RSIO - Storage Networking der nächsten Generation

SNW Europe 2010 – 27G15 / Deutsch
Data Center Technologies Networking Track

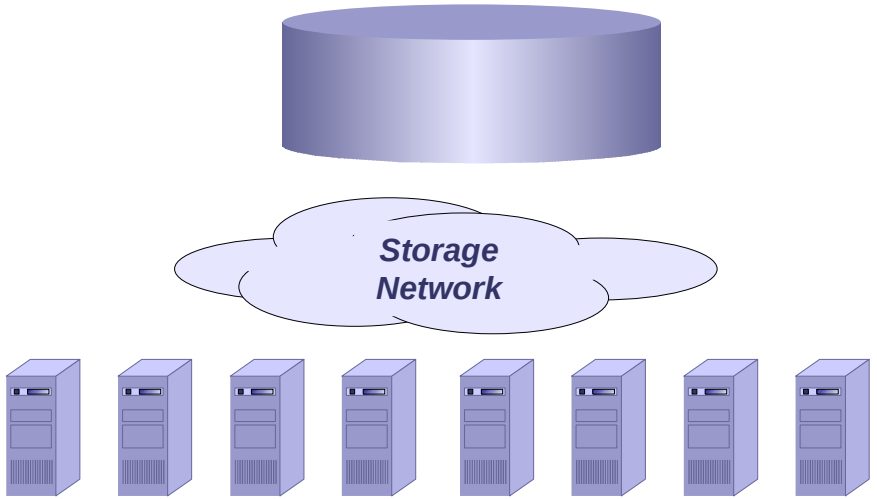
RSIO, das neue Protokoll von OSL für Block-I/O über Standard-LAN-Infrastrukturen, zielt auf die Anforderungen von Rechenzentren: Virtualisierter Speicher über LAN, skalierbarer, hoher Durchsatz und hohe Verfügbarkeit, Unterstützung für parallele und geclusterte Infrastrukturen und mulithtreaded Connections, die von modernen CPU-Architekturen profitieren. Erfahren Sie Hintergründe zur Technologie und verschaffen Sie sich einen Überblick zu den vielfältigen Funktionen und Möglichkeiten, zur einfachen Administration, zu Einsatzszenarien und handfesten Vorteilen...

Bert Miemietz

OSL Gesellschaft für offene Systemlösungen mbH

Speichernetzwerke

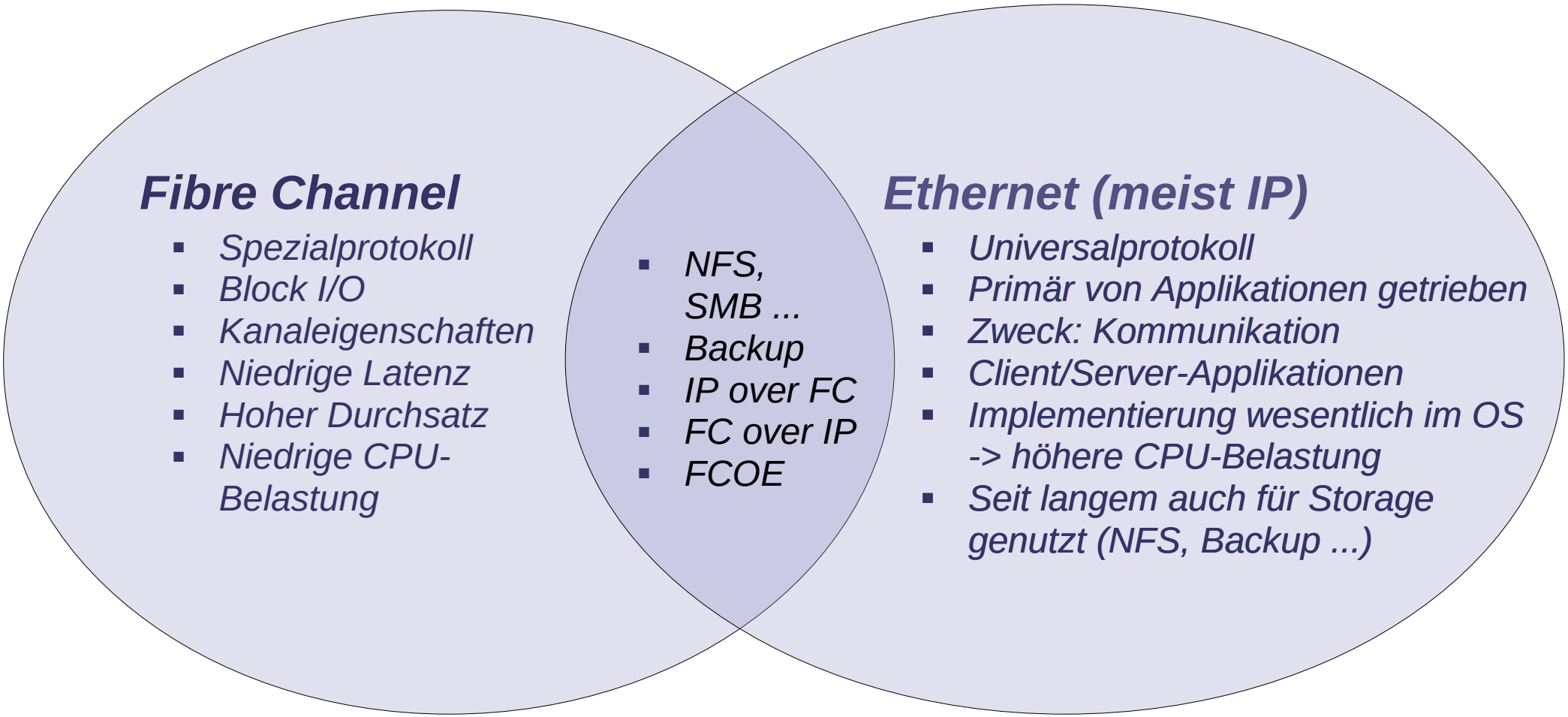
Heute Standard im Rechenzentrum – Warum eigentlich?



- *Spezialisierung und 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*

Speichernetzwerke in der Praxis

Die heute wichtigsten Technologien für kommerzielle Anwender



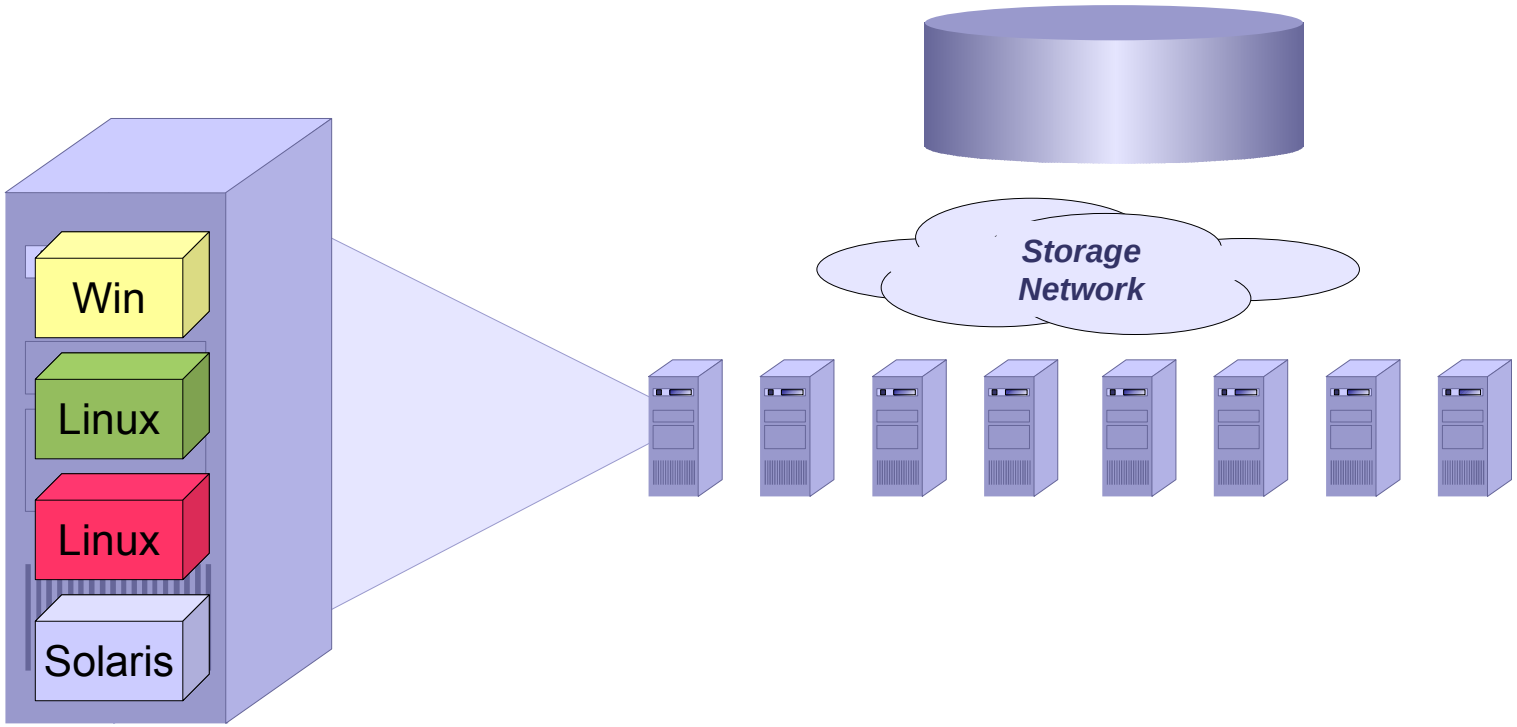
Warum Storage über Ethernet?
Anforderungen und Möglichkeiten

- **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 so schnell wie eine Festplatte*
  - *Gigabit-LAN heute mit applikationsadäquaten Durchsätzen*
  - *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 an 10GBit-Ethernet*

Warum Storage über Ethernet?

Noch ein ganz wichtiger Punkt ...

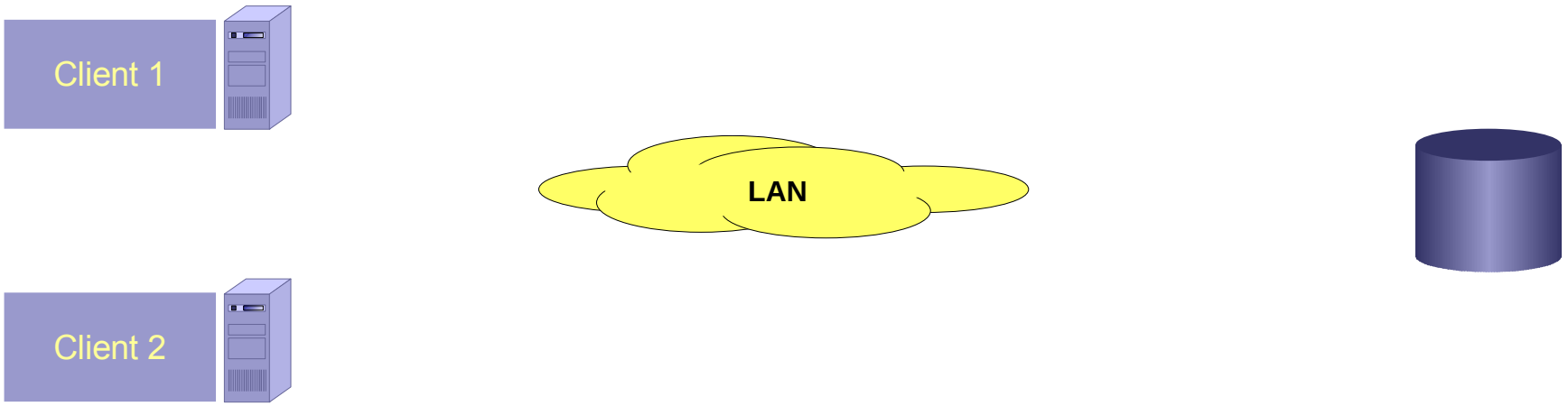


Vielfalt an Virtualisierungstechnologien, Plattformen ...
erhöht Uniformierungsdruck bei Connectivity

Storage über Ethernet heute: NFS, SMB, CIFS

NA(F)S hat durchaus einiges zu bieten*

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

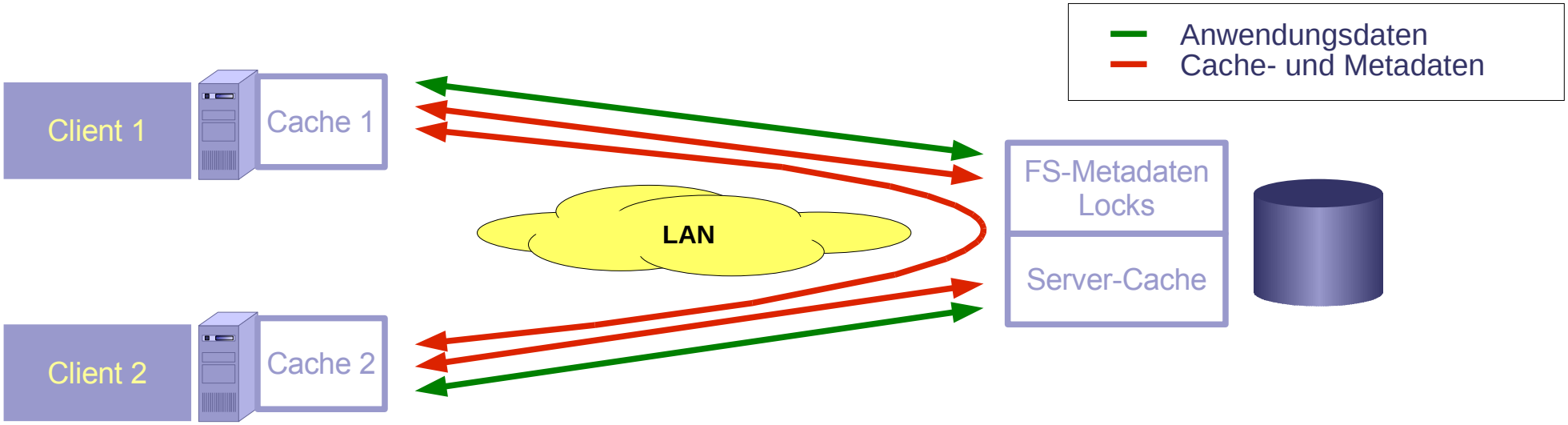


***NAFS – Network Attached Filesystem**

Storage über Ethernet: Die Kehrseite von NAFS*

Es gibt auch prinzipbedingte Nachteile

- *aufwendige Integration mit Server-OS (Zugriffskontrolle, User-Management)*
- *Cache- und Cohärenzproblematik, schwierige Nutzung der Client-Ressourcen*
- *eingeschränkter Durchgriff auf I/O-Steuerung für Applikationen (Datenbanken)*
- *nicht trivial: Skalierbarkeit, Parallelisierung, Hochverfügbarkeit, Multipathing*
- *feste Bindung an File-Access-Semantik*
- *erweiterte Funktionalität geht einher mit Komplexität und Inkompatibilitäten*

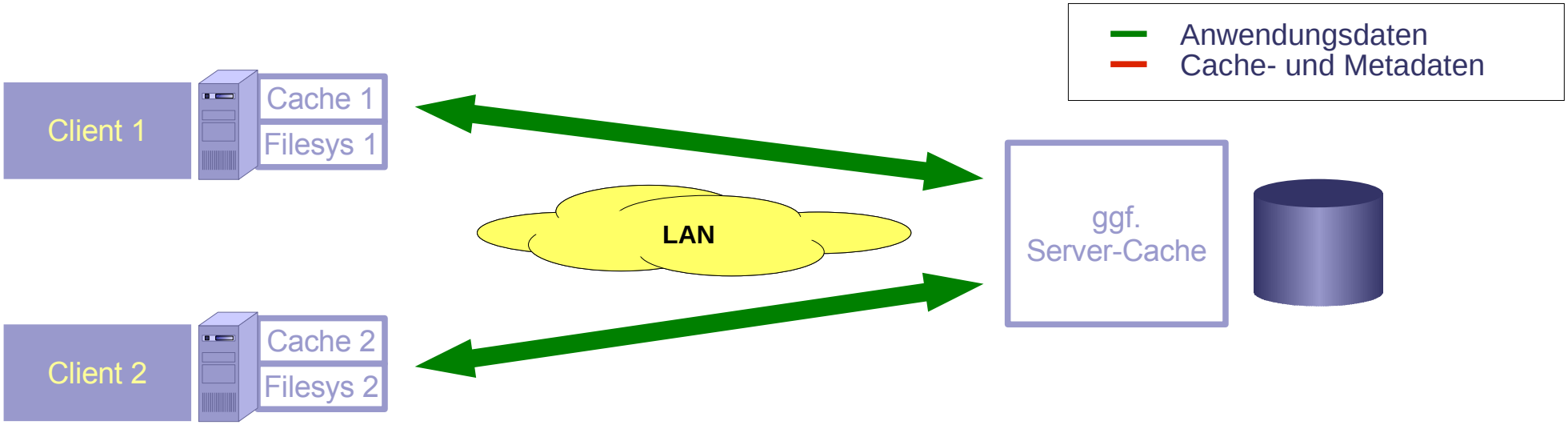


***NAFS – Network Attached Filesystem**

Storage über Ethernet: RZ-Anwender brauchen Block-I/O

Jenseits von Filesharing überwiegen die Vorteile

- *volle Kontrolle des Client-OS über das Storage-Device*
- *nutzbar für beliebige Filesysteme und Applikationen, IO-Verhalten gut steuerbar*
- *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*



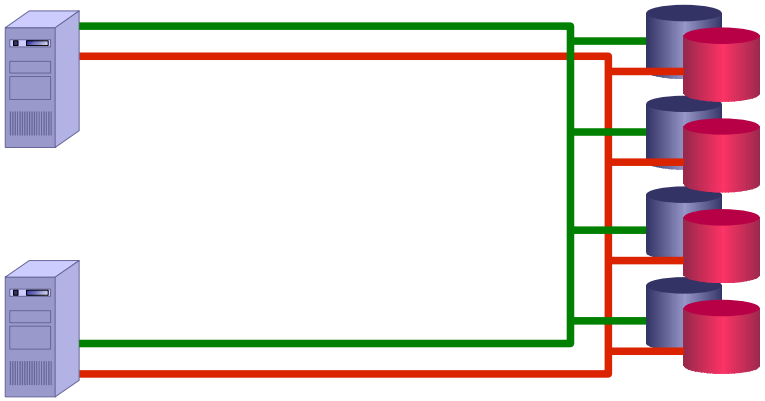
Block-I/O mit iSCSI

Bekanntes Protokoll über designfremdes Medium

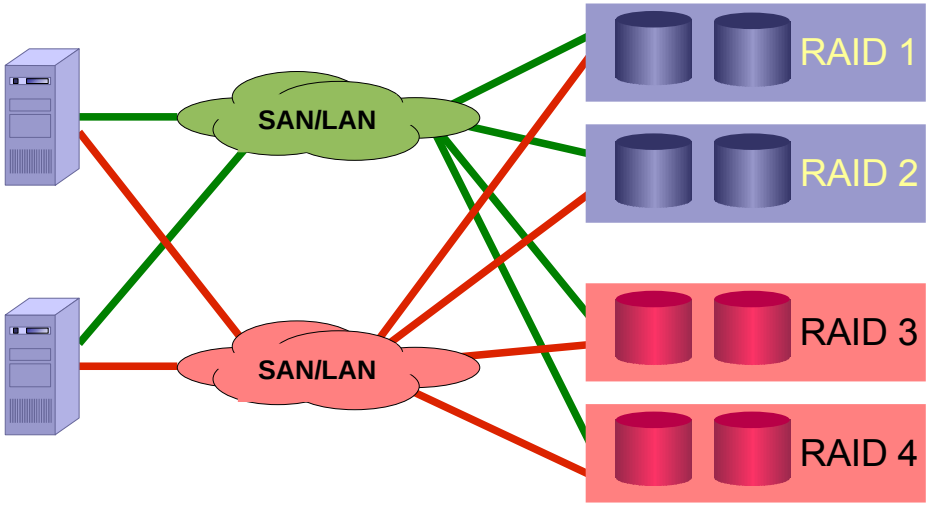
- *Low-Level-Protokoll auf IP umgesetzt (SCSI in TCP/IP-Paketen)*
- *Server (Target) kann alle Plattformen mit Initiator bedienen*
- *Starke Bindung an TCP, Offload-Engines auf Initiatorseite dennoch selten*
- *Tiefer Stack – nicht unerheblicher CPU-Bedarf*
- *Trotz zahlreicher SCSI-Funktionen Transparenzverlust*
  - *aus Anwendungssicht bringt die Weitergabe über verschiedene Protokollschichten 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 mit iSCSI

Bekanntes Protokoll über designfremdes Medium



Klassisch: Daisy Chain



Heute: Vernetzte Struktur

- *Compute Nodes geclustert*
- *Storage-Server geclustert*
- *mehrere Pfade zum Storage*
- *Storage Replication*
- *Storage Systeme sind oft SMP-Server*
- *viele über SCSI nicht abbildbare Zusatzfunktionen*
- *Plattengeometrie meist uninteressant*

Block-I/O über Ethernet – einmal anders gedacht

Worum geht es eigentlich bei Block-I/O über Netzwerke?

- *Für den **Administrator**: leichte Handhabung*
  - *leicht nachvollziehbare Abbildung der Verbindungen*
  - *administratorfreundliche Namen*
  - *Einbeziehung von Multipfad, Speichervirtualisierung, Clusterkonzepten etc.*
  - *Einbindung von Sonderfunktionen (Datenspiegel)*
- *Für den **Anwender**: Performance und Verfügbarkeit*
  - *skalierbarer Durchsatz, kurze Latenzen*
  - *Einbeziehung von Virtualisierungstechnologien, Zugriff auf Sonderfunktionen*
  - *Überbrückung von Ausfällen*

Block-I/O über Ethernet – einmal anders gedacht

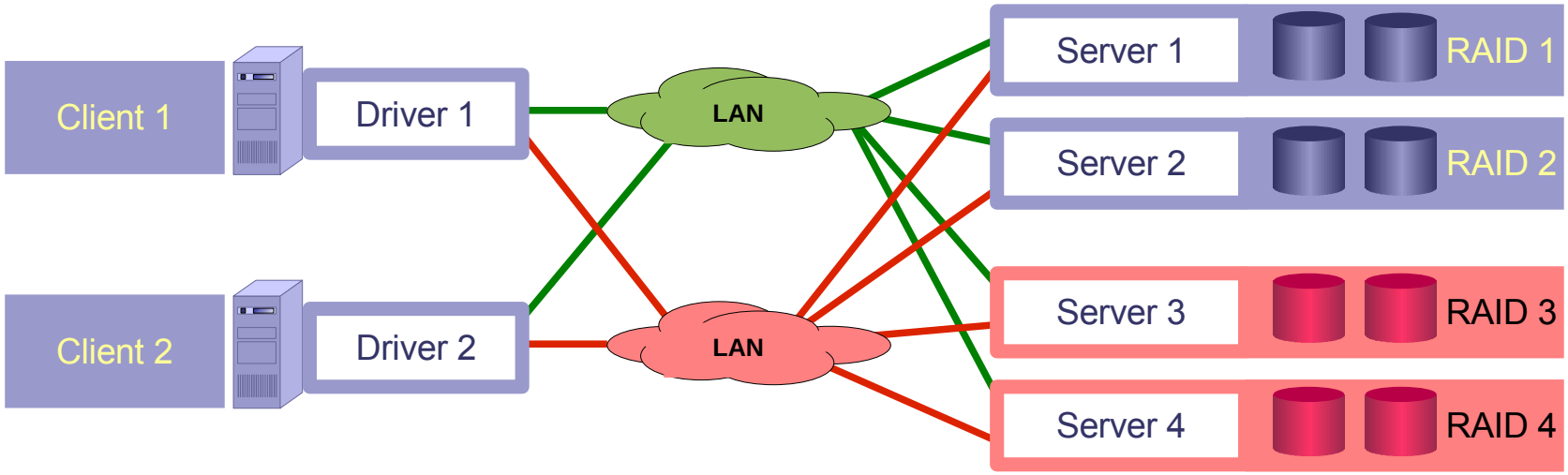
Worum geht es eigentlich bei Block-I/O über Netzwerke?

- *Für den **Programmierer**: Umsetzung/Wandlung der Schnittstellen*
  - *Netzwerke -> socket(), read(), write() stream-orientiert*
  - *Block-Device -> open(), lseek(), read(), write() request-orientiert*
  - *Block-I/O beschrieben durch Device, Offset, Bytes (Länge)*

- *Für den **Programmierer**: Verbergen von Komplexität*
  - *Umsetzung vieler Sockets (Pfade) auf ein Device*
  - *gleiche Programmiermodelle für direct attached und SAN-attached Storage*

Block-I/O über Ethernet – einmal anders gedacht

Für vernetzte Strukturen auch Netzwerkparadigmen anwenden



- *I/O-Requests senden read(), write(), ioctl()*
- *geeignete Kapselung*
- *Verbindungsauf- und Abbau, Überwachung*
- *Kanal-Multiplexing*

- *I/O-Requests verarbeiten read(), write(), ioctl()*
- *geeignete Kapselung*
- *Verbindungsauf- und Abbau, Überwachung*
- *Kanal-Multiplexing*

Block-I/O über Ethernet – einmal anders gedacht

Für vernetzte Strukturen auch Netzwerkparadigmen anwenden

Ist SCSI die Antwort?

- Geräteidentifikation und -Beschreibung einfacher möglich (Adresse, Port, Datenstrukturen)
- stärkere Berücksichtigung Netzwerk für Admin wünschenswert
- viele SCSI-Daten irrelevant, dafür sind viele interessante Funktionen kaum darstellbar
  - I/O-Requests senden
  - I/O-Requests verarbeiten
  - geeignete Kapselung
  - Verbindungsauf- und Abbau, Überwachung
  - Kanal-Multiplexing
- ohne SCSI keine Wandlung auf Low-Level-Protokoll erforderlich
- bestimmte SCSI-Mechanismen im Netz kontraproduktiv (z. B. Bus-Reset)
- reduzierter Kommunikationsaufwand möglich

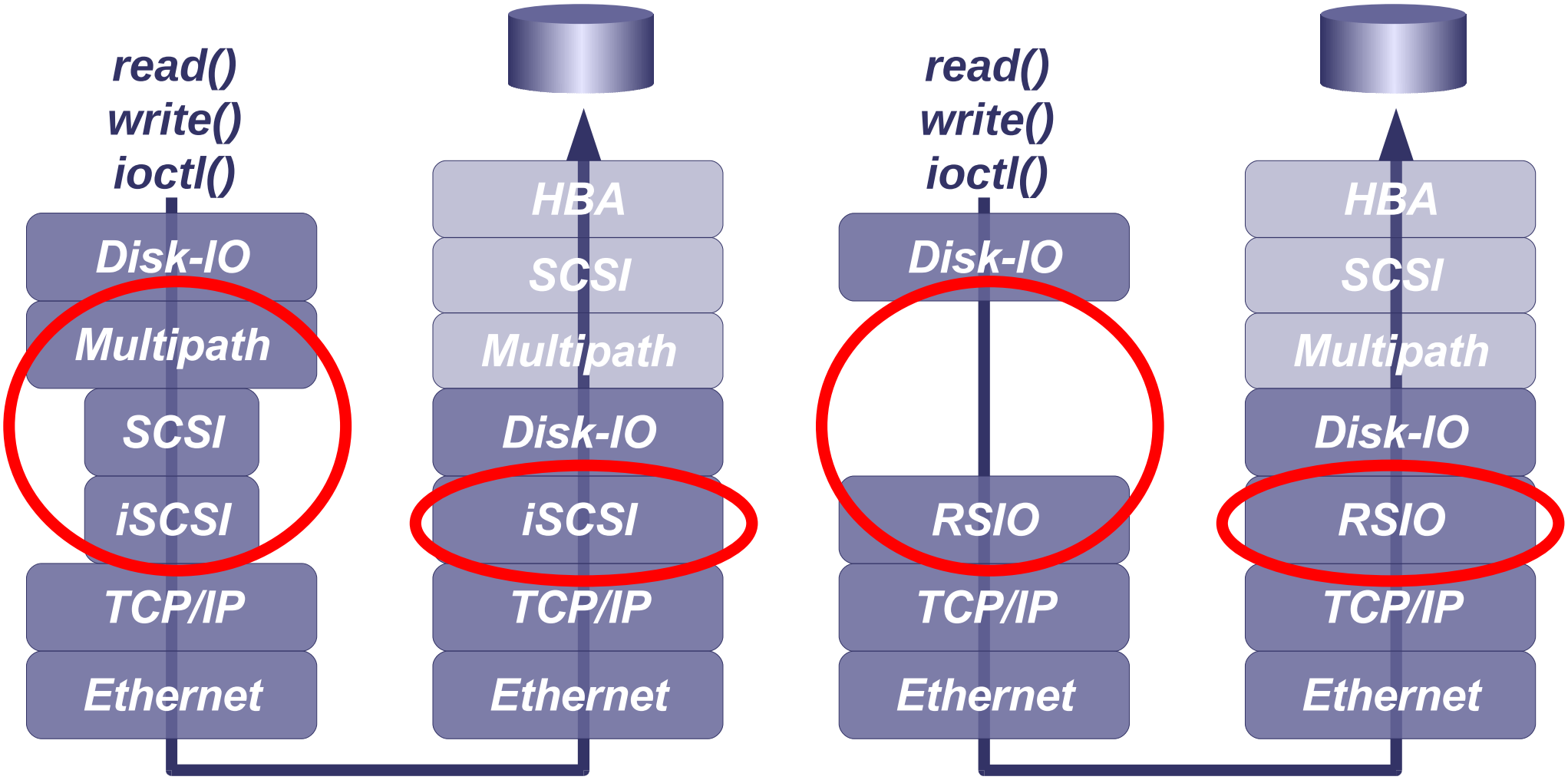
RSIO - Remote Storage I/O

Eckdaten des neuen Protokolls für LAN-attached (shared) Block Devices

- *neues, von OSL entwickeltes Protokoll*
- *direkter Transport aller relevanten IO-Aufrufe (read, write, ioctl)*
- *integriert Verbindungsaufbau, Überwachung, Path-Multiplexing, Trunking*
- *fähig zu Selbstkonfiguration und Error Recovery*
- *kann alle modernen Storage-Szenarien abbilden:*
  - *einfache Server und Clients, ggf. mit Multipathing*
  - *Cluster von Storage-Servern (Targets)*
  - *Cluster von Storage Clients (Initiators)*
  - *integrierte Cluster von Servern und Clients*
  - *Storage Server Farms*
  - *Cloud-Konzepte*
- *besondere Eignung für Kombination mit Speichervirtualisierung*
  - *eingängige Namen*
  - *fdisk (Partitionierung) auf Clientseite entfällt*
  - *On-Demand-Allokation und Online-Rekonfiguration*
  - *viele weitere Sonderfunktionen*
  - *ermöglicht Administration vom Client aus*

RSIO - Remote Storage I/O

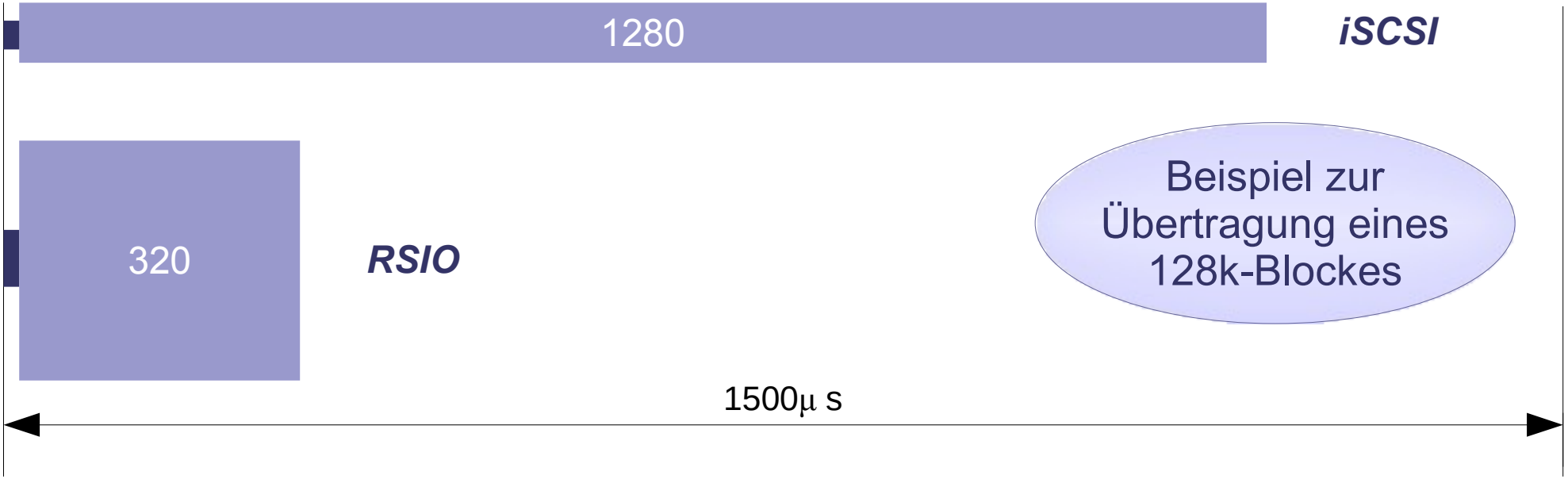
Vergleich der Protokollstacks



RSIO - Remote Storage I/O

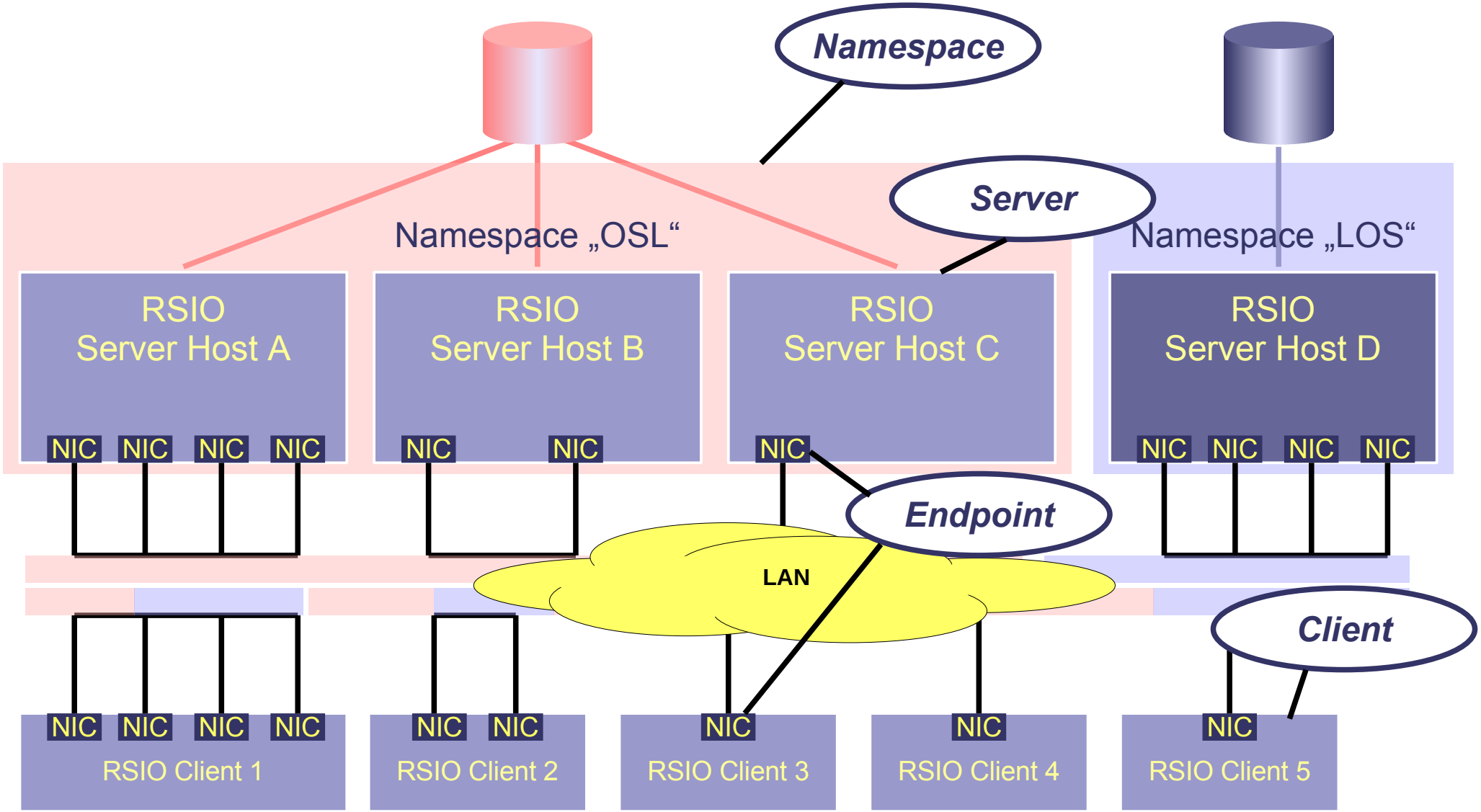
Einige Details zum Protokoll selbst

- *RSIO definiert eigene Frames*
  - *Unabhängigkeit von TCP o. ä.*
  - *ermöglicht erst Kanal-Multiplexing*
  - *ermöglicht Zusatzfunktionen wie Checksum / Encryption*
  - *Frames mit variabler Größe*
  - *Overhead per Frame nur 16 Bytes*



RSIO – Architektur im Überblick

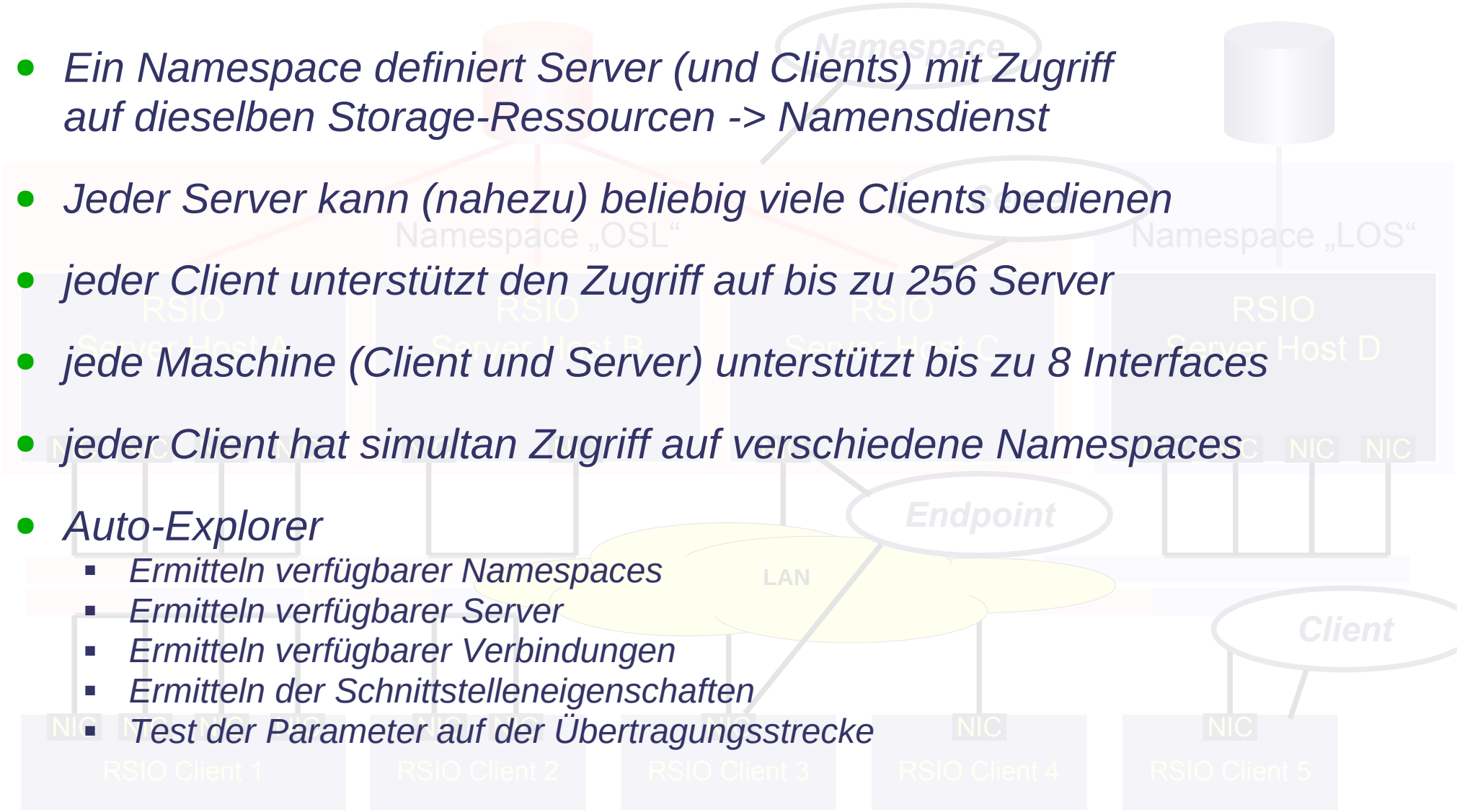
Klar gegliedertes und flexibles administratives Konzept



Parameter der RSIO-Architektur

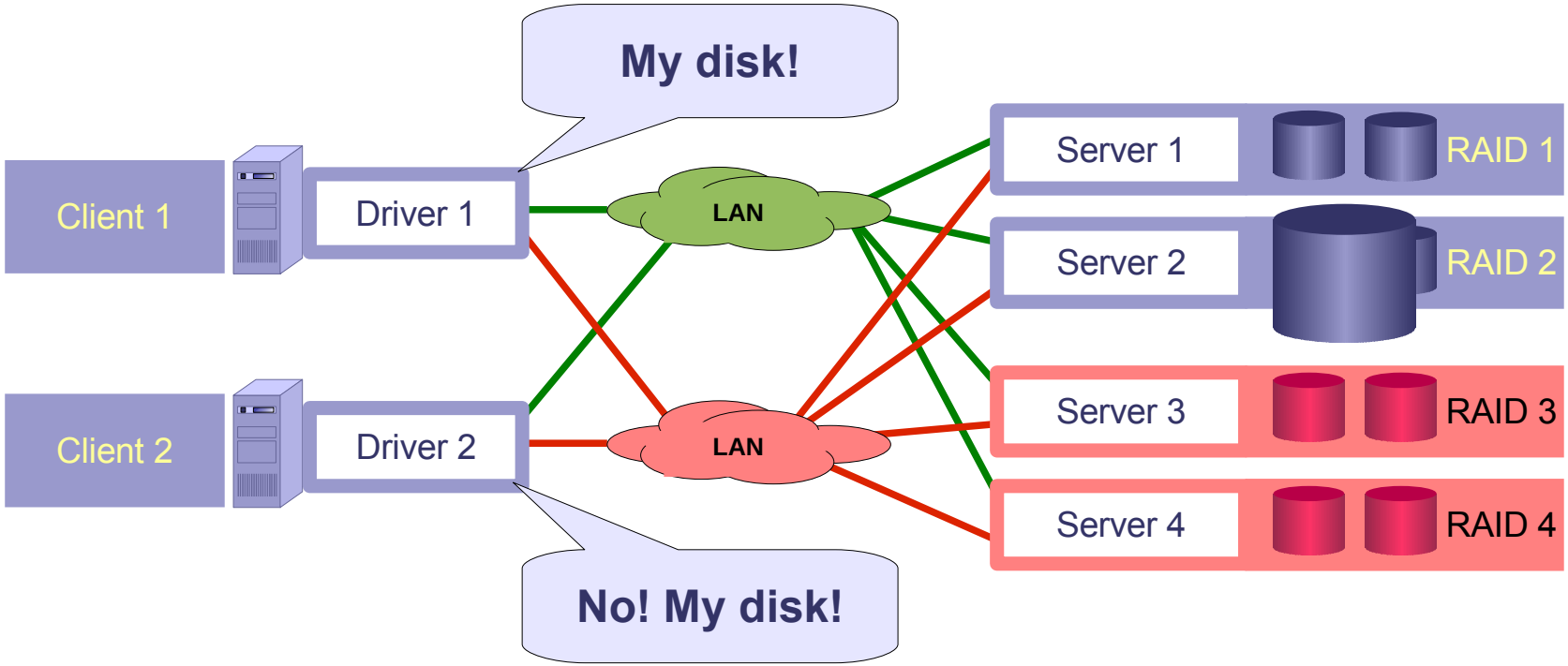
Flexible Client-Server-Implementierung

- *Ein Namespace definiert Server (und Clients) mit Zugriff auf dieselben Storage-Ressourcen -> Namensdienst*
- *Jeder Server 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*
- *jeder Client hat simultan Zugriff auf verschiedene Namespaces*
- *Auto-Explorer*
  - *Ermitteln verfügbarer Namespaces*
  - *Ermitteln verfügbarer Server*
  - *Ermitteln verfügbarer Verbindungen*
  - *Ermitteln der Schnittstelleneigenschaften*
  - *Test der Parameter auf der Übertragungsstrecke*



Warum RSIO nicht alles ist

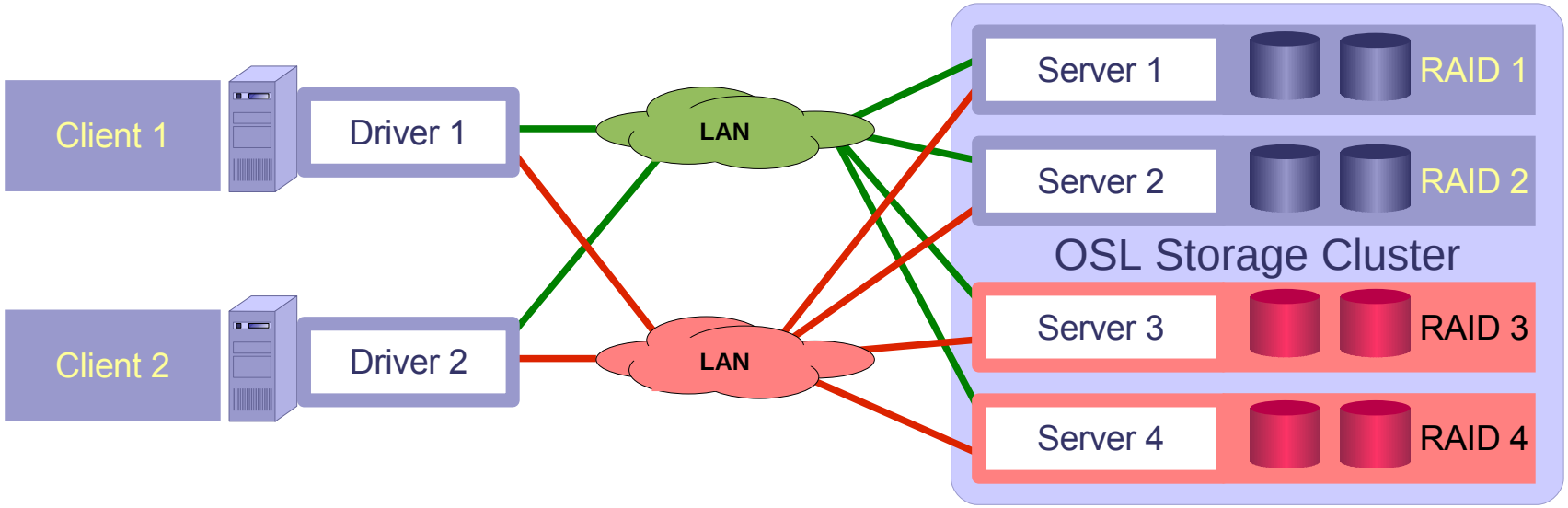
Storage-Transport allein ist heute meist zu wenig



- *Zusätzliche Funktionen sind wünschenswert, wie z. B.:*
  - *Zugriffsmanagement*
  - *Speichervirtualisierung*
  - *Clusterfähigkeit*
- *Die Kombination mit Cluster- und/oder Virtualisierungssoftware ist sinnvoll*
  - *Filesysteme, Volume-Manager, Clusterprodukte ...*

Wie OSL RSIO umgesetzt hat

Kombination mit dem OSL Storage Cluster



RSIO Client

- *Zugriff auf virtualisierten Speicher*
- *Zugriff auf Global Storage*
- *Nutzung des Global Namespace*
- *Multithreaded RSIO-Client*

OSL Storage Cluster + RSIO Server

- *Speichervirtualisierung*
- *Global Storage Pool*
- *Global Namespace*
- *Access Management*
- *Multithreaded RSIO Server*

Wie OSL RSIO umgesetzt hat

Vergleich der Darstellung von Ressourcen auf dem Client

So meldet sich eine iSCSI-Lun (“format” - Solaris)

```
29. c3t227d0 <DEFAULT cyl 1021 alt 2 hd 64 sec 32>
    /iscsi/disk@0000iqn.1986-03.com.sun%3A02%3A06df3360-bb85-ee33-bf59f2d03474f708.target-00001,0
30. c3t229d0 <DEFAULT cyl 1021 alt 2 hd 64 sec 32>
    /iscsi/disk@0001iqn.1986-03.com.sun%3A02%3A06df3360-bb85-ee33-bf59f2d03474f708.target-00001,0
```

Und so sieht der RSIO-Client Plattenressourcen

```
# rsconfig -q
000 osl
    clt: big-6
    srv: 000 big-5
         0  tvol1          disk        2097152 blocks of  512 bytes
         0  shadow         disk        2097152 blocks of  512 bytes
         0  ora_db         disk       10485760 blocks of  512 bytes
         0  postgres_db    disk       10485760 blocks of  512 bytes
         0  whole_zone     disk       41943040 blocks of  512 bytes
```

Und was ist mit der Performance?

Protokoll erlaubt hohe Performance und beeindruckende Skalierbarkeit

Server-Performance bei Cache Read / 8k

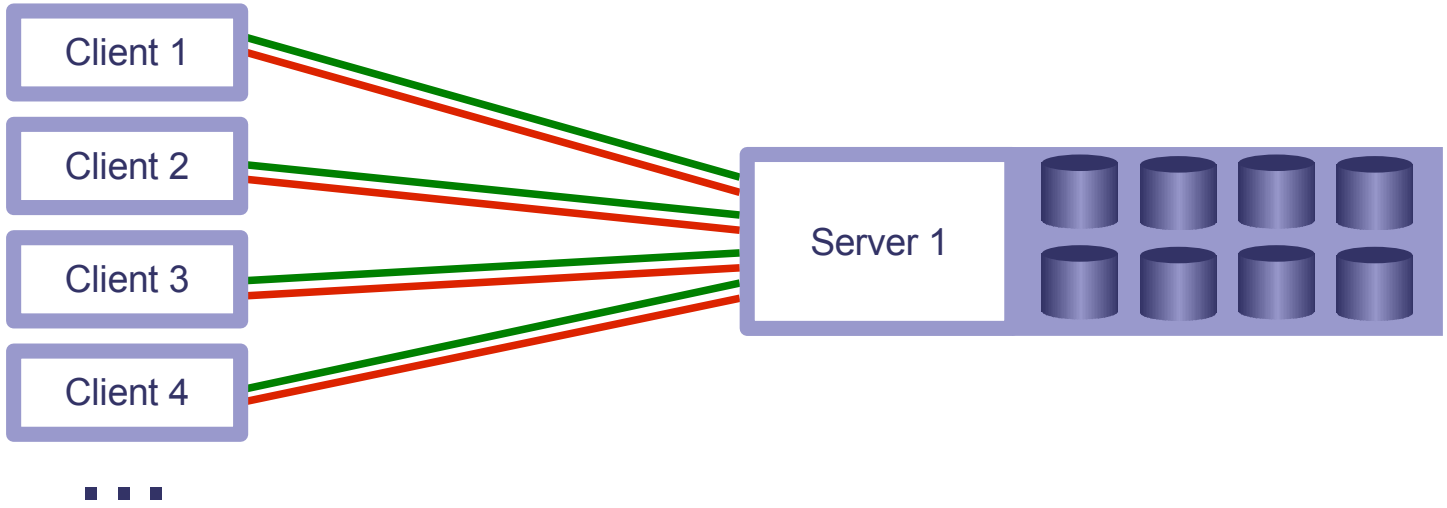
iSCSI	*10 Clients*	*100 Threads*	*7,6 Cores*	***31.000 IOPS***
iSCSI / comstar	*10 Clients*	*100 Threads*	*10,0 Cores*	***85.000 IOPS***
RSIO	*4 Clients*	*64 Threads*	*5,6 Cores*	***98.000 IOPS***
RSIO	*4 Clients*	*128 Threads*	*6,3 Cores*	***102.000 IOPS***

Client-Performance Throughput

RSIO	*1 x 1 GBit*	*ca. 0,5 Cores*	***> 110 MByte/s***
RSIO	*2 x 1 GBit*	*ca. 1,0 Cores*	***> 220 MByte/s***
RSIO	*4 x 1 GBit*	*ca. 2,0 Cores*	***> 440 MByte/s***
RSIO	*8 x 1 GBit*	*> 4,0 Cores*	***bis > 900 MByte/s***

Was kann ich mit RSIO aufbauen?

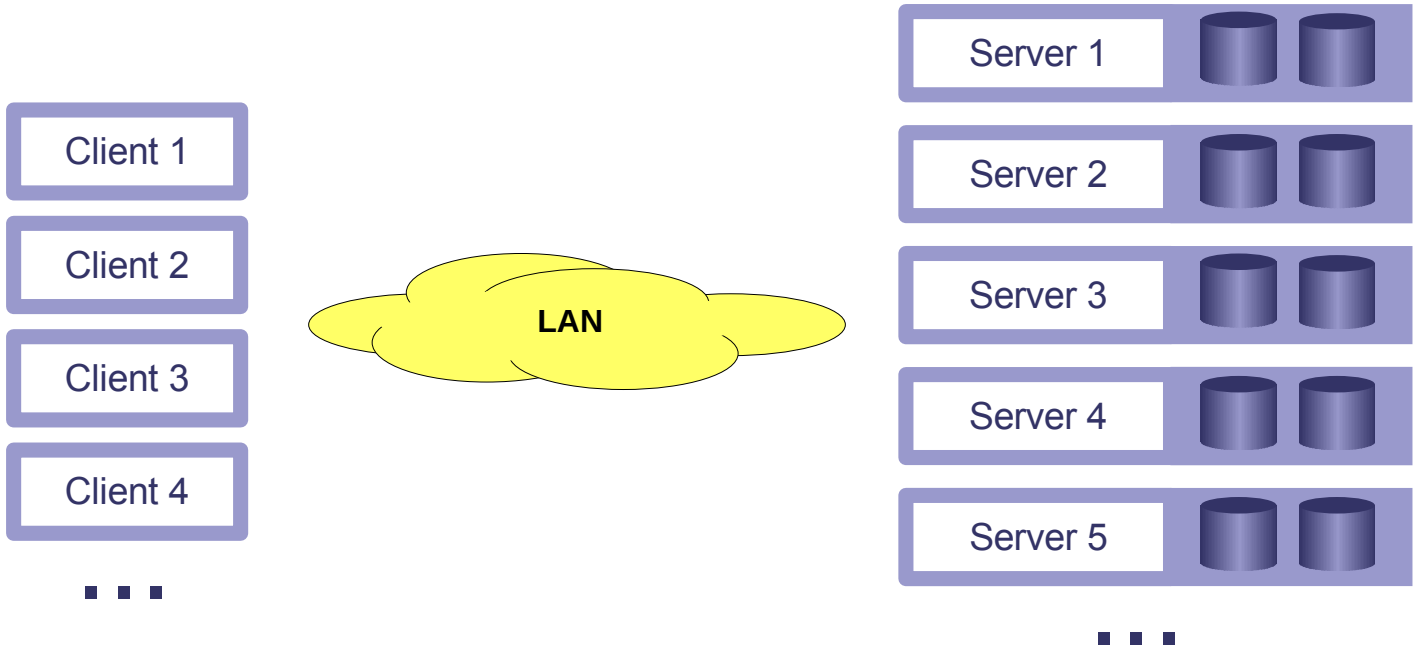
Beispiel 1: Einfacher Zugriff auf Plattenressourcen über LAN



- *Zugriff auf zentrales Speichersystem -> Global Pool, Global Namespace*
- *Virtualisierung und Cluster (HV) auf Clients einfach realisierbar*
- *Möglichkeit der Zentralisierung von Backup, Snapshots ...*
- *sehr preiswerte Speicheranbindung bei guter Performance*
- *redundante Datenpfade, Durchsatz je nach Bedarf skalierbar*

Was kann ich mit RSIO aufbauen?

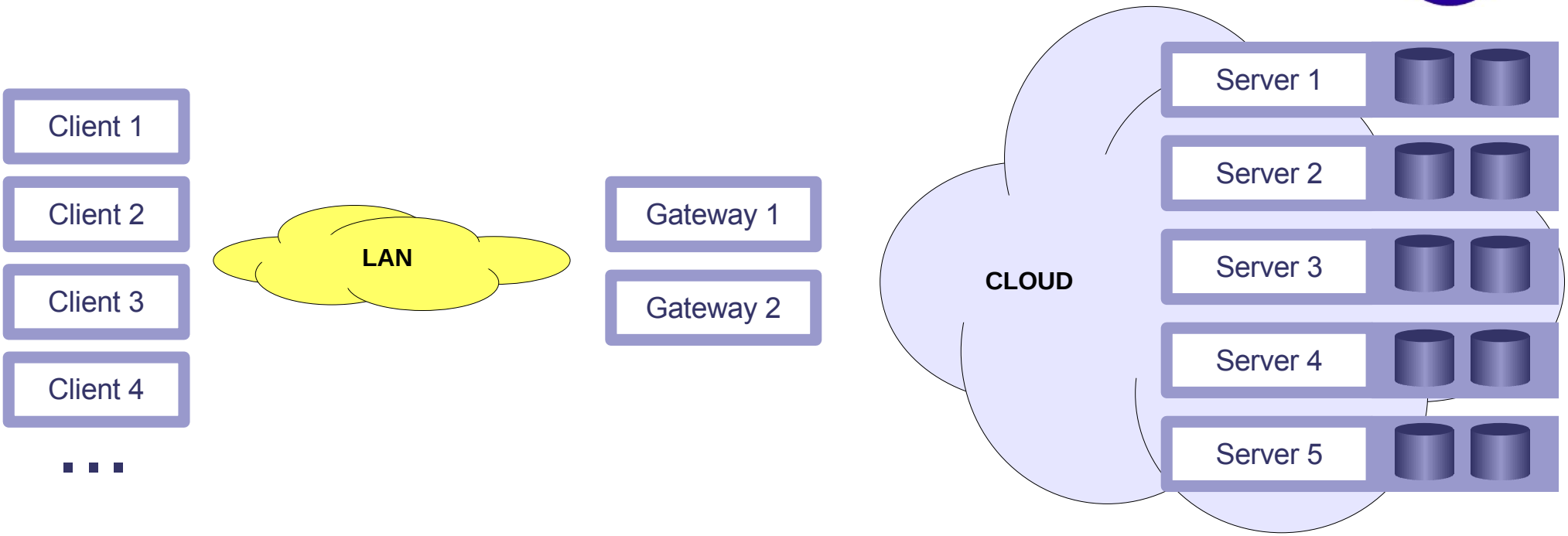
Beispiel 2: Storage Server Farm



- *Skalierung in Speichervolumen und Bandbreite*
- *jeder Server mit eigenem Namespace*
- *Storage-Kapazitäten “einsammeln” und so mit einfachen Mitteln große Kapazitäten und Bandbreiten darstellen*
- *nicht vergessen: Verfügbarkeit in der Server-Farm*

Was kann ich mit RSIO aufbauen?

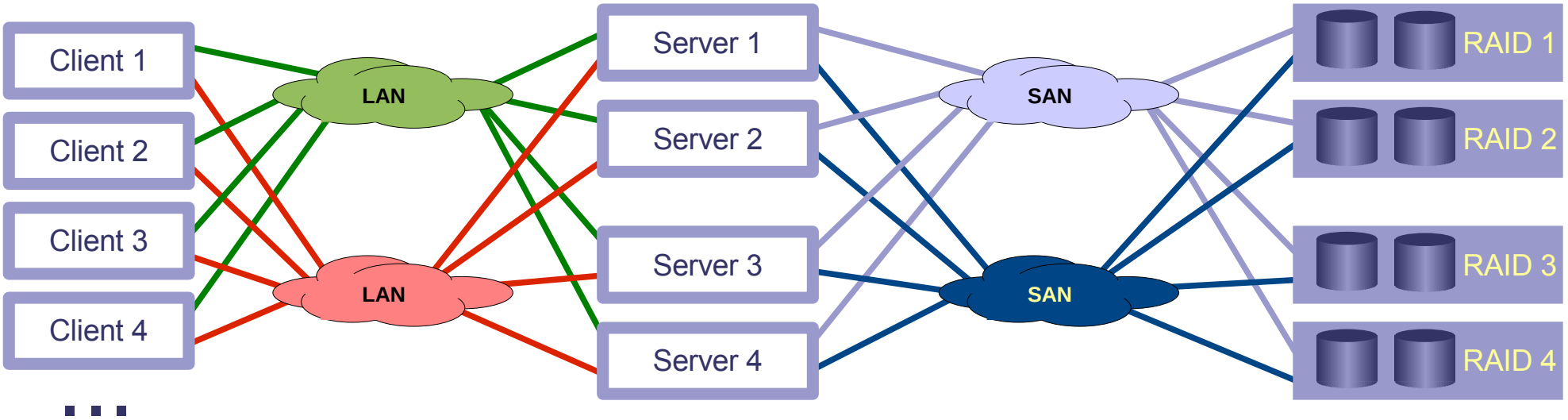
Beispiel 3: Szenario für Cloud Storage



- *Zugriff auf Speicherressourcen jenseits des LAN*
- *Mehrpfadigkeit, Bandbreite, Performance treten in den Hintergrund*
- *Gleichartige Administration wie bei RSIO im LAN*
- *Nutzt prinzipielle Routingfähigkeit von RSIO über IP*

Was kann ich mit RSIO aufbauen?

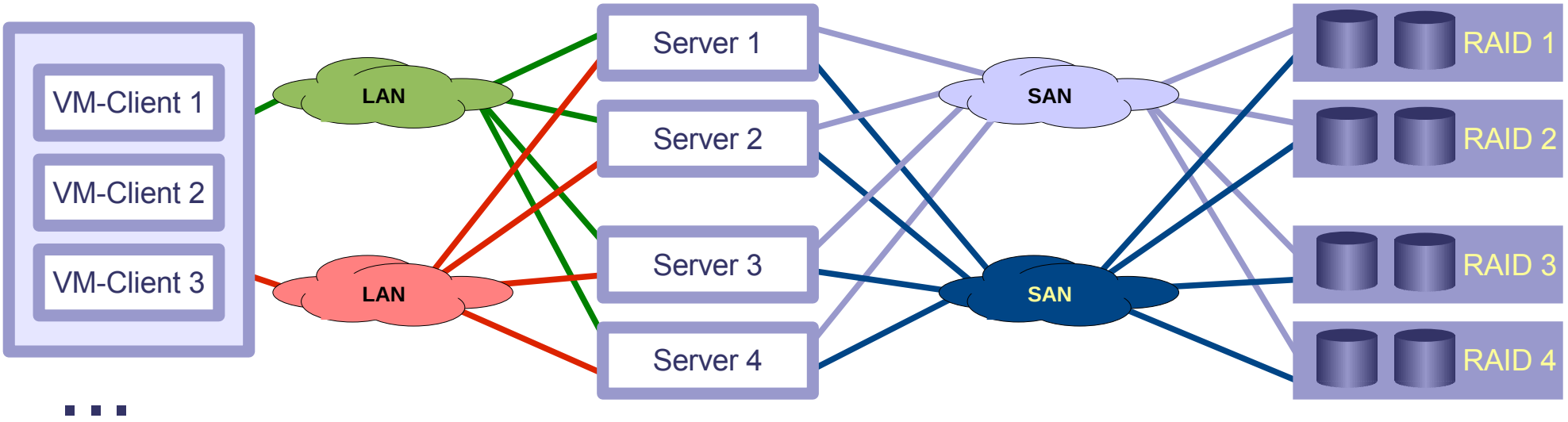
Beispiel 4: SAN-LAN-Konvergenz und geclusterte Storage-Server



- *SAN ins LAN hinein verlängern*
- *SAN-attached Server reichen “im Hintergrund” Storage-Ressourcen durch*
- *verbesserte Ausnutzung des SANs, Performance-Rightsizing*
- *hohe Performance, hohe Verfügbarkeit bei extrem niedrigen Kosten für RSIO-Clients*
- *weitere Verbesserung von Performance und Systemauslaustung möglich z. B. durch Nutzung freien Speichers als Cache*

Was kann ich mit RSIO aufbauen?

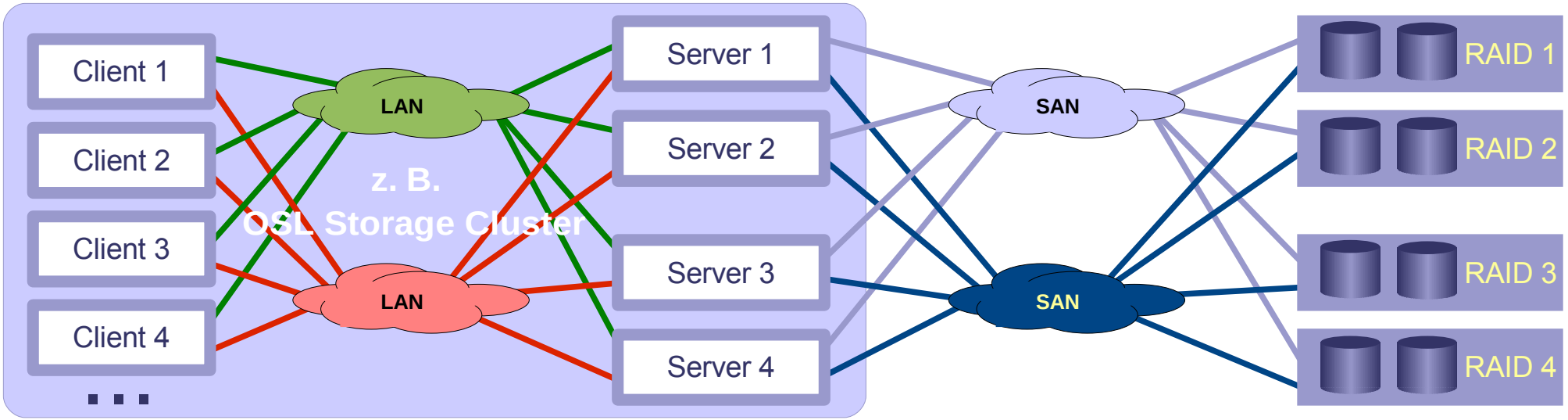
Beispiel 5: Einfacher Zugriff auf die gesamte Storage-Welt aus der VM



- *SAN reicht über die IP-Interfaces bis in die Virtuellen Maschinen hinein*
- *beliebige Devices erreichbar, daneben Selbstkonfiguration ...*
- *Aggregation für 10GbE, Nutzung von VMDq möglich*
- *enorme Vereinfachung*

Was kann ich mit RSIO aufbauen?

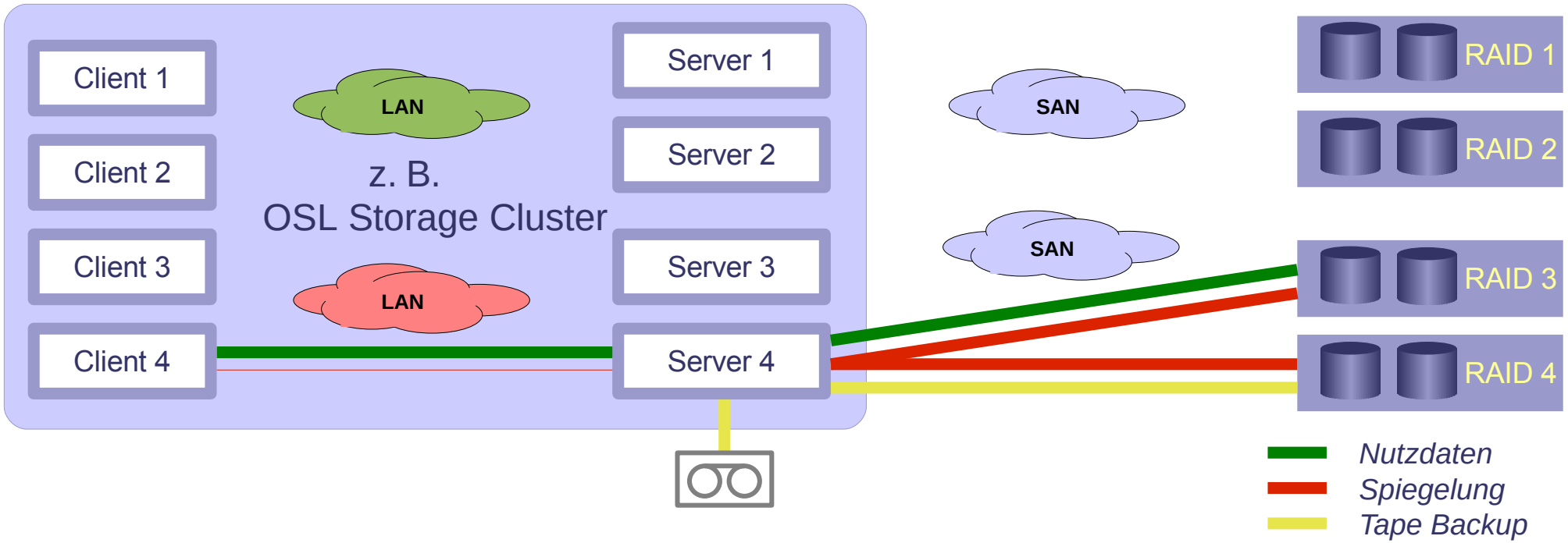
Beispiel 6: Server und Clients in einem Cluster integrieren



- *alle Funktionen wie in Beispiel 3 (SAN-LAN-Integration)*
- *zusätzlich weitere Storage-Management-Funktionen:*
  - *Storage-Allokation, -Management vom Client aus*
  - *applikationsbezogene Speichervirtualisierung vollumfänglich auf Client nutzbar*
  - *Möglichkeit der transparenten Nutzung von Datenspiegelung, Backup to Disk etc.*
- *Verschmelzung von Client und Server zu einer Einheit*
- *run applications everywhere*

Der Vorteil eines universellen Protokolls

Beispiel 7: Hochgeschwindigkeits-Backup für LAN-attached Blockdevices



- *über das LAN laufen nur Nutzdaten und die Steueranweisungen*
- *LAN-less Backup:*
  - *hohe Geschwindigkeit*
  - *vollständige Steuerung vom Client aus*
  - *applikationsbezogene Aktionen*

Zusammenfassung
Protokolle für Speichernetzwerke im kommerziellen RZ

- *iSCSI*
  - *Übertragung des SCSI-Protokolls via TCP(IP)*
  - *standardisiert, viele Plattformen*
  - *routingfähig*
  - *keine besondere Hardware erforderlich*
  - *Limitationen hinsichtlich Multipathing, Skalierbarkeit, Einsatzszenarien und Benutzerfreundlichkeit*
- *Fibre Channel over IP*
  - *Tunneln von Fibre Channel über IP-Netze, kann FC-SANs koppeln*
  - *ergibt nur Sinn in Verbindung mit existierender FC-Infrastruktur*
- *FCoE*
  - *Kapselung von Fibre Channel direkt in Ethernet-Frames unter Verzicht auf TCP/IP*
  - *damit leichte Performance-Vorteile, aber: nicht routingfähig*
  - *erfordert erhebliche Erweiterungen des Ethernet-Standards (Flow Control, Data Center Bridging, Lossless Ethernet)*
  - *erfordert damit zugleich neue Hardware (HBAs und LAN-Komponenten inkl. Switches)*
  - *bei Converged Networks erheblicher Zuwachs an Komplexität*
- *ATA over Ethernet, HyperSCSI*
  - *ähnliche Konzeption wie FCoE, aber auf Standard-Ethernet*
  - *kein IP, nicht routingfähig, nicht virtualisiert (“just raw disk”)*
- *Fibre Channel*
  - *Das Original, designed für Block-I/O, zumeist Transport von SCSI in FC-Frames*
  - *hoher Durchsatz, geringe Latenz, Kanaleigenschaften aber eben kein Ethernet/LAN, was aus technischer Sicht natürlich ein Vorteil ist*

Zusammenfassung

RSIO - Data Center Block I/O over Ethernet

- *direkter Transport aller relevanten IO-Aufrufe (read, write, ioctl)*
- *eigene Frames -> für verschiedene Träger geeignet*
  Implementierung über TCP(IP) natürlich routingfähig
- *integriert Verbindungsaufbau, Überwachung, Path-Multiplexing, Trunking, Selbstkonfiguration, Error Recovery*
- *Im LAN bereits mit heutiger Technik beeindruckende Skalierbarkeit und Durchsätze*
- *administorfreundliche Gliederung: Namespace -> Server -> Client*
- *damit sind vielfältige Storage-Szenarien abbildbar:*
  - *einfache Server und Clients*
  - *Cluster von Storage-Servern (Targets) und Cluster von Storage Clients (Initiators)*
  - *Storage Server Farms und Cloud-Konzepte*
- *besondere Eignung für Kombination mit Speichervirtualisierung*
- *”Huckepack” -Transport für andere Dienste (z. B. HV, Backup ...- alles über ein Kabel)*
- *Clients können mit Servern zu einem intelligenten Cluster verschmelzen, z. B. für:*
  - *automatisierte, applikationsbezogene Speicherverwaltung vom Client aus*
  - *Failover-Cluster*

RSIO - Storage Networking
der nächsten Generation

Danke für Ihre Aufmerksamkeit!

Wir freuen uns auf weitere Gespräche

Stand 26

Bert Miemietz

**OSL Gesellschaft für
offene Systemlösungen mbH**