Einführung in das Roofline Performance Modell

Schwerpunkt: Anwendung auf NVIDIA GPUs

Vorlesung: Data Science – High Performance Computing
Simon Grimm, 12. Mai 2025

1 Einleitung

Das **Roofline Performance Model** ist ein einfaches Analysewerkzeug zur Bewertung der Effizienz von Algorithmen auf modernen Rechnerarchitekturen. Es erlaubt, die maximal erreichbare Performance unter Berücksichtigung der Speicherbandbreite und Rechenleistung grafisch darzustellen. Besonders im Kontext von GPU-basierten Architekturen – wie NVIDIA GPUs – ist das Modell eine wertvolle Orientierungshilfe zur Identifikation von Engpässen und Optimierungspotenzialen.

2 Grundlagen des Roofline-Modells

Das Modell basiert auf zwei Hauptkomponenten:

- **Rechenleistung** P_{peak}: Maximale Anzahl an FLOPs pro Sekunde, die ein Prozessor leisten kann.
- **Speicherbandbreite** B: Maximale Rate, mit der Daten aus dem Speicher geholt werden können (in Bytes/s).

Die **relevante Kennzahl** eines Algorithmus ist die *rechnerische Intensität*:

$$I = \frac{\text{Anzahl der FLOPs}}{\text{Bewegte Bytes}} \quad [\text{FLOP/Byte}]$$

Die erreichbare Performance ist somit durch:

$$P = \min(P_{peak}, I \cdot B)$$

Diese Gleichung erzeugt ein Diagramm mit zwei Zonen:

- Oberhalb der Roofline: **Unerreichbarer Bereich**
- Links unter der Roofline: **Memory Bound** – Performance durch Speicherbandbreite begrenzt.
- Rechts unter der Roofline: **Compute Bound** – Performance durch maximale FLOPs begrenzt.

Ein Beispiel eines Roofline-Diagramms ist in Figure 1 gegeben. In dieser Grafik ist die x-Achse die Rechenintensität (FLOP/Byte) und die y-Achse die erreichte Leistung (FLOP/s). Der Knickpunkt markiert die Grenze zwischen speicher- und rechenlimitierten Regionen. An diesem Punkt wird die Hardware maximal ausgenutzt. Das erreichen des Knickpunktes kann zwar als Optimierungsziel angesehen werden, wird in der Realität allerdings selten erreicht, und ist auch nicht immer sinnvoll.

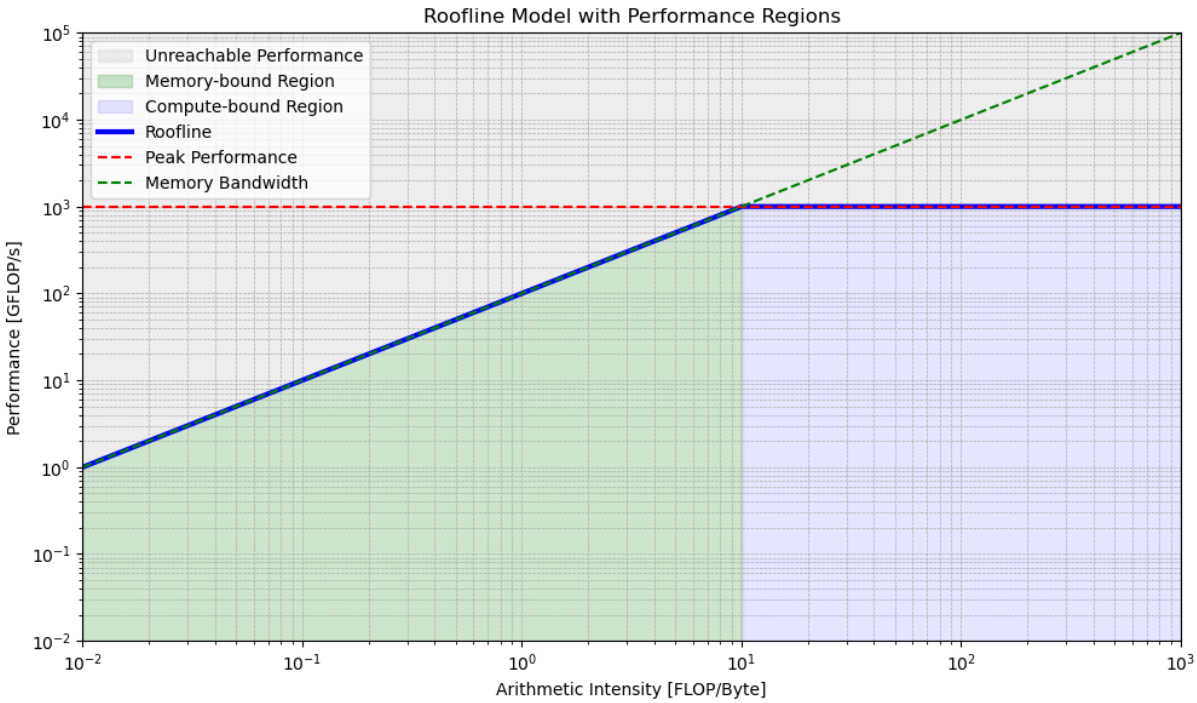

Figure 1: Schematische Darstellung eines Roofline-Modells (Platzhalter für NVIDIA GPU).

3 Das Roofline-Modell auf NVIDIA GPUs

NVIDIA GPUs wie die verfügen über mehrere Recheneinheiten:

- **CUDA-Kerne** für FP32/FP64-Operationen. Das Performance Verhältnis von single precision Performance FP32 zu double precision Performance FP64 variiert stark nach GPU Typ. Während Data-Center (Tesla) GPU ein Verhältnis von 2:1 bieten, beträgt es bei Desktop GPUs (GeForce) nur 32:1 oder gar 64:1.
- **Tensor Cores** für hochparallele FP16/BF16-Matrizenoperationen.
- **Speicherhierarchie**: Register, Shared Memory, L1/L2 Cache, Global Memory. Die Verschiedene Speichertypen haben stark unterschiedliche Bandbreiten.

Die Rechenleistung kann dabei mehrere **TFLOP/s** (TeraFLOP/s) betragen, während die Speicherbandbreite durch den HBM2/3-Speicher limitiert ist. Das Roofline-Modell ermöglicht es, diese verschiedenen Limits grafisch darzustellen und zu vergleichen.

In Tabelle 1 sind einige Hardwareparameter von NVIDIA GPUs aufgeführt.

Beispiele von theoretischen Hardwareparameter

Modell	FP32 Peak (GFLOP/s)	FP64 Peak (GFLOP/s)	Bandbreite (GB/s)
H100	51200	25600	2039
Tesla T4	8100	130	320
RTX 4070 Ti	39540	554	504

Table 1: Theoretische Hardwareparameter ausgewählter NVIDIA GPUs.

4 Anwendung des Roofline Models

4.1 Hardwareparameter

Der erste Schritt zur Anwendung des Roofline Moldes besteht darin, die theoretischen Hardwareparameter zu sammeln. Diese sind häufig von den Herstellern publiziert, können aber auch manchmal schwierig zu finden sein. Für NVIDIA GPUs listed die Wikipedia Seite fast alle Parameter zusammen:

`https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units`

4.2 Performance Messung

Der zweite Schritt ist zu messen wie lange ein bestimmter Code Bereich (z.B. CUDA kernel) für die ausführung braucht. Dazu kann im Code selbst eine Zeitmetrik implementiert werden. z.B. mit CudaEvents.

Genauer ist allerdings, einen Profiler zu verwenden. NVIDIA bietet dazu das NVIDIA Nsight Systems Paket an. Der Profiler kann folgendermassen benutzt werden:

```
nsys nvprof ./<application>
```

4.3 Zählen der FLOPs und bytes

Der nächste Schritt besteht darin zu zählen, wie viele Floating Point Operations und wie viele Speicherzugriffe verwendet wurden. Dazu schauen wir uns das folgende Beispiel von einem CUDA kernel an:

```
1 __global__ void addTest_kernel(double *a_d, double *b_d, double *c_d, int N){
2
3       int id = threadIdx.x * blockDim.x + blockIdx.x;
4
5       if(id < N){
6               c_d[id] += 2.0 * a_d[id] + 1.5 * b_d[id] + 0.8;
7       }
8 }
```

Für die Anzahl FLOPs werden nur die Floating Point Operationen in der Zeile 6 berücksichtigt. In diesem Fall sind das:

- +=: 1 FLOP
- *: 1 FLOP
- +: 1 FLOP

Insgesamt also 5 FLOPs.

Für die Anzahl bewegter bytes werden nur Zugriffe auf den globalen Speicher berücksichtigt. In diesem Fall sind das drei double precision Werte pro Element. Jede double precision Zahl braucht 8 bytes, daher sind es Total 24 bytes.

Die Arithmetische Intensität ist daher

$$I = \frac{5\text{FLOP}}{24\text{byte}} = 0.208\text{FLOP/byte} \tag{1}$$

4.4 Eintragen der Werte

Schlussendlich können die Werte von der arithmetischen Intensität und den gemessenen GFLOP/s im Diagramm eingetragen werden.

4.5 Automatische Analyse

Alternativ zu den Schritten zuvor kann eine Roofline-Analyse auch automatisch durchgeführt werden. Dazu bietet NVIDIA das Tool Nsight Compute an. Dieses benötigt jedoch root Zugriff auf das System, was nicht immer vorhanden ist. Eine Roofline analyze kann mit den folgenden Schritten erstellt werden:

- ./ncu-ui: starten von NSIGHT COMPUTE
- click auf "start activity"
- "Application Executable" angeben
- "Output File" angeben
- "Relay Mode Kernel" auswählen
- "Graph Profiling Node" auswählen
- click "Launch"
- click "details"
- bei "GPU Speed of Light Throughput" "Roofline Double Precision" auswählen.

Für CPUs gibt es z.B. das Intel Advisor Tool, dass ebenfalls Roofline-Analysen durchführen kann.

5 Vergleich: Roofline-Modell auf CPU und GPU

Das Roofline-Modell lässt sich auf verschiedene Architekturen anwenden. Während der Grundgedanke gleich bleibt, unterscheiden sich die Rooflines stark je nach Hardware:

- **CPUs** haben relativ hohe Speicherbandbreite pro Kern, aber geringe Rechenleistung.
- **GPUs** bieten extreme Rechenleistung, jedoch kann der globale Speicher schnell zum Flaschenhals werden.

Eigenschaft	**CPU (z.B. AMD Threadripper**	**GPU (z.B. NVIDIA H100)**
Rechenleistung	bis zu 3 TFLOP/s	bis zu 900 TFLOP/s (FP16)
Speicherbandbreite	bis 300 GB/s	bis zu 3.0 TB/s
Kerntypen	wenige, komplexe Kerne	viele, einfache Kerne
Vektorisierung	AVX2/AVX-512	massive SIMD + Tensor Cores

Table 2: Typische Unterschiede zwischen CPU- und GPU-Rooflines

Implikationen für Entwickler

Während CPUs bei speicherintensiven Algorithmen durchaus effizient sein können, ist bei GPUs die Hauptstrategie die Maximierung der Rechenintensität. Entwickler müssen daher:

- auf GPUs stärker auf Coalescing und Caching achten,
- datenintensive Operationen minimieren,
- und Tensor Cores gezielt einsetzen.

6 Zusammenfassung

Das Roofline-Modell bietet eine leistungsstarke und zugleich einfache Möglichkeit, Engpässe in GPU-basierten Anwendungen zu identifizieren. Es zeigt klar auf, ob ein Algorithmus durch Speicher oder Rechenleistung limitiert ist und ermöglicht eine fundierte Strategie zur Performance-Optimierung.

Gerade in einem Umfeld, in dem Modellgrößen und Datenmengen stetig wachsen, ist ein tiefes Verständnis solcher Modelle entscheidend für effiziente, skalierbare Software.

Allerdings gilt zu beachten, dass das Roofline Modell stark vereinfacht ist. Für eine gute optimierung braucht es weitere Strategien. Auch ist das erreichen der Peak Performance nicht immer wünschenswert. Oft kann eine Beschleunigung der Anwendung durche bessere und komplexere numerische Methoden erreicht werden, die aber auf Kosten der Peak Performance gehen und somit im Roofline-Diagramm schlechter erscheinen.