geant-val.cern.ch: GEANT Validation webpage

D. Konstantinov for geant-val developers

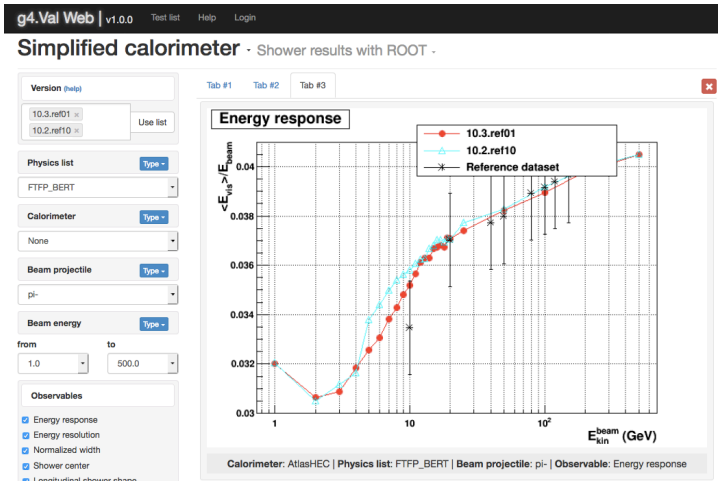
Introduction

The old Geant4 validation page (g4-val3.cern.ch) created by technical student George back in 2013 revealed some remarkable shortcomings:

not scalable – an addition of new test/histogram requires manual creation of new DB tables (keeps "histograms" in MySQL database) and also requires new web application for every new test.

no input/output format – no possibility to inject locally produced data or results produced on CERN LXF, no access to histograms

GRID submission only – all API scripts



New Developments

New developments started last summer with work of summer student (Ioana)

She create a nice validation suite prototype based on modern IT technologies such as:

node.js + angular.js – light-weighted web server and web application;

bootstrap – nice looking forms, buttons, navigation, adaptive web design;

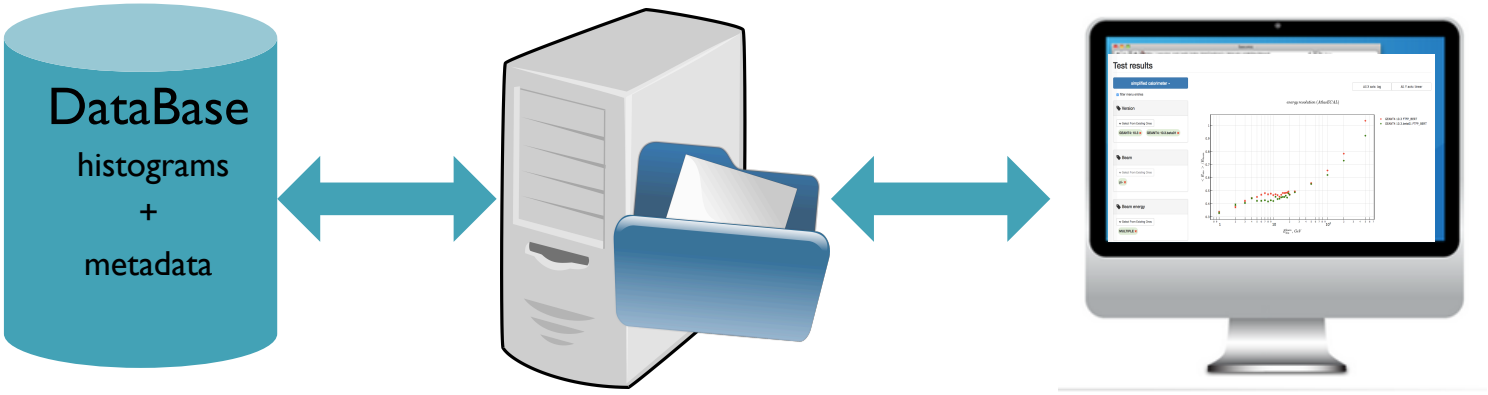
New Developments

production site: geant-val.cern.ch
development site: val.cern.ch:63080

As a database we use CERN "database on demand" with PostgreSQL

The DB contains the following information for each plot (histogram/ scatter plot), the DB schema is a reworked schema of DOSSIER DB.

plot with errors itself, package name, package version, physics model, projectile particle , projectile energy, target material, secondary particle etc



Web client
Node.js

JSON input/output format

In order to upload data to database we use JSON format.
Example of "simplified calo" histogram is presented below:

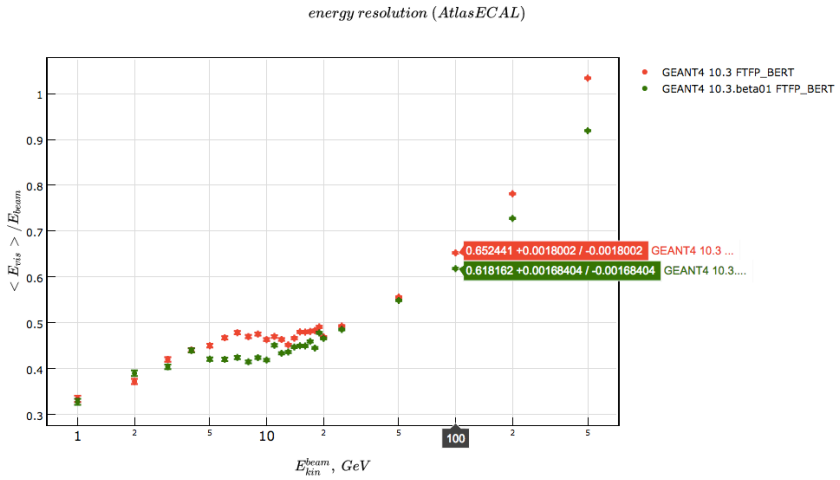

```
{
  "id": 134652,
  "article": {
    "inspireId": -1
  },
  "mctool": {
    "name": "GEANT4",
    "version": "10.3",
    "model": "FTFP_BERT"
  },
  "testName": "simplified calorimeter",
  "metadata": {
    "observableName": "energy resolution",
    "reaction": "particle production",
    "targetName": "AtlasECAL",
    "beamParticle": "pi-",
    "beamEnergies": [
      1,
      2
    ],
    "secondaryParticle": "None",
    "parameters": [

    ]
  },
  "plotType": "SCATTER2D",
  "chart": {
    "nPoints": 25,
    "xValues": [
      1,
      2
    ],
    "yValues": [
      0.333964,
      0.371813
    ],
    "xStatErrorsPlus": [
      0,
      0
    ],
    "yStatErrorsPlus": [
      0.00690332,
      0.00557068
    ],
    "xStatErrorsMinus": [
      0,
      0
    ],
    "yStatErrorsMinus": [
      0.00690332,
      0.00557068
    ],
    "xSysErrorsPlus": [
      0,
      0
    ],
    "ySysErrorsPlus": [
      0,
      0
    ],
    "xSysErrorsMinus": [
      0,
      0
    ],
    "title": "energy resolution",
    "xAxisName": "E_{kin}^{beam}, GeV",
    "yAxisName": "<E_{vis}>\/E_{beam}"
  }
}
```


all data can be downloaded in the same format.

http://val.cern.ch:63080/api/get/134652

Visualization tool

Initially as a plotting tool initially we selected JavaScript data visualization "plot.ly"



advantages:

- interactive
- nice plots without significant efforts
- input data in JSON format
- can handle LATEX

disadvantages:

- slow as executed on client side
- very slow handling of LATEX (redrawing several axis several times)
- no possibility to change axis and title positions
- no simple way to save in local file
- big object – all data kept in memory on client side -> creation of many plots are slow

ROOT as visualization tool

Insertion of test22 has been shown that "plot.ly" is extremely slow plotting more than ~20 plots on one page.

Hang my browser several times digesting data. ☹

Decided to try static ROOT plots.

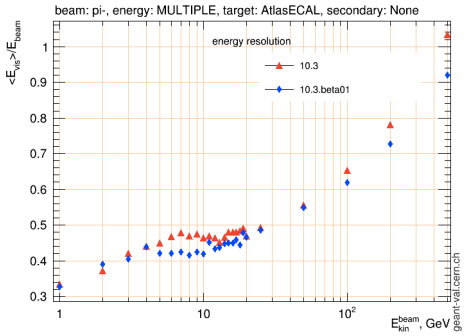
The very first version of "plotter" prototype is written.

(not production quality yet)

./plotter.exe 146466 146470 146474 146478

First results are very promising:

- very fast drawing
- more freedom with legends, titles



can be seen on dev web-page: http://val.cern.ch:63080

Put tick near ☑ ROOT plots

How to introduce new test

- compile and place a test executable for given GEANT version into geant4 cvmfs:

```
/cvmfs/geant4.cern.ch/opt/10.3.ref02/x86_64-slc6-gcc49-opt/bin/
├── Hadr00
├── StatAccepTest
├── test22_HARP
├── test22_main
├── test30
└── test46
```

Not simple procedure as we have different repositories for tests, different way to configure and build!

- run the test using lxbatch or DIRAC (for GRID)

Not simple as tests have different configuration style!

- using a dedicated python script extract histogram and fill input JSON files

Not simple as tests have different output, need a script for every test!

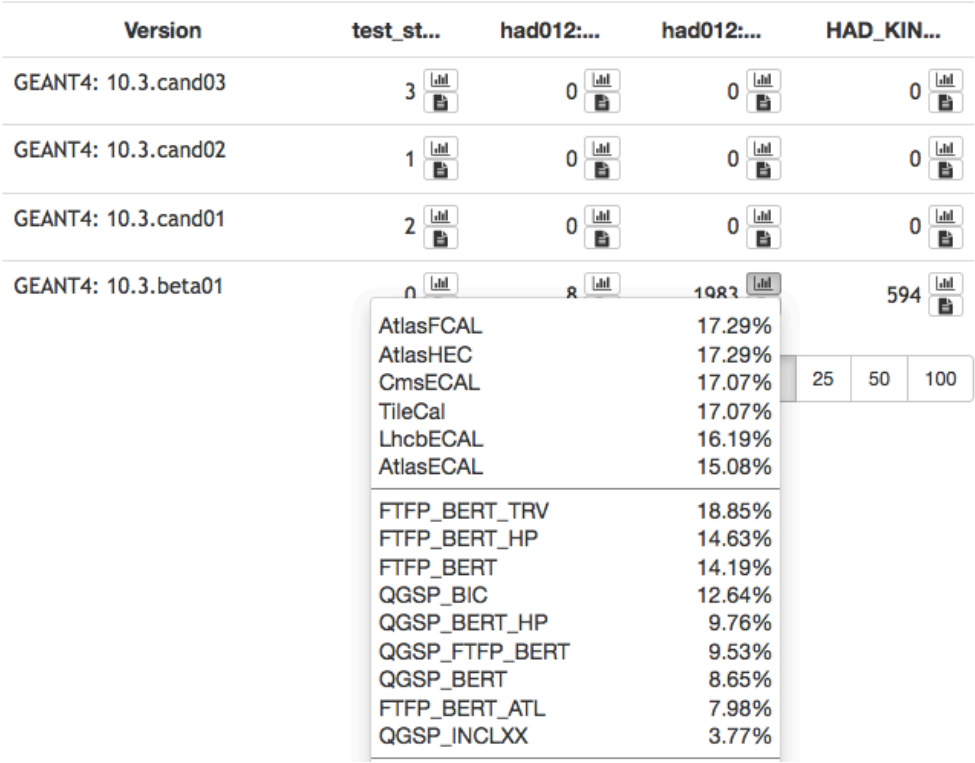
- fill DB with JSON files content.

That is it! You can browse/compare your data at geant-val.cern.ch!

“Simplified calo”: exceptions stats monitoring

geant-val.cern.ch/exceptions

For “simplified calo” we created special web layout for G4Exceptions monitoring – different error types, frequency with possibility to look at full error message.



Plans

- improve ROOT plots quality.
- add possibility to have ratio plots.
- add stat test:
  - start with chi2.
  - investigate possibility to use stat code developed by Marilena.
- setup new DIRAC server or use CLIC DIRAC instance for GRID production.
- integration of new tests, new tests, new tests…
- different presentation of results – starting from "summary" stat results
- a lot of minor improvements and bug fixes in pipe line

Backup slides

Node.js

- Node.js is an open-source, cross-platform JavaScript environment for developing a variety of tools and applications.

Angular.js

- Angular.js is a complete JavaScript-based open-source front-end web application framework. It lets you use HTML as your template language.

Bootstrap

- Bootstrap is a free and open-source front-end web framework for designing websites and web applications.