«Долгоиграющие» базы данных PostgreSQL

Юрий «@Jay7t» Бушмелев

echo
REAL-TIME WEB PLATFORM

LVEE 2013

«Долгоиграющие»?

- Время жизни
- Объём
- Нагрузка
- Высокая доступность

Проблемы

- Время жизни → «распухание» индексов и таблиц
- Объём → проблемы с бэкапом
- Нагрузка → блокировки
- Высокая доступность → особенности репликации

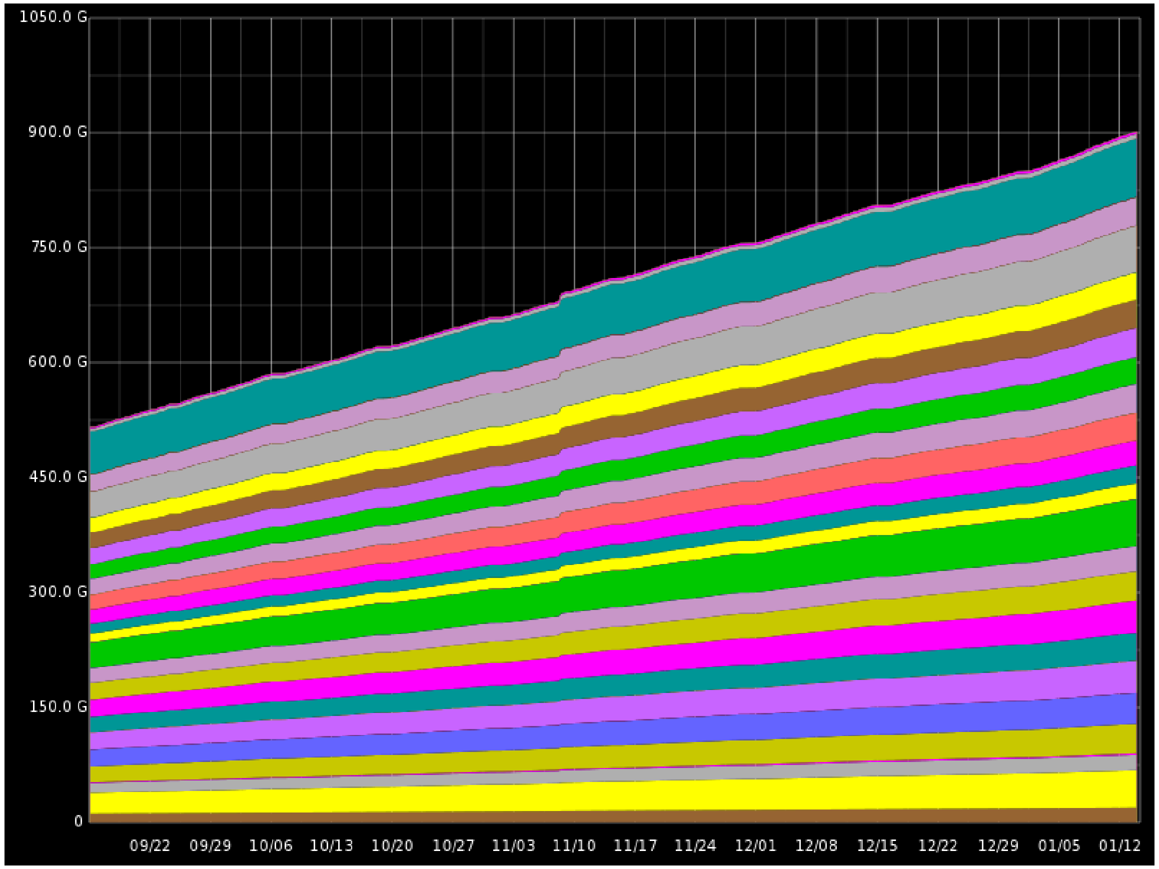
* в реальности связи сложнее

Опыт Echo

- 50 серверов БД
- Суммарный объём баз — более 4Tb
- Суммарная нагрузка — более 30 000 tps
- Потоковая репликация (streaming replication)
- Очень много индексов
  (100Gb — данные / 870Gb — 33 индекса)

«Опухшие»

- Таблицы и индексы «пухнут»...



… и «дохнут» :(

Оценка «пухлости» индексов — 1

```
SELECT current_database(), schemaname, tablename, ROUND(CASE WHEN otta=0 THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS tbloat, CASE WHEN relpages < otta THEN 0 ELSE bs*(sml.relpages-otta)::bigint END AS wastedbytes, iname, ROUND(CASE WHEN iotta=0 OR ipages=0 THEN 0.0 ELSE ipages/iotta::numeric END,1) AS ibloat, CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta) END AS wastedibytes FROM ( SELECT schemaname, tablename, cc.reltuples, cc.relpages, bs, CEIL((cc.reltuples*((datahdr+ma-(CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma END))+nullhdr2+4))/(bs-20::float)) AS otta, COALESCE(c2.relname,'?') AS iname, COALESCE(c2.reltuples,0) AS ituples, COALESCE(c2.relpages,0) AS ipages, COALESCE(CEIL((c2.reltuples*(datahdr-12))/(bs-20::float)),0) AS iotta -- very rough approximation, assumes all cols FROM (SELECT ma,bs,schemaname,tablename, (datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma END)))::numeric AS datahdr, (maxfracsum*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%ma END))) AS nullhdr2 FROM (SELECT schemaname, tablename, hdr, ma, bs, SUM((1-null_frac)*avg_width) AS datawidth, MAX(null_frac) AS maxfracsum, hdr+(SELECT 1+count(*)/8 FROM pg_stats s2 WHERE null_frac<>0 AND s2.schemaname = s.schemaname AND s2.tablename = s.tablename) AS nullhdr FROM pg_stats s, (SELECT (SELECT current_setting('block_size')::numeric) AS bs, CASE WHEN substring(v,12,3) IN ('8.0','8.1','8.2') THEN 27 ELSE 23 END AS hdr, CASE WHEN v ~ 'mingw32' THEN 8 ELSE 4 END AS ma FROM (SELECT version() AS v) AS foo) AS constants GROUP BY 1,2,3,4,5) AS foo ) AS rs JOIN pg_class cc ON cc.relname = rs.tablename JOIN pg_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname = rs.schemaname AND nn.nspname <> 'information_schema' LEFT JOIN pg_index i ON indrelid = cc.oid LEFT JOIN pg_class c2 ON c2.oid = i.indexrelid) AS sml ORDER BY wastedbytes DESC
```

http://wiki.postgresql.org/wiki/Show_database_bloat

Оценка «пухлости» индексов — 2

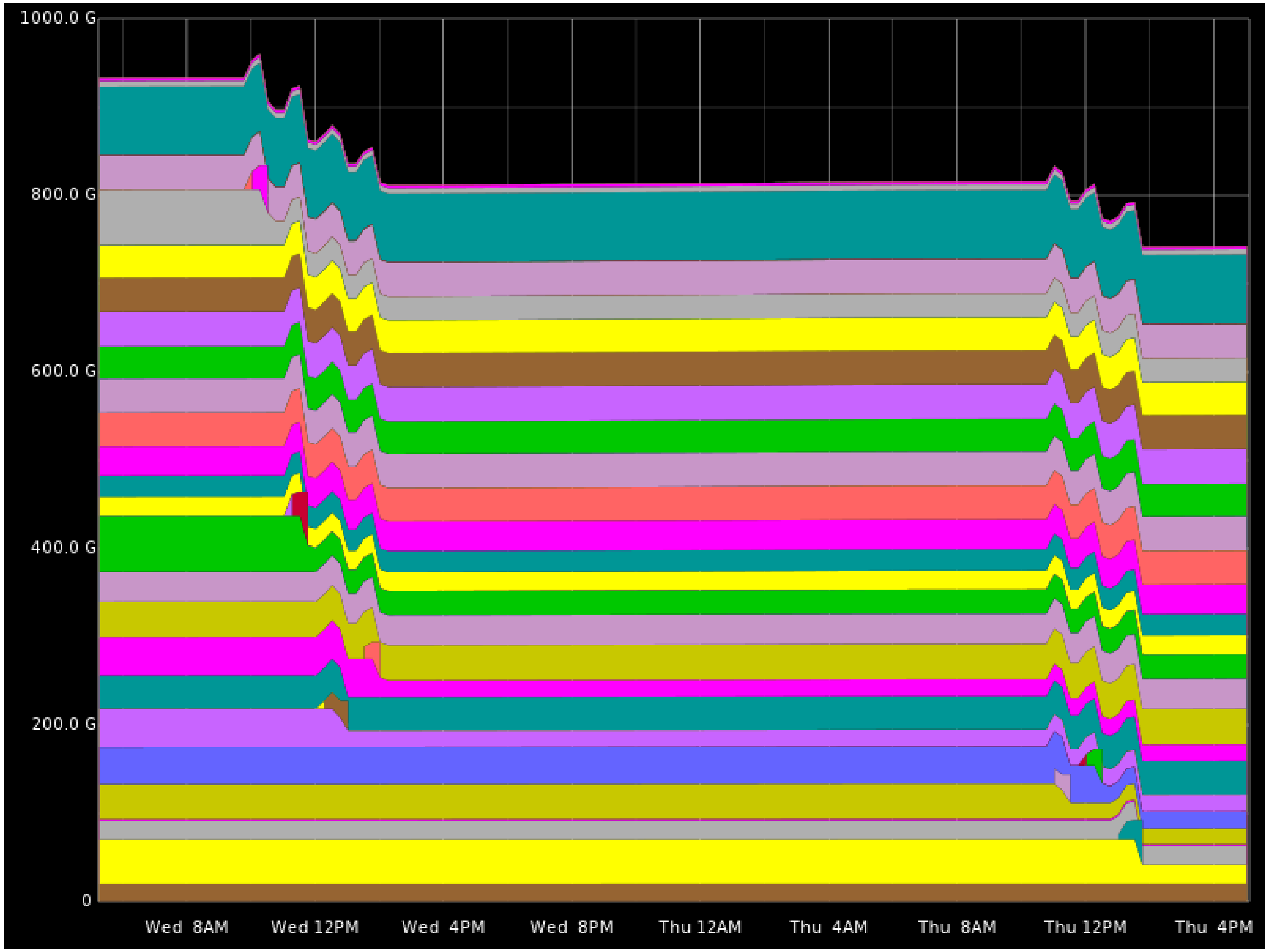
```
SELECT idx_name, pg_size_pretty(idx_size) AS size, pg_size_pretty(idx_ideal_size) AS ideal_size, CASE WHEN idx_size = 0 THEN
0 ELSE 100-idx_ideal_size*100/idx_size END AS bloat, idx_tup_read, regexp_replace(idx_def, 'INDEX ([^ ]+)', E'INDEX
CONCURRENTLY \\1_new') AS idx_def FROM (SELECT rel.relname AS rel_name, idx.relname AS idx_name, idx_tup_read,
pg_relation_size(idx.oid) AS idx_size,((ceil(idx.reltuples * ((constants.index_tuple_header_size + constants.item_id_data_size +
CASE WHEN (COALESCE(SUM(CASE WHEN statts.staattnotnull THEN 0 ELSE 1 END), 0::BIGINT) + ((SELECT
COALESCE(SUM(CASE WHEN atts.attnotnull THEN 0 ELSE 1 END), 0::BIGINT) FROM pg_attribute atts JOIN (SELECT
i.indkey[the.i] AS attnum FROM generate_series(0, i.indnatts - 1) the(i)) cols ON atts.attnum = cols.attnum WHERE atts.attrelid =
i.indrelid))) > 0 THEN (SELECT the.null_bitmap_size + constants.max_align - CASE WHEN (the.null_bitmap_size %
constants.max_align) = 0 THEN constants.max_align ELSE the.null_bitmap_size % constants.max_align END FROM (VALUES
(i.indnatts / 8 + CASE WHEN (i.indnatts % 8) = 0 THEN 0 ELSE 1 END)) the(null_bitmap_size)) ELSE 0 END)::DOUBLE PRECISION
+ COALESCE(SUM(statts.stawidth::DOUBLE PRECISION * (1::DOUBLE PRECISION - statts.stanullfrac)), 0::DOUBLE PRECISION)
+ COALESCE((SELECT SUM(atts.stawidth::DOUBLE PRECISION * (1::DOUBLE PRECISION - atts.stanullfrac)) FROM pg_statistic
atts JOIN (SELECT i.indkey[the.i] AS attnum FROM generate_series(0, i.indnatts - 1) the(i)) cols ON atts.staattnum = cols.attnum
WHERE atts.starelid = i.indrelid), 0::DOUBLE PRECISION)) / (constants.block_size - constants.page_header_data_size::NUMERIC -
constants.special_space::NUMERIC)::DOUBLE PRECISION) + constants.index_metadata_pages::DOUBLE PRECISION) *
constants.block_size::DOUBLE PRECISION)::BIGINT AS idx_ideal_size, pg_get_indexdef(idx.oid) AS idx_def FROM pg_index AS i
JOIN pg_class idx ON i.indexrelid = idx.oid JOIN pg_class rel ON i.indrelid = rel.oid JOIN pg_namespace ON idx.relnamespace =
pg_namespace.oid JOIN pg_stat_user_indexes AS sui ON (idx.oid = sui.indexrelid) LEFT JOIN (SELECT pg_statistic.starelid,
pg_statistic.staattnum, pg_statistic.stanullfrac, pg_statistic.stawidth, pg_attribute.attnotnull AS staattnotnull FROM pg_statistic JOIN
pg_attribute ON pg_statistic.starelid = pg_attribute.attrelid AND pg_statistic.staattnum = pg_attribute.attnum) AS statts ON
statts.starelid = idx.oid CROSS JOIN (SELECT current_setting('block_size'::TEXT)::NUMERIC AS block_size, CASE WHEN
substring(version(), 12, 3) = ANY (ARRAY['8.0'::TEXT, '8.1'::TEXT, '8.2'::TEXT]) THEN 27 ELSE 23 END AS tuple_header_size,
CASE WHEN version() ~ 'mingw32'::TEXT THEN 8 ELSE 4 END AS max_align, 8 AS index_tuple_header_size, 4 AS
item_id_data_size, 24 AS page_header_data_size, 0 AS special_space, 1 AS index_metadata_pages) AS constants WHERE NOT
i.indisprimary AND NOT i.indisunique GROUP BY rel.relname, rel.oid, idx.relname, idx.reltuples, idx.oid, i.indrelid, i.indkey, i.indnatts,
constants.block_size, constants.tuple_header_size, constants.max_align, constants.index_tuple_header_size,
constants.item_id_data_size, constants.page_header_data_size, constants.index_metadata_pages, constants.special_space,
idx_tup_read) AS t ORDER BY CASE WHEN idx_size = 0 THEN 0 ELSE 100-idx_ideal_size*100/idx_size END DESC, idx_tup_read
DESC
```

http://www.pgcon.org/2009/schedule/attachments/96_visualizing-postgres-2009-05-21.pdf

Борьба с «распуханием» индексов — 1

- REINDEX
- CREATE INDEX CONCURRENTLY

Борьба с «распуханием» индексов — 2



Борьба с «распуханием» вообще

- VACUUM FULL
- CLUSTER
- vacuum_table.pl → pgcompactor
- pg_reorg → pg_repack

Резервное копирование

- pg_dump
- Бэкап ФС + WAL-архивы
- Снапшот ФС + WAL-архивы
- pg_rman
- WAL-E

Потоковая репликация (streaming replication)

- Плюсы
  - Скорость
  - Стабильность
  - «Из коробки»
- Минусы
  - Полная копия мастера
  - Отсутствие гибкости

Создание «реплики» — 1

- scp -c arcfour{,128,256}
- rsync
- pg_basebackup
- repmgr

Создание «реплики» — 2

- rsync
  - RSYNC_RSH=
    "ssh -c arcfour256 -o Compression=no"
  - -rp --inplace –delete --compress-level=1
  - Сначала global/pg_control, потом все остальное
  - Незачем копировать pg_xlog, pg_log, pg_stat_tmp

«Ролевые игры» — 1

«Ролевые игры» — 2

«Ролевые игры» — 3

- Переключение ролей потоковой репликации без необходимости «наливки» слейва заново
  - Одинаковые postgresql.conf на мастере и слейве
  - Разные pg_hba.conf на мастере и слейве
  - Не используем trigger_file

«Ролевые игры» — 4

1. На старом мастере создать recovery.conf, поменять pg_hba.conf и перезапустить postgresql
2. На новом мастере удалить recovery.conf, поменять pg_hba.conf и перезапустить postgresql
3. На оставшихся слейвах – поменять recovery.conf и перезапустить postgresql

Вопросы?

До встречи на #ulcamp!

http://2013.ulcamp.ru