Service OrientedMarketing Decision Support Systems(SOMDSS) for Big Data in the Cloud. Some orchestration, provisioning and deployment challenges for marketing scientists

Michel CALCIU
Université Lille, RIME-Lab, France
mihai.calciu@univ-lille.fr

Jean-Louis MOULINS
Aix Marseille Université, CRETLOG, France
jean-louis.moulins@univ-amu.fr

Francis SALERNO
Université Lille, LEM, France
francis.salerno@univ-lille.fr

Abstract

The new era in information systems (IS) brought over bythe Big Data(BD) phenomenon, comes with important challenges for society and enterprises. Decision Support Systems (DSS), in this new context, should help managementto develop agility in reaction to the data tzunami inputs they are facing on a daily basis. DSS for Big Data are best implemented in the cloud in a servitized way. In Marketing who is at the forefront of Big Data generation and use, DSS need to follow this trend in order to cope with Big DataVolume, Velocity and Variety (3V).

We introduce an evolutionary view for DSS and specifically for MDSS in order to explain the shift from toolbox-centered DSS to service-centered DSS in the cloud. We then decline the several Service Oriented Approaches that have dominated service-centered DSS in order to distinguish SOMDSS in the Cloud as the strongly preferred pattern for dealing with Big Data. We finally insist on some new competences marketing scientist need to acquire in order to deal with Big Data computing in the Cloud.

Introduction

Two decades after the first commercial digital computers appeared (in 1954), Decision Support Systems were introducedas a new form of information systems that use models in order to assist managers' decisions. With the advent of microcomputers, they became an area of research of its own in the middle of the 1970s and developed further during the 1980s. In 1966, Kotler introduced the concept of a ''MarketingNerveCentre'', providing marketing managers with ''computer programs which will enhance their power to make decisions.'' (Wierenga & al., 2008, p. 561). The concept and framework of a marketing decision support systems (MDSS) has been introduced by Little (1979) and is based on the so called DDM (Dialogue, Data, Model) paradigm of Sprague and Carlson (1982) that formed the dominant architecture for DSS from the sixties throughout the eighties (Eom, 1995). The same author defined MDSS as a ''coordinated collection of data, systems, tools and techniques with supporting software and hardware by which an organization gathers and interprets relevant information from business and environment and turns it into an environment for marketing action'' (Little, 1979, p. 11). This definition exhibits premonition as to the scalability of the evoked components and remains suitableto SOMDSS for Big Data in the cloud as will be shown in this paper.

According to Sol & al. (1987) the definition and scope of DSS has been migrating over the years: in the 1970s DSS was described as "a computer-based system to aid decision making"; in the late 1970s the DSS movement started focusing on "interactive computer-based systems which help decision-makers utilize data bases and models to solve ill-structured problems"; in the 1980s DSS should provide systems "using suitable and available technology to improve effectiveness of managerial and professional activities", and towards the end of 1980s DSS faced a new challenge

towards the design of intelligent workstations. So the definition evolved from the single user and model-oriented DSS to group decision support systems (GDSS) and organizational decision support systems (ODSS). Wierenga & Van Bruggen (1997) used the term ‘‘marketing management support systems’’ for systems additionally supporting marketing decision-making in weakly-structured areas that are primarily qualitative, and knowledge-driven.
Wether at single user, group or enterprise level, servitizing DSS has become progressively attractive due to network advances.
Spohrer & al. (2007) defines a service as the application of competence and knowledge to create value between providers and receivers. Service systems include all human made systems that enable and/or grant diverse entities access to resources and capabilities such as transportation, water, food, energy, communications, buildings, retail, finance, health, education and governance (Spohrer & Demirkan, 2011). By analogy information systems can be seen as service systems as they support people in intentions and purposeful actions (Checkland & Howell, 2005). DSS who support purposeful actions are therefor also types of information systems and service systems (Demirkan &Dursun, 2013).
We introduce an evolutionary view for DSS and specifically for MDSS in order to explain the shift from toolbox-centered DSS to service-centered DSS in the cloud. We then decline the several Service Oriented Approaches that have dominated service-centered DSS in order to distinguish SOMDSS in the Cloud as the strongly preferred pattern for dealing with Big Data. We finally insist on some new competences marketing scientist need to acquire to deal with BD computing in the Cloud.

MDSS as Web Services

The trend in DSS is moving from toolbox-centered architectures towards service-centered architectures. In toolbox-centered architectures, application and data are situated on the user’s computer or local area network. In service-centered architectures, the tools and data are situated on remote computers, typically accessed through Internet connections.
An important difference in a web based MDSS is that its components: the model, the data, statistics and optimization are no longer necessarily located on the manager's desktop computer but can be accessed in a secure way from anywhere in the “universe”. Some additional differences between toolbox-centered and service-centered GMDSS are enumerated in table 1.

Table 1- Some differences between toolbox-centered and service-centered MDSS

	toolbox-centered (local)	**service-centered (web-based)**
Components power/location/integration	limited/concentrated/ integrated	complete/distributed/loosely integrated
Data access control	not available	available
Decision Support API	non-standard, proprietary	web application standards
Maintenance& upgrading	optional, user concerned	automatic, user not concerned
Software ecosystem resources	limited	unlimited

Toolbox systems use to integrate limited database management and computation facilities in a standalone piece of software while in service-centered ones the main DSS (Decision Support System) components are distributed, loosely integrated and complete. The latter can provide data access control, meaning that «data themselves may not be freely distributable, but certain derived products (such as visualizations or generalizations) may be» (Bivand & al, 2008, p.6).
Building model based decision support to solve a particular managerial problem must often rely on proprietary, non-standard Decision support API-s provided by toolbox-centered software while the service-centered approach can exclusively rely on open web-application standards.
Maintenance and upgrading of software and data in service-centered solutions is done by the provider and end users don't have to bother about such matters. For an example of a web service for Geo-Marketing decisions support see Calciu(2012) and Calciu & Willart (2012).

With web services the DSS software ecosystem[1] becomes virtually unlimited. The software ecosystem metaphor is suitable for open computer systems like Linux/Unix or Big Data calculations and management environments like Apache Hadoop. It also can apply to specialized software systems as the continuously evolving statistical system R (R Development Core Team, 2011) .
Service-centered approaches, unlike toolbox-centered ones, have unlimited access to such software ecosystem resources and can exploit their continuously improved facilities.
Embedding models as web services: Lilien and Rangaswamy (2000, 2008) distinguish marketing modeling applications as to their degree of visibility and integration. According to this classification, models can be visible or embedded and standalone or integrated. To the original two criteria degree of integration and degree of visibility we add the client-server dimension. We distinguish between local, remote and hybrid MDSS (Calciu, 2013). When both the visual (interactive) logic and the model logic are concentrated on the client computer the application is "local". When both are on the server it is "remote" and when the visual logic is on the client and the model logic is totally or partly on the server it is "hybrid".
Probably one of the nicest ideas for freely publishing, finding and flexibly embedding models over the Internet is the web services technology.
Web services deal with embedded models, be they standalone or integrated pieces of software. Embedded distributed applications can be accessed over the network at a lower level through RPC (Remote Procedure Calls) and at a higher level as Web services. The latter provide envelopes that make such applications "universally" discoverable, able to communicate and interact with over the network.

Building SOMDSS as Web services

Web services (WS) are self-contained, self-describing, modularapplications that can be published, located, and invoked across the Web (Calciu, 2013). They can easily integrate with other services, from the same or different companies, to create a complete business process. This interoperability allows to dynamically publish, discover and bind a range of Web services through the Internet.
The following standards play key roles in Web services: Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), Simple Object Access Protocol (SOAP). They constitute together with some additional technologies such as WS-Addressing, WS-ReliableMessaging, WS- Security the "Big" Web services technology stack (Pautasso & al., 2008, Richardson & Ruby, 2007) that defines the WS-* standard.
This WS-* standard came under criticism as to its presumed complexity. The alternative *REST* (*REpresentational State Transfer*) solution takes its inspiration from the web itself and shows that the same principles that have made the success of the World Wide Web can be used to solve enterprise application integration problems and to simplify service-oriented architectures. MDSS are aimed to integrate as enterprise applications and can therefore benefit form such technological trends (Calciu & al., 2013).
REST is a style of software architecture for distributed systems such as the World Wide Web. The WWW itself is probably the largest implementation of REST principles. REST was introduced and defined in his doctoral thesis by Roy Fielding (2000), one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification (Fielding & Tailor, 2002). A service is considered "RESTful" if conforming to REST principles (Calciu & Micheaux, 2014). RESTful Web services are gaining increased attention because of their publishing and consumption simplicity (Vinowski, 2002).
These web service standards and approaches, and particularly REST are used in service-oriented (SO) solutions for the cloud that are preferred in Big Data calculations. Any marketing decision support model, can be easily embedded as such a web service.
Probably the quickest and most efficient way to bring applied statistics and in particular marketing models to the market is *OpenCPU* (see http://opencpu.org). OpenCPU provides a mature and robust

1 "a collection of software systems, which are developed and co-evolve in the same environment" (Lungu, 2009)

system for hosting R based services. It exposes a simple HTTP API for calling R functions, scripts and managing data that in many aspects satisfies REST principles.

For marketing scientist and data scientists, who have massively adopted R as their statistical system, it is rather easy to discover and execute statistical functions, using familiar http addresses (url-s). To discover for example the function "rnorm" that generates normal random values it is enough to connect to the address http://marketing.iae.univ-lille1.fr/ocpu/library/stats/R/rnorm on the authors's openCPU service, by using a webclient be it a browser or a command-line program like "curl". The discovery mechanism, that is so important in defining web services, can be used progressively. It exposes the HTTP method GET. By first doing http://host/ocpu/library/ the service returns all available R packages on that server. Then intuitively asking for all functions available in the "stats" package by doing http://host/ocpu/library/stats/R/ one receives a long list containing also rnorm. Again by intuitively adding to the uri path ./rnorm one gets the arguments expected by the function. To execute the function with the chosen argument values a HTTP method POST will be used either from a HTML form on a web page or from command-line doing curl http://host/ocpu/library/stats /R/rnorm/print -d "n = 10&mean = 10". The latter will print the expected results, that is 10 normally distributed values around the mean 10.

Implementing a RESTfull SOMDSS: Using the same RESTfull approach we have implemented several SOMDSS for predicting customers' dynamic purchase behavior from their Recency and Frequency represented by "buy till you die" stochastic processes (Fader & al., 2005; Batislam & al., 2007). After having implemented and "packaged" in R most of the mentioned models, we have also developed, so called "adhoc" packages based upon the first.

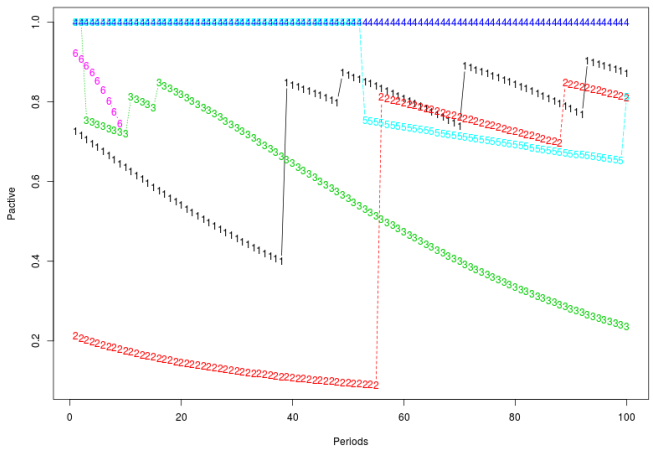
Someone, who wants to see how the probability of being active varies for customers defined by their purchase Recency and Frequency, can use the same intuitive approach as before. By doing http://host/ocpu/library in order to find in the long list of R packages the ones beginning with the name "adhoc" he would discover the package called "adhocpactive". Then by adding to the uri path ./adhocpactive/R he would discover that there is a function called "plotSimPactive" and, by adding it to the path, he would see its expected arguments. Even more by changing the path to ../man/plotSimPactive/ he would get access to the help document for that function. The latter is in text format by default. Other formats can be obtained when adding ./pdf or ./html to the path.

After this *process of RESTful service discovery* the function can be executed with a HTTP POST:

```
curl http://host/ocpu/library/adhocpactive/R/plotSimPactive/png -d 'modelname =
"BGNBD"&per = 100&n = 6' --output plotSimPactive.png
```

The resulting graph will show the predicted probability of being active for 6 randomly extracted customers from the "cdnow" database (see. Fader & al., 2005) with their given initial Recency and Frequency using the parameters of the already calibrated "BGNBD" model. Their future purchase behavior (buy or not) is simulated for the indicated 100 days period. Adding a simple web interface to this R package turns the models visible and makes the SOMDSS interactive and userfriendly, as can be seen at http://marketing.iae.univ-lille1.fr/ocpu/library/adhocpactive/www/.

Figure 1- Probability of being active for six customers during hundred days

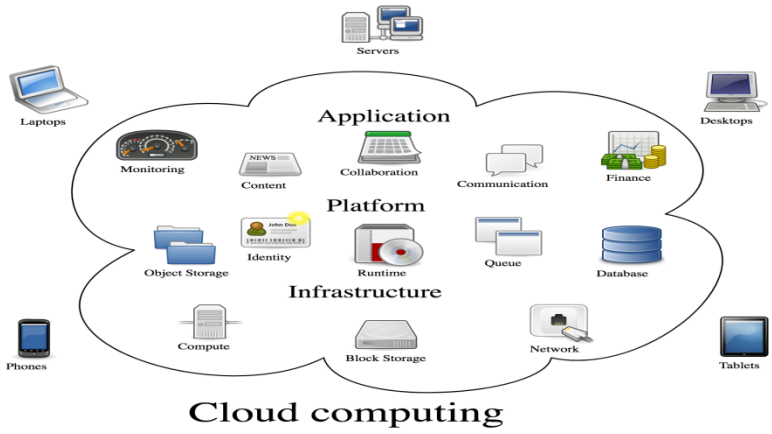


The day a customer buys, his probability of being active jumps to attain a peak and eventually slowly falls thereafter during the consecutive days without purchase with an acceleration that depends on the customer's initial Recency and Frequency. The simulated behaviorrespects each customer's Frequency. For example in Figure 1 two customers with very frequent purchase and none or very short inter-purchase periods maintain a very high probability of being active while for the others this probability visibly declines during inter-purchase periods that correspond to known customer frequency. For more details concerning Restfull SOMDSS one could refer to Calciu & al. (2012).

Cloud services and Big Data

Big Data come with additional difficulties for academic model based MDSS providers. Volume, Variety and Velocity that characterize such data need huge resources in order to provide solutions for agile managerial response. While offering classical MDSS as web services is possible using resources that normal individual researchers own or control, Big Data need huge computers or clusters of computers that normally are owned and controlled by entities or institutions like universities, big companies or cloud solutions providers, that can afford them. Such computing resources can flexibly be shared using cloud technology.

Figure 2- Cloud computing



The American National Institute of Standards and Technology (NIST) defines c*loud computing* "as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloudsare composed of three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), and they have four deployment models: Public, Private, Hybrid, Community.

The main enabling technology for cloud computing is virtualization. *Virtualization* software separates a physical computing device (bare metal environment) into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks. With operating system level virtualization essentially creating a scalable system of multiple independent computing devices, idle computing resources can be allocated and used more efficiently. *Virtualization* provides the agility required to speed up IT operations, and reduces cost by increasing infrastructure utilization. While SaaS tends towards autonomic computing the other two service models need some user intervention. *Autonomic computing* automates the process through which the user can provision resources on-demand. By minimizing user involvement, automation speeds up the process, reduces labor costs and reduces the possibility of human errors. (Hamdaqua & Tahvildari, 2012).

The difference in term of hardware and software elements that can and need to be managed on a cluster between the on premisses "bare metal" computers and the three cloud service models may be seen in Table 2.

Table 2Differences between the service models in the Cloud and on premisses

	On premises	In the Cloud		
You manage ?	Bare metal	IaaS	PaaS	SaaS
Applications	y	y	y	n
Runtime[2]	y	y	y	n
Middleware[3]	y	y	n	n
OS	y	y	n	n
Virtualization	y	n	n	n
Servers	y	n	n	n
Storage	y	n	n	n
Networking	y	n	n	n

Adapted from Watts (2017)

In the cloud everything is a service be it software applications, platforms or infrastructure (see figure 2). But cloud services are more complex than web services. They include both the ability to run a service and concepts of scalability, eventual multi-tenancy[4]or even a business component (billing, etc.). A Web Service is just a technological mechanism to implement a software capability. A service may be exposed as a web service, but just being a web service does not make something an example of Cloud Service.

Cloud computing finding its root in Services Oriented Architecture (SOA)all cloud services use web services of any category SOAP, JSON or REST.

Building Big Data MDSS in the Cloud

B*uilding MDSS for Big Data comes with additional complexity*.Besides complexities coming from the variety of data andthe velocity expected, Big Data volume imposes splitting data and calculations among several computers (nodes)that are organized in clusters and preferably managed behind clouds as virtual machines or, more recently, containers.Most mathematical calculations can be adapted for BigDatausing *MapReduce*, a programming model and an associated implementation that has “democratized” processing big data with a parallel, distributed algorithm on a cluster. The approach hides the complexities of massively parallel distributed computing, letting the marketing or data scientist concentrate on two procedures: the map method which associates keys and values to datasets forfurther filtering and sorting and the reduce method that performs summary operations based upon the previously associated keys. It is inspired from the map and reduce higher order functions commonly used in functional programming. Introduced as a proprietary Google technology in 2004, it became generic and its most popular implementation is Apache Hadoop.

MapReduce and its improvements like Apache Spark use Share-Nothing as the strongly preferred distributed-computing pattern in a clusterof commodity computers. At this stage of technology the alternative classical Share-Everything pattern is not a choice because network latency and congestion are the bottleneck relative to main memory and local storage. The key contributions of

2 Runtime is the period of time when a program is running. It begins when a program is opened (or executed) and ends with the program is quit or closed. When a program is in the runtime phase, the application is loaded into RAM. This includes the executable file and any libraries, frameworks, or other files referenced by the program. When the program is quit, the runtime period ends and the memory used by the program is made available for use by other programs. Programmers sometimes distinguish between what gets embedded in a program when it is compiled (compile time) and what gets embedded or used at runtime (or load time) like shared libraries like dlls.

3 Middleware is computer software that provides services to software applications beyond those available from the operating system. In distributed applications this role can be played by web servers, application servers or other tools that support application delivery.

4 software architecture in which a single instance of software runs on a server and serves multiple tenants. A tenant is a group of users who share a common access with specific privileges to the software instance.

MapReduce implementations are the scalability and fault tolerance achieved by optimizing the execution engine.
Hadoop as a federating MapReduce framework has become an ecosystem formed by numerous Apache Software Foundation projects that make up the services required by an enterprise to deal with Big Data in an agile way. For example Hortonworks, a major data analytics vendor, groups these open-source components into a Data Platform with 23 components[5] associated to five pillars: Data Management (2), Data Acces (13), Data Governance & Integration (3), Security (2), Operations (3). A model based MDSS will probably not use all the components mentioned above but for the computational aspects several of these Hadoop ecosystem components need to be installed on all Virtual Machines of a cluster and orchestrated.

Orchestrating[6] the MDSS infrastructure and ecosystem

Our MDSS computing power component is hosted on the authors' University cloud[7] that uses OpenStack, a free and open-source software platform for cloud computing, mostly deployed as an infrastructure-as-a-service (IaaS). Paradoxically public cloud platforms are dominated by private solutions like Amazon AWS, Google Cloud Platform, IBM Cloud or Microsoft Azure. Open-source platforms seem to loose ground in this field. On the other hand private cloud solutions for enterprises, or universities that don't outsource their computer centers are better served by open-source platforms like OpenStack.
Configuring[8] the local (client) computer to use the remote cloud: Although OpenStack has a quite powerful web-interface (called Horizon) which allows to launch virtual computer instances of various linux system versions and flavors[9] from available virtual machine images[10] these images usually don't have big data calculation software installed. Therefore the main command-line OpenStack clients need to be installed (keystone, nova, glance, cinder, neutron) in order to automate install operations by preparing shell commands that will be executed on the remote computer instances. First it is necessary to generate the ssh key[11] on the local computer and register it with the cloud[12] in order to be able to remotely access the cloud administration server. A configuration file to create the client environment consisting of username and password (also tenant name and the server's authentication url) allows the marketing scientist to remotely-control the Big Data MDSS infrastructure using OpenStack client command-line and shell commands. After all SSH[13] that allows to execute command-line "shell" on remote computers is the most common orchestration tool of all and preexisted all the others.

---

5 Apache Hadoop YARN, HDFS, Apache Hive , Apache Pig, MapReduce, Apache Spark, Apache Storm, Apache Hbase, Apache Tez, Apache Kafka, Apache Hcatalog, Apache Slider, Apache Solr, Apache Mahout, Apache Accumulo, Workflow Management, Apache Flume, Apache Sqoop, Apache Knox, Apache Ranger, Apache Ambari, Apache Oozie, Apache ZooKeeper

6 Orchestration means arranging or coordinating multiple systems. It is also means running the same tasks on several computers at once, but not necessarily all of them.

7 The cloud consists of 336 cores, 2.1 To RAM and 215 To de storage and a total power of de 6,5 Tflops

8 Configuration normally takes "facts" to make true about a server. For example insure that a configuration file contains a given instruction or verify that configuration files needed for an application are present. Configuration is part of provisioning. Provisioning often implies it's the first time you do it. Configuration management usually happens repeatedly.

9 flavors define the compute, memory, and storage capacity ofcomputing instances. A flavor is an available hardware configuration for a server. It defines the size of a virtual server that can be launched.

10 A virtual machine image is a single file which contains a virtual disk that has a bootable operating system installed on it

11 ssh-keygen -t rsa -f ${HOME}/.ssh/cloud-hpc-lille

12 nova keypair-add --pub-key=${HOME}/.ssh/cloud-hpc-lille.pub mcmac2key

13 SSH or Secure Shell is a cryptographic network protocol that gives users a secure way to access a computer over an unsecured network. It is mainly used for remote command execution, but any network service can be secured with SSH.

Provisioning[14] *the remote scalable MDSS infrastructure:* Controlling the MDSS infrastructure means launching virtual computer instances by choosing the operating system among several linux distribution images (in our case Ubuntu 14.04.2 - 64 bit) and a given flavor (in our case 8-CPU-12GB-RAM) as well as the network on which these virtual computers will form a cluster.
This can be done interactively by using command-line or a shell program as in Listing 2 (see Appendix) or pragmatically using an orchestration and/or provisioning tool like *Ansible*. The latter is considered one the most prominent tools applying infrastructure as code principles. Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools (Wittig & Wittig, 2016).
The launched instances can then be interactively or programmatically observed, stopped, started or even deleted. The size of the cluster can be adapted within the limits of the quota previously attributed by the cloud administrator[15].
Another aspect of Infrastructure as a Service (IaaS) is controlling network infrastructure by associating public IP addresses to some computers in the clusters and ensuring their firewall protection through association to security groups. Public IP addresses allow among others connecting to and controlling the instances and viewing on the web browser the calculation stages of the Big Data cluster engines that will be installed on each computer. For the Big Data calculations engine (here Apache-Spark) one computer instance will be given the role of a master and the others the role of workers (slaves).
Finally in order to allow all computer instances to interact during calculations, as if they were one computer, the computation engine user (here hadoop) on each machine needs password-less ssh network access to all cluster members. This is done by generating a public and private key for the computing engine user and distributing it together with the ssh configuration files to all cluster members.
Deploying[16] *the computational ecosystem components:* In order to deploy several of the Hadoop ecosystem components on all Virtual Machines we used Ansible for orchestration purposes. We adapted a solution suggested by Borisenko & al. (2016)[17] that uses Ansible playbooks in a loosely coupled way in order to be able to choose among any Apache Spark and Apache Hadoop versions and provide an ability to add any additional components of the Hadoop ecosystem. A reason for this is that Apache Spark, which is todays most powerful Big Data computing engine, is under heavy development and versions change more frequently than OpenStack releases.

MDSS in Containers

Containers, a new trend in virtualization, may substantially reduce the MDSS orchestration burden. Containerization has gained ground as an alternative to virtualization. In fact containerization can be seen as a special kind of virtualization that occurs at operating system level while in Virtual Machines it occurs at hardware-level.Historically, in Unix systems, the first containers just provided isolation of the root file system (via chroot). Later FreeBSD jails extended this to additional namespaces such as process identifiers. A modern container is more than just an isolation mechanism: it also includes an image that contains the files of the application that runs inside the container (Burns & al., 2016). While much more lightweight than Virtual Machines (VMs), containers provide resource-management tools that make running applications efficient. They also

14 Provisioning for operations professionals refers to getting computers or virtual hosts to use and installing needed libraries or services on them.
15 Our quota consisted of 8 instances with 8 cores each (meaning a total of 64 cores),12 Gb RAM (meaning 96Gb for the cluster) and 25 Gb disk storage (meaning potentially 200 Gb for the HDFS file system).
16 Deployment is the process of getting an application and its dependencies installed on a computer. For developers it may also mean the process of preparing the server, perhaps by installing libraries or daemons while operations professionals , would use the word “provisioning” for that.
17 https://github.com/ispras/spark-openstack

provide robust kernel-level resource isolation to prevent the processes from interfering with one another.
Docker is a tool that can package an application and its dependencies in a virtual container that can run on almost any server, out of the box, without installing any software. The resulting flexibility and portability allowing running applications everywhere will be illustrated here with the building and packaging of the RESTfull SOMDSS mentioned above.
Container based DevOps[18] *to implement MDSS using OpenCPU:* Because OpenCPU is completely open source, SOMDSS can be built and shipped on DockerHub, a cloud-based registry service which allows to ship Docker images in a public repository.
Downloading the combined opencpu/rstudio image and creating (running) a container giving the full flexibility of a Linux box, without the need to install anything on the host system (in our case a OS-X on a MacBooPro machine) is done with:
docker run -t -p 8004:8004 opencpu/rstudio
After this *http://localhost:8004/ocpu/* and *http://localhost:8004/rstudio/*can be accessed with a browser or command-line web client. Logging in via rstudio with user: opencpu (passwd: opencpu) allows to build or install packages and their web-ready versions called apps. Following the classification of Lilien & Rangaswamy (2000, 2008), while R packages support embedded models, apps, that are specific to OpenCPU are R packages coming with a full fledged web interface, make the models visible.
With*docker exec*one connects to a root shell of the minimal system running in the container. This gives full control allowing to install additional software in the server, customize the web server, modify R options, optimize performance by preloading data or packages.

Minimizing MDSS orchestration with Containers

As we could see containers can substantially reduce the MDSS building, deployment and orchestration burden on one machine. But what about MDSS for Big Data that need to do their calculations on several machines. Kubernetes is an open source container-centric management environment developed at Google, that facilitates both declarative configuration and automation. It orchestrates computing, networking, and storage infrastructure on behalf of user workloads. This provides much of the simplicity of Platform as a Service (PaaS) with the flexibility of Infrastructure as a Service (IaaS), and enables portability across infrastructure providers. Itfavors development of DSS that run in a cluster by making it easy to deploy and manage complex distributed systems, while still benefiting from the improved utilization that containers enable. In this approach containers can be grouped in so called pods that can play an equivalent role as Virtual Machines in a cluster.
As Kubernetes requires users to supply images that can be deployed into containers within pods, Apache Spark ships with a bin/docker-image-tool.sh script that can be used to build and publish the Docker images to a repository as can be seen in listing 1.

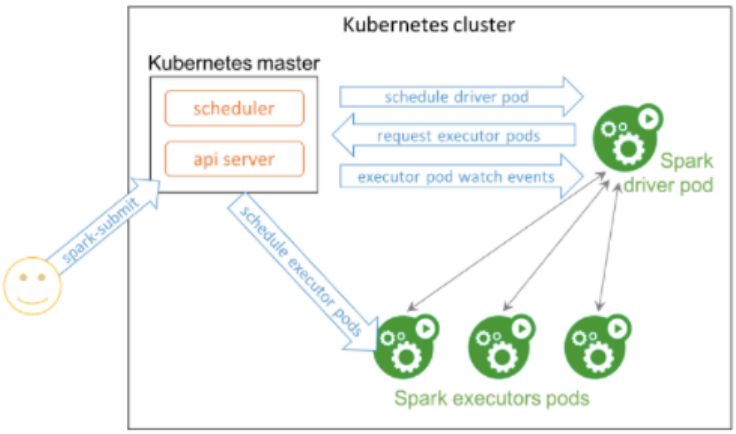
Listing 1

```
./bin/docker-image-tool.sh -m -t spark-docker build
./bin/docker-image-tool.sh -r <repo> -t spark-docker push
```

Spark creates a Spark driver running within a Kubernetes pod. The driver creates executors which are also running within Kubernetes pods and connects to them, and executes application code.

18 DevOps is a software development methodology that combines software development (Dev) with information technology operations (Ops).

Figure 3– Apache Spark on Kubernetes



Apache Spark running natively in a Kubernetes cluster

Source: https://kubernetes.io/blog/2018/03/apache-spark-23-with-native-kubernetes/

The listing that runs an example application on 3 executor instances as in figure 3 is given in Listing 3 in Appendix.

Discussions and Conclusion

Academic Marketing scientists develop decision support models or find new uses to existing models offering better managerial solutions to marketing problems. These solutions, in order to be used, need to be implemented as MDSS. We show that SOMDSS have become mainstream due to the diffusion of web services technologies and that their complexity increases with the advent of Big Data. In contrast with SOMDSS that don't need more than one computer, Big Data SOMDSS come with additional orchestration, provisioning and deployment challenges. Both can be implemented in the cloud and both benefit from the facilities and simplifications brought over by containerization. We adopted a hands-on approach trying to offer demonstrative implementations of SOMDSS both on one computer and on a cluster of computers in the cloud.

Although most of the orchestration, provisioning, deployment, configuration or even containerization challenges we have mentioned above might be transferred to intermediaries, being knowledgeable about them may significantly help bring academic marketing models to the market.

If we take the stochastic models we have mentioned in the paper, the authors who have improved the now classical ParetoNBD model by introducing BGNBD, that allows much faster estimation, have also launched a series of extensions and improvements to these models. Then other authors joined in creating new models like MBGNBD (Batislam & al, 2007), or later Pareto/GGG (Platzer & Reutterer, 2016) etc. Then a small team around one of the authors of the BGNBD model has developed a R package called BTYD meaning "Buy Till You Die", signifying the kind of purchase behavior that is represented by these models. In the summer 2014 they published it on the selective CRAN Repository containing thousands of very thoroughly tested R packages. At the end 2016 one of the authors of the Pareto/GGG published on the same repository the BTYDplus package that includes their models and the other significant models in this category, that were not included in the first package. In this way all those models came very close to market as embedded models.

A next step would have been to build the container image of the two R packages and the OpenCPU hosting R services system, and publish it on DockerHub. Therefrom other people could download that image and run the container on their servers or in the cloud and offer this embedded collection of models as a RESTfull web service equivalent to a SOMDSS.

Further research is needed to make the implementation of such models suitable for Big Data and adapt their calibration algorithms to MapReduce.

There is already much evidence that a growing part of business application software spending is and will be on software-as-a-service, instead of as product licenses. Also for more and more companies, the pay-as-you-go service-oriented computing model like cloud computing, with having someone

else worrying about maintaining the hardware and software are becoming very attractive (Marston, 2011).

For these reasons we think that marketing model builders and marketing scientists in general need to have a grasp of service oriented and cloud technologies as this is increasingly the way, and for Big Data it is probably the only way, their models and MDSS can be brought to the market.

References

Batislam, E. P., Denizel M, and Filiztekin A.(2007), Empirical Validation and Comparison of Models for Customer Base Analysis, *International Journal of Research in Marketing*, 24 (September), 201–209.

Bivand, R.S., Pebesma E.J. & Gomez-Rubio V. (2008). *Applied Spatial Data Analysis with R*, Springer

Borisenko, O., Pastukhov R., Kuznetsov S. (2016), Deploying Apache Spark virtual clusters in cloud environments using orchestration technologies, *Proceedings of ISP RAS*, 28, 6,111–120

Burns B., Grant B., Oppenheimer D., Brewer E. & Wilkes B. (2016), Borg, Omega and Kubernetes: Lessons learned from three container management systems over a decade. *Communications of the ACM*, May, 59, 5, 50-57.

Calciu M., Willart S. (2012) "Construction d`un Service Web d`Aide a la Decision Geo-Marketing a partir d`Outils OpenSource", *15eme Colloque International Etienne Thil sur la Distribution*, Lille, 29 - 30 novembre

Calciu M. (2012) "Model based Spatial Marketing Decision Support open Architecture over the Internet. An application to the Gravity Polygons Model.", *41st EMAC Annual Conference* , Lisbon, 22-25 May

Calciu M. (2013) "Local, Remote and Hybrid Web Solutions to bridge the Academic-Practitioner Divide in Marketing Decision Models", *29eme Congres de l'Association Française du Marketing*, La Rochelle, 16-17 Mai

Calciu M., Meyer-Waarden L., Salerno F., Willart S. (2013) Decision support for valuing customers as RESTful web services, *42nd EMAC Conference*, Istanbul, Turkey, June 4-6

Calciu M., Micheaux A. (2014) "Words fly, Documents rest. RESTfull Decision Supportive Documents for Marketing Managers. Towards a paradigmatic change in the interaction between managers and marketing scientists", *43rd EMAC Conference*, Valencia, Spain, June 4-6

Checkland, P., Holwell S. (2005), *Information, systems, and information systems: making sense of the field*, Wiley, Chichester, UK, 1998/2005.

Eom S.B. (1995) Decision support systems research: reference disciplines and a cumulative tradition. *Omega Int. J. Management Sci.*, 23, 511-523.

Demirkan, H. & Dursun D.(2013), Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud, *Decision Support Systems,* 55, 1, 412-421

Fader, P.S., Hardie B.G.S, Lee K.L. (2005) "Counting Your Customers the Easy Way: An Alternative to the Pareto/NBD", *Marketing Science*, Vol.24, Spring, 275-284.

Fielding, R.T. (2000), *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral dissertation, University of California, Irvine

Fielding, R.T.; Taylor, R.N. (2002), Principled Design of the Modern Web Architecture, *ACM Transactions on Internet Technology* (TOIT) (New York: Association for Computing Machinery) 2 (2)n, Mai: 115–150, doi:10.1145/514183.514185, ISSN 1533-5399

Hamdaqa, M., Tahvildari L. (2012) Cloud Computing Uncovered: A Research Landscape, *Advances in Computers*, 86, 41, 43-84, http://dx.doi.org/10.1016/B978-0-12-396535-6.00002-8

Keen, P. G. W. & Scott-Morton, M.S. (1978). *Decision support systems: an organizational perspective*. Reading, Mass., Addison-Wesley Pub. Co.

Lilien G.L & Rangaswamy A. (2000) Modeled to bits: Decision models for the digital, networked economy, *Intern. J. of Research in Marketin*g, 17, 227–235

Lilien G.L & Rangaswamy A. (2008), Marketing Engineering: Models that Connect with

Practice, in Wierenga B., Ed., *Handbook of Marketing Decision Models*, Springer, 527-560

Little, John D.C. (1979), Decision Support Systems for Marketing Managers, *Journal of Marketing*, 43, 3, 9-27.

Marston, S., Li Z., Bandyopadhyay S., Zhang J., Ghalsasi A. (2011), Cloud computing - the business perspective, *Decision Support Systems* , 51, 176-189

Pautasso, C., Zimmermann, O., Leymann, F. (2008), RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision, *17th International World Wide Web Conference* (WWW2008), Beijing, April

Richardson, L.; Ruby, S. (2007), *RESTful Web Services*, O'Reilly, ISBN 978-0-596-52926-0

Sol, H.G, Takkenberg, C. A.Th. & de Vries Robbé, P. F., Eds (1987). *Expert systems and artificial intelligence in decision support systems*: proceedings of the Second Mini Euroconference, Lunteren, The Netherlands, 17–20 November 1985. Springer

Sprague, R. and E. Carlson (1982) *Building effective decision support systems*. Prentice-Hall, Englewood Cliffs, NJ.

Spohrer, J., Demirkan H. (2011)*Understanding Service Innovation: A Closer Look at the Minority Reports*, Arizona State Universy & IBM Corp, working paper

Spohrer, J., Maglio P.P., Bailey J. & Gruhl D. (2007) , Steps toward a science of service systems, *IEEE Computer* , 40,1, 71–78.

Vinoski, S. (2002), Putting the "Web" into Web services: Interaction models, part 2. *IEEE Internet Computing*, 6,4,90–92.

Watts, S. (2017) *SaaS vs PaaS vs IaaS: What's The Difference and How To Choose*, September 22, https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/, last visited on 29/09/2018

Wierenga, B., van Bruggen, G.H., (1997), The integration of marketing problem solving modes and marketing management support systems. *Journal of Marketing,* 61, 21–37, July.

Wierenga, B., van Bruggen, G.H., Althuizen N.A.P. (2008), Advances in Marketing Management Support Systems, in Wierenga B., Ed., *Handbook of Marketing Decision Models*, Springer, 561-592

Wittig, A. &Wittig, M. (2016), *Amazon Web Services in Action.* Manning Press. p. 93. ISBN 978-1-61729-288-0

Appendix

Listing 2- Creating the cluster infrastructure with one master and seven workers on the cloud with OpenStack

```
openstack server create --flavor $OS_FLAVOR --image $OS_IMAGE --nic
netid=$ OS_NETID --key-name $OS_KEYNAME --user-data init-spark-master.sh
sparkmaster # Create Spark master VM
for i in {1..7}
do
openstack server create --flavor $OS_FLAVOR --image $OS_IMAGE --nic
netid=$ OS_NETID --key-name $OS_KEYNAME --user-data init-spark-worker.sh
sparkworker$ i # Create first Spark workers
done
```

Listing 3- Executing a Big Data calculation on a Kubernetes cluster with 3 executors

```
spark-submit --master k8s://https://192.168.99.101:8443 \
--deploy-mode cluster \
--name spark-pi \
--class org.apache.spark.examples.SparkPi \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=spark \
local:///opt/spark/examples/jars/spark-examples_2.11-2.3.0.jar
```