Testing and Comparing the Performance of Cloud Service Providers Using a Service Broker Architecture

Divyaa Manimaran Elango[1], Frank Fowley[1], and Claus Pahl[2(✉)]

[1] IC4, Dublin City University, Dublin, Ireland
[2] SwSE, Free University of Bozen-Bolzano, Bolzano, Italy
Claus.Pahl@unibz.it

Abstract. Service brokers are tools that allow different individual service providers to be integrated. An API can be a mechanism to provide a joint interface. Broker can actually also be use for more than integration. We use a cloud service broker that implements a multi-cloud abstraction API in order to carry out performance comparisons between different cloud services. The broker tool here is a multi-cloud storage API that integrates a number of provided storage services. The library supporting the API is organised into three services, which are a file, a blob and a table service. Using this broker architecture, we developed a performance test scenario to compare the different providers, i.e., to compare a range of storage operations by different providers.

1 Introduction

Integration is a key problem in the cloud services context. A cloud service broker is an intermediary application between a client and cloud provider service that can provide this integration [15]. Brokerage reduces the need for service consumers to analyze different types of services by different providers [1]. This enables a single platform to offer the client a common cloud storage service. This results in cost optimization and reduced level of back-end data management requirements.

For our performance evaluation, we use here a cloud service broker that implements a multi-cloud abstraction API. This multi-cloud storage broker supports GoogleDrive, DropBox, Microsoft Azure and Amazon Web Services as the provided storage services. The API library offers file, blob and table services. The API can facilitates the distribution of different types of cloud provider services [16]. The abstraction library allows the cloud broker to adapt to a rapidly changing marketplace.

Vendor lock-in is often referred to as a critical point in choosing a provider. In order to avoid lock-in, a broker can help. A multi-cloud abstraction library is a suitable mechanism that it makes it easy for the client to switch between cloud providers with different services that are supported by the broker.

Z. Á. Mann and V. Stolz (Eds.): ESOCC 2017, CCIS 824, pp. 117–129, 2018.
https://doi.org/10.1007/978-3-319-79090-9_8

Switching or migrating between providers can be driven by quality [27,32]. We use a broker implementation to compare the supported services [8,13,23] from a performance perspective [42]. Service brokers normally remedy interoperability problems [2–5,10]. However, based on this architecture, we look into other service qualities, namely performance which is of key importance for all cloud layers [28–30]. A performance test application was developed here to compare between the services provided through the broker [19]. The performance test scenario was used to compare a range storage operations across different supported providers.

2 Broker – Principles and Supported Services

In Fig. 1 we have outlined the core components of thee broker architecture. We also discuss the storage services supported by it in this section.

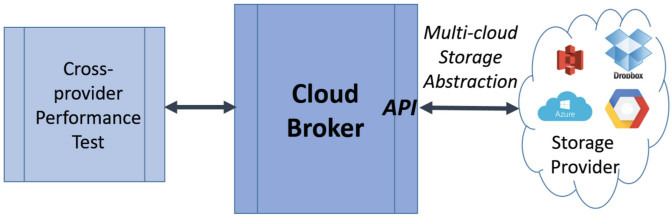

Fig. 1. Architecture of a multi-cloud storage broker.

2.1 Principle Properties of the Storage Broker

Cloud services are generally provided with specifications, but often constructed in a way that makes them hard to be used as part of a common interface, thus impeding on interoperability. We studies different multi-cloud libraries, including Apache Jclouds, DeltaCloud, Kloudless, SecureBlackBox, and SimpleCloud. The purpose was to adopt a successful solution template.

We decided to adopt an approach similar to the Apache Jclouds library for abstraction, as we will explain now. Jclouds [22] provides cloud-agnostic abstraction. A single instance context approach for the mapping of a user request in jclouds was used in our implementation. The purpose of having each class for each provider across different levels of service was adopted from a similar design in the SecureBlackBox library. Our concept of including a manager interface layer at each component level is adopted from LibCloud, another library.

2.2 Services Supported by the Broker

We have included storage services from Google, Dropbox, Azure and Amazon in our cloud storage broker.

- *Amazon Web Service S3* [35] is a file storage service which is built on REST and SOAP. Their SDK is available in all major development languages.
- *Azure Storage* [37] supports blob, file, queue and table services. The API is built on REST, HTTP and xml, and can be integrated with Microsoft visual studio, eclipse and GIT. The Azure SDK provides a separate API package for each service and has the same code flow across different service APIs.
- *DropBox* [36] is a file hosting service. It also enables synchronised backup and web sharing. The DropBox API is very light-weight and easy for a new user.
- *GoogleDrive* [38] offers a cloud file storage service. The GoogleDrive service includes access to a Google API client library.

This selection of service providers resulted in a grouping of the cloud providers and their services as shown in the table below[1] that summarises the main features of the services:

Service	Azure	AWS	Google	DropBox
File	Storage file	-	GoogleDrive	DropBox
Blob	Storage blob	AWS S3	-	-
Table	Storage Table, DocumentDB	DynamoDB, SimpleDB	-	-

3 Broker Architecture

Portability and interoperability are the key objectives of a cloud brokerage tool. Thus the objective of designing and developing an abstraction API is to produce an effective cross-service cloud delivery model [14]. Before describing how we use this to support performance evaluation, we still need to introduce the architectural principles. The main service-oriented functionalities of cloud providers are compute nodes, data volume, load balance, DNS and so on. The advantage of bringing these functionalities to a multi-cloud application provides (1) an easy way of importing and exporting data, (2) choice over price, (3) enhanced SLA, and (4) the elimination of vendor lock-in.

As concept and function integration is the key difficulty in constructing the broker, this broker implementation is based on an ontology that at conceptual level defines the integration. This Storage Abstraction Ontology describes the common naming and meaning approach of the abstraction API [25,26,34]. The model consists of four main layers, namely Service, Provider, Level-2 (composite storage objects) and Level-1 (core storage objects).

- *Level-4 Service*: The Service layer is the top layer and is directly integrated to the user interface layer. This layer basically describes the services that the multi-cloud storage abstraction API supports. There are three services currently supported. They are Blob, Table and File service.
- *Level-3 Provider*: The Provider layer is the second layer, which is one of the context object parameters mapped to the service layer. The multi-cloud storage abstraction supports four main providers, namely Microsoft Azure,

[1] https://www.google.com/drive/;
https://www.dropbox.com/;
https://aws.amazon.com/s3/;
https://azure.microsoft.com/en-us/services/storage/;

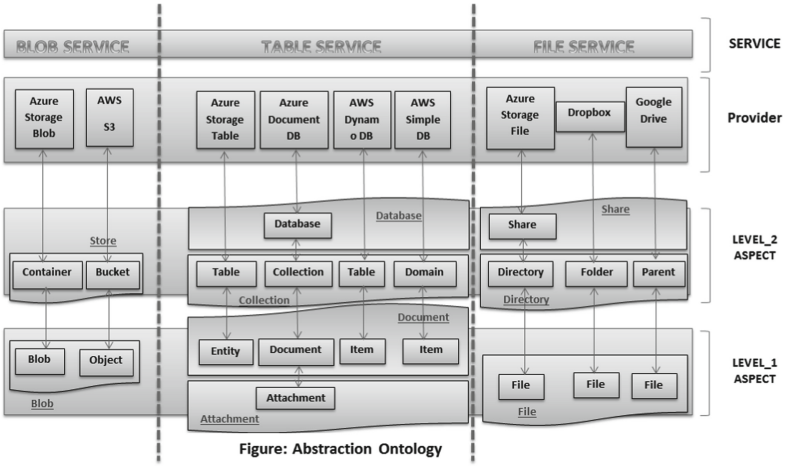

Fig. 2. Ontology-based layered broker architecture.

Amazon Web Services, GoogleDrive and DropBox. The corresponding services supported by the providers are shown below:

Service	Provider
Blob	Azure storage blob; AWS S3
Table	Azure storage table; Azure DocumentDB; AWS DynamoDB; AWS SimpleDB
File	Azure storage file; DropBox and GoogleDrive

– *Level-2 Composite*: Level-2 is the next layer. This layer represents the first level or higher level of composite object abstraction. This layer is service-neutral and brings out the common naming across the providers specific functionalities. Each layer is abstracted based on the common operations and aspect of how the main function is applied in that particular service. Common naming is represented to easily categorise storage resources and group them to make the development of the coding easier.
Based on the Abstraction Ontology Fig. 2, the Blob service has Store which groups Container from Azure Storage Blob and Bucket from AWS S3. The Table service has two different sub-layers - where Database belongs to Azure DocumentDB Database, and where Collection groups Table from Azure Storage Table, collection from Azure DocumentDB Collection, Table from AWS DynamoDB and Domain from AWS SimpleDB. The File service has two different sub-layers where Share belongs to Azure Storage File and Directory groups Directory from Azure Storage File, Folder from DropBox and Parent from the GoogleDrive service.
– *Level-1 Core*: Level-1 represents the lower level of core object abstraction. This layer contains the core functionalities of a particular service across different providers. The classes in this level are extended from an abstract class called AbstractConnector. The class implements the abstract methods defined in the AbstractConnector class. The mapping from Level-2 to Level-1

is performed by an interface class called Manager. This Manager identifies the provider class by its key. Basic CRUD operations on the storage resources are included as core methods. In order to achieve these functions, each operation request should pass through the Level-2 mappings and are then mapped across the service and providers.
The Blob service has Blob which groups Blob from Azure Storage Blob and Object from AWS S3. The Table service has two different sub layers. It has Item which groups Entity from Azure Storage Table, Document from Azure DocumentDB Document, Item from AWS DynamoDB and Item from AWS SimpleDB. Also, the second sub layer Attachment belongs to Azure DocumentDB. The File service has File which groups File from Azure Storage File, File from DropBox and File from GoogleDrive.

4 Performance Testing and Provider Comparison

We have used the broker to compare performance values for the four providers selected. The broker is instrumented to provide the response time results.

In this section, we desribe the performance test set-up and the results for the three service types blob, file and table across the different providers. We organise this section based on the storage types blob, file and table.

Not all providers support each of the storage types. So, the number of compared services provided varies between two and four. We report on the time consumed for a number of standard operations at the two important levels 1 and 2 of the layer architecture. In this way, we cover individual objects (items) and composites (collections) and a range of standard operations on them such as creating or deleting.

4.1 Blob Service Performance Test

The *Blob Service Performance Test* was performed on two providers, namely Azure Storage Blob and AWS S3. This performance test includes two object levels. Level-2 represents Store (which includes container and Bucket). Level-1 represents Blob (which includes Blob and Object).

- The total number of tests performed was 27 to fully cover the respective core and composite objects and the different relevant operations on them. The performance test compares the operations across the service providers.
- Each operation was run 10 times in order to avoid any accidential performance irregularities due to external factors, and the corresponding process time for each request from T1 to T10 was calculated.
- Each request was processed with the same blob size of 10.2 MB, which resembles a standard object size.

The result includes start time, end time, average time and total duration – see Figs. 3 and 4.

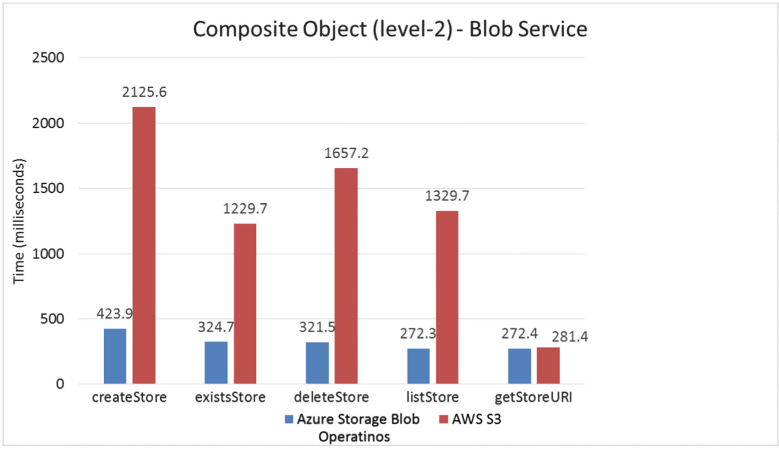

Fig. 3. Blob service Level-2 composite object.

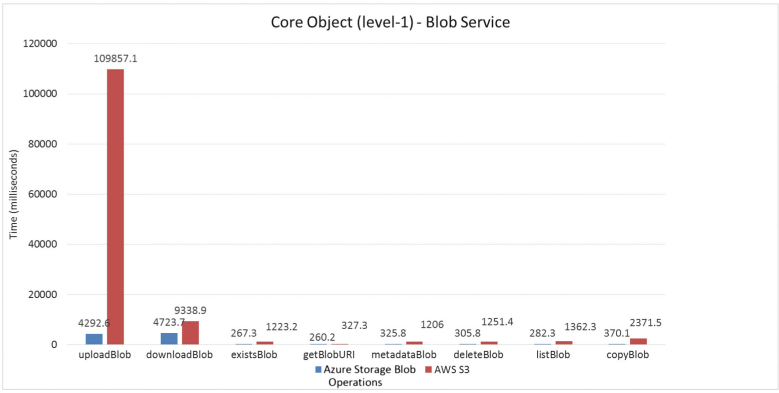

Fig. 4. Blob service Level-1 core object.

4.2 File Service Performance Test

The *File Service Performance Tests* were performed on Azure Storage File, GoogleDrive and DropBox. The tests include two object levels. Level-2 represents Share and Directory. Level-1 represents Files.

- The total number of tests performed was 26 to cover the combinations of different object types and different operations on them.
- The performance tests compare the operations across the service providers. Each operation was run 10 times to eliminate irregular single behaviour, and the corresponding process time for each request from T1 to T10 was calculated.
- Each request was processed with same file size of 10.2 MB as a common size for the object type in question.

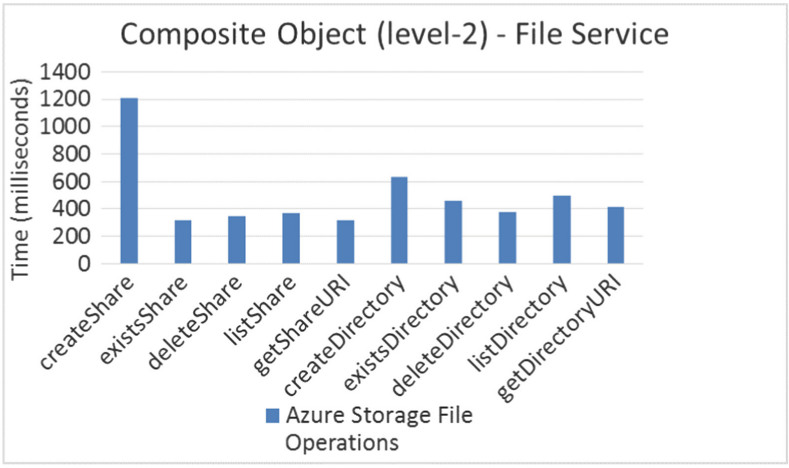

Fig. 5. File service Level-2 composite object.

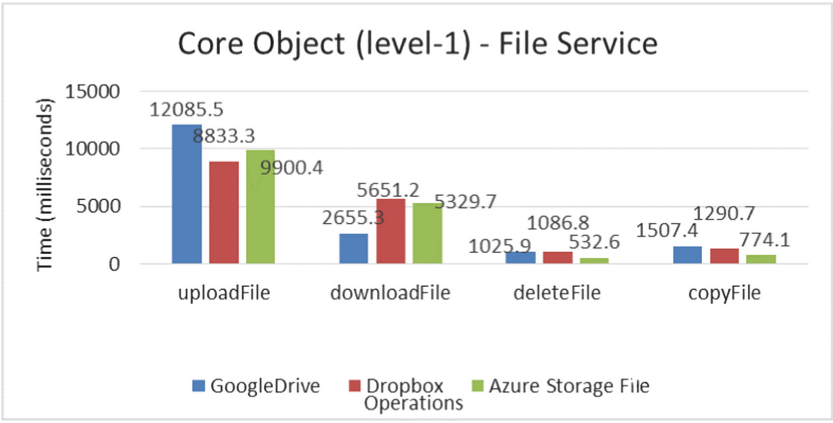

Fig. 6. File service Level-1 core object.

The results include start time, end time, average time and total duration – see Figs. 5 and 6.

4.3 Table Service Performance Test

The *Table Service Performance Tests* were performed on Azure Storage Table, Azure DocumentDB, AWS DynamoDB and AWS SimpleDB. The Tests include two object levels. Level-2 represents Database and Collections (which includes Table, Collections, Table and Domain). Level-1 represents Item (which includes Entity, Document, Table Item, Domain Item) and Attachment.

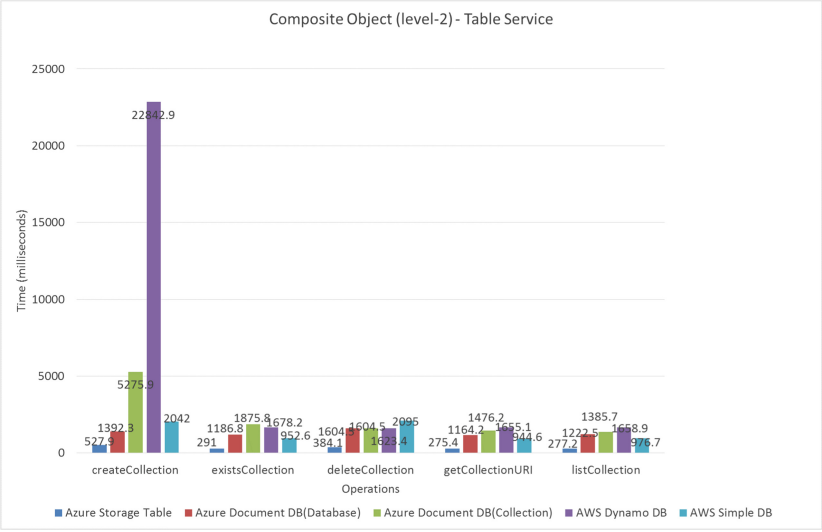

Fig. 7. Table service Level-2 composite object.

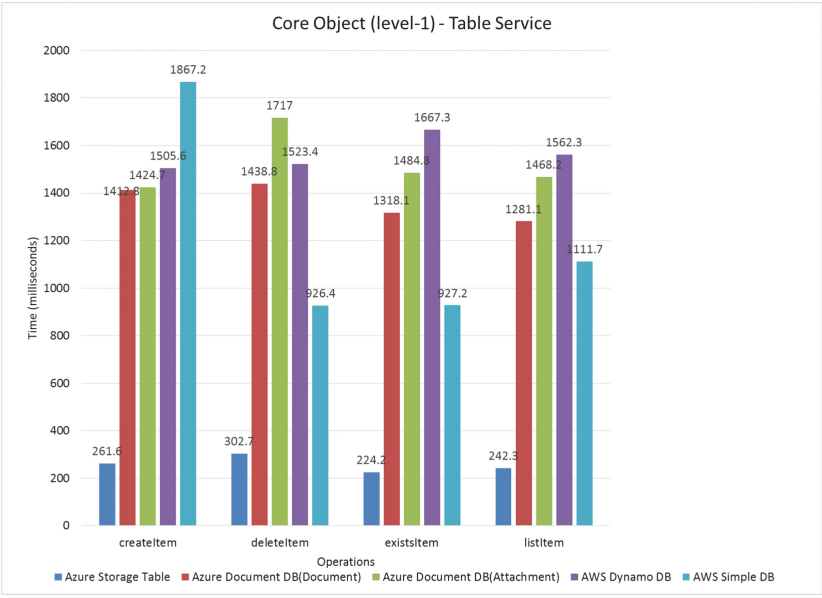

Fig. 8. Table service Level-1 core object.

– The total number of tests was 45. As already explained for the blob tests, this number covers the combination of different objects and the different operations on them. The performance tests compare the operations then across the service providers.

– Each operation was run 10 times, as earlier to avoid irregularities, and the corresponding process time for each request from T1 to T10 was calculated. Each request was processed with a single data record of approximately four columns.

The results include start time, end time, average time and total duration – see Figs. 7 and 8 (and also the performance details in Fig. 9).

SERVICE	PROVIDER	LEVEL OF ASPECTS	Operation Type	Parameter:	Inputs	FileSize	Start Time	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	End Time	Avg Time	Duration	Avg
BlobService	%zure Storage Blob	LEVEL_2	createContainer	ntainerNaiststoring		10.2 MB	1478.612	3004	105	114	117	112	116	238	111	184	138	1478.612	423	00:04.0	423.9
BlobService	%zure Storage Blob	LEVEL_2	existsContainer	ntainerNaiststoring		10.2 MB	1478.613	2500	76	83	73	93	71	73	79	82	117	1478.613	324	00:03.0	324.7
BlobService	%zure Storage Blob	LEVEL_2	deleteContainer	ntainerNaiststoring		10.2 MB	1478.616	2575	78	71	70	69	68	69	71	70	74	1478.616	321	00:03.0	321.5
BlobService	%zure Storage Blob	LEVEL_2	listContainer			10.2 MB	1478.613	2153	73	60	56	65	60	56	59	76	65	1478.613	272	00:02.1	272.3
BlobService	%zure Storage Blob	LEVEL_2	getContainerURI	ntainerNaiststoring		10.2 MB	1478.613	2338	68	38	35	41	35	40	39	47	43	1478.613	272	00:02.1	272.4
BlobService	AWS S3	LEVEL_2	createContainer	ntainerNaoicasso-bu		10.2 MB	1478.699	6387	1587	1711	1370	1426	1618	2172	1647	1530	1808	1478.699	2125	00:21.0	2125.6
BlobService	AWS S3	LEVEL_2	existsContainer	ntainerNaoicasso-bu		10.2 MB	1478.699	5493	841	723	757	910	744	707	727	710	685	1478.699	1229	00:12.0	1229.7
BlobService	AWS S3	LEVEL_2	deleteContainer	ntainerNaoicasso-bu		10.2 MB	1478.706	5394	1186	1071	1173	1845	1152	1188	1350	1167	1046	1478.706	1657	00:16.1	1657.2
BlobService	AWS S3	LEVEL_2	listContainer			10.2 MB	1478.701	4580	941	836	1061	893	737	1280	833	1294	842	1478.701	1329	00:13.0	1329.7
BlobService	AWS S3	LEVEL_2	getContainerURI	ntainerNaoicasso-bu		10.2 MB	1478.7	2649	30	17	15	16	15	17	27	14	14	1478.7	281	00:02.1	281.4
BlobService	%zure Storage Blob	LEVEL_1	Upload	ileP ath,Bld\\Project\		10.2 MB	1478.614	9781	3553	5137	3245	3110	3141	2914	4522	4103	3420	1478.614	4292	00:42.1	4292.6
BlobService	%zure Storage Blob	LEVEL_1	Download	oadFilePatject\\New		10.2 MB	1478.614	7225	5097	4984	4083	4576	4215	4351	4117	4139	4450	1478.614	4723	00:47.0	4723.7
BlobService	%zure Storage Blob	LEVEL_1	Exists	ne,BlobReontainer",		10.2 MB	1478.614	2023	69	66	63	68	76	82	77	76	73	1478.614	267	00:02.1	267.3
BlobService	%zure Storage Blob	LEVEL_1	URI	ne,BlobReontainer",		10.2 MB	1478.614	2277	37	35	33	34	33	41	32	46	34	1478.614	260	00:02.1	260.2
BlobService	%zure Storage Blob	LEVEL_1	Metadata	ne,BlobReontainer",		10.2 MB	1478.615	2879	44	54	45	43	41	39	38	37	38	1478.615	325	00:03.0	325.8
BlobService	%zure Storage Blob	LEVEL_1	Delete	ne,BlobReontainer",		10.2 MB	1478.616	2385	76	81	74	66	67	71	70	73	95	1478.616	305	00:03.0	305.8
BlobService	%zure Storage Blob	LEVEL_1	List	ntainerNaiststoring		10.2 MB	1478.615	2135	66	75	75	75	85	77	91	73	71	1478.615	282	00:02.1	282.3
BlobService	%zure Storage Blob	LEVEL_1	ListBatch	ntainerNaiststoring		10.2 MB	1478.615	2189	91	100	85	77	77	76	75	76	81	1478.615	292	00:02.1	292.7
BlobService	%zure Storage Blob	LEVEL_1	Copy	,DestinationContiaing		10.2 MB	1478.616	2251	136	224	144	179	154	174	147	148	144	1478.616	370	00:03.1	370.1
BlobService	AWS S3	LEVEL_1	Upload	>Path,Blobard\\Proje		10.2 MB	1478.702	11572	11311	9692	108463	106346	114377	109462	112648	111216	108084	1478.703	109857	18:18.1	109837.1
BlobService	AWS S3	LEVEL_1	Download	adFilePatProject\\N		10.2 MB	1478.704	18565	7761	10831	7672	7507	7502	8273	8285	9286	7707	1478.704	9338	01:33.0	9338.9
BlobService	AWS S3	LEVEL_1	Exists	>,BlobRefesso-bucke		10.2 MB	1478.704	4402	814	784	775	1449	798	809	854	787	760	1478.704	1223	00:12.0	1223.2
BlobService	AWS S3	LEVEL_1	URI	>,BlobRefesso-bucke		10.2 MB	1478.704	3061	25	20	21	30	30	21	25	21	19	1478.704	327	00:03.0	327.3
BlobService	AWS S3	LEVEL_1	Metadata	>,BlobRefesso-bucke		10.2 MB	1478.705	4758	844	1043	818	813	787	722	816	755	704	1478.705	1206	00:12.0	1206
BlobService	AWS S3	LEVEL_1	Delete	>,BlobRefesso-bucke		10.2 MB	1478.705	5307	740	697	730	707	841	1279	706	745	762	1478.705	1251	00:12.1	1251.4
BlobService	AWS S3	LEVEL_1	List	ucketNamoiasso-bu		10.2 MB	1478.705	5284	1018	912	899	1013	888	880	897	893	939	1478.705	1362	00:23.1	1362.3
BlobService	AWS S3	LEVEL_1	Copy	,DestinationContiaing		10.2 MB	1478.705	6883	1904	1756	1834	2032	2087	1829	1875	1781	1734	1478.705	2371	00:23.1	2371.5
FileService	Azure Storage File	LEVEL_2	createShare	ihareNam'testshare'		10.2 MB	1478.523	5052	985	916	770	878	741	562	683	768	728	1478.523	1208	00:12.0	1208.3
FileService	Azure Storage File	LEVEL_2	existsShare	ihareNam'testshare'		10.2 MB	1478.529	2670	80	39	64	64	77	63	37	38	55	1478.529	318	00:03.0	318.7
FileService	Azure Storage File	LEVEL_2	deleteShare	ihareNam'testshare'		10.2 MB	1478.539	2744	72	71	69	70	70	94	78	100	106	1478.539	347	00:03.0	347.4
FileService	Azure Storage File	LEVEL_2	listShare			10.2 MB	1478.532	2690	123	112	54	63	84	75	77	53	76	1478.532	370	00:03.1	370.7
FileService	Azure Storage File	LEVEL_2	getShareURI	ihareNam'testshare'		10.2 MB	1478.529	2252	86	113	92	111	93	111	94	114	91	1478.529	315	00:03.0	315.7
FileService	Azure Storage File	LEVEL_2	createDirectory	me,Directre","testdi		10.2 MB	1478.532	4317	310	303	108	178	145	193	189	445	162	1478.532	635	00:06.0	635
FileService	Azure Storage File	LEVEL_2	existsDirectory	me,Directre","testdi		10.2 MB	1478.532	3726	81	109	95	131	76	110	93	112	94	1478.532	462	00:04.1	462.7
FileService	Azure Storage File	LEVEL_2	deleteDirectory	me,Directre","testdi		10.2 MB	1478.539	2838	105	107	105	106	123	103	103	101	108	1478.539	379	00:03.1	379.9
FileService	Azure Storage File	LEVEL_2	listDirectory			10.2 MB	1478.534	2779	369	410	190	209	202	209	201	201	217	1478.534	498	00:04.1	498.7
FileService	Azure Storage File	LEVEL_2	getDirectoryURI	me,Directre","testdi		10.2 MB	1478.532	2481	205	204	205	205	205	362	88	87	109	1478.532	415	00:04.0	415.1
FileService	Azure Storage File	LEVEL_1	Upload	ame,FileP ismard\\Pi		10.2 MB	1478.534	24474	4738	11320	14444	7463	12013	9090	6784	4711	3967	1478.534	9900	01:39.0	9900.4
FileService	Azure Storage File	LEVEL_1	Download	ame,FilePd\\Project\		10.2 MB	1478.535	8569	4791	4844	4757	4990	4801	4992	5355	5557	4641	1478.535	5329	00:53.0	5329.7
FileService	Azure Storage File	LEVEL_1	Exists	oryName,F "testdirec		10.2 MB	1478.536	2779	296	315	299	311	305	308	304	279	334	1478.536	553	00:05.1	553
FileService	Azure Storage File	LEVEL_1	Metadata	oryName,F "testdirec		10.2 MB	1478.536	2728	157	131	110	157	134	146	103	110	131	1478.536	390	00:03.1	390.7
FileService	Azure Storage File	LEVEL_1	Delete	oryName,F "testdirec		10.2 MB	1478.537	2871	408	476	447	256	179	187	197	159	146	1478.537	532	00:05.0	532.6
FileService	Azure Storage File	LEVEL_1	Copy	ShareNam:t","testsh		10.2 MB	1478.537	4283	352	307	294	295	384	627	603	280	316	1478.537	774	00:07.1	774.1
FileService	DropBox	LEVEL_1	Upload	ileReferei ew folder		10.2 MB	1478.283	10746	9155	7385	8656	9846	7310	9354	8973	8459	8449	1478.283	8833	01:28.0	8833.3
FileService	DropBox	LEVEL_1	Metadata	eferenceh "test.pdf"		10.2 MB	1478.284	3330	501	503	557	510	482	482	471	506	520	1478.284	786	00:07.1	786.2
FileService	DropBox	LEVEL_1	Download	ileReferenzr/Dropbo		10.2 MB	1478.518	6933	4228	4904	4961	4890	5403	6672	5535	4693	8293	1478.518	5651	00:56.1	5651.2
FileService	DropBox	LEVEL_1	Delete	FilePath "test.pdf"		10.2 MB	1478.519	3385	722	666	770	1634	777	756	725	726	707	1478.519	1086	00:10.1	1086.8
FileService	DropBox	LEVEL_1	Copy	Destinatiopdf","Cop		10.2 MB	1478.519	3957	810	864	818	820	1344	1114	767	859	1554	1478.519	1290	00:12.1	1290.7
FileService	GoogleDrive	LEVEL_1	Upload	Path,FileTFolder\\Go		10.2 MB	1478.268	17282	15613	11321	10637	11029	13292	10830	10418	10210	10223	1478.268	12085.5	02:00.0	12085.5
FileService	GoogleDrive	LEVEL_1	List			10.2 MB	1478.28	2788	314	329	294	280	359	317	379	292	275	1478.28	562	00:05.1	562.7
FileService	GoogleDrive	LEVEL_1	Download	Path,FileRw folder\\		10.2 MB	1478.272	5484	2332	2303	2524	2475	2287	2423	2091	2868	1766	1478.272	2655	00:26.1	2655.3
FileService	GoogleDrive	LEVEL_1	Delete	eferenceh test.pdf		10.2 MB	1478.277	3892	673	854	745	661	696	666	692	745	635	1478.277	1025	00:10.0	1025.9
FileService	GoogleDrive	LEVEL_1	Copy	Destinatiopdf","cop		10.2 MB	1478.279	4400	1083	1240	1800	1101	1016	1119	1083	1163	1069	1478.279	1507	00:15.0	1507.4
TableService	%zure Storage Table	LEVEL_2	createTable	`ableNam("testtable"			1478.736	2634	402	306	280	275	268	324	252	275	263	1478.736	527	00:05.0	527.9
TableService	%zure Storage Table	LEVEL_2	existsTable	`ableNam("testtable"			1478.736	2470	49	49	44	44	40	72	49	44	49	1478.736	291	00:02.1	291
TableService	%zure Storage Table	LEVEL_2	deleteTable	`ableNam("testtable"			1478.739	2987	107	101	88	92	90	100	96	88	92	1478.739	384	00:03.1	384.1
TableService	%zure Storage Table	LEVEL_2	getTableURI	`ableNam("testtable"			1478.736	2338	42	43	40	42	47	43	53	53	53	1478.736	275	00:02.1	275.4
TableService	%zure Storage Table	LEVEL_2	listTable				1478.737	2372	48	43	51	45	42	42	42	43	44	1478.737	277	00:02.1	277.2
TableService	%zureDocumentDB	LEVEL_2	createDB	DBName "TestDB"			1478.782	4091	1233	1321	1027	1115	938	917	918	1136	1227	1478.782	1392	00:13.1	1392.3
TableService	%zureDocumentDB	LEVEL_2	existsDB	DBName "TestDB"			1478.782	3181	1195	851	1205	1024	116	1142	1176	1064	914	1478.782	1286	00:12.1	1186.8
TableService	%zureDocumentDB	LEVEL_2	deleteDB	DBName "TestDB"			1478.805	4318	1491	1458	1376	1194	1207	1236	1284	1305	1176	1478.805	1604	00:16.0	1604.5
TableService	%zureDocumentDB	LEVEL_2	getDBURI	DBName "TestDB"			1478.782	3807	886	800	830	744	803	1213	905	856	798	1478.782	1164	00:11.1	1164.2
TableService	%zureDocumentDB	LEVEL_2	listDB				1478.783	3340	947	922	918	1028	1015	1039	1023	1012	981	1478.783	1222	00:12.0	1222.5
TableService	%zureDocumentDB	LEVEL_2	createCollection	e,Collectic","TestCollection"			1478.784	4636	1841	2042	1739	1633	1561	3183	12362	11889	11873	1478.784	5275	00:52.1	5275.9
TableService	%zureDocumentDB	LEVEL_2	existsCollection	e,Collectic","TestCollection"			1478.784	6885	1215	1232	1609	1294	1126	1103	1393	1449	1452	1478.784	1875	00:18.1	1875.8
TableService	%zureDocumentDB	LEVEL_2	deleteCollection	e,Collectic","TestCollection"			1478.805	4318	1491	1458	1376	1194	1207	1236	1284	1305	1176	1478.805	1604	00:16.0	1604.5
TableService	%zureDocumentDB	LEVEL_2	getCollectionURI	e,Collectic","TestCollection"			1478.784	4233	1227	1023	1240	1208	1239	1320	1023	1131	1118	1478.784	1476	00:14.1	1476.2
TableService	%zureDocumentDB	LEVEL_2	listCollection	DBName "TestDB"			1478.784	3522	1129	1254	1109	1112	1302	973	1082	1244	1130	1478.784	1385	00:13.1	1385.7
TableService	AWSDynamoDB	LEVEL_2	createTable	`ableNam("testtable"			1478.815	26592	22075	21986	21475	26317	21638	21683	21883	23070	21710	1478.815	22842	03:48.0	22842.9
TableService	AWSDynamoDB	LEVEL_2	existsTable	`ableNam("testtable"			1478.815	6235	1207	1047	1229	1133	1325	1115	1138	1021	1332	1478.815	1678	00:16.1	1678.2
TableService	AWSDynamoDB	LEVEL_2	deleteTable	`ableNam("testtable"			1478.824	5176	1348	1229	1228	1171	1491	1228	1332	1026	1005	1478.824	1623	00:16.0	1623.4
TableService	AWSDynamoDB	LEVEL_2	getTableURI	`ableNam("testtable"			1478.815	5484	1338	1213	1243	1211	1192	1184	1227	1208	1351	1478.815	1651	00:16.1	1651.1
TableService	AWSDynamoDB	LEVEL_2	listTable				1478.824	8127	966	1046	898	954	912	921	922	922	921	1478.824	1658	00:16.1	1658.9
TableService	AWSSimpleDB	LEVEL_2	createDomain	>mainNarrestDomain"			1478.869	6269	1591	1561	1552	1638	1561	1579	1564	1552	1553	1478.869	2042	00:20.0	2042
TableService	AWSSimpleDB	LEVEL_2	existsDomain	>mainNarrestDomain"			1478.869	4861	701	506	485	609	517	467	460	465	455	1478.869	952	00:09.1	952.6
TableService	AWSSimpleDB	LEVEL_2	deleteDomain	>mainNarrestDomain"			1478.885	5779	1671	1656	1789	1703	1644	1677	1739	1630	1662	1478.885	2095	00:20.1	2095
TableService	AWSSimpleDB	LEVEL_2	getDomainURI	>mainNarrestDomain"			1478.869	5114	510	484	475	553	455	461	474	463	457	1478.869	944	00:09.0	944.6
TableService	AWSSimpleDB	LEVEL_2	listDomain				1478.869	5380	513	475	500	541	473	468	472	459	486	1478.869	976	00:09.1	976.7
TableService	%zure Storage Table	LEVEL_1	createEntity	inKey,Rovn", "smith	1 Record		1478.776	2264	38	40	36	38	39	42	40	39	40	1478.776	261	00:02.1	261.6
TableService	%zure Storage Table	LEVEL_1	deleteEntity	,Partition*, "smith", "jonson"	1 Record		1478.777	2329	79	84	86	74	73	73	79	83	67	1478.777	302	00:03.0	302.7
TableService	%zure Storage Table	LEVEL_1	existsEntity	,Partition*, "smith", "j	1 Record		1478.301	1861	40	36	36	40	40	52	57	41	39	1479.301	224	00:02.0	224.2
TableService	%zure Storage Table	LEVEL_1	listEntity	ame,Partitiatble", "sr	1 Record		1479.301	2054	46	36	40	46	39	36	36	35	55	1479.301	242	00:02.0	242.3
TableService	%zureDocumentDB	LEVEL_1	createDocument	e,Documion", "doc	1 Record		1478.79	3659	1168	1180	1207	1132	1154	1161	1175	1153	1139	1478.79	1412	00:14.0	1412.8
TableService	%zureDocumentDB	LEVEL_1	existsDocument	ctionNamollection",	1 Record		1478.791	3738	1143	1077	1053	1089	1065	1008	998	999	1011	1478.791	1318	00:13.0	1318.1
TableService	%zureDocumentDB	LEVEL_1	listDocument	e,Collectic, "TestColl	1 Record		1478.791	3610	997	1092	1023	1031	1059	988	983	1007	1021	1478.791	1281	00:12.1	1281.1
TableService	%zureDocumentDB	LEVEL_1	deleteDocument	ctionNamCollectio	1 Record		1478.795	3675	1300	1204	1231	1157	1183	1199	1137	1157	1145	1478.795	1438	00:14.0	1438.8
TableService	%zureDocumentDB	LEVEL_1	createAttachmen	e,Docume", "docume	1 Record		1478.791	3762	1179	1256	1154	1136	1234	1130	1118	1121	1157	1478.791	1424	00:14.0	1424.7
TableService	%zureDocumentDB	LEVEL_1	existsAttachmen	ame,Docum", "docu	1 Record		1478.793	3848	1168	1299	1177	1255	1222	1100	1264	1353	1162	1478.793	1484	00:14.1	1484.8
TableService	%zureDocumentDB	LEVEL_1	listAttachment	ctionNamCollection"	1 Record		1478.792	4125	1327	1219	1228	1140	1112	1128	1142	1074	1187	1478.792	1468	00:14.1	1468.2
TableService	%zureDocumentDB	LEVEL_1	deleteAttachmen	ame,Docuon", "docu	1 Record		1478.794	4039	1584	1397	1459	1501	1246	1469	1456	1502	1517	1478.794	1717	00:17.0	1717
TableService	AWSDynamoDB	LEVEL_1	createItem	me,Attribs,ble","jsonstring"			1478.821	5130	1128	1221	1130	1127	1133	1021	1041	1010	1115	1478.821	1505	00:15.0	1505.6

Fig. 9. Detailed performance measurements.

5 Discussion of Results and Conclusions

The aim of cloud service brokerage is customising or integrating existing services or making them interoperable. We have developed what based on common classification schemes in [11,12,39] is categorised as an integration broker. The purpose of a broker is intermediation between consumers and providers to provide advanced capabilities (interoperability and portability [33]) that builds up on an intermediary/broker platform to provide for instance a marketplace to bring providers and customers together and automatically facilitate multi-provider usage or portability across providers. The broker for cloud storage service providers implement a joint interface to allow

- easy portability for the user and
- easy extensibility for the broker provider.

This broker solution enables through the joint API also the opportunity for a cloud storage user to easily migrate between service providers and evolve the systems [9,21], without having sufficient standards [6,7,20].

We investigated here the usage of the broker to carry out comparative performance tests across the providers in order to support the user with the decision which provider to choose, if this is taken based on a performance criterion.

Our observations from the performance tests we described earlier are the following:

- (a) Core Objects: We can observe that the performance of core object storage operations varies significantly across Cloud providers. Azure outperforms AWS S3 by a factor of between 4 and 5 in our test scenario. For individual object operations, Azure is also up to 5 times faster in terms of access speed. For example, the common function of UploadBlob takes approximately 4 seconds on Azure and 10 seconds on AWS S3 for a 10.2 MB file.
- (b) Composite Objects: The tests of composite object operations [31] that relate to collections show that Azure has significantly more access performance than other providers. In particular, AWS DynamoDB has a unusually long access time for its CollectionCreate operation. The tests on individual table entity operations show Azure to be the fastest by a considerable margin with over 5 to 6 times lesser access speeds on average.
- (c) Upload and Download: The average of the combined file upload and download speeds do not vary considerably across the providers tested.

We have defined some parameters, such as object size, in a specific way. Other choices might result in different observations. Our aim here was not to recommend a particular provider. The aim was to demonstrate the usefulness of instrumenting brokers for either decision making or as an ongoing monitoring approach. Any selection can anyway not happen without considering other properties such as security.

In the future, we plan to consider more storage services. Furthermore, the impact of different architectures in terms on IaaS or PaaS with and without the use of container technologies [17,18,24,40,41] shall be explored.

Acknowledgements. This work was partly supported by IC4 (Irish Centre for Cloud Computing and Commerce), funded by EI and the IDA.

References

1. Ried, S.: Cloud Broker - A New Business Model Paradigm. Forrester (2011)
2. Elango, D.M., Fowley, F., Pahl, C.: An ontology-based architecture for an adaptable cloud storage broker. In: Advances in Service-Oriented and Cloud Computing. Springer CCIS (2018, to appear)
3. Benslimane, D., Dustdar, S., Sheth, A.: Services mashups - the new generation of web applications. Internet Comput. **12**(5), 13–15 (2008)
4. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., Morrow, M.: Blueprint for the inter-cloud: protocols and formats for cloud computing interoperability. In: International Conference on Internet and Web Applications and Services (2009)
5. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu, C.-H., Yang, L.T., Park, J.H., Yeo, S.-S. (eds.) ICA3PP 2010. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13119-6_2
6. Cloud Standards (2017). http://cloud-standards.org/
7. ETSI Cloud Standards (2017). http://www.etsi.org/newsevents/news/734-2013-12-press-release-report-on-cloudcomputing-standards
8. Fehling, C., Mietzner, R.: Composite as a service: cloud application structures, provisioning, and management. Inf. Technol. **53**(4), 188–194 (2011)
9. Pahl, C., Jamshidi, P., Weyns, D.: Cloud architecture continuity: change models and change rules for sustainable cloud software architectures. J. Softw. Evol. Process 29(2) (2017)
10. Pahl, C., Jamshidi, P., Zimmermann, O.: Architectural principles for cloud software. In: ACM Transactions on Internet Technology. (2018, to appear)
11. Fowley, F., Pahl, C., Zhang, L.: A comparison framework and review of service brokerage solutions for cloud architectures. In: 1st International Workshop on Cloud Service Brokerage (2013)
12. Fowley, F., Pahl, C., Jamshidi, P., Fang, D., Liu, X.: A classification and comparison framework for cloud service brokerage architectures. IEEE Trans. Cloud Comput. (2017). https://doi.org/10.1109/TCC.2016.2537333. http://ieeexplore.ieee.org/document/7423741/
13. Garcia-Gomez, S., et al.: Challenges for the comprehensive management of cloud services in a PaaS framework. Scalable Comput. Pract. Experience **13**(3), 201–213 (2012)
14. Elango, D.M., Fowley, F., Pahl, C.: Pattern-driven architecting of an adaptable ontology-driven cloud storage broker. In: University of Oslo, Department of Informatics, Research report 471, pp. 33–47 (2017)
15. Gartner: Cloud Services Brokerage. Gartner Research (2013). http://www.gartner.com/it-glossary/cloud-servicesbrokerage-csb
16. Grozev, N., Buyya, R.: InterCloud architectures and application brokering: taxonomy and survey. Softw. Pract. Experience **44**(3), 369–390 (2012)
17. Pahl, C., Jamshidi, P.: Microservices: a systematic mapping study. In: Proceedings CLOSER Conference, pp. 137–146 (2016)
18. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations and issues for migrating to microservices architectures: an empirical investigation. IEEE Cloud Comput. (2018). Accepted for publication

19. Hofer, C.N., Karagiannis, G.: Cloud computing services: taxonomy and comparison. J. Internet Serv. Appl. **2**(2), 81–94 (2011)
20. IEEE Cloud Standards (2015). http://cloudcomputing.ieee.org/standards
21. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)
22. jclouds: jclouds Java and Clojure Cloud API (2015). http://www.jclouds.org/
23. Ferrer, A.J., et al.: OPTIMIS: a holistic approach to cloud service provisioning. Future Gener. Comput. Syst. **28**(1), 66–77 (2012)
24. Gacitua-Decar, V., Pahl, C.: Structural process pattern matching based on graph morphism detection. Int. J. Softw. Eng. Knowl. Eng. **27**(2), 153–189 (2017)
25. Pahl, C.: Layered ontological modelling for web service-oriented model-driven architecture. In: Hartman, A., Kreische, D. (eds.) ECMDA-FA 2005. LNCS, vol. 3748, pp. 88–102. Springer, Heidelberg (2005). https://doi.org/10.1007/11581741_8
26. Pahl, C., Giesecke, S., Hasselbring, W.: Ontology-based modelling of architectural styles. Inf. Softw. Technol. **51**(12), 1739–1749 (2009)
27. Pahl, C., Xiong, H.: Migration to PaaS clouds - migration process and architectural concerns. In: IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems MESOCA (2013)
28. Konstantinou, A.V., Eilam, T., Kalantar, M., Totok, A.A., Arnold, W., Sniblel, E.: An architecture for virtual solution composition and deployment in infrastructure clouds. In: International Workshop on Virtualization Technologies in Distributed Computing (2009)
29. Jamshidi, P., Sharifloo, A., Pahl, C., Arabnejad, H., Metzger, A., Estrada, G.: Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. In: 12th International ACM SIGSOFT Conference on Quality of Software Architectures QoSA (2016)
30. Arabnejad, H., Jamshidi, P., Estrada, G., El Ioini, N., Pahl, C.: An auto-scaling cloud controller using fuzzy Q-learning - implementation in openstack. In: Aiello, M., Johnsen, E.B., Dustdar, S., Georgievski, I. (eds.) ESOCC 2016. LNCS, vol. 9846, pp. 152–167. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44482-6_10
31. Mietzner, R., Leymann, F., Papazoglou, M.: Defining composite configurable SaaS application packages using SCA. In: International Conference on Internet and Web Applications and Services, Variability Descriptors and Multi-tenancy Patterns (2008)
32. Pahl, C., Xiong, H., Walshe, R.: A comparison of on-premise to cloud migration approaches. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 212–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40651-5_18
33. Petcu, D., et al.: Portable cloud applications - from theory to practice. Future Gen. Comput. Syst. **29**(6), 1417–1430 (2013)
34. Javed, M., Abgaz, Y.M., Pahl, C.: Ontology change management and identification of change patterns. J. Data Semant. **2**(2–3), 119–143 (2013)
35. Amazon Simple Storage Service (S3) Cloud Storage AWS. https://aws.amazon.com/s3/
36. Dropbox. https://www.dropbox.com/
37. Azure Storage - Secure cloud storage. https://azure.microsoft.com/en-us/services/storage/
38. Google Drive - Cloud Storage & File Backup. https://www.google.com/drive/
39. Jamshidi, P., Pahl, C., Mendonca, N.C.: Pattern-based multi-cloud architecture migration. Softw. Pract. Experience **47**(9), 1159–1184 (2017)

40. Pahl, C., Brogi, A., Soldani, J., Jamshidi, P.: Cloud container technologies: a state-of-the-art review. IEEE Trans. Cloud Comput. (2017). https://doi.org/10.1109/TCC.2017.2702586. http://ieeexplore.ieee.org/document/7922500/
41. Aderaldo, C.M., Mendonca, N.C., Pahl, C., Jamshidi, P.: Benchmark requirements for microservices architecture research. In: 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering. IEEE (2017)
42. Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L.E., Pahl, C., Schulte, S., Wettinger, J.: Performance engineering for microservices: research challenges and directions. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (2017)