Real-Time Communications
Without Boundaries

ribbon

Microservices:

Optimizing Session Border Controllers in the Cloud

White Paper

Contents

Introduction

For many years, real-time communication (RTC) networks and services have been built by installing monolithic software applications running on either proprietary hardware or carefully selected and qualified off-the-shelf servers. As processing demands increased, capacity was added in units of "a box"—or sometimes a server card in a box. Scaling these legacy RTC networks has always been challenging, requiring advance planning, purchasing, and implementation with careful adherence to maintenance schedules and processes. In fact, it frequently became a hurdle to expansion, new service roll-out, and ultimately increasing revenues.

As an industry, we are increasingly becoming comfortable with the concept of applications and services moving to the Cloud. Virtualization and service orchestration technology are increasingly relied upon to simplify and expedite the introduction and scaling of services. If we are to really get the most benefit from applications and services in the Cloud, we need to look hard at how traditional software has been structured. Simply porting applications "as is" to the Cloud, only provides the most rudimentary of advantages. To really capitalize on the full potential of the Cloud, we need to use application software that is structured differently.

What Are Microservices

The term "microservice" refers to a style of application development where instead of developing one large monolithic software system, a suite of independently deployable, small, modular services is defined in which each service runs a unique process. These independent processes communicate through well-defined, lightweight mechanisms to serve a common business goal. Each microservice ideally performs one function, is optimized for their own scaling abilities/characteristics, and may utilize different development technologies when appropriate.

In many ways this is nothing new, but it's just the latest version of the age-old debate about how modular software should be, and how it can be divided into well-defined components with clear (loosely coupled) interfaces. Ultimately, the result of this modularity is that it allows individual parts to be replaced, scaled, or changed without impacts on other portions of the solution. Done correctly, the application of these principles speeds network function virtualization (NFV) deployments and makes service creation more efficient, so service providers are more capable of meeting "cloud scale" traffic demands.

Monolithic versus Microservices

An application built around microservices contrasts with one built on a monolithic approach in multiple key ways:

1. A monolithic application has to replicate all code subsystems each time it scales by one "unit", even if it is only part of the application code that is causing a capacity or processing bottleneck. An application using a microservices design only needs to scale each microservice as it becomes a bottleneck.
2. Microservices can each follow their own lifecycle, being built out of the most appropriate technology components and languages. They can also be rewritten/replaced over time without wholesale replacement of the application (so long as the interfaces are honored). Although this is not impossible in a monolithic approach, it is much more difficult.
3. Microservices allow for more reuse; as new applications are deployed, they can access the code already running, chaining microservices as needed. In a monolithic approach, separate instances of source code would typically be "compiled in" to the new application, complicating upgrades and maintenance.
4. Microservices enable service agility. An application that has been built by chaining together multiple microservices will have quicker reactions to changes in service demand or changes in functional needs than one built using a monolithic approach.
5. Microservices enable service efficiency, so when an application needs to scale it will only require resources that it will actually use. This is not possible with a monolithic approach.
6. Microservices naturally lend themselves to an "agile" development methodology and shortened cycle times, enabling faster service velocity than would be achieved with a monolithic approach.

Applying Microservices to the Ribbon Communications SBC

Ribbon has adopted a microservices architecture for our Session Border Controller Software Edition (SBC SWe) to ensure our customers receive maximum benefit from deploying virtual SBCs in the Cloud. Nominally an SBC must provide at least the following functions for real-time communications:

1. Signaling Services: security, protocol manipulation, and protocol interworking for the SIP session
2. Media Services: security, Quality of Service (QoS) control, Network Address Translation (NAT), media anchoring, port firewalling, and bandwidth policing for media
3. Transcoding Services: transcoding, transrating, DTMF tone detection/interworking, and RTCP reporting

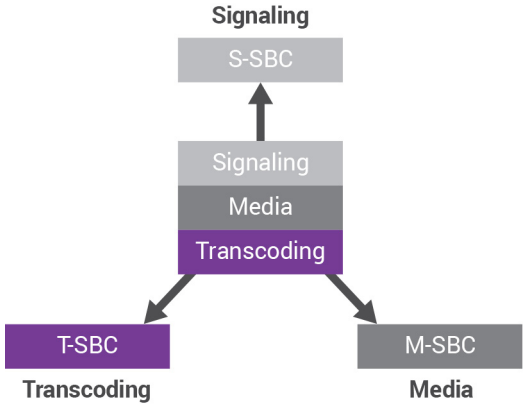

Figure 1. SBC SWe Microservices

Given the design construct of the SBC SWe, it made sense to divide the SBC SWe functionality into these three categories: Signaling, Media, and Transcoding. This is shown in Figure 1 above.

In our analysis, each of these categories has several key differences which lend themselves to take advantage of microservices.

Signaling

For signaling services, the first thing that stands out is how the packet per second processing is a different order of magnitude than what is needed for either media or transcoding. This is because the signaling function is only involved in handling SIP signaling messages for a call/session setup and teardown. However, the logic executed for each message can be very high, with large amounts of application code required to process a session. This makes it ideally suited to x86 CPU architectures. Two other attributes associated with signaling that make it ideal for a microservices implementation:

- Signaling processing demands can be highly variable, based on the type of call/session, and very different from media and transcoding
- User endpoints, such as IP phones, expect the SBC to be a single point of contact, even if it is really a cluster of Signaling SBC virtual network functions (VNFs) in a virtualized Cloud-based deployment. This leads to unique traffic distribution and load balancing requirements for signaling that would not otherwise apply to media and transcoding functions

Media

Media services require high packet throughput, but each packet is relatively small and has relatively small processing needs. Historically this has been implemented in network processors to handle security and scaling. In a Cloud deployment, this SBC function can be implemented using x86 CPU code, but with the SBC SWe adoption of microservices two additional options become available—one forward-looking, the other backward compatible.

Often Software-Defined Network (SDN) and NFV are discussed together when people talk about moving applications and services to the Cloud. This holds true for a forward-looking option to handling media packets in the Cloud. With microservices, it will be possible to off-load the handling of media packets by pushing the media packet processing logic down into a software-defined network infrastructure.

The backward compatible option uses microservices to off-load the processing of media packets by using traditional, hardware-based SBCs. Because Ribbon' SBC SWe is the same code base as the hardware-based SBCs, this option provides a seamless interworking between the signaling, transcoding, and media functions.

Transcoding

Transcoding services require high packet throughput and high intensity computation. This SBC function has traditionally implemented Digital Signal Processors (DSPs) in hardware appliances. In a Cloud deployment, where DSPs are not available, this SBC function can be implemented using x86 CPU code, but with greatly reduced performance. However, like media processing, with the SBC SWe adoption of microservices two additional options become available—one forward-looking, the other backward compatible.

A forward-looking option uses Graphical Processing Units (GPUs) instead of CPUs to perform the transcoding function. GPUs are designed for high intensity computation and directly applicable to transcoding performance and scale. GPUs provide 4x the transcoding as CPUs and do so at almost 2x lower power consumption. GPUs are now available from a number of Cloud Service Providers, such as Amazon Web Services, making this a near-term deployment option.

Similar to media processing, the backward compatible option leverages microservices to perform transcoding using DSPs on hardware-based SBCs. Because Ribbon' SBC SWe is the same code base as the hardware appliance–based SBCs, this too will be seamless interworking between the signaling, media, and transcoding functions.

Chaining Microservices to Create a Complete SBC Service

The Ribbon SBC SWe can be deployed as an integrated SBC or as a combination of three microservices, each capable of scaling independently and optimized using different deployment models. Because each microservice is independent, this might appear more complex than a monolithic design; however, in reality much of this complexity was always in the application—it was just hidden.

With microservices, managing inter-process communications is one of the trade-offs being made in exchange for the ability to chain together individual microservices on demand to form a much more scalable and efficient overall SBC service. The importance of this really shines when customers move to a full NFV implementation and want to be able to orchestrate multiple VNFs, one of which might be an SBC function. Now they can independently instantiate and scale the SBC itself as a series of VNFs for signaling, media, and transcoding.

Leveraging Microservices to Simplify Migration to the Cloud

As pointed out above, one of the benefits of moving the SBC SWe to microservices is the ability to simplify a service provider's migration of SBCs to the Cloud. With microservices, it is possible to provide backward compatibility for media and transcoding, giving service providers a way to re-use their existing investment in SBCs. While this interworking is seamless for Ribbon SBCs, because of the common code base between the software and hardware solutions, the microservices concept could extend this to non-Ribbon appliances using well-defined inter-processing communications, most likely using REST APIs.

Insights and Lessons Learned

As part of the process of moving from an integrated to a microservices model for the Ribbon SBC, there are several key insights and lessons we learned:

- The first was that simply exposing individual functions is not sufficient to make microservices work. It requires for thought in what and how inter-process communication needs to be done. It requires understanding performance when replacing direct function calls with API calls. Granularity is the key to this understanding, as there will be inherent trade-offs between the "chattiness" of a protocol and the scalability.
- The second insight was that sometimes technology choices look good on paper, but until you prototype (at scale) you won't fully gain an understanding of both functional and interworking performance.
- The third was that automation is important as more parts become visible. It is important to invest in automated discovery and to make the microservices as self-organizing as possible. It is important to be able to scale by having the microservices join/ leave clusters automatically, and to have update/broadcast mechanisms so other elements learn about changes.

Conclusion

As a market-leading provider of session border controllers, Ribbon is always looking for ways to make our solution better and more optimized for our customers. Nowhere has this been more important than in our adoption of the Ribbon SBC SWe to microservices. Microservices and the inherent advantages they offer for deploying a virtual SBC in the Cloud took innovative development and substantial testing using real-world traffic scenarios. With microservices, we are now enabling our customers to take full advantage of deploying virtual SBCs in the Cloud.

About Ribbon Communications

Ribbon is a company with two decades of leadership in real-time communications. Built on world class technology and intellectual property, Ribbon delivers intelligent, secure, embedded real-time communications for today's world. The company transforms fixed, mobile and enterprise networks from legacy environments to secure IP and cloud-based architectures, enabling highly productive communications for consumers and businesses. With locations in 28 countries around the globe, Ribbon's innovative, market-leading portfolio empowers service providers and enterprises with rapid service creation in a fully virtualized environment. The company's Kandy Communications Platform as a Service (CPaaS) delivers a comprehensive set of advanced embedded communications capabilities that enables this transformation.

To learn more visit RibbonCommunications.com

www.ribboncommunications.com