

---

# Interfaz genérica de acceso a datos masivos para entornos HPC y Big Data

**Javier García-Blas** y María S. Perez

*Universidad Carlos III de Madrid*

*fjblas@inf.uc3m.es*

---



**CABAHLA-CM: Convergencia Big dAta-Hpc: de Los  
sensores a las Aplicaciones**

# Motivation

---

- Nowadays, **most High-Performance Computing (HPC) applications use the POSIX I/O standard** interface to access data.
- There are **multiple problems** with this standard:
  - Handling complex data structures,
  - Lack of control over file system parameters,
  - Dependence on the operating system data layer.
- Currently-used data abstractions **are failing to adapt to the new scalability requirements and needs** of applications.

# Objectives

---

- Design and implementation of a **data container-based solution** for data-intensive applications.
- **High-level programming interface** that allows synergies between HPC and Big Data domains.
- Extend the POSIX interface to **hide the complexity of dealing with different storage patterns and sources.**

# Solution

---

- **New data model** based on a series of **activatable plugins** to facilitate its **extension to new storage systems**.
- Optimizations:
  - Memory management using C++ move operators.
  - Template-based generic programming.
  - Simultaneous I/O interference elimination system.

# Data containers

---

- **High-level abstractions:**

- Hide the complexity of dealing with different storage systems.
- Offer a standard and easy-to-use operation set.

- **Different types:**

- Depending on their use: input and output.
- Depending on the type of data: binary and text.

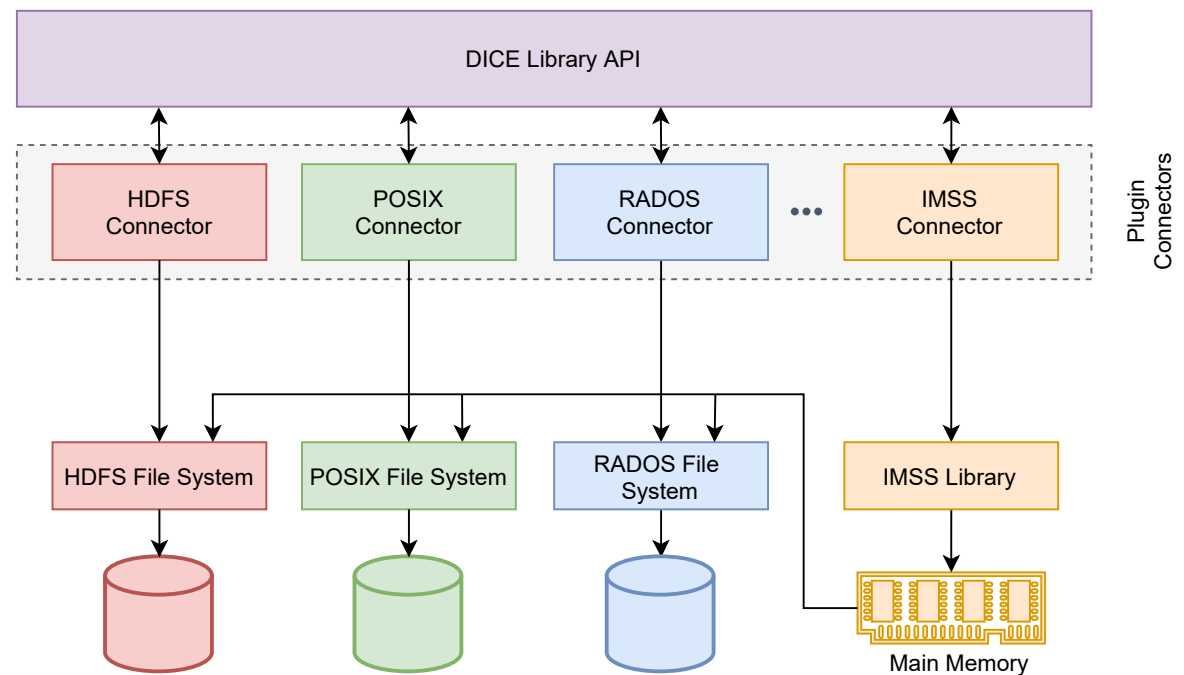
Storage System

`hdfs://users/data/twitter_data`

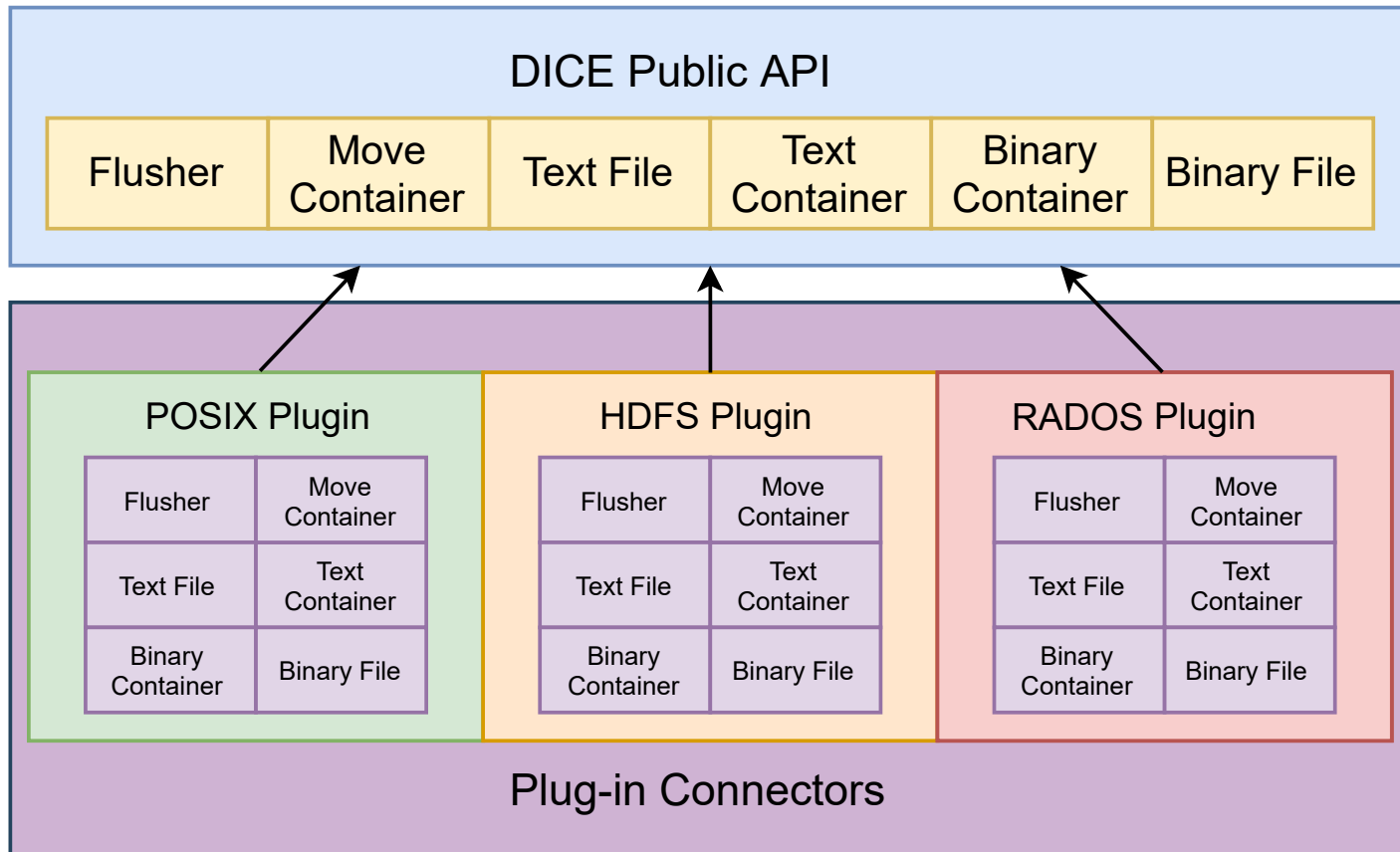
Dataset or collection path

# Back-end integration

- Plug-ins that enable access to the different storage systems.
- Four backends initially supported:
  - HDFS,
  - POSIX I/O,
  - RADOS (Ceph)
  - IMSS.



# Implementation



# Source code example

---

```
text_in_container mytest(argv[1], '\n');
```

```
flusher resultsw(argv[2], argv[3]);
```

```
regex rgx("\\#\\w+");
```

```
map<std::string,int> results;
```

```
for(auto line : mytest) {
```

```
    //Format the JSON
```

```
    auto j_tweet = json::parse(line);
```

```
    try{
```

```
        string text = j_tweet["text"];
```

```
        for( sregex_iterator it(text.begin(), text.end(), rgx), it_end; it != it_end; ++it ) results[(*it)[0]]++;
```

```
    }
```

```
    catch(const nlohmann::detail::exception& exc){}
```

```
}
```

```
//For each pair in the map
```

```
for (auto& kv: results) {
```

```
    std::string out = kv.first + ", " + std::to_string(kv.second) + "\n";
```

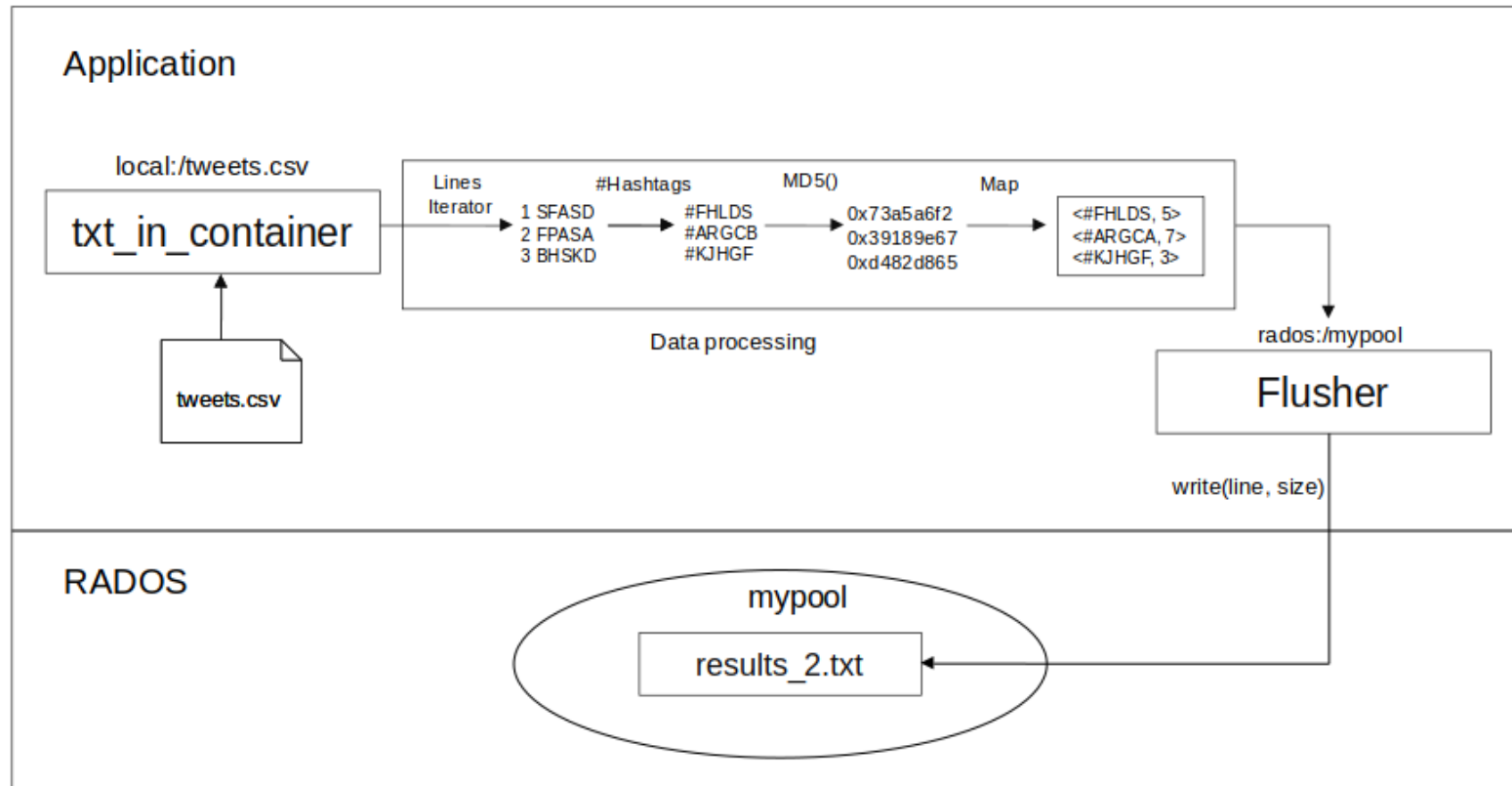
```
    //Erase the '#' character (which is the first one in the line) from the output before writing.
```

```
    resultsw << out.erase(0,1);
```

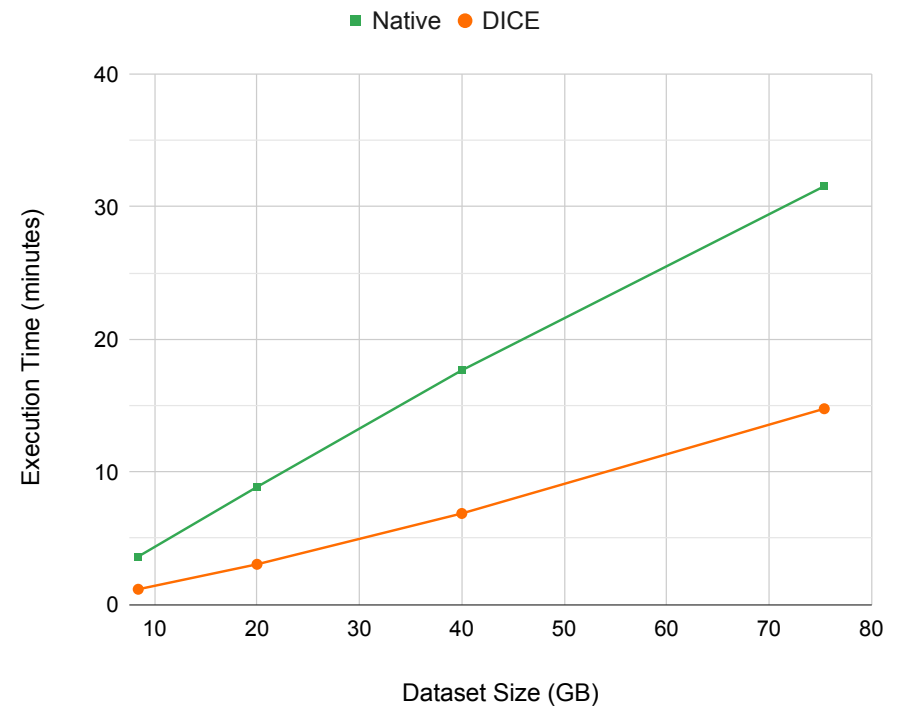
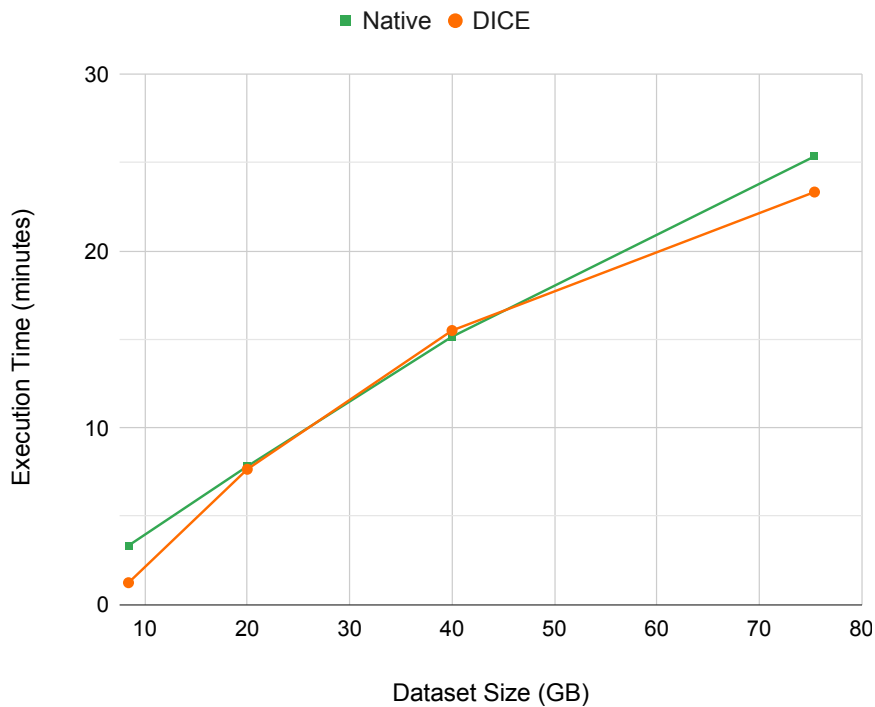
```
}
```



# Example



# Evaluation



# Results

---

- DICE, a generic data access abstraction for unifying HPC and Big Data worlds.
- We take advantage of modern C++ features that enable a generic code implementation for different storage systems.
- The solution has proven to be efficient in both Big Data and HPC domains obtaining an overhead under the 2%.
- CABHALA:
  - New interface for the RADOS API.
  - Working progress on a joint publication.

---

# Interfaz genérica de acceso a datos masivos para entornos HPC y Big Data

**Javier García-Blas** y María S. Perez

*Universidad Carlos III de Madrid*

*fjblas@inf.uc3m.es*

---



**CABAHLA-CM: Convergencia Big dAta-Hpc: de Los  
sensores a las Aplicaciones**