

# Exemple d'une migration de PostgreSQL sous Linux d'un système de suivi de production

Nicolas Relange

`nrelange@lemoinetechnologies.com`

Lemoine Automation Technologies

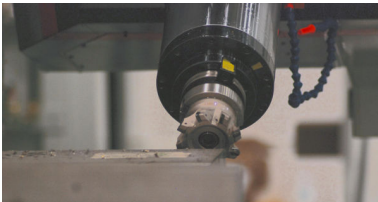
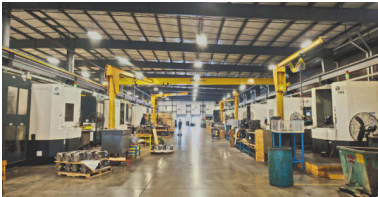
Session PostgreSQL 7, 24 septembre 2015



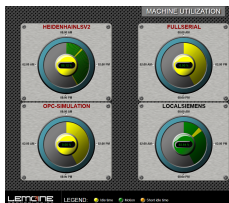
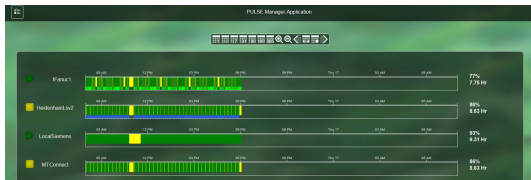
- 1 Contexte
  - Aperçu du logiciel
  - Historique
  - Enjeux de la migration
- 2 Migration sous Linux
  - Démarche
  - Hardware
  - Configuration
- 3 Retour d'expérience - Évolutions
  - Retour sur le hardware
  - Modifications applicatives
  - Maintenance



# Machines-outils à commande numérique



# Interfaces



# Historique

## Hier (2001)

- PostgreSQL 7.1
- Read-committed
- Borland C++
- Windows seulement
- Application dédiée aux moulistes

## Aujourd'hui

- PostgreSQL 9.3
- Serializable /  
Read-committed
- C#.Net / NHibernate /  
web ...
- Windows / Linux
- Nouveaux marchés  
(production de série)



# Enjeux de la migration sous Linux

## Pourquoi une migration sous Linux ?

- Meilleure réactivité (affichages temps réels)  
⇒ nouvelles contraintes de performance
- Flexibilité ⇒ base de données plus volumineuse
- Intégrité ⇒ transactions sérialisables
- Sécurité ⇒ données non corrompues et dupliquées
- Supervision ⇒ mise en place de *Nagios*
- Évolutions possibles



## Démarche choisie

- Certes des ressources sur le web
- Mais pas forcément adaptées à notre application
- Conseils sur le matériel et les configurations
- Désir de monter en compétences

⇒ Appel à une société de conseil spécialiste de PostgreSQL




# Hardware

- CPU : dual core → 8-core/16-thread
- RAM : 4 → 32 GB
- Contrôleur de disque : (1 → 2) \* *RAID1* (ou mieux, *RAID10*)  
avec Battery-Backed Unit / Flash
- Disques durs : 7200 → 15000 RPM





# Système d'exploitation

Debian stable 

Linux Logical Volume Manager (LVM)

## Configuration - disques

### Pas de vérification du système de fichier

```
tune2fs -i 0 -c 0 /dev/data1/pgdata  
tune2fs -i 0 -c 0 /dev/system/pgxlog
```

### Enlever le stockage des heures d'accès au disque

Dans /etc/fstab :

```
/dev/data1/pgdata /pgdata ext4 defaults,noatime 1 2  
/dev/system/pgxlog /pgxlog ext4 defaults,noatime 1 2
```



## Configuration - noyau

### /etc/sysctl.conf

```
vm.dirty_background_ratio = 5 (default : 10)
vm.dirty_ratio = 10 (default : 20)
vm.swappiness = 10 (default : 60)
vm.overcommit_memory = 2 (default : 0)
vm.overcommit_ratio = 80 (default : 50)
vm.zone_reclaim_mode = 0
kernel.shmmax = ...
kernel.shmall = ...(dépend de la RAM)
```



## Configuration - PostgreSQL

### postgresql.conf (extraits)

```
shared_buffers = 1/4 * RAM, max 10GB  
maintenance_work_mem = 128MB  
effective_io_concurrency = 2 (RAID1)  
wal_buffers = 16MB  
checkpoint_segments = 32  
checkpoint_completion_target = 0.9  
random_page_cost = 2  
effective_cache_size = 2/3 * RAM  
autovacuum_vacuum_cost_delay = 5ms
```



# Outils d'analyse des performances

- sar (paquet sysstat)
- vmstat
- iostat
- top
- iotop (paquet iotop)
- pgBadger
- PoWA
- OPM



## Retour sur le hardware

- Bons accès disques
- Base de données en mémoire vive
- CPU : plus de cœurs requis dans certaines configurations
- Serveurs virtualisés : bonnes performances constatées en général



# Modifications applicatives

## Constat initial

Nombreuses tâches parallèles et longues transactions

⇒ beaucoup de verrous sur les tables

Dans le cas des transactions sérialisables, rollback en cascade

- écriture de `pg_fkpart`  
([https://github.com/lemoineat/pg\\_fkpart](https://github.com/lemoineat/pg_fkpart)) ⇒  
partitionner les tables par machine
- NHibernate : patch
- Transactions plus courtes ou découpées



# Maintenance

Une maintenance hebdomadaire le dimanche matin

- VACUUM FREEZE ANALYZE
- REINDEX





# Bilan

- Gain en performance notable
- Meilleur contrôle de la base de données
- Nouveaux outils de diagnostic



# Futur

- Améliorer pgfktopart
- Nagios
- PoWA
- OPM



# Questions

Des questions ?

Merci !

