Foundations of High Performance Computing

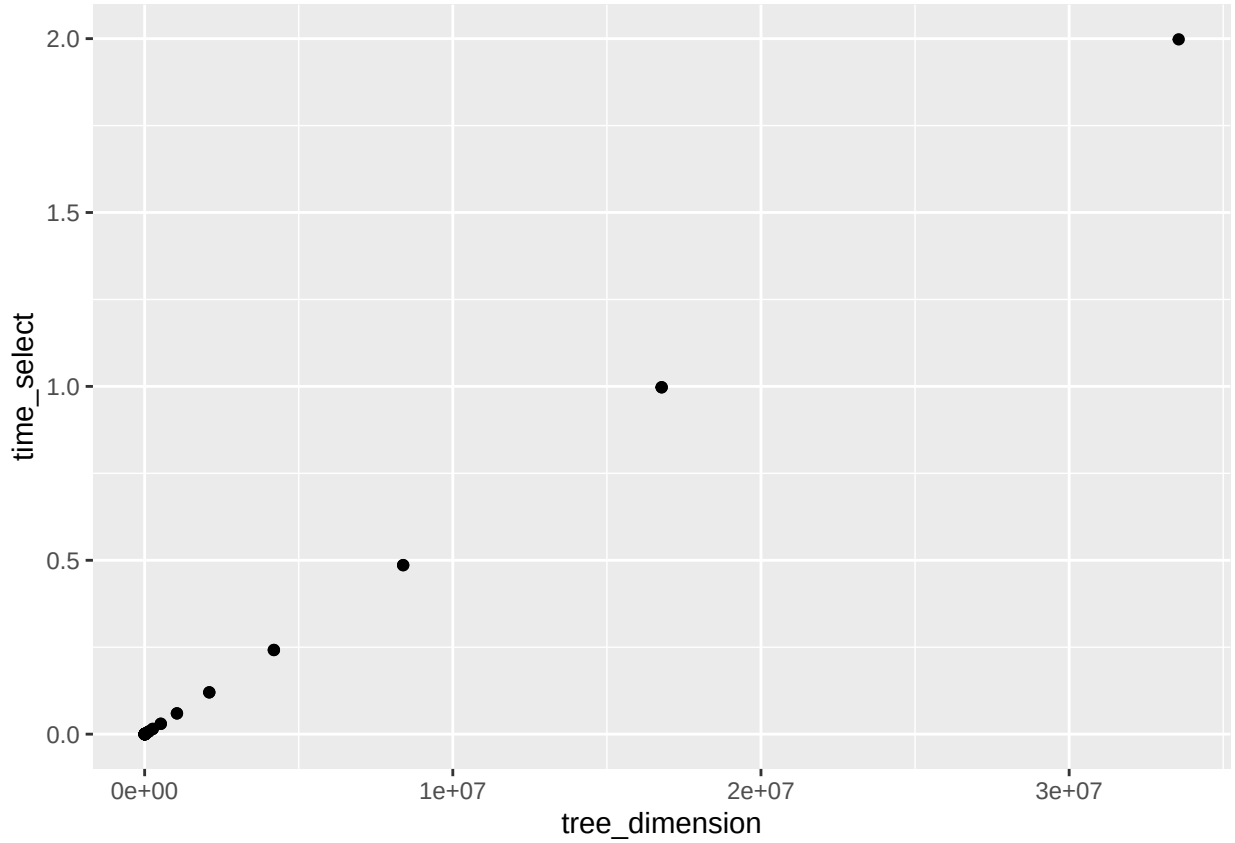
Assignment II

Lorenzo Trubian [mat. SM3500519]

27/02/2022

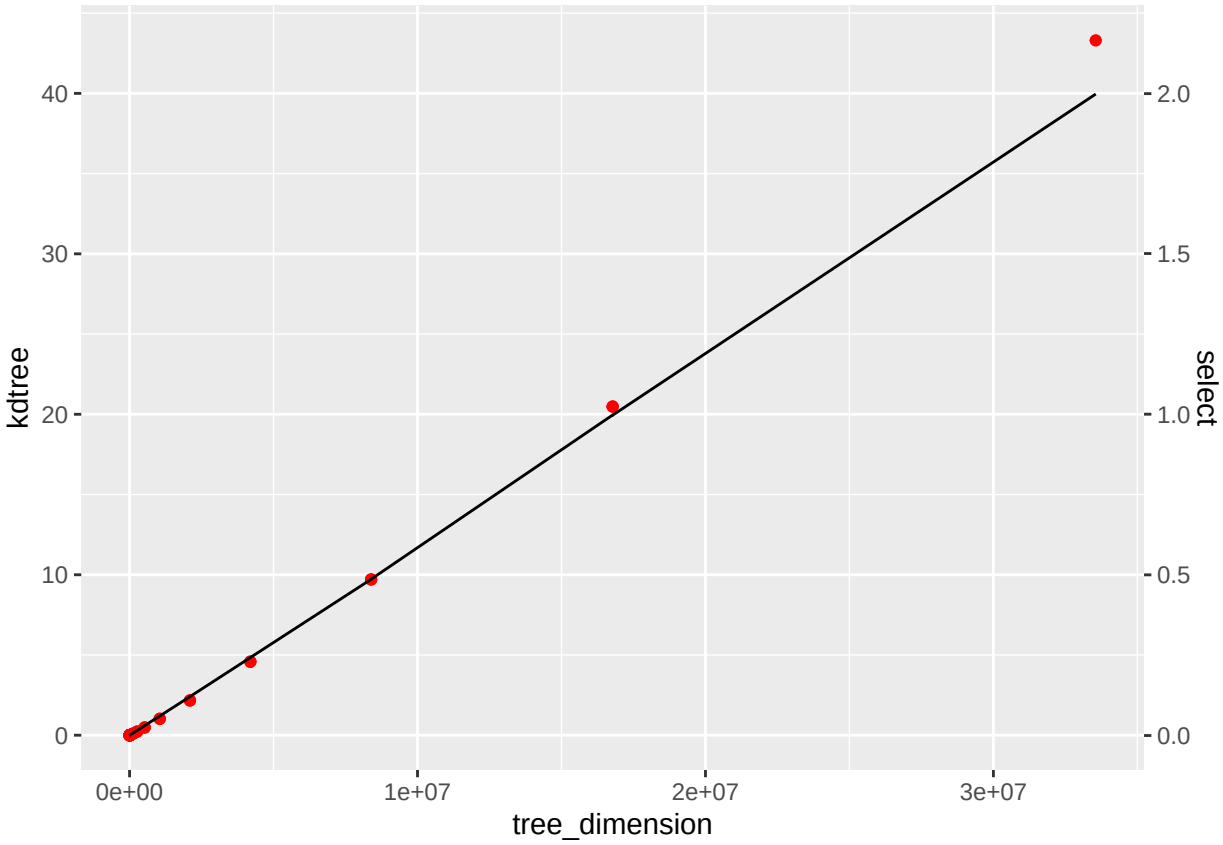
```
setwd("~/Pubblici/learn-git/assignment2/data")
library(ggplot2)
tab.2 <- read.csv("parallel-mpi/time_mpi-2-0.csv")
tab.2.prec <- read.csv("parallel-mpi/time_mpi-2-1.csv")
tab.serial <- read.csv("serial/time-serial_2.csv")
# tab.serial.prec
tab.serial$axis <- as.factor(tab.serial$axis)
tab.2$numproc <- as.factor(tab.2$numproc)


ggplot( data = tab.serial[,] )+
  geom_point( mapping = aes(x = tree_dimension, y = time_select)) #+
```



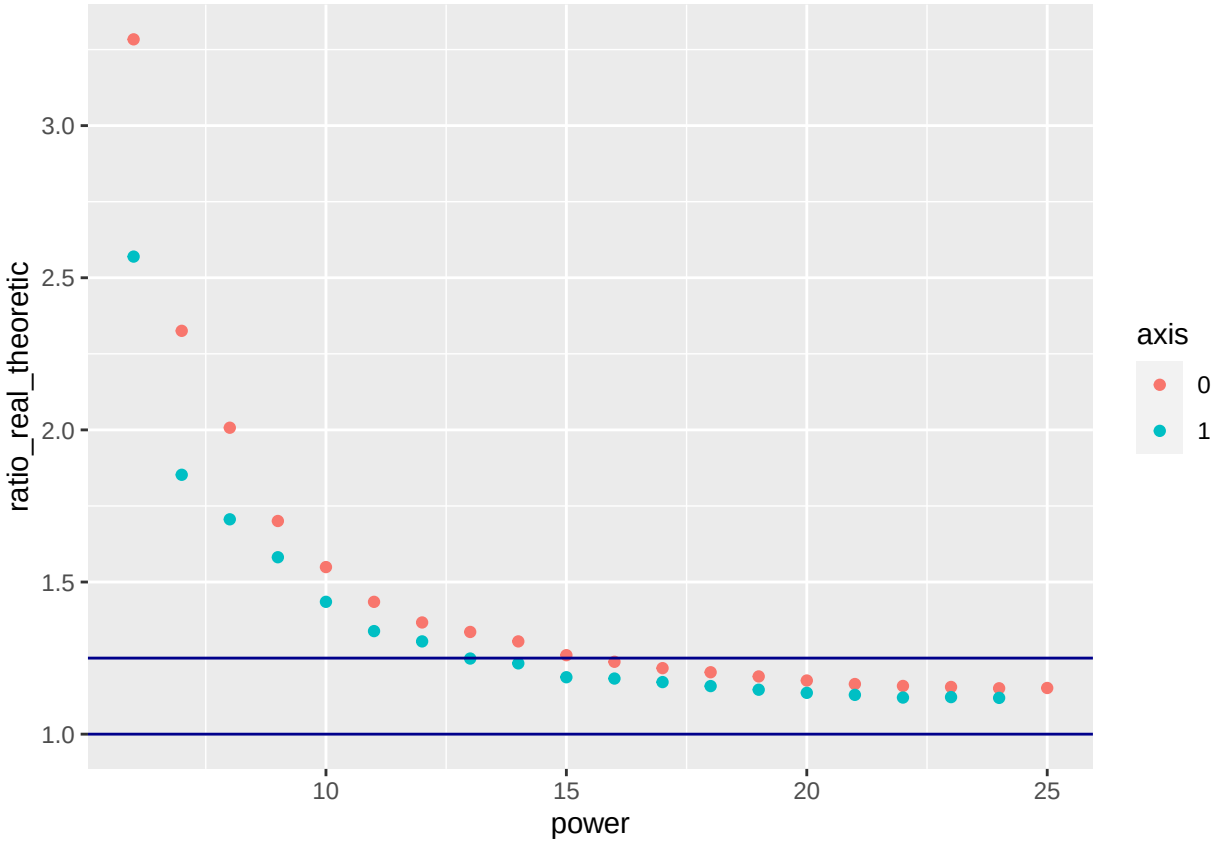
```
  # scale_x_continuous(trans='log2')

ggplot( data = tab.serial[,] )+
  geom_point( mapping = aes(x = tree_dimension, y = time_kdtree), color = "red")+
  scale_y_continuous(name = "kdtree",
                     sec.axis = sec_axis( trans=~./20, name="select"))+
  geom_line( mapping = aes(x = tree_dimension, y = time_select*20))#+
```

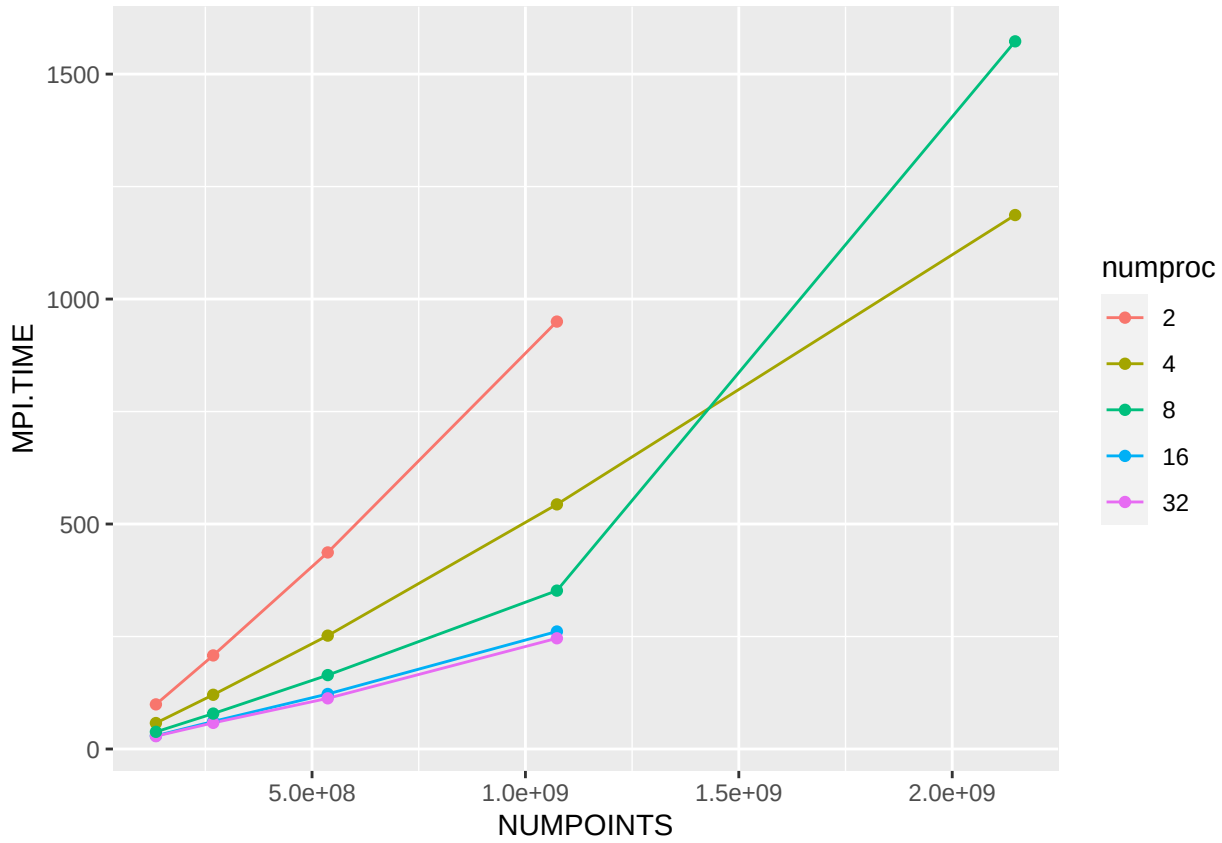



```
  # scale_x_continuous(trans='log2')

ggplot( data = tab.serial[tab.serial$power>5,] )+
  geom_point( mapping = aes(x = power, y = ratio_real_theoretic, color = axis)) +
  geom_hline(yintercept = 1, color = "darkblue") +
  geom_hline(yintercept = 1.25 , color = "darkblue")
```

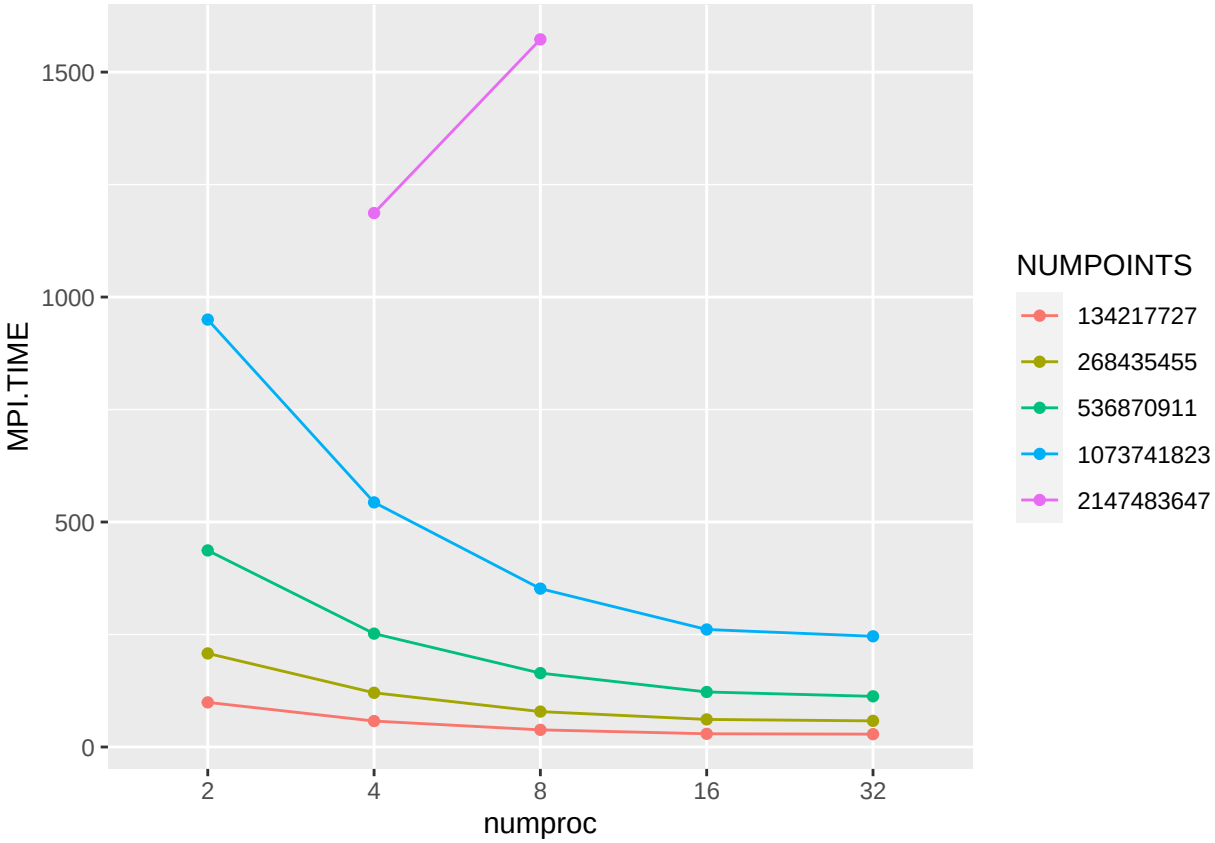



```
ggplot( data = tab.2[,] ,  mapping = aes(x = NUMPOINTS, y = MPI.TIME, color = numproc, group = numproc))
  geom_line() +
  geom_point() #+
```




```
  # scale_x_continuous(trans='log2')
# 134217727 268435455 536870911 1073741823 2147483647 4294967295

tab.2$NUMPOINTS <- as.factor(tab.2$NUMPOINTS)
ggplot( data = tab.2[,] ,  mapping = aes(x = numproc, y = MPI.TIME, color = NUMPOINTS, group = NUMPOINTS
  geom_line() +
  geom_point()
```



Introduction

The aim of this work was to analyze the performance improvement of building-kdtree algorithm through a parallelization. In particular, we try to investigate the difference using the distributed or shared memory approach, using MPI library in the first case and OpenMP API in the second. Before continuing, we recall briefly the type of structure that we want to obtain. Given a d-dimensional set of points, a kdtree is a binary and balanced tree where each node contains a point which is the median (according one chosen dimension) of all the points in the sub-tree having the node as the root.

Algorithm

here

Implementation