Probeklausur

Technische Grundlagen der Informatik

Universität:	Technische Universität Berlin
Kurs/Modul:	Technische Grundlagen der Informatik
Bearbeitungszeit:	120 Minuten
Erstellungsdatum:	September 19, 2025

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

(a) Beschreiben Sie die zentralen Bausteine eines Computers und erläutern Sie deren Aufgaben im kurzen Überblick: Rechenwerk, Speicher, Ein- und Ausgabesystem sowie Steuerwerk.

(b) Welche Rolle hat der Cache-Speicher in der Speicherhierarchie? Erklären Sie grob, wie Cache-Hits und Cache-Misses die Leistung beeinflussen und warum eine mehrstufige Cache-Hierarchie üblich ist.

(c) Was versteht man unter einer Befehlssatz-Architektur? Vergleichen Sie grob RISC- und CISC-Ansätze und nennen Sie je zwei Vor- und zwei Nachteile.

(d) Skizzieren Sie eine einfache Speicherhierarchie mit Hauptspeicher, Cache und Sekundärspeicher. Geben Sie typische Zugriffszeiten in groben Größenordnungen an und erläutern Sie den Grund für die Unterschiede.

Aufgabe 2.

(a) Definieren Sie Prozess und Thread; beschreiben Sie deren Lebenszyklus sowie typische Zustände. Geben Sie jeweils zwei Eigenschaften an, die sie unterscheiden.

(b) Welche Grundfunktionen hat ein Betriebssystem im Kontext von Ressourcenmanagement, Scheduling und Synchronisation? Erläutern Sie je zwei Beispiele pro Bereich.

(c) Erläutern Sie grob das Konzept der Speicherverwaltung mittels Paging und beschreiben Sie den Begriff der Seitentabelle. Wie werden virtuelle Adressen in physische Adressen abgebildet?

(d) Welche Mechanismen ermöglichen Isolation und Sicherheit in Mehrbenutzer-/Mehrprozessorsystemen? Nennen Sie zwei Beispiele und erläutern Sie kurz deren Nutzen.

Aufgabe 3.

(a) Beschreiben Sie das Schichtenmodell typischer Netzwerke. Nennen Sie die Kernaufgabe jeder Schicht und geben Sie ein Beispiel für eine Protokollfamilie pro Schicht.

(b) Erklären Sie den Unterschied zwischen verbindungsorientierter und verbindungsloser Kommunikation. Geben Sie jeweils ein Beispiel für eine Service- oder Protokollart.

(c) Beschreiben Sie grob den Aufbau einer TCP-Verbindung (Drei-Wege-Handshake). Nennen Sie zwei zentrale Eigenschaften von TCP und erläutern Sie ihre Relevanz für Zuverlässigkeit.

(d) Welche typischen Probleme verteilter Systeme treten auf? Diskutieren Sie Auswirkungen von Latenz, Ausfällen und Konsistenz auf Anwendungen.

Aufgabe 4.

(a) Definieren Sie ein verteiltes System. Skizzieren Sie zwei Replikationsstrategien und erläutern Sie deren Vor- bzw. Nachteile in Bezug auf Verfügbarkeit und Konsistenz.

(b) Beschreiben Sie das Konzept der Konsistenzmodelle, insbesondere stark konsistente versus eventually-consistent Systeme; Geben Sie je ein kurzes Beispiel, das die Unterschiede veranschaulicht.

(c) Nennen Sie typische Sicherheitsaspekte in verteilten Systemen, wie Authentisierung, Integrität und Vertraulichkeit, und erläutern Sie kurz deren Bedeutung.

(d) Beschreiben Sie eine einfache Kommunikationsarchitektur zwischen zwei verteilten Instanzen mit asynchronem Nachrichtenaustausch. Diskutieren Sie Vor- und Nachteile dieser Strategie.

Lösungen

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

(a) Die zentralen Bausteine eines Computers lassen sich wie folgt zusammenfassen: Das Rechenwerk (ALU) führt arithmetische und logische Operationen aus; der Speicher umfasst Register, Cache und Hauptspeicher und dient der zeitlich schnellen bzw. mittelfristigen Speicherung von Daten; das Ein- und Ausgabesystem (I/O) kümmert sich um die Kommunikation mit Peripherie und externen Geräten; das Steuerwerk (Control Unit) koordiniert Fetch-Decode-Execute-Zyklen, interpretiert Befehle und steuert die Datenpfade. Zusammen bilden sie die Grundlagen des Betriebsmittelversuchs, das Zusammenspiel der Bausteine ermöglicht das Funktionieren einer Zentraleinheit.

(b) In der Speicherhierarchie nimmt der Cache-Speicher eine zentrale Rolle ein: Er dient dazu, die Latenzzeit von Speicherzugriffen zu reduzieren, indem häufig bzw. wiederholt benötigte Daten näher an der CPU gehalten werden. Cache-Hits bedeuten einen schnellen Zugriff (Daten bereits im Cache vorhanden), während Cache-Misses ein Nachladen aus dem nächsten Speicherebenen erfordern. Eine mehrstufige Cache-Hierarchie (L1, L2, ggf. L3) verringert die durchschnittliche Zugriffslatenz, weil schnelle Ebenen den Großteil der Zugriffe abfangen. Typische Folgen: Je höher der Cache-Hit-Rate, desto schneller der Gesamtsystemzugriff; dabei steigt aber die Kosten- und Komplexität der Hardware.

(c) Unter einer Befehlssatz-Architektur (ISA) versteht man die Menge der maschinennahe Befehle, die eine CPU direkt ausführen kann. Grob unterscheiden sich RISC- und CISC-Ansätze:

- **RISC (Reduced Instruction Set Computing):**
  - Vorteile: einfache, feste Befehlssatzlänge erleichtern Pipeline-Umsetzungen; oft bessere Compiler-Unterstützung; tendenziell geringerer Hardwarekomplexität.
  - Nachteile: größere Code-Menge (höhere Speicheranforderung) aufgrund von Load/Store-Architektur.
- **CISC (Complex Instruction Set Computing):**
  - Vorteile: größere Code-Dichte, komplexe Befehle können mehrere Operationen in einem Schritt bündeln.
  - Nachteile: komplexer Hardwareentwurf, teurere Pipelines, potentiell schlechtere Optimierung durch Compiler.

Zwei Vor- bzw. zwei Nachteile je Ansatz werden hiermit aufgeführt.

(d) Eine einfache Speicherhierarchie lässt sich folgendermaßen skizzieren:

CPU Register → L1-Cache → L2-/L3-Cache → Hauptspeicher (RAM) → Sekundärspeicher (SSD/HDD)

Typische Zugriffszeiten (grobe Größenordnungen):

- Register: < 1 ns
- L1-Cache: 0.5–2 ns

- L2-Cache: 2–10 ns
- Hauptspeicher (RAM): 50–100 ns
- Sekundärspeicher (SSD): 50–300 s
- Sekundärspeicher (HDD): 5–20 ms

Begründung: Die Unterschiede entstehen aus der weltweiten Diskrepanz zwischen sehr geringer Latenzzeit von Cache- bzw. Registerhalten und der vergleichsweise hohen Latenz von RAM bzw. Sekundärspeichern aufgrund physikalischer Zugriffszeiten und Bandbreite. Eine mehrstufige Hierarchie reduziert die durchschnittliche Zugriffzeit durch Ausnutzung von Lokalisierungsprinzipien (Temporal- und Spatial Locality).

Aufgabe 2.

(a) *Prozess* und *Thread* lassen sich wie folgt definieren:

- **Prozess:** Ausführungseinheit mit eigenem Adressraum, eigener Abbildung und unabhängiger Ausführungskontext. Lebenszyklus: Neu -> ready -> running -> waiting/blocked -> terminated. Typische Zustände: neu, ready, running, waiting, terminated.
- **Thread:** Leichtgewichts-Ausführungseinheit innerhalb eines Prozesses; Teilt sich Adressraum und Resourcen des Prozesses, besitzt eigenen Ausführungskontext (Registersatz, Program Counter). Lebenszyklus analog zu Prozessen, jedoch geringerer Abhängigkeitsaufwand beim Kontextwechsel.

Unterschiede (je zwei Eigenschaften):

- Prozess vs. Thread 1: Adressraum
  - Prozess hat eigenen Adressraum (isoliert).
  - Thread teilt Adressraum des übergeordneten Prozesses.
- Prozess vs. Thread 2: Kontextwechselkosten
  - Prozesswechsel erfordert meist teureren Kontextwechsel (Seiten- und TLB-Flush etc.).
  - Threadwechsel ist in der Regel leichter, da Adressraum geteilt wird.

(b) Grundfunktionen des Betriebssystems im Kontext von Ressourcenmanagement, Scheduling und Synchronisation:

- *Ressourcenmanagement:* Speicherverwaltung (Paging/Segmentation), Geräte- bzw. I/O-Management (DMA, Treiber), Dateisystemressourcen.
- *Scheduling:* Zuteilung der CPU-Zeit (CPU-Scheduler), Planung der I/O-Operationen (I/O-Scheduler), Priorisierung von Prozessen/Threads.
- *Synchronisation:* Mechanismen zur Vermeidung von Race Conditions (Mutexe, Semaphoren, Monitore), Synchronisationsprimitive zur Koordination von Threads.

Je zwei Beispiele:

- Ressourcenmanagement: Paging, Swap-Bereich; DMA-basierte E/A.
- Scheduling: Preemptives Scheduling (z. B. Round-Robin, Priority Scheduling); I/O-Scheduling-Strategien (CFQ, Deadline).
- Synchronisation: Mutex, Semaphor, Condition Variables.

(c) *Paging* als Grundprinzip der Speicherverwaltung: Jede virtuelle Adresse wird in eine *Seitennummer* (Page Number) und einen *Offset* zerlegt. Die Seitennummer dient als Index in der *Seitentabelle*, die die Zuordnung von virtueller Seite zu physischer Rahmennummer festhält. Häufige Beschleunigung durch *TLB* (Translation Lookaside Buffer). Bei einem *Page Fault* wird die fehlende Seite aus dem Sekundärspeicher geladen und die Seitentabelle bzw. der TLB entsprechend aktualisiert. Virtuelle Adressen werden somit in physische Adressen abgebildet, teilweise unter Berücksichtigung weiterer Abstufungen (z. B. Seitengrenzen, Rechte).

Kurzform: virtuelle Adresse (VPN, $offset$) → Seitentabelle → physis. Frame → offset.

(d) *Isolation und Sicherheit* in Mehrbenutzer-/Mehrprozessorsystemen werden z. B. durch:

- **Hardware-Unterstützung:** MMU/Privilegienstufen (Kernel/User-Modus), Speicherschutz durch Seitenabbildung.
- **Betriebssystem-Mechanismen:** Adressraumtrennung der Prozesse, Zugriffskontrolllisten (ACLs), Benutzer-/Rollenkonzepte.

Je zwei Beispiele und Nutzen:

- MMU + Seitenabbildung: Schutz vor unberechtigtem Zugriff anderer Prozesse; robuste Speicherisolierung.
- Privilegienmodell (Kernelmodus vs. Benutzermodus): Betriebssystemkern kann hardwarenahe Funktionen sicher steuern; Anwendungen bleiben isoliert.

Aufgabe 3.

(a) *Schichtenmodell typischer Netzwerke* (OSi-Schichten, Kernaufgaben pro Schicht; Protokollfamilie pro Schicht):

- **Schicht 1 – Physical:** Übertragung von Bits über physikalische Medium; Kernaufgabe: Signalübertragung. Protokollfamilie: IEEE 802.x (z. B. 802.3 Ethernet).
- **Schicht 2 – Data Link:** Fehlererkennung, Rahmung, Flusskontrolle; Kernaufgabe: fehlerfreier Rahmenversand innerhalb eines Netzwerks. Protokollfamilie: IEEE 802.11 / Ethernet (PPP als Alternative).
- **Schicht 3 – Network:** Routing/Weiterleitung von Paketen; Kernaufgabe: Pfadwahl. Protokollfamilie: IP (IPv4/IPv6).
- **Schicht 4 – Transport:** Aufbau, Steuerung und Beendigung von End-to-End-Verbindungen; Kernaufgabe: Zuverlässigkeit und Flusskontrolle über den Transport. Protokollfamilie: TCP/UDP (TCP/IP-Familie).
- **Schicht 5 – Session:** Verwaltung von Dialogen zwischen Kommunikationspartnern; Kernaufgabe: Sitzungssteuerung (Aufbau/Abbau von Sitzungen). Protokollfamilie: NetBIOS / RPC (Beispiele).
- **Schicht 6 – Presentation:** Syntax/Formatierung, Verschlüsselung, Kompression; Kernaufgabe: Darstellung von Informationen. Protokollfamilie: TLS/SSL (Verschlüsselungs- und Darstellungsdienste).
- **Schicht 7 – Application:** Bereitstellung von Netzwerkdiensten für Endanwendungen; Kernaufgabe: Applikationsprotokolle. Protokollfamilie: HTTP/HTTPS, SMTP, FTP (Anwendungsprotokolle der Internet-Familie).

(b) *Unterschied verbindungsorientiert vs. verbindungslos*; Beispiele:

- **Verbindungsorientiert:** TCP – Aufbau einer zuverlässigen Verbindung mit Sequenznummern und Bestätigungen; geeignete Nutzung z. B. beim Dateiübertragen.
- **Verbindungslos:** UDP – sendet Datagramme ohne Verbindungsaufbau; geeignet z. B. für Streaming oder DNS-Anfragen.

(c) *Aufbau einer TCP-Verbindung (Drei-Wege-Handshake)*:

- *Schritt 1:* Client sendet SYN mit Initial Sequence Number (ISN) an den Server.
- *Schritt 2:* Server antwortet mit SYN-ACK, wobei er seine eigene ISN sendet und die empfangene Sequenznummer bestätigt.
- *Schritt 3:* Client beweist den Empfang mit ACK, die Verbindung wird etabli.

Zwei zentrale Eigenschaften von TCP und deren Relevanz für Zuverlässigkeit:

- **Zuverlässigkeit durch Bestätigungen und Retransmission:** Sequenznummern, ACKs und Retransmissionen sorgen dafür, dass verlorene Segmente erneut übertragen werden und in der richtigen Reihenfolge beim Empfänger ankommen.

- **Fluss- und Staukontrolle:** Das Sliding-Window-Verfahren (Flusskontrolle) sowie Staukontrollmechanismen (z. B. TCP-Cubic/Reno) regulieren Sendeverhalten, um Überlastung zu vermeiden und eine stabile Übertragung zu ermöglichen.

(d) *Typische Probleme verteilter Systeme*:

- **Latenz und Bandbreite:** Verzögerungen durch Netzwerkentfernung, Vermittlung, Queuing; beeinflusst Reaktionszeiten, insbesondere bei Remote-Diensten.
- **Ausfälle und Teilung (Partitionsschutz):** Knoten ausfallen oder Kommunikationspfade unterbrochen; erfordert Replikation, Failover-Strategien und robuste Fehlertoleranz.
- **Konsistenz vs. Verfügbarkeit (CAP):** In verteilten Systemen kann man nicht gleichzeitig Konsistenz, Verfügbarkeit und Partitionstoleranz in allen Fällen maximal gewährleisten; Designentscheidungen beeinflussen Anwendungslogik.

Aufgabe 4.

(a) *Zwei Replikationsstrategien*:

- **Master-Slave (Primary-Secondary):** Ein Masterknoten nimmt Schreibzugriffe entgegen und repliziert asynchron oder synchron auf Slaves. Vorteile: einfache Konsistenzkontrolle, klare Rolleverteilung; Nachteile: Abhängigkeit vom Master; Ausfall des Masters reduziert Verfügbarkeit bzw. erfordert Failover.
- **Quorum-basierte Replikation (Mehrheitsquoren):** Jeder Schreib-/Lesezugriff erfordert Kontakt zu einer Mehrheit von Knoten; Lese-/Schreiboperationen verwenden definierte Quoren. Vorteile: höhere Verfügbarkeit und Fehlertoleranz; Nachteile: potenziell höhere Latenz, komplexere Konsistenzmodelle.

(b) *Konsistenzmodelle*:

- **Stark konsistente Systeme:** Alle Lesungen sehen den zuletzt abgeschlossenen Schreibvorgang; Beispiel: Finanztransaktionen in einer Bank, bei denen der Kontostand nach jedem Abschlusssinus konsistent ist.
- **Eventually-consistente Systeme:** Schreibzugriffe werden asynchron repliziert; es kann vorübergehend zu Inkonsistenzen kommen, aber über Zeit konvergiert der Zustand. Beispiel: Facebook-Newsfeed, Social-Graph-Updates.

(c) *Typische Sicherheitsaspekte* in verteilten Systemen:

- **Autorisierung/Authentisierung:** Wer darf welche Operation ausführen; Identity- und Zugriffsmanagement.
- **Integrität:** Gewährleistung, dass Daten während Übertragung bzw. Speicherung unverändert bleiben (Checksummen, Signaturen).
- **Vertraulichkeit:** Schutz vor Abhören und unbefugtem Zugriff (Verschlüsselung, TLS).

(d) *Einfache asynchrone Kommunikationsarchitektur zwischen zwei verteilten Instanzen*:

- Architektur: Zwei Instanzen A und B kommunizieren asynchron über eine Messaging-Schicht (z. B. Message-Broker oder Queue). A sendet Nachrichten an eine Queue; B abonniert die Queue oder pollt aktiv.
- *Vorteile:* Entkopplung von Sender und Empfänger, bessere Fehlertoleranz, Skalierbarkeit, natürliche Dekomposition von Aufgaben.
- *Nachteile/Probleme:* Reihenfolgegarantie ist nicht zwingend; Nachrichten könnenDupliziert werden (At-least-once); Erfordernis von Deduplication-Logik; eventuelle Latenzen durch Queue-Verarbeitung.