GUIDE UTILISATEUR

BIOMAJ v1.1

Auteurs : David Allouche (allouche@toulouse.inra.fr)**, Olivier Filangi** (olivier.filangi@irisa.fr)

Remerciements à Christophe Caron, Olivier Collin et Hugues Leroy pour leurs contributions.

Site web : http://biomaj.genouest.org/

Date de création de ce document: 07/05/09

Table des matières

1 Introduction....5
1.1 Pourquoi BioMAJ ?....5
1.2 Historique....5
1.3 Présentation....6
2 Mise en place de BioMAJ....8
2.1 Contenu du livrable....8
2.1.1 Banques disponibles....8
2.1.2 Indexations / conversions disponibles....9
2.2 Pré requis....9
2.2.1 Système & matériel :....9
2.2.2 Applications utilitaires :....10
2.2.2.1 Logiciels généralement présents sur une installation standard Linux :....10
2.2.2.2 Logiciels nécessitant une attention particulière :....10
2.3 Installation :....11
2.3.1 Initialisation des variables d'environnement....11
2.3.2 Compilation....11
2.3.3 Initialisation de la variable d'environnement BIOMAJ_ROOT....12
2.3.4 Variables globales de l'application....12
2.3.5 Configuration générale : fichier global.properties....13
2.3.6 Première exécution de BioMAJ....13
2.3.7 Démonstrations....14
2.3.7.1 Banque « alu » du NCBI (avec processus formatdb) :....14
2.3.7.2 Banque « sts » du NCBI (avec processus fastacmd)....14
2.3.8 Paramétrage des répertoires utilisé par BioMAJ....14
2.3.9 Utilisation d'un Proxy....15
3 Utilisation de l'application....16
3.1 Comportement général de l'application....16
3.1.1 Le cycle de mise à jour....16
3.1.2 Organisation des données....18
3.2 Maintenance des données....19
3.2.1 Mise à jour d'une banque....19
3.2.2 Effacer une ou des versions d'une banque....23
3.2.3 Changer le nom d'une banque....23
3.2.4 Changer le répertoire de production de la banque....23
3.2.5 Reconstruction d'une version....24
3.2.6 Reprise sur erreur....24
3.2.7 Importation de données....25
3.2.8 Maintenance des fichiers d'état....26
3.3 Automatisation des mises à jour :....26
3.4 Supervision de l'entrepôt....28
3.4.1 Alerte Mail : suivi de session de mise à jour....28
3.4.2 Déverminage, recherche d'erreurs à l'exécution :....31
3.4.3 Exploration en ligne de commande : biomaj.sh --status....32
3.4.3.1 Exploration de l'entrepôt:....32
3.4.3.2 Exploration de l'état d'une banque:....33
3.4.4 Rapport Html....35

3.4.4.1 Génération du rapport....35
3.4.4.2 Consultation / déploiement du rapport....35
3.4.4.3 Structure du rapport....35
3.4.4.3.1 Pages Html....36
3.4.4.4 Statistiques....37
3.4.4.4.1 Statistiques globales....37
3.4.4.4.2 Statistiques par banque....38
4 Création d'un workflow....40
4.1 Généralités....40
4.2 Rappel de la Session de mise à jour d'une banque....40
4.2.1 Configuration de la source : description / classification / localisation....41
4.2.1.1 Description et classification....41
4.2.1.2 Localisation des données....41
4.2.2 Configuration de la phase de Synchronisation des données....41
4.2.2.1 Serveur distant et protocole de téléchargement....42
4.2.2.1.1 Le protocole ftp :....42
4.2.2.1.2 Le protocole http :....43
4.2.2.1.3 Le protocole rsync:....44
4.2.2.1.4 Le protocole local :....45
4.2.2.2 Construction des filtres de téléchargement (remote.files/local.files)....45
4.2.2.3 Obtenir un numéro de version....47
4.2.2.4 Consolidation des données....49
4.2.3 Configuration de la phase de pré-processing et post-processing....50
4.2.3.1 Fonctionnement général :....50
4.2.4 Définition des éléments....51
4.2.4.1 Bloc....51
4.2.4.2 Méta-processus....52
4.2.4.3 Processus....52
4.2.5 Exemple de workflow à partir des scripts disponibles dans BioMAJ....55
4.2.6 Déploiement....57
4.3 Surcharge d'information : global.properies....58
5 Développement / intégration de post ou pré traitements....60
5.1.1 Gestion de la communication : passage de messages....60
5.1.2 Gestion des dépendances de fichiers produits....61
5.1.3 Passage de contexte au script : Information sur le numéro de version....62
5.1.4 Code retour....63
5.1.5 Debugage....63
5.2 Notions diverses....63
5.2.1 Banque virtuelle....64
5.2.2 Banque calculée....64
5.2.2.1 Principe d'une banque calculée....64
5.2.2.2 Illustration : NR....64
5.2.2.3 Paramètres spécifiques aux workflow de banques calculées....65
6 Méta-donnés et classification des sources et des traitements :....68
6.1 Définition....68
6.2 Implémentation BioMAJ....68
6.3 Utilisations potentielles:....69
7 F.A.Q....70

8 Annexes....73
8.1 Exemple de Fichiers de configuration....73
8.1.1 Global.properties....73
8.2 Tableaux des propriétés Biomaj....74

1 Introduction

1.1 Pourquoi BioMAJ ?

Les données en biologie (domaine en « omique[1] ») sont un pré-requis indispensable à l'analyse bio-informatique. Elles forment une masse de données très volumineuse, de plusieurs Tera-octets, distribuée sur différents sites internationaux, et sous des formats hétérogènes (plusieurs standards existent à ce jour).

La nature et le nombre de ces banques sont en constante évolution, et la fréquence de leur mise à jour est variable. Enfin, dans la plupart des cas, elles nécessitent, avant leur utilisation locale, de multiples reformatages pouvant être extrêmement coûteux en temps de calcul (plusieurs jours par exemple pour le calcul des index SRS). Le suivi des mises à jour et des résultats des conversions de formats produisent en général des fichiers de traces (journalisation) qu'il faut analyser en cas d'erreurs (lecture d'un mail journalier de reporting, suivie éventuellement d'une analyse des fichiers de trace). La gestion de ces données est un travail lourd et fastidieux. Gestionnaire et utilisateurs des données peuvent parfois s'y perdre…

1.2 Historique

Le projet BioMAJ est né de la collaboration en 2005 de trois équipes : l'INRIA de Rennes , l'INRA de Toulouse et de Jouy-en-Josas. A l'époque aucune application gratuite ne répondait aux besoins. L'application la plus proche était citrina développée par Josh Goodman (Indiana University). Ce prototype était prometteur - toutefois encore loin de l'application recherchée - et malheureusement cet outil n'évoluait plus depuis 2004.

En 2006, trois équipes INRIA et INRA entreprennent le développement d'un nouveau moteur baptisé BioMAJ[2]. Sur la base de citrina 0.51, la quasi totalité du code a été réécrite, l'architecture de l'application et ses fonctionnalités ont été complètement repensées et considérablement étendues.

Courant 2007, l'application a été testée en grandeur réelle sur les trois sites porteurs du projet afin de la rendre robuste et adaptée au contexte de la bio-informatique à grande échelle. Elle est déployée en production sur les trois sites impliqués dans le projet.

[1] Génomique, protéomique, transcriptomique, métabolomique

[2] **Bio**logie **M**ise **A** **J**our

1.3 Présentation

BioMAJ propose :

- un moteur de workflow fiable capable de télécharger des données distantes d'une façon automatique et intelligente (reprise sur erreur, synchronisation de données locales et distantes), d'appliquer des formatages sur ces données et de les mettre en production (rendre disponible ces données pour un ensemble d'utilisateurs et/ou d'applications).
- Un ensemble de workflows pré-définis pour les principales banques biologiques.
- Une bibliothèque de scripts d'indexation (formatage pour des données biologique) à appliquer sur les workflows définis par l'administrateur ou sur les workflows pré-définis.

Principales caractéristiques fonctionnelles de l'application :

- Gestion des cycles de mise à jour d'une source de données ;
- Journalisation des workflows ;
- Contrôle de l'intégrité des données transférées ;
- Normalisation de l'organisation des données (arbre de répertoire normalisé) ;
- Gestion d'une classification des sources de données (selon la nature, la provenance, les index d'applications ...) ;
- Gestion de l'entrepôt de données local, consultation des versions disponibles et gestion des versions des sources (ajout, délétion de version et de source) ;
- Administration de chaque banque grâce à un ensemble de commandes;
- Reprise sur erreur, c'est-à-dire une reprise d'un cycle non terminé ;
- Reconstruction d'une version (release) donnée ;
- Description d'un workflow de post-traitement relativement complexe ;
- Supervision de l'application via des Alertes Mail, positionnées à des points clés du workflow de chaque source. Ce qui permet notamment lors d'utilisation de crontab d'alerter l'administrateur du bon ou du mauvais fonctionnement des sessions ;
- Suivi de l'entrepôt de données, génération de statistiques sur la composition de l'entrepôt de données. Suivi de l'évolution d'une source au cours du temps. Édition de rapports de synthèse web contenant une vue des données de l'entrepôt ainsi qu'un historique de son évolution source par source.

BioMAJ est un logiciel d'aide à la maintenance d'un entrepôt de données et a pour objectif d'assister l'administrateur des données dans sa démarche. BioMAJ peut gérer une grande masse de données hétérogènes, et permet de consolider localement un ensemble de sources distribuées et disponibles sur le réseau via des protocoles normalisés comme ftp, http ou rsync, puis d'automatiser des chaînes de traitements plus ou moins complexes sur ces données. L'application automatise le cycle de mise à jour et facilite la supervision du catalogue des banques de données gérées, tout en

assurant un historique des opérations de maintenance. L'application permet d'assurer la maintenance sur une plate-forme locale des principales banques de données biologiques mises à disposition par la communauté scientifique internationale. Le champ d'application du logiciel BioMAJ est toutefois plus large, et pourrait être étendu à tout domaine ayant à gérer des données massives et distribuées.

BioMAJ confère une souplesse dans la gestion des banques de séquences sur un site en autorisant la mise en place rapide de nouveaux workflows par simple création d'un fichier de description d'une banque.

2 Mise en place de BioMAJ

2.1 Contenu du livrable

- `bin/biomaj.sh` : exécutable de BioMAJ
- `bin/Make_biomaj_report.sh` : exécutable pour la génération de rapports
- `bin/clean_statefiles.sh` : exécutable pour nettoyer les données de workflow
- `bin/env.sh` : variables d'environnement à définir pour les traitements
- `conf/process` : scripts de traitement
- `conf/db_properties` : répertoire des fichiers propriétés utilisés par BioMAJ
- `conf/db_properties/samples` : exemple de fichier propriétés sans post-traitement.
- `conf/db_properties/samples-extended` : exemple de fichiers de propriétés avec post-traitement (indexation).
- `doc` : documentation BioMAJ
- `src` : sources
- `sbin` : utilitaires annexes
- `workflows` : scripts ant permettant l'exécution des mises à jour

2.1.1 Banques disponibles

Pour fonctionner, l'application a besoin d'un fichier de configuration contenant la description du workflow associé à chaque source de données. Ce fichier est appelé fichier « properties » de la banque.

La construction d'un fichier de propriétés n'est pas difficile, néanmoins il s'agit d'une tâche fastidieuse, qui nécessite avant tout une bonne connaissance de la source de données et de l'application BioMAJ. La validation du fichier « properties » n'étant pas tout à fait immédiate, nous proposons dans la distribution BioMAJ, des exemples de fichiers de configuration pour plusieurs banques courantes. Ils sont localisés dans les répertoires :

`$BIOMAJ_ROOT/conf/db_properties/workflow_withprocess`

et

`BIOMAJ_ROOT/ conf/db_properties/workflow_withoutprocess.`

Nous proposons de mutualiser les fichiers « properties ». Si vous en développez pour des banques particulières, et si vous souhaitez les partager vous pouvez les poster sur la liste : BioMAJ-users@lists.gforge.inria.fr

Après validation, ces fichiers seront intégrés aux prochaines versions du logiciel.

Concernant la validation, un post-processus, ou fichier « properties », devra pour être validé, être testé avec succès sur au moins deux sites utilisant BioMAJ. Une documentation minimale devra être fournie. Elle comprendra un synopsis du script, ainsi qu'une description de la procédure d'installation : pré-requis, variables d'environnement, configuration des applications utilisant les données issus du script.

Quelques exemples sont cités dans le paragraphe ci-après.

2.1.2 Indexations / conversions disponibles

En standard quelques post-processus sont mis à disposition avec le moteur. En voici la liste :

- Fastacmd : `conversion de banque blast en fichier fasta`
  `$BIOMAJ_ROOT/doc/process/fastacmdTLSE.html`

- Formatdb : `indexation d'une banque fasta pour ncbi blast`
  `$BIOMAJ_ROOT/doc/process/formatdbTLSE.html`

- SRS : `indexation SRS`
  `$BIOMAJ_ROOT/doc/process/indexSrsTLSE.html`

Autre process:

- SendMail : Script d'envoi d'alerte mail.
  `$BIOMAJ_ROOT/doc/process/sendMailTLSE.html`

Chacun de ces scripts dispose d'une aide de configuration personnalisée dans le répertoire *doc/process*. Notez que dans la plupart des cas, il est nécessaire d'adapter le workflow à votre configuration.

2.2 Pré requis

2.2.1 Système & matériel :

BioMAJ a été développé et testé massivement sur des architectures à base de processeurs AMD ou Intel X86 ou X86-64 et sous Linux. Néanmoins, compte tenu des technologies logicielles utilisées (java et ant), son utilisation ne devrait pas poser de problèmes pour une grande variété d'ordinateurs et de systèmes d'exploitation, incluant Mac OS, les versions les plus populaires d'Unix (BSD, Solaris, AIX, ...).

Les limites principales de compatibilité OS de l'application ne sont pas dûes au moteur mais aux post-processus utilisés. En effet les applications d'indexation ne sont pas forcément compatibles ou disponibles toutes les plates-formes.

L'utilisation de BioMAJ sous Windows est plus hasardeuse. Cela reste à vérifier, mais au prix de quelques modifications à faire dans les configurations, l'application devrait pouvoir être portée sous ces environnements, notamment grâce à l'émulateur cygwin.

BioMAJ est une application généraliste, elle est susceptible de synchroniser des sources de données de taille et de nature variables. Lors de son utilisation dans le contexte de la biologie, l'application doit effectuer des téléchargements massifs et des post traitements extrêmement lourds. L'architecture matérielle hébergeant l'application doit être dimensionnée en conséquence. L'application fonctionne sur un simple portable, mais pour une exploitation pleine et entière (récupération des données et indexations) des worksflows présents dans la distribution, **un multi-processeur (bi ou quadri) et un espace disque de 1 à 2 To sont fortement recommandés**.

Notez qu'idéalement, un espace de stockage centralisé accessible via une grappe de machines (cluster) peut permettre à BioMAJ de superviser la réalisation d'indexations plus rapides. Dans ce cas, un déport de calcul via rsh, ssh, ou un ordonnanceur quelconque comme Torque, Pbs, Sge, Lsf, Condor … est nécessaire. L'ensemble des machines de calcul doit avoir la capacité d'écriture sur un espace de stockage partagé. Pour y parvenir plusieurs solutions sont envisageables : utiliser NFS, un système de fichiers clusterisé, un NAS, un système de fichier réparti, etc. **Ces solutions sont relativement complexes à mettre en oeuvre, mais elles sont le prix à payer pour la gestion optimale de données à grande échelle.**

2.2.2 Applications utilitaires :

Ce chapitre contient les logiciels pré-requis à l'installation du moteur de l'application :

La liste des logiciels pré-requis se subdivise en deux sous parties :

2.2.2.1 Logiciels généralement présents sur une installation standard Linux :

- Wget 1.6 ou supérieur (http://www.gnu.org/software/wget/wget.html)
- GNU Tar 1.13 ou supérieur (http://www.gnu.org/software/tar/tar.html)
- Gzip 1.3.3 ou supérieur (http://www.gzip.org/)
- Bzip2 0.9.0 ou supérieur (http://sources.redhat.com/bzip2/)

2.2.2.2 Logiciels nécessitant une attention particulière :

- Java 1.6.x (http://java.sun.com/j2se/)
- Ant 1.6.5 or higher(http://ant.apache.org/)

 **Cette liste ne comprend pas les pré-requis nécessaires à chaque post-processus.**

En effet, chaque post traitement appelé par le moteur, fait généralement appel à un ou des utilitaires qui lui sont propres.

Exemple : Pour les besoins de la démo, l'installation du ncbi toolkit blast est nécessaire en plus des logiciels des listes précédentes. En effet les indexations font appels aux post-processus de formatage blast et fasta. Autre exemple, le post-processus d'indexation SRS nécessite en pré-requis l'installation de SRS et une configuration ad hoc de l'application (cf. doc/process/srsdb.i). Plus généralement, chaque post-process, comprend une documentation dédiée située dans le répertoire /doc/process/

2.3 Installation :

2.3.1 Initialisation des variables d'environnement

Pour fonctionner les variables ANT_HOME et JAVA_HOME doivent être initialisées, les exécutables ant et java doivent être accessibles dans le PATH.

Si votre shell est sh, bash ou ksh :

Modifiez votre fichier .bashrc situé dans votre HOME directory (ou celui de BioMAJ, si vous avez créé un utilisateur BioMAJ) en vous inspirant des lignes suivantes :

```
export ANT_HOME=/usr/local/ant/apache-ant-1.6.5
export JAVA_HOME=/usr/local/java
export PATH=$PATH:${ANT_HOME}/bin:${JAVA_HOME}/bin
```

Si votre shell est csh ou tcsh :

Dans votre fichier ~/.cshrc ou votre ~/.tcshrc adaptez les lignes ci-dessous :

```
setenv ANT_HOME /usr/local/ant/apache-ant-1.6.5
setenv JAVA_HOME /usr/local/java
setenv  PATH ${PATH}:${ANT_HOME}/bin
setenv PATH ${PATH}:${JAVA_HOME}/bin
```

2.3.2 Compilation

Avant compilation, l'archive doit être décompactée. Si vous lisez cette documentation vous venez très probablement de réaliser la commande :

```
tar xvfpz BioMAJ-${version}.tar.gz
```

ou

```
gunzip < BioMAJ-${version}.tar.gz | tar xvfp -
```

Et dès que ce désarchivage a été réalisé, pour compiler l'application il vous suffit de saisir les commandes suivantes :

```
cd BioMAJ-${version}
ant
```

En cas de compilations successives, nous vous recommandons de nettoyer les fichiers intermédiaires produits par les compilations précédentes, par l'utilisation de la commande (à exécuter dans le répertoire racine de BioMAJ) :

```
ant clean
```

2.3.3 Initialisation de la variable d'environnement BIOMAJ_ROOT

Pour fonctionner, l'application nécessite l'initialisation de la variable BIOMAJ_ROOT, qui doit contenir le nom du répertoire racine de l'arborescence de l'application.

Pour l'initialisation, et selon votre environnement, suivez les instructions suivantes :

- en bash, sh, ksh :

```
export BIOMAJ_ROOT=<répertoire racine de BioMAJ>
```

- en csh ou tcsh :

```
setenv BIOMAJ_ROOT <répertoire racine de BioMAJ>
```

2.3.4 Variables globales de l'application

Pour pouvoir exécuter les post-processus, les applications ont bien souvent besoin de variables d'environnements spécifiques.

Le moteur BioMAJ utilise le script `$BIOMAJ_ROOT/bin/env.sh` pour positionner les variables d'environnement nécessaires à l'exécution des post-processus (indexation, mail, etc...).

Ce script se révèle particulièrement utile lorsque l'administrateur de l'application n'a pas les droits suffisants pour pouvoir initialiser les variables au niveau global du système, ou lorsqu'il souhaite différencier son environnement de travail usuel de l'environnement d'exécution de l'application.

Exemple de script d'initialisation de l'environnement srs ou blast :

```
#!/bin/sh
export BLASTDB=/bank/blastdb #initialisation de la variable
                             #BLASTDB
.     /data/srs/srs/etc/prep_srs.sh     #initialisation     de
l'environnement
                                  #nécessaire à SRS
```

2.3.5 Configuration générale : fichier global.properties

Le fichier : `$BIOMAJ_ROOT/conf/db_properties/global.properties` contient un bon nombre de variables internes de l'application BioMAJ et qui sont déjà pré-initialisées dans ce fichier. (cf. section écriture de nouveaux workflows)

Pour plus d'information reportez vous à la section fichier global.properties

Avant d'exécuter BioMAJ (script $BIOMAJ_ROOT/bin/biomaj.sh), vous devez initialiser la racine de votre repository local de données.

- **Dans le fichier `$BIOMAJ_ROOT/conf/db_properties/global.properties`, l'initialisation de la variable data.dir est absolument nécessaire pour l'utilisation de BioMAJ**

2.3.6 Première exécution de BioMAJ

Deux exemples de banques vous sont fournis à des fins de démonstration. Il s'agit des banques « alu » et « sts » du ncbi.

Pour chacune, deux scenarii de workflows peuvent être utilisés :

- scénario sans post-process
- avec post-processus (indexation blast pour alu et conversion fasta pour sts)

Si vous souhaitez utiliser les workflows de démonstrations, prenez soin de bien positionner ces fichiers de propriété dans le répertoire `$BIOMAJ_ROOT/conf/db_properties/`

L'utilisation d'un scénario de démonstration avec post-processus nécessite d'avoir à disposition les exécutables contenus dans le package blast du ncbi. Ce dernier est disponible à l'url : ftp://ftp.ncbi.nih.gov/blast/executables/LATEST.

Avant utilisation, vous devez également mettre à jour les chemins des exécutables appelés dans le fichier : `$BIOMAJ_ROOT/conf/process/unix_command_system.cfg`

```
FASTACMD=/MY/CHEMIN/fastacmd
FORMATDB=/MY/CHEMIN/formatdb
```

Pour plus d'information sur la configuration et le fonctionnement des post-processus disponibles consulter le répertoire : `$BIOMAJ_ROOT/doc/process`

Remarques : La liste des paramètres utilisables dans les fichiers properties de vos banques est disponible dans la section annexe du présent document.

2.3.7 Démonstrations

2.3.7.1 Banque « alu » du NCBI (avec processus formatdb) :

Fichier de configuration : `$BIOMAJ_ROOT/conf/db_properties/alu.properties`

Pour ne pas exécuter l'indexation blast (formatdb), il faut commenter dans le fichier de configuration la ligne suivante (commenter la ligne en plaçant un caractère "#" en début de ligne) :

```
db.post.process=POST1
```

Lancer la commande : `$BIOMAJ_ROOT/bin/biomaj.sh --update alu --console`

2.3.7.2 Banque « sts » du NCBI (avec processus fastacmd)

Fichier de configuration : `$BIOMAJ_ROOT/conf/db_properties/sts.properties`

Pour ne pas exécuter l'indexation blast (formatdb), il faut commenter dans le fichier de configuration, la ligne suivante (caractère "#" en début de ligne):

```
db.post.process=POST1
```

Lancer la commande : `$BIOMAJ_ROOT/bin/biomaj.sh --update sts -console`

2.3.8 Paramétrage des répertoires utilisé par BioMAJ

La localisation des différents répertoires utilisés par BioMAJ :

- génération de la journalisation
- définitions des workflows (fichier de propriété)
- fichiers d'état des banques en production
- localisation des processus de post-traitement
- génération d'un rapport html

Ainsi que les différents outils utilisés par BioMAJ se trouve dans le fichier de configuration : `$BIOMAJ_ROOT/general.conf`. Ce fichier est généré automatiquement à la première utilisation de BioMAJ.

```
# Created by BioMAJ 0.9.2.1
# Date : 03/01/2008 14:36:30
# File : General configuration
[DIRECTORIES]
log.dir        =/home/biomaj/log
log-biomaj.dir =/home/biomaj/log/biomaj-runtime
statefiles.dir =/home/biomaj/statefile
workflows.dir  =/home/biomaj/conf/db_properties
process.dir    =/home/biomaj/conf/process
webrepport.dir =/home/biomaj/rapport

[APPLICATIONS]
```

```
tar.bin=/bin/tar
gunzip.bin=/usr/bin/gunzip
bunzip.bin=/usr/bin/bunzip2
unzip.bin=/usr/bin/unzip
wget.bin=/usr/bin/wget
```

2.3.9 Utilisation d'un Proxy

BioMAJ supporte les serveurs proxy à base de SOCKS4 et SOCKS5.

Si vous utilisez un proxy, vous devez définir les propriétés suivantes dans le fichier `$BIOMAJ_ROOT/conf/db_properties/global.properties` :

- proxyHost
- proxyPort

Pour un proxy avec identification vous devez également redéfinir ces propriétés :

- proxyUser
- proxyPassword

BioMAJ utilise l'outil Wget, pour son bon fonctionnement vous devez définir des variables d'environnement dans le fichier `$BIOMAJ_ROOT/bin/env.sh`

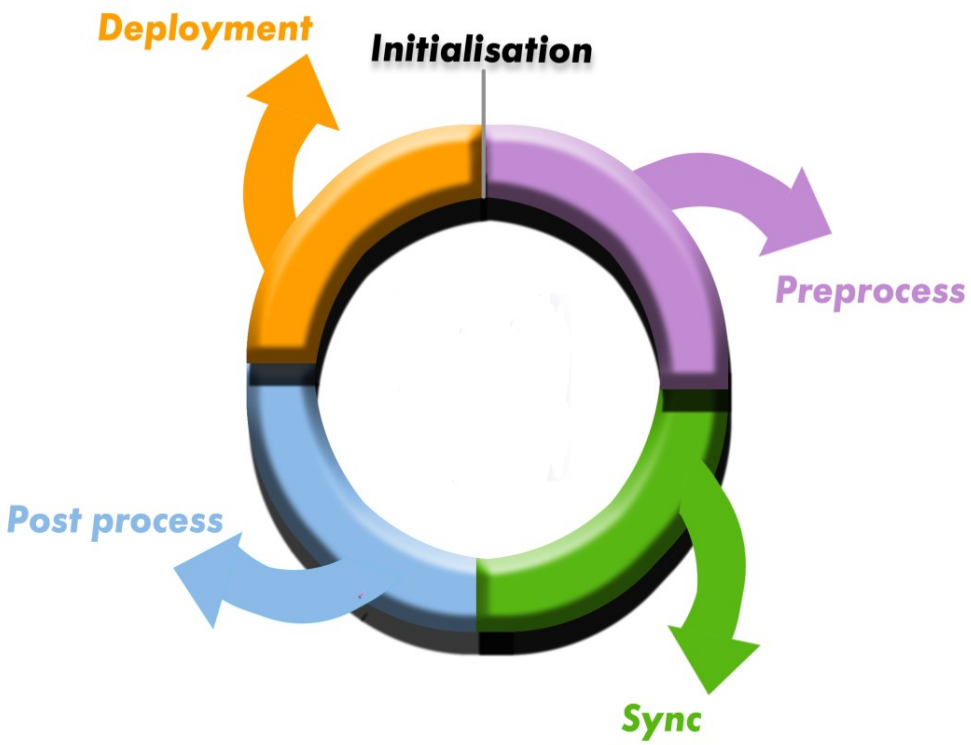
- `http_proxy http://login:mdp@[host]:[port]`
- `ftp_proxy http://login:mdp@[host]:[port]`
- `proxy on`

3 Utilisation de l'application

3.1 Comportement général de l'application

3.1.1 Le cycle de mise à jour

Dans le présent paragraphe, le cycle et les étapes qui le constituent sont décrits succinctement.



BioMAJ est conçu pour réaliser les cycles de mises à jour d'une source de données. Chaque cycle comprend 5 étapes.

1. **L'initialisation :**

   Le moteur charge le fichier de propriétés contenant la description du workflow et regarde l'état courant de la banque en parcourant le fichier d'état associé. Après détermination de l'état de la banque, l'application ouvre le cycle proprement dit ou si nécessaire essaye de compléter le précédent cycle non terminé (dans le cas de reprise sur erreur).


Une source donnée a un fichier associé nommé nécessairement *{bankname}*.properties (exemple pour la banque nr : nr.properties). L'exécution du fichier de propriétés décrivant le workflow (nr.properties) va générer ou alimenter un fichier d'état traçant le déroulement du workflow (date d'exécution des sous-taches, trace des fichiers manipulés...).
Le fichier de propriétés est localisé dans le répertoire désigné par la propriété *workflow.dir* du fichier de configuration *general.conf* (par défaut `$BIOMAJ_ROOT/conf/db_properties`).
Le fichier d'état est généré au format xml dans le répertoire désigné par la propriété *statefiles.dir* du fichier de configuration *general.conf* (par défaut `$BIOMAJ_ROOT/statefiles/`).

2. Pré processing

Il s'agit d'un sous-workflow exécuté en amont de la mise à jour de données. Il a les mêmes propriétés que la partie post-processing décrite ci-après. Son but est simple. Il permet de déclencher des tâches, des contrôles, des alertes préalables au reste du processus de mise à jour.

3. Synchronisation

Lors de cette étape le moteur se connecte sur la source, vérifie la présence de nouvelles données par rapport aux données déjà présentes localement. Il détermine la liste des fichiers à télécharger, attribue un nom de version. Il effectue ensuite le téléchargement, puis la décompression. Il consolide enfin les données pour produire une version complète de la banque, conformément aux contraintes définies dans le fichier de propriétés.

4. Post-processing

Lors de cette étape, le moteur exécute le sous-workflow de post traitement. Le formalisme permet de décrire des workflows relativement complexes. Il s'agit d'une succession de blocs de tâches, qui contiennent un ou des sous ensembles de méta-tâches pouvant elles-même être constituées de plusieurs traitements. Chaque bloc est exécuté séquentiellement. Dans un bloc donné, les méta-tâches qui le constituent sont exécutées parallèlement. Enfin dans une méta-tâche, les traitements sont exécutés à nouveau séquentiellement. En cas d'erreur sur un traitement, seule la branche à laquelle il appartient est arrêtée. L'ensemble décrit un graphe acyclique dirigé (D.A.G).

5. Déploiement

Si les étapes précédentes se sont déroulées sans erreurs, l'application met en ligne la nouvelle version de la source. Puis elle efface les versions obsolètes, ainsi que les fichiers temporaires produits lors de l'exécution des post-processus.

L'ensemble des étapes de la session d'exécution est consigné dans le fichier d'état. En cas d'erreur, lors de la session suivante, l'application va tenter de reprendre l'exécution à la première étape erronée de la session précédente, afin de compléter le cycle. Un seul cycle est associé à une version d'une source donnée. Une ou plusieurs sessions peuvent

au gré de l'exécution être nécessaires pour compléter un cycle.

3.1.2 Organisation des données

Lorsque BioMAJ gère des sources des données, l'application produit par défaut une arborescence pour chacune d'elle. En cas d'absence, les répertoires sont créés lors la première session de mise à jour. Un exemple d'arborescence type est décrit dans la figure ci-dessous.

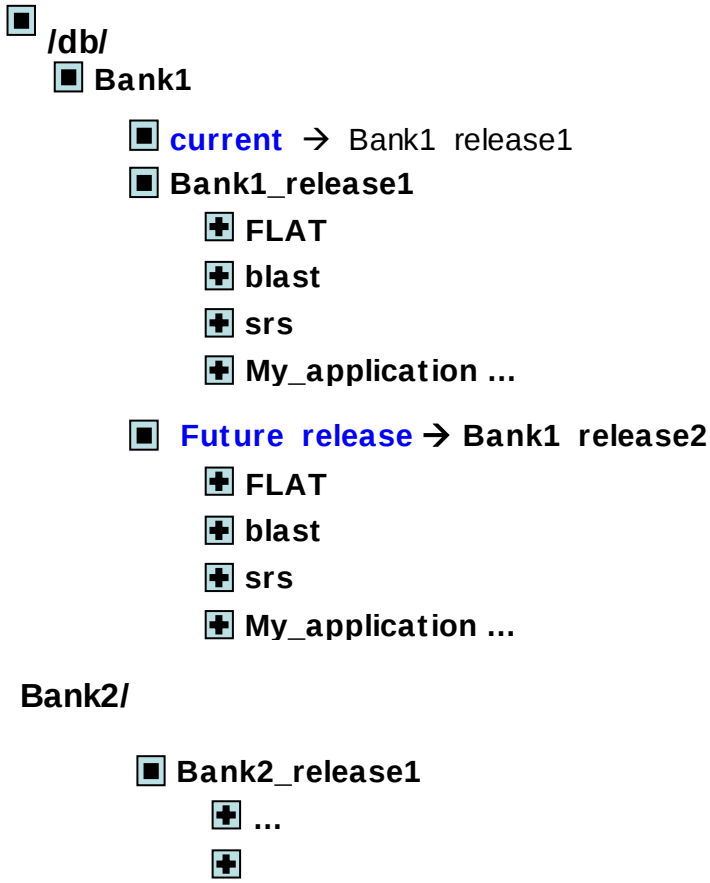


Illustration 1: Organisation du répertoire de production

L'arborescence de BioMAJ s'organise autour d'une architecture à 4 niveaux :

- `/db` : Il s'agit du chemin où seront localisées par défaut l'ensemble des sources maintenues par l'application (c'est le nom du répertoire racine de production des banques défini par la propriété ***data.dir***) .
- `/db/Bank[n]` : l'ensemble des banques biologiques en production sur le site. Ce deuxième niveau est définit par la propriété ***dir.version*** (redéfini pour chaque workflow), si celle-ci n'est pas définie, cette propriété prend par défaut la valeur de ***db.name*** (le nom de la source).
- `/db/Bank[n]/Bank[n]_release[p]` : chaque version téléchargée est placée dans un répertoire « version ». Le nom de ce répertoire prend par défaut la valeur : ***db.name_release***. Le nombre de release conservé dépend de la propriété ***keep.old.version***.

- `/db/Bank[n]/ Bank[n]_release_[p]/flat` : chaque version a une organisation identique. Les fichiers téléchargés ont été déplacés dans un répertoire flat après leur décompression éventuelle. Au même niveau, nous pouvons trouver plusieurs répertoires, chacun d'eux correspondant aux résultats des post-traitements d'indexations appliquées à la version considérée.
- `/db/Bank[n]/current` : Pour chaque banque, un lien current est positionné automatiquement par BioMAJ lors du déploiement d'une nouvelle version de la banque.
- `/db/Bank[n]/future_release` : De même, un lien future_release est positionné sur le répertoire d'une version en court de construction.

Les deux liens « current » et « future_release » ont un nom invariant. Ils sont donc stables dans le temps. Nous recommandons de les utiliser dans la configuration des applications qui réalisent les indexations (lien future_release) ou qui utilisent les données (lien current).

Néanmoins, les utilisateurs qui réalisent des traitements très longs sont invités à utiliser les chemins absolus des versions de banque. Ainsi en cas de mise à jour des données, ils ne seront pas impactés par le changement de positionnement du lien « current » alors que leurs calculs se poursuivent.

3.2 Maintenance des données

3.2.1 Mise à jour d'une banque

Pour la première mise à jour d'une banque vous avez deux possibilités : utiliser un fichier de configuration (description du workflow) déjà défini du répertoire :

`$BIOMAJ_ROOT/conf/db_properties/workflow_withprocess`

ou

`$BIOMAJ_ROOT/conf/db_properties/workflow_withoutprocess`

ou bien créer un nouveau workflow (reportez vous au chapitre Création d'un workflow).

Pour que le moteur BioMAJ puisse prendre en compte un nouveau workflow, un fichier properties doit nécessairement figurer dans le répertoire :

`$BIOMAJ_ROOT/conf/db_properties/`

Pour exécuter le workflow en mode console :

`$BIOMAJ_ROOT/bin/biomaj.sh --update [bank name] -console`

Le mode console est pratique pour visualiser l'exécution d'un workflow. Nous vous recommandons de l'utiliser ponctuellement pour vérifier le comportement d'un fichier de propriétés.

Exemple avec la banque alu :

```
$BIOMAJ_ROOT/bin/biomaj.sh --update alu --console
```

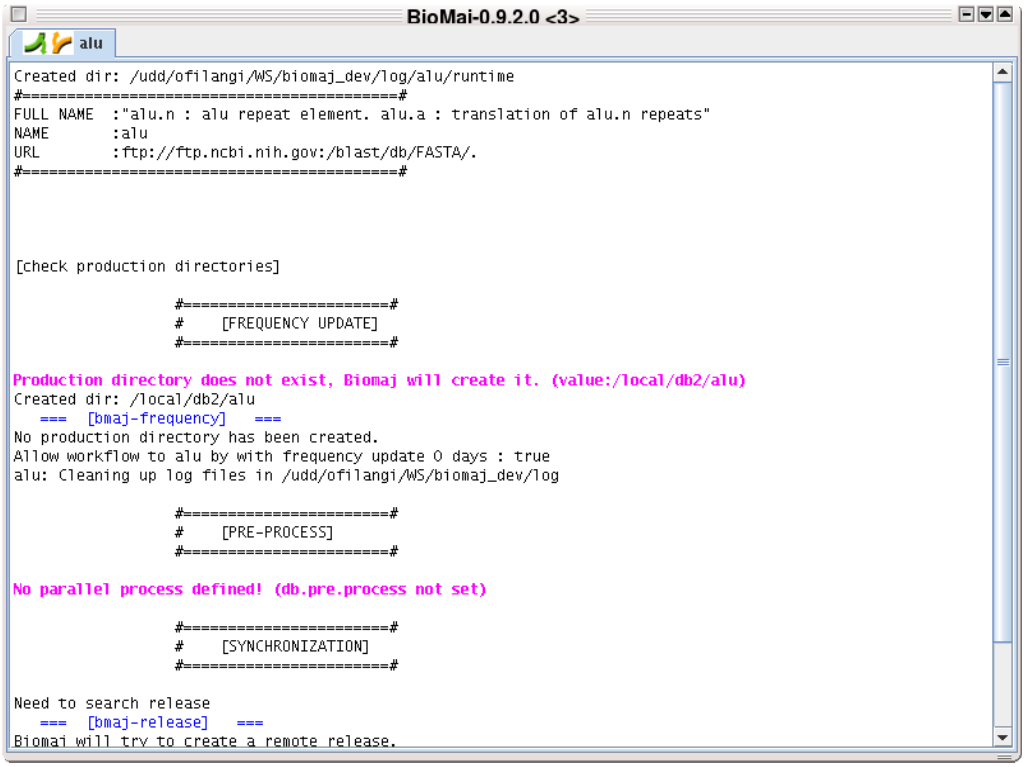


Illustration 2: Console d'une première mise en ligne de la banque alu

BioMAJ crée les répertoires « temporaire » et « production » s'ils n'existent pas. Dans cet exemple, remarquons qu'il n'y a pas de pré-processing.

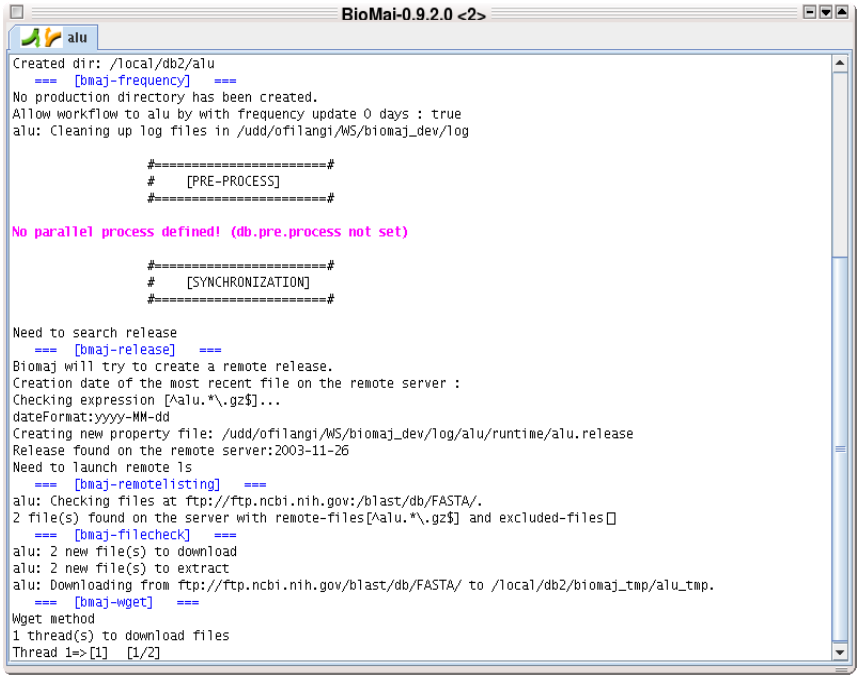


Illustration 3: Console - phase de synchronisation pour la banque alu

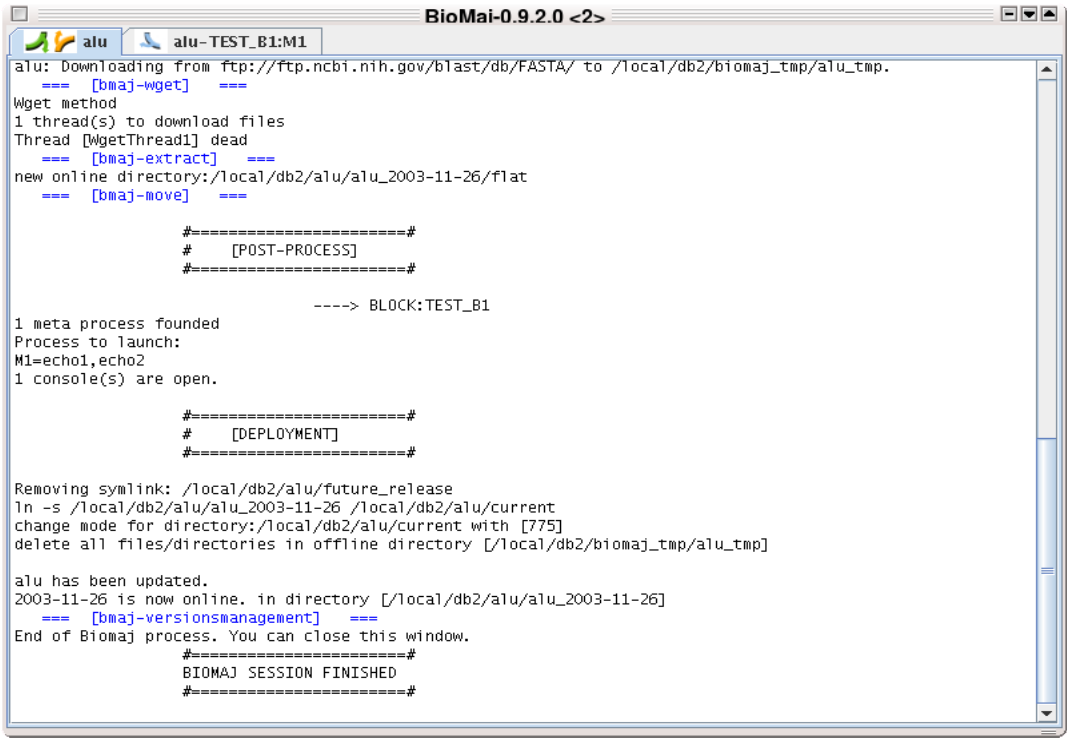


Illustration 4: Console – fin de phase de synchronisation et de déploiement pour la banque alu

Comme on peut le voir dans les illustrations ci-dessus (cf illustration 2 et 3), la phase de synchronisation se subdivise en 7 sous tâches : release ; remotelisting ; filecheck ; wget ; extract ; versionsmanagement ; move

- Dans un premier temps la tâche « **release** » permet de récupérer le numéro de version (ou par défaut l'application lui affecte la date du fichier le plus récent de la version) ;
- Ensuite la tâche « **remotelisting** » de BioMAJ détermine les éléments (fichiers, répertoires) constitutifs de la version distante ;
- Ils sont ensuite comparés aux fichiers locaux lors de la tâche **filecheck**. Dans l'exemple aucun fichier n'existe en local ;
- Ils sont récupérés lors de la phase **wget** de téléchargement ;
- Les fichiers téléchargés sont décompressés lors de la **tâche extract** ;
- Puis le répertoire de la nouvelle version est créé (tâche **versionsmanagement**) . Enfin les fichiers sont déplacés dans le futur répertoire de production lors de la tâche **move** (dans l'exemple : /local/db2/alu/alu_2003-11-26/flat) .

Suite à cette phase, une phase de post processus est exécutée dans un nouvel onglet (cf illustration 4).

En cliquant sur cet onglet vous obtenez les traces des post-processus (ici un simple echo). Chaque tâche **bmaj-execute** représente l'exécution d'un processus.

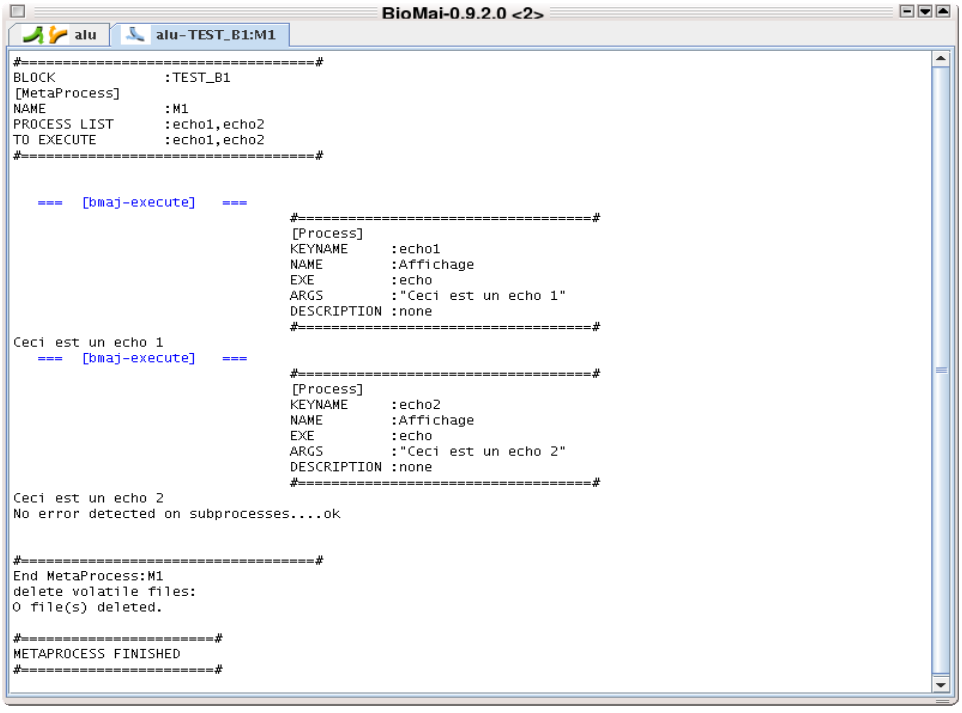


Illustration 5: Phase de post-processus

Lorsque la phase de post-processus est terminée, on peut suivre le déroulement de fin du workflow (tâche deployment) en cliquant sur le premier onglet «alu ». La tâche de déploiement crée

un lien symbolique « current » sur le répertoire de production qui vient d'être créé, modifie les droits d'accès pour rendre accessible ce répertoire aux utilisateurs et efface le contenu du répertoire temporaire de alu (fichiers générés mais non utilisés en production).

3.2.2 Effacer une ou des versions d'une banque

BioMAJ conserve un nombre de versions défini par l'affection du paramètre **keep.old.version**. Par défaut ce paramètre est égal à zéro (cf. global.properties). Le moteur gardera au plus une version en production et une version en cours de réalisation. Le paramètre **keep.old.version** peut être surchargé dans le fichier properties d'une banque quelconque. Il ne sera pris en compte que pour la source en question.

Il n'est pas rare que l'entrepôt de données soit parfois au gré de l'exploitation à la limite de la saturation. Il est alors nécessaire de faire le « ménage » assez rapidement.

Pour enlever une banque dépendante de la gestion de BioMAJ, la commande -- remove est :

```
$BIOMAJ_ROOT/bin/biomaj.sh --remove [bankname] --keep-dir-prod [true|false]
```

Cette option:

- Demande quelle version vous désirez effacer
- Puis efface le répertoire de trace `$BIOMAJ_ROOT/log/[bankname]`
- les fichiers d'états : `$BIOMAJ_ROOT/statefiles/bankname.xml`, `$BIOMAJ_ROOT/statefiles/bankname/*.xml`
- elle conserve les données des répertoires de productions si l'option `--keep-dir-prod` est utilisée.
- elle lance le remove process pour chaque version supprimée (see 4.2.4).

3.2.3 Changer le nom d'une banque

Il est impératif d'utiliser une commande de BioMAJ si l'on souhaite garder l'historique des sessions et répertoires de production et changer le nom d'une banque.

```
$BIOMAJ_ROOT/bin/biomaj.sh --change-dbname [name] [newName]
```

L'utilisateur n'a pas besoin de redéfinir la propriété db.name.

3.2.4 Changer le répertoire de production de la banque

Il y a deux façon de modifier le répertoire de production d'une banque.

- Modification de la propriété data.dir du fichier global.properties (Toutes les banques en production auront leur répertoire de production modifié)
- Modification de la propriété version.dir du fichier de propriété de la banque ([name].properties)

Exécution de la commande :

```
$BIOMAJ_ROOT/bin/biomaj.sh –move-production-directories [bankname]
```

 Si les propriétés version.dir ou data.dir sont modifiées, vous ne pourrez plus exécuter de mise à jours. Vous devez impérativement exécuter la commande –move-production-directories.

3.2.5 Reconstruction d'une version

Il est possible de faire marche arrière et de reconstruire spécifiquement, la dernière version d'une banque donnée.

Exemple *:* `$BIOMAJ_ROOT/biomaj.sh –rebuild Mybank`

Cette commande est interprétée de deux façons différentes :

- **Si localement aucune version de la source n'est disponible :**

i) L'application change l'état de la banque courant (version N), qui passe de l'état « online » à l' état « updating ». Pour ce faire le lien symbolique « current » est effacé et un lien « future_release » est repositionné sur le répertoire de la version.

ii) Les données autres que les données brutes sont effacées

iii) Un cycle de mise à jour est ré-exécuté. Ce qui aura pour conséquences - si les données n'ont pas évolué sur le serveur distant - de ré-exécuter les post-traitements sur les données brutes puis de refaire un déploiement.

- **Si localement au moins deux versions de la source sont disponibles :**

Dans ce cas, lors de l'étape i) citée plus haut, BioMAJ réalisera en plus des opérations déjà décrites, la remise en ligne de la version N-1, pendant que le traitement de la version N est réalisé. Concrètement, le lien current est repositionné sur la version N-1 , et le lien future_release sur la version N.

3.2.6 Reprise sur erreur

BioMAJ intègre différents modes de reprise sur erreur, en général ce processus est transparent à l'utilisation. En contexte de reprise, la commande utilisée est la même que celle d'une mise à jour c'est-à-dire :

- `biomaj.sh –d MyBank`

Cette dernière va provoquer une nouvelle session, qui tentera de compléter le dernier cycle en l'occurrence « non clôs» pour la ou les banque(s) en argument. Le comportement de l'application est fonction du contexte de l'erreur.

- Si une erreur s'est produite au court de l'étape de synchronisation (coupure réseau , arrêt du serveur distant), lors de la nouvelle session, après avoir re-construit la liste des fichiers inclus dans la version, l'application va scanner les données présentes localement et tenter de poursuivre le cycle .

- Si une erreur ponctuelle a lieu lors du téléchargement d'un fichier, plusieurs tentatives sont réitérées avant arrêt sur erreur du téléchargement. L'arrêt complet de la session sera effectif après le traitement de l'ensemble des fichiers de la liste.

Après chaque téléchargement, un contrôle d'intégrité est réalisé. Il comprend la vérification des attributs locaux du fichier (nom, taille, date) contre les mêmes attributs sur le serveur distant originel.

- Si une erreur se produit pendant l'étape de post-traitement, lors de la reprise BioMAJ reprendra l'exécution au premier traitement ayant un statut en erreur.

Dans le cas des post-traitements, les erreurs constatées sont souvent dues à des erreurs système. Il est important de préciser que si le traitement en faute n'émet pas un retour erreur (différent de zéro pour les shell scripts) lors de l'interruption erratique, BioMAJ ne le détectera pas et poursuivra la session jusque la mise en ligne. Cependant pour aider à la détection de ce type de problème, si la taille de la nouvelle version est inférieure à la précédente un warning sera émis par l'application.

Les données sont associées à une version, cette version est fonction du serveur distant. Si une version est incomplète localement alors que le serveur distant met à jour les données, la version locale ne pourra jamais être complétée. Dans ce cas, il sera nécessaire d'ouvrir un nouveau cycle de mise à jour, car le cycle courant est impossible à terminer.

Pour résoudre ce cas d'utilisation, l'option –N ou –new a été développée :

- `biomaj.sh --update <dbname> --new`

Cette commande aura pour effet de créer un nouveau cycle.

Cette commande ne doit être utilisée qu'en dernier recourt, après l'analyse de la situation (lecture des fichiers de trace, vérification du serveur distant,etc...) si vous avez la certitude que le cycle courant ne pourra pas être complété.

3.2.7 Importation de données

L'historique de la gestion d'une source est contenu dans le fichier d'état de la banque. A chaque session un fichier *.bak* est sauvegardé afin d'éviter sa perte. Néanmoins si par erreur, les versions du fichier sont détruites, BioMAJ perd alors toutes les informations concernant les données associées à la banque dans l'entrepôt local. Pour réintégrer les données et reprendre le contrôle de la banque une commande permettant de réimporter des données a été développée.

Un exemple de l'utilisation de cette commande est ci-dessous:

- `biomaj.sh --import nr`

Dans cet exemple, la commande produira le fichier d'état suivant :

`$BIOMAJ_ROOT/statefile/nr.xml`.

L'utilisation de cette commande implique que les données soient en cohérence avec le fichier de propriétés de la banque et que les données du répertoire de production doivent avoir une organisation « conforme » à l'arborescence BioMAJ :

La propriété **dir.version** doit être positionnéd sur le répertoire de production de la banque à importer. Ce répertoire doit contenir un répertoire de version (*dbname_version*) et un lien current pointant sur ce répertoire. Le répertoire de version doit contenir un répertoire flat contenant l'ensemble des données sources du serveur.

`conf/db_properties/nr.properties` (cf. section Organisation des données)

Il est également important de noter que :

- Seules les données contenues dans le répertoire flat (données brutes) de la version « current » seront importées.
- En présence de plusieurs versions, les versions antérieures à la dernière ne seront pas importées.

3.2.8 Maintenance des fichiers d'état

Le fichier statefile contient l'historique de la source. Notamment pour les banques contenant des milliers de fichiers, sa taille peut avec le temps devenir très importante. Ce qui peut provoquer des ralentissements lors l'utilisation de certaines commandes de BioMAJ. Pour pallier ce problème, une solution simple consiste à nettoyer le fichier statefile des informations inutiles.

Ainsi, la commande :

- `biomaj.sh --clean-statefile <dbname>`

va effacer les balises inutiles du statefile de la banque (dbname) placée en argument. Il s'agit des balises <files> des versions obsolètes ayant été effacées de l'entrepôt local.

Elle efface également les cycles "vides", c'est à dire les sessions qui se sont bien déroulées mais qui n'ont pas donné lieu à la génération d'une nouvelle version de la source.

Enfin elle efface les fichiers xml des traitements associés aux versions obsolètes, c'est à dire les "release" qui ont été effacées à la suite à mises à jour successives.

3.3 Automatisation des mises à jour :

L'automatisation des mises à jour peut être réalisée via une *crontab*. Vous devez à priori connaître la fréquence de mise à jours des sources dont vous voulez automatiser la maintenance.

Cron est un utilitaire Unix qui permet à un utilisateur d'exécuter des commandes à des intervalles de temps donnés.

Un exemple de *crontab* est disponible dans le répertoire : `$BIOMAJ_ROOT/misc/`

La mise en place de la *crontab* doit être réalisée sous l'identité de l'utilisateur BIOMAJ - i.e le login sous lequel est censée fonctionner l'application.

Les fichiers de profil des utilisateurs (.cshrc par exemple) ne sont pas lus avant l'exécution des commandes par cron. Cela peut entraîner des comportements étranges des scripts, parfois difficiles à comprendre pour un néophyte.

Pour éviter ce probleme, avant son utilisation, vous devez initialiser les variables d'environement prérequises pour le fonctionnement de BioMAJ.

Un récapitulatif des variables d'environnement nécessaires est présenté ci-aprés :

```
#définition du shell utilisé à l'ouverture de chaque session cron
SHELL=/bin/sh
# positionnement du répertoire de stockage des log générés
```

lors des exécutions cron :

```
HOME=$BIOMAJ_ROOT/log/cron
# positionnement de  la variable BIOMAJ_ROOT :
BIOMAJ_ROOT=/MY/RACINE/BIOMAJ
# définition de l'environnement java :
JAVA_HOME=/usr/local/java/jdk1.6.0
CLASSPATH=/usr/local/java/jdk1.6.0/dt.jar:.
# définition de l'environnement ant :
ANT_HOME=/usr/ant/apache-ant-1.6.5
```

- Pour metter en place la crontab vous devez taper la commande suivante :

- `crontab ./BioMAJ_crontab`

- Pour éditer le contenu d'un crontab, utilisez la commande :

- `crontab -e`

Cette commande ouvrira dans un éditeur de texte, le fichier *crontab* `BioMAJ_crontab`.

Pour chaque banque, il est très important de positionner la date de mise à jour locale, en fonction celle du site distante. Sinon vous prenez le risque de ne pouvoir jamais finir un cycle de mise à jour. La raison est simple, les fichiers, que vous comptez récupérer, seront effacés alors que vous êtes en train de les télécharger.

Si par erreur vous positionnez la session BioMAJ à la même heure que la mise à jour des données distantes, BioMAJ dispose de mécanisme de contrôle qui vous protégera de la construction de version chimérique. Une chimère est une version « corrompue » constituée pour partie de fichiers de la version passée et pour une autre partie complémentaire de fichiers de la nouvelle version de la source.

- Pour rajouter une mise à jour :

Il suffit de la rajouter dans le fichier crontab. Chaque ligne correspond à une commande à exécuter. Les lignes ont le format suivant :

```
moment          commande
```

Les commandes sont tout simplement celles que l'utilisateur taperait s'il les exécutait lui-même.

Un moment se décompose en 5 champs :

champ	valeurs possibles
minute	0-59
heure	0-23
jour du mois	1-31
mois	1-12
jour de la semaine	0-7 (0 et 7 pour dimanche) pour Linux. Pour les autres Unix, consulter le manuel pour la signification des chiffres.

Chaque champ est séparé par un espace et plusieurs éléments du même type sont séparés par des virgules (sans espace). Une étoile (*) signifie : "pour tous". Exemple sous Linux :

```
0,10,20,30,40,50 * 1 * * /usr/local/BioMAJ/bin/biomaj.sh –d gen-banknews
```

Cette ligne signifie :
« Exécute /usr/local/BioMAJ/bin/biomaj.sh –d genbanknews toutes les 0, 10, 20... minutes de toutes les heures du premier jour de chaque mois ».

En général on n'indique que les éléments qui sont pertinents et on met une * pour les autres. Vous obtiendrez la syntaxe complète d'une crontab à l'aide de la commande :

- `man –s 5 crontab`

Remarque : un alternative élégante à l'initialisation des variables, consiste à exécuter un script spécifique avant le lancement de l'exécutable biomaj.sh.

Exemple :

```
0,10,20,30,40,50 * 1 * 7 $HOME/.bashrc ;
/usr/local/BioMAJ/bin/biomaj.sh –d genbanknews
```

3.4 Supervision de l'entrepôt

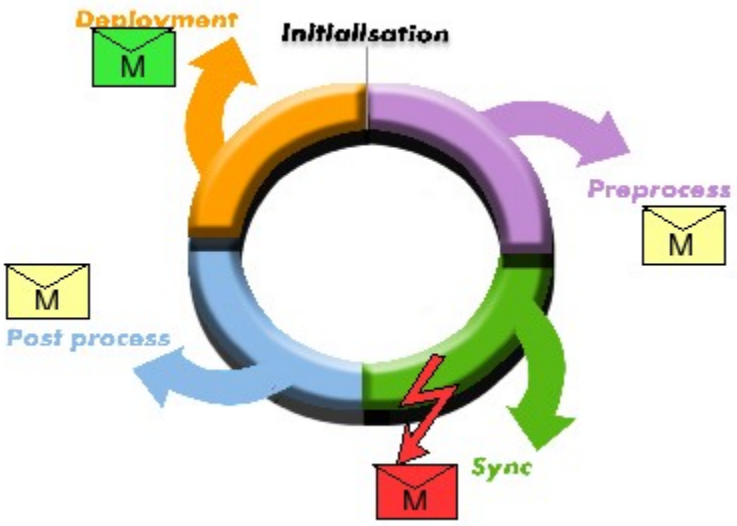
BioMAJ dispose de fonctionnalités permant de superviser son fonctionnement. Qu'entend-on par supervision ?

Cela recouvre plusieurs aspects à différentes échelles :

- Connaître le contenu de l'entrepôt ;
- Suivre l'évolution d'une banque ;
- Suivre le bon ou mauvais déroulement d'un cycle de mise à jour.

Comme nous le verrons dans cette section, trois modes d'accès aux informations sont proposés. Selon la nature de l'information, elle peut être obtenue : par mail, en mode console ou enfin grâce à votre navigateur favori via un rapport html.

3.4.1 Alerte Mail : suivi de session de mise à jour



A chaque fin de session de BioMAJ, un rapport est envoyé par courriel (e-mail) à l'administrateur des données. Les coordonnées du destinataire sont déclarées par défaut dans le fichier `global.properties`.

Il s'agit des propriétés : **mail.smtp.host**, **mail.admin** et **mail.from**

Comme leur nom le suggèrent, il s'agit respectivement de l'adresse du serveur de mail, l'adresse du destinataire, et afin l'adresse de l'expéditeur.

Le mail expédié reprend les informations relatives à la version mise à jour ainsi que les différentes étapes du cycle, les warnings et les erreurs.

Son champ « Subject » a été structuré pour qu'il puisse être facilement filtré par votre logiciel de consultation du courrier électronique.

Ce sujet est une chaîne de caractères variable composée des champs suivants :

```
BioMAJ message:BANK[$dbname]-STATUS[$status]-UDPATE[TRUE|FALSE]-$Release
```

Le sujet est donc associé à la banque ainsi qu'aux résultats de la session. Une session sans erreurs, aura un champ *STATUS[TRUE]* alors qu'une session problématique aura un *STATUS[FALSE]* dans son intitulé. Enfin, si la production de la session est une version, la chaîne : *UPDATE[TRUE]* suivi du nom de la version seront concaténées à la suite de l'objet du mail.

Deux exemples de rapports mails sont présentés ci-dessous :

Date: Thu, 27 Sep 2007 18:35:34 +0200 (CEST)
De: biomjtest@toulouse.inra.fr
À: dataman@toulouse.inra.fr
Objet: Biomaj message: BANK [human_genomic] - STATUS [TRUE]- UPDATE [TRUE] RELEASE:2007-10-01

Start :05-10-2007 13:04:10

```
End   :05-10-2007 14:02:04

****************** INFO RELEASE ***************
Number of session                 :1

Production directory :/db/ncbi/blast/human_genomic/human_genomic_2007-10-01
Release                         :2007-10-01
Download                      :1,995G
Bandwidth (Mo/s)       :3.1467621
Num files downloaded :3
Release                         :4,474G
------------
Processes:
Metaproc:[POST1] log:
[/db/biomaj/Bin/log/human_genomic/20071005130410/postprocess.POST1.log]
 --> fastacmd(Create Fasta File) --> sendMail(mail)

*********************************************
SESSION:
--------
LOG:/db/biomaj/Bin/log/human_genomic/20071005130410/mirror.log
----------- GLOBAL ERROR  ----------------------
---------------------------------------

----------- GLOBAL WARNING --------------------

----------- ERROR ON SUB-TASK -----------------------
            *** preprocess ***
            *** release ***
            *** check ***
            *** download ***
            *** extract ***
            *** addLocalFiles ***
            *** makeRelease ***
            *** postprocess ***
WARNING:  POST1::fastacmd_hg : the  environment  variable  BLASTDB  is not
set. Default value is /db/blastdb.
            *** deployment ***
---------------------------------------
```

Illustration 6: Exemple de message Alerte mail produit lors d'une session BioMAJ correcte avec mise à jour d'une version.

Date: Thu, 27 Sep 2007 18:35:34 +0200 (CEST)

De: biomjtest@toulouse.inra.fr

À: dataman@toulouse.inra.fr

Objet: *BioMAJ message: BANK [alu] - STATUS [FALSE]*

```
Start :27-09-2007 18:16:56
End   :27-09-2007 18:17:31
SESSION:
--------
LOG:/home/allouche/BioMAJweb/BioMAJ_dev/log/alu/20070927181656/mirror.log
----------- GLOBAL ERROR  ----------------------
ERROR : BioMAJ stopped.
See the logs to obtain more information.
-----------------------------------------
----------- GLOBAL WARNING --------------------
----------- ERROR ON SUB-TASK -----------------------
            *** preprocess ***
            *** release ***
            *** check ***
            *** download ***
            *** extract ***
WARNING:  extract: no match to be decompressed !
            *** addLocalFiles ***
            *** makeRelease ***
            *** postprocess ***

ERROR:    POST1::cptest : /bin/cp: cannot stat
`/bank/ncbi/blast/alu/future_release/flat': No such file or directory
ERROR:    POST1::cptest : /bin/cp: cannot create regular file
`/home/allouche/bank/ncbi/blast/alu/futur_release/testfasta/test.fa': No such
file or directory
ERROR:    POST1::cptest : "Process cptest (with executable cptest.exe) generate
an error."
read   log  files to obtain  more information.
-----------------------------------------
```

Illustration 7: Exemple de message Alerte mail produit lors d'une session BioMAJ erratique

3.4.2 Déverminage, recherche d'erreurs à l'exécution :

On peut obtenir les traces des sessions via :

- Le rapport html (voir Génération de rapport Html)
- Consultation des fichiers :
  - `$BIOMAJ_ROOT/log/[bankname]/[datesession]/mirror.log` : initialisation, synchronisation, déploiement
  - `$BIOMAJ_ROOT/log/[bankname]/[datesession]/[preprocess|postprocess]-[groupe-process].log` : log sur un exemple de process définis pour le post-processing ou le post-processing (voir La phase de pré-processing et post-processing)


Vous pouvez obtenir des informations de manière continue sur l'état d'un workflow, en exécutant dans un shell la commande :

```
> tail -f log/[bank]/[date]/mirror.log
```

3.4.3 Exploration en ligne de commande : biomaj.sh --status

3.4.3.1 Exploration de l'entrepôt:

- La commande : *biomaj.sh --status* liste les sources gérées par l'application.

Si la liste est relativement longue, pour aider son utilisation cette commande dispose en option de différents filtres qui permettent d'extraire de l'ensemble du catalogue un sous ensemble.

Plusieurs filtres sont disponibles :

--dbtype <dbtype>	Dans la liste globale filtre les sources dont le "dbtype" sera égal à l'argument passé.
--online	filtre dans la liste global, les source qui ne sont pas en cours de mise a jours .
--updating	filtre dans la liste global, les source qui ont un cycle de mise a jours ouvert.

Au final la commande –status peut être utilisée comme suit :

- `biomaj.sh -S --dbtype genome`

```
DbType          DbName                        Last release       Date Session Status
-------------------------------------------------------------------------------------
genome          Anopheles_gambiae             2007-06-01         15-06-2007   online
                Apis_mellifera                 23-01-2007         21-03-2007   online
                Arabidopsis_thaliana          2007-04-24         28-04-2007   online
                Bacteria                      2007-08-10         22-08-2007   online
                Bos_taurus                    2007-04-25         11-05-2007   online
                Caenorhabditis_elegans         16-02-2006         21-03-2007   online
                Canis_familiaris               23-01-2007         21-03-2007   online
                D_rerio                        01-03-2007         21-03-2007   online
                Drosophila_melanogaster        23-01-2007         21-03-2007   online
                Fungi                          2007-07-12         20-08-2007   online
                Gallus_gallus                  23-01-2007         21-03-2007   online
                H_sapiens                      2007-04-17         20-04-2007   online
                M_musculus                     2007-07-05         21-08-2007   online
                Macaca_mulatta                 23-01-2007         23-03-2007   online
                Monodelphis_domestica          06-03-2007         23-03-2007   online
                Pan_troglodytes                02-03-2007         23-03-2007   online
                Plasmodium_falciparum          2007-07-24         16-08-2007   online
                R_norvegicus                   23-01-2007         21-03-2007   online
                Saccharomyces_cerevisiae       2007-08-13         20-08-2007   online
                Schizosaccharomyces_pombe      05-03-2002         23-03-2007   online
                Strongylocentrotus_purpuratus 23-01-2007          02-04-2007   online
                Tribolium_castaneum            25-01-2007         23-03-2007   online
```

Illustration 8: Résultat de la commande status

Dans l'exemple de sortie présenté ci-dessus, on peut voir un filtrage "dbtype" qui a pour resultat l'affichage des sources maintenues dans le catalogue local classifié "genomes".

Le tableau de sortie comporte cinq champs :

- dbtype : le champ de classification de source

- dbname: le nom générique donné à la source

- Last release: le nom de la derniere version de la source

- Session Date: date de la mise à jour

–**Status**: le statut de la version (online ou updating selon le cas : si le dernier cycle est ouvert, ou fermé)

 Notez qu'Il est possible de combiner les filtres : Ainsi

- `biomaj.sh –S --updating -dbtype=genome`

donnera la liste des genomes en cours de mise à jour.

3.4.3.2 Exploration de l'état d'une banque:

Nous venons de voir que le contenu de l'entrepôt peut être obtenu via la commande:

- `$BIOMAJ_ROOT/bin/biomaj.sh --status`

On peut obtenir des informations détaillées sur chaque banque maintenue par BioMAJ. via la commande :

- `$BIOMAJ_ROOT/bin/biomaj.sh --status [bankname]`

Lorsqu'un nom de banque est précisé en argument, l'option --status affiche quatre blocs de données : le premier contient les attributs de la configuration (fichier de propriétés défini par l'utilisateur), le second comprends les informations associées à la version courantes, le troisième est relatif à la version en cours de mise à jours. Le quatrième bloc liste les versions en production de la banque.

 Les blocs « current release », « future release » et « list of production directories » sont facultatifs. Leur présence est conditionnée au contenu de l'entrepôt de données local.

Un exemple de sortie est présenté dans le cadre suivant, il est relatif à une banque disposant d'une version en production (current) et d'une version en cours de mise à jour (future release) :

```
#----------------
#           Properties           used           For           the           last           session
#----------------
Keyname                                                                          :estMM8_HG18_CANFAM2
Description                                          :"EST  :  Dog  (Canis  familiaris),  Human
(Homo                Sapiens),                Mouse                (Mus                musculus)"
#----------------

First     utilisation                                                          :22-09-2007     00:00:13
Url                                                                :ftp://hgdownload.cse.ucsc.edu//goldenPath
Remote      regular      expression                                          :canFam2/bigZips/est.fa.gz
hg18/bigZips/est.fa.gz                                                                mm8/bigZips/est.fa.gz
Remote                     excluded                          regular                          expression:
Local          regular          expression                                                              :.*
Version     directory                                                                  :/db/estMM8_HG18_CANFAM2
Offline    directory                                          :/db/biomaj_tmp/estMM8_HG18_CANFAM2_tmp
Log files on state file           :true
```

```
#----------------
# Current Realease
#----------------

Release                          :2007-08-31
Number of session                :13
Last session                     :03-09-2007 11:45:00
Duration                         :69:55:45:00
```

```
Production directory
:/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-08-31
Num files downloaded              :3
Bandwidth (Mo/s)                  :0.20482694
Download size                     :2,270G
Bank size                         :15,803G

View Last log                     :
/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20070903112705/mirror.log

--------------
Unparseable date: ""
<process args="'fasta/*.fa flat/*/*/*.fa' '-p F -o T -n est_dmh -t  est_canfam_mm8_hg18'"
biomaj_error="false" desc="Index blast"
elapsedTime="00:31:23" exe="formatdb.bash" keyname="formatdb" name="formatdb" start="03-09-2007
10:45" type="index" value="-1" >
Unparseable date: ""
<process args="'fasta/*.fa flat/*/*/*.fa' '-p F -o T -n est_dmh -t  est_canfam_mm8_hg18'"
biomaj_error="false" desc="Index blast"
elapsedTime="00:00:00" exe="formatdb.bash" keyname="formatdb" name="formatdb" start="03-09-2007
10:32" type="index" value="-1" >


Processes by metaproc:
Metaproc:[P1]
log:[/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20070903112705/postprocess.P1.log]
-->concat(concatenation des fichiers est au format fasta)
-->formatdb(Index blast)
```

```
#----------------
#                                   Futur                                   Release
#----------------
Release                                                                  :2007-09-23
Number      of      session                                                     :14
Production         directory
:/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-09-23
Num        files        downloaded                                             :3
Bandwidth       (Mo/s)                                                   :0.07028251
Download      size                                                          :2,270G
Bank      size                                                               :0K

View      Last      log                                                        :
/db/biomaj/Bin/log/estMM8_HG18_CANFAM2/20071001161020/mirror.log
--------------
Processes                                  by                             metaproc:
```

```
#                      List                      production                directories
20-06-2007                                    15:51:00
/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-06-16                      (6,512G)
03-09-2007                                    11:45:00
/db/estMM8_HG18_CANFAM2/estMM8_HG18_CANFAM2_2007-08-31                     (15,803G)
```

Illustration 9: Résultat de la commande status d'une banque données (biomaj.sh –S alu)

Comme on peut le voir dans l'exemple ci-dessus:
-un bloc d'information est lié à la configuration, il contient :

- Le nom de la banque
- Sa description (correspondant à **db.fullname**)
- La date de première session
- L'url de téléchargement
- Les fichiers à télécharger (expression régulière)
- Les fichiers à déplacer en production (expression régulière)

- figurent ensuite 2 blocs associés aux « Releases » current et future contenant :

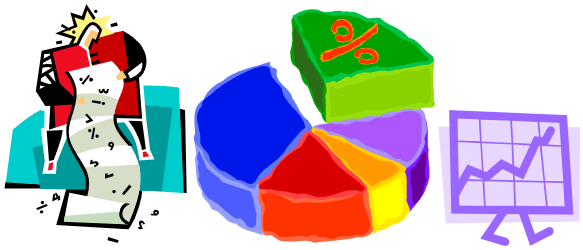
- La release

- La date de dernière session
- La durée de mise en ligne de la dernière release.
- Le répertoire de production
- Le nombre de fichiers téléchargés
- La bande passante
- La taille des fichiers téléchargés
- La taille des fichiers dans le répertoire de production
- Le nombre de fichiers en production
- Le nombre de méta-process appliqués au workflow
- Le nom du fichier log de la session.

- Enfin figure un dernier bloc avec :

- La liste des versions disponibles localement

3.4.4 Rapport Html



Le package de BioMAJ, permet de générer un rapport html comportant des informations sur chaque banque ainsi que des statistiques calculées à partir de l'ensemble des sessions appliquées à chaque banque.

3.4.4.1 Génération du rapport

Configurez le répertoire de destination du rapport HTML dans le fichier de configuration :

`BIOMAJ_ROOT/general.conf`
(propriété *webreport.dir, par défaut : `$BIOMAJ_ROOT/rapport/`*).

Puis générez le rapport :

```
$BIOMAJ_ROOT/bin/Make_biomaj_report.sh all
```

3.4.4.2 Consultation / déploiement du rapport

Pour visualiser le résultat, ouvrez avec votre navigateur favori le fichier crée, par exemple via :

```
firefox $BIOMAJ_ROOT/rapport/index.html
```

3.4.4.3 Structure du rapport

Le rapport html est obtenu par une remise en forme des fichiers d'état au format xml (ces fichiers se trouvent dans le répertoire `$BIOMAJ_ROOT/statefiles`) de chaque source. Il utilise également un index[3] résumant l'état des répertoires de production déployés par BioMAJ pour

[3] Il s'agit d'un fichier xml crée via la commande biomaj.sh -I . Ce fichier contient la liste des releases contenu dans le repository. (Toutes sources confondues)

chaque banque.

L'index (`$BIOMAJ_ROOT/rapport/index.html`) du rapport donne la liste des banques déployées par BioMAJ, pour chacune des banques une page historique est disponible. Elle comprend la configuration et les versions de la source. Enfin pour chacune des versions, une page recouvrant l'ensemble des sessions est mise à disposition.

Parallèlement, des informations sur les sessions de mise à jour sont exploitées pour générer des graphiques de synthèse d'une part sur l'entrepôt local dans son ensemble d'autre part sur chacune des sources.

3.4.4.3.1 Pages Html

Illustration 10: Page principale du rapport : liste du repository

La fenêtre principale contient une liste de l'ensemble des banques. Le menu à gauche représente les catégories de banques, pour visualiser seulement un sous-ensemble des banques (regroupé par le *db.type* définit dans chaque workflow).

La dernière colonne du tableau (*current release*) permet d'obtenir des informations supplémentaires sur l'ensemble des sessions d'une banque.

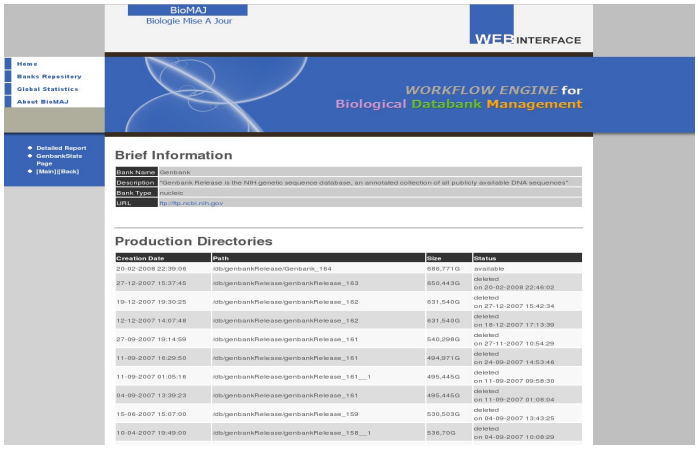


Illustration 11: exemple d'Information sur la banque

3.4.4.4 Statistiques

3.4.4.4.1 Statistiques globales

Les statistiques sont accessibles sur la page principale via le menu de gauche dans le portail.

Ces graphiques recouvrent les aspects suivants :

- La composition de l'entrepôt de donnée local (local repository).
- La répartition du poids de chaque classe de banque (dbtype).
- La répartition des sources dans chaque classe de banque.
- L'espace libre restant.
- La répartition de l'utilisation des serveurs distants.

Des exemples de résultats sont présentés dans les graphiques ci-après :

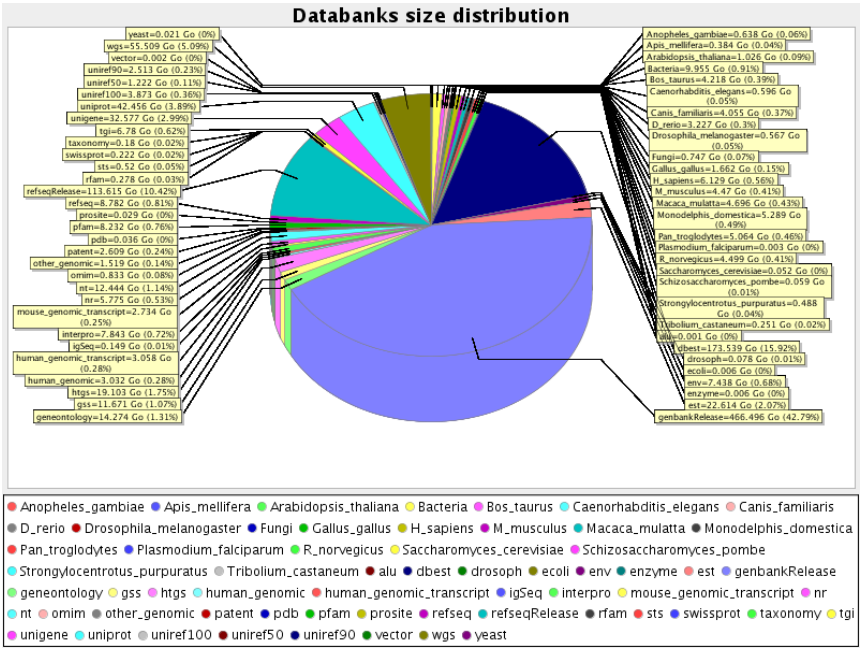


Illustration 12: Statistique : distribution de la taille des sources

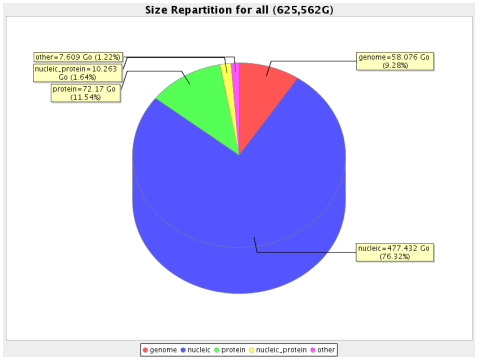


Illustration 13: Statistique : Répartition des données par type (dbtype)

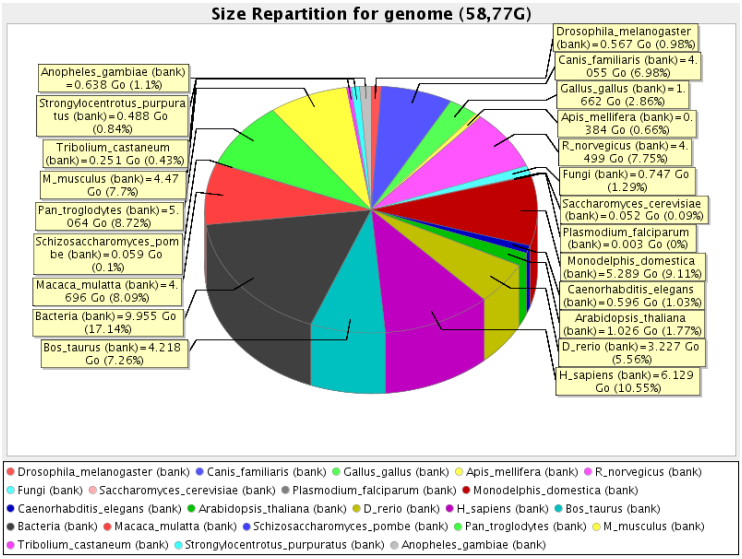


Illustration 14: Statistique : Répartition des données pour le type génome

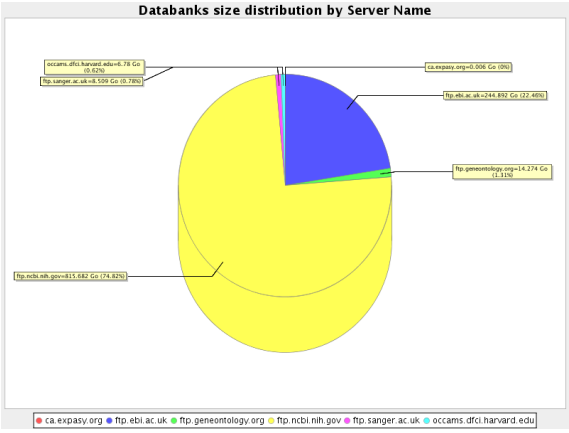


Illustration 15: Statistique : Répartition des données par serveur

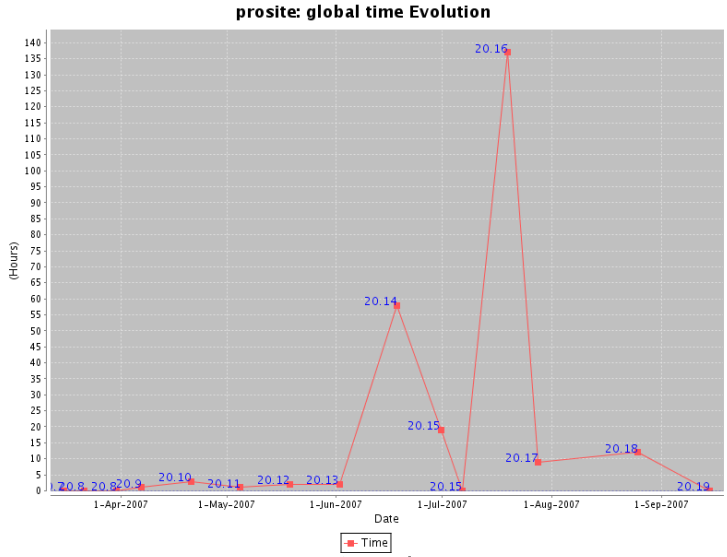
3.4.4.4.2 Statistiques par banque

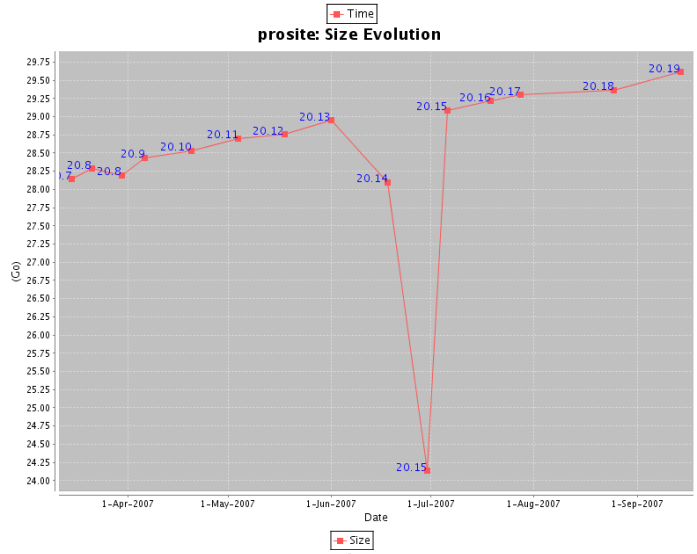
Les graphiques générés pour chaque banque représentent :

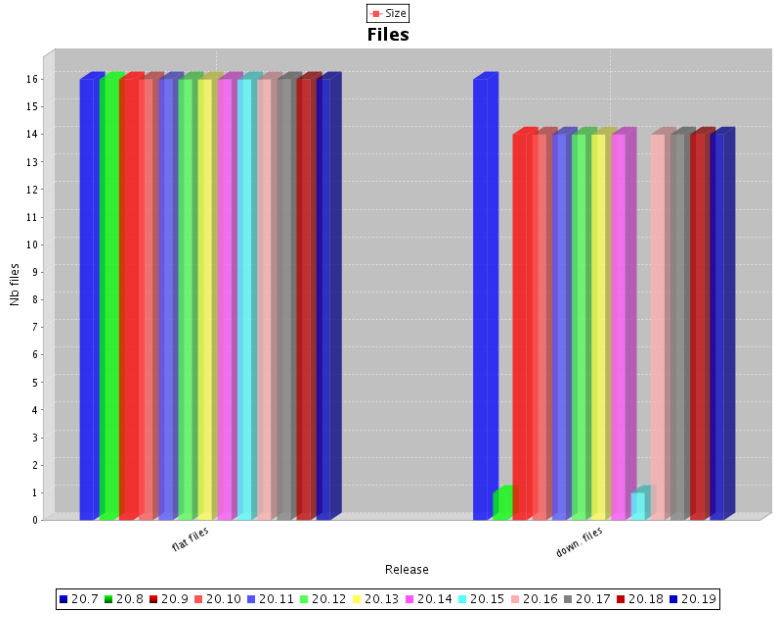
- L'évolution de la durée de traitement en fonction de la version
- L'évolution de la taille des versions d'une source
- L'évolution du nombre de fichiers de la source

Ces graphiques peuvent être utilisés à titre prospectifs pour anticiper l'espace disque nécessaire pour maintenir une source. Une inflexion importante entre la taille de deux versions consécutives d'une source est également très informative car elle est souvent révélatrice de problème. (Cet événement est traqué par ailleurs par le moteur BioMAJ, qui produira un warning »)

De exemples de graphique de banque sont présentés ci-dessous:


prosite: global time Evolution
(Hours)
Date
1-Apr-2007
1-May-2007
1-Jun-2007
1-Jul-2007
1-Aug-2007
1-Sep-2007
20.8
20.8
20.9
20.10
20.11
20.12
20.13
20.14
20.15
20.15
20.16
20.17
20.18
20.19
Time


prosite: Size Evolution
(Co)
Date
1-Apr-2007
1-May-2007
1-Jun-2007
1-Jul-2007
1-Aug-2007
1-Sep-2007
20.8
20.8
20.9
20.10
20.11
20.12
20.13
20.14
20.15
20.15
20.16
20.17
20.18
20.19
Size


Files
Nb files
flat files
down. files
Release
20.7
20.8
20.9
20.10
20.11
20.12
20.13
20.14
20.15
20.16
20.17
20.18
20.19

4 Création d'un workflow

4.1 Généralités

BioMAJ offre la possibilité de télécharger une source, de maintenir ses données à jour et d'exécuter des traitements sur ces données, (alerte, indexation, reformatage, processus de contrôles, analyses..). Le workflow décrivant cette suite de traitement est configurable à partir d'un fichier unique appelé un **fichier de propriétés**. La distribution BioMAJ comprend un ensemble de fichiers de propriétés pré-définies pour un large éventail de banques biologiques connues. Le présent chapitre a pour objectif de détailler le contenu et la structure des fichiers « properties » afin de vous aider à en définir de nouveaux.

Ces fichiers de propriétés doivent se trouver dans le répertoire `$BIOMAJ_ROOT/conf/db_properties` pour pouvoir être interprétés par BioMAJ.

4.2 Rappel de la Session de mise à jour d'une banque

Un fichier de propriétés contient les attributs des différentes étapes du cycle de mise à jour. Son contenu reflète le cycle de mise à jour.

Comme nous l'avons vu précédemment, le cycle de mise a jour est constitué de cinq étapes:

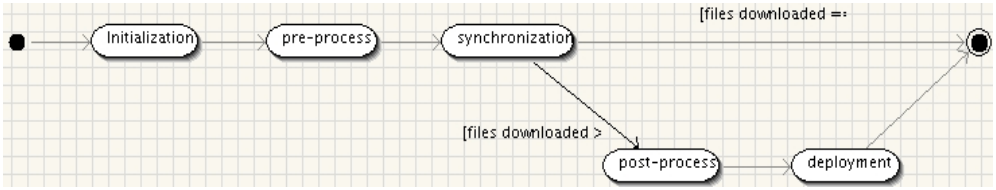


Illustration 16: Étapes du cycle de mise en ligne présenté sous forme séquentielle

(1) L'initialisation : Permet de créer les répertoires dans le cas d'une première session et de vérifier la cohérence des propriétés définies pour le workflow.

(2) L'exécution des pré processus : Exécute un sous workflow des pré processus définit par l'administrateur.

(3) La synchronisation : Récupération du numéro de version de la banque et téléchargement des nouveaux fichiers

(4) L'exécution des post-processus : Exécute un sous workflow de l'ensemble des post-processus définit par l'administrateur.

(5) Le déploiement : Mise en ligne de la version de la banque, et effacement des versions obsolètes.

Les phases (4) et (5) sont exécutées si au moins un fichier à été téléchargé ou effacé de la version précédente. Si la version locale est identique aux données distantes, le cycle de mise à jour ne produira pas de nouvelle version.

Une autre phase est exécutée quand une version de banque est supprimée pendant le déploiement. Il s'agit des remove processes définis dans le workflow par l'administrateur.

4.2.1 Configuration de la source : description / classification / localisation

4.2.1.1 Description et classification

Les premiers éléments de configuration que l'on trouve dans les propriétés d'une banque sont relatifs à sa description:

exemple: alu.properties

```
db.fullname="alu.n : alu repeat element. alu.a : translation of
alu.n repeats"
db.name=alu
db.type=nucleic_protein
db.formats=fasta,ncbi
```

- **db.name** : nom raccourci de la banque.
- **db.fullname** : nom complet de la banque.
- **db.type** : type sa classification.
- **db.formats** : liste des formats des données brutes qui la compose.

4.2.1.2 Localisation des données

Les propriétés suivantes définissent les informations de la banque liées à sa localisation dans l'entrepôt local.

data.dir=/bank/test
dir.version=ncbi/blast/alu
offline.dir.name=BioMAJ/ncbi/blast/alu_tmp

- **data.dir** : répertoire racine du repository. Il est souvent défini globalement dans le fichier *global.properties* pour toutes les sources
- **dir.version** ; répertoire racine pour la gestion des versions de la banque (**db.name** par défaut).
- **offline.dir.name** : répertoire temporaire utilisé localement pour le téléchargement, l'extraction des fichiers (**BioMAJ_tmp/db.name** par défaut).

4.2.2 Configuration de la phase de Synchronisation des données

Cette phase est découpée en sous-tâches :

1) Récupération du numéro de version de la banque.
2) Création d'une liste de fichiers correspondant à une version (interrogation du serveur distant).
3) Création d'une liste de fichiers à télécharger et d'une liste de fichiers à récupérer d'une

ancienne version.

4) Téléchargement et copie des fichiers listés par la phase 3) dans un répertoire temporaire.
5) Extraction des données compressées provenant du téléchargement.
6) Création d'un répertoire de production.
7) Délocalisation des fichiers temporaires (constituant la nouvelle version) vers le nouveau répertoire de production.

Ces sous-tâches font appel d'une part aux propriétés évoquées précédemment autour du nommage et la localisation et d'autre part à de nouvelles variables :

- L'adresse du serveur (**server**)
- Le protocole (**protocol**)
- Le répertoire racine de la banque sur le serveur (r**emote.dir**)
- Les expressions régulières sur les fichiers distants à télécharger (**remote.files**)
- Les expressions régulières sur les fichiers distants à exclure du téléchargement : (**remote.excluded.files**)
- Les expressions régulières sur les fichiers locaux pour appliquer des traitements (**local.files**)
- Le fichier où est contenu le numéro de version de la banque (**release.file**)
- L'expression régulière sur le numéro de version (**release.regexp**)

4.2.2.1 Serveur distant et protocole de téléchargement

La connexion au serveur de données implique la définition, des propriétés : **protocol, server et remote.dir.**

On peut configurer la synchronisation de la source avec 3 protocoles de téléchargement : *ftp, http, rsync* et un mode *local*. La variable r**emote.dir** est le répertoire distant à partir duquel on procède au téléchargement.

4.2.2.1.1 Le protocole ftp :

Voici un exemple de configuration pour le téléchargement des fichiers FASTA de la banque ALU au NCBI (ftp://ftp.ncbi.nih.gov/blast/db/FASTA/) :

```
#Mode ftp
protocol=ftp

#Serveur ftp
     server=ftp.ncbi.nih.gov

#Répertoire racine de téléchargement
remote.dir=/blast/db/FASTA/
```

Dans certain cas, si le serveur distant est lent et/ou chargé, il est possible de personnaliser la connexion en surchargeant les propriétés liés à la détection de perte de connexion et le nombre de tentative de reprise lors de la perte de communication avec le serveur.

```
#Time out lors de la phase de synchronisation (~20 minutes)
 ftp.timeout=1000000
#Nombre de reconnection au serveur autorisé lors de la phase de synchronisation
ftp.automatic.reconnect=5
```

La propriété **ftp.timeout** peut valoir -1, aucun timeout ne sera défini dans ce cas. L'unité de temps est en milliseconde.

Si la valeur de **ftp.timeout** est trop petite, la connexion ne pourra pas aboutir. Une erreur "java.net.SocketTimeoutException: Read timed out" sera émise.

Propriétés associées à la tâche de téléchargement :

```
#Nombre de thread de téléchargement
files.num.threads=3
#Options pour wget
wget.options=--tries=inf --wait=5 --random-wait --passive-ftp --timeout=50
```

Ces propriétés sont également valables pour le protocole http !

La propriété **file.num.threads** définit le nombre de téléchargements en parallèle autorisé par l'application.

La propriété **wget.options** permet de redéfinir la plupart des options de wget (http://www.gnu.org/software/wget/).
Il est impossible de redéfinir ces options:

- --cut-dirs
- --directory-prefix
- -a

4.2.2.1.2 Le protocole http :

Le protocole http est particulier car il est assez difficile de prévoir la forme des pages html et donc des attributs de fichier et de répertoire distant. BioMAJ offre la possibilité de redéfinir une expression régulière pour le parsing d'un répertoire (**http.parse.dir.line**) ou d'un fichier (**http.parse.dir.line**) et de sélectionner des groupes pour la récupération d'attributs (**http.group.dir.name, http.group.dir.date, http.group.file.name, http.group.file.date, http.group.file.size**).

```
#Mode Http
protocol=http

#Serveur http
server=astral.berkeley.edu

#Répertoire racine de téléchargement
remote.dir=/pdbstyle-1.71
```

```
#Expression régulière correspondant au parsing d'un répertoire
http.parse.dir.line=<img[\\s]+src=\"/icons/folder.gif\"[\\s]+alt=\"\\
[DIR\\]\".*href=\"([\\S]+)/\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:
[\\d]{2})

#Expression régulière correspondant au parse d'un fichier
http.parse.file.line=<a[\\s]+href=\"([\\S]+)\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]
{4}\\s[\\d]{2}:[\\d]{2})[\\s]+ ([\\d\\.]+[MKG]{0,1})
```

```
#Le nom du répertoire correspond au groupe 1 contenu par http.parse.dir.line
http.group.dir.name=1

#Le date du répertoire correspond au groupe 2 contenu par http.parse.dir.line
http.group.dir.date=2

#Le nom du fichier correspond au groupe 1 contenu par http.parse.file.line
http.group.file.name=1

#Le date du fichier correspond au groupe 2 contenu par http.parse.file.line
http.group.file.date=2

#Le taille du fichier correspond au groupe 3 contenu par http.parse.file.line
http.group.file.size=3
```

Cet exemple illustre le « parsing » d'une page html issue de l'url suivante: http://astral.berkeley.edu/pdbstyle-1.71.

On y trouve la balise suivante qui correspond au répertoire 0g daté du 31-Oct-2006 07:52 :

```
<img src="/icons/folder.gif" alt="[DIR]"> <a href="0g/">0g/</a>          31-Oct-
2006 07:52-
```

La propriété **http.parse.dir.line** contient l'expression régulière qui peut parser cette ligne. La valeur de **http.group.dir.name** (1), récupère le premier groupe parenthésé de **http.parse.dir.line** (([\\S]+/) correspond à « Og » dans notre exemple) et la valeur de **http.group.dir.date** (2) récupère le deuxième groupe parenthésé de **http.parse.dir.line** (([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:[\\d]{2}) correspond à « 31-Oct-2006 07:52 »).

De la même manière **http.parse.file.line** reconnait la balise html suivante :

```
<img src="/icons/unknown.gif" alt="[ ]"><a href="d10gsa1.ent">d10gsa1.ent</a>28-Oct-2006 00:00  82K
```

Les principales contraintes pour utiliser le protocole http avec BioMAJ sont :

- Si un répertoire existe, le nom et la date de création de ce répertoire doivent être présent.
- Le nom du répertoire et sa date de création doivent être sur la même ligne.
- Le nom, la date et la taille d'un fichier doivent être présent.
- Le nom, la date et la taille d'un fichier doivent être sur la même ligne.
- Une date doit être reconnu par une de ces trois expressions regulière :
  - [\\d]{2}-[\\d]{2}-[\\d]{4}\\s[\\d]{2}:[\\d]{2} (*ex : 02-04-2006 02:00*)
  - [\\d]{2}-[\\w]{3}-[\\d]{4}\\s[\\d]{2}:[\\d]{2} (*ex : 02-Avr-2006 02:00*)
  - [\\d]{2}-[\\d]{2}-[\\d]{4}\\s[\\d]{2}:[\\d]{2}:[\\d]{2} (*ex : 02-04:2006 02:00:00*)
- Une taille doit être reconnu par l'expression régulière suivante :
  - [\\d]+ ((\\.|,)[d++])? (G|M,K)? (ex: 82K ; 82 ; 1,2M ; 1.2M)

Il est également possible de redéfinir certaines propriétés pour le téléchargement : **wget.options** et **file.num.threads** (voir le protocole ftp ci-dessus).

4.2.2.1.3 Le protocole rsync:

Exemple de téléchargement de la banque enzyme sur bio-mirror avec le protocole rsync.

```
#Mode Rsync
protocol=rsync

#Serveur rsync
server=bio-mirror.net

#Répertoire racine de téléchargement
remote.dir=/biomirror/enzyme/
```

Le fonctionnement est similaire aux autres protocoles. Il est pour l'instant impossible de spécifier des options à l'utilitaire rsync.

Le protocole rsync est utilisé simplement comme protocole de téléchargement, la gestion de la différence entre le miroir et le répertoire de production est assurée par la phase de synchronisation des données de BioMAJ.

4.2.2.1.4 Le protocole local :

```
# Mode local (copie de fichiers sur la machine locale)
protocol=local
# Une seule valeur est valide : localhost pour ce mode
server=localhost
#Repertoire ou se trouve les données
remote.dir=/local/
```

Ce protocole peut être très pratique pour générer une version banque à partir de données déjà présentes sur le serveur qui héberge BioMAJ.

Ce protocole peut permettre notamment de définir des worflows de traitements de données locales, et d'assurer la supervision et le suivi des traitements grâce aux fonctionnalités de BioMAJ.

4.2.2.2 Construction des filtres de téléchargement (remote.files/local.files)

Les taches **2)** (création d'une liste de fichiers d'une version distante) et **3)** (création d'une liste de fichiers à télécharger et à récupérer d'une version locale) de la phase de synchronisation utilisent les expressions régulières en java (pour plus d'information consultez la document : SBIOMAJ_ROOT/doc/regexp.pdf).

Ces expressions permettent de définir les flux de fichiers des taches ultérieures (**4), 5) et 6)**). Ces deux flux correspondent aux fichiers à transférer du serveur distant (via la regexp **remote.files**) et aux fichiers à transférer du répertoire temporaire vers le futur répertoire de production (via la regexp **local.files**).

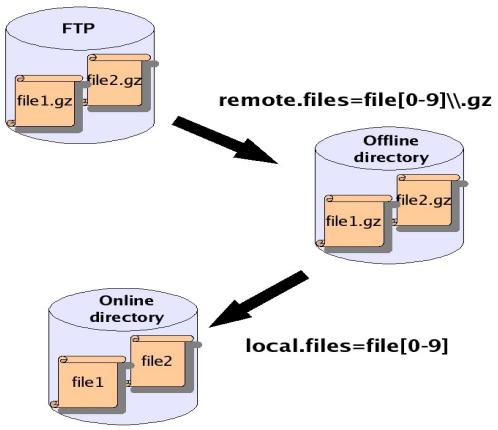


Illustration 17: Définition des propriétés remote.files et local.files

Exemple de définition pour la banque PFam:

```
remote.files=^Pfam.*\\.gz$    ^swisspfam\\.gz$     ^version.*$
^pfamseq\\.gz$

local.files=^Pfam.*$ ^swisspfam.*$ ^version.*$ ^pfamseq.*$
```

Dans l'exemple ci-dessus :

L'expression régulière remote.files sélectionne sur le serveur distant les fichiers ayant pour nom : Pfam.*.gz , swissfam.*.gz , version.*, pfamseq.gz

Ensuite L'expression régulière local.files sélectionne les fichiers nommés : Pfam.*, swissfam.*, version.*, pfamseq.* afin de constituer la liste des fichiers à déplacer.

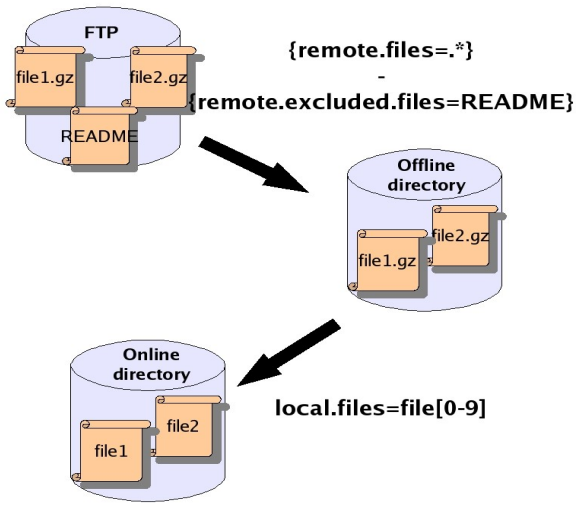


Illustration 18: Utilisation de la propriété remote.excluded.files

Il est parfois plus simple d'exprimer qu'on veut tout télécharger sauf certains fichiers. Il existe donc une troisième propriété (**remote.excluded.files**) qui exclura les fichiers non désirés de la propriété **remote.files.**

Voici un autre exemple tiré du workflow « Genome Anopheles gambiae (NCBI) » (*Anopheles_gambiae.properties*). Cet exemple illustre comment obtenir des sous répertoires spécifiques :

```
remote.files=^CHR_[\\w]+/.*$ ^maps/mapview/.*$ ^SNP/.*$
remote.excluded.files=.*\\.asn.*
local.files=[\\w]+$ [\\w+]/[\\w]+$ [\\w+]/[\\w+]/[\\w]+$
```

L'expression régulière ci-dessus sélectionne les répertoires commençant par CHR ainsi que les fichiers contenus et également le contenu du répertoire maps/mapview . Il est à noter que les fichiers *.asn ne sont pas téléchargés (cf. **remote.excluded.files**).

Un exemple de résultat de répertoire de production serait:

${dir.version}/current/flat/ :

- CHR_2/
  - NC_009071.gbk, NC_009071.faa,...
- CHR_3/
  - NC_009072.gbk,NC_009072.faa,...
- CHR_MT/
  - NC_002084.gbk,NC_002084.faa
- CHR_X/
  - NC_004818.gbk,NC_004818.faa
- maps/
  - mapview/
    - cyto.md,cytosat.md,cytoscf.md,...
- SNP/
  - mosquito_snp_20020625,mosquito_snp_readme.txt

(les fichiers *.asn ne sont pas téléchargés)

4.2.2.3 Obtenir un numéro de version

Pour chaque cycle BioMAJ tente de récupérer ou de créer un numéro de version au début de la phase de synchronisation de fichier.

Il existe trois façons de l'obtenir:

- Par défaut, la numéro de version correspond à la date du fichier le plus récent de la liste des fichiers à télécharger.
- Le numéro de version est contenue dans un fichier (**release.file**, **release.regexp** et **release.file.compressed** doivent être définis).
- Le numéro de version est contenu dans le nom du fichier (**release.regexp** doit être défini).

Quelques exemples:

```
#GenbankRelease : Nombre dans le fichier GB_Release_Number
release.file=GB_Release_Number
release.regexp=[\\d]+
release.file.compressed=false
```

Le résultat de cette recherche extrait les entiers correspondant au numéro de version de genbank. Dans le fichier GB_Release_Number.

Autre exemple avec la banque enzyme :

Le fichier contenant le numéro de version est enzuser.txt

Expression régulière est une date au format suivant : 24-Jul-2007

```
#Enzyme : Date dans le fichier enzuser.txt
release.file=enzuser.txt
release.regexp=[0-9]{2}-[\\w]{2,5}-20[0-9]{2}
release.file.compressed=false
```

Il est également possible d'extraire le numéro de version qui n'est pas forcément sur le serveur où se trouve les données. Le fichier contenant le numéro de version de la banque enzyme est obtenu via un protocole et une localisation différente.

```
#protocole ftp pour le telechargement
protocol=ftp
server=ca.expasy.org
remote.dir=/databases/enzyme/

#on peut utiliser un autre protocole que celui du téléchargement, exemple avec rsync.....
release.file=rsync://bio-mirror.net/biomirror/enzyme/enzuser.txt.Z
release.regexp=[0-9]{2}-[\\w]{2,5}-20[0-9]{2}
release.file.compressed=true

#Fichiers à telecharger du serveur ftp
remote.files=.*\\.txt$ .*\\.dat$ .*\\.get$
remote.excluded.files=
```

Depuis la version 0.9.3.0, il est possible de récupérer une partie de l'expression régulière via l'utilisation de parenthèse :

Exemple avec UniProt :

```
release.file=reldate.txt
#On récupère seulement le groupe parenthése
release.regexp=UniProt\\sKnowledgebase\\sRelease\\s+([\\d]+\\.*[\\d]*)
```

Le résultat sera : 12.8.

Dans un grand nombre des cas présentés dans ce chapitre, la compréhension des

expressions régulières définis en java est indispensable. Pour de plus amples information sur cette syntaxe, nous vous invitons à consulter le document suivant: `$BIOMAJ_ROOT/doc/regexp.pdf`

4.2.2.4 Consolidation des données

La consolidation des données a pour objectif de récupérér et rassembler localement les données dans un nouveau répertoire de production.

La phase de téléchargement nécessite la génération d'une liste de fichier de la version sur le serveur distant. Cette liste établie lors de la phase 2) va permettre de définir une liste de fichiers à télécharger et une liste de fichiers à récupérer localement (phase 3)).

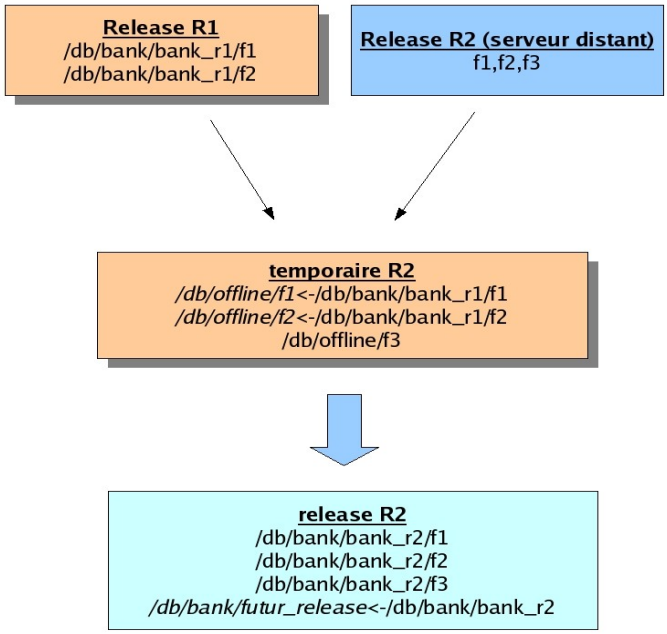


Illustration 19: Phase de synchronisation

Plusieurs propriétés sont impliquées dans le processus,

remote remote.files= expression régulière
remote.excluded.files= expression régulière

local.files= expression régulière ➔ filtrage des fichiers à déplacer du répertoire offline.dir.name au répertoire flat de la release.
No.extract=true| false

offline.dir.name = répertoire de travail temporaire

do.link.copy=true|false

Lorsque les fichiers sont téléchargés, ces fichiers sont décompressés dans un répertoire temporaire (**offline.dir.name***)*. La propriété **no.extract** (true|false) indique à BioMAJ qu'il doit passer l'étape de décompression.

Les fichiers définis grâce à **remote remote.files et remote.excluded.files,** qui sont en commun avec la nouvelle et l'ancienne version sont copiés ou liés symboliquement dans le

répertoire de travail temporaire lors de la tâche **versionsmanagement** .

(La propriété **do.link.copy=true** permet de choisir entre le lien ou la copie).

En suite la tache **versionsmanagement** crée dans le répertoire de production (**dir.version)**, le répertoire de la nouvelle version. Le nom de ce dernier est déterminé en fonction la définition des propriétés vues en 4.2.2.3. Enfin **versionsmanagement** positionné le lien *future_release* sur le répertoire créé.

Pour finir la tache **move** termine la consolidation des données. Elle déplace tout ou parti des fichiers du répertoire temporaire vers le répertoire future_release/flat. La liste des fichiers à déplacer est déterminée grâce à l'expression régulière définie dans la propriété **local.files.**

A ce stade les données brutes de la nouvelles release sont disponible localement pour réaliser la phase suivantes. Les post-processus de traitements peuvent débuter.

Lorsque les fichiers de la source sont en grande quantité plusieurs milliers si ces fichiers ne sont pas des archives (tar, zip, rar…). L'activation de la propriété log.files =false allégera considérablement, la taille du fichier log xml et aura pour conséquence d'optimiser le fonctionnement de l'application.

4.2.3 Configuration des phases de pré-processing, remove-processing et post-processing

4.2.3.1 Fonctionnement général :

Les données brutes sont disponibles dans data.dir/dir.version/future_release/flat.

BioMAJ permet de définir des workflows de post-traitement de topologie relativement complexe. Leurs définitions font appel à trois types d'éléments : les blocs, les meta-processus et les processus. Le pre-processing et le remove processing utilisent deux de ces éléments : les meta-processus et les processus. Le support des blocs pour les pre et remove processing pourrait être implémenté dans une future version.

Le workflow est une suite de blocs qui s'exécutent séquentiellement. Un bloc contient un ou plusieurs méta-processus. Les méta-processus d'un bloc sont exécutés en parallèle.

Un meta-processus contient une liste de processus (jobs, traitements, alerte, ..) qui sont exécutés séquentiellement. Ce formalisme permet de définir un workflow complexe représentant un graphe acyclique dirigé. (En anglais D.A.G pour directed acyclic graph). L'alternance séquentielle, parallèle est utile pour la synchronisation de processus (des « rendez-vous » entre processus) et paralléliser les traitements pour exploiter les performances des machines actuelles (multi-processeur, cluster de machine).

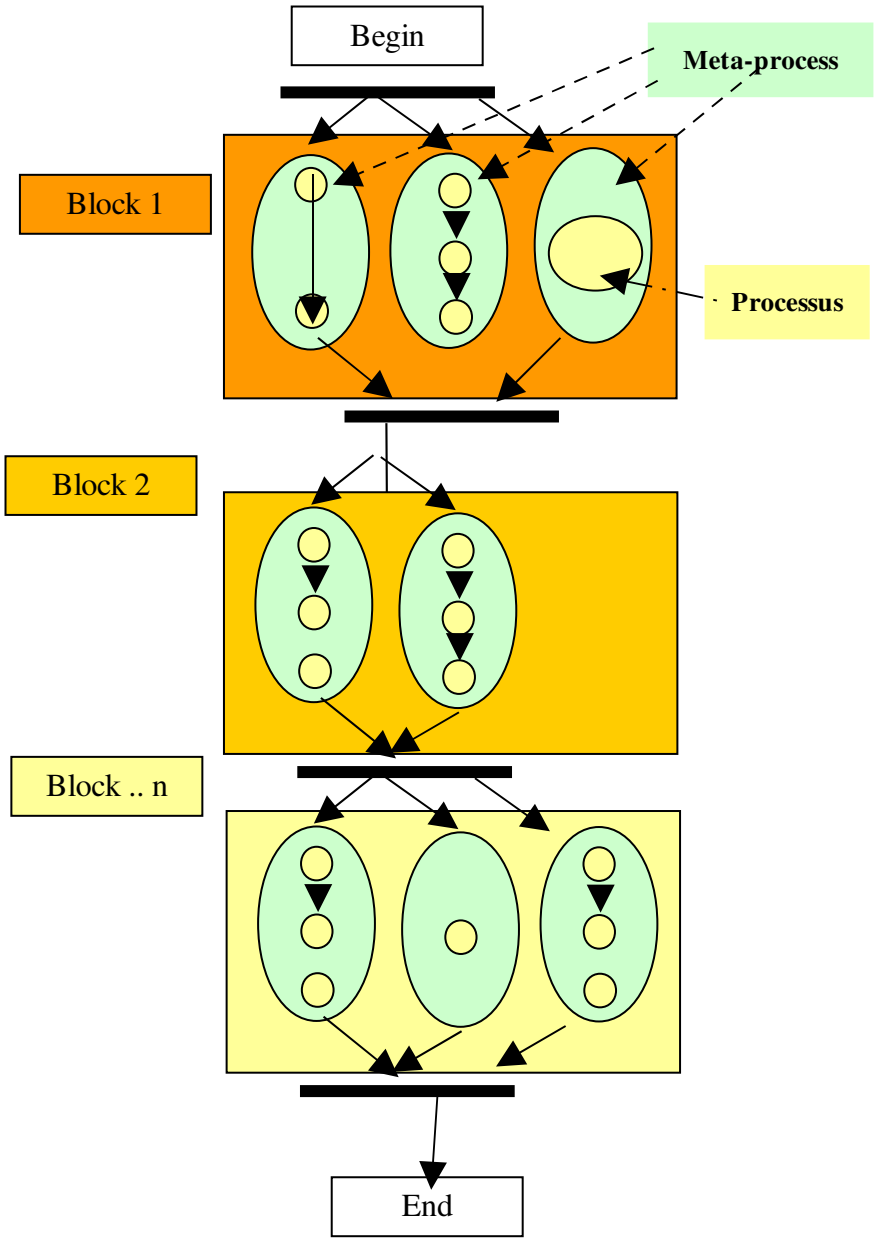


Illustration 20: représentation schématique des workflows des phases de pré et post processus : dépendance entre les blocs, les meta-process, et les process BioMAJ.

Ces phases permettent d'exécuter un ensemble de processus (synchrone ou asynchrone) avant (pré-processing) ou après (post-processing) la mise à disposition de la banque. Ces processus peuvent être informatifs (envoi de mail) ou générer des indexes, assurer des conversions de formats sur les données brutes, réaliser des extractions, des analyses ….

4.2.4 Définition des éléments

4.2.4.1 Bloc

Un bloc est défini par un nom. La liste des blocs de post processus est déclarée via la propriété BLOCKS.

Exemple :

*BLOCKS= **A,B,C***

Dans cet exemple, trois blocs A, B et C sont définis dans le workflow. L'ordre de déclaration définit celui de l'exécution.

4.2.4.2 Méta-processus

- Déclaration

Les méta-processus d'un bloc sont déclarés dans une propriété dérivant du nom de bloc d'appartenance suivie du suffix **.db.post.process**

Exemple :

***A**.db.post.process=META1,META2*

***B**.db.post.process=META3*

***C**.db.post.process=META4*

Dans l'exemple ci-dessus quatre méta processus sont définis dans le worklow :

- 2 dans le bloc A (META1, META2)
- 1 dans le bloc B (META3)
- enfin le méta processus META4 est défini dans le bloc C.

En ce qui concerne les pre-processus et les remove process, leurs meta processus sont définis en utilisant les propriétés db.pre.proccess and db.remove.process respectivement (sans notion de block).

- ***Initialisation***

Ensuite Les processus de chaque méta processus sont définis.

Exemple :

***META1**=P1,P2*

***META2=**Z1*

***META3**=P3,test7*

***META4**=message*

En l'occurrence dans l'exemple :

2 processus « P1 », « P2 » sont définis dans META1. le processus « Z1 » est déclaré dans META2. META3 contient deux processus « P3 » et « test7 » Enfin META4 contient un seul traitement nommé « message ».

Chaque méta-processus et processus a un nom unique. Il est constitué d'une chaîne de caractère de longueur libre ([a-z,A-Z,0-1]) sans espace. L'ordre de déclaration des processus définit l'ordre de leur exécution dans le méta-processus.

4.2.4.3 Processus

il faut ensuite, définir chaque processus déclaré dans les méta-processus :

Chaque processus est définit par 5 attributs : le nom, l'exécutable, les arguments de

l'exécutable, la description du processus, enfin son type. Chaque attribut est défini dans une propriété dont le nom dérive du nom du processus auquel elle se réfère. L'extension de la propriété est fixe :

- ***processus*.name** : nom du processus
- ***processus*.desc** : description du processus
- ***processus*.type** : type de processus
- ***processus*.exe** : chemin absolue d'un exécutable (ou relatif à $BIOMAJ_ROOT/conf/process)
- ***processus*.args** : paramètres de l'exécutable

Ainsi pour un processus exemple <MYprocess> nous aurons :

*MYprocess**.name=** MYprocess*

*MYprocess**.desc= processus test***

*MYprocess**.type=test***

*MYprocess**.exe=Myshell.sh*** ***# ce script doit être localisé dans le répertoire $BIOMAJ_ROOT/conf/process***

*MYprocess**.args= -a***

Il est possible de donner en argument des propriétés définies dans le fichier de propriété et également la propriété définie dynamiquement par le workflow ${remote.release} contenant le numéro de release.

Exemple :

```
print.name=echo
print.exe=echo
print.args=Bank : ${db.name} VERSION : ${dir.version} RELEASE : ${remote.release}
print.desc=Affichage des proprietes du workflow
print.type=Affichage
```

Affichage :

Bank : alu VERSION : TEST RELEASE : 2008-01-31

Si nous considérons l'exemple que nous avons défini dans le paragraphe précédent :

7 processus doivent être définis : *P1,P2,Z1,P3,test7,message*

Le fichier properties de la banque contiendra 35 déclarations (5X7).

P1 ➔	P1.name, P1.desc, P1.type, P1.exe, P1.args
P2 ➔	P2.name, P2.desc, P2.type, P2.exe, P2.args
Z1➔	Z1.name, Z1.desc, Z1.type, Z1.exe, Z1.args
P3➔	P3.name, P3.desc, P3.type, P3.exe, P3.args
Test7➔	test7.name, test7.desc, test7.type, test7.exe, test7.args

message➔	message.name, message.desc, message.type, message.exe, message.args
Z1➔	Z1.name, Z1.desc, Z1.type, Z1.exe, Z1.args

Notez que l'ordre de déclaration n'a pas d'importance. Elles doivent cependant être toutes déclarées. Deux processus peuvent faire appel au même exécutable.

4.2.5 Exemple de workflow à partir des scripts disponibles dans BioMAJ

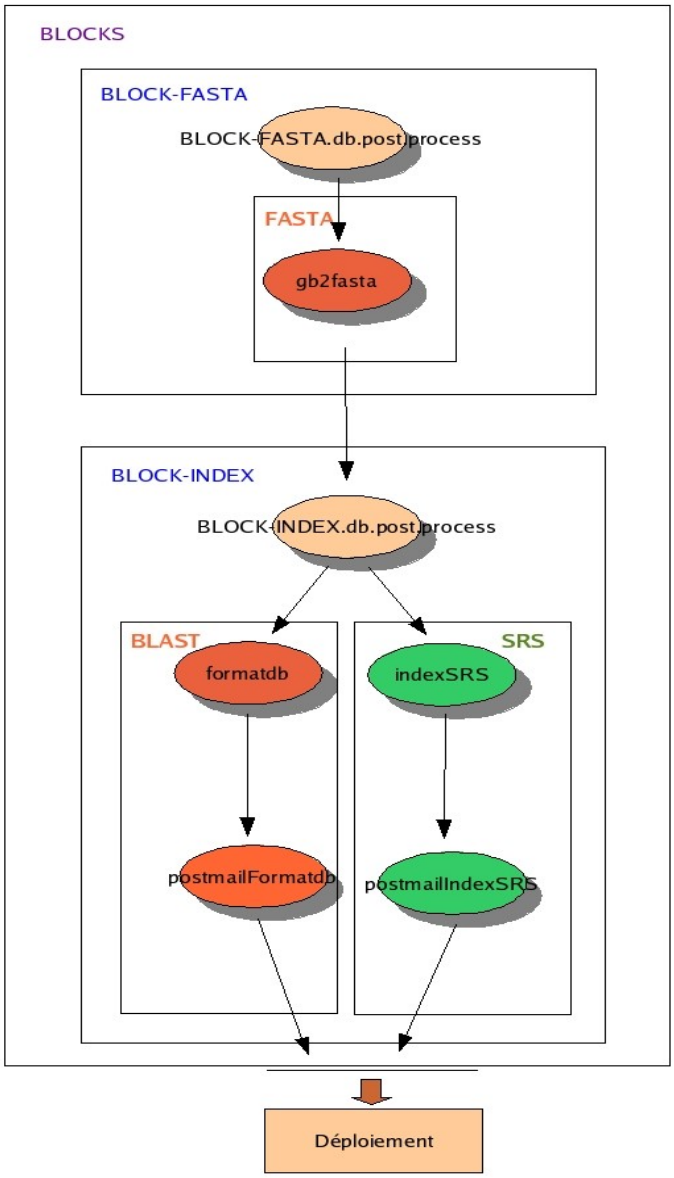


Illustration 21: Exécution des processus par BioMAJ

L'exemple utilise quatre post-traitements :

L'ensemble des scripts est distribué avec l'application :

- formatdbTLSE.pl : formatage des banques blast

  Documentation: $BIOMAJ_ROOT/doc/process/formatdbTLSE.html
- gb2fasta.sh: génération de fichier fasta à partir d'indexes blast
- indexSrsTLSE.pl : formatage des banques pour SRS

  Documentation: $BIOMAJ_ROOT/doc/process/indexSrsTLSE.html

- sendMailTLSE.pl : envoi de mail.

  Documentation: $BIOMAJ_ROOT/doc/process/sendMailTLSE.html

Configuration du fichier de propriétés :

```
#Definition des blocs, qui seront exécutés (exécution séquentielle)
BLOCKS=BLOCK-FASTA,BLOCK-INDEX
#Le bloc BLOCK_FASTA est composé d'un meta-process
BLOCK_FASTAdb.post.process=FASTA
#Le bloc BLOCK_INDEX est composé de deux méta-processus qui seront exécutés en parallèle
BLOCK_INDEX=SRS,BLAST
```

```
#Les méta-processus définissent un ensemble de processus (exécution séquentielle)
FASTA=gbtofasta
BLAST=formatdb,postmailFormatdb
SRS=indexSRS,postmailSRS
```

```
#DEFINITION DES PROCESSUS POUR FASTA
#-------------------------------------------------------
gbtofasta.name=gb2fasta
gbtofasta.exe=gb2fasta.sh
gbtofasta.args=
gbtofasta.desc=Index blast
gbtofasta.type=index
```

```
#DEFINITION DES PROCESSUS POUR BLAST
#-------------------------------------------------------

#définition du processus formatdb
formatdb.name=formatdbTLSE
formatdb.exe=formatdbTLSE.pl
formatdb.args= '*.seq' '.seq' gb
formatdb.desc=Index blast
formatdb.type=index

#définition du processus postmailFormatdb
postmailFormatdb.name=sendMail
postmailFormatdb.exe=sendMailTLSE.pl
postmailFormatdb.args=-s '[EBI - db.name remote.release] End Post Process formatdb' -
```

```
m 'local.time'
        postmailFormatdb.desc=mail
        postmailFormatdb.type=info
```

```
        #DEFINITION DES PROCESS POUR SRS
        #-------------------------------------------------

        #définition du processus indexSRS
        indexSRS.name=indexSRS
        indexSRS.exe=indexSrsTLSE.pl
        indexSRS.args=-v -d genbankrelease --pvm --execute pbs -c 6
        indexSRS.desc=Index srs
        indexSRS.type=index

        #définition du processus postmailSRS
        postmailSRS.name=sendMail
        postmailSRS.exe=sendMailTLSE.pl
        postmailSRS.args=-s '[EBI - db.name remote.release] End Post Process formatdb' -m
'local.time'
        postmailSRS.desc=mail
        postmailSRStype=info
```

4.2.6 Déploiement

Cette phase consiste en les opérations suivantes :

- Supprimer le lien *future_release* (mis en place après la phase de synchronisation et utiliser par la phase de post-processing)
- Appliquer à la nouvelle version les droits définis par la propriété **production.directory.chmod**.
- Créer ou déplacer le lien *current_release.*
- Détruire des versions obsolètes, c'est-à-dire les versions de rang supérieur à la propriété **keep.old.version**
- Pour chaque version supprimée, les remove processus sont lancés.

Exemple :
si **keep.old.version = 0** alors la version courante sera effacé après qu'une nouvelle version ait été constitué.
Si **keep.old .version =1**, le programme conservera deux versions. La version courante et la

version précédente.

Compte tenu du fonctionnement du cycle BioMAJ, il est important de préciser que juste avant le déploiement, une version est présente dans le répertoire future_release. Pour fonctionner, même si keep.old.version = 0, BioMAJ aura besoin d'un espace disque libre égale à la somme de la taille des deux versions!

4.3 Surcharge d'information : global.properies

Le fichier de propriétés **global.properties**[4] est utilisé pour définir les informations communes à l'ensemble des banques maintenues par l'application. Pour les besoins d'une source particulière, vous avez la possibilité de redéfinir certaines propriétés dans le fichier de propriétés de la banque afin d'adapter spécifiquement le comportement de BioMAJ au contexte.

Informations Générales (global.properties):

Paramètre du moteur :

- Nombre de workflows à exécuter en parallèle (**bank.num.thread**) : BioMAJ peut exécuter des workflows en parallèle lors d'une même session, cette propriété limite le nombre de banque accédant aux serveurs distants.

Génération des logs :

- Générer le nom de la tache dans les logs (**historic.logfile.task**)
- Générer le nom de la meta-tache dans les logs (**historic.logfile.target**)
- Générer l'ensemble des propriétés ant du workflow (**historic.logfile.properties**)
- Générer les logs correspondant à un niveau de journalisation (**historic.logfile.level**)

Organisation des données / Politique de transfert de données:

- Répertoire racine pour l'ensemble des banques (**data.dir**) : Ce répertoire correspond à /db dans les exemples cités.
- Droit d'accès sur les fichiers en production (**production.directory.chmod**)
- Nombre de versions de release à garder en production (**keep.old.version**)
- Méthode de récupération des fichiers locaux par lien symbolique ou copie (**do.link.copy**)

Chemin des exécutables

- Chemin de l'exécutable tar (**tar.bin**)
- Chemin de gunzip (**gunzip.bin**)
- Chemin de wget (**wget.bin**)
- Chemin de bunzip (**bunzip.bin**)

Format de la release

[4] Le contenu d'un fichier global.properties est présenté en annexe.

- le format d'une release si la politique de récupération est la date du fichier le plus récent (**release.dateformat**) (voir http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html)

Alerte :

- serveur smtp pour un envoi de mail (**mail.smtp.host**)
- mail des administrateurs (**mail.admin**)
- mail d'un administrateur (**mail.from**)

Propriétés par défaut du protocole Ftp/ Http

- Nombre de fichiers à télécharger en parallèle *(***file.num.thread***)*
- Options à donner en argument à wget (**wget.options**)

Propriétés par défaut du protocole Ftp

- Port par défaut (**port**)
- Nom du login (**username**)
- Mot de passe (**password**)
- Définition d'un time out (**ftp.timeout**)
- Nombre de reconnexion automatique (**ftp.automatic.reconnect**)

Propriétés par défaut du protocole Http

- Expression régulière d'une balise html comportant un répertoire (**http.parse.dir.line**)
- Expréssion régulière d'une balise html comportant un fichier avec attributs (**http.parse.file.line**)
- Numéro du Groupe parenthésé correspondant au nom d'un répertoire (**http.group.dir.name**)
- Numéro du Groupe parenthésé correspondant à la date d'un répertoire (**http.group.dir.date**)
- Numéro du Groupe parenthésé correspondant au nom d'un fichier (**http.group.file.name**)
- Numéro du Groupe parenthésé correspondant à la date d'un fichier (**http.group.file.date**)
- Numéro du Groupe parenthésé correspondant à la taille d'un fichier (**http.group.file.size**)

5 Développement / intégration de post ou pré traitements

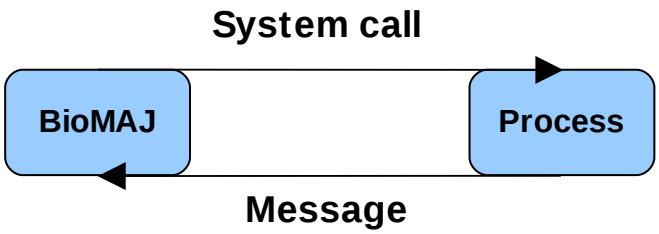


Illustration 22: communication BioMAJ Processus

BioMAJ permet d'exécuter des scripts/binaires utilisateurs. Le moteur interagit avec ces scripts via la définition de variables d'environnement. Les exécutables – pour être utilisable - doivent être placé dans le répertoire $BIOMAJ_ROOT/conf/process.

Comme nous l'avons évoqué, lors des phases post et pré processus, BioMAJ lance un appel système à des programmes et supervise leur exécution. Par défaut BioMAJ récupère simplement la valeur de retour en fin d'exécution de l'appel du programme. Si la valeur est nulle, le moteur considère que tout c'est bien passé et exécute le prochain processus. Si la valeur de retour est non nulle, le traitement à un statuts erreur et le cycle de mise à jour s'arrête. Les processus exécutés en parallèle seront arrêtés au prochain point de synchronisation (fin bloc ou de méta-processus).

Il est cependant possible d'aller plus loin et d'accroître l'interaction moteur/traitement. Pour ce faire il est nécessaire de prendre en comptes les spécifications suivantes dans le développement du code du programme.

L'intégration comprend trois axes :

- Le passage de messages
- La gestion des fichiers temporaires
- L'utilisation du contexte de la session BioMAJ

Ces trois axes vous permettrons de développer des traitements utilisant pleinement les capacités du moteur BioMAJ.

5.1.1 Gestion de la communication : passage de messages

Pour la communication avec l'administrateur BioMAJ utilise trois types de messages :
Les informations usuelles, les messages d'avertissements, les messages d'erreurs.

- Les informations usuels sont des messages à caractères informatifs.
- Les messages d'avertissement sont destinés à attirer l'attention sur un événement particulier. Il ne s'agit pas d'erreur grave qui nécessite l'arrêt de la session, mais il est important de les porter à la connaissance de l'administrateur afin qu'il puisse prendre -si nécessaire - une décision (après la fin session).
- Les messages d'erreurs sont émis lorsque l'application détecte un événement fatal à son fonctionnement. C'est souvent la dernière action effectuée avant l'arrêt de la session.

Des variables d'environnement sont utilisées pour « tagger » les messages émis et ainsi assurer leur classification dans l'un des trois niveaux : INFO,WARNING et ERROR.

Pour émettre un message sur les « canaux » INFO et ERROR. Il suffit d'utiliser les commandes classiques de Unix ou de votre langage de développement pour écrire sur la sortie standard pour le canal « INFO » et sur la sortie erreur standard pour ERROR.

Pour écrire un message d'avertissement, - canal WARNING - il suffit de « tagger » votre message warning en concaténant en amont du message la variable d'environnement PP_WARNING.

Exemple en perl :

Info → print STDOUT msg
Warning → print STDOUT ENV{PP_WARNING}msg
Erreur → print STDERR msg
PP_WARNING au message et de l'afficher sur la sortie standard.

Exemple en csh :

```
echo "sortie INFO"
echo $PP_WARNING"Ceci est un warning"
echo "sortie erreur" 1>&2
```

BioMAJ journalise ces informations (écriture sur le système de fichier). Une utilisation intensive des messages peut entraîner un net ralentissement notamment lors de l'utilisation du mode verbeux de l'application.

5.1.2 Gestion des dépendances de fichiers produits

Par défaut BioMaJ trace la liste des fichiers des versions qu'il supervise. Il utilise les informations notamment pour optimiser la re-exécution d'un workflow. Il peut utiliser un mécanisme de dépendance de fichier attaché à un processus. Chaque processus doit alors déclarer les fichiers qu'il génère. En cas de rejeu, BioMAJ re-exécute le processus si les fichiers déclarés n'ont pas été trouvé.

Les fichiers produits lors de traitement peuvent être classée en trois catégories: Les fichiers déclarés, les fichiers temporaires enfin des fichiers de statut inconnu.

-Les fichiers de statut inconnu, sont dans l'arborescence. BioMaJ ne les connait pas. Ils ne sont pas déclarés !

-Les fichiers temporaires sont effacés en fin de session. Il s'agit de fichier « volatile » qui disparaît en fin de cycle.

-Les fichiers déclarés quant à eux sont généralement des fichiers résultats qui correspondent au résultat du traitement.

Le programme appelé peut, via la déclaration, délégué au moteur des actions comme le contrôle d'intégrité de fichier ou l'effacement de fichier temporaire. Pour réaliser cette délégation,

le programme doit déclarer au moteur les fichiers dont il dépend ainsi que leur nature (volatile = temporaire et résultat). Une dépendance signifie que le fichier généré existera dans le répertoire de production après la mise en ligne de la banque. Une dépendance volatile signifie que le fichier sera effacé à la fin de l'exécution du workflow. Cette dernière catégorie est également utilisée pour la gestion des fichiers temporaires utilisés par plusieurs processus (concaténation de fichier et indexation du résultat par exemple).

La déclaration utilise le même principe que ce ceux utilisés pour les messages.

Il s'agit d'un message avec un tags écrit sur la sortie standard du programme exécuté.

Les messages ont la structures ci-dessous :

Déclaration des dépendances:

echo $PP_DEPENDENCE$CheminAbsolueFichier

echo $PP_DEPENDENCE_VOLATILE$CheminAbsolueFichier

En langage PERL il suffira de créer une fonction qui déclare les dépendances de fichier.

Fichier résultat final conservé → print STDOUT ENV{PP_DEPENDENCE}file

Fichier volatile (temporaire) → print STDOUT ENV{PP_DEPENDENCE_VOLATILE}file

BioMAJ efface les fichiers volatiles déclarés, il est déconseillé de les effacer manuellement.

5.1.3 Passage de contexte au script : Information sur le numéro de version

Lors du lancement de chaque traitement, BioMAJ met à disposition du processus exécuté plusieurs variables d'environnement. Elles permettent de faire transiter une partie du contexte du cycle de mise à jours BioMAJ et d'obtenir les informations dynamiques relatives à la nouvelle version de la banque. La liste de ces variables figure ci-dessous :

- **dbname** : nom de la banque
- **datadir** : répertoire racine de l'ensemble des répertoires de production
- **offlinedir** : répertoire temporaire
- **dirversion** : répertoire de production de la banque contenant l'ensemble des versions déjà téléchargés ainsi que la future version.
- **remotedir** : répertoire du serveur distant
- **noextract** : booléen pour savoir si les fichiers téléchargés sont décompressés
- **localfiles** : fichier téléchargés et qui seront disponible en production
- **remotefiles** : expression régulière des fichiers téléchargés
- **mailadmin** : mail de l'administrateur
- **mailsmtp** : serveur smtp pour l'envoie de mail
- **remoterelease** : numéro de la version (disponible seulement pour les post-processus)
- **removedrelease**: numéro de la version supprimée (seulement pour les remove processes)
- **PATH_PROCESS_BIOMAJ** : répertoire racine des post-processus.

- **PATH_LOG_BIOMAJ** : répertoire de journalisation définit dans le fichier general.conf.
- **PATH_WORKFLOW_BIOMAJ** : répertoire des fichiers de workflow (.properties).
- **RELEASE_ALL_COMPRESSED_FILES_LIST** : Liste des fichiers constituant la release téléchargé (equivalent à un ls distant).
- **RELEASE_ALL_UNCOMPRESSED_FILES_LIST** : Liste des fichiers générés dans le répertoire flat de la version en construction.
- **RELEASE_OLD_FILES_LIST** : Liste des fichiers qui ont été copié de l'ancienne release (sous-ensemble de **RELEASE_ALL_UNCOMPRESSED_FILES_LIST**).
- **RELEASE_NEW_FILES_LIST** :Liste des fichiers qui ont été téléchargés en complément de la liste **RELEASE_OLD_FILES_LIST** (sous ensemble de **RELEASE_ALL_UNCOMPRESSED_FILES_LIST**).

Remarque :
RELEASE_ALL_UNCOMPRESSED_FILES_LIST = RELEASE_OLD_FILES_LIST + RELEASE_NEW_FILES_LIST
Pour obtenir ces quatre dernière variables d'environnement, il faut initialisé à « true » la propriété *list.files.available* dans votre workflow.

Quelques conventions et utilisations de ces variables:
- Les données téléchargées se trouvent dans `$datadir/$dirversion/future_release/flat`
- Pour l'utilisation de la version en production : `$datadir/$dirversion/current_release/flat`
- Dans un cas d'indexation, créez un répertoire sous `$datadir/$dirversion/future_release/` portant le nom de l'indexation (exemple : `$datadir/$dirversion/future_release/blast`) et générez les fichiers dans ce répertoire.

5.1.4 Code retour

En cas de détection d'erreur, le script doit retourner une code retour non nul afin d'arrêter le workflow.

5.1.5 Debugage

Un nouveau fichier de propriétés peut être déroulé étape par étape au moyen de la commande ci-dessous

- `BioMAJ .sh –d <dbname> ––stage <preprocess|sync|postprocess|deployment>`

L'option --stage fait référence au point d'arrêt de la session.

Contrairement à la commande de mise à jour standard, le cycle sera stoppé à la phase spécifiée.

5.2 Notions diverses

On peut définir plusieurs types de banque biologique selon leurs utilisations et la façon dont

elles sont générées.

5.2.1 Banque virtuelle

Il s'agit d'un alias regroupant plusieurs banques indépendantes.

Exemple : genbank = genbank + genbank_new

- `biomaj.sh -d genbank,genbank_new`

Ou créer un fichier de propriété pour votre banque virtuelle contenant la ligne suivante :

virtual.list=genbank, genbank_new

Deux sessions sont ouvertes indépendamment l'une de l'autre ;

La liste des banques peut être supérieur à deux. Pour éviter une saturation du réseau, Le nombre de session ouverte est plafonné par la propriété **bank.num.thread.**

5.2.2 Banque calculée

5.2.2.1 Principe d'une banque calculée

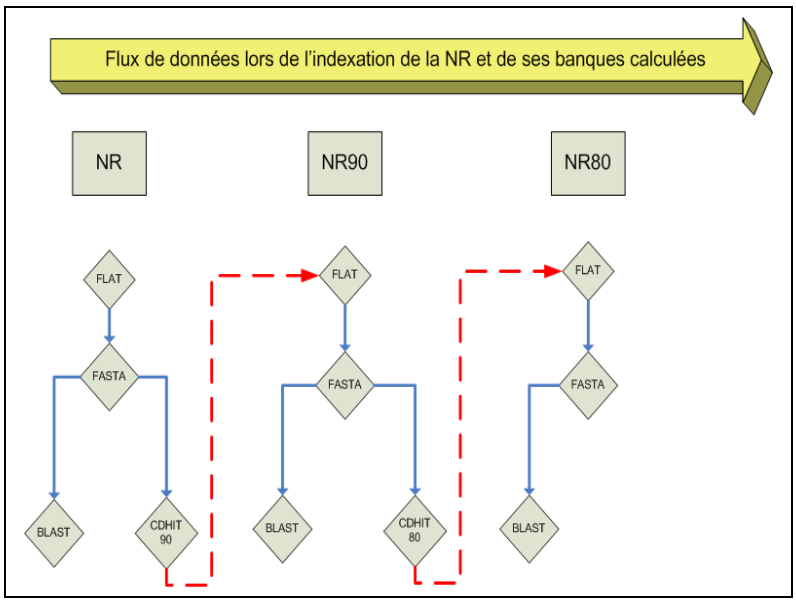
Une banque calculée est une banque dont la source est issue d'un traitement sur des données déjà téléchargés par le biais d'un workflow usuel (récupération de données sur un serveur distant), correspondant à une banque mère. Cette banque calculée (banque fille) peut être également la source d'une autre banque calculée. On peut ainsi générer en cascade des banques calculées, chacune issue d'un traitement de la précédente.

Pour automatiser la mise à jour et assurer la cohérence des répertoires de production des banques mères et filles, BioMAJ exécute les workflows des banques mères lorsque le workflow d'une banque calculée est exécuté et déploie les nouvelles versions de l'ensemble de ces banques simultanément.

Le principe de fonctionnement des banques calculées est illustré ci-après via l'exemple de la banque NR[5].

5.2.2.2 Illustration : NR

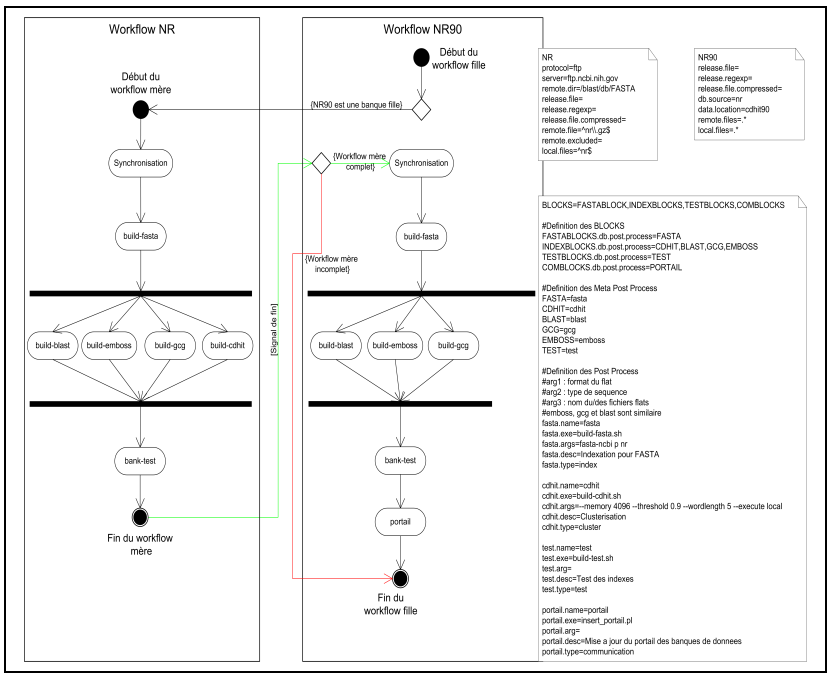
[5] (The protein entries in the Entrez search and retrieval system have been compiled from a variety of sources, including SwissProt, PIR, PRF, PDB, and translations from annotated coding regions in GenBank and RefSeq.).



La banque NR (Protéique Non Redondante) contient actuellement (09/2007) 5,5 millions d'entrés protéiques. Un certain nombre ont une homologie de séquence très proche voir quasi identique. Pour certaine problématique, comme l'annotation fonctionnelle de nouvelles protéines ou encore pour la construction de modele 3D sur base d'homologie structurale. Il est important de disposer d'une version « curetée » nettoyer des éléments fortement redondants. La nouvelle banque ne contenant qu'un sous-ensemble représentatif de la banque initiale.

Les nouvelles banques sont définies comme les ensembles de séquences ayant moins d'un certain pourcentage d'identité entre elles. Chacune de banques filles est construite via l'outil cd-hit (ref). Plusieurs niveau de redondance sont proposés à partir de la banque « mère » . ils correspondent à des niveaux d'homologie de 90% (3,4), 80%(2,8) et 70% (2,4) produisant respectivement les banques nr (90, 80 et 70). Leur principal avantage est de réduire de manière significative le « bruit » et la taille de la banque ce qui a pour résultat de diminuer le temps de calcul, tout en améliorant la sensibilité des recherches pour les homologues lointains.

Le workflow BioMaJ associé à la construction de la série nr , nr90 est présenté ci-dessous, il illustre la notion de banque calculée :



5.2.2.3 Paramètres spécifiques aux workflow de banques calculées

Une banque calculée est une banque local, il n'est donc pas nécessaire de définir le paramètre **protocol**. Elle ne dépend que d'une et une seule source défini via les propriétés :

db.source=<nom de banque>
data.location=<nom du répertoire à synchroniser>

ainsi pour l'exemple précédant (nr , nr90) nous aurons :

```
# INItialisation
db.fullname="non-redundant protein sequence database with entries from GenPept, Swissprot, PIR, PDF, PDB and RefSeq"
db.name=nr90
db.type=protein
dir.version=nr90
### Synchronization ###
db.source=nr
data.location=cdhit90
remote.files=.*
local.files=.*
```

Tableau 1: la banque calculée nr90 dépend de la banque nr

Dans cet exemple, lors de l'exécution du workflow de la banque calculée nr90, BioMAJ récupère le contenu du répertoire « cdhit90 » de la banque nr (*data.location=cdhit90, db.source=nr*). Pour assurer la validité des fichiers récupérés par nr90, le worflow de nr est exécuté préalablement et s'arrête juste avant la phase déploiement :

Ensuite, seulement commence l'exécution du workflow de nr90 (après celui de nr). Il y a

synchronisation des données à partir du répertoire cdhit90 puis exécution des post-processus. Pour finir le déploiement de la banque calculée et sa banque mère sont effectués simultanément.

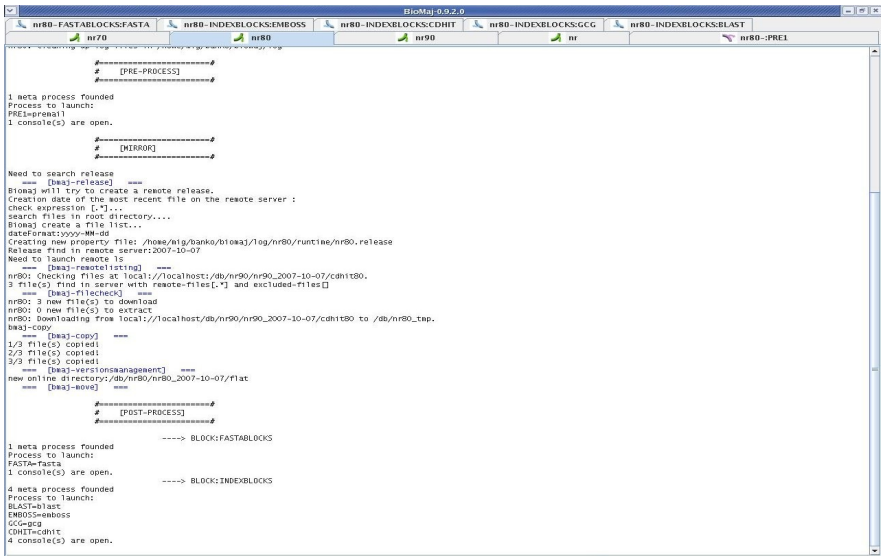


Illustration 23: mise à jour de nr70 qui provoque en cascade les mise à jour de nr80,nr90 et nr

Par conséquent il n est pas nécessaire d'exécuter le workflow de nr puisque nr90 l'exécutera pour chacune de ses sessions

Le mode de fonctionnement des banques calculées comporte des risques.
i) si le workflow de la banque mère (ici nr) est exécuté, les versions des banques filles ne seront pas produitent!
ii) de même par récursivité , si les workflows de nr 80 et nr90 sont exécutés, leurs banques filles -respectivement nr70 et nr80, nr 70 ne seront pas mise à jours.

en conclusion : dans un futur proche, des développements devront sans doutes être réalisés pour régler ce problème. La solution passera sans doute par la définition et l'implémentation d'une notion de dépendance entre banques plus généraliste que celle actuellement en place. L'objectif premier sera d'interdire les incohérences dans la synchronisation des données. Pour l'heure la fonctionnalité banque calculée bien que pas totalement mature, reste très utilise dans certains cas de figure.

6 Méta-donnés et classification des sources et des traitements :

6.1 Définition

- Méta-données :

Selon wikipedia, Une **méta-donnée** (du grec *meta* "après" et du latin *data* "informations") est une donnée servant à définir ou décrire une autre donnée.

- Schéma de classification :

Un **schéma de classification** correspond à l'information descriptive d'une organisation ou division d'objets en groupes, basés sur des caractéristiques communes des objets.

6.2 Implémentation BioMAJ

BioMAJ a pour objectif de gérer des sources et de tracer leurs maintenances. Des propriétés ont fonction de méta-données afin de permettre de classifier les sources et les traitements. Il s'agit de variables définies dans le fichier de propriétés lors de la déclaration de la source ainsi qu'à de la déclaration de chaque traitement.

➢ Source

Les sources peuvent être classés dans un arbre de classification.

Une feuille de l'arbre est défini par son nom BioMAJ , le dbname , sa description : le db.fullname, le formats de ses données, db.formats.

Les classes quand à elles sont définies dans la variable db.type.

Exemple :
```
db.fullname="Genome Gallus gallus (NCBI)"
db.name=Gallus_gallus
db.type=genome
db.formats=gb,fasta,gff
```

➢ Post-traitement

Pour information la propriété **db.type** peut très bien contenir une classification de type : `class1/class2/class3`

Les post-traitements ont également un système de classification similaire. Chaque post-processus dispose d'un champ de classification et de description : exemple:

```
process.name=Ncbi_blast_index
process.desc=formatage de la banque pour ncbi blast
process.type=blast
```

A chaque exécution la production des sessions hérite de ces attributs. Ils sont stockés dans le fichier d'états de la banque et dans un fichier d'index (généré par l'option « --index ») qui contient une vue synthétique des versions de banques contenues dans l'entrepôt local.

6.3 Utilisations potentielles:

Les méta-données BioMAJ peuvent être utilisées à plusieurs niveaux pour aider l'utilisateur et ou les applications à mieux appréhender le contenu de l'entrepôt local. Par exemple, dans le cadre de la génération des rapports pour classifier les sources.

Les fichiers de propriétés livrés avec BioMAJ sont classées en cinq groupes : *nucléique, protéique, nucléique_protéique, génome,* et *autre*.

C'est ces groupes que l'on retrouve dans le rapport web. On pourrait imaginer des classifications plus complexes notamment pour les génomes. Il devrait être possible de rattacher chaque génome à son noeud de classification taxonomique et ainsi améliorer l'information de classification.

Une autre application consisterait à exploiter plus intensivement des formats des données. Du travail reste a faire, mais il pourrait être davantage mise à profit des formats et des types de post-processus pour faire le lien entre applications et sources de données. On pourrait ainsi via le parcours du fichier d'index automatiser la configuration de certaines applications, ou tout simplement permettre à l'utilisateur d'interroger le contenu de l'entrepôt via le format des données.

7 F.A.Q

1) **Mon workflows ne marche plus ?**

   Verifiez le status du workflow (`biomaj.sh -S dbname`) et ensuite analyser les traces (repertoire `$BIOMAJ_ROOT/log/[dbname]/[date]`).

2) **La session est toujours updating ?**

   Verifier que Biomaj est toujours en marche : `ps -aux | grep biomaj`

   si l'application ne fonctionne plus, reportez vous à la question précédente.

3) **Comment automatiser le fonctionnement de BioMAJ ?**

   Il vous suffit de placer un appel à biomaj dans votre crontab (cf. Manuel)

4) **Comment superviser facilement le fonctionnement de l'application?**

   `biomaj.sh -S Mybank` et `make_biomaj_report.sh all`.
   Consultez le resultat dans le répertoire `$BIOMAJ_ROOT/rapport`.

5) **Comment mettre aux point un fichier properties ?**

   Créer votre fichier dans `$BIOMAJ_ROOT/conf/db_properties`.
   Executez pas à pas le workflow via la commande :
   `biomaj.sh -d <dbname> --stage sync`
   pour valider chacune des etapes. Le mode verbose peut s'averer egalement utile.

6) **Comment développer un nouveau post-processus ?**

   Utilisez le langage de votre choix, dans le source du script recuperez les variables d'environnement de la session BioMAJ afin récupérer les informations de contexte de session. Créez un répertoire de résultat et lancer l'appel système en prenant soins de placer les résultats dans ce répertoire. Pour plus d'informations (cf. Manuel chapitre 5)

7) **Comment BioMAJ supervise les postprocessus ?**

   `biomaj.sh --status <dbname>` ; puis faites un tail -f sur le fichier log associé au post-processus.

8) **Comment fonctionne la detection du numéro de version ?**

   Si rien n'est defini pour la récuperation du numéro version, le numéro de version sera la date du fichier le plus récent du serveur distant (ce fichier doit être exprimé dans la propriété **remote.files**) Cette valeur est paramétrable avec la propriété **release.dateformat** et prend comme valeur par défaut : *yyyy-MM-dd*

9) **Mes fichiers de trace sont trop volumineux , que dois je faire ?**

   `biomaj.sh --clean-statefile <dbname>` afin de nettoyer les informations

inutiles. Effacer le répertoire `$BIOMAJ_ROOT/log/[dbname]`

10) Comment faire pour générer une banques a partir de données déja sur place?

Utilisez la notion de banque calculée . (cf. Manuel chap. 5.2.2)

11) Comment savoir quelle banque sont en cours de mise en a jour ?

```
biomaj.sh -S -updating
```

12) Comment connaitre la fréquence de mise a jours d'une banque ?

Si l'appel crontab est fréquent. Consultez les statistiques de la banque afin d'avoir une indication sur la fréquence. Regarder la valeur average update frequency dans le rapport html.

13) Ou trouver du support ?

Mailing list utilisateur : **BioMAJ-users@lists.gforge.inria.fr** , dans les faq sur le site web biomaj.gforge.inria.fr

14) Comment optimiser le telechargement (via un protocole ftp ou http) d'une banque ?

Modifiez le paramètre : **files.num.threads** . Par défaut trois fichiers sont téléchargés en parallèle.

15) BioMAJ est lent comment faire pour accéler ?

Initialisez la propriétés **log.files** à false si un fichier téléchargé correspond à un fichier extrait (ce qui n est donc pas le cas pour une archive tar.gz). Netoyer le fichier d'état :

`biomaj.sh --clean-statefile <dbname>` , n'utilisez pas la console et vérifier la propriété **historic.logfile.level** dans le fichier global.properties.

16) Comment refaire l'indexation d'une banque ?

Si le post-processus gére la dépendance de fichier, il suffit d'éffacer les fichiers résultats et relancer `biomaj.sh -d <dbname>` . Sinom reconstruiser la banque via la commande. `biomaj.sh -rebuild <dbname>`.

17) Comment tester mon post-processus ?

`biomaj.sh -d <dbname> --stage postprocess` biomaj lancera un cylce de mise à jours des données et arretera la session apres les post-processus. Si les données sont déjà en place il lancera le post-processus. Vous recevrez un rapport mail contenant les erreurs eventuelles.

18) Comment configurer blast pour le rendre compatible avec BioMAJ ?

Il vous suffit de créer un répertoire repository de bank blast. Ensuite initialisez la variable avec le chemin absolue du répertoire `export BLASTDB = MYREP/BLAST/` déclarez exactement la variable dans le fichier `$BIOMAJ_ROOT/bin/env.sh`, enfin personnalisez le chemin d'accès aux exécutables - formatdb fastacmd blastall- dans le fichier `$BIOMAJ_ROOT/conf/process/unix_command_system.cfg`

19) Comment configurer SRS pour le rendre compatible avec BioMAJ ?

Modifier le fichier srsdb.i de l'application, initialisez en tête du fichier les variables globales

suivantes :

dataRoot:'/bank' ==> mettez la valeur de data.dir, onData:'current/flat/', onIndex:'current/srs/', offData:'futur_release/flat/', offIndex:'futur_release/srs/'. Pour chaque banque indexée : écrivez explicitement la valeur du chemin dir.version de la bank aprés le chemin $dataroot (la racine du repository) pour les offDir , indexDir et offIndexDir. dir:"($dataRoot)*/ebi/uniprot/*($onData)", offDir:"($dataRoot)*/ebi/uniprot/*($offData)", indexDir:"($dataRoot)*/ebi/uniprot/*($onIndex)", offIndexDir:"($dataRoot)*/ebi/uniprot/*($offIndex)"

definissez les variables suivantes dans le fichiers : $BIOMAJ_ROOT/conf/process/unix_command_system.cfg, EXECUTE_BATCH_OPTION_PBS_SRS=-q srsq -W block=true -A SRS -j oe -V, GETZ=/data/srs/srs/bin/linux73/getz, SRSCHECK=/data/srs/srs/etc/srscheck. Enfin initialisez l'environnement srs dans le fichier d'environnement BioMAJ :($BIOMAJ_ROOT/bin/env.sh) source /MYPATH/OF/srs/etc/prep_srs.sh

8 Annexes

8.1 Exemple de Fichiers de configuration

8.1.1 Global.properties

```
# File: global.properties

#----------------
# Mail Configuration
#----------------
#Uncomment thes lines if you want receive mail when the workflow is finished

#mail.smtp.host=
#mail.admin=
#mail.from=

#---------------------
#Proxy authentification
#---------------------
#proxyHost=
#proxyPort=
#proxyUser=
#proxyPassword=

#---------------------
# PROTOCOL
#-------------------
#possible values : ftp, http, rsync, local
protocol=ftp
port=21
username=anonymous
password=anonymous@nowhere.com


#The root directory where all databases are stored.
#If your data is not stored under one directory hirearchy
#you can override this value in the database properties file.
data.dir=/local/db2
production.directory.chmod=775
bank.num.threads=1

#
# Global system properties file

#Number of threads to use for downloading and processing
files.num.threads=3

#to keep more than one release increase this value
keep.old.version=0
#Link copy property
do.link.copy=true

#look here to specified a new format :
http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html
release.dateformat=yyyy-MM-dd

#The historic log file is generated in log/
#define level information for output : DEBUG, VERBOSE,INFO,WARN,ERR
#to have info about target and task ant use VERBOSE
historic.logfile.level=VERBOSE
historic.logfile.properties=false
historic.logfile.task=true
historic.logfile.target=true


PRE1=premail

premail.name=sendMail
premail.exe=sendMailTLSE.pl
premail.args=-s '[NCBI Blast - db.name] Start Biomaj session' -m 'local.time'
```

```
premail.desc=mail
premail.type=info

http.parse.dir.line=<a[\\s]+href=\"([\\S]+)/\".*alt=\"\\[DIR\\]\">.*([\\d]{2}-[\\w\\d]{2,5}-
[\\d]{4}\\s[\\d]{2}:[\\d]{2})
http.parse.file.line=<a[\\s]+href=\"([\\S]+)\".*([\\d]{2}-[\\w\\d]{2,5}-[\\d]{4}\\s[\\d]{2}:
[\\d]{2})[\\s]+([\\d\\.]+[MKG]{0,1})

http.group.dir.name=1
http.group.dir.date=2
http.group.file.name=1
http.group.file.date=2
http.group.file.size=3

#Needed if data sources are contains in an archive
log.files=true

local.files.excluded=\\.panfs.*

#~40mn
ftp.timeout=2000000
ftp.automatic.reconnect=2

wget.options=--tries=inf --wait=5 --random-wait --passive-ftp --timeout=50
```

8.2 Tableaux des propriétés Biomaj

propriétés	***valeurs***	***description***	***optionnel***	***Default value***	***Default location***
proxyHost	<host.proxy>	L'adresse du serveur de proxy	Oui	facultatif	global.properties
proxyPort	<port.proxy>	Le port du serveur de proxy	Oui	facultatif	global.properties
proxyUser	<login>	Login si proxy avec authentification	Oui	facultatif	global.properties
proxyPassword	<mdp>	Mot de passe si proxy avec authentification	Oui	facultatif	global.properties
mail.smtp.host	<hostname.Domain.xx>	Le serveur smtp pour l'envoi de mail	oui	facultatif	global.properties
mail.admin	<admin@MDomain.xx,**>	Mail de l'administrateur des banques	Non si *mail.smtp.host definie*	facultatif	global.properties ou dbname.properties
Mail.from	<admin@MDomain.xx>	Mail de l'administrateur des banques	Non si *mail.smtp.host definie*	facultatif	global.properties ou dbname.properties
port	Entier	Indication du port pour le protocole ftp	oui	21	global.properties ou dbname.properties
username	String	Nom d'utilisateur pour le protocole ftp	oui	anonymous	global.properties
password	String	Password pour le protocole ftp	oui	- (Your mail!)	global.properties
ftp.timeout	Entier positif ou -1	Time out en mms pour le protocol ftp. Si -1 pas de time out	oui	100000	global.properties

propriétés	*valeurs*	*description*	*optionnel*	*Default value*	*Default location*
ftp.automatic.reconnect	Entier positif	Nombre de reconnexion automatique pour le protocole ftp	oui	5	global.properties
Wget.options	String	Argument pour wget	Oui	--tries=inf --wait=5 --random-wait --passive-ftp --timeout=50	global.properties
data.dir	Chemin de répertoire	Répertoire racine des banques en local	non	To be define	global.properties
production.directory.chmod	String	Futur droit d'accès du répertoire en production	non	755	global.properties
bank.num.threads	Entier	Gestion du nombre banque géré en simultané	non	1	global.properties
files.num.threads	Entier	Nombre de fichier téléchargé en simultané	non	1	dbname.properties
Log.files	Booleen	Journalise des informations sur les fichiers téléchargés	Non	True	global.properties
protocol	ftp,http,rsync,local	Le protocole utilisé pour obtenir la banque	non	ftp	global.properties
do.link.copy	link,copy	Autorise des liens symboliques sur des fichiers en production	oui	true	global.properties
keep.old.version	Entier	nombre de release complete sauvegardé	oui	0	global.properties
frequency.update	Entier	Nbre de jours pendant laquelle la banque ne nécessite pas de mise à jour	oui	0	global.properties
release.dateformat	Expression DateFormat	Spécification du format de la release http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html	non	yyyy-MM-dd	global.properties
historic.logfile.level	DEBUG,VERBOSE,INFO,WARN,ERR	Définition du niveau d'affichage des logs et de la console	oui	DEBUG	global.properties
historic.logfile.properties	true,false	Écriture des propriétés ant définit par l'application dans les logs	oui	false	global.properties
historic.logfile.task	true,false	Écriture du nom des taches du workflow dans les logs	oui	false	global.properties
historic.logfile.target	true,false	Écriture du nom des meta-tache du workflow dans les logs	oui	false	global.properties
db.name	Cette propriété doit porter le même nom que le fichier properties.	Le nom de la banque virtuelle	non	-	dbname.properties
db.fullname	String	Le nom complet	non	-	dbname.properties

propriétés	*valeurs*	*description*	*optionnel*	*Default value*	*Default location*
db.type	String	Le type de banque	oui	-	dbname.properties
server	String	Adresse du serveur	non	-	dbname.properties
offline.dir.name	Chemin	Répertoire temporaire pour le téléchargement et les extractions	oui	(Facultatif) datadir/offdir/bname_tmp	dbname.properties
dir.version	Chemin	Répertoire de version de la banque	oui	(Facultatif) Datadir/dbname/	dbname.properties
remote.dir	Chemin	Localisation du répertoire où se trouve la banque sur le site distant	non	-	dbname.properties
remote.files	Expressions régulières	Filtre sur les fichier à télécharger	non	-	dbname.properties
Remote.excluded.files	Expressions régulières	Filtre sur les fichiers à ne pas téléchargés pouvant « matcher » avec remote.files	Oui	-	dbname.properties
local.files	Expressions régulières	Filtre sur les fichier à mettre en production	non	-	dbname.properties
local.files.excluded	Expressions régulières	Filtre sur les fichier à ne pas mettre en production	oui	-	
release.file	String	Non du fichier où se trouve la release	oui	-	dbname.properties
release.regexp	Expressions régulières	Expression régulière pour récupérer la release	oui	-	dbname.properties
release.file.compressed	true,false	Vrai si le fichier contenant la release est compressé	oui	false	dbname.properties
no.extract	True,false	Si Vrai, Supprime la phase de décompression des fichiers tar,gz,etc...	Oui	False	dbname.properties
list.files.available	True,false	Autorise l'initialisation des variables d'environement de liste de fichier (`RELEASE_ALL_COMPRESSED_FILES_LIST`,`RELEASE_ALL_UNCOMPRESSED_FILES_LIST`,`RELEASE_OLD_FILES_LIST`,`RELEASE_NEW_FILES_LIST`) pour tous les postraitemens.t	Oui	False	dbname.properties