Neopod: une base de données objet distribuée

Gaël Le Mignot — Bruno Dupuis
Pilot Systems

11 juillet 2011

Plan

Introduction

ZODB

La ZODB (Zope Object Database)

- Écrite en Python
- Créée pour Zope
- Base de données Objet
- Respect des contraintes d'ACIDité

Implémentations I

FS Storage

- Un fichier en append-only
- Le process Python lit et écrit directement
- Pas de concurrence possible
- Mauvaise scalabilité

Implémentations II

ZEO

- Architecture client-server
- Le server lit et écrit dans un fichier en append-only
- Le server gère la concurrence entre les clients
- Cache sur les clients
- Le server gère les invalidations de cache
- Mauvaise scalabilité
- Aucune tolérance à la panne

Implémentations III

ZEO-Raid

- Réplication de type RAID-1
- Scalabilité limitée : chaque nœud possède l'intégralité de la base

RelStorage

- Stockage des objets sous forme de pickles dans une base SQL
- PosgreSQL et MySQL
- Réplication possible au niveau SQL
- Scalabilité limitée : chaque nœud possède l'intégralité de la base

NEO

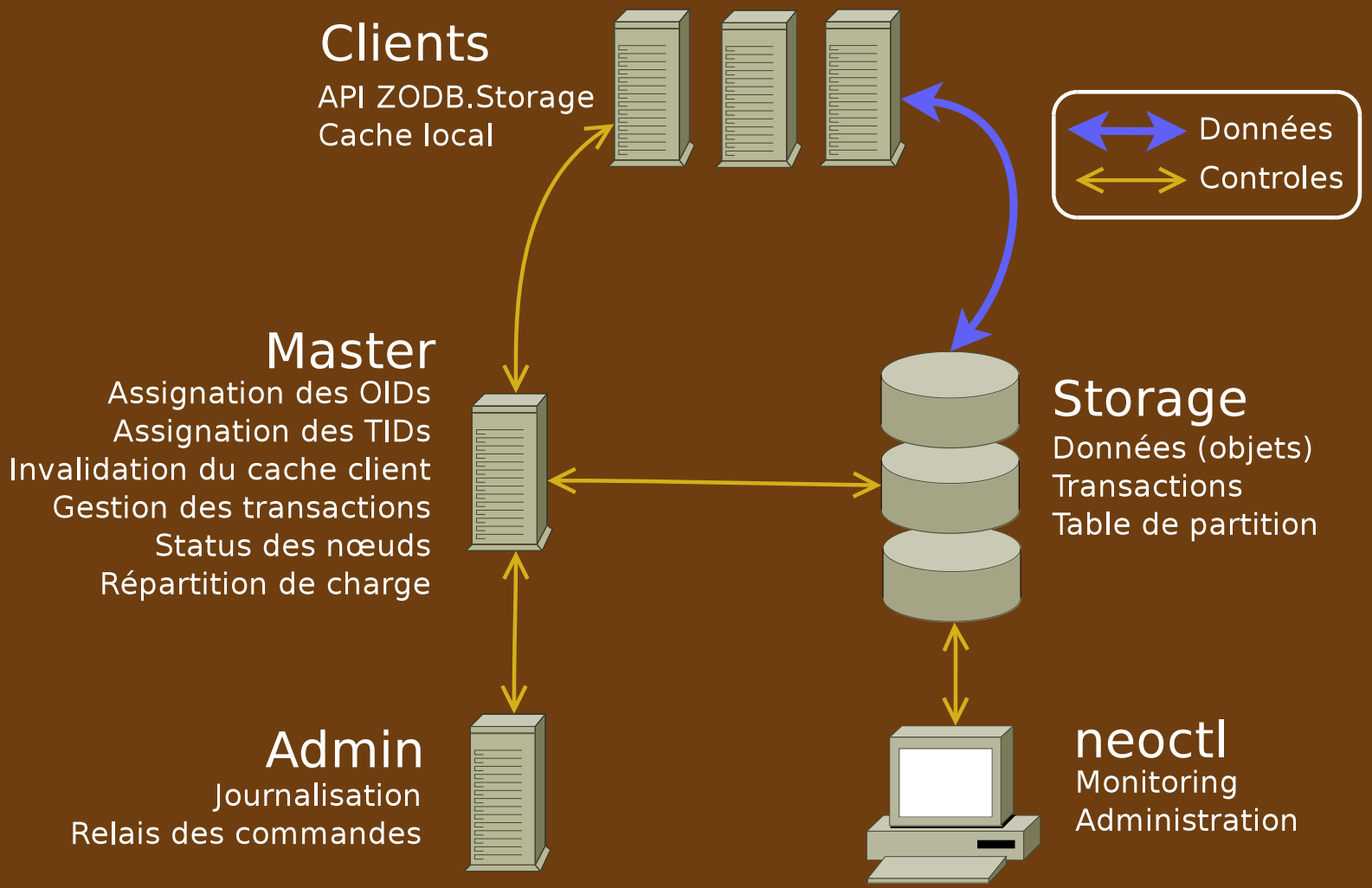
Historique

- 2005 : Démarrage du projet par Nexedi (ERP5)
- 2009 : Création du consortium System@tic
- fev. 2010 : Release
- Déploiements (ERP5, Plone3, Plone4)

System@tic

Architecture Générale

Schéma général



Réplication

Nœud maître

- Réplication simple
- Un primaire et plusieurs secondaires
- En cas de panne, élection d'un nouveau primaire

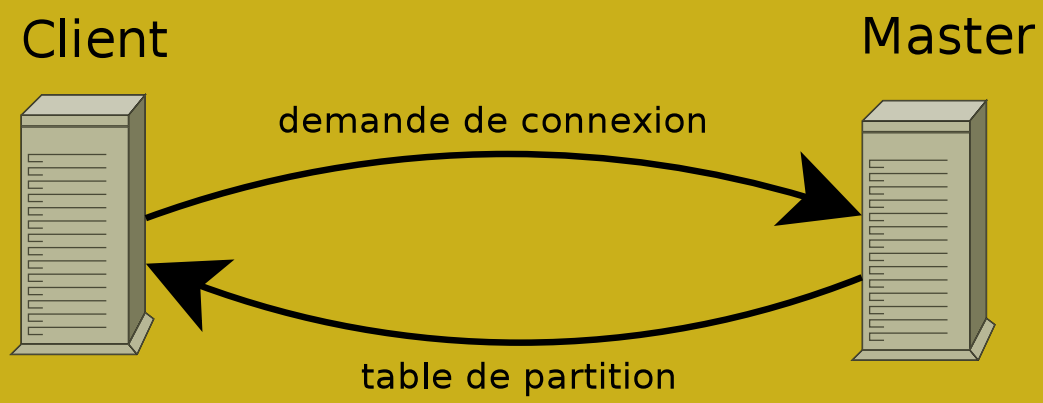
Nœud de stockage

- Choix du niveau de réplication
- Le nœud maître décide des données possédées par chaque nœud de stockage
- Le nœud maître maintient une table de partition qu'il partage avec les nœuds de stockage
- En cas de panne : le maître redistribue les données

Opérations internes

Connexion

Connexion du client



Handshake

- Le client se connecte sur le nœud maître
- Le nœud maître enregistre le client
- Le nœud maître envoie au client la table de partition actuelle

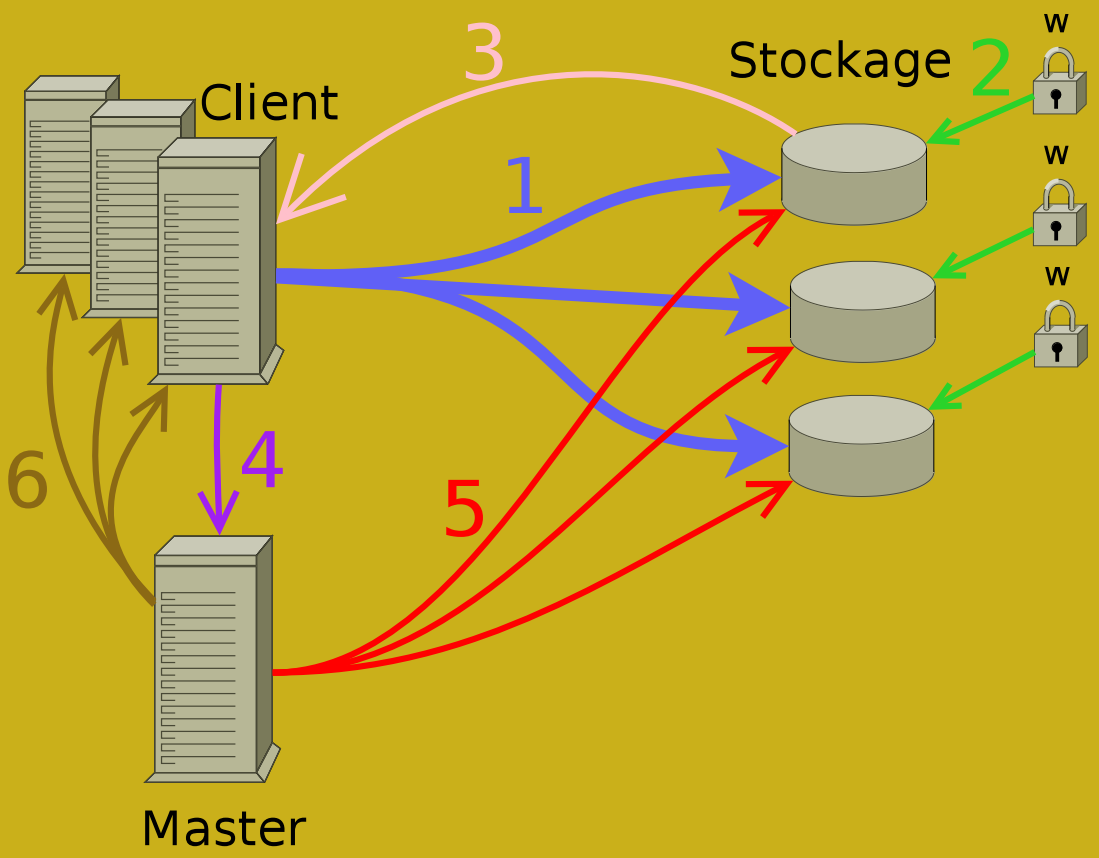
Lecture

Lecture

- Le client vérifie qu'il n'a pas l'objet en cache
- Le client trouve un nœud de Stockage qui contient la donnée
- Le client interroge directement le nœud de Stockage

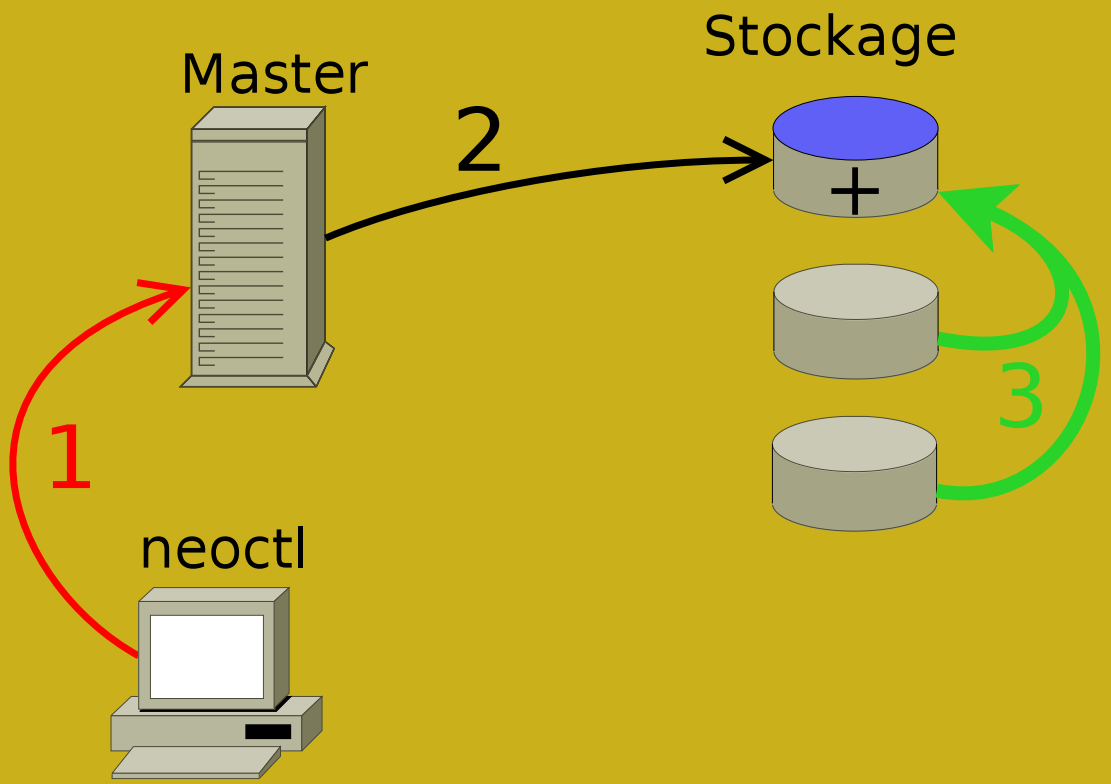
Commit I

Commit



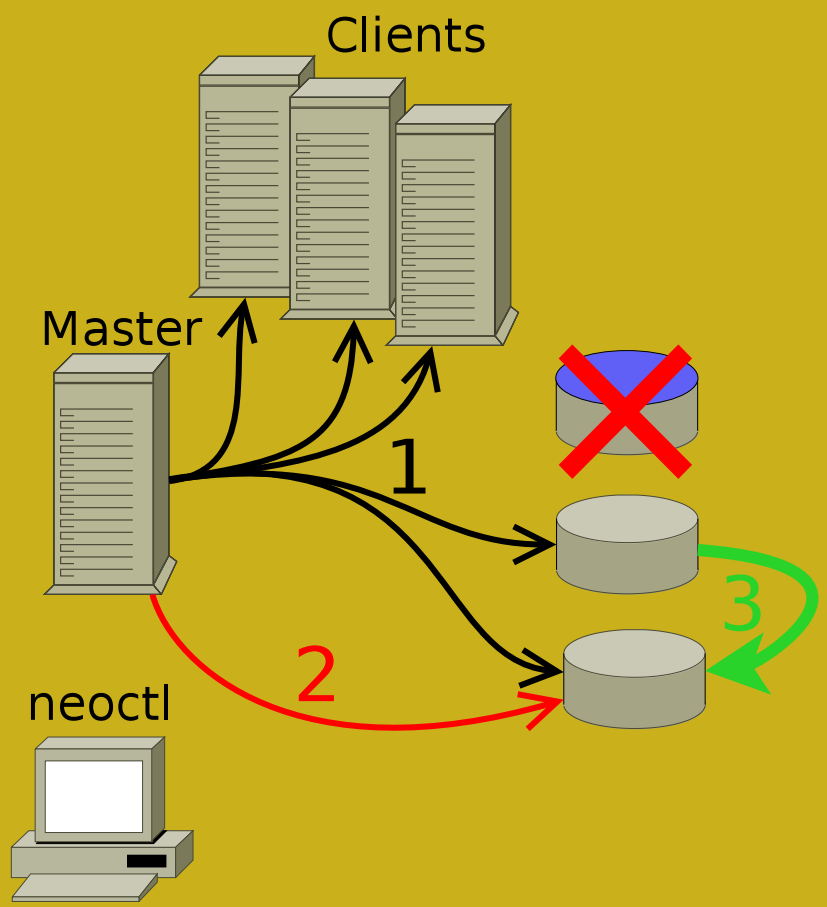
Ajout d'un nœud de stockage

Ajout d'un nœud



« Mort » d'un nœud de stockage

« Mort » d'un nœud



Objectis

Objectis

Présentation

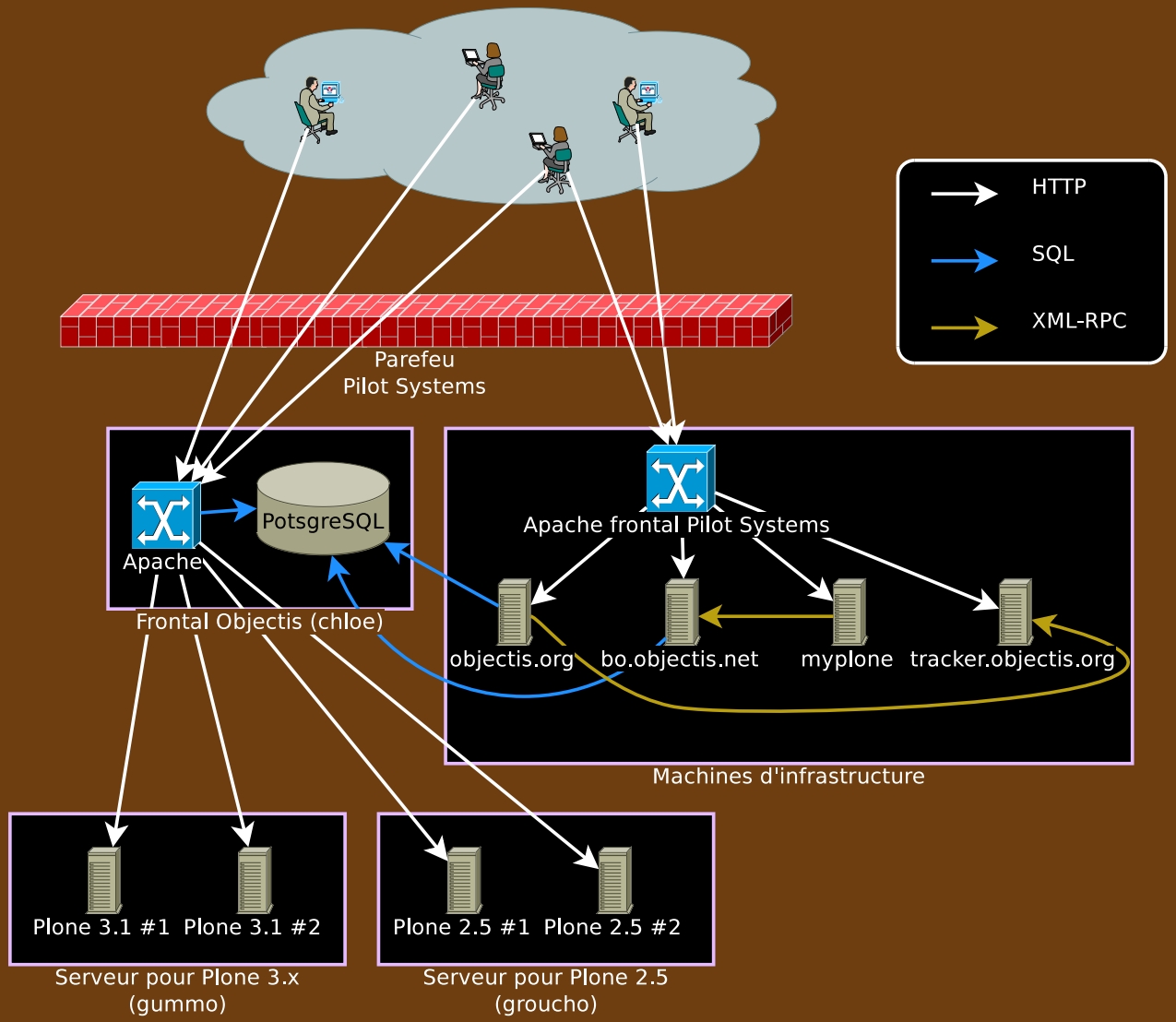
Objectis c'est...

- Une plateforme d'hébergement gratuit en Zope/Plone
- Actuellement, propose des sites en Plone 4
- Des instances contiennent des sites en Plone 1 à 3 et en Zope pur
- Nécessite un code de validation, via `myplone`

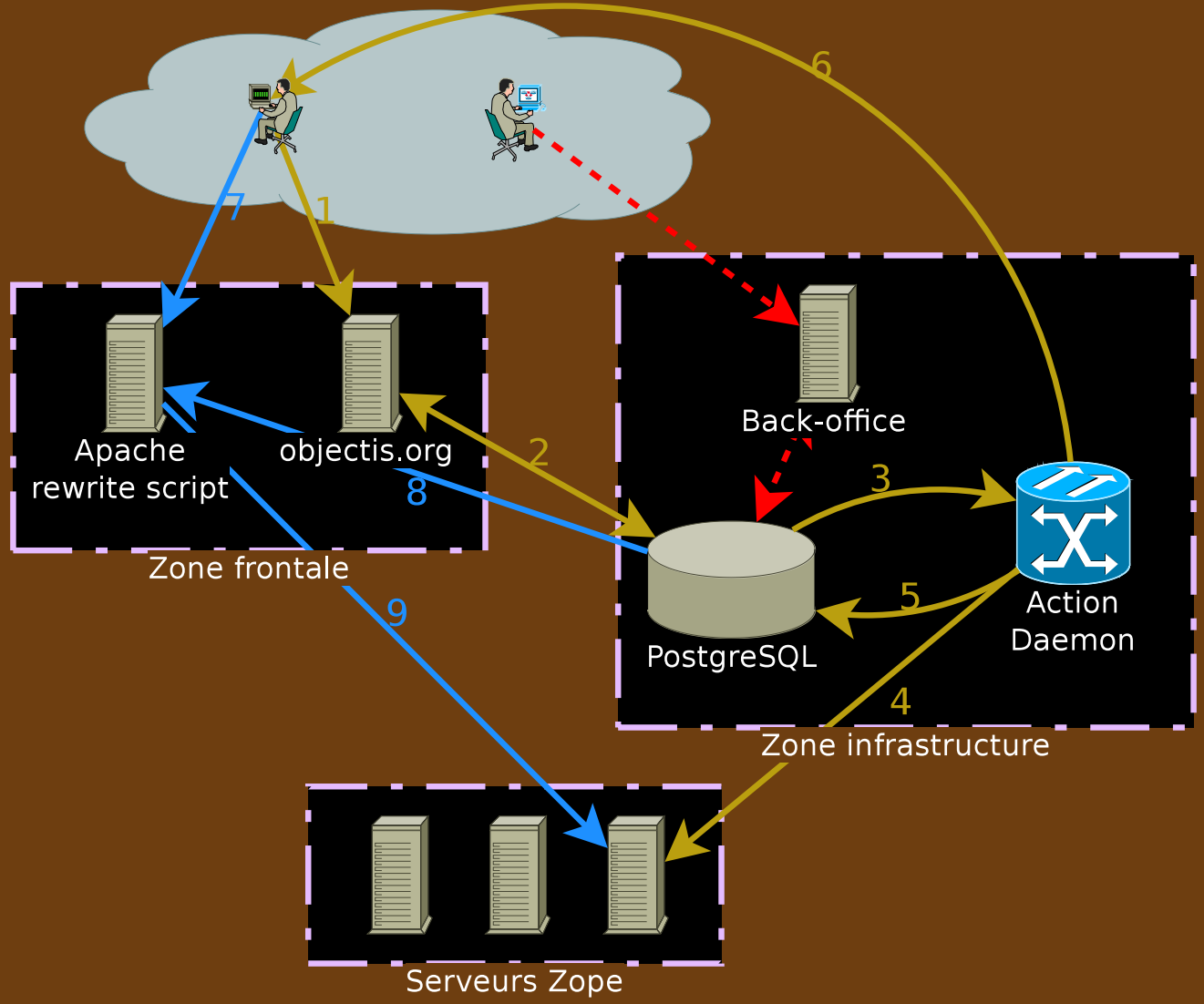
Objectis c'est aussi...

- Une communauté avec un forum et un tracker de support
- Un site traduit en 8 langues
- 18 257 sites au total

L'architecture



Création d'un site



Gestion des instances

Les slots et les instances

- Chaque instance Zope possède un type (Plone 3, Plone 4, ...)
- Sur chaque instance Zope, on défini des slots (400 à 600)
- On regroupe les instances similaires sur une VM

Utilisation des slots

- Les sites sont créés sur un Zope du bon type ayant des slots libres
- Supprimer un site libère le slot
- Quand il n'y a plus de slots, on ne peut plus créer de compte
- Il faut alors manuellement créer une autre instance

Objectis fonctionne mais...

Sur la gestion des slots

- Créer une instance Objectis est assez délicat
- À cause du GIL, un Zope ne peut utiliser plus d'un cœur
- La répartition de charge entre les instances est irrégulière
- Une fois une instance pleine, on ne peut plus grand chose pour elle

Autres problèmes

- Pas de tolérance de panne
- Instances vulnérables au DoS

Neoppod sur Objectis

Les objectifs visés

Meilleur tolérance de panne

- Une instance plantée ne doit pas couper ses sites
- À terme, on veut même pouvoir éteindre une VM sans couper le service

Meilleur gestion des instances

- Éviter le problème de l'instance ayant tous les gros sites
- Éviter de devoir configurer sans cesse des nouvelles instances
- Pouvoir ajouter des ressources supplémentaires simplement

La solution magique

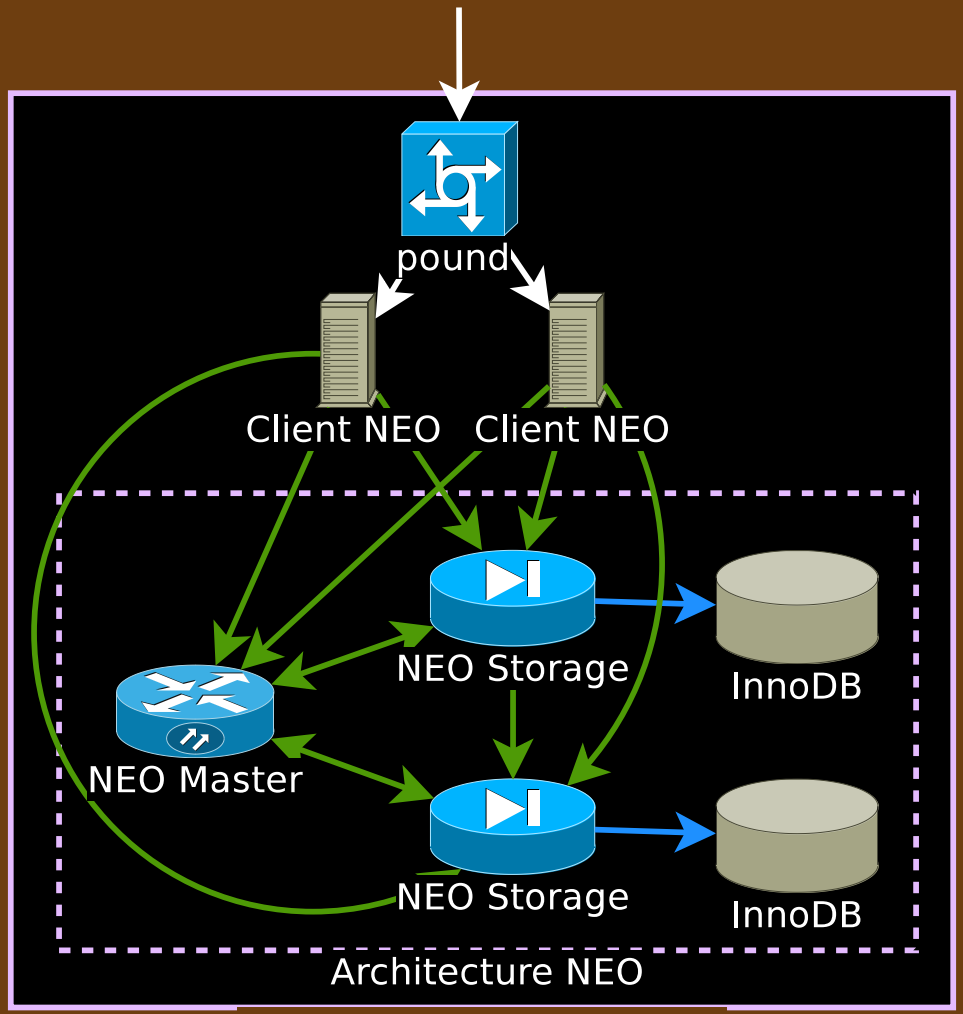
Neoppod

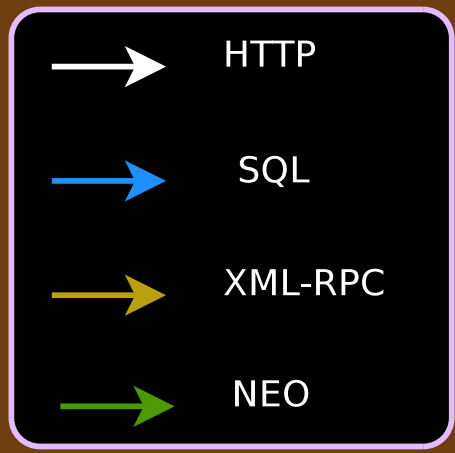
- On utilise Neoppod pour les données
- On met un cluster Neoppod par version de Plone

Effets

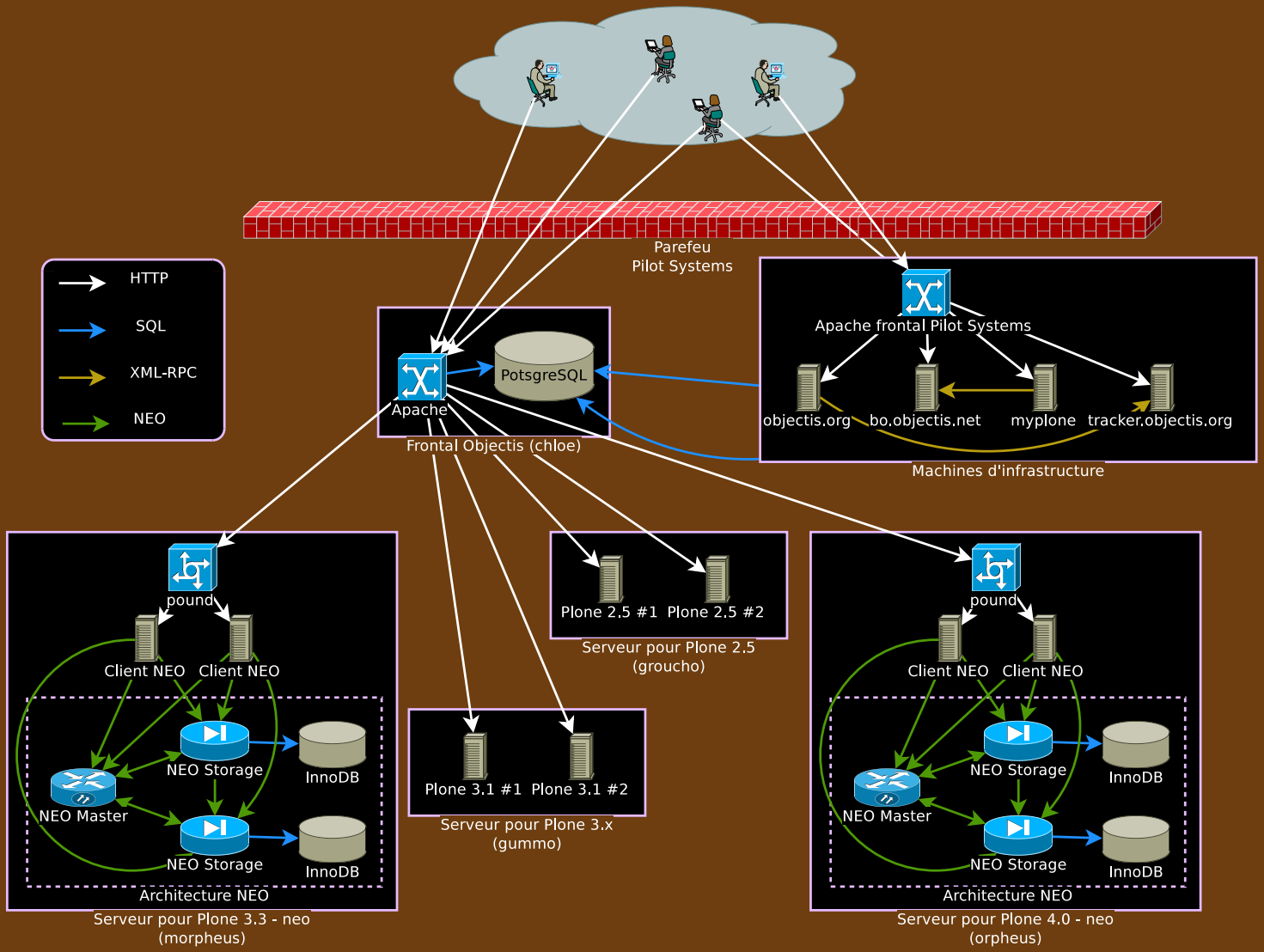
- On a de la tolérance de panne entre les instances
- On peut ajouter des instances Zope et des noeuds de stockage, à la volée
- Les ressources sont totalement mutualisées

L'architecture





L'architecture complète



Conclusion

Conclusion

Conclusion

L'état actuel

- Plone 3.3 et 4.0 sous Neoppod
- Tout fonctionne tranquillement
- Pour l'instant, une VM par version de Plone

Le futur

- Mise en place de tolérance de panne
- Répartition de charge entre les VMs
- Pour ça, il nous faut plus de monde !

En attendant

Un cadeau

- Un code offert pour 50 sites : `rmll2011-neo`

La page de pub

- Pilot Systems, société de services en logiciels libres : `http://www.pilotsystems.net`
- Slides en licence CC-By-Sa 
- `http://contributions.pilotsystems.net/`

Des questions ?