Prévision Numérique et Scriptage à Météo-France

Eric Sevault – CNRM/GMAP/ALGO
Contributions de Véronique Mathiot, Guillaume Beffrey, Philippe Marguinaud,
Yann Seity, Joël Stein, Gaëlle Kerdraon, Hervé Bénichou,
Dominique Birman, François bouyssel
et l'amicale communauté des utilisateurs OLIVE, Perl et Python

Sommaire

- Contexte : la PN à Météo-France
- Contexte : les moyens de calculs
- Merci Larry : expériences de recherche OLIVE en PN
- Encore merci Mr. Wall : step by step avec MTOOL
- Plonger dans le VORTEX ?
- Pourquoi Python ?

I. La PN à Météo-France

La prévision numérique

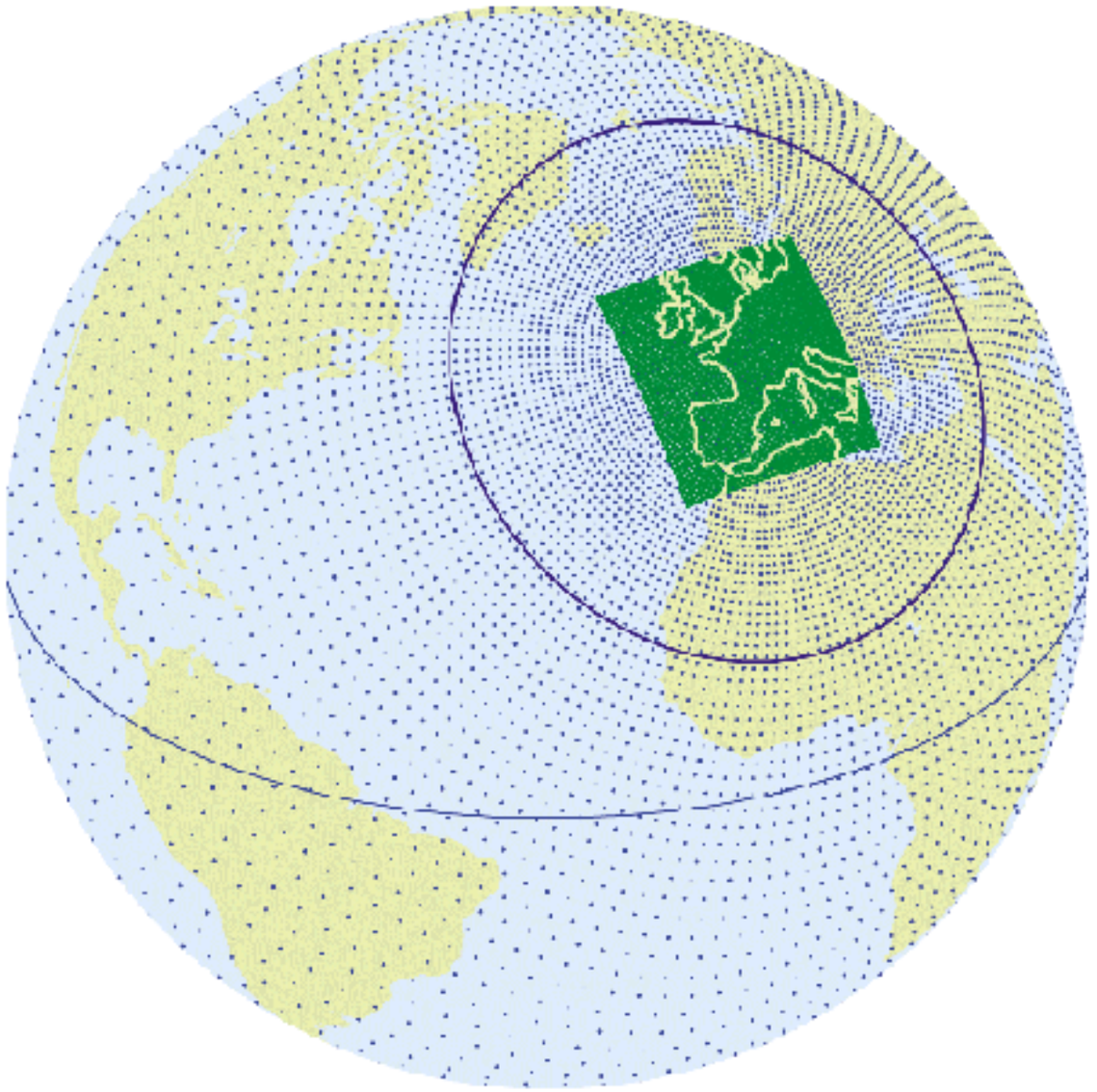
- La modélisation de l'atmosphère s'appuie sur des modèles numériques intégrant les équations générales de l'hydrodynamique : équation du mouvement, équation de continuité, thermodynamique, équation des gaz parfaits.
- L'intégration temporelle est réalisée par une approche Lagrangienne (ou particulaire) ou Eulérienne.
- Jeu de paramètres définissant l'état du modèle (dit à équations primitives) : composantes horizontales du vent, température et humidité sur quelques dizaines de niveaux jusque vers 50 à 100km d'altitude, paramètres de surface, concentrations de divers éléments : eau liquide ou glace, ozone, etc.
- Depuis 2000, la tendance des modèles est au relâchement de l'hypothèse hydrostatique pour des modèles à fine résolution horizontale.

Quelques modèles de PN

- Plusieurs types de modèles
  - Global (IFS / ARPEGE)
  - Aire Limitée (ALADIN / AROME)
- Diverses modalités d'utilisation
  - Modèle déterministe : prévision directe à partir d'un état analysé
  - Modèle ensembliste : perturbations initiales et panel de prévisions
- Diverses configurations d'assimilation
  - Trouver le meilleur état initial de l'atmosphère
  - En utilisation à Météo-France : Couplage externe, 3D-Var et 4D-Var
- Autres : modèles de vagues, chimie, dispersion, alertes diverses

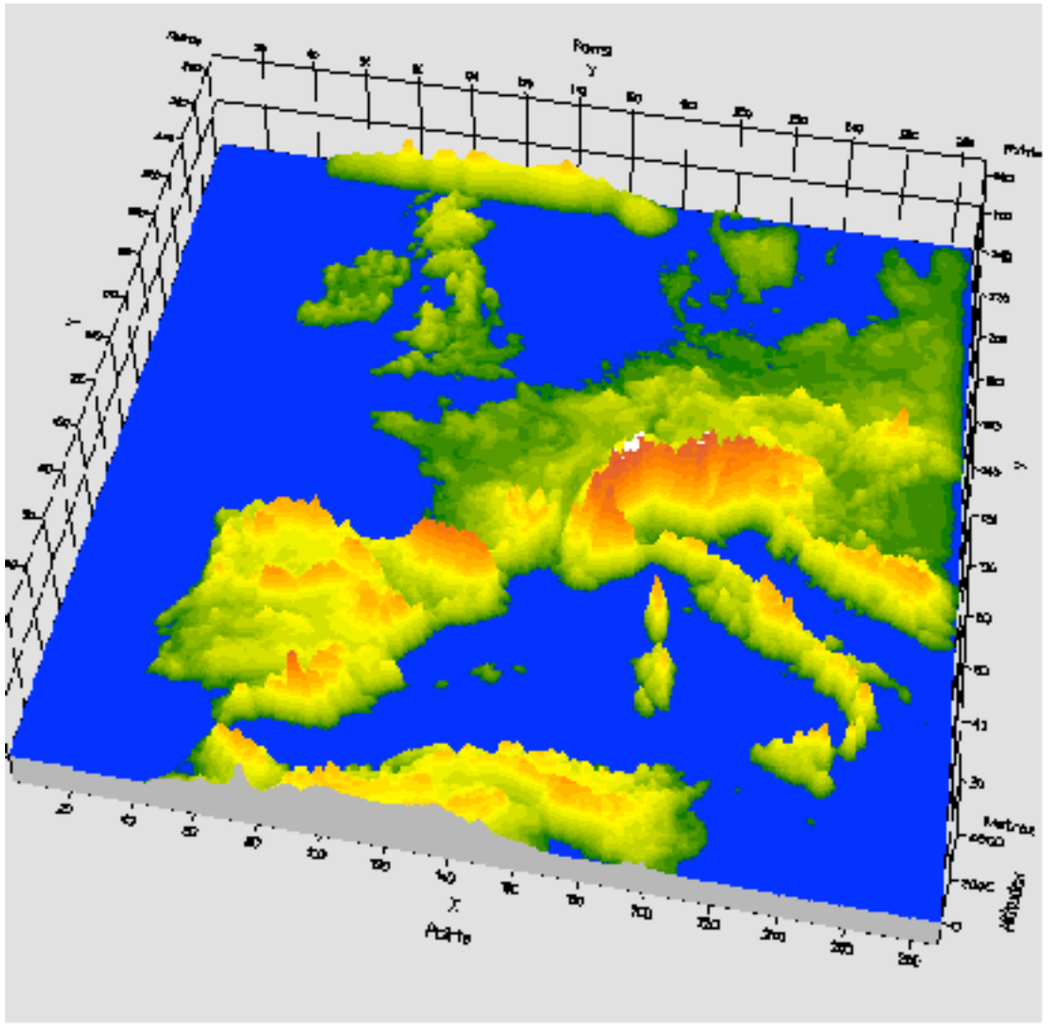
ARPEGE

- Modèle Global
  - Spectral T798
  - H : 10 km France
  - H : 60 km Antipode
  - L : 70
  - dT = 600 s.
  - de 72h à 106h
- Basculé / Étiré
- Collaboration internationale / IFS



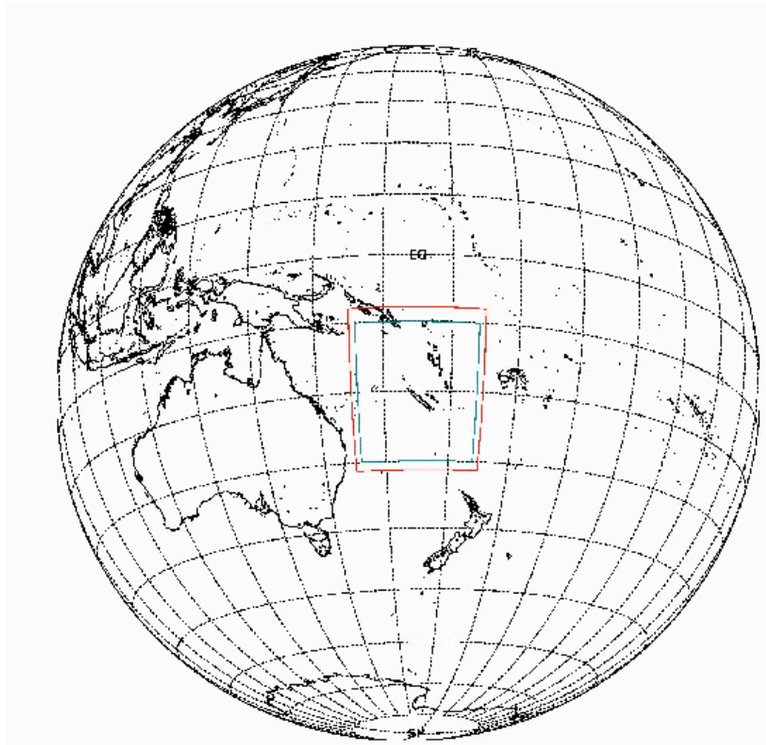
ALADIN France

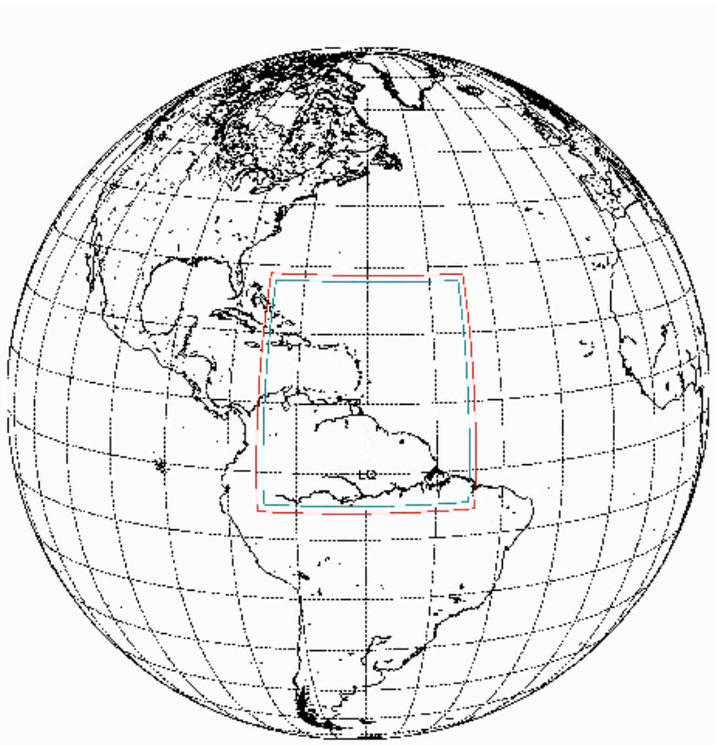
- Aire limitée
  - Bi-Fourier
  - H : 7.5 km L : 70
  - dT = 450 s.
  - de 36h à 54h
  - CPL ARPEGE % 3h
- Assim 3D-Var
- Collaboration internationale / Consortium HIRLAM



ALADIN Outre-Mer (en test)

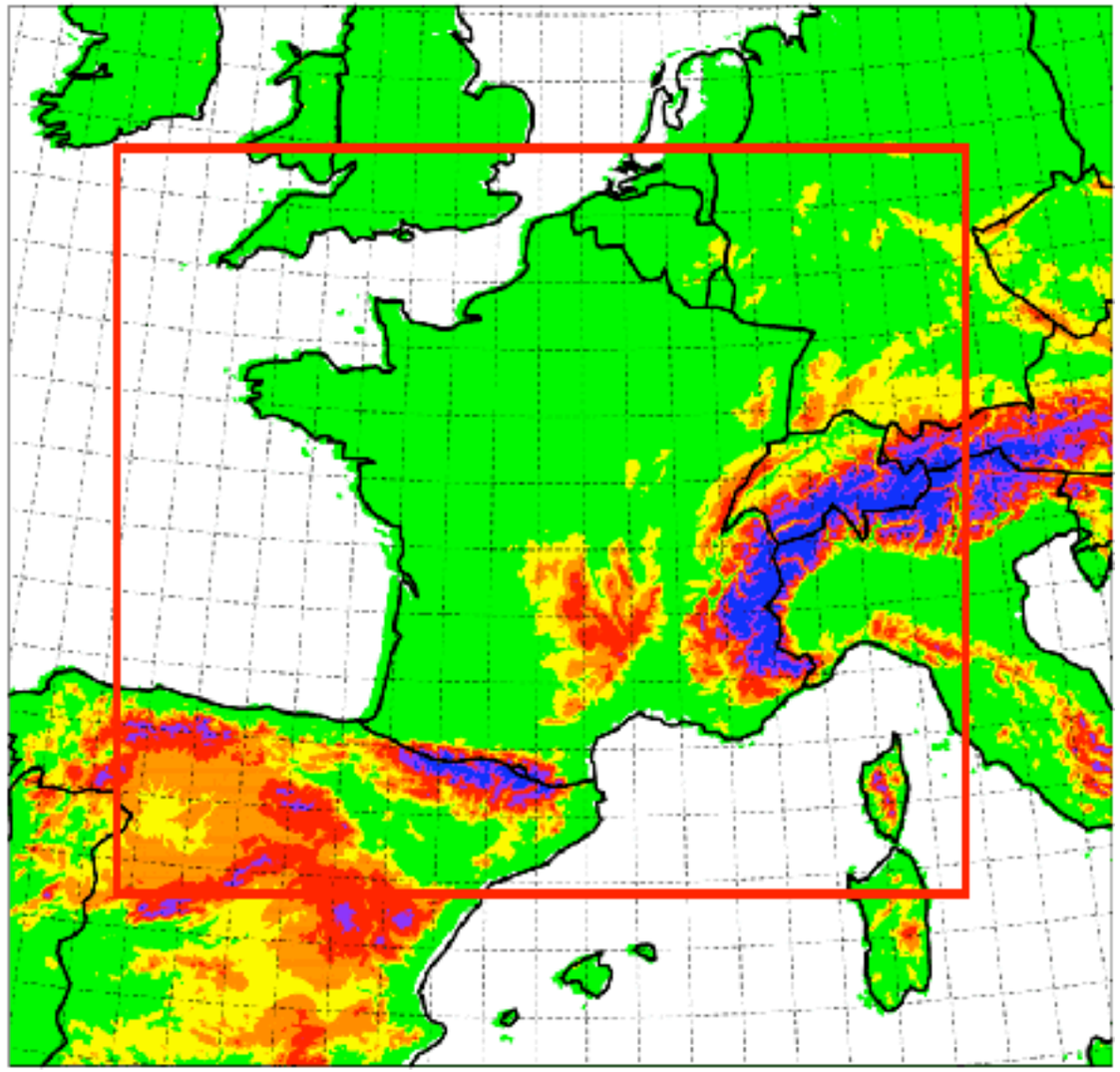
- Nouvelle-Calédonie / Polynésie / Antilles - Guyane
- Extractions obs / couplages IFS asynchrones / surface ARPEGE



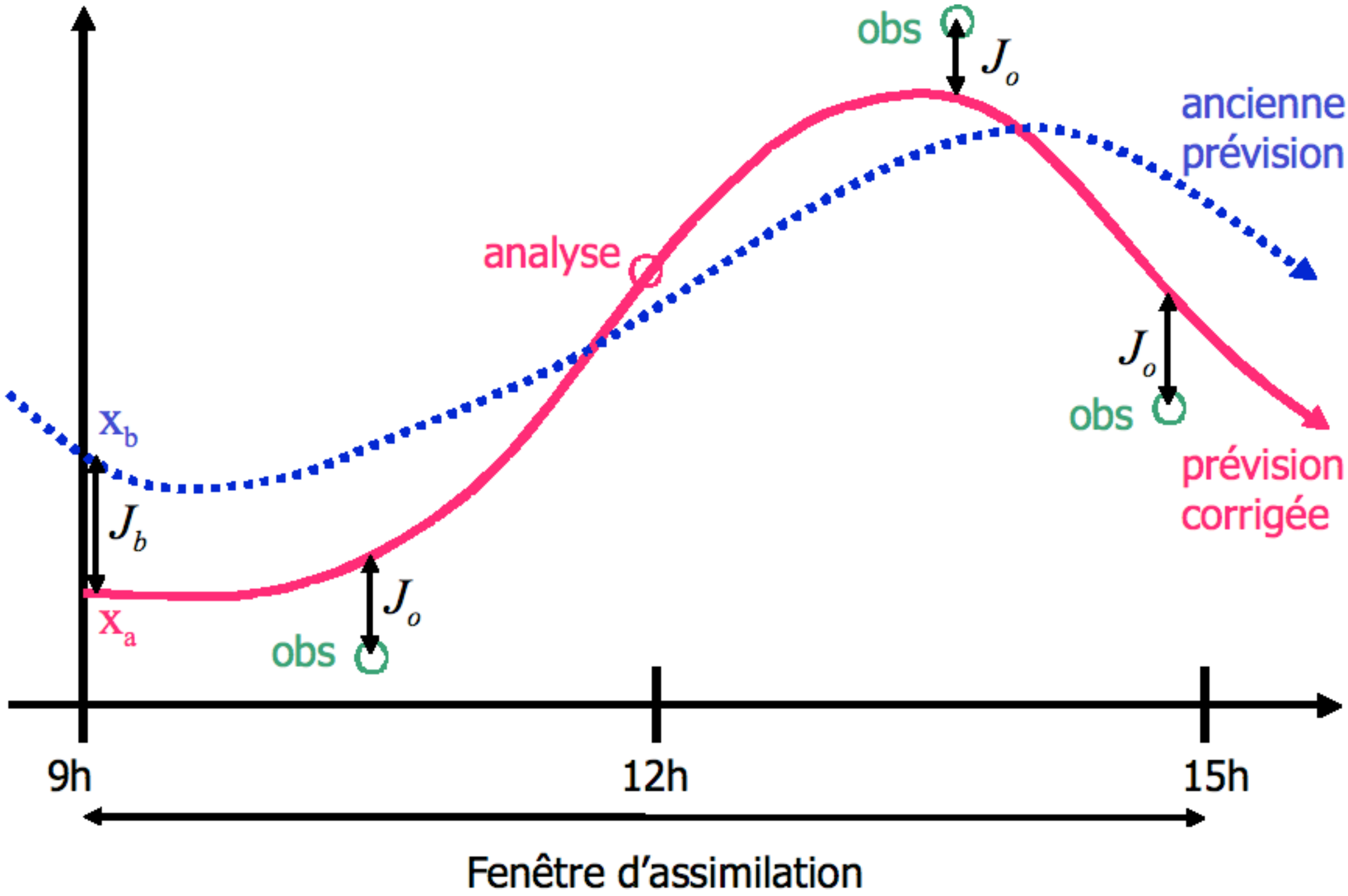


AROME France

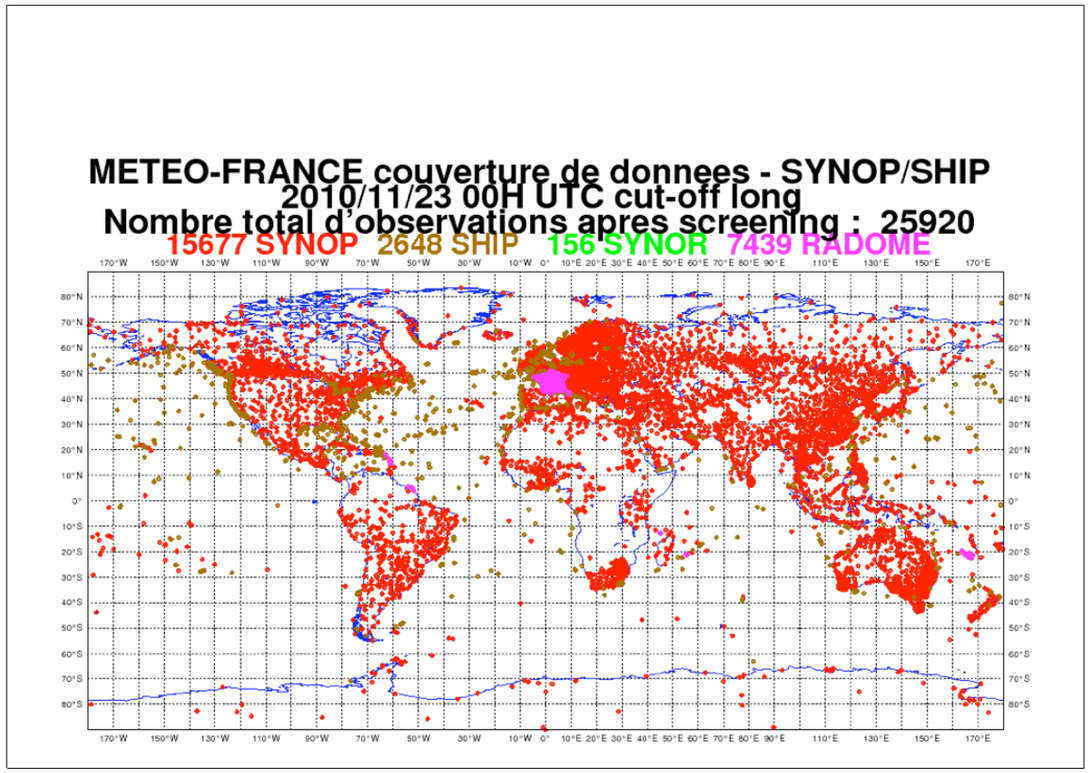
- Aire limitée NH
  - Bi-Fourier
  - H : 2.5 km L : 60
  - dT = 60 s.
  - 30h / R = 0, 6, 12, 18
  - 3h / R = 3, 9, 15, 21
  - CPL ARPEGE % 1h
- Assim 3D-Var

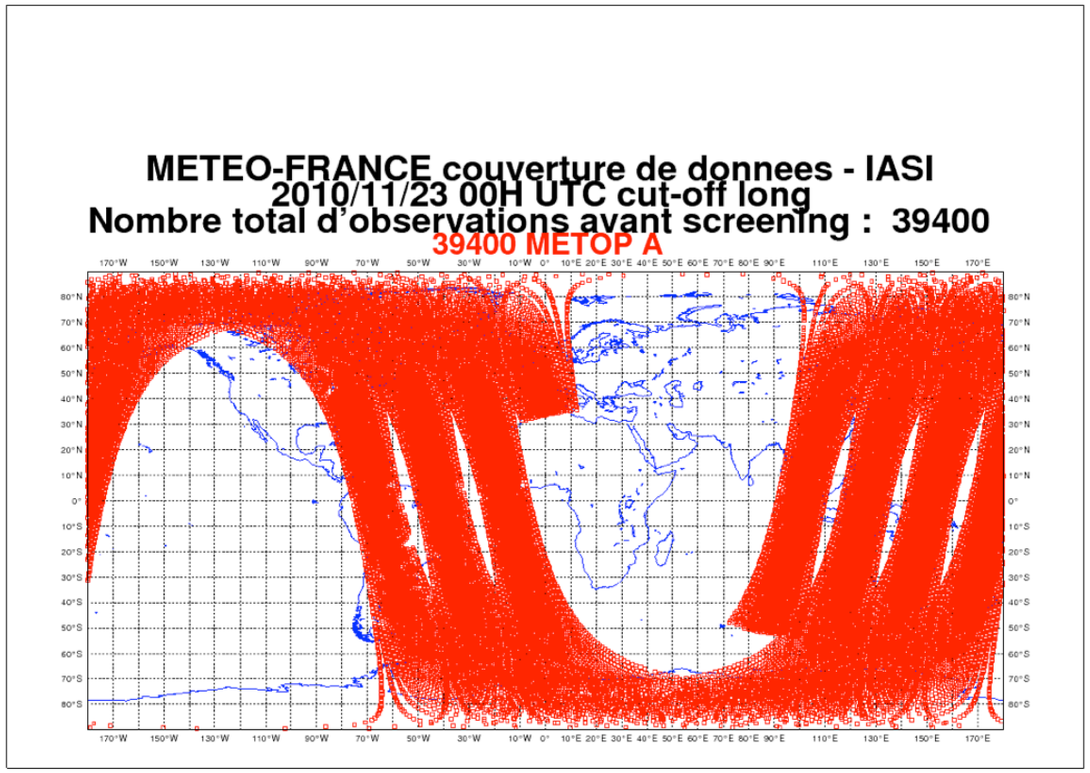


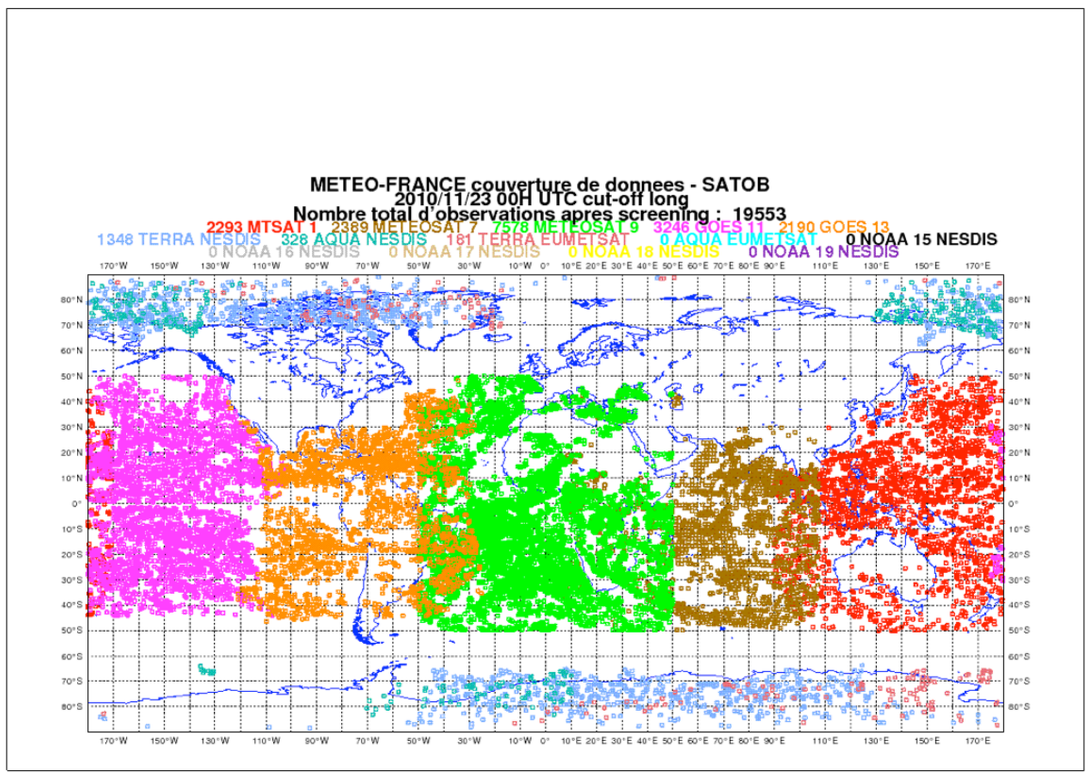
Principe du 4D-Var

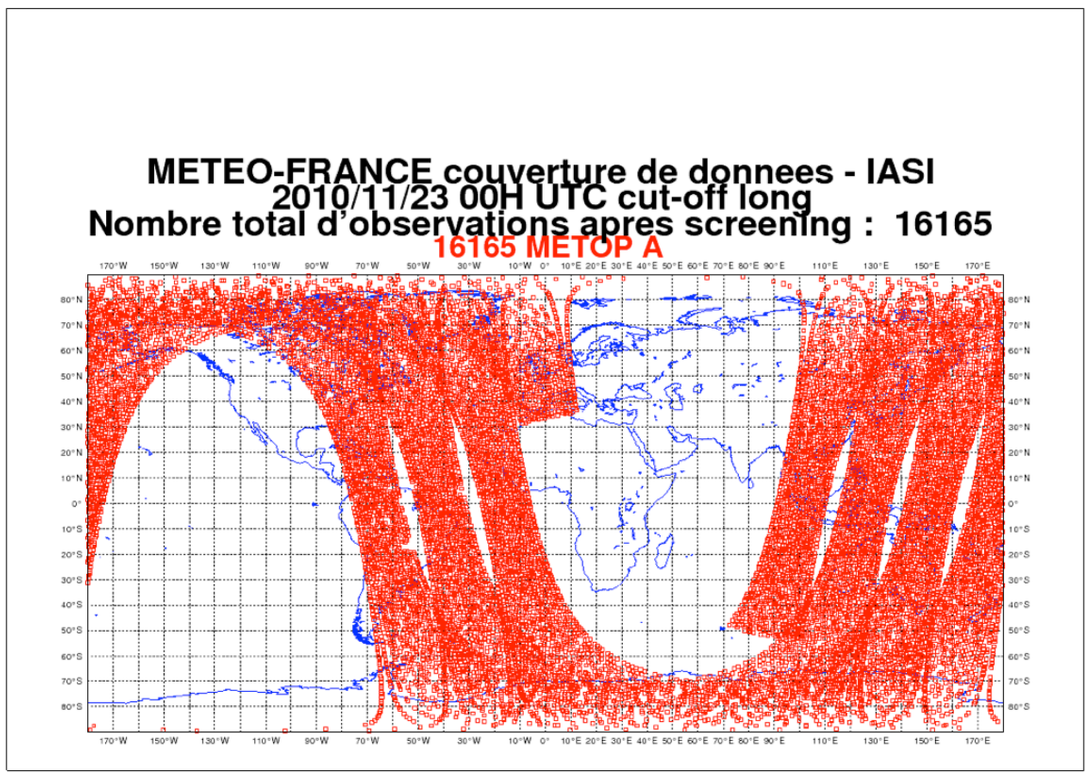


Observations conventionnelles & satellites

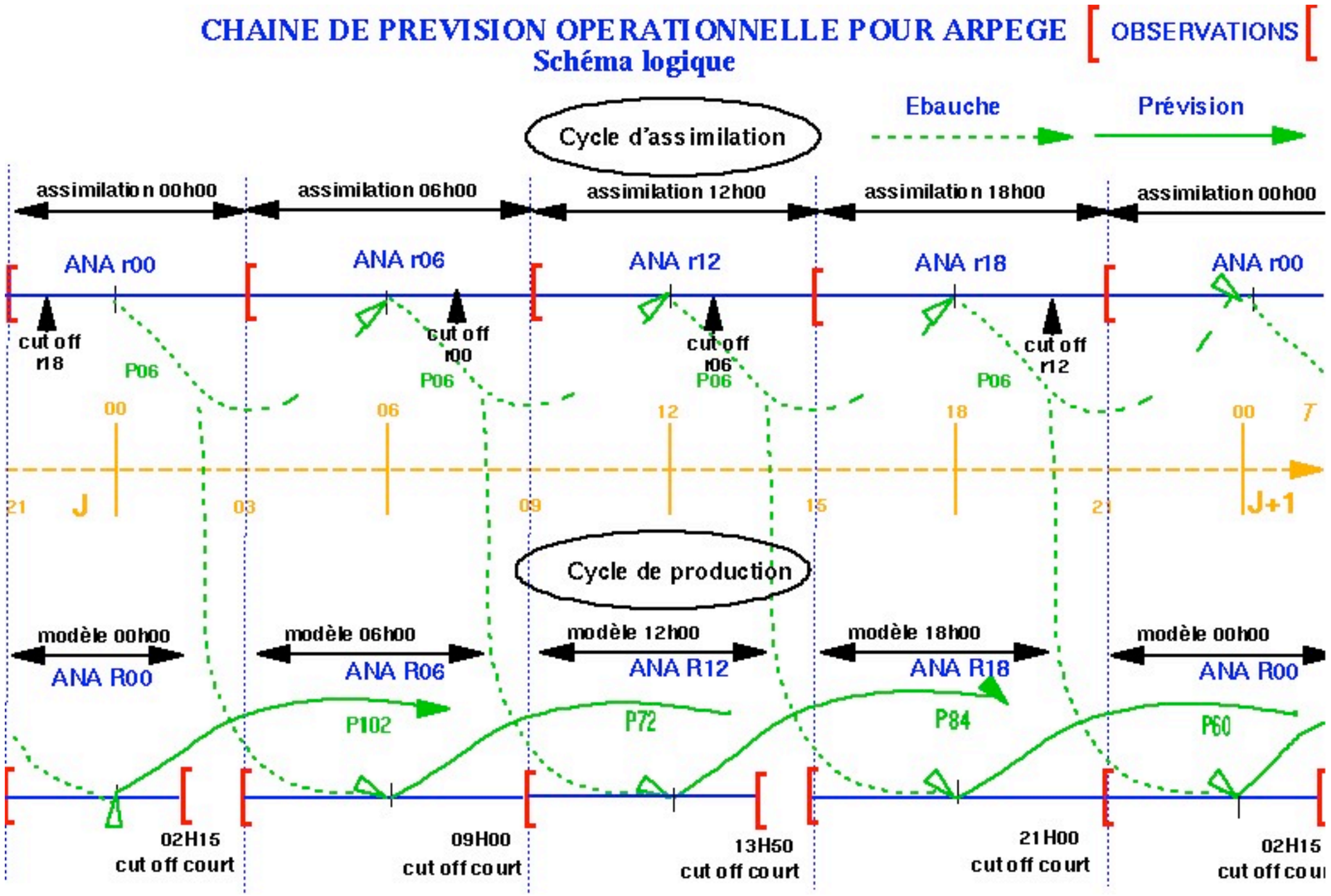








Enchaînement opérationel (exemple simplifié)



La fabrique à modèles (plus ou moins top)

- En plus d'être utilisés en opérations, tous ces modèles doivent être développés (!), de nouveaux schémas physiques ou algorithmiques doivent être expérimentés, validés, etc. Et ce sur un grand nombre de cas / périodes.
- Le volume de jobs recherche est en général bien supérieur à celui des opérations, sans en bénéficier des priorités, en terme d'exploitation des ressources.
- Les questions suivantes deviennent donc de première importance :
  - facilité de configuration, comparaison, partage entre utilisateurs ;
  - gestion efficace du flux de tâche, capacité de reprise ;
  - robustesse, mise au point, support applicatif ;

La PN d'un point de vue pratique

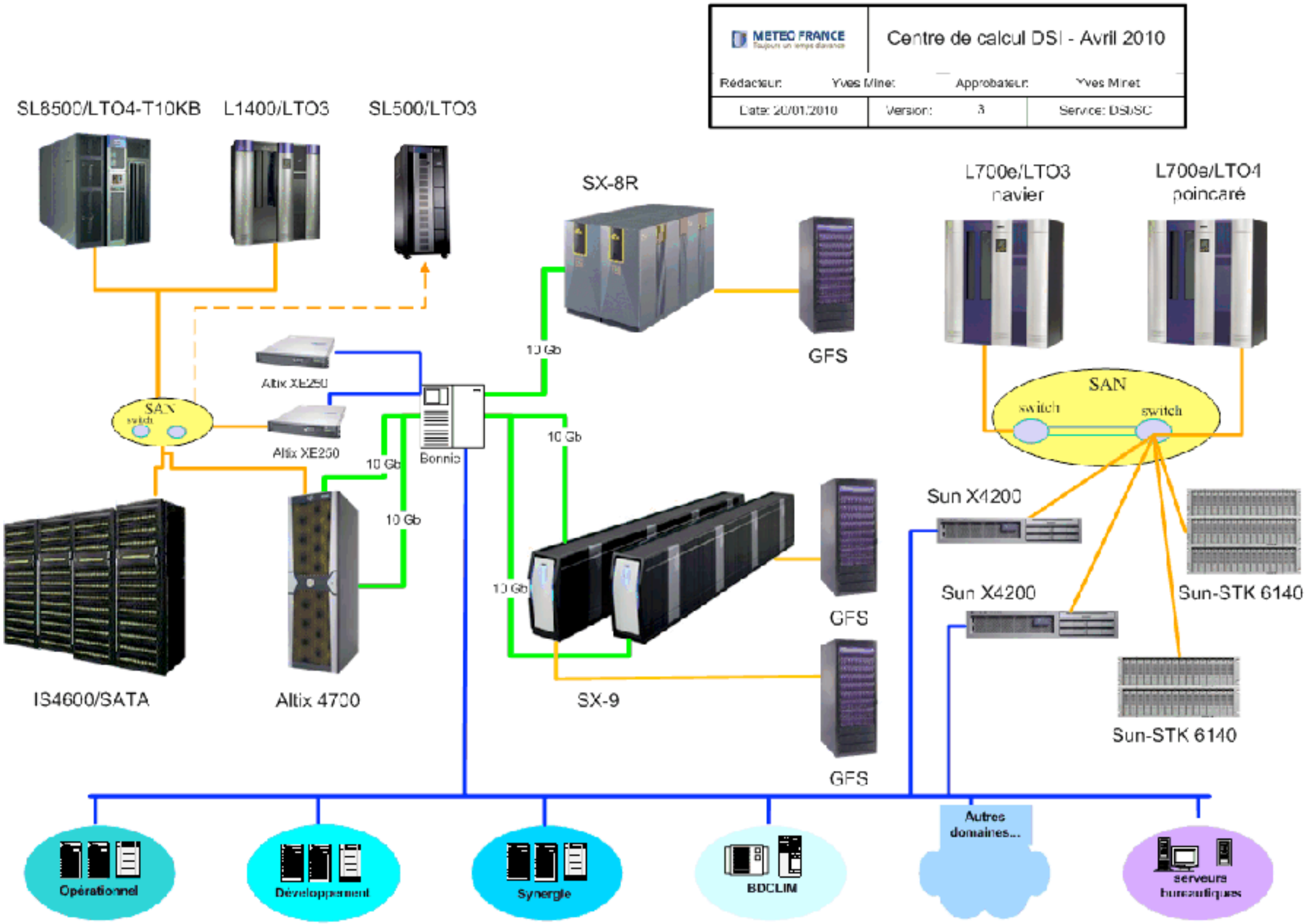
- Un grand nombre de tâches, sur des environnements potentiellement variés
- Un réseau de dépendances qui se complexifie
- Une exigence de service rendu (op & recherche)
- Une augmentation sans fin des volumes de données :
  - en entrée : jeux d'observations, états analysés
  - en sortie :
    - augmentation de la résolution horizontale et verticale
    - augmentation du nombre de champs diagnostiques, statistiques observations
    - multiplication des domaines de post-traitement

Une approche globale (encore lacunaire)

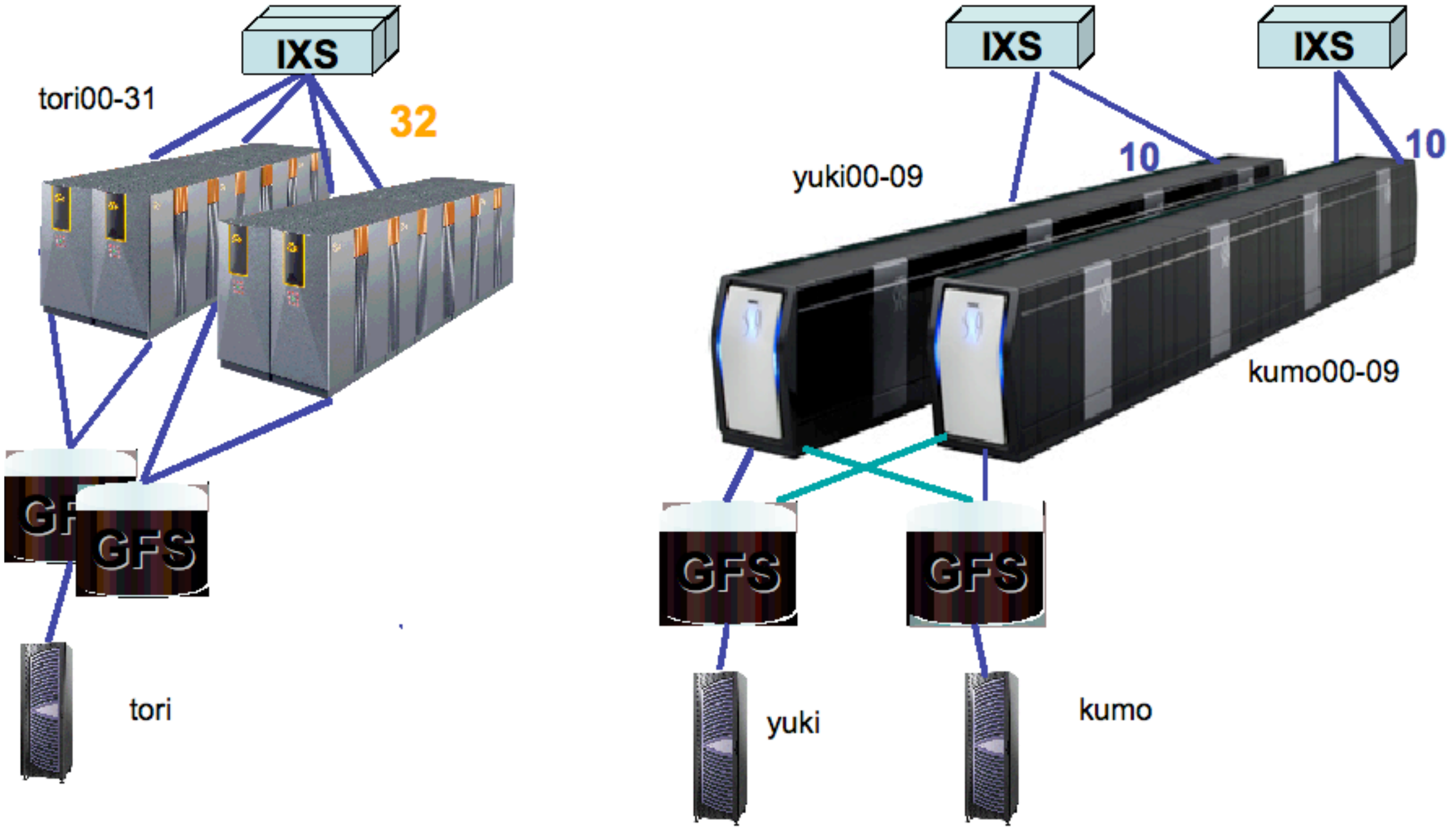
- De la configuration abstraite à l'exécution physique, du poste de travail aux super-calculteurs :
  - PC linux:
    - XCDP pour le monitoring, Firefox pour l'interface utilisateur
    - Post-traitement des données
  - Serveurs Linux
    - Apache / Mod_perl + SWAPP pour configurations d'expériences OLIVE
    - SMS / FSK pour le séquencement des tâches
  - Cibles de calcul : NEC Météo-France, IBM au CEPMMT
    - Multi-step MTOOL / utilitaires de bas niveau
- Les langages de scripts sont sollicités à tous les niveaux

2. Le centre de calcul

Vue d'ensemble



Les super-calculateurs NEC



Stockage

- Architecture
  - Sun-STK SL8500 (T10000, LTO4)
  - 2 IS4600
  - SGI Altix 4700 (24 CPU Itanium2, 96 Go RAM)
  - HSM : DMF
- Depuis 2010
  - 1.3 Po de disques SATA pour les utilisateurs
  - Capacité de stockage globale : 4.2 Po
- Mais... pas de système de fichier virtuel global => solutions ftp

3. Expérimentations de PN en mode recherche avec OLIVE

De quoi est-il question ?

- Fondamentalement les expériences de PN ne se distinguent pas du premier bout de code exécutable venu
- Agencement de séquences incluant :
  - une collecte des conditions aux limites
  - l'exécution(s) d'un noyau de calcul
  - la sauvegarde de résultats
- Et aussi :
  - une grande variété de données et de configurations
  - des exécutions itératives ou cycliques
  - une structure de dépendances pouvant se complexifier rapidement
  - une soif inextinguible en ressources de calcul et de stockage

Faire de nécessité vertu

- À la fin des années 90, il n'était plus possible de reproduire en mode recherche la chaîne opérationnelle et réciproquement
- Les tentatives de conception d'un script générique ksh pour toutes les configurations d'assimilation semblaient ne pas devoir aboutir : l'outil (ksh) n'était visiblement plus à même de gérer la complexité des configurations
- Projet OLIVE de janvier 2001 à janvier 2004
  - 2002/09 : v.x, webmars (CEPMMT) largement remanié et étendu
  - 2003/02 : v.0 pour gmapistes joueurs
  - 2003/12 : v.1, réécriture intégrale du noyau, bascule transparente, arrivée de la seconde charrette d'utilisateurs

Des spécifications très générales

- Enregistrer tous les changements des configurations en opérations et de leurs composants élémentaires
- Construire, dupliquer, modifier, piloter des expérimentations de recherche quelqu'en soit la nature
- Permettre quelques diagnostiques météo
- Accéder de la façon la plus uniforme possible à toutes ces applications : l'application qui visualise l'information est la même que celle qui la produit ou modifie
- Pour tout ce qui relève de la PN, rien ne doit être caché à l'utilisateur final qui doit pouvoir effectuer toute modification à caractère scientifique ou technique
- Utiliser des outils éprouvés de la communauté libre / météo

SWAPP : un cadre commun

- Shared Weird APPlications, est à la fois un cadre commun :
  - logiciel (framework de modules Perl)
  - de convivialité (communauté olive)
- L'implémentation prend la forme d'un système de fichiers virtuel (VFS) constitué d'objets persistants (BerkeleyDB) dotés d'attributs et de méthodes variés
- Les objets SWAPP sont des objets Perl sérialisés qui vivent dans un arbre dont les noeuds sont eux-mêmes des objets : chaque base de données a sa propre "racine" (root node), qui a son tour peut avoir des enfants (kids) et ainsi de suite... cloné sur Linux
- Les bases de données SWAPP sont analogues à des unités de disque montées à la racine ou dans une base déjà montée. L'utilisateur final ne voit qu'un arbre hiérarchique
- Le module Swapp::Vfs fournit une interface commune pour tous les types de stockage (ie: tous les types de bases SWAPP)

Généricité de SWAPP

- En lui-même n'est pas spécifiquement destiné à la PN Arpege / Aladin / Arome : OLIVE est une application dans SWAPP, avec d'autres applications ou plugins
- SWAPP n'a pas de limitation conceptuelle à ce qu'il peut accueillir : il suffit d'étendre ou de créer les classes d'objets désirés
- SWAPP est un ensemble logiciel Perl accessible sous trois contextes :
  - en mode shell émulé (psh)
  - sous son API naturelle (scriptage)
  - au travers d'un serveur Apache - mod_perl (99 %)

Exemple : le pseudo shell

~

Fichier Édition Affichage Terminal Onglets Aide

```
sxalgo1-verolive: cd etc/hosts
sxalgo1-verolive: ls
rwxr-xr-x  69656 Object::Host          root 22 mar 2005 lxgmap39
rwxr-xr-x  69657 Object::Host          root 22 mar 2005 sxalgo1
rwxr-xr-x  69658 Object::Host          root 22 mar 2005 sxcoope1
rwxr-xr-x  69659 Object::Host          root 22 mar 2005 sxproc1
rwxr-xr-x  69660 Object::Host          root 22 mar 2005 sxrecyf1
rwxr-xr-x  77542 Object::Host          root 06 jun 2005 basilic
sxalgo1-verolive: cat sxalgo1
bless( {
  'SWAPP_MASTER' => 'verolive',
  'SWAPP_HOSTNAME' => 'sxalgo1.cnrm.me
  'SWAPP_NICKNAME' => 'groucho',
  'SWAPP_PORT' => '8181'
}, 'Object::Host' )
sxalgo1-verolive:
```

~

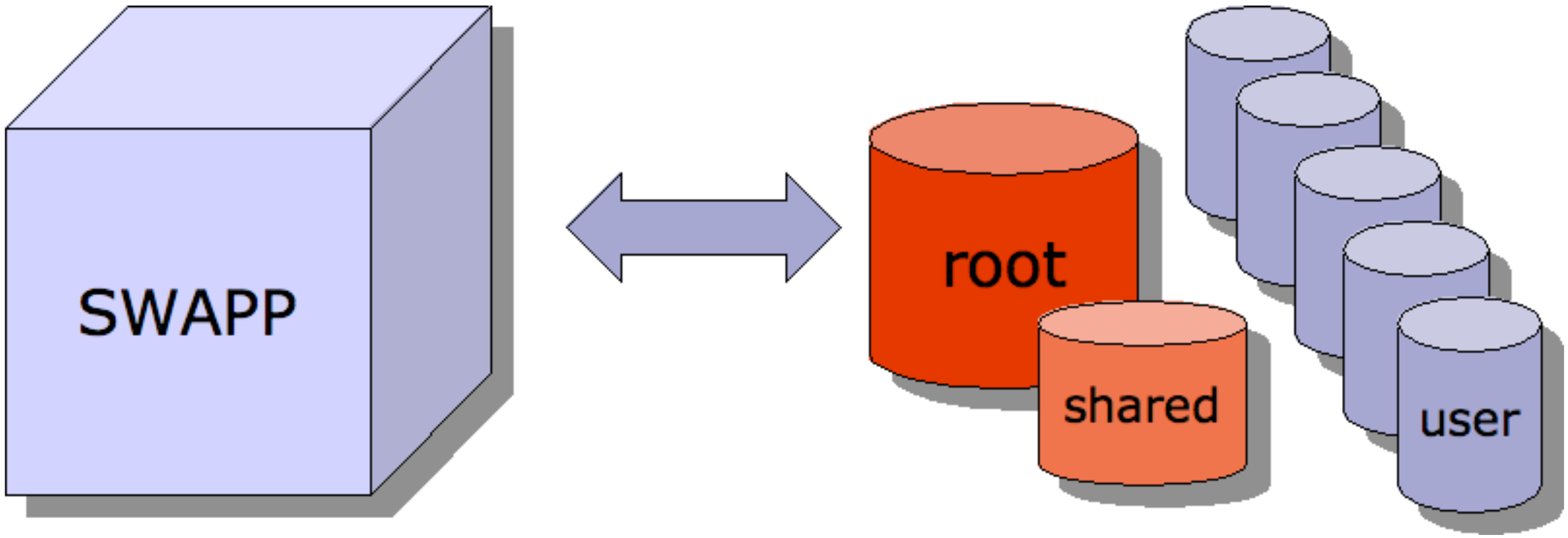
Fichier Édition Affichage Terminal Onglets Aide

```
sxalgo1-verolive: pwd
/home/mrpm/mrpm631
sxalgo1-verolive: ls
rwxr-xr-x      1 Object::Clipboard    mrpm631 08 nov 2005 clipboard
rwxr-xr-x      2 Personal::Trash      mrpm631 28 oct 2005 trash
rwxr-xr-x   7737 Personal::Directory  mrpm631 29 jun 2005 diagnostics
rwxr-xr-x   1701 Personal::Directory  mrpm631 02 nov 2005 experiments
rwxr-xr-x  87031 Object::Favorites    mrpm631 17 mar 2005 favorites
rwxr-xr-x  87043 Personal::Directory  mrpm631 30 jun 2005 tmp
rwxr-xr-x 241157 Personal::Directory  mrpm631 07 nov 2005 cleaning
rwxr-xr-x 254675 Personal::Directory  mrpm631 23 mar 2005 preferences
sxalgo1-verolive: ls preferences
rwxr-xr-x 228982 Sms::Monitor         mrpm631 03 nov 2005 sms_monitor_sxalgo1

sxalgo1-verolive: ls experiments
rwxr-xr-x  26391 Personal::Directory  mrpm631 16 jan 2004 validations
rwxr-xr-x 228195 Olive::Experiment    mrpm631 22 nov 2004 600N
rwxr-xr-x 230332 Olive::Experiment    mrpm631 25 nov 2004 6019
rwxr-xr-x 230385 Olive::Experiment    mrpm631 25 nov 2004 601M
rwxr-xr-x 230990 Olive::Experiment    mrpm631 25 nov 2004 601R
```

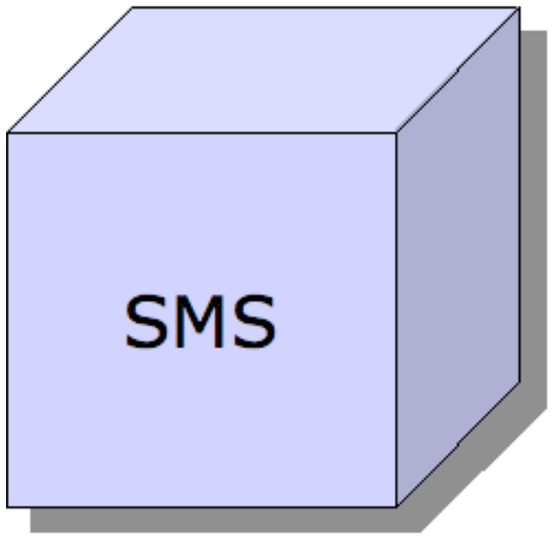
Où est stockée l'information ?

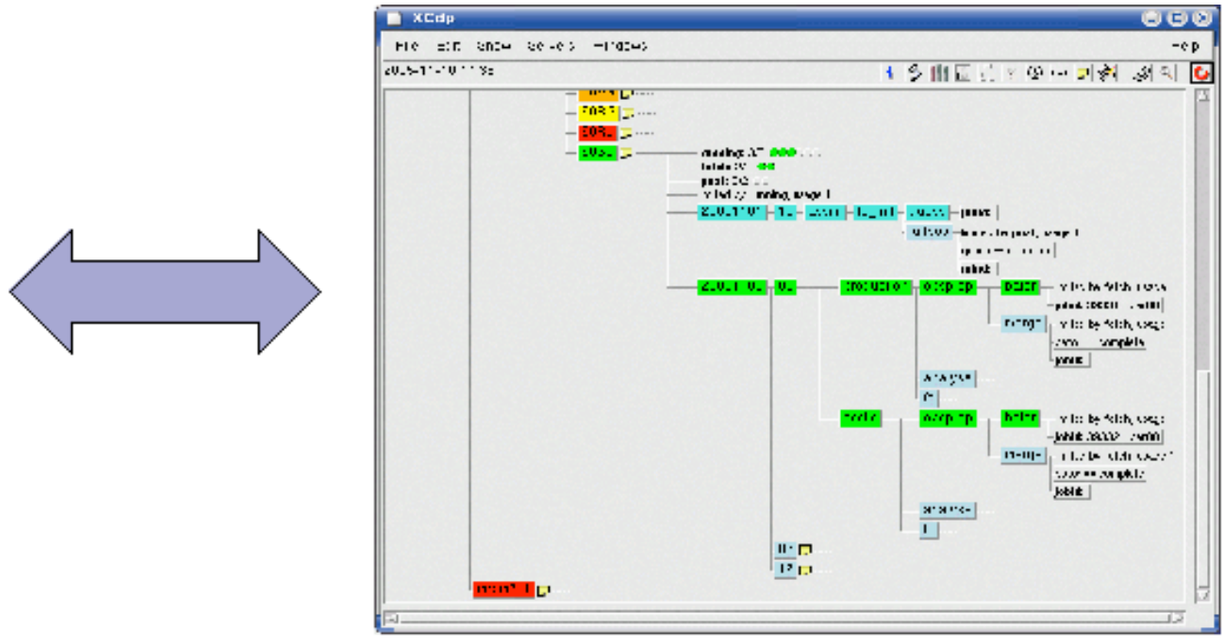
- Les objets persistants (et tout est objet) sont stockés dans des bases locales au serveur SWAPP, rien n'est prélevé sur les ressources propres du compte utilisateur
- Ces bases peuvent être dupliquées, sauvegardées, exportées en format ascii, etc.



SWAPP, SMS et FSK

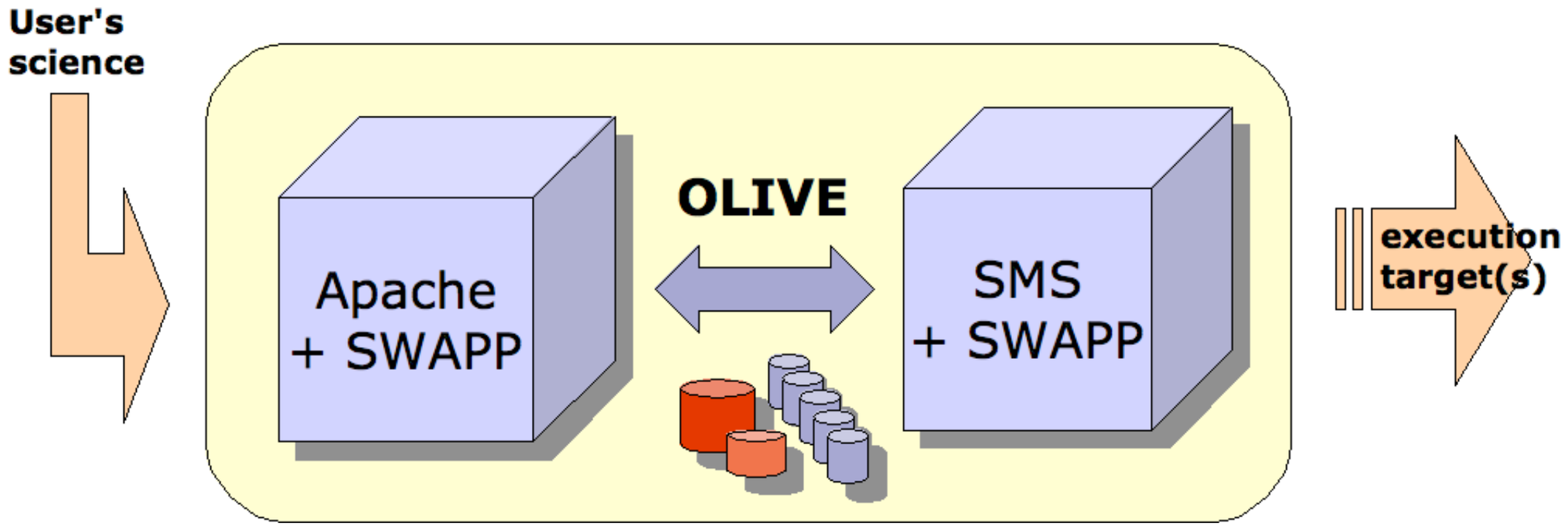
- SMS : Scheduling & Monitoring System — en charge du séquencement, de la gestion des dépendances, de la soumission des tâches — XCDP est son interface graphique
- Les principales interactions avec le monde réel, "fetch", "submit" et "kill" sont déléguées à un démon (fskd... écrit en Perl, Sys::Gamin) qui les traite de façon asynchrone





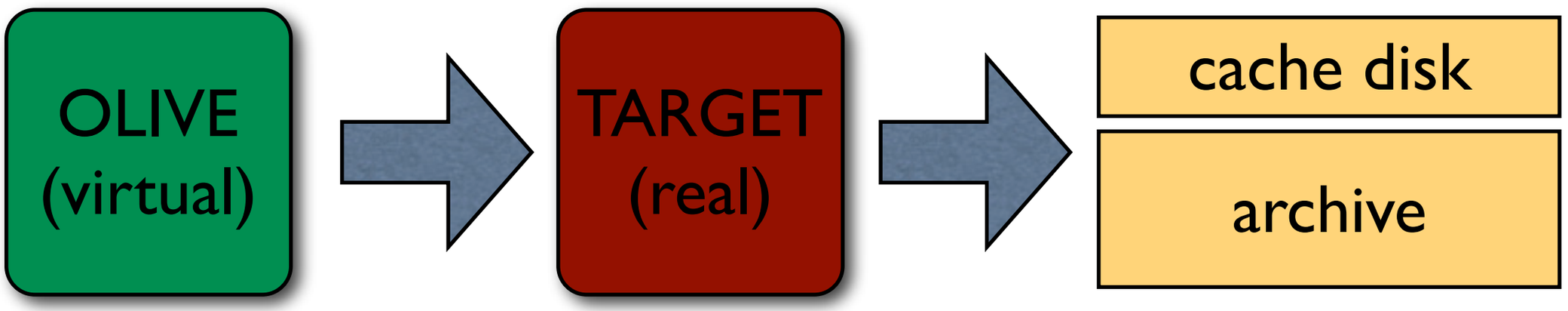
Le serveur OLIVE

- Ce que l'on nomme serveur "OLIVE" n'est donc rien d'autre qu'un hôte hébergeant des démons APACHE/SWAPP et SMS/FSK, ce dernier embarquant lui-même les modules SWAPP
- Accessoirement : perl, mod_perl, un certain nombre de modules du CPAN et BerkeleyDB !



Cibles d'exécutions

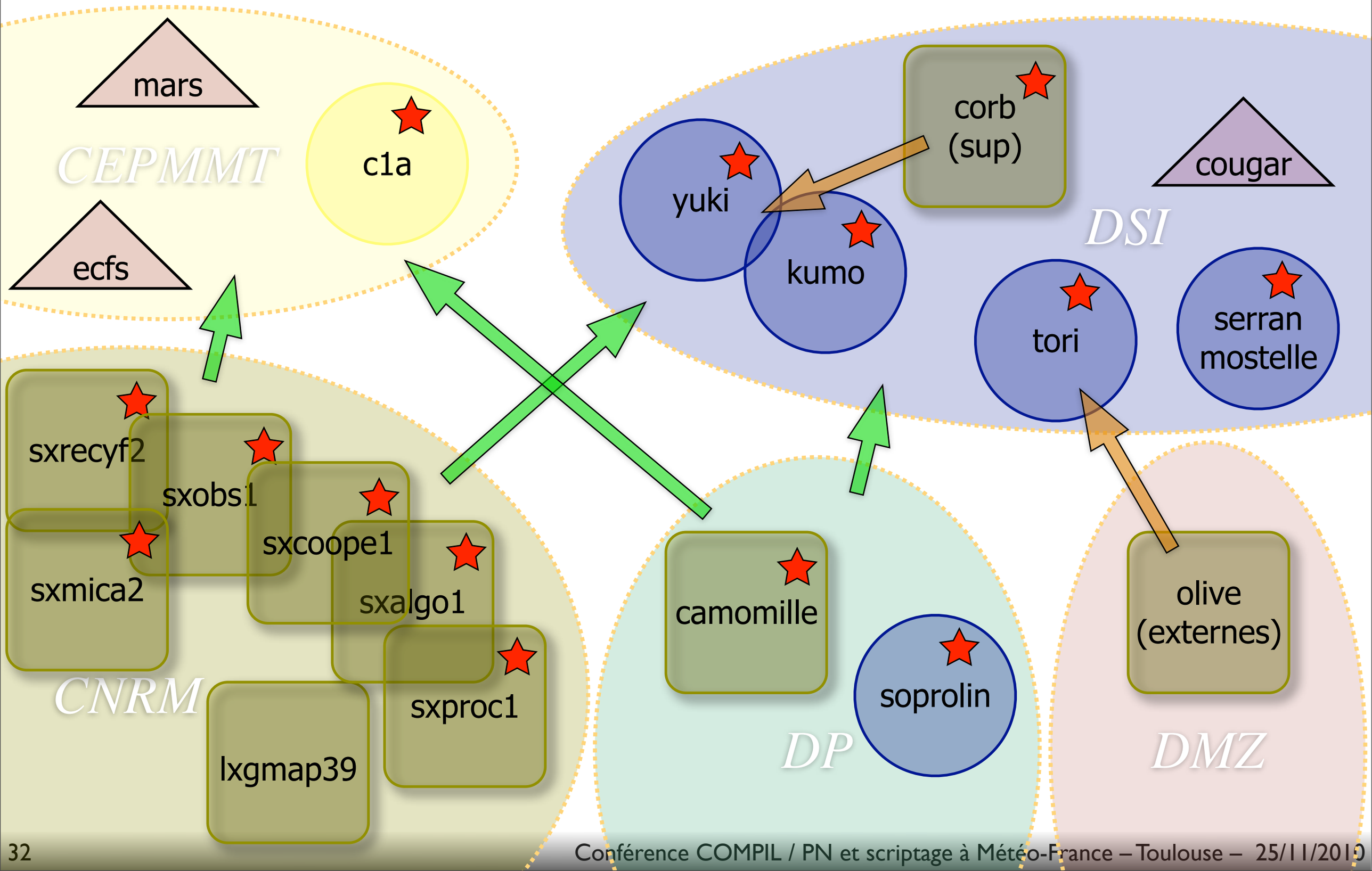
- Ce sont avant tout des objets répertoriés dans le VFS SWAPP comme étant des "target experiment"
- Ces calculateurs disposent d'un outillage minimal : la "toolbox" OLIVE, et d'un accès (ou pas) à des systèmes de stockage ainsi que la possibilité de communiquer avec un démon SMS
- Ils sont interfacés (ou pas) avec l'un des systèmes de soumission gérés par SWAPP (PBS, NQS2, SGE, LoadLeveller)
- Nos cibles favorites sont les 3 systèmes NEC de MF, mais aussi le super-calculateur recherche du CEPMMT, quelques cluster Linux, ou même notre PC !



Qu’est ce qui rend SWAPP particulier ?

- Il y a quelques années, pas grand chose n'était disponible dans le rayon des bases de données objets accessibles en ligne, sauf pour les pionniers de Zope ou les acharnés des mises à jour Java. Depuis, “Ruby on Rails” ou “Turbo Gears” (Python). Qu'est ce qui est un peu spécifique avec SWAPP ?
  - Tout est en Perl : le code, les objets. Quoi que vous vouliez faire, c'est faisable... et pas bien compliqué (d'ailleurs c'est peut être déjà fait)
  - Le code EST la configuration (idée reprise dans Ruby on Rails).
  - BerkeleyDB est une base à accès direct vraiment rapide, pratiquement sans installation ou administration
  - Le double système de cache (BerkeleyDB et celui de SWAPP)
  - La simplicité de la hiérarchie des objets calquée sur Linux : parents et enfants sont des éléments très communs dans toute structure sociale (points de montage, liens symboliques et hard)

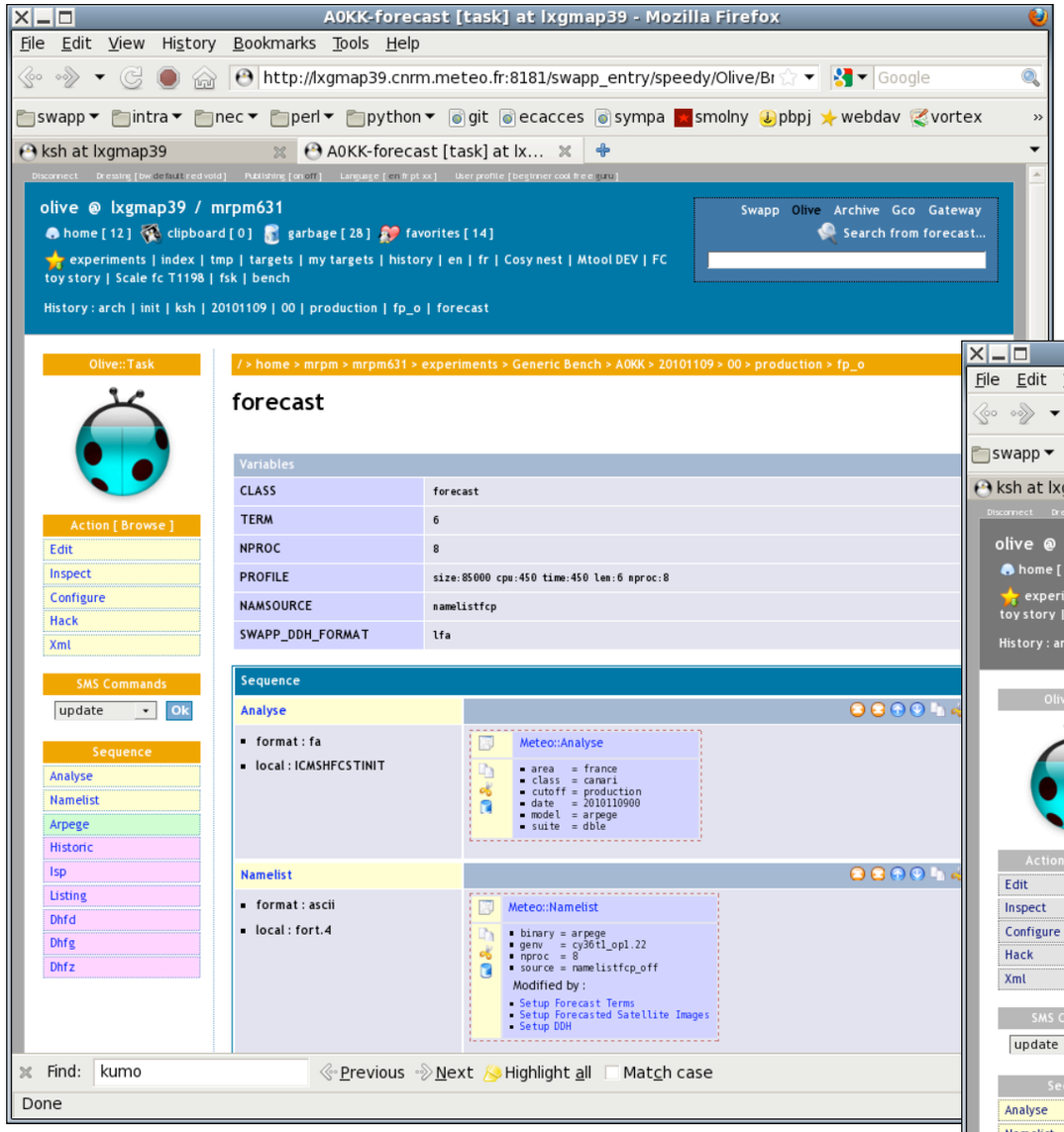
Cartographie OLIVE

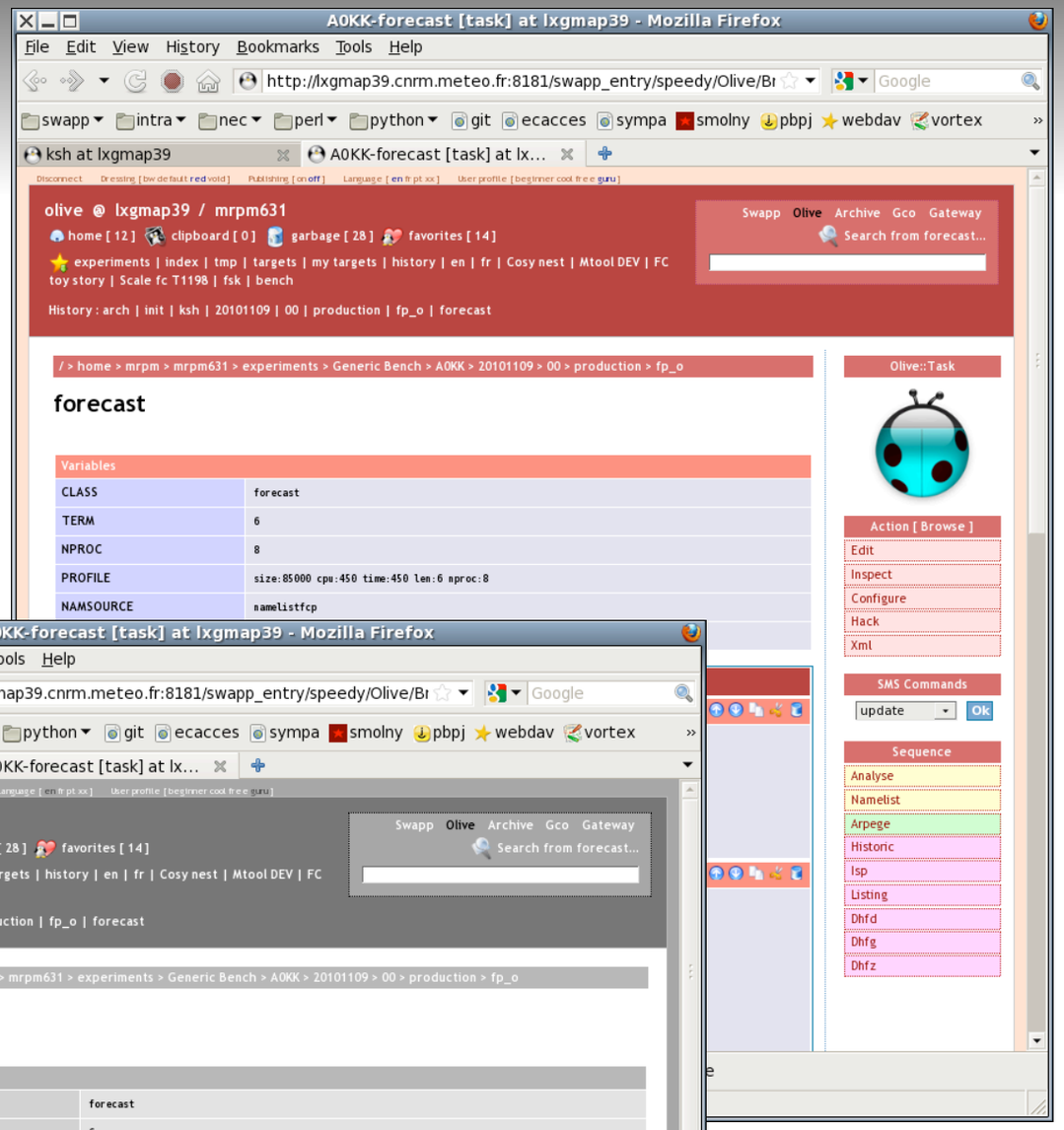


Fonctionnalités

- Celles de l'interface s'appliquent aux expériences pour élaboration ou évolution :
  - copié / coupé / collé / ordre
  - suppression, ajouts, renommage, liens symboliques, etc.
- Celles plus spécifiques à l'application OLIVE :
  - gestion des dates, réseaux, cutoff, échéances
  - temps réel, temps différé ou mixte
  - boucles complexes, activations conditionnelles, etc.
  - toutes dépendances, sélection de ressources, paramétrisations, ressources partagées, cibles d'exécution, etc.

OLIVE snapshots





OLIVE : un bilan (1)

- Les mêmes outils, le même vocabulaire, les mêmes plantages
- Toute correction bénéficie à tous
- La visibilité de toutes les expériences au travers des comptes et des serveurs
- La facilité de copie d'expériences, y compris entre serveurs
- Le monitoring XCDP mutualisé (tout le monde accède aux expériences en cours de déroulement)
- Une mise en train plus facile pour les stagiaires
- Fair-play : modularisation des tâches, limites sms, etc.
- Ne plus écrire de scripts de PN, c'est possible !

OLIVE : un bilan (2)

- Reproductibilité (au bit près) de l'opérationnel jamais prise en défaut depuis plus de 7 ans (mille mercis à l'équipe GCO)
- Traçabilité des configurations, dans leurs composants, comme dans leur structure
- Évaluation des chaînes en double sur des périodes plus nombreuses et plus longues
- Un espace de ressources partagé par tout l'établissement (moins de cloisonnement)
- simplification de l'interaction avec les différents systèmes informatiques et facilité du debugging
- utilisation au plus près des ressources informatiques

Des faiblesses

- Pas de transfert aux opérations qui soit quelque peu automatique
- Une trop faible gestion de la cohérence interne des expériences d'un point de vue météorologique
- Peu d'interaction avec l'état du parc informatique : on ne dispose pas d'une véritable grille de calcul
- Extensibilité perfectible de la "toolbox"
- Induit sans doute un manque de recul des utilisateurs sur la réalité du monde "physique" :
  - effet presse-bouton
  - surconsommation ?

4. MTOOL
Multi-Step et plus

Le problème

- Les ressources de calcul sont souvent un agrégat de systèmes
  - machine frontale
  - noeuds de calcul
  - noeuds d'IO
- Certaines séquences d'un job ne sollicitent pas les ressources de calcul
  - par exemple : séquences input / output
  - mais il est toujours commode de considérer le job comme une seule entité logique, dont nombre d'éléments sont communs
  - il serait pénible d'avoir à gérer explicitement l'enchaînement de sous-parties du job, de propager l'information de l'une à l'autre, etc.

Une solution : MTOOL

- Fonctionne comme un filtre en entrée du système de soumission dont il est indépendant (PBS, LoadLeveller)
  - le job initial est instrumenté par l'utilisateur avec des directives non intrusives qui peuvent être ignorées en cas d'utilisation "standard"
  - le job est analysé par le filtre qui produit autant de "step" que définis dans le code source, sans idée a priori de continuité dans celui-ci
    - par défaut le step n°1 est soumis, les suivants s'enchaînent
- Gère les profils multiples, les zones de code communes, les inclusions, la paramétrisation du multi-stepping, un espace de travail partagé, les exceptions, une log unifiée, etc.
- Trace et propage les modification du FS ou de l'ENV
- C'est en fait une façon bien plus propre d'écrire les jobs... même sans découpage par "steps" !

Syntaxe MTOOL

- Simple ligne de commentaire
  - #MTOOL commande [arg=value]...
- Exemples
  - #MTOOL autolog
  - #MTOOL set frontend=mypc
  - #MTOOL step id=fetch target=[this:fronted]
  - #MTOOL common
  - #MTOOL join id=fetch
  - #MTOOL ...

Intermezzo

qrun ? ... en perl naturellement...

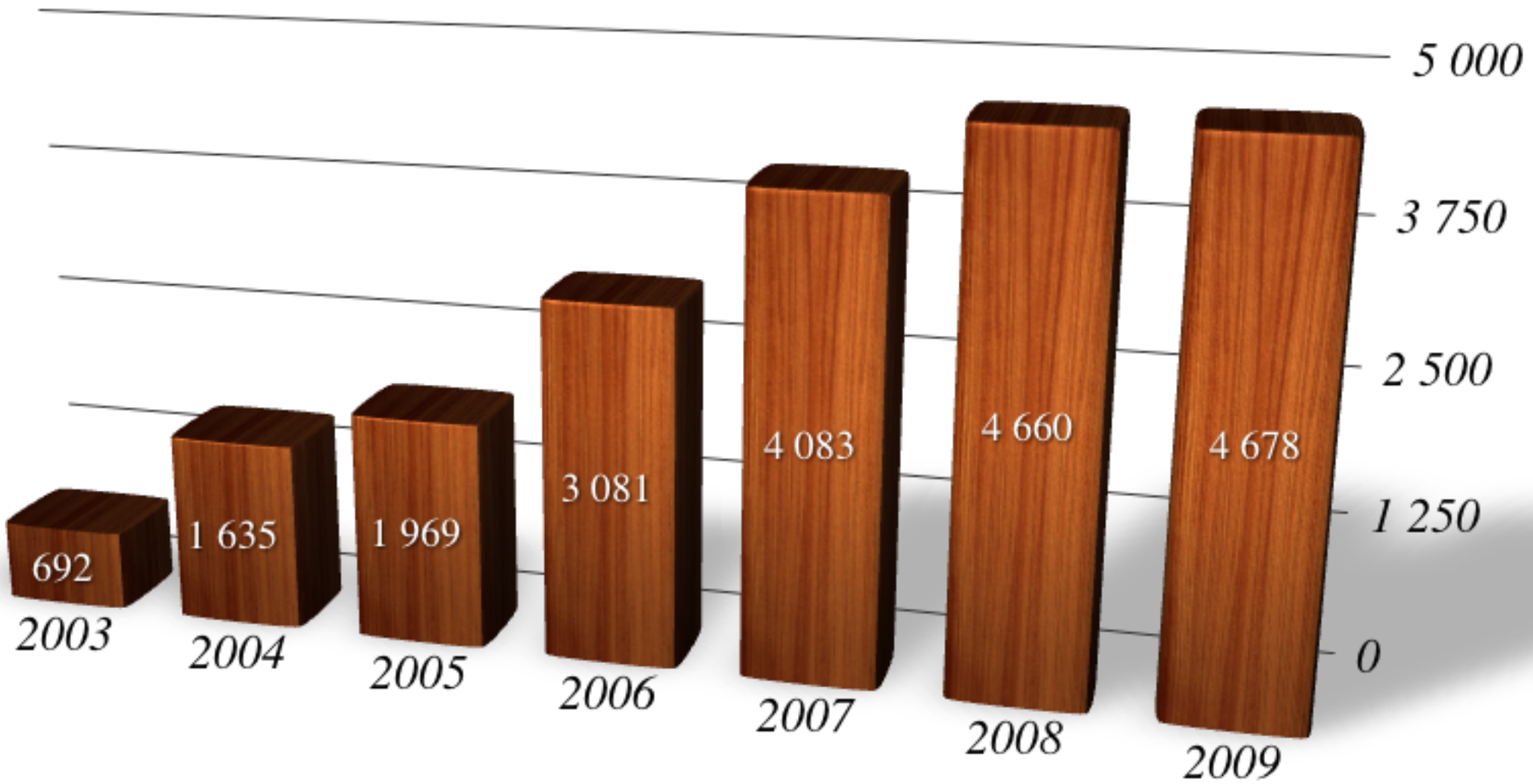
mrpm631@yuki:~/bench/mtool

Fichier Édition Affichage Terminal Aide

```
97506.yuki-batch NEATL36_forecast smer853  multi      0 QUE -    0.00B       0.00        0 Y Y Y    2 yuki06+08
99046.yuki-batch J0_ARM03_9_52419 mrmu285  nocpu      0 RUN -    4.97M       0.02    14717 Y Y Y    1 yuki
1302.yuki-batch  J0_FIRE3_6_64595 mrmu285  nocpu      0 RUN -    4.97M       0.03     2539 Y Y Y    1 yuki
956.yuki-batch   mn@77DI          mrpm627  rapid      0 RUN -  262.26G   12128.46     1768 Y Y Y    1 yuki02
1072.yuki-batch  flp@656Y         mrnp052  rapid      0 RUN -   17.97G    3131.92     3178 Y Y Y    1 yuki09
1084.yuki-batch  mn@77FC          mrpm627  rapid      0 RUN -  262.42G   12048.54     1768 Y Y Y    1 yuki02
1548.yuki-batch  scr@B1PR         mrpe710  rapid      0 RUN -   79.71G    3561.94      868 Y Y Y    1 yuki04
1805.yuki-batch  frc@65D6         mrpm637  rapid      0 RUN -  139.43G    9882.05      688 Y Y Y    1 yuki06
1844.yuki-batch  srf@77HG         mrpa678  rapid      0 RUN -   20.35G    1066.23      598 Y Y Y    1 yuki04
1850.yuki-batch  srf@65D0         mrpm637  rapid      0 RUN -   20.35G    2116.68      568 Y Y Y    1 yuki01
1897.yuki-batch  mn@77EV          mrpa675  rapid      0 RUN -   55.90G    3273.22      478 Y Y Y    1 yuki08
1908.yuki-batch  step.02          mrgm101  rapid      0 RUN -   14.75G     589.00      418 Y Y Y    1 yuki03
1916.yuki-batch  trj@B1PP         mrpe601  rapid      0 RUN -   73.18G     661.85      117 Y Y Y    1 yuki01
1924.yuki-batch  mn@7764          mrpa655  rapid      0 RUN -   26.60G     639.21      358 Y Y Y    1 yuki04
1930.yuki-batch  scr@97LA         mppi012  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki08
1942.yuki-batch  scr@97LA         mppi012  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki04
1945.yuki-batch  scr@97LA         mppi012  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki01
1951.yuki-batch  scr@77GE         mrpa666  rapid      0 RUN -   54.87G     555.87      238 Y Y Y    1 yuki03
1962.yuki-batch  nfl@65G7         mrpm611  rapid      0 RUN -   20.85G      28.54       28 Y Y Y    1 yuki01
1969.yuki-batch  nmx@77FC         mrpm627  rapid      0 RUN -   27.15G      19.00       28 Y Y Y    1 yuki03
1979.yuki-batch  R6HEX044         mrpm608  rapid      0 RUN -   11.54G      80.76       27 Y Y Y    1 yuki03
1994.yuki-batch  AN1P041          mrpm608  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki04
1997.yuki-batch  mn@97LA          mppi012  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki08
2004.yuki-batch  nfl@65G7         mrpm611  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki04
2005.yuki-batch  nfl@65G7         mrpm611  rapid      0 QUE -    0.00B       0.00        0 Y Y Y    1 yuki03
mrpm631@yuki:~/bench/mtool>
```

Super-calculateur yuki (NEC SX9) — Météo France — 24/11/2010 vers 19h

Expériences OLIVE jouées par années



Perl : intermède quantitatif

- En ne comptant que les modules (sans les éventuels scripts d'appel, templates, css, et tout autre enrobage) :
- SWAPP / OLIVE :
  - 572 modules / 40 707 lignes de code
- OLIVE Toolbox :
  - 255 modules / 20 248 lignes de code
- MTOOL :
  - 18 modules / 1623 lignes de code

Nombre de jobs MTOOL traités

- Sur tori (NEC SX8) / depuis 2007 :
  - plus de 3 900 000
- Sur yuki - kumo (NEC SX9) / depuis 2009 :
  - plus de 2 300 000
- Soit un total de plus de 6 200 000 jobs multi-step
  - raisonnablement robuste
  - maintenance quasi nulle
  - hors opérations

5. Plonger dans le VORTEX

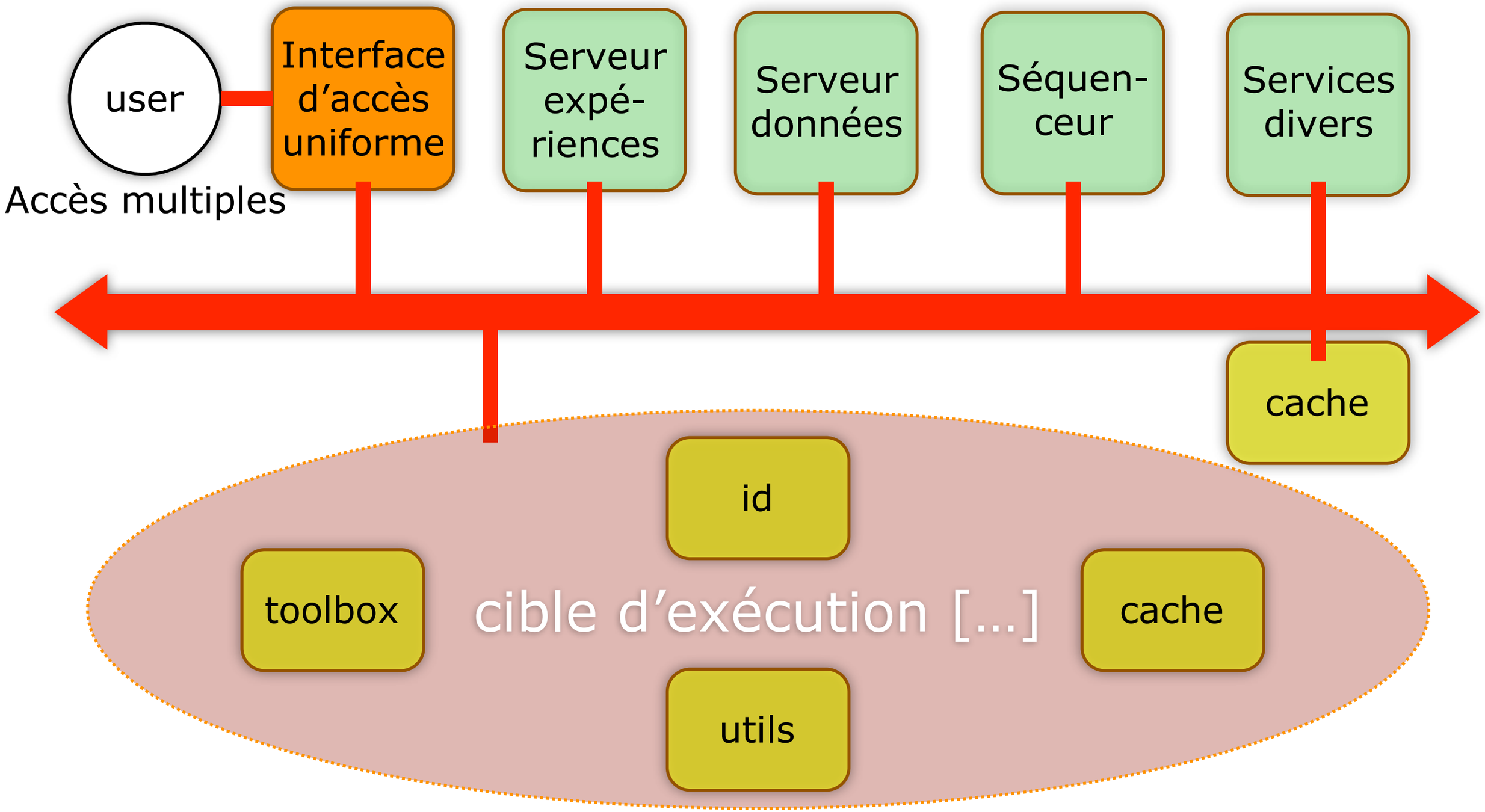
Quelles perspectives de développement ?

- À conserver absolument :
  - Support auto-organisé, disponible et volontaire (paraît aller de soi, mais rarement réalisé) – convivialité du développement informel que permet l'utilisation d'un langage de script évolué
  - Pas de distinction fondamentale entre un "administrateur" et un utilisateur / Les développeurs doivent être les tous premiers utilisateurs du système
  - Accès uniforme et distribué
- Logiciel libre
  - Réutilisation de modules ou couches logicielles éprouvées
  - Développement tiré par les applications des acteurs majeurs de l'internet (google, yahoo, netvibes, etc.)
  - Prendre en compte les choix de la communauté géophysique

Faire évoluer OLIVE

- La séparation entre le noyau SWAPP et la “toolbox” des cibles d’exécutions permet de faire évoluer la seconde tout en gardant le cadre général “olive”
- Cette première étape doit permettre la mutualisation des outils avec les opérations et donner un cadre commun à la majeure partie de la communauté PN
- Approche objet centrée sur la ressource (aussi bien une donnée, un élément de configuration, exécutable, etc.)
- C’est le projet VORTEX :
  - **V**ersatile **O**bjects
  - **R**ounded up in a **T**oolbox
  - for **E**nvironmental e**X**periements

Une vision simplifiée



6. Pourquoi Python ?

Une question récurrente

- Question souvent posée, dès les stades les plus primitifs de la réflexion sur l'évolution des systèmes de scriptage opérationnel ou de recherche
- Le choix de Python repose sur des critères objectifs mais aussi parfois largement subjectifs. C'est le tableau d'ensemble qui finalement permet d'opter pour Python
- Il faut préciser que l'alternative se situait entre Perl et Python uniquement, dans la mesure où ce sont les deux seuls langages de scriptage dont nous avions explicitement exigé la disponibilité tout au long du contrat passé avec NEC (spécification de l'appel d'offre)... ce qui excluait

Quelques éléments de réponse

- La barrière psychologique qui peut parfois exister au regard de la synthaxe Perl, n'existe pas pour le code Python, plus clair, plus structuré, sans doute moins ludique mais qui tend à imposer plus de rigueur ;
- L'évolution vers Perl 6 commence vraiment à traîner en longueur, induisant le sentiment, peut-être injustifié au demeurant, d'un langage en légère "perte de vitesse" ;
- Le mode interactif de Python fait partie intégrante de Python et n'est pas une excroissance comme le psh (Perl shell) plus ou moins commode ; ce caractère natif de l'interactif python est dans le cas de VORTEX une fonctionnalité essentielle ;
- L'interopérabilité avec le C, le C++ et le Fortran semble plus facile avec Python ;

Quelques éléments de réponse (suite)

- Le calcul numérique, y compris haute performance, peut s'appuyer sur des modules standards largement adoptés par la communauté : NumPy et SciPy ;
- Tout ce qui relève des communications dans tous les protocoles possibles et imaginables, ainsi que les éléments fondamentaux pour l'écriture de clients et serveurs, se trouve réuni dans une boîte à outils elle aussi largement intégrée dans nombre de développements : Twisted ;
- La communauté en sciences géophysiques a déjà largement basculé sur Python et développé des modules de haut niveau ;
- Les jeunes éléments ayant rejoint ces dernières années le CNRM, quand ils développent dans un langage de script évolué, le font souvent en Python, notamment pour ce qui concerne le post-traitement et les tracés de graphiques ;

Python en usage

- Scriptage simple
  - enchaînement de programmes, réseau
- Conversions
  - XML et variantes : KML, GPX, GDAL
  - NETCDF & al. : Pupynere, nappy
- Calcul scientifique & stats
  - NumPy, SciPy, MetPy, CDAT... et sortir du FORTRAN avec f2py
- Graphiques & Web
  - matplotlib, mapserver, basemap, CDAT
  - django et bindings Magics++, gribapi,

Conclusions

Les langages de script

- Langages de haut niveau
- Rapidité de mise en oeuvre
- Abstraction
- Logique naturelle (Cf. Perl / Larry Wall)
- Applications de plein droit
- Portabilité et continuité logicielle (un environnement de développement commun du PC local au super-calculateur)
- Richesse des bibliothèques de modules

Conclusions

- D'un point de vue théorique, la PN, et plus généralement la recherche en météorologie est essentiellement un processus collectif, qui synthétise les avancées de nombreux chercheurs dans diverses équipes
- Sa mise en œuvre pratique, peut également bénéficier de cette qualité. Elle permet d'envisager les aspects pratiques sous un angle plus motivant, voire ludique
- Le scriptage Perl a pleinement participé de cet état d'esprit, dans le processus de développement et dans la communauté qui s'est créée sur l'utilisation des outils développés
- Espérons que malgré des apparences plus austères, le passage à Python ne signifiera pas que l'on siffle la fin de la récréation !

Remerciements

- Pour des contributions, idées, discussions, encouragements :
  - MF : Florence Rabier, Véronique Mathiot, Philippe Marguinaud, Guillaume Beffrey, Eric Gerbier, Patrick Moll, Thomas Auligné, Jean Pailleux, Pierre Brousseau, Ryad El Khatib, Pascal Lamboley, Marc Tardy, Joël Stein, Jacques Anquetil, Marion Pithon, Serge Stamatiou, Vivianne Rey, Vivien Pourret, JAM, et tous les sympathiques beta testers permanents : Betty Gérard, Fatima Karbou, Nadia Fourié, Cécile Loo, Gaëlle Kerdraon, Françoise Taillefer, Gérald "hpcd" Desroziers, Loïk Berre, François Bouyssel, Yves Bouteloup, Thibaut "badluck" Montmerle, Claude "3d" Fischer, Karim Yessad, Jean Nicolau, Yann Michel, Ludovic Auger, Yann Seity, Vincent Guidard et beaucoup d'autres...
  - ECMWF un jour ou l'autre : Otto Pesonen, Baudouin Raoult, Sami Saarinen, Tuomo Kauranne, Nils Wedi, Claes Larsson, Jean-NöelThépaut, Claude Gibert, Paul Burton, Dominique Lucas.
  - NEC (ou ex-) : Daniel, François, JJ, Olivier