ADMIRE Users Day

The Software Heritage Analytics Framework

Barbara Cantalupo

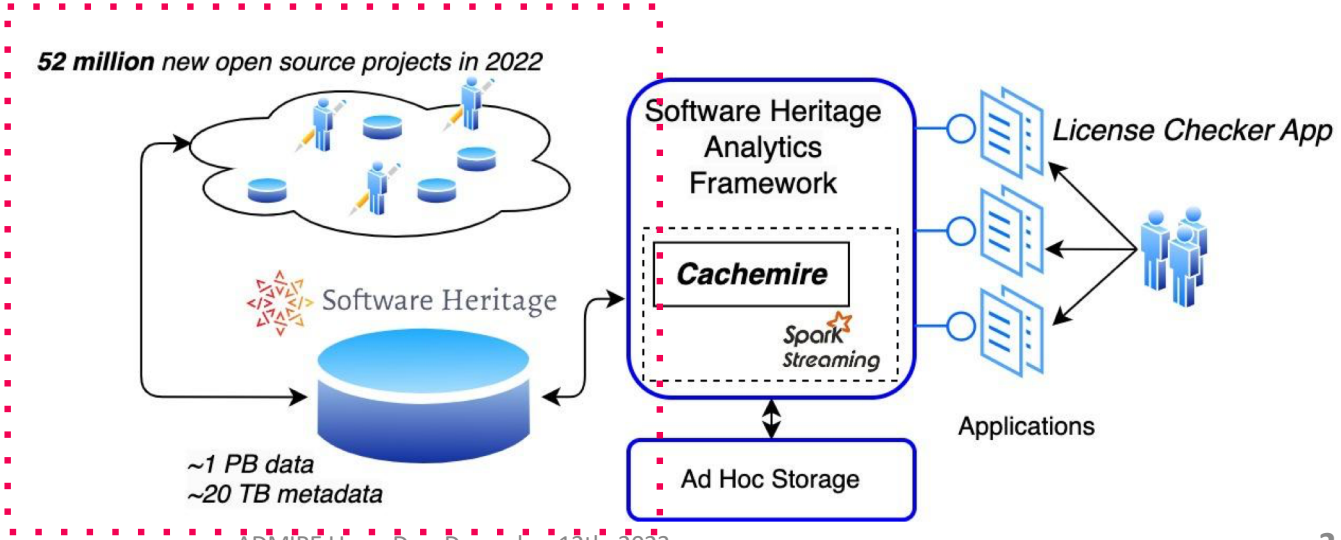
December 12th 2023

Barcelona Supercomputing Center

Grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1

Introduction

- An infrastructure designed and implemented within the ADMIRE project;
- Tailored to support applications for the analysis of Open-source code, exploiting the Software Heritage dataset;
- Created as a development environment to run user-defined applications.



What is open-source software?

- Typically, software is considered **open source** if:
  - It is available in **source code form without additional cost**, i.e., users can view and modify the code that comprises the software;
  - The source code can be **repurposed into other new software**, i.e., anyone can take the source code and distribute their own program from it.
- Open-source software is released through a specific kind of **license** that makes its source code **legally available to end-users**.

The value of open-source software

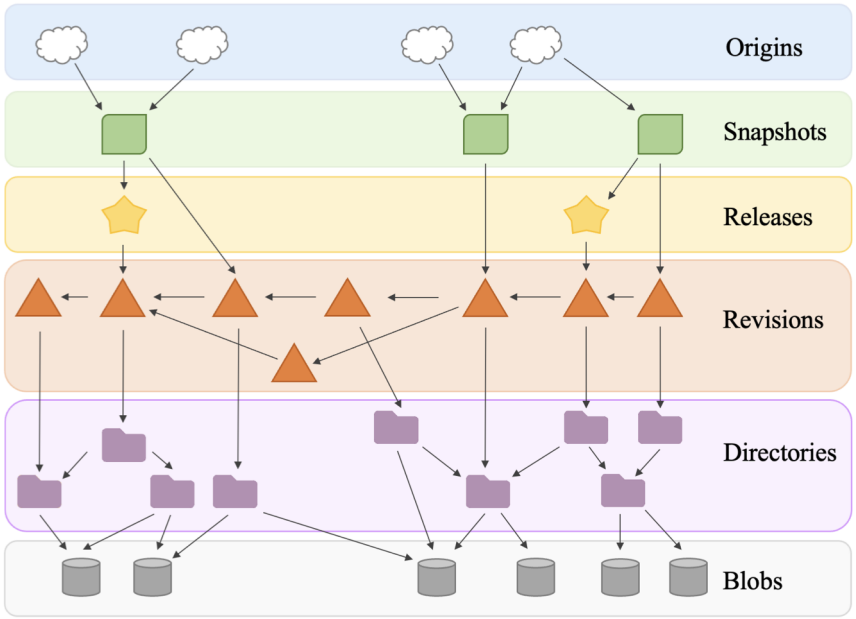
According to the 2022 GitHub report [1]

- ❏ Open source is the foundation of **more than 90%** of the world's software;
- ❏ In 2022 alone, developers started **52 million** new open source projects on GitHub.

The European Commission estimates that the use of open source software saves the European economy about **114 billion per year** directly in development costs [2].

[1] https://github.blog/2022-11-17-octoverse-2022-10-years-of-tracking-open-source/
[2] https://digital-strategy.ec.europa.eu/en/library/economic-and-social-impact-software-and-services-competitiveness-and-innovation

The Software Heritage dataset

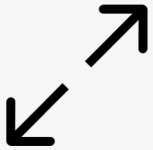


Software Heritage

The SWH archive:

- harvests source code from different sources and
- converts it into a single, universal data structure - an enormous **Merkle Directed Acyclic Graph** (DAG).

The Software Heritage dataset



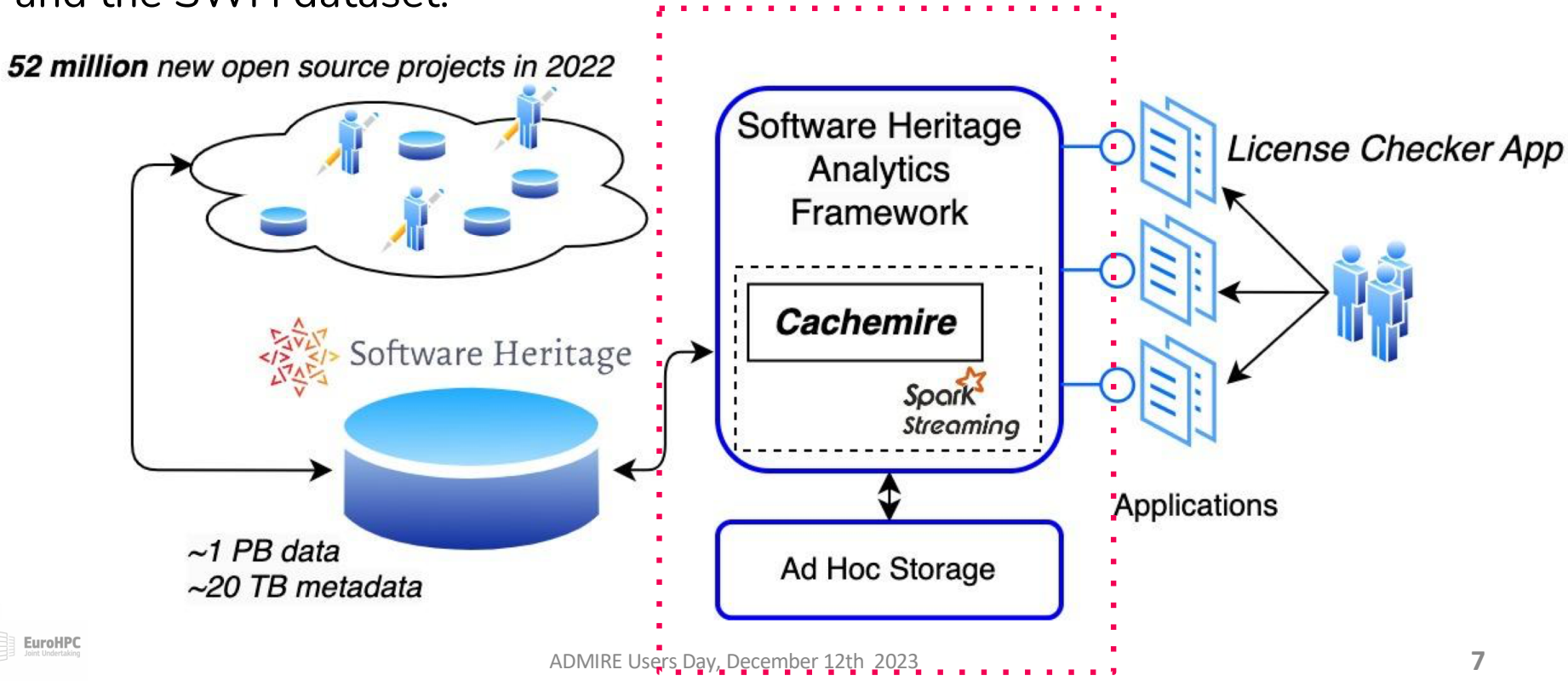
The SWH dataset is huge (almost 1PB in July 2023)!

Although being designed to archive and de-duplicate small files, **it can hardly efficiently feed** a BigData MapReduce (i.e., Spark) or an AI training/inference system

- Iterating across successive files of a query result might require traversing the 20TB metadata hash tree and jumping across 1PB of storage objects without any spatial locality.

The SWH Analytics Infrastructure

- **Objective**: bridging the performance gap between stream-based analytics and the SWH dataset.



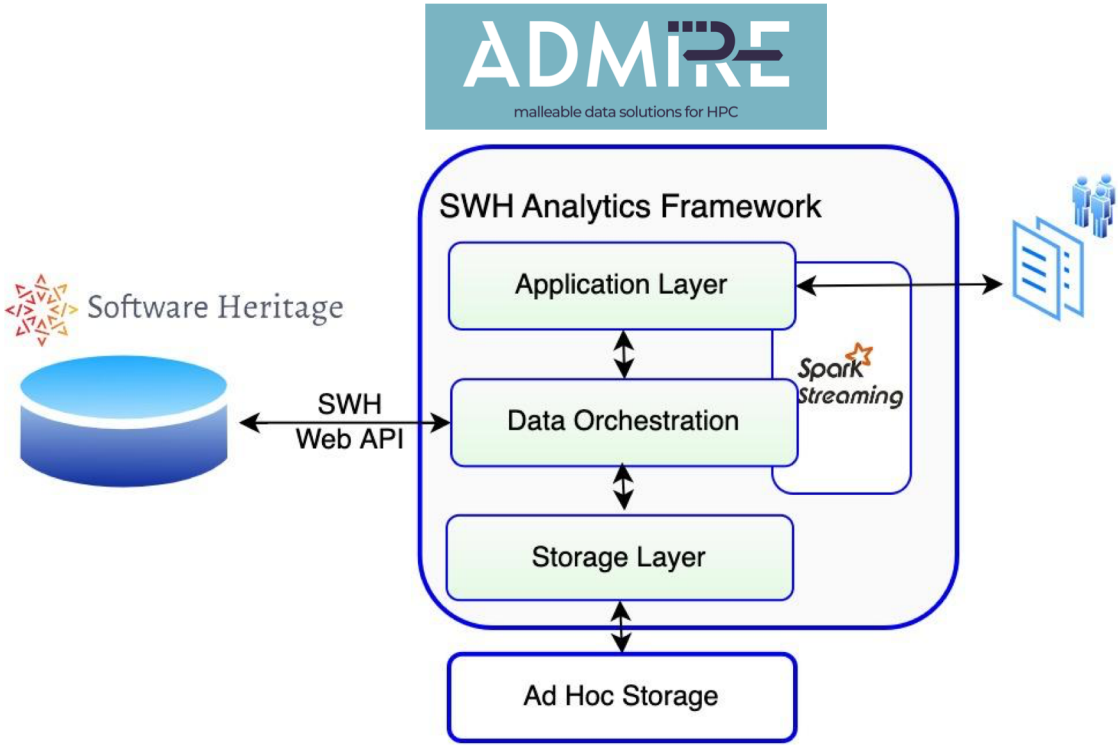
The SWH Analytics Infrastructure

❑ Architecture

The SWHA architecture is made up of **three main software layers**:

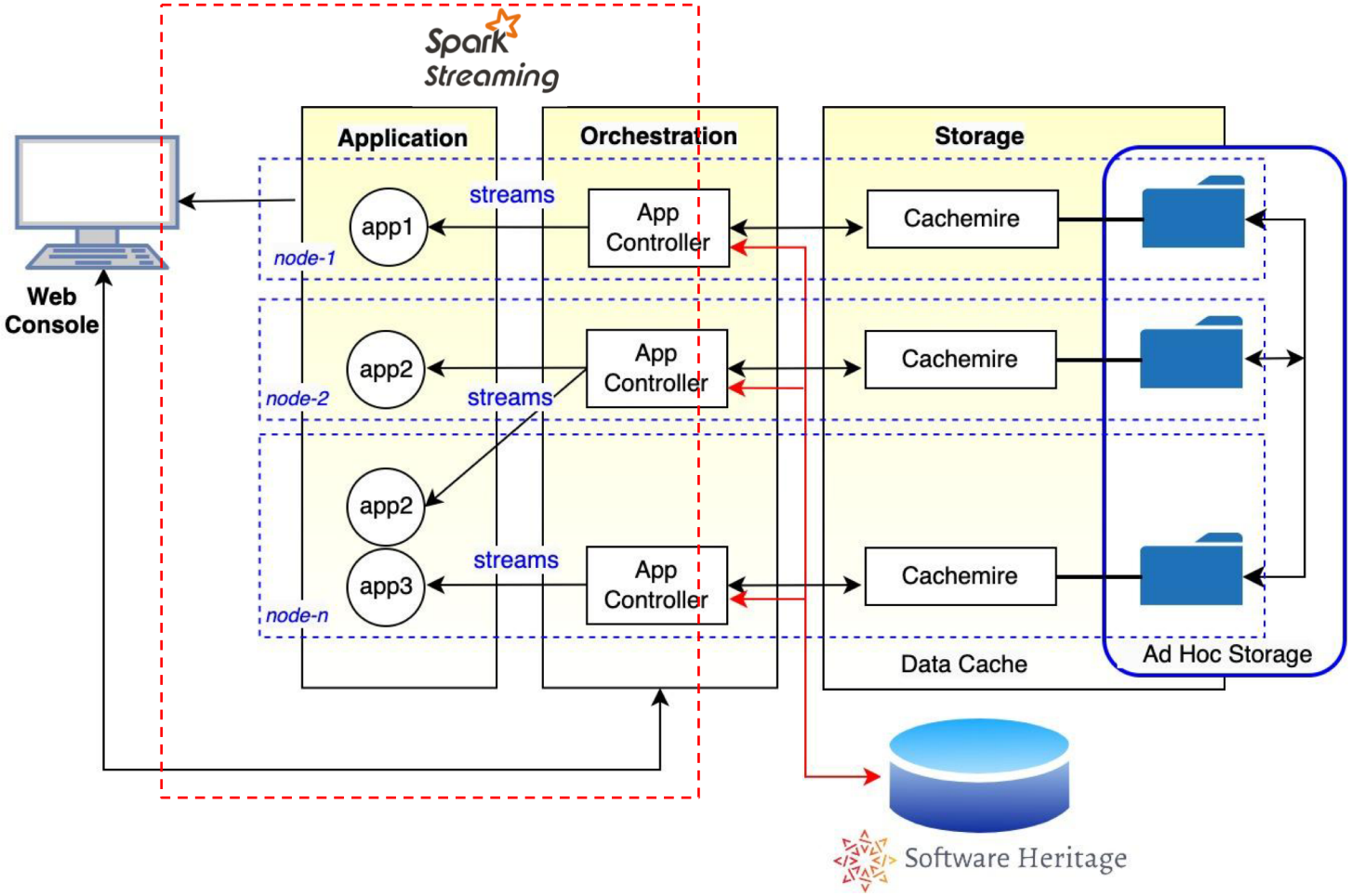
- ❑ storage,
- ❑ data orchestration, and
- ❑ application layers,

which cooperate in a parallel computing environment managed via the **Apache Spark Streaming Framework**.



The SWH-Analytics Data Flow

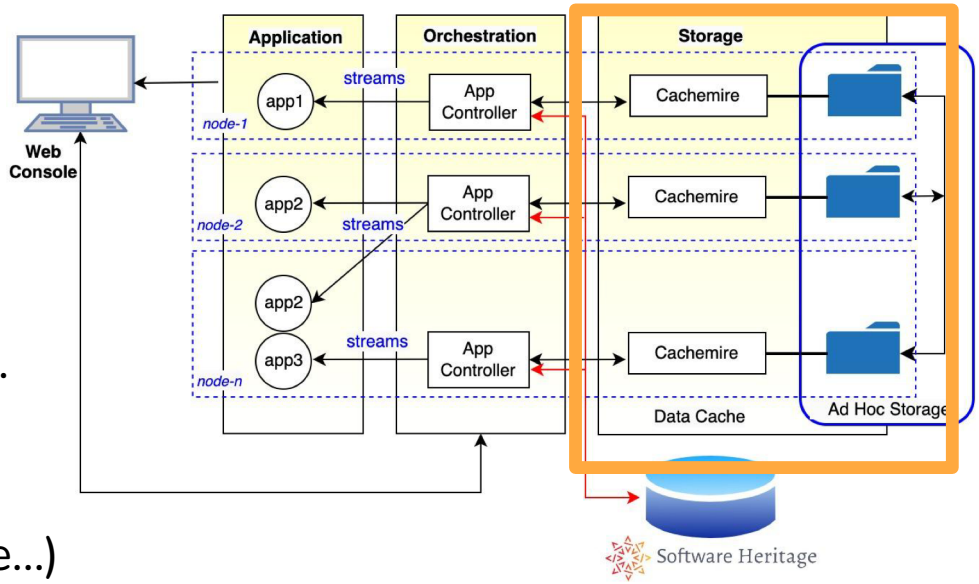
- Components



The Storage layer: Cachemire project cache

Functionalities

- ❑ Project cache implementing a distributed key-value storage.
  - key → project's id
  - value → project's source code
- ❑ Speeds up the data retrieval process.



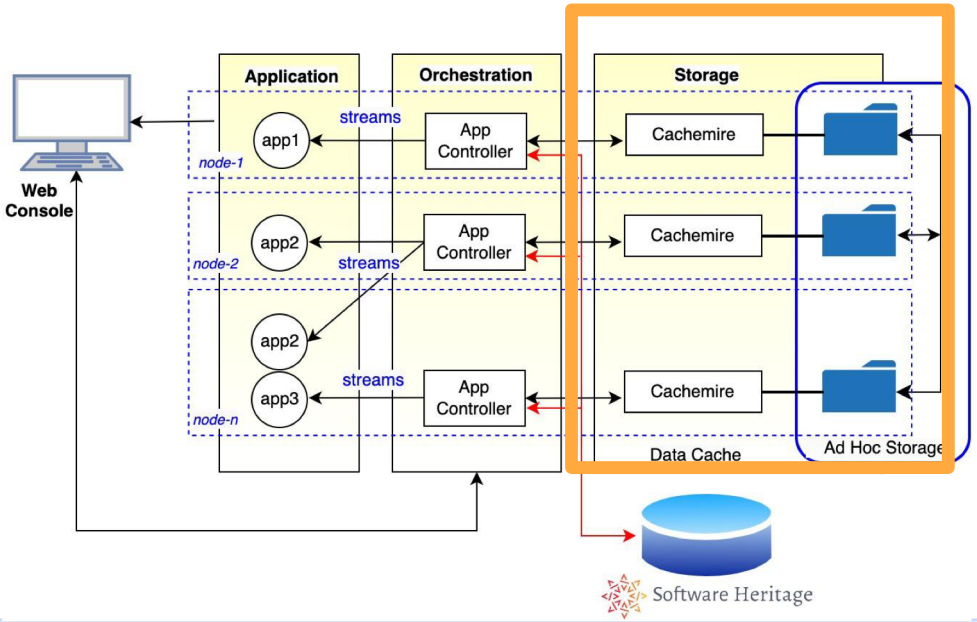
API and cache management

- ❑ PUT and GET functions (adding more...)
  Implementation relies on mechanisms offered by Posix-compliant file system primitives (file locking is not used).
- ❑ LRU (Least Recently Used) for cache replacement policy.
- ❑ External script monitors and maintains the size within a predefined threshold.

The Storage layer: Cachemire project cache

Dynamic scaling and workload balancing

- ❑ Ability to launch additional Cachemire instances dynamically.
  - Requires each node to offer a mounting point to a distributed file system.
- ❑ This storage infrastructure is seamlessly delivered by the ADMIRE framework
  - through specialized storage systems like GekkoFS or Hercules.

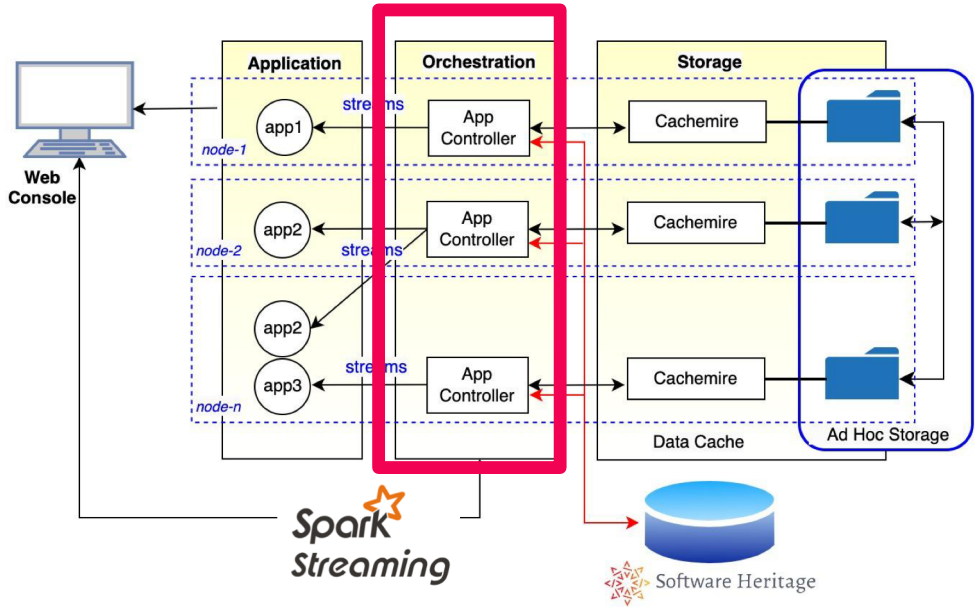


The synergy between a cache component optimized for use with such ad-hoc storage systems significantly augments the efficiency and reduces the completion time of applications developed within the SWHA framework.

The Orchestration layer

Application Request Management

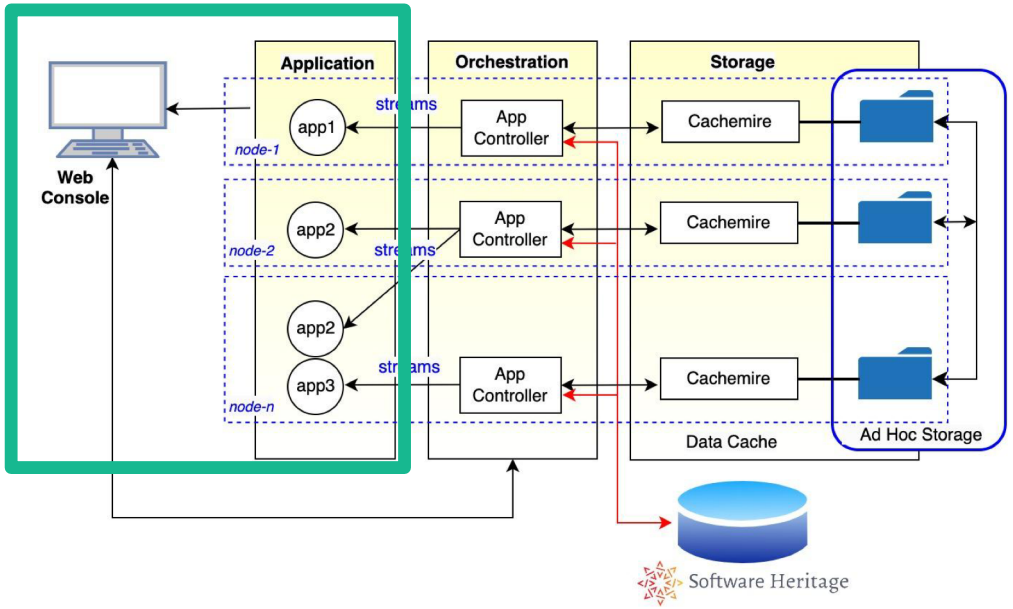
- Activates a pool of data stream generators (*app controllers*), which cooperate with Cachemire in a parallel computing environment managed via the Apache Spark Streaming Framework
- Uses the official SWH APIs to search and retrieve projects, which are then fed to Apache Spark workers.



The Application layer

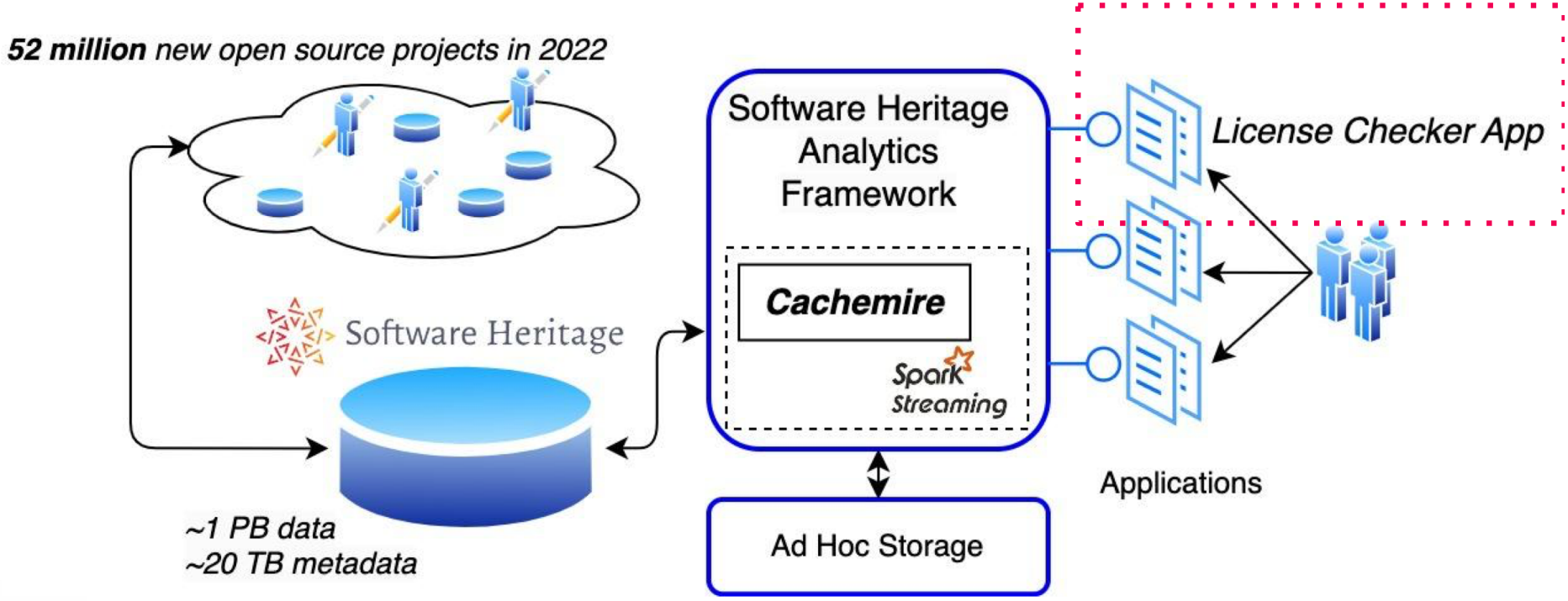
Application management

- Serves as the bridge for communication between an authenticated user and the SWHA system via a web-based console accessible through a web browser application.
- Allows the users to perform various actions, such as upload and execute their custom applications.



The License Checker Analytic App

- **Objective**: demonstrating how SWHA can be effectively exploited to study license usage in open-source software.



The importance of software licensing

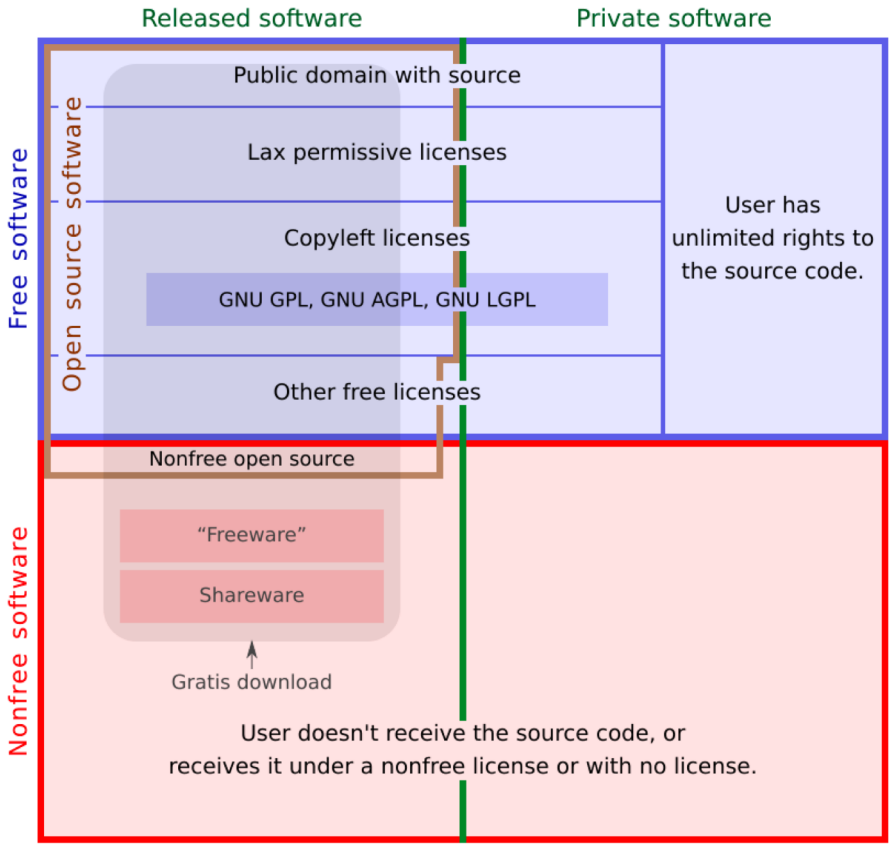
- Project licensing plays a critical role in companies as any violations of licenses can lead to substantial legal risks. Some examples:
  - **The mimemagic software library** [1]
    - Distributed under a (declared) MIT license, but incorporated the shared-mime-info library, distributed under the GPL license (a more restrictive license);
    - Impact: the mimemagic library was required by the Ruby on Rails web framework and affected 172 other packages, impacting an estimated 577,000 software repositories.
  - **The case of BusyBox** [2]
    - GPL license violation;
    - The use of BusyBox, which provides Unix utilities for embedded devices, in Monsoon Multimedia Inc.'s proprietary software resulted in a US court case.

[1] https://perma.cc/98DC-MWAU
[2] https://softwarefreedom.org/news/2007/sep/20/busybox/

A few details about software licensing

- The world of software licensing is complicated.
  - According to the ScanCode LicenseDB, there are 2174 different licenses [1].
  - The definition of a software license is standardized through the Software Package Data Exchange (SPDX) open standard.

[1] https://scancode-licensedb.aboutcode.org/



What we looked for

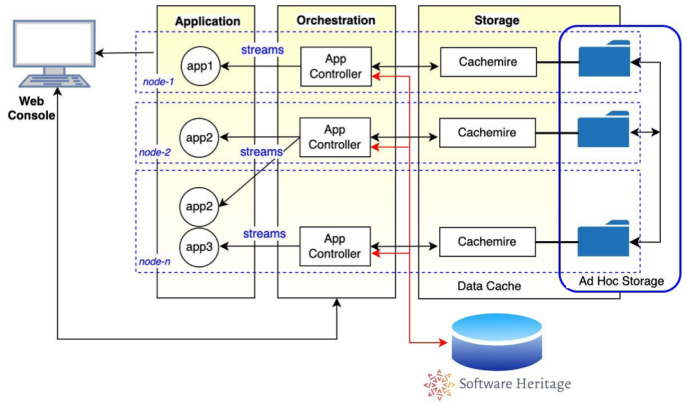
Given the extensive use of open-source software from public repositories in products, gaining a strategic understanding of license mismatches becomes essential.

RQ_1: Is there an **identifiable pattern** in the appearance of license inconsistencies or conflicts within open-source software projects?

RQ_2: Is the emergence of such problems **correlated** with the use of a given **programming language** or a specific **application domain**?

The license checker analytic app

❑ The application pipeline:



Dataset creation → Orchestration layer

License identification → Application's core logic

License compliance verification

The dataset creation phase

Dataset creation | License identification | License compliance verification

- Definition of the project set to analyze via an arbitrarily complex and customizable **recipe** to query the SWH archive via **web API**.
- **835** unique GitHub projects indexed by SWH
  - top 100 most starred projects
  - top 100 most forked projects
  - top 100 most-starred projects for each of the following programming languages:
    - C, Java, JavaScript, Julia, Kotlin, Python, R, and Rust.

The license identification phase

Dataset creation | License identification | License compliance verification

- For each streamed file, the application looks for
  - an **explicit license declaration** for the whole project (e.g., LICENSE.txt)
  - **in-code licenses** attached directly to files.
- The license is automatically detected with **ScanCode** to determine the restrictiveness of each license
  - github.com/nexB/scancode-toolkit

The license compliance verification phase

Dataset creation | License identification | License compliance verification

- **Finding inconsistencies and mismatches**
  - **Inconsistency** = use of **two different licenses** within the same project.
  - **Mismatch** = inconsistency involving licenses with varying degrees of restrictiveness.
  - **Conflicts = u**se of two different licenses **contradictory rights or incompatible obligations.** within the same project
- License compatibility matrix:
  - OSADL Open Source License Checklist project[1]

1] https://www.osadl.org/OSADL-Open-Source-License-Checklists.oss-compliance-lis[ts.0.html

The license checker analytic app - Results

RQ_1: Is there an identifiable pattern in the appearance of license inconsistencies or conflicts within open-source software projects?

Our analysis revealed a **positive correlation** between project **complexity**, indicated by its size, and the **number of licenses**, **mismatches**, and **conflicts**.

In line with previous literature, we identified that a substantial portion of **mismatches** occurs between **copyleft** and **strong copyleft licenses**, with **GPL** dependencies emerging as a **primary source of conflicts**.

The license checker analytic app - Results

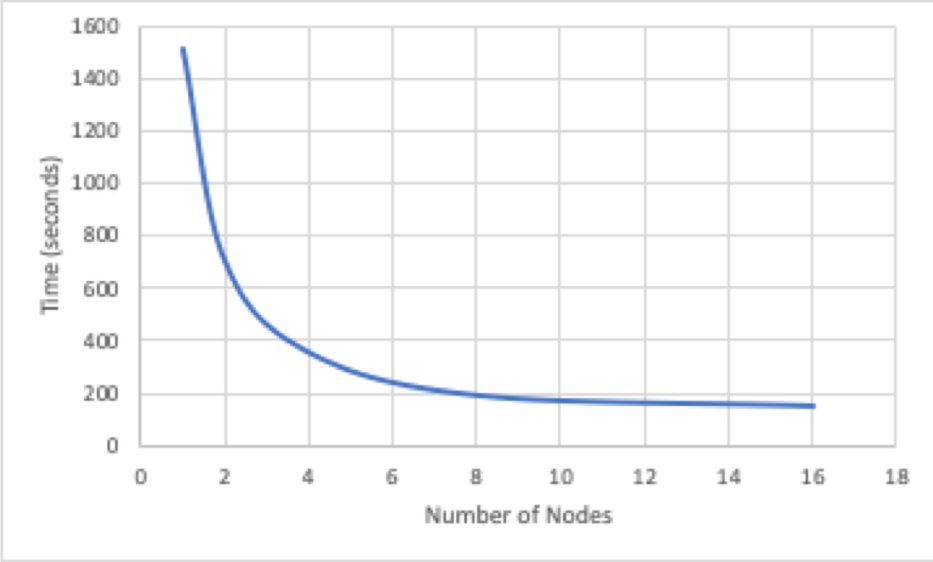
RQ_2: Is the emergence of such problems correlated with the use of a given programming language or a specific application domain?

A more in-depth examination of the relationships between the **programming language** employed and the **occurrences of licenses**, **mismatches**, and **conflicts** suggested the existence of **positive correlation**.

These patterns persisted when considering the **application domain**.

Application Performance - Scalability

- Tested on **HPC4AI**[1] (where ADMIRE components are being deployed and integrated)
  - 16 Broadwell nodes: OmniPath network, 2 x Intel(R) Xeon(R) CPU E5-2697 v4
  - Lustre parallel file system
- 100% Cache Hit

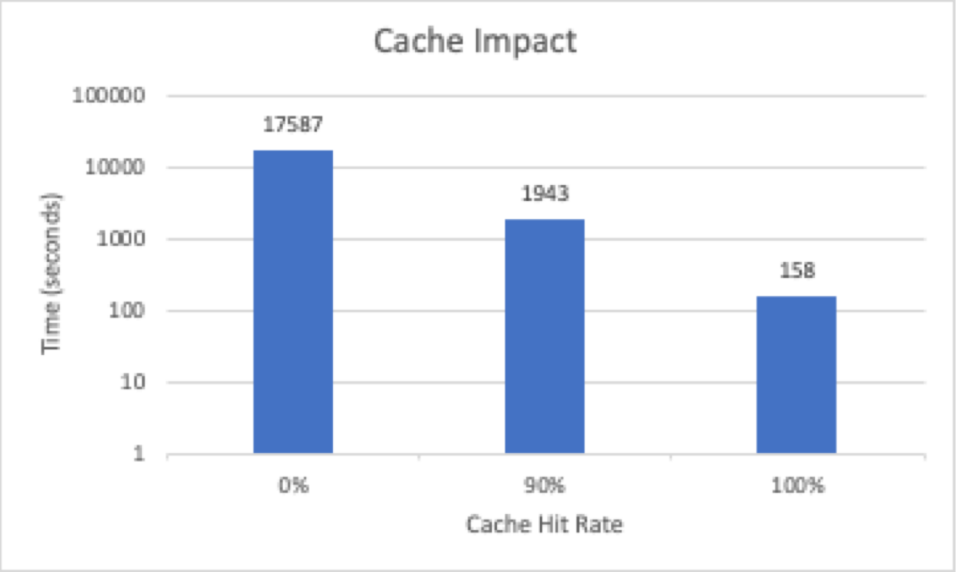


1] https://hpc4ai.unito.it/

Application Performance - Cache Impact

- Tested on **16 nodes**
  - L1: SWH projects already in cache
  - L2: Application-level info already in cache

- 0% Cache Hit:
  - All info need to be retrieved from SWH
- 90% Cache Hit:
  - 10% info are missing from cache
- 100% Chache Hit
  - All info already available in cache



Thank You!

https://github.com/alpha-unito/Software-Heritage-Analytics

barbara.cantalupo@unito.it

The team:

- Alessia Antelmi, Marco Aldinucci, Barbara Cantalupo - Depart Computer Science, University of Torino, Torino, Italy - HPC-KTT National Lab, CINI, Rome, Italy
- Massimo Torquati - Department of Computer Science, University of Pisa, Pisa, Italy - HPC-KTT National Lab, CINI, Rome, Italy
- Giacomo Corridori, Francesco Polzella, Gianmarco Spinatelli - Zerodivision systems Srl, Pisa, Italy