Data aggregation service in APPDS

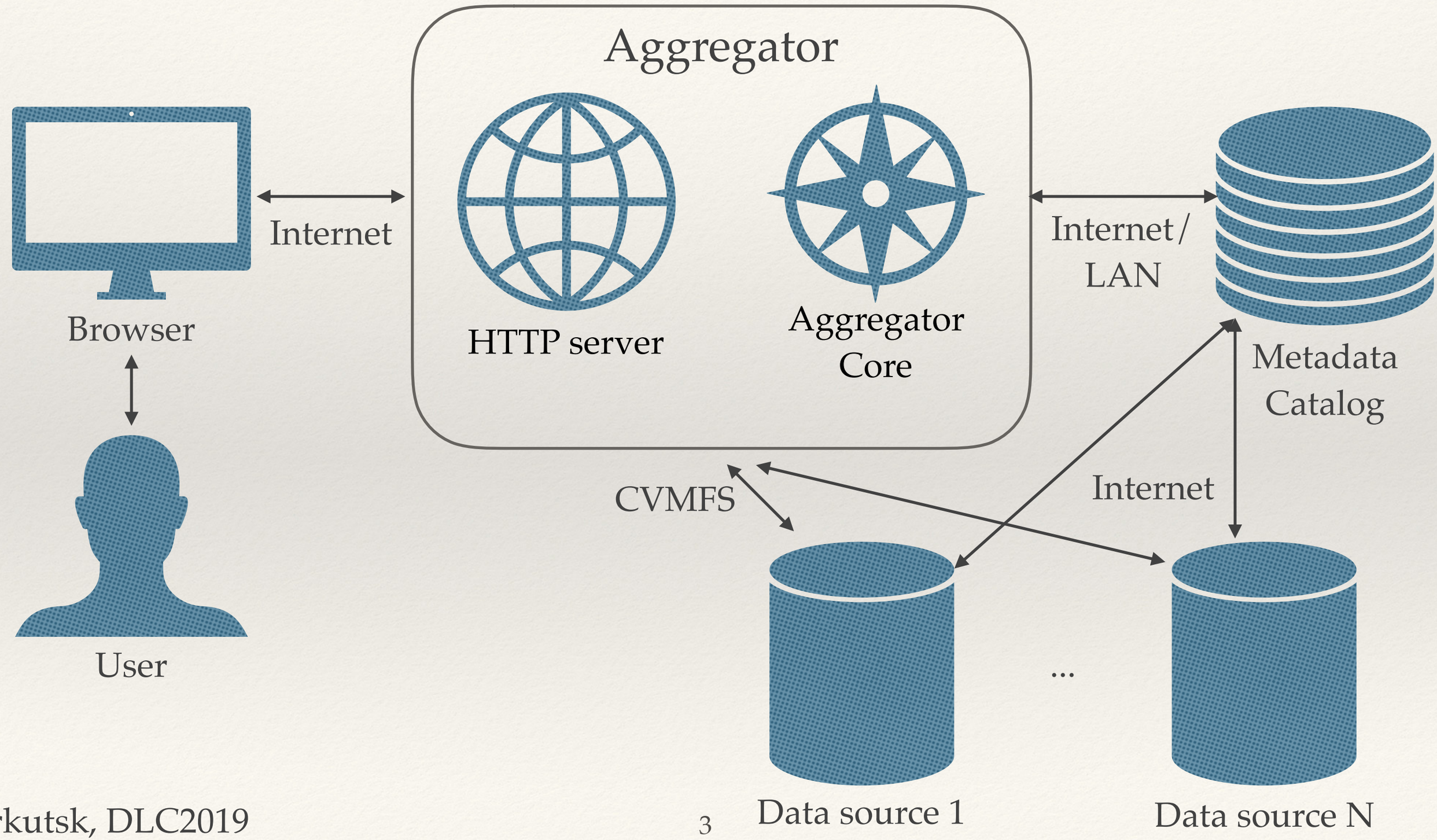
Minh-Duc Nguyen
Space research lab
SINP MSU

Irkutsk, DLC2019

Goals

- First working prototype before summer
- Beta-version this fall
- Implementation of the most used scenarios
- All happens in 2019

Architecture Overview



User queries

Query types

- Data from one facility or all facilities for a specific period
- Data for one fiscal year
- Testing data for a specific period
- One night data
- One run data
- ...

Query internals

- Query(ID: cafebabe) { files { start_dt, end_dt, facility, cluster, detector, channel, type } }

User responses

- Response (Query_ID: cafebabe)
  { URI: https://domain.com/query/cafebabe/response/ }
- Response (Query_ID: cafebabe)
  { error: { message: text }

Possible manipulations

- Browse, download, mount

Features

- The structure of files and directories is the same as in the original storage

Query processing - TAIGA case

- Query registration
- Cache hit/miss
- Checksum validation against the Metadata Catalogue
- Response size estimation
- Response generation
- First user response right after root directory generation
- Compressed archive preparation

Query processing - KCDC case

- Query registration
- Cache hit/miss
- Query delivery to KCDC data service (GraphQL)
- Checksum validation against the KCDC data service
- Response size estimation
- HTTP Streaming or Protocol Buffer?

Response internals - TAIGA case

- Generating file and directory structure
- Copying/Composing files
- Generating a CVMFS repository for the query
- Generating a compressed archive of the response
- Updating the query status in the cache system

Response internals - KCDC case

- Generating files (json or csv) using chunks received from KCDC
- Generating a CVMFS repository for the query
- Generating a compressed archive of the response
- Updating the query status in the cache system

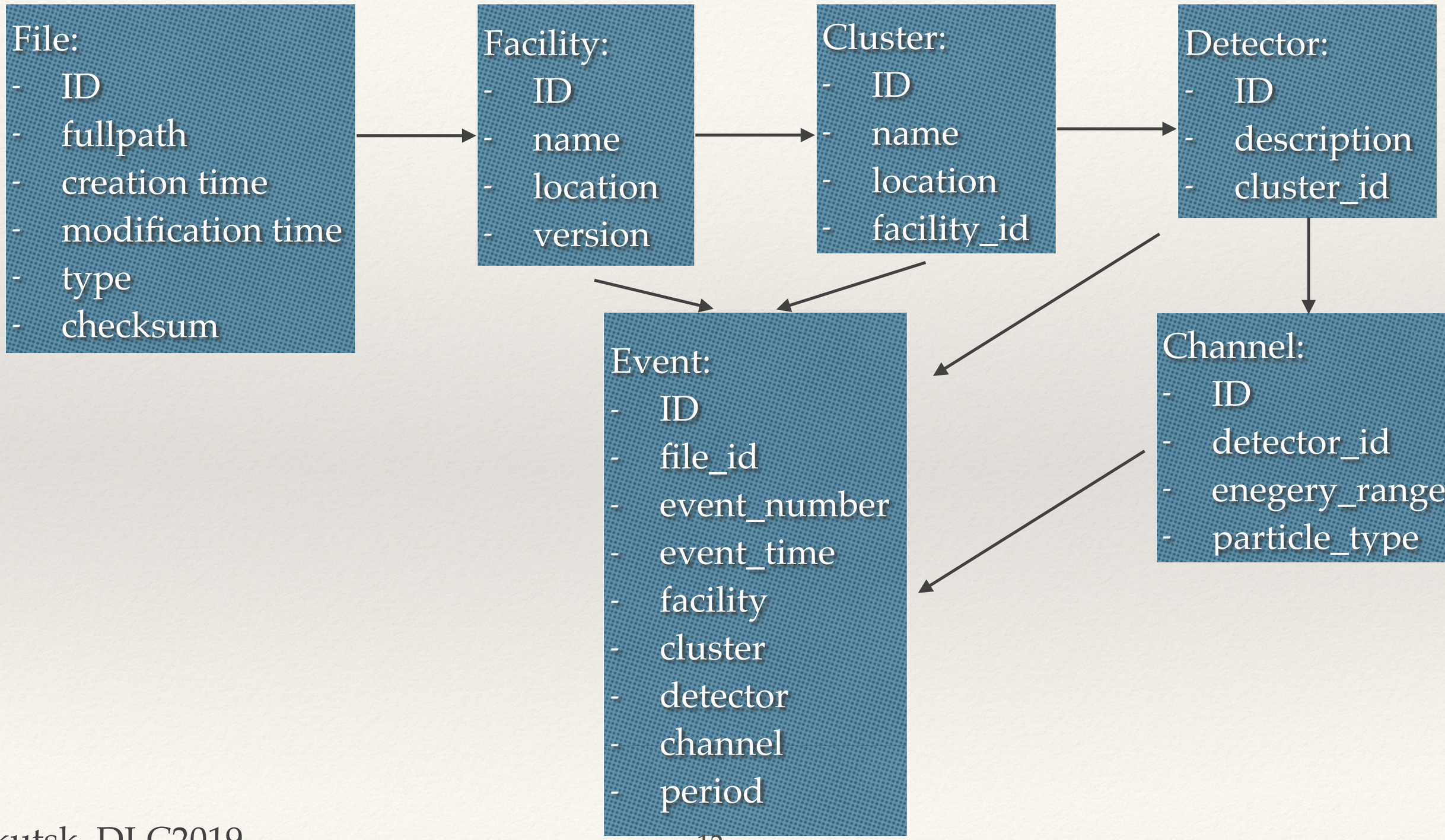
Stack of technologies

- Web server: nginx
- Business logic: Python, Django, Airflow
- Data source: CVMFS
- Query cache: Redis
- Front-end: React.js + Redux
- Database: PostgreSQL + Timescale plugin
- GraphQL for query, Python-based API for data access
- Devops: Docker

Approach to schema & table creation

- one facility => one schema
- denormalize the schemas & tables whenever possible to reduce join-operations and the search time
- fixed fields => standard PostgreSQL types
- nonformalized fields => PostgreSQL jasonb

Metadata DB structure



Thank you

- Minh-Duc Nguyen
- nguyendmitri@gmail.com