Bulletproofing BTCPay Server (.dll)

BTCPay Server Day 2019 - September 16, 2019

Who I am

twitter/telegram/keybase: @ketominer

GPG: B5F6 FEBB 4D88 0398 7C60
3E3F EAD1 75FA A6C4 B7DE

ketominer@nodl.it

- “ketominer”
  - electronics
  - (low level) code
  - networks
  - systems
  - putting weird things together and making them work
- Doing the **nodl box**, **nodl hosted services** and **host4coins**

What do we want to run?

- The “full stack” (as of now)
  - a bitcoin full archiving node
  - a lightning node
  - a mixer
  - a payment server
  - a wallet backend

In the previous episode...

- Factoring and making it redundant at the datacenter level:
  - Factor what can be factored
  - Provide reasonable redundancy
  - Keep it simple

	Factoring	Redundancy
bitcoind	YES	YES
LND (or other)	NO	Kinda
btcpayserver.dll	YES	YES
other stuff (web server, load balancer, …)	YES	YES

bitcoind

- pointless to run many full archiving nodes in a single network range
- one (or two, max) node for every range and/or ASN

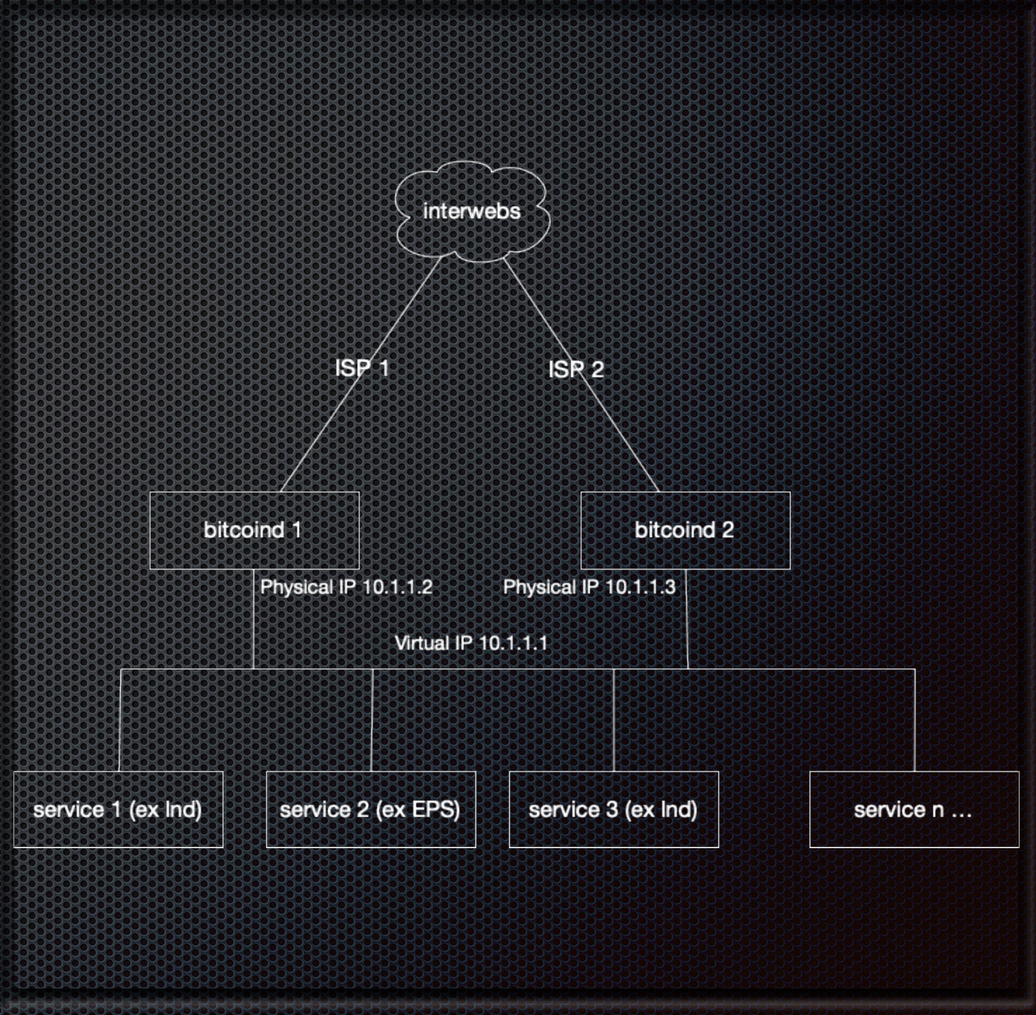
BTCPay Server

- BTCPay Server can’t be multitenant for lightning, but BTCPay Server can*
- Running one btcpayserver.dll, multiple LNDs (one per merchant/store)

*see what I did here? Please disambiguate the name!

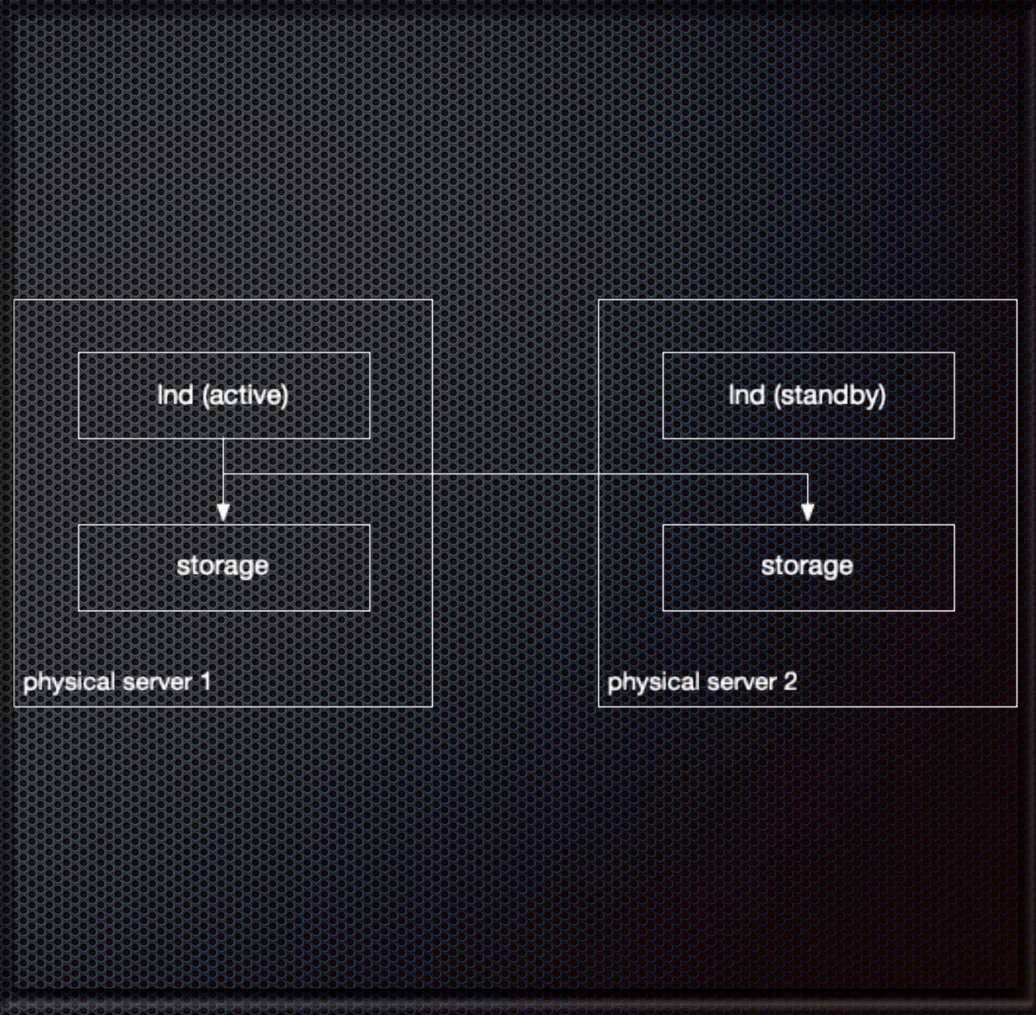
bitcoind

- run 2 (or more)
- expose RPC and ZMQ over shared VIP (Virtual IP)
- run them on separate public networks (AS) to make attacks (DDoS) harder



LND

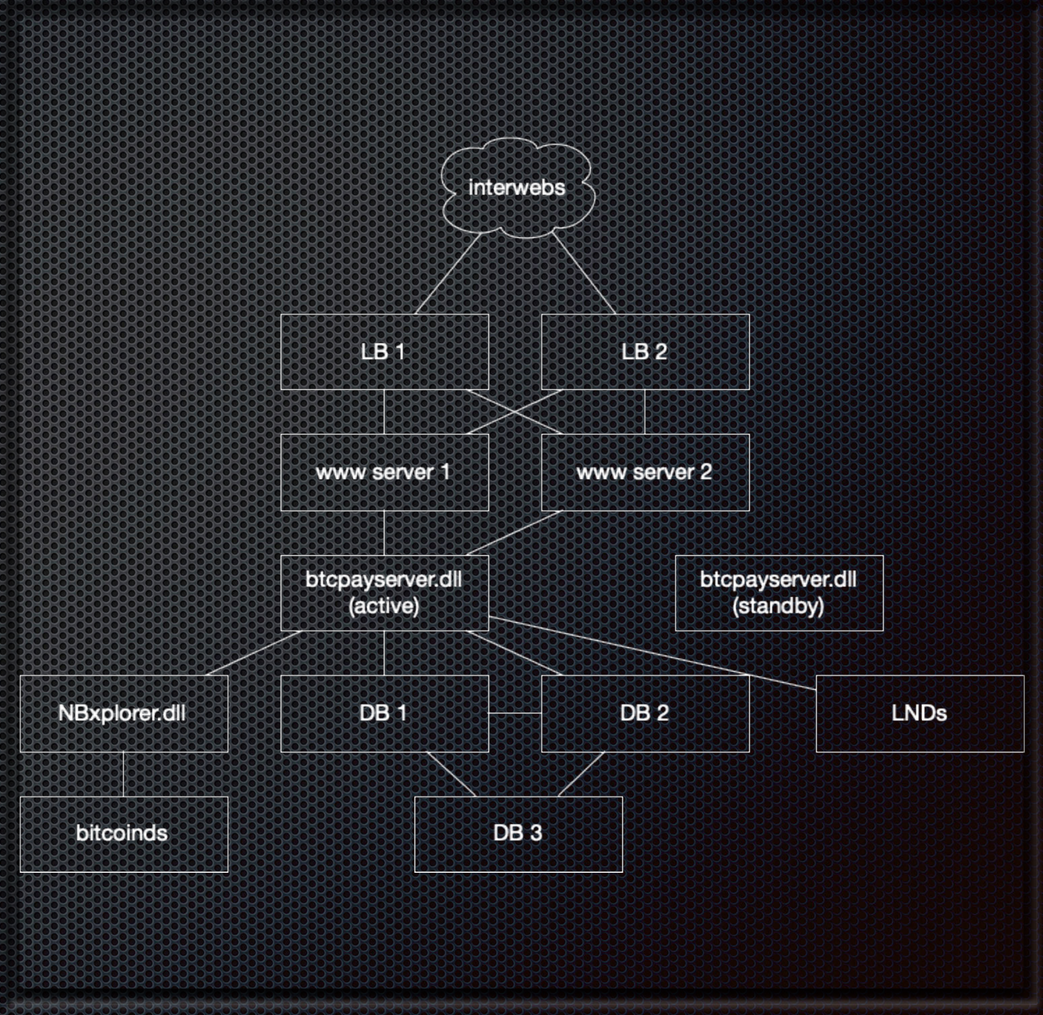
- backup / restore -> not ideal (closing channels)
- store .lnd on a distributed storage (ceph, glusterFS, DRBD, …)
- hope that the data will not be corrupted by crash of active instance
- restart another LND from same storage



BTCPay Server*

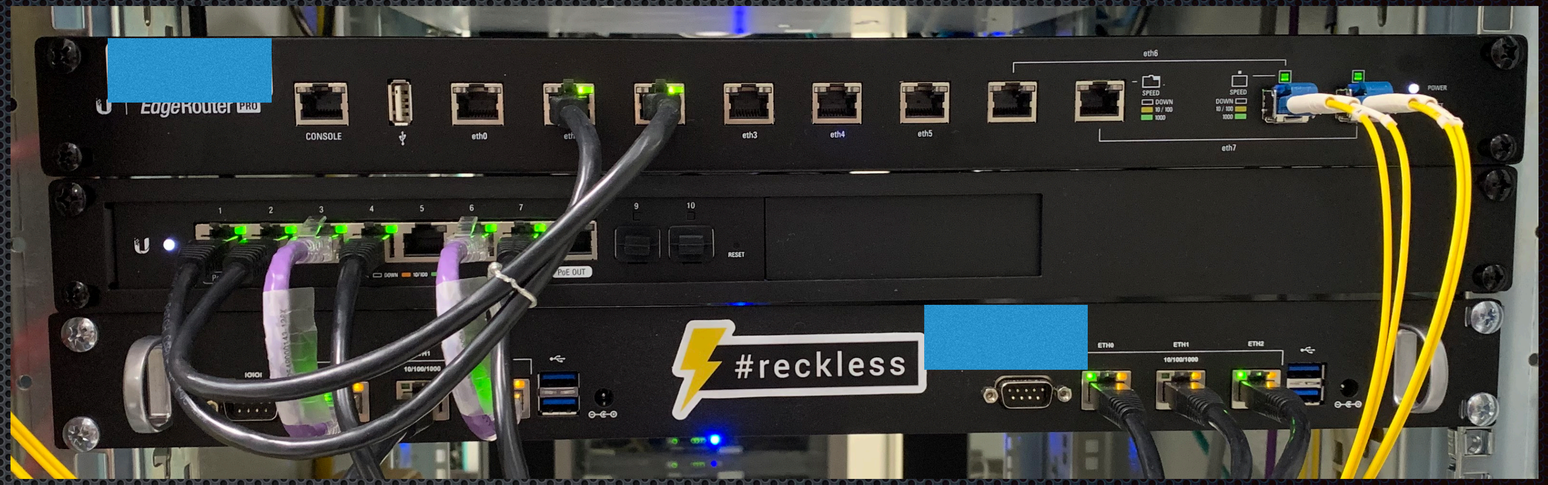
- Classic multi-tier web application
  - load balancer(s) (active/backup)
  - web server(s) (active/active)
  - app server(s) (active/backup)
  - middlware server(s) (TBD)
  - database server (cluster)

*please disambiguate the name!



Minimal setup (also for SMB)

- BGP Router
- Switch
- nodl “rack dual”
- two x86 based servers in one rack



- Runs 2 bitcoind’s, up to 10 LNDs, BTCPay Server, a Galera database cluster, and all the little stuff around

(actual picture of the nodl cloud infrastructure)

Go big or go home

- multi 10Gbps fabric
- lots of cores
- lots or RAM
- lots of SSD

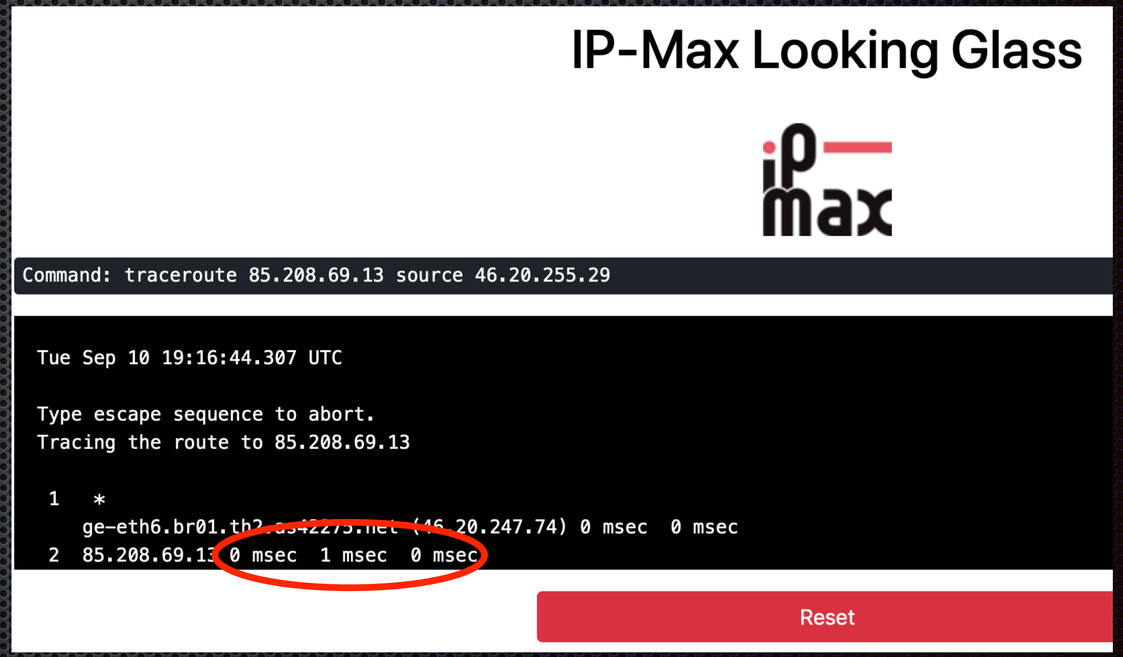
- scales up to thousands of lightning nodes

(actual picture of the nodl cloud infrastructure)

Also in the previous episode…

- Going anycast for bitcoind
  - 85.208.69.0/24 -> AS42275's anycast range (for now)
  - Same IP range announced from two locations: Geneva and Paris
  - Other nodes connect indifferently to any of the nodes (usually the closest, from a network point of view)
  - A node is announced (BGP) only if it's up and running (otherwise traffic goes to the next closest node)
  - Actually, there may be multiple bitcoinds running behind this IP in each datacenter
  - The nodes are inter-connected through private links (not Internet)
  - The 8.8.8.8 (or 9.9.9.9, or 1.1.1.1) of bitcoin (except the cool IP = $$)

85.208.69.13



traceroute from Geneva

traceroute from Paris

Average latency from Geneva to Paris = ~9ms

PoC||GTFO


```
tcp        0      0 85.208.68.13:40486      95.216.243.70:8333        ESTABLISHED 1000       659006747  10020/bitcoind
tcp        0     86 85.208.69.13:8333       172.58.110.210:53600      LAST_ACK    65534      0          -
tcp        0      0 85.208.68.13:46616      23.111.187.238:8333       ESTABLISHED 1000       659019003  10020/bitcoind
tcp        0      0 85.208.68.13:51748      178.63.25.155:8333        ESTABLISHED 1000       659024136  10020/bitcoind
tcp        0      0 85.208.69.13:8333       163.172.34.50:41768       ESTABLISHED 1000       676162738  10020/bitcoind
tcp        0      0 85.208.69.13:8333       163.172.101.59:37716      ESTABLISHED 1000       675898367  10020/bitcoind
tcp        0      0 85.208.69.13:8333       163.172.24.113:47466      ESTABLISHED 1000       675901441  10020/bitcoind
tcp        0      0 85.208.69.13:8333       195.37.209.62:49608       ESTABLISHED 1000       675532531  10020/bitcoind
tcp        0      0 85.208.69.13:8333       18.194.48.223:39207       ESTABLISHED 1000       675485985  10020/bitcoind
tcp        0      0 85.208.69.13:8333       88.99.167.186:7907        ESTABLISHED 1000       659045409  10020/bitcoind
tcp        0      0 85.208.68.13:60362      85.190.0.5:8333           ESTABLISHED 1000       659125409  10020/bitcoind
tcp        0      0 85.208.69.13:8333       162.218.65.60:17324       ESTABLISHED 1000       676041312  10020/bitcoind
tcp        0      0 85.208.69.13:8333       163.172.100.107:59158     ESTABLISHED 1000       676270858  10020/bitcoind
tcp        0      0 85.208.69.13:8333       163.172.101.194:51288     ESTABLISHED 1000       675898368  10020/bitcoind
```

Connections to/from other nodes in GVA and PAR

Outgoing uses physical range (85.208.70/24 for GVA, 85.208.68/24 for PAR)

Future expansion

- Anycast requires as many direct peerings with other ISPs as possible
- We have SwissIX and FranceIX
- 2019Q4 - Adding Frankfurt (and DE-CIX - biggest exchange in the world)
  - thus covering 50+% of global ISPs and making a full triangle
- Later adding Moscow and NYC (or SF) for better latency and resiliency

Big thank you

Fred and IP-Max for sponsoring space and connectivity for this POC

In this episode...

- Going one step further
- Geographic redundancy for btcpayserver.dll

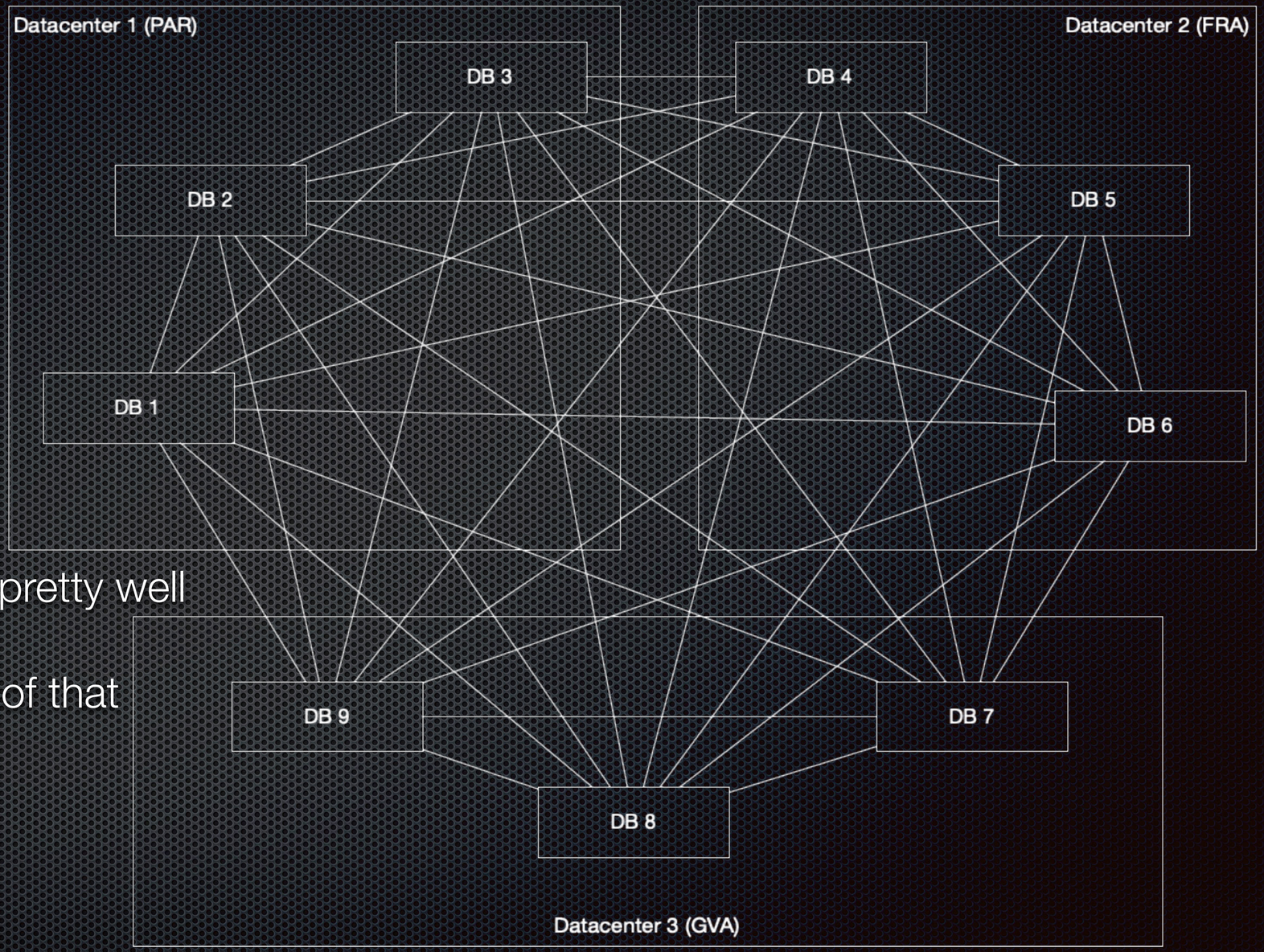
Enter Galera

- Master/Slave and Multi-Master replication sucks
- Galera provides “real” clustering functionality to MySQL engines (MySQL, Percona, MariaDB, …)
- We focus on MariaDB (fork of My by the original creator - My and Maria are the names of his daughters)
- Writes are (optionally) synchronous, whatever the size and distance is
- Large commits are (very) impacted by latency
- Actually used by banks in a worldwide synchronous setup

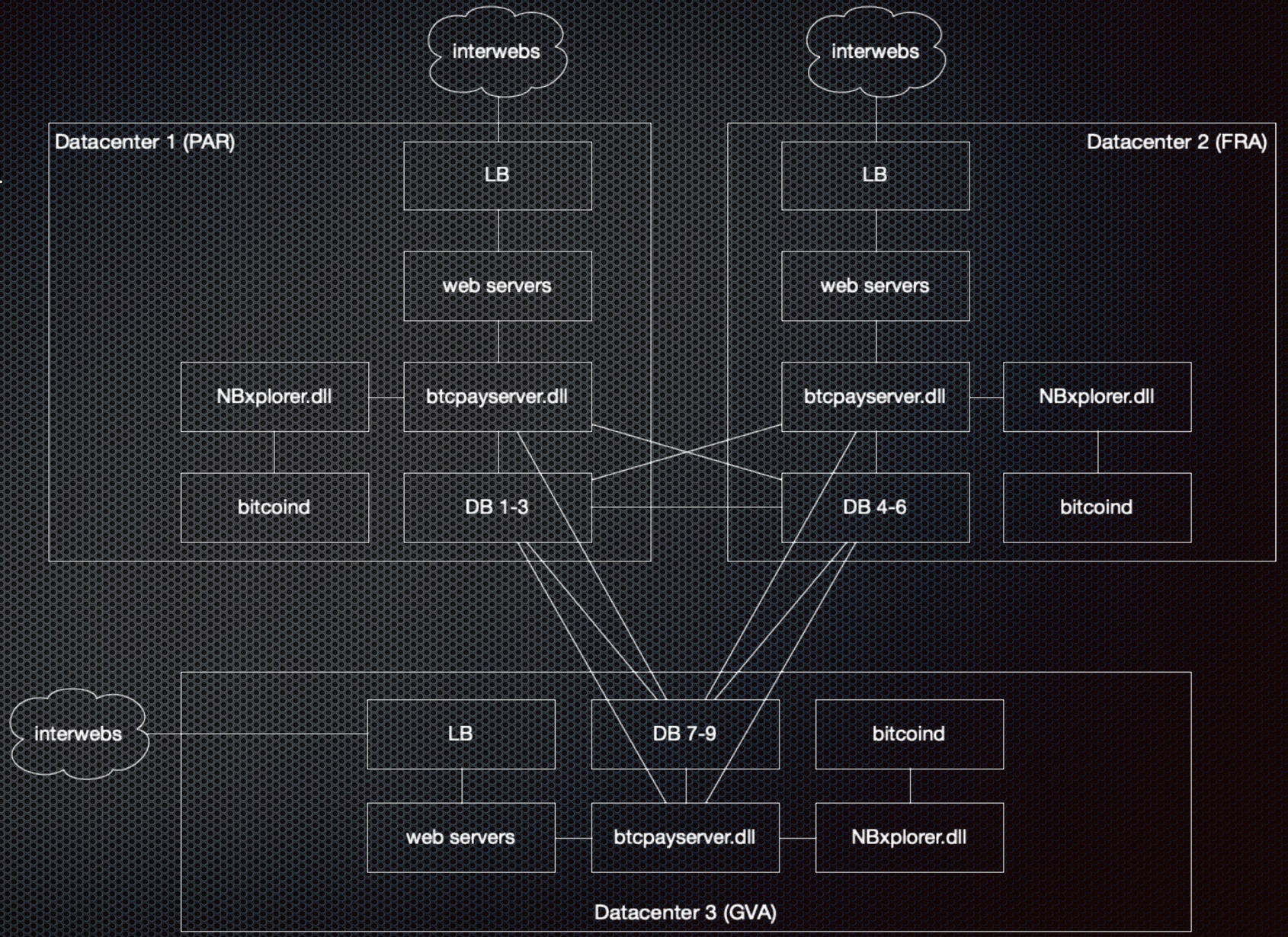
Assumptions

- Database operation latency is marginal compared to crypto calculations
- Payment operations are small (from a binlog point of view)
- Payment operations are independent from each other (two payments don’t affect each other, can happen on different locations without interfering - no dead locks in DB)
- We prefer data safety to speed (=~ write latency)
- Running the cluster with wsrep_causal_reads=1

- Fully synchronous database
- max(latency) ~30ms
- up to 200-300 is manageable for a WW cluster
- can write on any node
- can read on any node
- handles splits between datacenters pretty well
- still need to build our service on top of that
- Let’s zoom out!

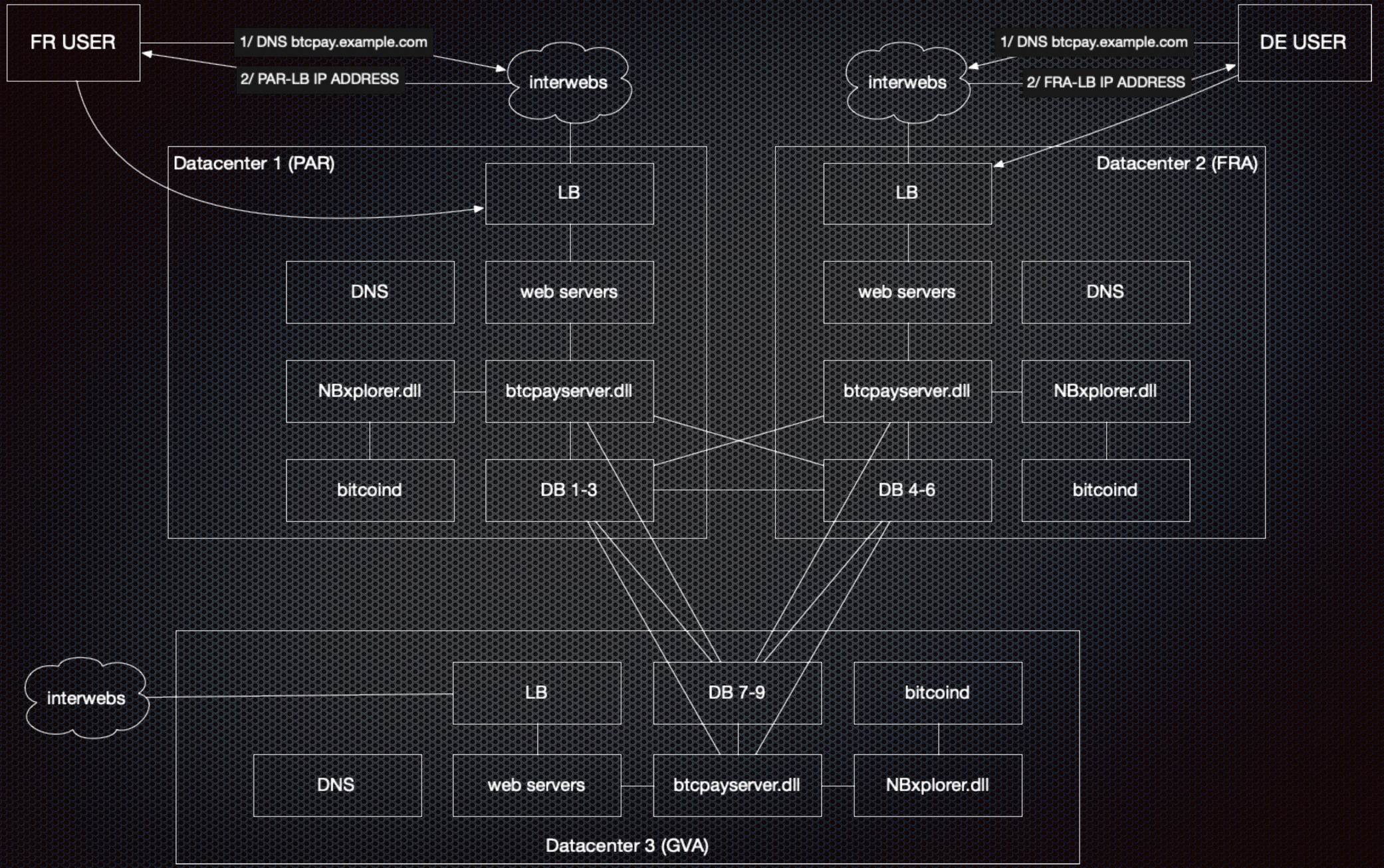


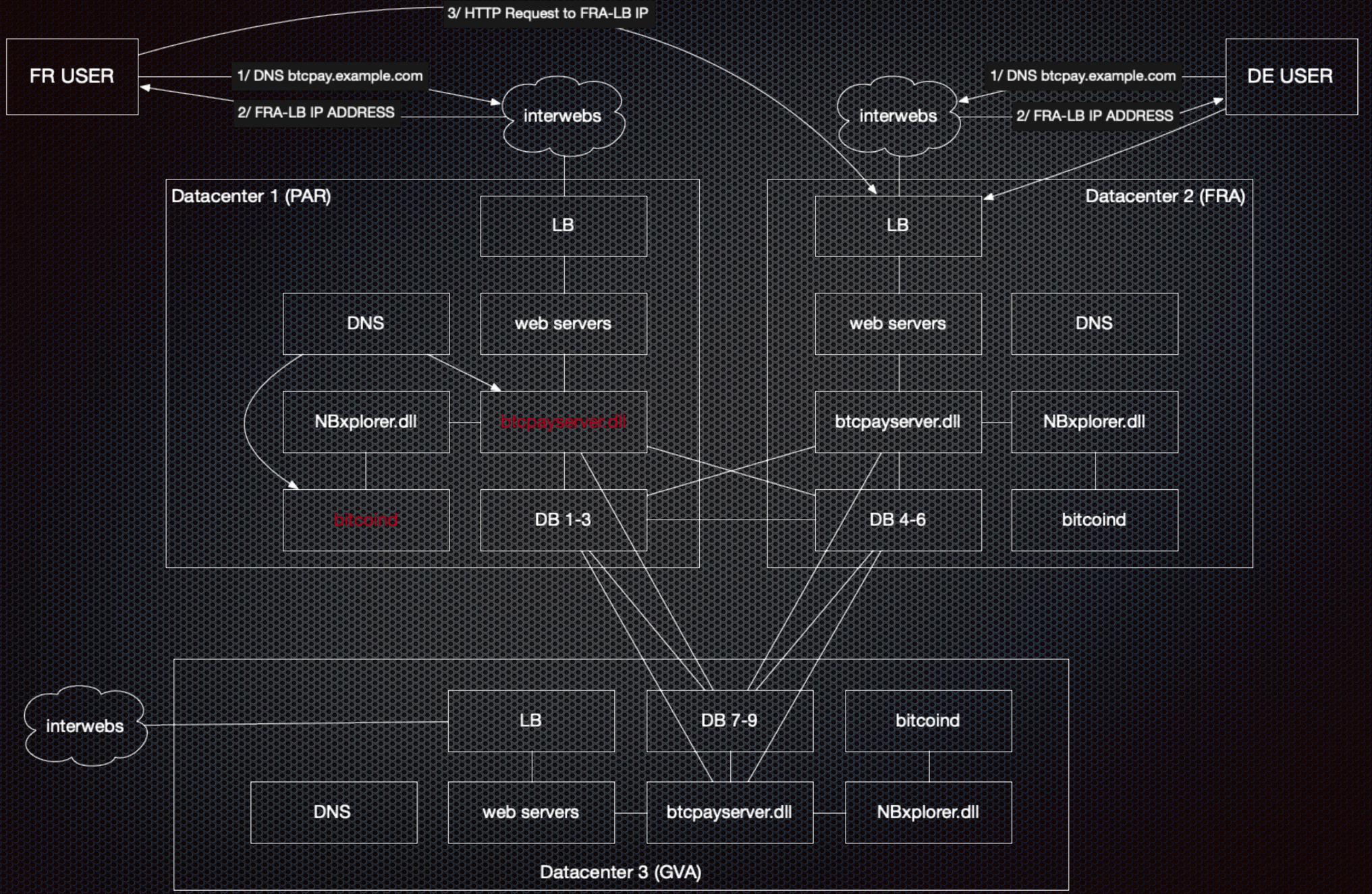
- This is great, but…
- Anycast works for bitcoind because it's not user facing, if a packet goes side-routed to another node, it will break the session and reconnect
- User facing TCP apps, especially HTTP, don't really like anycast
- Packets from a single exchange can go to two different servers
- Not great
- But one thing anycast is great for, is UDP, especially DNS



Enter… the geo-DNS server with healthchecks

- Several projects exist
- https://github.com/gdnsd/gdnsd
- Can make decisions based on the location of the user
- Can make decisions based on the health of the service
- So how does it look…


FR USER
1/ DNS btcpay.example.com
2/ PAR-LB IP ADDRESS
interwebs
DE USER
1/ DNS btcpay.example.com
2/ FRA-LB IP ADDRESS
interwebs
Datacenter 1 (PAR)
LB
DNS
web servers
NBxplorer.dll
btcpayserver.dll
bitcoind
DB 1-3
Datacenter 2 (FRA)
LB
web servers
DNS
btcpayserver.dll
NBxplorer.dll
DB 4-6
bitcoind
interwebs
LB
DB 7-9
bitcoind
DNS
web servers
btcpayserver.dll
NBxplorer.dll
Datacenter 3 (GVA)


3/ HTTP Request to FRA-LB IP
FR USER
1/ DNS btcpay.example.com
2/ FRA-LB IP ADDRESS
interwebs
1/ DNS btcpay.example.com
2/ FRA-LB IP ADDRESS
interwebs
DE USER
Datacenter 1 (PAR)
LB
DNS
web servers
NBxplorer.dll
btcpayserver.dll
bitcoind
DB 1-3
Datacenter 2 (FRA)
LB
web servers
DNS
btcpayserver.dll
NBxplorer.dll
DB 4-6
bitcoind
interwebs
LB
DB 7-9
bitcoind
DNS
web servers
btcpayserver.dll
NBxplorer.dll
Datacenter 3 (GVA)

One more thing…

Meltdown, Spectre, and the future of hardware vulnerabilities

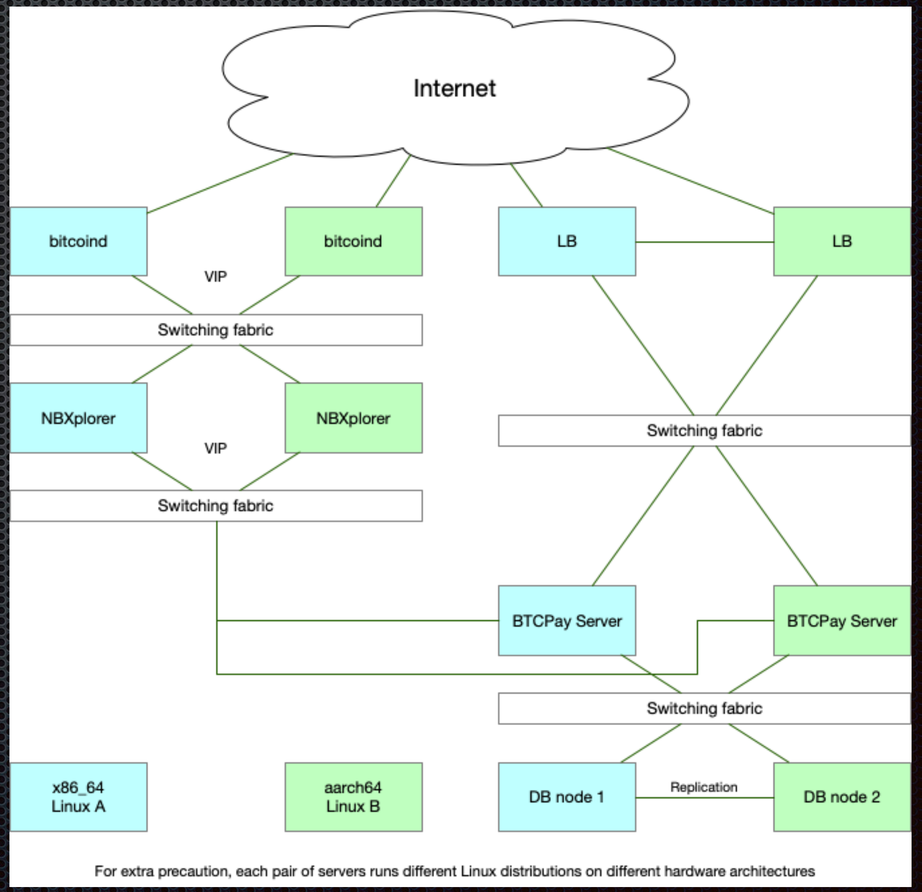
- Recent history proved that no one is safe
- TL;DR increasing the complexity in modern processors created vulnerabilities due to the architecture itself
- “quick summary of affected CPUs”
- Source: techarp.com

Here is a quick summary of the number of CPUs vulnerable to Meltdown or Spectre, according to the company, and the type of processor.

Company	Spectre 1	Spectre 2	Meltdown
AMD	295 Server CPUs 42 Workstation CPUs 396 Desktop CPUs 208 Mobile CPUs	295 Server CPUs 42 Workstation CPUs 396 Desktop CPUs 208 Mobile CPUs	None
Apple	13 Mobile SoCs	13 Mobile SoCs	13 Mobile SoCs
ARM	10 Mobile CPUs 3 Server SoCs	10 Mobile CPUs 3 Server SoCs	4 Mobile CPUs 3 Server SoCs
IBM	5 z/Architecture CPUs 10 POWER CPUs	5 z/Architecture CPUs 10 POWER CPUs	5 z/Architecture CPUs 10 POWER CPUs
Intel	733 Server / Workstation CPUs 443 Desktop CPUs 584 Mobile CPUs 51 Mobile SoCs	733 Server / Workstation CPUs 443 Desktop CPUs 584 Mobile CPUs 51 Mobile SoCs	733 Server / Workstation CPUs 443 Desktop CPUs 584 Mobile CPUs 51 Mobile SoCs
VIA	10 Desktop CPUs 12 Mobile CPUs	10 Desktop CPUs 12 Mobile CPUs	10 Desktop CPUs 12 Mobile CPUs
Total	**2816 CPUs**	**2816 CPUs**	**1868 CPUs**

Seems hard to mitigate, let's minimize the risks… an old (1 year!) idea

- Drafted a while ago, yet to apply (charter customer anyone?)
- Dual arch seems not enough now
- Intel + AMD + ARM could at least partly answer the idea
- Mixing different platforms makes some software attacks harder (different CPUs behave different way for buffer, stack and other attacks)
- Mixing different operating systems can also prevent OS-specific vulnerabilities



One last thing…

- Additional protections apply
- WAF (should define a rule-set, websocket usage may prove tricky)
- Run separate URLs with different protections for customer-facing and merchant-facing (ex. VPN only)
- Firewall (easy for HTTPS service)
- Haproxy (the LB we are using)

haproxy defense

- Limiting the connection rate per user (DOS)
- Limiting the HTTP request rate (DDoS)
- Detecting scans (weird and invalid requests)
- Deceiving the attacker (playing dead)
- Tons of examples on haproxy.com

Q ?