Building VPNs in EC2

Practical Experiences with Virtualization on Virtualization

Brandon Black - blblack@gmail.com

Cloud Tradeoffs

- Commoditizes the "deployed hardware" layer
- "You're not a datacenter company, so don't waste time and money on datacenter ops"
- Focuses more on your core functionality
- Commodity always comes at a cost:
  - Less customization
  - Accepting a fixed set of capabilities

Basic EC2 Networking Limitations

- Limited IP Proto Support: TCP, UDP, ICMP
- Your hosts are randomly sub-netted
- Divided into a topology you can't predict or control, by routers you don't control
- 1x Private IP per host...
- Private DNS constantly in flux, no fixed scheme
- 1x Public IP per host via 1:1 NAT...

Heartbeat ?

Standard resiliency tool on real hosts, but...

- One virtual eth0 per host:
  - no redundant paths
  - unable to reliably determine failure mode
- One fixed/random private IP:
  - no ability to failover a private IP
- STONITH via node termination is dangerous.
- You could fail a public IP, but it would probably bounce around given the above.

Elastic Load Balancing! (?)

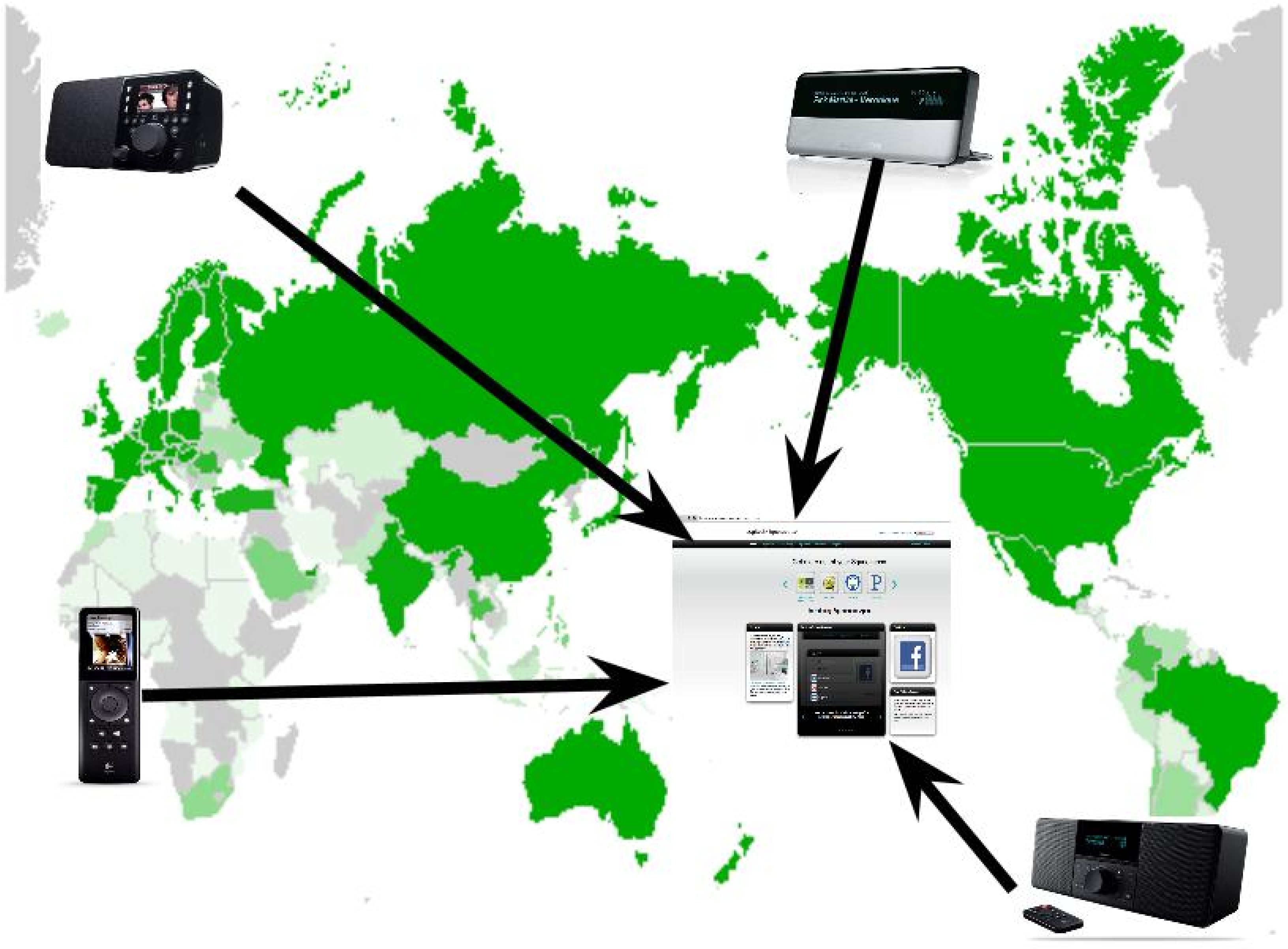
Great idea in theory, but...

- Resolves via cross-zone CNAME w/ short TTL:
  - Cannot ELB zone root
  - DNS latency increases
- Only basic HTTP/HTTPS proxied correctly
- Other TCP "works", but source IP lost
- Health check options limited/weak

MySqueezebox.Com

- Constant traffic from all over the globe
- ~160-300 Mbps public traffic (tx+rx)
- 206 Bytes Avg Packet Size
- = 100-200K pps
- ~250K concurrent TCP sessions, long-lived
- Majority is custom "SlimProto" protocol
- Protocol can't be changed (firmware limits)
- No Layer7 Proxy for it (yet)
- Infrastructure relies on Layer4 IPVS+NAT

ALL Players on SlimService - 2011-01-31 17:34:34



🟩	US: 52260
🟩	DE: 25655
🟩	CH: 9992
🟩	SE: 8318
🟩	NL: 6292
🟩	GB: 6274
🟩	NO: 5836
🟩	FR: 4108
🟩	CA: 3985
🟩	DK: 2451
🟩	BE: 1891
🟩	AT: 1176
🟩	AU: 1115
🟩	IT: 1107
🟩	FI: 877
🟩	IE: 664
🟩	ES: 509
🟩	JP: 299
🟩	CZ: 294
🟩	PL: 292
🟩	HK: 265

Migrating MySB to EC2

- Pushed by hosting/bandwidth cost/focus concerns, upcoming hardware purchases imminent due to userbase growth, but...
- Can't work on EC2 natively:
  - Existing IPVS+NAT vs EC2 network
  - Existing monitor/balance tools, etc...
- Developers cannot fundamentally change architecture to match EC2 on ops schedule
- We Need A VPN

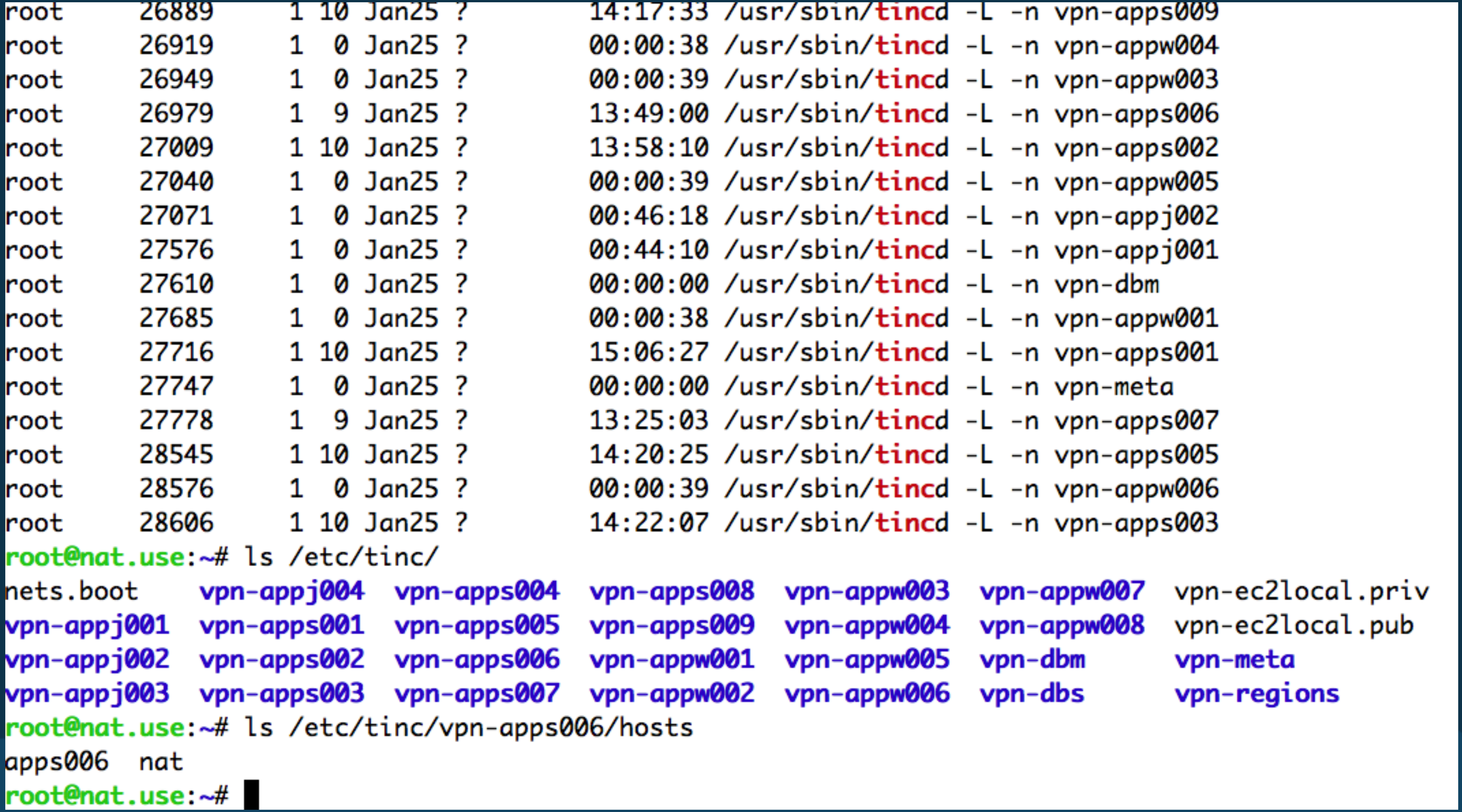
tinc!

- We chose tinc after surveying the available options:
  - Simple, well-maintained, high code quality
  - Very easy to configure/deploy [puppet]
  - Works over TCP or UDP [critical for EC2]
- TCP is a non-starter, because putting 100K+ TCP connections inside another TCP connection is a fundamentally bad idea.
- So we start testing w/ tinc on UDP.

Load Testing...

- Heavy CPU bottlenecking on Loadbalancer...
- Disabling auth/crypt helps a lot...
- Still too high: tinc is single-threaded, one CPU locks up, rest are idle.
- Split the VPN into point-to-point links

Point-to-Point VPN Setup...



```
root      26889      1 10 Jan25 ?        14:17:33 /usr/sbin/tincd -L -n vpn-apps009
root      26919      1  0 Jan25 ?        00:00:38 /usr/sbin/tincd -L -n vpn-appw004
root      26949      1  0 Jan25 ?        00:00:39 /usr/sbin/tincd -L -n vpn-appw003
root      26979      1  9 Jan25 ?        13:49:00 /usr/sbin/tincd -L -n vpn-apps006
root      27009      1 10 Jan25 ?        13:58:10 /usr/sbin/tincd -L -n vpn-apps002
root      27040      1  0 Jan25 ?        00:00:39 /usr/sbin/tincd -L -n vpn-appw005
root      27071      1  0 Jan25 ?        00:46:18 /usr/sbin/tincd -L -n vpn-appj002
root      27576      1  0 Jan25 ?        00:44:10 /usr/sbin/tincd -L -n vpn-appj001
root      27610      1  0 Jan25 ?        00:00:00 /usr/sbin/tincd -L -n vpn-dbm
root      27685      1  0 Jan25 ?        00:00:38 /usr/sbin/tincd -L -n vpn-appw001
root      27716      1 10 Jan25 ?        15:06:27 /usr/sbin/tincd -L -n vpn-apps001
root      27747      1  0 Jan25 ?        00:00:00 /usr/sbin/tincd -L -n vpn-meta
root      27778      1  9 Jan25 ?        13:25:03 /usr/sbin/tincd -L -n vpn-apps007
root      28545      1 10 Jan25 ?        14:20:25 /usr/sbin/tincd -L -n vpn-apps005
root      28576      1  0 Jan25 ?        00:00:39 /usr/sbin/tincd -L -n vpn-appw006
root      28606      1 10 Jan25 ?        14:22:07 /usr/sbin/tincd -L -n vpn-apps003
root@nat.use:~# ls /etc/tinc/
nets.boot    vpn-appj004  vpn-apps004  vpn-apps008  vpn-appw003  vpn-appw007  vpn-ec2local.priv
vpn-appj001  vpn-apps001  vpn-apps005  vpn-apps009  vpn-appw004  vpn-appw008  vpn-ec2local.pub
vpn-appj002  vpn-apps002  vpn-apps006  vpn-appw001  vpn-appw005  vpn-dbm      vpn-meta
vpn-appj003  vpn-apps003  vpn-apps007  vpn-appw002  vpn-appw006  vpn-dbs      vpn-regions
root@nat.use:~# ls /etc/tinc/vpn-apps006/hosts
apps006  nat
root@nat.use:~# 
```

More Load Testing ...

- CPUs much better now
- But VPN network is unreliable...
  - lots of TCP slowdowns
  - connect() failures
  - packet loss
  - data latency
  - poor user experience

Why?

- EC2 UDP traffic has re-ordering issues
- Frequent queue-jumping, e.g. 1 packet gets jumps ahead of 2000 others from the same socket in EC2's network queue.
- tinc uses packet seqnos to prevent replay attacks: when 1 packet jumps too far, other 2000 are discarded.

Patch it!

- We patched tinc 1.0.x to allow:
  - configurable replay window size
  - some queue-jumping resilience
  - disable replay protection via config
- And a few more tune-ables to play with:
  - configurable socket buffer sizes
  - IFF_ONE_QUEUE

Success

- Able to move all users to EC2 in time for xmas rush, VPN is stable and reliable.
- New limits approaching, architecture is a constant challenge as always, that's life.

Was it worth it?

- Yes!
- Operating expenses per user (hosting/traffic) roughly cut in half
- No more capital expenses on hardware replacement/expansion
- No more wasting smart engineers' time dealing with the physical layer all around the globe

Looking Forward...

- Today our LBs are m2.4xlarge and the VPN itself still consumes notable amounts of CPU.
- Tomorrow they could be cc1.4xlarge:
  - 10x network traffic
  - +30% CPU
- EC2 Overlay VPNs don't need security
- They should strive for being performance-transparent

Future Perf Hacks for VPN s/w?

- Security needs to be optional
- Event-driven design a must: epoll(2), kqueue(2)
- Threading for scaling over cores
- recvmmsg(2), sendmmsg(2)
  - Multiple packets per syscall/socket-lock
  - sendmmsg(2) Not Yet Implemented
- splice(2): avoiding kernel->user->kernel memcpy
- Kernel modules?