UNIVERSIDAD CARLOS III DE MADRID

DESIGN AND OPTIMIZATION OF FUCTIONS FOR MULTICORE EXECUTION IN R

Bachelor Thesis Summary

Bachelor in Computer Science and Engineering

15/06/2012

Carlos Villalba Coronado

Tutors:

Daniel Higuero Alonso-Mardones

and

Juan Manuel Tirado Martín

Index

1 Introduction

This Bachelor Thesis addresses the optimization of heavy computational functions of a mathematical package and its evaluation. Specifically, it considers the optimization of a graph library for multicore architecture. The library uses the Graph Theory to solve and analyze problems.

A graph is a group of nodes or vertices connected with other nodes or vertices by edges. The edges also can have a numerical value and direction for some problems. The Graph Theory is very useful for a lot of purposes: network optimization, social network analysis, state machines design and much more. As years go by, the Graph Theory is more important and used in different fields.

The heavy function selected to be optimized in this work determinate if a graph is a Small World. A Small World is a property of some graphs. A graph is considered a Small World if it has a short diameter and very high clustering coefficient.

The diameter is the maximum eccentricity of any two vertices of a graph. The eccentricity is the shortest path between two vertices of a graph.

The clustering coefficient is a property of a graph that determinates if a graph is highly connected.

Calculation of the diameter and clustering coefficient are heavy tasks if the graphs are very big and needs a lot of computational time.

2 Related work

In the previous study of the related work on the subject, different options of mathematical software, multicore optimization and graphs libraries have been studied.

2.1 Mathematical software

There are a lot of options of mathematical software which can import external libraries for others purposes like graph computing. This study evaluates four options that are very extended in academic and professional work:

Software	Operating Systems	Development languages	Commercial licensed price	Library availability
Matlab	Multi-platform	C/C++, Fortran and Java	2000€	Medium
Octave	Multi-platform	C/C++ and Fortran	Free	Medium
R	Multi-platform	C/C++ and Fortran	Free	High
S-Plus	Windows and Linux	C/C++, Fortran and Java	1000$ / year	Low

Comparison of mathematical software.

The final choice has been R because MatLab and S-Plus are more expensive. And R is the most accessible for developers and users. In R there are a lot of open source libraries that can be modified and improved easily.

2.2 Multiprocessing

The optimization is based on computer parallelism for multicore architectures. For that task, there are several options:

Option	Portabilility	Programming Languages	Development dificult
Intel TBB	High	C/C++	Low-Medium
OpenMP	High	C/C++ and Fortran	Low-Medium
Multithreading	Medium	All	High

Comparison of multicore optimization options.

The choice has been OpenMP because is the easier and more portable than Multithreading. And it is more appropriate for loop optimization than Intel Threading Building Blocks (TBB).

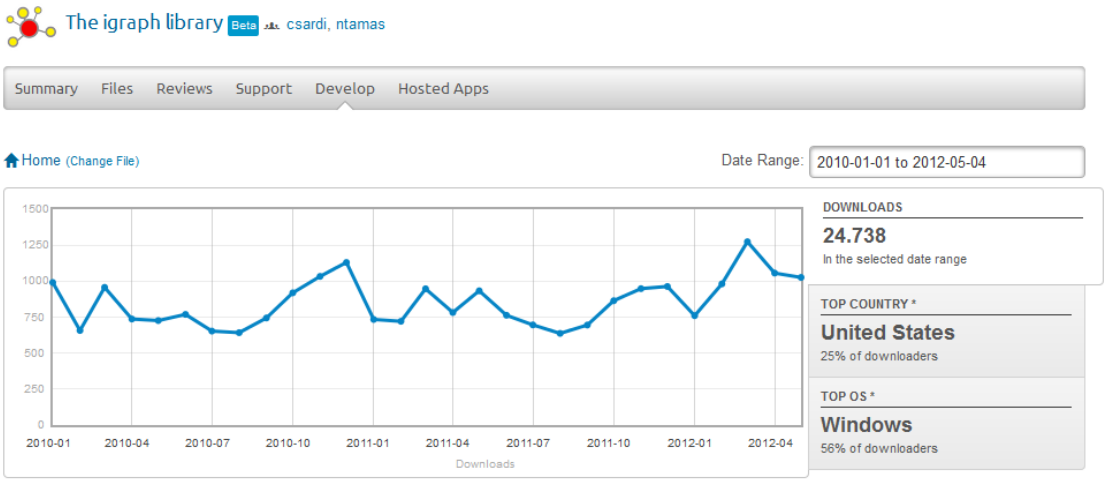
2.3 Graph library

The igraph library is the library has been the choice for many reasons. Igraph is an open source code for studying and working with graphs. It has a lot of functions to create, modify, import, export and calculate properties of graphs.

Igraph compute the functions in a sequential way. To improve the performance, igraph can be modified to do it in a parallel way. Igraph is perfect for this purpose because is an open source code. Also, igraph is written in C/C++. OpenMP makes easier the optimization because it supports C/C++ language.

Igraph has some interfaces. It has an R interface, but it has Python and Ruby interfaces too. The open source code in C/C++ makes easier the use without interface or to create your own interface.

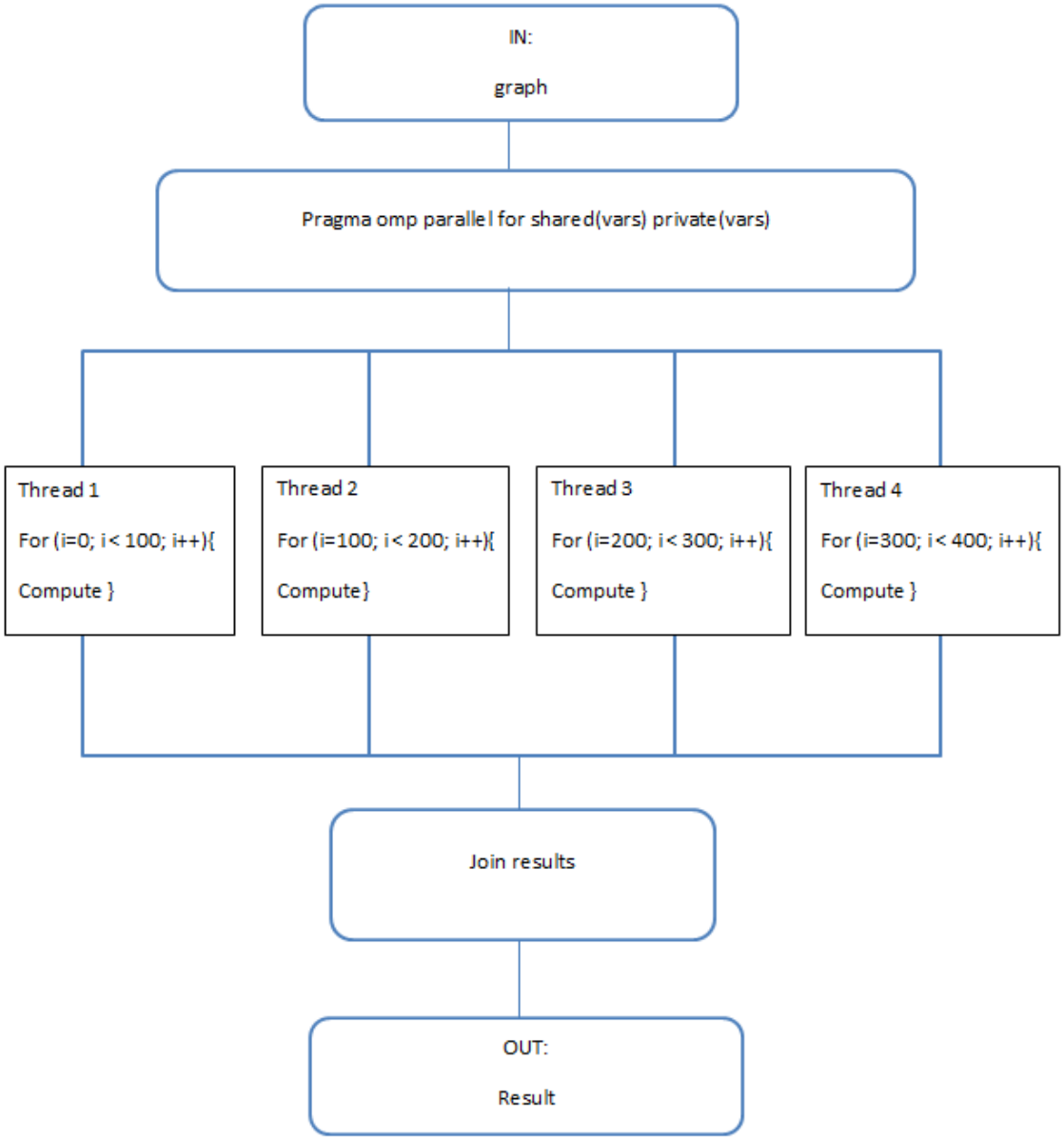
The last reason to choose igraph is because it is one of the most widely used graph library with an R interface. The following graph show the downloads in the last two years:



Graph X: Igraph downloads between 2010 and May of 2012.

3 Implementation

The optimization of the igraph functions are based in distributing the computational load of the function between the different cores of the computer. Most specifically, OpenMP uses a compilation directive or pragma to distribute the loop iterations between the processors creating different threads as the diagram shows:

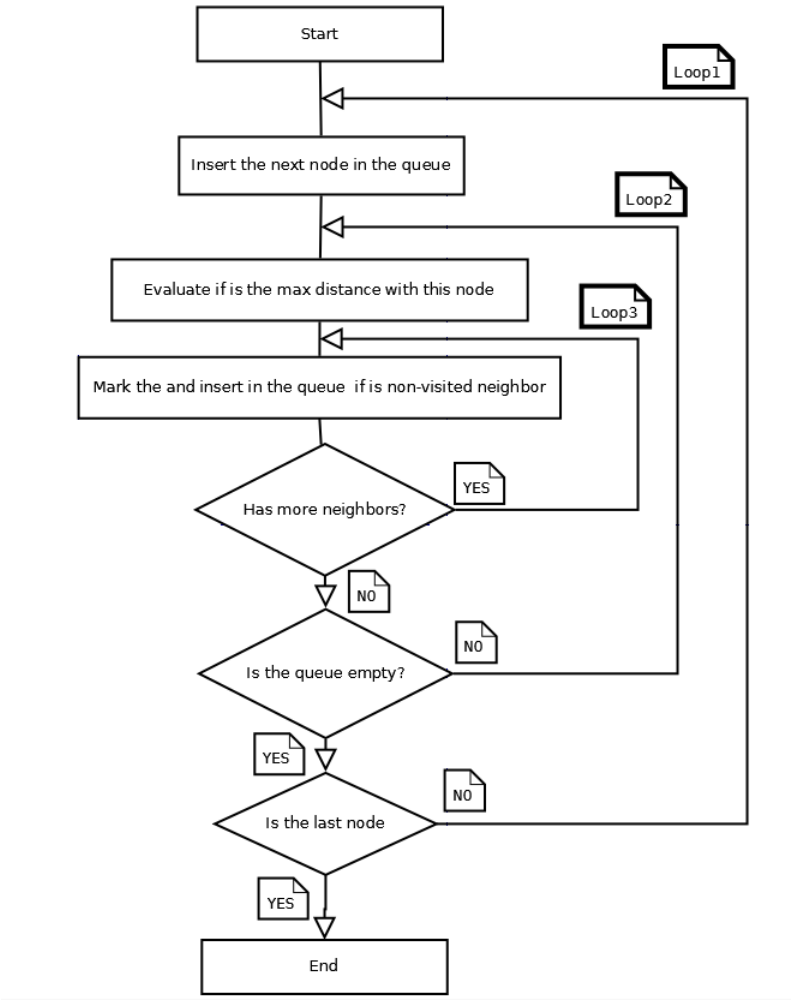


This example shows how OpenMP distributes a for loop of 400 iterations between four cores.

It is important to take care of the dependencies of the variables and use properly all the OpenMP resources: the private and the shared variables, the synchronization systems and the mutual exclusion access systems.

3.1 Diameter function

The diameter function uses the following flow diagram for the execution:



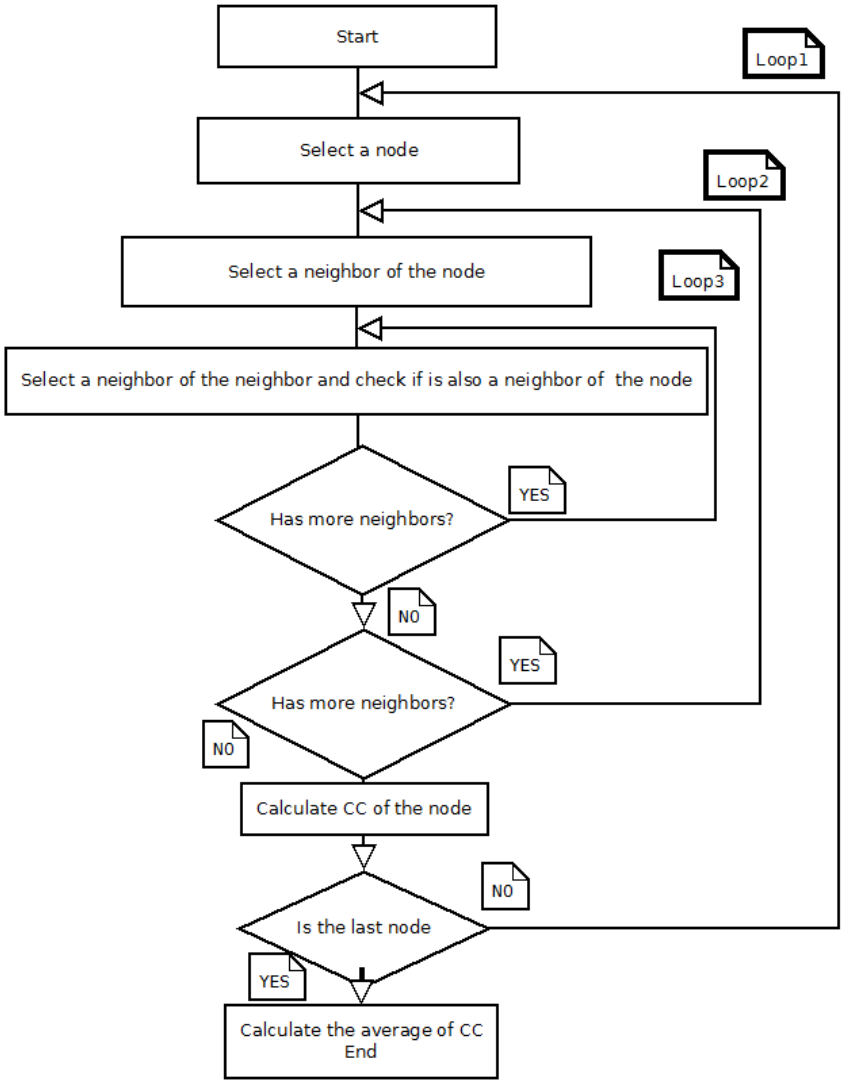
Flow diagram of diameter function.

It shows that the computational time of the function depends on the number of nodes and the number of graph edges. The complexity is O(Nodes*Edges).

The best way to use parallel computing to optimize this function is to distribute the iterations of the loop1 between the cores. This is done with an OpenMP pragma to optimize a loop distributing the iterations between the cores.

3.2 Clustering coefficient function

The clustering coefficient function uses the following flow diagram for the execution:



Flow diagram of clustering coefficient function.

It shows that the computational time of the function depends on the number of the nodes and the average number of neighbors that has the node. The complexity is O(Nodes*Average Neighbors^2).

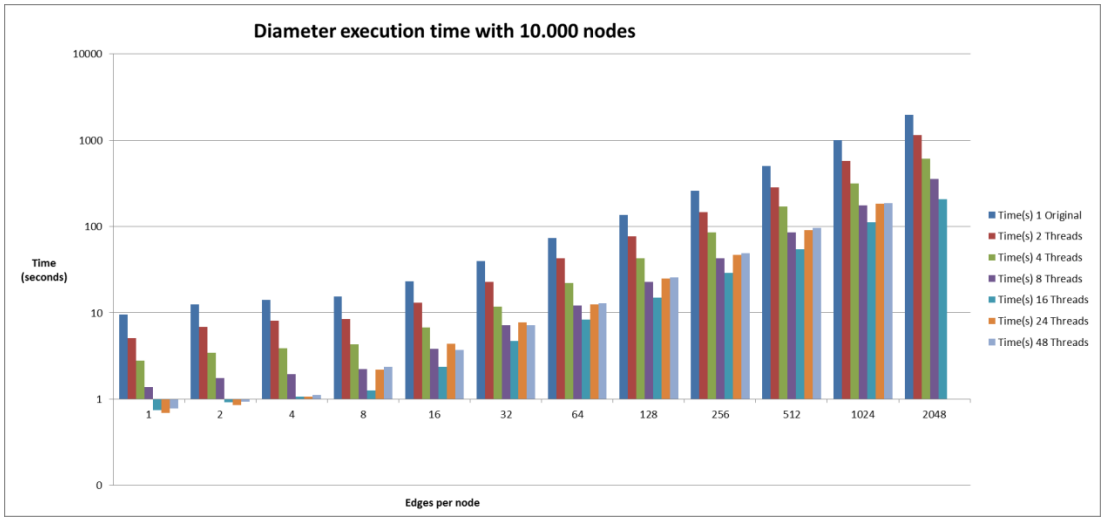
The best way to use parallel computing for optimize this function is distribute the iterations of the loop1 between the cores. This is doing with an OpenMP pragma to optimize a loop distributing the iterations between the cores.

4 Evaluation

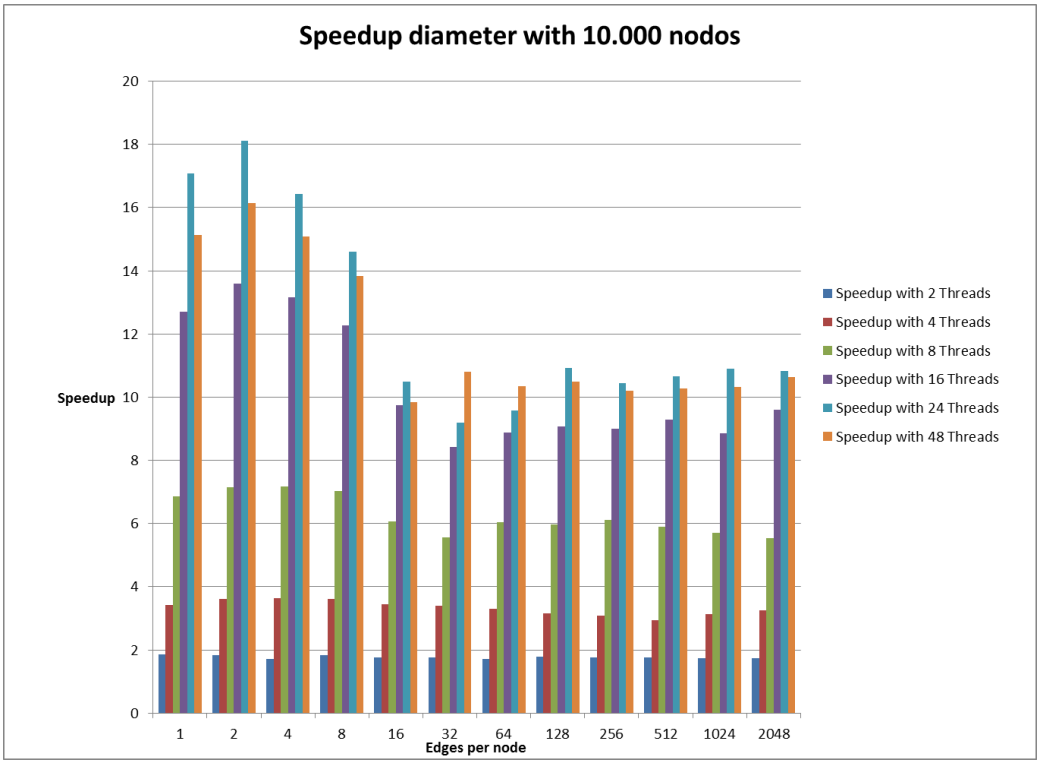
The evaluation has been done in an AMD Opteron(tm) Processor 6168 with 60GB of memory and 24 cores.

4.1 Diameter

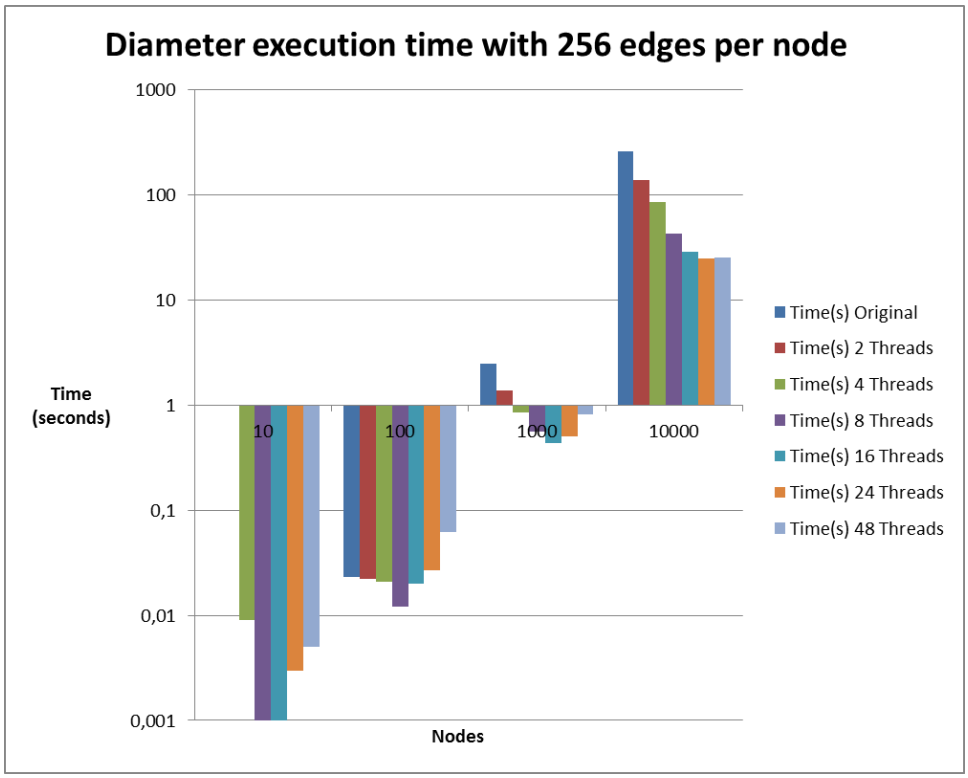
The diameter function depends on the number of nodes and the number of edges.



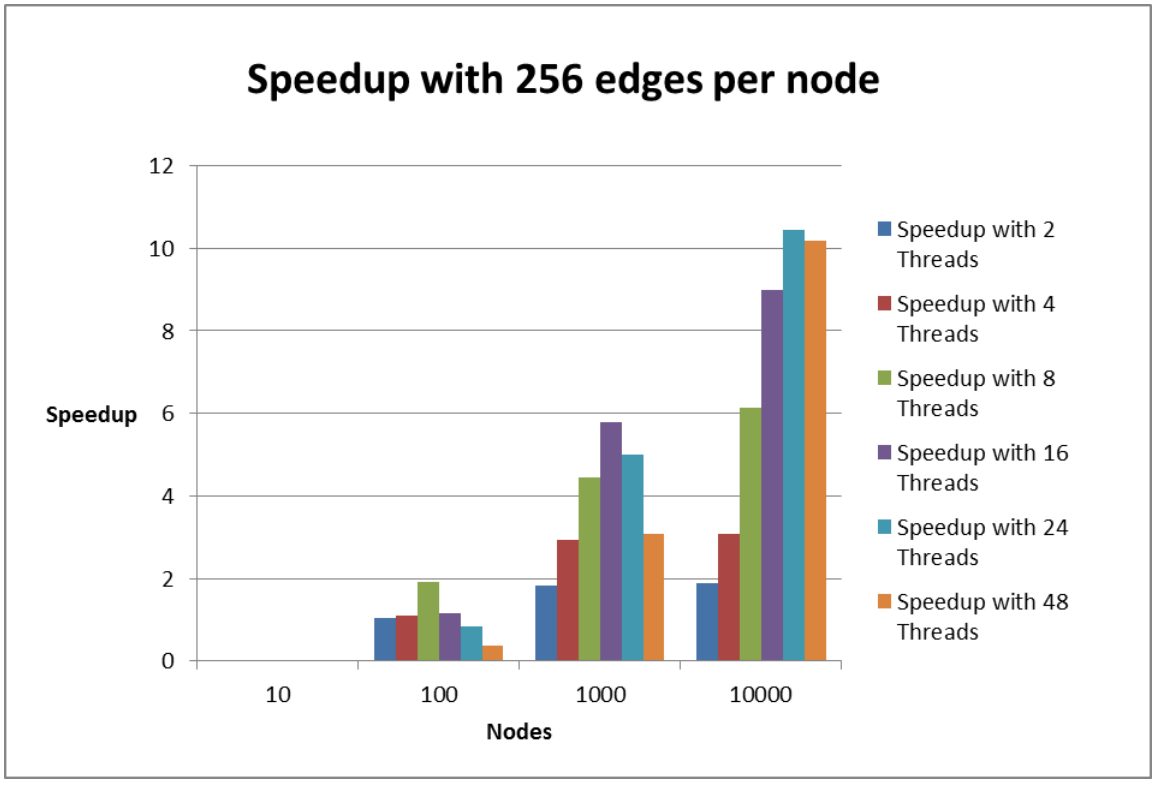
Execution time depends on the number of edges.



With 10.000 nodes, the speedup is better with less edges per node.



Execution time depends on the number of nodes.

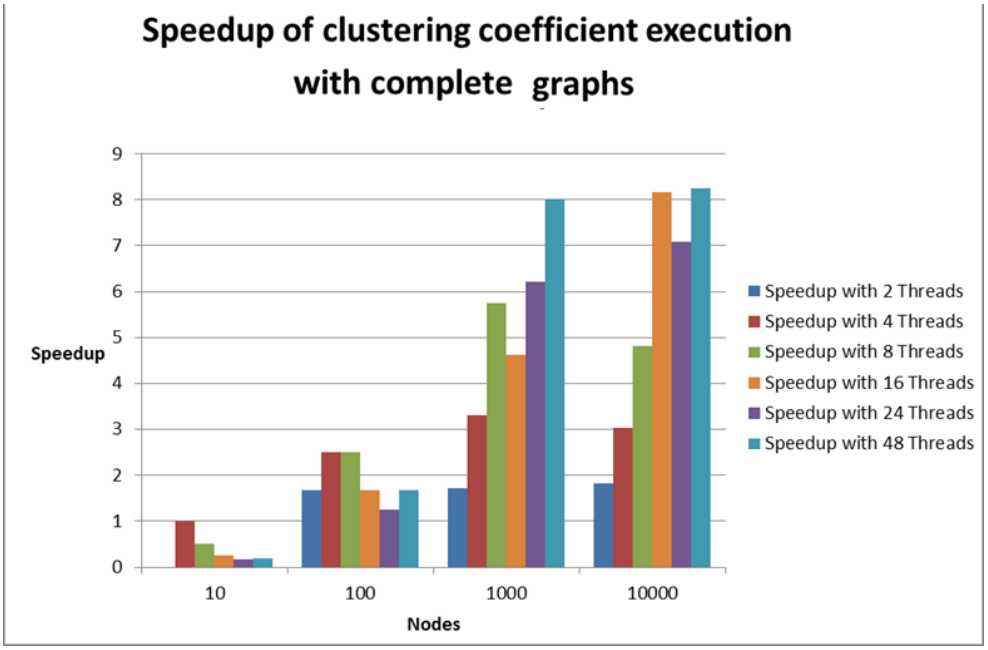


With 256 edges per node, the speedup is better with more nodes.

4.2 Clustering Coefficient

The clustering coefficient execution time depends much more on the quantity of neighbors than the quantity of nodes. The clustering coefficient computation has heavy load in complete graphs.

The time execution is very high with 10.000 nodes.



The speedup is good when a task is heavy.

5 Conclusions

The most important conclusions of the work are:

- Multicore machines can compute the same work several times more quickly than a single process. But it is important to take care of the parallel design to maintain the efficiency and the performance of the function.

- The current trend in multicore machines suggests that optimizations like the one done will become more important in the short term. Igraph is a good example of a program that is easy to improve with greats results.

- The speedup is not a linear function in multicore machines. This is because more cores also mean more synchronization and more delay in each core.

- OpenMP is a very useful API, it is very simple and flexible. It is perfect to parallel loops in architectures with multicore and shared memory.

- Igraph is a good, easy and complete library for working with graphs. But it cannot compute very large graphs with millions vertices and edges and it is not prepared to use all the potential of the multicore machines.

- Graph Theory is very useful and it has a lot of uses. Actually, it is essential in numerous different areas.