Cloud-Tiered Architectural Models are Bad Except When They Aren’t

Lori MacVittie, 2011-09-02

Database as a service is part of an emerging model that should be evaluated as an architecture, not based on where it might be deployed

These days everything is being delivered "as a Service". Compute, storage, platforms, IT, databases. The concept, of course, is sound and it is generally speaking a good one.

If you're going to offer an environment in which applications can be deployed, you'd best offer the services appropriate to the deployment and delivery of that application. And that includes data services; some kind of database.

Shortly after the announcement by Salesforce.com of its database as a platform service – database.com – Phil Wainewright posited that such an offering might "squish" smaller providers. Some vehemently disagreed, and Mr. Wainewright recently published a guest post, written by Matt McAdams, CEO of TrackVia, regarding the offering that disputes that prediction:

> Rather, Salesforce.com is making the existing database that currently underlies its CRM and Force.com platforms accessible to subscribers who don't have accounts on one of those two platforms. The target audience is programmers who want to build an application outside of Force.com, but want a hosted database.
>
> Unfortunately, web application developers will find the idea of hosting their data outside their application platform severely unappealing. The reason is latency.
>
> Database.com: nice name, shame about the platform

Mr. McAdams goes into more detail about the architecture of modern web applications and explains the logic behind his belief. He's right about latency being an issue and is backed up by research conducted by developer-focused Evans Data Corporation.

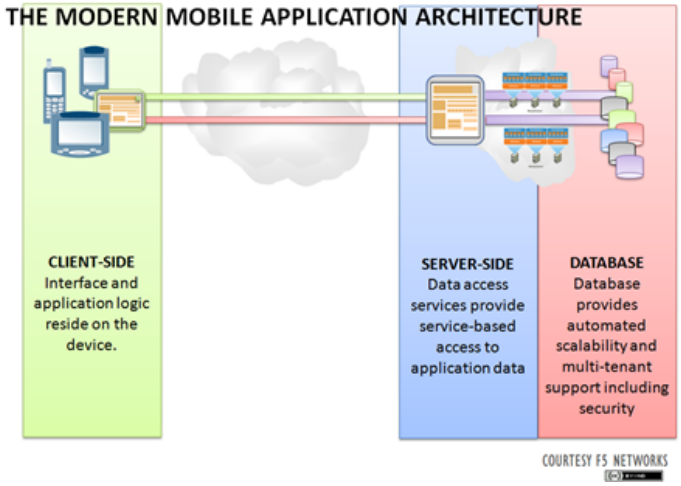
> Developers report that **performance is the second-most important attribute found in frameworks and platforms**. The ability of a framework or a platform to deliver high throughput, minimal latency and efficient use of computing resources is a major factor in decisions regarding which application frameworks to use for development. [emphasis added]
>
> -- Evans Data Corporation *Users' Choice: 2010 Frameworks (*April 2010)

So performance is definitely a factor, but there's more going on here than just counting ticks and some of what's happening completely obviates his concerns. That's because he ignored the **architectural** model in favor of narrowing in on a specific **deployment** model. There are a few critical trends that may ultimately make the Database as a Service (DaaS) **architectural** model (not necessarily the provider-based **deployment** model) a success.

The CLIENT-DATABASE ARCHITECTURAL MODEL

The consumerization of IT is well underway.

No one lightly dismisses the impact on application platform development of consumer-oriented gadgets like the iPad and the iPhone. More and more vendors as well as organizations are taking these "toys" seriously and subsequently targeting these environments as clients for what have traditionally been considered enterprise-class applications.



But the application architecture of the applications deployed on mobile devices is different from traditional web-based applications. In most cases an "app" for a mobile device employs a modernized version of the client-server model with nearly all client and application logic deployed on the device and only the data store – the database – residing on the "server". It's more accurate to call these "client-database" models than "client-server" as often it's the case that the "server" portion of these applications is little more than a set of services encapsulating database-focused functions. In other words, it's leveraging data ***as a service***.

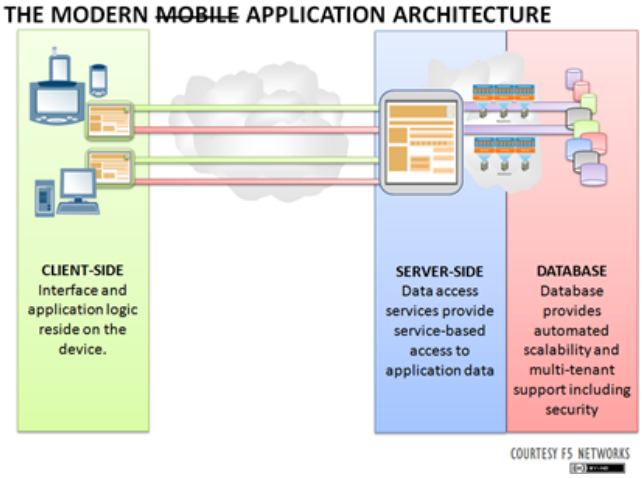
It's still a three-tiered application architecture, to be sure, but the tiers involved have slightly different responsibilities than a modern web application. A modern web application generally "hosts" all three tiers – web (interface or 'presentation' in developer-speak), application, and data – in one location. For all the reasons cited by Mr. McAdams in his guest post, developers and subsequently organizations are loathe to break apart those tiers and distribute them around the "cloud". I'd add that reliability and availability as well as security (in terms of access control and the enforcement of roles in a complex, enterprise-class application) play a part in that decision but it's the latency that really is the deal-breaker especially between the application and the database.

But Lori, you say, HTML5 and tablets are going to change all that. Applications will ***all*** be HTML5 and even mobile devices will return to the more comfortable three-tiered architecture.

Will it? HTML5 has some interesting changes and additions that make it more compatible with a mobile application architecture than earlier versions of the specification. Offline storage, more application logic capabilities, more control. HTML5 is a step toward a ~~fatter~~ robust, more complete application client platform. Couple that with the fact that web applications have been moving toward a more client-centric deployment model for years and you've got a trend toward a more client-database application architectural model. Web applications have been getting fatter and fatter with more and more application logic being pushed onto the client. HTML5 appears to support and even encourage that trend. Consider, for a moment, this web application written nearly entirely in JavaScript. Yes, that's right. Nearly all of the functionality in this application is contained within the client, in 80 lines of JavaScript code. That's a client-database model.

Even on mobile devices, on tablets designed to better support "traditional" web-applications, they are there almost as an after-thought. The browser is used for reading articles and watching video – not interacting with data. It's the applications, the ones developed specifically for that device, that make or break the platform these days. If that weren't the case, you wouldn't need separate "applications" for Facebook, or Twitter, or Salesforce.com. You'd just leverage the existing web application, wouldn't you? And you certainly wouldn't need APIs upon which ***applications*** could be built, would you?

Of course not. So what, you might be asking, does all this have to do with the success or failure of database as a service?



The answer is this: developers are lazy, and because we're lazy we're masters at architecting solutions that can be reused as much as possible to limit the amount of tedious and mundane coding we have to do. Innovation and change is often driven not by inspiration, but by the inherent laziness of developers looking for an "easier" way to do a thing. And supporting two completely separate application architectures is certainly in the category of "tedious".

WHEN BEING LAZY is a BONUS

As more and more organizations decide to support both mobile and web versions of their critical business applications, it's the development staff that's going to be called upon to provide them. And developers are - no offense intended as I still self-identify as a developer - lazy.

We don't like the tedious and the mundane; we don't like to repeat the same code over and over and over. We look for ways to reuse and simplify such that we can concentrate on those pieces of development that are exciting to us – and that doesn't include anything repetitious.

So as developers look at the two models, they're eventually going to abstract them, especially as they explore HTML5 and find more and more ways to align the two models. They're going to note the differences and the similarities in architecture and come to the conclusion that it is inefficient and potentially risky to support two completely different versions of the same application, especially when the option exists to simplify them. An architectural model in which the data access portions are shared – hosted as a service – for both mobile and traditional desktop clients makes a lot more sense in terms of cost to develop and maintain than does attempting to support two completely separate architectural models.

Given the movement toward more client-centric applications – whether because of platform restrictions (mobile devices) or increasing demand for functionality that only comes from client-deployed application logic (Web 2.0) – it is likely we'll see a shift in deployment models toward client-database models more and more often in the future. That means **data as a service** *is* **going to be an integral part** of a developers' lives.

That's really the crux of what Mr. McAdams is arguing against – that a data as a service **deployment** model is untenable due to the latency. But what he misses is that we're already half-way there with mobile device applications and we've been moving in that direction for several years with web-based applications *anyway.* APIs for web applications today exist to provide access to what – data. Even when they're performing what appears to be application "tasks" – say, following a Twitter user – they're really just manipulating *data* in the *database.* There is no real difference between accessing a data service over the Internet that is deployed in the enterprise data center versus in a cloud computing provider's data center. So the real question is not whether such an **architectural** model will be employed – it will – it's *where will those data services reside?*

And the answer to that question doesn't necessarily require a lot of technical discussion; ultimately that has to be a *business* decision.

- The Database Tier is Not Elastic
- 80-line JavaScript Web Application
- Does This Application Make My Browser Look Fat?

- HTTP Now Serving … Everything
- The New Distribution of The 3-Tiered Architecture Changes Everything
- The Great Client-Server Architecture Myth
- Infrastructure Scalability Pattern: Sharding Sessions
- Infrastructure Scalability Pattern: Partition by Function or Type
- Applying Scalability Patterns to Infrastructure Architecture
- Sessions, Sessions Everywhere

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com