zfp Compression Project Proposal

Erik Wallin

June 8, 2020

This is a proposal for the IRIS-HEP fellowship project "Exploring Floating Point Compression of HEP data with zfp", from Erik Wallin at Lund University, Sweden. The project will be done here in Lund, with access to (possibly distributed) CPU resources, as well as some limited GPU processing possibilities.

Data compression is necessary in high energy physics, because of the enormous amounts of data involved. ROOT implements several lossless compression algorithms (eg. ZLIB and LZMA), but the choices available for lossy compression of floats are few, especially for non-time series or non-image data. Float truncation by cutting of the mantissa is quick and simple, but there is a lot of exploration to be done for more *intelligent* compression methods. The difficulties of lossy floating point compression are clear in HEP, as one often work with high-entropic data that is difficult to compress.

The project will explore use of the zfp float compression algorithm for CMS MiniAOD data. The project will write a system for applying zfp compression on the data, as well as comparing it to other lossy (float truncation) and lossless (those present in ROOT) compression techniques. The metrics of interest are: The compression ratio, the residuals (percentual changes in the data), the CPU usage, memory usage and possibly GPU usage. zfp is also interesting for its efficient real-time usage, that could run on GPUs and even on FPGAs in the detector trigger for example.

The final product is an open-source git repository including the scripts (and documentation) used for the analysis and to reconstruct my results (useful for non-CMS users), as well as a prototype MiniAOD variant compressed with zfp.

Here is a 12 week schedule:

- **Week 1 - 3:** Familiarisation with the EDM to be used and how to read and write them. Get zfp working on at least a test sample, with the Python or C library. The largest hurdle in such a step could be getting

familiar with the CMS offline code, if that is needed to write MiniAODs correctly.

- **Week 4 - 5:** Start applying the compression on actual data, starting with analysing the compression errors. Analysis of CPU and memory usage of the compression and decompression of the MiniAOD. It should be tried on experimental data as well as MC data, to see that it works the same for both.

- **Week 6:** As boilerplate code for compressing and decompressing MiniAODs will be done at this point, cleaning it up and writing documentation for it will be done as a half-time checkpoint. A short presentation could also be done during this step.

- **Week 7 - 8:** Comparing the zfp compression with the float truncation compression, and later the lossless compression methods. An idea worth investigating is whether the lossless compression methods lose their efficiency if the data is zfp compressed before. Chaining compression techniques should be carefully looked at. Even the float truncation and zfp compression could be done on the same data.

- **Week 9 - 10:** As a buffer period, finish up lose threads from before. If there is time, play around with other ideas:

  - Analysing zfp on the GPU.
  - Exploring the information theoretic background to lossy compression and zfp to see if there is anything practical to learn from there.
  - Compare zfp versus fpzip (either lossless or lossy fpzip)
  - Test on MC data with some new physics signal sample, to see how it reacts.

- **Week 10 - 12:** Polish and refactor the code for a release that is easily usable by others. Write a presentation and make sure the code is documented well enough.