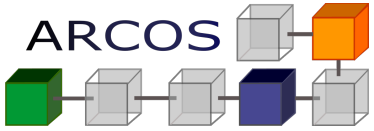


Técnicas de HPC aplicadas a procesamiento big data

José Rivadeneira

Félix García Carballeira

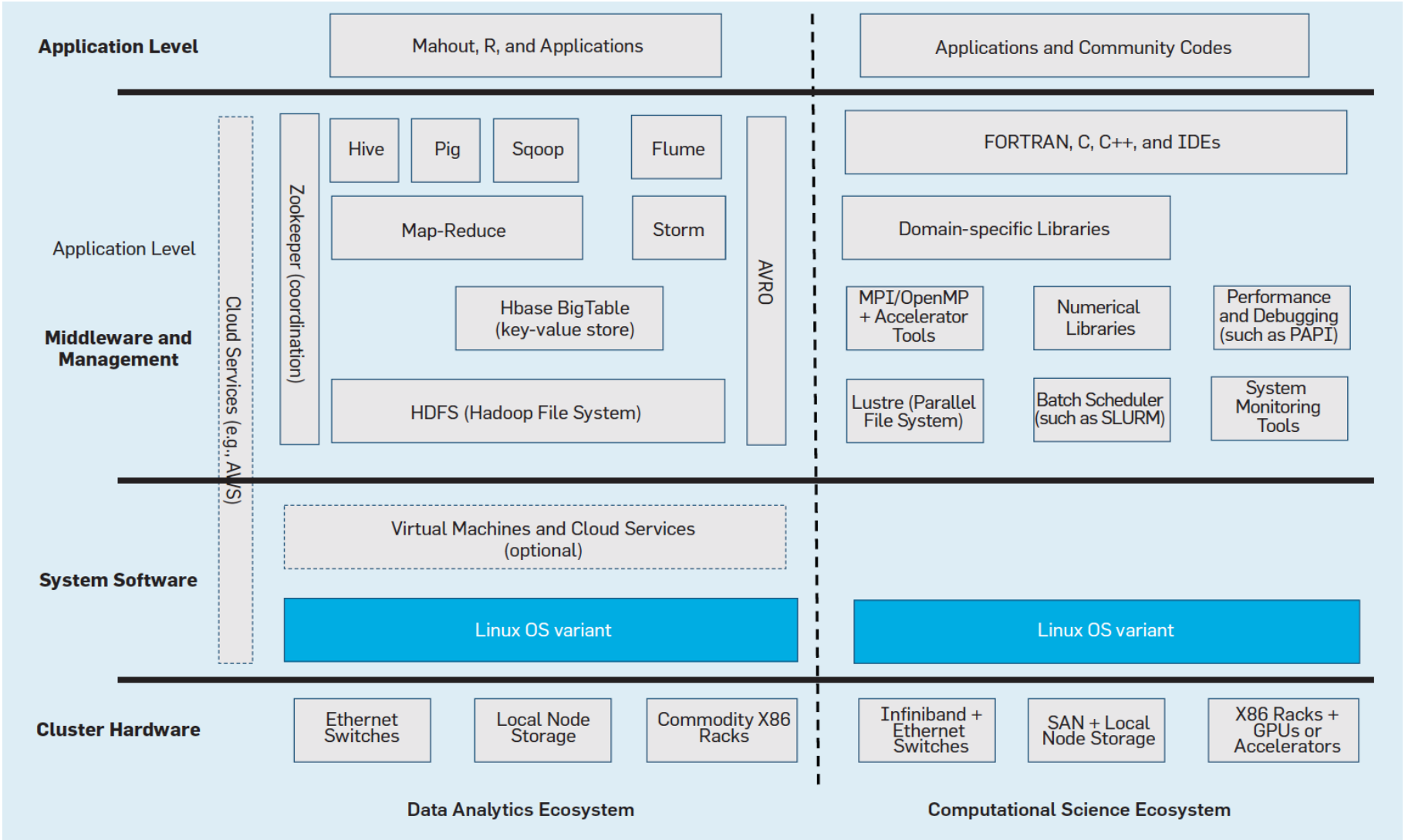
Jesús Carretero

Universidad complutense de Madrid

15 Junio 2022 Madrid - España

Índice

Background



Exascale computing and Big Data (Daniel A. Reed and Jack Dongarra)

Motivación

Problema: Los frameworks actuales de Map Reduce no funcionan de forma óptima en los entornos de HPC.

¿Que queremos mejorar?

- Las aplicaciones de Map Reduce diseñadas para HPC haciendo uso de técnicas utilizadas en Big Data.

Objetivos

Mejorar las aplicaciones de Map Reduce para el entorno de HPC utilizando la localidad de los datos.

- **O1:** Extender la interfaz de MPI-IO para poder hacer uso de la localidad de los datos.
- **O2:** Crear un nuevo conector para sistemas de ficheros de BigData dentro de MPI-IO.
- **O3:** Desarrollar una nueva propuesta de un framework de Map Reduce para HPC.
- **O4:** Desarrollar un sistema de almacenamiento ad-hoc de altas prestaciones para combinar y virtualizar almacenamiento HPC y BigData.

Extensión de MPI-IO

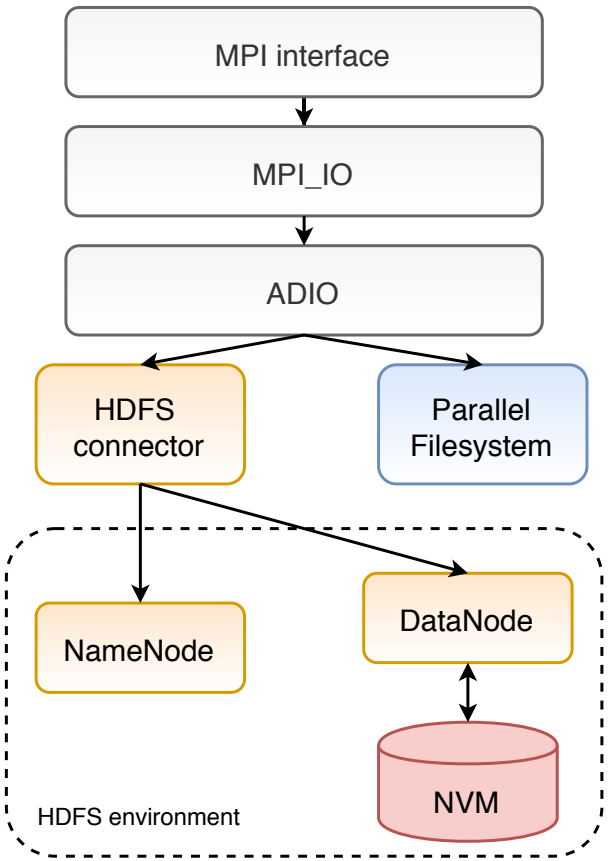
Con el objetivo de incluir la localidad de datos dentro de MPI-IO hemos propuesto dos nuevas funciones.

1. **MPIX_File_get_locality (MPI_File, MPI_Offset, MPI_Offset, char****, int *) :** Esta función devuelve la identidad del nodo(s) donde se encuentra almacenado el bloque indicado por argumento.

2. **MPIX_File_get_replication (MPI_File, int *) :** Esta función devuelve el número de réplicas almacenadas en el sistema de ficheros.

Añadiendo HDFS en MPI-IO

- Hemos diseñado un conector dentro de la interfaz abstracta para entrada y salida paralela (ADIO).
- Nuestra solución implementa el modelo un escritor, múltiples lectores.
- MPI_Info se pueden utilizar para personalizar como se crea un fichero y escribe un fichero.



Sobrecarga acceso HDFS desde MPI-IO

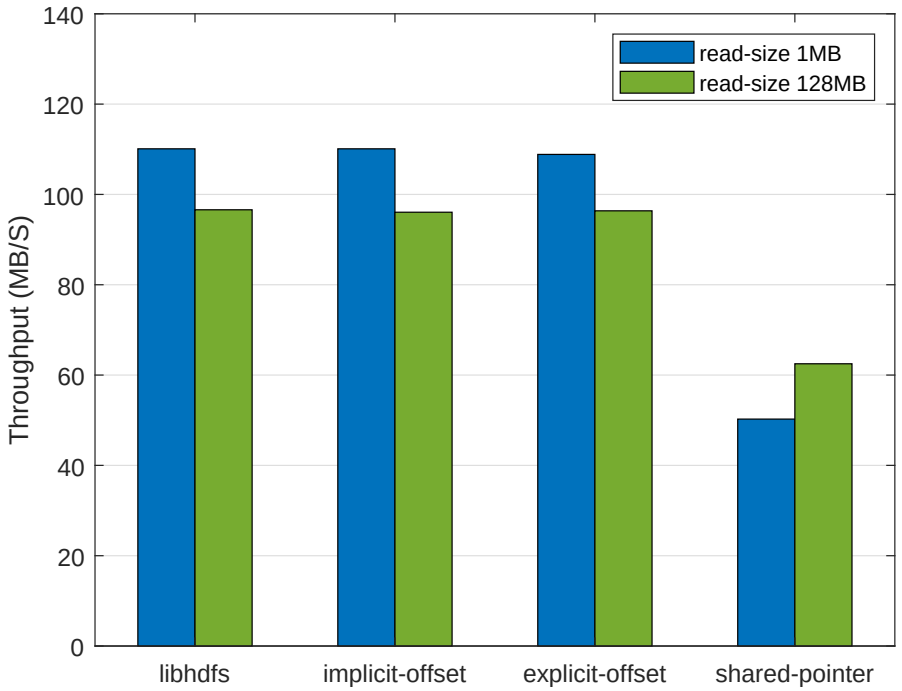


Figura: *MB/S lectura 32GB*

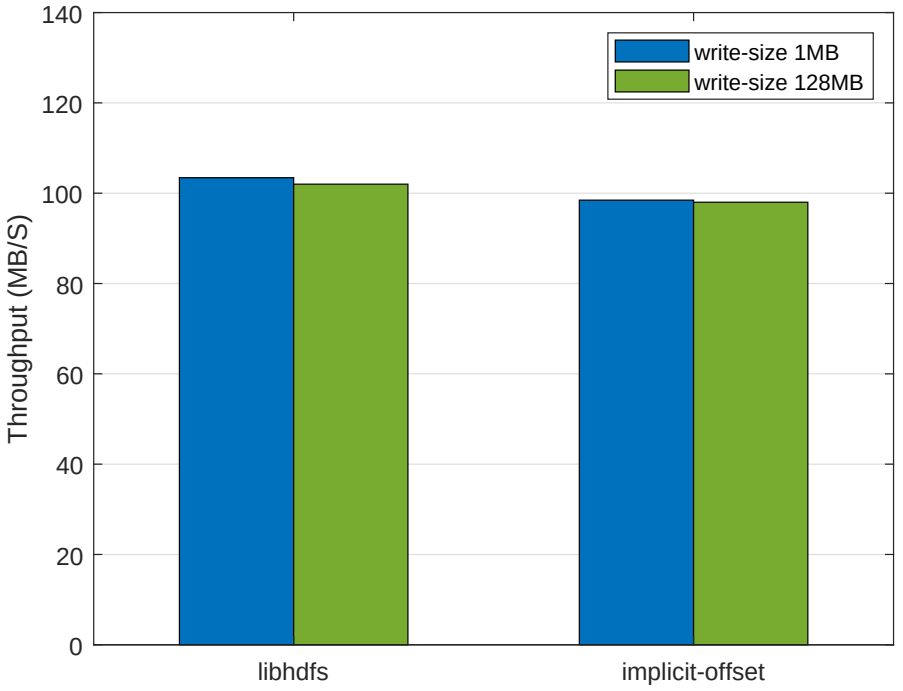
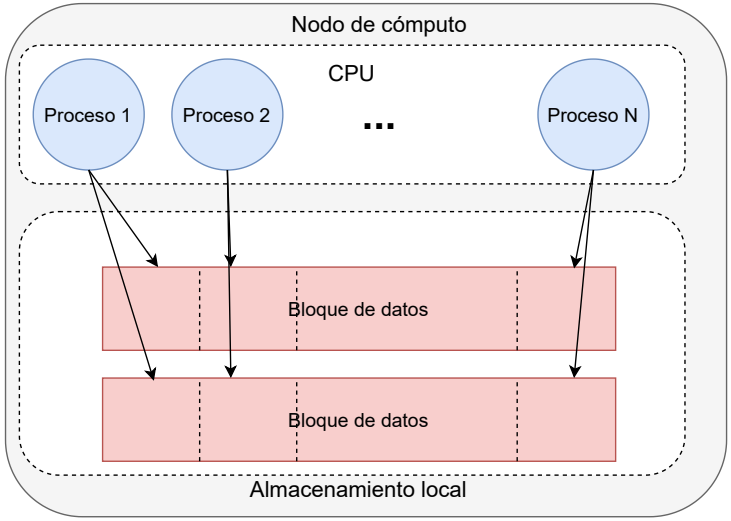


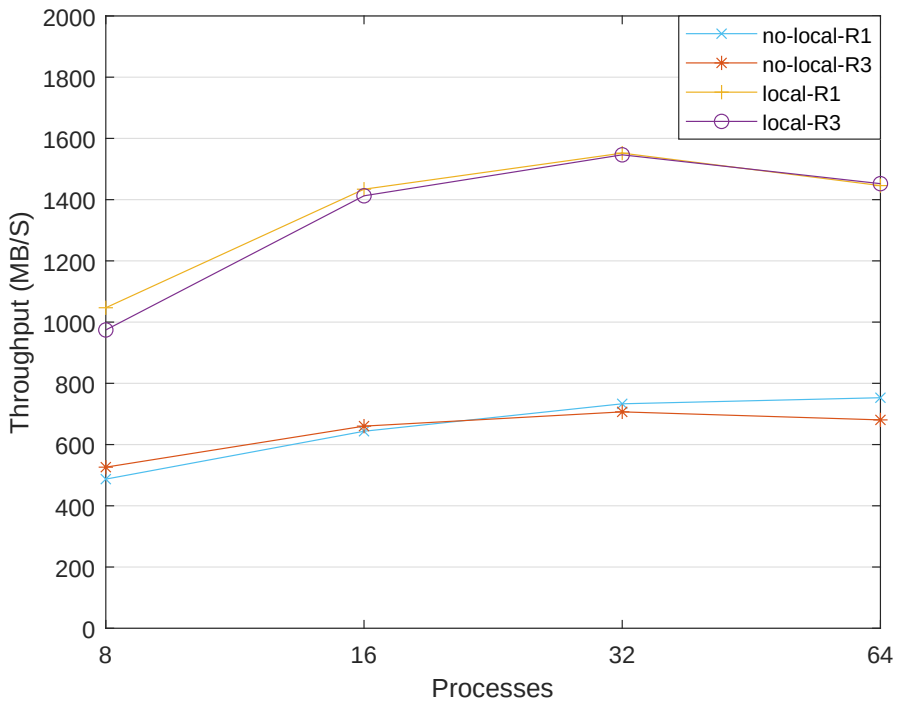
Figura: *MB/S escritura de 1GB*

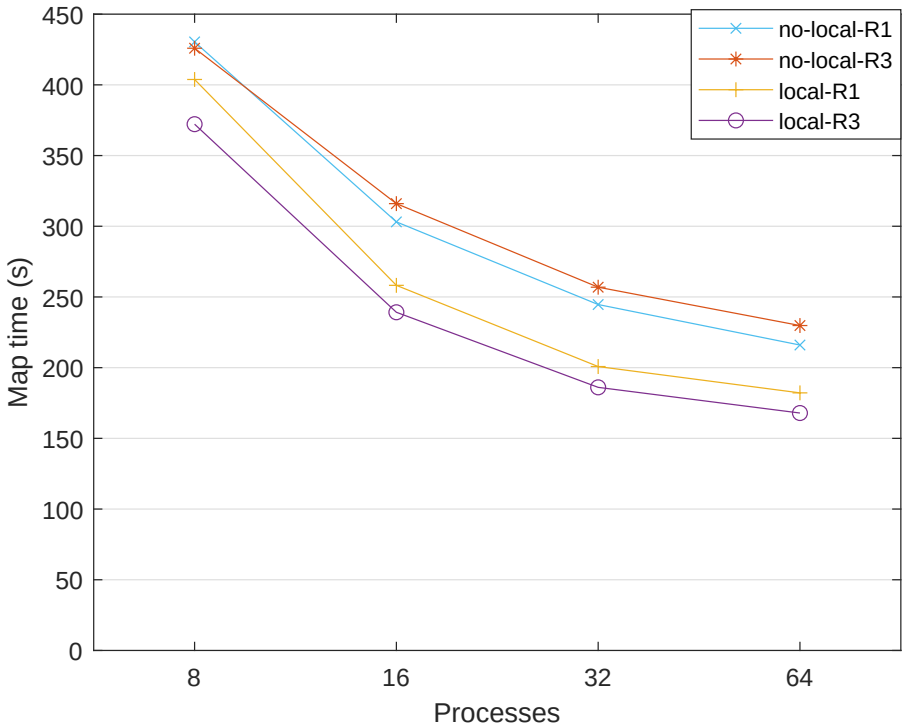
Evaluación de la propuesta de localidad de HDFS

- La propuesta inicial se evaluó con Mimir.
- La asignación de los bloques se hace por nodo.
- Si el bloque se encuentra almacenado en varios nodos se procesa por el nodo que tenga una menor carga de trabajo.



Rendimiento Mimir con localidad





Nuevo framework para Map Reduce escrito en c++

Caracterísiticas del nuevo framework

- Propuesta de alto rendimiento escrita en c++.
- Tolerante a fallos.
- Basado en threads en lugar de en procesos convencionales.
- Interfaz similar a Hadoop para que sea fácil de programar para los usuarios.

Modelo de programación

Map function

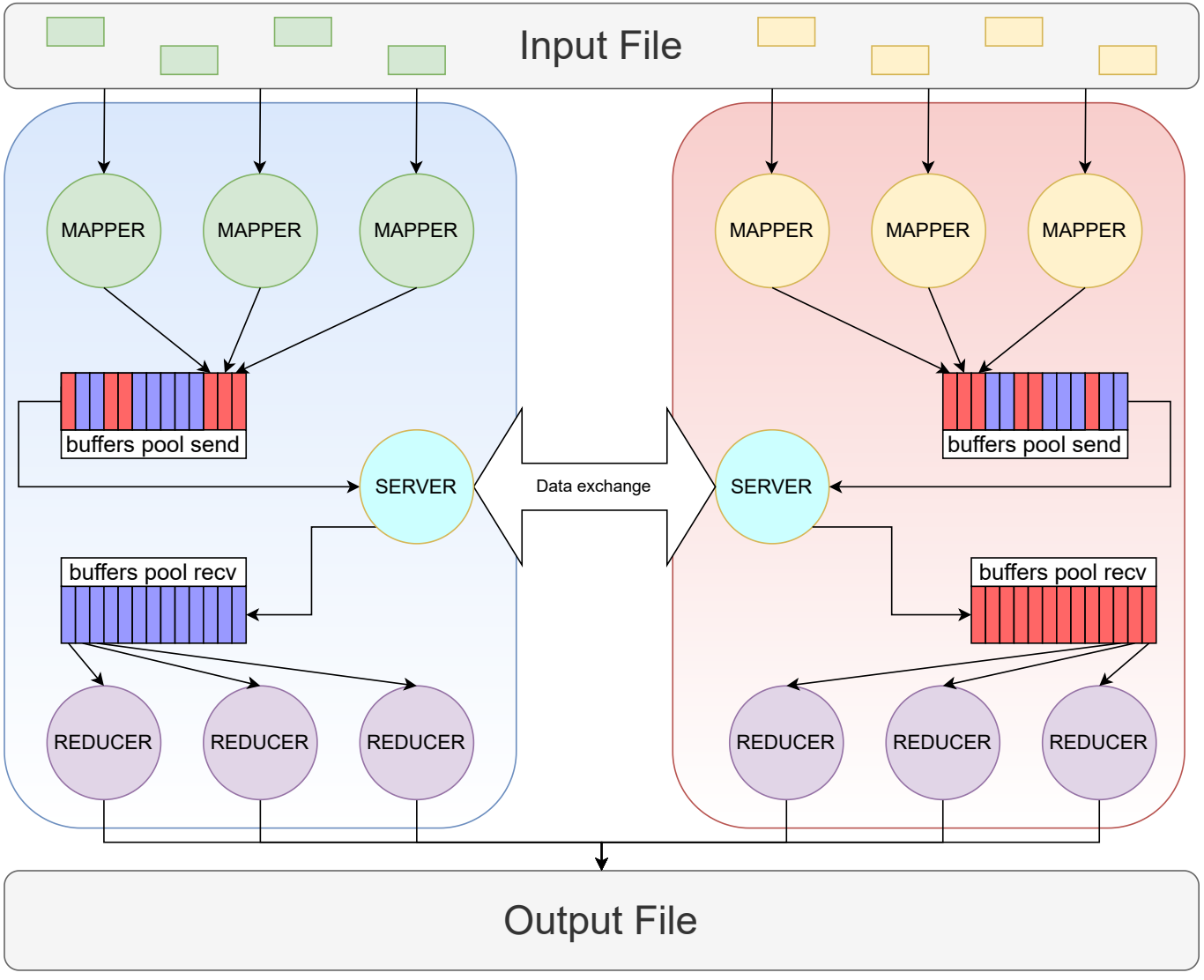
```
template<>
void mapper<char*,int>::map(){
    int one = 1;
    char *p;
    while((p = this->get_token(' '))){
        KeyValue<char*,int> arg(p,one);
        this->set_token(&arg);
    }
}
```

Modelo de programación

Reduce function

```
template<>
void reducer<char*,int>::reduce() {
    char key[MAX_KEY_LENGTH];
    int val = 0;
    int count = 0;
    get_key(key);
    while (get_values_by_key(key, &val)) {
        count = count + val;
    }
    emit_result(key,&count);
}
```

Diseño


Input File
MAPPER
MAPPER
MAPPER
MAPPER
MAPPER
MAPPER
buffers pool send
buffers pool send
SERVER
Data exchange
SERVER
buffers pool recv
buffers pool recv
REDUCER
REDUCER
REDUCER
REDUCER
REDUCER
REDUCER
Output File

Experimentos realizados

El cluster consta:

- 8 maquinas en total con una instalación de HDFS.
- Cada máquina dispone de 12 cores y 120GB de RAM.
- Cada máquina dispone para HDFS un disco duro de 2TB dedidcado.

Se han realizado dos tipos de experimentos;

1. Selección del tamaño de buffer óptimo.
2. Evaluación de las diferentes soluciones en el mismo entorno.

Selección tamaño del buffer de nuestra solución

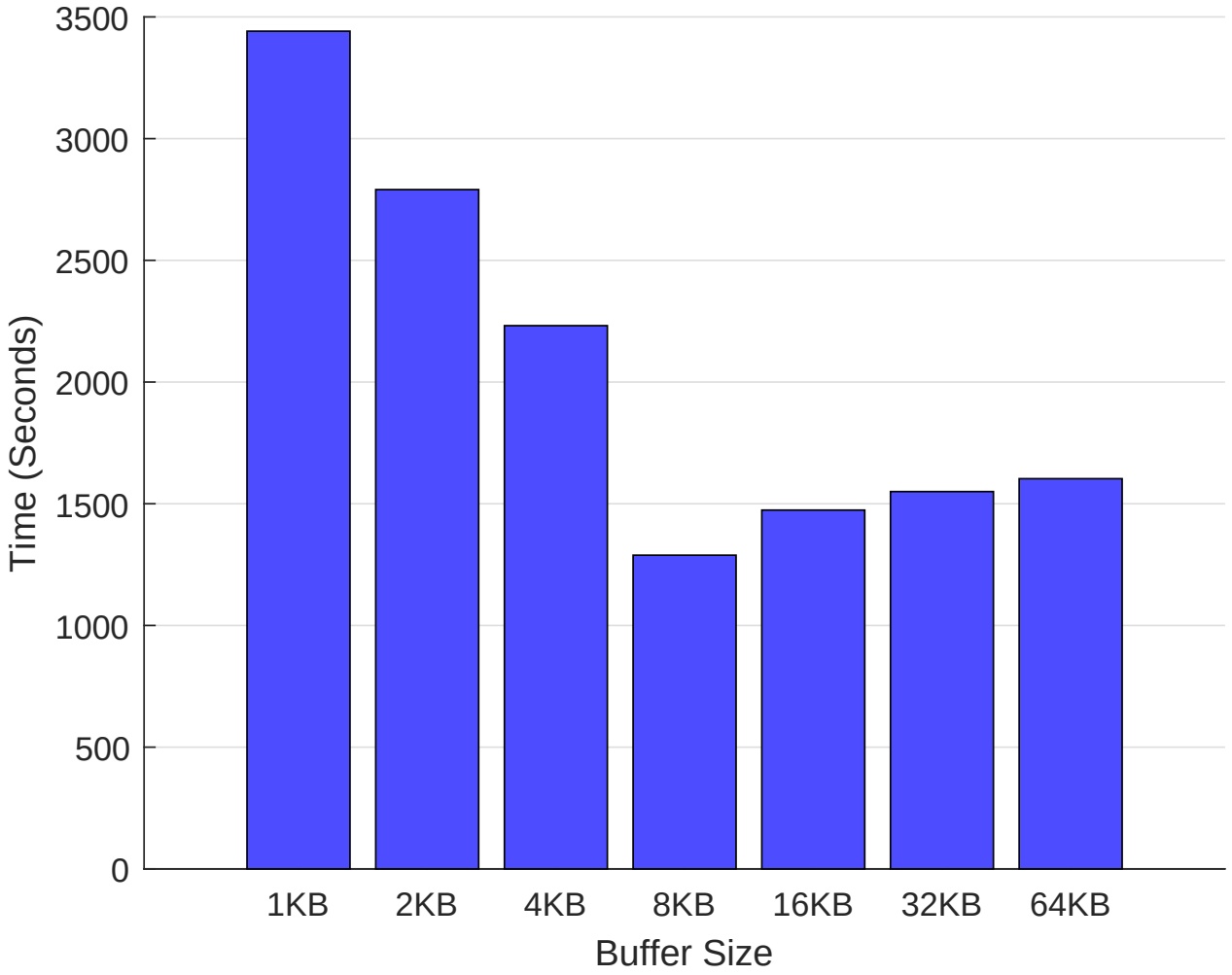


Figura: *Tiempo de ejecución según tamaño del buffer*

Evaluación diferentes soluciones

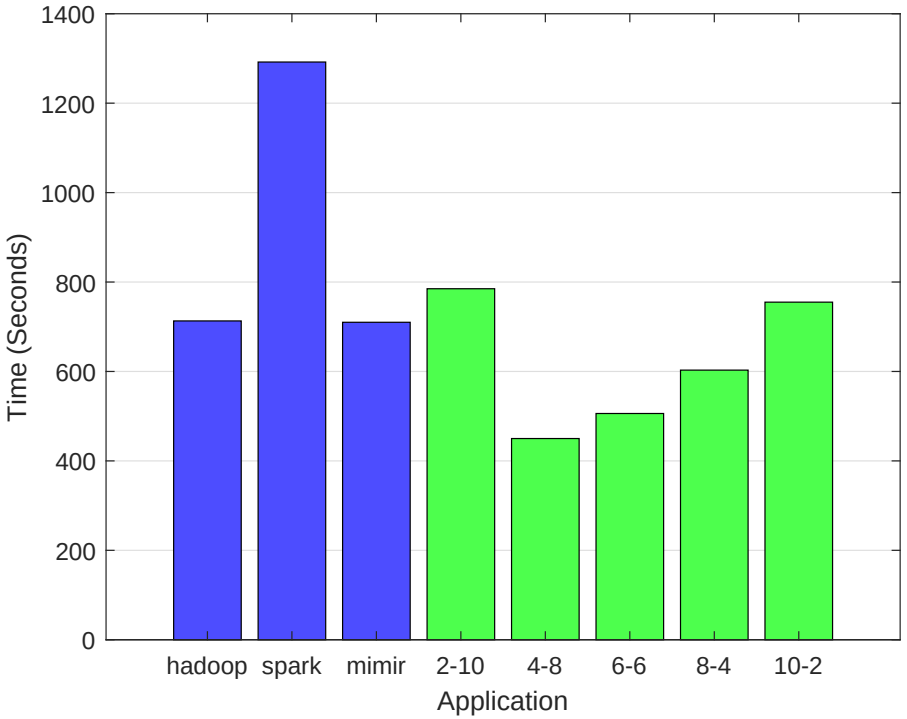


Figura: *Tiempos sin optimizaciones*

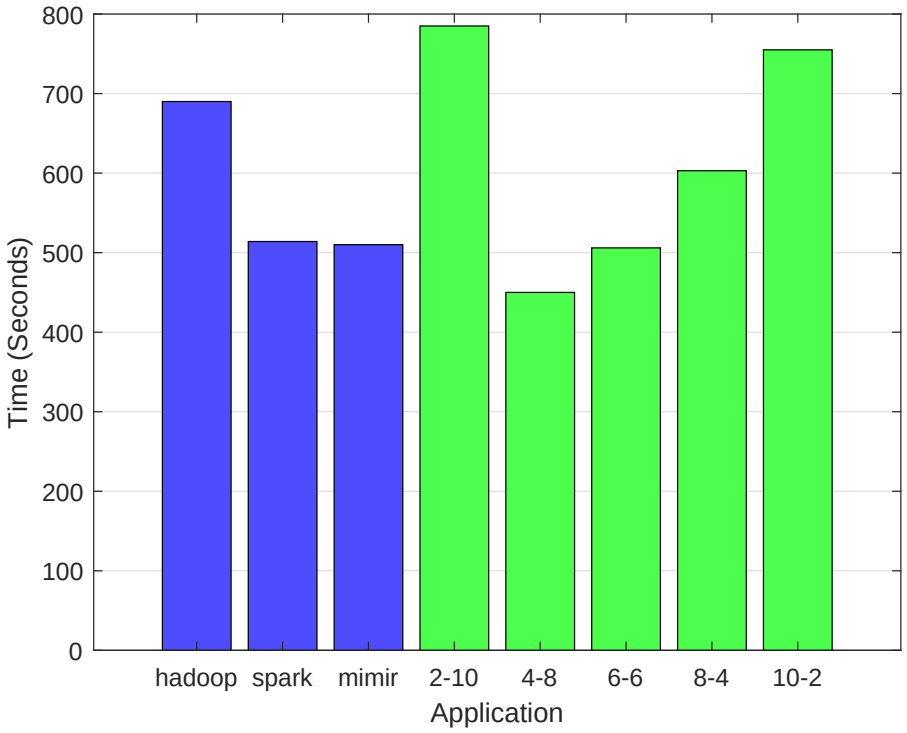


Figura: *Tiempos con optimizaciones*

Conclusiones

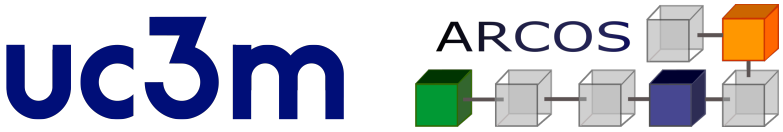
Conclusiones

1. El software actual para el procesado de datos no funciona correctamente en el entorno de HPC.
2. Incluir funciones de localidad de datos dentro de MPI puede ser una propuesta interesante para poder mejorar el rendimiento de las aplicaciones.
3. El uso de la localidad de los datos puede reducir el tiempo de ejecución de las aplicaciones de Map Reduce en HPC.
4. El uso de nuestra solución puede mejorar el rendimiento de las aplicaciones de Map Reduce en el entorno de HPC en el orden de un 11 % menos.

Trabajos futuros

Trabajos futuros

1. Hacer una propuesta para incluir la funcionalidad en el estandar de MPI.
2. Crear nuevos conectores dentro de ADIO para proveer información de la localidad a otros sistemas de ficheros.
3. Desarrollar un sistema de almacenamiento ad-hoc para combinar y virtualizar almacenamiento HPC en BigData.
4. Realizar nuevas evaluaciones en entornos con un mayor número de nodos.



Contacta con nosotros

José Rivadeneira
100303496@alumnos.uc3m.es

Félix García Carballeira
felix.garcia@uc3m.es

Jesús Carretero
jesus.carretero@uc3m.es