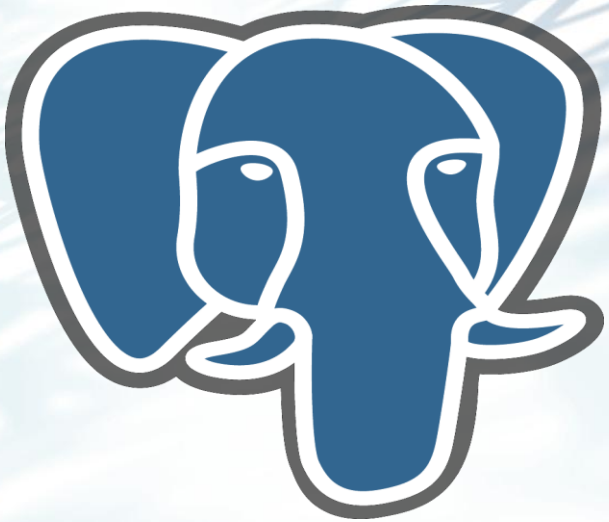
Architect of an Open World™

Ma base de données tiendrait-elle la charge ?



PG-Day-Fr
13 juin 2013

Philippe BEAUDOIN
Consultant Bases de Données

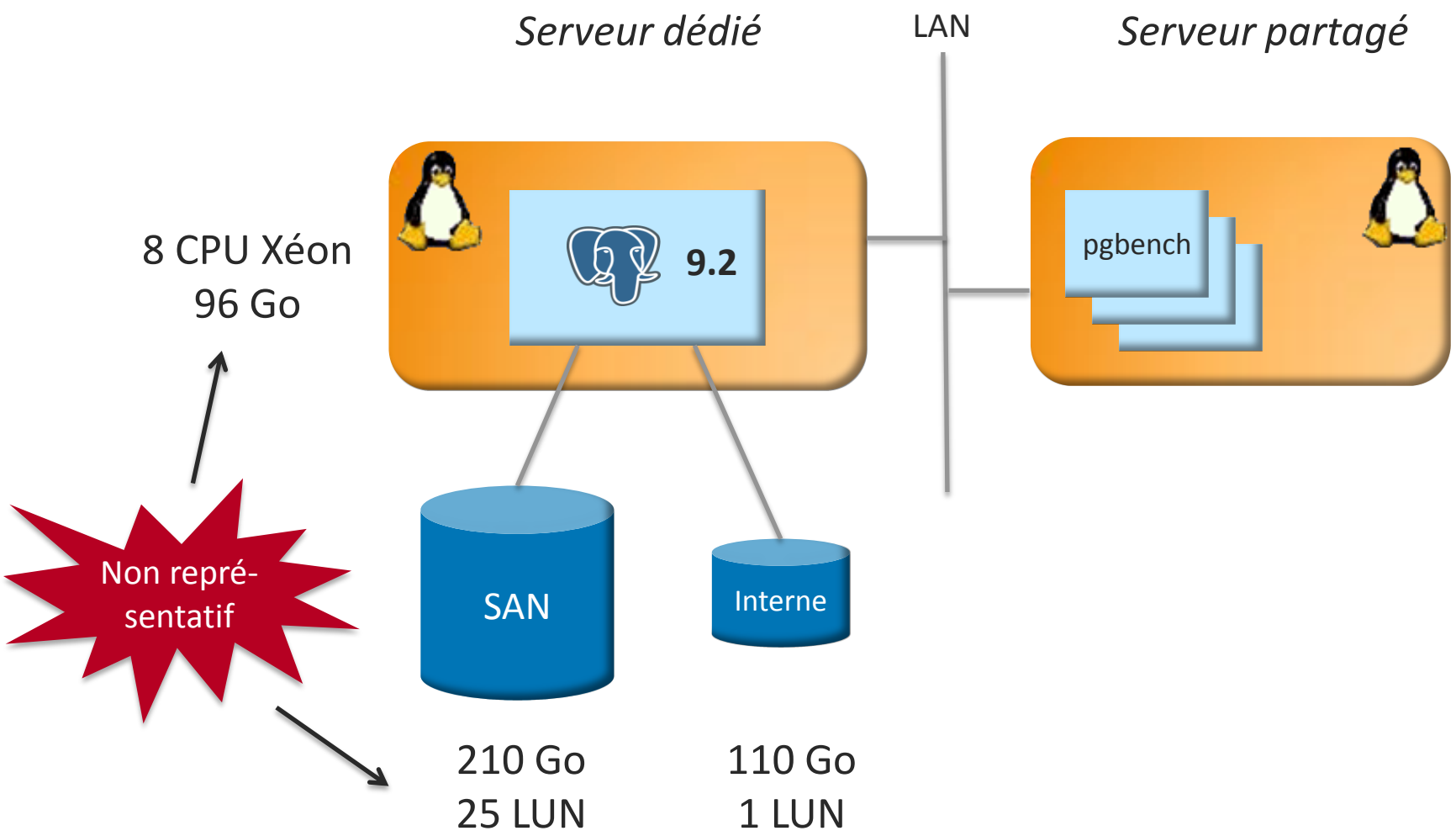
La question posée

- Travaux menés mi 2012 avec la C.N.A.F.
- Contexte CNAF actuel :
  - 120 bases de production PostgreSQL (2 par C.A.F.)
  - Entre 10 et 500 Go chacune, pour un total de 4 To
- Besoin de partage de données entre CAF
- Et si une seule base pour la France entière dans 3 ans ?
- PostgreSQL saura gérer les 10 To estimés
- Et la charge d'accès à la base ?

Démarche générale

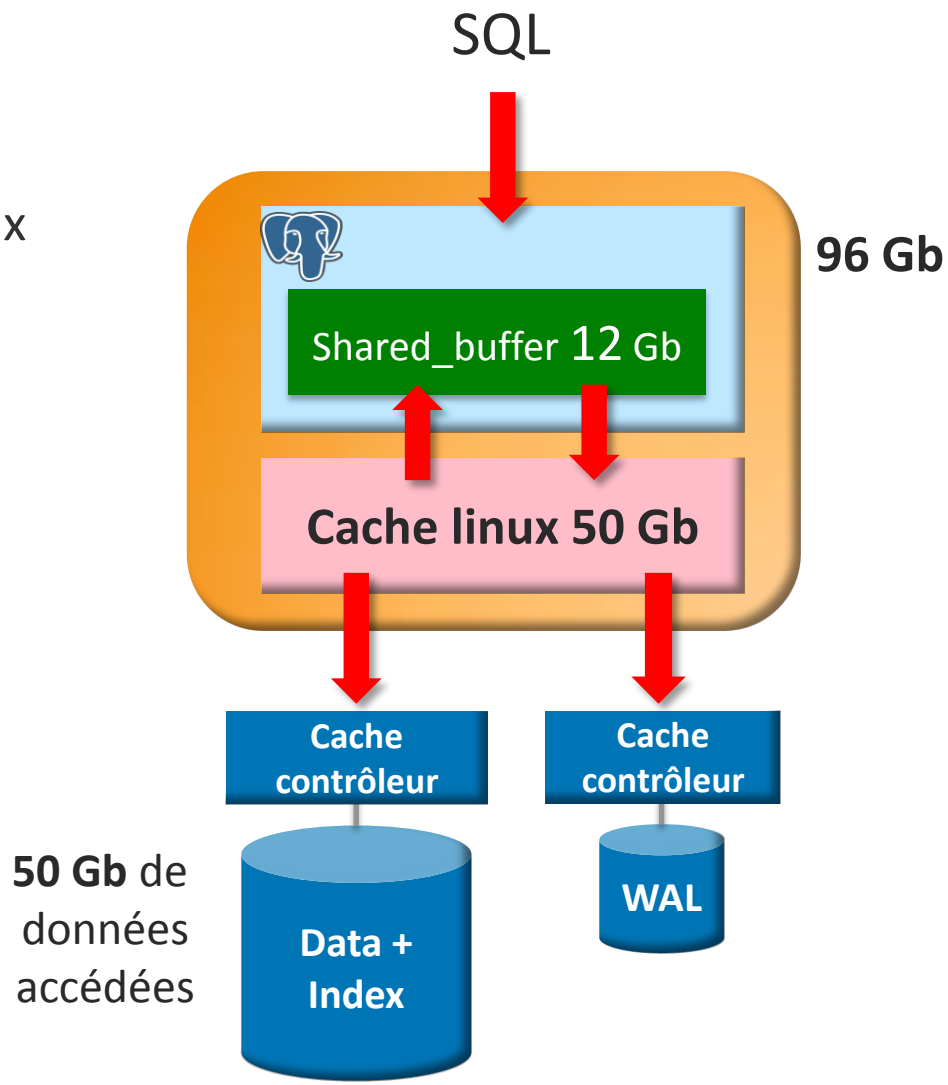
- L'activité transactionnelle seulement
- Paramétrage « type production »
- Injection avec pgbench

Moyens disponibles



Orientations prises

- Cacher les I/O en lecture
  - Données accédées < cache linux
- Mais représentativité
  - de la structure des tables
  - du SQL
  - de la charge d'écriture
  - des hit ratios shared_buffer / cache linux



Modélisation

- Stat des serveurs d'application de production
  - Cible de charge transactionnelle : 3300 Tx / sec
  - Dont 0-35ms = 70% ; 35-350ms = 28% ; >350ms = 2%
- Analyse log E-Maj après 1 journée d'enregistrement
  - 11,2% des transactions font des mises à jour
- Pour profiler les requêtes, pg_stat_statements

Extension pg_stat_statements (rappel)

- Statistiques agrégées en mémoire sur les requêtes exécutées
- Configuration
  - shared_preload_libraries = 'pg_stat_statements‘
  - pg_stat_statements.max = 10000 (def=1000)
  - .track = top, .track_utility = on, .track_save = on (=def)
- Une fonction pg_stat_statements_reset()
- Une vue pg_stat_statements avec 18 colonnes, dont
  - Query
  - Calls
  - Total_time
  - Rows
  - Shared_blks_hit
  - Shared_blks_read
  - Shared_blks_dirtied
  - …

Profil des requêtes

- pg_stat_statements pendant 1 journée en production
- Caractéristiques du SQL
  - Mono table, toutes les colonnes lues ou écrites
  - Accès sur clé primaire complète ou partielle
- Analyse sur tableur
  - Nb de requêtes par transaction = 24 => cible 79.200 req / sec
  - Répartition par type de requête

Type	%	Nb lignes moy.
INSERT	1,33%	1
UPDATE	1,19%	1
DELETE	0,25%	0,8
SELECT INTO	27,15%	1
Curseur	70,08%	2

- Hit-ratio sur les shared_buffers = 99,04%

```
SELECT 1 - sum(shared_blks_read) / sum(shared_blks_hit)
  FROM pg_stat_statements
```

Profil des accès aux tables

- Croisement requêtes exécutées / caractéristiques des tables

```
SELECT relname, relpages, reltoastrelid, reltuples, relnatts,
  CASE WHEN reltuples > 0
    THEN pg_relation_size(oid)/reltuples::integer
    ELSE 0 END AS "row_size"
FROM pg_class
WHERE relkind = 'r' AND
(relname LIKE 'cd%' OR relname LIKE 'ct%' OR relname LIKE 'st%')
```

- => 5 tables type de même charge d'accès (#20%)

Tables	Nb colonnes	Long. ligne
Tbl1	6	80
Tbl2	6	100
Tbl3	8	160
Tbl4	15	250
Tbl5	34	300

- Identification de « vraies » tables correspondantes => DDL

Création et chargement des tables

- Simulation de n CAF de p dossiers
- Ex :
  - `TRUNCATE tbl1;`
  - ```
    INSERT INTO tbl1
      SELECT numorg, ndomat, 'C12', j, k, 15,
    rpad('',30,md5(random()::text))::bytea
        FROM (
          SELECT * FROM
             generate_series (1,:nborg) numorg,
             generate_series (1,:nbdos) ndomat,
             generate_series (1,15) j,
             generate_series (1,2) k
          ORDER BY 1,2,3,4
              ) AS T;
    ```
  - `ALTER TABLE tbl1 ADD PRIMARY KEY (col10, col11, col12, col13, col14, col15);`
  - `VACUUM ANALYZE VERBOSE tbl1;`
- 1h10 pour 50 Go
- Vérification de la volumétrie
  - `SELECT pg_size_pretty(pg_database_size(current_database()));`
    ```

L'outil pgbench (rappel)

- Client fourni avec postgres
- Simule l'activité de n clients simultanés
- Joue un scénario standard (type TPC-B)
  - Création et remplissage d'un jeu de tables
    pgbench –i –s <scale factor> <db> …
  - pgbench –c <nb clients> -j <nb threads> -t <nb tx>|-T <secondes> <db> …
- … ou joue un scénario spécifique
  - pgbench … -f <fichier script> <db>
  - Fichier script contient
    - BEGIN; COMMIT;
    - SELECT INSERT UPDATE DELETE
    - Variables numériques valorisées par \set et \setrandom
- Restitue le nb transactions/sec. mesurés

Conception et écriture des scénarios

- 4 scénarios représentatifs de :
  - Caractéristiques du SQL
  - Répartitions par
    - Type de transactions (lecture/écriture, petites, moyennes, grosses)
    - Verbe SQL (SELECT/INSERT/UPDATE/DELETE)
    - Nombre de lignes traitées / requête
    - Tables accédées

Scénarios	1	2	3	4
Nb SQL	6 lect.	45 lect.	388 lect.	6 lect. + 6 màj
Cible tps	2035	830	65	370

- Scénarios avec mises à jour :  à la stabilité du benchmark

Protocole de mesures

- Chargement en mémoire linux des tables 'tbl%'
  - Pour avoir des commandes du type
    ```
    find <fichiers> -exec dd if='{}' of=/dev/null \;
    ```
  - ```
    SELECT 'find ' || pg_relation_filepath(oid) || '* -exec
    dd if=''{}'' of=/dev/null \;'
      FROM pg_class WHERE relname like 'tbl%';
    ```
  - Script exécuté en 5 mn
- Chargement des shared_buffers
  - Injection des scénarios de lecture pendant 3 mn
    - validé avec pg_buffercache
- VACUUM puis CHECKPOINT
- Exécution des pgbench en parallèle
  - ```
    (pgbench -p $port $database -n -M prepared -f scenario_1.sql -c 4
    -T $duree >/tmp/pgbench1) &
    (pgbench -p $port $database -n -M prepared -f scenario_2.sql -c 8
    -T $duree >/tmp/pgbench2) &
    …
    wait
    ```
- Moyenne des tps sur 3 runs de 30 mn

Premières observations

- Scénarios de lecture seulement

- Ajout des écritures
  - vmstat
    - 2 mn OK (idle 1-2%)
    - Puis consommation cpu erratique
    - De longues périodes avec idle 60% - 80%
    - Des checkpoints de 15mn !
      >> checkpoint_timeout (5mn) * checkpoint_completion_target (0.9)
      bien que checkpoint_segments (400) non atteint

  - iostat
    - Saturation I/O sur certains disques lors des flush linux
- Modification de l'utilisation des disques
  - Partitionnement des tables avec 6 tables fille / table mère
  - Répartis sur 24 LUN = 24 FS = 24 tablespaces

Résultats

Observations

- Saturation des 8 cpu
- I/O lecture : aucune
- I/O écriture : stable ; 55 - 80 K blocs / sec

Résultats

Scénario	Nb clients	Tx / sec	Ratio / cible	Req / tx	Req / sec	Ratio / cible
1	8	2 209	1,09	6	13 254	1,09
2	18	969	1,17	45	43 605	1,17
3	12	80	1,23	388	31 040	1,23
4	12	421	1,14	12	5 052	1,14
Total	**50**	**3 679**	**1,11**		**92 951**	**1,17**

Architect of an Open World™

Merci à …

… vous !