28026598
戴新颜

1. Design

Explain the detail of your implementations of three versions in diagrams, figures, sentences, you
also need to answer all the questions in the following aspects:

(a) What algorithm you choose to implement Pthread version? Why?

Dijkstra 算法。
sequence 的算法复杂度为 O(n^2)，在不考虑 barrier time 的情况下，多线程运行的算法复杂度接近于 O(n*n/threadnumber).

具体实现：
Dijkstra 算法嵌套了两个循环， PThread 对内部循环计算并行化， 在每一个内部循环里：

1. 使用类似于 mapReduce 的方式计算出下一个(V-S)集合中的最近顶点。
   a) 每个 thread 计算(V-S)集合中
      threadRank*N/threadnumber 到 （threadRank+1）*N/threadnumber 的最近顶点
      （复杂度 O(N/threadnumber)）
   b) 比较上一步所有 thread 挑选的最近顶点， 挑选最近顶点计入 S 集合
      （复杂度 O(threadNumber)）

2. 使用上一步挑选的顶点更新所有顶点到源顶点的距离。
   每个 thread 更新顶点编号属于
   threadRank*N/threadnumber 到 （threadRank+1）*N/threadnumber
   的距离。
   （复杂度 O(N/threadnumber)）

(b) What are the pros and cons of synchronous and asynchronous version?

两个版本的通信成本都非常高。
synchronous 版本需要在每一个迭代后面进行一次集体通信。asynchronous 版本都是自行判断与运算，直到自己运算结束，可以减少通信次数。

但是 asynchronous 版本使用 dual-pass ring 算法检测 terminal 条件，有可能频繁出现被发送停止消息的节点在发送后立马又被激活，对于存在大量由低 rank 连向高 rank 的图会有比较高的通信成本。

(c) Other efforts you've made in your program

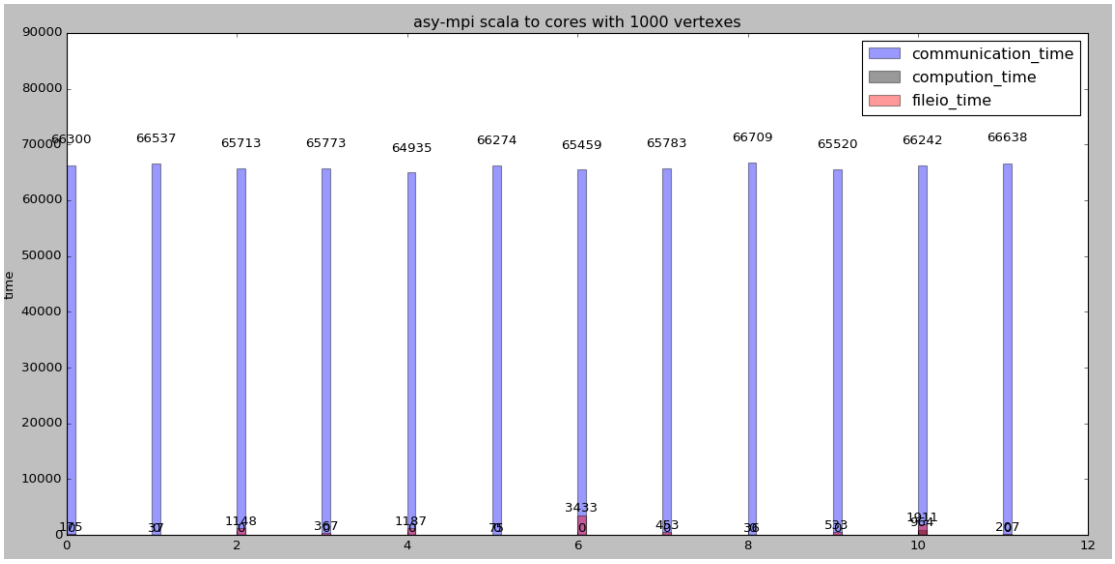
实现了一个 openmp 的版本。

2. Performance analysis

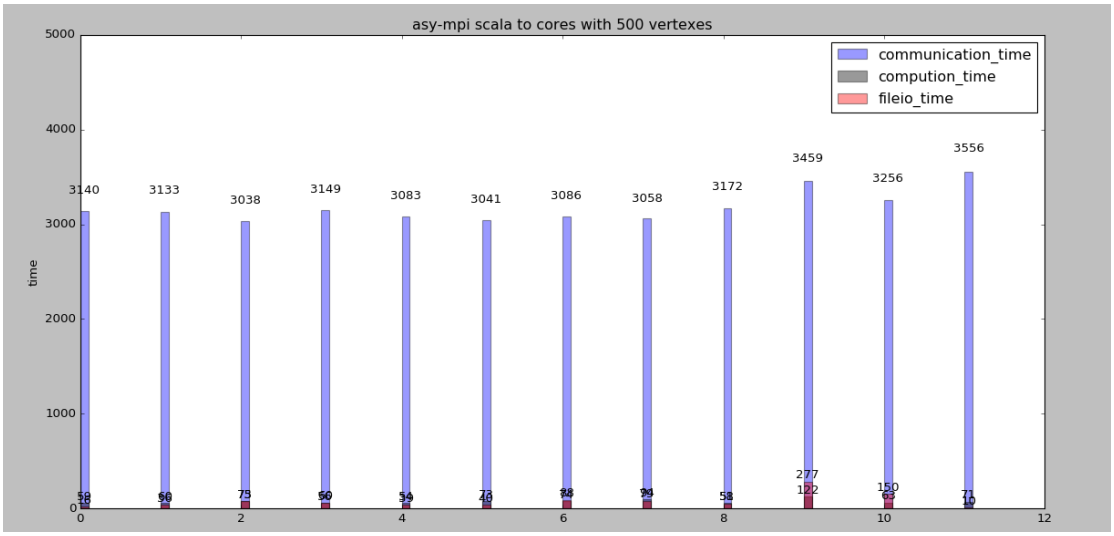
Strong scalability – scalability to number of cores (Problem size is fixed)

asychronize

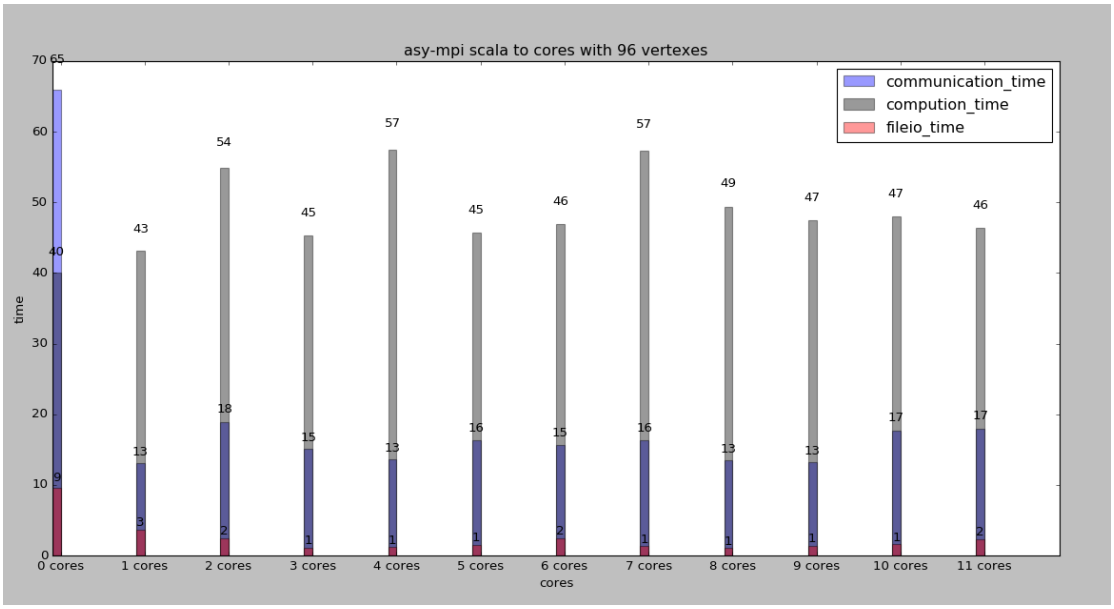
1000 v



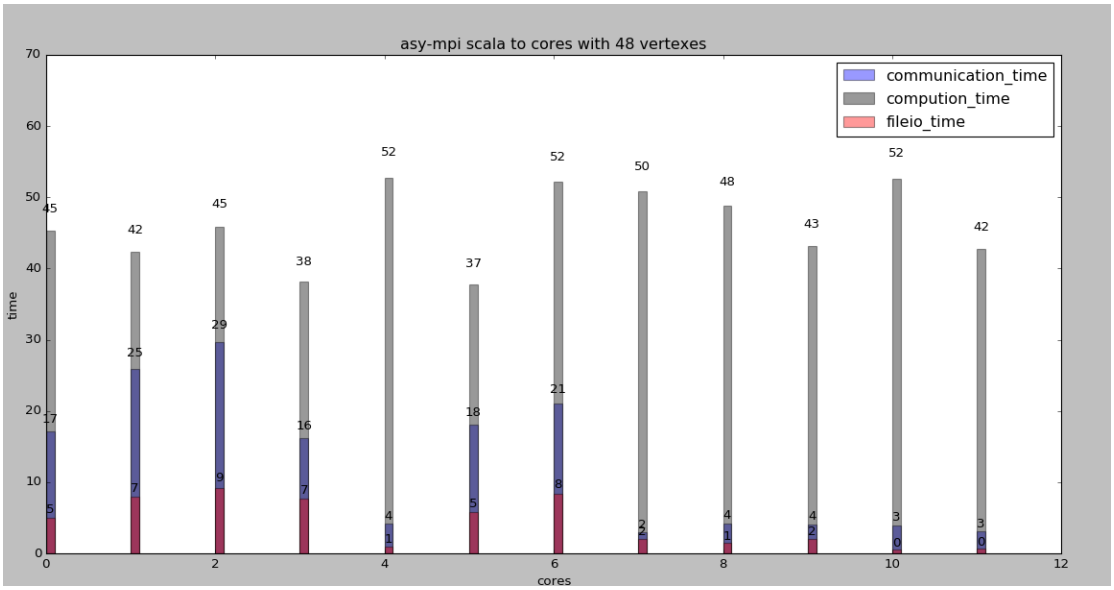
500v


asy-mpi scala to cores with 500 vertexes
communication_time
compution_time
fileio_time
time
3140
3133
3038
3149
3083
3041
3086
3058
3172
3459
3256
3556
277
122
150
71

96v

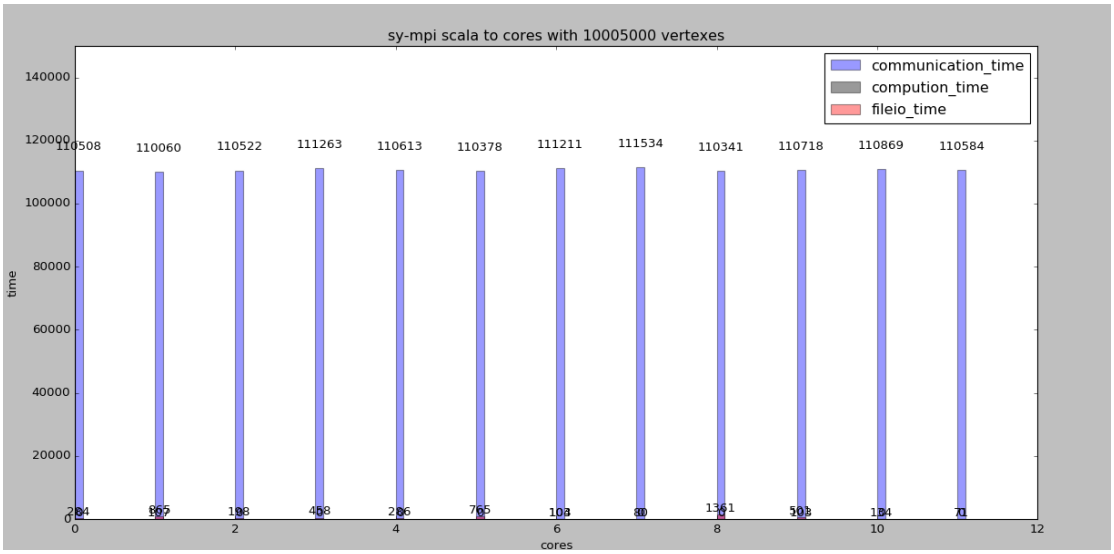

asy-mpi scala to cores with 96 vertexes
communication_time
compution_time
fileio_time
time
cores
0 cores
1 cores
2 cores
3 cores
4 cores
5 cores
6 cores
7 cores
8 cores
9 cores
10 cores
11 cores
65
43
54
45
57
45
46
57
49
47
47
46

48v

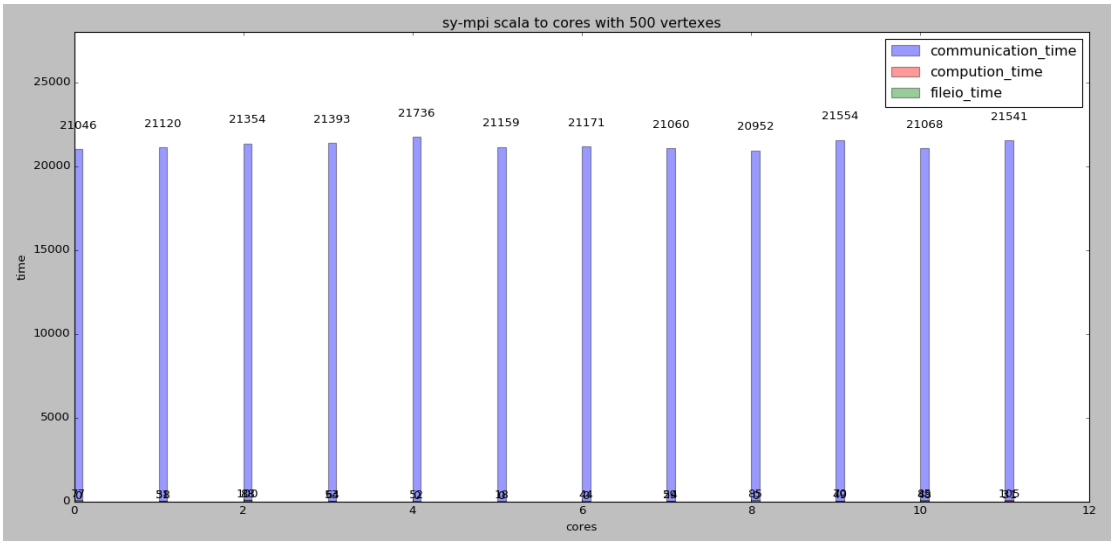


synchronize

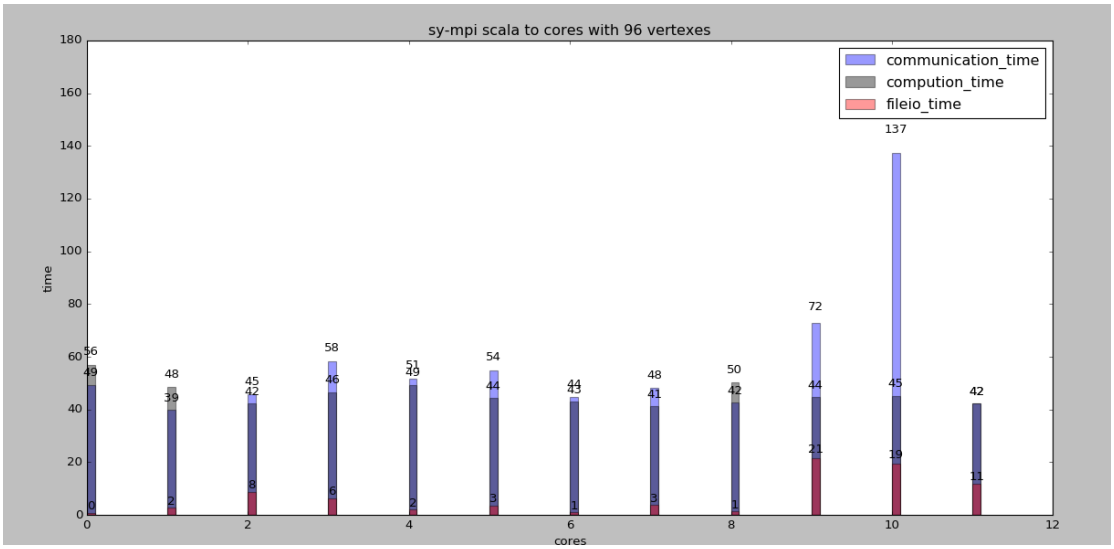
1000v



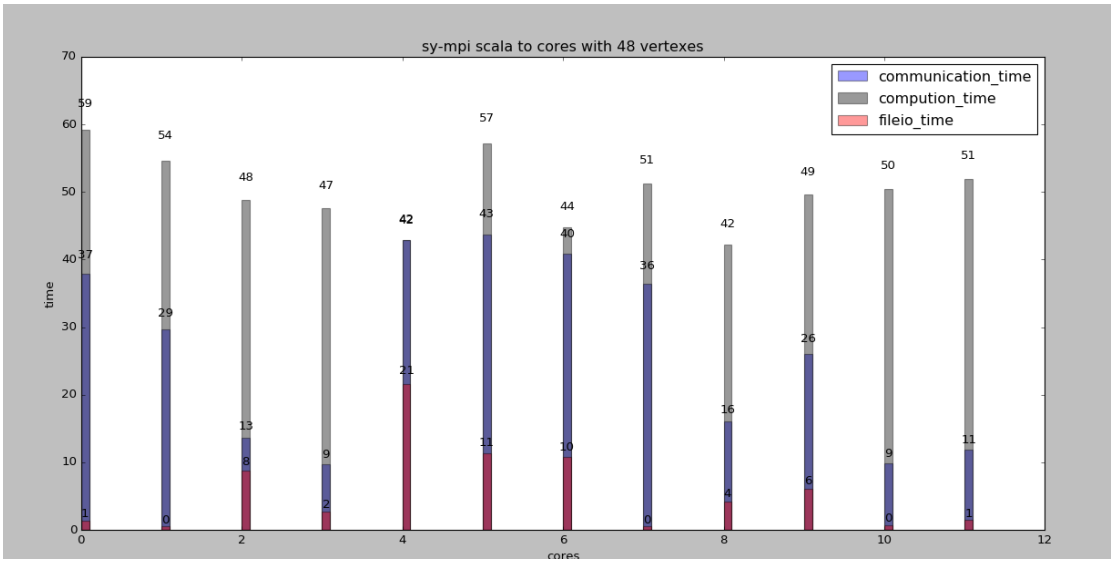
500v


sy-mpi scala to cores with 500 vertexes
communication_time
compution_time
fileio_time
time
cores
21046
21120
21354
21393
21736
21159
21171
21060
20952
21554
21068
21541

96v

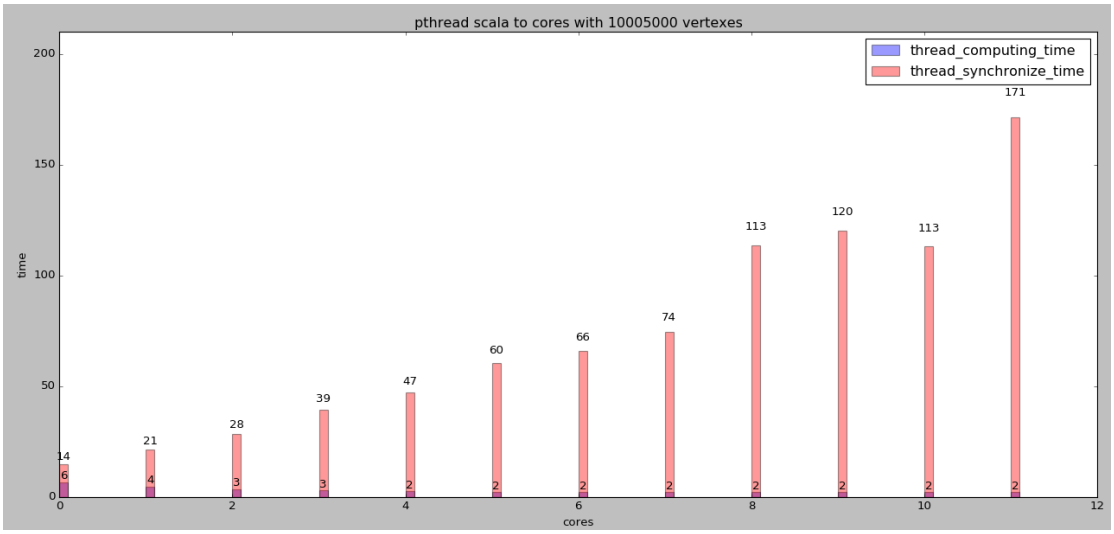

sy-mpi scala to cores with 96 vertexes
communication_time
compution_time
fileio_time
time
cores
56
48
45
58
51
54
44
48
50
72
137
45
42

48v

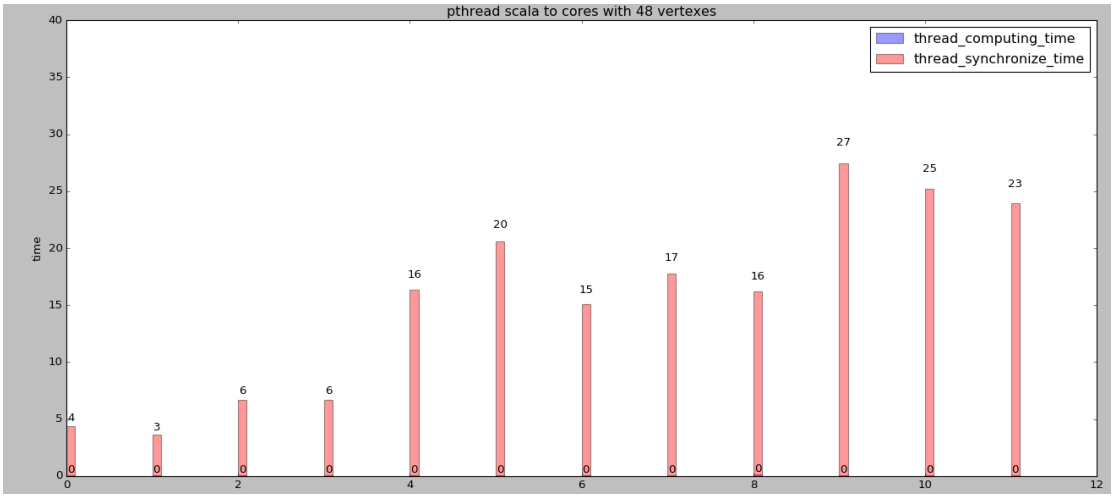


pthread

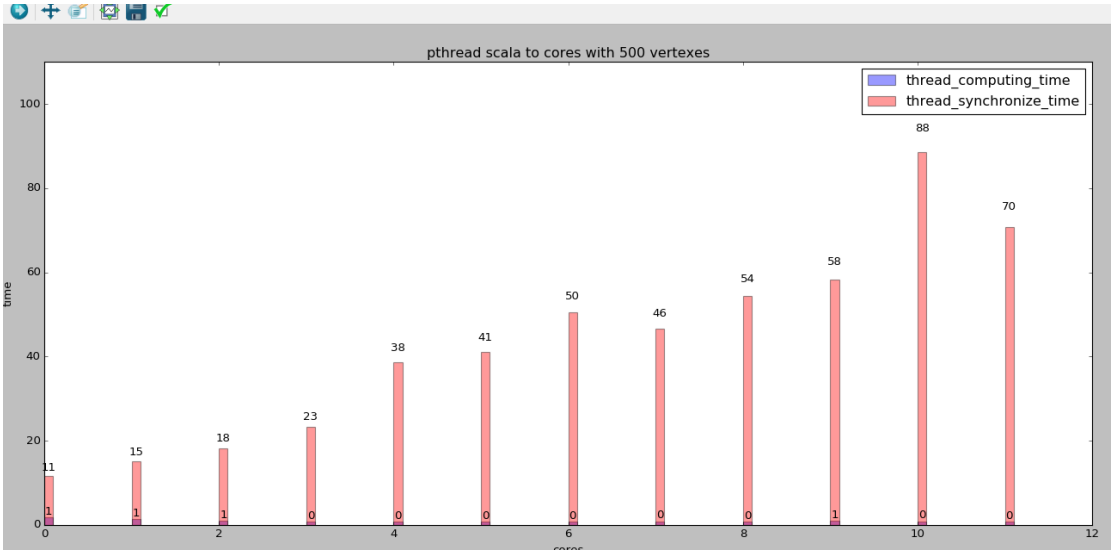
1000v



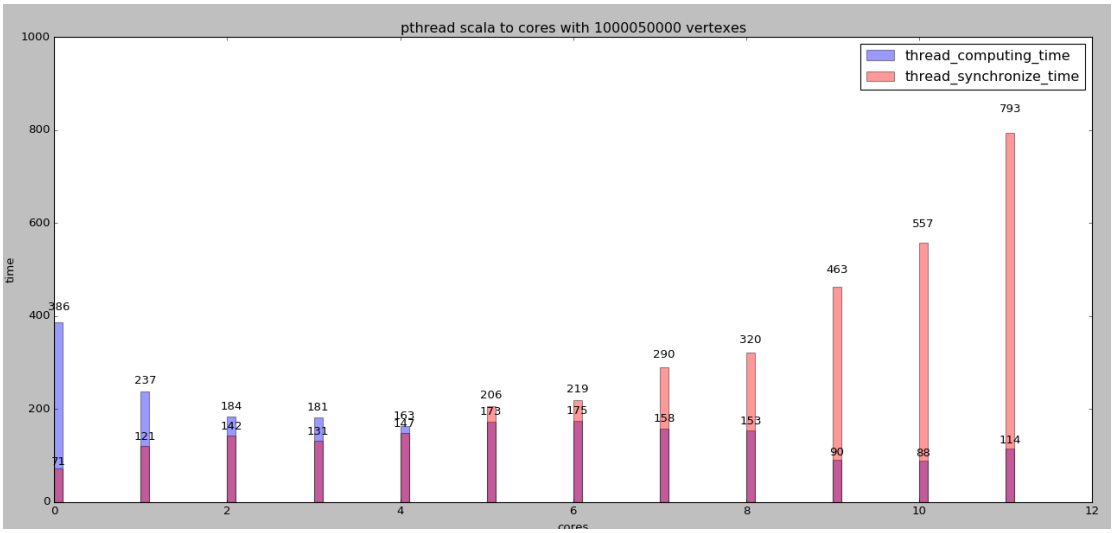
48v


pthread scala to cores with 48 vertexes
thread_computing_time
thread_synchronize_time
time
4
3
6
6
16
20
15
17
16
27
25
23

500v


pthread scala to cores with 500 vertexes
thread_computing_time
thread_synchronize_time
time
11
15
18
23
38
41
50
46
54
58
88
70

10000v



Time distribution for Pthread:

已经在上述各图中给出

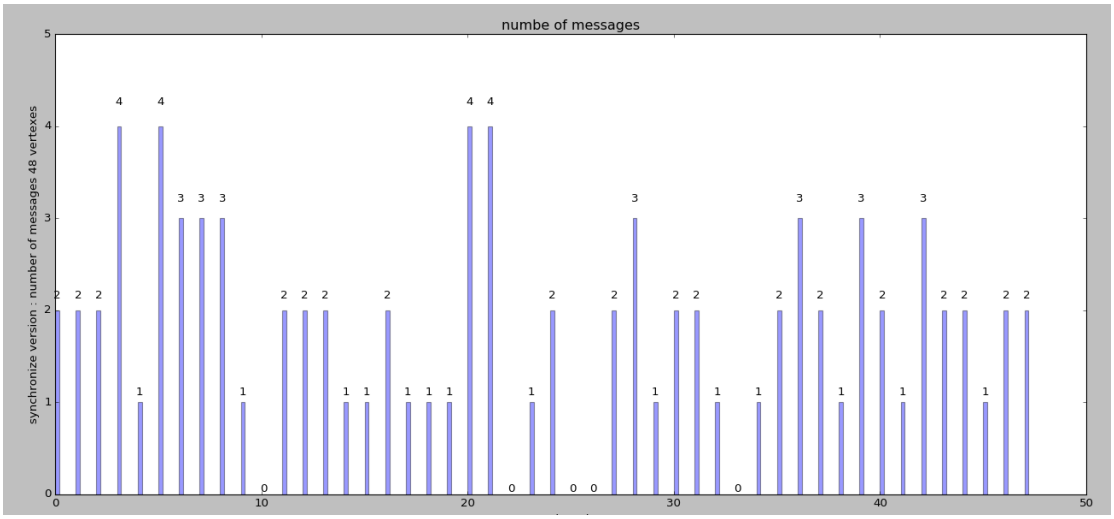
Time distribution for MPI:

已经在上述各图中给出

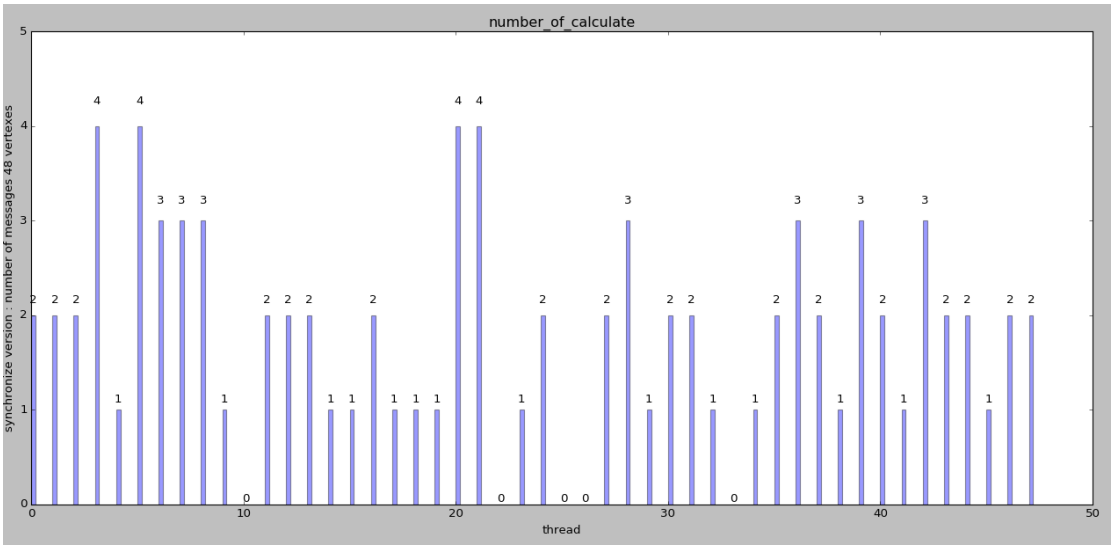
Load distribution on processes (vertices) for MPI:

asynchronize

1. number of messages

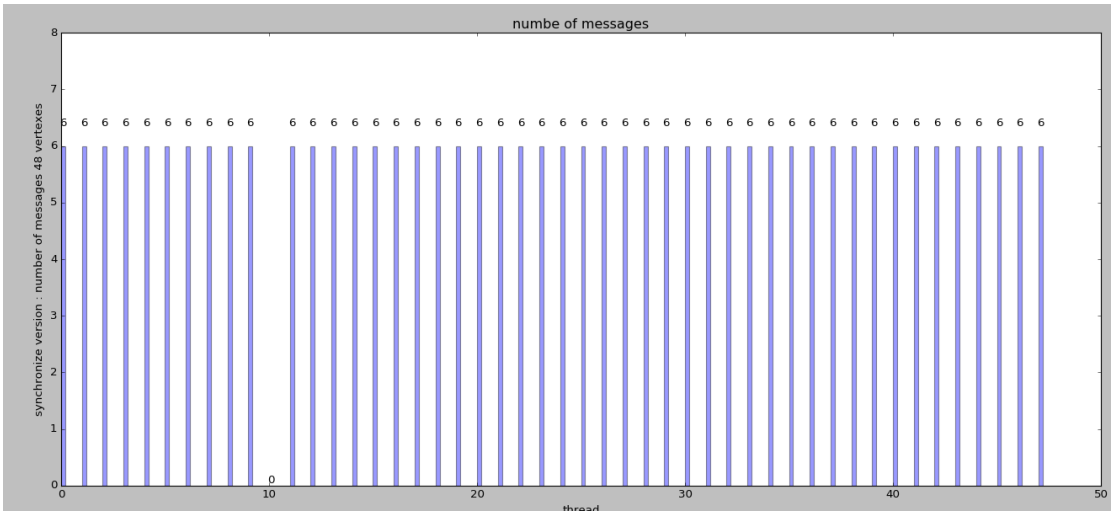


2. number of calcualte



3. number of update

synchronize



Experience and conclusion

(a) What have you learned and observed from this assignment?
MPI 的阻塞用法与非阻塞用法。费阻塞用法可以提高更高的效率，还可以避免死锁。
MPI 的集体通信。
pthread 的同步方法。

(b) What difficulty did you encounter when implementing this assignment?
MPI 在指定 process 运行的时候出现莫名的错误。