Proof of Work

GPU CUDA optimization

Pedro Abreu, 93240
João Gameiro, 93097
Grupo 2 - Turma 2

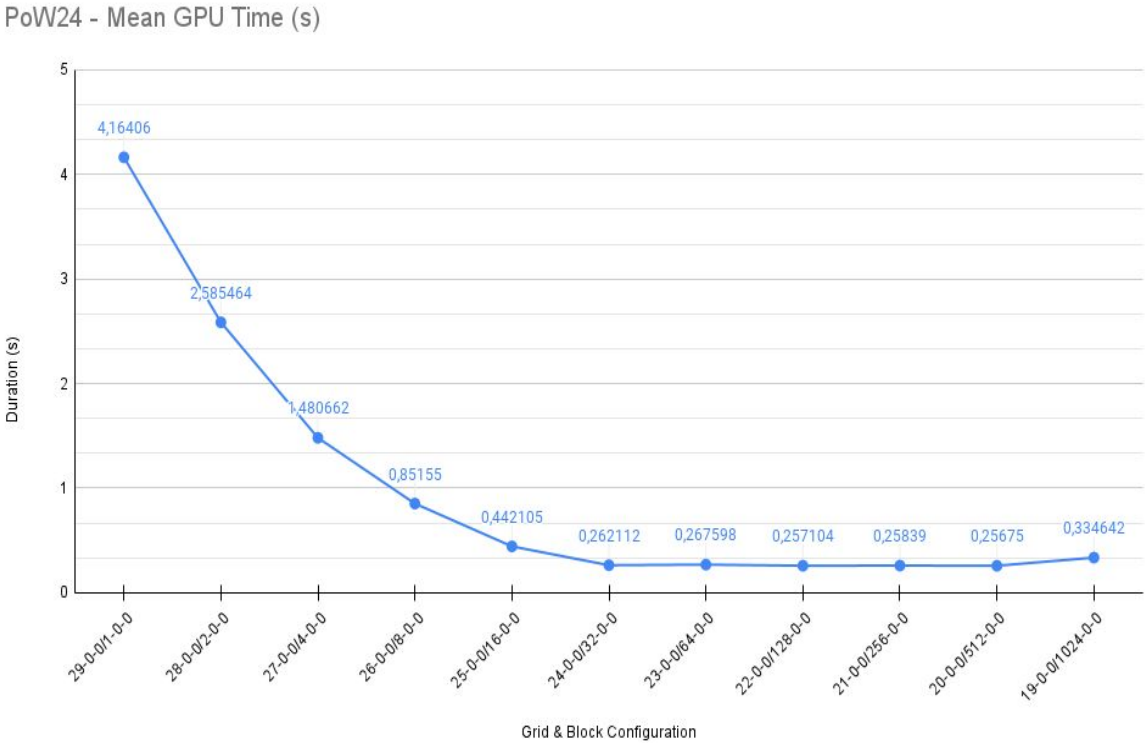
universidade
de aveiro

Assignment 2 - Arquiteturas de Alto Desempenho
prof. António Rui Borges
jan. 2022

PoW24 - Optimization of launch grid

- Best results are achieved for blockDim.x >= 32
- Best Value was block of 512 threads
- Performance starts to deteriorate for block of 1024 threads

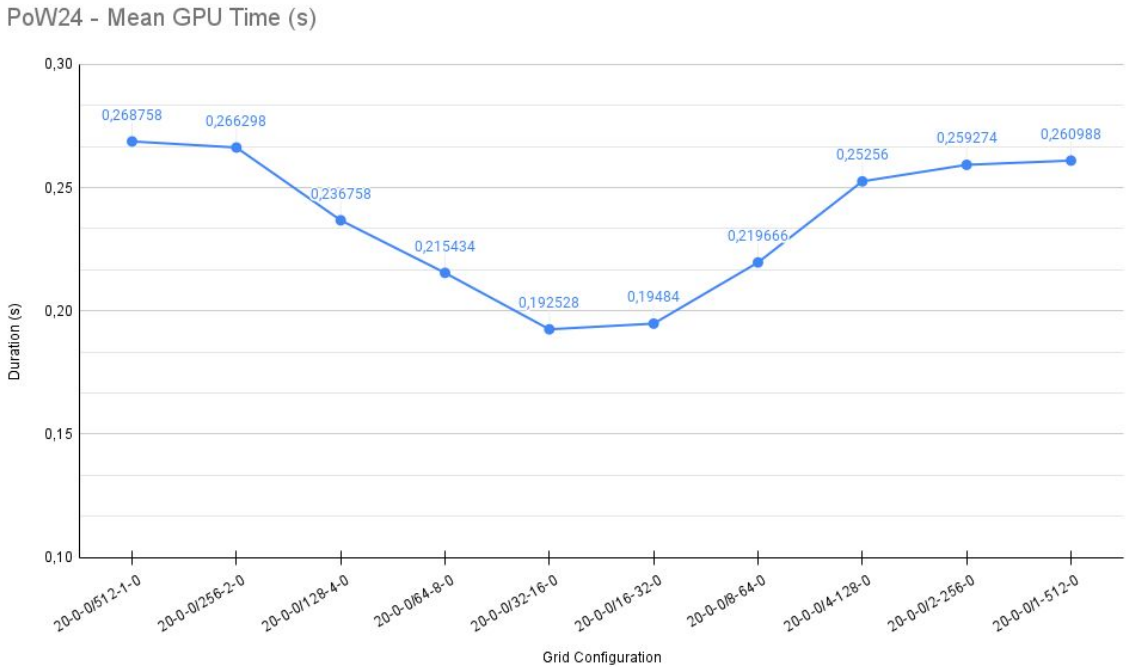
- Best values for performance occur when block size is multiple of 32 (Warp Size)



PoW24 - Optimization of launch grid

- After setting number of threads per block it was necessary to optimize block organization

- Best Value was block of blockDim.x=32 & blockDim.y=16



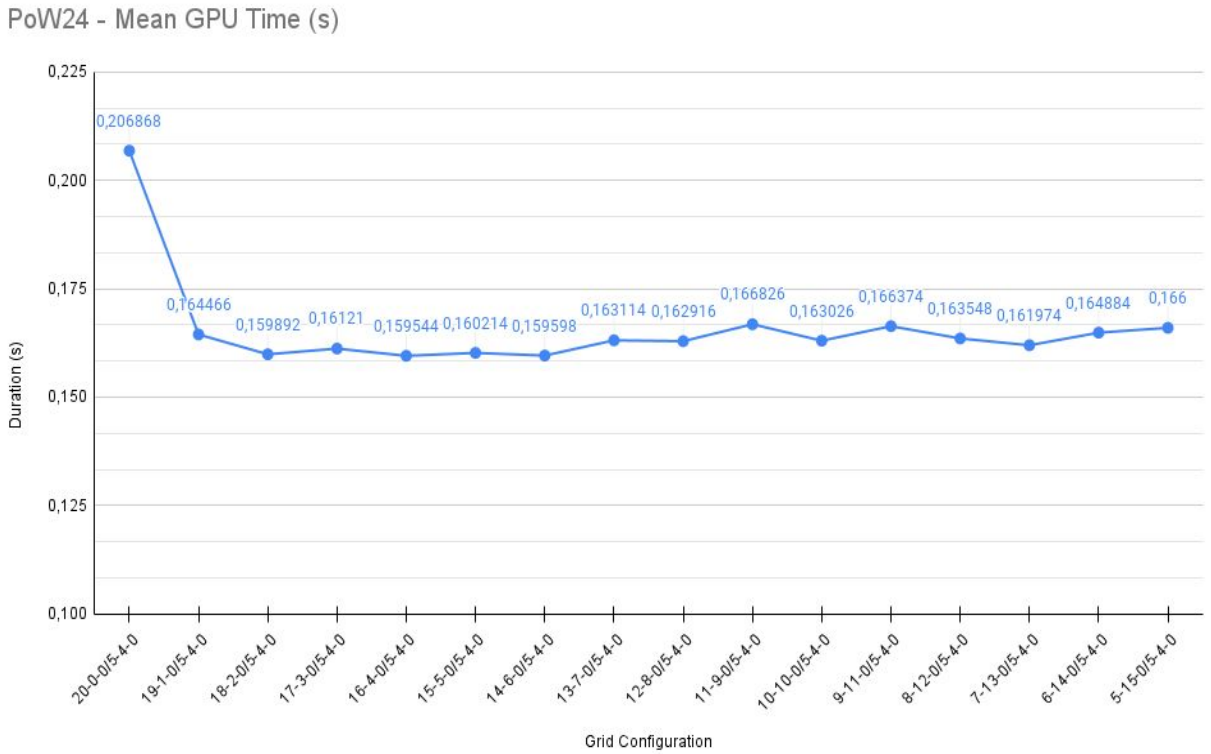

```
unsigned int x = (unsigned int)threadIdx.x + (unsigned int)blockDim.x * (unsigned int)blockIdx.x;
unsigned int y = (unsigned int)threadIdx.y + (unsigned int)blockDim.y * (unsigned int)blockIdx.y;
idx = y + (unsigned int)blockDim.y * (unsigned int)gridDim.y * x;

tokens += 16u * (idx % 32u);
```

PoW24 - Optimization of launch grid

- Finally all that was left was to optimize the gridDim.x and gridDim.y
- Performance is better for gridDim.y > 1

- Best Value was block of gridDim.x=16 & gridDim.y=4
- So the optimized launch grid was
  - {16u, 4u, 5u, 4u}

- speedUp obtained = 985.7018
- Best time was 0.159544 s

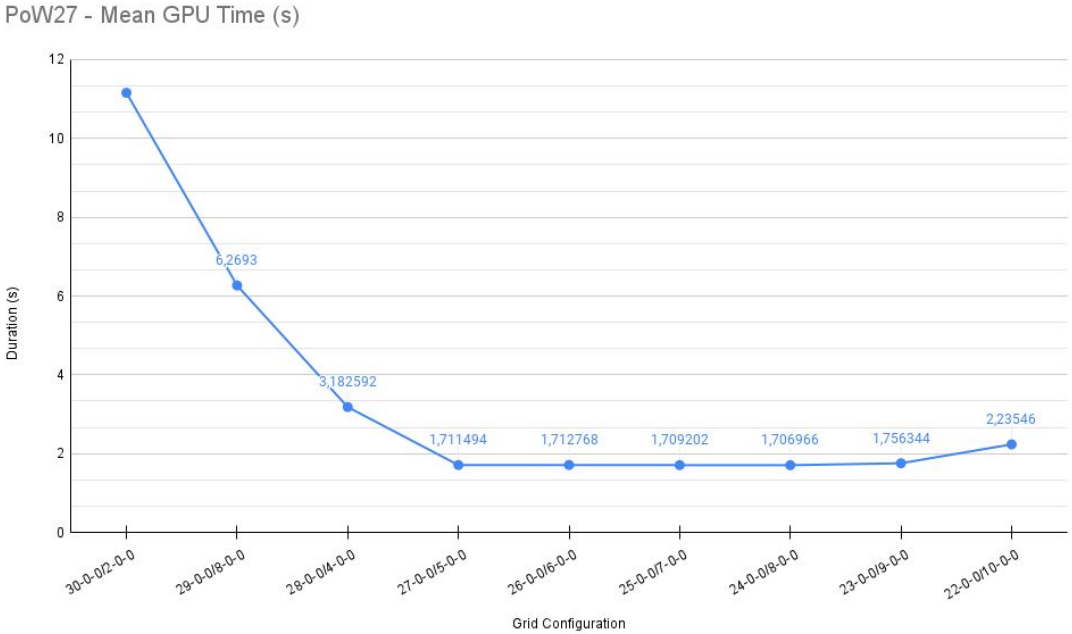


PoW27 - Optimization of launch grid

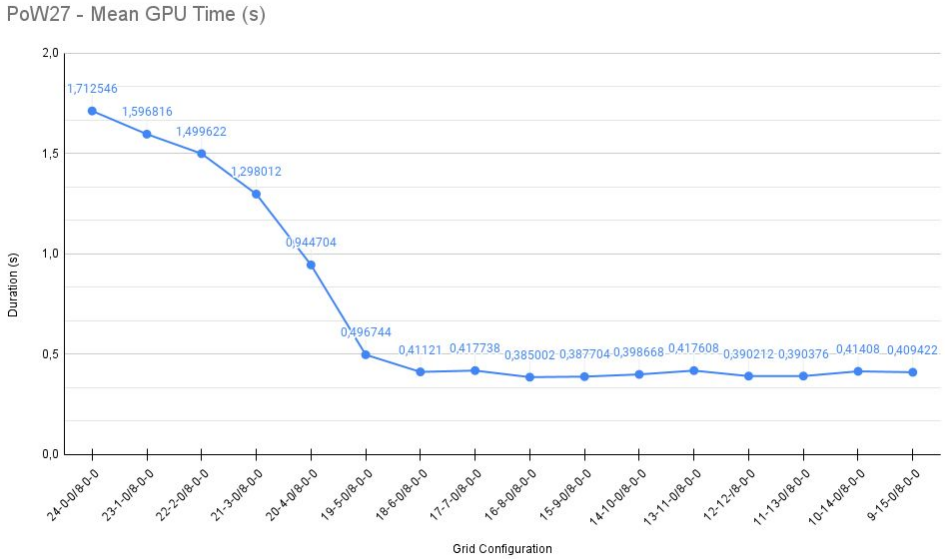
- In order to avoid threads having to perform more work than what is actually needed an if statement was inserted in the kernel that verifies if a solution was already found or not

```
//
// adjust the token_data pointer
//
tokens += 16u * (idx % 32u); //

if(tokens[15] == 0u)
  return;

idx /= 32u;                  //

//
// read the token data
//
m[ 0] = tokens[ 0];
```

- Once again best results were achieved for blockDim.x >= 32
- Best Value was block of 256 threads
- Performance starts to deteriorate for blockDim.x >= 512 threads



PoW27 - Optimization of launch grid



- Best Grid configuration was:
  - {16u, 8u, 8u, 0u}
  - Best time was 0.372882 s
  - speedUp obtained = 2.8217

- In this case it was observed that performance was worst when blockDim.x > 1
- So the best gridDim.y was calculated first and for values > 32 performance would improve
- After that we calculated the best block organization