Work on VecGeom Benchmark program

Kyungdon Choi

UFR Sciences et Technologies

MASTER SCIENCES DE LA MATIERE
DEUXIEME ANNEE

SPECIALITE : Physique des Particules

RAPPORT DE STAGE

Work on VecGeom Benchmark program

par

KyungDon CHOI

Responsable de stage : **Federico CARMINATI**

Juin 2015

Index

Abstract

GEANT-4 is s particle-material interaction simulator used in various fields. Now GEANT-V project is ongoing to replace GEANT-4 in the future due to the paradigm change at computer architecture. Core clock already reaches at the threshold of its limit; hence it is hard to handle scalar parameter faster than now on. For this reason GEANT-V use vector parameters can handle faster with certain CPU compilers and CUDA.

VecGeom is the geometry library which will be used at GEANT-V. A benchmark program for VecGeom is to verify if VecGeom is faster than other geometry libraries in ROOT or GEANT-4. XRayBenchmark for VecGeom provides a good way to verify if VecGeom is faster than other geometry libraries in ROOT or GEANT-4. While performing this task I found a bug at DistToCone function in TGeoCone, ROOT6.

1. Introduction

GEANT-V or GEANT-Vectorization is the brand new simulation tool for particle-material interaction which will support fast simulation.[1] This tool contains Compute Unified Device Architecture (CUDA) codes in order to use General-Purpose computing on Graphic Processor Units (GPGPU) for rapid calculation of vectorized parameters.

Vectorization in computer science is the generalization from scalar operators to vector operators, matrices and other high-dimensional arrays which are commonly used in scientific programming. Fortran 90, MATLAB, TK solver, Octave,

[1] GEANT-V official webpage http://geant.cern.ch/content/about-geant5

R, Cilk Plus, and NumPy, a module in Python, are examples of programming languages which support array programming. The single instruction multi data (SIMD) array capabilities, supported since Intel MMX, allow us to calculate vectorized parameters.

CUDA is the GPGPU technology used for GEANT-V. This technology has been developed by NVidia[2] and can be used in various fields. CUDA library helps C, C++ or FORTRAN code directly send to the GPU without using assembly language that the GPU may calculate the code directly. As a GPU has more calculation units than a CPU, it is expected to calculate arrays and matrices faster than a CPU. This describes that a GPU can be more powerful for the arrays or matrices (vectorized) variables.

While GEANT-V allows for fast computing, this program requires a benchmark program. For this reason I worked on a benchmark program for VecGeom which is a Geometry library for GEANT-V. The purpose of the benchmark program is to figure out if GEANT-V is significantly faster than GEANT-4. With this benchmark, GEANT-V team was able to draw an image by applying same algorithm to ROOT6, GEANT-4, and VecGeom and comparing the results. Cylindrical and spherical mapping, added to benchmark program recently, produce segmentation faults with ROOT6, especially DistToCone function in TGeoCone, but these work fine with GEANT-4. The team verified this problem using Valgrind and GNU Debugger (gdb). Reproducible debugging programs will be developed at the end of June 2015.

2. GEANT-4 Applications

[2] http://www.nvidia.com/object/cuda_home_new.html

GEANT-4 is a great tool to simulate particles passing through matters that can be used in various fields. It is first designed for HEP that it can simulate detectors very well. GEANT-4 can also be used for medical, biological (ex, radiation effect to DNA[3]), space and radiation simulation.

A. High Energy Physics (HEP) and Nuclear physics applications

Main purpose of GEANT-4 is to simulate HEP applications such as detectors and beamlines. The projects that use Large Hardon Collider (LHC), located at CERN, are the main users of GEANT-4, but Fermilab and the International Linear Collider (ILC) also use GEANT-4 for their research and applications[4]. The GEANT-4 users in HEP and Nuclear physics are as follow:

I. ATLAS : GEANT4 for the detector simulation
II. CMS : CMSSW and OSCAR experiment
III. LHCb : GAUSS mimics about LHCb experimental conditions and its performance. It has two phases. GAUSS first generate the p-p collisions and then simulate the particle tracks.
IV. ALICE : Particle transportation and detector simulations. This simulation also uses GEANT-3, which is written in FORTRAN, and FLUKA.
V. Fermilab : Various detector simulations such as ArgoNeuT[5], Calorimetry R&D[6], MINERvA[7] and etc.

---

[3] The GEANT4-DNA project http://geant4-dna.org/

[4] GEANT4 HEP application http://geant4.web.cern.ch/geant4/applications/hepapp.shtml

[5] ArgoNeuT webpage http://t962.fnal.gov/

[6] Calorimeter R&D webpage http://drcalorimetry.fnal.gov/

[7] MINERvA webpage
http://nusoft.fnal.gov/minerva/minervadat/software_doxygen/HEAD/MINERVA/index.html

VI. Rare Isotope Science Project (RISP)[8] : RISP which will be built in Deajeon, South Korea is using GEANT-4 for their beamline at RF/IF team.

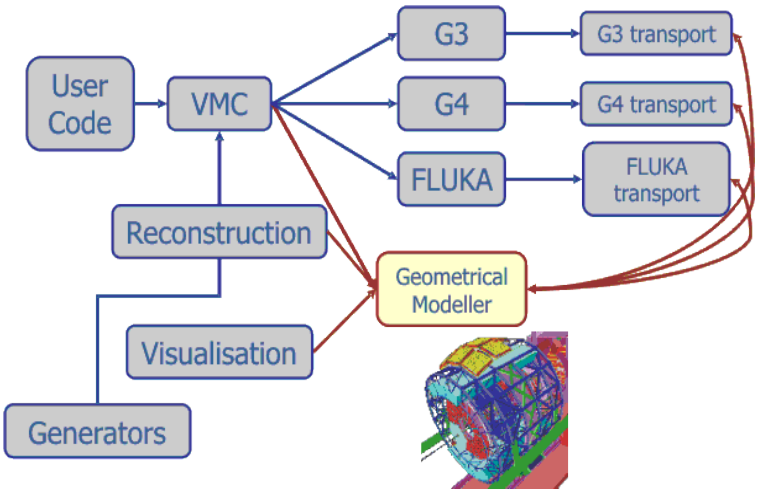

Figure 4 Alice Simulation Flow chart

B. Space and Radiation

The European Space Agency (ESA) is the heaviest user of GEANT-4 in the space field. Its GEANT-4 used projects are XMM-Newton Radiation Environment, Space Environment Information System (SPENVIS), Dose Estimation by Simulation of the ISS Radiation Environment (DESIRE) and Physics Models for Biological Effects of Radiation and Shielding. Other projects such as The Gamma Ray Large Area Space Telescope (GLAST) are also simulated with GEANT-4.

Space is full of space radiation that can damage humans, computer cores, welding points, etc. GEANT-4 is a suitable software to simulate such damages.

C. Medical fields

[8] RISP main webpage http://www.risp.re.kr/eng/pMainPage.do

GEANT4-DNA project is a very interesting project in medical field. This project is modeling early biological damage induced by ionizing radiation at the DNA scale. The goal of Geant-4 based Architecture for Medicine-Oriented Simulation (GAMOS) project is to carry out GEANT-4 based simulation without C++ coding. GATE is a simulation tool based on GEANT-4 that support positron emission tomography (PET), single photon emission computed tomography (SPECT), computed tomography (CT) and Radiotherapy experiments.

D. Other applications

After the Fukushima reactor accident, alternative energy sources are a hot issue of research. One of the solutions is thorium reactor. Thorium reactors have some benefits compared to uranium or plutonium based reactors. The expected nuclear waste produced is less than 1/1000 of ordinary reactors and the nuclear waste cannot be used to make nuclear weapons. Also thorium is not the material that do chain reaction such that if there is an accident like Fukushima reactor accident, a thorium reactor will automatically turn off instead of creating a meltdown. GEANT-4 is used in this field to simulate the lifecycle of thorium[9].

3. GEANT-V and Computer architecture

GEANT-V is the next generation of particle simulator including vectorization, GPGPU and other advanced technologies developed since GEANT-4 was published. Also, CPUs design has changed since then. Figure 1 shows that the numbers of transistors are increasing but because of the physical limitations,

[9] GEANT4 STUDIES OF THE THORIUM FUEL CYCLE Proceedings of 2011 Particle Accelerator Conference, New York, NY, USA

clock and power stay still. This is because as time passes, transistors are getting smaller such that more can place in a limited area. However, the amount of heat they produce depends on the amount of power that transistors consume. So if power usage of transistor or more efficient cooling method can found, the clock cannot be increased. This is why it is hard to expect clocks more than 4GHz for x86 architecture. The relation between CPU power and input voltage is as follows[10]:

$$P_{dynamic} = \alpha C V_{DD}^2 \mathrm{fA}$$

$P_{dynamic}$ is the total power applied to the CPU, C and A are transistor density factors, f is the operating frequency, and V_{DD} is the power from the power supplier or mother board. If a processor has extremely low input voltage, it can be run in extremely high frequency. However, from the relation above, it can be concluded that there exists a certain threshold to increase frequency due to the limited of input power. This explains why CPUs do not have high operating frequency despite adding more cores.

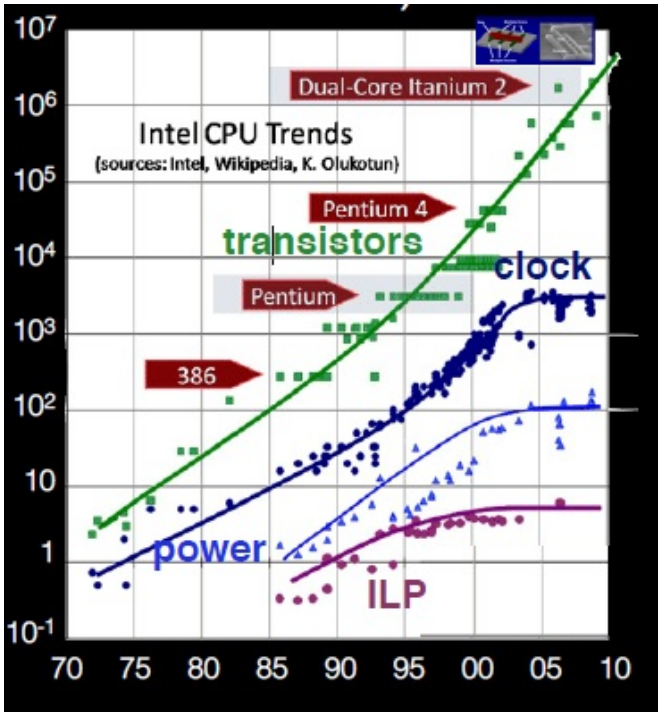

Figure 1 CPU development history

Runtime of scalar parameters highly depend on CPUs clock. Unfortunately modern CPUs already reached at the clock

[10] Power Consumption Analysis http://forums.anandtech.com/showthread.php?t=2281195

threshold. GEANT-V, which is designed to handle vectorized parameters, is expected to break through the limit of GEANT-4 and can benefit various fields' mentions in chapter 2.

4. Important software

A. Compute Unified Device Architecture (CUDA)

CUDA is a parallel computing platform and application programming interface (API) model developed by NVidia, producing well known producer of graphic processor units (GPU) and system on a chip units (SOC), located in Santa Clara, California. This API provides a software layer, which allows direct access to GPU's virtual instruction set with programming languages like C, C++, Fortran, etc.

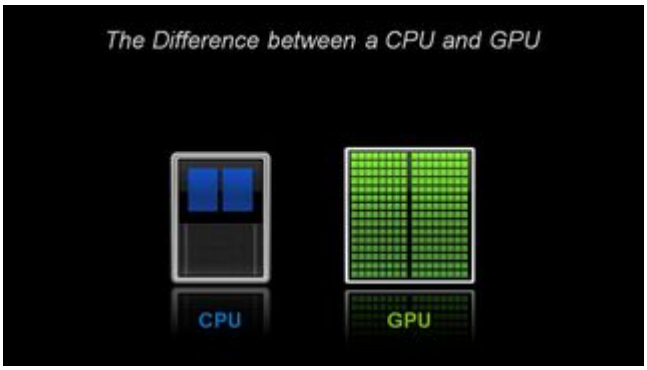

Figure 2 CPU and GPU

Requirement of CUDA originate from GPUs and CPUs architecture difference. The major difference between CPUs and GPUs is the numbers of cores. CPUs used to have 1 to 24 cores (with possibility for more) but GPUs (few hundreds MHz order in order of magnitude) have more than thousands of cores that have less instruction sets and are slower compare to

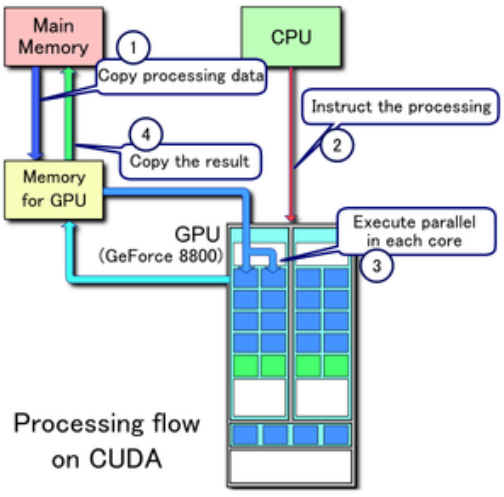

Figure 3 CUDA process flow

a CPU (GHz order) core. CPUs can calculate linear functions fast enough but these are not efficient for array calculation like drawing an image on the screen. In comparison GPUs are not good enough for recursive processes but are good enough to draw an image at the screen with 60Hz frequency. For example, HD display means 1920 pixels in a row and 1080 pixels in a column (2,073,600 pixels) that must be calculated in 1/60 of a second. This implies that GPUs are capable of calculating huge matrices at once.

CUDA works as flowing processing flows.[11] CUDA code copies data from main memory to GPU memory. Then the CPU instructs the process to the GPU. After receiving this instruction, the GPU will execute its cores in parallel. Finally, it will copy these results from the GPU memory to the main memory. However this doesn't work with all kinds of GPUs; only a few select NVIDIA GPUs are viable for CUDA.[12]

Advantages and disadvantages of CUDA are as follows:

Advantages

I. Scattered reads : code can read from arbitrary addresses in memory
II. Shared memory : CUDA exposes a fast shared memory region that can be shared amongst threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups.
III. Faster downloads and readbacks to and from the GPU
IV. Full support for integer and bitwise operations, including integer texture lookups

[11] CUDA Overview by Cliff Woolley, NVIDIA, page 5 to 7

[12] CUDA developer page https://developer.nvidia.com/cuda-gpus

Disadvantages

I. CUDA does not support the full C standard.
II. Copying between host and device memory may incur a performance hit due to limitations in system bus bandwidth and latency
III. Unlike OpenCL, CUDA-enabled GPUs are only available from some of NVIDIA GPU lineups
IV. CUDA (with computing capability 2.x) allows a subset of C++ class functionality. For example member functions may not be virtual.[13]
V. Valid C/C++ may sometimes be flagged and prevent compilation due to the optimization techniques the compiler is required to employ in order to use limited resources.

With those characteristics CUDA has a lot of applications in various fields ranging, from video games to academic research. The well-known science project SETI[14] is using a CUDA client program to analyze data. In HEP, CUDA is used to analyze lattice QCD.[15]

B. Jenkins

Jenkins[16] is an open source continuous integration tool written in Java. It was first called Hudson but after a dispute with Oracle in 2010 changed name to Jenkins. Jenkins is a freeware released under the MIT license.[17] Stable version was release in June 2015.

---

[13] UDA C Programming Guide 3.1 – Appendix D.6

[14] CUDA application list http://www.geforce.com/games-applications/pc-applications/setihome

[15] Generating SU(Nc) pure gauge lattice QCD configurations on GPUs with CUDA

[16] Jenkins Official page http://jenkins-ci.org/

[17] Open Source MIT License http://opensource.org/licenses/MIT

The GEANT-V team used Jenkins as a testing platform, mostly used for nightly test because, computing resource is limited during day time when, people use computing power for development of GEANT-V and VecGeom.

C. JIRA

Jira is an agile software development tool which provides bug tracking, issue tracking and project management functions. It is written in Java and is also a platform independent software. It integrates well with source control programs such as CVS, Git, Clearcase, etc. CERN officially provides JIRA, and the GEANT-V team uses Jira as a project manager tool to control the project efficiently.

5. Work on VecGeom Benchmark program

VecGeom is the new geometry libraries which will be used for GEANT-V. This library consists of fully vectorized parameters but not yet proven to be faster than scalar parameters. For this reason, building and maintaining the benchmark program is important for developing VecGeom.

A. Installation and required programs

To compare runtime with existing simulators, ROOT6, GEANT-4 and GEANT-V are obvious choices but there are other software required too[18]. GCC version should be more than 4.8.x (Current version is 5.1.1). It is good to use Devtoolset2.0 or higher provided at CERN webpage[19]. List of programs in Devtoolset2.0 provides

[18] GEANT-V installation Guide http://geant.cern.ch/content/installation

[19] Linux at CERN http://linux.web.cern.ch/linux/devtoolset/

tools as follow:

I. gcc/g++/gfortran : GNU Complier Collection (Version 4.8.1)
II. gdb : GNU Debugger (Version 7.6.34)
III. valgrind : Tool for finding memory management bugs in programs (Version 3.8.1)

These are main programs used to code benchmark program, though there are programs like binutils, elfutils, dwz, systemtap, oprofile and eclipse.

One of the goals for GEANT-V is fast simulation, to a degree that VecGeom would run faster than using other simulation programs such as GEANT-4. To perform this task benchmark program requires many programs. ROOT6 and GEANT-4 are main competitors for comparing the runtime. Also vectorization support libraries are required to run vectorized parameters with compilers. Following programs are required to use the benchmark program:

I. ROOT6 : The only capable version for ROOT is ROOT6 and tag v6-03-02.
II. GEANT-4 : Use the most resent version of GEANT-4.[20]
III. VC : SIMD library for C++. Version should be higher than 2.20 (Current version is 2.24)
IV. VecGeom : Geometry libraries for GEANT-V
V. PYTHIA8 : Current version of PYTHIA is 8.200 but it is not supported by ROOT. The version supported by LHC is PYTHIA 8.186.
VI. HepMC : An object oriented event record written in C++ for high

[20] GEANT-4 download page http://geant4.web.cern.ch/geant4/support/download.shtml

energy physics monte carlo generators.[21]

VC is the main key of Vectorization which is used to ease explicit vectorization of C++ code. It contains an intuitive API and provides portability between different compilers and compiler versions as well as portability between different vector instruction sets. This can be compiled for:

I. Advanced Vector Extensions (AVX): extensions to the x86 instruction set architecture for microprocessors from Intel and AMD. It was applied from Q1, 2011 with Intel Sandy Bridge and Q3, 2011 with AMD Bulldozer processor.
II. SSE2 up to SSE4.2 or SSE4a: Streaming SIMD Extensions 2 (SSE2) is one of the Intel SIMD processor supplementary instruction set first applied at Pentium 4 in 2001. It is applied to Athlon 64 with AMD in 2003.

B. Benchmark program

The major purpose of this benchmark program is to compare the runtime of GEANT-4, ROOT, and VecGeom. The function must be coded to calculate with GEANT-4, ROOT, and VecGeom, kept consistent algorithm for each simulation tools, and separated not to affect each other. The main flowchart of this benchmark program is as follows:

I. Read detector geometry from CMS2015.root which is the CMS detector for LHC run 2.

[21] M. Dobbs and J.B. Hansen, Comput. Phys. Commun. 134 (2001) 41.

II. Exception handling with parameters. At the beginning this benchmark had 5 parameters and now there are 6 parameters with scale factor of image.
III. Read the direction of X-ray and set the direction and scale of the output image.
IV. Load each functions written in ROOT6, GEANT-4 and VecGeom to measure the runtime of each functions and to draw the section of the part of detector.
V. In the end, draw the image of part and print colors depend on the X-ray decay length.

To verify if the benchmark program is working properly, this program prints out the volume image that displays total numbers of volumes passed through. Figure 5 shows the image in z direction of the calorimeter inside the CMS 2015 detector. This image is not renormalized and it is hard to see the clear detector image. The background is not perfectly black because void was counted as 1 volume. Even though it is not describing the detector shape clearly there is a remarkable observation in this image. It is clear to see that the center has no volume, which implies that the beamline part is empty. In order to describe the clear image of calorimeter, benchmark code was changed to use the total mass passed through. Figure 6 shows the

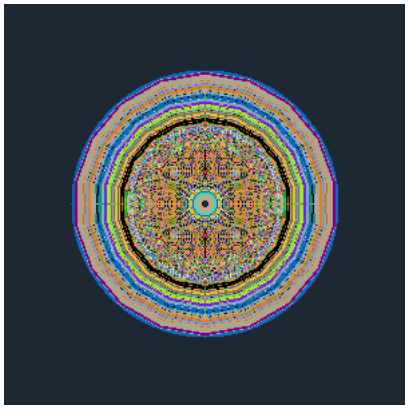

Figure 5 Detector Section in z axis by the passed volume.

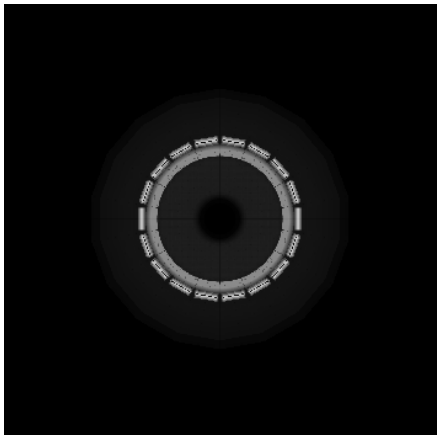

Figure 6 Monotone image of Calorimeter in Z axis

monotone image of calorimeter on z axis with the total mass in the pixel. The color of a pixel is close to white if a pixel has heavier mass and it is close to black if a pixel has 0 mass. This color mapping images are shown at Figure 7, where left one is on z axis and right one is on x axis. However colored images do not describe the structure of this calorimeter clearly.

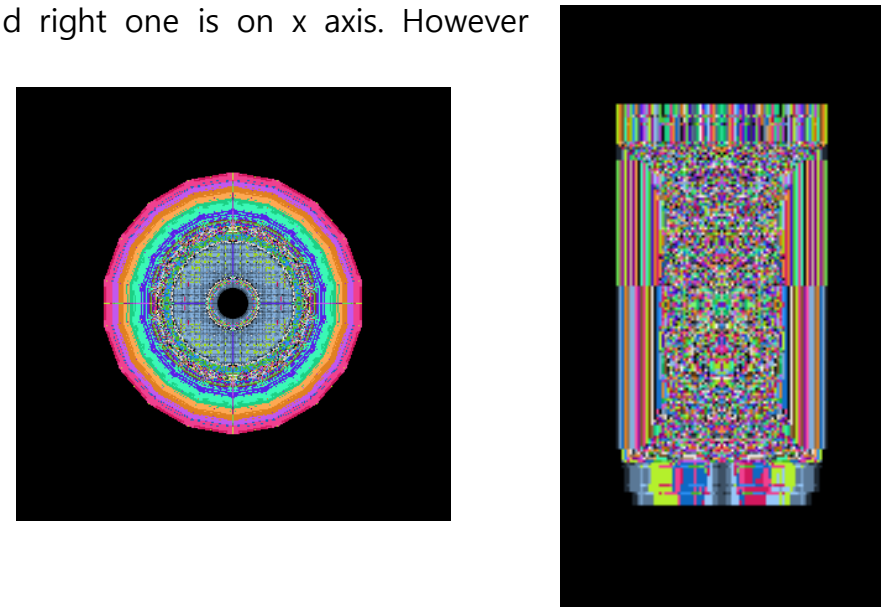

Figure 7 Color images of renormalized data

This problem was handled with log scale including ln, log2 and log10. Selecting scale is parametrized with the input parameter while running benchmark program. Compared to the Figure 7, the Figure 8 describe the calorimeter structure better.

After log scale mapping was done, new coordinate algorithm was required, which is a spherical mapping. The difference between spherical mapping and ordinary mapping is the change of direction. The X-Ray direction does not change while running with Cartesian

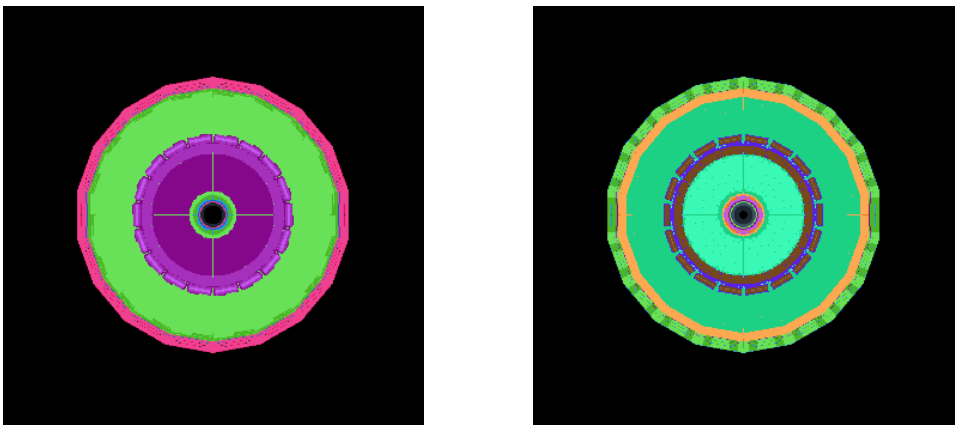

Figure 8 Log 10 and Log 2 image

coordinates, but the direction changes continuously while running with spherical mapping from the center. The notation is as follows:

$$\text{theat} = \pi\,((\text{total run number})/(\text{maximum run number} - 1))$$
$$\text{phi} = 2\pi\,((\text{total run number})/(\text{maximum run number} - 1))$$
$$\text{xdir} = \sin(theta)\cos(phi)$$
$$\text{ydir} = \sin(theta)\sin(phi)$$
$$\text{zdir} = \cos(theta)$$

Also, cylindrical mapping is similar to spherical mapping code but one thing is different; cylindrical mapping has 2 dimensional vectors instead of 3. The code for cylindrical mapping is as follows:

$$\text{phi} = 2\pi\,((\text{total run number})/(\text{maximum run number} - 1))$$
$$\text{xdir} = \cos(phi)$$
$$\text{ydir} = \sin(phi)$$
$$\text{zdir} = 0$$

Testing the benchmark program with ROOT6 spherical and cylindrical mapping crashed with TGeoCone in ROOT6. To test if the mapping code was working properly, spherical mapping was tested with simple box and worked fine. Additionally, another test code was written to figure out which volume crashes with mapping code. According to the test code, name of the crashing volumes were EEIMod6080, EEDee7a80, EEiEnvScr6b80 and EEBackQuad5e00. After running the test codes, segmentation fault was precisely

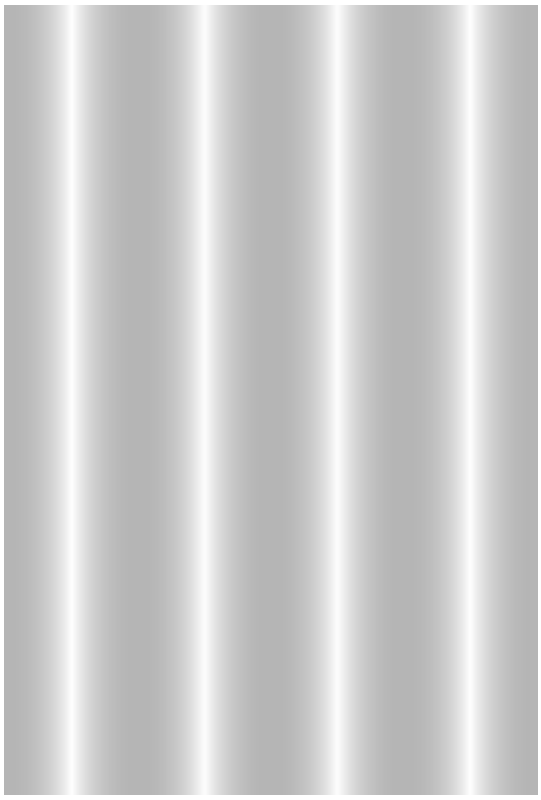

Figure 9 HB image with cylindrical mapping

described with Valgrind and gdb. As the result it is revealed that TGeoCone, which is a function of ROOT6, was making segmentation fault. Figure 9 shows the cylindrical mapping with a box shape due to the segmentation error with calorimeter.

To certify if this is only for ROOT6, mapping code was operated with GEANT-4. With the success of GEANT-4 code test, final objective is to make a reproducible debugging code for DistToCone in TGeoCone functions.

6. Conclusions

GEANT-4 is a very powerful simulation tool for particle – material interaction, but it had been published long before modern technologies were developed. GEANT-V is trying to apply those technologies, mostly vectorization and more suitable programs for modern computer architectures. Modern CPU compliers and GPGPU allows performing rapid calculation of vectorized parameters.

VecGeom, the Vectorized geometry library, is in developing process requiring benchmark program to certify the speed and quality of its functions. I participated in this job and made some graphical benchmark codes. One remarkable achievement while writing benchmark codes was accidentally figuring out a bug at ROOT6 in TGeoCone function. A reproducible debugging program is under development and is expect to committed until the end of the internship.