Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Sebastian Serth, Daniel Köhler, Leonard Marschke, Felix Auringer, Konrad Hanff,
Jan-Eric Hellenberg, Tobias Kantusch, Maximilan Paß, Christoph Meinel

Hasso Plattner Institute, University of Potsdam, Germany

Programming Education with MOOCs

OPEN HPI

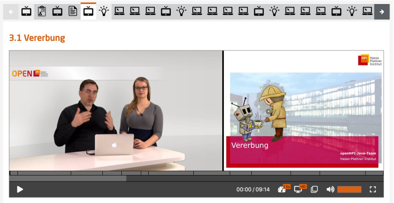
openSAP

mooc.house

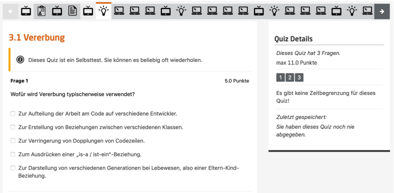
MOOCs

> **175.000 Enrollments**
> in Programming MOOCs

Videos



Quizzes



Programming Exercises

</> CodeOcean

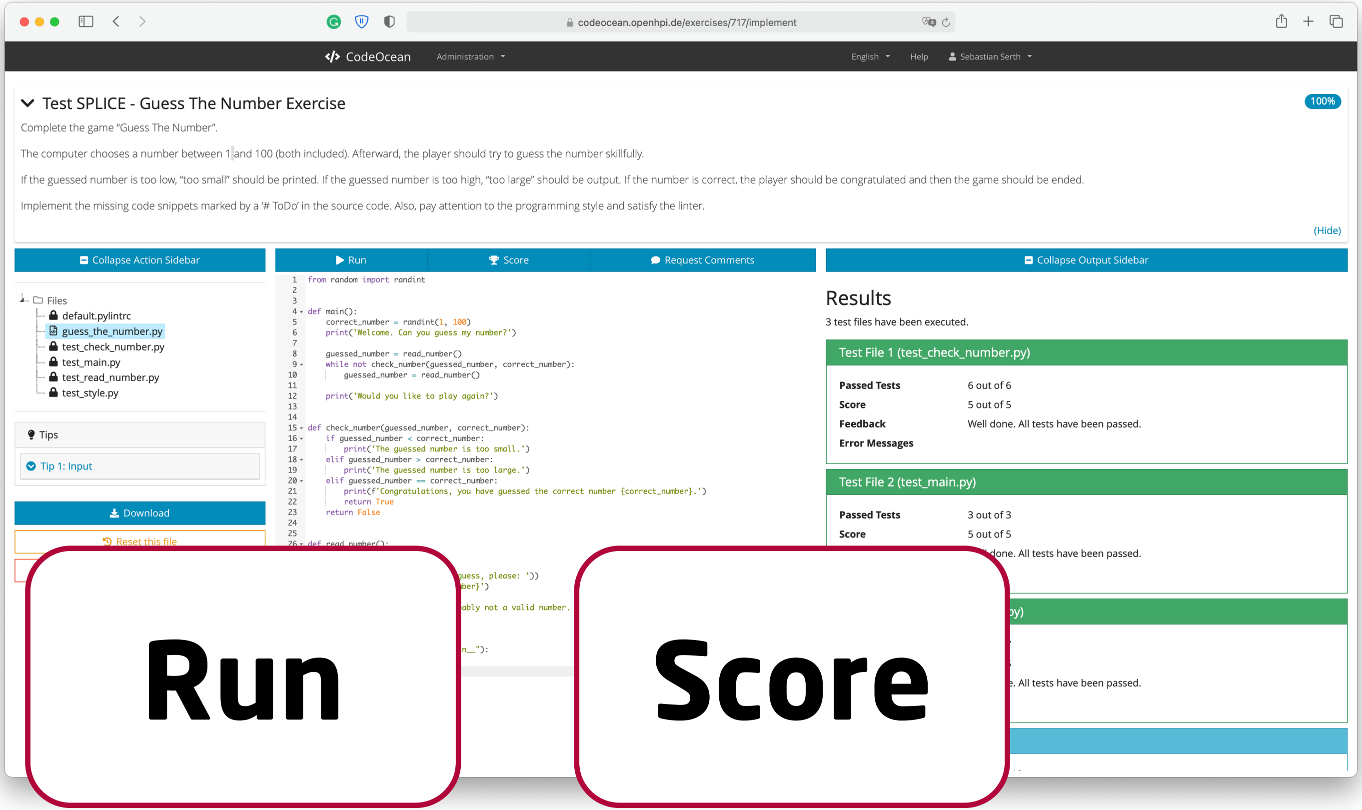
(Staubitz et al. 2016)

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

CodeOcean

Code Execution and Grading in the Browser

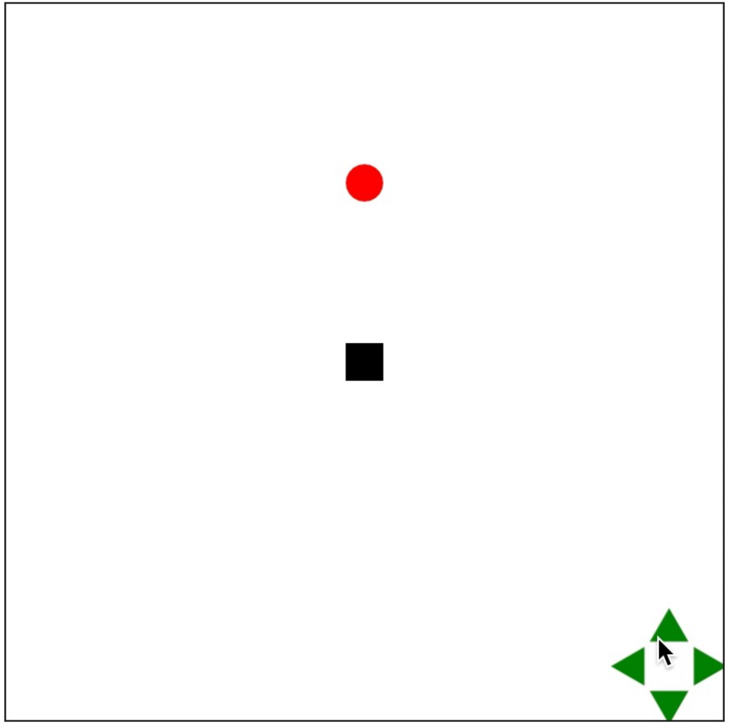


Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

CodeOcean

Interactive Elements and Detailed Score Output



Your guess, please: | Send

`Welcome. Can you guess my number?`

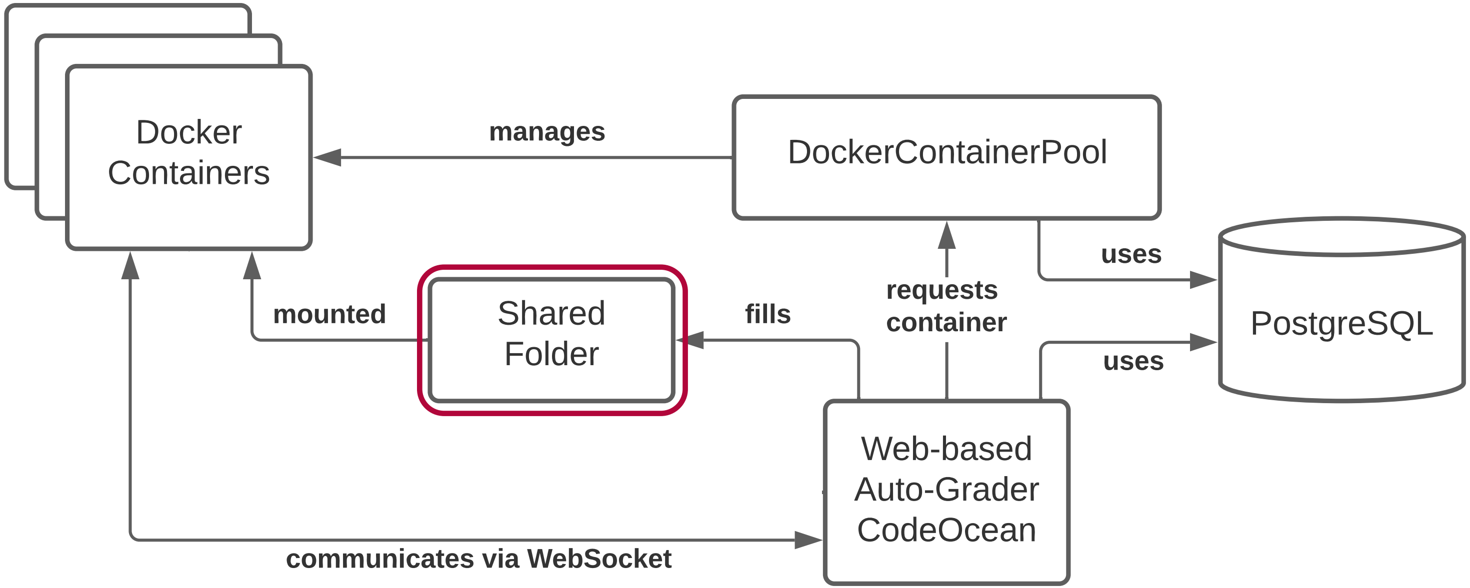
Linter Feedback (test_style.py)

Code Rating	4.64 out of 10
Score	0.46 out of 1
Feedback	Es gibt noch einige Hinweise vom Linter.

Messages

- convention:
  - superfluous-parens: Unnecessary parens after 'if' keyword (aufgabe314.py: 27)
  - wrong-import-position: Import "from turtle import *" should be placed at the top of the module (aufgabe314.py: 18)
  - multiple-statements: More than one statement on a single line (aufgabe314.py: 25, quadrat())
- error:
  - function-redefined: function already defined line 2 (aufgabe314.py: 5, dreieck())
- warning:
  - redefined-outer-name: Redefining name 'winkel' from outer scope (line 1) (aufgabe314.py: 6, dreieck())
  - redefined-builtin: Redefining built-in 'len' (aufgabe314.py: 14, dreieck())
  - unreachable: Unreachable code (aufgabe314.py: 17, dreieck())
  - unused-variable: Unused variable 'len' (aufgabe314.py: 14, dreieck())
  - pointless-string-statement: String statement has no effect (aufgabe314.py: 32)
- refactor:

CodeOcean
Current System Architecture



- **Architectural issues** with the shared folder:
  - Difficult horizontal scaling
  - Increased attack surface

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Requirements Analysis
Approach

- User-centered Design Thinking approach:
  (Meinel et al. 2016)
  1. Interviews with different stakeholder representatives
     a) Learners
     b) Teachers
     c) Administrators
  2. Derive personas to visualize user needs and their pain points
  3. Decide on a subset of features to address
  4. Evaluate technical solutions for these personas

- Analysis of past executions on CodeOcean since 2015

Related Work
Security Considerations

Assessing code submissions
(Garmann 2013)

Dynamic code analysis

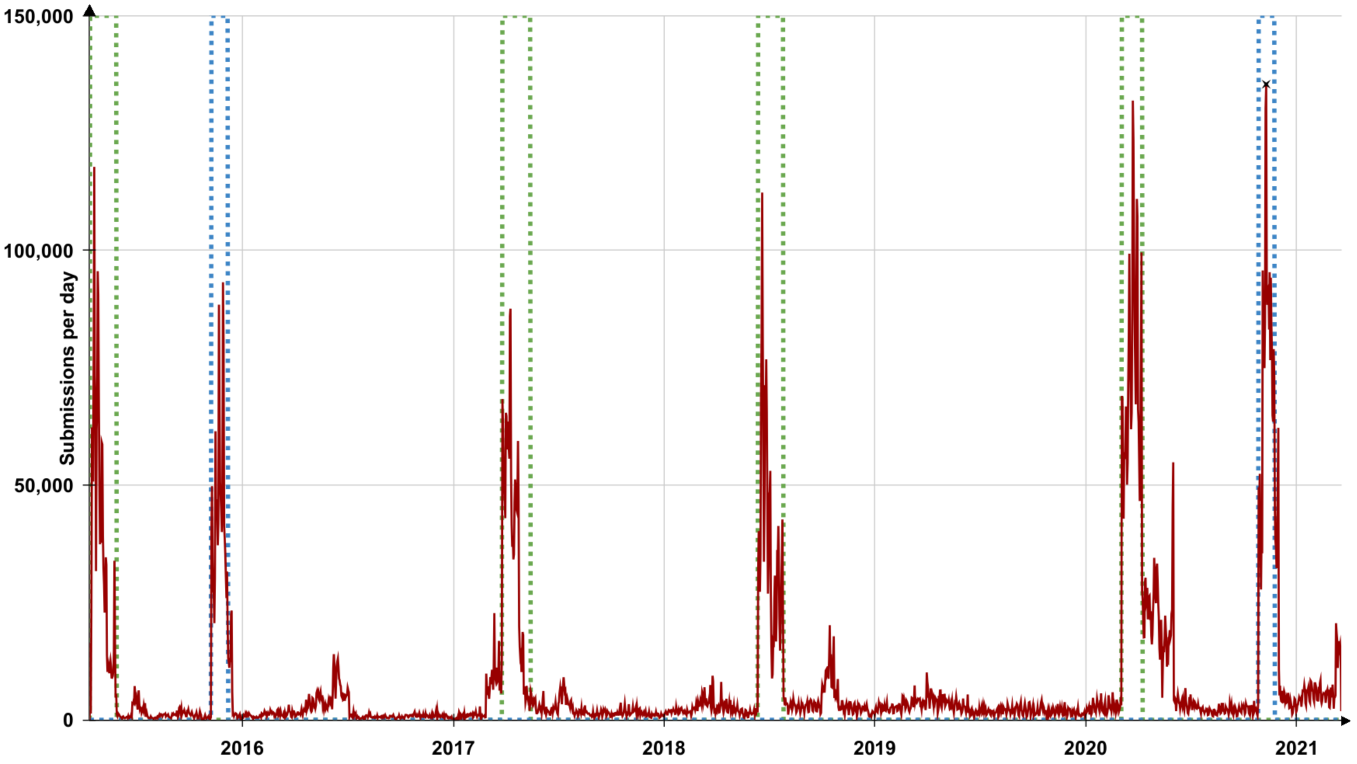
- Code under investigation is executed
  - Security implications
    - Language-specific
      using the Java Security Manager
      (Strickroth 2019)
    - Container-based
      using Docker
      (Breitner et al. 2016)

Static code analysis

- Code inspection without executing the code under investigation

→ Other scalability and security considerations

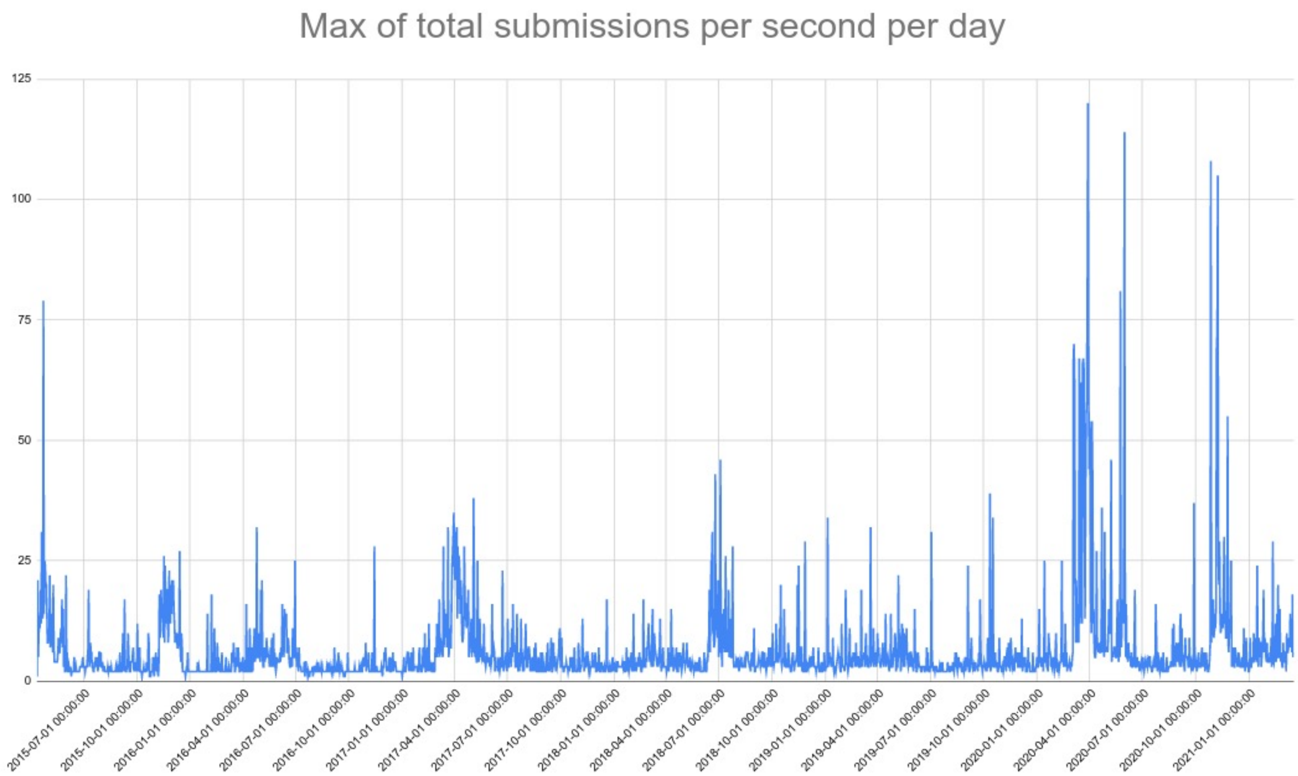
Requirements Analysis
Past Performance



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Requirements Analysis
Past Performance



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Identified Requirements

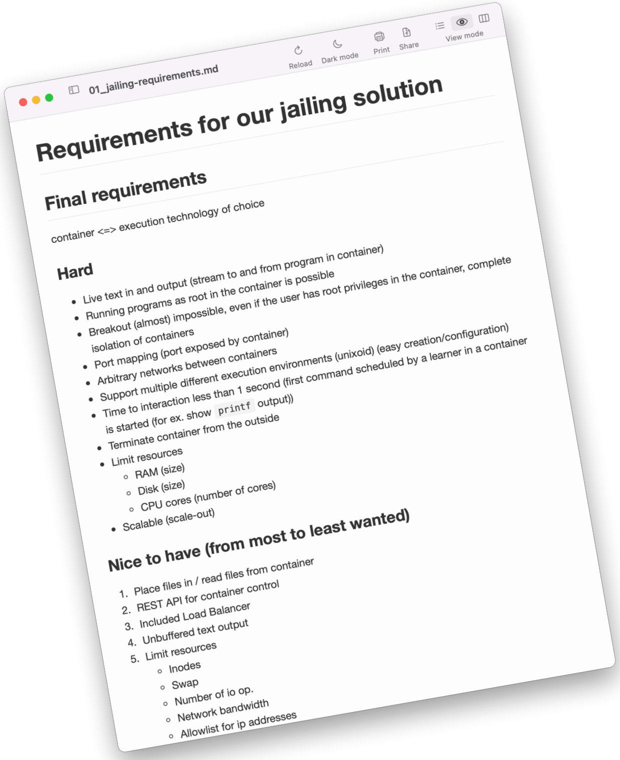
1. **Interactivity**
   - Synchronous channel between the execution and the users’ web browser
   - Real-time code executions
2. **Scalability**
   - Current system handles up to 120 execution requests per second
   - Mean execution time less than 10 seconds per execution
3. **Flexibility**
   - Compatibility to containerd ecosystem (e.g., through a Dockerfile)

Identified Requirements

1. **Interactivity**
2. **Scalability**
3. **Flexibility**

- Limit resources
  - CPU: Wall clock time, CPU shares
  - RAM
  - Network access
- Isolation between different executions
- Desired: Network orchestration with multiple nodes

Evaluation
Criteria


01_jailing-requirements.md

Reload Dark mode Print Share View mode

Requirements for our jailing solution

Final requirements

container <=> execution technology of choice

Hard

- Live text in and output (stream to and from program in container)
- Running programs as root in the container is possible
- Breakout (almost) impossible, even if the user has root privileges in the container, complete isolation of containers
- Port mapping (port exposed by container)
- Arbitrary networks between containers
- Support multiple different execution environments (unixoid) (easy creation/configuration)
- Time to interaction less than 1 second (first command scheduled by a learner in a container is started (for ex. show `printf` output))
- Terminate container from the outside
- Limit resources
  - RAM (size)
  - Disk (size)
  - CPU cores (number of cores)
- Scalable (scale-out)

Nice to have (from most to least wanted)

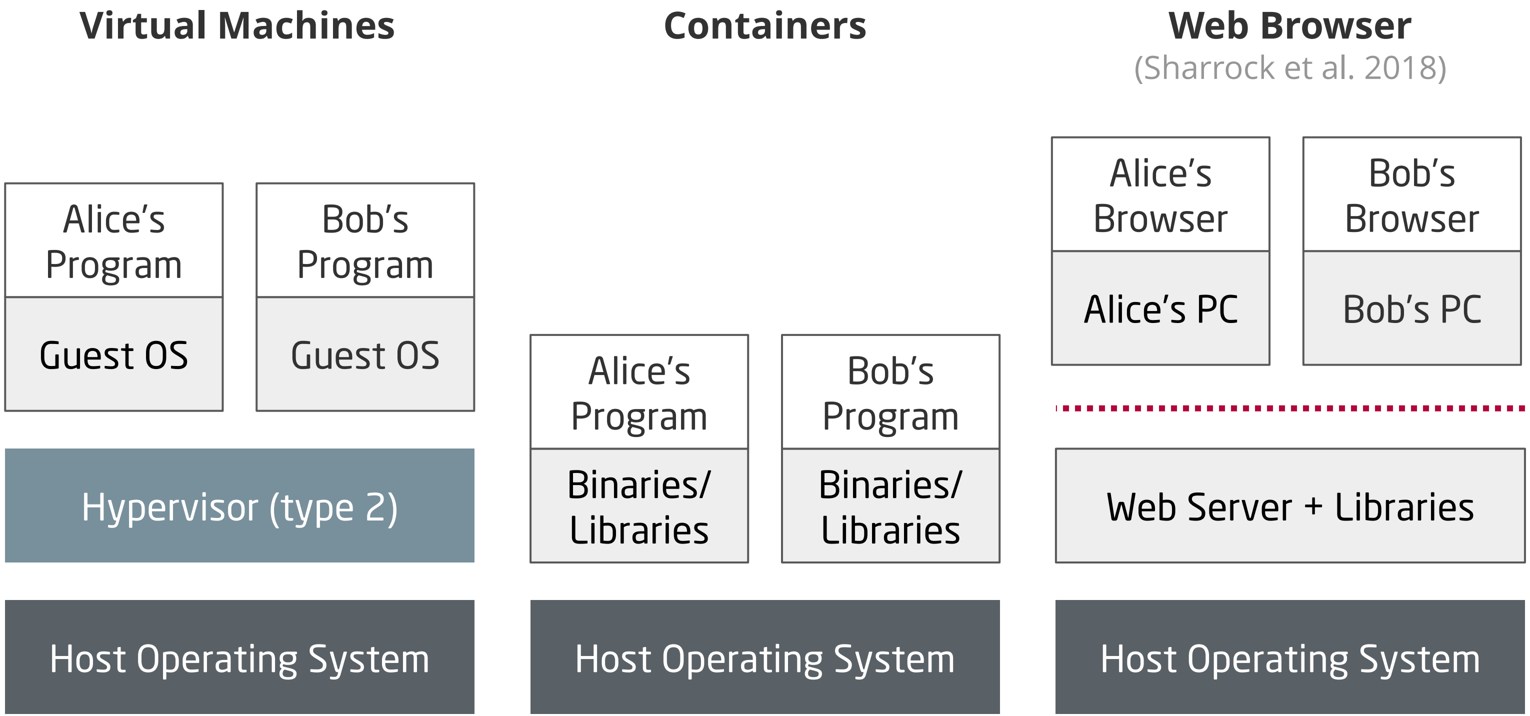
1. Place files in / read files from container
2. REST API for container control
3. Included Load Balancer
4. Unbuffered text output
5. Limit resources
   - Inodes
   - Swap
   - Number of io op.
   - Network bandwidth
   - Allowlist for ip addresses

- **Features**
  - Attach to stdout, stderr, stdin
  - Allow superuser access within the execution
- **Metrics**
  - Startup time
  - RAM / CPU usage
- **Technical details**
  - Maintenance and management
  - Isolation technologies and security implications

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Related Work
Execution Environments



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Execution Environments

Hasso Plattner Institut

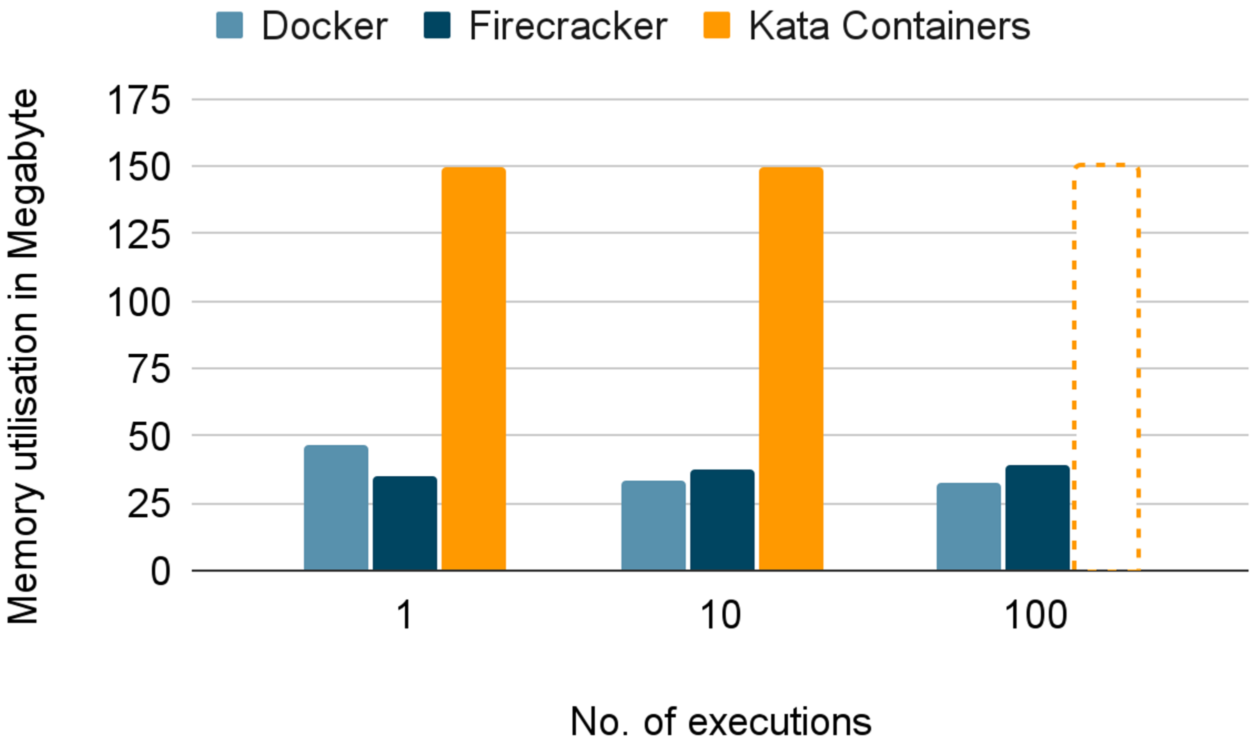
Excluded by the requirements	Approved for performance evaluation
Pouch	docker
rkt	Firecracker
LXC	katacontainers
VirtualBox	

Serth et al.

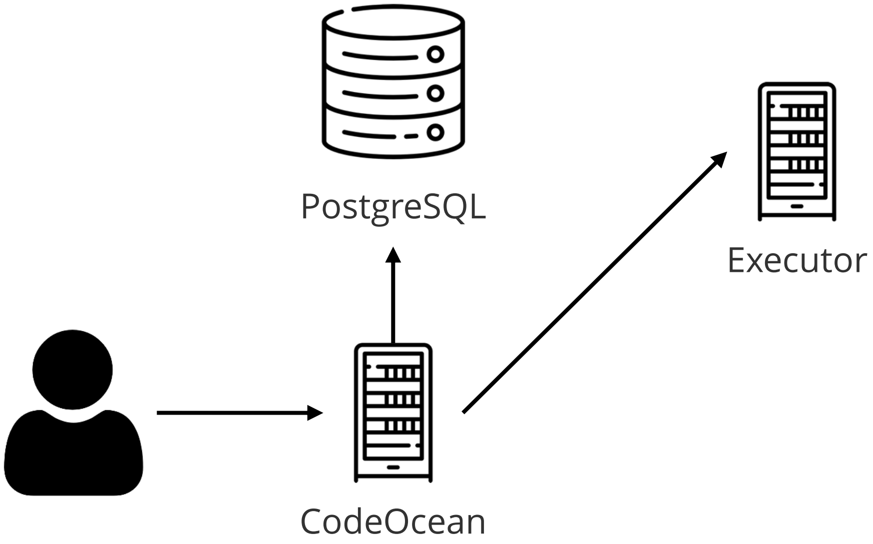
Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Execution Environments
Startup time

Execution Environments
Memory usage



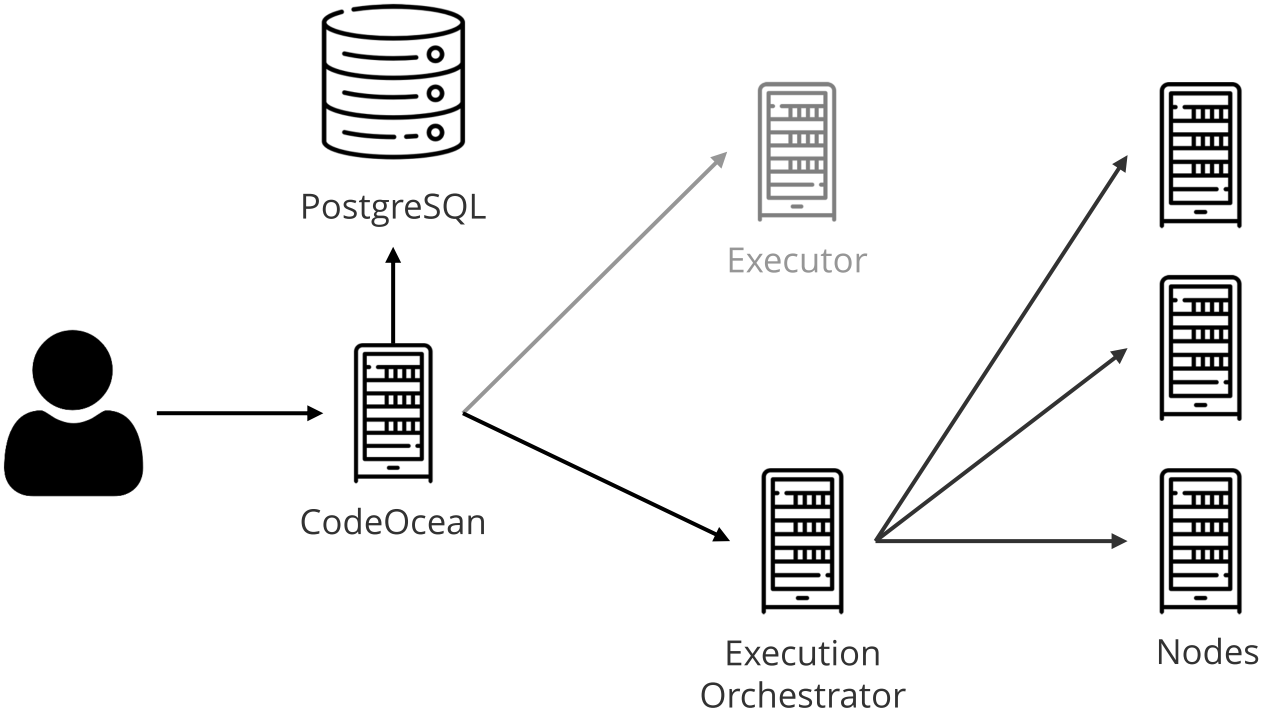
Requirement Evaluation
Scalability



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Requirement Evaluation
Scalability



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Execution Orchestrators

Hasso Plattner Institut

Excluded by the requirements

fission

Kubeless

RANCHER

OPENFAAS

Knative

Approved for performance evaluation

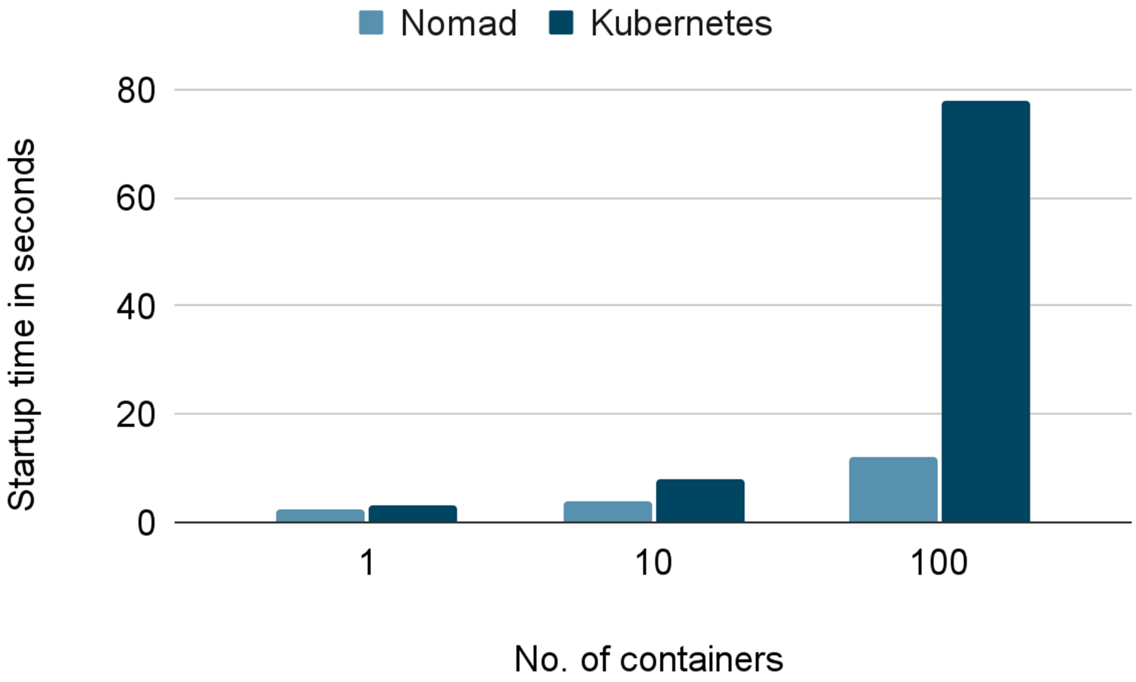
Nomad

kubernetes

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Execution Orchestrators
Nomad vs Kubernetes



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Evaluation
Comparison

Hasso Plattner Institut

Environments

Firecracker

docker

katacontainers

Orchestrators

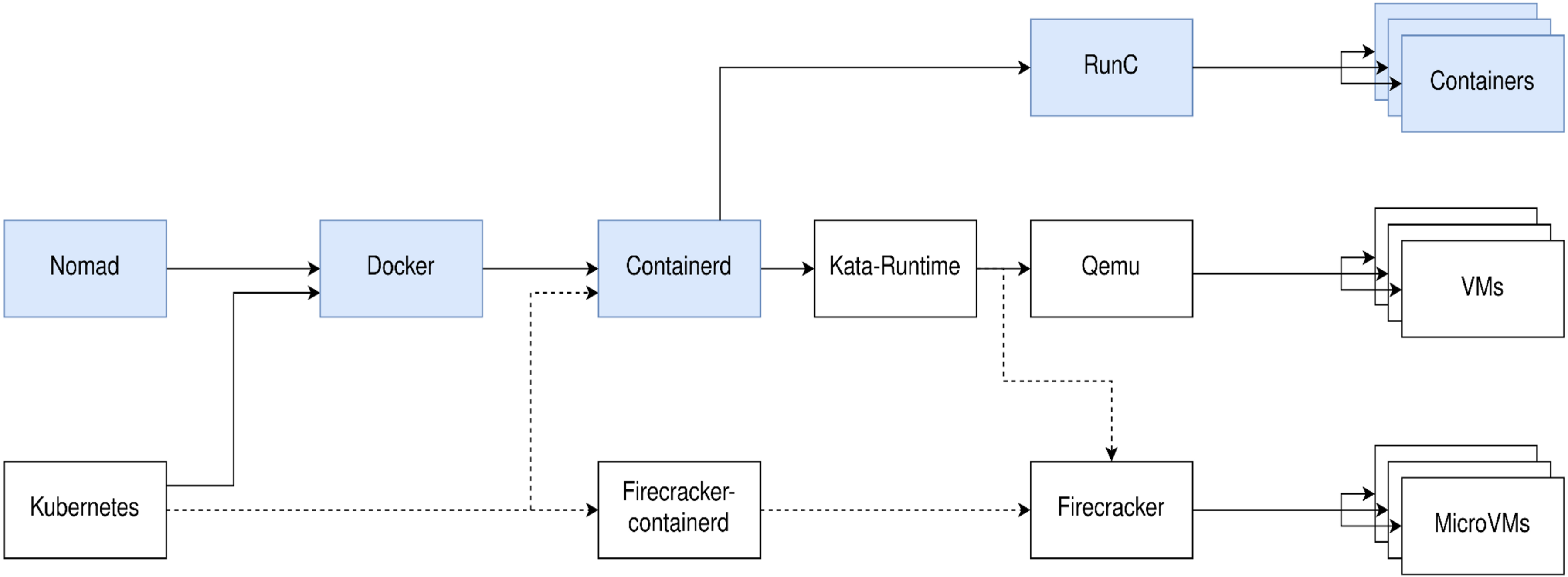
Nomad

kubernetes

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Requirement Evaluation
Flexibility



Evaluation
Comparison

Hasso Plattner Institut

Environments

Firecracker

docker

katacontainers

Orchestrators

Nomad

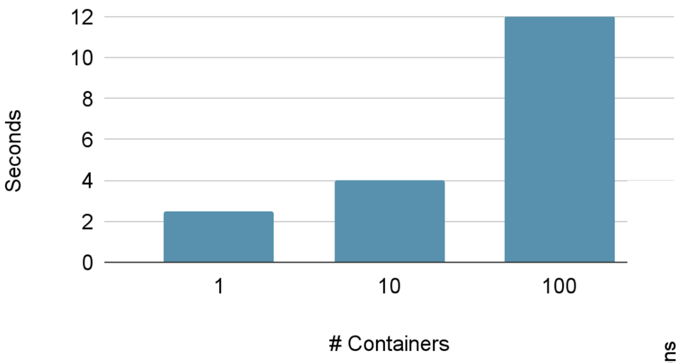
kubernetes

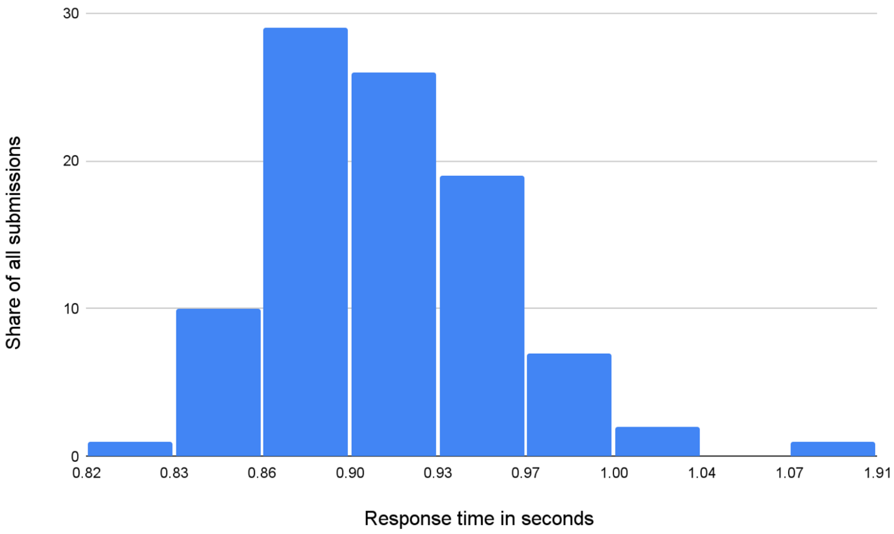
Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Evaluation
Prewarming

Startup time

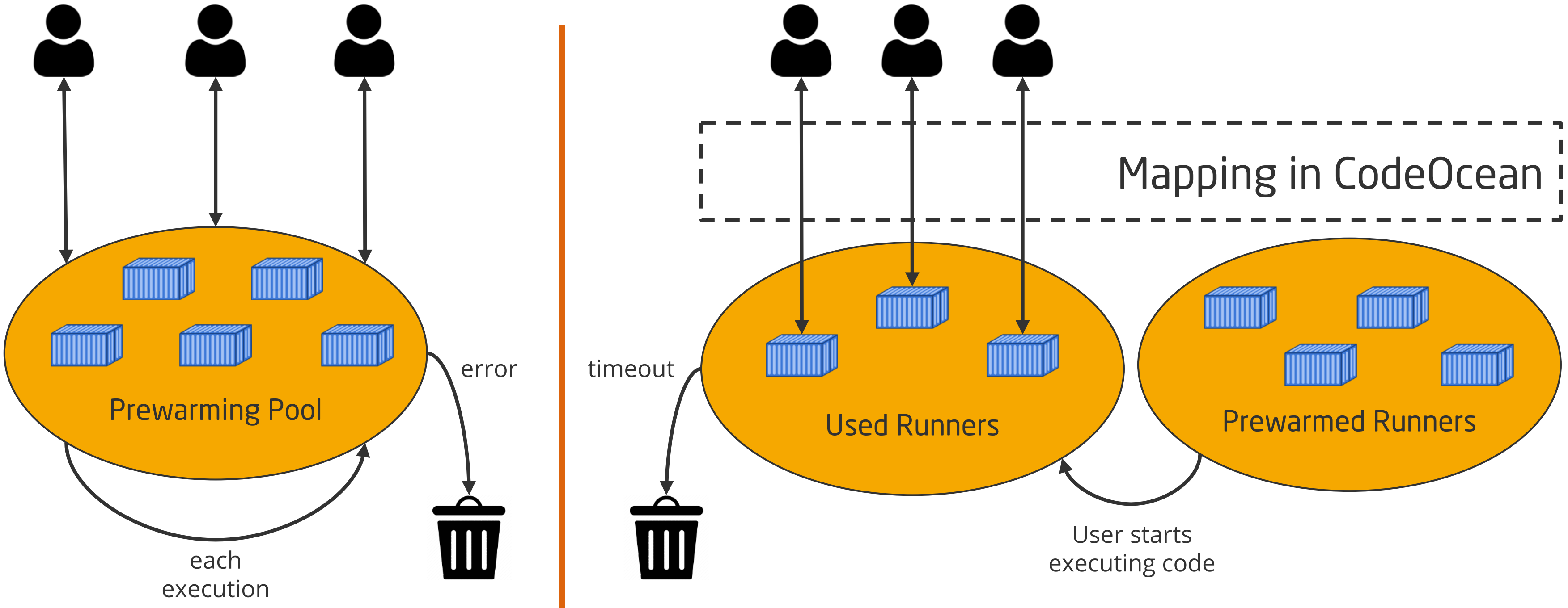




Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

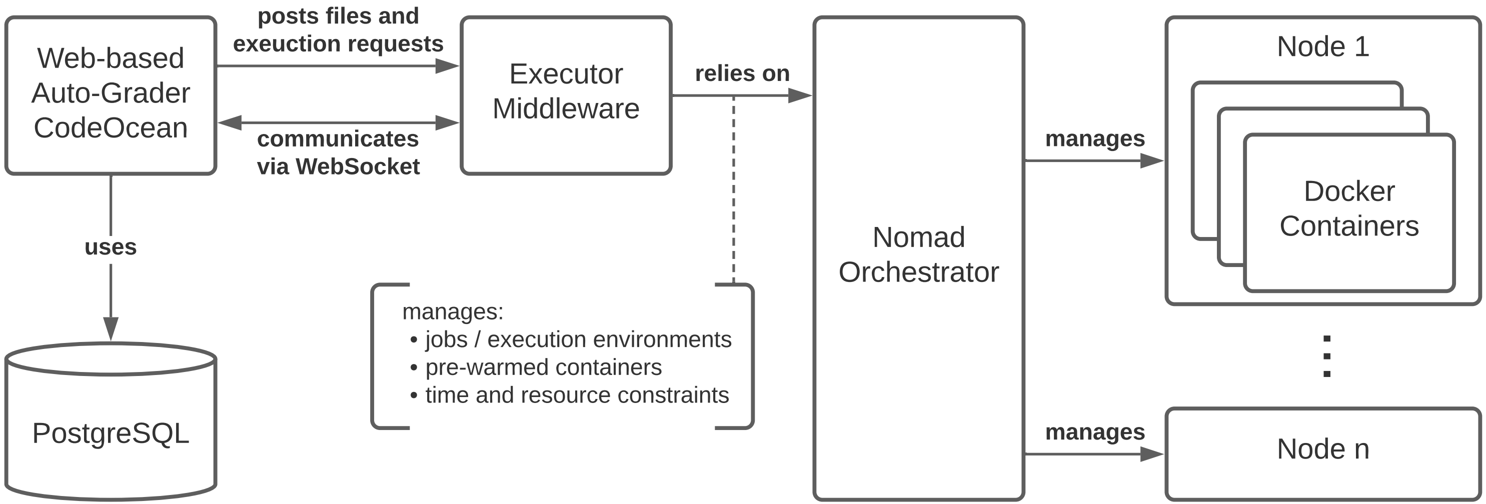
Evaluation
Container Reusing



Old

New

CodeOcean
New System Architecture



Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Future Work

- Integration of hardware resources
  - GPU for machine learning exercises (for teaching purposes at HPI)
  - Raspberry PI for embedded smart home courses on openHPI
- Networking scenarios
  - Multi-node setup and VPN access
- Long lasting executions
  - Debugging
- Pre-warming strategies and Function-as-a-Service (FaaS)
- Comparison of programming languages and their security concepts for system components (especially the executor middleware)

Conclusion

- Identification of three main requirements:

Interactivity	Scalability	Flexibility

- Evaluation of several containerization and orchestration technologies:

Docker
Firecracker > Kata Containers

Nomad > Kubernetes

- Executor middleware abstracts from
  - container management and pre-warming
  - specific interface of the orchestrator

→ **Result: Scalable and more secure architecture for execution environments**

Serth et al.

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Sebastian Serth, Daniel Köhler, Leonard Marschke, Felix Auringer, Konrad Hanff, Jan-Eric Hellenberg, Tobias Kantusch, Maximilan Paß, Christoph Meinel

Hasso Plattner Institute, University of Potsdam, Germany