Develop and Scale Mobile Services in Cloud Computing Scenarios

Martin Ott

equinux
ott@equinux.com

Abstract: Mobile app development occurs in a highly competitive, and fast-moving environment. Small teams or even individual developers build apps that can be very successful, sometimes literally over night. Many of these successful apps are driven by backend services that are either interfaces to established web applications or that exist just for the purpose to serve the mobile apps. We discuss our experience with cloud computing scenarios which enable a small team to develop and scale mobile services for hundreds of thousands or even millions of users.

1 Introduction

Mobile services can be developed and deployed in various cloud computing scenarios. We distinguish the following scenarios as described in [VRMCL08] and [Fac13]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Mobile Backend as a Service (MBaaS). In figure 1 we depict an exemplary 3-tier mobile web service architecture.

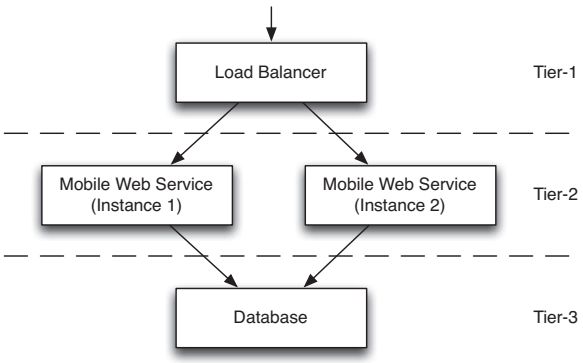

Figure 1: Example of a 3-tier mobile web service architecture

We use it as the common denominator in our discussion of cloud computing scenarios to compare how the tiers are implemented in the various scenarios.

Its public interface is based on the HTTP protocol. Tier-1 receives requests and dispatches them to instances of the mobile service on tier-2. This layer implements the public API of the service and runs one or more instances of the implementation. Tier-3 provides persistence for data of the mobile service by using a shared database.

This architecture has also been implemented in an IaaS-based, and a PaaS-based mobile service we have built, which serve as the backends for the iOS apps TV Movie [Mov13] and TV Movie HD [HD13]. At the time of this writing the mobile services powering these apps are in use by 875000 end-users.

On the basis of the exemplary architecture described above we discuss our experience with the programming interfaces offered in the different cloud computing scenarios, and how mobile services can be scaled by developers.

2 Programming Interfaces

Mobile apps are built on the APIs the vendors of mobile operating systems provide. When building mobile services in cloud computing scenarios, developers can also rely on APIs. For IaaS and PaaS scenarios their APIs provide ways to manage the infrastructure the service runs on. MBaaS service providers offer SDKs and APIs that are directly integrated into the mobile apps.

- `IaaS`: It offers compute and storage resources among other services as the basic building blocks. Vendors like Amazon Web Services [Ser13] let developers use SDKs and APIs to programmatically manage these resources. To build the described exemplary mobile web service architecture 4 compute instances with attached storage are required. On top of these compute instances developers would need to define and deploy their own software stacks, deploy a database of their choice and implement the mobile web service on top of that stack. Configuration management tools like Puppet [Pup13], and Chef [Che13] can help in the process of setting up and operating such stacks. On the one hand IaaS APIs make programmatic management of infrastructure possible and therefore lower the barrier for small teams to build distributed architectures for mobile services. On the other hand developers choose their complete software stack which lets them tune it for the specific requirements and purpose of the mobile service.

- `PaaS`: Predefined framework-specific and programming language-specific software stacks are deployed by the PaaS platform. Custom software stacks can be developed following the specifications given by the PaaS. The functionality of tier-1 is provided by the PaaS platform. Developers implement the mobile web service on tier-2 using the given software stack. Deployment workflows for the services in development are predefined. Developers can concentrate on developing the business logic of their service. APIs are provided from the PaaS platform to manage and interact with the running processes. Databases like the one we plan for tier-3 are often provided directly by PaaS platforms. For example, Heroku [Her13] provides PostgreSQL in

a SaaS scenario. Other databases are offered by various vendors. PaaS platform usually do not have specific SDKs in place for mobile services.

- `MBaaS`: Providers like StackMob [Sta13], Parse [Par13], Kinvey [Kin13], and FatFractal [Fat13] implement all three tiers of the architecture we have outlined above. The middle layer usually includes APIs to authenticate and manage users. The core is a REST-based API built on the principles described in [Fie00]. This API basically enables developers to read, write, update, and query arbitrary data. It acts as an persistence interface to tier-3 databases. Usually additional services like location queries, push notifications to clients, and the ability to run custom business logic from the app developers are provided in MBaaS scenarios. The API endpoints are not only implemented using REST-based protocols but are also provided as native SDKs for popular mobile operating systems. Developers can employ the same skill set for developing the mobile app and for interacting with the mobile service. On the other hand customization of the mobile service is either very limited or not possible at all.

3 Service Scalability

The widespread availability of mobile app stores, its top charts, and featured apps sections expose apps to many new users. This can result in fast user growth, even over a very short period of time like a few hours or days. When affected apps are backed by mobile services they need to be able to scale with this fast growth of users. For example, the backend service for the TV Movie app had to be scaled out over and over to accommodate the growing number of users. Since most of the mobile services implement their interfaces using HTTP-based protocols they can make use of architectures, techniques, and tools that have proven successful for the operation of large websites or web apps assuming the mobile service itself can run concurrently. Cloud computing tools promise to address these scalability needs:

- `IaaS`: Platforms like AWS EC2 and its corresponding services allow developers to manage the scalability needs down to compute and storage resources. For example, a mobile service that requires an in-memory database on tier-3 that cannot be scaled horizontally needs larger compute instances in order to scale up. IaaS often provides a multitude of specifically tuned compute resources for the different needs that mobile service architectures require. On the other hand scaling resources can be costly for developers because they need to manage their software stack right above the compute resources that IaaS offers. For example, scaling the web process layer on tier-2 usually requires not just to launch another process but to launch a new compute instance and dependent resources in order to run the new process.
- `PaaS`: Platforms like Heroku [Her13] or Google App Engine [Eng13] operate a routing layer which maps to tier-1 in our mobile web service architecture. This frees developers from maintaining tier-1. Developers can concentrate on designing tier-2 processes so that the service can scale out by controlling the number of running

instances. Tier-3 services like shared databases usually can not be deployed in PaaS environments. Software as a Service (SaaS) tools can be choosen as a replacement.

- `MBaaS`: The provider of the MBaaS platform transparently scales the mobile web service as needed. Developers do not have access to control any of the tiers that the MBaaS platform runs the mobile web service on.

4 Conclusion

The discussed cloud computing scenarios to build mobile services provide everything from basic infrastructure building blocks to SDKs that are directly embedded into mobile apps. On one end of the spectrum mobile services can be built on top of IaaS platforms where they can be tailored exactly for the specific requirements developers have for their products. Scaling is performed directly at the infrastructure level resources which are composed by developers to the desired architecture. But scaling operations are usually supported by automated mechanisms. MBaaS offerings on the other end of the spectrum enable developers to start with predefined services that can be accessed with native SDKs. They are scaled transparently. PaaS platforms provide a middle ground where the infrastructure, deployment workflows, and scaling schemes are predefined and developers need to implement the API for their mobile services using 3rd-party or custom frameworks.

References

[Che13] Chef. `http://www.opscode.com/chef/`, 2013. [Online; accessed 15-January-2013].

[Eng13] Google App Engine. `https://developers.google.com/appengine/`, 2013. [Online; accessed 15-January-2013].

[Fac13] Michael Facemire. Mobile Backend as a Service. `http://blogs.forrester.com/michael_facemire/12-04-25-mobile_backend_as_a_service_the_new_lightweight_middleware`, 2013. [Online; accessed 05-February-2013].

[Fat13] FatFractal. `http://fatfractal.com`, 2013. [Online; accessed 15-January-2013].

[Fie00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. AAI9980887.

[HD13] TV Movie HD. `http://www.equinux.com/goto/tvmoviehd`, 2013. [Online; accessed 15-January-2013].

[Her13] Heroku. `http://www.heroku.com`, 2013. [Online; accessed 15-January-2013].

[Kin13] Kinvey. `http://www.kinvey.com`, 2013. [Online; accessed 15-January-2013].

[Mov13] TV Movie. http://www.equinux.com/goto/tvmovie, 2013. [Online; accessed 15-January-2013].

[Par13] Parse. https://www.parse.com, 2013. [Online; accessed 15-January-2013].

[Pup13] Puppet. http://puppetlabs.com/puppet/puppet-open-source/, 2013. [Online; accessed 15-January-2013].

[Ser13] Amazon Web Services. http://aws.amazon.com, 2013. [Online; accessed 15-January-2013].

[Sta13] StackMob. https://www.stackmob.com, 2013. [Online; accessed 15-January-2013].

[VRMCL08] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.