On Benchmarking for Concurrent Runtime Verification

Mar 2021 · Luca Aceto, **Duncan Paul Attard**, Adrian Francalanza, Anna Ingólfsdóttir

SCS, Reykjavík University and CS, University of Malta

Introduction

“Do we really need another benchmarking tool?”

Deciding what to measure

CRV 2014: first step towards standardising RV benchmarks

Concurrent RV: do we measure in the same way?

Four sensible metrics for benchmarking concurrent RV

- Mean execution slowdown (s)
- Mean memory consumption (MB)
- Mean scheduler (or CPU) usage (%)
- Mean system response time (ms)

Deciding what to measure

CRV 2014: first step towards standardising RV benchmarks

Concurrent RV: do we measure in the same way?

Four sensible metrics for benchmarking concurrent RV

- Mean execution slowdown (s) ..less relevant
- Mean memory consumption (MB)
- Mean scheduler (or CPU) usage (%)
- Mean system response time (ms)

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking (scalability)
- High loads
- Parametrisability of model
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking (scalability)
- High loads (tests robustness)
- Parametrisability of model
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking (scalability)
- High loads (tests robustness)
- Parametrisability of model (benchmarks reproducibility)
- Repeatability of results

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking (scalability)
- High loads (tests robustness)
- Parametrisability of model (benchmarks reproducibility)
- Repeatability of results (shorter experiment convergence)

The nice-to-haves (or should we say, essential?)

Essential features for concurrent RV benchmarking

- Accurate metrics (precision)
- Different load profiles (scenario coverage)
- Growing and shrinking (scalability)
- High loads (tests robustness)
- Parametrisability of model (benchmarks reproducibility)
- Repeatability of results (shorter experiment convergence)

..And of course.. **adequate realism** in benchmarks

One way of doing things

Industry tradition

1. Deploy the system to be tested on a staging server
2. Use an established load testing tool, *e.g.* JMeter, Tsung, ...
3. Collect raw metrics, process and visualise

One way of doing things

Industry tradition

1. Deploy the system to be tested on a staging server
2. Use an established load testing tool, *e.g.* JMeter, Tsung, ...
3. Collect raw metrics, process and visualise

Good:

- Use existing tools
- Community support

One way of doing things

Industry tradition

1. Deploy the system to be tested on a staging server
2. Use an established load testing tool, *e.g.* JMeter, Tsung, ...
3. Collect raw metrics, process and visualise

Good:

- Use existing tools
- Community support

Bad:

- Depend on features offered
- Involved to set up
- Hard to reproduce

The middle way

> *Benchmark a simulated model of the system*

The middle way

> *Benchmark a simulated model of the system*

Bad:

- Needs to be developed

The middle way

“Benchmark a simulated model of the system”

Bad:

- Needs to be developed

Good:

- Packages moving parts
- Engineered for nice-to-haves

Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

- Periodic sampling

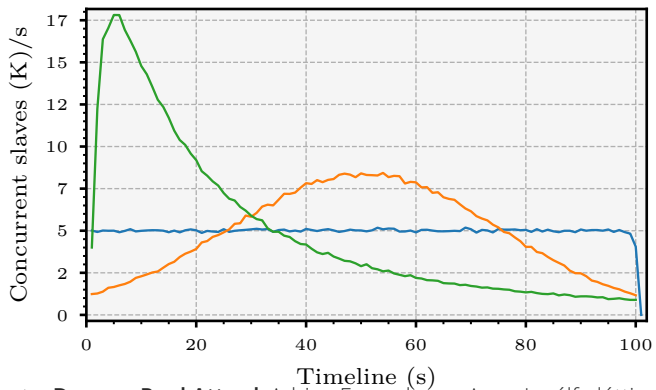
Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

- Periodic sampling
- Steady, Pulse, Burst models



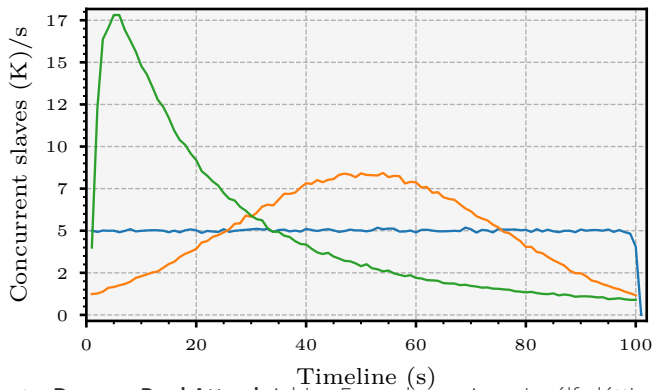
Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

- Periodic sampling
- Steady, Pulse, Burst models
- Dynamic process creation



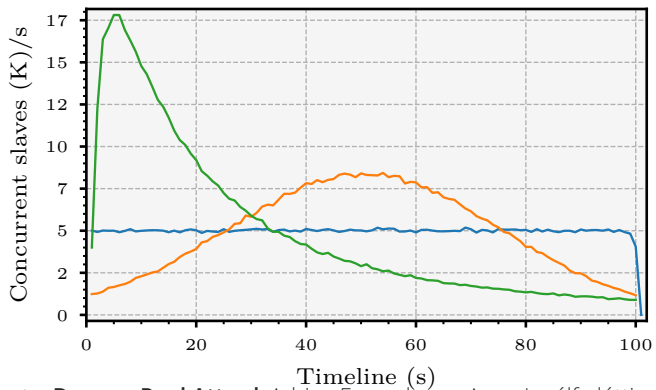
Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

- Periodic sampling
- Steady, Pulse, Burst models
- Dynamic process creation
- Lightweight processes



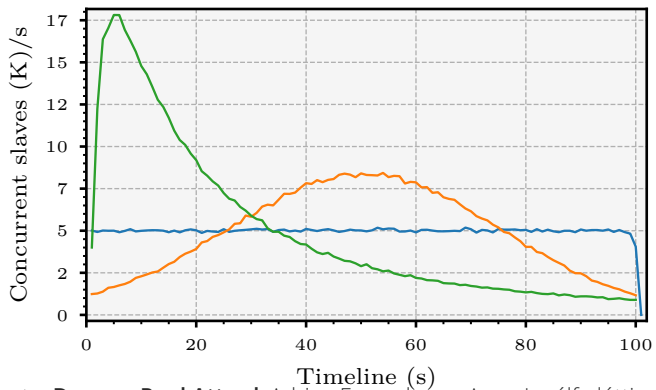
Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

Realised via:

- Periodic sampling
- Steady, Pulse, Burst models
- Dynamic process creation
- Lightweight processes
- Configurable probabilities



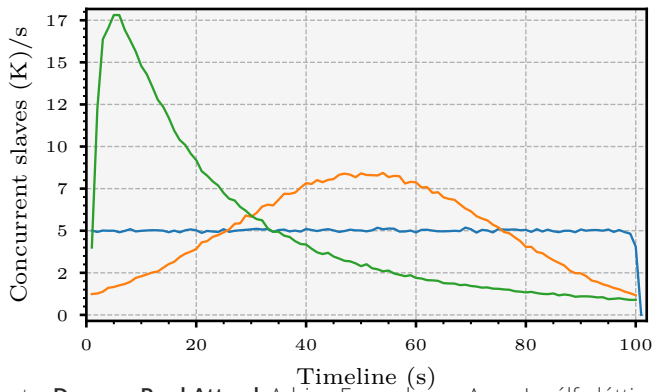
Design choices and implementation

Feature:

- Accurate metrics
- Different load profiles
- Growing and shrinking
- High loads
- Parametrisability of model
- Repeatability of results

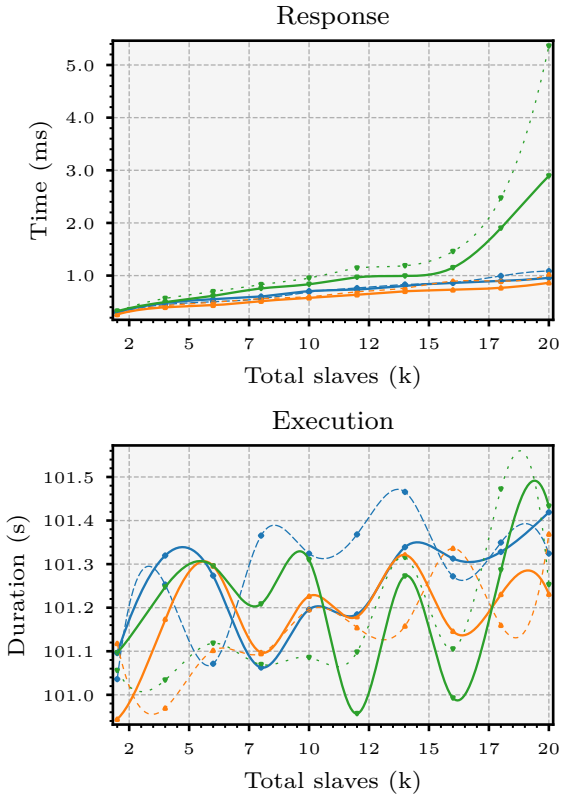
Realised via:

- Periodic sampling
- Steady, Pulse, Burst models
- Dynamic process creation
- Lightweight processes
- Configurable probabilities
- Configurable seeds



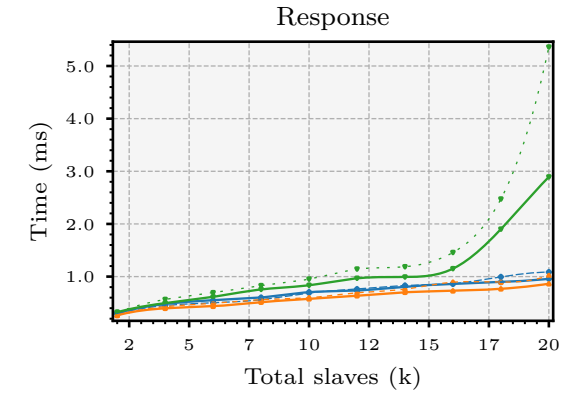
Concurent RV tool: Benchmark case study

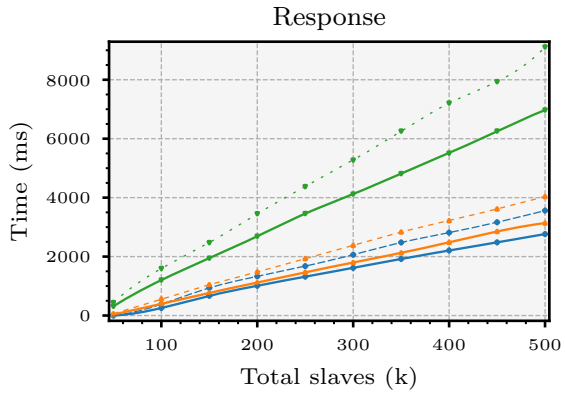
Steady, Pulse, and Burst loads induce **different behaviour**.

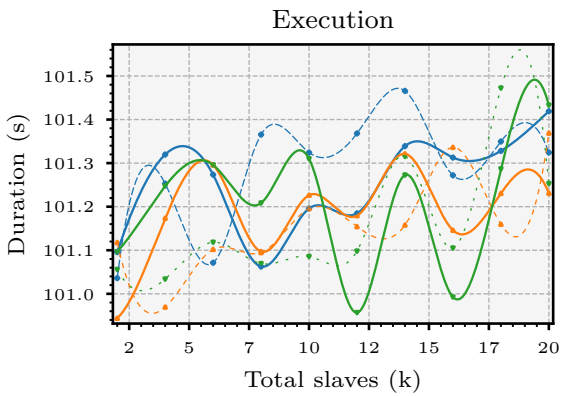


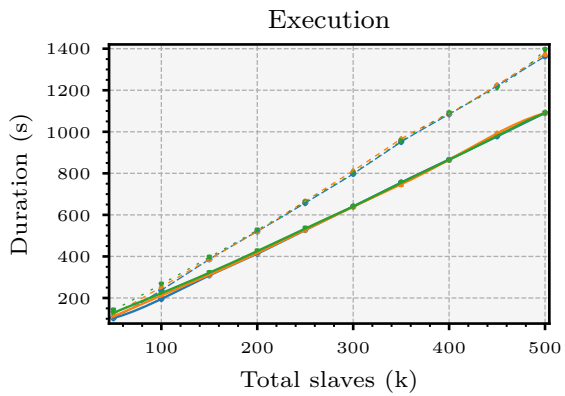
Concurent RV tool: Benchmark case study

Steady, Pulse, and Burst loads induce **different behaviour**.





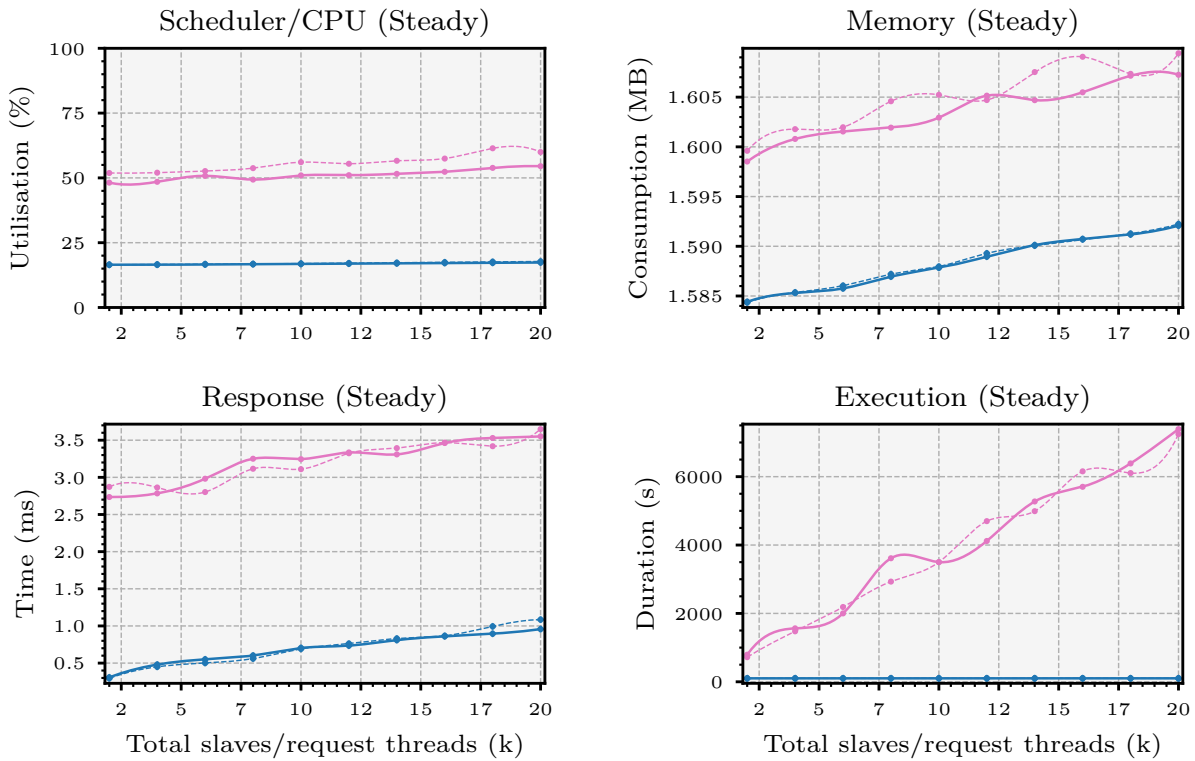




High loads (500k) enable us to confidently **extrapolate** results

Concurent RV tool: Synthetic vs. real system

Steady loads on synthetic and realistic set-ups for 20k



Different in measurements, but **corresponding** trends

Conclusion

“Do we really need another benchmarking tool?”

Multiple overhead metrics give a **comprehensive picture**

Different load profiles **increase coverage**

Scaling considerably to allow for **extrapolation**

Parametrisability enables **reproducibility of benchmarks**

Our tool captures the behaviour of **realistic** set-ups

Conclusion

“Yep! And with these features..”

Multiple overhead metrics give a **comprehensive picture**

Different load profiles **increase coverage**

Scaling considerably to allow for **extrapolation**

Parametrisability enables **reproducibility of benchmarks**

Our tool captures the behaviour of **realistic** set-ups

Thank you