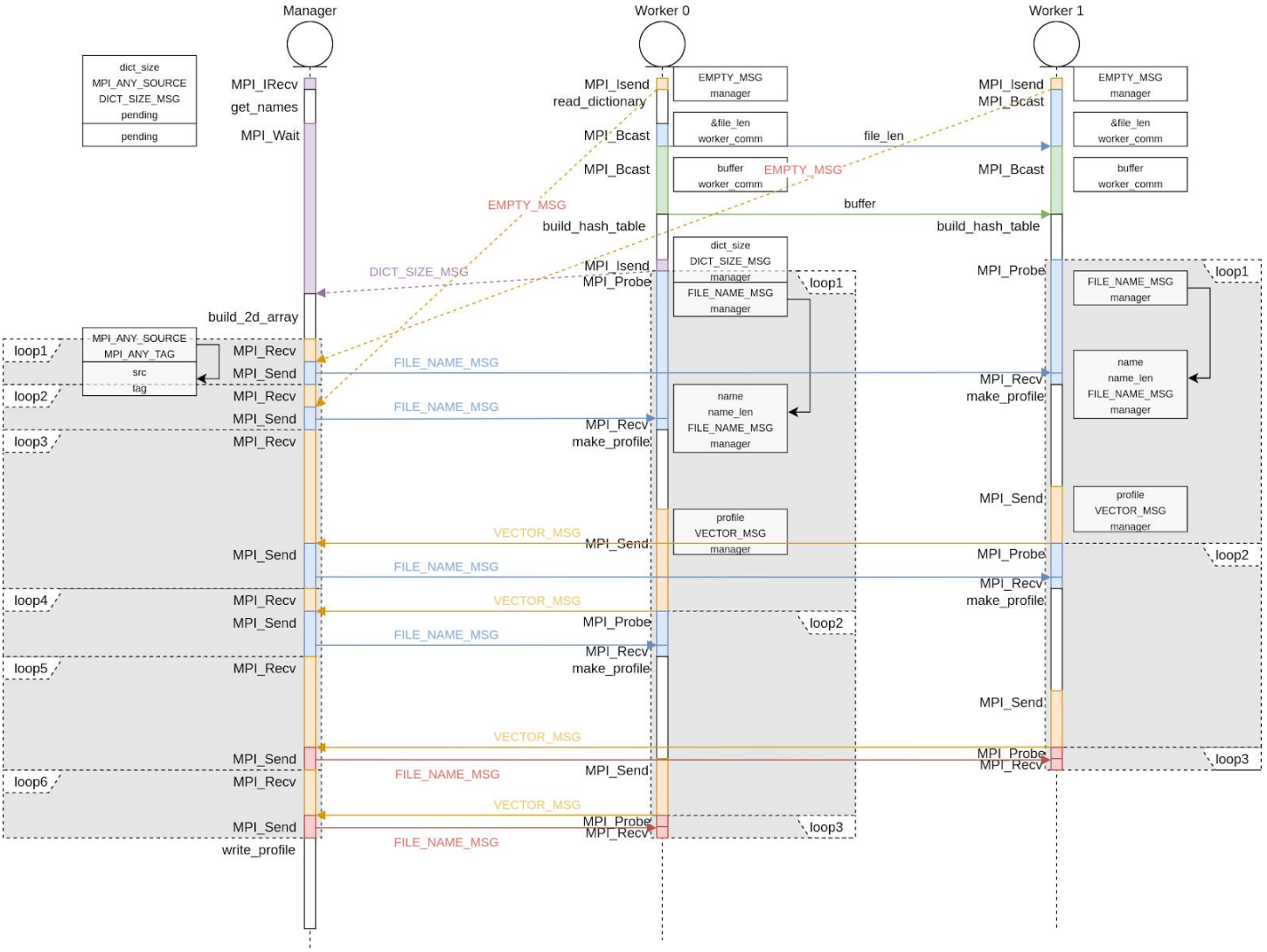
Parallel Programming Exercise 9

Author:	李子筠 (b06901145@ntu.edu.tw)
Student ID	B06901145
Department	Electrical Engineering

(If you and your team member contribute equally, you can use (co-first author), after each name.)

1 Sequence Diagram



2 Conclusion and Discussion

1. 使用 ISend 的原因

   MPI_Isend 是為了不等待傳訊息的時間，以及避免可能發生的 forever block、deadlock。

   A. 假設所有 buffer 都夠大

      在這個情況下，將所有的 MPI_Isend 換成 MPI_Send 不會有任何問題。就算沒有對應的 Recv 出現，只要能夠將資料 copy 到 send system buffer，MPI_Send 就能馬上 return，所以不會有 deadlock。（MPI_Bcast 只要等到 send system buffer 就可以 return，不需要對方執行到 MPI_Bcast，跟 MPI_Send 行為一樣）
      換成 MPI_Send 唯一的差別是原本 MPI_Isend 的地方會晚一點 return。

   B. 假設 manager 的 system buffer 不夠大

      如果 manager 的 system buffer 不夠大，就可能會發生 forever block。

      照原本 code 的順序，所有 worker 會用 MPI_Send 傳 EMPTY_MSG，就算 count 是 0，message 本身還是要包含 source 和 tag 的資訊，所以可能會將 manager 的 receive system buffer 佔滿。

      這時候 worker 本身只要將資料 copy 到自己的 send system buffer 就可以直接 return，不會被 block 住。因此可以順利執行到 MPI_Probe。

      不過這時候因為 manager 的 receive system buffer 是滿的，worker 0 send system buffer 的 dictionary size 無法傳給 manager，所以 manager 會被 MPI_Wait block 住。
      而所有的 worker 都會在 MPI_Probe block 住，因為 manager 沒有 MPI_Send filename 過去。

      這時候 manager 在等 dictionary size 從 worker 0 的 send system buffer 傳給自己的 buffer，但是只有在這次 wait 之後 manager 才會 MPI_Recv 去清空 receive system buffer。而 worker 在等 manager 更之後的 MPI_Send，被 block 在 MPI_Probe。

      主要是 manager 無法將自己的 send system buffer 清空，相當於在等永遠不會滿足的條件，所以導致 manager 自己和所有 worker 都被 block 住。

      不過這個情況可以藉由改變 Send 的時機來解決，只要在 worker 0 send dictionary size 之後再讓所有 worker 開始送 EMPTY_MSG 就可以確保 manager 一定會開始 MPI_Recv，去清空自己的 receive system buffer，也就不會發生 forever block。

      這個作法的缺點就是所有的 worker 都要等 worker 0，需要額外的 synchronization，比較沒有效率（而且這個情況可以把 MPI_Send 換成 MPI_ISend，不影響結果）。

   C. 假設 worker 的 system buffer 不夠大

      如果 worker 的 send system buffer 不夠大，就可能會發生 deadlock。

      如果 worker 的 send buffer 不夠大，在一開始的 MPI_Send message 後，EMPTY_MSG 將 send system buffer 佔滿，這時候 MPI_Bcast 就會被 block 住，如果 manager 的 system buffer 又不能裝下所有 worker 的 EMPTY_MSG，就會產生 deadlock。

      manager 在等 worker 0 送 dictionary size，worker 0 在等其他 worker 的 send buffer 清出空位，其他 worker 在等 manager 跑 MPI_Recv 讓自己可以清空 send buffer。因為有 circular wait 所以產生 deadlock。

      不過這個情況主要是因為 B. 假設的 manager buffer 不夠大產生的 deadlock。

2. 使用 MPI_Irecv 的原因

   MPI_Irecv 主要是為了讓 get_names 和 MPI 送訊息的時間可以重疊，兩件事情可以同時做。

   如果使用 MPI_Recv，就只是等到 worker 0 MPI_Isend，資料被傳到 manager receive system buffer 之後，才繼續往下做。

   這樣做的結果就是 manager get_names 要等到收完訊息才能做，整體的時間會拉長（計算時間+傳訊息的時間）。

3. 不會有 deadlock 的原因

   如果 manager buffer size 不夠大，而且 dictionary size 的那次 send 在 manager buffer 滿的時候送不出去。那樣的話就算用 MPI_Isend / MPI_Irecv 還是可能會產生上面 1.B. 的情況。

   除此之外的情況下，因為 Send / Recv 不需要等到對應的 Recv / Send 出現，所以 MPI_Irecv dictionary size 和 MPI_Isend EMPTY_MSG 可以交錯，因此不會有 deadlock。

   除非是使用 MPI_Ssend，Sender 會 block 直到 Receiver 收到訊息，否則無論是 MPI_Send 還是 MPI_Isend 都只會在 buffer 不夠大的情況下產生 forever block，不會有 deadlock 發生。

Appendix(optional):

(If something else you want to append in this file, like picture of life game)