SPM final project

Davide Neri

February 19, 2016

Master Degree in Computer Science and Networking A-Y 2015-16

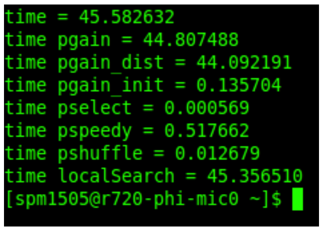
Student:	Davide Neri
Instructor:	Marco Danelutto
Course:	Distributed Systems: Paradigms and models

Abstract

The report describes the final project addressing the **StreamCluster (SC)** application. The sequential code is provided by *C* code (PARSEC and RODINIA Benchmark Suite). The parallel version is implemented using **FastFlow** framework (*C++*).

1 Application Analysis

Given a stream of points the appicatlion find the a predetermined number od medians. The points of the stream are organized in chuncks. For every chunk received, the application computes the intermediate centers in the chunk. When the stream of chunks is finished, the algorithm finds the center points among all the intermediate centers.
The figure 1 shows the sequential algorithm profiling with a `simlarge` imput test. Can be observe that the sequential algorithm spend most of the time evaluating the gain of opening a new center (`pgain` function).
The problem becomes computationally intesitive as the dimensinality of the points increases. The goal is to parallelize both the processing of different chunks in the stream (that are independent) and the processing of the single chunk.



```
time = 45.582632
time pgain = 44.807488
time pgain_dist = 44.092191
time pgain_init = 0.135704
time pselect = 0.000569
time pspeedy = 0.517662
time pshuffle = 0.012679
time localSearch = 45.356510
[spm1505@r720-phi-mic0 ~]$
```

Figure 1: Profiling with simlarge input (completiom times(s)).

2 Design choice

A *task-decomposition* pattern is used to model parallel execution of chunks points in the stream. A single chunk computation is parallelized using a *data-decompositon* pattern.
The application doesn't require any particular order of the chunks, because finding the intermediate centers points in different chunks is totally indipendent of the order in wich the chunks arrives. Howhewer the results can be different due to the fact that the algorithm use a `shuffle` procedure that performs a permutation of the points and can changes the final centers. Using a seed is not sufficient to guarantee equal permutations because each worker is indipendent.
The dependency analysis: different order of the chunks can have different centers computation. In this particular application is not required to have ordered chunks because in any case the results is a set of centers of the points received that can appear in any order in the stream.
The design evaluation :

- The target platform is a shared memory machine (16 cores on host machine or 240 cores on Xeon phi machine).
- Is possible to specify different parameters in order to have different parallel version. Is possible to have only the "task-decompositon" or only the "data decomposition" or a combination.

The parameters can be tuned for having different possibility: parallelism degree in the task-farm pattern and map parallel.
The task-parallelism is used to divide the stream of points in chunks and send the chunks to the workers. Each worker is in charge to compute a single chunk of points and find the centers. Data parallel pattern uses loop parallelism.

3 Compile,run and tests the project

the short user manual should be detailed enough to enable me to run tests with your software (what should I compile, how, which are the parameters to use to launch experiments, how can I vary parallelism degree, which input files do I need, where are they)
The structure of the folders are:

- `bin/` : contains the executables (automatic creation during the compilation).
- `run/` : contains the output results (automatic creation during the execution).
- `conf/` : contains the parameters of the application (binary path and input parameters).
- `src/` : contains the source code.
- `streamclusterMgmt.sh` runs different applications with different input on local machine or host.
- `upload.sh` uses *rsync* to synchronize the local project on the Phi machine.
- `run_*_on_mic.pl` runs the executbales multiple times in the mic0, returning the average completion times (map : run the map parallel version, farm: runs the farm version).

Compilation For compiling the project is required *cmake 3.x.* The steps are (starting from the root folder of the project):

a. `mkdir build && cd build` creates the build directory and goes inside.

b. `CXX=icc cmake ..` compile the executables. Two executables are olso directly copied in the mic0 machine.

Run

```
streamclusterMgmt.sh
```

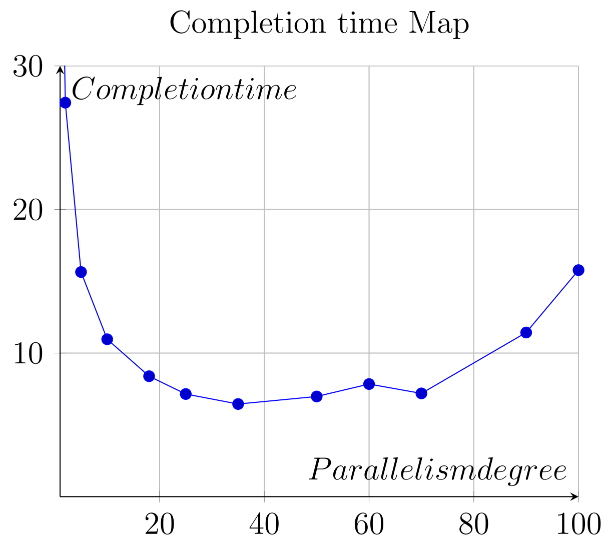
4 Experimantal results

The executable are runned on **mic0** machine with 60 cores and 4 contexts. I have runned three differents inputs for each of the two versions (only data-parallal or task-parallelism).

- `simmedium` is a collection of 8192 points with 64 dimensions each.
- *simlarge* is a collection of 16384 points with 128 dimensions each.
- *native* is a collection of 1000000 points with 128 dimensions each.

4.1 Data-parallel

scalability



References

[1] Mark Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne Marie Kermar- rec, Maarten van Steen, *Gossip-based peer sampling,* ACM Transaction on Computer Systems, October 2007.