UI Updates

Koichi Murakami (KEK)

Virtual Geant4 Collaboration Meeting 2020

Dropping Compoments

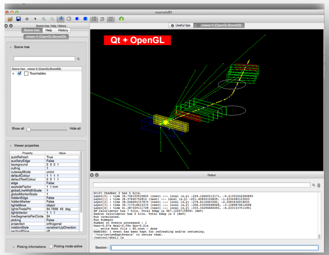
- Momo will be dropped in the v11 release.
  - Java Front-end for G4 application builder

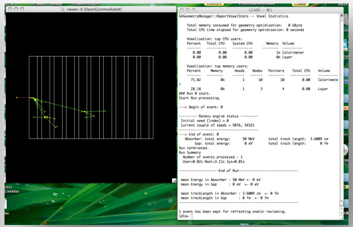
- Python2 : End of life
  - Python2 became End of Life in Apr/2020.
  - Python2 codes will be dropped in the v11 release.
  - Only support Python3 codes

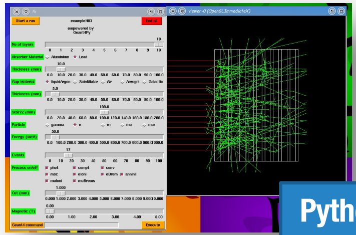
Python

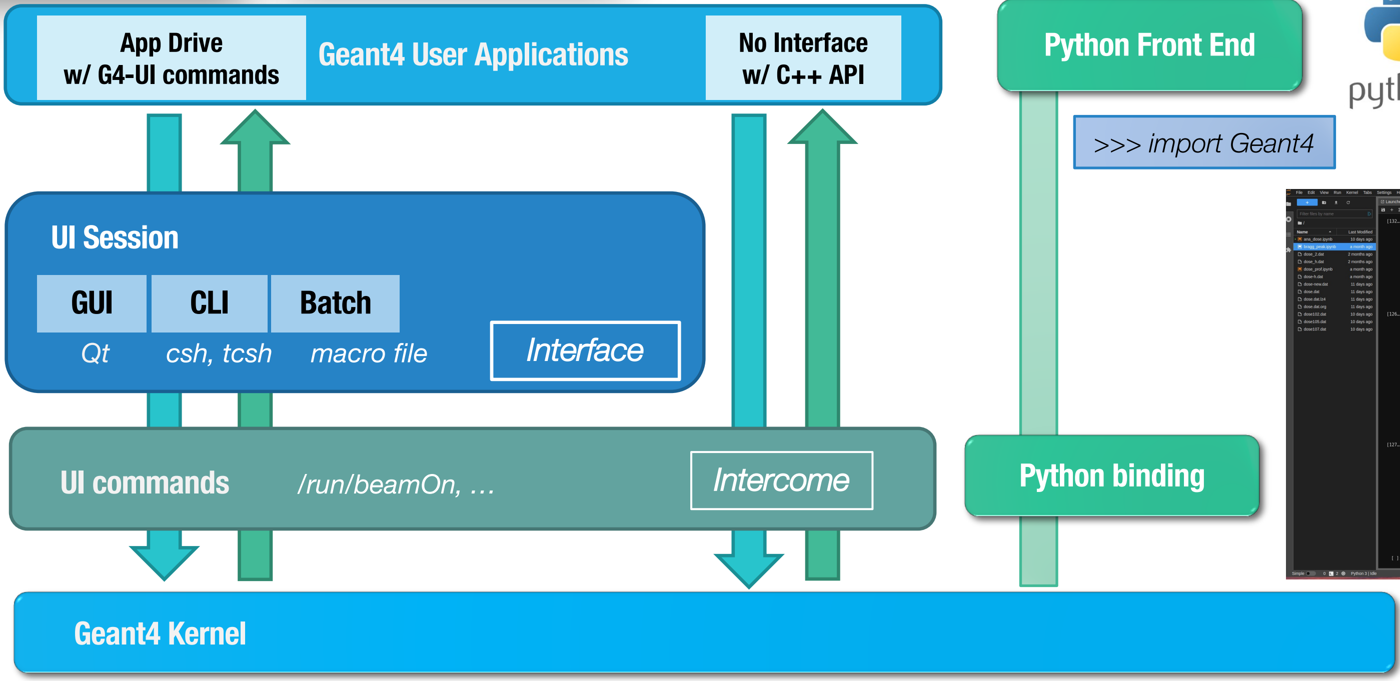
- Python is more popular environment in science.
  - important tool in data science
  - packages : Anaconda (numpy, scipy, …)
  - data management : pandas.DataFrame
  - plot tools : matplotlib, plotly, …
  - Jupyter (JupyterLab, JupyterNotebook) ecosystem

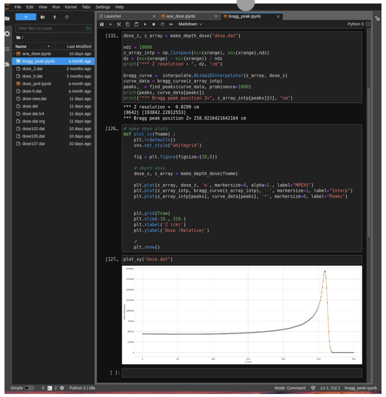
Geant4 UI & App.









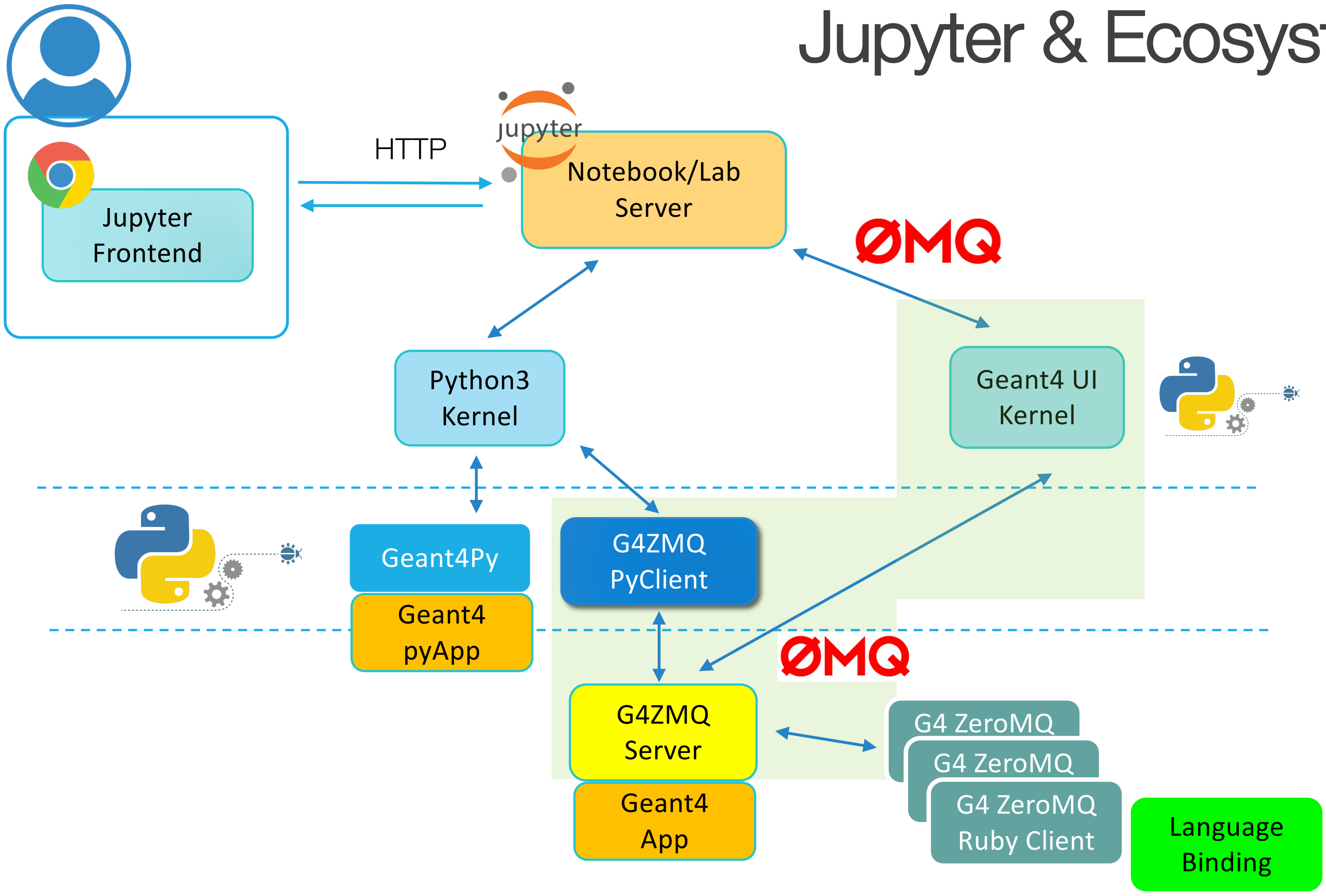


```
[132… dose_z, z_array = make_depth_dose("dose.dat")

      ndz = 10000
      z_array_intp = np.linspace(min(xrange), max(xrange),ndz)
      dz = (max(xrange) - min(xrange)) / ndz
      print("*** Z resolution = ", dz, "cm")

      bragg_curve =  interpolate.Akima1DInterpolator(z_array, dose_z)
      curve_data = bragg_curve(z_array_intp)
      peaks,_ = find_peaks(curve_data, prominence=1000)
      print(peaks, curve_data[peaks])
      print("*** Bragg peak position Z=", z_array_intp[peaks][0], "cm")

      *** Z resolution =  0.0299 cm
      [8642] [193842.22012553]
      *** Bragg peak position Z= 258.9216421642164 cm

[126… # make dose plots
      def plot_xy(fname) :
          plt.rcdefaults()
          sns.set_style("whitegrid")

          fig = plt.figure(figsize=(20,8))

          # depth dose
          dose_z, z_array = make_depth_dose(fname)

          plt.plot(z_array, dose_z, 'o', markersize=4, alpha=1., label="MPEXS")
          plt.plot(z_array_intp, bragg_curve(z_array_intp), '-', markersize=1, label="Interp")
          plt.plot(z_array_intp[peaks], curve_data[peaks], '*', markersize=8, label="Peaks")


          plt.grid(True)
          plt.xlim(-10., 310.)
          plt.xlabel('Z (cm)')
          plt.ylabel('Dose (Relative)')

          #
          plt.show()

[127… plot_xy("dose.dat")
```

Boost.python to Pybind11

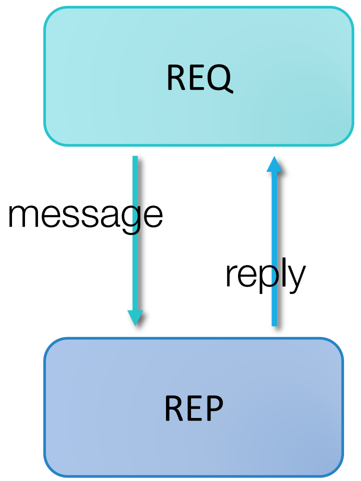
- Backend tools for C++ binding to Python
- pybind11
  - https://github.com/pybind/pybind11
  - Seamless operability between C++11 and Python
  - header only
  - C++11 (modern C++) support
  - STL container support
  - wrapper approach is very similar to boost.python
- reviewing binding Geant4 classes/methods
- new example using Jupyter-lab and save as ipynb file
  - Anaconda : CSV data + pandas + matplotlib -> full chain of simulation and analysis
  - VTK (paraview) + Jupyter / WebGL (glTF) + Jupyter

Jupyter & Ecosystem



ZeroMQ Message Backend

- light-weight socket API
  - very popular tool : stable, easy to install
- support different message patterns
  - REQ – REP (Request – Reply model)
  - send a message from client to sever
  - reply a message from sever to client
  - …
- Many language bindings
  - C/C++, Python, Ruby, PHP, Perl, Java, ...



G4ZMQServer

- An alternative UI session like
  - UI terminal, Qt session, Batch session, ...
  - /environments/zmq

```
auto zmq_session = new G4ZMQServer();
zmq_session-> SetEndpoint(endpoint);
// endpoint is like tcp://127.0.0.1:5555
zmq_session -> SessionStart();
```

- Waiting a message / Receive a message /
  Execute UI (primitive) commands / Send back an output

ZeroMQ client

- Clients can be in any languages that ZeroMQ is bound.
- Simple pyclient module : *g4zmq (written in Python)*

```
>>> import g4zmq
>>> g4zmq.connect()
>>> g4zmq.ls("/run")
>>> g4zmq.help("/run/beamOn")
>>> g4zmq.apply("/run/beamOn 10")
```

- Language bindings :
  - Python, Julia, Ruby, Perl, PHP, JAVA, ...
  - http://zeromq.org/bindings:_start

Notes on ZMQ interface

- ZMQ interface is an alternative of UI session.
- You can drive Geant4 app in the same way as UI terminal.
  - send a UI command, execute, get a response (output)
  - more friendly for scripting
- CANNOT directly access to Geant4 objects like native Python interface approach (Geant4Py).

IGeant4

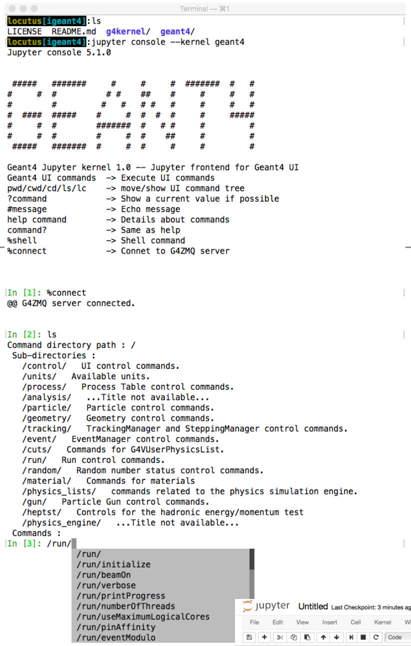
Geant4 UI Kernel for Jupyter

https://github.com/koichi-murakami/igeant4

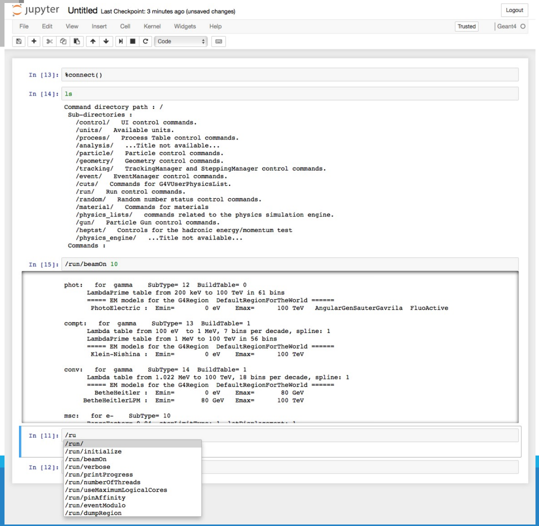
```
# jupyter console --kernel geant4
```

You can do:

- connect to G4 zmq server
- execute UI / shell commands
- completion
- history
- shell (bash) exec (ex. %shell ls)



```
locutus[igeant4]:ls
LICENSE  README.md  g4kernel/  geant4/
locutus[igeant4]:jupyter console --kernel geant4
Jupyter console 5.1.0


 #####   #######     #     #     #  #######  #   #
#     #  #          # #    ##    #     #     #   #
#        #         #   #   # #   #     #     #   #
#  ####  #####    #     #  #  #  #     #     #####
#     #  #        #######  #   # #     #         #
#     #  #        #     #  #    ##     #         #
 #####   #######  #     #  #     #     #         #

Geant4 Jupyter kernel 1.0 -- Jupyter frontend for Geant4 UI
Geant4 UI commands  -> Execute UI commands
pwd/cwd/cd/ls/lc    -> move/show UI command tree
?command            -> Show a current value if possible
#message            -> Echo message
help command        -> Details about commands
command?            -> Same as help
%shell              -> Shell command
%connect            -> Connet to G4ZMQ server



In [1]: %connect
@@ G4ZMQ server connected.


In [2]: ls
Command directory path : /
 Sub-directories :
   /control/   UI control commands.
   /units/   Available units.
   /process/   Process Table control commands.
   /analysis/   ...Title not available...
   /particle/   Particle control commands.
   /geometry/   Geometry control commands.
   /tracking/   TrackingManager and SteppingManager control commands.
   /event/   EventManager control commands.
   /cuts/   Commands for G4VUserPhysicsList.
   /run/   Run control commands.
   /random/   Random number status control commands.
   /material/   Commands for materials
   /physics_lists/   commands related to the physics simulation engine.
   /gun/   Particle Gun control commands.
   /heptst/   Controls for the hadronic energy/momentum test
   /physics_engine/   ...Title not available...
 Commands :
In [3]: /run/
             /run/
             /run/initialize
             /run/beamOn
             /run/verbose
             /run/printProgress
             /run/numberOfThreads
             /run/useMaximumLogicalCores
             /run/pinAffinity
             /run/eventModulo
```



```
In [13]: %connect()

In [14]: ls

Command directory path : /
 Sub-directories :
   /control/   UI control commands.
   /units/   Available units.
   /process/   Process Table control commands.
   /analysis/   ...Title not available...
   /particle/   Particle control commands.
   /geometry/   Geometry control commands.
   /tracking/   TrackingManager and SteppingManager control commands.
   /event/   EventManager control commands.
   /cuts/   Commands for G4VUserPhysicsList.
   /run/   Run control commands.
   /random/   Random number status control commands.
   /material/   Commands for materials
   /physics_lists/   commands related to the physics simulation engine.
   /gun/   Particle Gun control commands.
   /heptst/   Controls for the hadronic energy/momentum test
   /physics_engine/   ...Title not available...
 Commands :

In [15]: /run/beamOn 10

phot:   for  gamma    SubType= 12  BuildTable= 0
      LambdaPrime table from 200 keV to 100 TeV in 61 bins
      ===== EM models for the G4Region  DefaultRegionForTheWorld ======
       PhotoElectric :  Emin=        0 eV    Emax=      100 TeV   AngularGenSauterGavrila  FluoActive

compt:   for  gamma    SubType= 13  BuildTable= 1
      Lambda table from 100 eV  to 1 MeV, 7 bins per decade, spline: 1
      LambdaPrime table from 1 MeV to 100 TeV in 56 bins
      ===== EM models for the G4Region  DefaultRegionForTheWorld ======
       Klein-Nishina :  Emin=        0 eV    Emax=      100 TeV

conv:   for  gamma    SubType= 14  BuildTable= 1
      Lambda table from 1.022 MeV to 100 TeV, 18 bins per decade, spline: 1
      ===== EM models for the G4Region  DefaultRegionForTheWorld ======
        BetheHeitler :  Emin=        0 eV    Emax=       80 GeV
     BetheHeitlerLPM :  Emin=       80 GeV   Emax=      100 TeV

msc:   for e-    SubType= 10

In [11]: /ru
         /run/
         /run/initialize
         /run/beamOn
         /run/verbose
         /run/printProgress
         /run/numberOfThreads
         /run/useMaximumLogicalCores
         /run/pinAffinity
         /run/eventModulo
         /run/dumpRegion
In [12]:
```

MPI Issue

- MPI: A Message-Passing Interface Standard Version 3.1
  Message Passing Interface Forum
  says

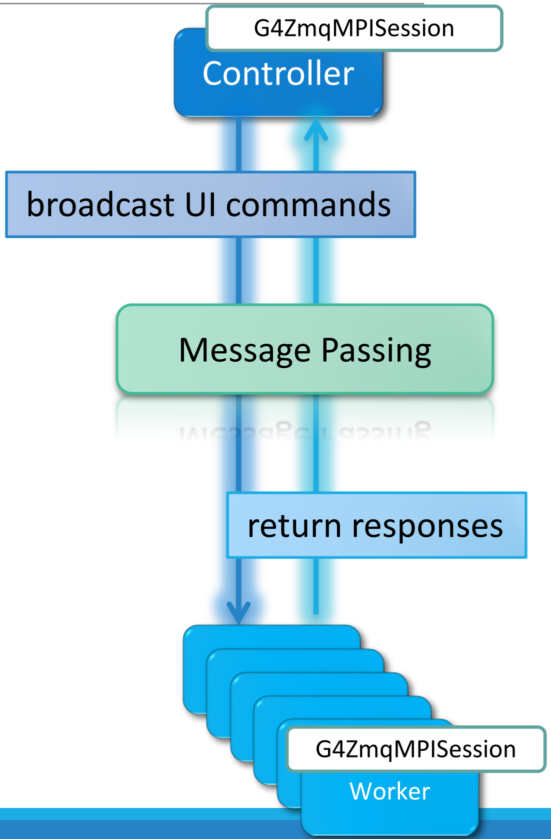
"The C++ bindings were deprecated as of MPI-2.2. The C++ bindings are removed in MPI-3.0. The namespace is still reserved, however, and bindings may only be provided by an implementation as described in the MPI-2.2 standard. "

  - The situation does not change in the 2019 draft specification, hoeplessly in MPI-4

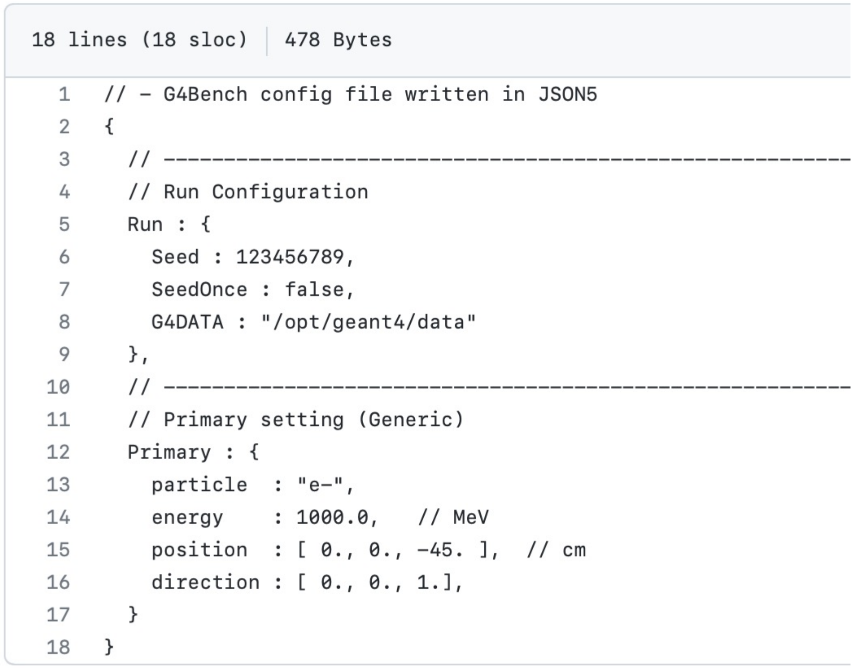
- openMPI v1.10.1 is the latest version of MPI-2, that supports C++ binding.
  - The software status is 'retired'.

Message Passing Scheme

- Geant4-MPI interface is based on simple scatter/gather model.
  - Implemented in “interface” layer.
  - Isolated from G4kernel
  - Broadcasting UI commands (from controller to workers)
  - No intercommunications between worker nodes
- ZeroMQ can be used as a backend of message passing.
  - plan to release in v11 (experimental)
  - message distributor is reusable.
  - additional functionalities of controller (node config, remote shell…)
  - communication model of efficient reduce for large scale computing



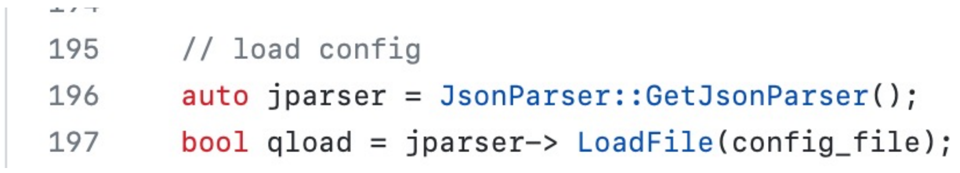
New proposal for JSON config



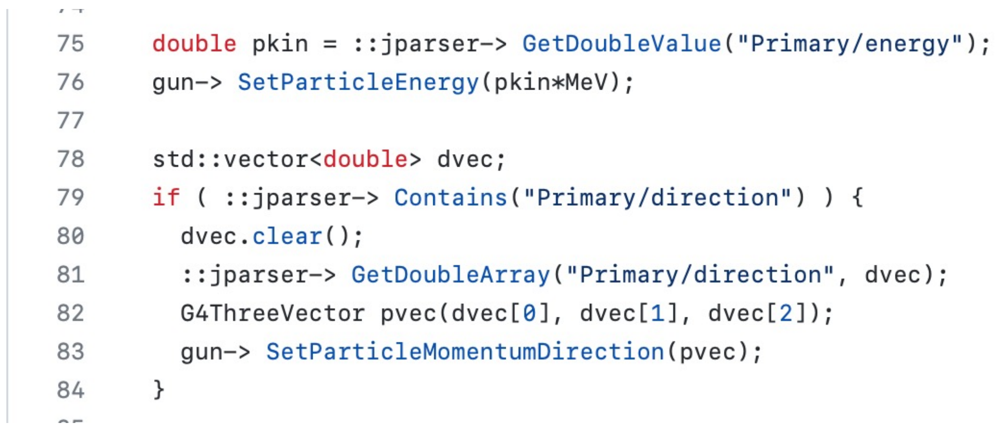
```
18 lines (18 sloc) | 478 Bytes
// - G4Bench config file written in JSON5
{
  // ---------------------------------------------------------
  // Run Configuration
  Run : {
    Seed : 123456789,
    SeedOnce : false,
    G4DATA : "/opt/geant4/data"
  },
  // ---------------------------------------------------------
  // Primary setting (Generic)
  Primary : {
    particle  : "e-",
    energy    : 1000.0,    // MeV
    position  : [ 0., 0., -45. ],  // cm
    direction : [ 0., 0., 1.],
  }
}
```

- Use a JSON as configuration file
  - This example is written in JSON5
  - So far, no association with G4 UI commands

How to use



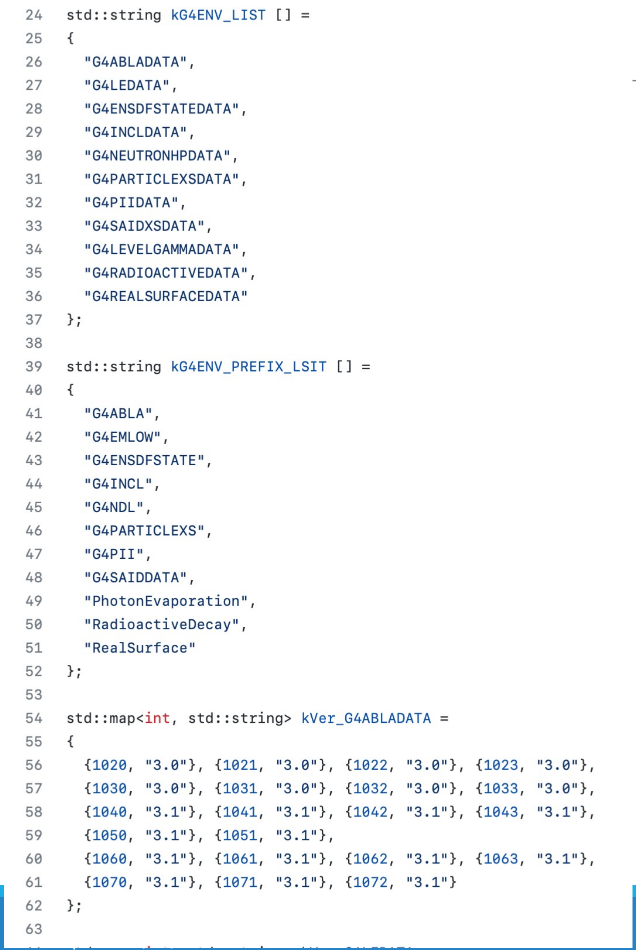
```
// load config
auto jparser = JsonParser::GetJsonParser();
bool qload = jparser-> LoadFile(config_file);
```



```
double pkin = ::jparser-> GetDoubleValue("Primary/energy");
gun-> SetParticleEnergy(pkin*MeV);

std::vector<double> dvec;
if ( ::jparser-> Contains("Primary/direction") ) {
  dvec.clear();
  ::jparser-> GetDoubleArray("Primary/direction", dvec);
  G4ThreeVector pvec(dvec[0], dvec[1], dvec[2]);
  gun-> SetParticleMomentumDirection(pvec);
}
```

- JsonParser class as a parser
  - PicoJSON is used as JSON parser.
  - nlohmann/json is too strict for some expressions
  - JSON / JSON5
  - namespace feature for multile files
- JSON-pointer-like access
  - “Primary/energy”
  - “Primary/direction”
- Currently 100% user code
- Can associate G4UI commands as pre-loader

What about this? : G4 Data



```
std::string kG4ENV_LIST [] =
{
  "G4ABLADATA",
  "G4LEDATA",
  "G4ENSDFSTATEDATA",
  "G4INCLDATA",
  "G4NEUTRONHPDATA",
  "G4PARTICLEXSDATA",
  "G4PIIDATA",
  "G4SAIDXSDATA",
  "G4LEVELGAMMADATA",
  "G4RADIOACTIVEDATA",
  "G4REALSURFACEDATA"
};

std::string kG4ENV_PREFIX_LSIT [] =
{
  "G4ABLA",
  "G4EMLOW",
  "G4ENSDFSTATE",
  "G4INCL",
  "G4NDL",
  "G4PARTICLEXS",
  "G4PII",
  "G4SAIDDATA",
  "PhotonEvaporation",
  "RadioactiveDecay",
  "RealSurface"
};

std::map<int, std::string> kVer_G4ABLADATA =
{
  {1020, "3.0"}, {1021, "3.0"}, {1022, "3.0"}, {1023, "3.0"},
  {1030, "3.0"}, {1031, "3.0"}, {1032, "3.0"}, {1033, "3.0"},
  {1040, "3.1"}, {1041, "3.1"}, {1042, "3.1"}, {1043, "3.1"},
  {1050, "3.1"}, {1051, "3.1"},
  {1060, "3.1"}, {1061, "3.1"}, {1062, "3.1"}, {1063, "3.1"},
  {1070, "3.1"}, {1071, "3.1"}, {1072, "3.1"}
};

```

- No need to be bothered by environment variables.
- If G4 environment variables are not set, try to use these values. (check for each)
  - if G4 envs are set, then use them
  - fully compatible with conventional way
  - stil need to specify where G4 data is located (top directory for data)
  - setenv() is not POSIX (concern)
    - use _putenv_s() for MSVS

Summary

- Dropping / Changing features
  - Momo
  - Python v2 codes
  - pybind11 : tool change for python-C++ binding

- ZeroMQ as message passing backend
  - G4 ZeroMQ Server as a UI session. -> Allows to interface with any languages
  - Geant4 UI language kernel for Jupyter
  - Alternative backend for MPI-like Message Passing Interface