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

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;

    float s = 0.0;
    for(i=0;i<n;++i)
                    s += a[vy+i] * b[vx+i*n];

    c[id] = s;
    }
```

Jeder Thread berechnet ein Element der Matrix. Die Matrixspalte 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 1000 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 cudaMallocHost

1024	Min	Max	Avg	Median	StdDV
time*	4,07136	6,84022	4,2739906	4,19192	0,2124928
GFLOPS	156,975	263,73	251,74995	256,146	10,63562
Memcpy AB	1,14349	2,79235	1,3025048	1,254015	0,1296619
Memcpy C	0,351616	0,848288	0,3865595	0,38376	0,0253804
Kernel	2,48848	4,24214	2,5607391	2,531055	0,1078837
Malloc	0	0	0	0	0
Sync	9,184E-41	0,045376	0,0060542	9,184E-41	0,0092427
Free	0	1,9656	0,0079705	0	0,0947497

Simple Malloc

1024	Min	Max	Avg	Median	StdDV
time*	4,54576	5,94986	4,8507635	4,769725	0,1817098
GFLOPS	180,465	236,207	221,64814	225,1155	7,8342647
Memcpy AB	1,06349	2,01261	1,3052738	1,25987	0,1105454
Memcpy C	0,872704	1,53946	0,9628799	0,944464	0,0636112
Kernel	2,48358	3,14179	2,5571385	2,53178	0,0676479
Malloc	0	0	0	0	0
Sync	0	0,064992	0,0017705	0	0,0068201
Free	4,592E-41	4,592E-41	4,592E-41	4,592E-41	1,025E-54

2048	Min	Max	Avg	Median	StdDV
time*	24,7137	27,1219	25,139674	24,99295	0,3208342
GFLOPS	316,716	347,578	341,74267	343,6945	4,2547508

2048	Min	Max	Avg	Median	StdDV
time*	26,7029	30,4027	27,515046	27,4958	0,5365273
GFLOPS	282,539	321,686	312,30703	312,4095	5,9813658

Memcpy AB	4,71318	7,00282	5,1608915	5,02725	0,2762679
Memcpy C	1,30701	2,2127	1,3599818	1,34928	0,0427196
Kernel	18,5243	19,6244	18,594692	18,5618	0,0939767
Malloc	0	0	0	0	0
Sync	9,184E-41	0,053856	0,0038639	9,184E-41	0,0089173
Free	0	5,35517	0,0108717	0	0,2094153

Memcpy AB	4,7567	8,57373	5,1742859	5,04264	0,3141374
Memcpy C	3,10643	5,44563	3,7058759	3,839105	0,3819952
Kernel	18,5081	19,3565	18,611126	18,5499	0,1447408
Malloc	0	0	0	0	0
Sync	0	0,040992	0,0035625	0	0,0079251
Free	4,592E-41	2,58995	0,026123	4,592E-41	0,2376638

4096	Min	Max	Avg	Median	StdDV
time*	163,118	175,692	166,30676	166,162	1,3753697
GFLOPS	391,135	421,287	413,23723	413,5685	3,3909494
Memcpy AB	19,3296	26,426	20,140434	19,71745	0,891597
Memcpy C	5,13632	5,97424	5,2442444	5,24064	0,0670424
Kernel	138,457	151,024	140,88606	140,903	0,9850845
Malloc	0	0	0	0	0
Sync	9,184E-41	0,132992	0,0155059	0,013472	0,0099771
Free	0	11,7296	0,2391352	0	1,4991395

4096	Min	Max	Avg	Median	StdDV
time*	170,484	184,767	175,47906	175,4235	2,1347221
GFLOPS	371,926	403,084	391,66835	391,7345	4,7412498
Memcpy AB	19,3324	30,3247	20,156469	19,65525	1,1178266
Memcpy C	12,1167	24,074	14,833525	15,41065	1,4551799
Kernel	137,7	150,659	140,45705	140,3515	1,0172193
Malloc	0	0	0	0	0
Sync	0	0,05456	0,0132438	0,01184	0,0113593
Free	4,592E-41	8,56243	0,1101977	4,592E-41	0,926086

8192	Min	Max	Avg	Median	StdDV
time*	1234,69	1290,79	1250,8837	1250,53	4,1849247
GFLOPS	425,908	445,257	439,49877	439,619	1,4603504
Memcpy AB	76,3994	98,5012	79,149555	78,29075	2,7063597
Memcpy C	20,4095	21,7081	20,530708	20,51935	0,1027108
Kernel	1137,13	1180,43	1151,1612	1151,245	2,7225433
Malloc	0	0	0	0	0
Sync	9,184E-41	0,042176	0,0166362	0,01296	0,0075229
Free	0	47,5131	1,2294351	0	7,1463988

8192	Min	Max	Avg	Median	StdDV
time*	1265,47	1303,23	1283,5868	1282,845	5,1337371
GFLOPS	421,841	434,43	428,30343	428,544	1,710706
Memcpy AB	76,6571	98,6102	79,621877	78,7785	2,6465592
Memcpy C	47,7187	65,3011	51,613802	50,59145	3,4395067
Kernel	1136,38	1160,16	1152,3156	1152,455	2,7184182
Malloc	0	0	0	0	0
Sync	0	0,0728	0,0173955	0,013696	0,0087534
Free	4,592E-41	39,2644	2,6995996	4,592E-41	9,1327474

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

Shared cudaMallocHost

1024	Min	Max	Avg	Median	StdDV
time*	3,48051	7,38285	3,6734605	3,61955	0,1746216
GFLOPS	145,437	308,501	292,78149	296,65	10,626725
Memcpy AB	1,11232	4,98406	1,2870407	1,24971	0,1541721
Memcpy C	0,354304	0,702912	0,3864022	0,386144	0,0182991
Kernel	1,91962	2,50106	1,9767759	1,9625	0,0462429
Malloc	0	0	0	0	0
Sync	9,184E-41	0,09168	0,0007139	9,184E-41	0,004837
Free	0	1,22467	0,0050619	0	0,0726717

Shared Malloc

1024	Min	Max	Avg	Median	StdDV
time*	4,04237	5,86912	4,423978	4,339745	0,2077646
GFLOPS	182,948	265,622	243,1979	247,4205	10,432011
Memcpy AB	1,02294	2,69395	1,298605	1,253245	0,1314083
Memcpy C	0,893056	2,01613	1,1158326	1,096385	0,0997476
Kernel	1,92147	2,67427	1,9846921	1,96694	0,0589361
Malloc	0	0	0	0	0
Sync	0	0,03856	0,0017038	0	0,0045784
Free	4,592E-41	4,592E-41	4,592E-41	4,592E-41	1,025E-54

2048	Min	Max	Avg	Median	StdDV
time*	20,4411	23,3727	20,745271	20,5732	0,3697147
GFLOPS	367,52	420,229	414,19326	417,529	7,082879
Memcpy AB	4,92685	7,5449	5,1597053	5,025025	0,2947296
Memcpy C	1,31293	2,36973	1,3568424	1,347955	0,0481771
Kernel	14,1228	14,9174	14,204708	14,156	0,1258028
Malloc	0	0	0	0	0
Sync	9,184E-41	0,057664	0,0025359	9,184E-41	0,0074539
Free	0	2,93158	0,0056223	0	0,1257087

2048	Min	Max	Avg	Median	StdDV
time*	22,4515	26,2594	23,398019	23,1859	0,4610628
GFLOPS	327,118	382,6	367,26063	370,481	7,0251219
Memcpy AB	4,88099	7,80067	5,1742376	5,03058	0,3023247
Memcpy C	3,20838	5,5591	3,9797984	3,893875	0,2119543
Kernel	14,1199	15,0907	14,217358	14,17645	0,1246896
Malloc	0	0	0	0	0
Sync	0	0,077344	0,0030605	0	0,0093115
Free	4,592E-41	4,592E-41	4,592E-41	4,592E-41	1,025E-54

4096	Min	Max	Avg	Median	StdDV
time*	127,482	140,875	129,71732	129,4175	1,1994893
GFLOPS	487,803	539,051	529,80796	530,9905	4,8281019
Memcpy AB	19,2993	25,0475	20,242316	19,79305	0,965899
Memcpy C	5,13472	6,22317	5,2349689	5,232915	0,0750469
Kernel	101,452	112,466	104,20467	104,1375	0,6180919
Malloc	0	0	0	0	0
Sync	9,184E-41	0,257568	0,0126716	0,01264	0,0136532

4096	Min	Max	Avg	Median	StdDV
time*	136,092	149,516	140,60192	140,3555	1,5805885
GFLOPS	459,614	504,947	488,81325	489,61	5,442879
Memcpy AB	19,3103	27,5022	20,551932	20,3162	1,0999072
Memcpy C	12,1033	20,8324	15,842569	15,595	0,7361919
Kernel	102,444	112,417	104,17511	104,123	0,6239416
Malloc	0	0	0	0	0
Sync	0	0,077056	0,013623	0,012144	0,0111741

Free	0	15,3909	0,169089	0	1,427228

Free	4,592E-41	4,592E-41	4,592E-41	4,592E-41	1,025E-54

8192	Min	Max	Avg	Median	StdDV
time*	959,774	986,062	967,80954	967,446	2,5797021
GFLOPS	557,527	572,797	568,04535	568,255	1,5092701
Memcpy AB	76,4801	96,4425	79,157434	78,56205	2,2395637
Memcpy C	20,4078	21,33	20,530134	20,52355	0,0922711
Kernel	861,778	872,218	868,08339	868,1	1,4061365
Malloc	0	0	0	0	0
Sync	9,184E-41	0,094496	0,0183201	0,01632	0,0096548
Free	0	55,8043	7,0429729	0	16,40987

8192	Min	Max	Avg	Median	StdDV
time*	990,441	1021,71	1002,8886	1003,11	4,5462573
GFLOPS	538,076	555,062	548,18355	548,0505	2,4843319
Memcpy AB	76,3972	96,4633	79,490934	78,89975	2,3358661
Memcpy C	47,5777	71,8161	55,290573	56,3573	3,6798717
Kernel	860,823	871,936	868,07629	868,25	1,6333311
Malloc	0	0	0	0	0
Sync	0	0,058368	0,0167937	0,013152	0,0084852
Free	4,592E-41	4,592E-41	4,592E-41	4,592E-41	1,025E-54

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
time*	10,44%	-14,96%	11,89%	12,11%	-16,94%
GFLOPS	13,02%	-11,65%	-13,58%	-13,78%	-35,76%
Memcpy AB	-7,52%	-38,74%	0,21%	0,46%	-17,29%
Memcpy C	59,71%	44,90%	59,85%	59,37%	60,10%
Kernel	-0,20%	-35,02%	-0,14%	0,03%	-59,48%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	30,18%	-241,95%	0,00%	-35,52%
Free	100,00%	0,00%	0,00%	100,00%	0,00%

4096	Min	Max	Avg	Median	StdDV
time*	4,32%	4,91%	5,23%	5,28%	35,57%
GFLOPS	-5,16%	-4,52%	-5,51%	-5,57%	28,48%
Memcpy AB	0,01%	12,86%	0,08%	-0,32%	20,24%
Memcpy C	57,61%	75,18%	64,65%	65,99%	95,39%
Kernel	-0,55%	-0,24%	-0,31%	-0,39%	3,16%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-143,75%	-17,08%	-13,78%	12,17%
Free	100,00%	-36,99%	-117,01%	100,00%	-61,88%

2048	Min	Max	Avg	Median	StdDV
time*	7,45%	10,79%	8,63%	9,10%	40,20%
GFLOPS	-12,10%	-8,05%	-9,43%	-10,01%	28,87%
Memcpy AB	0,91%	18,32%	0,26%	0,31%	12,06%
Memcpy C	57,93%	59,37%	63,30%	64,85%	88,82%
Kernel	-0,09%	-1,38%	0,09%	-0,06%	35,07%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-31,38%	-8,46%	0,00%	-12,52%
Free	100,00%	-106,77%	58,38%	100,00%	11,89%

8192	Min	Max	Avg	Median	StdDV
time*	2,43%	0,95%	2,55%	2,52%	18,48%
GFLOPS	-0,96%	-2,49%	-2,61%	-2,58%	14,63%
Memcpy AB	0,34%	0,11%	0,59%	0,62%	-2,26%
Memcpy C	57,23%	66,76%	60,22%	59,44%	97,01%
Kernel	-0,07%	-1,75%	0,10%	0,10%	-0,15%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	42,07%	4,36%	5,37%	14,06%
Free	100,00%	-21,01%	54,46%	100,00%	21,75%

Bei der Matrix-Größe von 1024 ist ein Leistungsgewinn von ca. 12 % Zeit und 14 % GFLOPS anfangs beachtlich. Mit zunehmender Matrix-Größe sinkt dieser deutlich. Bereits bei 2048 beträgt er nur noch knapp 9 % (Zeit und GFLOPS), bei 4096 nur noch ca. 5 % (Zeit und GFLOPS), bei 8192 fällt er auf unter 3 % (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
time*	13,90%	-25,79%	16,96%	16,60%	15,95%
GFLOPS	20,50%	-16,14%	-20,39%	-19,90%	-1,87%
Memcpy AB	-8,74%	-85,01%	0,89%	0,28%	-17,32%
Memcpy C	60,33%	65,14%	65,37%	64,78%	81,65%
Kernel	0,10%	6,48%	0,40%	0,23%	21,54%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-137,76%	58,10%	0,00%	-5,65%
Free	100,00%	0,00%	0,00%	0,00%	0,00%

4096	Min	Max	Avg	Median	StdDV
time*	6,33%	5,78%	7,74%	7,79%	24,11%
GFLOPS	-6,13%	-6,75%	-8,39%	-8,45%	11,30%
Memcpy AB	0,06%	8,93%	1,51%	2,58%	12,18%
Memcpy C	57,58%	70,13%	66,96%	66,44%	89,81%
Kernel	0,97%	-0,04%	-0,03%	-0,01%	0,94%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-234,26%	6,98%	-4,08%	-22,19%
Free	100,00%	0,00%	0,00%	100,00%	0,00%

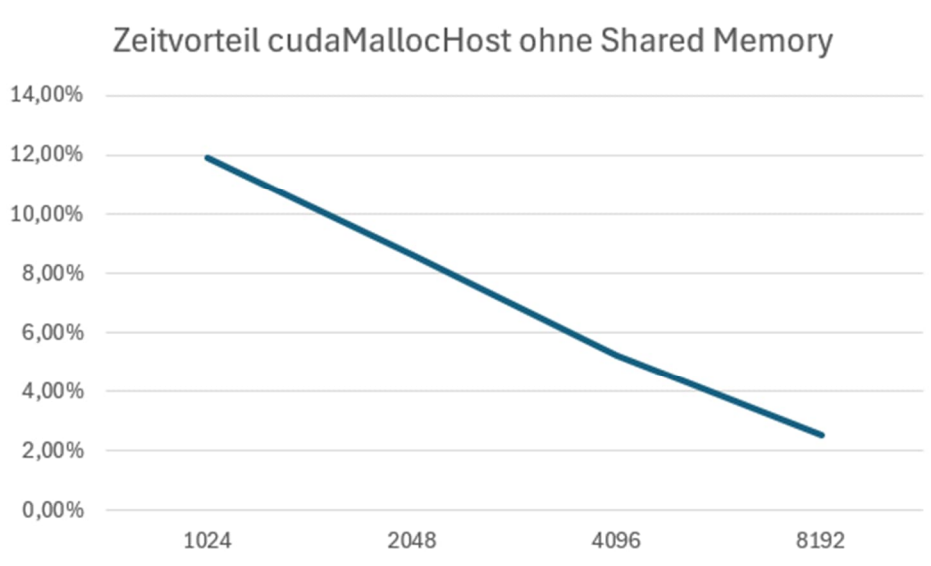
2048	Min	Max	Avg	Median	StdDV
time*	8,95%	10,99%	11,34%	11,27%	19,81%
GFLOPS	-12,35%	-9,84%	-12,78%	-12,70%	-0,82%

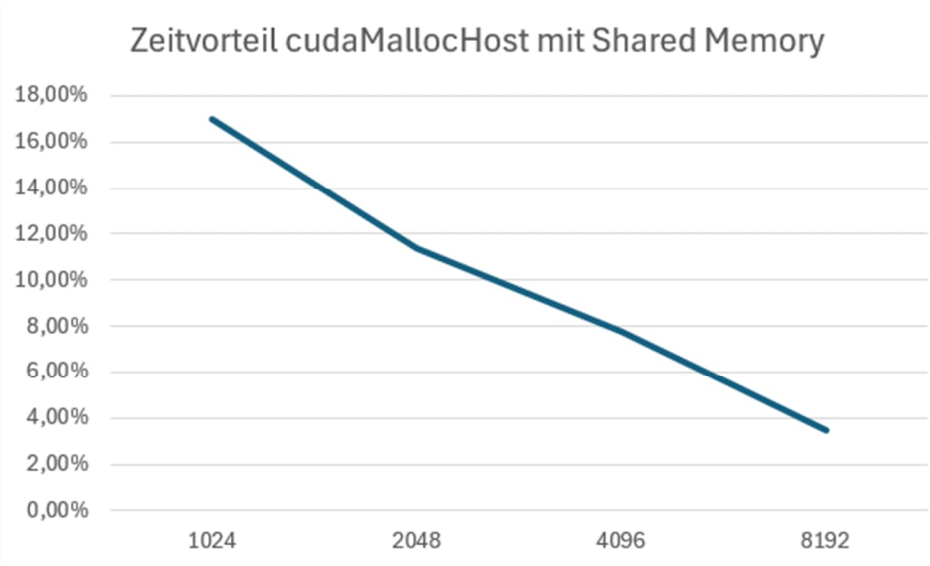
8192	Min	Max	Avg	Median	StdDV
time*	3,10%	3,49%	3,50%	3,56%	43,26%
GFLOPS	-3,61%	-3,20%	-3,62%	-3,69%	39,25%

Memcpy AB	-0,94%	3,28%	0,28%	0,11%	2,51%	Memcpy AB	-0,11%	0,02%	0,42%	0,43%	4,12%
Memcpy C	59,08%	57,37%	65,91%	65,38%	77,27%	Memcpy C	57,11%	70,30%	62,87%	63,58%	97,49%
Kernel	-0,02%	1,15%	0,09%	0,14%	-0,89%	Kernel	-0,11%	-0,03%	0,00%	0,02%	13,91%
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%	25,44%	17,14%	0,00%	19,95%	Sync	0,00%	-61,90%	-9,09%	-24,09%	-13,78%
Free	100,00%	0,00%	0,00%	100,00%	0,00%	Free	100,00%	0,00%	0,00%	100,00%	0,00%

Auch hier trägt die Reduktion in Memcpy für C im Wesentlichen zum Leistungsgewinn von jeweils ca. 65 % bei. Mit zunehmender Größe fällt der Leistungsgewinn insgesamt ab, beginnend von 17 % Zeit (GFLOPS 20 %) bis auf ca. 3,5 % (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 $O(n^3)$, mit Parallelität in den hier verwendeten Modellen auf bis zu $O(n^2)$ reduzierbar)

zurückfällt. Da die Abszisse exponentiell ist, würde die Kurve bei linearer Streckung der Abszisse leicht konvex sein und gegen einen Wert im kleineren einstelligen Prozentbereich konvergieren.

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
time*	11,07%	1,36%	8,80%	9,01%	-14,34%
GFLOPS	-1,38%	-12,45%	-9,72%	-9,91%	-33,16%
Memcpy AB	3,81%	-33,85%	0,51%	0,53%	-18,87%
Memcpy C	-2,33%	-30,96%	-15,88%	-16,09%	-56,81%
Kernel	22,63%	14,88%	22,39%	22,31%	12,88%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	40,67%	3,76%	0,00%	32,87%
Free	0,00%	0,00%	0,00%	0,00%	0,00%

4096	Min	Max	Avg	Median	StdDV
time*	20,17%	19,08%	19,88%	19,99%	25,96%
GFLOPS	-23,58%	-25,27%	-24,80%	-24,99%	-14,80%
Memcpy AB	0,11%	9,31%	-1,96%	-3,36%	1,60%
Memcpy C	0,11%	13,47%	-6,80%	-1,20%	49,41%
Kernel	25,60%	25,38%	25,83%	25,81%	38,66%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-41,23%	-2,86%	-2,57%	1,63%
Free	0,00%	100,00%	100,00%	0,00%	100,00%

2048	Min	Max	Avg	Median	StdDV
time*	15,92%	13,63%	14,96%	15,67%	14,07%
GFLOPS	-15,78%	-18,94%	-17,60%	-18,59%	-17,45%
Memcpy AB	-2,61%	9,02%	0,00%	0,24%	3,76%
Memcpy C	-3,28%	-2,08%	-7,39%	-1,43%	44,51%
Kernel	23,71%	22,04%	23,61%	23,58%	13,85%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-88,68%	14,09%	0,00%	-17,49%
Free	0,00%	100,00%	100,00%	0,00%	100,00%

8192	Min	Max	Avg	Median	StdDV
time*	21,73%	21,60%	21,87%	21,81%	11,44%
GFLOPS	-27,55%	-27,77%	-27,99%	-27,89%	-45,22%
Memcpy AB	0,34%	2,18%	0,16%	-0,15%	11,74%
Memcpy C	0,30%	-9,98%	-7,12%	-11,40%	-6,99%
Kernel	24,25%	24,84%	24,67%	24,66%	39,92%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	19,82%	3,46%	3,97%	3,06%
Free	0,00%	100,00%	100,00%	0,00%	100,00%

Durch shared memory konnten ebenfalls Gewinne in der Gesamtperformane erzielt werden, welche mit zunehmender Größe ca.9 % (Zeit) bzw. 10 % (GFLOPS) bis hin zu etwas über 20 % bzw. knapp 30 % (GFLOPS) zunahmen. Die Zunahme im Gewinn nahm jedoch mit größer werdender Matrix ab, die Gewinnkurve verflacht damit. Haupttreiber ist der Gewinn im Kernel welcher sich mit zunehmender Matrixgröße steigert, allerdings verschlechtert sich der Gewinn von der Größe 4096 auf 8192. Die Werte bei den Speicherübertragungen sind unwesentlich, die übrigen Werte sind hier ebenfalls aussagelos.

3.2.2. Performance-Vergleich shared-memory mit CudaMallocHost

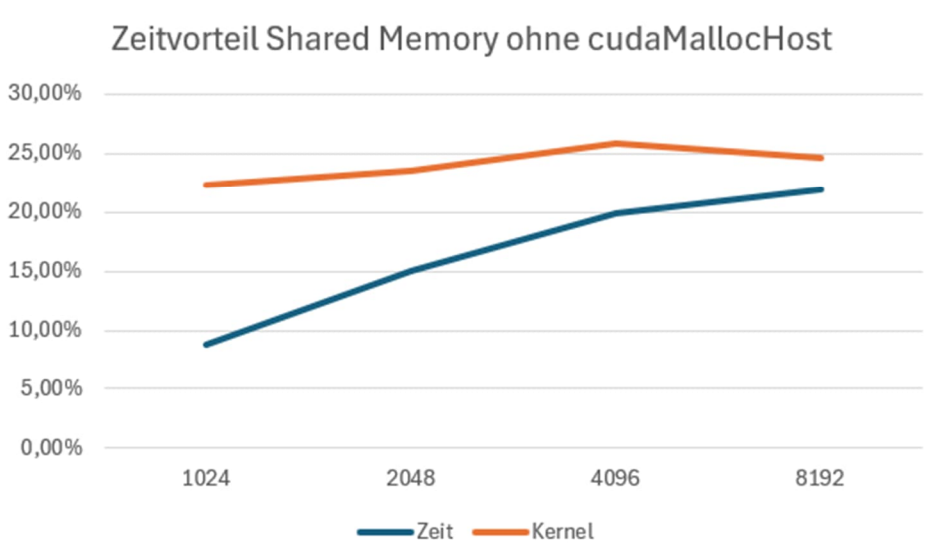
1024	Min	Max	Avg	Median	StdDV
time*	14,51%	-7,93%	14,05%	13,65%	17,82%
GFLOPS	7,35%	-16,98%	-16,30%	-15,81%	0,08%
Memcpy AB	2,73%	-78,49%	1,19%	0,34%	-18,90%
Memcpy C	-0,76%	17,14%	0,04%	-0,62%	27,90%
Kernel	22,86%	41,04%	22,80%	22,46%	57,14%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-102,05%	88,21%	0,00%	47,67%
Free	0,00%	37,69%	36,49%	0,00%	23,30%

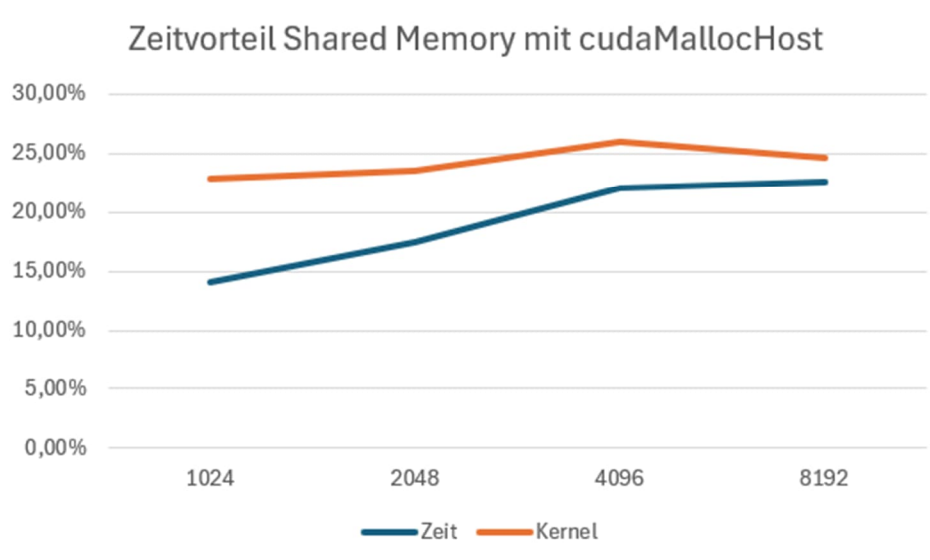
4096	Min	Max	Avg	Median	StdDV
time*	21,85%	19,82%	22,00%	22,11%	12,79%
GFLOPS	-24,71%	-27,95%	-28,21%	-28,39%	-42,38%
Memcpy AB	0,16%	5,22%	-0,51%	-0,38%	-8,33%
Memcpy C	0,03%	-4,17%	0,18%	0,15%	-11,94%
Kernel	26,73%	25,53%	26,04%	26,09%	37,25%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-93,67%	18,28%	6,18%	-36,85%
Free	0,00%	-31,21%	29,29%	0,00%	4,80%

2048	Min	Max	Avg	Median	StdDV
time*	17,29%	13,82%	17,48%	17,68%	-15,24%
GFLOPS	-16,04%	-20,90%	-21,20%	-21,48%	-66,47%
Memcpy AB	-4,53%	-7,74%	0,02%	0,04%	-6,68%
Memcpy C	-0,45%	-7,10%	0,23%	0,10%	-12,78%
Kernel	23,76%	23,99%	23,61%	23,74%	-33,87%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-7,07%	34,37%	0,00%	16,41%
Free	0,00%	45,26%	48,28%	0,00%	39,97%

8192	Min	Max	Avg	Median	StdDV
time*	22,27%	23,61%	22,63%	22,64%	38,36%
GFLOPS	-30,90%	-28,64%	-29,25%	-29,26%	-3,35%
Memcpy AB	-0,11%	2,09%	-0,01%	-0,35%	17,25%
Memcpy C	0,01%	1,74%	0,00%	-0,02%	10,16%
Kernel	24,21%	26,11%	24,59%	24,59%	48,35%
Malloc	0,00%	0,00%	0,00%	0,00%	0,00%
Sync	0,00%	-124,05%	-10,12%	-25,93%	-28,34%
Free	0,00%	-17,45%	-472,86%	0,00%	-129,62%

Auch hier konnte durch shared memory eine Leistungssteigerung mit ähnlichen Ausprägungen erreicht werden. Eine Tendenz einer Gewinnzunahme zeigte sich hier wie oben. Der wesentliche Anteil macht der Zugewinn direkt im Kernel aus, ebenfalls im Schritt von den Matrixgrößen 4096 auf 8192 zeigt sich eine Gewinnabnahme im Kernel. 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.