MIF32 - Compte-Rendu du projet CAN

Adrien FAURE(pXXXXXXXX) & Raphaël CAZENAVE-LEVEQUE (p1410942)

2014-2015

Abstract

Nous présentons une implémentation en C/MPI d'une DHT utilisant le protocole CAN. Nos choix d'implémentation permettent une recherche efficace dans la DHT depuis n'importe quel noeud. De nombreux tests unitaires sont présents dans notre projet. Nous offrons aussi des représentations graphiques en SVG et textuelles de l'état de la DHT. Ces représentations sont générées automatiquements lors de l'execution de la DHT, ou à la demande si l'on utilise l'interface en ligne de commande que nous avons écrite.

1 Fonctionalités disponibles

Les cinq étapes ont été

2 Choix de conception

3 Tests unitaires

Au cours du projet, il est devenu nécessaire pour nous d'écrire des tests unitaires afin d'accélérer le débuggage et d'assurer le bon fonctionnement de notre code. Nous avons donc isolé les zones de codes comportant nos algorithmes des zones utilisants MPI, cela nous a permit d'obtenir une bonne couverture du code par les tests, et d'être relativement sûr du comportement de notre code.

4 Algorithme de routage

5 Relocation des données

6 Les structures de données mises en oeuvre

Chaque noeud possède une liste capable de stocker les données dont le noeud est responsable. Ces données sont encapsulées dans une structure afin de pouvoir y conserver la position des données.

Les noeuds possèdent une liste de frontières représentant le voisinage proche du noeud.

7 Ajout dynamique de processus lors de l'execution

Nous avons souhaité ajouter la possibilité d'ajouter des processus au cours de l'execution. Nous avons effectué des essais en utilisant *MPI_Spawn*, puis *MPI_Intercomm_merge* pour obtenir un communicateur intra contenant tous les processus ainsi que le nouveaux processus. Malheureusement, il n'est pas possible de re-merger ce communicateur [1] lors de spawn successifs, car *MPI_Intercomm_merge* est une routine collective et donc chaque processus précédément spawné doivent pouvoir participer au prochain merge.

A Annexe 1 - README

```
# pour builder et tester:
```
make
```

# Pour builder et executer les tests
```
make runtest
```

# Pour builder sans executer les tests
```
make test
```

# pour builder l'appli ni executer les tests (snif)
```
make mpi_can
```

# execution avec prompt :
```
mpirun -np 7 ./mpi_can debug
```

# execution sans prompt :
```
mpirun -np 7 ./mpi_can
```
```

[1] Nous avons du abandonner cette idée suite à la lecture de ce topic de la mailling-list d'openMPI - https://www.open-mpi.org/community/lists/users/2007/10/4312.php

```
# execution avec prompt et en fixant une seed spécifique au générateur pseudo-aléat
```
```
mpirun -np 7 ./mpi_can seed 42 debug
```

```
# execution sans prompt et en fixant une seed spécifique au générateur pseudo-aléat
```
```
mpirun -np 7 ./mpi_can seed 42
```

```
# Prompt:
```
```
> status     : show log about the state of the DHT
> insert 2   : insert the node 2 in the overlay
> insert all : insert all nodes in the overlay
> log        : add a textual/SVG log on logs/ directory
```