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

Videos
Quizzes
Programming Exercises

> 175,000 Enrollments
in Programming MOOCs

Serth et al.
Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

(Staubitz et al. 2016)
CodeOcean

Code Execution and Grading in the Browser

Run

Score

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

Serth et al.
Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs

Chart 4
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
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

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
Max of total submissions per second per day

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

- **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
Related Work

**Execution Environments**

### Virtual Machines
- Alice’s Program
- Bob’s Program
- Guest OS
- Hypervisor (type 2)
- Host Operating System

### Containers
- Alice’s Program
- Bob’s Program
- Binaries/Libraries
- Host Operating System

### Web Browser
(Sharrock et al. 2018)
- Alice’s Browser
- Bob’s Browser
- Alice’s PC
- Bob’s PC
- Web Server + Libraries
- Host Operating System

Serth et al.

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

Excluded by the requirements
- Pouch
- rkt
- LXC
- VirtualBox

Approved for performance evaluation
- docker
- Firecracker
- katacontainers

Serth et al.
Improving the Scalability and Security of Execution Environments for Auto-Graders in the Context of MOOCs
Execution Environments

Startup time

Serth et al.

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

Chart 15
Execution Environments
Memory usage

Chart 16
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

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

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**

```
Nomad -> Docker -> Containerd -> Kata/Runtime -> Qemu

Containers

Kubernetes

Firecracker-Containerd

Firecracker

VMs

MicroVMs
```
Evaluation Comparison

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

Serth et al.

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

Container Reusing

Old

- Prewarming Pool
  - Each execution
  - Error

New

- Mapping in CodeOcean
- Used Runners
  - Timeout
- Prewarmed Runners
  - User starts executing code
CodeOcean
New System Architecture

- Web-based Auto-Grader CodeOcean uses PostgreSQL
- posts files and execution requests to Executor Middleware, which communicates via WebSocket
- Nomad Orchestrator relies on Executor Middleware and manages:
  - jobs / execution environments
  - pre-warmed containers
  - time and resource constraints
- Nomad Orchestrator manages Docker Containers and Node n

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
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, Maximilian Paß, Christoph Meinel

Hasso Plattner Institute, University of Potsdam, Germany