Laborversuch 3: Interprozesskommunikation

Lernziele:

In diesem Laborversuch sollen Sie lernen, Verfahren der Interprozesskommunikation (IPC) kennenzulernen und gegeneinander abzuschätzen.

Erfolgskontrolle:

Um den Versuch testiert zu bekommen, müssen Sie bei Versuchsbeginn die Vorbereitungsfragen in Moodle beantwortet haben und mindestens 50% der Punkte erreicht haben. Am Ende des Versuchstages müssen Sie ein fertiges Pipe-Programm hochladen und eine Grafik, sowie Fragen zur gemessenen Leistungen und zum Thema IPC beantworten können.

Einführung:

Interprozesskommunikation (IPC) ermöglicht den Datenaustausch zwischen den Prozessen auf einem System. In diesem Labor wollen wir die verschiedenen Möglichkeiten auf dem PC messen. Wie in der Vorlesung besprochen, gibt es verschiedene Charakteristika. Die wichtigsten sind Latenz, Bandbreite und Nachrichten pro Sekunde.

Jede IPC-Methode hat Vor- und Nachteile, bspw. offensichtliche wie niedriger Overhead, hohe Bandbreite, aber auch weniger offensichtliche, wie eine klare Software-Architektur. Sie sollten diese Vor- und Nachteile jeder Methode herausarbeiten.

Zeitmessung von Code-abschnitten ist ungenau, wenn der zu messende Abschnitt sehr kurz ist. Ist die Messung mittels Time-Stamp Counter (TSC) mit der `rdtsc`-Instruktion des Prozessors nicht möglich, muss die Ausführungszeit verlängern werden. Meistens misst man viele (tausend) Ausführungen und bildet den Mittelwert. In den folgenden Test-Programmen liefert das Makro `MEASUREMENTS` die Anzahl, wenn diese nicht bereits von Ihnen beim kompilieren gesetzt wurde:
`gcc -DMEASUREMENTS=1`
setzt diesen Wert auf eine einzige Messung.

Ihre Aufgabe: vermessen und analysieren Sie verschiedene IPC-Methoden

Das erste Programm `rdtsc.c` zeigt, wie kurze Code-Abschnitte gemessen werden können: 50 Mio. Iterationen geben die gemittelte Ausführungszeit. Da der Schleifenkörper selbst auch zwei Instruktionen braucht (ein Dekrement- und eine Sprung-Instruktion) wird die Messung zweimal durchgeführt. Einmal mit einer `rdtsc`- und einmal mit zwei `rdtsc`-Instruktionen; die Differenz ergibt die Zeiten für eine Instruktion.

Laden Sie den Source Code der Programme `bench_rdtsc.c`, `bench_mmap.c`, `bench_process_vm_readv.c` und die Header-Datei `bench_utils.h` aus Moodle herunter. Übersetzen Sie den ersten Benchmark mit dem Aufruf: `gcc -std=gnu99 -Wall -O2 -o bench_rdtsc bench_rdtsc.c`

Dieser Benchmark misst nur die Ausführungszeit einer `rdtsc`-Instruktion.

Die echten Kommunikationsbenchmarks `bench_mmap.c` zur Kommunikation mittels Shared Anonymous Memory Maps und `bench_process_vm_readv.c` zur Kommunikation mit dem gleichnamigen Systemcall kommunizieren weitaus schneller.

- Machen Sie erste Tests mit diesen beiden Benchmarks
- **Programmieren Sie** anhand eines der obigen Benchmarks einen eigenen `bench_pipe.c`, welcher mittels Unix-Pipes kommuniziert.
- Automatisieren Sie die Ausführung, damit die Größen von 64 Bytes, 256, 1024 Bytes usw. bis 16 MB (immer mit Vervierfachung der Schrittweite) gemessen werden.
- Machen Sie Tests mit bis zu 16 MB an Daten und stellen die Ergebnisse bspw. mit Excel, mit Libreoffice oder mit GnuPlot in einem Diagramm dar (X-Achse entspricht der Paketgröße, die Y-Achse der erzielten Bandbreite). Das Diagramm kann eine PDF- oder ein PNG-Datei sein.
- Laden Sie Ihr Programm (C-Datei) und das Diagramm in Moodle hoch.