10-2009

A Best Practice Model for Cloud Middleware Systems

Ajith Harshana Ranabahu
Wright State University - Main Campus

E. Michael Maximilien

A Best Practice Model for Cloud Middleware Systems

Ajith Ranabahu[1] and E. Michael Maximilien[2]

[1] Knoesis Center,Wright state University, Dayton OH 45435, USA,
ajith@knoesis.org
[2] IBM Almaden Research Center, San Jose CA 95120, USA
maxim@us.ibm.com

Abstract. Cloud computing is the latest trend in computing where the intention is to facilitate cheap, utility type computing resources in a service-oriented manner. However, the cloud landscape is still maturing and there are heterogeneities between the clouds, ranging from the application development paradigms to their service interfaces,and scaling approaches. These differences hinder the adoption of cloud by major enterprises. We believe that a cloud middleware can solve most of these issues to allow cross-cloud inter-operation. Our proposed system is *Altocumulus*, a cloud middleware that homogenizes the clouds. In order to provide the best use of the cloud resources and make that use predictable and repeatable, Altocumulus is based on the concept of *cloud best practices*. In this paper we briefly describe the Altocumulus middleware and detail the cloud best practice model it encapsulates. We also present examples based on real world deployments as evidence to the applicability of our middleware.

1 Introduction

Cloud computing is resulting in very cheap, on-demand, seemingly unlimited compute resources. Any enterprise can easily provision, manage, and sustain (keep up to date and scale) their entire operations on a compute cloud for a fraction of the cost of owning the resources. In essence, cloud computing results in commoditization of typical computing resources which are now, via compute clouds, available as a service.

While most enterprises could make use of such cheap resources to deploy their applications, there remains various concerns to address and that could help accelerate that transition. These challenges can be summarized into three categories: 1) quality of service, 2) security and privacy, and 3) dynamic scaling to meet application customer demands.

The current state of the art in the industry is to capture or define *best practices*. Best practices are known good patterns of software configurations and setup (with appropriate updates) that are known to be reliable or that address some of the specific issues listed above. For example, a common best practice to scale a relational database to a large number of (predominantly read)

requests is to use an in-memory cache such as *memcached*[1]. By running a properly configured memcached process, database query results can be saved in the memory cache so that frequently repeated queries are not served from the database but rather fetched from the cache. Naturally, adding a memory cache alone does not solve the entire scaling issue and other steps need to be taken to truly scale the system. However, adding a memory cache is a known scaling solution for read intensive database applications and is considered a *best practice*.

Making efficient use of a cloud computing environment in order to reap the benefits while trying to address the key concerns listed above can be done using best practices. However, such best practices are typically tacit knowledge and they evolve and change over time. The ideal situation would be to try to automate best practice-based cloud deployments for the different cloud frameworks available, as well as having a means to allow such best practices to evolve and new ones to be added.

The IBM Altocumulus research project is a new kind of cloud platform and middleware that attempts to address exactly this issue. In addition to enabling deployments addressing some of the quality, security, and scaling issues that are common in Web applications development, IBM Altocumulus allows such deployments to be done in a cloud-agnostic fashion. Finally, it also enables migrations of deployments across compatible clouds.

1.1 Organization

The remainder of this paper is organized as follows. Section 2 discusses related work such as software patterns and cloud middleware. Section 3 gives an architectural overview of our middleware and explains some of the key use cases. Section 4 gives a detailed description of our notion of a *cloud best practice* along with some specific examples around several key pain points cloud users experience. We discuss where the project currently stands and initial results of our deployment of the platform inside IBM in Section 5 and conclude after describing some ongoing and future work in section 6.

2 Related Work

2.1 Middleware

Middleware, in the context of distributed computing systems, was first described by Bernstein et al. [2] as a set of intermediaries for the components in a distributed computing system. This concept has been extensively utilized during the uprising of the Service-Oriented Architecture (SOA) where the services in question were in fact provided by middleware systems. Middleware in general is used to abstract the differences between heterogeneous systems and expose a uniform interface.

2.2 Cloud Computing

Cloud computing is defined as both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [3]. The landscape for computing clouds is still evolving and there are three major paradigms of cloud services available today.

- **Infrastructure as a Service (IaaS)** : The raw computing resources are exposed to the consumers. There are options for including support software but usually there is no abstraction over the system complexities. This is in fact exactly the same as getting access to a physical computer attached to the Internet. Amazon EC2 [4] is the leading IaaS provider and the most prominent cloud service provider today.
- **Platform as a Service (PaaS)** : Consumers get access to an application development platform possibly hosted on a infrastructure cloud. The platform completely abstracts the complexities of the underlying host system. The platform also guarantees load balancing and scaling in a transparent manner to the cloud consumer. However these platforms typically are restrictive and requires the hosted applications to strictly adhere to certain specialized Application Programming Interfaces (API), frameworks, and programming languages. Google AppEngine [5] is an example of a PaaS.
- **Software as a Service (SaaS)** : Consumers get access to specialized software suites hosted in a cloud environment. Unlike the platform offering, there is no room for custom application development. However the offered software suites are generally extensively customizable. The SalesForce.com Customer Relationship Management(CRM) solution is an example where software is offered as a service.

In this work, we primarily focus on the Infrastructure and Platform service clouds (IaaS and PaaS). In the subsequent uses of the word *cloud* we refer to either a IaaS or PaaS unless otherwise stated.

2.3 Patterns

Patterns in software engineering (or *Design Patterns*) was pioneered by Beck et al. [6] and has gained popularity due to the applications in the widely used Object-Oriented Programming (OOP) languages [7]. Patterns are well accepted in the industry and deemed essential for a software engineer, primarily to avoid inefficient programming practices and stop *reinventing the wheel*. For example when constructing a user interface driven application, the best known way to engineer it is to follow the Model View Controller (MVC) pattern [7]. Following the MVC pattern not only simplifies the construction but also improves reuse and maintainability of the application. Use of patterns can also be seen in the higher layers of software engineering such as for enterprise applications [8].

Middleware for clouds has been discussed even during the earliest days of the cloud [9]. However in terms of functionality, a *cloud middleware* was expected

to either manage or configure the respective cloud itself. The work described in this paper deviates from that idea significantly by positioning the cloud middleware in the perspective of the cloud consumer. In other words the middleware and the best practice model described in this paper is aimed at homogenizing the cloud interfacing in the perspective of the cloud consumer. We believe that such a perspective change is necessary since the service interfaces for clouds are significantly different and vastly complex.

3 Middleware Architecture

3.1 Overview

The primary service that IBM Altocumulus provides to cloud users is convenient deployment of applications (primarily Web applications) to a variety of clouds. Additionally, IBM Altocumulus allows other services such as backup and restore of database content as well as cloud image creation and cloud image migration.

The Altocumulus middleware consists of three major components that interact with each other, as illustrated in figure 1.

1. **Core**: The primary functional component of the middleware. The Core manages the interactions with different cloud providers and exposes a private RESTful API.
2. **Dashboard**: The primary user interaction component. The dashboard provides a comprehensive web based user interface for human users. The dashboard also provides credential and artifact management facilities and uses the Core via the private API.
3. **Core-API**: The public API for the core. Core-API exposes a well documented RESTful API for applications and acts as a façade to the Core. Core-API facilitates the controlled exposure of the middleware functionalities for external applications.

Apart from these three components, there is a support component that manages images for various clouds called the image repository. The term *image* is used in this context to represent a serialization of the artifacts of a working system, in accordance to the use of the term Amazon Machine Image (AMI) by Amazon. The function of the image repository is to store and manage such images. While the role of the image repository is not considered central to the Altocumulus platform, it is indeed useful and acts as a support component.

All IBM Altocumulus deployments are done via the utilization of cloud best practices that attempt to provide pre-packaged software stack and related configurations for a specific type of application. For example, using IBM Altocumulus, a deployment of an IBM WebSphere sMash application on a public cloud (such as the Amazon EC2) can be done repeatably including all necessary software configurations. If the target cloud needs to be changed, the change to the deployment is minimal and users are shielded from the complexities that arise from such a difference.

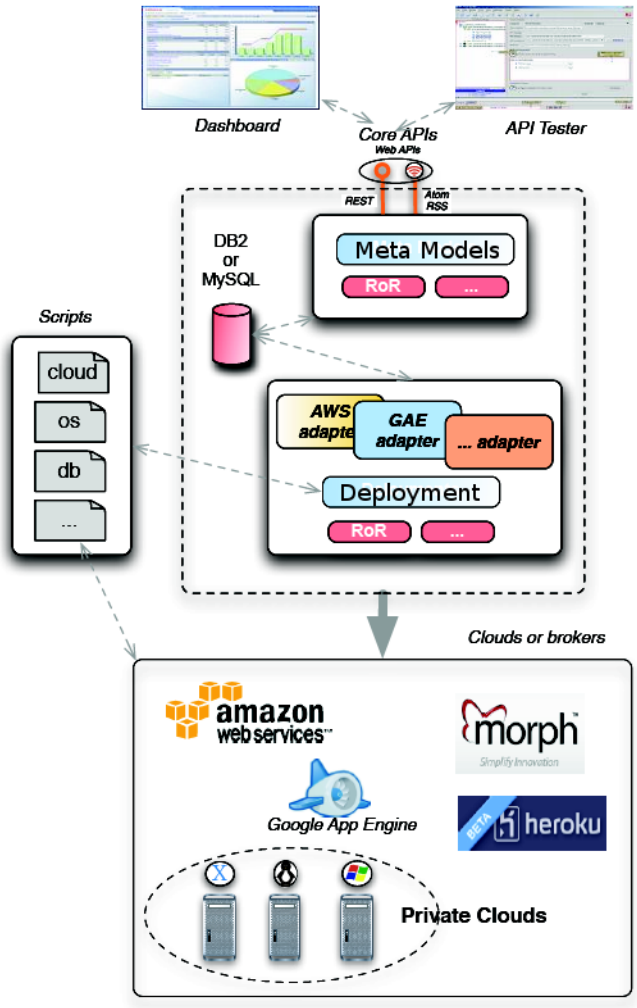


Fig. 1. High level view of the Altocumulus Middleware Platform

IBM Altocumulus currently supports Amazon EC2 and Amazon clones such as Eucalyptus [10], Google AppEngine [5] and the IBM HiPODS Cloud computing solution.

We use *adapters* to interface different clouds. The current implementation uses manually programmed adapters. We discuss the possibility of generating these adapters by a semi-automated process in section 6.

4 Best Practice Model

4.1 Overview

Our best practice model is designed around three principles.

1. Provide logically arranged place holders for different types of data required for a cloud deployment.
2. Be applicable to different cloud paradigms.
3. Have a reasonable balance between completeness and complexity.

The first design principle caters for the fact that there are significant number of different data items that is needed for a deployment. Although these data items may vary significantly between different deployments, they can be logically grouped and arranged.
The second principle states that such a grouping needs to be compatible with the different cloud paradigms described in section 2.2.
The third principle states that this grouping needs to be reasonably complete but not overly complex since it is meant ultimately for a human. This is especially important considering that various components of a best practice would evolve over time, i.e. will have different versions. The goal here is that a user would be able to redeploy a cloud application with an updated best practice using a one-click action. This can be significantly important when a new version of a best practice component is released for security and privacy purposes, e.g. a patch solving a security vulnerability.

We now describe the details of the *Altocumulus Best Practice* (BP). A BP in its simplest form is a particular sequence of process steps (selected either automatically or manually) that best suits the cloud management task at hand. A BP however, differs from a simple workflow since it may additionally include provisions to define other aspects of a deployment such as scaling. A BP consists of two primary components.

- **Topology** : An indication of the structure of a particular deployment. In the simplest form, the topology can be a single instance or an application space. However the topologies can be arbitrarily complex. In section 4.2 we discuss several common topologies.
- **Scaling Strategy** : Encapsulates the process for scaling. For some clouds, such as platform clouds, the scaling strategy would be empty. However for infrastructure clouds (IaaS providers) the scaling strategy includes details on what resources to manage when a scale-out or scale-in operation is required. The scaling strategy is tightly coupled with the topology.

The Scaling Strategy includes customizable *rules* that can be used to control the behavior of the scaling process. For example, for a strategy of scaling horizontally by replicating, the rules would control the load limit to spawn a new replication and the maximum number of replications allowed. These limits may be set by a user or a process.

The Topology is based on *instance groups*. An instance group refers to a set of functionally equal compute nodes or a cluster. Each instance group attaches to a *configuration bundle* that in turn includes an ordered set of *configurations*. A *configuration* represents a single step in a simplified work flow of setting up the node. For example installing a database software is a configuration. Configurations (optionally) have parameters, most of them user tunable. To use the above example, the default user name and password for the database would be part of the user tunable configuration parameters.
Figure 2 illustrates the structure of a Best Practice in UML notation.

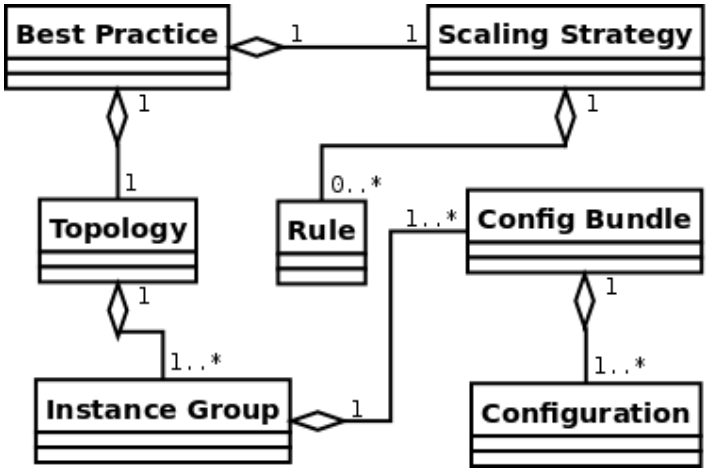


Fig. 2. Structure of a Best Practice

While some simple BPs have only on instance, advanced BPs allow deployments to be spread across groups of instances that can grow and shrink according to some rules. Due to the complexity of building a custom BP from scratch however the Altocumulus platform contains a number of pre-built BP templates for the most common cases. Regular users are allowed to customize these BP templates and only privileged users can construct them from scratch. We further discuss the issues involved in custom BPs in section 5.1.

4.2 Example Best Practices

Now we present three examples that clearly illustrate the flexibility of the best practice model and it's coverage of the design principles stated in section 4.1. We also describe some of the pain points these BPs try to overcome.

A Single Instance Best Practice This is the most simplistic BP where all the configurations apply to a single instance. There is no scaling strategy. The

topology is in its simplest form where one instance contains everything. To cater for such simple scenarios, the Altocumulus platform includes an empty scaling strategy and a fixed topology.

A Fixed Cluster Best Practice A popular use of compute clouds, especially among the scientific researchers, is to use them for parallelizable computations using the map-reduce paradigm [11]. Apache Hadoop [12] is one of the popular map-reduce computation frameworks used for large computations. Typically when setting up a map-reduce task the major pain point is setting up the management nodes and the worker nodes (worker nodes are generally large in numbers). The number of nodes for a computation task is generally fixed and does not fluctuate during a computation.

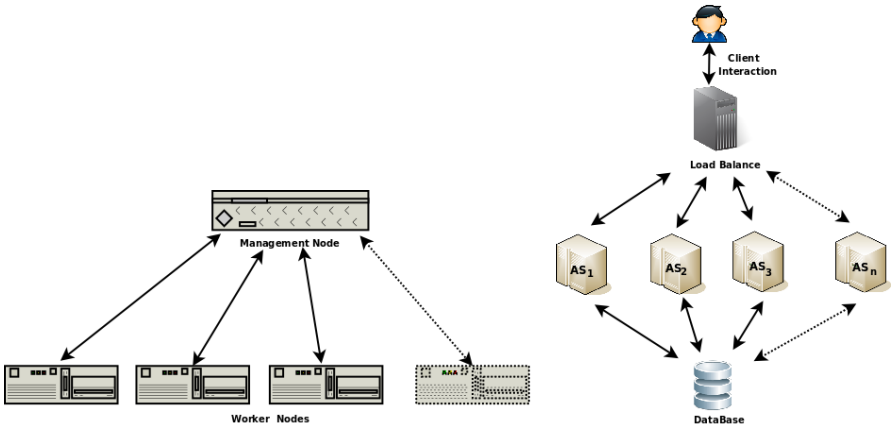
The fixed cluster BP supports this type of set up where the nodes may be heterogeneous but fixed in number. Currently this BP supports only the Hadoop framework and includes a multi-instance star topology with 1) A central management node and 2) A number of worker nodes. The scaling strategy is empty since there is no dynamic scaling involved. This type of setup is illustrated in figure 3(a).

A Horizontally Replicating Best Practice Todays typical framework based Web sites are two tiered, i.e. they have a presentation front-end that displays content fetched from a database back-end. Majority of public Web sites such as wikis, forums or blogging platforms are structured this way. The popular open source multi purpose Web platform Drupal [13] is one of the prominent examples for a two tiered Web application platform. Scaling strategies for such two tiered Web applications have been well studied and one of the accepted strategies is to replicate the presentation component under a load balancer [14]. The pain points in such a scaling process includes 1) replicating the complete configuration of the presentation component which is typically a script language based application. and 2) updating the load balancer configuration with the details of newly added (or removed) instances.

This BP includes a multi-instance layered topology that includes 1) load balancing layer 2) presentation / Application Server (AS) layer and 3) database layer. The scaling strategy is horizontal replication and includes rules that trigger replications. Figure 3(b) illustrates this architecture.

5 Discussion

IBM Altocumulus is a new kind of cloud middleware: a cloud broker that facilitates cloud agnostic deployment and management tasks. The primary goal of this project is to attempt to homogenize clouds with a layer and API that would allow higher-level services to be created. From our Altocumulus deployment in the IBM Technology Adoption Program (TAP) (An internal portal of emerging technologies with access only to IBM employees) we have learned that there is



(a) Star Topology for Hadoop

(b) Layered Topology for Horizontal scaling

Fig. 3. Different Best Practice Topologies

indeed a demand for this type of a cloud abstraction middleware. Recent developments in the hybrid cloud deployments, even across different paradigms [15] underscores the importance of cloud homogenization.

5.1 Custom Best Practices

Ultimately the intention of the best practice model is to provide cloud users with reasonable power and flexibility in cloud application management. However creating an advanced BP template needs a significant level of knowledge and testing. For example creating the fixed cluster BP described in section 4.2 requires in depth understanding of the Hadoop framework. Due to this reason we have limited the BP creation capability only to privileged users. The regular users can always customize existing templates. We believe this is not a serious limitation since majority of the use cases are covered by our existing BP templates. Our TAP deployment experience also confirms this fact.

5.2 Other Lessons learned

Apart from the above mentioned experiences, we also learned important lessons in applying a middleware platform in the cloud context.

1. Some clouds, especially platform services, mandate the use of certain libraries for application development. Thus these applications become *locked-in* to the specific cloud provider. Our middleware cannot solve this lock-in yet. A different and completely new application development paradigm is needed to generate portable cloud applications. Although such an effort is out of scope for this research, we believe cloud agnostic application management is the first step in truly portable cloud applications.

2. Hybrid cloud deployments are deemed important amidst the mounting privacy and data protection considerations. Although Altocumulus doest not provide hybrid deployments yet, we consider this to be the most important next step.

6 Future Work

In terms of improving the middleware capabilities, we plan to expand the support for application frameworks and relational databases as well as other clouds. Supporting hybrid cloud deployments our key next step.

An interesting avenue of research we plan to investigate is the use of semantic web technologies to enhance the functionality of this cloud middleware. We believe that some of the semantic web techniques can be used to help facilitate the creation of cloud adapters as well as discovery and usage of best practices. An interesting case would be a system capable of suggesting a suitable best practice given an application. Similar scenarios have been investigated heavily in the context of semantic Web services and we anticipate further research can yield the adaptation of such technologies in the cloud context.

The experience gained in semantic annotations can also be used in the context of assembling user defined BPs. A possible line of future research includes using meta data annotations to provide assistance to users for assembling BP components and hence may provide a solution for the dilemma discussed in section 5.1.

References

1. Fitzpatrick, B.: Distributed caching with memcached. Linux journal (124) (2004)
2. Bernstein, P.A.: Middleware: a model for distributed system services. Commun. ACM **39**(2) (1996) 86–98
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009)
4. Inc, A.: Amazon Elastic Compute Cloud (Amazon EC2) (2008)
5. Sanderson, D.: Programming Google App Engine: Rough Cuts Version (2008)
6. Beck, K., Cunningham, W.: Using pattern languages for object-oriented programs. Specification and Design for Object-Oriented Programming (OOPSLA-87) **67** (1987)
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. (1995)
8. Fowler, M.: Patterns of enterprise application architecture. Addison-Wesley (2004)
9. Babaoglu, O., Jelasity, M., Kermarrec, A.M., Montresor, A., van Steen, M.: Managing clouds: a case for a fresh look at large unreliable dynamic networks. SIGOPS Oper. Syst. Rev. **40**(3) (2006) 9–13
10. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid-Volume 00, IEEE Computer Society (2009) 124–131

11. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. OSDI (2004) 1
12. Bialecki, A., Cafarella, M., Cutting, D., OMalley, O.: Hadoop: a framework for running applications on large clusters built of commodity hardware. Wiki at http://lucene. apache. org/hadoop (2005)
13. Mercer, D.: Drupal: Creating Blogs, Forums, Portals, and Community Websites. (2006)
14. Talwar, V., Wu, Q., Pu, C., Yan, W., Jung, G., Milojicic, D.: Comparison of approaches to service deployment. In: 25th IEEE International Conference on Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. (2005) 543–552
15. Chohan, N., Bunch, C., Pang, S., Krintz, C., Mostafa, N., Soman, S., Wolski, R.: AppScale Design and Implementation. Technical report, UCSB Technical Report Number 2009 (2009)