findeq: multithreaded search of files with equal data

Group C11

An overview of the program design

- **6 step of the program design**

  1. Input.
  2. Open & Read.
  3. Initialization
  4. Divide&Conquer
  5. Parallelism
  6. Intergration

An overview of the Parallelism

- **Multi threads**
- **Parallelism**
- **Signal**

Synchronizations of threads,

- **Key parts for implementing thread synchronization of code.**

  1. Mutex.

  - ‘mutex’ field in the ‘Task’ structure.

  - ‘mutex_queue’ mutex.

  2. Condition Variable.

  - ‘queue_cond’ condition variable.

Synchronizations of threads,

- **Key parts for implementing thread synchronization of code**

  3. Thread Create and Join

  - `thread_create()`

  - `thread_join()`

  4. Locking

Demonstrations of program executions,

Benchmarks

1. Various types of files
2. Detect redundant files in different directories
3. Show the results of the different options

Demonstrations of program executions,

Benchmarks

1. Various types of files
   a. regular file, link.
   b. .pdf, .zip, etc.
2. Detect redundant files in different directories
   a. temp/
   b. temp/c/
3. Show the results of the different options
   a. size of file
   b. output

Demonstrations of program executions,

Benchmarks

1. Various types of files
    a. regular file, link.
    b. .pdf, .zip, etc.
2. Detect redundant files in different directories
    a. temp/
    b. temp/c/
3. Show the results of the different options
    a. size of file
    b. output

Demonstrations of program executions,

Benchmarks

1. Various types of files
    a. regular file, link.
    b. .pdf, .zip, etc.
2. Detect redundant files in different directories
    a. temp/
    b. temp/c/
3. Show the results of the different options
    a. size of file
    b. output

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Speed
2. Accuracy
3. Proper Clean Up

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Speed
    a. 1 thread vs 64 threads, extreme case
    b. 1 thread vs 2 threads, efficiency
2. Accuracy
    a. 1 thread vs 64 threads, extreme case
    b. 1 thread vs 2 threads, efficiency
3. Proper Clean Up
    a. does the program end?
    b. are the results properly stored/printed?

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Speed
   a. 1 thread vs 64 threads, extreme case
   b. 1 thread vs 2 threads, efficiency

1 thread	64 threads
54.323640 seconds	*NA

- Metrics
  - minimum file size = 1024 bytes
  - run on the same directory

1 thread	2 threads
54.323640 seconds	33.216258 seconds

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Speed
    a. 1 thread vs 64 threads, extreme case
    b. optimal
2. Accuracy
    a. 1 thread vs 64 threads, extreme case
    b. 1 thread vs 2 threads, efficiency
3. Proper Clean Up
    a. does the program end?
    b. are the results properly stored/printed?

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Accuracy
   a. 1 thread vs 64 threads, extreme case
   b. 1 thread vs 2 threads, efficiency

1 thread	64 threads
13 redundant files	13 redundant files

- Metrics
  - minimum file size = 1024 bytes
  - run on the same directory

1 thread	2 threads
13 redundant files	13 redundant files

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Speed
    a. 1 thread vs 64 threads, extreme case
    b. 1 thread vs 2 threads, efficiency
2. Accuracy
    a. 1 thread vs 64 threads, extreme case
    b. 1 thread vs 2 threads, efficiency
3. Proper Clean Up
    a. does the program end?
    b. are the results properly stored/printed?

Analysis on how program performance changes depending on the number of used threads.

Benchmarks

1. Proper Clean Up
    a. does the program end?
    b. are the results properly stored/printed?

1 thread	64 threads
Yes	No
Yes	Yes

- Metrics
    - minimum file size = 1024 bytes
    - run on the same directory

Conclusion

- Simply increasing the # of threads doesn’t necessarily show an increase in **performance**.
- Could easily view memory wasted for redundant files.

Limitations

- Implement a way to end threads endlessly waiting for a task.

Future Work

- Show the total amount of memory being wasted.
- More efficient algorithms & data structures.