Laboratory Notebook for a Multi-Threaded Version of Quicksort

Maxime Chevalier

January 20, 2017

Contents

1 Indroduction

Ce journal est le compte rendu d'un élève de RICM4 à Grenoble.

2 Caractéristiques de la machine

2.1 Version du système d'exploitation

```
uname -a;
```

```
Linux chevamax-Satellite-Pro-R50-B 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37
```

2.2 CPU

```
lscpu;
```

Architecture:	$x86_{64}$				
Mode(s)	opératoire(s)	des	processeurs :32-bit,	64-bit	
Byte	Order:	Little	Endian		
CPU(s):	4				
On-line	CPU(s)	list:	0-3		
Thread(s)	par	cœur :	2		
Cœur(s)	par	socket :	2		
Socket(s):	1				
Nœud(s)	NUMA :	1			
Identifiant	constructeur :GenuineIntel				
Famille	de	processeur :6			
Modèle :	69				
Model	name:	Intel(R)	Core(TM)	i5-4210U	C
Révision :	1				
Vitesse	du	processeur	en	MHz :997.875	
CPU	max	MHz:	2700,0000		
CPU	min	MHz:	800,0000		
BogoMIPS:	4788.54				
Virtualisation :	VT-x				
Cache	L1d :	32K			
Cache	L1i :	32K			
Cache	L2 :	256K			
Cache	L3 :	3072K			
NUMA	node0	CPU(s):	0-3		
Flags:	fpu	vme	de	pse	t

2.3 RAM

```
free -m;
```

total	utilisé	libre	partagé	tamp/cache	disponible	
Mem:	7898	1702	1133	366	5062	5751
Partition	d'échange:	7996	0	7996		

3 Test 1

Réexecution du code sur ma machine

```
"exports both"body
```

3.1 Analyse des résultats

On voit ici que les résultats sont complétement différents (et même plutot incohérents...). A se demander si la machine utilise ses différents coeurs!

3.2 Retestons en branchant !

```
"exports both"body
```

3.3 Analyse

C'est pas mieux ...

3.4 Vérifions que le programme utilise plusieurs coeurs

```
perf stat -e cpu-cycles -e instructions -e cache-references -e cache-misses -e LLC-loa
```

Sequential quicksort took: 0.281554 sec. Parallel quicksort took: 0.448588 sec. Built-in quicksort took: 0.272621 sec.

Performance counter stats for 'CPU(s) 0-3':

S0-C0 2 3 118 920 870 cpu-cycles (66,70%) S0-C0 2 2 941 573 615 instructions (83,41%) S0-C0 2 8 480 420 cache-references (83,40%) S0-C0 2 2 811 502 cache-misses (83,40%) S0-C0 2 4 283 818 LLC-loads (83,40%) S0-C0 2 1 822 019 LLC-stores (83,11%) S0-C1 2 1 707 679 255 cpu-cycles (66,70%) S0-C1 2 952 446 235 instructions (83,41%) S0-C1 2 8 421 277 cache-references (83,40%) S0-C1 2 3 175 977 cache-misses (83,40%) S0-C1 2 3 255 670 LLC-loads (83,40%) S0-C1 2 1 912 330 LLC-stores (83,11%)

1,036287854 seconds time elapsed

Ca a l'air de tourner sur deux des 4 cpu.

4 Test 2

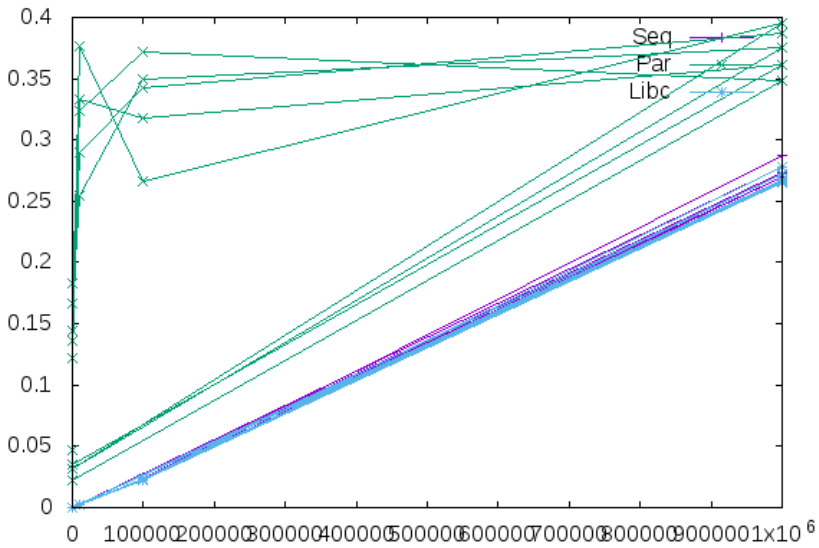
Nous allons garder les mêmes valeurs en mélangeant l'ordre d'execution

```
OUTPUT_DIRECTORY=data/‘hostname‘_‘date +%F‘
mkdir -p $OUTPUT_DIRECTORY
OUTPUT_FILE=$OUTPUT_DIRECTORY/measurements_‘date +%R‘.txt

touch $OUTPUT_FILE

for rep in ‘seq 1 5‘; do
for i in 100 1000 10000 100000 1000000; do
echo "Size: $i" >> $OUTPUT_FILE;
./src/parallelQuicksort $i >> $OUTPUT_FILE;
done
done

FILENAME="data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_14:57"
perl scripts/csv_quicksort_extractor2.pl < "$FILENAME.txt" > "${FILENAME}_wide.csv"
echo "
  set terminal png size 600,400
  set output '${FILENAME}_wide.png'
  set datafile separator ','
  set key autotitle columnhead
  plot '${FILENAME}_wide.csv' using 1:2 with linespoints, '' using 1:3 with linespoints
" | gnuplot
echo [[file:${FILENAME}_wide.png]]
```



Oups...

```
library("ggplot2")
df <- read.csv("data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_14:57_wide.c
ggplot()+ geom_point(data=df, aes(x=Size, y=Seq, color="green"))+
  geom_point(data=df, aes(x=Size, y=Par, color="red")) +
  geom_point(data=df, aes(x=Size, y=Libc));
```

Petit test en R, mais je ne maitrise pas du tout ggplot ... Du coup difficile d'analyser.

5 Test 3

Dans ce troisieme test nous allons affiner la courbe en faisant des tests sur des tableaux de taille 10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000 et 1000000.

Nous allons effectuer le test 10 fois pour chaque valeur.

```
scripts/run_benchmarking.sh

OUTPUT_DIRECTORY=data/‘hostname‘_‘date +%F‘
mkdir -p $OUTPUT_DIRECTORY
OUTPUT_FILE=$OUTPUT_DIRECTORY/measurements_‘date +%R‘.txt

touch $OUTPUT_FILE
for i in 10 50 100 500 1000 5000 10000 50000 100000 500000 1000000; do
    for rep in ‘seq 1 10‘; do
        echo "Siz
```

Pour trasformer le fichier texte en fichier on utilise le script fourni et la commande :

```
FILENAME="data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_16:12"
perl scripts/csv_quicksort_extractor2.pl < "$FILENAME.txt" > "${FILENAME}_wide.csv"

library("ggplot2")
df <- read.csv("data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_16:12_wide.c
ggplot()+ geom_point(data=df, aes(x=Size, y=Seq, color="green"))+
  geom_point(data=df, aes(x=Size, y=Par, color="red")) +
  geom_point(data=df, aes(x=Size, y=Libc));
```

5.1 Compilons en -O3

```
FILENAME="data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_16:24"
perl scripts/csv_quicksort_extractor2.pl < "$FILENAME.txt" > "${FILENAME}_wide.csv"
```

```
library("ggplot2")
df <- read.csv("data/chevamax-Satellite-Pro-R50-B_2017-01-19/measurements_16:12_wide.c
ggplot()+ geom_point(data=df, aes(x=Size, y=Seq, color="green"))+
  geom_point(data=df, aes(x=Size, y=Par, color="red")) +
  geom_point(data=df, aes(x=Size, y=Libc));
```