CS406/CS531 Final Project

Utku Alkan
Ekin Nalbantoğlu

Algorithm 5 ParSpaRyser $(mat, cptrs, rows, cvals, dim, start, end)$

1: $p \leftarrow 1$
2: **for** $i = 1 \dots N$ **do**
3: $\quad rowSum \leftarrow 0$
4: $\quad$ **for** $j = 1 \dots N$ **do**
5: $\quad\quad rowSum \leftarrow rowSum + mat[i,j]$
6: $\quad x[i] \leftarrow mat[i,n] - rowSum/2$
7: $\quad p \leftarrow p \times x[i]$
8: **for** *each thread* **do**
9: $\quad myX \leftarrow x$
10: $\quad myP \leftarrow 0$
11: $\quad myStart \leftarrow start + \texttt{threadId} \times chunkSize$
12: $\quad myEnd \leftarrow min(start + (\texttt{threadId} + 1) \times chunkSize, end)$
13: $\quad$ calculate myX using $\texttt{GrayCode}_{myStart-1}$
14: $\quad prod \leftarrow \prod_{n=1}^{\texttt{dim}} myX[i]$ for $myX[i] \neq 0$
15: $\quad zeroNum \leftarrow \sum_{n=1}^{length(myX)} myX[i]$ for $myX[i] = 0$
16: $\quad$ **for** $g = myStart \dots myEnd - 1$ **do**
17: $\quad\quad j \leftarrow \log_2(\texttt{GrayCode}_g \oplus \texttt{GrayCode}_{g-1}) + 1$
18: $\quad\quad s \leftarrow 2 * \texttt{GrayCode}_g[j] - 1$
19: $\quad\quad$ **for** $i = cptrs[j] \dots cptrs[j]$ **do**
20: $\quad\quad\quad$ **if** $myX[rows[i]] == 0$ **then**
21: $\quad\quad\quad\quad zeroNum \leftarrow zeroNum - 1$
22: $\quad\quad\quad\quad myX[rows[i]] \leftarrow myX[rows[i]] + s \times cvals[i]$
23: $\quad\quad\quad\quad prod \leftarrow prod \times myX[rows[i]]$
24: $\quad\quad\quad$ **else**
25: $\quad\quad\quad\quad prod \leftarrow prod / myX[rows[i]]$
26: $\quad\quad\quad\quad myX[rows[i]] \leftarrow myX[rows[i]] + s \times cvals[i]$
27: $\quad\quad\quad\quad$ **if** $myX[rows[i]] == 0$ **then**
28: $\quad\quad\quad\quad\quad zeroNum \leftarrow zeroNum + 1$
29: $\quad\quad\quad\quad$ **else**
30: $\quad\quad\quad\quad\quad prod \leftarrow prod \times myX[rows[i]]$
31: $\quad\quad$ **if** $zeroNum == 0$ **then**
32: $\quad\quad\quad myP \leftarrow myP + (-1)^g * prod$
33: $\quad AtomicAdd(p, myP)$
34: **return** $p \times (4 \times (n \bmod 2) - 2)$

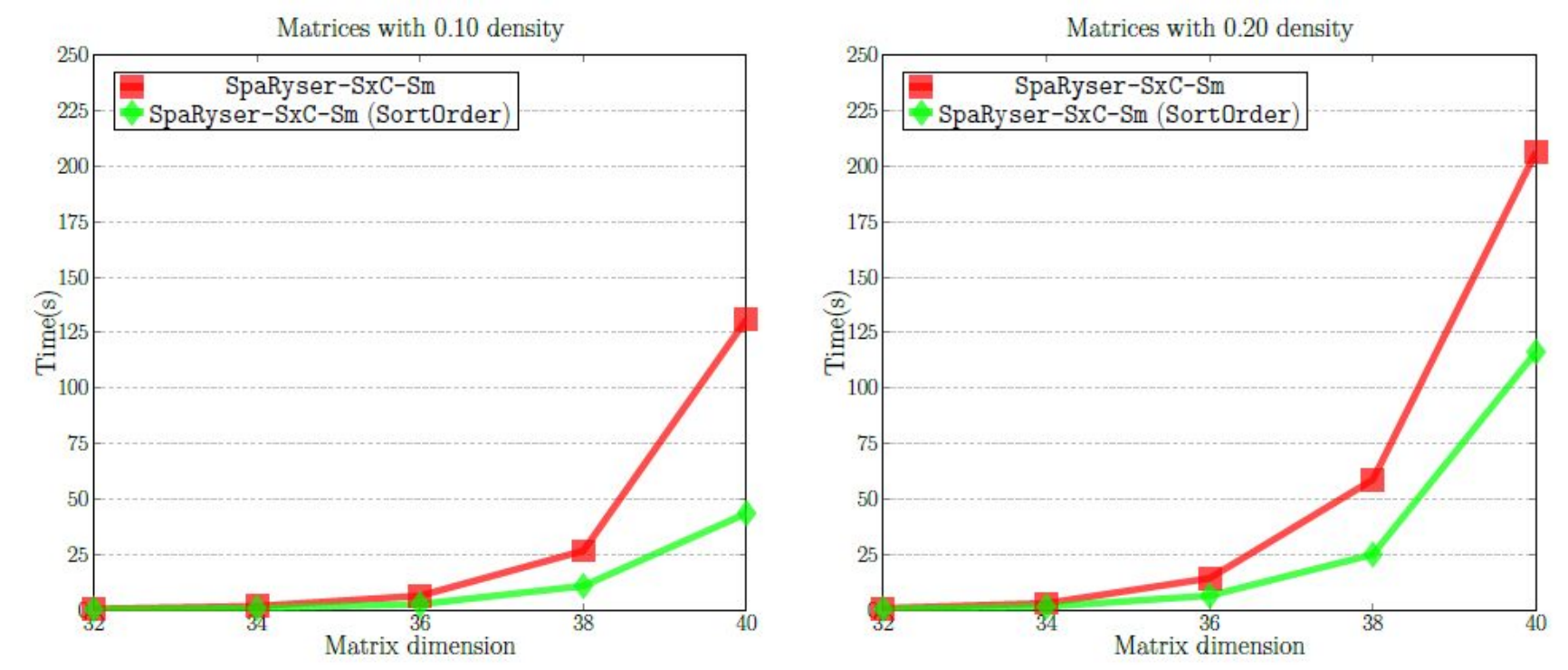
COMPUTING MATRIX PERMANENTS AND COUNTING PERFECT MATCHINGS ON GPUS

by
B. Yağlıoğlu

Algorithm 5 ParSpaRyser: Ryser algorithm to calculate permanents of sparse matrices with a parallelizable block.

Preprocessing

```
sortMatrixColumnsByNonZeros(matrix, n);
```



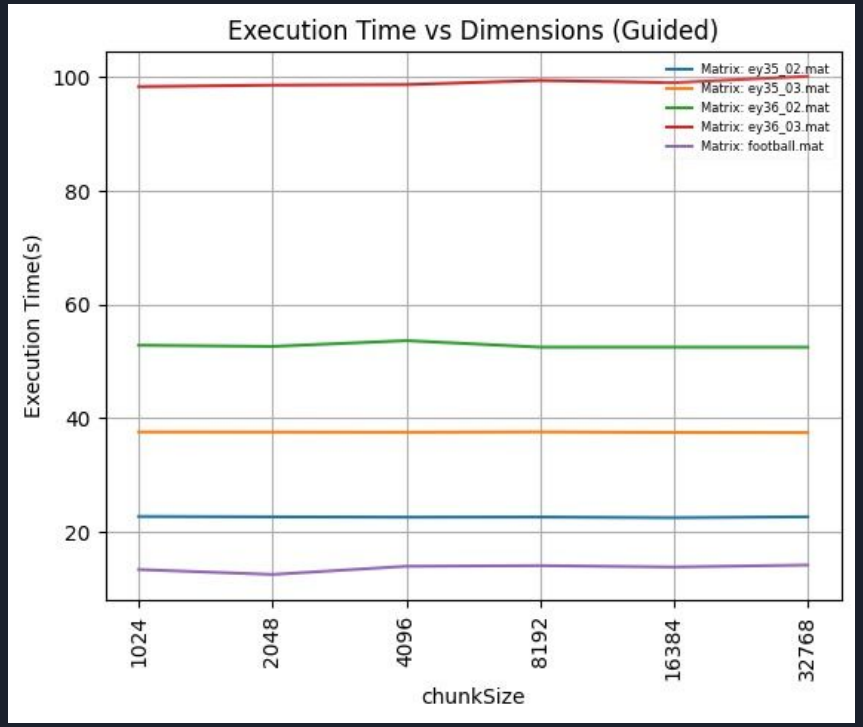
Integration of dynamic scheduling by dividing the load into chunks

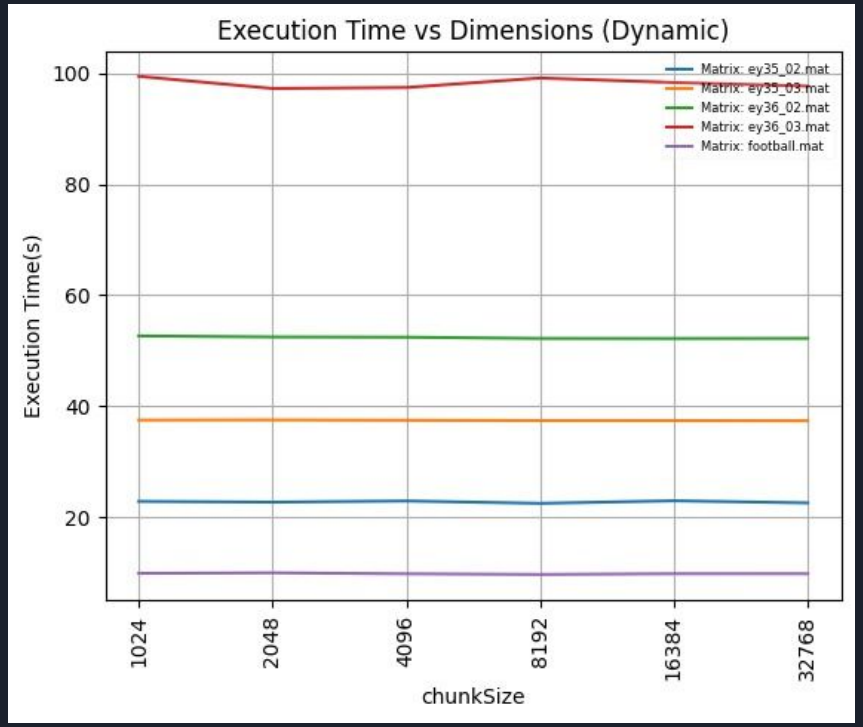
```
502     long long chunkSize = (end - start) / numChunks + 1;
503
504     #pragma omp parallel num_threads(numThreads)
505     {
506         double myX[N];
507         #pragma omp for schedule(dynamic, 1)
508         for (int chunkID = 0; chunkID < numChunks; chunkID++)
509         {
510
511             for (int i = 0; i < N; i++) {
```

Determining the chunk size

```
numChunks = 8192;

start = omp_get_wtime();
double perm_spaRyser = ParSpaRyserD(ccs.cptrs,
end = omp_get_wtime();
cout << perm_spaRyser << "\t"<<end-start<<endl;
```





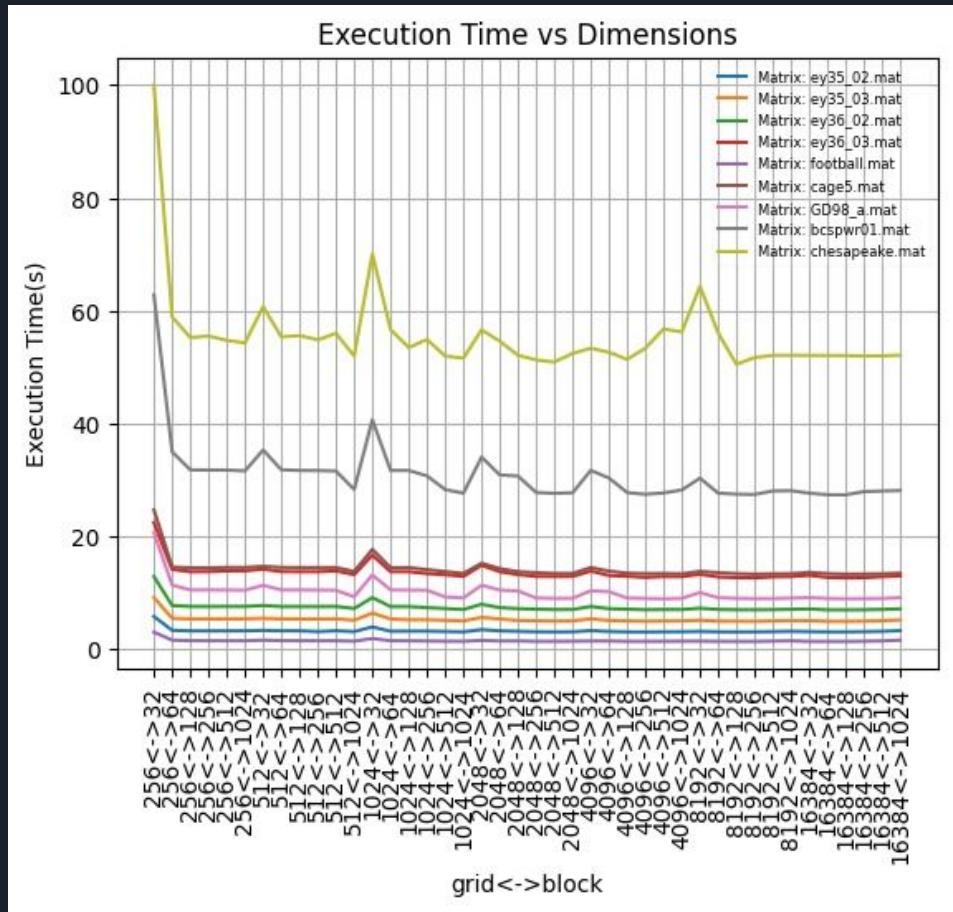
CUDA implementation kernel call for 1 gpu

```
558     int gridsize = 16384;
559     int blocksize = 256;
```

```
long long int start = 1;
long long int end = (1LL << (N-1));
int numThreads = numBlocks * blockSize;
long long int chunkSize = (end - start) / numThreads + 1;

ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs, d_rows,
d_cvals, N, d_p, d_x, 1, (1LL << (N-1)), chunkSize);

cudaDeviceSynchronize();
```



2 and 4 GPUs

```
if(gpuCounter == 0){
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs, d_rows,
    d_cvals, N, d_p, d_x, start, (end/2), chunkSize);
}else{
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs, d_rows,
    d_cvals, N, d_p, d_x, end/2, end, chunkSize);
}
```

```
double startTime = omp_get_wtime();
if(gpuCounter == 0){
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs,
    d_rows, d_cvals, N, d_p, d_x, start, 3*end/8, chunkSize);
}else if(gpuCounter==1){
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs,
    d_rows, d_cvals, N, d_p, d_x, 3*end/8, 6*end/8, chunkSize);
}else if(gpuCounter==2){
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs,
    d_rows, d_cvals, N, d_p, d_x, 6*end/8, 7*end/8, chunkSize);
}else if(gpuCounter==3){
    ParSpaRyserCudaKernel<<<numBlocks, blockSize>>>(d_cptrs,
    d_rows, d_cvals, N, d_p, d_x, 7*end/8, end, chunkSize);
}
```

```
cs406.ekinn@nebula:~/proj/twoGPU$ make
nvcc -Xcompiler -fopenmp -o gpu4 parSpaRyserFourGPU.
cu
./gpu4 chesapeake.mat
GPU ID: 3 synchronizes in 16.781465 seconds
GPU ID: 2 synchronizes in 17.602388 seconds
GPU ID: 1 synchronizes in 19.010036 seconds
GPU ID: 0 synchronizes in 19.458299 seconds
spaRyser Result: 1.31735e+13
20.318988 seconds
```

Thank you

Kaya, K. (2019). Parallel algorithms for computing sparse matrix permanents. Turkish Journal of Electrical Engineering and Computer Sciences, 27, 4284–4297.

Yağlıoğlu, B. (2021). Computing matrix permanents and counting perfect matchings on gpus [Master's thesis]. https://research.sabanciuniv.edu/id/eprint/42490/1/10364196.pdf