TECHNISCHE UNIVERSITÄT BERLIN

ElastiBench: Scalable Continuous Benchmarking on Cloud FaaS Platforms

Trever Schirmer, Tobias Pfandzelter, David Bermbach | WOSCx3

3S

Microbenchmarks

```
func BenchmarkFib20(b *testing.B) {
    for n := 0; n < b.N; n++ {
        Fib(20)
    }
}
```

Unit-Test sized

Increase N until function takes one* second

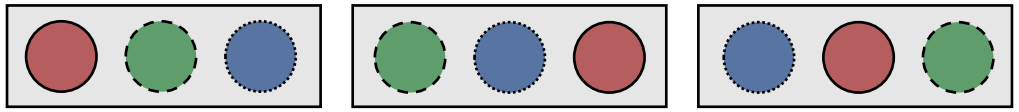
```
$ go test -bench=. ./examples/fib/
goos: Darwin
goarch: amd64
pkg: fib
BenchmarkFib20-8 28947 40617 ns/op
PASS
ok fib 1.602s
```

#CPU cores

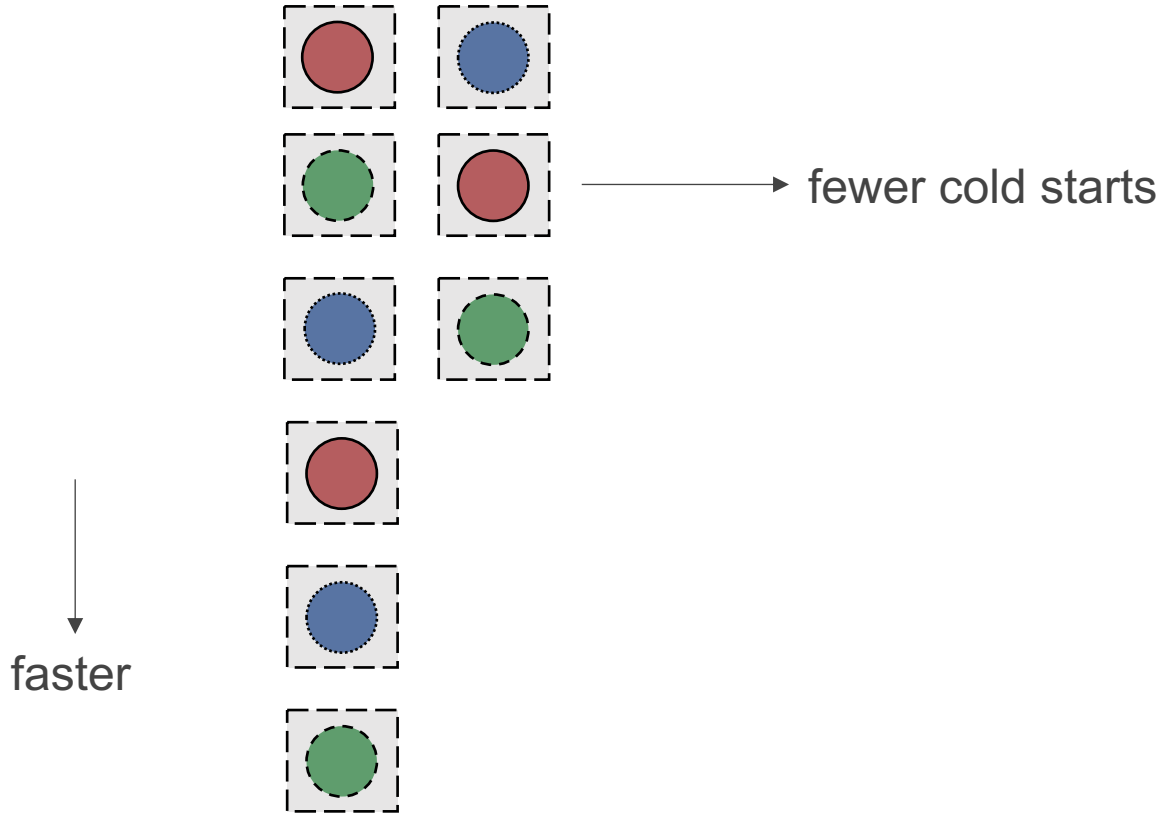
Final `b.N` from above

https://dave.cheney.net/high-performance-go-workshop/gophercon-2019.html

Cloud Microbenchmarks



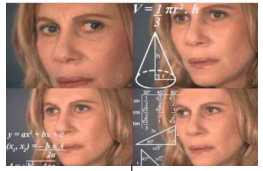
FaaS Microbenchmarks



Evaluation

- Two SUT Versions
- Go Toolchain
- Prepopulated Build Cache
- Instance Build Cache
- λ Caller

Evaluation: Compare to VM-based Microbenchmarks from our previous work.



ElastiBench: Scalable Continuous Benchmarking on Cloud FaaS Platforms

Trever Schirmer
TU Berlin & ECDF
Berlin, Germany
ts@3s.tu-berlin.de

Tobias Pfandzelter
TU Berlin & ECDF
Berlin, Germany
tp@3s.tu-berlin.de

David Bermbach
TU Berlin & ECDF
Berlin, Germany
db@3s.tu-berlin.de

ABSTRACT

Running microbenchmark suites often and early in the development process enables developers to identify performance issues in their application. Microbenchmark suites of complex applications can comprise hundreds of individual benchmarks and take multiple hours to evaluate meaningfully, making running those benchmarks as part of CI/CD pipelines infeasible. In this paper, we reduce the total execution time of microbenchmark suites by leveraging the massive scalability and elasticity of FaaS (Function-as-a-Service) platforms. While using FaaS enables users to quickly scale up to thousands of parallel function instances to speed up microbenchmarking, the performance variation and low control over the underlying computing resources complicate reliable benchmarking. We present *ElastiBench*, an architecture for executing microbenchmark suites on cloud FaaS platforms, and evaluate it on code changes from an open-source time series database. Our evaluation shows that our prototype can produce reliable results (~95% of performance changes accurately detected) in a quarter of the time (≤15min vs. ~4h) and at lower cost ($0.49 vs. $1.18) compared to cloud-based virtual machines.

KEYWORDS

Serverless, FaaS, Microbenchmarks

1 INTRODUCTION

Microbenchmarks offer a fine-grained insight into the performance of an application by repeatedly benchmarking a small part of the application, e.g., on the function or package level [32]. This allows developers to gain detailed insight into the impact their code changes have on specific code paths. To achieve this, microbenchmarks can be run as part of a CI/CD (continuous integration and deployment) pipeline, which lets developers detect and fix performance regressions

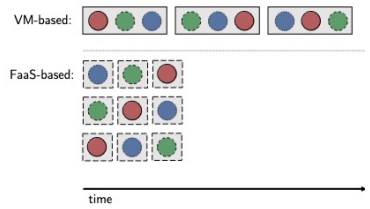


Figure 1: The traditional approach of microbenchmarking (top) relies on executing tasks (different circles) in random order multiple times on different virtual machines (gray boxes) to get reliable results. Using FaaS, these tasks can be executed on multiple function instances in parallel (bottom). With instance parallelism (three in this example), the duration of the suite run can be drastically reduced while also reducing inter-microbenchmark influences.

than 40 times on different virtual machines to achieve reliable results. This took ~4h and cost ~$1.14 for *VictoriaMetrics*[1], and took ~11h and cost ~$3.15 for *InfluxDB*[2] [23]. These benchmark durations and costs are prohibitive for integrating performance regression detection in CI/CD pipelines for *every* release (let alone every commit), where continuous evaluation with rapid developer feedback is paramount.

We propose leveraging Function-as-a-Service (FaaS) to massively scale out microbenchmark execution with little lead time, as shown in Figure 1: The elastic scalability of cloud FaaS platforms allows us to run hundreds of microbenchmarks in parallel, achieving robust results in a fraction of the

https://arxiv.org/pdf/2405.13528