Practical Considerations

CS 40 | February 28, 2024

Agenda

1. Which Cloud Provider?
2. x86 vs ARM
3. Web App Architectures
4. Cross Region/Hybrid Deployments

Which Cloud Provider?

http://tinyurl.com/2024cloudprovidercomparison

Cloud Providers (IaaS)

- Other providers include Oracle, various GPU focused startups (Lambda Labs)
- Many FaaS/edge compute providers (e.g. Netlify, Fly.io, Cloudflare)
    - Cloudflare in particular has strong free tier services

Shared Functionality

	AWS	Azure	GCP
On Demand VMs	EC2	Virtual Machines	Compute Engine
Containers	ECS, EKS	AKS, Container Instances	Kubernetes Engine
Container Registries	ECR	Container Registry	Artifact Registry
Object Storage	S3	Blob Storage	Storage
Function as a Service	Lambda	Azure Functions	Cloud Functions

Takeaway: you probably won't miss too much general functionality

Default Heuristic

- **Definition**: When the cost of acquiring new information is high and the consequence of deviating from a default choice is low, sticking with the default will likely be the optimal choice
  - From *The Good Parts of AWS*

- Another way of looking at things: if you go with a sane default provider, most common use cases will be well-supported
  - *“Nobody ever got fired for buying $BIG_VENDOR”*

AWS

- Default option
  - If you hit a problem, you're not the first one
  - Oldest, largest, well supported

- Well regarded features:
  - Support is generally helpful
  - Generous with credits, even for smaller companies
  - DynamoDB

- Downsides: not many, but other cloud providers may be more optimized for your use case

Azure

- Strong enterprise support
  - Benefits from legacy and control of platform
  - Everything Windows is very well supported

- Well regarded features:
  - Azure Active Directory (Windows domain integration)
  - Support is helpful if you pay

- Downsides: support for smaller businesses isn't particularly strong, high reliance on legacy features

GCP

- Smallest of the big 3→ willing to offer good deals to new customers

- Well regarded features:
  - Kubernetes Engine
  - Strong educational support (especially at Stanford!)

- Downsides: (formerly) high migration fees/vendor lock-in, Google's reputation for inconsistency and removing features, poor reputation for support

General Considerations

- **Expertise**: whoever you have experience with
- **GPUs**: whoever will give them to you
- **Cost**: whoever's pricing structure fits your use case the best
- **Credits**/Migration

GPUs

- Do you really need them?
- Quantity/type
- Specific requirements (high memory, strong intranode bandwidth)
- Overall answer: whoever will sell you GPUs (probably a startup)

Cost

- *Highly nuanced!*

- Focus on largest cost centers
  - Usually network and compute, storage is generally cheaper
  - Anything can be excessively expensive if mismanaged!

- Common footgun: network egress
  - Many horror stories about this

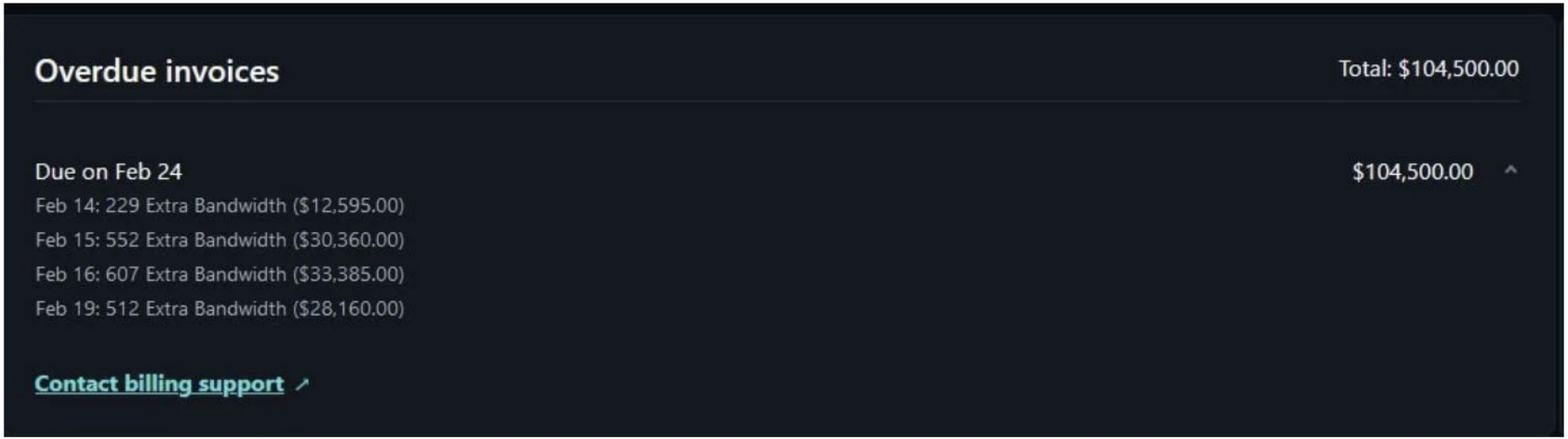
r/webdev • 2 days ago
liubanghoudai24

Monday, February 26, 2024 at 9:45:02 PM PST

Netlify just sent me a $104K bill for a simple static site

Question

So I received an email from Netlify last weekend saying that I have a $104,500.00 bill overdue. At first I thought this is a joke or some scam email but after checking my dashboard it seems like I am truly owing them 104K dollars:



That's 190TB bandwidth in 4 days

http://tinyurl.com/2024cloudegresscomparison

Credits/Migration

- Free credits are designed to get you into the platform
  - Can be very generous depending on your sales assistant
  - Don't get locked into a platform you don't want to
  - Grab all the credits you can from a platform you're committed to

- Migration
  - Easier to move from AWS to another Provider (API compatibility)
  - Harder to move from GCP to another Provider
  - Azure: many custom Microsoft specific workloads

Networking

Cloud switching just got easier: Removing data transfer fees when moving off Google Cloud

January 11, 2024

Our General Recommendations

1. Go with the provider you know best

2. Go with the provider who gives you (or your VC) the most free stuff to start

3. If you can't decide, just use AWS

x86 vs ARM

x86 vs ARM

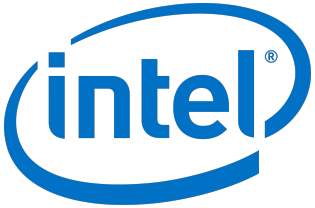
- **Instruction Set Architecture (ISA)**: A set of instructions that a processor understands and can execute

- All programs are compiled to either x86 or ARM, programs compiled on one can't run on another (without a lot of work)

- **x86**: older, generally higher performance, very well supported, *default choice*

- **ARM**: newer (but not new), lower single thread performance, well supported but not as well

x86



ARM

Amazon's Arm-based Graviton2 Against AMD and Intel: Comparing Cloud Compute

by Andrei Frumusanu on March 10, 2020 8:30 AM EST

Posted in Servers CPUs Cloud Computing Amazon AWS Neoverse N1 Graviton2

ISA Considerations

1. If your workload only supports x86, use x86
    a. Most workloads support ARM
    b. Some VM/container images only support one or the other

2. If you don't save money with ARM, use x86
    a. Depends on cloud provider – AWS usually saves money

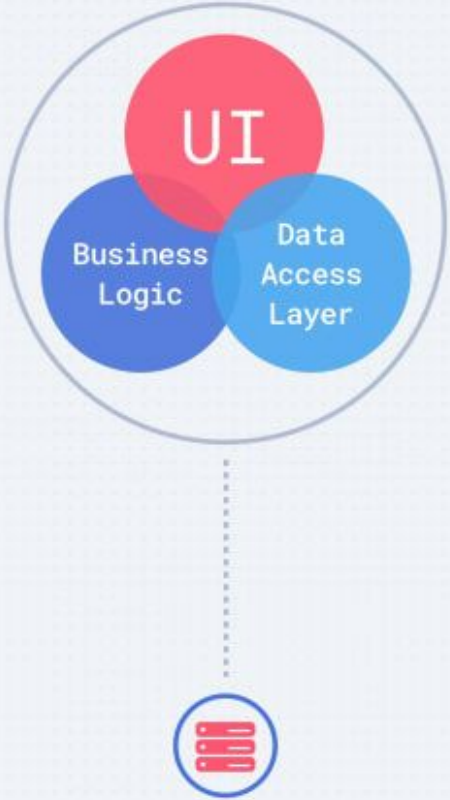
3. Otherwise, consider using ARM
    a. CS 40 has exclusively used ARM

Additional Notes on ISA

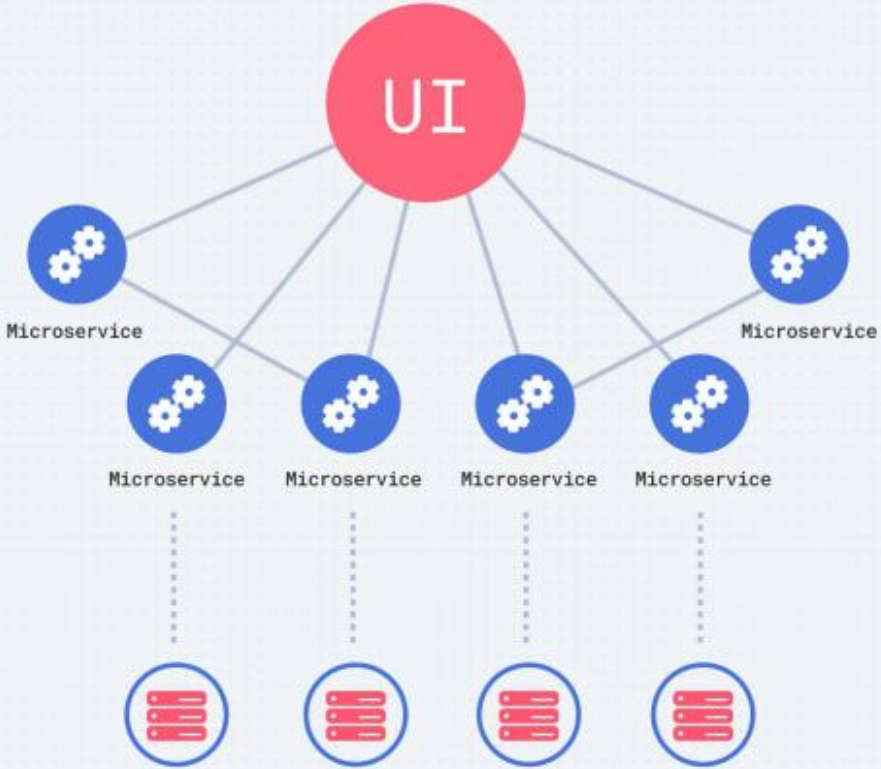
- Windows has very poor ARM support, Linux has very strong ARM support
- Most applications have an ARM64 option these days
- High level languages have very good ARM support
  - JS, TS, Python, Go, etc
  - In many cases, no need to care about ISA
- RISC-V: the super new kid on the block, not likely to be relevant in the near future

Web App Architectures

Monolithic Architecture



Microservices Architecture

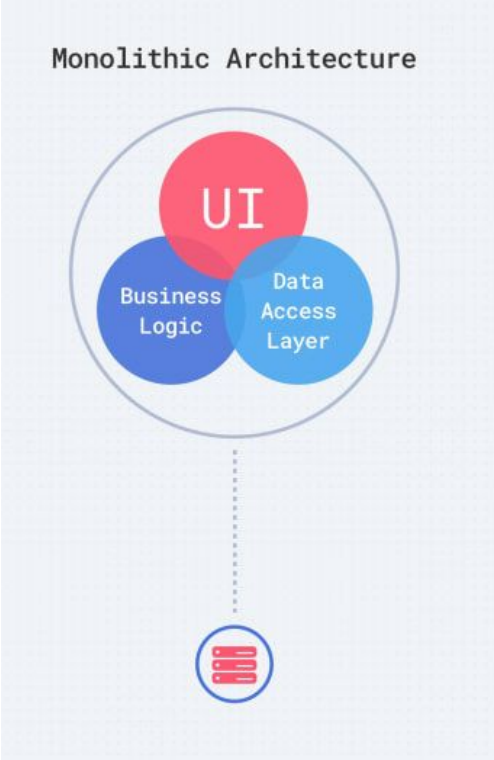


Monolith

- **Monolith**: developed, built, and deployed as a single unit
  - One backend with many endpoints
  - One (sharded) DB

- Upside: simple

- Downside:
  - Can face scaling difficulties
  - Lower fault tolerance
  - Security: single point of failure

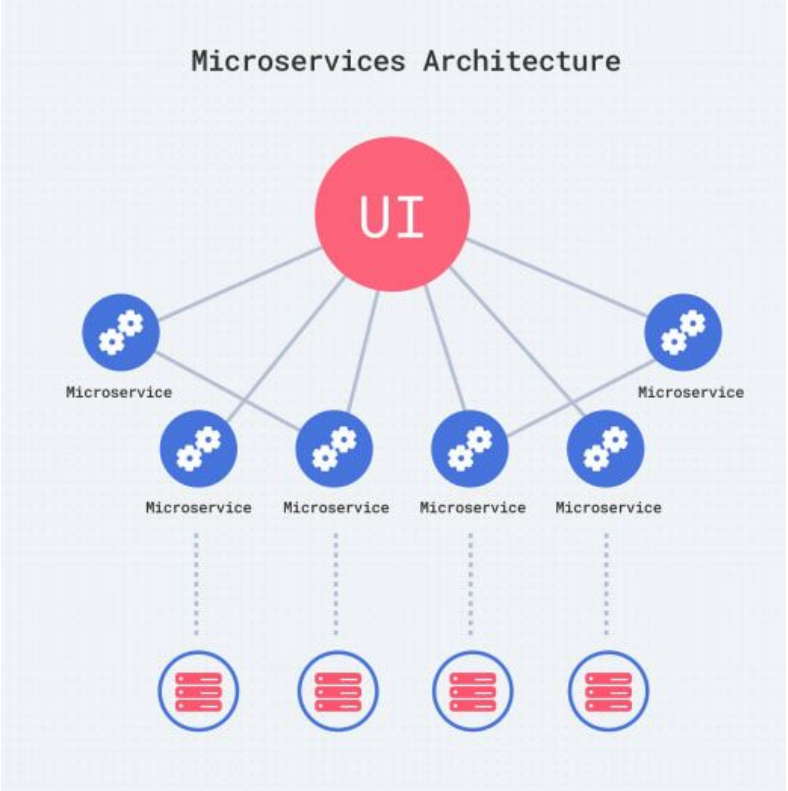


Monolith vs Microservices

- **Microservices**: composed of loosely coupled, independently deployable services
  - Each service has its own DB
  - Frontend connects to many backend servers
  - All the rage starting ~mid 2010s

- Upsides:
  - Fault tolerance
  - Scales better over many containers
  - Each team can manage own service

- Downside: *significantly* more complexity and engineering overhead



Case Study: Netflix

- Context:
  - Netflix had been slowly migrating to microservices, but mostly still a monolith
  - Successfully transitioned to AWS

- Failures:
  - 2008: entire website went down for several hours due to a missing semicolon
  - 2011: major failures in us-east force Netflix to manually move services to other regions
  - 2012: Christmas Eve, failures in ELBs prevented many users from reaching backend servers
    - Catalyzed their transition to microservices

- Fast growth made limitations of monolithic architectures clear

Our General Recommendation

- Start off by building your web app as a monolith internally composed of services
  - Keep different functionality in different files/directories
  - Avoid spaghetti code – internal services should remain logically separate

- Break off services into separate components if the limitations of monoliths prevent effective scaling

Cross Region/Hybrid Deployments

Major Pieces of Regulation

- **GDPR**: General Data Protection Regulation in EU
  - If selling to EU, all data collected about EU citizens must remain in EU
  - Very large piece of regulation with wide ranging consequences
  - e.g., may need to establish a new AWS region in EU

- **FedRAMP**: US Federal Government regulation
  - If selling to US government, entire stack storing data must be FedRAMP certified
  - Including application code and cloud provider
  - e.g., may need to use AWS **GovCloud** – a partition specifically for US government

Cross Region Deployments

- Regulatory compliance

- Higher reliability
  - Failures can occasionally span availability zones
  - Multi region presence can reduce risk

- Speed: keep application server closer to customers
  - Works in conjunction with a CDN

- Difficulty: Many IaC tools can only deploy to one region at a time, cross region deploys must be managed manually

General Concepts Behind Hybrid Deployments

- Cloud:
  - Expensive, provider must make a profit
  - Elastic, can scale up and down easily

- On-premise:
  - High upfront costs, but potentially cheaper in the long term
  - Inelastic, cannot scale physical infrastructure easily
  - *This observation is what created AWS, for Amazon to sell excess capacity on their servers*

Benefits of Hybrid Deployments

- Lower costs
  - Maintain a baseline capacity on prem to support non burst workloads
  - Buy cloud capacity when baseline capacity is exceeded

- Data sovereignty
  - Common use case is to keep sensitive customer data/IP entirely on prem
  - Regulatory compliance

Risks of Hybrid Deployments

- Complexity
  - More points of failure
  - Need to consider on prem challenges
  - Security risks posed by complex architecture

- Regulatory
  - Data movement between cloud and local datacenter may violate data transfer laws

- Cost: can potentially increase if deployment is mismanaged