Presentations:

- Introduction:
    - Welcome to our two new conveners:
        - **Marc Paterno** and **Mircho Rodozov**
    - A big thank you to our outgoing conveners:
        - **Ben Morgan** and **Martin Ritter**
    - Please let us know topics/discussion you'd like to see covered
        - Needless to say we'll be more than happy to hear from you

- Linux Systems Performance: Tracing, Profiling, and Visualization:
    - Dr Guilherme Amadio gave a very nice presentation on the subject
    - We've primarily covered *perf*, having a few case studies based on G4
    - We've agreed to discuss the second half of the talk in another session
        - Starting from Memory Access and Latency Analysis

Discussion points:

- Some discussion over the examples in slides 69 and 71. Vectorization was brought up by Patrick and asked if it was tried. Guilherme said yes, but proved to be challenging (due to G4Log usage).
- Hadrien raised the point that getting the frame-pointer reliably is challenging (see chat below). In *perf*, *--call-graph=dwarf* is a good option but has compromises.
- Stephen mentioned llvm-mca and Intel ACA (see chat below). Guilherme said he used these in the past, they're indeed very useful in analyzing especially small code snippets.

Chat:

Here is all the discussion on the chat (in chronological order) to preserve the information. The main questions/discussions were largely based on these as well:

Hadrien Grasland:
To rule 4 I'd add that even if you do eventually need the fancy algorithm, the dumb algorithm can see some use in your validation test suite. So you'll likely need to implement it anyway.

Stephen Swatman:
(you can also pass -e a comma-separated list of events and it will record each of them in one go)

Serhan Mete:
There are a few other implementation for the FlameGraph diff here for those who are interested: http://www.brendangregg.com/blog/2014-11-09/differential-flame-graphs.html

Hadrien Grasland:
Yeah, frame pointer is really hard to get reliably (not just in the application, but in every library that it uses) unless you basically rebuild your entire Linux distro. Which is not exactly unheard of in the HEP community, though.

Another trick to use with --call-graph=dwarf is to put the output file on /dev/shm. But this limits you to files no bigger than your system RAM. Those files will already take a lot of time to process though, so whether you really want more data is an open question.

(well, half your system RAM with the default /dev/shm setup, to be precise.)

You can also reduce the stack unwinding failure rate (but not to zero, unfortunately) by recording bigger stack samples with e.g. "--call-graph=dwarf,64000". Unfortunately, profiling and storage overhead scale linearly with the stack sample size, and sampling more than 64KB is forbidden by perf.

Stephen Swatman:
https://llvm.org/docs/CommandGuide/llvm-mca.html

https://software.intel.com/content/www/us/en/develop/articles/intel-architecture-code-analyzer.html