Federated dCache

Pools at Different Pool-Sites, Single Federated Management

Christian Voß, for DESY-HH dCache Operations Team and dCache Development Team

Basic Ideas of dCache

Solving the Questions of Scale

Features

- High horizontally scalable storage system
- Expose a single unified namespace
- Supports many protocols
- Supports many authorisation schemes
- Micro-service architecture

Scale-Out Mechanisms

- Extent storage by adding additional storage nodes
- No rebalancing of the cluster necessary
- Extent/Mitigate user-traffic by adding new access points
- No impact on existing access points
- Management cells can be replicated to increase resilience and mitigate load

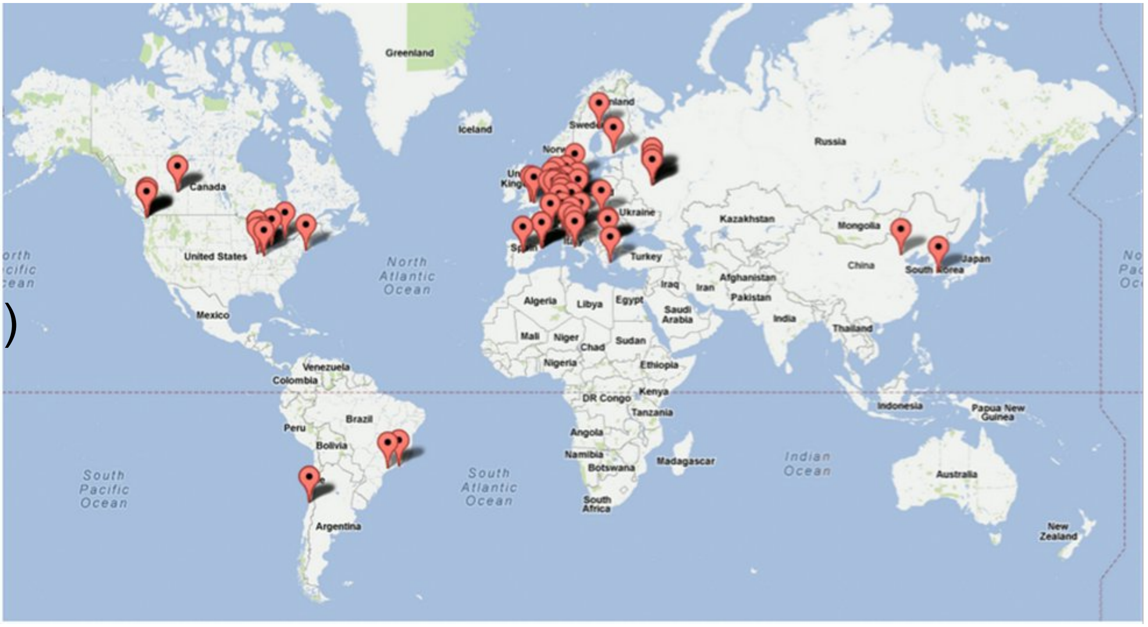
Everything shown is available in dCache but requires deep knowledge of the system and a its configuration

Mass-Storage for LHC and Belle II in WLCG

dCache.org

dCache as Central Mass Storage for HEP communities

- Central element in overall storage strategy
- Collaborative development under open source licence by
  - DESY
  - Fermilab
  - Nordic E-Infrastructure Collaboration (inoffically NDGF)
- Particle Physics in general
  - In production at 9 of 13 WLCG Tier-1 centres
  - In use at over 60 Tier-2 sites world wide
  - 75% of all remote LHC data stored on dCache
  - In addition: Tevatron and HERA data
- DESY
  - Raw-Data for smaller Particle Physics experiments
  - Raw-Data for Photon Science Archival
  - Mass-storage for user data during analysis
  - Long-term archival



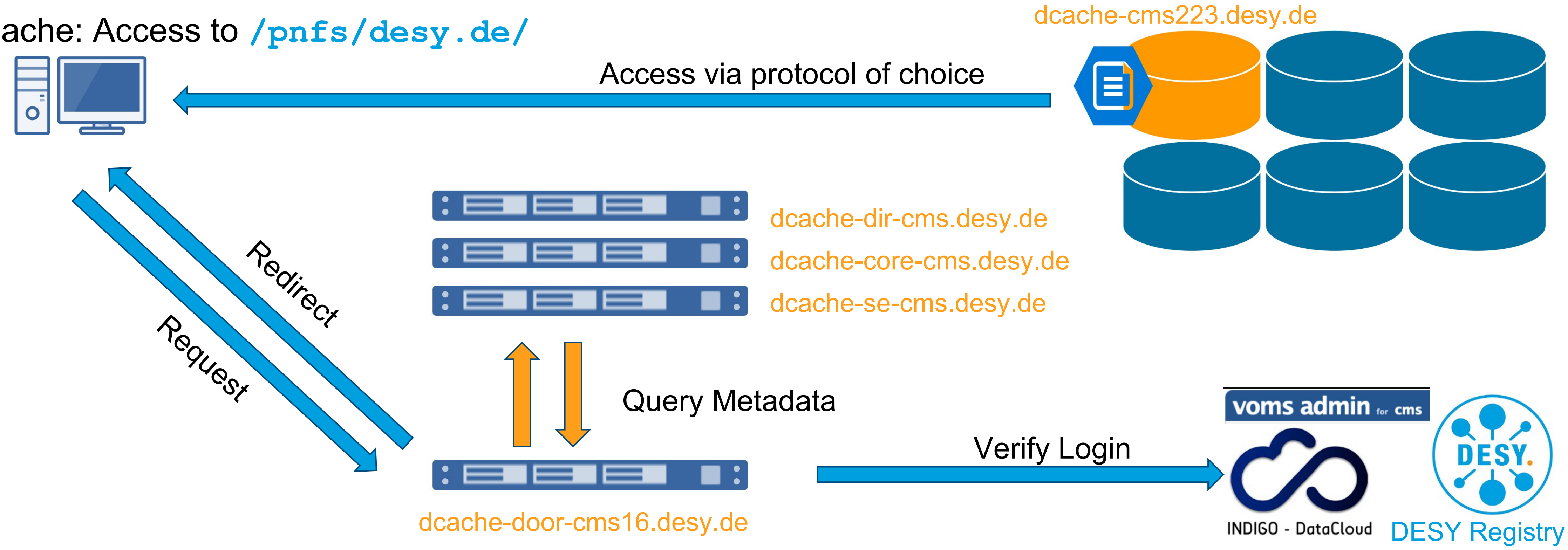
Features

- Highly horizontally scalable storage system
- Expose a single unified namespace
- Supports many protocols
- Supports many authorisation schemes
- Micro-service architecture

Basic Setup

Standard Single Site Setup

- Use dCache: Access to `/pnfs/desy.de/`



- dCache instances for Photon Science/Machine, European XFEL, ATLAS, CMS, Belle/ILC/DPHEP, Sync&Share
- Similar layout: three head-nodes, doors for requested protocols and pools nodes
- Scale-out horizontally: ~100 pools for HIFIS and ~4000 for European XFEL
- Scale-out horizontally: client always to connect to pools for transfer, no data access through doors

How Does dCache Work in more Detail

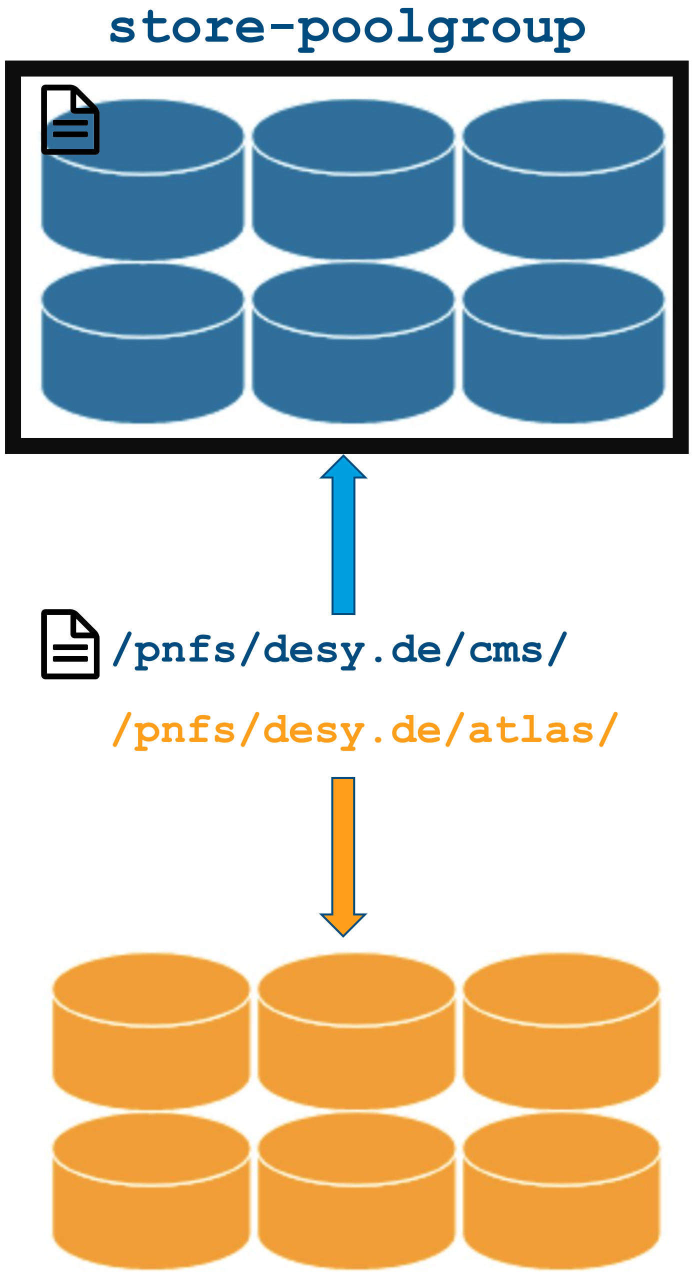
How Files End up on the Pools — In a Nutshell

Customer Setup

- Two distinct customers, **CMS** and **ATLAS**
- Pledged storage for each experiment ➜ Assign storage to customer
- Have ATLAS storage independent of CMS

dCache Pool Selection

- dCache links directories typically to collections of pools (pool-groups)
- Links are based and store, network and protocol units
- Units have wild-card capabilities
  - **store:** based on directory tags; files are written to any pool linked to it (basically the only units used by most sites)
  - **net:** clients matching certain subnets in IPv4 and IPv6
  - **protocol**: match depending on the protocol

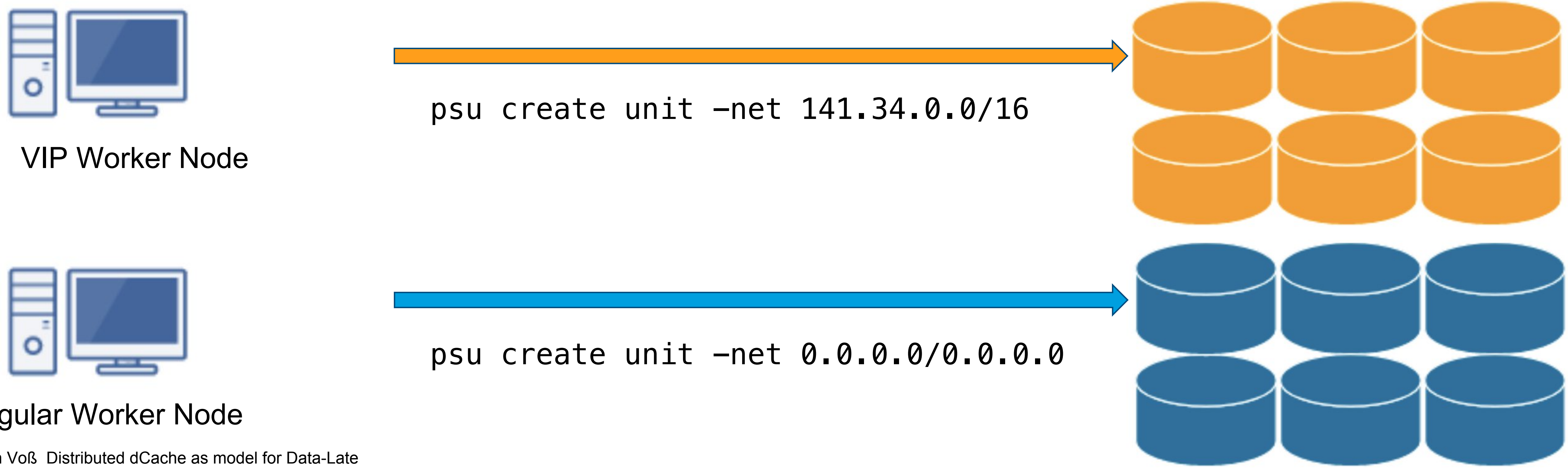


Examples for Using Different Pool Selection Units

Cluster with NFS Access to Fast SSD Pools

Dedicated compute cluster for VIP usage equipped with SSDs

- Users having local access using the faster disks
- Keep a unified namespace
- Since access protocols are typically universal for all customers
  ➜ Use a network unit covering the specific cluster (configurable down to the individual nodes)



VIP Worker Node

```
psu create unit -net 141.34.0.0/16
```

Regular Worker Node

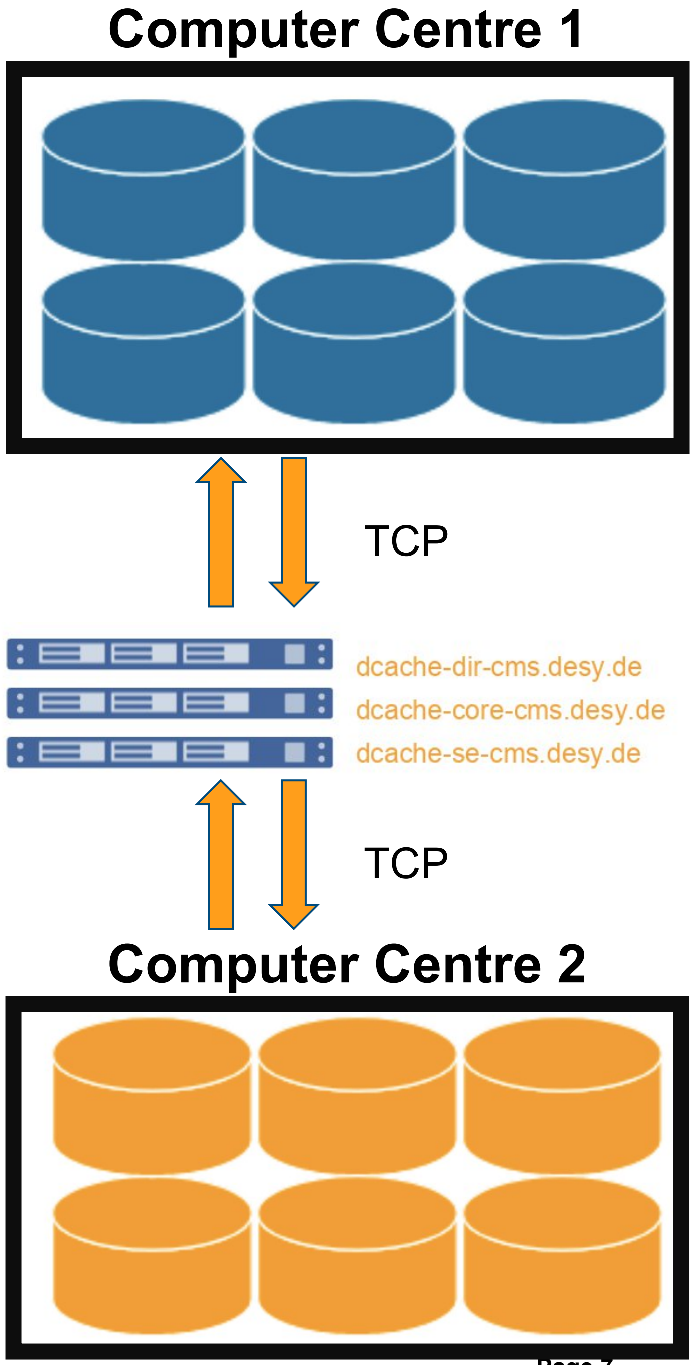
```
psu create unit -net 0.0.0.0/0.0.0.0
```

How Does dCache Work in more Detail

dCache Pools — In a Nutshell

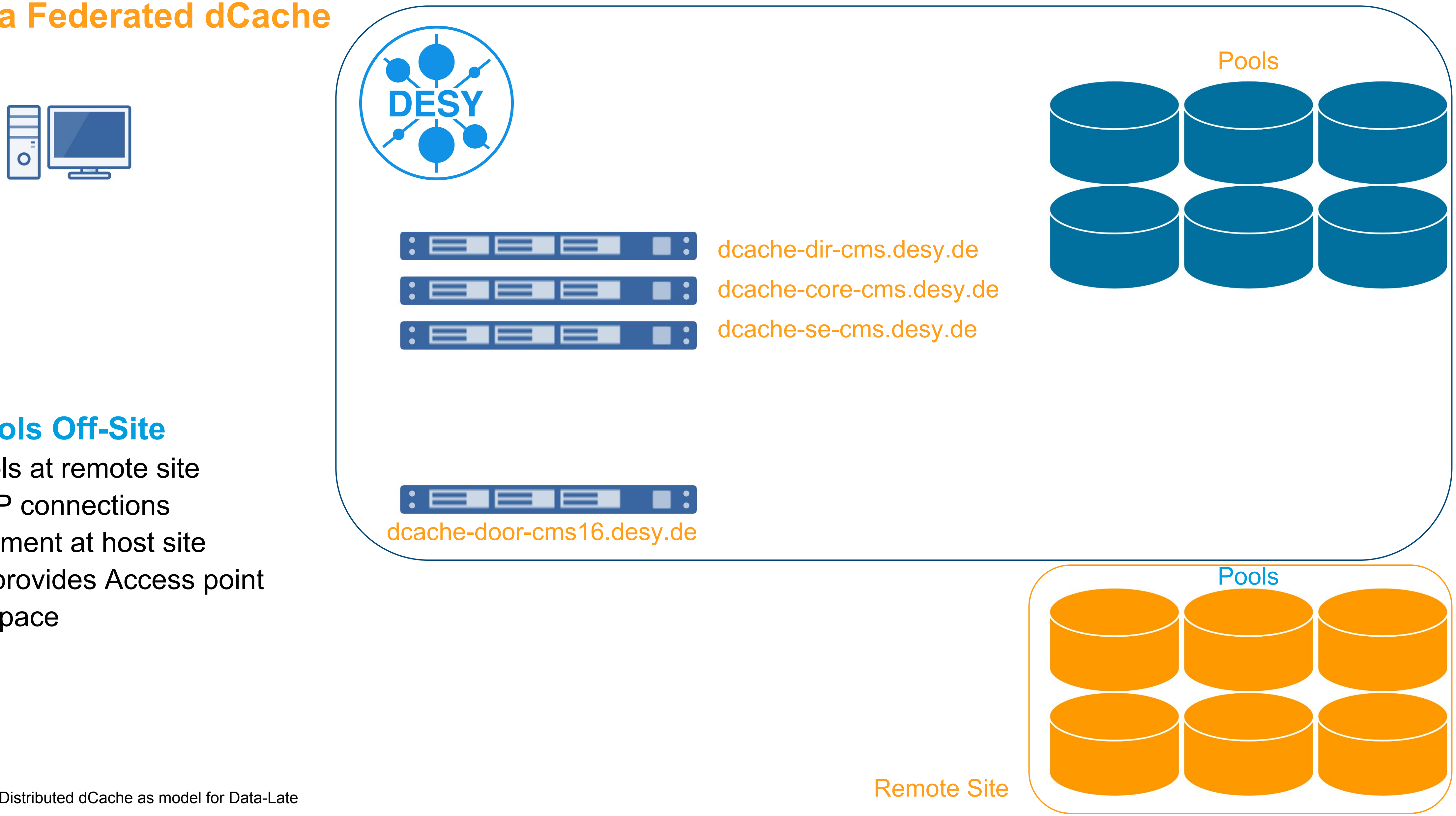
Core Features for Horizontal Scaling

- Pools are an atomic service in dCache
- By default: all data-traffic between pool and client
- All pools are independent of each other
- Pools keep their inventory stored in their own database
- Pools register themselves with dCache
- Through the dCache messaging system pool push and pull the needed information
- dCache pools can be located almost anywhere so long as there is a TCP connection available between pool, core services and client



Placing dCache Pools Off-Site

Forming a Federated dCache

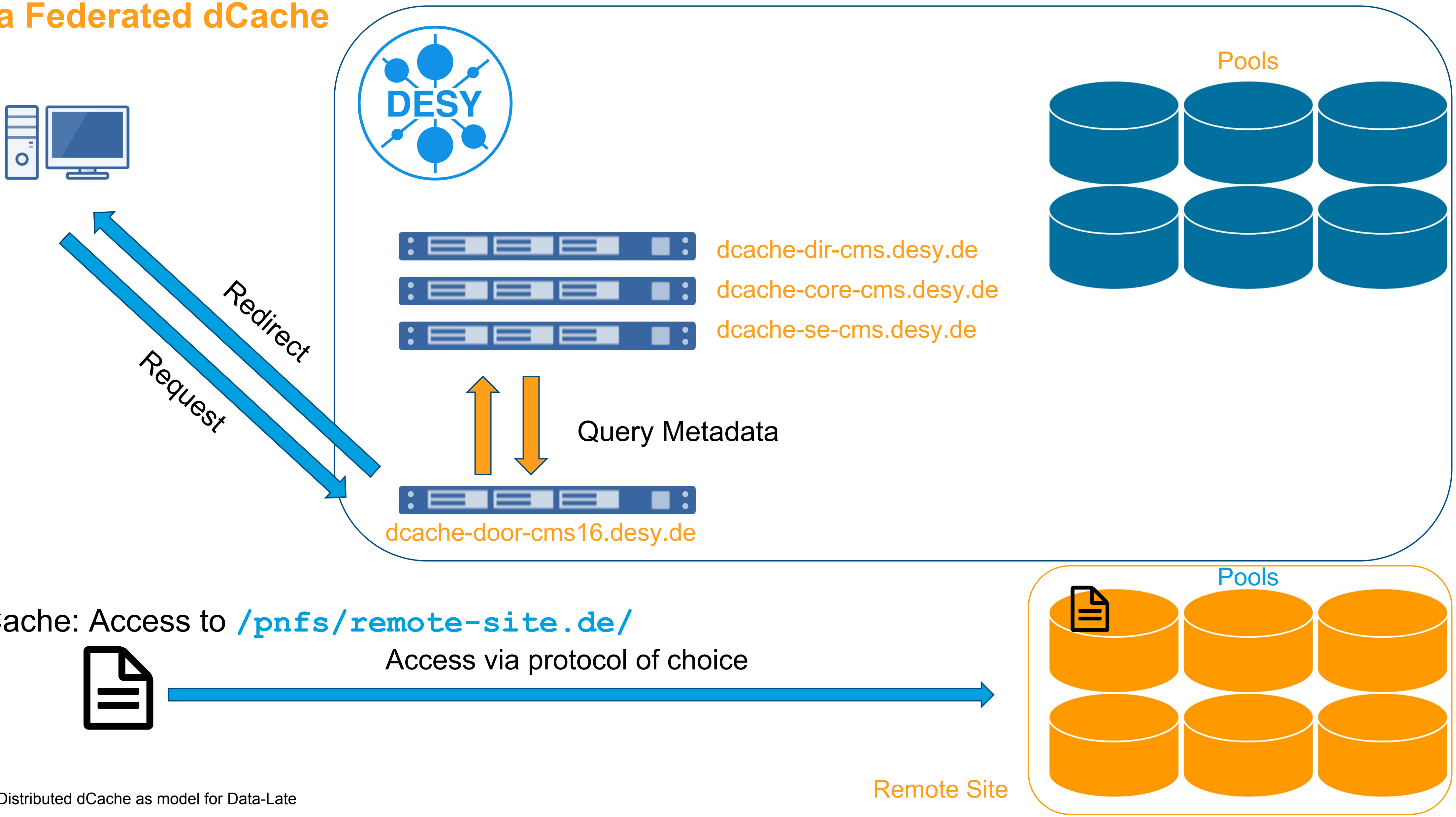


Running Pools Off-Site

- Deploy pools at remote site
- Ensure TCP connections
- All management at host site
- Host sites provides Access point and namespace

Placing dCache Pools Off-Site

Forming a Federated dCache



- Use dCache: Access to `/pnfs/remote-site.de/`

Redirect

Request

DESY

dcache-dir-cms.desy.de

dcache-core-cms.desy.de

dcache-se-cms.desy.de

Query Metadata

dcache-door-cms16.desy.de

Pools

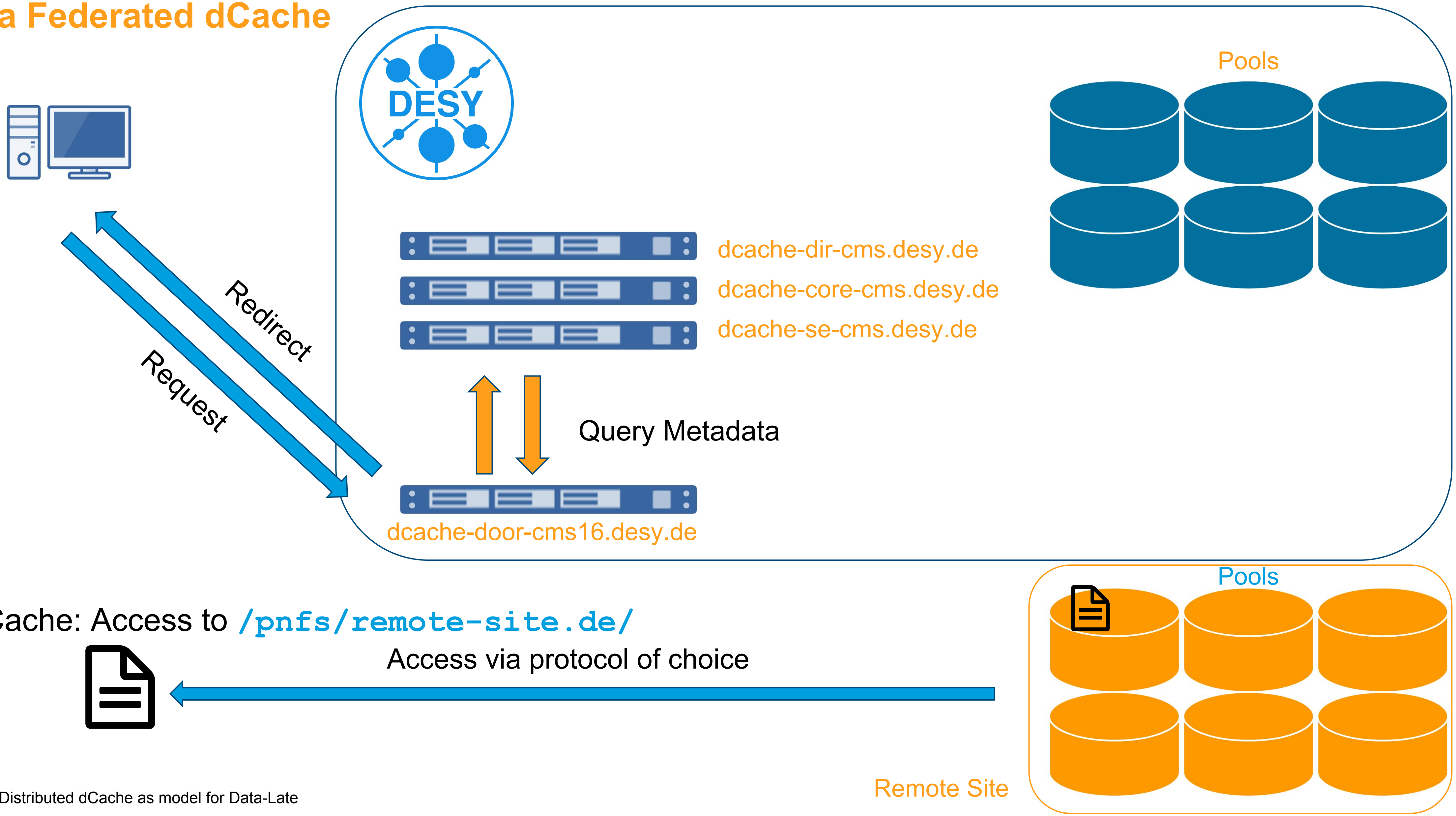
Pools

Access via protocol of choice

Remote Site

Placing dCache Pools Off-Site

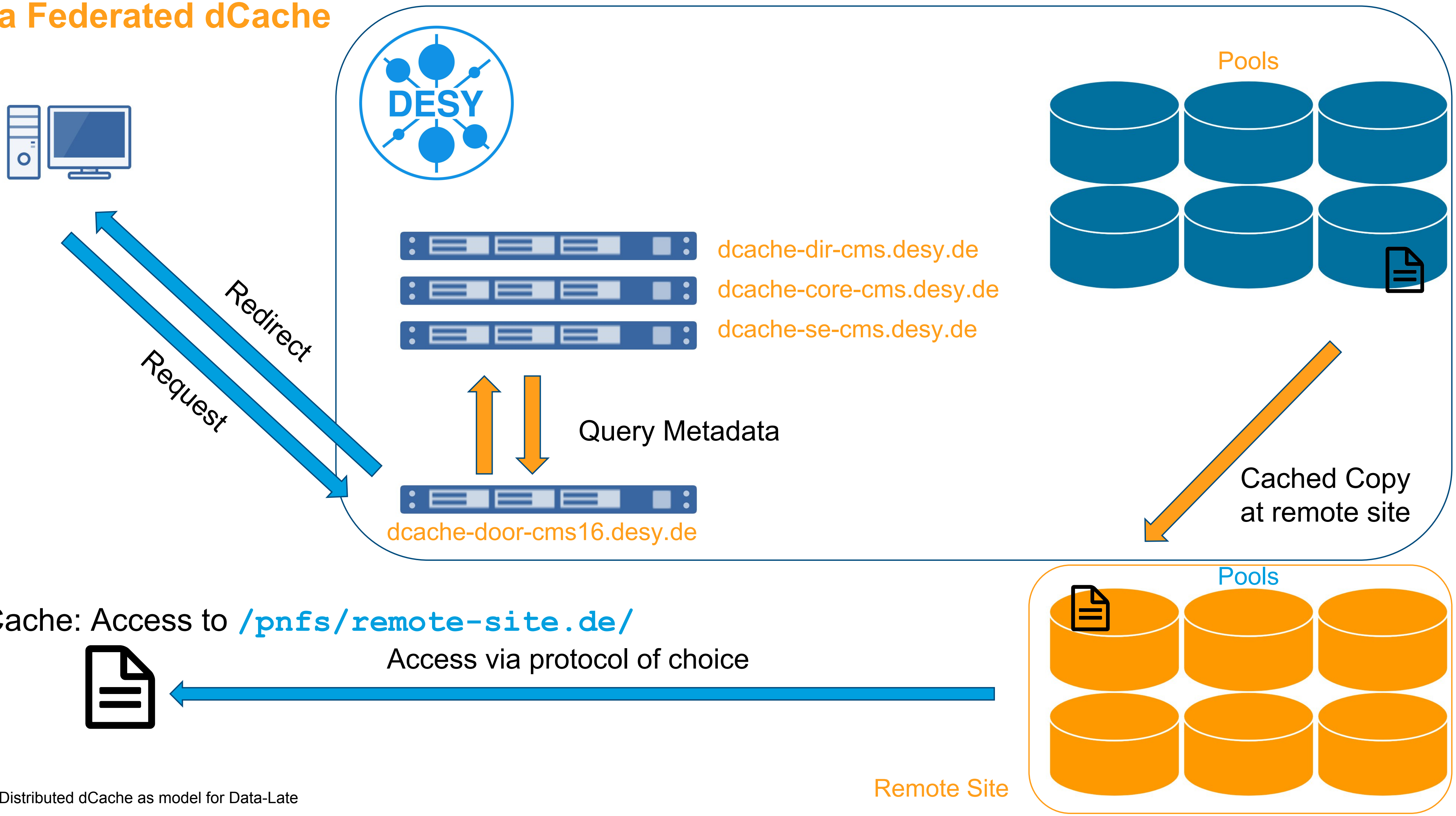
Forming a Federated dCache



- Use dCache: Access to `/pnfs/remote-site.de/`

Placing dCache Pools Off-Site

Forming a Federated dCache



- Use dCache: Access to `/pnfs/remote-site.de/`

How to Populate Remote Pools

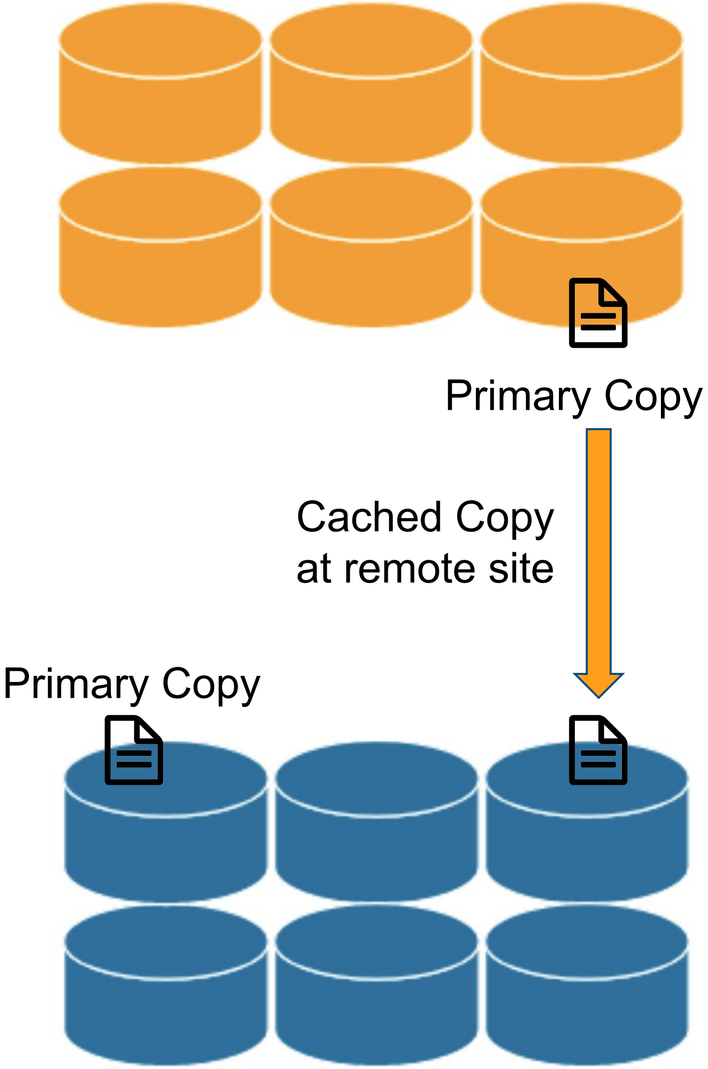
From Classic Disk Site to Cached on Demand Site

Primary Copy

- Straight forward, works exactly like on the host site
- Will never be removed from disk unless deleted by user
- No copy in host sites

Secondary (i.e. Cached) Copy

- Primary copy on host site; cached copy remotely
- Cached copy similar to tape interaction
  - No copy on remote pool ➜ file is NEARLINE
  - Remote copy can be generated on access ➜ files is staged from the host site (like from tape)
  - Remove copy creation can be triggered ➜ triggering at stage through bring-online mechanisms
- Tape workflows could be employed here, but also QoS in dCache (pinning vs. controlled state)



How to Use dCache to Build Remote Caches/Federated Pools

Overview of Possible Scenarios

1. Independent Cached-Only Site

2. Cache-Only Site Connected to Primary Site with shared namespace

3. Cache-Only Site Attached to Primary Site with independent namespace

4. True Satellite Site with varying resilience

Independent Cached-Only Site

Site Run Like a Normal dCache Site — But Cached Only

- Least attractive Scenario
- Require the full setup of head nodes, doors and pools
- Need database infrastructure to keep and manage the namespace
- Requires an dCache expert to be on remote site
- Site can be configured to keep files cached only as a ringbuffer

Cache-Only Site Connected to Primary Site

With Different Namespace Scenarios

- Remote site only hosts pools
- Different scenarios on how to place files:
  1. Cached site is a mirror of files on the primary site
  2. Cached sites receives their own primary site through experiments (Cached site looks like an independent site)
- Update cycle links and mostly undertaken by primary site (think NDGF model)

Cached site is a mirror of files on the primary site

- Use the staging mechanisms to populate the remote storage

Cached sites receives their own primary site through experiments

- Primary sites offers entry point through the experiment data management

True Satellite Site with Varying Resilience

Akin to Managing the Existing University Tier-2 Sites from Primary Site

- Would be attractive Scenario for the existing Tier-2 structure
- Might still be interesting for local Tier-3 clusters (idea behind the studies done with Wuppertal)
- Idea of a completely independent client site namespace
- There are no primary copies of satellite site data on primary site
- All management services can be managed at primary site including end-points (NFS being more complicated)
- Local site only operates storage nodes

Concerns with Resilience

- Primary sites hosts all services: local site bound to network stability
- Can be mitigated by replicating services locally ➜ increases support load on remote site or primary site

Summary

Advantages and Disadvantages

Features

- Centralised namespace (think of it like Rucio)
- Centralised interface/connection to the AAI
- Resilience configurable
- Configured as permanent storage or cache

Advantages

- Provide a variety of protocols for the users
- Expose only one endpoint to experiments
- Single, centralised configuration and administration
- Little load on admin at remote sites (pools only)
- Reduce invest costs at remote site (pools only)
- Similar setups in production: NDGF-Tier1, Great Lakes Tier2

Disadvantages

- Reduced independence of remote site
- Reliance on stability and responsiveness of local/remote sites
- Reliance on WAN for all metadata operations (can be mitigated)
- Setup requires expert knowledge ➜ **future funding important to make this more steam lined**