Challenges for Scheduling Scientific Workflows on Cloud Functions

Joanna Kijak, Piotr Martyna, Maciej Pawlik, Bartosz Balis and **Maciej Malawski**

Department of Computer Science,
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Kraków, Poland

DISTRIBUTED COMPUTING ENVIRONMENTS TEAM

http://dice.cyfronet.pl

Outline

- Motivation: scientific workflows in clouds
- Experiments with HyperFlow
- Scheduling challenges
- Experiments with SDBWS algorithm
- Results on AWS Lambda
- Conclusions

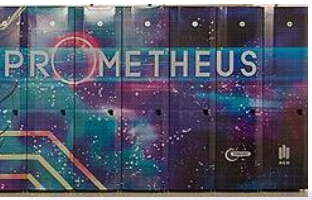
DICE Team

- Investigation of methods for building complex scientific collaborative applications
- Elaboration of environments and tools for e-Science
- Integration of large-scale distributed computing infrastructures
- Knowledge-based approach to services, components, and their semantic composition

AGH University of Science and Technology (1919)

16 faculties, 36000 students; 4000 employees

http://www.agh.edu.pl/en



Academic Computer Centre CYFRONET AGH (1973)

120 employees

http://www.cyfronet.pl/en/

Faculty of Computer Science, Electronics and Telecommunications (2012)

2000 students, 200 employees

http://www.iet.agh.edu.pl/

Other 15 faculties

Department of Computer Science AGH (1980)

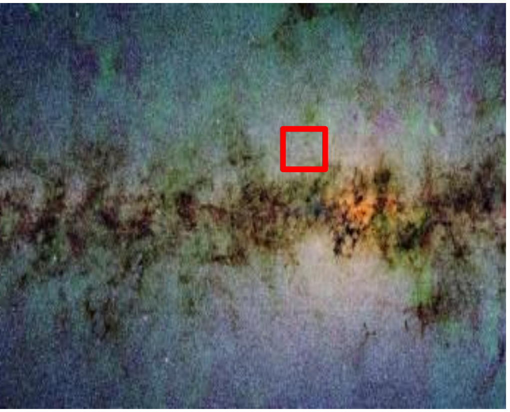
800 students, 70 employees

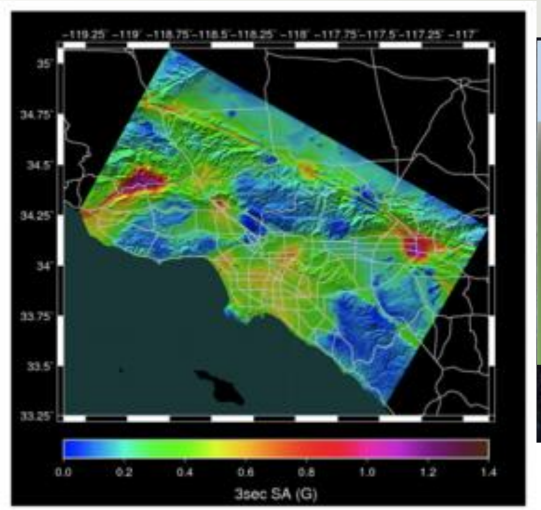
http://www.ki.agh.edu.pl/uk/index.htm

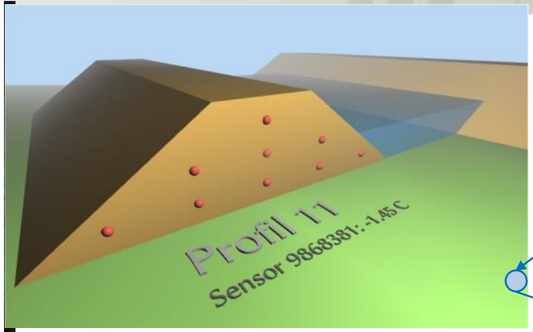
DISTRIBUTED COMPUTING ENVIRONMENTS TEAM

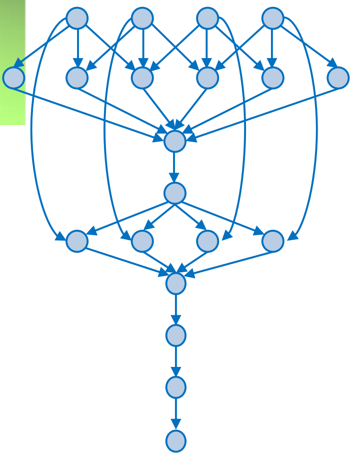
http://dice.cyfronet.pl

Motivation: Scientific Workflows

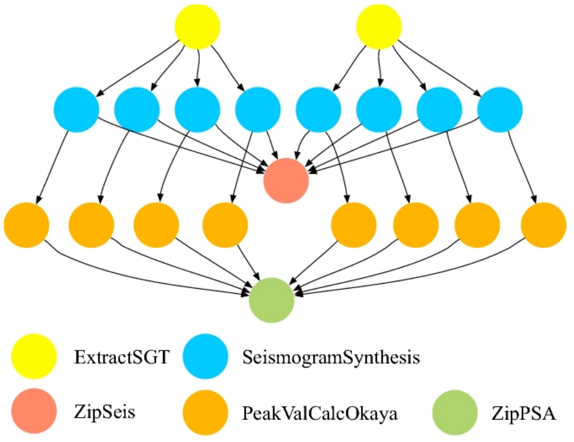








- Astronomy, Geophysics, Genomics, Early Warning Systems …
- Workflow = graph of tasks and dependencies, usually directed acyclic graph (DAG)
- Granularity of tasks
  - Large tasks (hours, days)
  - Small tasks (seconds, minutes)



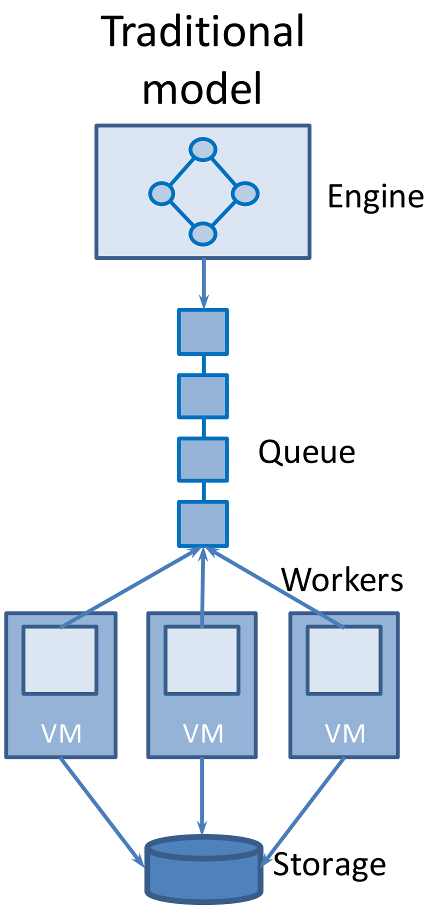
Infrastructure – from clusters to clouds

- Traditional HPC clusters in computing centers
  - Job scheduling systems
  - Local storage

- Grids to Clouds
  - Infrastructure as a service
  - Globally distributed
  - Virtual machines (VMs)
  - On-demand
  - Cost in $$ per time unit

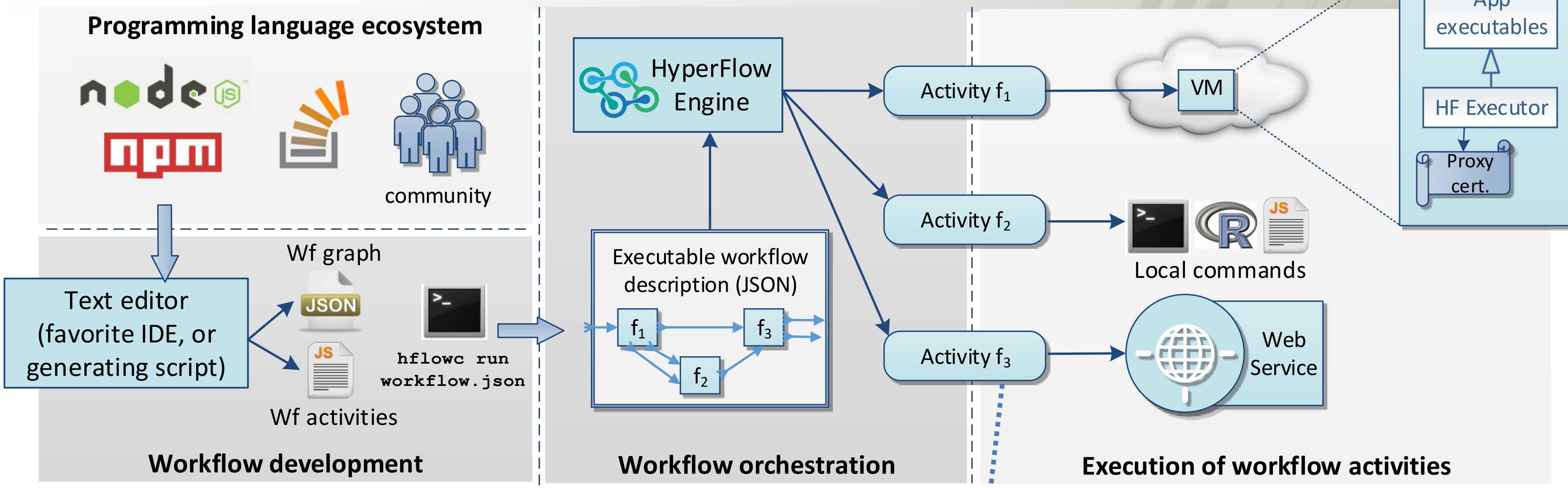
Workflow execution model in (traditional) clouds

- Workflow engine manages the tasks and dependencies
- Queue is used to dispatch ready tasks to the workers
- Worker nodes are deployed in Virtual Machines in the cloud
- Cloud storage such as Amazon S3 is used for data exchange
- Examples
  - **Pegasus**, Kepler, Triana, Pgrade, Askalon, …
  - **HyperFlow** (AGH Krakow)



HyperFlow

Lightweight workflow programming and execution environment developed at AGH



Simple wf description (JSON)

```
{
 "name": "PlotDataStatistics",
 "processes": [ {
   "name":    "ComputeStats",
   "ins":     [ "data.csv" ],
   "outs":    [ "stats.txt" ]
   "config":  {
     "executor": {
       "executable": "cstats.sh",
       "args": "data.csv -o stats.txt"
      } },
  }, {
   "name":     "PlotChart",
   "ins":      [ "stats.txt" ],
   "outs":     [ "stats.png" ],
   "config":   {
      "executor": {
         "executable": "plot.sh",
         "args": "stats.txt"
      } },
   } ]
```

Advanced programming of wf activities (JavaScript)

```
function getPathWayByGene(ins, outs, config, cb) {
  var geneId = ins.geneId.data[0],
      url = ...

  http({"timeout": 10000, "url": url },
  function(error, response, body) {
    ...
    cb(null, outs);
  });
}
```

Running a workflow – simple command line client

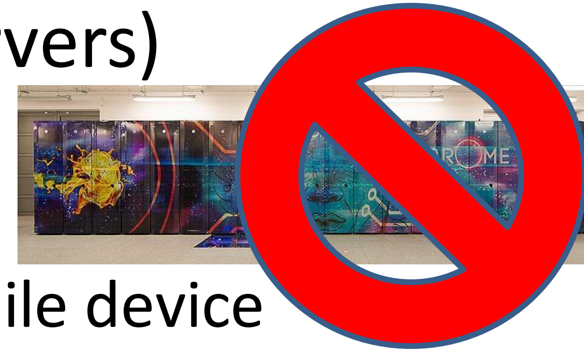
hflowc run <workflow_dir>

<workflow_dir> contains:

- File **workflow.json** (wf graph)
- File **workflow.cfg** (wf config)
- Optionally: file **functions.js** (advanced workflow activities)
- Input files

New challenges – serverless architectures

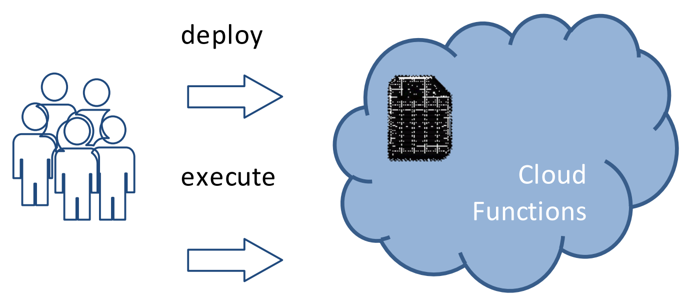
- Serverless – no traditional VMs (servers)
- Composing of applications from existing cloud services
  - Typical example: web browser or mobile device interacting directly with the cloud



- Examples of services:
  - Databases: Firebase, DynamoDB
  - Messaging: Google Pub/Sub
  - Notification: Amazon SNS
- **Cloud Functions**:
  - Run a custom code on the cloud infrastructure

Cloud Functions – good old RPC?

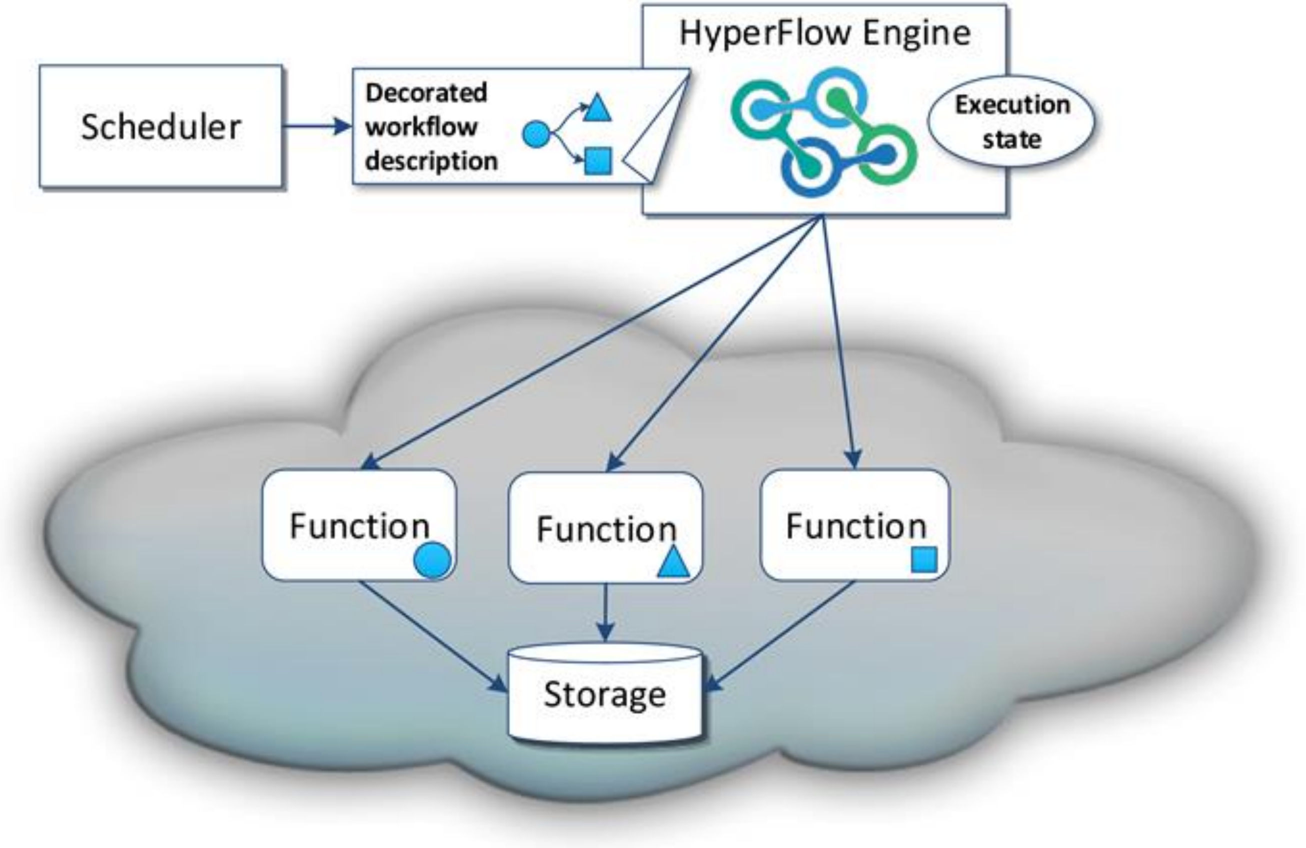
- Examples:
  - AWS Lambda
  - Google Cloud Functions (beta)
  - Azure Functions
  - IBM Bluemix OpenWhisk
- Functional programming approach:
  - Single function (operation)
  - **Not** a long-running service or process
  - Transient, stateless
- Infrastructure (execution environment) responsible for:
  - Startup
  - Parallel execution
  - Load balancing
  - Autoscaling
- Triggered by
  - Direct HTTP request
  - Change in cloud database
  - File upload
  - New item in the queue
  - Scheduled at specific time
- Developed in specific framework
  - Node.js, Java, Python
  - Custom code, libraries and binaries can be uploaded
- **Fine-grained pricing**
  - **Per 100ms * GB (Lambda)**



Scheduling challenges

- Which size of cloud functions should be allocated to each task of a workflow?
- Which tasks should be executed on FaaS and which ones on IaaS?
- What is the performance variability of the cloud functions infrastructure and how to deal with it?
- What are the limits of concurrency that we can expect when running multiple tasks as cloud functions in parallel?
- How to address the problem of data transfer be-tween tasks?

Execution model



[9] B. Baliś. Hyperflow: A model of computation, programming approach and enactment engine for complex distributed workflows

Serverless Deadline-Budget Workflow Scheduling

- Adaptation of existing DBWS[2] heuristic for serverless model
- Finds mapping between tasks and resources (cloud functions) to meet the deadline constrait and tries to meet the budget constraint
- Assumes the knowledge of task runtime estimates on each resource type

Idea of algorithm

- For each workflow level compute the maximum execution time

$$Level^{j}_{execution} = \max_{l(t_i)==j} \{ET_{max}(t_i)\}$$

- We divide the deadline into sub-deadlines for each level:

$$Level^{j}_{DL} = Level^{j-1}_{DL} + D_{user} * \frac{Level^{j}_{execution}}{\sum_{1 \leq j' \leq l(t_{exit})} Level^{j'}_{execution}}$$

Resource selection

- Resource selection is based on the time and cost:

$$Time_Q(t_{cur}, r) = \frac{\xi * S_{DL}(t_{cur}) - FT(t_{cur}, r)}{FT_{max}(t_{cur}) - FT_{min}(t_{cur})}$$

$$Cost_Q(t_{cur}, r) = \frac{Cost_{max}(t_{cur}) - Cost(t_{cur}, r)}{Cost_{max}(t_{cur}) - Cost_{min}(t_{cur})} * \xi$$

$$\xi = \begin{cases} 1 & \text{if } FT(t_{cur}, r) < S_{DL}(t_{cur}) \\ 0 & \text{otherwise} \end{cases}$$

Resource selection (2)

- We select the resource which maximizes the quantity:

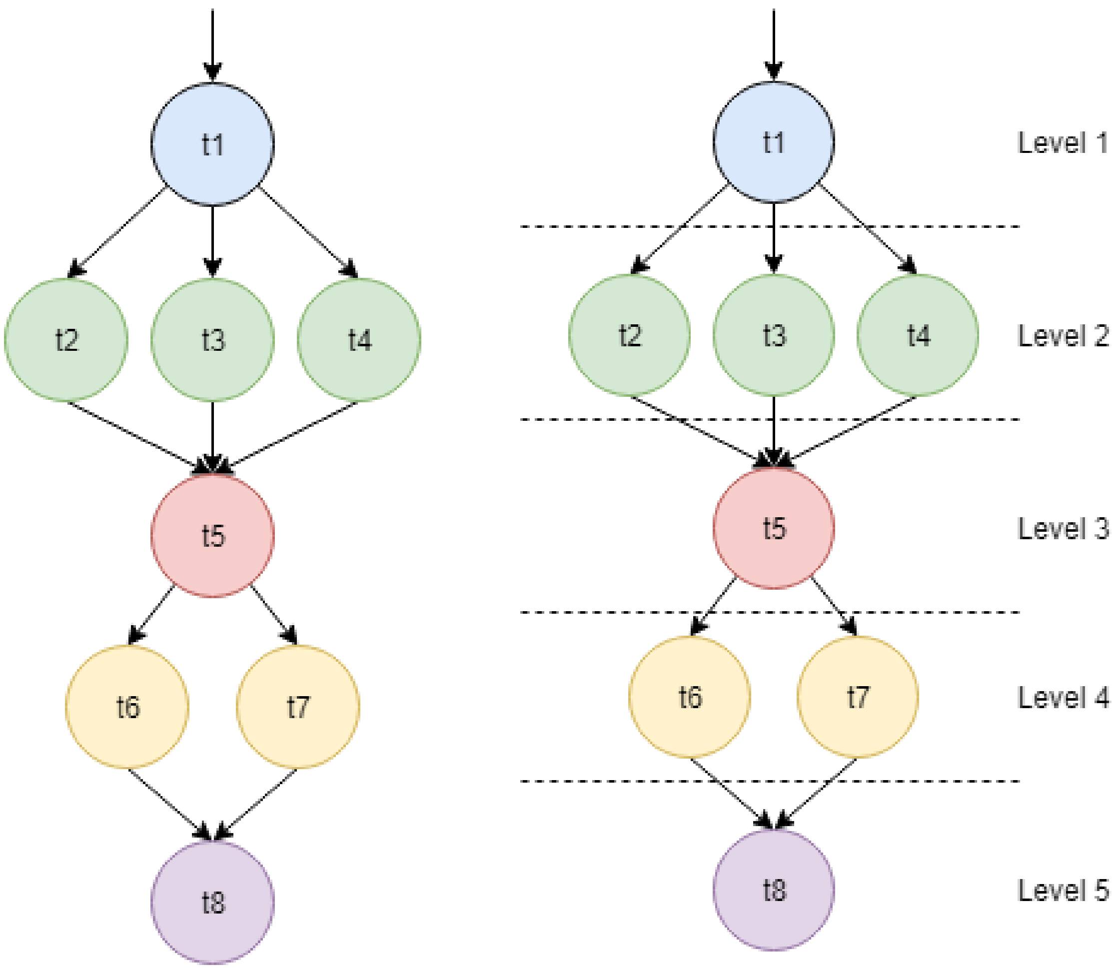
$$Q(t_{cur}, r) = Time_Q(t_{cur}, r) * (1 - C_F) + Cost_Q(t_{cur}, r) * C_F$$

- Where CF is a cost factor:

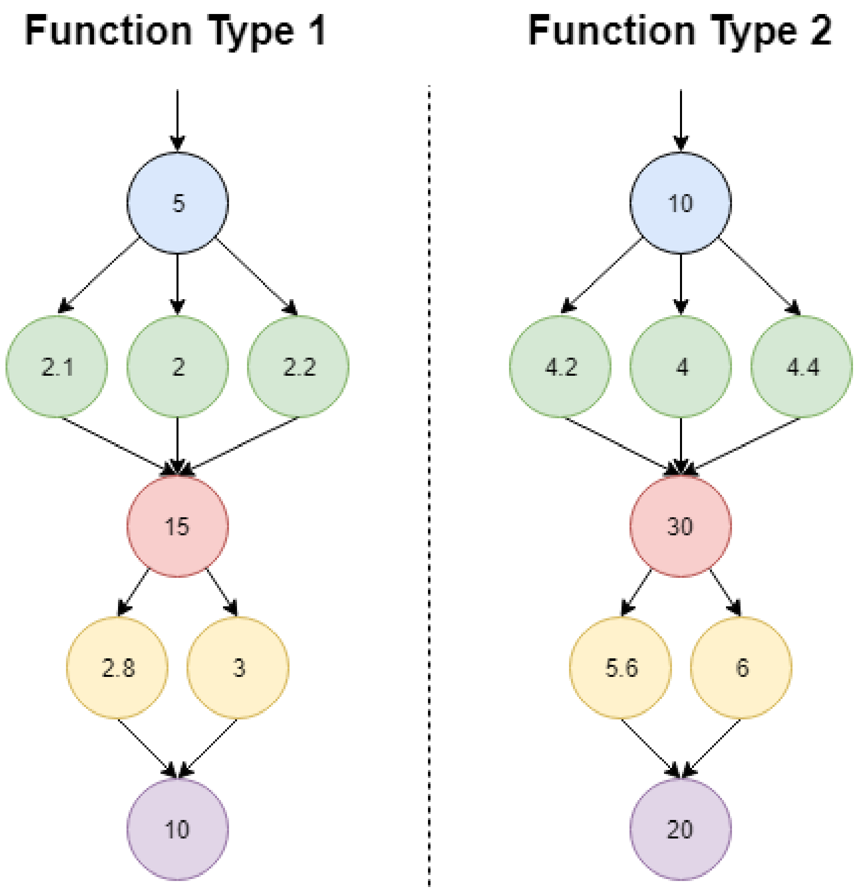
$$C_F = \frac{Cost_{low}(DAG)}{B_{user}}$$

- It represents user preferences:
  - Lower value means we prefer to pay more for faster execution
  - Higher value means we prefer cheaper and slower solutions

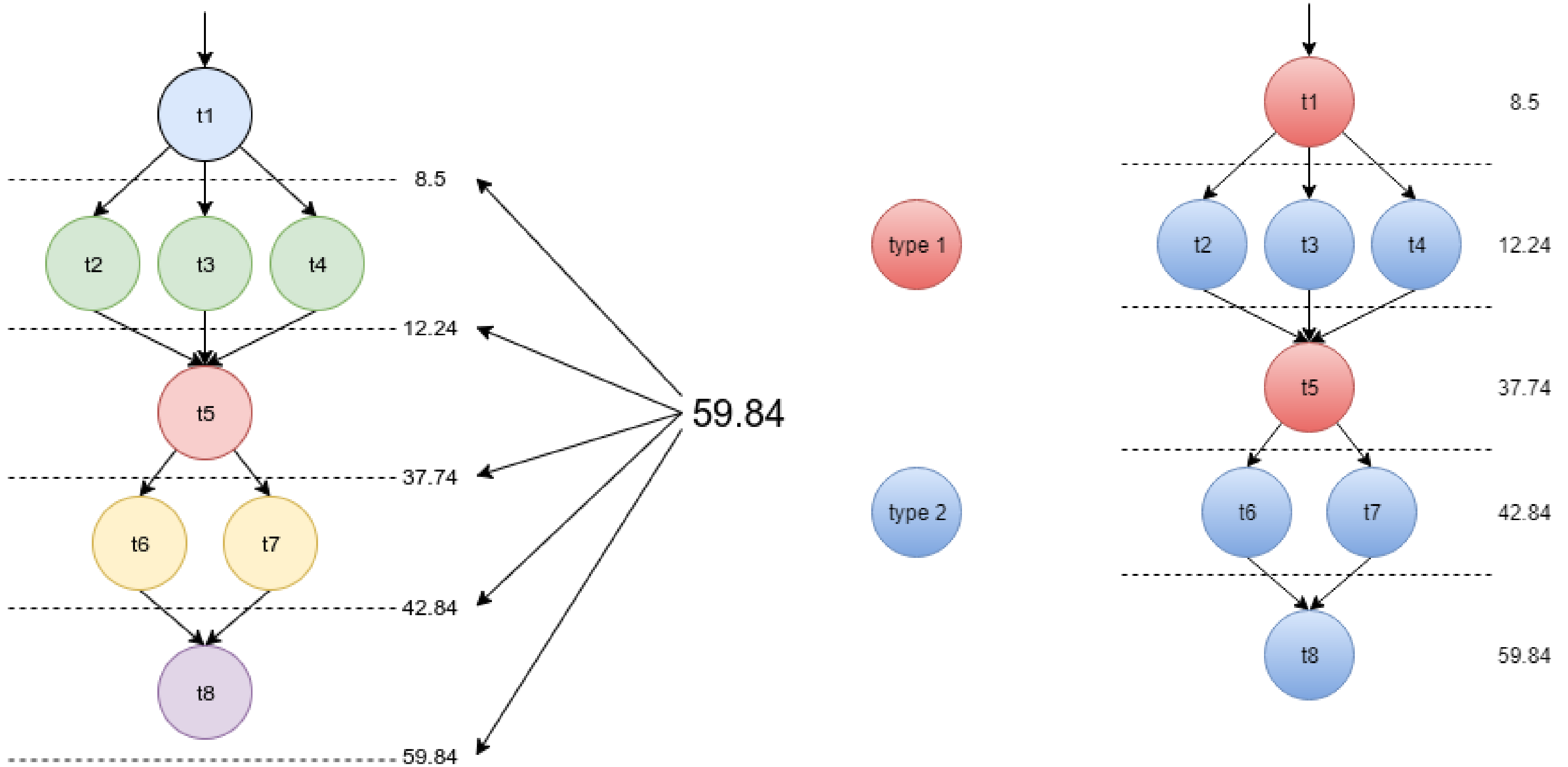
Example DAG


t1
t2
t3
t4
t5
t6
t7
t8
t1
Level 1
t2
t3
t4
Level 2
t5
Level 3
t6
t7
Level 4
t8
Level 5

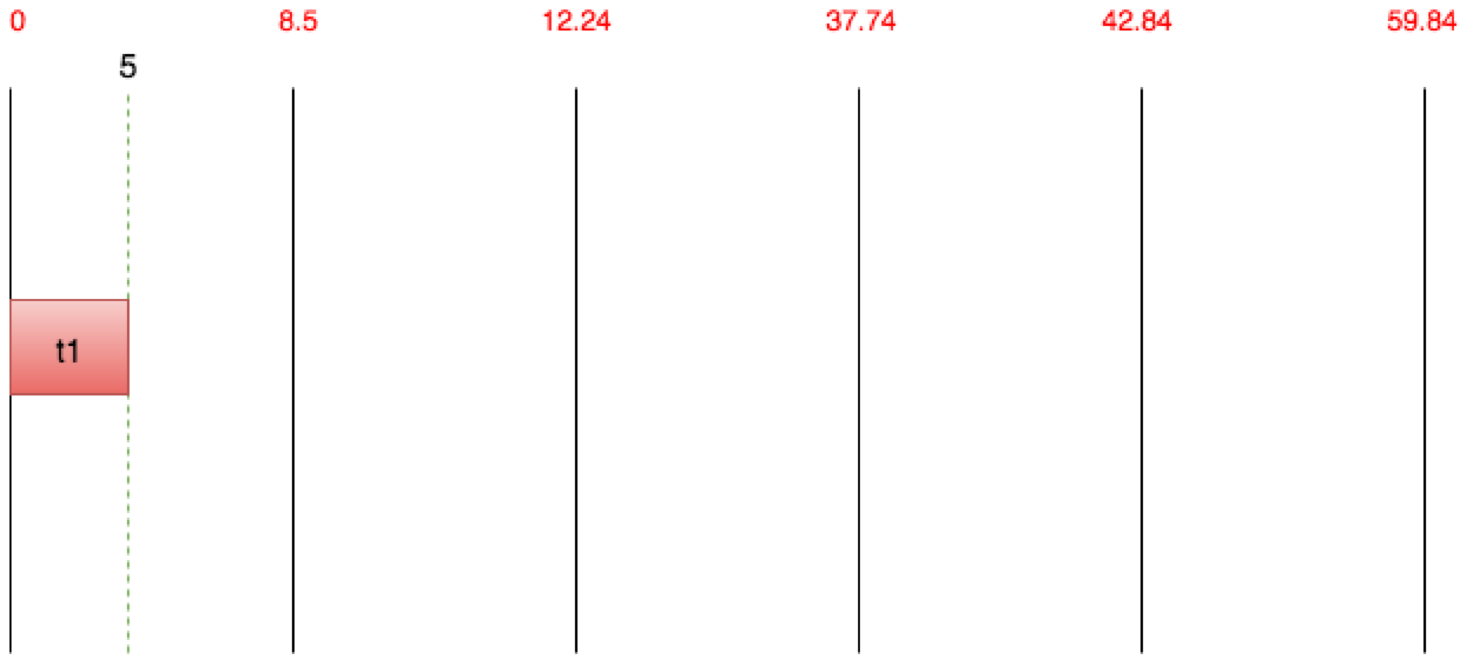
Task runtime estimates



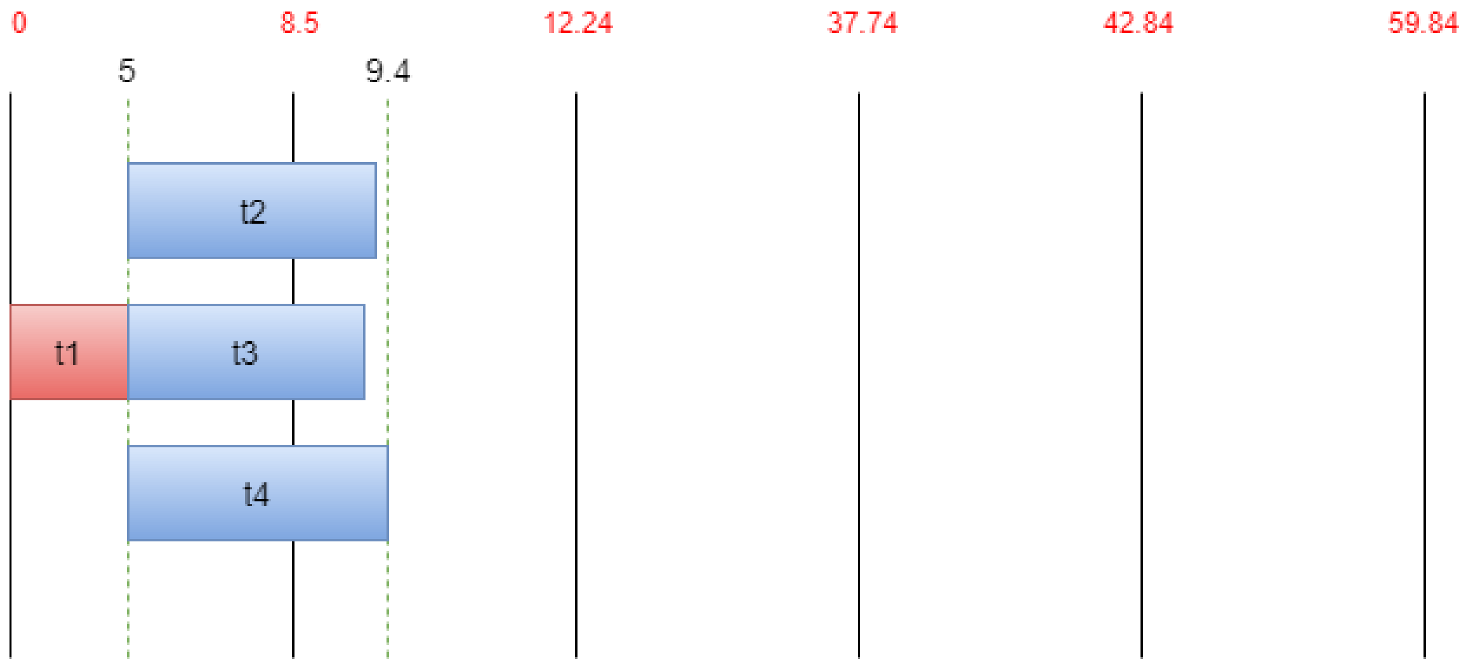
Sub-deadlines and resource allocation



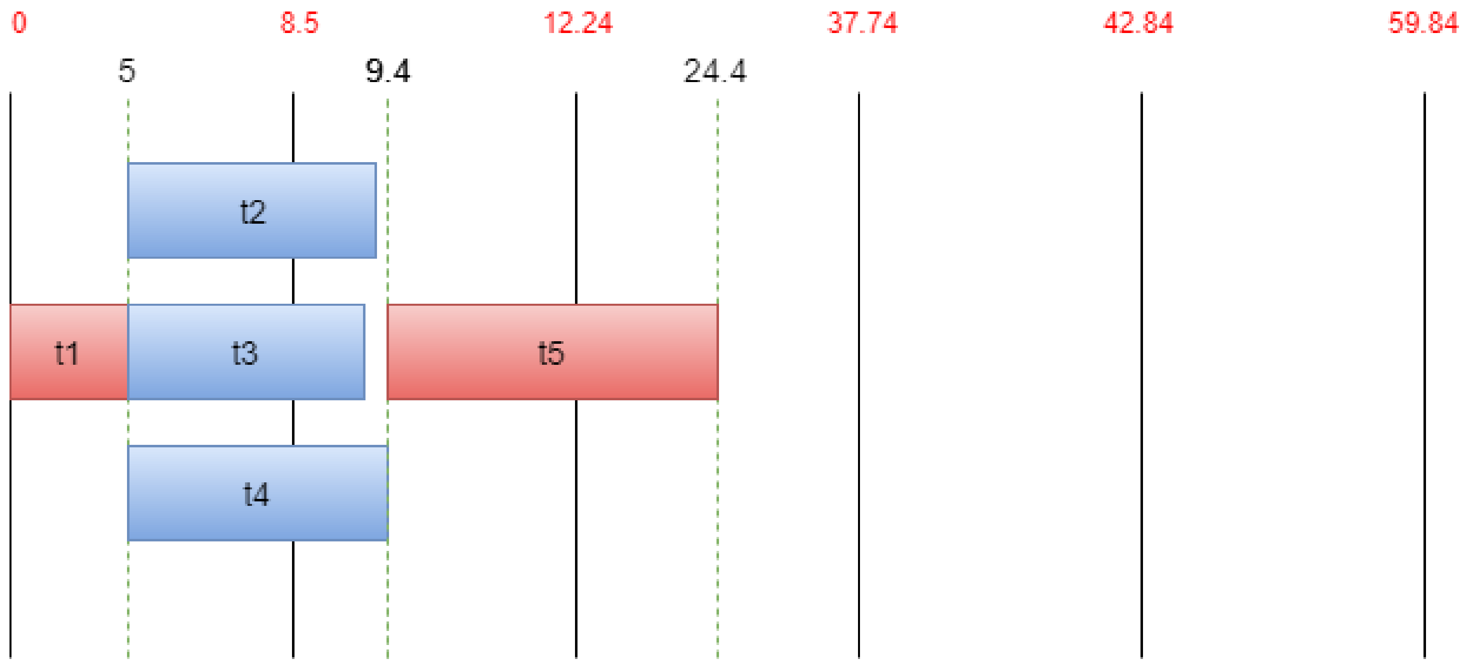
Schedule



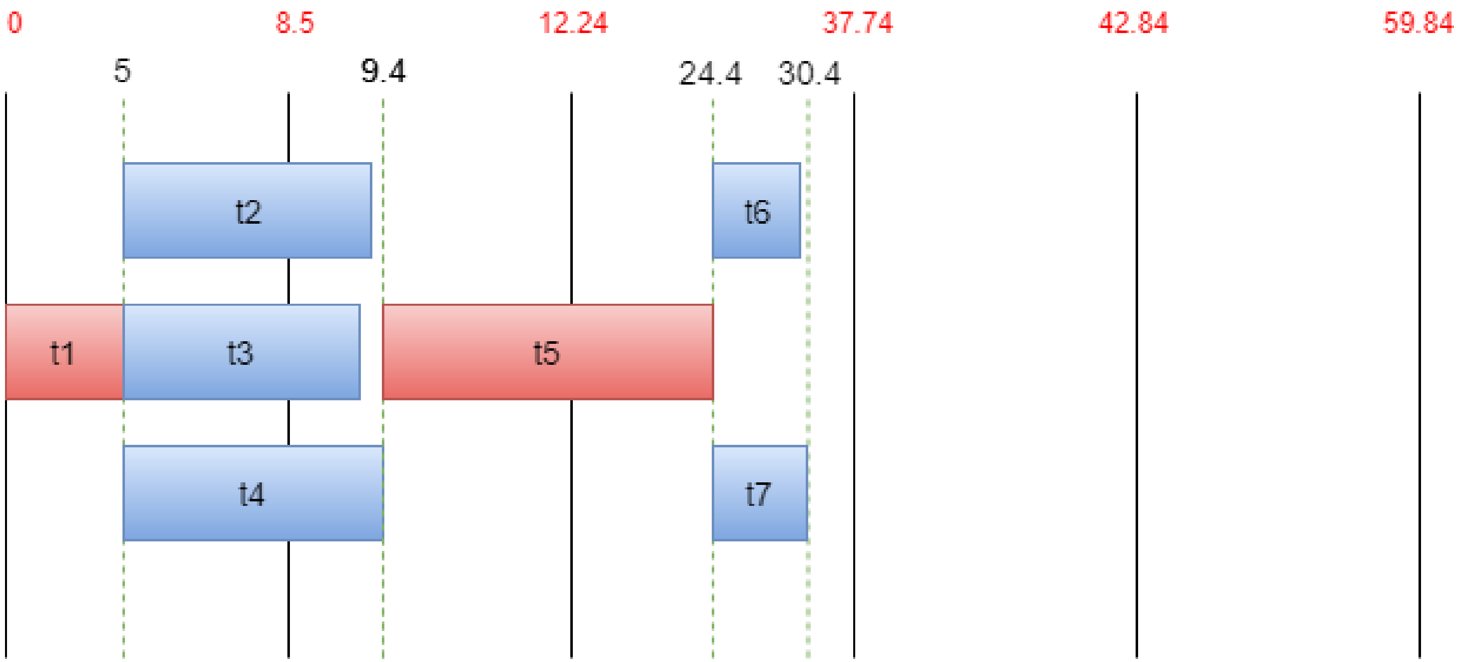
Schedule



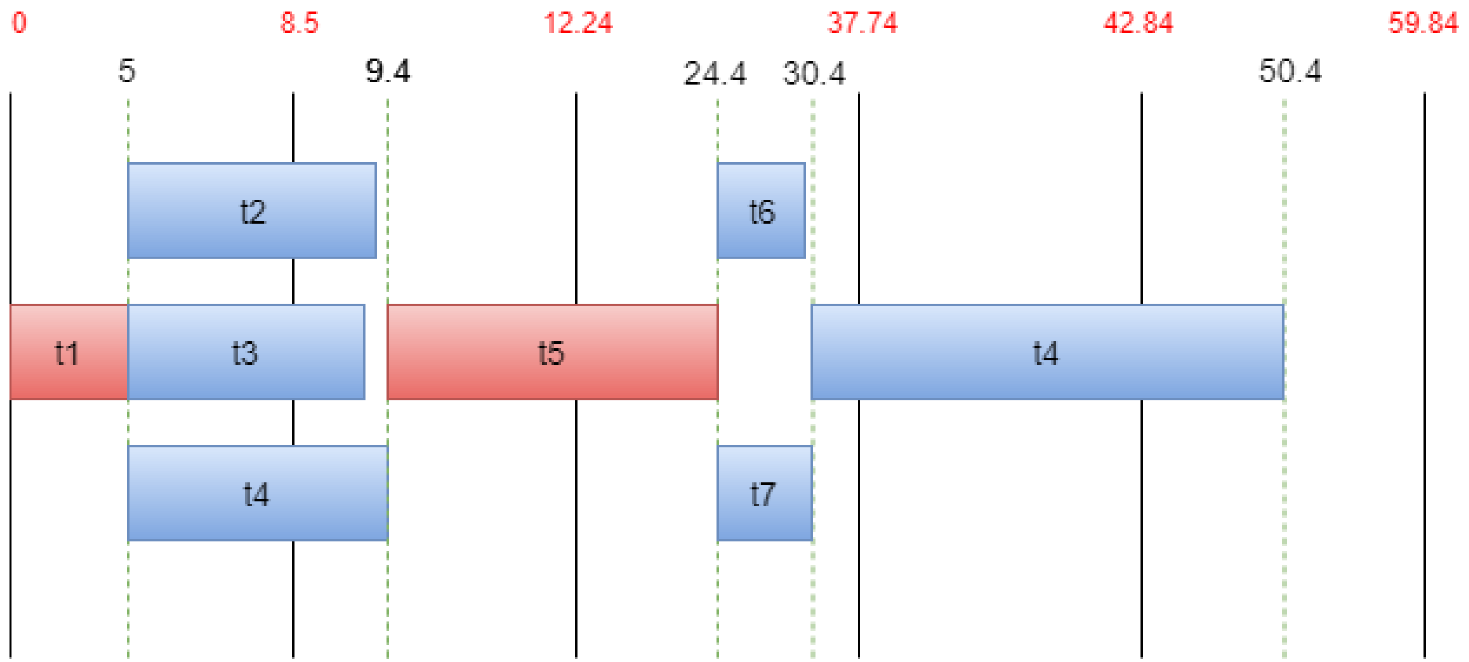
Schedule



Schedule



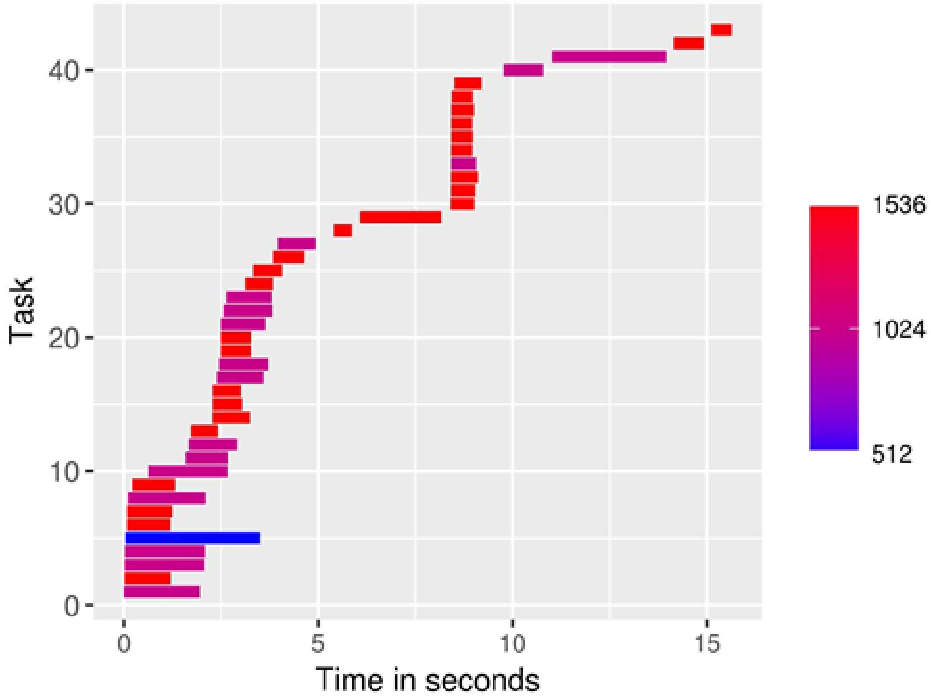
Schedule

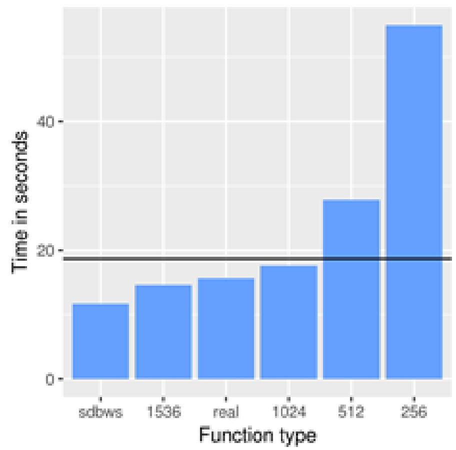


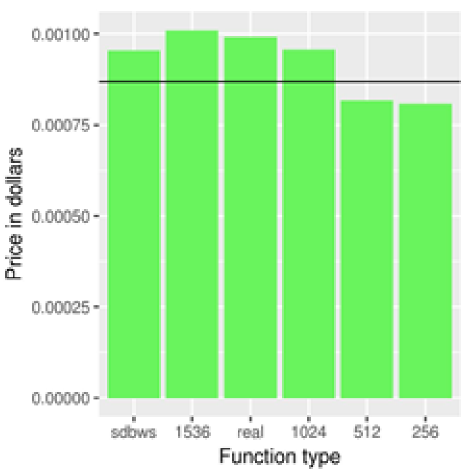
Tests on AWS Lambda

- Montage workflow, 43 tasks
- Function size: 256, 512, 1024, 1536 MB
- Execution times estimated based on runs on homogeneous resources
- Limits adjusted to fit between minimum and maximum measured values

Experiment 1

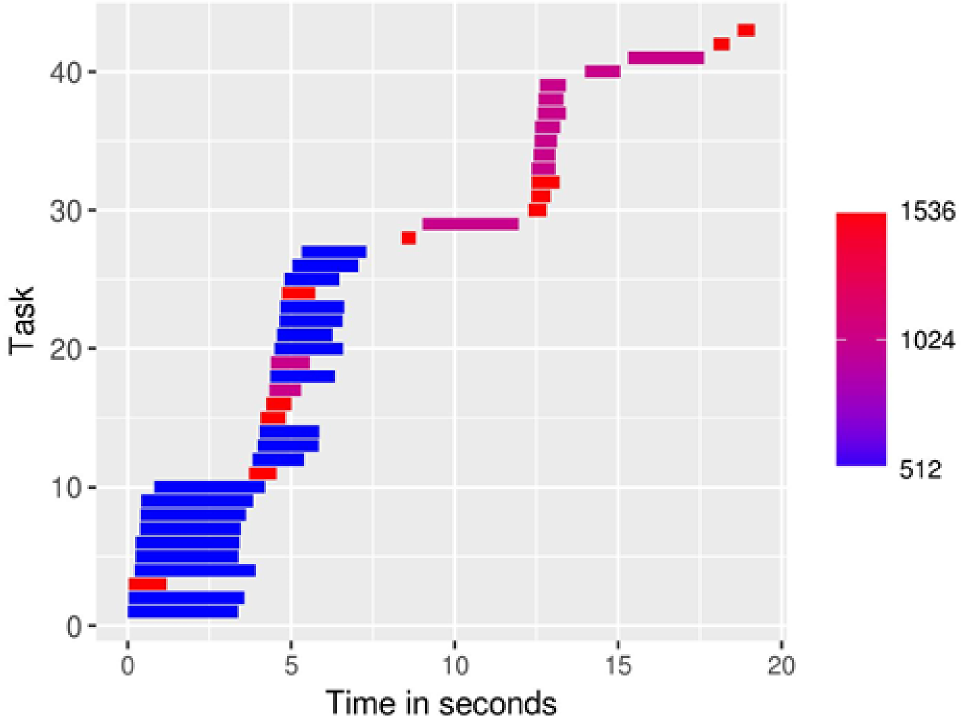


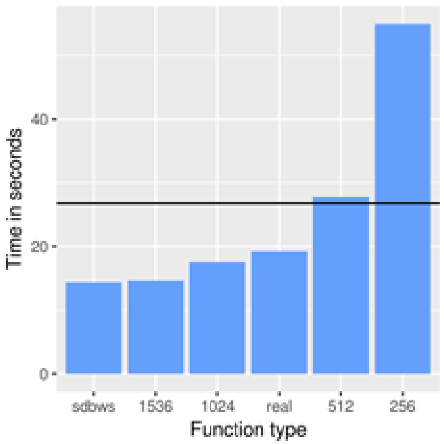


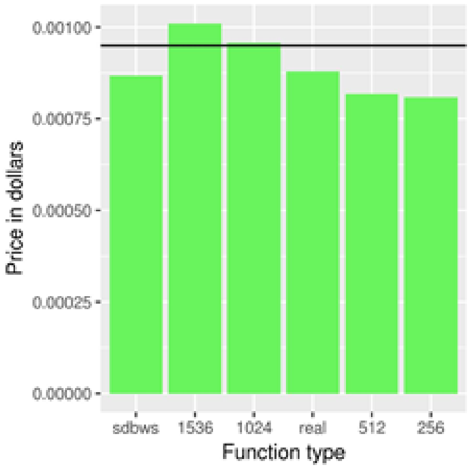


- Deadline: 18,6s
- Budget: $0,00086
- 10,8% fastert han 1024MB
- 3,6% more expensive than 1024MB

Experiment 2

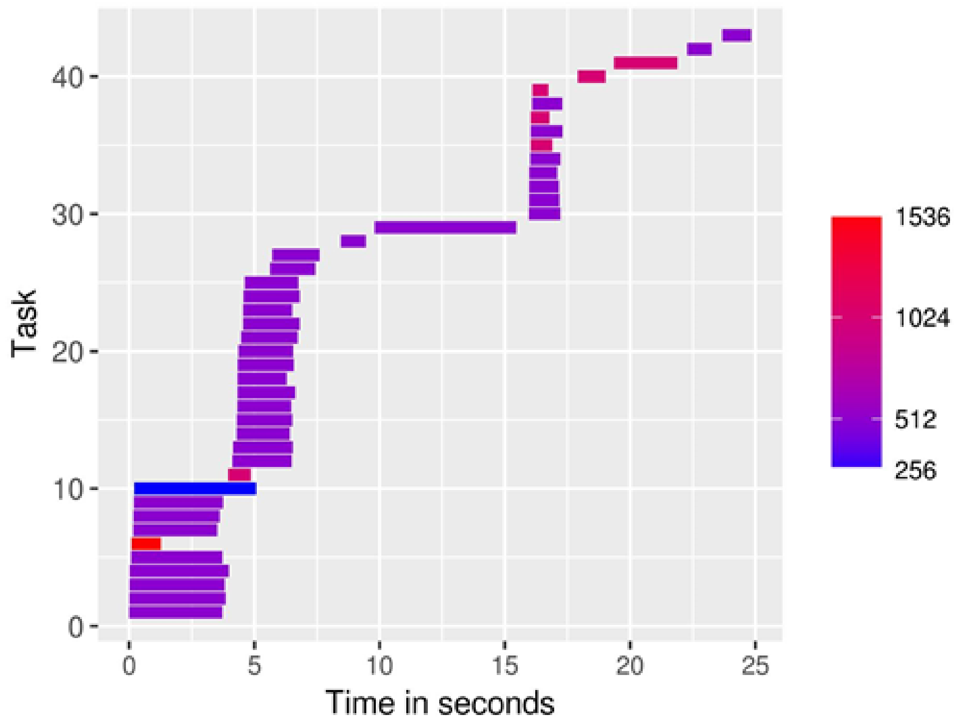


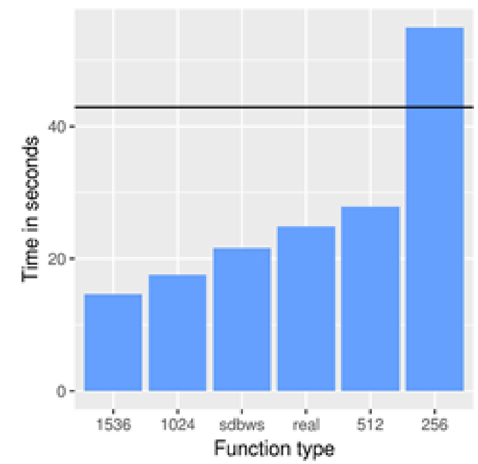


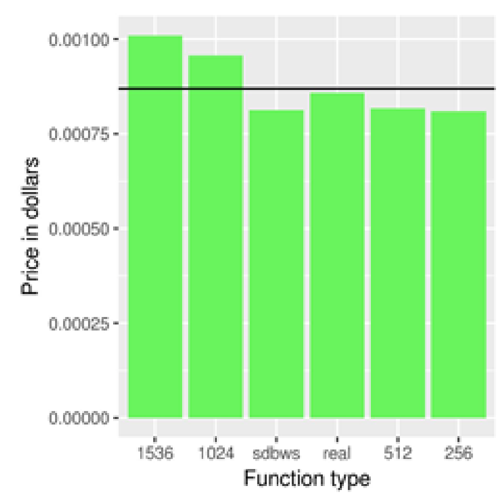


- Deadline: 26,7s
- Budget: $0,00094
- 30,9% faster than 512MB
- 7,4% more expensive than 512MB

Experiment 3







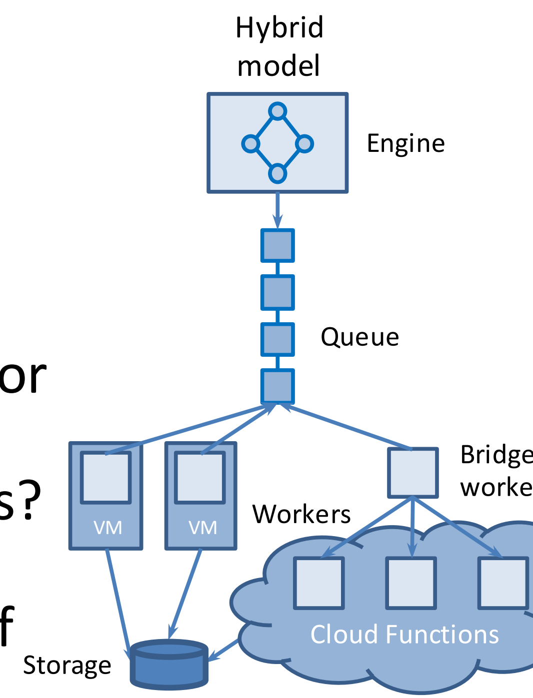
- Deadline: 42,8s
- Budżet: $0,00086
- 10,6% szybciej niż 512MB
- 4,9% drożej niż 512MB

Conclusions

- Serverless and other highly-elastic infrastructures are interesting options for running high-throughput scientific workflows
- Cloud functions are heterogeneous
  - Technologies, APIs
  - Resource management policies (over/under provisioning)
  - Performance variations and guarantees
- Their provisioning model may change the game of resource management
- Experiments with SDBWS show that heterogeneous execution may have advantages, but more tests are needed

Future Work

- Evaluation of parallelism limits and influence of delays
- Key parameter: elasticity
  - How quickly the infrastructure responds to the changes in workload demand
  - How fine-grained pricing can be?
  - Granularity of tasks vs. granularity of resources
- Example questions:
  - Which classes of tasks/workflows are suitable for such infrastructures?
  - How to dispatch tasks to various infrastructures?
  - Can we actually save costs when using such resources (e.g. for tight deadlines/high levels of parallelism)?



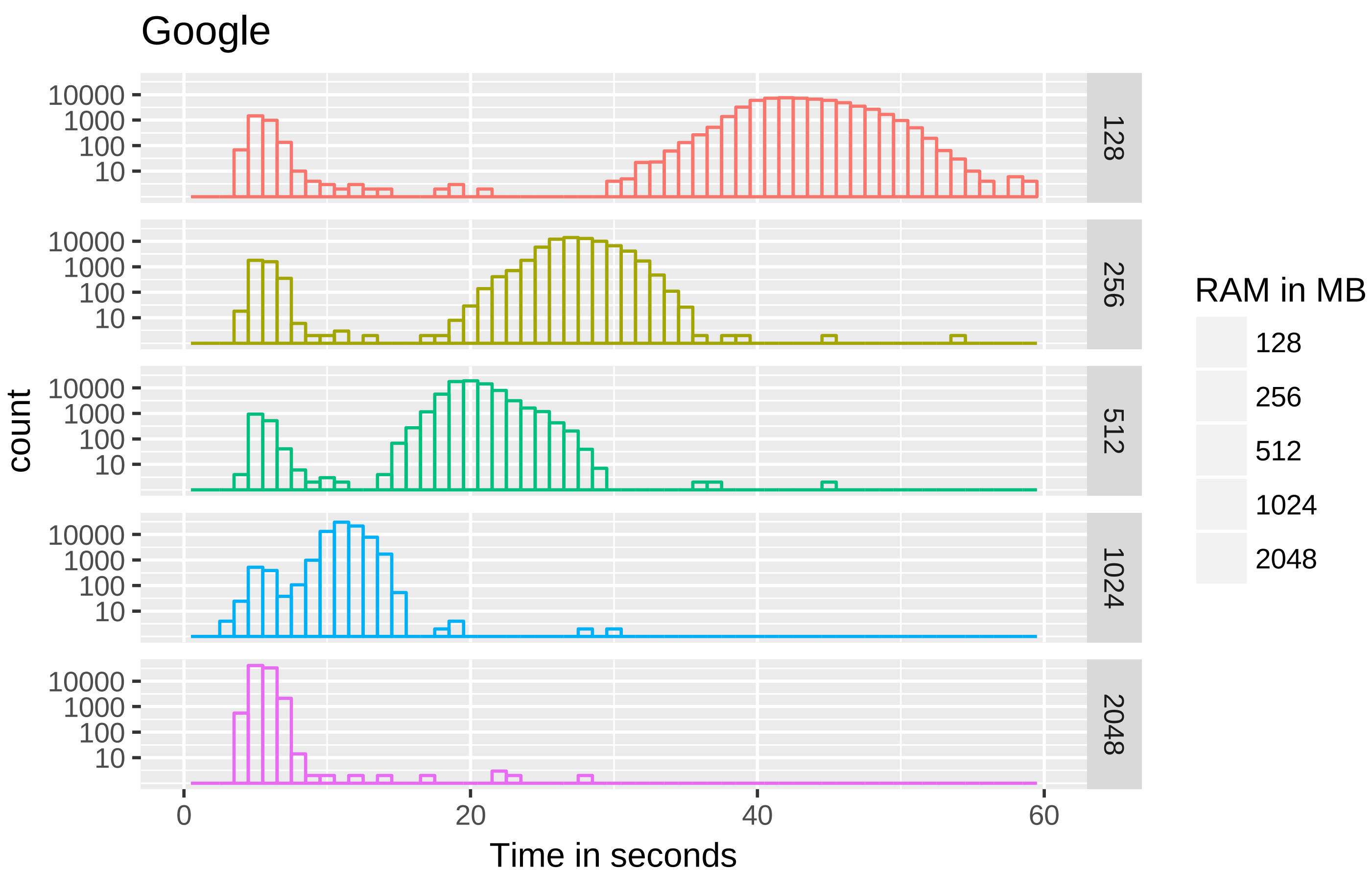
Thank you!

- DICE Team at AGH & Cyfronet
  - Marian Bubak, Piotr Nowakowski, Bartosz Baliś, Tomasz Gubała, Maciej Pawlik, Marek Kasztelnik, Bartosz Wilk, Tomasz Bartyński,
    Jan Meizner, Daniel Harężlak
- Collaboration
  - USC/ISI:
    - Ewa Deelman & Pegasus Team
  - Notre Dame:
    - Jarek Nabrzyski
- Projects & Grants
  - National Science Center (PL)
  - ISMOP (PL)
- References:
  - HyperFlow: https://github.com/dice-cyfronet/hyperflow/
  - DICE Team: http://dice.cyfronet.pl

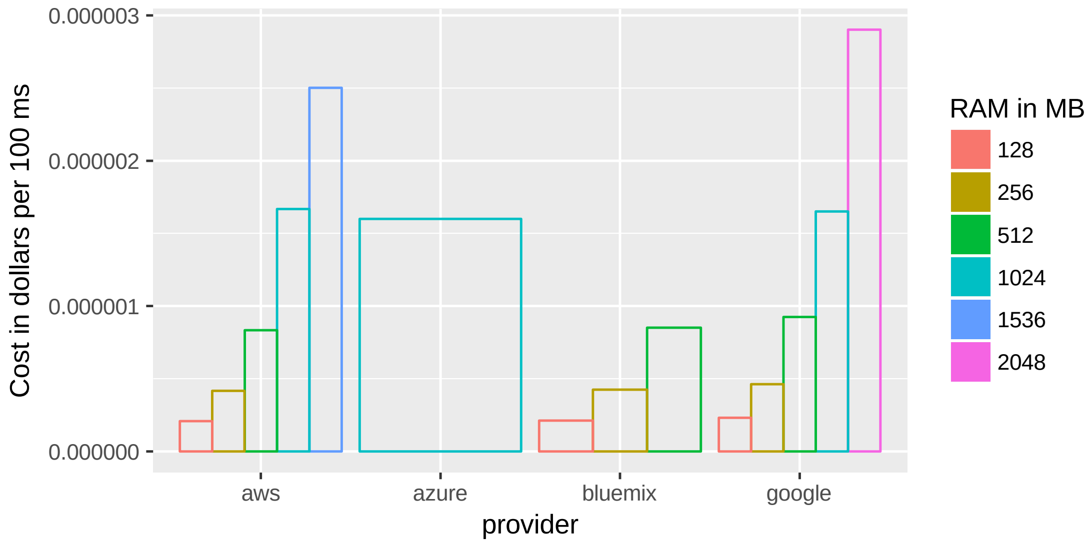
Backup slides

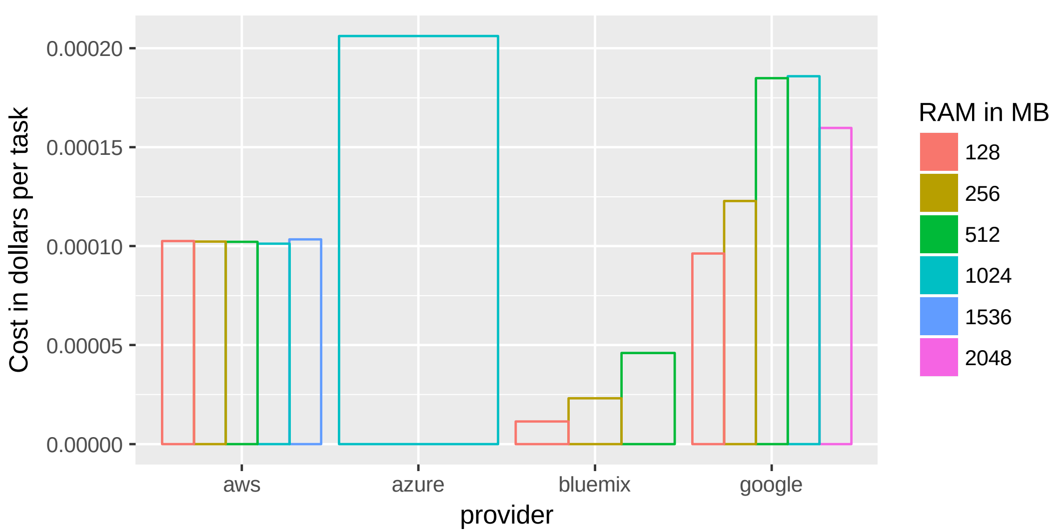
Detailed Google Cloud Functions Preformance Results

- Functions often run much faster than expected
- How often? About 5% times.



Cost analysis





- List price vs. price/performance
- Different models:
  - AWS – proportional
  - IBM – invariant
  - Google: mixed
- For Azure we assume 1024 MB

References

[1] H. Arabnejad and J. G. Barbosa. List scheduling algorithm for heterogeneous systems by an optimistic cost table.
[2] M. Ghasemzadeh, H. Arabnejad, and J. G. Barbosa. Deadline-budget constrained scheduling algorithm for scientific workflows in a cloud environment.
[3] A. Ilyushkin, B. Ghit, and D. Epema. Scheduling workloads of workflows with unknown task runtimes.
[4] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski. Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization.
[5] M. Malawski, K. Figiela, A. Gajek, and A. Zima. Benchmarking heterogeneous cloud functions.
[6] M. Malawski, K. Figiela, and J. Nabrzyski. Cost minimization for computational applications on hybrid cloud infrastructures.
[7] M. Malawski, A. Gajek, A. Zima, and K. Figiela. Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions.
[8] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski. Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds.
[9] B. Baliś. Hyperflow: A model of computation, programming approach and enactment engine for complex distributed workflows