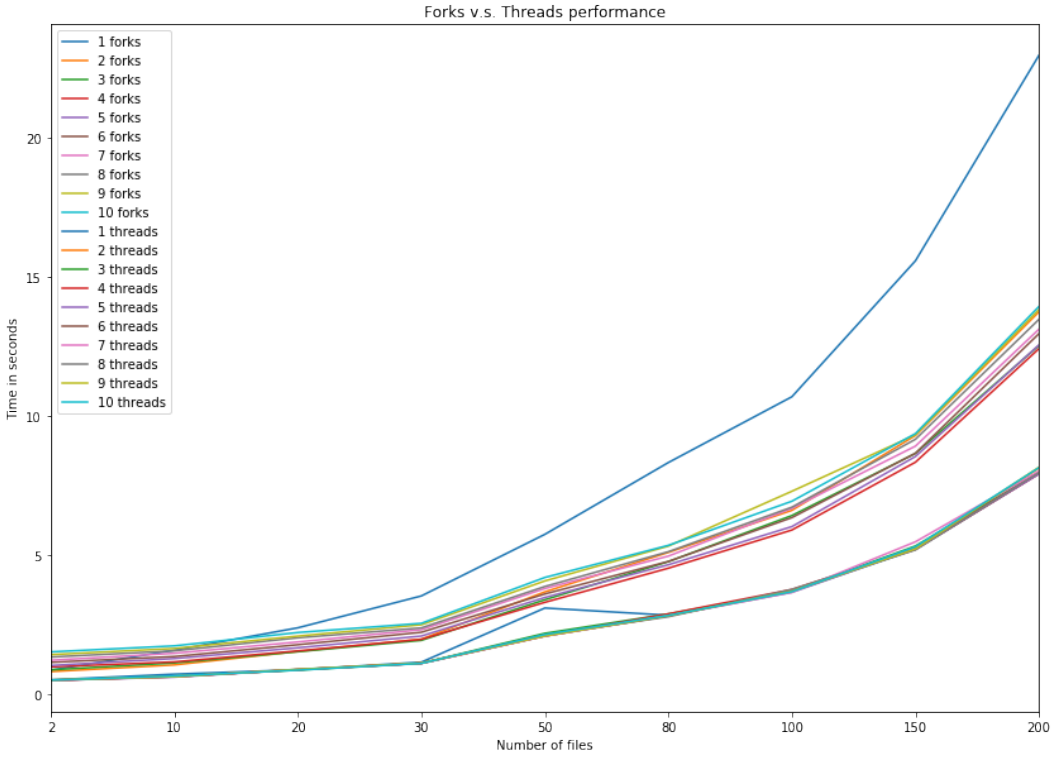
Sistemes Operatius 2 - Pràctica 5

Taras Yarema

3 de gener del 2020

Millora d'execució

Per a realitzar les proves, s'ha utilitzat el codi del fitxer `test.c` i les dades obtingudes de la pràctica anterior. El procés s'ha realitzat amb l'ajut de Jupyter Notebook.



Podem observar que, a diferència del cas amb *forks*, on el temps varia substancialment depenent del número, el cas *multi-threading* no varia tant. He fet proves amb un nombre més elevat de fils i els resultats no varien gaire igualment (veure `doc/bench_data_p5_3.csv`, tests fins a 500 fils).

Analitzant els resultats, sembla que utilitzar fils, en aquest cas, és la solució més òptima.

Comparació d'estratègies

La proposta **b** és millor donat que el fet de bloquejar tot l'arbre ralentitza molt l'actualització dels nodes.

No he fet proves comparatives, ja que comportaria dues estructures diferents de l'arbre i els nodes. Però és obvi que aturar l'actualització de l'arbre amb cada node és la opció més lenta.

Implementació

La implementació realitzada és una modificació de la **b**. En general, hi han dos moments que ha calgut sincronitzar: la lectura dels fitxer de la base de dades i l'actualització dels nodes de l'arbre. Per a això,

s'han creat dues estructures addicionals a `config.h`: `file` i `arguments`.

Estructura `file`

```
typedef struct _file {
    char name[MAX_CHARS];
    pthread_mutex_t mutex;
    int read;
} file;
```

Per a poder llegir els fitxers de forma correcta (un fil per fitxer, i només els fitxer que encara no han estat processats) he trobat pertinent implementar aquesta estructura. D'aquesta manera, cada fitxer de la base de dades disposa d'un monitor compartit per tots els fils i un *flag* que denota si ja ha estat processat o no.

Estructura `arguments`

```
typedef struct _args {
    rb_tree *tree;
    file *files;
    int num_files;
} arguments;
```

Serveix per a poder passar paràmetres a la funció `pthread_create` (`main.c` línies 149-163). D'aquesta manera la funció `process_list_files_mutex` (`process-threads.c` línia 139) que executarà cada fil secundari pot tenir accés a la informació del arbre i els fitxers de la base de dades a processar.

Altres modificacions

A part d'aquestes, òbviament, s'ha modificat `node_data` per a que tingui una variable *mutex* del tipus `pthread_mutex_t` (`red-black-tree.h` línies 26-36).

```
typedef struct node_data_ {
    char *key;
    int num_times;
    pthread_mutex_t mutex;
} node_data;
```