Testing in Simulation

Software TIM in Glasgow
08 June 2016

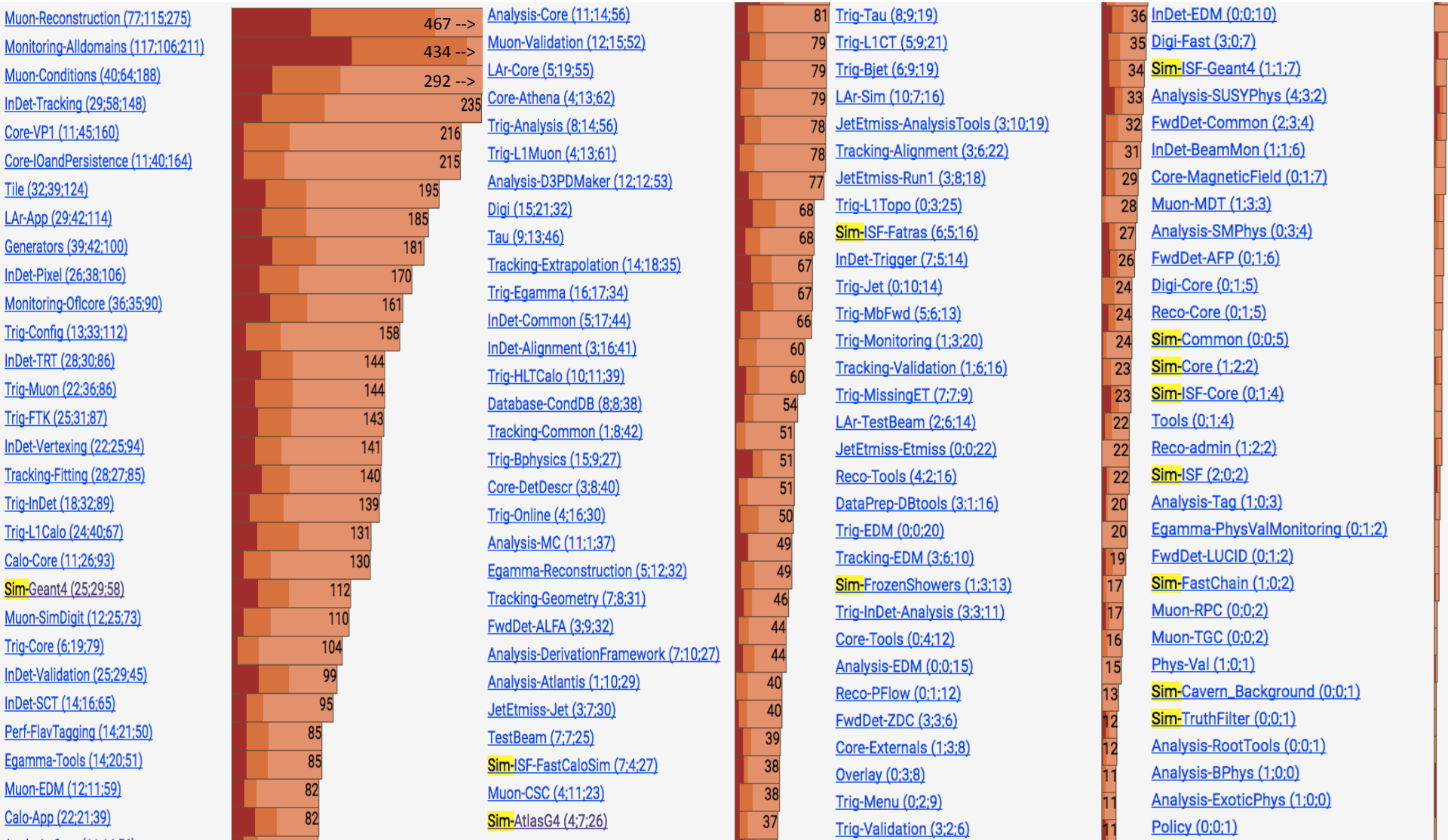
Elmar Ritsch (CERN)
for the Simulation Team

How we compare

Coverity Statistics

How we compare

Lizard Statistics



How we compare

RTT Usage

- Table taken from Brinick's talk at end of March (indico):

% CPU usage	% total jobs	Package
30.9%	7.8%	TrigInDetValidation
25.1%	7.1%	TriggerTest
11.1%	4.6%	TrigAnalysisTest
5.0%	22.5%	DerivationFrameworkRTT
4.4%	7.5%	Tier0ChainTests
4.0%	3.7%	RecJobTransformTests
3.8%	9.2%	ISF_Validation
2.4%	1.4%	TrigEgammaValidation
2.3%	9.2%	SimCoreTests
1.7%	4.5%	DigitizationTests
(9.3% CPU usage – 44 other packages)		

Developer-level Tests

- **Very few unit and "low-level" integration tests**
  - Just getting started in that field (more on next slides)
- **"Manual validation"**
  - e.g. running Athena simulation job a number of times to verify correctness of output during ongoing local development
  - Probably most of our testing done here
  - Tests are usually very specific to a new feature / bugfix
  - Testing code usually **not** put into `atlasoff` codebase
  - Probably too high a resource usage if we added those high-level tests to ATN/RTT for every development/bugfix

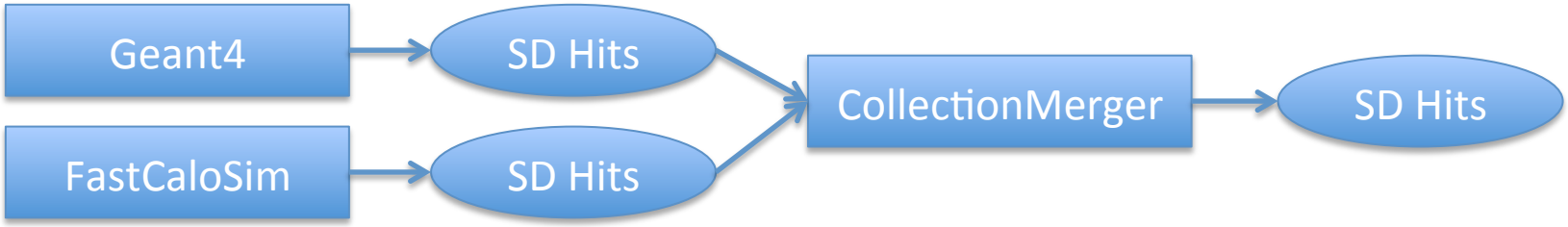
Nightly Tests

- **ATN tests**
  - Check that different simulation job configurations finish with status code 0, using only 1-2 events (e.g. fast simulation, very fast simulation, full simulation, …)
  - Not looking at the output
- **RTT regression tests**
  - Various production-like setups run through full chain: EVNT->HITS->RDO->ESD->AOD
  - Comparing plots from one nightly to the next:
    - HITS-level: number, location and energy of sensitive detector hits
    - RDO-level: number and location of energy deposits
    - ESD/ADO-level: `InDetStandardPlots.root` and `PhysValMon.root`
  - Due to high resource usage, running only a limited number of events:
    - FullG4: 4 ttbar events
    - ATLFASTII: 250 ttbar events
    - ATLFASTIIF: 2000 ttbar events

Validation Group Tests

- **Physics validation**
  - Used for **final sign-off** of new developments and bugfixes of current and future production releases
  - Due to high statistics and experts interpreting plots, usually very useful feedback in case there are problems
  - Good chance that even minute problems are identified
  - **BUT:** had at least one case where **output changed due to newly introduced bug**, **was noticed** in physics validation, but release **still passed** physics validation

First Experience using GoogleTest for Unit and Integration Testing



- Needed to implement `CollectionMerger` from scratch
- Used GoogleTest to test CollectionMerger implementation from start
- **Speeds up development process**
  - no need to re-run simulation job over and over again to check the output! (takes ~5 mins until end of first event is reached)

- Testing code:
  - https://svnweb.cern.ch/trac/atlasoff/browser/Simulation/ISF/ISF_Core/ISF_Algorithms/trunk/test/CollectionMerger_test.cxx?rev=749814

Where we want to improve I

- **More unit and "low-level" integration tests**
  - Much faster turn-around time during development
  - Potential for Test Driven Development (TDD)

- **Problem**
  - Currently hardly any tests at this level implemented.
  - Therefore seems less work to just fix the bug and not also go through the hassle to make the already implemented code testable (would often require refactoring)

Where we want to improve II

- **Introduce code reviews**
  - Not a “test” strictly speaking
  - **Spreads the knowledge** of commonly developed code base and programming knowledge among entire developer group
  - Improves readability of code / reduces code complexity
    ➔ improves robustness

- **Problem**
  - Currently knowledge of code base is not evenly distributed among developers
    - Reviewers in the beginning will be overwhelmed
  - Some people are very protective of “their code” or “their packages”
    - I think is a consequence of not having code reviews, e.g.:
      “I don’t want anyone to mess up the clearly structured code that I wrote and that has my name in it”
    - With a review one has a better guarantee that other people are not “messing up your code”