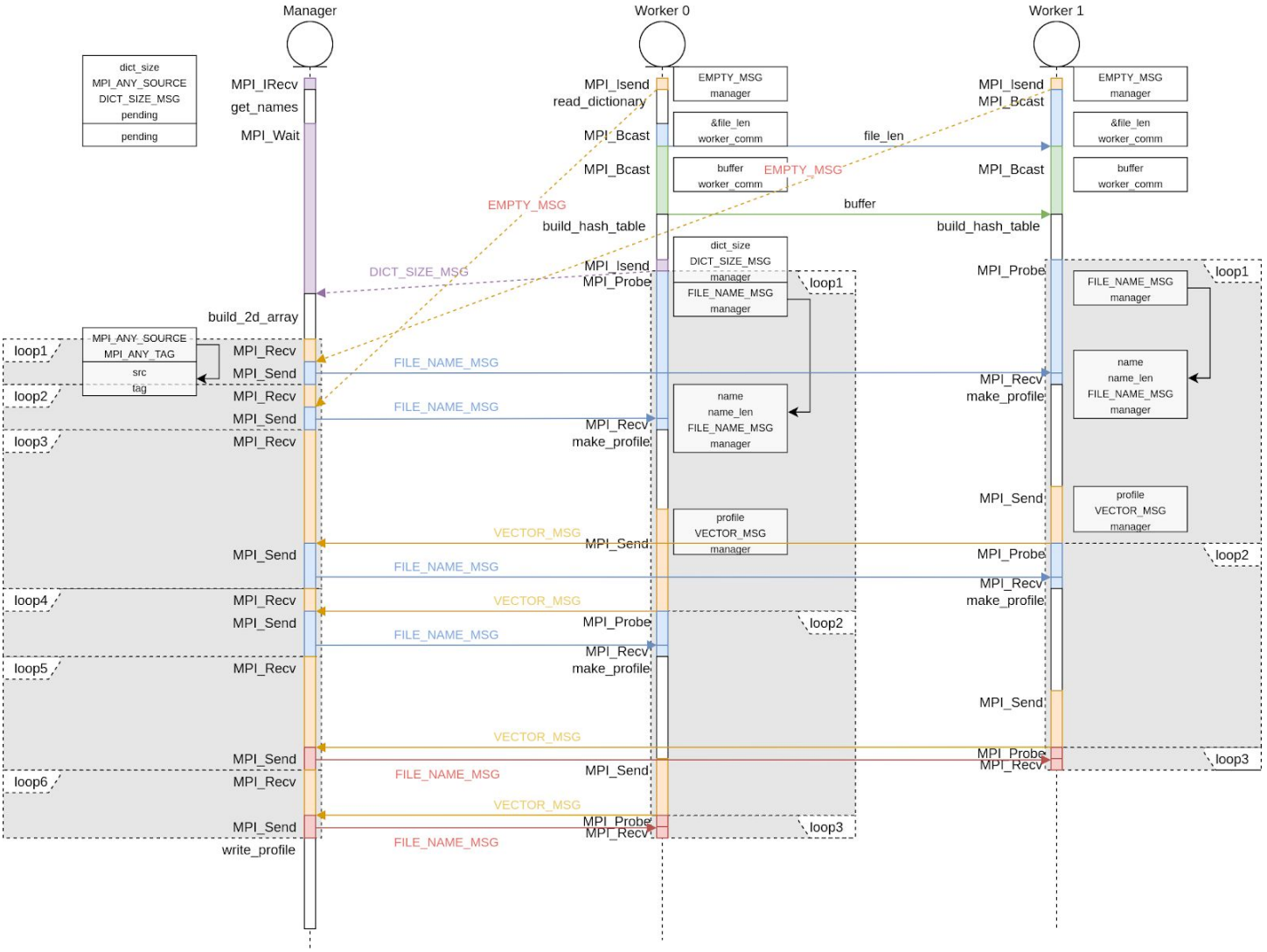
Parallel Programming Exercise 9

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



(為了簡潔，假設所有 blocking call (MPI_Probe, MPI_Wait)都在收到 message 之後馬上 return, 實際上可能在更晚才 return)

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 deadlock。

MPI 標準裡面對 MPI_Send 規範是 Implementation 可以決定要不要進行 buffer 。
將所有的 MPI_Isend 換成 MPI_Send 之後，可以分成兩種狀況：

A. 假設完全沒有 Buffer，或是 Buffer 不夠大

在這個情況下，MPI_Send 需要等到對應的 receive 出現，才能保證 Send 端的 user memory buffer 可以被修改。

這時候就會發生 deadlock：worker 0 的 MPI_Send 處於 block 的狀態，等待 manager 的 MPI_Recv。而 manger 的 MPI_Wait 也處於 block 的狀態，等待 worker 0 的 MPI_Send 傳 DICT_SIZE_MSG。

B. 假設 Send 端有 Buffer，而且 Buffer 夠大

在這種情況下，MPI_Send 只要等 message 被 copy 到 send 端的 send system buffer 就可以直接 return 。把所有的 MPI_Isend 換成 MPI_Send 不會有任何問題，worker 0 等到 MPI_Send 的 message 被存到 buffer 之後就可以繼續執行，最終會傳 DICT_SIZE_MSG 給 manager。

這個作法跟用 MPI_Isend 的差別就是要多等 copy 到 system buffer 的時間。

A program is "safe" if no message buffering is required for the program to complete.
One can replace all sends in such program with synchronous sends, and the program will still run correctly. (MPI Standard 3.1, page 44, line 1-3)

如果使用 MPI_Send 的話就要保證 Buffer 夠大才能正確執行，所以不能算是 "safe" 的程式。而使用 MPI_Isend 就沒有這個問題，無論 buffer 多大，都可以直接 return，繼續執行，跟上面 B. 的強況差不多，而且少了等待 copy 到 buffer (buffer mode) 或是傳送訊息 (synchronous mode) 的時間。

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 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，計算和傳訊息無法同時做，整體的時間會拉長（計算+傳訊息的時間）。

3. 不會有 deadlock 的原因

MPI_Isend 保證了不會有 deadlock 發生。

會產生 deadlock 主要是上面 1.A 的情況，worker 0 和 manager 的 cyclic waiting。如果用 MPI_Isend 就能保證 worker 0 不去等 manager，直接送出 DICT_SIZE_MSG，讓 deadlock 的條件無法滿足。

不過就算用 MPI_Isend 還是有可能發生 buffer size 不足的問題。如果發生在 receive 端，過多的 EMPTY_MSG 佔滿了 buffer (就算 count 是 0，message envelope 還是要紀錄 source 和 tag，還是需要一定的大小來儲存)，讓 manager 無法接到 DICT_SIZE_MSG 的話，manager 就會進入 forever blocking。因為 MPI_Recv 是在 MPI_Wait 之後，所以 buffer 裡面永遠不會清空。(也有可能是 MPI 先送出 buffer 不夠的錯誤)

> Quality implementations of MPI should ensure that this happens only in "pathological" cases. That is, an MPI implementation should be able to support a large number of pending nonblocking operations. (MPI Standard 3.1, page 47, line 31-33)

不過根據標準的說法，這種情況應該不太會發生。

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