Utilisation de PostgreSQL
Cas d'utilisation et retour d'expérience

Qui sommes nous ?

- Jean-Samuel Reynaud, Administrateur systèmes et réseaux
- Société Elma, filiale technique du groupe Maximiles
- Le programme de fidélisation « online » Maximiles.com
- Des produits blancs ou dérivés

Objectif

- Présenter trois cas d'utilisation
- La supervision et le monitoring en lien avec les cas d'utilisation

Cas d'utilisation de PostgreSQL

- Cas simple: Petite base de données (5Go) et application Web
- Cas avec volume: base plus importante (150Go) et application Web
- Cas concurrentiel: base, application web et scripts métier

Cas simple: Petite base de données

- 5Go de données sur disque
- Un serveur Web avec peu de trafic
- Des accès en lectures/écritures modérés

Cas simple: l'architecture

Cas simple: Le Web

- La base est en général peu sollicitée sauf en cas d'action marketing (événementiel)
- Les serveurs web sont simples à ajouter

Cas simple: La base

- Paramétrer sa configuration PostgreSQL pour refléter sa configuration Web
- Le cache disque est pertinent

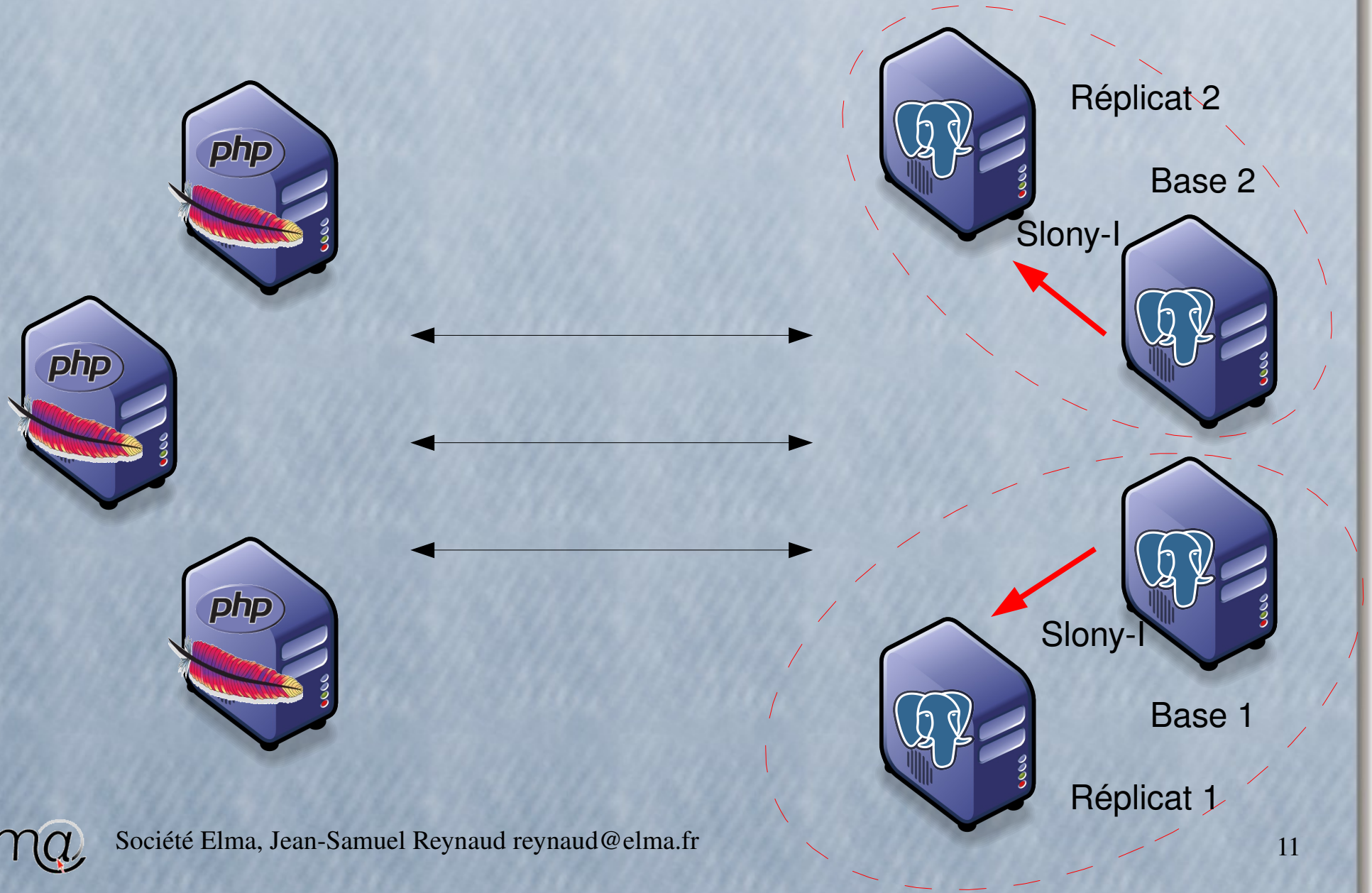
Cas simple: Ne pas oublier

- Montée en charge web face aux activités événementielles
- Contrôler le nombre de connexions à la base
- Les risques concurrentiels qui peuvent apparaître en condition stressée
- Anticiper/surveiller le volume de la base

Cas avec volume: base plus importante

- 150Go de données sur disque
- Un service web (plusieurs serveurs)
- Trafic relativement important
- Plus d'écritures que de lectures (70/30)

Cas avec volume: l'architecture



Cas avec volume: le Web

- Écritures: absorber le flux
  - Peu d'optimisation de ce côté
  - Aller à l'économie « SQL », requêtes efficaces
  - Limiter leur nombre quand c'est possible
- Lecture: Limiter l'impact (sur la base maître)
  - Déporter les lectures sur des réplicats
  - Cacher les données (memcache et/ou session)

Cas avec volume: mesurer l'activité

- Capter le flow des requêtes (pgspy)
- Comprendre/analyser le résultat
- Permet de comprendre les erreurs applicatives:
  - Volume de requêtes anormal
  - Requêtes répétées
- Diagnostique type « pompier »

Cas avec volume: la base

- Optimisation hardware: Raid 10
- Utilisation de réplicat slave pour les lectures (slony)
- L'application ne peut plus bénéficier d'un cache disque total
- Les contraintes référentielles doivent être utilisées judicieusement

Cas avec volume: Répliquer avec Slony

- Un système de réplication de PostgreSQL
- Basé sur des triggers
- Mono-Master
- Multi-Slave
- Réplication asynchrone
- Contrôle de la synchronisation obligatoire en production

Cas avec volume: le hardware

- Des disques rapides (RAID 10)
- De la ram pour le cache
- Du CPU si il y a des calculs

Cas avec volume: Ne pas oublier

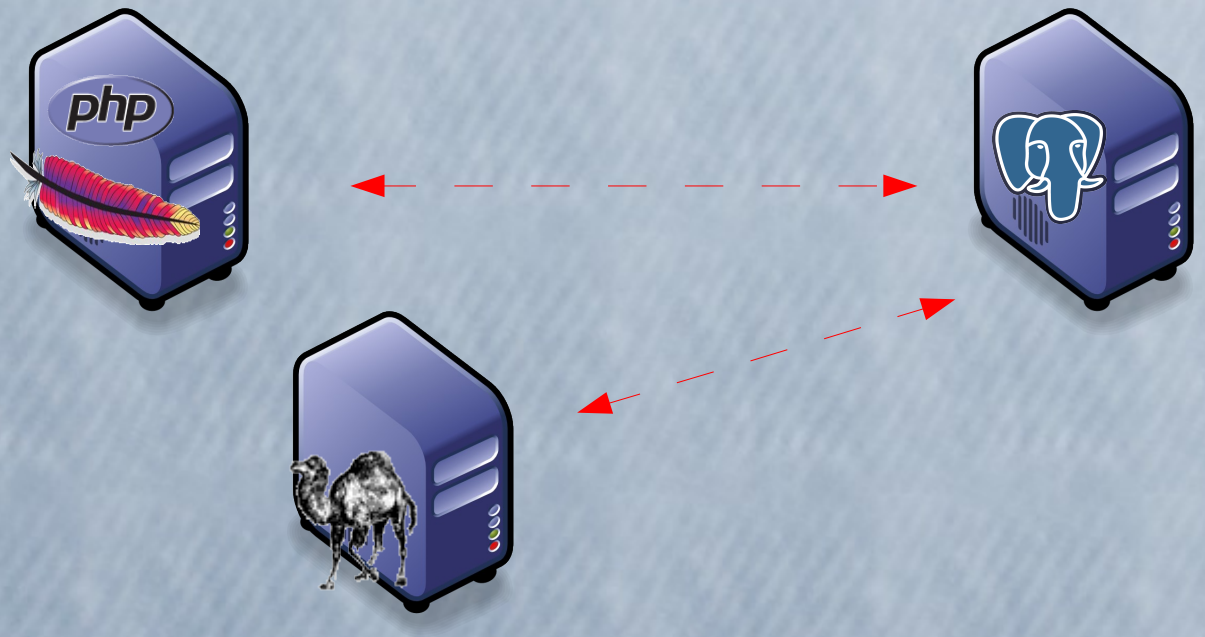
- Tout n'est plus faisable dans la base
- L'approche monolithique n'est plus possible
- Toutes les situations de concurrence se produisent
- On ne peux pas faire grossir le hardware indéfiniment
- Le monitoring est indispensable

Cas concurrentiel

- Un serveur Web (peu de trafic)
- Une base (volume limité)
- Des scripts métiers

=> beaucoup de concurrence dans l'ensemble

Cas concurrentiel: l'architecture



Cas concurrentiel: la base

- La configuration par défaut de PostgreSQL est suffisante
- Le moteur fonctionne très bien

Cependant:

- Monitorer la base pour contrôler l'accumulation de verrous
- Planifier les opérations de maintenance aux heures appropriées (vacuum full)

Cas concurrentiel: L'application

- Opérations métiers:
  - Obtenir la ressource nécessaire mais pas plus
  - Faire ses modifications
  - Valider (commit)
- Deux approches:
  - Isolation des transactions (sérialisation)
  - Méthode du « verrouillage »

Cas concurrentiel: Isolation des transactions

- Considérations à prendre en compte
  - Comprendre la logique
  - Faire face aux échecs de la sérialisation
    - Gérer dans l'application – éviter les plantages
    - Rejouer si nécessaire
  - Limites de la lecture et des tests d'existence:
    - On vérifie qu'une donnée n'existe pas
    - On l'insère si nécessaire

Cas concurrentiel: le verrouillage

- Considérations à prendre en compte
  - Choisir son niveau
  - Comprendre les impacts dans la base
    - Sur les contraintes référentielles
    - Sur les indexes
  - Comprendre les impacts dans l'application
    - Sur les autres process métier
    - Sur le web

Cas concurrentiel: Ne pas oublier

- Éviter les deadlocks
  - Vérifier l'ordre des verrouillages
  - Respecter la logique métier
- Utiliser les verrous « informatif » (pg_advisory_lock)
  - Dans l'application
  - Attachés à sa logique métier

Il faut gérer tout ça !

- Superviser
  - Remonter les alertes en cas de problèmes
  - Garder un historique des incidents
- Monitorer
  - Piloter son activité
  - Anticiper les problèmes
  - Planifier les investissements

Supervision: Nagios

- L'outil de référence
- Beaucoup de plugins de contrôle disponibles
- Contrôles:
  - Accès à la base
  - Espaces disques et autres paramètres hardware
  - Nombres de verrous
  - Réplication pas trop en retard (slony)

Nagios: les plugins

- Outils standard
  - check_pgsql: Contrôler la disponibilité de la base
  - check_disk/check_snmp: Contrôler l'espace disque
- Outils développés en interne
  - check_remote_slony.pl: Contrôler le lag slony
  - check_pg_locks.pl: Contrôler le nombre de verrous

Monitoring: Grapher quoi ?

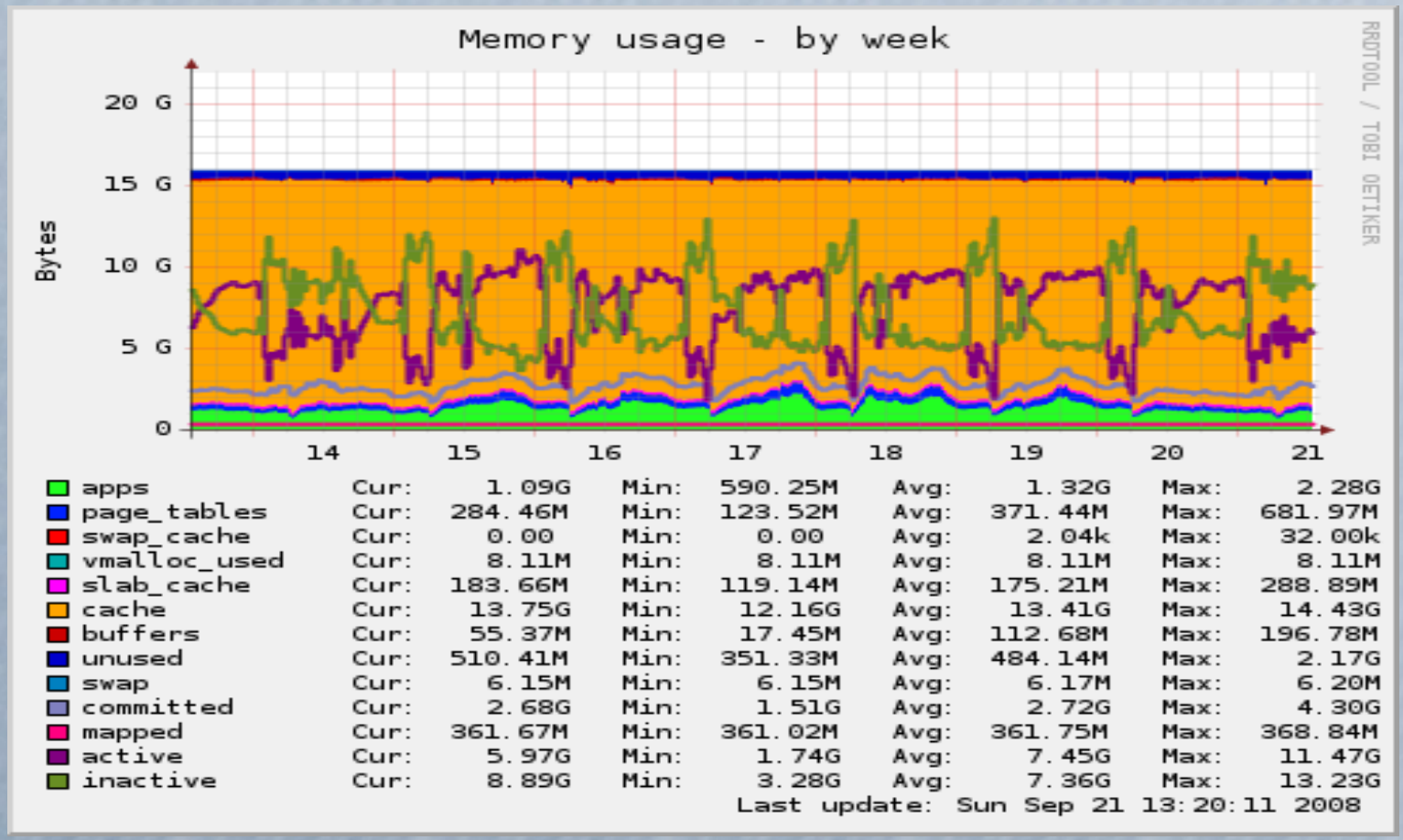
- Le hardware
- L'activité de la base:
  - commit/rollback
  - Les verrous lock
  - Requêtes (stats_row_level=on)
  - L'évolution du lag slony

Monitoring: munin

- Simple à mettre en œuvre
- De nombreux plugins disponibles:
  - postgres_commits_: les commits et les rollbacks par base
  - postgres_connections: le nombre de connexions au serveur
  - postgres_locks: le nombre de verrous
  - postgres_queries_: les types de requêtes par base

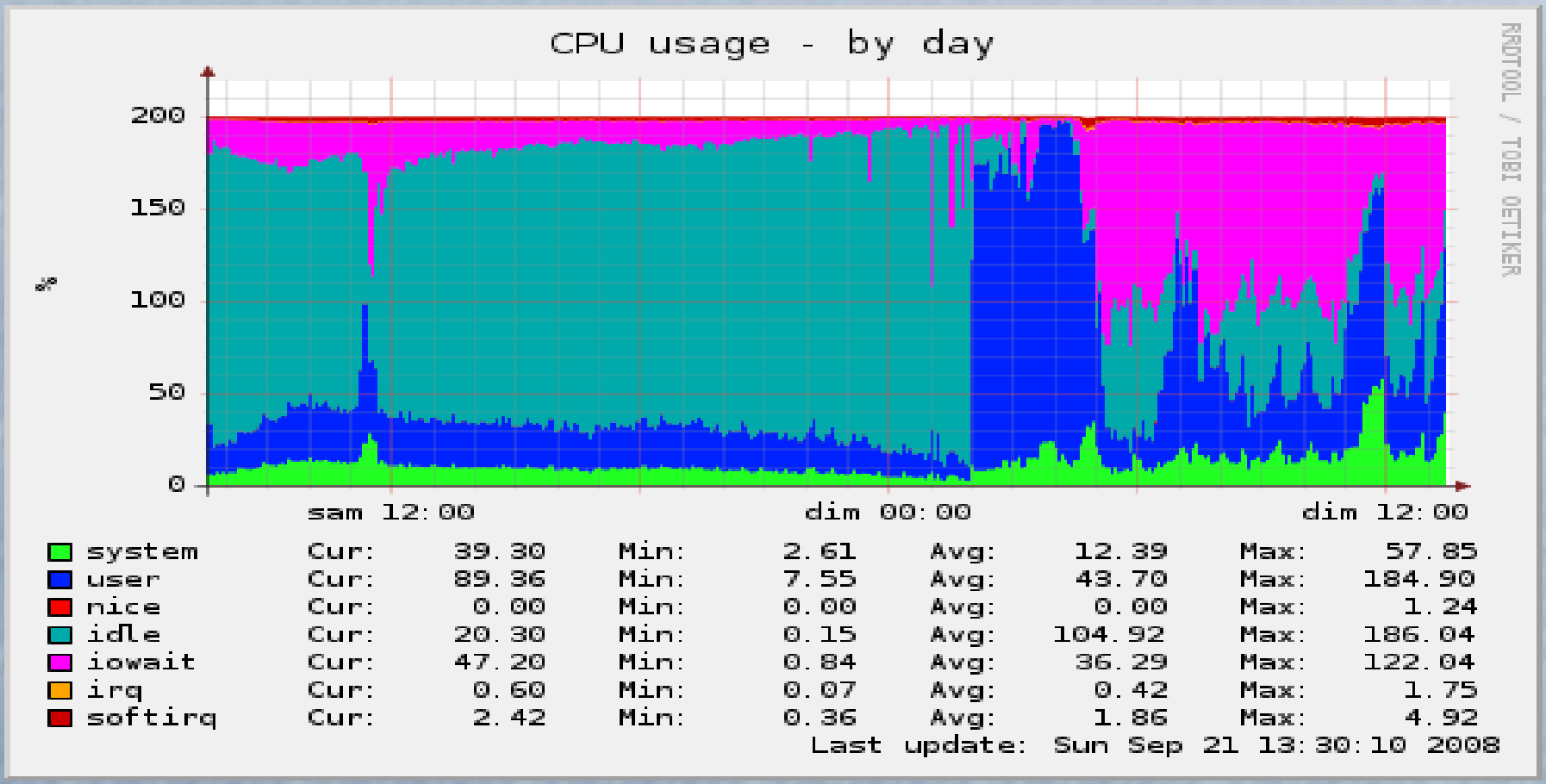
Monitoring: mesures du hardware

- Graphe de l'utilisation de la Ram



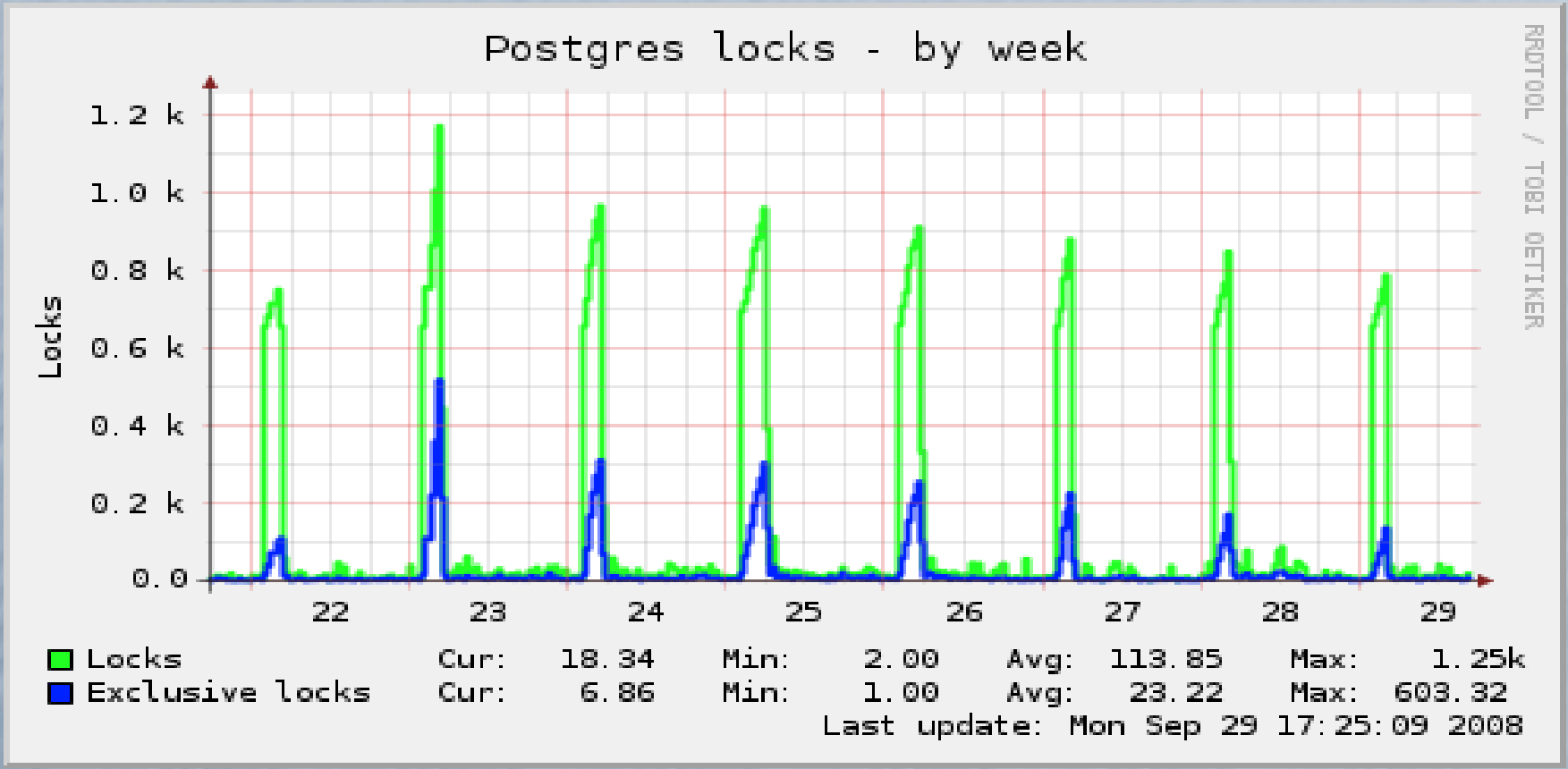
Monitoring: mesures du hardware

- Graphe de l'utilisation des CPU



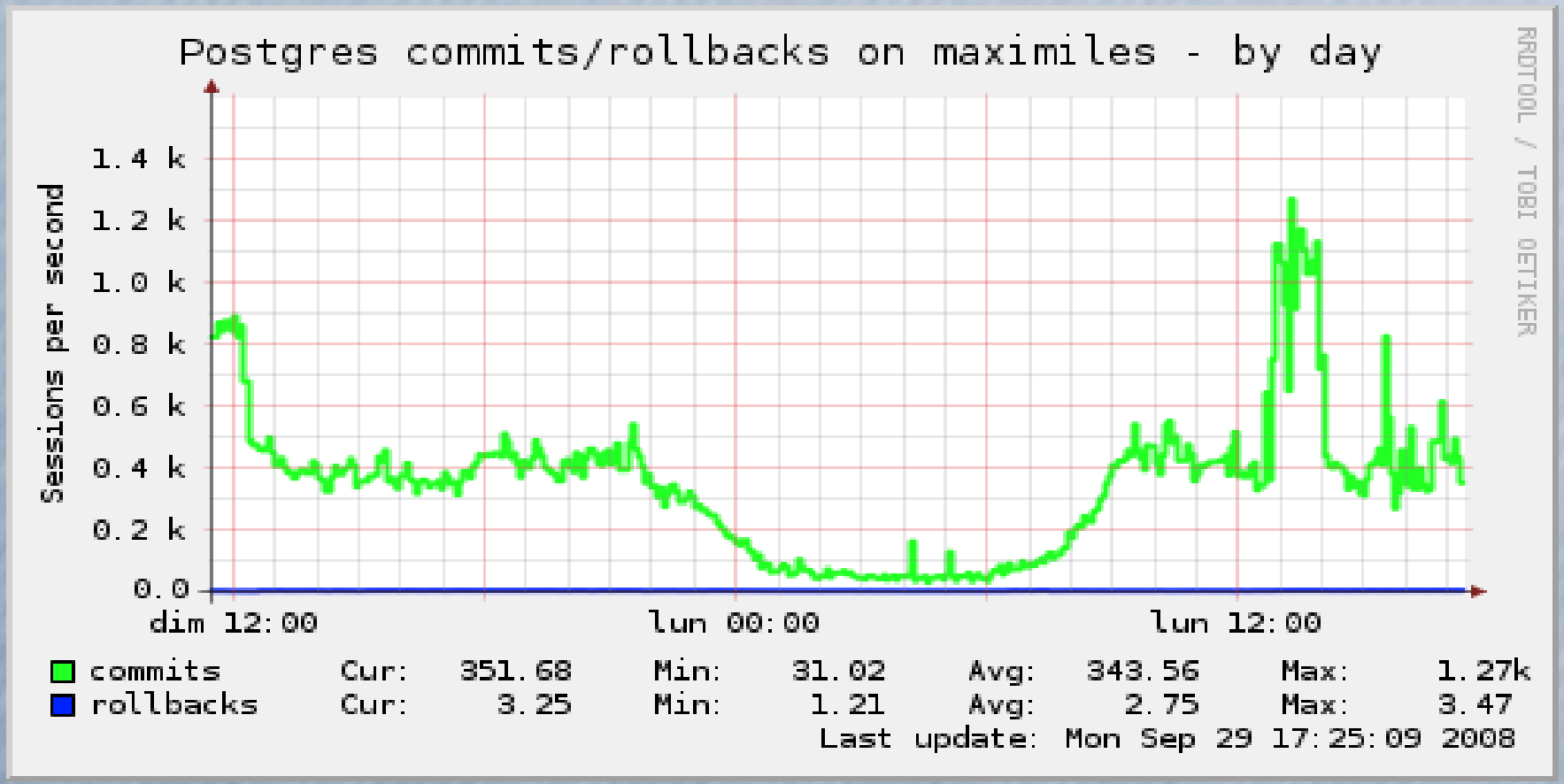
Monitoring: mesure de la base

- Les verrous



Monitoring: mesure de la base

- Les commits/rollbacks



Support de la communauté

- On détecte un problème sur un select for update et On remonte un bug sur pgsql-hackers@postgresql.org via un exemple simple
- 40 minutes après: Tom Lane détaille une analyse du bug
- 1h50 après un patch est commité. Il corrige notre problème
- Support Gold ou Platinium ?!

Conclusion

- Une base très solide
- Une communauté très active
- Beaucoup d'outils disponibles
- Questions ?