ConPaaS: a Platform for Hosting Elastic Cloud Applications

Guillaume Pierre Corina Stratan

Vrije Universiteit Amsterdam

CLOUD computing opens new perspectives for hosting applications. From an application developer's point of view, migrating an application to the cloud may be as simple as replacing one physical machine from a traditional IT system with an equivalent virtual machine in the cloud. However, the cloud provides many more opportunities for building sophisticated applications. For example, executing an embarrassingly parallel application incurs the same costs whether one uses one virtual machine for N hours or N virtual machines for one hour. The availability of "spot instances" with variable prices allows one to further reduce the execution cost of non-urgent tasks. For online applications, the almost-unlimited number of available resources allows one to continuously adjust the quantity of provisioned computing resources to the current workload of the application.

Making full use of the cloud for hosting complex applications is however not an easy task. Most clouds today offer "Infrastructure-as-a-Service" (IaaS) functionality, which means they rent out basic resources such as virtual machines, disks and networks. Application developers must therefore handle the complexity of deploying applications composed of many inter-related components, implementing automatic resource provisioning, orchestrating application reconfigurations such that users do not notice any downtime, developing fault-tolerance mechanisms, etc. Tailoring such mechanisms for each application is tedious and error-prone, as subtle implementation bugs may appear only in production.

To address these concerns, the Contrail European project is designing and developing ConPaaS, an open-source runtime environment for hosting applications in the cloud. ConPaaS belongs to the Platform-as-a-Service (PaaS) family, which refers to a variety of systems aiming at offering the full power of the cloud to application developers while shielding them from the associated complexity of the cloud. ConPaaS is designed to host both high-performance scientific applications and online Web applications. It automates the entire life-cycle of an application, including collaborative development, deployment, performance monitoring, and automatic scaling. Finally, it runs on a variety of public and private clouds, and is easily extensible. This allows developers to focus their attention on application-specific concerns rather than on cloud-specific details.

Application model

ConPaaS aims at supporting familiar programming models so that existing applications can easily be ported to the cloud. It provides a collection of *services*, where each service acts as a replacement for a commonly used runtime environment. For example, to replace a MySQL database, ConPaaS provides a cloud-based MySQL service which acts as a high-level database abstraction. The service uses real MySQL databases internally, and therefore makes it easy to port a cloud application to ConPaaS. Unlike a regular centralized database, however, it is self-managed and fully elastic: one can dynamically increase or decrease its processing capacity by requesting the service to reconfigure itself with a different number of virtual machines.

ConPaaS currently contains six services: two web application hosting services respectively specialized for hosting PHP and JSP applications; a MySQL database service; a NoSQL database service

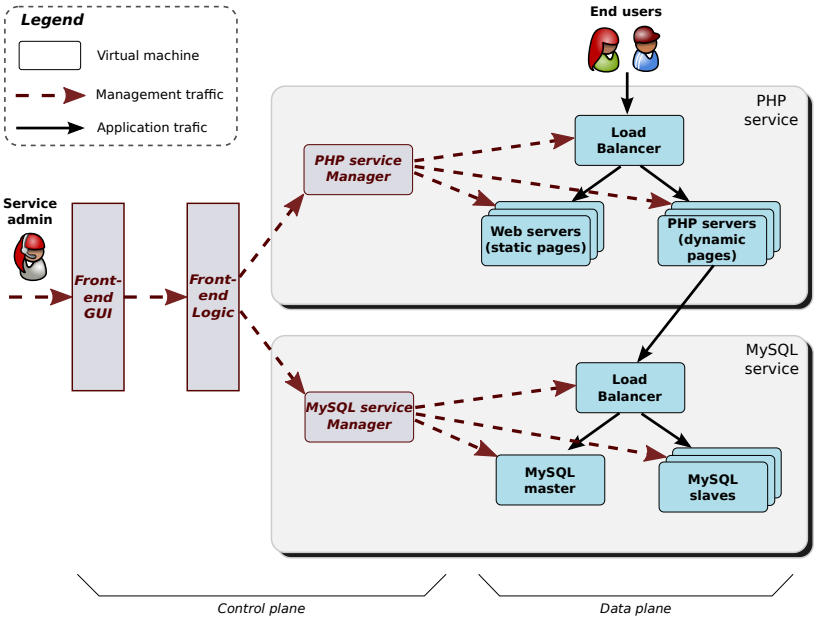

Figure 1: ConPaaS system architecture

built around the Scalarix key-value store; a MapReduce service; and a TaskFarming service for high-performance batch processing. In addition, it contains XtreemFS, a shared file system for the cloud.

ConPaaS applications can be composed of any number of services. For example, a bio-informatics application may store genomic data in XtreemFS, make use of a PHP and a MySQL service to host a Web-based frontend, and link this frontend to a MapReduce backend service for conducting high-performance genomic computations on demand.

System organization

Figure 1 shows ConPaaS hosting an application composed of two services (PHP and MySQL). This application does not need to be designed specifically for the cloud. It could be a pre-existing one, such as for example WordPress.

ConPaaS distinguishes the application traffic from the management traffic. The application traffic is generated by end users accessing the application. End user requests are handled by load balancers, web servers and database servers following a classical multi-tiered hosting architecture. The servers in charge of processing HTTP traffic belong to the PHP service, while those in charge of database queries belong to the MySQL service.

Each service is under the control of one "manager" virtual machine. This virtual machine does not run any application code but has the responsibility of carrying all administrative tasks within the service. It is in charge of starting or stopping "agent" VMs to vary the processing capacity of the service, uploading new versions of the application code to them, coordinating reconfigurations, centralizing performance monitoring information, etc. Application administrators can control their services using a graphical front-end or a command-line tool.

ConPaaS aims at being able to run over a wide variety of public and private IaaS clouds. We currently support OpenNebula and Amazon EC2, and are planning for more cloud backends in the near future. We hope that this will allow us to smoothly migrate applications from one cloud to another (for

example because of variations in the pricing of cloud resources), without the need to stop the application at any time. The fact that ConPaaS is open-source will greatly help such migrations: one can expect the exact same system to be running in the source and the destination cloud, and coordinate their actions to organize the migration such that end users notice no service interruption. Similarly, one may imagine running an application across several cloud systems on a permanent basis. This may help an application to place its points of presence close to its users, and to remain available even in the case of an entire cloud outage.

Performance control

An important goal of ConPaaS is to automate performance control. The workload processed by an online application can vary according to predictable and unpredictable patterns. Predictable patterns include workload fluctuations between day and night. On the other hand, unpredictable patterns may be created by flash crowds, for example. In such cases it is important that the system reacts quickly by dynamically adjusting capacity to address the overload situation. Conversely, the system should also deprovision resources when the load decreases in order to reduce the hosting costs.

Dynamically provisioning an application such as the one from Figure 1 is not an easy task.

For example, the application from Figure 1 is composed of virtual machines with different roles such as load balancers, static web servers, PHP servers, and database servers. Whenever load increases and it necessary to add capacity to the system, one needs to find the performance bottleneck of the application to decide which role the newly created VMs should take.

Most existing PaaS environment address this issue using rules based on real-time monitoring data – for example, adding a new VM instance to the application if the load on the current machine exceeds a certain threshold. However, such simple rules do not allow one to accurately identify the optimal reconfiguration. For instance, such a rule may reprovision the application server tier while the actual performance bottleneck was in the database layer.

Our approach consists of modeling the performance of the entire application. This is realized by letting each service of the the application monitor its own performance and model the impact that adding or removing resources would have on its own performance. These performance predictions are in turn used to compute the effect that provisioning changes would have on the entire application.

A second challenge originates in the fact that performance in the cloud often varies from one VM to the next. We plan to address these issues using techniques previously developed in our group [1]. Each time a new VM is created, it will first be benchmarked using small synthetic applications to identify its individual performance profile. This profile allows one to derive the net effect that this particular instance would have if it was used in the service. This allows us to build instance-specific performance models which indicate which role this VM should take to maximize the overall application performance. The same algorithms can also decide which VM should be released when the load decreases.

Application deployment

We want ConPaaS to be as developer-friendly as possible, and therefore try to anticipate their needs. For example, deploying a complex application composed of several services may be cumbersome, as the user would need to create and configure each required service one by one. To automate the deployment process of an entire multi-service application we are designing a "manifest" language to specify the entire structure of a ConPaaS application.

A manifest specifies the list of services which should be created, the location of code and data that need to be uploaded to each service, and all other necessary configuration informations necessary for the good execution of the application. This will allow for example to package entire multi-service applications in a single bundle that end users can easily instantiate without needing to configure anything.

The landscape of Platform-as-a-Service environments

The cloud computing market is very dynamic, and several commercial and academic Platform-as-a-Service environments are readily available. These systems all aim at simplifying application hosting in the cloud, but they have different functionalities and limitations.

Google AppEngine[a] offers support for Web applications written in Python, Java and Go. It also provides several data storage services (MySQL, NoSQL and object/file storage). The main drawback of AppEngine is its lack of flexibility: for example, AppEngine applications run exclusively in Google's cloud. Besides, developers must use AppEngine's proprietary APIs, which restricts application portability to other clouds.

Another well-known Platform-as-a-Service system is Elastic Beanstalk from Amazon Web Services[b]. Beanstalk also supports several types of applications and data storage services. It can scale the deployments automatically based on monitoring data. Like Google's AppEngine, Beanstalk is available only in the Amazon cloud.

Windows Azure[c] is Microsoft's Platform-as-a-Service environment. As one would expect, it mostly focuses on hosting Windows applications in Microsoft's data centers.

OpenShift Flex[d] and Cloud Foundry[e] are recent projects which aim at developing open-source PaaS systems with support for multiple cloud vendors. At this moment, however, they each can run only on a limited number of clouds, and do not provide support for automatic scaling.

The main features that distinguish ConPaaS from other PaaS systems are its approach to automatic application scaling and its interoperability with multiple clouds. Instead of implementing simple performance-based triggers which provision each tier individually based on its own performance, we model the performance of the entire application, taking into account the fact that virtual instances may have different roles in the application. This allows us to identify the performance bottlenecks in complex applications, and thus to minimize the number of cloud resources necessary to host the applications.

[a] `http://appengine.google.com`
[b] `http://aws.amazon.com/elasticbeanstalk/`
[c] `http://www.windowsazure.com/`
[d] `https://openshift.redhat.com/app/flex`
[e] `http://www.cloudfoundry.com`

One interesting functionality of application manifests is that certain services may require to be attached with a specific IP address. In such a case, ConPaaS will automatically create a virtual private network between machines belonging to the application, and attach specific nodes to specific IP addresses within the VPN. This is very useful to allow different services of the application to communicate with each other. For example, a database service may request to be reachable at IP address 10.0.0.99 so that other services which need access to the database know they can always access their database at this address.

Application development

Although it is easy to port pre-existing applications to ConPaaS, we also think that ConPaaS should actively support the process of developing and testing applications for the cloud. This includes for example the ability to run two different versions of the same application simultaneously (one for production traffic, the other one for testing). We also plan to natively support non-regression testing by building a new service around the Selenium system[1]. Selenium allows one to record and playback entire Web browsing scenarios, which allows developers to fully automate the testing of Web applications.

[1] http://seleniumhq.org/

Extending the ConPaaS platform

ConPaaS is also designed to facilitate the development of new services. All services derive from a single "generic service" which provides the basic functionalities for starting and stopping VMs, configuring and initializing the right services in them, etc. All the virtual machines in ConPaaS also rely on the same generic VM image. A service implementation therefore consists of a few hundred lines of Python (implementing the service-specific parts of the service manager) and Javascript (extending the front-end GUI with service-specific information and control). This simple structure allows us to plan developing extra services in the future, and also enables third-party developers to design their own custom services.

Use cases

The Contrail European project is composed of 11 universities, research centers and companies. The design and implementation of ConPaaS is mostly realized at Vrije Universiteit in Amsterdam, Zuse Institute in Berlin and XLAB in Ljubljana. Among the other partners, four are actively building real applications making use of ConPaaS, and thereby giving us immediate feedback from demanding users.

Two use cases are scientific applications, respectively dedicated to the analysis of results from Small Angle Neutron Scattering experiments (a technique to investigate the structure of various substances at a mesoscopic scale of about 1–1000 nm) and to ChIP-sequencing (a technique to analyze protein interactions with DNA). These two services make intensive use of the MapReduce and TaskFarming services from ConPaaS as their main runtime for high-performance big-data computations. Both applications also plan to offer Web-based frontends to their users.

The other two applications are complex web-based interactive applications. One is a multimedia processing marketplace which allows users to access multimedia content and process them on the fly (for example by automatically translating speech to different languages). The other one is a virtual tourist guide which gives users a 3D representation of the earth, augmented with several layers of georeferenced data.

These users report that using ConPaaS significantly reduces the time-to-market of cloud applications: developers can focus their attention on application-specific concerns rather than on making their applications suitable for the cloud. They become more productive because they no longer need to build their own VM images any more, implement application deployment mechanisms, performance monitoring, resource provisioning etc.

Conversely, the variety of requirements from these four use cases help us to better understand the needs of actual users, and also to demonstrate the ability of ConPaaS to host nontrivial applications. The development of these use cases has started recently, giving us invaluable feedback about ConPaaS.

By the time this article gets published, a stable public release of ConPaaS should be available at `www.conpaas.eu`. We also operate a public testbed so that anyone can try the system for free, without having to install anything. We particularly welcome any kind of feedback, be it positive or negative. Positive feedback is always encouraging for us; while (constructive) negative feedback may help us building a better system.

Acknowledgments

This work is partially funded by the FP7 Programme of the European Commission in the context of the Contrail project under Grant Agreement FP7-ICT-257438.

References

[1] Jiang Dejun, Guillaume Pierre, and Chi-Hung Chi. Resource provisioning of Web applications in heterogeneous clouds. In *Proceedings of the 2nd USENIX Conference on Web Application Development*, June 2011.