f\_par\_pro\_WiSe 2024/25;

2. Aufgabe: Matrix-Matrix Multiplikation;

Ersteller: Norbert Baumstark

Verwendete Hardware:

* HP Victus TG02-2301ng
* NVIDIA® GeForce RTX™ 4060 (8 GB GDDR6 dediziert)

Verwendete Programmierung-Umgebung: Microsoft Visual Studio auf Windows 11 Home

Inhalt

[1. Problemlösungsansatz 2](#_Toc182746820)

[2. Messergebnisse 4](#_Toc182746821)

[3. Vergleiche und Schlussfolgerungen 7](#_Toc182746822)

[3.1. CudaMallocHost statt Malloc 7](#_Toc182746823)

[3.1.1. Performance-Vergleich CudaMallocHost vs. Malloc ohne Shared-Memory 7](#_Toc182746824)

[3.1.2. Performance-Vergleich CudaMallocHost vs. Malloc mit Shared-Memory 8](#_Toc182746825)

[3.2. Verwendung von shared memory 10](#_Toc182746826)

[3.2.1. Performance-Vergleich shared-memory mit Malloc 10](#_Toc182746827)

[3.2.2. Performance-Vergleich shared-memory mit CudaMallocHost 11](#_Toc182746828)

# 1. Problemlösungsansatz

Es sind insgesamt vier Programmvarianten zu erstellen, eine einfache Version ohne und eine mit shared memory und jeweils eine Version für die Ergebnismatrix mit malloc und mit cudaMallocHost. Der einfache Kernel ohne shared memory stellt sich wie folgt dar

\_\_global\_\_ void dgemm\_gpu\_simple(const float\* a, const float\* b, float\* c, const int n) {

int x = threadIdx.x + blockIdx.x \* blockDim.x;  
 int y = threadIdx.y + blockIdx.y \* blockDim.y;  
 int id = x + y \* blockDim.x \* gridDim.x;

int vx = id % n; // Spalte  
 int vy = id - vx; // Zeile

int i;

for(i=0;i<n;++i)  
 c[id] += a[vy+i] \* b[vx+i\*n];  
}

Jeder Thread berechnet ein Element der Matrix. Die Matrixspalt ergibt sich aus der Modulooperation, die Zeile durch Abzug der Spaltennummer von dem jeweiligen gewählten Matrixelement.

Der Kernel mit shared memory baut auf der zweidimensionalen Threadnutzung auf, indem zweidimensionale Arrays für das shared memory verwendet werden, für die Zeilen die Y-Dimension und für die Spalten die X-Dimension:

\_global\_\_ void dgemm\_gpu\_shared(const float\* a, const float\* b, float\* c, const int n) {

\_\_shared\_\_ float sA[BLOCK\_SIZE][BLOCK\_SIZE]; // Tile size of 32x32  
 \_\_shared\_\_ float sB[BLOCK\_SIZE][BLOCK\_SIZE];

int Row = blockDim.y \* blockIdx.y + threadIdx.y;  
 int Col = blockDim.x \* blockIdx.x + threadIdx.x;

float Cvalue = 0.0;  
 sA[threadIdx.y][threadIdx.x] = 0.0;  
 sB[threadIdx.y][threadIdx.x] = 0.0;

int Row\_n = Row \* n;  
 int anz\_BL = (n + BLOCK\_SIZE - 1) / BLOCK\_SIZE; // Number of Blocks in Matrix  
 int ph, j;

for (ph = 0; ph < anz\_BL; ph++) {

// Load  
int ph\_BLOCK\_SIZE = ph \* BLOCK\_SIZE;  
sA[threadIdx.y][threadIdx.x] = a[Row\_n + threadIdx.x + ph\_BLOCK\_SIZE];  
sB[threadIdx.y][threadIdx.x] = b[(threadIdx.y + ph\_BLOCK\_SIZE) \* n + Col];  
\_\_syncthreads();

// Calc  
 for (j = 0; j < BLOCK\_SIZE; ++j) {  
 Cvalue += sA[threadIdx.y][j] \* sB[j][threadIdx.x];  
 }  
 }

//Store   
 c[Row\_n + Col] = Cvalue;  
}

Es folgen dann die Lade- sowie die Rechenphase innerhalb der äußeren Schleife. Die Schleifendurchlaufanzahl ergibt sich aufgerundet aus der Anzahl der Elemente durch die Blockgröße. Im ersten Teil der Schleife lädt jeder Thread jeweils eine Stelle in die Summanden-Matrizen A und B. Da dies im shared memory erfolgt, sind an der \_\_syncthreads-Stelle beide Summanden-Matrizen kollektiv blockweise befüllt und können für die folgende Rechenphase verwendet werden, die pro Block als innere Schleife ein Element für die Ergebnismatrix berechnet. Auch hier berechnen die Threads einzeln die Zellen der Ergebnismatrix, das Laden der Summanden-Matrizen erfolgt jedoch effizienzsteigernd kollektiv pro Block. Abschließend wird der berechnete Wert in dem jeweiligen Element der Ergebnismatrix gespeichert.

Der eigentliche Kernelaufruf mit den vor- und nachgelagerten Speicherübertragungen unterscheidet sich in beiden Varianten nicht:

cudaMemcpy(d\_a, h\_a, size, cudaMemcpyHostToDevice);  
cudaMemcpy(d\_b, h\_b, size, cudaMemcpyHostToDevice);

dgemm\_gpu\_shared << < gridDim, blockDim >> > (d\_a, d\_b, d\_c, n);

cudaMemcpy(h\_c, d\_c, size, cudaMemcpyDeviceToHost);

Dieser Kernel bleibt auch identisch, wenn für die Ergebnismatrix im Host statt malloc (h\_c = (float\*)malloc(size);) cudaMalloc (cudaMallocHost((void\*\*)&h\_c, size);) verwendet wird (und entsprechend cudaFreeHost(h\_c); statt free(h\_c);). Die Lösungsansätze sind damit lokal begrenzt und miteinander kombinierbar.

# 2. Messergebnisse

Es wurden pro Programmvariante und pro Matrixgröße jeweils 25 Durchläufe gemessen. Davor wurde das Programm dreimal durchlaufen lassen, um Initialverzögerungen in der Gesamtheit zu vermeiden.

Performance-Werte (time\* bezeichnet die Zeitmessung aus der Vorlage) **ohne** shared-memory

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Simple** | **Host** |  |  |  |  |  | **Simple Malloc** | |  |  |  |  |
| 1024 | Min | Max | Avg | Median | StdDV |  | 1024 | Min | Max | Avg | Median | StdDV |
| time\* | 7,9695 | 8,84365 | 8,2775116 | 8,23094 | 0,275462688 |  | time\* | 8,54682 | 10,0484 | 8,7973168 | 8,66202 | 0,369273484 |
| GFLOPS | 121,414 | 134,731 | 129,85796 | 130,452 | 4,210520808 |  | GFLOPS | 106,857 | 125,631 | 122,24848 | 123,96 | 4,653007072 |
| Memcpy AB | 1,2057 | 1,92874 | 1,3412284 | 1,26586 | 0,170103734 |  | Memcpy AB | 1,21338 | 2,20432 | 1,318218 | 1,25242 | 0,203478401 |
| Memcpy C | 0,35936 | 0,425056 | 0,39341952 | 0,39232 | 0,017256503 |  | Memcpy C | 0,906272 | 1,36867 | 0,97998696 | 0,955488 | 0,099489998 |
| Kernel | 6,35267 | 6,99056 | 6,5184732 | 6,45763 | 0,179722812 |  | Kernel | 6,34646 | 6,83626 | 6,478102 | 6,4064 | 0,15266868 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0 | 0,050592 | 0,00649728 | 0 | 0,012280469 |  | Sync | 0 | 0,011072 | 0,00532224 | 0,009856 | 0,005122061 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 2048 | Min | Max | Avg | Median | StdDV |
| time\* | 57,5964 | 60,624 | 58,370568 | 58,1321 | 0,705052269 |  | time\* | 59,6924 | 61,7639 | 60,470584 | 60,3257 | 0,510244536 |
| GFLOPS | 141,692 | 149,14 | 147,18316 | 147,766 | 1,744907601 |  | GFLOPS | 139,077 | 143,903 | 142,06156 | 142,393 | 1,191885568 |
| Memcpy AB | 4,96218 | 7,4663 | 5,3230076 | 5,01981 | 0,609615009 |  | Memcpy AB | 4,97971 | 6,91194 | 5,3551144 | 5,19539 | 0,436427909 |
| Memcpy C | 1,32128 | 1,67082 | 1,4354972 | 1,42275 | 0,095057438 |  | Memcpy C | 3,2705 | 5,4681 | 3,5484208 | 3,43958 | 0,446091195 |
| Kernel | 51,0103 | 52,1054 | 51,578532 | 51,5711 | 0,254581774 |  | Kernel | 50,9673 | 52,2811 | 51,536856 | 51,5544 | 0,326201475 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0 | 0,050528 | 0,01966592 | 0,013696 | 0,011552711 |  | Sync | 0 | 0,036384 | 0,01644928 | 0,011872 | 0,010798253 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4096 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 506,859 | 521,217 | 509,90148 | 508,921 | 3,019611049 |  | time\* | 515,145 | 528,439 | 517,69092 | 516,736 | 2,735290082 |
| GFLOPS | 131,844 | 135,579 | 134,77476 | 135,03 | 0,786349301 |  | GFLOPS | 130,042 | 133,398 | 132,74596 | 132,988 | 0,692603291 |
| Memcpy AB | 20,0248 | 25,0946 | 21,455296 | 21,2568 | 1,107443334 |  | Memcpy AB | 19,4637 | 25,7523 | 21,356536 | 20,8605 | 1,52698667 |
| Memcpy C | 5,16403 | 5,4111 | 5,2615456 | 5,256 | 0,049764989 |  | Memcpy C | 12,5129 | 14,8018 | 13,303728 | 13,0653 | 0,652983144 |
| Kernel | 481,335 | 495,211 | 483,14136 | 482,352 | 3,113073502 |  | Kernel | 481,572 | 494,838 | 483,0028 | 482,581 | 2,467475374 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0,011072 | 0,039136 | 0,01768704 | 0,014048 | 0,00829655 |  | Sync | 0,012384 | 0,037344 | 0,01708672 | 0,013536 | 0,007922411 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8192 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 3945,25 | 3964,86 | 3955,2328 | 3956,14 | 5,255462316 |  | time\* | 3982,05 | 4009,7 | 3995,1108 | 3992,79 | 7,072001652 |
| GFLOPS | 138,657 | 139,346 | 138,99476 | 138,963 | 0,184692237 |  | GFLOPS | 137,106 | 138,059 | 137,60756 | 137,687 | 0,24369474 |
| Memcpy AB | 76,8313 | 82,7437 | 78,670272 | 77,8856 | 1,850850315 |  | Memcpy AB | 77,1926 | 85,2304 | 79,467156 | 79,6447 | 1,849406781 |
| Memcpy C | 20,4359 | 20,6283 | 20,54494 | 20,5356 | 0,04929447 |  | Memcpy C | 48,7881 | 53,815 | 50,292948 | 49,8421 | 1,11202297 |
| Kernel | 3846,85 | 3866,24 | 3855,9776 | 3857,64 | 5,223605866 |  | Kernel | 3852,44 | 3881,27 | 3865,3268 | 3863,26 | 6,911455835 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0,011136 | 0,032256 | 0,01561856 | 0,013312 | 0,006025232 |  | Sync | 0,009344 | 0,035104 | 0,01519488 | 0,013568 | 0,006027398 |
| Free | 0 | 36,7902 | 1,471608 | 0 | 7,209377403 |  | Free | 0 | 26,9029 | 1,076116 | 0 | 5,271870208 |

Performance-Werte (time\* bezeichnet die Zeitmessung aus der Vorlage) **mit** shared-Memory

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Shared Host** |  |  |  |  |  |  | **Shared Malloc** | |  |  |  |  |
| 1024 | Min | Max | Avg | Median | StdDV |  | 1024 | Min | Max | Avg | Median | StdDV |
| time\* | 3,57709 | 3,76378 | 3,6573508 | 3,64845 | 0,044514874 |  | time\* | 4,15789 | 5,53658 | 4,3801448 | 4,29302 | 0,279895875 |
| GFLOPS | 285,283 | 300,172 | 293,62784 | 294,301 | 3,54447273 |  | GFLOPS | 193,936 | 258,242 | 246,00172 | 250,113 | 13,58648163 |
| Memcpy AB | 1,20064 | 1,29354 | 1,2491036 | 1,25325 | 0,019464737 |  | Memcpy AB | 1,19738 | 1,87312 | 1,3392492 | 1,26221 | 0,147634621 |
| Memcpy C | 0,36464 | 0,480128 | 0,39323904 | 0,386976 | 0,02411022 |  | Memcpy C | 0,91728 | 1,32362 | 0,97891384 | 0,951744 | 0,091034358 |
| Kernel | 1,96963 | 2,03325 | 1,995918 | 1,99277 | 0,0179026 |  | Kernel | 1,948 | 2,3672 | 2,0385172 | 1,99834 | 0,101684362 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0 | 0,00992 | 0,0056768 | 0,009248 | 0,004638636 |  | Sync | 0 | 0,034976 | 0,00429056 | 0 | 0,007773073 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 2048 | Min | Max | Avg | Median | StdDV |
| time\* | 20,5047 | 22,5798 | 21,00188 | 20,8028 | 0,558626127 |  | time\* | 22,3883 | 26,0487 | 23,260036 | 22,8093 | 0,990395958 |
| GFLOPS | 380,426 | 418,925 | 409,28668 | 412,922 | 10,48335964 |  | GFLOPS | 329,765 | 383,679 | 369,93188 | 376,598 | 14,84501271 |
| Memcpy AB | 4,96038 | 6,33187 | 5,2754828 | 5,06781 | 0,396866796 |  | Memcpy AB | 4,97226 | 7,25891 | 5,4119724 | 5,17952 | 0,592618165 |
| Memcpy C | 1,32826 | 1,41782 | 1,3529952 | 1,34707 | 0,020827221 |  | Memcpy C | 3,20195 | 5,44474 | 3,5320632 | 3,30979 | 0,49500492 |
| Kernel | 14,174 | 15,0701 | 14,3479 | 14,2276 | 0,258252224 |  | Kernel | 14,169 | 14,5666 | 14,292832 | 14,2099 | 0,137384703 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0 | 0,02848 | 0,00944768 | 0,009344 | 0,007249792 |  | Sync | 0 | 0,033088 | 0,00667776 | 0 | 0,009291062 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4096 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 129,475 | 137,619 | 131,21436 | 130,286 | 2,00591174 |  | time\* | 135,816 | 150,116 | 139,99216 | 138,823 | 3,764821342 |
| GFLOPS | 499,346 | 530,757 | 523,8378 | 527,451 | 7,763880293 |  | GFLOPS | 457,775 | 505,975 | 491,22384 | 495,014 | 12,75996324 |
| Memcpy AB | 19,9682 | 26,6268 | 21,288444 | 20,8619 | 1,43640647 |  | Memcpy AB | 19,6337 | 29,4847 | 21,517256 | 20,7439 | 2,080440651 |
| Memcpy C | 5,15546 | 5,64749 | 5,2655956 | 5,24522 | 0,098294077 |  | Memcpy C | 12,4258 | 14,2086 | 13,325216 | 13,4102 | 0,601460087 |
| Kernel | 103,449 | 112,279 | 104,62364 | 104,329 | 1,591258468 |  | Kernel | 103,152 | 113,584 | 105,11664 | 103,644 | 3,154904599 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0,009504 | 0,033344 | 0,01601024 | 0,013248 | 0,00698908 |  | Sync | 0,009536 | 0,101088 | 0,02324864 | 0,013824 | 0,019196687 |
| Free | 0 | 0 | 0 | 0 | 0 |  | Free | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8192 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 956,253 | 965,238 | 959,76692 | 959,208 | 2,124512385 |  | time\* | 986,696 | 1012,97 | 992,65364 | 990,993 | 5,495888622 |
| GFLOPS | 569,555 | 574,906 | 572,80412 | 573,135 | 1,265667992 |  | GFLOPS | 542,714 | 557,168 | 553,84104 | 554,752 | 3,031736512 |
| Memcpy AB | 76,8539 | 82,3763 | 78,580764 | 78,3627 | 1,591657074 |  | Memcpy AB | 77,1562 | 99,4691 | 81,455704 | 80,3295 | 4,820455076 |
| Memcpy C | 20,4108 | 20,7192 | 20,540012 | 20,5273 | 0,063635854 |  | Memcpy C | 48,5895 | 55,9249 | 51,50348 | 51,4688 | 1,83675841 |
| Kernel | 858,634 | 867,865 | 860,60612 | 860,342 | 1,711226305 |  | Kernel | 857,776 | 865,089 | 859,66544 | 859,578 | 1,345219687 |
| Malloc | 0 | 0 | 0 | 0 | 0 |  | Malloc | 0 | 0 | 0 | 0 | 0 |
| Sync | 0,011392 | 0,032384 | 0,01598976 | 0,013472 | 0,006747911 |  | Sync | 0,012 | 0,04048 | 0,02168064 | 0,01504 | 0,010893353 |
| Free | 0 | 40,1623 | 4,70778 | 0 | 12,75079258 |  | Free | 0 | 0 | 0 | 0 | 0 |

# 3. Vergleiche und Schlussfolgerungen

## 3.1. cudaMallocHost statt malloc

### 3.1.1. Performance-Vergleich cudaMallocHost vs. malloc ohne shared-memory

Werte: (malloc - cudaMallocHost) / malloc

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1024 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 6,75% | 11,99% | 5,91% | 4,98% | 25,40% |  | time\* | 1,61% | 1,37% | 1,50% | 1,51% | -10,39% |
| GFLOPS | -13,62% | -7,24% | -6,22% | -5,24% | 9,51% |  | GFLOPS | -1,39% | -1,63% | -1,53% | -1,54% | -13,54% |
| Memcpy AB | 0,63% | 12,50% | -1,75% | -1,07% | 16,40% |  | Memcpy AB | -2,88% | 2,55% | -0,46% | -1,90% | 27,48% |
| Memcpy C | 60,35% | 68,94% | 59,85% | 58,94% | 82,66% |  | Memcpy C | 58,73% | 63,44% | 60,45% | 59,77% | 92,38% |
| Kernel | -0,10% | -2,26% | -0,62% | -0,80% | -17,72% |  | Kernel | 0,05% | -0,08% | -0,03% | 0,05% | -26,16% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | -356,94% | -22,08% | 100,00% | -139,76% |  | Sync | 10,59% | -4,80% | -3,51% | -3,78% | -4,72% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 3,51% | 1,85% | 3,47% | 3,64% | -38,18% |  | time\* | 0,92% | 1,12% | 1,00% | 0,92% | 25,69% |
| GFLOPS | -1,88% | -3,64% | -3,61% | -3,77% | -46,40% |  | GFLOPS | -1,13% | -0,93% | -1,01% | -0,93% | 24,21% |
| Memcpy AB | 0,35% | -8,02% | 0,60% | 3,38% | -39,68% |  | Memcpy AB | 0,47% | 2,92% | 1,00% | 2,21% | -0,08% |
| Memcpy C | 59,60% | 69,44% | 59,55% | 58,64% | 78,69% |  | Memcpy C | 58,11% | 61,67% | 59,15% | 58,80% | 95,57% |
| Kernel | -0,08% | 0,34% | -0,08% | -0,03% | 21,96% |  | Kernel | 0,15% | 0,39% | 0,24% | 0,15% | 24,42% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | -38,87% | -19,55% | -15,36% | -6,99% |  | Sync | -19,18% | 8,11% | -2,79% | 1,89% | 0,04% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | -36,75% | -36,75% | 0,00% | -36,75% |

Bei der Matrix-Größe von 1024 ist ein Leistungsgewinn von ca. 6 % (Zeit und GFLOPS) anfangs beachtlich. Mit zunehmender Matrix-Größe sinkt dieser deutlich. Bereits bei 2048 beträgt er nur noch knapp 3,5 % (Zeit und GFLOPS), bei 4096 nur noch 1,5 % (Zeit und GFLOPS), bei 8192 fällt er auf 1 % (Zeit und GFLOPS). Ein wesentlicher Faktor ist die deutlich verkürzte Zeit Memcpy-Zeit für C, welche allgemein bei ca. 60 % liegt. Die Differenzen in den übrigen Größen (Kernel, Sync, Free, Malloc, Memcpy AB) dürften zufällig sein.

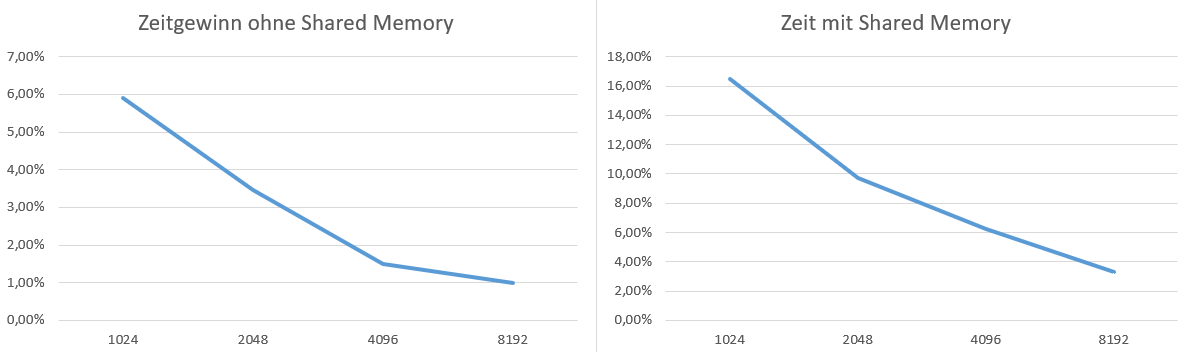
### 3.1.2. Performance-Vergleich cudaMallocHost vs. malloc mit shared-memory

Werte: (malloc - cudaMallocHost) / malloc

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1024 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 13,97% | 32,02% | 16,50% | 15,01% | 84,10% |  | time\* | 4,67% | 8,32% | 6,27% | 6,15% | 46,72% |
| GFLOPS | -47,10% | -16,24% | -19,36% | -17,67% | 73,91% |  | GFLOPS | -9,08% | -4,90% | -6,64% | -6,55% | 39,15% |
| Memcpy AB | -0,27% | 30,94% | 6,73% | 0,71% | 86,82% |  | Memcpy AB | -1,70% | 9,69% | 1,06% | -0,57% | 30,96% |
| Memcpy C | 60,25% | 63,73% | 59,83% | 59,34% | 73,52% |  | Memcpy C | 58,51% | 60,25% | 60,48% | 60,89% | 83,66% |
| Kernel | -1,11% | 14,11% | 2,09% | 0,28% | 82,39% |  | Kernel | -0,29% | 1,15% | 0,47% | -0,66% | 49,56% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | 71,64% | -32,31% | 0,00% | 40,32% |  | Sync | 0,34% | 67,01% | 31,13% | 4,17% | 63,59% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 8,41% | 13,32% | 9,71% | 8,80% | 43,60% |  | time\* | 3,09% | 4,71% | 3,31% | 3,21% | 61,34% |
| GFLOPS | -15,36% | -9,19% | -10,64% | -9,65% | 29,38% |  | GFLOPS | -4,95% | -3,18% | -3,42% | -3,31% | 58,25% |
| Memcpy AB | 0,24% | 12,77% | 2,52% | 2,16% | 33,03% |  | Memcpy AB | 0,39% | 17,18% | 3,53% | 2,45% | 66,98% |
| Memcpy C | 58,52% | 73,96% | 61,69% | 59,30% | 95,79% |  | Memcpy C | 57,99% | 62,95% | 60,12% | 60,12% | 96,54% |
| Kernel | -0,04% | -3,46% | -0,39% | -0,12% | -87,98% |  | Kernel | -0,10% | -0,32% | -0,11% | -0,09% | -27,21% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | 13,93% | -41,48% | 0,00% | 21,97% |  | Sync | 5,07% | 20,00% | 26,25% | 10,43% | 38,05% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |

Auch hier trägt die Reduktion in Memcpy für C im Wesentlichen zum Leistungsgewinn von jeweils ca. 60 % bei. Mit zunehmender Größe fällt der Leistungsgewinn insgesamt ab, beginnend von 16,5 % Zeit (GFLOPS 20 %) bis auf ca. 3 % (Zeit und GFLOPS). Die Auswirkungen auf die Kernelzeit sind unwesentlich, ebenso die Veränderungen bei den übrigen Stationen.

Im Ergebnis zeigt sich die optimierende Wirkung von CudaMallocHost hinreichend deutlich.



Verwendet wurden die Average-Werte.

Die Abnahme im Zeitgewinn erklärt sich daraus, dass mit zunehmender Matrixgröße der lineare Performance-Gewinn in der Übertragung hinter der höherkomplexen Berechnung (Matrixmultiplikation: grundsätzlich O(n³), mit Parallelität in den hier verwendeten Modellen auf bis zu O(n²) reduzierbar) zurückfällt. Bei Shared Memory wirkt dieser zeitersparend mit einem Faktor von ca. 3. Der Verlauf ist jedoch hinreichend ähnlich, der Betrag der negativen Steigung nimmt jedoch etwas früher ab.

## 3.2. Verwendung von shared memory

### 3.2.1. Performance-Vergleich shared-memory mit malloc

Werte: (ohne shared – mit shared) / shared

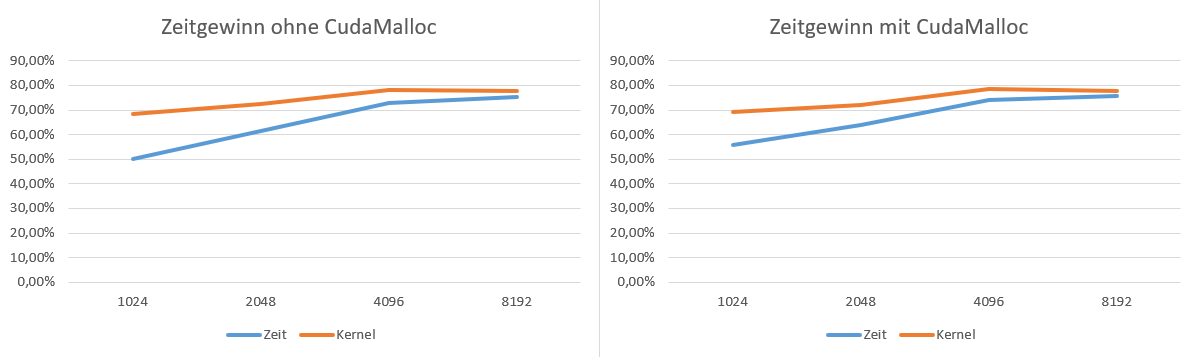
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1024 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 51,35% | 44,90% | 50,21% | 50,44% | 24,20% |  | time\* | 73,64% | 71,59% | 72,96% | 73,13% | -37,64% |
| GFLOPS | -81,49% | -105,56% | -101,23% | -101,77% | -191,99% |  | GFLOPS | -252,02% | -279,30% | -270,05% | -272,22% | -1742,32% |
| Memcpy AB | 1,32% | 15,03% | -1,60% | -0,78% | 27,44% |  | Memcpy AB | -0,87% | -14,49% | -0,75% | 0,56% | -36,24% |
| Memcpy C | -1,21% | 3,29% | 0,11% | 0,39% | 8,50% |  | Memcpy C | 0,70% | 4,01% | -0,16% | -2,64% | 7,89% |
| Kernel | 69,31% | 65,37% | 68,53% | 68,81% | 33,40% |  | Kernel | 78,58% | 77,05% | 78,24% | 78,52% | -27,86% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | -215,90% | 19,38% | 100,00% | -51,76% |  | Sync | 23,00% | -170,69% | -36,06% | -2,13% | -142,31% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 62,49% | 57,83% | 61,53% | 62,19% | -94,10% |  | time\* | 75,22% | 74,74% | 75,15% | 75,18% | 22,29% |
| GFLOPS | -137,11% | -166,62% | -160,40% | -164,48% | -1145,51% |  | GFLOPS | -295,84% | -303,57% | -302,48% | -302,91% | -1144,07% |
| Memcpy AB | 0,15% | -5,02% | -1,06% | 0,31% | -35,79% |  | Memcpy AB | 0,05% | -16,71% | -2,50% | -0,86% | -160,65% |
| Memcpy C | 2,10% | 0,43% | 0,46% | 3,77% | -10,96% |  | Memcpy C | 0,41% | -3,92% | -2,41% | -3,26% | -65,17% |
| Kernel | 72,20% | 72,14% | 72,27% | 72,44% | 57,88% |  | Kernel | 77,73% | 77,71% | 77,76% | 77,75% | 80,54% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | 9,06% | 59,40% | 100,00% | 13,96% |  | Sync | -28,42% | -15,31% | -42,68% | -10,85% | -80,73% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 100,00% | 100,00% | 0,00% | 100,00% |

Durch shared memory konnten deutlich Gewinne erzielt werden, welche mit zunehmender Größe zunahmen, jedoch in den Matrixgrößen von 4096 zu 8192 keine bedeutende Steigerung mehr aufwiesen bzw. beim Kernel sogar ein leichter Rückgang. Die Werte bei den Speicherübertragungen sind unwesentlich, die übrigen Werte sind hier ebenfalls aussagelos.

### 3.2.2. Performance-Vergleich shared-memory mit CudaMallocHost

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1024 | Min | Max | Avg | Median | StdDV |  | 4096 | Min | Max | Avg | Median | StdDV |
| time\* | 55,12% | 57,44% | 55,82% | 55,67% | 83,84% |  | time\* | 74,46% | 73,60% | 74,27% | 74,40% | 33,57% |
| GFLOPS | -134,97% | -122,79% | -126,11% | -125,60% | 15,82% |  | GFLOPS | -278,74% | -291,47% | -288,68% | -290,62% | -887,33% |
| Memcpy AB | 0,42% | 32,93% | 6,87% | 1,00% | 88,56% |  | Memcpy AB | 0,28% | -6,11% | 0,78% | 1,86% | -29,70% |
| Memcpy C | -1,47% | -12,96% | 0,05% | 1,36% | -39,72% |  | Memcpy C | 0,17% | -4,37% | -0,08% | 0,21% | -97,52% |
| Kernel | 69,00% | 70,91% | 69,38% | 69,14% | 90,04% |  | Kernel | 78,51% | 77,33% | 78,35% | 78,37% | 48,88% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | 80,39% | 12,63% | 0,00% | 62,23% |  | Sync | 14,16% | 14,80% | 9,48% | 5,69% | 15,76% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2048 | Min | Max | Avg | Median | StdDV |  | 8192 | Min | Max | Avg | Median | StdDV |
| time\* | 64,40% | 62,75% | 64,02% | 64,21% | 20,77% |  | time\* | 75,76% | 75,66% | 75,73% | 75,75% | 59,58% |
| GFLOPS | -168,49% | -180,89% | -178,08% | -179,44% | -500,80% |  | GFLOPS | -310,77% | -312,57% | -312,10% | -312,44% | -585,28% |
| Memcpy AB | 0,04% | 15,19% | 0,89% | -0,96% | 34,90% |  | Memcpy AB | -0,03% | 0,44% | 0,11% | -0,61% | 14,00% |
| Memcpy C | -0,53% | 15,14% | 5,75% | 5,32% | 78,09% |  | Memcpy C | 0,12% | -0,44% | 0,02% | 0,04% | -29,09% |
| Kernel | 72,21% | 71,08% | 72,18% | 72,41% | -1,44% |  | Kernel | 77,68% | 77,55% | 77,68% | 77,70% | 67,24% |
| Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Malloc | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |
| Sync | 0,00% | 43,64% | 51,96% | 31,78% | 37,25% |  | Sync | -2,30% | -0,40% | -2,38% | -1,20% | -11,99% |
| Free | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% |  | Free | 0,00% | -9,17% | -219,91% | 0,00% | -76,86% |

Auch hier konnte durch shared memory eine deutliche Leistungssteigerung mit ähnlichen Ausprägungen erreicht werden. Eine Tendenz einer Gewinnzunahme zeigte sich hier wie oben mit einem Knick von 4096 auf 8192. Der wesentliche Anteil macht der Zugewinn direkt im Kernel aus. Die Werte im Speicherübertragungen sind auch hier unwesentlich, die übrigen Werte sind ebenfalls aussagelos.



Verwendet wurden die Average-Werte. Die Zeitvorteile nehmen zu. Dies dürfte damit zusammenhängen, dass die eigentliche Arbeit in der höherkomplexen Matrixmultiplikation stattfindet. Das leichte Abfallen im Kernel von 4096 zu 8192 ist insoweit gegenläufig.