RAT-PAC Geometry and Chroma Update

James Shen
University of Pennsylvania
Theia LBL Meeting, 04/22/2024

Rat-pac setup

- Full build chain CI is now setup: ratpac-setup -> ratpac2 -> ratpacExperiment
- Dockers are available: https://hub.docker.com/repository/docker/ratpac/ratpac-two/general
- Singularity images can be built from docker easily
- Currently supported dependencies:
    - Built by default:
        - root
        - Geant4
    - Not built, but supported by ratpac-setup:
        - Chroma (plan on providing dedicated tagged builds)
        - Cry
        - Tensorflow (plan on providing dedicated tagged builds)
        - torch
        - ratpac (dedicated tag exists)
        - nlopt

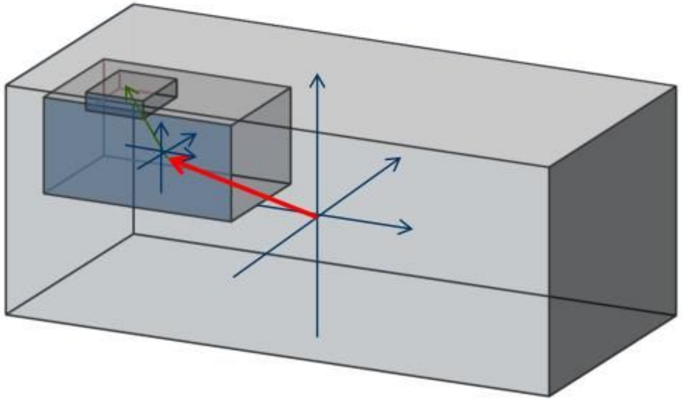
Geant4 Geometry Structure

Geant4 geometries are defined as a hierarchy of “volumes”. The volume tree is traversed during simulation to compute particle intersections.

“Overlap”: Geant4 jargon for violations of this hierarchy.

- Sibling volumes that have a non-zero intersecting volume
- Child volumes protruding out of their parent.

Overlaps won’t “break” a simulation, but will result in unexpected/undefined simulation results.



RAT-PAC2/TheiaSimulation Optimization

Geant4 Voxelizes volumes by their smallers neighbors.

Tips for improving CPU Performance of programs using Geant4 (https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4PerformanceTips):

“Create a hierarchy of volumes, if possible, when **dealing with thousands of volumes**, rather than placing all volumes in one flat space. Especially if there are areas of a setup or detector which have very different typical volume size (eg millimeters near an interaction point, meters far away) there will be a benefit in navigating if the parts of the setup are separated into different volumes.”

- PMTs were previously directly placed in the inner detector volume.
- Add a “fiducal volume” inside the inner detector that covers majority of the inner detector but not the PMTs.
- Speed up is 100x and above!

RAT enforces *zero* overlap checks. In reality, the simulation errors due to overlaps are minimal (given that the overlaps are minimal).

However, the same is not true for chroma!

Chroma adopts a mesh-based geometry: material is assigned by triangles at volume surfaces, “volume” as a structure does not exist.

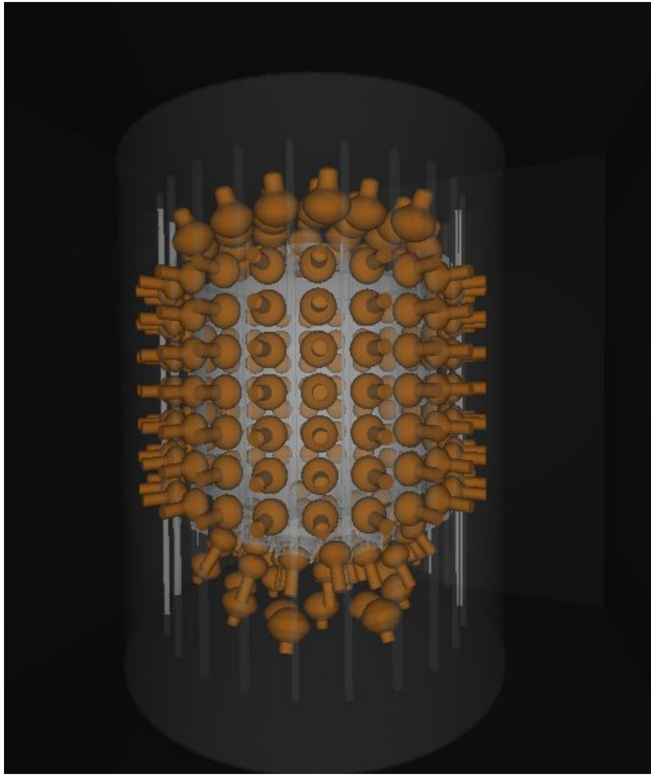
Meaning: *Small amount of overlap may result in completely bogus simulation results!*

Geant4 overlap checks

- Geant4 has the capability of performing overlap checks on its geometry using the macro: `/geometry/test/ run`
- Unfortunately, this is not supported by the custom geometry class GLG4TorusStack.
    - We need to implement GetPointOnSurface() – but this is not trivial due to how volume tolerance is handled in TorusStack.

Chroma automatic geometry generation

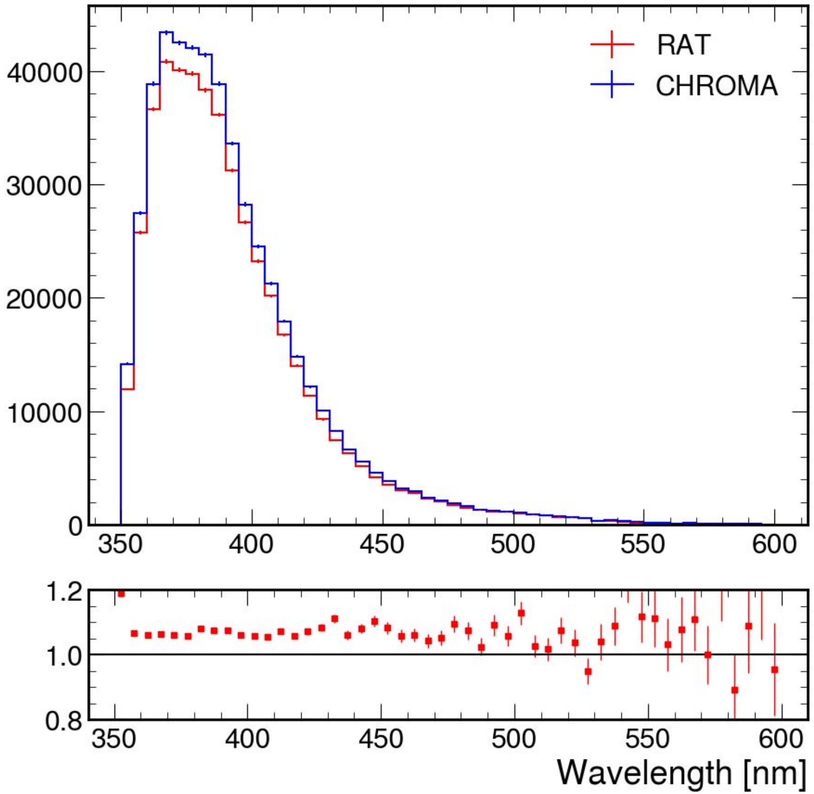
- Ratpac2/geant4 geometry
- GDML & Ratdb Json dump
  - Copied geant4's own gdml writer to ratpac2, allow small changed to conform to our specific needs (write torus stacks, dichroic surface info, etc…)
  - GDML: all geometry definition, material properties (as seen by geant4)
  - Ratdb is only used to extract PMTInfo at the moment
- **Chroma RatGeoLoader**
  - Generate meshes for all geant4 volumes using gmsh
  - Conform all meshes (merge shared surfaces) – **requires G4 geometry to be 100% free of overlaps!**
  - Keep track of and assign materials to the correct triangles
  - **Takes about 20 minutes to complete for EOS**
- Chroma.detector object, pickled for ease of use during simulation



Eos in Chroma

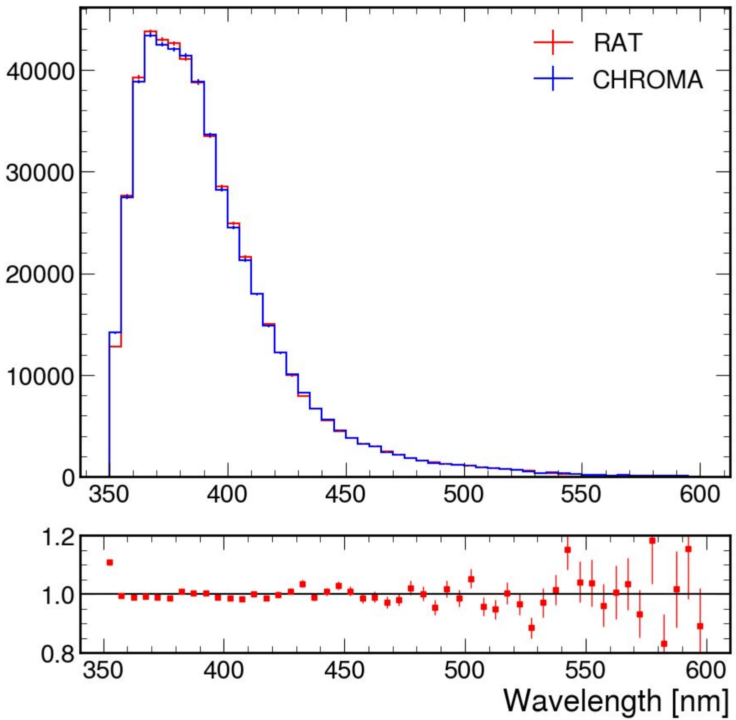
Wavelength Distribution

- 2MeV electron, simulated at the AV Center
- 10% WbLS



Wavelength Distribution, global efficiency calibrated

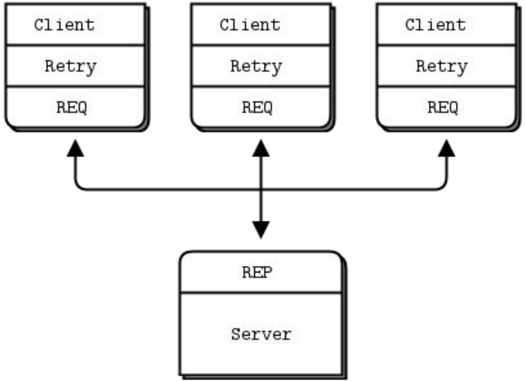
- Multiply RAT Light yield by 1.07



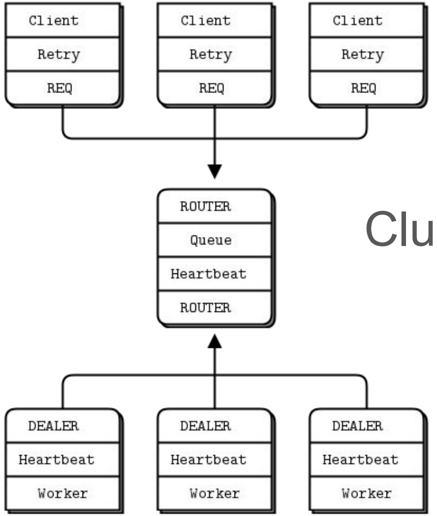
Running simulation with Chroma

Introducing ChAR0N: Chroma And Rat 0mq Network

- CHAR0N + Chroma acts as a server, while rat act as a client.
- RAT requests a simulation to be completed, chroma responds with the simulated result.
- Run as a “simple worker”: Single chroma instance, can communicate with multiple rat clients
- Or as a cluster: A persistent router instance connects to all rat clients in the front end, and deploy jobs to chroma workers in the backend.



SimpleWorker



Cluster

Handshake procedure

- Ping-Pong handshake:
  - Rat sends PING, server/router replies PONG
- Detector Info exchange
  - Rat requests DETINFO, server responds with its PMT channel position/type mapping
- PHOTONDATA / SIM request
  - Rat sends all photon vertices it has generated for a specific event (pos, dir, pol, t, wvl).
  - Chroma responds with the PEs that it has propagated (ch_id, t, wvl)
- Rat writes the simulated PEs in a separate output file. Plan to incorporate it to the rat DSWriter and have it written in ntuples just like regular PEs.

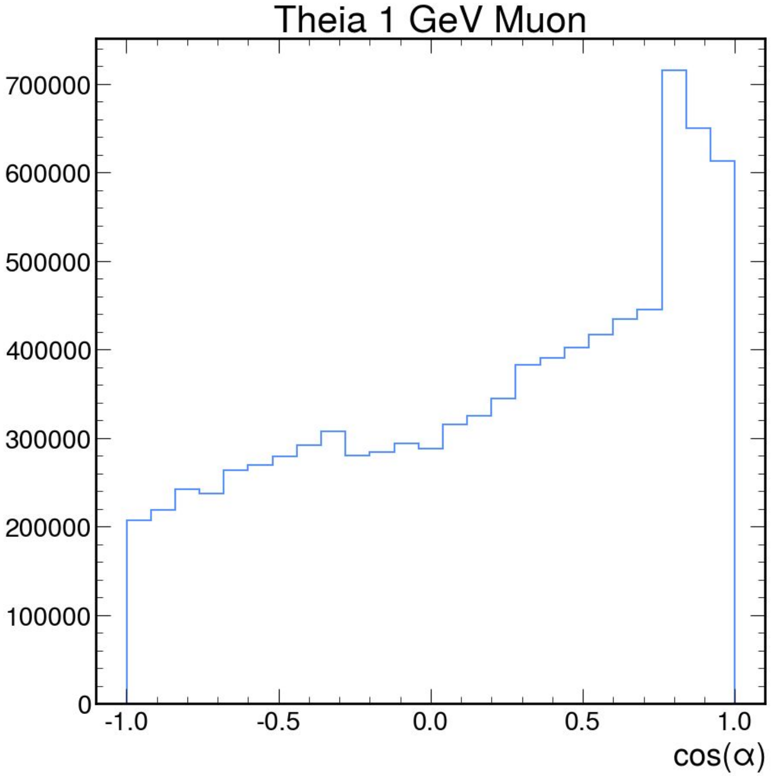
Benchmark

- Consider rat "event source" time – no processors
- Eos:
  - 2MeV electron, 10% WbLS
    - RAT: 0.071s/evt
    - Chroma: 0.041s/evt
  - 100MeV electron, 10% WbLS, 1 rat process
    - RAT: 3.8s/evt
    - Chroma: 0.57s/evt (1 rat process)
    - Chroma: Chroma: 0.21s/evt (3 rat process)
  - 1 GeV muon, 10% WbLS
    - RAT: 11.314 s/evt
    - Chroma: 2.4 s/evt (1 rat thread)
    - Chroma: 0.37s/evt(8 rat process)

Benchmark

Theia:

- Wbls 5pct, 1 GeV Muon
    - Rat: 26s/evt
    - Chroma: 2.0s/evt (1 rat process)
    - Chroma: 0.4s/evt (6 rat process)



Recently identified broader interest in Chroma in other collaborations:

(nEXO, LEGEND, MicroBoone, etc)

**Chroma Developer/user Slack:*

https://join.slack.com/t/slack-fyz2512/shared_invite/zt-2ef8n7w9h-3_2kwi_xogrzPdYeF8_Drw