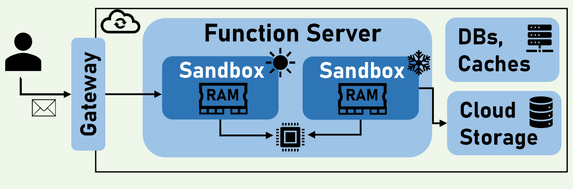
Accepted Paper | Thursday | Session-7.1

Larissa Schmid, Marcin Copik, Alexandru Calotoiu, Laurin Brandner, Anne Koziolek, Torsten Hoefler

SeBS-Flow: Benchmarking Serverless Cloud Function Workflows

Performance of Serverless Computing



In serverless, cloud platforms take full control over scheduling and resource allocation.
Lower costs and easier deployments come with a trade-offf: performance analysis becomes difficult.

- Black-box systems that are hard to analyze.
- Shared environments with performance variance.
- Lack of a standardized benchmarking suite.

SeBS: The Serverless Benchmark Suite

Functions

- Website and utility functions.
- Multimedia processing.
- Machine learning inference.
- Scientific applications.
- Microbenchmarks (I/O, latency).

Platforms

APACHE OpenWhisk, * K, * fission

Languages

python, node, C++

spcl/serverless-benchmarks

* Experimental beta support.

Next Generation of Serverless: Workflows

```
"init": {
  "Type": "Pass",
  "Result": "States.Array(0,
            1, 2, 3)",
  "ResultPath": "$.array",
  "Next": "map"
},
"map": {
  "Type": "Map",
  "ItemsPath": "$.array",
  "Parameters": {
    "payload.$":
      "$$.Map.Item.Value"
  },
  "Iterator": {
    "StartAt": "process",
    "States": {
      "process": {
        "Type": "Task",
        "Resource": "arn:proc",
        "Parameters": {
          "payload.$":
            "$.payload"
        },
        "End": true
      }
    }
  },
  "ResultPath": "$.res",
  "End": true
}
```

Serverless workflows compose applications from functions connected with control-flow and data-flow policies. Cloud operator is still responsible for the full execution.

- Build large serverless applications!
- Workflow models across platforms are not compatible with each other.

```
tasks = []
for i in range(4):
  tasks.append(context.call_activity("process", i)
res = yield context.task_all(parallel_tasks)
```

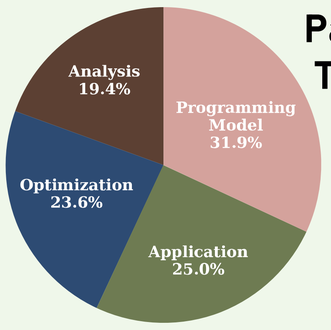
Azure Durable Functions
Dynamic execution with orchestrator.

AWS Step Functions
State machine in JSON

```
"assign_array": {
    "assign": [
      {"array": [0, 1, 2, 3]}]
},
"process": {
    "call":"exp.exec.map",
    "args":{
      "workflow_id":"map",
      "arguments":"${array}"
    },
    "result":"res"
}
"separate map-workflow:"
"main": {
"params":[ "elem" ],
"steps":[
  {
  "map":{
    "call":"http.post",
    "args":{
      "url":"google.process",
      "body":{
        "payload":"${elem}"
      }
    },
    "result":"elem" }
  },
  { "ret": {
      "return":
      "${elem.body}" } }
] }
```

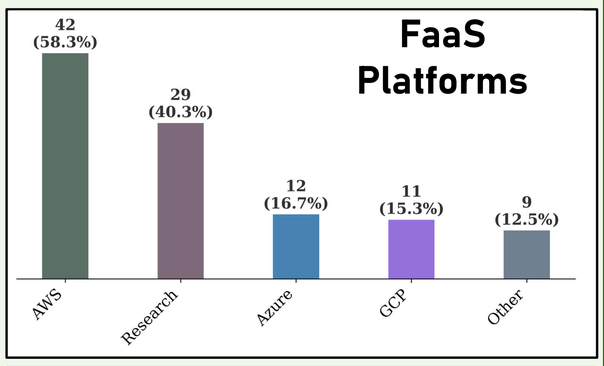
Google Cloud Workflows
State machine in JSON/YAML

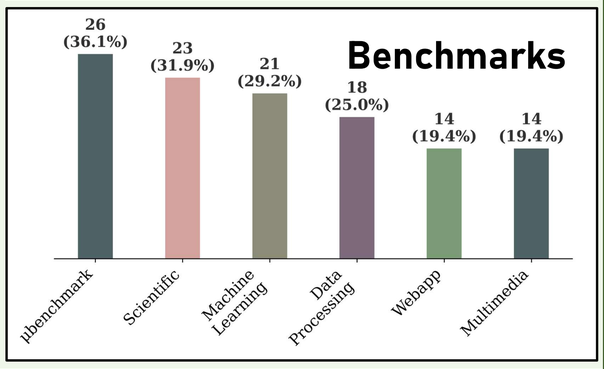
Workflows in Practice



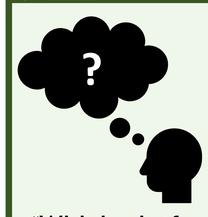
How researchers evaluate serverless workflows? Analysis of evaluation in 72 papers published between 2017 and 2024.

Benchmarking methodology is inconsistent and fragmentary, blocking effective meta-analysis.





Next Generation of Serverless Benchmarking: SeBS-Flow



"Which platform is best for my workflow? Can my application scale? What are the performance limits of each platform?"

Functions + Data + Workflow Model

Deploy with SeBS

- Allocate Cloud Resources
- Build & Deploy Functions
- Upload Data
- Create Workflow

Experiment with SeBS

- Run Experiment
- Download Measurements

"The end-to-end latency is not what I expected because of lower parallelism and variable I/O performance."

- Fully automatic pipeline
- Cloud-agnostic representation
- Representative benchmarks
- Detailed performance analysis

Workflow Benchmark Selection

Trip Booking
Web application with SAGA pattern and NoSQL database.

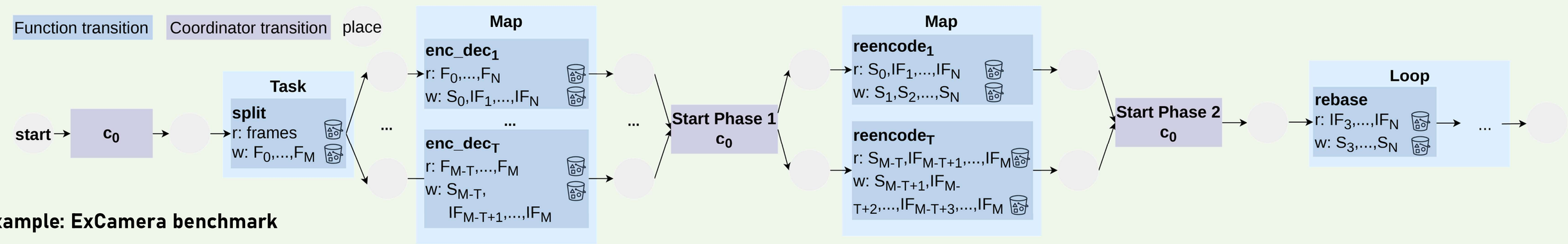
Video Analysis
Object detection in video, parallelized version of benchmark from vSwarm.

MapReduce
Word counting with sequential shuffling and parallel reduction.

ExCamera
Parallel video encoding from a popular serverless application (Fouladi et. al, NSDI'17)

Machine Learning
Concurrent training of two classifiers: Support Vector Machines and Random Forests

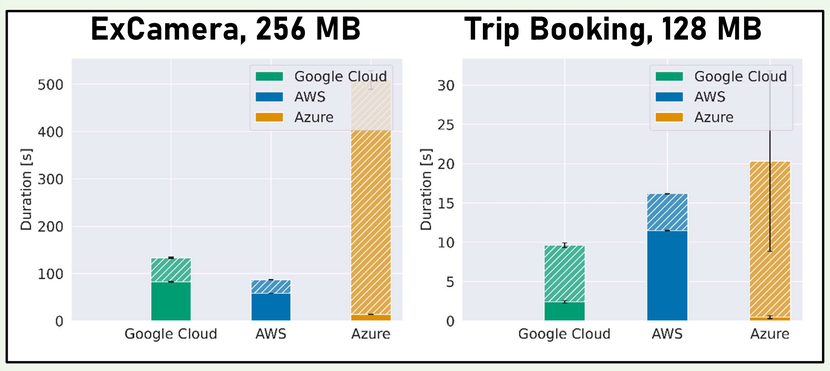
1000Genomes
Scientific workflow that identifies mutation overlaps.



Example: ExCamera benchmark

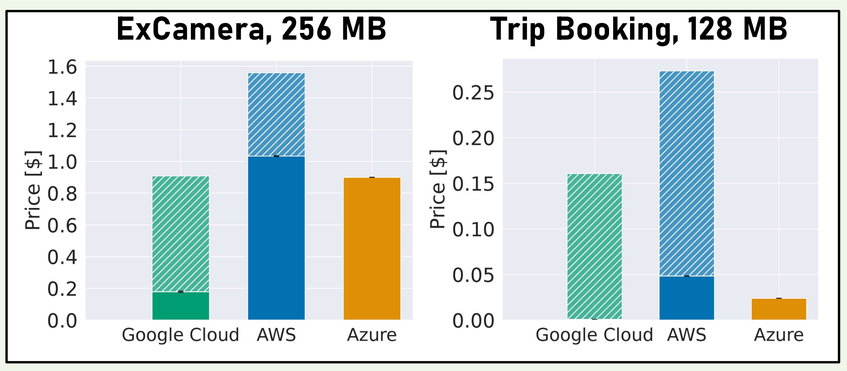
Evaluation: Performance, Cost & Scalability

Latency



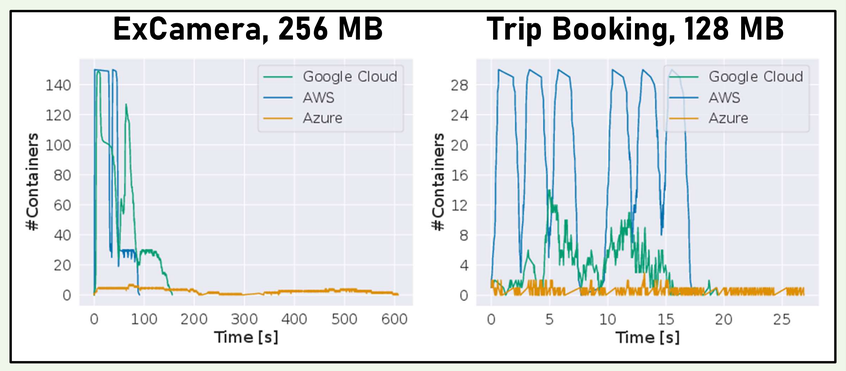
Critical path (opaque) and overhead (hatched).

Execution Cost



Function costs (opaque) and state transition costs (hatched).

Scalability



Number of distinct function containers for 30 invocations.

We use the lowest common memory configuration that successfully executes the workflow on AWS and Google Cloud, at least 256 MB for compute-intensive and 128 MB for web applications. We invoke the application benchmarks in burst mode, triggering 30 executions at once, and we repeat the measurements until we obtain tight confidence intervals (within 5% of median).

< EUROSYS'25 > | SeBS Repository | SeBS-Flow Paper | High-Performance Serverless Projects

ETH zürich | KIT Karlsruher Institut für Technologie