HPC-Systeme
HPC und Storage

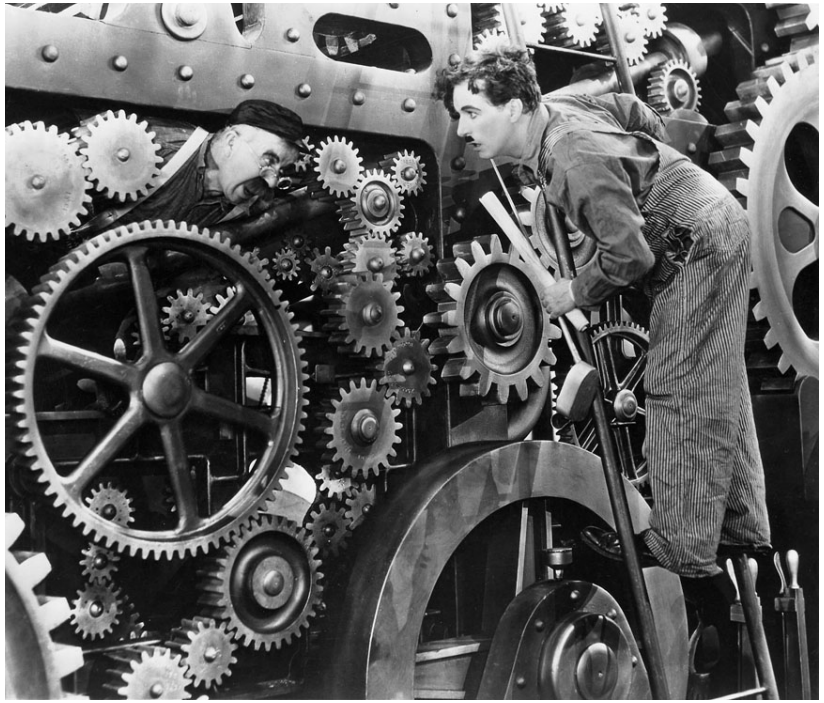
Prof. Dr. Volker Gülzow, Dr. Yves Kemp, Dr. Sergey Yakubov

SS 2018

Storage und Storage-Systeme für HPC

- Wenn man „Computing“ in HPC eng auslegt, dann betrifft dies nur das “Rechnen“
- Häufig wird Storage und Storage-Nutzung in HPC Vorlesungen (und auch Planungen...) stiefmütterlich behandelt
- Für den erfolgreichen Aufbau eines kompletten HPC-Systems ist allerdings auch vernünftiger Storage notwendig
- Für die erfolgreiche Nutzung von HPC-Systemen sind einige Kenntnisse über Storage-Systeme und Storage-Nutzung wichtig

Zwei Sichten auf Storage:



Admin-Sicht auf Storage
Erste Hälfte dieser Vorlesung

Nutzer-Sicht auf Storage
Zweite Hälfte

http://selfstorageindavis.com

http://cinemalacrum.blogspot.de/2011/08/top-ten-thursday-silent-films.html

Storage != Data

- Mit „Data“ und „Data Access“ wird im HPC Vorlesungen häufig Daten und Datenzugrif im/auf Arbeitsspeicher oder L1/2/3 Caches der CPUs gemeint.
- Die Festplatte im Server eines HPC-Workernode spielt typscherweise eine untergeordnete Rolle

- Unter Storage verstehe ich in der Vorlesung grosser, zentraler Storage

Anwendungsfall von Storage im HPC

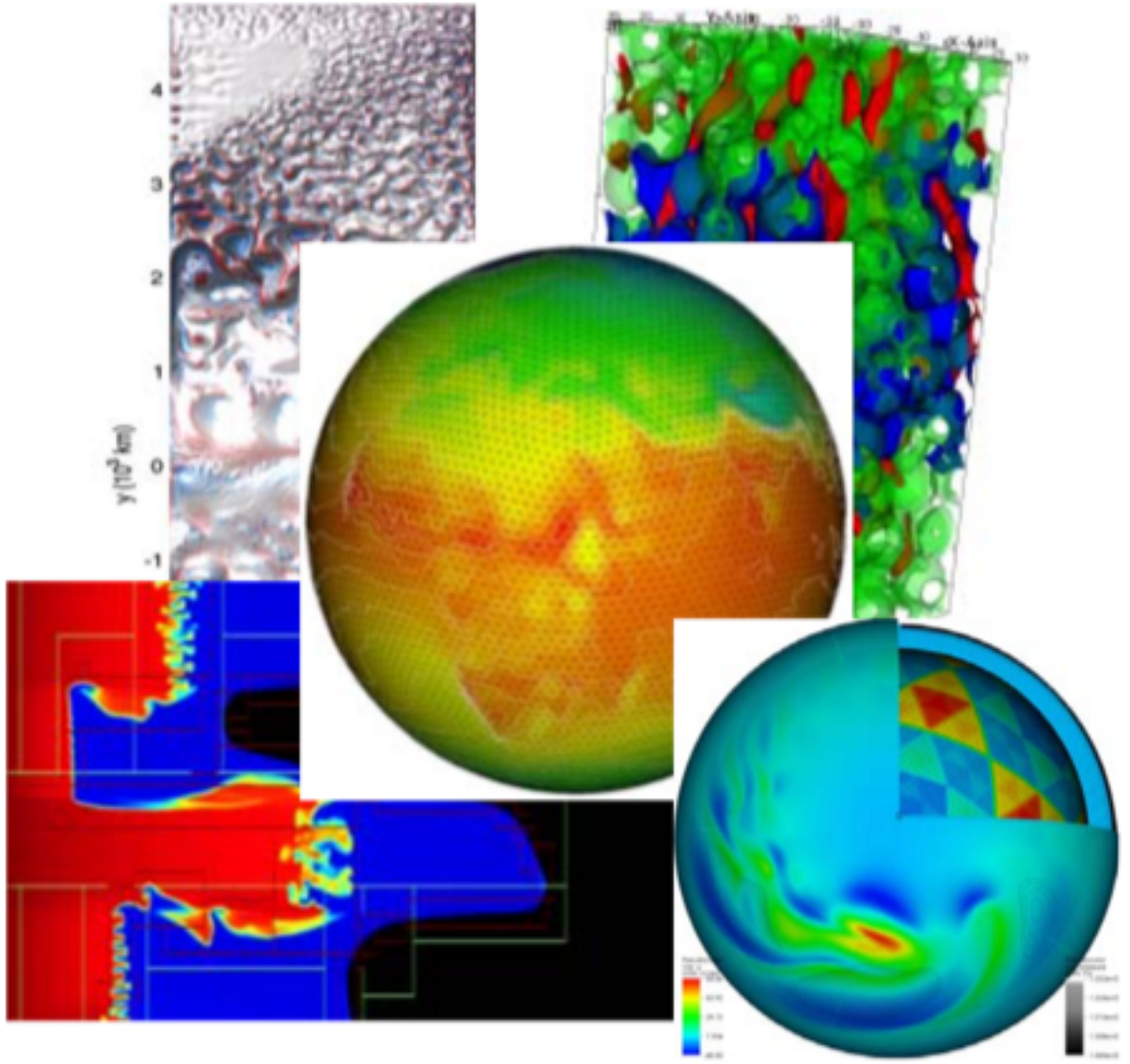
- Input-Daten für Simulation (oder Analyse)
- Lagerung von Output-Daten einer Simulation (oder Analyse)
- Projekt-Daten für Zwischenschritte
- Nutzer-“$HOME“

Anwendungsfall von Storage im HPC

- Checkpointing: Snapshot der aktuellen Berechnung auf Storage zu schreiben, um im Fall eines teilweisen oder ganzen Ausfalls vom letzten Snapshot aus weiterzurechnen

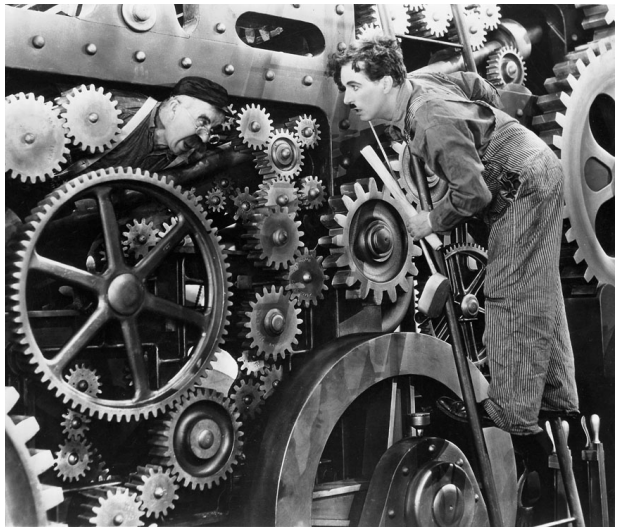
Storage & IO

- In der wissenschaftlichen Community steigen die Anforderungen an IO kontinuierlich
- IO ist ein Bottleneck für viele Nutzer
- Erforderlicher Storage-Platz muss mit System-Speicher (RAM) skalieren
- Paralleles IO ist für grosse Nutzer unerlässlich
- Typische grosse HPC Cluster haben mehre 10 PB Storage



Diese und nächste Folien mit Material von Richard Gerber (NERSC)
https://www.olcf.ornl.gov/wp-content/uploads/2013/05/OLCF-Data-Intro-IO-Gerber-FINAL.pdf

Zwei Sichten auf Storage:



Admin:
Daten? Ja, das Zeugs was von einem Worker-Node auf einen Storage-Server Transferiert wird und dort auf Speichermedien liegt

Nutzer:
Daten? Ja, Abbildung meines Systems im Code: Grid Zellen, Teilchen, ...

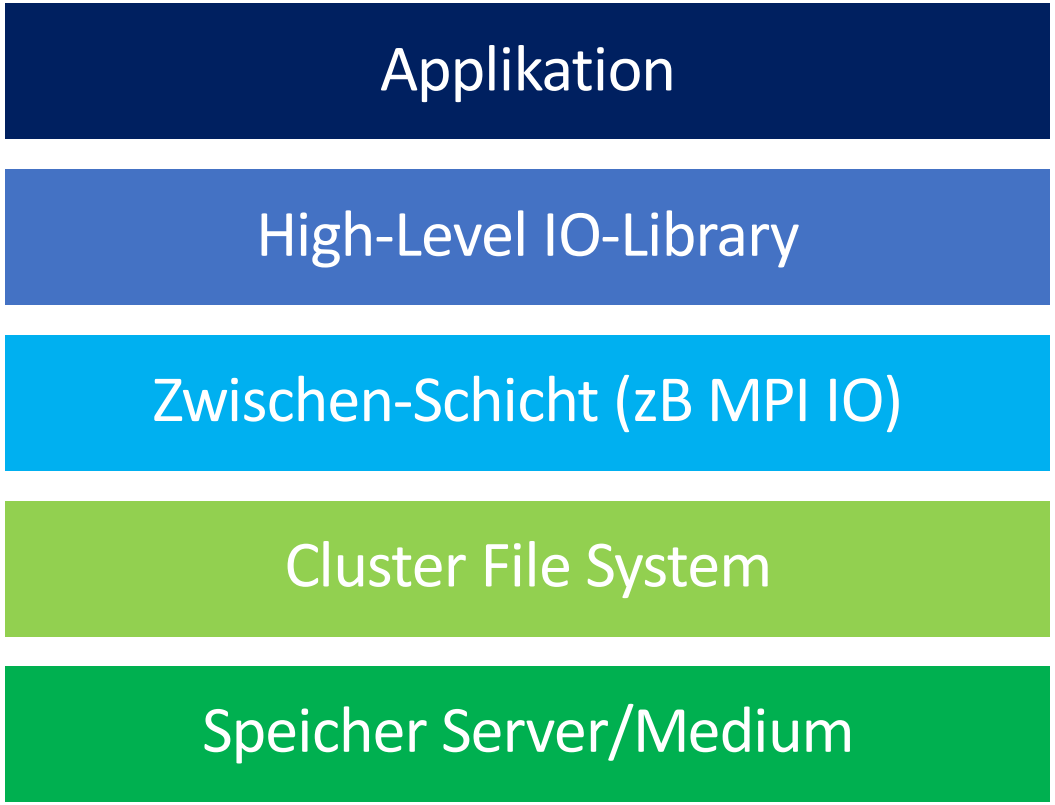
http://selfstorageindavis.com

http://cinemalacrum.blogspot.de/2011/08/top-ten-thursday-silent-films.html

Relationship: It‘s complicated

- Nutzer müssen ihre IO Muster verstehen - und ggf anpassen – um gute Performance zu erreichen
- Admins brauchen möglichst detailliertes Wissen über die Anforderungen der Nutzer um Storage-Systeme zu designen und zu tunen
- Manches kann durch spezialisierte IO-Libraries abgefedert werden

IO Hierachie



Welche Speicher Medien?

- Sehr schnell: Solid-State
  - FLASH / 3D-Xpoint (neue Technologie von Intel & Micron) / ...
  - Sehr schnell, sehr teuer, vergleichsweise wenig Kapazität
- Schnelle aber kleine rotierende Platten
  - SAS, typischerweise 2.5“ mit 10kRPM oder 15kRPM
  - 600 Gbyte – 1.8 TB aktuell typische Kapzitäten
- Nearline-SAS Platten
  - NL-SAS, 3.5“ mit 7.2kRPM
  - 8-12 Tbyte aktuell typische Kapazitäten
- Bänder
  - Langsam, gross, nur streaming IO, unhandlich, günstig (je nach Rechnung)

Welche Speicher Medien?

- Sehr schnell: Solid-State
  - In Storage-Systemen zB als Burst-Buffer, Cache oder Meta-Daten-Speicher benutzt
- Schnelle aber kleine rotierende Platten
  - Nutzung nimmt ab zugunsten von Solid-State
- Nearline-SAS Platten
  - Stellen typischerweise den Grossteil des Plattenplatzes
- Bänder
  - Eventuell Archiv oder Backup. Typischerweise nicht direkt aus dem Filesystem adressierbar

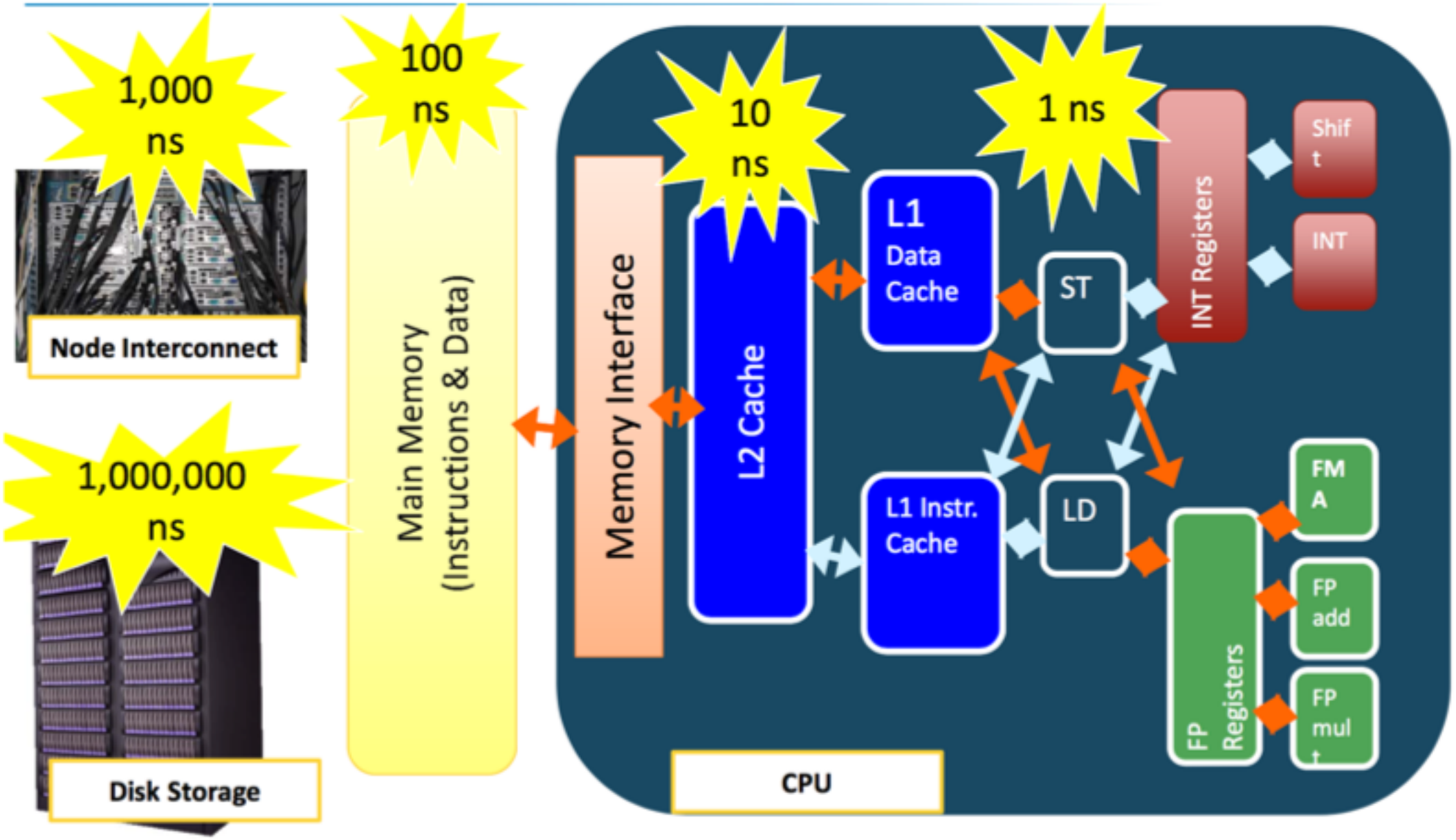
Eine Festplatte macht noch kein Storage-System

- <milchmädchenrechnung>
  - Um 10 Pbyte zu speichern braucht man aktuell 1000 x 10 Tbyte Platten
  - NL-SAS Platte hat MTBF von 1.2 Mio Stunden ... 1.2 Mio h / 1000 / 24h = Ein Fehler alle 50 Tage
  - Oder: NL-SAS Platte hat Bit Error Rate von 1/10^15 bits ~= 110 TByte. 10 Pbyte einmal vollschreiben: ~100 Bit Fehler.
- Man braucht also
  - Fehlerkorrekturen und Redundanzen
  - Etwas was 1000 Einzelschicksale in wenige, grössere Blöcke gruppiert
- (In Wirklichkeit deutlich mehr „Einzelschicksale“)

RAID und Erasure Coding

- RAID: Redundant Array of In[expensive|dependent] Disks
  - Heute Typischerweise RAID-6: n+2 Parity
  - Realisiert typischerweise über Hardware-RAID Controller
- Erasure Coding
  - RAID-6 eine spezielle Form von Erasure Coding (ggf in Hardware)
  - Aktuell viel Entwicklung um Erasure Coding in die Filesysteme zu bekommen – ohne HW-Controller!
  - Einige kommerzielle Produkte setzten dies um, OpenSource Varianten noch in den Kinderschuhen, holen aber auf

Latenzen: Zeit zum Lesen des ersten Byte



Richard Gerber

Bandbreiten

Wie schnell können Daten gestreamt werden von/zu Platte?

- N*10 bis N*100 GByte/s sind heute typisch für grosse Systeme (N*10 PB)
  - Aggregiert! Also „viele WNs reden mit vielen Servern“
  - Single-Stream ist begrenzt von Netzwerk-interface und ggf Speichermedium
- Hängt allerdings von der Applikation ab:
  - Maximale Bandbreite ist nur abrufbar wenn grosse Blöcke gelesen werden, und wenn dies im Streaming-Modus passiert!

Latenzen und Bandbreiten-Optimierung

- Schreiben: Buffering
  - Schneller Speicher sammelt Daten und schreibt sie dann (ggf. streaming modus) auf langsameren Speicher
- Lesen: Caching
  - Ein ganzer Block wird vom langsamen in den schnellen Speicher gelesen, auch wenn nur ein Teil des Blocks angefragt wurde. Der Block bleibt erstmal im Cache
- Unterschiedliche Orte: In der Nähe der CPU, im Worker-Node, dedizierter IO-Node, spezialisierte Storage-Server, zusätzliche Platte in Storage Server, im RAID-Controller oder auf der Platte selber

Local / Global aus Sicht des Worker-Node

- Local Storage: Die Festplatte (on-board)
  - Altbekannt: Für OS und /tmp. Uninteressant
  - Aber: Eventuell spezieller IO-Node mit schnellem Cache (SSD). „Local Read-Only Cache“ (LROC im IBM-GPFS Jargon)
- Cluster-Lokaler Storage:
  - Network Attached Cluster File System
  - Nur im dedizierten Cluster Interconnect (schnell) verfügbar, typischerweise nur für ein HPC System
- Globaler Storage:
  - Im Campus Netzwerk (oder sogar weltweit) verfügbar.
  - ZB Ausgangsort für Rohdaten oder permanenter Speicherort

Verschiedene Arten von Clustered Storage

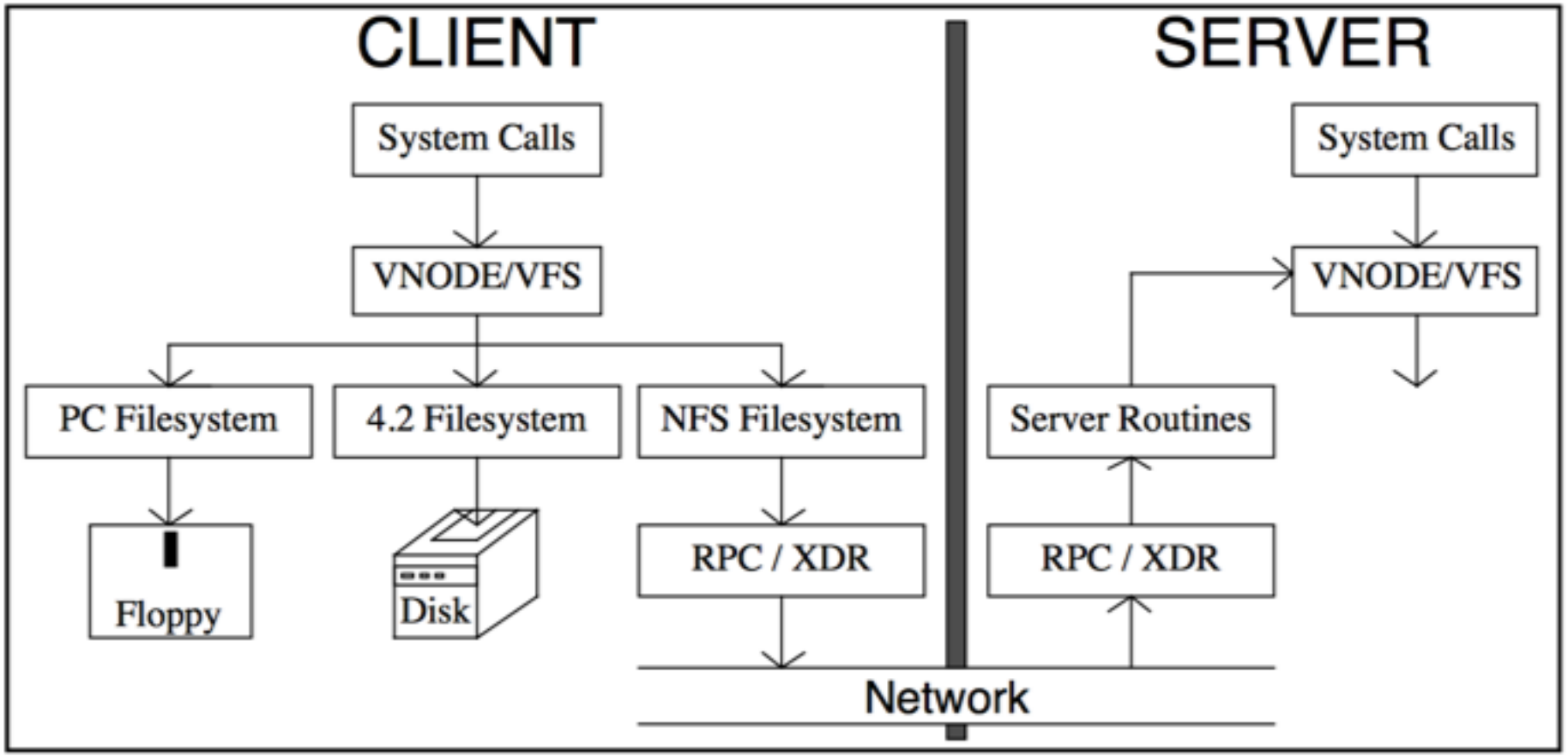
- Network attached storage
- Shared-disk FS
- Verteiltes FS

- https://en.wikipedia.org/wiki/Clustered_file_system

Network attached storage

- Typischerweise NFS (Network File System) Protokoll
  - CIFS/SMB oder Andere sind im UNIX-lastigen HPC Umfeld eigentlich nicht zu finden
- NFS ist in Linux quasi überall zu finden
  - Kernel-NFS Server, NFS Client Modul...
  - Mounten ist trivial: mount –t nfs server:/export/path /local/path
- Viele Hersteller bieten NFS Appliances an

NFS Client <-> Server Kommunikation

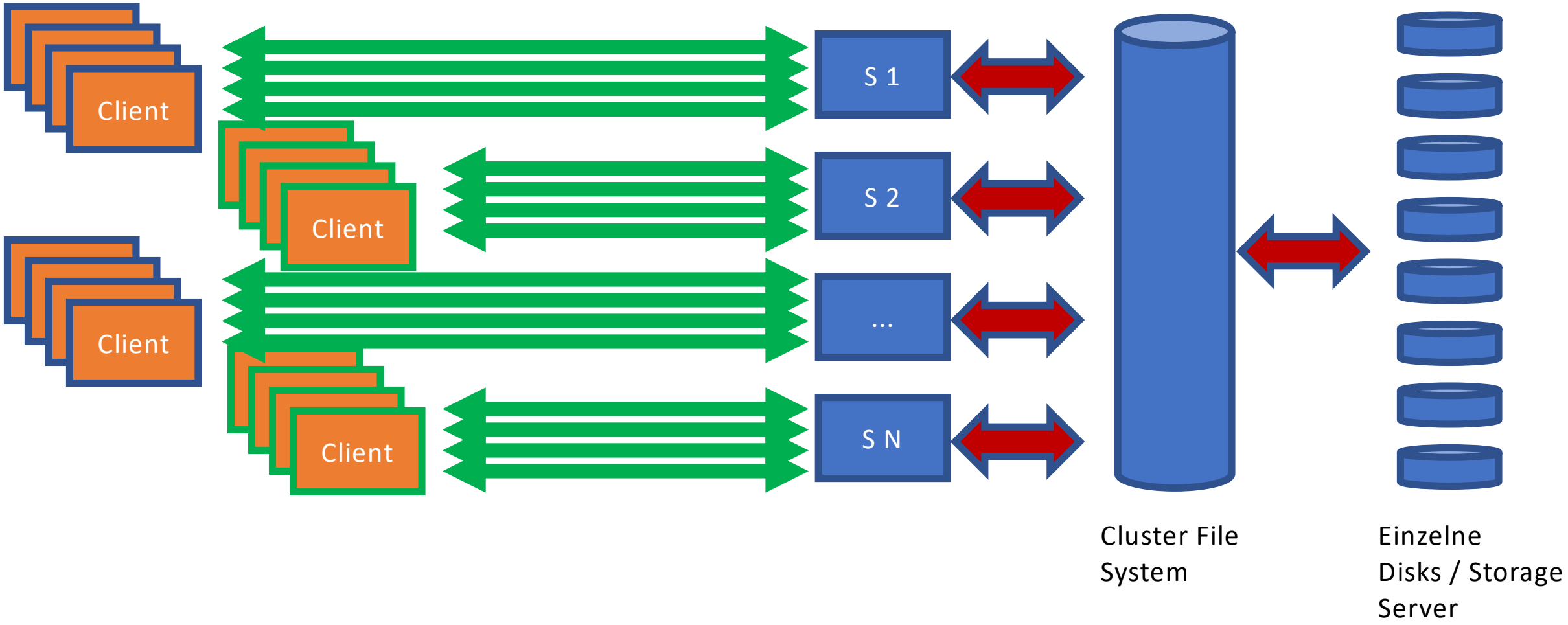


http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.7841 (Russel Sandberg) das Paper stammt von 1986 ☺

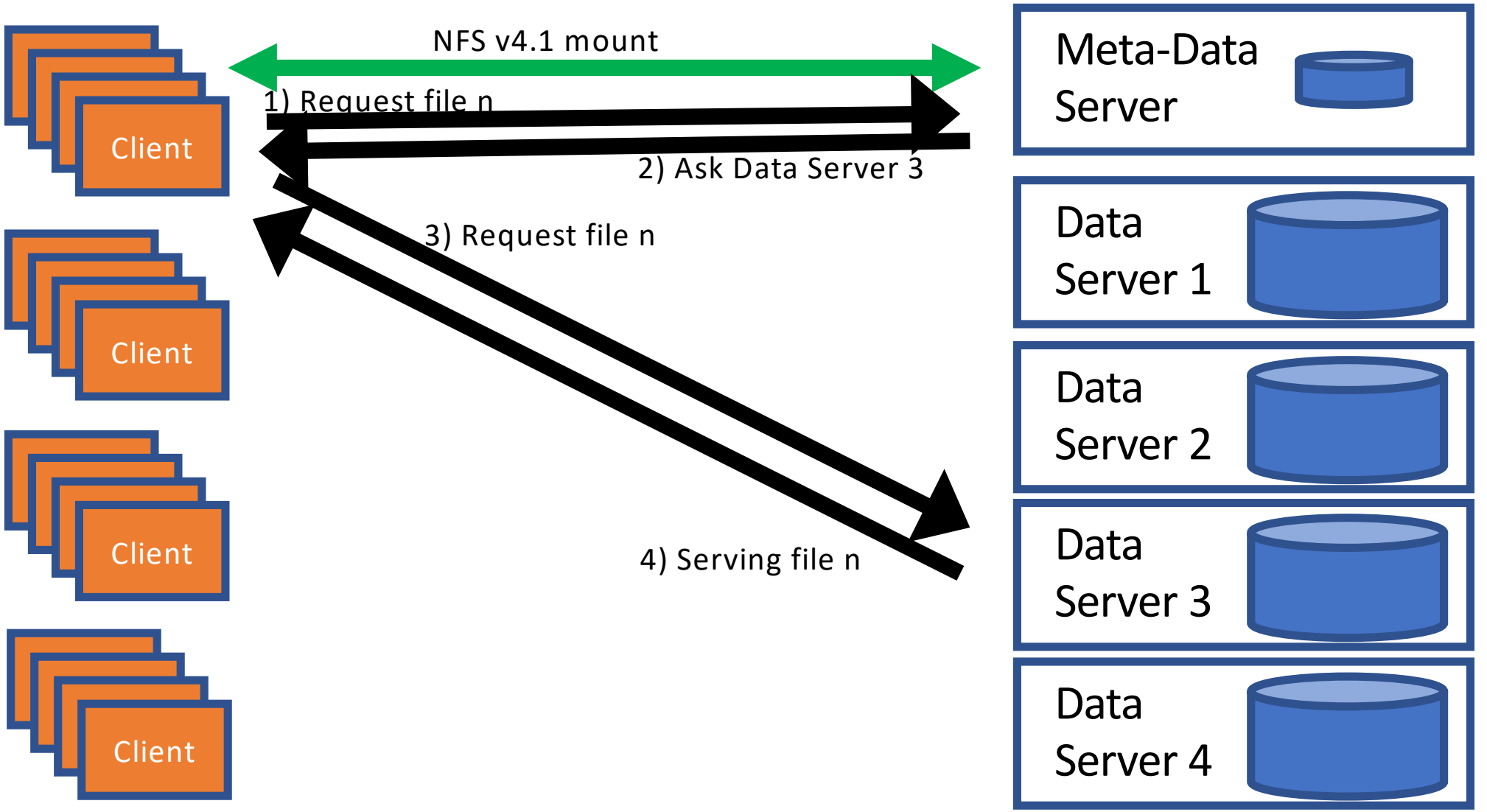
Skalierung? NFS v3 und NFS v4.0

- Die Kommunikation geht immer zwischen Client und Server
- Mehr Plattenplatz im Server
  - Bottleneck: Netzwerkanbinung Server
- Mehrere Server
  - Bandbreiten-Skalierung über #Server
  - Unterschiedliche Namespaces
    - /mnt/nfs/server1
    - …
    - /mnt/nfs/servern

Skalierung in NFS v3 und NFS v4.0, Beispiel



Skalierung mittels NFS v4.1 / pNFS



NFS v4.1/pNFS vergleichsweise neu

- Einige grosse Storage-Hersteller unterstützen NFS v4.1/pNFS
- Server haben noch nicht den Reifegrad von grossen NFS v3 und NFS v4.0 Appliances
- Client Code noch nicht so ausgereift wie v3 und v4.0
- Hersteller und Nutzer sind zögerlich

- DESY war mit NFS v4.1/pNFS Server in dCache Vorreiter
- DESY hat ca 1000 Client im Einsatz die NFS v4.1/pNFS mounten
  - Allerdings nicht im HPC Umfeld☺

NFS Security @ HPC

Security?

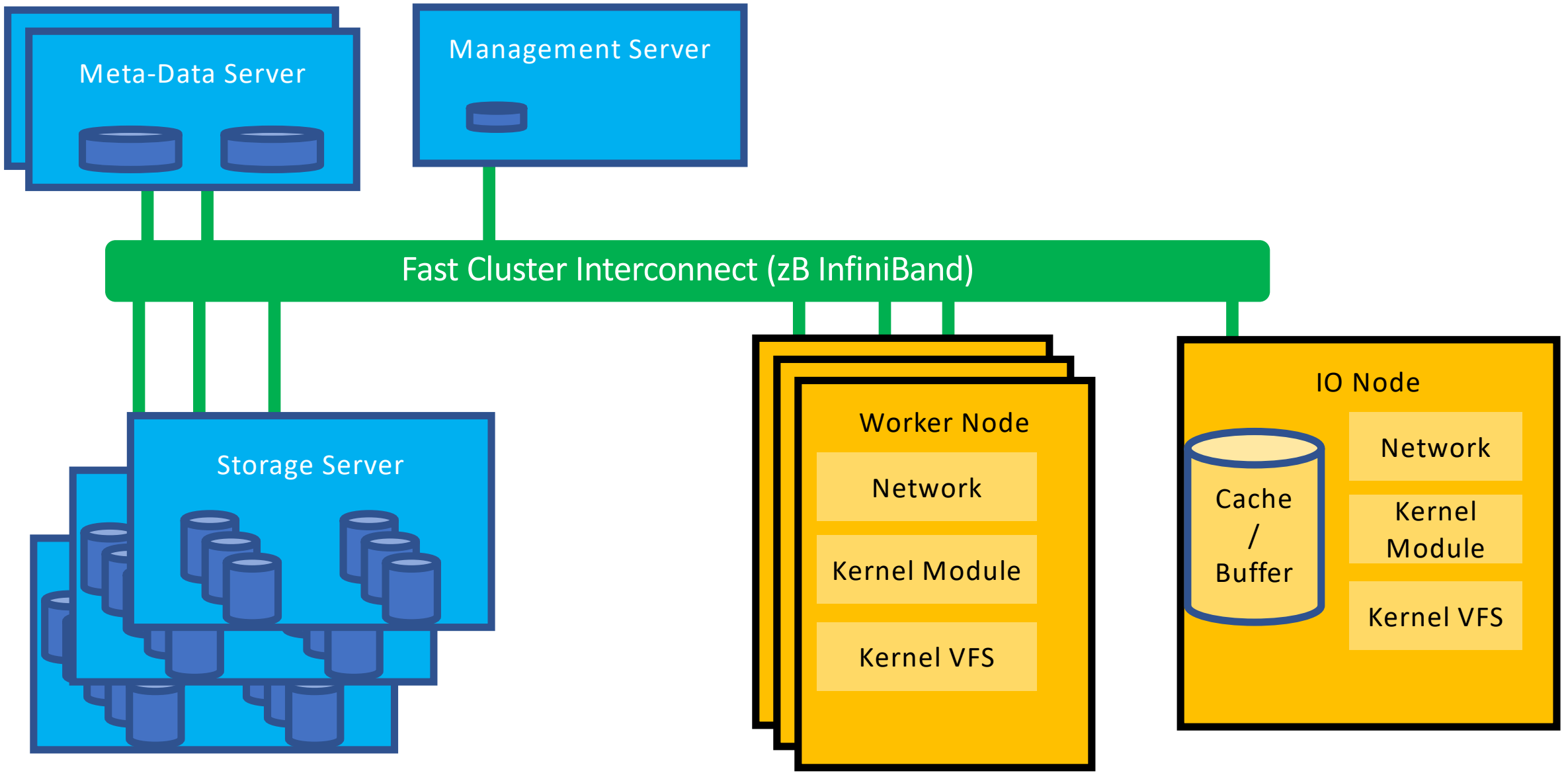
NFS Security

- OK, es gibt Kerberised NFS (mit NFS v4)
  - Macht nur keiner im RZ-Bereich (ich kenne keinen...)
- Wer darf mounten?
  - Server hat eine Liste von IP Adressen / Host-Namen von Clienten die mounten dürfen
  - HPC Cluster: IP Adressen und Host Namen sind sicher™, und können nicht gefaked werden
- Wer darf auf Daten eines Mounts zugreifen?
  - UID/GID basierende Security. Üblicherweise simple UNIX-ACLs (User/Group/Others)
  - HPC Cluster: UID/GID sind sicher™, und können nicht gefaked/missbraucht werden

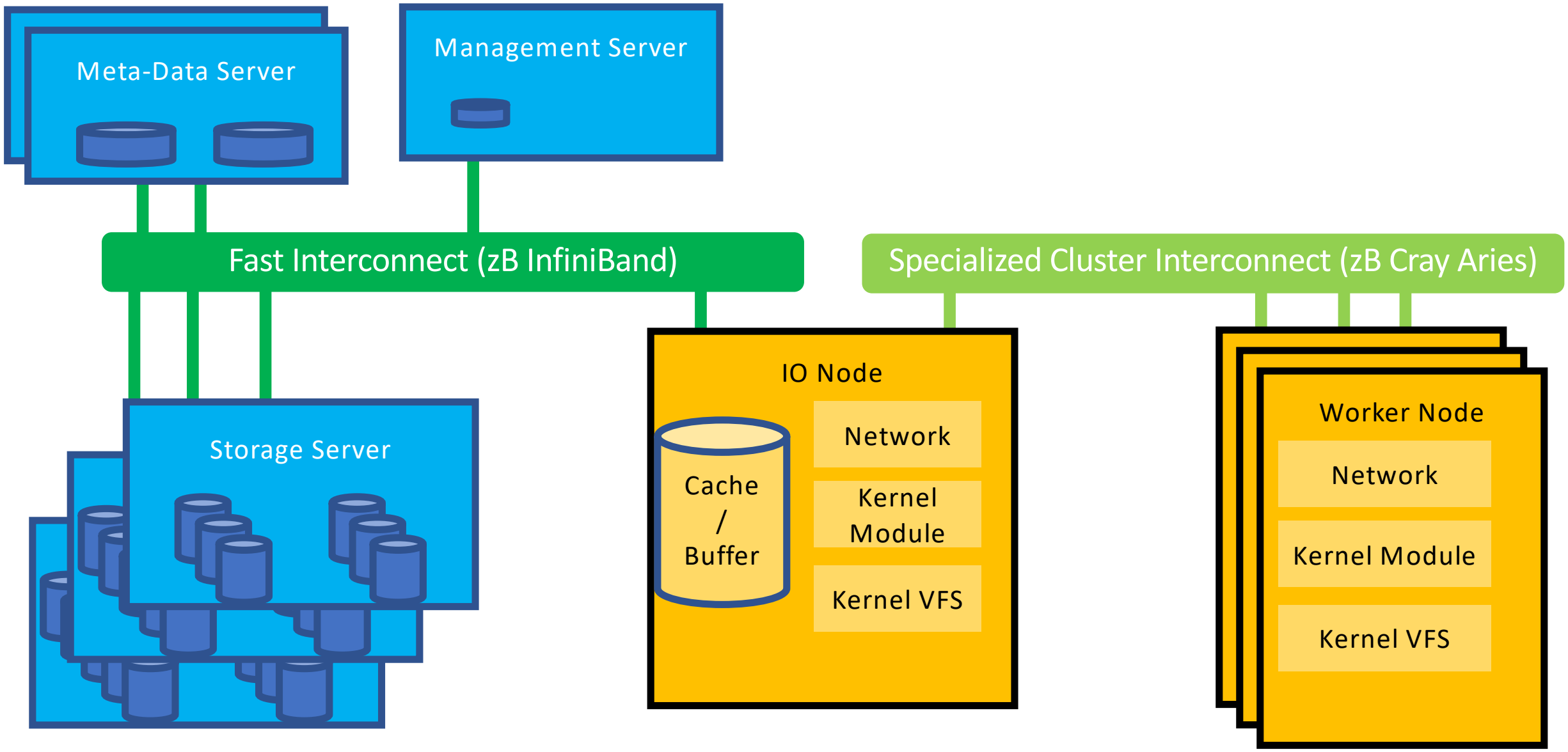
„Echte“ Cluster File Systeme

- Am DESY im Einsatz
- Lustre: HPC Storage am Standort DESY/Zeuthen
- BeeGFS: Günstiger Projekt-Space im HPC Cluster DESY/HH
- GPFS (Jetzt Spectrum Scale): Online-Datennahme der neueren DESY Experimente, schnelle Analyse

Generisches Schema von Cluster-File-System



Oder auch:


Meta-Data Server
Management Server
Fast Interconnect (zB InfiniBand)
Specialized Cluster Interconnect (zB Cray Aries)
Storage Server
IO Node
Cache / Buffer
Network
Kernel Module
Kernel VFS
Worker Node
Network
Kernel Module
Kernel VFS

Burst Buffer

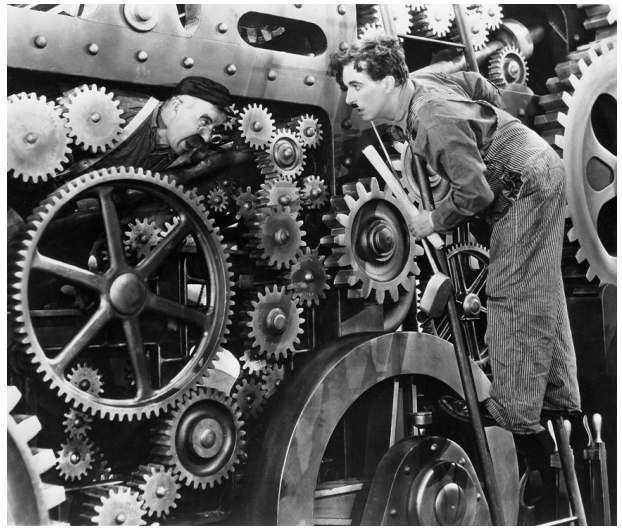
- Netzwerktopologisch sehr nahe an den Compute-Nodes
- Dadurch hohe Bandbreite, zB CORI @NERSC
  - „approximately 1.7 TB/second of peak I/O performance with 28M IOPs, and about 1.8PB of storage“
  - http://www.nersc.gov/users/computational-systems/cori/burst-buffer/burst-buffer/
- Wichtig zB für Snapshots / Checkpointing
  - Das (koordinierte) Schreiben der Snapshots auf die Burst Buffer geht sehr schnell
  - Wenn die Berechnung wieder angelaufen ist werden die Daten vom Burst Buffer auf den eigentlichen Storage geschrieben. Dies geschieht deutlich langsamer

Zusammenfassung

- IO wichtige Komponente für das Funktionieren eines HPC Clusters
- IO & Storage wird häufig vernachlässigt ... In der Ausbildung, Nutzer-Planung (und manchmal auch Cluster-Planung)
- Storage ein weites Feld
- Jetzt wissen Sie alles aus Admin-Sicht gelernt
- Gleich lernen Sie alles aus Nutzer-Sicht

"A supercomputer is a device for turning compute-bound problems into I/O-bound problems." Ken Batcher

Und jetzt die User/Entwickler-Sicht



Nutzer:
Daten? Ja, Abbildung meines Systems im Code: Grid Zellen, Teilchen, ...

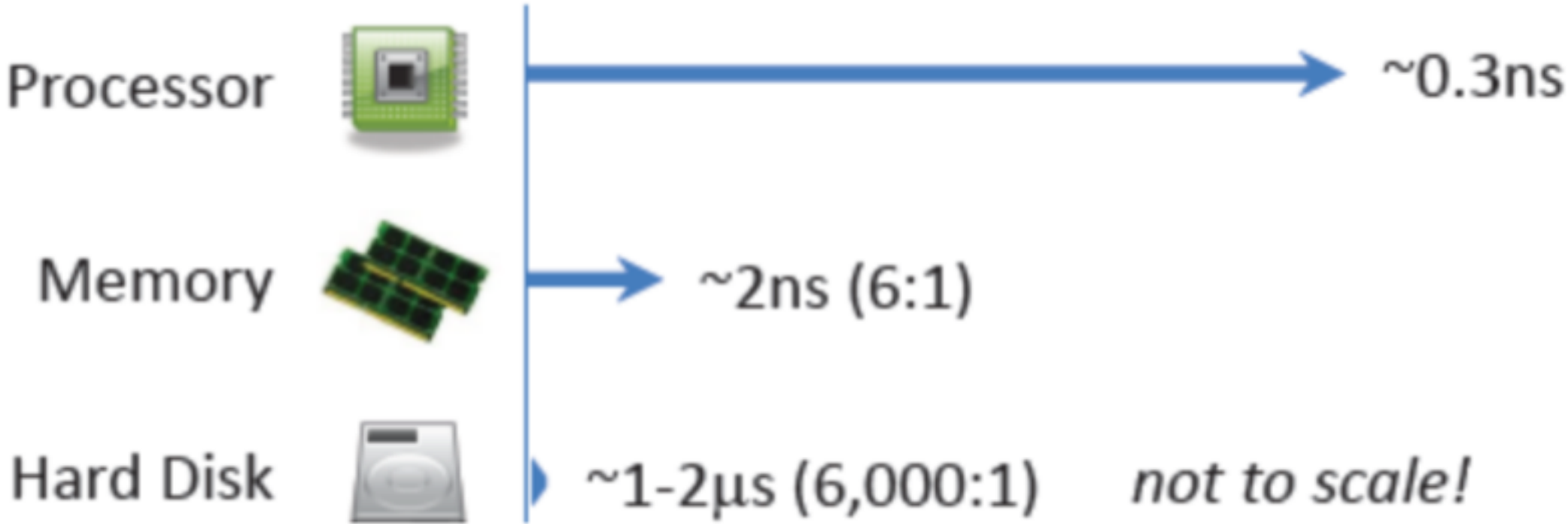
Admin:
Daten? Ja, das Zeugs was von einem Worker-Node auf einen Storage-Server Transferiert wird und dort auf Speichermedien liegt

http://selfstorageindavis.com

http://cinemalacrum.blogspot.de/2011/08/top-ten-thursday-silent-films.html

Performance

- Gesamt Laufzeit = Compute Zeit + Kommunikations Zeit + IO Zeit



https://www.tacc.utexas.edu/documents/13601/900558/MPI-IO-Final.pdf/eea9d7d3-4b81-471c-b244-41498070e35d

Welches Storage-System soll ich benutzen?

- Es gibt leider noch keine eierlegende Wollmilchsau
- Ein Beispiel: (Login-Portal des DESY Maxwell-Cluster)

```
Filesystem                                      Size  Used Avail Use% Mounted on
/dev/sda3                                        39G   27G  9.9G  73% /
devtmpfs                                        126G     0  126G   0% /dev
tmpfs                                           126G     0  126G   0% /dev/shm
tmpfs                                           126G  202M  126G   1% /run
tmpfs                                           126G     0  126G   0% /sys/fs/cgroup
itsoftware                                      1.1T  335G  720G  32% /software
max-home                                        7.9T  1.2T  6.7T  16% /home
/dev/sda6                                       9.5G  2.2G  6.9G  25% /var
/dev/sda7                                        48G   75M   46G   1% /scratch
/dev/sda5                                       9.5G   89M  8.9G   1% /tmp
/dev/sda1                                       7.6G  4.8G  2.5G  67% /var/cache/afs
tmpfs                                            26G     0   26G   0% /run/user/22640
netapp91.desy.de:/vol/ithpc                      33T   32T  736G  98% /data/netapp
AFS                                             2.0T     0  2.0T   0% /afs
beegfs_nodev                                    219T  147T   72T  68% /data/fhgfs
cxicommon                                       1.1T   33G 1022G   4% /gpfs/cfel/cxi/common
cxidata                                         601T  437T  164T  73% /gpfs/cfel/cxi/data
cxilabs                                          49T  221G   49T   1% /gpfs/cfel/cxi/labs
cxiscratch                                      201T   93T  108T  47% /gpfs/cfel/cxi/scratch
fsdslabs                                         74T   70T  4.0T  95% /gpfs/cfel/fsds/labs
ufoxcommon                                     1019G  320M 1019G   1% /gpfs/cfel/ufox/common
ufoxdata                                        9.9T  320M  9.9T   1% /gpfs/cfel/ufox/data
ufoxlabs                                        9.9T   33G  9.8T   1% /gpfs/cfel/ufox/labs
ufoxscratch                                     5.0T  320M  5.0T   1% /gpfs/cfel/ufox/scratch
exfldata                                        301T  110T  191T  37% /gpfs/exfel/data
core1                                           1.3P  1.2P  160T  88% /asap3
p3-scratch                                       11T  8.1T  2.4T  78% /gpfs/petra3/scratch
cmicommon                                       1.1T  264M  1.1T   1% /gpfs/cfel/cmi/common
cmidata                                          28T  1.7T   26T   6% /gpfs/cfel/cmi/data
cmilabs                                         9.9T  1.7T  8.2T  17% /gpfs/cfel/cmi/labs
cmiscratch                                      9.9T  3.7T  6.2T  38% /gpfs/cfel/cmi/scratch
tmpfs                                            26G     0   26G   0% /run/user/23691
dcache-dir-photon.desy.de://pnfs/desy.de/cssb   1.0E  4.7P 1020P   1% /pnfs/desy.de/cssb
```

Filesystem	Size	Used	Avail	Use%	Mounted on	
itsoftware	1.1T	335G	720G	32%	/software	**GPFS**
max-home	7.9T	1.2T	6.7T	16%	/home	**GPFS**
netapp91.desy.de:/vol/ithpc	33T	32T	736G	98%	/data/netapp NetApp	**NFS 4.0**
AFS	2.0T	0	2.0T	0%	/afs	**AFS**
beegfs_nodev	219T	147T	72T	68%	/data/fhgfs	**BeeGFS**
cxicommon	1.1T	33G	1022G	4%	/**gpfs**/cfel/cxi/common	
cxidata	601T	437T	164T	73%	/**gpfs**/cfel/cxi/data	
cxilabs	49T	221G	49T	1%	/**gpfs**/cfel/cxi/labs	
cxiscratch	201T	93T	108T	47%	/**gpfs**/cfel/cxi/scratch	
fsdslabs	74T	70T	4.0T	95%	/**gpfs**/cfel/fsds/labs	
ufoxcommon	1019G	320M	1019G	1%	/**gpfs**/cfel/ufox/common	
ufoxdata	9.9T	320M	9.9T	1%	/**gpfs**/cfel/ufox/data	
ufoxlabs	9.9T	33G	9.8T	1%	/**gpfs**/cfel/ufox/labs	
ufoxscratch	5.0T	320M	5.0T	1%	/**gpfs**/cfel/ufox/scratch	
exfldata	301T	110T	191T	37%	/**gpfs**/exfel/data	
core1	1.3P	1.2P	160T	88%	/asap3	**GPFS**
p3-scratch	11T	8.1T	2.4T	78%	/**gpfs**/petra3/scratch	
cmicommon	1.1T	264M	1.1T	1%	/**gpfs**/cfel/cmi/common	
cmidata	28T	1.7T	26T	6%	/**gpf**s/cfel/cmi/data	
cmilabs	9.9T	1.7T	8.2T	17%	/**gpfs**/cfel/cmi/labs	
cmiscratch	9.9T	3.7T	6.2T	38%	/**gpfs**/cfel/cmi/scratch	
dcache-dir-photon.desy.de://pnfs/desy.de/cssb	1.0E	4.7P	1020P	1%	/pnfs/desy.de/cssb	**dCache NFS 4.1 mount**

Wieso so viele?

- Storage-Systemen unterscheiden sich:
  - In den Zugangsprotokollen (NFS, AFS, GPFS, ...)
  - In der verfügbaren Grösse (erstmal eine Deployment-Sache
  - Im Quality of Service (Scratch, Mit Backup, ...)
  - In der Verfügbarkeit (Cluster-Lokal, Campus, Weltweit)
  - In der Semantik (POSIX etc.)
  - Performance (Schnell, Bandbreite, Throughput, Metadaten...)
  - Vom Besitzer (Darf ich überhaupt drauf schreiben?)
  - ...
- Sehr verwirrend für den Nutzer der von zuhause nur /home/$USER/ kennt

Zugansprotokolle

- ... Am Ende alles (mehr oder weniger) Posix
- Wenn es denn gemountet ist.

- Das Zugangsprotokoll ist erstmal was für den Admin
  - Aber Sie als Nutzer sollten das nicht aus den Augen verlieren

- Manche Zugangsprotokolle haben inhärente Einschränkungen

POSIX ?

- Portable Operating System Interface
- Basis-Definitionen
  - Eine Liste der im Standard benutzten Konventionen, Definitionen und Konzepte
- System-Schnittstelle
  - Die C-Systemaufrufe und dazugehörige Header-Dateien
- Kommandozeileninterpreter und Hilfsprogramme
  - Eine Liste der Hilfsprogramme und der Kommandozeileninterpreter
- Erklärungen über den Standard
- Ein *Betriebsystem* ist POSIX compliant (oder auch nicht)
  - Linux wurde nie offiziell zertifiziert, hält sich aber weitestgehend an den Standard

POSIX Filesystem Semantics ... Unter anderem:

- Hierarchische Dateinamen
- umask/unix permissions, 3 verschiendene filetimes (c/m/a)
- Umbenennung auf dem gleichen Filesystem atomar
- fsync()/dirfsync() Dauerhaftigkeit … auch auf Netzwerk-FS!
- Read-Modify-Write
- Unterschiedliche Locking-Methoden

Hierarchische Filesysteme:

- Sind praktisch
  - Einfach: Menschen ticken so
- Sind unpraktisch
  - Namespace bestimmt Organisation
  - Nur *eine* Metadaten-Kategorisierung

Metadaten Problematik

- MacBook Pro mit SSD
  - $HOME lokal

```
$ time find $HOME -type f | wc -l
  381714
real 0m6.430s
user 0m0.330s
sys 0m3.024s
```

- Fetter Server
  - $HOME auf AFS, einem Netzwerk-Filesystem

```
$ time find $HOME -type f | wc -l
311721

real 2m17.338s
user 0m0.806s
sys 0m23.707s
```

Lösung Object-Store?

- Objekte (Trivialste Vorgehensweise: 1 File = 1 Objekt)
- Separate Meta-Daten
  - Mit DB-artigen Abfrage-Möglichkeiten
- Kein Mount ins Filesystem
  - FUSE ?
- Zugriff direkt aus der Applikation

- ?

Das Problem: stat() übers Netzwerk

- Latenzen sind tödlich!

- Also: Vermeiden Sie Metadaten-Operationen!
- find / ls –R / ...
- Compilieren in vielen Verzeichnissen / Suchpfade mit vielen Verzeichnissen

- Sie wollen wissen was die Applikation macht? Zb `strace -tt -T -p <PID>`

Ein Negativ-Beispiel aus der Praxis

11:03:25.098793 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasOffline/14.2.25") = 0 <0.001753>
11:03:25.101416 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasOffline/14.2.25/External") = -1 ENOENT (No such file or directory) <0.000849>
11:03:25.103024 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasAnalysis/14.2.25") = 0 <0.000680>
11:03:25.104636 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasAnalysis/14.2.25/External") = 0 <0.001953>
11:03:25.107471 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasAnalysis/14.2.25/External/AtlasCORAL") = -1 ENOENT (No such file or directory) <0.000991>
11:03:25.109975 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasTrigger/14.2.25") = 0 <0.001485>
11:03:25.112306 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasTrigger/14.2.25/External") = 0 <0.002253>
11:03:25.115346 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasTrigger/14.2.25/External/AtlasCORAL") = -1 ENOENT (No such file or directory) <0.001435>
11:03:25.118019 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasReconstruction/14.2.25") = 0 <0.000606>
11:03:25.119216 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasReconstruction/14.2.25/External") = 0 <0.001146>
11:03:25.121424 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasReconstruction/14.2.25/External/AtlasCORAL") = -1 ENOENT (No such file or directory) <0.001702>
11:03:25.123925 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasEvent/14.2.25") = 0 <0.001196>
11:03:25.125492 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasEvent/14.2.25/External") = 0 <0.001503>
11:03:25.127930 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasEvent/14.2.25/External/AtlasCORAL") = -1 ENOENT (No such file or directory) <0.001342>
11:03:25.130174 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasConditions/14.2.25") = 0 <0.001354>
11:03:25.132187 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasConditions/14.2.25/External") = 0 <0.000540>
11:03:25.134295 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasConditions/14.2.25/External/AtlasCORAL") = -1 ENOENT (No such file or directory) <0.001286>
11:03:25.136193 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasCore/14.2.25") = 0 <0.001591>
11:03:25.138768 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasCore/14.2.25/External") = 0 <0.001081>
11:03:25.140819 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/AtlasCore/14.2.25/External/AtlasCORAL") = 0 <0.000918>
11:03:25.142627 stat("cmt/requirements", {st_mode=S_IFREG|0755, st_size=292, ...}) = 0 <0.000318>
11:03:25.143949 chdir("/afs/naf.desy.de/group/atlas/software/kits/14.2.25/**AtlasCore/14.2.25/External/AtlasCORAL/cmt**") = 0 <0.002168>

Verzeichnisse sind auf nur Dateien

- Dateien innerhalb eines Verzeichnisses sind über das Verzeichnis gekoppelt
- Das Filesystem muss Methoden bereithalten um das Verzeichnis immer aktuell zu halten
- Die funktionieren nicht gut wenn unterschiedliche Clienten gleichzeitig den Inhalt eines Verzeichnisses modifizieren

Verzeichnisse

Schlecht:

```
Server 1:
> ./job.exe > /nfs/dir/file1
Server 2:
> ./job.exe > /nfs/dir/file2
```

Besser:

```
Server 1:
> ./job.exe > /nfs/dir1/file1
Server 2:
> ./job.exe > /nfs/dir2/file2
```

Noch besser:

```
Server 1:
> ./job.exe > /local/file1
> mv /local/file1 /nfs/dir(1)/file1
Server 2:
> ./job.exe > /local/file2
> mv /local/file2 /nfs/dir(2)/file2
```

Streaming vs. Random Access

- Random Access auf der Platte: Der Kopf muss ständig neu positioniert werden
- Random Access über das Netzwerk: Latenzen kommen hinzu

- Cache (File-System-Cache auf Client oder Server, spezialisierte Caches auf dem Server) können das Problem lindern
- Besser: Sie machen möglichst viel Streaming!

Striping

- Cluster-Filesysteme stripen typischerweise Dateien auf mehrer Platten/Server (quasi RAID-0)
- Manchmal kann dies vom Nutzer speziell konfiguriert werden
  - zB Lustre, auf Verzeichnis-Basis
- Wenn nur wenige Prozesse parallel laufen kann dies von Vorteil sein
- Wenn viele unterschiedliche Prozesse laufen kann dies von Nachteil sein

- In nullter Näherung sollten Sie der Einstellung des Admins vertrauen

Staging von Input/Output-Dateien

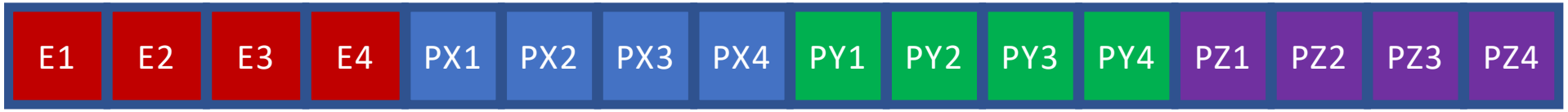
- Kopieren vor/nach dem Job
- Bei Input-Dateien wird eventuell mehr kopiert als unbedingt benötigt
- Kopieren ganzer Dateien ist aber Streaming
- Benchmarken Sie ggf ihre Applikation. Je nachdem wieviel Sie von der Datei lesen ist „staging vor Jobstart“ oder „direktes Lesen vom Netzwerk“ die bessere Strategie
  - Bedenken Sie auch Skalierungseffekte!`

Innere Organisation der Datei.

- Beispiel aus der Teilchenphysik:
- nTuple aus Vierervektoren werden gespeichert
  - (Energie, Px, Py, Pz)

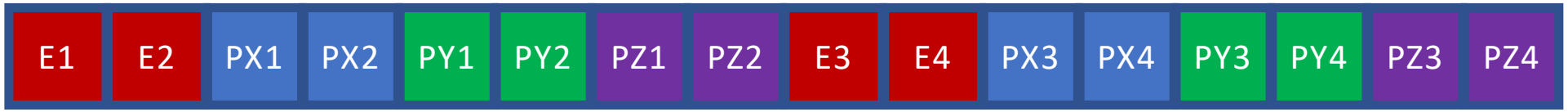
Gruppierung nach Teilchen?

Gruppierung nach Variable?

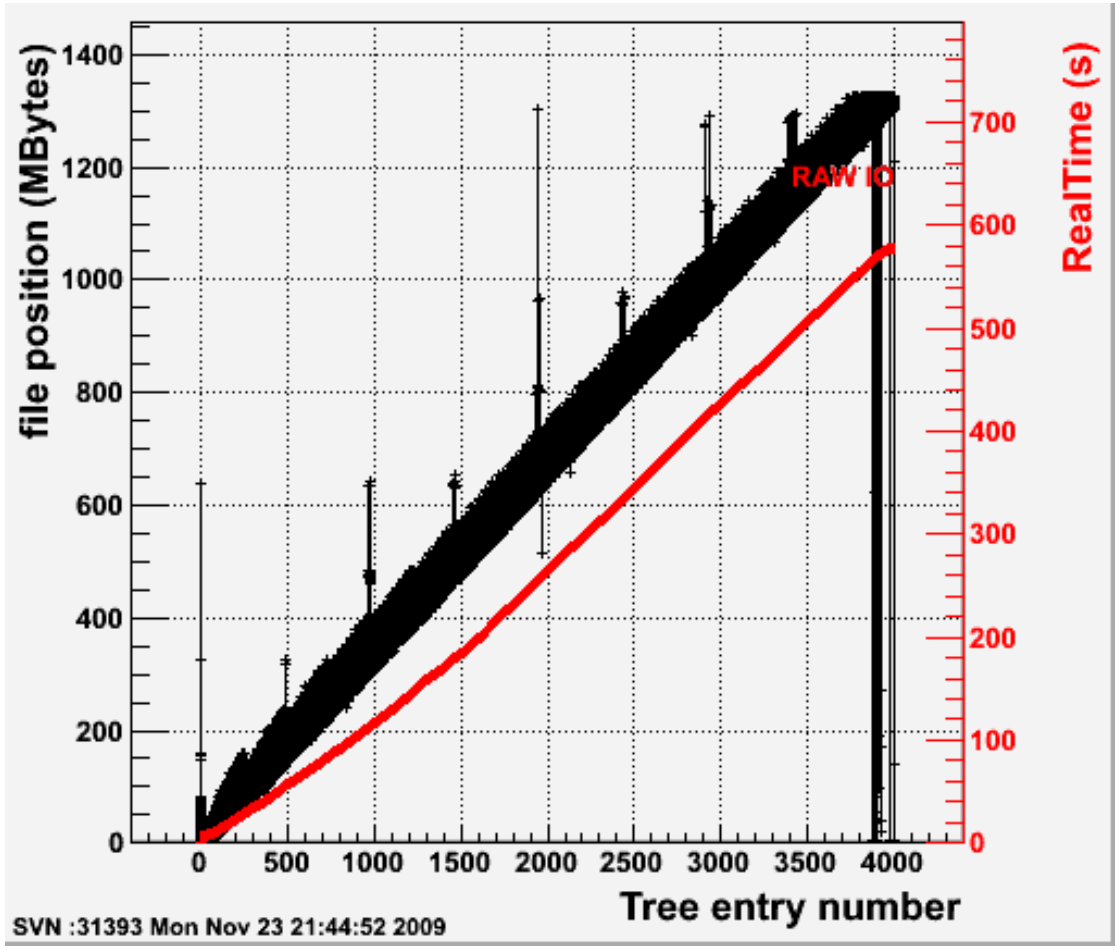


Hängt davon ab

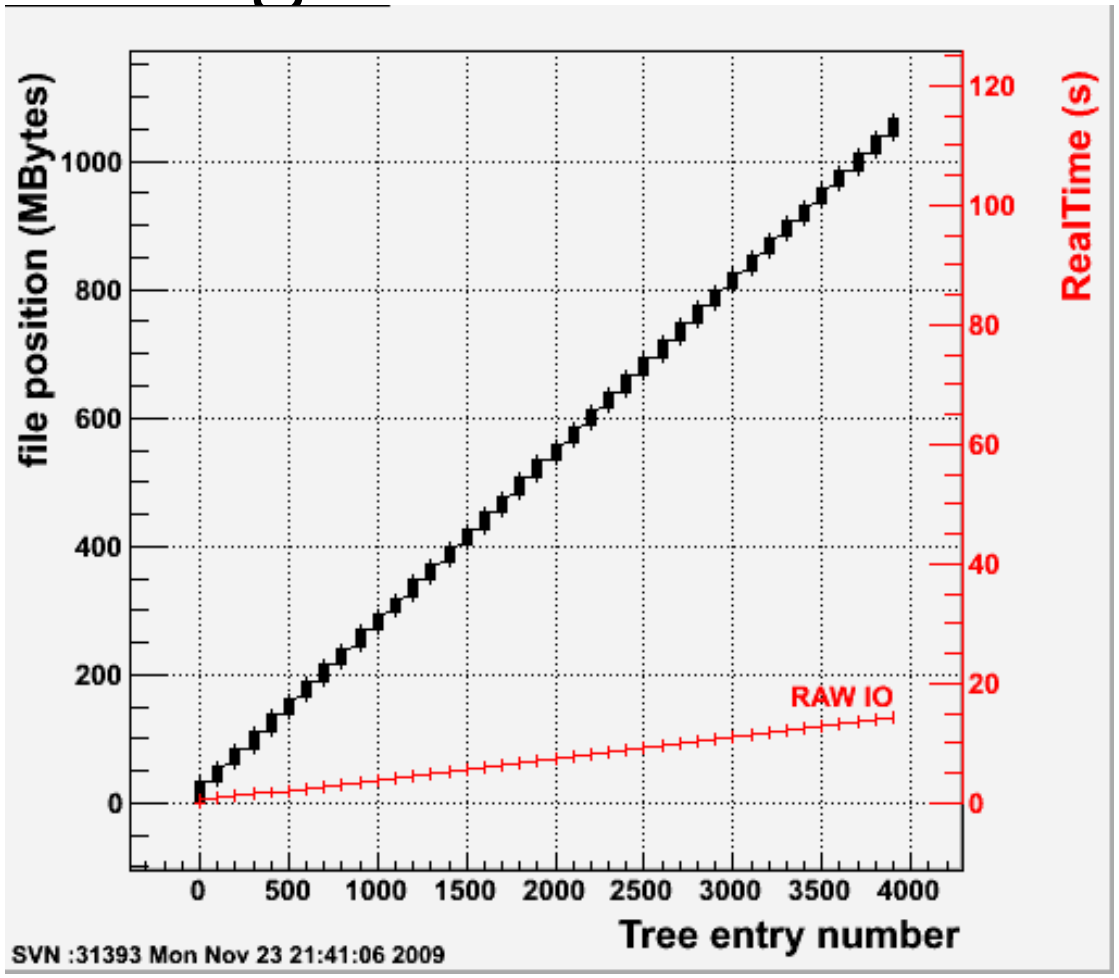
- Loop über die Teilchen, Berechnung von $m^2=E^2-(Px^2+Py^2+Pz^2)$ pro Teilchen
- Loop über die Teilchen, Häufigkeitsverteilung von E
- Machen Sie sich Gedanken! Machen Sie Benchmarks! Ggf Mischformen als Kompromiss!
- Als IO-Framework Designer: Schaffen Sie Optionen die Nutzer einstellen können
- Und denken Sie dran: Die SSD in ihrem Entwicklerlaptop ist nicht das Mass aller Dinge!



Der Cache kann manches abfangen



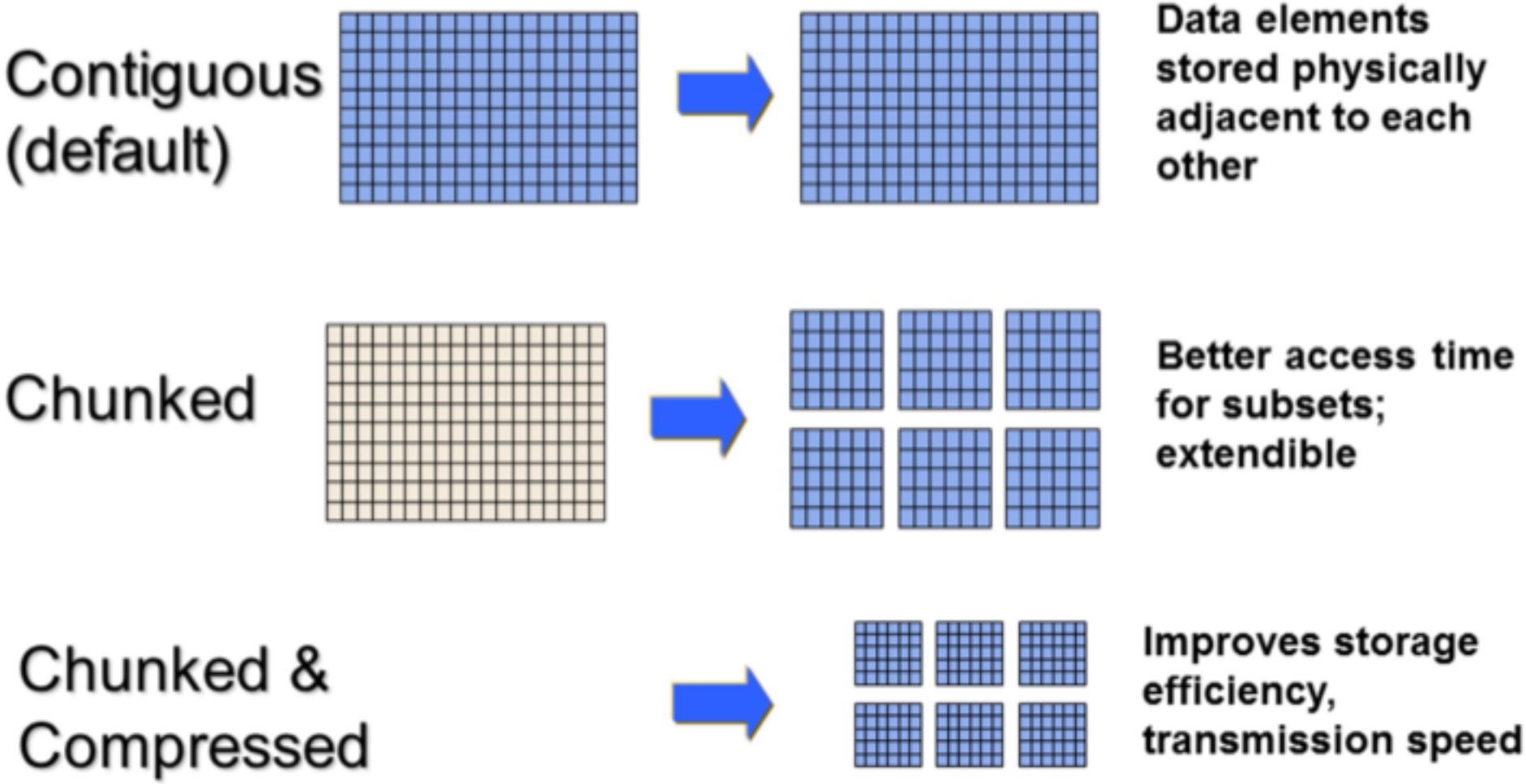
Ohne Cache



Mit Cache

https://root.cern.ch/root/v526/Version526.news.html

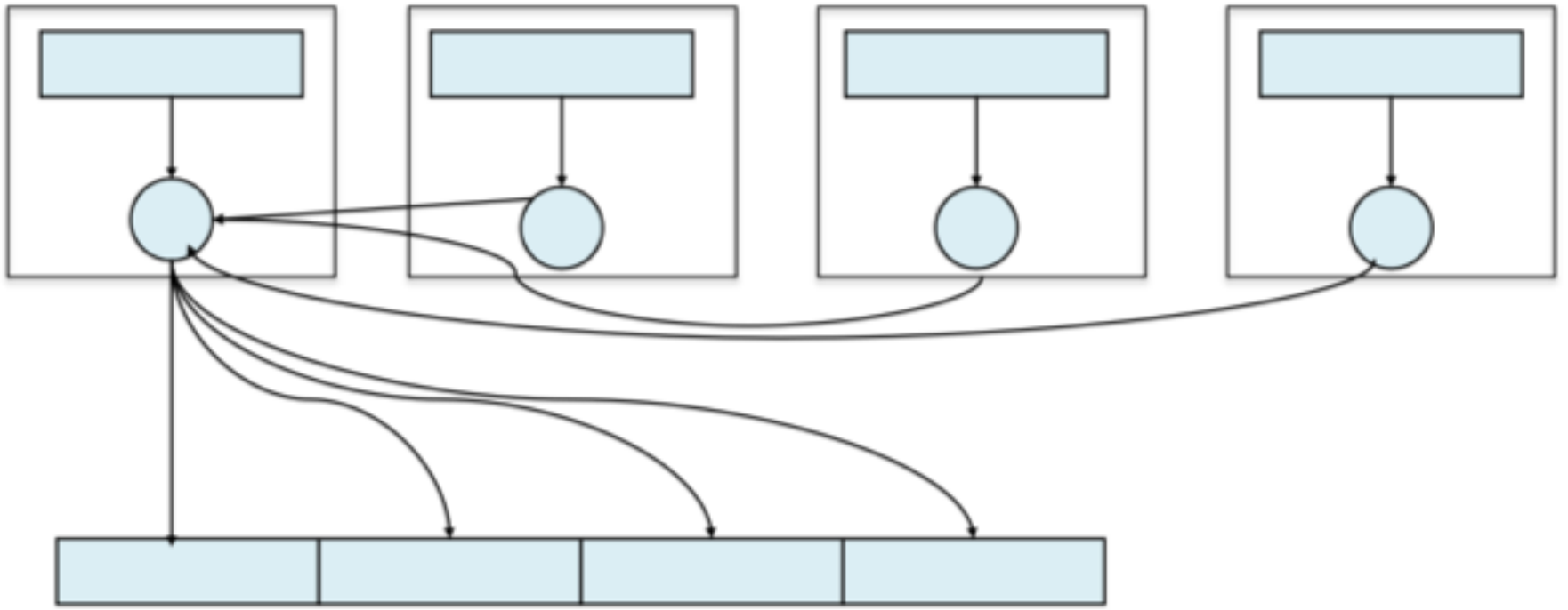
Beispiel: HDF5



https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf

Wie organisiere ich IO?

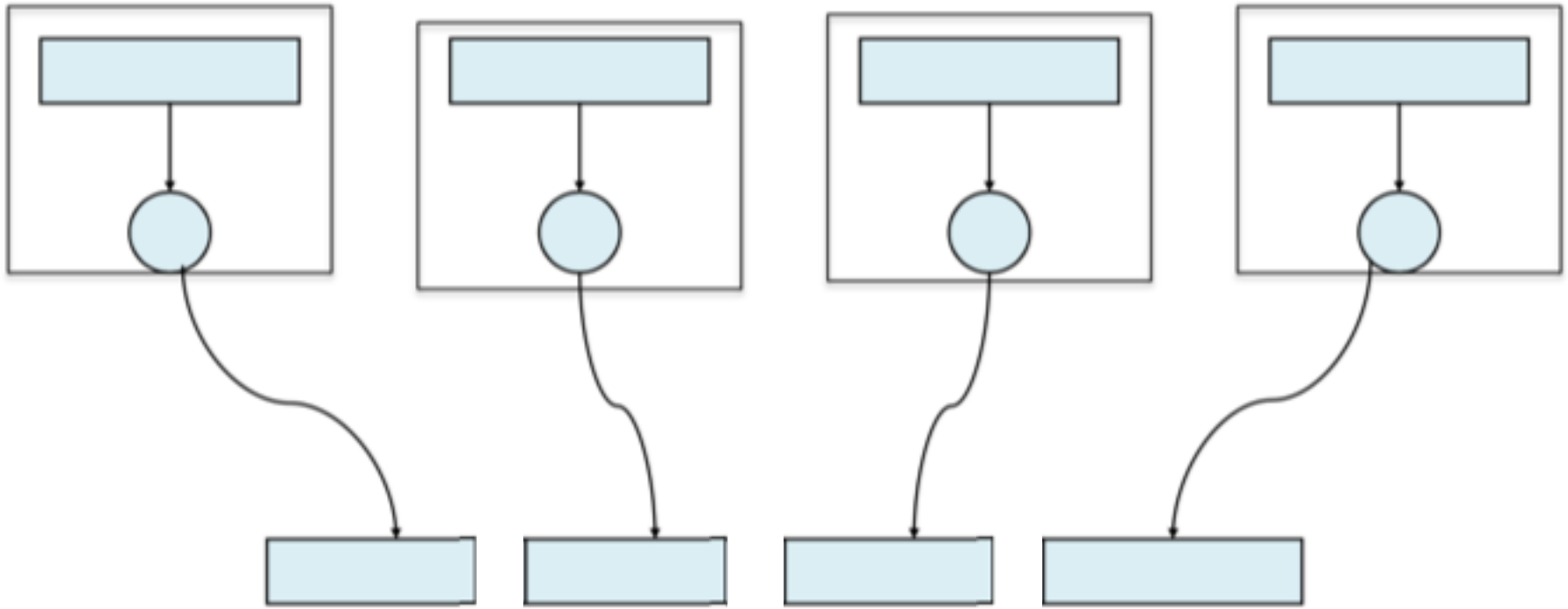
- Kommunikation zum Master, dieser schreibt als Einziger eine Datei - sequentiell



https://www.tacc.utexas.edu/documents/13601/900558/MPI-IO-Final.pdf/eea9d7d3-4b81-471c-b244-41498070e35d

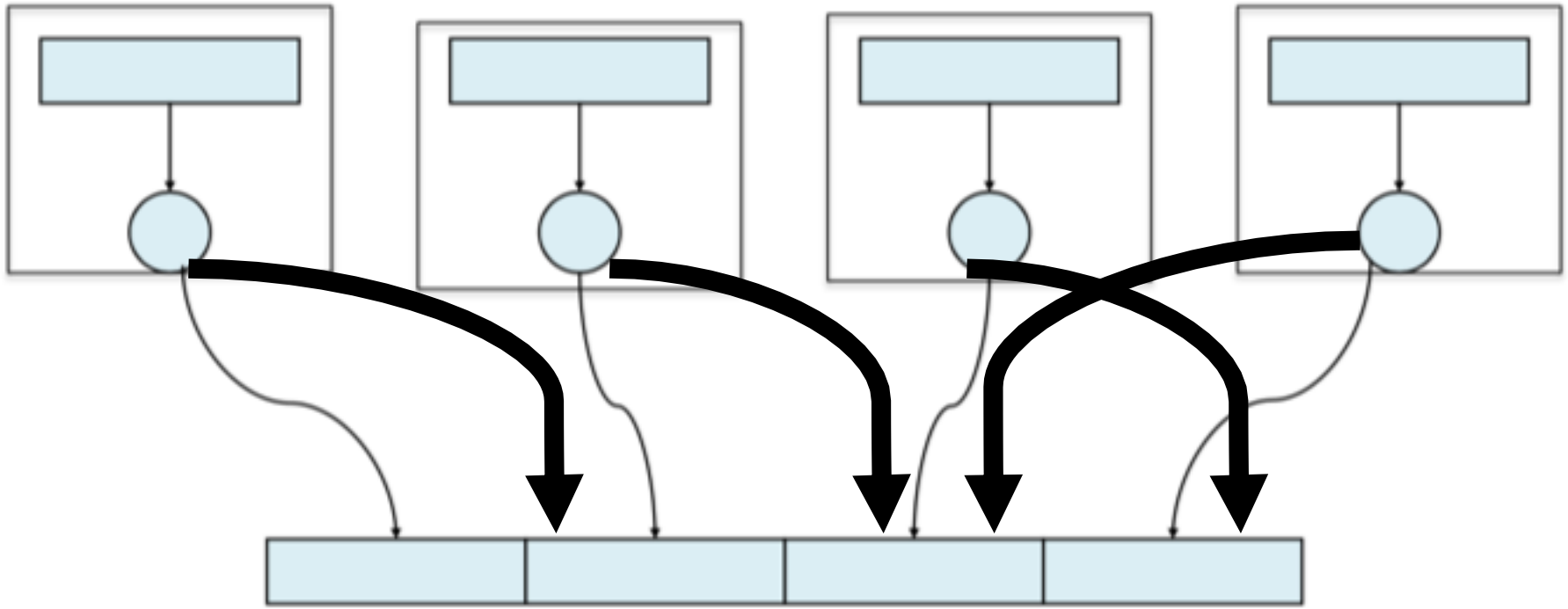
Wie organisiere ich IO?

- Jeder Node beschreibt eine Datei



Wie organisiere ich IO?

- Alle Nodes beschreiben gemeinsam eine Datei
  - Kann kollektiv oder unabhängig erfolgen



Dateigrössen

- Was ist die optimale Dateigrösse?

- Metadatenoperationen: Sind schlecht.
- Also: Unterteilen Sie ihren Datensatz so, dass nicht „zuviele“ Dateien entstehen ... Zumindest nicht in einem Verzeichnis
- “ls -l“ wird ewig dauern, Parallelisierung ist schwierig
- Wenn Sie eine interne Struktur haben:
  - ZB .../conf, .../logs, .../error, .../thumbnail , .../raw , .../exif
  - Mit jeweils kByte – wenige Mbyte grossen Dateien
  - Dann können Sie auch spezielle Formate benutzen die dies zusammensetzen

Einige Formate aus der Wissenschaft

- zB ROOT in der Teilchenphysik
- zB FITS in der Astronomie
- zB HDF5 im HPC Umfeld, aber auch in Photon Science

- Einige davon können auch parallel (im Sinne von HPC) verwendet werden, zB HDF5

MPI-IO in a nutshell

- Ein Teil der MPI Spezifizierung
- User-Sicht: Schreiben/Lesen ist im Code und Vorgehen vergleichbar zu Kommunikation zwischen Nodes
- Syntax vergleichbar zu normaler POSIX Syntax
- Kollektive und unabhängige Calls
- Unter der Haube auf Performance optimiert

Zusammenfassung

- IO wichtiges Thema für Gesamt-Performance
- Wichtig als Nutzer / Entwickler die Implikationen der Applikation/Algorithmus bis auf die Bits der Festplatten zu verstehen
- Viele Tools um IO zu machen
  - Low-Level: POSIX
  - Mid-Level: MPI-IO
  - High-Level: Frameworks
- Versuchen Sie, Frameworks zu benutzen, aber verlieren Sie die Grundlagen nicht aus den Augen!