HPC I/ODC

I/O in the Data Center Workshop

PIOM-PX: A Framework for Modeling the I/O Behavior of Parallel Scientific Applications

Authors: Pilar Gomez-Sanchez, Sandra Mendez, Dolores Rexachs, Emilio Luque

June 2017

HPC4EAS Research Group

Universitat Autònoma de Barcelona

lrz Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften

escoles universitàries gimbernat

ADSCRITA A LA UAB

Introduction

- Parallel applications produce a **huge amount of data** that represents a challenge for modern I/O systems.

- Depending on the **I/O behavior of parallel applications** and the processing performed in **each layer of the I/O software stack,** the **performance obtained can differ significantly from the maximum performance expected**.

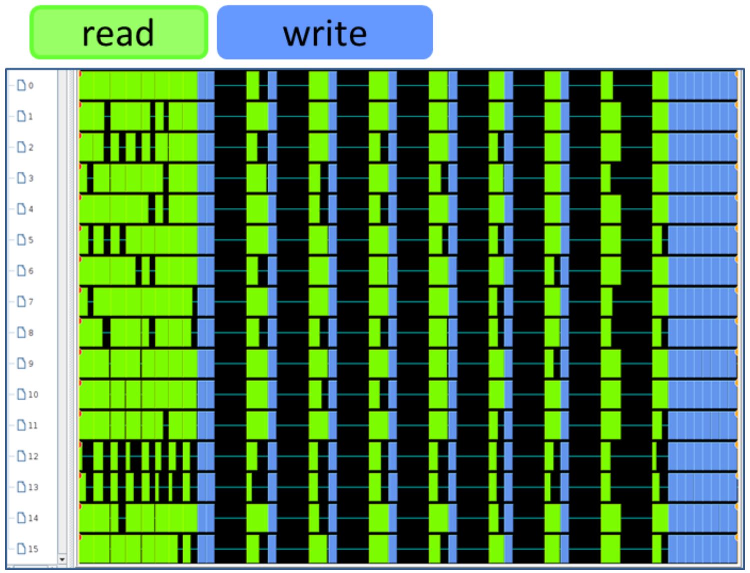
- Understanding I/O behavior **is fundamental** to evaluate the I/O performance of the HPC applications.

Introduction

- Most parallel application have a **repetitive behavior** when accessing a specific file.

- Due to the high cost of I/O operations, is normally intended to **reduce the number of accesses,** resulting in sporadic systematics bursts of I/O operations.

- The knowledge of I/O behavior allows us to determine the **I/O requirements of the application** and to **evaluate their impact** on different I/O configurations.



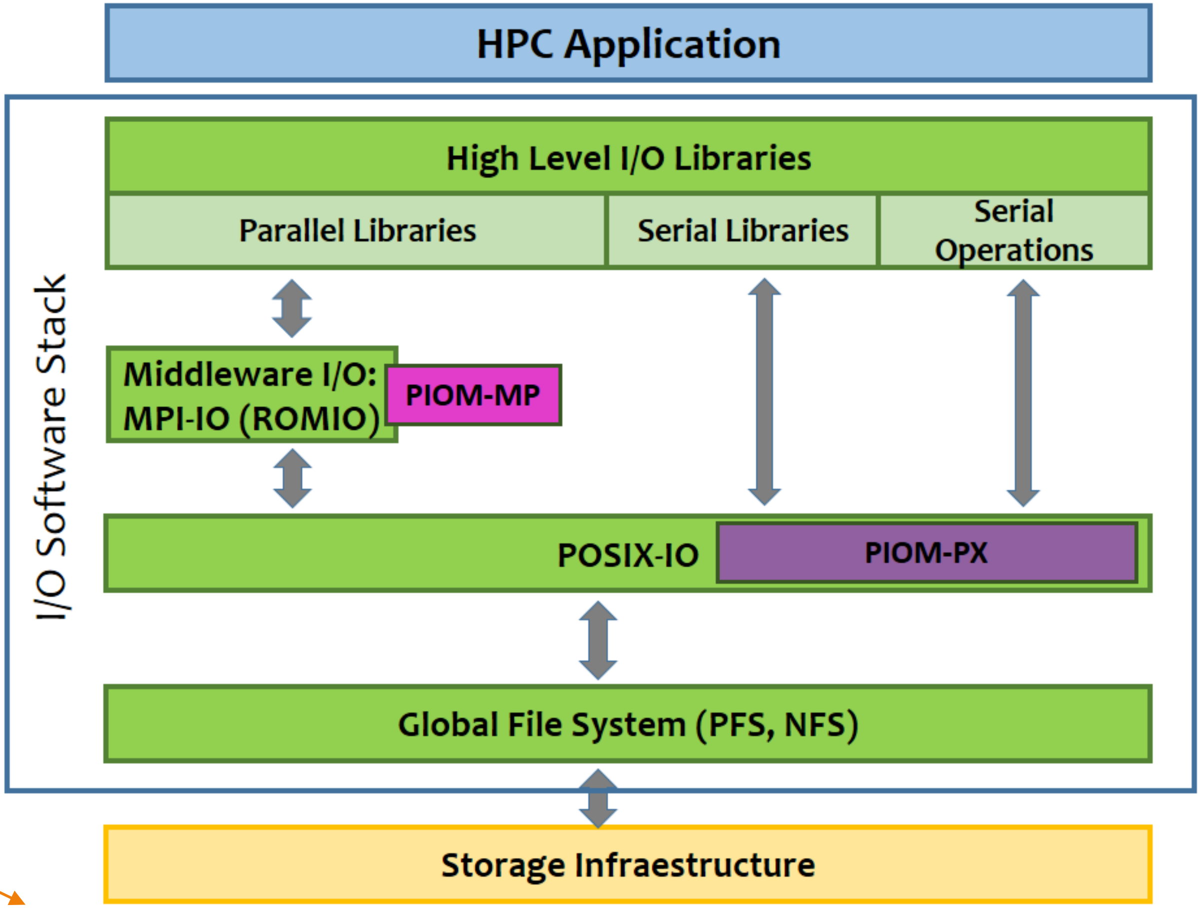
Objectives

Define an **I/O behavior model based on I/O Phase** at the **POSIX-IO level.**

- Select the **main parameters** at **POSIX-IO level** to have a determined **portable** model.

- Define a **design framework , PIOM-PX.**

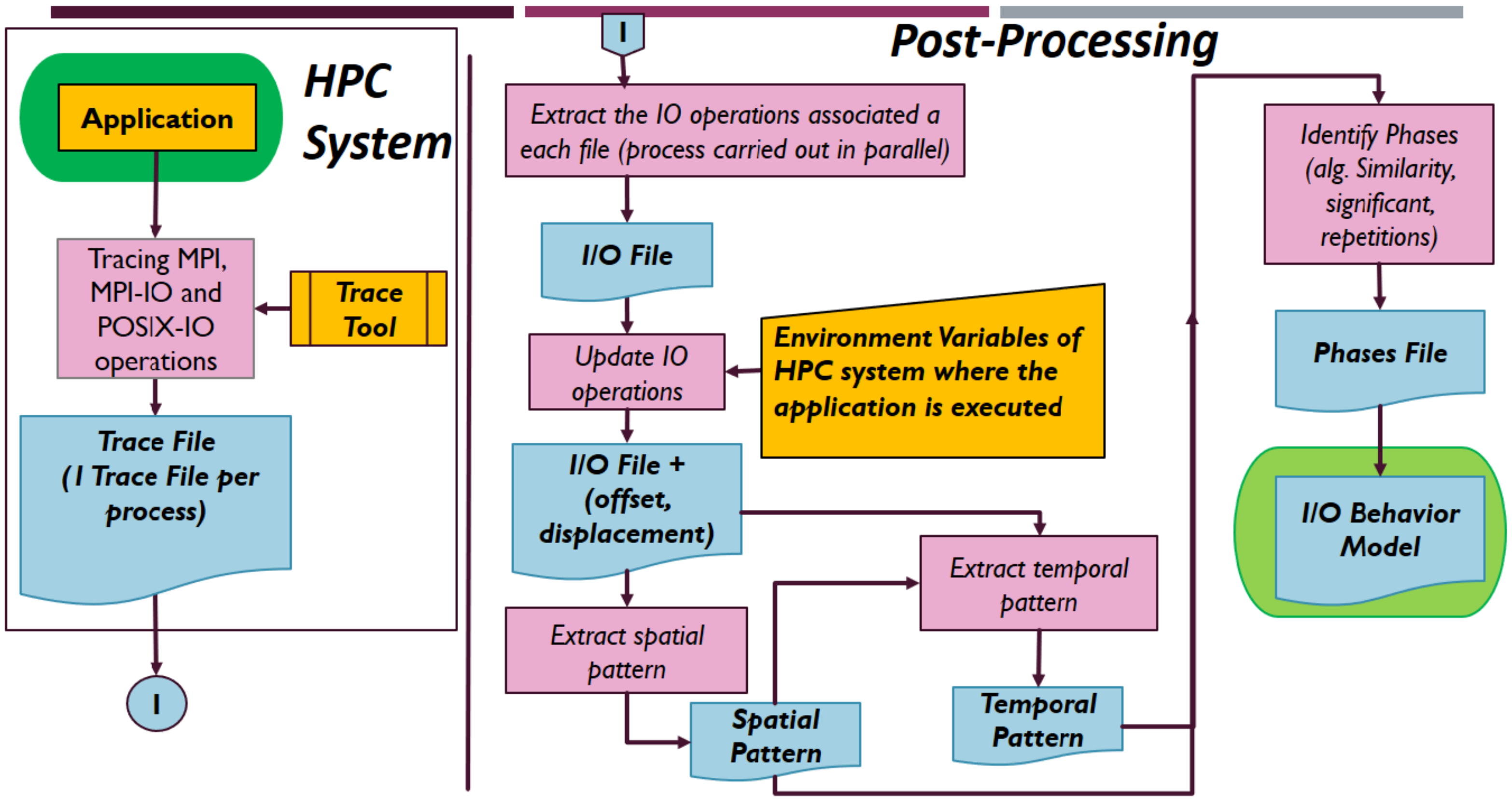
Framework



We classify the application features as parameters for PIOM-PX into three levels: application, file, and phase.

PIOM-PX was integrated with PIOM-MP, which allows us to trace I/O activities at MPI and POSIX-IO level.

Framework Modules

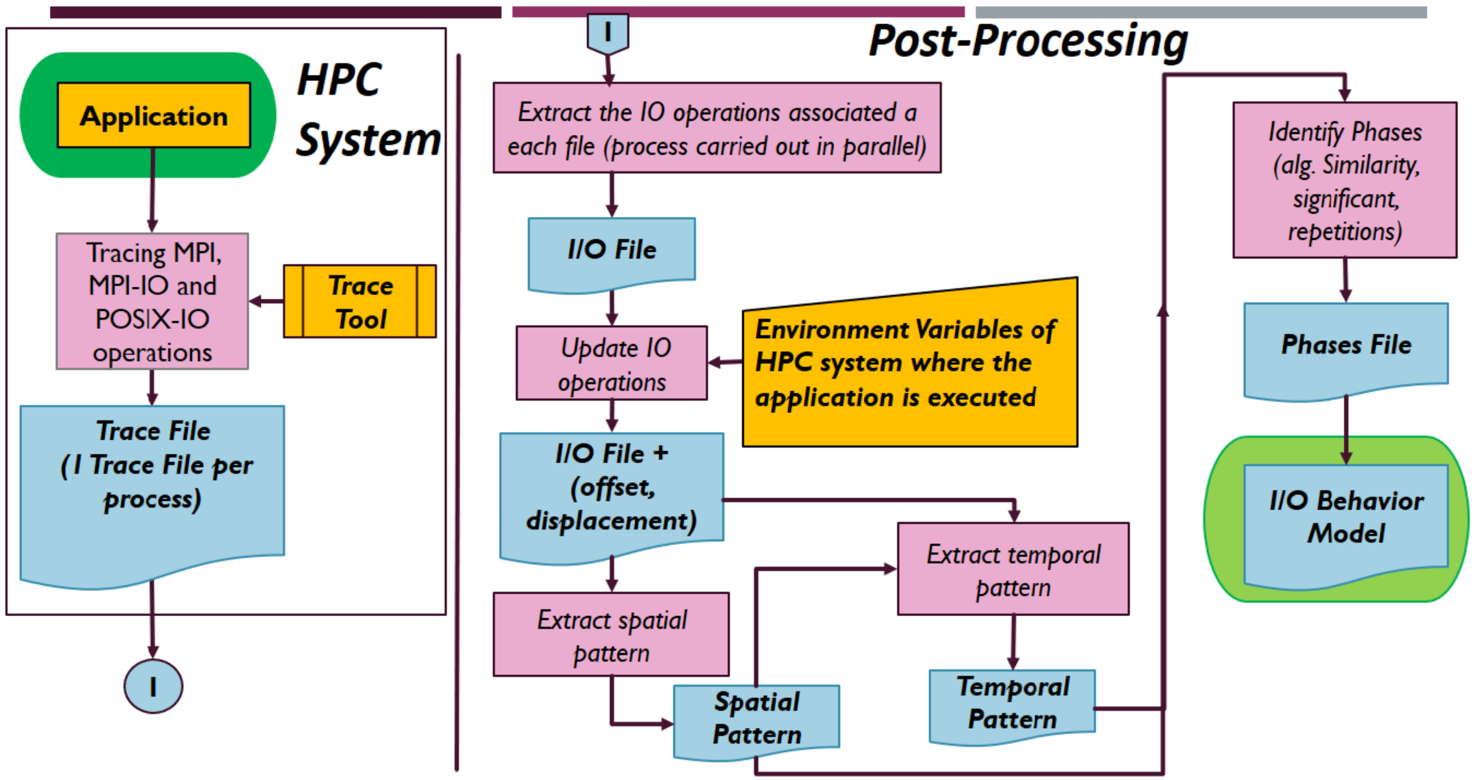


Trace module:

- Analyzed MPI Applications.
- A trace file is generated for each MPI process.
- Intercept the MPI events and the most renowned POSIX-IO operations.

Framework Modules

Post-processing



- **Extracting I/O operations** : extract the I/O operations per file opened by the application of each trace file into a new file.
- **Updating I/O operations:** detect when the offset and request size (rs) informed require evaluating another operation to obtain the request or offset.

Post-processing

- **Analyzing 1 File: (Example 1 file x process)**

file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep

MPI events

Compute

open and close

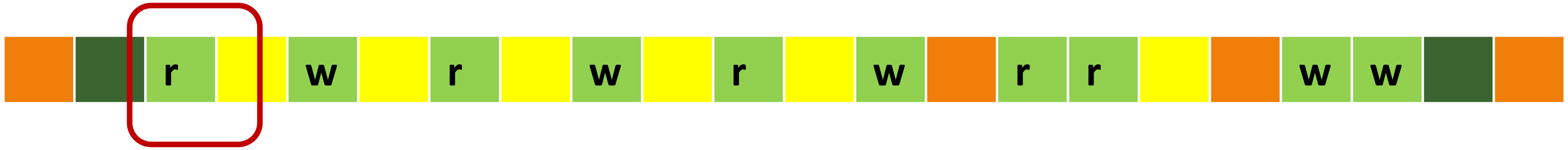
Read and write POSIX-IO operations

Post-processing

- **Analyzing 1 File: (Example 1 file x process)**

file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep



MPI events

Compute

open and close

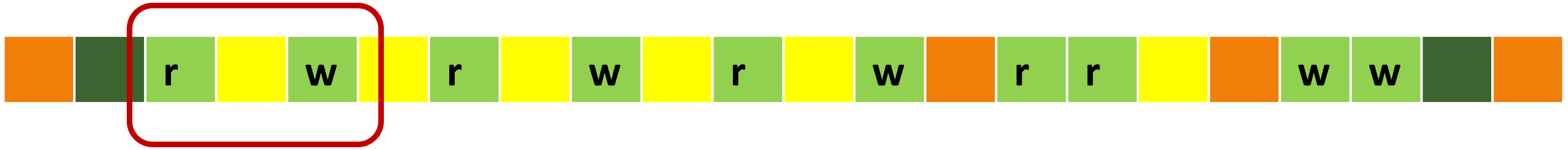
Read and write POSIX-IO operations

Post-processing

- **Analyzing 1 File: (Example 1 file x process)**

file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep



MPI events

open and close

Compute

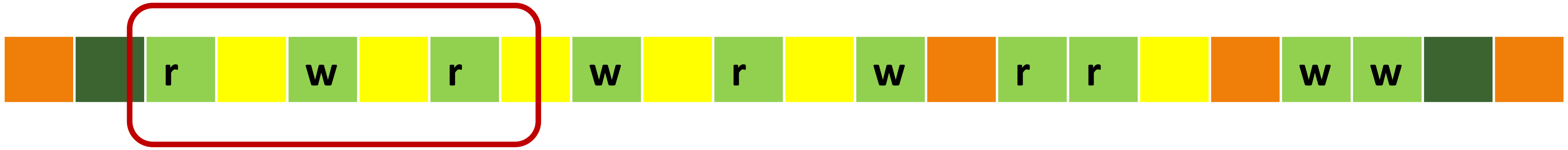
Read and write POSIX-IO operations

Post-processing

- **Analyzing 1 File: (Example 1 file x process)**

file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep



MPI events

Compute

open and close

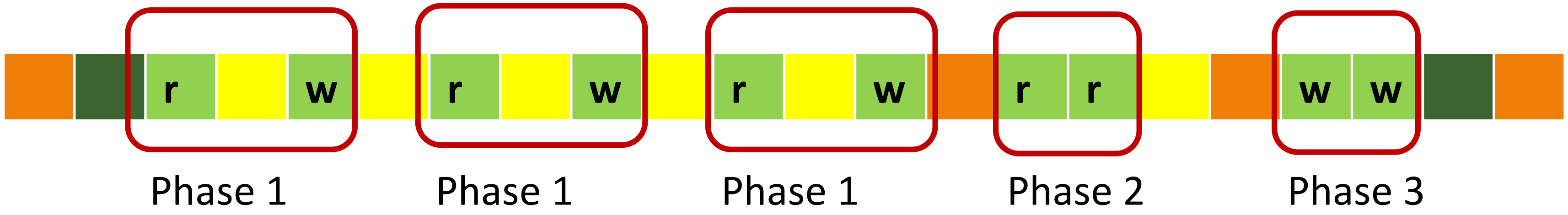
Read and write POSIX-IO operations

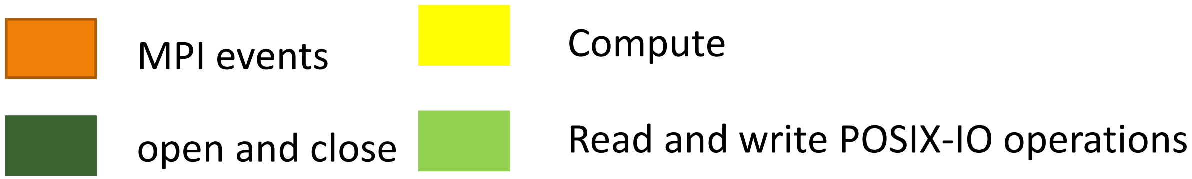
Post-processing

- **Analyzing 1 File: (Example 1 file x process)**

file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep



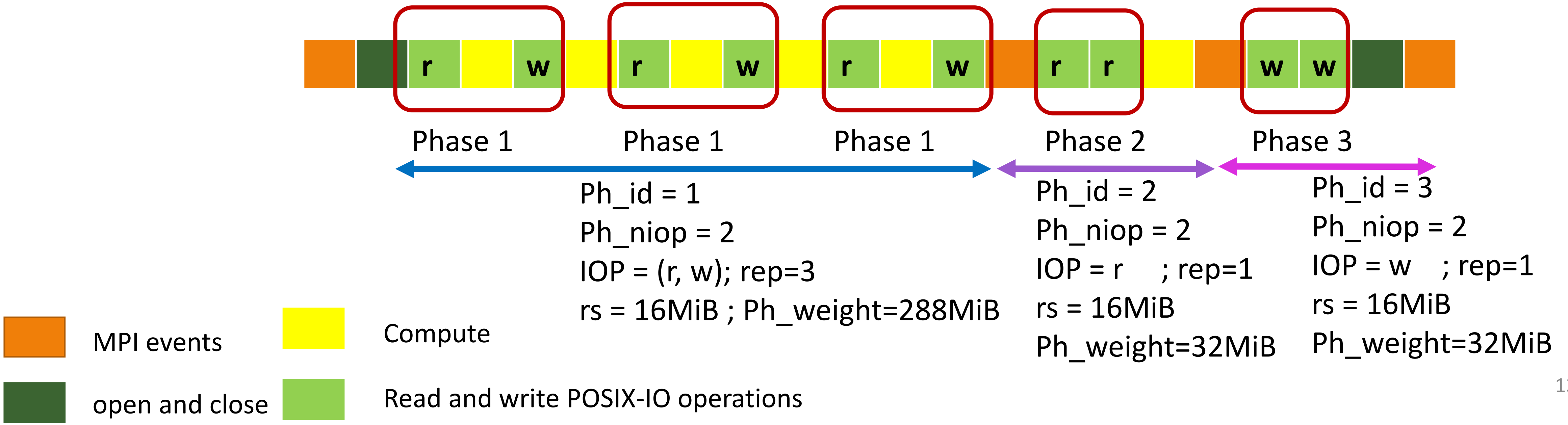


Post-processing

- **Analyzing 1 File: extracting the Spatial pattern.**

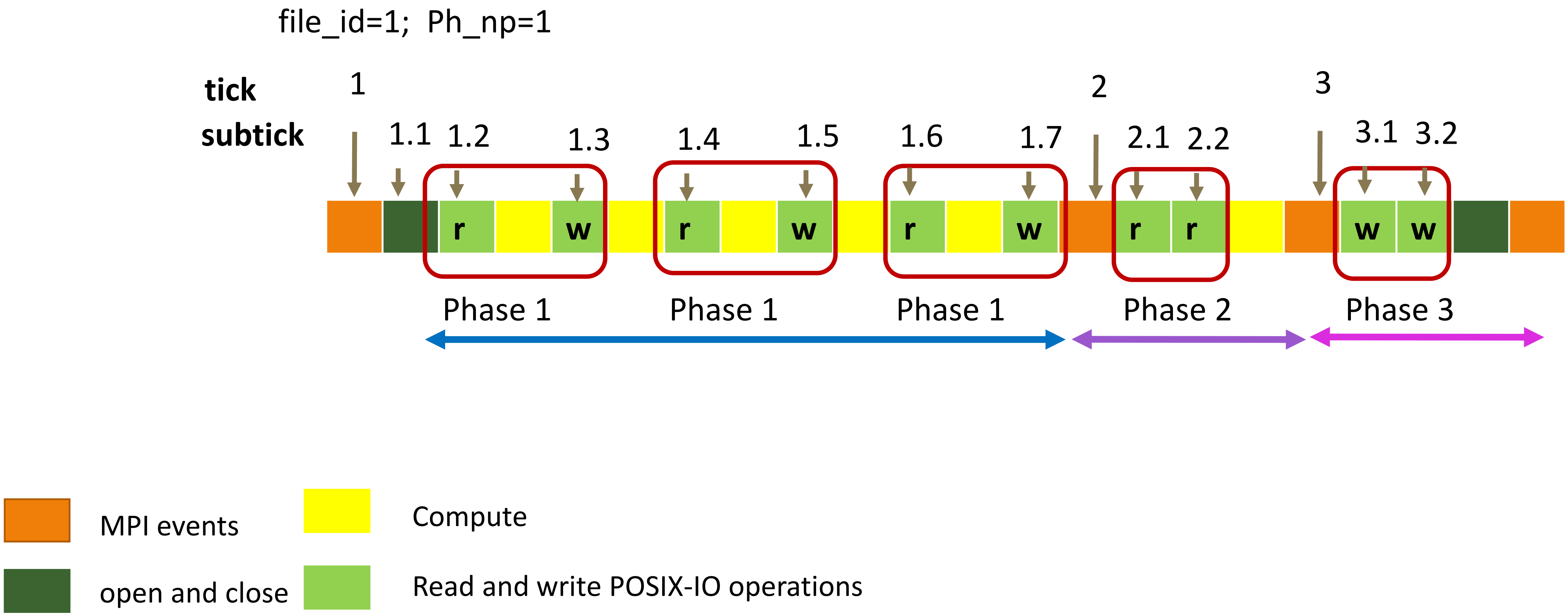
file_id=1; Ph_np=1

Phase Properties
Ph_weight =Ph_np x rs x Ph_niop x rep



Post-processing

- **Analyzing 1 File: extracting the Temporal Pattern.**



Experimental Case in different HPC systems

- 1 File per Process using POSIX interface.

```
IOR -a POSIX -s 1 -b 8m -t 1m -F
```

- 1 File per Process using MPI-IO interface.

```
IOR -a MPIIO -s 1 -b 8m -t 1m -F
```

- A single shared file using collective buffering technique in automatic mode for a strided pattern.

```
IOR -c -a MPIIO -s 16 -b 512k -t 512k
```

- A single shared file using collective buffering technique in enable mode for a strided pattern.

```
romio_cb_read = enable
romio_cb_write = enable
IOR -c -a MPIIO -s 16 -b 512k -t 512k
```

1 File per Process using POSIX interface

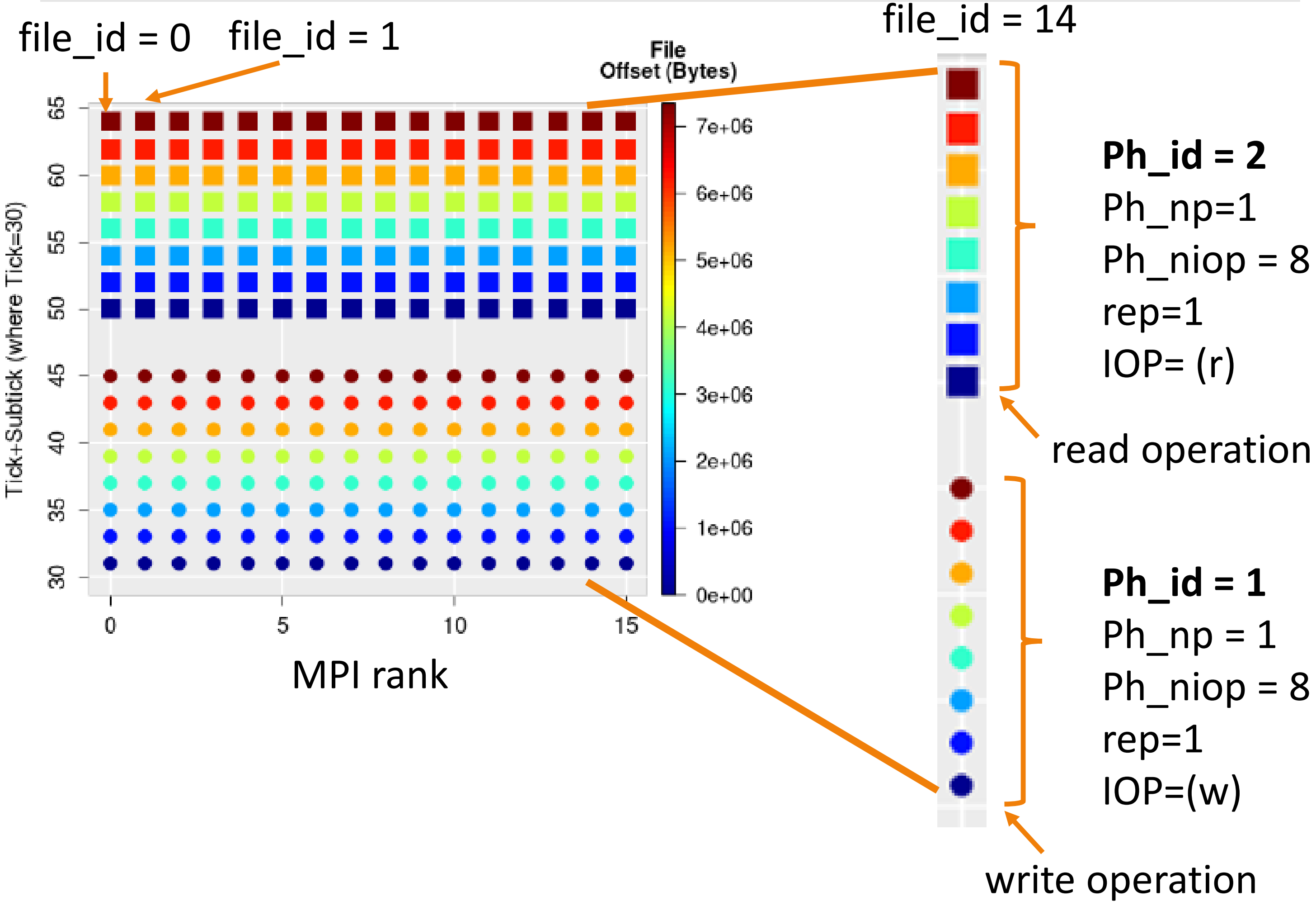
Identifier	Values
app_np	16
app_nfiles	16
app_st	128 MiB
File	
file_name	testFile<IdProcess>
file_size	8 MiB
file_accessmode	Seq
file_fileaccesstype	W/R
file_accesstype	1Fx1Proc
file_nphase	2
file_np	1

```
IOR -a POSIX -s 1 -b 8m -t 1m -F
```

1 File per Process using POSIX interface

Identifier	Values
app_np	16
app_nfiles	16
app_st	128 MiB
File	
file_name	testFile<IdProcess>
file_size	8 MiB
file_accessmode	Seq
file_fileaccesstype	W/R
file_accesstype	1Fx1Proc
file_nphase	2
file_np	1

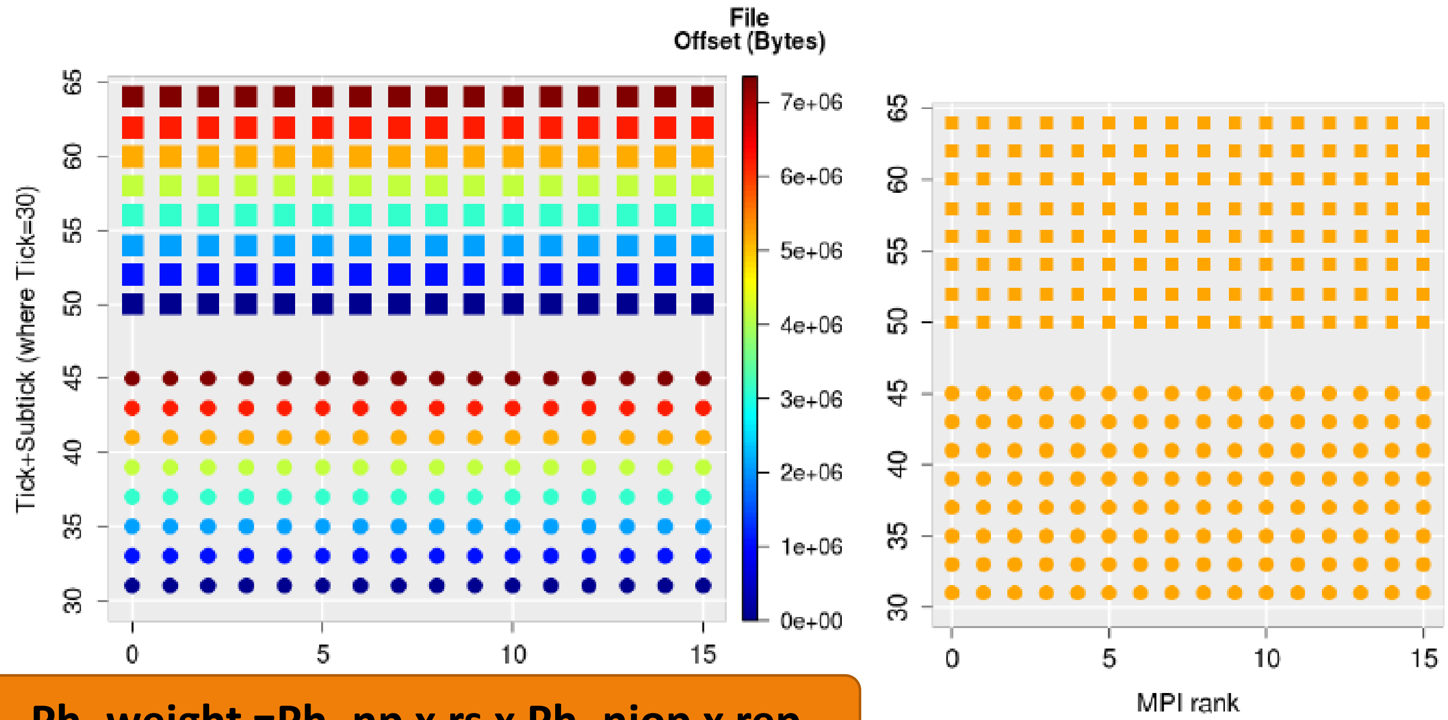
`IOR -a POSIX -s 1 -b 8m -t 1m -F`



1 File per Process using POSIX interface

Identifier	Values
app_np	16
app_nfiles	16
app_st	128 MiB
File	
file_name	testFile<IdProcess>
file_size	8 MiB
file_accessmode	Seq
file_fileaccesstype	W/R
file_accesstype	1Fx1Proc
file_nphase	2
file_np	1

```
IOR -a POSIX -s 1 -b 8m -t 1m -F
```



Ph_weight =Ph_np x rs x Ph_niop x rep

(b) Phase Weight

Ph_id = 1 (w) , 2 (r) ; IOP =(w) (r) ; Ph_weight (w)= 1 x 1 x 8 x 1MiB = 8MiB
Ph_np = 1 ; rep =1 ; Ph_weight (r)= 1 x 1 x 8 x 1MiB = 8MiB
Ph_niop = 8 ; rs = 1MiB

Experimental Results

Environment

Components	Finisterrae2	SuperMUC
Compute Nodes	306	9216
CPU cores (per node)	24	16
RAM Memory	128GB	32GB
Local Filesystem	ext4	ext3
Global Filesystem (GFS)	NFS	NFS
Capacity of GFS	1.1TB	10x564x10TB
Global Filesystem (PFS)	Lustre	GPFS
Capacity of PFS	695TB	12PB
Data servers	4 OSS and 12 OSTs	80 NSD
Metadata Servers	1	
Stripe Size	1MiB	8MiB
Interconnection	IB FDR@56Gbps	IB FDR10

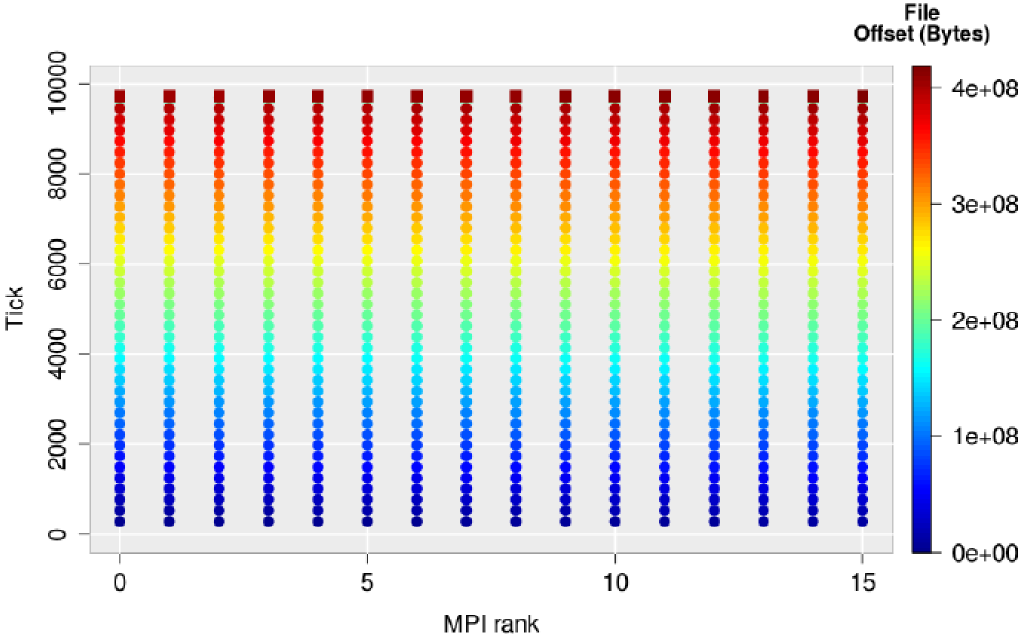
Applications

BT-IO Full, Class: A, B and C

PIOM-PX parameters for the BT-IO benchmark subtype FULL

Identifier	Class A	Class B	Class C
app_np	16	16	36
app_nfiles	1	1	1
app_st	400 MiB	1.6 GiB	6.4 GiB
File			
file_name	btio.full.out	btio.full.out	btio.full.out
file_size	400 MiB	1.6 GiB	6.4 GiB
file_accessmode	Strided	Strided	Strided
file_fileaccesstype	W/R	W/R	W/R
file_accesstype	Shared	Shared	Shared
file_nphase	41	41	41
file_np at MPI-IO level	16	16	36
file_np at POSIX-IO level	1	1	3

Experimental Results



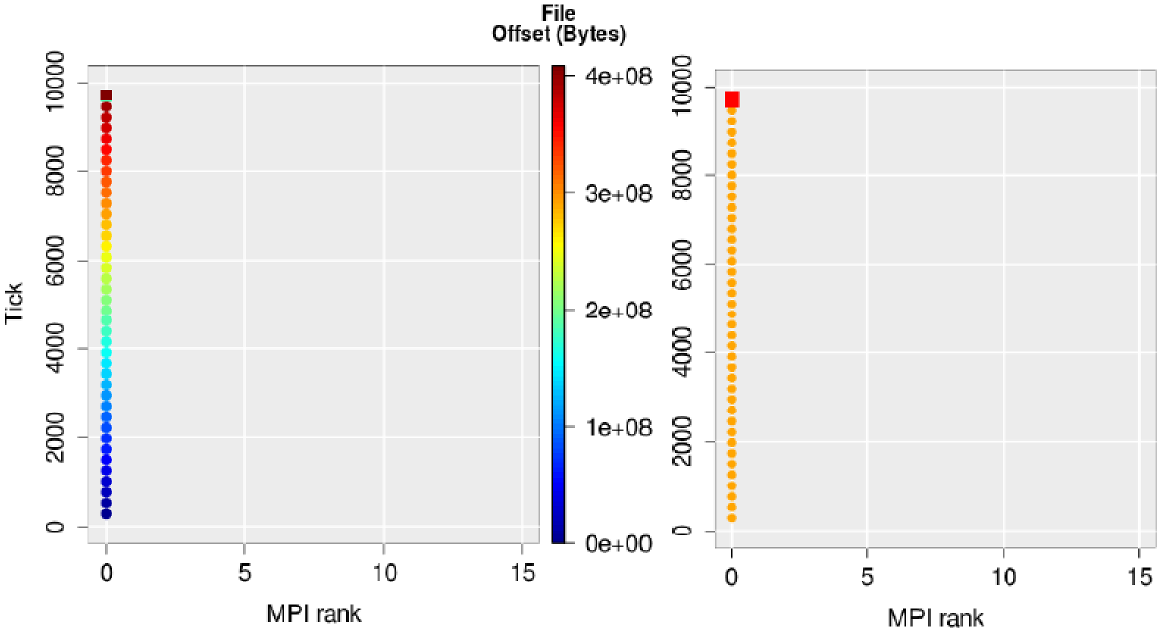
File offset at MPI level by using PIOM-MP

Weight Phase : 1 Write operation 655360 bytes

Write Weight Phase	Read Weight Phase
655360 bytes x 16 bytes	655360 bytes x 16 x 40

File offset and Phase Weight at POSIX-IO using PIOM-PX

Write Weight Phase	Read Weight Phase
10485760 bytes	10485760 x 40 bytes



Conclusions

- Our approach allows us to obtain the application's I/O behavior at phase level.
- We can observe different I/O behavior at different I/O level
- I/O behavior helps to understand the relationship between the application and the I/O system.
- Our framework makes it possible to have accurate information over the I/O phases.

HPC I/ODC

I/O in the Data Center Workshop

PIOM-PX: A Framework for Modeling the I/O Behavior of Parallel Scientific Applications

Pilar Gomez Sanchez

pilar.gomez@uab.es

http://grupsderecerca.uab.cat/hpc4eas

June 2017

HPC4EAS Research Group

UAB Universitat Autònoma de Barcelona

lrz Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften

escoles universitàries gimbernat ADSCRITA A LA UAB