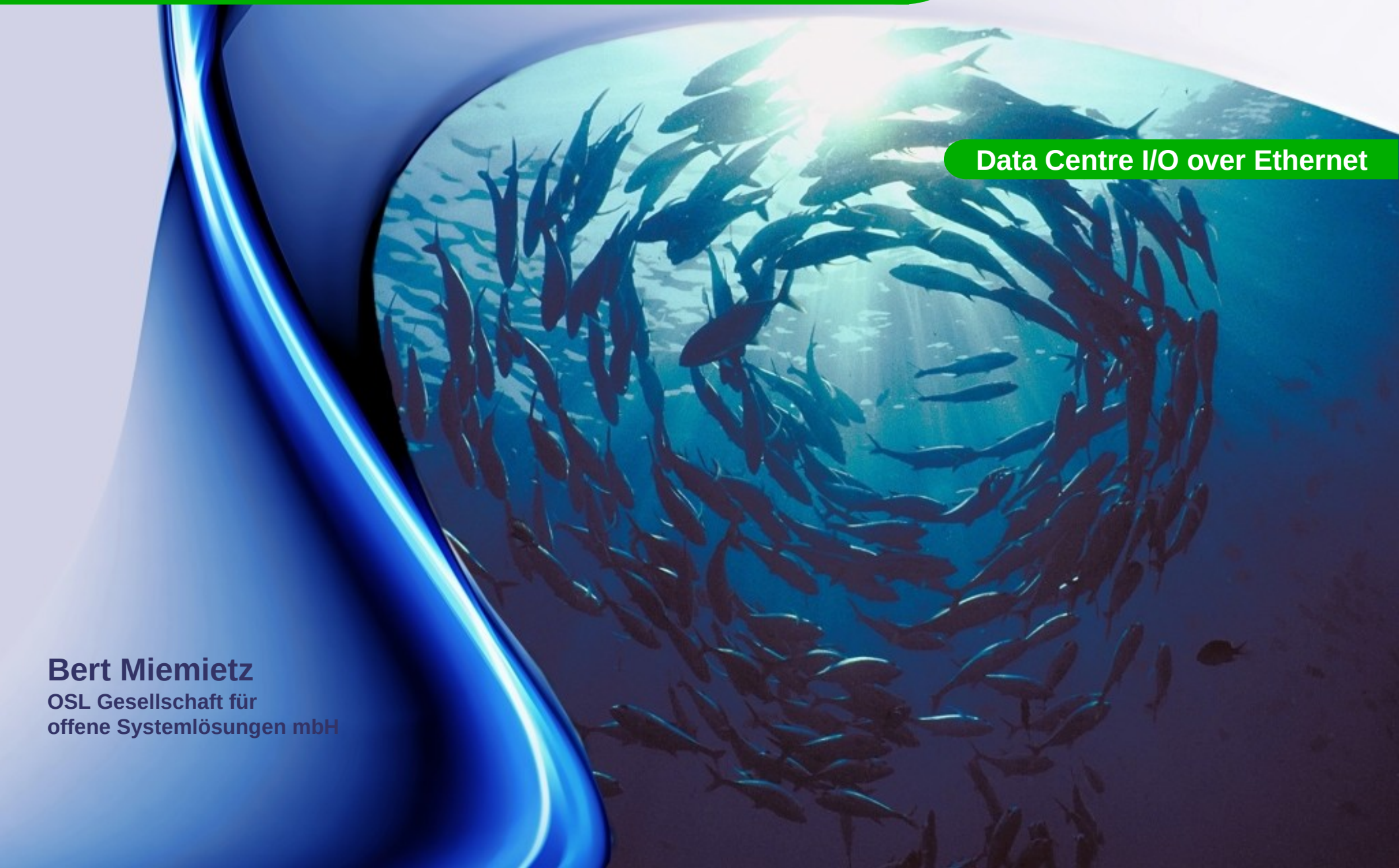
Remote Storage über Standard-Netzwerktechnik

Wie und warum?

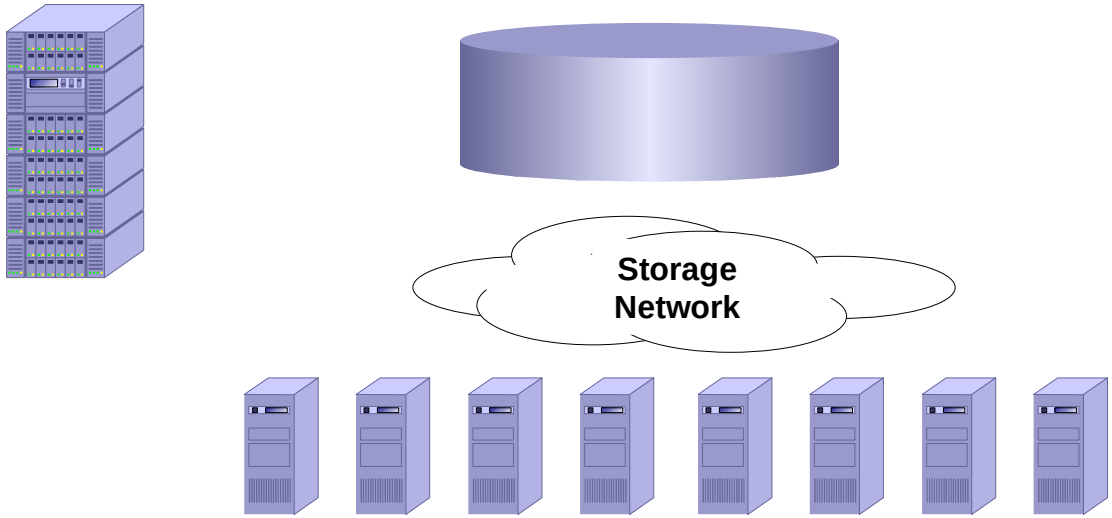


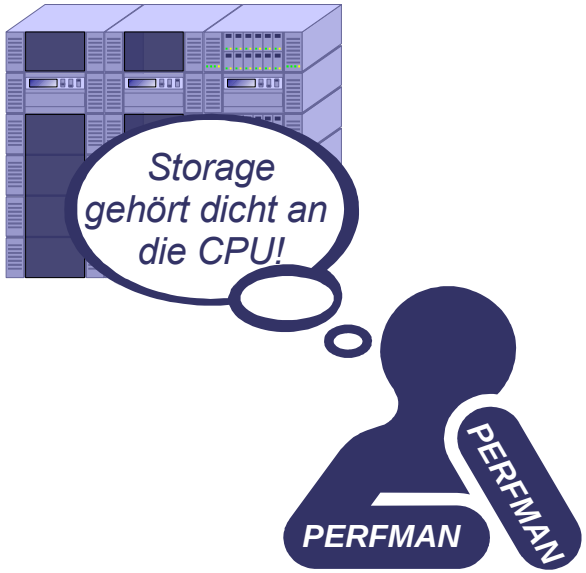
Data Centre I/O over Ethernet

Bert Miemietz
OSL Gesellschaft für
offene Systemlösungen mbH

Speichernetzwerke

Heute Standard im Rechenzentrum – Warum eigentlich?

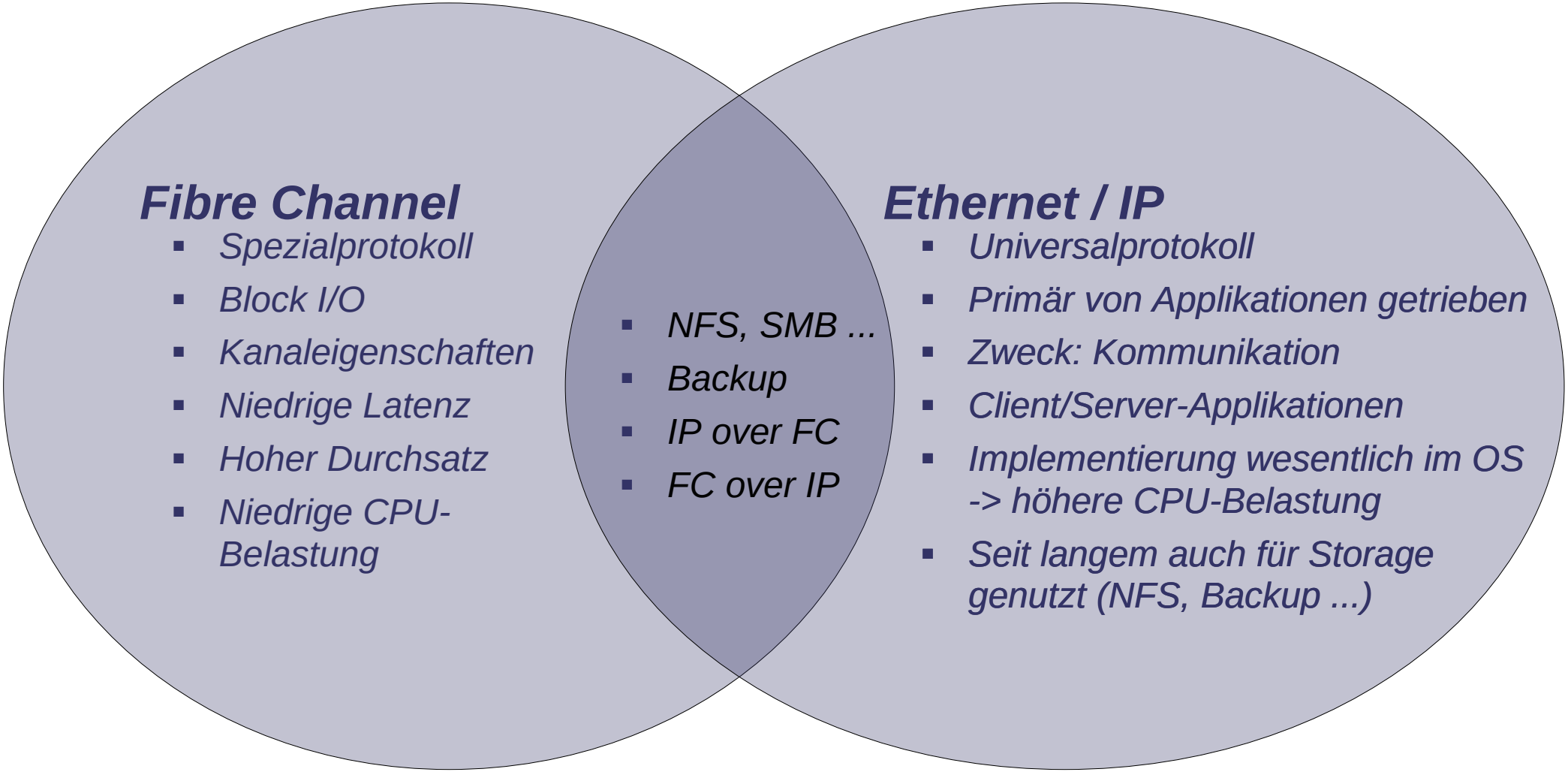




- *Spezialisierung / Funktionsteilung*
  - *Spezialsysteme für einfachere Handhabung, bessere Verfügbarkeit und Performance*
- *Flexibilität*
  - *Trennung Compute Node – Storage erlaubt Anwendungsmobilität, HV, DR*
  - *Trennung ermöglicht diverse Virtualisierungsansätze*
- *Heutige Massenspeichertechnologien sind (relativ) langsam und fehleranfällig*
  - *Netzwerke versprechen Skalierbarkeit und bessere Verfügbarkeit*

Klassisch: Spezialisierte Netzwerke

Verschiedene Netzwerke für verschiedene Anwendungen



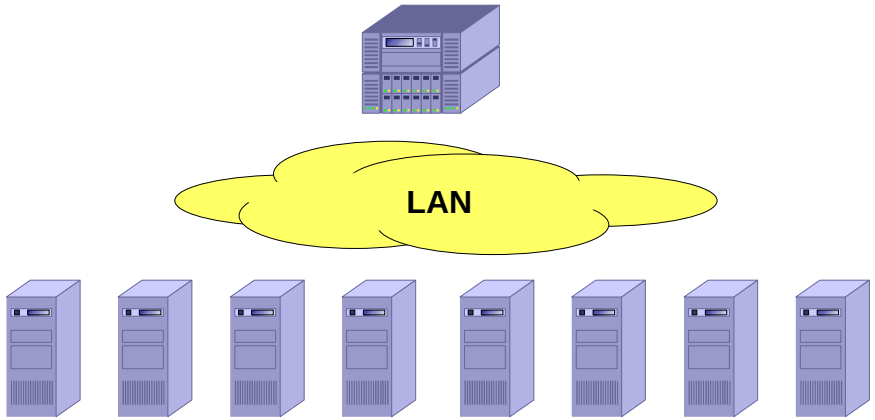
Warum Storage über Ethernet?
Anforderungen und Möglichkeiten

- *Anforderungen und Erwartungen*
  - *Erfordernisse der Anwendungen und Protokolle (Kommunikation, Filesharing etc.)*
  - *Preisliche Motivationen*
  - *Einheitliche Infrastruktur, weniger Ports ?*
  - *Einfachheit, Flexibilität ?*
  - *Virtualisierungstechnologien, Verfügbarkeit von Treibern*
  - *Zusatzfunktionen (Konvertierungen, Filesystemsnapshots ...)*
- *Möglichkeiten*
  - *Gigabit-LAN heute vergleichsweise preiswert*
  - *Gigabit-LAN heute in vernünftiger Relation zur Geschwindigkeit einer Festplatte bzw. eines RAID-Systems*
  - *Gigabit-LAN heute in günstiger Relation zu Durchsatz-Anforderungen der Applikationen*
  - *Mehrere Gigabit-Ports je Server*
  - *Ethernet ist eigentlich (fast) kein Ethernet mehr -> Switching-Technologie*
  - *RAID-Systeme / Filer sprechen direkt die erforderlichen Protokolle*
  - *Neue Performance-Erwartungen durch 10GBit-Ethernet*

Storage über Ethernet heute: NFS, SMB, CIFS

NA(F)S präsentiert sich mit handfesten Vorteilen

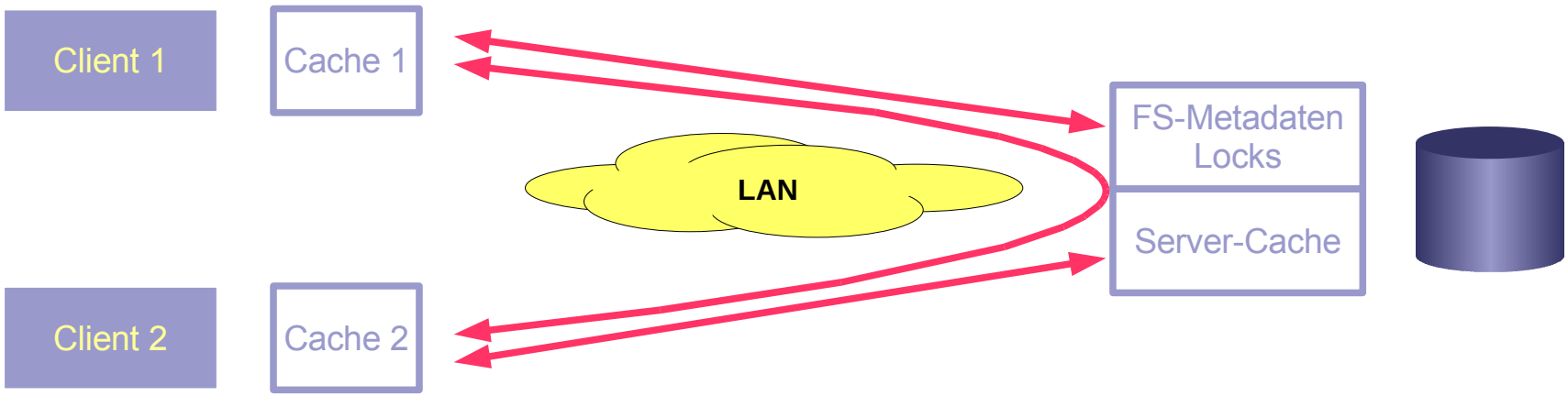
- *Spezialisierung auf Fileservices, dafür relativ einfache Handhabung*
- *ermöglichen Filesharing*
- *Können komplexe RAID-Funktionen verbergen*
- *dateisystemtypische Funktionalitäten wie Snapshots und weitere Sonderfunktionen*
- *weite Verbreitung und Unterstützung der wichtigsten Protokolle*
- *im Rahmen des heute Vorstellbaren Möglichkeiten nahezu ausgereizt*



Die Kehrseite des NAFS*-Ansatzes

Es gibt auch prinzipbedingte Nachteile

- *aufwendige Integration mit Server-OS (Zugriffskontrolle, User-Management)*
- *Cache- und Cohärenzproblematik, schwierige Nutzung der Client-Ressourcen*
- *nicht trivial: Skalierbarkeit, Parallelisierung, Hochverfügbarkeit, Multipathing*
- *feste Bindung an File-Access-Semantik*
- *mit zunehmender Funktionalität auch Zunahme von Komplexität und ggf. Inkompatibilitäten*

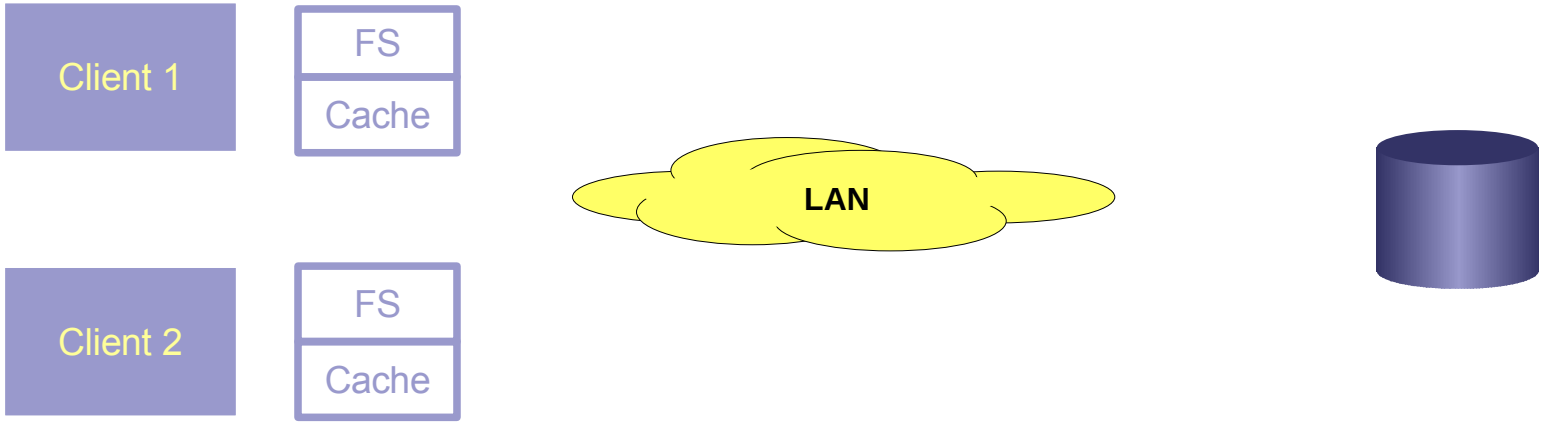


***NAFS – Network Attached Filesystem**

Starke Argumente für Block-I/O im RZ

Jenseits von Filesharing überwiegen die Vorteile

- *Volle Kontrolle des Client-OS über das Storage-Device*
- *Nutzbar für beliebige Filesysteme*
- *Keine Kopplung an Server-OS (Isolation, privates Identity Management)*
- *Nur Übertragung von I/O, nicht von Cache-Inhalten*
- *Cache liegt beim Client -> schnellster Zugriff, Client-Caches summieren sich auf*
- *Einfache Administration, schlankes Protokoll, hohe Geschwindigkeit*



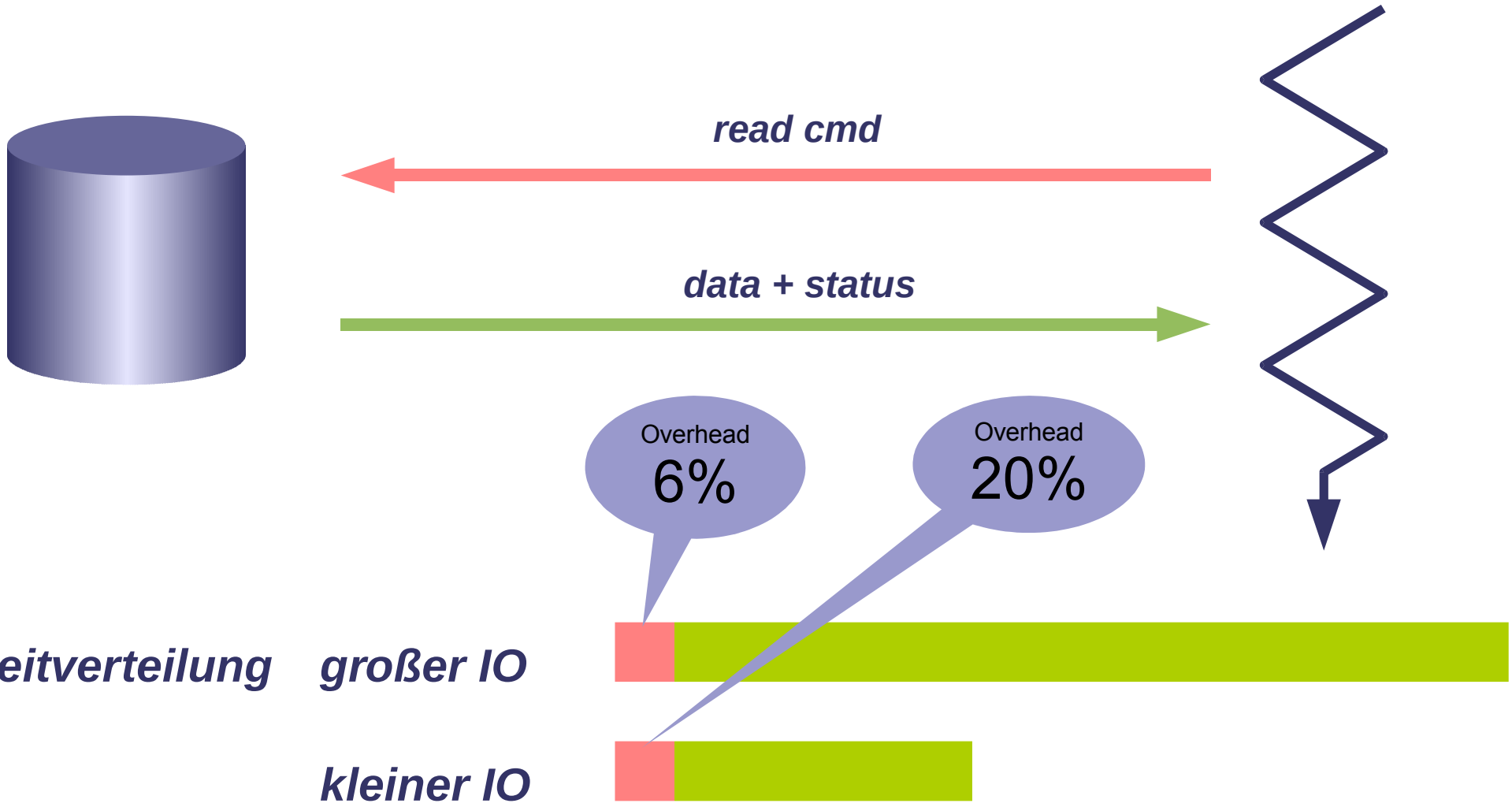
Block-I/O über Ethernet/IP - Performance

Performance hat viele Aspekte

- *Latenz / Service Time*
  - *Relevant für einzelnen I/O und single threaded I/O*
- *Maximaler Durchsatz*
  - *Relevant für multithreaded I/O*
- *Skalierbarkeit / Parallelisierbarkeit*
  - *Thema für Multipathing*
- *IO-Größe*
  - *Relevant für Nutzdatenanteil am Gesamtdurchsatz*

Bedeutung von Latenzen und IO-Größe

Heutige Paradigmen setzen Grenzen



Storage over Ethernet: Aktzeptable Latenzen?

Betrachtungen zur Performance

Zeit zur Übertragung von 32 Byte an 1 GBit-Lanboard

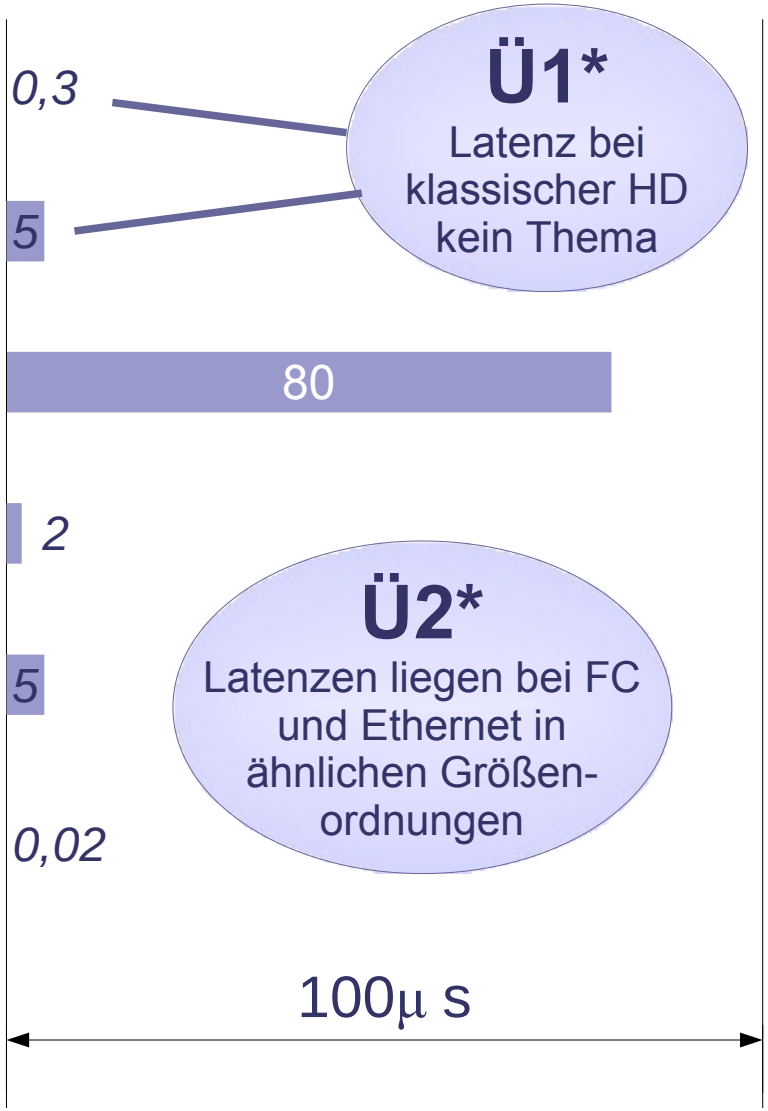
Zeit zur Übertragung von 512 Byte an 1GBit-Lanboard

Zeit zur Übertragung von 8kByte an 1GBit-Lanboard

memcpy 8k auf 600MHz-System

Threadwechsel per CV auf 600MHz-System

32Bit-Typkonvertierung auf 600MHz-System



Storage over Ethernet: Aktzeptable Latenzen?

Betrachtungen zur Performance

Zeit zur Übertragung von 32 Byte an 1 GBit-Lanboard

Zeit zur Übertragung von 512 Byte an 1GBit-Lanboard

Es geht also:

Zeit zur Übertragung von 8kByte an 1GBit-Lanboard

1. Weniger um die Hardware.

2. Entscheidend um das Protokoll.

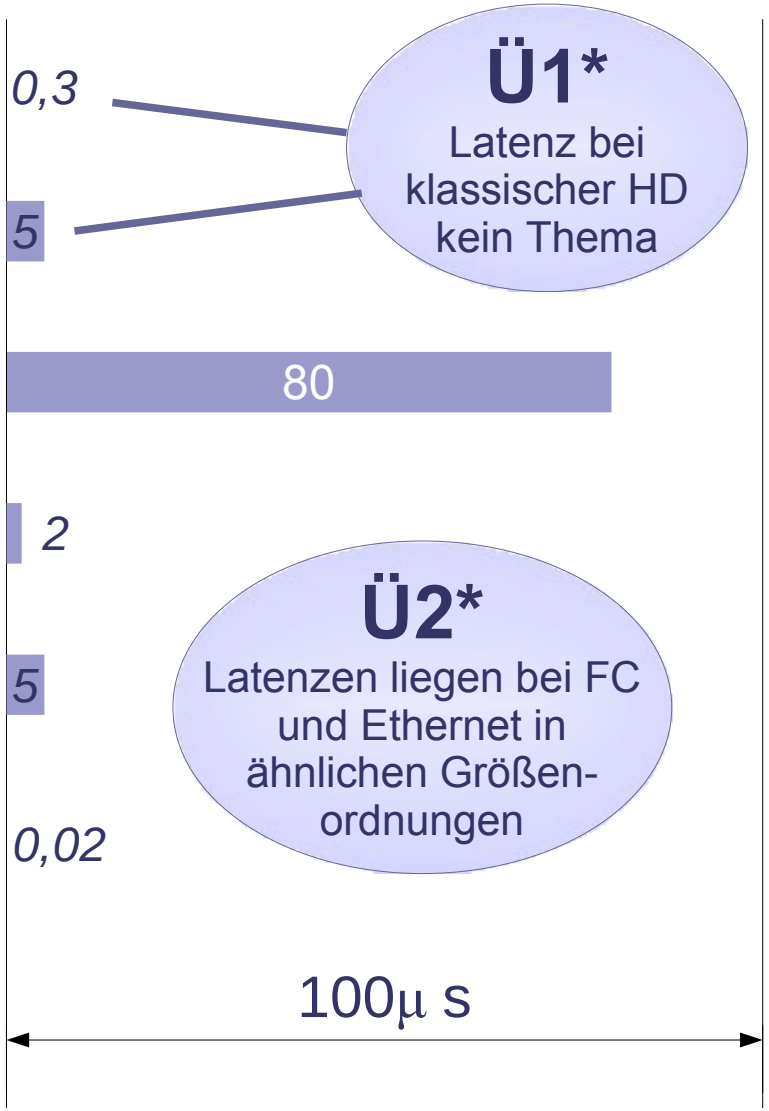
memcpy 8k auf 600MHz-System

3. Entscheidend um die Systemsoftware.

Threadwechsel per CV auf 600MHz-System

4. Nicht unerheblich um die Applikation.

32Bit-Typkonvertierung auf 600MHz-System



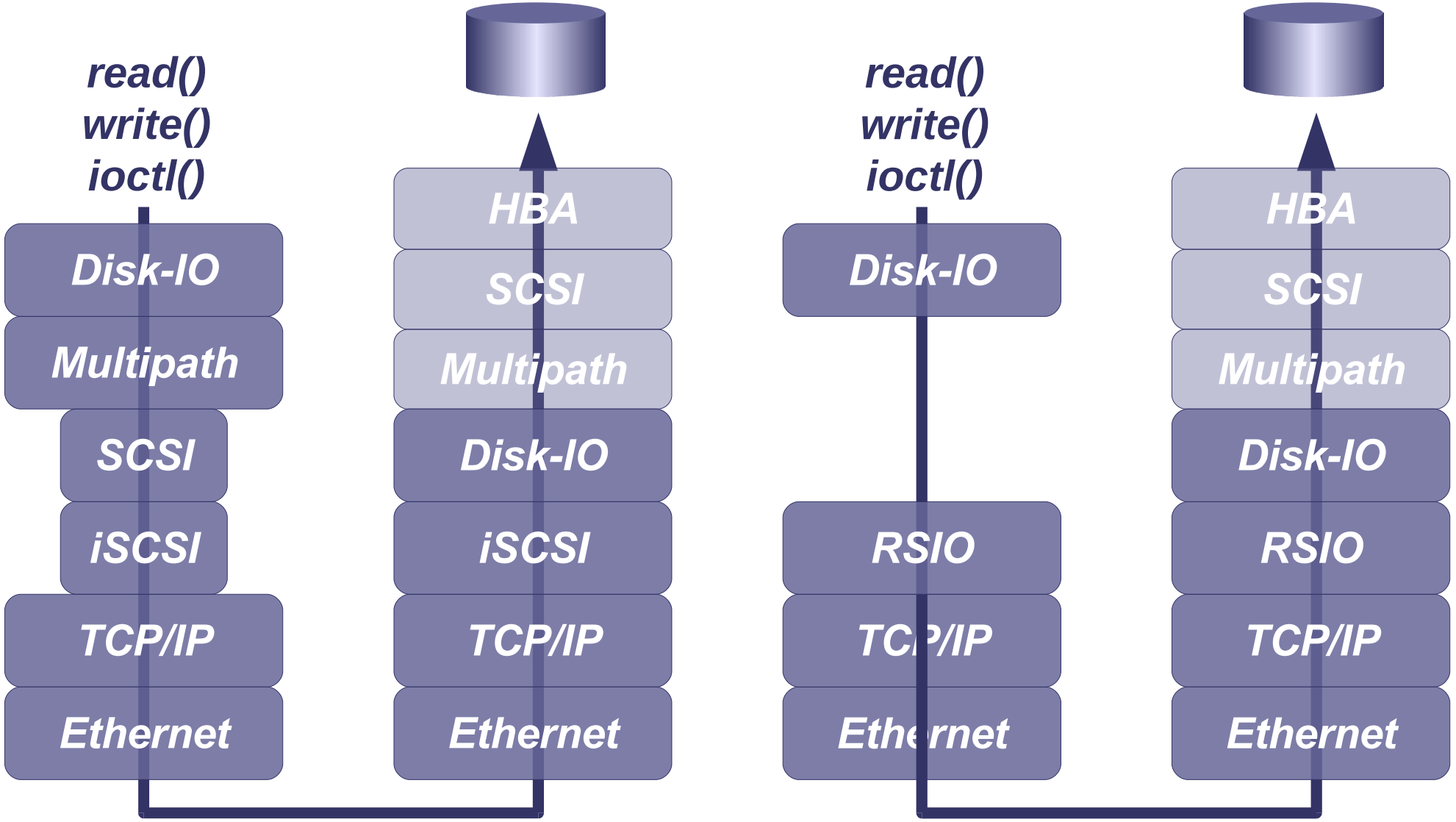
Block-I/O mit iSCSI

Bekanntes Protokoll über neues Medium

- *Low-Level-Protokoll auf IP umgesetzt*
- *Server (Target) kann alle Plattformen mit Initiator bedienen*
- *Starke Bindung an TCP, Offload-Engines auf Initiatorseite dennoch selten*
- *Tiefer Stack – nicht unerheblicher CPU-Bedarf*
- *Zahlreiche SCSI-Funktionen, aber: aus Sicht der Anwendungen bringt die Weitergabe über verschiedene Protokollschichten dennoch Verlust an Funktionalität -> Storage-Management meist über andere Protokolle*
- *Kaum spezialisierter Support für vernetzte, geclusterte Speichersysteme*
- *Höherer Schwierigkeitsgrad bei gehobenen Anforderungen:*
  - *Multipathing*
  - *Clustering / Parallelisierung*
  - *Nomenklatur*
  - *Target Portal Groups*

Block-I/O über Ethernet/IP – auch ohne SCSI ?

Ein anderer Ansatz



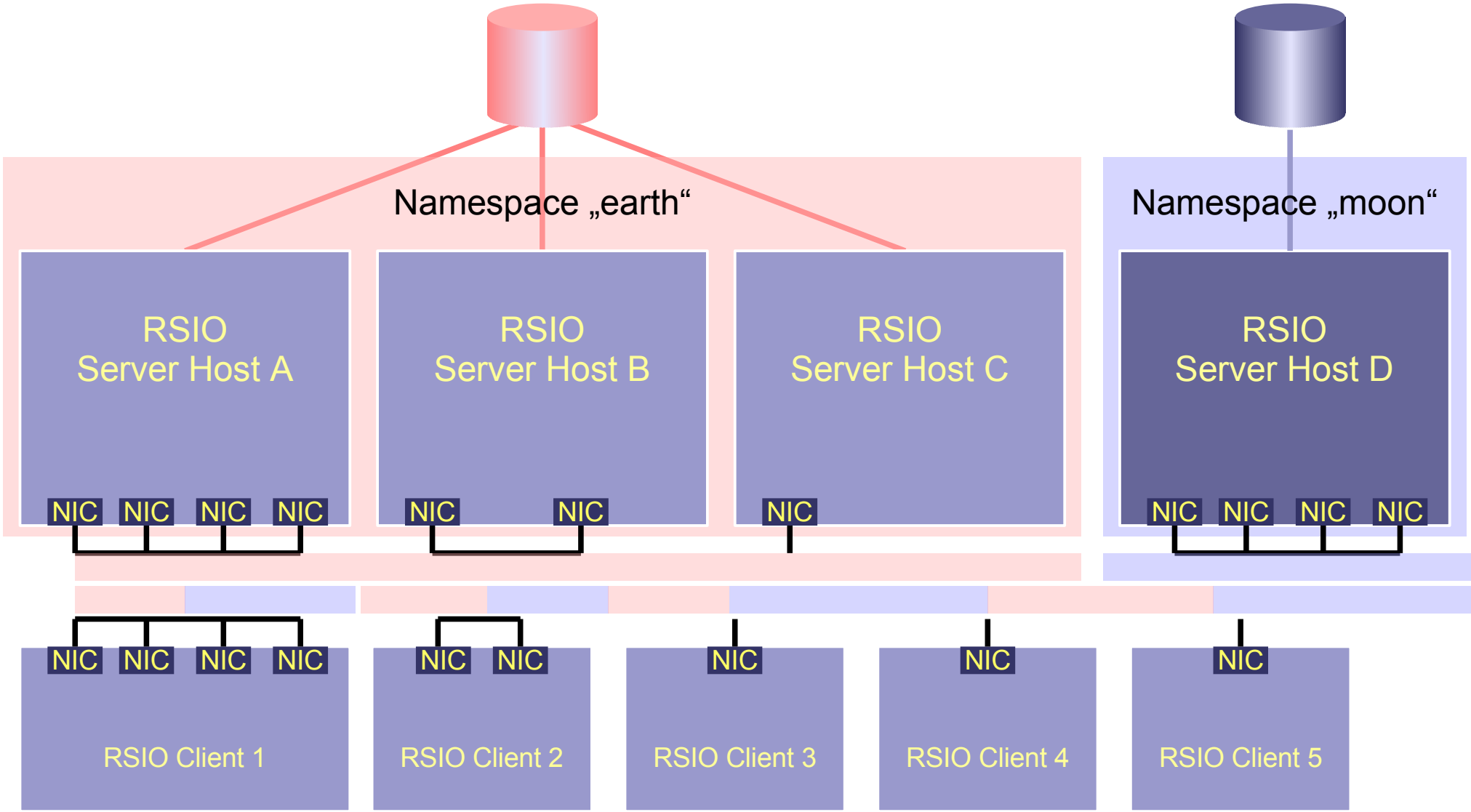
RSIO - Remote Storage IO

Eckdaten einer neuen Technologie

- *von OSL entwickeltes Protokoll*
- *natürliche Erweiterung der Speichervirtualisierung für Transport über Netzwerke*
- *voller Clustersupport möglich (Client und Server)*
- *adressiert LAN-attached (shared) Block Devices*
- *zielt auf vollständige Umsetzung der vorgenannten Aufgabenstellungen*
- *hochportable Implementierung*
- *internes Layout berücksichtigt moderne CPU- und Serverkonzeptionen*
- *guter Durchsatz und gute Verfügbarkeit mit heutiger Technik*
- *Design zugleich fokussiert auf einfache Handhabung*

RSIO - Architektur im RZ

Klar gegliedertes und flexibles administratives Konzept



Parameter der RSIO-Architektur

Flexible Client-Server-Implementierung

- *Ein Namespace definiert Server (und Clients) mit Zugriff auf dieselben Storage-Ressourcen*
- *Auf einem Serverhost können (nahezu) beliebig viele Server(prozesse) laufen*
- *Jeder Serverhost kann (nahezu) beliebig viele Clients bedienen*
- *jeder Client unterstützt den Zugriff auf bis zu 256 Server*
- *jede Maschine (Client und Server) unterstützt bis zu 8 Interfaces*
- *Das Protokoll erlaubt Clients simultan Zugriff auf mehrere Namespaces*
- *Integriertes Multipathing/Trunking mit Auto-Explorer*
  - *Ermitteln verfügbarer Verbindungen*
  - *Ermitteln der Schnittstelleneigenschaften*
  - *Test der Parameter auf der Übertragungsstrecke*

RSIO – weitere Designschwerpunkte

Details aus dem Anforderungskatalog

- *Zuverlässigkeit und Skalierbarkeit*
- *einfache Handhabung auch in komplexeren Topologien (kein Zoning)*
- *Unterstützung heutiger wie zukünftiger Transport-Technologien*
- *Nutzbarkeit preiswerter Komponenten ermöglichen*
- *vollständige Abbildung aller relevanten IO-Aufrufe*
- *Multithreading-Support*
- *mit IP: Routingfähigkeit*
- *Erweiterbarkeit, Raum für intelligente IO-Lösungen*
- *Einbindung in Clustertechnologien*
- *Applikationsbezogene, integrierte, bequeme Administration vom Host aus*
- *Gute Performance bei Standard-Netzwerkparametern*

RSIO – Übersicht zum Protokoll

Details zur technischen Umsetzung

- *Definition eigener Frames*
  - *Unabhängigkeit von TCP*
  - *ermöglicht Loadbalancing und Multipathing*
  - *Optionen wie Checksum / Encryption*
  - *Frames mit variabler Größe*
  - *Overhead per Frame nur 16 Byte*
- *Trennung von Treiber und Transport*
  - *größerer Funktionsumfang bei hoher Portabilität*
  - *besseres Error-Handling*
  - *Performance offensichtlich kein Problem*
  - *hochflexibler Multithreading-Support*
  - *bessere Abschirmung des Kerns*
- *Integriertes Multipathing und Trunking (s. Frames)*
- *Selbstkonfiguration und Error Recovery*
- *Unterstützung geclusterter Server*

RSIO – Performanceeigenschaften des Protokolls

Performance folgt dem Design

Server-Performance bei Cache(Mem) Read / 8k

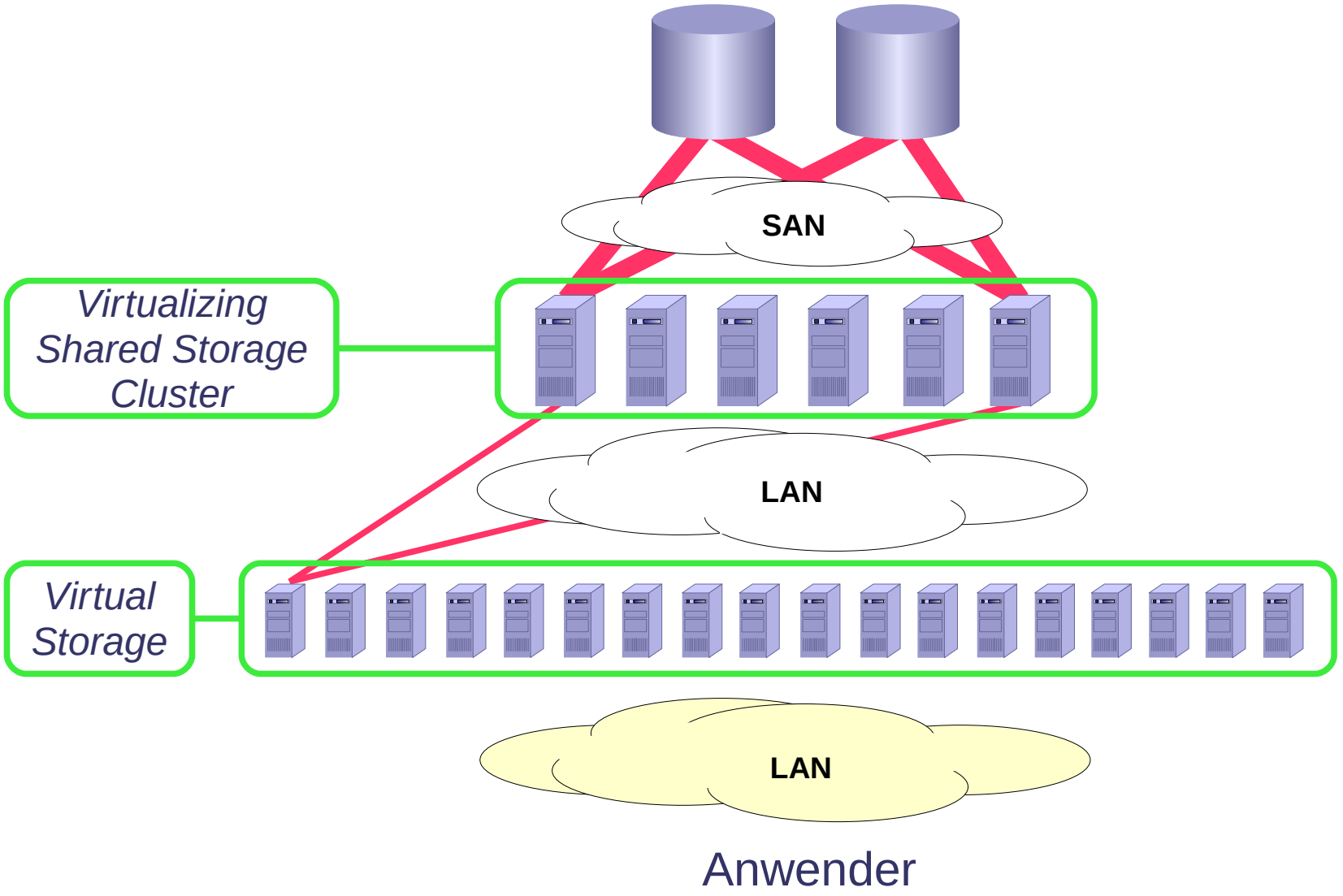
iSCSI (Generic OS)	*10 Clients*	*100 Threads*	*7,6 Cores*	***31.000 IOPS***
iSCSI (Special OS)	*10 Clients*	*100 Threads*	*10,0 Cores*	***85.000 IOPS***
RSIO (Generic OS)	*4 Clients*	*64 Threads*	*5,6 Cores*	***98.000 IOPS***
RSIO (Generic OS)	*4 Clients*	*128 Threads*	*6,3 Cores*	***102.000 IOPS***

Client-Performance Throughput

RSIO	*1 x 1 GBit*	*< 0,5 Cores*	***> 110 MByte/s***
RSIO	*2 x 1 GBit*	*< 1,0 Cores*	***> 220 MByte/s***
RSIO	*4 x 1 GBit*	*< 2,0 Cores*	***> 440 MByte/s***

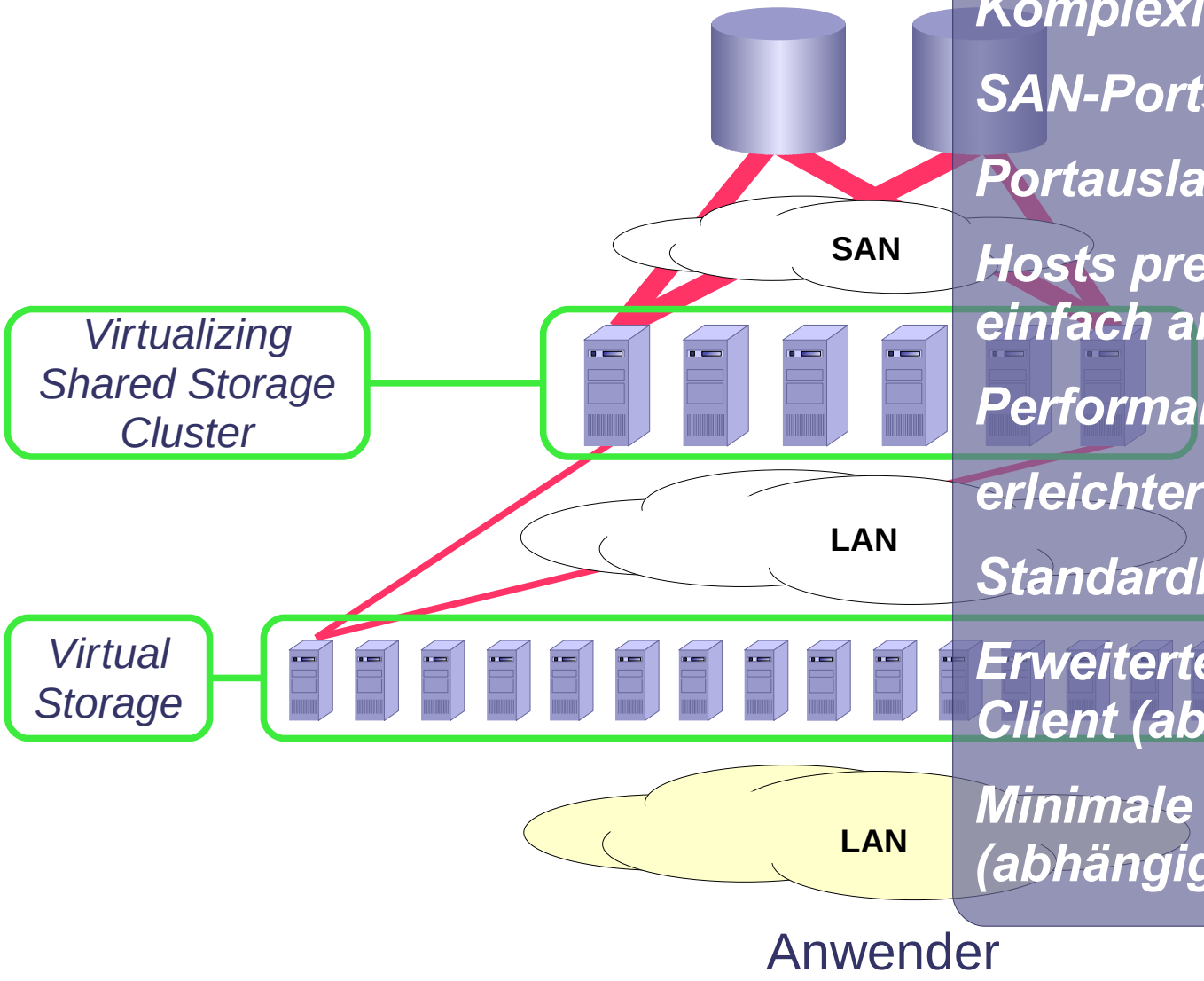
Einsatzszenarien für RSIO - Allgemein

Clustered Server – clustered Clients



Einsatzszenarien für RSIO - Allgemein

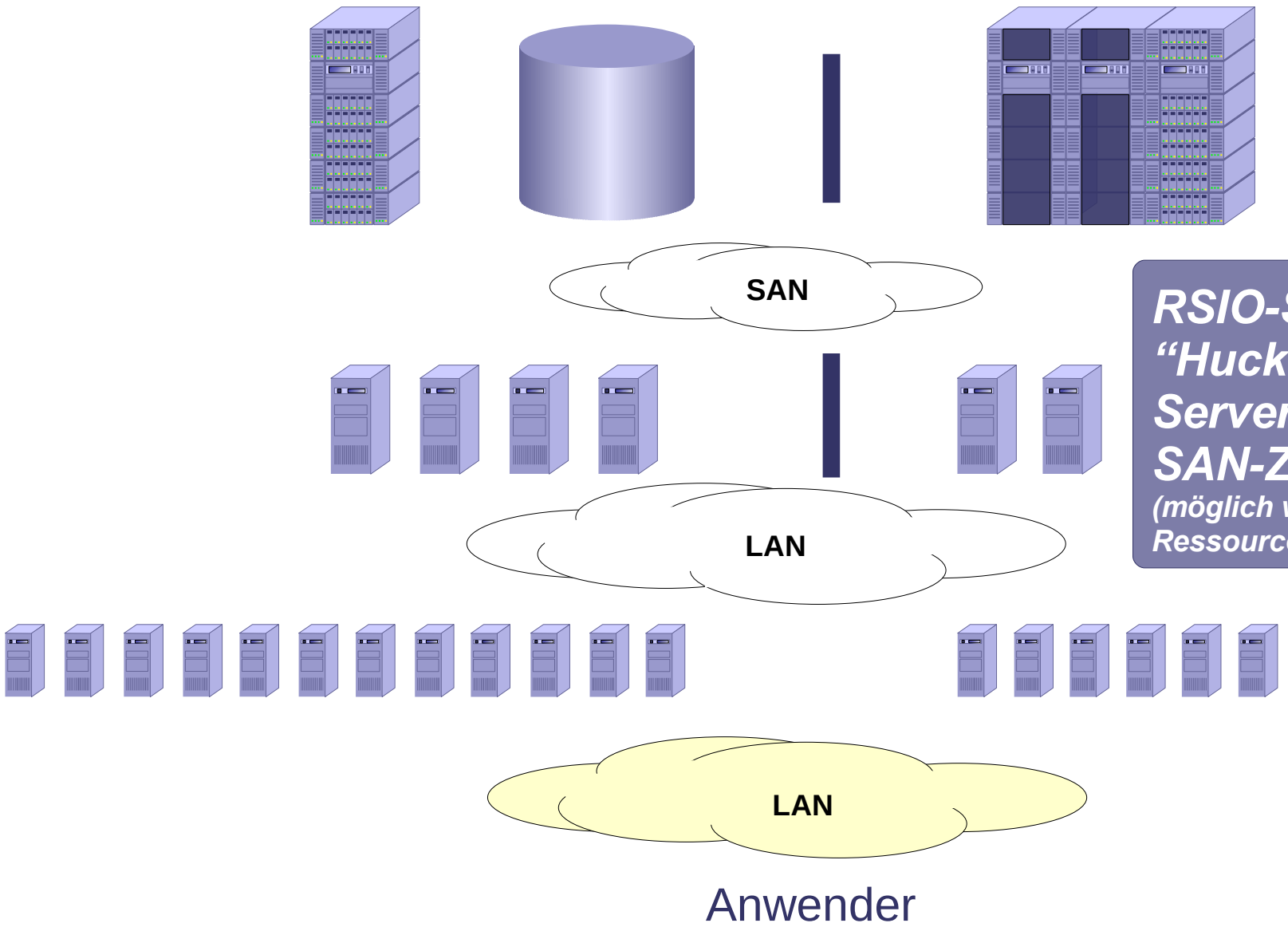
Clustered Server – clustered Clients



- Komplexität reduzieren
- SAN-Ports einsparen
- Portauslastung verbessern
- Hosts preiswert und einfach anbinden
- Performance-Rightsizing
- erleichterte OS-Virtualisierung
- Standardkomponenten
- Erweiterter Funktionsumfang für Client (abhängig vom Server)
- Minimale LAN-Belastung (abhängig vom Server)

Einsatzszenarien für RSIO – Real Data Centre

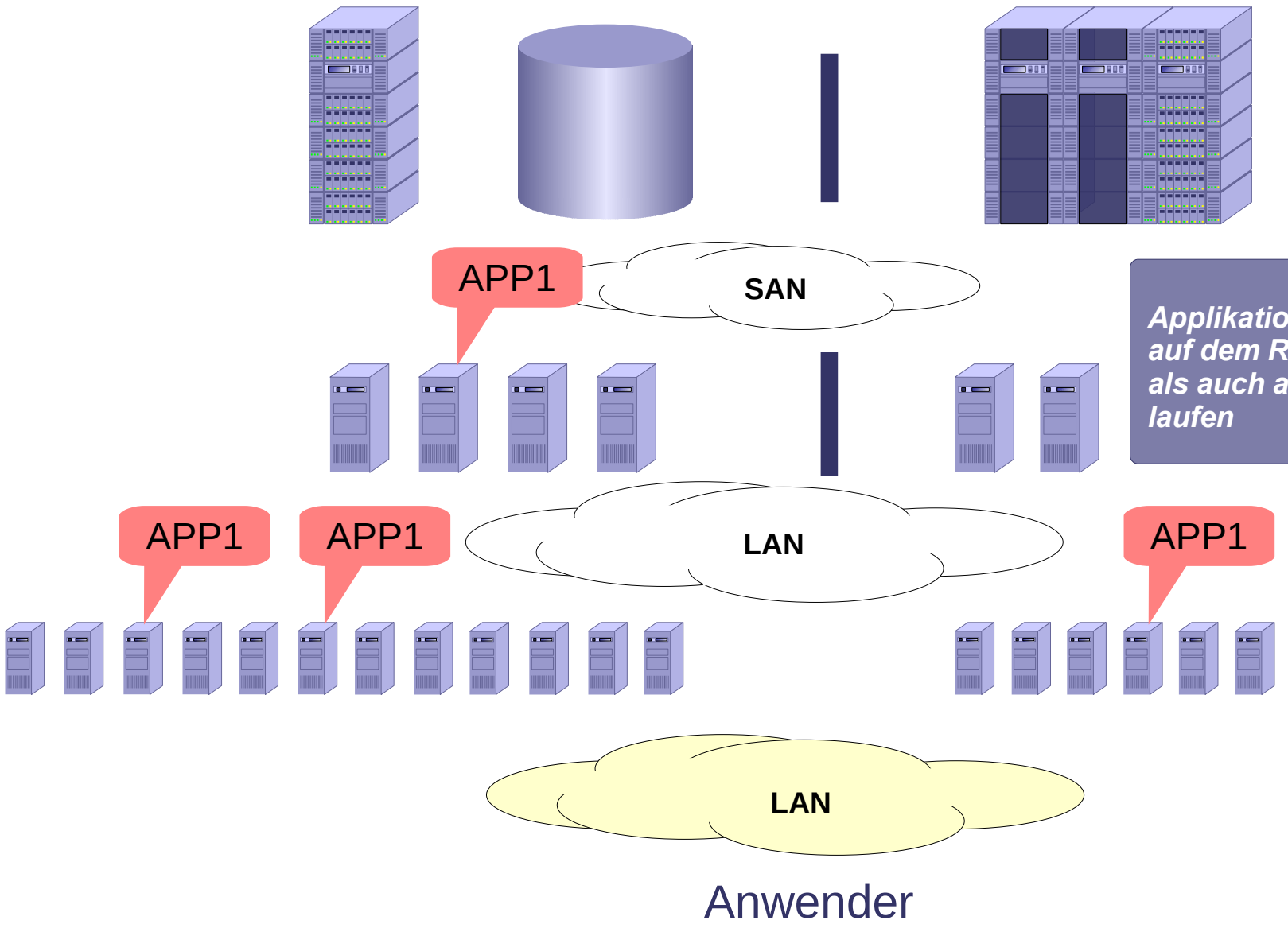
SAN-LAN-Konvergenz



RSIO-Server läuft "Huckepack" auf Servern mit SAN-Zugriff
(möglich wegen des niedrigen Ressourcenbedarfes)

Einsatzszenarien für RSIO – Hochverfügbarkeit

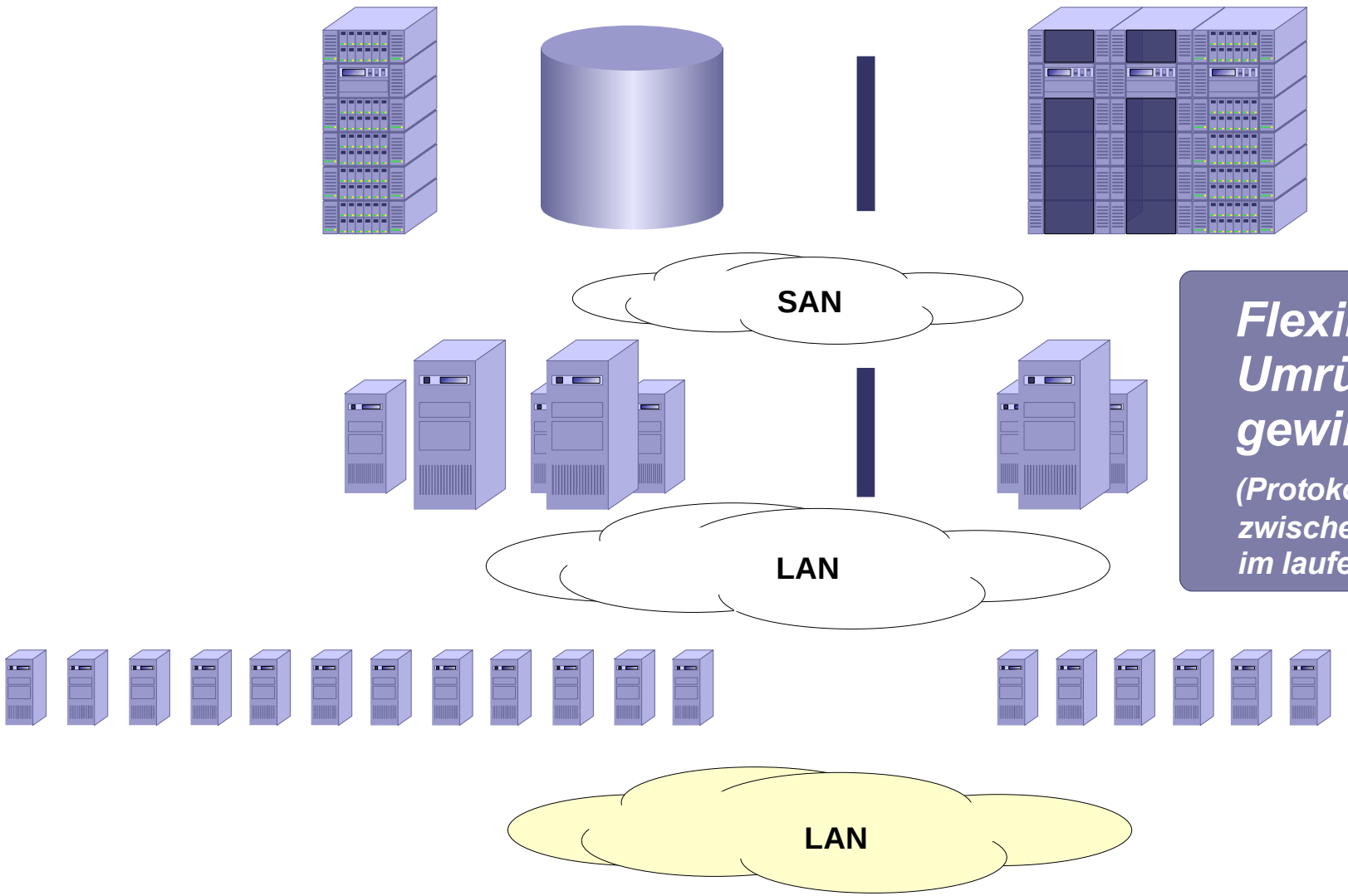
Auch hier: SAN-LAN-Konvergenz



Applikationen können sowohl auf dem RSIO-Server (SAN) als auch auf dem Client (LAN) laufen

Einsatzszenarien für RSIO – Flexibles RZ

Von den Protokolleigenschaften profitieren

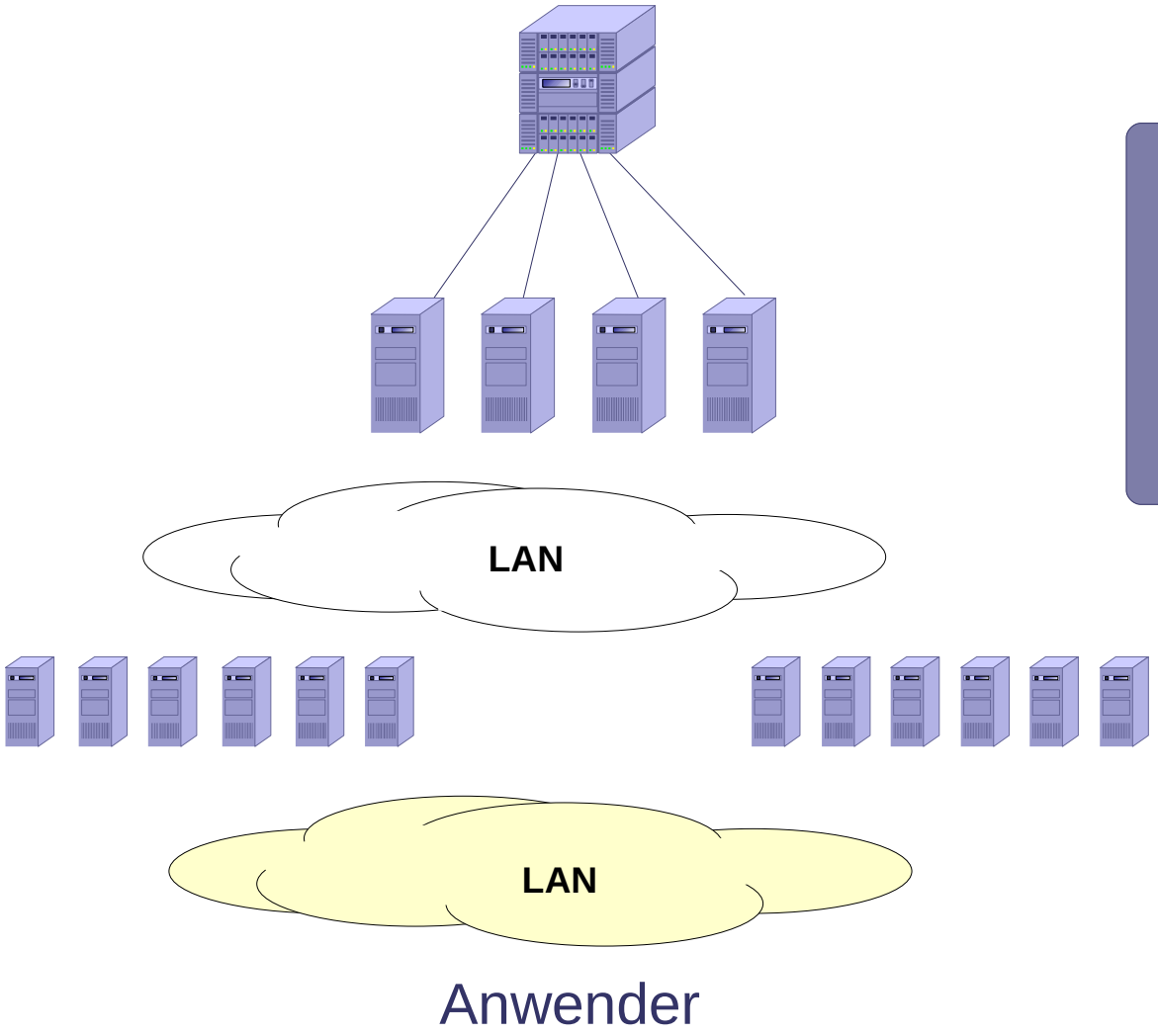


Flexibilität bei Umrüstungen gewinnen

(Protokoll ermöglicht Wechsel zwischen Servern im laufenden Betrieb)

Einsatzszenarien für RSIO – Auch das geht ...

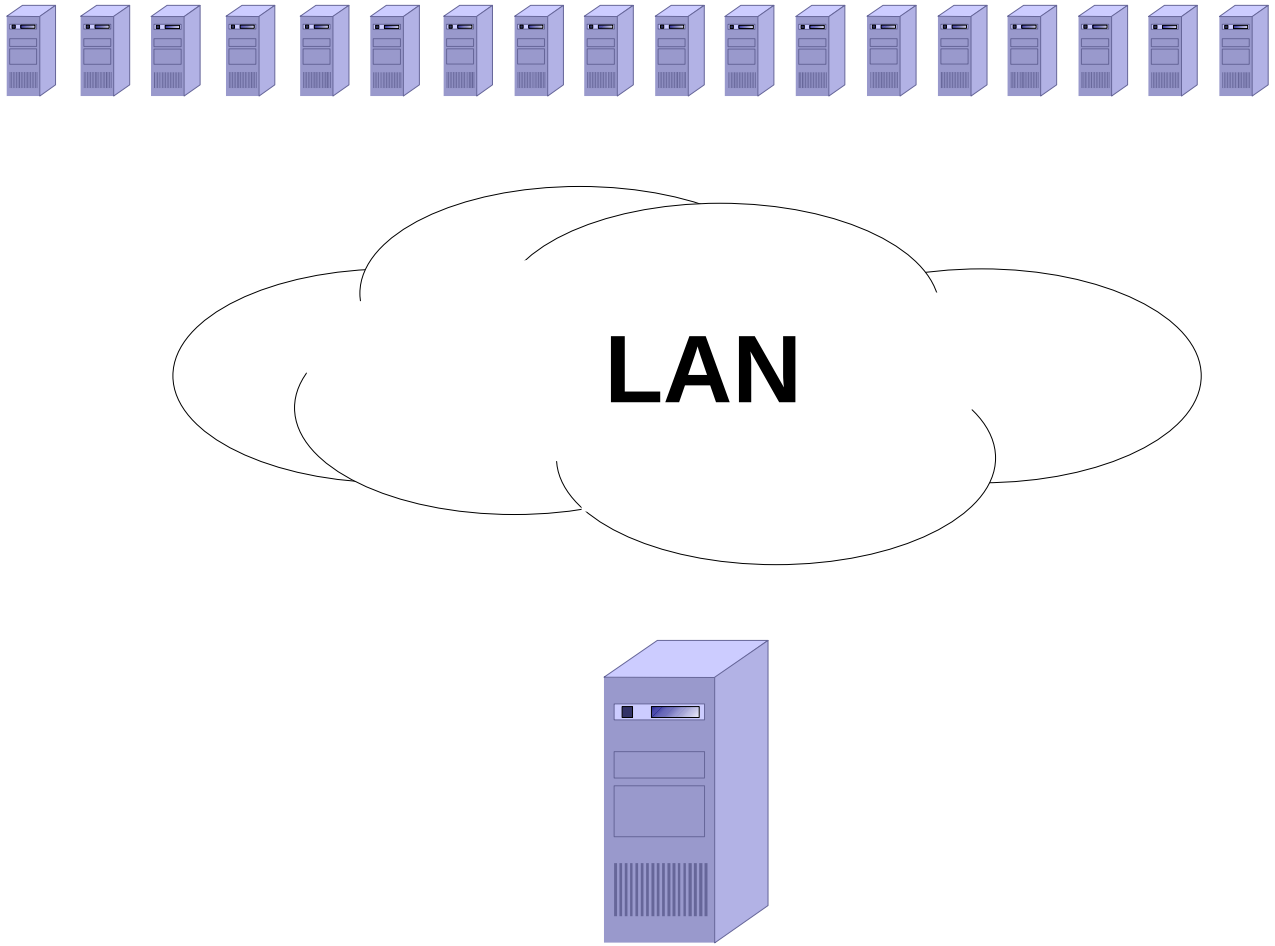
Collapsed SAN – Gar kein SAN



ganz auf das SAN verzichten, nicht aber auf Funktionalität

Einsatzszenarien RSIO – Storage “einsammeln”

Mögliche Aggregation von Kapazität und Bandbreite



Storage over Ethernet: Erste Erfahrungen

Was man empfehlen kann und was nicht

- *Extreme Lasten (>400 Mbyte/s je Client) gehören auf Fibre Channel*
- *Zusätzliche Virtualisierung auf der Client-Seite kann Performancenachteile mit sich bringen*
- *Spiegelung, RAID mit voller Datenkopie, Backup über LAN ?*
- *Anbindung mit 2 bis 4 Ports -> gute Performance bei niedriger CPU-Last*
- *Nutzung über 10GBit-Interfaces -> Auslastung?*
- *Virtualisierung auf der Server-Seite -> sehr gute Performance*
- *Kombination mit hostbasierter Virtualisierung auf Serverseite sehr zu empfehlen*

Der Funktionsbooster: Speichervirtualisierung

Serverseitige blockbasierte Virtualisierung – bedarfsgerecht + performant

Basis-Virtualisierung

clusterweit

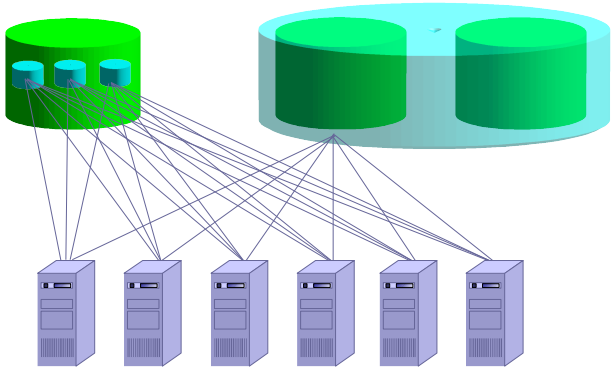
globale Pools

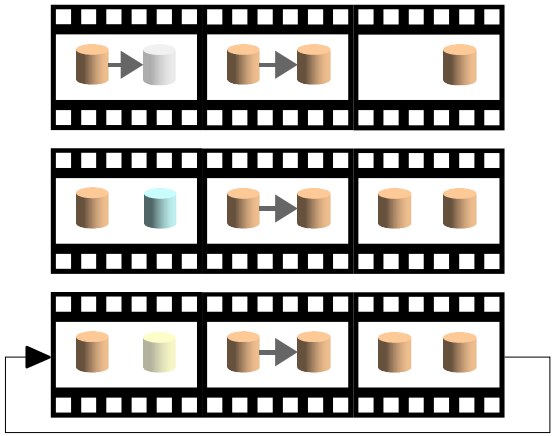
Daten verschieben

Daten clonen

Daten spiegeln

Sonderfunktionen





Virtual Volumes
linear oder integriert (simple, concat, stripe)
HW-Abstraktion und IO-Multipathing
systemgestützte Allokation
Online-Konfig./-Dekonfig./-Vergrößerung

global devices / global namespace
access management

rechnerübergreifend
global inventory
verschnittfreie Ausnutzung

hochverfügbar

online Daten verschieben / reorganisieren

anwendungsorientiert

einmalig oder kalendergesteuert Ziele kopieren
Operationen für mehrere Volumes

skalierbar

Dauerhafte Beziehung Master -> Image
inkrementeller Resync
Operationen für mehrere Volumes
Überbrückung Fehler

Extended Controls
Bandbreitensteuerung
detaillierte Statistik

Zusammenfassung: Storage over Ethernet

Bewertung der Situation im RZ

- *Block-IO bei vielen Anforderungsprofilen unverzichtbar*
- *4/8GBit FC erlaubt den meisten Applikationen mehr IO-Durchsatz als 10GBit Ethernet -> klarer Fall bei extremen IO-Anforderungen.*
- *Brauche ich in jedem Fall > 300MB/s pro Port?*
- *Über 2xGigabit sind nominell 234 MB/s möglich.*
- *Storage over IP bedeutet höhere CPU-Belastung, aber:*
- *heutige Multicore-Systeme bieten i.d.R. reichlich Rechenleistung*
- *auch größere DB-Systeme fordern oft nur 40-60 MByte/s ab*
- *Via Dual-/Quad-Port Gigabit kann ich Systeme unschlagbar preiswert anbinden, dabei zugleich performant und redundant*

Zusammenfassung: Storage over Ethernet

Die Argumente für ein anderes Herangehen

- *sich anbahnendes 10GBase Ethernet ermöglicht Durchsätze in neuen Dimensionen*
- *Server bringen ab Werk bereits mehrere (Gigabit)-Ethernet Ports mit, weitere lassen sich preiswert nachrüsten*
- *Hardware für effektives Multipathing vorhanden*
- *Durchsatz für viele Anwendungen ausreichend*
- *SAN-Administration nicht immer leicht – Vereinfachungen möglich*
- *ggf. Verbesserungen in der Bediensicherheit (Verfügbarkeit) möglich*

RSIO in der Familie der Block-I/O-Protokolle

Die richtige Positionierung

- *RSIO wird zumeist schwächer sein als FC*
  - *nutzt “schwächeres” Universalprotokoll TCP/IP*
  - *bei IP werden erhebliche Kommunikationsteile im OS gerechnet*
  - *u. U. stärker dort, wo der Server nicht über SAN/SCSI auf Storage zugreift*
- *Performance-Boost gegenüber NAFS durch IO-Vermeidung*
  - *virtueller IO-Cache mit exklusivem Zugriff*
  - *dadurch Performance-Vorteil gegenüber NFS/SMB*
  - *in manchen Konstellationen aber auch leichte Nachteile denkbar*
  - *Shared Block Device ist aber kein Shared Filesystem!*
- *RSIO hat per Design Performance-Vorteile gegenüber anderen IP-Block-I/O Lösungen*
  - *erheblich schlankeres Protokoll*
  - *auf Parallelisierung optimiertes Design*
- *Performance-Gewinn durch Multithreading*

Was wir von RSIO erwarten ...

Den Anforderungen aus der Praxis stellen

- *überlegene Performance*
  - *keine Spezialsettings für TCP/IP -> Performance “out of the Box”*
  - *bis jetzt TCP, UDP wird folgen*
  - *prinzipielle Eignung für beliebige Medien*
  - *noch diverse Verbesserungsmöglichkeiten*
- *gewaltige Gestaltungs-, Entwicklungs- und Tuningmöglichkeiten*
- *extrem schlankes Kernelmodul*
- *Isolation der Serverprozesse möglich*
- *Server kann sich automatisch an die anstehende Last anpassen*
- *Integriert viele Problemstellungen (TCP/IP-Handling, Trunking ...)*
- *vielfältige Nutzungsmöglichkeiten / unglaublich viele Szenarien darstellbar*

Wir freuen uns auf Ihre Ideen!