Report Algoritmo Recursive Doubling Fault Tolerant

Luca Micarelli

April 2025

1 Recursive Doubling

- Ogni processo conserva informazioni sui processi attivi e inattivi
- Uso due comunicatori, uno globale e uno per gli attivi
- Tutti i processi entrano nella computazione della recursive doubling
- I processi attivi computano, gli altri osservano
- All inizio di ogni passo ho una barriera che viene usata per notificare i fault [comm globale]
- Ad ogni passo prima di fare sendrecv vedo se ho avuto un errore al passo precedente
- Se si chiamo errhandler:

    Controllo fault su comunicatore globale e lo aggiusto, ma mantengo il vecchio comunicatore globale [1.0]

    Se è fallito un rank inattivo sistemo la struttura inactive ranks e conto i rank inattivi che sono falliti [2.0]

    Se è fallito un rank attivo, tutti i nodi controllano chi è fallito nel comunicatore attivo [3.0]

        Se abbiamo abbastanza rank inattivi [3.1], mi calcolo:

            Chi deve svegliare inattivi e mandare dati ai rank corrotti [3.1.0]

            Chi è corrotto [3.1.1]

            Chi è inattivo e deve essere svegliato [3.1.2]

        Altrimenti se i rank inattivi sono insufficienti: [3.2]

            Mi calcolo la potenza di due minore più vicina [3.2.0]

            Scelgo i rank che saranno attivi [3.2.1]

        Creo nuovo comunicatore degli attivi [3.3]

        Aggiusto i dati se necessario mandando i dati corretti ai processi che hanno i dati corrotti [3.4]

    Aggiusto la struct in base ai nodi falliti [4.0]

    Calcolo se sono un nodo attivo o no [5.0]

- Continuo con la recursive doubling
- Alla fine gli inattivi aspettano che un attivo gli manda il risultato [6.0]
- Gli attivi mandano il risultato agli inattivi corrispettivi [7.0]
- Alla fine tutti hanno il risultato corretto

2 Note

1.0 Mantengo il vecchio comunicatore cha ha fallito, perché i rank nella struttura data hanno il valore relativo al vecchio comunicatore e viene aggiustato solo alla fine. Uso quindi il vecchio comunicatore per prendermi il gruppo dei rank quando creo il nuovo comunicatore degli attivi e per fare le send/recv tra i rank inattivi che devo svegliare e i rank attivi che devono svegliarli.

2.0 Abbiamo due step:

- 2.1 Check se un rank inattivo è fallito:
  - 2.1.0 Come primo punto controlliamo se esistono rank inattivi.
  - 2.1.1 Poi per ogni rank fallito cerco una corrispondenza nell'array degli inattivi; se la trovo ritorno 1, altrimenti ritorno 0.
- 2.2 Sistema la struttura inactive ranks: itero per ogni rank inattivo, e controllo se è fallito, se si incremento un contatore altrimenti riscrivo il rank nell'array facendolo shiftare alla posizione (i - counter)

3.0 Per ogni rank fallito cerco una corrispondenza nell'array degli attivi se la trovo ritorno 1 altrimenti ritorno 0

- 3.1 Check se numero di rank attivi falliti $\leq$ numero di rank inattivi sopravvissuti
  - 3.1.0 Abbiamo blocchi di distance / 2 ranks che condividono gli stessi dati, per ogni blocco, il primo rank che non è fallito e non ha i dati corrotti sarà il master del blocco ed è incaricato di svegliare gli inattivi, inviargli i dati e di inviare i dati ai rank corrotti (sempre all'interno dello stesso blocco). Dati due rank r1 e r2 se $r1/distance == r2/distance$, significa che siamo nello stesso blocco, prendo $distance/2$, perchè sto considerando i fallimenti del passo precedente. Per i rank corrotti devo assicurarmi che non siano falliti altrimenti invio i dati a un rank fallito che non li riceverà mai e rimango in attesa infinita
  - 3.1.1 Per ogni rank fallito r calcolo il corrispettivo corrotto corr = rank xor distance / 2, se corr == rank siamo noi i corrotti
  - 3.1.2 I processi inattivi da svegliare vengono presi dalla coda dei rank degli inattivi, per cui ogni processo inattivo, calcola la sua posizione nell'array, se la nostra posizione è $\geq$ (numero totale di processi inattivi - numero di processi attivi falliti) allora ci aspettiamo di essere svegliati da un processo attivo
- 3.2 Altrimenti se numero di nodi attivi falliti > numero di rank inattivi sopravvissuti:
  - 3.2.0 Riduco alla potenza minore di due più vicina $p' = 2^{\log_2 p}$, dove p è il numero attuale di processi attivi sopravvissutie e p' sarà il nuovo numero di processi attivi
  - 3.2.1 Calcolo il nuovo valore di distance d', dato che per passare da p a p' abbiamo ridotto la grandezza del comunicatore attivo di k, $d' = d/k$, avremo quindi adesso d blocchi da d' rank che condividono gli stessi dati e rifaremo il passo d'.
    Per costruire il nuovo array degli attivi, itero su ogni rank attivo, e controllo non sia fallito, se ho ancora spazio nel blocco lo inserisco tra gli attivi, altrimenti lo metto tra gli inattivi
- 3.3 Il vecchio comunicatore old comm ha i rank relativi alla struttura attuale data, uso quindi quel comunicatore per prendermi il gruppo di tutti i rank, poi creo il gruppo dei sopravvissuti a partire da active ranks che sarà un sottogruppo del gruppo originale e alla fine creo il nuovo comunicatore degli attivi
- 3.4 Se siamo precedentemente entrati nel passo 3.1 sarà necessario svegliare uno o più rank inattivi e inviare i dati sia a loro che ai rank che hanno dati corrotti, per cui in base alle flag calcolate al passo 3.1 chi deve svegliare e inviare dati farà le Send ai relativi ranks, mentre chi deve ricevere si metterà in attesa di ricevere i dati

4.0 Per ogni processo attivo itero sui processi falliti, se il rank è maggiore del rank del processo fallito incremento un contatore k, alla fine decremento il valore del rank di k, in sintesi devo calcolare il nuovo valore del rank nel nuovo comunicatore globale, decrementando ogni rank di 1 per ogni processo fallito che ha rank minore, poi eseguo lo stesso procedimento per i processi inattivi

5.0 Per calcolare se sono un processo attivo o inattivo nel nuovo comunicatore globale, mi prendo il rank relativo ad esso, poi vado a cercare se il mio rank è presente nell'array degli attivi, se si setto il valore di active a 1 ed esco, altrimenti alla fine avrò settato 0

6.0 Se sei un rank inattivo, mettiti in attesa di ricevere il risultato finale, settiamo il comunicatore globale come quello di riferimento per la chiamata e la flag MPI ANY SOURCE che ci permetterà di ricevere da qualunque nodo attivo ci invierà il risultato

7.0 Se sei attivo e il tuo rank i relativo al comunicatore degli attivi è minore del numero totale di rank inattivi allora farai la send al rank i-esimo della struttura degli inattivi

3 Edge Cases

- Errori diversi da processo fallito

  Quando intercetto un errore mi assicuro che l'errore sia l'intero 75, prima di gestirlo, se dovesse essere diverso, aborto tutto il comunicatore globale (Warning: 75 non è un errore di MPI Standard, ma indica MPIX ERRP ROC FAILED)

- Situazione in cui ho troppi fault per il passo in cui mi trovo

  All'i-esimo passo avremo distance = 2^i; ovvero i rank a blocchi di grandezza distance hanno gli stessi dati, per cui possiamo supportare failure a meno che non abbiamo tutti i rank di un blocco che sono falliti o corrotti.

- Failure esterni alla recursive doubling

  La fault tolerance viene gestita all'interno del body della recursive doubling, ma non è tollerata nè prima, nè dopo, nè all'interno dell'errhadler stesso, in tutti questi altri casi, semplicemente abortisco l'intero comunicatore

4 Generalizzazione per p non potenza di 2

- Calcolo p' minore potenza di 2 più vicina
- I rank $\geq$ p' si segnano come inattivi
- rank inattivi mandano dati ai corrispettivi attivi: i-esimo inattivo manda a i-esimo attivo, per come viene calcolato p' abbiamo che numero di rank inattivi < numero di rank attivi