HPC-Systeme
HPC und Storage
... Die User/Entwickler-Sicht

Prof. Dr. Volker Gülzow

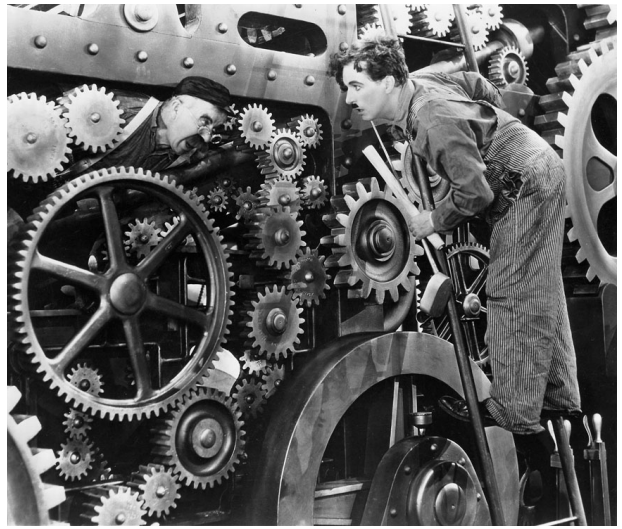
Dr. Yves Kemp

SS 2017

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:



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

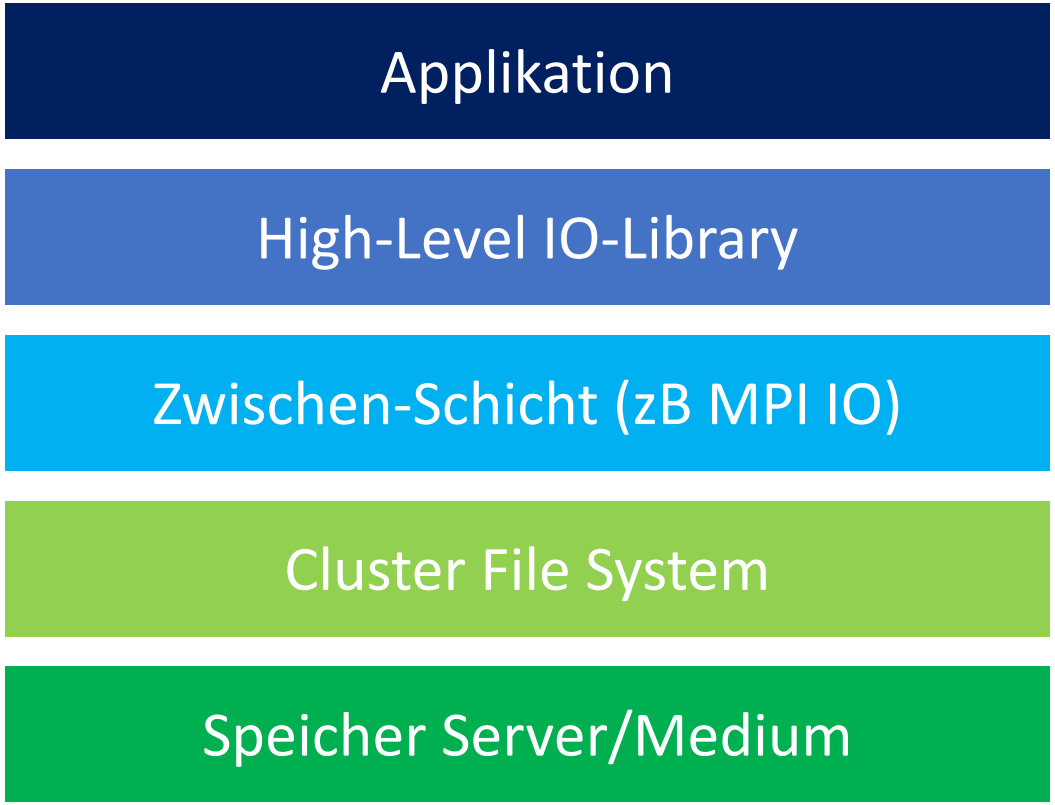
http://selfstorageindavis.com

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

Relationship: It‘s complicated

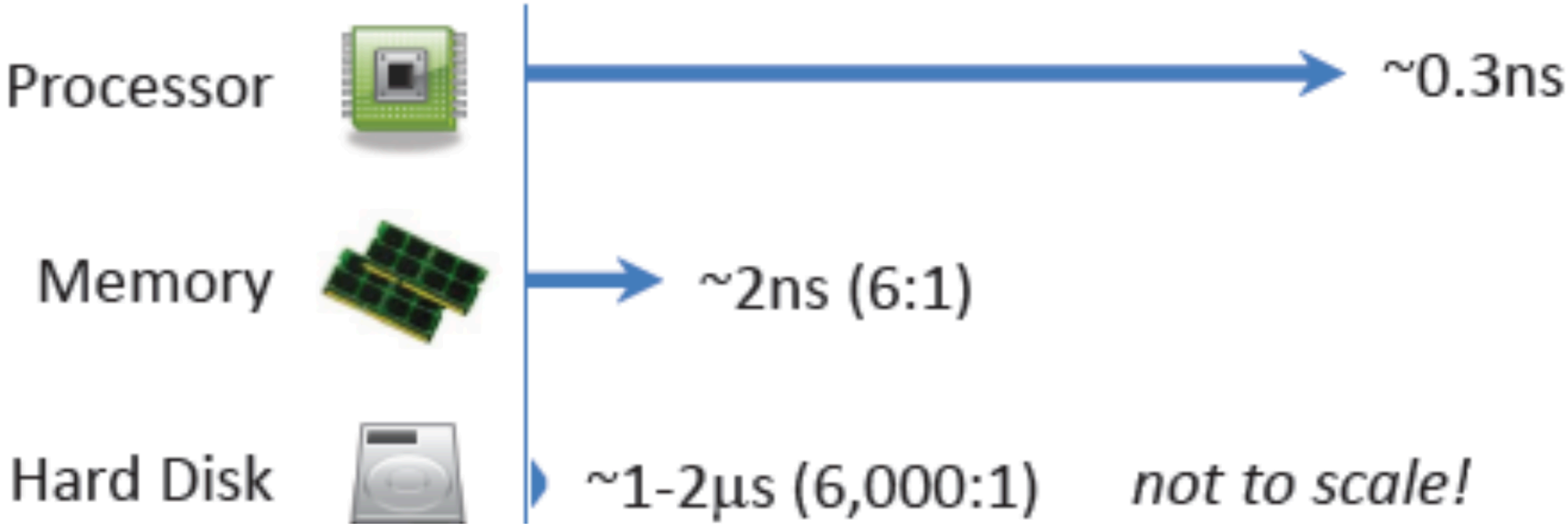
- Nutzer müssen ihre IO Muster verstehen - und ggf anpassen – um gute Performance zu erreichen
- Admins brauchen möglichst detailliertes Wissen über die Anforderungen der Nutzer um Storage-Systeme zu designen und zu tunen
- Manches kann durch spezialisierte IO-Libraries abgefedert werden

IO Hierachie



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 inherente 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 weitesgehent 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

Wieviele Dateien sehen Sie?

```
>ls -la
drwxr-xr-x   5 kemp it         2048 Sep  5  2016 .
drwxrwxrwx 237 kemp it        98304 Jun 24 10:31 ..
-rw-r--r--   1 kemp it         2645 Aug  4  2016 test1
-rw-r--r--   1 kemp it         2822 Sep  5  2016 test2
```

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

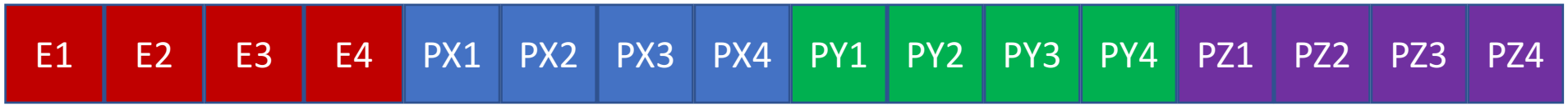
- Kopieren vor/nach dem Job
- Bei Input-Dateien wird eventuell mehr kopiert als unbedingt benötigt
- Kopieren ganzer Dateien ist aber Streaming
- Benchmarken Sie ggf ihre Applikation. Je nachdem wieviel Sie von der Datei lesen ist „staging“ oder „lesen vom Netzwerk“ die bessere Strategie
  - Bedenken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



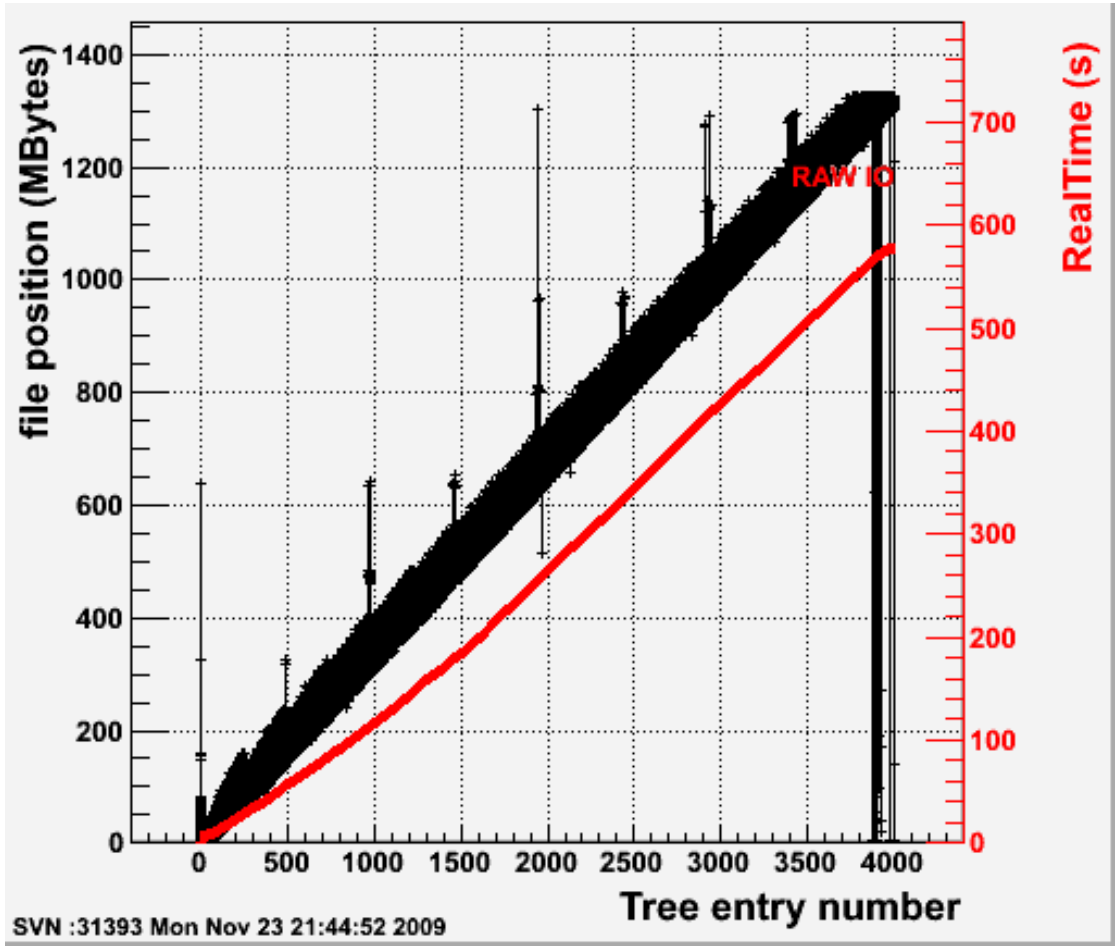
Hängt davon ab

- Loop über die Teilchen, Berechnung von $m^2=E^2-(Px^2+Py^2+Pz^2)$ pro Teilchen

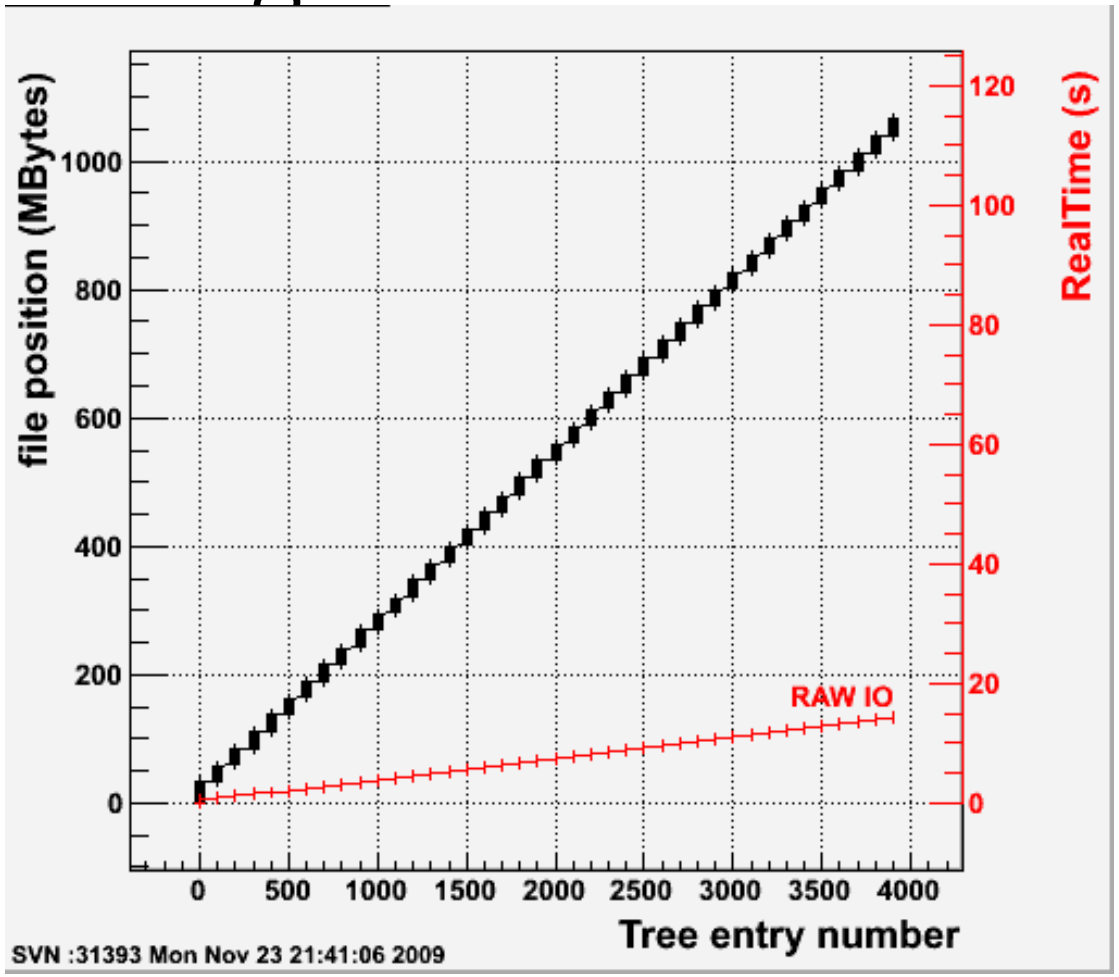
- Loop über die Teilchen, Häufigkeitsverteilung von E

- 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



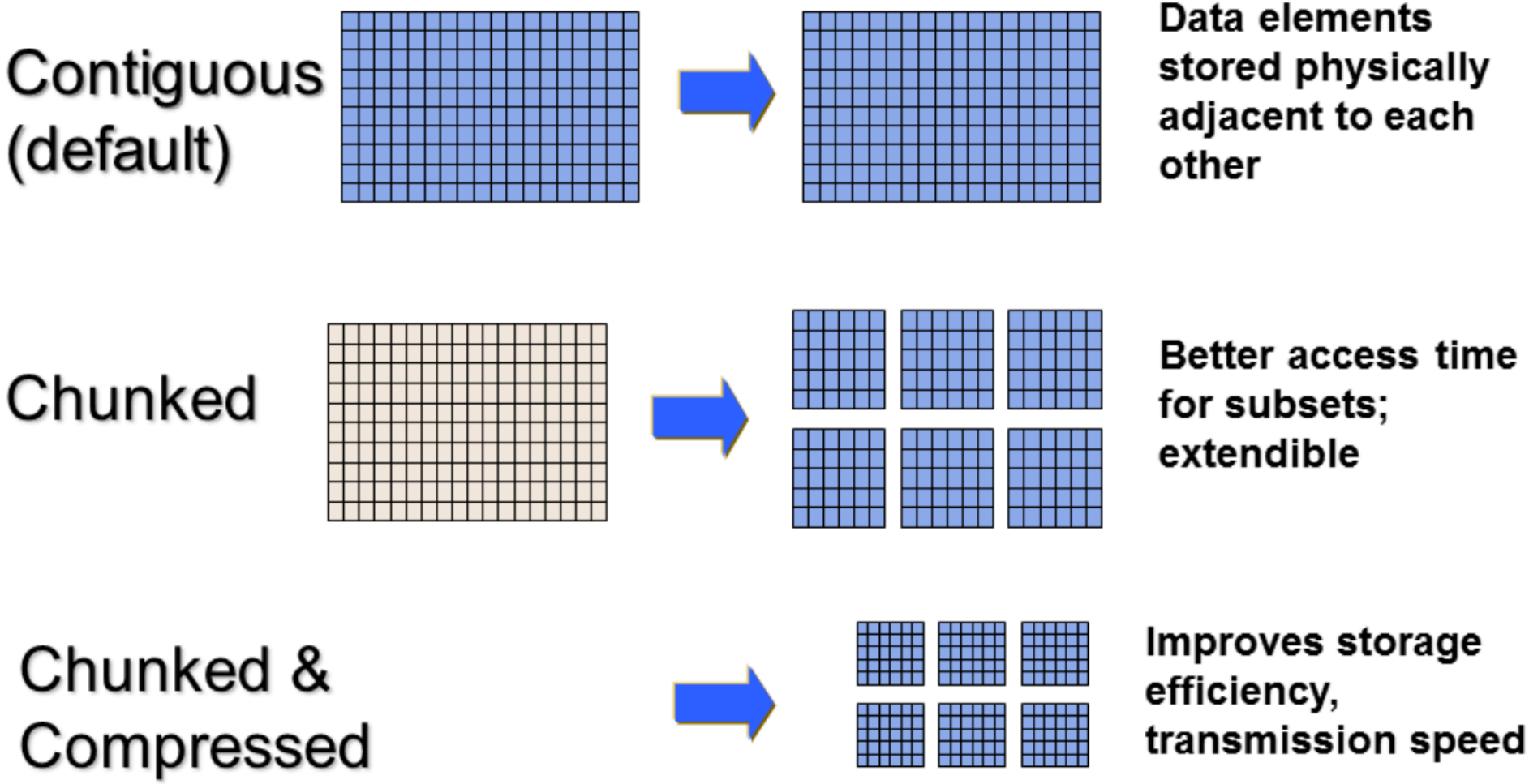
Ohne Cache



Mit Cache

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

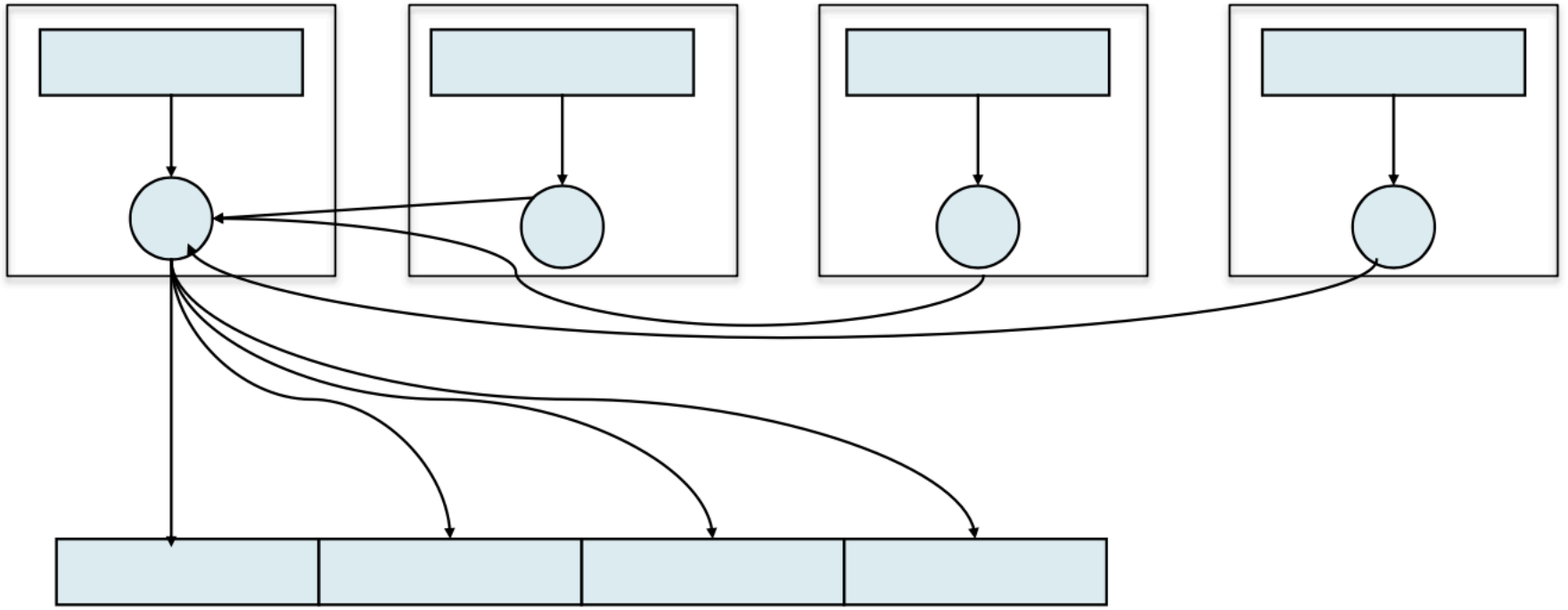
Beispiel: HDF5



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

Wie organisiere ich IO?

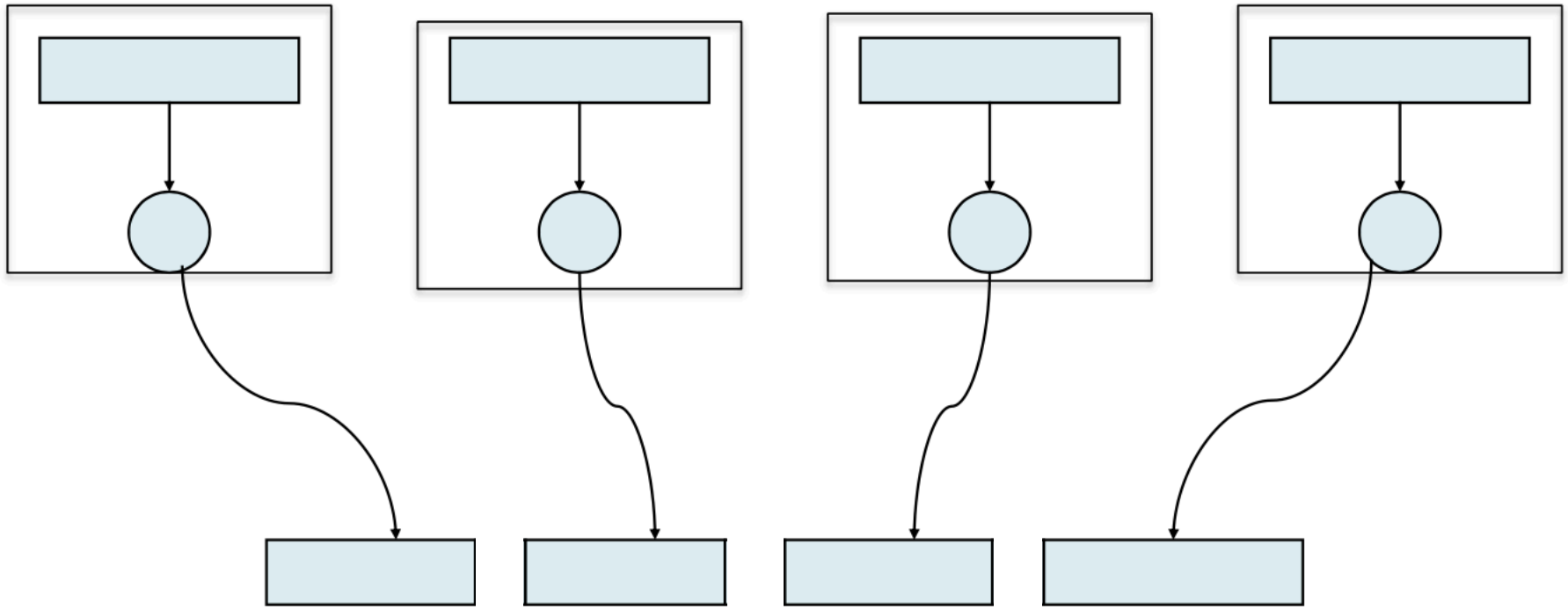
- Kommunikation zum Master, dieser schreibt als Einziger eine Datei - sequentiell



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

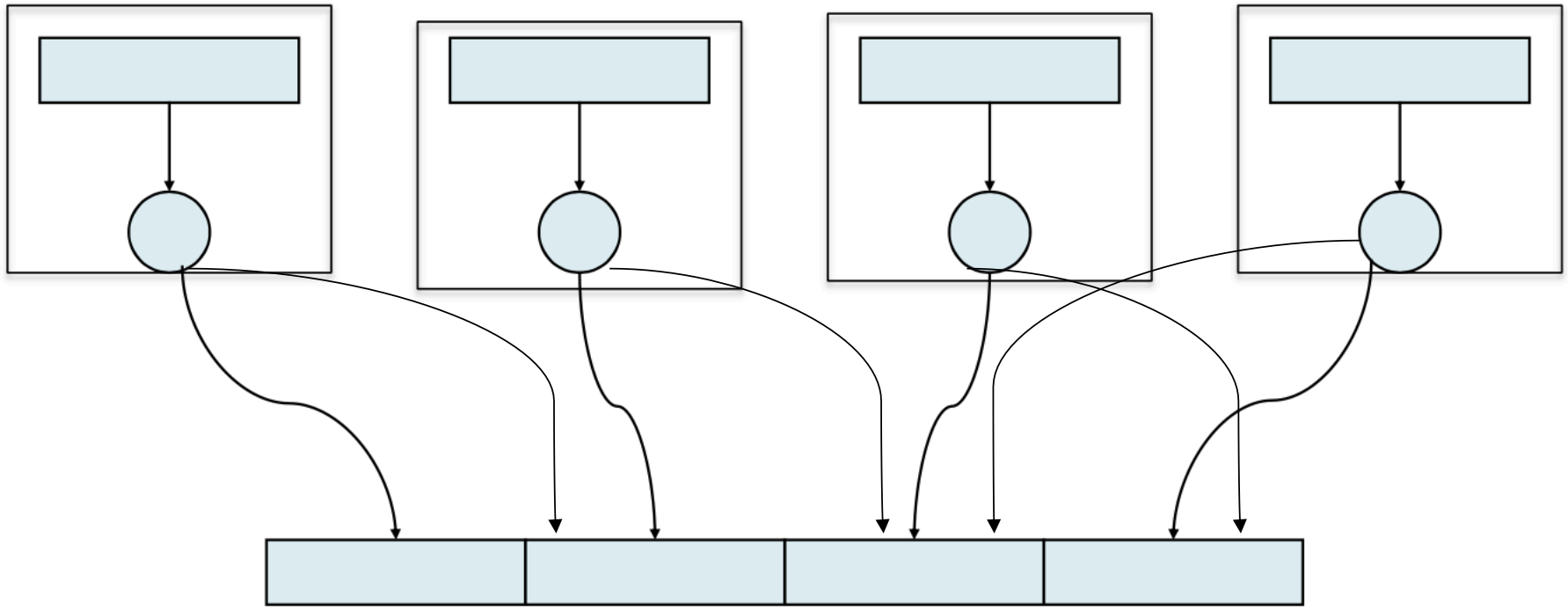
Wie organisiere ich IO?

- 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

- Aber zuerst eine Kurzvorstellung von MPI-IO

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

Kurzvorstellung

- Entstanden ab 1994 am CERN
- C++ basierender Nachfolger eines Fortran-Analyse-Paketes
- Hoch-Energie und Nuklear-Physik lastig, aber auch in der Medizin und Finanz-Welt
- Framework für Daten-Speicherung und Daten-Analyse
  - Mit vielen hilfreichen Werkzeugen:
  - Command Line Interpreter, Histograms and Fitting, Writing a Graphical User Interface, 2D Graphics, Input/Output , Collection Classes, Script Processor.
  - 3D Graphics, Parallel Processing (PROOF), Run Time Type Identification (RTTI), Socket and Network Communication, Threads
- http://root.cern.ch/

Etwas längere Vorstellung von HDF5

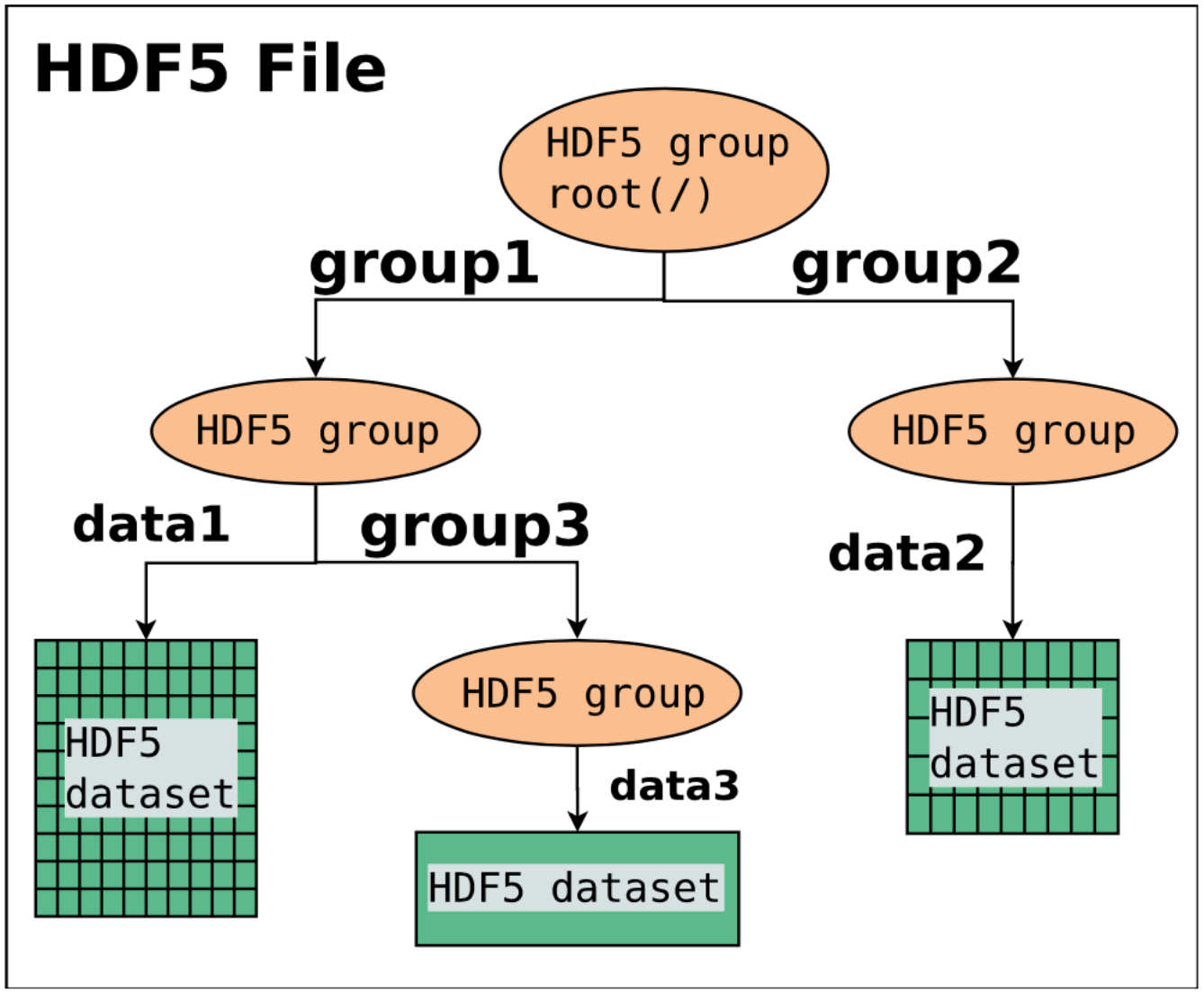
- HDF: Hierarchical Data Format
- Entstanden aus der Notwendigkeit eines portablen Datenformat für wissenschaftliche Daten
  - NCSA und NASA und andere grosse Organisationen
- HDF5: Aktuelle Version (seit 2002)
- Grundlage für weitere FOrmate
  - NetCDF, Nexus, ...

HDF5 interne Struktur

Datasets: Multidimensional arrays eines homogeneous Typ

Groups: Container Strukturen für datasets und andere Gruppen

Data is organized in a tree like structure:

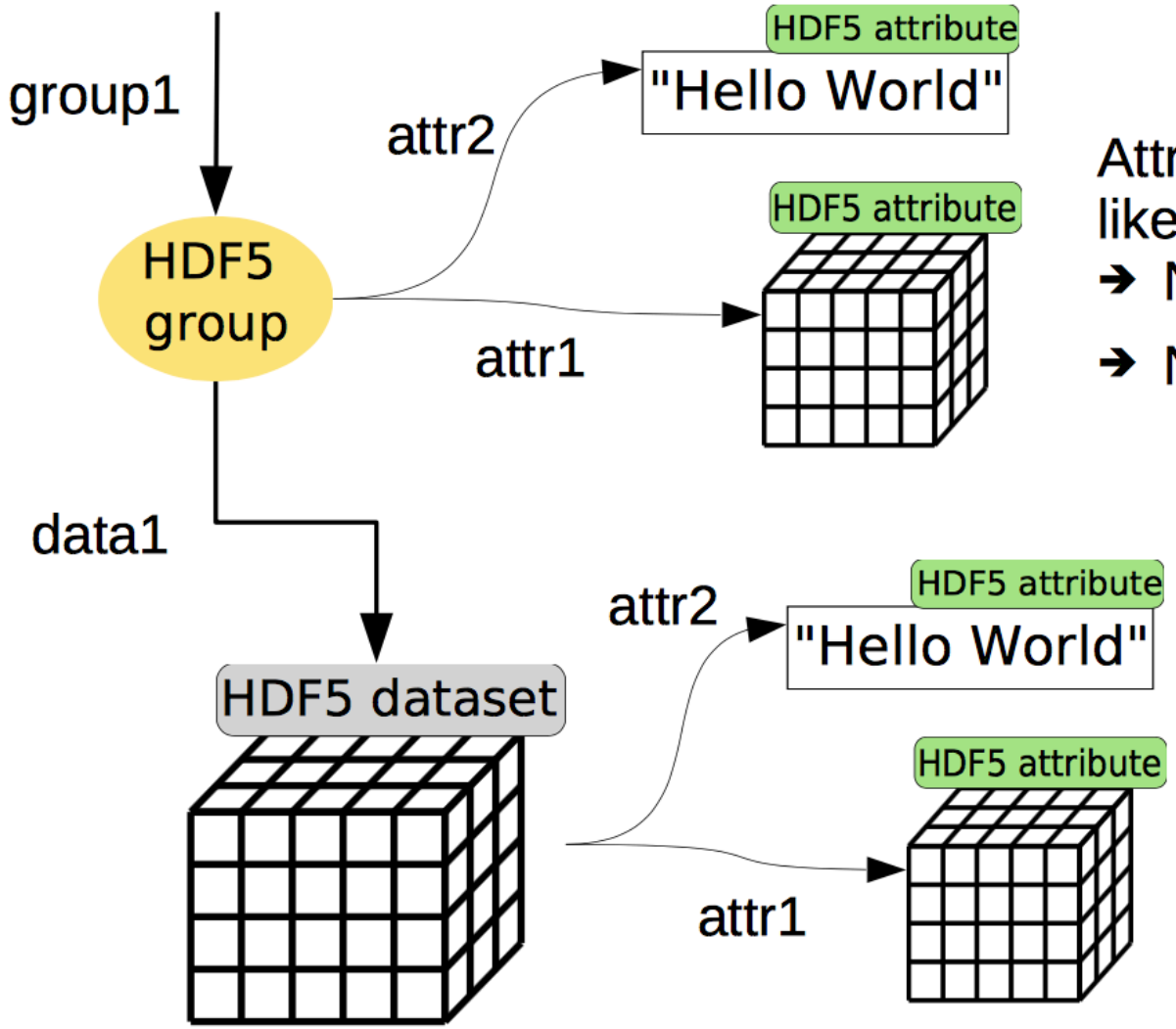


Objects are addressed via paths: `/group1/group3/data3`

http://www.desy.de/dvsem/SS13/wintersberger_talk.pdf

Attribute für Metadaten

- Attribute können an Gruppen und Datasets für Metadaten gebunden werden



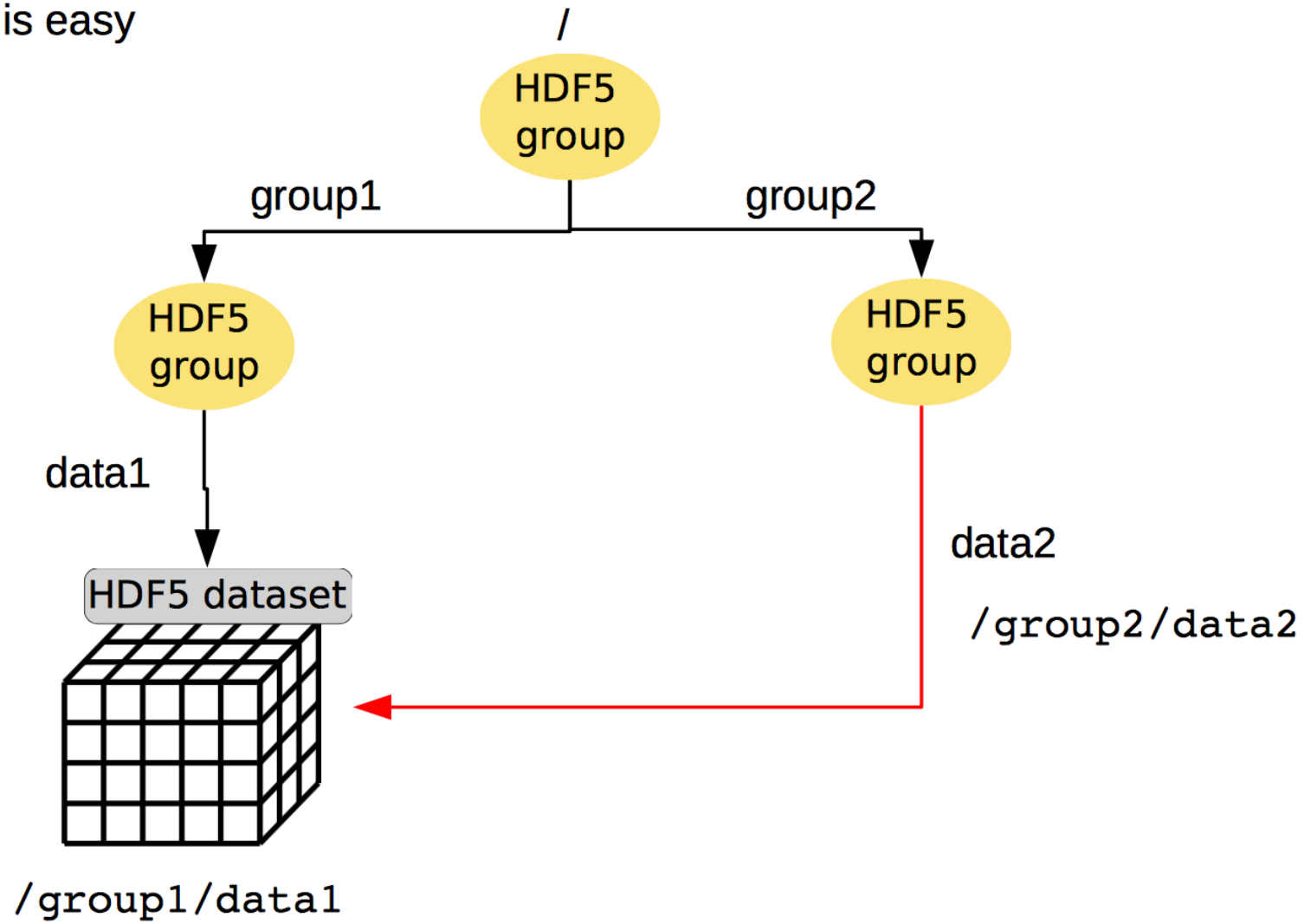
Attributes behave pretty much like datasets, but

➔ No filters

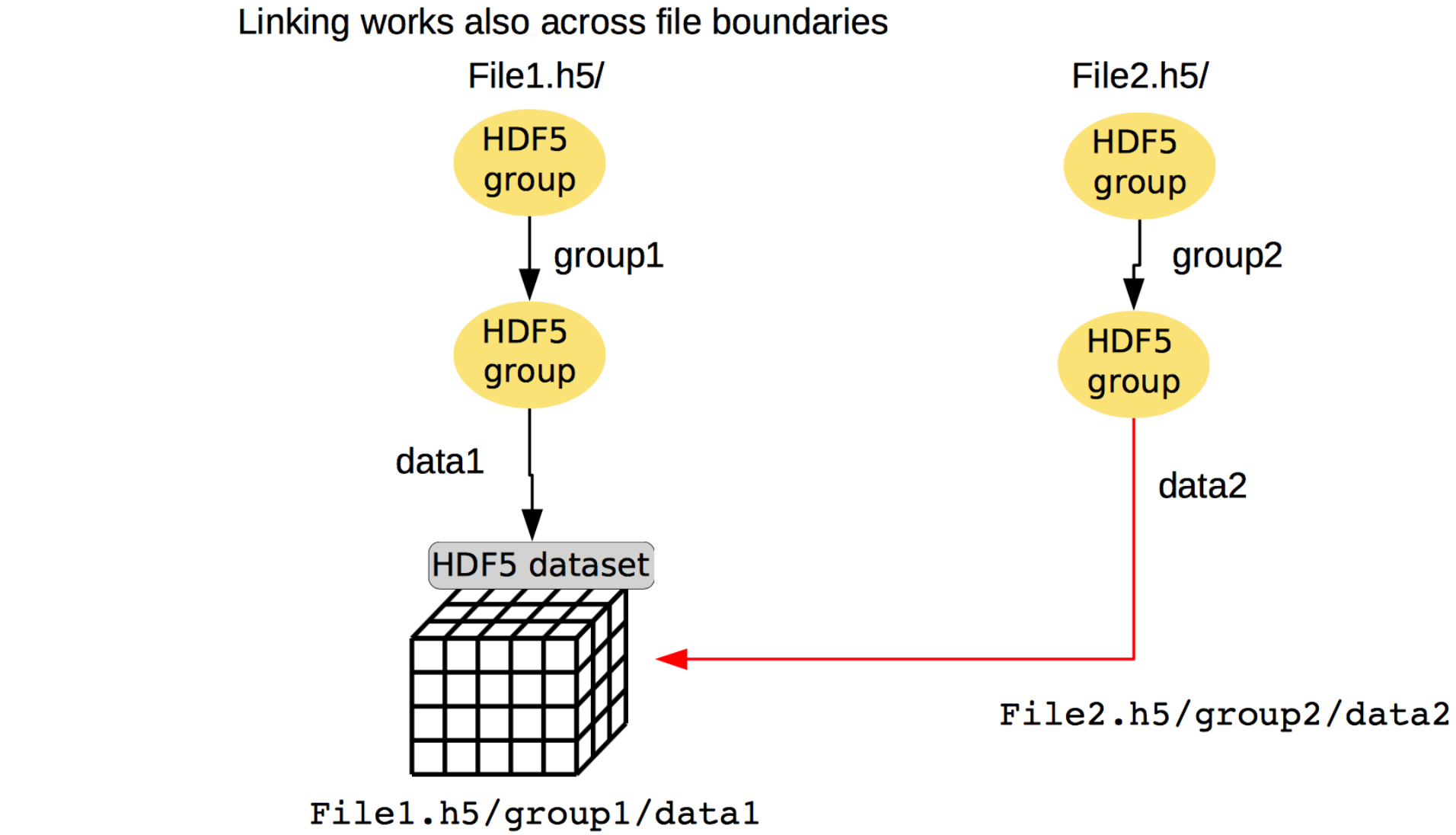
➔ No partial IO

Interne Links

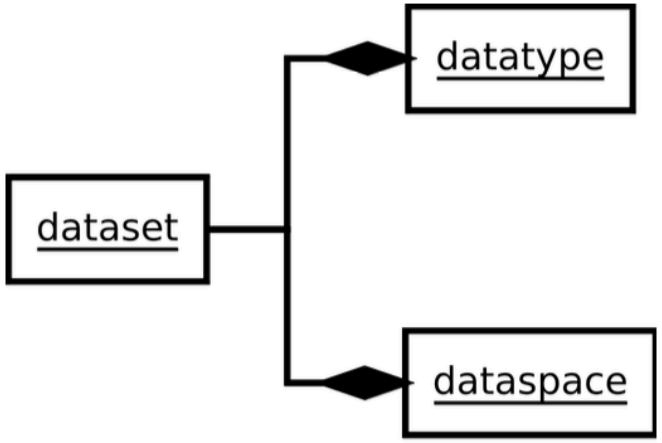
As an HDF5 file can be considered as a bunch of nodes connected by graphs linking is easy



Externe Links



Im Innern eines Datasets

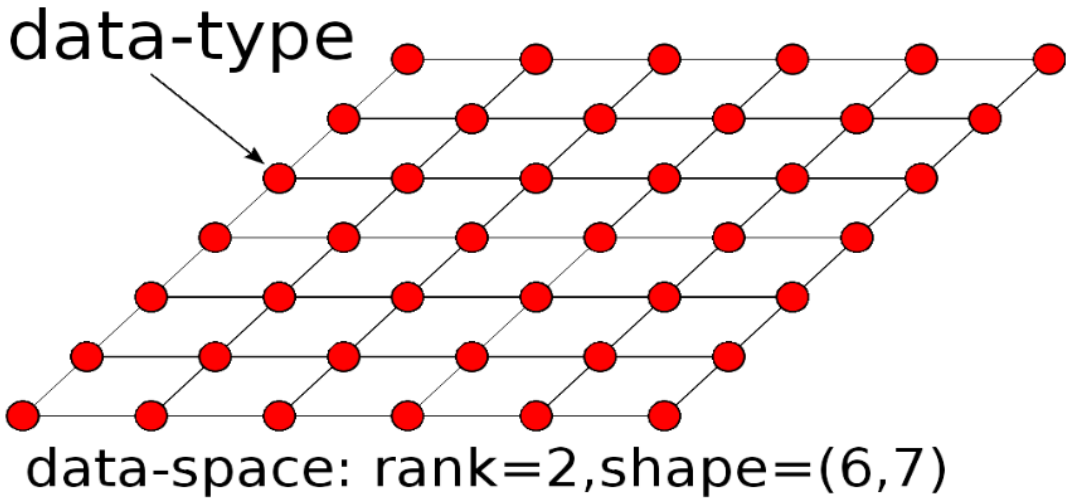


Datatype: what shall be stored.

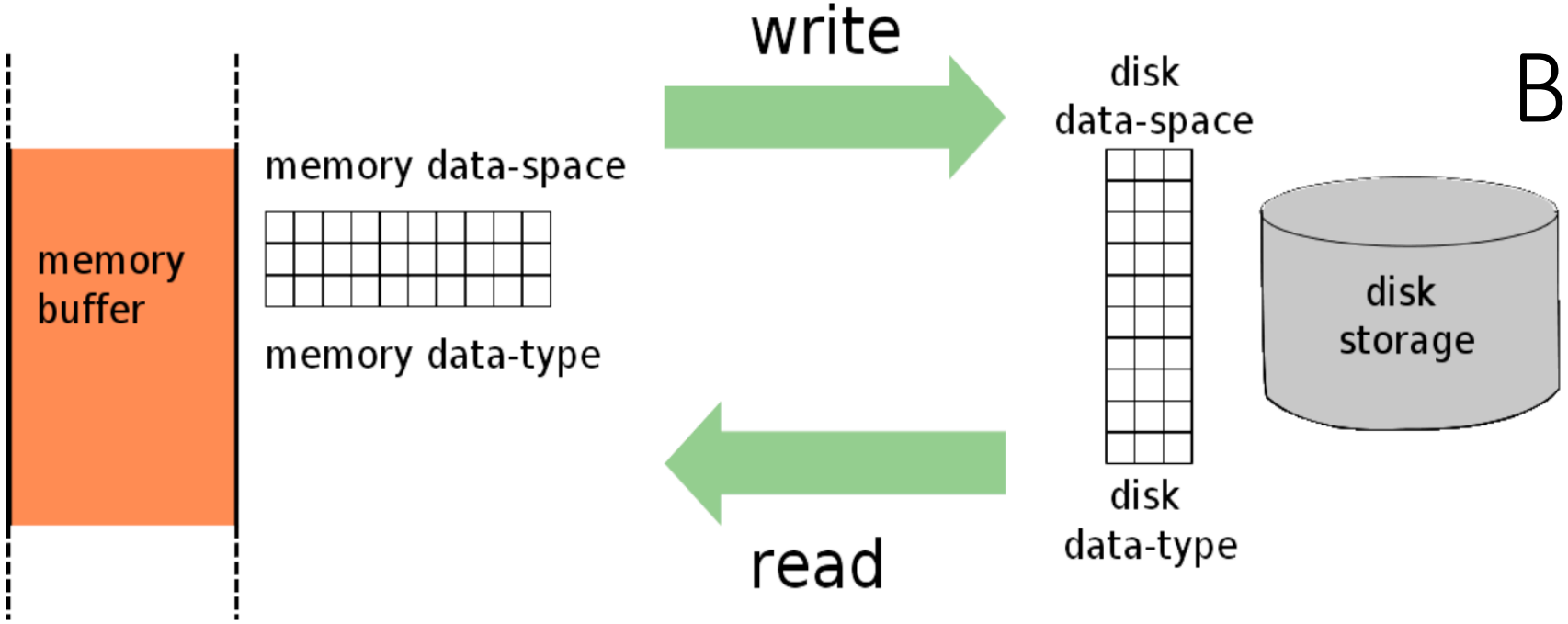
Dataspace: logical organization of the data.

Terminology:

- **rank**: number of dimensions
- **shape**: number of elements along each dimension.



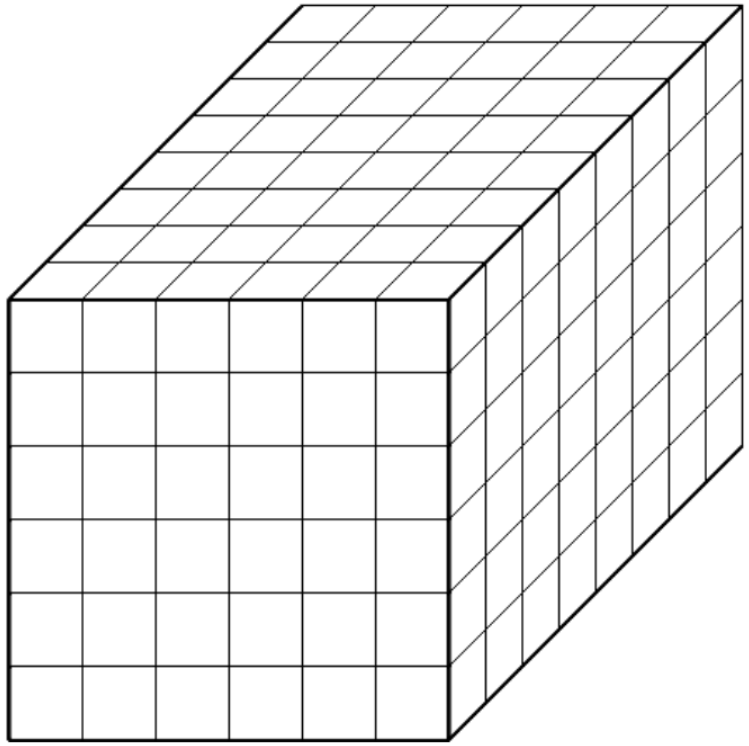
Basic IO mit HDF5



- Aufgaben der HDF5 Library
  - Konvertierung der Daten zwischen Speicher und Platte
  - Eventuell Konvertierung Litte/Big Endian
  - Filter auf Daten anwenden

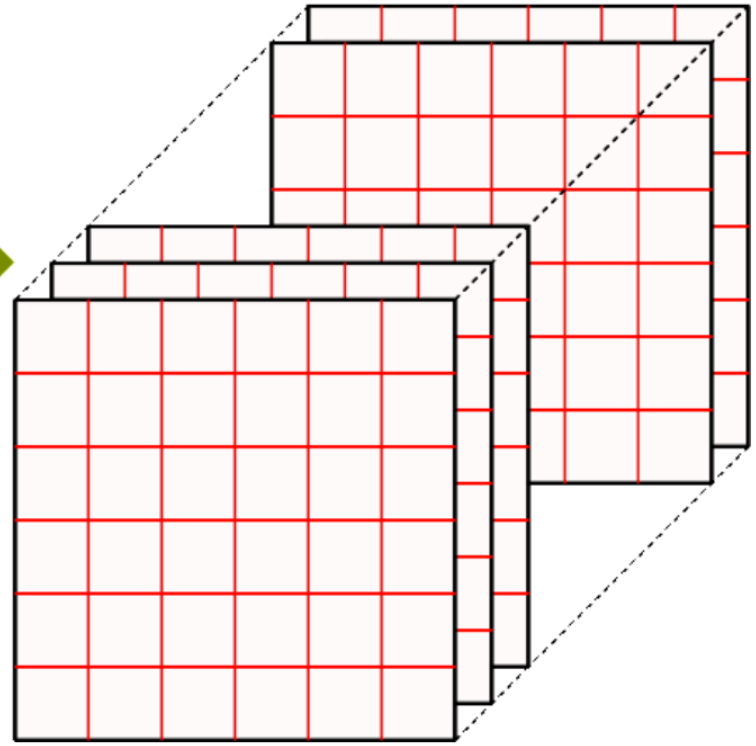
Organisation von grossen Datasets

contiguous layout



- All data is read or written
- Filters are applied to the entire data

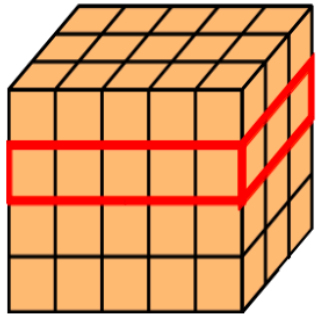
chunked layout



- Data is read/written in portions of chunks
- Filters are applied to individual chunks

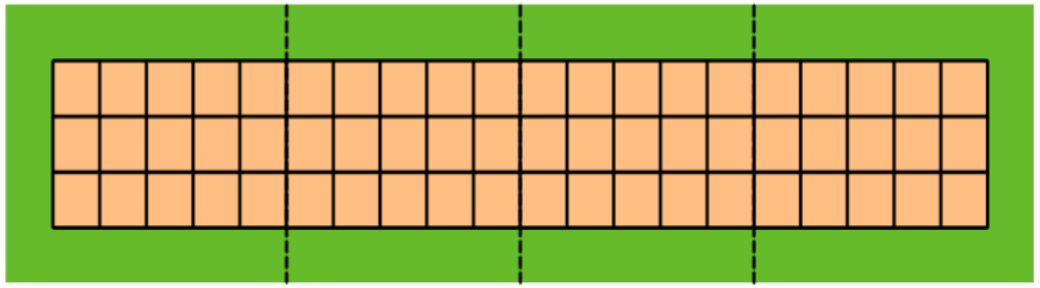
In RAM und auf Platte

data in memory



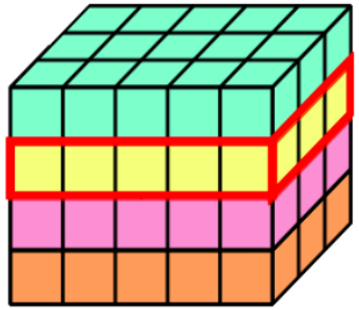
Contiguous layout

data on disk

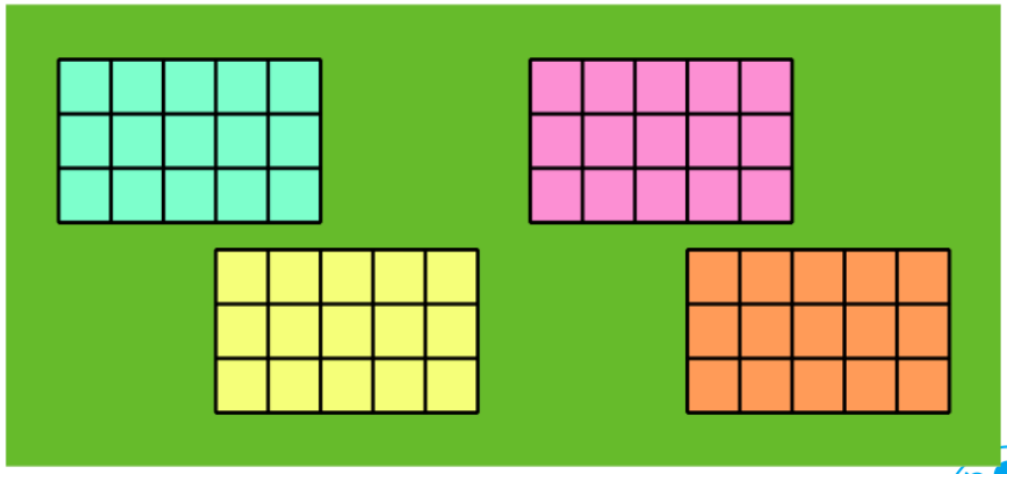


Chunked layout

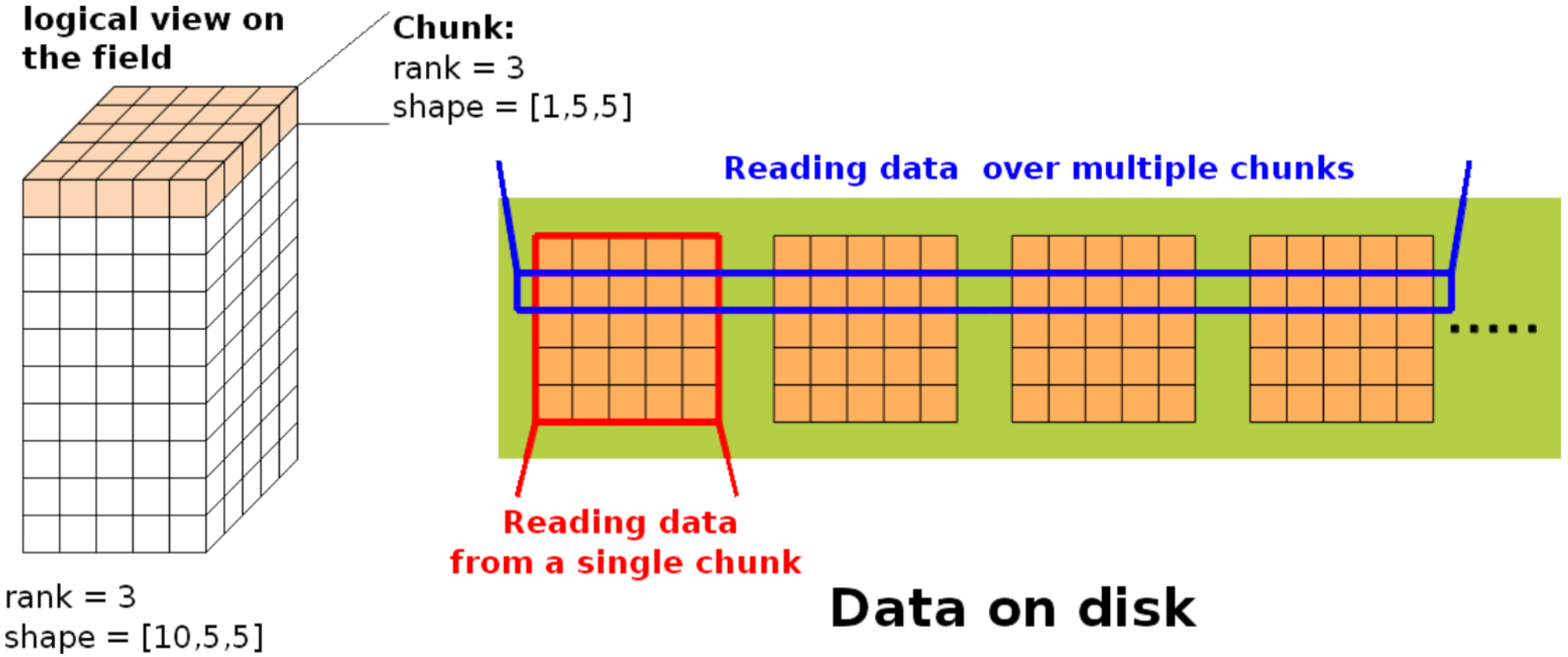
data in memory



data on disk



Wichtig für die Performance



Access to elements within a single chunk is fast!

Filters are applied to individual chunks.

Sprachen, Software, weiter Möglichkeiten

- C (native language for the HDF 5 library)
- C++
- Fortran (90, 95, 2003)
- Java
- Python (h5py, pytables)
- .NET
- Perl
- R
- …

- **Software support**
- Matlab
- Mathematica
- IDL

- Und ansonsten noch weitere Tools um Daten zu analysieren, managen, ….

Parallel HDF5: Erweiterung für parallele IO

- Wichtig: Manche Operationen müssen gleichzeitig erfolgen
  - ZB Metadaten (Attribute) müssen gleichzeit upgedated werden
- Deshalb spezielle Erweiterung von HDF5 (auf Basis von MPI-IO) um Concurrency sicherzustellen
- Datentransfer selber kann collective oder unabhänig erfolgen
- Wichtig: Parallel-HDF5 und (seriell-)HDF5 Dateien sind kompatibel

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!

… One more thing: Prüfung

- Prüfung am 27.7. ab 14:00
- Mündliche Prüfung bei Herrn Prof. Gülzow, Beisitzer Y. Kemp
- Dauer ca 20 Minuten pro Prüfling
- Anmeldung und konkrete Termine werden rechtzeitig bekannt gegeben.
- Inhalt: Kurs und Übungen, Materialien siehe Folien

Uebungen

- MPI_IO
- Angelehnt an // https://www.tacc.utexas.edu/documents/13601/900558/MPI-IO-Final.pdf/eea9d7d3-4b81-471c-b244-41498070e35d
- Weiteres Material
  - http://www.training.prace-ri.eu/uploads/tx_pracetmo/pio1.pdf
  - https://www.hpc.ntnu.no/display/hpc/Writing+to+MPI+Files
  - http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture32.pdf
  - https://support.hdfgroup.org/HDF5/Tutor/pprog.html

MPI IO very basic example: write

```
#include<stdio.h>
#include "mpi.h"
int main(int argc, char **argv){
  int i, rank, size, offset, nints, N=16 ;
  MPI_File fhw;
  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  int buf = rank;
  offset = rank*sizeof(int);
  MPI_File_open(MPI_COMM_WORLD, "datafile",
MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fhw);
  printf("\nRank: %d, Offset: %d\n", rank, offset);
  MPI_File_write_at(fhw, offset, &buf, 1, MPI_INT, &status);
  MPI_File_close(&fhw);
  MPI_Finalize();
  return 0;
}
```

MPI IO very basic example: read

```
#include<stdio.h>
#include "mpi.h"
int main(int argc, char **argv){
  int i, rank, size, offset, nints, N=16 ;
  MPI_File fhw;
  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  int buf;
  offset = rank*sizeof(int);
  MPI_File_open(MPI_COMM_WORLD, "datafile", MPI_MODE_RDONLY,
MPI_INFO_NULL, &fhw);
  MPI_File_read_at(fhw, offset, &buf, 1, MPI_INT, &status);
  MPI_File_close(&fhw);
  printf("\nRank: %d, Offset: %d, Content: %d\n", rank, offset,buf);
  MPI_Finalize();
  return 0;
}
```

Compile and run (and check)

- mpicc-mpich-mp -o exe code.c
- mpirun –n <N> ./exe

- hexdump -e '8/4 "%10d "' -e '"\n"' datafile

MPI IO slightly more advanced: write

```
#include<stdio.h>
#include "mpi.h"
int main(int argc, char **argv){
  int i, rank, size, offset, nints, N=16 ;
  MPI_File fhw;
  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  int buf[N];
  for (i=0;i<N;i++) buf[i]=(N-i) + rank*N;
  offset = N*rank*sizeof(int);
  MPI_File_open(MPI_COMM_WORLD, "datafile",
MPI_MODE_CREATE|MPI_MODE_WRONLY, MPI_INFO_NULL, &fhw);
  printf("\nRank: %d, Offset: %d\n", rank, offset);
  MPI_File_write_at(fhw, offset, buf, N, MPI_INT, &status);
  MPI_File_close(&fhw);
  MPI_Finalize();
  return 0;
}
```

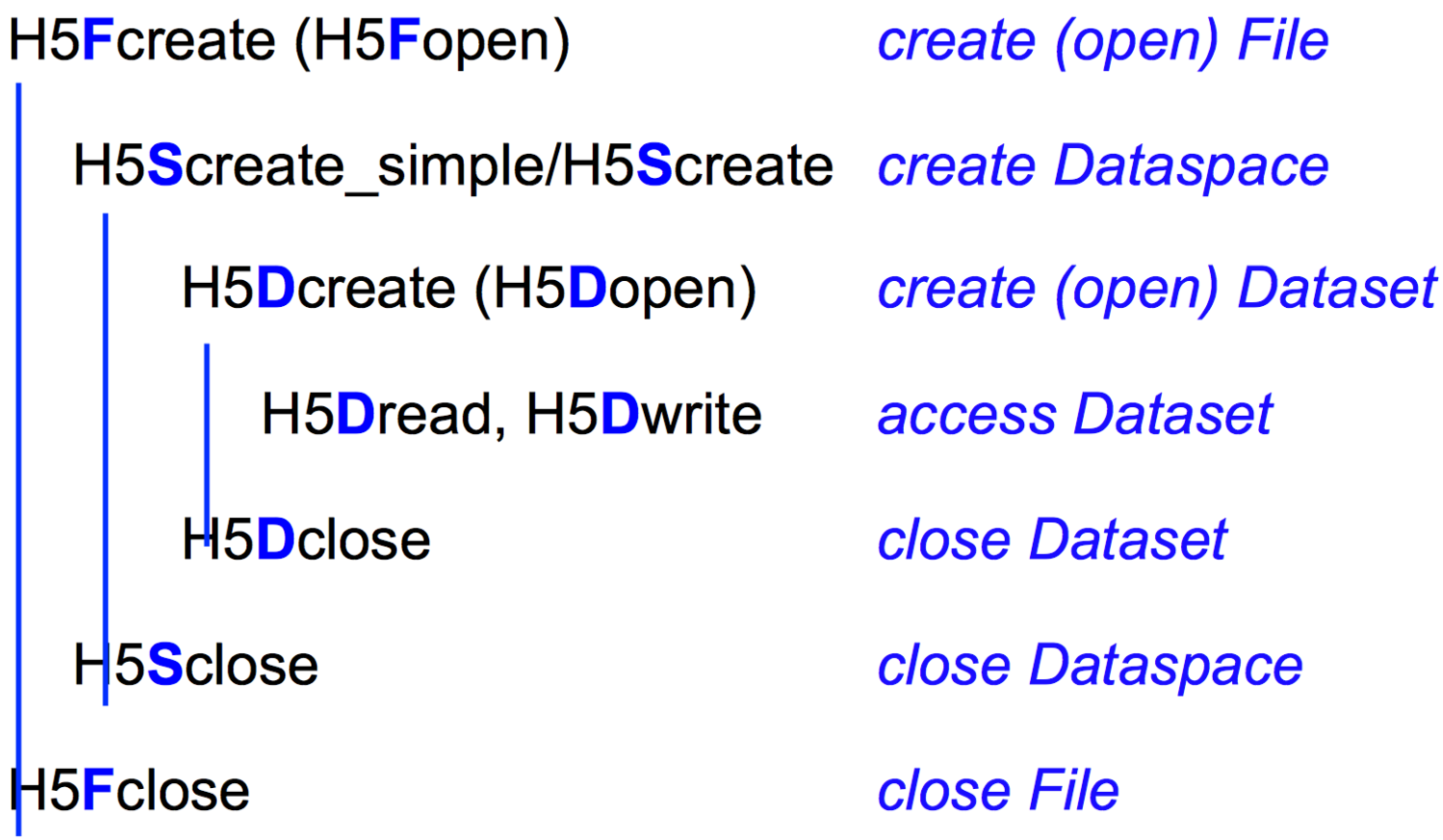
MPI IO slightly more advanced: read

```
#include<stdio.h>
#include "mpi.h"
int main(int argc, char **argv){
  int i, rank, size, offset, nints, N=16 ;
  MPI_File fhw;
  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  int buf[N];
  offset = N*rank*sizeof(int);
  MPI_File_open(MPI_COMM_WORLD, "datafile", MPI_MODE_RDONLY, MPI_INFO_NULL, &fhw);
  printf("\nRank: %d, Offset: %d\n", rank, offset);
  MPI_File_read_at(fhw, offset, buf, N, MPI_INT, &status);
  MPI_File_close(&fhw);
  printf("Rank: %d ",rank);
  for (i=0;i<N;i++)    printf("%d=%d ", i , buf[i]);
  printf("\n");
  MPI_Finalize();
  return 0;
}
```

HDF5 basic example

- ... Kein parallel HDF5 in den Beispielen

Basic HDF5 Operationen



Compile and dump file

- apt-get install hdf5-tools hdfview libhdf5-serial-dev

- gcc -o write write.c -I/opt/local/include/ -L/opt/local/lib -lhdf5
  - Oder wo auch immer die header und libs liegen
  - Vermutlich /usr/include und /usr/lib
- h5dump somefile.h5

HDF5: basic write

```
#include "hdf5.h"

int main() {
   hid_t       file_id, dataset_id, dataspace_id;  /* identifiers */
   hsize_t     dims[2];
   herr_t      status;

   /* Create a new file using default properties. */
   file_id = H5Fcreate(“test.h5”, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

   /* Create the data space for the dataset. */
   dims[0] = 4;
   dims[1] = 6;
   dataspace_id = H5Screate_simple(2, dims, NULL);

   /* Create the dataset. */
   dataset_id = H5Dcreate2(file_id, "/dset", H5T_STD_I32BE, dataspace_id,
                           H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);

   /* End access to the dataset and release resources used by it. */
   status = H5Dclose(dataset_id);

   /* Terminate access to the data space. */
   status = H5Sclose(dataspace_id);

   /* Close the file. */
   status = H5Fclose(file_id);
}
```

HDF5: basic read

```
#include "hdf5.h"

int main() {
   hid_t       file_id, dataset_id;  /* identifiers */
   herr_t      status;
   int         i, j, dset_data[4][6];

   /* Open an existing file. */
   file_id = H5Fopen(“test.h5”, H5F_ACC_RDWR, H5P_DEFAULT);

   /* Open an existing dataset. */
   dataset_id = H5Dopen2(file_id, "/dset", H5P_DEFAULT);

   status = H5Dread(dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
                    dset_data);

   for (i = 0; i < 4; i++) {
     printf("i:%d : ",i);
      for (j = 0; j < 6; j++)     printf("j:%d: %d ",j,dset_data[i][j]) ;
     printf("\n");
}

   /* Close the dataset. */
   status = H5Dclose(dataset_id);

   /* Close the file. */
   status = H5Fclose(file_id);
}
```