science + computing

| an atos company

Ohne Tempolimit: I/O-Performanceproblemen in Linux-Systemen auf der Spur

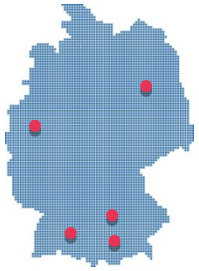
Daniel Kobras
science + computing ag
IT-Dienstleistungen und Software für anspruchsvolle Rechnernetze
Tübingen | München | Berlin | Düsseldorf

science+computing ag

science + computing
| an atos company

Gründungsjahr 1989

Standorte



Tübingen
München
Berlin
Düsseldorf
Ingolstadt

Mitarbeiter 275
Hauptaktionär Atos SE (100%)
Umsatz 2013 30,70 Mio. Euro

Portfolio

IT Service für komplexe Berechnungsumgebungen
Umfassende Lösungen für Linux- und Windows-basiertes **HPC**

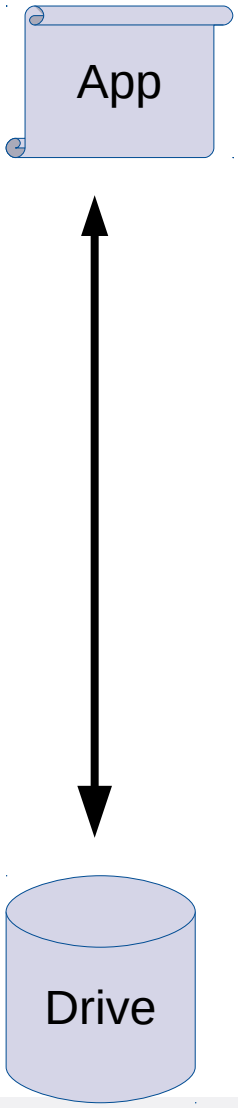
scVENUS Systemmanagement-Software zur effizienten Administration homogener und heterogener Netze

Motivation

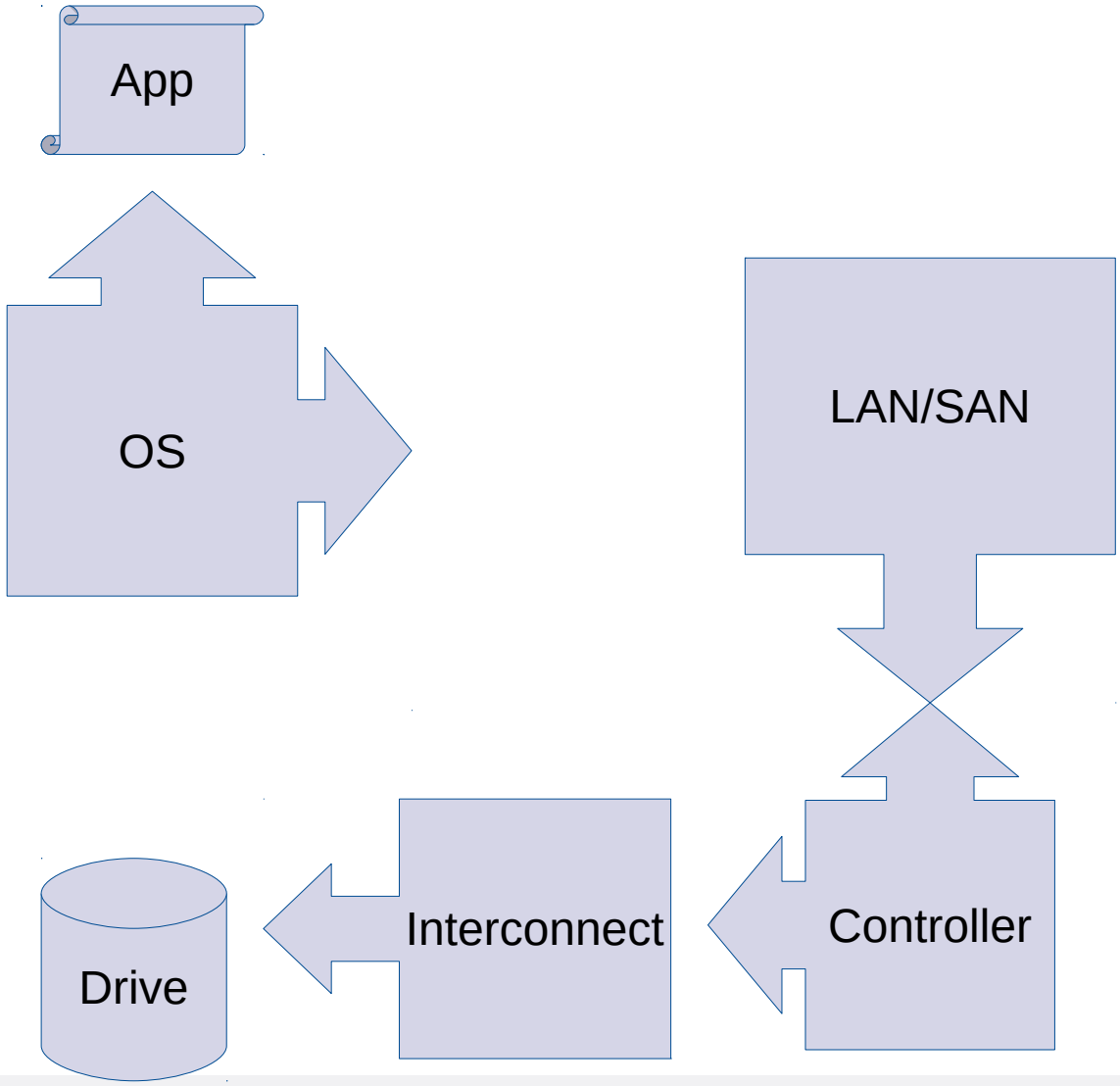
science + computing
| an atos company

- Performance-Probleme schwer zu debuggen
  - Subjektiver Eindruck des Anwenders oft Anlass für Beschwerden
  - Häufig keine klaren Fehlermeldungen im Systemprotokoll
  - Zahlreiche beteiligte Subsysteme als potentielle Fehlerquellen
- Echtes Fehlverhalten? Überlastete Ressourcen? Überzogene Erwartungen?
- Idealerweise bei Inbetriebnahme umfassende Soll-Performancewerte erfasst (aka Theorie)
- Sonst „blindes“ Performance-Debugging mit unzureichenden Informationen (aka Praxis)

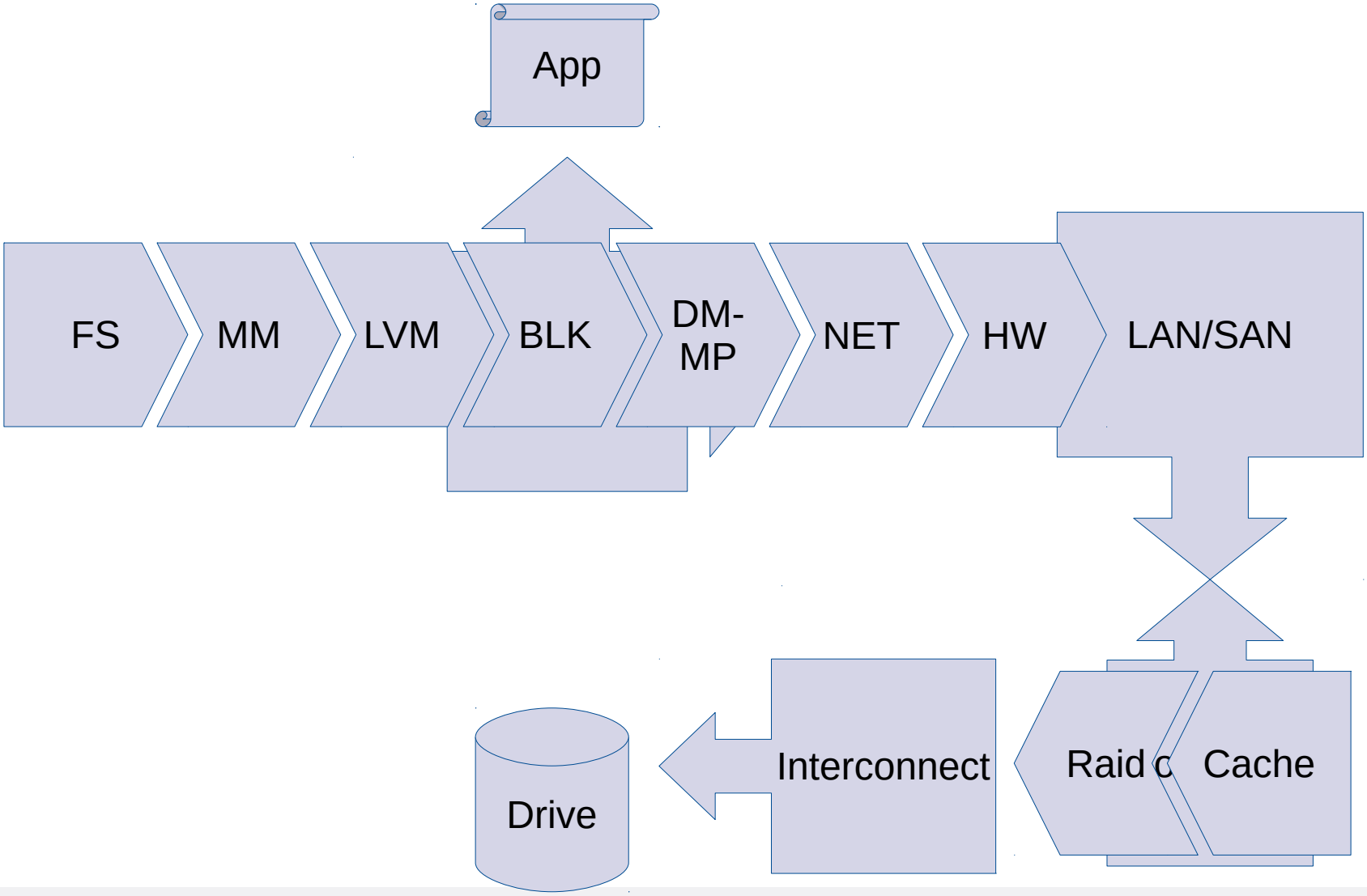
Übersicht



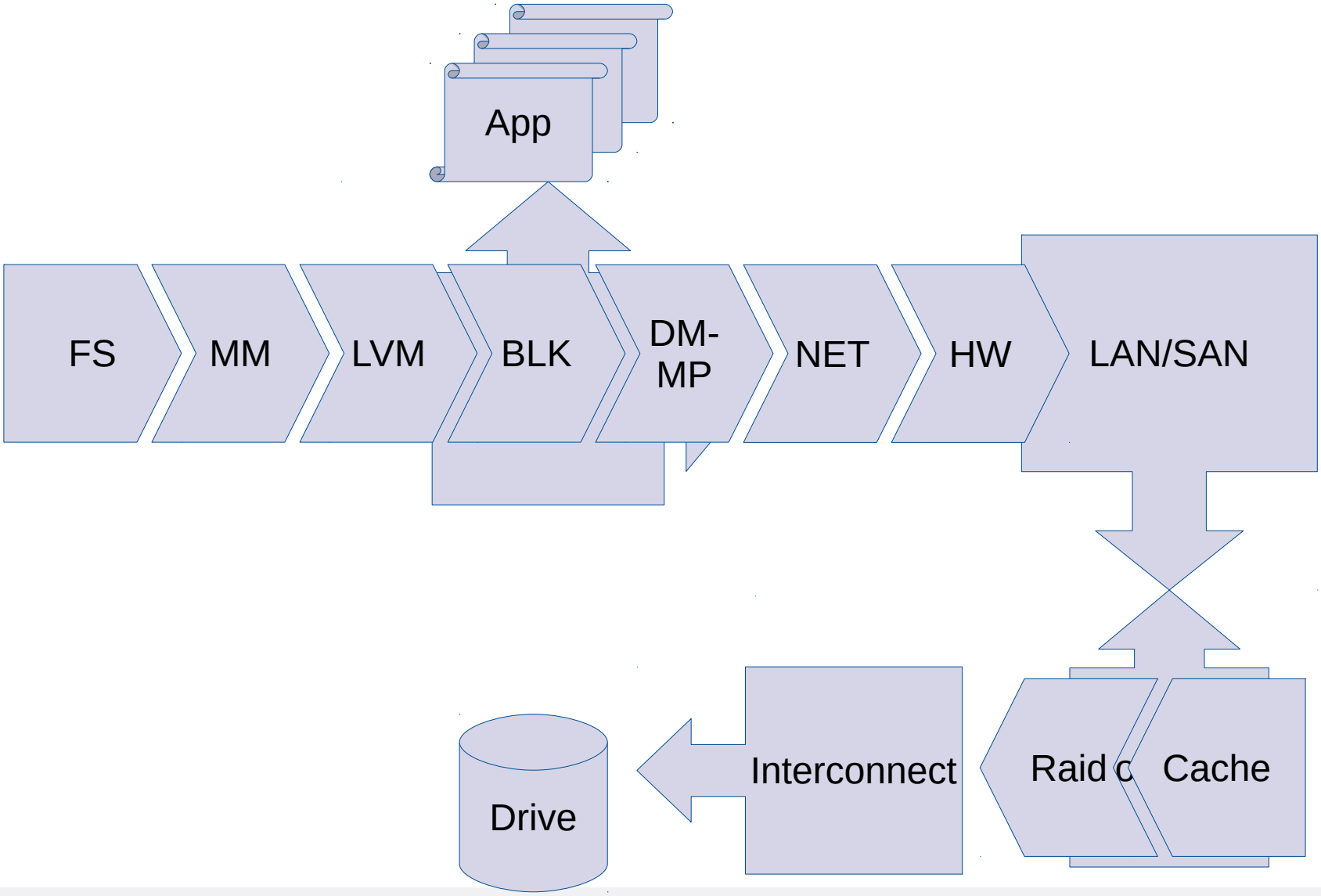
Übersicht



Übersicht

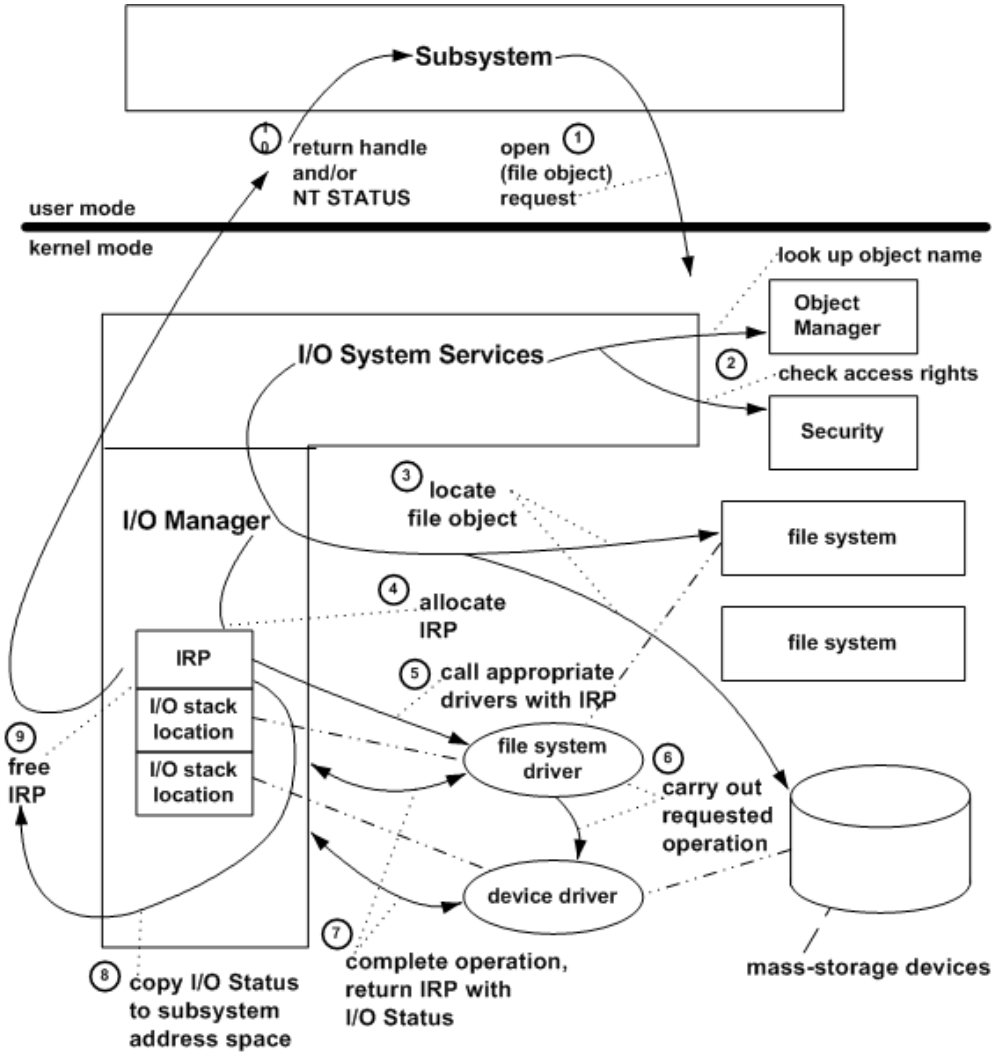


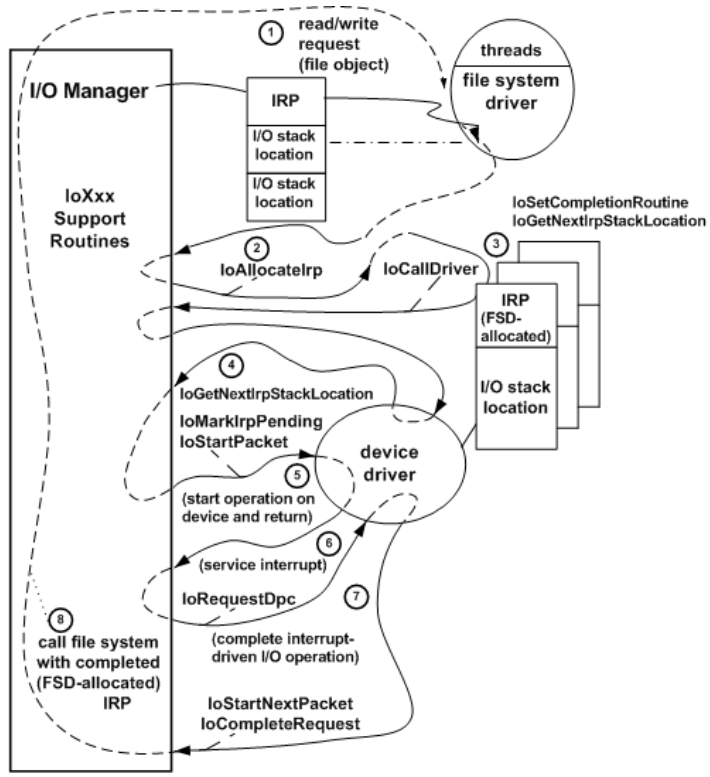
Übersicht



What could possibly go wrong?

science + computing
| an atos company

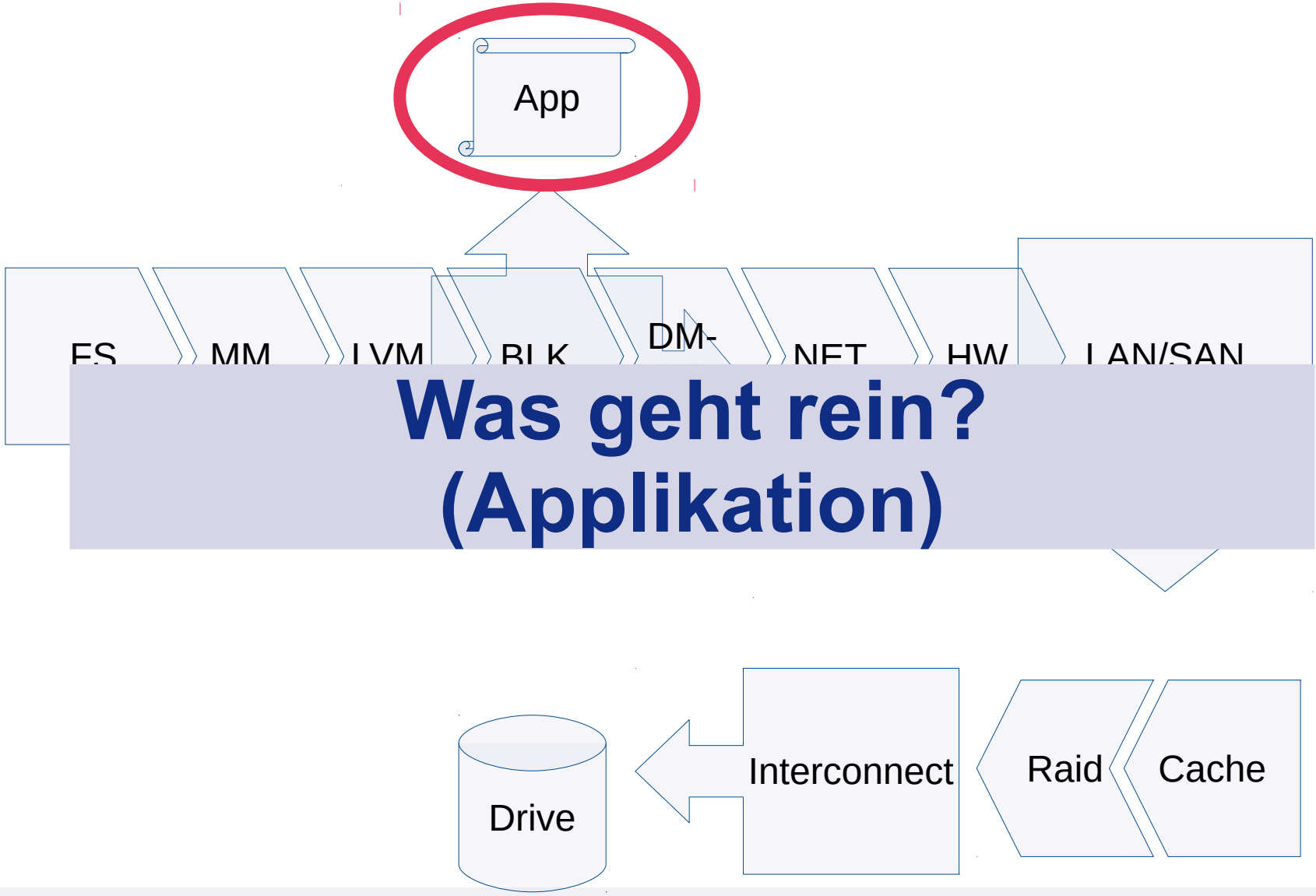




Quelle: http://msdn.microsoft.com/en-us/library/windows/hardware/ff544536%28v=vs.85%29.aspx

Voraussetzungen

- Datenträger definiert I/O-Muster, das er optimal verarbeiten kann
- Applikation muss dieses I/O-Muster liefern
- I/O-Stack muss dafür sorgen dass, I/O-Muster sich auf dem Weg zum Datenträger nicht verschlechtert

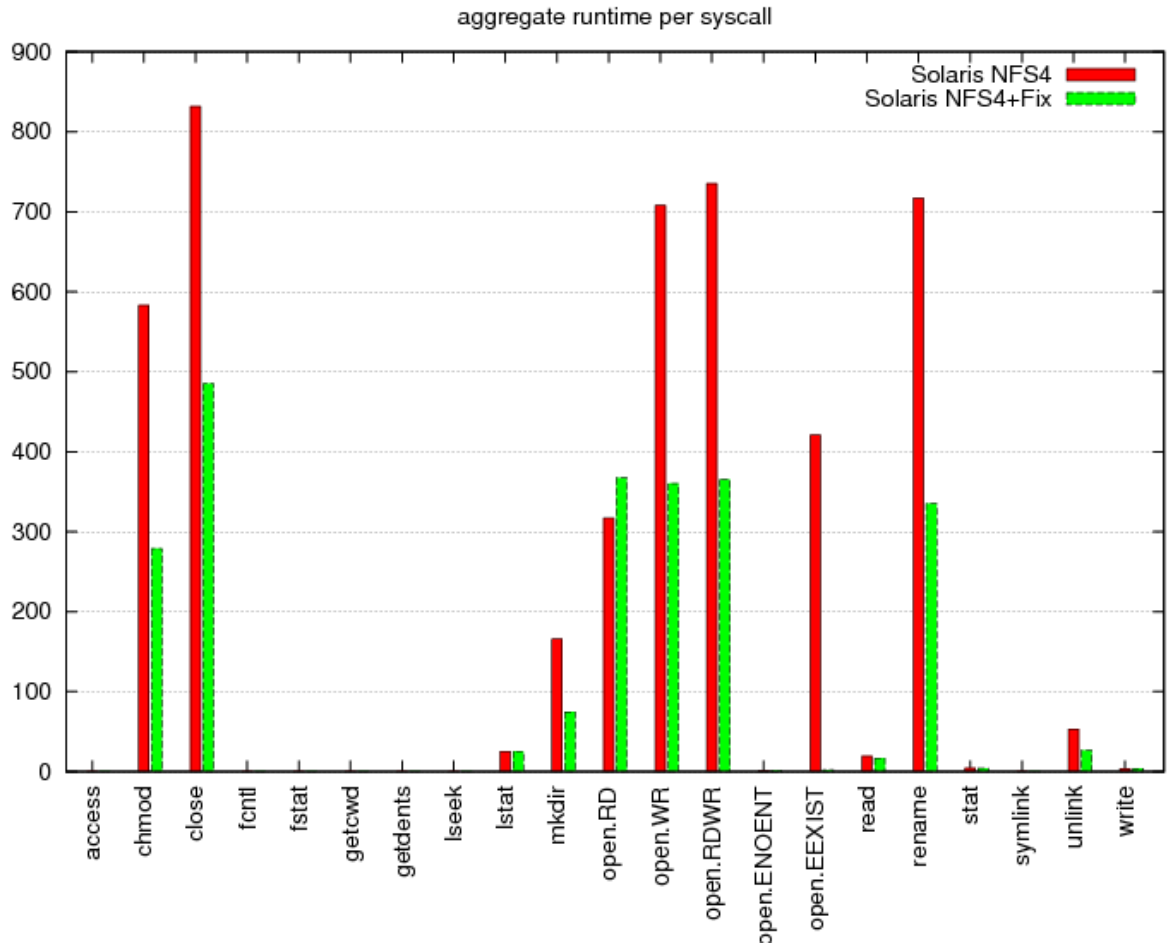


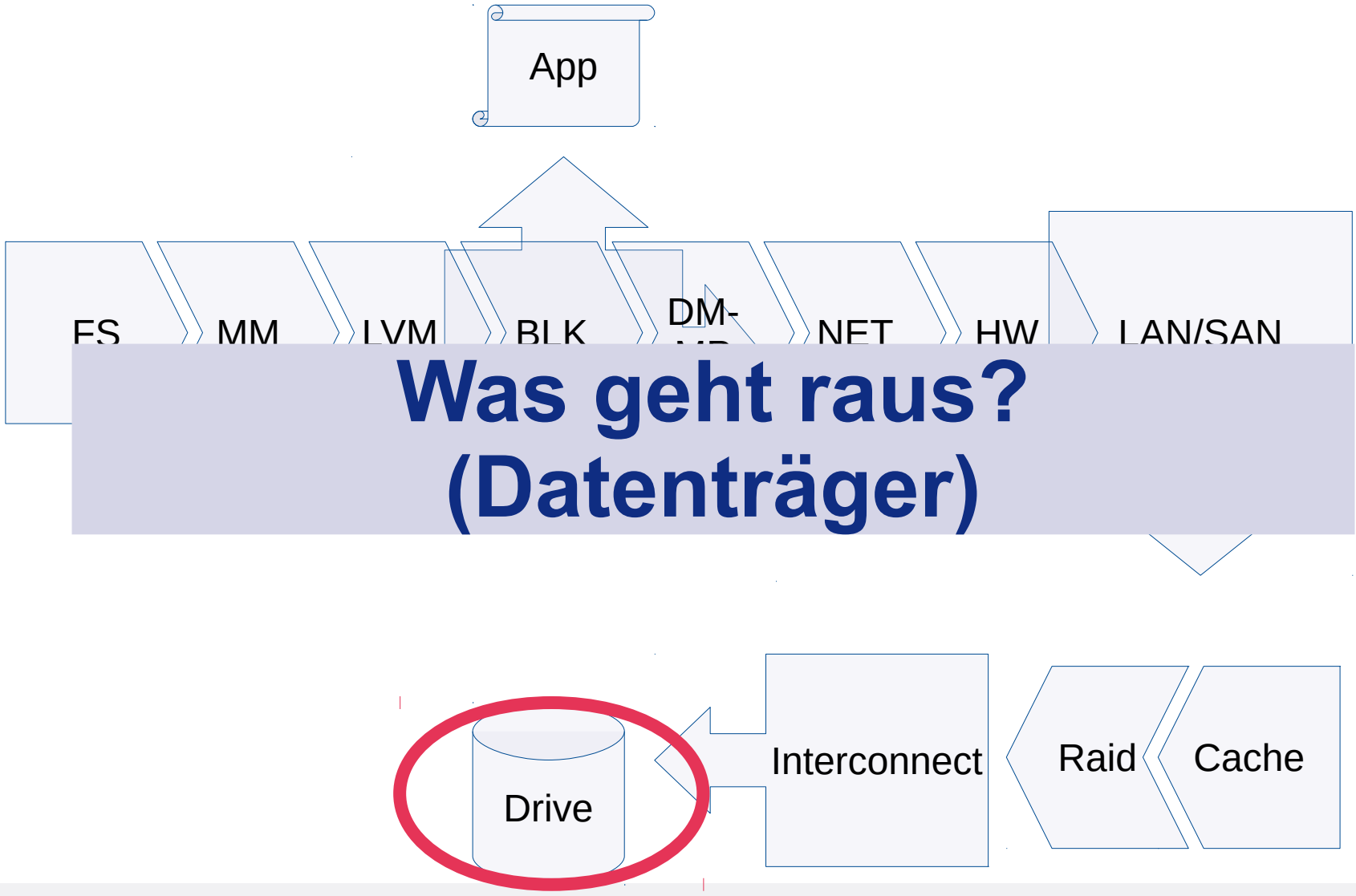
Was geht rein? (Applikation)

Applikationsprofile

- I/O-Profil (ausgeführte I/O-Operationen):
  - Welche?
  - Wie viele?
  - Wie lange?
  - Wie groß?
- Syscall-Tracing (*read()/*write()/*seek()/open()/close())
  - eg. strace -T (oder perf, dtrace, ...)
  - Problematisch: I/O über mmap()

Beispiel: Applikationsdebugging





Was geht raus? (Datenträger)

Datenträger

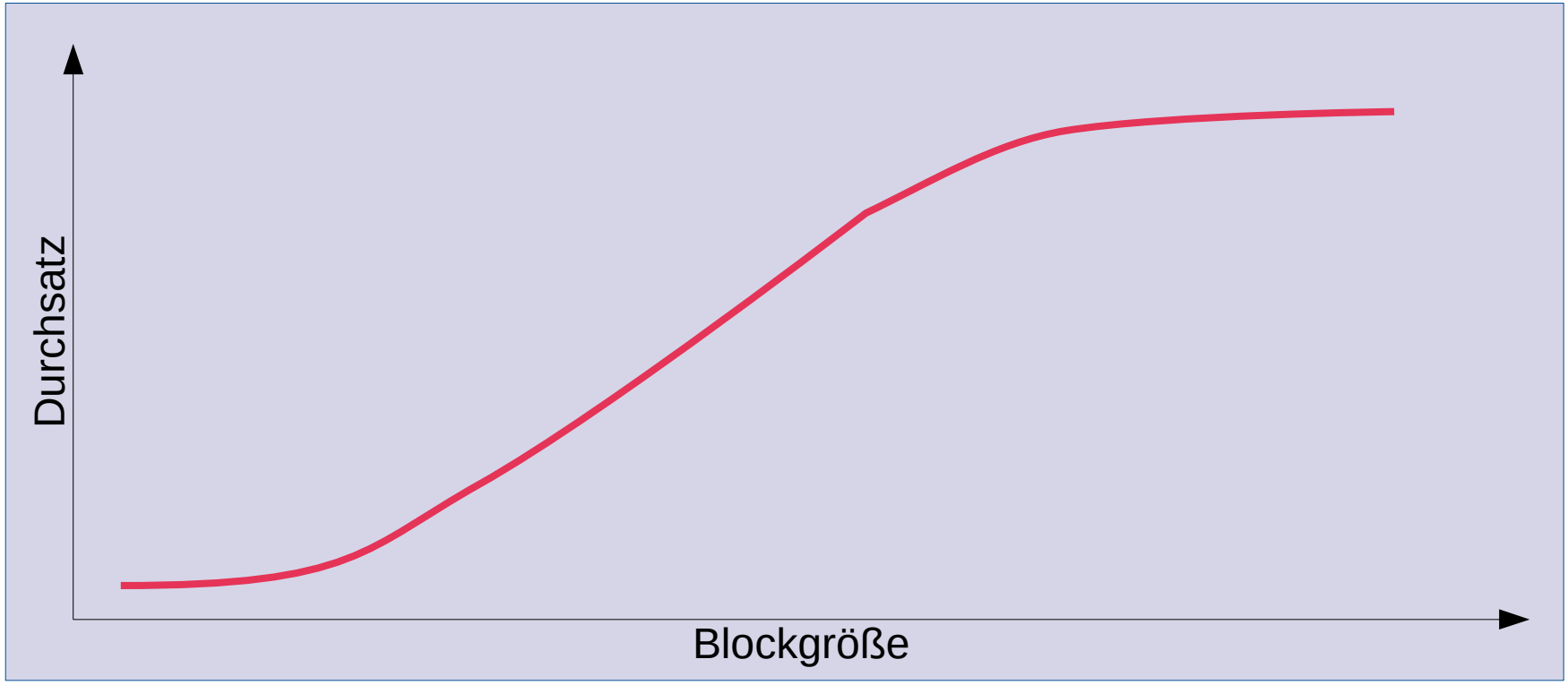
science + computing
| an atos company

- Dauerhaft benötigte Daten landen früher oder später auf Plattenlaufwerken (HDD, SSD)
- Typische Unterscheidung: Random I/O vs. Bulk I/O
- Typische Reflex: SSD für Random I/O, HDD für Bulk I/O
- Aber: Wann ist I/O „Random“

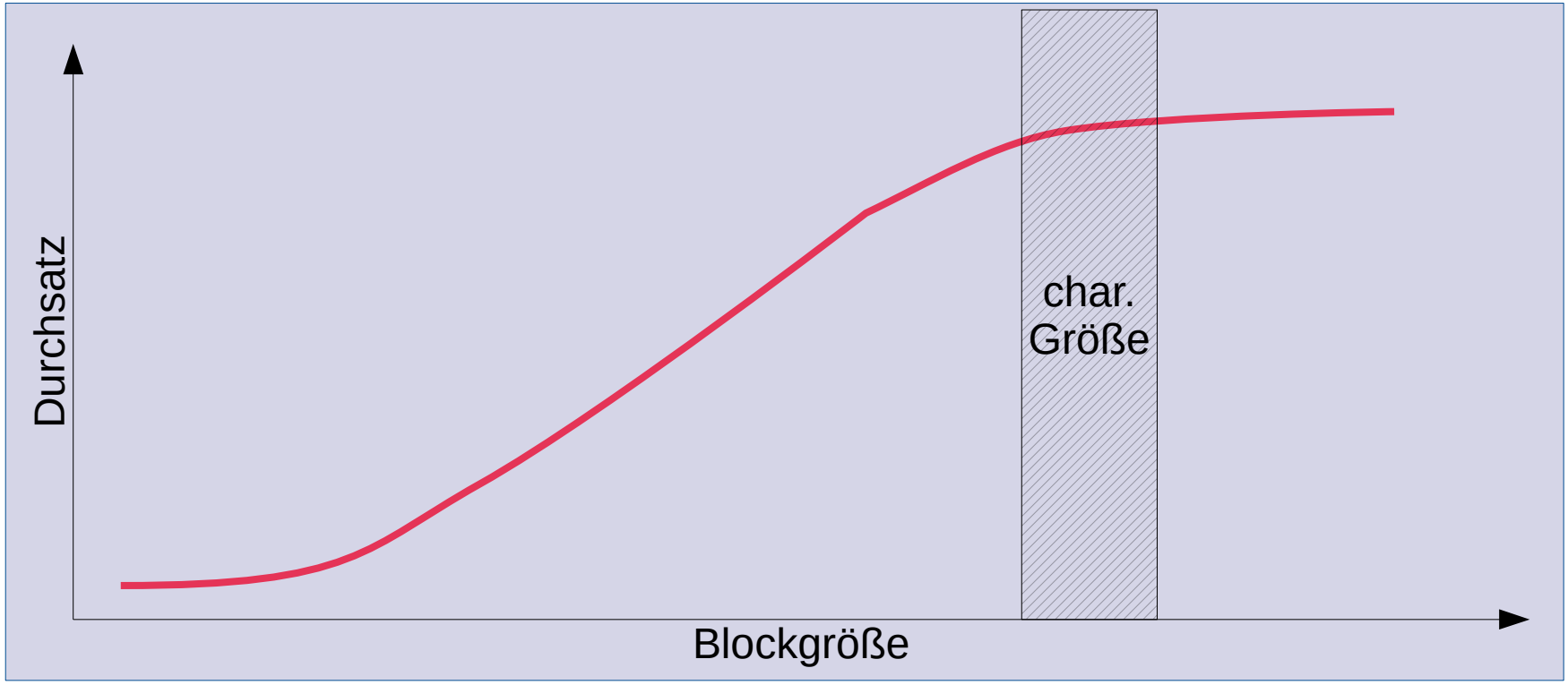
Datenträger

- Hardware-Spezifikation:
    - maximaler Durchsatz (max. MB/s)
    - Minimale Latenz bzw. maximale I/O-Rate (max. IOPS)
- IOPS beziehen sich auf nicht-zusammenhängende I/O-Operationen
- Verhältnis aus Durchsatz zu I/O-Rate gibt I/O-Größe als charakteristische Größe für optimalen Durchsatz
- Beispiel-HDD: max. Durchsatz 100 MB/s, max. 100 IOPS -> char. Größe 1 MB
- I/O kleiner char. Größe -> Laufwerke durch IOPS limitiert, kein max. Durchsatz möglich („Random I/O“)
- I/O größer char. Größe -> Laufwerk durch max. Durchsatz limitiert („Bulk I/O“)

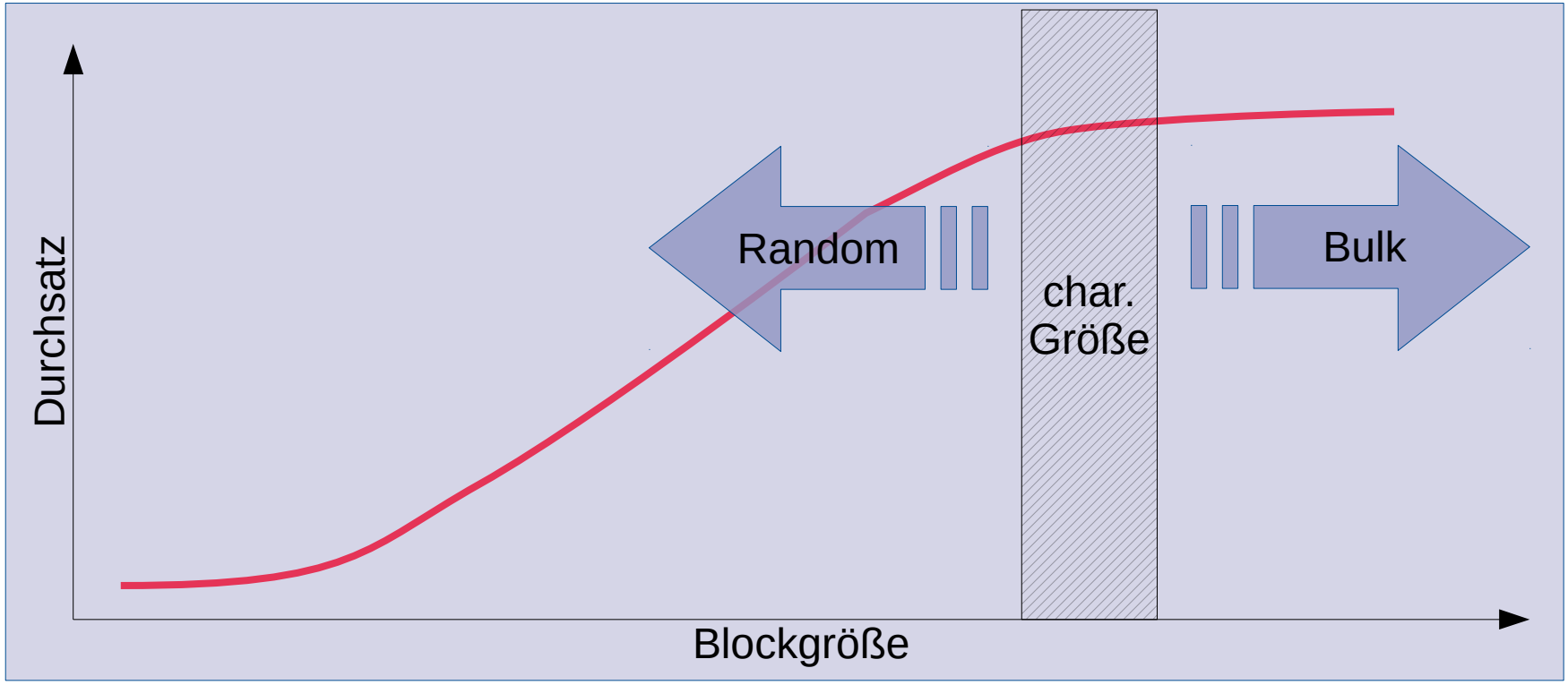
Datenträger



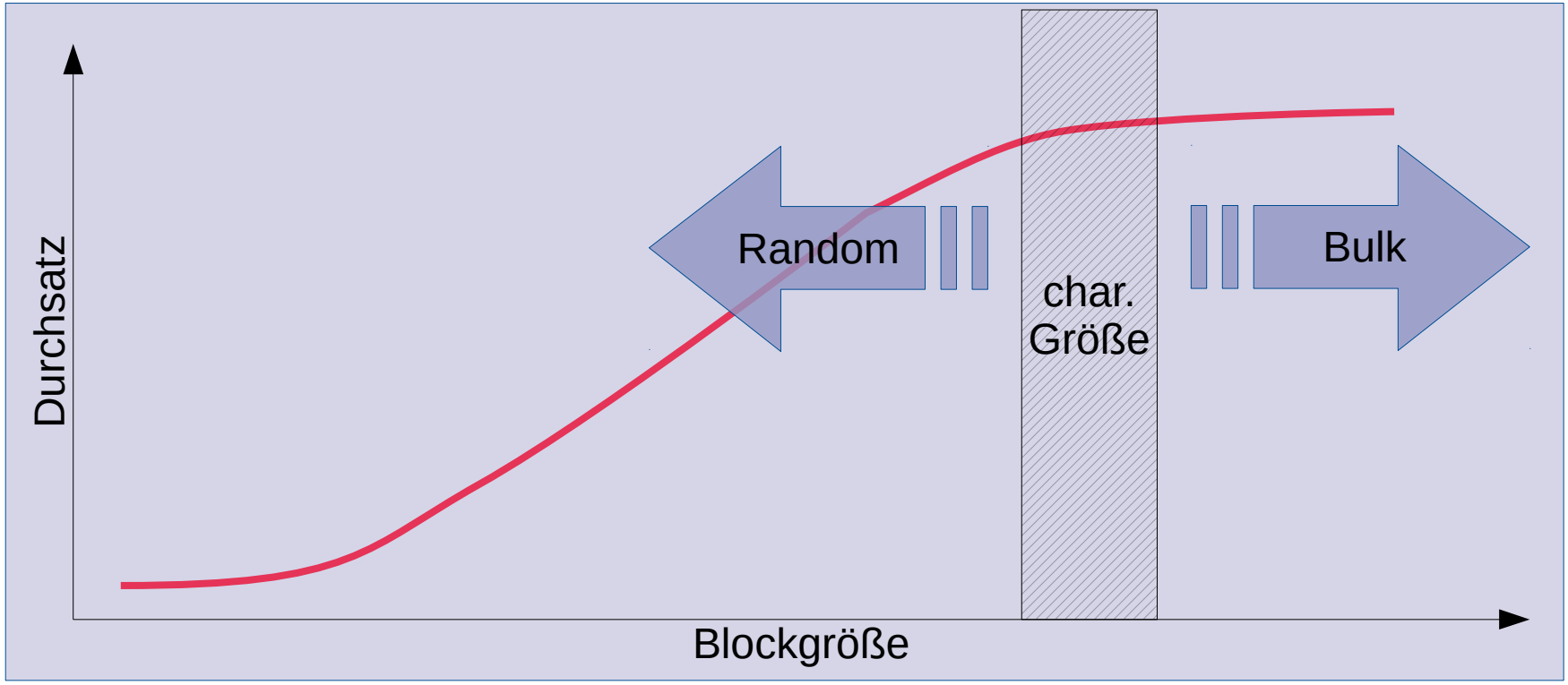
Datenträger



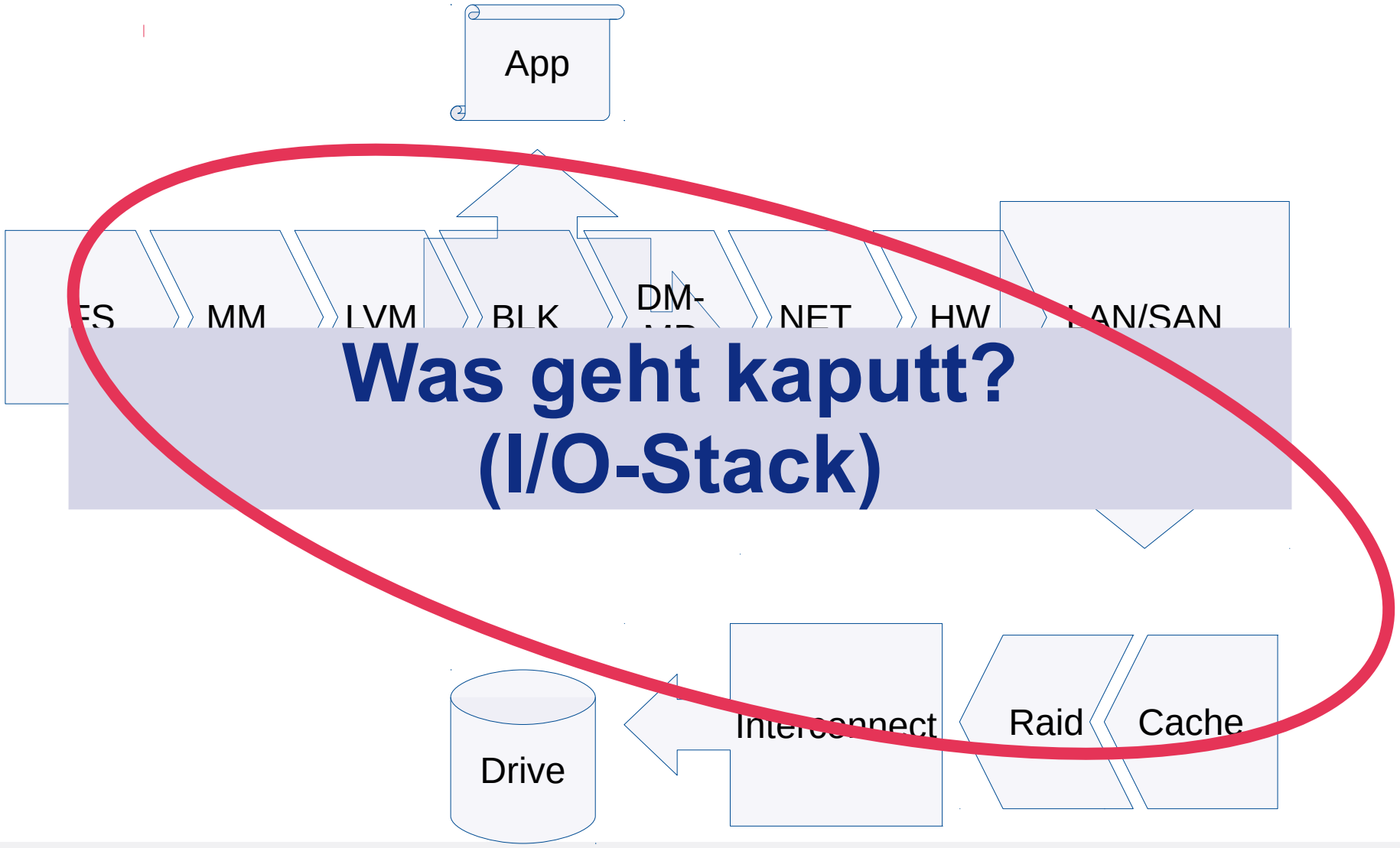
Datenträger



Datenträger



- SSD vs. HDD: Performancevorteil durch erheblich höhere IOPS -> max. Durchsatz über breiteres I/O-Spektrum erreichbar
- SSD: Unterscheidung Read/Write-Cell, Erase-Block



Was geht kaputt? (I/O-Stack)

Raid-Controller

- Datenstrom wird auf mehrere Laufwerke verteilt
- Optimaler Durchsatz, wenn I/O auf jedem Laufwerk char. Größe übersteigt
- I/O-Größe im Datenstrom >= char. Größe * #Datenplatten
- Bei kleineren I/O-Größen Wahl der Raid-Chunk-Size kritisch, um Performanceverlust zu vermeiden (Vervielfachung benötigter IOPS)
- Beispiel: Random-I/O in 1MB-Einheiten auf Raid0 aus 2 Beispiel-HDDs
    - Chunk-Size 1MB: 200 MB/s Durchsatz
    - Chunk-Size 512kB: 100 MB/s Durchsatz, da IOPS-limitiert! (aus 1MB wird 2*512kB auf HDD)

Raid-Controller

science + computing
| an atos company

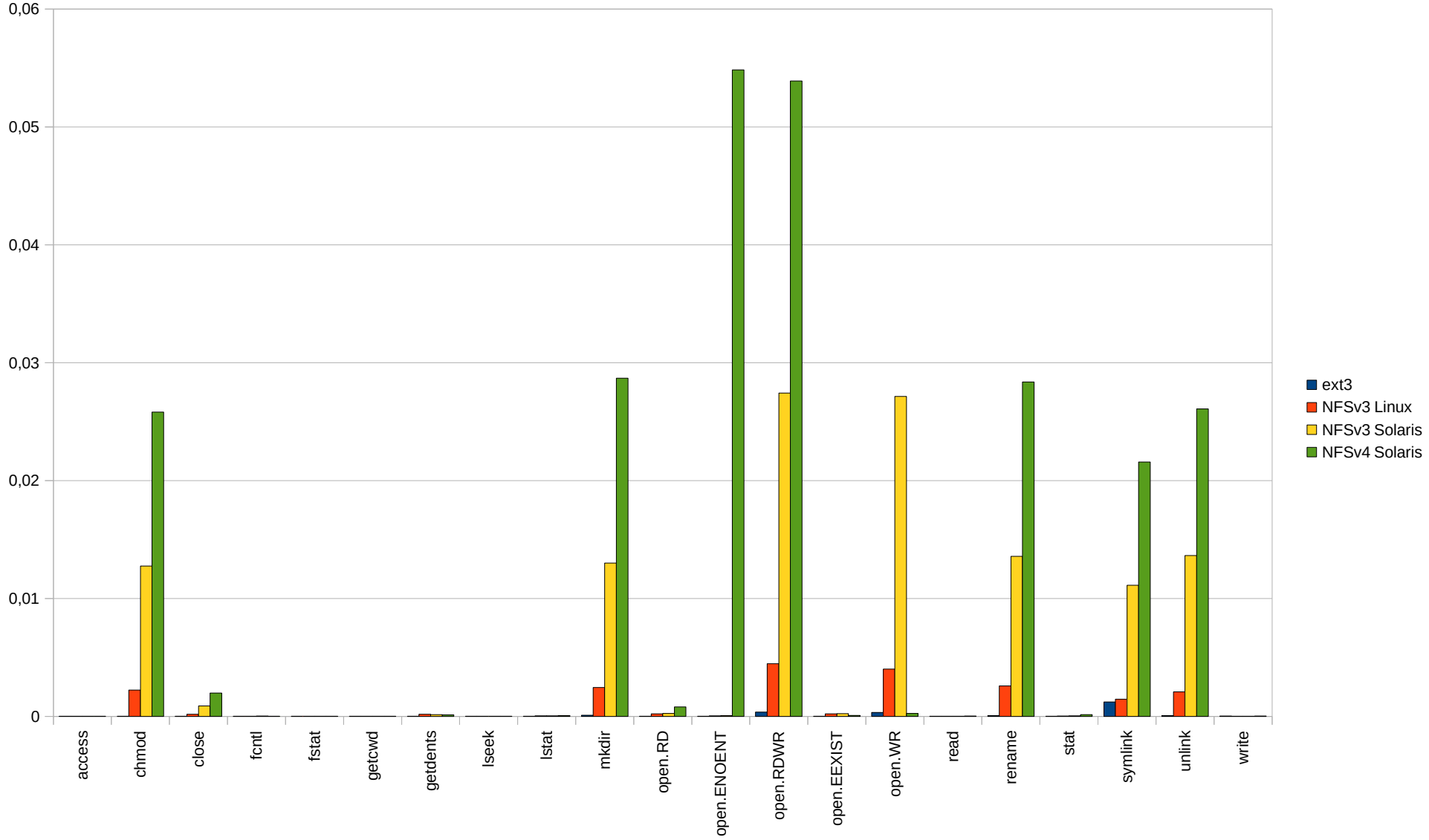
- Paritätsbasierte Raid-Level benötigen Read-Modify-Write, falls schreibende Zugriffe in Einheiten < Stripe-Größe
- Beispiel: 1MB Random-Writes auf Raid6 (8+2) -> maximale Chunk-Size 128kB
- Abstraktionsschichten für phys. Datenträger
    - Logical Volumes
    - Declustering
- I/O auf unterschiedliche LUNs konkurrieren ggf. um dieselben phys. Ressourcen
- Pagesize des Controller-Caches beeinflusst max. I/O-Größe innerhalb Controller → ggf. I/O-Verdopplung

Netzwerk

science + computing
| an atos company

- Zugriffslatenz
- Zugriffsprotokoll
- Overhead durch Hardware-Treiber/IP-Stack
- RDMA-Zugriff

Beispiel: Netzwerk-Overhead



SCSI-Subsystem/SAN

- Fragmentierung, falls I/O max. Größe für Scatter/Gather-Tabelle übersteigt
- Tuning-Parameter (Linux):
    - /sys/block/.../queue/max_segments + /sys/block/.../queue/max_segment_size
    - /sys/class/scsi_host/host.../sg_tablesize (read-only, ggf. änderbar über low-level-Treiber, eg. SAS, FC, IB-SRP)

Block-I/O

- Datenströme mehrerer Applikationen fließen zusammen
- Mehrere sequentielle Datenströme → effektiv Random I/O
- I/O-Requestgrößen, Request-Merging, Scheduler
- Tuning-Parameter:
    - Queue-Tiefe: sollte Request-Merging auf char. Größe der Datenträger ermöglichen
      (Linux: /sys/block/…/queue/nr_requests)
    - Max. I/O-Größe
      (Linux: /sys/block/…/queue/max_sectors_kb)
- Hilfsmittel: iostat -x (avg. reqsz, avg. queue depth,service time), blktrace (s. später)

Multipathing/Volume-Management

science + computing
| an atos company

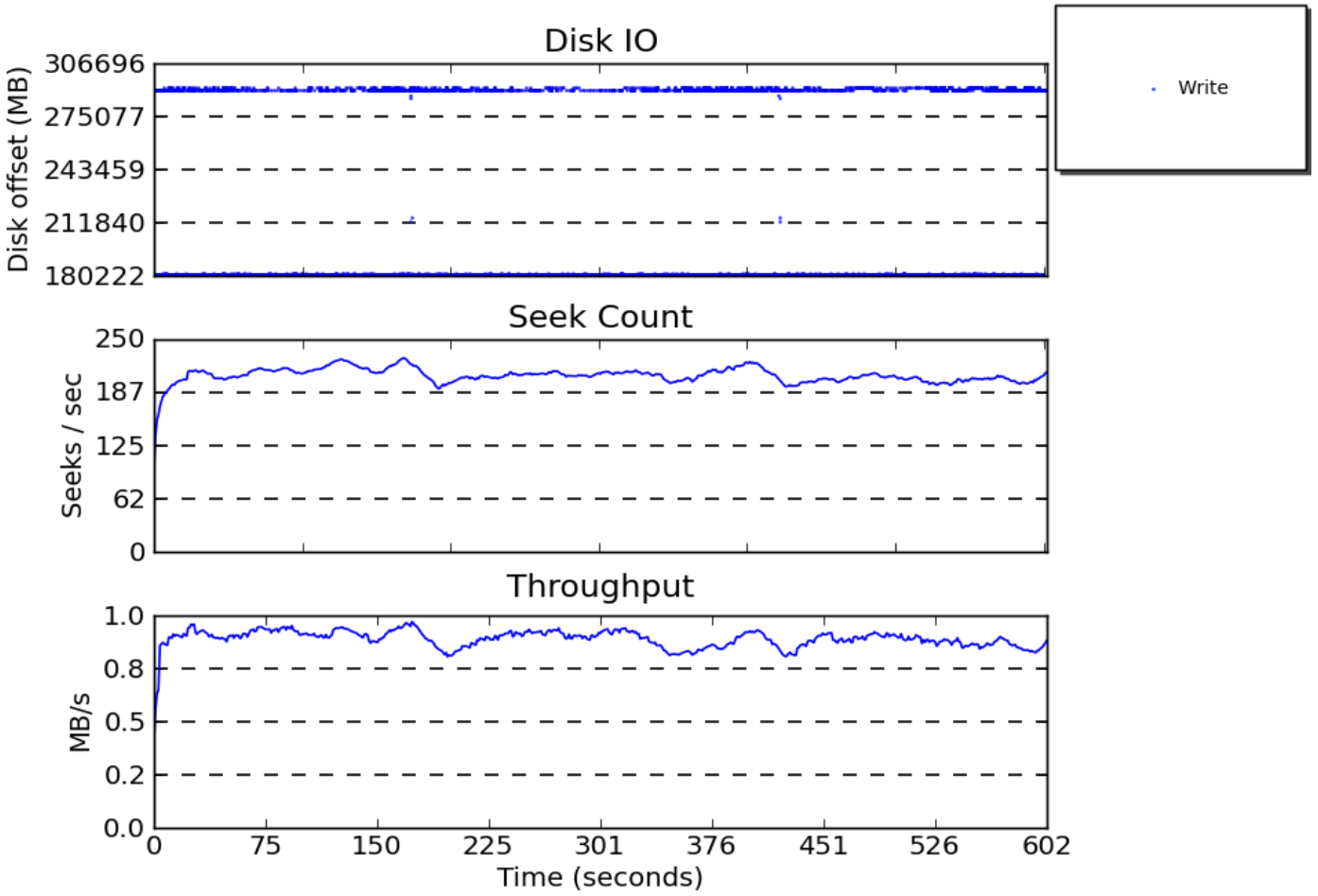
- Abstraktion logischer vs. physischer Hardware:
  - Mis-Alignment
  - Request-Vervielfachung
- Multipathing
  - Hindert ggf. Request-Merging
  - Linux: BIO (eg. RHEL5) vs. Request-based (eg. RHEL6+) dm-multipath
  - Mehrere aktive Pfade → rr_min_io möglichst klein wählen für optimale Auslastung aller Pfade

Dateisystem

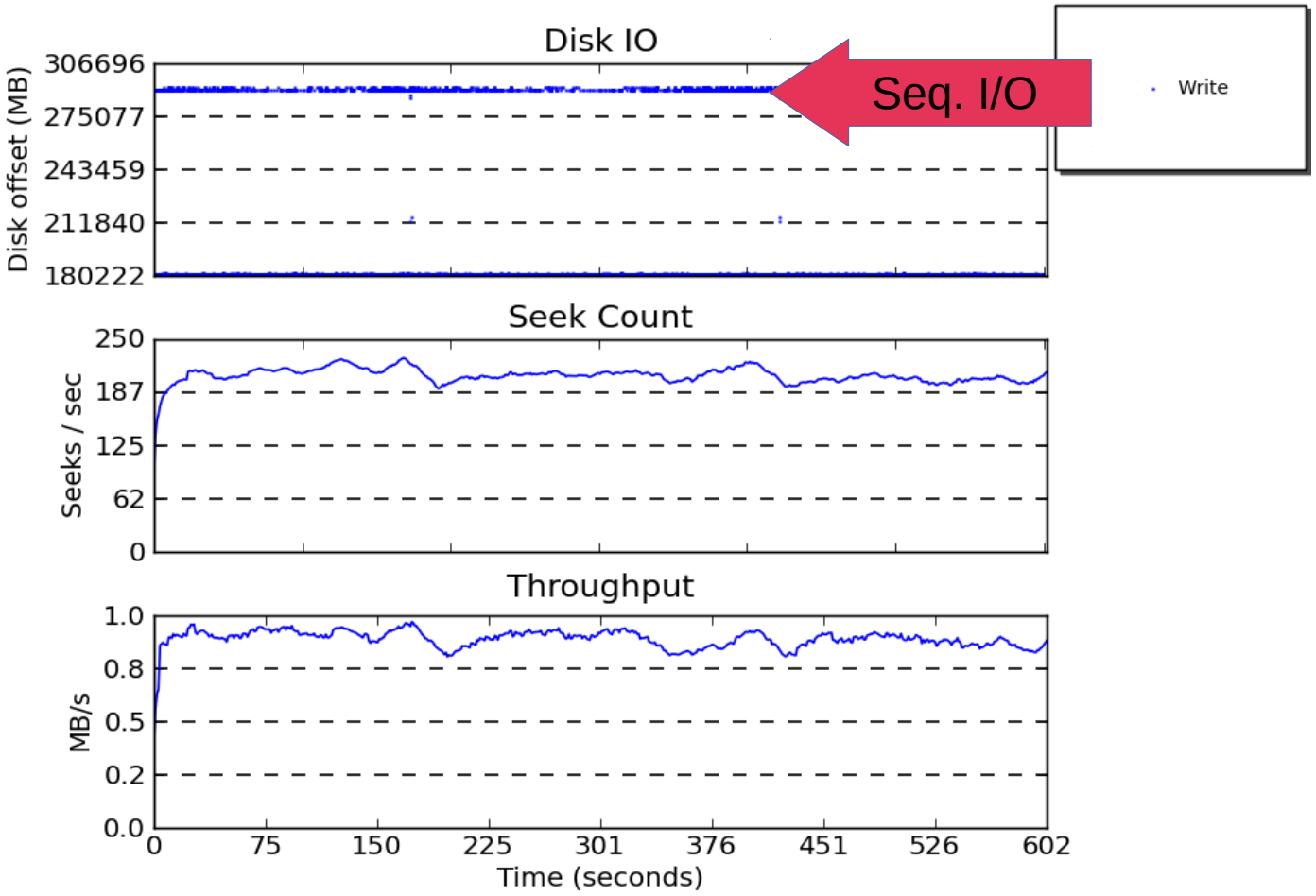
science + computing
| an atos company

- Blockgrößen
- Allokationsmuster (scattered, extents, COW)
- Füllstand
- Vervielfachung von I/O-Operationen (z.B. Metadaten-Updates, Journalling, Quota-Management)
- Evtl. erzwungene Synchronisation/Barriers
- Dateisystem-Eigenheiten (Sync on close(), Oplock-Breaks, SEEK_END)
- I/O-Größenverteilung:
    - FS-spezifische Quellen (GPFS: mmdiag --iohist, Lustre: brw_stats)
    - allgemein: über Blockdevice (eg. blktrace, seekwatcher)
- Beispiel: leere MySQL-DB auf ext4 befüllen

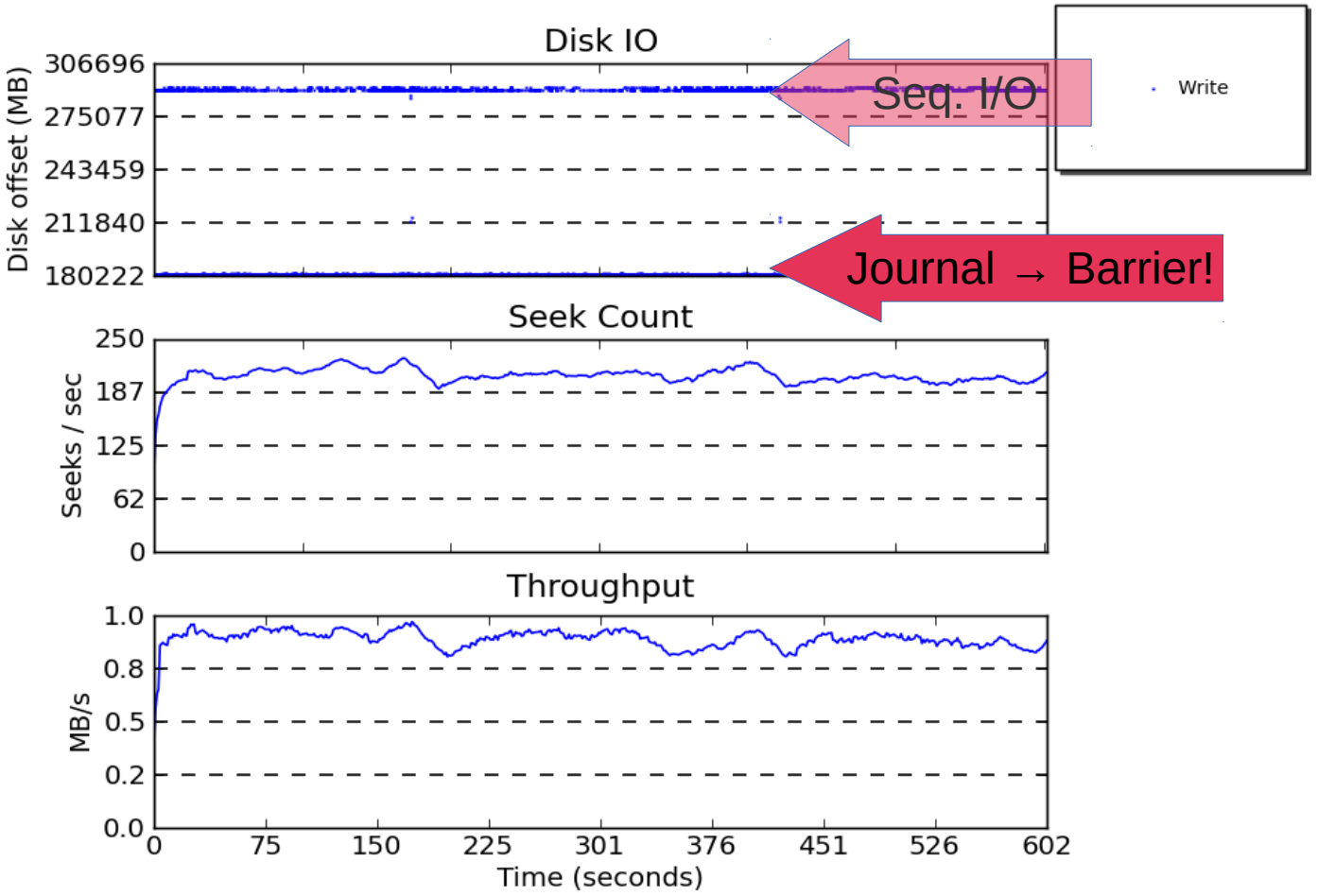
Beispiel: Seekwatcher



Beispiel: Seekwatcher



Beispiel: Seekwatcher



Speicherverwaltung

- Aktuell in der Regel NUMA-Architektur
- Diverse Tuning-Parameter (Linux: sys.vm.*)
- Hilfsmittel: slabtop, /proc/zoneinfo
- Beispiel: sys.vm.zone_reclaim_mode != 0

author	Mel Gorman <mgorman@suse.de>	2014-06-04 23:07:14 (GMT)
committer	Linus Torvalds <torvalds@linux-foundation.org>	2014-06-04 23:53:59 (GMT)
commit	4f9b16a64753d0bb607454347036dc997fd03b82 (patch)	
tree	26169e829082644a3e6ee6bd4a9991b061425c50	
parent	944d9fec8d7aee3f2e16573e9b6a16634b33f403 (diff)	

mm: disable zone_reclaim_mode by default

When it was introduced, zone_reclaim_mode made sense as NUMA distances punished and workloads were generally partitioned to fit into a NUMA node. NUMA machines are now common but few of the workloads are NUMA-aware and it's routine to see major performance degradation due to zone_reclaim_mode being enabled but relatively few can identify the problem.

Those that require zone_reclaim_mode are likely to be able to detect when it needs to be enabled and tune appropriately so lets have a sensible default for the bulk of users.

Zusammenfassung

science + computing
| an atos company

- I/O-Muster auf Datenträger entscheidet über (Langzeit-)Performance
- Auf HDD bestimmt Größe der einzelnen I/O-Operationen über den Durchsatz
- Applikation muss passende I/O-Operationen liefern
- I/O-Stack muss so angepasst werden, dass Größe der I/O-Operationen nicht verschlechtert wird
- Verfolgen der I/O-Größe über Subsysteme erlaubt Identifikation von Performance-Problemstellen

science + computing

an atos company

Vielen Dank für Ihre Aufmerksamkeit.

Daniel Kobras

science + computing ag

www.science-computing.de

Telefon 07071 9457-0

info@science-computing.de