10-2009

IBM Altocumulus: A Cross-Cloud Middleware and Platform

E. Michael Maximilien

Ajith Harshana Ranabahu
Wright State University - Main Campus

Roy Engehausen

Laura Anderson

IBM Altocumulus: A Cross-Cloud Middleware and Platform

E. Michael Maximilien
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
+1 (408) 927-2124
maxim@us.ibm.com

Ajith Ranabahu
Wright State University
3640, Colonel Glenn Highway
Dayton, OH 45435
+1 (706) 614-4324
ranabahu.2@wright.edu

Roy Engehausen
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
+1 (408) 927-2708
hengehau@us.ibm.com

Laura Anderson
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
+1 (408) 927-1120
lca@us.ibm.com

Abstract

Cloud computing has become the new face of computing and promises to offer virtually unlimited, cheap, readily available, "utility type" computing resources. Many vendors have entered this market with different offerings ranging from infrastructure-as-a-service such as Amazon, to fully functional platform services such as Google App Engine. However, as a result of this heterogeneity, deploying applications to a cloud and managing them needs to be done using vendor specific methods. This "lock in" is seen as a major hurdle in adopting cloud technologies to the enterprise. IBM Altocumulus, the cloud middleware platform from IBM Almaden Services Research, aims to solve this very issue of managing applications across multiple clouds. It provides a uniform, service oriented interface to deploy and manage applications in various clouds and also provides facilities to migrate instances across clouds using repeatable best practice patterns. In this demonstration we will present the latest version of the IBM Altocumulus platform and also reveal some of the latest additions on scaling and the ability to perform map-reduce type computations.

Categories and Subject Descriptors

C.5.5 [**Computer Systems Organization**]: Computer System Implementation – *servers.*

General Terms

Management, Experimentation, Standardization.

Keywords

Cloud computing, Cloud Platform, Clouds, Middleware, Web service, Web APIs, Cloud Management

1. Introduction

Cloud computing allow IT infrastructure resources and application platforms to be exposed and consumed as services. Various styles of compute cloud infrastructure are now available. Some expose purely raw computing resources, some focus on application spaces, and others focus on enabling private and public hybrids of the previous.

While the heterogeneity of cloud approaches can encourage innovation and some level of adoption, it also results in islands of uninteroperability and confusions in the marketplace. In this demonstration we will show a new kind of middleware and platform that aims to homogenize the different cloud paradigms, thereby helping solve the cloud interoperability issues.

OOPSLA 2009, October 25-29, 2009, Orlando, Florida, USA.
ACM 978-1-60558-768-4/09/10.

2. Problem

The key problem that we are trying to address with our middleware is simply: *enabling interoperability amongst and across compute clouds*. Ideally the interoperation would be for different styles of compute clouds:

1. Infrastructure-based clouds such as Amazon Elastic Compute Cloud (EC2), RackSpace, and IBM Developer Cloud.
2. Platform-based clouds such as Google App Engine and Microsoft Azure.
3. Private clouds such as IBM CloudBurst, Eucalyptus, and NC State University Virtual Compute Lab (VCL).

Additionally, interoperation should allow different types of workloads and applications to be deployed, managed, and scaled without respect of cloud types and locations. Specifically, we believe that interoperation across clouds should enable cloud-agnostic features around:

1. *Application frameworks*. These include supporting leading Web application frameworks and languages such as: Ruby on Rails, Python/Django, JavaEE, Java/Spring, and PHP/Zend, as well as frameworks for scalable batch-mode operations, e.g., Apache Hadoop and Apache Cassandra.
2. *Deployments* of different workloads from the different application frameworks into different clouds. Typical deployment workflows such as creating instances, running an image, installing various components, configuring software

components into running instances, need to be done seamlessly across clouds.

3. *Scalability* such that rules, topologies, and best practices around scaling a particular workload can be done reliably and repeated across clouds.

4. *Monitoring* providing various metrics such as CPU, disk, and memory usages across all instances of a deployment. Collected data not only helps with scalability rules but also in understanding quality of service and thus decision-making for managing the life-cycle of deployments.

In a nutshell, we want to enable cloud best practices (or cloud usage patterns) for different workloads that can be utilized in and across different clouds.

3. Altocumulus

The IBM Altocumulus platform and middleware is a research project from IBM Almaden Research Center and has been available to all IBM employees since July 2009. Release 2 of the Altocumulus project addressed most of the requirements listed in Section 2, except for scaling. The up-coming release 3 addresses scalable deployments via scalable best practices.

3.1 Architecture

The IBM Altocumulus uses a distributed architecture that divides the platform into three components:

1. *Dashboard* is the principal interaction points for end users. There, users can: a) select the best practices that they want to use, b) provide credentials for the different clouds to target, and c) deploy and manage their applications into different clouds.

2. *API and API Tester* is a Web application that exposes the Altocumulus REST API and gives an easy UI to invoke the API and browse documentation about it.

3. *Core* is the back-end orchestrator of the platform. It implements the API which is how the Dashboard (and other Altocumulus applications) can communicate with the Core. The Core contains an elaborated set of scripts, rules, and cloud adapters to allow execution of actions seamlessly across different clouds for different users.

Figure 1 illustrates the various components of the platform as well as different clouds currently fully supported and in experimental support.

3.2 Implementation

The IBM Altocumulus platform and middleware is fully implemented in Ruby and Ruby on Rails. The various Web applications (e.g., the Dashboard and API Tester) are implemented in Ruby on Rails. The API implementation makes use of Rails 2.0 RESTful API support.

The Core back-end scripts and cloud adapters are implemented in pure Ruby with no specific tie-in dependency with the Rails framework. However, where useful, various aspect of the Rails philosophy, e.g., small tools, conventions over configurations, and metaprogramming, was also used on the scripts and adapters.

The current deployment of IBM Altocumulus uses the platform itself to deploy a version into an IBM CloudBurst internal private cloud.

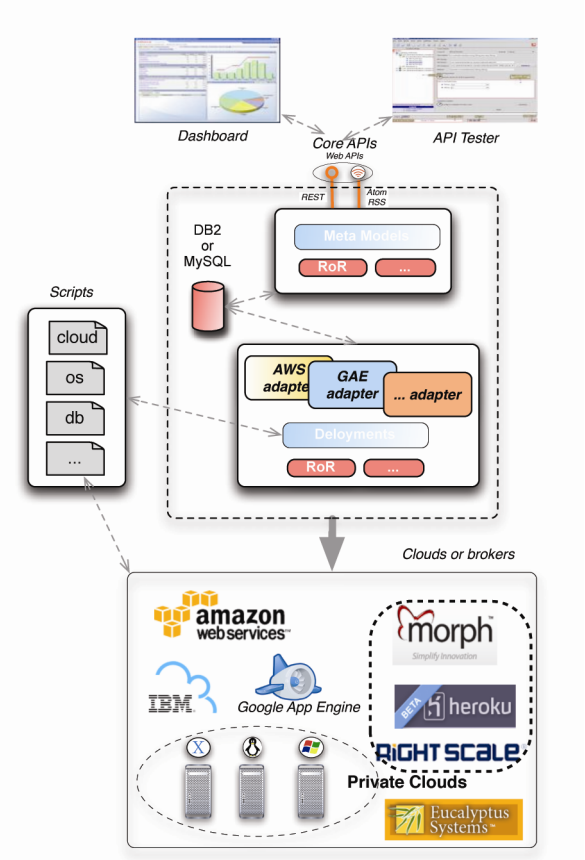

Figure 1 Architecture overview of the IBM Altocumulus cross-cloud platform and middleware

4. Discussion

The unique aspects of the IBM Altocumulus are its cross-cloud capabilities: APIs and codification of cross-cloud best practices for various Web application frameworks. The best practices are knowledge information that comes from communities of users of these frameworks and clouds and is designed to evolve as new versions of the frameworks evolve or new frameworks and cloud functionality are added.

Future works include supporting additional clouds and supporting automatic scaling between clouds. That is the ability to complement a private cloud deployment with public cloud resources to match a burst in demand and then scaling down when the demand appeases. Another interesting direction is to look into an abstract cloud application service layer which would direct application writers to create applications that are easily scalable.

5. Reference

[1] Maximilien, E. M., Ranabahu, A., Engehausen, R., and Anderson, L. Toward Cloud-Agnostic Middlewares. In *Proceedings of ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA),* Orlando, FL, Oct. 2009.