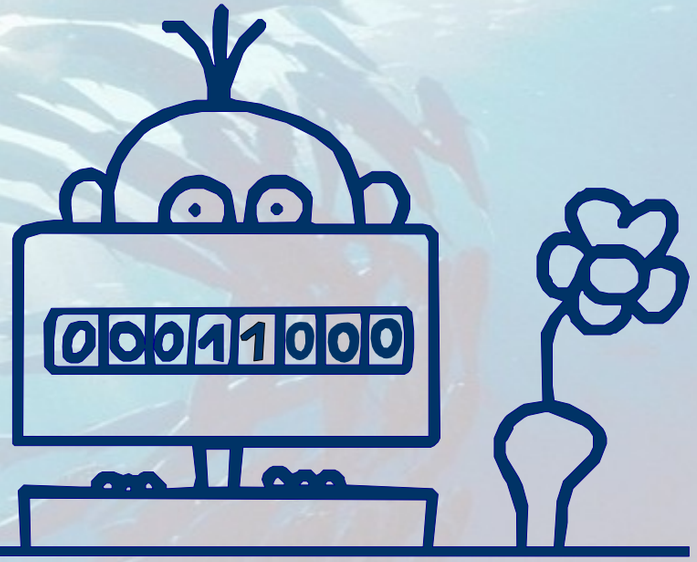
Ein hochintegrierter Virtualisierungsstack unter Linux

OSL Storage Cluster und OSL Unified Virtualisation Environment



24. Kieler Open Source und Linux Tage

Kiel • 18. September 2026

bert.miemietz@osl.eu

Einige Gedanken zu digitaler Autonomie und digitaler Souveränität

Autonomie und Souveränität

Versuch einer orientierenden Begriffsklärung

- Sonst gern ignoriert, sind beide Begriffe in Krisensituationen in aller Munde

- **Autonomie** = Selbstbestimmung, Entscheidungsfreiheit von Akteuren in einem System
  - anwendbar auf Individuen, Gruppen, Unternehmen usw.

- **Souveränität** = Autonomie von Staaten
  - vorrangige Verwendung des Begriffes als Attribut von staatlichen Akteuren
  - grundsätzliche Herrschaftsgewalt eines Staates über sein Territorium und seine Bevölkerung
  - Unabhängigkeit gegenüber anderen Staaten

- Beide Begriffe beinhalten also die Aspekte Selbstbestimmung und Unabhängigkeit

Autonomie und Souveränität in der digitalen Welt

Konkretisierung

Digitale Autonomie

Fähigkeit, für das eigene Handeln kritische Daten, digitale Ressourcen und Prozesse selbstbestimmt zu gestalten, zu kontrollieren und dies nötigenfalls unabhängig von einzelnen externen Akteuren weiterzuführen.

Aspekte:
- Technologische Autonomie
- Datenautonomie
- Operative Autonomie
- Entscheidungsautonomie
- rechtliche und strategische Autonomie

Völlige Unabhängigkeit ist illusorisch, es geht vielmehr um Optionen und Spielräume.
Der digitalen Souveränität ist dabei eine nationale Komponente eigen.

Gefahren für digitale Autonomie und Souveränität

Eine Auswahl

- Technologische Abhängigkeiten:
  Zentrale Hard- oder Software, Cloud-Infrastruktur, Betriebssysteme,Plattformen unter der Kontrolle weniger Akteure
- Datenkonzentration und Plattformmacht
  Wer große Datenmengen kontrolliert, kann den Zugriff, Informationsflüsse, Geschäftsmodelle, Entscheidungen beeinflussen
- Datenkonzentration + KI = Gigantischer Vorteil in jeder Form von Wettbewerb
  Schafft neue Abhängigkeiten
- Fehlende Standardisierung oder Abhängigkeit von proprietären Standards
  Fehlende Interoperabilität kann Akteure praktisch an einen Anbieter oder ein Ökosystem binden
- Cyberangriffe und digitale Erpressung
  Ransomware, Sabotage, Verlust kritischer Daten beeinträchtigen die Handlungsfähigkeit
- Kompetenzverlust
  Verlieren Akteure das Wissen, ihre Systeme selbst zu verstehen, zu warten, zu wechseln, entsteht faktische Abhängigkeit
- Regulatorische und geopolitische Abhängigkeiten
  Exportbeschränkungen, Sanktionen, Datenschutzvorgaben oder Abhängigkeiten von ausländischer Infrastruktur

Vielleicht die größten Gefahren: Disruption und Zeitfaktor

Disruption × Zeitdruck × Abhängigkeit = sinkende Autonomie

- Technologie verbreitet sich schneller als die Nutzer bzw. Institutionen lernen können
- Disruption verändert Wirtschaftsarchitekturen mit der Folge gravierender Kapital- und Machtverschiebungen und entsprechenden Einbußen an Autonomie bzw. Souveränität, oft entstehen gewaltige Profite aus der Substanz der “Opfer der Disruption”
- Schnelle Kumulation von Abhängigkeiten resultiert später in großen Wechselkosten
- Daten und Wissen fließen ab, Positionen wirtschaftlicher Stärke gehen verloren
- Zeitdruck beeinflußt Entscheidungen, deren Reversibilität nimmt ab
- “Wer kontrolliert welche Technologie“ muß in der Dynamik betrachtet werden

Wege zu mehr digitaler Autonomie und Souveränität

Eine Auswahl

- Datenhoheit (Zugriff und Schutz)
- Offene Schnittstellen & Standards, Portabilität
- Anbieterdiversifikation
- Eigenes Know-how
- realistische Exit-Strategien

} gilt für Cloud und Eigenbetrieb

Daneben spielen eine Rolle:

- Globaler Wettbewerb und Innovationsgeschwindigkeit
  Bemühungen um digitale Souveränität sollten sich nicht nur darauf fokussieren, fremde Technologien zu ersetzen.
  Besser ist es, selbst fortschrittliche Technologien entwickeln, die andere nicht ohne Weiteres ersetzen oder kopieren können.
- Wirtschaftlichkeits- und Kostenaspekte

Open Source als *ein* Baustein digitaler Autonomie

Zweimal hingeschaut

Licht:

- Geringerer Vendor Lock-in, große Community
- Kontrolle über den Lebenszyklus
- Quellcodezugriff / Anpassbarkeit
- keine Lizenkosten

Schatten:

- Prämissen großer Projekte stimmen u. U. nicht mit den eigenen überein
- Weniger „Vendor Lock-in“ kann sich als „anderer Lock-in“, z. B. hinsichtlich Plattform oder Technologie herausstellen
- Eigene Anpassungen bedeuten langfristige Verpflichtungen
- Wettbewerb von Softwareanbietern wird ggf. zum Dienstleisterwettbewerb

Provokation: Souverän ist nicht unbedingt, wer den Quellcode besitzt. Souverän ist, wer Handlungsalternativen hat.

Bei Infrastrukturlösungen zählen Funktionalität, eine offene, modulare Architektur, stabile Standards, Austauschbarkeit von Komponenten und Portabilität.

Heutige Herausforderungen für das RZ im Eigenbetrieb

Die Zwickmühle für das RZ im Eigenbetrieb

Klassische Hardware und Applikationen aber cloudtypische Erwartungen

- Zumeist eher “klassische” (Legacy-) Applikationen (Datenbanken, Verfahren)
  - Gänzlich anderes Design als cloud-native Applikationen mit hochautomatisierten, hochskalierbaren Microservices
  - Einbeziehung von Web-Technologien macht die Bilanz für den On-Prem-Stack noch ungünstiger
  - Oft wildes Sammelsurium von Applikationen, die “nur für sich allein gedacht sind”
- Commodity-Hardware statt Spezial-Hardware mit HW-Offloading usw.
- “Economy of Scale” sehr weit unter der der Hyperscaler
- Klassisches, auf einzelne Systeme orientiertes Management scheint aus der Zeit gefallen
  - Es ist oft egal, ob die Bereitstellung eines Systems 5 oder 15 Minuten dauert, nicht aber, wenn es 5 Tage sind
  - Es ist mitunter egal, ob das System 7x24h läuft, aber es muss verfügbar sein, wenn es gebraucht wird (z. B. arbeitstäglich 04-22 Uhr)
  - Silos im klassischen RZ benötigen eine große Mannschaft für den Betrieb, dazu kommen Reibungsverluste zwischen den Bereichen
  - Cloud-Frameworks im klassischen Klein-RZ bedeuten oft: Automatisierung wird aufwendiger als manuelle Administration
- Viele unvorteilhafte Abhängigkeiten führen zum Gegenteil von Autonomie

Die Zwickmühle für das RZ im Eigenbetrieb

Klassische Hardware und Applikationen aber cloudtypische Erwartungen

- Zumeist eher “klassische” (Legacy-) Applikationen (Datenbanken, Verfahren)
  - Gänzlich anderes Design als cloud-native Applikationen mit hochautomatisierten, hochskalierbaren Microservices
  - [illegible]
  - Oft wildes Sammelsurium von Applikationen, die “nur für sich allein gedacht sind”
- Commodity-Hardware statt Spezial-Hardware mit HW-Offloading usw.
- “Economy of Scale” sehr weit unter der der Hyperscaler
- Klassisches, auf einzelne Systeme orientiertes Management scheint aus der Zeit gefallen
  - Es ist oft egal, ob die Bereitstellung eines Systems 5 oder 15 Minuten dauert, nicht aber, wenn es 5 Tage sind
  - Es ist mitunter egal, ob das System 7x24h läuft, aber es muss verfügbar sein, wenn es gebraucht wird (z. B. arbeitstäglich 04-22 Uhr)
  - Silos im klassischen RZ benötigen eine große Mannschaft für den Betrieb, dazu kommen Reibungsverluste zwischen den Bereichen
  - Cloud-Frameworks im klassischen Klein-RZ bedeuten oft: Automatisierung wird aufwendiger als manuelle Administration
- Viele unvorteilhafte Abhängigkeiten führen zum Gegenteil von Autonomie

Was bei Hyperscalern
durch Spezial-Hardware, Automatisierung, Skalierungseffekte überzeugt,
gerät im kleinen Maßstab oft teuer, komplex und fehleranfällig.

Legacy-Applikationen mit klassischen Betriebsmodellen fressen mich auf!

Was erwarten kleine und mittlere RZ-Betreiber ?

Kleine Rechenzentren im Eigenbetrieb – ein bis mehrere hundert Server*

Infrastrukturlösungen sollen liefern:

- Vereinfachung durch Integration
- Agilität und Betriebssicherheit
- Auffangen heterogener Applikationen
- Öffnung für Modernisierung
- Autonomie / Resilienz / Flexibilität vor Optimierung und Effizienz

* natürlich geht auch mehr ...

Was macht nun OSL?

Autonomie, Kreativität und Kooperation als Programm

Ziele hinter dem Projekt “Infrastruktursoftware Made in Gemany”

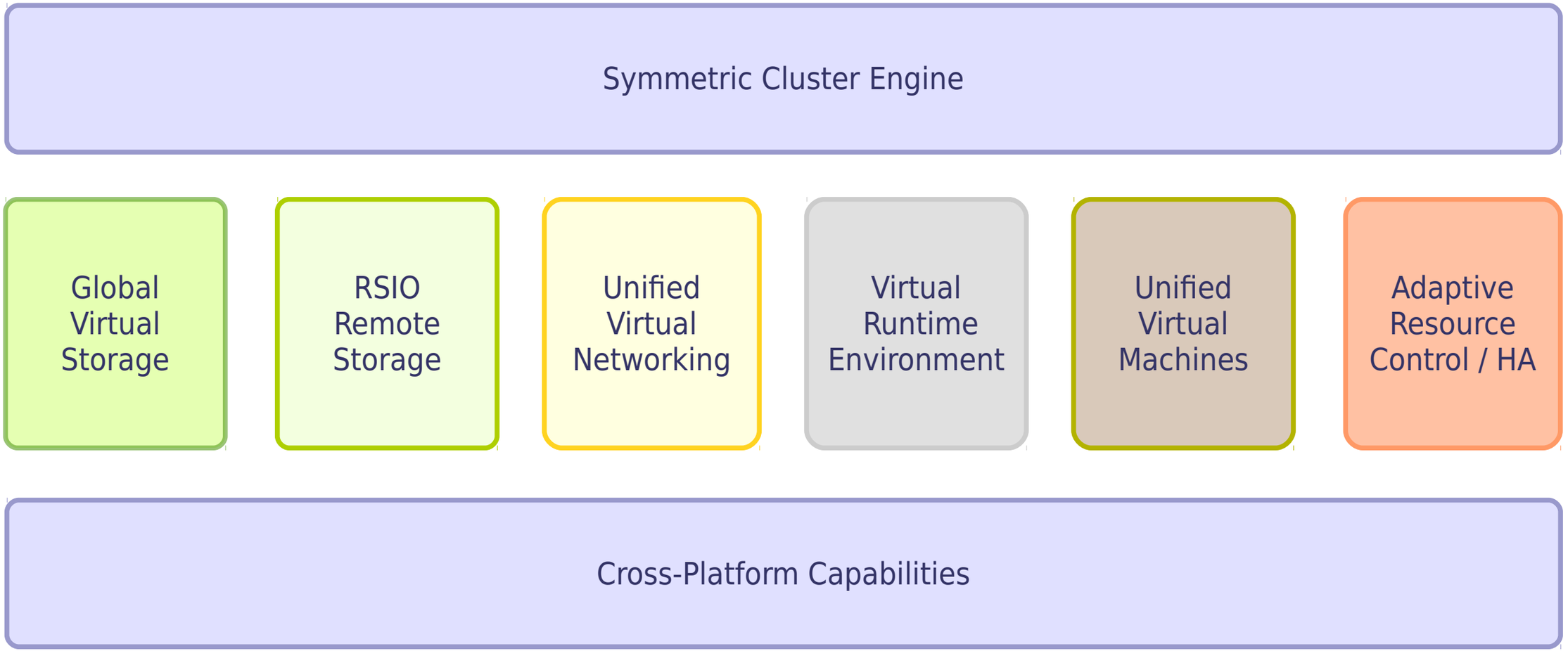
- Entwicklung von Infrastruktursoftware vom Treiber bis zur Oberfläche

Schwerpunkte:	
Virtual Storage Hochverfügbarkeit	komplett eigene Technologie
Virtual Network Virtual Server Hyperkonvergente Infrastruktur	Standardtechnologie + OSL-Erweiterungen

- Eigene Technologie als Grundlage für Autonomie / Umsetzbarkeit eigener Ideen
- Helle Köpfe (Admins, Systemarchitekten, Programmierer) zusammenbringen
- Unterstützung für Kunden weniger als Dienstleistung, sondern vielmehr durch Software
- Fokus auf einfache Bedienung, Robustheit und langfristig stabile RZ-Prozesse

OSL-Infrastruktursoftware

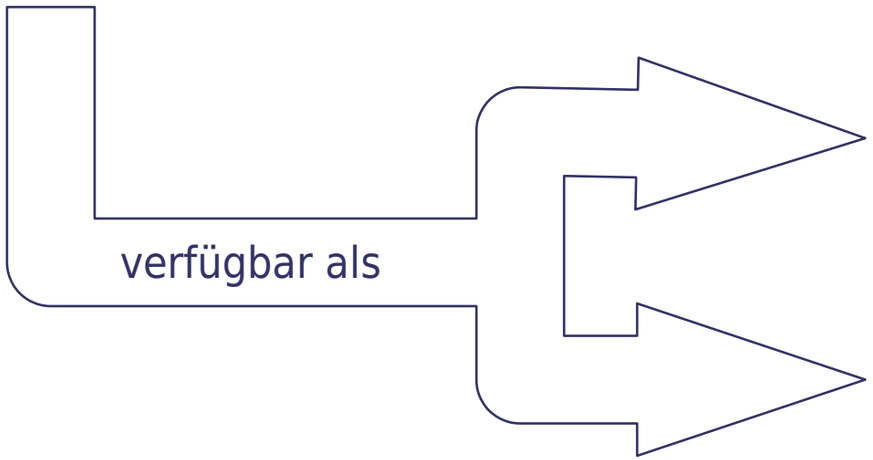
Ziel: Softwaredefiniertes Rechenzentrum mit Hardwareabstraktion und Cross-Plattform-Fähigkeiten



Unsere Kern-Technologien

High-Tech nach dem Prinzip „Schlicht und einfach statt großartig und komplex”

- 1 - Hostbasierte, skalierbare und clusterfähige Speichervirtualisierung
- 2 - Ausgereifte, mit der Speichervirtualisierung integrierte Clusterengine
- 3 - RSIO: Data-Center-Block-I/O über Ethernet / Infiniband
- 4 - Netzwerk-Virtualisierungsframework



Einzelbausteine
(z. B. Simple RSIO)

integrierte Pakete “aus einem Guß”
(z. B. OSL SC und OSL UVE)

Die Bausteine im Komplettpaket

OSL Storage Cluster

OSL Storage Cluster

Der storagezentrierte Cluster (um den Shared Storage herum)

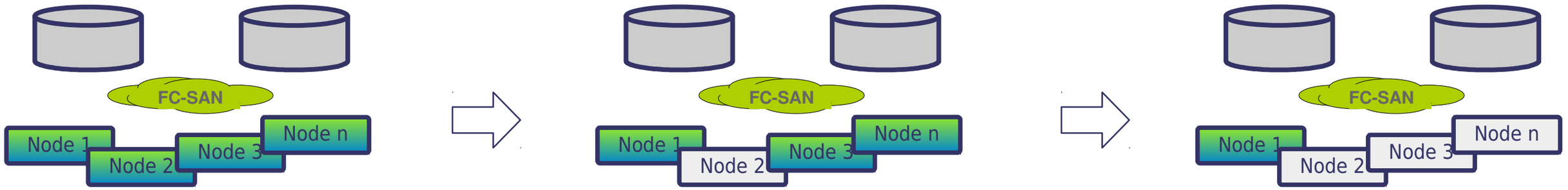
Wenn Ihre Infrastruktur z. B. so aussieht:

(schematische Darstellung, Ethernet nicht eingezeichnet)

- Sie haben eben erst in zentrale RAID-Systeme und eine FC-Infrastruktur investiert
- Sie wollen / müssen / können eine mehrgliedrige Administration beibehalten
  (Storage, SAN, Netz, Hosts, Virtualisierung ...)
- Spezifische Anforderungen zwingen Sie zu einem konservativen Setup
- Es soll “einfach nur der Hypervisor gewechselt werden”
- Sie wollen (auch) OLTP-Systeme auf Bare-Metal-Knoten betreiben

OSL Storage Cluster

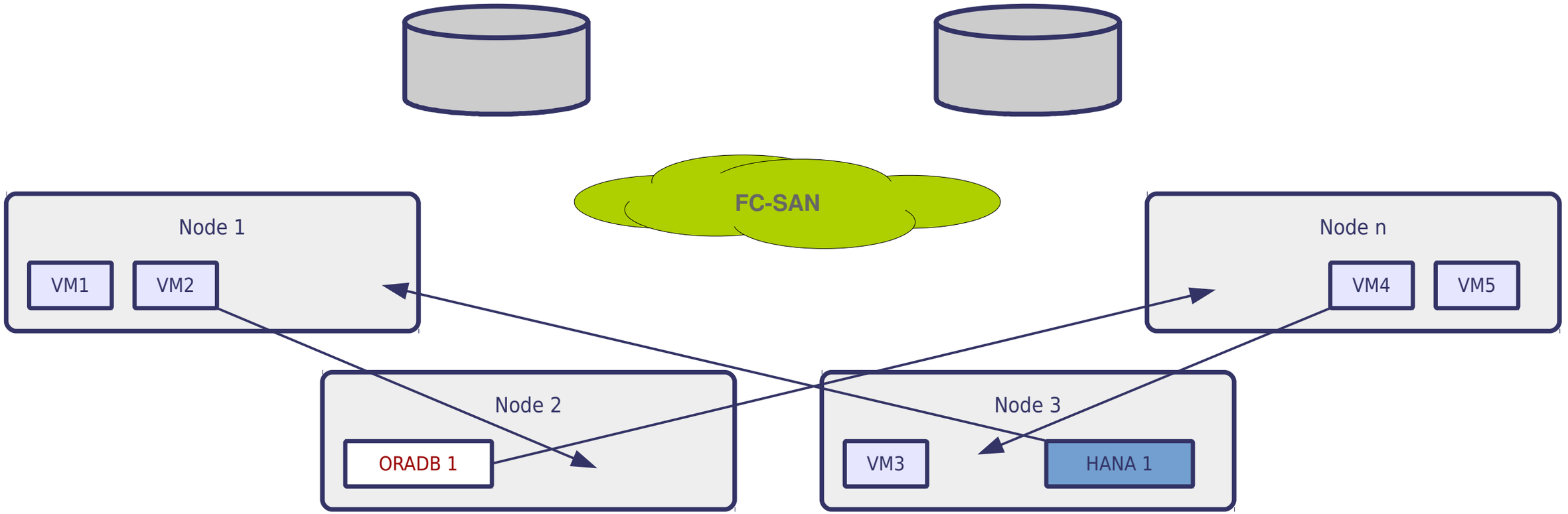
Einfach „nur den Hypervisor wechseln“?



- Physische Infrastruktur ähnlich → keine oder nur geringe Änderungen erforderlich
- “Alte” und “neue” Welt können koexistieren, auch auf Dauer
- Schrittweiser Umstieg “auf leisen Sohlen”

OSL Storage Cluster

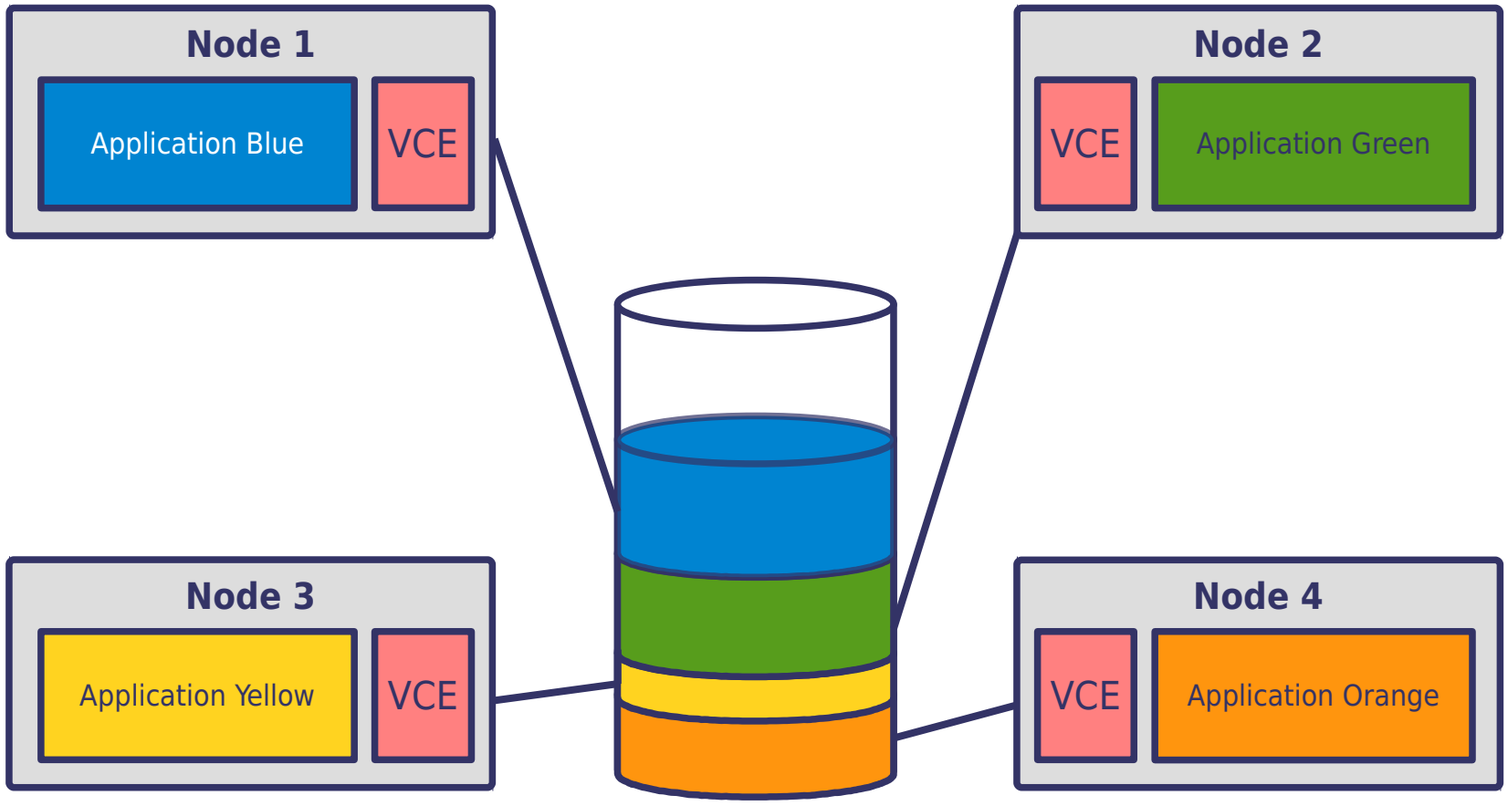
Cluster-/Admin-Framework, Hochverfügbarkeit und mehr



- Legacy- und Bare-Metal-Applikationen können mit VM-Workloads gemischt werden
- Cross-Platform-Fähigkeiten (verschiedene Linux-/Solaris-Versionen und CPU-Architekturen)

Das steckt dahinter: Clusterfähige Speichervirtualisierung

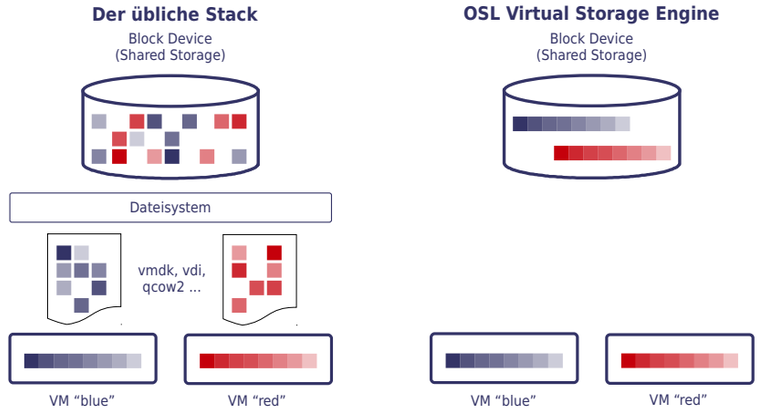
Die Vorteile



- Es ist nur Software
  → langlebig, kein Hardware-Verschleiß
  → keine zusätzlichen In-Band-Geräte
  damit minimierte Latenzen
- stark vereinfachte Administration
  (RAID, SAN, Nodes)
- Global Pool / Global Namespace
- kein Verschnitt
- von vornherein clusterfähig
- Load-Balancing zwischen Servern
- Block-Device-Paradigma
- keine Bottlenecks im SAN
- IO-Peaks / Disk-Overload vermeiden

Das steckt dahinter: Blockorientierte Speichervirtualisierung

Abgrenzung von filesystemorientierten Konzepten (vereinfacht)

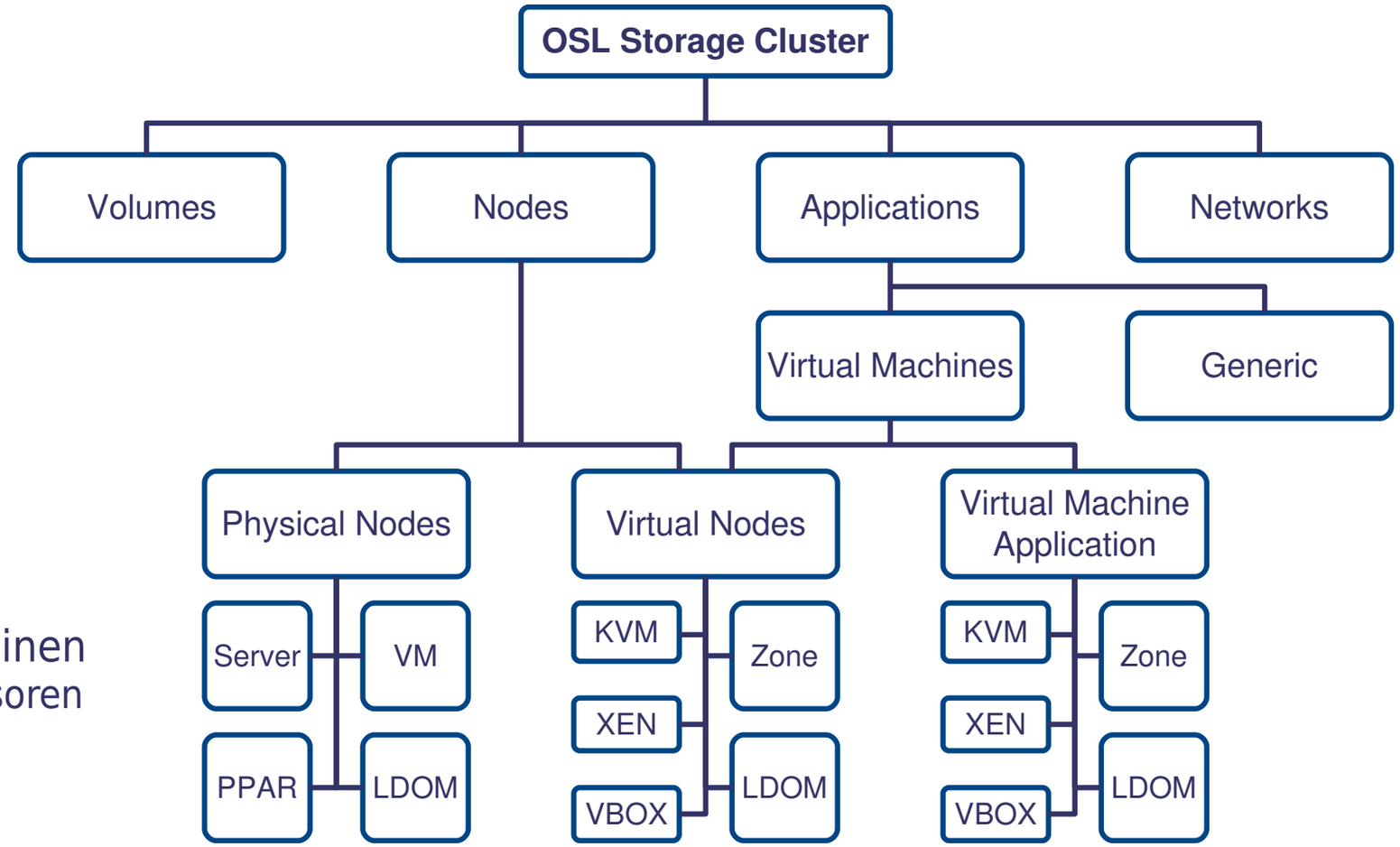


- Das virtuelle Block-Device der Speichervirtualisierung wird direkt an die VM angebunden
  - wird von jedem Hypervisor verstanden – Wechsel leicht möglich
  - keine plattformspezifischen Layer (Linux, Solaris, ISA, ...)
  - minimaler Rechenaufwand, überzeugende Performance
  - funktioniert auch ohne laufenden VM-Monitor
- Block-Devices stehen unabhängig vom VM-Monitor clusterweit zur Verfügung
  - diverse Vorteile, u. a. leichte Migration zwischen Knoten
- Netzwerk- oder Clusterfilesysteme im Stack nicht erforderlich
  - diese bieten Komfort, bergen aber insbesondere in Clustern weitere Komplikationen
  - Stack mit der OSL Virtual Storage Engine flacher
- Block-Sequenzen bleiben erhalten
  - damit bleiben auch Performance-Attribute des Devices bleiben erhalten
  - konsistentere Schreib-Lese-Performance
  - natürlich geht auch Striping, es verliert bei Verwendung schneller SSDs allerdings an Bedeutung
  - geringere Interdependenz verschiedener I/O-Ströme (verschiedener VMs)

Das steckt dahinter: Die OSL-Clusterengine

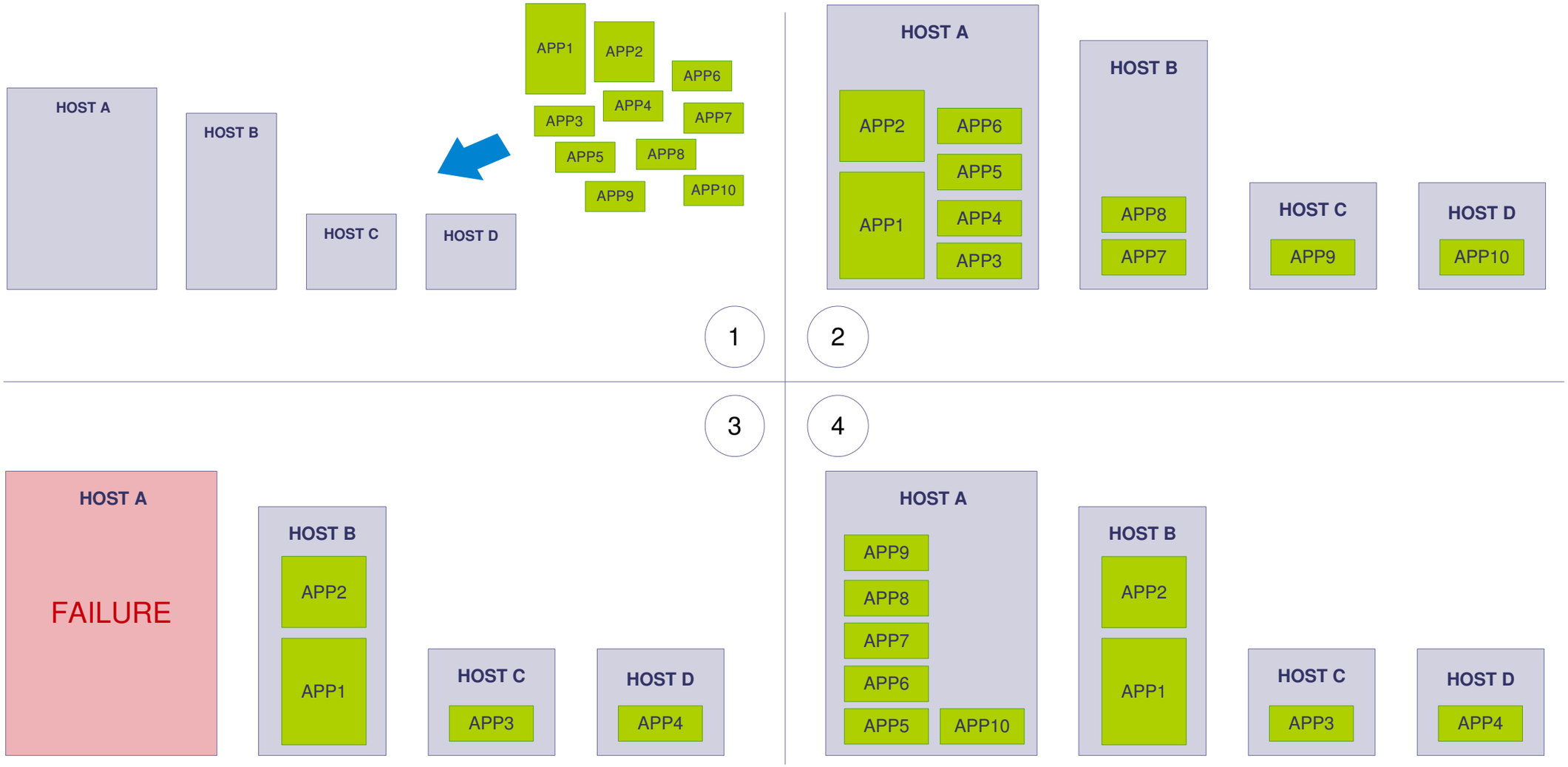
Der Cluster um Shared Storage - Speichervirtualisierung und Clustering in einem

- komplette OSL-Eigenentwicklung
- Grundtechnologie seit über 20 Jahren bewährt und stetig weiterentwickelt
- Hochskalierbar (128 Nodes)
- HA/Selbstmanagement eingebaut
- Unified Networking und Plattform-Mix (Solaris – Linux – Sparc - x86 - ARM)
- Applikationsbewußtsein (Storage, Ressourcen, VNO)
- Abstrakte Integration virtueller Maschinen
  → identische Administration aller Hypervisoren
- Administration und Monitoring
- viele nützliche Helfer und Details



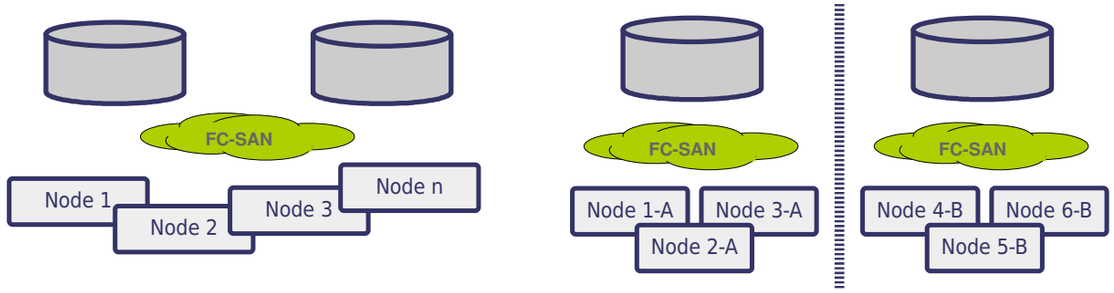
OSL-Clusterengine

Das Grundprinzip des ressourcenbasierten Selbstmanagements



OSL Storage Cluster - Zusammenfassung

Das “Schweizer Offiziersmesser”



- Typisch mit SAN und Shared RAID-Storage
  (andere Formen, etwa mit RSIO auch möglich)
- Vollsymmetrisches Design, Clusterengine
  mit gleicher Technologiebasis wie UVE
- Unterstützt generische Applikationen und VMs
  z. B. für SAP HANA: Bare Metal-, Multi-Tenant- und VM-Setup im gleichen Cluster möglich
- Bedienung analog zum UVE (CLI + WebGUI)
- Alle grundlegenden Funktionen für den Betrieb einer VM-Infrastruktur vorhanden:
  VMs konfigurieren | Ressourcenmanagement | Start, Stopp, LIve-Migration | Monitoring | Hochverfügbarkeit
- Leistungsfähiges Framework, vielfältige Funktionen (inkl. Speicher- & Netzwerkvirtualisierung ...)

Aber:

- Keine HCI-typische One-Stop-Administration (RAID, SAN, Switches separat zu administrieren)
- Nicht das Einsparpotential und die gleichen Möglichkeiten bzw. Freiräume bei Hardware wie im UVE
- Kein so hochgradiger Zuschnitt auf VM-Betrieb wie im UVE (Replikations-Offloading, Dedup, Netzwerk ...)

Die Bausteine im Komplettpaket

Hyperkonvergente VM-Infrastruktur

mit dem

OSL Unified Virtualisation Environment

Erneuerung bietet eine weitere Option: Hyperkonvergenz

Mit einem zeitgemäßen Konzept neue Möglichkeiten erschließen

Hyperkonvergente Infrastruktur (HCI):

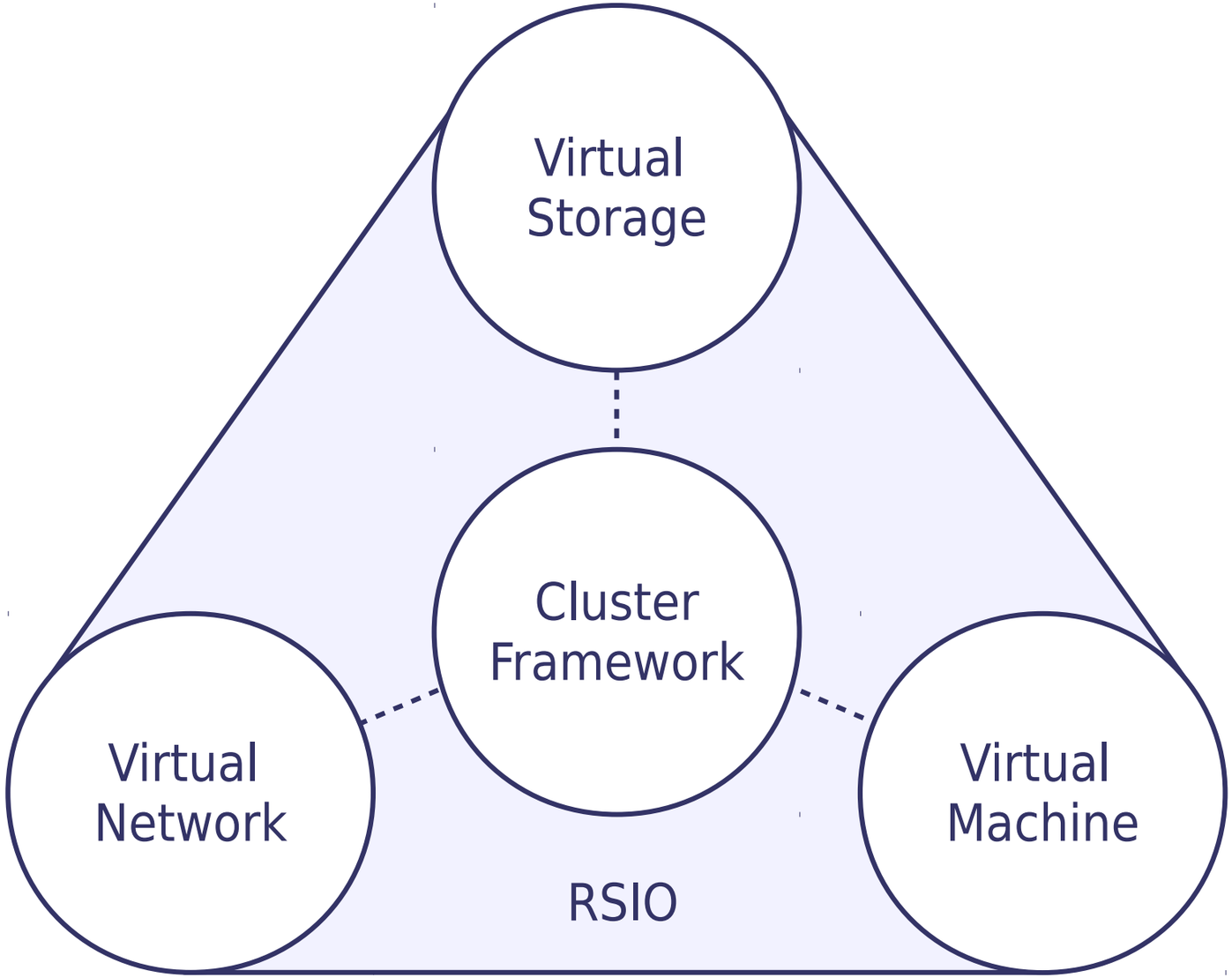
Softwarezentrierte Architektur, die Massenspeicher, Netzwerk und Servervirtualisierung in einem von Grund auf integrierten Paket implementiert und wie ein einziges System zu administrieren ist.

Das ermöglicht:

- einfachere, agilere und rationellere Administration,
- Aufgabe überkommener, aufwendiger Paradigmen (SAN, Storage-Monolithen …),
- enorme Vereinfachung der Infrastruktur und Systemarchitektur,
- Verwendung von Standard-Servern mit Standard-Komponenten,
- direkte und kostengünstige Einbeziehung moderner Technologien (NVMe, Netzwerk).

OSL Unified Virtualisation Environment

HCI-Schlüsseltechnologien in einem ganzheitlichen Konzept

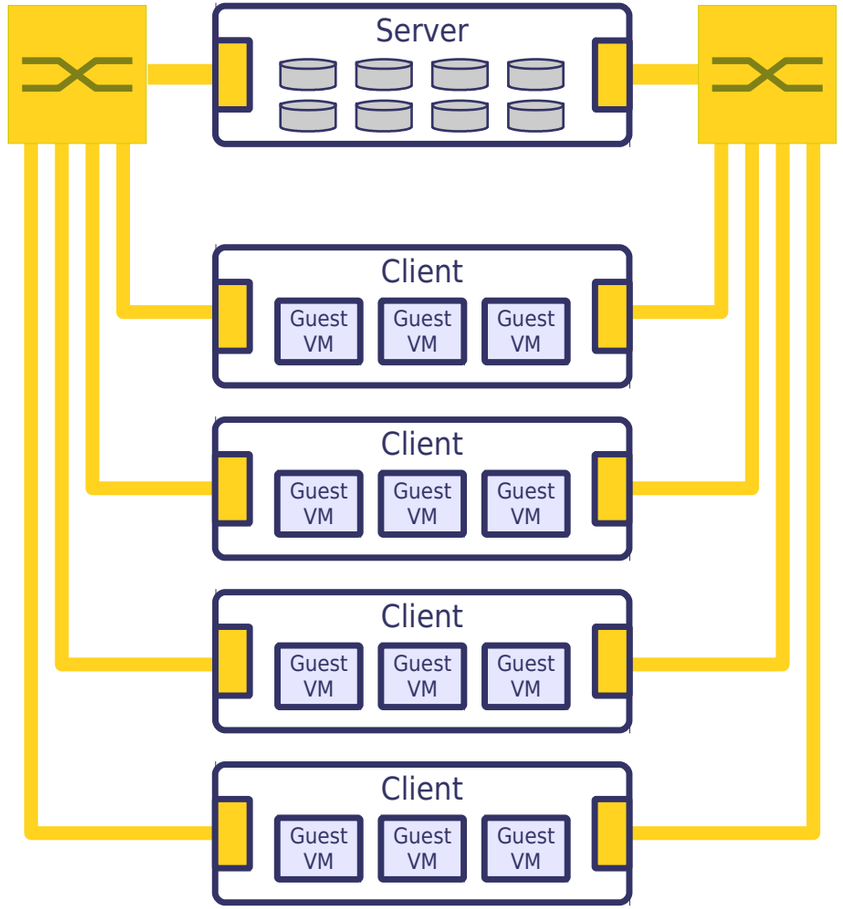


- Eigenständige, aber optimal aufeinander abgestimmte Technologien, eigens für diesen Zweck entworfen
- Hochportable Implementierung, d. h. Beschränkung auf Minimalstandards
- Interoperabel: Solaris, Linux, Sparc, x86
- Made in Germany:
  Alle Schlüsseltechnologien von OSL selbst entwickelt (außer VM-Monitor)
- Abstraktion vom VM-Monitor:
  KVM/QEMU, VirtualBox (LDoms, Xen ...)

OSL Unified Virtualisation Environment

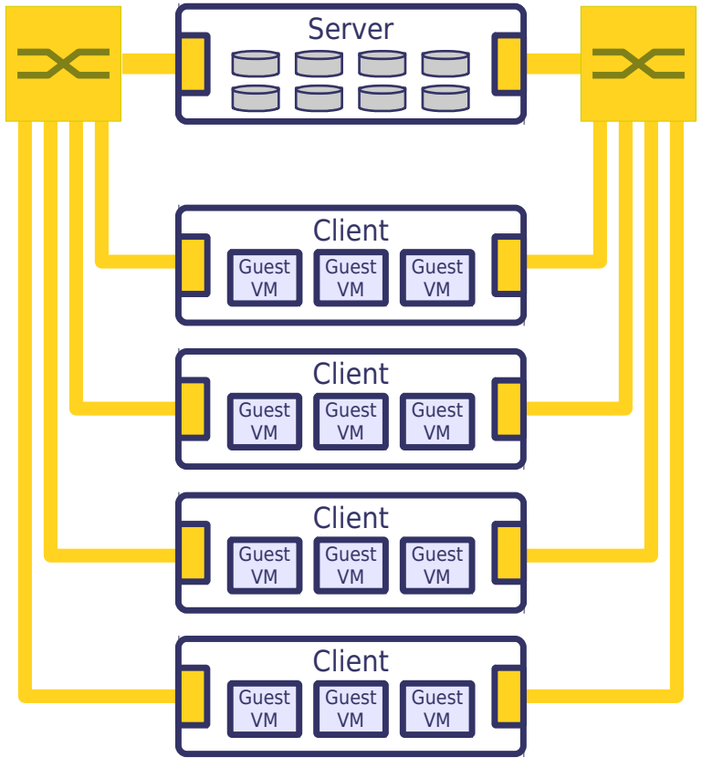
Architektur mit Client-Server-Modell

Unified Virtualisation **UVS** Server	**Single Point of Data Centre Definition & Administration**
	Central Point of Infrastructure Service Delivery
Unified Virtualisation **UVN** Network	**Converged Networking** **Generic Networking + Block-I/O**
Unified Virtualisation **UVC** Client	**Compute Node Farm VM-Execution**



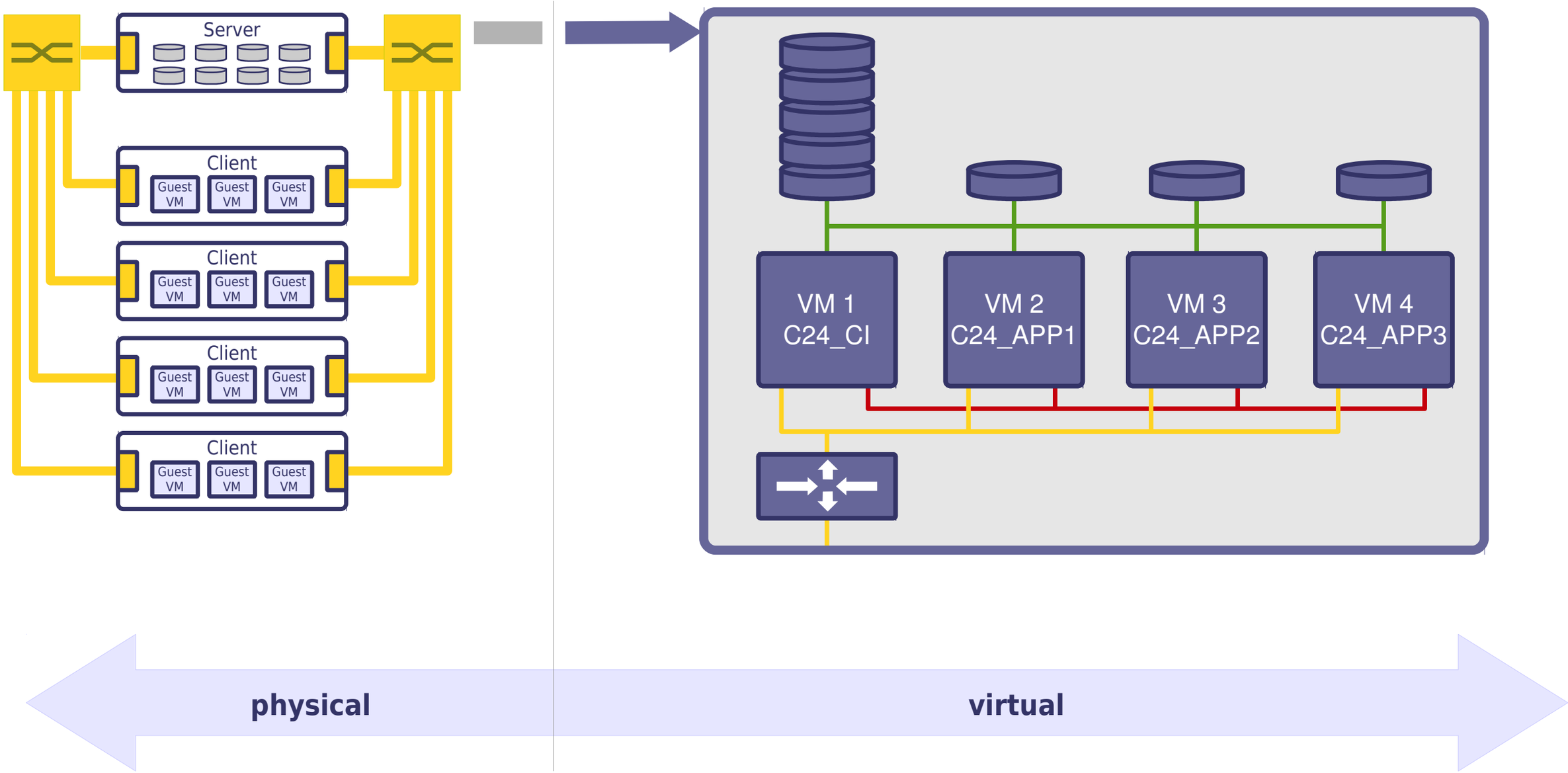
OSL Unified Virtualisation Environment

Abstraktion und Virtualisierung: Neue Infrastrukturen per Software



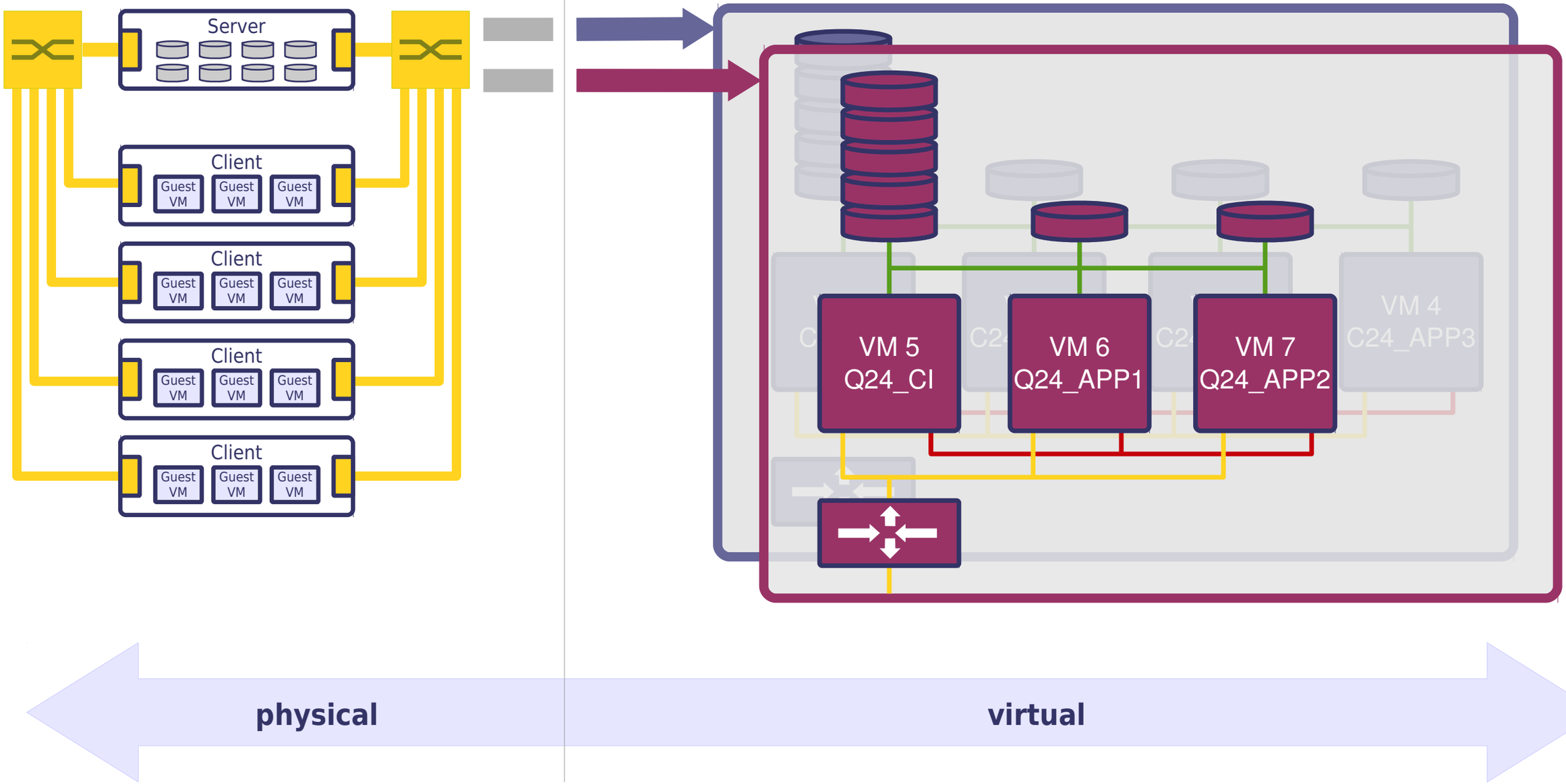
OSL Unified Virtualisation Environment

Abstraktion und Virtualisierung: Neue Infrastrukturen per Software



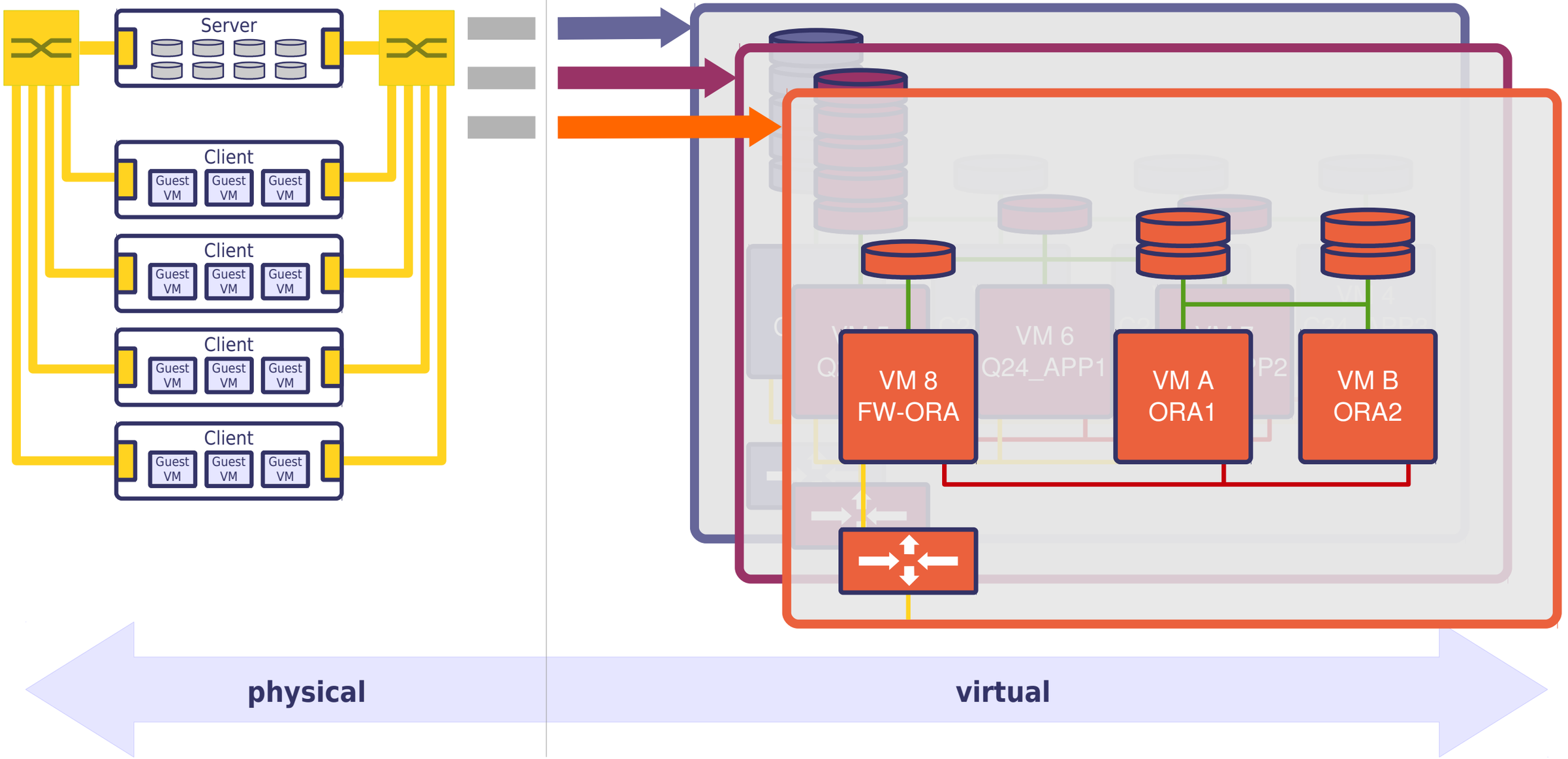
OSL Unified Virtualisation Environment

Abstraktion und Virtualisierung: Neue Infrastrukturen per Software



OSL Unified Virtualisation Environment

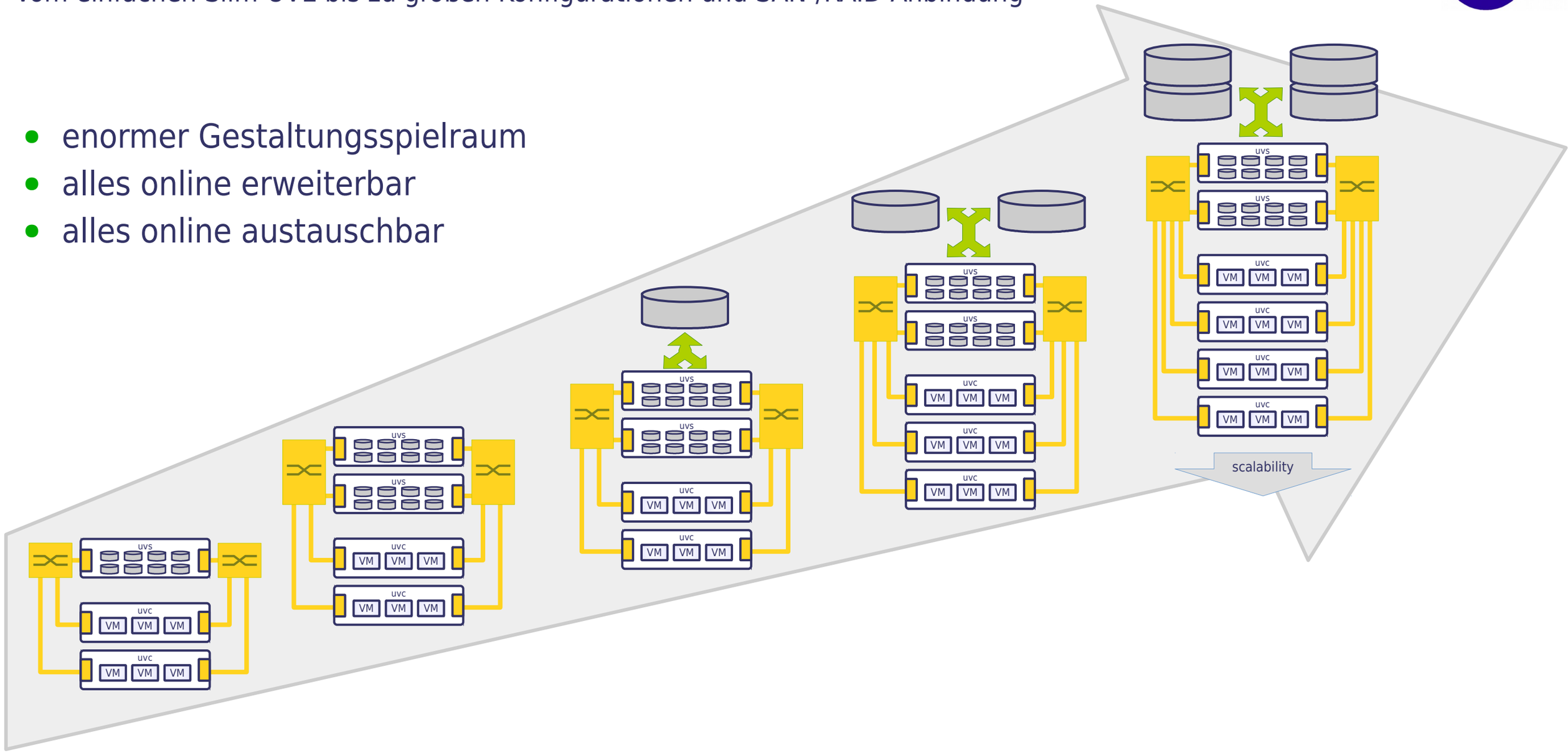
Abstraktion und Virtualisierung: Neue Infrastrukturen per Software



OSL UVE: Beispielkonfigurationen

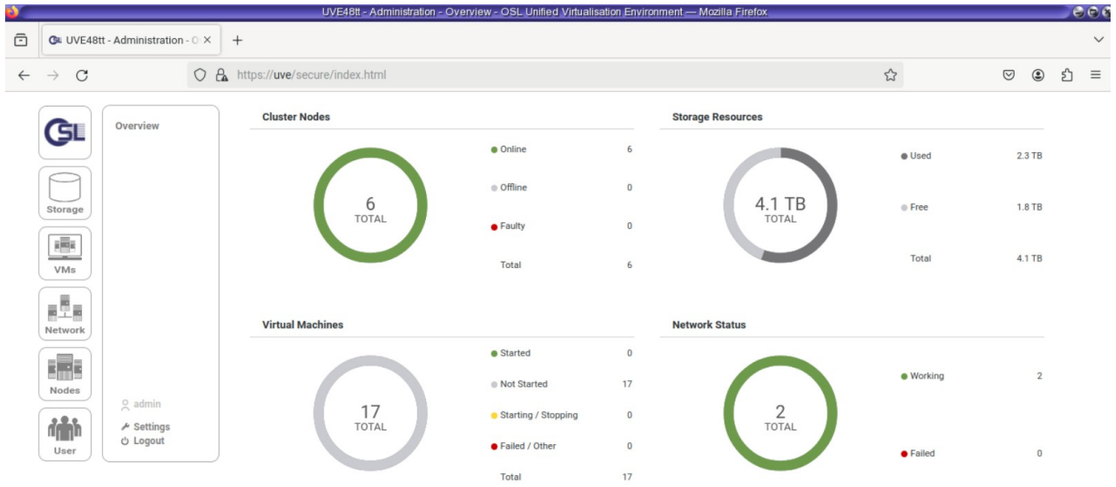
Vom einfachen Slim-UVE bis zu großen Konfigurationen und SAN-/RAID-Anbindung

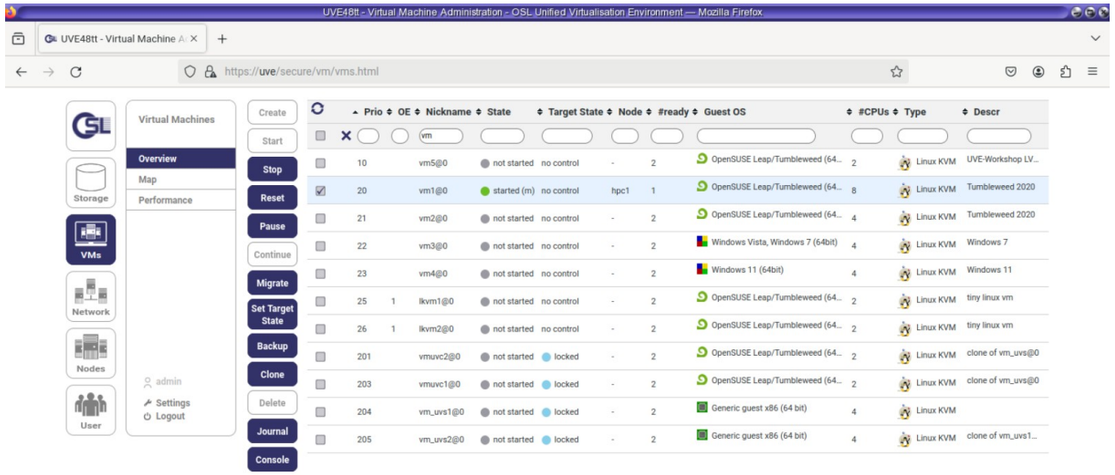
- enormer Gestaltungsspielraum
- alles online erweiterbar
- alles online austauschbar

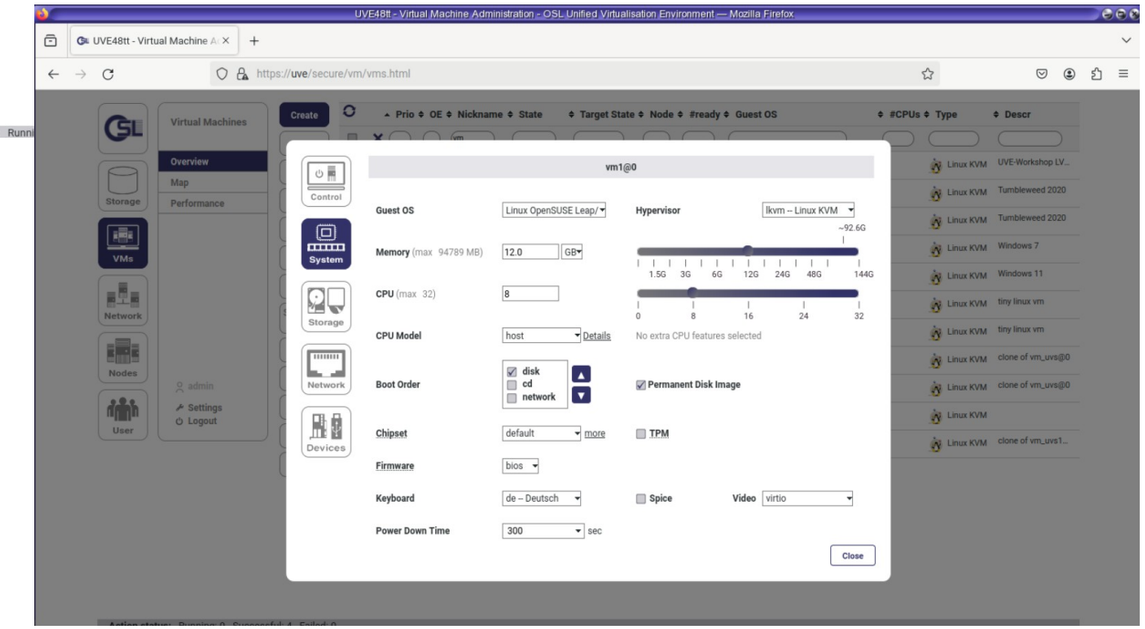


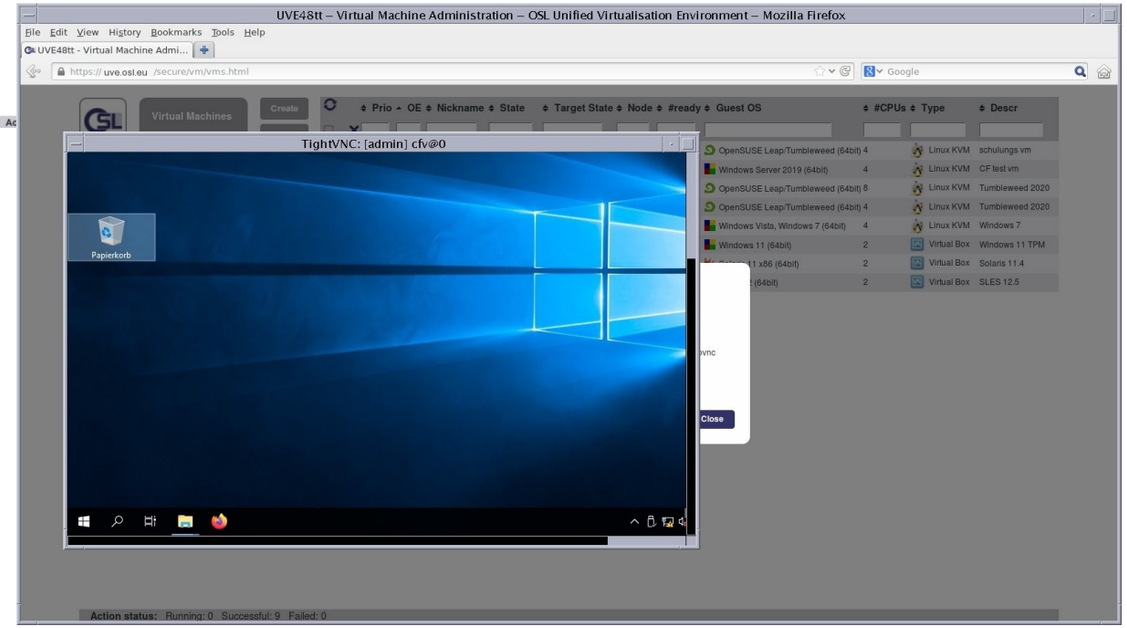
Administration

Über CLI oder WebGUI – Beispiele zur WebGUI





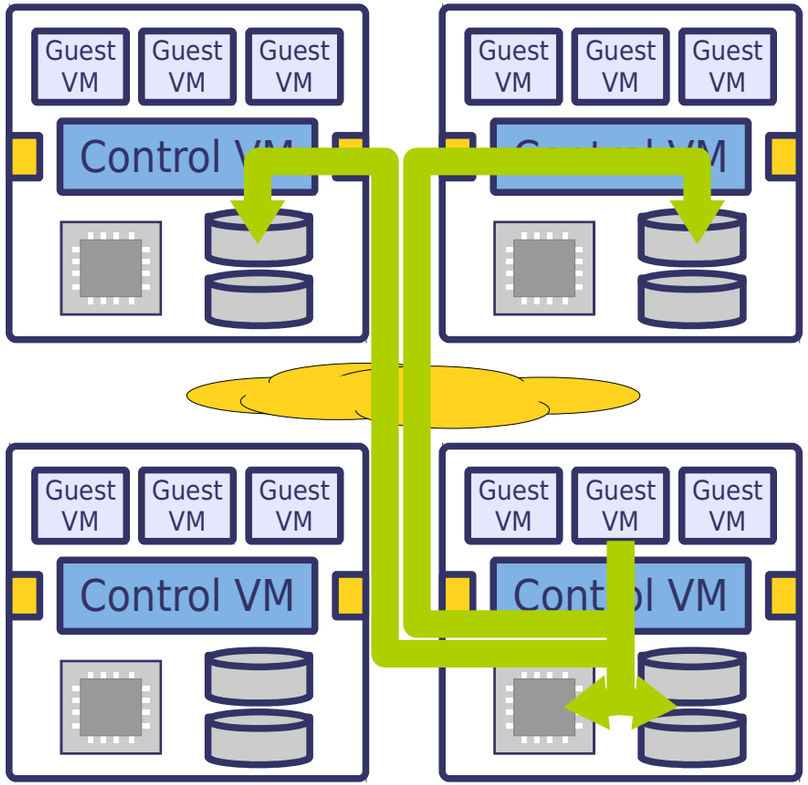




Warum so?

Zum Vergleich: Brick-Design

Ein bei HCI-Lösungen verbreiteter Ansatz



Herausforderungen:

⇒ Mehr Hops im I/O-Pfad

⇒ Höheres Datenvolumen im Netz

⇒ Nutz- und Metadaten laufen mehrfach durch Control-VMs bzw. Storage-Hypervisoren

⇒ Höhere Latenz/Service-Zeit: Es geht nicht ohne SSD

⇒ CPU-Last aus dem I/O kann (situationsabhängig) auf den Compute-Nodes sehr hoch sein

Motivation:

⇒ Nutzung vorhandener Technologien

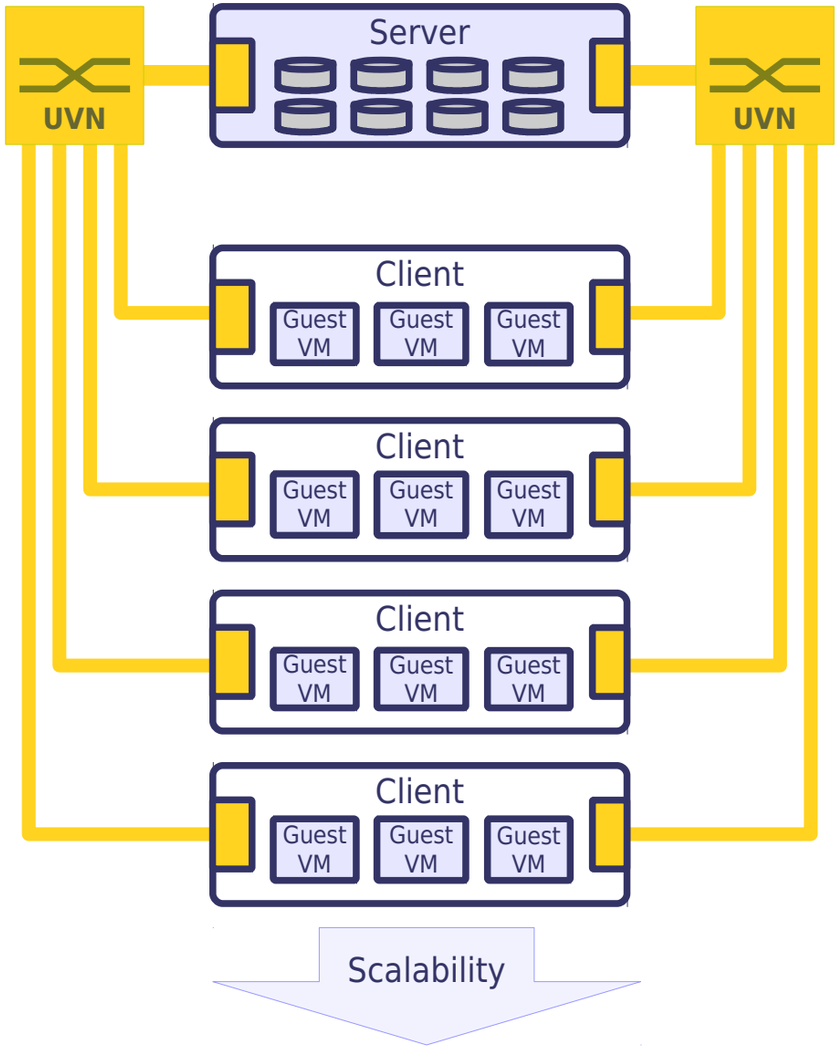
⇒ Horizontale Skalierbarkeit (z. B. Scale-Out-Storage)

⇒ Spezielle Einsatzszenarien (z. B. Globale Replikation)

Warum wählt OSL ein Client-Server-Modell?

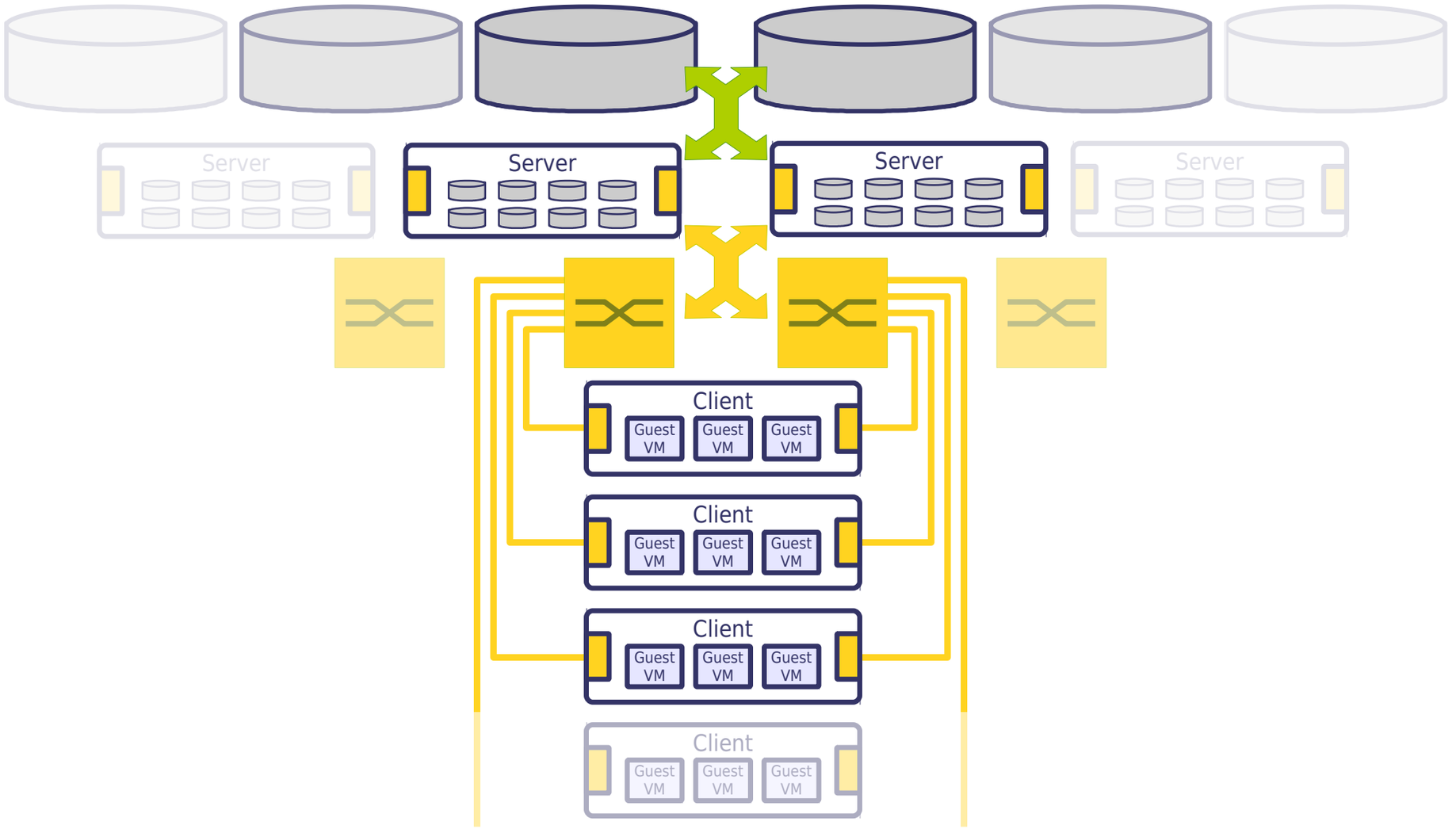
Für viele RZ-Szenarien überlegen: Asymmetrisches Design mit einfacher Handhabung

- Server (UVS) liefert sämtliche Infrastruktur-Dienste:
  Virtual Storage | Virtual Networking | Virtual Machine Control
- Clients (UVC = Compute Nodes) liefern:
  CPU | RAM | VM-Execution
- Mehr VMs? → Skalierung über neue Clients
- Spezialisierter Network-Block-I/O (RSIO)
  → UVN transportiert nur Netto-Daten der VM
  → enorme Entlastung für das Netzwerk
  → kurze I/O-Pfade ohne Umweg über VMs
- Klare Aufgabentrennung:
  → Hypervisorknoten (UVC) braucht nur LAN, CPU, RAM
  → Hypervisorknoten (UVC) ohne Storage-Virtualisierungslast, alleinige Fokussierung auf VM/Compute
  → Reboot oder Ausfall von UVCs für Daten ohne Bedeutung
  → Daten auch ohne UVC und selbstverständlich ohne VM zugreifbar
  → UVS hochspezialisiert, Betrieb ähnlich einem RAID-System



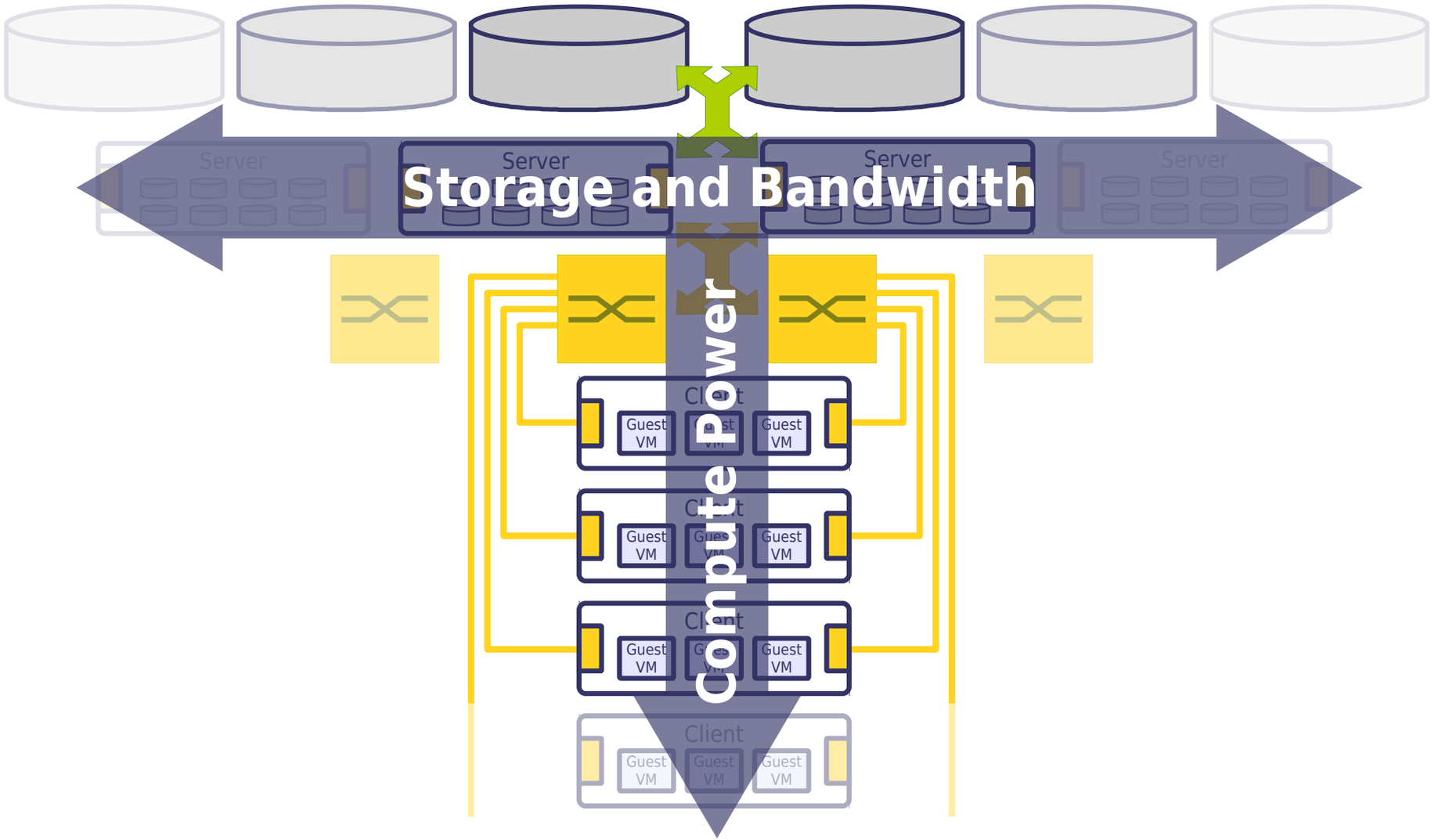
Und was ist mit der Skalierung?

Offenheit, Scale-Up und Scale-Out für Rechenleistung, Netzwerk- und I/O-Bandbreite sowie Speicherkapazität



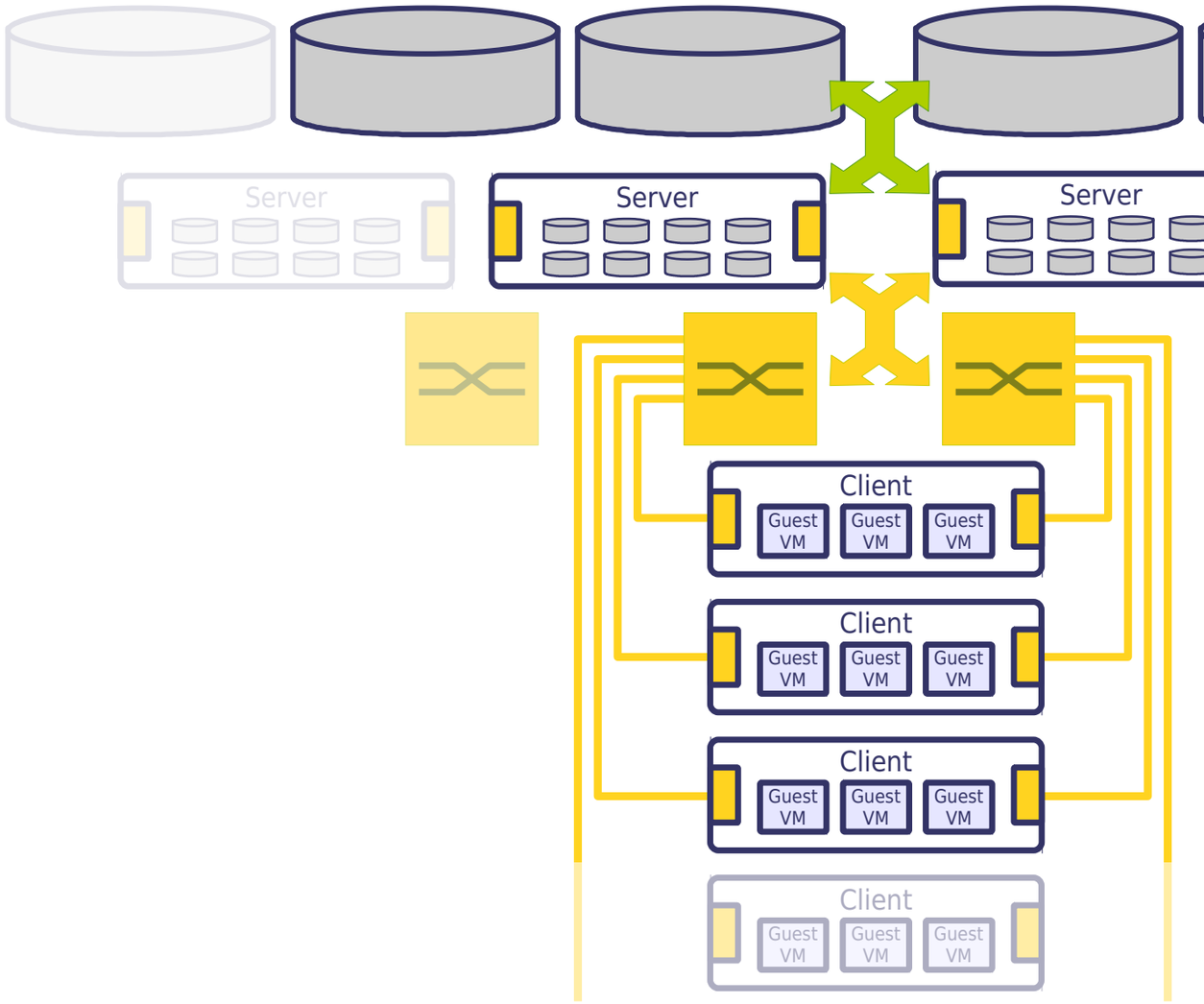
Und was ist mit der Skalierung?

Offenheit, Scale-Up und Scale-Out für Rechenleistung, Netzwerk- und I/O-Bandbreite sowie Speicherkapazität



OSL UVE: Client-Server-Modell mit hybriden Eigenschaften

Offenheit, Scale-Up und Scale-Out für Rechenleistung, Netzwerk- und I/O-Bandbreite sowie Speicherkapazität

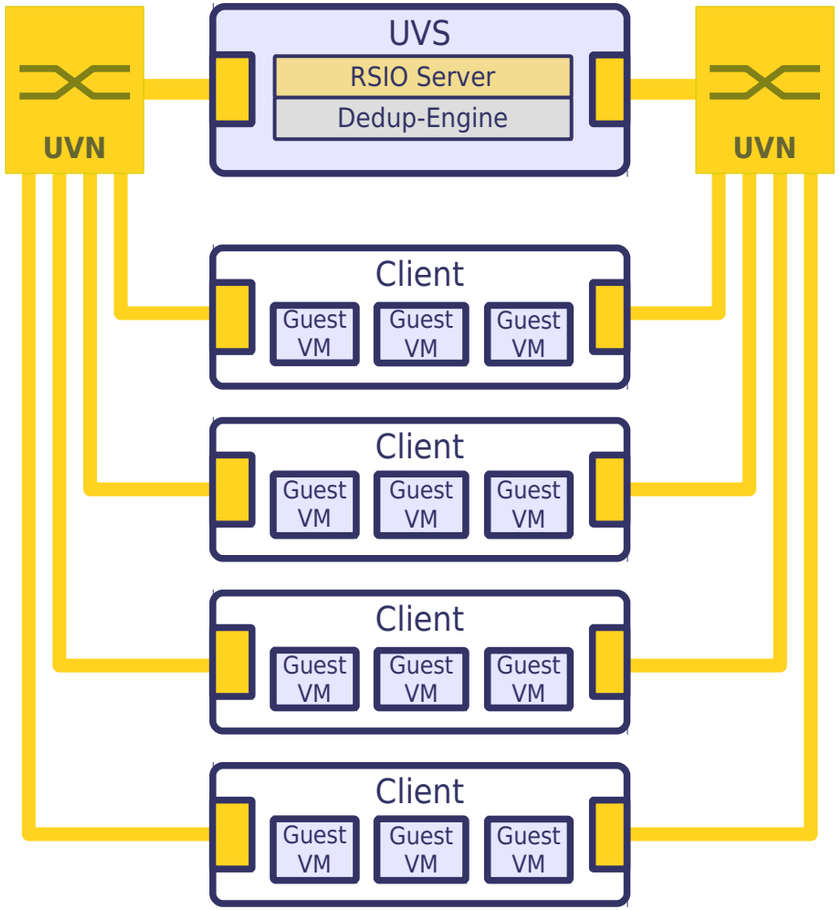


- Vorhandene und neue Hardware nutzbar
- Feingranulare Skalierung (horizontal/vertikal)
- Variabilität: Z. B. Fat Nodes als UVC möglich
- Externer Storage integrierbar
- Datenreplikation ohne Client-Beteiligung
  → max. I/O-Effizienz, minimale Netzwerklast

Auch Deduplizierung profitiert von diesem Design

Gut nutzbare Deduplizierung für Online-Daten und Backup

- Dedup profitiert von Normalisierung / Zentralisierung, (im Gegensatz zu verteilten Lösungen)
- Alle Knoten, alle VMs können in einem Pool arbeiten
- Dedup zieht keine CPU-Performance auf den UVC
- Große Effizienz durch 4k-Blöcke
- VSD/RSIO erlaubt Live-Konvertierung, Spiegelung usw.
- Mitunter schneller als auf normaler SSD
- Praxis-Empfehlung: MegaRAID mit großem Cache



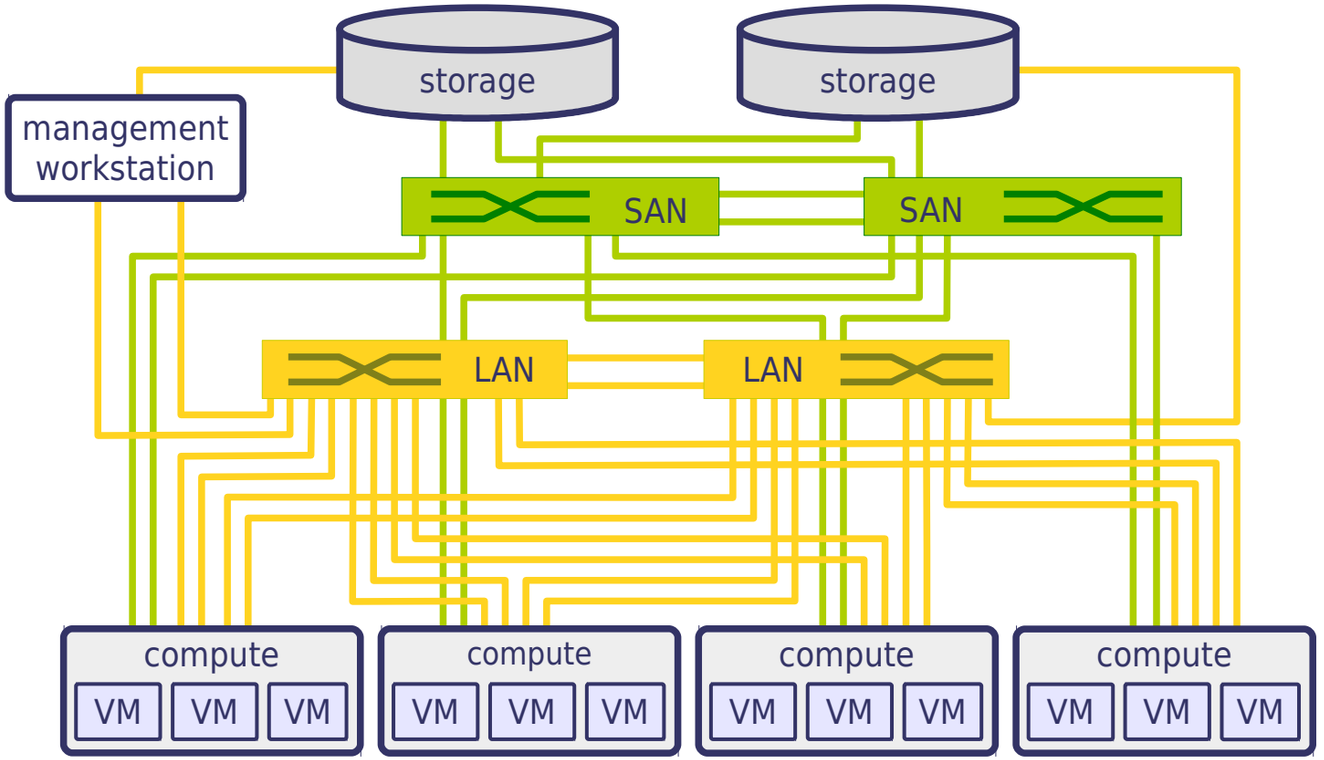
s.a.: https://www.osl.eu/aktuelles/aktuelle-themen/deduplizierung-im-uve

Ein Blick in die Praxis

Beispiel Hardware - konventionell

Konventionelle Vergleichskonfiguration mit Shared Storage

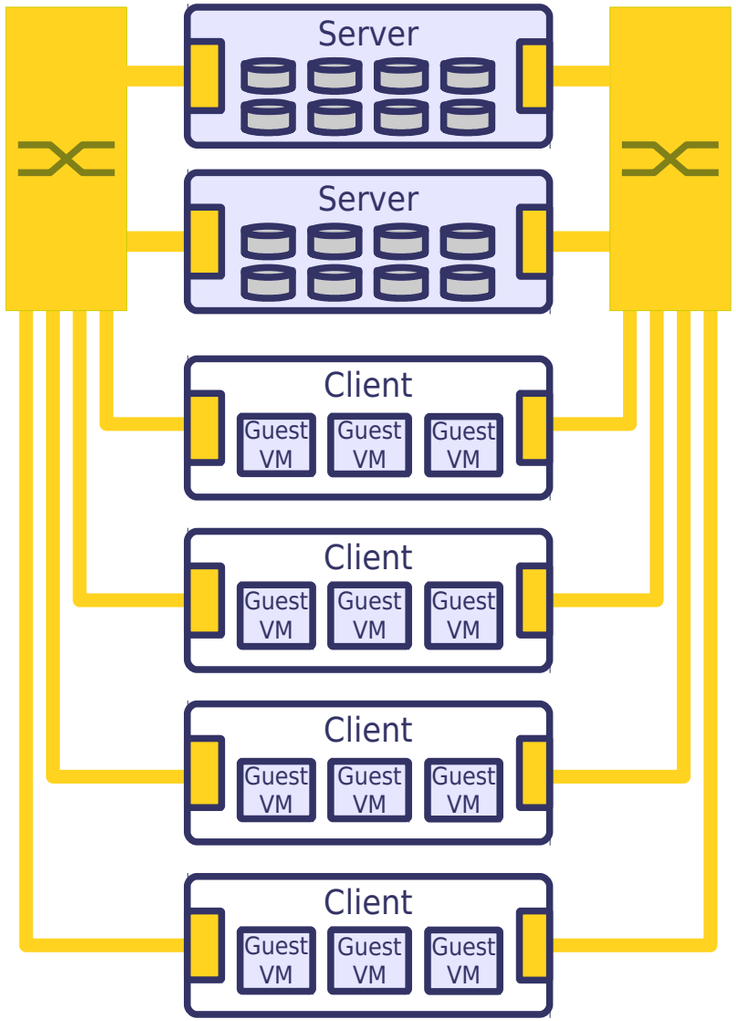
- redundante Raid-Systeme
  - 2x All Flash 90TB 32G
- redundantes Fibre-Channel SAN
  - 2x 32G FC-Switch 24 Ports
  - redundante Verkabelung
  - 24x SFP+ Tranceiver
- 4 Hosts
  - Intel Dual Xeon
  - 2x Intel Xeon Gold 5418Y 24 Cores, 2.00 GHz
  - 128GB Hauptspeicher
  - 32GB FC HBA Dualport PCIE 4.0
  - 10GbE Controller onboard
- redundantes 10GbE Netzwerk
  - 10G Ethernet Switches
  - redundante Verkabelung



Beispiel Hardware – OSL UVE

Konfiguration mit OSL UVE

- 2 UVS mit internem Storage
  - 2x AMD EPYC 7413 24C 2.65GHz
  - 128GB Hauptspeicher
  - 100GbE Controller PCIE 4.0 x16 2Ports
  - 6x 16TB NVMe SSD U.3 → 96TB
- redundantes 100GbE Netzwerk
  - 16 Port 100GbE Switch
  - redundante Verkabelung
- 4 UVC-Hosts
  - Intel Dual Xeon
  - 2x Intel Xeon Gold 5418Y 24 Cores, 2.00 GHz
  - 128GB Hauptspeicher
  - 100GbE Controller PCIE 4.0 x16 2Ports
  - 10GbE Controller onboard



Was gewinne ich?

Offenheit und Integration

Architektonische Offenheit und große Integrationstiefe stärken die Autonomie des Anwenders

- Hauptaspekt nicht unübertroffene Funktionsvielfalt sondern Integration und Vereinfachung (Administration, Hardware, Prozesse)
- OS-, Hypervisor- und Hardwareabstraktion → langfristig stabile Prozesse
- Offene Architektur geht über die Annahme “Open Source = Handlungsfreiheit“ hinaus: HW- und Softwarekomponenten austauschbar ohne das Gesamtkonzept in Frage zu stellen
- Niedrige Einsteigshürden (auch beim Know-how), minimaler Lock-In, einfacher Exit
- Tiefe Integration eröffnet neue Möglichkeiten und reduziert Anforderungsvielfalt
  Beispiele: Automatisierung, Provisionierung, Reorganisation sind Teil des Frameworks
- Für RZ-Betreiber zügige, risikoarme Implementierung (Projektbegleitung durch Partner), professioneller Support, dauerhaft niedriger Dienstleistungsbedarf

Warum Linux eine starke Basis ist

Eine Betrachtung mit Fokus auf professionelle Anwender

OSL Storage Cluster und OSL Unified Virtualisation Environment sind portabel bzw. plattformneutral konzipiert, aber

Linux bietet:

- Alle benötigten Funktionen
- Überzeugende Hardwareunterstützung, insbesondere für moderne, schnelle Speicher- und Netzwerkkomponenten, GPUs usw.
- Entsprechend den Erfordernissen / Anwenderwünschen wählbare Distributionen und Supportkonzepte von Community-Versionen bis hin zu LTS-Enterprise-Plattformen
- Stabilität, Performance und ein lebendiges Ökosystem

Fragen und Antworten

Fragen und Antworten

Im Anschluss an den Vortrag auf dem Kielux

Frage:

Wer kümmert sich beim OSL UVE eigentlich um die Storage-Redundanz bzw. Replikation?

Antwort:

- Grundsätzlich können autonom arbeitende Storage-Lösungen wie RAID-Systeme (auch gegeneinander gespiegelt, also Metro-Cluster u. ä.) beibehalten werden.
- Die UVS können Daten zwischen zwei RAID-Systemen auch eigenständig und synchron spiegeln, auch zwischen RAID-Systemen unterschiedlicher Hersteller
- Die UVS können für internen Storage RAID-Controller oder Software-Schutzmechanismen verwenden und sie können bei redundanter Auslegung auch für die synchrone Replikation in dem jeweils anderen UVS sorgen

Fragen und Antworten

Im Anschluss an den Vortrag auf dem Kielux

Frage:

Wie funktioniert das mit den Lizenzen und der Implementierung eines UVE-Projektes, über Partner?

Antwort:

- Die Software erhalten Sie unter einer OSL-spezifischen, konsequent am deutschen Recht orientierten, fairen und liberalen Lizenz. Ihr Nutzungsrecht ist also im deutschen Recht zu Hause, die Auslegung und rechtliche Handhabung für hiesige Anwender damit vermutlich einfacher als bei einem global bzw. US-amerikanisch gepägten Lizenzmodell.
- Für jeden UVS und jeden UVC erwerben Sie eine unbefristete Lizenz mit jeweils einem Jahr Maintenance. Lizenzkosten enstehen dabei nur für die UVS, d. h. unter dem Aspekt der Nutzungsrechte können Sie den Cluster ohne zusätzliche Kosten um weitere UVC erweitern. Allerdings: SW-Maintenance muss immer alle Knoten eines Clusters umfassen. Mehr UVC-Knoten bedeuten also höhere Maintenancekosten. Natürlich können Sie den Cluster nach Ablauf der einjährigen Maintenance auch ohne diese weiterbetreiben, haben dann jedoch keinen Zugriff auf Korrekturen und Updates.
  Weitere Informationen finden Sie auch im Web: https://www.osl.eu/software/lizenzen-und-maintenance/
- Bei der Planung und Umsetzung von Projekten unterstützen Sie OSL-Partner bzw. auch OSL selbst.
  In Schleswig-Holstein können Sie sich z. B. an baltic-online wenden.

Fragen und Antworten

Im Anschluss an den Vortrag auf dem Kielux

Frage:

Wenn es im Vortrag um Datendeduplizierung ging, fiel einem beim Linux natürlich sofort als Filesystem Btrfs ein oder aber was wahrscheinlicher ist, ZFS. Also beim UVS setzt ihr vermutlich ZFS ein?

Antwort:

- Prinzipiell können alle auf der jeweiligen Plattform verfügbaren Dateisysteme verwendet werden. Grundsätzlich arbeitet die Speichervirtualisierungsengine jedoch blockorientiert. Die Brücke zu nützlichen, eher dateisystemorientierten Funktionen bildet dabei das LFS-Subsystem (LFS = Lokal File Storage) in den UVS. Wir empfehlen ZFS für UVS, die unter Solaris laufen. Die im Kernel integrierte Linux-Alternative ist am ehesten BTRFS.
  Details: https://www.osl.eu/aktuelles/aktuelle-themen/osl-storage-engine-48-mit-lfs/
- Eine Deduplizierung mit ZFS wurde durch OSL evaluiert, aber nicht weiterverfolgt.
  Derzeit favorisieren wir die seit 6.9 im Linux-Kernel verankerte. blockorientierte Permabit-Technologie, die auch online eine beeindruckende Performance liefert. Sie wird üblicherweise über den LFS-Layer (XFS oder BTRFS) eingebunden.
  Details: https://www.osl.eu/aktuelles/aktuelle-themen/deduplizierung-im-uve/

Fragen und Antworten

Im Anschluss an den Vortrag auf dem Kielux

Frage:

Das Bootimage der VMs wird wahrscheinlich per PXE über die superschnelle, redundante Netzanbindung gestartet und der UV Server (oder eine VM darauf) ist wahrscheinlich der dazugehörige TFTP-Server?

Antwort:

- Die UVC müssen nicht wissen, wo ihre virtuellen Festplatten liegen. Das wird komplett durch die UVS gemanagt.
- Ein virtuelles Festplatten-Image gelangt per RSIO (Netzwerk) zu den UVC und wird im Hypervisor als Raw-Device eingebunden.
  Aus Sicht der VM handelt es sich um eine normale Festplatte, von der die VM auch bootet.
  Weitere Disks können bei Bedarf (auch live) hinzugefügt werden. Auch eine Größenänderung ist online möglich.
  Auf diese Weise wird Plattformneutralität (Big Endian/Little Endian, Solaris/Linux) und Hypervisorneutralität erreicht.
- PXE und TFTP sind am Start einer VM üblicherweise gänzlich unbeteiligt. Man kann aber natürlich VMs auch über ihren Netzwerkstack booten – das ist aber in der Gesamtheit für den Admin deutlich aufwendiger.

virtualization and clustering – made simple