Universität Stuttgart

Fakultät Informatik, Elektrotechnik und Informationstechnik

The Nefolog & MiDSuS Systems for Cloud Migration Support

Mingzhu Xiu and Vasilios Andrikopoulos

Report 2013/08

Version 1.0, November 2013

Institut für Architektur von Anwendungssystemen

Universitätsstraße 38
70569 Stuttgart
Germany

CR: D.2.1, D.2.9, H.3.3, H.5.2

1 Introduction

With the development of Cloud computing an increasing number of enterprises and organizations consider to migrate their applications to the Cloud in order to scale with demand and reduce their capital expenses. A decision support system is in many cases required to help users to find a cost-efficient offering among the Cloud services which own similar features [ASL13b].

A decision support system called MDSS [ASL13a] was implemented by Zhe Song in early 2013 as part of his Diploma Thesis [Zhe13]. MDSS can help users to find a matching offering between user demands and collected data in a knowledge base which contains information from providers Google and Windows Azure. Furthermore, it can also calculate the costs of each candidate offering and the details per month in a diagram with spline. In addition, the results can be enumerated with their provider information as well as locations of data center. Finally, the results can be ordered by different Ranking parameters. By referring to the Ranking results, users can find a best solution to migrate their applications to the Cloud.

Based on the accumulated knowledge while developing MDSS, a new decision support system is proposed in [Min13]. It consists of a set of *decision support services* which operate with an expanded w.r.t. MDSS knowledge base and a series of web application interfaces. The new knowledge base contains more providers than the previous one in MDSS. As a result, the classification of service types is refined in order to cover all the Cloud offerings. Furthermore, the available information on the location of the data centers is also increased. The functions which were offered by MDSS as a Web application are re-designed as decision support services to provide candidates search and costs calculation functionalities. The services are exposed as Web APIs (in a RESTful manner) and can be used by a decision support system which focuses on selecting a suitable cost of the whole migration project among different migration types.

The rest of this report is aimed to explain the new decision support system which is designed and implemented in [Min13]. It is structured as following: Section 2 summarizes the *Nefolog* system which contains the collection of decision support services and the knowledge base. Section 3 summarizes the implementation of MiDSuS which is based on Nefolog and realizes decision support for different migration types. Section 4 concludes some existing issues in each system and collects some suggestions for the future work.

2 Nefolog

2.1 Architecture of Nefolog

Figure 1 provides an overview of the architecture of Nefolog. All the services are defined by different URIs which can also present the requirements of users and are used to handle the interactions between users and the Cloud provider knowledge base. In addition,

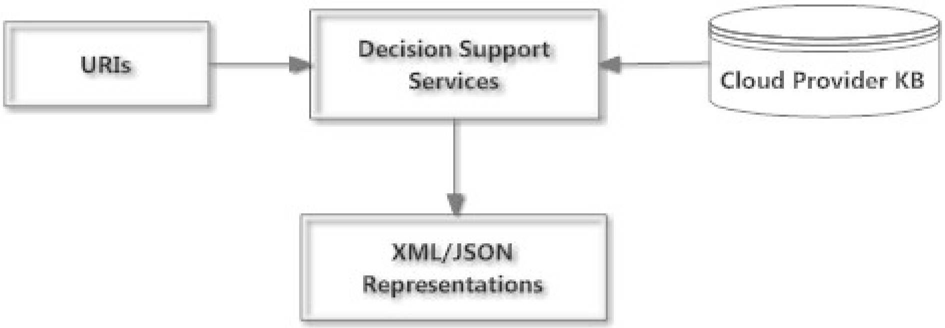


Figure 1: Architecture of Nefolog

each service has two representations, XML and JSON.

The knowledge base is a relational database and built by PostgreSQL[1] RDBMS which is free and released under the PostgreSQL license. Data are organized by tables which are linked by foreign keys. Moreover, data queries are implemented by SQL language. Web services are implemented in the Restlet framework and operated in conjunction with the knowledge base. There are two main decision support services offered: candidate search and cost calculator [Min13]. The candidate search service is accomplished by the comparisons between user demands which are presented by the query part of URIs and data in knowledge base. The cost calculator service is used to calculate the costs of candidate offerings with the help of cost formulas.

2.2 Knowledge Base

Figure 2 shows the data model of the knowledge base (KB) which contains all the information from 6 providers: Google, Amazon Web Services (AWS), Windows Azure, Hp Cloud, Rackspace and Flexiscale[2]. Each provider provides more than one offering and all these offerings can be summarized in different service types. Each offering contains one or more configurations while each configuration has different performance characteristics and a total cost. Furthermore, the total cost consists of upfront cost, data transfer cost and service cost. Each cost is calculated by a cost formula. Different geographical areas and usage amounts of necessary variables can lead to different cost formulas. There is a special provider in the KB, Flexiscale, which calculates the cost in different way. This cost is defined by a pricing list and can be identified according to the units amount which is calculated by a formula. The formulas for each configuration are different and are defined by extraction of the relevant information from the providers' Web sites, as discussed in [ASL13a].

In the current KB, there are in total 58 offerings which are from 6 providers and

[1] `http://www.postgresql.org/`

[2] `http://www.flexiscale.com/products/flexiscale/`

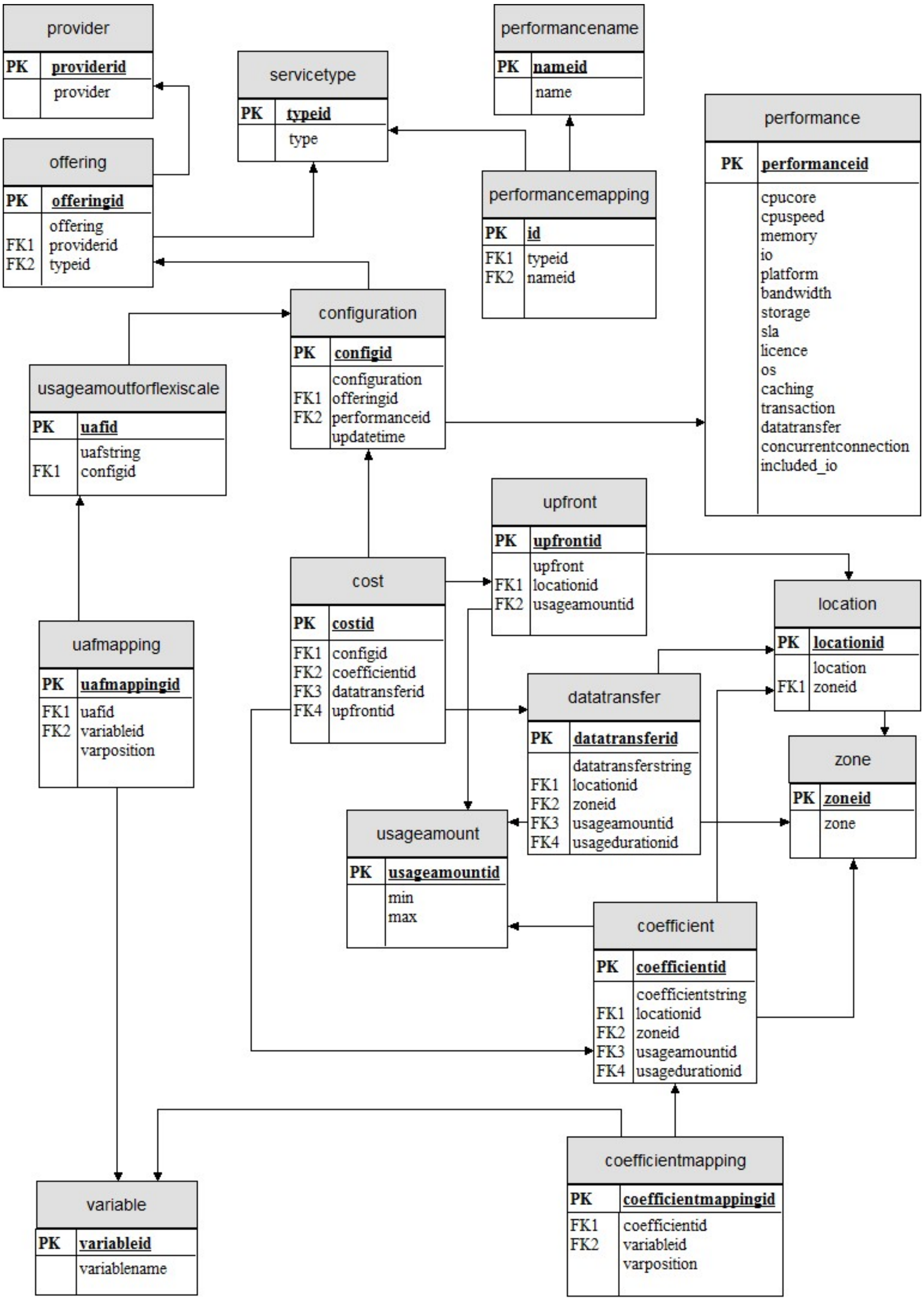


Figure 2: Data Model of Knowledge Base [Min13]

contain 520 configurations. 472 of them contain the information of performance characteristics. 58 offerings are classified into 12 service types. The total cost table consists of 21856 combinations among 3686 upfront costs, 185 data transfer cost formulas and 4733 service cost formulas. These costs (formulas) are defined through different geographical areas or zones and different usage ranges. The data which were transplanted from MDSS were updated on Nov. 06th, 2012 while the rest data were collected on Apr. 13rd, 2013.

2.3 Decision Support Services

All the Web services are implemented in the Restlet framework [LTB12]. Nefolog is hosted on TomCat which is configured to listen on port 8080. All the URIs available in Nefolog are listed in the following[3]:

.../serviceTypes

- Returns: all the defined service types [Min13] and their URIs

Defined Service Types	Available Values in Database
Application	application
Web Sites	webSite
SQL Database	sqlDatabase
NoSQL Database	noSqlDatabase
Block Storage	blockStorage
Object Storage	objectStorage
Archival Storage	archivalStorage
Caching	caching
Infrastructure	infrastructure
Monitoring	monitoring
Network	network
DNS	domainNameSystem

- Short example for XML representation:

```
 <?xml version="1.0" encoding="UTF−8"?>
- <resource>
   <name>serviceTypes</name>
 - <content>
     - <type>
           <name>application</name>
           <uri>/serviceTypes/applications</uri>
       </type>
     - <type>
```

[3] Note: the string '...' in the URI denotes the address of the nefolog deployment (`http://129.69.214.239:8080/nefolog`)

```
            <name>sqlDatabase</name>
            <uri>/serviceTypes/sqlDatabase</uri>
        </type>
...
```

- Short example for JSON representation: .../serviceTypes?media=json

```
{"servicetypes":[
  {"name":"application","uri":"/serviceTypes/applications"},
  {"name":"sqlDatabase","uri":"/serviceTypes/sqlDatabases"},
    ...
]}
```

.../serviceTypes/{serviceTypeName}s

- Returns: for a certain service type, all the offerings with their URIs, providers and providers' URIs
- Short example for XML representation: .../serviceTypes/applications

```
...
    - <offering>
            <name>applicationPaaS</name>
            <uri>/offerings/applicationPaaS</uri>
          - <provider>
                <name>HpCloud</name>
                <uri>/providers/HpCloud</uri>
            </provider>
        </offering>
      - <offering>
            <name>AppEngine</name>
            <uri>/offerings/AppEngine</uri>
          - <provider>
                <name>Google</name>
                <uri>/providers/Google</uri>
            </provider>
        </offering>
...
```

.../providers

- Returns: all the providers and their URIs

Providers	Available Values in Database
Windows Azure	WindowsAzure
Google	Google
Amazon Web Services	AmazonWebServices
HP Cloud	HpCloud
rackspace	rackspace
Flexiscale	Flexiscale

- Short example for XML representation:

```
...
  - <provider>
      <name>Google</name>
      <uri>/providers/Google</uri>
    </provider>
   - <provider>
      <name>rackspace</name>
      <uri>/providers/rackspace</uri>
    </provider>
...
```

.../providers/{providerName}

- Returns: for a certain provider, all the offerings with their URIs, service types and service types' URIs
- Short example for XML representation: .../providers/Google

```
...
  - <offering>
      <name>AppEngine</name>
      <uri>/offerings/AppEngine</uri>
     - <servicetype>
        <name>application</name>
        <uri>/serviceTypes/applications</uri>
      </servicetype>
    </offering>
   - <offering>
      <name>cloudSqlPackage</name>
      <uri>/offerings/cloudSqlPackage</uri>
     - <servicetype>
        <name>sqlDatabase</name>
        <uri>/serviceTypes/sqlDatabases</uri>
      </servicetype>
    </offering>
...
```

.../offerings

- Returns: all the offerings with their URIs, service types and providers, also service types' URIs and providers' URIs
- Short example for XML representation:

```
...
  - <offering>
      <name>cloudSearch</name>
      <uri>/offerings/cloudSearch</uri>
     - <provider>
        <name>AmazonWebServices</name>
```

```
              <uri>/providers/AmazonWebServices</uri>
           </provider>
         - <servicetype>
              <name>application</name>
              <uri>/serviceTypes/applications</uri>
           </servicetype>
       </offering>
     - <offering>
           <name>cloudService</name>
           <uri>/offerings/cloudService</uri>
         - <provider>
              <name>WindowsAzure</name>
              <uri>/providers/WindowsAzure</uri>
           </provider>
         - <servicetype>
              <name>application</name>
              <uri>/serviceTypes/applications</uri>
           </servicetype>
       </offering>
...
```

.../offerings/{offeringName}

- Returns: for a certain offering, all the configurations with their URIs
- Short example for XML representation: .../offerings/VM

```
...
   - <configuration>
           <name>large</name>
           <uri>/offerings/VM/configuration_21</uri>
       </configuration>
     - <configuration>
           <name>small</name>
           <uri>/offerings/VM/configuration_19</uri>
       </configuration>
...
```

.../offerings/{offeringName}/configuration_{configid}

- Returns: all the performance characteristics and their values in the KB

Performance Characteristics	**Available Values in Database**	**Units/Allowed Values**
CPU Cores	cpuCores	
CPU Speed	cpuSpeed	GHz
RAM	memory	GB
I/O Performance	io	very high, high, moderate, low
Platform	platform	32, 64
Bandwidth	bandwidth	Mbps
Local Disk	storage	GB
SLA	sla	[0, 1]
Sites	sites	
Licence	licence	MySQL, Oracle, SQLServer, SQLWeb, SQLStandard
Operating System	os	Linux, Windows, Enterprise Linux
Caching	caching	MB
Transactions	transactions	
Data Transfer	datatransfer	
Concurrent Connection	concurrentConnectio	
Included I/O	included_IO	

- Short example for XML representation: .../offerings/VM/configuration_22

```
...
    - <performance>
           <name>cpuSpeed</name>
           <value>1.6</value>
       </performance>
     - <performance>
           <name>cpuCores</name>
           <value>8</value>
       </performance>
...
```

.../candidateSearch

- Returns: besides all the available performance characteristics, also the service type, offering and provider
- Short example for XML representation:

```
...
     <param>servicetype</param>
     <param>provider</param>
     <param>offering</param>
     <param>cpuCores</param>
     <param>cpuSpeed</param>
...
```

.../candidateSearch?all

- Returns: for each service type, the maximum values of each available numerical performance characteristic, and all existing values of each available non-numerical performance characteristic
- Short example for XML representation:

```
...
     - <Servicetype>
          <name>application</name>
          <uri>/serviceTypes/applications</uri>
        - <offering>
             <name>cloudService</name>
             <uri>/offerings/cloudService</uri>
           - <param>
              - <cpuSpeed>
                   <maxvalue>1.6</maxvalue>
                </cpuSpeed>
                ...
              - <io>
                   <value>moderate</value>
                   <value>high</value>
                   <value>low</value>
...
```

.../candidateSearch?{query}

- Returns: all the candidate offerings with their URIs which satisfy the user demands
- Syntax of the query part: All the demand are connected through symbol '&'. A demand is built by a performance characteristic and an expected value. The performance characteristic should be formulated through the available value in the database. The numerical values are interpreted as minimum, and the non-numerical values are exact
- Short example for XML representation: .../candidateSearch?servicetype=infrastructure&cpuCores=8&cpuSpeed=1.2&memory=10&os=Windows

```
...
     - <configuration>
```

```
          <name>30GB(Windows)</name>
          <uri>/offerings/cloudServers/configuration_487</uri>
       </configuration>
     - <configuration>
        - <name>
              m2.2xlarge Light Utillization High-Memory On-Demand
                  Instances
          </name>
          <uri>/offerings/elasticComputeCloud/configuration_381
       </configuration>
...
```

.../costCalculator

- Returns: besides all the available variables also configuration id, location_zone and usage_pattern

Variables	**Available Names in Database**
hours per month	Hour
storage size in GB per month	GB
transactions per month	Transactions
reports per hour	ReportsPerHour
days per month	Day
number of months	Month
messages per month	Messages
I/O operations per month	I/OOperation
number of Apps per month	App
number of accounts	Account
write operations per month	WriteOps
read operations per month	ReadOps
small operations per month	SmallOps
Stanza size per month	Stanza
number of opened channels	Channel
set of live SNI certificate per month	setsOfLiveSNIcertificate
put, copy, post or list requests per month	PutCopyPostListRequests
get and all other (except put, copy, post and list) requests per month	GetAndAllOtherRequests
storage size in GB per month	GBStorage
I/O requests per month	I/O

data in GB transfer out of service per month	GBExternalNetworkEgress
number of caching operations per month	NumberOfCaching
number of virtual IPs per month	NumberOfVirtualIP
Bandwidth in GB per month	GBBandwidth
bath upload requests per month	BathUploadRequests
GB of data stored in search domain per month	IndexDocument
glacier archive and restore requests per month	GlacierArchiveAndRestoreRequests
item size in KB per month	KBItemSize
read operations per second per month	SecondRead
write operations per second per month	SecondWrite
number of instances	Instance
metrics per instance per month	MetricsPerInstance
alarms over instance per month	AlarmsPerInstance
get, list or put requests per month	GetListOrPutRequests
HTTP requests per month	HTTPrequests
HTTPS requests per month	HTTPSrequests
upload and retrieval requests per month	UPLOADandRETRIEVALRequests
hosted zones per month	hostedZone
queries per month	Queries
get, head requests per month	GetHeadRequests
compute cycles per month	ComputeCycles
SSL capabilities per month	SSLcapabilities
Microsoft SQL Server storage in GB per month	GBmsSqlDBStorage
Domain name registration per month (per year)	DomainNameRegistration
number of servers	Server
File storage size in GB per month	GBFileStorage
number of VLANs per month	VLAN
number of usable IP addresses per month	NumberOfUsableIPaddresses
number of firewalled IP addresses	NumberOfFirewalledIPaddresses

Available Zones	Available Locations
NorthAmerica(US)	NY
	Oregon
	N.California
	LA
	Dulles
	Dallas
	Chicago
LatinAmerica	SaoPaulo
Africa	
MiddleEast	
EU	Ireland
	London
AsiaPacific	Singapore
	Tokyo
	Sydney
	HongKong
Worldwide	

- a short example for XML representation

```
...
     <variable>configid</variable>
     <variable>Hour</variable>
     <variable>GB</variable>
     <variable>Transaction</variable>
...
```

.../costCalculator?configid={configuration id}

- Returns: for a certain configuration, besides the necessary variables, also location_zone and usage_pattern
- Short example for XML representation: .../costCalculator?configid=332

```
...
     <variable>Hour</variable>
     <variable>Month</variable>
     <variable>GBExternalNetworkEgress</variable>
     <variable>location_zone</variable>
...
```

.../costCalculator?{query}

- Returns: for a certain configuration, the update date and all the initial condition, in each provided geographical area a total cost through adding upfront cost, data transfer cost and service cost together, also details per month for each single cost
- Syntax of the query part: All the demands are connected through symbol '&'. A demand is built by a variable and an excepted value. The excepted value of usage_pattern has a special syntax. All the necessary value of usage_pattern are formulated in parentheses. In the parentheses, the first part means the variable name which will be changed in the calculation. The second part shows the month of beginning while the third part shows the month of end. The last part means the changing rate of the variable.
- Short example for XML representation: .../costCalculator?configid=316& Hour=240&GBStorage=500&usage_pattern=(Hour,start=1,end=12,rate=10)

```
...
     - <querycollection>
          <staticquery>Hour=240</staticquery>
          <staticquery>configid=316</staticquery>
          <staticquery>GBExternalNetworkEgress=5000</staticquery>
          ...
       </querycollection>
     - <result>
          <location_zone>Northern Virginia</location_zone>
        - <cost>
              $5258.11
              <upfront>$1450.0</upfront>
            - <service>
                  $3808.11
                  <Month_Service>1st Month=$178.08</Month_Service>
                  <Month_Service>2nd Month=$195.89</Month_Service>
                   ...
               </service>
             - <datatransfer>
                   $0
                  <Month_Datatransfer>1st Month=$0</Month_Datatransfer>
                  <Month_Datatransfer>2nd Month=$0</Month_Datatransfer>
...
```

3 MiDSuS

3.1 Architecture of MiDSuS

MiDSuS is a new decision support system which focuses on finding a best cost of users' whole migration projects among different migration types[4]. There is an overview of

[4] MiDSuS: `http://129.69.214.239:8080/MiDSuS`

MiDSuS in Figure 3. The system is built through several JSP pages and Web APIs which are exposed from Nefolog. MiDSuS starts from selecting migration types with the help of application descriptions. Then the candidate offerings are found according to the user demands. After that, the costs of some candidate offerings which are selected by user from all the candidate offerings are calculated with the help of initial conditions. The user demands are collected into a dynamic table with the “Add” and “Delete" buttons. Finally, user can order the costs results by different Ranking parameters which in a drop down list.

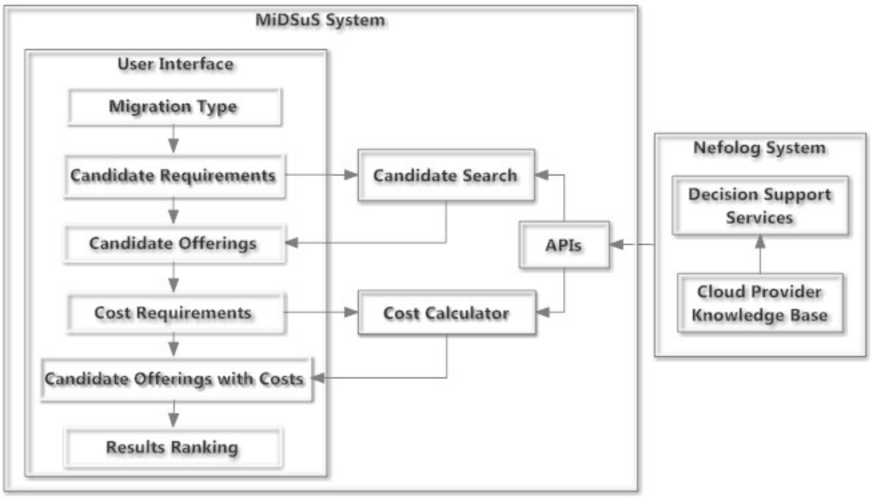
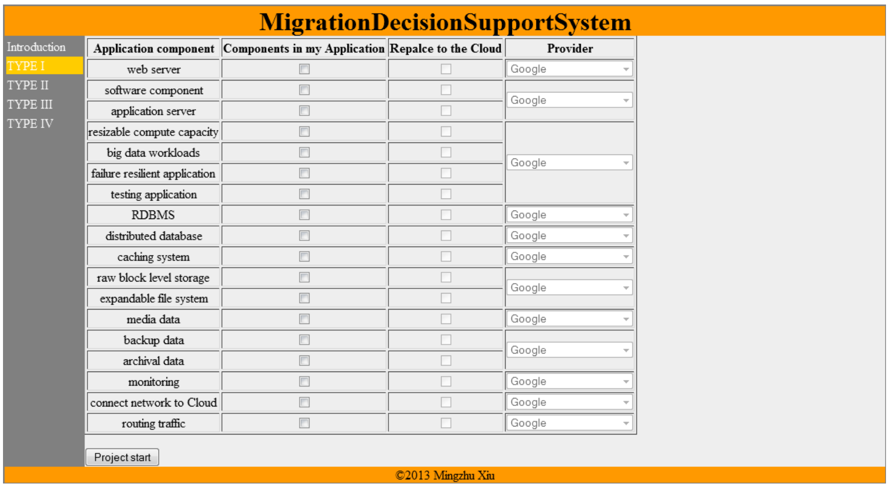


Figure 3: Architecture of MiDSuS


MigrationDecisionSupportSystem

Application component	Components in my Application	Repalce to the Cloud	Provider
web server	□	□	Google
software component	□	□	Google
application server	□	□	
resizable compute capacity	□	□	Google
big data workloads	□	□	
failure resilient application	□	□	
testing application	□	□	
RDBMS	□	□	Google
distributed database	□	□	Google
caching system	□	□	Google
raw block level storage	□	□	Google
expandable file system	□	□	
media data	□	□	Google
backup data	□	□	Google
archival data	□	□	
monitoring	□	□	Google
connect network to Cloud	□	□	Google
routing traffic	□	□	Google

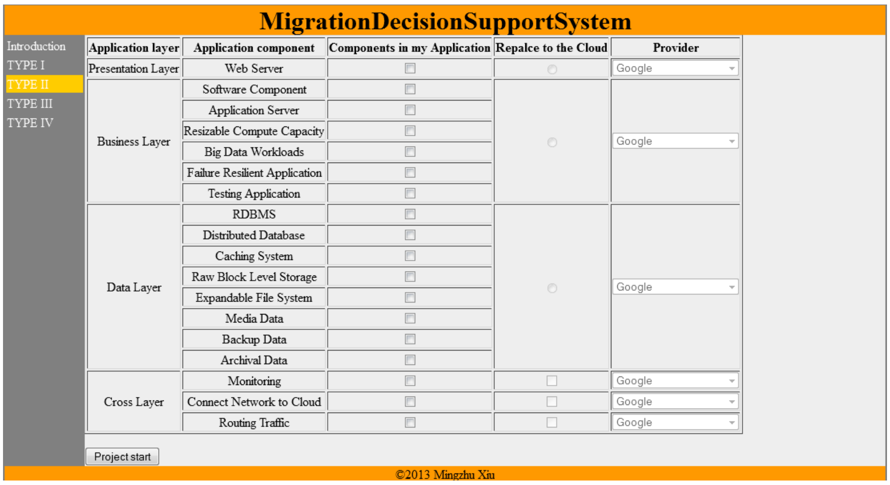
©2013 Mingzhu Xiu

Figure 4: User Interface of Migration Type I

3.2 Migration Type I

Migration Type I in [ABLS13] is a partial migration which distributes applications physically. Figure 4 shows all the distributed components of an application which are defined in [Min13]. Furthermore, the components are defined as different service types. Users can migrate some components with check boxes to the Cloud by the provider which they prefer. In other words, the migrated service types are determined as well as the provider. After that, the system runs by the steps in the user interface of Figure 3. In each service type, there are a set of candidate offerings with their costs and monthly details in a table.

3.3 Migration Type II

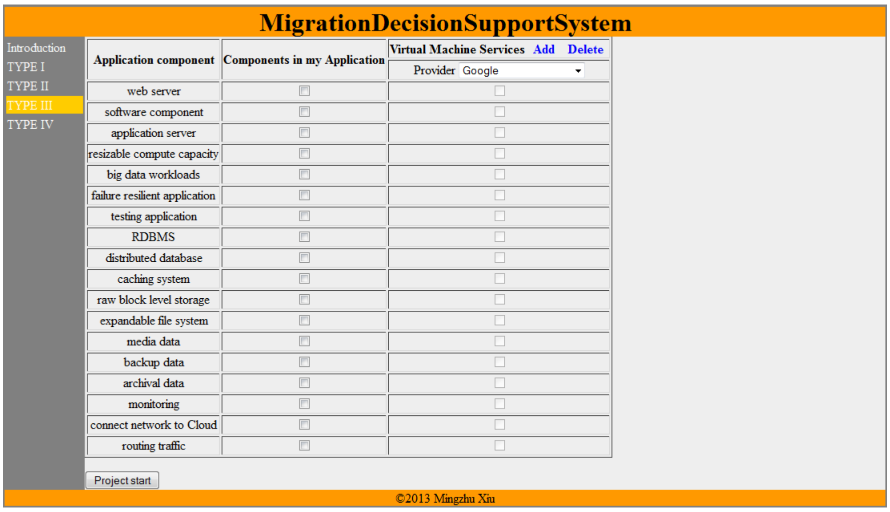


Application layer	Application component	Components in my Application	Repalce to the Cloud	Provider
Presentation Layer	Web Server	□	○	Google
Business Layer	Software Component	□	○	Google
	Application Server	□		
	Resizable Compute Capacity	□		
	Big Data Workloads	□		
	Failure Resilient Application	□		
	Testing Application	□		
Data Layer	RDBMS	□	○	Google
	Distributed Database	□		
	Caching System	□		
	Raw Block Level Storage	□		
	Expandable File System	□		
	Media Data	□		
	Backup Data	□		
	Archival Data	□		
Cross Layer	Monitoring	□	□	Google
	Connect Network to Cloud	□	□	Google
	Routing Traffic	□	□	Google

Figure 5: User Interface of Migration Type II

Migration Type II in [ABLS13] is also a partial migration which distributes applications with a three layer application architecture [Fow02]. Figure 5 illustrates the three layers, presentation layer, business layer and data layer. Besides the three layers, the other components which interact between two layers are defined as cross layer. The components in one layer should have an inner-relationship with other components. In MiDSuS, there are one or more service types in one layer. During the migration process only one layer can be migrated to the Cloud through a radio box. This means that the migration of each application layer is implemented through the migration of the service types which compose the selected application layer. If a user wants to choose more than one layer, another migration type should be considered. The next steps follow the order which is described in Figure 3.

3.4 Migration Type III

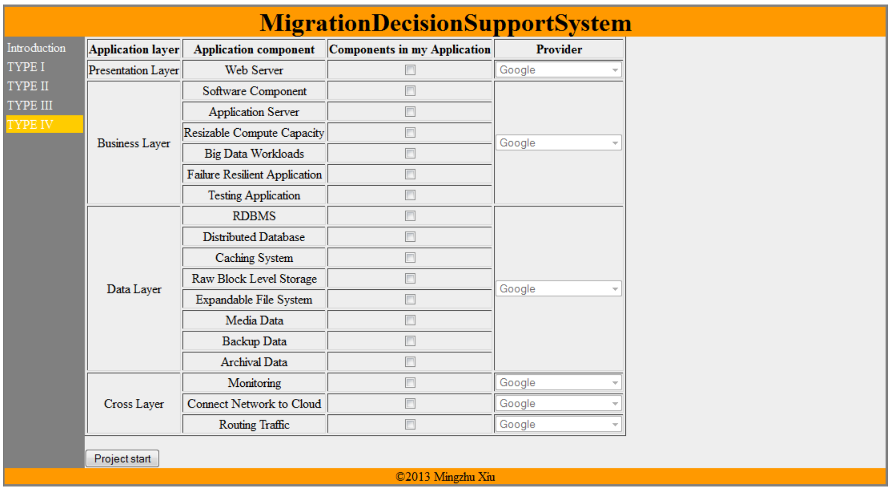


MigrationDecisionSupportSystem

Application component	Components in my Application	Virtual Machine Services Add Delete Provider Google
web server	☐	☐
software component	☐	☐
application server	☐	☐
resizable compute capacity	☐	☐
big data workloads	☐	☐
failure resilient application	☐	☐
testing application	☐	☐
RDBMS	☐	☐
distributed database	☐	☐
caching system	☐	☐
raw block level storage	☐	☐
expandable file system	☐	☐
media data	☐	☐
backup data	☐	☐
archival data	☐	☐
monitoring	☐	☐
connect network to Cloud	☐	☐
routing traffic	☐	☐

Figure 6: User Interface of Migration Type III

Migration Type III in [ABLS13] is a full migration with VM service. Furthermore, the distribution of application is similar with Migration Type I. It can be seen in Figure 6 that one VM offering can migrate one or more components and each component can be migrated through one or more VM offerings. So the table in the page of Type III is developed as a dynamic table and user can click the "Add" button to increase one more offering with expected provider. In fact, this migration type contains only one service type, infrastructure. All the following steps are processed around the offerings of the infrastructure service type.



MigrationDecisionSupportSystem

Application layer	Application component	Components in my Application	Provider
Presentation Layer	Web Server	☐	Google
Business Layer	Software Component	☐	Google
	Application Server	☐	
	Resizable Compute Capacity	☐	
	Big Data Workloads	☐	
	Failure Resilient Application	☐	
	Testing Application	☐	
Data Layer	RDBMS	☐	Google
	Distributed Database	☐	
	Caching System	☐	
	Raw Block Level Storage	☐	
	Expandable File System	☐	
	Media Data	☐	
	Backup Data	☐	
	Archival Data	☐	
Cross Layer	Monitoring	☐	Google
	Connect Network to Cloud	☐	Google
	Routing Traffic	☐	Google

Figure 7: User Interface of Migration Type IV

3.5 Migration Type IV

Migration Type IV [ABLS13] is a complete migration with any offering except VM service. Figure 7 shows a similar interface with Migration Type II, but all the composed layers should be migrated to the Cloud. The next steps follow the flow diagram in Figure 3.

4 Open Issues & Future Work

Nefolog

In the Nefolog system, a database updated with the latest data is essential. The candidate searches and cost calculations are based on the data which are provided by the providers. It makes no sense that the comparisons or calculations are implemented with expired data. Moreover, the meticulous categorization of similar service types are necessary in order to supply more opportunities for users to choose a suitable migration plan. Furthermore, with the increasingly offerings the description of the features of the offerings should be added into the list of performance characteristics. In the meantime, system should also achieve the changing of number of instances or storages [Zhe13]. Besides the technical performance characteristics the risks during a migration process should not be ignored either [KHSBT11].

MiDSuS

MiDSuS utilizes the decision support services from Nefolog. Among the services, the costs of selected candidate offerings are calculated one by one with the help of cost calculator service. In fact, the calculations can be solved in parallel in order to reduce the process time. In addition, an account management system is helpful for users who would compare all the migration plans to find a best one.

Actually, the viability of all the migration types in MiDSuS is yet validated. All the possible combinations of different offerings from different providers are listed in MiDSuS, but in fact, some of them are infeasible. The infeasible combinations should be removed from the candidate list in the future work.

Acknowledgments

The research leading to these results has received funding from projects 4CaaSt (grant agreement no. 258862) and Allow Ensembles (grant agreement no. 600792) part of the European Union's Seventh Framework Programme (FP7/2007-2013).

Special thanks to Santiago Gómez Sáez for his help with the execution environment.

References

[ABLS13] V. Andrikopoulos, T. Binz, F. Leymann, S. Strauch. How to adapt applications for the Cloud environment. *Computing*, 95(6):493–535, 2013.

[ASL13a] V. Andrikopoulos, Z. Song, F. Leymann. Supporting the Migration of Applications to the Cloud Through a Decision Support System. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, CLOUD '13, pp. 565–572. IEEE Computer Society, Washington, DC, USA, 2013. doi:10.1109/CLOUD.2013.128. URL `http://dx.doi.org/10.1109/CLOUD.2013.128`.

[ASL13b] V. Andrikopoulos, S. Strauch, F. Leymann. Decision Support for Application Migration to the Cloud. In *Proceedings of CLOSER'13*, pp. 149–155. SciTePress, 2013.

[Fow02] M. Fowler. *Patterns of enterprise application architecture.* Addison-Wesley Longman Publishing Co., Inc., 2002.

[KHSBT11] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, P. Teregowda. Decision support tools for cloud migration in the enterprise. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 541–548. IEEE, 2011.

[LTB12] J. Louvel, T. Templier, T. Boileau. *Restlet in Action: Developing RESTful Web APIs in Java.* Manning, 2012.

[Min13] Mingzhu Xiu. *Decision support for different migration types of applications to the Cloud.* Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2013. URL `http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIP-3472&engl=0`.

[Zhe13] Zhe Song. *A decision support system for application migration to the Cloud.* Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2013. URL `http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIP-3381&engl=0`.