**MASTER-WORKER**

- Un manager thread M e molti worker thread Ws.

- M esegue il dispatching delle richieste provenienti dai client ai Ws.

- I Ws servono le richieste. Se il client C invia la richiesta x, W --esegue y = F(x) ed invia un messaggio di risposta contenente y a C.

- Il messaggio da M ai Ws può avere diversa natura:

- un'intera connessione di un client appena connesso viene fatta gestire da uno dei Ws.

- una singola richiesta di un client già connesso viene fatta gestire da uno dei Ws.

- I thread Ws:

- possono essere lanciati dinamicamente per ogni nuova richiesta di connessione ("un thread per connessione").

- possono far parte di un pool di thread predefinito ("pool di thread" vuol dire gruppo di thread già spawnati e pronti per servire richieste, al termine del servizio il thread ritorna).

- mix delle soluzioni precedenti.

- Il dispatching delle richieste può essere fatto attraverso una coda --concorrente condivisa da tutti i Ws.

Soluzione base

- "un thread per connessione", W serve tutte le richieste di un client.

- M esegue continuamente la accept() sul listen socket.

- Il descrittore ritornato dalla accept() viene passato come parametro ad --un W tipicamente spawnato in modalità *detached*.

- Pro:

- soluzione molto semplice da implementare.

- funziona bene per carichi non troppo elevati.

- Contro:

- overhead legato allo spawning dinamico dei thread.

- ci possono essere molti Ws inattivi.

Soluzione con thread pool

- Pool di worker sempre attivi.

- W gestisce una o più richieste di qualsiasi client. W non è associato --ad una specifica connessione.

- M ha il compito di:

- accettare nuove connessioni.

- controllare quali, tra i descrittori di connessioni già aperte, sono pronti in lettura utilizzando la select().

- M 🡪 Ws comunicano tramite una coda concorrente contenente i --descrittori pronti in lettura (cioè sui quali la read() non si blocca).

I descrittori inviati ai thread non vengono più ascoltati da M.

- Problema: quando la richiesta è stata servita da uno dei Ws del pool, --come viene detto a M di riascoltare nuovamente il descrittore?

- Ws 🡪 M comunicano tramite:

- una pipe il cui endpoint di lettura è registrato sulla select().

- la pipe contiene descrittori che potrebbero bloccare una read().

- la scrittura sulla pipe è atomica (non servono mutex).

- oppure variabili condivise.

- la select() deve gestire un timeout.

