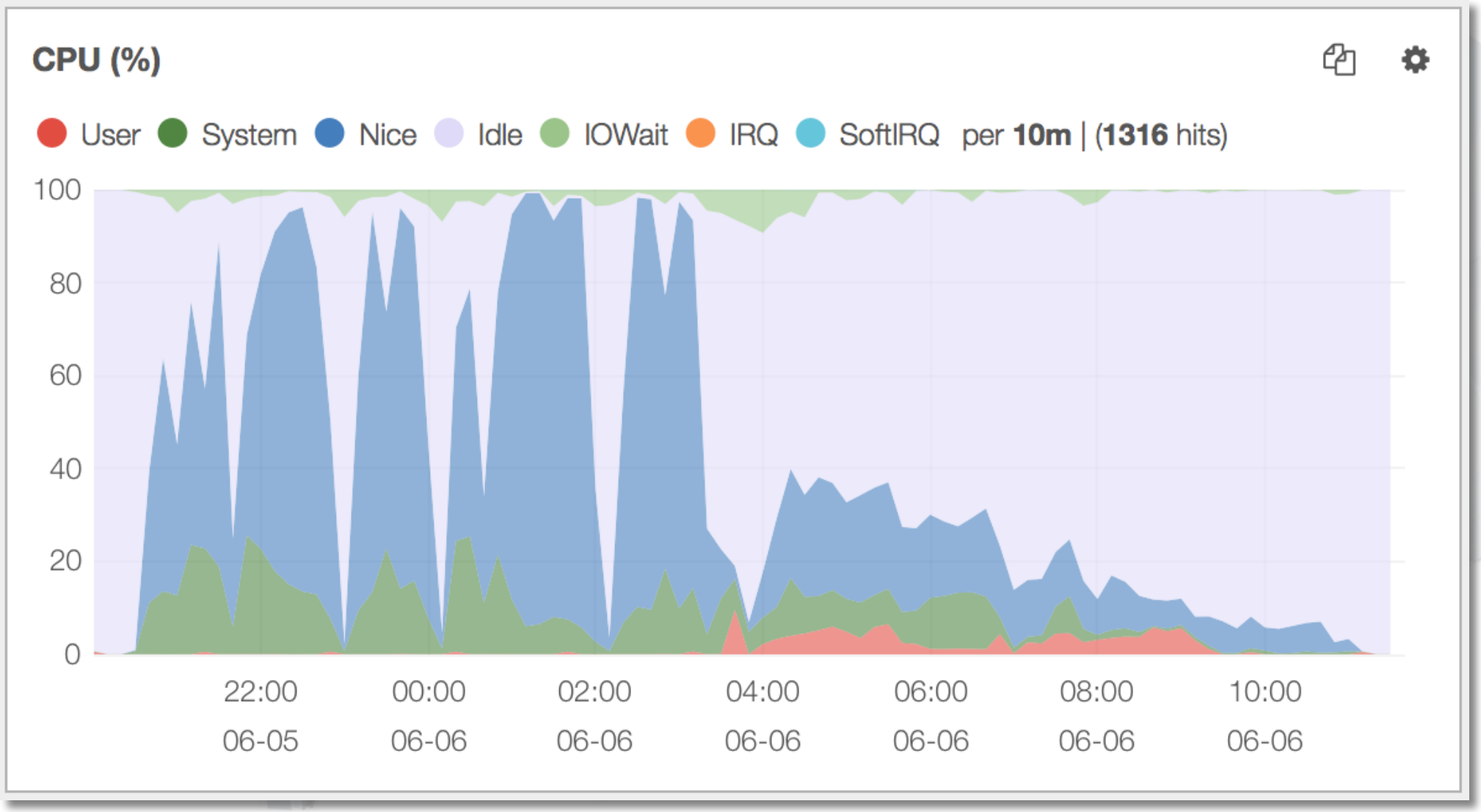
CERN

ATLAS EXPERIMENT

Build Steps

- Building a given project takes many steps
  - All taken care of by NICOS of course
  - Much of this was originally designed on top of CVS+CMT
- Checkout
  - Currently checking out 20-500 packages from SVN
  - Should simplify a lot with Git. But in the end, not a major bottleneck.
- Configuration
  - Setting up the environment done with asetup, in O(seconds)
  - Running CMake scales not the best with the number of packages (numbers shown later a bit unreliable as well)
    - Still, probably can't do miracles at this point
- Build
  - Other than using something other than GNU Make, can't change much here
  - Currently the build system doesn't make a big difference. Could change drastically when introducing continuous integration.
- Installation, RPM building
  - Takes care of moving O(10k), O(1GB) files
  - Can take significant O(10min) time
- Testing
  - Building the test code currently done as part of the normal build. Not insignificant by now.
  - Running done at the end of the build. After installing the code on AFS.
    - Personally I'm happy with this setup.

Build Time (devval,rel_1)



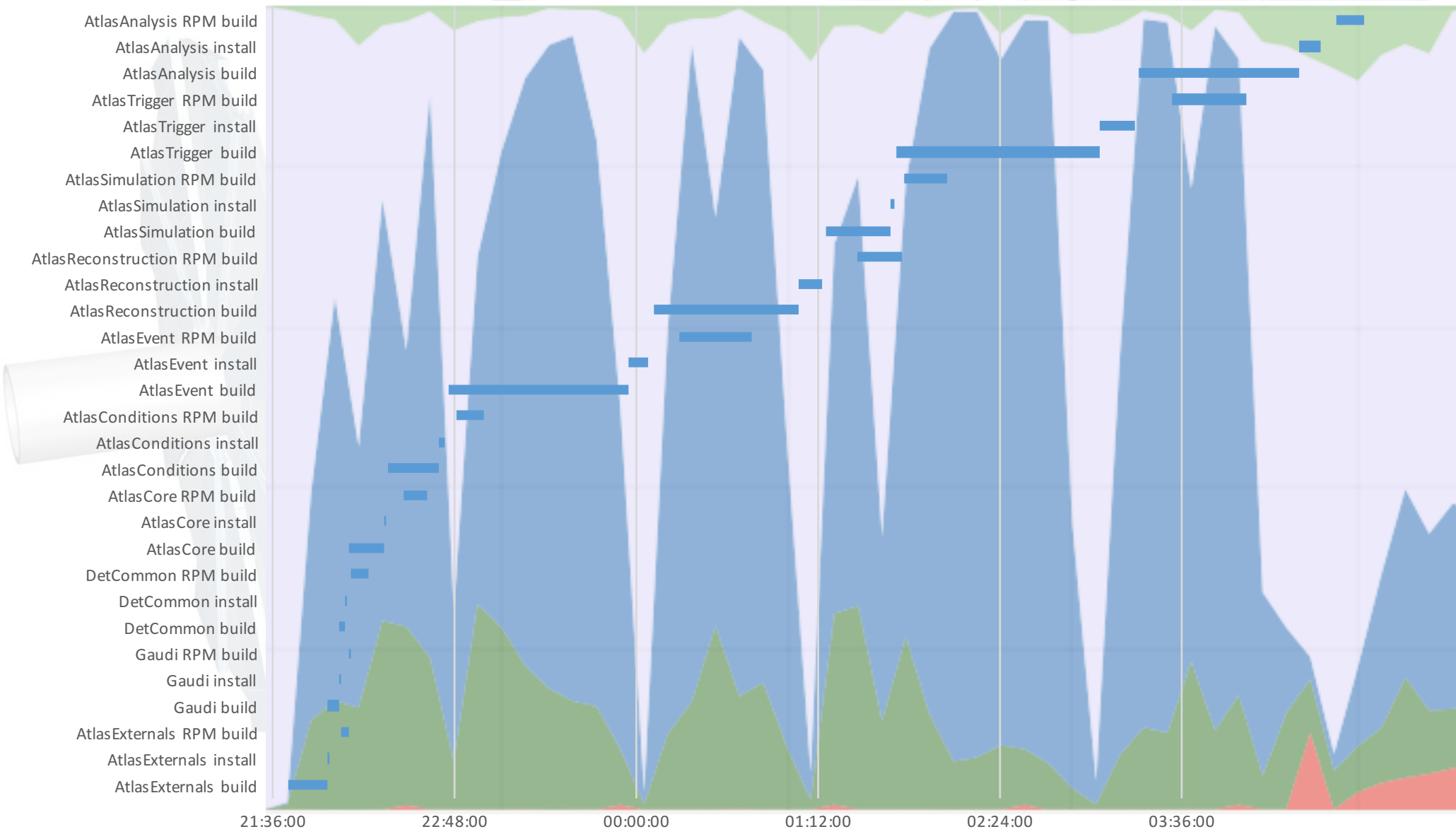
Build

Testing

Build Time (devval,rel_1)

	Configuration	Build	Installation	RPM Build
AtlasExternal	7s	15m 34s	4s	2m 42s
Gaudi	18s	4m 26s	1s	11s
DetCommon	34s	2m 29s	39s	7m 15s
AtlasCore	1m 25s	13m 57s	46s	2m 42s
AtlasConditions	31s	20m 39s	1m 51s	11m 14s
AtlasEvent	2m 15s	1h 11m 20s	8m 12s	28m 6s
AtlasReconstruction	5m 18s	56m 51s	9m 30s	17m 46s
AtlasSimulation	4m 20s	25m 41s	1m 40s	17m 18s
AtlasTrigger	3m 28s	1h 20m 30s	13m 55s	29m 15s
AtlasAnalysis	4m 11s	1h 3m 28s	8m 45s	17m 38s
AtlasOffline	59s	4m 52s	46s	2m 41s
AtlasHLT	49s	3m 21s	14s	25s
Total	24m 15s	6h 3m 8s	46m 23s	2h 17m 13s

Build Time (devval,rel_1)



Merging Projects

- Should help reduce the downtime in the builds
- Which projects should stay separate?
  - AtlasExternals must be a separate project for technical reasons
  - DetCommon (probably) needs to stay separate
    - Although I don't know exactly yet how this project is going to be used by the online software with CMake…
  - Any good reason to keep AtlasSimulation separate?
  - Could we merge AtlasCore, AtlasConditions, AtlasEvent, AtlasReconstruction, AtlasTrigger, AtlasAnalysis (plus possibly AtlasSimulation and AtlasOffline)?
- Would help a lot with finding CMake configuration problems early
  - Currently if a CMake error is collected into a package in AtlasTrigger, that will only bomb the build in the middle of the night
    - No chance for the release shifter to catch it early, take the tag out, and restart the build
  - Building everything up until AtlasCore takes ~30 minutes. A configuration problem in this gargantuan project would be discovered within ~45 minutes of starting the build

Testing

- A lot of unit testing code is built as part of the normal build at the moment
  - When building the code "by hand" during development, this is not a bad thing
    - Some of the unit tests can uncover problems already at build time
  - I'm not in favour of actively running unit tests as part of the normal build, even in "development mode" though
    - Much more prefer to require the user to run "`make test`" after running "`make`"
- Propose to change the nightly build procedure a bit
  - Don't build the unit test executables during the normal build. Imagine it to be done with something like:

```
cmake -DNO_AUTO_TEST_BUILD=TRUE …
make
```

  - Then run the ATN tests, after the normal build has finished, like:

```
make atlas_tests
ctest --label-regex "^AthContainers$" –output-on-failure
…
```

Externals (1)

- Currently (almost) all the `Find<Foo>.cmake` modules are in either `AtlasCMake` or `AtlasLCG`
  - It made sense for the initial development, but will become very hard to manage very soon
- Tried something slightly different with `External/AtlasGoogleTest`
  - The package builds GTest/GMock with its `CMakeLists.txt` file
  - It also provides a `FindGMock.cmake` file (`FindGTest.cmake` is picked up from CMake itself)
- Should we outsource all the `Find<Foo>.cmake` files into the "old" glue packages?

Externals (2)

- Currently use externals with:
  - Picking them up from LCG
  - Picking them (TDAQ) up from custom locations on AFS, and from custom RPMs
  - Building them as part of AtlasExternals
- Moving between the first and third is relatively easy
  - Currently we build CLHEP 2.2.0.4 inside AtlasExternals.
    - Only had to collect `External/AtlasCLHEP` into AtlasExternals for this. (And all the other externals using CLHEP.)
- This was discussed for a long time by now: Should we just ditch LCG completely, and build everything ourselves?
  - Currently AtlasExternals builds 18 packages
  - We pick up ~80 packages from LCG
- At this point more of a political question than a technical one…

Summary

- Currently very far from the 3 hour build time target
  - Cutting the build of the unit tests should help somewhat
  - Delaying RPM builds to the end could also help
  - Merging projects could be good
  - But probably not going to reach 3 hours with all of this. We'll have to modify the code itself to reach that goal.
- Adding more externals to the build will just make this worse
- Incremental builds should help a **lot**
  - But will probably require substantial development