Fachpraktikum Parallele Programmierung, Wintersemester 2024/25

3. Aufgabe: Histogramm

*Johannes Becker, Norbert Baumstark, Stefan Butz*

# Aufgabenstellung

Es sollten Routinen zur Erstellung eines Histogramms auf einer GPU implementiert und hinsichtlich ihrer Laufzeit untersucht werden. Dabei sollte gezählt werden, wie oft einzelne Bytewerte in einem Array von Bytes vorkommen. Das Hochzählen der Anzahlen in den Bins des Histogramms sollte durch atomare Operationen erfolgen. Es waren zwei Varianten zu untersuchen: zum einen das Hochzählen direkt im globalen Speicher; zum anderen sollten zunächst mehrere Instanzen der Bins im Shared Memory angelegt und diese erst abschließend zu einem einzigen Histogramm akkumuliert werden.

Die Eingabedaten waren nach Annahme Bytewerte zwischen 1 und 128. Es waren zwei Varianten des Histogramms zu erstellen, eine mit 128 Bins für die einzelnen Bytewerte sowie eine weitere mit jeweils einem Bin für die Buchstaben ‚A‘ bzw. ‚a‘ bis ‚Z‘ bzw. ‚z‘ und einem weiteren Bin für alle anderen Zeichen, d.h. insgesamt 27 Bins.

# Verzeichnisstruktur

Auf der obersten Verzeichnisebene befinden sich zwei Dateien, nämlich ein Makefile sowie ein Python-Skript perform\_measurements.py, welches alle Messungen ausführt.

Es gibt folgende Unterverzeichnisse:

* src/ enthält den Quellcode.
* analysis/: enthält ein Jupyter-Notebook zur Auswertung der Messergebnisse.
* doc/ enthält diese Dokumentation.
* input\_data/ enthält die Testdaten, die der Aufgabenstellung beigefügt waren.
* nvidia/ enthält das mit den CUDA-Entwicklungstools mitgelieferte Beispielprogramm deviceQuery, welches Geräteinformationen über die GPU ausgibt.
* bin/ wird ggf. von make angelegt und dient als Zielverzeichnis für die Objektdateien sowie die ausführbaren Dateien deviceQuery und histogram.
* measurements/ wird ggf. von perform\_measurements.py angelegt und dient der Ablage der Geräteinformationen und Messergebnisse.

In den abgegebenen Dateien sind die Messergebnisse enthalten, die dieser Dokumentation zugrunde liegen.

# Aufruf

Nach Erstellen der ausführbaren Dateien durch

make

startet man durch

./perform\_measurements.py

die Messungen. Gegebenenfalls sind die Parameter im Makefile auf die verwendete Architektur und im Python-Skript der Pfad zum Python-Interpreter anzupassen.

Das Python-Skript ermittelt zunächst die Geräteinformationen durch Aufruf von bin/deviceQuery und ruft dann bin/histogram mit verschiedenen Kommandozeilenargumenten auf, um verschiedene Szenarien zu messen. Insbesondere wird bin/histogram angewiesen, pro Szenario und Kernel 100 Messungen vorzunehmen. Neben den der Aufgabenstellung beiliegenden Testdaten umfassen die gemessenen Datengrößen den Bereich von Bytes bis GiB in Zweierpotenzen. Die Ausführungsdauer des Python-Skripts ist also erheblich. Sollen aus Gründen der Zeitersparnis weniger als 100 Durchläufe je Szenario erfolgen, kann die Konstante N\_RUNS im Python-Skript entsprechend geändert werden.

Die Geräteinformationen werden als Textdatei, die Messergebnisse im JSON-Format in einem Unterverzeichnis von measurements/ gespeichert. Das angelegte Unterverzeichnis trägt einen Zeitstempel als Namen.

Einzelne (wiederholte) Messungen können direkt durch Aufruf des Binarys bin/histogram mit entsprechenden Kommandozeilenargumenten vorgenommen werden.

Beispiele:

bin/histogram ./input\_data/test.txt

führt mit jedem der vier Kernels eine Messung für die Beispieldatei aus, wobei 128 Bins verwendet werden.

bin/histogram -- 10500 asl 7

führt für pseudo-zufällige Daten der Grösse 10500 Bytes mit den Kernels histogram\_atomic\_private (‘a’) und histogram\_atomic\_private\_stride (‘s’) jeweils 7 Messungen aus, wobei 27 Bins (‘l’) verwendet werden.

Die Usage-Information, die nach Aufruf von bin/histogram ohne Argumente erscheint (sie findet sich auch am Anfang von src/histogram.cu), erklärt die Bedeutung der möglichen Kommandozeilenargumente.

# Der Programmcode – histogram.cu

#### Kernels

Die Kernels sind als Template-Funktionen ausgestaltet. Für den Template-Parameter Mapping kann eine struct übergeben werden, welche die Zuordnung von Zeichencodes zu Bins definiert. Wir verwenden zwei Mappings, entsprechend den beiden Aufgabenteilen.

Für Aufgabenteil a werden die Zeichencodes 1 bis 128 den Bins 0 bis 127 zugeordnet:

struct Mapping128 {

constexpr static size\_t numBins = 128;

constexpr static \_\_host\_\_ \_\_device\_\_ unsigned char map(

unsigned char c

) {

return (c - 1u) & 0x7f;

}

};

Für Aufgabenteil b werden die Buchstaben ‘A’/’a’ bis ‘Z’/’z’ (Zeichencodes 65 bzw. 90 bis 97 bzw. 122) den Bins 1 bis 26 zugeordnet; alle übrigen Zeichencodes dem Bin 0:

struct MappingLetter {

constexpr static size\_t numBins = 27;

constexpr static \_\_host\_\_ \_\_device\_\_ unsigned char map(

unsigned char c

) {

c = (c & 0xdf) - 64u;

return c & (0u - (c <= 26u));

}

};

Es werden vier Kernels untersucht.

*histogram\_kernel\_atomic\_global (*«*global*»)

Die «Baseline» ist ein einfacher Kernel, bei dem jeder Thread genau ein Zeichen bearbeitet und dann den entsprechenden Bin im globalen Speicher hochzählt:

template<typename Mapping>

\_\_global\_\_ void histogram\_kernel\_atomic\_global(

unsigned char \* input, BinType \* bins, size\_t numElements

) {

unsigned int idx = blockIdx.x \* blockDim.x + threadIdx.x;

if (idx >= numElements) return;

unsigned char c = input[idx];

atomicAdd(&bins[Mapping::map(c)], 1);

}

Beim Aufruf des Kernels wird berechnet, wie viele Blöcke benötigt werden, in Abhängigkeit von der Eingabegrösse:

constexpr size\_t nThreadsPerBlock = 256;

// …

dim3 dimGrid(

(numElements + nThreadsPerBlock - 1) / nThreadsPerBlock, 1, 1

);

dim3 dimBlock(nThreadsPerBlock, 1, 1);

histogram\_kernel\_atomic\_global<Mapping> <<<dimGrid, dimBlock>>> (

input, bins, numElements

);

*histogram\_kernel\_atomic\_private (*«*private*»)

Der zweite Kernel unterscheidet sich vom ersten nun in der Verwendung von Shared Memory. Jeder Block erhält eine private Instanz des Bins-Arrays im Shared Memory. Diese Instanzen werden abschliessend im globalen Speicher aggregiert.

template <typename Mapping>

\_\_global\_\_ void histogram\_kernel\_atomic\_private(

unsigned char \* input, BinType \* bins, size\_t numElements

) {

unsigned int idx = blockIdx.x \* blockDim.x + threadIdx.x;

\_\_shared\_\_ BinType sBins[Mapping::numBins \* sizeof(BinType)];

for (

unsigned int t = threadIdx.x; t < Mapping::numBins;

t += blockDim.x

) {

sBins[t] = 0;

}

\_\_syncthreads();

if (idx < numElements) {

unsigned char c = input[idx];

atomicAdd(&sBins[Mapping::map(c)], 1);

}

\_\_syncthreads();

for (

unsigned int t = threadIdx.x; t < Mapping::numBins;

t += blockDim.x

) {

atomicAdd(&bins[t], sBins[t]);

}

}

Der Aufruf erfolgt entsprechend wie beim ersten Kernel – es werden so viele Blöcke angefordert, wie für die Eingabe benötigt werden.

*histogram\_kernel\_atomic\_private\_stride (*«*private\_stride*»)

Beim dritten Kernel ist die Anzahl der Threads fest, unabhängig von der Eingabegrösse. Pro Schritt («stride») wird ein Block von nebeneinanderliegenden Zeichen bearbeitet, bis die Eingabe abgearbeitet ist. Es wird wieder mit privaten Instanzen des Bins-Arrays für die einzelnen Blöcke gearbeitet.

template <typename Mapping>

\_\_global\_\_ void histogram\_kernel\_atomic\_private\_stride(

unsigned char \* input, BinType \* bins, size\_t numElements

) {

unsigned int idx = blockIdx.x \* blockDim.x + threadIdx.x;

// initialisiere Array im shared memory mit 0

\_\_shared\_\_ BinType sBins[Mapping::numBins \* sizeof(BinType)];

for (

unsigned int t = threadIdx.x; t < Mapping::numBins;

t += blockDim.x

) {

sBins[t] = 0;

}

\_\_syncthreads();

{

// baseLimit ist die kleinste Zeichenposition, ab der ein

// ab baseLimit beginnender Stride genau am letzten Zeichen

// des Inputs endet oder über den Input hinausragt.

// Im Prinzip wäre das die letzte Iteration der Schleife. Da

// allerdings hier durch eine if-Abfrage geprüft werden

// müsste, ob idx noch innerhalb des Inputs liegt, spart es

// etwas Zeit, den letzten Stride separat zu behandeln.

int stride = blockDim.x \* gridDim.x;

size\_t baseLimit = numElements >= stride ?

numElements - stride : 0;

size\_t base = 0;

for (; base < baseLimit; base += stride) {

unsigned char c = input[base + idx];

atomicAdd(&sBins[Mapping::map(c)], 1);

}

if (base + idx < numElements) {

unsigned char c = input[base + idx];

atomicAdd(&sBins[Mapping::map(c)], 1);

}

}

\_\_syncthreads();

for (

unsigned int t = threadIdx.x; t < Mapping::numBins;

t += blockDim.x

) {

atomicAdd(&bins[t], sBins[t]);

}

}

Die Anzahl Threads ist nun unabhängig von der Eingabegröße:

constexpr size\_t nThreadsPerBlock = 256;

// …

dim3 dimGrid(1024, 1, 1);

dim3 dimBlock(nThreadsPerBlock, 1, 1);

histogram\_kernel\_atomic\_private\_stride<Mapping> <<<dimGrid, dimBlock>>> (

input, bins, numElements

);

Anmerkung: Die Anzahl Blöcke wurde mittels «Trial and Error» und aufgrund von allgemeinen Empfehlungen aus dem Web als 1024 festgelegt. Die gemessenen Ausführungszeiten scheinen nicht wesentlich von der Anzahl der Blöcke abzuhängen, sofern diese nicht zu klein wird. Bei einer grossen Anzahl Blöcken wird der Kernel wohl äquivalent zum Kernel ohne Stride. Die optimalen Parameterwerte sind wahrscheinlich geräteabhängig und nur mittels eines Skripts sinnvoll zu ermitteln.

*histogram\_kernel\_atomic\_global\_stride (*«*global\_stride*»)

Der vierte Kernel behält das Stride-Konzept bei, verzichtet jedoch auf die Verwendung von Shared Memory – die Bins werden direkt im globalen Speicher hochgezählt:

template <typename Mapping>

\_\_global\_\_ void histogram\_kernel\_atomic\_global\_stride(

unsigned char \* input, BinType \* bins, size\_t numElements

) {

unsigned int idx = blockIdx.x \* blockDim.x + threadIdx.x;

{

int stride = blockDim.x \* gridDim.x;

size\_t baseLimit = numElements >= stride ?

numElements - stride : 0;

size\_t base = 0;

for (; base < baseLimit; base += stride) {

unsigned char c = input[base + idx];

atomicAdd(&bins[Mapping::map(c)], 1);

}

if (base + idx < numElements) {

unsigned char c = input[base + idx];

atomicAdd(&bins[Mapping::map(c)], 1);

}

}

}

Der Aufruf erfolgt wie beim dritten Kernel – die Anzahl der Threads ist unabhängig von der Eingabegrösse.

Der Kernel «global\_stride» wird bei den nachfolgenden Ausführungen in diesem Dokument jeweils an zweiter Stelle, direkt nach «global», aufgeführt, da sich die Messergebnisse von «global» und «global\_stride» meist kaum unterscheiden.

# Vorgehen bei den Messungen

Das Programm allokiert zunächst (einmalig) Host- und Device-Speicher. Danach wird für *jeden* der zu messenden Kernels Folgendes durchgeführt:

* Um Verfälschungen der Messungen durch niedrigeren Takt der GPU im Idle-Zustand auszuschliessen, werden zunächst 200 Warmup-Runs mit einer Datengrösse von 100 MiB durchgeführt, d.h. der jeweilige Kernel wird 200mal aufgerufen.
* Danach werden die mittels Kommandozeile spezifizierte Anzahl von Malen
  + die Eingabedaten vom Host- zum Device-Speicher transferiert,
  + der Kernel ausgeführt, d.h. das Histogramm erstellt,
  + das Histogramm vom Device- in den Host-Speicher übertragen.

Die Zeiten für jeden dieser drei Schritte werden mittels CUDA-Events gemessen.

* Das Ergebnis der letzten Wiederholung wird durch Vergleich mit einem auf der CPU erstellten Histogramm auf Korrektheit geprüft.

Das Programm gibt die Aufrufparameter sowie die Messergebnisse im JSON-Format auf stdout aus.

# Eingabedaten

Das Programm erlaubt sowohl das Einlesen einer Textdatei als auch die Generierung synthetischer Daten beliebiger Grösse. Da zu erwarten ist, dass die Laufzeit von der Anzahl auftretender Konflikte beim atomaren Zugriff auf den Speicher abhängt (was sich durch die Messwerte bestätigt), verwenden wir zweierlei synthetische Daten:

* Pseudozufällige Daten (Werte zwischen 1 und 128), die durch einen Lehmer-Zufallszahlengenerator erzeugt werden. Der Seed des Zufallszahlengenerators ist fest, d.h. jeder Durchlauf arbeitet mit den gleichen Daten. Die Implementierung des Zufallszahlengenerators entstammt der Wikipedia. Es ist nicht anzunehmen, dass die Zufallszahlen von guter Qualität sind – allerdings sind sie für den vorliegenden Zweck wohl ausreichend, da es nur darum geht, sehr kurze, systematische Muster zu vermeiden. Die Zahlenwerte von 1 bis 128 sind in den Daten annähernd gleichverteilt. Bei generierten Zahlenwerten hat jeder der 128 Zahlenwerte eine relative Häufigkeit von 0.7817 %, und der Unterschied in den relativen Häufigkeiten des seltensten und des häufigsten Zahlenwerts beträgt Prozentpunkte.
* Gleichförmige, d.h. konstante Daten, die nur aus dem Zeichen ‘a’ bestehen.

# Hardware

Die Messungen wurden auf folgenden Geräten durchgeführt:

* NVIDIA Jetson Xavier NX 16 GB in einem Seeed Studio reComputer J2022;  
  GPU gemäss Datenblatt: 384-core NVIDIA Volta GPU with 48 Tensor Cores,  
  CPU gemäss Datenblatt: 6-core NVIDIA Carmel ARM v8.2 64-bit CPU 6MB L2 + 4MB L3.
* NVIDIA Tesla V100-SXM2-32GB,

funkel.fernuni-hagen.de.

Da es einfach durchzuführen war, wurde der CUDA-Code mit dem Tool hipify-perl von AMD nach HIP konvertiert und mittels hipcc für AMD-GPUs kompiliert. Es waren keine wesentlichen Anpassungen am konvertierten Code erforderlich. Zum Vergleich mit den GPUs von NVIDIA wurden die Messungen dann auch auf folgender Grafikkarte durchgeführt:

* AMD Radeon RX 6800 XT, 16 GB, auf einem AMD Ryzen 9 7950X mit 64 GB Hauptspeicher.

Der Code für AMD ist der Abgabe *nicht* beigefügt, findet sich aber auf Github unter

<https://github.com/sbutz/parallel-programming/tree/aa9be8dc6a4372c00e3cefb66a6b3a3e101f62ff/000_johannes/03_amd> .

Die Ausgaben von deviceQuery für die einzelnen Geräte finden sich in Anhang A.

# Messergebnisse für die Eingabedaten aus der Beispieldatei

Die Beispieldatei ./input\_data/test.txt enthält 10532866 Zeichen, ist also rund 10 MiB gross. Abbildung 1 (sowie die Tabelle in Anhang B) zeigen die Häufigkeitsverteilung der einzelnen Zeichencodes.

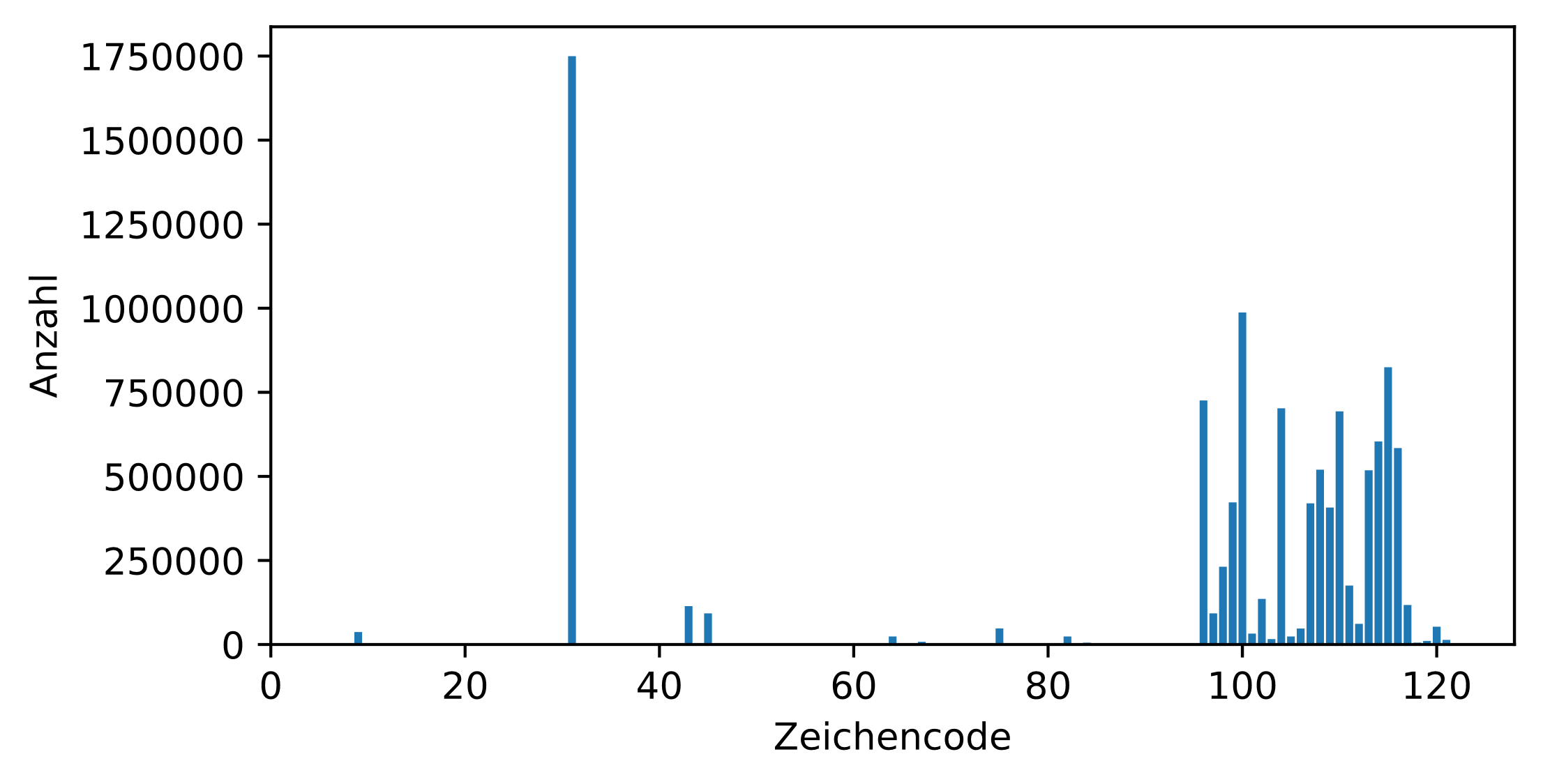


Abbildung 1. Häufigkeiten der einzelnen Zeichencodes in der Beispieldatei.

Die Tabellen in Anhang C zeigen eine Zusammenfassung der Messergebnisse:

* die Zeit für den Datentransfer vom Host zum Device,
* die Ausführungszeit auf dem Device,
* die Zeit für den Datentransfer vom Device zum Host,
* die Summe dieser drei Zeiten («Gesamtzeit»).

Es sind jeweils die minimale Zeit, das 10%-Quantil, der Median, das 90%-Quantil und die maximale Zeit angegeben.

Es fällt auf, dass die Transferzeiten teilweise eine recht grosse Variation aufweisen. Abbildung 2 zeigt den zeitlichen Verlauf der Zeiten für den Transfer vom Host zum Device über die jeweils 100 Wiederholungen.

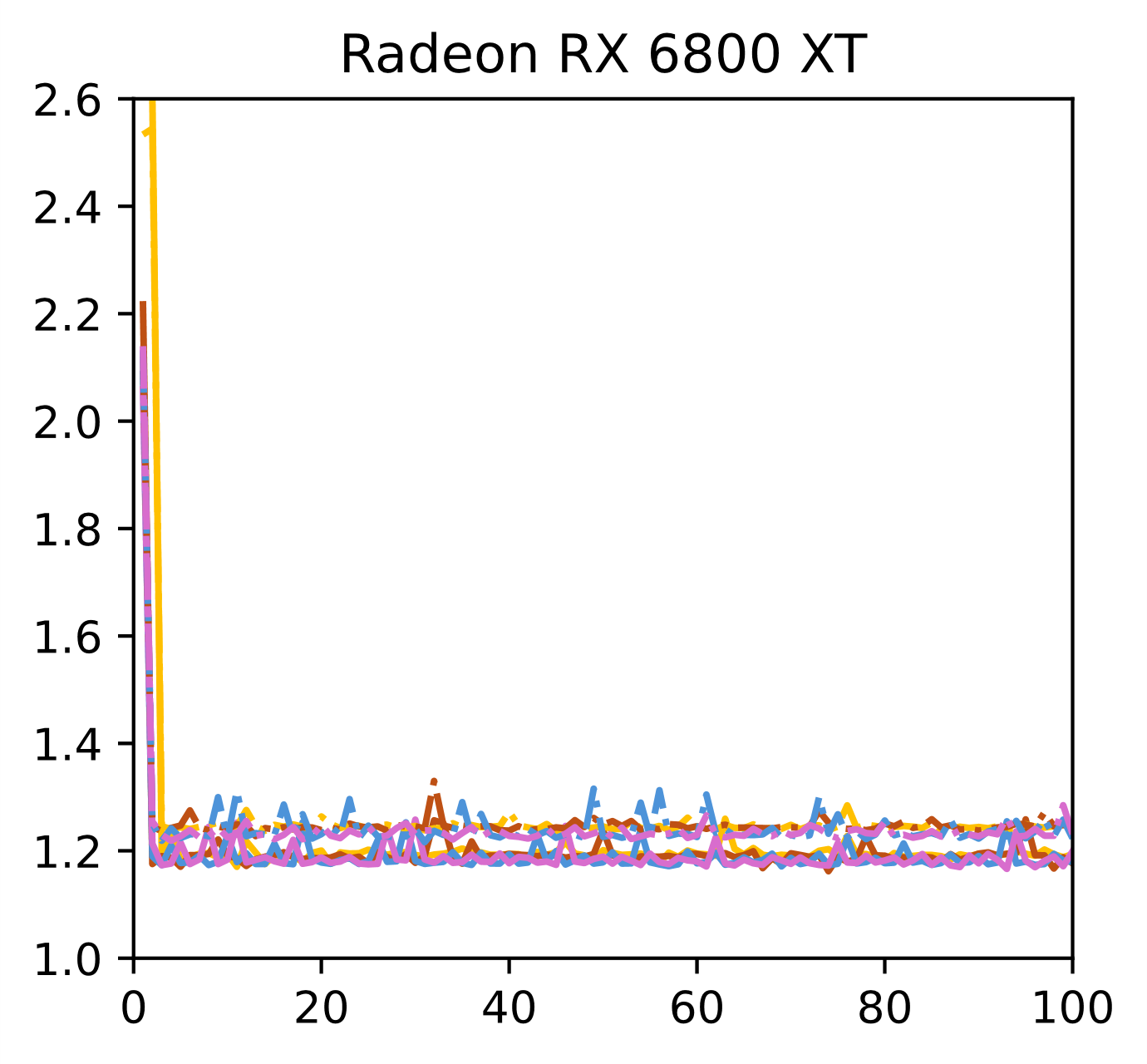
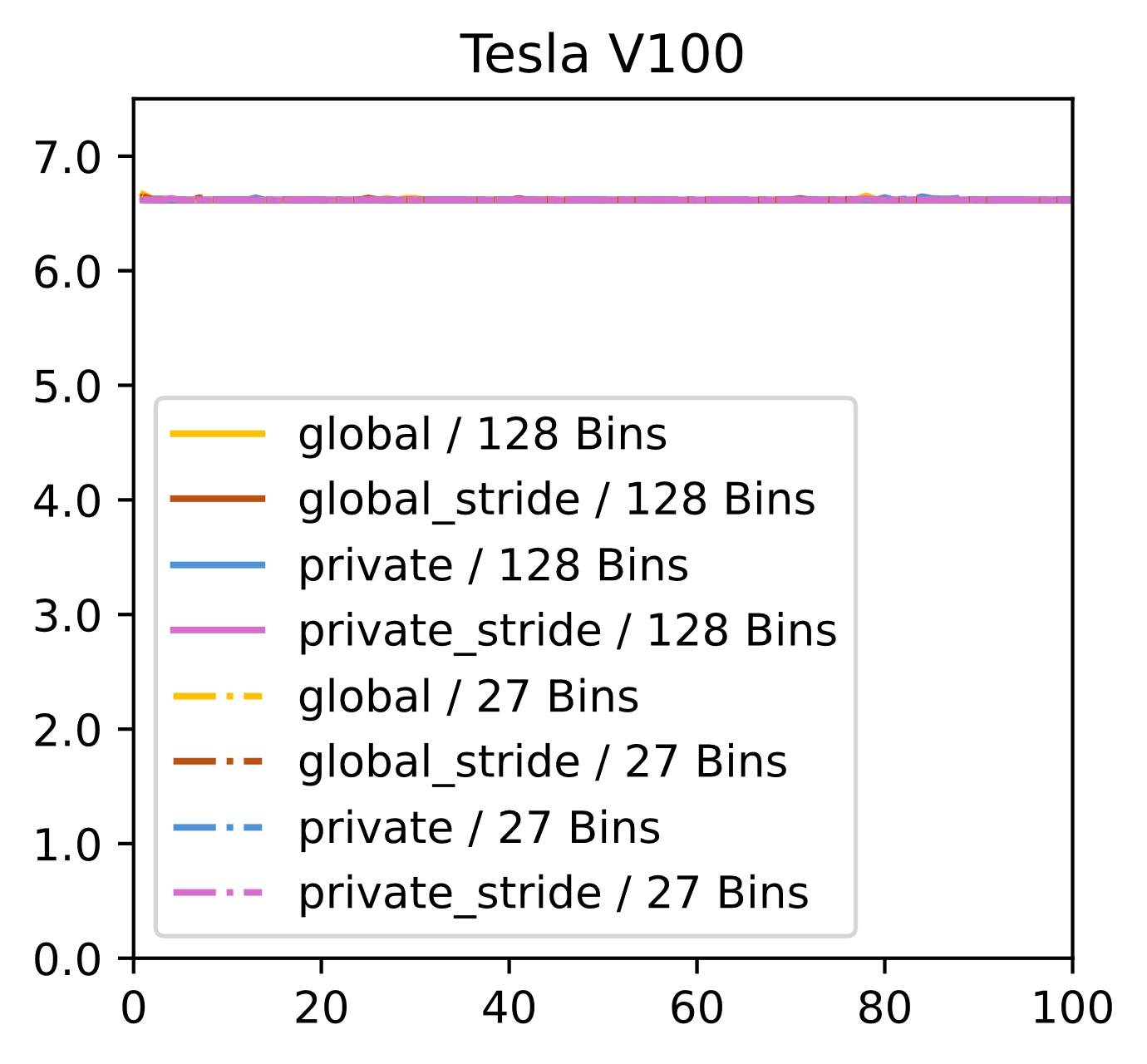
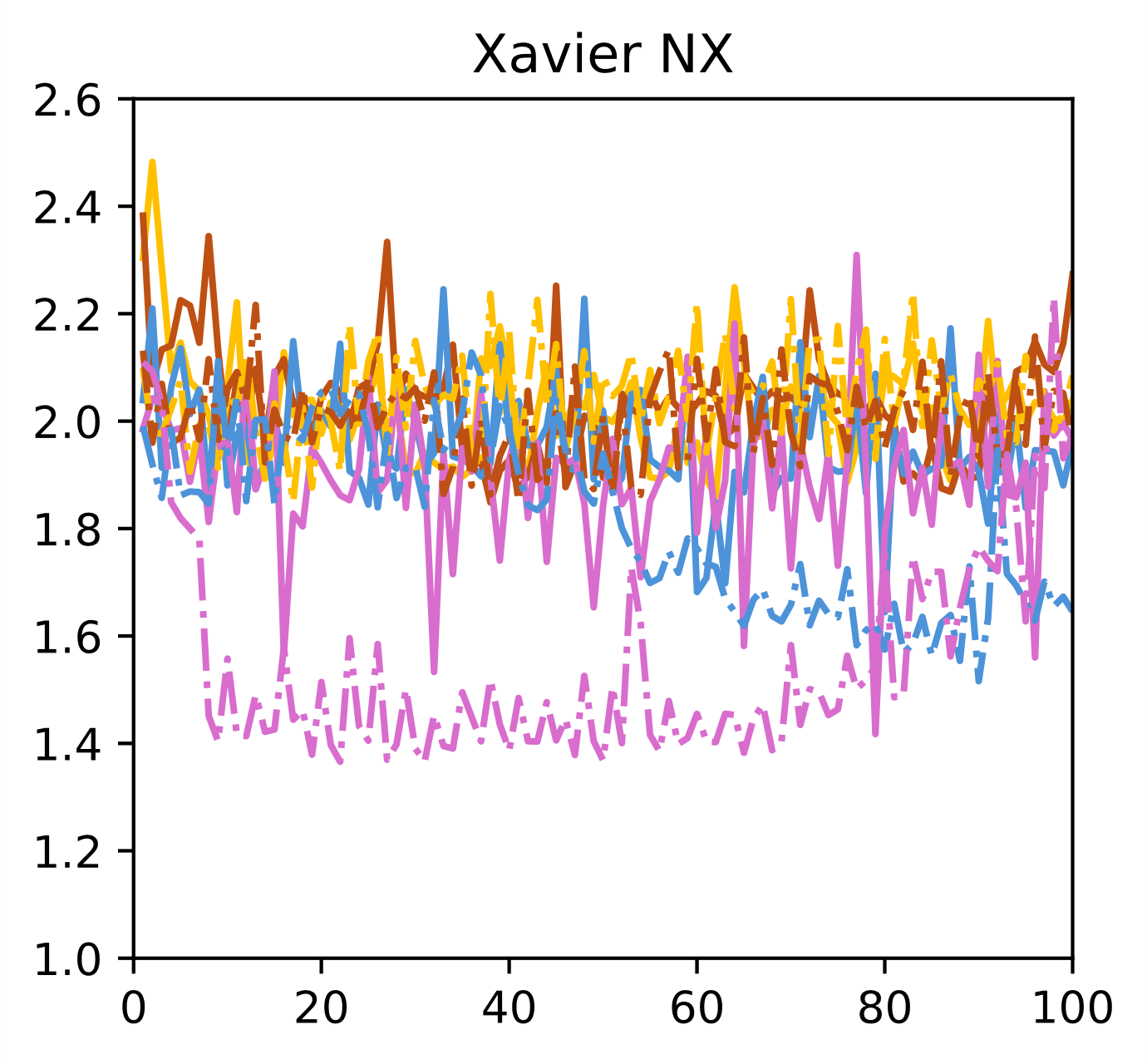


Abbildung 2. Zeiten für den Transfer der Eingabedaten aus der Beispieldatei vom Host zum Device. Die horizontale Achse zeigt den zeitlichen Verlauf über die jeweils 100 Wiederholungen. Auf der vertikalen Achse ist jeweils die gemessene Transferzeit in Millisekunden (ms) abgetragen. Man beachte die unterschiedliche Skalierung der vertikalen Achsen in den drei Diagrammen.

Während die Transferzeiten auf die Tesla-GPU sehr konstant sind, erkennt man bei den anderen beiden Geräten erhebliche Fluktuationen. Bei der Radeon-GPU fällt auf, dass der jeweils erste Transfer weitaus länger dauert als die folgenden. Die Transfers dauern generell bei den Varianten mit 27 Bins (oberes Liniencluster im Diagramm) länger als bei den Varianten mit 128 Bins (unteres Liniencluster im Diagramm), obwohl die gleichen Daten übertragen werden und die Bins an dieser Stelle des Programms noch keine Rolle spielen sollten. Generell scheint sowohl bei der Xavier-GPU als auch bei der Radeon-GPU der Transfer für einige Kernels länger zu dauern als für andere, wenngleich dies bei beiden GPUs nicht für die gleichen Kernels der Fall ist. Die Fluktuationen scheinen aber nicht rein zufällig zu sein. Wir haben für dieses Verhalten keine plausible Erklärung. Möglicherweise hängt es mit Compileroptimierungen oder Cache-Effekten zusammen. Es ist jedenfalls compiler- oder hardwareabhängig. Es wurde *nicht* geprüft, ob sich das Verhalten mit Änderungen am Programmcode, insbesondere einer anderen Ausführungsreihenfolge der Kernels, ändert.

In Abbildung 3 sind der Vollständigkeit halber die Zeiten für den Transfer vom Device zum Host dargestellt. Aufgrund der geringen Datenmenge (128 bzw. 27 Werte vom Typ unsigned int) sind die Messwerte wohl wenig aussagekräftig und die Schwankungen als «Noise» zu betrachten.

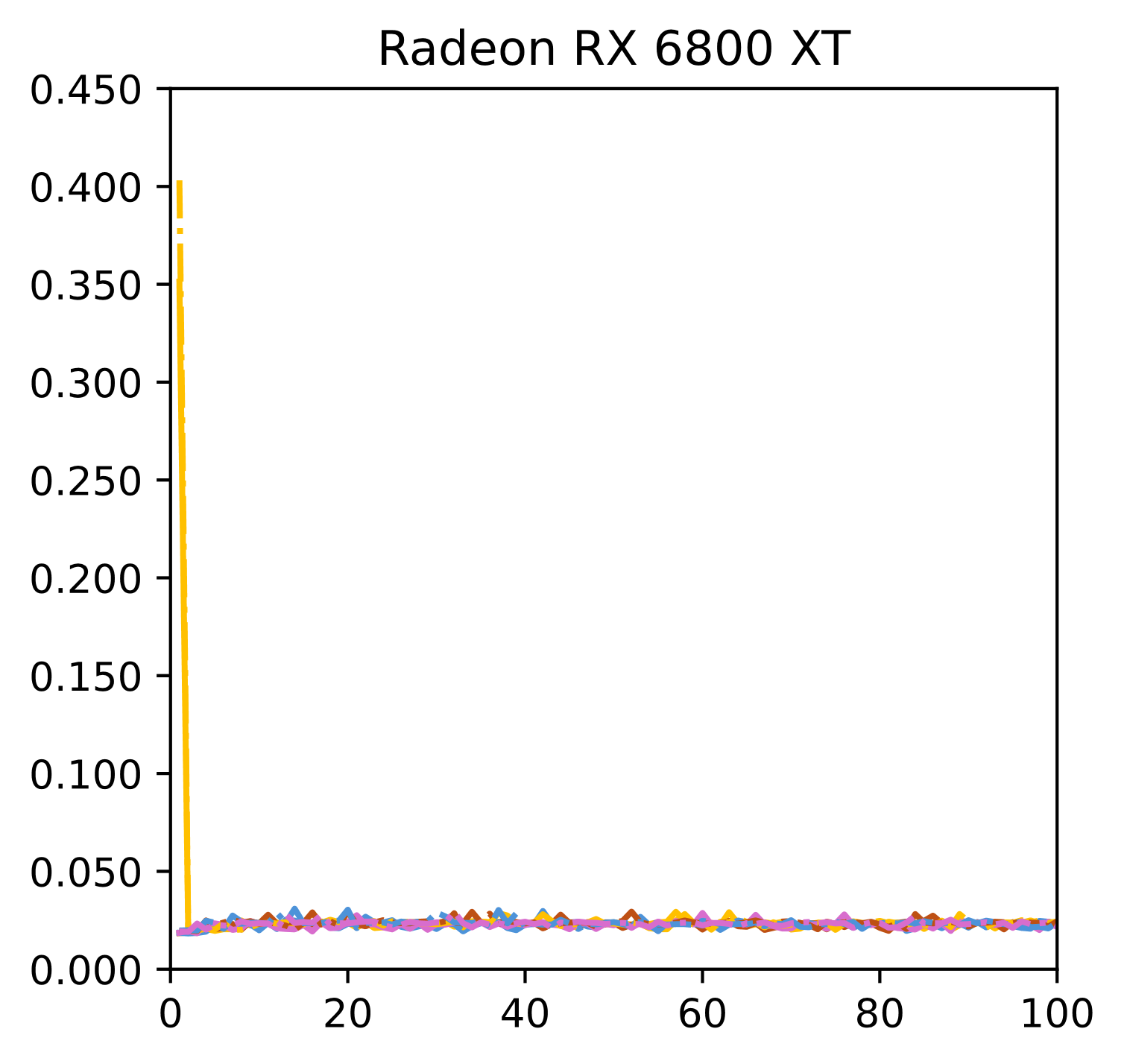
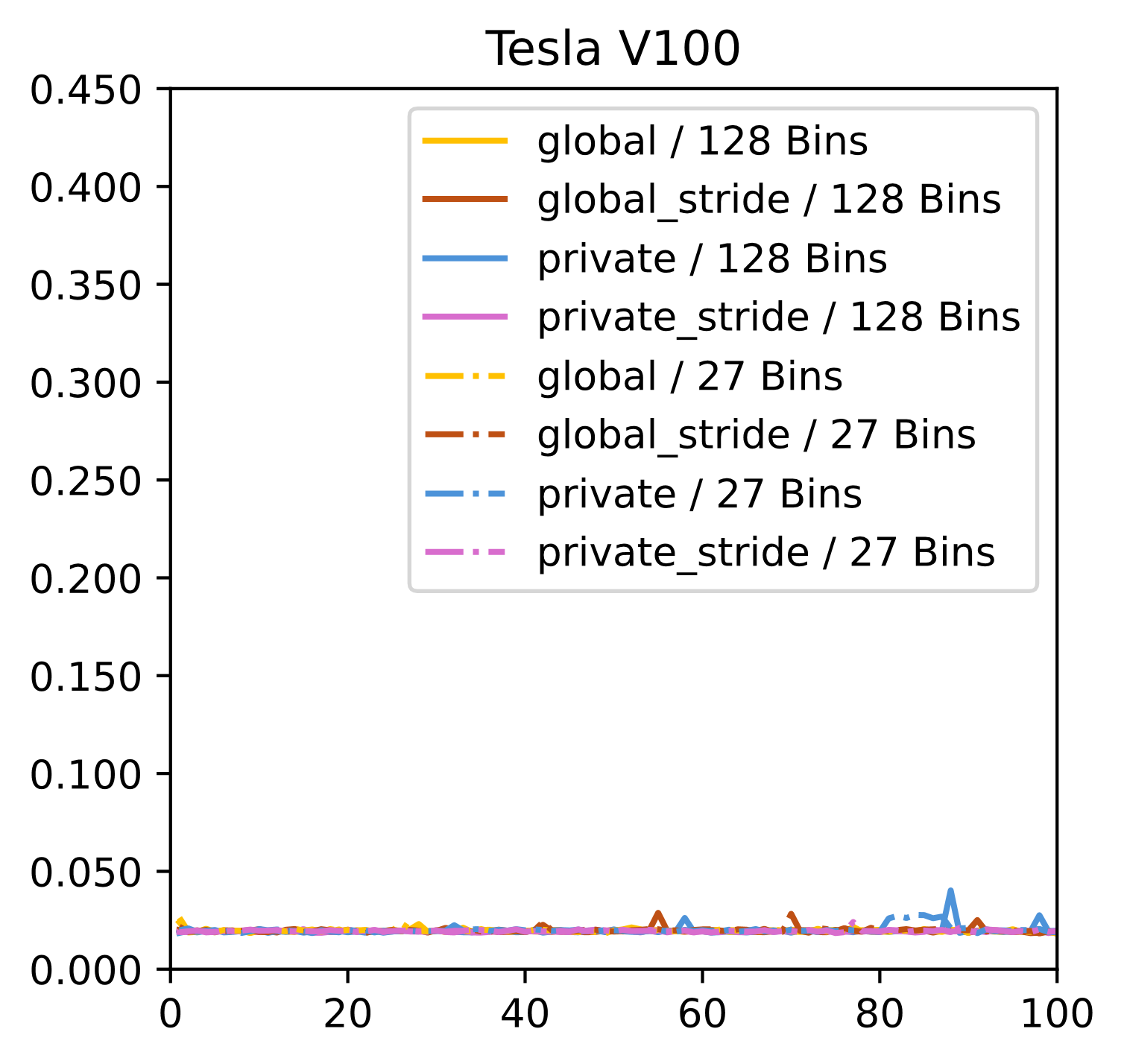
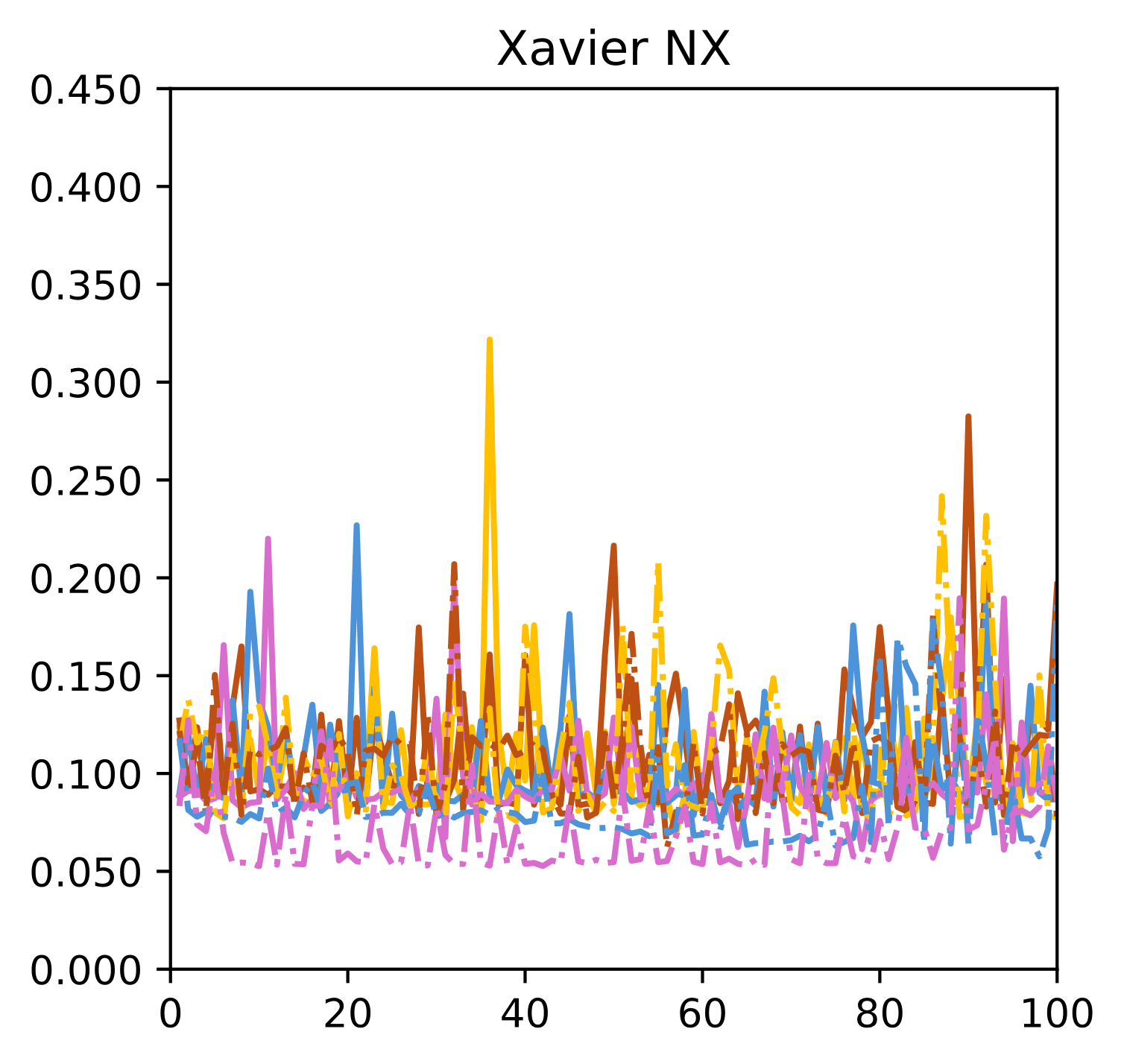


Abbildung 3. Zeiten für den Transfer der Resultate vom Device zum Host. Die horizontale Achse zeigt den zeitlichen Verlauf über die jeweils 100 Wiederholungen. Auf der vertikalen Achse ist jeweils die gemessene Transferzeit in Millisekunden (ms) abgetragen.

Wir kommen nun zur Analyse der Ausführungszeiten. Abbildung 4 zeigt, dass die Ausführungszeiten im Laufe der 100 Wiederholungen jeweils sehr konstant sind; wir können also beim Vergleich der Kernels und Mappings mit den Medianwerten arbeiten.

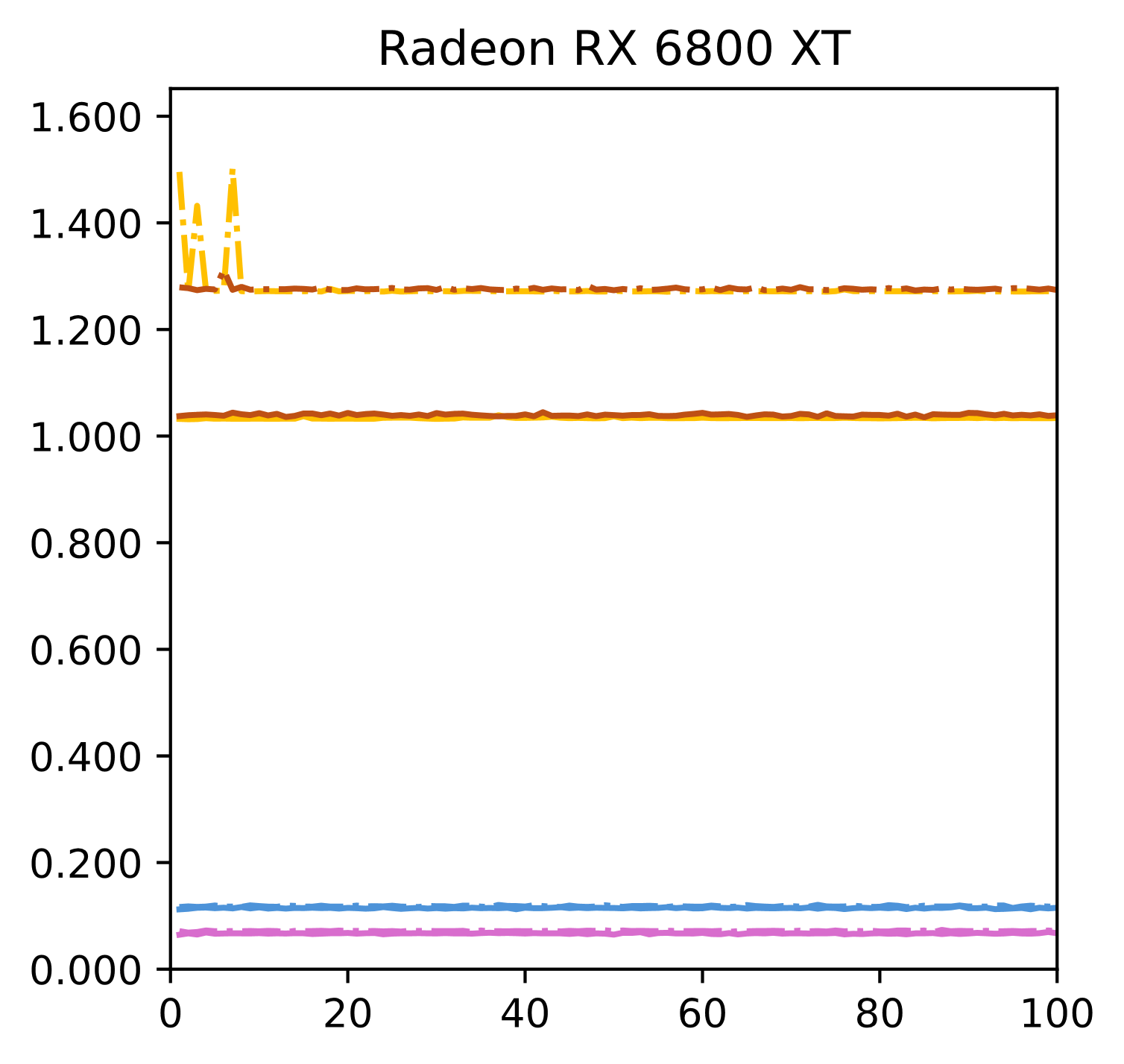
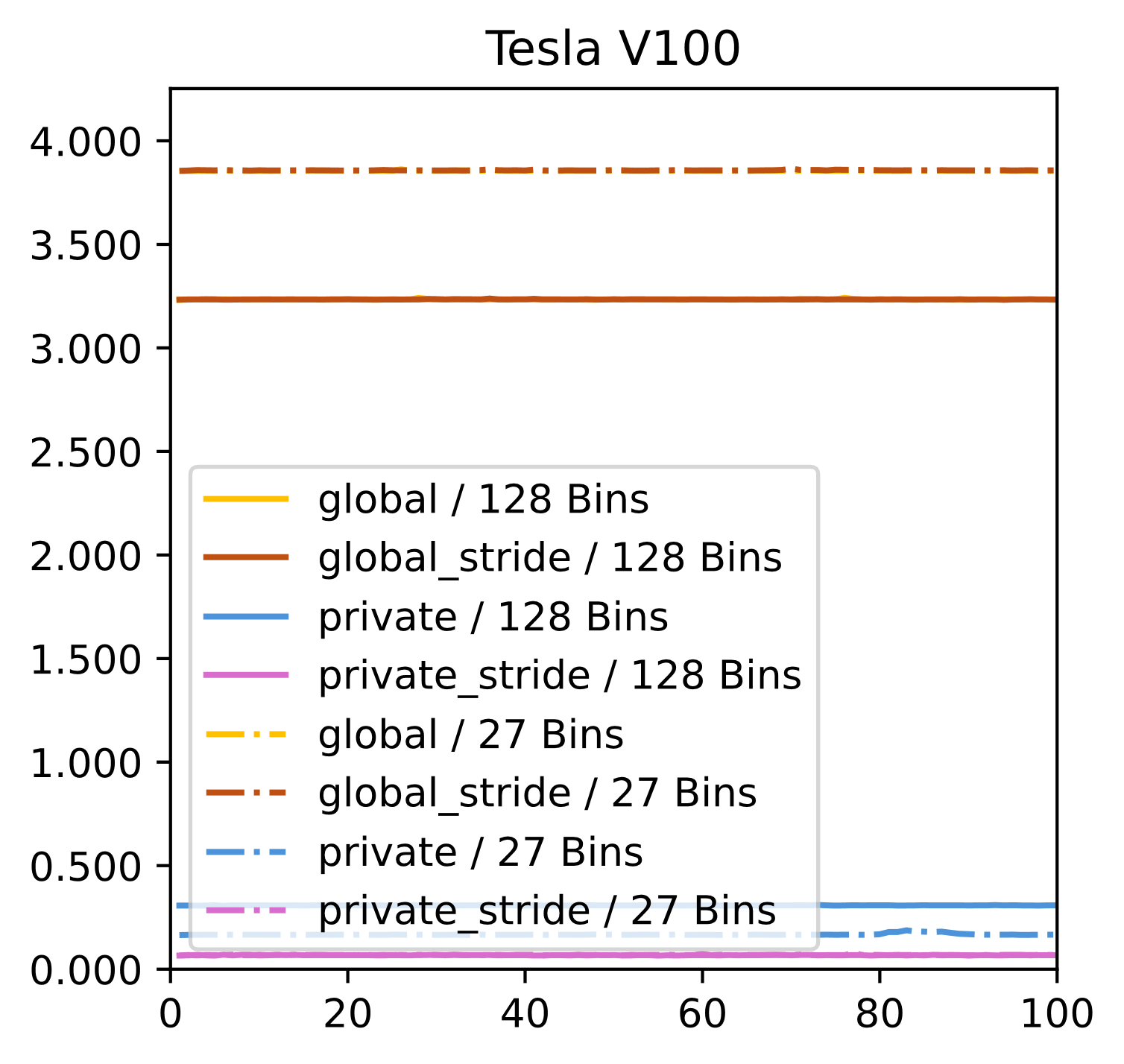
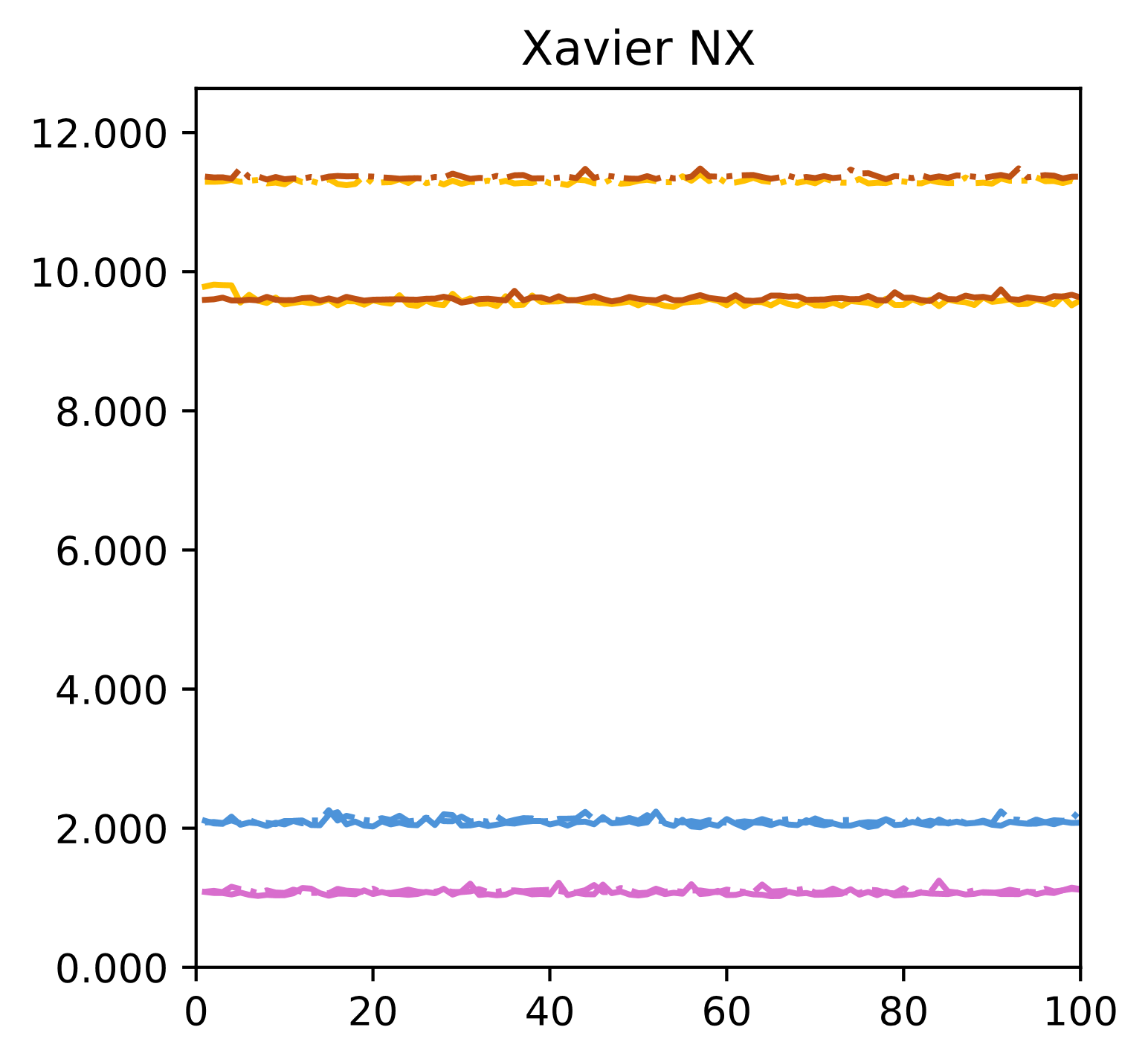


Abbildung 4. Ausführungszeiten auf dem Device für die Eingabedaten aus der Beispieldatei. Die horizontale Achse zeigt den zeitlichen Verlauf über die jeweils 100 Wiederholungen. Auf der vertikalen Achse ist jeweils die gemessene Transferzeit in Millisekunden (ms) abgetragen. Man beachte die unterschiedliche Skalierung der vertikalen Achsen.

Abbildung 5 zeigt die Mediane der Ausführungszeiten. Es fällt zweierlei auf.

Zum einen: Während bei ausschliesslicher Verwendung des globalen Speichers (Kernels global und global\_stride) das Stride keinen nennenswerten Unterschied macht, führt bei Verwendung von Shared Memory (Kernels private und private\_stride) der Stride zu einem deutlichen Zeitvorteil. Die uns am wahrscheinlichsten scheinende Erklärung für dieses Verhalten ist der verringerte Zugriff auf den globalen Speicher bei der abschliessenden Aggregation durch die niedrigere Anzahl an Blöcken ( Blöcke ohne Stride vs. 1024 Blöcke mit Stride).

Zum anderen: Bei ausschliesslicher Verwendung globalen Speichers führt eine höhere Anzahl von Bins zu einem Zeitvorteil. Dieser Effekt ist bei Verwendung von Shared Memory nicht zu sehen. Eine mögliche Erklärung wäre, dass bei einer höheren Anzahl von Bins seltener Kollisionen auftreten, d.h. verschiedene Threads gleichzeitig denselben Bin hochzählen möchten. Bei Verwendung von Shared Memory treten solche Kollisionen prinzipbedingt seltener auf.

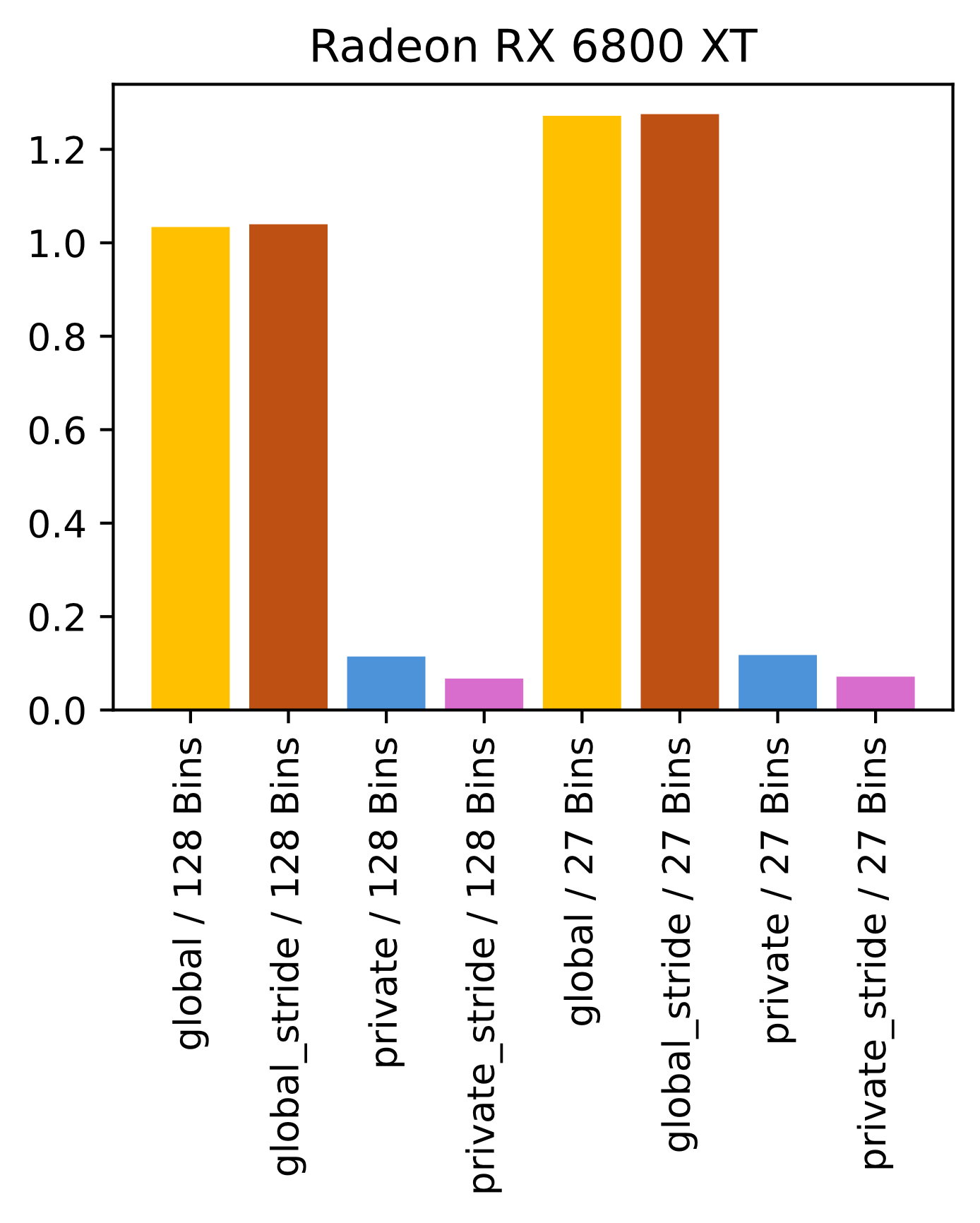
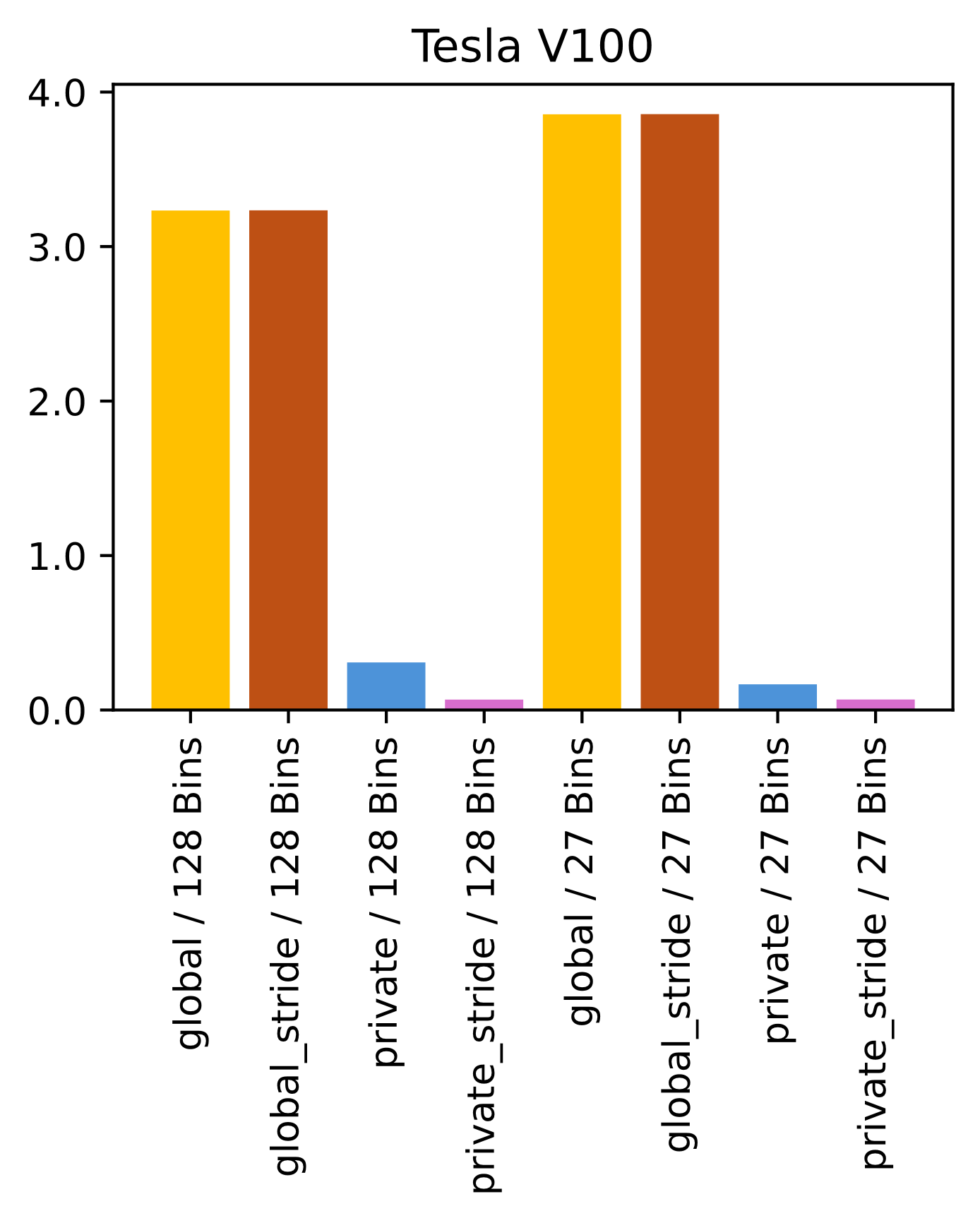
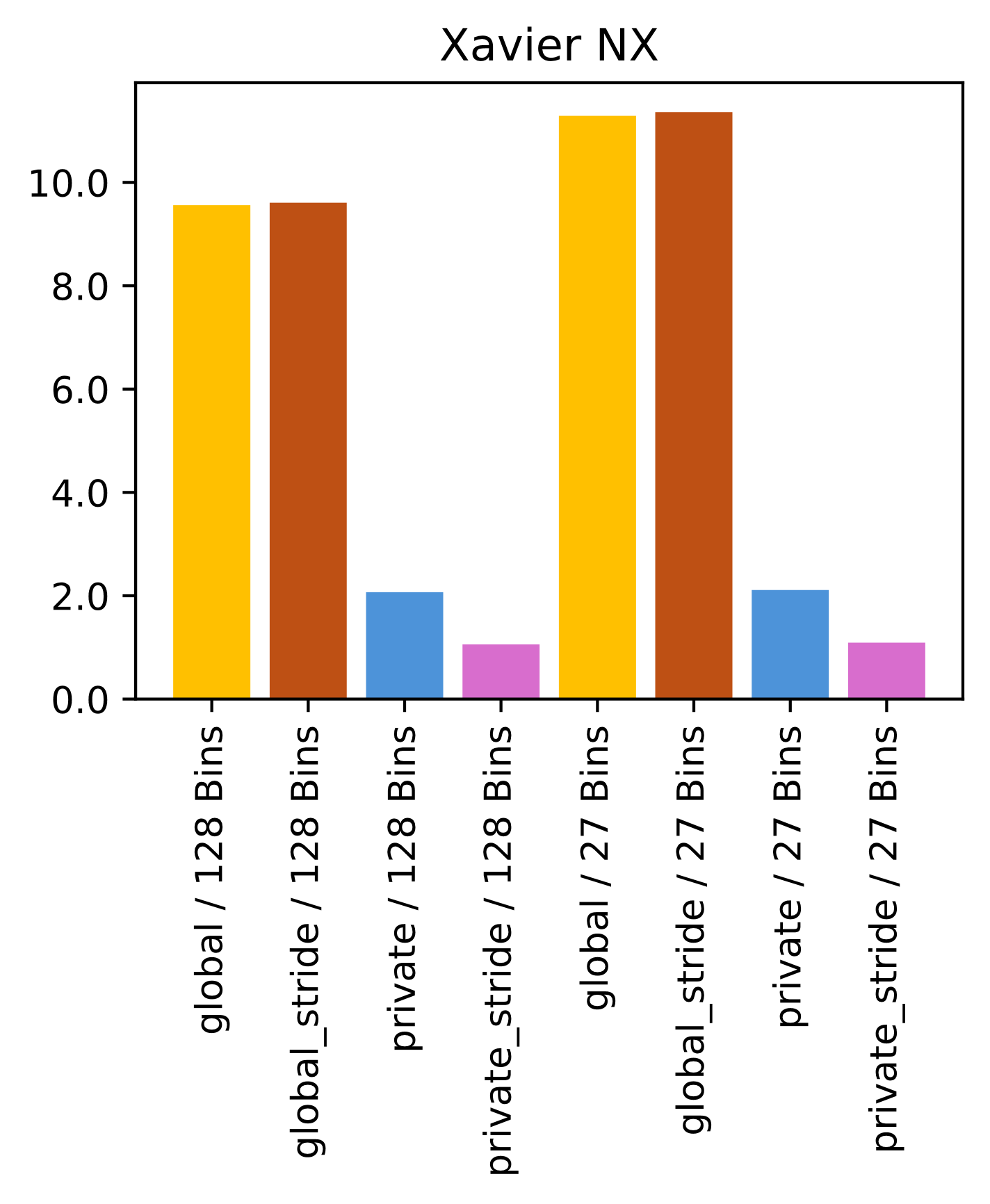


Abbildung 5. Median der Ausführungszeit (in ms). Man beachte die unterschiedliche Skalierung der vertikalen Achsen.

Die Tabellen Tabelle 1 und Tabelle 2 zeigen den Speedup im Vergleich zum Kernel global für 128 bzw. 27 Bins. Tabelle 3zeigt den Faktor, um den die Ausführungszeit bei 27 Bins gegenüber 128 Bins höher ist.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Speedup gegenüber global, 128 Bins** | | | |
| **Device** | **global** | **global\_stride** | **private** | **private\_stride** |
| Xavier NX | 1.00 | 1.00 | 4.62 | 9.04 |
| Tesla V100 | 1.00 | 1.00 | 10.50 | 47.81 |
| Radeon RX 6800 XT | 1.00 | 0.99 | 9.02 | 15.38 |

Tabelle 1. Speedup gegenüber der untersuchten Kernels gegenüber dem Kernel „global“ für die Eingabedaten aus der Beispieldatei bei Verwendung von 128 Bins.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Speedup gegenüber global, 27 Bins** | | | |
| **Device** | **global** | **global\_stride** | **private** | **private\_stride** |
| Xavier NX | 1.00 | 0.99 | 5.35 | 10.35 |
| Tesla V100 | 1.00 | 1.00 | 23.19 | 56.35 |
| Radeon RX 6800 XT | 1.00 | 1.00 | 10.80 | 17.77 |

Tabelle 2. Speedup gegenüber der untersuchten Kernels gegenüber dem Kernel „global“ für die Eingabedaten aus der Beispieldatei bei Verwendung von 27 Bins.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Ausführungszeit bei 27 Bins im Verhältnis zur Ausführungszeit bei 128 Bins** | | | |
| **Device** | **global** | **global\_stride** | **private** | **private\_stride** |
| Xavier NX | 1.18 | 1.18 | 1.02 | 1.03 |
| Tesla V100 | 1.19 | 1.19 | 0.54 | 1.01 |
| Radeon RX 6800 XT | 1.23 | 1.23 | 1.03 | 1.06 |

Tabelle 3. Ausführungszeit bei 27 Bins im Verhältnis zur Ausführungszeit bei 128 Bins für die verschiedenen Kernels und GPUs.

# Messergebnisse für unterschiedliche Eingabegrössen

Um zu untersuchen, wie die Ausführungszeiten von der Grösse der Eingabe abhängen, haben wir, wie oben beschrieben, generierte Daten verwendet. Da zu vermuten ist, dass der Umfang der Kollisionen (d.h. wie viele Threads auf den gleichen Bin zu schreiben versuchen) einen Einfluss auf die Ausführungszeit hat, haben wir, wie oben beschrieben, zum einen pseudozufällige Daten verwendet, bei denen jeder Zeichencode zwischen 1 und 128 annähernd gleich häufig vorkommt, und zum anderen konstante Daten, welche nur aus dem Buchstaben ‚a‘ bestehen.

Es wurden als Eingabegrössen alle Zweierpotenzen zwischen Bytes und GiB untersucht. Bei Bytes überschreitet bei den Kernels ohne Stride die Anzahl der nötigen Threads die Anzahl möglicher Threads, so dass für diese beiden Kernels und diese Eingabegrösse kein Messwert existiert. Anhang D enthält eine Auswahl an Zahlenwerten. Wir beschränken uns hier auf die graphische Darstellung.

Abbildung 6 zeigt die Ausführungszeiten in Abhängigkeit von der Eingabegrösse. Die durchgezogenen Kurven repräsentieren jeweils den Median; die (aufgrund der meist geringen Schwankungen kaum sichtbaren) gestrichelten Kurven das erste und letzte Dezil. Abbildung 7 zeigt den Speedup der einzelnen Kernels im Vergleich zum Kernel «global», wieder in Abhängigkeit von der Eingabegrösse.

Es fällt auf:

* Für pseudozufällige Eingabedaten ergeben sich meist geringere Laufzeiten als für konstante Eingabedaten. Dieser Effekt ist bei 128 Bins stärker ausgeprägt als bei 27 Bins. Eine plausible Erklärung ist wieder, dass Kollisionen einen negativen Effekt auf die Laufzeit haben.
* Für grosse Eingaben ist der Kernel private\_stride den anderen Kerneln überlegen. Als Grund vermuten wir, dass dieser Kernel aufgrund der konstanten Anzahl an Blöcken die Zugriffe auf den globalen Speicher minimiert.
* Für kleine Eingaben ist die Laufzeit bei allen Kerneln zunächst über einen gewissen Bereich nahezu konstant. Dies ist wohl so lange der Fall, solange die Hardware die Ausführung noch perfekt parallelisieren kann.
* Der Kernel atomic\_private\_stride ist für kleine Eingaben den übrigen Kerneln unterlegen, wobei der Effekt auf unterschiedlichen GPUs unterschiedlich ausgeprägt ist. Verantwortlich für die Unterlegenheit ist wohl die konstante Anzahl an Blöcken, welche für kleine Eingaben grösser als optimal ist. Dieser Effekt tritt jedoch beim Kernel global\_stride nicht auf. Betrachtet man den Programmcode, fällt auf, dass beim Kernel private\_stride auch die «überflüssigen» Blöcke an der abschliessenden Aggregation teilnehmen. Es wäre zu prüfen, ob sich eine Laufzeitverbesserung ergäbe, wenn man explizit prüft, ob Wert in der privaten Instanz im Shared Memory nicht 0 ist, bevor man die atomare Addition in den globalen Speicher ausführt. Andererseits erscheint es merkwürdig, dass der Compiler eine solche Optimierung nicht selbst vornimmt, wenn sie sinnvoll wäre.

Abbildung 8 zeigt das Verhältnis der Ausführungsdauer für 27 Bins zur Ausführungsdauer bei 128 Bins.

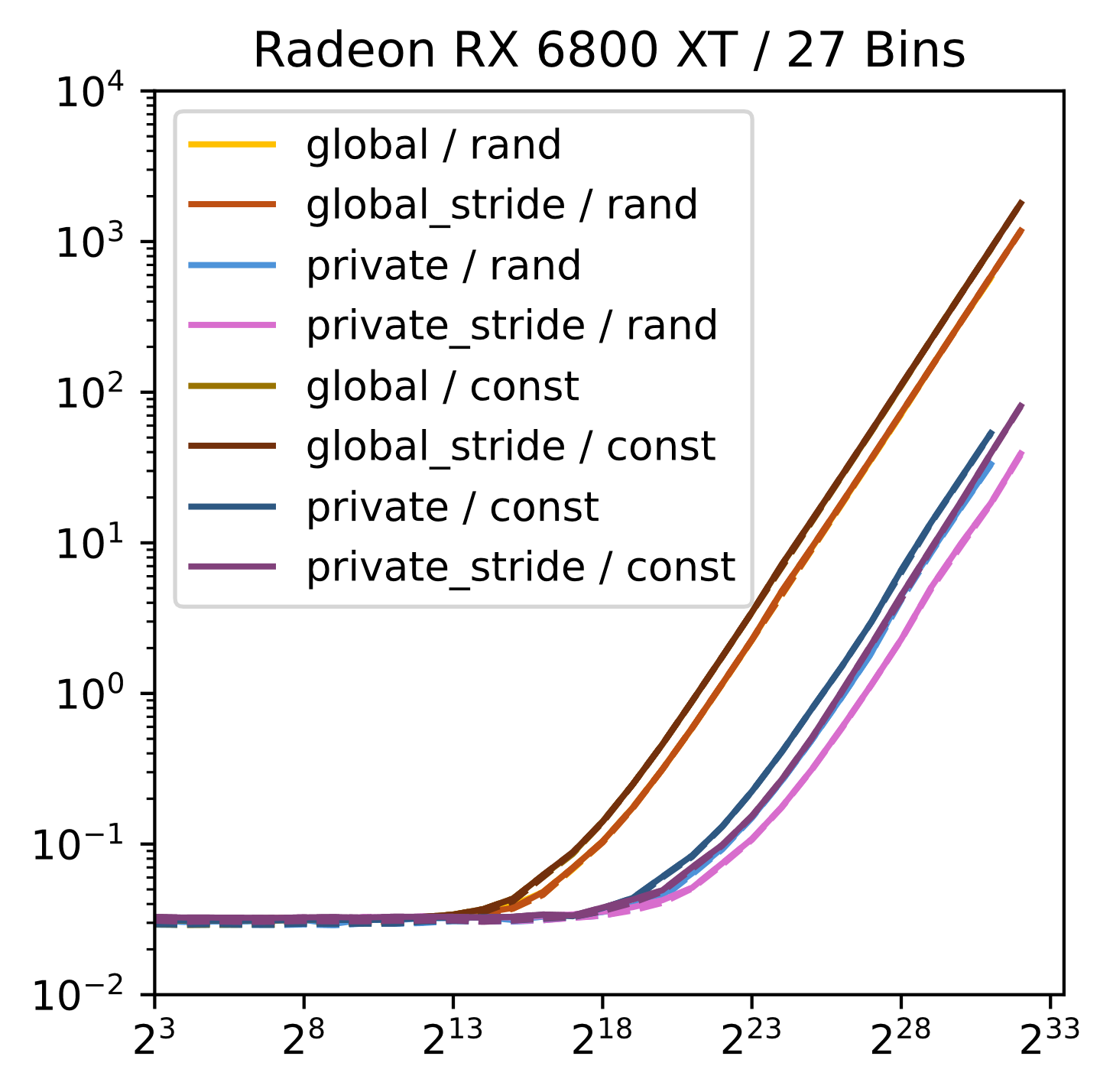
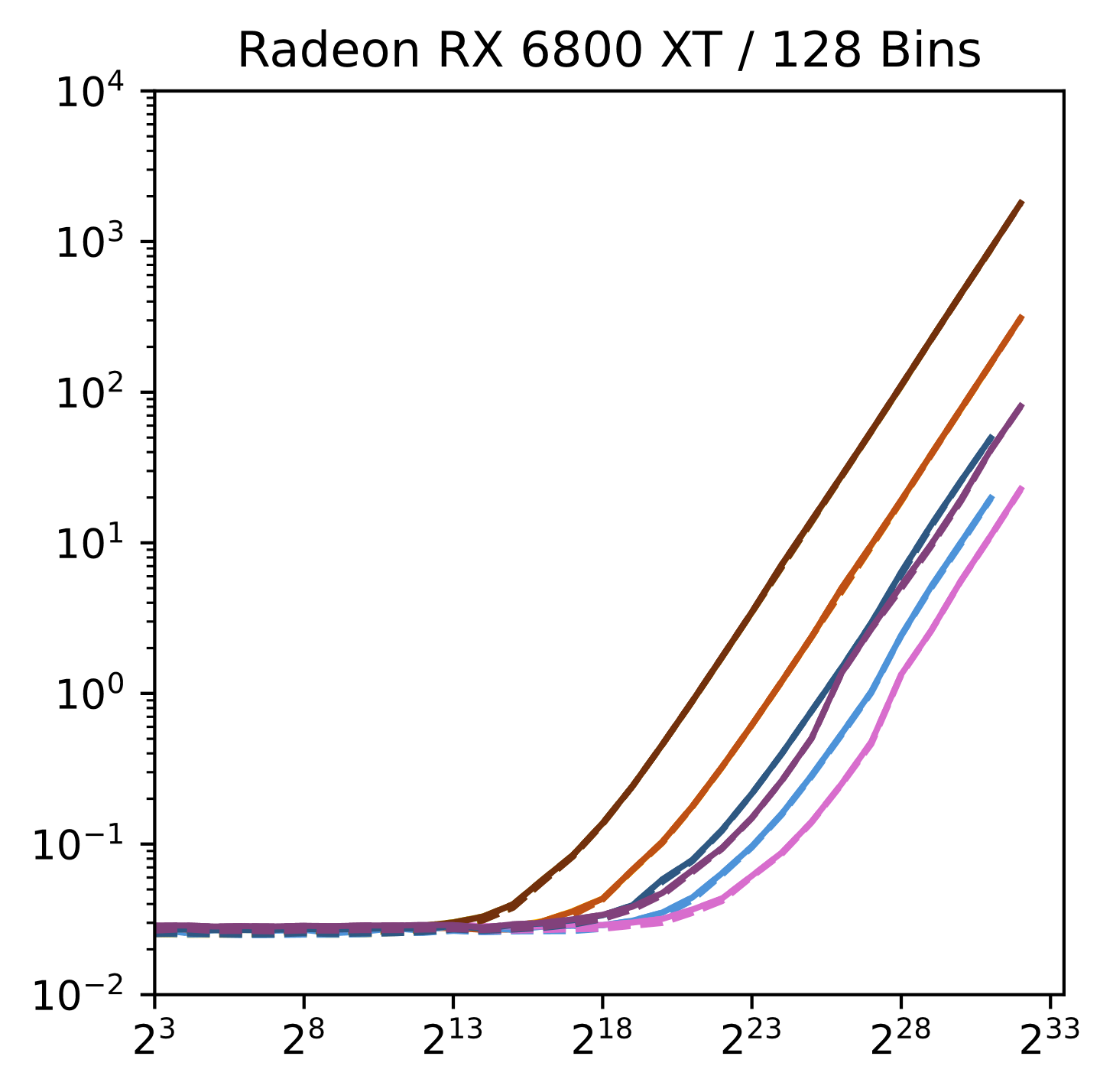
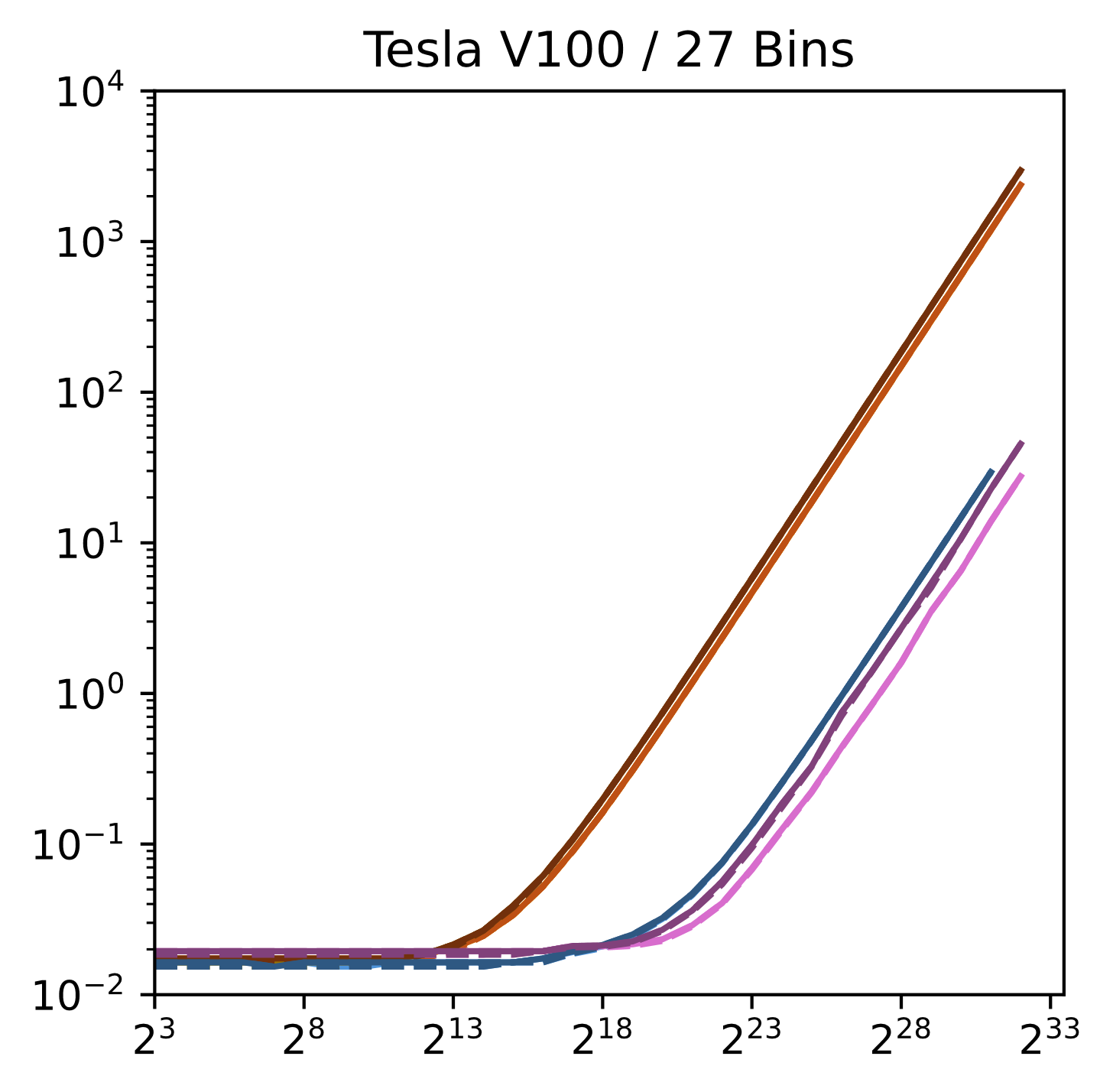
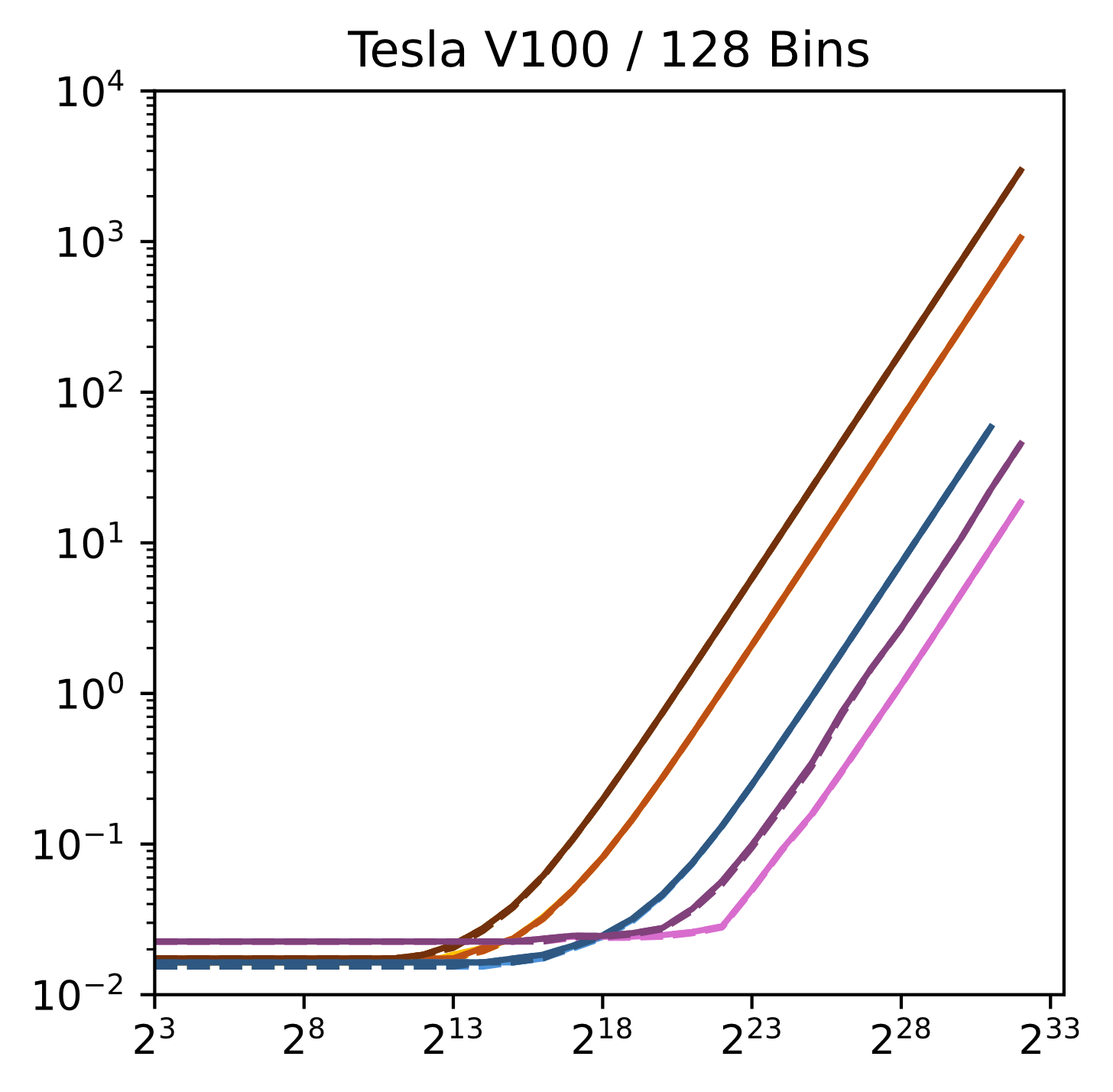
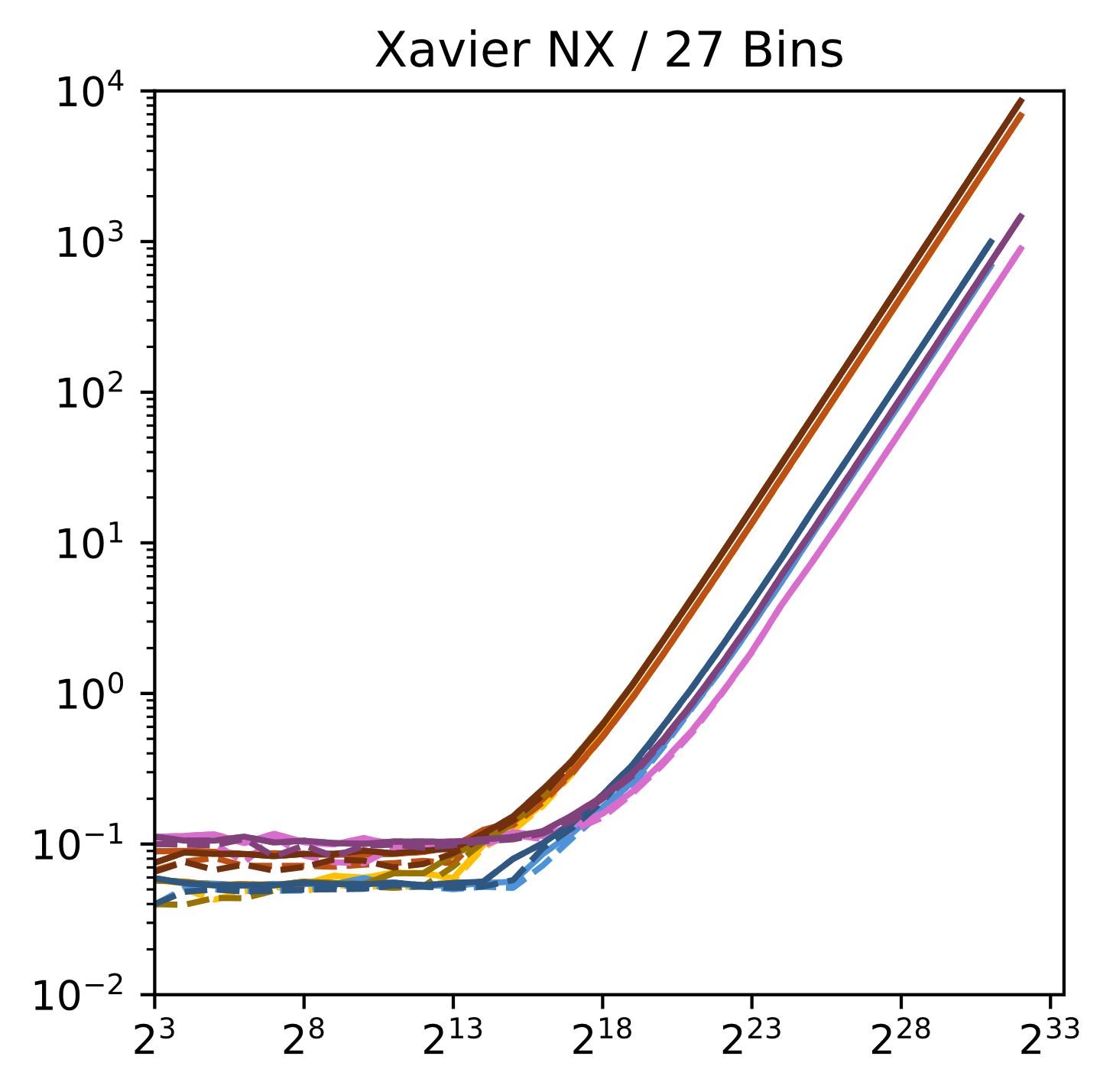
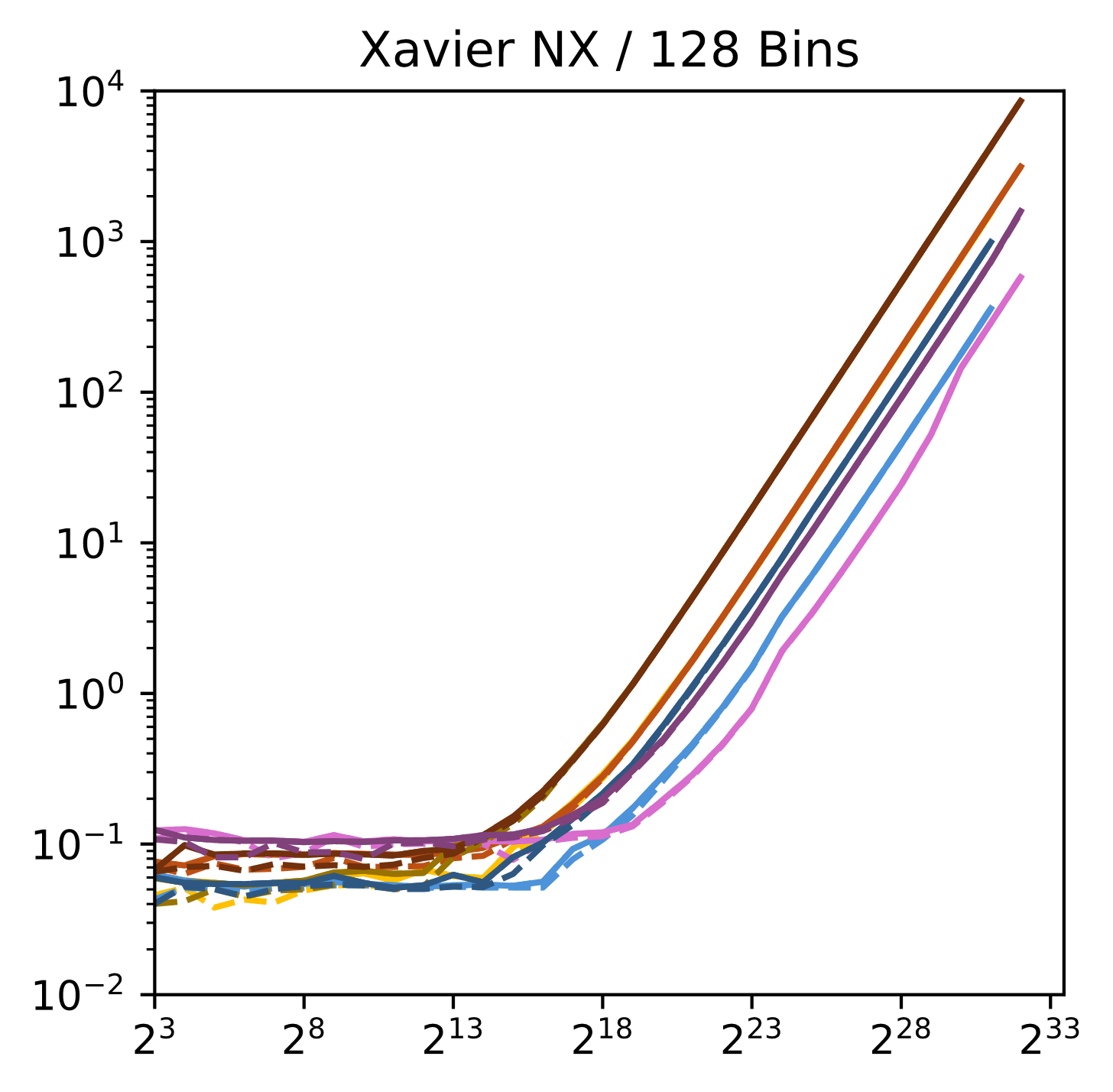


Abbildung 6. Ausführungszeiten (in ms, vertikale Achse) in Abhängigkeit von der Eingabegröße (horizontale Achse). Die helleren Kurven („rand“) repräsentieren die Zeiten bei pseudo-zufälligen Eingabedaten, die dunkleren Kurven („const“) bei konstanten Eingabedaten. Die Kurven für den Kernel „global“ werden weitestgehend durch die Kurven für „global\_stride“ verdeckt. Die durchgezogenen Kurven repräsentieren jeweils den Median. Das erste und letzte Dezil sind jeweils gestrichelt eingezeichnet, aber auch diese Kurven werden weitestgehend verdeckt.

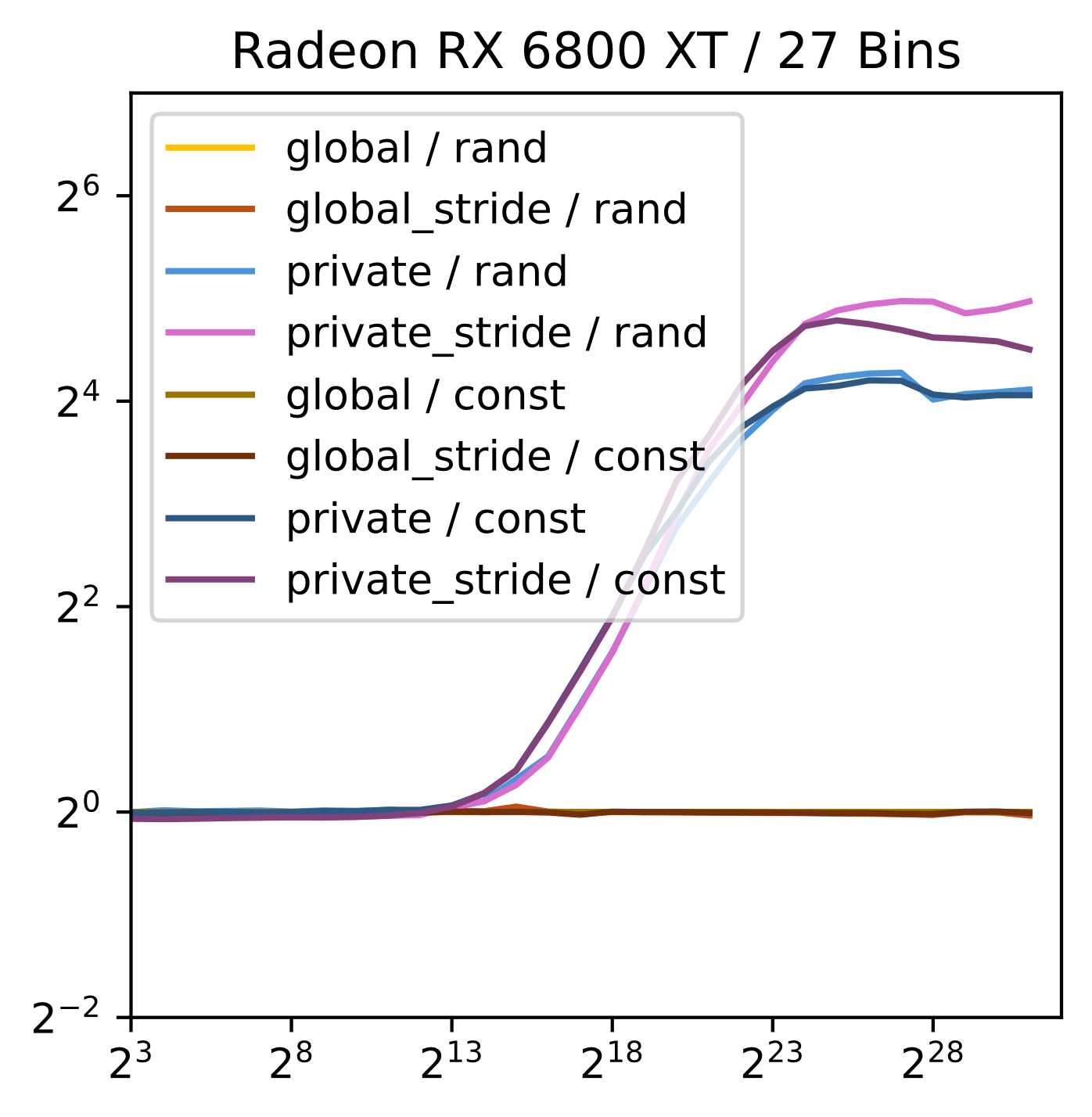
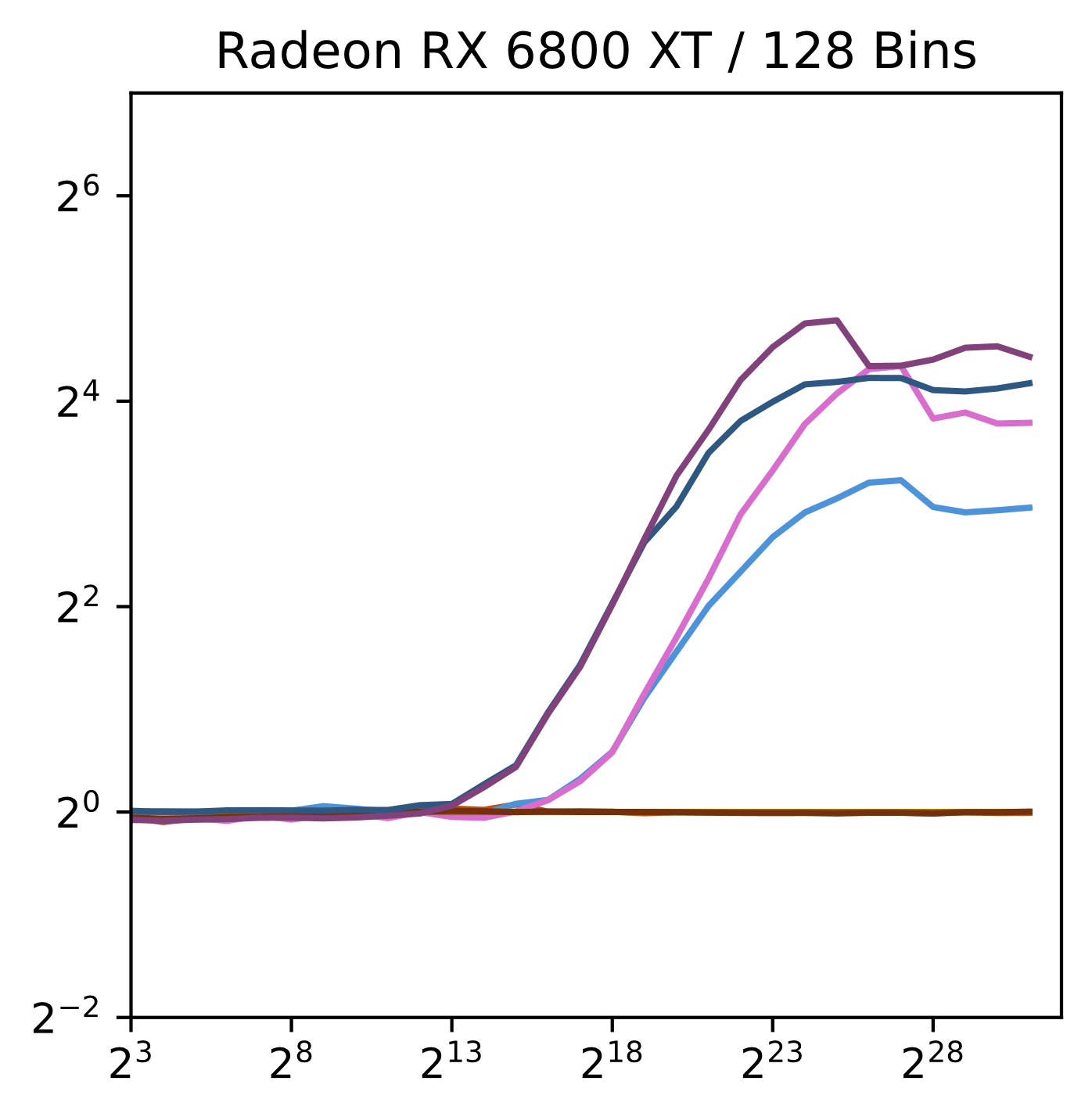
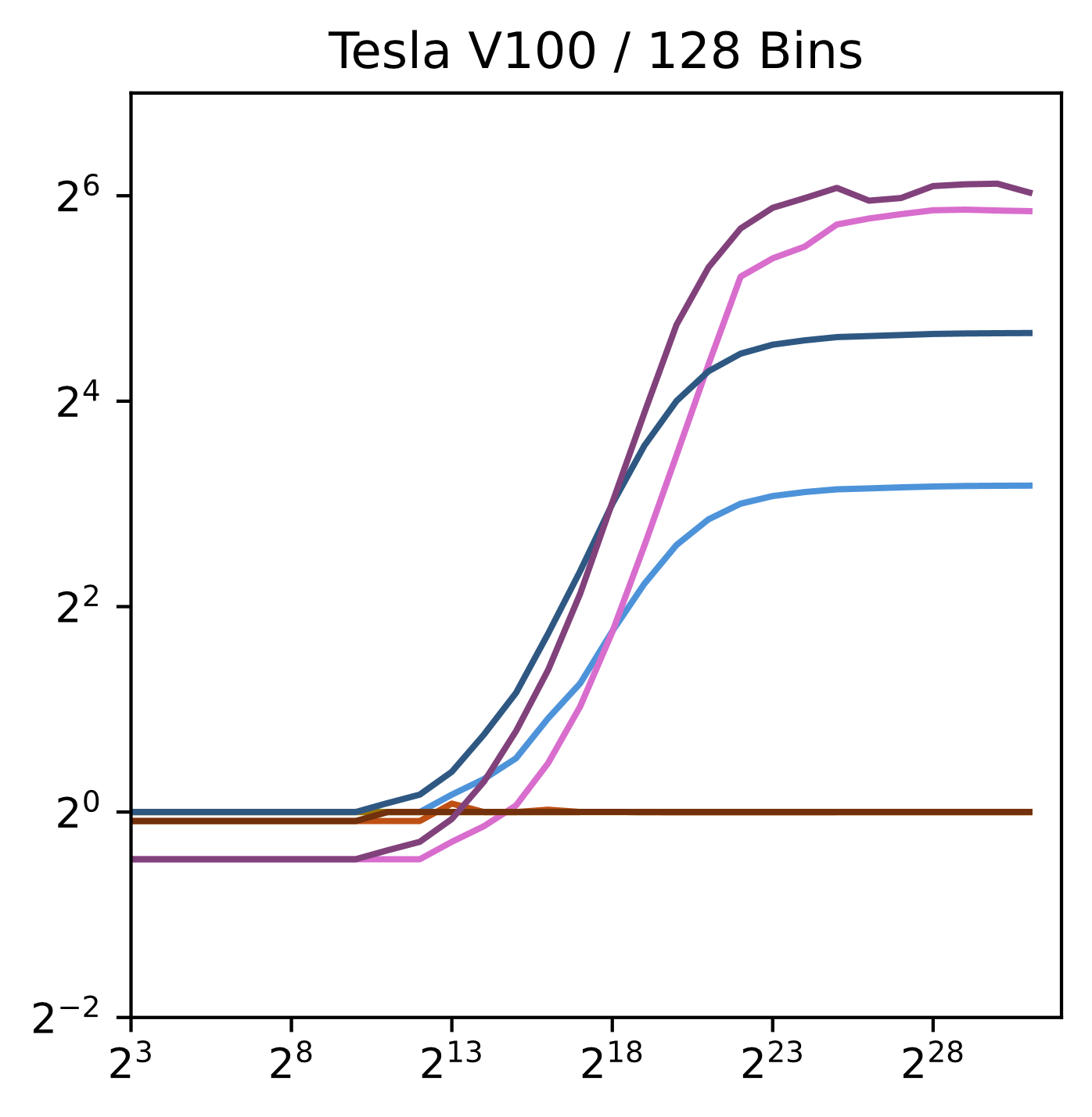
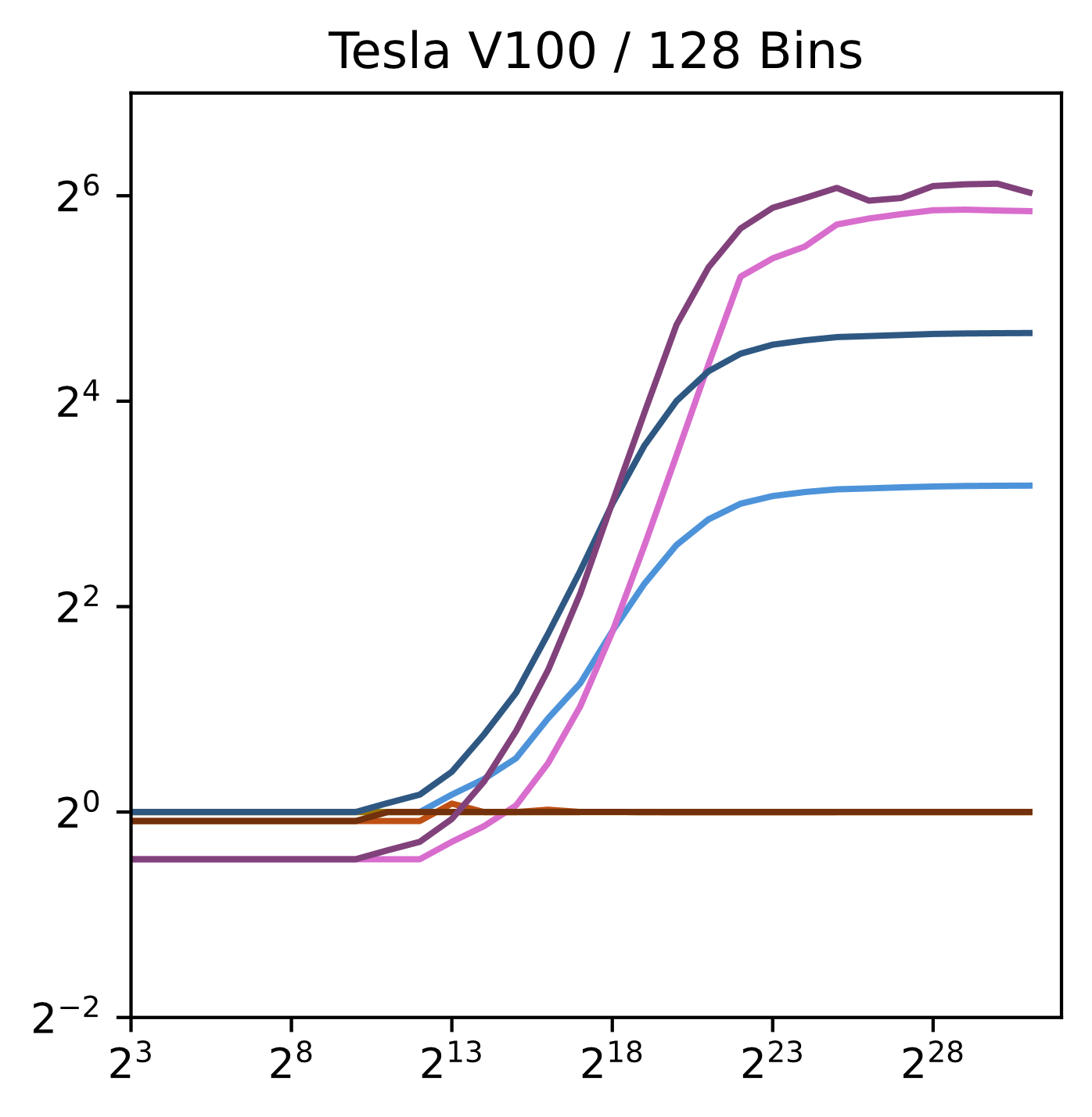
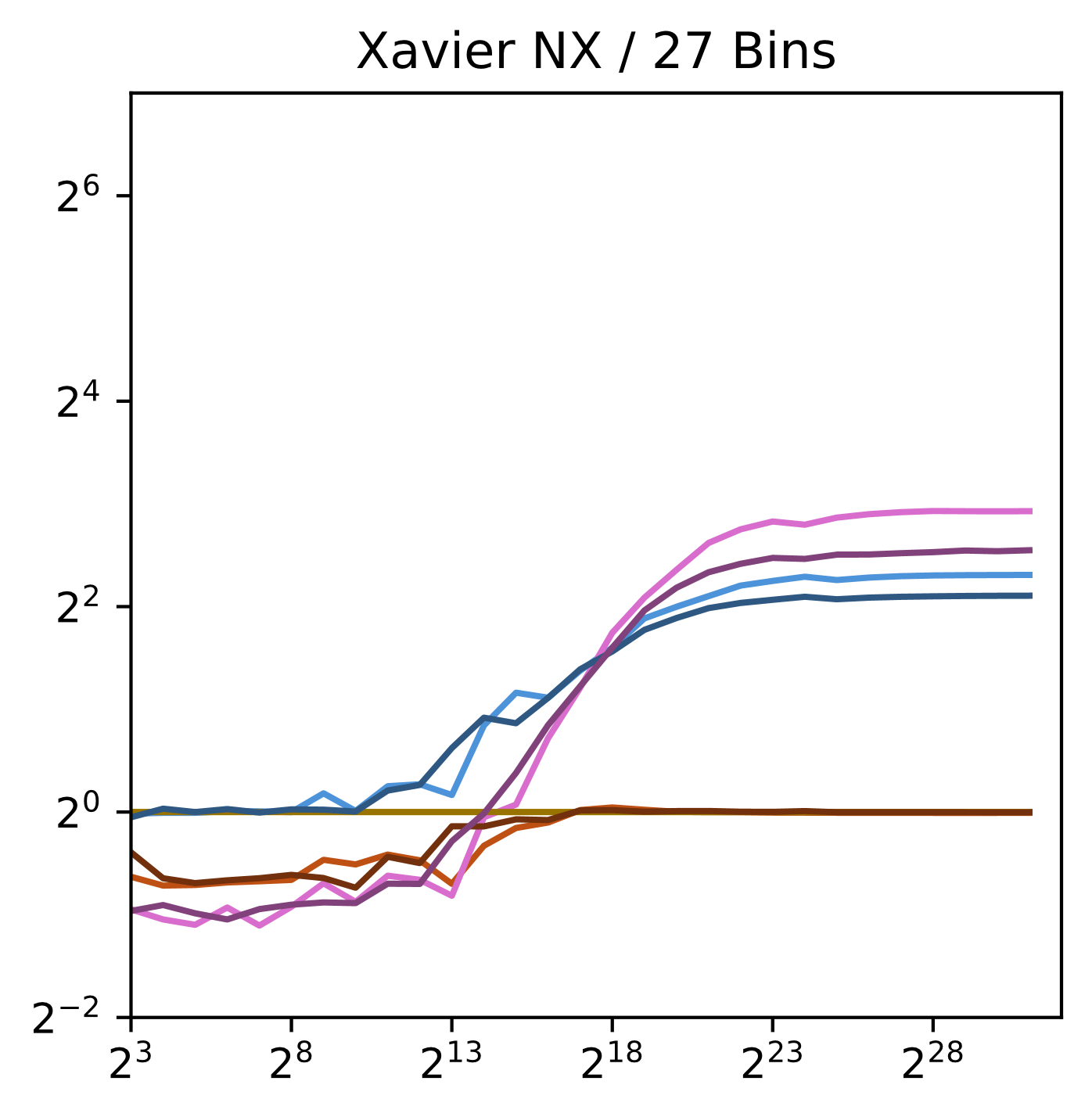
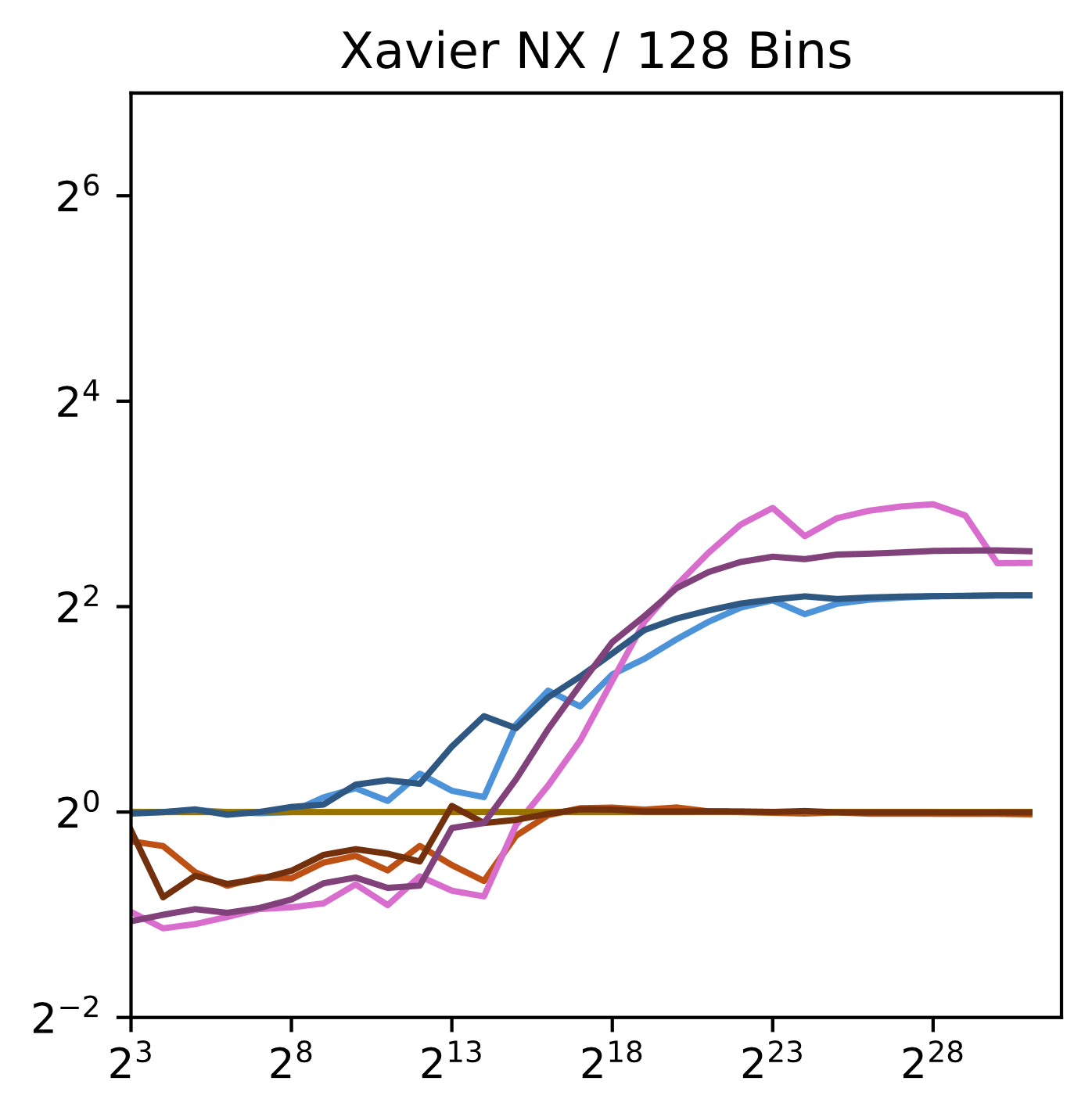


Abbildung 7. Speedup (vertikale Achse) gegenüber dem Kernel „global“ in Abhängigkeit von der Eingabegröße (horizontale Achse). Die helleren Kurven („rand“) repräsentieren die Ergebnisse bei pseudo-zufälligen Eingabedaten, die dunkleren Kurven („const“) bei konstanten Eingabedaten.

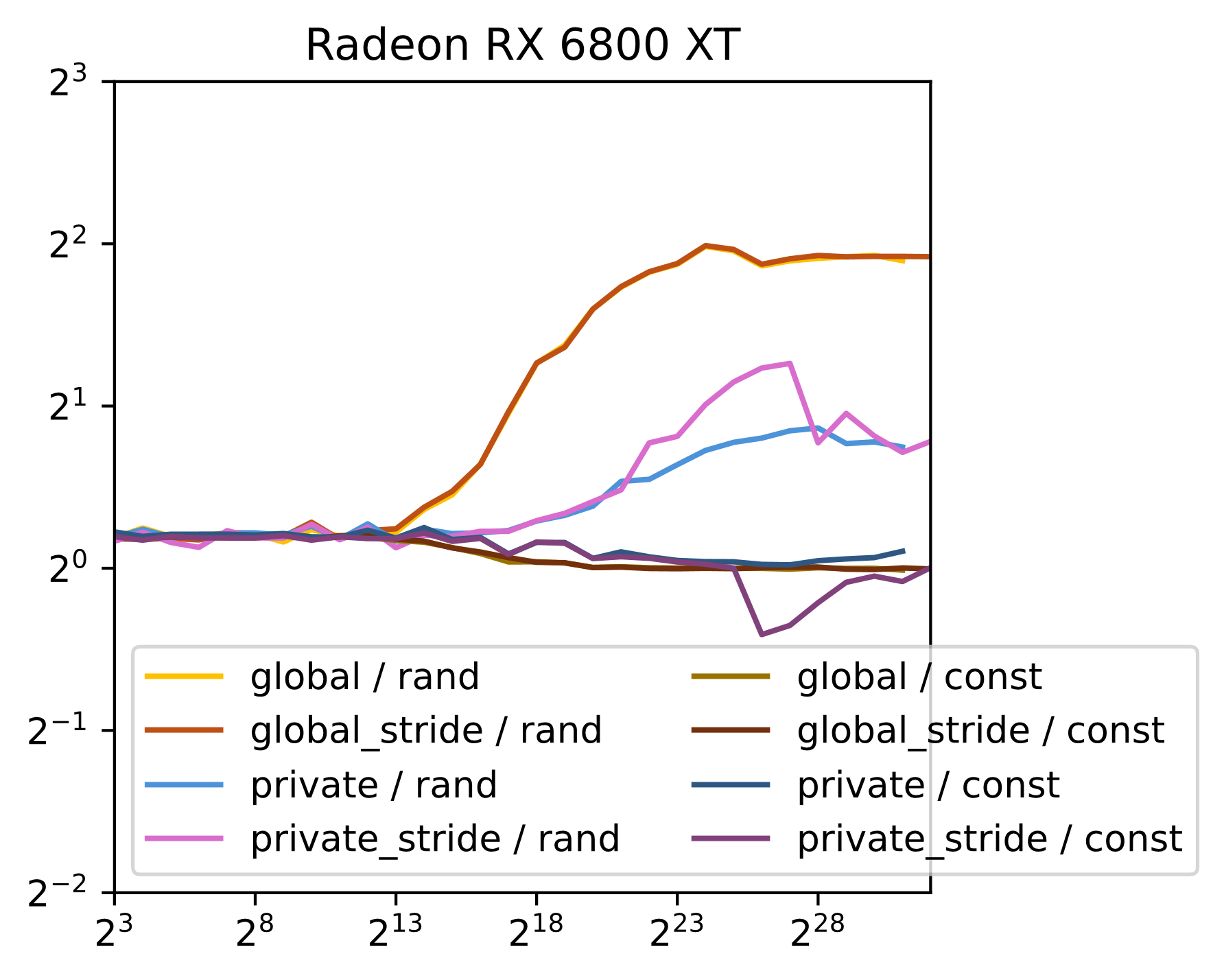
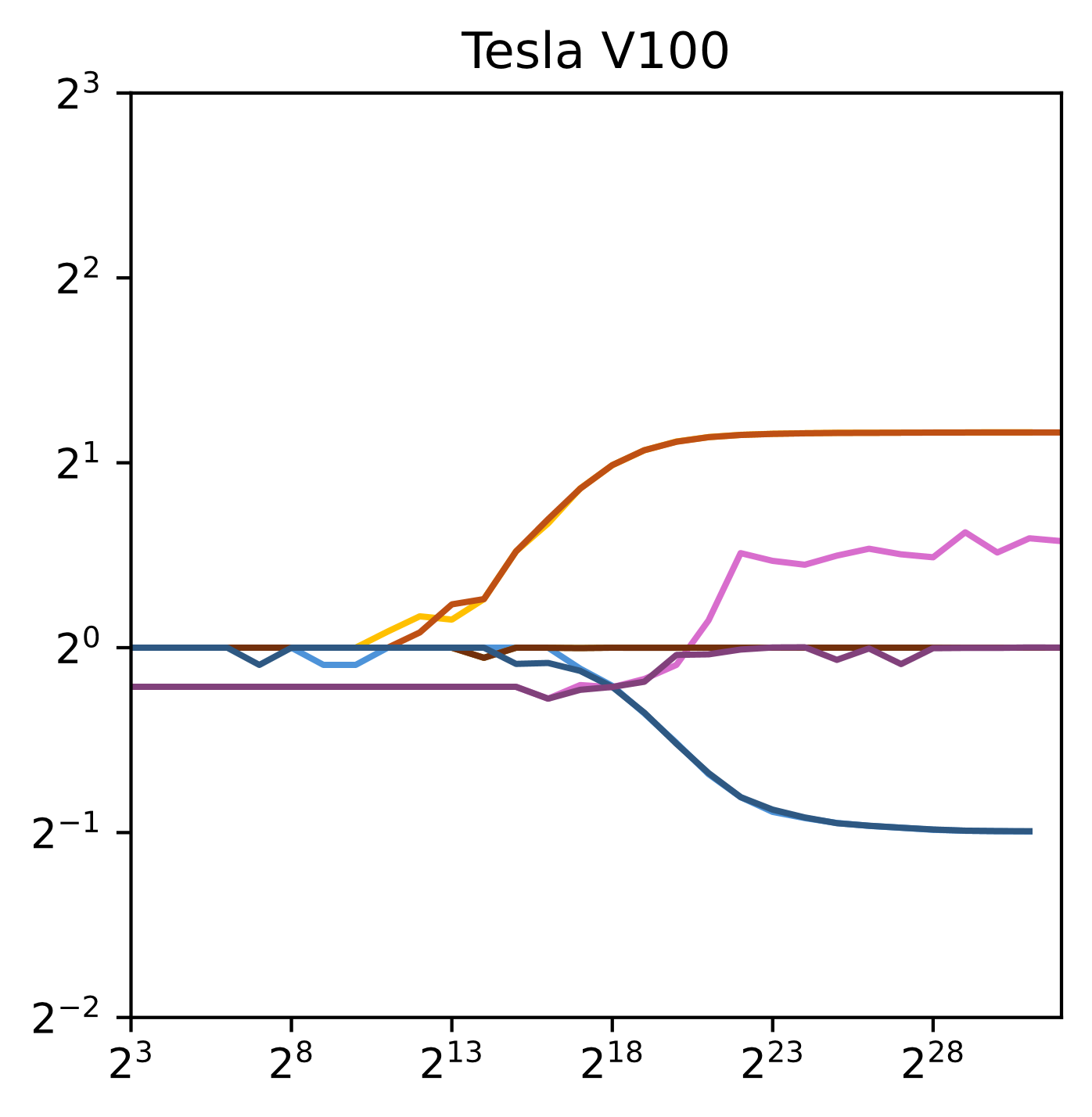
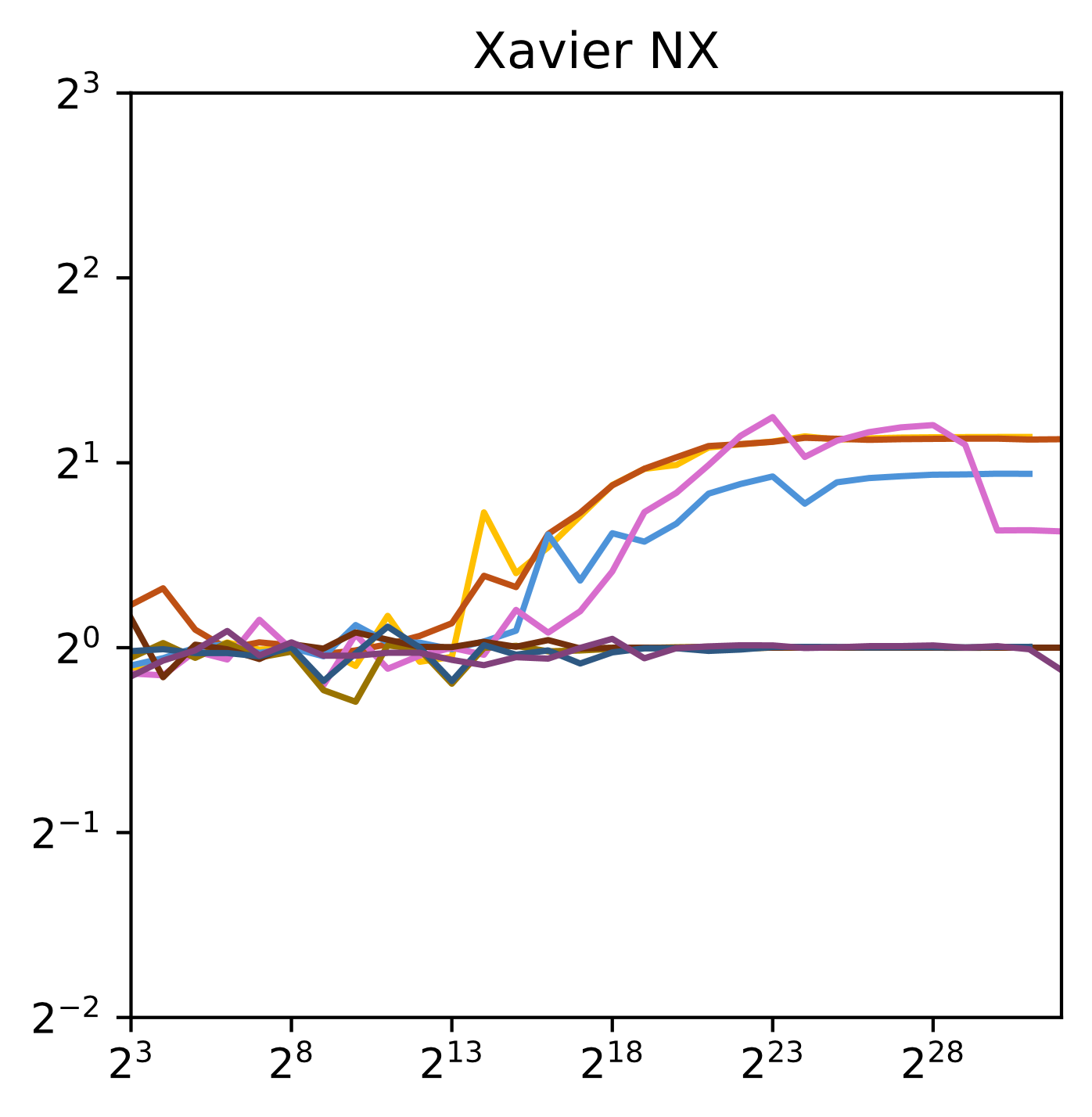


Abbildung 8. Verhältnis der Ausführungsdauer bei 27 Bins zur Ausführungsdauer bei 128 Bins in Abhängigkeit von der Eingabegröße. Der Quotient ist auf der vertikalen Achse aufgetragen, die Eingabegröße auf der horizontalen Achse. Die helleren Kurven („rand“) repräsentieren die Ergebnisse bei pseudo-zufälligen Eingabedaten, die dunkleren Kurven („const“) bei konstanten Eingabedaten.

Anhang A. Ausgabe von deviceQuery für die verwendete Hardware.

###### Xavier NX

[…]

Device 0: "Xavier"

CUDA Driver Version / Runtime Version 10.2 / 10.2

CUDA Capability Major/Minor version number: 7.2

Total amount of global memory: 15827 MBytes (16596041728 bytes)

( 6) Multiprocessors, ( 64) CUDA Cores/MP: 384 CUDA Cores

GPU Max Clock rate: 1109 MHz (1.11 GHz)

Memory Clock rate: 1109 Mhz

Memory Bus Width: 256-bit

L2 Cache Size: 524288 bytes

Maximum Texture Dimension Size (x,y,z) 1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)

Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers

Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Max dimension size of a thread block (x,y,z): (1024, 1024, 64)

Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 1 copy engine(s)

Run time limit on kernels: No

Integrated GPU sharing Host Memory: Yes

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled

Device supports Unified Addressing (UVA): Yes

Device supports Compute Preemption: Yes

Supports Cooperative Kernel Launch: Yes

Supports MultiDevice Co-op Kernel Launch: Yes

Device PCI Domain ID / Bus ID / location ID: 0 / 0 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.2, CUDA Runtime Version = 10.2, NumDevs = 1

Result = PASS

###### Tesla V100

[…]

Device 0: "Tesla V100-SXM2-32GB"

CUDA Driver Version / Runtime Version 12.4 / 12.3

CUDA Capability Major/Minor version number: 7.0

Total amount of global memory: 32494 MBytes (34072559616 bytes)

(80) Multiprocessors, ( 64) CUDA Cores/MP: 5120 CUDA Cores

GPU Max Clock rate: 1530 MHz (1.53 GHz)

Memory Clock rate: 877 Mhz

Memory Bus Width: 4096-bit

L2 Cache Size: 6291456 bytes

Maximum Texture Dimension Size (x,y,z) 1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)

Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers

Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Max dimension size of a thread block (x,y,z): (1024, 1024, 64)

Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 4 copy engine(s)

Run time limit on kernels: No

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Enabled

Device supports Unified Addressing (UVA): Yes

Device supports Compute Preemption: Yes

Supports Cooperative Kernel Launch: Yes

Supports MultiDevice Co-op Kernel Launch: Yes

Device PCI Domain ID / Bus ID / location ID: 4 / 4 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

[…]

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.4, CUDA Runtime Version = 12.3, NumDevs = 4

Result = PASS

###### AMD Radeon RX 6800 XT

[…]

Device 0: "AMD Radeon RX 6800 XT"

HIP Driver Version / Runtime Version 50422.0 / 50422.0

HIP Capability Major/Minor version number: 10.3

Total amount of global memory: 16368 MBytes (17163091968 bytes)

MapSMtoCores for SM 10.3 is undefined. Default to use 64 Cores/SM

MapSMtoCores for SM 10.3 is undefined. Default to use 64 Cores/SM

(36) Multiprocessors, ( 64) HIP Cores/MP: 2304 HIP Cores

GPU Max Clock rate: 2575 MHz (2.58 GHz)

Memory Clock rate: 1000 Mhz

Memory Bus Width: 256-bit

L2 Cache Size: 4194304 bytes

Maximum Texture Dimension Size (x,y,z) 1D=(16384), 2D=(16384, 16384), 3D=(16384, 16384, 8192)

Total amount of constant memory: 2147483647 bytes

Total amount of shared memory per block: 65536 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Max dimension size of a thread block (x,y,z): (1024, 1024, 1024)

Max dimension size of a grid size (x,y,z): (2147483647, 2147483647, 2147483647)

Maximum memory pitch: 2147483647 bytes

Texture alignment: 256 bytes

Run time limit on kernels: No

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Device has ECC support: Disabled

Supports Cooperative Kernel Launch: No

Supports MultiDevice Co-op Kernel Launch: No

Device PCI Domain ID / Bus ID / location ID: 0 / 3 / 0

Compute Mode:

< Default (multiple host threads can use ::hipSetDevice() with device simultaneously) >

[…]

deviceQuery, HIP Driver = HIPRT, HIP Driver Version = 50422.0, HIP Runtime Version = 50422.0, NumDevs = 2

Result = PASS

Anhang B. Häufigkeit der einzelnen Zeichencodes in der Beispieldatei

|  |  |  |
| --- | --- | --- |
| **Code** | **Anzahl** | **relative Häufigkeit** |
| 10 | 37022 | 0.351 % |
| 32 (‘ ’) | 1749664 | 16.611 % |
| 44 (‘,’) | 114062 | 1.083 % |
| 46 (‘.’) | 92662 | 0.880 % |
| 65 (‘A’) | 23754 | 0.226 % |
| 67 (‘C’) | 2568 | 0.024 % |
| 68 (‘D’) | 8132 | 0.077 % |
| 76 (‘L’) | 47722 | 0.453 % |
| 78 (‘N’) | 2568 | 0.024 % |
| 83 (‘S’) | 23754 | 0.226 % |
| 85 (‘U’) | 5350 | 0.051 % |
| 97 (‘a’) | 725674 | 6.890 % |
| 98 (‘b’) | 92662 | 0.880 % |
| 99 (‘c’) | 231334 | 2.196 % |
| 100 (‘d’) | 422650 | 4.013 % |
| 101 (‘e’) | 987396 | 9.374 % |
| 102 (‘f’) | 32528 | 0.309 % |
| 103 (‘g’) | 135676 | 1.288 % |
| 104 (‘h’) | 16050 | 0.152 % |
| 105 (‘i’) | 702348 | 6.668 % |
| 106 (‘j’) | 23754 | 0.226 % |
| 107 (‘k’) | 47508 | 0.451 % |
| 108 (‘l’) | 419868 | 3.986 % |
| 109 (‘m’) | 520020 | 4.937 % |
| 110 (‘n’) | 407242 | 3.866 % |
| 111 (‘o’) | 693360 | 6.583 % |
| 112 (‘p’) | 175266 | 1.664 % |
| 113 (‘q’) | 61418 | 0.583 % |
| 114 (‘r’) | 518094 | 4.919 % |
| 115 (‘s’) | 603908 | 5.734 % |
| 116 (‘t’) | 824542 | 7.828 % |
| 117 (‘u’) | 584220 | 5.547 % |
| 118 (‘v’) | 117486 | 1.115 % |
| 119 (‘w’) | 5350 | 0.051 % |
| 120 (‘x’) | 10700 | 0.102 % |
| 121 (‘y’) | 52858 | 0.502 % |
| 122 (‘z’) | 13696 | 0.130 % |
| **Summe** | **10532866** | **100.000 %** |

Anhang C. Gemessene Zeiten für die Eingabedaten aus der Beispieldatei

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **128 Bins** | | | | | | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Transferzeit Device zu Host (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **Device** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| Xavier NX | global | 1.860 | 1.904 | 2.016 | 2.016 | 2.483 | 9.493 | 9.516 | 9.561 | 9.561 | 9.814 | 0.068 | 0.085 | 0.092 | 0.092 | 0.322 | 11.452 | 11.558 | 11.674 | 11.674 | 12.389 |
| global\_stride | 1.849 | 1.895 | 2.044 | 2.044 | 2.383 | 9.555 | 9.586 | 9.606 | 9.606 | 9.745 | 0.081 | 0.084 | 0.096 | 0.096 | 0.283 | 11.562 | 11.604 | 11.756 | 11.756 | 12.148 |
| private | 1.644 | 1.867 | 1.934 | 1.934 | 2.245 | 2.012 | 2.037 | 2.069 | 2.069 | 2.240 | 0.075 | 0.085 | 0.091 | 0.091 | 0.227 | 3.794 | 4.010 | 4.129 | 4.129 | 4.400 |
| private\_stride | 1.417 | 1.737 | 1.916 | 1.916 | 2.309 | 1.023 | 1.038 | 1.058 | 1.058 | 1.217 | 0.061 | 0.085 | 0.090 | 0.090 | 0.220 | 2.536 | 2.889 | 3.075 | 3.075 | 3.434 |
| Tesla V100 | global | 6.615 | 6.615 | 6.616 | 6.616 | 6.666 | 3.232 | 3.233 | 3.234 | 3.234 | 3.242 | 0.019 | 0.019 | 0.019 | 0.019 | 0.024 | 9.867 | 9.868 | 9.870 | 9.870 | 9.922 |
| global\_stride | 6.615 | 6.616 | 6.617 | 6.617 | 6.641 | 3.232 | 3.234 | 3.234 | 3.234 | 3.239 | 0.018 | 0.019 | 0.020 | 0.020 | 0.029 | 9.868 | 9.869 | 9.871 | 9.871 | 9.894 |
| private | 6.615 | 6.615 | 6.616 | 6.616 | 6.640 | 0.306 | 0.307 | 0.308 | 0.308 | 0.310 | 0.019 | 0.019 | 0.019 | 0.019 | 0.040 | 6.940 | 6.942 | 6.944 | 6.944 | 6.968 |
| private\_stride | 6.615 | 6.616 | 6.616 | 6.616 | 6.620 | 0.065 | 0.066 | 0.068 | 0.068 | 0.075 | 0.018 | 0.019 | 0.019 | 0.019 | 0.021 | 6.701 | 6.702 | 6.703 | 6.703 | 6.710 |
| Radeon RX 6800 XT | global | 1.170 | 1.184 | 1.194 | 1.194 | 11.573 | 1.032 | 1.033 | 1.034 | 1.034 | 1.039 | 0.019 | 0.021 | 0.023 | 0.023 | 0.351 | 2.226 | 2.242 | 2.251 | 2.251 | 12.957 |
| global\_stride | 1.162 | 1.178 | 1.190 | 1.190 | 2.217 | 1.035 | 1.037 | 1.040 | 1.040 | 1.045 | 0.018 | 0.020 | 0.023 | 0.023 | 0.029 | 2.225 | 2.239 | 2.252 | 2.252 | 3.273 |
| private | 1.171 | 1.175 | 1.179 | 1.179 | 2.088 | 0.112 | 0.114 | 0.115 | 0.115 | 0.118 | 0.018 | 0.020 | 0.023 | 0.023 | 0.030 | 1.307 | 1.312 | 1.317 | 1.317 | 2.219 |
| private\_stride | 1.166 | 1.174 | 1.181 | 1.181 | 2.060 | 0.065 | 0.066 | 0.067 | 0.067 | 0.070 | 0.018 | 0.020 | 0.023 | 0.023 | 0.029 | 1.255 | 1.262 | 1.271 | 1.271 | 2.143 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **27 Bins** | | | | | | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Transferzeit Device zu Host (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **Device** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| Xavier NX | global | 1.850 | 1.937 | 2.045 | 2.045 | 2.240 | 11.243 | 11.266 | 11.289 | 11.289 | 11.403 | 0.076 | 0.078 | 0.093 | 0.093 | 0.242 | 13.200 | 13.323 | 13.440 | 13.440 | 13.684 |
| global\_stride | 1.851 | 1.897 | 2.006 | 2.006 | 2.216 | 11.324 | 11.338 | 11.362 | 11.362 | 11.485 | 0.062 | 0.079 | 0.111 | 0.111 | 0.207 | 13.283 | 13.361 | 13.477 | 13.477 | 13.701 |
| private | 1.516 | 1.623 | 1.840 | 1.840 | 2.128 | 2.059 | 2.076 | 2.110 | 2.110 | 2.268 | 0.058 | 0.065 | 0.078 | 0.078 | 0.190 | 3.650 | 3.795 | 4.010 | 4.010 | 4.326 |
| private\_stride | 1.366 | 1.390 | 1.478 | 1.478 | 2.230 | 1.055 | 1.071 | 1.091 | 1.091 | 1.247 | 0.053 | 0.054 | 0.057 | 0.057 | 0.190 | 2.489 | 2.538 | 2.640 | 2.640 | 3.421 |
| Tesla V100 | global | 6.611 | 6.617 | 6.619 | 6.619 | 6.667 | 3.854 | 3.855 | 3.856 | 3.856 | 3.863 | 0.018 | 0.019 | 0.020 | 0.020 | 0.026 | 10.486 | 10.493 | 10.494 | 10.494 | 10.548 |
| global\_stride | 6.616 | 6.617 | 6.619 | 6.619 | 6.648 | 3.855 | 3.857 | 3.858 | 3.858 | 3.866 | 0.018 | 0.019 | 0.020 | 0.020 | 0.028 | 10.493 | 10.494 | 10.497 | 10.497 | 10.525 |
| private | 6.617 | 6.617 | 6.619 | 6.619 | 6.648 | 0.165 | 0.166 | 0.166 | 0.166 | 0.188 | 0.018 | 0.019 | 0.020 | 0.020 | 0.028 | 6.801 | 6.803 | 6.805 | 6.805 | 6.857 |
| private\_stride | 6.616 | 6.617 | 6.619 | 6.619 | 6.632 | 0.066 | 0.067 | 0.068 | 0.068 | 0.078 | 0.018 | 0.019 | 0.020 | 0.020 | 0.024 | 6.703 | 6.705 | 6.706 | 6.706 | 6.722 |
| Radeon RX 6800 XT | global | 1.237 | 1.240 | 1.245 | 1.245 | 2.544 | 1.270 | 1.271 | 1.272 | 1.272 | 1.502 | 0.019 | 0.021 | 0.024 | 0.024 | 0.403 | 2.531 | 2.535 | 2.541 | 2.541 | 4.432 |
| global\_stride | 1.227 | 1.239 | 1.244 | 1.244 | 2.128 | 1.273 | 1.274 | 1.275 | 1.275 | 1.312 | 0.019 | 0.022 | 0.024 | 0.024 | 0.029 | 2.525 | 2.539 | 2.544 | 2.544 | 3.426 |
| private | 1.216 | 1.224 | 1.230 | 1.230 | 2.132 | 0.115 | 0.117 | 0.118 | 0.118 | 0.121 | 0.020 | 0.021 | 0.023 | 0.023 | 0.031 | 1.355 | 1.366 | 1.372 | 1.372 | 2.268 |
| private\_stride | 1.219 | 1.225 | 1.231 | 1.231 | 2.140 | 0.069 | 0.071 | 0.072 | 0.072 | 0.074 | 0.019 | 0.021 | 0.023 | 0.023 | 0.029 | 1.312 | 1.319 | 1.326 | 1.326 | 2.230 |

Anhang D. Gemessene Zeiten für die generierten Eingabedaten

Es ist . Dargestellt sind nur die Messergebnisse für n = 8, 16, 24, 32. Auf die Darstellung der Zeiten für den Transfer vom Device zum Host wurde verzichtet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Xavier NX, pseudozufällige Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.018 | 0.018 | 0.019 | 0.019 | 0.087 | 0.038 | 0.049 | 0.054 | 0.054 | 0.111 | 0.107 | 0.112 | 0.128 | 0.128 | 0.238 |
|  | atomic\_global\_stride | 0.018 | 0.018 | 0.019 | 0.019 | 0.060 | 0.059 | 0.071 | 0.085 | 0.085 | 0.123 | 0.118 | 0.138 | 0.152 | 0.152 | 0.255 |
|  | atomic\_private | 0.018 | 0.018 | 0.019 | 0.019 | 0.056 | 0.048 | 0.051 | 0.054 | 0.054 | 0.182 | 0.105 | 0.112 | 0.132 | 0.132 | 0.267 |
|  | atomic\_private\_stride | 0.018 | 0.018 | 0.019 | 0.019 | 0.111 | 0.075 | 0.088 | 0.103 | 0.103 | 0.174 | 0.133 | 0.156 | 0.174 | 0.174 | 0.357 |
| 16 | atomic\_global | 0.037 | 0.038 | 0.039 | 0.039 | 0.306 | 0.116 | 0.118 | 0.128 | 0.128 | 0.206 | 0.194 | 0.196 | 0.219 | 0.219 | 0.570 |
|  | atomic\_global\_stride | 0.037 | 0.038 | 0.039 | 0.039 | 0.085 | 0.122 | 0.126 | 0.130 | 0.130 | 0.209 | 0.203 | 0.206 | 0.220 | 0.220 | 0.338 |
|  | atomic\_private | 0.037 | 0.038 | 0.040 | 0.040 | 0.079 | 0.051 | 0.051 | 0.056 | 0.056 | 0.140 | 0.128 | 0.129 | 0.152 | 0.152 | 0.253 |
|  | atomic\_private\_stride | 0.037 | 0.038 | 0.039 | 0.039 | 0.080 | 0.077 | 0.104 | 0.107 | 0.107 | 0.199 | 0.155 | 0.184 | 0.204 | 0.204 | 0.333 |
| 24 | atomic\_global | 2.969 | 3.012 | 3.108 | 3.108 | 3.458 | 12.260 | 12.268 | 12.284 | 12.284 | 12.428 | 15.321 | 15.395 | 15.485 | 15.485 | 15.822 |
|  | atomic\_global\_stride | 2.844 | 2.881 | 2.985 | 2.985 | 3.284 | 12.372 | 12.386 | 12.403 | 12.403 | 12.688 | 15.329 | 15.386 | 15.468 | 15.468 | 15.745 |
|  | atomic\_private | 2.765 | 2.849 | 2.938 | 2.938 | 3.219 | 3.207 | 3.216 | 3.231 | 3.231 | 3.297 | 6.092 | 6.161 | 6.254 | 6.254 | 6.520 |
|  | atomic\_private\_stride | 2.287 | 2.453 | 2.797 | 2.797 | 3.062 | 1.889 | 1.894 | 1.909 | 1.909 | 2.018 | 4.279 | 4.474 | 4.769 | 4.769 | 5.045 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 600.253 | 601.015 | 602.647 | 602.647 | 609.545 | 3134.966 | 3139.629 | 3146.011 | 3146.011 | 3172.035 | 3738.008 | 3741.836 | 3749.600 | 3749.600 | 3774.030 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 568.542 | 568.940 | 570.526 | 570.526 | 617.709 | 579.631 | 579.976 | 580.133 | 580.133 | 580.493 | 1148.526 | 1149.107 | 1150.832 | 1150.832 | 1198.083 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Xavier NX, konstante Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.017 | 0.019 | 0.020 | 0.020 | 0.089 | 0.039 | 0.050 | 0.057 | 0.057 | 0.119 | 0.108 | 0.112 | 0.131 | 0.131 | 0.287 |
|  | atomic\_global\_stride | 0.017 | 0.018 | 0.019 | 0.019 | 0.061 | 0.059 | 0.071 | 0.085 | 0.085 | 0.123 | 0.116 | 0.133 | 0.151 | 0.151 | 0.231 |
|  | atomic\_private | 0.017 | 0.018 | 0.019 | 0.019 | 0.073 | 0.048 | 0.052 | 0.055 | 0.055 | 0.086 | 0.105 | 0.110 | 0.126 | 0.126 | 0.216 |
|  | atomic\_private\_stride | 0.017 | 0.017 | 0.018 | 0.018 | 0.067 | 0.074 | 0.088 | 0.103 | 0.103 | 0.148 | 0.132 | 0.156 | 0.165 | 0.165 | 0.288 |
| 16 | atomic\_global | 0.038 | 0.039 | 0.045 | 0.045 | 0.365 | 0.199 | 0.204 | 0.222 | 0.222 | 0.336 | 0.278 | 0.291 | 0.318 | 0.318 | 0.739 |
|  | atomic\_global\_stride | 0.038 | 0.040 | 0.048 | 0.048 | 0.107 | 0.206 | 0.211 | 0.225 | 0.225 | 0.290 | 0.287 | 0.297 | 0.334 | 0.334 | 0.445 |
|  | atomic\_private | 0.037 | 0.038 | 0.039 | 0.039 | 0.080 | 0.073 | 0.098 | 0.102 | 0.102 | 0.180 | 0.152 | 0.179 | 0.203 | 0.203 | 0.306 |
|  | atomic\_private\_stride | 0.038 | 0.038 | 0.040 | 0.040 | 0.082 | 0.117 | 0.122 | 0.127 | 0.127 | 0.190 | 0.199 | 0.202 | 0.219 | 0.219 | 0.317 |
| 24 | atomic\_global | 2.830 | 2.875 | 2.984 | 2.984 | 3.557 | 33.876 | 33.883 | 33.895 | 33.895 | 33.981 | 36.813 | 36.847 | 36.968 | 36.968 | 37.635 |
|  | atomic\_global\_stride | 2.857 | 2.936 | 3.031 | 3.031 | 3.320 | 33.638 | 33.648 | 33.667 | 33.667 | 33.757 | 36.590 | 36.679 | 36.782 | 36.782 | 37.075 |
|  | atomic\_private | 2.809 | 2.858 | 2.959 | 2.959 | 3.314 | 7.873 | 7.882 | 7.906 | 7.906 | 8.294 | 10.779 | 10.854 | 10.959 | 10.959 | 11.546 |
|  | atomic\_private\_stride | 2.816 | 2.890 | 2.970 | 2.970 | 3.407 | 6.123 | 6.132 | 6.150 | 6.150 | 6.241 | 9.028 | 9.101 | 9.199 | 9.199 | 9.658 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 579.709 | 595.695 | 600.437 | 600.437 | 608.195 | 8582.007 | 8582.134 | 8582.925 | 8582.925 | 8594.232 | 9162.101 | 9181.184 | 9183.991 | 9183.991 | 9201.661 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 584.012 | 597.749 | 599.504 | 599.504 | 611.621 | 1537.560 | 1558.552 | 1592.471 | 1592.471 | 1670.969 | 2136.375 | 2158.716 | 2191.524 | 2191.524 | 2269.287 |
|  |  | **Tesla V100, pseudozufällige Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.011 | 0.012 | 0.012 | 0.012 | 0.025 | 0.015 | 0.015 | 0.016 | 0.016 | 0.019 | 0.043 | 0.044 | 0.045 | 0.045 | 0.067 |
|  | atomic\_global\_stride | 0.011 | 0.012 | 0.012 | 0.012 | 0.015 | 0.016 | 0.016 | 0.017 | 0.017 | 0.022 | 0.044 | 0.045 | 0.046 | 0.046 | 0.051 |
|  | atomic\_private | 0.011 | 0.012 | 0.012 | 0.012 | 0.014 | 0.015 | 0.015 | 0.016 | 0.016 | 0.019 | 0.043 | 0.044 | 0.045 | 0.045 | 0.049 |
|  | atomic\_private\_stride | 0.011 | 0.012 | 0.012 | 0.012 | 0.015 | 0.022 | 0.023 | 0.023 | 0.023 | 0.025 | 0.050 | 0.051 | 0.052 | 0.052 | 0.055 |
| 16 | atomic\_global | 0.054 | 0.054 | 0.055 | 0.055 | 0.068 | 0.032 | 0.032 | 0.033 | 0.033 | 0.038 | 0.102 | 0.103 | 0.105 | 0.105 | 0.128 |
|  | atomic\_global\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.060 | 0.032 | 0.032 | 0.032 | 0.032 | 0.034 | 0.102 | 0.102 | 0.104 | 0.104 | 0.114 |
|  | atomic\_private | 0.054 | 0.054 | 0.054 | 0.054 | 0.056 | 0.017 | 0.017 | 0.017 | 0.017 | 0.022 | 0.087 | 0.088 | 0.089 | 0.089 | 0.093 |
|  | atomic\_private\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.056 | 0.023 | 0.023 | 0.024 | 0.024 | 0.025 | 0.092 | 0.093 | 0.095 | 0.095 | 0.098 |
| 24 | atomic\_global | 10.502 | 10.503 | 10.504 | 10.504 | 10.548 | 4.167 | 4.168 | 4.168 | 4.168 | 4.179 | 14.689 | 14.690 | 14.692 | 14.692 | 14.742 |
|  | atomic\_global\_stride | 10.503 | 10.503 | 10.505 | 10.505 | 10.532 | 4.169 | 4.169 | 4.171 | 4.171 | 4.205 | 14.691 | 14.691 | 14.695 | 14.695 | 14.756 |
|  | atomic\_private | 10.502 | 10.503 | 10.504 | 10.504 | 10.524 | 0.480 | 0.480 | 0.481 | 0.481 | 0.488 | 11.001 | 11.002 | 11.004 | 11.004 | 11.025 |
|  | atomic\_private\_stride | 10.503 | 10.503 | 10.504 | 10.504 | 10.516 | 0.089 | 0.090 | 0.092 | 0.092 | 0.107 | 10.611 | 10.613 | 10.615 | 10.615 | 10.642 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 2673.875 | 2673.885 | 2673.893 | 2673.893 | 2675.952 | 1061.546 | 1061.553 | 1061.564 | 1061.564 | 1070.549 | 3735.467 | 3735.475 | 3735.492 | 3735.492 | 3744.480 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 2673.872 | 2673.881 | 2673.892 | 2673.892 | 2676.035 | 18.487 | 18.518 | 18.553 | 18.553 | 18.607 | 2692.397 | 2692.444 | 2692.478 | 2692.478 | 2694.614 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Tesla V100, konstante Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.012 | 0.012 | 0.012 | 0.012 | 0.024 | 0.015 | 0.015 | 0.016 | 0.016 | 0.018 | 0.043 | 0.044 | 0.045 | 0.045 | 0.066 |
|  | atomic\_global\_stride | 0.012 | 0.012 | 0.012 | 0.012 | 0.017 | 0.016 | 0.016 | 0.017 | 0.017 | 0.019 | 0.044 | 0.045 | 0.046 | 0.046 | 0.054 |
|  | atomic\_private | 0.012 | 0.012 | 0.012 | 0.012 | 0.015 | 0.015 | 0.015 | 0.016 | 0.016 | 0.019 | 0.043 | 0.044 | 0.045 | 0.045 | 0.052 |
|  | atomic\_private\_stride | 0.012 | 0.012 | 0.012 | 0.012 | 0.014 | 0.023 | 0.023 | 0.023 | 0.023 | 0.025 | 0.051 | 0.051 | 0.052 | 0.052 | 0.056 |
| 16 | atomic\_global | 0.054 | 0.054 | 0.055 | 0.055 | 0.069 | 0.060 | 0.060 | 0.061 | 0.061 | 0.063 | 0.131 | 0.131 | 0.134 | 0.134 | 0.158 |
|  | atomic\_global\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.057 | 0.060 | 0.060 | 0.061 | 0.061 | 0.062 | 0.130 | 0.131 | 0.133 | 0.133 | 0.139 |
|  | atomic\_private | 0.053 | 0.054 | 0.054 | 0.054 | 0.055 | 0.017 | 0.017 | 0.018 | 0.018 | 0.018 | 0.087 | 0.088 | 0.089 | 0.089 | 0.091 |
|  | atomic\_private\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.056 | 0.023 | 0.023 | 0.024 | 0.024 | 0.025 | 0.092 | 0.094 | 0.095 | 0.095 | 0.097 |
| 24 | atomic\_global | 10.505 | 10.506 | 10.507 | 10.507 | 10.555 | 11.626 | 11.627 | 11.628 | 11.628 | 11.653 | 22.151 | 22.152 | 22.155 | 22.155 | 22.217 |
|  | atomic\_global\_stride | 10.505 | 10.506 | 10.507 | 10.507 | 10.531 | 11.626 | 11.626 | 11.627 | 11.627 | 11.633 | 22.150 | 22.152 | 22.153 | 22.153 | 22.182 |
|  | atomic\_private | 10.505 | 10.506 | 10.506 | 10.506 | 10.521 | 0.480 | 0.481 | 0.482 | 0.482 | 0.487 | 11.005 | 11.006 | 11.007 | 11.007 | 11.022 |
|  | atomic\_private\_stride | 10.505 | 10.506 | 10.506 | 10.506 | 10.517 | 0.174 | 0.175 | 0.185 | 0.185 | 0.190 | 10.698 | 10.700 | 10.710 | 10.710 | 10.722 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 2673.885 | 2673.894 | 2673.903 | 2673.903 | 2676.022 | 2970.182 | 2970.189 | 2970.206 | 2970.206 | 2979.272 | 5644.111 | 5644.129 | 5644.149 | 5644.149 | 5653.215 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 2673.888 | 2673.893 | 2673.903 | 2673.903 | 2676.014 | 45.312 | 45.357 | 45.373 | 45.373 | 45.428 | 2719.243 | 2719.287 | 2719.311 | 2719.311 | 2721.460 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Radeon RX 6800 XT, pseudozufällige Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.021 | 0.023 | 0.025 | 0.025 | 0.048 | 0.024 | 0.025 | 0.027 | 0.027 | 0.061 | 0.063 | 0.071 | 0.074 | 0.074 | 0.499 |
|  | atomic\_global\_stride | 0.021 | 0.023 | 0.025 | 0.025 | 0.032 | 0.025 | 0.026 | 0.028 | 0.028 | 0.132 | 0.065 | 0.071 | 0.075 | 0.075 | 0.181 |
|  | atomic\_private | 0.018 | 0.023 | 0.025 | 0.025 | 0.030 | 0.022 | 0.025 | 0.027 | 0.027 | 0.030 | 0.056 | 0.069 | 0.074 | 0.074 | 0.136 |
|  | atomic\_private\_stride | 0.019 | 0.023 | 0.025 | 0.025 | 0.031 | 0.024 | 0.027 | 0.029 | 0.029 | 0.030 | 0.059 | 0.073 | 0.076 | 0.076 | 0.087 |
| 16 | atomic\_global | 0.022 | 0.023 | 0.024 | 0.024 | 0.438 | 0.029 | 0.029 | 0.031 | 0.031 | 0.047 | 0.069 | 0.074 | 0.078 | 0.078 | 0.893 |
|  | atomic\_global\_stride | 0.022 | 0.023 | 0.024 | 0.024 | 0.235 | 0.028 | 0.029 | 0.031 | 0.031 | 0.033 | 0.069 | 0.074 | 0.078 | 0.078 | 0.290 |
|  | atomic\_private | 0.020 | 0.023 | 0.024 | 0.024 | 0.030 | 0.023 | 0.026 | 0.028 | 0.028 | 0.091 | 0.064 | 0.071 | 0.075 | 0.075 | 0.141 |
|  | atomic\_private\_stride | 0.020 | 0.023 | 0.024 | 0.024 | 0.030 | 0.024 | 0.027 | 0.028 | 0.028 | 0.030 | 0.061 | 0.071 | 0.075 | 0.075 | 0.084 |
| 24 | atomic\_global | 1.610 | 1.742 | 3.095 | 3.095 | 3.295 | 1.201 | 1.201 | 1.203 | 1.203 | 1.403 | 2.833 | 2.966 | 4.320 | 4.320 | 4.726 |
|  | atomic\_global\_stride | 1.605 | 2.922 | 3.094 | 3.094 | 3.415 | 1.200 | 1.204 | 1.206 | 1.206 | 1.507 | 2.831 | 4.179 | 4.319 | 4.319 | 4.639 |
|  | atomic\_private | 1.588 | 1.593 | 1.607 | 1.607 | 2.658 | 0.154 | 0.157 | 0.159 | 0.159 | 0.165 | 1.767 | 1.772 | 1.786 | 1.786 | 2.835 |
|  | atomic\_private\_stride | 1.588 | 1.591 | 1.606 | 1.606 | 2.503 | 0.085 | 0.086 | 0.088 | 0.088 | 0.094 | 1.695 | 1.699 | 1.715 | 1.715 | 2.605 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 206.038 | 206.718 | 207.263 | 207.263 | 207.949 | 307.547 | 309.423 | 311.516 | 311.516 | 313.280 | 514.998 | 516.651 | 518.834 | 518.834 | 520.519 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 205.586 | 205.896 | 206.507 | 206.507 | 213.529 | 22.198 | 22.477 | 22.753 | 22.753 | 23.354 | 228.221 | 228.598 | 229.312 | 229.312 | 236.515 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Radeon RX 6800 XT, konstante Daten, 128 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.023 | 0.023 | 0.025 | 0.025 | 0.048 | 0.025 | 0.027 | 0.028 | 0.028 | 0.059 | 0.070 | 0.073 | 0.076 | 0.076 | 0.485 |
|  | atomic\_global\_stride | 0.022 | 0.023 | 0.026 | 0.026 | 0.039 | 0.026 | 0.026 | 0.028 | 0.028 | 0.100 | 0.068 | 0.073 | 0.077 | 0.077 | 0.148 |
|  | atomic\_private | 0.019 | 0.023 | 0.025 | 0.025 | 0.030 | 0.022 | 0.026 | 0.027 | 0.027 | 0.029 | 0.059 | 0.073 | 0.075 | 0.075 | 0.144 |
|  | atomic\_private\_stride | 0.018 | 0.024 | 0.025 | 0.025 | 0.030 | 0.024 | 0.027 | 0.029 | 0.029 | 0.030 | 0.059 | 0.074 | 0.077 | 0.077 | 0.092 |
| 16 | atomic\_global | 0.022 | 0.023 | 0.026 | 0.026 | 0.445 | 0.056 | 0.056 | 0.058 | 0.058 | 0.062 | 0.101 | 0.103 | 0.107 | 0.107 | 0.962 |
|  | atomic\_global\_stride | 0.022 | 0.023 | 0.024 | 0.024 | 0.144 | 0.051 | 0.056 | 0.058 | 0.058 | 0.062 | 0.096 | 0.103 | 0.106 | 0.106 | 0.227 |
|  | atomic\_private | 0.020 | 0.023 | 0.025 | 0.025 | 0.029 | 0.026 | 0.028 | 0.030 | 0.030 | 0.078 | 0.063 | 0.074 | 0.077 | 0.077 | 0.125 |
|  | atomic\_private\_stride | 0.020 | 0.023 | 0.025 | 0.025 | 0.031 | 0.025 | 0.028 | 0.030 | 0.030 | 0.063 | 0.063 | 0.074 | 0.077 | 0.077 | 0.111 |
| 24 | atomic\_global | 2.494 | 2.504 | 2.519 | 2.519 | 2.947 | 6.875 | 6.876 | 7.145 | 7.145 | 7.383 | 9.401 | 9.416 | 9.680 | 9.680 | 10.683 |
|  | atomic\_global\_stride | 2.491 | 2.502 | 2.516 | 2.516 | 2.619 | 6.913 | 6.924 | 7.187 | 7.187 | 7.213 | 9.440 | 9.459 | 9.720 | 9.720 | 9.836 |
|  | atomic\_private | 1.620 | 1.625 | 1.641 | 1.641 | 2.533 | 0.396 | 0.398 | 0.399 | 0.399 | 0.402 | 2.042 | 2.048 | 2.063 | 2.063 | 2.950 |
|  | atomic\_private\_stride | 1.621 | 1.629 | 1.660 | 1.660 | 2.633 | 0.261 | 0.263 | 0.264 | 0.264 | 0.272 | 1.904 | 1.916 | 1.947 | 1.947 | 2.916 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 226.236 | 226.475 | 226.892 | 226.892 | 228.848 | 1794.445 | 1796.035 | 1803.207 | 1803.207 | 1816.006 | 2021.211 | 2023.682 | 2030.177 | 2030.177 | 2043.649 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 226.386 | 226.733 | 227.304 | 227.304 | 229.530 | 78.592 | 80.093 | 80.811 | 80.811 | 82.160 | 306.108 | 307.260 | 308.217 | 308.217 | 310.558 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Xavier NX, pseudozufällige Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.018 | 0.018 | 0.022 | 0.022 | 0.064 | 0.042 | 0.049 | 0.054 | 0.054 | 0.128 | 0.107 | 0.111 | 0.131 | 0.131 | 0.268 |
|  | atomic\_global\_stride | 0.018 | 0.018 | 0.022 | 0.022 | 0.061 | 0.055 | 0.072 | 0.086 | 0.086 | 0.129 | 0.115 | 0.143 | 0.157 | 0.157 | 0.240 |
|  | atomic\_private | 0.018 | 0.018 | 0.020 | 0.020 | 0.063 | 0.037 | 0.049 | 0.054 | 0.054 | 0.114 | 0.107 | 0.112 | 0.131 | 0.131 | 0.244 |
|  | atomic\_private\_stride | 0.018 | 0.019 | 0.020 | 0.020 | 0.065 | 0.072 | 0.084 | 0.103 | 0.103 | 0.166 | 0.130 | 0.151 | 0.182 | 0.182 | 0.331 |
| 16 | atomic\_global | 0.035 | 0.037 | 0.039 | 0.039 | 0.435 | 0.175 | 0.180 | 0.186 | 0.186 | 0.260 | 0.251 | 0.256 | 0.277 | 0.277 | 0.697 |
|  | atomic\_global\_stride | 0.037 | 0.038 | 0.044 | 0.044 | 0.072 | 0.183 | 0.189 | 0.200 | 0.200 | 0.274 | 0.260 | 0.267 | 0.297 | 0.297 | 0.383 |
|  | atomic\_private | 0.037 | 0.037 | 0.038 | 0.038 | 0.078 | 0.058 | 0.072 | 0.086 | 0.086 | 0.150 | 0.135 | 0.151 | 0.164 | 0.164 | 0.258 |
|  | atomic\_private\_stride | 0.036 | 0.037 | 0.039 | 0.039 | 0.103 | 0.084 | 0.108 | 0.113 | 0.113 | 0.163 | 0.161 | 0.187 | 0.205 | 0.205 | 0.276 |
| 24 | atomic\_global | 2.975 | 3.020 | 3.110 | 3.110 | 3.383 | 27.121 | 27.126 | 27.135 | 27.135 | 27.195 | 30.184 | 30.230 | 30.329 | 30.329 | 30.635 |
|  | atomic\_global\_stride | 2.880 | 2.948 | 3.040 | 3.040 | 3.223 | 26.966 | 26.978 | 27.245 | 27.245 | 27.360 | 29.945 | 30.075 | 30.304 | 30.304 | 30.576 |
|  | atomic\_private | 2.872 | 2.968 | 3.062 | 3.062 | 3.396 | 5.510 | 5.528 | 5.544 | 5.544 | 5.931 | 8.477 | 8.584 | 8.703 | 8.703 | 9.158 |
|  | atomic\_private\_stride | 2.871 | 2.954 | 3.049 | 3.049 | 3.349 | 3.886 | 3.893 | 3.902 | 3.902 | 4.067 | 6.860 | 6.919 | 7.042 | 7.042 | 7.391 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 572.285 | 573.327 | 574.698 | 574.698 | 605.823 | 6874.745 | 6874.786 | 6875.043 | 6875.043 | 6882.925 | 7447.218 | 7448.568 | 7451.293 | 7451.293 | 7480.732 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 565.509 | 566.793 | 572.418 | 572.418 | 610.911 | 897.024 | 897.138 | 897.272 | 897.272 | 897.577 | 1462.915 | 1464.115 | 1469.853 | 1469.853 | 1508.225 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Xavier NX, konstante Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.018 | 0.018 | 0.020 | 0.020 | 0.084 | 0.039 | 0.053 | 0.056 | 0.056 | 0.127 | 0.105 | 0.112 | 0.132 | 0.132 | 0.305 |
|  | atomic\_global\_stride | 0.018 | 0.018 | 0.019 | 0.019 | 0.093 | 0.056 | 0.071 | 0.086 | 0.086 | 0.126 | 0.115 | 0.135 | 0.155 | 0.155 | 0.249 |
|  | atomic\_private | 0.017 | 0.018 | 0.022 | 0.022 | 0.075 | 0.044 | 0.050 | 0.055 | 0.055 | 0.121 | 0.108 | 0.112 | 0.135 | 0.135 | 0.242 |
|  | atomic\_private\_stride | 0.018 | 0.018 | 0.020 | 0.020 | 0.078 | 0.074 | 0.098 | 0.105 | 0.105 | 0.194 | 0.135 | 0.160 | 0.184 | 0.184 | 0.291 |
| 16 | atomic\_global | 0.038 | 0.039 | 0.047 | 0.047 | 0.323 | 0.201 | 0.207 | 0.219 | 0.219 | 0.288 | 0.280 | 0.289 | 0.326 | 0.326 | 0.667 |
|  | atomic\_global\_stride | 0.039 | 0.040 | 0.049 | 0.049 | 0.085 | 0.206 | 0.211 | 0.231 | 0.231 | 0.317 | 0.287 | 0.298 | 0.340 | 0.340 | 0.431 |
|  | atomic\_private | 0.037 | 0.038 | 0.039 | 0.039 | 0.097 | 0.073 | 0.094 | 0.101 | 0.101 | 0.192 | 0.150 | 0.173 | 0.188 | 0.188 | 0.302 |
|  | atomic\_private\_stride | 0.037 | 0.037 | 0.039 | 0.039 | 0.091 | 0.109 | 0.118 | 0.122 | 0.122 | 0.194 | 0.186 | 0.198 | 0.210 | 0.210 | 0.316 |
| 24 | atomic\_global | 2.891 | 2.990 | 3.117 | 3.117 | 3.504 | 33.631 | 33.666 | 33.912 | 33.912 | 34.016 | 36.684 | 36.881 | 37.107 | 37.107 | 37.568 |
|  | atomic\_global\_stride | 2.608 | 2.881 | 2.993 | 2.993 | 3.304 | 33.629 | 33.647 | 33.690 | 33.690 | 34.039 | 36.564 | 36.644 | 36.792 | 36.792 | 37.306 |
|  | atomic\_private | 2.817 | 2.860 | 2.967 | 2.967 | 3.246 | 7.905 | 7.917 | 7.931 | 7.931 | 8.236 | 10.820 | 10.882 | 10.978 | 10.978 | 11.320 |
|  | atomic\_private\_stride | 2.805 | 2.844 | 2.948 | 2.948 | 3.251 | 6.118 | 6.128 | 6.143 | 6.143 | 6.249 | 9.027 | 9.077 | 9.172 | 9.172 | 9.588 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 566.573 | 567.751 | 575.397 | 575.397 | 586.327 | 8581.193 | 8581.416 | 8582.334 | 8582.334 | 8594.198 | 9148.009 | 9150.052 | 9158.773 | 9158.773 | 9176.809 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 573.721 | 574.259 | 576.022 | 576.022 | 628.082 | 1458.498 | 1463.966 | 1464.023 | 1464.023 | 1489.438 | 2033.499 | 2038.479 | 2041.332 | 2041.332 | 2092.294 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Tesla V100, pseudozufällige Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.012 | 0.012 | 0.012 | 0.012 | 0.024 | 0.015 | 0.015 | 0.016 | 0.016 | 0.018 | 0.043 | 0.044 | 0.045 | 0.045 | 0.067 |
|  | atomic\_global\_stride | 0.012 | 0.012 | 0.012 | 0.012 | 0.015 | 0.016 | 0.016 | 0.017 | 0.017 | 0.018 | 0.044 | 0.045 | 0.046 | 0.046 | 0.051 |
|  | atomic\_private | 0.012 | 0.012 | 0.012 | 0.012 | 0.014 | 0.015 | 0.015 | 0.016 | 0.016 | 0.018 | 0.043 | 0.044 | 0.045 | 0.045 | 0.049 |
|  | atomic\_private\_stride | 0.012 | 0.012 | 0.012 | 0.012 | 0.013 | 0.018 | 0.018 | 0.019 | 0.019 | 0.022 | 0.046 | 0.047 | 0.048 | 0.048 | 0.052 |
| 16 | atomic\_global | 0.054 | 0.055 | 0.056 | 0.056 | 0.068 | 0.051 | 0.052 | 0.052 | 0.052 | 0.054 | 0.122 | 0.124 | 0.125 | 0.125 | 0.147 |
|  | atomic\_global\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.060 | 0.051 | 0.052 | 0.052 | 0.052 | 0.056 | 0.122 | 0.123 | 0.124 | 0.124 | 0.129 |
|  | atomic\_private | 0.053 | 0.054 | 0.055 | 0.055 | 0.056 | 0.016 | 0.016 | 0.017 | 0.017 | 0.018 | 0.087 | 0.087 | 0.089 | 0.089 | 0.092 |
|  | atomic\_private\_stride | 0.053 | 0.054 | 0.055 | 0.055 | 0.061 | 0.018 | 0.019 | 0.019 | 0.019 | 0.020 | 0.089 | 0.090 | 0.091 | 0.091 | 0.097 |
| 24 | atomic\_global | 10.504 | 10.504 | 10.505 | 10.505 | 10.549 | 9.315 | 9.316 | 9.317 | 9.317 | 9.326 | 19.839 | 19.840 | 19.841 | 19.841 | 19.890 |
|  | atomic\_global\_stride | 10.504 | 10.505 | 10.505 | 10.505 | 10.528 | 9.315 | 9.316 | 9.317 | 9.317 | 9.324 | 19.839 | 19.840 | 19.842 | 19.842 | 19.863 |
|  | atomic\_private | 10.504 | 10.505 | 10.505 | 10.505 | 10.523 | 0.252 | 0.253 | 0.254 | 0.254 | 0.260 | 10.776 | 10.777 | 10.778 | 10.778 | 10.797 |
|  | atomic\_private\_stride | 10.504 | 10.504 | 10.505 | 10.505 | 10.510 | 0.120 | 0.123 | 0.125 | 0.125 | 0.130 | 10.645 | 10.647 | 10.650 | 10.650 | 10.657 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 2673.937 | 2673.944 | 2673.954 | 2673.954 | 2676.044 | 2379.249 | 2379.255 | 2379.263 | 2379.263 | 2388.235 | 5053.223 | 5053.238 | 5053.256 | 5053.256 | 5062.234 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 2673.934 | 2673.942 | 2673.950 | 2673.950 | 2676.086 | 27.624 | 27.652 | 27.667 | 27.667 | 27.707 | 2701.602 | 2701.629 | 2701.651 | 2701.651 | 2703.804 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Tesla V100, konstante Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.012 | 0.012 | 0.012 | 0.012 | 0.026 | 0.015 | 0.015 | 0.016 | 0.016 | 0.018 | 0.043 | 0.044 | 0.045 | 0.045 | 0.068 |
|  | atomic\_global\_stride | 0.011 | 0.012 | 0.012 | 0.012 | 0.015 | 0.016 | 0.016 | 0.017 | 0.017 | 0.018 | 0.045 | 0.045 | 0.046 | 0.046 | 0.050 |
|  | atomic\_private | 0.011 | 0.012 | 0.012 | 0.012 | 0.015 | 0.015 | 0.015 | 0.016 | 0.016 | 0.018 | 0.043 | 0.044 | 0.045 | 0.045 | 0.052 |
|  | atomic\_private\_stride | 0.011 | 0.012 | 0.012 | 0.012 | 0.014 | 0.018 | 0.018 | 0.019 | 0.019 | 0.022 | 0.046 | 0.047 | 0.048 | 0.048 | 0.052 |
| 16 | atomic\_global | 0.054 | 0.054 | 0.055 | 0.055 | 0.069 | 0.060 | 0.060 | 0.061 | 0.061 | 0.065 | 0.131 | 0.132 | 0.133 | 0.133 | 0.155 |
|  | atomic\_global\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.057 | 0.060 | 0.060 | 0.061 | 0.061 | 0.063 | 0.130 | 0.131 | 0.132 | 0.132 | 0.137 |
|  | atomic\_private | 0.053 | 0.054 | 0.054 | 0.054 | 0.055 | 0.016 | 0.016 | 0.017 | 0.017 | 0.018 | 0.086 | 0.087 | 0.088 | 0.088 | 0.094 |
|  | atomic\_private\_stride | 0.053 | 0.054 | 0.054 | 0.054 | 0.057 | 0.019 | 0.019 | 0.019 | 0.019 | 0.023 | 0.089 | 0.090 | 0.091 | 0.091 | 0.096 |
| 24 | atomic\_global | 10.507 | 10.508 | 10.516 | 10.516 | 11.364 | 11.626 | 11.627 | 11.630 | 11.630 | 15.949 | 22.153 | 22.155 | 22.166 | 22.166 | 27.340 |
|  | atomic\_global\_stride | 10.507 | 10.507 | 10.508 | 10.508 | 10.573 | 11.625 | 11.626 | 11.627 | 11.627 | 11.662 | 22.152 | 22.153 | 22.155 | 22.155 | 22.267 |
|  | atomic\_private | 10.506 | 10.507 | 10.508 | 10.508 | 10.528 | 0.253 | 0.254 | 0.255 | 0.255 | 0.262 | 10.779 | 10.781 | 10.782 | 10.782 | 10.805 |
|  | atomic\_private\_stride | 10.506 | 10.507 | 10.508 | 10.508 | 10.522 | 0.173 | 0.175 | 0.185 | 0.185 | 0.198 | 10.700 | 10.702 | 10.712 | 10.712 | 10.736 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 2673.560 | 2673.571 | 2673.915 | 2673.915 | 2675.945 | 2970.186 | 2970.192 | 2970.207 | 2970.207 | 2979.209 | 5643.794 | 5643.807 | 5644.156 | 5644.156 | 5653.195 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 2673.897 | 2673.904 | 2673.914 | 2673.914 | 2676.051 | 45.362 | 45.377 | 45.389 | 45.389 | 45.436 | 2719.295 | 2719.319 | 2719.335 | 2719.335 | 2721.519 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Radeon RX 6800 XT, pseudozufällige Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.020 | 0.023 | 0.025 | 0.025 | 0.055 | 0.029 | 0.030 | 0.031 | 0.031 | 0.055 | 0.070 | 0.075 | 0.078 | 0.078 | 0.481 |
|  | atomic\_global\_stride | 0.020 | 0.022 | 0.025 | 0.025 | 0.029 | 0.029 | 0.030 | 0.032 | 0.032 | 0.119 | 0.069 | 0.074 | 0.078 | 0.078 | 0.161 |
|  | atomic\_private | 0.019 | 0.022 | 0.025 | 0.025 | 0.030 | 0.026 | 0.029 | 0.031 | 0.031 | 0.033 | 0.061 | 0.074 | 0.078 | 0.078 | 0.140 |
|  | atomic\_private\_stride | 0.018 | 0.022 | 0.025 | 0.025 | 0.028 | 0.027 | 0.031 | 0.033 | 0.033 | 0.035 | 0.064 | 0.074 | 0.078 | 0.078 | 0.087 |
| 16 | atomic\_global | 0.021 | 0.023 | 0.024 | 0.024 | 0.583 | 0.046 | 0.046 | 0.048 | 0.048 | 0.051 | 0.088 | 0.092 | 0.095 | 0.095 | 1.146 |
|  | atomic\_global\_stride | 0.022 | 0.023 | 0.023 | 0.023 | 0.129 | 0.045 | 0.046 | 0.048 | 0.048 | 0.049 | 0.087 | 0.090 | 0.094 | 0.094 | 0.200 |
|  | atomic\_private | 0.020 | 0.023 | 0.023 | 0.023 | 0.028 | 0.028 | 0.031 | 0.033 | 0.033 | 0.075 | 0.065 | 0.075 | 0.079 | 0.079 | 0.124 |
|  | atomic\_private\_stride | 0.018 | 0.022 | 0.023 | 0.023 | 0.029 | 0.027 | 0.031 | 0.033 | 0.033 | 0.035 | 0.061 | 0.076 | 0.079 | 0.079 | 0.087 |
| 24 | atomic\_global | 2.247 | 2.253 | 2.264 | 2.264 | 2.695 | 4.491 | 4.493 | 4.759 | 4.759 | 4.985 | 6.765 | 6.778 | 7.040 | 7.040 | 8.032 |
|  | atomic\_global\_stride | 2.244 | 2.254 | 2.264 | 2.264 | 2.614 | 4.503 | 4.522 | 4.787 | 4.787 | 4.822 | 6.787 | 6.818 | 7.072 | 7.072 | 7.424 |
|  | atomic\_private | 1.379 | 1.381 | 1.386 | 1.386 | 2.272 | 0.259 | 0.263 | 0.264 | 0.264 | 0.265 | 1.663 | 1.667 | 1.674 | 1.674 | 2.552 |
|  | atomic\_private\_stride | 1.376 | 1.379 | 1.385 | 1.385 | 2.353 | 0.173 | 0.175 | 0.177 | 0.177 | 0.180 | 1.574 | 1.578 | 1.585 | 1.585 | 2.548 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 204.174 | 204.395 | 204.741 | 204.741 | 205.473 | 1166.275 | 1174.485 | 1179.170 | 1179.170 | 1189.967 | 1370.550 | 1379.748 | 1383.837 | 1383.837 | 1394.385 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 204.168 | 204.448 | 205.269 | 205.269 | 206.141 | 37.960 | 38.278 | 39.135 | 39.135 | 41.036 | 242.572 | 243.374 | 244.178 | 244.178 | 246.621 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Radeon 6800 XT, konstante Daten, 27 Bins** | | | | | | | | | | | | | | |
|  |  | **Transferzeit Host zu Device (ms)** | | | | | **Ausführungszeit (ms)** | | | | | **Gesamtzeit (ms)** | | | | |
| **n** | **Kernel** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** | **Min** | **Q10** | **Med** | **Q90** | **Max** |
| 8 | atomic\_global | 0.020 | 0.023 | 0.025 | 0.025 | 0.047 | 0.029 | 0.031 | 0.031 | 0.031 | 0.060 | 0.070 | 0.077 | 0.080 | 0.080 | 0.540 |
|  | atomic\_global\_stride | 0.023 | 0.024 | 0.026 | 0.026 | 0.035 | 0.029 | 0.032 | 0.032 | 0.032 | 0.121 | 0.074 | 0.079 | 0.082 | 0.082 | 0.171 |
|  | atomic\_private | 0.017 | 0.023 | 0.025 | 0.025 | 0.030 | 0.027 | 0.030 | 0.031 | 0.031 | 0.033 | 0.062 | 0.075 | 0.080 | 0.080 | 0.137 |
|  | atomic\_private\_stride | 0.022 | 0.023 | 0.025 | 0.025 | 0.036 | 0.029 | 0.031 | 0.033 | 0.033 | 0.034 | 0.069 | 0.077 | 0.081 | 0.081 | 0.093 |
| 16 | atomic\_global | 0.022 | 0.023 | 0.024 | 0.024 | 0.444 | 0.058 | 0.060 | 0.062 | 0.062 | 0.066 | 0.103 | 0.106 | 0.109 | 0.109 | 0.958 |
|  | atomic\_global\_stride | 0.022 | 0.023 | 0.024 | 0.024 | 0.139 | 0.058 | 0.060 | 0.062 | 0.062 | 0.066 | 0.103 | 0.106 | 0.110 | 0.110 | 0.224 |
|  | atomic\_private | 0.019 | 0.023 | 0.024 | 0.024 | 0.030 | 0.028 | 0.032 | 0.034 | 0.034 | 0.080 | 0.062 | 0.075 | 0.081 | 0.081 | 0.129 |
|  | atomic\_private\_stride | 0.019 | 0.023 | 0.024 | 0.024 | 0.031 | 0.028 | 0.032 | 0.034 | 0.034 | 0.035 | 0.062 | 0.078 | 0.081 | 0.081 | 0.087 |
| 24 | atomic\_global | 2.489 | 2.504 | 2.523 | 2.523 | 2.914 | 6.877 | 6.878 | 7.147 | 7.147 | 7.318 | 9.394 | 9.413 | 9.684 | 9.684 | 10.574 |
|  | atomic\_global\_stride | 2.493 | 2.498 | 2.515 | 2.515 | 2.624 | 6.911 | 6.917 | 7.186 | 7.186 | 7.202 | 9.438 | 9.450 | 9.717 | 9.717 | 9.815 |
|  | atomic\_private | 1.626 | 1.631 | 1.645 | 1.645 | 2.525 | 0.407 | 0.409 | 0.410 | 0.410 | 0.415 | 2.058 | 2.065 | 2.078 | 2.078 | 2.951 |
|  | atomic\_private\_stride | 1.626 | 1.644 | 1.662 | 1.662 | 2.526 | 0.266 | 0.267 | 0.269 | 0.269 | 0.279 | 1.915 | 1.936 | 1.955 | 1.955 | 2.813 |
| 32 | atomic\_global |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_global\_stride | 225.563 | 225.732 | 226.023 | 226.023 | 226.965 | 1787.229 | 1792.958 | 1798.073 | 1798.073 | 1807.456 | 2013.279 | 2019.195 | 2024.317 | 2024.317 | 2033.143 |
|  | atomic\_private |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | atomic\_private\_stride | 225.372 | 225.749 | 226.134 | 226.134 | 226.802 | 79.102 | 80.256 | 81.032 | 81.032 | 82.052 | 304.886 | 306.422 | 307.093 | 307.093 | 308.878 |