HPC-Systeme: Cluster &Job Management Storage

Prof. Dr. Volker Gülzow & Dr. Sergey Yakubov

Dr. Yves Kemp

SS 2019

Management Tool

Management der Systeme aus Sicht der Nutzer:

Job Handling, Workflow, Benutzung durch mehrere Nutzer

Auch ”Scheduling” genannt. Wir betrachten aber nur “das grosse Scheduling”, nicht was zB innerhalb des Prozessors / des Betriebssystems an Scheduling passiert

- Vorlesung und Übung

Management der Systeme aus Sicht von Sys-Admins:

Installation, Konfiguration, Management, Life-Cycle-Management

- Speziell für grosse homogene Cluster
- -> Erster Teil dieser Vorlesung

The point of the HPC scheduler is to keep everyone equally unhappy

Aus: E-Mail Signatur von James A. Peltier, HPC Coordinator, Simon Fraser University
… es gibt aber unzählige Varianten von diesem Satz

Zuteilung von Rechenzeit

- Grosse (wissenschaftliche) HPC Zentren:
- Erstmal:
  - Typischerweise Anträge, die dann begutachtet werden
  - Viel Forschung
  - Viel Papierarbeit
  - ... Und dann erst eventuell die Möglichkeit zu rechnen

Beispiel: **Application for Projects on JUQUEEN**

Regular calls and calls for large-scale-projects
The GCS/NIC-call is open for project leaders, whose affiliation is in Germany (or is a foreign office of a German institution). Also eligible are German scientists if they are working in an international organisation with significant German participation (e.g. CERN, ESA, ESO) and do not have a permanent position there. Interested researchers from abroad (within Europe) are invited to apply via PRACE.
Project applications for JUQUEEN may be submitted by any scientist qualified in his or her respective field of research. Computing resources are allocated on the basis of independent referees' reports. Apart from the **scientific relevance of the project**, an important criterion for the allocation of computing resources is that the project can make **efficient use of the computer and use a large number of processors in parallel for the simulations**.

http://www.gauss-centre.eu/gauss-centre/EN/HPCservices/HowToApply/LargeScaleProjects/largeScaleJUQUEEN.html

Beispiel: **Application for Projects on JUQUEEN**

The following criteria must be met by the projects in order to be eligible:

- Scientific excellence.
- Clear scientific goals and verifiable milestones on the way to reach these goals.
- Preliminary studies that show **good scaling behaviour** of the program to very high processor numbers (at least 8192 cores) under production conditions (i.e. typical parameter sets and problem sizes of the planned project, including I/O). Contact sc@fz-juelich.de for a test account.
- A detailed and clearly arranged work schedule in form of a table or Gantt chart.

Well-founded and detailed demonstration of the required runtime of the program and the total required CPU time.

Please consult also the detailed technical guidelines for projects applying for JUQUEEN
The smallest amount of computing time that should be requested is 5 Mill. core hours.

http://www.gauss-centre.eu/gauss-centre/EN/HPCservices/HowToApply/LargeScaleProjects/largeScaleJUQUEEN.html

Beispiel: **Application for Projects on JUQUEEN**

Review process

Large-scale projects (35 million core hours or more) are peer-reviewed by a committee of the GCS. If approved they will get increased user support and preferential processing of their jobs.

Regular projects are peer-reviewed by a committee of the John von Neumann Institute for Computing (NIC) on behalf of GCS.

http://www.gauss-centre.eu/gauss-centre/EN/HPCservices/HowToApply/LargeScaleProjects/largeScaleJUQUEEN.html

Beispiel: **Application for Projects on JUQUEEN**

Application

Computing time periods are yearly - with the possibility of application twice per year - and will begin on 1 May and on 1 November each year.

The next call for proposals will be announced on 10 July 2017. The deadline for the next application will be on **14 August 2017, 17:00 CET** for the computing time period from 1 November 2017 until 31 October 2018.

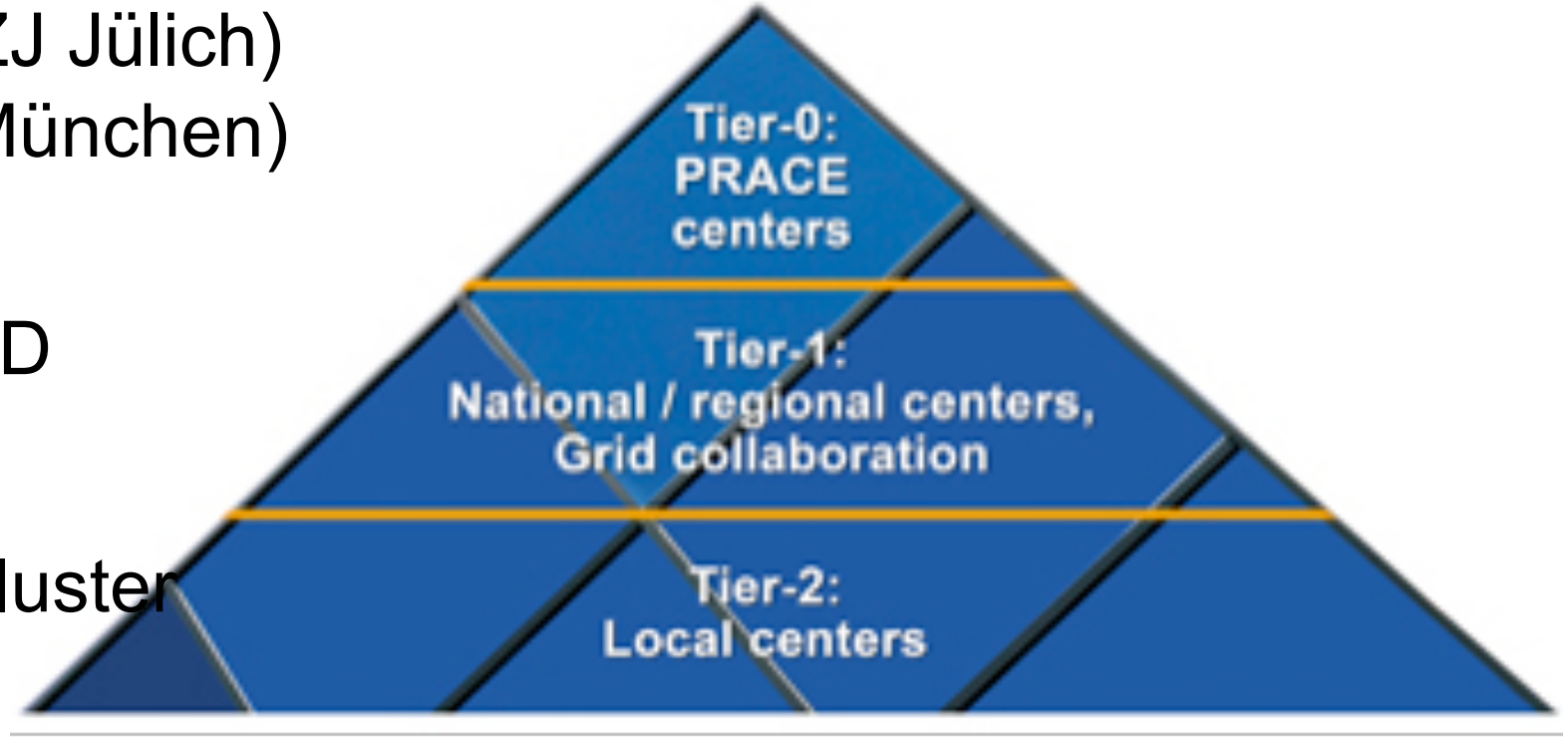
http://www.gauss-centre.eu/gauss-centre/EN/HPCservices/HowToApply/LargeScaleProjects/largeScaleJUQUEEN.html

Die HPC Tier Pyramide

ZB. JUQUEEN (FZJ Jülich)
SuperMUC (LRZ München)
... Europaweit

Einige Systeme in D

ZB DESY Maxwell Cluster



http://cnx.org/contents/Ug-5YUe5@1/The-European-e-Infrastructure-

Leben in einem Tier-2 Zentrum

- Keine formellen Anträge, keine formelle Begutachtung
- Es wird aufgepasst (von Admins und von freundlichen Mitnutzern) dass die Jobs „HPC-artig“ sind
- DESY investiert über IT in eine allgemeine Grund-Ausstattung
- Manche Projekte investieren Geld in die Infrastruktur, und erkaufen sich damit gewisse Rechte
- Andere Projekte nutzen die Ressourcen die „übrig bleiben“

- ... Wie das technisch und organisatorisch funktioniert lernen wir in den nächsten Folien

Job Verteilung

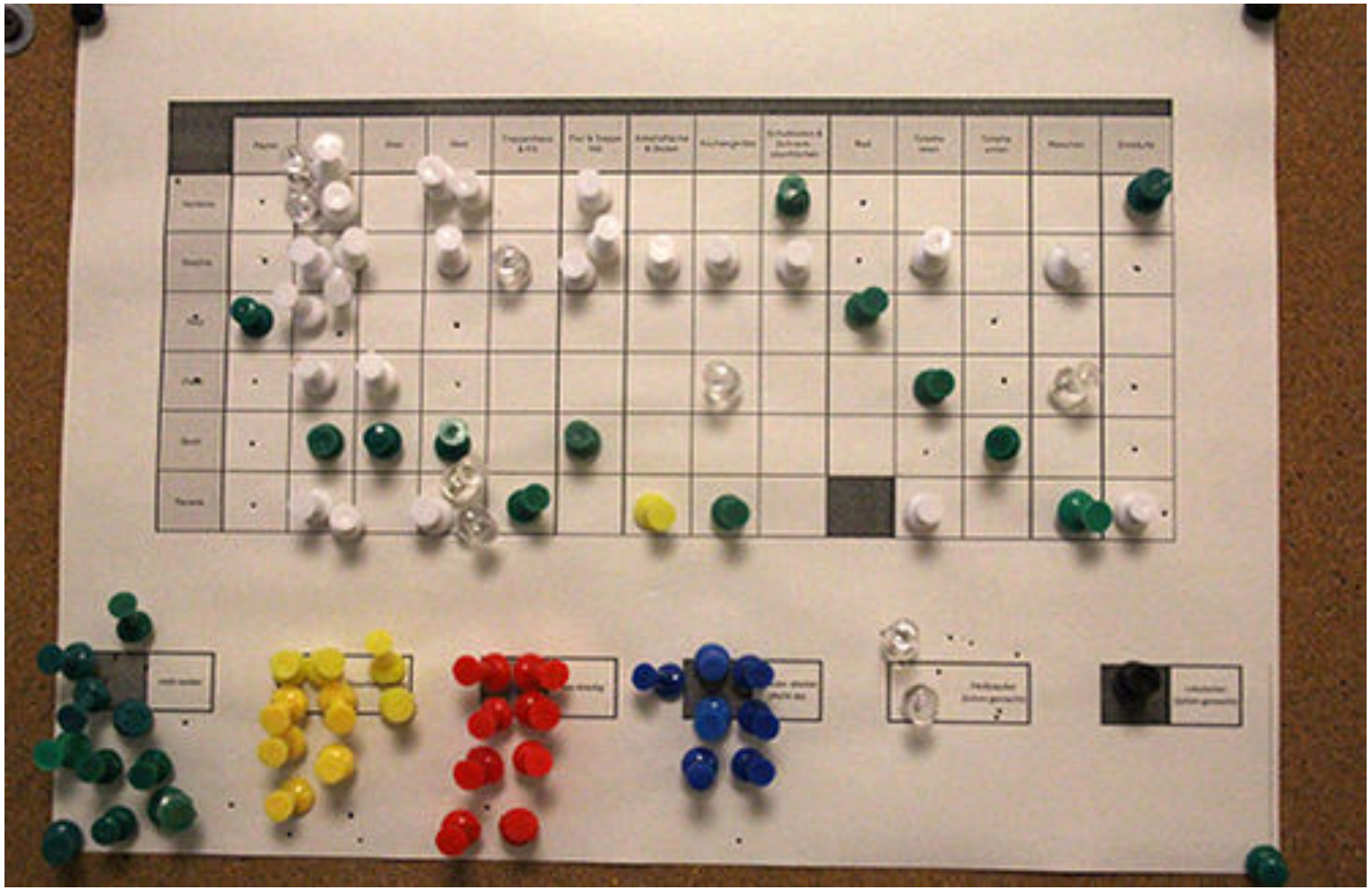
```
> cat steer.list
server1 23
server2 42
...

> cat steer.list | while read node seed; do
    (ssh $node „/home/kemp/run_simulation.sh $seed“ &)
done
```

Probleme

- Woher weiss ich, ob die Knoten gerade frei sind?
  - Ich kann einen anderen Job laufen haben
  - Jemand anderes kann einen Job laufen haben
  - Die Knoten können zB wegen Wartung ausser Betrieb sein

- Wir brauchen einen Scheduler!
  - Mein Wörterbuch sagt: Scheduler heisst u.a. auch: Planer, Disponent
  - Scheduler hat also sowohl eine Software-Seite, als auch eine organisatorische Seite

Einfachstes Beispiel eines Schedulers: Kalender



... Oder deren elektronische Variante

ssh und node-List auf Basis des Kalenders

https://www.zu-daily.de/daily/tiefenbohrung/2015/11-17-bathon-wg-putzplan.php

Kalender als Scheduler

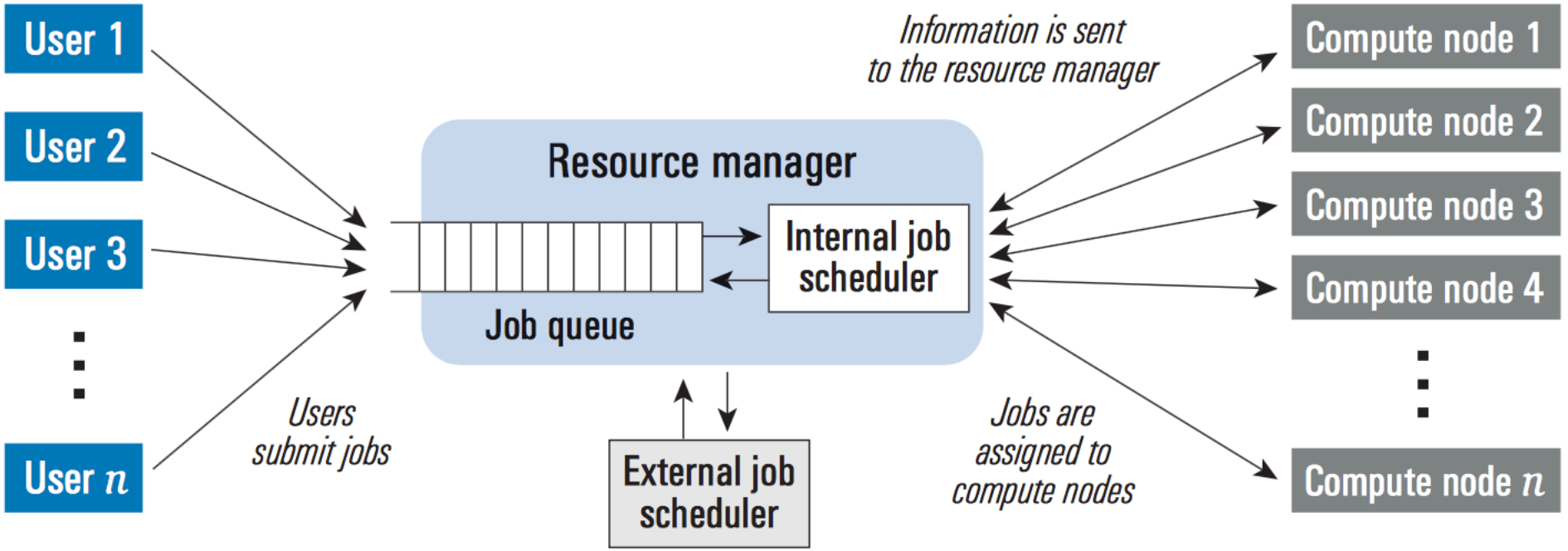
- Vorteile:
  - Kalender ist einfach zu bedienen
  - Jobs sind einfach per ssh abzuschicken, kein Anpassen der Software notwendig
  - Interaktivität ist gewährleistet
  - Feste Reservierungen (zB für Strahlzeiten) möglich
- Nachteile:
  - Sehr statisch, kann nicht auf variierende Laufzeiten eingehen
  - Verschnitt von Ressourcen
  - Reservieren auf Verdacht
- Die Nachteile nehmen schnell Überhand

Scheduler ist nicht alles

- Der Scheduler teilt nur ein: Wer darf wann wo rechnen?
- Das Batch-System nimmt die Jobs entgegen, verwaltet sie, übernimmt das Dispatching auf die Worker-Nodes und Verwaltung der Resultate
  - Das Batch-System arbeitet auch dann wenn Sie schlafen ☺

- Die Trennung ist heute verschwommen, viele Produkte können beides.

Typisches Setup



http://www.dell.com/downloads/global/power/ps1q05-20040135-fang.pdf

Einige Kriterien für die Auswahl

- Skalierbarkeit
  - Manche Systeme skalieren über 100k Server
  - Manche Systeme skalieren über 1M Jobs (running & queued)
  - Mittlerweile immer wichtiger: Skalierbarkeit in Clouds möglich?
- Mächtigkeit des Schedulers, implementierte Algorithmen
  - FairShare, Pre-Emption, Backfill, Prioritäten, Reservierung, ...
- Für HPC enorm wichtig:
  - Unterstützung von parallelen Jobs? Also Jobs die gleichzeitig mehrere Server umspannen?
- Stabilität, Administrierbarkeit, Integrierbarkeit mit Meta-Schedulern (zB Grid)
- Kosten

HTC und HPC

- HTC: High-Throughput-Computing
  - Typischerweise viele, lose gekoppelte Jobs (“trivial parallelisierbar“)
  - Entscheidend ist nicht die Peak-Power, sondern der Gesamt-Durchsatz innerhalb eines längeren Zeitraums
- HPC: High-Performance-Computing
  - Typischerweise parallele Jobs mit Inter-Process-Communication
  - Peak-Power ist wichtig, ebenso gute Kommunikation innerhalb des Jobs

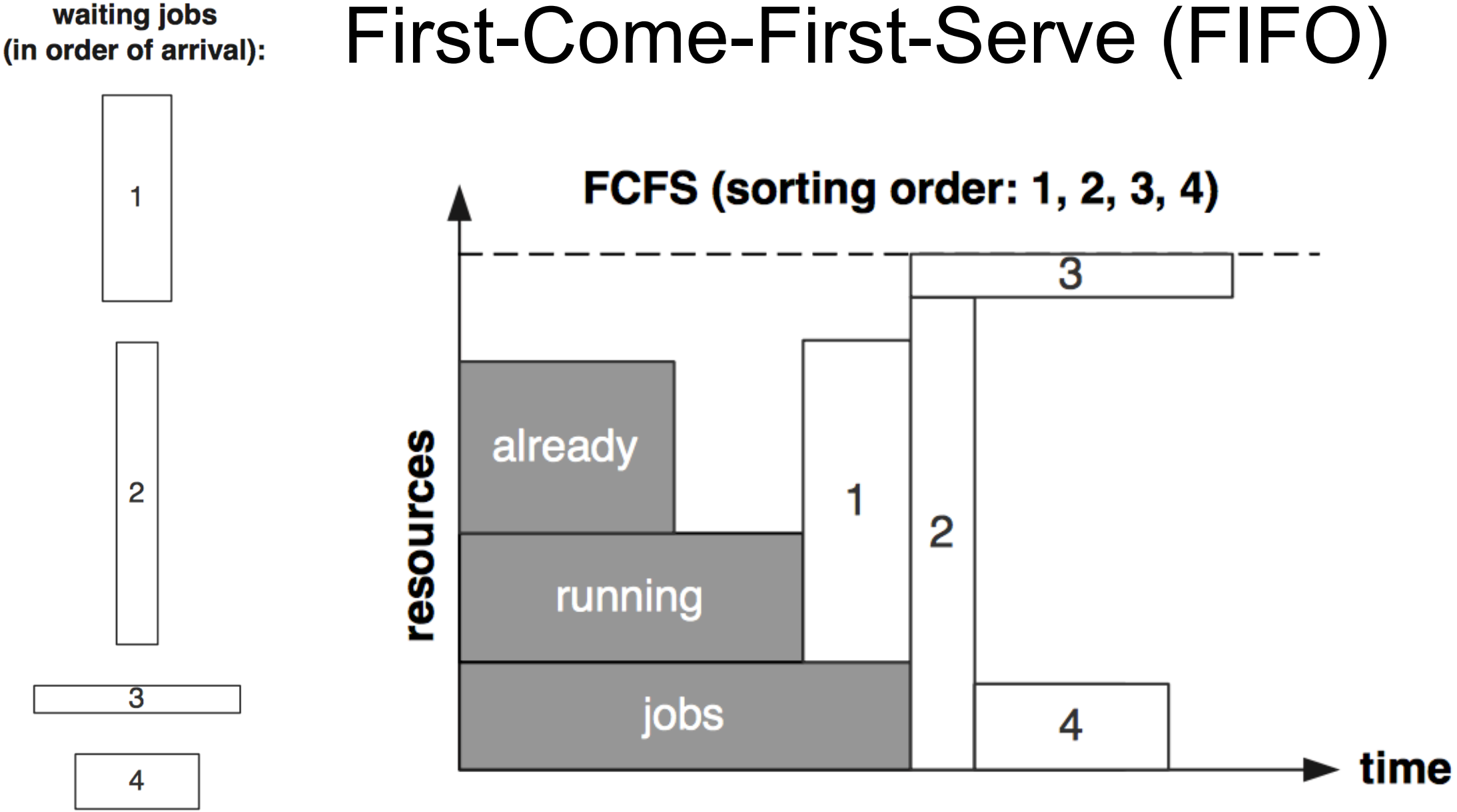
HPC

Quelle: http://auto.ferrari.com/de_DE
Folie inspiriert von HTCondor Enwickler

HTC

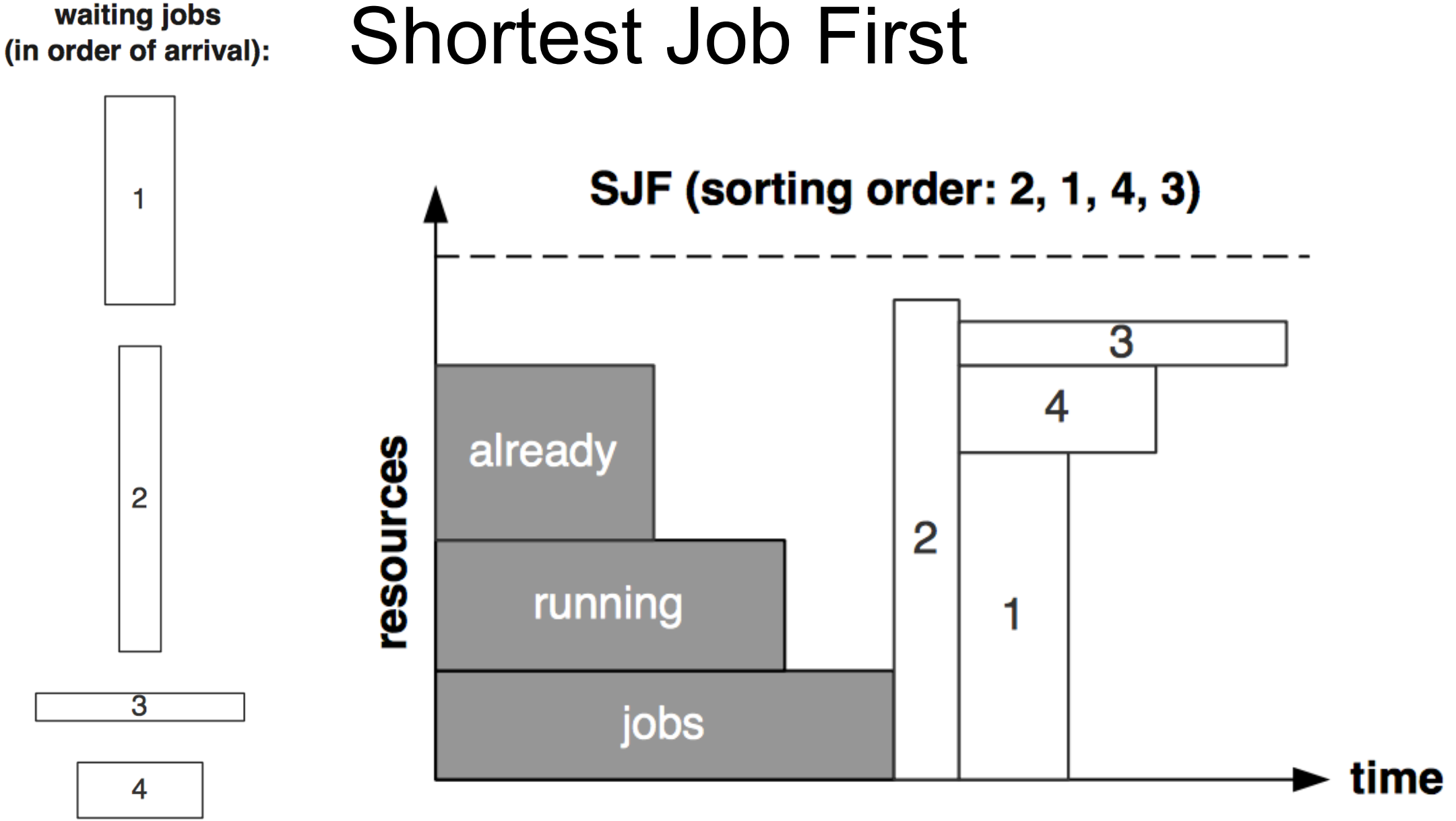
Quelle: http://www.mirror.co.uk/news/world-news/worlds-worst-traffic-jam-drone-6594560

First-Come-First-Serve (FIFO)



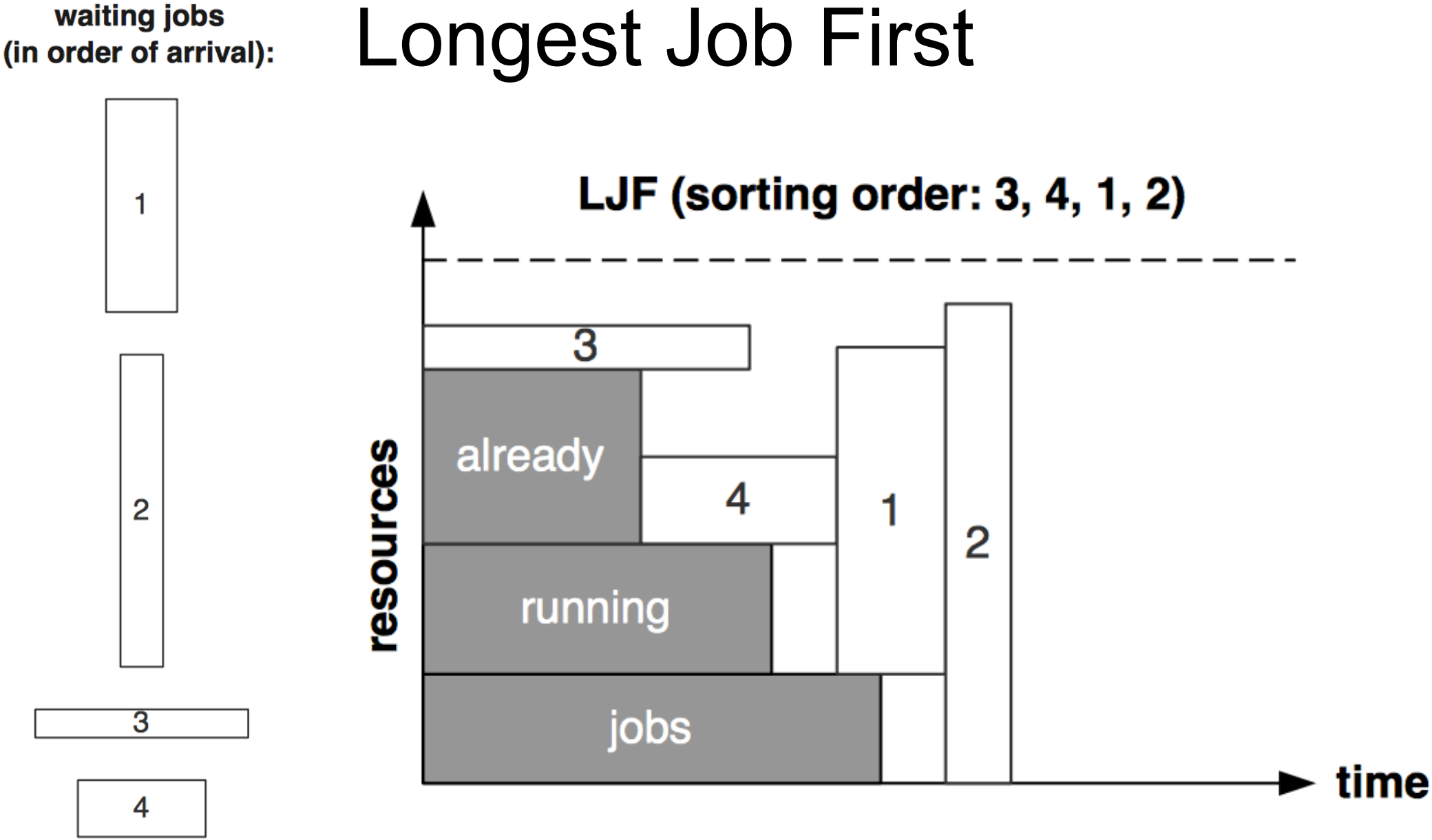
Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

Shortest Job First



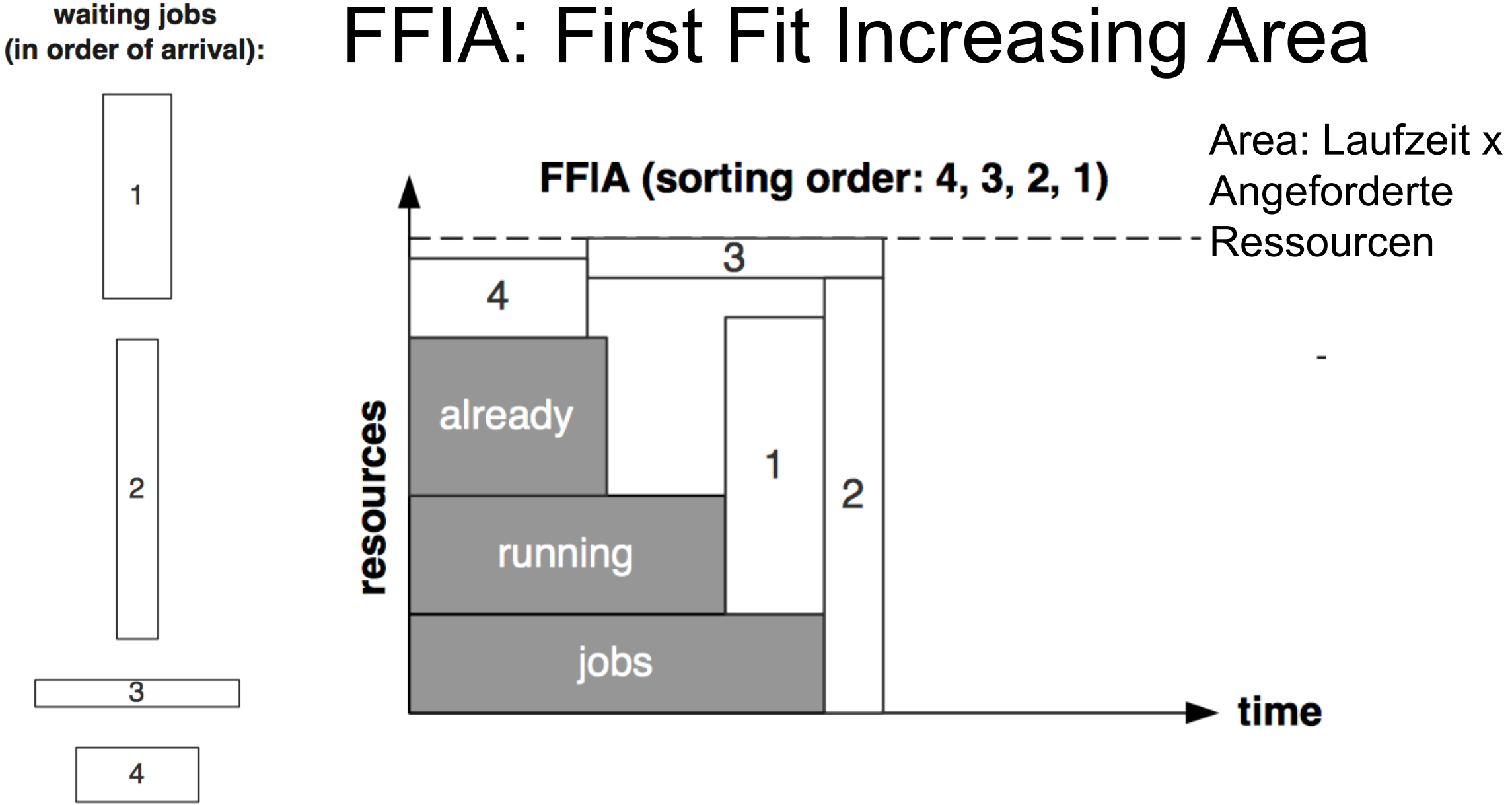
Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

Longest Job First



Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

FFIA: First Fit Increasing Area



Area: Laufzeit x Angeforderte Ressourcen

-

Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

Allgemeiner: Sortierungs-Kriterien

- Submit-Zeitpunkt
  - First-Come-First-Serve
  - Last-Come-First-Serve (Unüblich, ... Wäre ein Stack)
- Laufzeit
  - SJF: Shortest Job First (Variante: First Fit Increasing Height = Laufzeit)
  - LJF: Longest Job First (Variante: First Fit Decreasing Height = Laufzeit)
- Fläche = Ressourcen-verbrauch = Laufzeit x angeforderte Ressourcen
  - FFIA: First Fit increasing Area: Abwandlung von SJF, mit Berücksichtigung von Ressourcenverbrauch
- Job-Gewicht
  - ZB. Projekt, oder User-definiertes Gewicht, oder Wartezeit
- Smith-Ratio
  - Job-Gewicht / Fläche
- ...

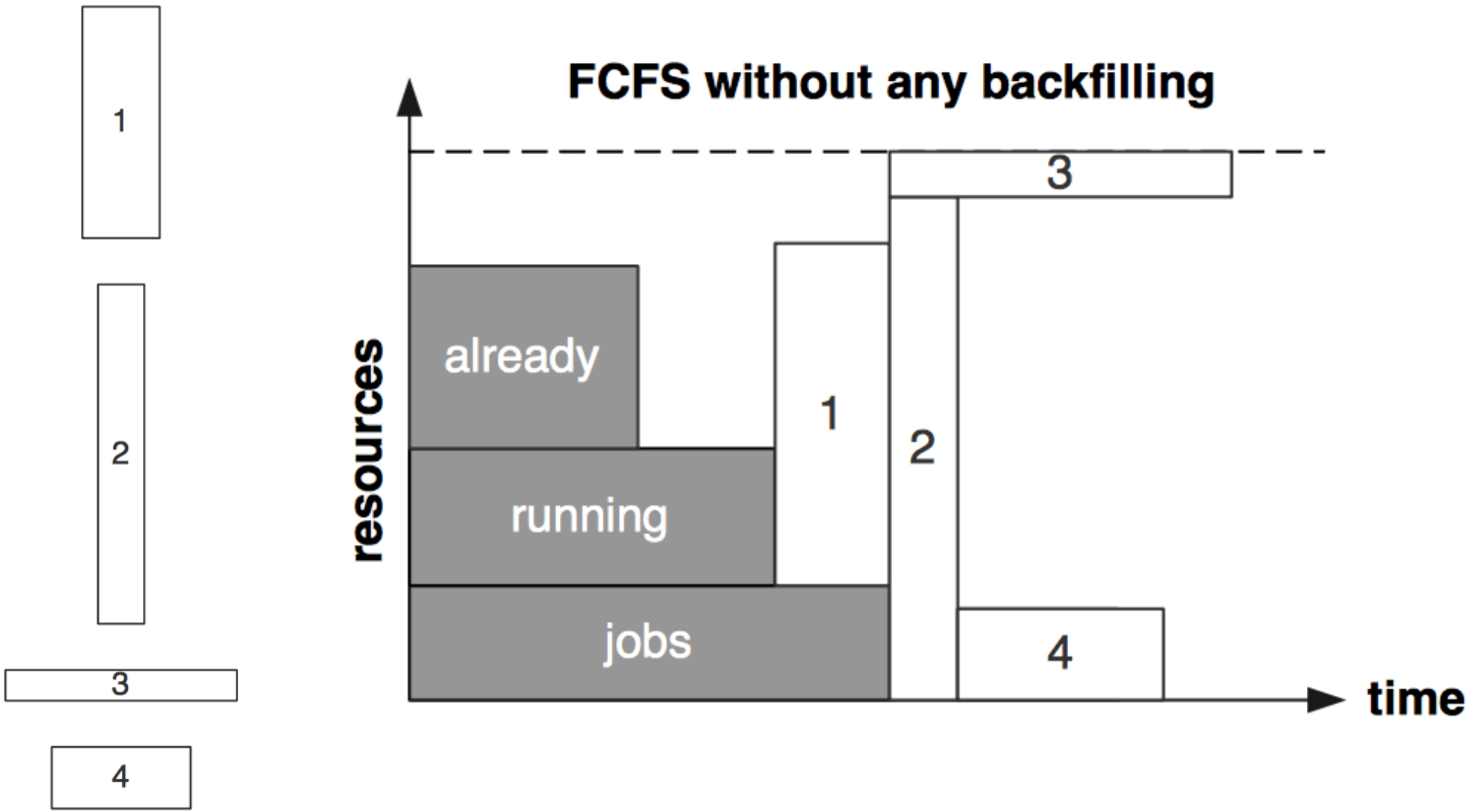
Auswahl-Kriterien

- „Front“: Immer der erste Job wird ausgewählt
  - Macht bei manchen Sortierungen Sinn, zB FCFS
  - Ggbfs. werden die Ressourcen nicht optimal ausgelastet
- „First Fit“: Der erste, passende Job wird ausgewählt
  - Macht bei manchen Sortierungen Sinn, zB FirstFitIncreasingArea
  - Ggbfs. müssen grössere Jobs länger warten
- „Best Fit“: Die ganze Liste wird untersucht, und der am besten passende Job ausgesucht – Bei Gleichstand davon der Erste
  - Beste Auslastung, ggbfs. müssen grössere Jobs länger warten
  - Scheduling-Auffand sehr hoch

Verbesserung: Backfilling

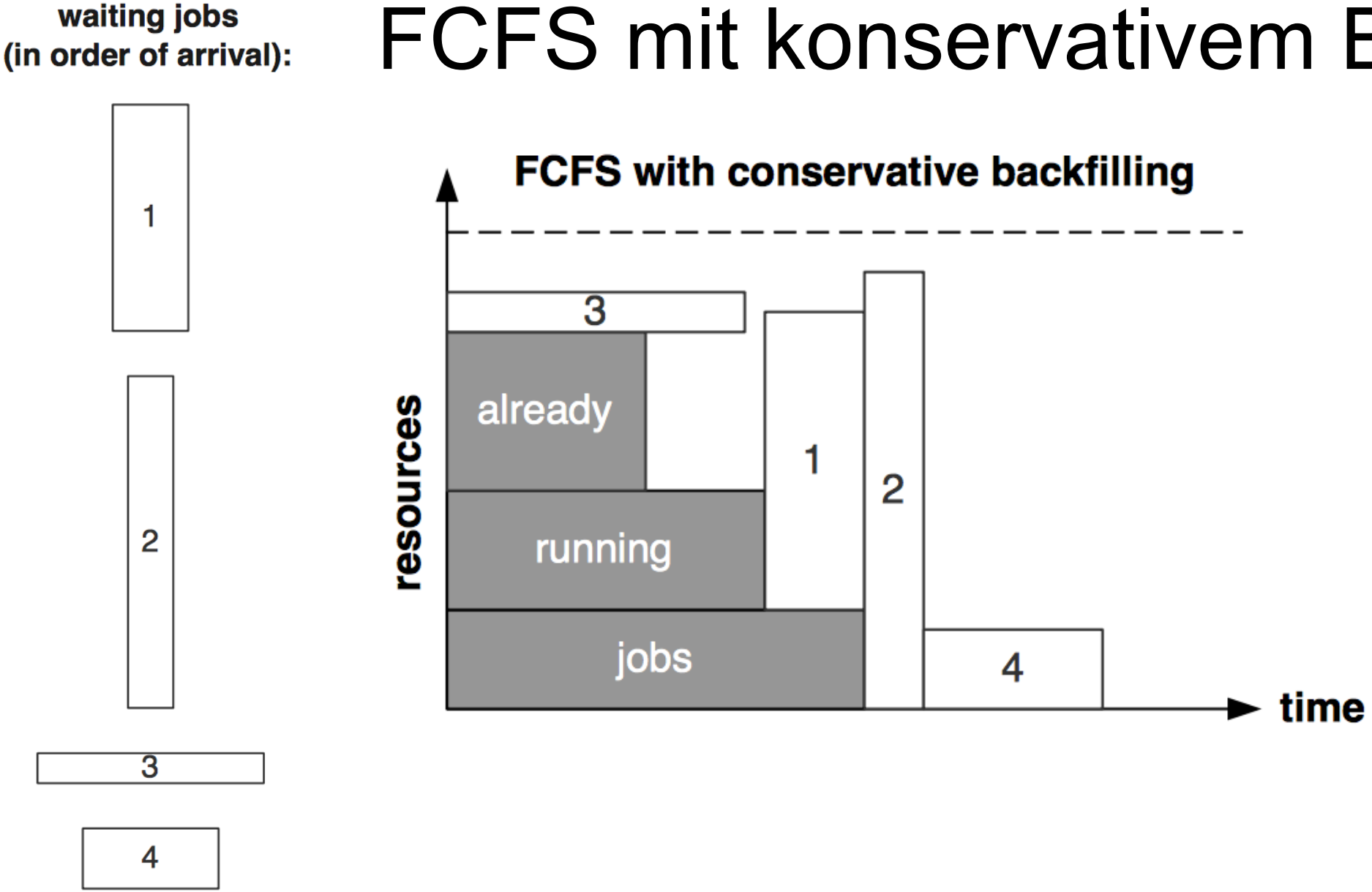
- Idee:
  - Man nehme ein Sortier-Verfahren welches fair und einfach zu verstehen ist, aber eine schlechte Ressourcennutzung bewirkt (Viel Partitionierung bzw. Leerstand)
  - Der Leerstand wird dann mit anderen Jobs befüllt, die deutlich später dran kämen, aber in die Lücken passen würden
- Zwei prinzipielle Arten von Backfilling
  - Konservatives Backfilling: Der Startzeitpunkt der priorisierten Jobs wird nicht geändert
  - EASY Backfilling: Der Startzeitpunkt der priorisierten Jobs kann verzögert werden weil ein Backfill-Job etwas länger braucht und nicht genügend Ressourcen frei sind

FCFS Ohne Backfilling



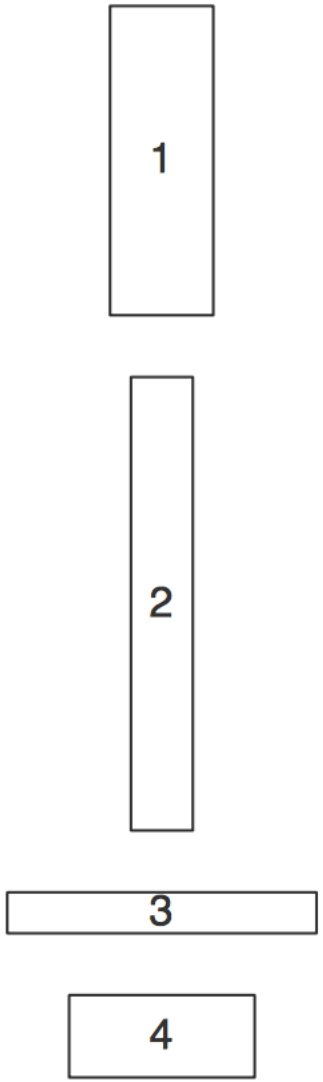
Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

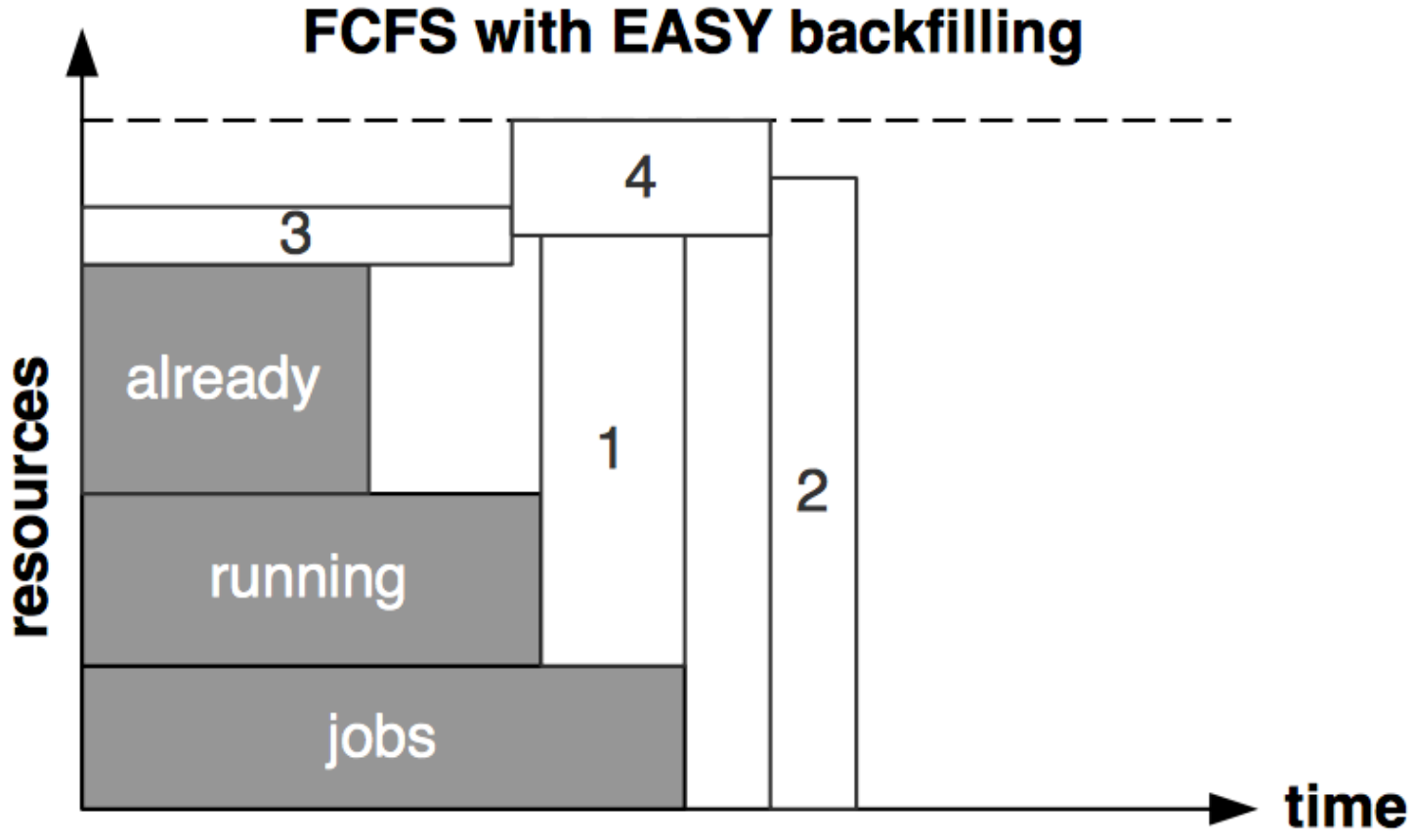
FCFS mit konservativem Backfilling



Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

FCFS mit EASY Backfilling





Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, A. Streit,

Job-Prozess-Knoten Zuordnung für HPC

- Ein Job kann auf mehrere Knoten aufgeteilt werden
- Typischerweise wird aber jeder Knoten exklusiv von einem Job belegt
- Es wird aktuell vorsichtig mit Co-Location experimentiert, dies sind dann aber zwei oder sehr wenige Jobs die gleichzeitig einem Knoten belegen
  - zB IO-Intensiv und Compute-Intensiv
- Typischerweise gilt aber die exklusive Nutzung eines Knoten durch einen Job, was das Scheduling deutlich vereinfacht
- Im HTC Bereich ist die Grund-Einheit „ein Core“

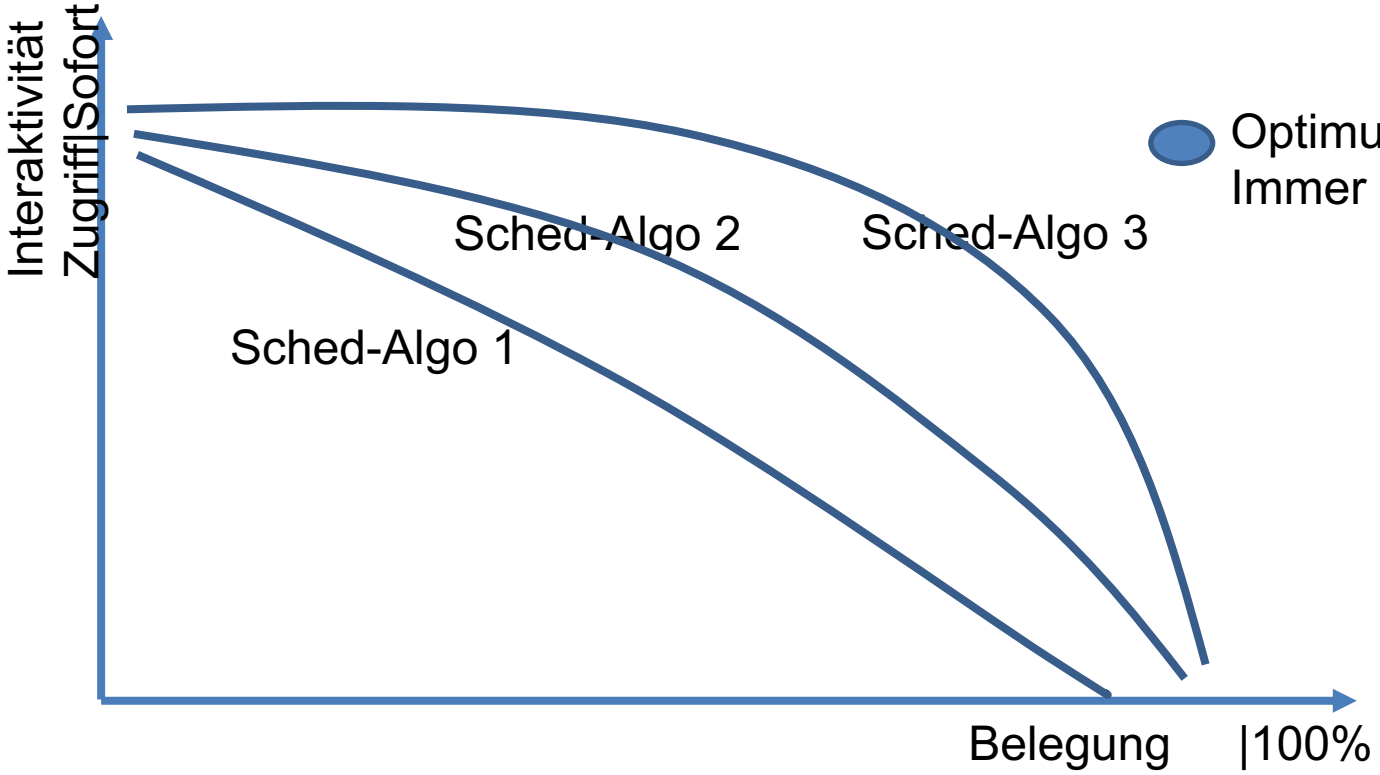
Fair-Share-Algorithmus

- Oft haben unterschiedliche Gruppen (oder Nutzer, oder Projekte) unterschiedliche Soll-Anteile an einem Cluster
  - zB entsprechend ihrem Anteil an der Beschaffung der Hardware, oder politischen Randbedingungen...
- Der Fair-Share-Algorithmus ist eine Sortier-Methode, die anhand dem Verhalten der Gruppe in der Vergangenheit die Job-Liste so anpasst, dass der Soll-Zustand erreicht werden soll
- Typischerweise gewichtet der Fair-Share-Algorithmus die jüngere Vergangenheit stärker
- Ihre Meingung: Ist das „fair“?

Preemption: Suspenden / Killen von Threads oder Jobs ... Und Checkpointing

- Normalerweise läuft ein Job, bis er fertig ist, oder bis seine beantragten Ressourcen erschöpft sind (meist beantragte Laufzeit)
- Es gibt Klassen von Jobs, die keinen/wenig Status haben, und keine hohen Anforderungen an zeitnahe Beendigung stellen
- Idee: Solche Jobs können, bei Bedarf, suspended / beendet werden
  - Damit können evtl. grössere Jobs schneller drankommen, die Gesamteffizient des Clusters wird erhöht
- Diese Art von Jobs hat meist eine Art von Checkpointing:
  - Zwischenresultate werden regelmässig weggeschrieben
  - Oder die Beendigung erfolgt zweistufig, und es gibt etwas Zeit, die Jobinformationen zu sichern und ordentlich zu beenden
- Spätere Wiederaufnahme an der Stelle des Checkpoints

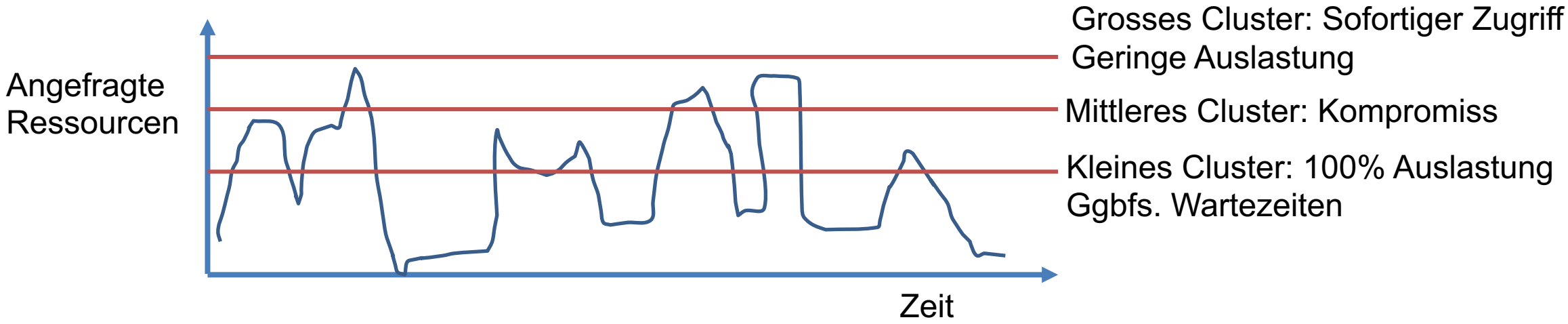
Unterschiedliche Scheduling-Algorithmen



- Das Optimum ist üblicherweise nicht erreichbar
- Verschiedene Scheduling-Algorithmen und Einstellungen können eine bessere Auslastung bewirken
- Typischerweise aber auch sehr viel Psychologie im Spiel

Dimensionierung Cluster

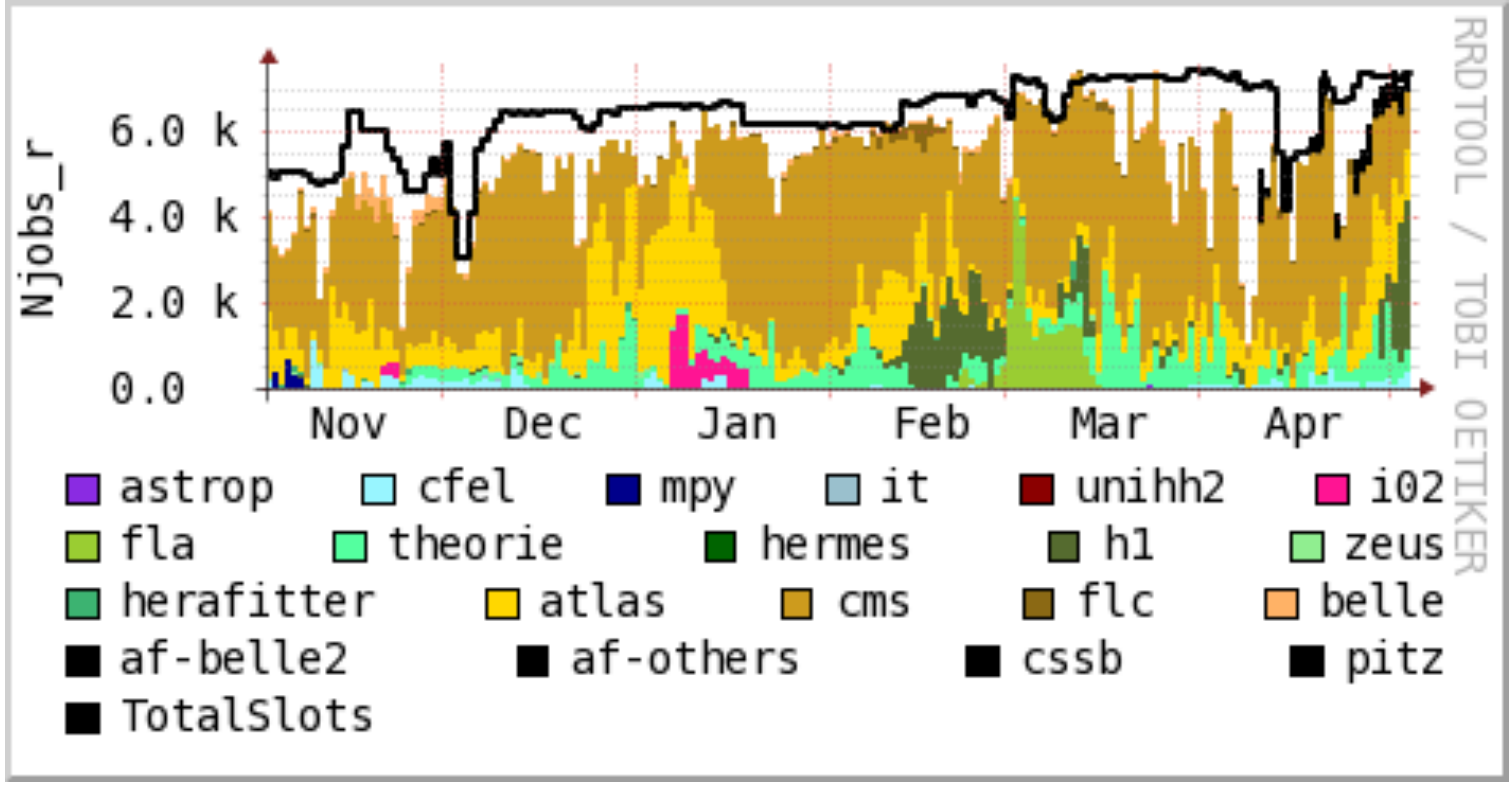
- Bei gegebenem Job-Submission-Profil

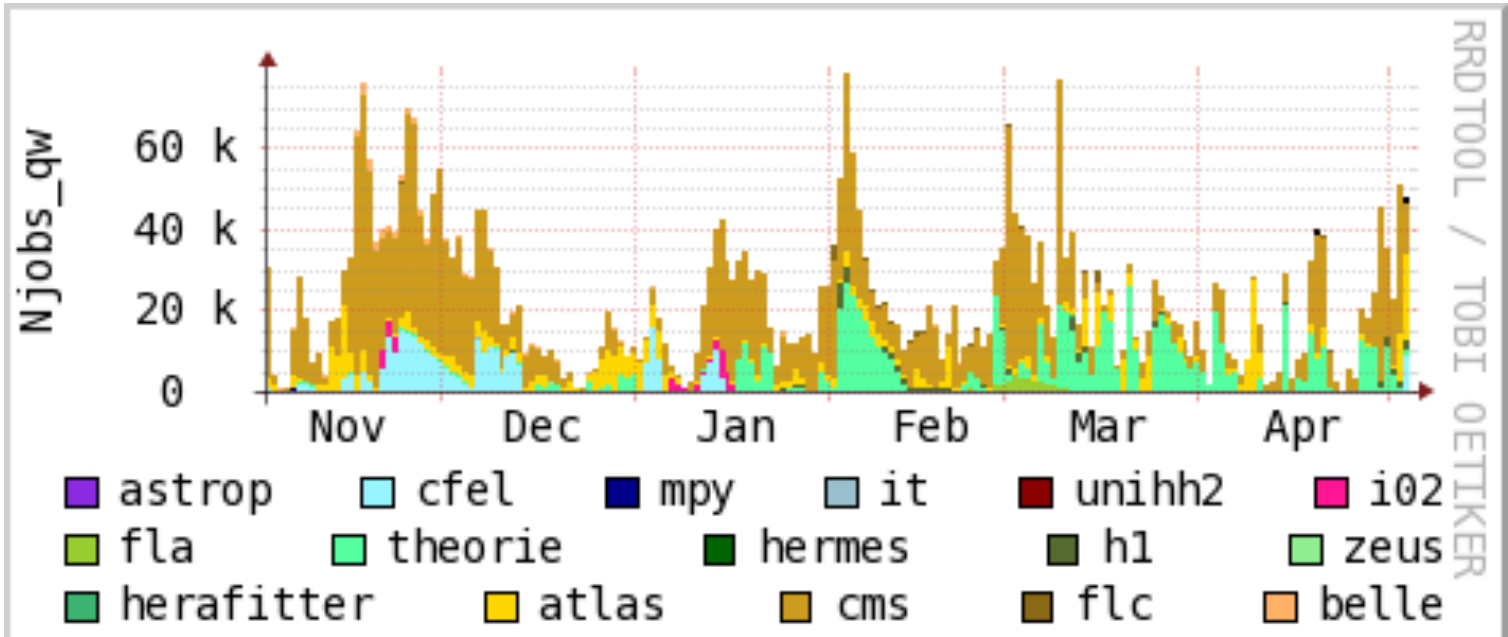


- Wieviel Geld haben Sie? Wie zeitkritisch sind die Applikationen? Wie gut können Sie ihren Nutzern Wartezeiten (und ggbfs. entstehende „Ungerechtigkeiten“ vermitteln)?

Beispiele für Auslastung

DESY: NAF/BIRD
Viele single-core jobs,
„konferenz-getrieben“





Scheduling im Zeitalter von Cloud

- Die Cloud – unendliche Ressourcen... Wir schreiben das Jahr 2006. Dies ist die Geschichte des ersten kommerziellen Cloud-Anbieters, der mit enormer Rechenleistung unterwegs ist um neue Formen von Computing zu erforschen. Viele Rechenzentren von dem was wir kennen entfernt, dringen Cloud-Anbieter in Grössenordnungen vor, die nie ein Mensch zuvor gesehen hat.

http://s3.amazonaws.com/trektv/trektv/ttv_47/trek_tv_47.jpg

Das Cloud-Versprechen: Genug Rechenleistung

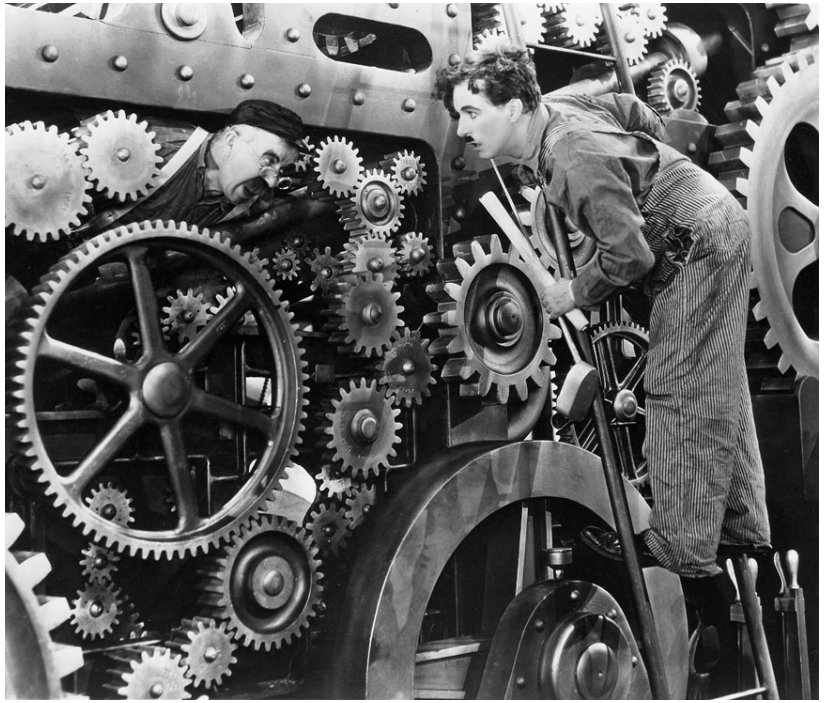
- Die Cloud-Anbieter können schneller ihre Rechenzentren skalieren als dass die Nutzung zunimmt
- Durch die enorme Grösse der Ressourcen mitteln sich Peaks weg
- Braucht man denn noch Scheduling und Queueing und Batch-Systeme?
- Ja:
  - Einmal werden auch HPC Systeme nicht komplett in kommerziellen Clouds verschwinden
  - Dann braucht man auch in Clouds ein Job Management System
  - Kommerzielle Cloud-Anbieter haben einen weiteren Steuerungs-Faktor: Den Preis im Spot-Market, und dazugehöriges „Preemption“.

Übung: Nutzung vom Maxwell Scheduler

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 Teil der kommenden Folien

Nutzer-Sicht auf Storage
Zweiter Teil der kommenden Folien

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

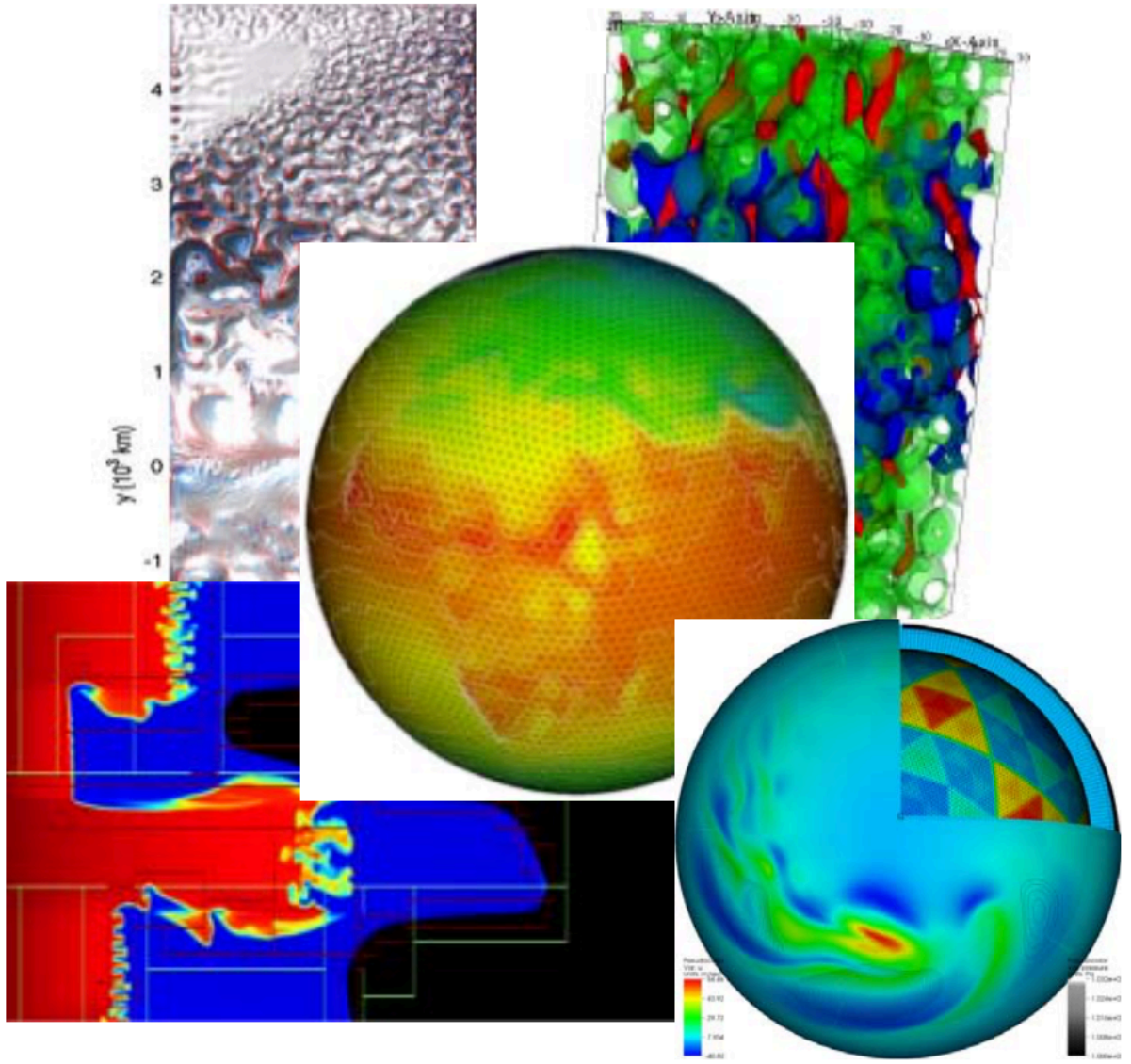
- Input-Daten für Simulation (oder Analyse)
- Lagerung von Output-Daten einer Simulation (oder Analyse)
- Projekt-Daten für Zwischenschritte
- Nutzer-“$HOME“

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

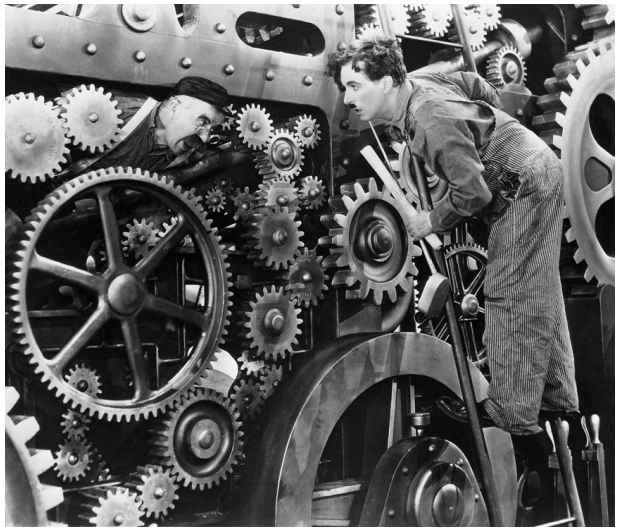
Storage & IO

- 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-

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, ...

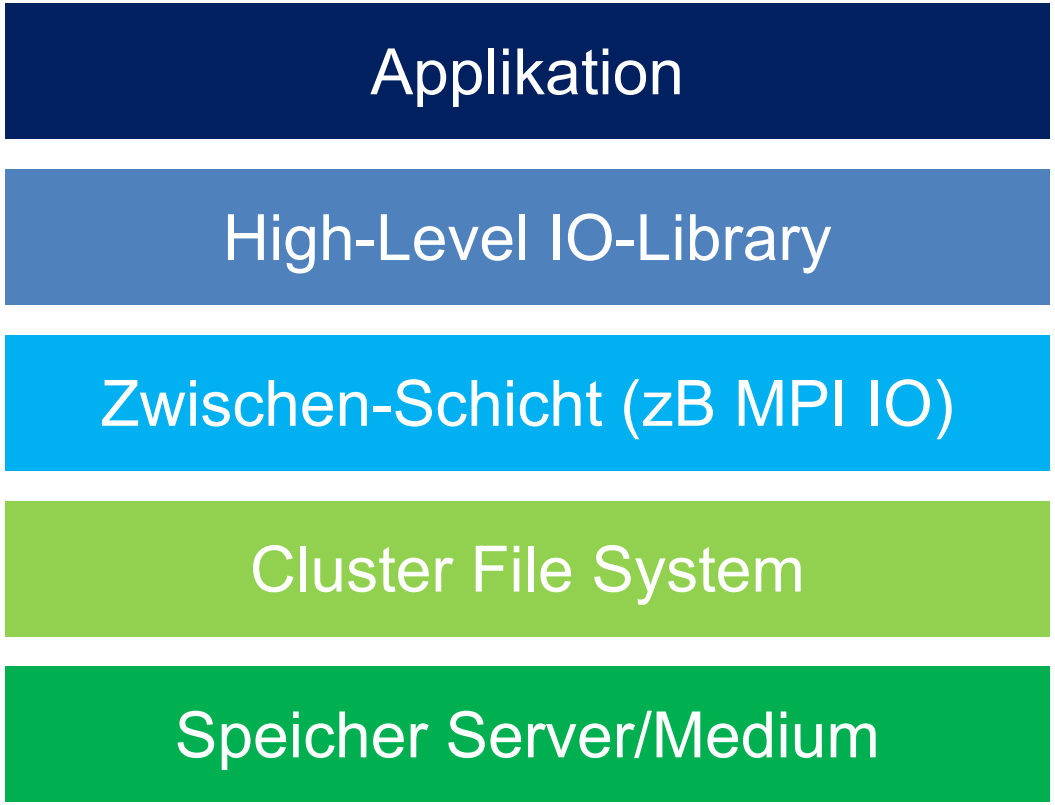
http://selfstorageindavis.com

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, Handelsname Optane) / ...
  - 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

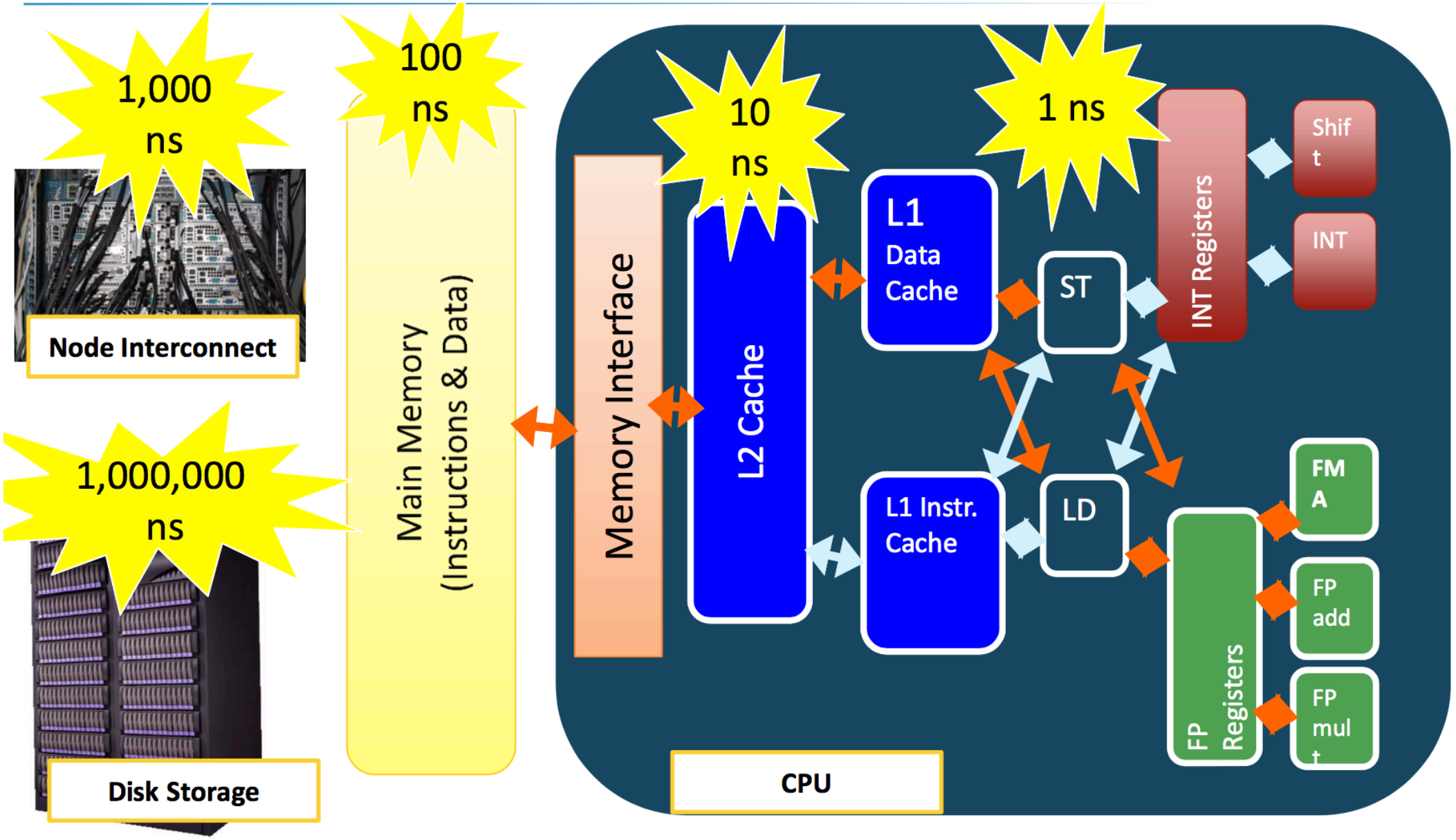
Eine Festplatte macht noch kein Storage-System

- <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



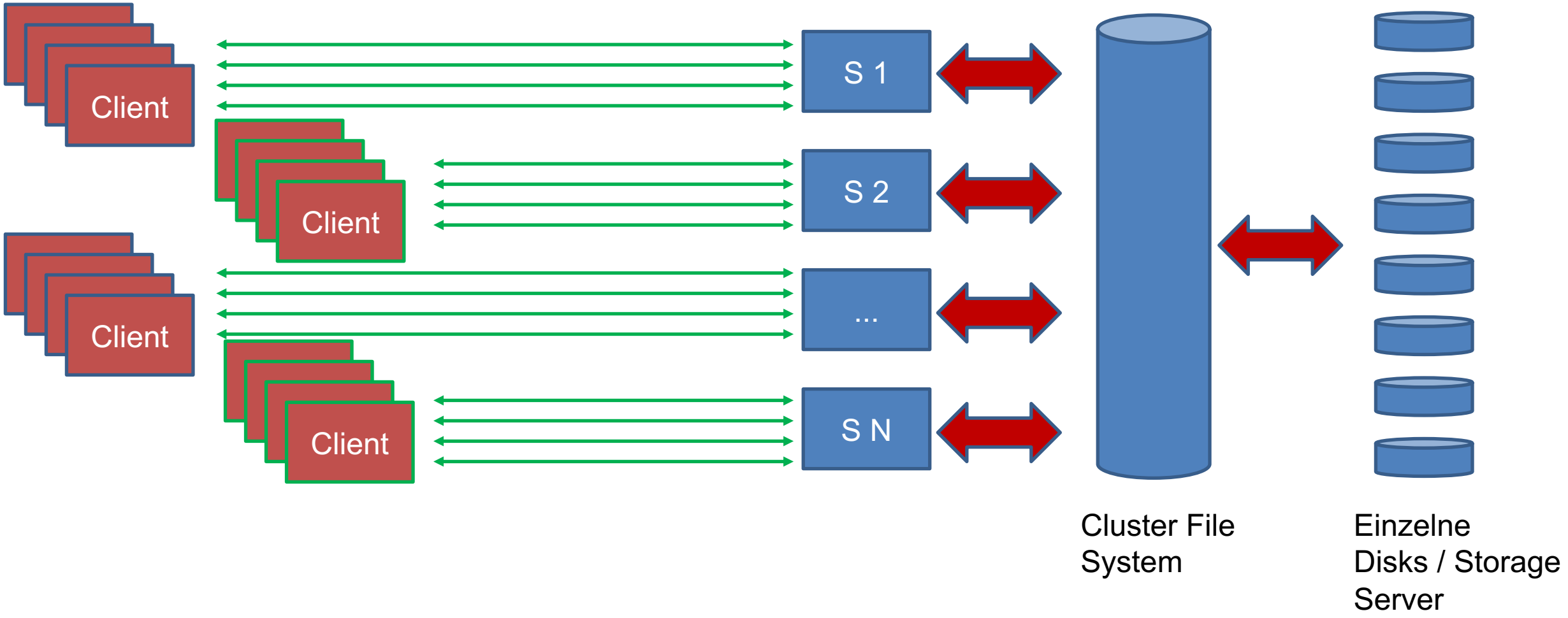
Richard Gerber

Bandbreiten

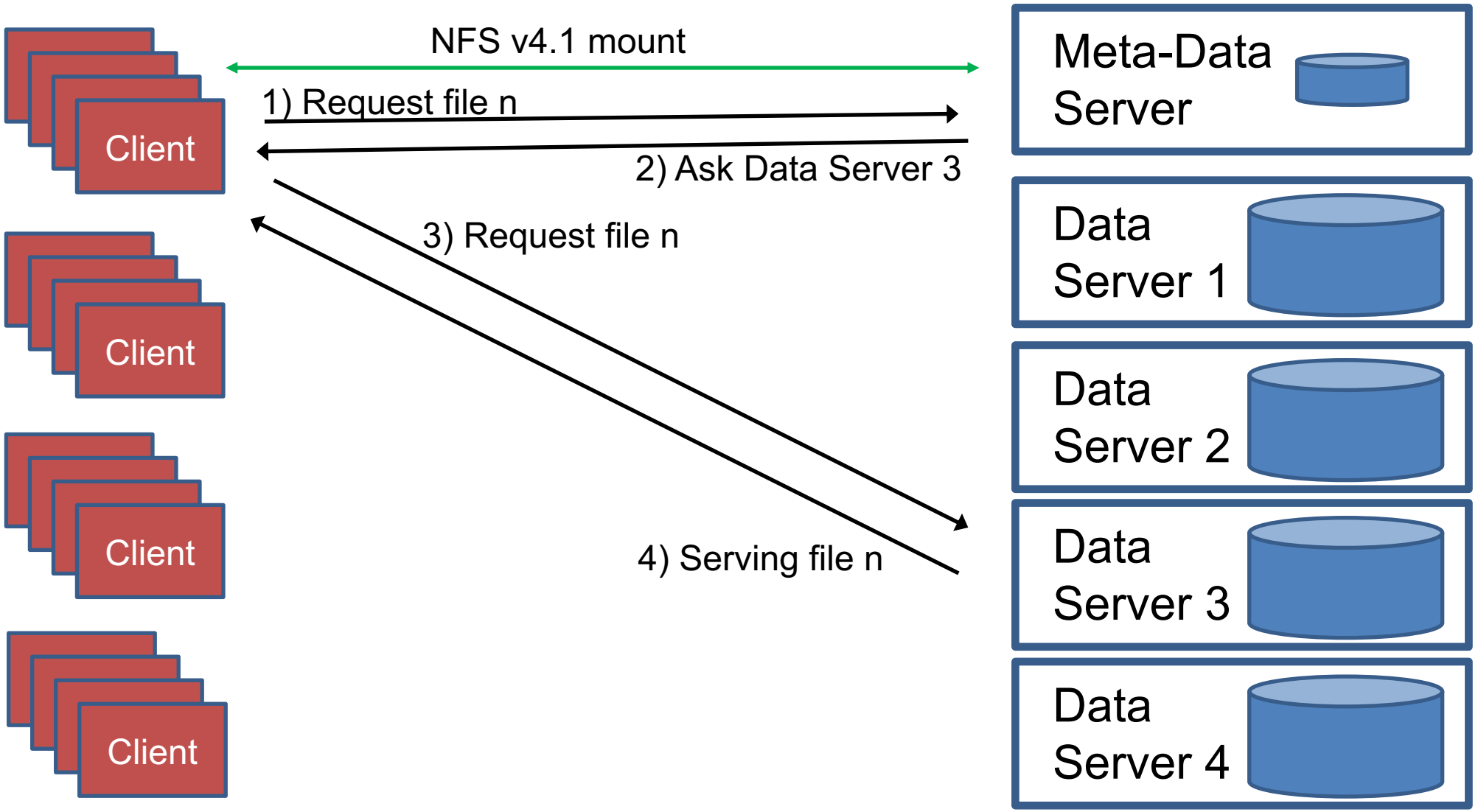
Wie schnell können Daten gestreamt werden von/zu Platte?

- 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!

Skalierung in NFS v3 und NFS v4.0, Beispiel



Skalierung mittels NFS v4.1 / pNFS



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

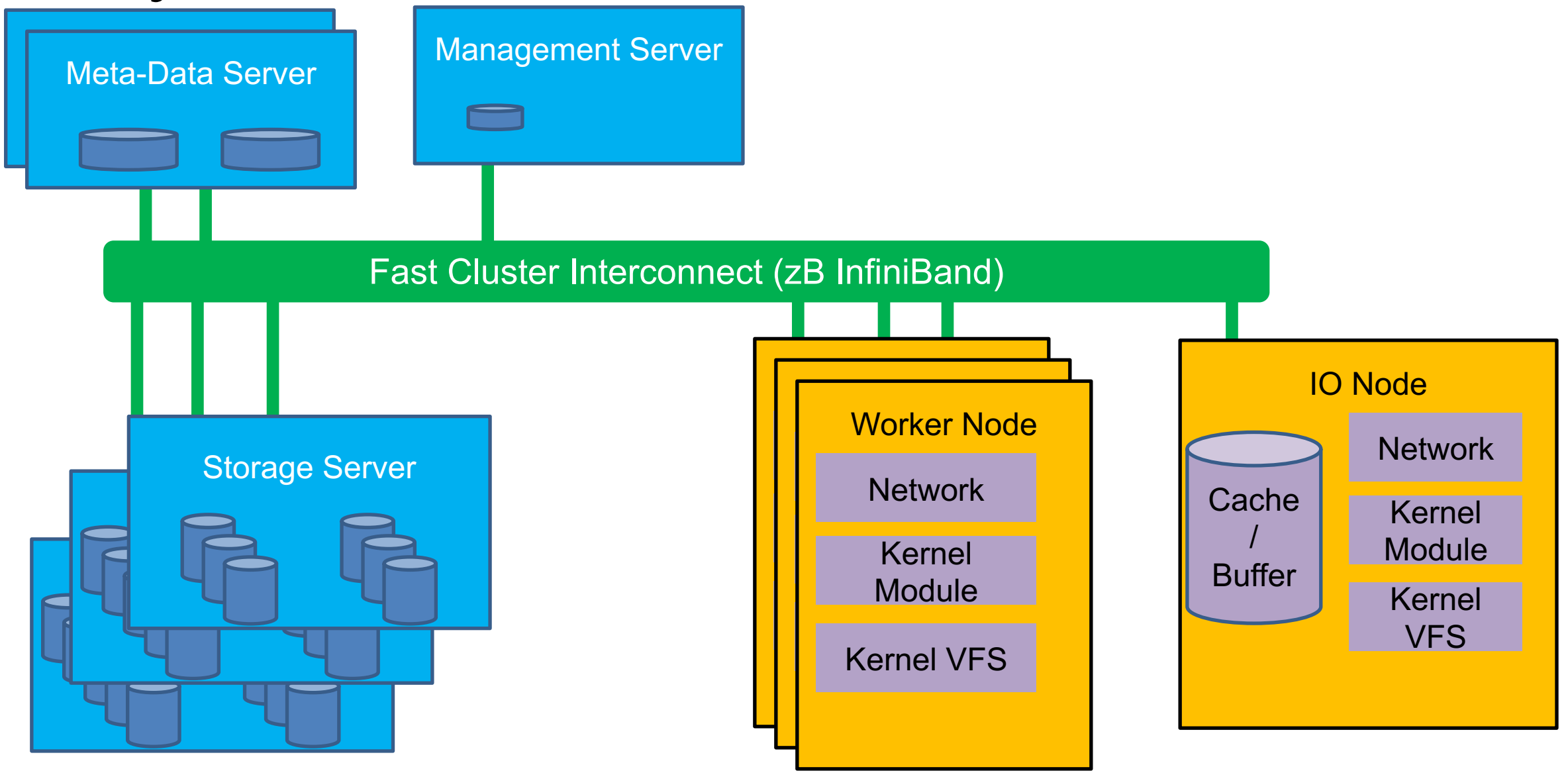
- Am DESY im Einsatz
- Lustre: HPC Storage am Standort DESY/Zeuthen
- BeeGFS: Günstiger Projekt-Space im HPC Cluster DESY/HH
- GPFS (Jetzt Spectrum Scale): Online-Datennahme der neueren DESY Experimente, schnelle Analyse

lustre®
File System

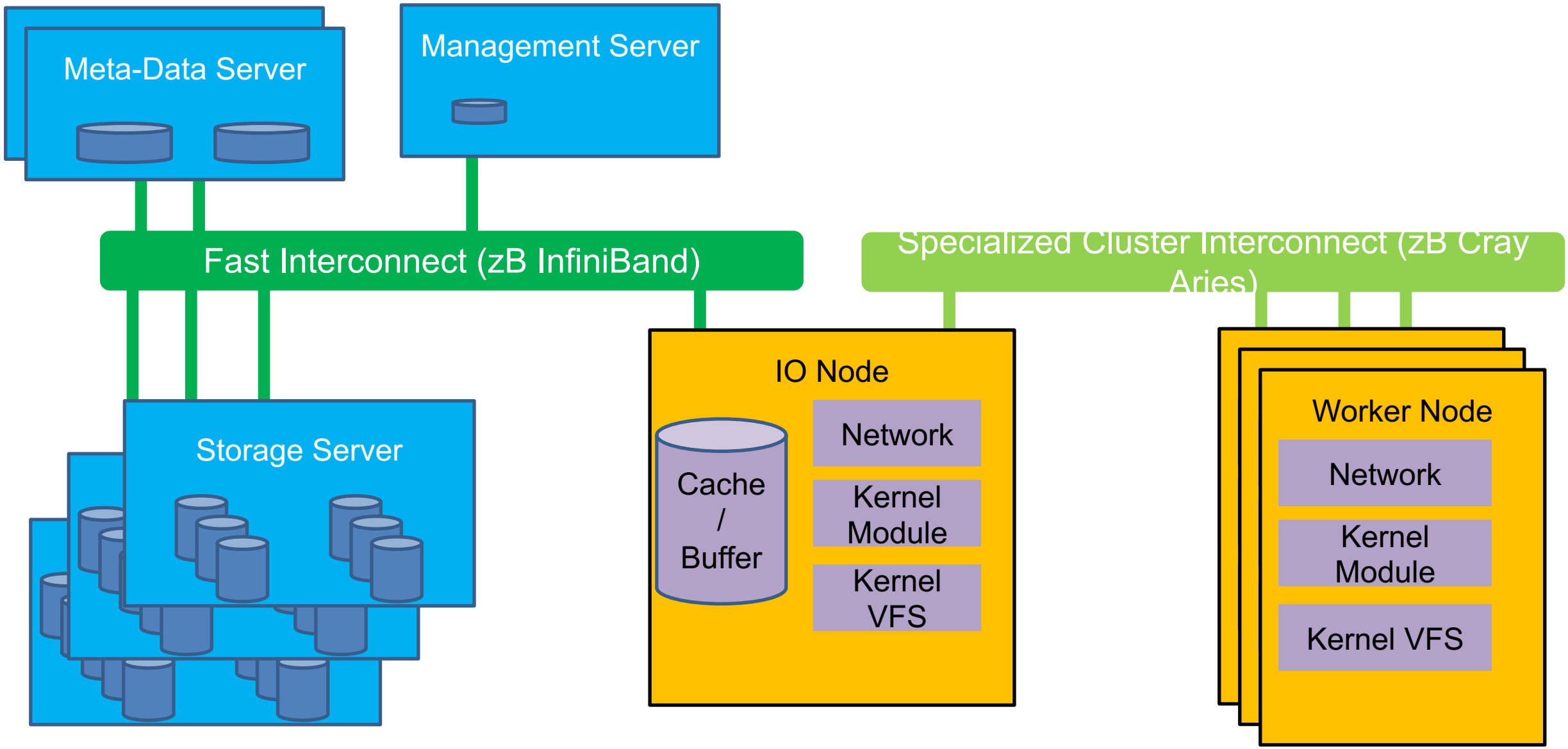
BeeGFS®

IBM®
GPFS

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

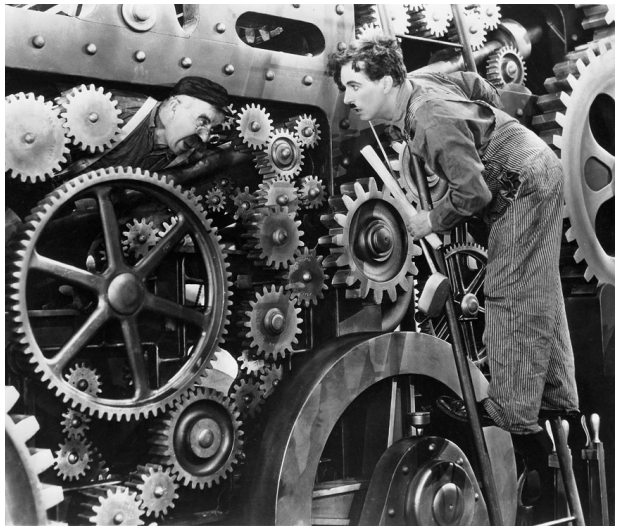
- Netzwerktopologisch sehr nahe an den Compute-Nodes
- Dadurch hohe Bandbreite, zB CORI @NERSC
  - „approximately 1.7 TB/second of peak I/O performance with 28M IOPs, and about 1.8PB of storage“
  - http://www.nersc.gov/users/computational-systems/cori/burst-buffer/burst-buffer/
- Wichtig zB für Snapshots / 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, ...

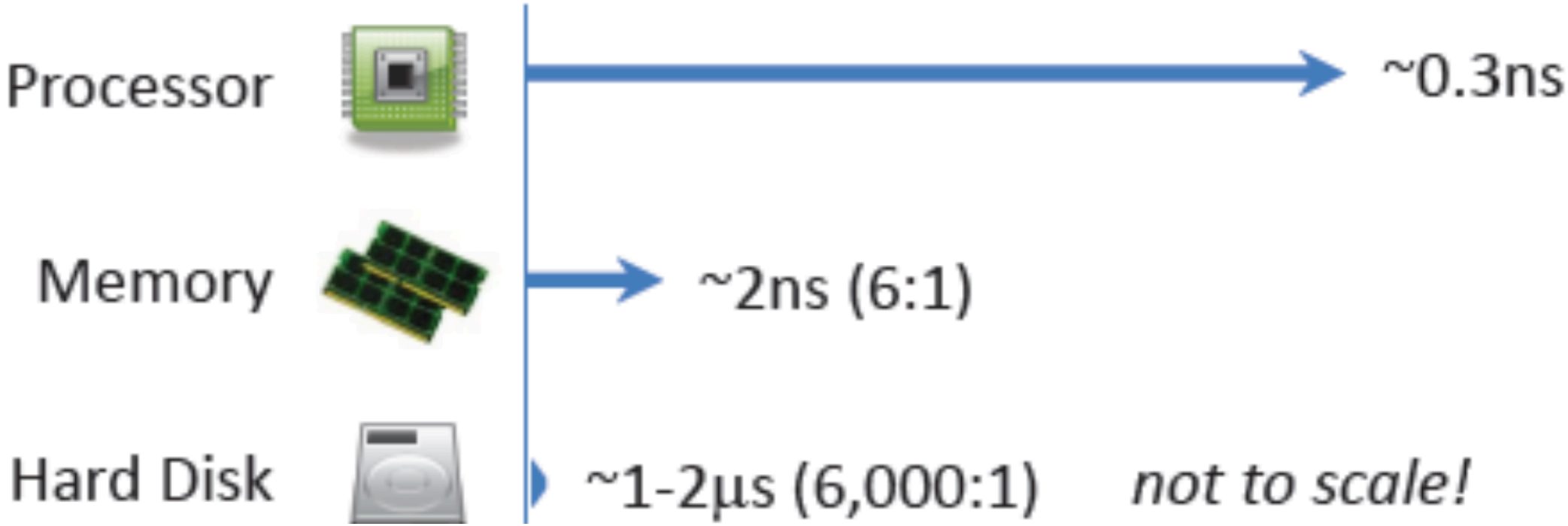
Admin:
Daten? Ja, das Zeugs was von einem Worker-Node auf einen Storage-Server Transferiert wird und dort auf Speichermedien liegt

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

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>`

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!

Staging von Input/Output-Dateien

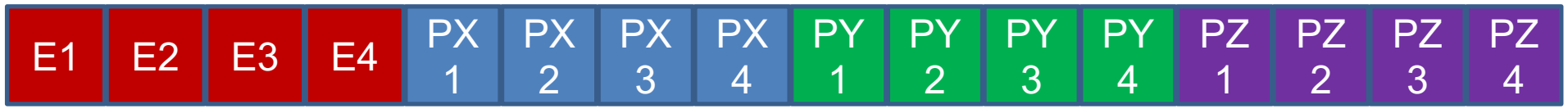
- Kopieren vor/nach dem Job
- Bei Input-Dateien wird eventuell mehr kopiert als unbedingt benötigt
- Kopieren ganzer Dateien ist aber Streaming
- Benchmarken Sie ggf ihre Applikation. Je nachdem wieviel Sie von der Datei lesen ist „staging vor Jobstart“ oder „direktes Lesen vom Netzwerk“ die bessere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

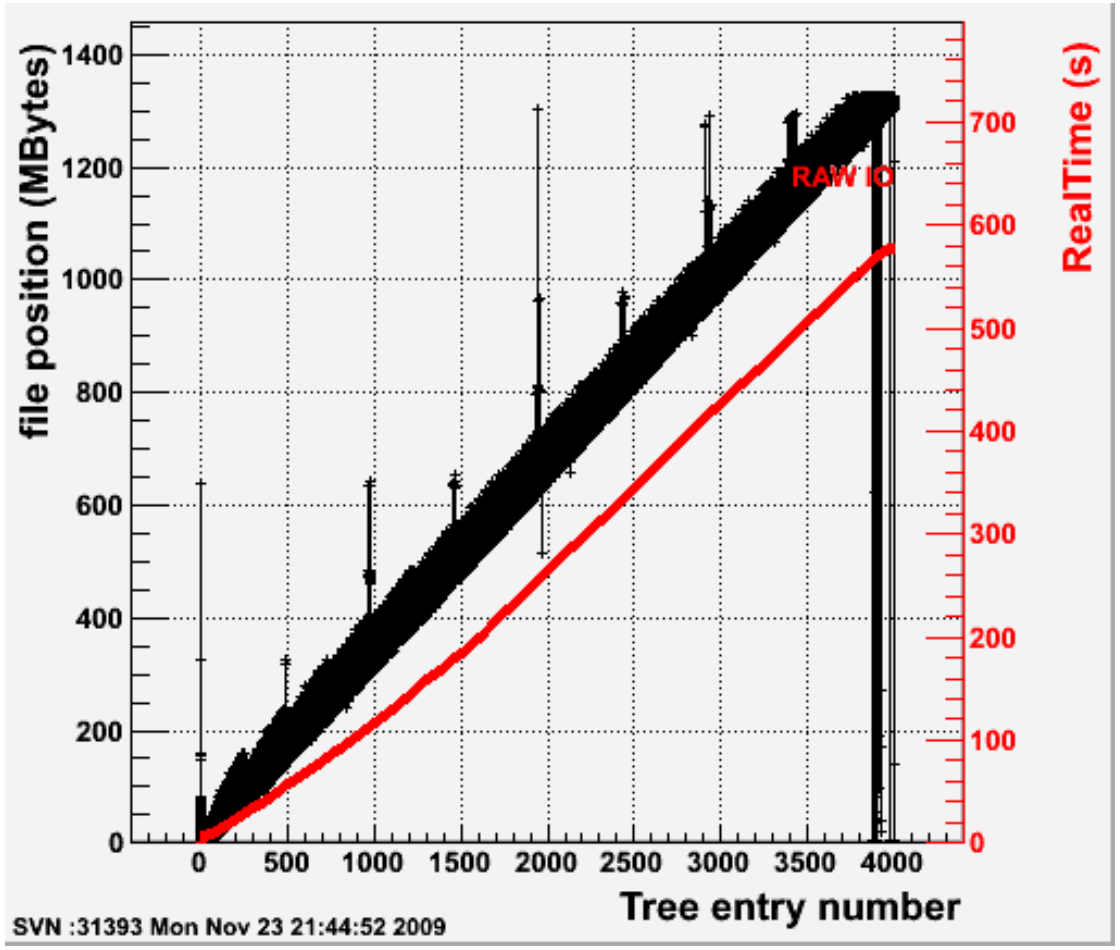
Gruppierung nach Variable?



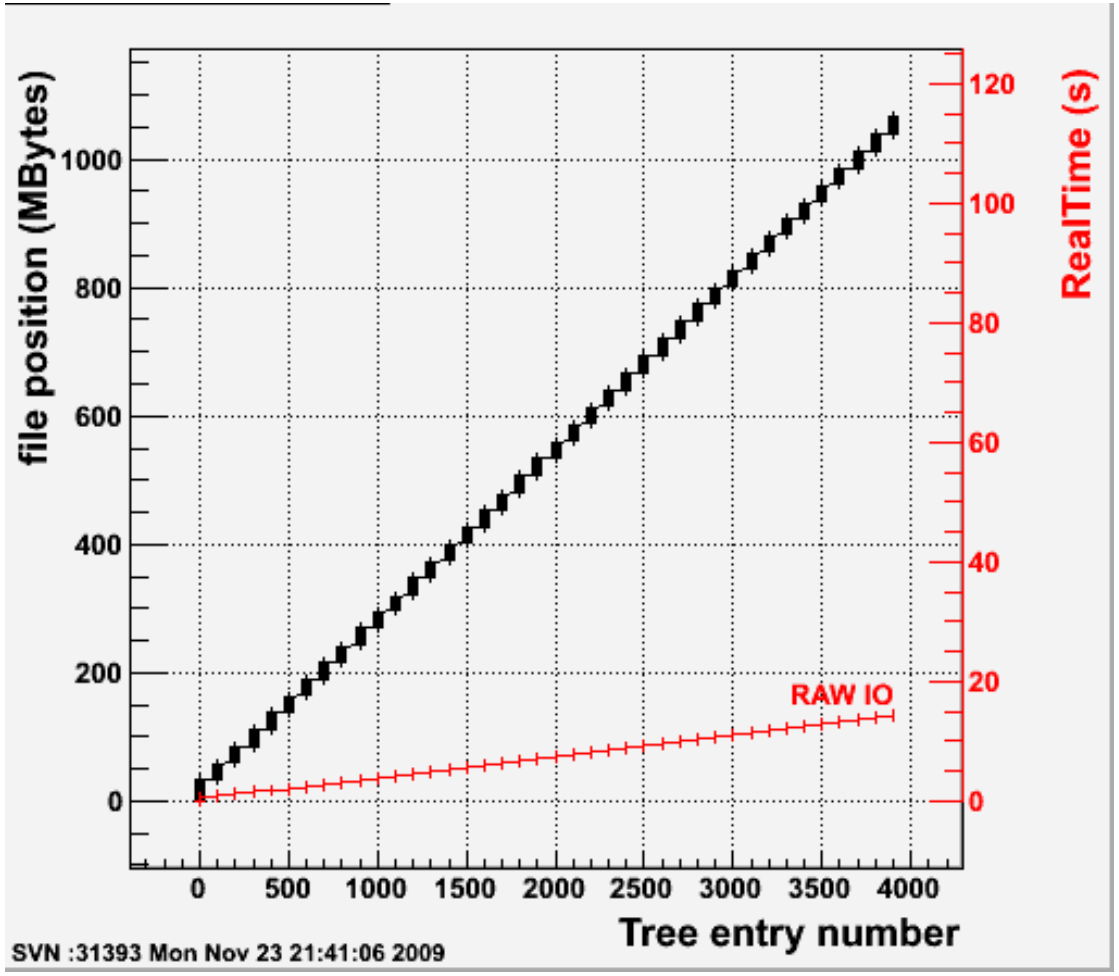
Hängt davon ab

- 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

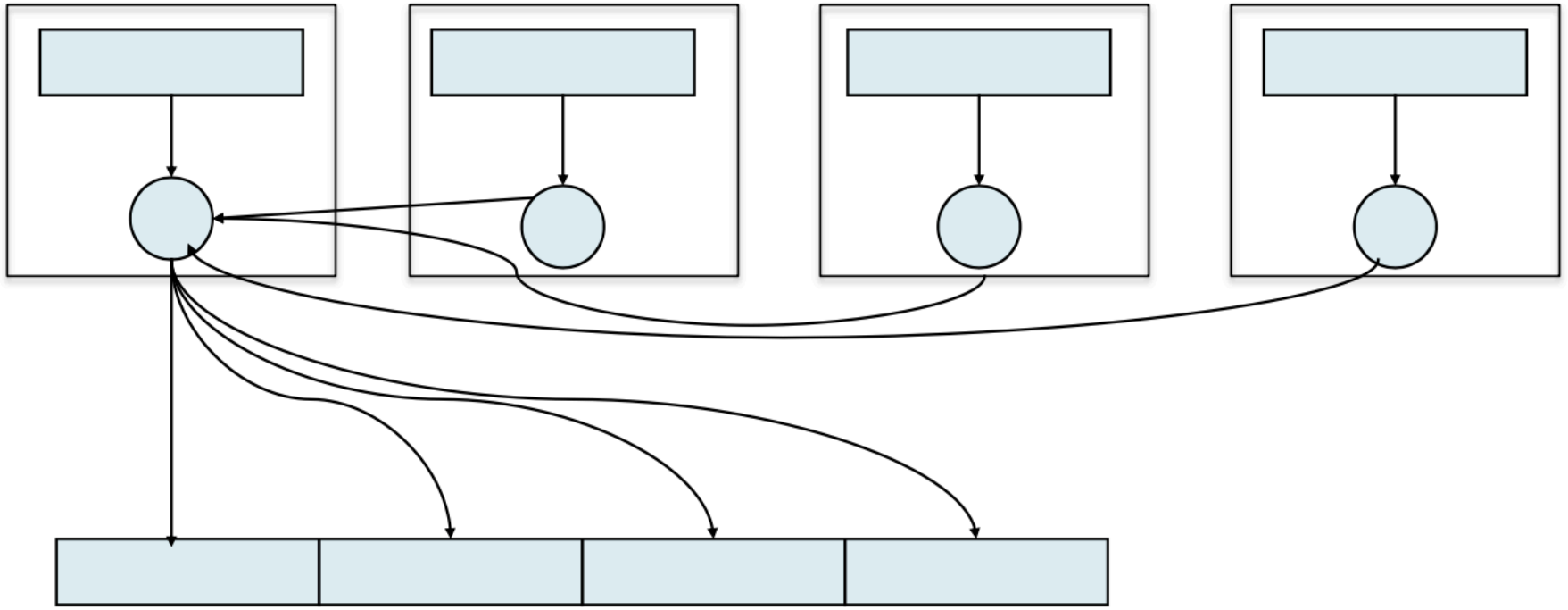


Mit Cache

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

Wie organisiere ich IO?

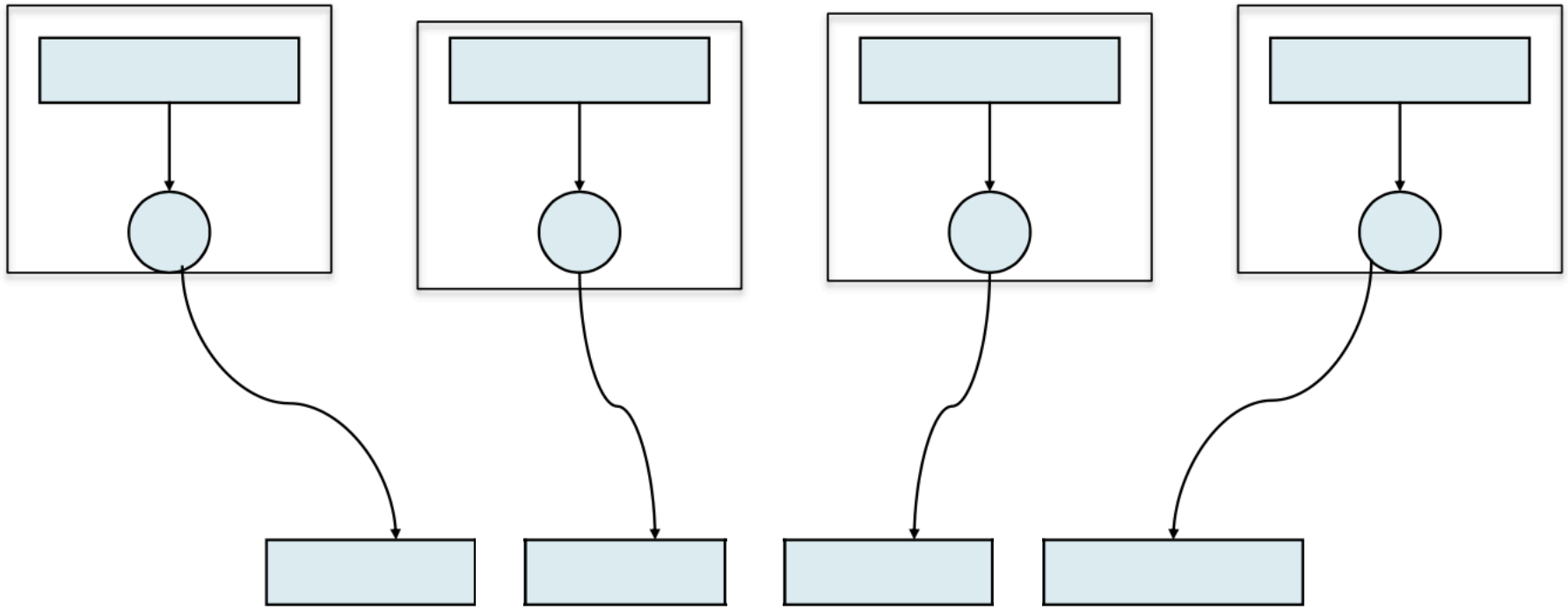
- Kommunikation zum Master, dieser schreibt als Einziger eine Datei - sequentiell



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

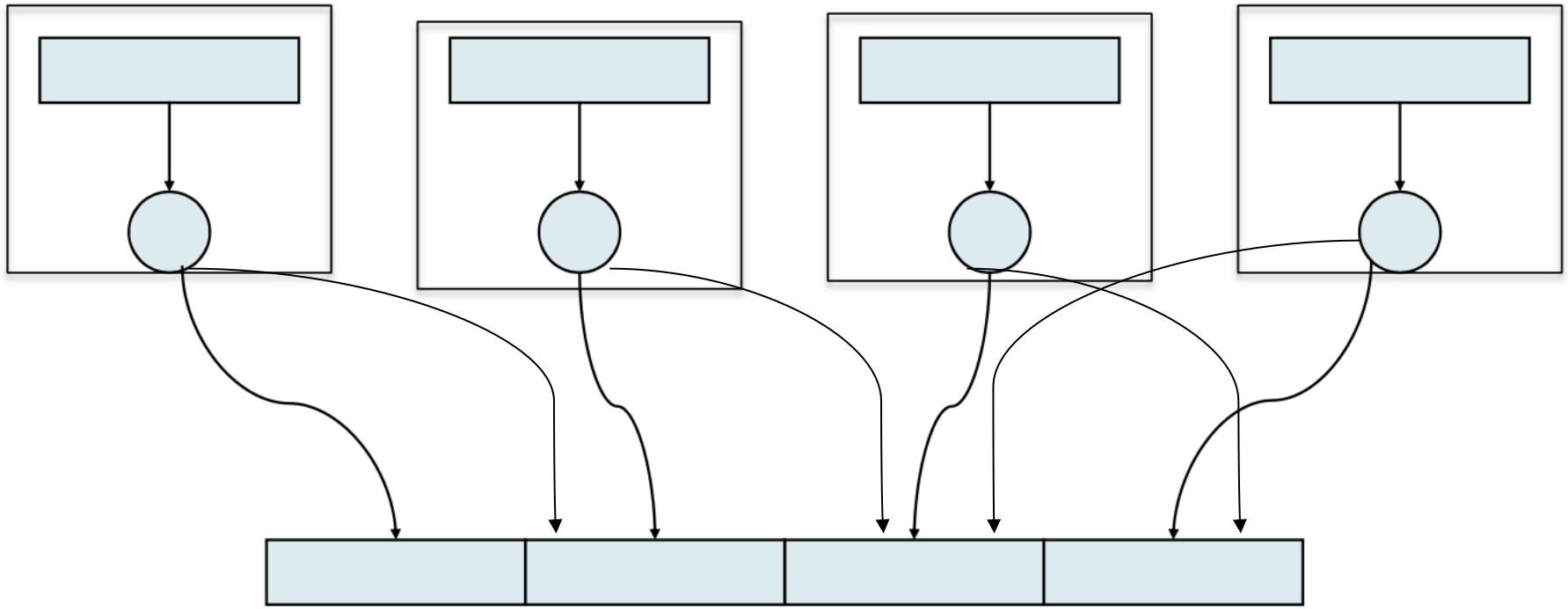
Wie organisiere ich IO?

- Jeder Node beschreibt eine Datei



Wie organisiere ich IO?

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



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 (ROOT, HDF5, ...)
- Versuchen Sie, Frameworks zu benutzen, aber verlieren Sie die Grundlagen nicht aus den Augen!