System Resource Utilization Monitoring for Docker Containers

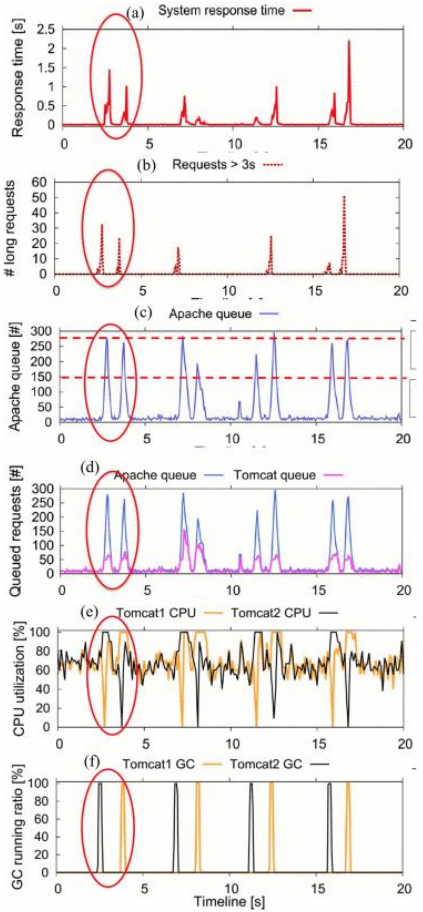
By: Joseph Azevedo
and Bhanu Garg

Motivation

- Increasing adoption of containerization technologies
  - 55% in 2017 to 87% in 2019
- Many solutions for monitoring containers
- Lack of tooling with **high granularity** and **low overhead**

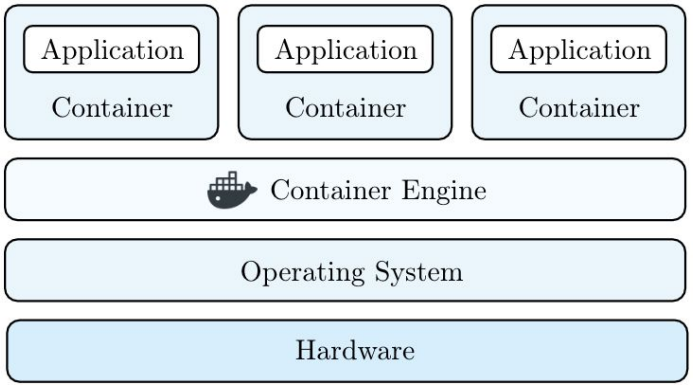
Millibottlenecks

- Brief resource bottlenecks that propagate through an application system
  - Short intervals
- Become amplified due to dependencies and their effect on downstream components
- Long tail problem
- Example: Google reported a loss in up to 20% of revenue from latencies > 500 milliseconds



Docker

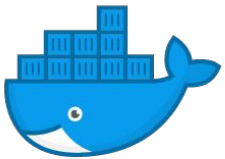
- Package applications with their process-level configuration and dependencies
- Images can be instantiated as containers on a virtual environment
- Uses host OS’s kernel



Project Evolution

- Original goal was to adapt the WISE microblog benchmark to run under Kubernetes and Docker
- Began to examine monitoring tools
  - Docker stats
  - Sysdig
  - cAdvisor
  - Moby

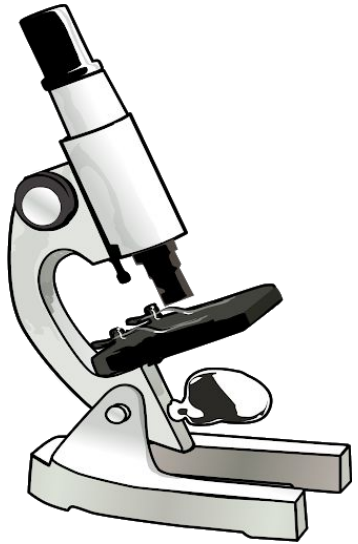
kubernetes



docker

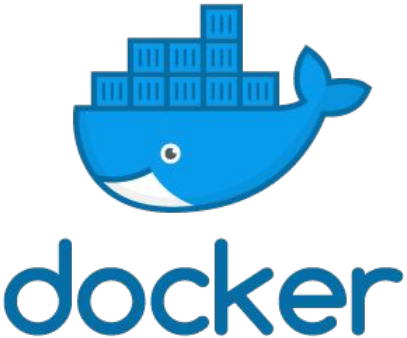
Monitoring Tools

- **High Granularity**
  - consistent under load
  - precision of sampled data collected in each interval
- **Low Overhead**
  - low impact on application performance



Docker Stats

- Native to Docker
- Polls the Docker engine every 1 second
  - Can not be decreased
- Can not achieve high granularity



cAdvisor

- CLI tool developed by Google
- Geared toward system administrators
- Operates as a background daemon
  - Collects statistics about running containers
- Output log provides high granularity
  - Excessive monitoring leads to high overhead

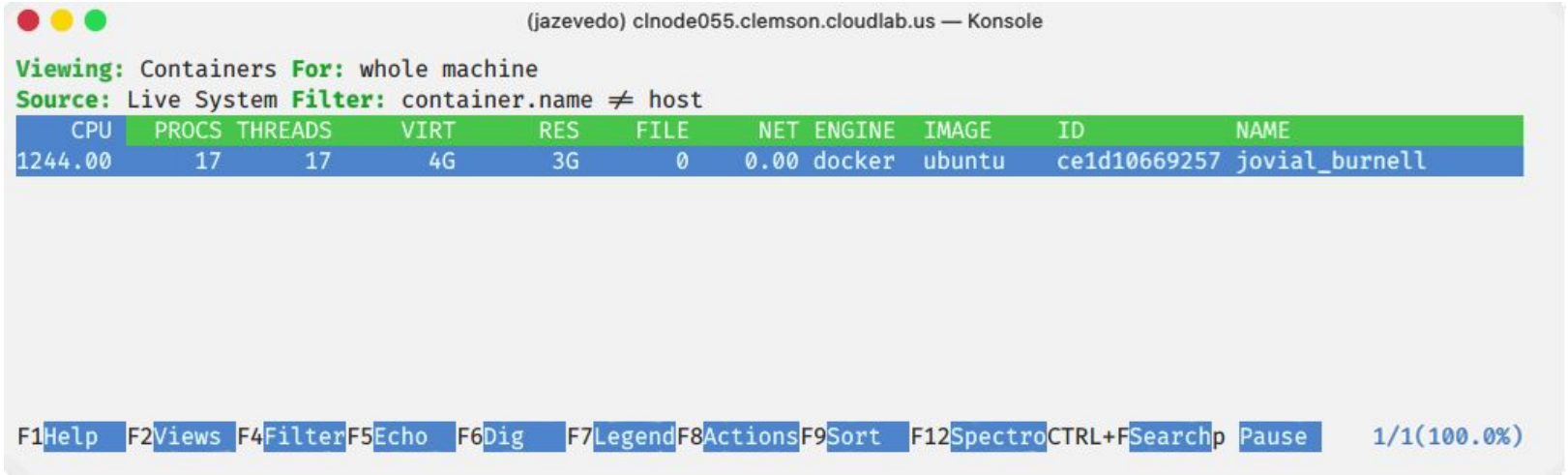
cAdvisor Output


```
cName=/system.slice/system-modprobe.slice host=localhost:8086
memory_usage=0 memory_working_set=0 rx_bytes=0 rx_errors=0
tx_bytes=0 tx_errors=0 timestamp=1583873824045024464
cpu_cumulative_usage=1435685
cName=/system.slice/systemd-logind.service host=localhost:8086
timestamp=1583873824056061533 cpu_cumulative_usage=4974510763
memory_usage=1798144 memory_working_set=1564672 rx_bytes=0
rx_errors=0 tx_bytes=0 tx_errors=0
```

Sysdig

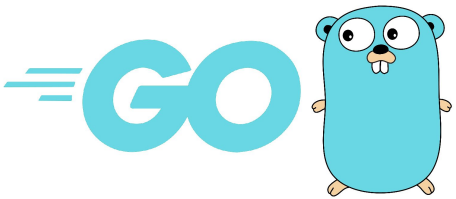
- Curses based CLI tool
- Translating the outputs to a log file is difficult
- Collection interval changing does not work
    - Limited granularity

(C)Sysdig Output



Moby

- Open-source framework that is part of the core Docker engine
  - Built in Golang
- Modified Moby fork to add configuration
  - Allowed to control collection granularity
- Drawbacks
  - Adds complexity, hard to extend



Implementation

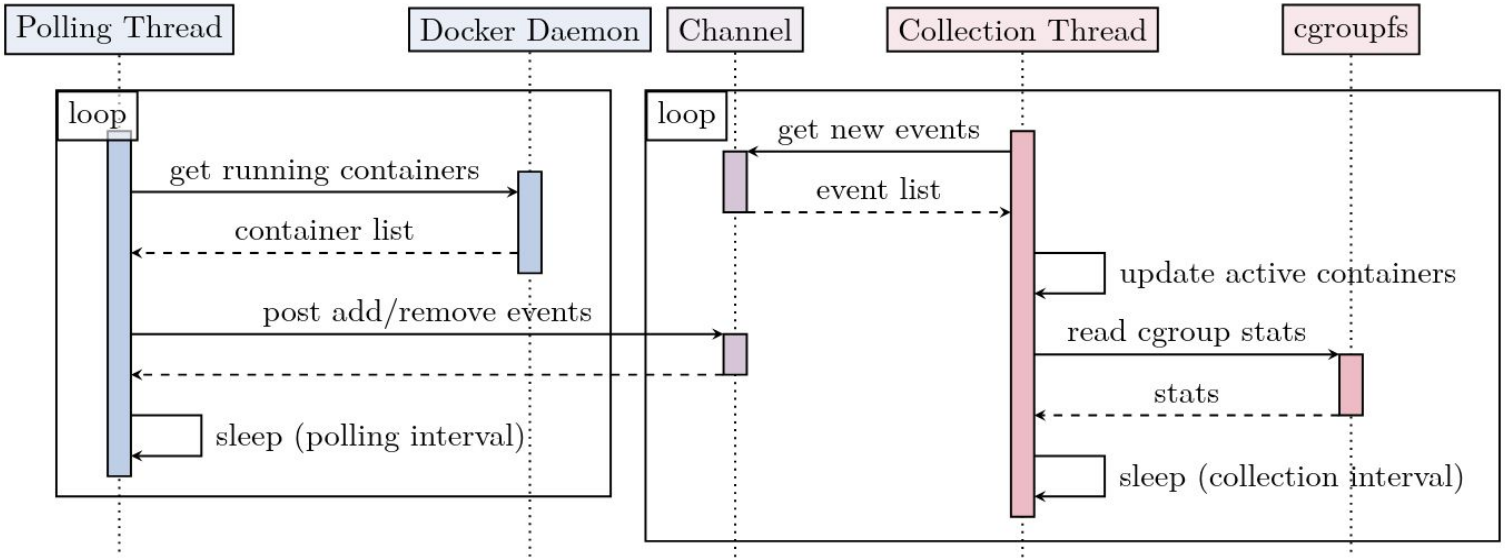
rAdvisor

rAdvisor

- Built in Rust
  - similar performance to C allows for lower overhead
  - no garbage collection allows for consistency in sampling
- Collects statistics for active containers at the specified collection interval
- Uses Linux cgroups to monitor containers
  - Operates efficiently to minimize overhead

rAdvisor Structure

rAdvisor



Demo

Evaluation

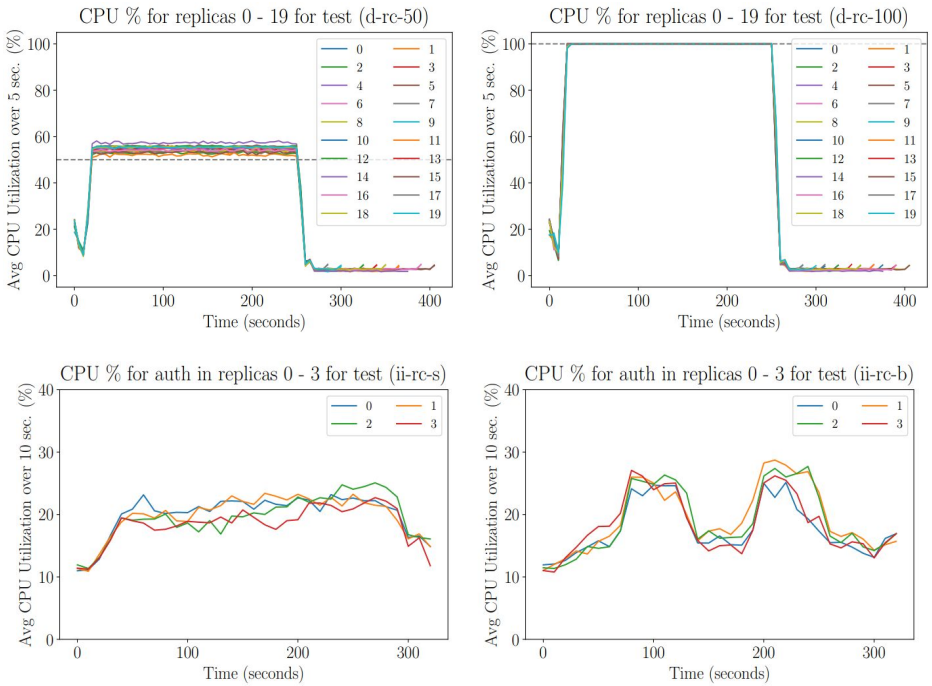
rAdvisor vs Moby

- Primary criteria
  - Low overhead
  - High granularity
- Developed list of specific requirements to meet these
  - Low measurable performance impact when running
  - Consistent, low observed collection intervals

rAdvisor

Experiments

- 3 different experiment configurations
  - 2 synthetic
  - 1 using WISE microblog benchmark
- Used experimental matrix
- Ran 30 replicas of each config



Tooling

- Experimental workflow scripts
- Automation scripts

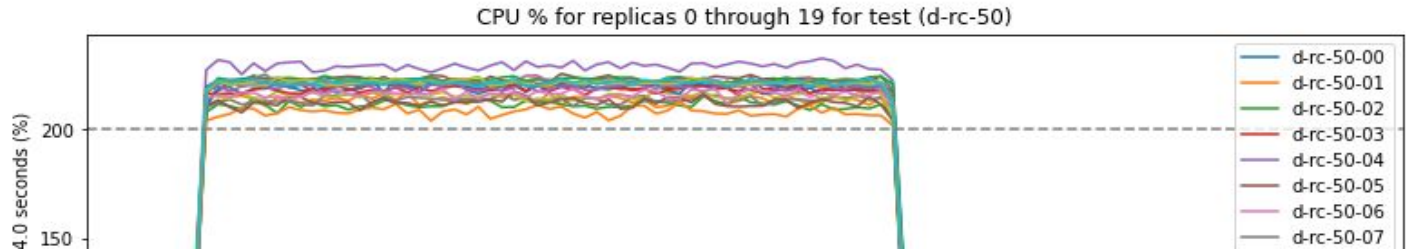

```
[ INFO 13:20:06.241] [ii-rc-s-24] Instantiating experiment ii-rc-s-24
(9fbbd8df-87c0-11ea-b1eb-e4434b2381fc) profile = MicroblogBareMetalC8220
[DEBUG 13:20:06.266] [ii-rc-s-24] Waiting for experiment to become ready
[ INFO 13:31:51.361] [ii-rc-s-24] Successfully provisioned new experiment from
cloudlab: MicroblogBareMetalC8220 (9fbbd8df-87c0-11ea-b1eb-e4434b2381fc) profile =
ii-rc-s-24
                                  | clnode075.clemson.cloudlab.us...
[ INFO 13:31:51.362] [ii-rc-s-24] Using working/ii-rc-s-24 as the working directory
[ INFO 13:31:51.362] [ii-rc-s-24] Wrote rendered config file to
working/ii-rc-s-24/config.sh
```

Parsers and Analysis

- Python parsing scripts
- Jupyter Notebooks
  - Iterative analysis
  - Easy visualization

[9]

```
block_size = 20
examine_single_replica_test("d-rc-50", replica_block_size=block_size, sampling_period=4.0, peak=200)
```



Results

Over 360 experiments ran

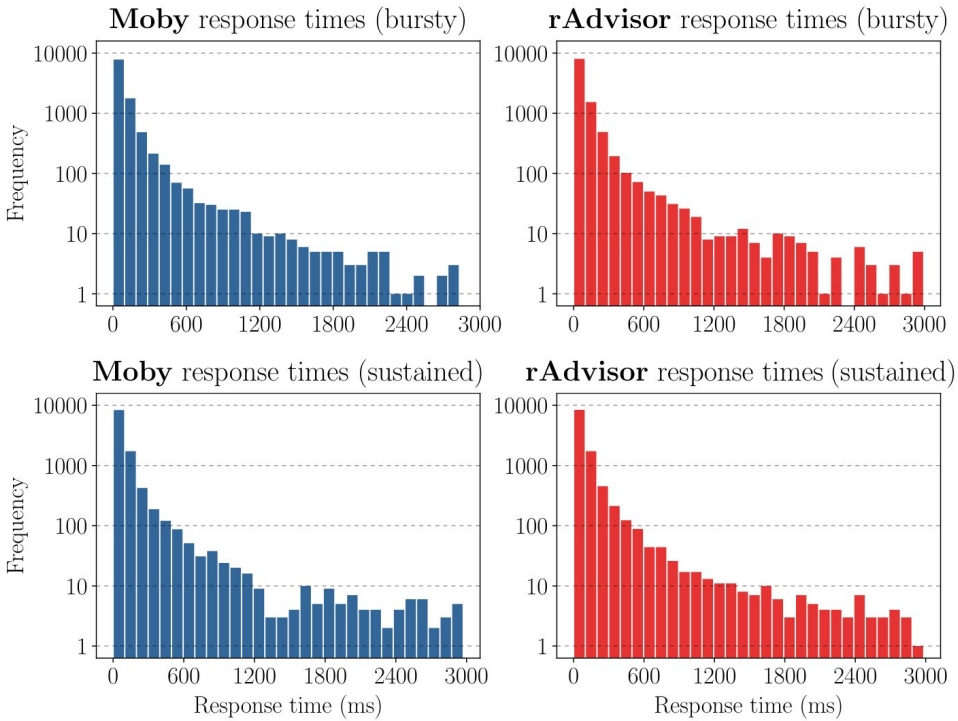
Synthetic Test Results

- Direct CPU utilization comparisons
  - rAdvisor had significantly less performance overhead than Moby under **load**
  - Moby has less performance overhead than rAdvisor at idle
    - This isn't as important
- Synthetic indirect test results
  - Moby had less of an indirect effect on CPU throughput in nbench benchmark

	50 % peak CPU utilization		
Aggregate Score	**Control**	**Moby**	**rAdvisor**
memory	29.16	−0.72 %	−2.03 %
integer	26.49	−0.77 %	−1.92 %
floating point	41.54	−0.74 %	−2.00 %

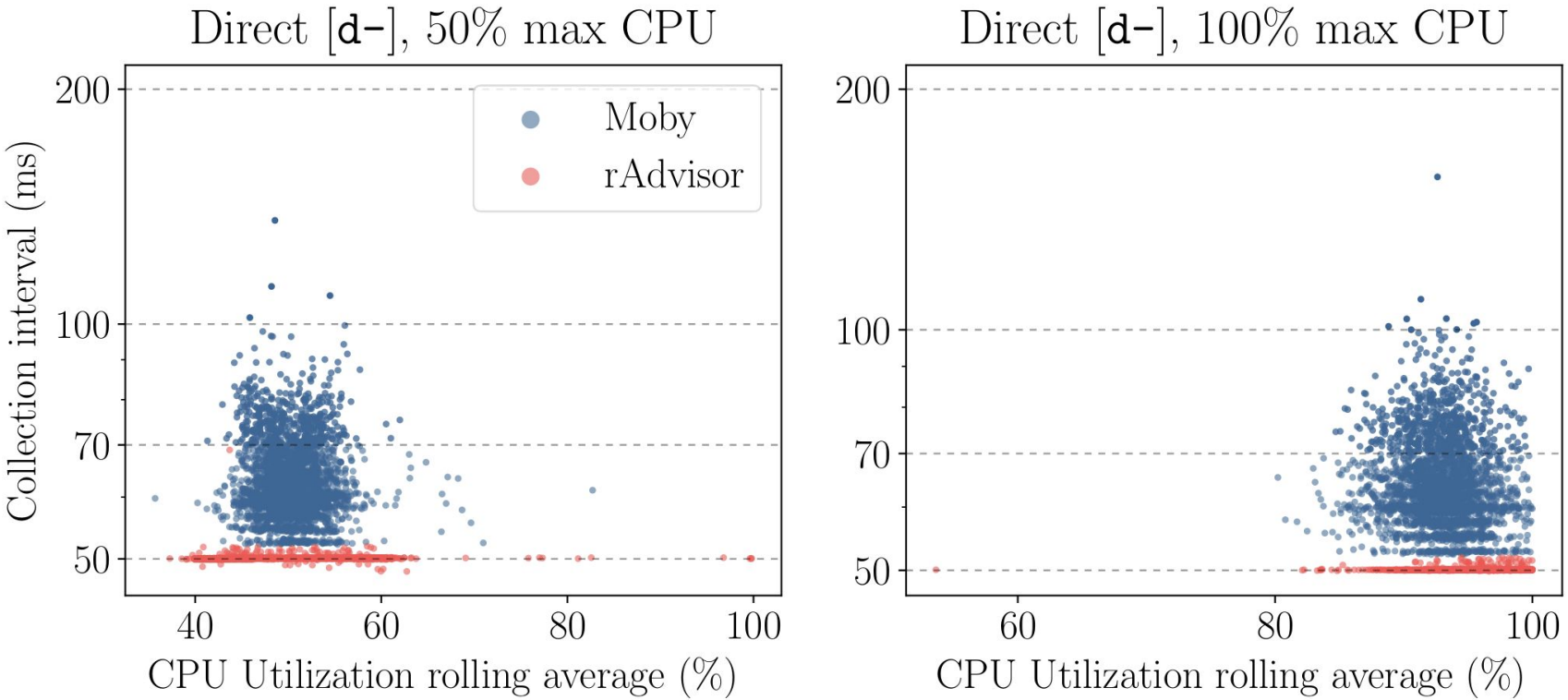
	100 % peak CPU utilization		
Aggregate Score	**Control**	**Moby**	**rAdvisor**
memory	29.17	−0.78 %	−2.09 %
integer	26.54	−0.77 %	−1.56 %
floating point	41.62	−0.81 %	−1.65 %

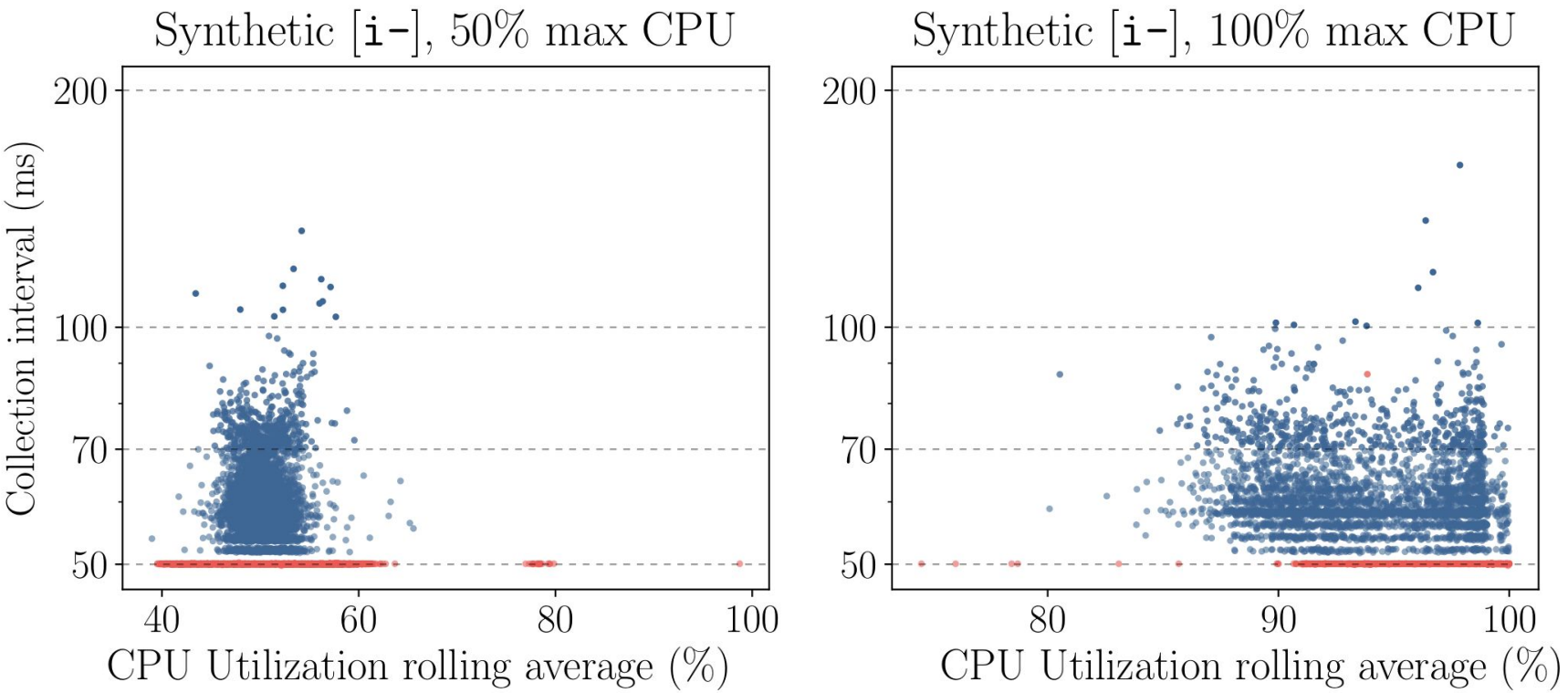
Response Time

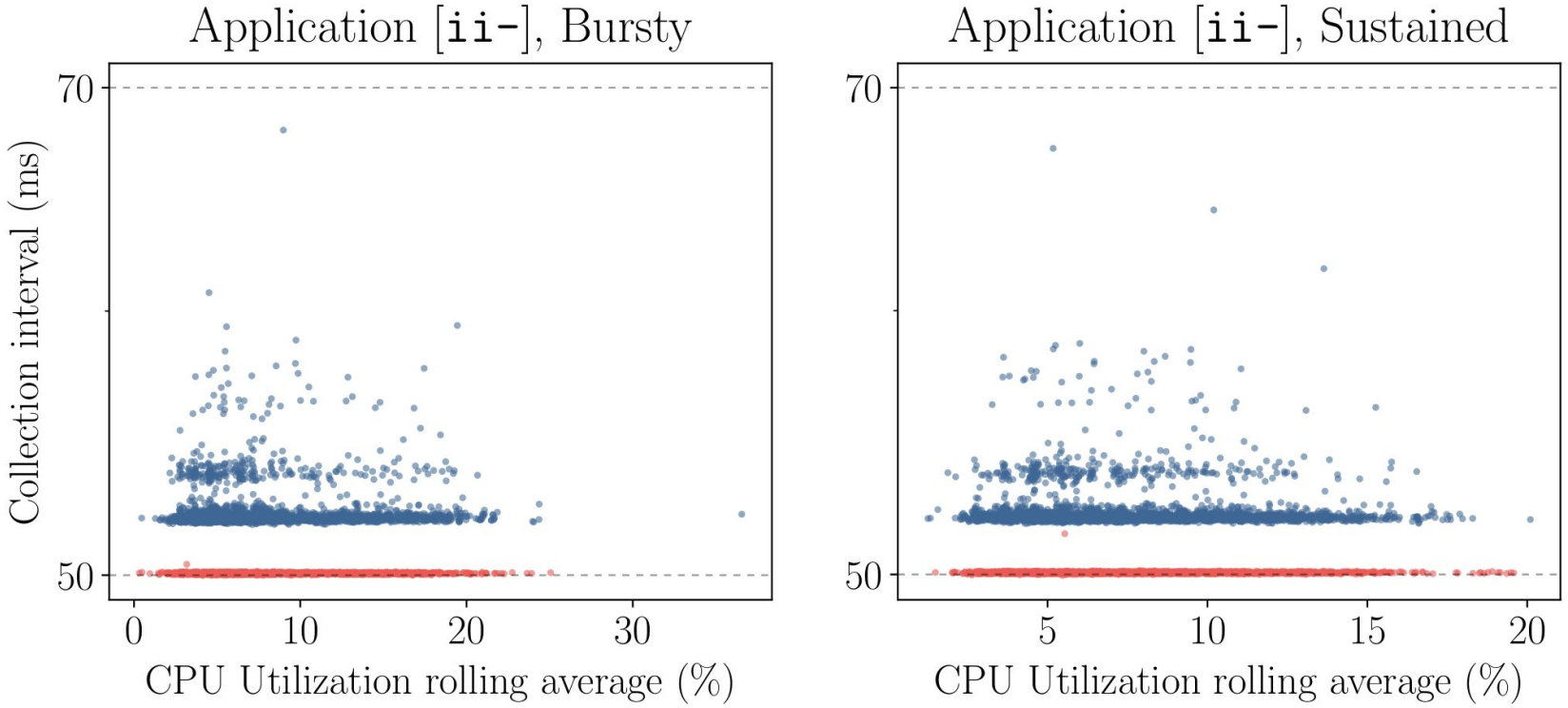


- No noticeable difference between Moby and rAdvisor response times
- Observed emergence of the long tail phenomenon

Observed Collection Intervals

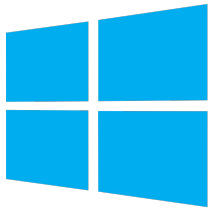



Synthetic [i-], 50% max CPU
Synthetic [i-], 100% max CPU
Collection interval (ms)
200
100
70
50
40
60
80
100
90
CPU Utilization rolling average (%)


Application [ii-], Bursty
Application [ii-], Sustained
Collection interval (ms)
70
50
0
10
20
30
5
10
15
20
CPU Utilization rolling average (%)
CPU Utilization rolling average (%)

Future Work

- Extend/improve experimental workflow
  - Finish adapting WISE microblog benchmark to Kubernetes
  - Test additional configuration options, such as target collection interval
  - Add better synchronization measures in distributed experiments
- Extend/improve rAdvisor
  - Improve performance of polling thread
    - Use lightweight HTTP client or just get the list of active containers by listing the parent docker cgroup folder
  - Make it work in Windows with the Host Compute Services



Conclusion

- Investigated several monitoring tools
- Developed rAdvisor to detect Millibottlenecks
  - Achieved high granularity and low overhead
- Able to maintain at most 100ms collection intervals
  - Over 3 million collection samples
- Experienced low overhead during periods of load