Using the Raspberry Pi and Docker for Replicable Performance Experiments

Holger Knoche and Holger Eichelberger

University of Kiel and University of Hildesheim

April 13, 2018 @ ICPE

Motivation

Replicability is a fundamental property of scientific experiments.

Motivation

Replicability is a fundamental property of scientific experiments.

But: Replicating performance benchmarks is difficult

Motivation

Replicability is a fundamental property of scientific experiments.

But: Replicating performance benchmarks is difficult

Why? Researchers use the hardware and software environment that happens to be available to them

Research Questions

RQ 1 Which types of performance experiments can be appropriately replicated on a Pi?

RQ 2 Can Docker on the Pi facilitate the replicability of performance experiments?

RQ 3 Can we identify reasons for the response time fluctuations reported in our earlier work?

Agenda

1. Introduction ✓
2. Experimental Results
3. Fluctuation Cause Analysis
4. Conclusions

Experimental Approach

We...

1. ...bought three Raspberry Pi 3 devices
   - Two (D_1, D_2) from the same retailer within two weeks as a set with an SD card and a power supply
   - One (D_3) from another retailer a few months later

Experimental Approach

We...

1. ...bought three Raspberry Pi 3 devices
   - Two (D_1, D_2) from the same retailer within two weeks as a set with an SD card and a power supply
   - One (D_3) from another retailer a few months later
2. ...created a master SD card image
   - Based on Raspbian Stretch Lite
   - Included Docker

Experimental Approach

We...

1. ...bought three Raspberry Pi 3 devices
   - Two (D_1, D_2) from the same retailer within two weeks as a set with an SD card and a power supply
   - One (D_3) from another retailer a few months later
2. ...created a master SD card image
   - Based on Raspbian Stretch Lite
   - Included Docker
3. ...shared the master image among the authors

Experimental Approach

We...

1. ...bought three Raspberry Pi 3 devices
    - Two (D_1, D_2) from the same retailer within two weeks as a set with an SD card and a power supply
    - One (D_3) from another retailer a few months later
2. ...created a master SD card image
    - Based on Raspbian Stretch Lite
    - Included Docker
3. ...shared the master image among the authors
4. ...ran the preconfigured benchmarks on the devices

Experiments

E_1 Microbenchmarks with the Java Microbenchmark Harness (JMH)

- 4 benchmarks regarding method invocations with different compiler settings
- 2 benchmarks regarding file and network I/O

E_1 Microbenchmarks with the Java Microbenchmark Harness (JMH)

- 4 benchmarks regarding method invocations with different compiler settings
- 2 benchmarks regarding file and network I/O

E_2 Monitoring Overhead Benchmark MooBench

Experiments

E_1 Microbenchmarks with the Java Microbenchmark Harness (JMH)

- 4 benchmarks regarding method invocations with different compiler settings
- 2 benchmarks regarding file and network I/O

E_2 Monitoring Overhead Benchmark MooBench

E_3 Web Service built on Spring Boot and JPA

- Web service and database deployed on different Pi devices
- Load driver on separate machine

Experiments

E_1 Microbenchmarks with the Java Microbenchmark Harness (JMH)

- 4 benchmarks regarding method invocations with different compiler settings
- 2 benchmarks regarding file and network I/O

E_2 Monitoring Overhead Benchmark MooBench

E_3 Web Service built on Spring Boot and JPA

- Web service and database deployed on different Pi devices
- Load driver on separate machine

E_4 Java EE Benchmark SPECjEnterprise 2010

Selected Results from Benchmark 1 (Method Invocation)

Configuration	99.9% CI Throughput *(inv/s)*
D_2 + H_1, native	[12,314,426 ; 12,329,982]
D_3 + H_1, native	[12,307,645 ; 12,320,718]
D_2 + H_1, Docker	[12,290,439 ; 12,308,655]

Selected Results from Benchmark 1 (Method Invocation)

Configuration	99.9% CI Throughput *(inv/s)*
D_2 + H_1, native	[12,314,426 ; 12,329,982]
D_3 + H_1, native	[12,307,645 ; 12,320,718]
D_2 + H_1, Docker	[12,290,439 ; 12,308,655]

Selected Results from Benchmark 5 (File I/O)

Setup	99.9% CI Throughput
D_2 + H_1, native	[541,361 ; 566,493]
D_3 + H_1, native	[536,530 ; 561,079]
D_2 + H_1, Docker	[852,325 ; 912,492]

?

Selected Results from Benchmark 6 (Network I/O)

Setup	99.9% CI Throughput
D_2 + H_1, native	[192,311 ; 199,128]
D_3 + H_1, native	[192,632 ; 198,823]
D_2 + H_1, Docker	[184,944 ; 192,154]

Selected Results

Experiment	95% CI RT (μs, D_1)	... (D_2)	... (D_3)
Baseline	[0.5; 0.5]	[0.5; 0.5]	[0.5; 0.5]

E_2: MooBench

Selected Results

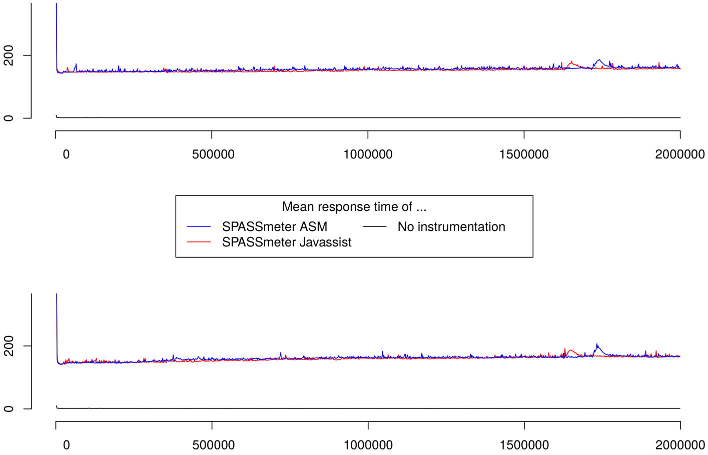
Experiment	**95% CI RT** (μs, D_1)	**...** (D_2)	**...** (D_3)
Baseline	[0.5; 0.5]	[0.5; 0.5]	[0.5; 0.5]
SPASS native	[153.5; 153.5]	[145.0; 145.0]	[151.6; 151.7]
SPASS Docker	[152.0; 152.0]	[147.6;147.8]	[155.0; 155.4]

Selected Results

Experiment	95% CI RT (μs, D_1)	... (D_2)	... (D_3)
Baseline	[0.5; 0.5]	[0.5; 0.5]	[0.5; 0.5]
SPASS native	[153.5; 153.5]	[145.0; 145.0]	[151.6; 151.7]
SPASS Docker	[152.0; 152.0]	[147.6;147.8]	[155.0; 155.4]
Kieker native	[118.8; 124.3]	[113.6; 118.3]	[116.2; 121.1]
Kieker Docker	[128.7; 134.2]	[120.8; 125.8]	[118.2;122.8]

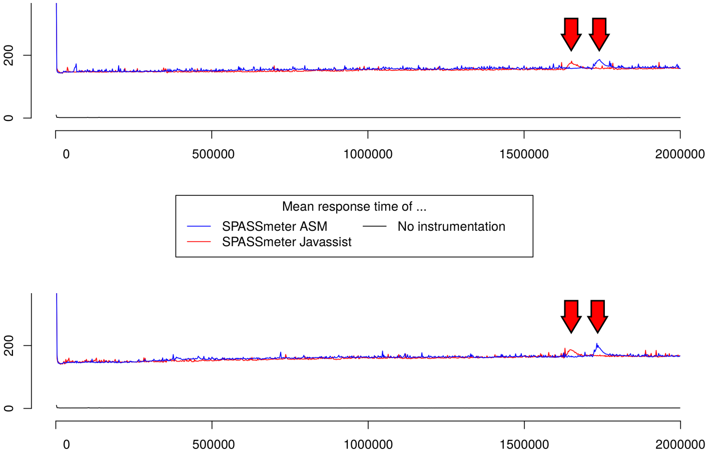
E_2: MooBench

Experimental Results



E_2: MooBench

Experimental Results



Selected Results (Updating Service Method)

Setup	**95% CI RT** (μs)
Web: D_2 + H_1, DB: D_3 + H_2, native	[354,663 ; 357,726]
Web: D_3 + H_1, DB: D_2 + H_2, native	[352,361 ; 355,460]
Web: D_2 + H_1, DB: D_3 + H_2, Docker	[353,324 ; 356,351]
Web: D_3 + H_1, DB: D_2 + H_2, Docker	[379,799 ; 382,993]
Web: D_2 + H_2, DB: D_3 + H_1, native	[549,424 ; 553,402]

Results for Method “Create Vehicle” (EJB)

Setup	95% CI RT
Web: D_2 + H_1, DB: D_3 + H_2, native	[0.228 ; 0.242]
Web: D_3 + H_1, DB: D_2 + H_2, native	[0.256 ; 0.275]
Web: D_2 + H_1, DB: D_3 + H_2, Docker	[0.236 ; 0.251]

Results for Method “Create Vehicle” (EJB)

Setup	95% CI RT
Web: $D_2 + H_1$, DB: $D_3 + H_2$, native	[0.228 ; 0.242]
Web: $D_3 + H_1$, DB: $D_2 + H_2$, native	[0.256 ; 0.275]
Web: $D_2 + H_1$, DB: $D_3 + H_2$, Docker	[0.236 ; 0.251]

Results for Method “Create Vehicle” (WS)

Setup	95% CI RT
Web: $D_2 + H_1$, DB: $D_3 + H_2$, native	[0.411 ; 0.484]
Web: $D_3 + H_1$, DB: $D_2 + H_2$, native	[0.807 ; 1.012]
Web: $D_2 + H_1$, DB: $D_3 + H_2$, Docker	[0.641 ; 0.795]

?

Results for Method “Create Vehicle” (EJB)

Setup	95% CI RT
Web: $D_2 + H_1$, DB: $D_3 + H_2$, native	[0.228 ; 0.242]
Web: $D_3 + H_1$, DB: $D_2 + H_2$, native	[0.256 ; 0.275]
Web: $D_2 + H_1$, DB: $D_3 + H_2$, Docker	[0.236 ; 0.251]

Results for Method “Create Vehicle” (WS)

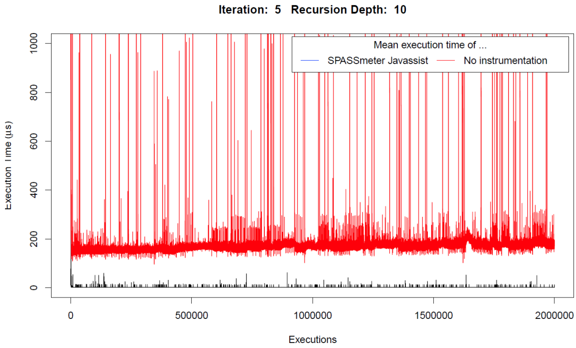
Setup	95% CI RT
Web: $D_2 + H_1$, DB: $D_3 + H_2$, native	[0.411 ; 0.484]
Web: $D_3 + H_1$, DB: $D_2 + H_2$, native	[0.807 ; 1.012]
Web: $D_2 + H_1$, DB: $D_3 + H_2$, Docker	[0.641 ; 0.795]

?

And: Benchmark **fails** due to insufficient throughput.

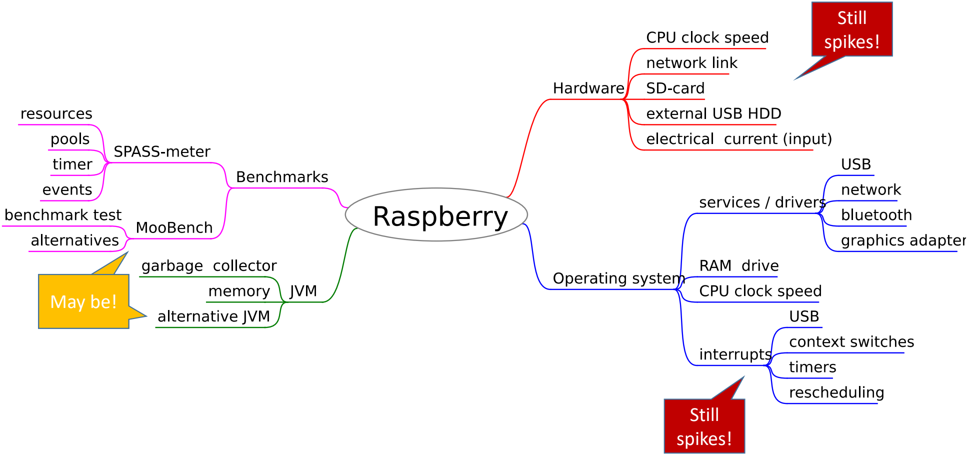
SPASS-meter Results Revisited

- Results indicate good replicability, but
  - Already baseline fluctuates $\sigma \approx 8 \cdot \bar{\mu}$, $max \approx 65 \cdot \bar{\mu}$
  - Raw data very noisy, e.g., SPASS-meter



$\sigma \approx 3 \cdot \bar{\mu}$ $max \approx 125 \cdot \bar{\mu}$

Cause Tree


Raspberry
Hardware
CPU clock speed
network link
SD-card
external USB HDD
electrical current (input)
Still spikes!
Operating system
services / drivers
USB
network
bluetooth
graphics adapter
RAM drive
CPU clock speed
interrupts
USB
context switches
timers
rescheduling
Still spikes!
Benchmarks
SPASS-meter
resources
pools
timer
events
MooBench
benchmark test
alternatives
JVM
garbage collector
memory
alternative JVM
May be!

Detailed Results

Experiment	SPASS-meter				
	mean	σ	min	max	peaks
from SSP'18	164.8	44.1	91.9	19,228.7	1,155

Detailed Results

Experiment	SPASS-meter				
	mean	σ	min	max	peaks
from SSP'18	164.8	44.1	91.9	19,228.7	1,155
object pools	152.3	142.5	89.8	370,604.0	**818**
parallel GC	**194.4**	56.7	110.1	27,715.9	**6,901**
time measurement	**146.3**	34.9	88.5	13,034.8	406
one CPU core	**492.8**	**427.1**	86.0	13,560.1	**37,360**

Detailed Results

Experiment	SPASS-meter				
	mean	σ	min	max	peaks
from SSP'18	164.8	44.1	91.9	19,228.7	1,155
object pools	152.3	142.5	89.8	370,604.0	**818**
parallel GC	**194.4**	56.7	110.1	27,715.9	**6,901**
time measurement	**146.3**	34.9	88.5	13,034.8	406
one CPU core	**492.8**	**427.1**	86.0	13,560.1	**37,360**
no recursion	17.6	1.8	11.35	3,361.3	**53**

Detailed Results

Experiment	SPASS-meter				
	mean	σ	min	max	peaks
from SSP'18	164.8	44.1	91.9	19,228.7	1,155
object pools	152.3	142.5	89.8	370,604.0	**818**
parallel GC	**194.4**	56.7	110.1	27,715.9	**6,901**
time measurement	**146.3**	34.9	88.5	13,034.8	406
one CPU core	**492.8**	**427.1**	86.0	13,560.1	**37,360**
no recursion	17.6	1.8	11.35	3,361.3	**53**

- Cause appears to be related to the method under test
- Effect is also observable on other machines
- $\Rightarrow$ Not specific to the Pi

Summary

RQ1 Which types of performance experiments can be appropriately replicated on a Pi?

- Pi is well suited for (non I/O intensive) microbenchmarks
- Macro benchmarks may work, but... peripherals, storage devices
- Less suited for enterprise-scale benchmarks

Summary

RQ2 Can Docker on the Pi facilitate the replicability of performance experiments?

- Docker is a valuable tool,...
- ..., but may affect performance and variances

RQ3 Can we identify reasons for the response time fluctuations reported in our earlier work?

- Yes, cause not specific to the Pi platform

Lessons Learned

- Docker facilitates benchmarks, fosters experimentation
- For I/O-heavy workloads, don't use SD cards
- Additional peripherals may threaten power supply
- Container networking can be tricky
- License issues may impede replication / publication[1]

[1] Materials on Zenodo: https://doi.org/10.5281/zenodo.1100975

Future Work

Near future

- More experiments on deviations
- Other single-board computers, next Pi generation
- Larger number of devices

Future Work

Near future

- More experiments on deviations
- Other single-board computers, next Pi generation
- Larger number of devices

Not-so-near future

- Foster community practice
- Public benchmark repository for sharing of experiments
- Address license issues impeding replication
- Investigate or develop further suitable platforms