In “8 Steps to 3.7 TFLOP/s on NVIDIA V100 GPU: Roofline Analysis and Other Tricks” we see the steps that Yang and their team took in order to optimize the General Plasmon Pole in order to bring the peak TFLOP/s from 2.337 to 3.710 in the Si-214 test and from 2.216 TFLOP/s to 3.638 in te SI-510 test. From the paper we learn that we are able to improve the performance of a parallelized program through optimizing it. The optimization of this specific kernel had to do with understanding what the code was individually understanding the relationship between register usage, SM usage, memory bandwidth usage, memory access pattern for different arrays, instruction latency and arithmetic intensity. Many of the steps along the way in optimizing this kernel came from replacing operations that would use floating point operations. Because floating point operations take more instruction cycles in order to complete than integer division and FMA, if we are able to replace these instructions with others which utilize integer or FMA we can increase our total FLOPS.

One of the other steps that they took in order to optimize the GPP kernel was to reduce the amount of unnecessary branching. While in this step they were unable to achieve better performance they generally recommend it to avoid divergence within a wrap. This step helps with the further methods in order to achieve better performance. The other types of performance improvements they used was through ensuring that memory was being reused as often as possible and doesn't need to be rewritten.

I think the most important thing that there was to learn from this was that there are 3 total forms in how to achieve better performance, while a single step applied may make the overall performance worse in conjunction with each other they can provide better performance.

1. Reduce operations which take a relatively large amount of operations that can be done in fewer cycles. We can see this in them replacing Divide instructions, and the Abs() function with non Floating point operations.
2. Minimize that amount of unnecessary operations/branches being done.
3. Modify the storage of Data in order to maximize reuse, and localize like information together.

I think that this paper was very interesting and showed that there are a lot of different steps that can be taken in order to improve the overall performance of a kernel. This comes through understanding what type of arithmetic operations are being done, and understanding the dataflow and pattern. While the paper teaches a lot, I found some things like GPP was not really explained as to what it is and what SI-256- Si-512 tests where. If i were to improve this paper i would like to include a small section providing a general overview of what GPP and the BerkeleyGW code is.