

Петабайты в PostgreSQL доступные для многих

Михаил Жилин



Михаил Жилин

- Физтех
- 15 лет в нагрузочном тестировании и Java разработке
- Занимаюсь производительностью PostgreSQL и не только
- Контрибьютор в Open Source проекты



О компании Postgres Professional

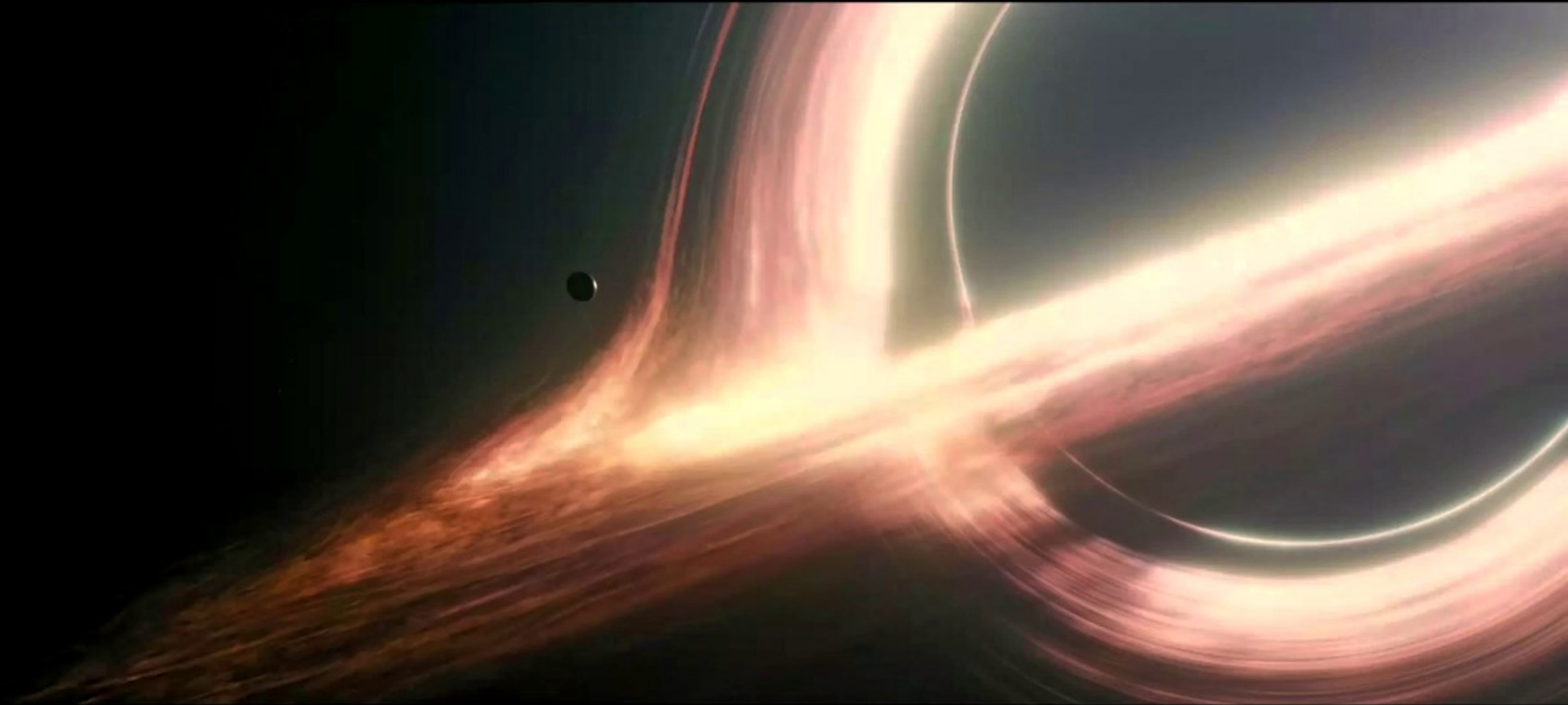
- Ведущий разработчик базы данных PostgreSQL
- Делает коммерческие продукты
 - Быстродействие
 - Новый функционал
 - Масштабируемость
- Образование, обучение, конференции...
- aka PgPro

О чём сегодня будет разговор?

- Небольшая история под пиво
- Про Postgres, OS, железки
- Про петабайт или почти триллион строк в базе

А с чего началась история?

- На календаре 10 декабря...
- Петабайт
- Распределенный PostgreSQL (PgPro Shardman)
- Commodity железо – чем дешевле, тем лучше
- Когда: **до 20 января**



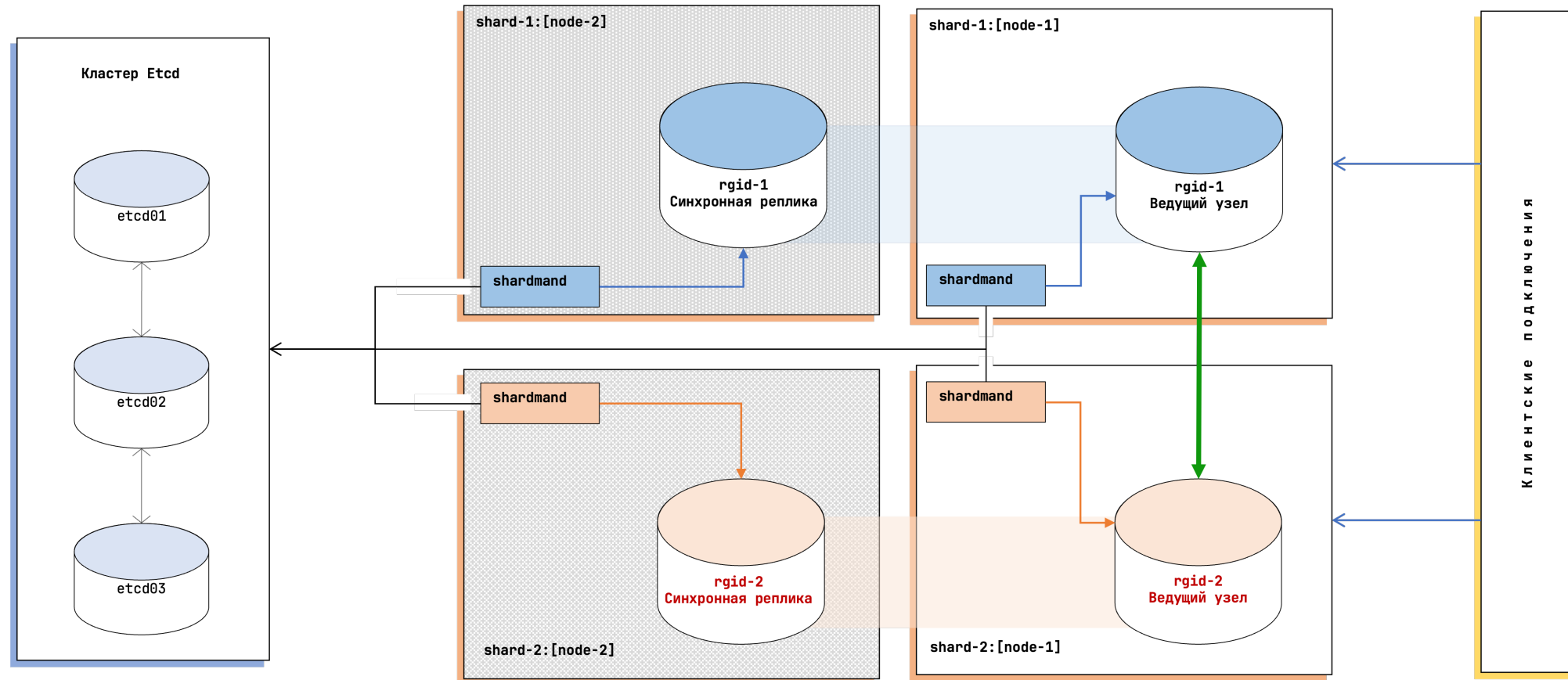
Who is Mr. Shardman?



Who is Mr. Shardman?

- Распределенная база данных поверх PostgreSQL
 - Шардирование через партиционирование
 - Share-nothing
 - Коммерческая, закрытый код
- Честный ACID
- Экосистемные утилиты (pg_probackup, расширения и т.п.)
- <https://postgrespro.ru/docs/shardman/14/index>

Who is Mr. Shardman?



Who is Mr. Shardman?

- Отказоустойчивость
 - Дополнительное место = доп стоимость
- Нагрузки
 - Любые бенчмарки для OLTP
- Цель тестов
 - Залить 1 петабайт и нагрузить запросами

Yahoo! Cloud Serving Benchmark



Yahoo! Cloud Serving Benchmark

- NoSQL бенчмарк 2010 год
- Простые операции
- Никаких join-ов, агрегаций, сложных запросов
- Никаких distributed транзакций
- Идеально для простейшего OLTP теста

Yahoo! Cloud Serving Benchmark

Partitioned table "public.usertable"

Column	Type	Collation	Nullable	Default
ycsb_key	character varying(64)		not null	
field0	character varying(100)			
field1	character varying(100)			
field2	character varying(100)			
field3	character varying(100)			
field4	character varying(100)			
field5	character varying(100)			
field6	character varying(100)			
field7	character varying(100)			
field8	character varying(100)			
field9	character varying(100)			

Partition key: HASH (ycsb_key)

Indexes:

"usertable_pkey" PRIMARY KEY, btree (ycsb_key)

Yahoo! Cloud Serving Benchmark

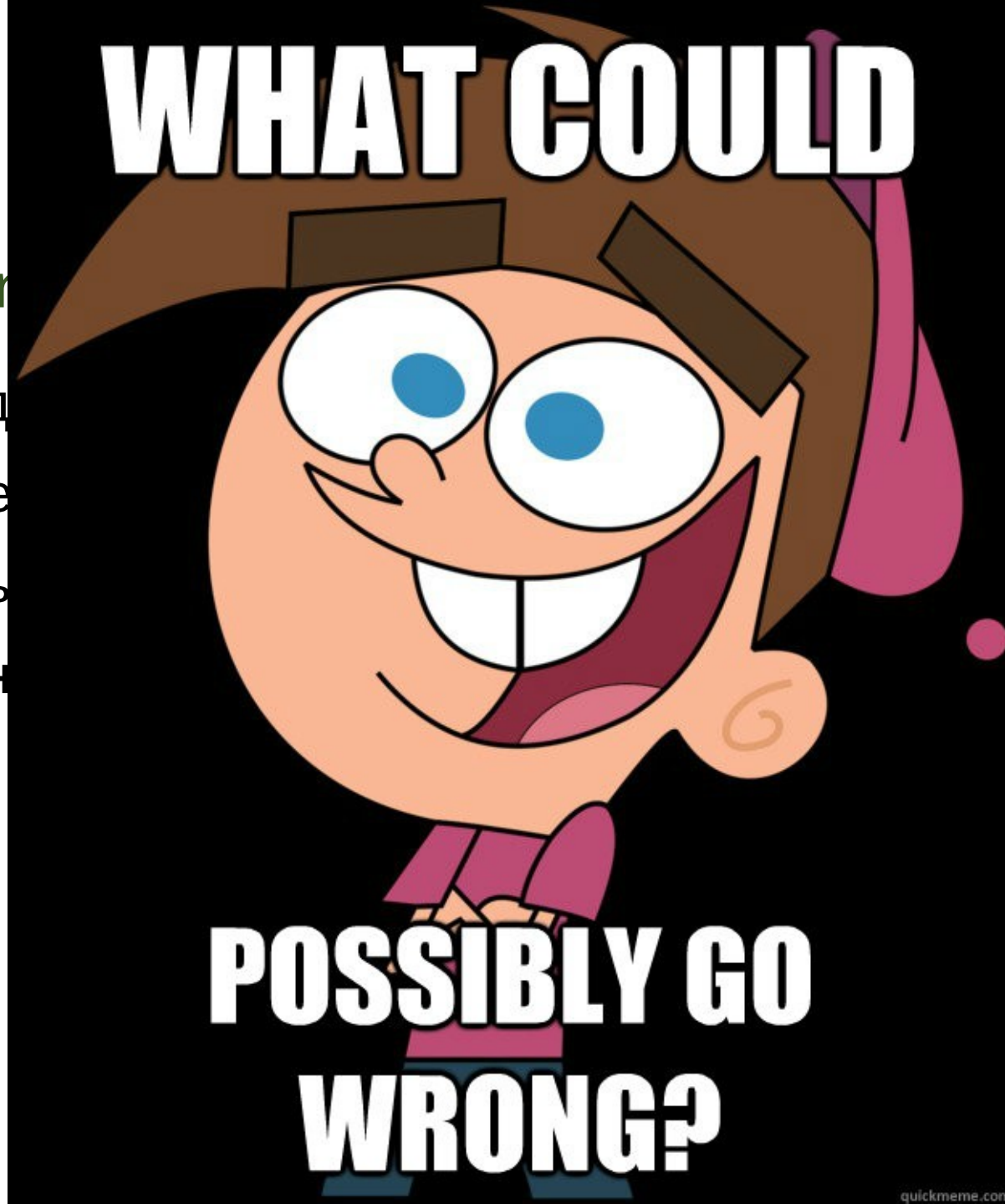
- `ycsb_key = 'user' + hash(seq)`
- Остальные колонки – случайный байты
- Одна строка – 1100 байт
- Почти триллион строк
- 1024 терабайта за 2 недели ~ 70 TiB в сутки ~ 1 гигабайт в секунд

Yahoo! Cloud Serving Benchmark

- Находим оборудование
- Генерим данные за 2 недели
- Запускаем тесты
- **Отчёт на стол начальству 20 января**

Yahoo! Cloud Ser

- Находим оборуд
- Генерим данные
- Запускаем тесты
- Отчёт на стол н



Cache, Hash, Trash*

“Computer Science has only three ideas: cache, hash, trash”

– Greg Ganger, CMU

["Trash" was more because it rhymed than because it was the ideal third item ;).]



Cache, Hash, Trash

- **Проверить генерацию данных на слабом оборудовании**
- 10 виртуалок по 8 ядер
- 50 GB за 30 минут... Мало!
- Иногда не генерит данные совсем

Cache, Hash, Trash

```
for (long seq = 0; seq < X; seq++)  
{  
    ycbs_key = 'user' + fnv1a(seq)  
    insert_row(ycbs_key, ...);  
}
```

Cache, Hash, Trash

seq	fnv1a(seq)
0	6284781860667377211
1	6284782960179005422
2	6284784059690633633
3	6284785159202261844
4	6284786258713890055
5	6284787358225518266
6	6284788457737146477
7	6284789557248774688
8	6284773064574351523
9	6284774164085979734

Cache, Hash, Trash

- Hash функция с коллизиями
- Значения начинают повторяться
- Повторения целыми интервалами
- Отбрасывание повторений

```
insert into ... on conflict ignore;
```

Cache, Hash, Trash

- Ускоряем - пачками заливаем!
- Увы, пошли Deadlock
 - Значения повторяются на небольшом расстоянии
 - Нарушается упорядоченность (A-B vs. B-A)

Cache,

- Ускор
- Увы, г
 - Зн
 - На



ЯНИИ

Cache, Hash, Trash

- Очередная попытка ускорения
 - Убираем хэширование
 - Заливаем по 100 строк за раз (batching)
 - Заливаем напрямую на нужный сервер (node-wise)
 - Убрать random-изацию данных на каждую строку
 - Предварительная загрузка 1000000 случайных значений

Cache, Hash, Trash

- Итоги
 - 150 GiB за 5 минут – 0.5 GiB в секунду
 - Статусная таблица
1 строка раз в 5 минут
 - Java реализация
 - И это просто какие-то виртуалочки...



Go!



Go!

Узел	Сервера planck-1 - planck-7	Сервер schrodinger
Процессор	2 × Intel Silver 4314 (16x2.4 ГГц HT)	
Память	256 ГБ — 8 × 32 ГБ DDR4 ECC Reg	
Диск	2 × 480 ГБ SSD SATA Enterprise 10 × 15360 ГБ SSD NVMe Enterprise	2 × 480 ГБ SSD SATA Enterprise
Сетевые карты	2 × 10 GE + MC-LAG to Private network 10 Гбит/s	
Материнская плата	Intel M50CYP1UR212 MB	X12DPI-N6

Go!

- ОС Debian 12
- RAID-0 для 10 x 15TiB дисков = 140 TiB под PGDATA
- Мониторинг для истории
 - pgpro-otel-collector
 - pgpro_stats / pgpro_pwr

Go!

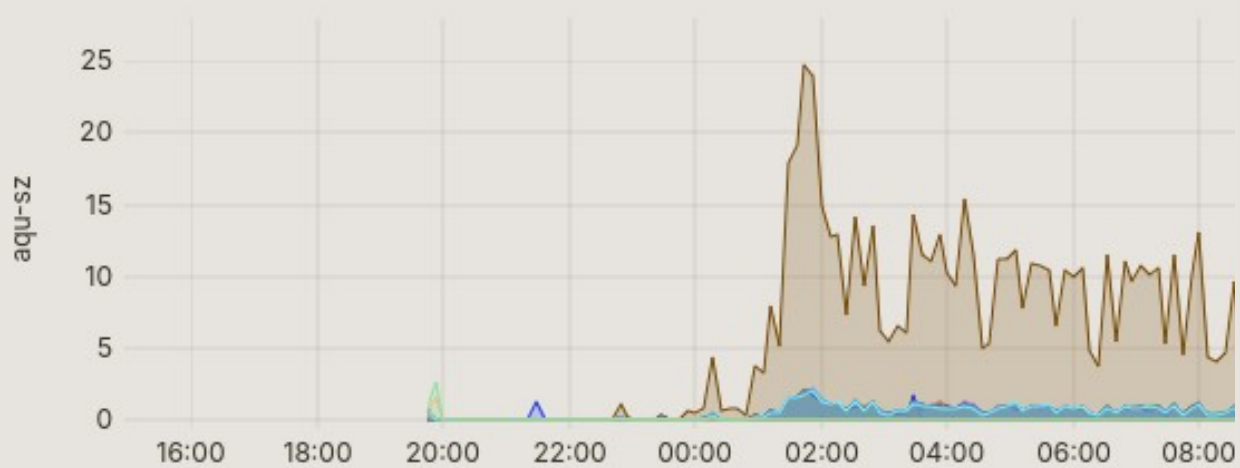
- ~150 ГБ за 10 минут на каждой из нод
- Запускали в ночь
- Хороший старт
 - planck-1: 629 GB
 - planck-2: 632 GB
 - planck-3: 620 GB
 - planck-4: 632 GB
 - planck-5: 631 GB
 - **planck-6: 975 GB**
 - planck-7: 626 GB



Go!

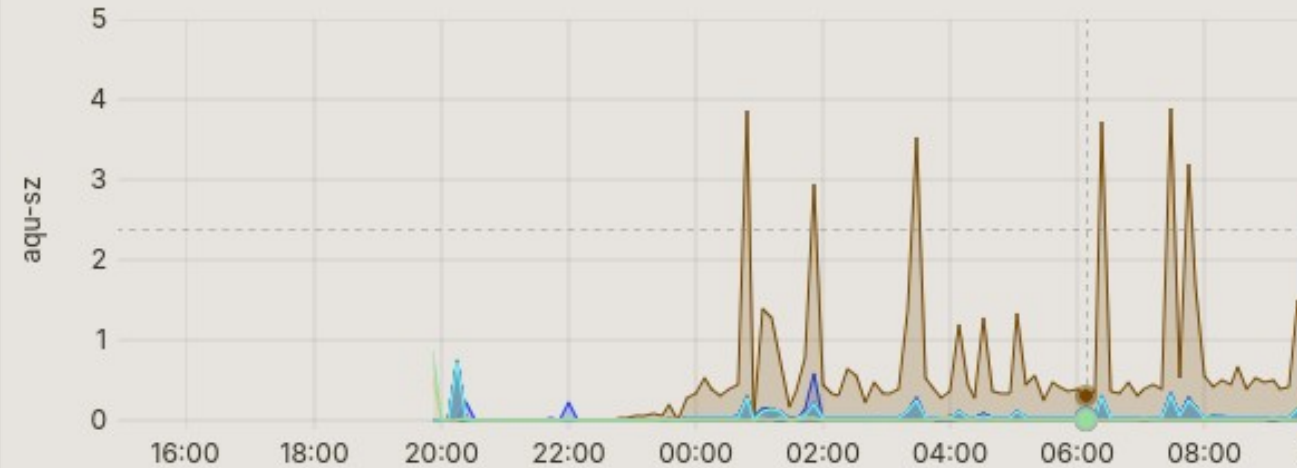
- Спустя 12 часов:
 - planck-1: 11T
 - planck-2: 11T
 - **planck-3: 3.5T**
 - planck-4: 11T
 - planck-5: 11T
 - **planck-6: 4.1T**
 - planck-7: 11T

Average Queue Size ⓘ



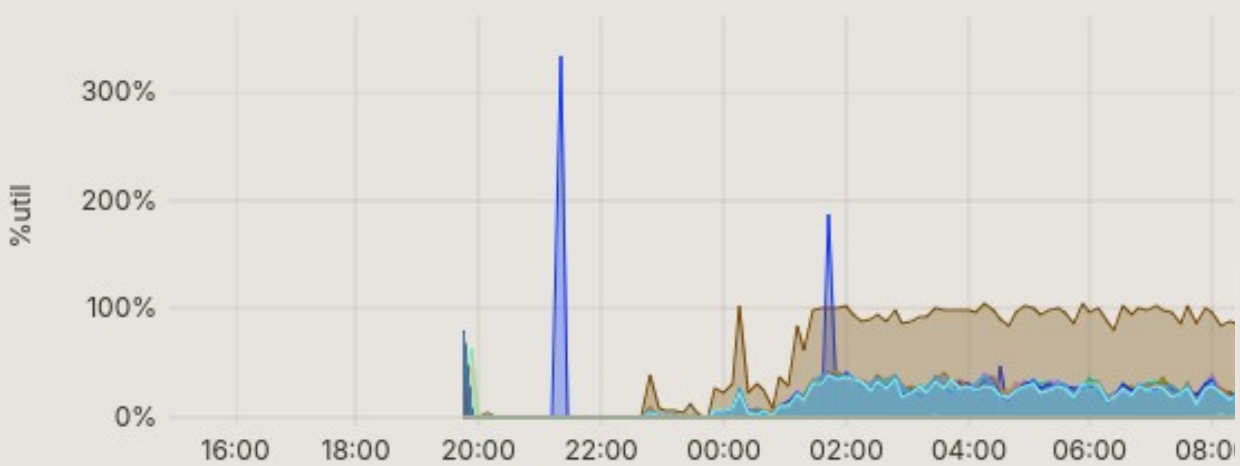
Name	Mean
md42	0
md43	0.0269
md44	7.16
nvme0n1	0.581

Average Queue Size ⓘ



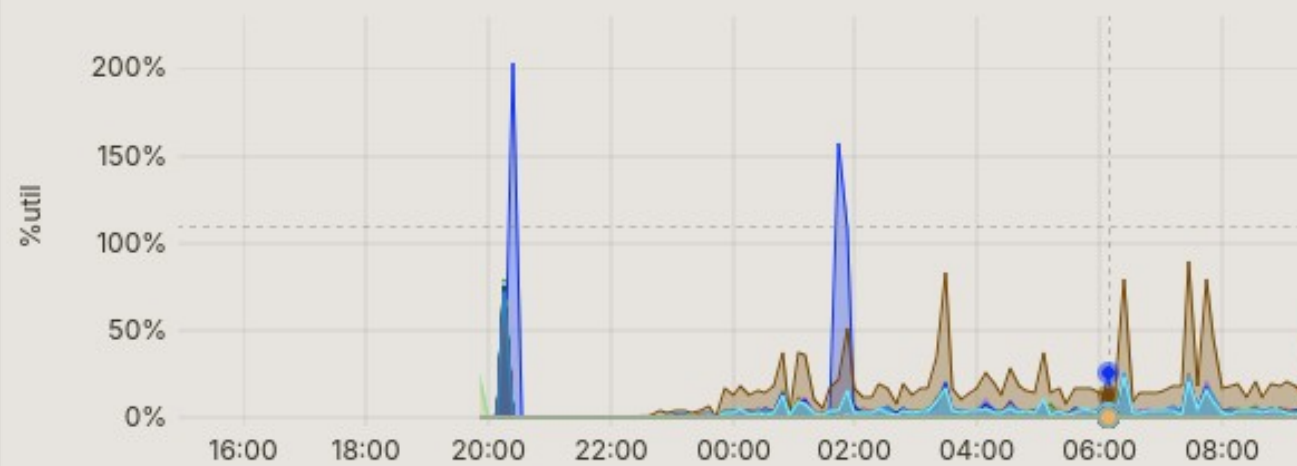
Name	Mean
md42	0
md43	0.0102
md44	0.574
nvme0n1	0.0547

Time Spent Doing I/Os ⓘ



Name	Mean
------	------

Time Spent Doing I/Os ⓘ



Name	Mean
------	------

Houston, we have a problem!



Houston, we have a problem

- **Попытка #1**
 - выполнили smart проверку дисков - **ОК**
 - сверили версии прошивок проблемных дисков, и прошивок дисков с серверов где проблем со скоростью записи не наблюдалось - **ОК**
 - перезагрузили сервер для проверки настроек bios - **ОК**
 - произвели замену кабелей для подключения NVMe дисков

Houston, we have a problem

- Результаты попытки #1
 - Времена обращения к диска упали с 10мс до 1мс
 - Но потом снова подросли и скорость записи упала

Houston, we have a problem

- **Попытка #2**
 - разобрать программный RAID на серверах planck-3 и planck-6
 - на каждом диске сделать по своей файловой системе и примонтировать в разные каталоги
 - подцепить их как tablespaces в shardman
 - для каждой партии нашей таблицы usertable использовать свой tablespace в Shardman-e

Houston, we have a problem

- **Попытка #2**
 - разобрать программный RAID на серверах planck-3 и planck-6
 - на каждом диске сделать по своей файловой системе и примонтировать в разные каталоги
 - **подцепить их как tablespaces в shardman**
 - для каждой партии нашей таблицы usertable использовать свой tablespace в Shardman-e

Houston, we have a problem

```
postgres=# CREATE TABLESPACE u02_01 LOCATION '/u02_01';
```

ERROR: local tablespaces are not supported

HINT: use "global" option to create global tablespace

Houston, we have a problem

- Глобальные табличные пространства
 - Одинаковая схема табличных пространств в кластере
 - Дополнительный аргумент `global`
- Два сервера – это не весь кластер
- Магическая настройка – `shardman.sync_schema`

Houston, we have a problem

```
postgres=# set shardman.sync_schema = off;
```

```
postgres=# CREATE TABLESPACE u02_01 LOCATION '/u02_01';
```

```
ERROR:  tablespace location template doesn't specify all  
necessary substitutions
```

```
HINT:   expected to see rigid word in location template
```

Houston, we have a problem

- **HINT: expected to see rgid word in location template**
- **rgid word** – resource group identifier, номер сервера
- **location template** – путь к табличному пространству
- Формат {rgid}

Houston, we have a problem

```
mkdir /u02_01/3
```

```
mkdir /u02_02/3
```

```
...
```

```
postgres=# CREATE TABLESPACE u02_01 LOCATION '/u02_01/{rgid}';
```

```
CREATE TABLESPACE
```

```
postgres=# CREATE TABLESPACE u02_02 LOCATION '/u02_02/{rgid}';
```

```
CREATE TABLESPACE
```

Houston, we have a problem

- Результаты попытки #2
 - Генерация побежала с хорошей скоростью
 - Мы пошли отмечать Новый Год!
 - Генерация продолжала до 500TiB

Ещё проблемы?

Здесь должна быть
шутка про **память**.
Но мы её забыли...
пог уф-лампой



Ещё проблемы?

- На календаре 9 января
- В логах на всех серверах кроме 7ого

**2025-01-09T16:58:52.754082+03:00 planck-3 kernel:
[1021273.775923] mce: [Hardware Error]: Machine check
events logged**

**2025-01-09T16:58:52.754511+03:00 planck-3 kernel:
[1021273.776582] EDAC skx MC5: HANDLING MCE MEMORY ERROR**

Ещё проблемы?

- Проверка через memtest
- Результаты
 - Замена планок памяти в 4 серверах
 - Замена 1 вентилятора в planck-2

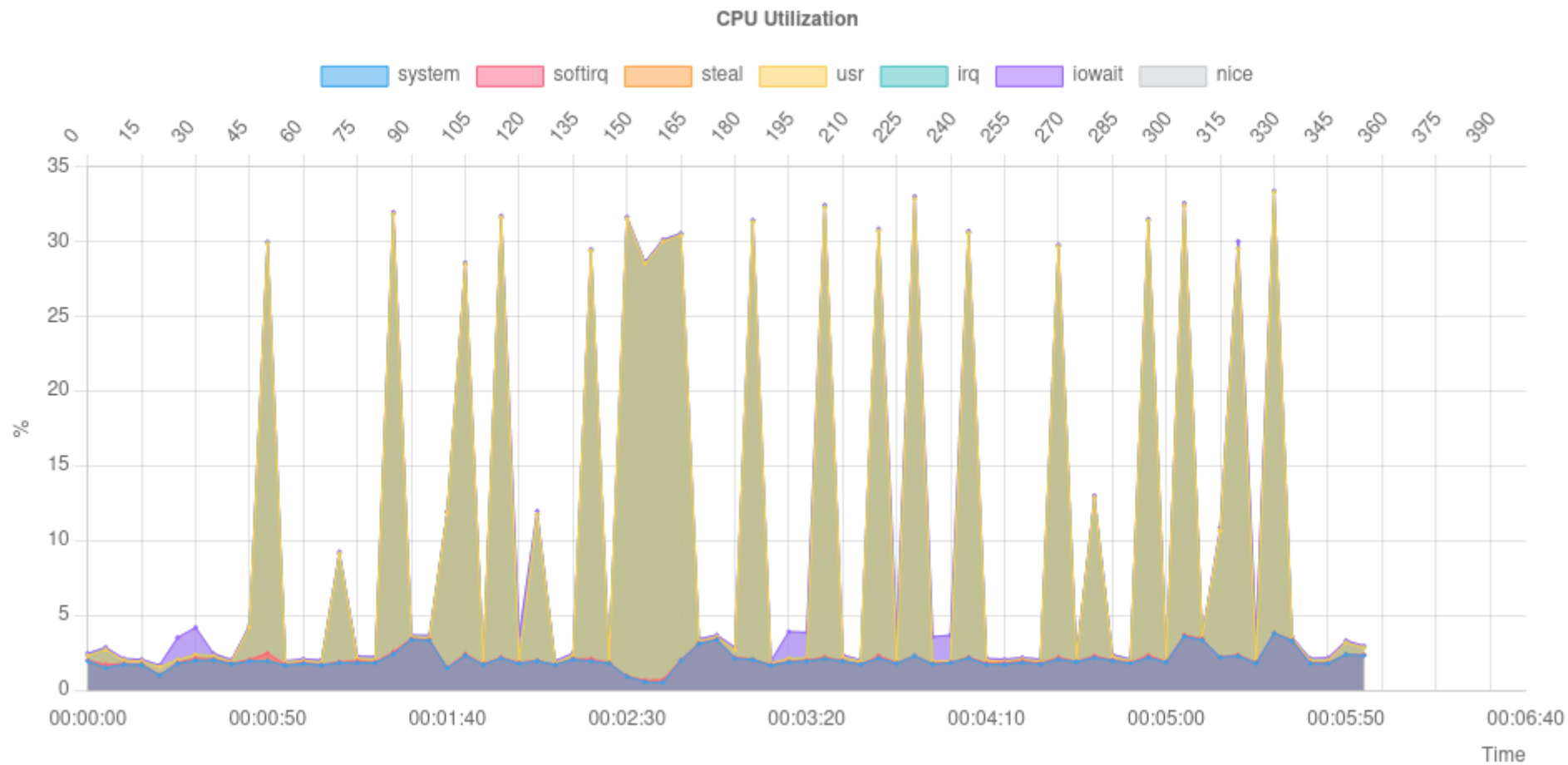
Ещё проблемы?

- Текущее состояние
 - 500 TiB сгенерили
 - Открытых проблем с генерацией нет
 - **Но никто не проверял OLTP нагрузку**

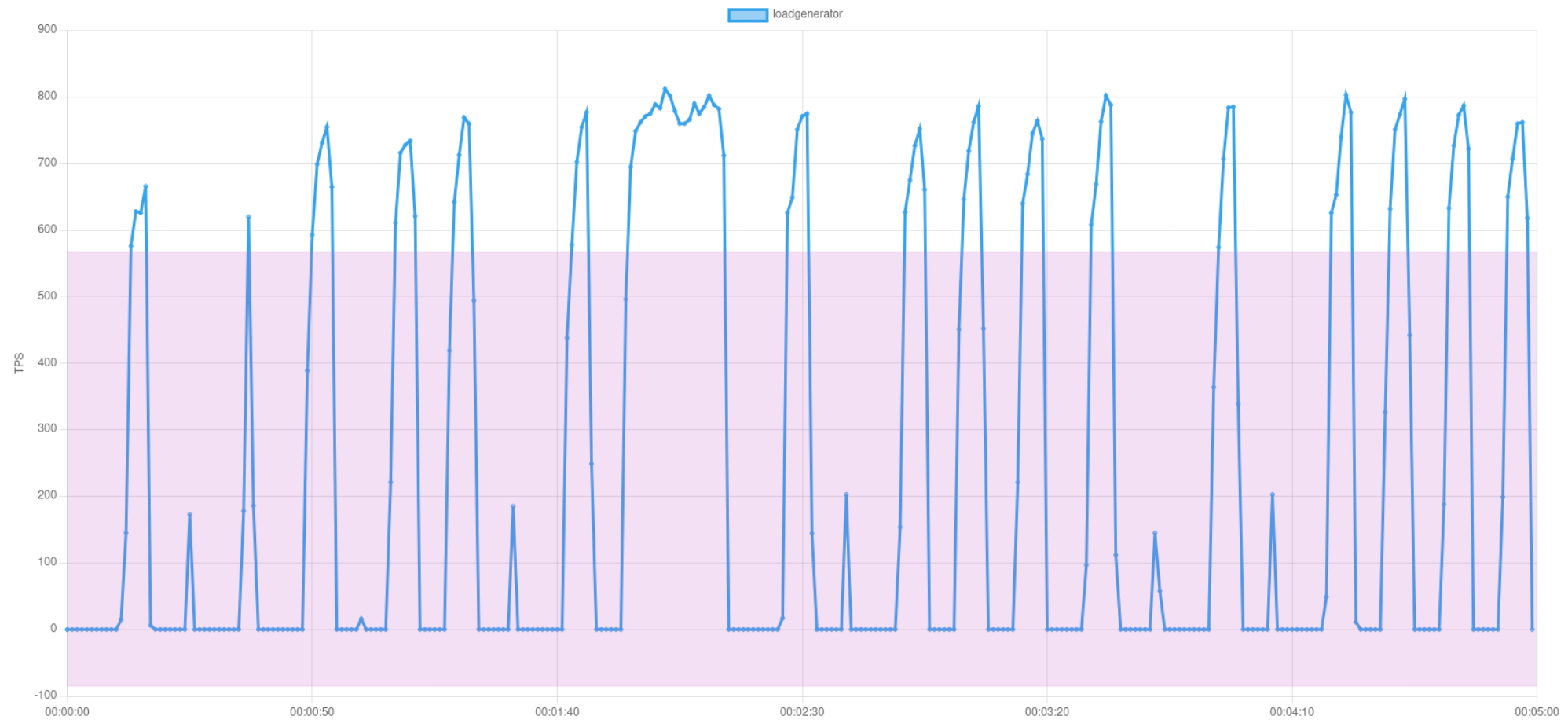
Ещё проблемы?

To\From	planck-1	planck-2	planck-3	planck-4	planck-5	planck-6	planck-7
planck-1	15805	710	764	649	751	734	242
planck-2	741	10987	725	629	677	697	229
planck-3	984	905	23673	825	914	931	333
planck-4	779	634	670	10220	623	633	214
planck-5	771	712	764	628	7830	695	
planck-6	810	751	802	654	693	37807	219
planck-7	223	236	219	206	210	330	1663

Ещё проблемы?

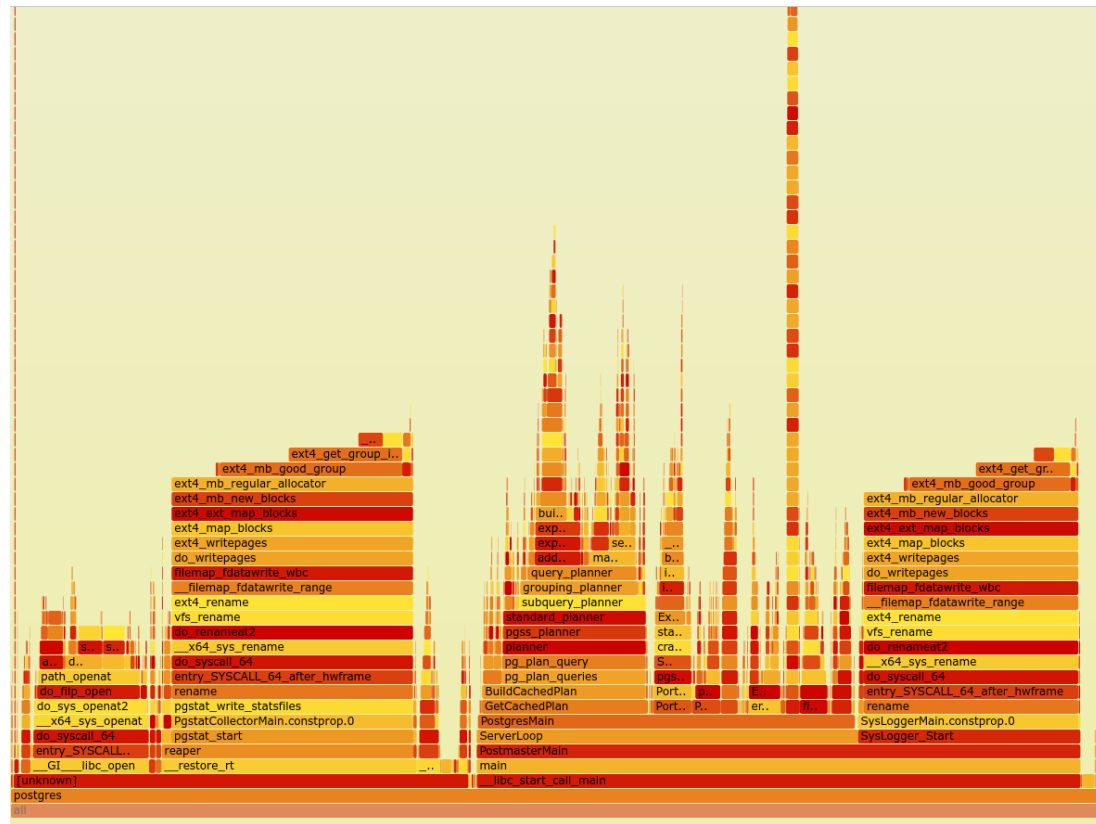


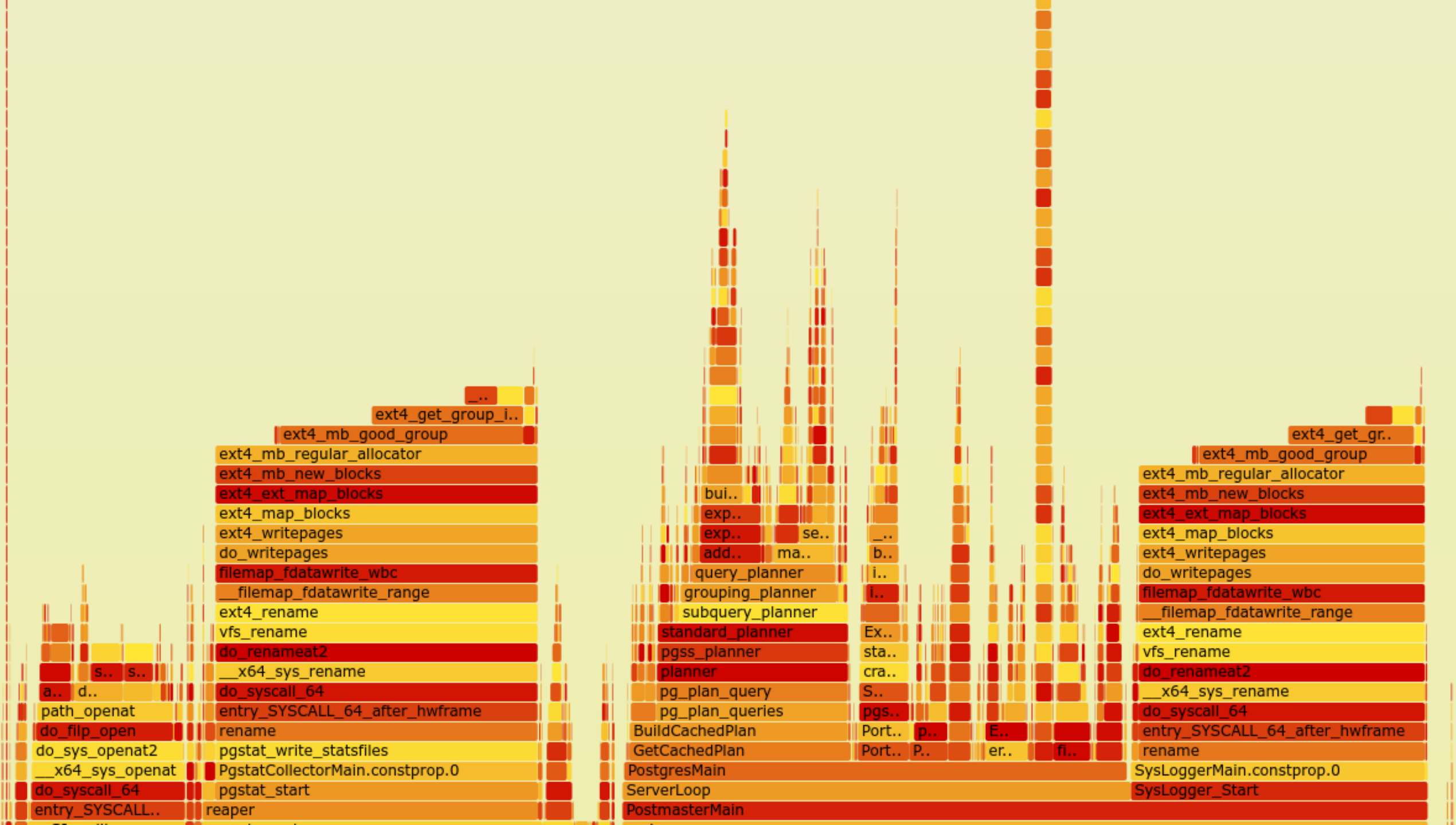
Ещё проблемы?



Ещё проблемы?

- `perf record -F 99 -a -g --call-graph=dwarf sleep 30`





Ещё проблемы?

53

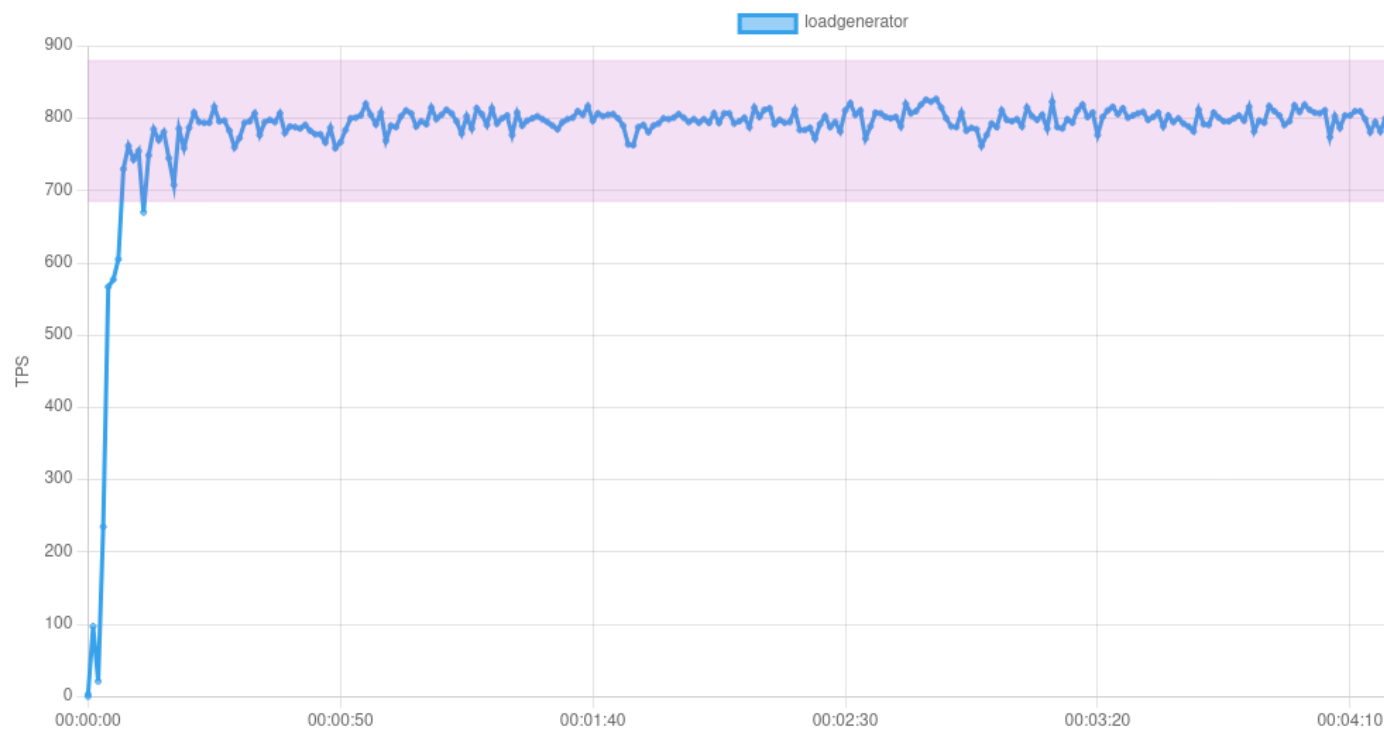
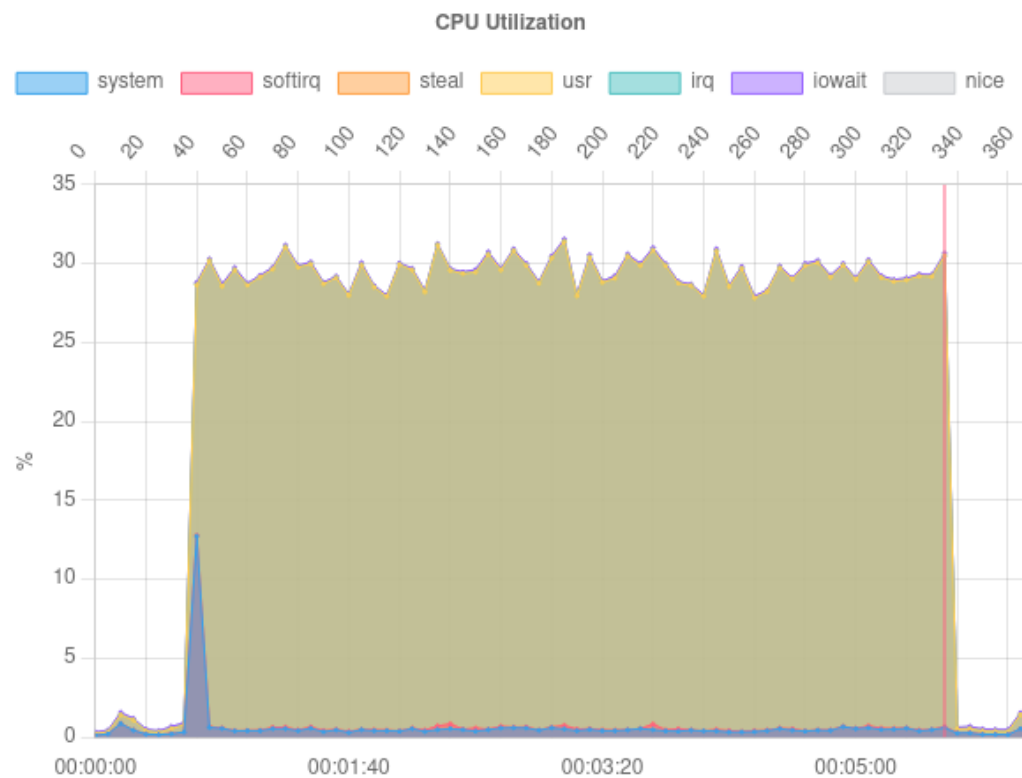
Ещё проблемы?

- ext4_mb_good_group
- https://bugzilla.kernel.org/show_bug.cgi?id=217965
- Patch landed in 6.8-rc1
- https://ahelpme.com/linux/degraded-ext4-performance-with-millions-files-because-of-high-system-time-in-ext4_mb_good_group-100-kworker-u32flush/
- “it seems similar problem and I remounted with the stripe=0 and so far it looks good”.
- stripe=1280:

```
/dev/md44 /u02 ext4 rw,relatime,stripe=1280 0 2
```

Ещё проблемы?

- stripe=0 + remount



Ещё проблемы?

- `man ext4`

stripe=N

Number of file system blocks that `mballoc` will try to use for allocation size and alignment. For RAID5/6 systems this should be the number of data disks * RAID chunk size in file system blocks.

Полумарафонский финиш...



Полумарафонский финиш...

- К 15 января...

Узел	\l+ postgres	Кол-во загруженных строк по данным monitoring_insert
planck-1	126 TB	799447789415
planck-2	126 TB	799506350001
planck-3	107 TB	799998897115
planck-4	126 TB	799586514368
planck-5	126 TB	799559145747
planck-6	126 TB	799857655389
planck-7	126 TB	799346369872
	Итого: 863 TB	max: 799998897115

Полумарафонский финиш...

- pg_stat_activity – автовакуум процессы
- pg_stat_progress_cluster

pid	duration	wait_event	mode	database	table	phase	table_size	total_size	scanned	scanned_pct
1957485	1 day 01:41:41.759642	Timeout.VacuumDelay	wraparound	postgres	usertable_0	scanning heap	9309 GB	13 TB	3210 GB	34.5
1957487	1 day 01:41:40.691321	LWLock.WALWrite	wraparound	postgres	usertable_7	scanning heap	9266 GB	13 TB	3230 GB	34.9
1957491	1 day 01:41:39.662693	IO.WALWrite	wraparound	postgres	usertable_56	scanning heap	9266 GB	13 TB	3553 GB	38.3
1957504	1 day 01:41:38.557464	LWLock.WALWrite	wraparound	postgres	usertable_63	scanning heap	9266 GB	13 TB	3562 GB	38.4
2207587	10:57:22.773513	Timeout.VacuumDelay	regular	postgres	usertable_21	scanning heap	11 TB	13 TB	9856 GB	88.6
2207588	10:57:21.801343	LWLock.WALWrite	regular	postgres	usertable_35	scanning heap	11 TB	13 TB	9855 GB	88.6
2207593	10:57:20.791863	Timeout.VacuumDelay	regular	postgres	usertable_14	scanning heap	11 TB	13 TB	9856 GB	88.6
2207606	10:57:19.790292	Timeout.VacuumDelay	regular	postgres	usertable_28	scanning heap	11 TB	13 TB	9856 GB	88.6

Полумарафонский финиш...

- Ускоряем раз!
- `autovacuum_vacuum_cost_delay = 0`

pid	duration	wait_event	mode	database	table	phase	table_size	total_size	scanned	scanned_pct
2429208	00:42:25.247109	LWLock.WALWrite	wraparound	postgres	usertable_5	scanning heap	12 TB	13 TB	452 GB	3.7
2429213	00:42:24.36491	LWLock.WALWrite	wraparound	postgres	usertable_12	scanning heap	12 TB	13 TB	778 GB	6.3
2429216	00:42:23.352458	LWLock.WALWrite	wraparound	postgres	usertable_19	scanning heap	12 TB	13 TB	451 GB	3.7
2429231	00:42:22.34729	LWLock.WALWrite	wraparound	postgres	usertable_33	scanning heap	12 TB	13 TB	452 GB	3.7
2429233	00:42:21.352138	LWLock.WALWrite	wraparound	postgres	usertable_26	scanning heap	12 TB	13 TB	451 GB	3.7
2429238	00:42:20.35141	LWLock.WALWrite	wraparound	postgres	usertable_40	scanning heap	12 TB	13 TB	451 GB	3.6
2429240	00:42:19.350775	IO.WALSync	wraparound	postgres	usertable_47	scanning heap	12 TB	13 TB	450 GB	3.6
2429242	00:42:18.349118	LWLock.WALWrite	wraparound	postgres	usertable_54	scanning heap	12 TB	13 TB	450 GB	3.6
2429246	00:42:17.336848	LWLock.WALWrite	wraparound	postgres	usertable_61	scanning heap	12 TB	13 TB	78 GB	0.6
2429259	00:42:16.347052	LWLock.WALWrite	wraparound	postgres	usertable_68	scanning heap	12 TB	13 TB	78 GB	0.6

(10 rows)

Полумарафонский финиш...

- Ускоряем два!
- pg_wal -> tmpfs

pid	duration	wait_event	mode	database	table	phase	table_size	total_size	scanned	scanned_pct
2500393	00:02:33.715813	LWLock.WALWrite	wraparound	postgres	usertable_5	scanning heap	12 TB	13 TB	816 GB	6.6
2500397	00:02:32.790738	IO.DataFileWrite	wraparound	postgres	usertable_12	scanning heap	12 TB	13 TB	1141 GB	9.2
2500398	00:02:31.788569	LWLock.WALWrite	wraparound	postgres	usertable_19	scanning heap	12 TB	13 TB	813 GB	6.6
2500407	00:02:30.788247		wraparound	postgres	usertable_33	scanning heap	12 TB	13 TB	814 GB	6.6
2500408	00:02:29.788889	LWLock.WALWrite	wraparound	postgres	usertable_26	scanning heap	12 TB	13 TB	813 GB	6.6
2500422	00:02:28.786883		wraparound	postgres	usertable_40	scanning heap	12 TB	13 TB	812 GB	6.5
2500426	00:02:27.785951	IO.WALWrite	wraparound	postgres	usertable_47	scanning heap	12 TB	13 TB	811 GB	6.6
2500427	00:02:26.785828		wraparound	postgres	usertable_54	scanning heap	12 TB	13 TB	811 GB	6.6
2500432	00:02:25.785364	IO.DataFileRead	wraparound	postgres	usertable_61	scanning heap	12 TB	13 TB	440 GB	3.6
2500438	00:02:24.732249		wraparound	postgres	usertable_68	scanning heap	12 TB	13 TB	438 GB	3.6

(10 rows)

Полумарафонский финиш...

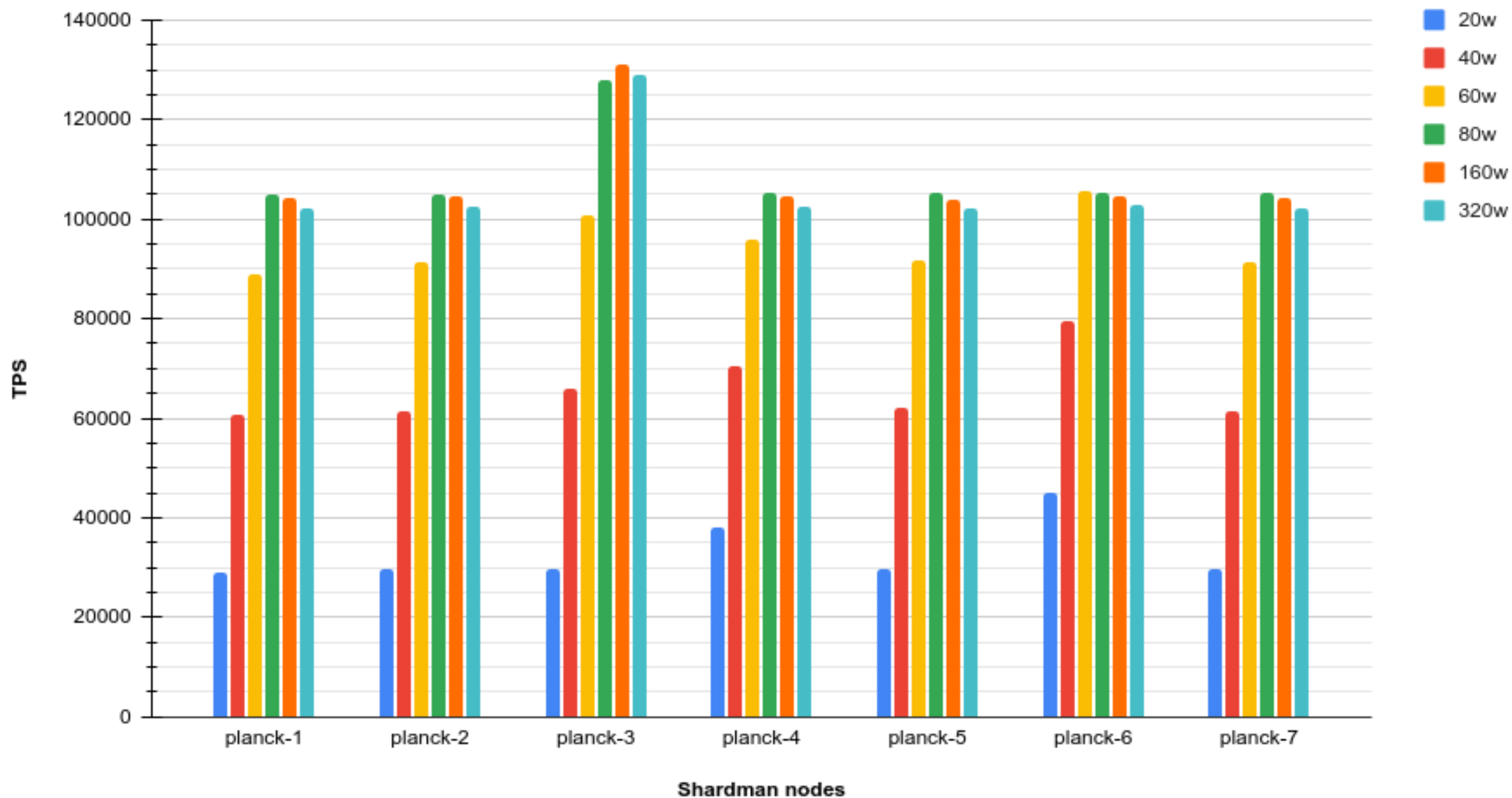
- Через 18 часов автовакуумы закончили работу
- Потом были замеры, страдания, боли
- Написание на коленках генераторов нагрузки ака YahooSimple*
- И – итоги...

ИТОГИ



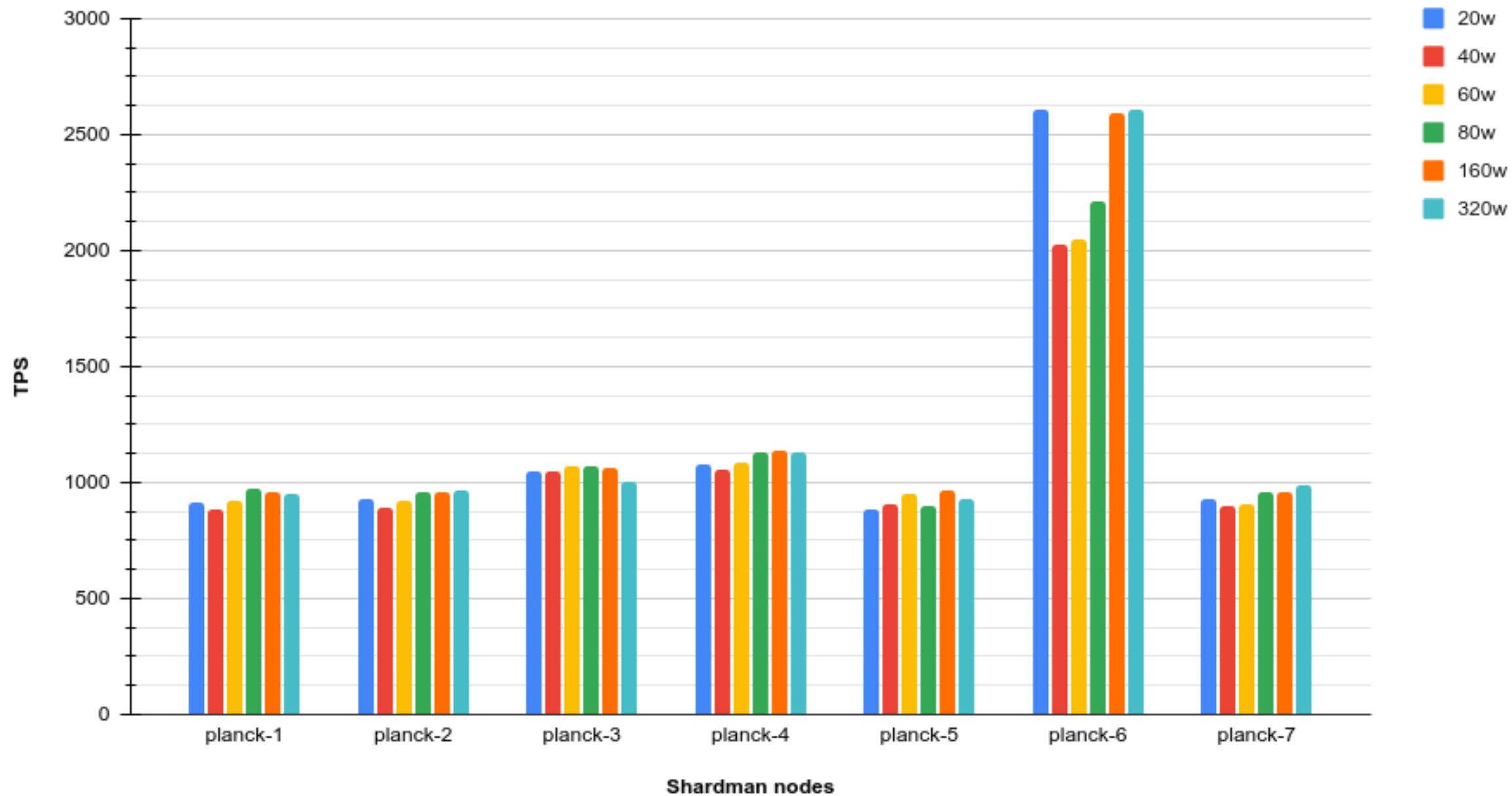
YahooSimpleSelect

Reading from the node itself, different num of workers



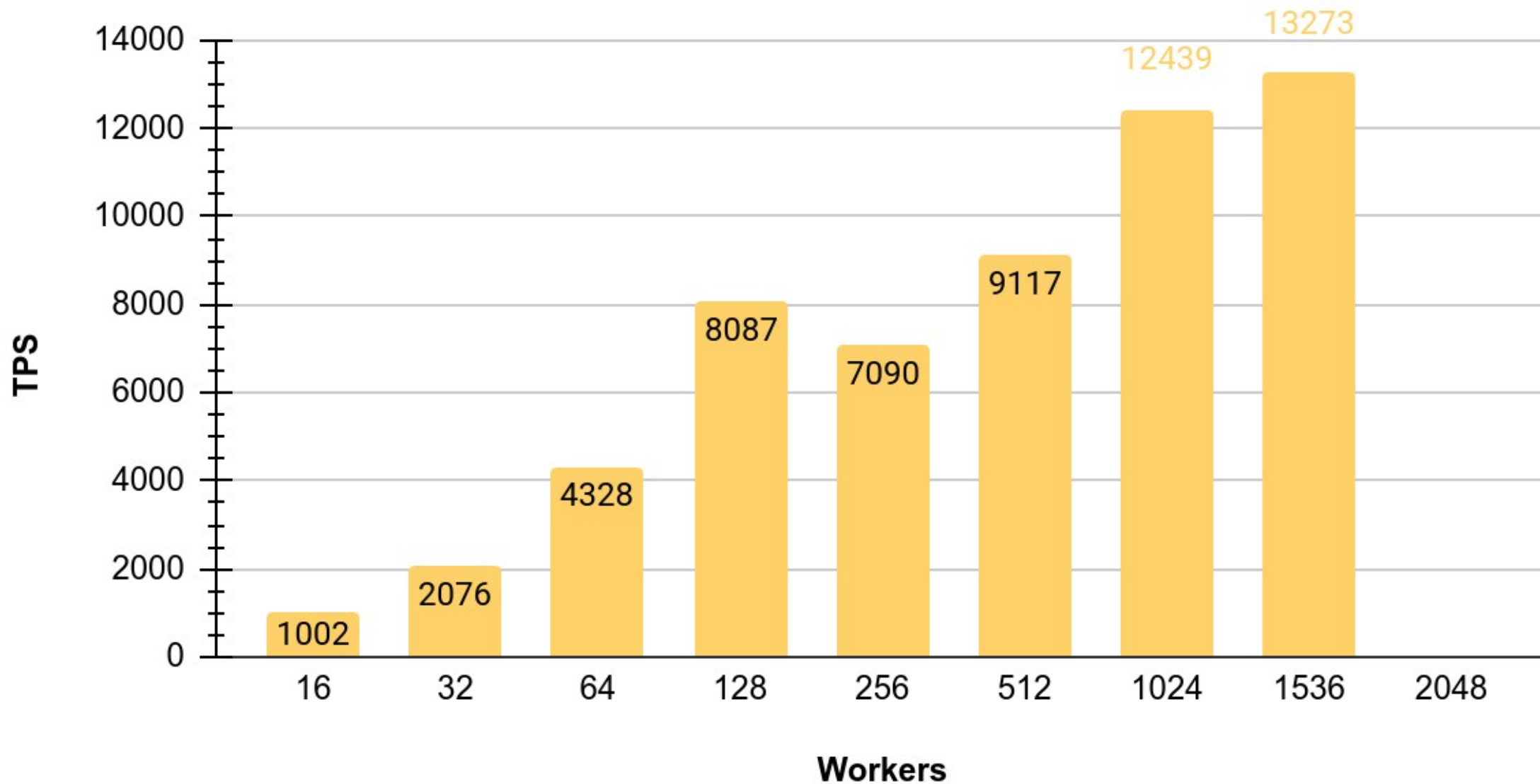
YahooSimpleUpdate

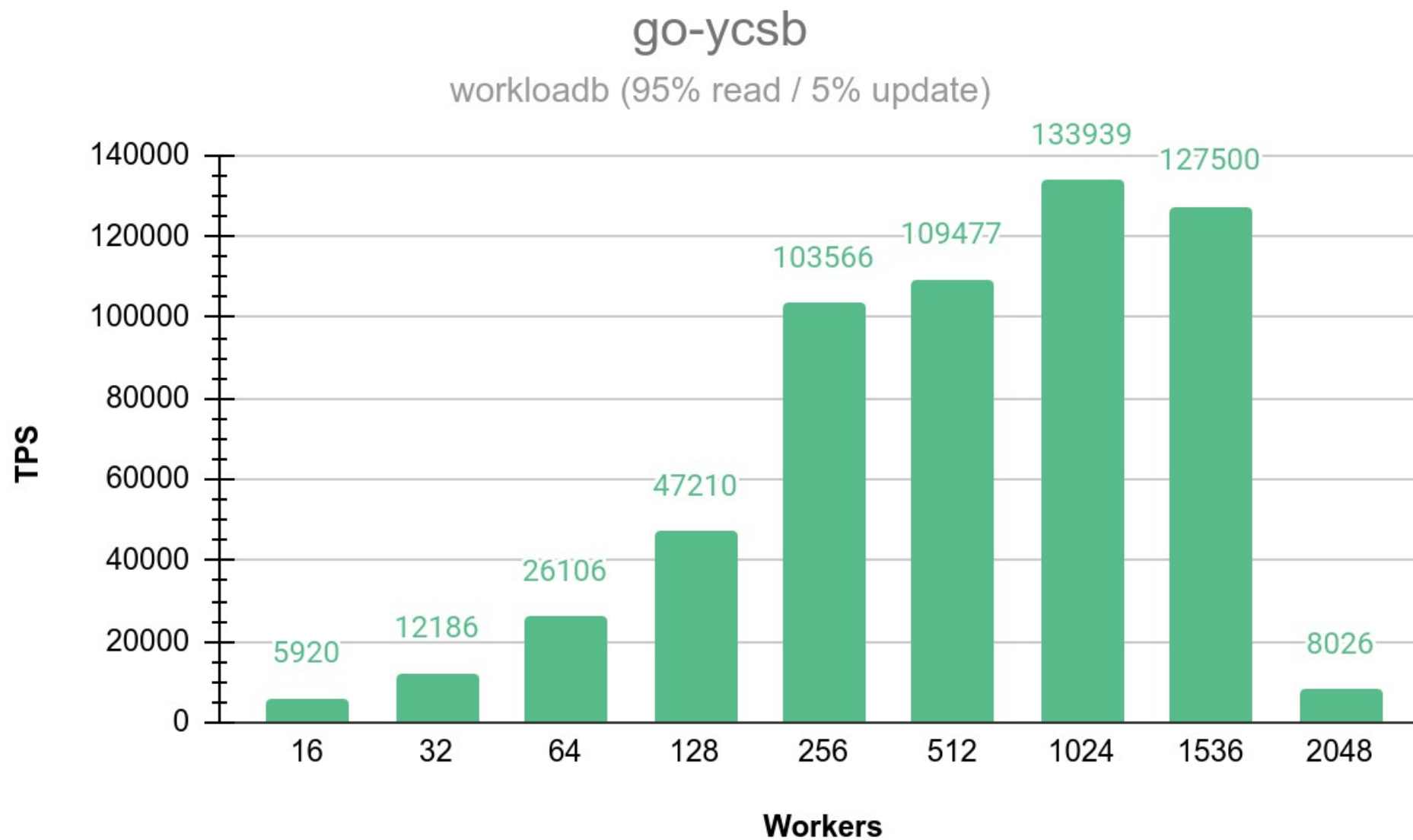
Updating in the node itself, different num of workers



go-ycsb

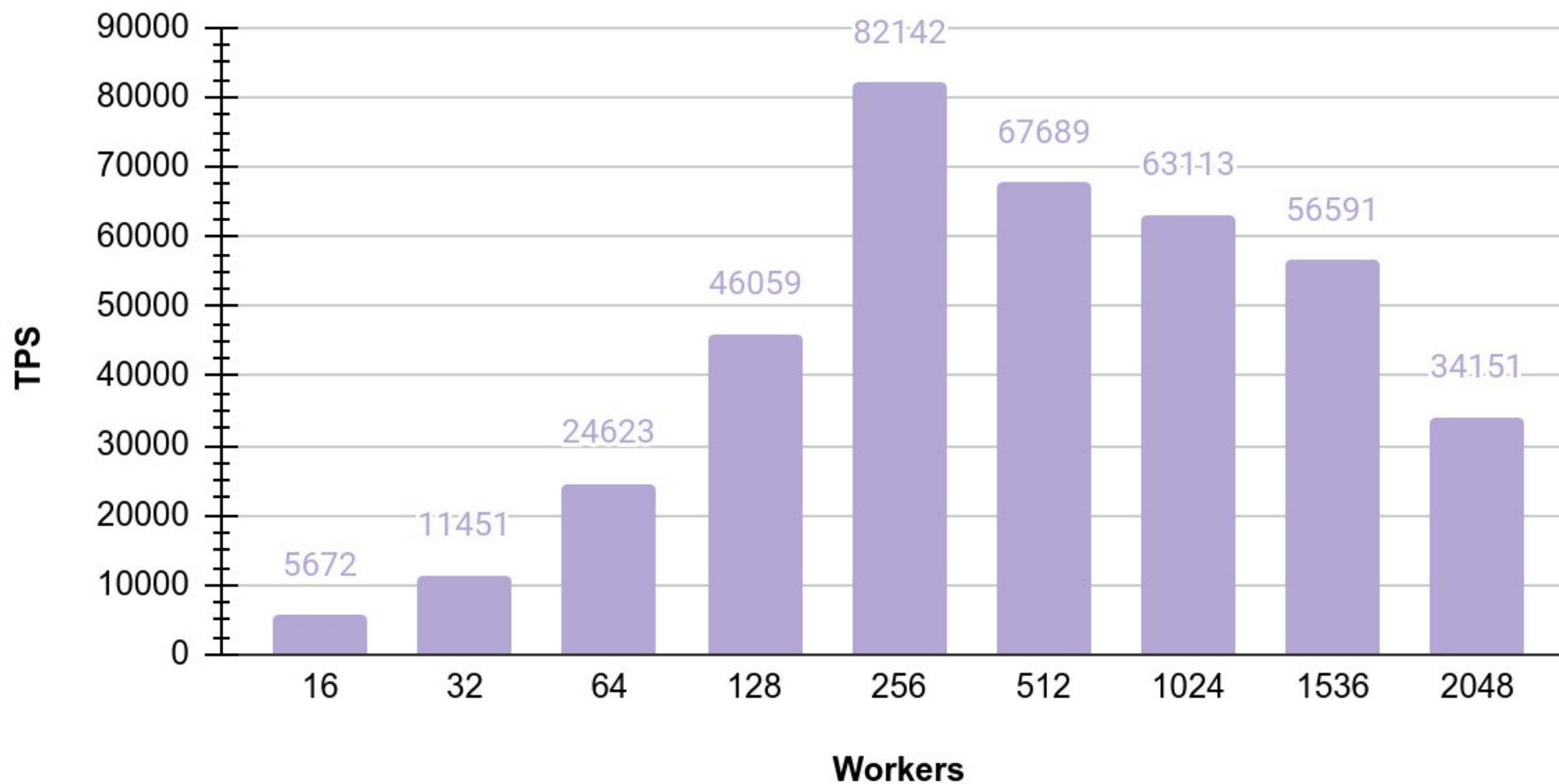
workloada (50% read / 50% update)





go-ycsb

workloadf (50% read / 50% read-modify-write)



Итоги

- А 20 января засбоила очередная планка...
- Но это была уже другая история



ИТОГИ

- Достижения
 - 1200 сообщений во внутреннем чате
 - 23 часа overtime-а сотрудников в новогодние праздники
 - Много сбойных планок памяти
 - 1 сбойный вентилятор
 - 100+ GiB данных (логи, метрики, результаты замеров)

ИТОГИ

- Стоило бы сделать (или хотя бы в следующий раз)
 - Запас по дискам и времени
 - Надо писать бенчмарк с нуля (для понимания РiВ бизнеса)
 - Магические свойства planck-6 и planck-3

ИТОГИ

- Новое узнали
 - rasdaemon – следит за сбоями оборудования
 - Настройки OS: проверять stripe в ext4 на 6.6
 - Много нового про Shardman

ИТОГИ

- Планы на будущее
 - Повторить на своём оборудовании
 - Подумать над новым benchmark-ом
 - Хочется больше TPS – как минимум 1 миллион TPS

ИТОГИ

<https://habr.com/ru/companies/postgrespro/articles/908604/>



mizhka



Как мы под Новый Год загрузили в PostgreSQL петабайт данных и что из этого вышло

 Средний

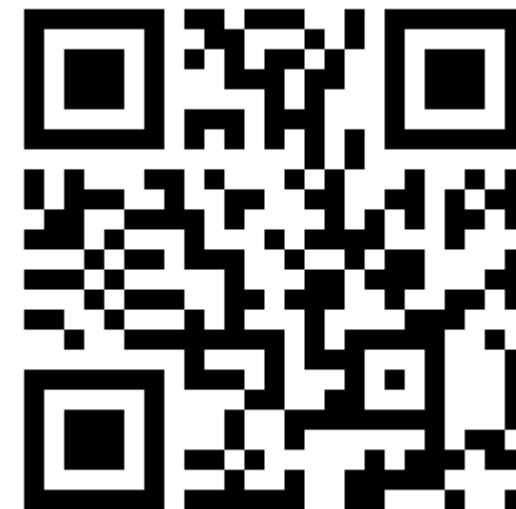
 19 мин

 6

Блог компании [Postgres Professional](#), PostgreSQL*, Администрирование баз данных*

Эта история началась с шутки на офисной кухне 10 декабря, но, как водится, у каждой приличной шутки, она вдруг стала интересной для воплощения, а в конце переросла в не самую технически простую реализацию с хождением по многочисленным граблям.

А началось всё просто: пока все вокруг спорят как настраивать железо и тюнить



Спасибо за внимание!

Михаил Жилин

