DOCUMENT SUR LE GARBAGE COLLECTOR DE BOEHMS:

I- TELECHARGEMENT ET INSTALATION DE GC:

Le téléchargement de GC suit la procédure suivante:

Creer le repertoire projet à l’aide de la commande: **mkdir projet**

Cloner le projet via la lien github:

**git clone git://github.com/ivmai/bdwgc.git**

Cloner la librairie libatomic\_ops via le lien github suivant:

**git clone git://github.com/ivmai/libatomic\_ops.git**

Créer un lien symbolique de la librairie libatomic\_ops dans GC à L’aide de la commande: (ici j’ai considéré que le repertoire projet etait sur le bureau)

**ln – s /*home*/armel/Desktop/projet/libatomic\_ops /*home*/armel/Desktop/projet/bdwgc/libatomic\_ops**

Aller dans le repertoire de GC avec la commande:

**cd /*home*/armel/Desktop/projet/bdwgc**

//Exécuter les commandes suivantes: **autoreconf -vif** et **automake –add-missing**

**II- TELECHARGEMENT DE PHOENIX MAPREDUCE:**

Pour télécharger phoenix MapReduce suivre la procédure suivante:

Aller dans le répertoire projet:

**cd /*home*/armel/Desktop/projet/**

Cloner le projet Phoenix à l’aide de la commande suivante:

**git clone https://github.com/kozyraki/phoenix**

Aller dans le dossier src de phoenix-2.0 à l’aide de la commande:

**cd /*home*/armel/Desktop/projet/phoenix/phoenix-2.0/src/**

Executer la commande suivante: **make**

**III- COMPILATION ET INTEGRATION DE GC DANS PHOENIX:**

Pour compiler GC on doit saisir les commandes suivantes:

**make clean**

**./configure --disable-threads --prefix=/home/armel/Desktop/projets/projet/phoenix/phoenix-2.0 --enable-static --enable-checksums --enable-mmap –disable-munmap --disable-shared**

**→** enable-static : pour générer les lib .a

→ disable-shared: pour que les lib .so ne soient pas générées dans le préfix (sinon elles sont utilisées par défaut à la place des .a)

→ enable-checksums : nécessaire pour les logs s’affichent

→ disable-munmap : nécessaire pour enable le checksums

**//--disable-silent-rules**

**make**

**make check**

**make install**

Le paramètre --prefix dans la commande ./configure permet de specifier ou GC durant le make install ira copier les fichiers de GC dans phoenix. Toujours dans configure on active la géneration de la librairie statique et on desactive la géneration de la librairie dynamique, on specifie a GC que lors de la première allocation initiale (le gros malloc donc parlait le prof il utilise la methode mmap), on désactive aussi le mode silencieux et les threads.

La commande make check qui est optionnelle permet juste d’executer une série de tests.

La commande make install copie les fichiers de GC dans phoenix mapReduce.

**IV- EXECUTION DES APPLICATIONS CLIENTES DANS PHOENIX MAPREDUCE:**

il faut d’abord inclure gc.h et remplacer les appels de malloc

Dans un .h dans le dossier include/ mettre :

#define malloc(n) GC\_MALLOC(n)

#define calloc(m,n) GC\_MALLOC((m)\*(n))

#define free(p) GC\_FREE(p)

#define realloc(p,n) GC\_REALLOC((p),(n))

Puis inclure ce .h dans stddefines.h

Avant la compilation de chaque programme ne pas oublier de rajouter dans le Makefile :

GC=gc

Et à la fin de la directive LIBS:

-l$(GC)

**Activer le mode incremental et avoir les stats à l’exécution**

**→ dans le terminal**

GC\_ENABLE\_INCREMENTAL='GC\_ENABLE\_INCREMENTAL'  
GC\_PRINT\_VERBOSE\_STATS='GC\_PRINT\_VERBOSE\_STATS'  
GC\_USE\_GETWRITEWATCH='GC\_USE\_GETWRITEWATCH'export GC\_ENABLE\_INCREMENTAL  
export GC\_PRINT\_VERBOSE\_STATS  
export GC\_USE\_GETWRITEWATCH

pour le faire pour tous les utilisateurs:

\* dans /etc/profile.D/ rajouter un script boehm.sh dans lequel mettre tous les exports précédents et redémarrer (il n’y aura donc plus besoin de le faire pour chaque terminal ou user)

→ dans chaque appli (.c)

#define SOFT\_VDB

\* dans le main

GC\_INIT();

GC\_enable\_incremental();

Pour executer les applications dans phoenix hummm c’est un peu compliqué car après avoir fait le make install dans gc qd tu vas dans:

**cd /*home*/armel/Desktop/projet/phoenix/phoenix-2.0/test**

et que tu exécutes:

**make**

hummm en principe le Makefile de la racine du projet est censé compiler tous les applications dans le dossier tests de phoenix mais à cause des configurations compliquées de GC cela ne marche pas toujours à tout les coups, je te conseilles de le faire programme par programme et de voir ceux qui posent les problèmes.

Mais voici comment moi je compile l’application word\_count-seq que j’utilises souvent lors des présentations.

**cd /*home*/armel/Desktop/projet/phoenix/phoenix-2.0/tests/word\_count/**

**cc -I/home/armel/Desktop/projets/projet/phoenix/phoenix-2.0/include/ -c word\_count-seq.c -o word\_count-seq.o**

**gcc -o word\_count-seq word\_count-seq.o -L/home/armel/Desktop/projets/projet/phoenix/phoenix-2.0/lib/ -lgc -lphoenix**

Pour executer le programme tu devras téléchargé les fichiers de test qui seront passé en paramètre au programme. Voici le lien de téléchargement des fichiers de test:

- [histogram](http://csl.stanford.edu/~christos/data/histogram.tar.gz)

- [linear regression](http://csl.stanford.edu/~christos/data/linear\_regression.tar.gz)

- [string match](http://csl.stanford.edu/~christos/data/string\_match.tar.gz)

- [reverse index](http://csl.stanford.edu/~christos/data/reverse\_index.tar.gz)

- [word count](<http://csl.stanford.edu/~christos/data/word_count.tar.gz>)

Une fois après le téléchargement tu les extraits. Moi je les ai laissé dans le dossier téléchargement puis tu lances le programme avec la commande:

**./word\_count-seq /home/armel/Downloads/word\_count/word\_count\_datafiles/word\_100MB.txt**