Homework 4: MapReduce

Implementation

jobtracker & task tracker:

map & reduce:

job tracker 在 map 與 reduce 的階段會分別與 task trackers 進行三次溝通。

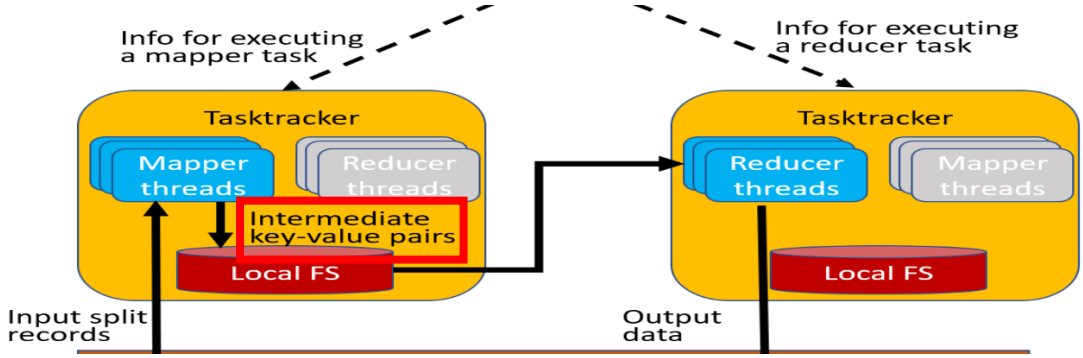
1. 接收 task trackers 的 job request:
   當 job tracker 接收到 request 後，會根據 data locality 去分配工作。透過便利全部的 jobs，當發現存在 job 的 node id 跟 request node id 相同的話，就會優先將該 job 分配給該 node。反之，就會根據 FIFO 的原則，分配 job 給 node。
2. 當全部的 jobs 都被指派出去：
   當全部的 jobs 都被指派出去，job tracker 如果再接收到 job request，就會發送訊息給 task trackers，告之已經沒有工作要做了，讓 task tracker 不用在發送 job request。
3. 發送 job 完成訊息給 job tracker:
   當 jobs 都完成之後，每個 task trackers 會再傳送 jobs 的訊息給 job tracker，像 task id、execution time 等等。方便接下來的紀錄。

mapper threads & reducer threads

1. 程式一開始會先 create 很多 threads，並讓這些 threads 去 job queue 拿 task 執行。當 task trackers 接收到工作之後，會將 job submit 給 thread pool。當 threads 在拿 task 時，首先會將 mutex 鎖住，接下來會檢查 task queue 是否有 task，如果 queue 是空的，就會 wait on 一個 condition variable，直到有新的 task 被加入到 task queue 時，才會重新 signal condition variable，讓 threads 重新拿到新的 task。
2. 當 task tracker 向 job tracker 要求 job request 時，也會用 mutex 跟 condition variable 來操控。每次在發送 request 前，會先檢查每個 threads 是否都有 job，如果每個 job 都有工作的話，就會 wait on condition variable，直到有 threads 空出來，才會重新 signal 這個 condition variable。

Intermediate file

從 spec 可以看到，當 mapper threads 處理完一個 task 時，就會產生 intermediate key-value pairs，我會將他先儲存成 file 的形式。當 job tracker 接收到 task tracker 的完成訊息後，job tracker 就會先創建 N 的 reducer 檔案，並讀取每個 mapper task 產生的 intermediate key-value pairs，並依照 partition function 分配給特定的 reducer。



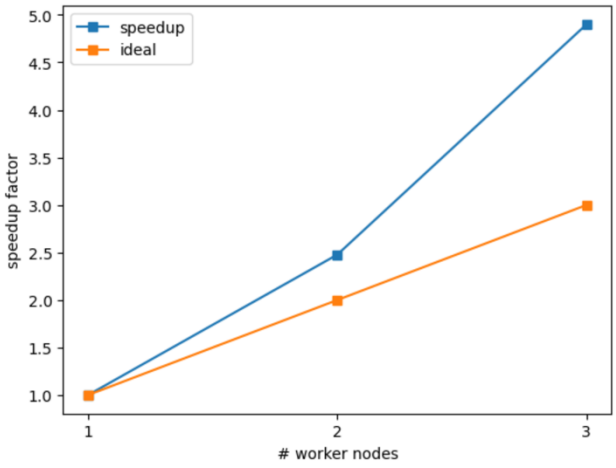
Experiment

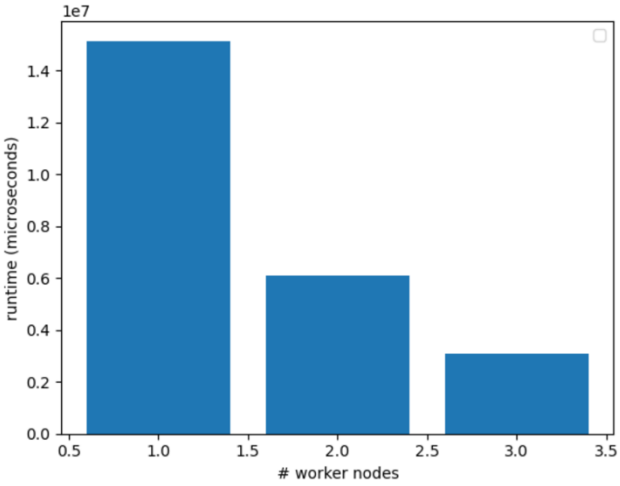
1. System spec: 學校的系統環境

   - 1 login node (apollo31) (200%CPU max)
   - 18 compute nodes (1200% CPU max)
   - Use squeue to view SLURM usage
   - Cluster monitor: http://apollo.cs.nthu.edu.tw/monitor
   - 48GB disk space per user

2. Speedup Factor & Time Profile

   我用的測試參數是 testcases/04.json，並調整 node 的數量。





因為有一個 node 固定要是 job tracker，因此 worker node 最大數量為 3。從 speedup factor 的圖表上可以看到，當 worker nodes 的數量增多的時候，speedup factor 也有隨之上升。且隨著 worker nodes 持續增加，speedup 的程度甚至超過 ideal 的線條，我推測是因為有考慮 data locality。因此可以了解到，程式的 scalability 非常好。我另外也把 time profile 給畫出來，可以看到 runtime 大幅度的下降。

conclusion

透過這一次的實驗，我除了更加了解 mapreduce 的整體架構之外，對於 MPI 與 pthread 的操作也更加熟悉。對於 data locality 對於程式的影響也有更加

深刻的認識。