Paraloop: une boucle parallèle intégrée à une chaîne de traitements

Emmanuel Courcelle et Jérôme Gouzy
Laboratoire des Interactions Plantes Micro-organismes,
UMR CNRS-INRA 2594-441, B.P. 52627, 31326 CASTANET-TOLOSANE Cedex, France
Emmanuel.courcelle@toulouse.inra.fr, jerome.gouzy@toulouse.inra.fr

http://lipm-bioinfo.toulouse.inra.fr/paraloop

Une chaîne de traitements:

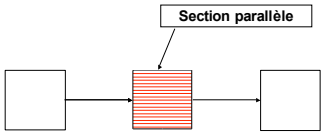
On rencontre fréquemment le cas suivant: :

- Le fichier d'entrée est une suite d'enregistrements.
- On doit traiter chaque enregistrement de manière indépendante.
- Il est donc simple, d'un point-de-vue algorithmique, de paralléliser cette boucle ...

Mais d'un point-de-vue pratique, c'est une autre affaire !

Le programme écrit dépendra :

- Du traitement.
- Du format des données d'entrée.
- De l'architecture de la machine utilisée.

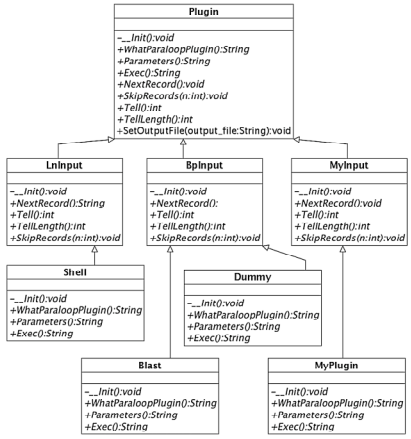


- **Comment éviter de réécrire tout lorsqu'on change de machine ?**
- **Comment ne pas réécrire sans cesse les routines d'entrée-sortie ?**
- **Comment s'assurer que les événements importants sont « loggés » ?**
- **Comment faire en sorte que le programme puisse être interrompu et redémarré à tout moment ?**

paraloop, un script perl orienté objets

paraloop est un **programme appelé à partir du script principal** pour traiter les **sections parallèles** :

- Le code correspondant à *l'architecture* de la machine est encapsulé dans un objet appelé « **Scheduler ».**
- Le code correspondant au *traitement* réalisé est encapsulé dans un objet appelé « **Plugin »** (cf. ci-contre) . Le mécanisme **d'héritage** permet de factoriser le code correspondant à l'accès aux fichiers **d'un format donné.**
- Les paramètres permettant d'agir sur le Scheduler et le Plugin sont regroupés dans un ou plusieurs fichiers de paramètres.
- **Il n'est pas nécessaire d'être root pour installer paraloop.**



Lorsqu'on change de machine, ***la ligne de commande reste la même.***

Tout est loggé, *avec différents niveaux de* verbosité.

On peut demander à paraloop de s'arrêter : il s'interrompra entre deux traitements, sauvera son état, et sera capable de redémarrer à partir de l'enregistrement suivant.

Entrées-sorties

```
paraloop.pl -cfg paraloop.cfg -prog Blast -input file.fast -db database -output output/file.fasta -ncpus 10 --wait
```

Contient, entre autres, les caractéristiques de la machine

Nom du plugin utilisé, il encapsule votre code

Attend que tous les processeurs aient fini leur travail pour rendre la main

Les architectures supportées

La distribution contient des « schedulers » pour les architectures suivantes :

- **SMP**
  (on utilise un fork pour lancer plusieurs processes)
- **clusters de calcul**
  (on utilise rsh/ssh pour lancer le calcul sur les différents noeuds)
- **Machines utilisant un système de queue** (PBS)
  (PBS se charge de la distribution des jobs)
- ... Si le type de machine dont vous disposez n'est pas supporté par paraloop vous pouvez écrire votre propre scheduler.

Téléchargement et site web

- Vous pouvez télécharger le programme à partir de `ftp://ftp.toulouse.inra.fr/pub/paraloop`
- paraloop est un logiciel libre, il est couvert par la license CeCILL version 2 (adaptation de la GPL au droit français)
- Si vous avez développé un nouveau plugin ou scheduler, *merci de nous l'envoyer*, nous serons heureux de l'intégrer à la distribution standard.
- Sur `http://lipm-bioinfo.toulouse.inra.fr/paraloop`:
  La documentation complète
  Une Foire Aux Questions
  Des forums, ...

Utilisation d'un système de queue via paraloop

Soumettre un travail en *batch* via paraloop assure une **meilleure utilisation** des ressources disponibles :

- Les **paramètres du qsub** (reservation de ressources) peuvent être écris dans le fichier de paramètres.
- Plusieurs tests sont réalisés **avant** l'appel du qsub, ce qui permet de détecter certaines erreurs (mauvais nom de fichier d'entrée, ...)
- On peut utiliser paraloop **sur la machine frontale**, à des fins de test (mise au point des paramètres).
- Lors de l'exécution de très gros jobs mettant en jeu de nombreux enregistrements, il est possible **d'interrompre** paraloop et de le **redémarrer** aisément.
- Lorsque c'est possible, paraloop s'interrompt **avant** d'être tué par le système de queue pour cause de dépassement de temps CPU. Avant de s'interrompre, il se "resoumet" lui-même, de sorte qu'il est prêt à redémarrer automatiquement, sans gaspillage de CPU.