Vrije Universiteit Amsterdam

Bachelor Thesis

A Trace-Based Validation Study of OpenDC

Author: Jaro Bosch (2598955)

1st supervisor: ir. Laurens Versluis
2nd reader: prof.dr.ir. Alexandru Iosup

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

December 2, 2020

Abstract

With the rapid digitalization of our society and our increased reliance on online services, comes a rising strain on the data centers that make all of this possible. For this reason, it is valuable for datacenter managers to be aware of a good configuration for their particular datacenter, not only to ensure good end-user results, but also to increase efficiency in terms of energy consumption, utilization, and maintenance cost among others.
To gain such insights without having to actually reconfigure and test the entire datacenter, practitioners and scientists can use simulators to configure and test datacenters virtually. One such simulator is OpenDC, developed by the AtLarge research group, which specializes in massivizing computer systems. OpenDC is a platform which can be used to model datacenters to explore aspects such as resource management and operation efficiency.
It is important to know if OpenDC can accurately simulate the operations of a datacenter, but so far this remains a challenging goal.

The purpose of this thesis is to assess the accuracy of OpenDC when simulating datacenter operations. To this end, we present a general validation of OpenDC, where we explore the effects that different parameters have on the simulation result and whether OpenDC operates as expected. Moreover, we present a validation of OpenDC's simulation results, in which we select existing relevant and workload-driven examples of datacenter operations, and compare metrics of these workloads to metrics of these workloads simulated by OpenDC. Furthermore, we construct a novel workload based on operations executed on the Galaxy Project Europe server, a public biomedical workload cluster. This workload is different from other workloads which have previously been used on OpenDC. This ensures an additional perspective when it comes to validating OpenDC.
In this thesis we found that OpenDC generally functions as one would expect it to. However, its simulation results often lack in accuracy.

Contents

1 Introduction **4**
1.1 Problem Statement 4
1.2 Research Question(s) 4
1.3 Research Approach 5
1.4 Main Contribution 5

2 Background **6**
2.1 OpenDC 6
2.2 Workflows 7
2.3 Workflow traces 8
2.4 Galaxy trace 8

3 OpenDC validation design **11**
3.1 Experiments 11
3.2 Traces used 12
3.3 Topologies and Setup 12

4 Experimentation **14**
4.1 Trace Analysis 14
4.2 General Validation 17
4.2.1 Event Validity 17
4.2.2 Internal Validity 18
4.2.3 Parameter Variability 18
4.3 Trace Based Experiments 20

5 Discussion **26**

6 Related Work **28**

7 Conclusion **29**
7.1 Directions for future research 29

1 Introduction

The shift towards an increasingly digital and online society is in full throttle. To accommodate more parts of our daily lives going online, datacenters are a crucial factor. Almost every piece of data we interact with, is stored somewhere on a remote datacenter[1]. Datacenters are also used by the scientific community, where scientists might want to run large and complex computations on so-called clusters. Clusters are groups of resources and servers which act like single systems, providing large amounts computational power.
To make sure a datacenter functions efficiently, it is important for datacenter engineers to consider the physical layout of its racks, which contain the components. Factors such as airflow can have a large impact on performance[2]. However, designing a layout can be a difficult task, partly because of the fact that it is often costly to reconfigure the datacenter. This means that an adequate layout cannot be found through simple trial and error by physically rearranging components.
This problem can be solved with datacenter simulators, in which the user can create custom layouts and simulate datacenter operations. Based on these operations, the user can make changes to optimize efficiency, performance, resource utilization and other important factors. in this thesis we focus on a specific simulator, namely OpenDC.

1.1 Problem Statement

OpenDC[5] is an open source datacenter simulator developed by the AtLarge research group[10], which is specialized in massivizing computer systems. For datacenter engineers to use OpenDC in practice, it should be able to realistically simulate the inner workings of a datacenter. However, whether this is currently the case is unknown. This research aims to answer that question.

1.2 Research Question(s)

To assess whether OpenDC can provide realistic simulations, we ask the following research question to validate OpenDC:

Research Question
Can OpenDC realistically simulate a datacenter? To help us answer this question we asked the following sub-questions:

Sub-question 1: How realistic is OpenDC's internal behavior?
By answering this question we aim to get more insight into how OpenDC functions internally, and whether OpenDC can accurately model a datacenter and its operations. This question is more focused on which factors have an impact on the simulation results and if this is according to what one would expect.

Sub-question 2: How realistic are OpenDC's simulation metrics when compared to metrics from operations in a physical datacenter?
By answering this question we aim to get more insight into how OpenDC's simulations compare to operations in physical datacenters. We want to achieve this by simulating workloads in OpenDC, which were originally executed on compute clusters. We use three existing workloads, and a newly created one, all of which are introduced in §3.2. We try to model the physical datacenter into OpenDC as accurately as our situation allows us.

1.3 Research Approach

To answer the first sub-question we will perform different validation experiments as proposed by Sargent[3]. The experiments we will conduct concern event validity, internal validity, and parameter variability. To test event validity we verify whether all tasks in the workload actually start, finish, and adhere to their dependencies when simulated by OpenDC. For internal validity we test whether or not simulation results differ when the exact same experiment is repeated. To test parameter variability we alter input parameters to observe how they relate to OpenDC's simulation results. For these experiments we use existing workloads as an input for OpenDC.
To answer the second sub-question we will compare metrics of workloads simulated by OpenDC with metrics of the original execution of the workload. For this comparison we use three different existing workloads and one which we constructed with execution data from a public compute cluster. For the newly constructed workload we can increase OpenDC's degree of realism by assigning the tasks in the simulation similar resources as the tasks received in their original executions. We primarily compare the simulations to their workloads based on runtimes.

1.4 Main Contribution

In Chapter 2 we provide background information on the topic. The main contribution of this thesis is a validation study of OpenDC. This is achieved by both validating OpenDC's fucntionality as well as validating its results. Our validation setup is described in Section 3, after which we present the results in Chapter 4. These results can be used by the OpenDC team in their effort to increase OpenDC's functionality and simulation results. Moreover, the framework for all experiments is publicly available for future use. In Chapter 4 we also highlight a new scientific workload trace, obtained from the Galaxy Project's EU-server. This trace is published on the Workflow Trace Archive[4].
We discuss the results of our experiments in Chapter 5, which show that OpenDC's internal behavior functions as expected. However, OpenDC's simulation accuracy is left to be desired. Here we also touch upon the limitations of this thesis. Chapter 6 discusses related work in the field. In Chapter 7 we answer our main research question and discuss future work.

2 Background

The following section introduces some necessary concepts for the remainder of this thesis. Additionally, we describe the construction of a newly obtained workload trace in detail.

2.1 OpenDC

This research is mainly concerned with OpenDC, which is a tool that can be used to simulate datacenters[5]. Developed by the Atlarge research group, it provides a platform to explore different datacenter topics, both for professional and educational environments.

OpenDC consists of four main parts[5], also depicted in figure 1: a frontend, a web-server, a database and a simulator.
To run a simulation on OpenDC, one needs a topology and a workload in the form of a trace. A topology is a specification of the layout of the datacenter that is being modelled. It can contain different rooms with racks which can in turn house a number of machines. These machines are then specified in terms of RAM and CPU model. We expand on the concept of traces in §2.3. All these parameters can be specified through the GUI on the frontend.
The frontend in turn communicates with the web-server which processes the generated requests, in the form of experiments in the database.
The simulator monitors the database and simulates the experiments using the specified topology and workload. Afterwards, the simulator writes the results back to the database, which can then be retrieved by the web-server.

In this thesis we focus on the simulator part of OpenDC, because this component is responsible for the actual simulation of the workload.

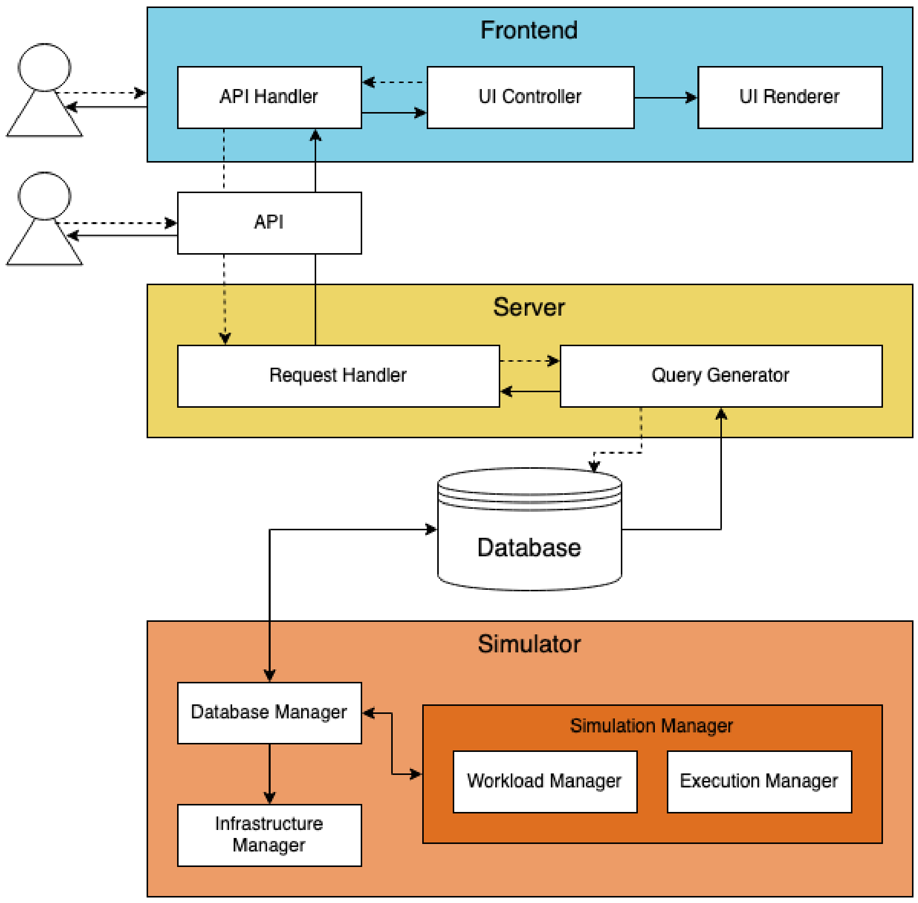

Figure 1: OpenDC's components.

2.2 Workflows

A workflow is a series of subsequent operations, where the output of one operation often functions as the input of the following operation. Usually workflows are used to automate tasks which have to be repeated regularly with different data. A researcher might, for example, retrieve data from a database, re-format the data, and run an analysis on it. Workflows can be used to concatenate these tasks so that the researcher does not have to initiate them. Workflows can consist of a single simple task or many complex tasks, which can run both in parallel or in a single line of execution.
We use the workflow model defined by Coffman and Graham[6], in which a workflow is defined as a Directed Acyclic Graph (DAG). A DAG consists of directed edges and vertices, without containing cycles. The vertices in a DAG represent the tasks in a workflow and the edges represent a connection where the output of the first task acts as the input of the predecessor. There are virtually no restrictions on the number of parents(dependencies of a specific task) and children(dependents of a specific task) that a task has. Figure 2 shows an example of a workflow represented as a DAG.

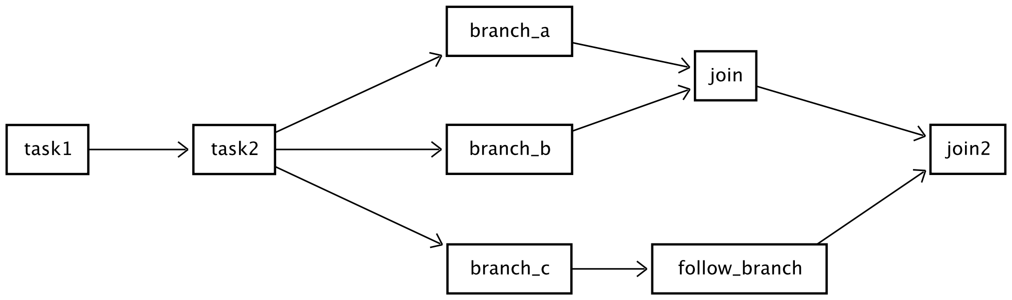

Figure 2: Example of a workflow represented as a DAG.

2.3 Workflow traces

A workflow trace essentially describes a particular execution of a workflow. This is achieved by capturing different metrics associated with the execution. Some useful metrics which are often captured in a trace are: submit time, wait time, start time, duration and the number of tasks.

A trace offers valuable insight into an execution of a workflow, which can prove useful in multiple ways. One of which is the reproducibility of experiments. When researchers publish a trace of a workflow that they executed, that trace contains information about that specific execution. This information can then be used[7] by another researcher who might want to recreate the experiment to potentially verify the results.
Another use-case for traces is in simulation. A trace can be used to simulate the workload on which it is based originally, which is how traces will be used in this thesis. To validate OpenDC, it is important to use traces which are both realistic and diverse.

In this thesis we use traces hosted in the Workflow Trace Archive (WTA)[4], and a newly constructed biomedical trace. The WTA is an initiative by the AtLarge research group which aims to provide a central place to share valuable and representative workflow traces. All traces used are parquet files in the WTA-format[9].

2.4 Galaxy trace

To properly validate OpenDC, we wanted to simulate workloads on OpenDC which had not been simulated before. To achieve this we constructed a new workload trace from execution data from the Galaxy Project's Europe server. The Galaxy Project is an open platform for biomedical research. Their platform allows users to perform many different operations on data, and turn these into workflows. Galaxy's platform can be used locally on a machine of choice, or users can utilize the web-based platform, using Galaxy's remote computational resources.

Through direct contact with a former admin of the Galaxy Europe server, we were able to obtain over seven years worth of workflow execution data. However, not all workflows could be incorporated in the final workload, as some executions lacked metrics which were essential aspects to the final workload.

We parsed this data into the WTA-format[1], which posed some difficulties as Galaxy's database is large and complex. It was not immediately clear which tables were actually needed to construct the complete workload. Additionally, Galaxy's terminology is slightly different from the terminology that OpenDC uses. In §4.1 we highlight properties of the Galaxy trace and compare it to the other traces used in this thesis.

Figure 3 shows the tables from Galaxy's database which we used to construct the trace[2]. Some fields were not provided as they could contain sensitive user information, these fields are struck trough in the figure. In Galaxy's database, workflows are stored in the workflow table, in which workflow_id links to a number of workflow steps in the workflow_step table. Through the workflow_step_input and workflow_step_connection tables we can retrieve the dependencies of the tasks. The workflow_invocation table contains invocations of the actual workflow executions. One workflow_invocation_id is linked to multiple invocation of workflow steps in the workflow_invocation_step table. Through the job_id field in this table we obtained metrics of a particular task execution from the job_metric_numeric table.

The fact that we had direct contact with a former admin allowed us to gather more execution parameters to be able to mimic Galaxy's execution environment more closely. For example, Galaxy uses a First-Come-First-Serve Scheduling policy. Furthermore, we know the task placement policy which Galaxy uses, as in the amount of resources each task was assigned upon its original execution. Galaxy uses a statically mapped algorithm, meaning that each type of operation is assigned a number of resources statically[3]. Using original execution metrics we aimed to assign tasks in OpenDC similar resources as they received in their original execution. The Galaxy trace can be found on the Workflow Trace Archive.

[1]The script used to parse the data into WTA-format can be found at `https://github.com/atlarge-research/wta-tools/tree/master/parse_scripts/parquet_parsers`.

[2]A complete schema of Galaxy's database can be found at `https://galaxyproject.github.io/training-material/topics/admin/tutorials/database-schema/tutorial.html`.

[3]Galaxy's task placement policy can be found at `https://github.com/usegalaxy-eu/infrastructure-playbook/blob/master/files/galaxy/dynamic_rules/usegalaxy/tool_destinations.yaml`.

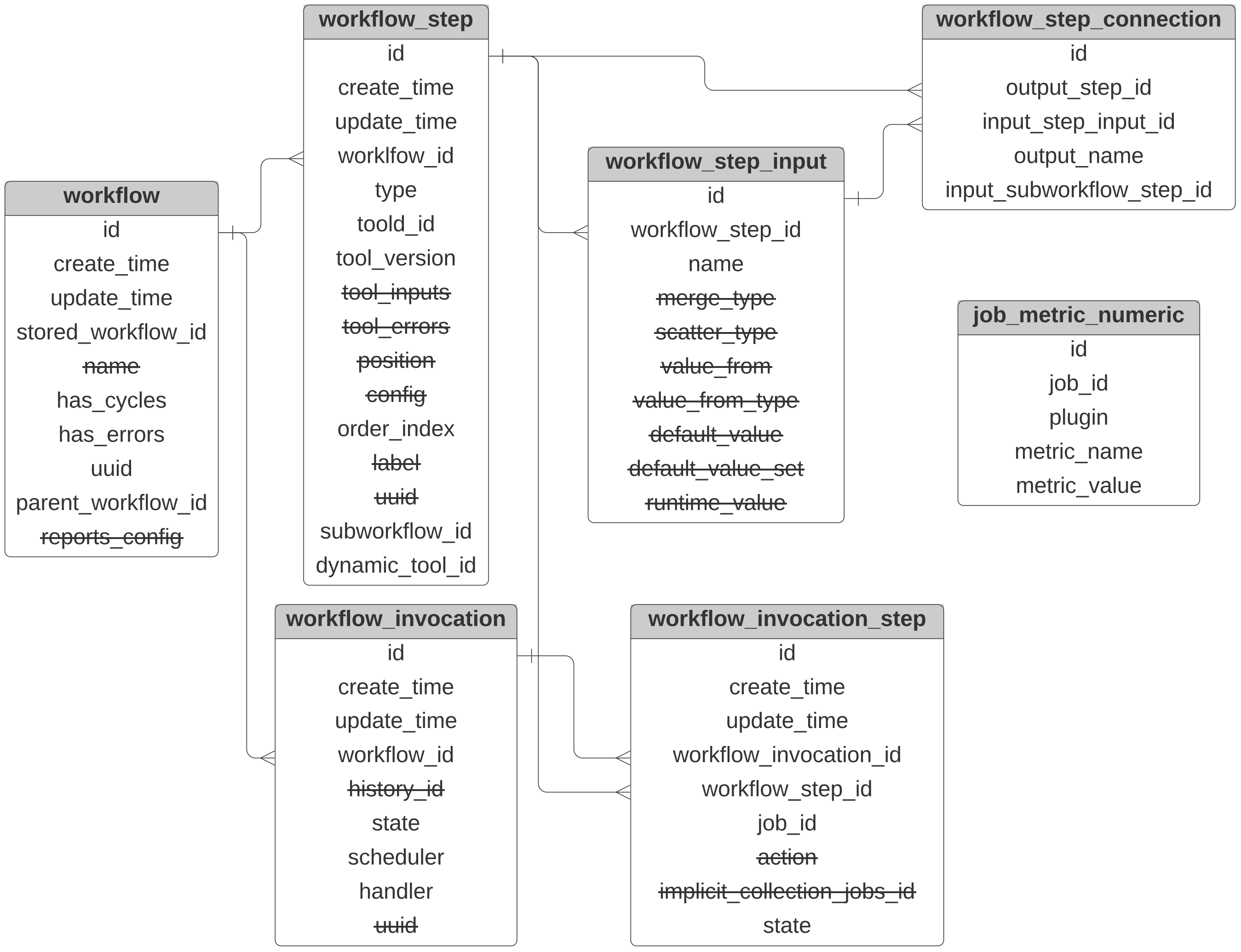

Figure 3: An overview of the tables used to create the Galaxy trace.

3 OpenDC validation design

In the following section we describe our experimental design used to validate OpenDC. This design is also summarised in Table 1.

3.1 Experiments

The experiments we performed fall in two categories; General Validation and Trace Based Validation.

The General Validation is aimed to answer the first sub-question: How realistic is OpenDC's internal behavior? We conduct three different experiments, which are less focused on the output, but more on whether the internals of the simulator behave as one would expect them to. To test Event Validity we explore whether all tasks in the trace start and finish, and whether they do so according to their dependencies. For Internal Validity, we test whether the simulation output varies when running the same simulation multiple times. The metrics we use for this are the task runtimes and the task execution order. For these experiments we use four WTA traces and a single topology. The traces are further described in §3.2. The Parameter Variability test is used to explore the effect certain parameters have on OpenDC's simulation. More specifically, we test how the runtime is affected by these parameters. To run the simulation we use a single WTA trace but slightly altered for each experiment. The same holds for the topology.

With the Trace-based Validation, our aim was to answer the second sub-question: How realistic are OpenDC's simulation metrics wen compared to metrics from operations in a physical datacenter? We perform experiments for each individual trace, but now we put more emphasis on the accuracy of the simulation, specifically the runtime of workflows. We selected this metric as it is a simple yet important aspect of workloads as well as datacenter operations. We compare the output of the simulation to the trace we used as input to see whether the simulation results are accurate. For these experiments we use all four traces and a single topology.

Table 1: Summary of the experiments carried out in this research.

Experiment Category	**Experiment focus**	Inputs	Repetitions	Metrics
General Validation	Event Validity	4 traces, 1 Topology	5	-
	Internal Validity	4 traces, 1 Topology	5	runtime, execution order
	Parameter Variability	1 varying trace, 1 varying Topology	1	runtime
Trace-Based Validation	Comparison	4 traces, 2 Topologies	1	runtime

Table 2: A summary of the traces used in this thesis.

Workload	Domain	Workflows	Tasks
Galaxy	Biomedical	804	5,395
Askalon EE	Engineering	3,552	121,891
Shell	Industrial	3,403	10,208
Pegasus P7	Scientific	38	2,757

3.2 Traces used

To perform the validation of OpenDC, we carried out the experiments using three different publicly available traces, and one which we obtained directly from the Galaxy Project's[8] EU server. All are summarised in Table 2. The three publicly available workloads were all obtained from the WTA.
The Askalon workload consists of traces of workloads executed on the Askalon cluster. The next workload is a set of traces obtained from Shell's Chronos production environment[4]. The third workload consists of traces obtained from the Pegasus Workflow Engine[11].

To properly validate OpenDC we needed realistic traces representing workflows executed on real clusters. Alongside the WTA traces we constructed a novel workload, based on execution data from the Galaxy Project's Europe server. The fact that the platform is free to use by anyone, ensures that the traces obtained from it are very diverse in terms of task count, runtime and the amount of resources each tasks requires.

Furthermore, we also know the amount of resources each task receives in terms of core count and Gigabytes of memory. For this reason, we can run simulations in a more realistic environment. However, due to a lack of information we can only make an educated guess as to the specific CPU model used by Galaxy[12].

In §4.1 we compare the different traces in more depth, with an emphasis on the Galaxy trace.

3.3 Topologies and Setup

For the Event Validity and Internal Validity experiments we used a datacenter topology with 32 machines, each containing an Intel i7 CPU with 4 cores and a clockspeed of 4.1GHz. This CPU is selected as it is one of the two CPU's OpenDC currently supports on its web-based platform. From here on, we will refer to this topology as the 'standard topology'.

We used two topologies for the Parameter Variability experiments, the standard topology, and a topology where we increased the number of CPU's after each iteration. All contain the same Intel i7 CPU's as mentioned before.

In the trace based validation experiments we used a custom topology for the Galaxy trace. This topology consists of 72 machines with Intel Xeon 4850 CPU's with 20 cores and a clockspeed of 2GHz. For the experiments with the other traces we used the standard topology. The experiments are performed on the OpenDC version used for the SC18 article[13] with some added functions to gather the necessary metrics.

4 Experimentation

The following section presents the results of the experiments we conducted to validate OpenDC.
Firstly, we present an in-depth comparison between the traces that were used, with an emphasis on the Galaxy trace. Secondly, the results of the General Validation experiments are presented. Finally we present the results of OpenDC's result validation.

4.1 Trace Analysis

To explore the distinctive features of the Galaxy trace, we compare it to the other three traces used in this research.
Figure 4 shows the runtime distributions of all four traces used in this thesis. We define the runtime of a workflow as the sum of the runtimes of all its tasks. Figure 4 shows that the Galaxy trace provides a unique addition to the set of traces in terms of its workflow runtime distribution. It has some similarities to the Askalon trace shown by Table 3, where they have similar runtime mean, minimum and maximum values. However, Table 2 shows that the Askalon trace contains more workflows and tasks. In addition, the traces both represent a different domain.

In terms of makespan, the Galaxy trace has no similars and contains the broadest range of the four traces, as shown by Figure 5. In this thesis we use the same definition of makespan as Ilyushkin[14], namely; the time between the start of its first task until the completion of its last task. Table 3 also shows that the minimum and maximum makespan values are equal to the corresponding runtime values. This implies that both the smallest and largest workflow in the trace are in single lines of execution.

Table 3: Detailed look at the traces used. All measurements are in milliseconds.

Workload	Runtime mean	Runtime min	Runtime max	Makespan mean	Makespan min	Makespan max
Galaxy	5,361,791	4,000	357,238,000	3,629,349	4,000	357,238,000
Askalon EE	4,405,234	1	531,173,108	613,449	1	43,101,128
Shell	26,221	13,337	42,656	10,510	5,191	29,251
Pegasus P7	78,362,552	10,000	672,703,000	5,563,389	5,074	133,557,214

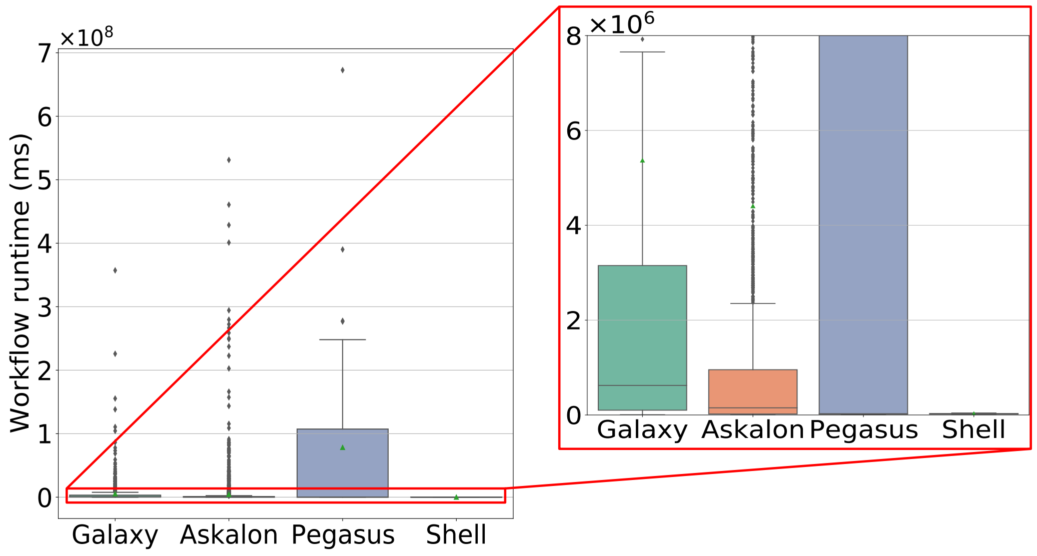

Figure 4: Runtime Distributions.

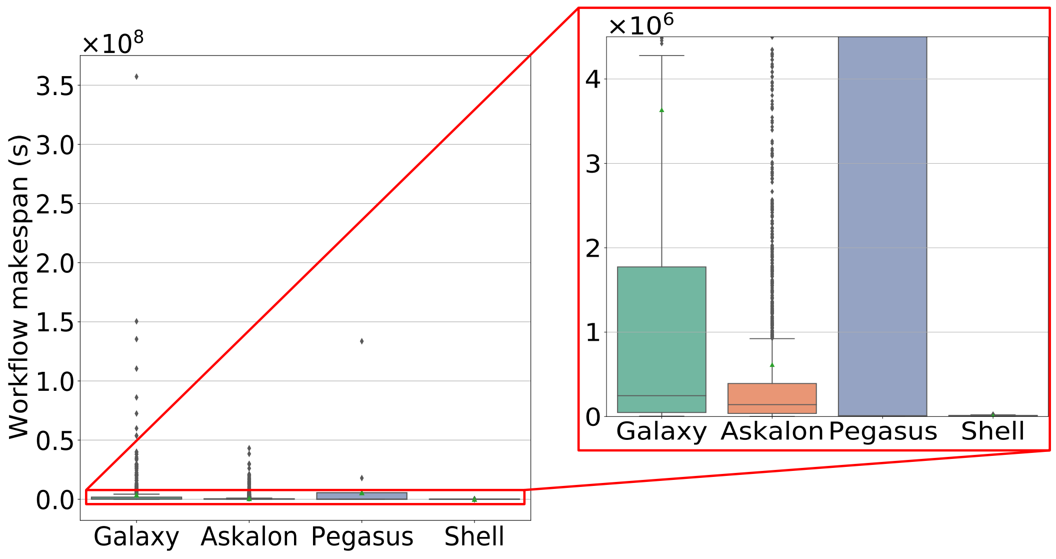

Figure 5: Makespan Distributions.

We can deduce from Table 2 that the Galaxy trace has a relatively low average number of tasks per workflow (~ 7) compared to the Askalon trace (~ 34) and the Pegasus trace (~ 73). Only the Shell trace has a lower average; ~ 3. Figure 6 gives additional insight into the distribution of the number of tasks per workflow. The Figure shows that the Galaxy trace has a relatively small range of tasks per worklfow, with the majority of the workflows consisting of less than fifty tasks. This is in stark contrast with the Askalon trace, which has a substantial amount of workflows with more than fifty tasks. Another noteworthy observation is the distribution of tasks per workflow for the Shell trace, which only consists of workflows containing 3 tasks.

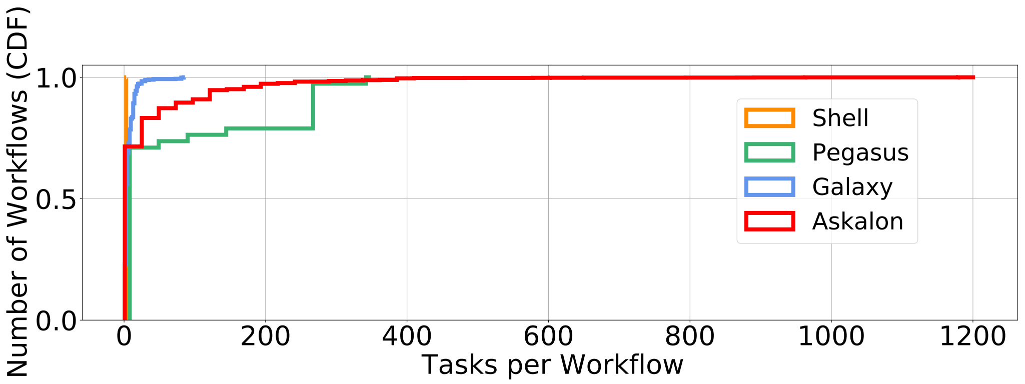

Figure 6: CDF with the number of tasks per workflow.

In terms of workflow structures, the Galaxy trace is similar to the Askalon trace, as shown by figure 7. This figure depicts the workflow structures for each source. We display the workflow structures using the same type of figure as Versluis et al. [4][4]. We also adopt the same workflow structures and their definitions: scatter (distribution of data), shuffle (data redistribution), gather (data aggregation), pipeline (single line of execution), and standalone (process). We see that for the Galaxy, Askalon, and Shell traces the majority of the workflows have a pipeline structure, with the shell trace consisting exclusively of pipeline workflows. The Pegasus trace mainly consists of shuffle and gather workflows.

Overall the Galaxy trace is a diverse workload containing certain properties which make it distinctive on the Workflow Trace Archive. Moreover, the Galaxy trace is the first trace from the biomedical domain on the WTA.

[4]The code to generate this figure can be found on `https://github.com/atlarge-research/wta-analysis/tree/master/workflow-structure-analysis`.

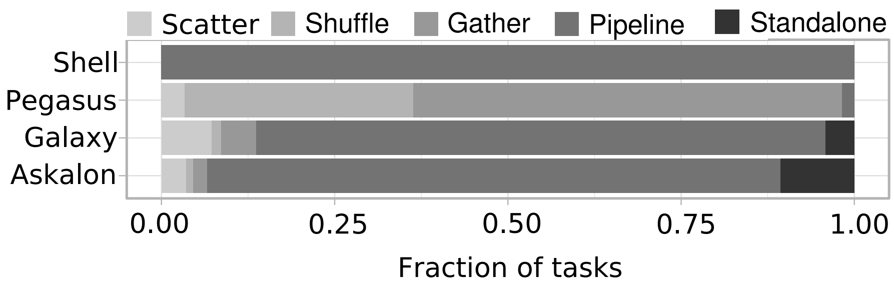

Figure 7: Workflow structures per source.

4.2 General Validation

In this section we describe the experiments we performed to validate the inner workings of OpenDC. Based on work by Sargent [3], we selected three basic validation techniques; Event Validity testing, Internal Validity testing and Parameter Variability testing.
Using these techniques we aim to gain a better understanding of the inner workings of OpenDC and whether it behaves in the way we expect it to behave.

4.2.1 Event Validity

The Event Validity experiment is concerned with whether events in the simulation correspond to events in the real system, which, in our case, is represented by a workflow trace. The events we are looking to validate are the following:

- **Task start**. Here we want to explore whether all tasks present in the trace are actually started in OpenDC.
- **Task finish**. Exploring whether all started tasks also finish.
- **Task dependencies**. Exploring whether all tasks start according to their dependencies. To expand, a task cannot start before its dependencies are also started (and finished).

We used a large part of functionality already offered by OpenDC to capture the events we would like to validate. We implemented checks to verify the number of tasks that started, the number of tasks that finished, as well as the total amount of tasks present in the trace.

To validate whether tasks in OpenDC do not start before their dependencies are finished, we implemented a function which, every time a tasks is started, checks whether the dependencies of that task have already finished. A text message was printed if this was not the case.

We ran these experiments with all four traces. The topology used in this experiment is the standard topology introduced in §3.3. For the Galaxy trace we used the custom topology. The results of these experiments are visualized in Table 4.

Table 4: Results of the Event Validity experiment.

Workload	Started Tasks	Finished Tasks	According to Dependencies
Galaxy	✓	✓	✓
Askalon EE	✓	✓	✓
Shell	✓	✓	✓
Pegasus P7	✓	✓	✓

As Table 4 shows, OpenDC started all tasks, finished all tasks, and adhered to the dependencies of all tasks.

On our initial run of the experiment we found that OpenDC did not execute all tasks that were present in the Galaxy trace. We obtained the tasks which were not executed by filtering out the tasks which were executed from the complete workload. Through manual inspection of the tasks that were not executed, we found a small bug in the Galaxy trace where some tasks contained their own task ID as a dependency. This would cause them to never start, as they would wait for themselves to finish. This bug has since been corrected in the Galaxy trace, after which OpenDC operated as intended.

4.2.2 Internal Validity

The Internal Validity experiment is meant to explore whether OpenDC's output is consistent when running a single simulation multiple times. We added multiple functions in OpenDC's codebase to output the ID and the runtime for each finished task. For this experiment we used all four traces and ran a simulation five times per trace. Here we used Galaxy's custom topology for the Galaxy trace and the standard topology for the other three traces.

After running the exact same simulation five times for each trace, we observed no differences in terms of task execution order or runtimes.

4.2.3 Parameter Variability

In the Parameter Variability tests we tested the effects certain parameters have on OpenDC's simulations. We increased the specific parameters after each iteration, and calculated the average workflow runtime of the simulation. While we focused on one parameter, the other parameters contained their original values. Although some unlikely, the situations that were simulated are not impossible. The values that were scaled up and down were:

- Runtimes of tasks in the trace.
  We increased the average task runtime in the trace from roughly 8 ms to around 2.2 billion ms, over 25 iterations.

- The amount of resources requested by tasks in the trace.
  We increased the amount of resources requested in the trace from 1 to 100 over 17 iterations.

- The number of CPU's in the topology.
  We increased the number of available CPU's in OpenDC from 1 to 48 over 12 iterations.

For this experiment we used an altered version of the Shell trace and two topologies, the standard topology, and one where we increased the number of CPU's after each iteration. For the runtimes and the resources requested we expect see a correlation where if each respective value increases, we see an increase in the average simulated workflow runtime. For the number of available cores in OpenDC, we expect little change given the workflow structures of the Shell trace. Figure 7 shows that the Shell trace exclusively consists of pipeline-structured workflows. This means that tasks in a single workflow will not be able to be executed in parallel, as the tasks have to wait until their dependency has finished. Therefore, the amount of CPU's should not have a large impact on the average simulated workflow runtime. The results of this experiment are visualised in Figures 8, 9 and 10.

Figure 8 shows a linear relationship between the resource amount requested and the average simulated workflow runtime, it also shows that the increase from around one to four cores requested, did not have an impact on the simulated runtime. This is probably caused by the fact that the standard topology still provides enough resources to support the small increase in the requested resources. Figure 9 also shows a linear relationship between the runtime in the trace and the average simulated workflow runtime. Both these result are according to our expectancy of how OpenDC should function.
To examine whether these results were due to the nature of the Shell trace, we ran the same simulations with altered versions of the Askalon trace. These resulted in linear relationships in both respective cases as well.

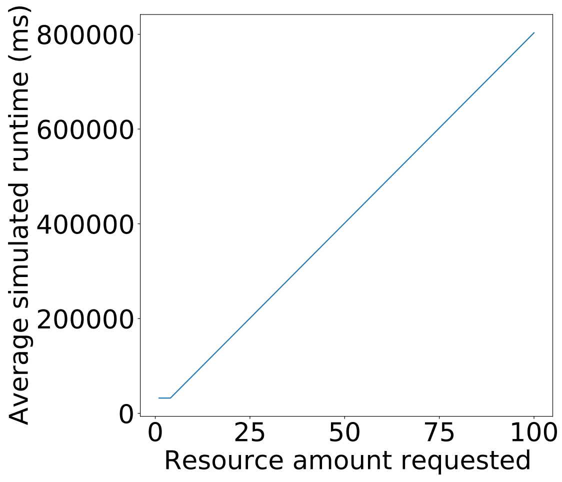

Figure 8: Relationship between the amount of resources requested in the trace and the average simulated workflow runtime.

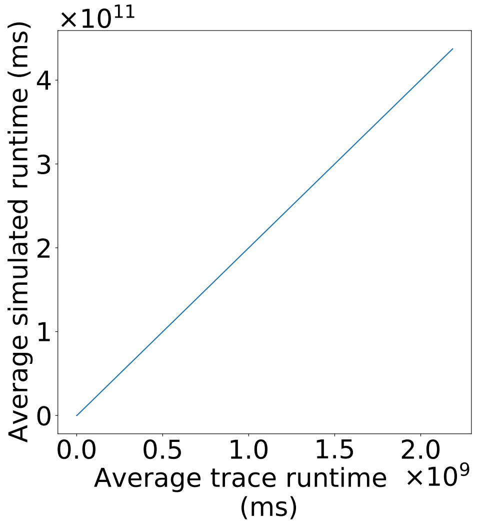

Figure 9: Relationship between runtime in the trace and average simulated workflow runtime.

Figure 10 shows a clear negative exponential relationship between the number of CPU's available and the average simulated workflow runtime. We see that the first adding the first few CPU's caused the simulated runtime to decrease substantially, with it starting to flatten out around 15 CPU's. This is not according to our expectation, as we expected the average simulated workflow runtime not to change significantly. However, this result implies that OpenDC can use multiple CPU's to simulate the execution of a single task as the used Shell trace contains no workflows with tasks which can be executed in parallel.
We ran the same simulations with the Askalon trace to examine whether these results were due to the nature of the Shell trace. We increased the amount of CPU's until we saw minimal change in the average simulated workflow runtime. Figure 7 shows that the Askalon trace contains workflows with a scatter, shuffle and gather stucture, which means that these workflows contain tasks which can be executed in parallel. The results are shown in Figure 11. The pattern which emerges is very similar to the one we saw with the Shell trace. This raises some question as to whether OpenDC attempts to optimize execution by executing certain tasks in parallel.

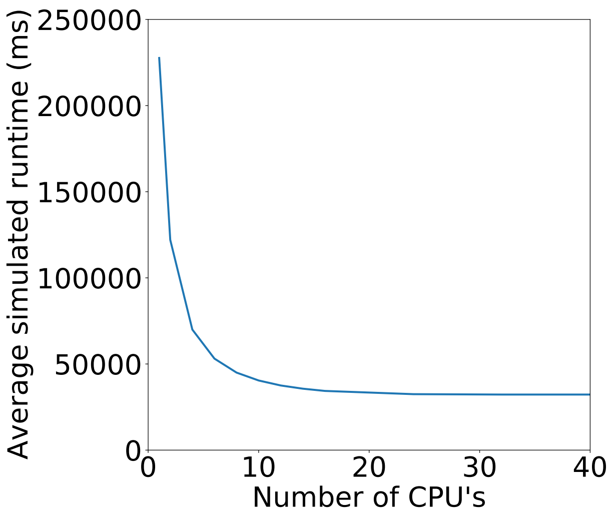

Figure 10: Relationship between the number of CPU's avaialble in OpenDC and the average simulated workflow runtime of the Shell trace.

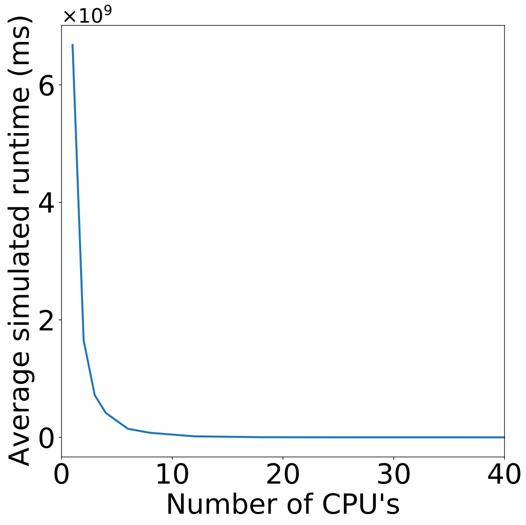

Figure 11: Relationship between the number of CPU's avaialble in OpenDC and the average simulated workflow runtime of the Askalon trace.

4.3 Trace Based Experiments

Using the newly obtained Galaxy trace, we aim to validate OpenDC by comparing the runtime distribution of its simulation to the runtime distribution of the trace itself. Using Galaxy's task scheduling and task placement policy, we can increase the level of realism with which we simulate Galaxy's operations.
Galaxy uses a first-come-first-serve task scheduling algorithm where the tasks are scheduled in the order they are submitted. This scheduling algorithm is part of OpenDC's built-in functionality.

Galaxy's task placement policy is a statically mapped algorithm, where each kind of operation is statically assigned to a particular amount of resources. Using this, we can assign the tasks in OpenDC similar resources as in the original execution on the Galaxy Europe server.

We perform the same comparisons with the other traces. Note that, for the other traces, the original resources assigned to the tasks are unknown, therefore we used the standard topology in combination with a first-fit resource selection policy. To test whether the differences between the distributions are significant, we apply a k-sample Anderson-Darling test to all pairs of distributions. This test is discussed by Scholz and Stephens[23], and is an alteration of the original Anderson-Darling test. The test is meant to measure the agreement between an arbitrary (in this case two) amount of distributions. For each trace we use the following H_0*: The workflow runtime distribution from the trace and the workflow runtime distribution from the simulation are similar enough that they could have originated from the same population.*
Table 5 shows an overview of the simulated workflow runtimes of each trace.

Table 5: A summary of the simulated workflow runtimes. Percentages show the change relative to the original trace values (see Table 3).

Workload	Runtime mean	Runtime min	Runtime max
Galaxy	12,907,056 (+140.72%)	1 (-99.98%)	1,220,563,168 (+241,67%)
Askalon EE	742,199 (-83.15%)	1 (-)	27,581,501 (-94.81%)
Shell	32,257 (+23.02%)	16,351 (+22.60%)	52,701 (+23.55%)
Pegasus P7	73,413,901 (-6.32%)	184,101 (+1,741.01%)	179,073,151 (-73.38%)

Figure 12 shows the distribution of the simulated workflow runtimes by OpenDC together with the runtimes of the original Galaxy trace. We can see that the overall distribution of runtimes looks to have been stretched out in both directions by the simulation. When comparing the workflow runtime metrics from the traces, shown in Tables 3, to the simulated workflow runtimes, Table 5, we see that the maximum simulated runtime increased to 1.2e−9 ms, from a maximum trace runtime of around 0.4e−9 ms. This is an increase of around 140%. However, the opposite happened with the minimum runtime, where the simulated runtime is one millisecond compared to a 4000 ms minimum trace runtime, a fraction of its original trace value.

This means that a portion of workflows were simulated faster than their original execution, and a portion slower. This is potentially caused by the implementation of the resources in OpenDC, where the high resource-demanding workflows might have received too little resources in OpenDC, and the low-demanding workflows too many.

The Anderson-Darling test gives a test statistic of roughly 46.23, which is greater than all critical values. Therefore we reject the H_0 and can say that the two distributions could not have come from the same population. This means that the distributions are significantly different. More specifically, OpenDC's simulation of the Galaxy trace did not properly reflect the original execution of the Galaxy workload. This is despite the fact that we were able to partly model Galaxy's execution environment in OpenDC.

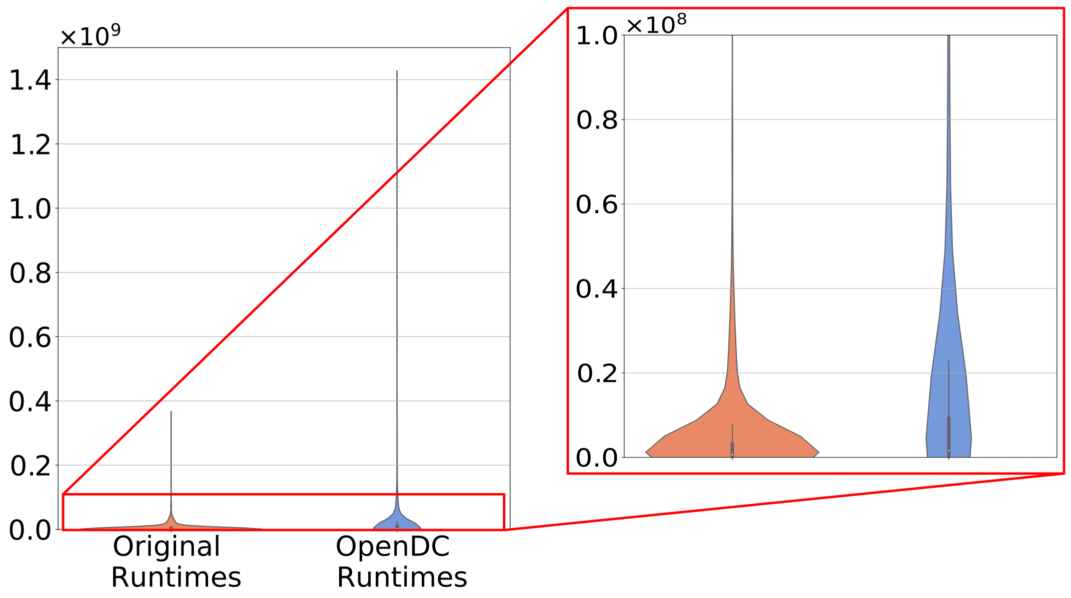

Figure 12: Comparison of workflow Runtime distributions of the original Galaxy trace and its simulation on OpenDC.

Where the workflow runtime distribution of the Galaxy trace looks to have been stretched out, figure 13 shows the opposite seems to have happened with the Askalon trace. Compared to the trace, the simulation seems to have shortened the overall workflow runtimes considerably. This overall decrease in runtimes is also reflected by Table 5, where we see that the mean and maximum value have decreased noticeably relative to the original trace values.

This shows that OpenDC simulated the runtimes overall to be shorter than the runtimes in the original trace. This could imply that the standard topology offers enough resources to run more tasks in parallel, thus decreasing overall workflow runtimes.

The differences between the distributions is significant, because the Anderson-Darling test gives a test statistic of around 252.52, which is greater than all critical values. We reject the H_0 that the distributions could have come from the same population. This in turn indicates that OpenDC's simulation of the Askalon trace did not reflect the original execution of the workload.

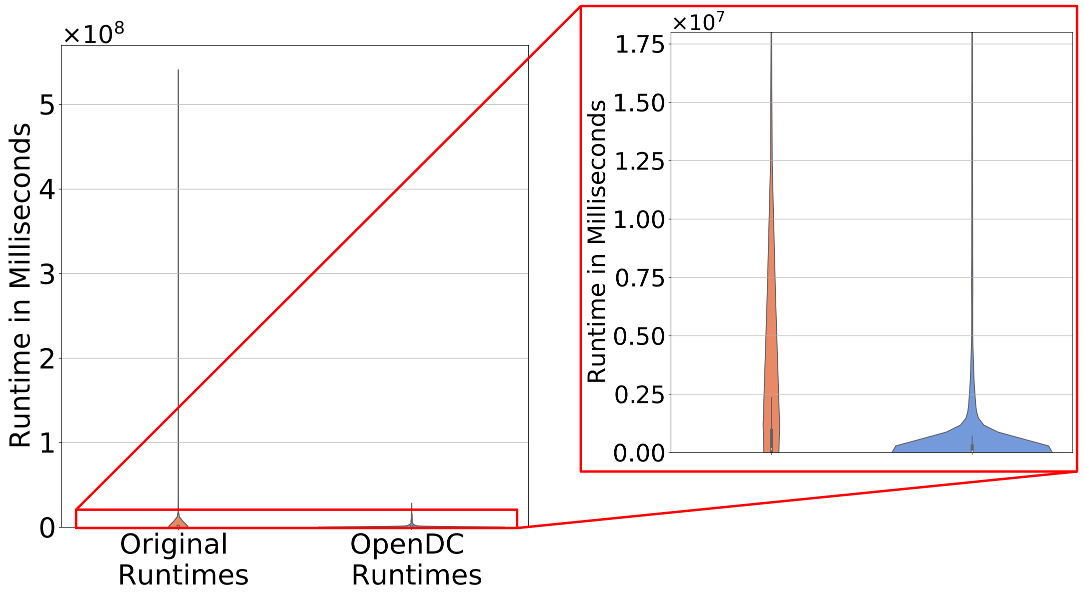

Figure 13: Comparison of workflow Runtime distributions of the original Askalon trace and its simulation on OpenDC.

Figure 14 shows the results of the experiment executed with the Shell trace. We see that the simulation came close to the actual trace in terms of workflow runtime distribution. Two notable differences are the fact that the overall simulation distribution shifted upward around 2.500 milliseconds, and it seems be be stretched out a small amount, which is both confirmed by Table 5. This means that OpenDC simulated the runtimes to be a small amount longer than in the original trace. Overall, both distributions remain very similar, which is potentially caused by the homogeneity of the original trace.

However, even though the distributions look similar visually, the Anderson-Darling test tells us that they differ significantly. The resulting test statistic is roughly 582.17, which is substantially larger than the critical values. Thus we reject H_0; the distributions could not originate from the same population. Again, this indicates that OpenDC's simulation of the Shell trace does not reflect the original execution of Shell workload, despite the fact that the workflow runtime distributions appeared to be similar.

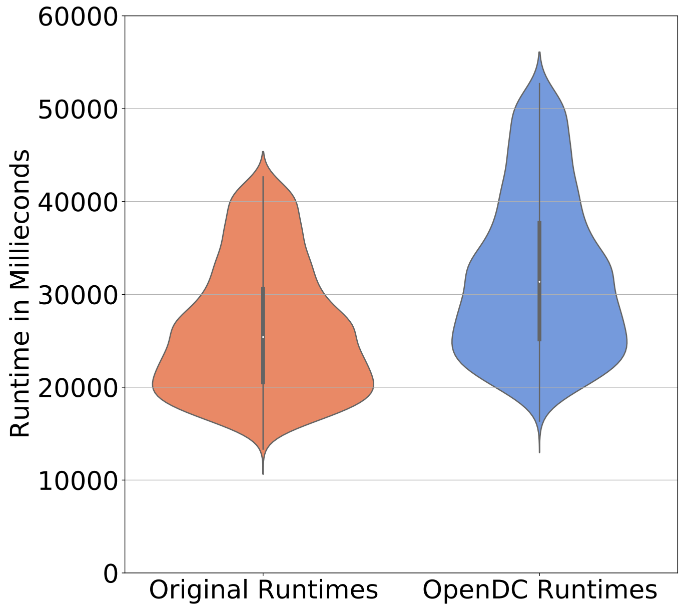

Figure 14: Comparison of workflow Runtime distributions of the original Shell trace and its simulation on OpenDC.

For the Pegasus trace, the simulation has squeezed the distribution considerably, depicted in figure 15. The large difference in distribution is potentially caused by the small amount of workflows in this trace, 38 namely. Therefore, a few shifting data points may have a large impact on the overall distribution.

Again, we observe a significant difference between the two distributions. The Anderson-Darling test gives a test statistic of around 15.39, which causes us to reject the H_0. Overall, OpenDC's simulation of the Pegasus trace did not reflect the original execution of the workload.

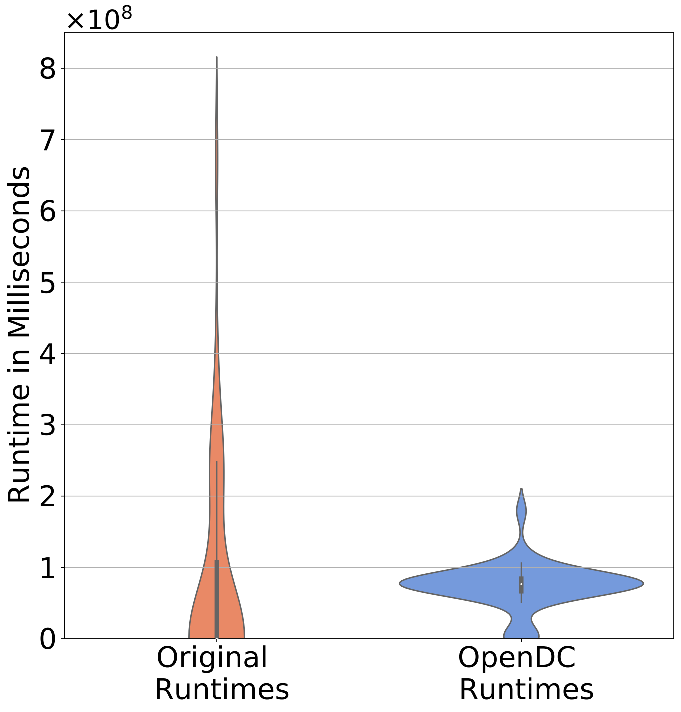

Figure 15: Comparison of workflow Runtime distributions of the original Pegasus trace and its simulation on OpenDC.

When comparing the results of the simulations with the standard topology, we do not see a recurring pattern when it comes to OpenDC's simulation of the workflow runtimes. Both with the Askalon and Pegasus trace we saw the runtime distribution shrink, where Shell's simulated runtime distribution remained nearly identical. We only observed a small increase in its minimum runtime, compared to the original trace, which we also saw occur with the Pegasus trace. This lack of consistency between the simulations indicates that the differences in runtime distribution were largely dictated by the individual properties of each trace. This could indicate that OpenDC's simulations are not biased in a fundamental way.
Additionally, we saw the H_0 being rejected for every trace, which means that the simulated workflow runtime distributions are significantly different from their original trace counterparts. This implies that OpenDC does not model the original execution environment of the workloads as desired.

5 Discussion

Although simple tests, both the Event Validity and Internal Validity tests are important to get an idea of the inner workings of OpenDC. The results of these tests represented in §4.2.1 and §4.2.2, both increase confidence in OpenDC's simulation model. We do not encounter any irregularities in terms of tasks started, tasks finished and task dependencies. The same holds for the Internal Validity, where we see no differences in terms of task execution order or workflow runtimes. This means that OpenDC (1.0) has no randomness built in which could alter the simulation results.
The Parameter Variability results are positive as well. We see that the results of the experiments concerning the task runtimes and resources requested matched the expectations we had set beforehand. However, for the available CPU's in OpenDC, we expected OpenDC to simulate tasks in parallel where possible. This did not seem to be the case.
These three tests are aimed to answer our first sub-question: How realistic is OpenDC's internal behavior? Overall we can say that OpenDC's internal behavior is realistic and that it behaves as one would expect it to.

The results presented in §4.3 indicate that OpenDC can perform simulations with moderate accuracy. For the Galaxy trace we attempt to mimic the environment in which the original tasks were executed as much as possible. We see that the simulated workflow runtime distribution differs from the original workflow runtime distribution. Specifically, the distribution is stretched out. The opposite happened with the Askalon and Pegasus trace, whose distributions were both squeezed. The only situation where this is not the case is the simulation of the Shell trace, which shows great similarity to the original trace. This is possibly explained by the homogeneity of the data, which is almost normally distributed. However, all simulated workflow runtime distributions showed significant differences from their respective trace runtime distributions. We concluded this from the Anderson-Darling test which made us reject the H_0 that the trace and simulated distributions could have come from the same population. This means that OpenDC was not able to accurately model the original execution environments of the workloads.
Furthermore, we found no real consistencies or pattern in their degree of deviation from the original trace. This could indicate that OpenDC does not have (much) bias built in. This could imply that the differences in deviations are not necessarily a product of irregularities of OpenDC's simulation model, but rather of the varying features of the traces to which the simulation model is applied. This again increases confidence in OpenDC's simulation model.
The trace-based tests are aimed to answer our second sub-question: How realistic are OpenDC's simulation metrics when compared to metrics from operations in a physical datacenter? Overall we have to conclude that OpenDC's capabilities to accurately simulate datacenters are unsatisfactory.

We recognise that these results do not represent the full functionality of OpenDC as we based our experiments primarily on workflow runtimes. The scope could be expanded by doing more research into OpenDC's resource simulation, however this requires detailed resource utilization metrics of the traces, which we do not posses.
We are also aware that OpenDC's poor simulation results are potentially caused by incomplete data on the original execution environments of the workloads. For the Galaxy trace we were, for example, unaware of the specific hardware used in the Europe server. Moreover, for the other workloads we did not possess any information regarding the original execution environments. However, with the information we did posses, especially Galaxy's task placement, we feel confident that we are able to apply such a level of realism to get a good idea of OpenDC's capabilities, and whether it can accurately model datacenters.

6 Related Work

There have been other studies done on validating a specific datacenter simulator. One of which is [16], a PhD dissertation on validating the performance of three different cloud computing simulators: Cloudsim[17], Greencloud[18] and Mininet[19]. Alshammari later published an article, focusing[20] on Cloudsim. They recorded metrics of workloads executed on a Raspberry Pi, modeled the Raspberry Pi in the simulators and ran the workloads using the metrics obtained.
These projects validated whether the simulators in question could accurately simulate microdatacenters[21], where this thesis aims to validate OpenDC using workloads obtained from large-scale computing environments.

Workflowsim is a datacenter simulator built on Cloudsim[22] with an emphasis on workflow management. The developers of Workflowsim validated their simulator by executing a workflow in a compute cluster and extracted traces, trained Workflowsim with these traces, after which they ran a simulation, comparing its results to the original traces. However, they did not test the simulator for General Validity as Workflowsim relies on the underlying Cloudsim to provide that functionality.

Another study[3] details an approach to validate simulation models in general. They describe four different approaches to validating simulation models and provide different techniques to develop an idea of a simulator's capabilities. Throughout this thesis we use some of their suggested validation techniques.

7 Conclusion

This research aimed to validate OpenDC as a datacenter simulator. The main research question we attempted to answer was: Can OpenDC realistically simulate a datacenter? Through two sub-questions focussing on General Validation and Trace-based validation we can deduce that OpenDC can adequately model the inner workings of a datacenter where it behaves as one would expect it to. However, we found that OpenDC's simulation results lacked in terms of accuracy.

This leads us to the conclusion that OpenDC is very suitable to explore the effects different parameters have on operations in a datacenter. However, OpenDC may not (yet) be suited to simulate datacenters in an absolute sense.

7.1 Directions for future research

There are certain avenues related to this research that have not been explored:

- **Different ways to validate OpenDC**: In this thesis we chose to validate OpenDC in both a general sense, whether it behaves as one would expect it to, and a comparative sense, where we compared OpenDC's simulation results to real-world computing clusters.
  Future work might want to validate OpenDC by running simulations in an environment even closer to the original as possible. Taking inspiration from [20], tasks can be executed on systems with known hardware models. Using metrics of the tasks executed on the real systems, a custom trace can be constructed, which can then be used to run a simulation in OpenDC. With the specific hardware models known, these systems can be more accurately modeled in OpenDC, which makes a comparison between the original execution and the simulation bare more weight.
  Another way to potentially validate OpenDC is by comparing it to other datacenter simulators. These could be the most used simulators whose operations have been validated in the past. These results could give more indication of OpenDC's capabilities itself, as it is compared to the best in the field. OpenDC could also be compared to similar simulators in terms of overall project size. This would give more insights in how OpenDC stacks up to these other simulators.

- **Validating OpenDC 2.0**: This thesis is solely focused on validating OpenDC's core mechanics, as it is based on OpenDC 1.0. However, with the release of OpenDC 2.0, multiple features have been added, energy consumption being one of them. The methods mentioned earlier could be used to validate OpenDC's new features.

References

[1] *Why do we need data centers?*(n.d.) Retrieved October 18 2020. https://www.dutchdatacenters.nl/en/data-centers/why-do-we-need-data-centers/

[2] Lu, H., Zhang, Z., Yang, L. (2018). *A review on airflow distribution and management in data center. Energy and Buildings*, 179, 264-277.

[3] Sargent, R. G. (2010, December). *Verification and validation of simulation models.* In Proceedings of the 2010 winter simulation conference (pp. 166-183). IEEE.

[4] Versluis, L., Mathá, R., Talluri, S., Hegeman, T., Prodan, R., Deelman, E., Iosup, A. (2020). *The Workflow Trace Archive: Open-Access Data From Public and Private Computing Infrastructures.* IEEE Transactions on Parallel and Distributed Systems, 31(9), 2170-2184.

[5] Iosup, A., Andreadis, G., Van Beek, V., Bijman, M., Van Eyk, E., Neacsu, M., ... Visser, M. (2017, July). *The OpenDC vision: Towards collaborative datacenter simulation and exploration for everybody.* In 2017 16th International Symposium on Parallel and Distributed Computing (ISPDC) (pp. 85-94). IEEE.

[6] Coffman, E. G., Graham, R. L. (1972). *Optimal scheduling for two-processor systems.* Acta informatica, 1(3), 200-213.

[7] Carvalho, L. A., Belhajjame, K., Medeiros, C. B. (2016, October). *Converting scripts into reproducible workflow research objects.* In 2016 IEEE 12th International Conference on e-Science (e-Science) (pp. 71-80). IEEE.

[8] Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg (2018, July). *The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update.* In Nucleic Acids Research, Volume 46, Issue W1, Pages W537–W544, doi:10.1093/nar/gky379

[9] *Workflow Trace Archive.* (2019). WTA-Format. https://wta.atlarge-research.com/traceformat.html

[10] *Atlarge Research.* (2019). https://atlarge-research.com/

[11] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, *Pegasus: a Workflow Management System for Science Automation,* Future Generation Computer Systems, vol. 46, p. 17–35, 2015., (Funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575)

[12] *Actual dedicated hardware for ELIXIR.* (n.d.). Retrieved September 10 2020, from https://wiki.metacentrum.cz/wiki/Elixir

[13] Andreadis, G., Versluis, L., Mastenbroek, F., Iosup, A. (2018, November). *A reference architecture for datacenter scheduling: design, validation, and experiments.* In SC18: International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 478-492). IEEE.

[14] Ilyushkin, A. S. (2019). *Scheduling Workloads of Workflows in Clusters and Clouds (Doctoral dissertation, Delft University of Technology).*

[15] Singla, A., Godfrey, P. B., Kolla, A. (2014). *High throughput data center topology design.* In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) (pp. 29-41).

[16] Alshammari, D. (2018). *Evaluation of cloud computing modelling tools: simulators and predictive models* (Doctoral dissertation, University of Glasgow).

[17] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R. (2011). *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.* Software: Practice and experience, 41(1), 23-50.

[18] Kliazovich, D., Bouvry, P., Khan, S. U. (2012). *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers.* The Journal of Supercomputing, 62(3), 1263-1283.

[19] De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., Prete, L. R. (2014, June). *Using mininet for emulation and prototyping software-defined networks.* In 2014 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-6). IEEE.

[20] Alshammari, D., Singer, J., Storer, T. (2017, June). *Does cloudsim accurately model micro datacenters?.* In 2017 IEEE 10th International Conference on Cloud Computing (CLOUD) (pp. 705-709). IEEE.

[21] Bahl, V. (2015). E*mergence of micro datacenter (cloudlets/edges) for mobile computing.* Microsoft Devices Networking Summit 2015.

[22] Chen, W., Deelman, E. (2012, October). *Workflowsim: A toolkit for simulating scientific workflows in distributed environments.* In 2012 IEEE 8th international conference on E-science (pp. 1-8). IEEE.

[23] Scholz, F. W., Stephens, M. A. (1987). *K-sample Anderson–Darling tests.* Journal of the American Statistical Association, 82(399), 918-924.