Euclid Science Archive technical evolution to the Petabyte scale

P. Pérez[1], P. de Teodoro[2], J. Prados [1], J. Olveira [2], V. Olmedo [1], J.L. Cánovas [1], J. Saiz [1], J. Férnandez [1], C. Ríos [1], V. Fernández [1], R. Prieto [1], M. López-Caniego [2], ESA PIT and S. Nieto

ESDC at European Space Astronomy Center, Madrid, Spain

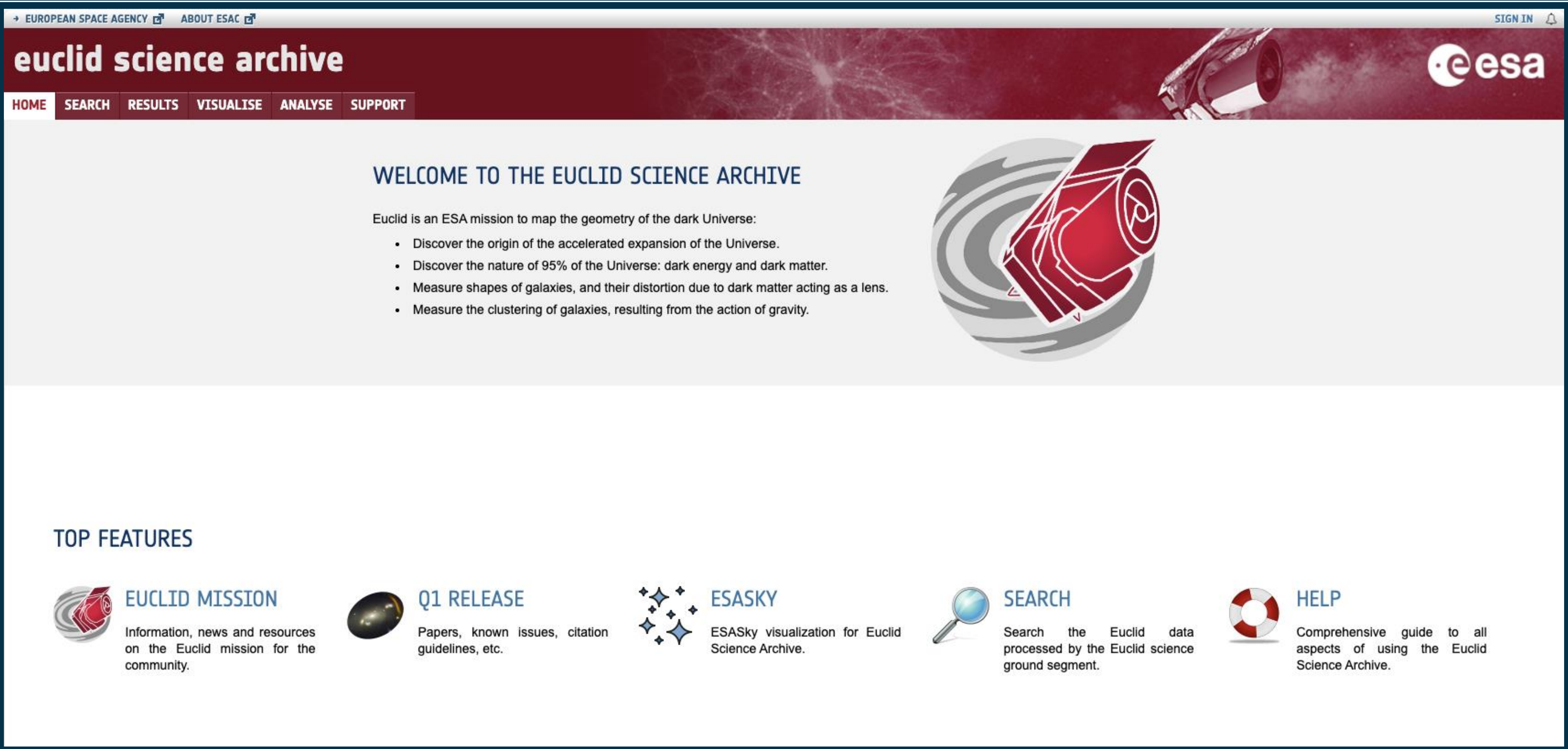
1 Starion Group for ESA, 2 Aurora Technology for ESA

The Euclid mission

- The Euclid mission of the European Space Agency (ESA), launched in July 2023
- Investigates the origin of the Universe and the nature of dark energy, dark matter and gravity
- Will be operating for 6 years
- Expected over 30PB of processed data at the end of the mission

- Euclid is a BIG DATA mission

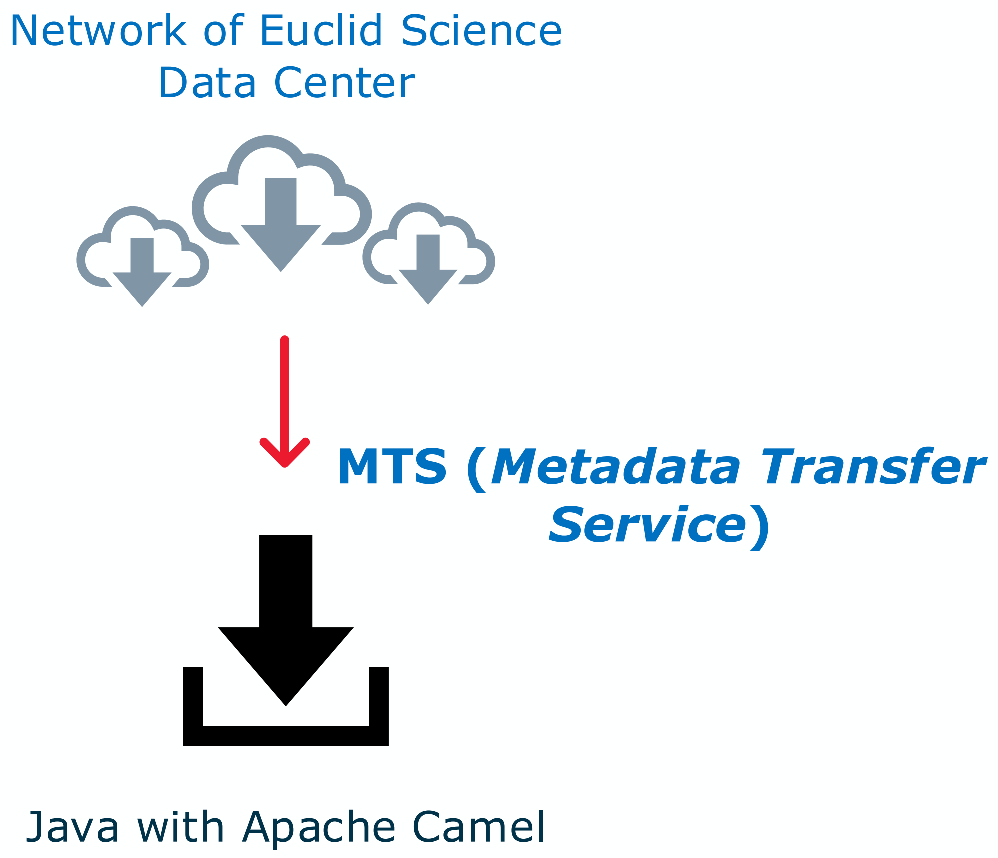
The Euclid Science Archive



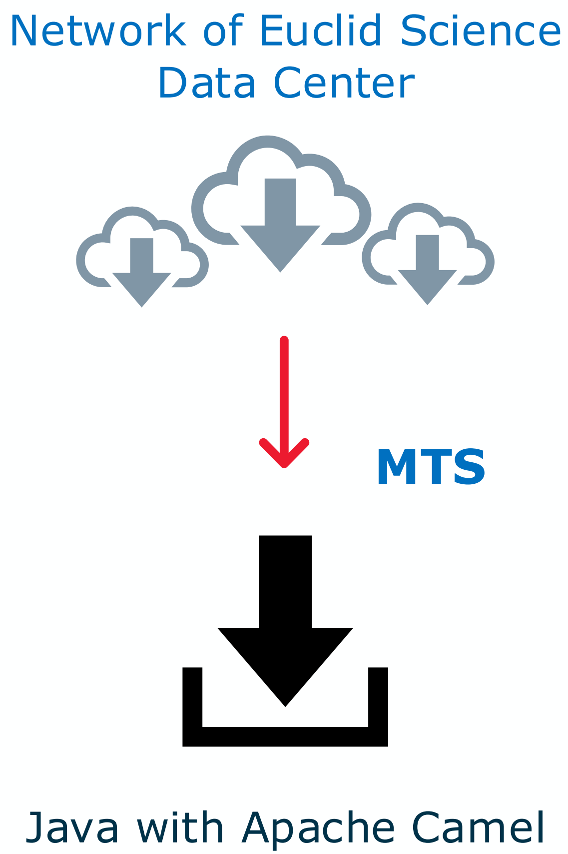
- https://euclid.dataspace.esa.int/data-discovery
- https://eas.esac.esa.int/sas/

EAS Data Transfer Pipeline

EAS Data Transfer Pipeline

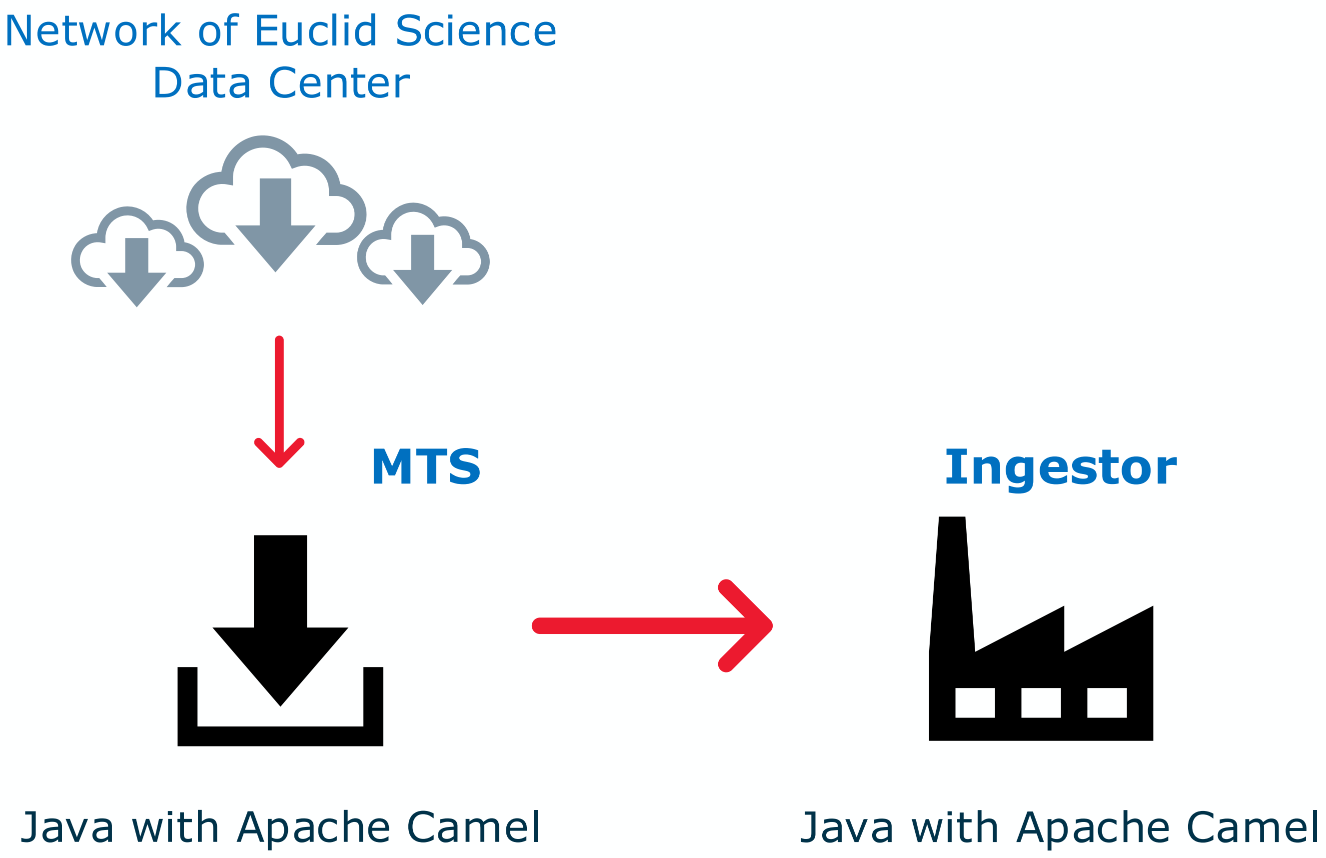


EAS Data Transfer Pipeline

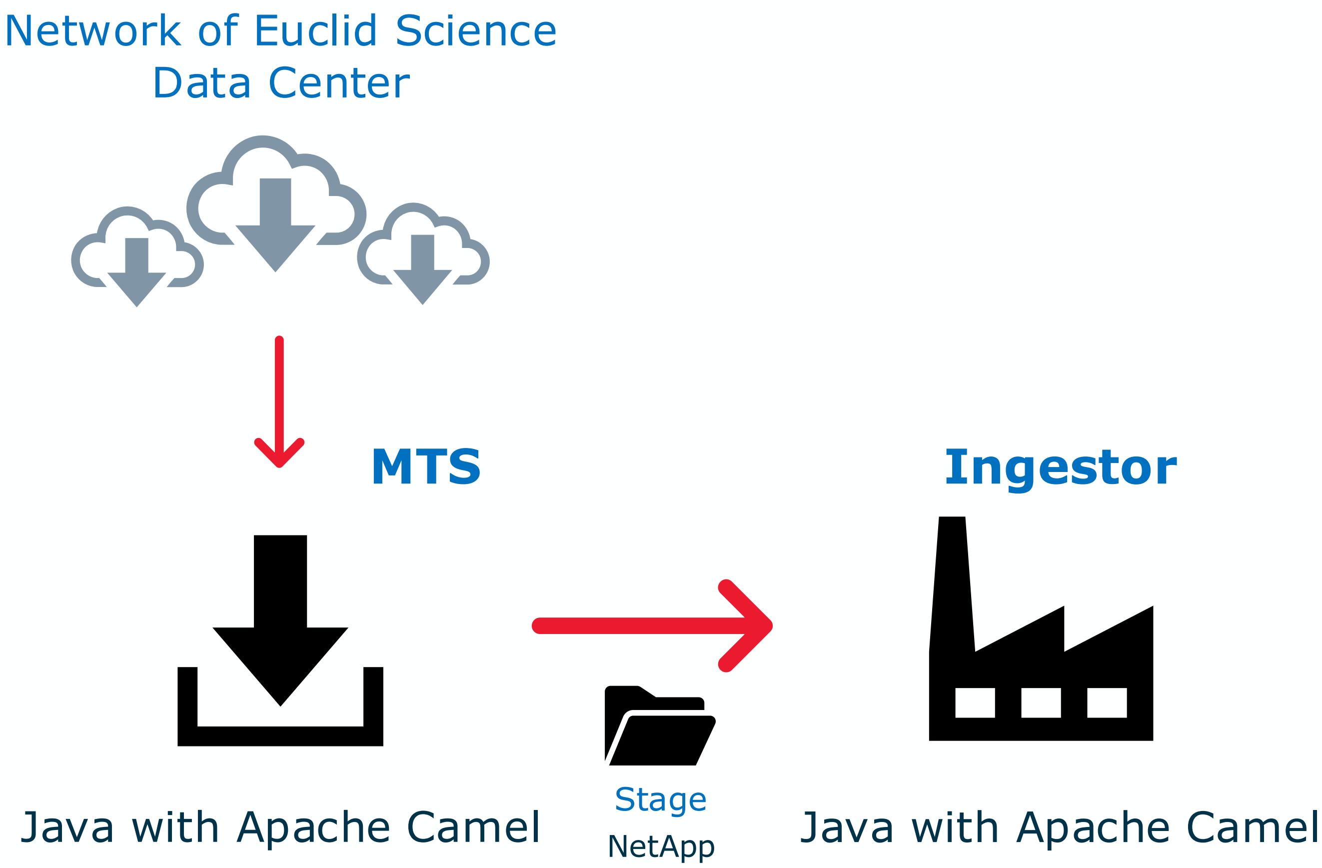


1. Queries the DPS (*Data Processing System)* for relevant products
2. Downloads the XML
3. Downloads the associated files from the different *Distibuted Storage System* (DSS) nodes
    1. Multiple countries across the world

EAS Data Transfer Pipeline

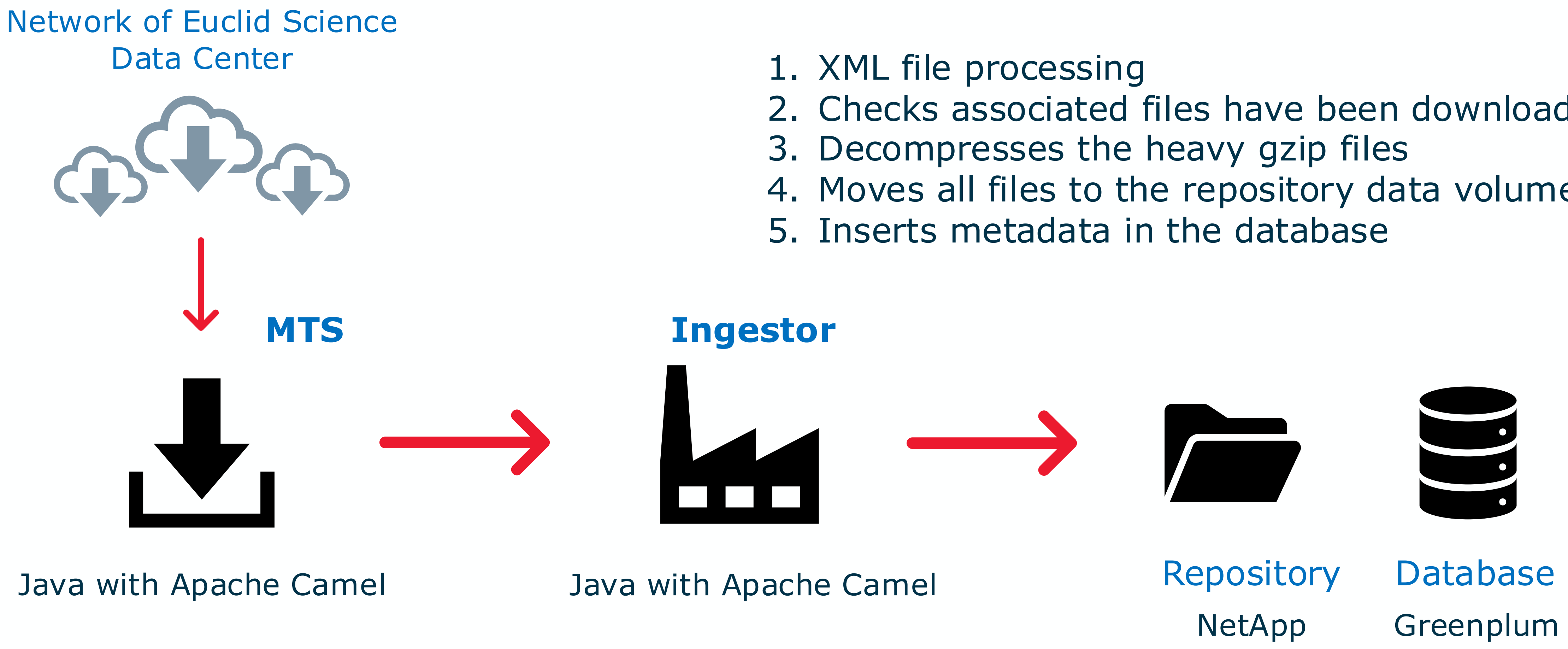


EAS Data Transfer Pipeline

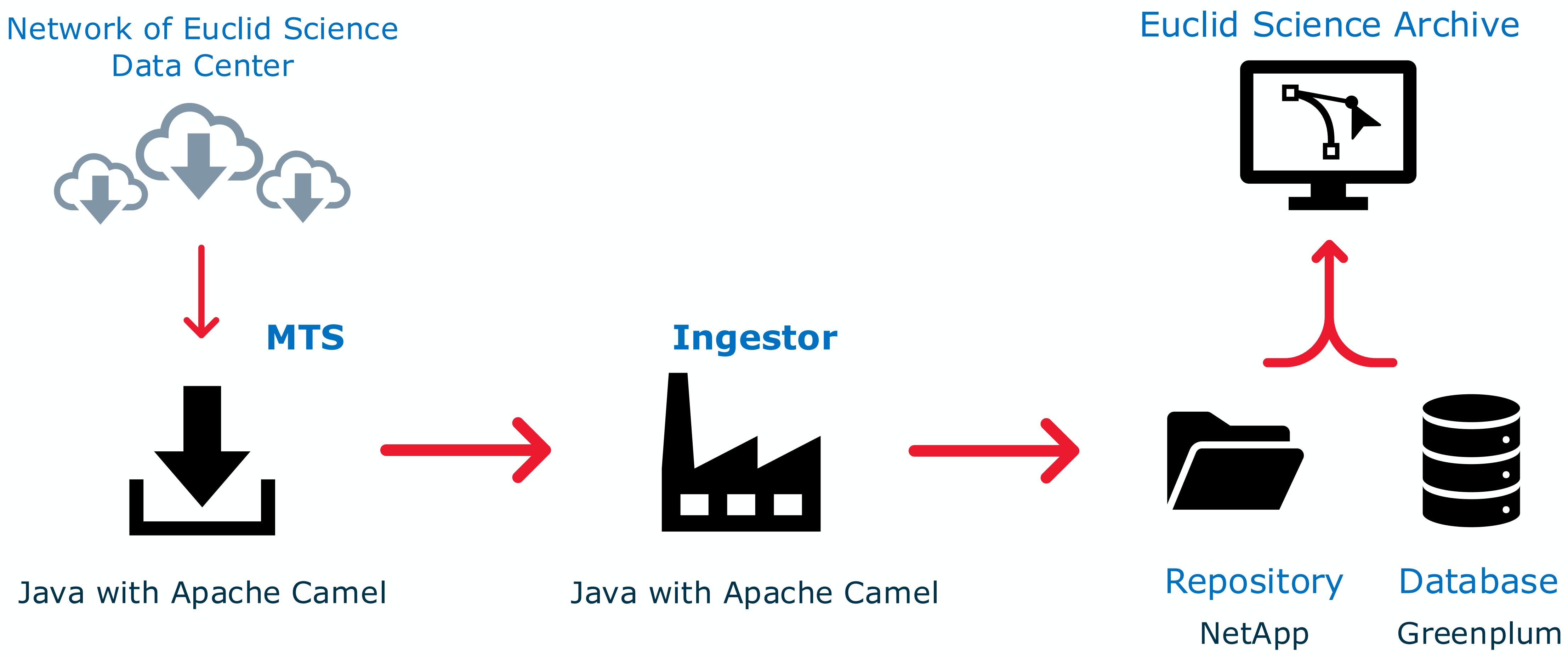


EAS Data Transfer Pipeline

1. XML file processing
2. Checks associated files have been downloaded
3. Decompresses the heavy gzip files
4. Moves all files to the repository data volume
5. Inserts metadata in the database



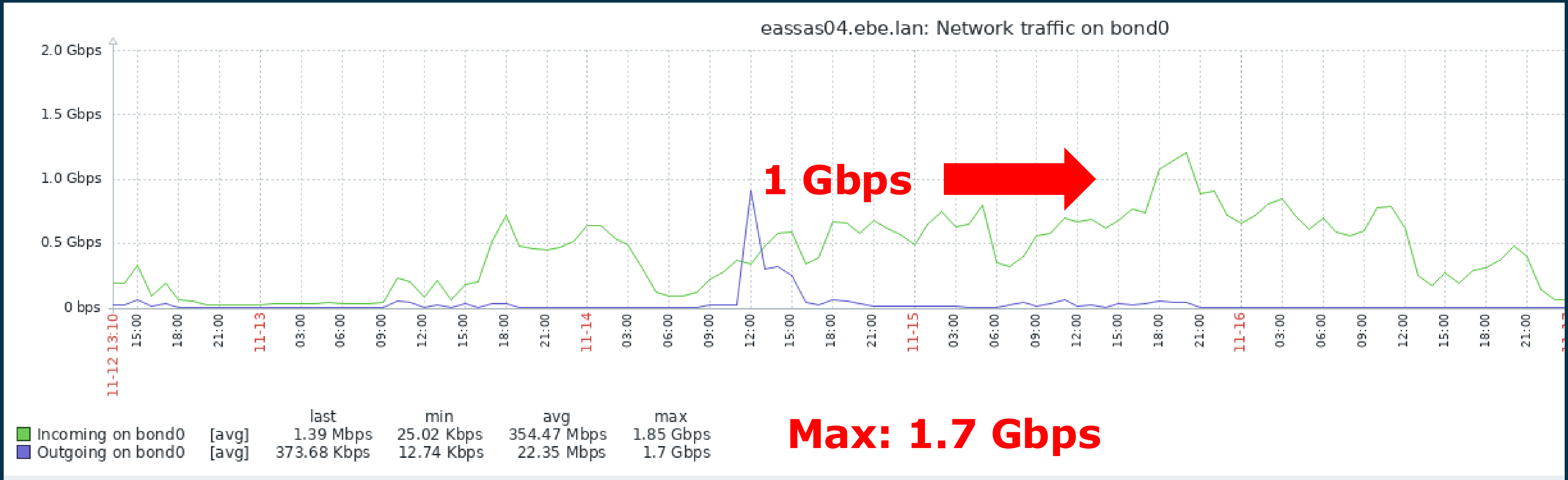
EAS Data Transfer Pipeline



EAS Data Transfer Pipeline

Throughput before improvements

- Up to **4 TB/day** of ingested data
- **Is this enough for Euclid?**



Looking into Euclid IDR1

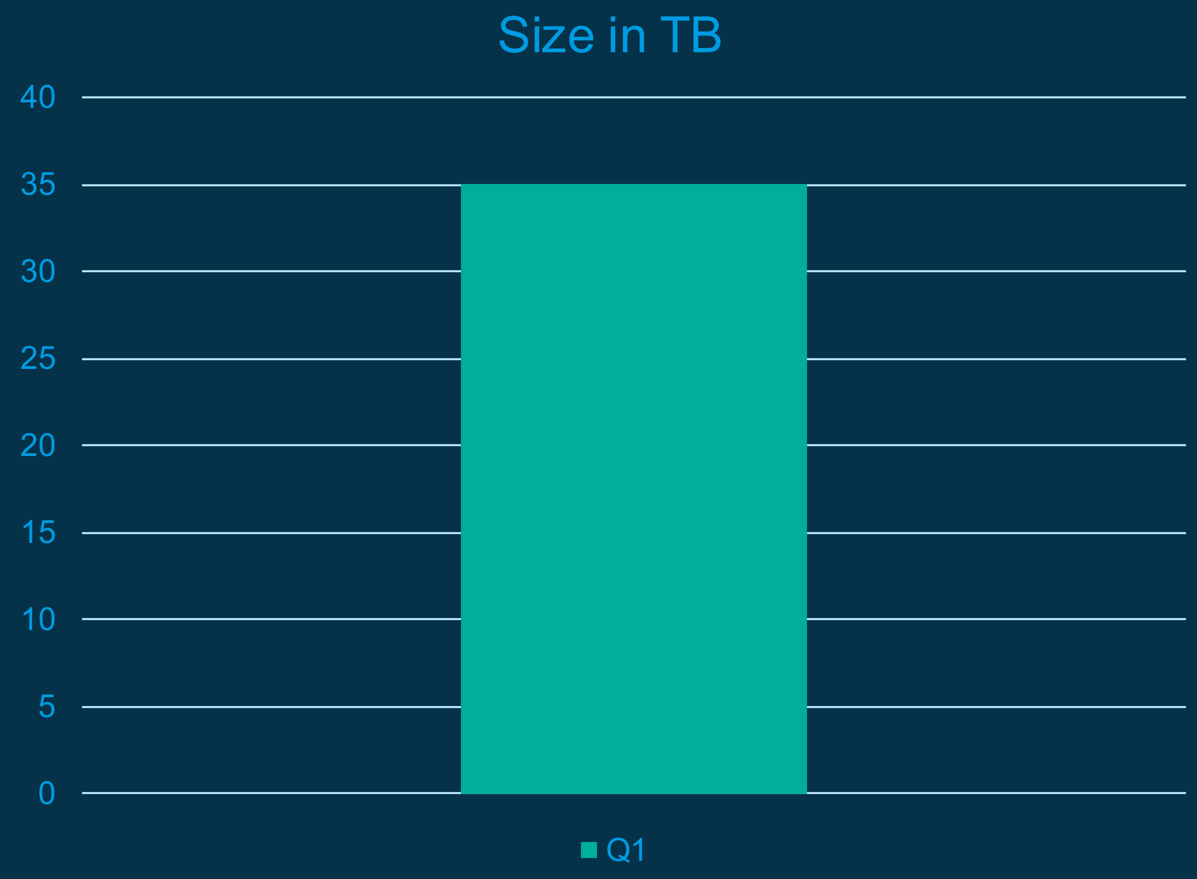
What is Euclid IDR1?

- Euclid IDR1: first major release of the Euclid mission
- Targeted to Euclid Consortium members
- The data contents of IDR1 will be released to the public Q4 of 2026 as the **DR1**
    - First major public release of Euclid

Looking into Euclid IDR1

IDR1 in context

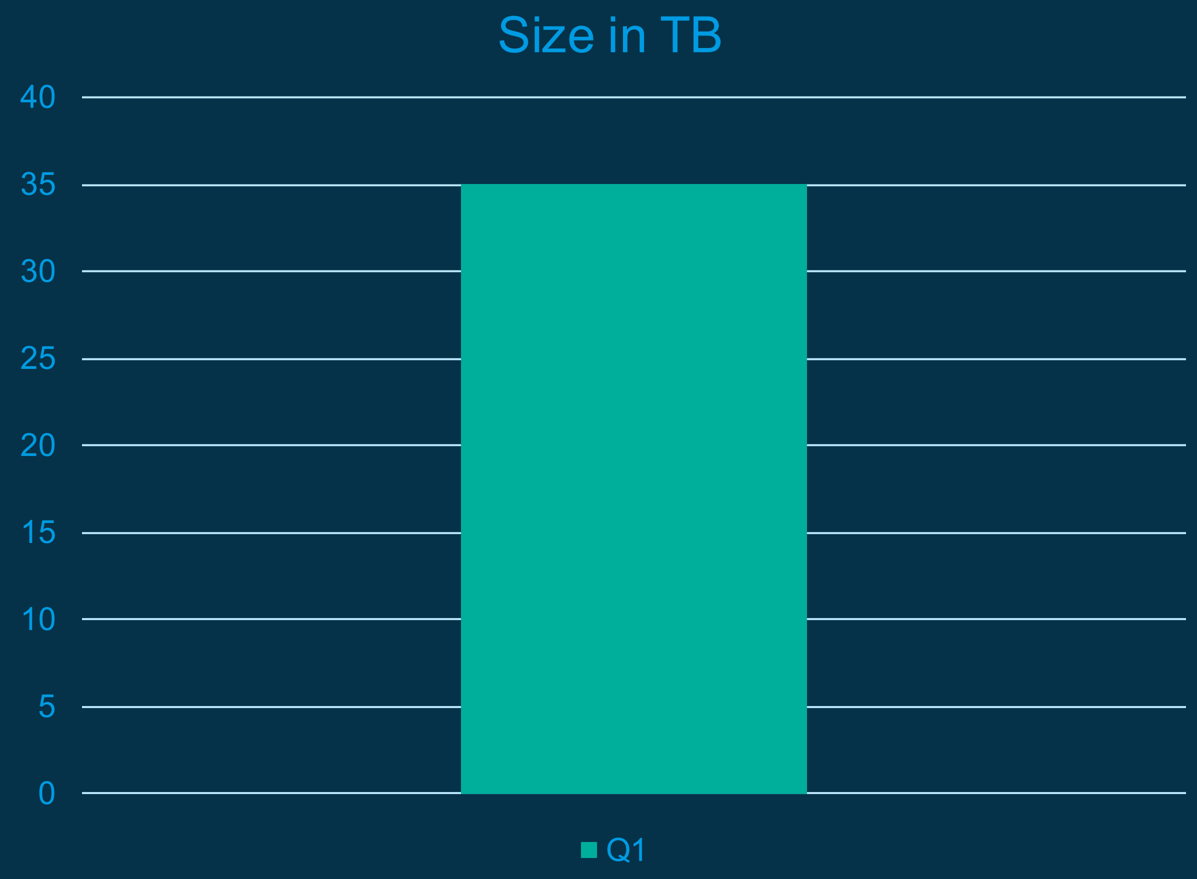
- Euclid Q1 released March 19th 2025
  - First public release of the Euclid mission
  - 35 TB



Looking into Euclid IDR1

IDR1 in context

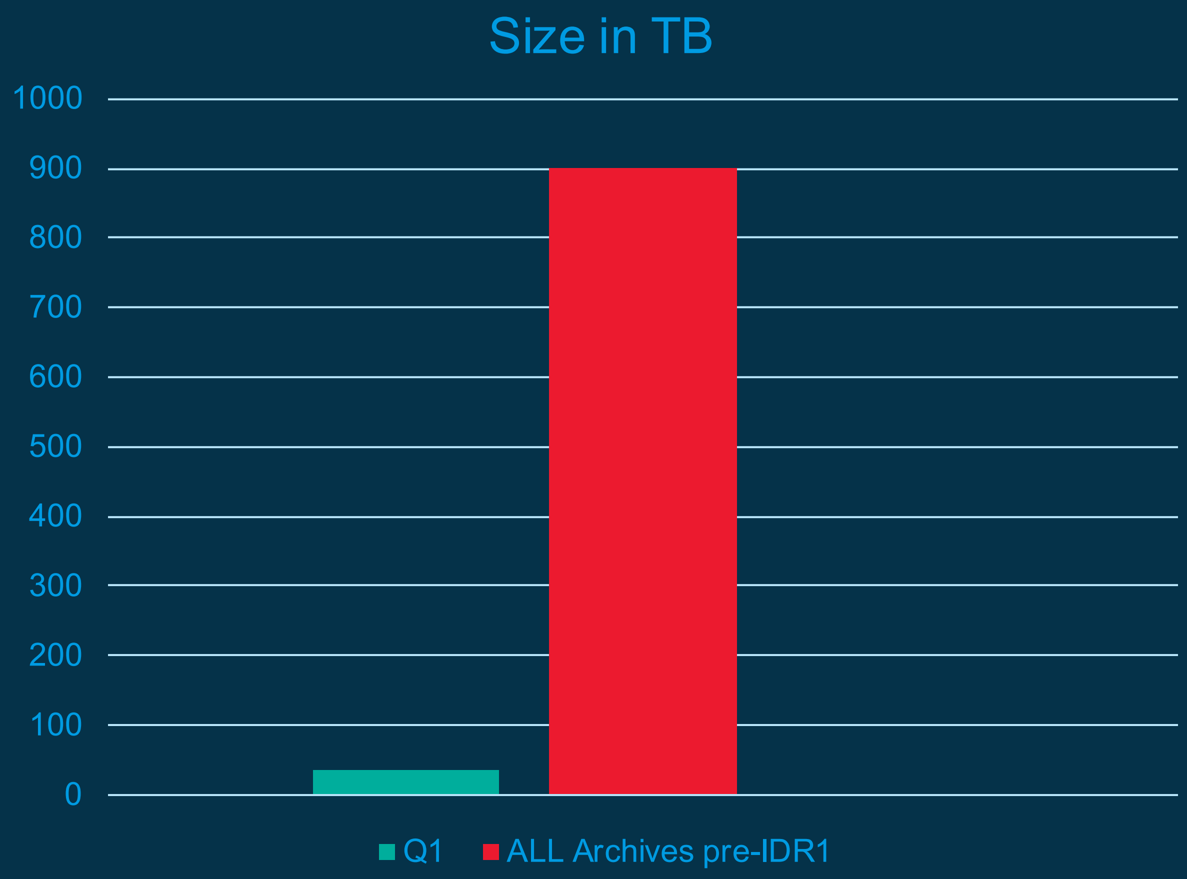
- Euclid Q1 released March 19th 2025
    - First public release of the Euclid mission
    - 35 TB
- ALL Archives before Euclid IDR1
    - ~900 TB
    - 30 years worth of missions data



Looking into Euclid IDR1

IDR1 in context

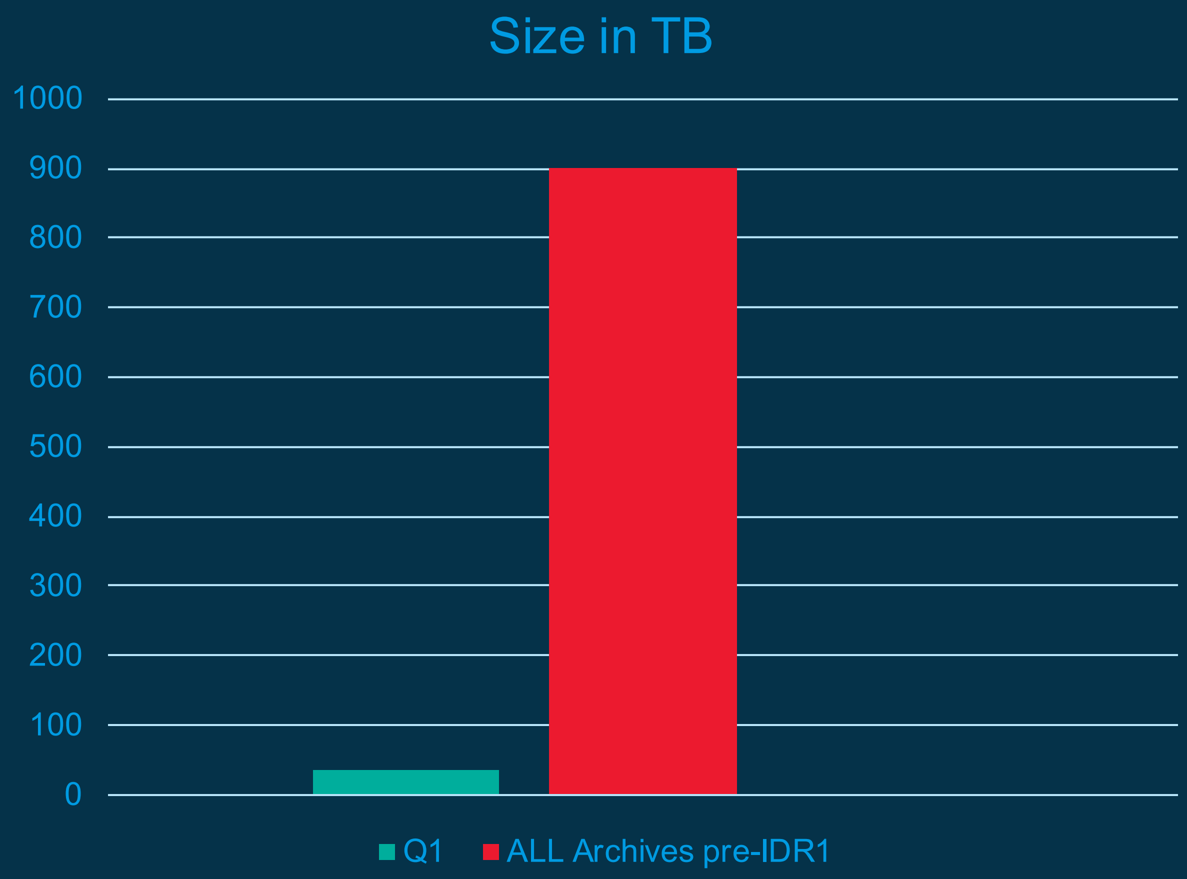
- Euclid Q1 released March 19th 2025
    - First public release of the Euclid mission
    - 35 TB
- ALL Archives before Euclid IDR1
    - ~900 TB
    - 30 years worth of missions data



Looking into Euclid IDR1

IDR1 in context

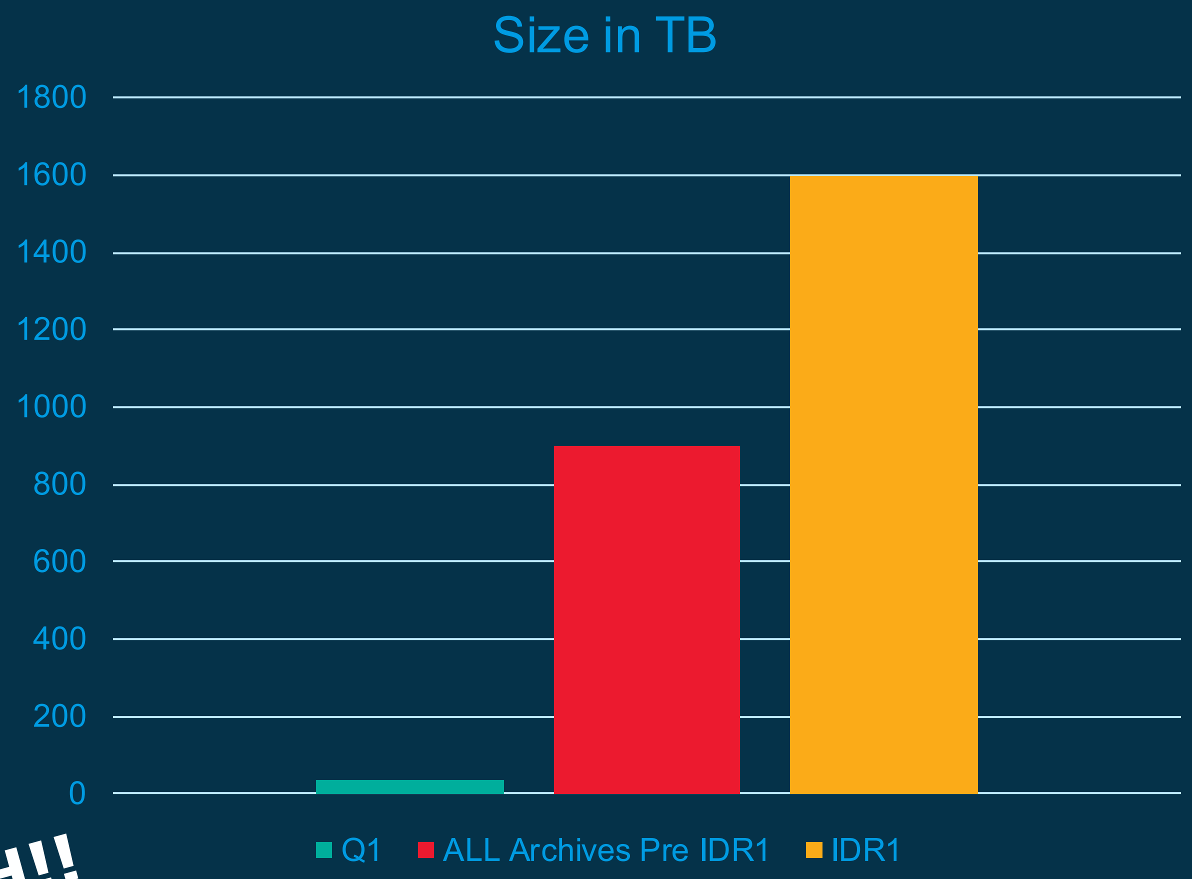
- Euclid Q1 released March 19th 2025
  - First public release of the Euclid mission
  - 35 TB
- ALL Archives before Euclid IDR1
  - ~900 TB
  - 30 years worth of missions data
- Euclid IDR1
  - 1.5 PB



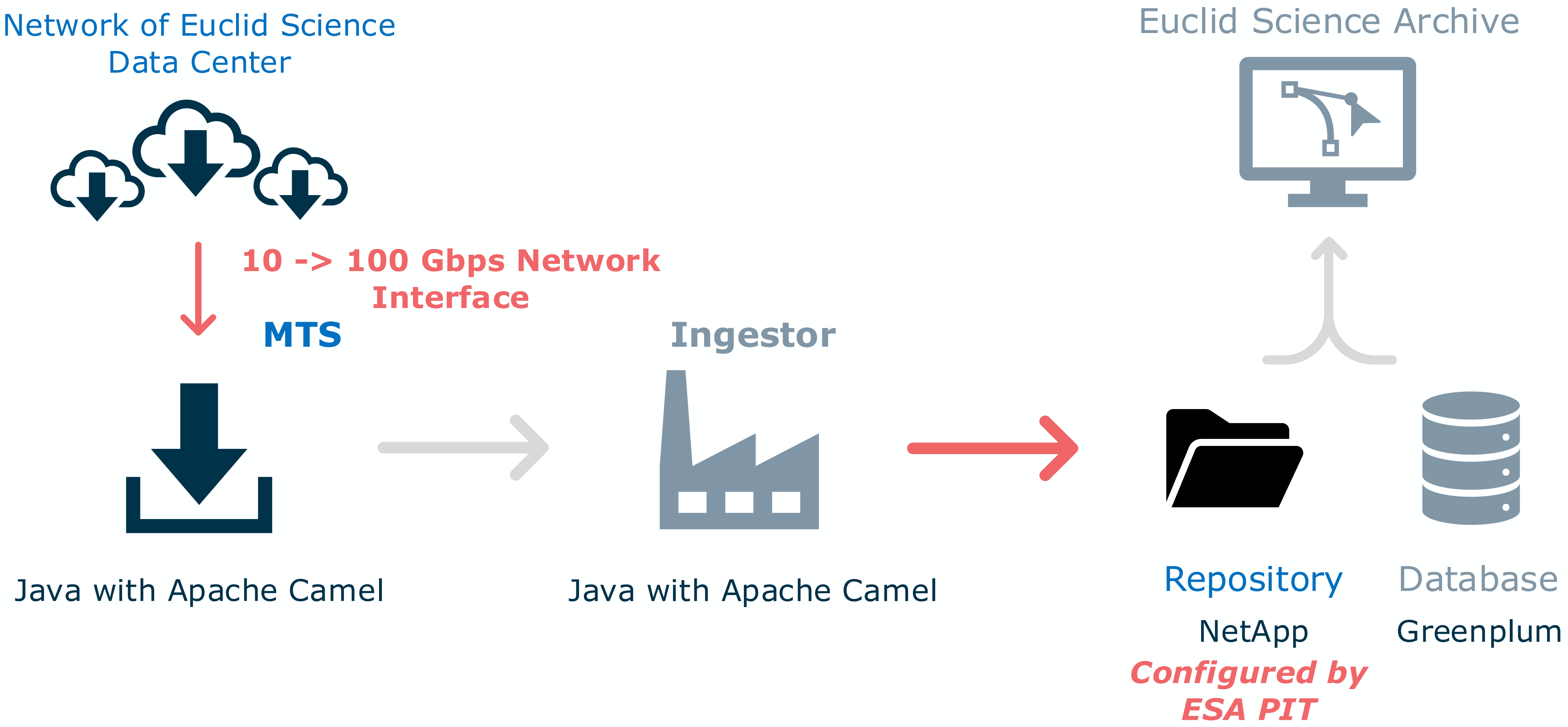
Looking into Euclid IDR1

IDR1 in context

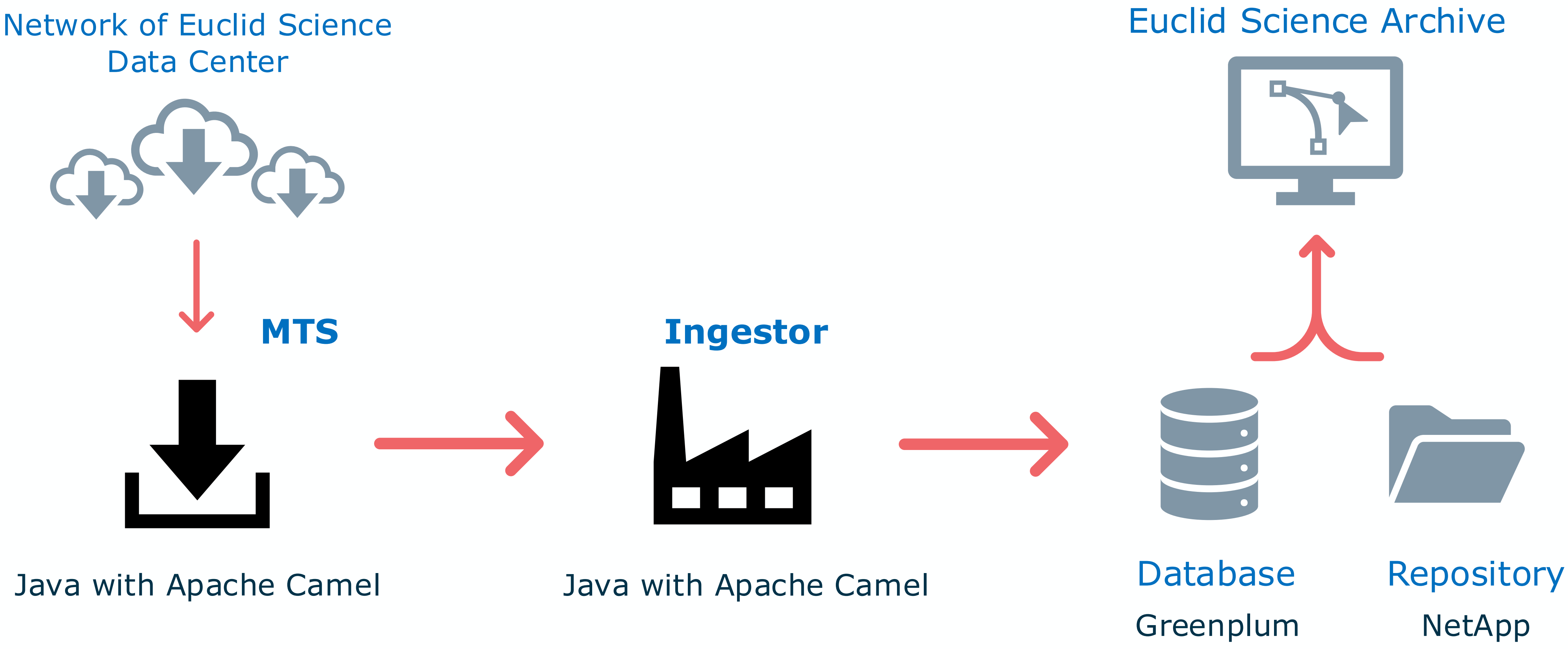
- Euclid Q1 released March 19th 2025
  - First public release of the Euclid mission
  - 35 TB
- ALL Archives before Euclid IDR1
  - ~900 TB
  - 30 years worth of missions data
- Euclid IDR1
  - 1.5 PB
  - Needs to be finished in around 2 months
  - At previous speed -> **250 days** **NOT ENOUGH!!**



Improving the infrastructure



Improving the Code



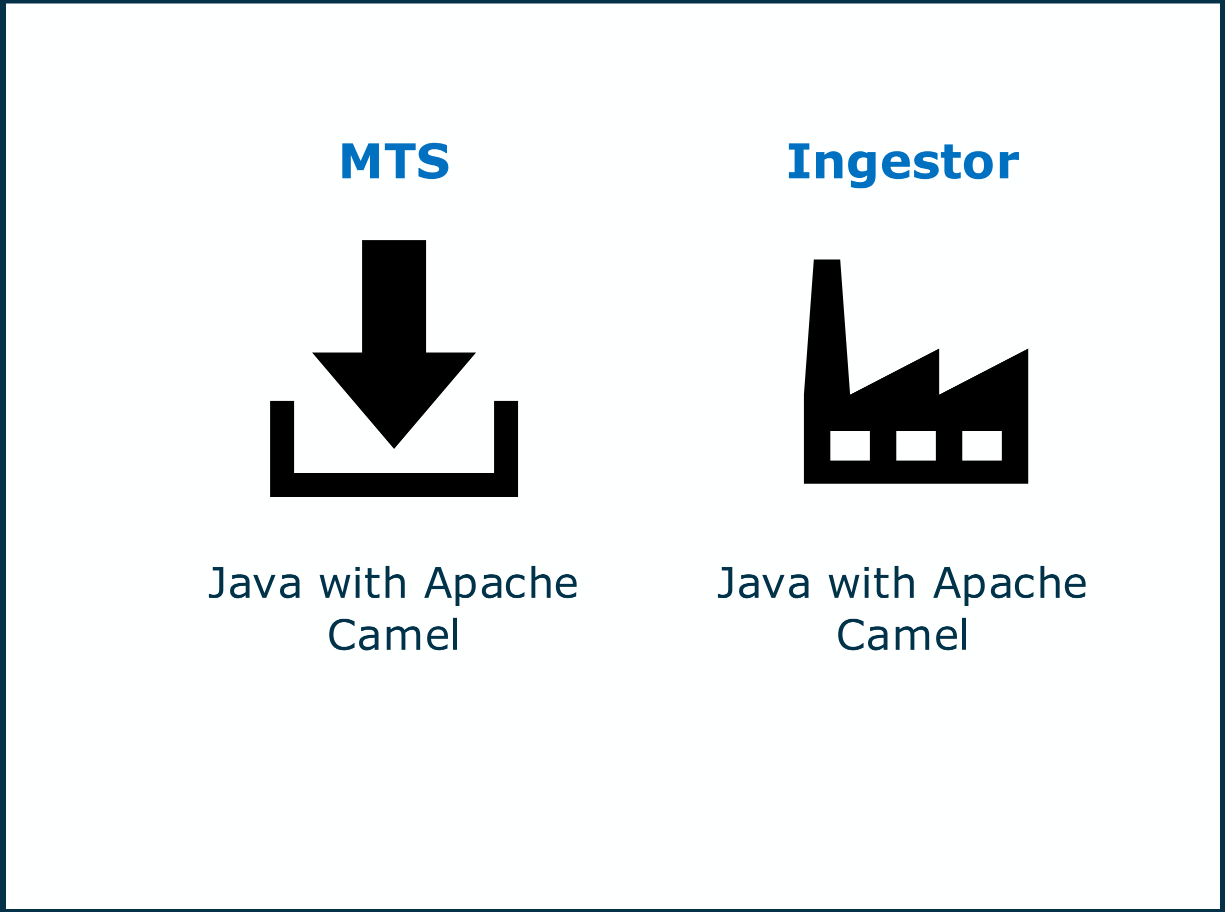
Improving the Code

- **CHALLENGE**
  - Moving from not parallelized code to fully parallelized multithreaded code
- **DIRECTION**
  - Taking advantage of Apache Camel's parallelization tools and efficient message routing management
- Complete rework of the code is a big risk
  - Could take too much time without improvements and time is a big constraint
  - Ingestion needs to **keep going** while the team works on the upgrade
- **DECISION**
  - **Incremental approach** -> divide improvement in multiple releases

Improving the Code

Incremental approach - **Planning**

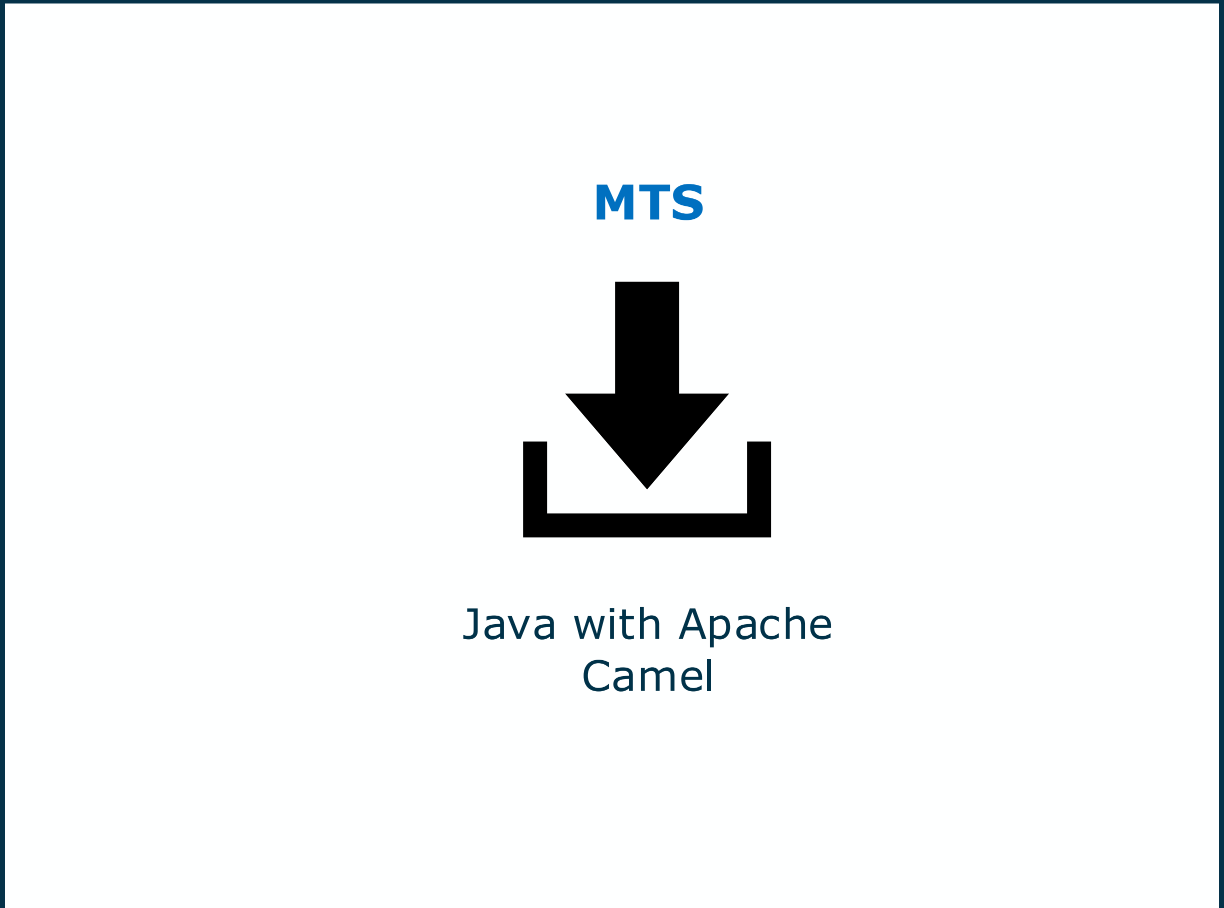
1. MTS
2. Ingestor -> file movement
3. Ingestor -> file processing and metadata ingestion



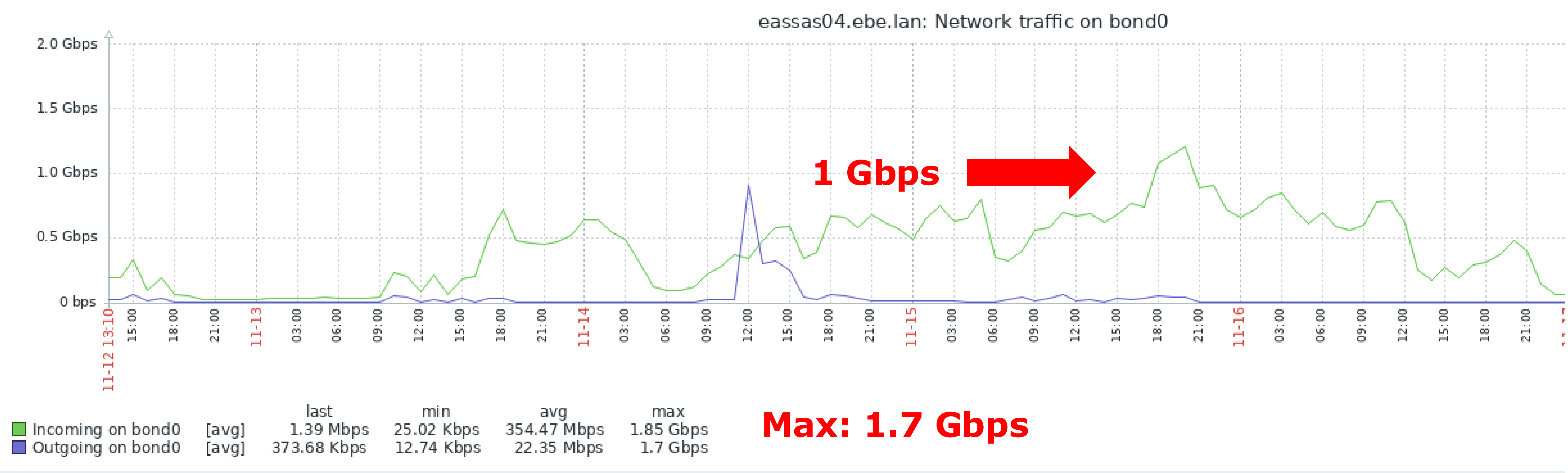
Improving the Code

Incremental approach - **MTS**

- Only downloads, easier use case
- Moved away from big Java functions
  - Cleaned the code
  - Small, specific functions
  - All the steps of the process done via Camel routes and Camel processors
- Took advantage of *toD* and *seda* Camel components
- Results: **4x download speed**

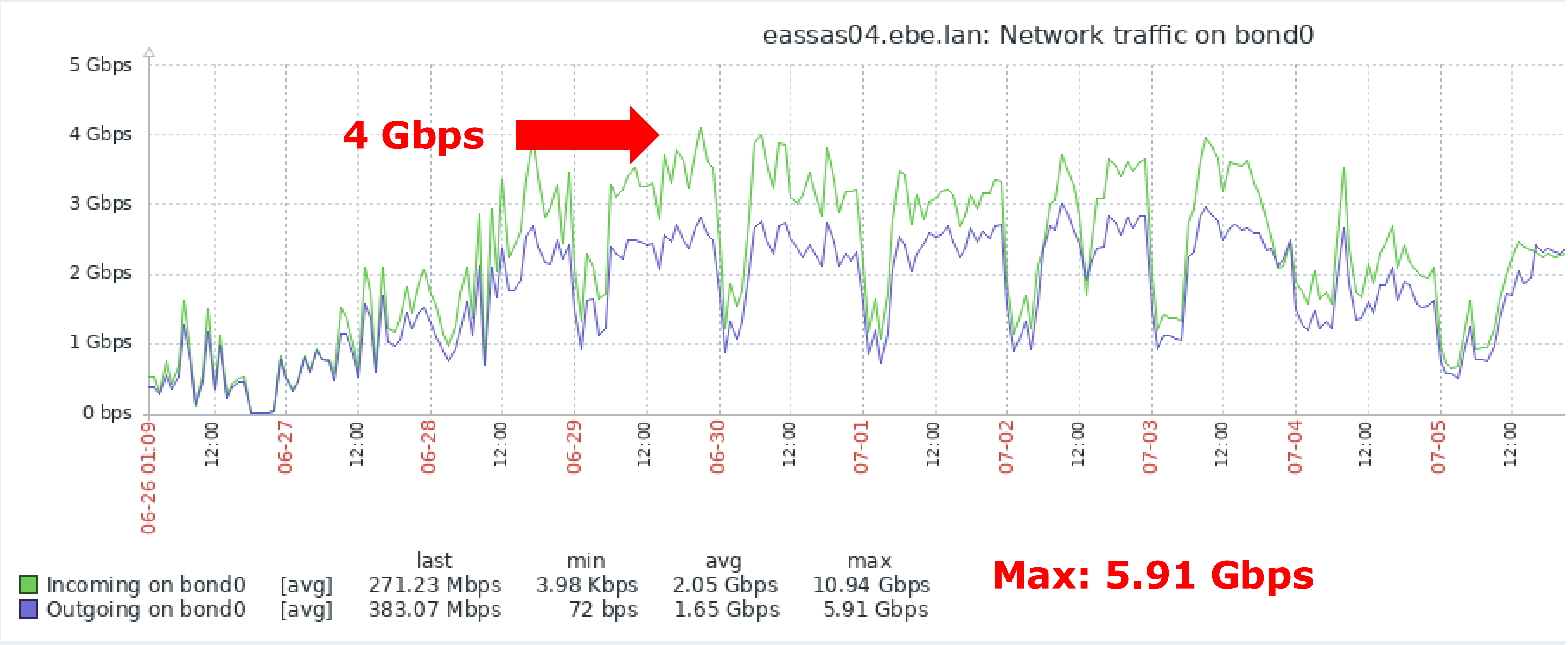


MTS Improvement - before



MTS improvement - after

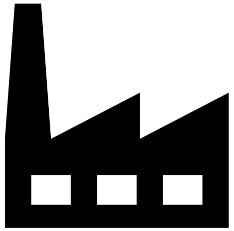
4x download speed



Incremental approach – **Ingestor** – Part 1

- Two steps of the process
    1. Move the files from staging to repository
    2. Process XML files and prepare metadata for the database
- **1. Parallelise the file movement**
    - Similar code rework as MTS
    - All Euclid product types will benefit

Ingestor

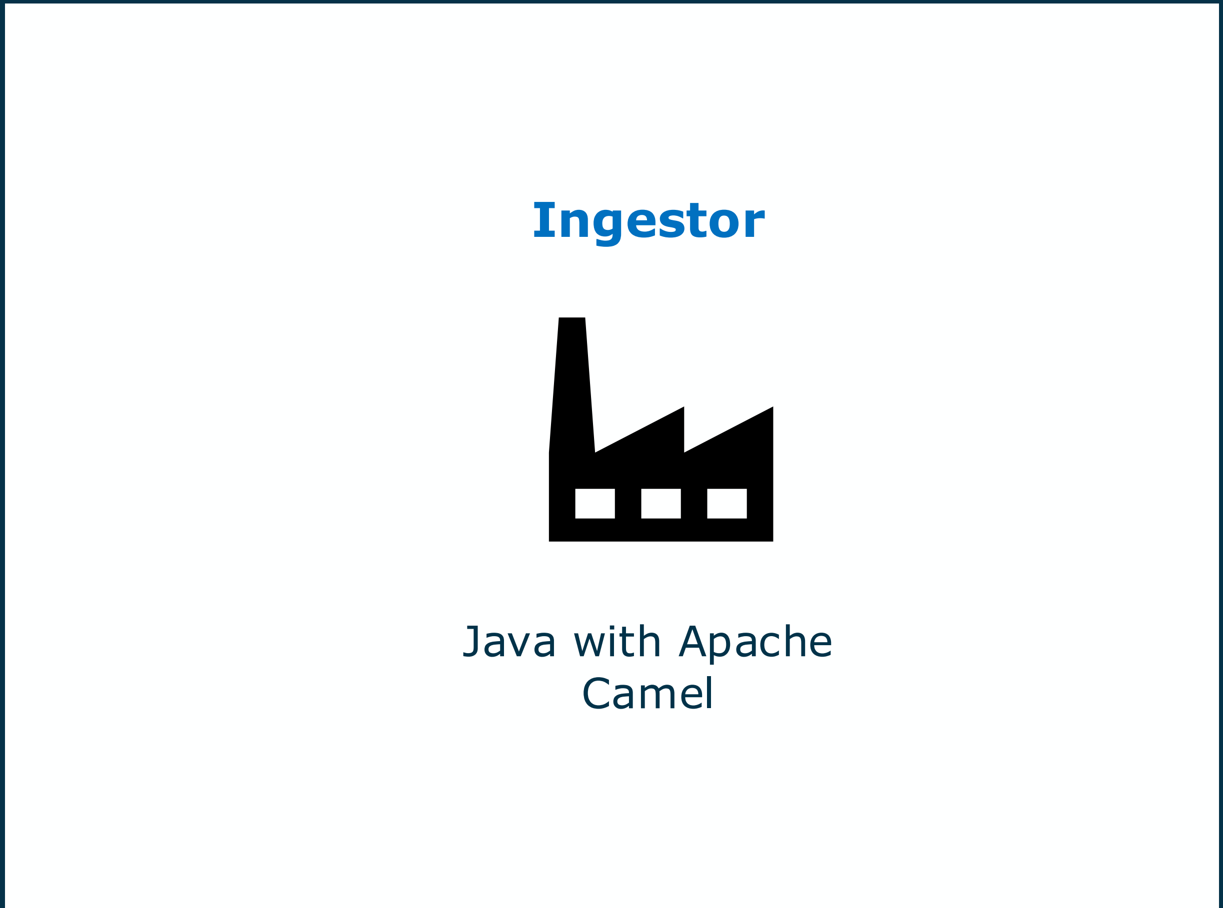


Java with Apache Camel

Improving the Code

Incremental approach – **Ingestor** – Part 2

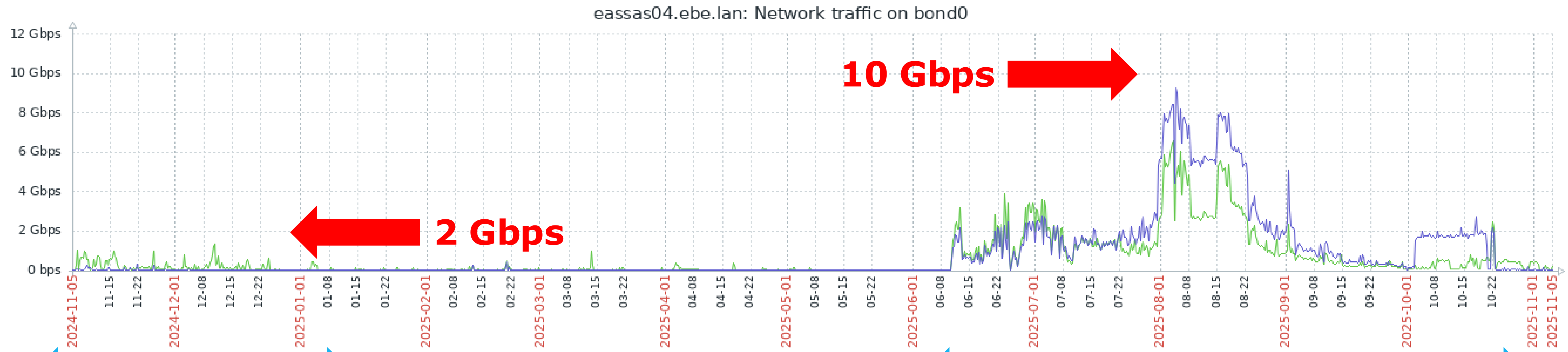
- **2. Parallelise XML processing and metadata insertion**:
    - Complex process
    - Multiple scenarios with *a lot* of race conditions
- Divide the implementation further:
    1. First, parallelise the biggest products (bottleneck)
    2. Then, product by product



Final picture

- All products take advantage of Apache Camel’s processing
  - Moved away from extensive, complex Java functions
  - Small and specific functions are called by the different steps of the process
  - Used *threadPools*, *onCompletion*, *choice* and *toD* componentes to achieve it
- Number of threads is dynamically configurable in database – **per product type!**
  - Threads for XML processing
  - Threads for file movement
  - **Benefit of the incremental approach**

Current performance



Performance before

Performance during improvement

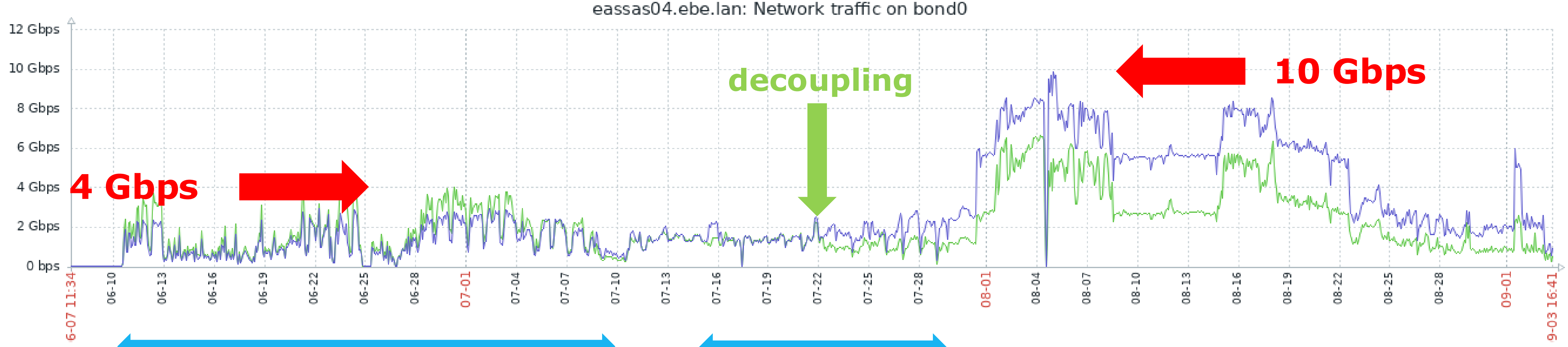
Zoom in

Current performance

Green: download

Blue: ingestion

Max: 11.8 Gbps



1. MTS improvement

2. Ingestor: File movement

3. Ingestor: XML processing (main products)

4. Ingestor: XML processing (product by product)

Current performance

- Up to **64 TB/day**
    - More than **x10** speed at max performance
- Full dataset Ingested in around 2 months
    - From 250 days

IT'S MUCH BETTER!!

Current performance

- Up to **64 TB/day**
  - More than **x10** speed at max performance
- Full dataset Ingested in around 2 months
  - From 250 days
- Done with incremental approach – not full potencial
  - ~20 threads for moving files
  - ~20 threads for processing
- **STRONG FOUNDATION**
  - The performance can **still improve** if applied to all products from the beginning, with **more** threads
  - (*60 threads?*)

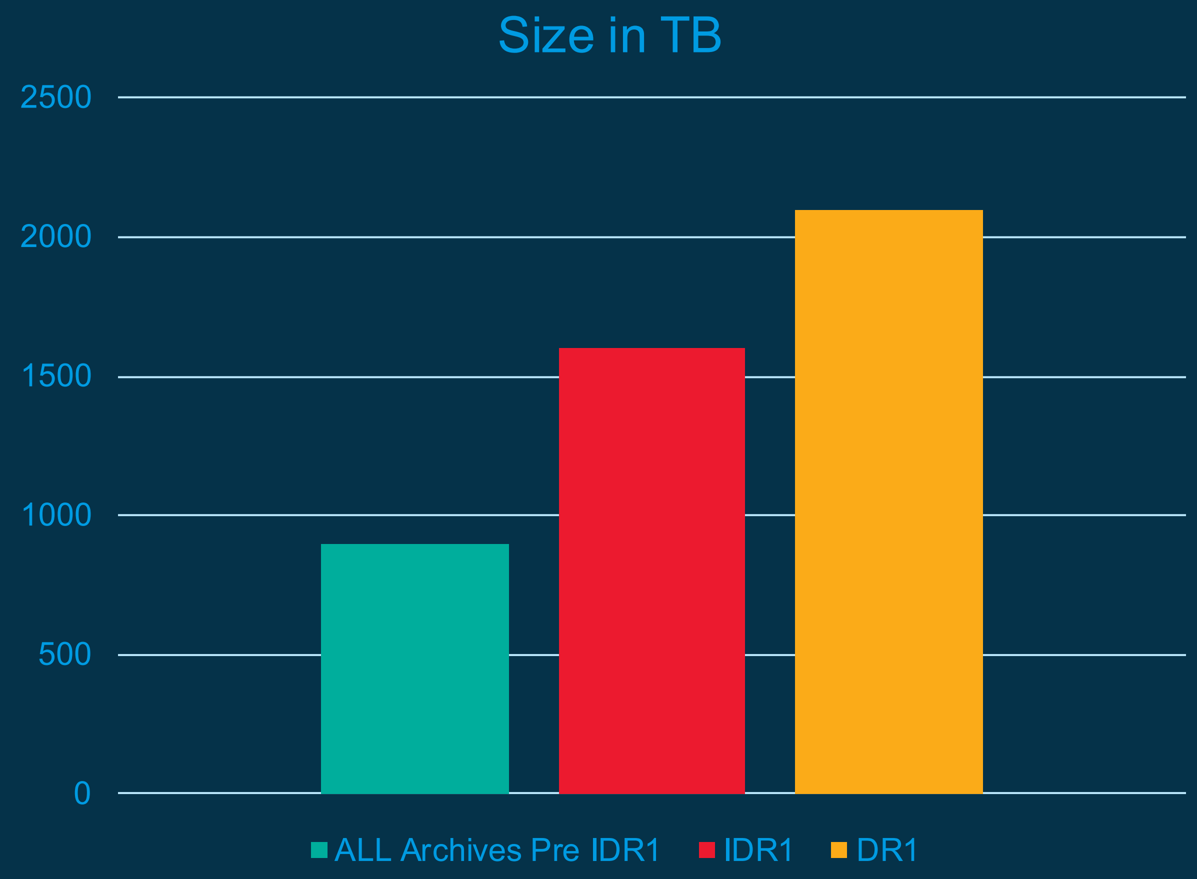
Next challenges

Next challenges

- **Euclid DR1** – 2.1 PB
    - *expected Q4 2026*

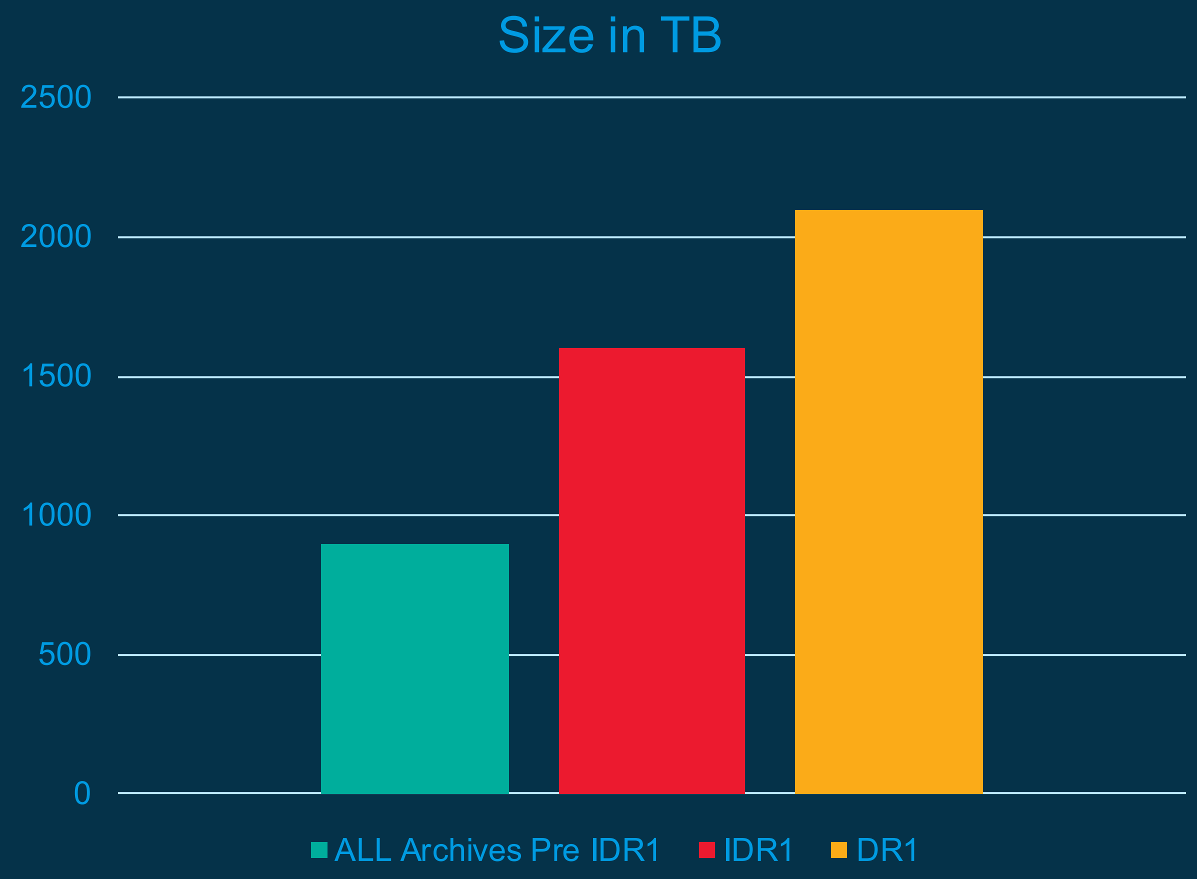
Next challenges

- **Euclid DR1** – 2.1 PB
  - *expected Q4 2026*



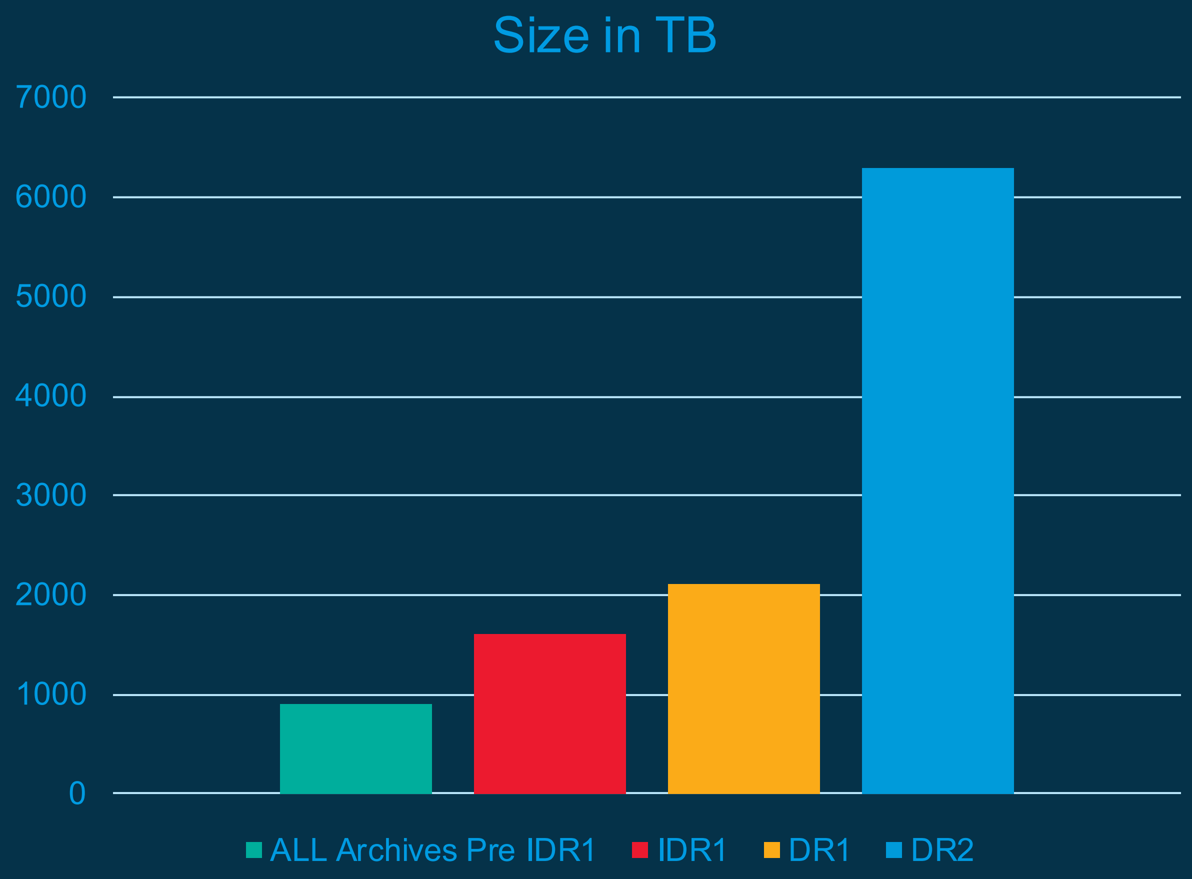
Next challenges

- **Euclid DR1** – 2.1 PB
  - *expected Q4 2026*
- **Euclid DR2** – 6.3 PB
  - *expected 2029*



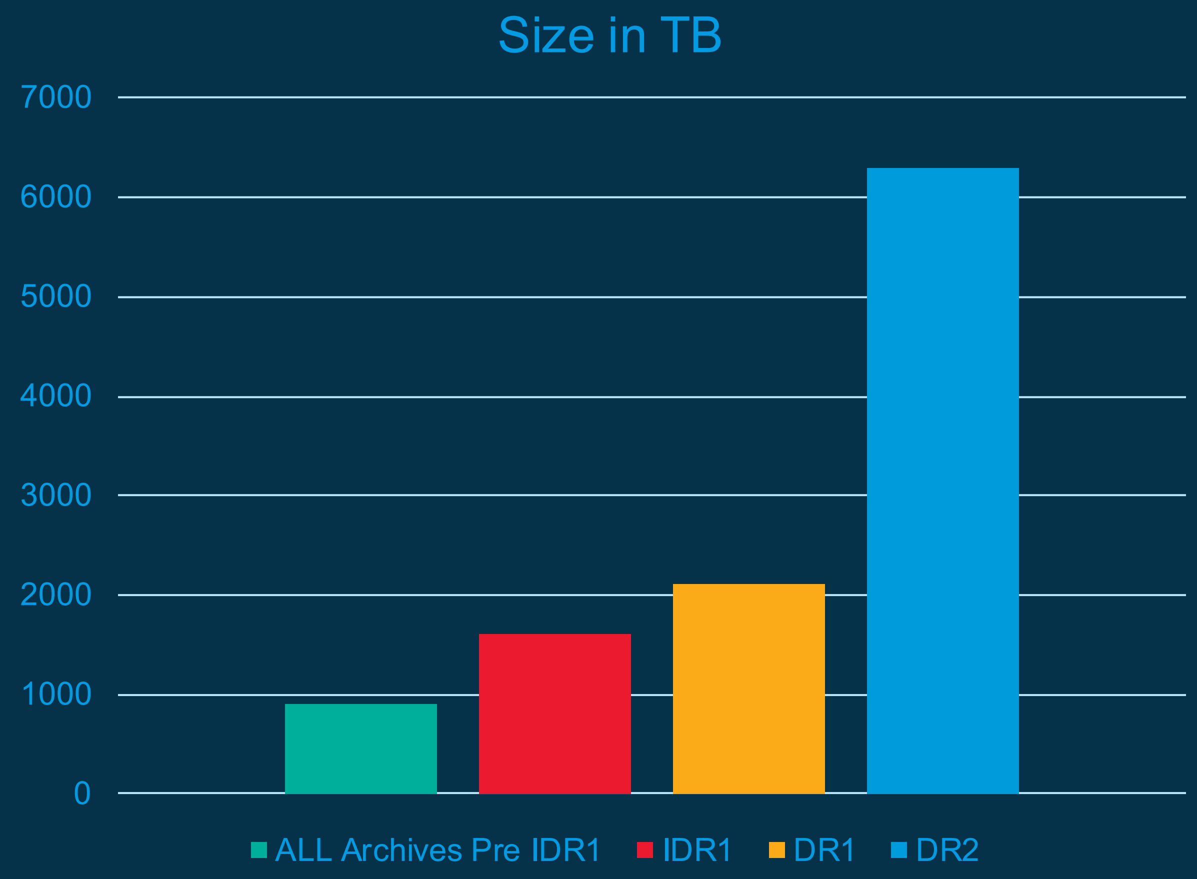
Next challenges

- **Euclid DR1** – 2.1 PB
  - *expected Q4 2026*
- **Euclid DR2** – 6.3 PB
  - *expected 2029*



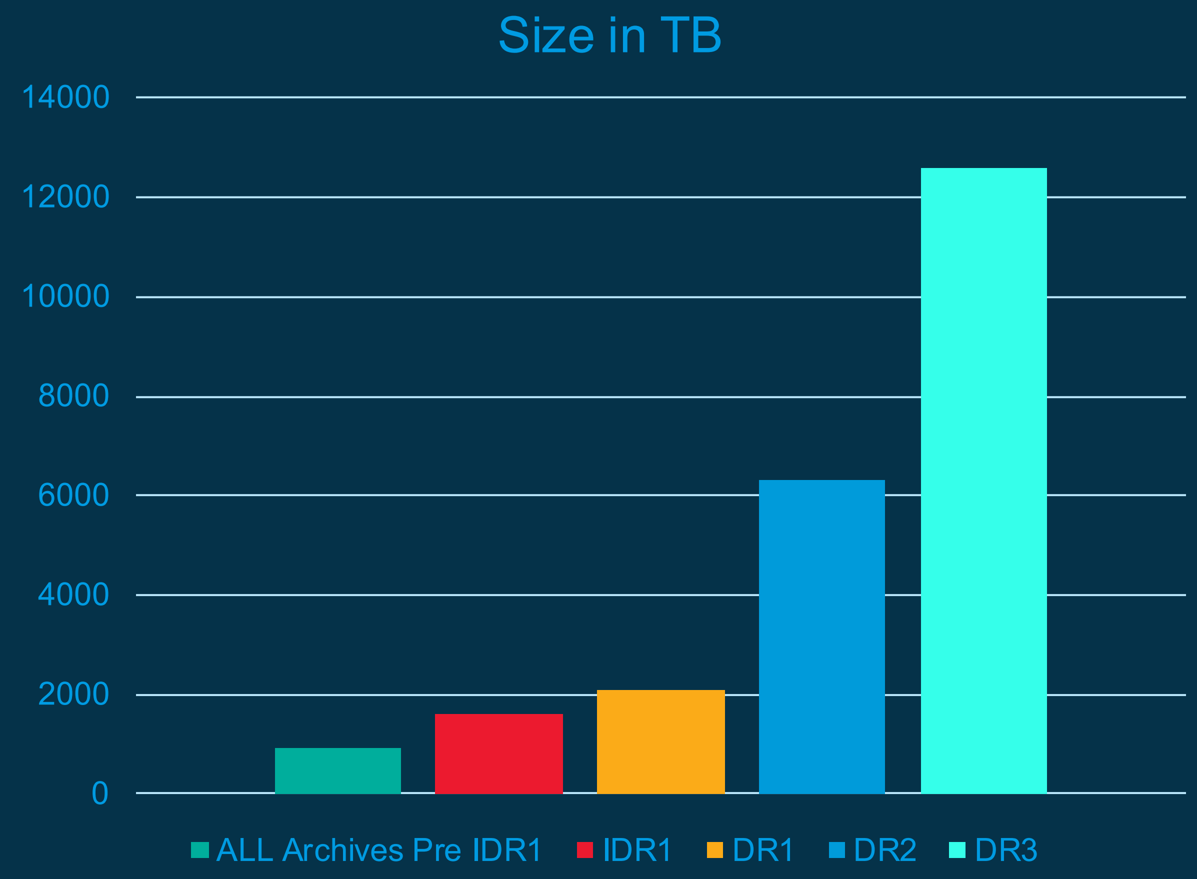
Next challenges

- **Euclid DR1** – 2.1 PB
  - *expected Q4 2026*
- **Euclid DR2** – 6.3 PB
  - *expected 2029*
- **Euclid DR3** – 12.6 PB
  - *expected 2031*



Next challenges

- **Euclid DR1** – 2.1 PB
  - *expected Q4 2026*
- **Euclid DR2** – 6.3 PB
  - *expected 2029*
- **Euclid DR3** – 12.6 PB
  - *expected 2031*



THANK YOU