Simulation Framework for Research and Education in Distributed Computing

Oleg Sukhoroslov

Kharkevich Institute for Information Transmission Problems

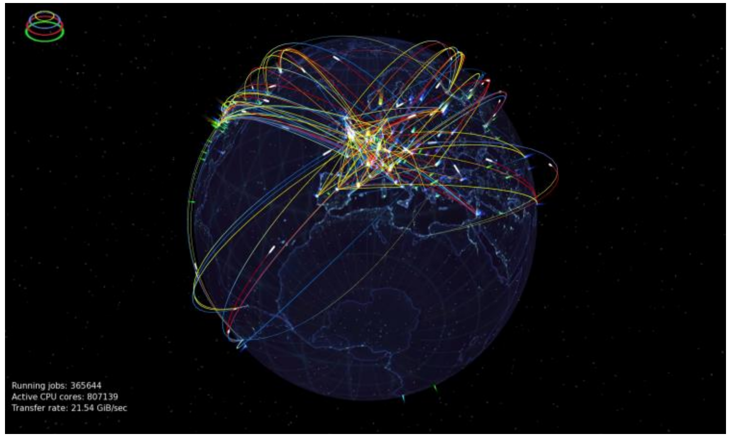
HSE UNIVERSITY

Modern Systems are Distributed

- Big Compute/Data
- Fault-tolerant
- Scalable
- Decentralized
- Flexible

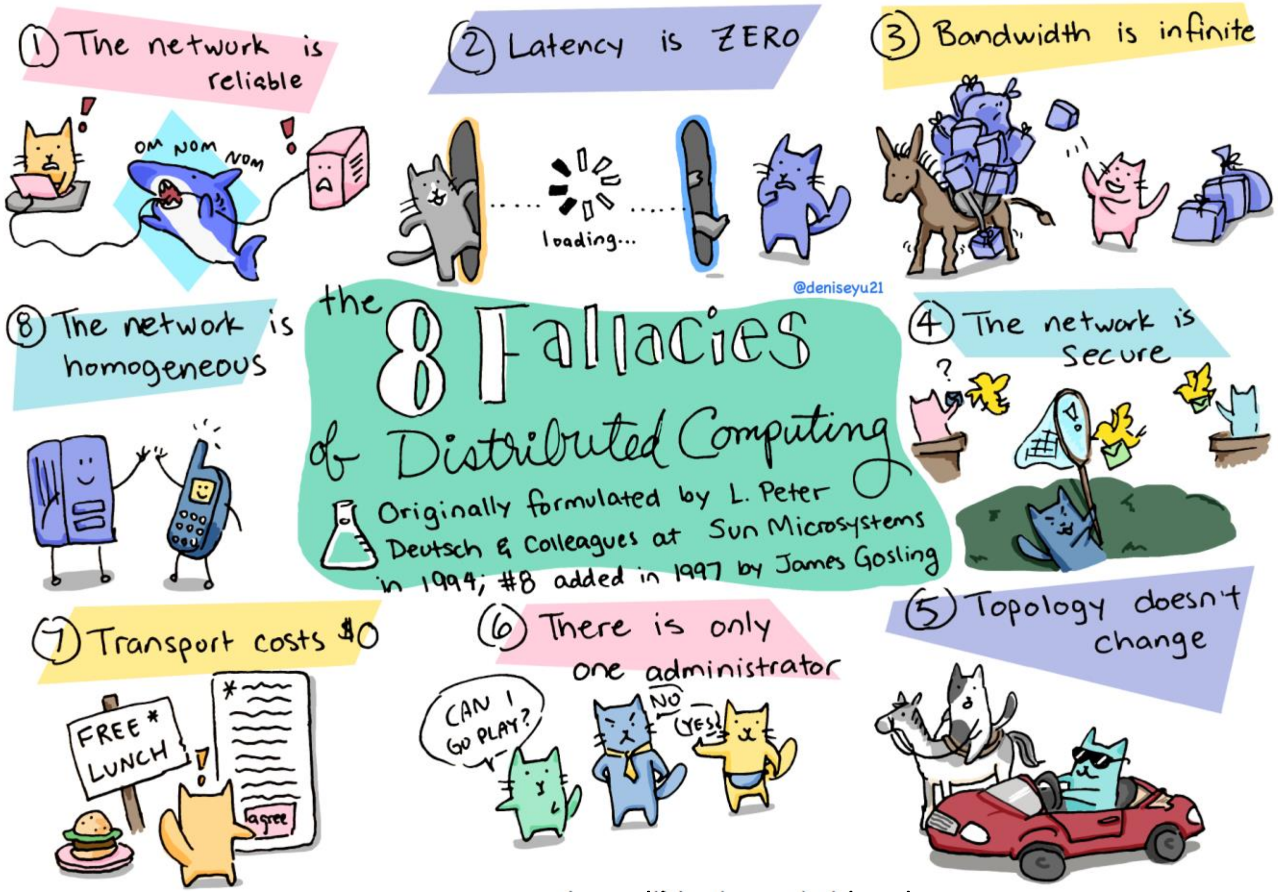
= Distributed

- set of independent machines connected by a network
- set of processes running on machines
- aggregates resources of individual machines
- appears to clients as a single computing system



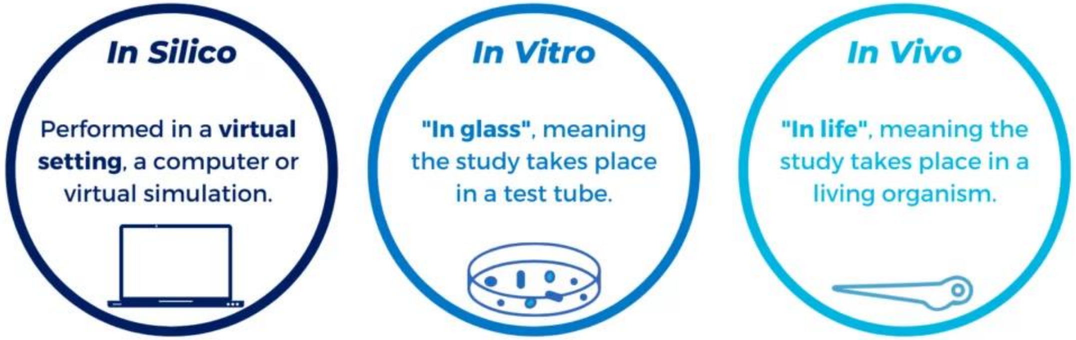
Distributed Systems are Hard

- Asynchrony
- Concurrency
- Absence of a global clock
- Partial failures
- Heterogeneity
- Dynamicity
- Large scale



Source: https://deniseyu.io/art/

Typical Challenges

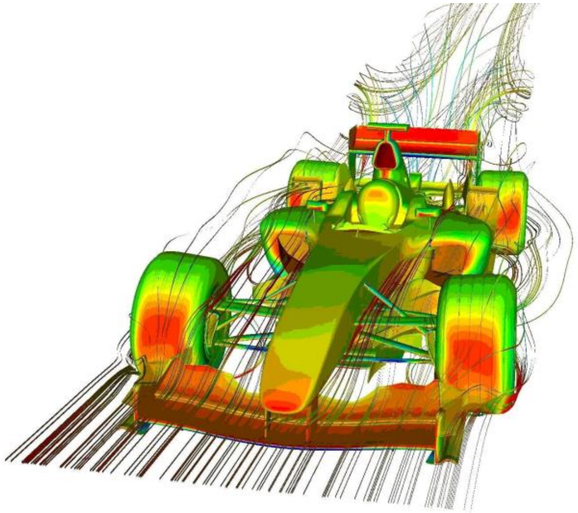


- Researchers
  - How to evaluate proposed approach?
  - Analytical models are not enough
- Practitioners
  - How to evaluate design of new system?
  - How to improve operation of existing system?
  - How to answer “what if” questions?
  - Experiments on real-scale systems are expensive, long, risky and unreproducible
- Educators
  - How to expose students to problems that occur in modern systems?
  - Small lab environments are not enough

Simulation Modeling to the Rescue

The studied system is replaced by a computer model that imitates the real system (components, processes) with sufficient accuracy.

- Real system is not needed
- Inexpensive, moderate resource requirements
- Faster experiments, no real-time
- Full control over environment
- Reproducibility
- Any system configuration or scenario



Tools for Simulation of Distributed Systems

	SimGrid	CloudSim (Plus)	OpenDC
Domain-specific	No (HPC roots)	Yes (cloud)	Yes (datacenter)
General-purpose simulation API	No (actors model)	Only events, no control	No
Asynchronous programming	Yes	No	Yes
Network modeling	Advanced	Basic	Basic
Validation	High	Medium	Medium
Extensibility	Limited	Limited (Low)	No
Performance/Scalability	Medium	Low	Low
Documentation	Great	Poor (Good)	Poor
Language	C++	Java	Kotlin
Other languages support	Yes (Python)	No	No
License	LGPL-2.1	GPL-3.0	MIT
Last stable release	Jun 2023	Jun 2019 (May 2023)	May 2021

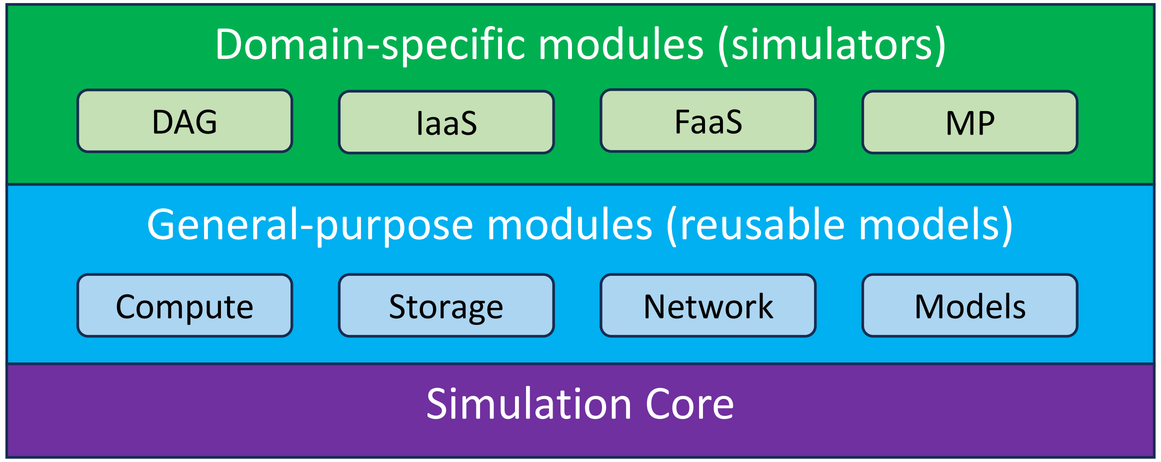
Why Another Simulation Framework?

- Versatility and extensibility
  - Not tailored to a specific domain or a use case
  - Ability to extend/adapt to any domains and use cases
- Convenient and flexible programming model
  - Simple event-based model with precise simulation control
  - Support for asynchronous programming
- High performance and ability to simulate large-scale systems
  - Support systems with millions of entities and millions of events per second
  - Support parallel execution of multiple simulations for large experiments

DSLab

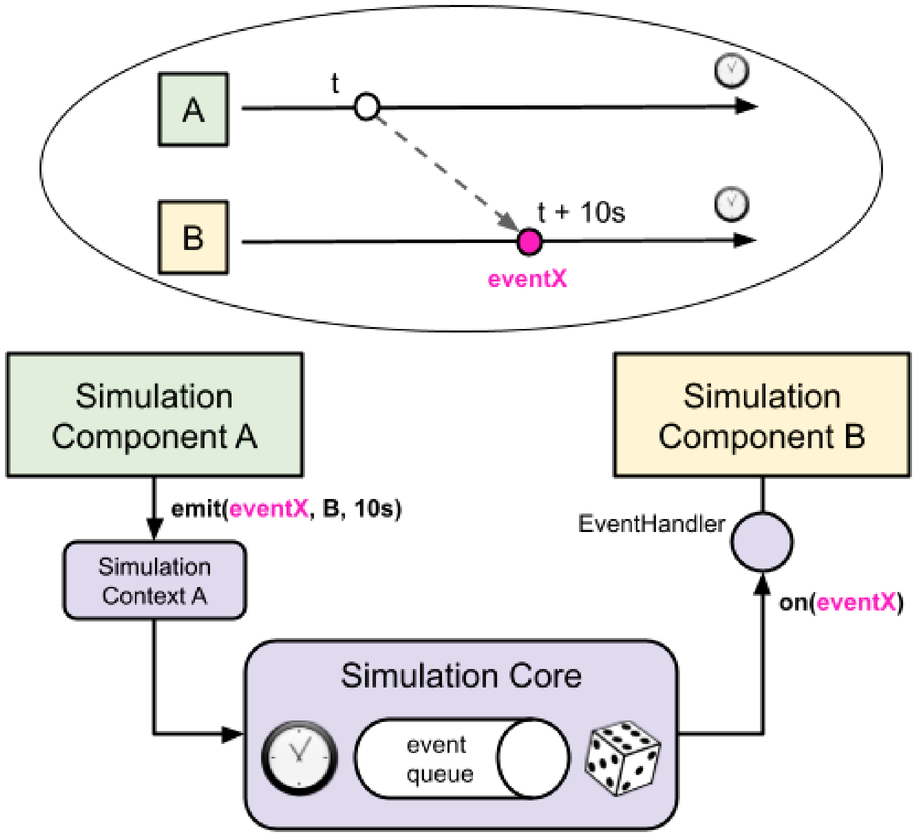
Modular framework for simulation of distributed systems

- Developed since 2021, used in different research domains and educational activities
- Open source, implemented in Rust programming language (performance, safety, tooling)
- Loosely-coupled modules built around the generic simulation core



Simulation Core

- Generic **discrete-event simulation** engine
- Simulation consists of user-defined **components** producing and consuming user-defined **events**
- Manages **simulation state** (clock, event queue, RNG) and provides access to it for components
- Processes events in their timestamp order by invoking their destination components
- Provides APIs for producing and consuming events
- Events can be arbitrary delayed and cancelled
- Single threaded, multiple threads can be used to run different simulations in parallel



Asynchronous Programming Support

- Default callback-based model is not convenient for describing multi-step activities
- Experimental support for asynchronous programming is added to DSLab core
- Both approaches can be used together to combine their advantages

```
pub struct Worker {
 tasks: HashMap<u64, TaskInfo>,
 computations: HashMap<u64, u64>,
 reads: HashMap<u64, u64>,
 writes: HashMap<u64, u64>,
 downloads: HashMap<usize, u64>,
 uploads: HashMap<usize, u64>,
}

fn on_task_request(request);
fn on_data_read_completed(r_id);
fn on_comp_started(comp_id);
fn on_comp_finished(comp_id);
fn on_data_write_completed(r_id);

fn on_data_transfer_completed(*);
```

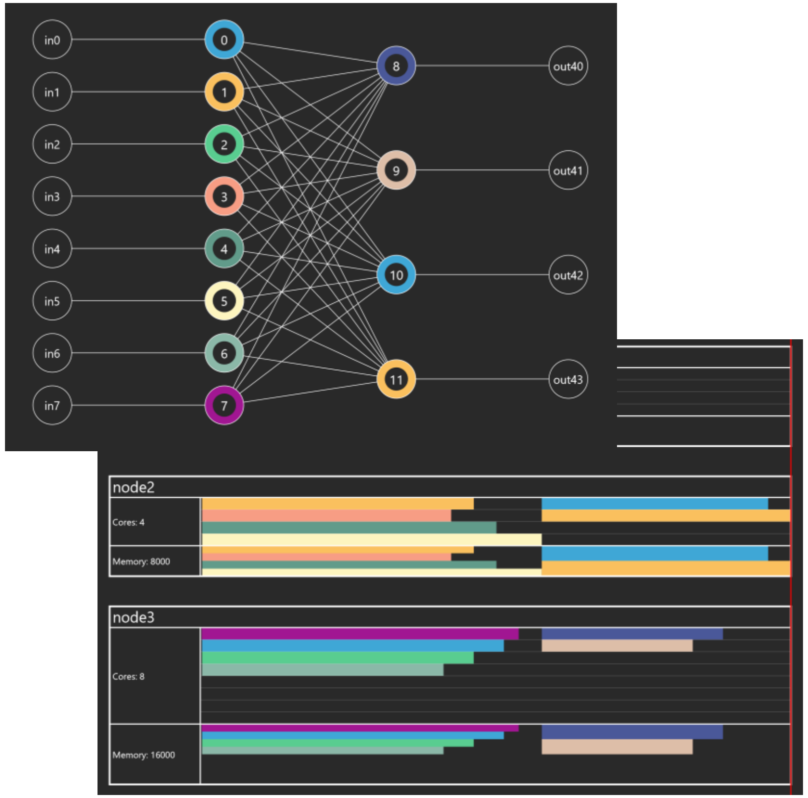
```
fn on_task_request(req) {
  self.ctx.spawn(
    self.process_task(req));
}

async fn process_task(&self, req:
    TaskRequest) {
  let mut task = TaskInfo {req};

  self.download_data(&task).await;
  self.read_data(&task).await;
  self.run_task(&task).await;
  self.write_data(&task).await;
  self.upload_result(&task).await;
}
```

Models

- Compute
  - Computing resource (flop/s) which performs compute tasks (flops)
  - Multiple cores, memory, speedup function
- Storage
  - Raw data storage which performs data read and write operations
  - Disk model with configurable bandwidth model (sharing, degradation…)
- Network
  - Performs communication operations (message passing, data transfer)
  - Basic single-link models, advanced topology model, bandwidth sharing
- Throughput
  - Generic model of sharing resource with limited throughput (used in above models)
  - Supports modeling of degradation, variability and dependence on operation
- Power
  - Power consumption models for CPU, disk and memory

DSLab DAG

- Library for studying the scheduling of computations represented as directed acyclic graphs (DAG)
- Components:
  - distributed computing system with multiple resources
  - DAG job consisting of multiple tasks
  - job runner using the specified scheduling algorithm
  - interface for algorithm implementation
- User-defined: DAG scheduling algorithms

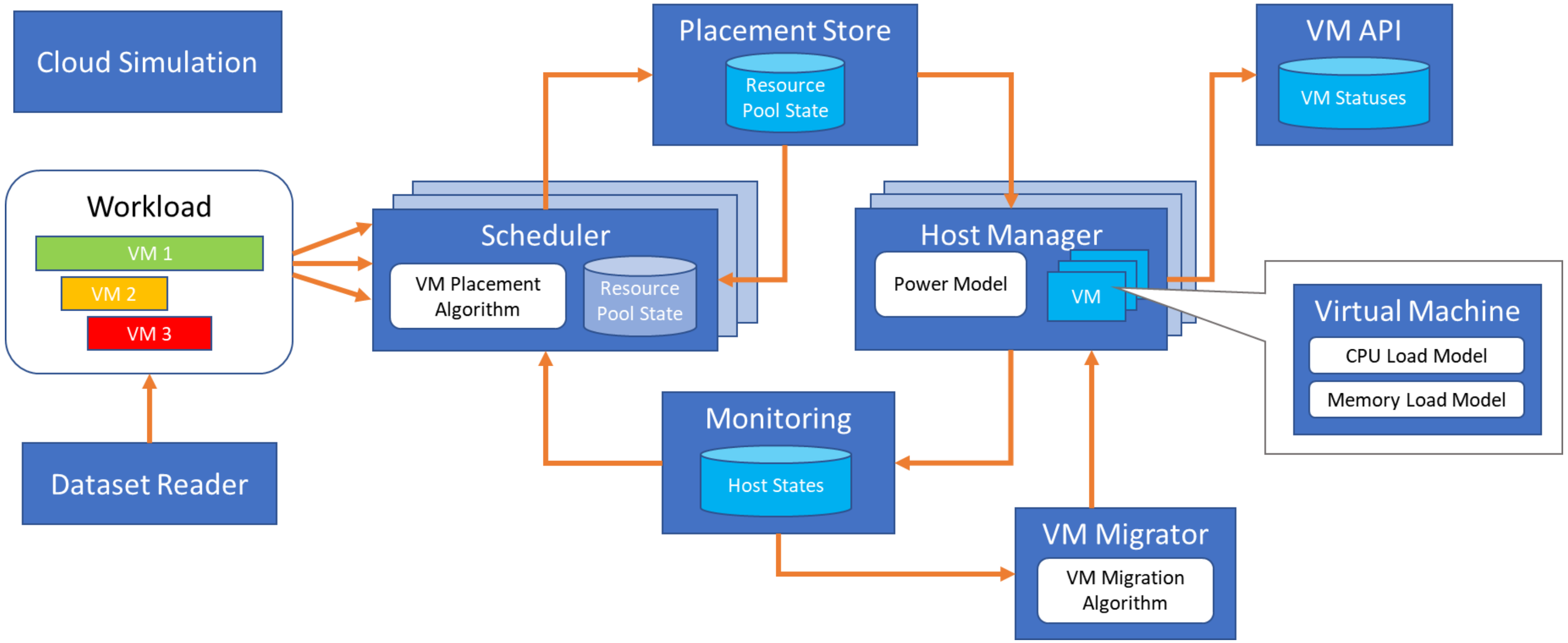


Sukhoroslov O. Scheduling of Workflows with Task Resource Requirements in Cluster Environments (PaCT 2023)

Sukhoroslov O., Gorokhovskii M. Benchmarking DAG Scheduling Algorithms on Scientific Workflow Instances (Russian Supercomputing Days 2023)

DSLab IaaS

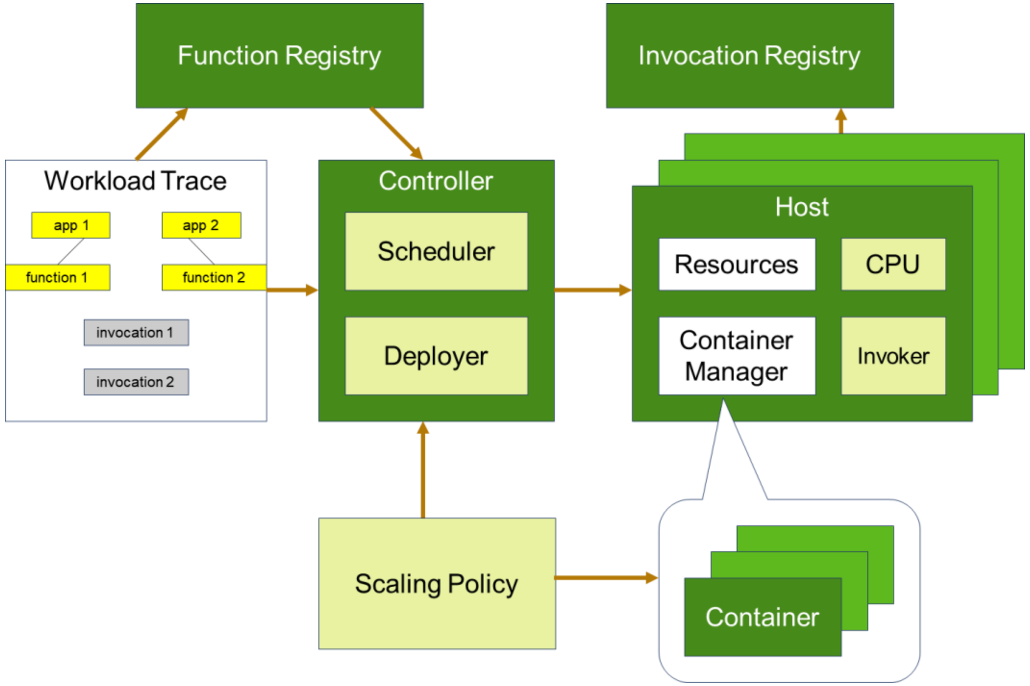
Library for studying resource management in Infrastructure as a Service (IaaS) clouds.



Sukhoroslov O., Vetrov A. Towards Fast and Flexible Simulation of Cloud Resource Management (MoNeTec 2022)

DSLab FaaS

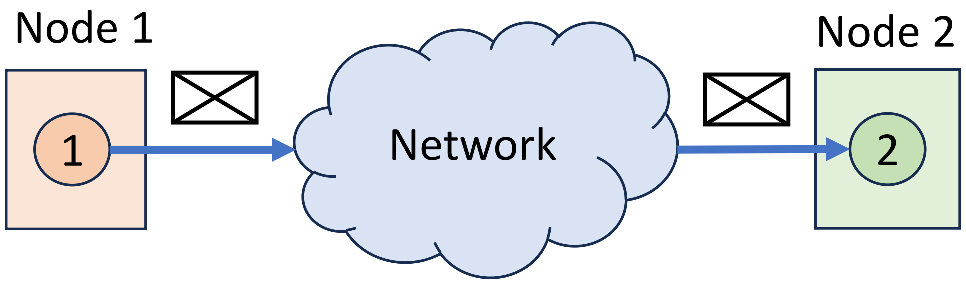
Library for studying resource management in Function-as-a-Service (IaaS) clouds.



Semenov Y. Sukhoroslov O. DSLab FaaS: Fast and Accurate Simulation of FaaS Clouds (GRID 2023)

DSLab MP

- Implementation of message passing model: processes running on nodes communicate with each other by sending messages via network
- Network can drop, duplicate, corrupt, delay and reorder messages
- Deterministic execution and debug output
- Process logic can be implemented in Python
- Used in homework assignments for Distributed Systems course in HSE
- Hand-crafted and randomized tests
- Experimental support for model checking




```
--- [EXACTLY ONCE ORDERED] DROPPED ---

   0.000     sender <<< local      MESSAGE {"text": "distributed"}
   0.000     sender --> receiver   MESSAGE {"text": "distributed"}
   0.000     sender <<< local      MESSAGE {"text": "systems"}
   0.000     sender --> receiver   MESSAGE {"text": "systems"}
   1.000   receiver <-- sender     MESSAGE {"text": "distributed"}
   1.000   receiver >>> local      MESSAGE {"text": "distributed"}
   1.000   receiver <-- sender     MESSAGE {"text": "systems"}
   1.000   receiver >>> local      MESSAGE {"text": "systems"}
   1.000     sender <<< local      MESSAGE {"text": "need"}
   1.000     sender --> receiver   MESSAGE {"text": "need"}
   2.000   receiver <-- sender     MESSAGE {"text": "need"}
   2.000   receiver >>> local      MESSAGE {"text": "need"}
   2.000     sender <<< local      MESSAGE {"text": "some"}
   2.000     sender --> receiver   MESSAGE {"text": "some"}
     !!!     sender --x receiver   MESSAGE {"text": "some"} <-- message dropped
   2.000     sender <<< local      MESSAGE {"text": "guarantees"}
   2.000     sender --> receiver   MESSAGE {"text": "guarantees"}
   3.000   receiver <-- sender     MESSAGE {"text": "guarantees"}
   3.000   receiver >>> local      MESSAGE {"text": "guarantees"}

FAILED: Message {"text": "some"} is not delivered (observed count 0 < expected count 1)
```

Performance Benchmarks

Ping-Pong: *N* processes communicating with *P* peers by exchanging Ping and Pong messages

				Simulation Time (s)	
Processes	Peers	Iterations	Network Model	DSLab	SimGrid
2	1	1000000	N	0.17	5.27
2	1	10000000	N	1.61	52.29
1000	10	1000	N	0.35	5.93
10000	100	1000	N	4.27	159.81
100000	100	1000	N	82.88	-
1000000	100	1000	N	946.20	-
2	1	1000000	Y	0.26	7.19
2	1	10000000	Y	2.52	72.48s
1000	10	1000	Y	0.53	71.20
10000	100	1000	Y	8.49	7739.58
100000	100	1000	Y	253.94	-
1000000	100	1000	Y	4202.09	-

Master-Workers: Distributed computing system with master node scheduling tasks to worker nodes

			Simulation Time (s)	
Workers	Tasks	Bandwidth Sharing	DSLab	SimGrid
100	10000	N	0.08	0.63
1000	10000	N	0.08	7.47
1000	100000	N	0.97	41.05
10000	1000000	N	16.31	14015.21
100000	1000000	N	21.53	-
1000000	1000000	N	26.11	-
1000000	10000000	N	320.32	-
100	10000	Y	0.13	0.60
1000	10000	Y	0.17	20.60
1000	100000	Y	5.97	335.53
10000	1000000	Y	977.40	-

Other Performance Comparisons

IaaS: Simulation of cloud resource pool processing VM allocation requests from the public cloud traces.

Huawei Cloud trace:

Length	Hosts	VMs	Simulation Time (s) DSLab IaaS	CloudSim Plus
10 hours	40	2502	0.8	18
1 day	50	4483	2.4	77
7 days	150	26953	44	2136
30 days	600	120003	595	34112

Microsoft Azure trace:

Length	Hosts	VMs	Simulation Time (s) DSLab IaaS	CloudSim Plus
1 hour	900	15130	1.4	503
1 day	15000	436371	1430	-
7 days	50000	2455264	25004	-

DAG: Execution of scientific workflows in the system consisting of 10 resources.

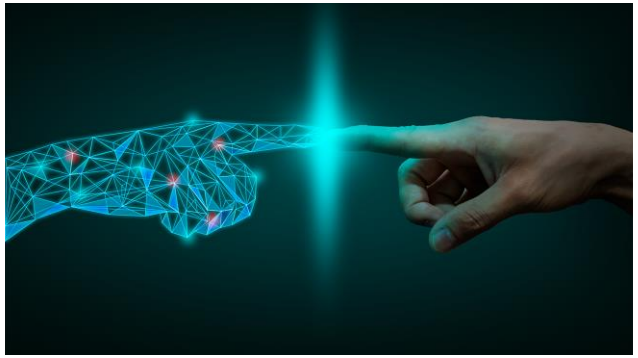
		Simulation Time (s)		Memory (MB)	
Workflow	Tasks	DSLab	WRENCH	DSLab	WRENCH
Montage	4991	0.20	9.08	16	459
Epigenomics	9995	0.44	12.46	27	649
Srasearch	9997	0.66	38.88	42	2444
Genome	9998	0.64	48.40	34	790

FaaS: Simulation of FaaS platform processing cloud function invocations from Microsoft Azure 2019 trace.

RPS	DSLab time (s)	OpenDC time (s)	FaaS-Sim time(s)
10	0.51	2.298	83.9
20	1.07	4.306	206.718
30	1.68	70.211	795.168
40	2.25	70.732	-
50	2.92	8.885	-
60	3.66	9.892	-

Verification and Validation

- Most of the framework modules are covered by tests
- Reproduced experiments from research papers
- Comparison of results with other simulators
- Next: detailed validation studies



Conclusion

- Simulation modeling is important for research, development and studying of distributed systems
- DSLab framework strives to improve on state-of-the-art with convenient and versatile programming model, extensible modular design, high performance and ability to simulate large-scale systems
- Current work demonstrates the viability of the proposed approach

Acknowledgements

DSLab contributors: Maksim Gorokhovskii, Andrey Vetrov, Yury Semenov, Tagir Kuskarov, Roman Ilgovskiy, Anthony Udovichenko, Konstantin Amelichev, Artem Makogon, Marina Gruzdeva, Sofia Semenova-Zvenigorodskaya, Ksenia Petrenko

DSLab development is supported by:

- HSE University (project “Development of tools for modeling and testing of distributed systems”)
- Russian Science Foundation (project “A study of resource management and scheduling methods for modern large-scale distributed systems”)

Thank you!

https://github.com/osukhoroslov/dslab

sukhoroslov@iitp.ru