ONE NATIONAL HEALTH SYSTEM - ONE POSTGRES DATABASE

(ARCHITECTURE AND PERFORMANCE REVIEW)

Boro Jakimovski, PhD
Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius University in Skopje

Dragan Sahpaski
Sorsix

Who are we?

- Boro Jakimovski PhD
  - Responsible for the Infrastructure of MojTermin
  - Assistant professor & Head of Computer Center
    Faculty of Computer Science and Engineering,
    Ss. Cyril and Methodius University in Skopje

- Dragan Sahpaski
  - Part of the team developing and maintaining MojTermin
  - Programmer at Sorsix

Special thanks for helping prepare this presentation

- Vladislav Bidikov & Bozidar Proevski
  - System and network admins
  - Computer Center Faculty of Computer Science and Engineering,
    Ss. Cyril and Methodius University in Skopje

What is this presentation about?

- Moj Termin (My Appointment)
  - National Health Management System
  - http://mojtermin.mk
- Optimized* System
  - Various constraints
  - Tight budget
  - No downtime in ~ 9 months
  - Maintaining active development
  - Constant delivery of new features

Optimized: running with no disruptions. It doesn't mean optimal system.

Agenda

- Overview
- Architecture
- Hardware
- Replication
- Query Optimization
- Development
- Monitoring
- Critical situations

Agenda

- **Overview**
- Architecture
- Hardware
- Replication
- Query Optimization
- Development
- Monitoring
- Critical situations

Overview of the System

- Electronic Health Record
- Referrals
- Prescriptions
- Drugs Register
- Ambulance module
- Hospital module
- Referent Code lists
- Diabetes and Insulin Module
- Transplantation waiting lists
- IVR Module
- Surgeries
- Activity calendar
- Health Registers (flu, cancer, diabetes, deaths, births, …)
- Medical staff training
- Vaccination
- Mammography Screenings
- Patient notifications
- Integrations (Health Fund, Third party software vendors)

Users of the system

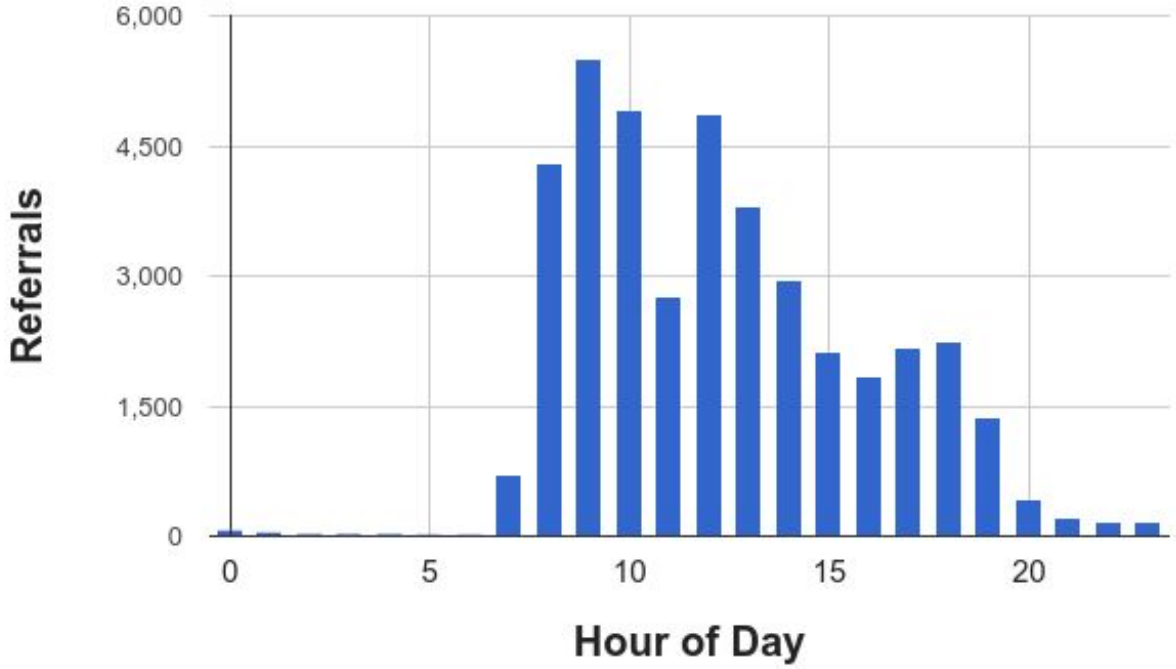
- > 15.000 user accounts
  - > 30 roles: doctors, nurses, medical staff, pharmacies, administrators, …
  - State and private owned medical institutions, pharmacies, institutes, …
- 62 Third party medical software vendors
  - Registered with an API key

Adoption and Success

- Rough daily statistics
  - At the monthly peak (first week of each month)
  - 170k prescriptions
  - 40k referrals
  - 2M API requests
  - 1M web page requests
  - 15M DB Queries

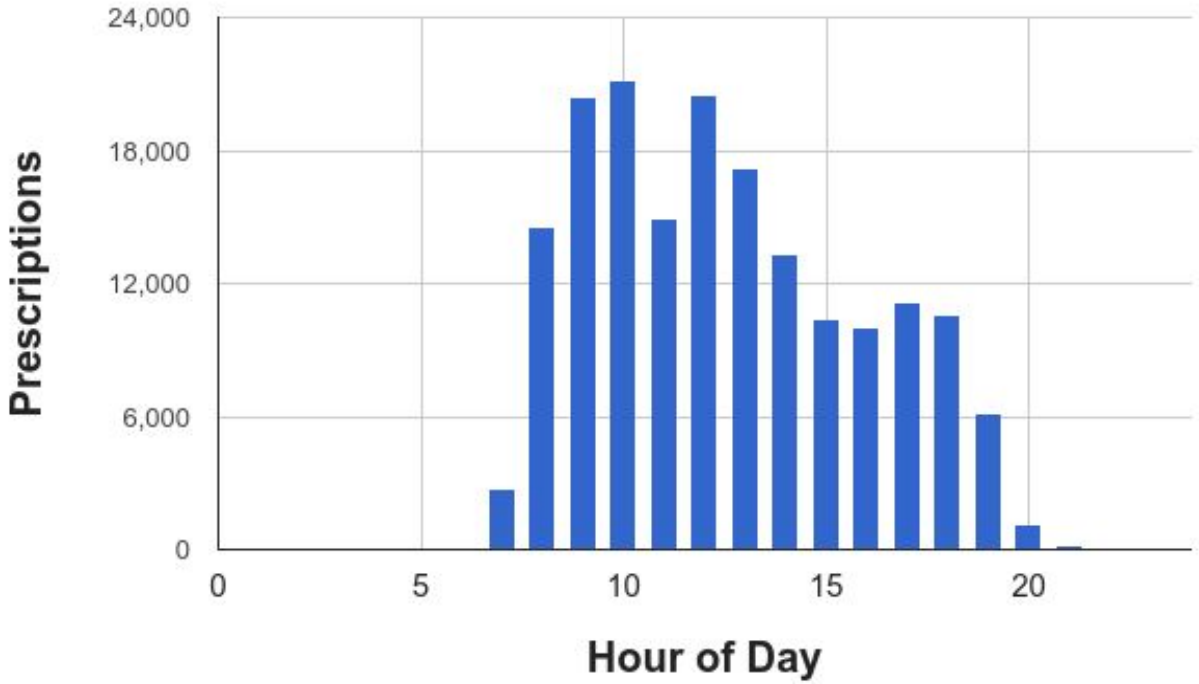
05.10.2015 First Monday of October

Referrals: 40,825



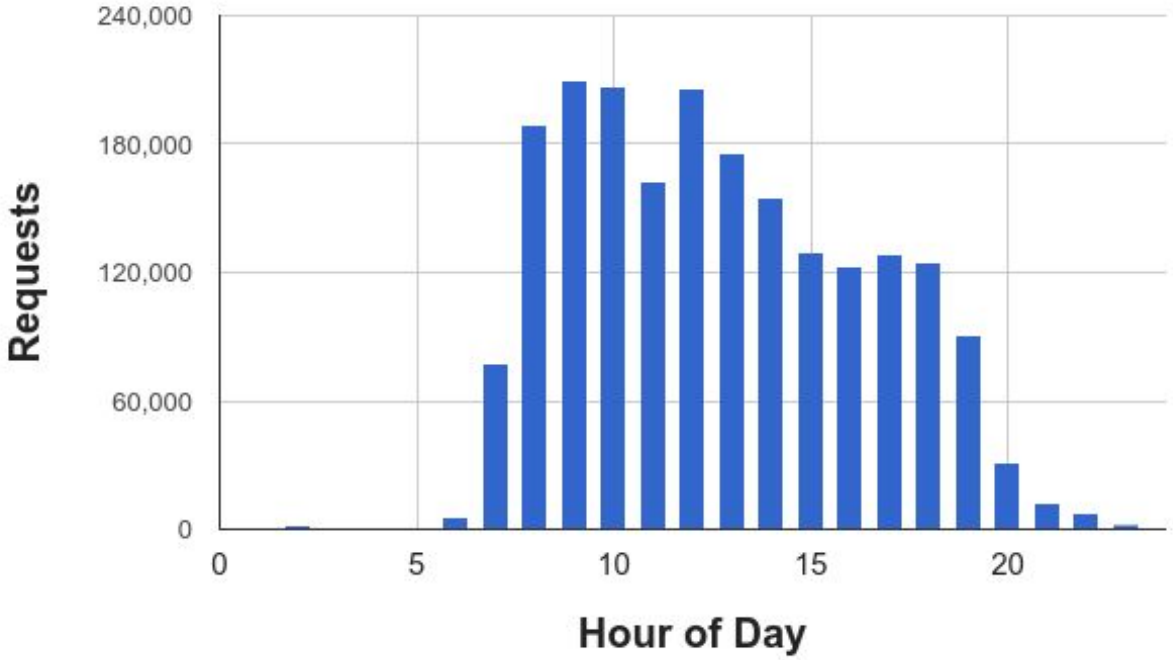
05.10.2015 First Monday of October

Prescriptions: 174,421



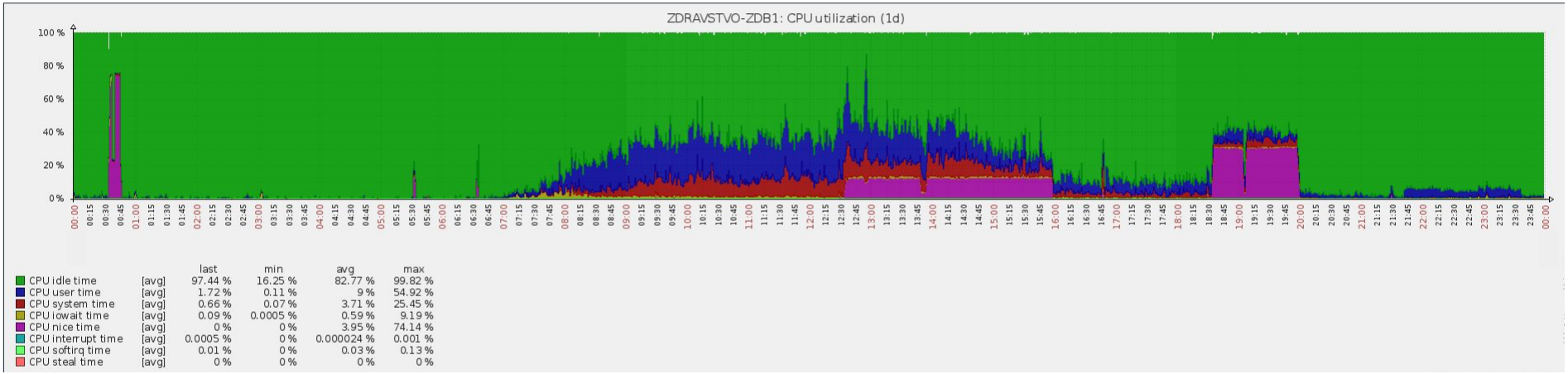
05.10.2015 First Monday of October

Requests: 2,045,205



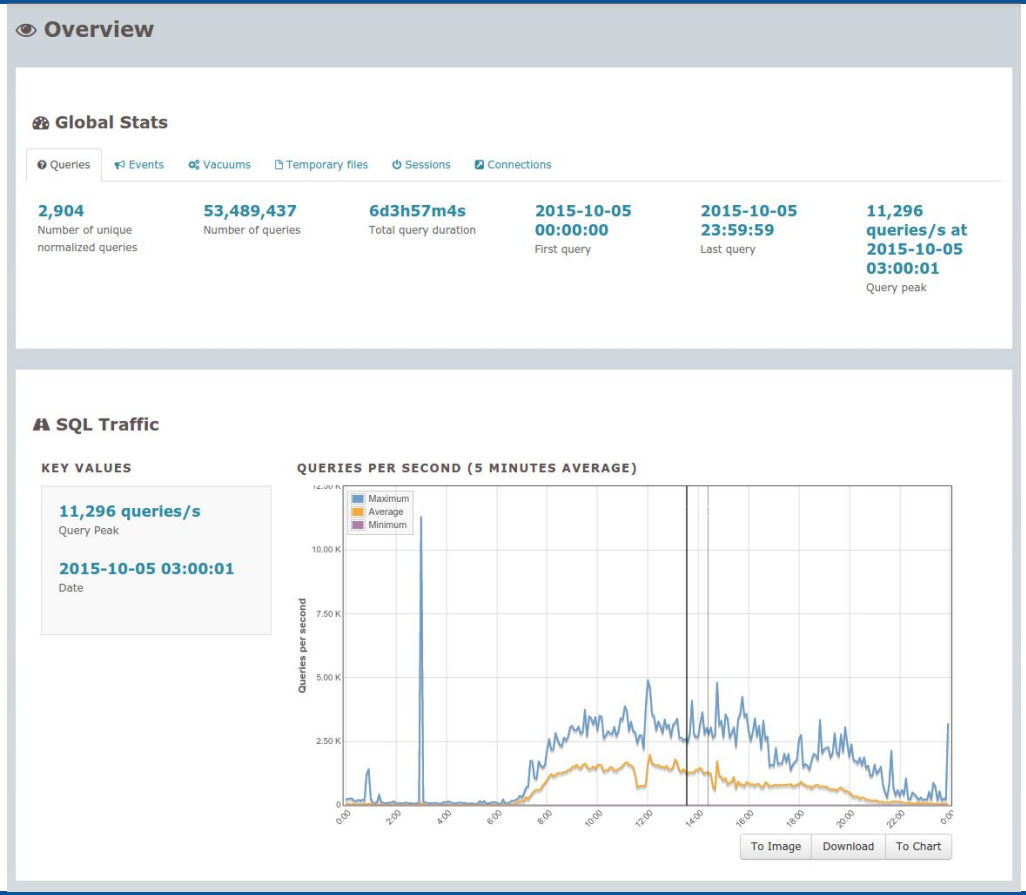
05.10.2015 First Monday of October

DB server - CPU Utilization



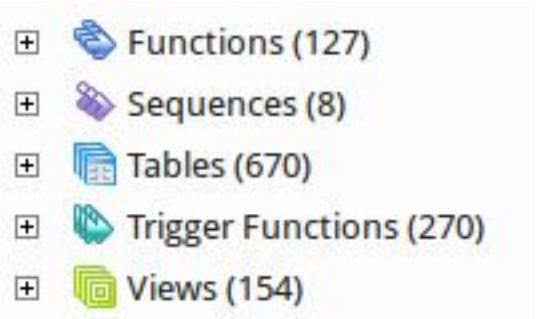
05.10.2015 First Monday of October

PgBadger



System size

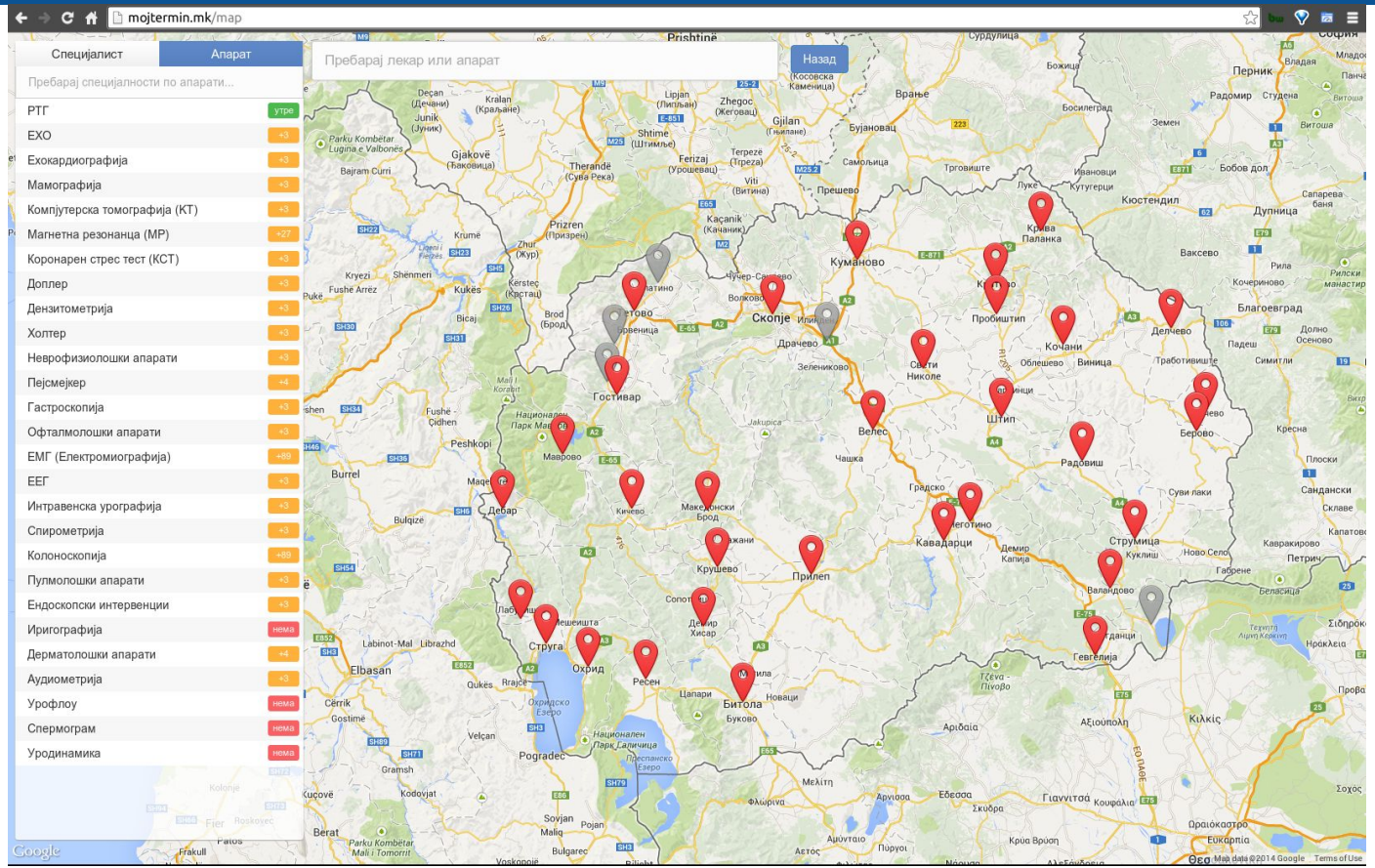
- Database



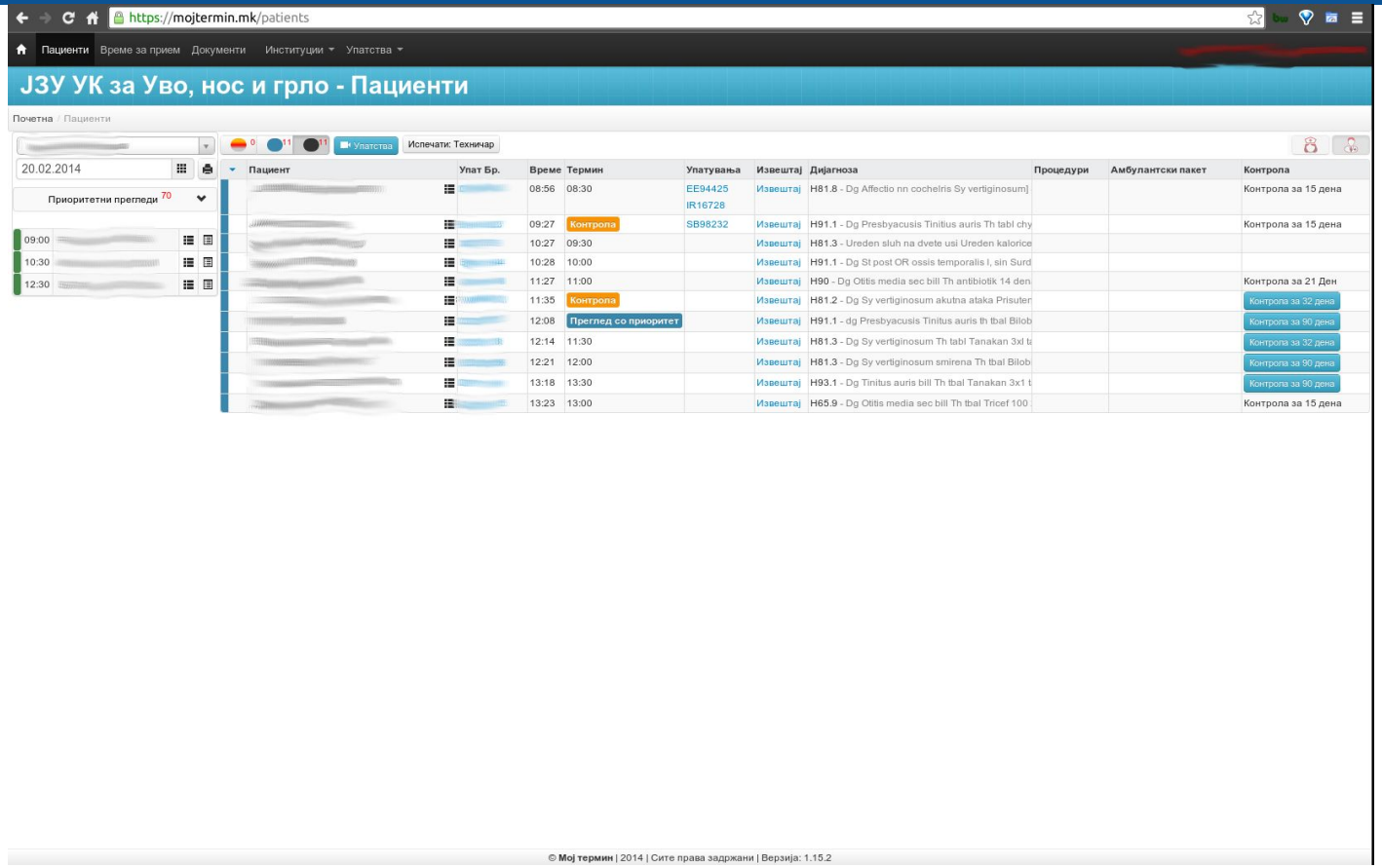

```
health=# select pg_size_pretty(pg_database_size('health'));
 pg_size_pretty
----------------
 329 GB
(1 row)
```

- 11 Applications:
  - core, health, services, services-portal, public-portal, drugs, jobs, sms, mammo, fzo, live-dashboard
  - Over 300k LOC: java, javascript, sql, pl/pgsql, bash ...

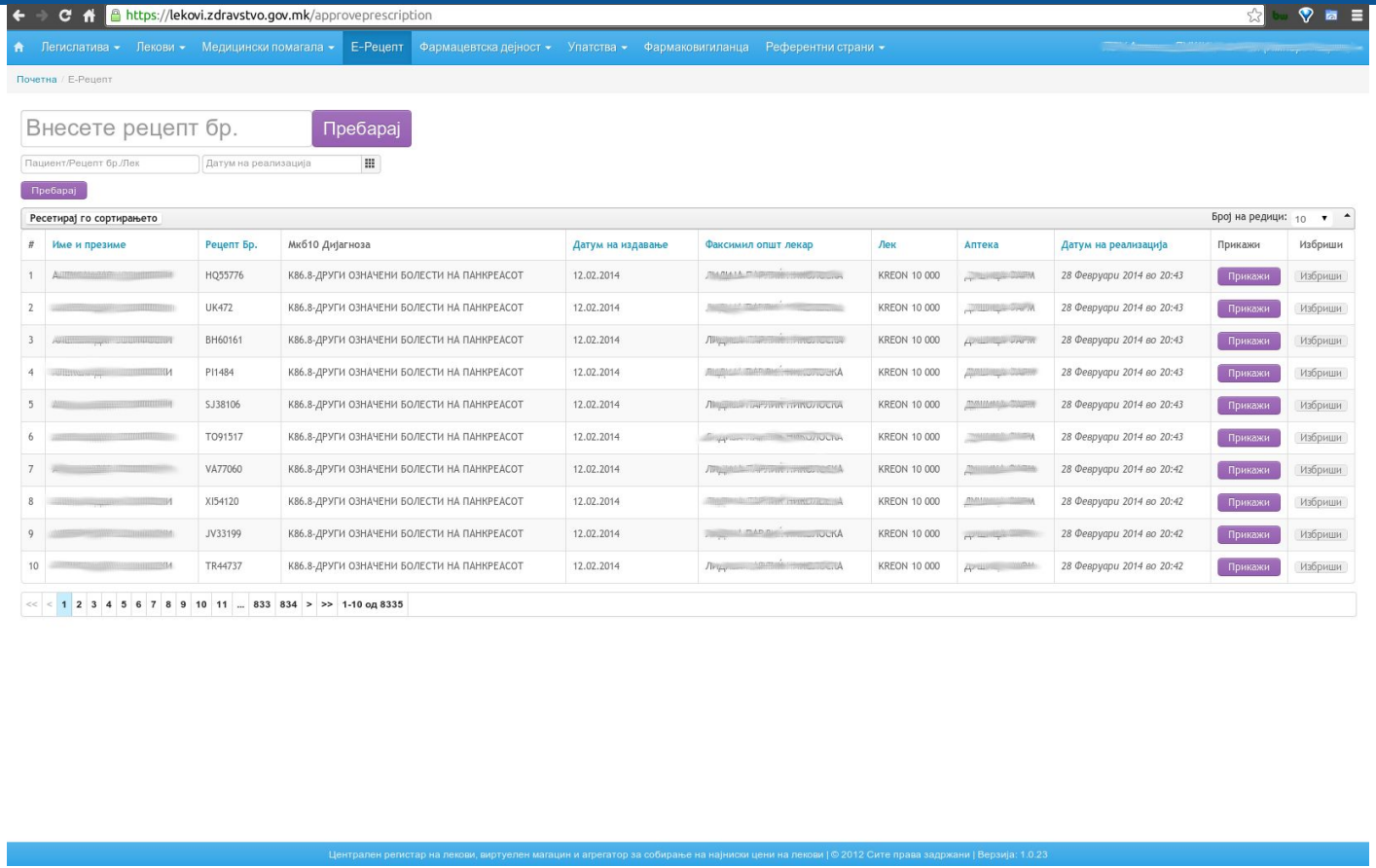
Screenshots



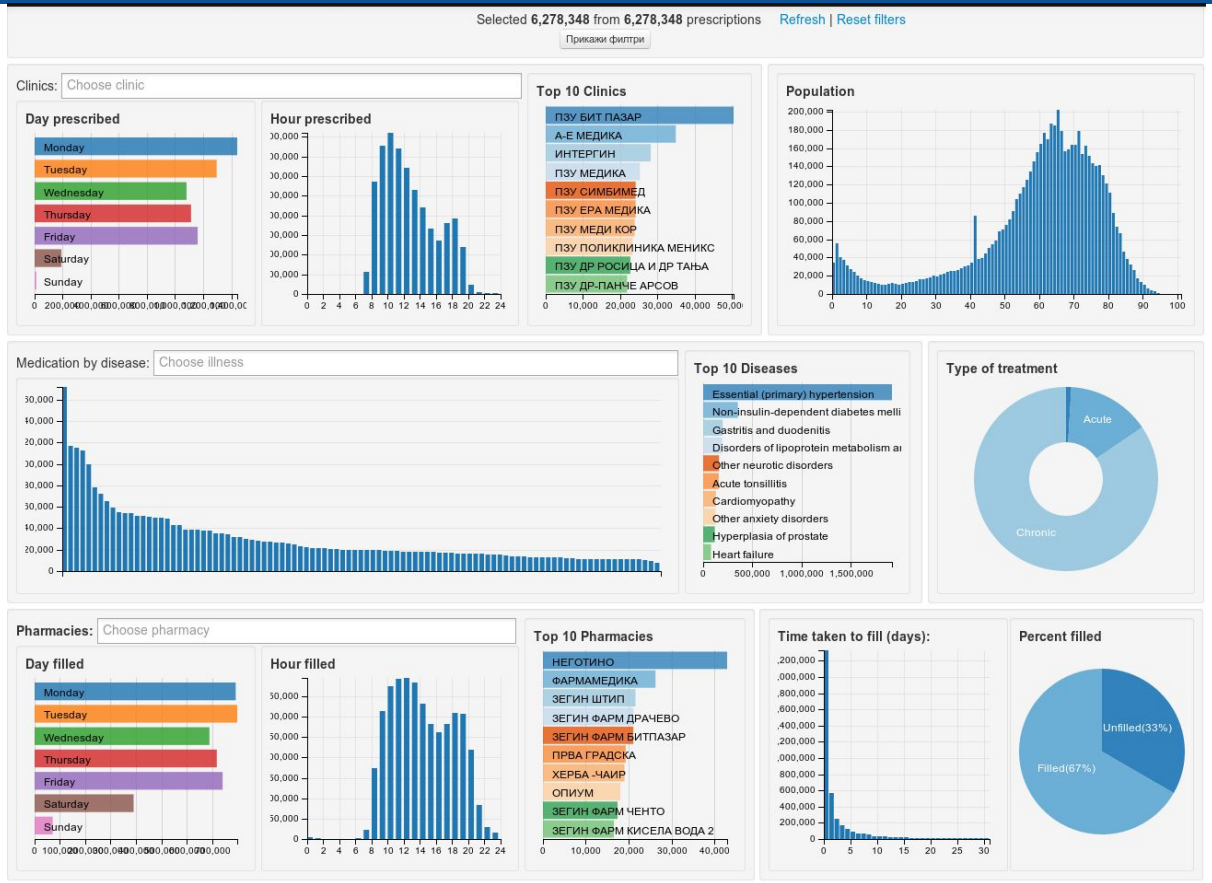
Screenshots



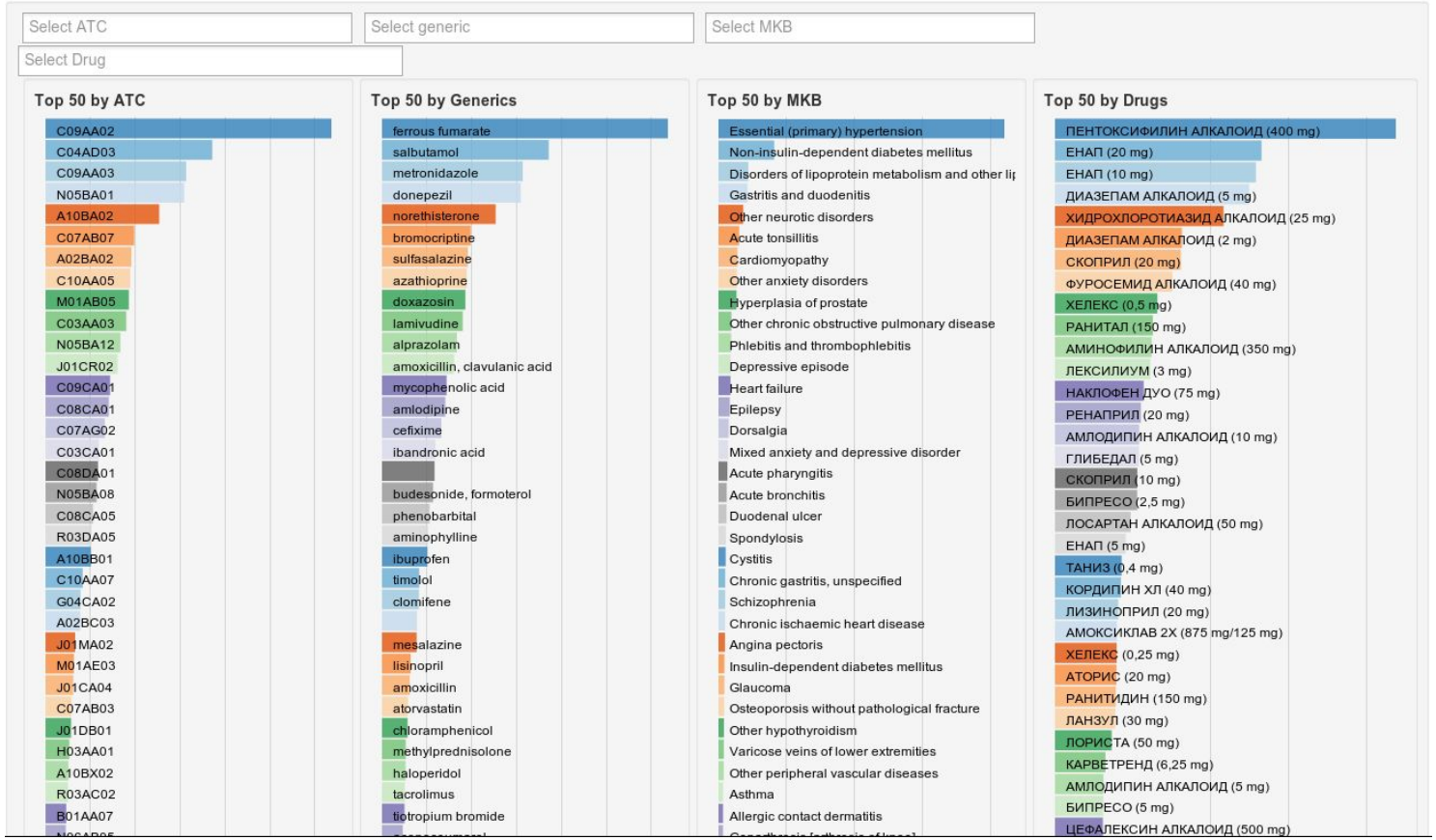
Screenshots



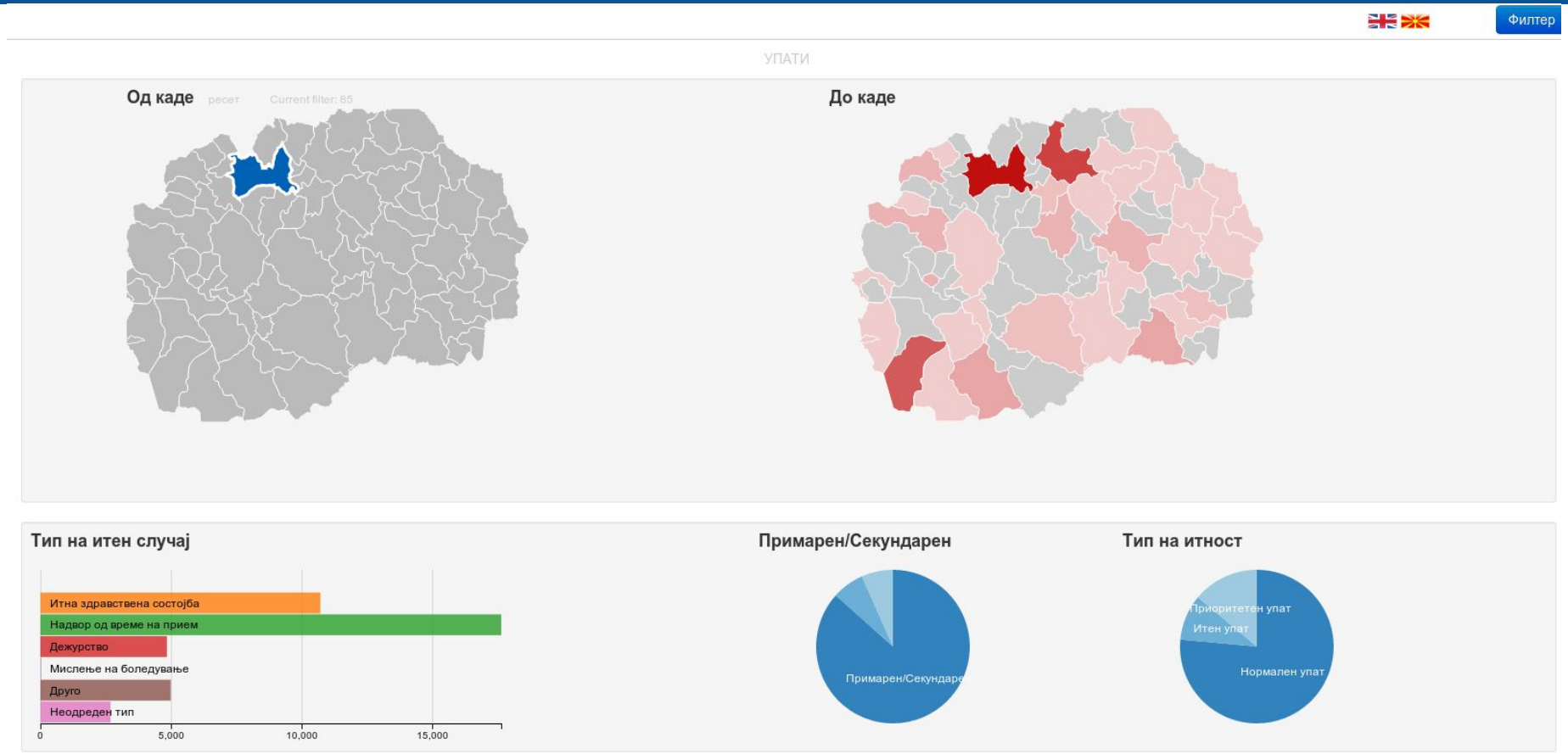
Screenshots



Screenshots



Screenshots



International Recognition: Jan 2015

Home ▸ International Indexes ▸ Euro Health Consumer Index

Euro Health Consumer Index 2014

http://www.healthpowerhouse.com/files/EHCI_2014/EHCI_2014_press_release.pdf

the EHCI 2014 shows competition at the top getting much harder.

Bronze medallists are Norway at 851 points; the very high *per capita* spend on healthcare services finally paying off! Finland (4th, 846 points) has made a remarkable advance, and seems to have rectified its traditional waiting time problems! Denmark (5th, 836 points) has shown a continuous rise since the start of the comparisons.

Some eastern European EU member systems are doing surprisingly well, particularly the Czech Republic and Estonia, considering their much smaller healthcare spend in Purchasing Power Parity (PPP) adjusted dollars per capita. The FYR Macedonia is making the most remarkable advance in the EHCI scoring of any country in the history of the Index, from 27th to 16th place, largely due to more or less eliminating waiting lists by implementing their real time e-Booking system!

Some key conclusions

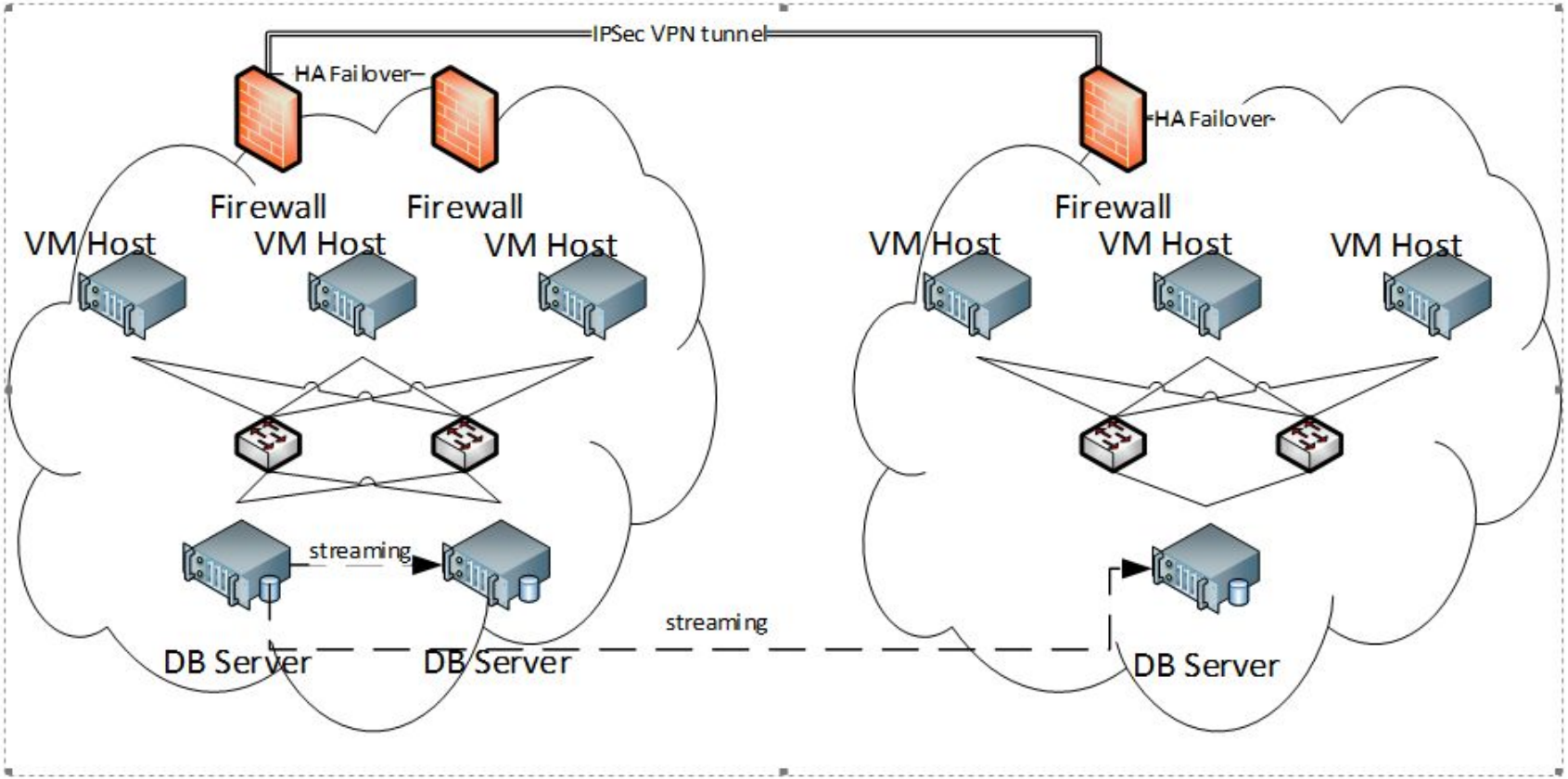
Consumer and patient rights are improving. In a growing number of European countries there is healthcare legislation explicitly based on patient rights and a functional access to your own medical record is becoming standard. Hospital/clinic catalogues with quality ranking used to be confined to two – three countries for years; the 2014 number of nine countries hopefully is a sign that something is happening in this area, supporting active consumer choice. Medical travel supported by the new EU patient mobility directive can accelerate the demand for performance transparency.

- The financial crisis has resulted in a slight but noticeable increase of *inequity* of healthcare services across Europe
- There is a widening performance gap between wealthy and poor European countries
- Overall, medical treatment results keep improving
- Delays and/or restrictiveness have been common on the introduction of novel pharmaceuticals.

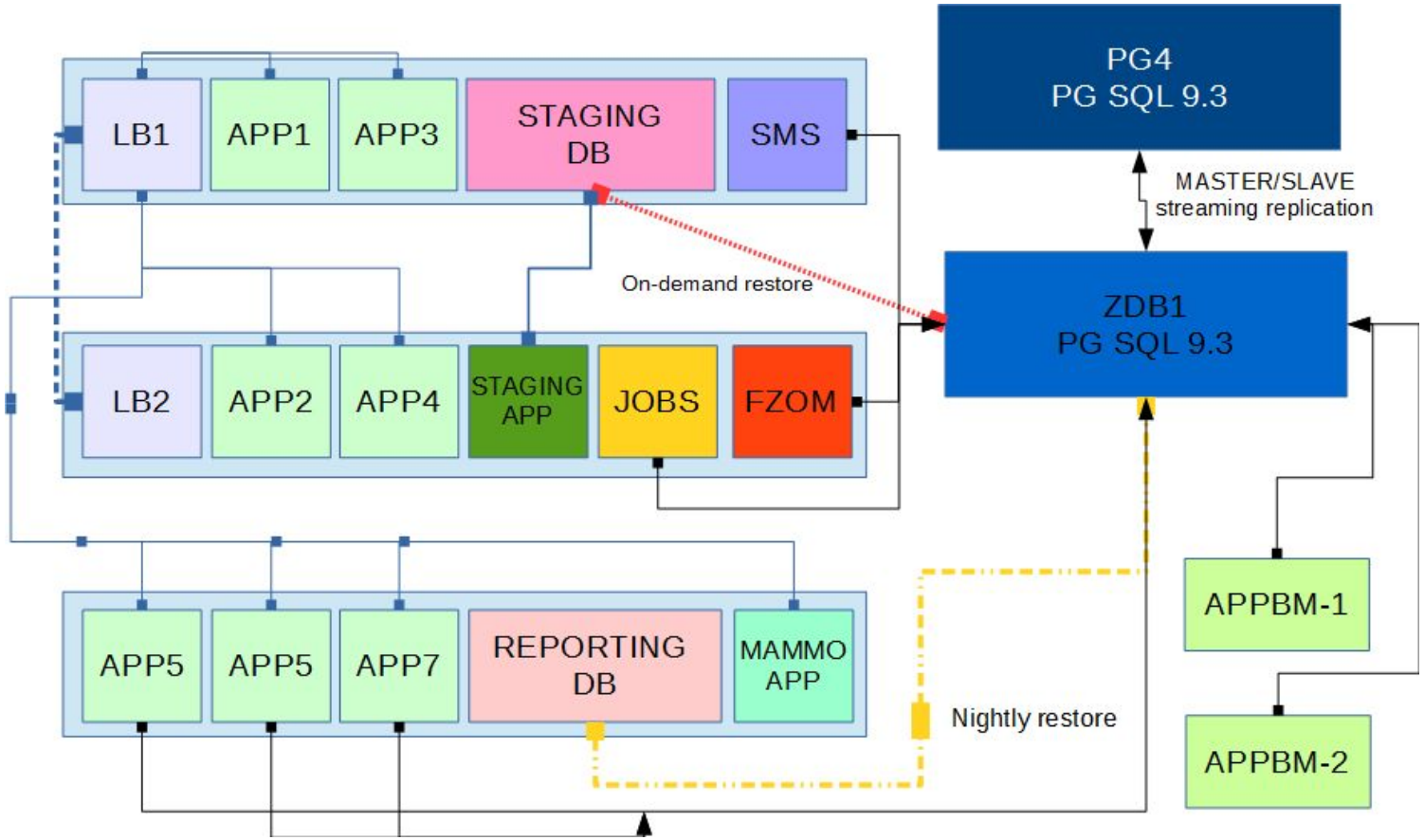
Agenda

- Overview
- **Architecture**
- Hardware
- Replication
- Query Optimization
- Development
- Monitoring
- Critical situations

Architecture: Two data centers



Architecture: Primary data center



Architecture: Server list

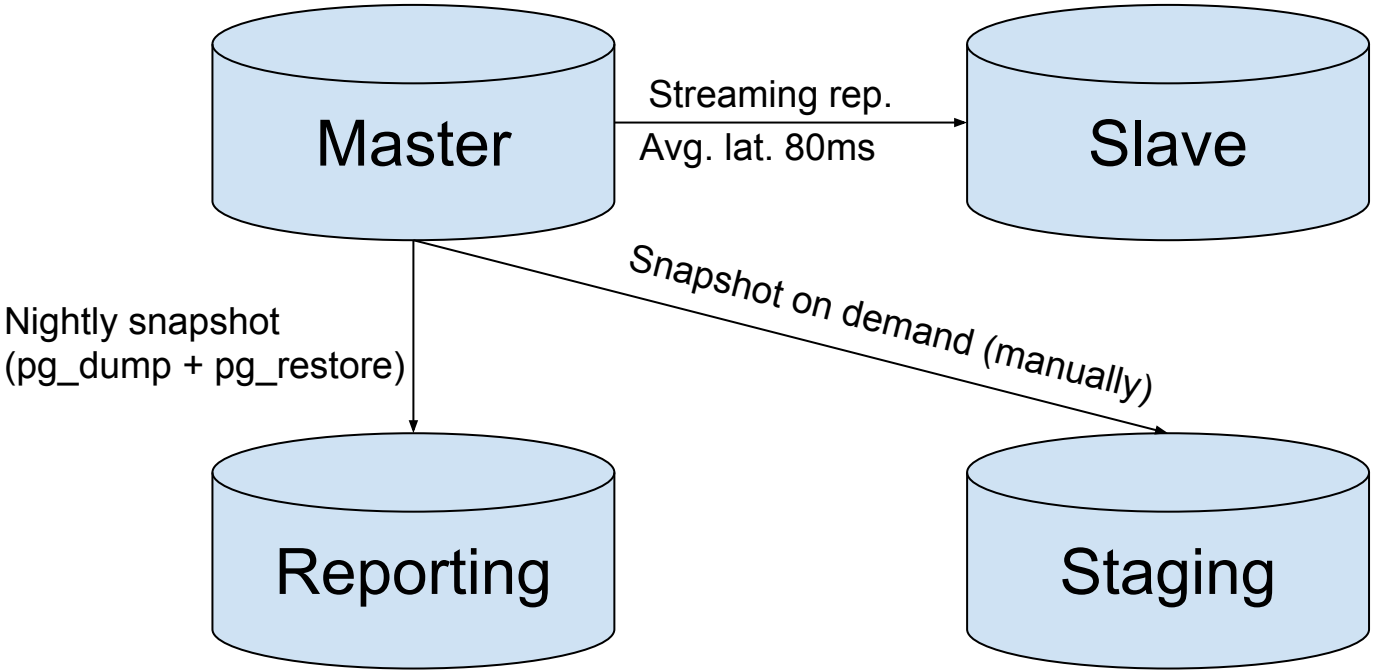
Name	Hardware	Function	Module
ZDB1	DELL 920 Series	DB	Master DB
PG4	HP DL380 G7	DB	Slave DB
STAGING DB	VM	DB	STAGING
REPORTING DB	VM	DB	REPORTING

Name	Hardware	Function	Module
APP1	VM	APP	Main medical app + ucinok
APP2	VM	APP	Main medical app
APP3	VM	APP	Main medical app
APP4	VM	APP	Main medical app
APP5	VM	APP	Public portal
APP6	VM	APP	Public portal
APP7	VM	APP	Documentation for services
MAMMO	VM	APP	Mammography app

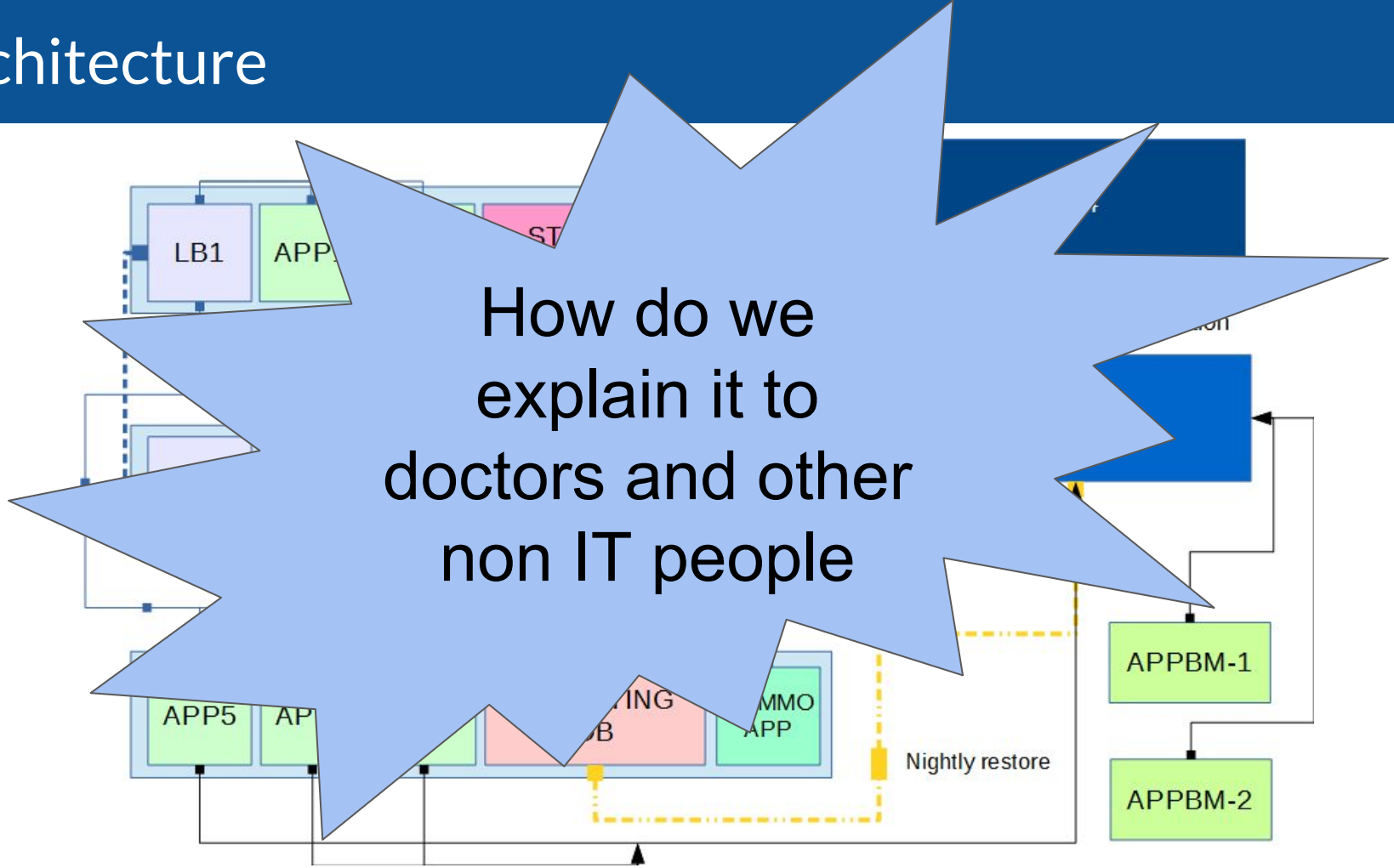
Name	Hardware	Function	Module
APP1BM	IBM x3950 Series	APP	API
APP2BM	IBM x3950 Series	APP	API

Name	Hardware	Function	Module
SMS	VM	Special	SMS integration with Mobile Providers
FZOM	VM	Special	Integration with Fund for Health
JOBS	VM	Special	Job processor
LB1	VM	Special	Primary LB
LB2	VM	Special	Secondary LB

DB Architecture



Architecture



How do we explain it to doctors and other non IT people

Reporting and BI module

Public portal

Web services

Drugs register

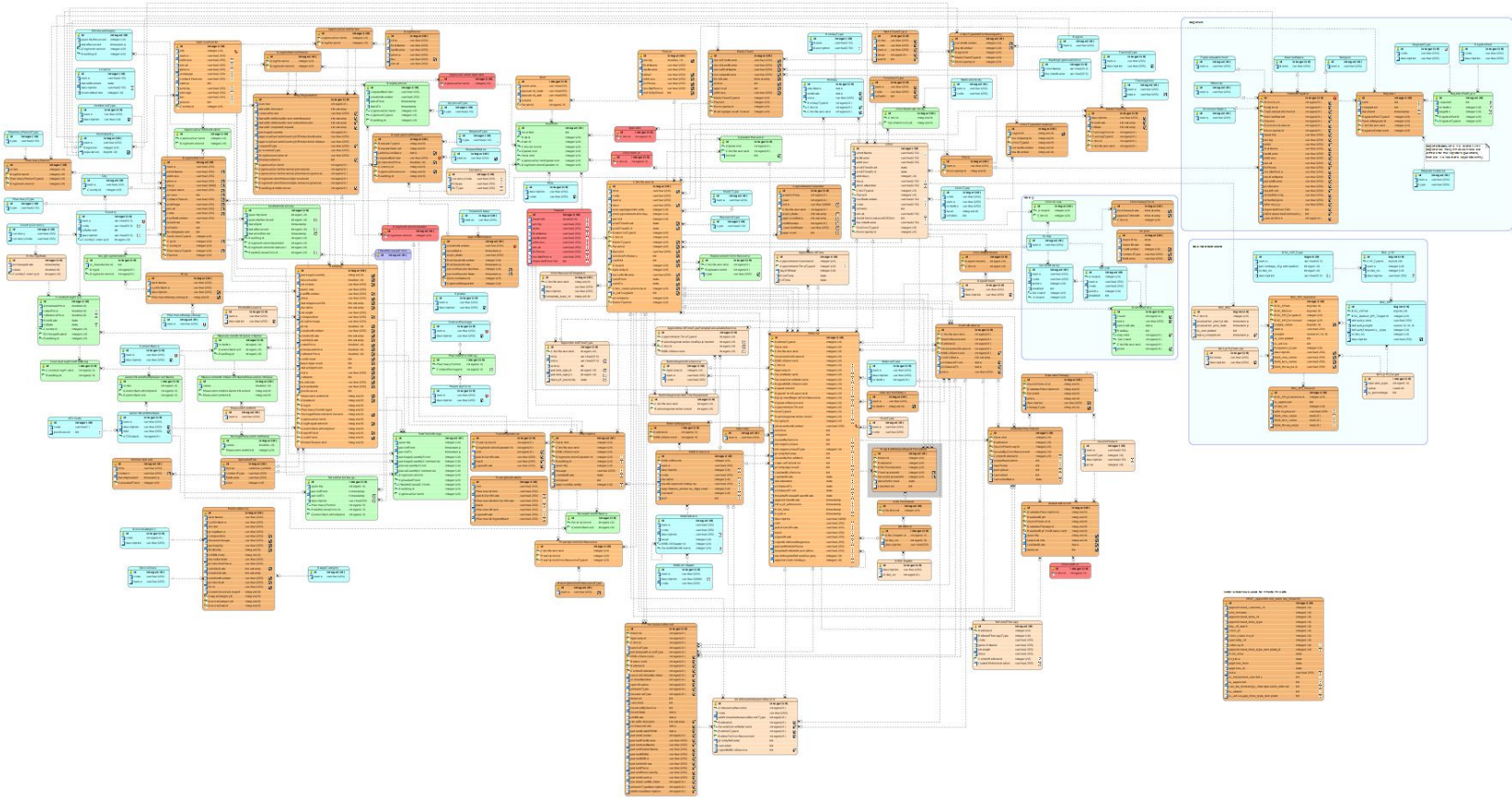
Main web app

Integrated Database

DB Design

- DB Design is done “by the book” for OLTP
- Just a few materialized views/tables
- Normalization using 3NF
- We follow best practice design using FKs, unique and check constraints, indexes, triggers, views
- For reporting we use new SQL features: window functions, recursive queries etc.

DB Design: ER Diagram



Agenda

- Overview
- Architecture
- **Hardware**
- Replication
- Query Optimization
- Development
- Caching
- Monitoring
- Critical situations

Hardware

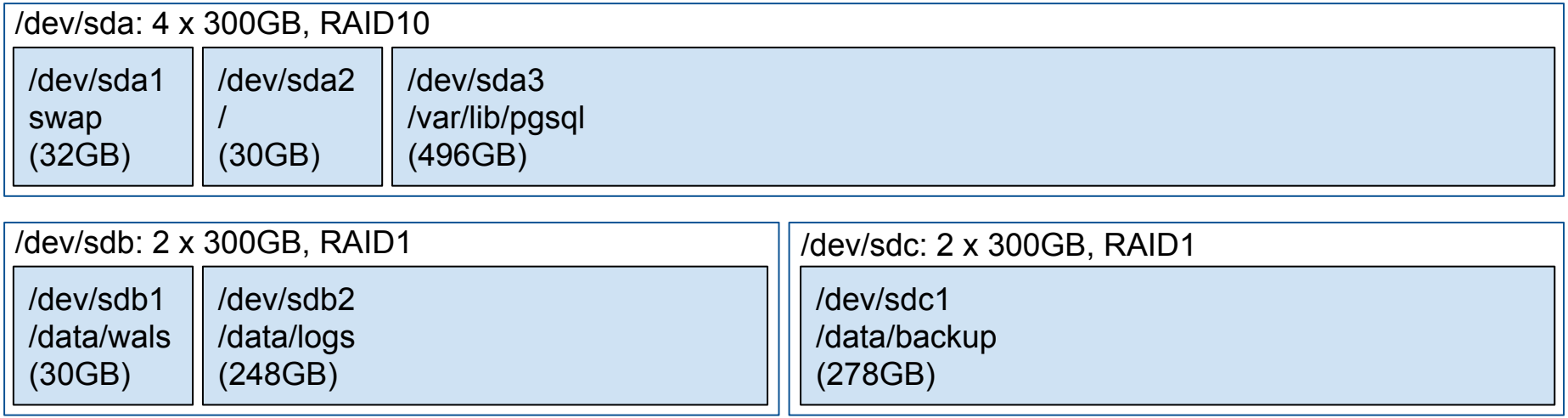
Current master DB Server specs:

CPU: 16 HT cores (32 vCPU cores), 2.66GHz (4xE7-8837)
RAM: 128 GB DDR3 RDIMM
HDD: 8 x 300GB 15K RPM SAS 6Gbps
RAID Controller, 1Gb NV Cache, RAID 0/1/5/10
1Gbps direct interconnection with slave DB server

New master DB Server specs:

CPU: 48 HT cores (32 vCPU cores), min. 2GHz
RAM: 256 GB DDR3 RDIMM, 1600MT/s
HDD:
2 x 1TB 7.2K RPM Near-Line SAS 6Gbps
4 x 300GB 15K RPM SAS 6Gbps
4 x 400GB Solid State Drive SAS Mixed Use MLC 12Gbps
RAID Controller, 2Gb NV Cache, RAID 0/1/5/10
Upgrade to 10 Gbit network

Filesystems and Disk Setup



- Currently using ext4 on all partitions
- db partition used to be XFS, reformatted due to problems with high kernel time and segfaults (did not investigate further)

postgresql.conf

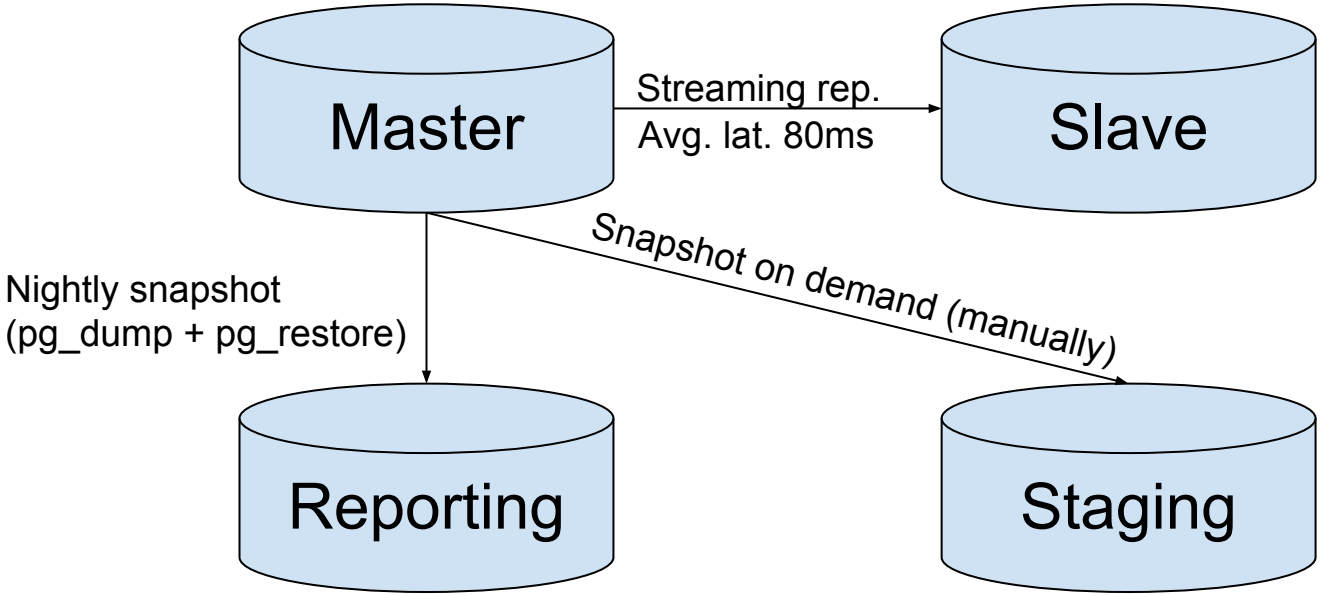
```
#37.5% from full system mem (25%+25%/2)
shared_buffers = 49152MB

fsync = off
#not acceptable
#really awful performance on the first master
#issues on the current master
#to be changed on the new master server

checkpoint_segments = 32
checkpoint_timeout = 10min
checkpoint_completion_target = 0.75

#follow best practices from docs
cpu_tuple_cost = 0.0030
cpu_index_tuple_cost = 0.0010
cpu_operator_cost = 0.0005
```

```
#Prevent many joins to slow the optimizer
from_collapse_limit = 8
join_collapse_limit = 8

#75% of system RAM, for optimized caching
effective_cache_size = 98304MB

#DPDP requirements
log_connections = on
log_disconnections = on
log_statement = 'all'
log_min_duration_statement = 0
log_rotation_age = 6h

autovacuum = on
autovacuum_max_workers = 30
autovacuum_naptime = 20min
autovacuum_vacuum_threshold = 5000
autovacuum_analyze_threshold = 4000
```

Agenda

- Overview
- Architecture
- Hardware
- **Replication**
- Query Optimization
- Development
- Monitoring
- Critical situations

DB Replication



- When slave gets out of sync, it is rsync-ed manually
- Promotion of slave is manual (zabbix monitors the master DB and alerts for errors via SMS)

Agenda

- Overview
- Architecture
- Hardware
- Replication
- **Query Optimization**
- Development
- Monitoring
- Critical situations

Query Optimization

- Workflow
  - Optimize SELECT queries
- Case study
  - The most significant optimization we've done
  - Two months preparation

Query Optimization: Workflow for optimizing SELECTs

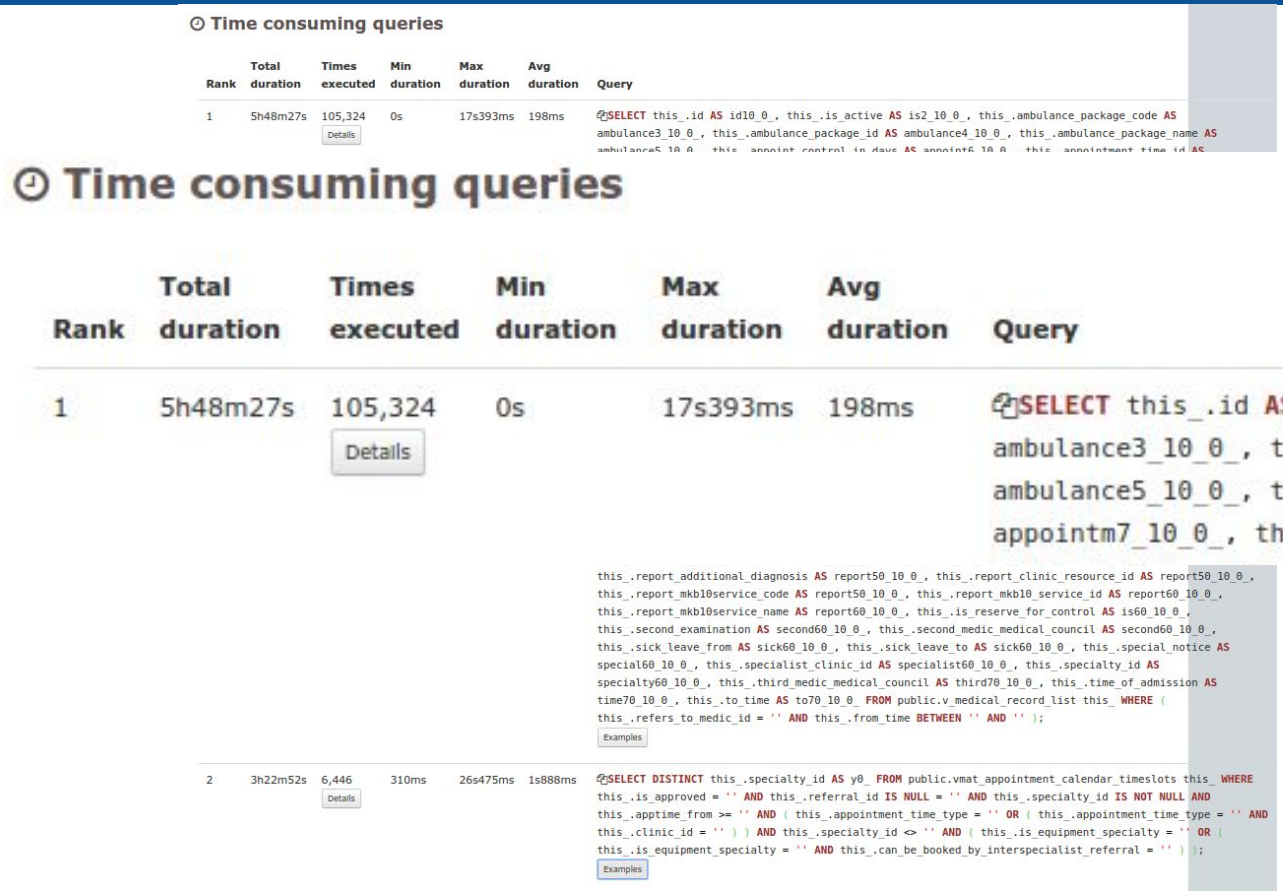
- Step 1: pgBadger
  - ***Time consuming queries*** report
  - Select queries that look ***non-optimal***
- Step 2: Explore the execution plan (explain analyze)
  - We use the excellent **explain.depesz.com** visualization tool
- Step 3: Try optimizing the query
  - Indexes
  - Eliminate/use a cheaper query (app specific - greatest impact)
  - Rewrite the query: Very rarely, mainly due to **from_collapse_limit** and **join_collapse_limit** values
  - Denormalization (triggers)

Query Optimization: Step 1: pgBadger

Time consuming queries

Rank	Total duration	Times executed	Min duration	Max duration	Avg duration	Query
1	5h48m27s	105,324 Details	0s	17s393ms	198ms	SELECT this_.id AS id10_0_, this_.is_active AS is2_10_0_, this_.ambulance_package_code AS ambulance3_10_0_, this_.ambulance_package_id AS ambulance4_10_0_, this_.ambulance_package_name AS ambulance5_10_0_, this_.appoint_control_in_days AS appoint6_10_0_, this_.appointment_time_id AS appointm7_10_0_, this_.appropriately_referred AS appropri8_10_0_, this_.clinic_full_name AS clinic9_10_0_, this_.consultative_opinion_repeating_time AS consult10_10_0_, this_.control_clinic_resource_id AS control10_10_0_, this_.control_date AS control10_10_0_, this_.control_referral_intervention_status AS control10_10_0_, this_.embg AS embg10_0_, this_.is_emergency_case AS is10_10_0_, this_.equipment_report_clinic_resource AS equipment10_10_0_, this_.examinations AS examina10_10_0_, this_.ezbo AS ezbo10_0_, this_.faksimil AS faksimil10_0_, this_.fifth_medic_medical_council AS fifth20_10_0_, this_.final_consultative_opinion AS final20_10_0_, this_.finding AS finding10_0_, this_.first_medic_medical_council AS first20_10_0_, this_.fourth_medic_medical_council AS fourth20_10_0_, this_.from_time AS from20_10_0_, this_.insurance_type_id AS insurance20_10_0_, this_.intervention_number AS interve20_10_0_, this_.intervention_status AS interve20_10_0_, this_.issued_by_medic_id AS issued20_10_0_, this_.is_issued_by_rural_doctor AS is30_10_0_, this_.medic_full_name AS medic30_10_0_, this_.mkb10_code AS mkb30_10_0_, this_.mkb10_name AS mkb30_10_0_, this_.not_appropriately_referred_description AS not30_10_0_, this_.operator_resource_id AS operator30_10_0_, this_.opinion_of_the_consultative_body AS opinion30_10_0_, this_.oportunity_for_rehabilitation_note AS oportunity30_10_0_, this_.parent_referral_id AS parent30_10_0_, this_.first_name AS first30_10_0_, this_.last_name AS last40_10_0_, this_.mobile_phone AS mobile40_10_0_, this_.patient_id AS patient40_10_0_, this_.priority_approved AS priority40_10_0_, this_.priority_note AS priority40_10_0_, this_.is_priority_referral AS is40_10_0_, this_.is_private_mkb10 AS is40_10_0_, this_.real_intervention_status AS real40_10_0_, this_.reason_for_priority AS reason40_10_0_, this_.referral_date_created AS referral40_10_0_, this_.description AS descrip50_10_0_, this_.referral_id AS referral50_10_0_, this_.realized_date AS realized50_10_0_, this_.referral_specialty_id AS referral50_10_0_, this_.referral_type AS referral50_10_0_, this_.referral_type_id AS referral50_10_0_, this_.refers_to_medic_id AS refers50_10_0_, this_.report_additional_diagnosis AS report50_10_0_, this_.report_clinic_resource_id AS report50_10_0_, this_.report_mkb10service_code AS report50_10_0_, this_.report_mkb10_service_id AS report60_10_0_, this_.report_mkb10service_name AS report60_10_0_, this_.is_reserve_for_control AS is60_10_0_, this_.second_examination AS second60_10_0_, this_.second_medic_medical_council AS second60_10_0_, this_.sick_leave_from AS sick60_10_0_, this_.sick_leave_to AS sick60_10_0_, this_.special_notice AS special60_10_0_, this_.specialist_clinic_id AS specialist60_10_0_, this_.specialty_id AS specialty60_10_0_, this_.third_medic_medical_council AS third70_10_0_, this_.time_of_admission AS time70_10_0_, this_.to_time AS to70_10_0_ FROM public.v_medical_record_list this_ WHERE (this_.refers_to_medic_id = '' AND this_.from_time BETWEEN '' AND ''); Examples
2	3h22m52s	6,446 Details	310ms	26s475ms	1s888ms	SELECT DISTINCT this_.specialty_id AS y0_ FROM public.vmat_appointment_calendar_timeslots this_ WHERE this_.is_approved = '' AND this_.referral_id IS NULL = '' AND this_.specialty_id IS NOT NULL AND this_.apptime_from >= '' AND (this_.appointment_time_type = '' OR (this_.appointment_time_type = '' AND this_.clinic_id = '')) AND this_.specialty_id <> '' AND (this_.is_equipment_specialty = '' OR (this_.is_equipment_specialty = '' AND this_.can_be_booked_by_interspecialist_referral = '')); Examples

Query Optimization: Step 1: pgBadger

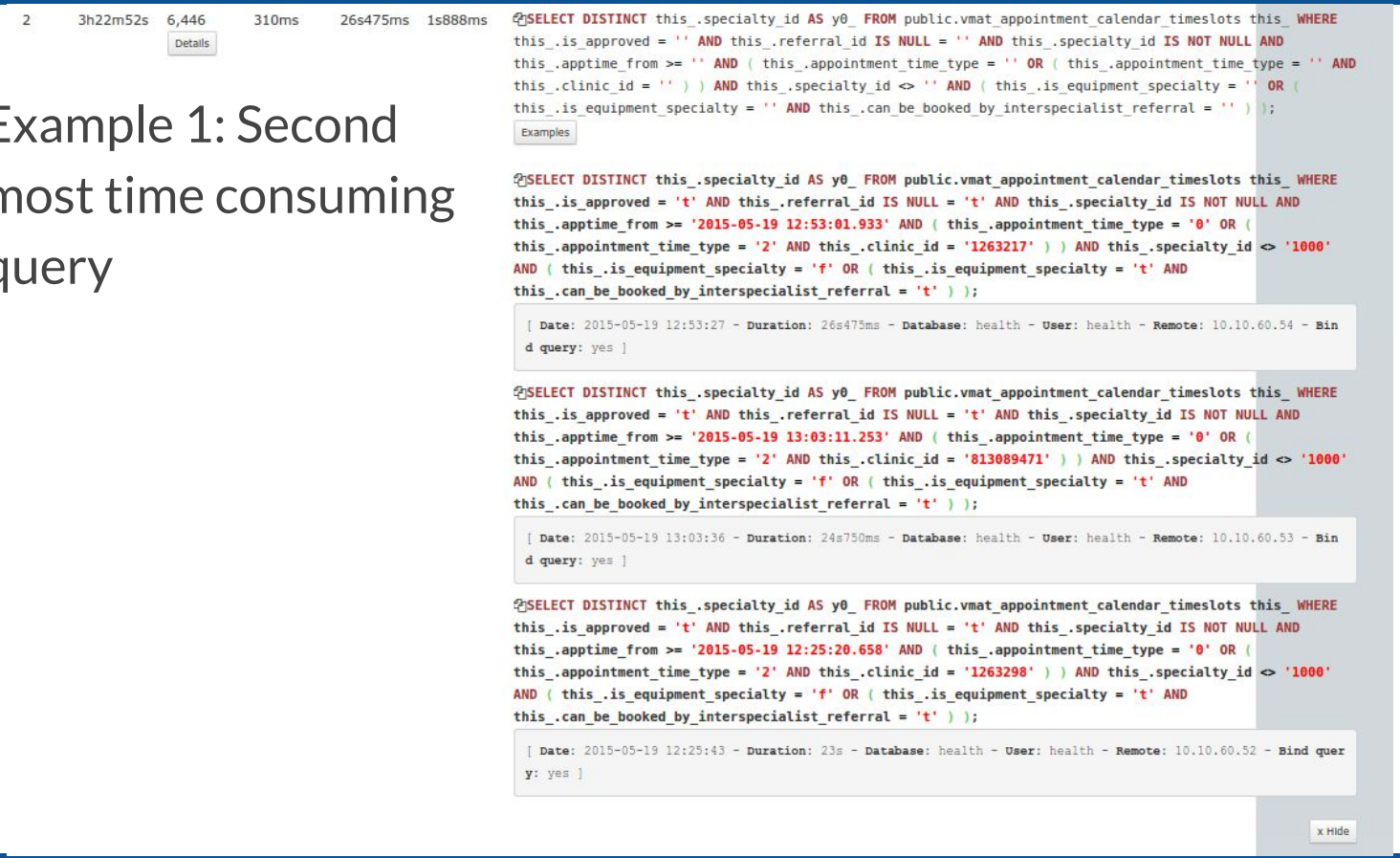


Time consuming queries

Rank	Total duration	Times executed	Min duration	Max duration	Avg duration	Query
1	5h48m27s	105,324 Details	0s	17s393ms	198ms	SELECT this_.id AS id10_0_, this_.is_active AS is2_10_0_, this_.ambulance_package_code AS ambulance3_10_0_, this_.ambulance_package_id AS ambulance4_10_0_, this_.ambulance_package_name AS ambulance5_10_0_, this_.appoint_control_in_days AS appoint6_10_0_, this_.appointment_time_id AS
2	3h22m52s	6,446 Details	310ms	26s475ms	1s888ms	SELECT DISTINCT this_.specialty_id AS y0_ FROM public.vmat_appointment_calendar_timeslots this_ WHERE this_.is_approved = '' AND this_.referral_id IS NULL = '' AND this_.specialty_id IS NOT NULL AND this_.apptime_from >= '' AND (this_.appointment_time_type = '' OR (this_.appointment_time_type = '' AND this_.clinic_id = '')) AND this_.specialty_id <> '' AND (this_.is_equipment_specialty = '' OR (this_.is_equipment_specialty = '' AND this_.can_be_booked_by_interspecialist_referral = '')); Examples

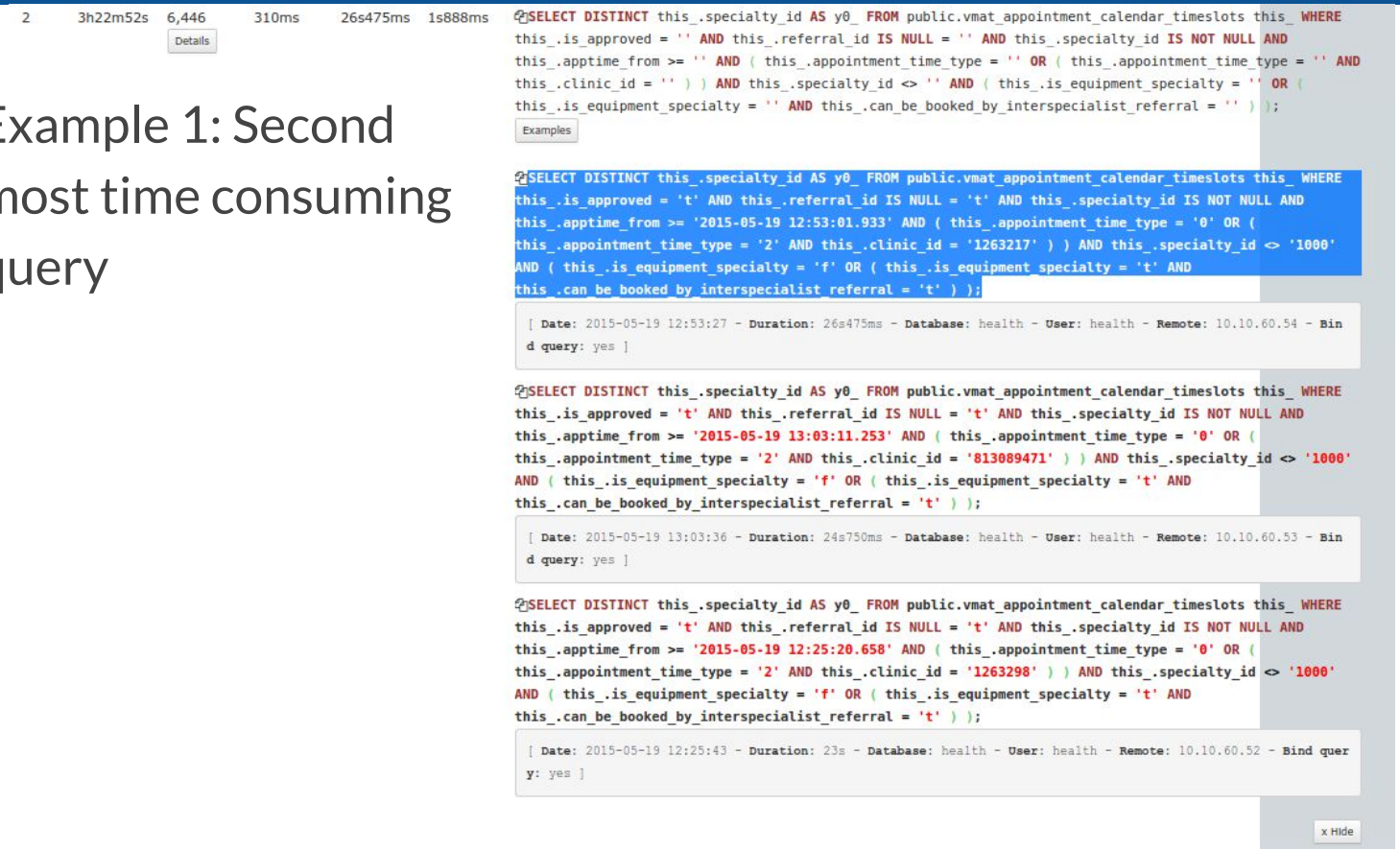
Query Optimization: Step 1: pgBadger

- Example 1: Second most time consuming query



Query Optimization: Step 1: pgBadger

- Example 1: Second most time consuming query



Query Optimization: Step 2: Execution plan

- Example 1: Second most time consuming query

```
explain analyze
SELECT DISTINCT this_.specialty_id AS y0_
FROM public.vmat_appointment_calendar_timeslots this_
WHERE this_.is_approved = 't'
AND this_.referral_id IS NULL = 't'
AND this_.specialty_id IS NOT NULL
AND this_.apptime_from >= '2015-05-19 12:53:01.933'
AND ( this_.appointment_time_type = '0' OR ( this_.appointment_time_type = '2' AND this_.clinic_id = '1263217' ) )
AND this_.specialty_id <> '1000'
AND ( this_.is_equipment_specialty = 'f' OR ( this_.is_equipment_specialty = 't' AND this_.can_be_booked_by_interspecialist_referral = 't' ) );
```

```
HashAggregate  (cost=368163.02..368163.04 rows=8 width=8) (actual time=5336.278..5336.292 rows=85 loops=1)
  ->  Seq Scan on vmat_appointment_calendar_timeslots this_  (cost=0.00..367665.07 rows=995888 width=8) (actual time=0.046..5176.554 rows=949006 loops=1)
        Filter: (is_approved AND (referral_id IS NULL) AND (specialty_id IS NOT NULL) AND ((NOT is_equipment_specialty) OR (is_equipment_specialty AND can_be_booke
        Rows Removed by Filter: 10878120
Total runtime: 5336.350 ms
```

#	exclusive	inclusive	rows x	rows	loops	node
1.	5,352.570	5,352.570	↓ 10.6	85	1	➡ HashAggregate (cost=368,163.02..368,163.04 rows=8 width=8) (actual time=5,352.557..5,352.570 rows=85 loops=1) -> Seq Scan on vmat_appointment_calendar_timeslots this_ (cost=0.00..367665.07 rows=995888 width=8) (actual time=0.048..5193.089 rows=949 Filter: (is_approved AND (referral_id IS NULL) AND (specialty_id IS NOT NULL) AND ((NOT is_equipment_specialty) OR (is_equipment_spec Rows Removed by Filter: 10878119 Total runtime: 5352.644 ms

Query Optimization: Step 2: Execution plan

- Example 1: Second most time consuming query

```
explain analyze
SELECT DISTINCT this_.specialty_id AS y0_
FROM public.vmat_appointment_calendar_timeslots this_
WHERE this_.is_approved = 't'
AND this_.referral_id IS NULL = 't'
AND this_.specialty_id IS NOT NULL
AND this_.apptime_from >= '2015-12-19 12:53:01.933'
AND ( this_.appointment_time_type = '0' OR ( this_.appointment_time_type = '2' AND this_.clinic_id = '1263217' ) )
AND this_.specialty_id <> '1000'
AND ( this_.is_equipment_specialty = 'f' OR ( this_.is_equipment_specialty = 't' AND this_.can_be_booked_by_interspecialist_referral = 't' ) );
```

More restrictive predicate

⇒ Index Only Scan

```
Index Only Scan using ix_vmat_appointment_calendar_timeslots_1 on vmat_appointment_calendar_timeslots this_  (cost=0.14..67079.81 rows=43773 width=8) (actual time=0.026..15.216 rows=9410
  Index Cond: ((apptime_from >= '2015-12-19 12:53:01.933'::timestamp without time zone) AND (referral_id IS NULL) AND (specialty_id IS NOT NULL) AND (is_approved = true))
  Filter: (is_approved AND ((NOT is_equipment_specialty) OR (is_equipment_specialty AND can_be_booked_by_interspecialist_referral)) AND (specialty_id <> 1000::bigint) AND ((appointment_t
  Rows Removed by Filter: 1856
  Heap Fetches: 0
Total runtime: 15.755 ms
```

#	exclusive	inclusive	rows x	rows	loops	node
1.	15.216	15.216	↑ 4.7	9,410	1	→ Index Only Scan using ix_vmat_appointment_calendar_timeslots_1 on vmat_appointment_calendar_timeslots this_ (cost=0.14..67,079.81 rows=43,773 width=8) (actual time=0.026..15.216 rows=9,410 loops=1) Index Cond: ((apptime_from >= '2015-12-19 12:53:01.933'::timestamp without time zone) AND (referral_id IS NULL) AND (specialty_id IS NOT NULL) AND (is_approved = true)) Filter: (is_approved AND ((NOT is_equipment_specialty) OR (is_equipment_specialty AND can_be_booked_by_interspecialist_referral)) AND (specialty_id <> 1000::bigint) AND ((appointment_time_type = 0) OR ((appointment_time_type = 2) AND (clinic_id = 1263217::bigint)))) Rows Removed by Filter: 1856 Heap Fetches: 0

Query Optimization: Step 2: Execution plan

- Example 2: In top 5 time consuming queries

```
explain analyze
SELECT *
FROM public.v_medical_record_list_from_absent_medics this_
WHERE this_.refers_to_medic_clinic_id = '1263079'
AND this_.from_time BETWEEN '2015-05-19 00:00:00'
AND '2015-05-20 00:00:00'
AND this_.specialty_id = '38';
```

Query Optimization: Step 2: Execution plan

```
Nested Loop Left Join  (cost=16.87..684.78 rows=1 width=238) (actual time=64.073..603.278 rows=4 loops=1)
  ->  Nested Loop  (cost=16.81..676.71 rows=1 width=210) (actual time=64.041..603.174 rows=4 loops=1)
        ->  Nested Loop  (cost=16.73..669.78 rows=1 width=169) (actual time=63.986..602.983 rows=4 loops=1)
              ->  Nested Loop  (cost=16.61..373.26 rows=11 width=122) (actual time=0.444..272.924 rows=21865 loops=1)
                    ->  Nested Loop  (cost=16.53..345.02 rows=26 width=80) (actual time=0.225..27.862 rows=6950 loops=1)
                          ->  Bitmap Heap Scan on health_clinic_resources cr  (cost=12.31..16.31 rows=1 width=72) (actual time=0.128..0.199 rows=71 loops=1)
                                Recheck Cond: ((specialty_id = 38::bigint) AND (clinic_id = 1263079::bigint))
                                ->  BitmapAnd  (cost=12.31..12.31 rows=1 width=0) (actual time=0.121..0.121 rows=0 loops=1)
                                      ->  Bitmap Index Scan on ix_clinic_resources_speciality_id  (cost=0.00..4.08 rows=16 width=0) (actual time=0.063..0.063 rows=79 loops=1)
                                            Index Cond: (specialty_id = 38::bigint)
                                      ->  Bitmap Index Scan on ix_clinic_resources_clinic_id  (cost=0.00..8.17 rows=78 width=0) (actual time=0.055..0.055 rows=78 loops=1)
                                            Index Cond: (clinic_id = 1263079::bigint)
                          ->  Bitmap Heap Scan on health_appointment_calendars ac  (cost=4.22..328.45 rows=89 width=16) (actual time=0.039..0.360 rows=98 loops=71)
                                Recheck Cond: (clinic_resource_id = cr.id)
                                ->  Bitmap Index Scan on uq_con_app_calendars_clinic_resource_year_weekyear  (cost=0.00..4.22 rows=89 width=0) (actual time=0.027..0.027 rows=98 loops=71)
                                      Index Cond: (clinic_resource_id = cr.id)
                    ->  Index Scan using ix_appointment_times_appointment_calendar_id on health_appointment_times at  (cost=0.09..1.08 rows=1 width=58) (actual time=0.026..0.034 rows=3 l
                          Index Cond: (appointment_calendar_id = ac.id)
                          Filter: is_absent
                          Rows Removed by Filter: 13
              ->  Index Scan using ix_referrals_appointment_time_id on health_referrals r  (cost=0.11..26.95 rows=1 width=63) (actual time=0.015..0.015 rows=0 loops=21865)
                    Index Cond: (appointment_time_id = at.id)
                    Filter: ((is_active IS TRUE) AND (from_time >= '2015-05-19 00:00:00'::timestamp without time zone) AND (from_time <= '2015-05-20 00:00:00'::timestamp without time zon
                    Rows Removed by Filter: 1
        ->  Index Scan using ix_patients_id on health_patients pa  (cost=0.09..6.93 rows=1 width=49) (actual time=0.042..0.042 rows=1 loops=4)
              Index Cond: (id = r.patient_id)
  ->  Index Scan using auth_persons_pkey on auth_persons p  (cost=0.06..8.06 rows=1 width=36) (actual time=0.016..0.017 rows=1 loops=4)
        Index Cond: (id = cr.person_id)
Total runtime: 603.459 ms
```

Query Optimization: Step 2: Execution plan

#	exclusive	inclusive	rows x	rows	loops	node
1.	0.036	603.278	↓ 4.0	4	1	→ Nested Loop Left Join (cost=16.87..684.78 rows=1 width=238) (actual time=64.073..603.278 rows=4 loops=1)
2.	0.023	603.174	↓ 4.0	4	1	→ Nested Loop (cost=16.81..676.71 rows=1 width=210) (actual time=64.041..603.174 rows=4 loops=1)
3.	2.084	602.983	↓ 4.0	4	1	→ Nested Loop (cost=16.73..669.78 rows=1 width=169) (actual time=63.986..602.983 rows=4 loops=1)
4.	8.762	272.924	↓ 1,987.7	21,865	1	→ Nested Loop (cost=16.61..373.26 rows=11 width=122) (actual time=0.444..272.924 rows=21,865 loops=1)
5.	2.103	27.862	↓ 267.3	6,950	1	→ Nested Loop (cost=16.53..345.02 rows=26 width=80) (actual time=0.225..27.862 rows=6,950 loops=1)
6.	0.078	0.199	↓ 71.0	71	1	→ Bitmap Heap Scan on health_clinic_resources cr (cost=12.31..16.31 rows=1 width=72) (actual time=0.128..0.199 rows=71 loops=1) Recheck Cond: ((specialty_id = 38::bigint) AND (clinic_id = 1263079::bigint))
7.	0.003	0.121	↓ 0.0	0	1	→ BitmapAnd (cost=12.31..12.31 rows=1 width=0) (actual time=0.121..0.121 rows=0 loops=1)
8.	0.063	0.063	↓ 4.9	79	1	→ Bitmap Index Scan on ix_clinic_resources_speciality_id (cost=0.00..4.08 rows=16 width=0) (actual time=0.063..0.063 rows=79 loops=1) Index Cond: (specialty_id = 38::bigint)
9.	0.055	0.055	↑ 1.0	78	1	→ Bitmap Index Scan on ix_clinic_resources_clinic_id (cost=0.00..8.17 rows=78 width=0) (actual time=0.055..0.055 rows=78 loops=1) Index Cond: (clinic_id = 1263079::bigint)
10.	23.643	25.560	↓ 1.1	98	71	→ Bitmap Heap Scan on health_appointment_calendars ac (cost=4.22..328.45 rows=89 width=16) (actual time=0.039..0.360 rows=98 loops=71) Recheck Cond: (clinic_resource_id = cr.id)
11.	1.917	1.917	↓ 1.1	98	71	→ Bitmap Index Scan on uq_con_app_calendars_clinic_resource_year_weekyear (cost=0.00..4.22 rows=89 width=0) (actual time=0.027..0.027 rows=98 loops=71) Index Cond: (clinic_resource_id = cr.id)
12.	236.300	236.300	↓ 3.0	3	6,950	→ Index Scan using ix_appointment_times_appointment_calendar_id on health_appointment_times at (cost=0.09..1.08 rows=1 width=58) (actual time=0.026..0.034 rows=3 loops=6,950) Index Cond: (appointment_calendar_id = ac.id) Filter: is_absent Rows Removed by Filter: 13
13.	327.975	327.975	↓ 0.0	0	21,865	→ Index Scan using ix_referrals_appointment_time_id on health_referrals r (cost=0.11..26.95 rows=1 width=63) (actual time=0.015..0.015 rows=0 loops=21,865) Index Cond: (appointment_time_id = at.id) Filter: ((is_active IS TRUE) AND (from_time >= '2015-05-19 00:00:00'::timestamp without time zone) AND (from_time <= '2015-05-20 00:00:00'::timestamp without time zone) AND (intervention_status_id = 1)) Rows Removed by Filter: 1
14.	0.168	0.168	↑ 1.0	1	4	→ Index Scan using ix_patients_id on health_patients pa (cost=0.09..6.93 rows=1 width=49) (actual time=0.042..0.042 rows=1 loops=4) Index Cond: (id = r.patient_id)
15.	0.068	0.068	↑ 1.0	1	4	→ Index Scan using auth_persons_pkey on auth_persons p (cost=0.06..8.06 rows=1 width=36) (actual time=0.016..0.017 rows=1 loops=4) Index Cond: (id = cr.person_id)

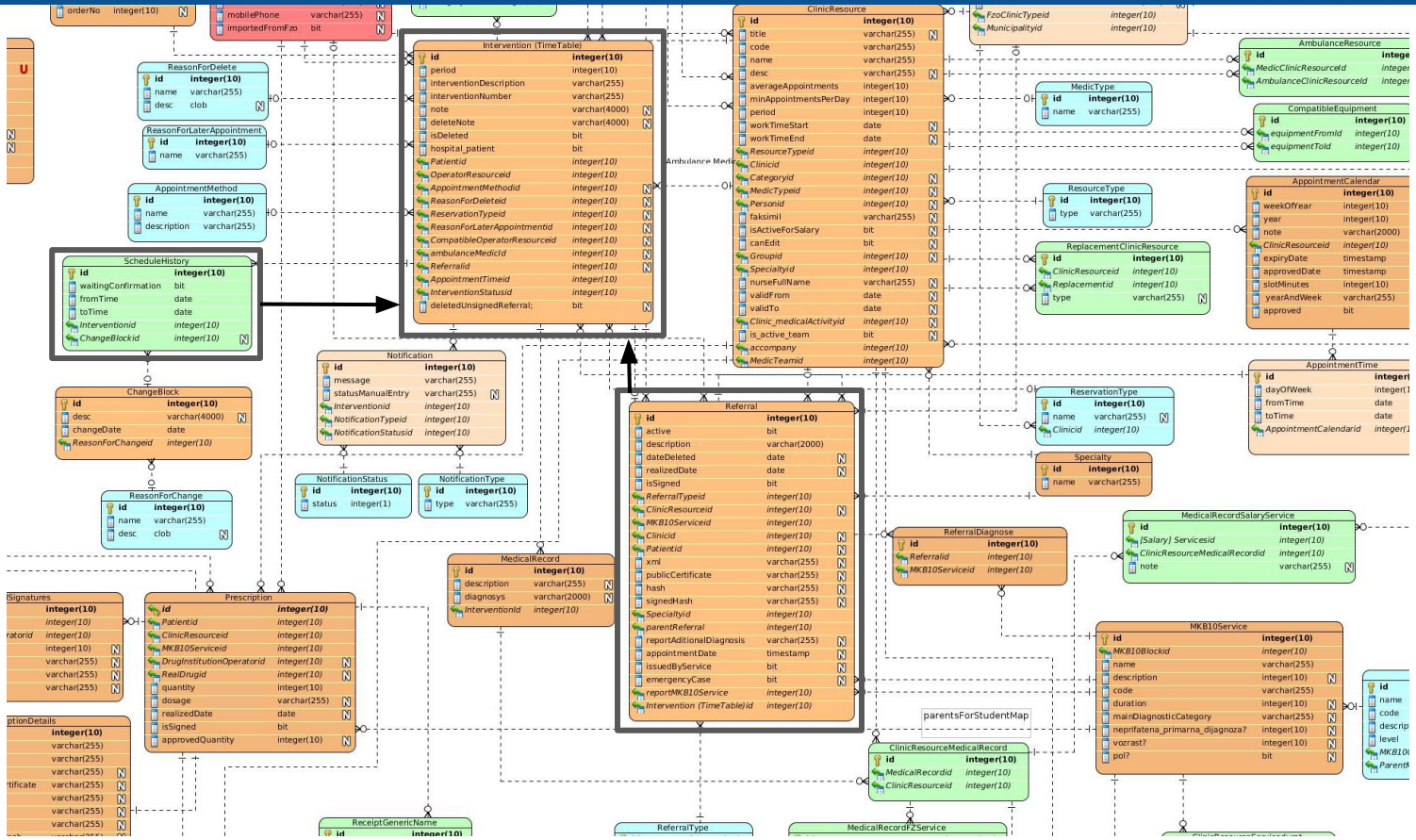
Query Optimization: Workflow for optimizing SELECTs

- Step 1: pgBadger
  - ***Time consuming queries*** report
  - Select queries that look ***non-optimal***
- Step 2: Explore the execution plan (explain analyze)
  - We use the excellent **explain.depesz.com** visualization tool
- Step 3: Try optimizing the query
  - Indexes
  - Eliminate/use a cheaper query (app specific - greatest impact)
  - Rewrite the query: Very rarely, mainly due to **from_collapse_limit** and **join_collapse_limit** values
  - Denormalization (triggers)

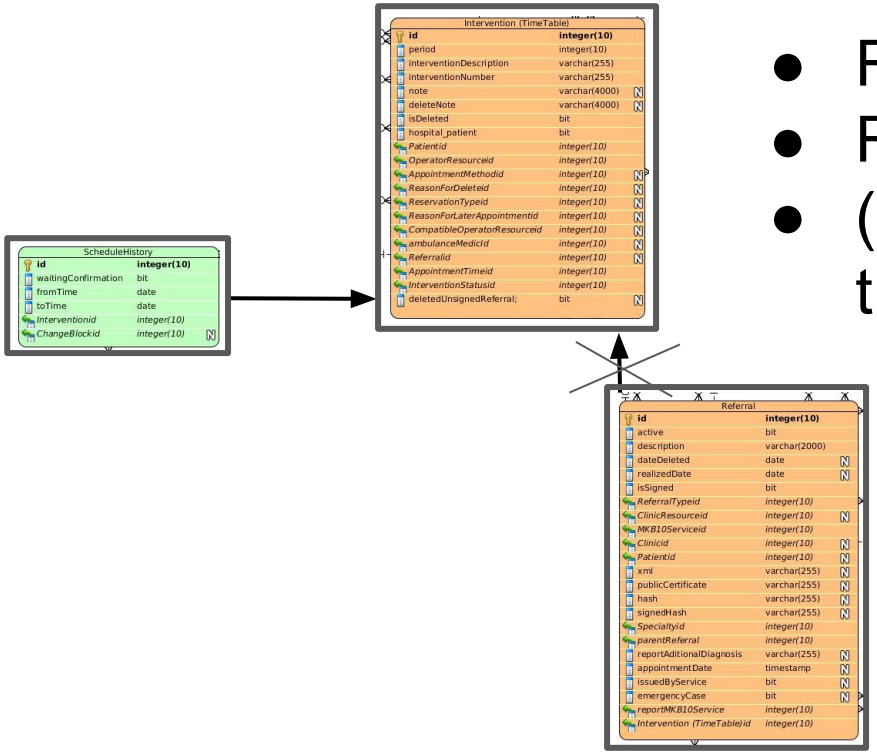
Query Optimization

- Workflow
  - Optimize SELECT queries
- Case study
  - The greatest optimization we've done so far
  - Two months preparation

Case study: The greatest optimization we've done so far

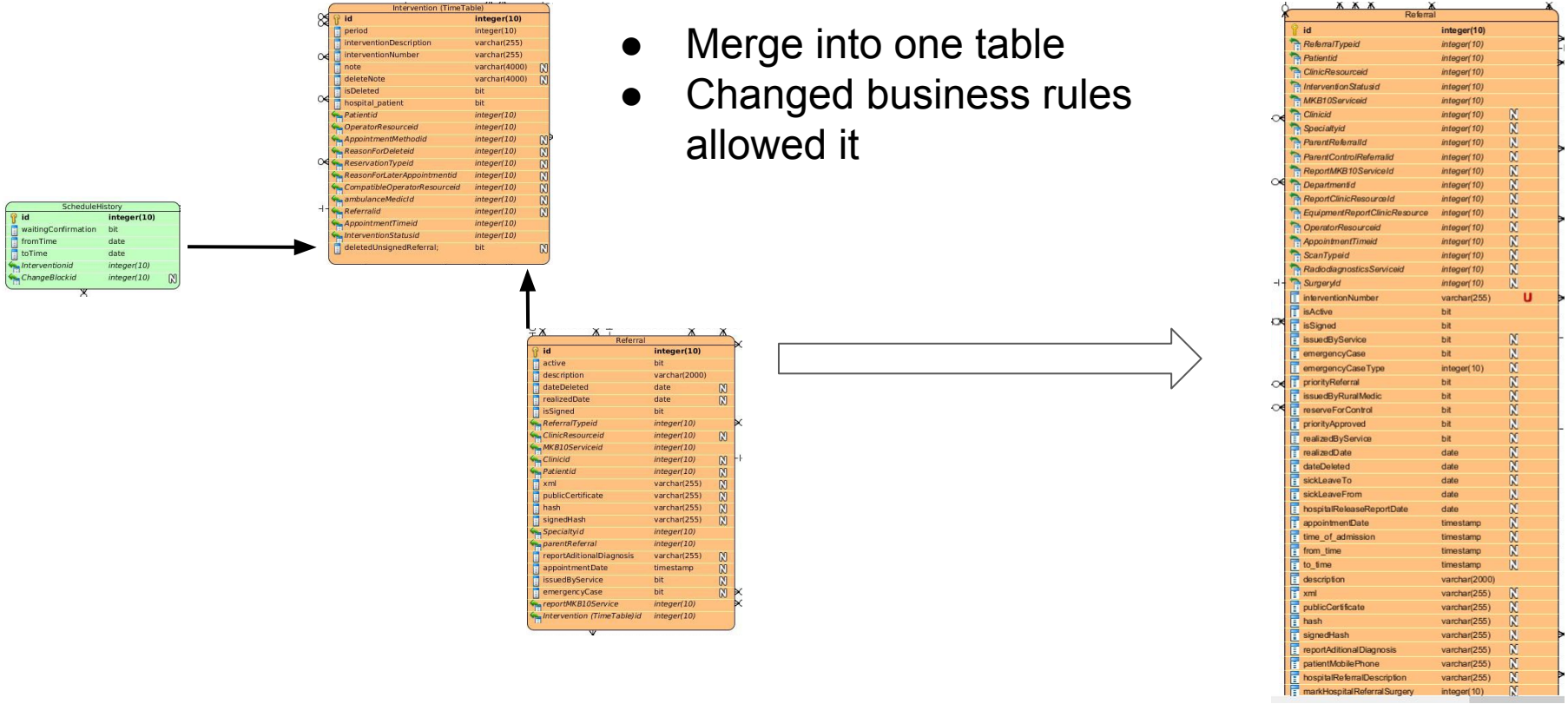


Case study: The greatest optimization we've done so far



- Remove FK Constraint
- Reduced Locking
- (May have been solved by rewriting transactions - time consuming)

Case study: The greatest optimization we've done so far



- Merge into one table
- Changed business rules allowed it

Case study: The greatest optimization we've done so far

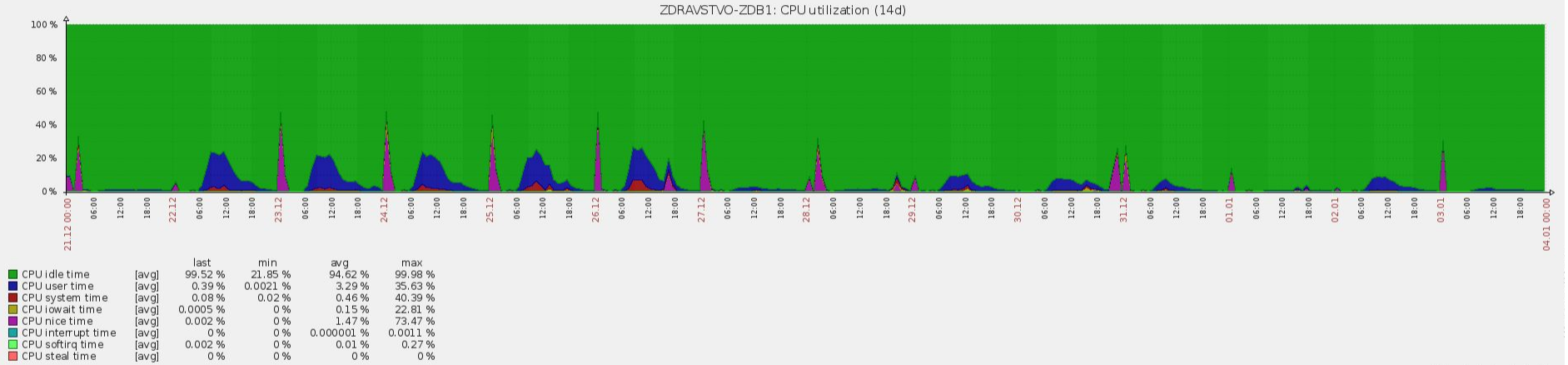
- Migration
  - ~ 1600 SQL Lines of Code
  - Rewrite of 30% (~35 at the time) of all views (mostly trivial rewrites)
  - Rewrite of key triggers for materialization of 2 key tables for free appointment slots lookup
  - Impossible to go back. No rollback script.
  - Scheduled for Sunday 28 Dec 2014. We needed a few days to see its behavior in production, before the start of Jan (start of month when the system has highest load)

Case study: The greatest optimization we've done so far

- Result
  - Less than 2 hours downtime. Started at 21:00, finished before 23:00
  - **Over 50% reduction of system load (~0.3 to ~0.15)**
  - Less locking and system time
  - No bugs from the migration, system continued with operation seamlessly

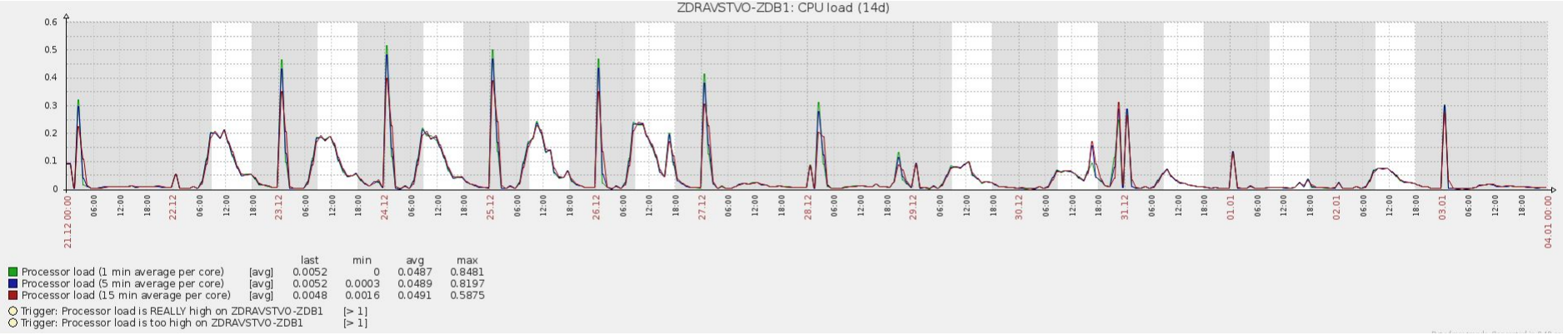
Case study: The greatest optimization we've done so far

- CPU Utilization



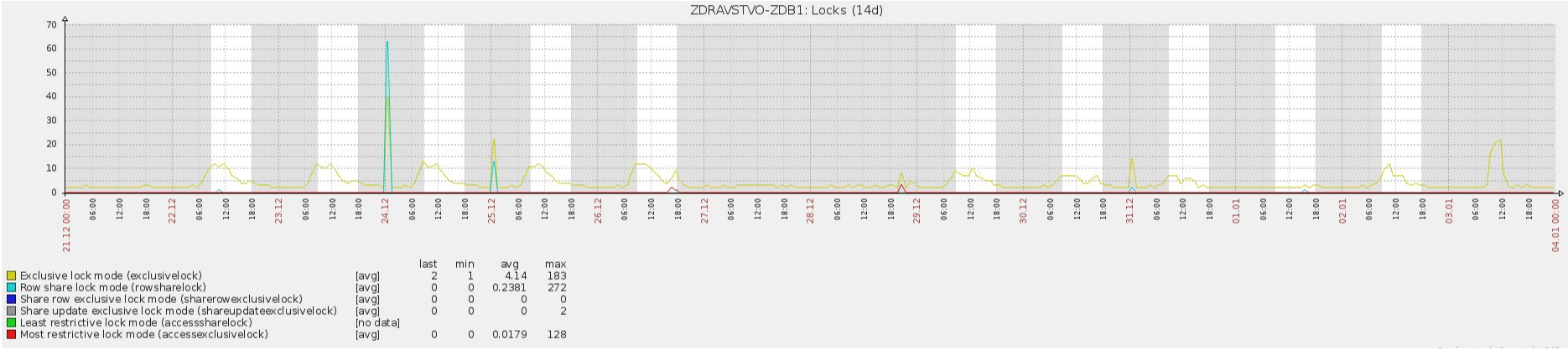
Case study: The greatest optimization we've done so far

- CPU Load



Case study: The greatest optimization we've done so far

- Locks



Agenda

- Overview
- Architecture
- Hardware
- Replication
- Query Optimization
- **Development**
- Monitoring
- Critical situations

Development

- Fast cycles of development and delivery
- Very rare for similar projects in our surroundings
- Uses migration and rollback scripts
- Test on staging environment
  - Test functionalities
  - Can’t test real live load
- Monitor after deployment
  - System time
  - Locks
  - IO wait time
  - Network
  - Overall load

Development: Versions

date	time	version
22-Oct-2015	10:28:20 AM	1.42.1
21-Oct-2015	4:38:35 PM	1.42.0
20-Oct-2015	2:28:40 PM	1.41.2
19-Oct-2015	4:09:22 PM	1.41.1
19-Oct-2015	1:39:38 PM	1.41.0
24-Sep-2015	2:51:08 PM	1.40.2
24-Sep-2015	2:22:00 PM	1.40.1
24-Sep-2015	12:46:48 PM	1.40.0
10-Sep-2015	12:32:19 PM	1.39.2
9-Sep-2015	5:48:24 PM	1.39.1
9-Sep-2015	3:55:15 PM	1.39.0
27-Aug-2015	3:31:32 PM	1.38.0
26-Aug-2015	5:37:07 PM	1.37.3
26-Aug-2015	12:05:40 PM	1.37.2
25-Aug-2015	4:43:53 PM	1.37.1
25-Aug-2015	3:28:47 PM	1.37.0
9-Jul-2015	12:29:58 PM	1.36.2

date	time	version
9-Jul-2015	10:58:01 AM	1.36.1
8-Jul-2015	3:54:52 PM	1.36.0
8-Jul-2015	11:38:18 AM	1.35.2
8-Jul-2015	11:30:56 AM	1.35.1
8-Jul-2015	11:01:28 AM	1.35.0
6-Jul-2015	1:30:50 PM	1.34.2
2-Jul-2015	12:17:46 PM	1.34.1
26-Jun-2015	3:56:42 PM	1.34.0
24-Jun-2015	11:19:05 AM	1.33.0
18-Jun-2015	3:58:13 PM	1.32.0
15-Jun-2015	3:39:54 PM	1.31.0
15-Jun-2015	2:55:55 PM	1.30.0
12-Jun-2015	4:49:33 PM	1.29.0
12-Jun-2015	9:44:21 AM	1.28.0
3-Jun-2015	3:09:53 PM	1.27.0
2-Jun-2015	4:23:05 PM	1.26.2
2-Jun-2015	3:49:32 PM	1.26.1

Development: Monitoring after deployment

- Monitor after deployment
  - System time
  - Locks
  - IO wait time
  - Network
  - Overall load

Agenda

- Overview
- Architecture
- Hardware
- Replication
- Query Optimization
- Development
- **Monitoring**
- Critical situations

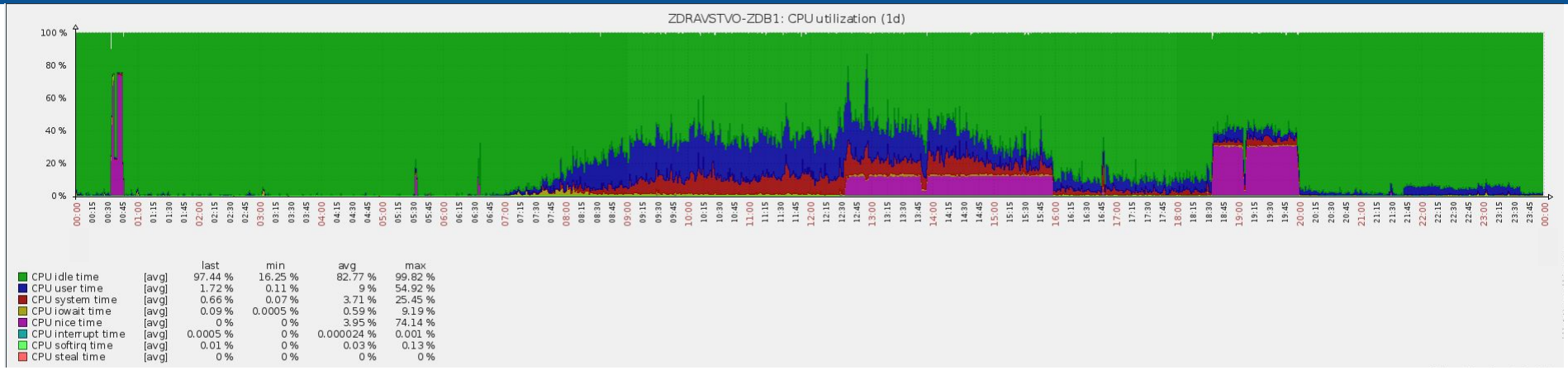
pgBadger

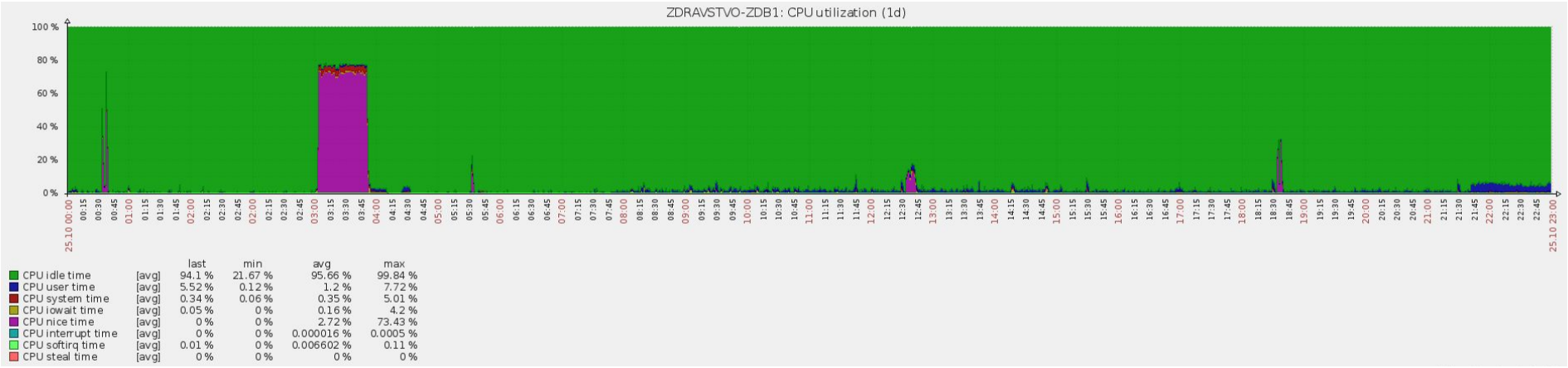
- we log everything
- helps optimization by dev team
- publishes results on internal web location
- runs four times a day
  - `nice -n 19`
  - `-j $NUMPROC` (load dependent, static hardcoded values)

Zabbix templates

- Based on zabbix wiki template by K0k
- Additional custom zabbix items for monitoring additional parameters: detailed db connections, import jobs, replication
- Triggering and notification to admins via email/SMS based on severity level

Zabbix: Example





Zabbix triggers

Important triggers for early detection of problems:

- Free disk space on volumes (that can shrink rapidly if there is a problem)
  - WAL archives
  - Logs
- CPU load
- Failed backups
- Network contention

Agenda

- Overview
- Architecture
- Hardware
- Replication
- Query Optimization
- Development
- Monitoring
- **Critical situations**

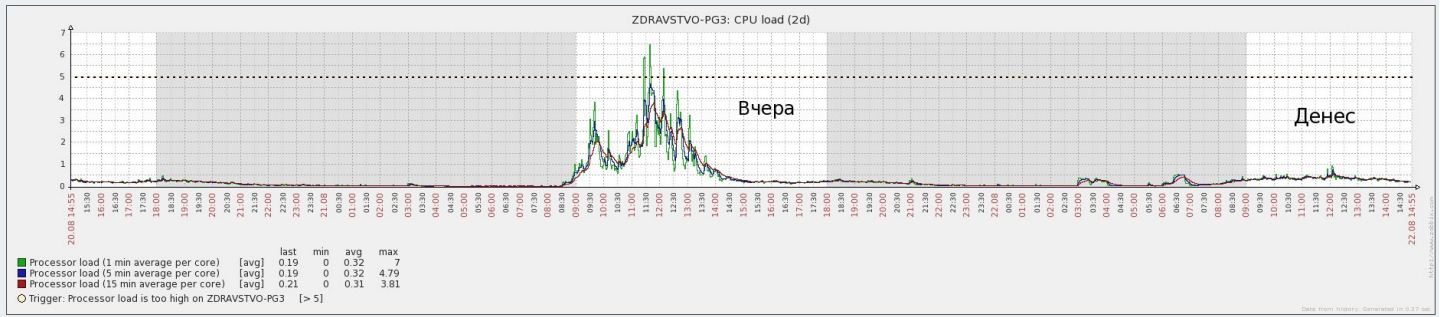
Critical situations

- Critical situations
  - Downtime
  - Data corruption
  - Hardware Failure
  - Failure - third party critical systems
  - Attacks

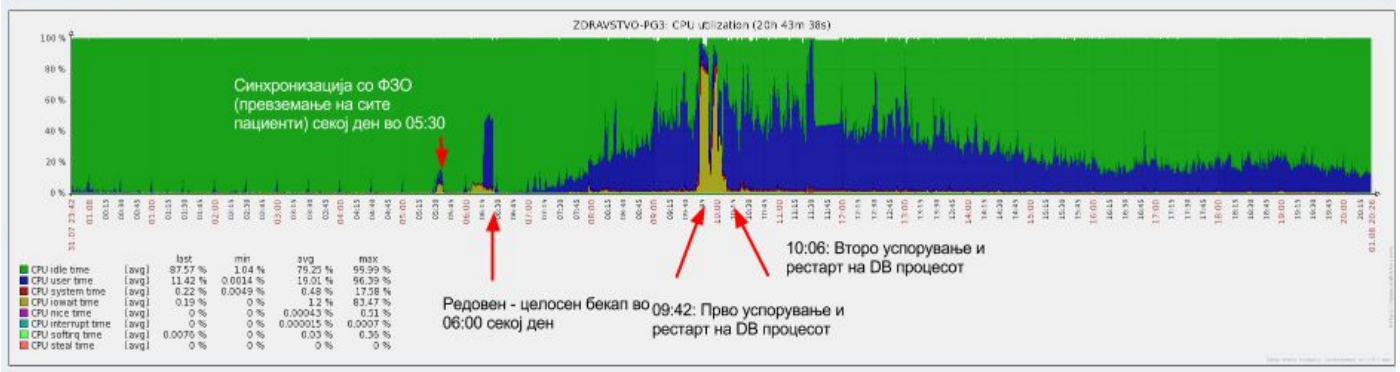
Critical situations

- 1st July 2013
  - Emergency Processor upgrade
- 26 Oct 2014 - DD day
  - **`dd if=/dev/zero of=/dev/sda`**
  - replication slave promoted to master
  - Master
    - recovery of the deleted conf files and scripts
    - configured and started as “slave”, then synced and promoted back to master

Critical situations: Some interesting graphs

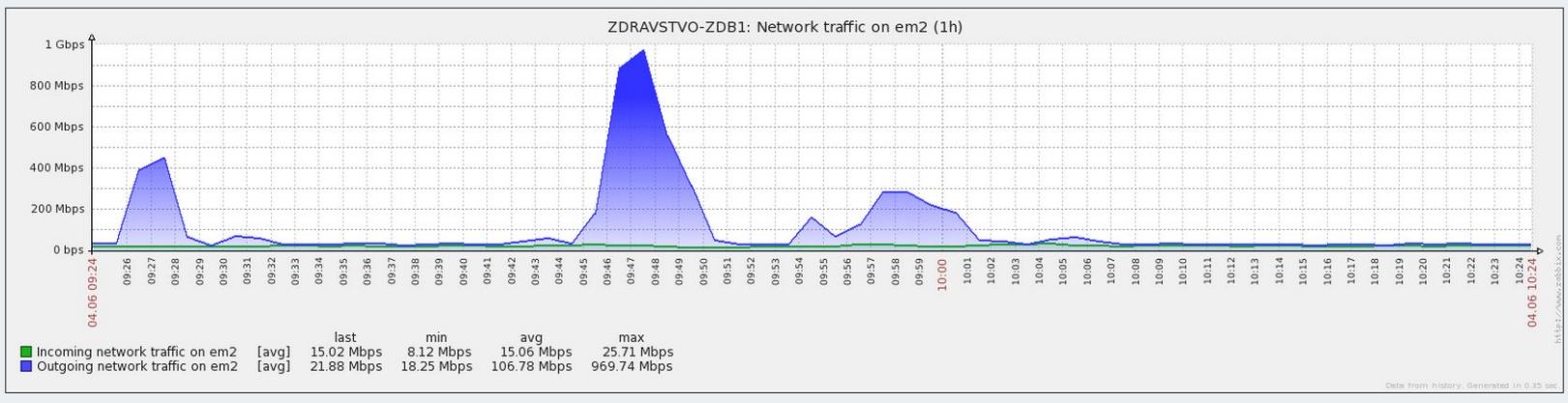


Critical situations: Some interesting graphs



Critical situations: Some interesting graphs

Critical situations: Some interesting graphs



Thank You