Sudoku 4x3 GPU Exact Enumeration Research Log

Verification

Verification of 2006 Pettersen/Silver result used 1035 CPU-hours over a twelve-day period between 8 June 2022 and 20 June 2022.

Note on 3x4 vs 4x3

I have been using 3x4 and 4x3 somewhat interchangeably. I starting using just 3x4 because that's how the Wikipedia table of results had it. After reconnecting with Kjell after all these years, I saw that he used 3x4 and 4x3 to refer to two distinct methods for the exact enumeration.

- In a 4x3 count, the 144,578 gangsters (equivalence class representatives) of a four-box, 12x4 band are determined and used with the other two bands.
- In a 3x4 count, the 2230 gangsters of a three-box, 12x3 stack are determined and used with the other three stacks.

Both the original 2006 enumeration, and this 2022 verification, are 4x3. Both methods must give the same result, but both Kjell and I believe that 4x3 is more efficient. Kjell has recently been thinking about the 3x4 version, and he may yet develop an efficient way to do it.

I think Kjell is right that I should be naming all this work 4x3, not 3x4, but there is a lot of 3x4 in filenames and the like because that's where I started. If you see 3x4, keep in mind that the method under investigation is 4x3.

Workload

CPU-hours is easy to tally but not a great way to measure the workload. It's very dependent on the machines I happened to have available, and tends to be biased by the slowest ones. It is a measure of serial hours, i.e. the time that would have been needed if the machines were not run in parallel. It also was affected by some difficulty I had in getting Windows and Ubuntu to run the cores at max speed, instead of trying to conserve power.

A parallel measure of CPU-hours would be easy to tally if all machines were run for the entire time—it would be essentially the same as the formula for parallel resistance. But that was not the case in practice.

Parallel execution on multiple cores of a single CPU is not the same as parallel execution on separate machines, because the parallel threads compete with each other for shared resources. Each of the 144,578 gangsters is enumerated independently, and the time for each one is defined and recorded as the elapsed time divided by the number of parallel threads.

Computers

The first six were used in the verification run. The speed is the average time to enumerate one gangster with a 32-gangster benchmark run of 865 – 896 (the first 32 in group 1).

Name	GHz	Cores	Threads	CPU	OS	Compiler	Speed
PT2017	3.10	4	8	x64 Xeon E3-1535M v6	Windows	MS VC++	16.5
PT2019	2.11	4	8	x64 i7-8650U	Windows	MS VC++	24
Judy7		4	8	x64	Windows	MS VC++	
PT2015		2	4	x64	Ubuntu	GCC	
Judy6		2	4	x64	Windows	MS VC++	
CGNX	2.90	2	4	x64 i7-7600U	Windows	MS VC++	
EPT2022	2.26	8	8	ARM-64 v8.2	Ubuntu	GCC	10

The speed (actually seconds/gangster) numbers are somewhat variable, run to run. For example, most of the PT2019 runs are in the 23.8 – 24.5 range, but a small number came in at around 22.1. I don't understand this 2 seconds/gangster bimodal variation on this particular machine. My current speculation is that the variation in parallel thread order may interact with the data caches and hyperthreading and occasionally produce this weird bimodal behavior.

More detailed timing follows. This version is slightly different than the baseline verification run. The order of the DoubleBoxCount outer loop was modified to try to achieve slightly better data cache performance. It may have made a very small improvement in speed, barely measurable.

PT2017

```
Using 8 threads
1,180,382 cache misses 4,087,416 code calls
Read count file gridCount_1-.txt, total time so far 0.46 hours

Profile tree:
Sudoku3x4                                                    535.947
  RowCode Init                      15400 *        0.0661 ->   0.001
  ColCode Init                       5775 *        0.0579 ->   0.000
  Row Tables                       369600 *        0.1185 ->   0.044
  Column Tables                    138600 *        0.1487 ->   0.021
  BoxCompatible Init                  715 *        1.6134 ->   0.001
  Column Nodes                      31104 *       14.3970 ->   0.448
  BandGang Construct                                           0.247
  Verify BandGang Tables                                       0.019
  Band Gangsters                                               3.790
  Fix gang cache                300155625 *        0.0022 ->   0.672
  Read/verify gangsters            144578 *        1.2478 ->   0.180
  Replace cache codes           300155625 *        0.0021 ->   0.645
  Construct GridCounter           1998150 *        0.2037 ->   0.407
  Grid counter                                               528.615
    GridCounter Setup                                          0.906
      Big tables                   119716 *        7.3130 ->   0.875
      Sort                                                     0.026
      Overhead                                                 0.005
    Main count loop                    32 * 16490904.0063 -> 527.709
    Overhead                                                   0.000
  Overhead                                                     0.856
```

PT2019

```
Using 8 threads
1,175,430 cache misses 4,070,668 code calls
Read count file test1_1-.txt, total time so far 0.00 hours

Profile tree:
Sudoku3x4                                                    772.867
  RowCode Init                      15400 *        0.0625 ->   0.001
  ColCode Init                       5775 *        0.0472 ->   0.000
  Row Tables                       369600 *        0.1273 ->   0.047
  Column Tables                    138600 *        0.1496 ->   0.021
  BoxCompatible Init                  715 *        1.6615 ->   0.001
  Column Nodes                      31104 *       15.4951 ->   0.482
  BandGang Construct                                           0.256
  Verify BandGang Tables                                       0.020
  Band Gangsters                                               5.599
  Fix gang cache                300155625 *        0.0023 ->   0.697
  Read/verify gangsters            144578 *        1.2698 ->   0.184
  Replace cache codes           300155625 *        0.0023 ->   0.676
  Construct GridCounter           1998150 *        0.2092 ->   0.418
  Grid counter                                               763.586
    GridCounter Setup                                          0.951
      Big tables                   119716 *        7.6898 ->   0.921
      Sort                                                     0.030
      Overhead                                                 0.000
    Main count loop                    32 * 23832356.1656 -> 762.635
    Overhead                                                   0.000
  Overhead                                                     0.879

Using 8 threads
8.5697 [144578]; 1,176,526 cache misses 4,075,140 code calls
1,176,526 cache misses 4,075,140 code calls

Profile tree:
Sudoku3x4                                                    719.754
  RowCode Init                      15400 *        0.0735 ->   0.001
  ColCode Init                       5775 *        0.0550 ->   0.000
  Row Tables                       369600 *        0.1282 ->   0.047
  Column Tables                    138600 *        0.1622 ->   0.022
  BoxCompatible Init                  715 *        1.8829 ->   0.001
  Column Nodes                      31104 *       14.8154 ->   0.461
  BandGang Construct                                           0.259
  Verify BandGang Tables                                       0.029
  Band Gangsters                                               5.337
  Fix gang cache                300155625 *        0.0023 ->   0.702
  Read/verify gangsters            144578 *        1.2762 ->   0.185
  Replace cache codes           300155625 *        0.0023 ->   0.676
  Construct GridCounter           1998150 *        0.2091 ->   0.418
  Grid counter                                               710.708
    GridCounter Setup                                          0.963
      Big tables                   119716 *        7.8065 ->   0.935
      Sort                                                     0.028
      Overhead                                                 0.001
    Main count loop                    32 * 22179517.5406 -> 709.745
    Overhead                                                   0.000
  Overhead                                                     0.909
```

EPT2022

```
Using 8 threads
1,158,310 cache misses 4,009,004 code calls
Read count file gridCount_1-.txt, total time so far 0.46 hours

Profile tree:
Sudoku3x4                                                    326.179
  RowCode Init                      15400 *        0.1489 ->   0.002
  ColCode Init                       5775 *        0.0968 ->   0.001
  Row Tables                       369600 *        0.2255 ->   0.083
  Column Tables                    138600 *        0.3447 ->   0.048
  BoxCompatible Init                  715 *        4.3080 ->   0.003
  Column Nodes                      31104 *       23.6487 ->   0.736
  BandGang Construct                                           0.250
  Verify BandGang Tables                                       0.022
  Band Gangsters                                               4.772
  Fix gang cache                300155625 *        0.0019 ->   0.581
  Read/verify gangsters            144578 *        0.5562 ->   0.080
  Replace cache codes           300155625 *        0.0022 ->   0.649
  Construct GridCounter           1998150 *        0.2685 ->   0.536
  Grid counter                                               317.587
    GridCounter Setup                                          2.208
      Big tables                   119716 *        6.0924 ->   0.729
      Sort                                                     1.478
      Overhead                                                 0.001
    Main count loop                    32 *  9855592.8388 -> 315.379
    Overhead                                                   0.000
  Overhead                                                     0.829
```

Jetson AGX Xavier GPU Basics

```
    Clock rate 1377000 kHz
    L2 cache size 524288
    Max blocks per multiprocessor 32
    Max grid size 2147483647.65535.65535
    Max block dimension 1024.1024.64
    Max threads per block 1024
    Max threads per multiprocessor 2048
    Multiprocessor count 8
    Reserved shared memory per block 0 bytes
    Shared memory per block 49152 bytes
    Shared memory per multiprocessor 98304 bytes
    Total global memory on device 32517738496 bytes
    Warp size in threads 32
```

Cuda First Cut

First cut at a Cuda GPU program for executing the main grid counting loop for the Sudoku 4x3 exact count. It makes very poor use of GPU resources, and is actually slower than running the CPU code on the Jetson's 8-core ARM v8.2 processors. The purpose of this first cut is to confirm that I understand the Nvidia tool chain and the most basic operations of Cuda. The code runs and gets the correct results.

The first step in using the GPU properly will be to deal with the poor memory access pattern, which radically degrades the GPU's memory bandwidth and stalls the compute elements.

Here is a pure GPU run—one thread runs all the setup, and calls Cuda code to run 16 blocks of 32 threads to do the main counting loops.

```
Using 1 thread
1,052,036 cache misses 3,653,317 code calls
Read count file gridCount_1-.txt, total time so far 0.46 hours

Profile tree:
Sudoku3x4                                                    389.234
  RowCode Init                      15400 *        0.1559 ->   0.002
  ColCode Init                       5775 *        0.1008 ->   0.001
  Row Tables                       369600 *        0.2489 ->   0.092
  Column Tables                    138600 *        0.3945 ->   0.055
  BoxCompatible Init                  715 *        5.3005 ->   0.004
  Column Nodes                      31104 *       23.1097 ->   0.719
  BandGang Construct                                           0.252
  Verify BandGang Tables                                       0.028
  Band Gangsters                                              10.687
  Fix gang cache                300155625 *        0.0019 ->   0.580
  Read/verify gangsters            144578 *        0.5620 ->   0.081
  Replace cache codes           300155625 *        0.0019 ->   0.562
  Construct GridCounter           1998150 *        0.2820 ->   0.564
  Give band counts to GPU       300155625 *        0.0026 ->   0.795
  Grid counter                                               373.769
    GridCounter Setup                                          2.582
      Big tables                   119716 *        6.3462 ->   0.760
      Sort                                                     1.616
      Tables -> GPU                                            0.205
      Overhead                                                 0.001
    Main count loop                    32 * 11599569.5722 -> 371.186
    Overhead                                                   0.000
  Overhead                                                     1.045
```

Here is a heterogeneous run—seven threads run on the ARM cores and one thread calls the Cuda/GPU code:

```
Using 8 threads
1,158,288 cache misses 4,009,055 code calls
Read count file gridCount_1-.txt, total time so far 0.46 hours

Profile tree:
Sudoku3x4                                                    336.808
  RowCode Init                      15400 *        0.1379 ->   0.002
  ColCode Init                       5775 *        0.0980 ->   0.001
  Row Tables                       369600 *        0.1976 ->   0.073
  Column Tables                    138600 *        0.2959 ->   0.041
  BoxCompatible Init                  715 *        3.6988 ->   0.003
  Column Nodes                      31104 *       24.4196 ->   0.760
  BandGang Construct                                           0.253
  Verify BandGang Tables                                       0.022
  Band Gangsters                                               4.569
  Fix gang cache                300155625 *        0.0020 ->   0.589
  Read/verify gangsters            144578 *        0.5550 ->   0.080
  Replace cache codes           300155625 *        0.0021 ->   0.629
  Construct GridCounter           1998150 *        0.2684 ->   0.536
  Give band counts to GPU       300155625 *        0.0028 ->   0.854
  Grid counter                                               327.460
    GridCounter Setup                                         36.896
      Big tables                   119716 *        6.1662 ->   0.738
      Sort                                                    35.850
      Tables -> GPU                                            0.306
      Overhead                                                 0.001
    Main count loop                    32 *  9080136.8877 -> 290.564
    Overhead                                                   0.000
  Overhead                                                     0.935
```

Compilation Command Line

```
nvcc --m64 --std c++17 --compiler-options -std=c++17,-march=armv8-a+simd,-
Ofast,-Wno-format,-DJETSON --linker-options -pthread --include-path . -o
sudoku3x4 bignumMT.cpp profile.cpp general.cpp timer.cpp Sudoku3x4.cpp
gridCount.cu
```