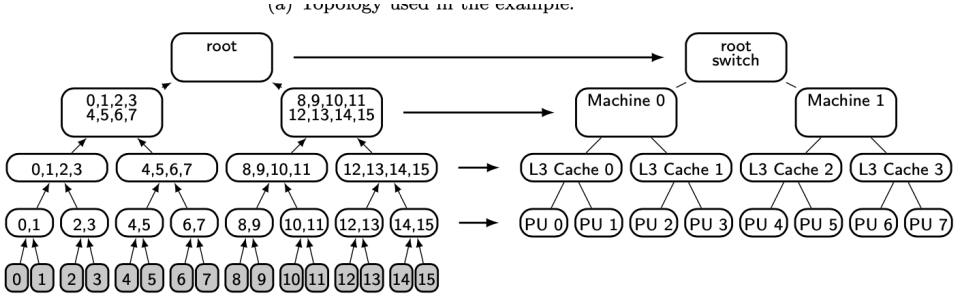
Eager map or Choice map 改进

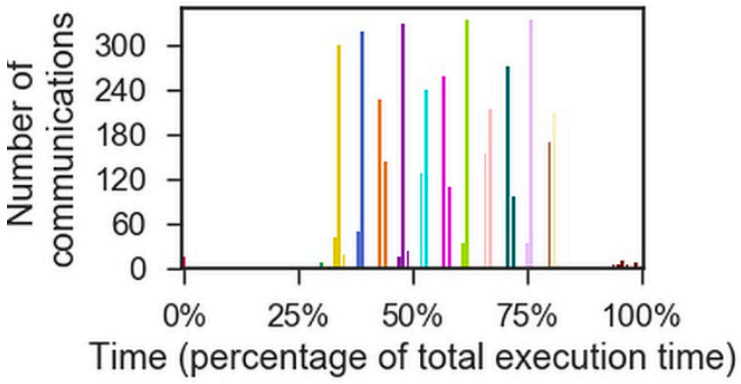
由于这两个算法只注重locality[1],而忽视sockets 的负载平衡。
考虑在对（t1,t2）双线程组成的线程组进行映射时，引入balance机制：
如下图 (0,1),(2,3) 与 （4,5),(6,7）尽量避免组内同时通信，（8,9),(10,11）与（12，13)，(14，15）同理



(b) Task group tree. (c) Architecture hierarchy tree.

思路如下：
首先使用Engermap,对线程（0, 1, 2, 3, 4,.....15）进行分组:(0,1), (2,3)....., (14,15)
分组仍是自底向下，将通信量大的并为一组。
其次，对这些分好的组，进行numa节点（socket）划分,尽量避免两个同时进行通信的组分到一个socket.
需要统计：
(0, 1）通信时间戳集合 [t1,t12,.....tn]
(2, 3）通信时间戳集合 [t1,t12,.....tn]
.
.
.

借鉴DeLoc[2]的方案，将上面个线程组的通信时间戳整合为一个map:
map<int, vector<pair(int,int)> > m = {t1:[(0,1),(2,3)],t2:[(0,1),(4,5)],...,tn:[(8,9),(2,3),(14,15)]}
定义一个comm_map,用于进行k-means分类，统计每个时间戳对应的通信量
map<int,int> comm_map = comm_map = {t1 :2, t2 :2, ...,tn :3}
大致效果如下：




将时间戳靠近的，且通信量相当的分为一组，图中每种颜色代表一类。
k的取值借鉴[3]x-means中BIC的计算.

将每个group的线程组如（0,1）通过m提取出来:group1:{(0,1),(2,3),(8,9)},group2:{(6,7),(4,5),(0,1)}
group3{(10,11),(14,15),(0,1)}.....
需要注意的是，每个group内也可能包括相同的线程组如group1与group3都有(0,1)
group排序：

$$Wp = \frac{Acomm_p}{\sum_{i=1}^{P} Acomm_i}, \quad L_c = \sum_{i=1}^{P_c} W_{p_i}$$

$P_c$: number of thread pairs in group
$P$: total number of pairs

Lc大致表示group内通信量的大小，Wp表示，一个线程对的通信量占所有通信总量

*From the highest to the lowest* $L_c$

| Group-2 | Group-3 |
|---|---|
| Pair-3 (T4, T5) | Pair-2 (T2, T3) |
| Pair-4 (T6, T7) | Pair-5 (T0, T1) |
| Pair-2 (T2, T3) | Pair-4 (T6, T7) |
| Pair-1 (T1, T2) | Pair-1 (T1, T2) |

*From the highest to the lowest* $Wp$

通过Lc,与Wp排序，得到group集合，靠前的group表明这个group最容易发生memory堵塞
需要优先处理。
处理时在同一group的线程对，需要分散放置到不同的socket上
遍历处理每个group，直至处理完所有，如果当前pair已被分配socket，则不处理

之后则每个socket处理各自的，将每个socket内的线程pair用Eagermap的方法映射到L3 cache上，
然后继续往上层映射，直至到达socket层。每个socket可以并行处理。

主要改进是在线程对已生成后，对线程pair进行一个socket分组，目的是减少socket上的访问堵塞，之后就按照之前的算法向上层映射，每个socket内可并行处理。

实现难点：
1 thread pair 通信时间戳提取
2 k-means聚类(或许需要[2]开源代码)

可能的缺陷：
可能时间复杂度较大（进行k-means),目前未评估.

文献源：

sc2018-poster.pdf
1014.37KB

cpe.4692.pdf
1.52MB

Xmeans.pdf
317.71KB

DeLoc_A_Locality... _.pdf
5.3MB

[1] Soomro PN, Sasongko MA, Unat D. BindMe: a thread binding library with advanced mapping algorithms. *Concurrency Computat Pract Exper*. 2018; e4692. https://doi.org/10.1002/cpe.4692

[2]Mulya Agung , Muhammad Alfian Amrizal ,Ryusuke Egawa ,Hiroyuki Takizawa. DeLoc: A Locality and Memory-congestion-aware Task Mapping Method for Modern NUMA Systems January 2020 IEEE Access PP(99):1-1 DOI: 10.1109/ACCESS.2019.2963726

[3] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *Icml*, vol. 1, 2000, pp. 727–734.