Hochschule **RheinMain**

VERTEILTE INFRASTRUKTUR

Verteilte Systeme

Prof. Dr. Georg Hinkel

05.07.2024

GLIEDERUNG

Datum	Vorlesung	Übungsblatt	Abgabe
19.04.2024	Einführung	HamsterLib	06.05.2024
26.04.2024	Netzwerkprogrammierung	Theorie	
03.05.2024	World Wide Web	HamsterRPC 1	20.05.2024
10.05.2024	Remote Procedure Calls	Theorie	
17.05.2024	Webservices	HamsterRPC 2	03.06.2024
24.05.2024	Fehlertolerante Systeme	Theorie	
31.05.2024	Transportsicherheit	HamsterREST	17.06.2024
07.06.2024	Architekturen für Verteilte Systeme	Theorie	
14.06.2024	Internet der Dinge	HamsterIoT	01.07.2024
21.06.2024	Namen- und Verzeichnisdienste	Theorie	
28.06.2024	Authentifikation im Web	HamsterAuth	15.07.2024
05.07.2024	Infrastruktur für Verteilte Systeme	Theorie	
12.07.2024	Wrap-Up	HamsterCluster (Bonus)	16.08.2024

AGENDA UND LERNZIELE

Hochschule RheinMain

Agenda

- Verteilter Speicher
  - Grundlagen verteilter Dateisysteme
  - Verteilte Dateisysteme: NFS, WebDAV
  - Speichernetze: RAID, NAS, SAN
- Verteilte Verarbeitung
  - Docker, Kubernetes
  - Terraform

Lernziele

- Funktionsweise von NFS und WebDAV erklären können
- RAID-Level erklären können, geeignete Level auswählen können
- Geeignete Speichernetzarchitektur auswählen können
- Kubernetes und Terraform anwenden können, um Cluster zu spezifizieren

EINFÜHRUNG

Hochschule RheinMain

- Bisher: Verteilte Systeme, um Kommunikation zwischen Teilen eines Systems zu realisieren
  - Annahme: einzelne Teile des Systems können die Ihnen zugewiesene Aufgabe bewerkstelligen
- Realität: Leistungsanforderungen bei größeren Workloads von einzelner Maschine häufig nicht leistbar
  - Begrenzte Speicherkapazität
  - Begrenzte Rechenkapazität
  - Begrenzte Netzwerkkapazität
- Lösung durch verteilte Infrastruktur
  - Verteilter Speicher, verteilte Dateisysteme
  - Verteilte Rechen-/Netzwerkkapazitäten
- Ziel: Verteilung der Infrastruktur transparent für verteilte Anwendung

Hochschule RheinMain

WIE VERTEILEN WIR SPEICHER?

Verteilte Dateisysteme, Speicherarchitekturen

EINFÜHRUNG VERTEILTER SPEICHER

Hochschule RheinMain

- Dateien als grundsätzliche Abstraktion über Speichersysteme
  - Datei = benannter Speicherbereich, hierarchisch organisiert
  - Klassischerweise vom Betriebssystem verwaltet, inkl. Nebenläufigkeit und Rechteverwaltung
  - Einheitliche Programmierschnittstelle für (lokale) Anwendungen
  - Sehr universell eingesetzt

- Idee für verteiltes System: Verteilung durch verteiltes Dateisystem
  - Unabhängig von der Anwendung ➔ auch für Legacy-Anwendungen geeignet
  - Transparente Replikation ➔ Ausfallsicherheit

- Architekturmodell
  - Client/Server ➔ dedizierter File-Server
  - Peer-to-Peer ➔ jeder Teilnehmer kann Dateien bereitstellen

EINFÜHRUNG
Ansätze

Hochschule RheinMain

Isolierte Dateisysteme
- Dateizugriffe ausschließlich lokal
- Upload und Download von Dateien zwischen isolierten Dateisystemen

- Keinerlei Transparenz

- Beispiel: RCP, SCP

Adjungierte Dateisysteme
- Zugriff auf entfernte Dateien
- Explizite Ortsangabe als Bestandteil des Dateipfads

- Zugriffstransparenz

- Beispiel: Newcastle Connection, WebDAV

Verteilte Dateisysteme
- Einheitliches Dateisystem für alle Maschinen im Netzwerk

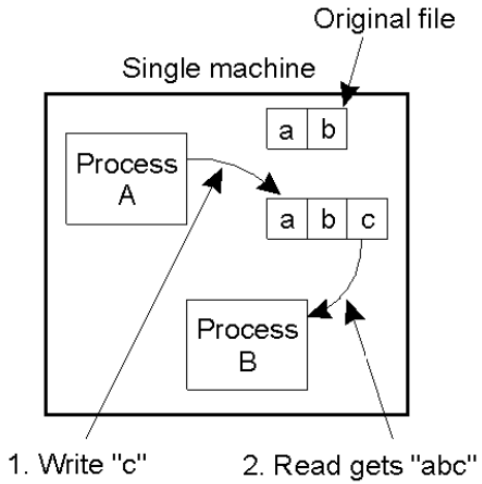
- Zugriffstransparenz, Ortstransparenz

- Beispiel: NFS, SMB, …

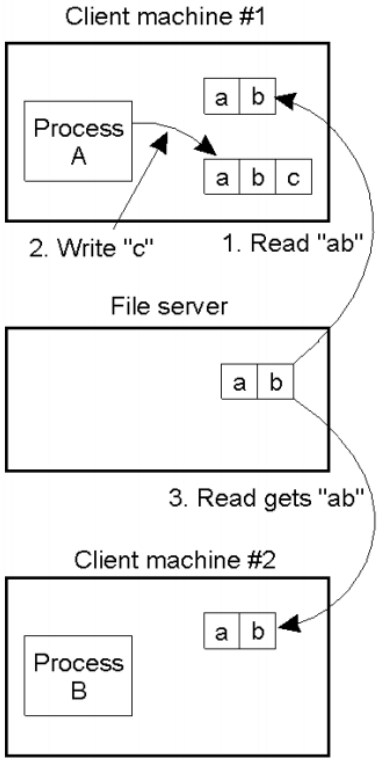
EINFÜHRUNG
Zugriffskonsistenzproblem

Hochschule RheinMain

Strenge Konsistenz
- Änderungen für jeden unmittelbar sichtbar



Eventual Consistency
- Dateien können veraltet sein, Aktualisierung beim Schließen



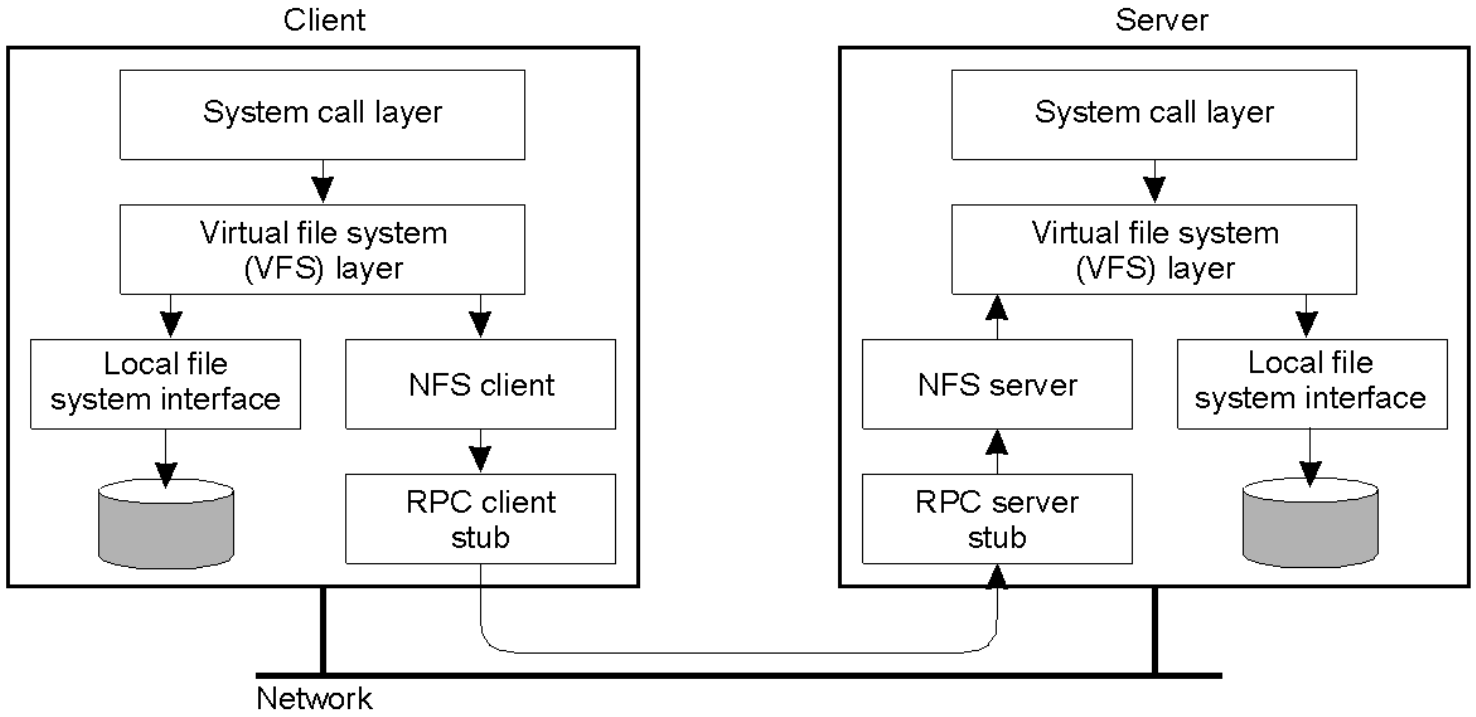
[Abb. aus Tanenbaum/Steen]

NETWORK FILE SYSTEM (NFS)
Einführung

Hochschule RheinMain

- Ursprünglich 1984 von Sun entwickelt, Nutzung anfangs für diskless Workstations
  - Basierend auf Sun-RPC

- Industriestandard durch Offenlegung von Schnittstelle und Implementierung



[Abb. aus Tanenbaum/Steen]

NETWORK FILE SYSTEM (NFS)
Funktionsweise

Hochschule RheinMain

- Mounting
  - Client erfragt Datei-Handle vom Server für Verzeichnis, das gemountet werden soll
  - Handle typischerweise ausreichend um Datei schnell zu finden (unter Unix: Disk ID + Inode-Nr)
  - Client speichert entfernten Handle im eigenen Dateisystem ab (via Vnode-Schnittstelle)
  - Client führt beim Zugriff auf die Datei READ-Prozedur aus, übergibt Datei-Handle

- Jede Maschine kann prinzipiell sowohl Client als auch Server sein

NETWORK FILE SYSTEM (NFS)

Unterschiede bis v3 und v4

Hochschule RheinMain

Frühere NFS-Versionen

- Eigener Mounting-Dienst (mountd)
- Eigener Locking-Dienst
- Separater Portmapper-Dienst, verwaltet auf welchen Ports welche Dienste laufen
- Standardmäßig UDP, TCP möglich
- Authentifikation durch Unix-Benutzer und Gruppennummern, standardmäßig im Klartext übertragen
  - Alternativen Diffie-Hellman oder Kerberos, aber geringe Verbreitung
- Zustandslos

NFSv4

- Komplette Neuimplementierung
- Mounting und Locking als Bestandteile des Protokolls
- Jegliche Kommunikation über TCP Port 2049
- Bündelung von mehreren Anfragen in einer Nachricht ➔ Nutzbarkeit in WAN
- Authentifikation durch user@domain
- Verschlüsselung Teil des Protokolls
- Zustandsbehaftet

NETWORK FILE SYSTEM (NFS)
Alternativen

Hochschule RheinMain

- Server Message Block (SMB), früher auch Common Internet File System (CIFS)
  - Ursprünglich 1983 durch Barry Feigenbaum bei IBM entwickelt
  - Proprietärer aber offengelegter Standard von Microsoft (Versionen 2.0 und 3.0)
  - Unterstützt neben verteiltem Dateisystem auch andere Dienste (e.g. Druckerdienste)
  - Unter Windows stark verbreitet, in Unix mittels Samba verfügbar
  - Version 1.0 typisches Angriffsziel von Malware
- Andrew File System (AFS) und Coda
  - Ursprünglich universitäres Projekt an der CMU, dann von IBM gekauft, dann Open Source (OpenAFS)
  - Ortstransparenter Dateinamensraum, hohe Skalierbarkeit, Abstraktion von Replikationen
- Spezielle Dateisysteme zum Einsatz in Clustern
  - Beispiele: Lustre, GlusterFS, BeeGFS

Hochschule RheinMain

GAB ES IN HTTP NICHT AUCH PUT UND DELETE, UM RESSOURCEN ANZULEGEN UND ZU LÖSCHEN?

WebDAV

WEB-BASED DISTRIBUTED AUTHORING AND VERSIONING (WEBDAV)

Hochschule RheinMain

- Idee: HTTP war eigentlich gedacht, um Ressourcen auch editieren zu können
  - Manko: Server implementieren die Funktionalität nicht, PUT und DELETE üblicherweise mit 405 „Method not allowed“ beantwortet

- WebDAV (RFC 4918): Dateiaustausch über HTTP
  - Port 80, bzw. 443 (HTTPS)
  - Zusätzliche Anfragemethoden
    - PROPFIND: Eigenschaften einer Ressource abfragen
    - PROPPATCH: Ändert Eigenschaften einer Ressource
    - MKCOL: Verzeichnis anlegen
    - COPY: Ressource kopieren
    - MOVE: Ressource verschieben
    - LOCK: Ressource sperren
    - UNLOCK: Ressource entsperren

- WebDAV von den meisten Betriebssystemen unterstützt
  ➔ Server lässt sich dann in Dateimanager einbinden

Hochschule RheinMain

ABER WIE WIRD SPEICHER IM NETZWERK VERWALTET?

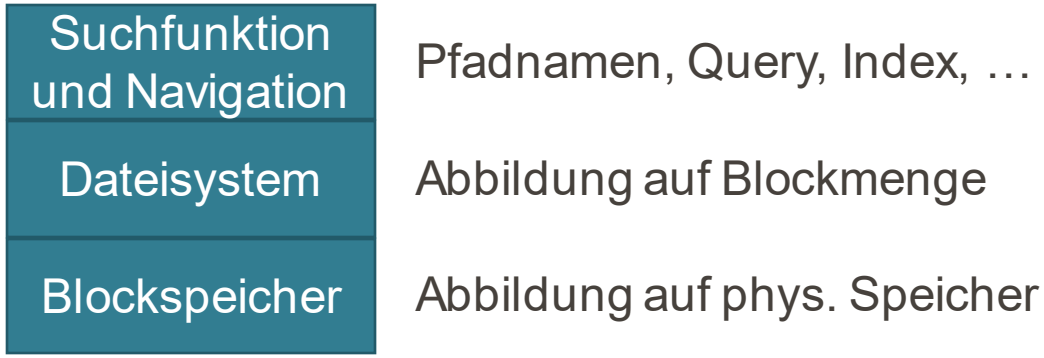
Speichernetze

SPEICHERNETZE
Motivation

Hochschule RheinMain

- Kostenreduktion
  - Geringere bereitgestellte Speicherkapazitäten
  - Zentrale Administration
- Flexibilität der Zuordnung
  - Schnelle Anpassbarkeit an neue Bedürfnisse
- Skalierbarkeit
  - Kleine bis sehr große Speicherkapazitäten
- Möglichkeit für Desaster Recovery
  - Datenspiegelung an entfernte Stellen

Aufteilung der Funktionalität von Speichern



REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID)

Hochschule RheinMain

- RAID erlaubt verschiedene Festplatten zusammenzufassen
  - Fehlertoleranz
  - Performance
  - Speicherkonsolidierung
- Zusammenfassung gekapselt in RAID-Controller ➔ Platten erscheinen als logische Einheit
  - Automatische Behandlung eines Plattenausfalls (hot spare)
- Verschiedene Modi (siehe rechts)
  - Kombinationen von Verbünden (nested RAID)
  - Bsp.: RAID-50 bzw. RAID 5+0

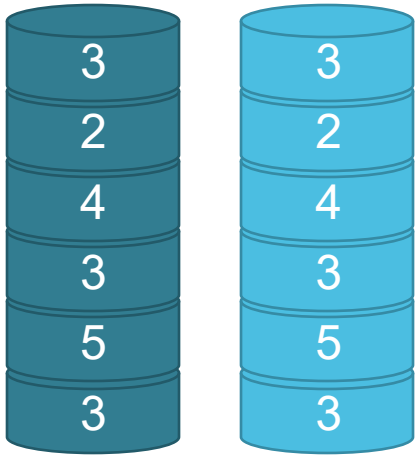
Level	Beschreibung	Min # Platten	Fehler-toleranz
RAID-0	Blockweise Striping	2	Keine
RAID-1	Spiegelung	2	Ausfall $n-1$ Platten
RAID-2/3/4	Striping mit dedizierter Parität	3	Ausfall 1 Platte
RAID-5	Blockweise Striping, Verteilte Parität	3	Ausfall 1 Platte
RAID-6	Blockweise Striping, Verteilte Parität (redundant)	4	Ausfall 2 Platten

REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID)

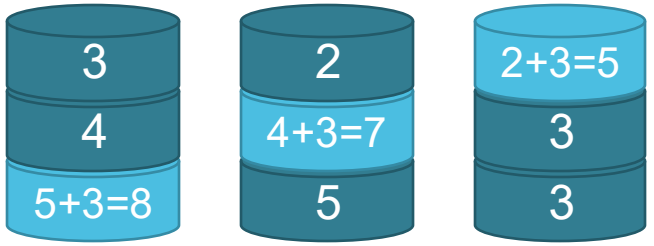
Hochschule RheinMain

RAID-0
- Daten werden in Blöcken auf verschiedenen Platten gespeichert
- Keine Redundanz

RAID-1
- Daten werden komplett gespiegelt
- Bei Ausfall wird Replikat verwendet



RAID-5
- Daten werden in Blöcken auf verschiedene Platten gespeichert
- Zusätzlich wird Parität gespeichert (verteilt)

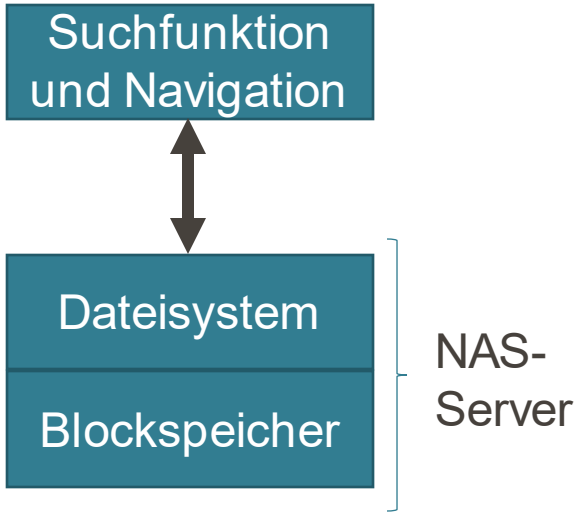


SPEICHERNETZE
Überblick

Hochschule RheinMain

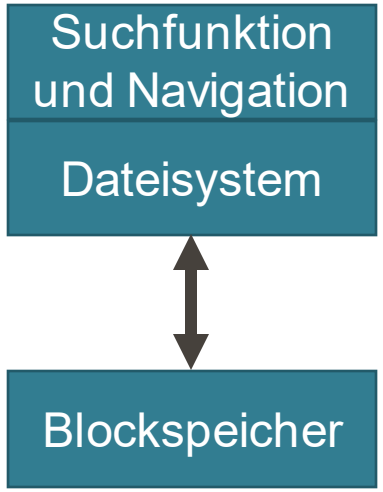
Direct Access Storage (DAS)

- Speicher direkt an Server angeschlossen
  - Keine Verteilung
- Server hält Dateisystem und Gerätetreiber



Network Attached Storage (NAS)

- Dedizierter Server für Dateidienste
- NAS-Server bietet verteiltes Dateisystem
  - NFS, SMB



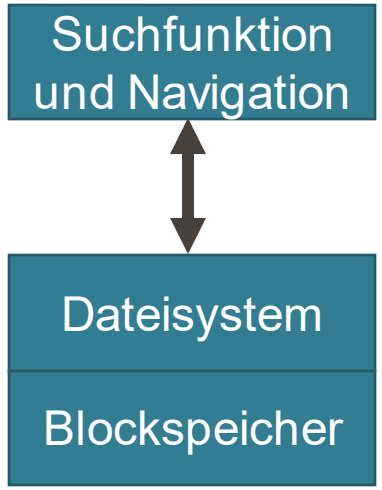
Storage Area Network (SAN)

- Storage erscheint als logische Einheit
- Server-Betriebssystem stellt Dateisystem zur Verfügung
- Replikation, Desaster Recovery

SPEICHERNETZE
Network Attached Storage (NAS)

Hochschule RheinMain

- Idee: Separater Server für Dateizugriff
  - Besonders geeignet für Anwendungen, die auf Dateizugriff basieren
  - E.g., Home-Verzeichnisse
- Ermöglicht transparente Replikation, Skalierbarkeit, Erweiterbarkeit, Administrierbarkeit
- Kommerzielle NAS-Server
  - Typischerweise neben Dateisystem weitere Aufgaben wie Replikation, Desaster Recovery
  - Beispiele: Western Digital, Synology, QNAP, TrueNAS (tw. Open-Source)

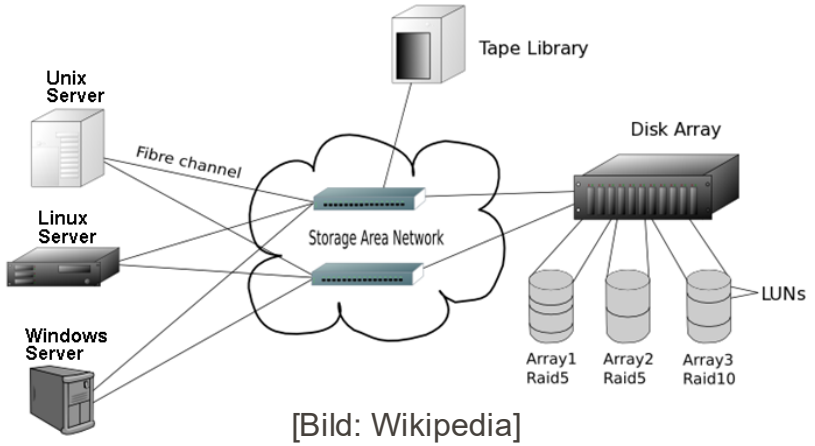


- Problem: Dateizugriff typischerweise über TCP/IP, belastet LAN

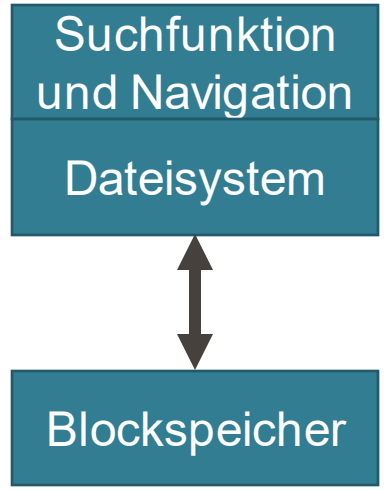
SPEICHERNETZE
Storage Area Network (SAN)

Hochschule RheinMain

- Idee: Separates Netzwerk für Speicherzugriff
  - „Network behind the servers“
  - Typischerweise blockweiser Zugriff auf Speicher
- Bitübertragung über Glasfaser (Fibre Channel) oder Kupferkabel (InfiniBand)
  - Netzwerk (LAN) bleibt unbelastet
  - Ausnahme: iSCSI, verwendet TCP/IP (Kosten)
- SANs sind bootfähig
- Kaum standardisiert ➔ teuer



[Bild: Wikipedia]



SPEICHERNETZE
Nutzung im Cloud Computing

Hochschule RheinMain

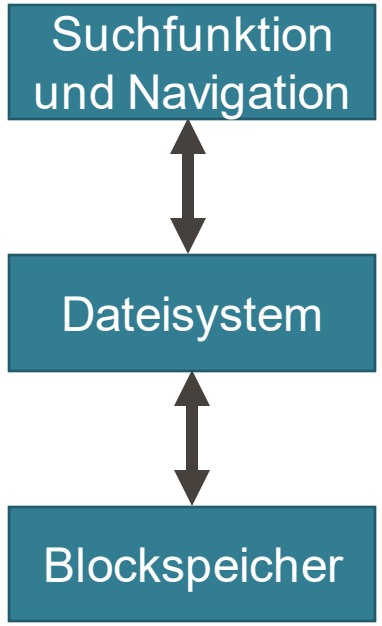
- Data Center (insb. von Cloud-Anbietern) arbeiten i.d.R. viel mit SANs
  - ➔ Sie können CPU/RAM und Speicher meist unabhängig voneinander auswählen
  - ➔ Sie können CPU/RAM und Speicher meist unabhängig voneinander skalieren
  - ➔ Sie müssen CPU/RAM und Speicher meist unabhängig voneinander bezahlen

- NAS in der Cloud eher weniger relevant
  - Keine integrierte Versionskontrolle (vgl. Google Docs, Sharepoint)
  - Meist teurer als reine Speichertechnologie (Amazon S3, Azure Blob Storage)
  - Nutzung vorrangig zur Integration von Altanwendungen

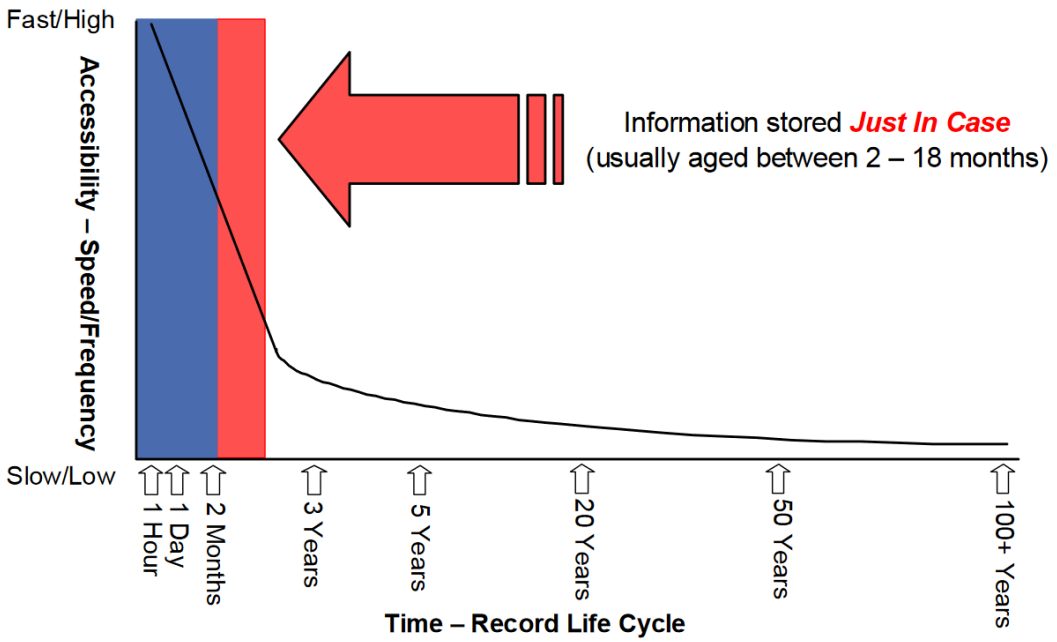
SPEICHERNETZE
Kombination aus SAN und NAS

Hochschule RheinMain

- In großen verteilten Dateisystemen wird die eigentliche Speicherung wieder an eigenes Netzwerk ausgelagert
  - Verbindet Skalierbarkeit/Fehlertoleranz von SANs mit Zugriffsmuster von NAS



DATEN LEBENSZYKLUS



[Jon Tate et al., „Introduction to Storage Area Networks“, IBM RedBooks, 2017]

- Hardware kann Daten zeitlich unbegrenzt lagern
  - Festplatten gehen eher wegen Schreib-/Leseoperationen kaputt
- Interesse an Daten typischerweise schnell abflachend
  - Meiste Zugriffe auf Daten der letzten 2 Monate
  - Aber: Speicher vorhalten kostet Geld
- Policy notwendig, wann Daten gelöscht werden können/sollten/müssen
  - Tw. rechtliche Rahmenbedingungen (DSGVO)

Hochschule RheinMain

WIE VERTEILEN WIR RECHENLEISTUNG TRANSPARENT?

Containerisierung, Kubernetes, Terraform

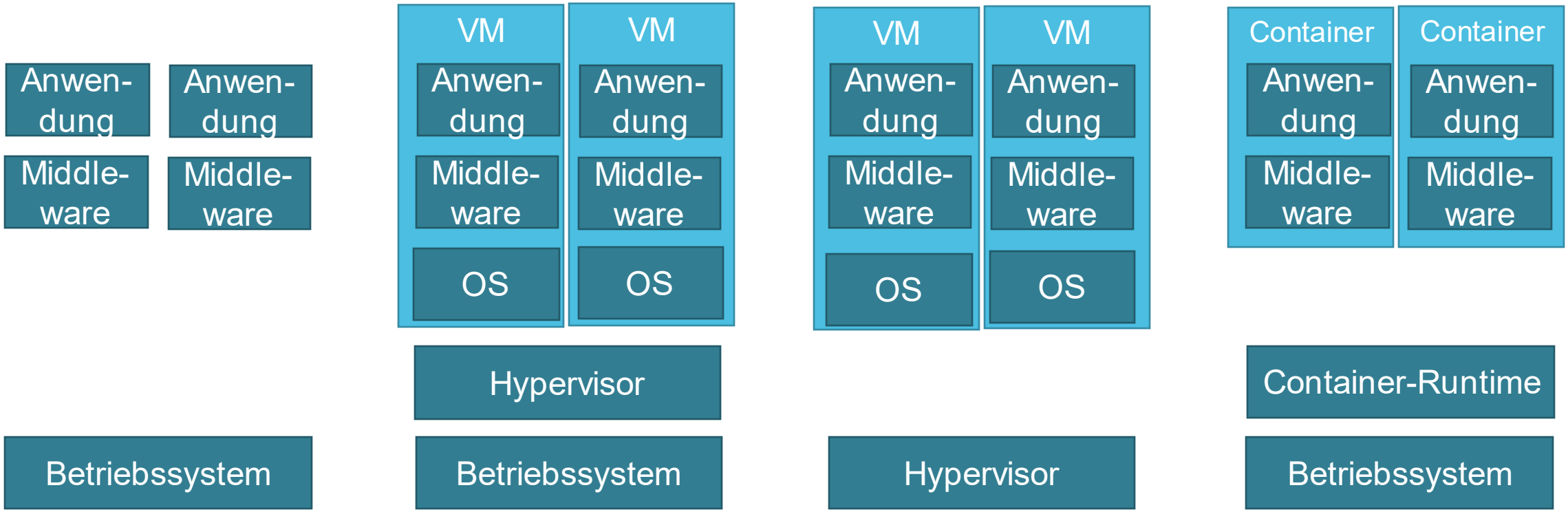
CONTAINERISIERUNG
Einführung

Hochschule RheinMain

- Problem: Wenn wir die Rechenkapazität verteilen wollen, was genau verteilen wir dann?
  - Anwendungen
  - Middleware?
  - Betriebssystem?
  - Hardware?
- Abhängigkeiten
  - Anwendung abhängig von verwendeter Middleware
  - Middleware abhängig von Betriebssystem (aber API oft plattformunabhängig)
  - Betriebssystem typischerweise unabhängig von verwendeter Hardware
- Ziele
  - Minimierung des Laufzeit-Overheads, Isolation der Container
- Beispiele
  - LXC, Solaris container, Docker, OpenVZ, DragonFly BSD vkernels, FreeBSD jails

CONTAINERISIERUNG
Historische Entwicklung

Hochschule RheinMain



Traditionelle Anwendung

Virtualisierung Type-2 Hypervisor

Virtualisierung Type-1 Hypervisor

Containerisierung (OS-level virtualization)

DOCKER
Einführung

Hochschule RheinMain

- Leichtgewichtige Container eingeführt 2013 von Docker, Inc.
  - Breite Unterstützung, insb. Linux und Windows

- Bestandteile
  - Dockerfile: Eigene Sprache, um Container hierarchisch zu spezifizieren (Images)
    - Basiscontainer
    - Befehle für die Initialisierung
    - Enthaltene Dateien
    - Geöffnete Ports
  - Container-Runtime um Images in Containern auszuführen (Linux oder Windows)
  - Dediziertes Tool Docker Compose erstellt Images aus Dockerfiles
  - Verwaltung von Docker Container-Images in der Cloud (DockerHub)

- Docker-Container mittlerweile auch sehr verbreitet für CI/CD-Builds

DOCKER
Command Line Interface

Hochschule RheinMain

Befehl	Bedeutung
docker run <image>	Startet einen neuen Container mit dem angegebenem Image
docker port <container>	Zeigt das Port-Mapping an
docker stop <container>	Stoppt den angegebenen Container
docker ps [-a]	Zeigt eine Liste aller laufenden Container [inklusive der gestoppten]
docker rm <container>	Löscht den angegebenen Container
docker container prune	Löscht alle gestoppten Container
docker pull <image>	Lädt das angegebene Image aus DockerHub
docker build	Erstellt ein neues Image

DOCKER
Dockerfile

Hochschule RheinMain

Präprozessor-Direktive

```
# syntax=docker/dockerfile:1
```

Basis-Image

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 as build-env
WORKDIR /src
```

Arbeitsverzeichnis im Container setzen

Inhalte in den Container kopieren

```
COPY src/*.csproj .
```

Befehl im Container ausführen

```
RUN dotnet restore
COPY src .
RUN dotnet publish -c Release -o /publish
```

Basis-Image wechseln

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 as runtime
WORKDIR /publish
```

Dateien aus vorherigem Container kopieren

```
COPY --from=build-env /publish .
```

Port sichtbar machen

```
EXPOSE 80
```

```
ENTRYPOINT ["dotnet", "myWebApp.dll"]
```

Einstiegspunkt setzen

WARUM WURDE IM BEISPIEL DAS CONTAINER-IMAGE GEWECHSELT?

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 as build-env
WORKDIR /src
COPY src/*.csproj .
RUN dotnet restore
COPY src .
RUN dotnet publish -c Release -o /publish

FROM mcr.microsoft.com/dotnet/aspnet:6.0 as runtime
WORKDIR /publish
COPY --from=build-env /publish .
EXPOSE 80
ENTRYPOINT ["dotnet", "myWebApp.dll"]
```

DOCKER
Alternativen

Hochschule RheinMain

- Podman
  - Verwendet gleiches Format wie Docker Images
  - Open-Source
  - Kann ohne Adminrechte gestartet werden

podman

Hochschule RheinMain

WIE VERTEILEN WIR DIE CONTAINER JETZT?

Kubernetes

KUBERNETES

Hochschule RheinMain

- Container-Orchestrierungssystem von Google, 2014
  - Basierend auf Docker oder virtuellen Maschinen
  - Implementierung in vielen Cloud-Anbietern
  - Beispiele: Minikube (lokal), EKS, AKS, GKE, RKE, …
- Erlaubt es flexibel Gruppen (Deployments) von Containern (Pods) anzulegen, die repliziert auf mehreren Rechnern (Nodes) ausgeführt werden
  - Zusätzlicher Dienst auf jedem Rechner für administrative Zwecke (Kubelet)
  - Lastausgleich je nach Clusterimplementierung automatisch
  - Rollierende Deployments möglich (neue Version eines Images wird versetzt auf Container ausgerollt)
- Problem: Imperatives Modell
  - Deployments werden explizit angestoßen

TERRAFORM

- Open-source Infrastructure-as-Code Tool von HashiCorp, 2014
- Idee: deklarative Beschreibung der gewünschten Infrastruktur
  - Beschreibung in Form von domänenspezifischer Sprache (HCL) oder JSON
  - Terraform gleicht Beschreibung mit Ist-Stand ab und führt notwendige Änderungen aus
  - Provider-Konzept, um unterschiedliche Infrastrukturtypen abbilden zu können
- Mehr im Praktikum (optional)

```
provider "aws" {
  region = var.region
}

…

module "eks" {
  source  = "terraform-aws-modules/eks/aws"
  version = "19.5.1"

  cluster_name    = local.cluster_name
  cluster_version = "1.24"

  …

  eks_managed_node_groups = {
    one = {
      name = "node-group-1"

      instance_types = ["t3.small"]

      min_size     = 1
      max_size     = 3
      desired_size = 2
    }

    …
  }
}
```

ZUSAMMENFASSUNG

Hochschule RheinMain

- Verteilte Dateisysteme ermöglichen nebenläufigen Zugriff unterschiedlicher Nutzer auf Daten unabhängig vom Speicherort
  - NFS als häufigster Vertreter von verteilten Dateisystemen
  - WebDAV als Protokoll über HTTP
- Aufteilung der Funktionalität zur Informationsspeicherung ermöglicht Skalierbarkeit, Fehlertoleranz und Erweiterbarkeit
  - RAID-Verbünde
  - DAS, NAS und SAN
- Verteilung von Rechenleistung
  - Containerisierung, e.g. Docker
  - Clusterverwaltung, e.g. Kubernetes

MÖGLICHE PRÜFUNGSAUFGABEN

Hochschule RheinMain

- Nennen Sie Beispiele für verteilte Dateisysteme!
- Was bedeutet es, ein Verzeichnis zu mounten?
- Erläutern Sie Eventual Consistency bei verteilten Dateisystemen!
- Wie kann man HTTP als verteiltes Dateisystem verwenden?
- Welcher RAID-Verbund hat die höchste Fehlertoleranz? Welcher die niedrigste?
- Warum wird bei RAID selten eine dedizierte Parität (RAID-2 bis RAID-4) eingesetzt?
- Was ist die wesentliche Schnittstelle für ein Network Attached Storage (NAS)?
- Was ist die wesentliche Schnittstelle für ein Storage Area Network (SAN)?