SMPI Collective Operations Analysis

Arnaud Legrand

March 17, 2015

Contents

1 Parsing

Let's parse the data collected by Martin

```
grep -i precious collectives/log/* > res.log
```

```
use Data::Dumper;
open(INPUT,"res.log");
open(OUTPUT,"> res.csv");

my(@fields)=qw(date host algo numproc msgsize hostTime hostMem simTime);
print OUTPUT (join(',',@fields)."\n");

while(defined($line=<INPUT>)) {
    chomp $line;
    if(!($line =~ /PRECIOUS_RESULT/)) { next; }
    my(%res)=();

```

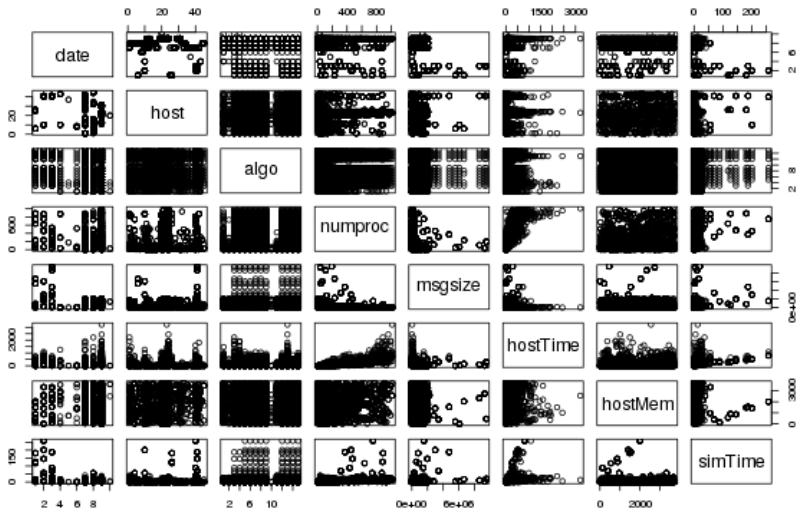
```
    ($head,$tail)=split(/PRECIOUS_RESULT /,$line);
    $head =~ s/.*log\///;
    $head =~ s/.org:$//;
    ($date,$host) = split(/_/,$head);
    #        print "$tail\n";

    my(%res) = split(/[: ]/, $tail);
    $res{date}=$date;
    $res{host}=$host;
    #        print Dumper(\%res);

    my(@out)=();
    foreach(@fields) { push @out, $res{$_}; }
    #        print Dumper(\@out);
    print OUTPUT (join(',',@out))."\n";
}
close(OUTPUT);
```

2 Analysis

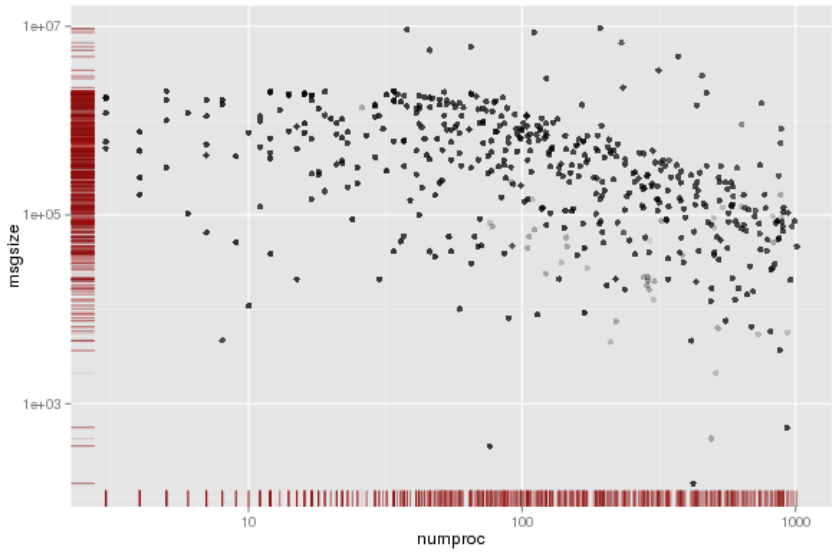
2.1 Global overview

```
library(ggplot2)
library(plyr)
df=read.csv("res.csv",header=T,strip.white=T);
```

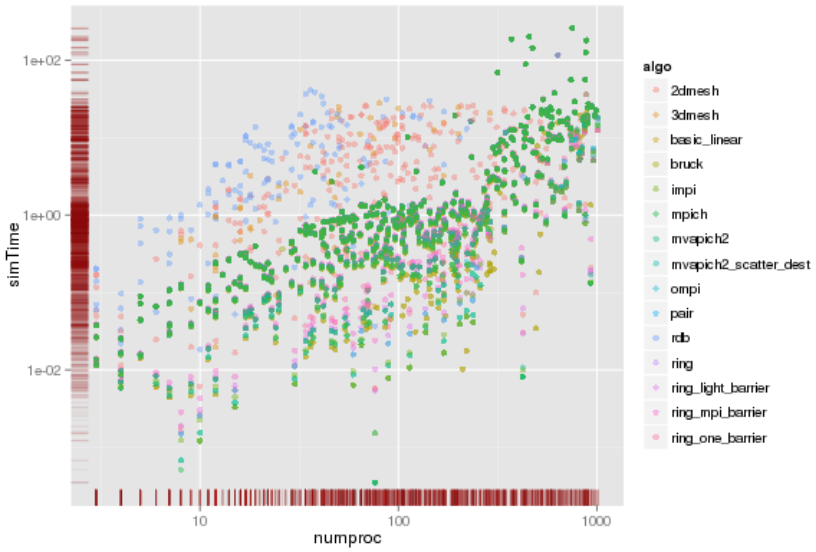
```
plot(df)
```




```
ggplot(data=df,aes(x=numproc,y=msgsize)) + geom_point(alpha=.1) +
  geom_rug(alpha=.05, color="dark red") + scale_x_log10()  + scale_y_log10()
```



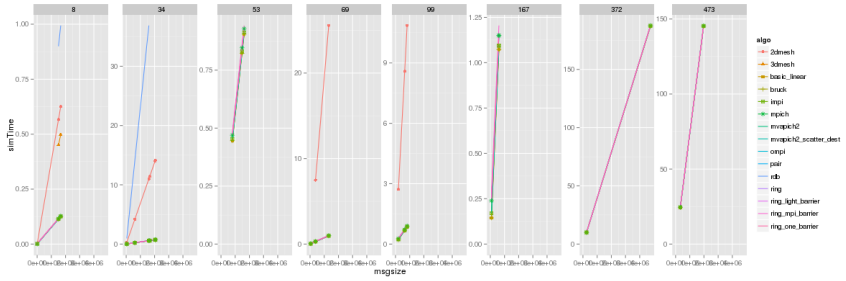

```
ggplot(data=df,aes(x=numproc,y=simTime,color=algo)) + geom_point(alpha=.4) +
  geom_rug(alpha=.05, color="dark red") + scale_x_log10()  + scale_y_log10()
```




```
ggplot(data=df,aes(x=msgsize,y=simTime,color=algo)) + geom_point(alpha=.4) +
  geom_rug(alpha=.05, color="dark red") + scale_x_log10() + scale_y_log10()
```

It's hard to really tell from the previous figures because there is too much data with different orders of magnitude but as can be seen in the next graph, `simTime` is roughly linear in `msgsize` and in `numproc`.

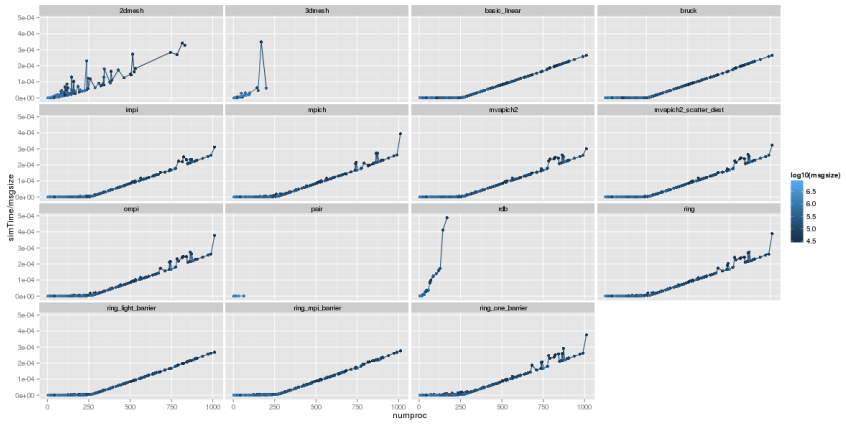
```
#sizes = sample(df[df$numproc>256,]$numproc , size=8, replace=F)
sizes = c(8, 34, 53, 69, 99, 167, 372, 473)
ggplot(data=df[df$numproc %in% sizes,],
       aes(x=msgsize, y=simTime, color=algo, shape=algo)) +
    geom_line() + geom_point() + ylim(0,NA) +
    facet_wrap(~numproc,scales="free_y",nrow=1)
# + scale_colour_brewer(palette="Set1")
```



Therefore, in the following, I focus on `simTime/msgsize` and to avoid weird effects, I arbitrarily decide to focus on `msgsize` > 25K. This allows me to get `msgsize` "out of the picture".

2.2 Linearity

```
ggplot(data=df[df$msgsize > 25000,],
       aes(x=numproc, y=simTime/msgsize, color=log10(msgsize))) +
    geom_line() + geom_point() +
    facet_wrap(~algo) # + scale_colour_brewer(palette="Set1")
```

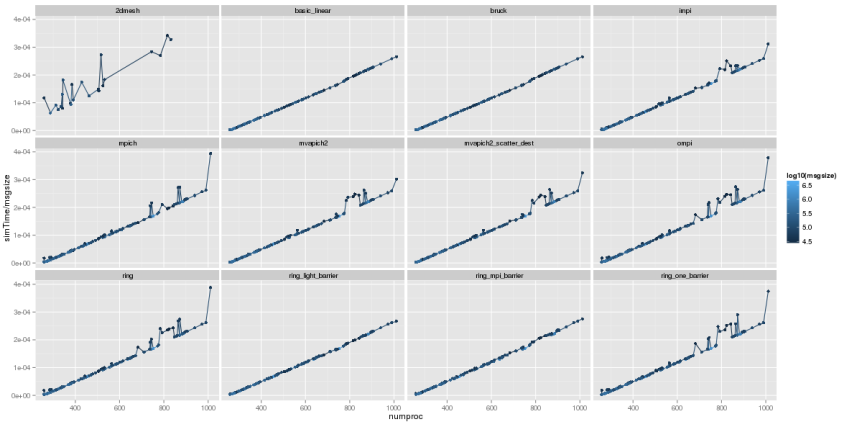


Interesting. It is piece-wise linear. This can easily be explained by the

fact that everything fits in a cabinet for `numproc` $<= 256$. Let's look at both settings separately.

2.3 Large deployments

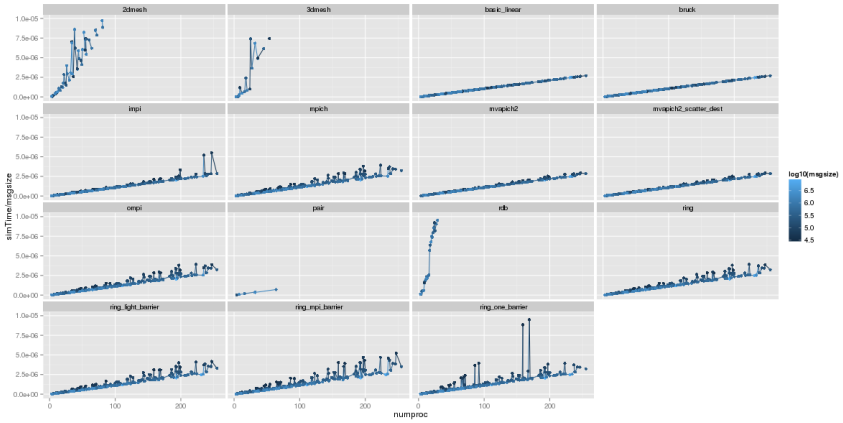
```
ggplot(data=df[df$msgsize > 25000 & df$numproc>256,],
       aes(x=numproc, y=simTime/msgsize, color=log10(msgsize))) +
    geom_line() + geom_point() +
    facet_wrap(~algo) # + scale_colour_brewer(palette="Set1")
```



Some algorithms (`basic_linear` and `bruck` `ring_light_barrier` and `ring_mpi_barrier`) perform quite nicely. They are very stable and seem to all have the same best performance. Surprisingly `mpich`, `ompi`, `mvapich2` or `impi` have much more irregular behavior and sometime perform quite bad decision. It is surprising that this is solely due to the chosen algorithm and not to the instability of the platform...

2.4 Small Deployments

```
ggplot(data=df[df$msgsize > 25000 & df$numproc<256,],
       aes(x=numproc, y=simTime/msgsize, color=log10(msgsize))) +
    geom_line() + geom_point() + ylim(0,1E-5) +
    facet_wrap(~algo) # + scale_colour_brewer(palette="Set1")
```



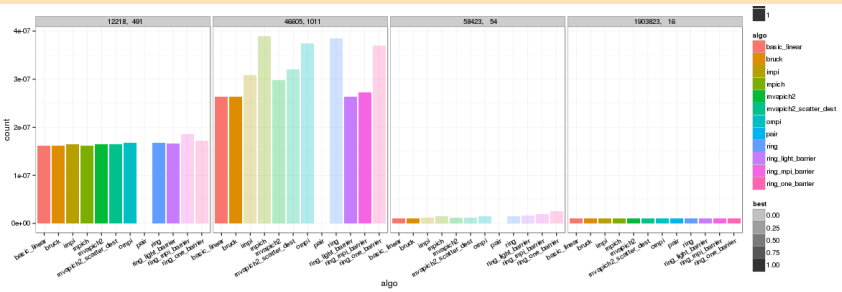
Again, some algorithms (`basic_linear` and `bruck`, and `mvapich2`) behave very well and are good regardless of the number of nodes involved. Again `mpich`, `impi` and `ompi` exhibit some surprising variability. The `ring_light_barrier` and `ring_mpi_barrier` are now less stable than in the large deployment setting.

2.5 Is there a "best" algorithm ?

Note that in the previous graphs, it is clear that some algorithms are more stable than others but they may be actually slightly less good in term of performance than more unstable ones. Let's find out by computing whenever for a given configuration (`msgsize`, `numproc`) an algorithm is within 5% of the best one.

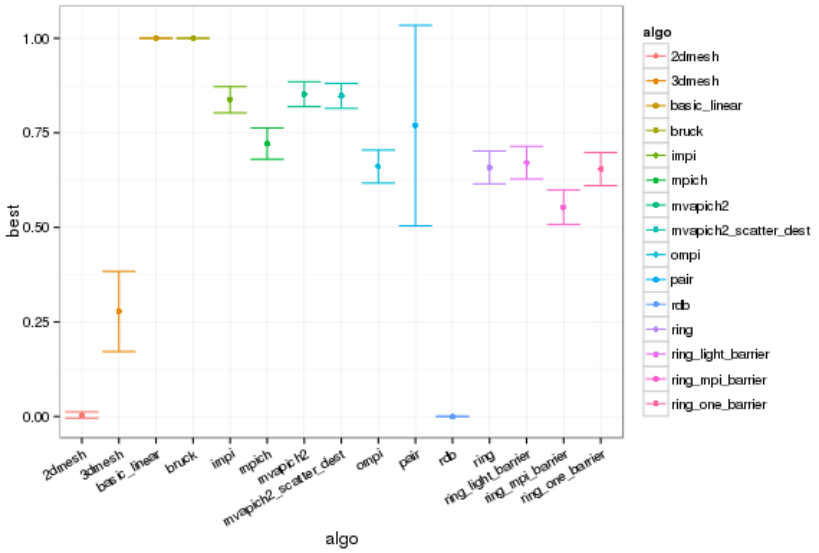
```
df = ddply(df,c("msgsize","numproc"), transform, min = min(simTime))
df = df[!is.na(df$min),]
df$best = 0
df[df$simTime <= 1.05 * df$min,]$best = 1
```

```
algo_excluded= c("2dmesh","3dmesh", "rdb")
msgsize = sample(df$msgsize , size=4, replace=F)
d = df[df$msgsize %in% msgsize & !(df$algo %in% algo_excluded),]

ggplot(data=d,
       aes(x=algo, weight=simTime/msgsize/numproc, fill=algo,
           alpha=best, size=factor(best))) +
    geom_bar() + theme_bw() + scale_alpha(range = c(0.3, 1)) +
    theme(axis.text.x = element_text(angle = 30, hjust=1)) +
    facet_wrap(~msgsize*numproc, nrow=1)
```



From such graphs, it really looks again that `bruck` and `basic_linear` always perform better than the other algorithms. Let's compute some statistics...

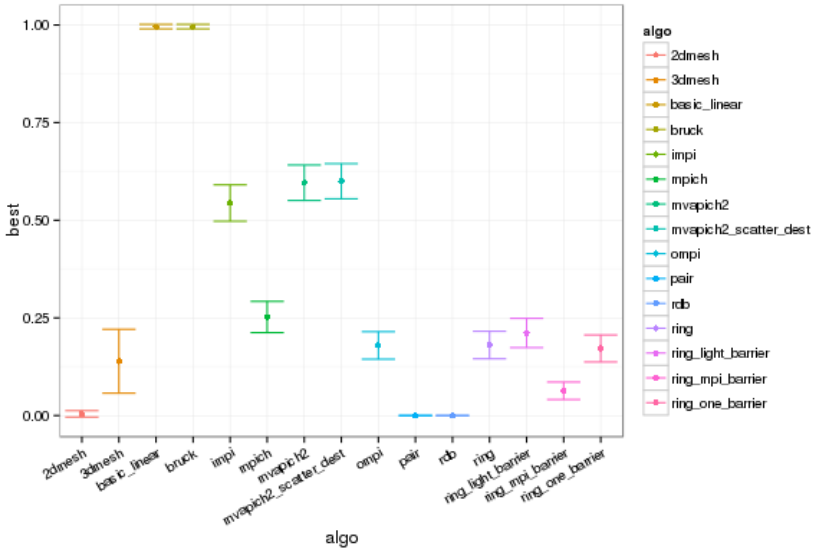
```
d = summarySE(df,c("best"),c("algo"))

ggplot(data=d, aes(x = algo, y=best, ymin=best-ci,
                   ymax=best+ci, color=algo)) +
  geom_point() + geom_errorbar() + theme_bw() +
   theme(axis.text.x = element_text(angle = 30, hjust=1))
```



OK. Now, it is clear that `bruck` and `basic_linear` are always within 5% of the optimal whereas `impi`, `mpich`, `mvapich2` and `ompi` are definitely not.

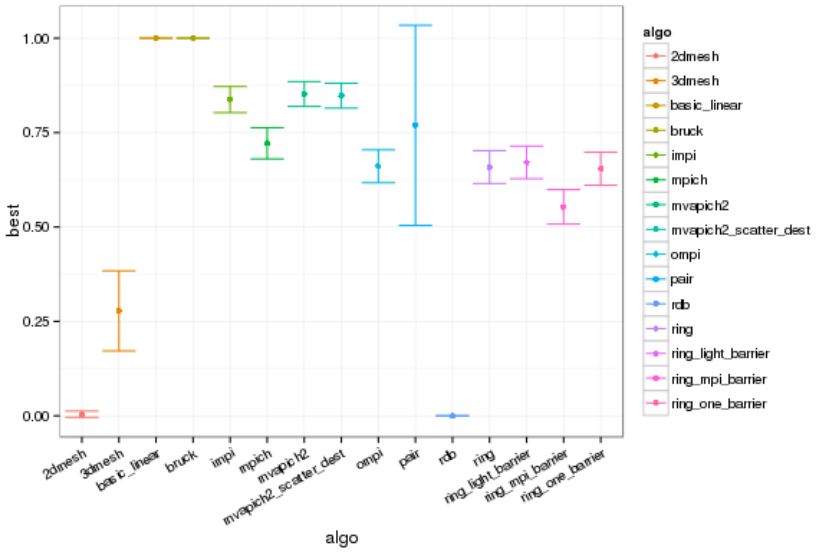
The same keeps holding true with 2%

```
df$best = 0
df[df$simTime <= 1.02 * df$min,]$best = 1

d = summarySE(df,c("best"),c("algo"))

ggplot(data=d, aes(x = algo, y=best, ymin=best-ci,
                   ymax=best+ci, color=algo)) +
  geom_point() + geom_errorbar() + theme_bw() +
   theme(axis.text.x = element_text(angle = 30, hjust=1))
```



The same keeps holding true with 10%

```
df$best = 0
df[df$simTime <= 1.1 * df$min,]$best = 1

d = summarySE(df,c("best"),c("algo"))

ggplot(data=d, aes(x = algo, y=best, ymin=best-ci,
                   ymax=best+ci, color=algo)) +
  geom_point() + geom_errorbar() + theme_bw() +
   theme(axis.text.x = element_text(angle = 30, hjust=1))
```



2.6 Conclusion

In this particular setup, `bruck` and `basic_linear` perform much better than other more elaborate solutions and the smart optimizations of classical MPI implementations seems not only ineffective but even harmful.

A similar analysis should probably be conducted with other topologies.