Распределенные хранилища данных для крупных научных экспериментов и управление ими на основе метаданных провенанса

А. П. Крюков, А. П. Демичев

НИИЯФ МГУ

Семинар Московской Секции ACM SIGMOD
28.03.2019

Исследование выполнено за счет грантов Российского научного фонда No. 18-41-06003 и No.18-11-00075

Distributed Data Srorages for Large Scientific Experiments and their Management Based on Provenance Metadata

A. Kryukov and A. Demichev

SINP MSU

Seminar of the Moscow
ACM SIGMOD Chapter
28.03.2019

Supported by RSF grants
No.18-41-06003 & No.18-11-00075

Part 1

Distributed Data Storages for Large Scientific Experiments

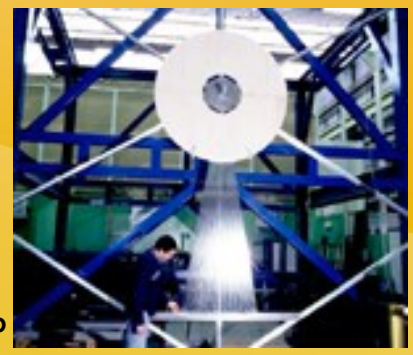
A.Kryukov, etc. "Distributed data storage for modern astroparticle physics experiments.", Proceedings of the 3-d Int. Workshop "Data Life Cycle in Physics Experiments", Irkuts, 2019, CEUR-WS (to be pulished)

Examples of Large Experiments and Distributed Storages: WLCG (1/2)

- The Worldwide LHC Computing Grid (WLCG)
  - It was designed by CERN to handle the prodigious volume of data produced by Large Hadron Collider (LHC) experiments in high-energy (elementary particle) physics
    - approximately 25 petabytes per year
  - an international collaborative project
  - **grid**-based computer network infrastructure incorporating over 170 computing/storage centers in 36 countries

CMS ***LHCb*** ***ATLAS*** ***ALICE***

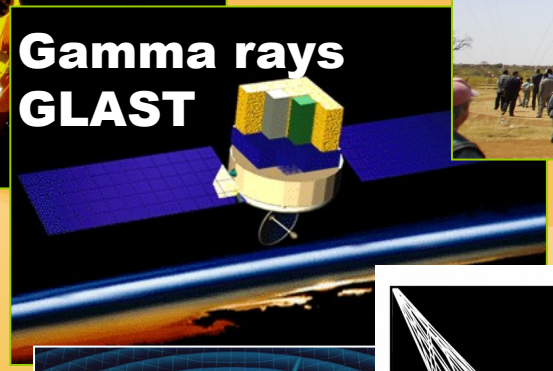


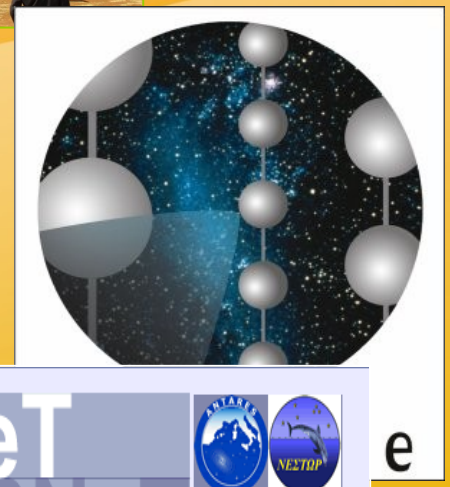



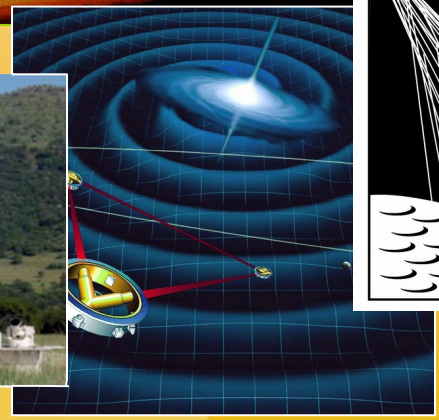
Examples of Large Experiments and Distributed Storages: WLCG (2/2)

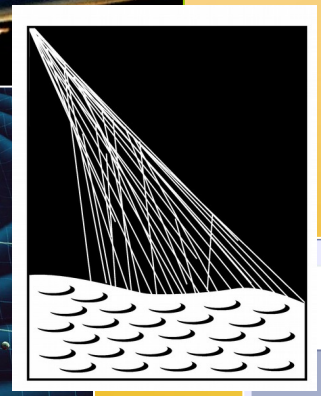
- time of active work of LHC ⇒ generation of big scientific data, is several tens of years, and the processing time of the data will be at least twice as much
  - without detailed and correct PMD comparing the results obtained with an interval, for example, in a few years, will be simply impossible

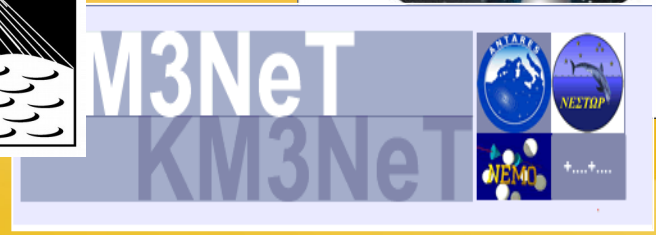
Multimessenger astronomy





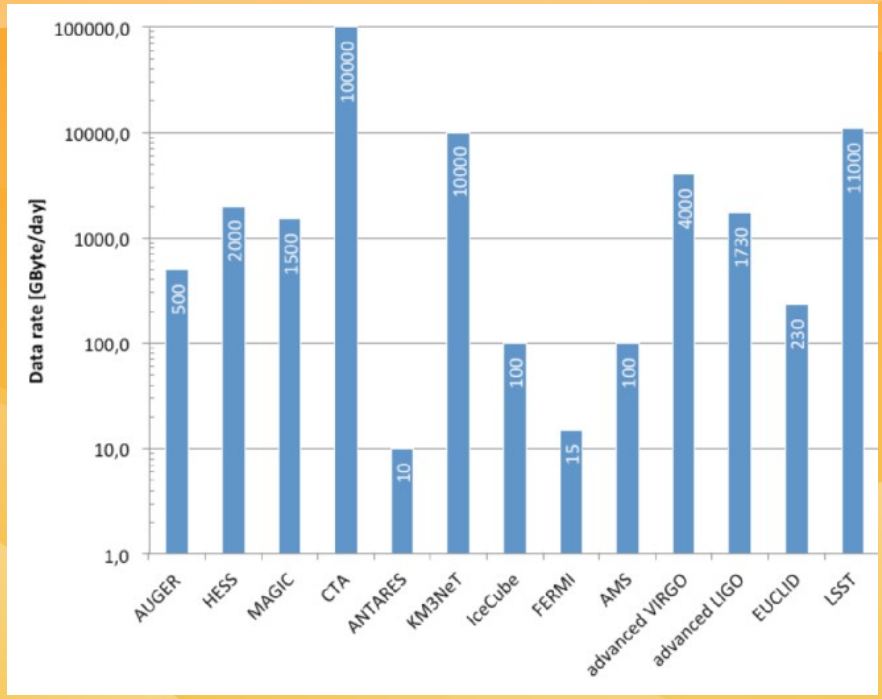


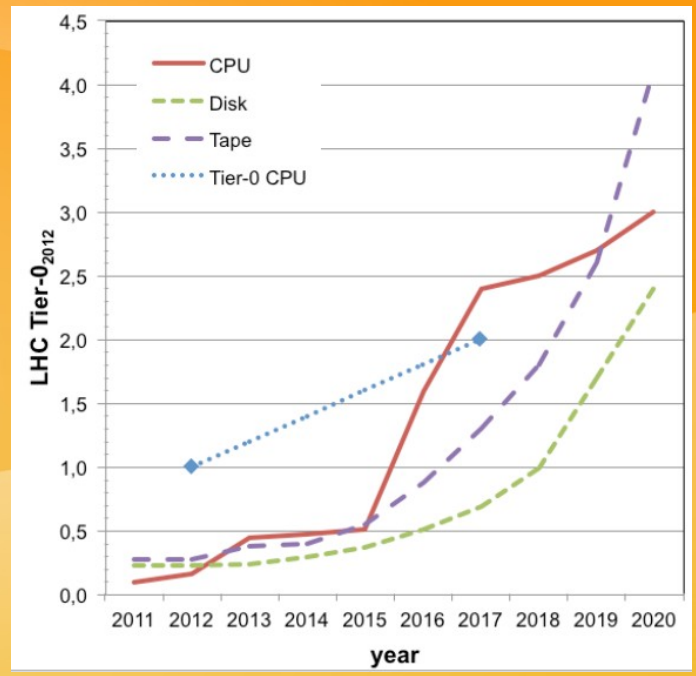




Examples of Large Experiments and Distributed Storages: Astrophysics

Berghöfer, T., et al. "Towards a model for computing in european astroparticle physics." ArXiv: 1512.00988 (2015)





- present-day experimental facilities generate data sets ranging in size from 100's to 1000's of terabytes per year.

Data challenge in astronomy

- LSST imaging at this rate will generate about 15 terabytes (15 trillion bytes) of raw data per night and 30 petabytes over its 10-year survey life.
  - A petabyte is approximately the amount of data in 200,000 movie-length DVDs.
- Even after processing, that's still a 15 PB (15,000 TB) store.

Sky Survey Projects	Data Volume
DPOSS (The Palomar Digital Sky Survey)	3 TB
2MASS (The Two Micron All-Sky Survey)	10 TB
GBT (Green Bank Telescope)	20 PB
GALEX (The Galaxy Evolution Explorer)	30 TB
SDSS (The Sloan Digital Sky Survey)	40 TB
SkyMapper Southern Sky Survey	500 TB
PanSTARRS (The Panoramic Survey Telescope and Rapid Response System)	~ 40 PB expected
LSST (The Large Synoptic Survey Telescope)	~ 200 PB expected
SKA (The Square Kilometer Array)	~ 4.6 EB expected

Types of storages: extremal cases

- Centralized
  - problems:
    - very expensive ⇒ funding ?
    - planning in advance the necessary storage capacity
- P2P-storage with special mechanisms of coding, fragmentation and distribution
  - problems:
    - to ensure a stable pool of resource providers,
    - before such a P2P-based storage can work stably, it requires significant technical, organizational and time costs in the absence of a result guarantee

Types of storages: intermediate solution

- organizations participating in a large project
  - integrate their local storage resources into a unified distributed pool
  - if necessary, rent in addition cloud storage resources, perhaps from multiple providers.
- may be particularly advantageous
  - if there is a need to store large amounts of data for a limited duration of a project
  - in a situation where the project brings together many organizationally unrelated participants
- ⇒ dynamically changing distributed environment

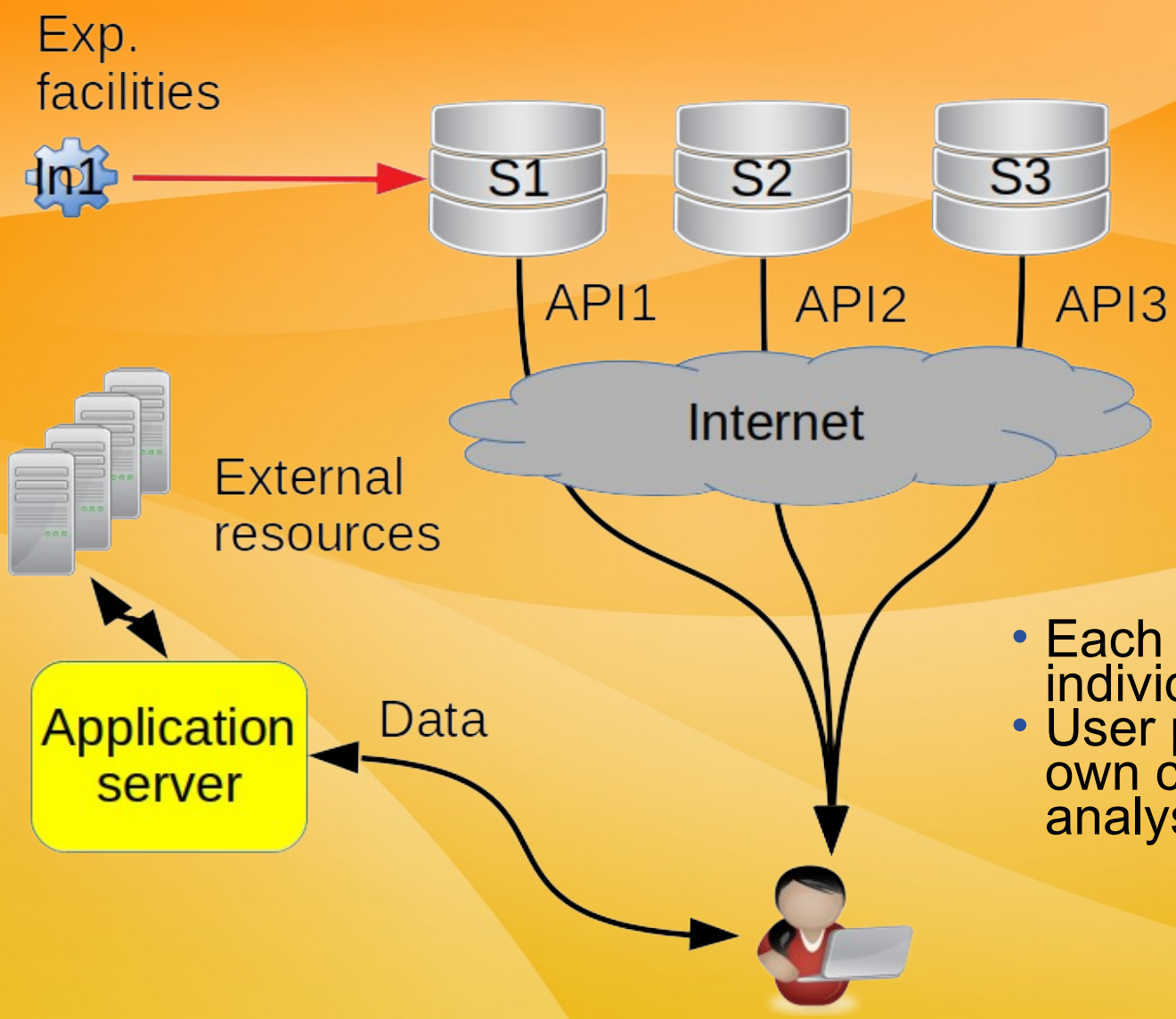
Architecture of DS for megascience experiments

Motivation

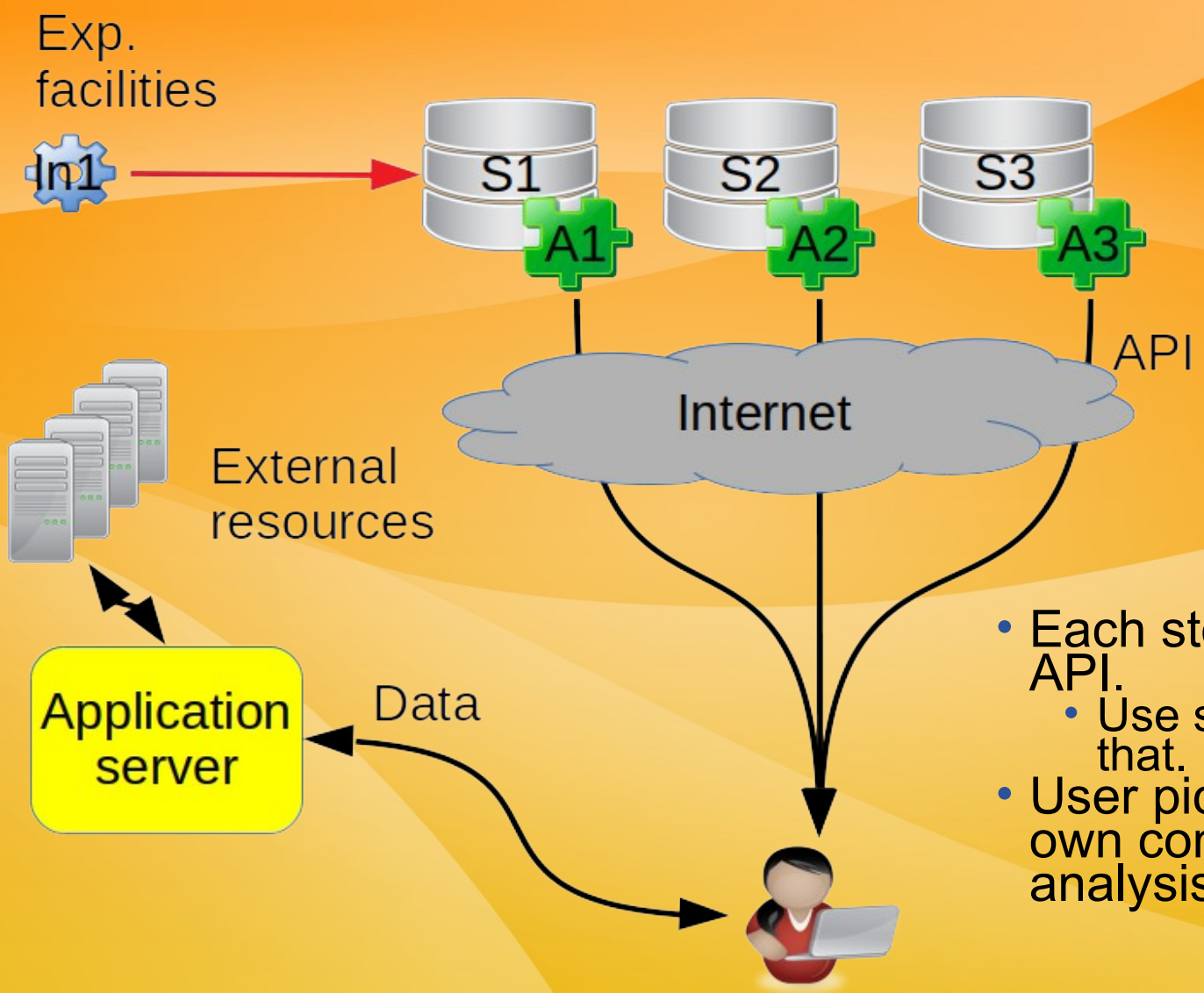
- Modern installations generate terabytes and petabytes of data. The collaboration brings together hundreds and thousands of researchers from many organizations. Thus, it is necessary to organize access for scientists who use these repositories to perform data analysis.
- Most collaboration store data as a collection of files.
  - Special case: DB-oriented storage.
- They have a long history and established practice of working with data.
  - No change to existing site infrastructure, only add-ons
- Open Science is a modern trend in the physics.

The main ideas that are embedded in the architecture

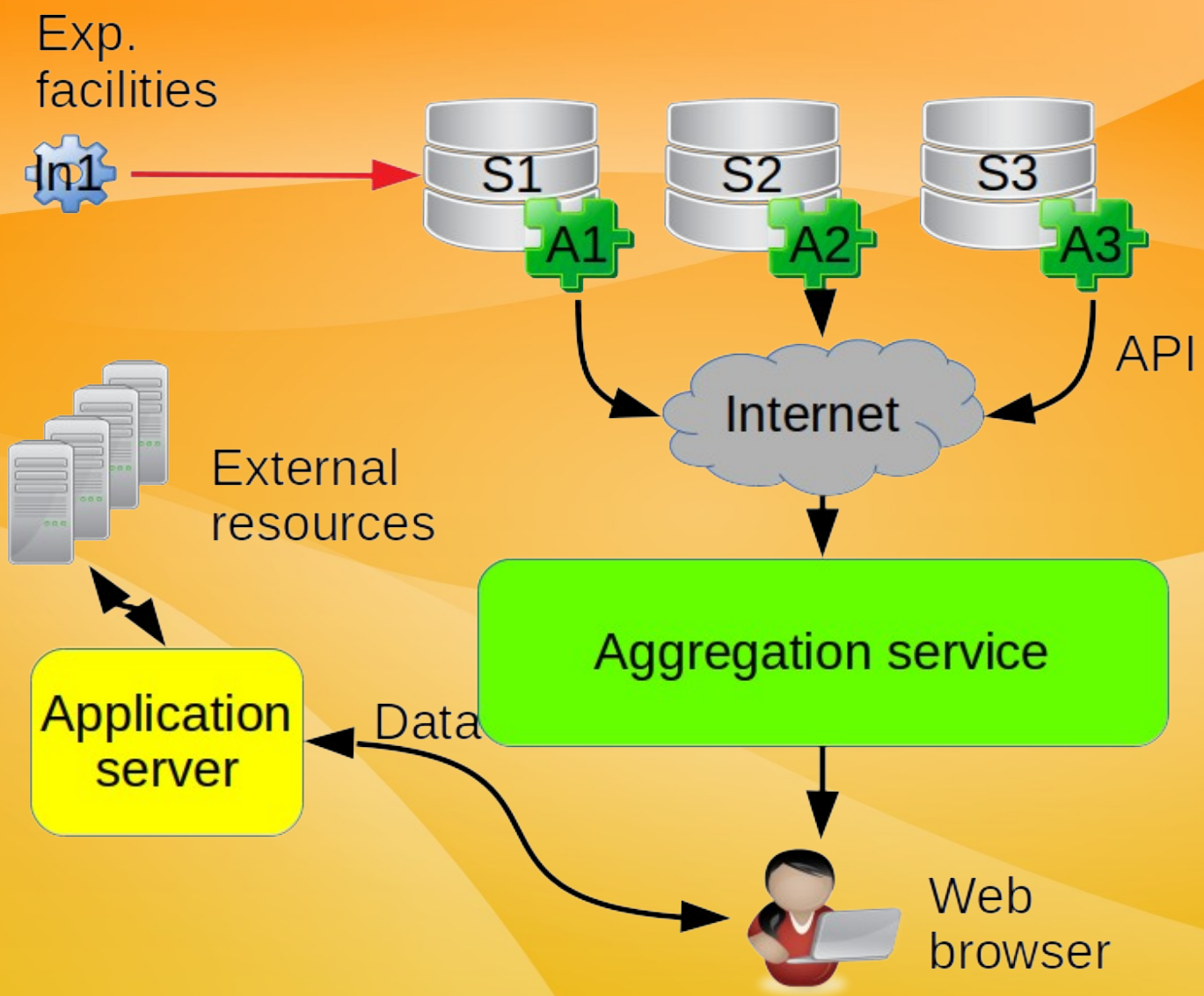
- Read-only oriented distributed storage (DS).
  - Remote access to data as local file systems
  - On-demand data transfer by requests only
- No intervention into local storage, special adapters are used to access data.
- User requests are processed on a dedicated server based on metadata only.
- Metadata is extracted from primary and/or secondary data in semi-automatic mode.
  - Binary format description language is used for serialize/deserialize binary data.



- Each storage has individual API
- User pick up data to the own computer and make analysis



- Each storage has unified API.
  - Use special adapters for that.
- User pick up data to the own computer and make analysis


Exp.
facilities
In1
S1
S2
S3
A1
A2
A3
API
Internet
External
resources
Aggregation service
Application
server
Data
Web
browser

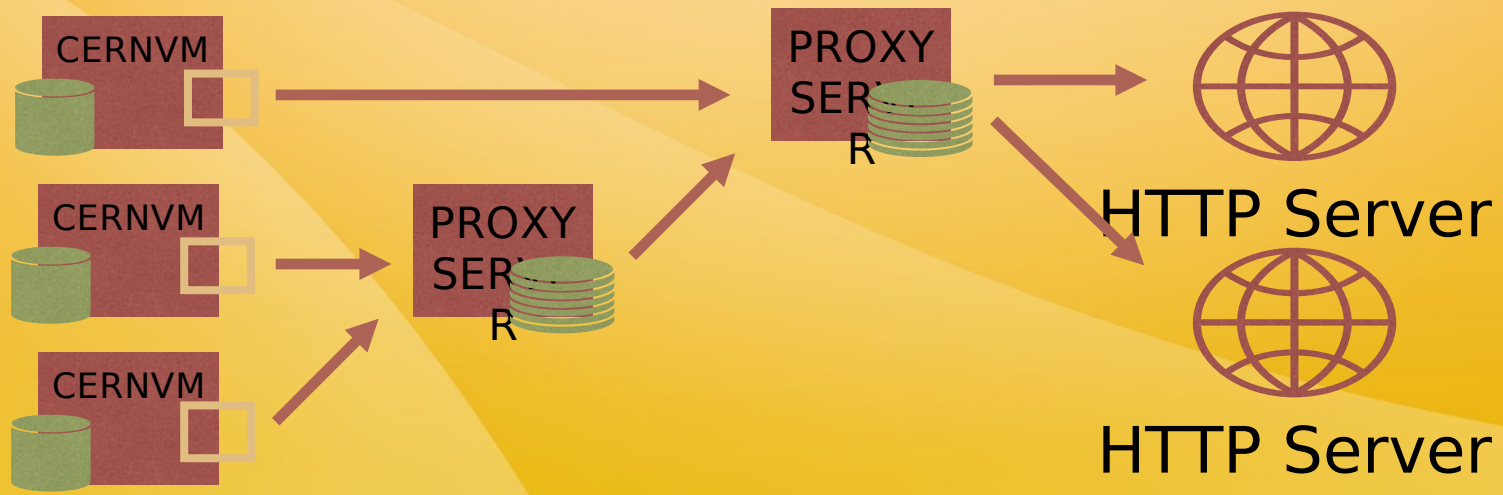
CERN VM-FS as an adapter

- Data are left untouched in their own file system
- CernVM-FS indexes the data and changes, stores only the metadata (indices, checksums, locations, etc.) and data tree
- CernVM-FS uses HTTP as the data transfer protocol, so there's no firewall problem
- Data transfer starts only on actual reads
- Multilevel cache-proxy servers

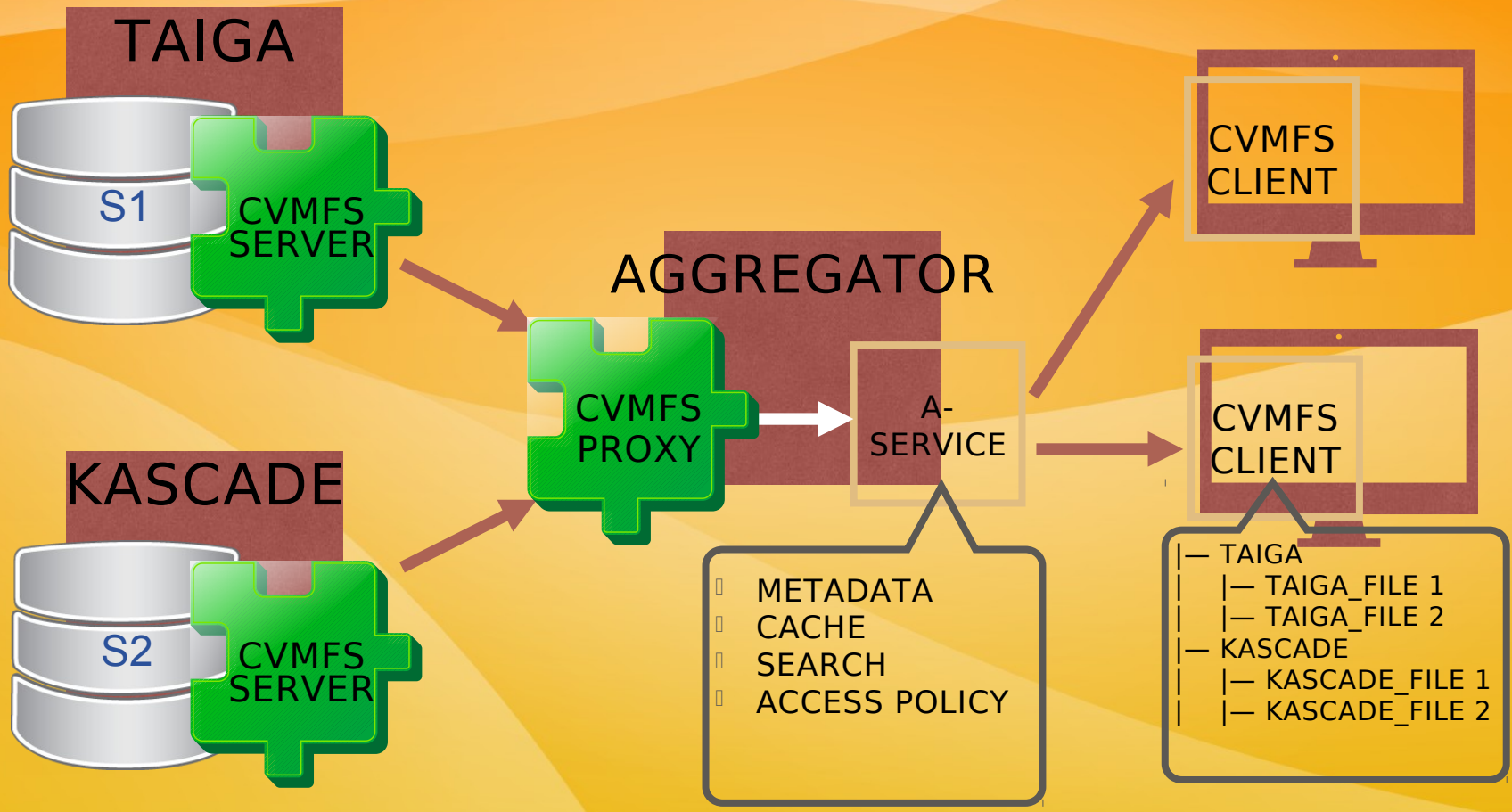
CERNVM-FS

- DATA UPDATE

- DATA DISTRIBUTION

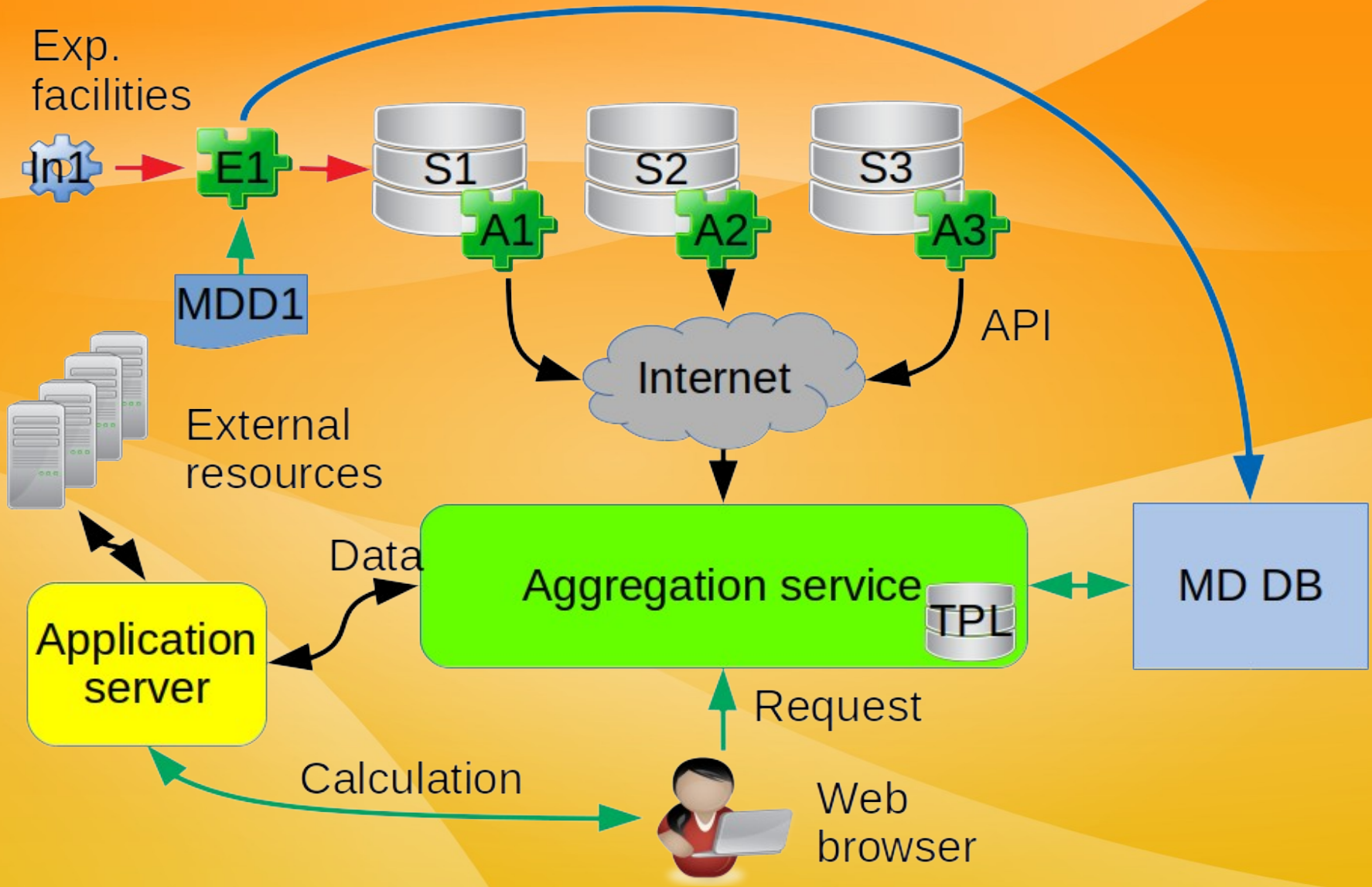


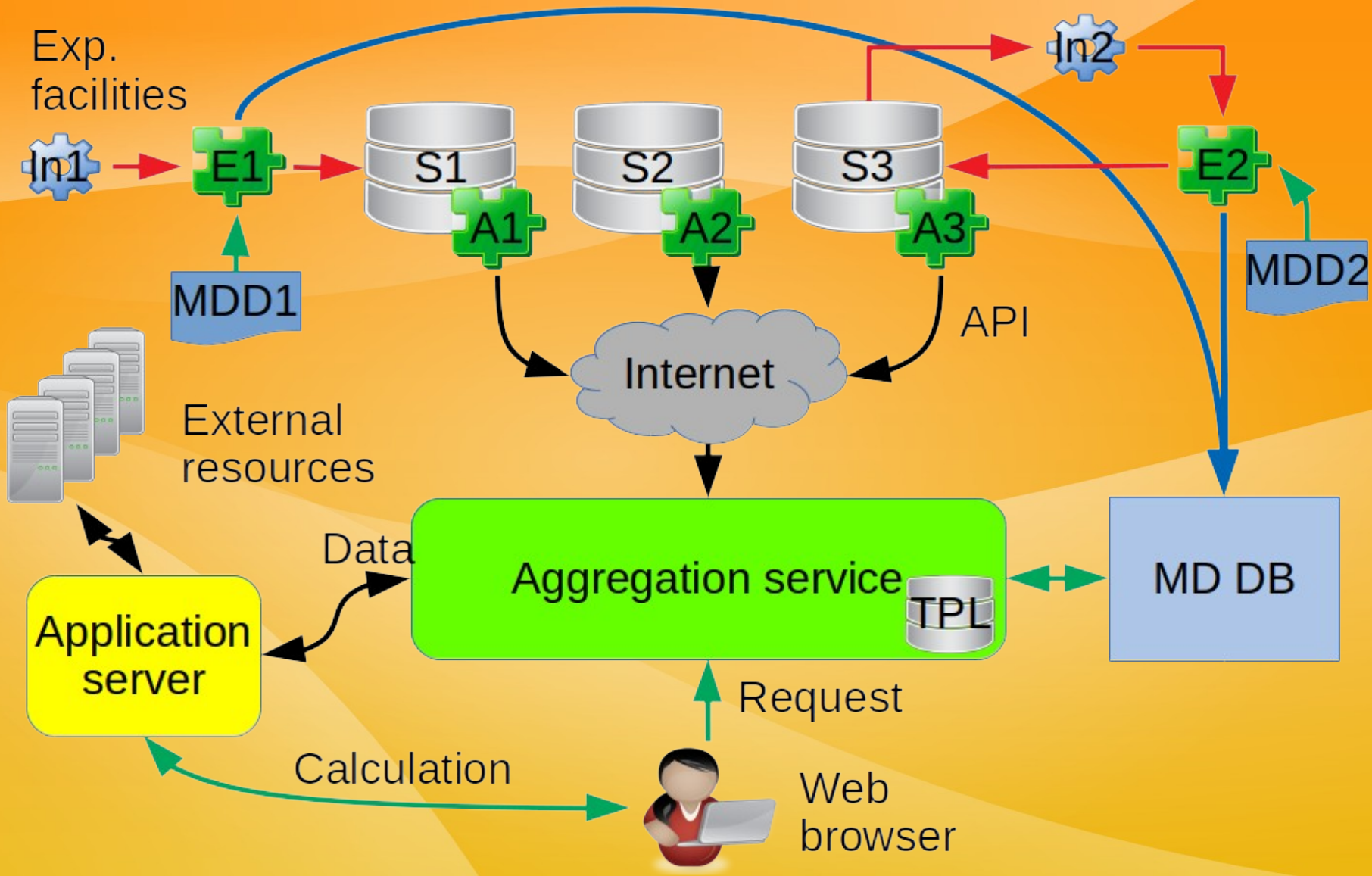
CERNVM-FS


TAIGA
S1
CVMFS SERVER
KASCADE
S2
CVMFS SERVER
AGGREGATOR
CVMFS PROXY
A-SERVICE
METADATA
CACHE
SEARCH
ACCESS POLICY
CVMFS CLIENT
CVMFS CLIENT
|— TAIGA
| |— TAIGA_FILE 1
| |— TAIGA_FILE 2
|— KASCADE
| |— KASCADE_FILE 1
| |— KASCADE_FILE 2

Information system

- Any user request is processed on MDDB service.
  - No direct request to the local storage.
- We use special programs – called extractors to extract metadata.
- Two level metadata:
  - File level MD (experiment, detector, date, session,…)
  - Event level MD (energy, type of primary particle, …)
    - This MD is usually the result of raw data processing.


Exp.
facilities
In1
E1
S1
S2
S3
A1
A2
A3
MDD1
API
Internet
External
resources
Data
Aggregation service
TPL
MD DB
Application
server
Request
Calculation
Web
browser


Exp.
facilities
In1
E1
S1
S2
S3
A1
A2
A3
In2
E2
MDD1
MDD2
API
Internet
External
resources
Data
Application
server
Aggregation service
TPL
MD DB
Request
Calculation
Web
browser

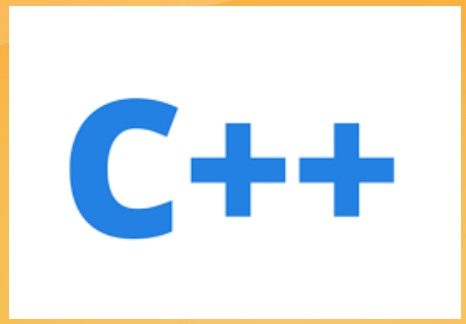
Metadata extractors

- Extract meta data from primary and/or secondary data
- The list of extracted MD should cover all allowed user requests.
- The answer of MDDB is a list of URL(URI) where necessary data locate.

Kaitai Struct

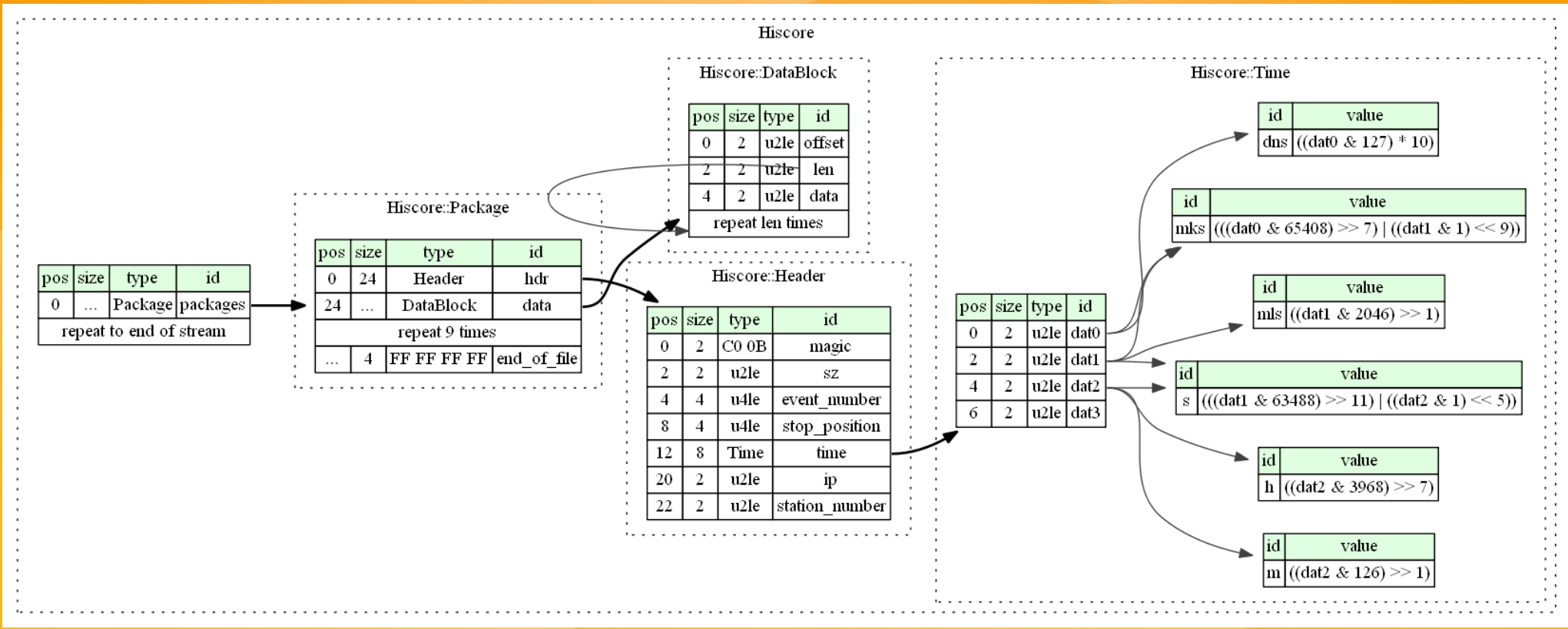
- Declarative language for describing binary formats.
- Allows:
  - Describe the format in YAML
  - Check using visualization tools (ksv)
  - Compile to library in target language (ksc)
  - Use received API
- License
  - Compiler - GPLv3 +
  - Library for reading files - MIT or Apache v2

Kaitai Target languages



Etc

MD description. HiScore example



HiSCORE format specification expressed in Kaitai Struct

Part I

```
meta:
  id: hiscore
  title: HiSCORE data
  license: Unlicensed
seq:
  - id: packages
    type: package
    repeat: eos
types:
  package:
    seq:
    - id: hdr
      type: header
    - id: data
      type: data_block
      repeat: expr
      repeat-expr: 9
    - id: end_of_package
      contents:
      [0xFF, 0xFF, 0xFF, 0xFF]
```

Part II

```
header:
  seq:
  - id: magic
    contents: [0xC0, 0x0B]
  - id: sz
    type: u2le
  - id: event_number
    type: u4le
  - id: reserved
    type: u4le
  - id: time
    type: time
  - id: ip
    type: u2le
  - id: station_number
      type: u2le
data_block:
  seq:
  - id: offset
  type: u2le
  - id: len
    type: u2le
  - id: data
    type: u2le
    repeat: expr
    repeat-expr: len
```

Part III

```
time:
  seq:
  - id: dat0
    type: u2le
  - id: dat1
    type: u2le
  - id: dat2
    type: u2le
  - id: dat3
    type: u2le
  instances:
    dns:
    value: '(dat0 & 0x7f) * 10'
    mks:
    value: '((dat0 & 0xff80) >> 7) | (dat1 & 1) << 9'
    mls:
    value: '(dat1 & 0x7fe) >> 1'
    s:
    value: '((dat1 & 0xf800) >> 11) | ((dat2 & 1) << 5)'
    m:
    value: '(dat2 & 0x7e) >> 1'
    h:
    value: '(dat2 & 0xf80) >> 7'
```

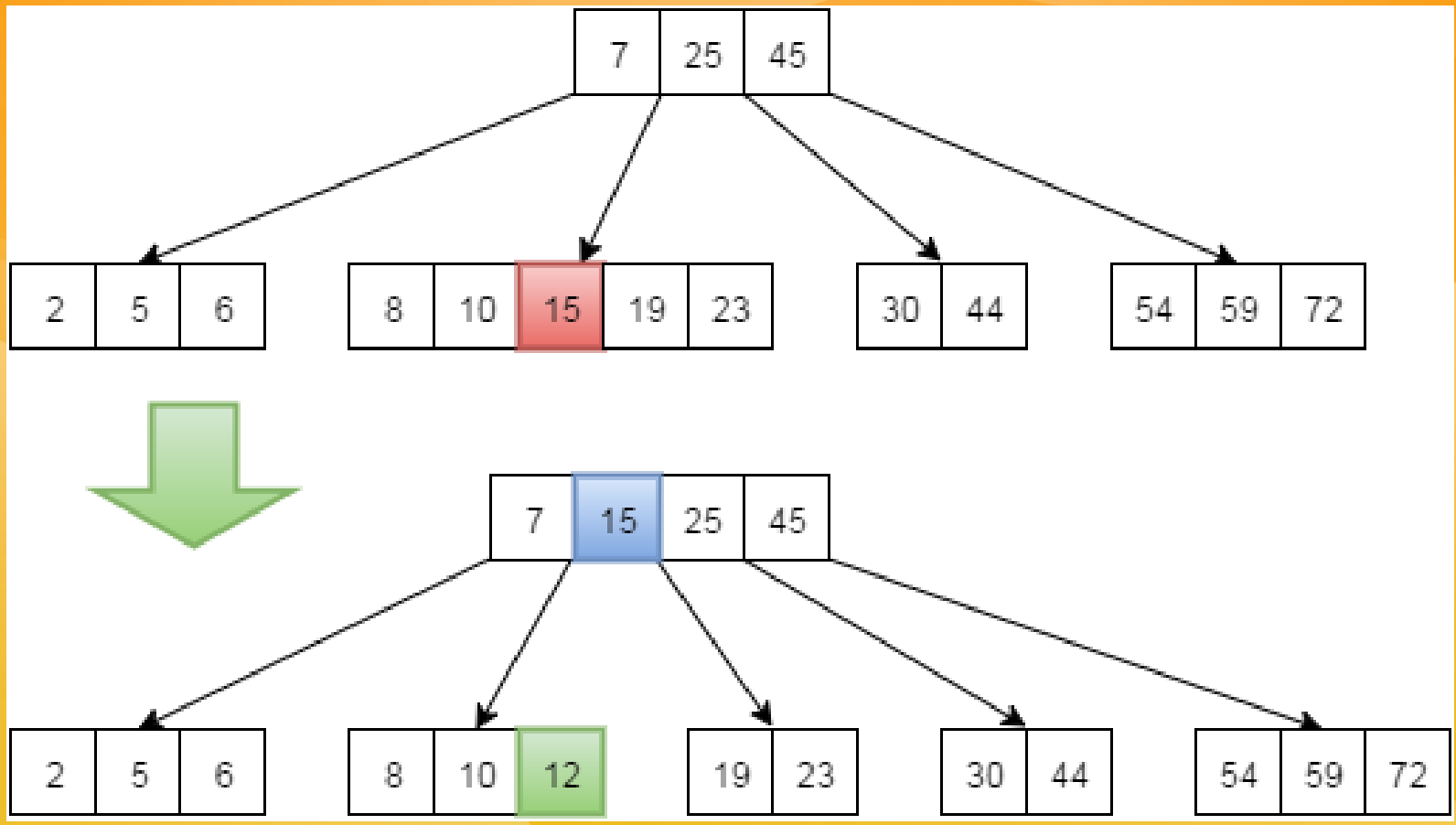
HiScore Kaitai auto generates program

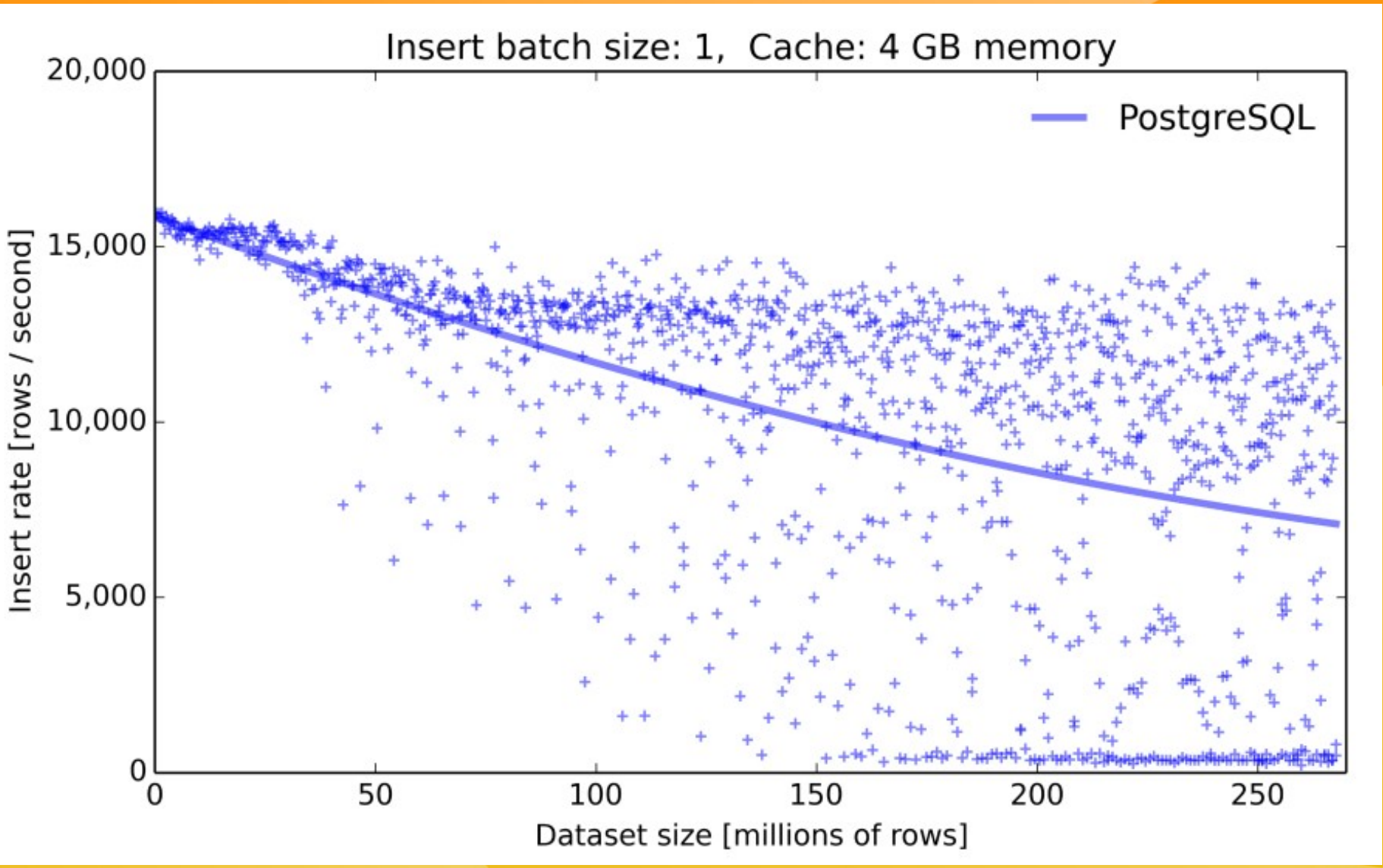
```
public static void main(String[] args) throws IOException {
        Hiscore hiscore = Hiscore.fromFile(FILE_NAME);
        int pckgNumber = 0;
        for (Hiscore.Package pckg : hiscore.packages()) {
                System.out.println("Package: " + pckgNumber);
                System.out.println(String.format("Magic: %02X %02X", pckg.hdr().magic()[0], pckg.hdr().magic()[1]));
                System.out.println("Size: " + pckg.hdr().sz());
                System.out.println("Event number: " + pckg.hdr().eventNumber());
                System.out.println("Stop position: " + pckg.hdr().stopPosition());
                System.out.println(String.format("H %d: M %d: S %d: DNS %d: MKS %d: MLS %d",
                        pckg.hdr().time().h(), pckg.hdr().time().m(),
                        pckg.hdr().time().s(), pckg.hdr().time().dns(),
                        pckg.hdr().time().mks(), pckg.hdr().time().mls()));
                System.out.println("IP: " + pckg.hdr().ip());
                System.out.println("Station number: " + pckg.hdr().stationNumber());
                int size = pckg.data().size();
                int dataBlockNumber = 0;
                for (Hiscore.DataBlock data: pckg.data()) {
                        System.out.println("Data block: " + dataBlockNumber);
                        System.out.println("Offset: " + data.offset());
                        System.out.println("Length: " + data.len());
                        int n = data.len() > DATA_COUNT_TO_VIEW ? DATA_COUNT_TO_VIEW : size;
                        System.out.print("[");
                        for (int i = 0; i < n; i++) {
                                System.out.print(data.data().get(i) + ", ");
                        }
                        System.out.print("...]");
                        System.out.println();
                        ++dataBlockNumber;
                }
                ++pckgNumber;
}
```

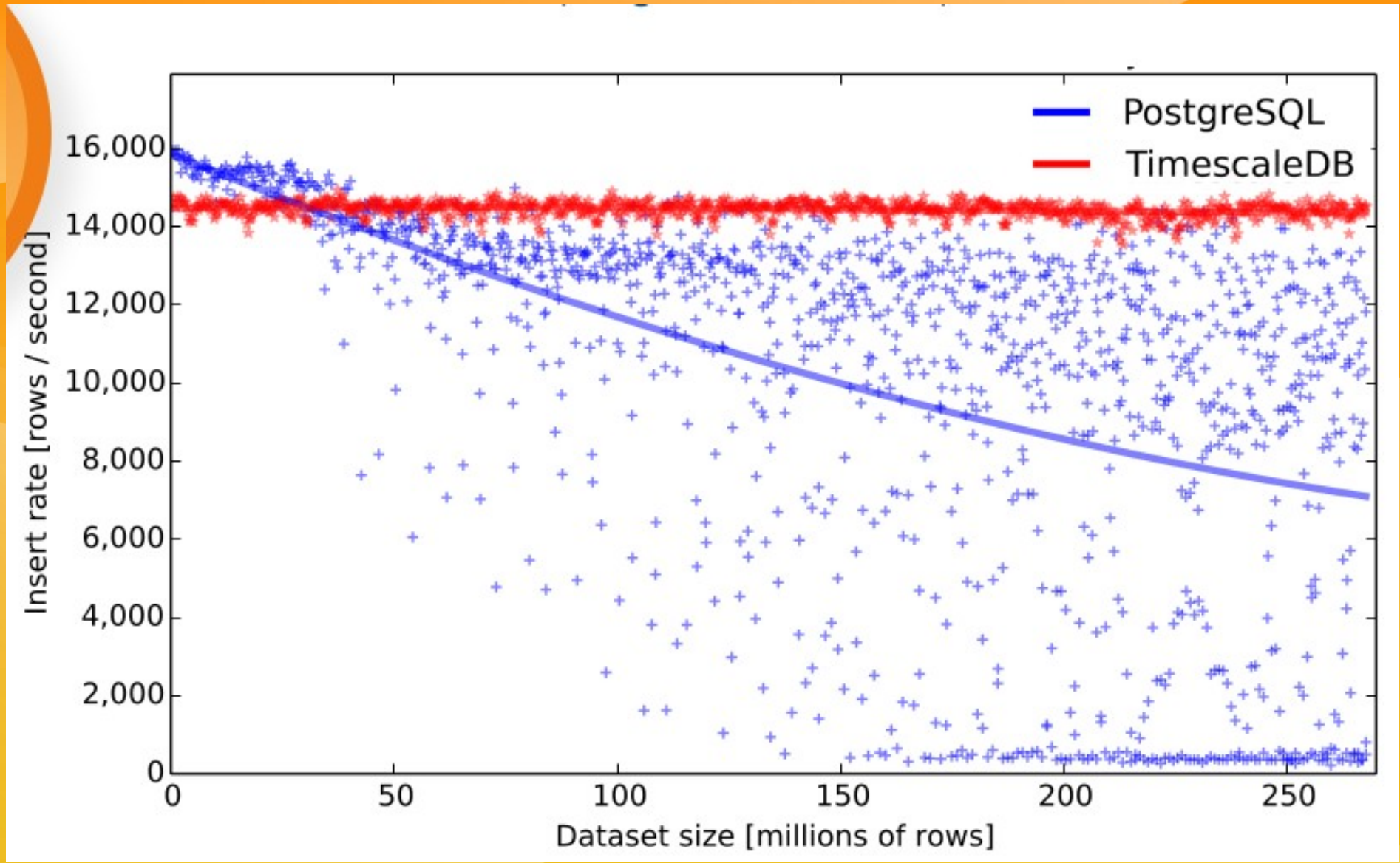
DB for information system

- Relation DB (MySQL, PostgreSQL, …)
  - Degradation of speed.
  - Reshuffle of indexes under massive insert operations.
- Time series DB (TimeScale DB)

Insertion of element into B-tree




Insert batch size: 1, Cache: 4 GB memory
PostgreSQL
Insert rate [rows / second]
20,000
15,000
10,000
5,000
0
0
50
100
150
200
250
Dataset size [millions of rows]


PostgreSQL
TimescaleDB
Insert rate [rows / second]
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
0
50
100
150
200
250
Dataset size [millions of rows]

File level request processing

- MDDB returns a list of URLs where the required data is located.
- The aggregation service re-exports to the user only those files that are in the list, obtained from MDDB.

Event level request processing

- MDDB returns a list of URLs where the required data is located.
- The aggregation service scans these files and extracts those events that satisfy the user's request.
- As a result, a new collection is formed, which contains only the necessary events.
- This collection is exported to the user.

Conclusion for part 1

- Modern information technologies can provide scientists with convenient access to large data distributed throughout the world.
- Distributed storage provides analysis of instant messengers and intelligent management of data access rights.
- A custom data request can contain both file level filters and more detailed filters, such as events.

Any questions for part 1?

Part 2

Distributed Data Storages and Provenance Metadata

Status: R&D, proof-of-concept

A.Demichev, A.Kryukov & N.Prikhodko "The Approach to Managing Provenance Metadata and Data Access Rights in Distributed Storage Using the Hyperledger Blockchain Platform", Proceedings of Ivannikov ISPRAS Open Conference, 2019, IEEE Xplore (to be pulished)

Provenance, Metadata & Provenance Metadata (PMD) (1/2)

W3C: https://www.w3.org/2005/Incubator/prov/wiki/What_Is_Provenance

- Provenance of a resource/asset is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource
  - from the French *provenir* = 'to come from/forth'
- Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility
  - provenance is often represented as metadata, but not all metadata is necessarily provenance (e.g., descriptive MD)

Provenance Metadata (PMD) (2/2)

- Metadata describing data, provide context and are vital for accurate interpretation and use of data
- One of the most important types of metadata is provenance metadata (PMD)
  - tracking the stages at which data were obtained
  - ensuring their correct storage, reproduction and interpreting
  - ⇒ ensures the correctness of scientific results obtained on the basis of data
- The need for PMD is especially essential when large volume (big) data are jointly processed by several research teams

PMD MS Construction: Centralized (1/2)

- mainly used for workflow management systems and databases
- inadequate
  - for the use in distributed environments that includes different administrative domains
  - the possibility of using metadata by organizationally unrelated or loosely related research communities
- any central service is a bottleneck and a point of failure of the system
- Examples: VisTrails, ZOOM, PASS, VDS,...

PMD MS in Distributed Environment

- the basis of the environment is a set of storages
  - clouds, file servers, tape storage, etc.
  - each is managed by its data management system (DMS)
- on the other hand, a team of researchers is formed, perhaps administratively unrelated to each other for the joint implementation of a project
  - virtual organization (VO)
- it is assumed that the implementation of such a project requires the use of distributed data storage.

PMD MS: Operational Requirements in the Case of Distributed Data Storages (DDS) (1/3)

- the ability to track the complete and detailed history of data states in DDS of data processing
  - by recording the provenance metadata about each transaction carried out with the data in a distributed registry without a centralized service;
- resistance to erroneous or deliberate changes in metadata records of provenance "after the fact"
  - all transaction records must be permanently and unchanged stored in the system
- scalability

PMD MS: Operational Requirements in the Case of Distributed Data Storages (DDS) (2/3)

- the ability to retrieve PMD at the request of users;
- the distributed nature of the system, the absence of bottlenecks and points of failure;
- absence of the need to attract significant computer resources to maintain the system's operation and security

PMD MS: Operational Requirements in the Case of Distributed Data Storages (DDS) (3/3)

- consensus on the procedure for data operations is needed to resolve possible conflicts related to the use of data between the VO/project participants
- conflicts can be related to priority issues when obtaining the results of data processing, violation of access rights, financial issues, etc.
  - consensus on the order of data transactions

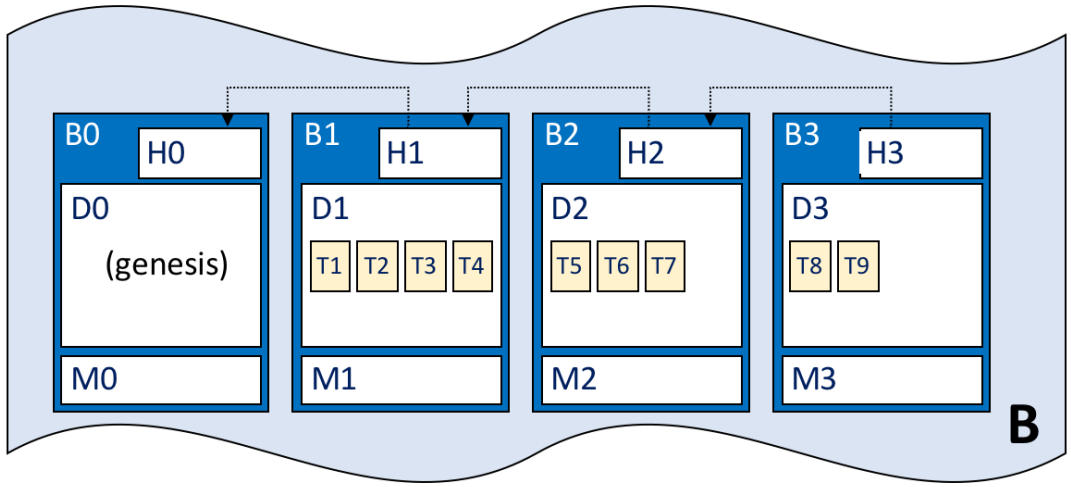
PMD MS Construction: Distributed Solution (1/2)

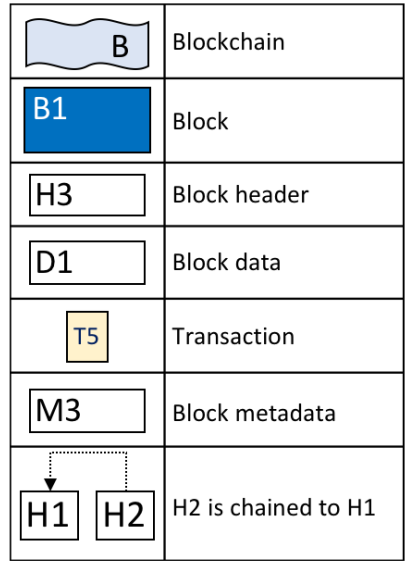
- distributed environment ⇒ distributed registry for PMD
- we suggested to use the blockchain technology which provides
  - that no records were inserted into the registry in hindsight
  - no entries were changed in the registry
  - the registry has never been damaged or branched
  - monitoring and restoring the complete history of data processing and analysis

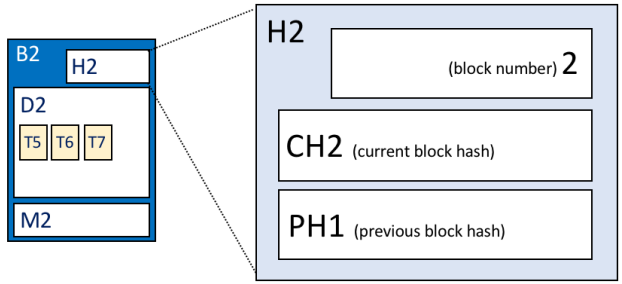
PMD MS Construction: Distributed Solution (2/2)

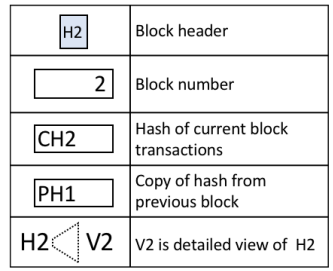
- immutable history of data in distributed environment
  - Important for investigation and resolving possible conflicts between project participants, as well as owners of the storages
- large experiments and/or DDS may involve various parties with their own interests ⇒ necessity for a blockchain

Blockchain structure









PMD MS Construction: Which Blockchain (1/2)

- type of the blockchains
  - permissionless blockchains, in which there are no restrictions on the transaction handlers
  - permissioned blockchains, in which transaction processing is performed by specified entities
- permissionless:
  - algorithms are based on
    - Proof-of-Work – highly resource-consuming, probability of reaching a consensus, which grows with time elapsing, ...
    - Proof-of-Stake – Nothing-at-Stake problem,…
  - suitable for open (public) networks of participants (Bitcoin, etc.)

PMD MS Construction: Which Blockchain (2/2)

- Permissioned:
  - there is a fixed number of trusted transaction/block handlers
    - from different administrative domains
  - the handlers must come to a **consensus** about the content and the order of the recorded transactions
    - distributed consensus algorithm should be involved
  - form a more controlled and predictable environment than permissionless blockchains
  - suitable for networks with naturally existing trusted parties
    - our case: DMS, data owners,...

PMD Projects Based on Permissionless Blockchains: ProvChain (1/3)

- intended for a cloud storage (no DDS, no different administrative domains)
  - real-time monitoring of user operations and MDP collection to support access control policies and the detection of malicious intrusions;
- permissionless ⇒ artificial solutions, e.g.:
  - consensus process for the blockchain creation is carried out on the basis of Proof-of-Stake algorithm (control of stakes by a special service)
  - by dedicated trusted virtual machines housed in a federated cloud computing environment
    - no real consensus among the potentially conflicting parties

PMD Projects Based on Permissionless Blockchains: SmartProvenance/DataProv (2/3)

- also intended for a cloud storage (no DDS, no different administrative domains)
- implemented on top of the Ethereum Blockchain platform
- permissionless ⇒ artificial solutions + resource intensive (=), e.g.:
  - block mining
  - cryptocurrency (ether)
  - two subsystems (on-chain module & of-chain module) loosely coupled
    - Consensus about content of transactions is achieved by of-chain module

PMD Projects Based on Permissionless Blockchains: Storj & Sia (3/3)

- intended for a P2P network of public storage resources
- public (permissionless) blockchain
  - mainly for providing mutual settlements between suppliers and consumers of (P2P) resources
  - special cryptocurrencies (SJCX for Storj & Siacoins)
- private data on public resources ⇒ main efforts to provide:
  - privacy of data
  - data integrity
  - fault tolerance
- very restricted PMD facilities

DDS State = System state (1/2)

- The state of the entire distributed storage = aggregated state of the set of files stored in it with their states at the moment
- The state of a data file is determined by PMD:
  - global ID + attributes, including:
    - local file name in a storage: fileName;
    - storage identifier: storageID;
    - creator identifier: creatorID;
    - owner identifier: ownerID
    - type: type=primary/secondary/replica
    - …

DDS State = System state (2/2)

- source: source={expFacility,expFacilityID}/{{file,fileID}, {tool,toolName}}
- date/time of creation: dateTime={yyyy.mm.dd;hh.mm.ss}
- number of file downloads: downloads=integer
- users who downloded the file: dUsers={{userID,txID}, {userID,txID},…,{userID,txID}}
- URI of file description: metadataURI

- The set of values for all file attributes = its current state

Basic operations ⇒ transactions

- new file upload
- file download
- file deletion
- file copy
- copying a file to another repository
- transferring a file to another repository
  - each active transaction ⇒ update of some state attributes
    - for example, after the transaction "file download" the values of the keys change: "number of file downloads" and "users who downloaded the file".

Management of access rights

- additional task: providing distributed management of access rights to data is set.
  - Example 1: the owner of a data file (the user who created the data, the organization to which it belongs) must be able to manage its access rights for other users.
  - Example 2: a cloud storage service grants access to data stored on it only to users from organizations that have paid for storage services.

HyperLedger Fabric (1/4)

- Analysis of existing platforms shows that the formulated problems most naturally can be solved on the basis of the
  - Hyperledger Fabric blockchain platform (HLF; www.hyperledger.org)
  - together with Hyperledger Composer (HLC; hyperledger.github.io/composer) = set of tools for simplified use of blockchains
- permissioned blockchains
  - transactions are processed by a certain list of trusted network members

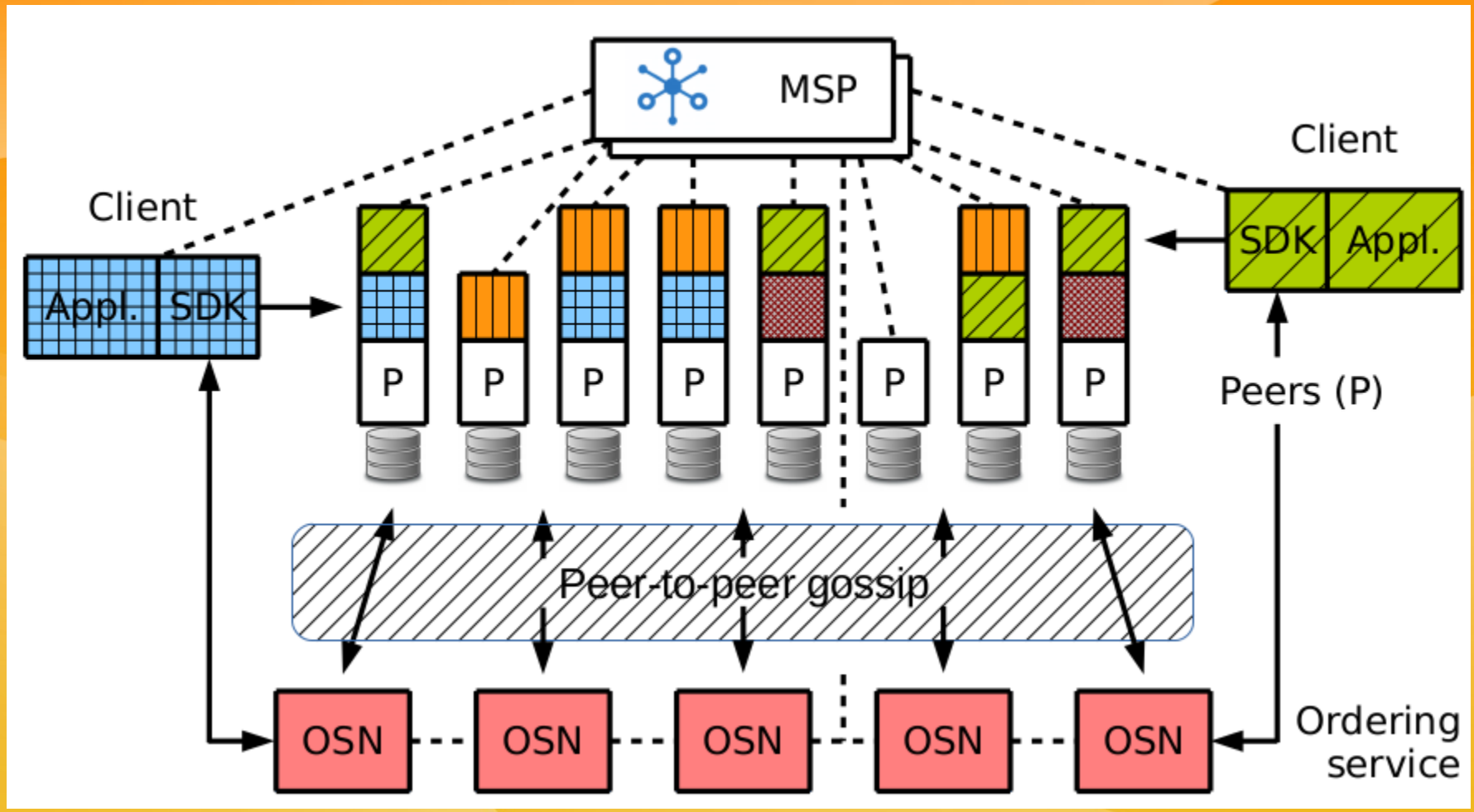
Blockchain platforms

Название	Используемый алгоритм консенсуса	Скорость обработки транзакций	Под-держка умных контрактов	Виртуальная машина	Шиф-рова-ние дан-ных	Активность разработчиков (GitHub)	Уровень использования в прикладных проектах	Компания-производитель
Multichain	Round robin (diversity)	100-1000/s	No	No	No	Medium	Medium	Coin Sciences
Quorum	Time and vote based	12-100/s	Yes	EVM	Yes	Medium	High	J.P. Morgan
Hyperledger Fabric	PBFT	10k-100k/s	Yes	Chaincode	Yes	High	High	IBM
OpenChain	Partitioned	Thousands/s	No	Yes	Yes	Low	Medium	Coinprism
Chain Core	Federated consensus	N/A	No	Yes	Yes	High	High	Chain
Corda	BFT, etc.	N/A	Yes	JVM	Yes	High	Medium	R3
Monax	Tendermint	10k/s	Yes	EVM	No	Medium	High	Monax

HyperLedger Fabric(2/4)

- operation of smart contracts (chaincode) for the business process of sharing storage resources
- advanced means of managing access rights\
  - access rights can be managed by network members within their competence
- provides a record of transactions & advanced query tools
- thanks to its modular structure, it allows using different algorithms to reach consensus between business process participants
- has a developed built-in security system based on PKI

HyperLedger Fabric (3/4)



From: *E. Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in Proc 13th EuroSys Conf. 2018*

HyperLedger Fabric (4/4)

- Unlike public blockchain networks, which allow non-authenticated users to participate in their work, members of the (HLF&C)-network must be registered with Membership Service Provider (MSP),
  - among other things, performs the functions of Certification Authority (CA).

Business process within (HLF&C)-platform

- **Assets** are tangible or intellectual resources, records of which are kept in registers
  - in our case, the assets are data files; their properties (attributes) are provenance metadata
- **Participants** are members of the business network.
  - they can own assets and make transaction requests
  - can have any properties if necessary
- **Transaction** is the mechanism of interaction of participants with assets
- **Events:** messages can be sent by transaction processors to inform external components of changes in the blockchain

HyperLedger Fabric → ProvHL (1/3)

- ProvHL = Provenance HyperLedger
  - status: Proof of concept
- operation of smart contracts (chaincodes)
  - sophisticated adaptation of HLF for the business process of sharing storage resources
- provides a record of transactions & advanced query tools
- advanced means for managing access rights
  - access rights can be managed by network members within their competence

HyperLedger Fabric → ProvHL (2/3)

- Participants
  - Person
  - StorageProvider
- Assets
  - File
  - Storage
  - Operation
  - Group
- Transactions
  - FileAccessGrant
  - FileAccessRevoke
  - OperationUploadCreate
  - OperationUploadSetState
  - ...

HyperLedger Fabric → ProvHL (3/3)

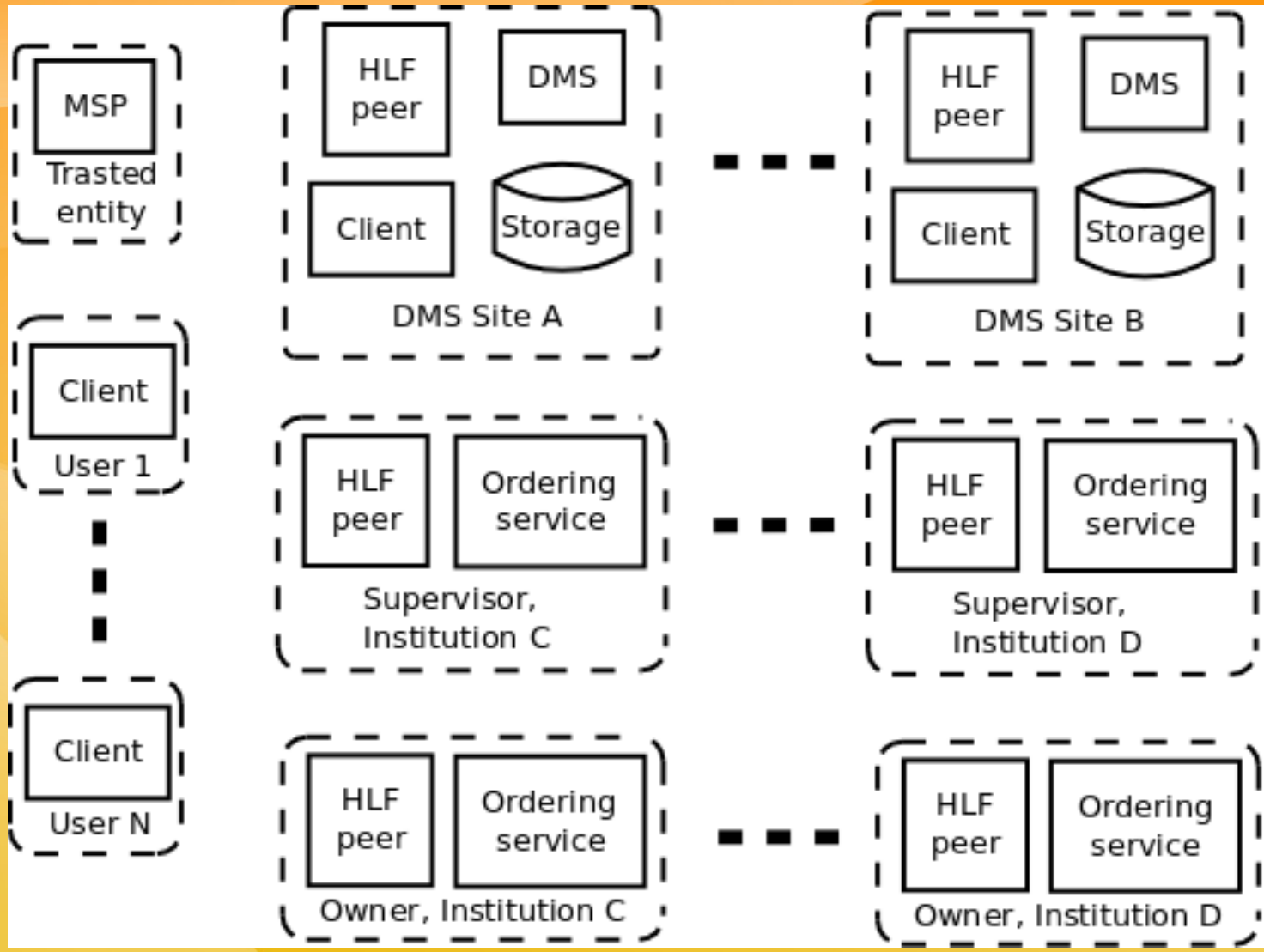
- thanks to its modular structure, it allows using different algorithms to reach consensus between business process participants
- has a developed built-in security system based on PKI

PMD driven data management

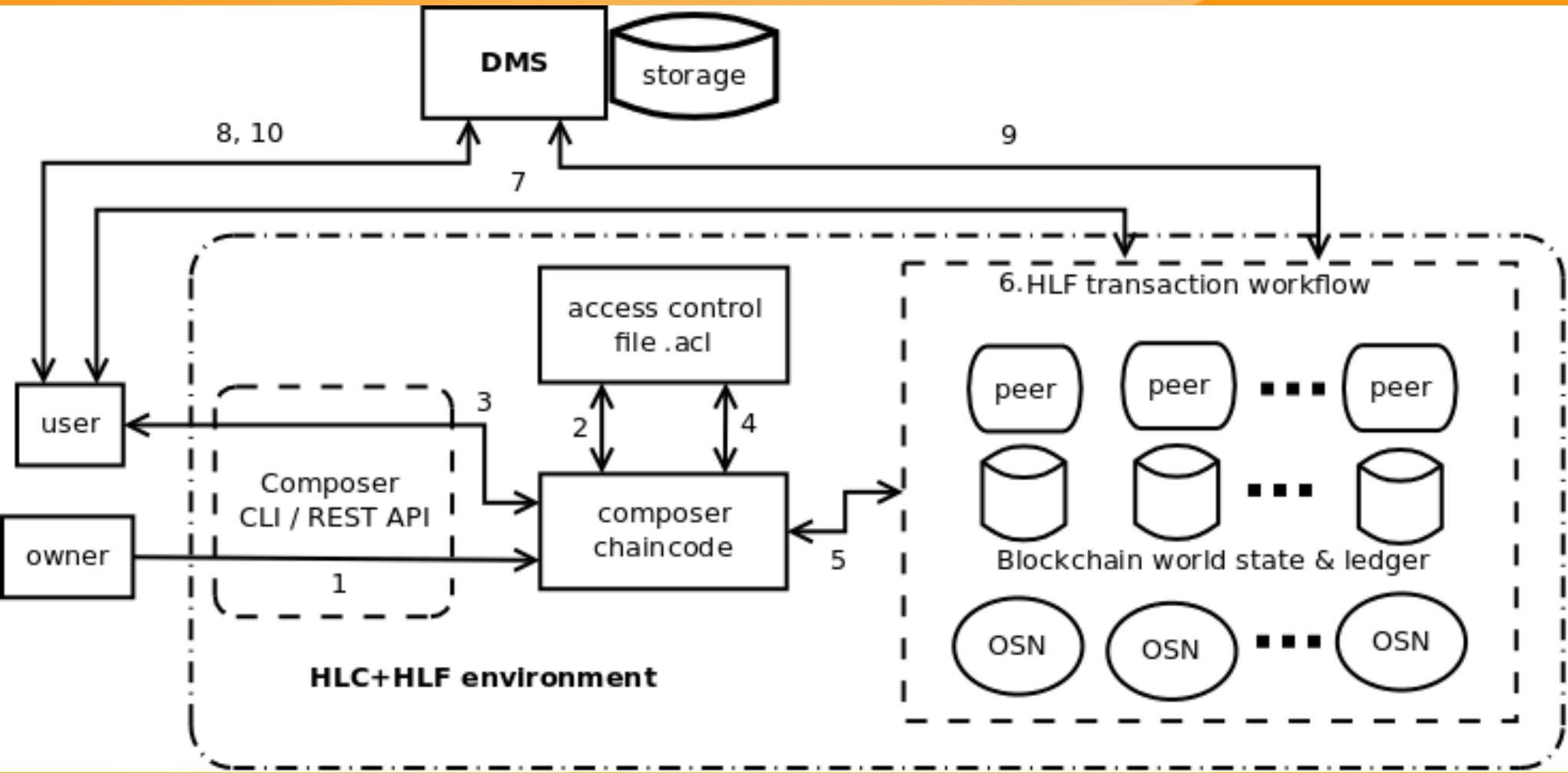
- two approaches are possible
  - data management systems (DMS) manage data and use a blockchain simply as a distributed log
    - data driven data management
  - metadata is written to the blockchain beforehand, and DMSs refer to the blockchain and performs the transactions recorded there
    - metadata driven data management
- ProvHL implements the second approach

Basic principles of the ProvHL design (1/2)

Distribution of the main HLF software modules by the administrative domains of the distributed storage environment



ProvHL operation (1/3)



Simplified scheme for recording transactions with provenance metadata and managing data access rights based on HLF&C

ProvHL operation (2/3)

- Operations with files comprise of two types of transactions recorded in the blockchain:
  - first corresponds to client requests,
  - second corresponds to server responses
- Operation states
  - STARTED
  - PENDING
  - COMPLETED
  - ERROR

ProvHL operation (3/3)

- Example - "new file upload" transaction:
  - a new asset — a data file — with the "temporary" label is first recorded in the blockchain
  - only after the actual upload of the file in the storage, DMS initiates a transaction removing the label "temporary" and turns the uploaded file into a fully valid asset.
- Together with the splitting of transactions into the client and server parts ⇒ level of correspondence (history recorded in blockchain) ⇔ (real history of the data in the distributed storage) practically acceptable.

Problems: Oracle problem

- As is known:
- ensuring full compliance of the real world history with the history recorded in a blockchain is outside the scope of the blockchain technology
- the so-called “Oracle Problem”,
  - see e.g., https://medium.com/@DelphiSystems/the-oracle-problem-856ccbdbd14f

Rights Delegation in ProvHL (1/2)

- the definition of operations as the assets makes the mechanism of the delegation very natural and flexible
  - contains the obligatory attributes “requester” and “executor”
  - inherits “file owner” attributes from the file asset definition
- DMS_Storage1 initiates, on behalf of the User, the operation of uploading the required file to destination Storage2

Rights Delegation in ProvHL (2/2)

- For this aim DMS_Storage1 interacts with the chaincode which, among other actions, defines that
  - while for the initial copy operation
    - requester attribute is equal to the User
    - executor is DMS_Storage1
  - for the induced upload operation
    - requester is DMS_Storage1
    - executor is DMS_Storage2
- the owner of the file copy on the Storage2 is the same as the owner of source file on the Storage1

Consensus Algorithms

- data management via PMD requires a method of ensuring consensus among participants in the business process about the content and order of transactions with data
- thanks to its modular structure, HLF/ProvHL allows using various algorithms to reach consensus between business process participants
- there exits a number of consensus algorithms that do not require resource-consuming and slow “proof-of-work” mechanism which is intrinsic for cryptocurrency blockchain networks

Consensus Algorithms: a choice

- The choice of the consensus algorithm depends on the operational conditions:
  - Testbed level ⇒ Solo (single OSN)
  - Production system in friendly environment ⇒ Kafka
    - crash tolerant
  - Production system in competitive environment with possible attempts at malicious actions ⇒ BFT
    - Byzantine fault tolerant
      - e.g., PBFT/BFT-SmaRt, Hashgrph, ...

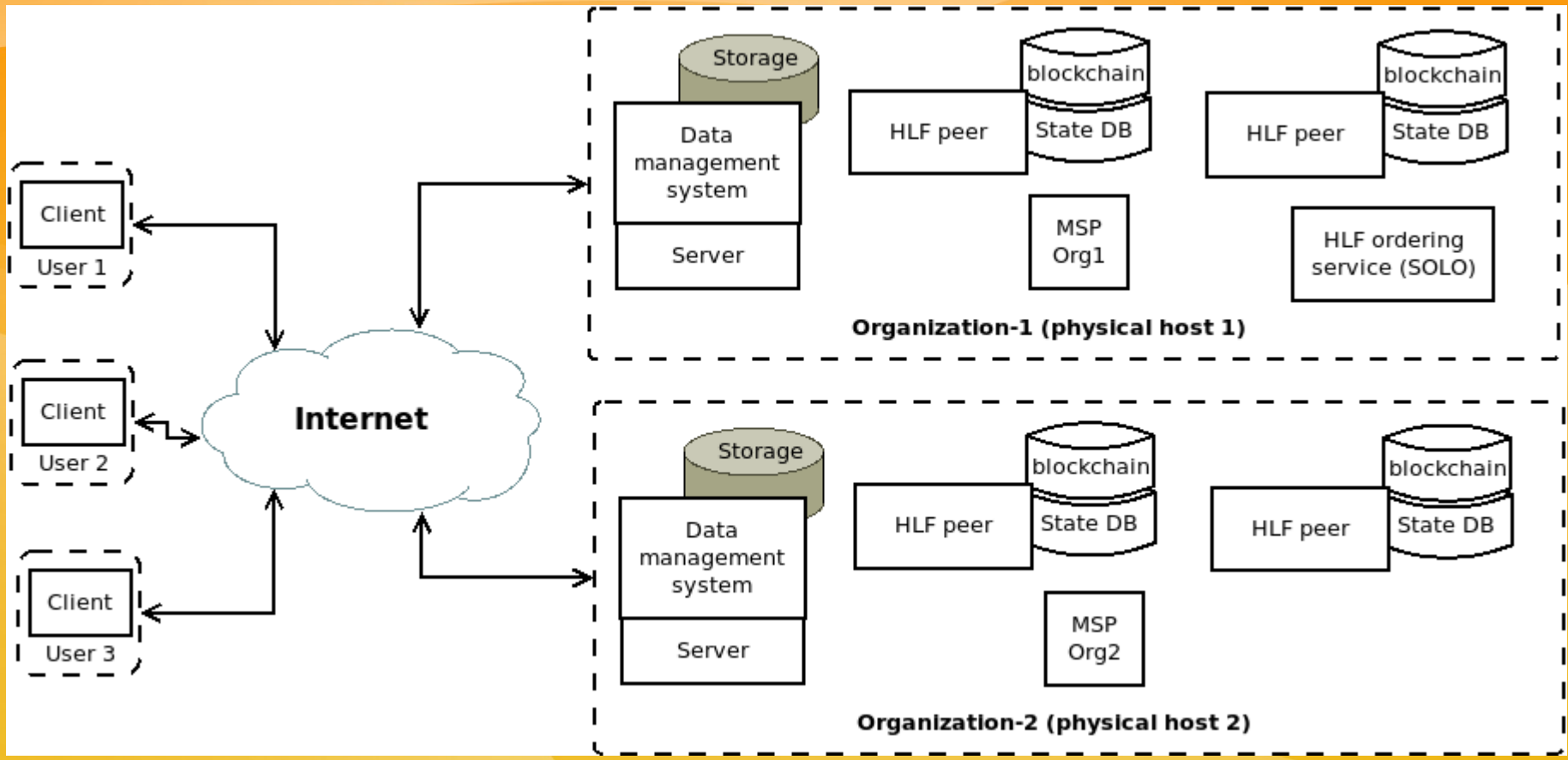
Compliance with Provenance Standards

- The W3C PROV standard defines a data model, serializations, and definitions to support the interchange of provenance information on the Web
  - include its data model,an XML schema for that model, mapping that model to RDF, and to Dublin Core
- The compliance of ProvHL to PROV is provided by the correspondence of the basic triples in the both models:
  - Asset (ProvHL) ⇔ Entity (PROV)
  - Operation/Transaction ⇔ Activity
  - Participant ⇔ Agent
- ⇒ MD from ProvHL can be presented in PROV form

ProvHL Testbed (1/2)

- At present, a testbed has been created on the basis of the SINP MSU
  - a preliminary version of the ProvHL prototype was deployed to implement the developed principles and refine the algorithms of the system
  - a trivial consensus algorithm is currently used (centralized orderer Solo in the terminology of HLF).
  - Kafka (crash tolerant) is under implementation
  - TODO: full-fledged Byzantine fault tolerant consensus algorithms

ProvHL Testbed (2/2)



Conclusion (1/4)

- we have formulated a general problem and functional requirements for the management system of provenance metadata and data access rights
  - support the implementation of business processes for storing and exchanging data in a distributed environment, with administratively unrelated or loosely connected organizations, participating in joint projects, or simply exchanging data on certain conditions.
- new approach based on the integration of blockchain technology, smart contracts and metadata driven data management

Conclusion (2/4)

- the principles and algorithms of the system, entitled ProvHL, are developed that are fault-tolerant, safe and secure management system of provenance metadata, as well as access rights to data in distributed storages.
- the problems of optimal choice of the blockchain type for such a system, as well as the choice of the blockchain platform are studied. Namely, it is proposed to use an permissioned type of blockchain and the Hyperledger blockchain platform, on the basis of which the ProvHL system (Provenance HyperLedger) is implemented.

Conclusion (3/4)

- At present, a testbed has been created on the basis of the institute, where a preliminary version of the ProvHL prototype is deployed to implement the developed principles and refine the algorithms of the system.
- In this preliminary version, a trivial consensus algorithm is used, in which the transaction recording order is determined by a single server (centralized orderer Solo in the terminology of HLF).
- in the future it is supposed to use full-fledged Byzantine fault tolerant consensus algorithms, in particular PBFT

Conclusion (4/4)