



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



HIGH PERFORMANCE CODING HPC

THREADING BUILDING BLOCKS

João Miguel Domingues Pedrosa
Loïc Haas

2 juin 2016

Table des matières

1	Introduction	2
2	Installation	3
2.1	Installation via apt-get	3
2.1.1	Code exemple	3
2.2	Installation via l'archive	3
3	Exemple	5
3.1	Code	5
3.2	Mesures	5
4	Analyse	6
5	Bibliographie	7

1 Introduction

L'objectif de ce projet est de comprendre, essayer et exploiter un outil d'optimisation. Dans notre cas, nous avons choisi TBB (Threading Building Blocks). Il s'agit d'une librairie d'Intel fait pour du C++.

Sa caractéristique est de simplifier au maximum l'implémentation de programme parallèle pour des systèmes multicœur. Le programmeur pourra ainsi faire un programme portable car c'est la librairie qui va se charger d'utiliser la bonne implémentation de thread (exemple : POSIX pour linux ou les threads Windows). Pour cela, elle met en place différentes fonction et objet lié à la programmation parallèle et à la gestion de concurrence.

Pour ce projet, nous avons du faire l'installation de la librairie, la procédure sera expliqué plus loin dans le rapport. Il y aura aussi un code exemple auquel on aura fait des mesures de performances afin de voir les optimisation apporté. Nous finirons par une analyse de notre constats tout au long de nos essais.

2 Installation

L'installation des libraires a été fait sur des machines possédant un OS Linux Ubuntu. Les installations suivantes ont été via ligne de commande.

2.1 Installation via apt-get

Pour installer sur nos machines, nous avons utilisé le gestionnaire de package `apt-get`. Le package installé est `libtbb2`. Cela nous donne la commande suivante :

```
sudo apt-get install libtbb2
```

Une fois installé, on peut commencer à programmer avec les librairies `threads`. Pour tester l'installation, on va compiler et exécuter un code exemple afin de voir s'il n'y a pas de problème. (commande pour la compilation `g++ <filename>.cpp -ltbb`)

2.1.1 Code exemple

```
1 //src: http://stackoverflow.com/questions/10607215/simplest-tbb-example
2 //Ce code se contente d'afficher les valeurs d'un vecteur
3 //Evidemment l'ordre d'apparition sera différent à chaque exécution
4 //car on ne sait pas quel thread aura la main à quel moment
5 #include "tbb/parallel_for_each.h"
6 #include "tbb/task_scheduler_init.h"
7 #include <iostream>
8 #include <vector>
9
10 struct mytask {
11     mytask(size_t n)
12         : _n(n)
13     {}
14     void operator() () {
15         for (int i=0; i<1000000; ++i) {} // Deliberately run slow
16         std::cerr << "[" << _n << "]"<br>";
17     }
18     size_t _n;
19 };
20
21 template <typename T> struct invoker {
22     void operator() (T& it) const {it();}
23 };
24
25 int main(int, char**) {
26
27     tbb::task_scheduler_init init; // Automatic number of threads
28     // tbb::task_scheduler_init init(4); // Explicit number of threads
29
30     std::vector<mytask> tasks;
31     for (int i=0; i<1000; ++i)
32         tasks.push_back(mytask(i));
33
34     tbb::parallel_for_each(tasks.begin(), tasks.end(), invoker<mytask>());
35     std::cerr << std::endl;
36
37     return 0;
38 }
```

2.2 Installation via l'archive

1. Il faut premièrement télécharger. Pour cela, il faut se rendre sur le site d'Intel pour TBB et aller dans la section download¹. Ensuite, on copie l'adresse sur la release et l'OS que l'on veut (dans notre cas Linux). On entre ensuite la commande suivante :

```
$ wget https://www.threadingbuildingblocks.org/sites/default/files/software_releases/linux/tbb<version>.tgz
```

2. Maintenant que l'on a téléchargé l'archive, il faut l'extraire pour cela on fait la commande suivante :

```
$ tar xzf tbb<version>.tgz
```

3. Un fois extrait, il faut se rendre dans le dossier bin et modifier le script tbbvars.sh. Il faut changer la ligne suivante :

```
TBBROOT = SUBSTITUTE_INSTALL_DIR_HERE
```

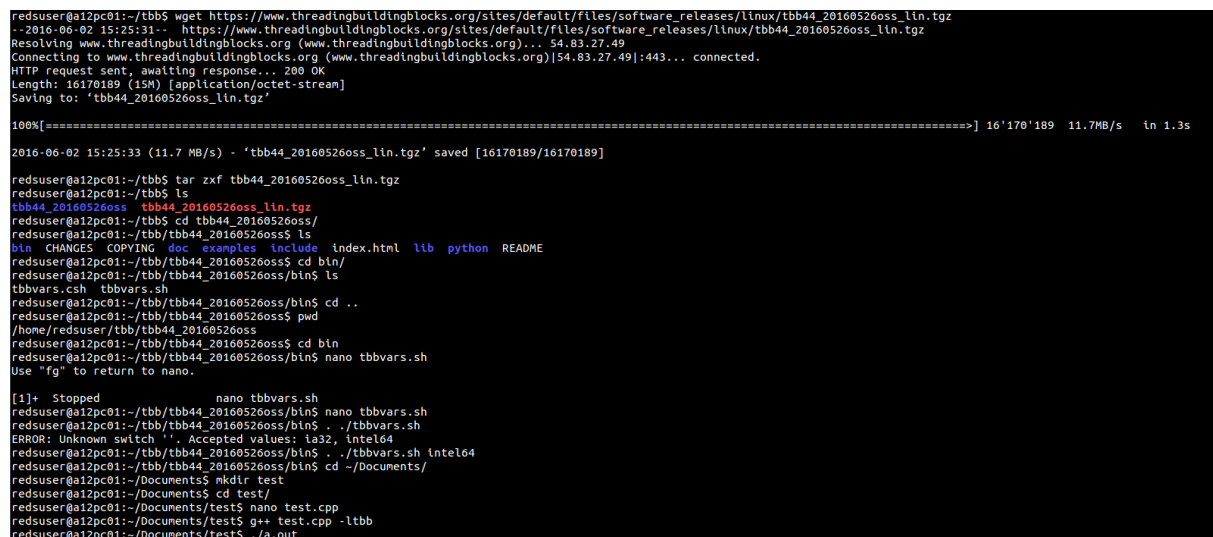
par

```
TBBROOT = <path_tbb_directory>
```

Cela permettra lors de pouvoir utiliser la librairie facilement lors de la compilation.

4. Après, on exécute le script avec la bonne option (ia32 pour une architecture 32 bits ou intel64 pour du 64 bits) et on peut commencer à faire des programmes avec TBB.

5. Pour tester si tout a été bien installé, on compile le code précédent et on l'exécute. Si il n'y a pas eu de problème durant la compilation et l'exécution c'est que tout est bon



```

reduser@ai2pc01:~/tbb$ wget https://www.threadingbuildingblocks.org/sites/default/files/software_releases/linux/tbb44_20160526oss_lin.tgz
--2016-06-02 15:25:31-- https://www.threadingbuildingblocks.org/sites/default/files/software_releases/linux/tbb44_20160526oss_lin.tgz
Resolving www.threadingbuildingblocks.org (www.threadingbuildingblocks.org)... 54.83.27.49
Connecting to www.threadingbuildingblocks.org (www.threadingbuildingblocks.org)|54.83.27.49|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16170189 (15M) [application/octet-stream]
Saving to: 'tbb44_20160526oss_lin.tgz'

100%[=====] 16'170'189 11.7MB/s in 1.3s

2016-06-02 15:25:33 (11.7 MB/s) - 'tbb44_20160526oss_lin.tgz' saved [16170189/16170189]

reduser@ai2pc01:~/tbb$ tar xzf tbb44_20160526oss_lin.tgz
reduser@ai2pc01:~/tbb$ ls
tbb44_20160526oss  tbb44_20160526oss_lin.tgz
reduser@ai2pc01:~/tbb$ cd tbb44_20160526oss/
reduser@ai2pc01:~/tbb/tbb44_20160526oss$ ls
bin  CHANGES  COPYING  doc  examples  include  index.html  lib  python  README
reduser@ai2pc01:~/tbb/tbb44_20160526oss$ cd bin/
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ ls
tbbvars.csh  tbbvars.sh
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ cd ..
reduser@ai2pc01:~/tbb/tbb44_20160526oss$ pwd
/home/reduser/tbb/tbb44_20160526oss
reduser@ai2pc01:~/tbb/tbb44_20160526oss$ cd bin
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ nano tbbvars.sh
Use "fg" to return to nano.

[1]+  Stopped                  nano tbbvars.sh
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ nano tbbvars.sh
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ . ./tbbvars.sh
ERROR: Unknown switch "-". Accepted values: ia32, intel64
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ . ./tbbvars.sh intel64
reduser@ai2pc01:~/tbb/tbb44_20160526oss/bin$ cd ~/Documents/
reduser@ai2pc01:~/Documents$ mkdir test
reduser@ai2pc01:~/Documents$ cd test/
reduser@ai2pc01:~/Documents/test$ nano test.cpp
reduser@ai2pc01:~/Documents/test$ g++ test.cpp -ltbb
reduser@ai2pc01:~/Documents/test$ ./a.out

```

FIGURE 1 – Capture d'écran des commandes utilisées

1. <https://www.threadingbuildingblocks.org/download>

3 Exemple

Le programme utilisé pour l'exemple s'occupe de convertir une image couleur en une avec des nuances de gris. Nous avons choisi le code suivant car il se prête bien à la parallélisation et que nous avons déjà fait des optimisation avec d'autre outils. Cela nous permet ainsi de faire des meilleurs critiques au niveau des performances.

3.1 Code

3.2 Mesures

4 Analyse

5 Bibliographie

- Tutoriel d'Intel : <https://www.threadingbuildingblocks.org/intel-tbb-tutorial>
- Documentation d'Intel : <https://software.intel.com/en-us/tbb-documentation>