SCC: Project

# General details (REMOVE!)

Topic: Multi-objective task scheduling system

Strategy: MOHEFT [1]

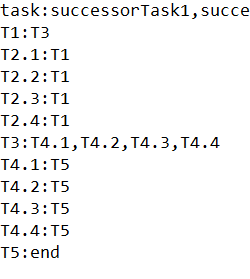
Workflow: Custom

Objectives: makespan & cost

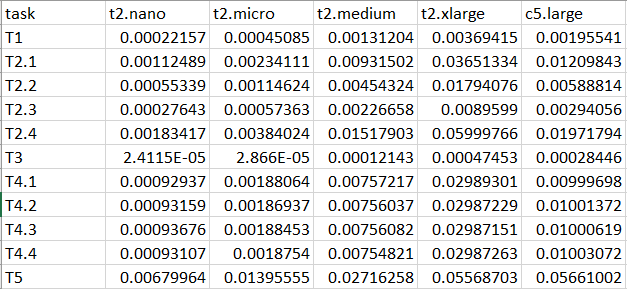
# OBJECTIVE

With this project, we want to implement a real-world workflow scheduling algorithm and evaluate its performance scheduling a custom workflow which could be executed on Amazon EC2 instances. We use the workflow scheduling algorithm MOHEFT [1] to schedule the execution of our workflow while optimizing multiple objectives, namely makespan and cost.

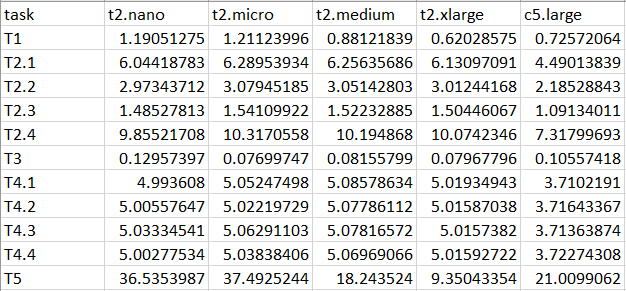
# INPUT

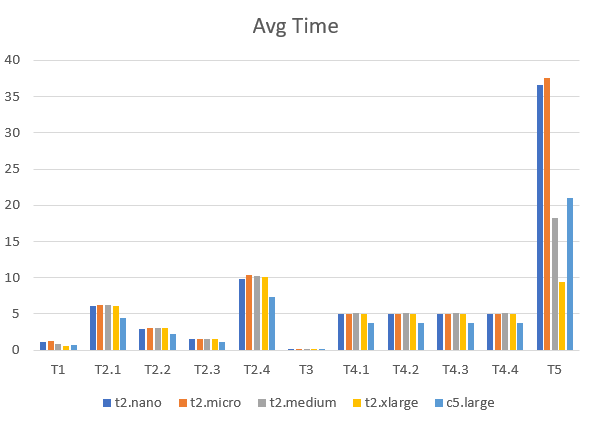
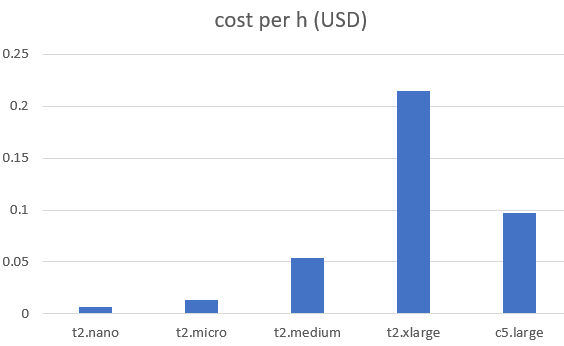
We performed benchmarking on EC2 instances of the types t2.nano, t2.micro, t2.medium, t2.xlarge and c5.large in order to establish the makespan of the individual tasks on different instance types. Next, we combined the makespan information (makespan.CSV) with the price per hour for the instances to arrive at the cost of execution for every task on every instance type (costs.CSV). A textual representation of the DAG representing our workflow, together with makespan.CSV and costs.CSV forms the decision base of our approach.

Costs.CSV



Makespan.CSV





# diagram Overview of approach



# IMPLEMENTATION

All presented artefacts were developed and evaluated on a Virtual Machine running the operating system “Ubuntu 16.04.4 LTS”. The used workflow scheduling algorithm based on MOHEFT, as well as the calculation of the Nadir point and Utopia point were realized using Java 10. For the calculation of the Hypervolume (HV), we reused a Python-based script developed by Simon Wessing (TU Dortmund University) [2]. The “run script” which executes the workflow scheduling algorithm and calculates the hypervolume of the results of a run are incorporated in a single Python script (run.py), which encompasses the whole evaluation process (execution, HV-calculation, comparison of results) depicted in the overview-figure above.

# TASK SCHEDULING ALGORITHM

We based our task scheduling algorithm on the Cloud-aware MOHEFT algorithm presented in [1].

# Evaluation

1. Calculate optimal solution (feasible due to small problem)
2. Perform multiple runs of the task scheduling algorithm to compute approximated solutions
3. Compare solutions from different runs with hypervolume
4. Evaluate distance between best approximation and optimum

# RESULTS

The execution of the workflow scheduling algorithm yields several trade-off solutions, which form a pareto-front and additionally have an adequate crowding distance. Using a maximum number of trade-off solutions (K) of 6 (graph above), the hypervolume of the best solution is 82.9% of the optimum hypervolume (utopia point to nadir point). Increasing K to 1000, the hypervolume of the best solution increases to be 85.74% of the optimal hypervolume. Even larger values for K only lead to marginal improvements.

# EXAMPLE USAGE



# EXAMPLE USAGE (continued)



# References

|  |  |
| --- | --- |
| [1] | J. Durillo and R. Prodan, “Multi-objective workflow scheduling in Amazon EC2,” *Cluster computing,* vol. 17, no. 2, pp. 169--189, 2014.  [2] Pure Python-based Hypervolume calculation script by Simon Wessing (TU Dortmund University), obtainable from https://ls11-www.cs.tu-dortmund.de/rudolph/hypervolume/start |

It should be more like a combination of a technical paper and program documentation, where full description of the developed scheduling algorithm should be presented.

Within the 6-8 pages, you should shortly describe the problem area. Then you should proceed with short definition of the problem that you want to solve.

Then you should present your approach. This description should contain detailed algorithm.

Afterwards you proceed by giving details on the implementation of the algorithm (such as which programing languages and technologies you used).

At the end you simply finish with a simple running tutorial and simple execution results.