dCache.org

HA dCache as cloud back-end

for dCache Team

Workshop on Cloud Services for Synchronisation and Sharing

Fermilab

NDGF
Nordic DataGrid Facility

DESY

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Three problems to solve
(we are in the late '90s)

dCache.org

I. Data never fits into a single server
   - Multiple servers
   - Off-load to tape
II. Growing number of clients
   - Main frame vs. Linux cluster
III. We want our own HW/OS selection
   - Better offers
   - Local expertise

The MISSION:

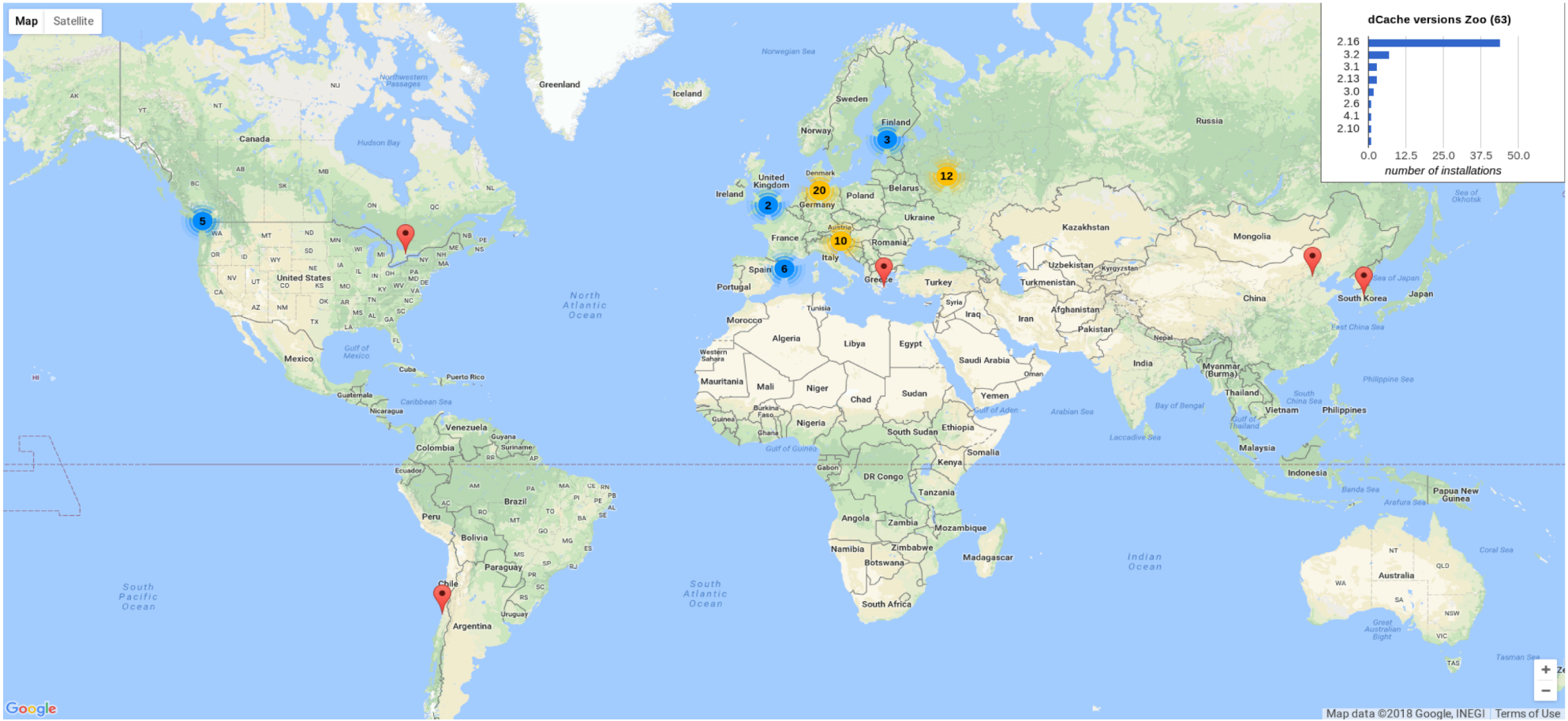
dCache.org

"... to provide a system for storing and retrieving huge amounts of data, distributed among a large number of heterogeneous server nodes, under a single virtual filesystem tree with a variety of standard access methods."

16 Sep. 2000

Michael Ernst, Patrick Fuhrmann, Martin Gasthuber, Rainer Mankel

dCache around the world



US sites in Stealth mode as they don't publish to BDII

Project Logistic

dCache.org

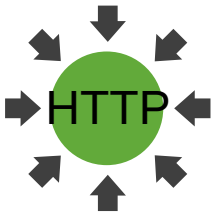
- Joint effort between DESY, FNAL and NDGF
  - developing sites use dCache as main storage system
- Open source
  - code hosted on GitHub
- Time based release model
  - one ‘golden’ release every year
    - long term (2 years) supported branch
  - quarterly feature branches
  - 861 releases since 2007

Four main components

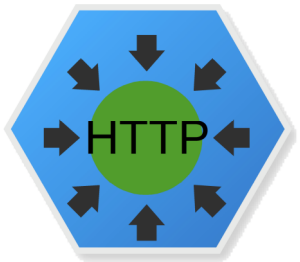
dCache.org

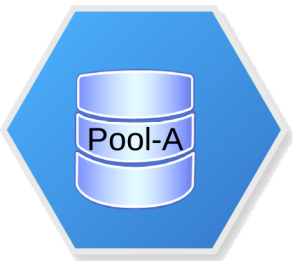
- DOOR
  - user entry points (NFS, FTP, WEBDAV)
- POOL
  - data storage nodes, talk all protocols
- Namespace
  - metadata DB, POSIX layer
- PoolManager
  - request distribution unit

Minimal Setup

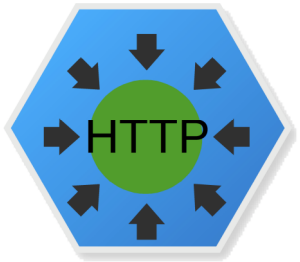


Minimal Setup

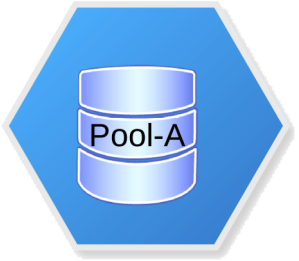




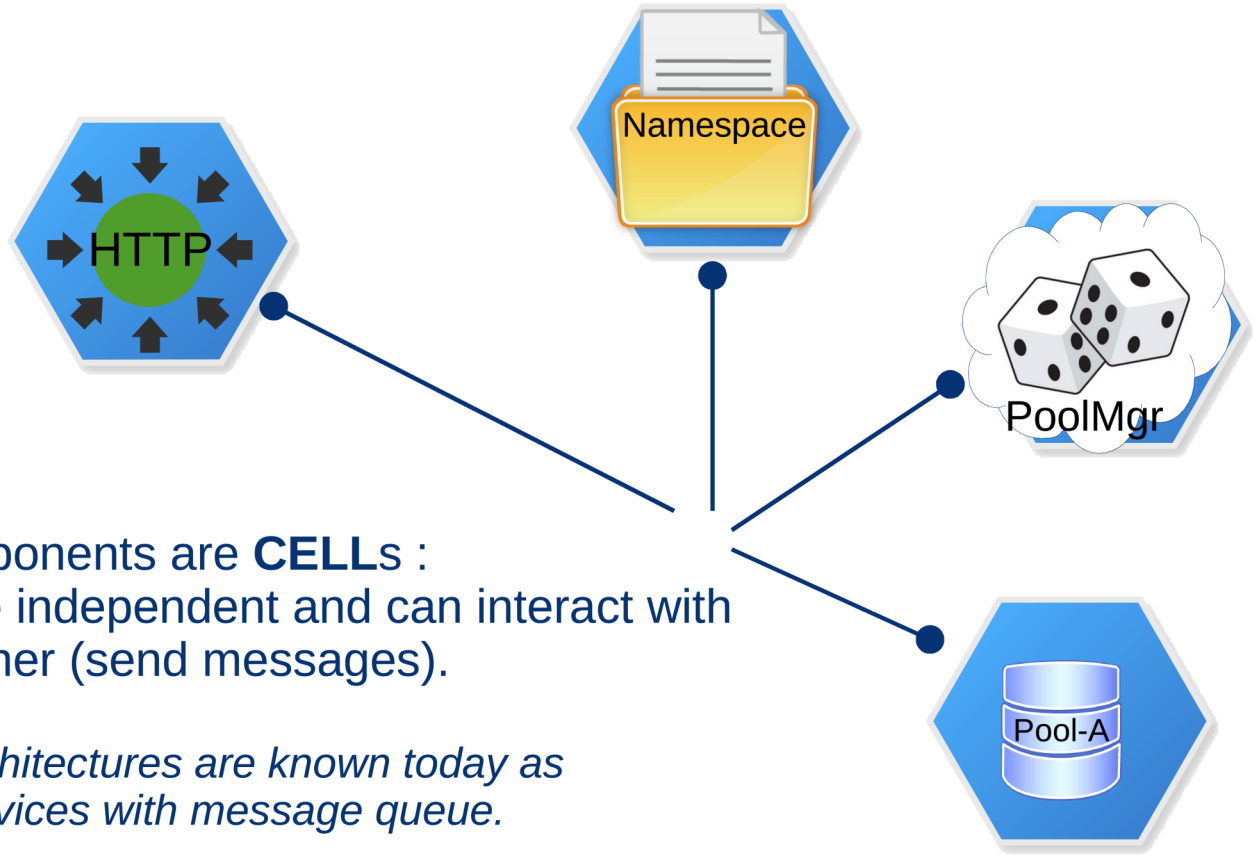
Minimal Setup



All components are **CELL**s :
they are independent and can interact with each other (send messages).



Minimal Setup

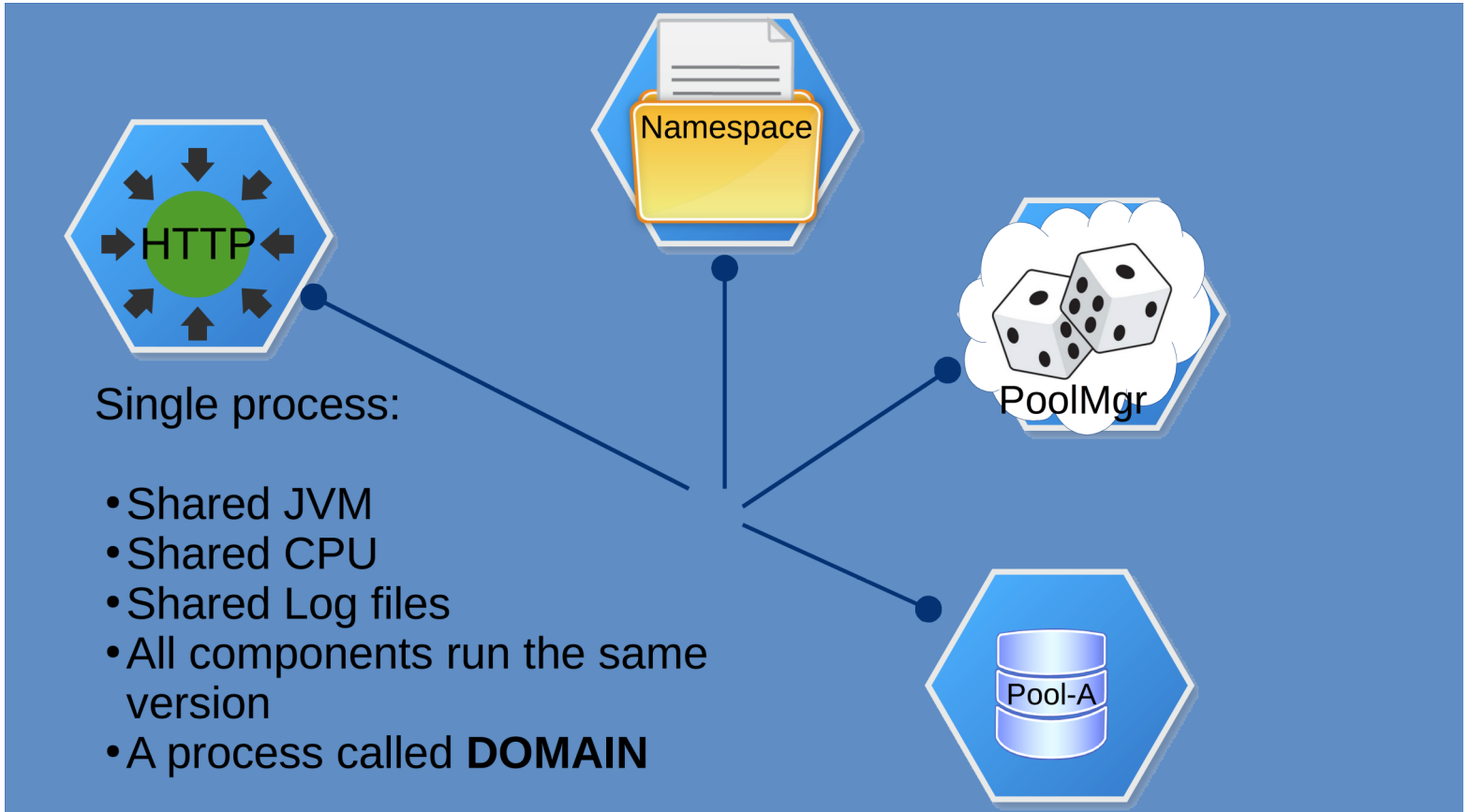


All components are **CELL**s :
they are independent and can interact with each other (send messages).

Such architectures are known today as microservices with message queue.

Grouping CELLs

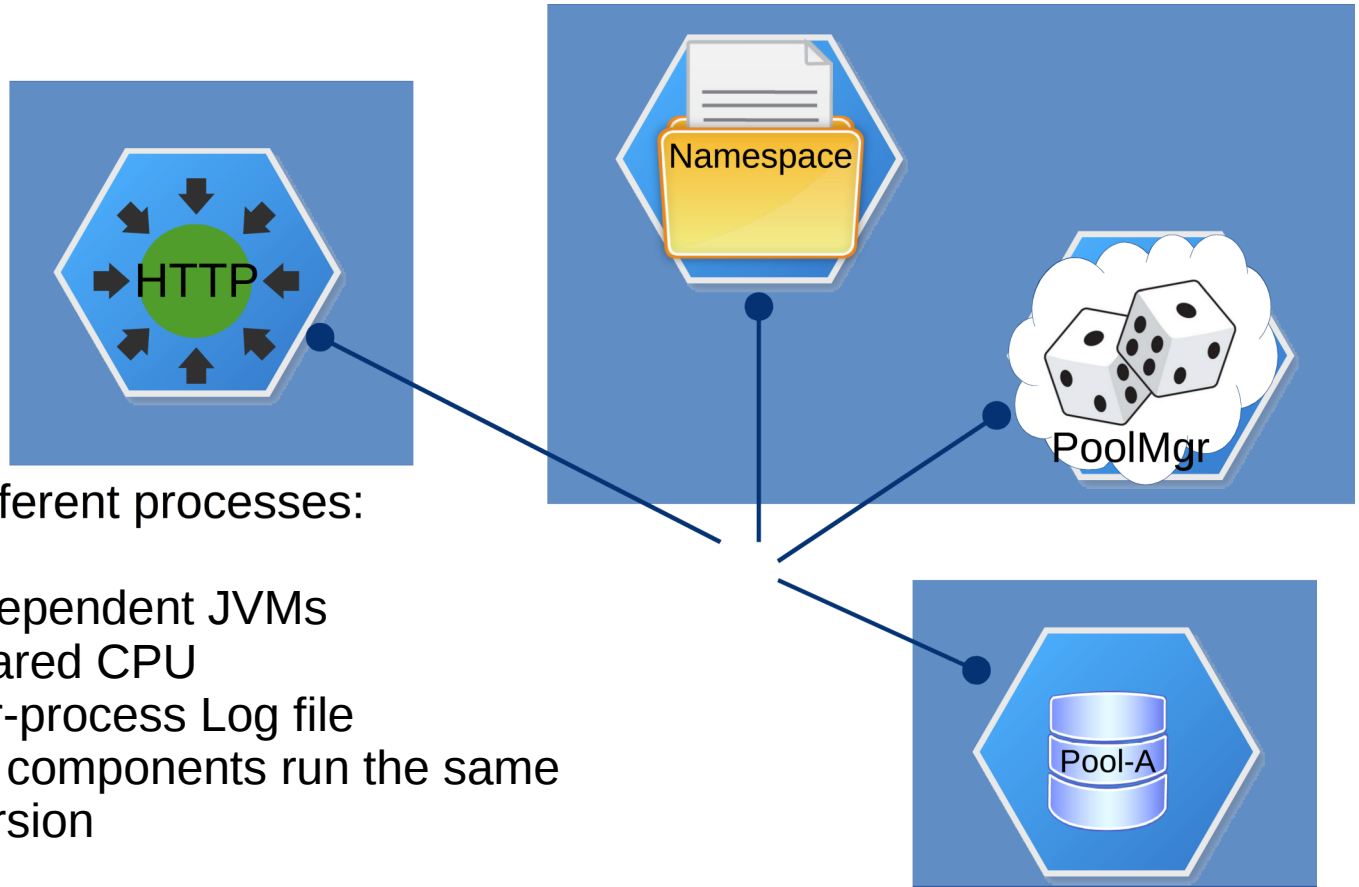
Grouping CELLs



Single process:

- Shared JVM
- Shared CPU
- Shared Log files
- All components run the same version
- A process called **DOMAIN**

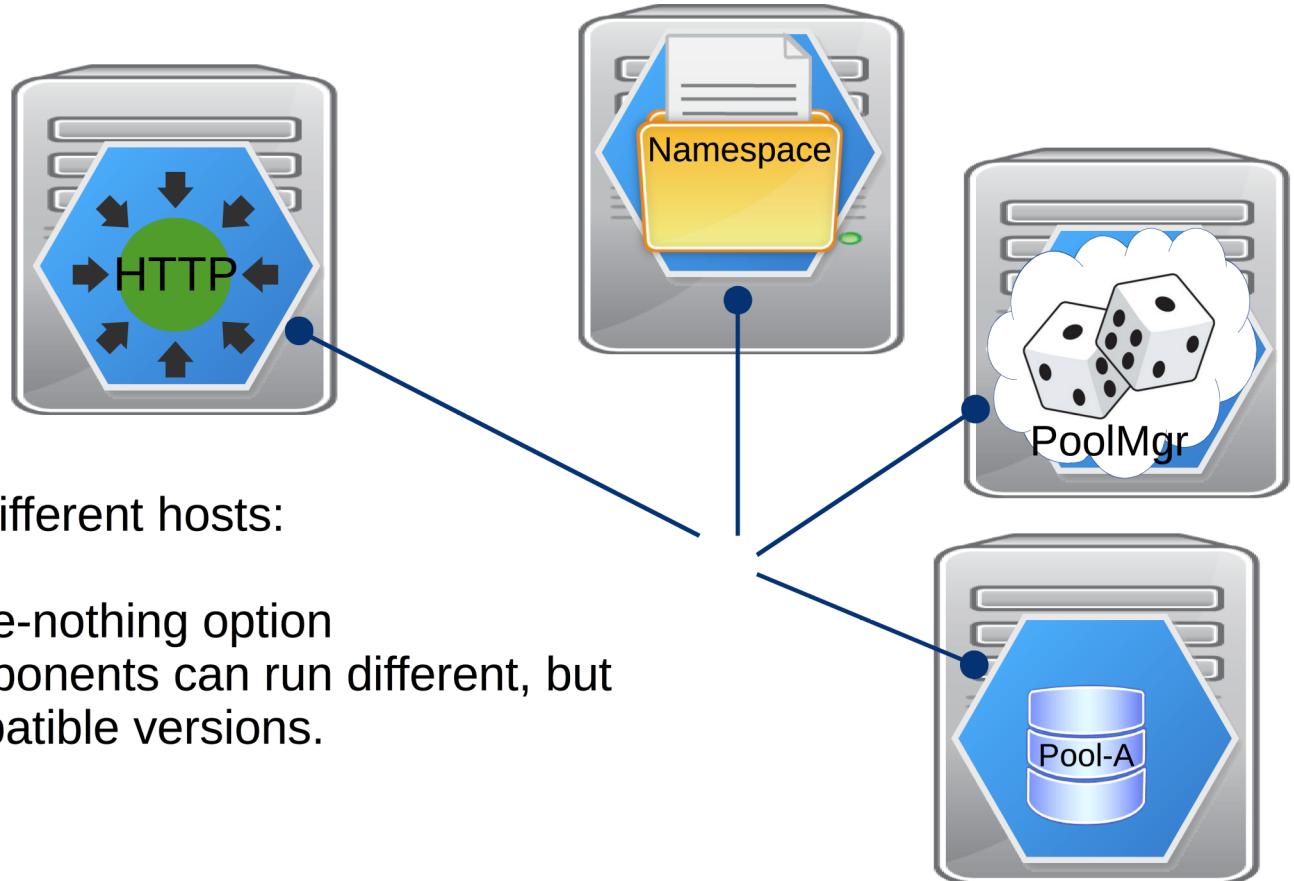
Grouping CELLs



In different processes:

- Independent JVMs
- Shared CPU
- Per-process Log file
- All components run the same version

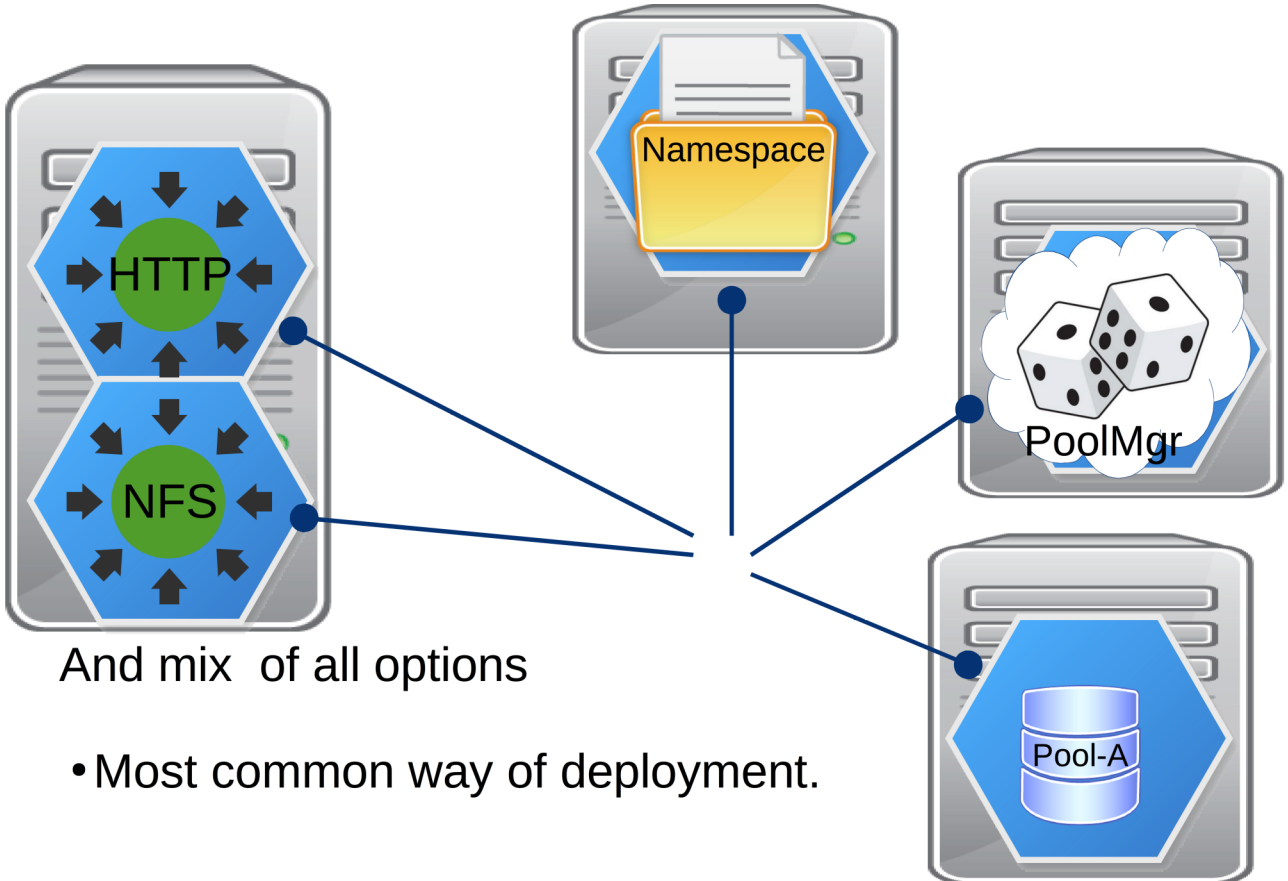
Grouping CELLs



On a different hosts:

- Share-nothing option
- Components can run different, but compatible versions.

Grouping CELLs



And mix of all options

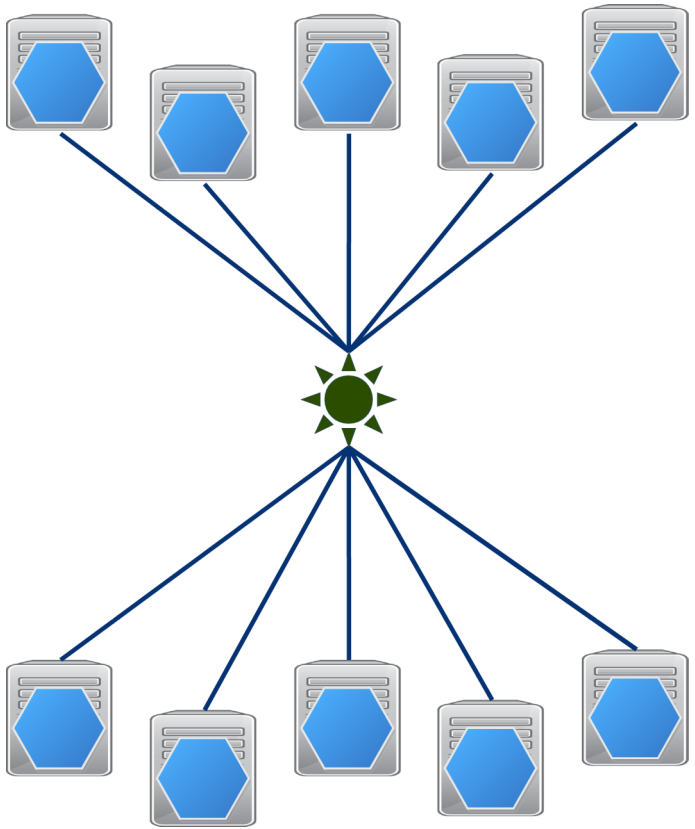
- Most common way of deployment.

Interconnecting components

dCache.org

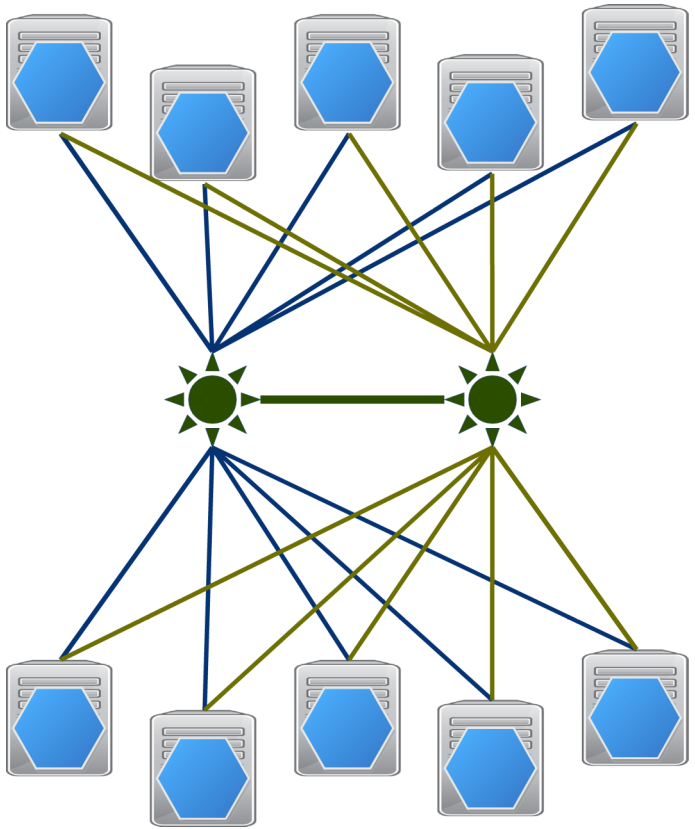
- CELL messages
  - used for inter component communication
  - Starting dCache v3.2 supports TLS
- ZooKeeper
  - 3rd party product (from the Hadoop universe)
    - a distributed hierarchical key-value store, providing distributed configuration & synchronization services and naming registry for large distributed systems
  - used by dCache for service discovery
  - no TLS support in stable release, but can be secured with **stunnel**

Cell messaging 101



- Star like topology
- Selected node configured as a hub called **CORE** domain
- All communication goes through CORE domains
- Other domains called **SATELLITE**

Cell messaging 101



- Star like topology
- Selected node configured as a hub called **CORE** domain
- All communication goes through **CORE** domains
- Multiple **CORE** domains make communication fault tolerant

Internal Communication

dCache.org

- Inter-cell communication
  - Message passing
  - Routing
- ZooKeeper
  - Service discovery
  - Coordination
  - Configuration

ZooKeeper in dCache

- Distributed key-value-store
- A central registry of CORE domains
  - Similar to DNS or routing table
- Master election for single-man operations
  - Some actions must be done only once
- Must be deployed as a cluster
  - built-in version for kick-off

Securing communication

- dCache-3.2 supports TLS for CELL messages
- ZooKeeper doesn't support TLS out of box
  - We recommend **stunnel** to use as proxy

STUNNEL

- Proxy to add TLS to legacy software
- No code change requires
- Easy to install
  - required on both endpoints

ZooKeeper with STUNNEL

dCache.org

dcache.zookeeper.connection = localhost:2181

TLS

```
[zookeeper]
; client mode
client = yes
accept = 2181
connect = stunnel.noname.de:2182
cert = /etc/stunnel/stunnel.pem
CApath= /opt/noname/certs
checkHost= stunnel.noname.de
```

```
[zookeeper]
; server mode
accept = 2182
connect = zoocluster1.noname.de:2181
cert = /etc/stunnel/stunnel.pem
CApath= /opt/noname/certs
checkHost= stunnel.noname.de
```

Network Authentication

- TLS based
- Client/server authentication based on host certificates
- Ideally, dedicated CA
  - Works with dCache and stunnel

Fault tolerance

dCache.org

- All core services can run multiple instances (replicable)
  - Namespace
  - Pool Manager
- Door/Pool crashes can be handled by clients
  - NFS
  - dcap
  - xrootd
- File replication protects against data loss
- Master/slave postgres config required
  - dCache detects which node runs as master when both provided

Upgrades

dCache.org

- Replicable services can be upgraded at any time
- Pools/doors may require draining
- Postgres can be upgraded within same major version
  - 3rd party tools available for fail-over management

norden NordForsk neic Nordic e-Infrastructure Collaboration

HA dCache upgrades

- With the new HA features in dCache we can do system updates including reboot into new kernels with no downtime
- Can typically be done in a day, but takes a bit of watching to make sure we don't interrupt any client accesses
- Can also do dCache upgrades of headnodes without anyone noticing, over a couple of days
  - Unless something goes wrong, of course
- Hardware and headnode upgrades on different days
  - Headnode upgrades depends on haproxy draining state – this is reset by a reboot of the hardware that runs haproxy

Mattias Wadenstein, NDGF Site Report, HEPiX Spring 2017

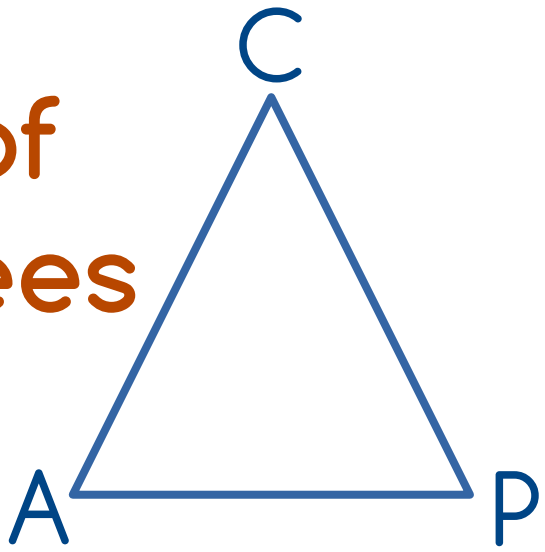
5

CAP

- Consistency
  - Every read receives the most recent write or an error.
- Availability
  - Every request receives a (non-error) response – without guarantee that it contains the most recent write.
- Partition tolerance
  - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

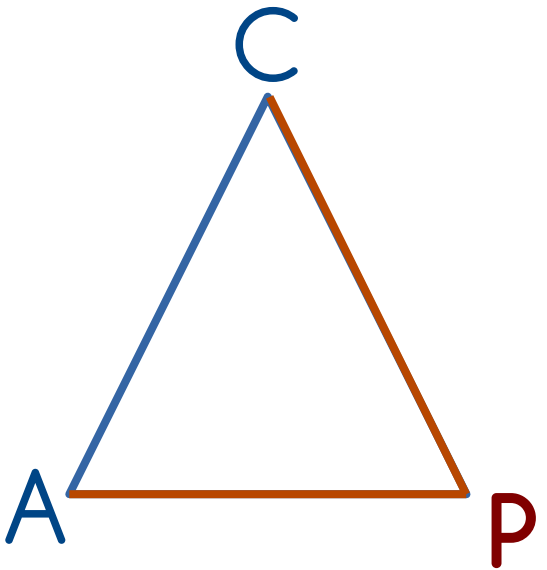
The CAP Theorem:

it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees



CAP theorem explained

- If the network is OK, both availability and consistency can be satisfied, but...
- No distributed system is safe from network failures.
- The choice is between consistency and availability is forced when a network partition or failure happens.



dCache and CAP

- dCache provides consistency over availability.
- All clients will see the same data at the same point in time.
- A timeout or error will be returned, if consistency can't be guaranteed.

Summary and Conclusions

dCache.org

- dCache has a long tradition in providing reliable and scalable storage for many sites around the world.
- Fault-tolerant setup allows zero-downtime operation

- This predestines dCache as the platform for DESY cloud storage
- And maybe yours?