Advanced Methods for Scientific Computing (AMSC)
Lecture title: Using Google Benchmark for Benchmarking C++ Codes

Luca Formaggia

MOX
Dipartimento di Matematica
Politecnico di Milano

A.Y. 2024/2025

Introduction

Google Benchmark is a library designed to help developers measure and optimize the performance of their C++ code. It provides a simple API to benchmark code snippets, enabling developers to identify bottlenecks and make informed improvements.

Installation

1. Clone the Repository:

```
git clone https://github.com/google/benchmark.git
cd benchmark
```

2. Build and Install:

```
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make
sudo make install
```

3. Link Against Google Benchmark:

```
g++ my_benchmark.cpp -lbenchmark -lpthread -o my_benchmark
```

Writing a Benchmark

1. Include the Benchmark Header:

```
#include <benchmark/benchmark.h>
```

2. Define a Benchmark Function:

```
static void BM_StringCreation(benchmark::State& state) {
  for (auto _ : state) {
    std::string empty_string;
  }
}
BENCHMARK(BM_StringCreation);
```

3. Run the Benchmark:

```
int main(int argc, char** argv) {
  benchmark::Initialize(&argc, argv);
  benchmark::RunSpecifiedBenchmarks();
}
```

Example Benchmark

Here's a complete example demonstrating how to benchmark the creation of a `std::vector`:

```
#include <benchmark/benchmark.h>
#include <vector>

// Function to benchmark
static void BM_VectorCreation(benchmark::State& state) {
  for (auto _ : state) {
    std::vector<int> v;
    v.reserve(state.range(0));
  }
}

// Register the function as a benchmark
BENCHMARK(BM_VectorCreation)->Arg(1024)->Arg(2048)->Arg(4096);

// Main function
int main(int argc, char** argv) {
  benchmark::Initialize(&argc, argv);
  benchmark::RunSpecifiedBenchmarks();
}
```

Interpreting Results

When you run the benchmark executable, it outputs results like:

```
-----------------------------------------------------------
Benchmark                    Time             CPU Iterations
-----------------------------------------------------------
BM_VectorCreation/1024      5 ns            5 ns   100000000
BM_VectorCreation/2048      6 ns            6 ns   100000000
BM_VectorCreation/4096      7 ns            7 ns   100000000
```

Key Metrics

- **Time:** Wall time taken for the operation.
- **CPU:** CPU time consumed.
- **Iterations:** Number of iterations run to achieve statistically significant results.

Advanced Features

- Custom Time Units: Specify time units (e.g., milliseconds) for better readability.
- Complex Arguments: Use `Args({a, b, c})` to pass multiple arguments.
- Fixture Benchmarks: Use fixtures to set up and tear down benchmarks.
- Custom Counters: Track additional metrics, such as memory usage or cache hits.

Conclusion

Google Benchmark is a powerful tool for measuring the performance of C++ code. By providing a simple API and robust features, it helps developers identify performance bottlenecks and optimize their applications effectively.