Best Practices for Benchmarking Diverse Architectures with Varied Workloads

Taking the Software Developer Perspective
November 17, 2021

Hartwig Anzt
Steinbuch Centre for Computing (SCC)

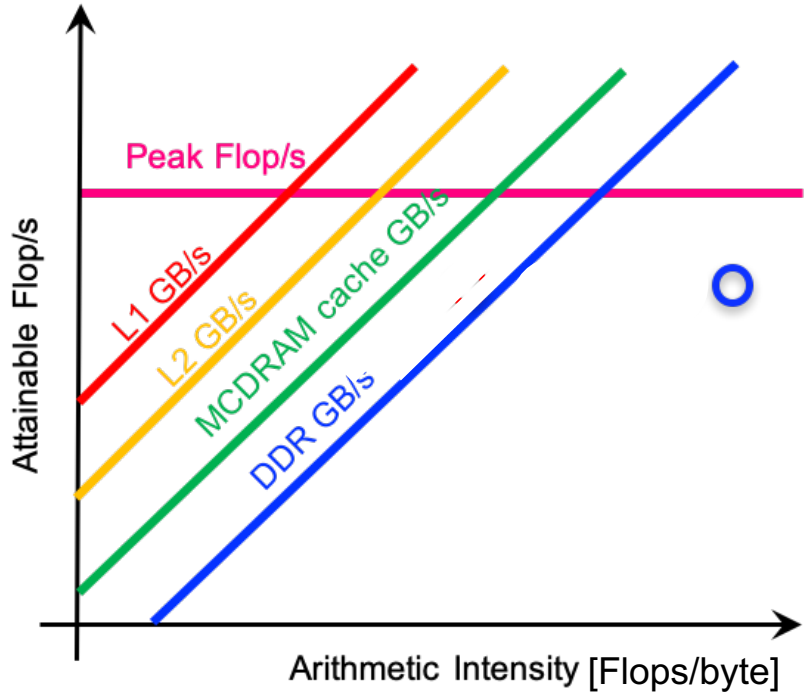
Challenge I: Finding meaningful benchmarks and understanding performance

Why does the code run at 2 TFLOP/s, the hardware has 188 TFLOP/s double precision performance?

Challenge I: Finding meaningful benchmarks and understanding performance

Why does the code run at 2 TFLOP/s, the hardware has 188 TFLOP/s double precision performance?

- We need to take into account what limits the performance on a certain hardware.
- The Roofline Model is a useful tool to understand what limits the performance of an application.

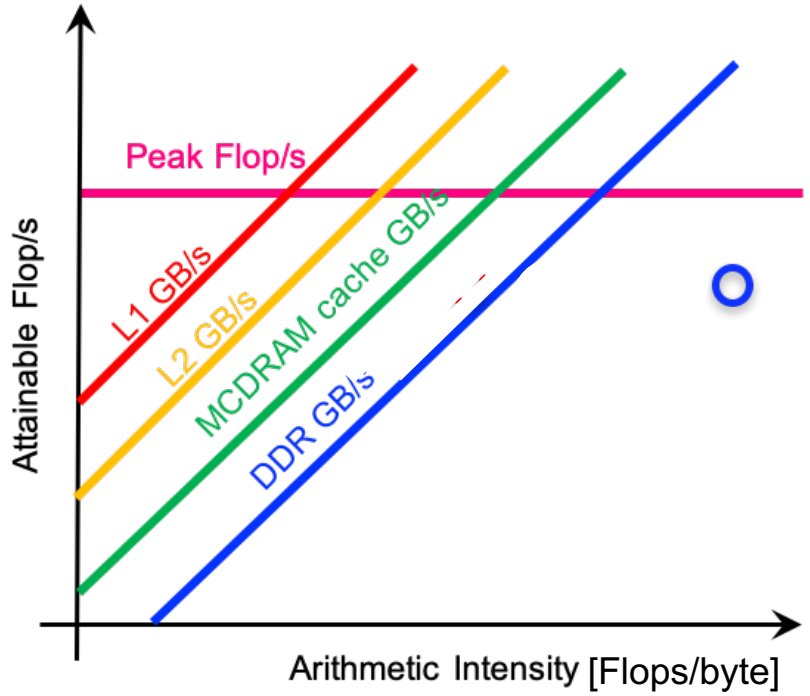


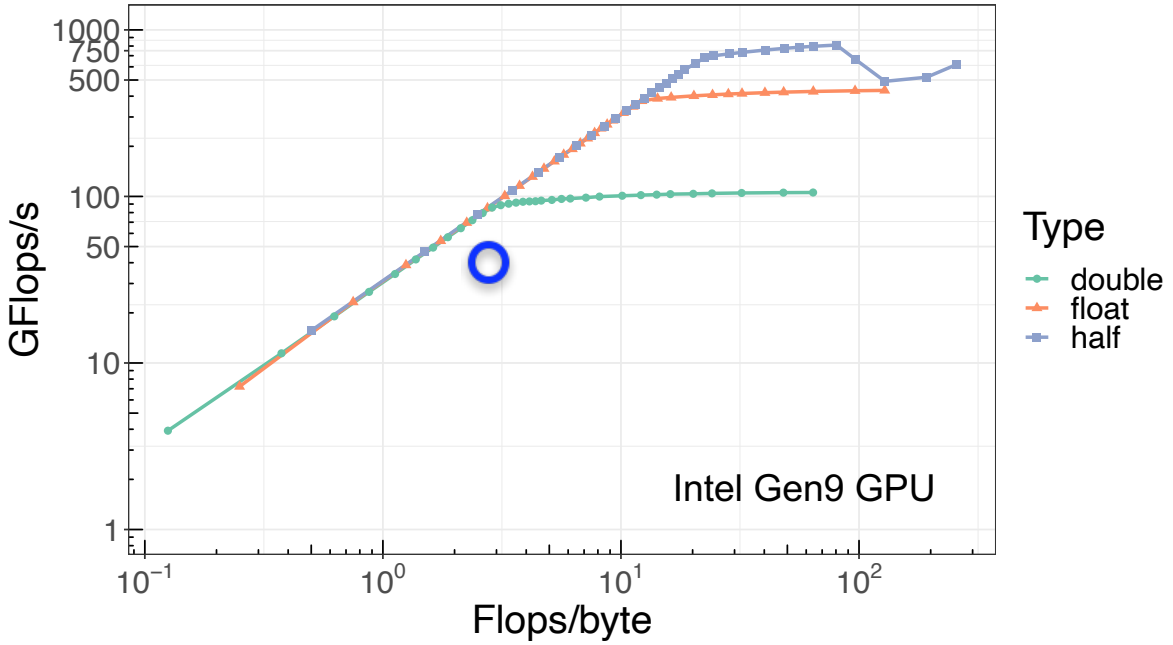
https://docs.nersc.gov/tools/performance/roofline/

Challenge I: Finding meaningful benchmarks and understanding performance

Why does the code run at 2 TFLOP/s, the hardware has 188 TFLOP/s double precision performance?

- We need to take into account what limits the performance on a certain hardware.
- The Roofline Model is a useful tool to understand what limits the performance of an application.
- The experimental Roofline helps evaluating whether the hardware delivers what it promises.





https://docs.nersc.gov/tools/performance/roofline/

Challenge II: Making performance results reproducible

Why do I not get the same performance like the authors of the paper?

Challenge II: Making performance results reproducible

Why do I not get the same performance like the authors of the paper?

- Benchmark results are often presented without sufficient environment details.
- Hardware configuration, OS, driver version, software version, build options, and benchmark parameters all play an important role.
- Security updates can make a previous-achieved benchmark result irreproducible.
- We should make the reproducibility of benchmark results mandatory, e.g., submission of benchmark executable and all metadata a requirement.
- *Open questions: Licensing? Open data?*

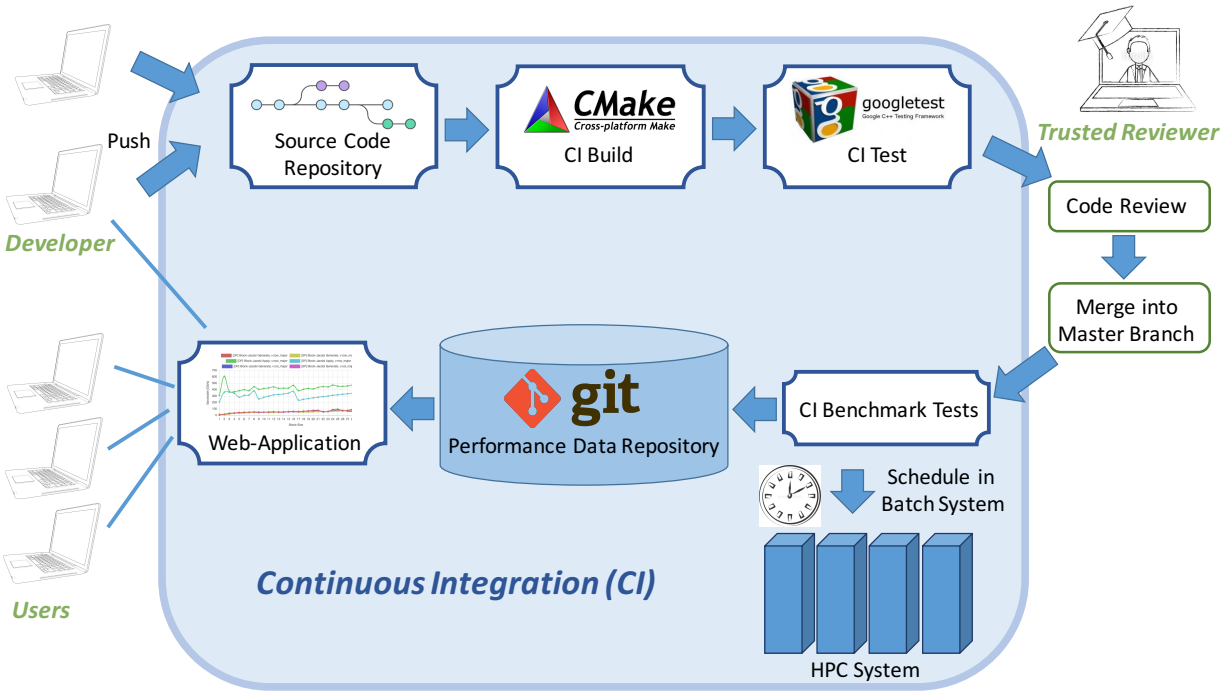
Challenge III: Continuous Benchmarking

How can we be sure that the latest change in the code does not degrade the performance of the application?

Challenge III: Continuous Benchmarking

How can we be sure that the latest change in the code does not degrade the performance of the application?

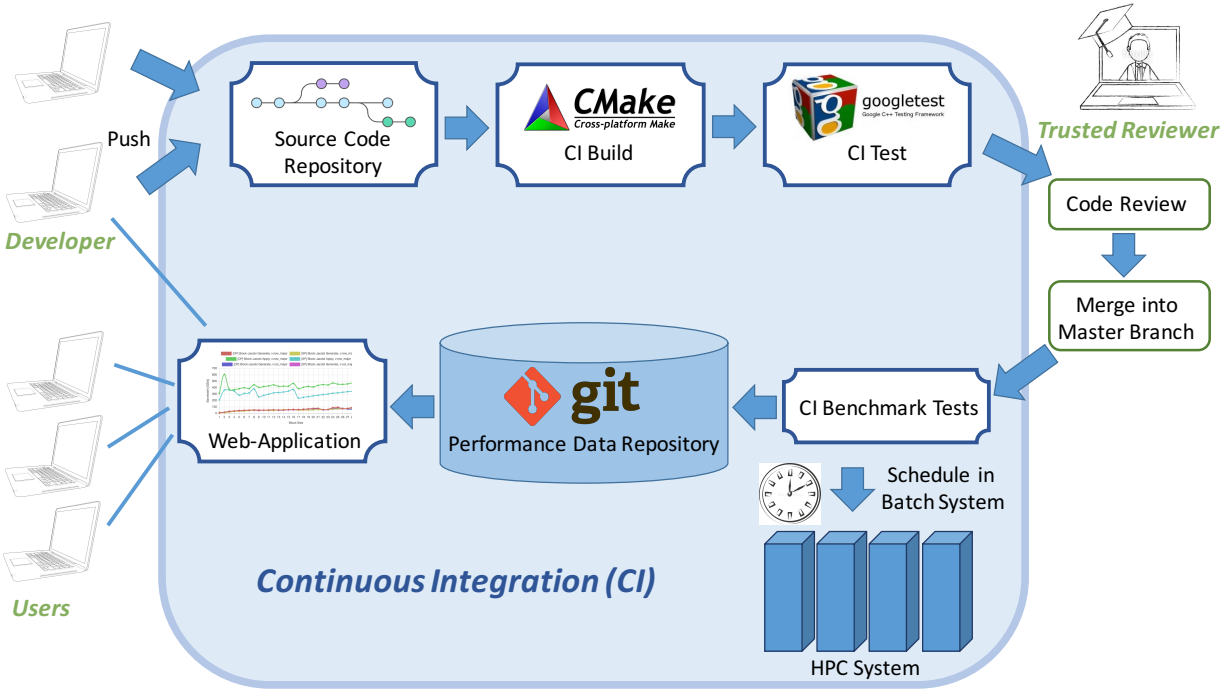
- We need to continuously benchmark relevant code on the target machines (Continuous Benchmarking).
- Comparing with archived performance data allows to identify performance degradation caused by code changes or software environment updates (driver version, security updates, etc.).
- Interactive visualization of public performance results over time allows users to identify performance peaks / corner cases.



Challenge III: Continuous Benchmarking

How can we be sure that the latest change in the code does not degrade the performance of the application?

- We need to continuously benchmark relevant code on the target machines (Continuous Benchmarking).
- Comparing with archived performance data allows to identify performance degradation caused by code changes or software environment updates (driver version, security updates, etc.).
- Interactive visualization of public performance results over time allows users to identify performance peaks / corner cases.



- *Continuous Benchmarking needs valuable compute cycles on leadership systems*
  - who pays for that?
- *This requires an automated framework to run jobs on HPC systems*
  - what about security, 2FA?