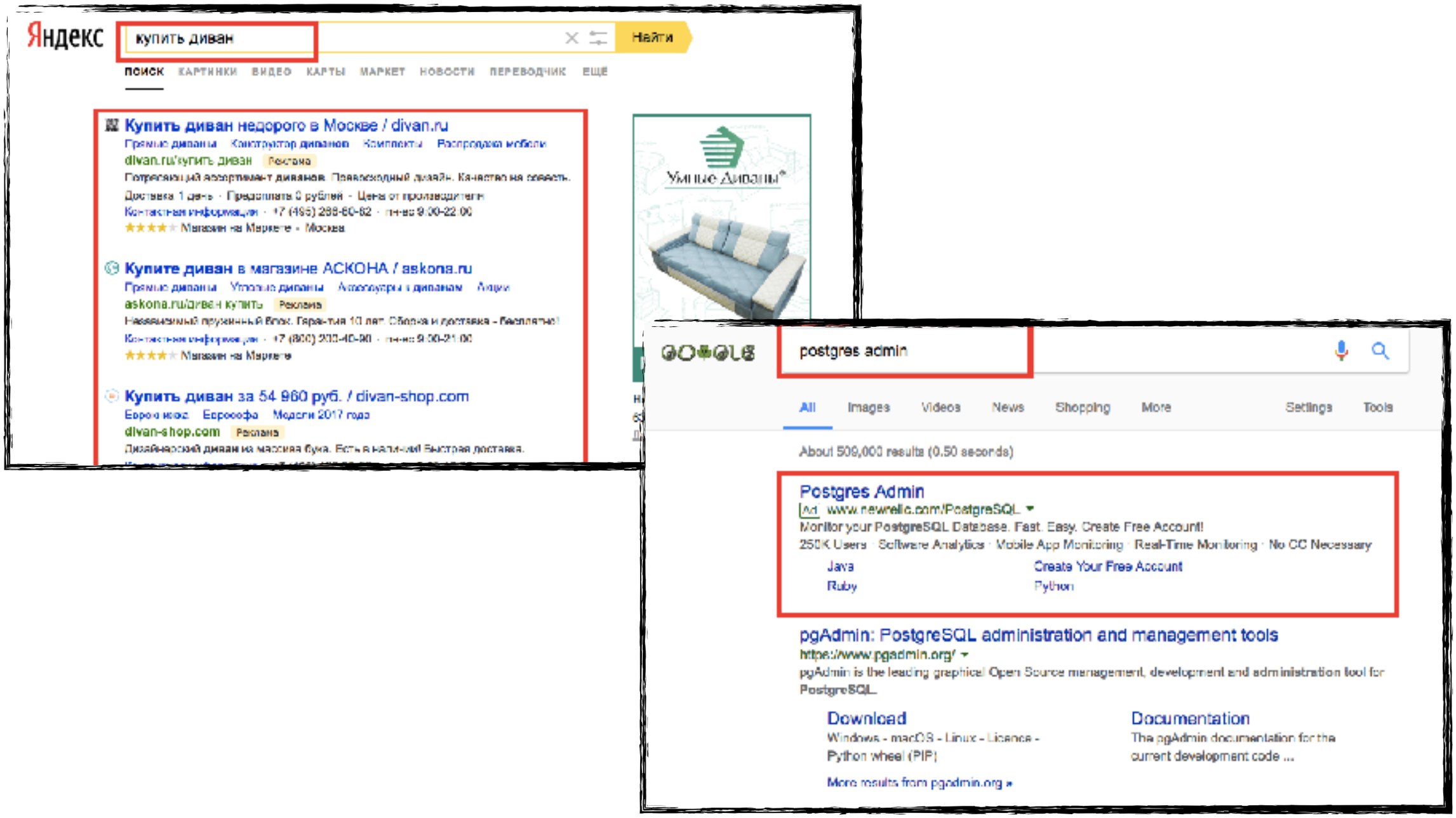
Postgres at Alytics storage layer.

Max Vikharev, Alytics, 2017

Contextual Advertising

Contextual Advertising



Advertising platforms

- Google Adwords
- Yandex Direct
- Facebook
- Instagram
- …

Goals tracking Analytics Systems

- Google Analytics
- Yandex Metric
- User CRM, 1C Bitrix, amoCrm
- Calltracking (Comagic, Calltracking, Alloka)
- Custom integrations

Performance Marketing

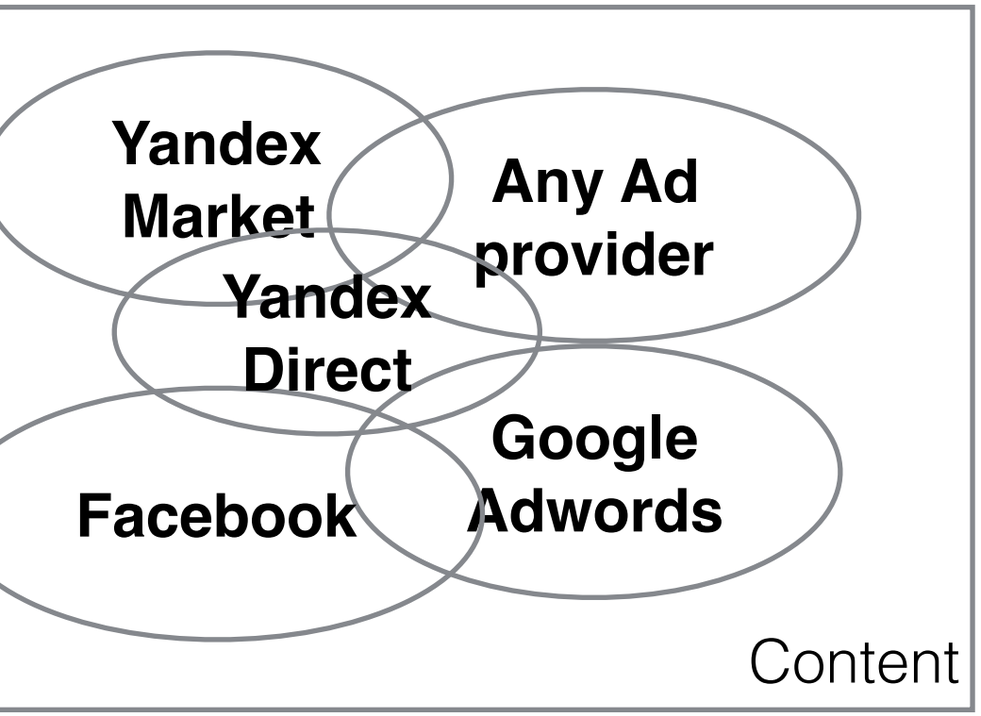
- Run (create) advertising content
    - campaigns, ads, phrases
- Measure results: collect clicks/costs/goals statistics
- Budget decision
- Manage content
    - statuses
    - **bids**
    - update content

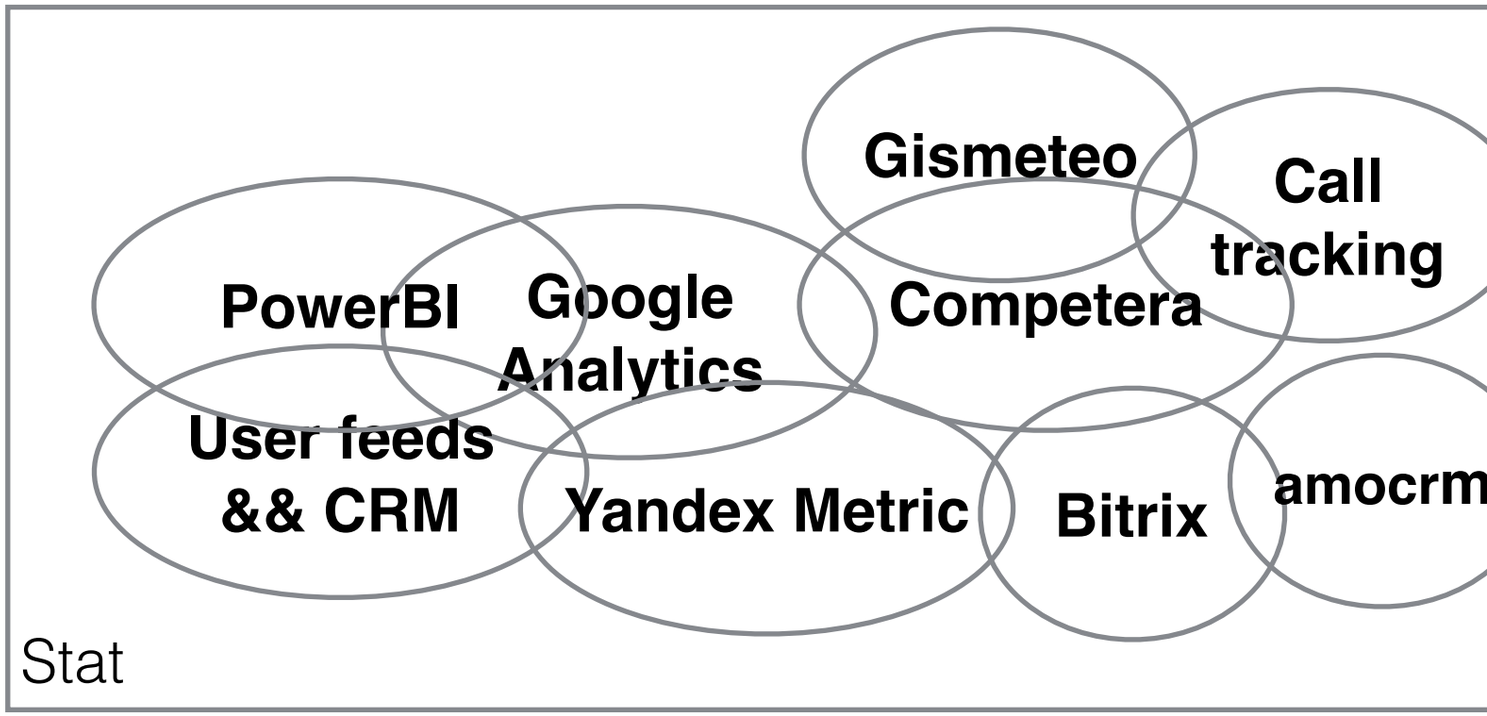
Our clients

- context advertising agencies
- marketing departments of big e-commerce projects
- self employed specialists
- business owners

- fetch && regular sync **advertising content**
- fetch costs and revenue **metrics** daily, calculate efficiency (CPA/ROI)
- show efficiency dashboards (content levels: project, services, campaigns, groups, ads, keywords)
- sync bids, statuses
- generate ads from YML
- automate decisions (strategy, istrategy)

External Services





Internal Services

Semantic

Ad Generator

404 checker

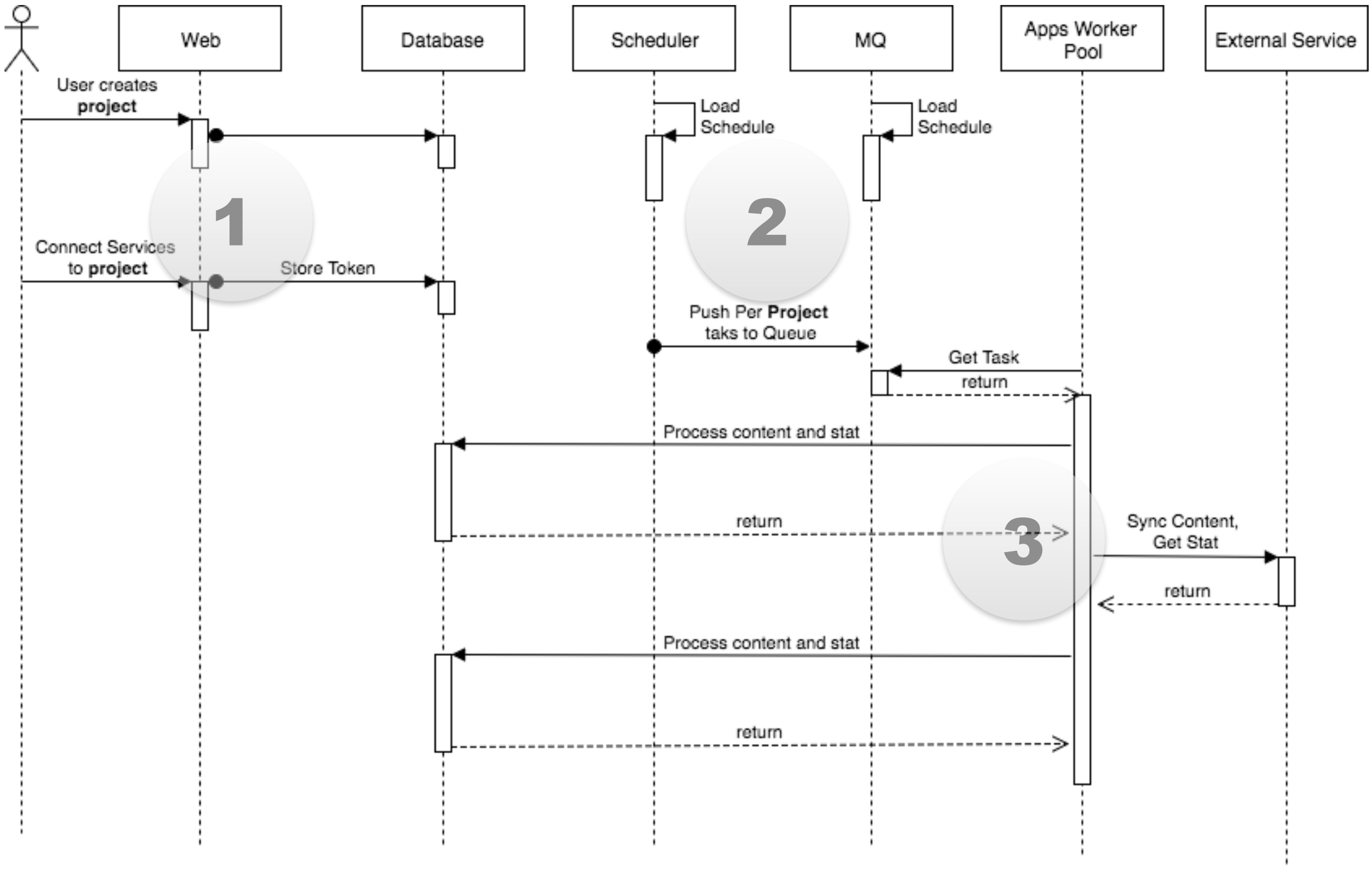
AutoRules

Static Bidding

Stat Reports

iStrategy

Typical architecture of queues processing system.



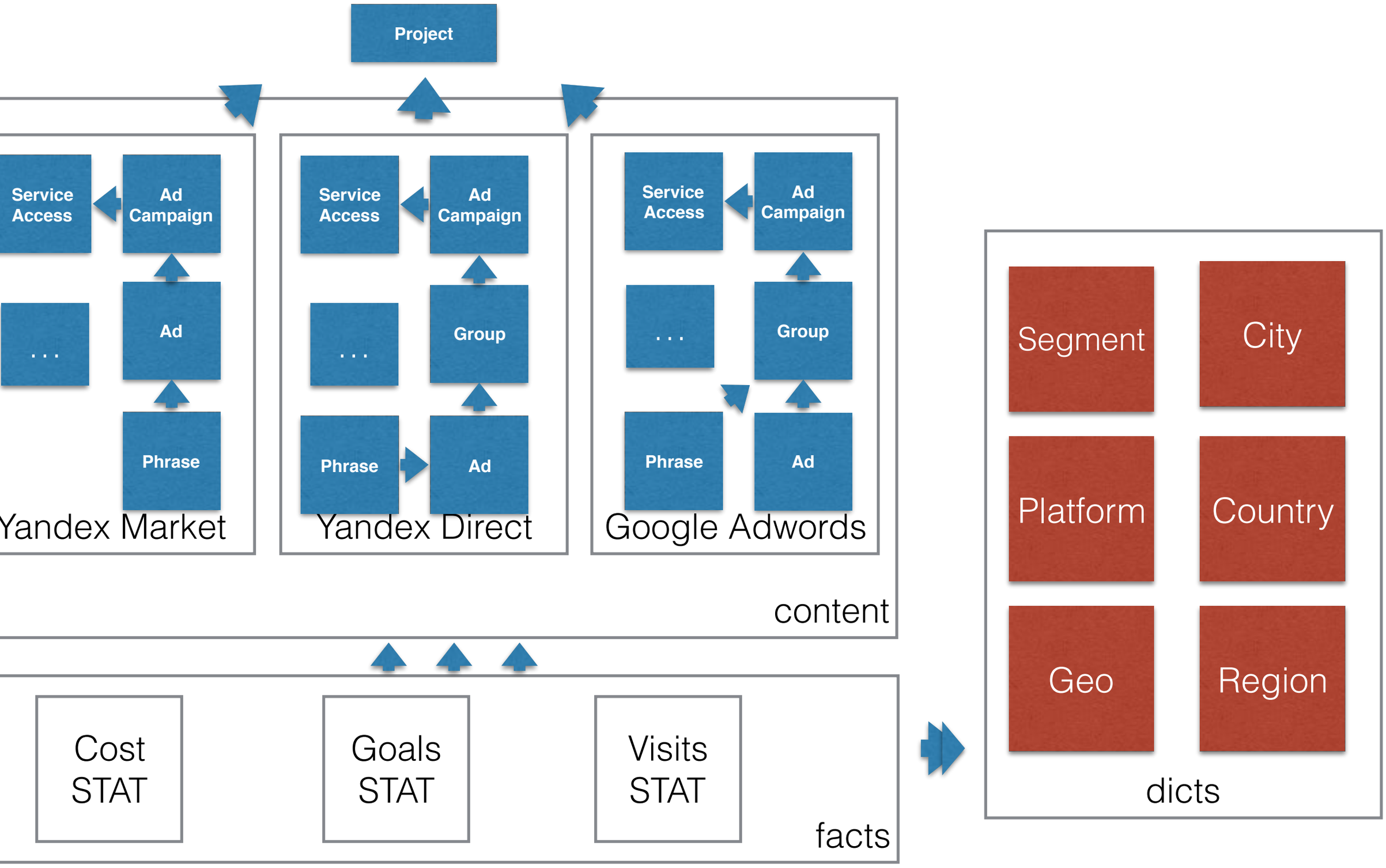
Apps tasks (current)

- 30 django apps
- 2 pgsql apps
- 359 Rabbit queues, Few PgQ queues
- Different tasks (concurrency, resource load, events)

Tasks in queues

- **Task** per **project of user**
- Regular chains of tasks: workers sync content, creates new content, process attributes.
- Frequent tasks (bidding, getprices)
- Nightly workload (metrics collect: costs, goals and revenue, visits collector)
- Reporting (UI + CSV) (low concurrency OLAP queries)
- Autorules, istrategy algorithms (hight concurrency OLAP queries + OLTP updaters)
- ETL Low concurency batch processings)

Dimensions - metrics



Online queries (OLTP)

- background tasks with high concurrency updates content
    - full CRUD workload
    - django ORM multiJOIN SELECTs
- stat models INSERT + DELETE (nightly workload)

Analytical queries (OLAP)

- Historical sampling: date_from;date_to
- Filter by **parent content and attributes**
- Total +Total filtered
- TOPN Rows (Sorted and Paginated)
- All Rows in CSV reports
- Calculated formulas on multiple metrics
- Historical Finance multipliers

- Questions?

Dev Start

- 2011 summer
- python 2.7, django ORM only!
- **Autotests**! (High level autotests -> module attests + ui tests)
- Minimal CI (git push -> test develop -> deploy testing)
- Env automation: puppet, **vagrant**, fabric
- Zabbix - **proactive** monitoring
    - basic services triggers
    - CPU Utilization (Iowait), Network, Memory
- First app server VDS 200 Gb, 4 x CPU, 8 Gb RAM

Limitations on the start

- Django 1.4 (psycopg autocommit conflict!) o_O bloat, locks
- minimal DBA expertise: PostgreSQL 9.0 (default config) :-(
- Poor plain schema (user <- project <- service <- content (hierarchy) <- stat)

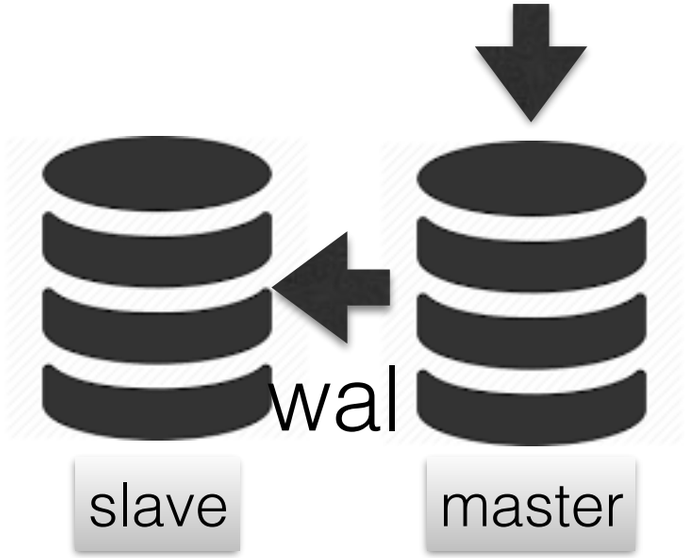
Operations

- ~~dedicated admin~~
- infra as a code from the start (#asan, #nabam ;-))
- ORM queries, minimal raw SQL -> ~~dedicated DBA~~
- wait the moment ;-) -> proactive monitoring (collect everything)

On the Start

- **Python ORM**
- **Python ORM OLAP (python inmem)**
- **iops/checkpoint/wal**

UI + reports, workers



First problems

- **Multiple JOINs are slow on hierarchies (OLTP, OLAP sad)**
- **Filter by parent attribute is slow (OLAP sad)**
- **Scheduled reports (first OLAP workers)**
- **Slow SELECT on growing content**
- **+Bills of rows in stat nightly (OLAP sad)**

Also

- **soft written and tested**
- **business needs features**
- **clients are coming**

Django Slow Query Monitor

- How to find slow ORM queries?
- ELK Based + Redis transport
- Logging Middleware + Python traceback analyser filter in log stash
- From distributed workers network
- Application Profiling detailed to line/task/app

How to SELECT content

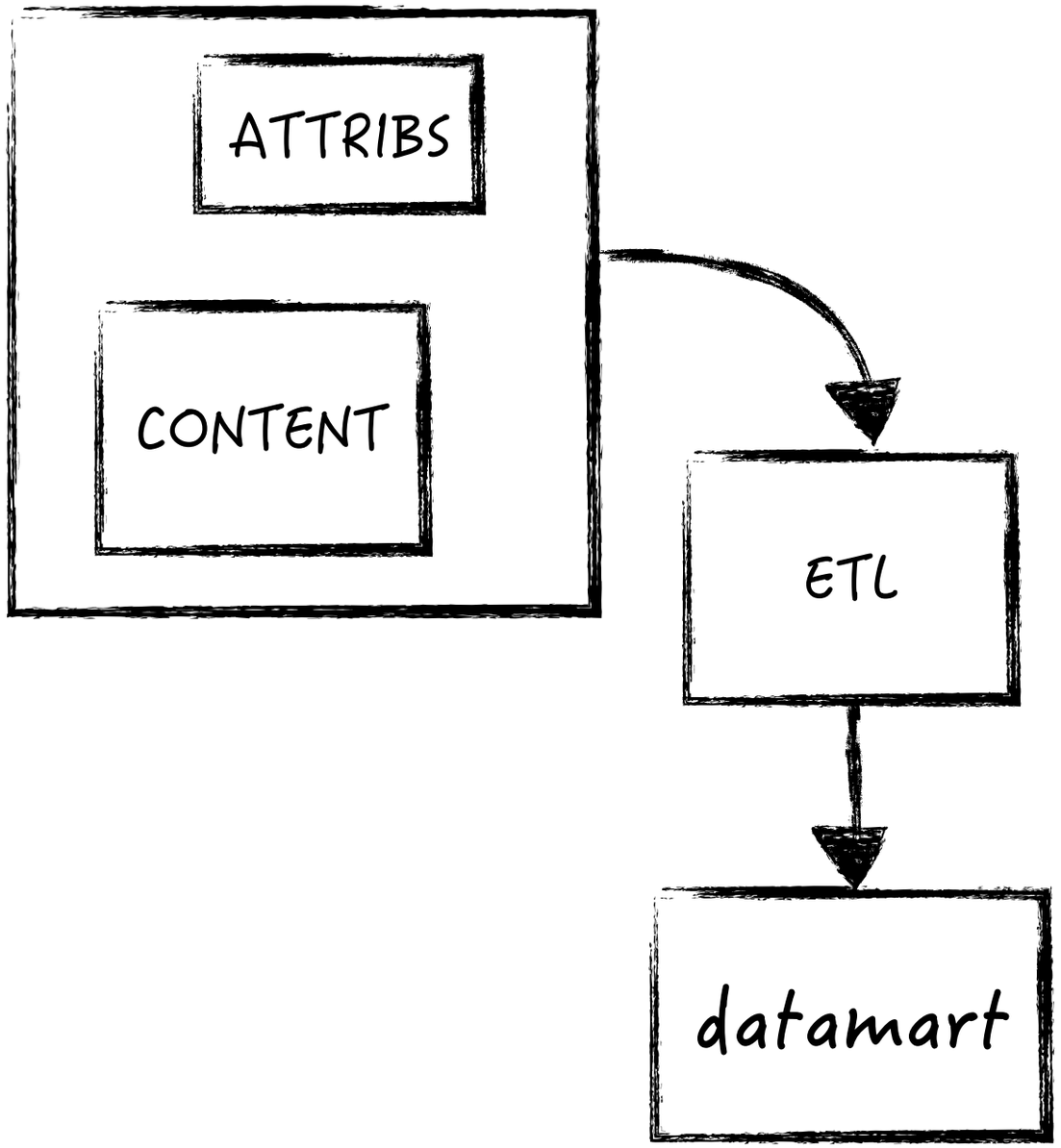
- **Poor Schema. No DDD on the start - very sad**
- **Monolith content hierarchies - one relation**
- **Use Index, Luke! Ok, 22 Indexes added**

How to store stat

- **Stat partitioning (pg_partman)**
  - **by week**
  - **by day**
  - **create N sections in future nightly (cronjob)**

Reduce JOIN

- **Denormalisation (denorm)**
- **JOINS -> ETL**



First ETL

- **full datamarts rebuild**
- ***/2 updates (business sad)**
- **15 min hard workload (OLTP sad)**

Increment

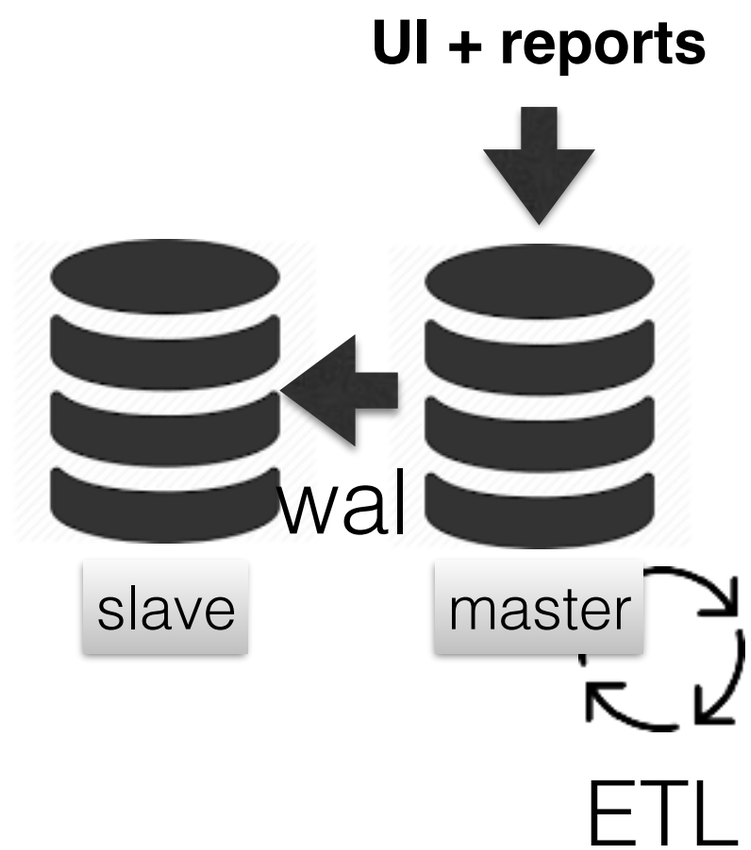
- **Increment ETL (#dshirokov)**
  - **Select to staging table (time based in content)**
    - ***/15 sec**
    -
```
updated = models.DateTimeField(verbose_name=u'Дата обновления',
    auto_now=True, default=datetime.datetime.now())

class UpdatedQuerySet(models.query.QuerySet):
    def update(self, *args, **kwargs):
        return super(UpdatedQuerySet, self).update(*args,
            updated=datetime.now(), **kwargs)


class ContentManager(models.Manager):
    queryset_cls = UpdatedQuerySet
```
    - **All raw SQL upgrade**

Problems

- **Lost changes (long transactions)**
- **Bloat (hello aggressive autovacuum)**
- **stupid autocommit!**
- **OLTP cache missed (more RAM plz)**

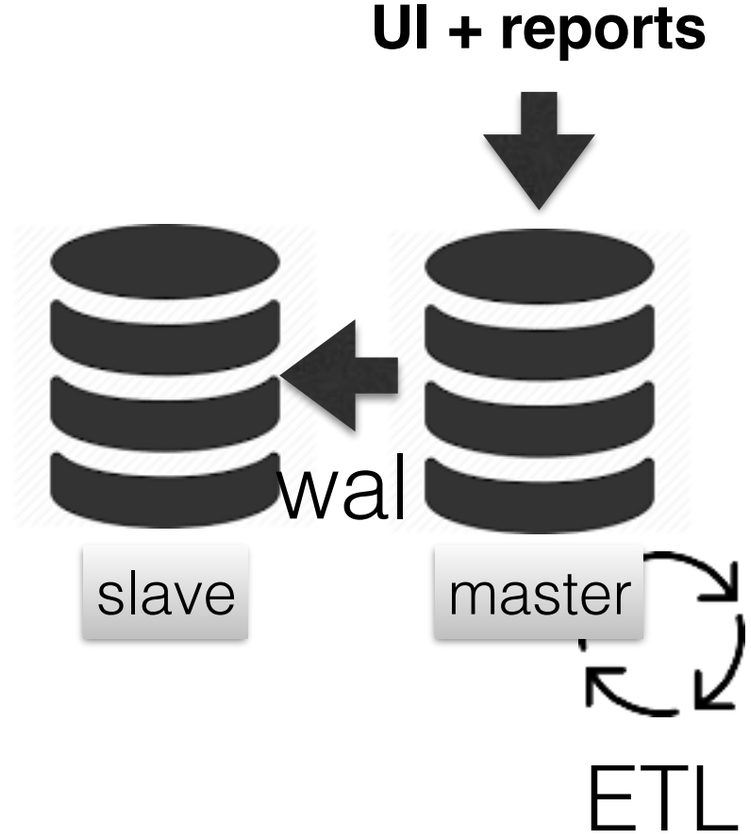
ETL on Queues

- **PgQ**
  - **max_count**
  - **max_lag**
  - **idle_period**
- **inf. ETL worker**



Problems

- **OLTP very sad**



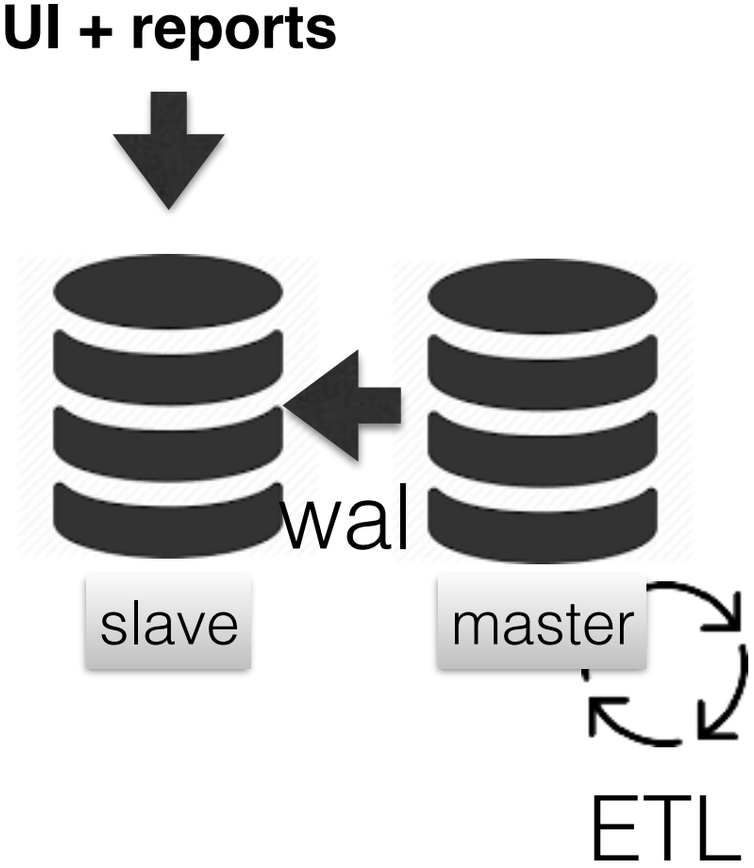
Upgrade Infra

- **2 x SATA -> 4 x SAS -> 4 x SSD**

- **continuous infra moving**

OLAP to slave

- **Easy**



B
buiseness

Problems

D
devs

- **D: Cancelled transactions (feedback on=bloat) BBbbbbloat (Hello, 22 indexes) Fast growing indexes = huge autovacuum blocks :-((:-(**

- **B: new apps: auto_rules, istrategy. OLAP queries usage up :-)**

- **D: nightly OLTP workload conflicts with OLAP :-(**

- **D: concurrency UP :-(**

- **D: OperationalError: canceling statement due to conflict with recovery (User was holding shared buffer pin for too long) Several small transactions holds autovacuum blocks, that cancelled transactions o_O**

- **B: sad**

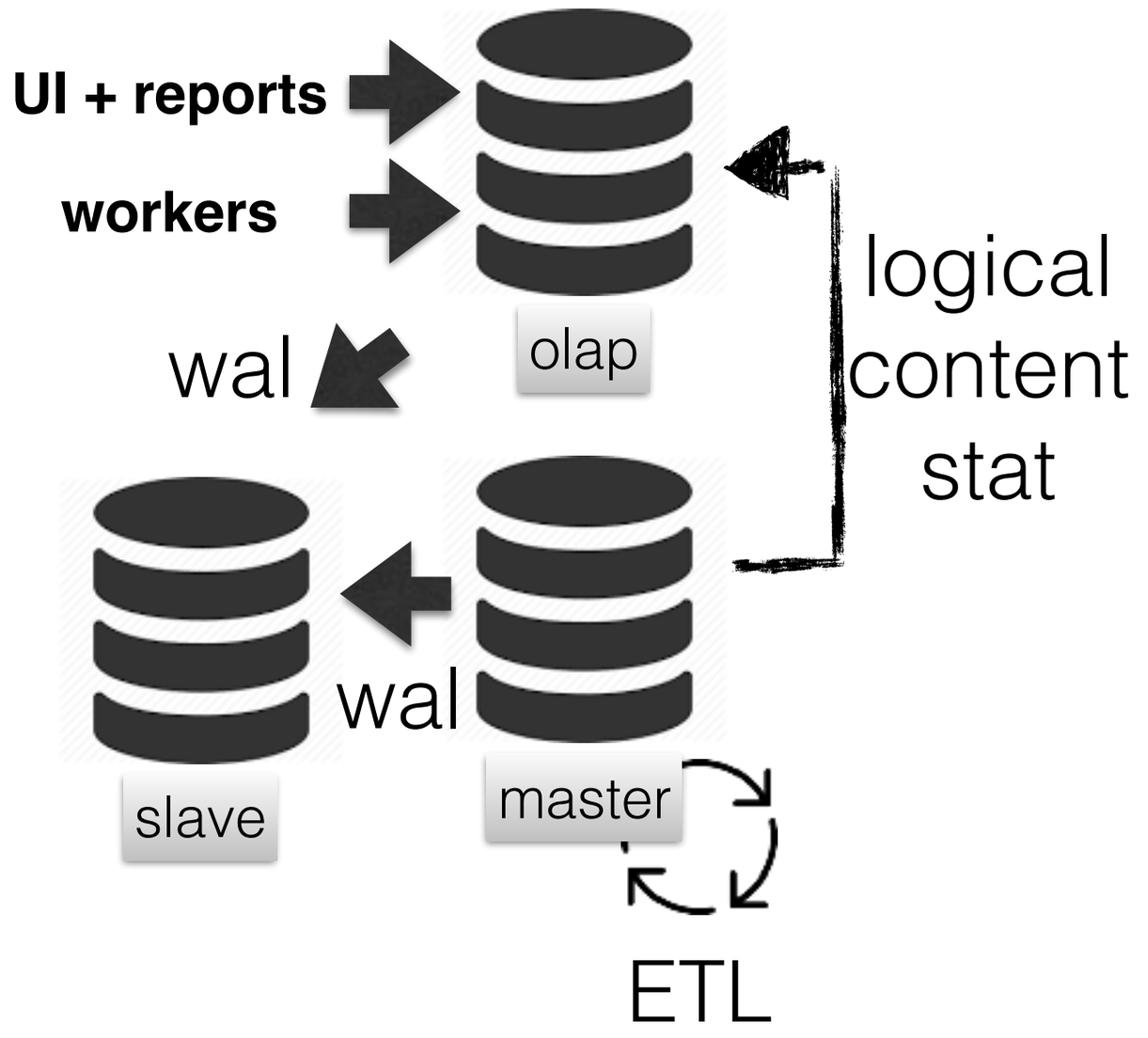
Reduce conflicts

- **APP: Batch workload in nightly (stupid partman triggers):**
  - **UNLOGGED _tmp_table_**
  - **INSERT TO target_stat FROM _tmp_table_ GROUPBY**
- **APP: Aggregates for high level content**

- **Still .. cancelled transaction**

Logical Split

- **Logical replication:**
  - **pgq -> londsite**
- **Stable OLAP queries !**



Skytools Problems

- **Initial copy x100 sections extremely slow**

- **max_count not works on huge transactions**

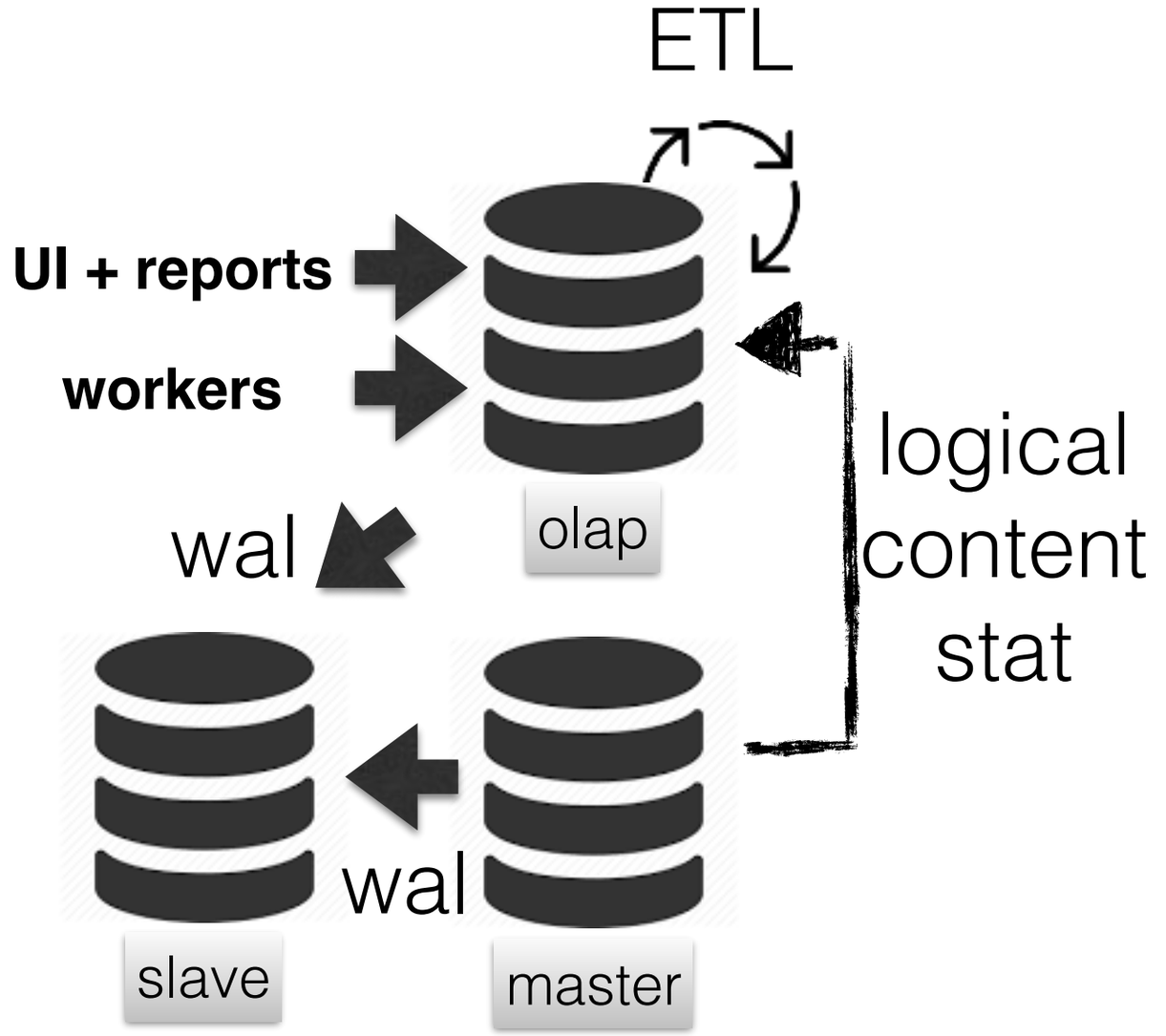
App Revolution: CTE SQL

- **APP: Parallel ORM -> CTE SQL generator #dshirokov**
- **B: workers concurrency UP**
- **D: ETL+OLTP=bad**

All non-OLTP to OLAP

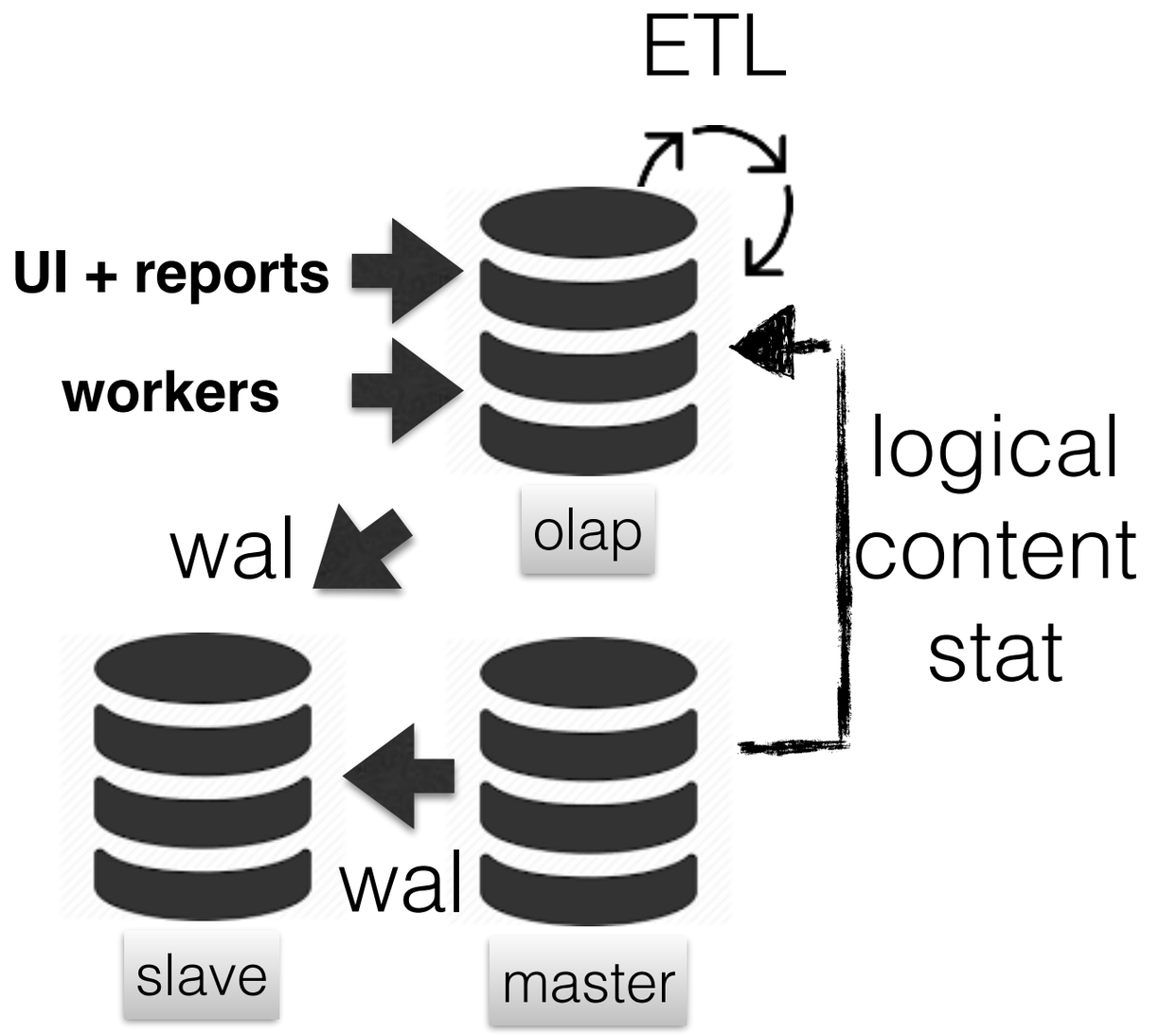
- **D: Move ETL to OLAP**

- **D: Move nightly stat batches to OLAP**



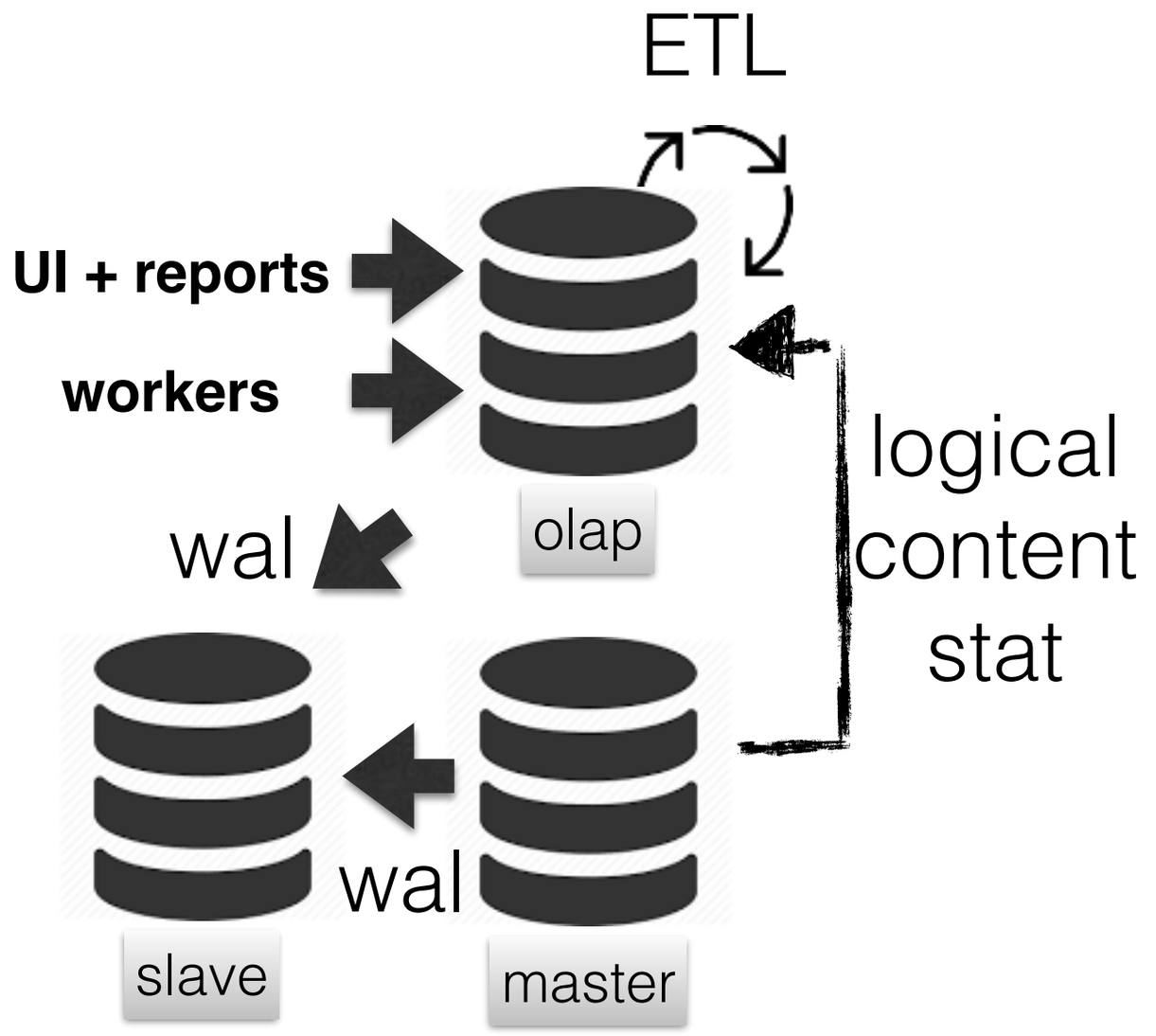
Upgrades

- **ETL + OLTP = bad**



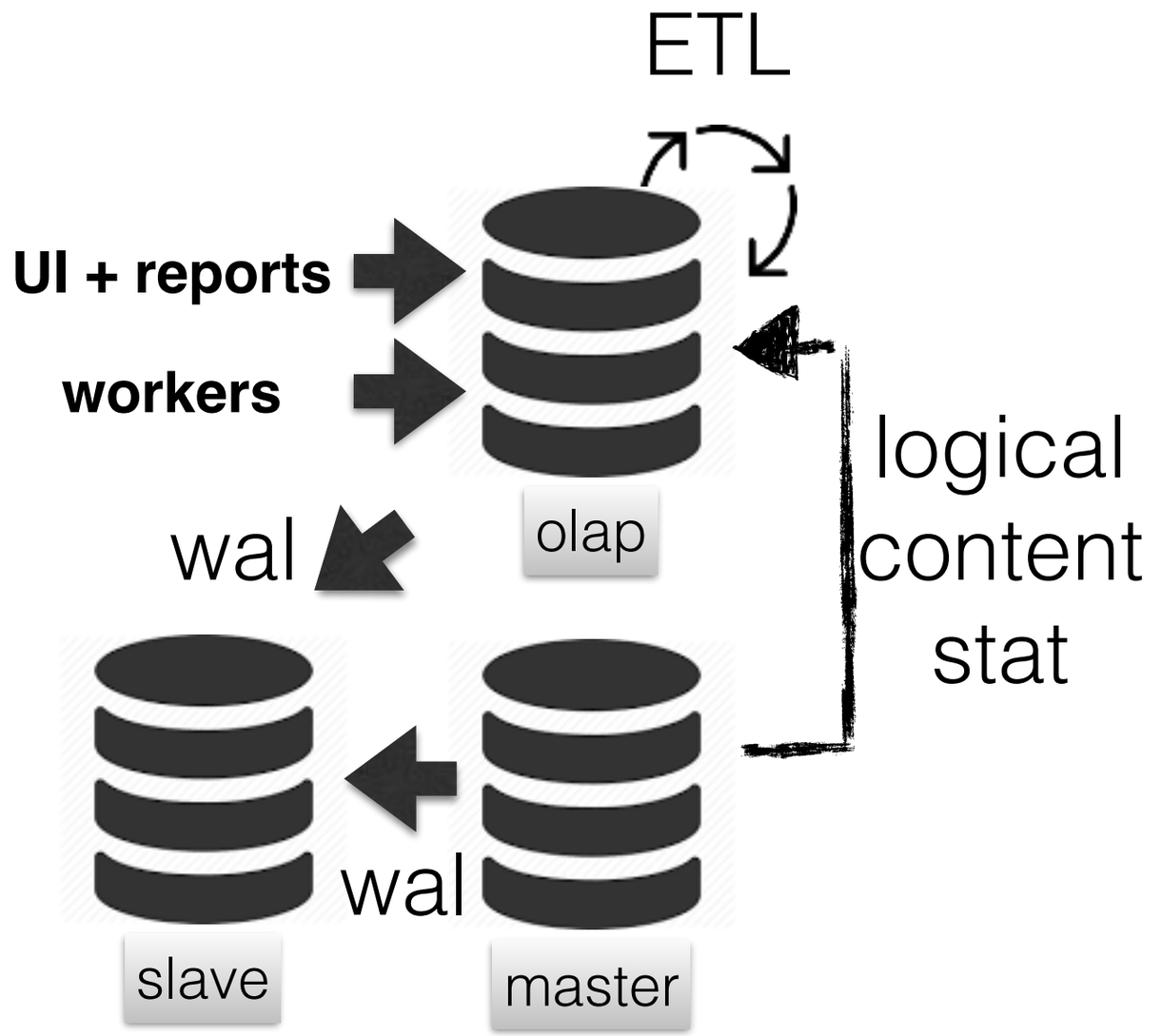
Upgrades

- **ETL + OLTP = bad**
- **ETL + OLAP = worst**



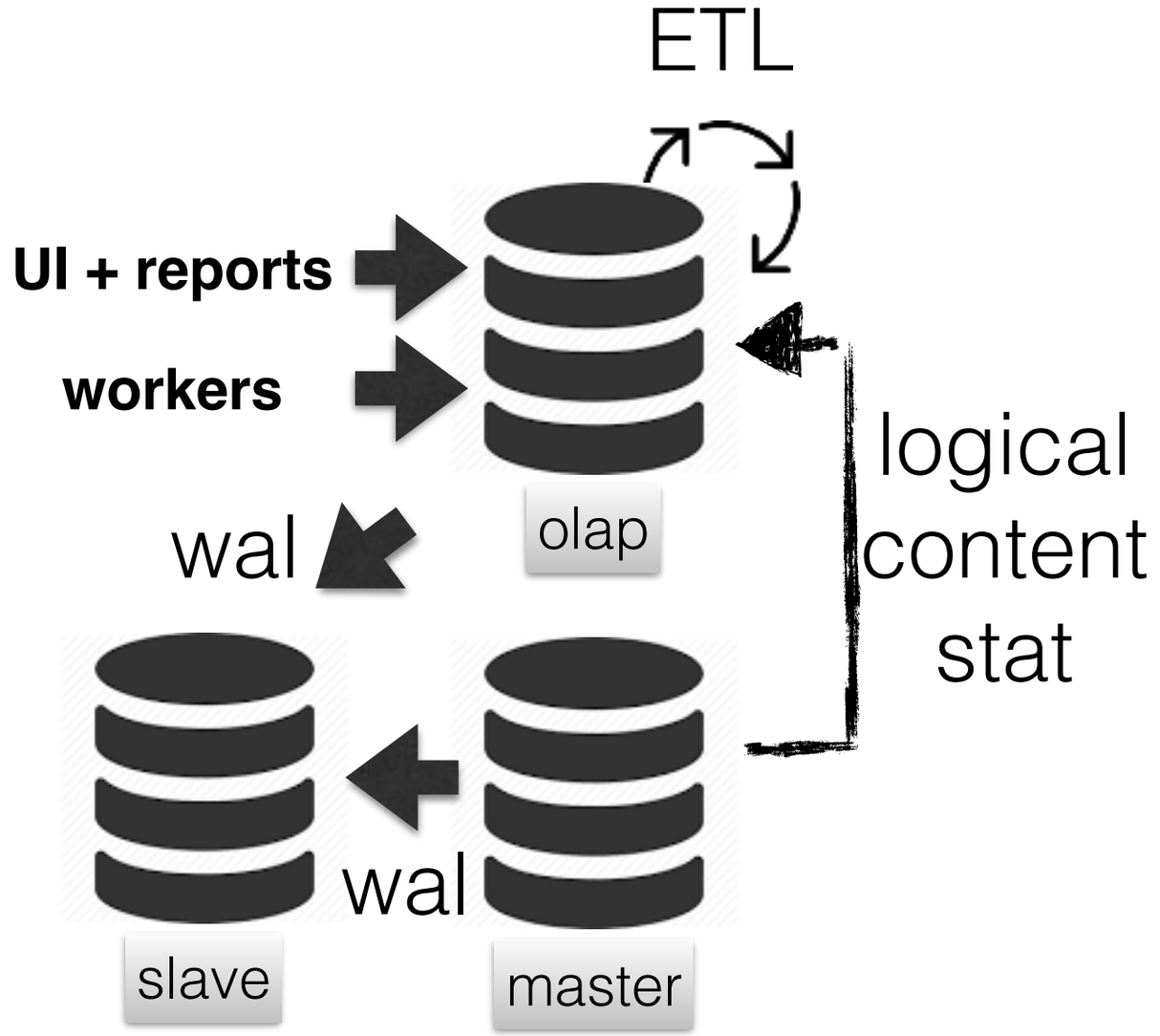
Overload

- **ETL, AGG, OLAP SQL**

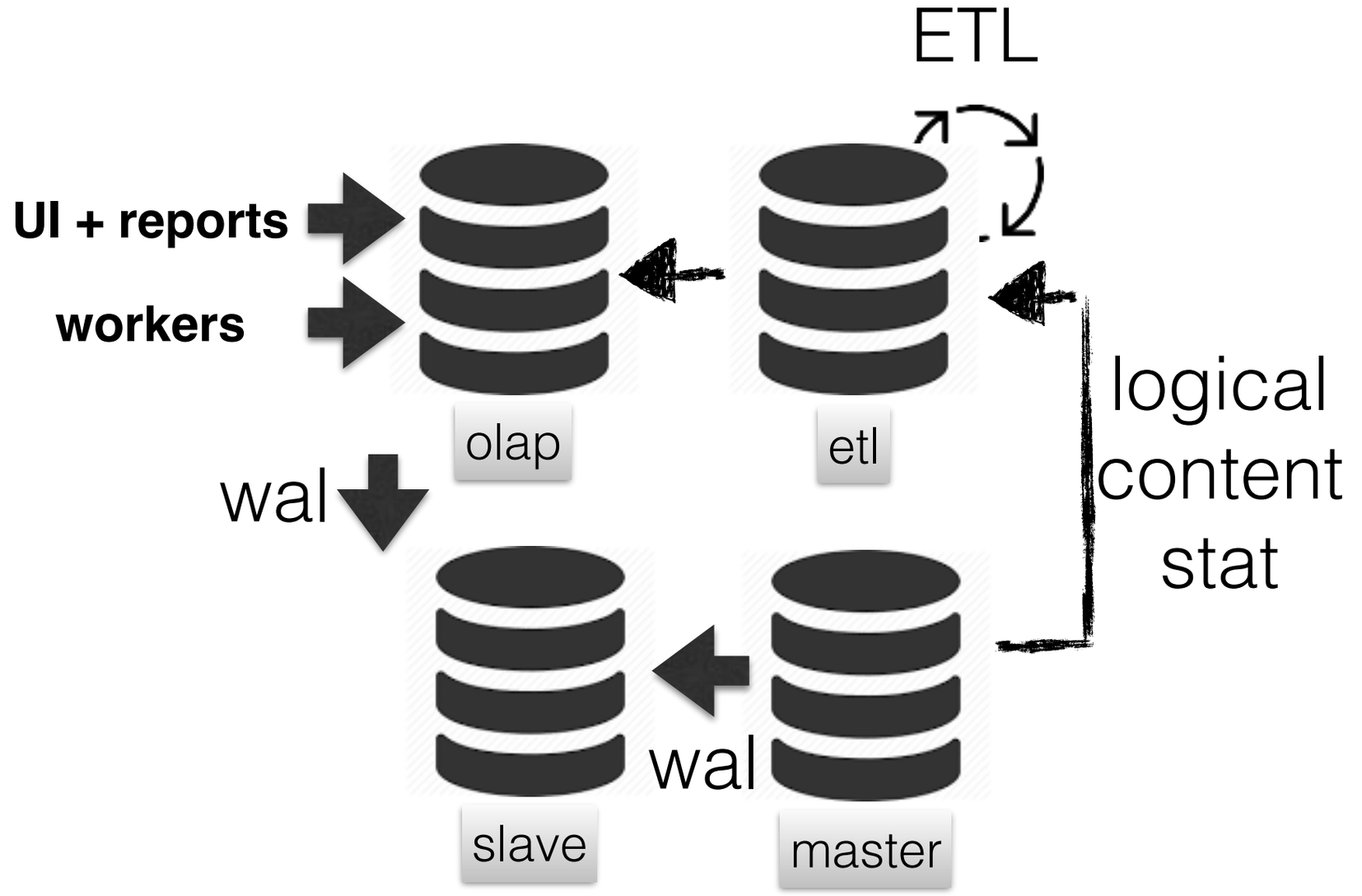


Overload

- **ETL, AGG, OLAP SQL**
- **OLAP: Out of memory**



Individual ETL node



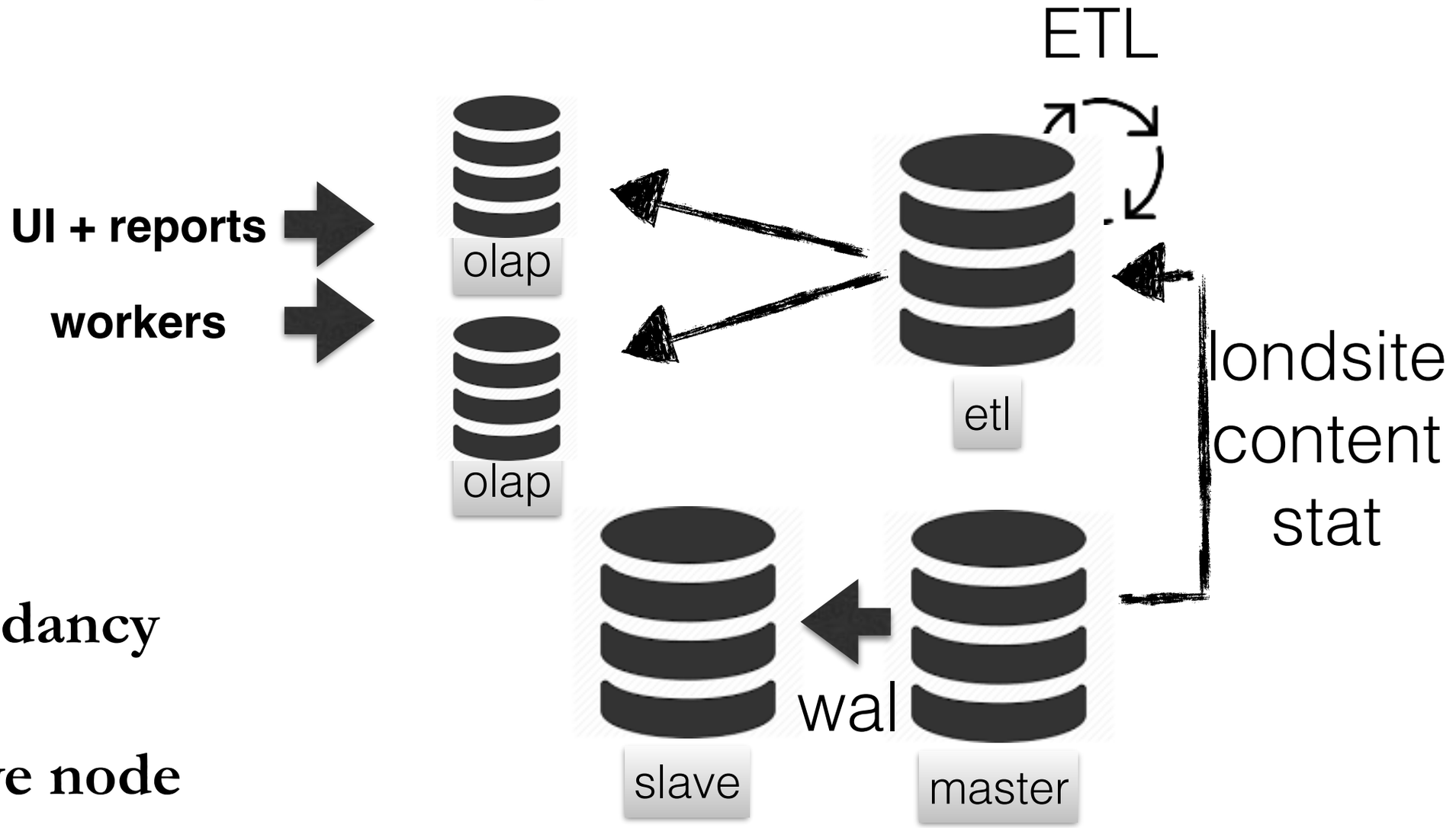
What about slave

- **new wal from OLAP needed**

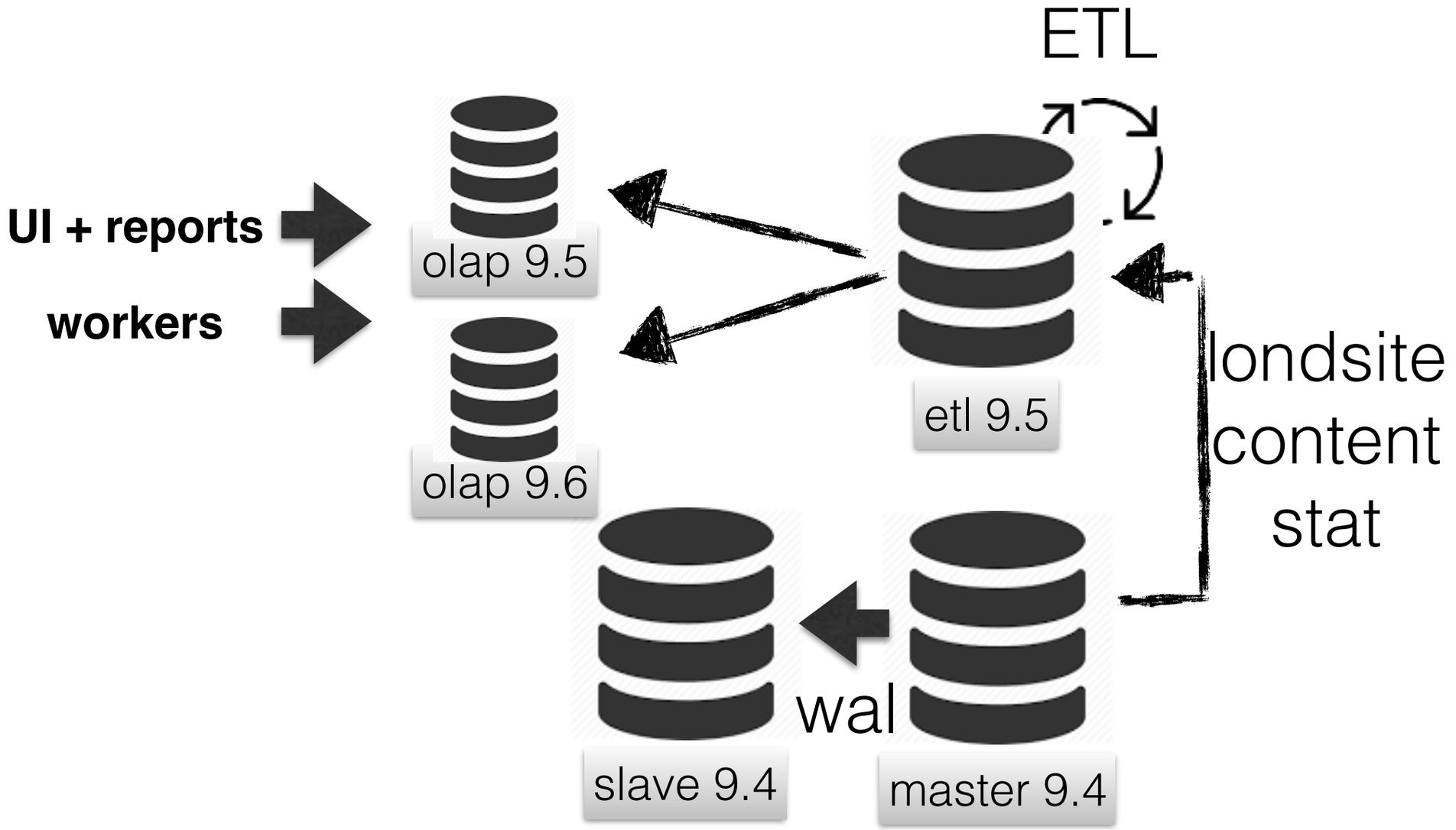
- **restore londsite from wal slave is so hard**

Write 2 places

- **Redundancy**
- **Reserve node**



Try new versions



•

Main shard benchmarks

- content
  - 0.85 bill rows
  - 23 indexes
  - 170 Gb Data
  - 500 Gb Indexes
- datamarts/hierarchies
  - 0.85 bill rows
  - 500Gb Data
- stat + agg
  - 1500 Tb

- OLTP 8K tps
- OLTP -> ETL -> OLAP ~ max 10sec increment
- OLAP queries up to millions rows (up to 10 mins)

Active data ~ 70%

Space and Cache

- **RAM cache content**
- **SSD cache (content + last month stat data)**
- **SATA cache (old stat sections to archive)**

Hardware (main shard)

- Intel® Xeon® E5-1650 v3 Hexa-Core Haswell
- RAM256 GB DDR4 ECC RAM (active data in cache)
- SSD + SATA
  - db-master RAID10 4 x 900 Gb SSD
  - db-slave RAID10 4 x 900 Gb SSD
  - db-etl RAID10 4 x 900 Gb RAM SSD
  - db-olap RAID0 2 x 900 SSD, 2 x 4 Tb SATA (archive)

Software

- pgbouncer, skytools
- db-master postgresl 9.4
- db-slave postgresl 9.4
- db-etl postgresl 9.5
- db-olap postgresl 9.6 + PARALLEL Scan

Summary

- **Business got planned features**

- **Devs got skills described in JSON-configurable multicluster cookbooks (show it, man!)**

But

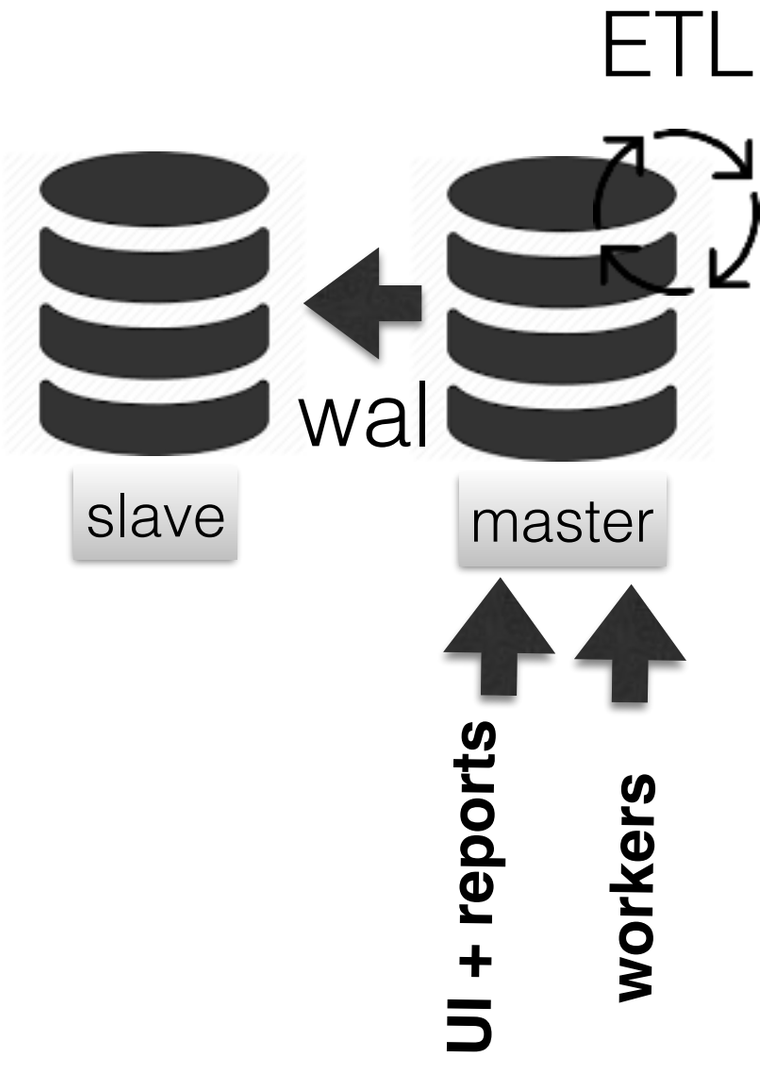
- **hard maintenance: reindex, vacuum, repack, compat (TBD)**
- **hard schema updates**
- **DBAs do not like ruby**

Whats next

- window functions
- column storage + incremental stat updates
- MDX?
- maybe small shards?

What shards?

- “Shard as late as possible” - not good (3 years) dedicated servers for big user (up to 256Gb per node)
- single master-slave (ETL, AGG, OLAP, OLTP)
- easy maintenance&deploy
- fully automated (chef), cross-shard common workers
- use workers nodes for pg clusters. cheap hw (default SSD drives up to 480 Gb).



So

- **Poor schema makes life more difficult! Think DDD schema plz (New apps confirmed)**
- **Event bad schema engineer make it works**
- **Shard as late as possible! (c) ?<?**
- **Proactive monitoring rulezzZz**
- **Learn limitations and simplify**

Waiting From community ;-)

- **postgres 10**
- **pushdown aggregates**
- **native logical replication or stable pglogical (plzZZz)**
- **native columnar storage cstore_fdw(UPDATE?) or Optimized Row Columnar (ORC) format (VOC ?)**
- **native partitioning (pg_pathman?) with improved planning**

Max Vikharev

mvikharev@alytics.ru

Maintenance

- Devops
- Hardware
- Software
- Monitoring
- Failover
- Incidents management
- Basic HW Tune up
- Tune more
- Slow Queries
- BULK updates
- OLTP Bloat
- OLAP Bloat
- EXTENSIONS
- Migrations
- REINDEX
- Autovacuum and reorg