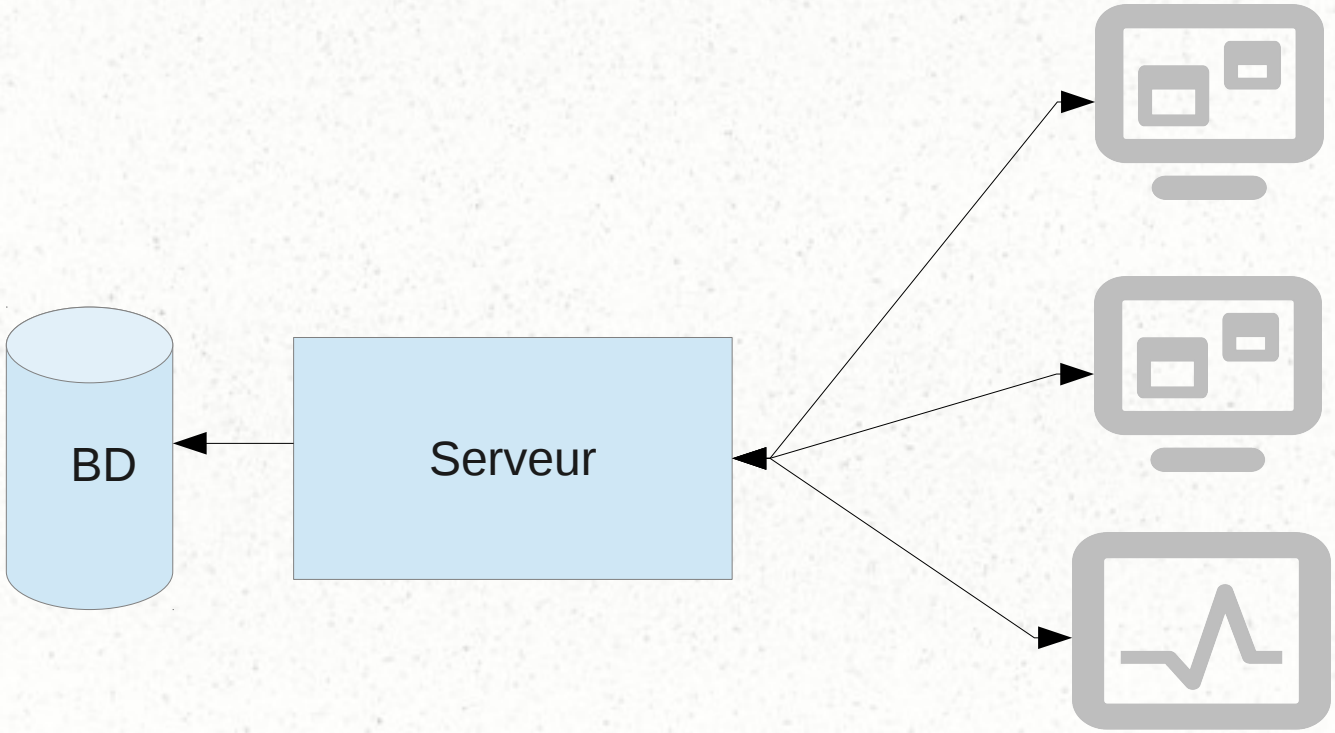
LAMP tout en python devient LTEP (Linux, Twisted, Elixir, Python)

et en plus GTK côté client

Sylvain Ferriol <s.ferriol@ipnl.in2p3.fr>

Atelier ARAMIS jeudi 18 octobre 2012

Contexte



Spécifications

- Persistance des données
- Le minimum à installer coté client
- Multi protocole
- Communication bidirectionelle
- Synchronisation des clients à différentes étapes du programme
- Interface graphique client complexe
- Développement rapide

Solution

- Un seul langage : Python
- Communication réseau: Twisted avec le protocol Perspective Broker
- Persistance des données : Elixir
- Interface graphique du client : GTK

Partie Persistance : Elixir

- Mapping objet-relationnel simple
- Respecte le design pattern Active Record
- 4 relations entre entitées : OneToMany, ManyToOne, ManyToMany, OneToOne
- Création automatique des tables et des clés étrangères

Elixir : exemple

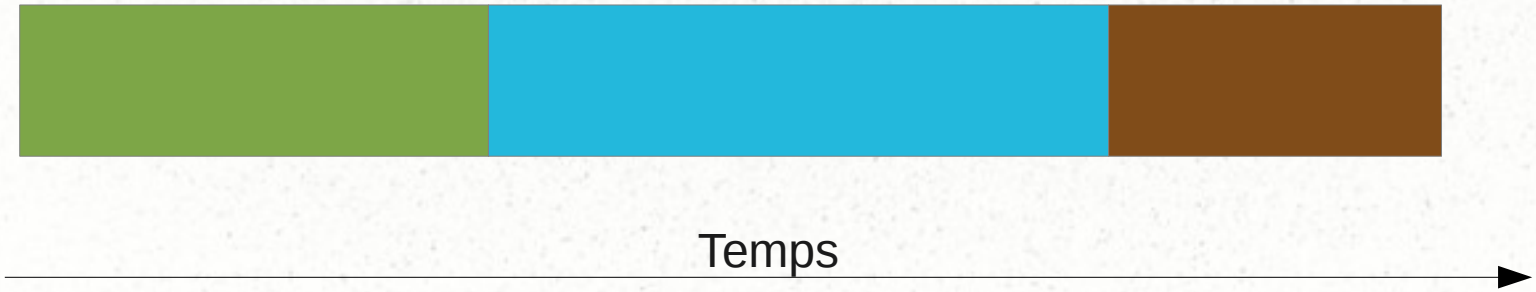
```
class Personne( Entity ):
    nom = Field( String(128) )
    age = Field( Integer )
    adresses = OneToMany( 'Adresse' )

class Adresse( Entity ):
    email = Field( Unicode(128) )
    principale = Field( Boolean )
    proprietaire = ManyToOne( 'Personne' )
```

Partie réseau : Twisted

- « An event-driven networking engine »
- Framework réseau basé sur TCP,UDP
- Implémentation des protocoles : HTTP, FTP, SSH, IRC, ….
- Programmation asynchrone s 'exécutant sur un seul thread

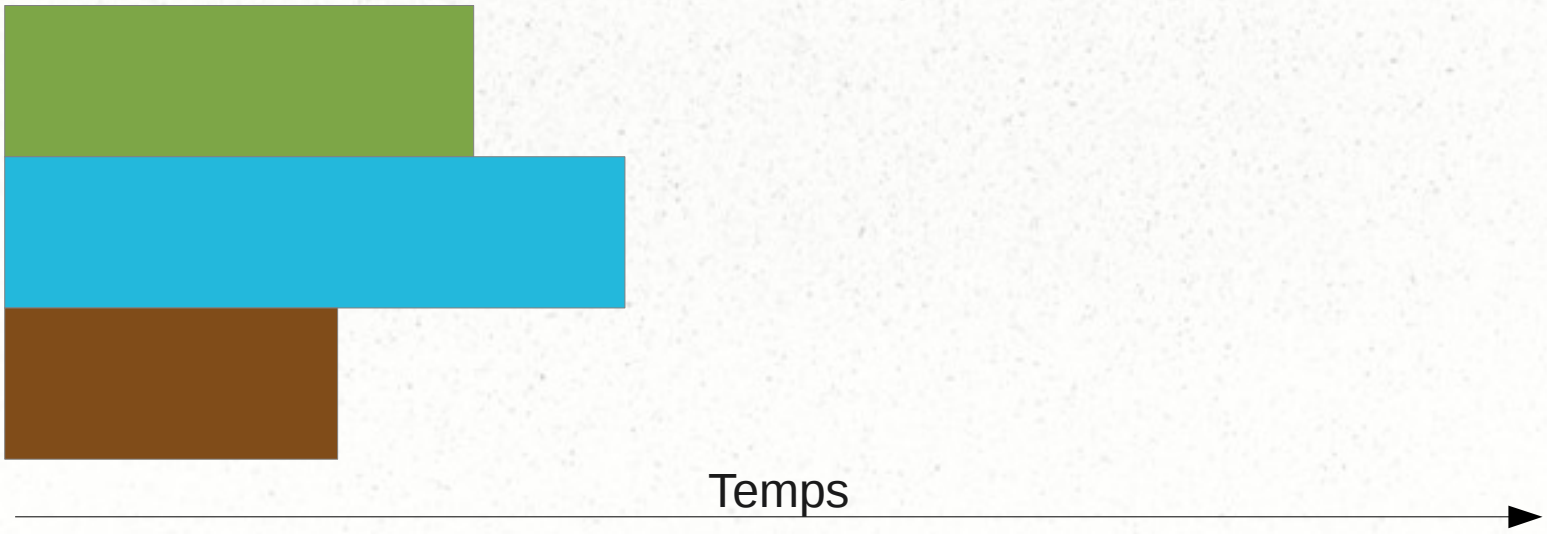
Twisted : modèle synchrone

- Une tâche après l'autre
- Un seul flux d'exécution



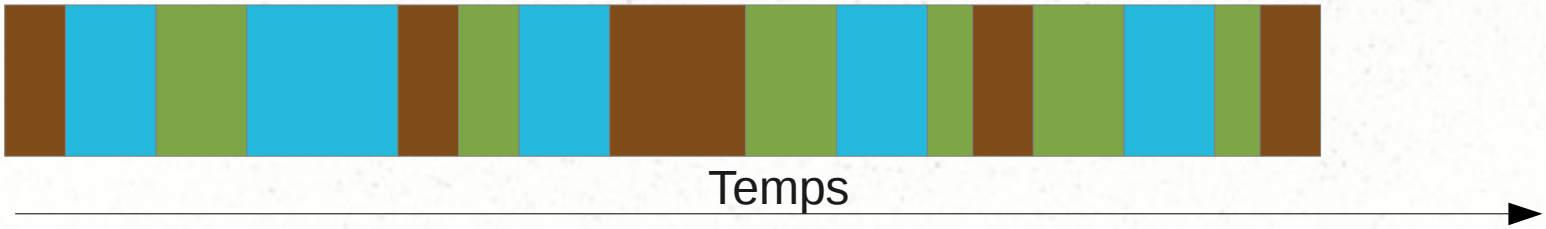
Twisted : modèle multithread

- Exécution en parallèle
- Coordination complexe des flux d'exécution
- Synchronisation difficile



Twisted : modèle asynchrone

- Les tâches sont entrelacées
- Enchaînement de petites étapes déclenchées par des événements réseaux
- Pas de parallélisme, synchronisation facile
- Programmation plus complexe car non séquentielle



Twisted : modèle asynchrone

```
def tache() :

    deferred = etape_1()
    deferred.addCallback( etape_2 )
    deferred.addCallback( etape_3 )

    return deferred
```

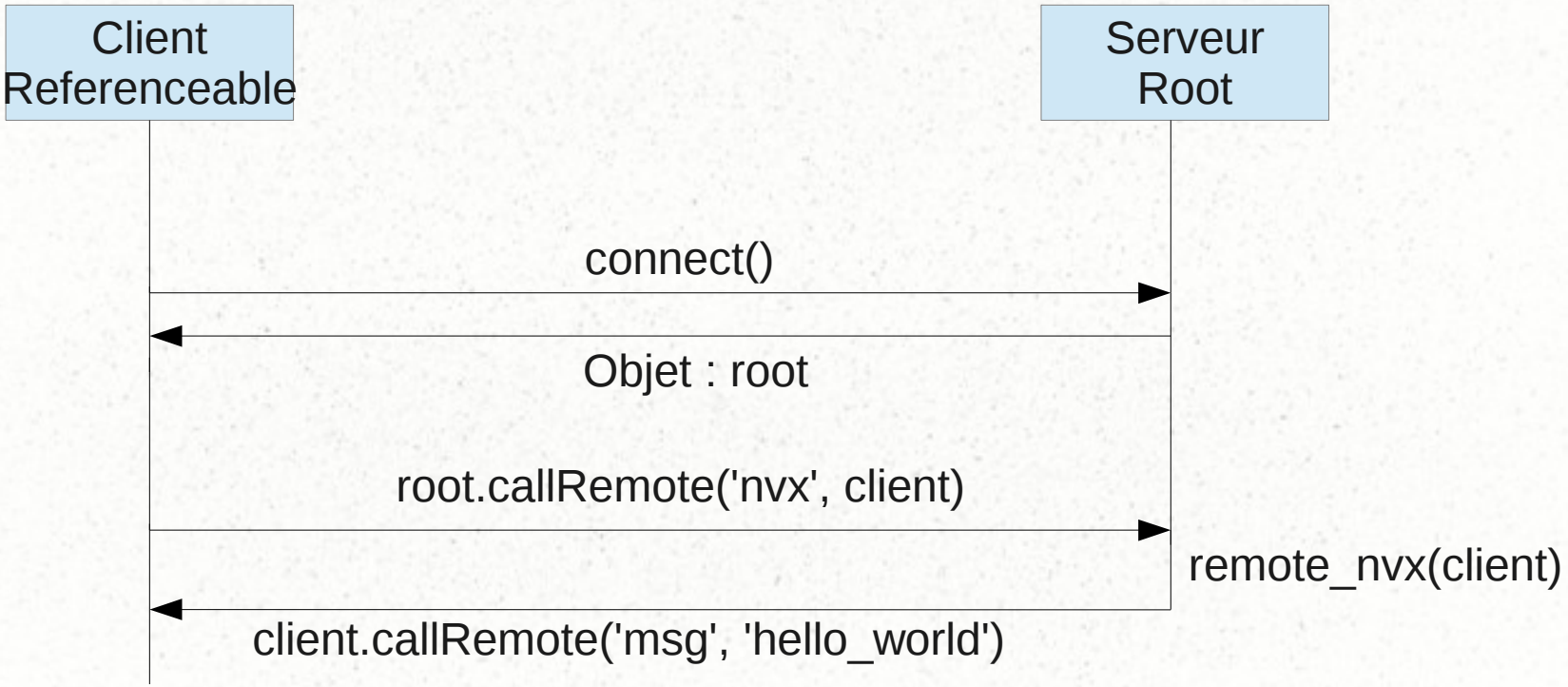
Twisted : modèle asynchrone

```
@defer.inlineCallbacks
def tache() :

    yield etape_1 ()
    yield etape_2 ()
    yield etape_3 ()

    return deferred
```

Twisted : Perspective Broker (PB)

- Protocole RPC transparant
- Appel de méthode sur des objets distants
- Les objets peuvent être transférés sur le réseau
  - Par référence (Referenceable)
  - Par copie (Copyable)
- Les méthodes accessibles par l'extérieur doivent juste avoir 'remote_' comme préfix

Twisted : Perspective Broker (PB)



Partie graphique côté client : GTK

- Design des interfaces avec GLADE stockées dans un fichier XML

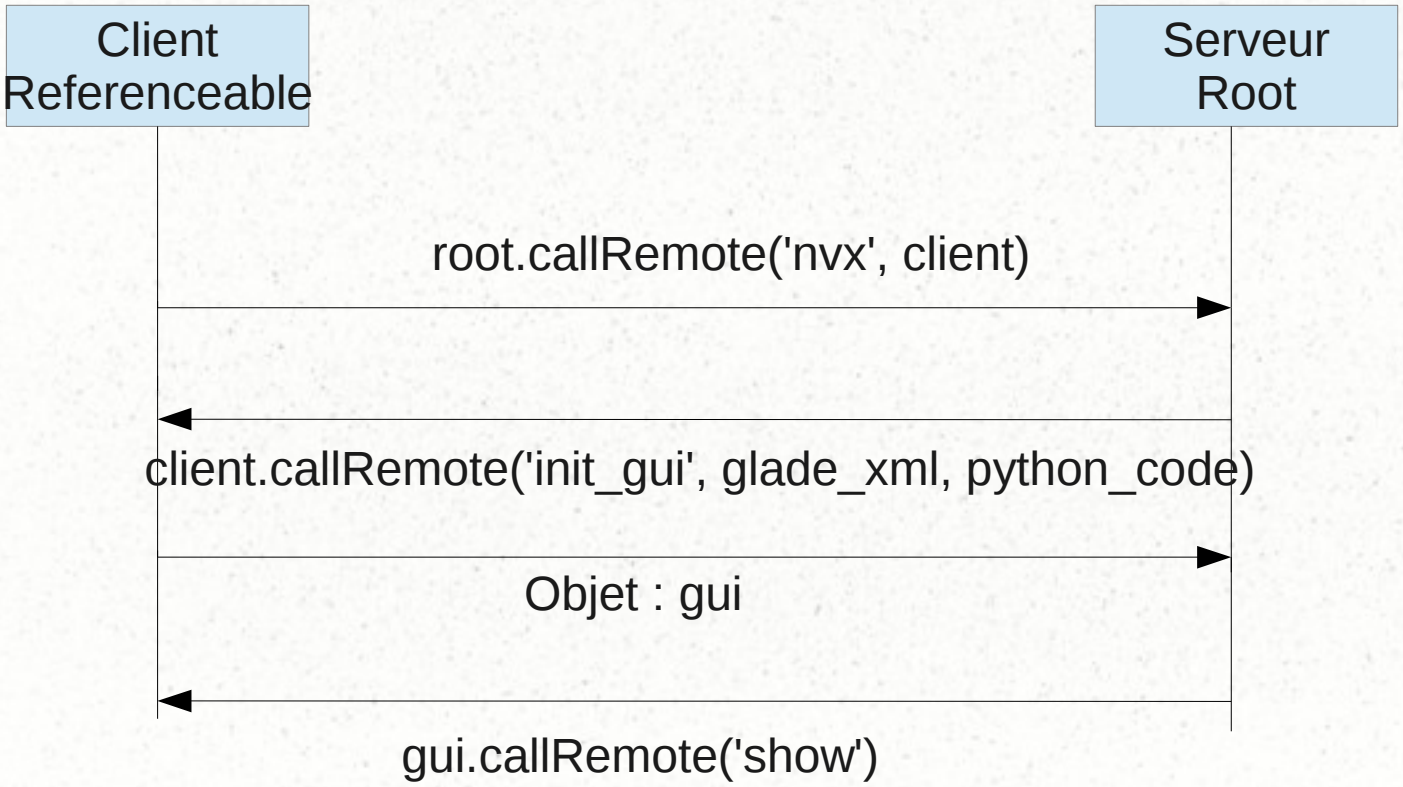
```
ui = gtk.glade.XML('mon_fichier.glade')
```

- Association automatique entre les callbacks déclarés dans l'interface, et ceux définis dans le contrôleur codé en Python

```
ui.signal_autoconnect( obj )
```

- Intégration de Twisted et GTK dans une même boucle d' événements

WEB en python ?
(html, javascript) => (glade, python)



Merci

Questions ?