Cloud Architecture for Dynamic Service Composition

Jiehan Zhou[1,2], Kumaripaba Athukorala[1], Ekaterina Gilman[1], Jukka Riekki[1], Mika Ylianttila[1]*
[1]University of Oulu, PL 4500, Oulu, Finland
[2] University of Toronto, Canada
Email:{firstname.secondname@ee.oulu.fi}

ABSTRACT

Service composition provides value-adding services through composing basic Web services, which may be provided by various organizations. Cloud computing presents an efficient managerial, on-demand, and scalable way to integrate computational resources (hardware, platform, and software). However, existing Cloud architecture lacks the layer of middleware to enable dynamic service composition. To enable and accelerate on-demand service composition, we explore the paradigm of dynamic service composition in the Cloud for Pervasive Service Computing environments, and propose a Cloud-based Middleware for Dynamic Service Composition (CM4SC). In this approach, we introduce the CM4SC 'Composition as a Service' middleware layer into conventional Cloud architecture, to allow automatic composition planning, service discovery, and service composition. We implement the CM4SC middleware prototype utilizing Windows Azure Cloud platform. The prototype demonstrates the feasibility of CM4SC for accelerating dynamic service composition, and convinces us that the CM4SC middleware-accelerated Cloud architecture offers a novel way for realizing dynamic service composition.

Keywords: cloud computing, service computing, service composition, composition as a service, pervasive service computing

INTRODUCTION

Pervasive Service Computing is regarded as a Web services-based solution that realizes the pervasive computing paradigm (Zhou et al. 2009). Pervasive Service Computing emphasizes providing users with services when and where desired in everyday environments. In other words, the system provides users with on-demand services over a network (e.g. the Internet). Service composition leverages a Service Oriented Architecture (SOA) approach, and provides a contemporary technology for developing complex applications from existing service components. In the pervasive domain, a solution is needed to increase service availability, to provide on-demand service composition, and to enable adaptation to constantly changing situations. In short, dynamic service composition is needed.

A number of research efforts have focused on service composition and integration in terms of Web services (Cabrera & Kurt, 2005; Varia et al., 2010; Vaquero et al., 2009), orchestration and choreography (Erl, 2005), etc. However, the conventional service composition for service orchestration and choreography is realized within one individual

*corresponding author, email:jiehan.zhou@ee.oulu.fi

organization, or a limited number of organizations. As a result, service composition is limited to these organizations' services, instead of Internet-scale service composition and adaptive service provisioning, which is essential for the Pervasive Service Computing paradigm. In addition, communication between services in a service composition is usually realized as a simple client request/server response pattern, which requires the services involved to be always online. This kind of service composition is so tight-coupled that it does not match with loosely coupled scenarios in pervasive computing environments.

Cloud computing describes a new supply, consumption, and delivery model for IT services based on the Internet. This model treats computing service like a public utility, such as electric power supply (Miller, 2008; Hung et al., 2011). Vaquero et al. (2009) provide an encompassing definition of Cloud computing, describing its main features as follows:

(1) Large scale. Cloud usually consists of thousands of servers, which provide users with supercomputing power.
(2) Virtualization. Cloud computing provides the ability to virtualize computing resources into the Cloud.
(3) High reliability. Cloud uses multiple data copies and redundant computing facilities to ensure reliable service and security against data loss.
(4) Versatility. Cloud computing can support various application developments through combining computing resources.
(5) On-demand services. Cloud contains huge computing resources, which can satisfy various user needs. Users can use Cloud computing on a pay-per-use basis.
(6) Low cost. Cloud computing reduces the costs associated with outsourcing hardware and software resources.

Varia et al. (2010) claims that Cloud computing can solve the following five key difficulties in large-scale data processing. The first difficulty is to get as many machines as an application needs. The second is to get the machines when one needs them. The third is to distribute a large-scale job on different machines, and provide another machine to recover if one machine fails. The fourth is to auto-scale up and down, based on dynamic workloads. The last difficulty is to get rid of all these machines when the job is done.

Traditional service composition faces limits regarding all the above five difficulties. In traditional Web service orchestration, it is difficult to get on-demand service composition within the boundary of one organization, unless the organization can provide all necessary services for composition. Consequently, it probably cannot afford dynamic service composition in Pervasive Service Computing. By taking advantages of Cloud computing, we are motivated to explore new opportunities for realizing dynamic service composition for Pervasive Service Computing. Only a few research efforts have recently been reported on service composition in the Cloud. This paper explores dynamic service composition in the Cloud, and aims to provide users with a Cloud-based Middleware for Dynamic Service Composition (or, CM4SC) for supporting on-demand service composition over the Cloud.

Our main contribution is the CM4SC 'Composition as a Service' middleware layer, between the application layer and the platform layer in the conventional Cloud architecture, for accelerating dynamic service composition in the Cloud. The prototype implementation demonstrates the feasibility of CM4SC to accelerate dynamic service composition, and is suggested as a novel solution to realize the paradigm of Pervasive Service Computing, especially in accommodating support for on-demand dynamic service composition.

The rest of the paper is organized as follows: next sections define the terminology used in the paper, in the context of dynamic service composition in the Cloud and overview related work in this area. Then the paper proposes Cloud architecture for dynamic service composition, and specifies the CM4SC middleware layer and describes the implementation of a CM4SC prototype for a proof of concept with Windows Azure Cloud platform. Finally, the paper discusses lessons learned from the implementation, and draws conclusions.

TERMINOLOGY

Service-oriented computing is based on the idea of composing applications by discovering and invoking available network services, rather than by building new applications or invoking available applications to accomplish tasks (Papazoglou et al., 2007). Web services are modular, self-describing, self-contained applications that are accessible over the Internet (Dustdar & Papazoglou, 2008). The traditional Web service model consists of interaction between the service provider, the service registry, and the service consumer.

Service composition can be static or dynamic. In static composition, the process model is built manually, and the Web service is automatically selected by the system. In dynamic composition, however, both the process model and service selection are done automatically by programs.

Clouds are large pools of easily usable and accessible virtualized resources (hardware, development platforms, and services). These resources can be dynamically reconfigured to adjust to a variable load (scalability), and to allow optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model (Vaquero et al., 2009).

We define service composition middleware as sets of software components providing a service composition environment, in which users can describe task requirements, make composition plans, discover services, and generate, execute, and monitor service compositions. 'Composition as a Service' middleware in the Cloud is a functional layer for service composition. Hence, systems and applications based on this middleware can use Cloud resources and services to enable on-demand dynamic service compositions.

Cloud service refers to hardware and software accessed via Internet, and hosted by data centers. Users can utilize Cloud services on a pay-per-use basis.

CLOUD COMPUTING BACKGROUND

Benefits and Architecture

Cloud computing shares certain aspects of the conventional technologies of grid and utility computing (Zhang et al., 2010; Li et al., 2011). Grid computing coordinates networked resources to achieve a common computation-intensive objective. Beyond this, utility computing provides on-demand computing, and allows charging customers based on usage. Cloud computing realizes utility computing by leveraging virtualization technologies at multiple levels (hardware and application platforms). And Cloud computing provides the capability for pooling computing resources from clusters of servers, and for dynamically assigning virtual resources to applications on demand. Table 1 lists business benefits of Cloud computing.

The generally acknowledged Cloud computing architecture consists of four layers (Zhang et al., 2010). First, the hardware layer is responsible for managing the physical resources. Second, the infrastructure layer (also known as the virtualization layer) creates a pool of storage and computing resources, by partitioning the physical resources using virtualization technologies such as Xen (Hagen, 2008), KVM (Adelstein & Lubanovic2007), or VMWare (Troy & Helmke, 2009). Third, the platform layer consists of operating systems and application frameworks, whose purpose is to minimize the burden of deploying applications directly into virtual machine containers. And fourth, the application layer consists of the actual Cloud applications, which can leverage the automatic-scaling feature to achieve better performance, availability, and lower operating cost.

Applications can be developed efficiently by composing them from existing services. To address this specific purpose in the Cloud, a middleware layer is required for service composition. In this paper, we apply "Composition as a Service" and introduce a CM4SC 'composition as a service' middleware layer for existing Cloud architecture. The CM4SC middleware layer sits between the application layer and the platform layer, to create and manage a pool of service resources for accelerating service composition.

Table 1. Business Benefits of Cloud Computing (Varia, 2008)

Benefits	Description
Almost zero upfront infrastructure investment	With utility-style computing, there is no fixed startup cost.
Just-in-time infrastructure	By deploying applications in the Cloud with dynamic capacity management, software architects do not have to worry about pre-procuring capacity for large-scale systems.
More efficient resource utilization	With Cloud architecture, users can manage resources more effectively and efficiently by having applications which request and relinquish resources only as needed.
Usage-based costing	Utility-style pricing allows billing the customer only for the infrastructure that has been used.
Potential for shrinking the processing time	If a job requires 500 hours on a single computer, Cloud architecture makes it possible to spawn and launch 500 instances, and to process the same job in one hour.

Cloud Service Types

Tai (2009) proposes Cloud Service Engineering as a discipline that combines business and technology. Three new aspects that Cloud Service Engineering in particular addresses are the following: (1) Everything is a service. (2) Services have costs and value. (3) Services constitute value networks. Lenk et al. (2009) state that the Cloud is the "grass-root" evolution of a wide range of technologies from successful Web 2.0 enterprises. They propose an integrated Cloud computing stack architecture that classifies Cloud technologies and services into different computing services: infrastructure as a service, platform as a service, software as a service, and human as a service.

Middleware is usually regarded as a set of services that glues application software and operating systems together, in order to reduce complexity in distributed applications. We cite 'Middleware as a Service' in the Cloud, to connect the platform layer and the application layer for accelerated application development. 'Composition as a Service' is an example of 'Middleware as a Service' in the Cloud, which allows resources for composition to be managed separately from their users and providers. In this paper, our proposed CM4SC 'Composition as a Service' middleware provides a set of Cloud services that allow user task description, composition planning, service discovery, and service composition for applications development.

Commercial Cloud Products and Frameworks

The Amazon Web Services (Murty, 2008) are a set of Cloud services that together make up a Cloud computing platform, offered over the Internet by Amazon.com. The most well-known services are Amazon Elastic Compute Cloud (EC2) and Amazon S3.

Google App Engine (Sanderson, 2008) is a platform for traditional Web applications in Google-managed data centers. Currently, the supported programming languages are Python and Java. Web frameworks that run on the Google App Engine include Django, CherryPy, Pylons, and web2py, as well as a custom Google Web application framework similar to JSP and ASP.NET.

Microsoft's Windows Azure platform (Mackenzie, 2011) consists of three components: Windows Azure, Microsoft SQL Azure, and AppFabric. Each of these provides a specific set of services to the user. Windows Azure is a Cloud service operating system that serves as the development, service hosting, and service management environment for the Windows Azure platform. Microsoft SQL Azure is the secure relational database based on familiar SQL server

technologies. AppFabric helps developers to connect applications and services in the Cloud. In this paper, we use the Windows Azure Cloud platform to implement the CM4SC middleware prototype.

RELATED WORK

Service Composition and Process Planning

Requirements for designing service composition include interoperability, discoverability, adaptability, context awareness, QoS management, security, spontaneous management, and autonomous management. Six different issues in service composition are identified by Dustdar et al. (2005). These are Web service coordination and transaction, context awareness, conversation modeling, execution monitoring, and infrastructure. Two directions are recognized by Chakraborty et al. (2005). One is to define languages in order to formally specify simple and composite services in terms of service input/output, service pre-conditions and post-conditions, fault handling, joint exception handling, and service invocation mechanisms. Another direction is to develop architecture that enables service composition. Ibrahim et al. (2009) define such service composition as a sequence of four steps: translation, generation, evaluation, and finally execution.

Kalasapur et al. (2007) present overall system architecture in which (1) resources are exported as services, (2) services are registered to the directory, (3) users query the directory for services, (4) the directory performs lookup/composition on registered services, and (5) the directory returns results to users. Fuji et al. (2009) propose a dynamic service composition framework which allows users to request applications using natural language, autonomously composing new applications based on requested service semantics. The framework is depicted as follows: A request analyzer translates users' requests into an internal system language, using a graph-based approach. A semantic analyzer and a service composer then produce the composition workflow, ready to be executed by the service performer. The workflow respects the semantic matching composition rules, and the correctness is guaranteed via an evaluator module.

Medjahed et al. (2003) propose architecture for automatic composition of Web services. This approach consists of four conceptually separate phases: specification, matchmaking, selection, and generation. Rao et al. (2005) propose a general process of automatic service composition which includes five phases: (1) single service presentation; (2) language translation; (3) composition process generation; (4) composite service evaluation, and (5) composite service execution.

There are many methods used for generating processes. Medjahed et al. (2003) develop a matchmaking algorithm to generate corresponding composition plans. McIlraith et al. (2002) adapt and extend the situation calculus-based Golog language for automatic generation of service composition. They conceive service composition as a planning and execution task. The input and output parameters of the service act as knowledge preconditions and effects in a planning context. Service composition must choose between several services, each sharing some of the same effects. Users' activities can be viewed as customizations of reusable, high-level generic procedures. And a composite service is a set of atomic services connected by procedural programming language constructs (if-then-else, while, etc.).

The above solutions more or less contain the functionality needed for performing service composition. However, they are not targeting the vision of "Composition as a Service" within Cloud computing environments. In the absence of operational tools provided by Cloud computing, their users and providers are exposed to significant operational costs and risks in managing service composition. In this paper, we encapsulate the basic functionalities of service composition into CM4SC middleware, and deploy CM4SC 'Composition as a Service' to the Cloud computing environment. The novel CM4SC approach leverages Cloud computing and reduces the cost and risk burdens of performing service composition.

Service Deployment in the Cloud

To address the complexity of deploying business applications in a data center, Maghraoui et al. (2006) present a model-based approach, using object-relationship models to declaratively describe the required deployment solution, and the capabilities of provisioning environments. A planning algorithm is developed to infer provisioning orders. The authors focus on dynamic resource assignment for a given application in the infrastructure layer.

To make service deployment fast, efficient, and error-free, Arnold et al. (2007) present a pattern-based service deployment method. Based on examining the best practices and patterns, the pattern-based method is designed to capture the essential outline and requirements of a deployment solution, without specific resource bindings. The authors focus on minimizing the burden of service deployment in the platform layer.

To ensure efficient handling of changes in IT systems, Keller et al. (2004) present the CHAMPS system for change management with planning and scheduling. The established method automates the construction of change plans using a task graph builder and a planner. A task graph consists of tasks and precedence constraints that link these tasks together. A change plan is documented by the workflow language (BPEL4WS).

In contrast, our approach emphasizes Cloud resource management between the platform layer and the application layer, by presenting the CM4SC middleware layer to accelerate dynamic service composition. In our prototype implementation, we describe user tasks in XML, and generate process plans in the format of Extensible Application Markup Language (XAML). Moreover, we focus on verifying the CM4SC feasibility, and leave the details of task representation and process planning for future studies.

Service Composition in the Cloud

Few works have recently been reported on service composition in the Cloud. La and Kim (2010) present a conceptual framework for enabling context-aware mobile Cloud services. Their framework enables context capturing, context-specific adaptation, tailoring candidate services, and running the adapted services. Menzel et al. (2010) present a Cloud-based platform that supports on-demand creation and orchestration of composed applications and services. The authors are most interested in security aspects of such a system. Zhu et al. (2010) create a flowable service model for seamless integration of services to seek maximum satisfaction of both service providers and consumers, while decreasing the delivery costs of services in open Cloud environments. The authors consider two highly important characteristics: flowability and constitutional similarity. Ma et al. (2009) present an abstract model for a federation of services, together with their semantics and quality characteristics. The authors urge that Cloud services must be described by means of ontology in order to be effectively discovered. Fu et al. (2010) propose a higher concept for service composition in the Cloud—the so called virtual services. These can connect the physical, real one and enable the composition process to succeed. The goal is to enable the planning process to go through, even though the traditional planning process might fail.

In contrast, we approach dynamic service composition in the Cloud by presenting the CM4SC middleware layer to existing Cloud architecture, for accelerating service composition. Our approach applies middleware as a service, bridging conventional service composition middleware and Cloud computing, with the concern for increasing service availability and composition automation.

CLOUD ARCHITECTURE FOR DYNAMIC SERVICE COMPOSITION

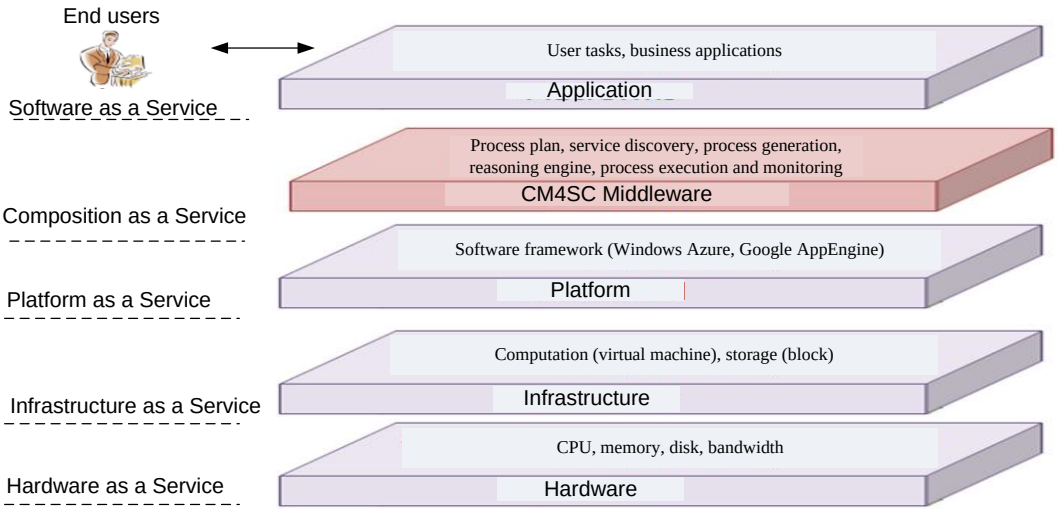
Service composition accelerates application development. We have investigated that existing Web service composition solutions are not suitable for dynamic service composition in the Cloud. They not only lack a process-layer standard for service composition, but also are built on proprietary standards and technologies. Also, existing Cloud computing platforms do not provide services to support service compositions. Obviously, the barriers between the increasing needs of on-demand service composition and the capabilities of Cloud platforms increase the burdens of cost and risk for the providers and users seeking to manage and

develop composition applications. To accelerate Web service composition in the Cloud, we apply "Composition as a Service" and encapsulate sets of services into CM4SC middleware for service composition. This section presents the CM4SC-accelerated Cloud architecture for dynamic service composition, by inserting the CM4SC 'Composition as a Service' layer. The CM4SC middleware layer connects the platform layer and the application layer in existing Cloud architecture; it consists of sets of services for dynamic service composition.

CM4SC-accelerated Cloud Architecture

To accelerate service composition and rapid application development, we extend the conventional Cloud architecture by inserting a special 'Composition as a Service' layer for dynamic service composition, namely, CM4SC-accelerated Cloud architecture (Figure 1). The CM4SC encapsulates sets of fundamental services for executing the users' service requests and performing service composition. These services, including process planning, service discovery, process generation, reasoning engine service, process execution, and monitoring, are detailed in next section. These components form the "Composition as a Service" that is hosted in the Cloud.

Figure 1. Cloud architecture for dynamic service composition



CM4SC Middleware Design

We first elaborate each basic service given in the CM4SC middleware layer (Figure 1). The CM4SC middleware contains a set of services which provide basic units needed for performing service composition. These are:

Client service, which allows users to describe complex activities and task requirements in a high-level language.

Process planner service, which transforms high-level task descriptions into formal and precise sets of activity descriptions and their associations. These activities are represented by Web services in the Cloud.

Service search engine, which takes activity descriptions as input, and carries users through to acquire the status of a corresponding service, including its availability, functionality, and interaction interface.

Process Generator service, which tries to solve user tasks by composing Cloud services advertised by service providers. The Process Generator takes the activity description and service discovery results as input, and outputs a process model that describes a composite service. The process model describes a set of selected Cloud services, the control flow, and the data flow among these services.

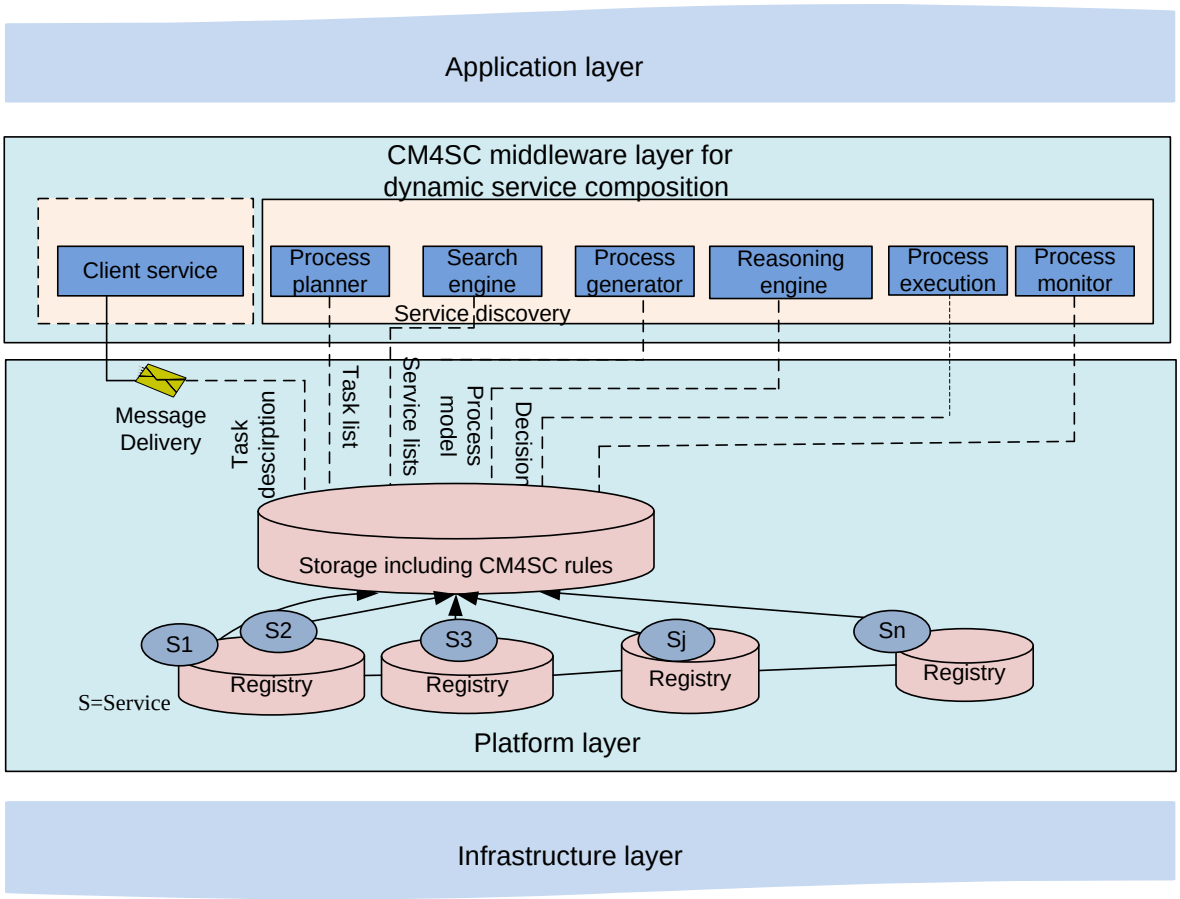
Reasoning engine service, which takes contextual information from various resources as input, processes them against the CM4SC rules in the storage, and provides decision information for process planner and generator.

Process execution service, which provides a composition execution environment.

Process monitor service, which enables users to monitor the control and data flow in a composition.

Figure 2 illustrates the CM4SC middleware design in more detail. The figure presents the functional modules in the CAM4SC middleware and layer relationships among the CM4SC middleware layer, the application layer, and the platform layer. The CM4SC middleware layer provides supporting services for the application layer to accelerate service composition. The platform layer provides storage and registration services for theCM4SC middleware to realize loose-coupled service interaction and discovery.

Figure 2. CM4SC middleware design for dynamic service composition



AZURE-BASED CM4SC MIDDLEWARE IMPLEMENTATION

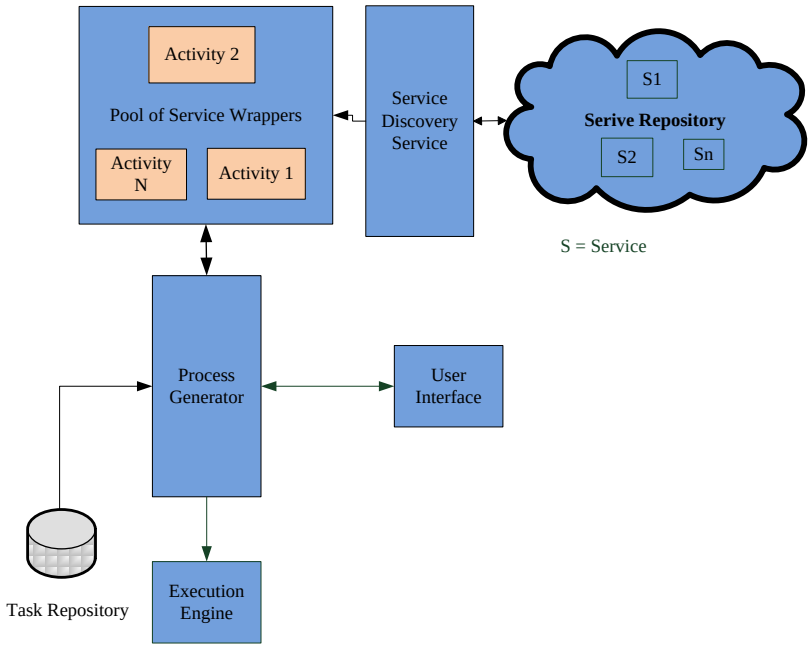
We develop the CM4SC middleware prototype utilizing Windows Azure Cloud platform, Windows Communication Foundation (WCF), and Windows Workflow Foundation (WF). The prototype aims to provide a "Composition as a Service" for accelerating dynamic service composition. This section first examines a campus scenario to illustrate the interaction between CM4SC services. The scenario is the following:

Alice is working with Professor Jade, and whenever she needs to get a document signed by Jade, she inquires about his whereabouts from a location service hosted in the Cloud. The location service gives the current location of a professor when his name is provided. However, Alice needs more detailed information, such as what to do if the professor is not available in his room. Therefore, an advisory service is required to respond to such requests.

Through using a service composition interface exposed in the Cloud, Alice succeeds in managing her tasks by composing location service and advisory service.

As can be seen from the scenario, our focus is on small atomic services, applicable in a large variety of situations. The above scenario presents the major design issues: how to dynamically compose location service and advisory service, and how to properly deal with interactions between Cloud services to adaptively fulfill user task requirements. The preliminary demo implements the CM4SC middleware layer, which provides semi-automatic service composition. In the demo, the user directs the composition planner to build a composition plan, and the service search engine provides the user with a list of composable services. The prototype allows dynamic composition, because the user can create new tasks on the fly by composing Cloud services. Figure 3 illustrates the Azure-based CM4SC middleware implementation, which consists of the following components:

Figure 3. Azure-based CM4SC middleware implementation



Service Repository. The service repository hosts WCF services. All services in service composition are considered to be WCF-based services. WCF provides the platform for developing service-oriented solutions with .NET. One advantage of utilizing WCF is that it provides features that directly support the application of service-orientation principles. The SOA-friendly qualities such as loose coupling, autonomy, statelessness, composability, discoverability, and reusability can be achieved through WCF.

Service Discovery. Another attractive feature of WCF is that it enables services to be discovered at runtime in an interoperable way (Cibraro et al., 2010). WCF services can be configured to be discoverable. Client applications can search for the services that meet a set of criteria. These criteria could include the service contract type, keywords, or scope on the network. There are two modes of discovering WCF service. The first is Ad-Hoc Mode, in which all discovery messages and client requests are sent in a multicast fashion. By default, .NET framework provides support for Ad-Hoc discovery over the UDP protocol. A client sends a probe message for service discovery using a multicast protocol. If a service finds itself matching the criteria of the probe request, it responds back to the client with the ProbeMatch message. The second mode of discovery is Managed Mode, in which there is a centralized server called a discovery proxy. The discovery proxy maintains information about the available services. Services make announcements to the discovery proxy. The discovery proxy populates its information based on these service announcements. The service

announcement mechanism and the way the discovery proxy populates itself with information depend on the approach followed by the developer. In this implementation, we utilize the Ad-Hoc discovery mode because it is simple and easy to use. However, Ad-Hoc discovery mode is not the most efficient method for service discovery, because it can increase the amount of network traffic.

Process Generator. We utilize the Windows Workflow Foundation (WF) to implement tasks based on the discovered services. WF provides a platform specifically for workflow technologies. A workflow can be considered as a set of steps required to get a job done. A common characteristic of workflow is the execution of a flow of individual steps that lead to some desired result. These individual steps in a workflow can be referred to as activities. An activity, in the context of WF, is defined as a unit of work in a business process. Therefore, if we use activities to access services, and combine these activities to build a workflow, then we can achieve service composition. WF also includes a tracking service, which allows users to keep track of the workflow execution. This is one other advantage in using WF to compose services.

There are several different approaches for representing services and service composition. In some research, a translation mechanism is provided to convert the available service technologies and service descriptions into one model (Ibrahim et al., 2009). Another approach would be providing a wrapper to create a uniform access interface to services (Constantinescu et al., 2004). We used the latter approach in representing the services. This wrapper is an activity option provided in WF. In this demo, an activity is designed to perform four basic tasks. These are: 1) declaring input of a service, 2) declaring output of a service, 3) using service discovery to locate the service, and 4) invoking a service.

Initially each task is composed of an atomic service, but it evolves with service composition to more complex composite tasks. Each task is designed as a workflow. In WF, workflows can be created using code as well as Extensible Application Markup Language (XAML). XAML is a declarative XML-based language created by Microsoft (Likness, 2011). In this demo, we create each task using XAML. In creating composite tasks we combine XAML descriptions of each task to create a new composite task. Figure 4 illustrates an atomic service, wrapped up in an activity to create a task. Here, the atomic service is called ProfessorLocation. This service is wrapped up in an activity which is also named as ProfessorLocation. In this task, we have a sequence with a single activity. The task is also given the same name as the name of the service, according to the naming convention of this prototype. Figure 5 gives the XAML description of a task.

Figure 4. A task containing an atomic service

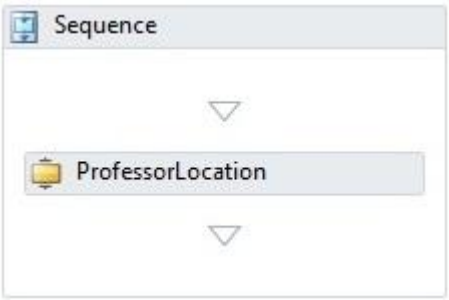


Figure 5. XAML description of a task containing an atomic service

```
<Activity x:Class="PlannerService.Task1"
xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities"
xmlns:a="clr-namespace:Activity1;assembly=Activity1"
xmlns:sad="clr-
namespace:System.Activities.Debugger;assembly=System.Activities"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <x:Members>
```

```
      <x:Property Name="ProfName" Type="InArgument(x:String)" />
      <x:Property Name="ProfLocation" Type="OutArgument(x:String)"
/>
    </x:Members>
    <Sequence sad:XamlDebuggerXmlReader.FileName="Task1.xaml">
      <a:Activity1 Location="[ProfLocation]" ProfName="[ProfName]"
/>
    </Sequence>
  </Activity>
```

Execution Engine. Windows Workflow Foundation provides a workflow execution engine. We use the execution engine for the execution of workflows. The first step in the execution is to load the XAML file.

User Interface. This is the main client interface provided for users to interact with CM4SC. The user interface is implemented as a Web role. Figure 6 illustrates the main user interface. Here, the user is presented initially with a list of all the tasks and their details. These tasks are stored in the task repository. The planner maintains an XML document containing details about each task. This XML document mainly contains the input, output parameters of the task, and a description of the task. Each task is represented by an atomic service. Users can either execute atomic tasks, or compose the tasks to create a new composite task. According to the scenario, the secretary, Alice, will execute the atomic task "ProfessorLocation" by providing the Professor's name as the input. However, Alice could also compose the "ProfessorLocation" task with the "Advice" task, to create one composite task dynamically. The system guarantees that Alice's composition plan contains only composable services. This is done by making a call to the Process Generator service whenever Alice selects a service from the first list. The Process Generator finds the services that can be composed with the "ProfessorLocation" service by comparing the output of "ProfessorLocation" service with the input of the rest of the services available in the repository. The second service list is then populated with services that are matching with "ProfessorLocation" service. In this way, the system guarantees that Alice creates only valid composition plans. She can then assign a name to the newly created composite task, and save it in the task list. The user can dynamically build more tasks by combining either atomic or composite tasks.

Figure 6. Screen shot of the user interface

DISCUSSION AND CONCLUSIONS

This paper is written with a motive to research and develop CM4SC-accelerated Cloud architecture for dynamic service composition in the Cloud. We recognize service composition in the Cloud to be a relatively novel research topic. Through reviewing existing works and studying the Azure platform, we apply "Composition as a Service" into Cloud computing, and propose the CM4SC middleware layer in the Cloud architecture for supporting dynamic

service composition. Further, we implement the CM4SC prototype. The implementation makes us realize that the basic components needed to address service composition in the Cloud are similar to the components in conventional service composition. However, CM4SC middleware as a service releases the burden of costs and risks for users and providers in using and managing those components. Meanwhile, CM4SC gives the user more chances to get computational services and provide on-demand dynamic service composition. And Cloud environments also leverage CM4SC services to automatically scale workloads. In summary, CM4SC accelerates rapid dynamic service composition and increases the availability and scalability of service composition in ways that are hard to guarantee through conventional composition approaches. The trial implementation also demonstrates that existing Cloud computing environments offer available tools to implement CM4SC middleware as a service for dynamic service composition. This work points out a novel way to realize the paradigm of Pervasive Service Computing.

Technically, we have also learned how to combine WCF and WF to compose services. We utilized WCF to implement Cloud services, then utilized WF to describe the internal flow, states, and transitions of a task. Even though WF 4.0 provides core activities that facilitate flow-control options, we had to implement custom activities to integrate real world services and service discoverability into WF.

In the future, we are interested in user-oriented task description and context-awareness. In user-oriented task description, the idea is to develop a high level language for non-technical people, for naturally describing ordinary daily activities and tasks, including complex workflow structure definitions for supporting autonomous workflow generation. We aim to support daily user activities in a pervasive computing environment. That means that services should be dynamically provided at right time, right place, and right situation. This raises questions of context-awareness. How can the users describe the contextual dependencies to meet their task requirements? Naturally, users cannot exhaustively perceive all context dependencies relevant to the occurring activities. The system should be intelligent enough to sense, capture, and reason contexts surrounding the user, and the service composition should be able to adapt to surrounding contexts. An initial context-aware service composition architecture has been studied in our previous work [41].

ACKNOWLEDGMENT

This work was carried out in the Pervasive Service Computing project, funded in the Ubiquitous Computing and Diversity of Communication (MOTIVE) program by the Academy of Finland. The first author would like to thank Prof. Hans Arno Jacobsen from Electrical and Computer Engineering Department, University of Toronto for hosting him for carrying on this study.

REFERENCES

Adelstein Tom & Lubanovic Bill (2007). *Linux System Administration*. O'Reilly Media, Inc.

Arnold, W., Eilam, T., Kalantar, M., Konstantinou, A. V., & Totok, A. A. (2007). *Pattern based SOA deployment*. Paper presented at the Proceedings of the 5th International Conference on Service-Oriented Computing, Vienna, Austria. 1-12.

Cabrera Luis Felipe & Kurt Chris (2005). *Web Services Architecture and Its Specifications: Essentials for Understanding WS-**. Microsoft Press.

Chakraborty, D., Joshi, A., Finin, T., & Yesha, Y. (2005). Service composition for mobile environments. *Mob.Netw.Appl.*, 10(4), 435-451.

Cibraro Pablo, Claeys Kurt, Cozzolino Fabio & Grabner Johann (2010). *Professional WCF 4: Windows Communication Foundation with .NET 4*. Wrox.

Constantinescu Ion, Faltings Boi, & Binder Walter. (2004). *Large scale, type-compatible service composition*. Paper presented at the Proceedings of the IEEE International Conference on Web Services, ICWS '04 (Washington, DC, USA)

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. *International Journal of Web and Grid Services*, 1(1), 1-30.

Dustdar Schahram & Papazoglou Mike P. (2008). Services and service composition - an introduction. *It - Information Technology*, 50(2), 86-92.

Erl Thomas (2005) *Service-oriented architecture (SOA): Concepts, technology, and design*. Prentice Hall.

Fu, J., Tu, H. W., Biao, M., Ma, Baldwin, J., & Bastani, F. B. (2010). *Virtual services in cloud computing*. Paper presented at the Proceedings of the 6th World Congress on Services (SERVICES-1), 467-472.

Fujii, K., & Suda, T. (2009). Semantics-based context-aware dynamic service composition. *ACM Trans.Auton.Adapt.Syst.*, 4(2), 1-31.

Hagen William von (2008). *Professional XenVirtualization*.Wrox.

Hung Shih-Hao, Shieh Jeng-Peng & Lee Chen-Pang (2011) Migrating Android Applications to the Cloud, *International Journal of Grid and High Performance Computing (IJGHPC)* 3(2):14-28.

Ibrahim, N., Le Mouel, F., & Frenot, S. (2009). *MySIM: A spontaneous service integration middleware for pervasive environments*. Paper presented at the Proceedings of ICPS '09 Proceedings of the 2009 International Conference on Pervasive Services, London, United Kingdom. 1-10.

Ibrahim Noha & Mouël Frédéric Le. (2009). A survey on service composition middleware in pervasive environments. *International Journal of Computer Science Issues, Special Issue on Pervasive Computing Systems and Technologies*, 1(1): 1-12.

Kalasapur, S., Kumar, M., & Shirazi, B. A. (2007). Dynamic service composition in pervasive computing. *Parallel and Distributed Systems, IEEE Transactions on*, 18(7), 907-918.

Keller, A., Hellerstein, J. L., Wolf, J. L., Wu, K. & Krishnan, V. (2004). *The CHAMPS system: Change management with planning and scheduling*. Paper presented at the Proceedings of Network Operations and Management Symposium, IEEE/IFIP, Vol.1, 395-408.

La, H. J. & Kim, S. D. (2010). *A conceptual framework for provisioning context-aware mobile cloud services*. Paper presented at the Proceedings of IEEE 3rd International Conference on Cloud Computing, 466-473.

Lenk, A., Klems, M., Nimis, J., Tai, S., & Sandholm, T. (2009). *What's inside the cloud? an architectural map of the cloud landscape*. Paper presented at the Proceedings of CLOUD '09: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 23-31.

Li Wing-Ning, Baran Jonathan & Schweiger Tom (2011) A Grid and Cloud Based System for Data Grouping Computation and Online Service. *International Journal of Grid and High Performance Computing (IJGHPC)* 3(4):39-52.

Likness Jeremy (2011). *Designing Silverlight Business Applications: Best Practices for Using Silverlight Effectively in the Enterprise*. Addison-Wesley Professional

Ma, H., Schewe, K. & Wang, Q. (2009). *An abstract model for service provision, search and composition*. Paper presented at the Proceedings of IEEE Asia-Pacific Conference on Services Computing (APSCC 2009), 95-102.

Mackenzie Neil (2011). *Microsoft Windows Azure Development Cookbook*. Packt Publishing.

Maghraoui, K. E., Meghranjani, A., Eilam, T., & Konstantinou, E. V. (2006). *Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools*. Paper presented at the Proceedings of Middleware 2006, ACM/IFIP/USENIX.

Mcilraith,S.(2002). *Adapting Golog for composition of semantic web Services*. Paper presented at the Proceedings of the 8th Int'l Conf. on the Principles of Knowledge Representation and Reasoning. Toulouse: Morgan Kaufmann Publishers, pp. 482-493

Medjahed, B., Bouguettaya, A., & Elmagarmid, A. K. (2003). Composing web services on the semantic web. *The VLDB Journal*, 12(4), 333-351.

Menzel, M., Warschofsky, R., Thomas, I., Willems, C., & Meinel, C. (2010). *The service security lab: A model-driven platform to compose and explore service security in the*

cloud. Paper presented at the Proceedings of 6th World Congress on Services (SERVICES-1), 115-122.

Miller Michael (2008). *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Que.

Murty James (2008). *Programming Amazon Web Services - S3, EC2, SQS, FPS, and SimpleDB*. O'Reilly Media.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, 40(11), 38-45.

Rao, J. & Su, X. (2005). *A survey on automated web service composition methods*. Paper presented at the Proceedings of Springer LNCS: Semantic Web Services and Web Process Composition, vol. 3387, pp. 43-54.

Sanderson Dan (2008). *Programming Google App Engine*. O'Reilly Media, Inc.

Tai, S. (2009). *Cloud service engineering. Enabling Technologies: Infrastructures for Collaborative Enterprises*. Paper presented at the Proceedings of WETICE '09 18th IEEE International Workshops on, 3-4.

Troy Ryan & Helmke Matthew (2009). *VMware Cookbook*. O'Reilly Media, Inc.

Vaquero L.M., Rodero-Merino L., Caceres J.& Lindner M. (2009). A break in the clouds: towards a cloud definition. *SIGCOMM Comput.Commun.Rev.*, 39(1): 50-55

Varia J. (2008). *Cloud architectures*. Technical Report on Amazon Web services. [WWW document] http://jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf (accessed 18th May 2011)

Zhang Q., Cheng L., Boutaba R. (2010) Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1), 7-18.

Zhou J., Riekki J. & Sun J. (2009) *Pervasive service computing toward accommodating service collaboration*. Paper presented at the IEEE Proceedings of Pervasive Service Computing and Application 2009 in Conjunction with FCST 2009, Shanghai, China.

Zhu, Y., Shtykh, R. Y., & Jin, Q. (2010). *Provision of flowable services in cloud computing environments*. Paper presented at the 5th International Conference on Future Information Technology (FutureTech),1-6.