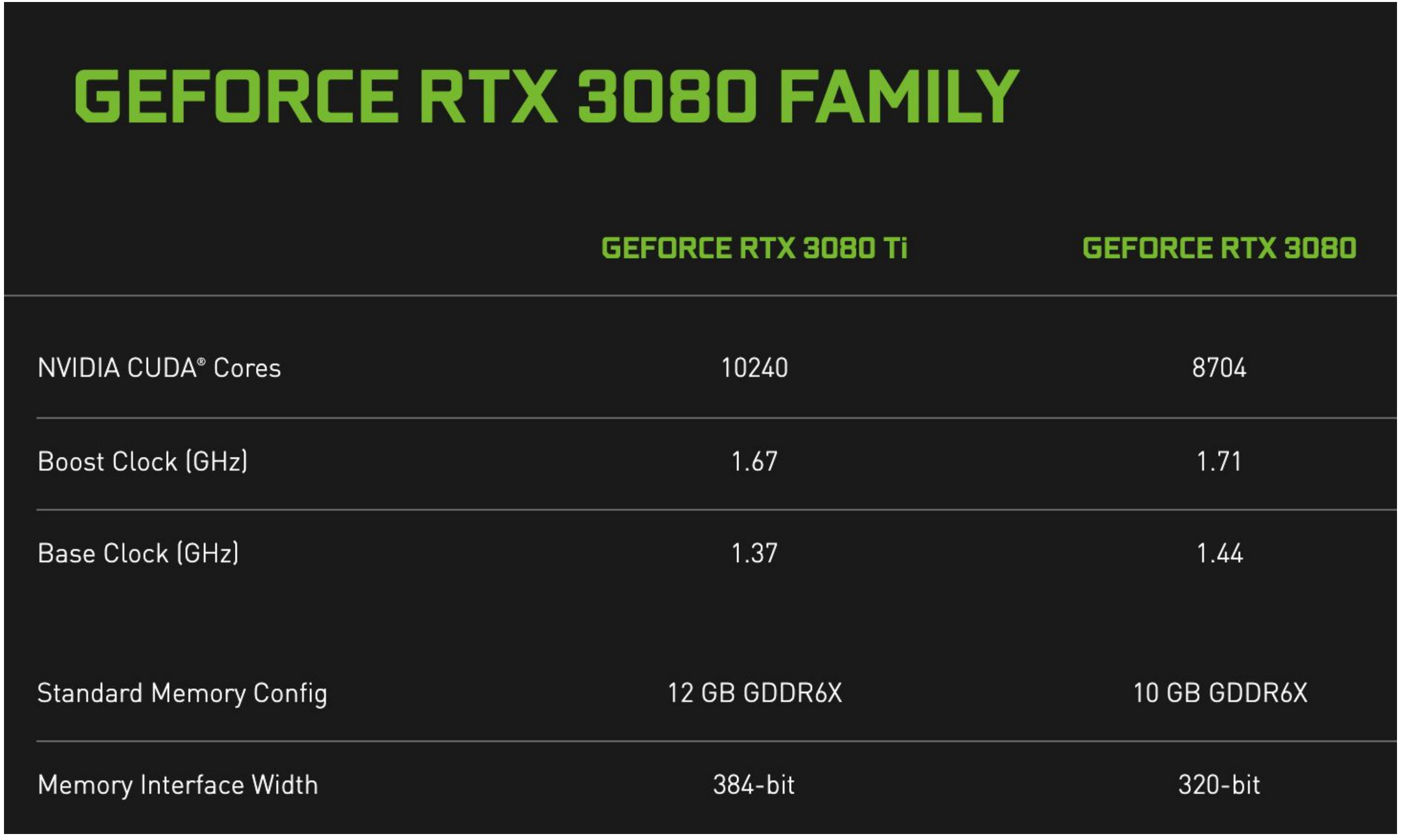
Backblaze

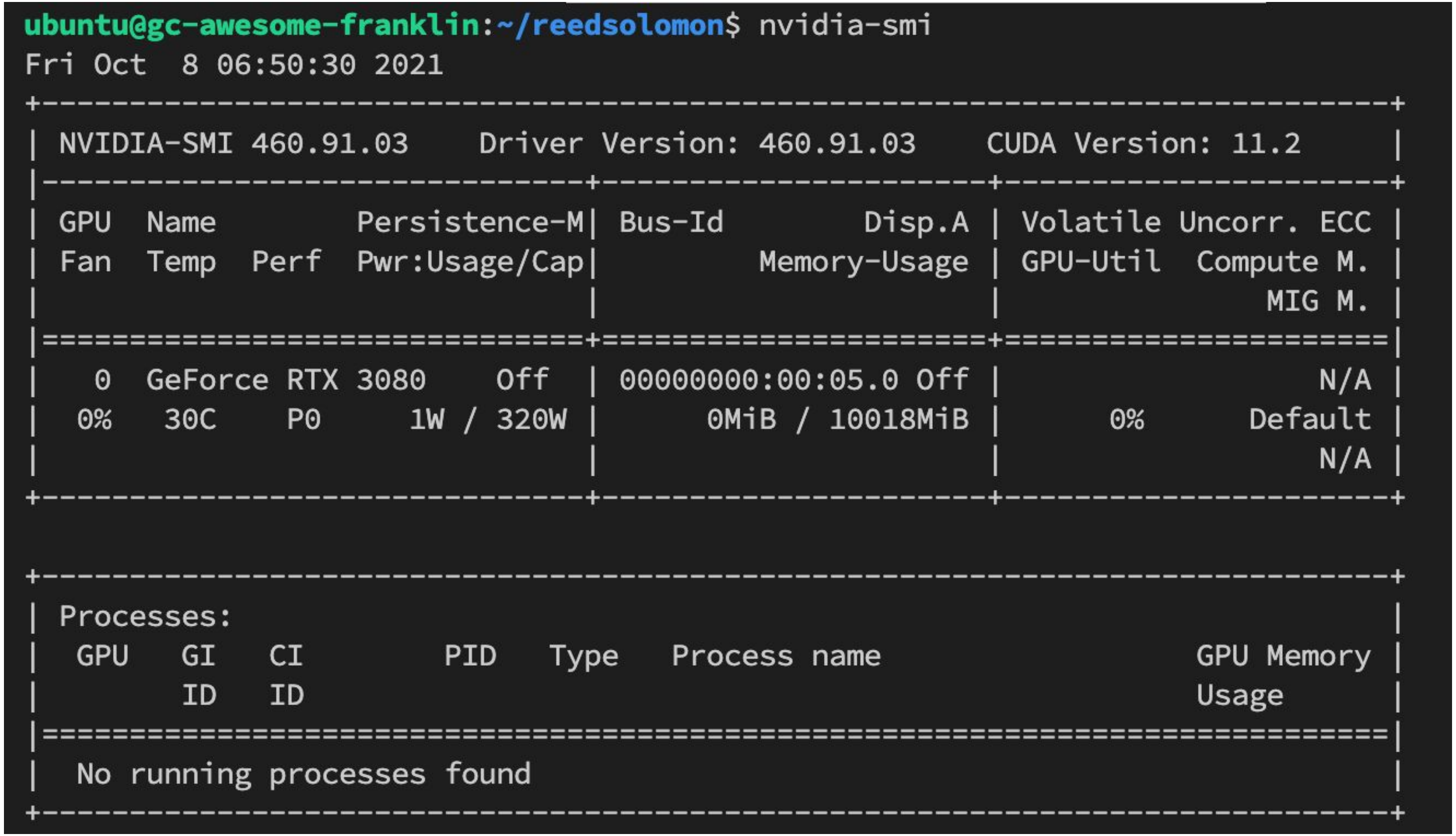
BzGPS
Backblaze GPU Parity Sharder

October 2021



GEFORCE RTX 3080 FAMILY

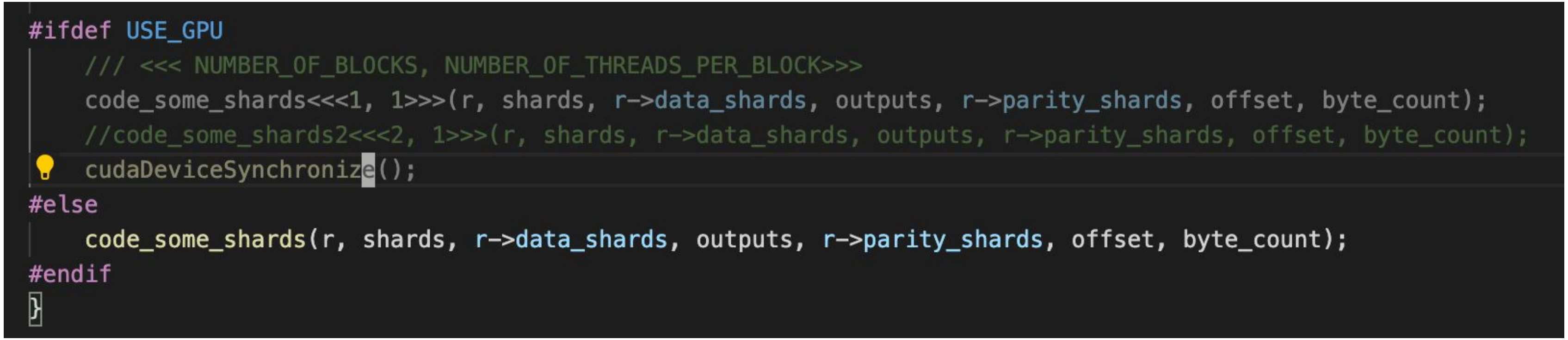
	GEFORCE RTX 3080 Ti	GEFORCE RTX 3080
NVIDIA CUDA® Cores	10240	8704
Boost Clock (GHz)	1.67	1.71
Base Clock (GHz)	1.37	1.44
Standard Memory Config	12 GB GDDR6X	10 GB GDDR6X
Memory Interface Width	384-bit	320-bit



```
ubuntu@gc-awesome-franklin:~/reedsolomon$ nvidia-smi
Fri Oct  8 06:50:30 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.91.03    Driver Version: 460.91.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce RTX 3080    Off  | 00000000:00:05.0 Off |                  N/A |
|  0%   30C    P0     1W / 320W |      0MiB / 10018MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

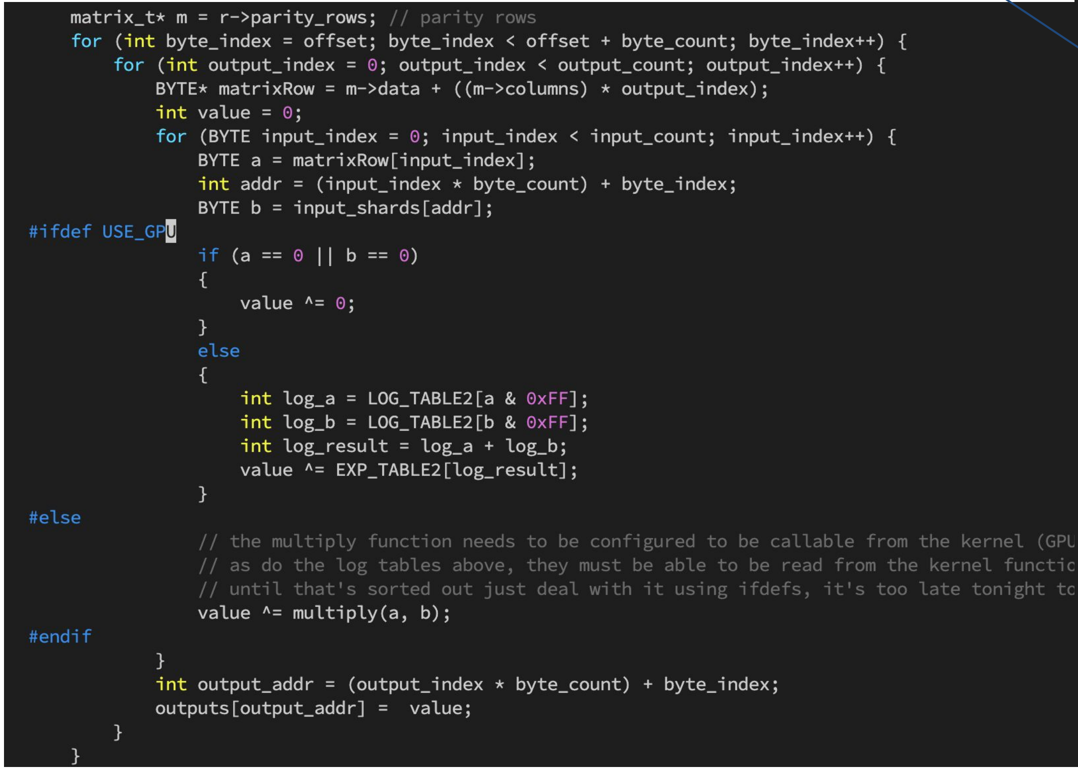
Blocks, threads per block and grids



```
#ifdef USE_GPU
    /// <<< NUMBER_OF_BLOCKS, NUMBER_OF_THREADS_PER_BLOCK>>>
    code_some_shards<<<1, 1>>>(r, shards, r->data_shards, outputs, r->parity_shards, offset, byte_count);
    //code_some_shards2<<<2, 1>>>(r, shards, r->data_shards, outputs, r->parity_shards, offset, byte_count);
    cudaDeviceSynchronize();
#else
    code_some_shards(r, shards, r->data_shards, outputs, r->parity_shards, offset, byte_count);
#endif
}
```

Backblaze

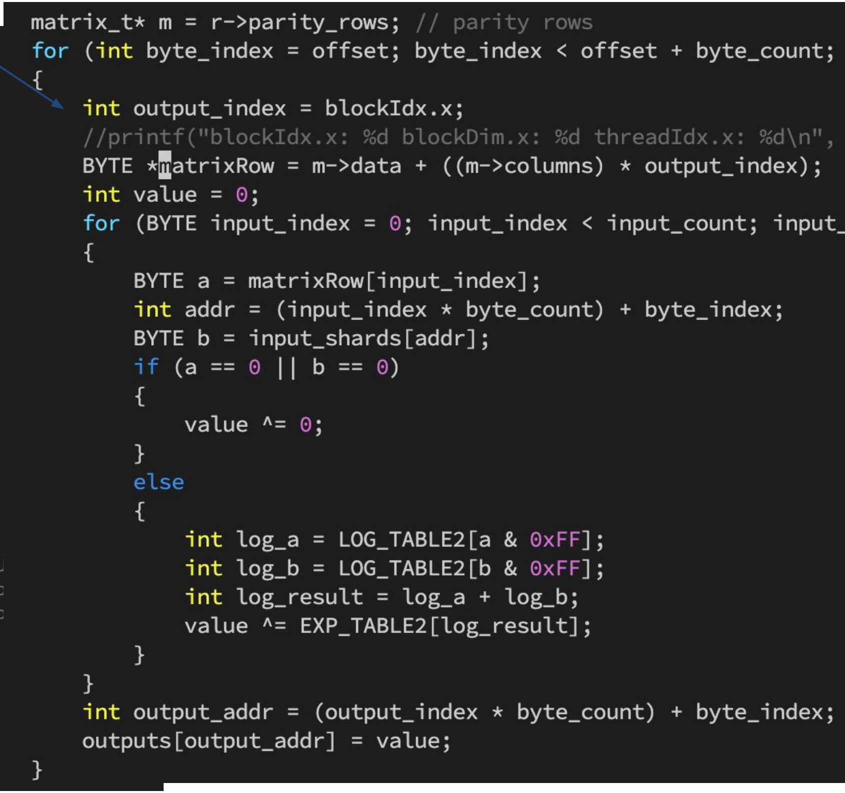
Left (C port/version)



```
  matrix_t* m = r->parity_rows; // parity rows
  for (int byte_index = offset; byte_index < offset + byte_count; byte_index++) {
    for (int output_index = 0; output_index < output_count; output_index++) {
      BYTE* matrixRow = m->data + ((m->columns) * output_index);
      int value = 0;
      for (BYTE input_index = 0; input_index < input_count; input_index++) {
        BYTE a = matrixRow[input_index];
        int addr = (input_index * byte_count) + byte_index;
        BYTE b = input_shards[addr];
#ifdef USE_GPU
        if (a == 0 || b == 0)
        {
          value ^= 0;
        }
        else
        {
          int log_a = LOG_TABLE2[a & 0xFF];
          int log_b = LOG_TABLE2[b & 0xFF];
          int log_result = log_a + log_b;
          value ^= EXP_TABLE2[log_result];
        }
#else
        // the multiply function needs to be configured to be callable from the kernel (GPU
        // as do the log tables above, they must be able to be read from the kernel functio
        // until that's sorted out just deal with it using ifdefs, it's too late tonight to
        value ^= multiply(a, b);
#endif
      }
      int output_addr = (output_index * byte_count) + byte_index;
      outputs[output_addr] =  value;
    }
  }
```

for() loop optimized away to a CUDA block

Right (GPU optimized)

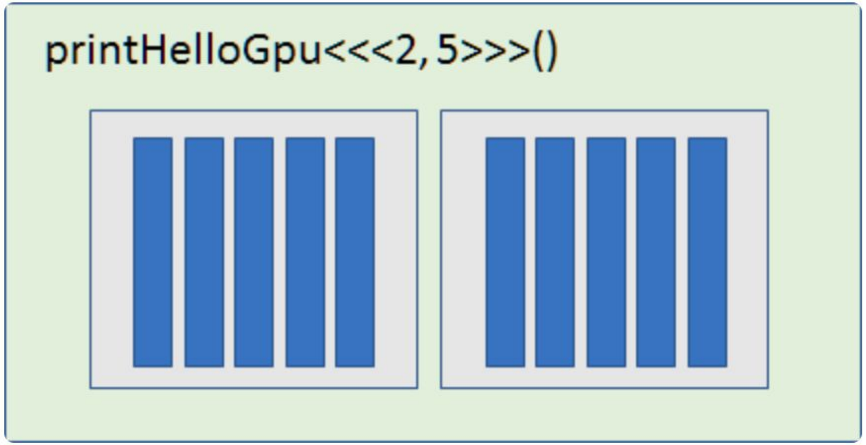


```
matrix_t* m = r->parity_rows; // parity rows
for (int byte_index = offset; byte_index < offset + byte_count;
{
  int output_index = blockIdx.x;
  //printf("blockIdx.x: %d blockDim.x: %d threadIdx.x: %d\n",
  BYTE *matrixRow = m->data + ((m->columns) * output_index);
  int value = 0;
  for (BYTE input_index = 0; input_index < input_count; input_
  {
    BYTE a = matrixRow[input_index];
    int addr = (input_index * byte_count) + byte_index;
    BYTE b = input_shards[addr];
    if (a == 0 || b == 0)
    {
      value ^= 0;
    }
    else
    {
      int log_a = LOG_TABLE2[a & 0xFF];
      int log_b = LOG_TABLE2[b & 0xFF];
      int log_result = log_a + log_b;
      value ^= EXP_TABLE2[log_result];
    }
  }
  int output_addr = (output_index * byte_count) + byte_index;
  outputs[output_addr] = value;
}
```

Backblaze

Execution Configuration

Let me try to explain this graphically:



```
printHelloGpu<<<2, 5>>>()
```

In the drawing, each blue rectangle represents a **thread**. Each gray rectangle represents a **block**.The green rectangle represents the **grid**.

Performance Demonstration

Faster?

- Unwind additional loops in the Reed Solomon erasure coding
    - At least the number of parity shards (4x) is possible - may restructure the loops for additional performance beyond 4x
    - Consider using a larger galois field

- Move driver into the kernel to avoid unnecessary copies - think sendfile(); today consider this
    - ReadFile - kernel → userspace
    - WriteFile - userpace → nvidia driver
        - Nvidia driver → GPU card/memory
        - GPU → Nvidia driver
    - ReadFile - nvidia driver → userspace
    - WriteFile - userspace → kernel filesystem write

- See if the file can be memory mapped directly into the GPU; avoiding any reads - or pass in memory as read-only into the GPU avoid synchronization

Backblaze

What do researchers say?

https://ieeexplore.ieee.org/document/4536322

> Performance results show that the GPU can outperform a modern CPU on this problem by an order of magnitude and also confirm that a GPU can be used to support a system with at least three parity disks with no performance penalty.
>
> *-- Accelerating Reed-Solomon coding in RAID systems with GPUs*

What's left?

- Implement decoding (file restore) loop
- Turn it into a service
- Consider additional GPU speedups
- Investigate moving it into the kernel

Backblaze

GPUs were meant for more than Bitcoin.

Thank You!!!

Backblaze