Techniques for Optimisation

1. Identifying work that can be removed

- The simplest form of optimisation is not doing some work

- Use a profile trace to identify sections of code that don’t need to be run

- E.g. reducing the number of memory allocations by reserving

- You may need to think of an alternative approach in order to remove a chunk of code

- Evaluate lazily

- Evaluate asynchronously

43

Techniques for Optimisation

1. Identifying work that can be removed

- The simplest form of optimisation is not doing some work
- Use a profile trace to identify sections of code that don’t need to be run
  - E.g. reducing the number of memory allocations by reserving
- You may need to think of an alternative approach in order to remove a chunk of code
  - Evaluate lazily
  - Evaluate asynchronously

```cpp
#include <vector>

static void Standard(benchmark::State& state)
{
  for (auto _ : state)
  {
    std::vector<double> vec;

    for (int i = 0; i < 1'000; ++i)
      vec.push_back (0.0);
  }
}
BENCHMARK(Standard);

static void Reserve(benchmark::State& state)
{
  for (auto _ : state)
  {
    std::vector<double> vec;
    vec.reserve (1'000);

    for (int i = 0; i < 1'000; ++i)
      vec.push_back (0.0);
  }
}
BENCHMARK(Reserve);

static void Init(benchmark::State& state)
{
  for (auto _ : state)
  {
    std::vector<double> vec (1'000, 0.0);
    benchmark::DoNotOptimize (vec);
  }
}
BENCHMARK(Init);
```

https://quick-bench.com/q/9uTY-mY6yFS1gIE7hIDBFjU5uNw