Journal of Physics: Conference Series **219** (2010) 042023 doi:10.1088/1742-6596/219/4/042023

The ATLAS RunTimeTester Software

Brinick Simmons[1], **Peter Sherwood**[1], **Krzysztof Ciba**[2], **Alex Richards**[1]

[1] University College London, Gower Street, London WC1E 6BT, UK

[2] AGH-University of Science, (FPACS AGH-UST) al. Mickiewicza 30, PL - 30059 Krakow, Poland

E-mail: rtt@hep.ucl.ac.uk

Abstract. The ATLAS experiment's RunTimeTester (RTT) is a software testing framework into which software package developers can plug their tests, have them run automatically, and obtain feedback via email and the web. The RTT processes the ATLAS nightly build releases, using acron to launch runs on a dedicated cluster at CERN, and submitting user jobs to dedicated LSF batch queues. Running computationally longer tests, up to 24 hours long, it is thus complementary to the ATLAS ATN framework which feeds back rapidly on few event tests run directly on ATLAS build machines.

1. Introduction

The offline software code base for the ATLAS experiment at CERN runs into millions of lines across ∼2000 packages[1]. ATLAS makes nightly builds of its software using the NICOS[2] system, and employs various testing frameworks to test those builds, each framework fulfilling a particular testing need.

The Kit Validation (KV) framework validates ATLAS code distributions at non-CERN sites. The AtNight (ATN)[3] framework, as part of NICOS, provides several hundred, short (∼5min) smoke-tests directly on the build nodes. The RTT[4] system, that we will discuss in this paper, fulfills a major need within ATLAS for computationally longer tests, up to 24 hours, allowing access to rarer bugs. It recently integrated two other ATLAS testing frameworks: the Full Chain Tests (FCT) and Tier-0 Chain Tests (TCT) frameworks, both of which test complete routes through the Generation to Reconstruction and Analysis software chain. Integration of the three has enabled ATLAS to pool hardware resources and harmonise presentation of test results. Furthermore, RTT clients are only obligated to familiarise themselves with one particular system in order to access what was, previously, three different frameworks.

2. The RTT

The RTT is coded in the Python programming language, and amounts today to ∼30000 lines of code. It is a major part of the ATLAS software validation effort, running daily in production at CERN on a dedicated batch cluster.

The RTT provides ATLAS developers with a common automation tool for the setting up and running of test jobs, and the presentation of results. It provides two modes of running: "Linux Interactive" mode and "Automated, Batch" mode, each with a different intended use-case. In the first instance, the developer downloads the RTT and runs it interactively on their local Linux node. Typically it is used in this mode to execute a few, short RTT jobs on the developer's

locally, checked-out package in order, for example, to verify the package before committing or tagging in the ATLAS software repository.

In the second mode, the developer has put their package into an ATLAS nightly build, and the RTT, firing automatically via the acron facility, finds the release, finds the package and runs the defined RTT jobs in batch. This mode is employed many times daily at CERN, typically involving hundreds of RTT jobs across tens of developer packages. It is used to validate entire components of the ATLAS software, such as reconstruction and simulation.

It is worth emphasising that whilst ATLAS developers define their RTT tests within their package, that which is being tested is more than the code within the package. These are integration tests, not unit tests, and are designed to evaluate the correct performance of a whole chain of ATLAS software leading up to and including the developer package. The RTT does not prohibit developers from performing unit tests, however it is not its primary use case.

2.1. RTT jobs

A RTT job is a three step phase consisting of a pre-job, a job and a post-job[1]. Typically the job itself involves running some aspect of the ATLAS Athena software; the pre-job, as a minimum, involves creating and setting up the job run directory with required files for the job, and the post-job phase is a sequential execution of actions and/or tests such as making plots, performing regression tests, etc. Sequence order is determined by the RTT, but can be overridden by the developer. Finally, "keep" files are transferred from the job run directory to a web-served area for browseability.

2.2. Developer-RTT interaction

2.2.1. Unified Test Configuration File The developer's interface to the RTT is a single XML file known as the "Unified Test Configuration" package XML file. Within this file, which lives inside the developer's package, the developer describes, in pre-defined syntax, the RTT jobs that the RTT should run. It is known as a "unified" file because it is the means by which developers can talk to any of the three (ATN, KV, RTT) test frameworks. The basic three-part structure for a test XML file is:

```
<unifiedTestConfiguration>
    <atn>....</atn>
    <kv>....</kv>
    <rtt>....</rtt>
</unifiedTestConfiguration>
```

Within the RTT parent tags, we again find a three-part structure:

```
<rtt>
    <mailto>....</mailto>
    <jobList>....</jobList>
    <jobGroups>....</jobGroups>
</rtt>
```

where:

- The `<mailto>` tag contains email addresses to which the RTT will send a report if any jobs or post-job tests fail.
- The `<jobList>` tag contains the list of (typically Athena) jobs to run.

[1] In what follows, when the term "job" without the qualitative "RTT" is used, it means step 2 of a RTT job.

- The `<jobGroups>` tag contains the list of job groups defined by the package. A job group defines pre- and post-job actions/tests to execute. All RTT jobs must belong to one (and only one) job group, and all jobs in the same group receive the same treatment. Each job group also inherits from a parent job group, with the RTT itself providing common, base actions in a "Top" group from which every group ultimately inherits.

2.2.2. Into the ATLAS nightly build Once they have created this XML file, the developer must also add two lines to the package's CMT[5] requirements file which a) flag that this package has RTT jobs to run, and b) indicates the relative location within the package of the XML file. Subsequently, as shown in Figure 1, the developer will tag their package in the code repository and submit it to the ATLAS TagCollector[6] for inclusion within the next ATLAS nightly build of a particular branch. The next day, the RTT, running on this new nightly build, will detect the new package, read its unified configuration XML file, parse it into RTT jobs, submit them to batch and publish results to the web for the developer to peruse.

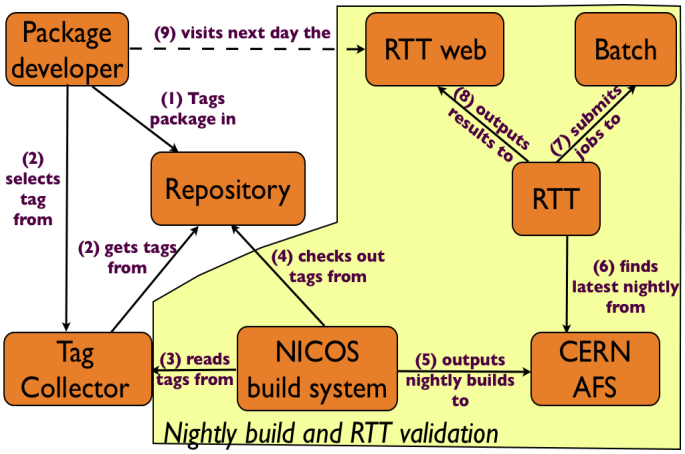

Figure 1. The nightly build procedure from new package tag to RTT validation of that tag.

2.3. RTT initialisation
The RTT is driven off an XML configuration file that, amongst other things, defines the exact ATLAS release to process. If its target is the latest nightly, the RTT checks on AFS for the "build ready" flag from NICOS. If unavailable, because NICOS is still building it, the RTT goes to sleep for 5 minutes before checking again. Once available, the RTT sets up for that particular nightly and, using CMT, requests all packages in the release that have the two lines mentioned previously. From this, the RTT obtains the list of paths to XML files within the release and, ultimately, the RTT jobs to submit to batch.

2.4. RTT Web Pages
Results from running client jobs in production at CERN are made available through the RTT website - `http://atlasrtt.cern.ch`. The site is designed to be cross-browser and cross-platform compatible. It has been demonstrated to perform equally well under major browsers such as Firefox, Mozilla, Safari, Opera, Internet Explorer, running on CERN Scientific Linux, Windows XP/Vista and Mac OS X platforms.

The site is also developer-customisable. It implements the concept of workspaces, to which packages of interest are added, or deleted, by the developer. Within the browser window,

Figure 2. Multiple workspaces can be created, appearing as tabs in the browser.

multiple workspaces appear as multiple tabs (Figure 2) which can be renamed, deleted or created. State is maintained between visits using PHP sessions.

Starting from a single, bare workspace on first visit, the developer, via the slide-out settings panel, selects those packages in those nightly branches that are of interest to them, and adds them to the workspace. Within a particular workspace, package order can be redefined using drag-n-drop.

Within a given package, multiple tabs are displayed, one for each nightly (Figure 3). Developers can thus conveniently flip between different nightlies of the same ATLAS build for the same package. Each tab displays two pictograms: one representing the logical AND of the status of all jobs run within the package for that nightly, the other a logical AND of all post-job tests executed. Selecting a particular nightly reveals a summary of performance for that nightly, and, below this, a number of rows, one row per RTT job. Each "job row" displays the job name, and two pictograms representing the job status and a logical AND of all post-job tests executed for this RTT job. Each job can be clicked and expanded *in situ*. Doing so reveals a two column structure with, on the left side, an overview of the RTT job (status of the (athena) job, statii of any post-job tests, history through the batch system, links to documentation for this job), and, on the right side, links to "keep" files for the job which the developer can download (Figure 4).

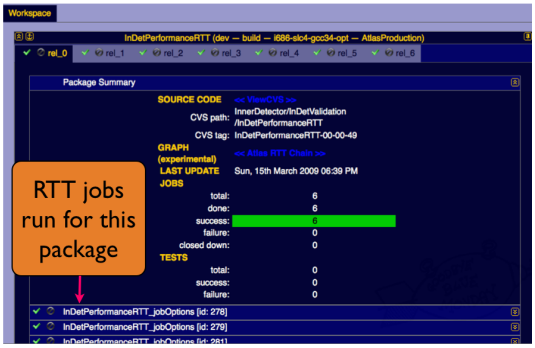

Figure 3. Results for a package for a particular ATLAS nightly, showing the RTT jobs below a package summary.

3. Hardware Resources

At CERN, the RTT launches several times daily, via the acron system tool, on one of thirteen launch nodes. Approximately 1300 client RTT jobs, from the short (∼1 hour) to the long (∼24 hours), are sent to one of three dedicated CERN LSF batch queues that map onto 480 cores spread over 70 dedicated batch nodes. Each core has 2GB of RAM. The 70-node batch cluster is heterogenous, containing three different processor types:

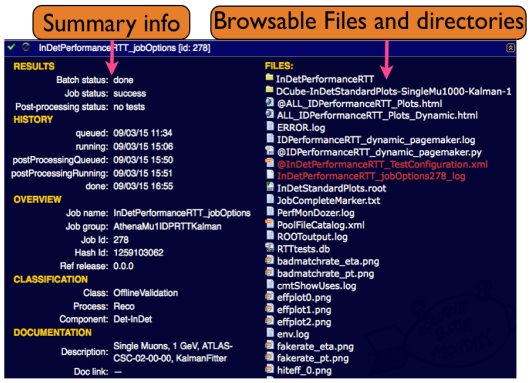

Figure 4. Unfolding a specific RTT job in a package reveals a summary and a list of downloadable "keep" files.

- 20 x Intel(R) Xeon(R) CPU 5150 @ 2.66GHz
- 33 x Intel(R) Xeon(R) CPU E5345 @ 2.33GHz
- 17 x Intel(R) Xeon(R) CPU E5410 @ 2.33GHz

Currently, the RTT results fill 1.4TB of CERN AFS disk space.

4. Future

The subsequent version of the results web interface will, for each package, provide a graph of its RTT jobs. This will be particularly interesting for chained jobs from the TCT and FCT frameworks where jobs are interdependent. In such cases, developers will see a tree like structure, an RTT job at each node of a branch, that will each turn green (ok) or red (fail) as the jobs complete. Clicking on a completed job will provide a drop-down list of the "keep" files for that job. This visual representation of a complicated structure should significantly aid understanding.

Finally, the RTT code base will continue to be tweaked for performance and stability improvements.

5. Conclusion

The ATLAS RunTimeTester software is a framework that provides a tool for automating setting up and running of developer jobs, and for presenting web-available results in a concise and intuitive manner. It is used by ATLAS in their offline software development cycle, fulfilling a need for computationally long tests, and feeding back problems each day on many ATLAS nightly builds.

References

[1] Luehring F et al. 2009 "Organisation, management and documentation of ATLAS offline software releases" *cf. these proceedings*
[2] http://www.usatlas.bnl.gov/computing/software/nicos/
[3] https://twiki.cern.ch/twiki/bin/view/Atlas/ATNightTesting
[4] http://cern.ch/atlasrtt
[5] www.cmtsite.org
[6] http://atlastagcollector.in2p3.fr