Greenplum observabitily

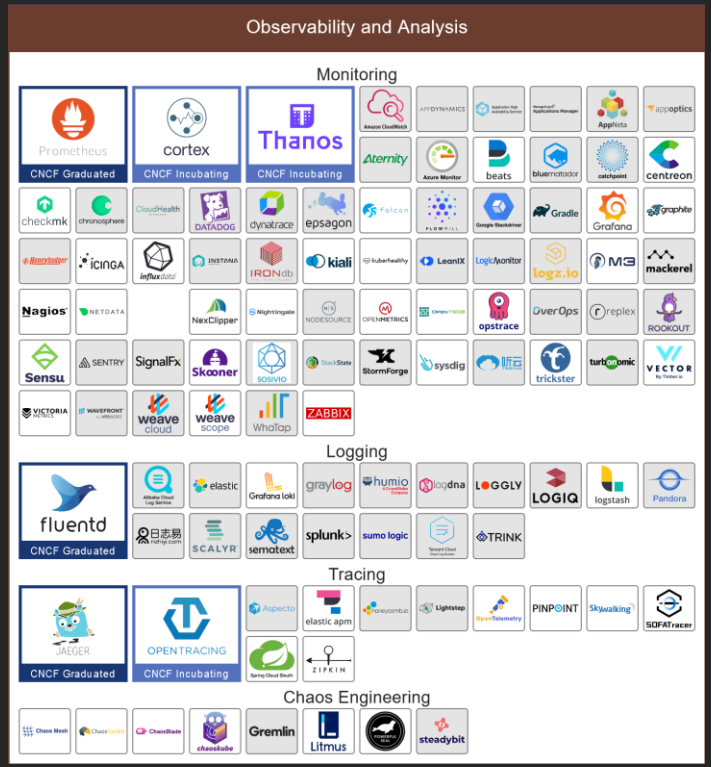
Как всесторонне контролировать работу GP без использования проприетарных инструментов

Дмитрий Ибрагимов
Data Platform Site Reliability Engineer
2021

Введение

Observability: Definition

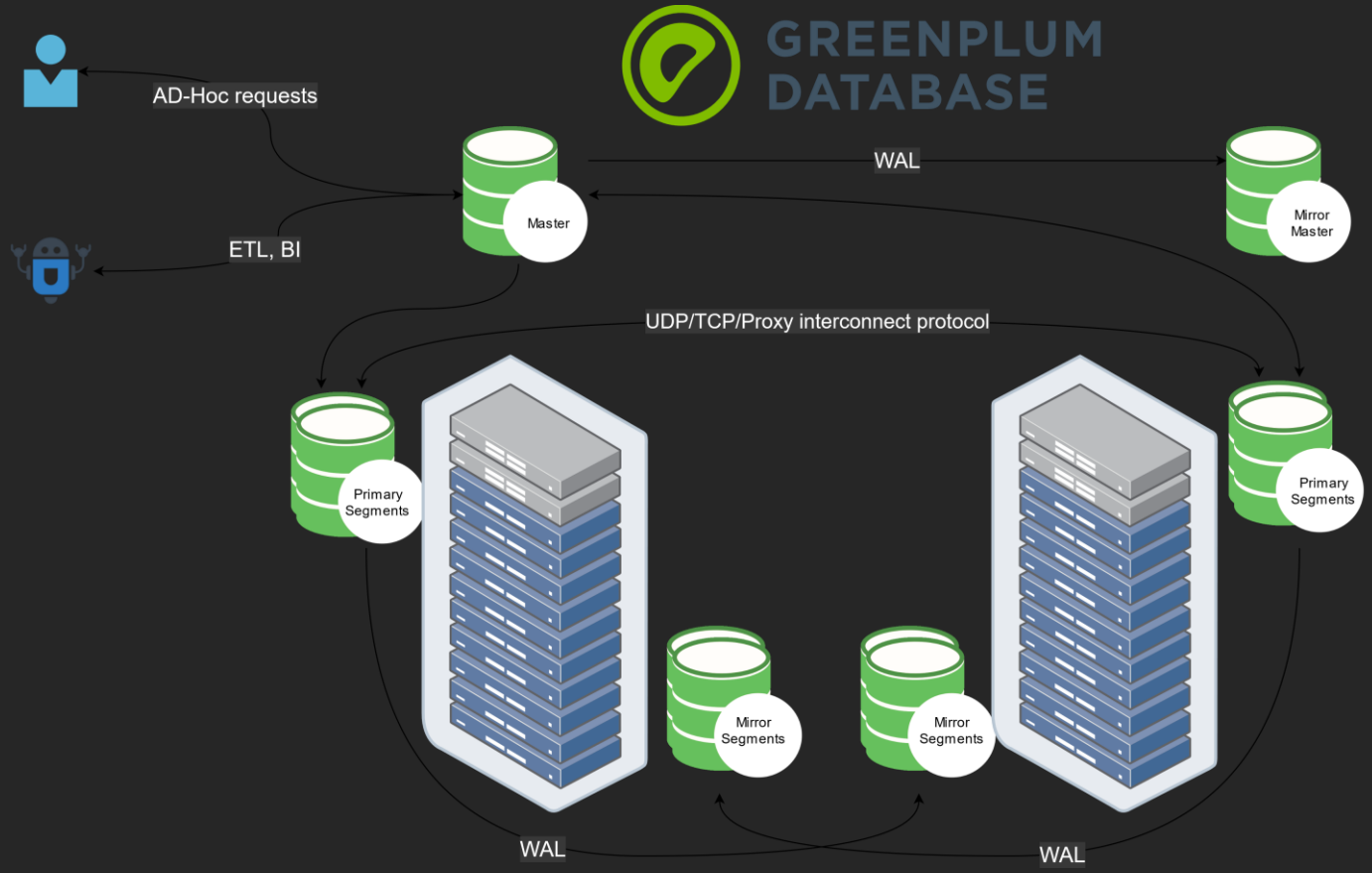
*«**Observability** is tooling or a technical solution that allows teams to actively debug their system. Observability is based on exploring properties and patterns not defined in advance.»* - Google



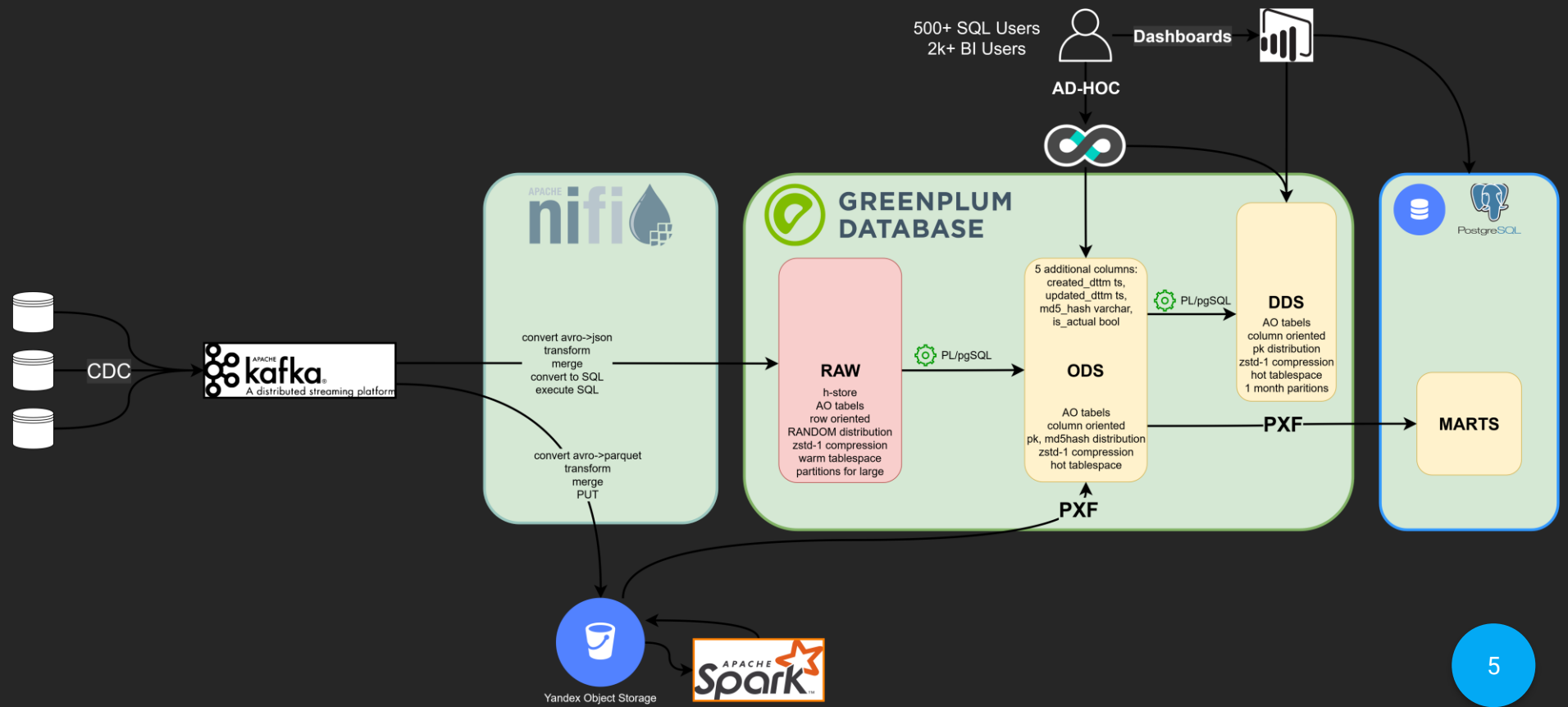
CNCF Landscape – observability tools

https://landscape.cncf.io

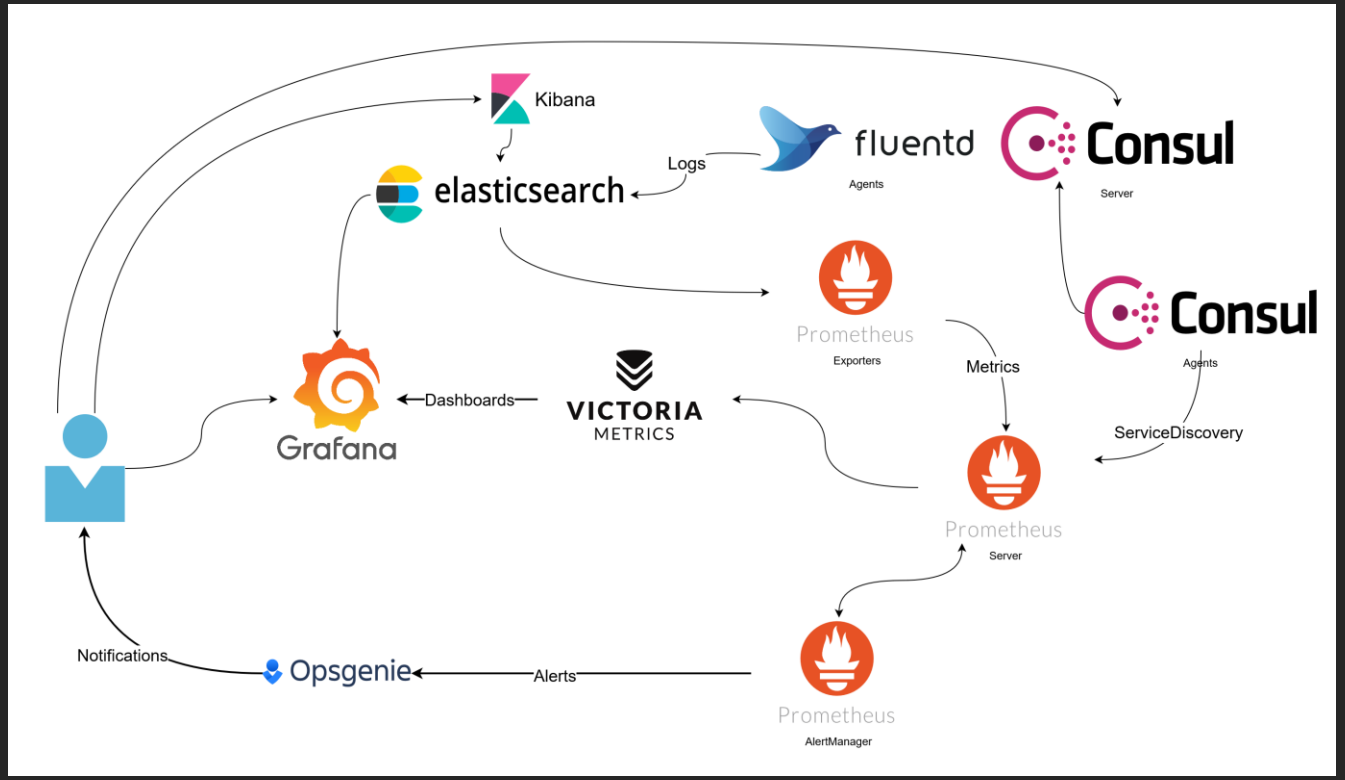
Архитектура



Архитектура



Архитектура мониторинга



Базовые метрики

Собираем с помощью стандартного **node-exporter**:
https://github.com/prometheus/node_exporter

В графане дашборд **Node Exporter Full**:
https://grafana.com/grafana/dashboards/1860

По интересующим нас метрикам нужно собрать свой **агрегированный дашборд**

- CPU
- Disks
- Memory
- IO (read/write)

Базовые метрики

- CPU

```
sort_desc(1 - (avg_over_time (avg by (instance)
(irate(node_cpu_seconds_total{instance=~"p-dtpl-[sm]dw.*",mode="idle"}[1m]))
[$period:1m])))
```

Базовые метрики

- CPU Alerts

CPU Unbalanced, CPU Waits:

- https://github.com/diarworld/greenplum-exporter-queries/blob/main/base-alerts.yml

Базовые метрики

- Disks

Read/write:

```
sort_desc(sum by (instance)
(avg_over_time(irate(node_disk_read_bytes_total{cluster=~"$cluster", instance
=~".*sdw.*"} [1m]) [$period:1m])))

sort_desc(sum by (instance)
(avg_over_time(irate(node_disk_written_bytes_total{cluster=~"$cluster", instance
=~".*sdw.*"} [1m]) [$period:1m])))
```

% Free space:

```
100 - ((sum(node_filesystem_avail_bytes{cluster=~"$cluster", instance
=~".*sdw.*",mountpoint=~"/data[0-9]+",fstype!="rootfs"}) * 100) /
sum(node_filesystem_size_bytes{cluster=~"$cluster", instance
=~".*sdw.*",mountpoint=~"/data[0-9]+",fstype!="rootfs"}))
```

Базовые метрики

- Disks space alerts

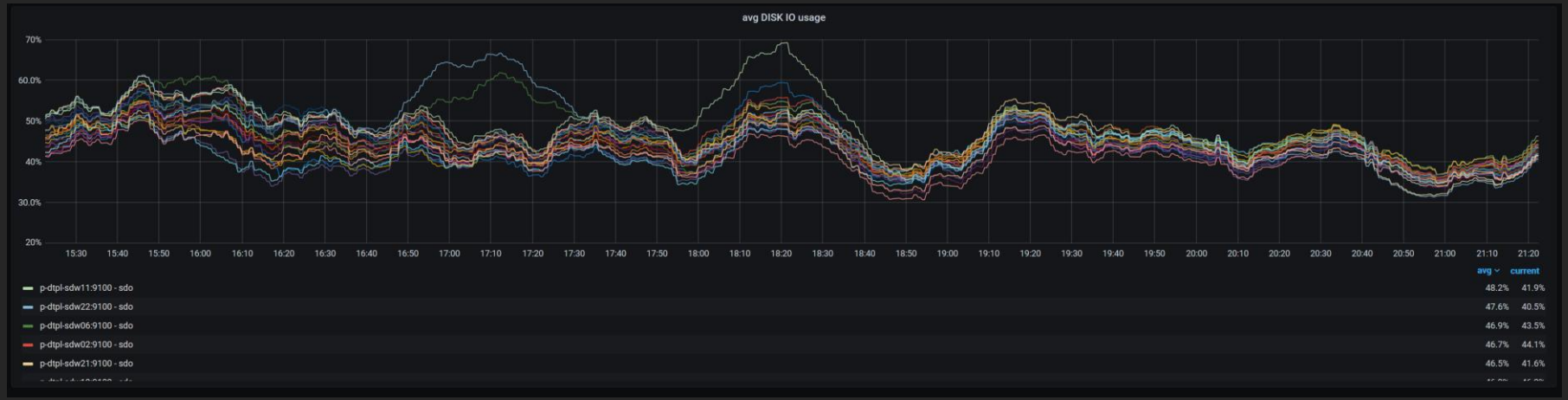
Алертить нужно при >70% использованного места в data разделах. При 10% есть риск остановки кластера, из-за генерации спиллов:

- https://github.com/diarworld/greenplum-exporter-queries/blob/main/base-alerts.yml

Базовые метрики

- IO usage

```
sort_desc(avg_over_time(irate(node_disk_io_time_seconds_total{instance=~"p-dtpl-sdw.*",job="consul", device=~"$device"} [1m]) [$period:1m]))
```



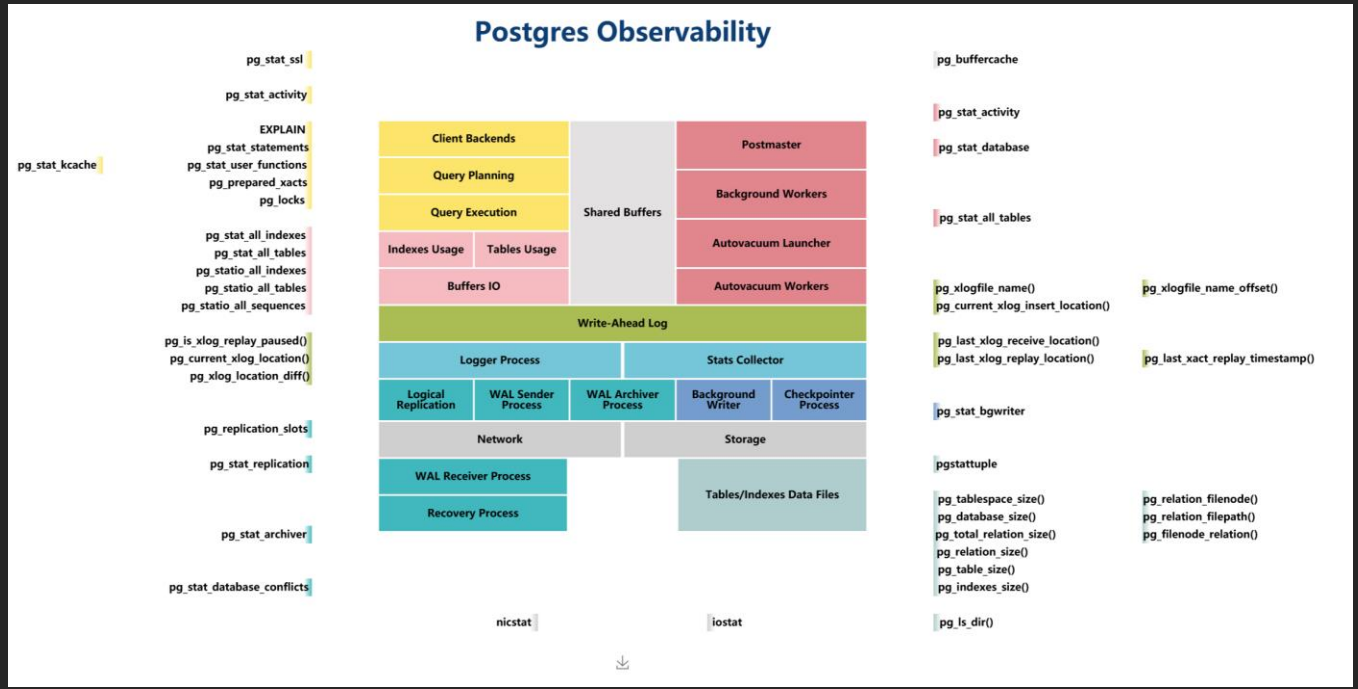
Базовые метрики

- IO usage alerts

Алертить нужно при >90% IO usage. Также можно алертить при расхождении IO usage >10% между сегмент нодами по аналогии с CPU Unbalanced:

- https://github.com/diarworld/greenplum-exporter-queries/blob/main/base-alerts.yml

Метрики GP-inside



https://pgstats.dev/

Метрики GP-inside

Для сбора внутренних GP-метрик используем:

- https://github.com/free/sql_exporter
- https://github.com/prometheus-community/postgres_exporter

Метрики GP-inside

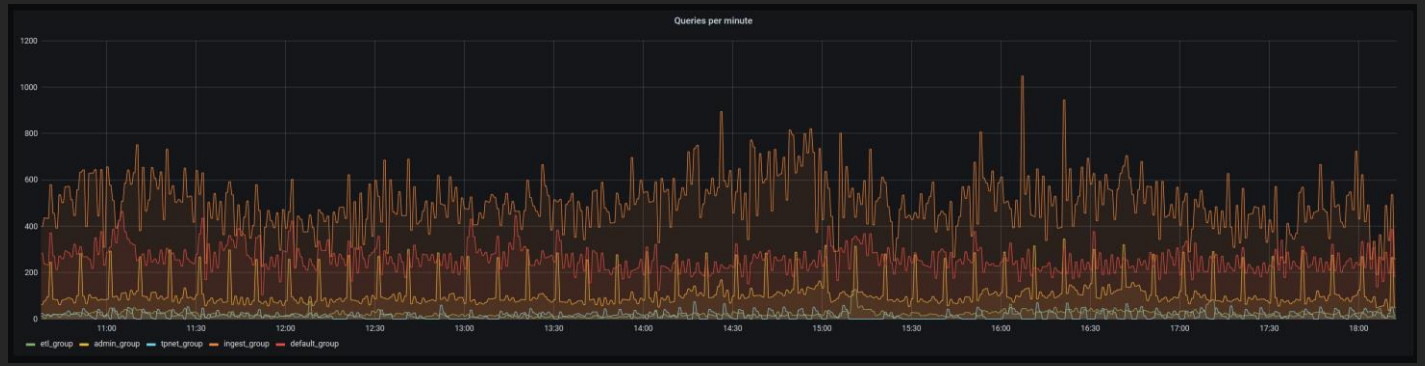
Метрики с мастера:

- gp_uptime
- gp_connection
- gp_total_segments
- gp_total_primaries
- gp_mirror_as_primaries
- gp_segment_up
- gp_resgroup_status
- gp_max_tx
- gp_locks
- gp_spill_file
- gp_resgroup_status_per_host
- gp_resgroup_config
- gp_queries_history <- gpperfmon only
- gp_workfile_per_user
- gp_max_connections
- gp_superuser_connections
- gp_wait_ses
- gp_checkpoint_count
- gp_total_table
- gp_total_schema
- gp_bloat
- gp_stats_missing

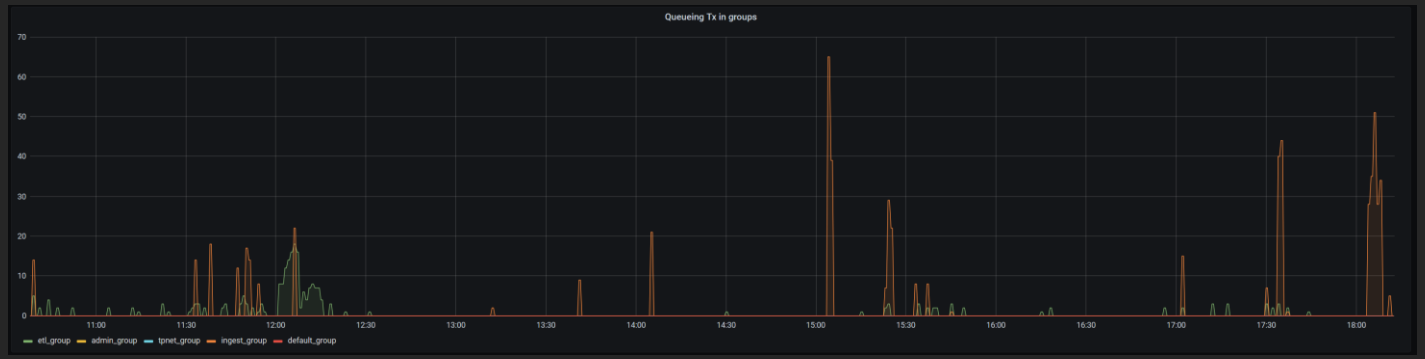
https://github.com/diarworld/greenplum-exporter-queries/blob/main/queries.yml

Метрики GP-inside

Графики:




```
increase(gp_resgroup_status_num_executed{cluster="$cluster"}[1m:1m])
```




```
gp_resgroup_status_num_queueing{cluster="$cluster"}
```

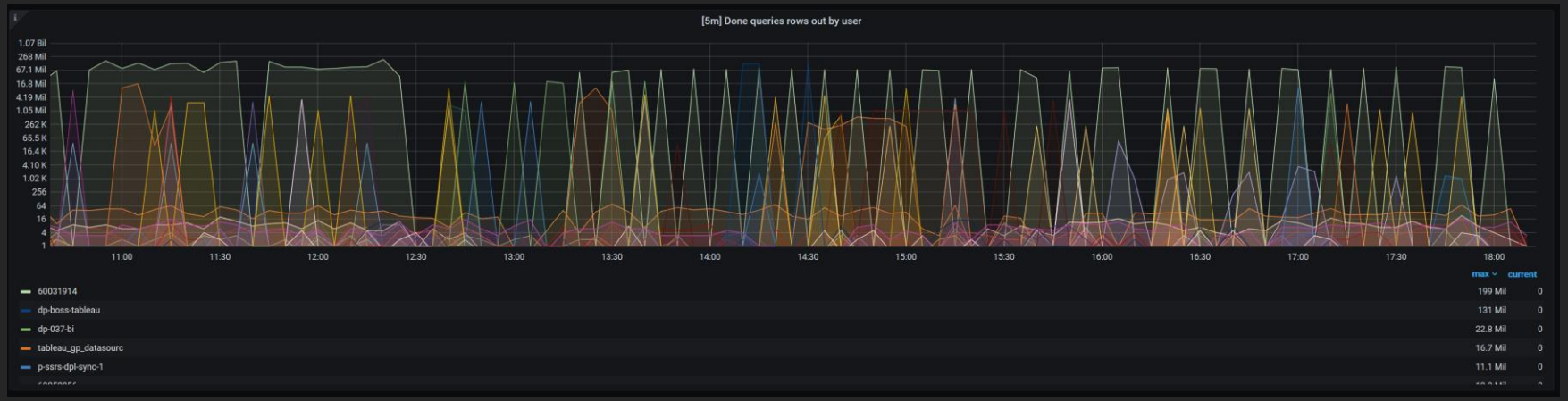
Метрики GP-inside

Графики:


```
sort(gp_connection_sum{cluster="$cluster"}) as {{username}}
sum(gp_connection_sum{cluster="$cluster"}) as Total
gp_max_connections_limit as Limit
```

Метрики GP-inside

Графики (gpperfmon only!):




```
sum(gp_queries_history_rows_out{cluster="$cluster"}) by (username)
```

Метрики GP-inside

Алерты:

- Упавшие сегменты
- Очереди в ресурсных группах
- Падение числа успешных запросов до нуля
- Большое число коннектов
- Спиллы/workfiles (если не ограничены)
- Лаг репликации сегментов

https://github.com/diarworld/greenplum-exporter-queries/blob/main/alerts.yml

Метрики Process exporter

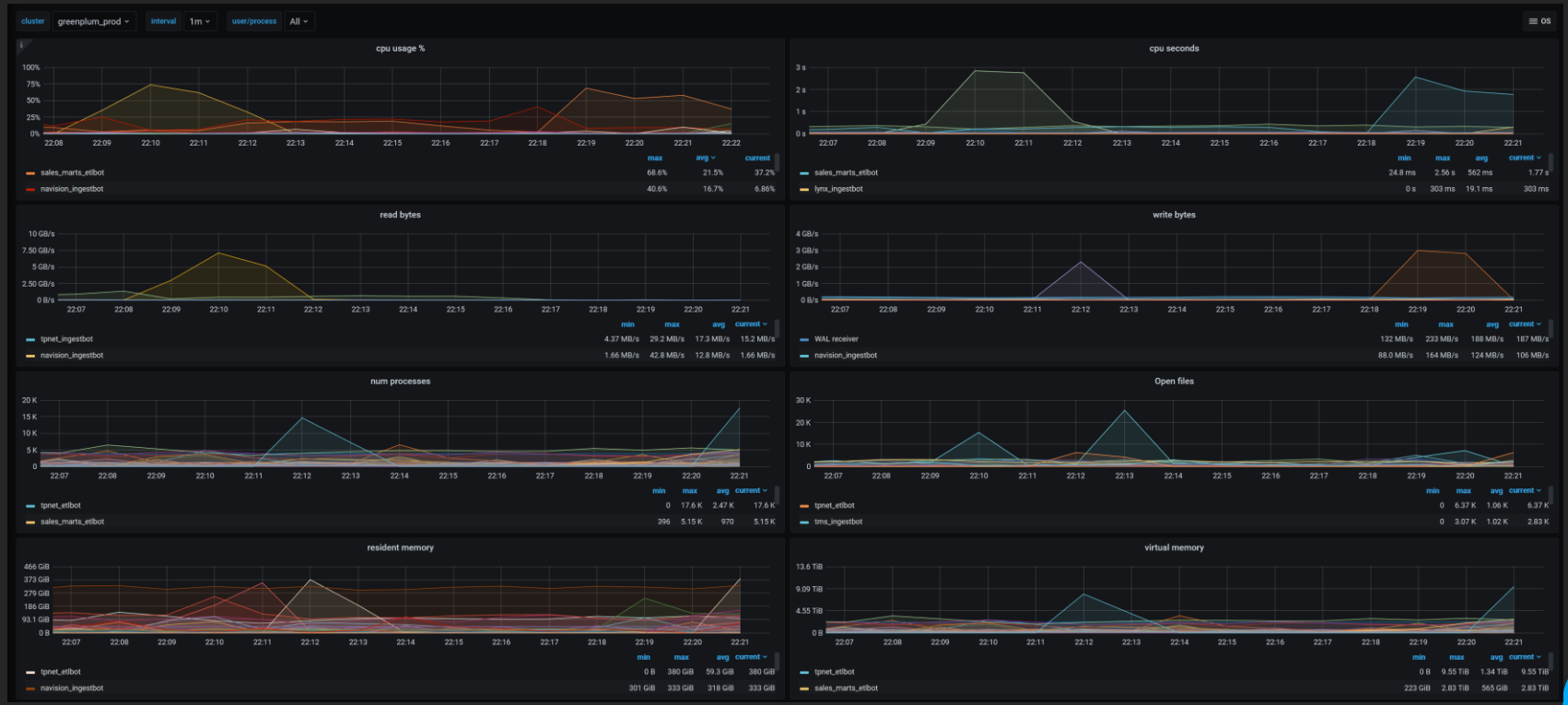
Используем:
https://github.com/ncabatoff/process-exporter

Конфиг экспортера:
https://github.com/diarworld/greenplum-exporter-queries/blob/main/process-exporter.yml

Дашборд:
https://grafana.com/grafana/dashboards/249

Метрики Process exporter

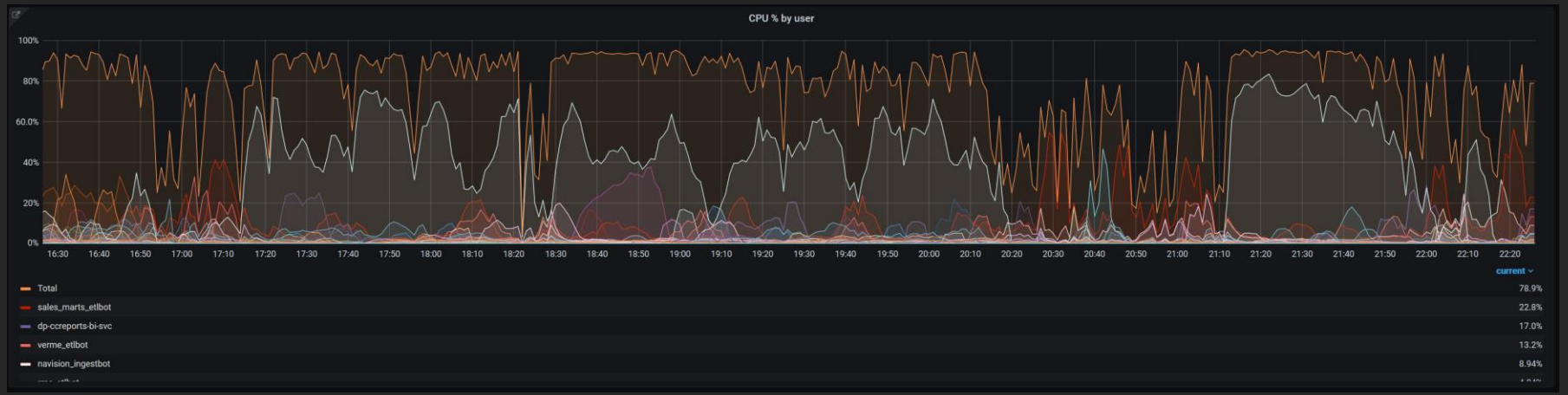
Дашборд



Метрики Process exporter

- Считаем % использования CPU в разрезе пользователей

```
(1 - avg(irate(node_cpu_seconds_total{mode='idle',cluster="$cluster"}[5m]))) * (sum
by (username)
(label_replace(rate(namedprocess_namegroup_cpu_seconds_total{cluster="$cluster",
groupname!~"WAL.*"}[5m]), "username", "$1", "groupname", "(.*):.*")) /
sum(rate(namedprocess_namegroup_cpu_seconds_total{cluster="$cluster"}[5m])))
```



Метрики Process exporter

- CPU Burst позволяет запросам использовать ресурсы из соседних ресурсных групп:

«A resource group may utilize more CPU than its CPU_RATE_LIMIT when other resource groups are idle. In this situation, Greenplum Database allocates the CPU resource of an idle resource group to a busier one. This resource group feature is called CPU burst.» - GPDB Docs

В GP 6.16 появился параметр `gp_resource_group_cpu_ceiling_enforcement`, который можно выставить в `true`, чтобы отключить CPU burst.

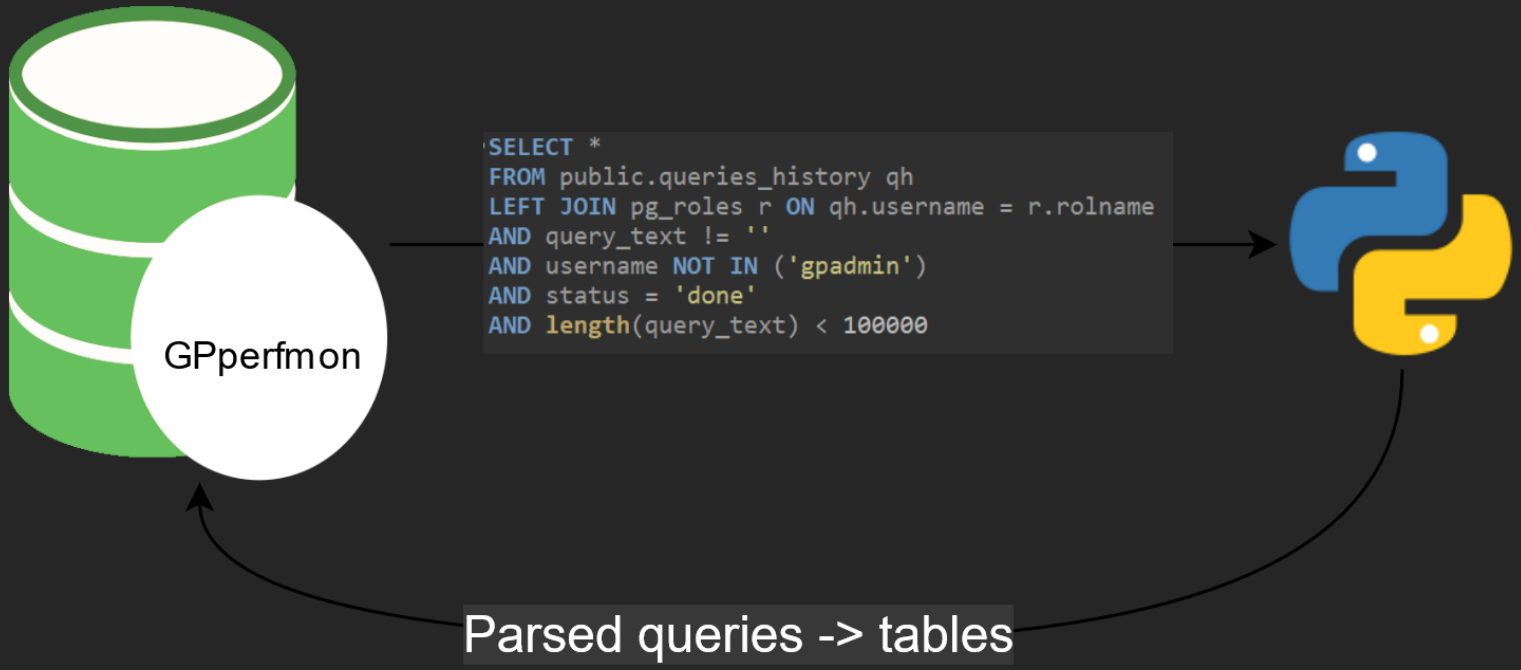
Метрики Process exporter

- Алерты

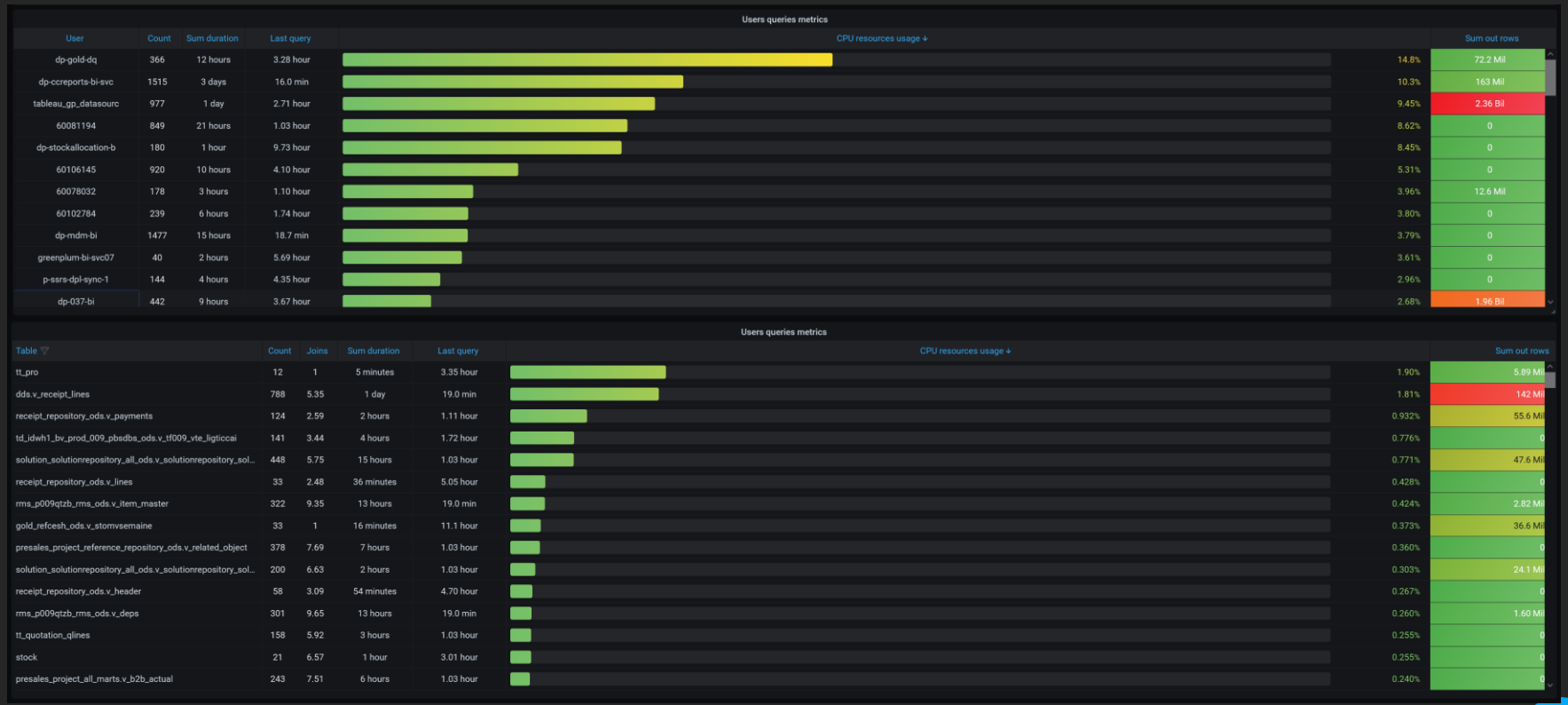
Парсинг gpperfmon

Необходимо иметь представление о популярности объектов – частоте запросов, числе JOIN’ов, средней длительности запросов.

https://github.com/macbre/sql-metadata



Парсинг gpperfmon



Users queries metrics

User	Count	Sum duration	Last query	CPU resources usage ↓		Sum out rows
dp-gold-dq	366	12 hours	3.28 hour	14.8%		72.2 Mil
dp-ccreports-bi-svc	1515	3 days	16.0 min	10.3%		163 Mil
tableau_gp_datasourc	977	1 day	2.71 hour	9.45%		2.36 Bil
60081194	849	21 hours	1.03 hour	8.62%		0
dp-stockallocation-b	180	1 hour	9.73 hour	8.45%		0
60106145	920	10 hours	4.10 hour	5.31%		0
60078032	178	3 hours	1.10 hour	3.96%		12.6 Mil
60102784	239	6 hours	1.74 hour	3.80%		0
dp-mdm-bi	1477	15 hours	18.7 min	3.79%		0
greenplum-bi-svc07	40	2 hours	5.69 hour	3.61%		0
p-ssrs-dpl-sync-1	144	4 hours	4.35 hour	2.96%		0
dp-037-bi	442	9 hours	3.67 hour	2.68%		1.96 Bil

Users queries metrics

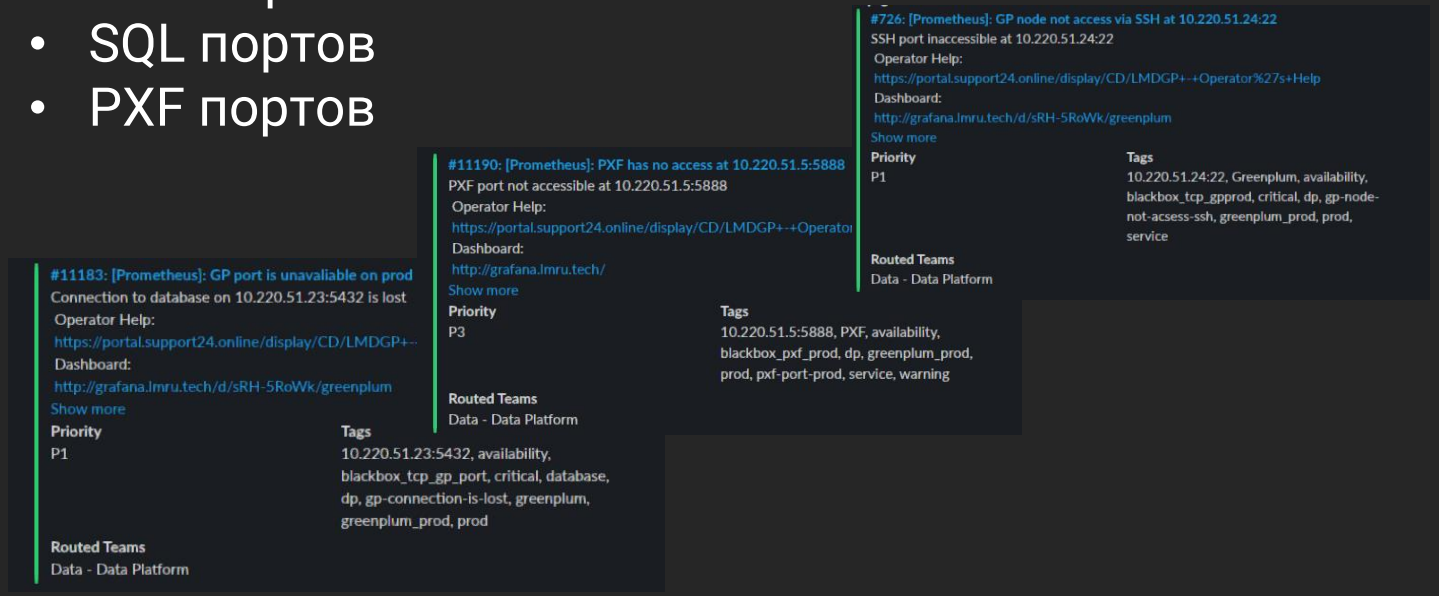
Table	Count	Joins	Sum duration	Last query	CPU resources usage ↓		Sum out rows
tt_pro	12	1	5 minutes	3.35 hour	1.90%		5.89 Mil
dds.v_receipt_lines	788	5.35	1 day	19.0 min	1.81%		142 Mil
receipt_repository_ods.v_payments	124	2.59	2 hours	1.11 hour	0.932%		55.6 Mil
td_idwh1_bv_prod_009_pbsdbs_ods.v_tf009_vte_ligticcai	141	3.44	4 hours	1.72 hour	0.776%		0
solution_solutionrepository_all_ods.v_solutionrepository_sol...	448	5.75	15 hours	1.03 hour	0.771%		47.6 Mil
receipt_repository_ods.v_lines	33	2.48	36 minutes	5.05 hour	0.428%		0
rms_p009qtzb_rms_ods.v_item_master	322	9.35	13 hours	19.0 min	0.424%		2.82 Mil
gold_refcesh_ods.v_stomvsemaine	33	1	16 minutes	11.1 hour	0.373%		36.6 Mil
presales_project_reference_repository_ods.v_related_object	378	7.69	7 hours	1.03 hour	0.360%		0
solution_solutionrepository_all_ods.v_solutionrepository_sol...	200	6.63	2 hours	1.03 hour	0.303%		24.1 Mil
receipt_repository_ods.v_header	58	3.09	54 minutes	4.70 hour	0.267%		0
rms_p009qtzb_rms_ods.v_deps	301	9.65	13 hours	19.0 min	0.260%		1.60 Mil
tt_quotation_qlines	158	5.92	3 hours	1.03 hour	0.255%		0
stock	21	6.57	1 hour	3.01 hour	0.255%		0
presales_project_all_marts.v_b2b_actual	243	7.51	6 hours	1.03 hour	0.240%		0

Blackbox exporter

https://github.com/prometheus/blackbox_exporter

Мониторим и алертим по недоступности:

- SSH портов
- SQL портов
- PXF портов

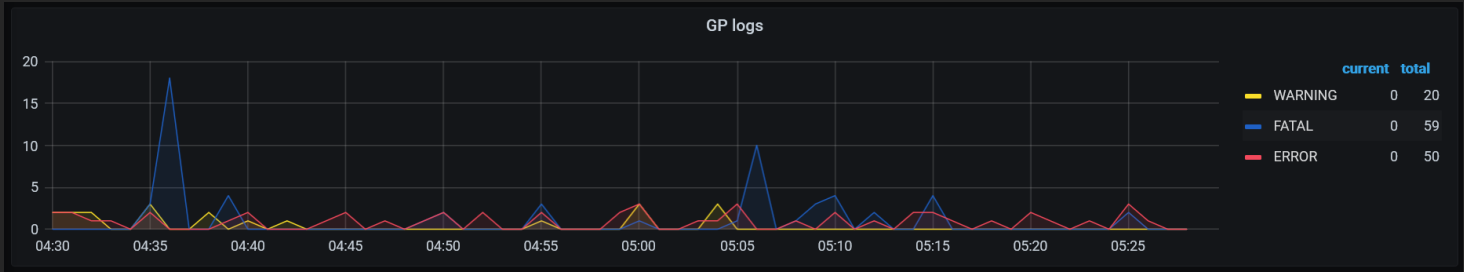


Elasticsearch query exporter

https://github.com/braedon/prometheus-es-exporter

Мониторим и алертим по:

- PANIC в логах
- Резко возросшему числу FATAL/ERROR в логах



GP logs

Severity	User	Message	Count
FATAL	team_lg_tech_user	connection to client lost	13
ERROR	gpadmin	ValueError: Extra data: line 5 column 1 - line 505 column 1 (char 50 - 7136) (plpy_elog.c:121)	12
WARNING	drbot	skipping "tmp_dump" --- cannot analyze non-tables or special system tables	9
ERROR	wms_etlbot	column "numrec" does not exist	9

Bash-изм

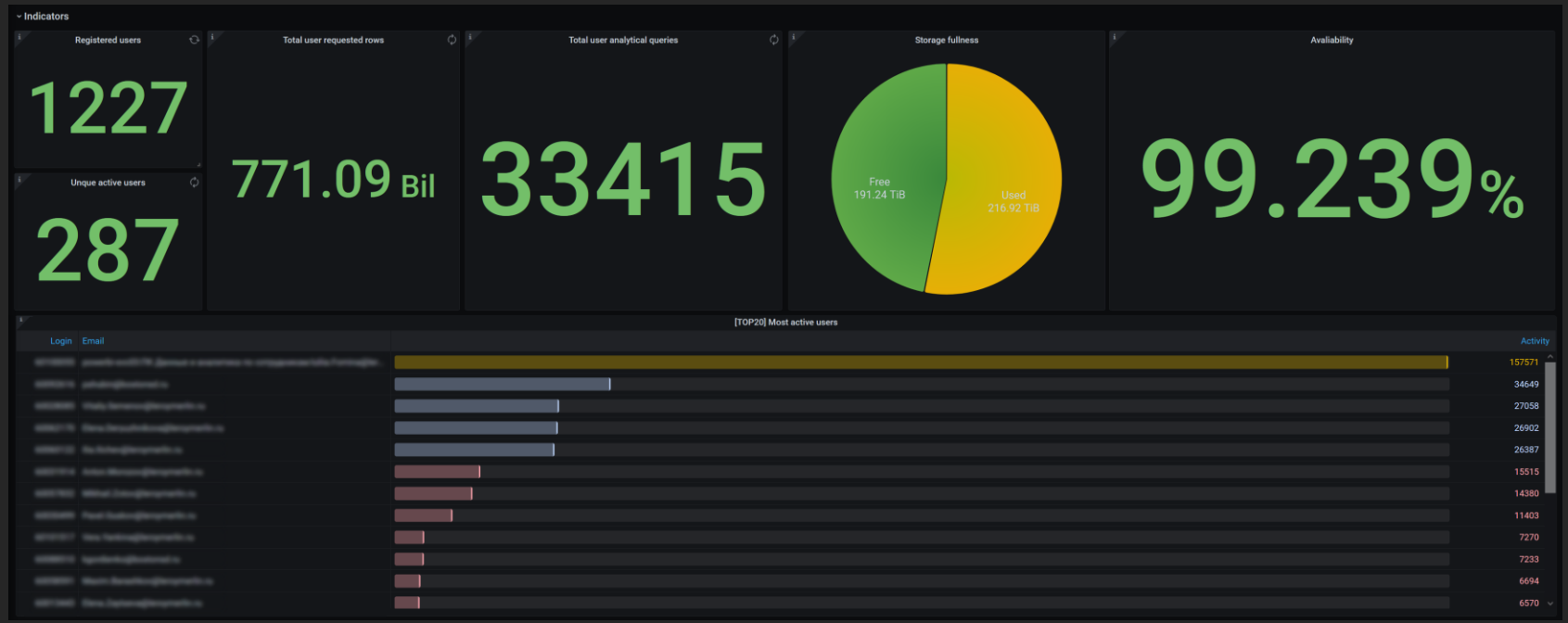
Отстреливаем запросы через pg_terminate_backend:

- Запросы, которые бегут (active) на кластере дольше 1 часа
- Дублированные запросы
- Idle in transaction > 10 минут
- Idle коннекты > 2 часов

https://github.com/diarworld/greenplum-exporter-queries/blob/main/gp-bashism.sh

SLA/SLI/SLO

Создаем SLA/SLI/SLO дашборд



SLA/SLI/SLO

Считаем SLI:
Количество минут, когда за минуту в кластере не пробежало ни одного запроса (кроме admin_group):

```
(sum(increase(gp_resgroup_status_num_executed
{cluster="greenplum_$cluster", rsgname!="admin_group"}[1m]))
OR on() vector(0)) < 1
OR vector(1)
```

Error budget:
SLA 99.5% = Monthly 3h 39m 8s

=> Если мы за месяц исчерпали наш бюджет ошибок – обслуживания и обновления переносим на следующий. Если не исчерпали, проводим доп. обслуживание.

SLA/SLI/SLO

Какие SLO можно придумать для GP:

- Среднее время подключения к базе < 5 секунд
- Среднее время ожидания начала выполнения запроса в ресурсной группе < 1 секунды
- Время сбора GP-inside метрик (scrape duration) < 10 секунд
- Время работы пайплайнов (расчета витрин, ODS/DDS слоев) колеблется в пределах 2-5%
- Количество таблиц без статистики растет не более 100 в день (gp_toolkit.gp_stats_missing)
- Генерация спиллов происходит не более чем для 0.5% запросов
- Для >80% популярных/тяжелых запросов созданы витрины и DDS
- Средний размер output’a AD-HOC запроса < 100.000 строк
- И многое другое

Планы на будущее

Что еще можно посмотреть:

- auto_explain модуль + сбор и парсинг результатов
- Метрики **pgbouncer** (show pools/stats)
- **Envoy** прокси для postgres: https://www.envoyproxy.io/docs/envoy/latest/configuration/listeners/network_filters/postgres_proxy_filter.html
- Анализ метрик с сегментов (pg_statio_user_tables, pg_stat_user_tables) – seq scans и heap blocks read

Подводим итоги

Инсайты и выводы

С помощью хорошо настроенных инструментов мониторинга можно:

- Быстро находить кривые запросы и юзеров которые их написали

Инсайты и выводы

С помощью хорошо настроенных инструментов мониторинга можно:

- Быстро находить кривые запросы и юзеров которые их написали
- Находить перекосы в раскладке данных

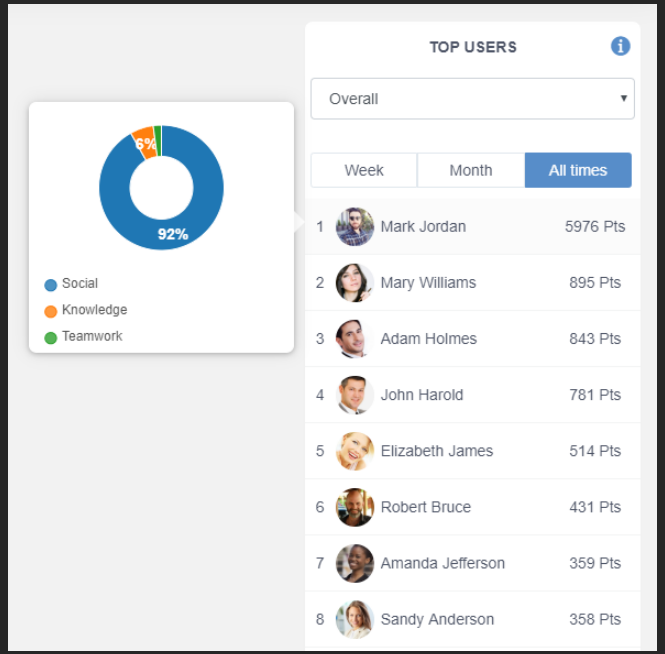
Инсайты и выводы

С помощью хорошо настроенных инструментов мониторинга можно:

- Быстро находить кривые запросы и юзеров которые их написали
- Находить перекосы в раскладке данных
- Находить наиболее популярные данные и строить по ним витрины

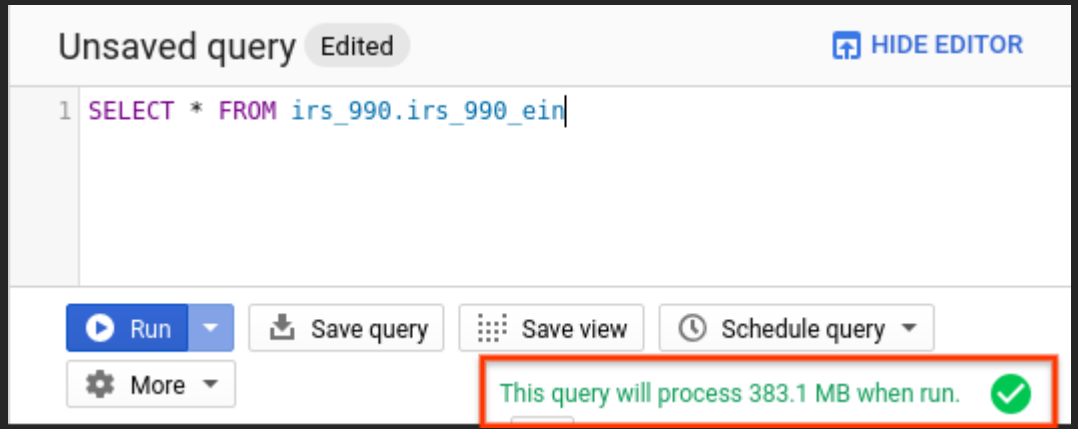
Инсайты и выводы

- Создать инструмент геймификации (медальки, gold status, etc)



Инсайты и выводы

- Создать свой аналог BigQuery – расчитывать стоимость запросов на основе сгенерированной нагрузки на кластер



Спасибо за внимание!

Дмитрий Ибрагимов
Data Platform Site Reliability Engineer
Dmitry.Ibragimov@leroymerlin.ru
Github/telegram: @diarworld

IN
DATA
WE
TRUST