COSC 416: Topics in Databases (DBaaS)

TOPIC 2: DBAAS ARCHITECTURE

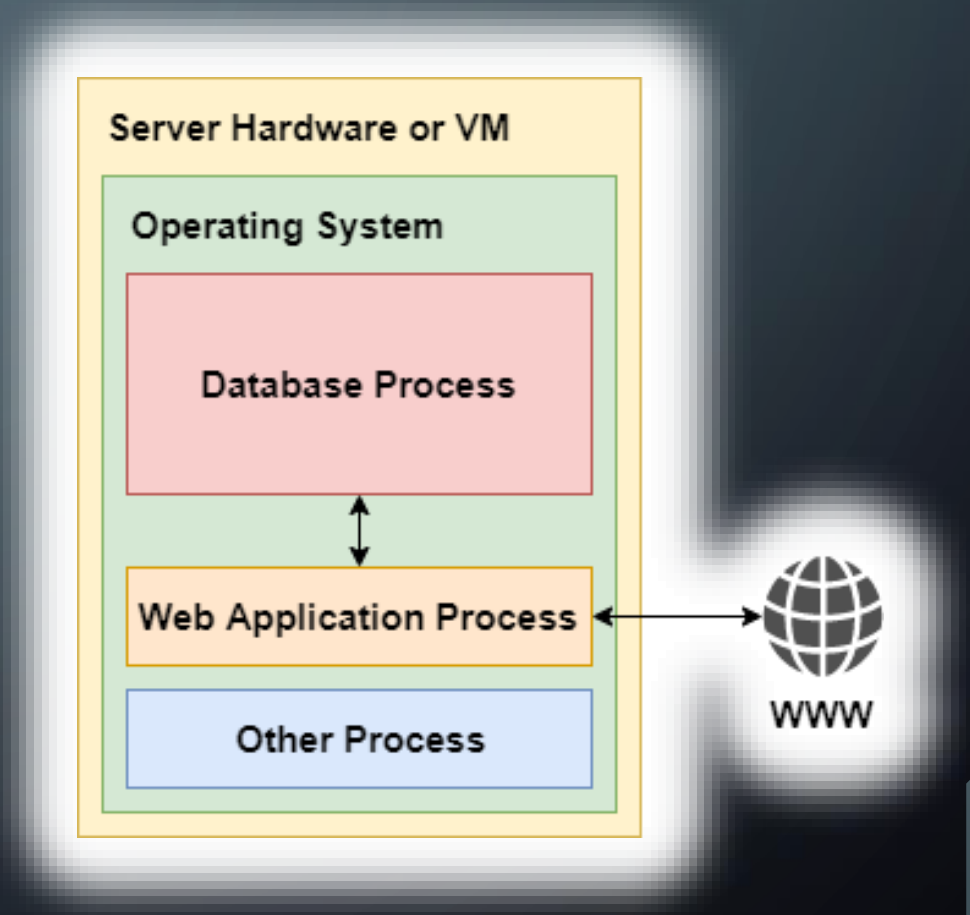
SCHEDULE

1. *DBaaS Architecture Rundown*
2. *Benefits and Drawbacks of DBaaS Architecture*
3. *A look at Amazon RDS*

DBAAS ARCHITECTURE

A TRADITIONAL DATABASE SERVER

- Let's consider a traditional database server for a moment
- The server may be a standalone database server or shared with an application
- The server may be on physical hardware or on a virtual machine



A HOLISTIC APPROACH

- Using actual hardware or a VM, we are dealing with an entire *software stack* (and potentially hardware stack) to run our database
- This software stack includes the operating system, the DBMS, and any dependencies for the DBMS
- System configuration is ultimately up to the system administrator or developer

WHAT ABOUT CLUSTERS?

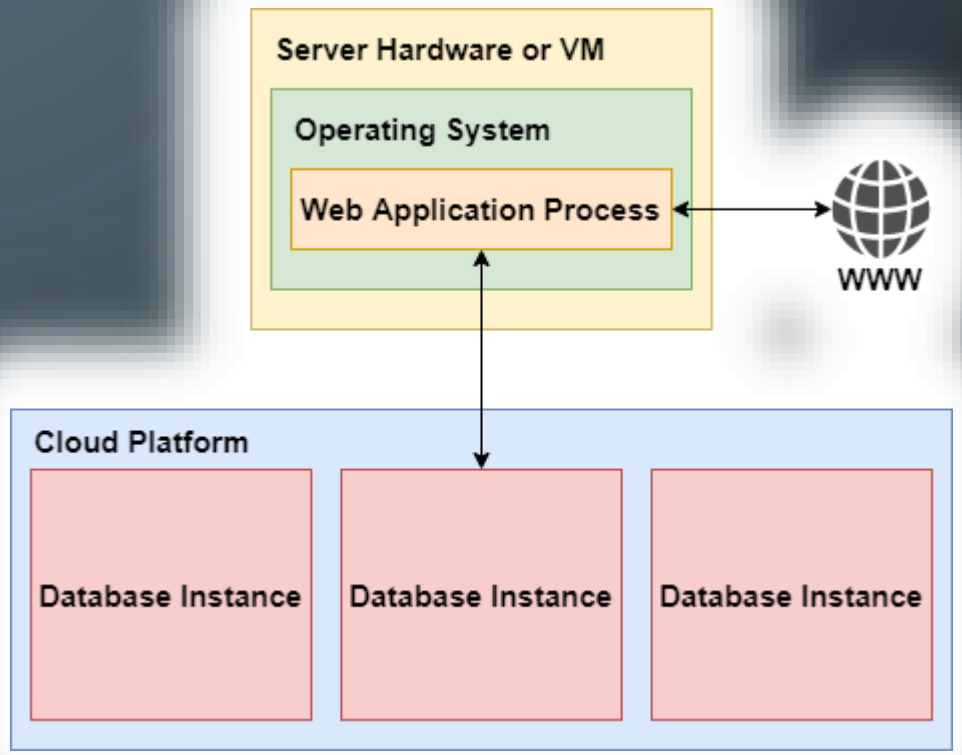
- You might wonder how cluster-based databases like MongoDB fit into this
- Traditional installations still require that each node be configured → Still using a software stack
- Multiplies work, but also allows for things like replication (automatic recovery) and better scaling through the use of multiple nodes

SO HOW IS DBAAS DIFFERENT?

- Using a Database-as-a-Service gives us a database process, without the underlying *software stack*
- This eliminates the need to handle the operating system, dependencies, and underlying server itself

DBAAS ARCHITECTURE

- Instead of running the database ourselves, we offload that work to a cloud platform
- The database instance is accessed remotely by the client or application process



WHAT EXACTLY IS “THE CLOUD”

- Amazon likes to helpfully point out that RDS is hosted “In the cloud”, but what does that mean exactly?
- The cloud is really just a collection of networked dedicated servers, which run hypervisors that manage individual guest machines, which in turn run the database instances
- The guest machines are preconfigured for the database and since they are virtual, they are disconnected from the hardware itself (aiding scalability)

HOW DOES DBAAS DIFFER FROM A VM?

- DBaaS systems are very much like virtual machines that you might rent from Digital Ocean or Linode
- Where they differ is the level of *management* offered
- With a virtual machine, you're still usually responsible for installing and maintaining your software stack
- Additionally, most VMs aren't specialized for database use (emphasis on network connection speed and disk I/O speed)

MANAGED SOFTWARE SOLUTIONS

- The idea of a *managed software solution* is that it is easy and fast to create, use, and teardown a database instance, without having to deal with the messy parts (configuration and maintenance)
- Virtual machines remove the need for hardware management in servers, while DBaaS solutions remove the need for software management, beyond the DBMS itself

AT THE ENTERPRISE LEVEL

- At a large-scale enterprise level, managed software solutions may even include regular database maintenance and design
- This includes having dedicated database engineers and administrators on hand to help design and maintain the database itself
- Totally managed solutions are, understandably, quite expensive, but are considered worthwhile at the enterprise level, where downtime could result in a considerable cost

COMMUNICATING WITH THE DATABASE

- In a traditional database server architecture, we interact with the database in one of two ways:
  - Inside the database console, performing CRUD operations either locally or via remote connection
  - Outside the database console: creating backups, sizing databases, starting and stopping the DBMS

COMMUNICATING WITH A DBAAS

- With a DBaaS system, we can still connect to the database directly to perform database operations via a remote connection
- Since we usually don’t have access to the Operation System (Often no SSH access), how do we perform administrative tasks?
  - Most DBaaS solutions offer an API for administrative tasks

DBAAS API CONTROL

- The DBaaS API will usually allow you to create, modify, destroy, stop, start, and resize database instances
- This is done via a standard HTTPS web API, allowing your application to use standard web requests to manage your DBMS instances
- There’s a security element to this (requests require a secret key), so not just anyone can make these requests

A CAVEAT ABOUT THE TERMINOLOGY

- As with most things at the forefront of technology, the term DBaaS is used pretty loosely (see: “the cloud”)
- Some services describe themselves as DBaaS when they’re more of a preconfigured, but still unmanaged virtual machine
- Some DBaaS services don’t offer APIs, and simply offer standalone database instances without administrative functions
- The general idea however, is that *DBaaS systems use a preconfigured instance optimized for databases without requiring configuration by the client.*

PROS AND CONS OF DBAAS SYSTEMS

BENEFITS OF TRADITIONAL DATABASE SERVERS

- There are some benefits to using a traditional database architecture:
  - You have complete control over the system – operating system, dependencies, etc
  - If you already have a server or VM for your application, it’s often pretty trivial to spin up a database on the server

LONG STORY SHORT

- For small applications, or applications where the database size isn’t going to fluctuate greatly and is small in size, traditional database systems are fine
- There’s not necessarily a point to making things more complicated than needed if you just want a MySQL database for your application

DRAWBACKS TO A TRADITIONAL ARCHITECTURE

- There are a lot of drawbacks though:
  - Hardware or VM rental costs ($)
  - Software licensing costs (Paid enterprise-level DBMS products)
  - Requirement that you perform maintenance and patches as required
  - Requires monitoring to maintain uptime, as well as a good stable host server or VM
  - Depending on the configuration, can become complicated quickly (clusters, running multiple databases at once on the same machine, handling remote access securely)

WHERE DBAAS SHINES

- DBaaS systems really shine in providing enterprise-level databases at a fraction of the cost
  - Fees based on instance size and usage
  - No need to pay for hardware
  - Software licensing fees usually not required, or wrapped into usage fees
  - Is optimized for database usage and doesn't require regular maintenance or upkeep, is monitored by the provider

DBAAS SCALABILITY

- DBaaS systems are also very preferable if we need a *scalable* system
- Database instances can be quickly resized to accommodate larger or smaller databases
- This is generally a lot smoother than trying to resize a physical server, or resizing a standalone virtual machine (inherent OS issues, and potentially requiring that the database be taken down during resizing)

ADDITIONAL VALUE-ADDED BENEFITS

- Depending on the DBaaS you use, there may be additional benefits:
  - Pre-baked backup systems that allow you to take database snapshots and perform recoveries quickly
  - HTTP APIs that simplify application management of the database
  - Access to many different database instance types, with a common API (for example, quickly spinning up MySQL, Oracle, and PostgreSQL instances via HTTP)

DRAWBACKS TO DBAAS SYSTEMS

- DBaaS systems are of course, not perfect:
  - Some control is ceded to the database provider; you lack access to the operating system and dependencies
  - Problems on the provider's end can jeopardize your database (both in terms of uptime, and security)
  - Costs can rack up quickly if you have to resize to larger instances

SOMETHING ELSE TO CONSIDER: DATA LAWS

- With more legislation being passed to protect data privacy and security, DBaaS solutions may not be the best way to handle some data
- Sensitive data like medical records may be legally required to be stored in-country
  - Can you guarantee your DBaaS instance will be hosted in Canada?
- Putting such data on cloud servers would be a massive liability – in these cases, local datacenters or private servers would be a much better option for safeguarding the data

SO, WHEN TO USE A DBAAS?

- A DBaaS system makes sense if you need a rapidly scalable database system, or the ability to quickly create and destroy database instances
- It also makes sense if you want to minimize the maintenance and monitoring of your database, in particular if you’re running many instances or using clusters
- Generally, a good option for businesses that want enterprise database capabilities but can’t justify dedicated local hardware
  - Popular with start-ups, where operating capital is constrained and the ability to quickly wind down database servers is valuable if the business fails

DBAAS FOR OTHER APPLICATIONS

- So far, we've been approaching this from the point of view of a web application
- However, DBaaS services offer great potential for desktop applications
- Allows you to completely eliminate a standalone server, and have your desktop application directly interface with the DBaaS instance
- This also avoids the bottleneck of using a single application-database connection by giving each user their own connection

AMAZON RDS PREVIEW

AMAZON RELATION DATABASE SERVICE

- The Amazon Relational Database Service (RDS) is a relational database DBaaS provided by Amazon, as a companion product to their AWS virtual server service
- RDS is a general database platform that offers many common relational database engines

SUPPORTED DBMS'S

- RDS offers the following supported databases:
  - MySQL
  - MariaDB
  - MS SQL
  - Oracle DB
  - PostgreSQL

RDS FEATURES

- Amazon RDS offers a scalable database instances with a common HTTP API for management
- Automated backups and replication
- Built-in monitoring
- Support for replication as both a means of backup, and dealing with increased traffic (load can be distributed across replicas)

RDS SIZING AND PRICING

- Amazon RDS offers instances ranging in size from single-core micro machines with 1 GB of memory, up to 64-core machines with massive memory (500 GB) and substantial throughput
- Pricing is based on a combination of instance size, instance usage, and software license costs depending on the DBMS chosen

NEXT WEEK

- Next week we'll be diving right in to creating Amazon RDS machines, and communicating with them from an application
- We'll talk about the RDS web interface, as well as the API used to manage our DB instances, and how we can integrate it into our application

SCHEDULE

1. *DBaaS Architecture Rundown*
2. *Benefits and Drawbacks of DBaaS Architecture*
3. *A look at Amazon RDS*

SO LONG, FOLKS!