

Message Passing con YARP

PREREQUISITI:

Conoscenza dei thread, dei semafori, delle porte e degli oggetti Bottle in YARP.

DATI:

Per ogni numero naturale n sia data la successione (detta anche traiettoria o volo di n):

$$a_0 = n, \quad n \in \mathbb{N}$$
$$a_i = \begin{cases} \frac{a_{i-1}}{2}, & \text{se } a_{i-1} \text{ pari} \\ 3 \cdot a_{i-1} + 1, & \text{altrimenti} \end{cases} \quad \forall i > 0$$

La congettura di Collatz (o $3n+1$) afferma che per qualsiasi valore iniziale di n il volo corrispondente si assesta sul ciclo limite 4, 2, 1, 4, 2, 1, ... o equivalentemente che la successione termina una volta raggiunto il valore 1.

La congettura è stata verificata per molti valori di n (fino a circa 2×10^{18}), ma rimane tutt'oggi indimostrata. Sebbene la comunità dei matematici, impegnata nella determinazione di una prova risolutiva, ritenga la congettura molto probabilmente vera, sarebbe sufficiente un solo controesempio per concludere circa la sua falsità: si giustificano quindi gli sforzi per cercare tale controesempio nell'insieme numerabile dei naturali facendo uso di programmi al calcolatore.

RICHIESTI:

Allo scopo di verificare progressivamente la congettura di Collatz, si progetti un'architettura distribuita di tipo server-client (Figura 1) basata sulla comunicazione a porte YARP.

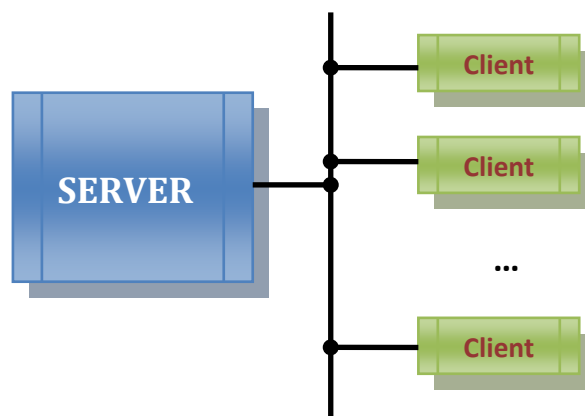


Figura 1. Architettura distribuita: i client eseguono verifiche concorrenti girando su un cluster di computer.

Parte Client

Il Client deve:

- Comunicare con il Server secondo il protocollo descritto in seguito. La comunicazione ha lo scopo di richiedere al Server un naturale n e una soglia t , per poi restituire l'esito della verifica.
- Verificare la coppia ricevuta (n,t) . La verifica si conclude positivamente nel caso in cui il volo di n ad un certo passo risulti inferiore o uguale a t . In caso negativo il loop di verifica non terminerà; dovrà dunque essere il Server a monitorare lo stato corrente delle richieste rimaste pendenti.

Parte Server

Il Server deve:

- Comunicare con i Client secondo il protocollo descritto in seguito.
- Mantenere e gestire una lista interna che raggruppi le richieste pendenti secondo la seguente strategia:
 1. Allo start-up s'inizializza un contatore a 0.
 2. Ad ogni richiesta s'incrementa il contatore inserendolo in coda alla lista e s'invia al client la risposta contenente la coppia $n=contatore$ e $t=head-1$, dove $head$ è l'elemento in testa alla lista.
 3. Se la richiesta del client contiene il risultato di una verifica, si rimuove dalla lista l'elemento corrispondente.
- Monitorare periodicamente il contenuto della lista allo scopo di identificare elementi a lungo permanenti, per i quali cioè non è stato ancora ricevuto dal client il risultato della verifica. Tali elementi risulterebbero essere possibili controesempi alla congettura.

Suggerimento: implementare il core del Server in una classe ereditata sia da `PortReader` (per la gestione delle richieste in entrata con la ridefinizione del metodo `read()`), sia da `RateThread` (per effettuare il monitoring periodico della lista).

Protocollo di comunicazione Server-Client

Formato del messaggio (Bottle) di richiesta Client => Server:

vocab_req	n
------------------	----------

Dove **vocab_req** è un opportuno identificatore Vocab, mentre **n** è il naturale su cui è positivamente terminata la verifica precedente, oppure è pari a 0 allo start-up.

Formato del messaggio (Bottle) di risposta Server => Client:

vocab_item	n	t
-------------------	----------	----------

Dove **vocab_item** è un opportuno identificatore Vocab, mentre **n** è il naturale su cui si richiede la verifica con soglia corrispondente **t**.