Übung 1 Bericht

1. Einleitung

Pipes gehören zu den grundlegendsten Mechanismen zur Kommunikation zwischen Prozessen. In diesem kleinen Projekt wird die Verweildauer einer Nachricht im Kernel während eines Ping-Pong-Protokolls über eine Pipe untersucht, um Rückschlüsse auf die Latenzcharakteristik und das Scheduling-Verhalten von Linux zu gewinnen.

Die Messungen wurden unter Windows Subsystem for Linux 2 (WSL2) durchgeführt, wodurch zusätzliche Virtualisierungs- und Schedulingeffekte auftreten. Gemessen wurde mit einem HP Elitebook 845 G7 Notebook mit den folgenden Spezifikationen: 32GB Arbeitsspeicher, AMD Ryzen 7 PRO 4750U 1.7GHz, AMD Radeon Graphics.

2. Messmethode

Die Versuchsimplementierung erstellt durch fork() einen Kindprozess. Die Kommunikation erfolgt über zwei Pipes (Parent → Kind und zurück). Der Elternprozess misst die Round-Trip-Zeit mittels einer monotonen Uhr (std::chrono::steady_clock), halbiert diese und speichert die Einweg-Verweildauer in einer CSV-Datei. Insgesamt wurden 100.000 Messwerte aufgezeichnet, wobei eine Warmup-Phase vor Beginn der eigentlichen Messung störende Initialeffekte minimiert.

Die Messung erfasst nicht ausschließlich die reine Übertragungszeit, sondern auch Kontextwechsel, Kernel-Scheduling, Hypervisorinteraktionen sowie Caching- und Interrupt-Effekte.

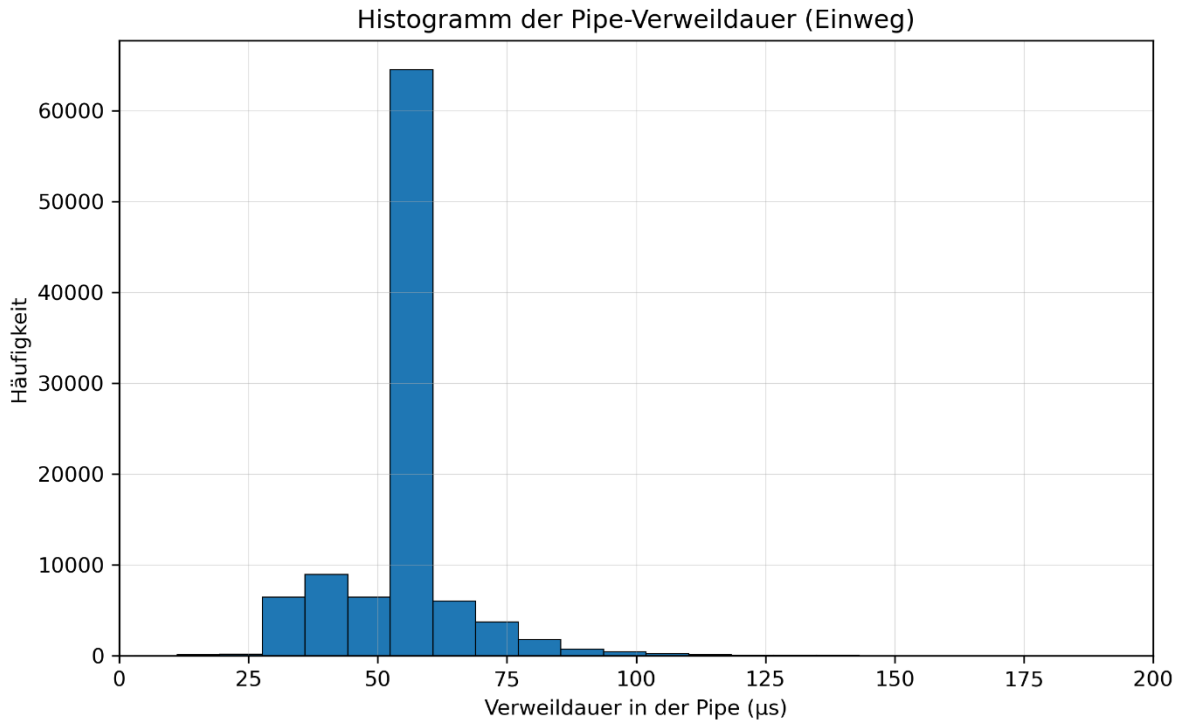
3. Statistische Kennzahlen

Kennwert	Ergebnis
Minimum	**4.249 ns**
Maximum	**522.512 ns**
Mittelwert	**53.492 ns**
Median	**56.454 ns**
Standardabweichung	**14.325 ns**
95. Quantil	**73.913 µs**
99. Quantil	**94.788 µs**
95% Konfidenzintervall	**[55.774, 55.926] µs**

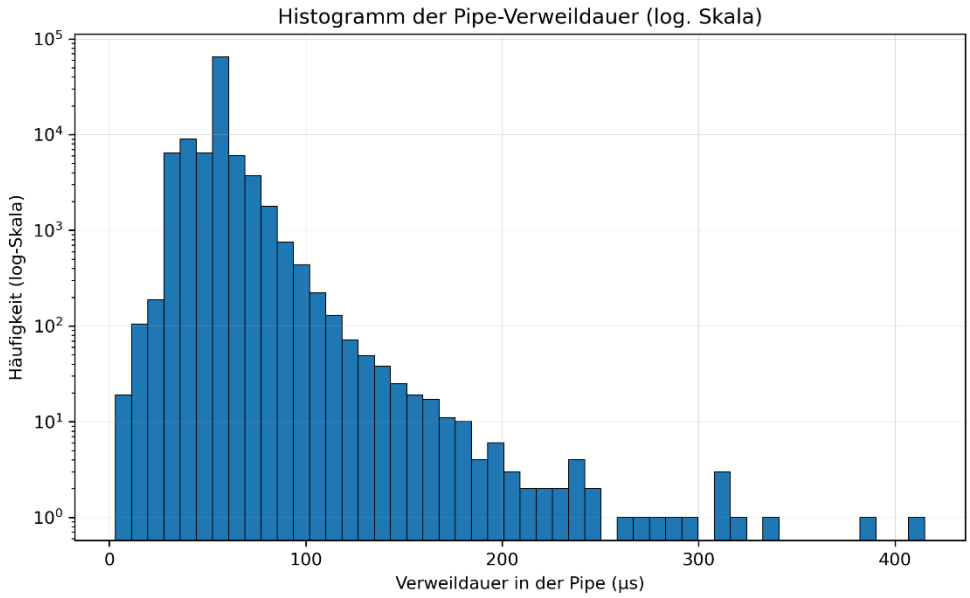
Der Median übersteigt den Mittelwert leicht, was bereits auf eine asymmetrische Verteilung hinweist. Die Standardabweichung ist im Verhältnis zum Mittelwert deutlich ausgeprägt, was auf systematische Schwankungen zurückzuführen ist.

4. Analyse der Messdaten

4.1 Verteilung der Messwerte



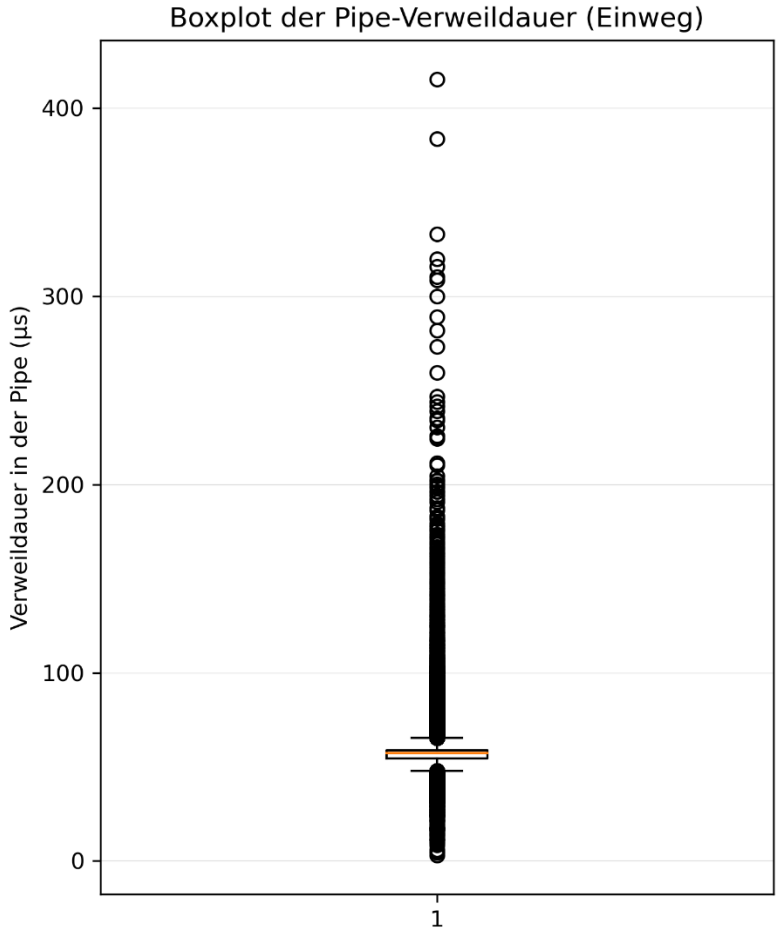
Das Histogramm zeigt eine deutlich rechtsschiefe Verteilung der Messwerte. Ein Großteil der Daten liegt im Bereich von etwa 50–70 µs. Einzelne deutlich erhöhte Messwerte ragen jedoch heraus und markieren Verzögerungsereignisse, die sich nicht zufällig, sondern systembedingt ergeben.



Durch die logarithmische Darstellung wird sichtbar, dass auch Latenzen über 200 µs und vereinzelt über 300 µs auftreten. Diese Messwerte stellen real vorkommende Störsituationen dar, die insbesondere in virtualisierter Umgebung auftreten.

Die Ausprägung der Ausreißer zeigt, dass die Pipe-Kommunikation zwar im Normalfall performant ist, jedoch keinesfalls Echtzeitanforderungen erfüllt.

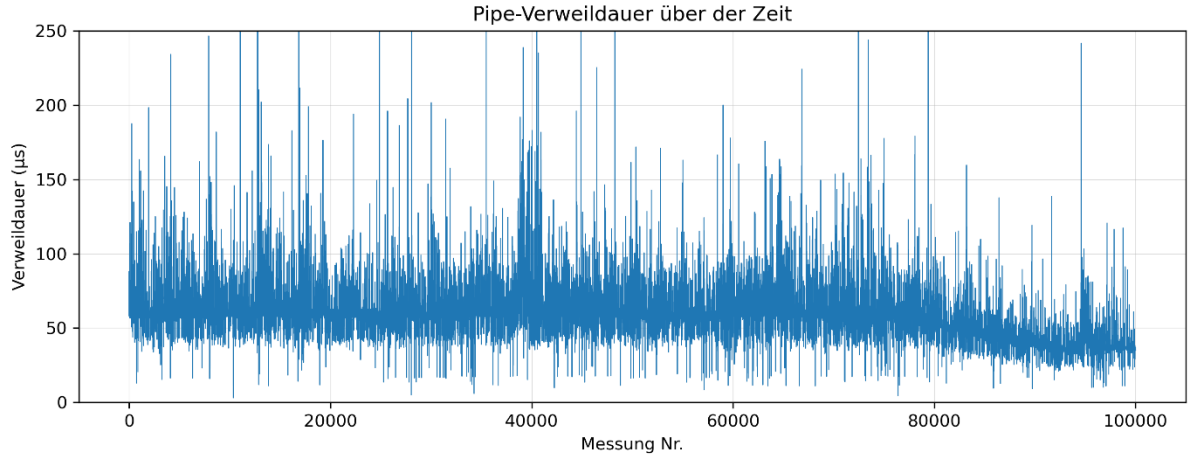
4.2 Ausreißer- und Jitteranalyse



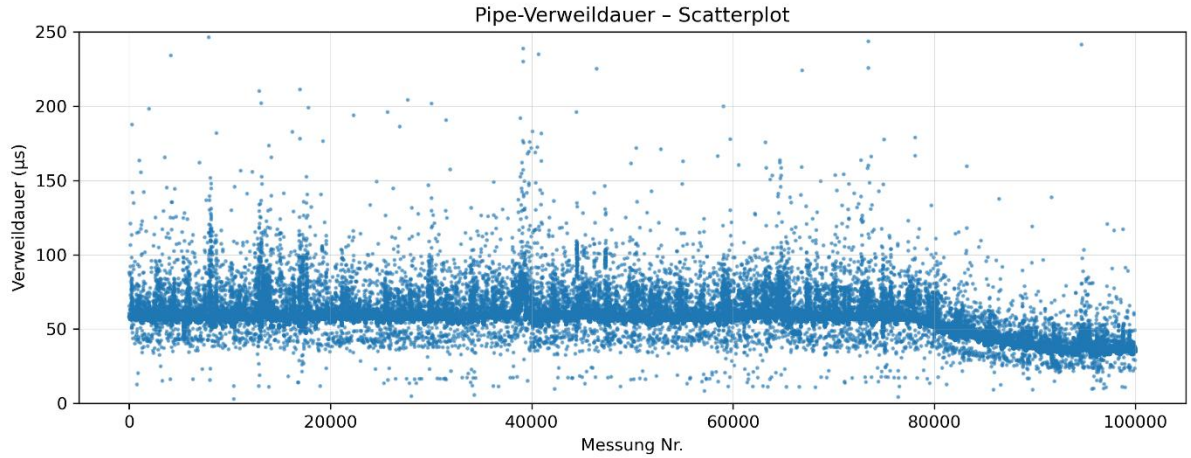
Der Boxplot weist eine vergleichsweise enge Box (Interquartilsabstand) aus, jedoch eine Vielzahl an Ausreißern nach oben. Hieraus lässt sich schließen, dass die Kommunikation im störungsfreien Zustand stabil ist, Interrupts jedoch die Messung signifikant beeinflussen können. Ursachen dafür sind unter anderem:

- Kontextwechsel durch den Windows Scheduler
- Hypervisorbedingte Synchronisation zwischen WSL2 und Windows
- CPU-Frequenzskalierungsmechanismen
- Cache- und Speicherinterferenzen

4.3 Zeitabhängigkeit des Latenzverhaltens

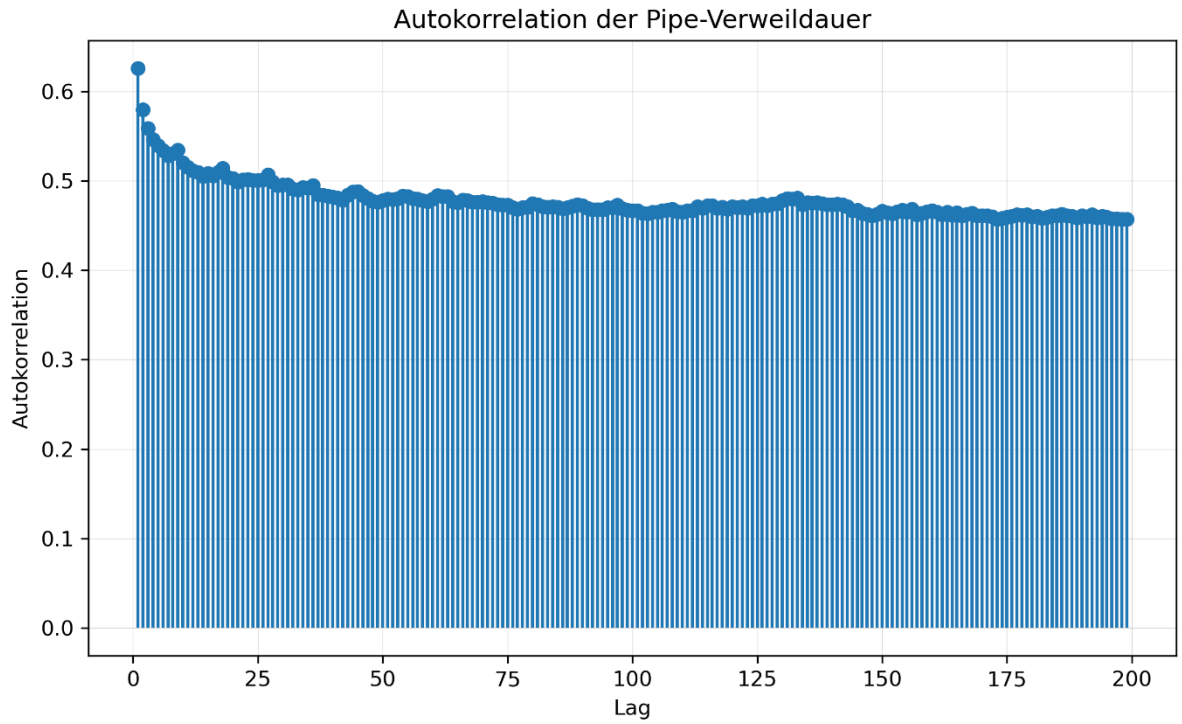


In der Zeitreihe treten periodische Schwankungen in Clustern auf. Diese Clusterbildung lässt auf zeitlich stark korrelierte Systemzustände schließen, z. B. Laständerungen oder thermische CPU-Regulierung.



Die Darstellung im Scatterplot macht die Existenz von „Timing Peaks“ sichtbar. Diese sind einzelne starke Verzögerungen, die unabhängig vom zeitlichen Kontext auftreten. Dies ist typisch für Interrupts oder Reschedulingeffekte.

4.4 Autokorrelation



Die Autokorrelationsfunktion fällt nicht direkt auf Null ab, sondern zeigt eine über viele Messungen hinweg persistente Korrelation. Dies bestätigt, dass bestimmte Systemzustände (z. B. CPU-Frequenz, Hintergrundlast) über längere Zeit wirksam bleiben. Werden viele schnell aufeinanderfolgende Nachrichten gesendet, ist die Latenz daher nicht unabhängig und identisch verteilt.