**Parallel / Distributed computing – Assignment 2 document**

컴퓨터공학과 20151619 최준수

**Experiment #1 – mixture of a large file and a small file**

Test files:

1. NAVER.html (Naver homepage) : 205,293 bytes
2. test.txt (custom dataset): 143 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| **# of threads** | **# of requests** | **Total bytes received** | **Execution Time** |
| 2 | 2 | 410948 | 0.020403s |
| 2 | 10 | 2054740 | 0.077249s |
| 10 | 2 | 2054740 | 0.053196s |
| 2 | 50 | 10273700 | 0.355284s |
| 10 | 10 | 10273700 | 0.250041s |

**Observations:**

I have mixed a large-sized file with a small-sized custom file and see how server performed. client program creates threads using <pthread.h> library with the number of threads inserted through command line argument specified in readme file. Then each thread repeatedly send requests to the server with the number of requests inserted through command line. Threads alternate the requested file listed in filelist.txt.

You can see that when requesting same bytes, using more threads generally takes less time than sending more requests from less threads.

**Experiment #2 – only small file**

Test files:

1. test.txt (custom dataset): 143 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| **# of threads** | **# of requests** | **Total bytes received** | **Execution Time** |
| 2 | 2 | 648 | 0.001668s |
| 2 | 10 | 3240 | 0.004669s |
| 10 | 2 | 3240 | 0.007504s |
| 2 | 50 | 16200 | 0.019529s |
| 10 | 10 | 16200 | 0.026256s |

**Observations:**

You can see different result from Experiment #1. In Experiment #2, you can see that using less threads took less time than using more threads. This is because file that the client requested was a small-sized file and overhead of creating and deleting threads was bigger than time spent to actually receive the file. Thus, using 2 threads and repeatedly sending requests from the same thread took less time.

**Experiment #3 – only large file**

Test files:

1. so.html (Stack overflow homepage): 303,179 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| **# of threads** | **# of requests** | **Total bytes received** | **Execution Time** |
| 2 | 2 | 1212792 | 0.051894s |
| 2 | 10 | 6063960 | 0.229203s |
| 10 | 2 | 6063960 | 0.151357s |
| 2 | 50 | 30319800 | 1.154361s |
| 10 | 10 | 30319800 | 0.681619s |

**Observations:**

However in Experiment #3, I used the biggest html file I downloaded from Stack Overflow. Unlike Experiment #2, file was big enough to benefit from using multiple threads. Thus, using more threads is beneficial and you can see drastic difference in last 2 results. Approximately 30MB was received from the server and using 10 threads and dividing the load by the factor of 5 resulted in almost half the time to complete its job.

We can thereby conclude that using more threads is not always beneficial but it is generally better unless the files are too small.

**Experiment #4 – repeatedly request 1 file**

Test files:

1. Daum.html (Daum homepage): 231,616 bytes

|  |  |  |  |
| --- | --- | --- | --- |
| **# of threads** | **# of requests** | **Epoch** | **Execution Time** |
| 1 | 1 | 1 | 0.010955s |
| 1 | 5 | 1 | 0.011314s |
|  |  | 2 | 0.009034s |
|  |  | 3 | 0.008525s |
|  |  | 4 | 0.008448s |
|  |  | 5 | 0.008552s |
| 1 | 10 | 1 | 0.011269s |
|  |  | 5 | 0.008392s |
|  |  | 10 | 0.008538s |

**Observations:**

Experiment #4 was done to see the caching effect. I have repeatedly requested Daum.html from a single thread. As you can see, it normally takes around 0.01second to receive Daum.html. However, as you repeat the request, server caches the requested file and execution time decreases until 0.0084seconds. From 1 thread 10 requests 10 epoch, we can see that execution time is no different from 5 epoch.

From the experiment I have found that caching decreased the execution time by around 26%. This is quite a big performance boost.