Stephan Wiesand
DESY - DV-
2005-06-21

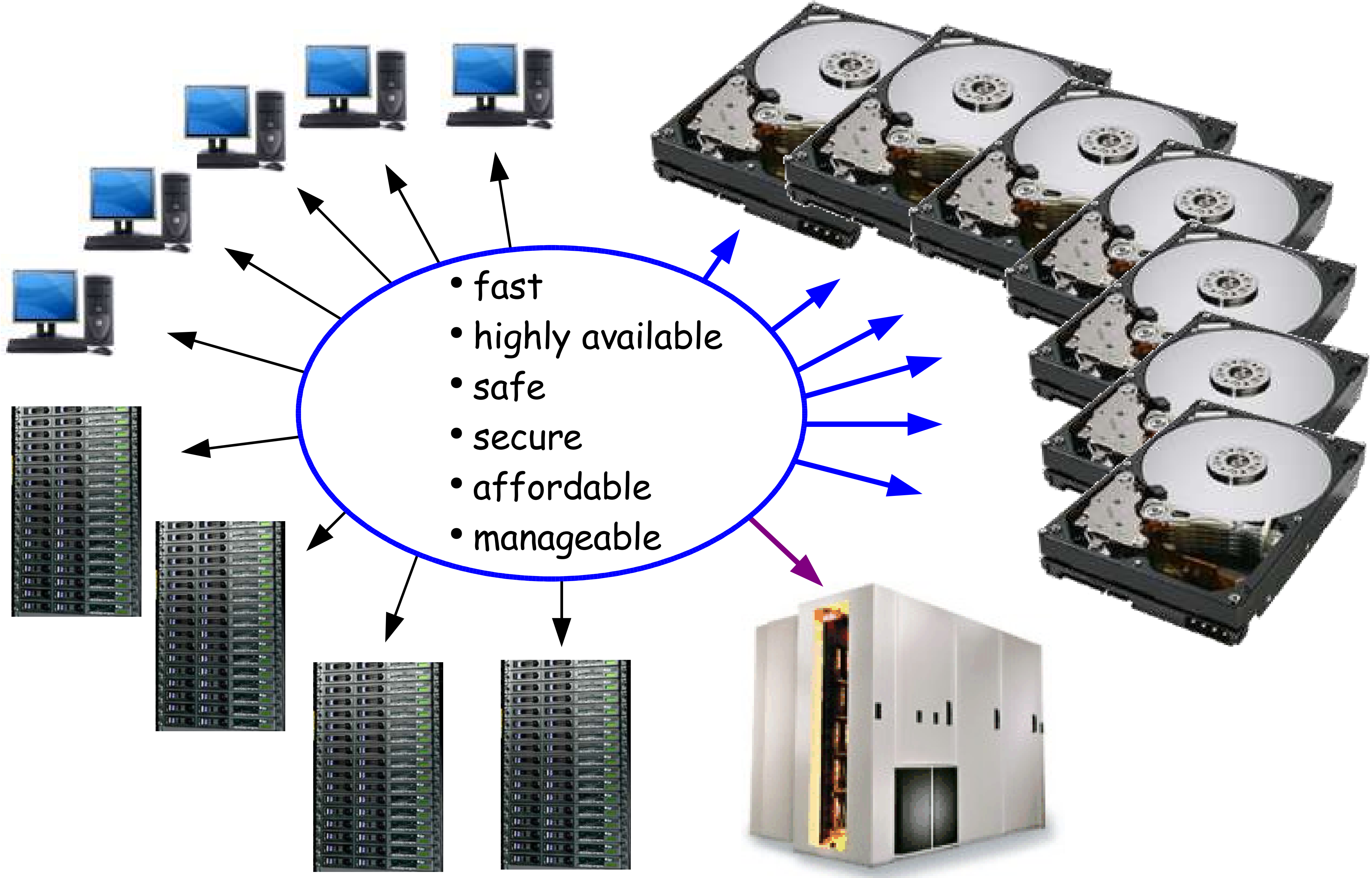
(Disk) Storage:

Hardware
Filesystems

and how to get most out of them

Standort Zeuthen

The Challenge



- fast
- highly available
- safe
- secure
- affordable
- manageable

Hard Disks: Facts

	SATA	SCSI/FC
Capacity	250/300/400 GB	300 GB
Heads	10	10
Cache	8 MB	8 MB
RPM	7200	10000
media transfer rate	750 Mb/s	1 Gb/s
interface transfer rate (max)	150 MB/s	320/150 MB/s
sustained data rate (max)	60 MB/s	90 MB/s
average seek time	8.5 ms	4.5 ms
non-recoverable error rate	1E-14	1E-15
mean time to failure	1 M hours	1.4 M hours
MTTF qualified by	"at low I/O duty cycle"	-
relative price per GB	1	≥3

- table compiled from specifications of typical current drives
- most SATA drives have no MTTF specified at all
  - instead: start-stop cycles (-> intended for desktop use)

More about Hard Disks

- Experience:
  - ATA disks (& vendor test programs) tend to be more than optimistic about the drive's health
    - “full test ok” -> reinsert -> fails within hours
- Fact:
  - SCSI drives tell the controller when they remapped a block
    - event is logged
    - early sign of drive failure
- Rumour:
  - SATA protocol does not allow the disk to signal
    - remapped a block
    - recalibrating, wait a few ms
- Rumour:
  - Models exist in two flavours
    - one is high quality & available from large integrators only
      - extra tests & selection
      - distinguished production lines
    - the other is what's available from retail shops

RAID Levels

- RAID0: Striping
  - every other stripe (typical size: 64 kB) resides on one of n disks
  - fast & cheap, but unreliable
    - a single broken disk kills the whole array
- RAID1: Mirroring
  - all data resides on both of 2 disks
  - reasonably fast, very fault tolerant, but only 50% net capacity
- RAID3/4:
  - n data disks + 1 parity disk => net capacity n/(n+1)
  - reasonably fast & fault tolerant (data survives if 1 disk breaks)
- RAID5: like 3/4, but rotates parity
  - avoids “hot” parity disk (bottleneck & wear)
- RAID6: 2 distributed parities (slower, but 2 disks may die)

RAID5

- writing on RAID5 is expensive:

    1) read back old data

    2)read back old parity data

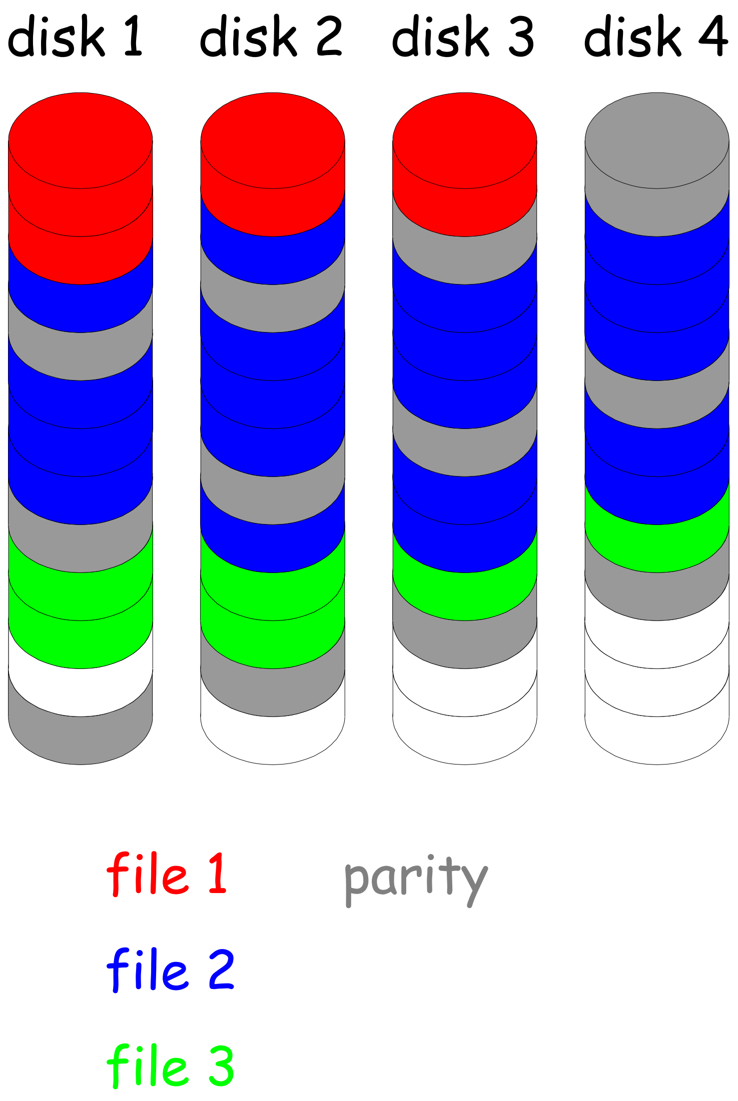
    3)calculate new parity data

    4)write new data

    5)write new parity data

- most efficient if writes are multiples of the stripe size

- => avoid small write requests on RAID5



How Safe is RAID5 ?

- (S)ATA drives running at not-so-low-I/O duty cycle
  - MTTF is much rather 1E5 hours than 1E6
- => each array typically looses a drive every few months
- a rebuild takes several hours
- => probability for a second drive to fail during rebuild: O(1 ‰)
- => it's going to happen, sooner or later !
- RAID5 is NOT a replacement for backup
- not even RAID1 is
- redundant RAID levels boost availability
  - they don't make your data safe

Backups are not for cowards only

- other potential sources of data loss:
  - accidental deletion (by user or admin)
  - failure of OS, software, firmware, hardware, wetware
  - crime (theft) & security breach (hacking)
- other things that don't replace backup:
  - automatic rsync to 2nd location
  - storing everything on tape
    - they do fail, even though very rarely
    - note OSM allows automatic cloning by subdirectory
- we can't backup all data
- => users ***MUST* distinguish what to store where**

The full cost of backup

- hardware:
  - robot, drives, server
    - maintenance contracts
  - tapes (cloned)
  - possibly disk cache
  - bandwidth
    - network, tape drives - and on the disk storage device
- backup software: licenses (clients, server) & maintenance
- labour
  - tape handling (removing clones,...)
  - managing, monitoring & troubleshooting the service

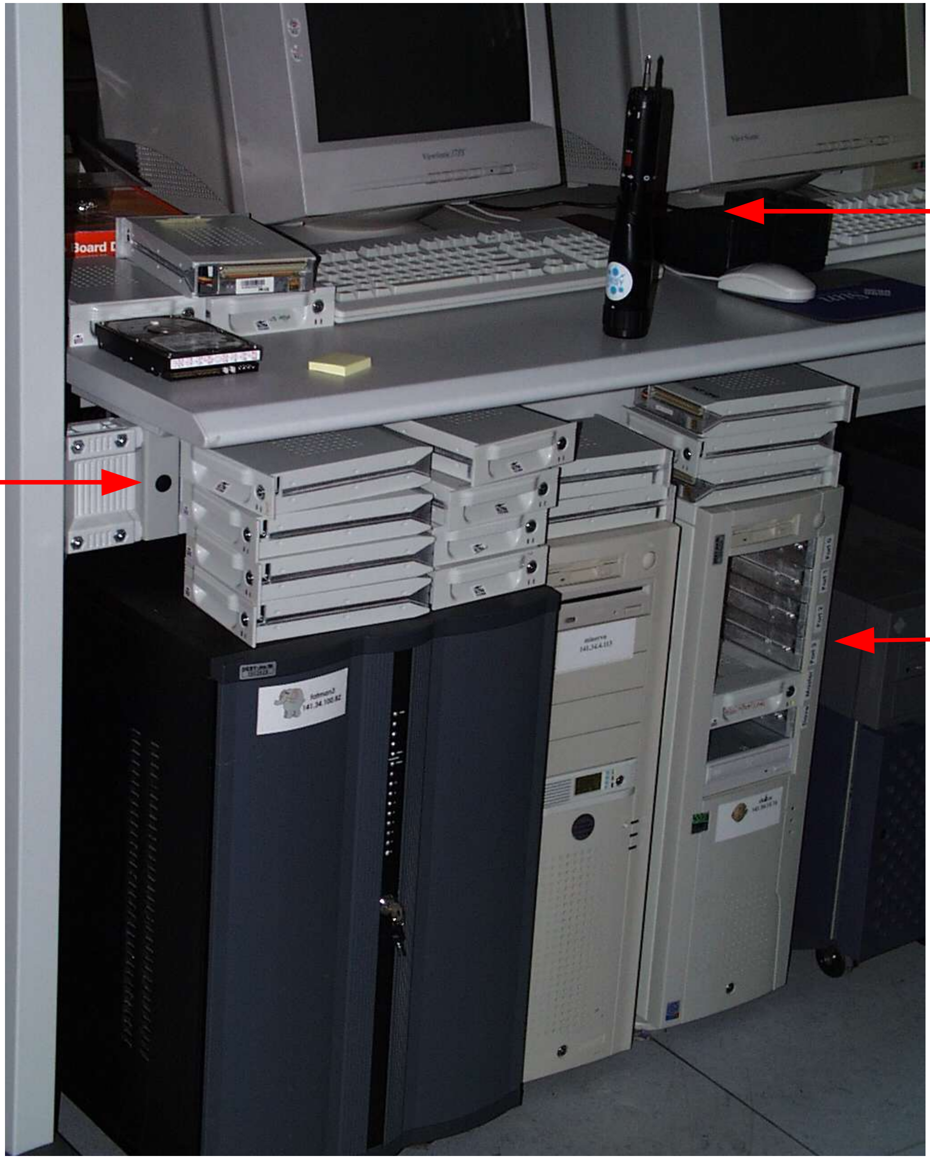
Classes of storage

- local disk (desktops, farm nodes, pubs, ...)
  - cheap, but completely unreliable, no backup - just scratch
  - right place for building software, keeping working copies of data
  - absolutely the wrong place for code (incl. TeX), results, ...
  - desktop/notebook drives die early when abused as fileservers
- AFS & NFS on cheap redundant fileservers
  - w/o backup: still scratch, but better (& wider) availability
  - w/ backup: general purpose, but limited amounts only
- Tape - reasonably safe especially if cloned
- Home Directory - **best hardware with daily backup**
  - this is the right place for code, results, ...

DELFI

- first generation cheap linux fileservers (2001)
- 20 x 75GB ATA (data)
- 2 x 30 GB ATA (system)
- 3 x 8-port RAID controller
  - 3ware 6800
- 2 x PIII 850 MHz, 512 MB
  - soon upgraded to 1 GB
- Gigabit Ethernet (fibre)
- benchmarks looked promising

DELFI + Care Pack



dedicated, battery driven screwdriver

spare drives
(for about two weeks)

dedicated system for disk tests, firmware upgrades, and burn-in

DELFI: Drive Problems

- 1st set of hard drives (IBM 75 GB) turned out to be unusable
  - died extremely fast, often in ways that crashed the controller
- 2nd set of hard drives (WD 80 GB) had their own problems
  - controller would often remove them due to timeouts
  - manufacturer's fitness test declared most of them good
    - some worked for months afterwards, some did not
  - drive firmware update from manufacturer did not resolve this
- 3rd try (Maxtor 120 GB) later worked well - alas, too late
- drive tests, firmware updates, reusing drives, bookkeeping
- major time sink

DELFI: Controller Problems

- several rounds of firmware/driver/daemon updates before stable operation was possible, this took many months
- last firmware update ever for our hardware was luckily the one that solved most problems
- firmware/driver/daemon versions must match
  - but no more fw updates soon after deployment
  - driver has to match kernel
  - daemon must match OS (and the old one was very unstable)
  - => experiments to get non-matching versions to work together
    - was possible until recently; but latest kernels don't work anymore
- major time sink

DELFI: Other Problems

- benchmarks had been run with RAID0
  - impossible to use the servers like this
- started deploying them with RAID5 (arrays of four disks)
  - worked, but was really slow (and net capacity down to 75%)
- replacing failed drives is an expert task
  - no “red LED” on failed disk, controller messages not always reliable
  - lost one array because someone pulled the wrong drive
- redeployed more critical servers with RAID1
  - acceptable speed, minimal risk of data loss
  - but net capacity down to 50%

DELFI: Decommissioning 2005

- many components not hotswappable and not monitored
  - fans
    - about every 2nd one in the power supplies not running anymore
    - drive cage fan status unknown - not accessible w/o removing cage
- repair work inside the system likely to break other things
  - delicate ATA cabling
- software problems (driver or daemon)
- finally running out of spare drives
  - even though many have been reused after 1st failure
- R.I.P.

DELFI: Lessons learned

- buy complete storage systems (drives & enclosure), not components thrown together
- buy from a vendor specialized in designing or composing these
- avoid daemons and special drivers if possible
- regular media scans are mandatory for reliable operation
  - find & remap bad blocks on unused disk regions - weekly
- reliable alerts are important
- locating and replacing failed drives must be easy and failsafe
- having failed drives replaced by the vendor must be easy (and fast!) as well
- only redundant RAID levels are acceptable, even for “scratch”

Cheap fileservers, next try:

- external SATA<->SCSI RAID
  - 16 disks (250 or 400 GB)
  - multiple arrays possible:
    - 1 x 7 + 1 x 8 disks RAID5
    - 1 x global hot spare
  - attached to one or two servers
  - looks to server like a SCSI disk
    - no extra drivers etc.
  - controller sends alerts by mail
  - red LED on failed drives
  - very fast, even with RAID5
    - 1 GB ECC cache w/ battery backup

A few months later

controllers replaced on suspicion (last option)

screwdriver now used for SATA as well

SATA <-> SCSI RAID: Problems

- devices tend to work well for several months
- then they start crashing or becoming incredibly slow
  - every two weeks, then once a week, ...
- NO indication from controller what's wrong, for months
- sometimes, eventually a disk dies and life goes on
- sometimes, need to disassemble the arrays
  - a simple dd then reveals one disk is extremely slow
- only power cycles help
- firmware odyssey

Recent Developments

- top: drive set made serious problems with three controllers and two backplanes
  - not in production since months
  - now latest controller model & firmware
    - latest fw officially still not available for our "legacy device" purchased last August
    - new hardware borrowed from vendor
  - alas, controller crashed sunday evening
    - => now 4 controllers and three backplanes
- bottom: device frozen two weeks ago
  - since then running unofficial beta firmware
  - so far 3 disk timeout alerts, one with removal of disk from array and rebuild

SATA <-> SCSI RAID: Conclusion

- would be a nice solution - if it worked
  - no (major) loss of data due to the crashes yet
  - but frequent service interruptions completely unacceptable
  - every crash takes hours of work to recover from
- => devices cannot be used in the intended way
  - how to proceed with these ? - unknown yet
- lessons learned:
  - try to get a complete solution from a single source
    - not: manufacturer (TW) -> technical account manager (UK) -> integrator (DE) -> reseller (DE)
  - try to find a solution allowing fallback to native drive access

Relatively cheap fileservers: 3rd try

- DELFI-like, again
- Dell 2850 servers
- 6 x 300 GB disks, SCSI !
- internal RAID controller
- alerts from daemon AND remote management card
- easy location of failed drives
- complete system from single vendor, certified for our OS
  - no more fingerpointing
- first 4 systems (being) deployed, 4 more are ordered

Hardware: Conclusions

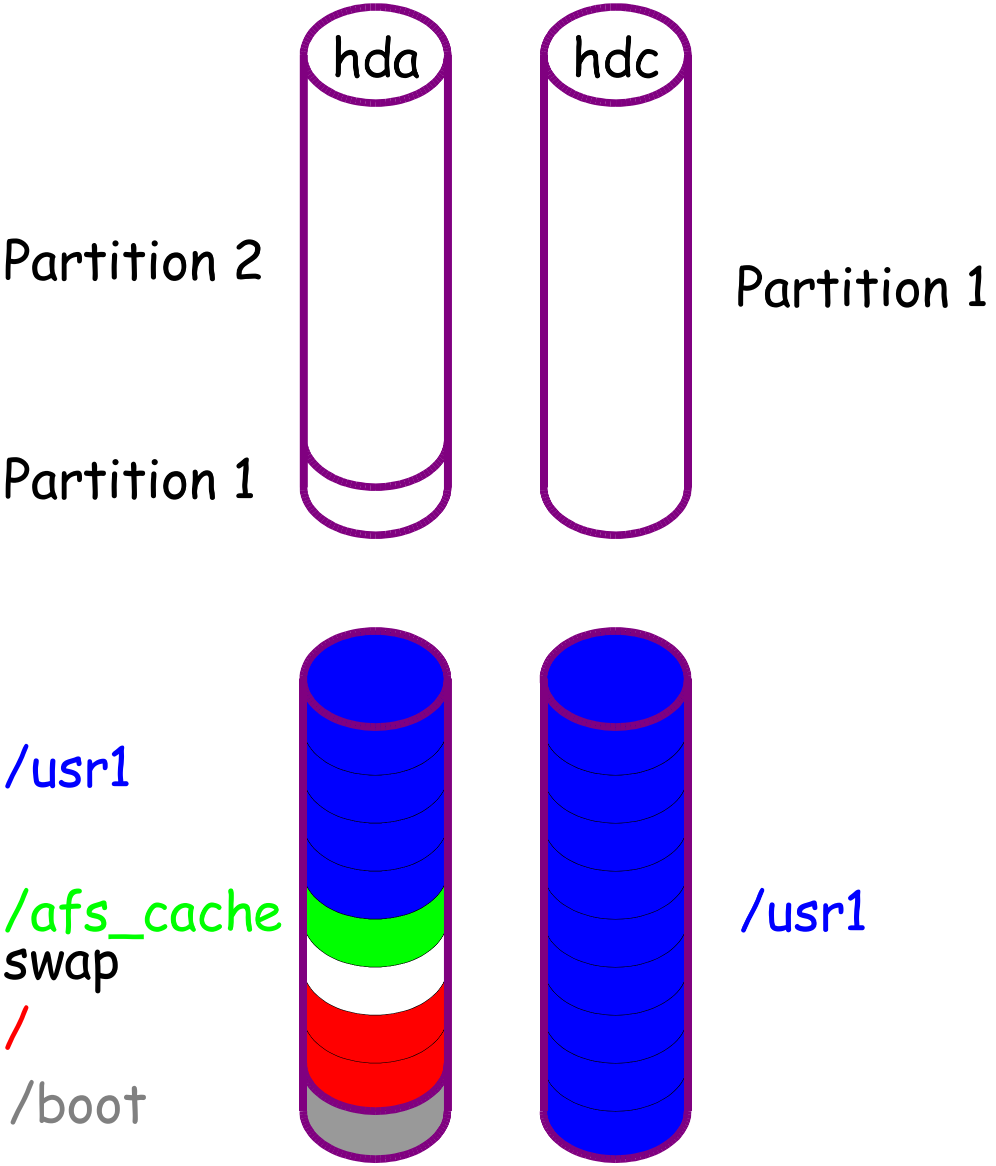
- (too) cheap hardware consumes inordinate amounts of labour
- NB: even expensive storage hardware often does
  - FC <-> FC redundant controller arrays purchased in Zeuthen and Hamburg for critical data (home directories, ...)
    - months and months, dozens of mails and phone calls, several firmware updates, finally replacement of hardware components before stable operation in Zeuthen
    - devices in Hamburg still not fully operational
- there is no such thing as cheap storage
- there is no such thing as cheap storage
- there is no such thing as cheap storage
- don't waste it, and use it efficiently !

Theory: Common Size Limits

- 2GB
  - file size limit on older filesystems
  - largest signed integer representable with 32 bits
- 1 TB
  - block device size limit on SL <= 3.0.4, Solaris <= 9 update x, ...
  - 40 bits ? - devices are addressed by block (512 bytes => +9 bits)
- 2 TB
  - block device size limit on many current operating systems
  - Solaris 10 lifts its, as does 64-bit linux 2.6
  - even so, 2TB limit often still applies
    - Solaris 10 SCSI driver

LVM: Logical Volume Management

- physical volumes (PVs)
  - devices (whole disk)
  - partitions (of disks)
- volume groups (VGs)
  - built from PVs
- logical volumes (LVs)
  - allocated from VG
- ice39 (2x20 GB ATA)
  - hda1, hdc1 form VG00
  - /, swap, /afs_cache, /usr1 on LVs in VG00

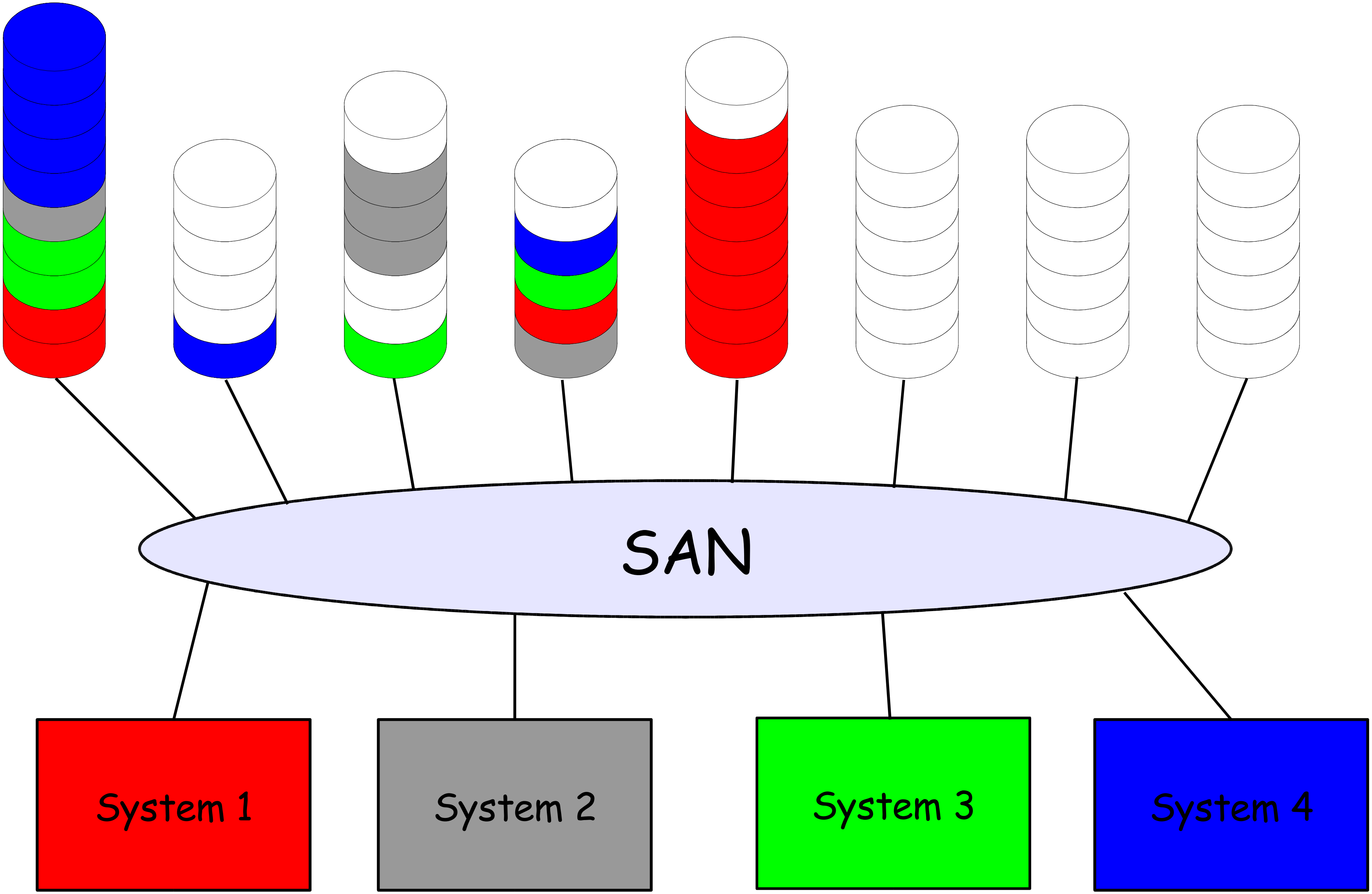


LVM possibilities (Linux; Solaris similar)

- create & remove volume groups - online
- add physical volumes to a volume group - online
- remove free physical volumes from a volume group - online
- move data between physical volumes - online
  - dangerous for striped logical volumes (only)
- create, remove, grow, shrink logical volumes - online
  - filesystems still need to be resized as well
  - allocation schemes: next free, contiguous, striped (= RAID0)
    - Linux 2.6 device mapper adds mirroring (= RAID1)
- snapshots
- => stop thinking in terms of of disks, start thinking of volumes

SAN: Storage Area Network

- storage is not attached to servers but to the SAN
  - typically through a storage controller (...)
- chunks of storage are mapped on one or more systems
  - where they appear as block devices
  - -> filesystems, logical volumes
- every computer may have access to any storage device
  - -> shared storage (high availability solutions)
  - NOT a shared filesystem, just shared storage
- the catch: cost
  - controllers, switches, cabling, host bus adapters, expertise
- Zeuthen operates (non-switched) “mini SAN” (mainly homedirs)


SAN
System 1
System 2
System 3
System 4

Local Filesystems: Solaris

- UFS
  - journaled (optional)
    - fast recovery after crashes w/o need for fsck
  - no resizing
  - not particularly fast (should have improved with Solaris 9)
  - default in Zeuthen (not journaled)
- commercial 3rd party products
  - Veritas VXFS
- ZFS (“Zetabyte FS”)
  - next generation filesystem that can do anything
  - was scheduled for Solaris 10, but not quite ready yet

Local Filesystems: Linux

- ext2/ext3
  - ext3 is journaled
    - different modes
  - resizable (up & down)
    - SL3: offline, SL4: online
    - growth limit for older fs
  - very reliable
  - very good fsck
  - many say it's slow
    - esp. ext3
  - the only filesystem supported on RHEL
  - ext3 is default in Zeuthen
    - metadata-only journaling for bulk data

- xfs (from SGI)
  - all features incl. online resize
    - journals metadata only
  - supposed to be very fast
  - not available with RH/SL
    - needs hacked kernel (CERN does this, but...)
- JFS (from IBM)
  - journaled filesystem w/ many features
  - no experience yet
- ReiserFS(4)
  - no experience yet
  - burned fingers with V3

Shared Filesystems: NFS (V3)

- 1995, enhancements over V2 (1989)
- initially designed to be a stateless protocol
  - impossible to get close to usual filesystem semantics w/o state
  - => additional mount protocol, lock manager
  - outage of server or client now is a significant event
  - => additional reboot notification service (statd)
- usage is still simple
  - server: `exportfs -o rw,async client.ifh.de:/data`
  - client: `mount server.ifh.de:/data /nfs/data`
- servers and clients aren't

NFS V3 in practice

- fast and fairly reliable
  - as long as ratio clients/server << 10
  - as long as ALL users of a server are knowledgeable & careful
    - any access to one directory with 40000 files renders server unusable for all clients (at least with ext2/3)
      - exporting reiserfs does not work perfectly, xfs should but no experience
- linux client still problematic
  - frequently have to reboot clients after server/network downtimes
- users sitting in mount points prevent automatic recovery all the time
- recently stability problems with linux server (needs restart)

NFS V3 in practice, continued

- no quotas on large fileservers (avoid slower start & operation)
  - also avoid the 5th daemon/service
  - limited # of independent filesystems per server
  - => how to do NFS on shared multi-TB fileservers ?
- some clients often starve when a server is under high load
- interoperability problems between vendor implementations
- linux TCP implementation not yet stable
  - last stress test was complete desaster
- UDP limits requests to 8 kB
  - makes RAID5 arrays even slower to write on
  - rfcp is recommended for NFS writes, NOT cp

rfio

- developed at CERN, predecessor of CASTOR
- TCP based client server application
- userland server -> write operations happen in larger chunks
    - faster on RAID5
  - reading makes little difference, but possible as well
- **`rfcp`** as installed in Zeuthen is a wrapper script
  - allows using rfcp like cp on NFS paths
  - falls back to cp if destination not reachable by rfio
    - AFS, unknown/unconfigured server
- rfio installation in Zeuthen is currently unmaintained but still works

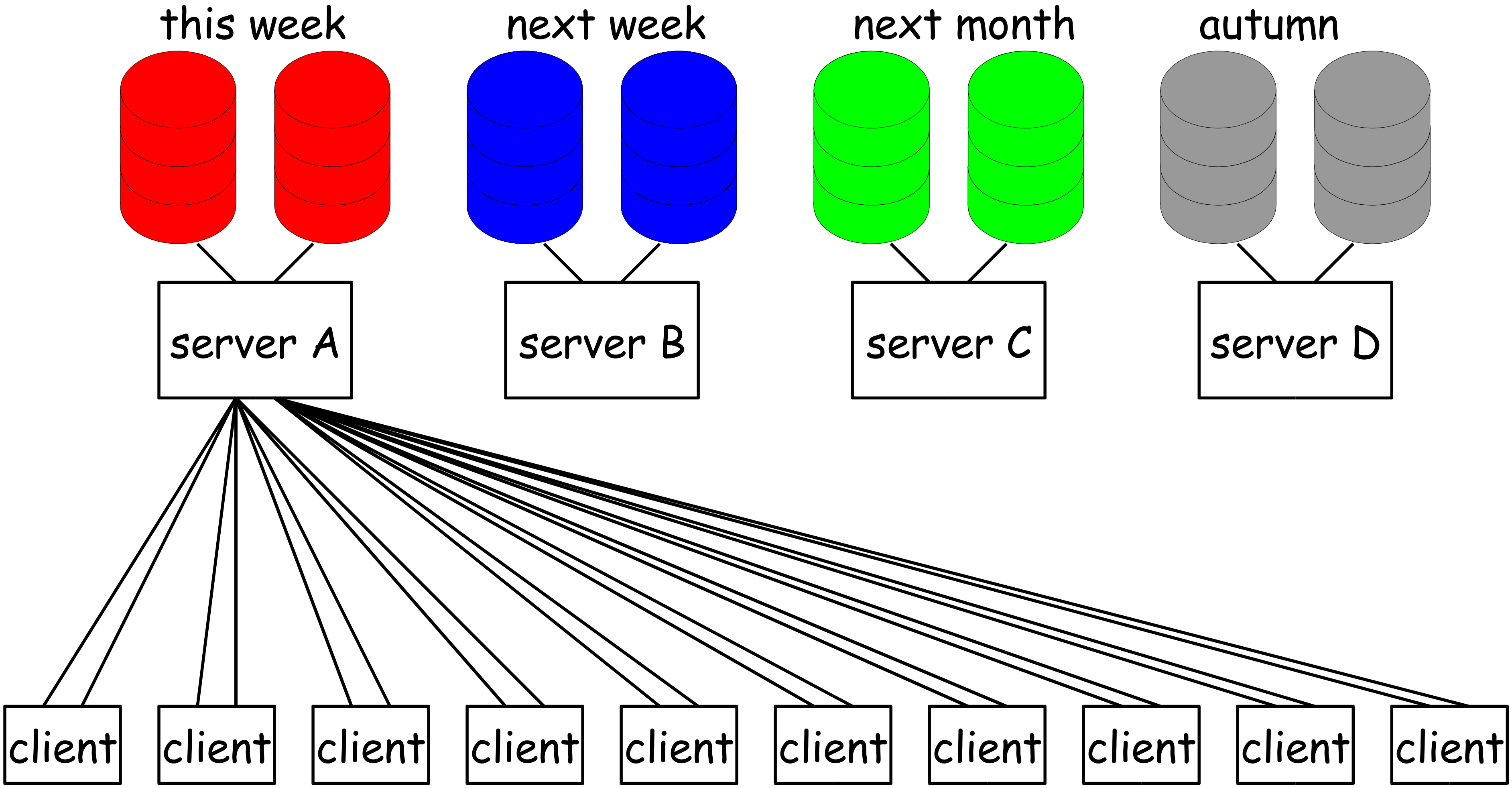
NFS V3 limitations

- security: only host based authentication
  - export into untrusted networks/subnets simply hazardous
    - actually includes desktops, notebooks, HH, other sites
  - to access someone else's data in NFS:
    - find a host the filesystem is exported to
    - hack it, or replace it by your notebook, or steal it's IP address
    - assume any ID you like and read/delete/modify/add data as you please
    - [don't really do or even try this - it's too easy but still illegal]
- data is tied to the server it resides on
  - w/o SAN + LVM it's even tied to the storage attached to the server
- location and mount points need to be maintained on any client

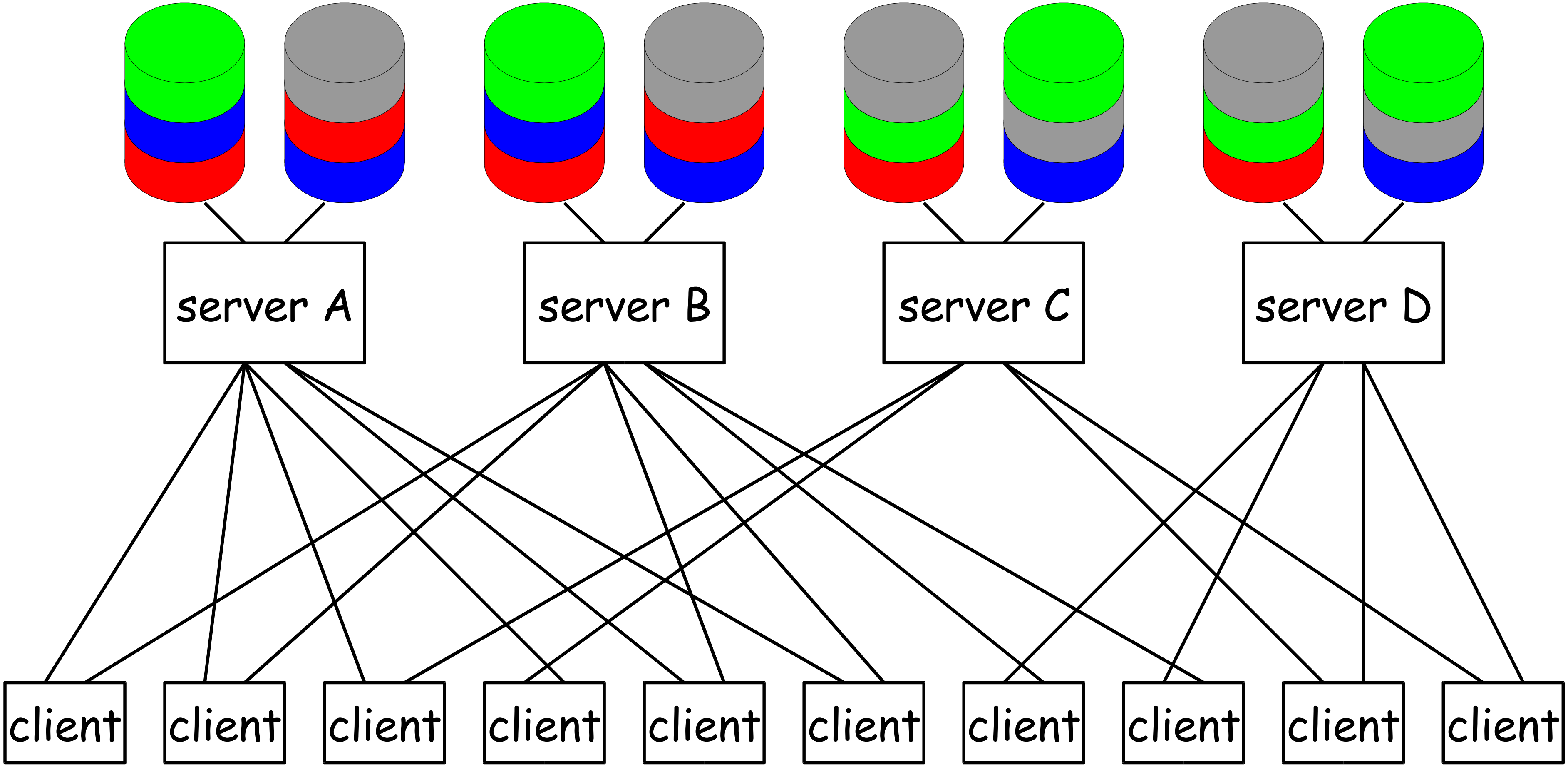
NFS V3 limitations, continued

- result: moving data to different storage and/or servers
  - is extremely tedious & impossible without interrupting the service
    - "Dear user xyz, we urgently have to replace a bunch of old breaking hard drives. Your data now available under /net/x/y/z will be available read-only starting 2005-x-y h1:m1 to hh:mm when we copy it to the new location /net/a/b/c, where it will be available read/write after the completion of the copy process which is scheduled for 2005-x-y h2:m2. ..."
- for the same reasons: load balancing not feasible
- no replication of data => load sharing impossible
  - except if users distribute their data across locations, according to anticipated usage
  - in practice, they never do this (actually tend to do the opposite)

typical access scheme



desired access scheme



NFS V4: many improvements over V3

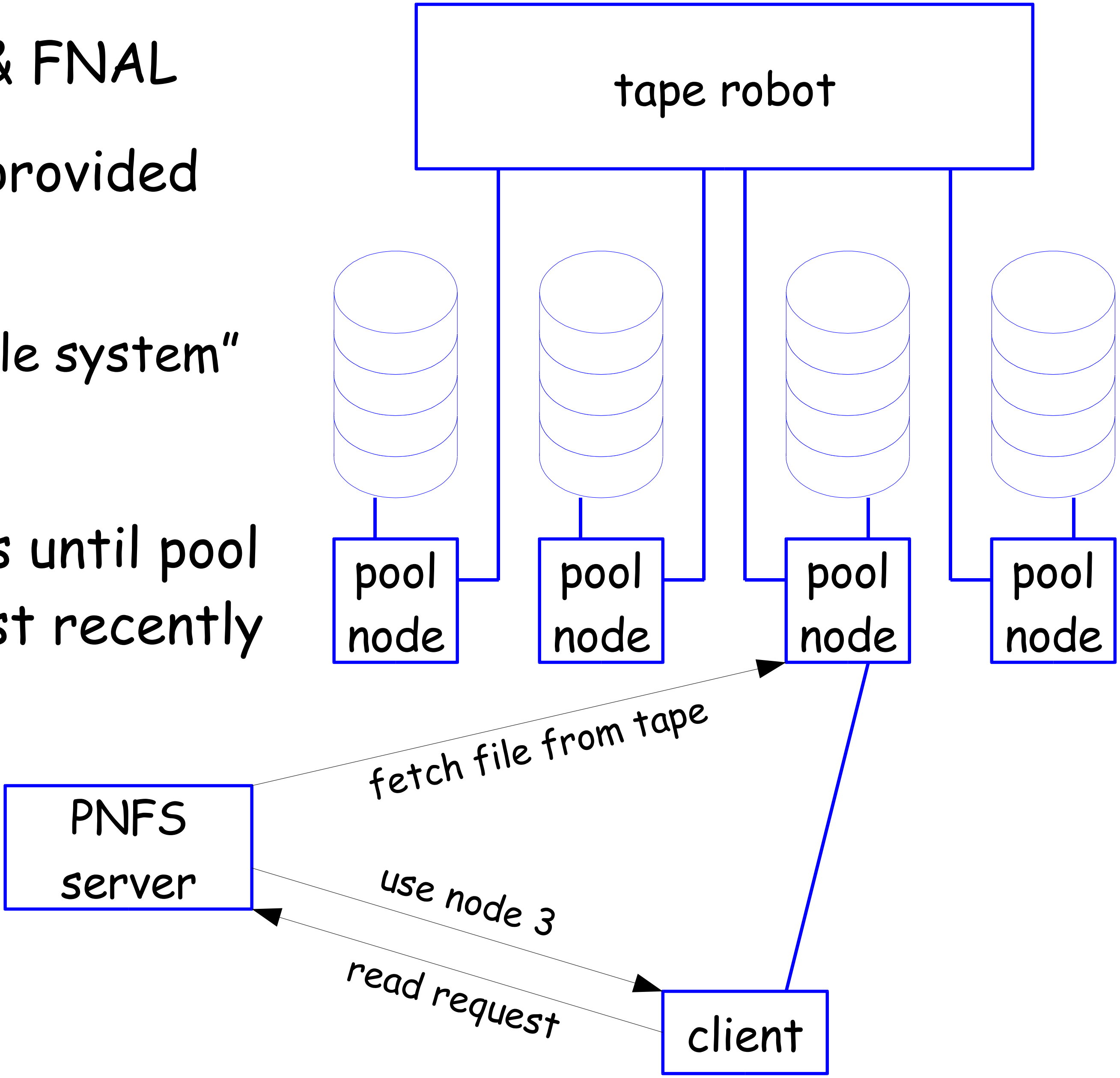
- Kerberos V authentication => reasonably secure
- single protocol over TCP => WAN usage feasible
- client side caching with well defined semantics & validation
    - V3 clients cache "for about 1 second" without validation
- richer protocol for improved performance and concurrency
- replication of readonly data with automatic client failover
- migration of data between servers
- alas:
    - only the first two items are actually implemented yet
    - only available with latest OS releases (SL4, Solaris 10)

NFS: Summary

- current V3 has severe shortcomings in
  - security
  - scalability
  - availability
  - manageability
- V4 will improve on this significantly, once fully available
  - today, it solves the security problem only
    - at the cost of simplicity and ease of use
  - next year, it may solve more problems
  - for the time being, developers seem to be fighting several of the same problems V3 has in practice (reboot recovery...)

dCache

- joint project: DESY & FNAL
- namespace (/acs/...) provided by PNFS server
  - "perfectly normal file system"
  - actually, NFS V2
- pool nodes cache files until pool full, then discard least recently used file
- client does not care where data is cached



using dCache

- client access by
  - **dccp** command (copy whole files)
  - dcap library (dc_open, dc_close, dc_seek, dc_read, ...)
    - /opt/products/dcache/default/{include|lib}
    - ROOT interface exists
    - tunable via environment variables
      - readahead, deferred writes, ...
  - preload library
    - replaces normal library calls (open, ...) with dcache versions
    - **LD_PRELOAD=.../libpdcap.so my_app**
      - my_app may now call open() etc. on cached files from tape
    - also tunable via environment
    - does not work for all applications

dCache features

- load balancing and load sharing
  - more than one pool node may have a copy of a file
  - new copies via pool->pool or tape->pool if pools overloaded
- separate read/write pools
  - cheap read pools, best quality write pools
  - possibility to transfer files from write to read pool between servers, no need to retrieve file from tape
- may even be used without tape backing at all
- resilient mode: keeps a desired number of copies on different nodes (which may be very cheap or unreliable then...)
- can be configured to be secure (Kerberos 5)

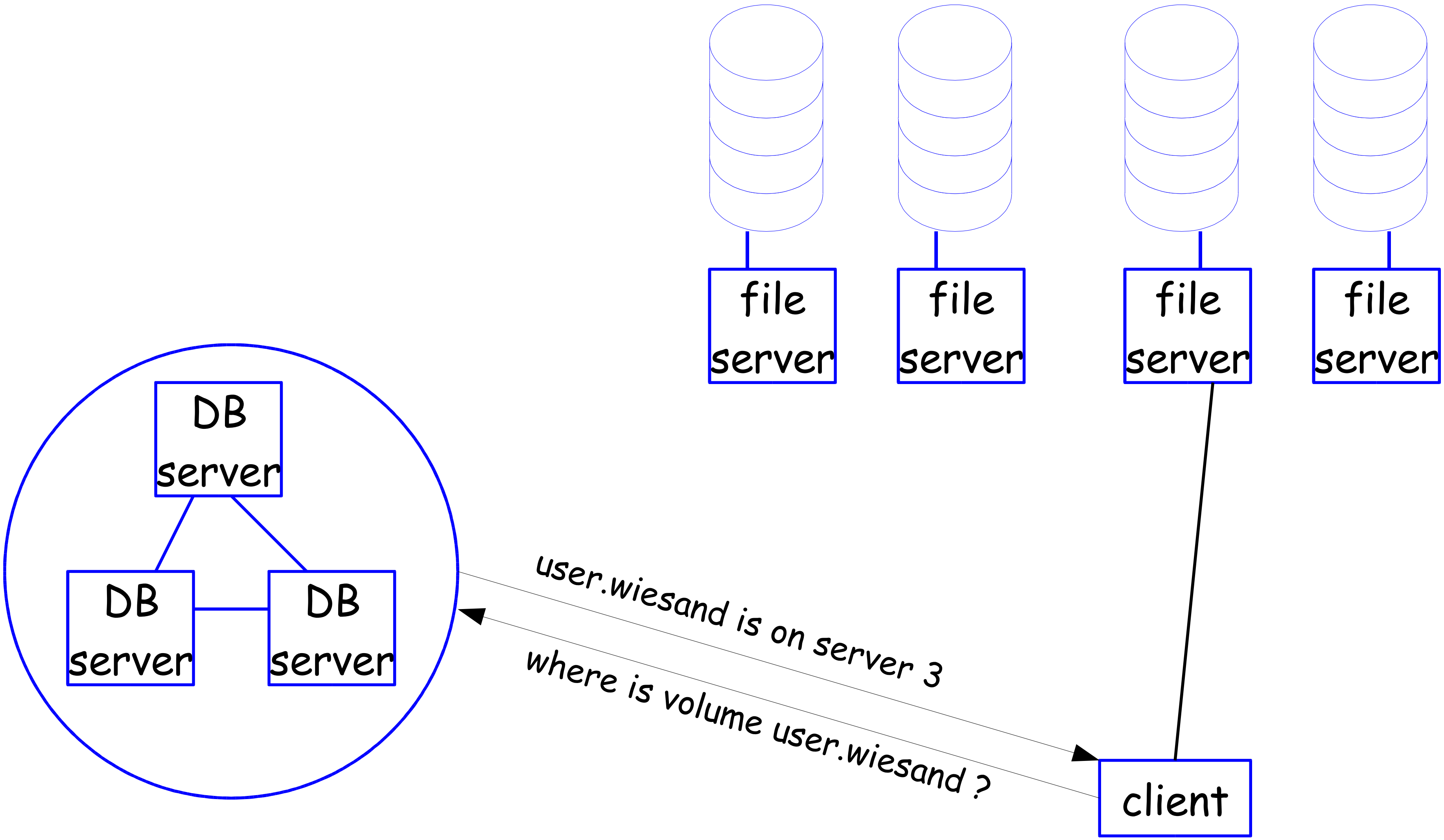
dCache limitations & drawbacks

- PNFS server is single point of failure
- files can not be modified, only deleted and rewritten
- not an “ordinary” filesystem
- files are not available for reading before closed
- proper design & configuration is NOT simple...
- not open source
- limit for concurrent clients applies per pool, not per node
  - problem if multiple pools on one node
- stores files on pool nodes in a single flat directory per pool
  - problem if pools are large (many files/directory)

dCache in Zeuthen

- still limited setup
- write pools are sufficient, several smaller read pools have existed for a while
- recently deployed a first fast & large read pool node (1.5 TB)
  - still assessing optimal read cache volume
- no secure setup yet
  - like osmcp, only available on farms, wgs, and few exceptions
- dCache will be the backend for our GRID storage elements
  - that's secure, of course...
- other than that, future deployment will depend on user's demand/acceptance

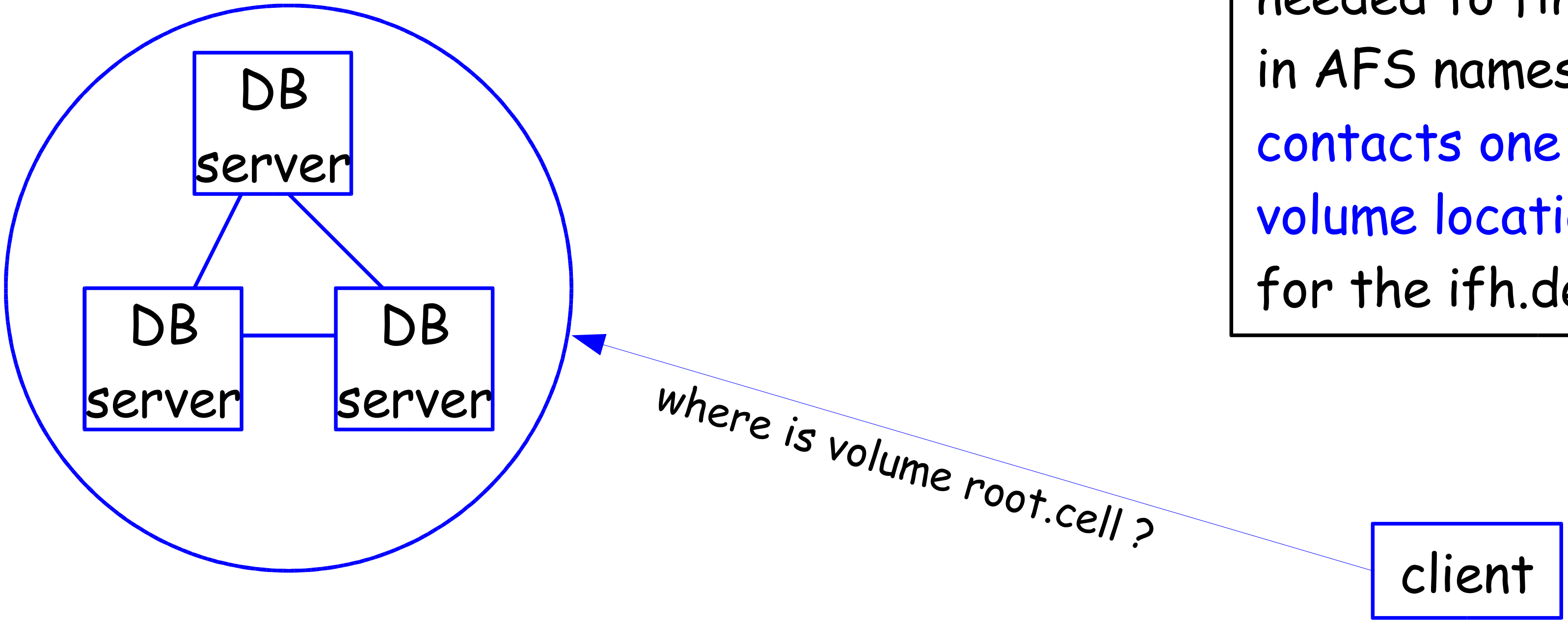
AFS


file server
file server
file server
file server
DB server
DB server
DB server
user.wiesand is on server 3
where is volume user.wiesand ?
client

Accessing a file in AFS - Step 1: /afs/ifh.de

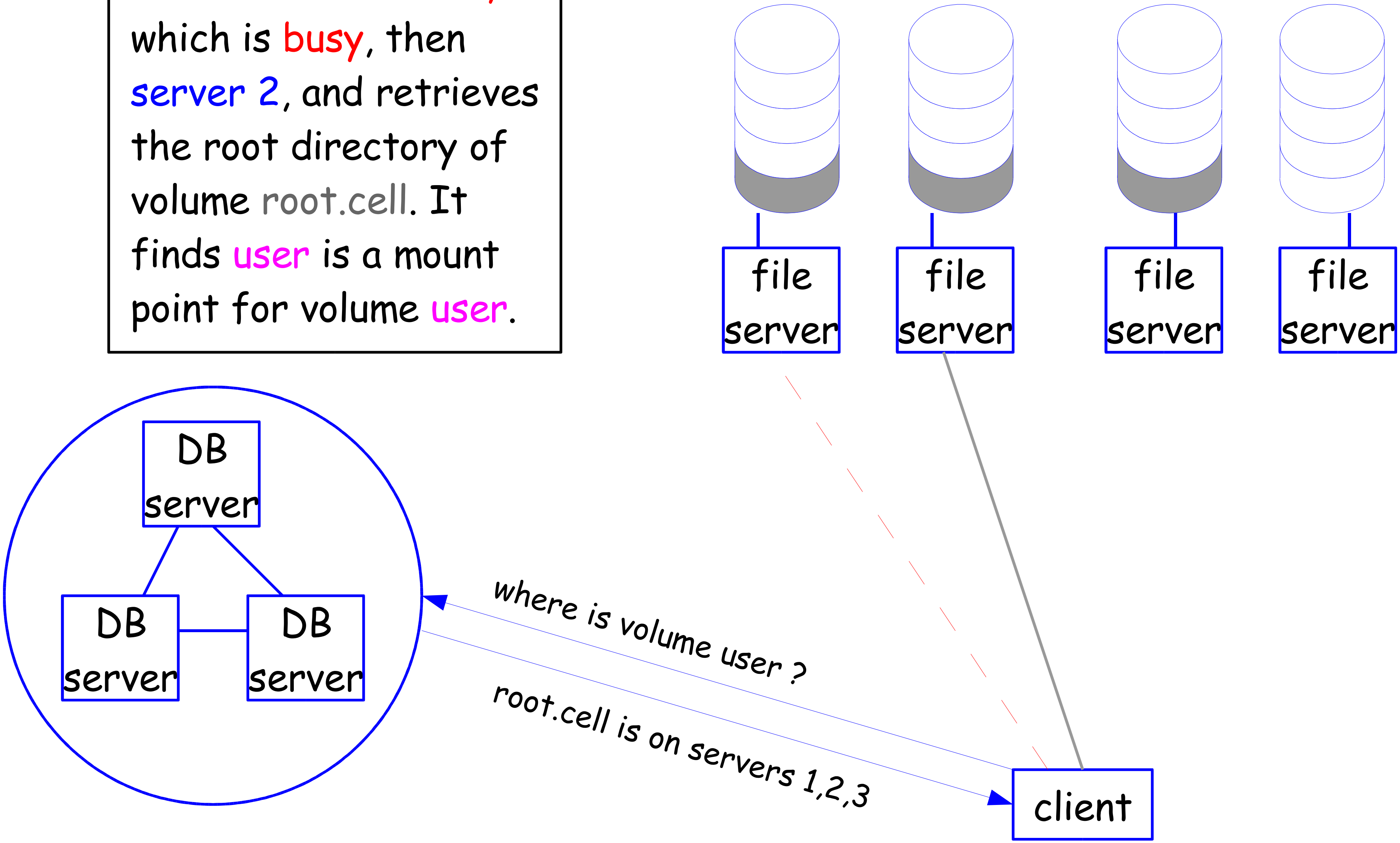
DB servers provide a highly available service with automatic client failover

Client consults its CellServDB file or DNS (all client configuration needed to find any file in AFS namespace) and contacts one of the volume location servers for the ifh.de cell



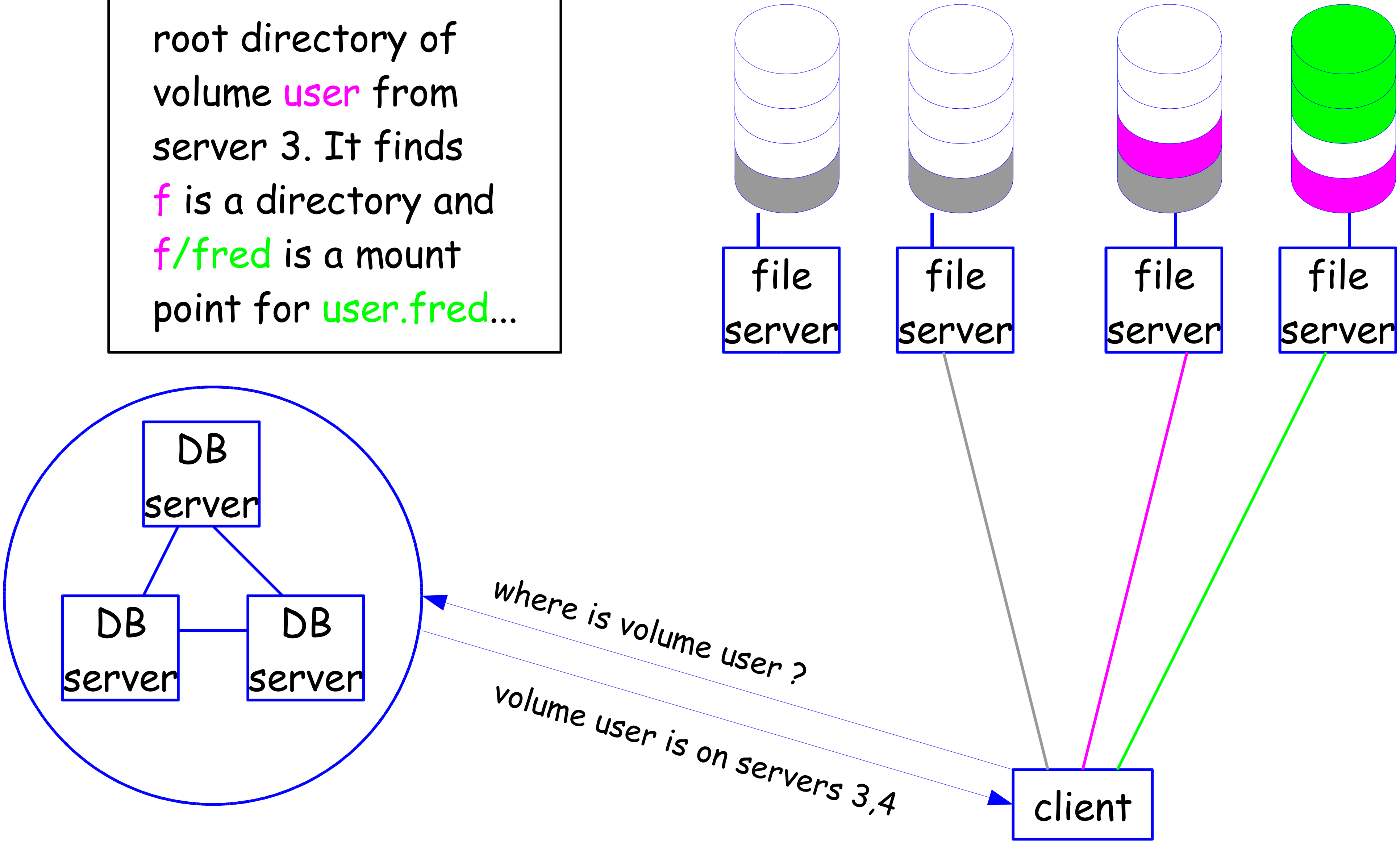
Step 2: /afs/ifh.de/user

Client tries server 1, which is busy, then server 2, and retrieves the root directory of volume root.cell. It finds user is a mount point for volume user.



Step 3: /afs/ifh.de/user/f/fred

Client retrieves the root directory of volume user from server 3. It finds f is a directory and f/fred is a mount point for user.fred...



AFS: Overview

- data is organized in volumes and handled by fileservers
- volume location is retrieved from volume location servers
  - 1,3,5,... independent systems communicating with each other
  - read-write replication of location data if >n/2 systems alive
    - read-only if <= n/2
- mount points for other volumes are embedded in the directory tree on the volumes themselves
  - actually a special kind of symlink
  - can be created by any user who can create files in a directory
  - the mounted volume then appears as a subdirectory
- this + DB server IP addresses builds a global /afs filesystem

AFS Volumes can be...

- created & deleted
- moved between fileservers, transparent for the client (!)
  - => load balancing
- read-only replicated to one or more fileservers
  - path to r/o: /afs/ifh.de/... path to r/w: /afs/.ifh.de/...
  - => load sharing & high availability for read-only data
  - automatic client failover
  - replicas can be removed as well if no longer needed
- mounted 0, 1, or more times anywhere in the /afs/<cell> tree
- => build your directory tree the way you like
  - without shuffling data around

More AFS features

- backup volumes (= copy-on-write snapshots)
  - ~/.OldFiles : snapshot of home directory taken last night
    - immediately retrieve files accidentally deleted today
  - can be created for any volume (but may double space usage)
- volume quotas can be grown and shrunk anytime
  - fileserver partitions can be overbooked
    - => efficient usage of space
    - when partition fills, move some volumes to new servers
- access control lists (ACLs), interoperable across OSs
- user defined groups (for use with ACLs)
- file servers provides access statistics (for load balancing)

AFS security

- kerberos authentication - AFS token = kerberos ticket
- all AFS space can be accessed from desktops, notebooks, home PCs, in Hamburg, at CERN, other labs...
  - servers are not accessible from anywhere, but firewall exceptions are added if reasonable
- to access someone else's data in AFS:
  - find a system the user is currently logged on to
  - hack the user's account or root on this system
  - you can now use the user's AFS token - until it expires
    - after 25 hours, you have to succeed in hacking again
  - [same disclaimer as for NFS: don't do it, don't try it!]

AFS drawbacks

- 2 GB filesize limit in current stable OpenAFS release
- ACLs are per directory, not per file
- host based authentication is impossible
  - need special solutions for cron, batch, ...
- no read-write replication (are there production-ready filesystems that do this?
- volumes & files can't be striped across fileservers
  - no load sharing on sub-volume level
  - => organize your data in volumes of reasonable size
    - years / months / run ranges, ...
- server performance is ok, but client performance is terrible

AFS performance

- before you complain about performance:
  - check how busy the server is (rxdebug command), you are sharing it with a few hundred other users
  - make sure it's not yourself who's overloading the server
    - NO single fileserver can provide file service to 200 jobs on 100 fast farm nodes at reasonable speed, especially if they do little else than I/O !
    - organize your data in volumes of reasonable size, spread across servers
      - this does NOT mean you have to access it under N different paths
    - consider read-only replicas for very popular volumes
    - **stop moving data around on remote filesystems**
      - there is NO need to mv gigabytes of data from one AFS location to another,
      - instead, mount volumes under different paths
      - if absolutely necessary, move volumes (server to server transfer)

AFS client performance

- persistent client side cache with well defined semantics
  - often helps a lot, but sometimes is a major burden
    - large files, especially when mv'ed or cp'ed from AFS to AFS
  - there is some room left for tuning, time permitting
  - could use memory cache instead of disk cache
- but it's possible to bypass the cache:
  - `Atrans` command (try the manpage) reads/writes without using the AFS client
    - `afscp` wrapper script by Dirk Pleiter is more convenient/failsafe
  - single client still only gets 70% wire speed, but is no longer busy
  - many concurrent clients works better than with NFS

Recommended use in batch jobs

- organize your data in volumes of reasonable size so that it can be spread across servers
- consider using several read-only replicas of data that will be accessed by many clients simultaneously
- in jobs:
  - transfer all input files to the local disk using Atrans
  - create output files on the local disk
  - at job end, transfer output files back using Atrans
    - if using NFS, use rfcp, not cp
  - retry transfers if they fail
  - don't use lockfile to limit concurrent access
    - check server load with rxdebug if you feel like it

Advanced (?) AFS commands

- which volume is this directory in, what's the quota ?
  - `fs listquota <path>`
- on which fileserver is this directory located ?
  - `fs whereis <path>`
- is this a mount point, and for which volume ?
  - `fs lsmount <directory>`
- create/remove a mount point
  - `fs mkmount/rmmount <directory> <volume>`
- where are all the r/w and r/o copies of this volume?
  - `vos examine <volume>`
- where is this volume mounted ?
  - `lsmount <volume>`

Volume Management

- group admins can create volumes using `afs_admin`
  - within global group/project quotas
- they can also create read-only replicas
- they can not (yet?) move volumes
  - actually, load balancing will start after hardware problems have been solved
    - no chance to balance something that's hundreds of GB large, though...
- volume naming convention: no backup if second letter is “n”
  - “gn.rz. ...”
  - “un.<user>. ...”

About data handling

- reading/writing large amounts of data consumes precious resources: bandwidth, memory, load
  - on the server, the network, and the client (even w/o AFS...)
- try not to waste those, they're expensive and not abundant
- try to spread the necessary amount as evenly as possible
  - if users don't do this, we'll have to spend MUCH more money on solutions that can be used without much thought
    - SAN + cluster filesystem at least 5 x more expensive than direct attached storage + AFS/dCache
- and: **don't abuse the public login systems** for this work
  - this won't be tolerated anymore
  - instead, get yourself a farm node with **`qrsh`**

Summary

- storage hardware is a problem we have to solve
- its total cost is always much higher than the price tag
- users must classify their data and store it in the right place
- make sure you get backup for data that's worth it
  - remember, it's going to happen...
- make sure you don't waste it for bulk data rewritten daily...
- 2 ways to cope with ever more farm nodes, space, & I/O:
  - either dCache + AFS + intelligent use
    - NFS for few special solutions where needed
  - or an expensive SAN + expensive commercial filesystem