Analysis of MPI point to point operations to calibrate SMPI

A. Degomme A. Legrand

October 5, 2015

If needed, you should install the right packages (plyr, ggplot2, and knitr) with the install.packages command.

```
## Loading required package:  plyr
## Loading required package:  ggplot2
## Loading required package:  methods
## Loading required package:  XML
```

```
read_csv <- function(file) {
  df <- read.csv(file, header=FALSE, strip.white=TRUE)
  names(df) <- c("ResourceId","Value","Size","Start","Duration")
  df
}

df_send <- read_csv('load_send.csv')
df_sendrecv_same <- read_csv('load_sendrecv_same.csv')
df_sendrecv_diff <- read_csv('load_sendrecv_diff.csv')
df_alltoall <- read_csv('load_alltoall.csv')
```

In this file, we have the complete set of values produced during the run. These values are loaded from 4 csv files generated during the run.
How much sender processes do we have :

```
num_senders = ((max((df_send$ResourceId)+1))/2)
print(num_senders)

## [1] 6

#number of iterations of each experiment
num_iters = 50

#threshold of the number of concurrent comms after which we get a "stable" bandwidth.
# This is for instance 10 if we have 10 senders with 1Gb links saturating a 10Gb backbone link
thresh = 10
```

First experiment : a simple send/recv, incrementing from one node sending to half of them sending to the other half, each time adding one. this tests the full capacity of the link (n steps at the end)

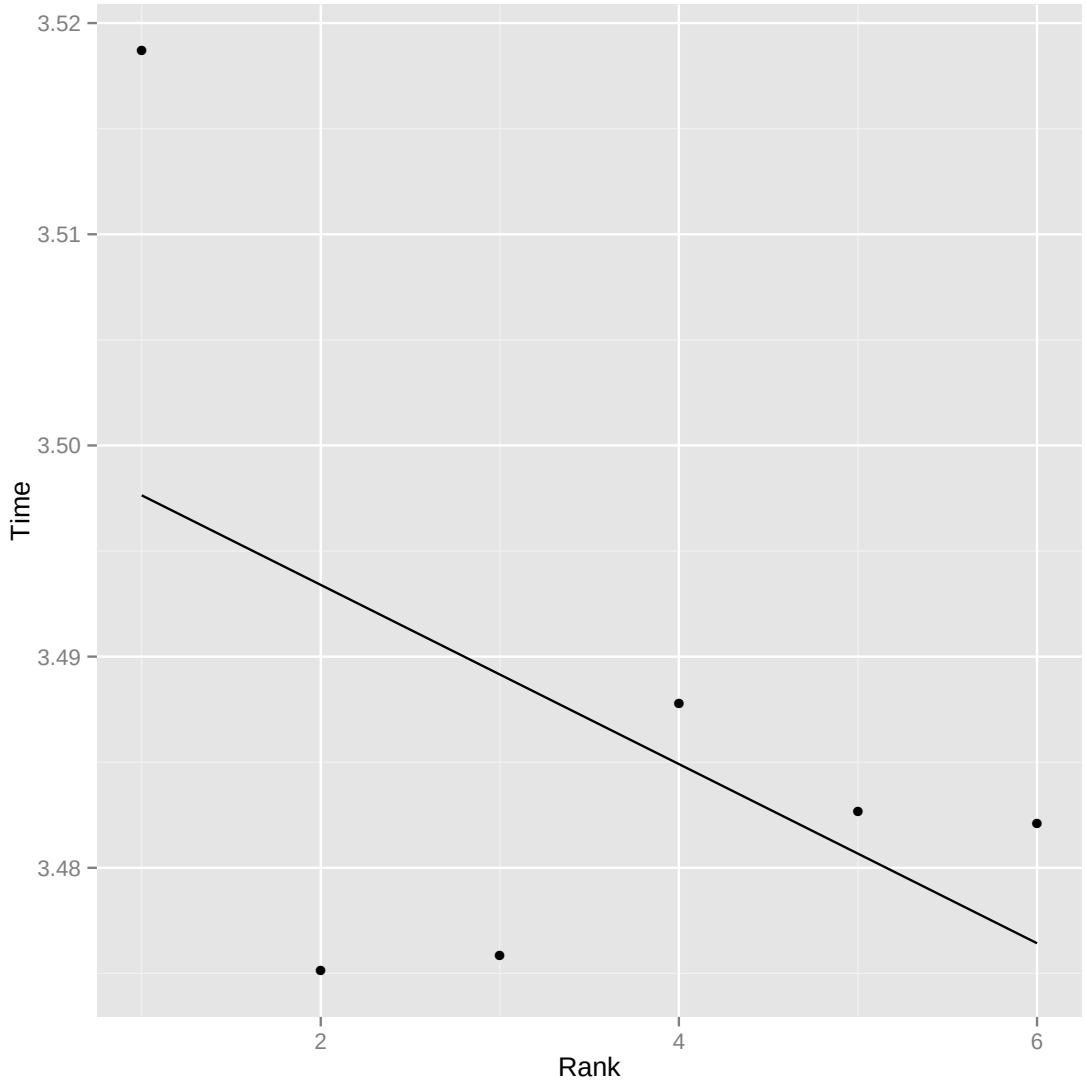
```
index<-c()
for(rank in 0:(num_senders-1)) {
    index<-c(index,rank+1)
}
```

```
times <-df_send[df_send$ResourceId==0,]$Duration
means <-apply(matrix(times,ncol=num_senders,nrow=num_iters), 2, mean)
timing<-data.frame()
timing <-cbind(means, index)
timing<-as.data.frame(timing)
names(timing)= c("Time", "Rank")

lm_small <- lm(Time ~ Rank, data=timing[timing$Rank<=thresh,])
if(num_senders>thresh){
lm_large <-lm(Time ~ Rank, data=timing[timing$Rank>thresh &timing$Rank!=34&timing$Rank!=35,])
}else
lm_large <-lm_small
```

```
send_function <- function(x) ifelse(x<=thresh, x *( coefficients(lm_small)[2] )+ coefficients(lm_sma

timing$values_test = send_function(timing$Rank)
```

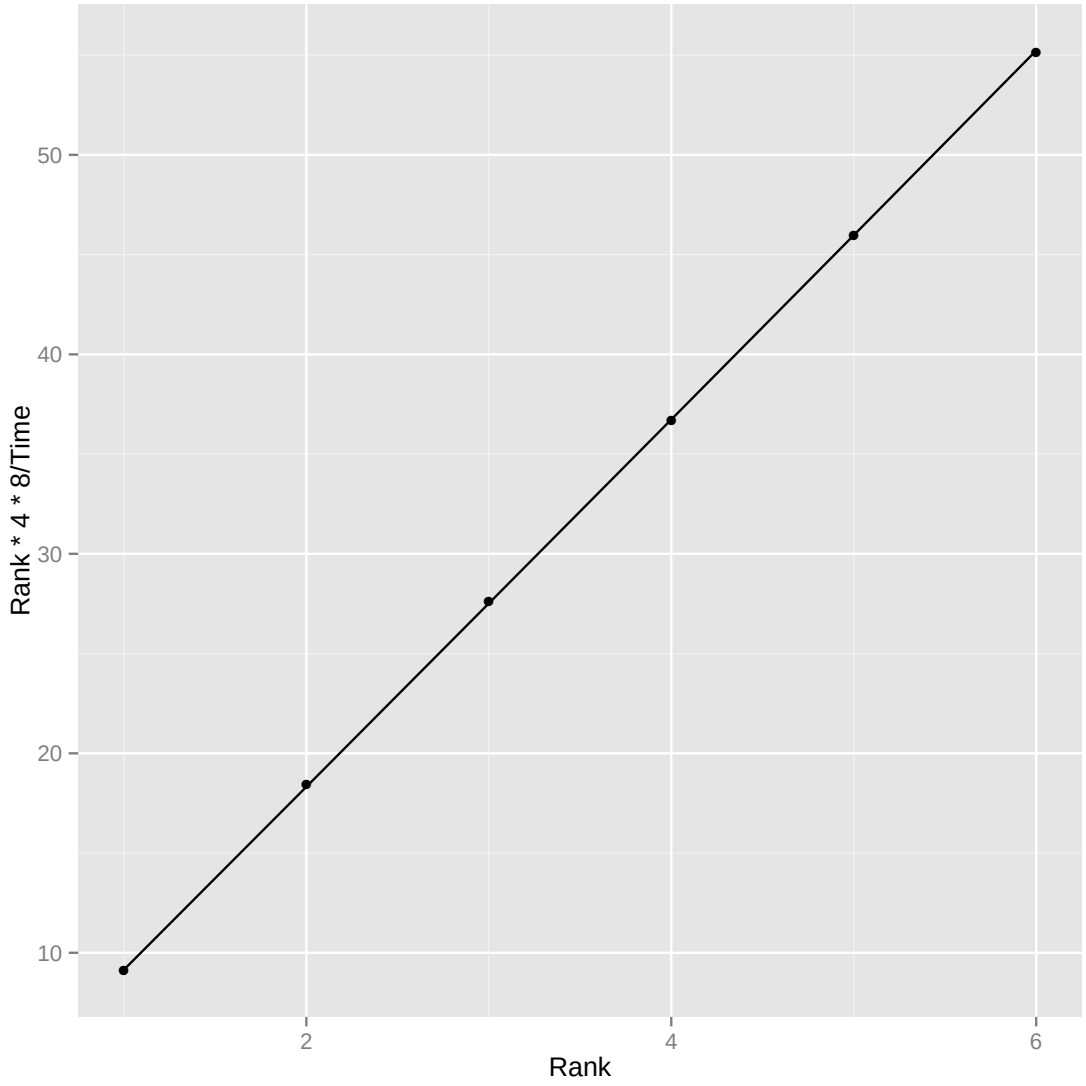
Print the mean time taken for performing a 4MB send at each step, with more and more nodes:

```
pl <- ggplot(data=timing, aes(x=Rank,y=Time))+geom_point()
pl + geom_line(aes(x=Rank,y=values_test), colour="black")
```



Print the bandwidth reached when performing num_iters sends at each step, with more and more nodes (in Mbps):

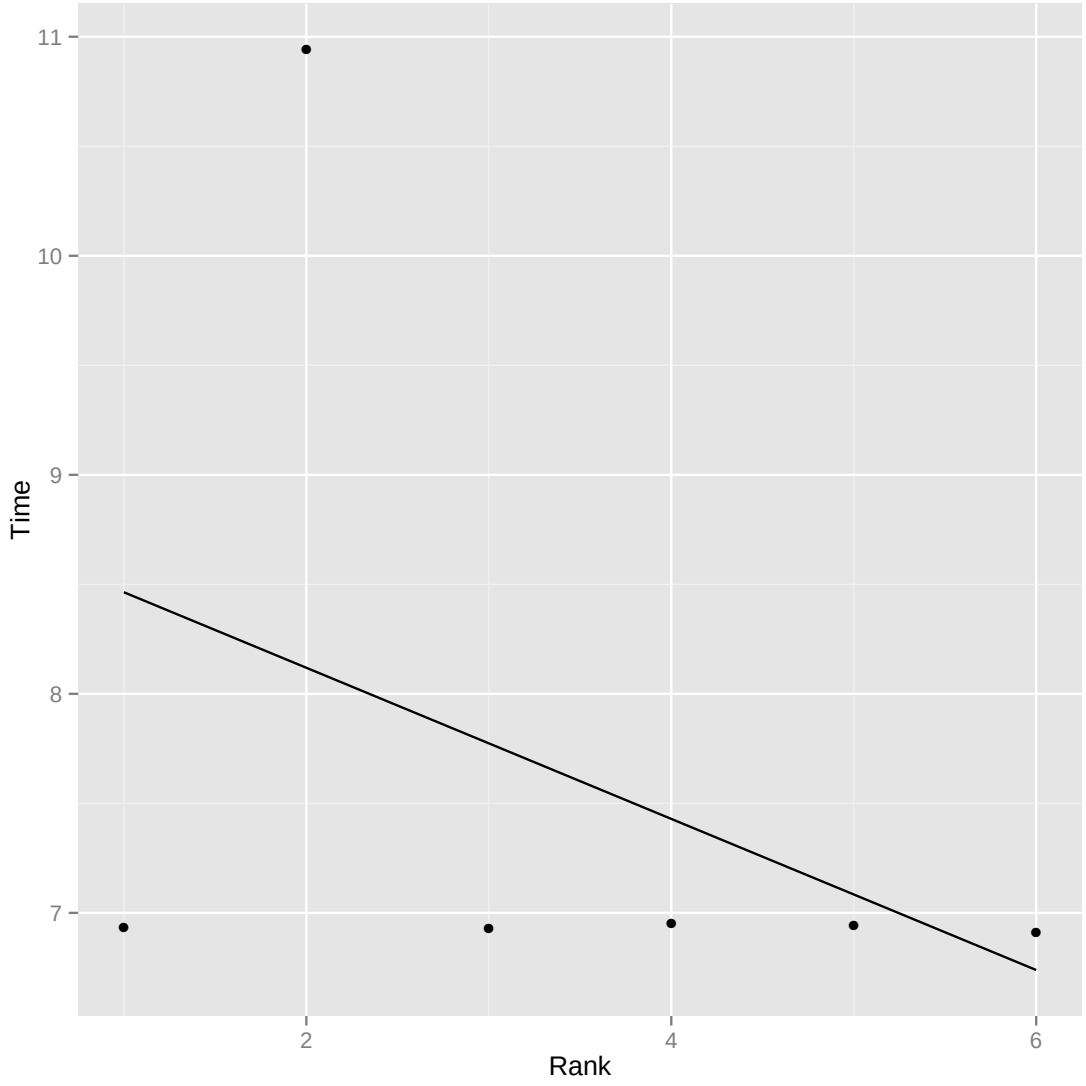
```
pl <- ggplot(data=timing, aes(x=Rank,y=Rank*4*8/Time))+geom_point()
pl + geom_line(aes(x=Rank,y=Rank*4*8/values_test), colour="black")
```



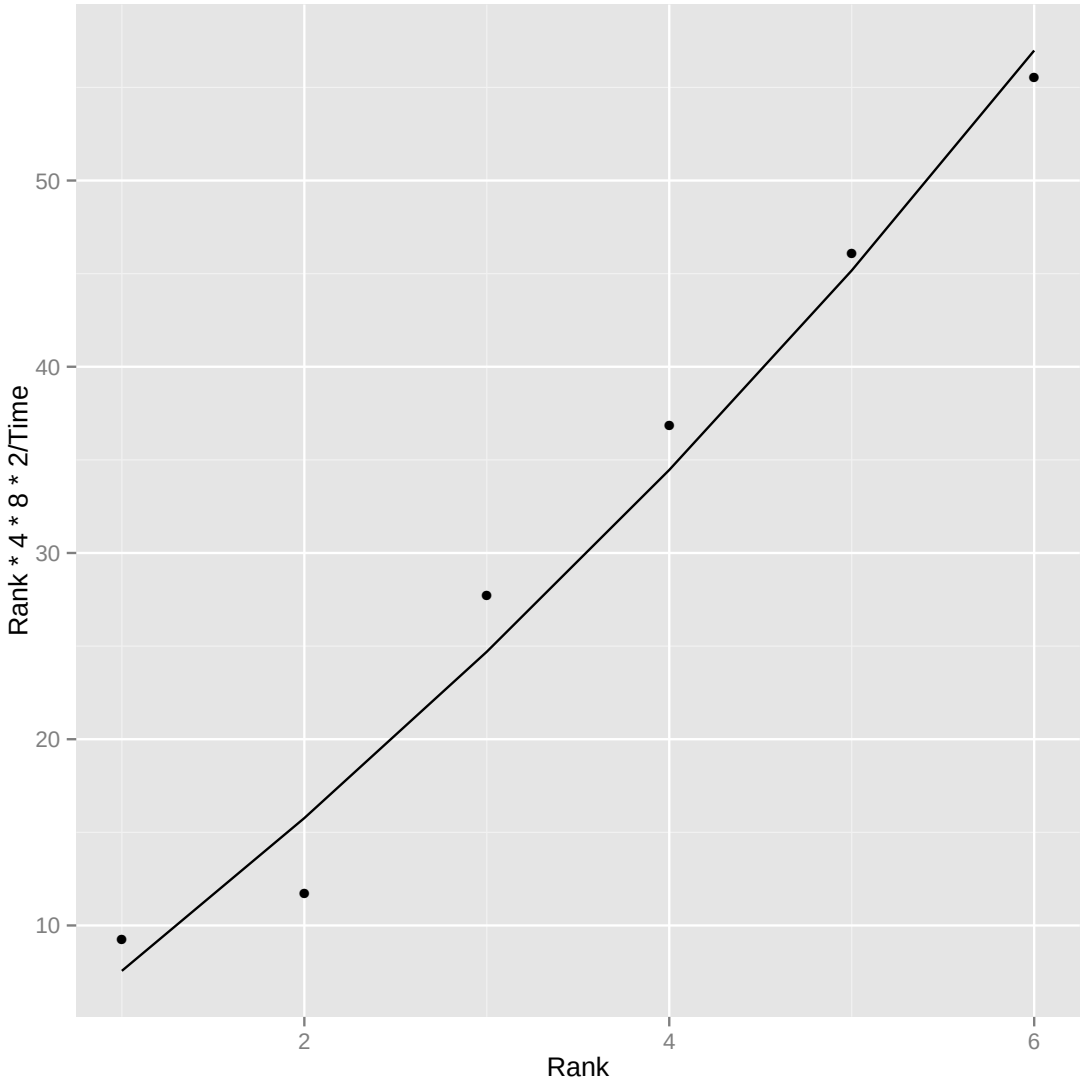
Test 2 : a sendrecv, incrementing from one sender each time, tests the fullduplex capacity of the links (n steps at the end)

```
times <-df_sendrecv_same[df_sendrecv_same$ResourceId==0,]$Duration
means <-apply(matrix(times,ncol=num_senders,nrow=num_iters), 2, mean)
timing<-data.frame()
timing <-cbind(means, index)
timing<-as.data.frame(timing)
names(timing)= c("Time", "Rank")
lm_small <- lm(Time ~ Rank, data=timing[timing$Rank<=thresh,])
if(num_senders>thresh){
lm_large <-lm(Time ~ Rank, data=timing[timing$Rank>thresh ,])
}else
lm_large <-lm_small

sendrecv1_function <- function(x) ifelse(x<=thresh, x *( coefficients(lm_small)[2] )+ coefficients(l

timing$values_test = sendrecv1_function(timing$Rank)
```

```
pl2 <- ggplot(data=timing, aes(x=Rank,y=Time))+geom_point()
pl2 + geom_line(aes(x=Rank,y=values_test), colour="black")
```



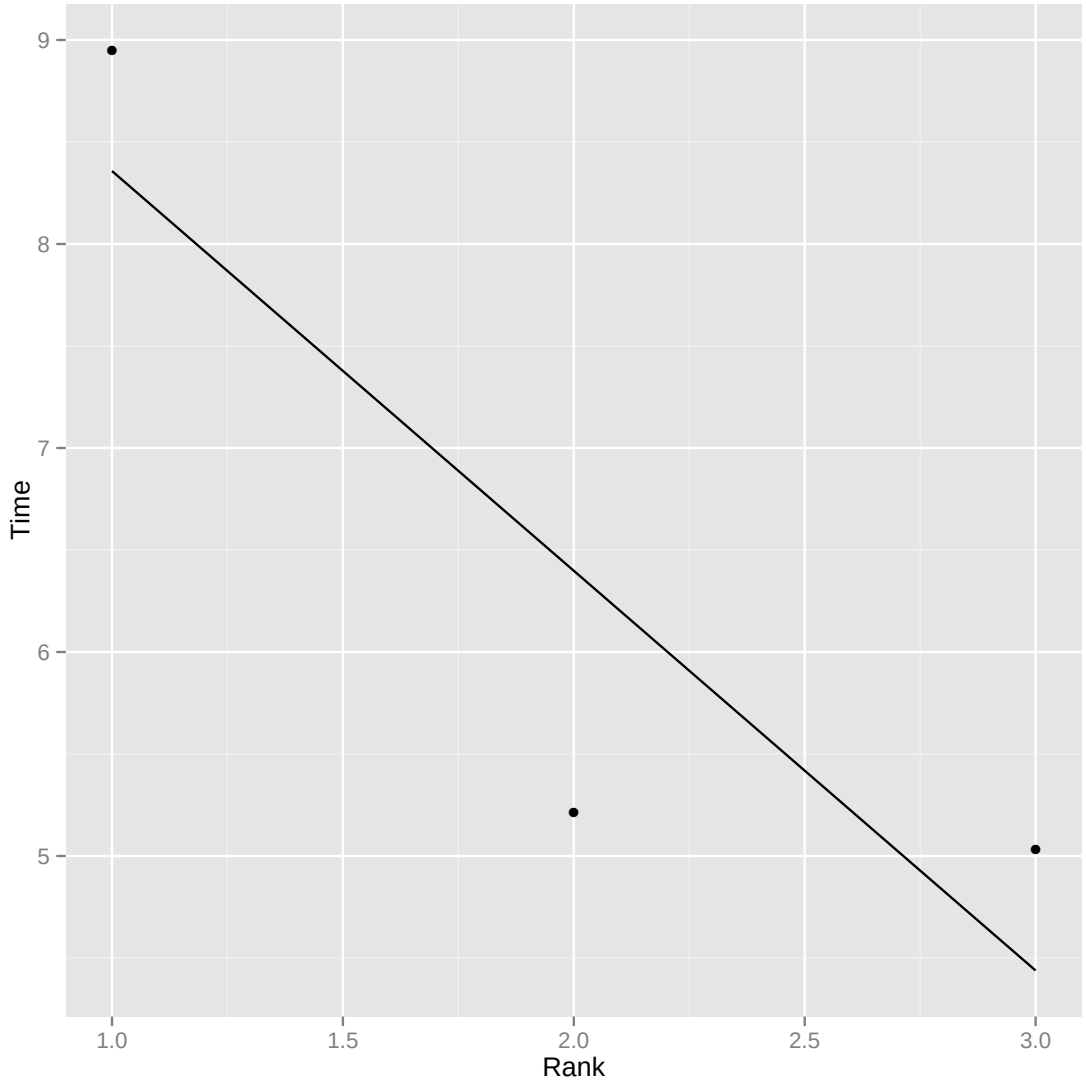

```
pl <- ggplot(data=timing, aes(x=Rank,y=Rank*4*8*2/Time))+geom_point()
pl + geom_line(aes(x=Rank,y=Rank*4*8*2/values_test), colour="black")
```



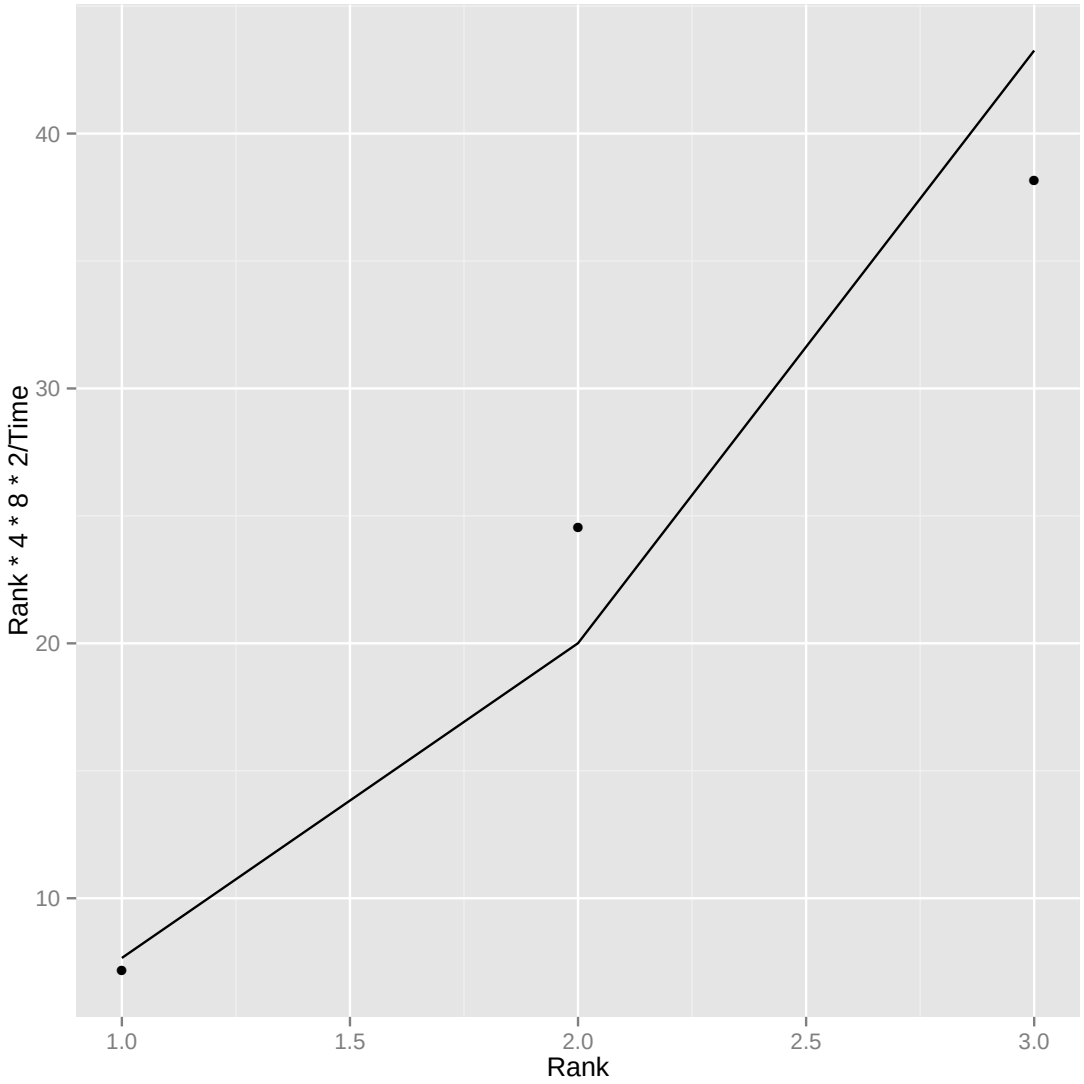
Test 3 : have each process send and receive at the same time, but from a different node. (two pairs of processes are used to cross their streams) So at each step we add two more nodes at each side (n/2 steps at the end)

```
index2<-c()
for(rank in 0:(num_senders/2-1)) {
    index2<-c(index2,rank+1)
}
times <-df_sendrecv_diff[df_sendrecv_diff$ResourceId==0,]$Duration
means <-apply(matrix(times,ncol=num_senders/2,nrow=num_iters), 2, mean)
timing<-data.frame()
timing <-cbind(means, index2)
timing<-as.data.frame(timing)
names(timing)= c("Time", "Rank")
lm_small <- lm(Time ~ Rank, data=timing[timing$Rank<=thresh,])
if(num_senders>thresh){
lm_large <-lm(Time ~ Rank, data=timing[timing$Rank>thresh,])
}else
```

```
lm_large <-lm_small



sendrecv2_function <- function(x) ifelse(x<=thresh, x *( coefficients(lm_small)[2] )+ coefficients(l

timing$values_test = sendrecv2_function(timing$Rank)

ggplot(data=timing, aes(x=Rank,y=Time))+geom_point() + geom_line(aes(x=Rank,y=values_test), colour="
```




```
pl <- ggplot(data=timing, aes(x=Rank,y=Rank*4*8*2/Time))+geom_point()
pl + geom_line(aes(x=Rank,y=Rank*4*8*2/values_test), colour="black")
```


40
30
20
10
Rank * 4 * 8 * 2/Time
1.0
1.5
2.0
2.5
3.0
Rank