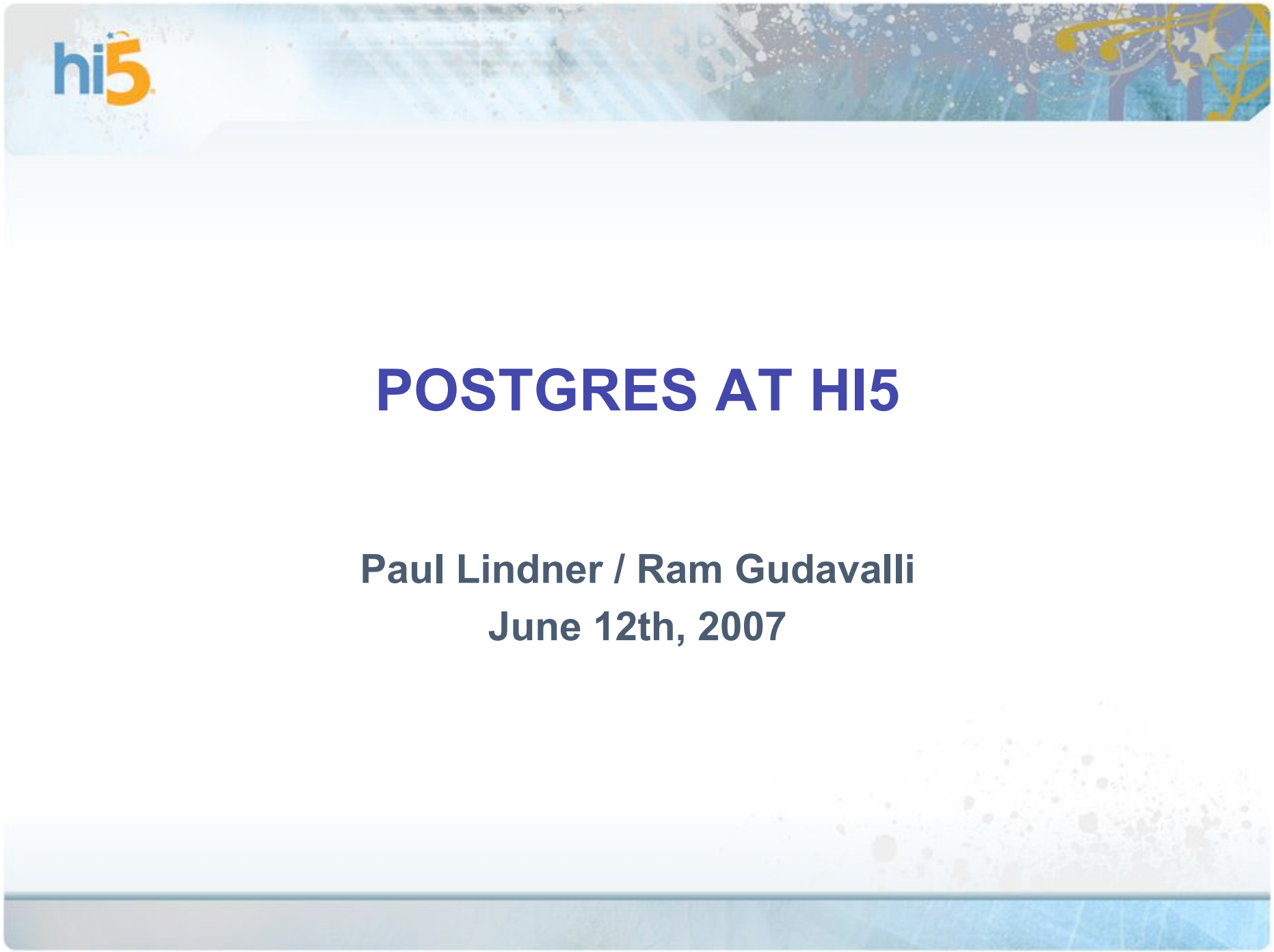


POSTGRES AT HI5

Paul Lindner / Ram Gudavalli

June 12th, 2007

WHAT IS HI5?

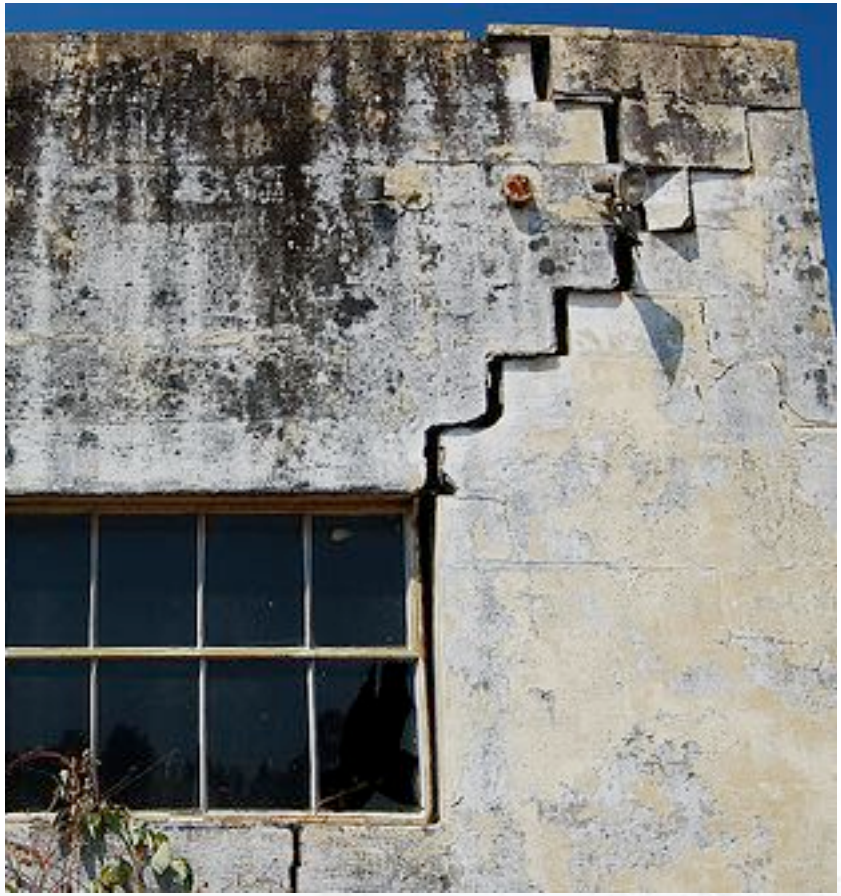
- 3rd largest social networking website worldwide behind MySpace and Orkut
- Offering Messaging, Friends, Video, Music, Groups and more.
- Large international focus
- Top 15 Website (Alexa)
- 7+ Billion Pageviews / month
- 30+ Million Active users

BEGINNINGS OF HI5

- Founded by Ramu Yalamanchi and Akash Garg
- Launched in December 2003
- Reached 1M members by July 2004
- Started as a basic N-tiered Java architecture
  - 6 application servers
  - 1 Postgres 7.4.x DB server
  - Lasted until about 6.5M registered members

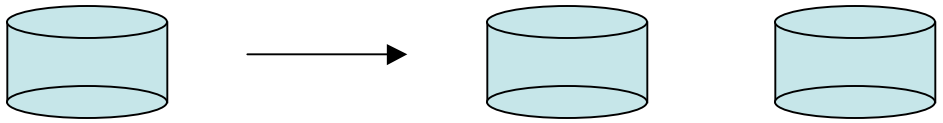
CRACKS IN THE FOUNDATION

- DB would lock up under a burst of write activity
- Inefficient schema design
- Very expensive checkpoints and vacuums for large tables



PROGRESSION OF ENLIGHTENMENT

- **Database partitioning**
  - **Schema separation across multiple databases**
  - **Provides separation of I/O characteristics by schema requirements**



Schema Users & Messages

Schema Users

Schema Messages

Problem: Tables were starting to get too large…

DB PARTITIONING (CONT.)

- Table partitioning
  - Partition a large table into many smaller tables w/ software routing of queries
    - Friends DB currently has 120 tables across 5 DB servers
  - Reduces checkpoint and vacuum times
  - Creates smaller indexes for Postgres to manage
  - Makes it much easier to shift data around when performance bottlenecks arise

GENERAL DISTRIBUTED DB ARCHITECTURE

QUERY FRAMEWORK

- Generalized software framework for distributed database querying and management
  - Routing of queries based on key mappings
  - Support for Broadcast queries, data aggregation and post-processing
  - Load-balancing support
  - Poor-Man's Replication (Non-transactional)
  - DB Lockout and failover
  - Maintenance hooks (such as the ability to set a DB to Read-Only mode and the ability to change DB layout dynamically)

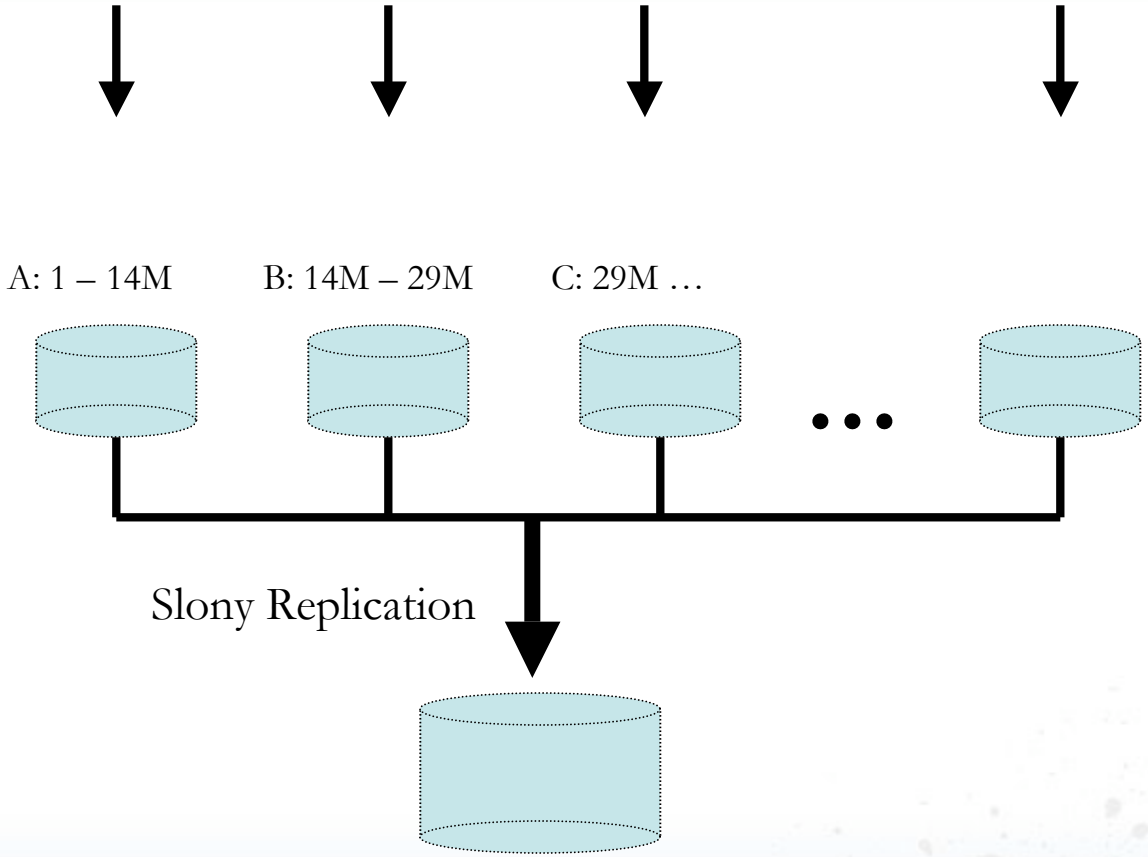
QUERY FRAMEWORK – iBatis

- All Queries are centrally maintained in XML files
  - Easy to audit, no suprises
  - Easy to optimize / verify (grep -i count *.xml !)
  - Maps relational fields to Java Beans
- Extended to support our partitioning
  - ctrl:partitionKey="id”
  - ctrl:pointCut="GetNextId(userlogin,users_seq,id)”
  - ctrl:pointCut="DeleteCache(UserLoginBean,email)”

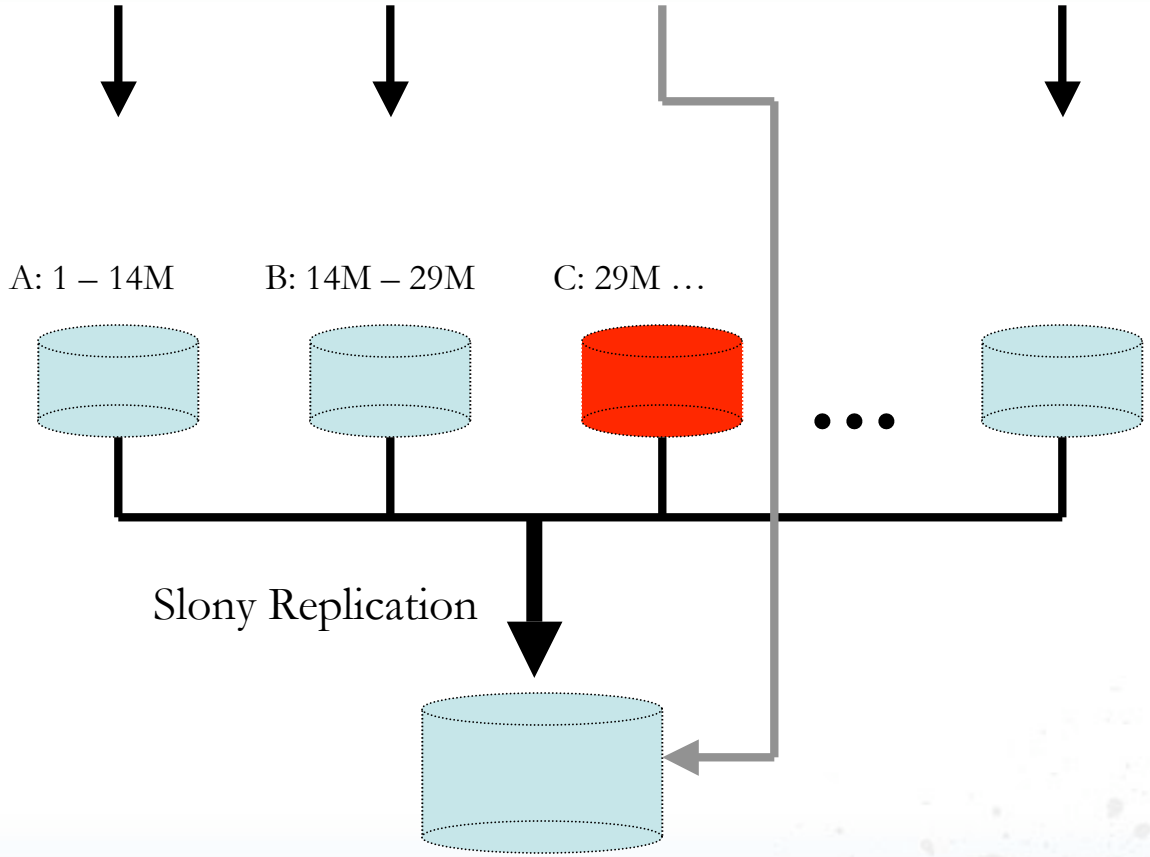
QUERY FRAMEWORK – Configs

- All DB structure defined in two files
  - db.xml (Logical Structure)
    - <tablespace name="AuthCluster">
      <Cluster>
      <DBInstance name="authdb_r0”/>
      <DBInstance name="authdb_c0" readOnly="true"/>
      </Cluster>
      <table name="userlogin"/>
      </tablespace>
  - Torque.properties (DSNs/Pools)
    - Torque.dsfactory.authdb_c0.factory=com.friend.db.RecoverableDataSourceFactory
      torque.dsfactory.authdb_c0.pool.maxActive=6
      torque.dsfactory.authdb_c0.pool.maxIdle=6
      torque.dsfactory.authdb_c0.pool.maxWait=600000
      torque.dsfactory.authdb_c0.pool.testOnBorrow=true
      torque.dsfactory.authdb_c0.pool.validationQuery=SELECT 1
      torque.dsfactory.authdb_c0.connection.driver = org.postgresql.Driver
      torque.dsfactory.authdb_c0.connection.url=
      jdbc:postgresql://10.100.9.50:5432/friend?loginTimeout=10

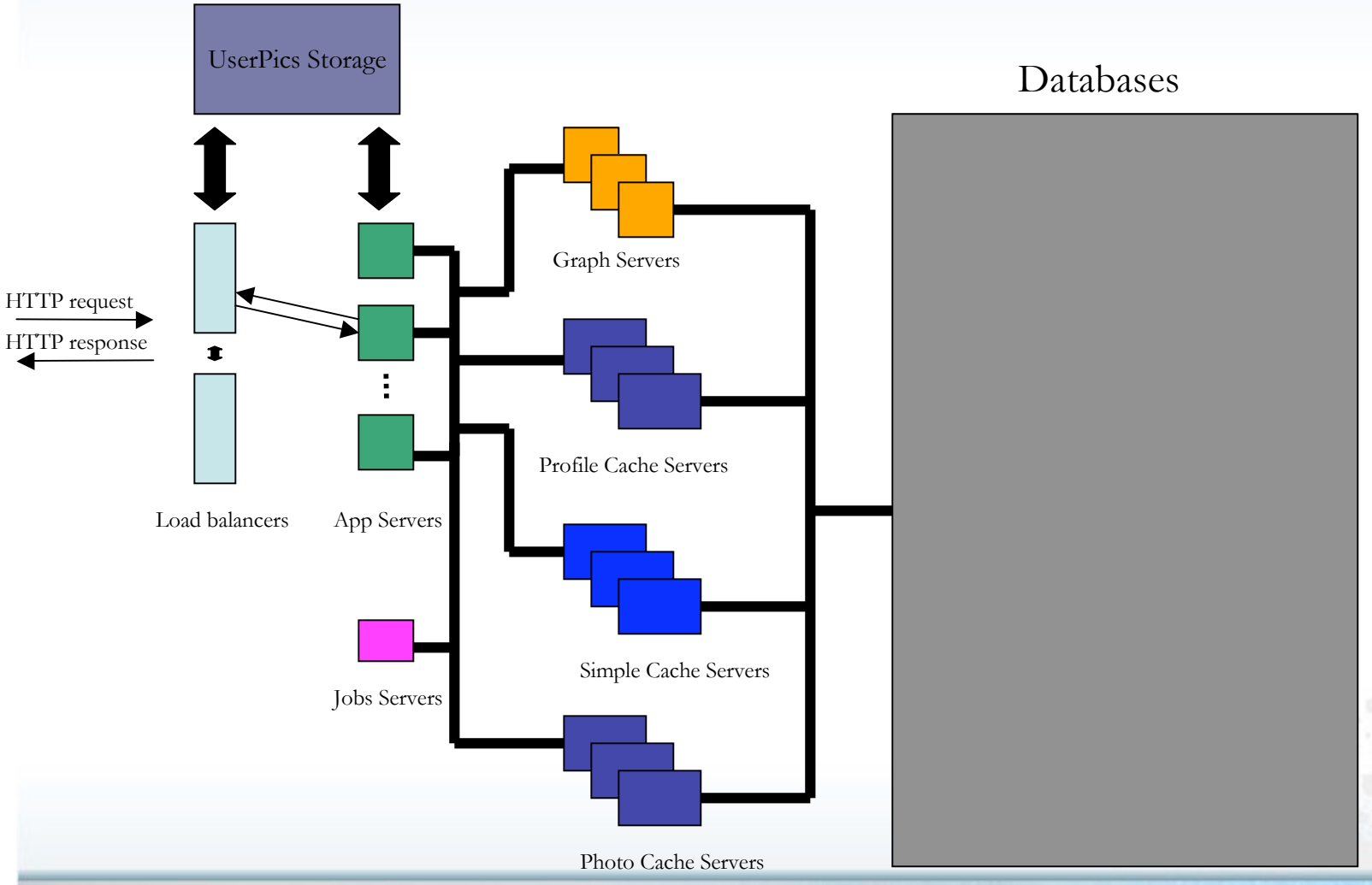
USER CLUSTER ARCHITECTURE



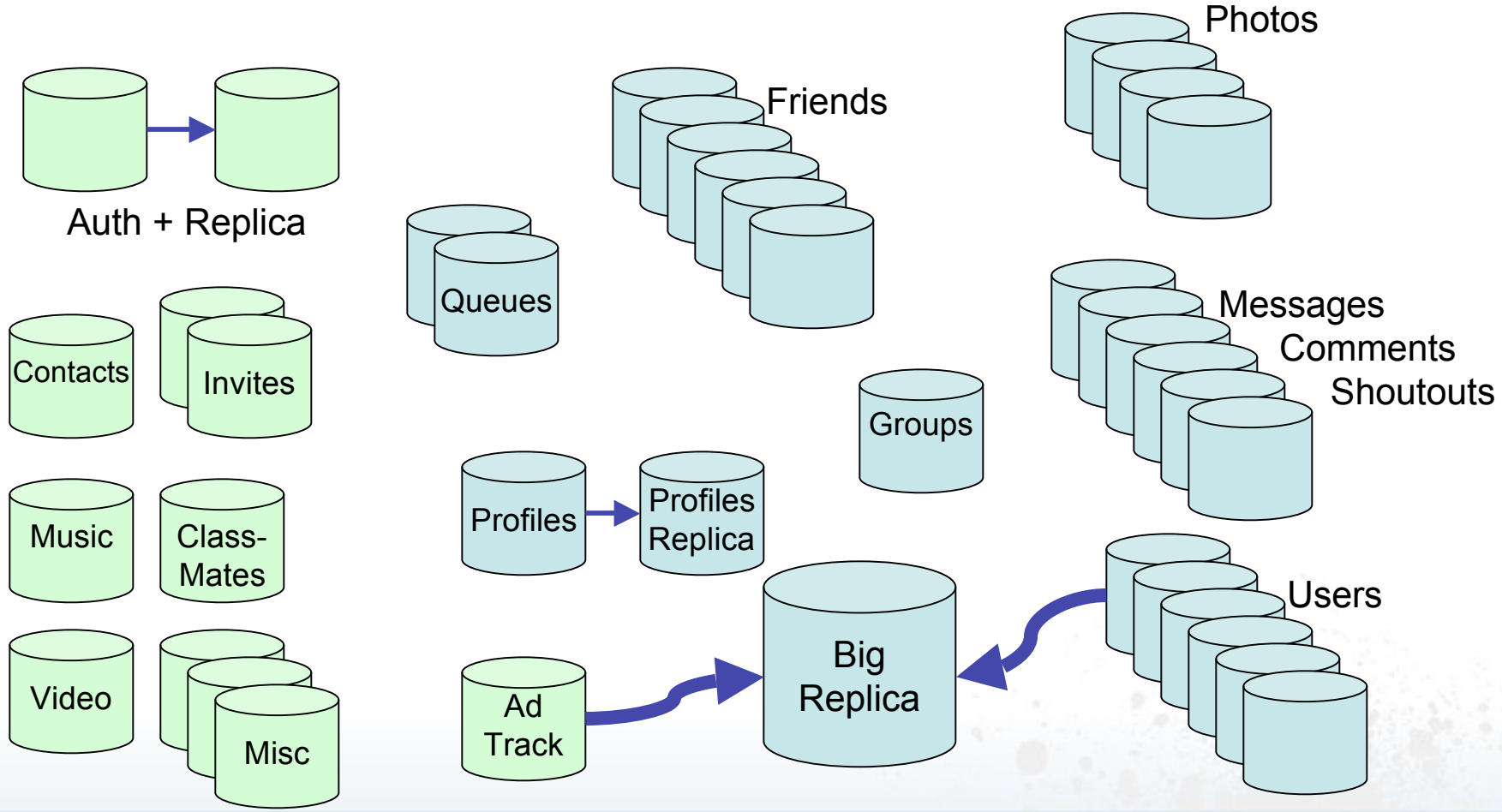
USER CLUSTER ARCHITECTURE



HI5 N-TIER ARCHITECTURE



HI5 PRODUCTION DATABASES



HARDWARE - OLD

- Meet the old Moe
- Dual Xeon with Powervaults
  - 4 Sale - Cheap!
- NetApp NFS
- Plus daakman, aldar, simon, lenny, laddie, stampy, krustofski, michellepfeiffer

HARDWARE

- 50+ servers, 8-way Opteron, 4-way Opteron
- NetApp Storage
  - Fibre Channel
  - iSCSI

HARDWARE - FUTURE

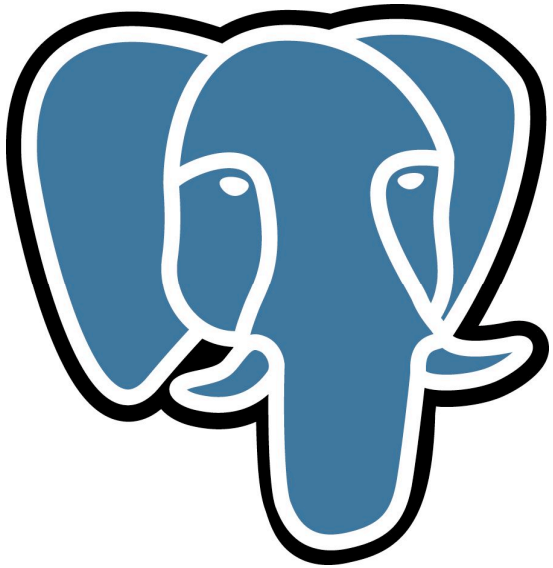
- Phase out dual Xeon
- Phase out Powervaults
- More and more 8-ways
- More Netapp Shelves
- More and more databases

POSTGRESQL OLD

- Lots of 7.4.x versions
- Compiled on each box
- In /usr/local/pgsql
- Application managed replication

Postgresql Today

- Stock 8.2.4 as an RPM package
- Managed with cfengine
- Slony 1.2.9 RPM
- Consistent Replication

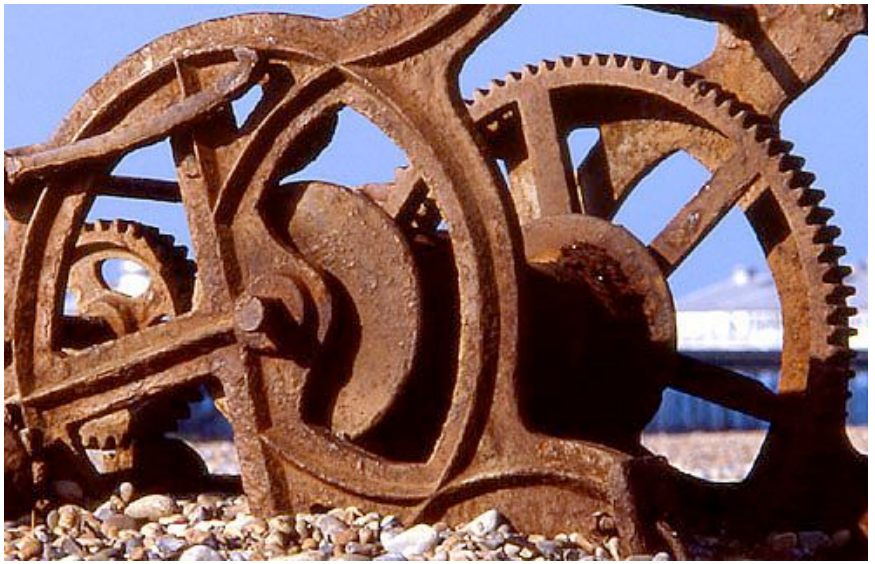


POSTGRESQL TOMORROW?

- High Availability/Failover
  - Steeleye, VCS?
- EnterpriseDB?
- Postgresql 8.3
  - HOT – great for update-oriented tables
  - GIN built-in

DBA POSTGRESQL TOOLS

- pgfouine
  - Run off our central logging server
- pgspy
- Custom Scripts
  - pg-status – queries all DBs for sizing
  - pg-fsck
    - Clean up permissions
    - Remove orphan rows
  - Idle connection killer
- Hyperic HQ Monitoring



HYPERIC HQ


Indicator Charts
View: Update Default
99.2% — Availability
Commits per Minute
(PostgreSQL 8.2)
LOW: 4,981.2
AVG: 40,503.0
PEAK: 89,247.8
93,461
70,288
47,115
23,941
768
Rollbacks per Minute
(PostgreSQL 8.2)
LOW: 0.0
AVG: 9.9
PEAK: 44.8
47
35
24
12
0
Locks Held
(PostgreSQL 8.2)
LOW: 2.0
AVG: 48.1
PEAK: 620.0
651
488
325
163
0
06/12/2007 03:02:59 AM
06/12/2007 02:50:59 PM

POSTGRESQL TOOLS FUTURE

- Pgbouncer
  - Needed for using 100s of Application Servers
  - Looks good for maintenance
- Pgmemcache
  - Would allow for transactional invalidation of cache data, including “by-hand” SQL.
  - … If only it could create serialized Java Objects…

LOVE POSTGRES

- Concurrent Index Builds
- Fast Queries (when working set is in memory)
- Consistent logical dumps
- MVCC
- Non-Blocking DDL (add columns without downtime)
- Amazing Community Support
  - Mailing lists, freenode, and more

LOVE POSTGRES

- Transactional DDL
  - BEGIN TRANSACTION;
    ALTER TABLE foo RENAME …
    CREATE TABLE foo
    COMMIT;
- Used heavily for queue systems
  - Writes go to the raw table
  - Queue job grabs that table and works on it

LOVE POSTGRES

- Inheritied Tables with table Rules
  - CREATE TABLE foo_01 () inherits foo;
    (repeat for 02-12)
  - CREATE RULE foo_01_r AS ON INSERT TO foo
    WHERE (date_part('month', new.ts) = 1
    DO INSTEAD
    INSERT INTO foo_01(….) VALUES(….)
  - Use TRUNCATE to clean out tables
- Used for stats/logging/tracking
- Inheritence allows us to query the entire set

DISLIKE POSTGRES

- Vacuuming
  - Takes much too long for very large datasets
- Autovacuuming
  - 3000 Transactions per second
  - 3000 * 60 * 60 *24 = 259,200,000
  - autovacuum_freeze_max_age = 200,000,000
- Checkpoints
  - Amount of writes can overflow BBU cache
- MVCC scans/count(*) all that.

WRAPUP

- Hi5 + Postgres + Memcache etc = Alexa #11
- You will hit plateaus as you grow
  - Postgres major releases solve some of these
  - Otherwise expect to get your hands dirty.
- Looking forward to lots of great new features.