About Yammer

- Founded 2008
- Enterprise Social Network
- Acquired 2012 by Microsoft
- Typical workloads:
  - Micro services & Rails monolith
  - 90% “real-time” - web queries
  - Batch workloads runs off mostly non-production env
  - Many service-to-service dependencies
  - Large varieties of data stores
Yammer loves PostgreSQL

- Reliability and stability
- Rich feature sets (data types, extensions, replication, etc.)
- Great libraries/clients for all major languages
- Awesome community and toolings
PostgreSQL in Production

- 14 PostgreSQL clusters in Production (4-8 nodes/cluster, 2 DCs)
- Asynchronous cascading replication
- Use replica read whenever timeline consistency is accepted
- Peak ~30k RPS for masters (reads + writes) + ~25k RPS for replicas
- Runs on Fusion IOs, no SAN
- Disaster Recovery replicas for non-realtime production traffic and all analytics workload
- Configuration management using Puppet, assisted by bouncie, an in-house tool
PostgreSQL Traffic

- 100% E2E TLS
- AES SHA256 at rest
- 1.2 Gbps peak egress
Once Upon A Time ...

- Service Discovery was not a thing
- Diverged/customized configuration per cluster
- Dummy pgbouncer on each PostgreSQL node
- Clients aware of all nodes
- Losing track of cluster memberships
- Replication management is an art
- Astronomical MTTR
- Ganglia-backed metrics system
Bouncie - The PgBouncer Cluster

**Diagram Description:**

- **Postgres master** connected to **PgBouncer Master pool** via **TLS**.
- **Postgres replica** connected to **PgBouncer Replica pool** via **TLS**.
- **ZK** connected to **PgBouncer Master pool** and **PgBouncer Replica pool** with **control data**.
- **PgBouncer** reloads on change.
- **Master identification** and **Replication graph**.
- **Haproxy** with **TCP passthrough**.
- **Clients** connected to **Haproxy**.
Scaling, Partitioning and Sharding

- Master is SPOF for write
  - Know the limits (RPS, IO, storage)
  - Avoid synchronous whenever possible (instead write to a queue)

- Do not let a single cluster grow too big
  - IO/Storage constraint
  - Expensive maintenance
  - Slow replica rebuild

- Vertical partitioning may create more SPOFs
  - Availability, MTBF
  - Compound latency on critical path

- Keep it simple! (no crazy joins, layers of subqueries, views, etc.)
- “Sharding is hard”
The Tale of Caching

- Increasing reliance on caching layer (req hit rate > 90%)
  - ~400k RPS mcrouter/memcached at peak
- Inconsistency, invalidation problems
- mcrouter comes to rescue with
  - Memcached hardware failures
  - Cold cache, cross-DC cache replication
- Troubleshooting the cache is hard
  - Hot keys
  - Eviction
  - Sometimes involving TCP packet analysis
- Rely more on replica reads instead of cache!
Replication lag!

- `pg_current_xlog_location() - pg_last_xlog_replay_location() = lag in bytes`
- Lag in bytes/WAL rate → time value
Keeping PostgreSQL Happy

- Set statement_timeout on client side!
  - Most clients just walk away after timeout on their end
- Consider a watchdog service to kill long running SELECT queries
  - This saved us numerous times
- Lower lock_timeout to get out of bad locking situation quickly
- Control pool_size on PgBouncers/clients
- Tune timeouts on pgbouncer
  - Server_lifetime
  - Idle_transaction_timeout
- Separate disks/partitions for data, xlog and logs
Backups & Maintenance

● PITR is a must have
  ○ Barman (https://github.com/2ndquadrant-it/barman)
  ○ Wal-E (https://github.com/wal-e/wal-e)
  ○ Monitor your backup system
  ○ Exercise recovery runbook regularly (implement continuous recovery test)

● Logging & analysis
  ○ Pgbadger for offline query analysis (https://github.com/dalibo/pgbadger)
  ○ Pg_hero (https://github.com/ankane/pghero)
  ○ PgBouncer log to ELK for auth error detection

● Table compaction, reindexing
  ○ Pgcompact (https://github.com/grayhemp/pgtoolkit#pgcompact)
  ○ Extremely resource-intensive and risky process

● Upgrade :)
Metrics and Dashboards
The Road to Azure

- 99.99% SLA
- Automation, automation and automation
  - Provisioning of VMs, VM rotation
  - Replica failover by moving attached disks to healthy, standby VMs
  - Master failover TBD
- Bouncie improvements
  - Removing DNS dependency
  - Real-time update
  - Graceful failover/rotation
- Hashicorp Vault integration for credentials rotation
- Celling, data locality, multi-region replicas
- Patroni has a lot of potential! ([https://github.com/zalando/patroni](https://github.com/zalando/patroni))
We are Hiring!

Come join us, we have openings in both San Francisco and Redmond!

- [https://medium.com/yammer-engineering](https://medium.com/yammer-engineering)
- Our stack
  - Linux
  - Mesos, Marathon, Docker, Hashicorp Vault
  - PostgreSQL, HBase, ElasticSearch
  - Azure
- Contacts: [cnguyen@yammer-inc.com](mailto:cnguyen@yammer-inc.com)
QUESTIONS?