Runtime Instrumentation for Reactive Components

ECOOP 2024 · L. Aceto · **D. P. Attard** · A. Francalanza · A. Ingólfsdóttir

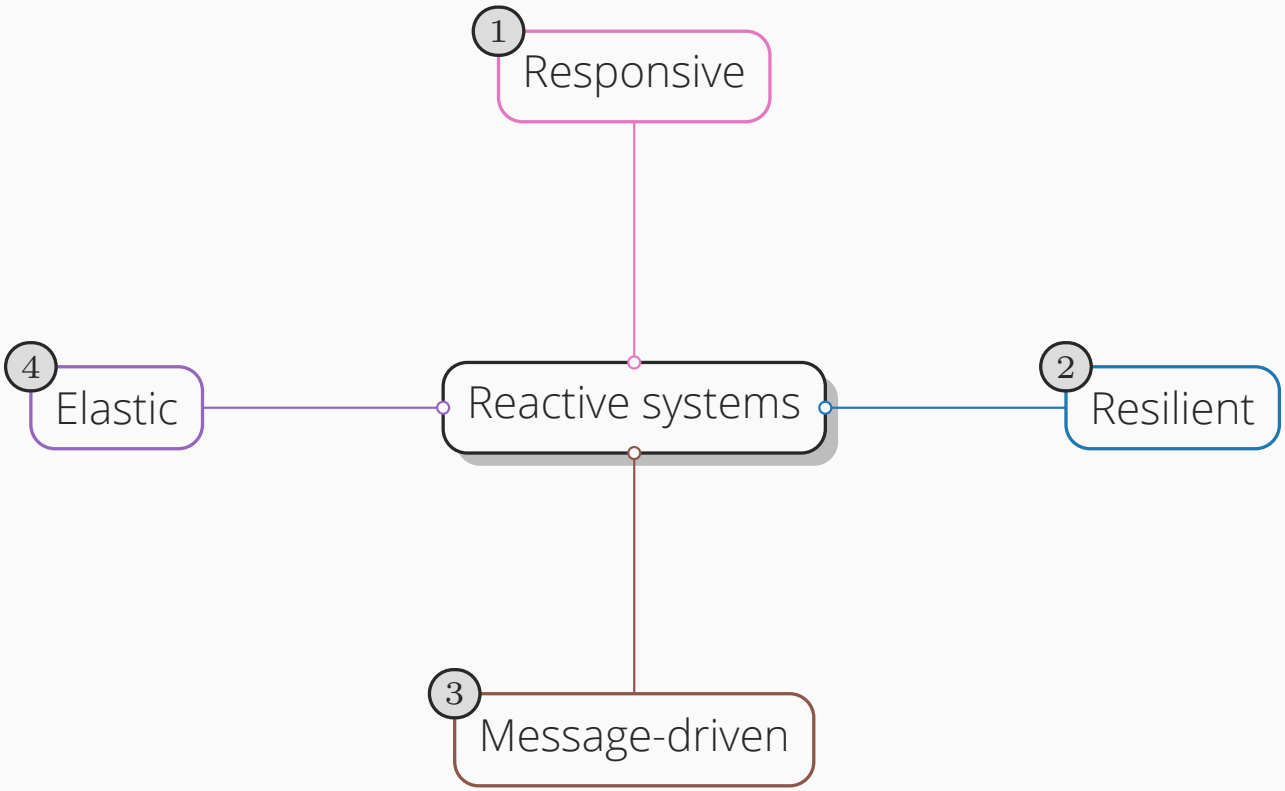
University of Glasgow

L-Università ta' Malta

Háskólinn í Reykjavík
Reykjavik University

GSSI
GRAN SASSO SCIENCE INSTITUTE

Reactive systems



Runtime monitoring... Why?

> *Understanding system behaviour requires the system to run*

profiling · resource usage analysis · security audit trails

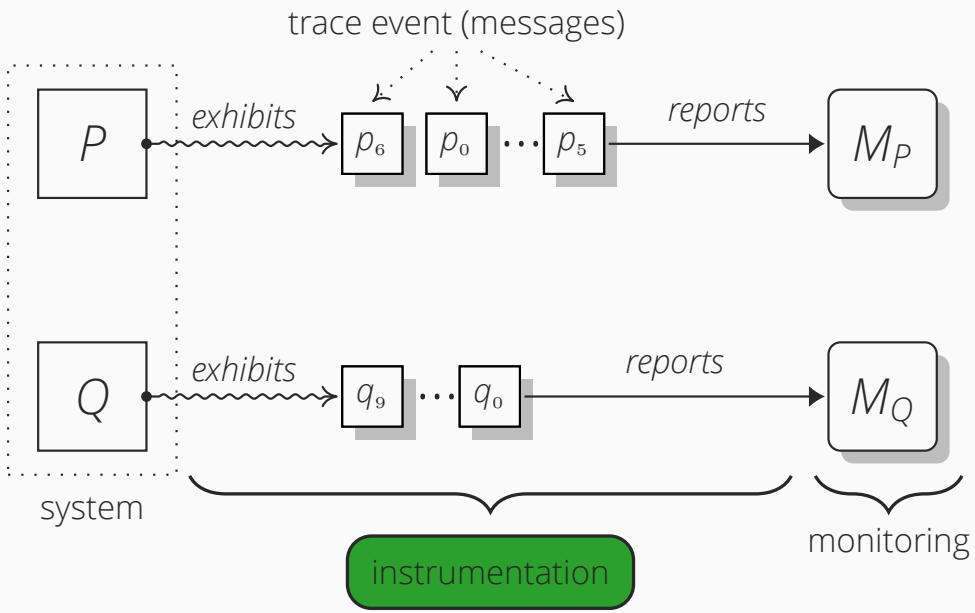
OR

> *Correctness of system is hard to analyse statically*

debugging · information flow · runtime verification

Runtime monitoring... How?

> *Extract runtime information and report it to monitors*



Runtime monitoring requirements

> *Instrumentation **must** preserve the reactiveness of the system*

Runtime monitoring requirements

> *Instrumentation **must** preserve the reactiveness of the system*

Low overhead preserves the Responsive attribute

Independent failure preserves the Resilient attribute

Non-blocking preserves the Message-driven attribute

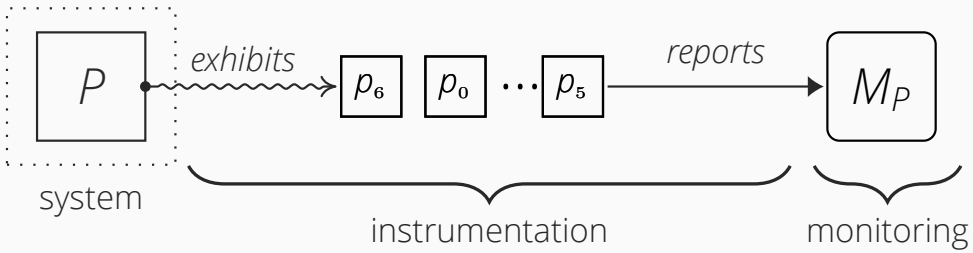
Grows and shrinks preserves the Elastic attribute

Runtime monitoring requirements

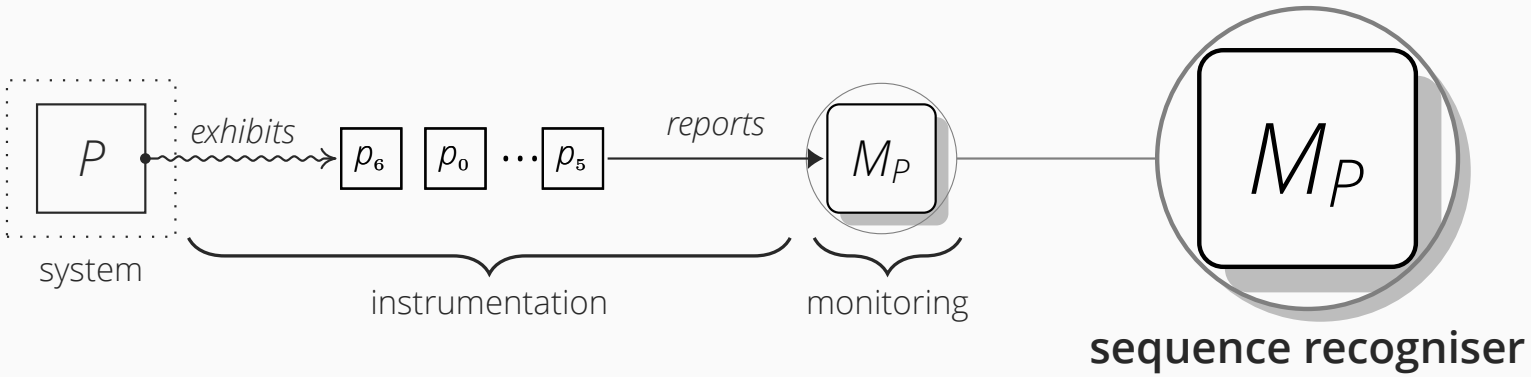
> *Instrumentation **must** preserve the reactiveness of the system*

= the instrumented system **remains** Reactive

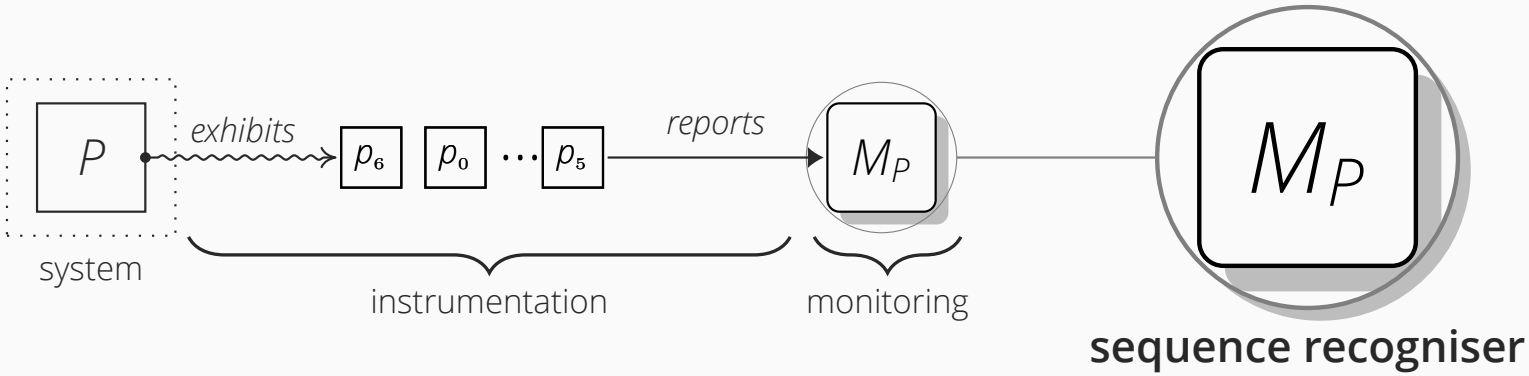
Runtime verification requirements



Runtime verification requirements



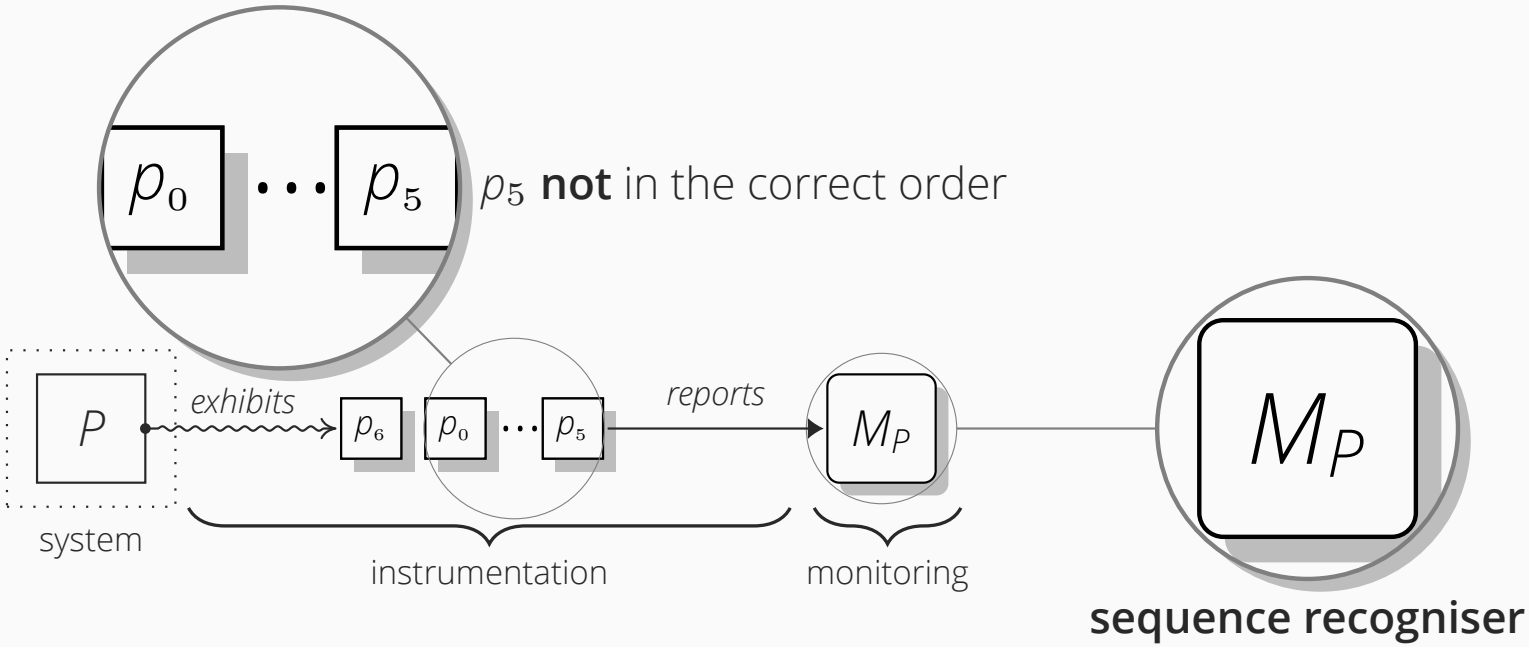
Runtime verification requirements



Trace soundness

- **Complete:** trace contains **all** the events exhibited by *P* so far
- **Consistent:** events reflect the **same order** *P* exhibits them

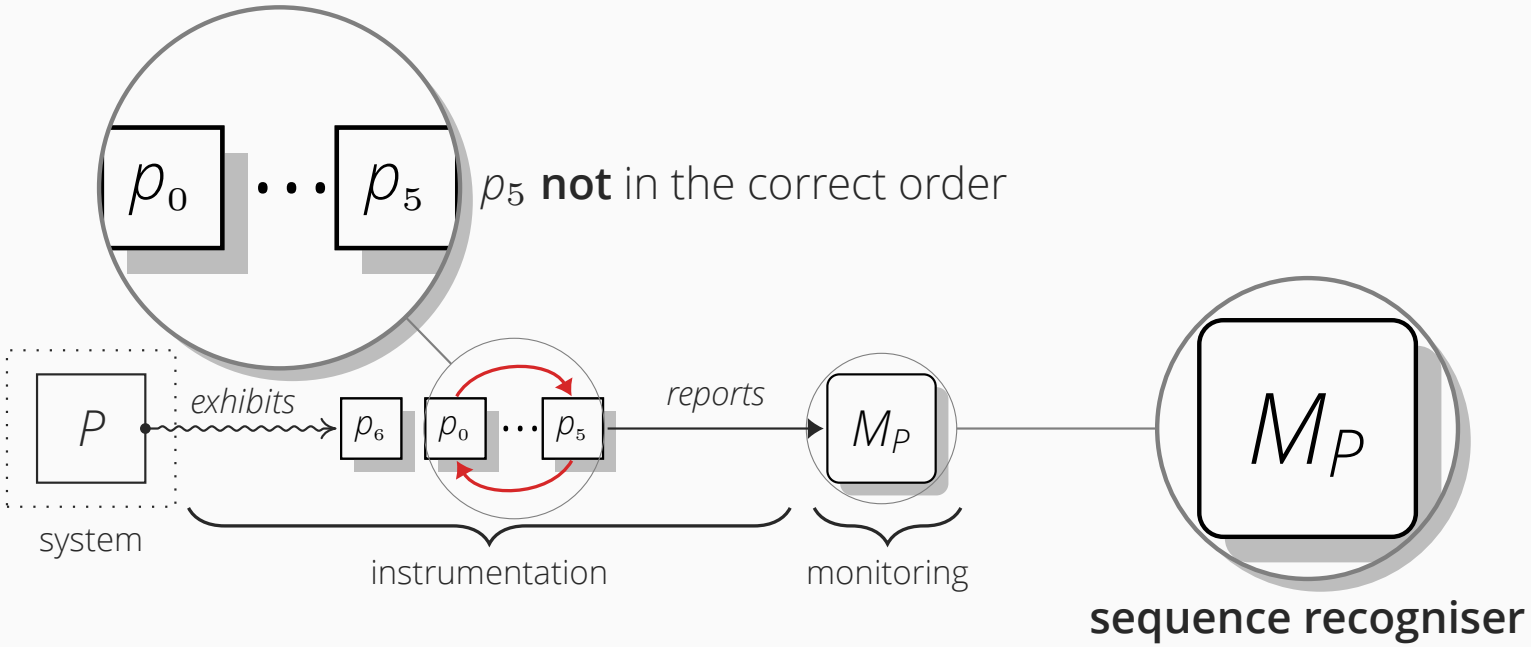
Runtime verification requirements



Trace soundness

- **Complete:** trace contains **all** the events exhibited by *P* so far
- **Consistent:** events reflect the **same order** *P* exhibits them

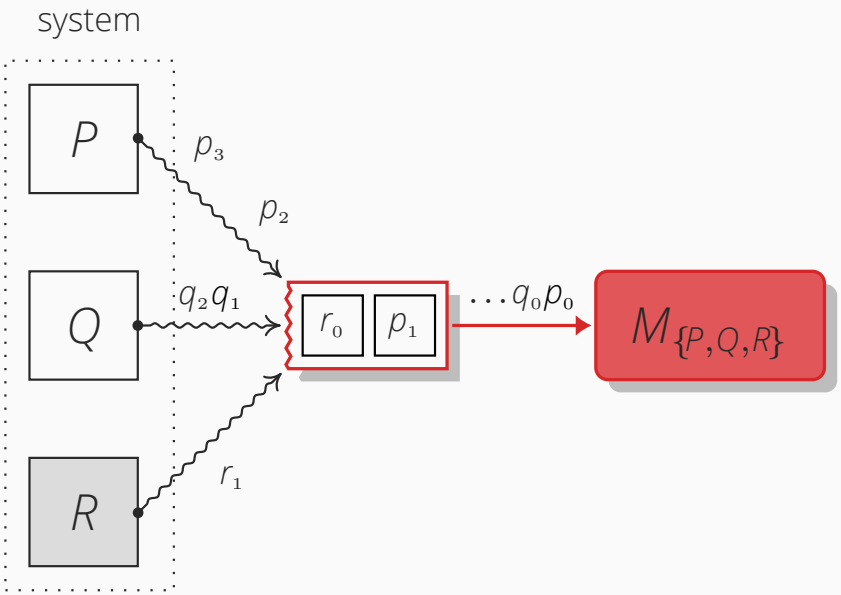
Runtime verification requirements



Trace soundness

- **Complete:** trace contains **all** the events exhibited by *P* so far
- **Consistent:** events reflect the **same order** *P* exhibits them

Centralised outline instrumentation



High overhead ✗ Responsive

Suffers from SPOF ✗ Resilient

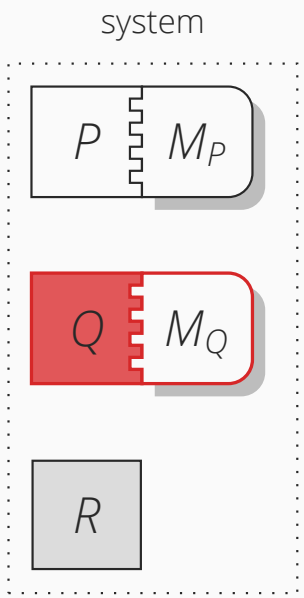
Asynchronous ✓ Message-driven

Does not shrink ✗ Elastic

Not scalable due to contention and singleton monitor

Requires **demultiplexing** for analysing events

Inline instrumentation



Typically low overhead ✓ Responsive

Embeds monitors ✗ Resilient

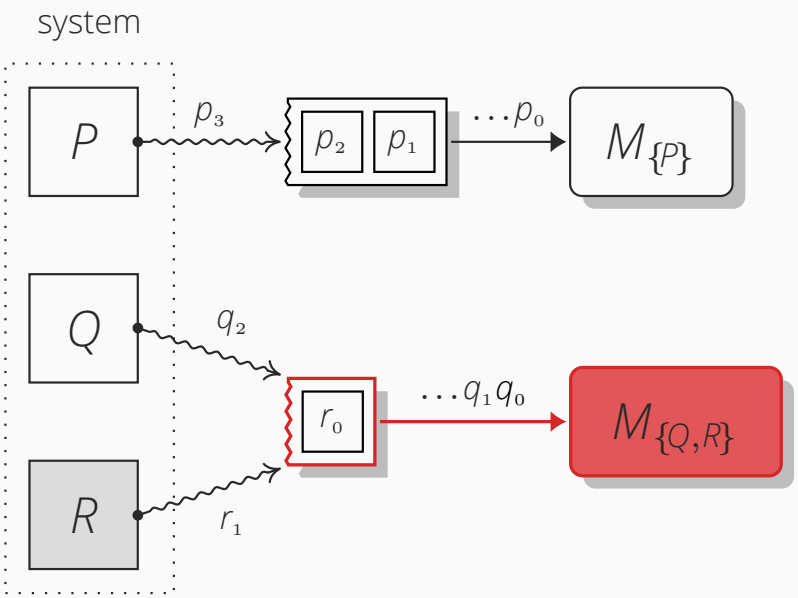
Synchronous ✗ Message-driven

Grows and shrinks ✓ Elastic

Inapplicable when code modification is not possible

Slow monitors may **impact latency**

Decentralised outline instrumentation



Feasible overhead ✓ Responsive

Isolates monitors ✓ Resilient

Asynchronous ✓ Message-driven

Grows and shrinks ✓ Elastic

Tracing uses an asynchronous **tracing infrastructure**

Dynamic outline instrumentation = **challenging** engineering

Criticism against decentralised outline monitoring

> *Decentralised outline monitoring induces **high** overhead*

Criticism against decentralised outline monitoring

> *Decentralised outline monitoring induces **high** overhead*

What **is** runtime overhead?

Many take the **execution duration** as an overhead metric

Criticism against decentralised outline monitoring

> *Decentralised outline monitoring induces **high** overhead*

What **is** runtime overhead?

~~Many take the **execution duration** as an overhead metric~~ ✗

Latency

Memory consumption + Scheduler usage ✓

RIARC

*“ A **reactive** decentralised outline instrumentation algorithm ”*

L. Aceto · **D. P. Attard** · A. Francalanza · A. Ingólfsdóttir · RU · GSSI · UM · U*of*G

Core idea of RIARC (details in paper)

> *Buffers react to **key trace events** to reorganise monitors*

Core idea of RIARC (details in paper)

> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($-\diamond$), exit ($*$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)

Core idea of RIARC (details in paper)

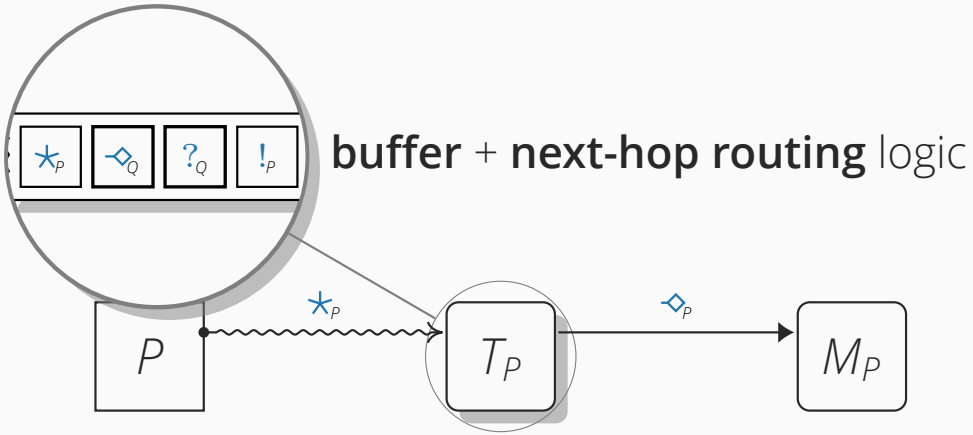
> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($\multimap\!\diamond$), exit ($*$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)



Core idea of RIARC (details in paper)

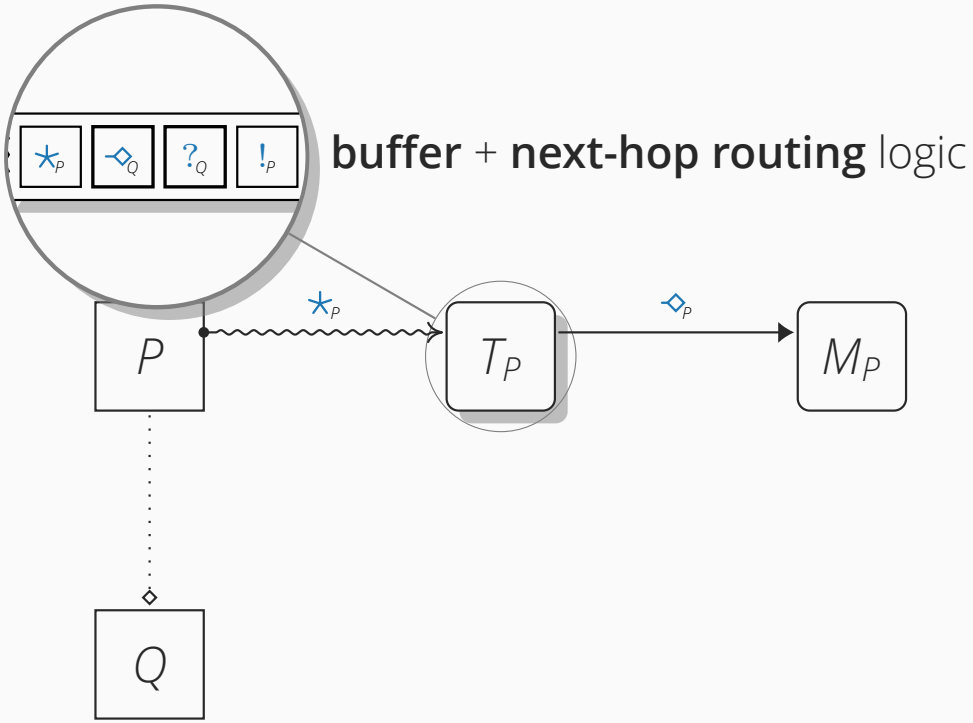
> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($\multimap\!\diamond$), exit ($*$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)



Core idea of RIARC (details in paper)

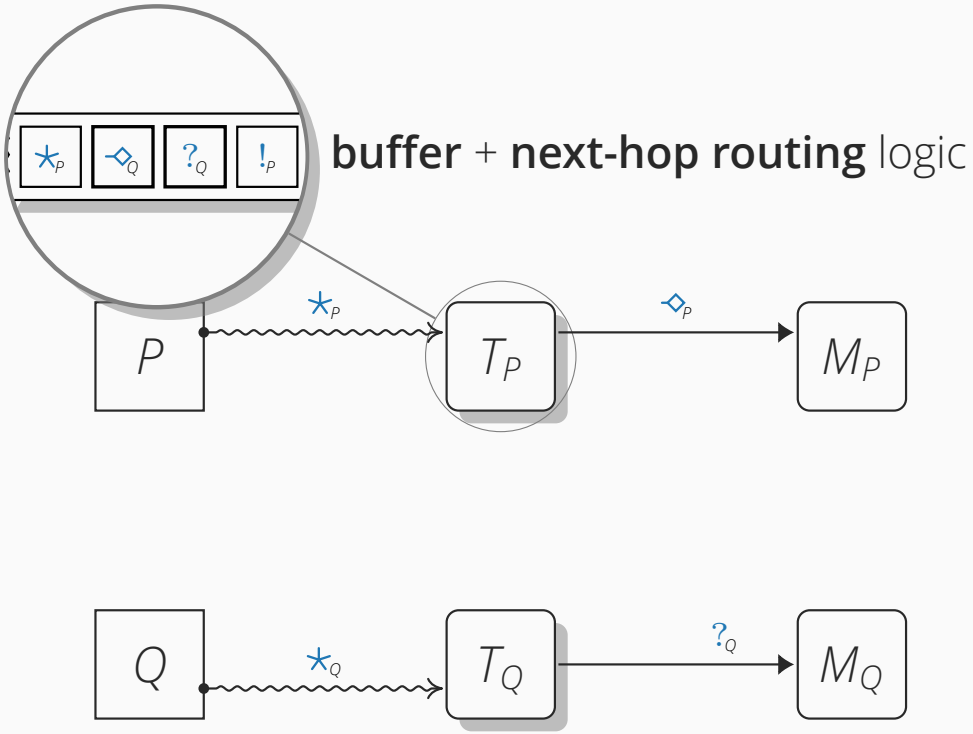
> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($\multimap$), exit ($\ast$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)



Core idea of RIARC (details in paper)

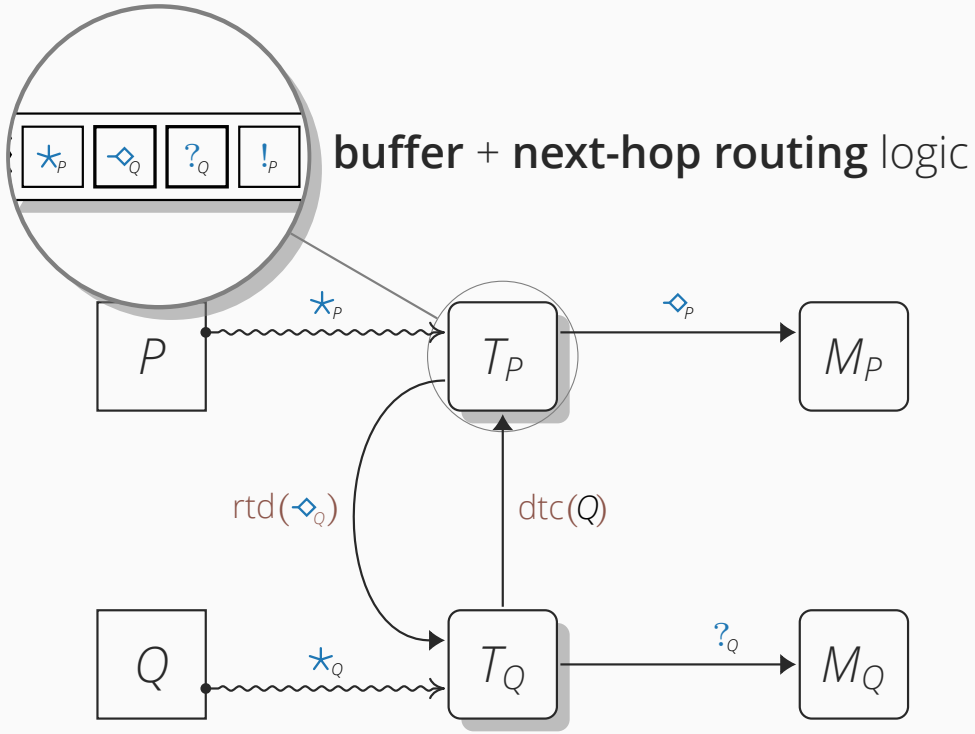
> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($\multimap\!\diamond$), exit ($\ast$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)



Core idea of RIARC (details in paper)

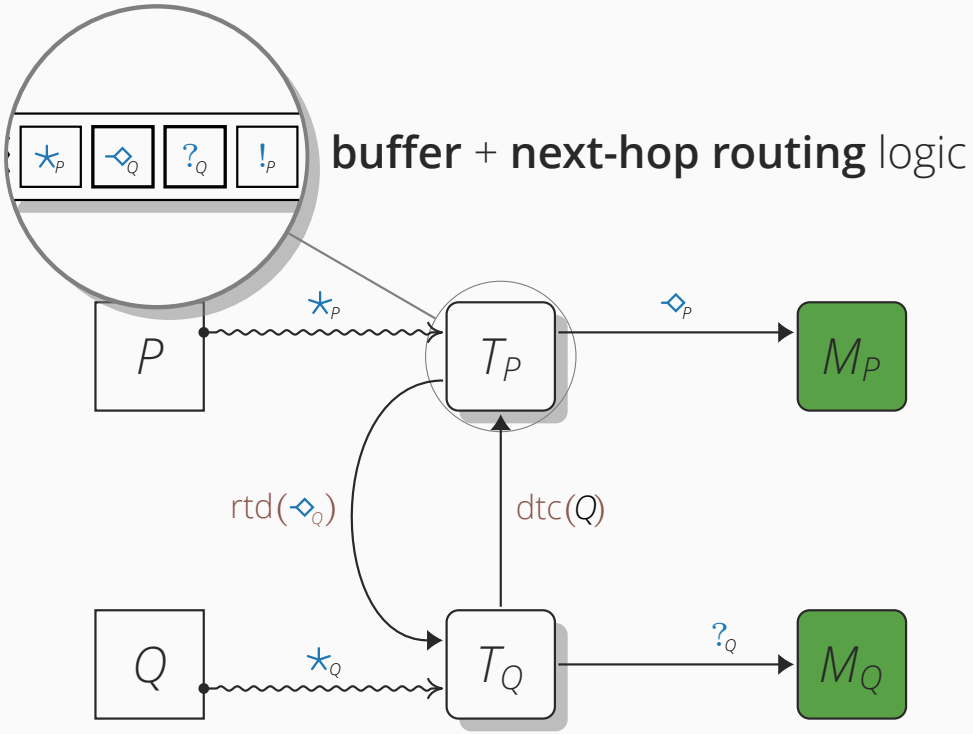
> *Buffers react to **key trace events** to reorganise monitors*

Trace events

- spawn ($\multimap\!\diamond$), exit ($\ast$)
- send ($!$), receive ($?$)

Control messages

- route packet (rtd)
- detach request (dtc)



Evlauating RIARC

L. Aceto · **D. P. Attard** · A. Francalanza · A. Ingólfsdóttir · RU · GSSI · UM · U*of*G

Evlauating RIARC

Implementability

To confirm that RIARC can be used in **practice**

Evaluating RIARC

Implementability

To confirm that RIARC can be used in **practice**

Correctness

To confirm that all traces RIARC reports are **sound**

Evaluating RIARC

Implementability

To confirm that RIARC can be used in **practice**

Correctness

To confirm that all traces RIARC reports are **sound**

Performance

To confirm that RIARC preserves the system **reactiveness**

Summary of empirical experiments (details in paper)

Limited hardware - **40 M** events

High concurrency experiments

(short-lived, computationally-light tasks)

Summary of empirical experiments (details in paper)

Limited hardware - **40 M** events

High concurrency experiments

(short-lived, computationally-light tasks)

Commodity hardware - **200 M** events

Moderate concurrency experiments

(long-lived, computationally-intensive tasks)

Summary of empirical experiments (details in paper)

Limited hardware - **40 M** events

High concurrency experiments

(short-lived, computationally-light tasks)

Commodity hardware - **200 M** events

Moderate concurrency experiments

(long-lived, computationally-intensive tasks)

Centralised is **impractical**

Inline is the **most efficient**

Latency RIARC ≈ latency Inline

Summary of empirical experiments (details in paper)

Limited hardware - **40 M** events

High concurrency experiments

(short-lived, computationally-light tasks)

Centralised is **impractical**

Inline is the **most efficient**

Latency RIARC ≈ latency Inline

Commodity hardware - **200 M** events

Moderate concurrency experiments

(long-lived, computationally-intensive tasks)

RIARC **scales** and uses schedulers

Latency RIARC **=** latency Inline

Summary of empirical experiments (details in paper)

Limited hardware - **40 M** events

High concurrency experiments

(short-lived, computationally-light tasks)

Centralised is **impractical**

Inline is the **most efficient**

Latency RIARC ≈ latency Inline

Commodity hardware - **200 M** events

Moderate concurrency experiments

(long-lived, computationally-intensive tasks)

RIARC **scales** and uses schedulers

Latency RIARC **=** latency Inline

Inline prone to **slow analysis**

Instrumentation yields **most** overhead %

Take away

Centralised instrumentation

Applicable ✗

Inline instrumentation

Low overhead ✓

May impact system ✗

Not always applicable ✗

RIARC

- Leaves the system **reactive** ✓
- Guarantees **trace soundness** ✓
- **Low overhead** feasible for soft real-time applications ✓

Further details

Paper (extended version)

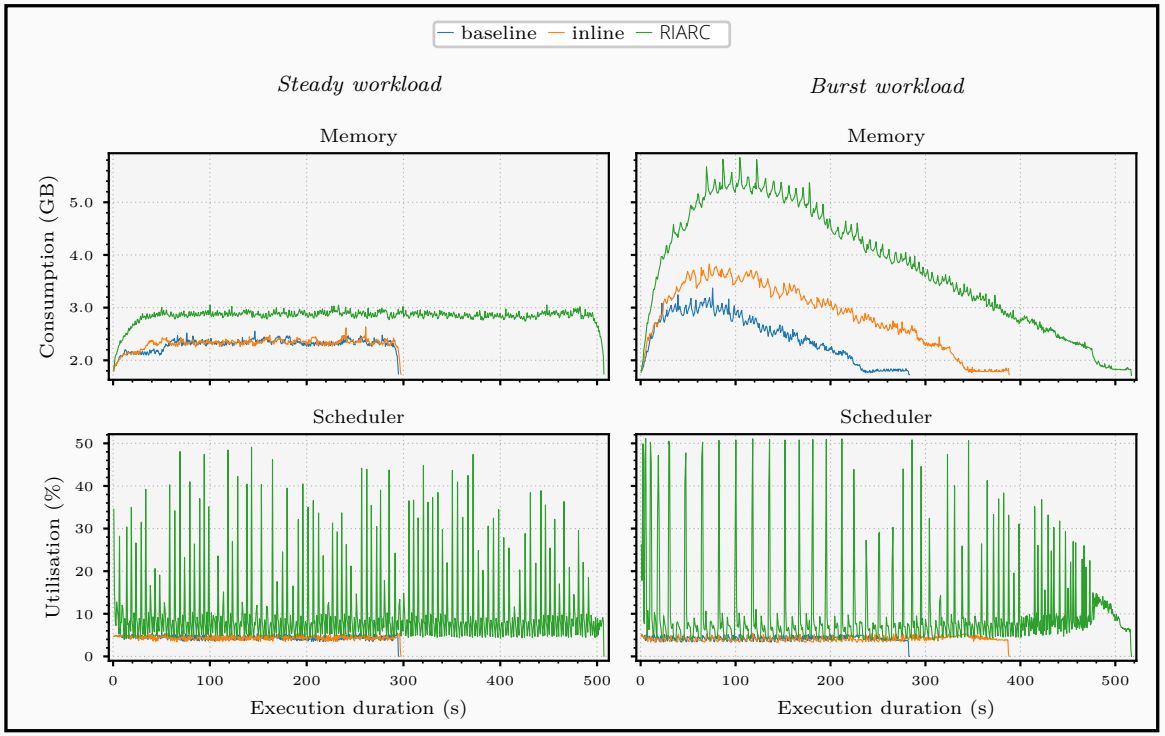
RIARC use in detectEr

Backup slides

Aim: study scalability and elasticity of monitoring

Resource usage

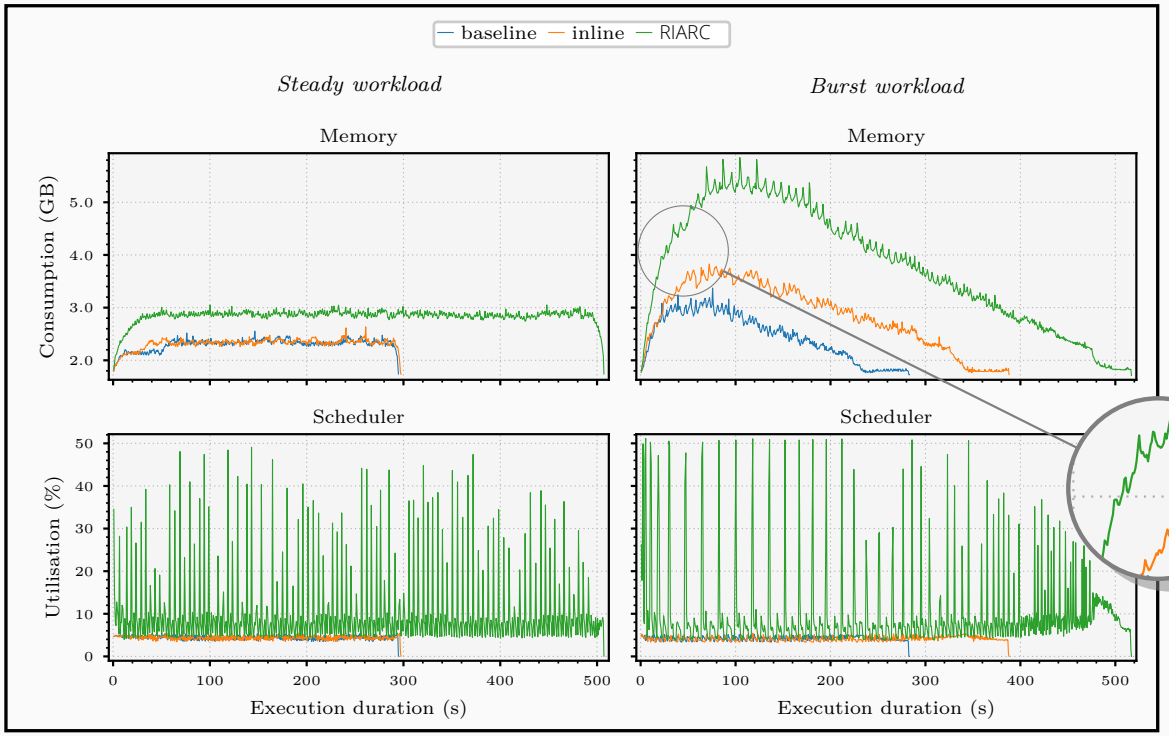
(complete system runs)



Aim: study scalability and elasticity of monitoring

Resource usage

(complete system runs)



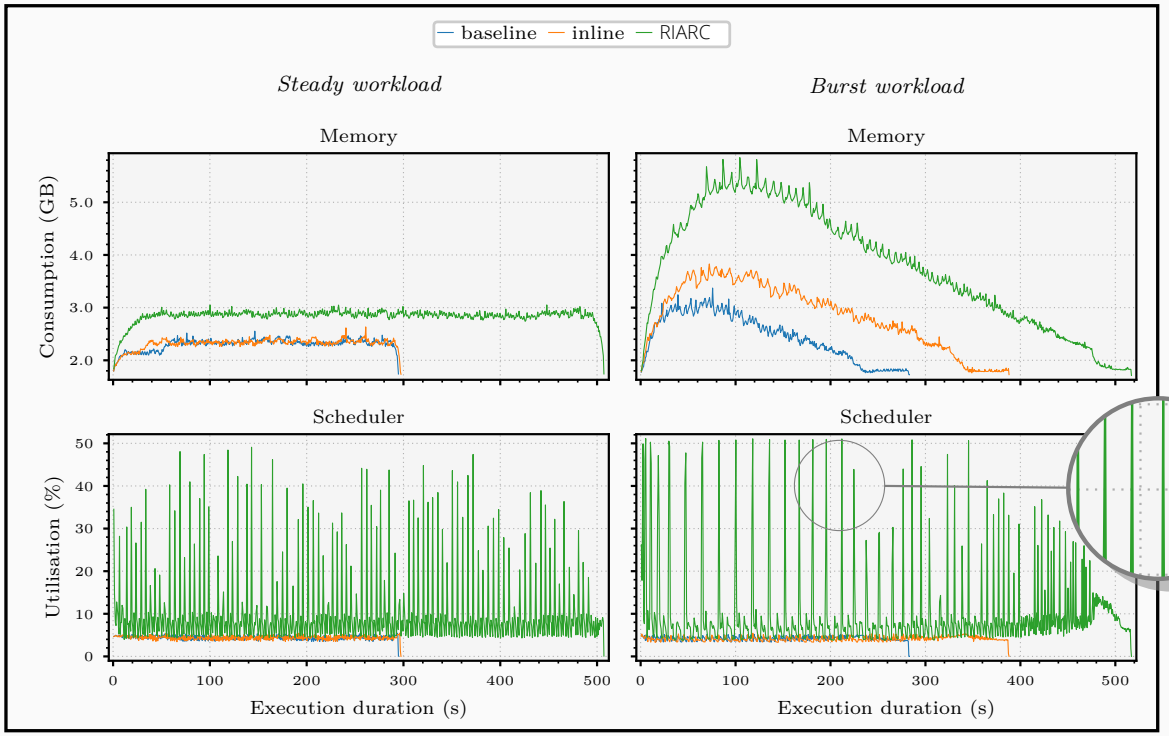
Inline RIARC

Elastic and follow shape of baseline workload

Aim: study scalability and elasticity of monitoring

Resource usage

(complete system runs)



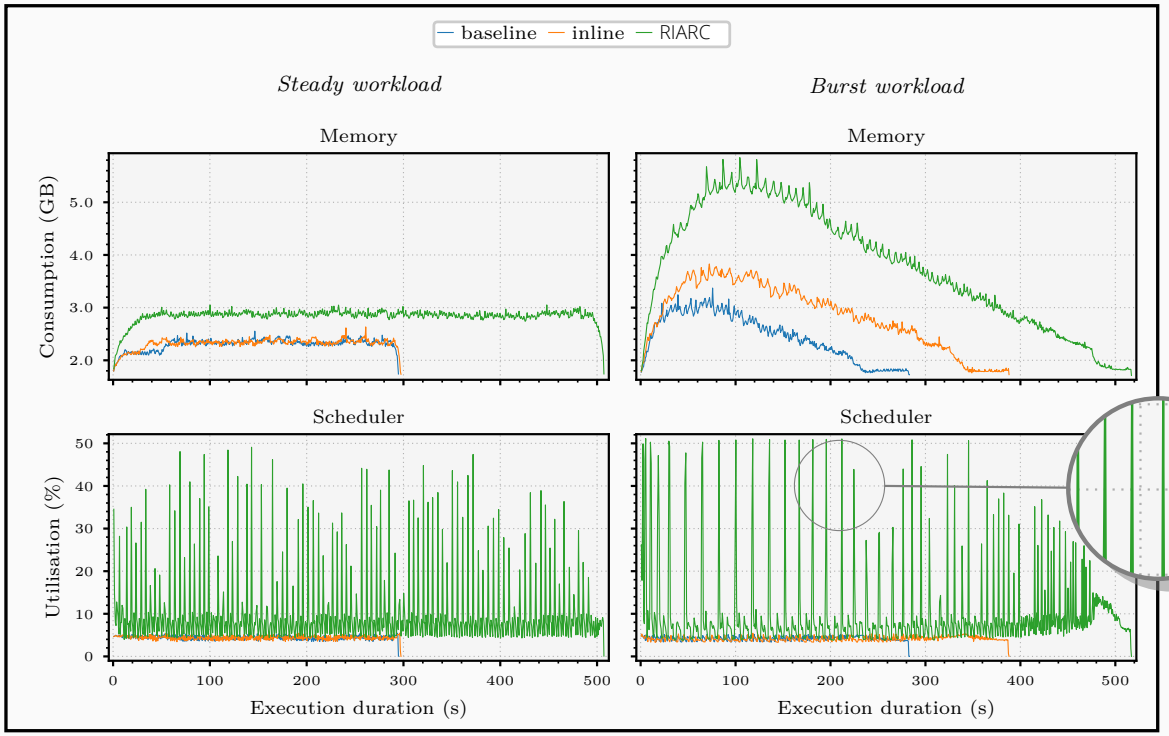
RIARC

Tracer **reconfiguration** evident in scheduler oscillations

Aim: study scalability and elasticity of monitoring

Resource usage

(complete system runs)



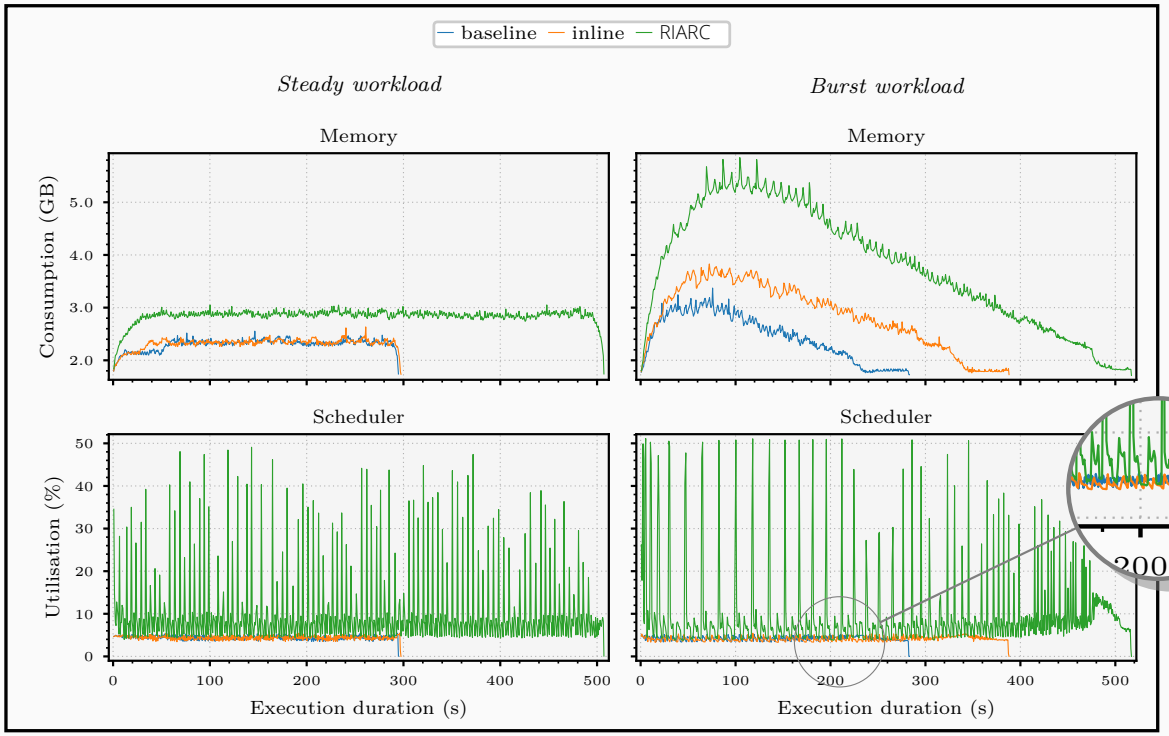
RIARC

Tracer **activation** is shown as peaks

Aim: study scalability and elasticity of monitoring

Resource usage

(complete system runs)



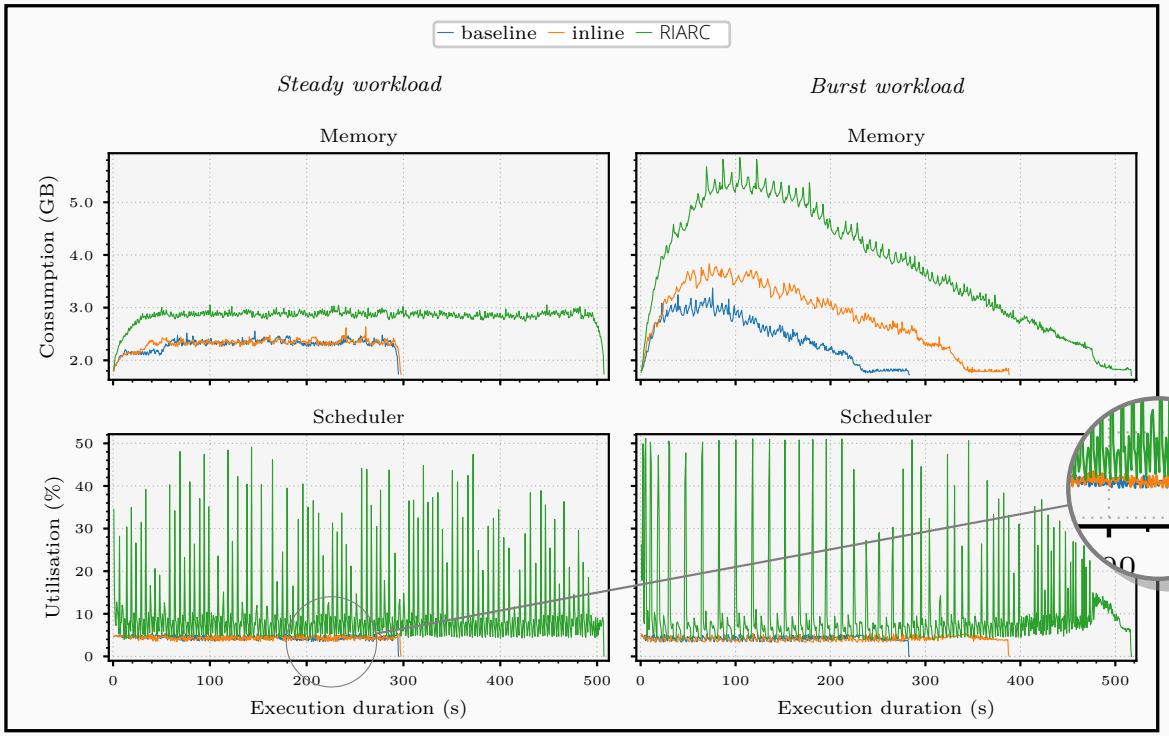
RIARC

Tracer **idle** period is shown as troughs = **message driven**

Aim: study scalability and elasticity of monitoring

Resource usage

(complete system runs)



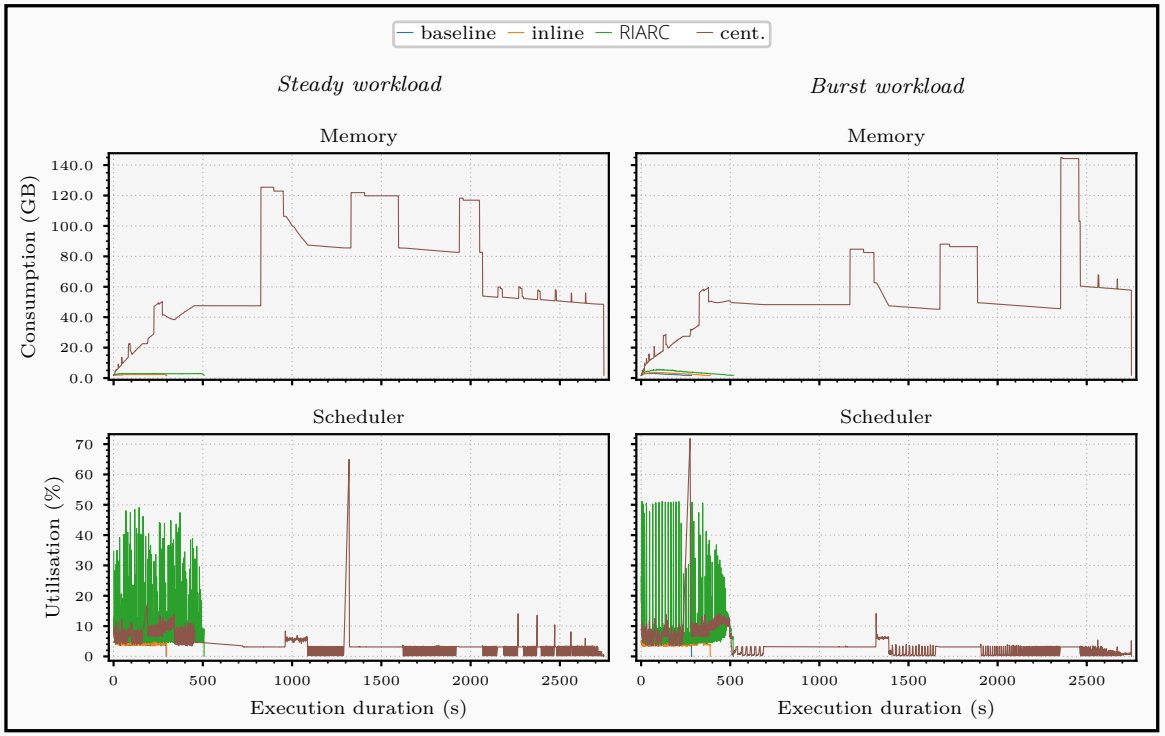
Inline

Minimal oscillations due to **lock-step** execution with system

Aim: inline and RIARC vs. centralised monitoring

Resource usage

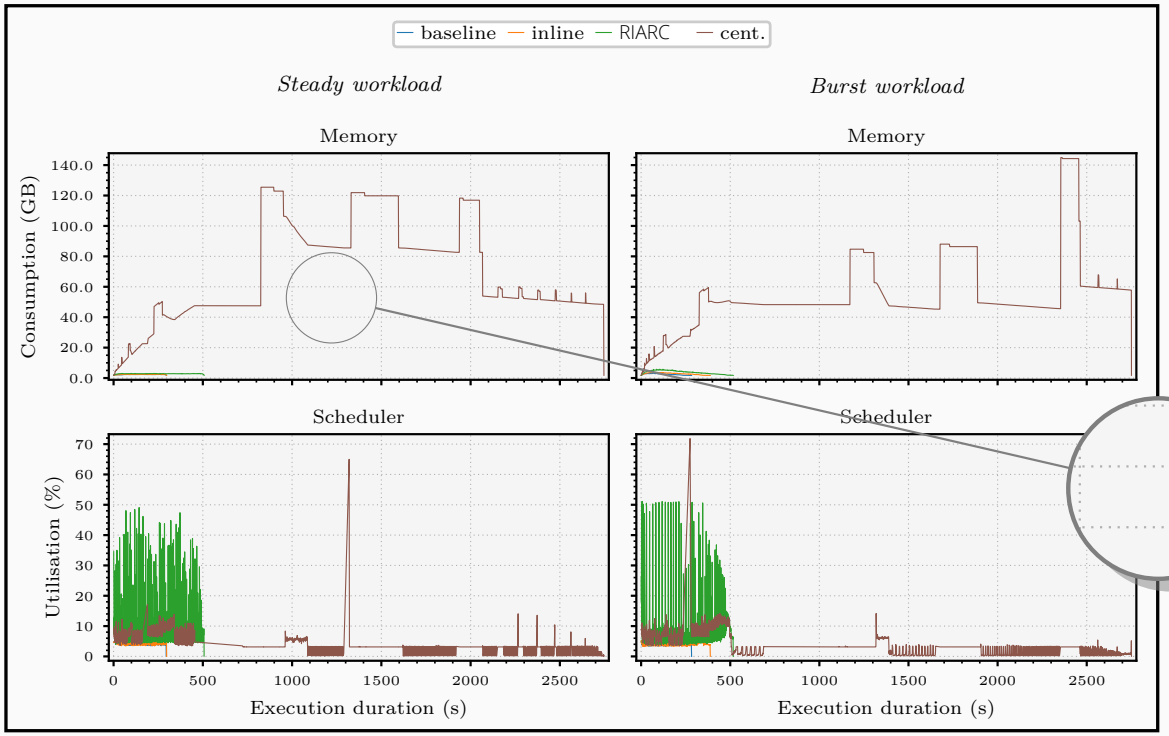
(complete system runs)



Aim: inline and RIARC vs. centralised monitoring

Resource usage

(complete system runs)



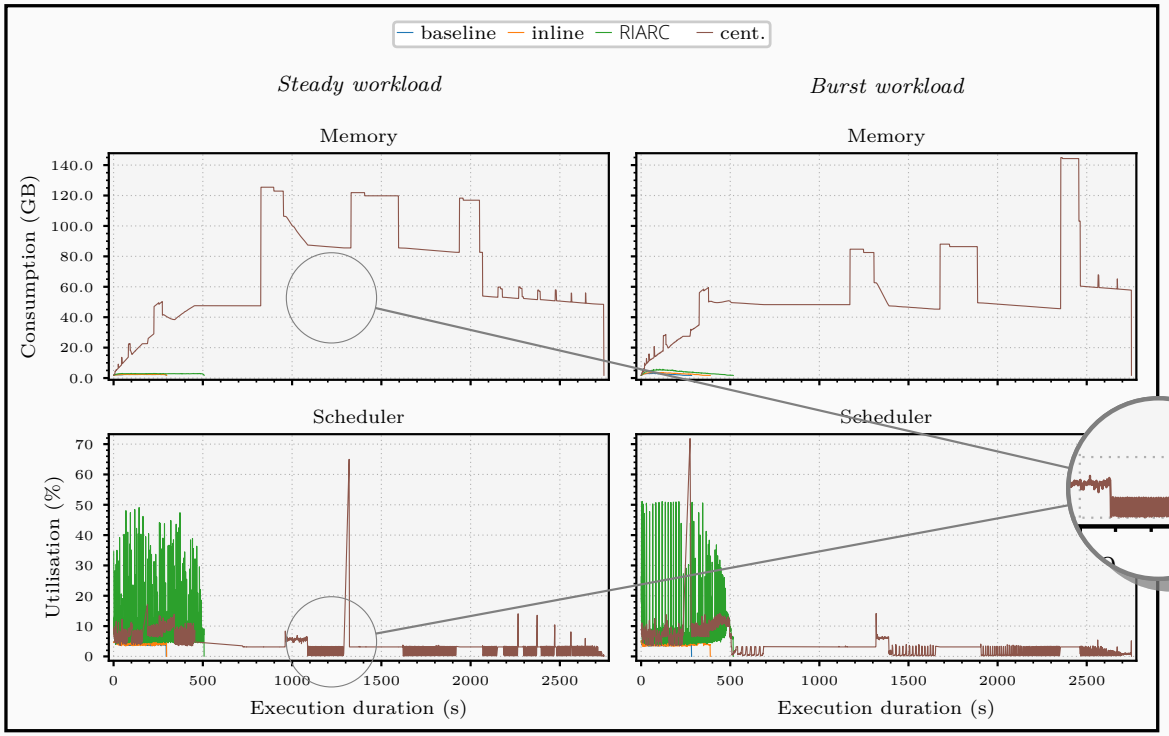
Centralised

Insensitive to workload shape

Aim: inline and RIARC vs. centralised monitoring

Resource usage

(complete system runs)



Centralised

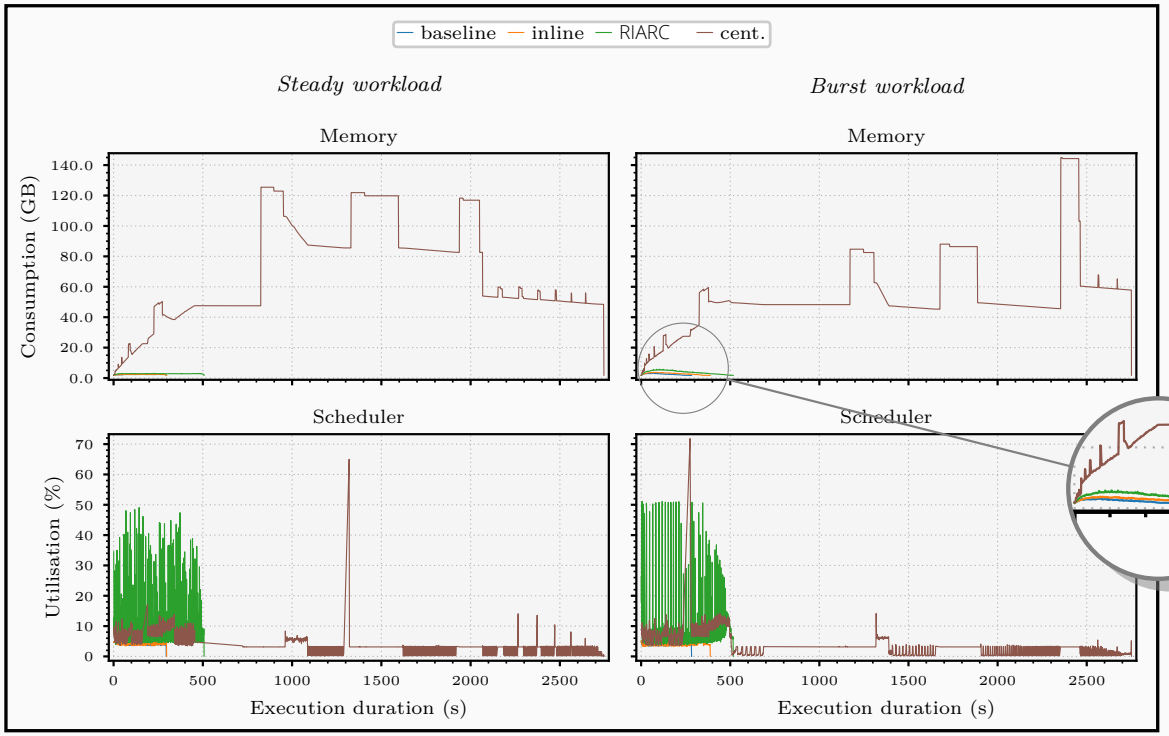
Does **not scale**

(*cf.* RIARC)

Aim: inline and RIARC vs. centralised monitoring

Resource usage

(complete system runs)



Inline RIARC

Dwarfed by memory requirements of centralised