Torque (et MAUI) : une solution pour construire simplement une ferme de calcul.

Jacques Foury

Institut de Mathématiques de Bordeaux

Journée *JoSy* Septembre 2007

Plan

Constats

- les systèmes d'exploitations courants sont maîtrisés dans nos laboratoires
- ces machines sont la plupart du temps mal utilisées pour le calcul : le nombre de calculs simultanés et leur utilisation mémoire n'est nullement contrôlée
- l'administrateur système n'a pas toujours le temps de se pencher sur les technologies de fermes de calcul... les utilisateurs préfèrent lancer leurs calculs en direct et '' *voir*'' ce qui se passe pendant le calcul
- les systèmes "tout en un" (OSCAR, ROCKS) écrasent tout, et imposent *leur* gestion, *leur* système d'exploitation...
- comment concilier tout ça, et faire tout de même une ferme de calcul avec soumission de batches, répartition de charge, et éventuellement politique d'usage ?

Constats

Exemple : une machine de calculs Linux

- les utilisateurs lancent leurs calculs sans regarder la charge en cours
- la mémoire est potentiellement saturée
- Charge système > nombre de CPUs : fort ralentissement
- Mémoire saturée : *swap* : très fort ralentissement
- Possibilité de plantage de la machine $\Rightarrow$ tous les calculs en cours sont alors plantés !

Constats

Le noeud de calcul

- C'est une machine banalisée (mais c'est mieux si elle ne fait que du calcul)
- Pas de modification du système d'exploitation : on installe un composant logiciel simple (ou deux)
- La machine peut garder un usage interactif... à vos risques et périls !

Solution

- Il faudrait proposer un système simple d'usage, qui contrôle en temps réel l'utilisation des machines
- qui gère aussi un nombre de machines plus grand
- qui ordonnance les calculs, en fonction du temps de calcul, de la mémoire utilisée
- qui permet aussi de suivre et mesurer les calculs en cours
- qui répartit les calculs, selon leurs besoins, sur les différentes machines de la ferme

Solution

C'est ce qu'on se propose de faire

- **Torque + MAUI** est largement adopté
- Dans les systèmes de grande taille (océanographie, recherche pétrolière...)
- Ce sont les logiciels embarqués dans les "tout en un", OSCAR, ROCKS...

Torque+MAUI

solution logicielle

- pas de modification du noyau, ni de l'OS
- solution légère, seulement 2 composants logiciels sur le serveur, et un sur chaque noeud de calcul
- multiplate-forme (Linux, Mac OS X, Windows, divers Unix...)
- gère de nombreuses (petites) machines ensemble, à la manière d'un gros SMP
- permet de passer un temps minimum sur le déploiement de la solution

Solution peu intrusive

Deux couches logicielles

- logiciel en couche applicative, pas de modification du système d'exploitation
- 2 processus en tout sur le serveur, 1 processus sur les noeuds
- installation très simple (rpm, deb...) sur les Linux

Solution peu intrusive

Contraintes

- Système de fichiers partagé sur tous les noeuds (non nécessaire sur le maître)
- Les noeuds et le maître se connaissent (DNS, /etc/hosts)
- Pour soumettre depuis un noeud différent du noeud maître : /etc/hosts.equiv

Principe de fonctionnement

Torque, le gestionnaire de la ferme

Gère les ressources matérielles sur les noeuds de la ferme

- les processeurs/coeurs de calcul
- la mémoire

Gère les files d'attente des batches

- (*optionnel*) files de routage : distribue les jobs selon les critères prédéfinis
- files d'attente : lancement des jobs, arrêt, retour des sorties standard/erreur

Principe de fonctionnement

MAUI : l'ordonnanceur

- L'ordonnancement : beaucoup plus compliqué qu'on ne le pense au premier abord
- Décisionnel : quel job a priorité, selon quels critères ?
- Choix des noeuds qui exécuteront le job ?
- Surveillance des jobs en cours de calcul : sont-il finis, respectent-t'ils les demandes en ressources ?
- Possibilité d'arrêter provisoirement les jobs qui dépassent la mémoire demandée $\Rightarrow$ vérification humaine

Peu de fichiers

Sur le serveur

- Un pour Torque
- Un pour MAUI
- Un pour lister les noeuds
- Possibilité de modifications en direct, sans passer par les fichiers
- Beaucoup, **beaucoup** de possibilités (documentation très importante)

Peu de fichiers

Sur les noeuds

- Un pour dire qui est le serveur
- Un pour la gestion plus fine : vitesse relative du noeud, méthode de retour des fichiers de sortie...

Historique

- Nous sommes partis sur une politique appliquant simplement le "bon usage"
- Nous avons créé des files d'attente uniquement selon le temps demandé
- Rapidement un constat : sans imposer de limite, de cadre, les chercheurs utilisent sans compter $\Rightarrow$ dépassements de capacité, et ne spécifient pas toujours les minima
- Nous avons refondu les files d'attente, créé des files de routage, contraint à spécifier les critères minimum
- Les chercheurs doivent donner à chaque job soumis : le nom, le temps maximum (walltime), la mémoire demandée, l'architecture (32 ou 64 bits)

Politique déployée à l'IMB

- 2 architectures (32 et 64 bits) $\Rightarrow$ 2 clusters, virtuellement. L'utilisateur choisit par le nom dans quelle architecture va tourner son calcul.
- Les limites sont posées pour éviter les dépassements de capacité CPU ou mémoire : chaque file d'attente impose des limites
- Files d'attente selon le temps, le choix est fait automatiquement en fonction des demandes de l'utilisateur

Politique déployée à l'IMB

- MAUI surveille l'utilisation mémoire, et arrête provisoirement les jobs qui dépassent
- $\Rightarrow$ on intervient régulièrement, pour dialoguer avec le chercheur, comprendre et faire comprendre, puis relancer le job (2 commandes en ligne)
- On a consolidé en virtualisant le maître, qui utilise très peu de ressources.
- On surveille via Ganglia l'usage de la ferme.

Les paramètres de Torque

Fichiers du serveur - extraits

- Le fichier server_priv/nodes liste les noeuds de calcul et leurs propriétés
- Le fichier de paramètres principal est interprété par "qmgr". Il déroule une série de lignes "set contexte variable = valeur" :

La liste des noeuds du cluster

```
...
callas10 np=4 opteron callas
callas11 np=4 opteron callas
callas99 np=8 opteron callas
callas00 np=4 opteron callas
callas01 np=2 opteron callas
callas02 np=2 opteron callas
...
abeille04 np=2 xeon ruche
abeille05 np=2 xeon ruche
...
```

maui

Le fichier du serveur

```
set server default_queue = ruche
set server scheduler_iteration = 200
set server node_ping_rate = 300
set server node_check_rate = 600
set server tcp_timeout = 6
set server node_pack = True
```

Les files d'attente : type exécution

```
create queue q1jour
set queue q1jour queue_type = Execution
set queue q1jour resources_max.walltime = 24:00:00
set queue q1jour resources_default.walltime = 24:00:00
set queue q1jour resources_min.walltime = 1:00:00
set queue q1jour resources_max.mem = 2000mb
set queue q1jour resources_default.mem = 100mb
set queue q1jour resources_default.neednodes = ruche
set queue q1jour resources_default.nodes = 1:ruche
set queue q1jour enabled = True
set queue q1jour started = True
```

Les files d'attente : type routage

```
create queue ruche
set queue ruche queue_type = Route
set queue ruche route_destinations += q1heure
set queue ruche route_destinations += q1jour
set queue ruche route_destinations += q1semaine
set queue ruche route_destinations += infini
set queue ruche resources_default.walltime = 01:00
set queue ruche resources_default.mem = 10mb
set queue ruche enabled = True
set queue ruche started = True
```

Les paramètres de Torque

Fichiers des noeuds - extraits

Fichier server_name : <NomDuServeur>

Fichiers des noeuds - extraits

Fichier mom_priv/config : personnalisation des noeuds

fichier mom_priv/config

```
## permet de savoir si on peut lancer
## encore des jobs sur ce noeud
## (ici on a un bi-processeur)
$ideal_load 2.0
$max_load 3.5

## Le serveur qui a le droit de gérer le noeud
$clienthost ulmo.math.u-bordeaux1.fr

## Comment on fait la copie des fichiers à
## la fin des jobs (ici copie sur NFS)
$usecp *:/home /home
$usecp *:/scratch /scratch
```

Les paramètres de MAUI

maui.cfg

- Enormément de paramètres possibles
- Se limiter au minimum, sous peine de conflit de paramètres
- Choisir une politique, lui appliquer les paramètres qui conviennent, éviter de mélanger
- Tout ce qui est dans la doc ne fonctionne pas : MAUI est la version libre de MOAB

fichier maui.cfg <1>

```
SERVERHOST              ulmo.math.u-bordeaux1.fr
ADMIN1                  root
ADMIN2 labesse depouill foury

## politique de fairshare
RESOURCEWEIGHT 20
XFACTORWEIGHT 100
FAIRSHAREWEIGHT 100

## politique de remplissage sur les noeuds
BACKFILLPOLICY          BESTFIT
RESERVATIONPOLICY       CURRENTHIGHEST
NODEALLOCATIONPOLICY FIRSTAVAILABLE
```

XFactor ▸▸ zapper xfactor

XFACTOR

XFactor

- XFACTOR = 1 + <QUEUETIME> / <EXECUTIONTIME>
- Paramètre d'autant plus important que le temps passé à attendre est grand...

retour

fichier maui.cfg <2>

```
## Gestion des limites
# 1 heure de depassement autorise
JOBMAXOVERRUN 01:00:00

# si la mémoire demandée dans le batch
# est dépassée par le processus => job suspendu
# mémoire job parallèle = somme(processus)
RESOURCELIMITPOLICY MEM:ALWAYS:SUSPEND
#RESOURCELIMITPOLICY PROC:ALWAYS:SUSPEND
#RESOURCELIMITPOLICY SWAP:ALWAYS:SUSPEND

# partitionnement du cluster, permet d'étanchéifier
# les 2 parties
CLASSCFG[callas]
CLASSCFG[ruche]
```

voir le fichier nodes

Références

- Cluster Resources
- Pages de la cellule de l'IMB
- ClusterBuilder : pour voir plus large
- Un howto sommaire