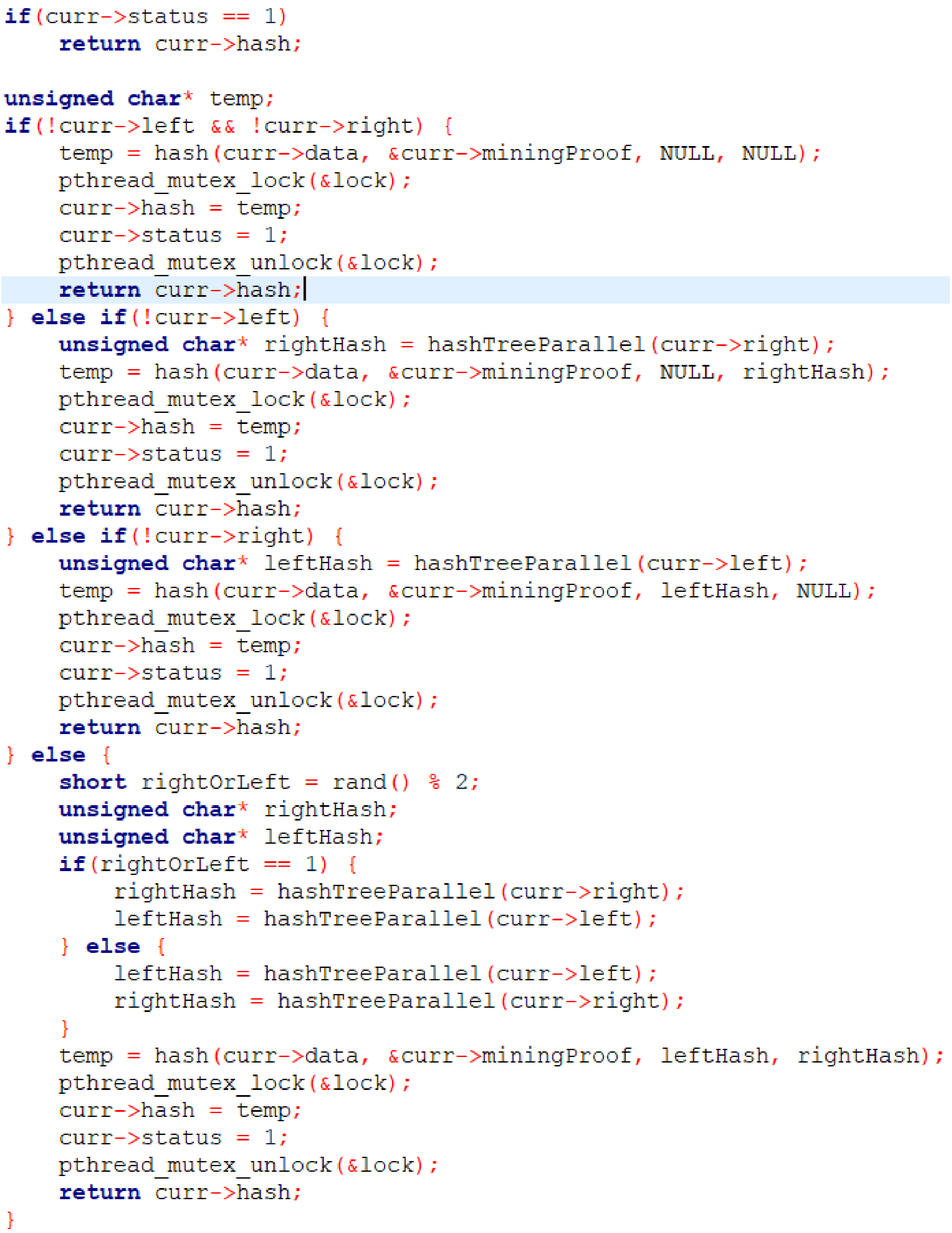
Parallel Progrmg: Sci & Engrg

Machine Problem 4

yhyuan2

A. Q: Post the snippet of the critical section of your code for hashTreeParallel and briefly explain the logic behind it. You need to show that your threads are being utilized to its maximum capacity.

   A:



```c
if(curr->status == 1)
    return curr->hash;

unsigned char* temp;
if(!curr->left && !curr->right) {
    temp = hash(curr->data, &curr->miningProof, NULL, NULL);
    pthread_mutex_lock(&lock);
    curr->hash = temp;
    curr->status = 1;
    pthread_mutex_unlock(&lock);
    return curr->hash;
} else if(!curr->left) {
    unsigned char* rightHash = hashTreeParallel(curr->right);
    temp = hash(curr->data, &curr->miningProof, NULL, rightHash);
    pthread_mutex_lock(&lock);
    curr->hash = temp;
    curr->status = 1;
    pthread_mutex_unlock(&lock);
    return curr->hash;
} else if(!curr->right) {
    unsigned char* leftHash = hashTreeParallel(curr->left);
    temp = hash(curr->data, &curr->miningProof, leftHash, NULL);
    pthread_mutex_lock(&lock);
    curr->hash = temp;
    curr->status = 1;
    pthread_mutex_unlock(&lock);
    return curr->hash;
} else {
    short rightOrLeft = rand() % 2;
    unsigned char* rightHash;
    unsigned char* leftHash;
    if(rightOrLeft == 1) {
        rightHash = hashTreeParallel(curr->right);
        leftHash = hashTreeParallel(curr->left);
    } else {
        leftHash = hashTreeParallel(curr->left);
        rightHash = hashTreeParallel(curr->right);
    }
    temp = hash(curr->data, &curr->miningProof, leftHash, rightHash);
    pthread_mutex_lock(&lock);
    curr->hash = temp;
    curr->status = 1;
    pthread_mutex_unlock(&lock);
    return curr->hash;
}
```

Every thread will check whether hash of the current node is calculated. If it is calculated, thread only returns hash of that node. If it’s not, thread will check how many children the node has and use DFS to recursively traverse the tree. If the node has two children, thread will randomly choose which child is the next node to search.

B. Q: Provide the running time for the following tree sizes for both naive and multi-threaded

a. 2056 nodes

Type	Single	-t 2	-t 4
Time	50.874461s	26.685822s	16.436230s

b. 6168 nodes

Type	Single	-t 4	-t 8
Time	152.613917s	43.790480s	25.582452s

c. 10240 nodes

Type	Single	-t 8	-t 12
Time	253.228094s	38.384650s	26.416489s

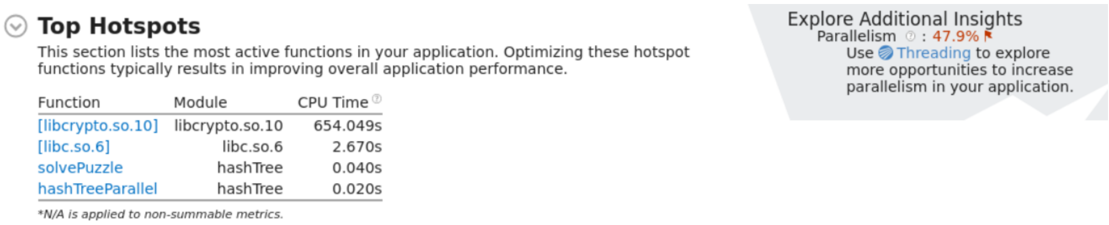
d. 20480 nodes

Type	Single	-t 12	-t 16
Time	506.386670s	53.388536s	55.459922s

C. Q: For Vtune, what contributes towards the spin time and what can cause your program to have poor or less than optimal CPU cores utilization?

A: Because there is mutex lock in the code. When thread is waiting for lock, it will cause spin time. If there is too much synchronization which causes spin time, this will make program have poor CPU utilization.

D. Vtune Analysis



Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

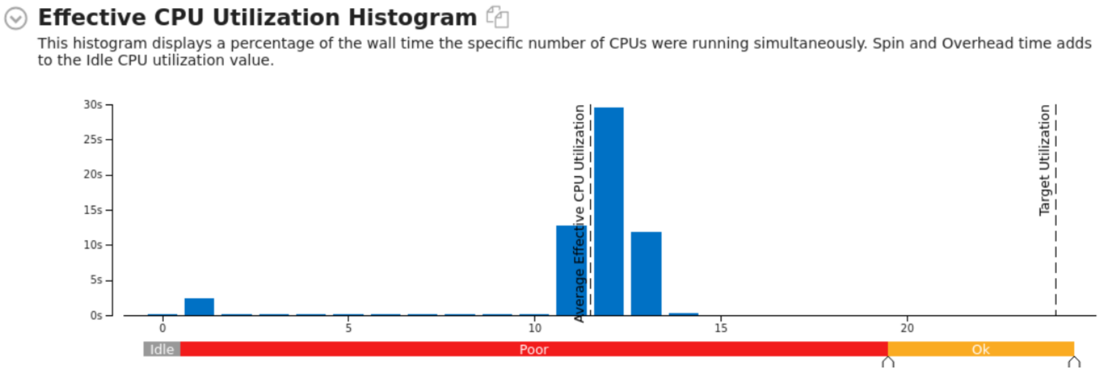
Function	Module	CPU Time
[libcrypto.so.10]	libcrypto.so.10	654.049s
[libc.so.6]	libc.so.6	2.670s
solvePuzzle	hashTree	0.040s
hashTreeParallel	hashTree	0.020s

*N/A is applied to non-summable metrics.

Explore Additional Insights

Parallelism : 47.9%

Use Threading to explore more opportunities to increase parallelism in your application.



Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.