Capacity Leasing Service

Juin 2013

Modules

- Inventaire

- Planning de réservation

- Labels

Inventaire

Pourquoi ?

- Pas encore de CMDB

Comment ?

- Interrogation de Nova
- Avec à Nova Client (client en python)

```
novaclient.hypervisors.list()
novaclient.hypervisors.get(host_id).__dict__
```

Fonctionnalités :

- Base de données [id | family | properties]

  id = id dans Nova

  family = md5sum des cpu_infos (pour l'instant...)

  properties = string de propriétés à charger avec json.loads()

Inventaire

Fonctionnalités (suite) :

- Mise à jour de la base à intervalle régulier
- Accessible via RPC
- Méthodes :
    - get_hosts(family=None)
    - get_host_family(host_id)
    - get_family_capacity(family)
    - get_host_details(host_id)

Planning de réservation / labels

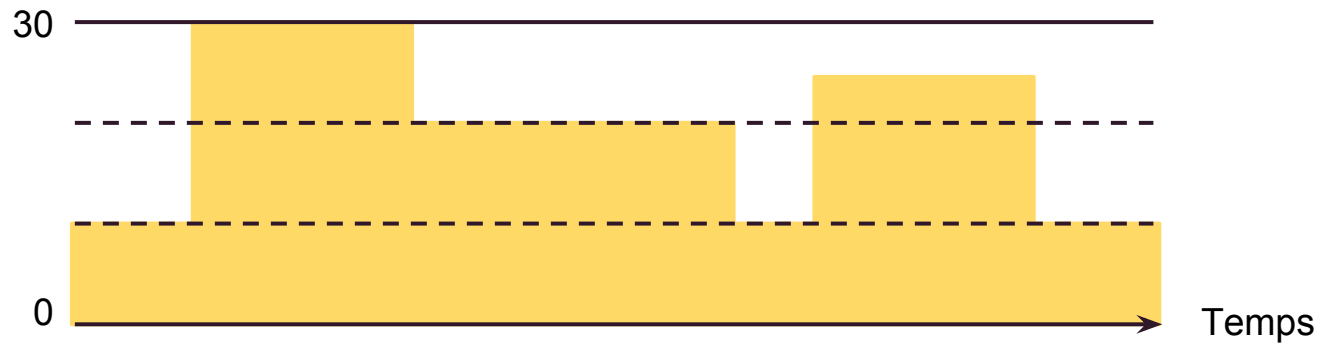
- Planning permet de connaître, pour chaque famille :
    - Périodes complètes
    - Périodes libres
    - Propositions de réservations

- Base de données
  [id | start | end | quantity | family | owner | date | confirmed | rank]

- Méthodes :
    get_label_catalog() => nom, description, type
    get_label(reservation_id, label_name=None)
    make_proposals(start_time, end_time, quantity, duration,
                   family, label_filters, max_proposals)
    confirm_proposal(reservation_id)

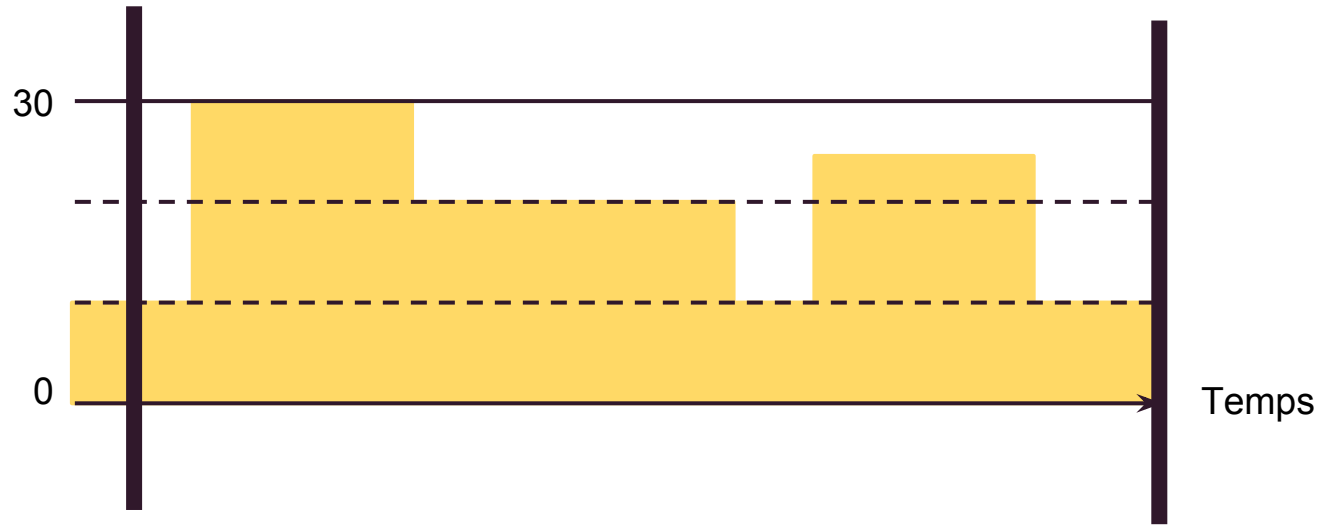
Planning de réservation / labels

Planning de la famille "x" :



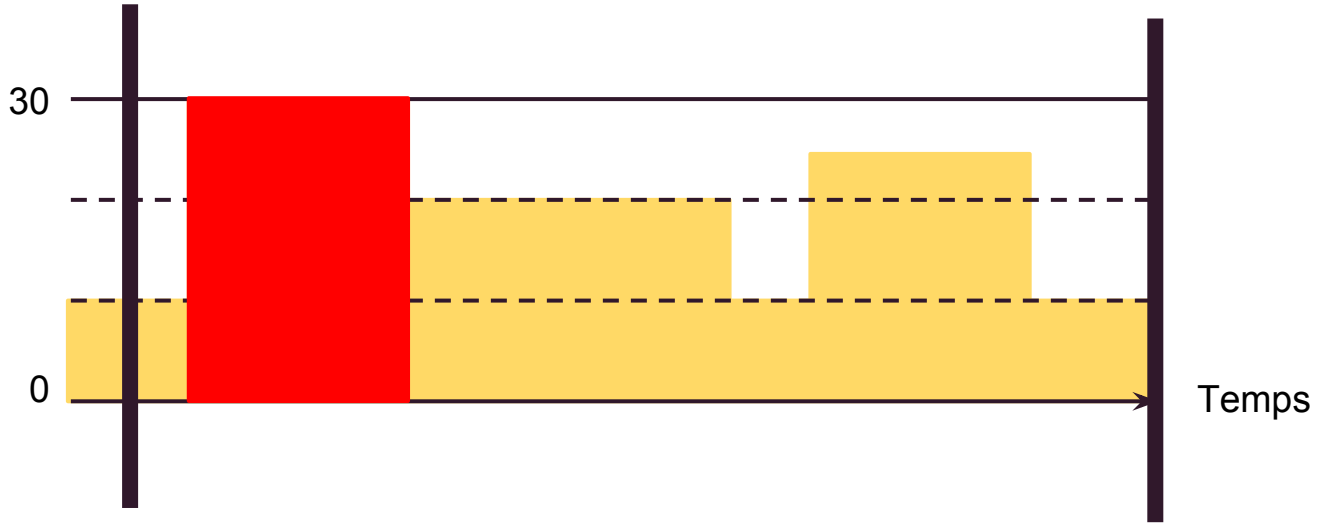
Planning de réservation / labels

Requête : 10



Planning de réservation / labels

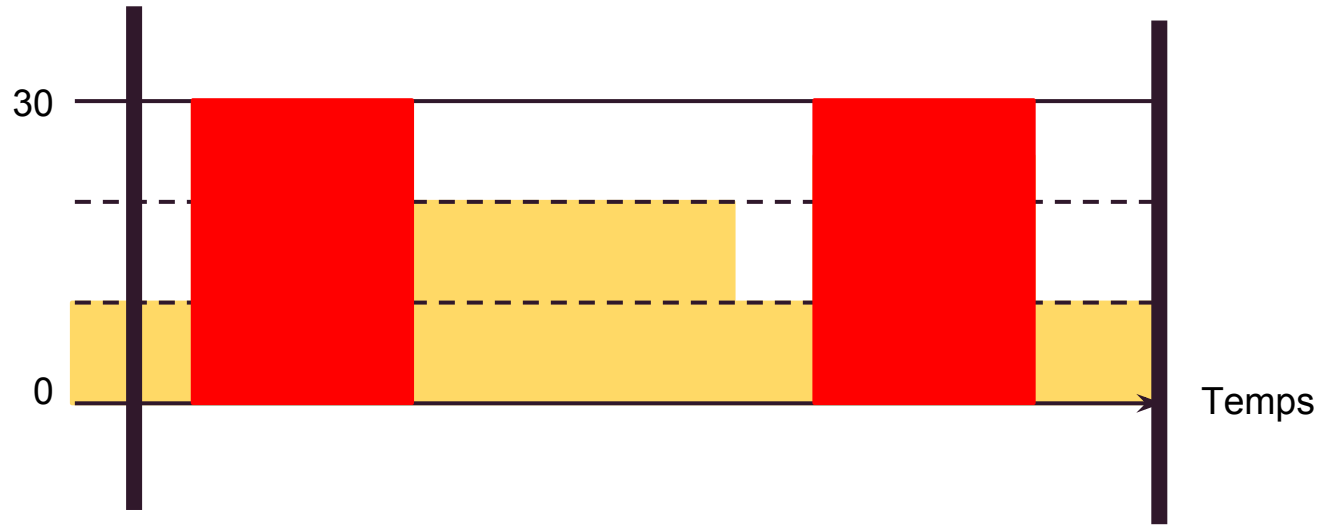
Requête : 10



Périodes pleines ?

Planning de réservation / labels

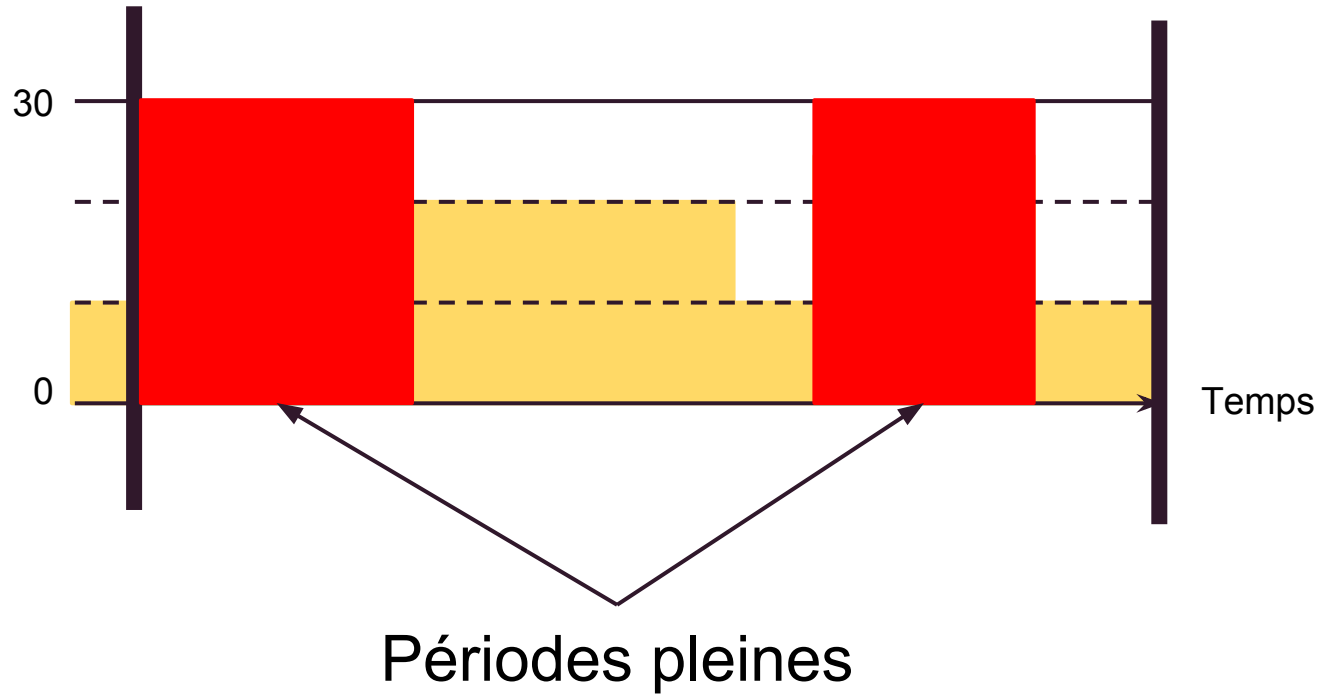
Requête : 10



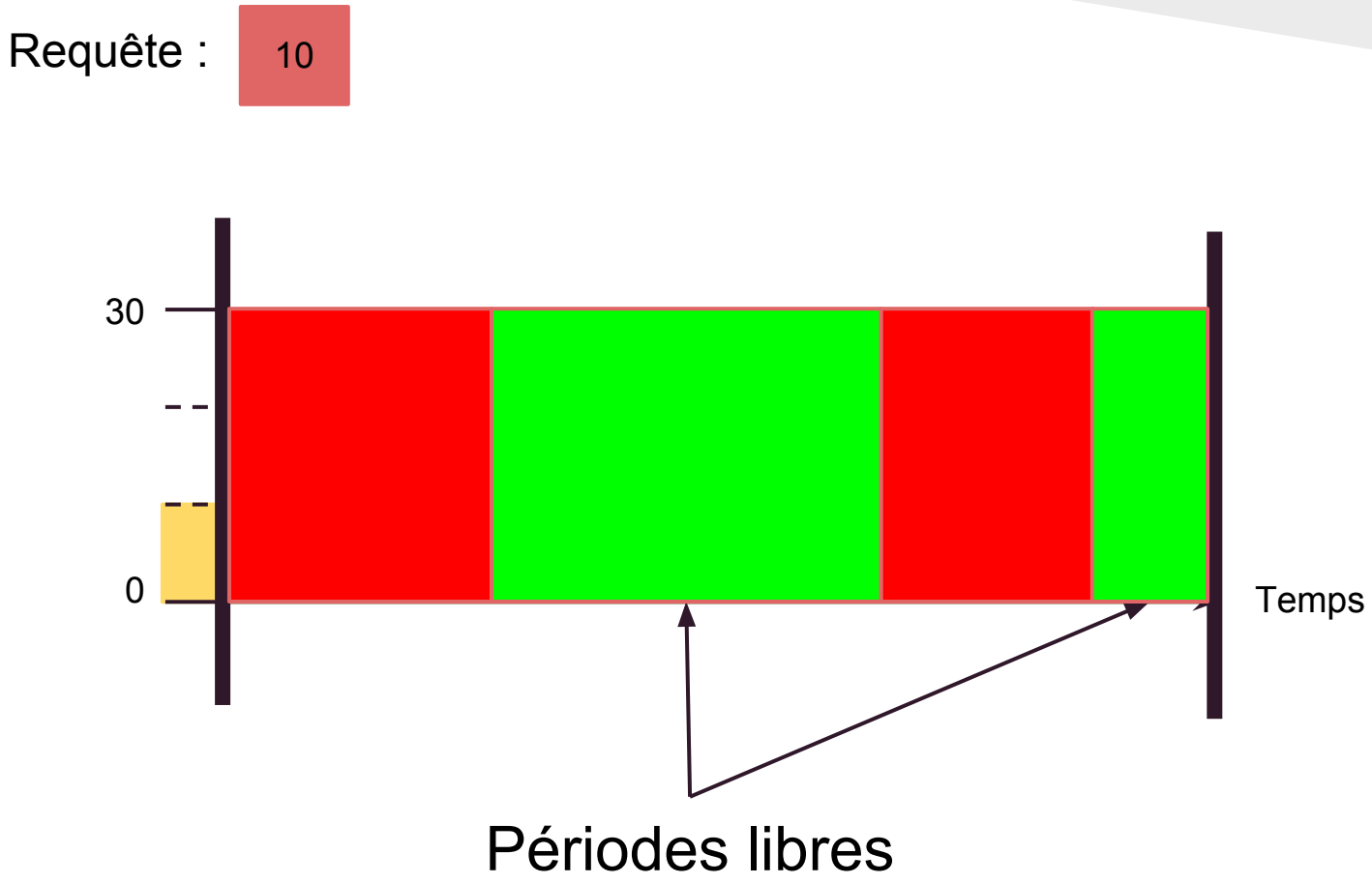
Périodes pleines ?

Planning de réservation / labels

Requête : 10

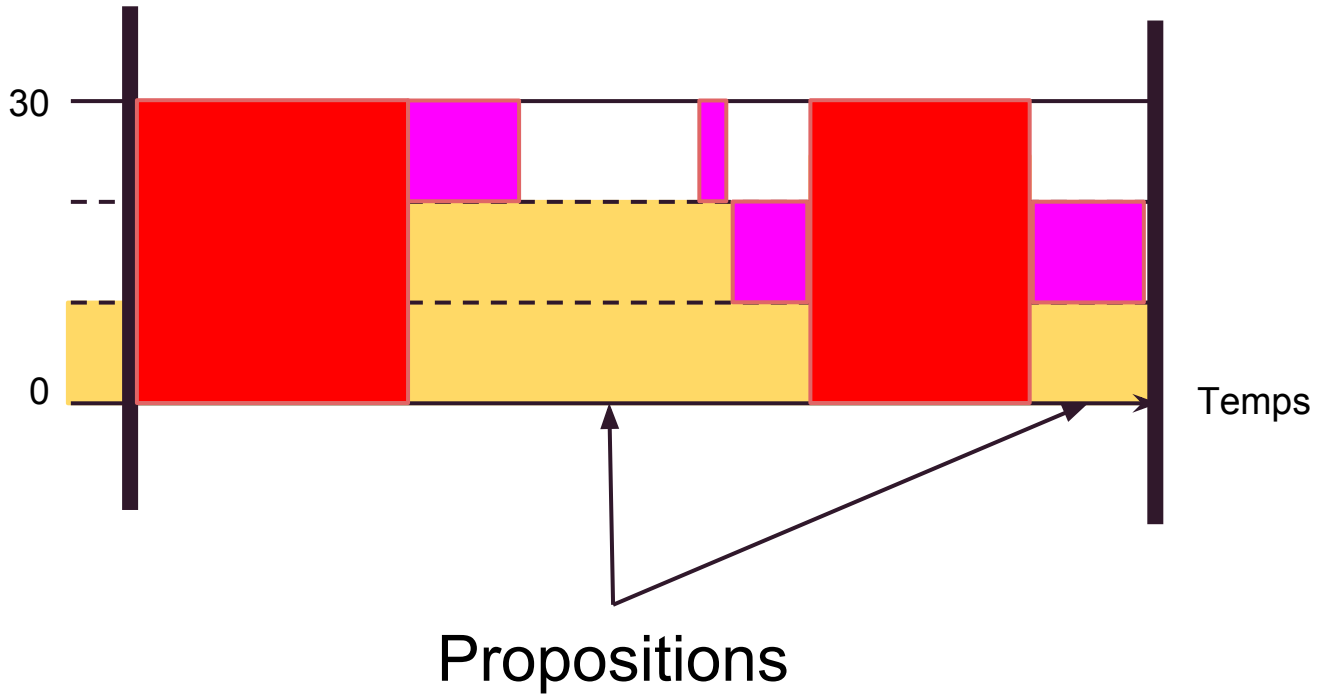


Planning de réservation / labels



Planning de réservation / labels

Requête : 10



Planning de réservation / labels

- Ressources "verrouillées" au moment de la réservation
- Délai pour confirmer la proposition (par exemple 3 min)
- Au delà du délai :
    - Possible de confirmer si personne n'a eu besoin de ces ressources
    - Impossible sinon

Labels

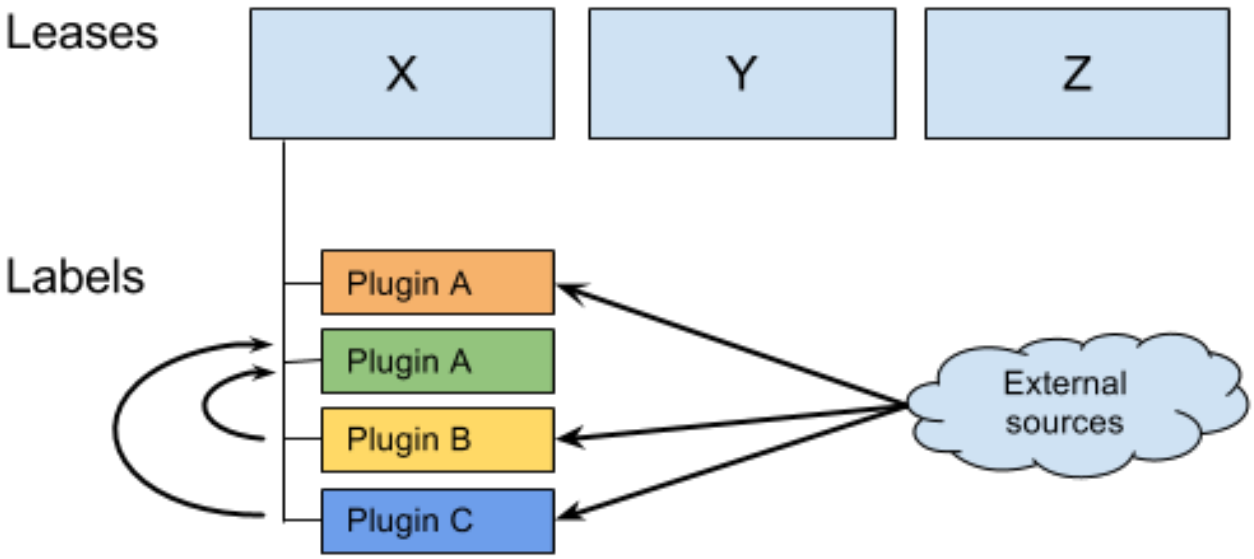
- Hiérarchie de nommage :
  - energy.min
  - energy.max
  - energy.range

- Publics ou privés (utilisation interne uniquement)

- Chaque plugin s'enregistre auprès d'un driver commun
  - Construction du catalogue de labels (nom, description, type)
    - Permet à l'utilisateur de spécifier ses prérequis
  - Dépendances entres labels, éventuellement de plugins différents
    - Exemple : watts + performances = efficacité

- Base de données [id | reservation_id | name | value | rank]
  Cache (un label est évalué une seule fois)

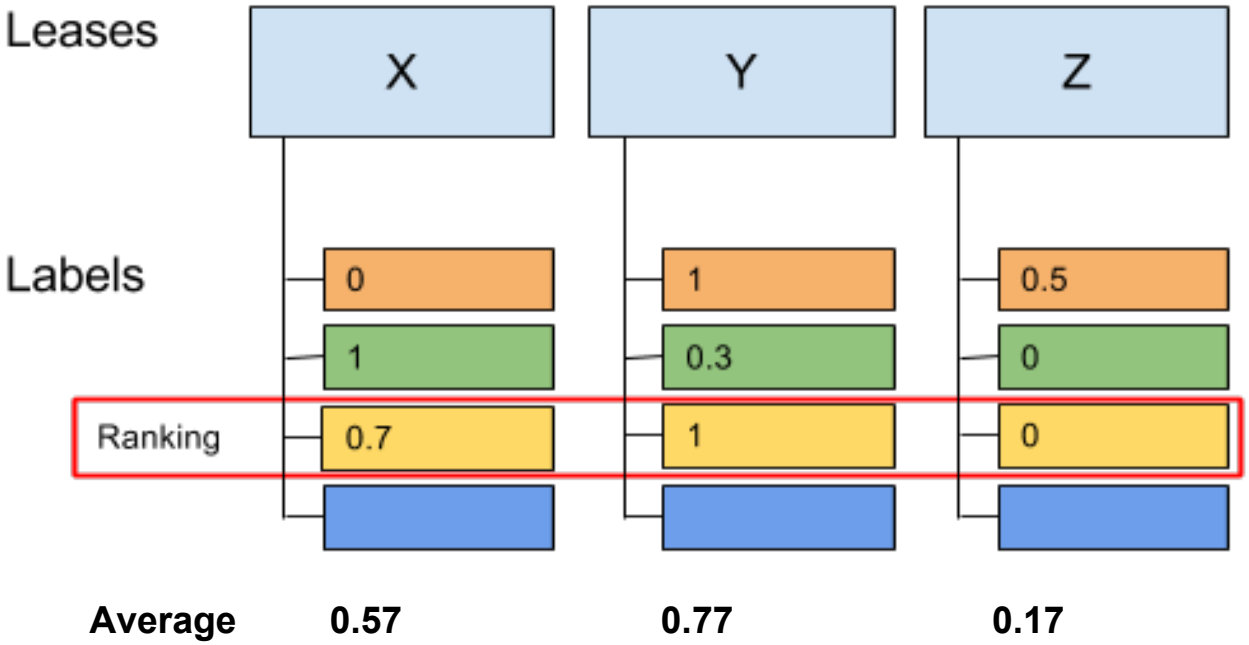
Labels


Leases
X
Y
Z
Labels
Plugin A
Plugin A
Plugin B
Plugin C
External sources

Labels

- Notation des labels entre 0 et 1 (pire au meilleur)
- Chaque plugin est responsable de la notation de ses labels :
    - Car il sait ce qui est bien et mal
    - Notation proportionnelle si possible (valeurs étirées entre 0 et 1)
- Note des leases = moyenne des notes des labels (pas de pondération)
- Les valeurs bruts des labels sont filtrées grâce au Json Filter extrait de Nova. Permet des filtres complexes.

Labels


Leases
X
Y
Z
Labels
0
1
0.5
1
0.3
0
Ranking
0.7
1
0
Average
0.57
0.77
0.17

Technique

Code réutilisé de Nova :

- Json Filter (légèrement adapté)

```
['and',
        ['<=', '$energy.max', 300],
        ['>=', '$energy.min', 100]
]


['<=', '$energy.range', 200]
```

Base de données :

- Ming (surcouche de PyMongo)
- Mapping direct depuis les classes Python
- Garantie la structure contrairement à PyMongo
- À terme : plus générique
    - Implémentation PyMongo + SQL-Alchemy
    - Mais on perd le côté pratique de Ming…

Et après...

- Focus sur les use-cases de base :
    - Réservation dans Nova d'une lease immédiate
    - Puis après on verra pour les leases Best-Effort et Schedule

- On réserve des ressources abstraites ou des hôtes physiques ?
    => Hôtes physiques

- Homogénéité des hôtes réservées ?
    => Finalement pas strictement identiques... mais doivent correspondre aux contraintes de la requête

- Catalogue de labels, contraintes complexes, notation entre 0 et 1
    => Abandon / reporté à plus tard ?
    => Utiliser plutôt Nova weighing pour la pondération "énergétique"