# TEMPORAL AGGREGATION

## For Voluminous Trace Visualization

AEP 10 - 12th June 2014

**Damien Dosimont**
Guillaume Huard
Jean-Marc Vincent

Inria, Grenoble University
first.last@imag.fr

# How an Internet Provider Loses a Customer

# RECORDING A STREAMING WITH A SET-TOP BOX

→ A customer wants to see the last episode of Game of Throne on his TV

→ Unfortunately, the episode is released the morning

→ So, he decides to record the streaming on a hard drive thanks to his set-top box

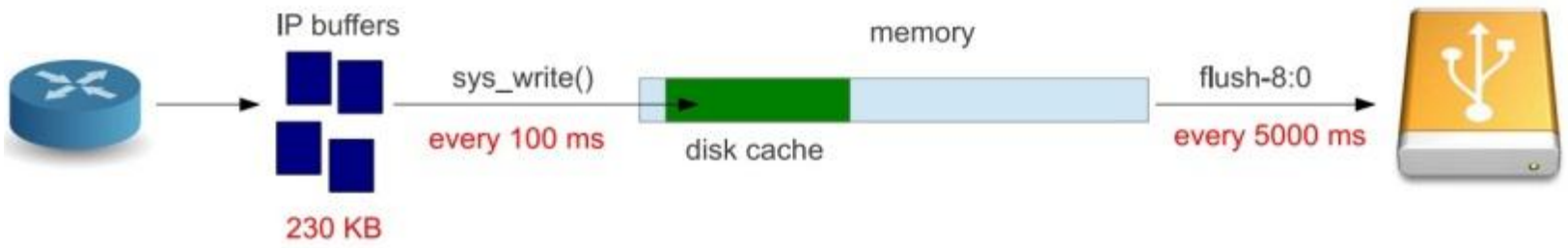
→ Eventually, he watches the episode just after the work

→ But...

# THE VIDEO

Play

# ST MICROELECTRONICS' TS RECORD USE CASE




→ HD video streaming : **big quantity of data** transmitted through the **network**

→ Data stored in **IP buffers**, waiting to be sent to the disk

→ **sys_write()** function send the data to the disk every 100 ms

→ The **kernel flushes** the disk cache every 5000 ms

# Trace Visualization Issues

# HOW TO DEBUG OUR APPLICATION?

## Tracing

Collect **information** about the **application behavior**

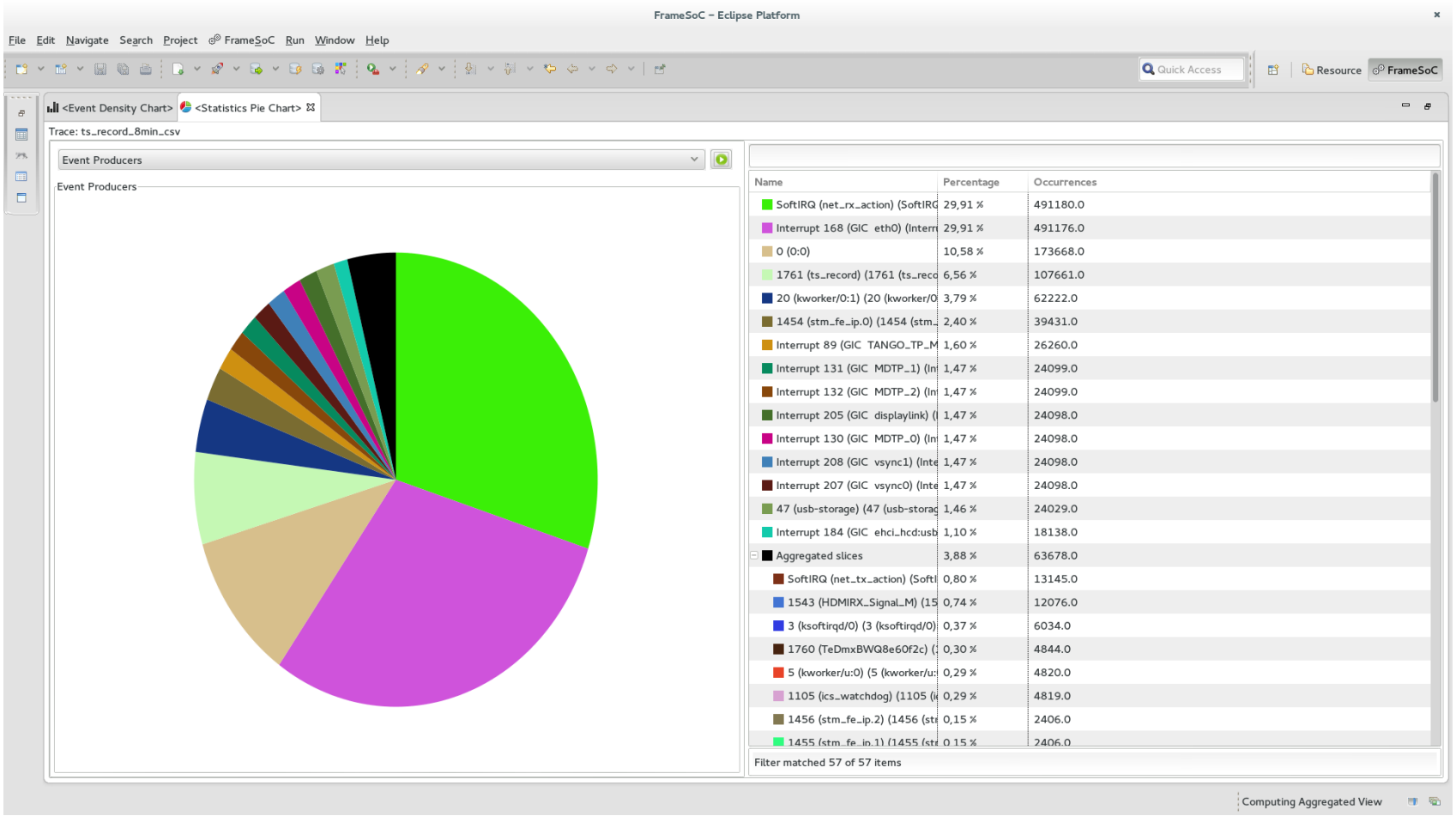
→ **Structure** :
- → **hardware** components (machines, cores, dedicated hardware)
- → **software** components (processes, threads, system, middlewares)

→ **Timestamped events** : function calls, synchronization, communications, CPU load, memory utilization etc.

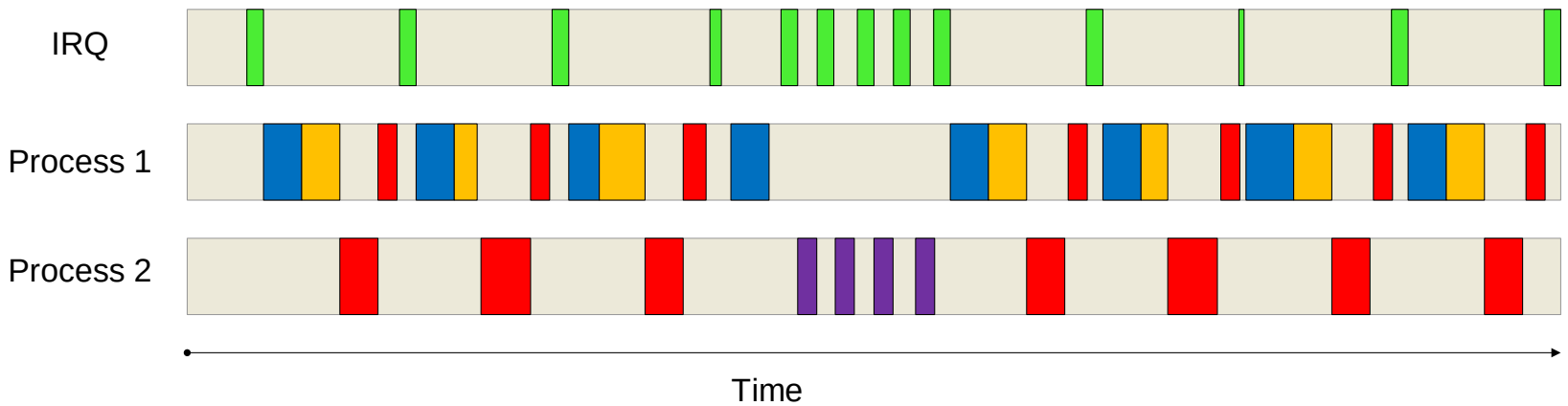
## Visualization

Statistics, Gantt chart, call graphs etc.

# STATISTICS


FrameSoC – Eclipse Platform
File Edit Navigate Search Project FrameSoC Run Window Help
Quick Access
Resource
FrameSoC
<Event Density Chart>
<Statistics Pie Chart>
Trace: ts_record_8min_csv
Event Producers
Event Producers
Name
Percentage
Occurrences
SoftIRQ (net_rx_action) (SoftIRQ 29,91 % 491180.0
Interrupt 168 (GIC eth0) (Interr 29,91 % 491176.0
0 (0:0) 10,58 % 173668.0
1761 (ts_record) (1761 (ts_reco 6,56 % 107661.0
20 (kworker/0:1) (20 (kworker/0 3,79 % 62222.0
1454 (stm_fe_ip.0) (1454 (stm_ 2,40 % 39431.0
Interrupt 89 (GIC TANGO_TP_M 1,60 % 26260.0
Interrupt 131 (GIC MDTP_1) (In 1,47 % 24099.0
Interrupt 132 (GIC MDTP_2) (In 1,47 % 24099.0
Interrupt 205 (GIC displaylink) ( 1,47 % 24098.0
Interrupt 130 (GIC MDTP_0) (In 1,47 % 24098.0
Interrupt 208 (GIC vsync1) (Inte 1,47 % 24098.0
Interrupt 207 (GIC vsync0) (Inte 1,47 % 24098.0
47 (usb-storage) (47 (usb-storag 1,46 % 24029.0
Interrupt 184 (GIC ehci_hcd:usb 1,10 % 18138.0
Aggregated slices 3,88 % 63678.0
SoftIRQ (net_tx_action) (Softi 0,80 % 13145.0
1543 (HDMIRX_Signal_M) (15 0,74 % 12076.0
3 (ksoftirqd/0) (3 (ksoftirqd/0) 0,37 % 6034.0
1760 (TeDmxBWQ8e60f2c) ( 0,30 % 4844.0
5 (kworker/u:0) (5 (kworker/u: 0,29 % 4820.0
1105 (ics_watchdog) (1105 (i 0,29 % 4819.0
1456 (stm_fe_ip.2) (1456 (st 0,15 % 2406.0
1455 (stm_fe_ip.1) (1455 (st 0,15 % 2406.0
Filter matched 57 of 57 items
Computing Aggregated View

# EXAMPLE OF GANTT CHART



# VISUALIZATION ISSUES

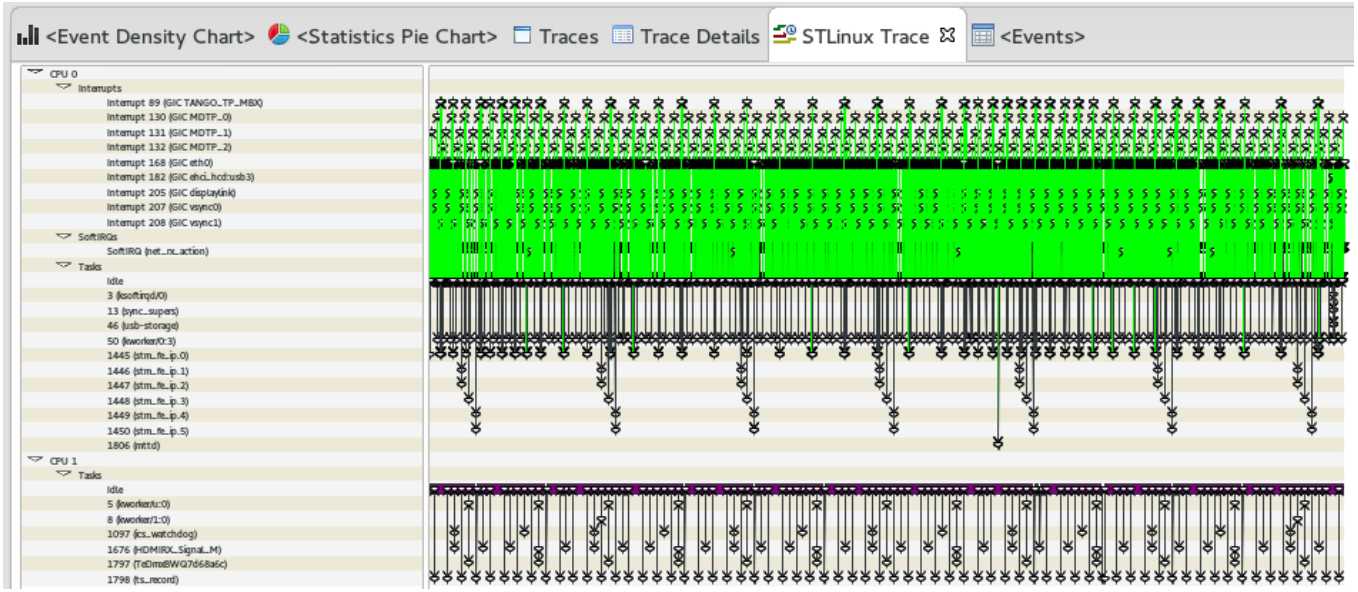
→ **Statistics** are not able to show temporal perturbations

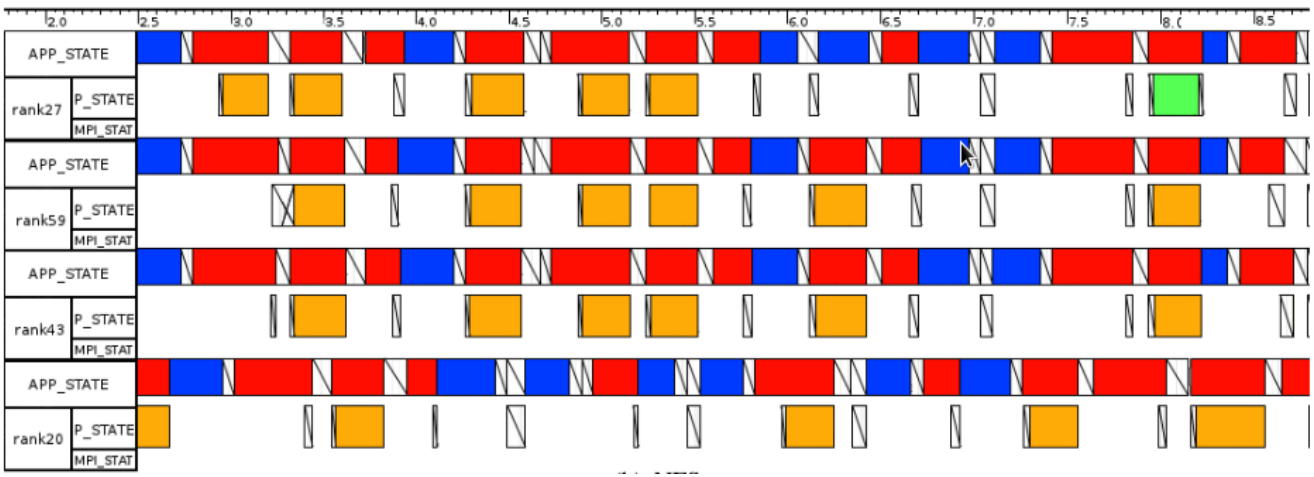
→ Traces are **huge** (we have worked with traces up to 200 million events)

→ Traditional temporal visualization does not respect **Shneiderman methodology**
*Overview first, zoom and filter, then, get details-on-demand*

→ Some tools propose **aggregation techniques** but they do not convey **meaning** to the user

# EXAMPLE OF GANTT CHART ISSUES



# OUR PROPOSAL: OCELOTL

**A temporal aggregation technique that provides an overview over time**

→ We use a **meaningful aggregation** algorithm that gathers temporal parts of the trace where the behavior is similar

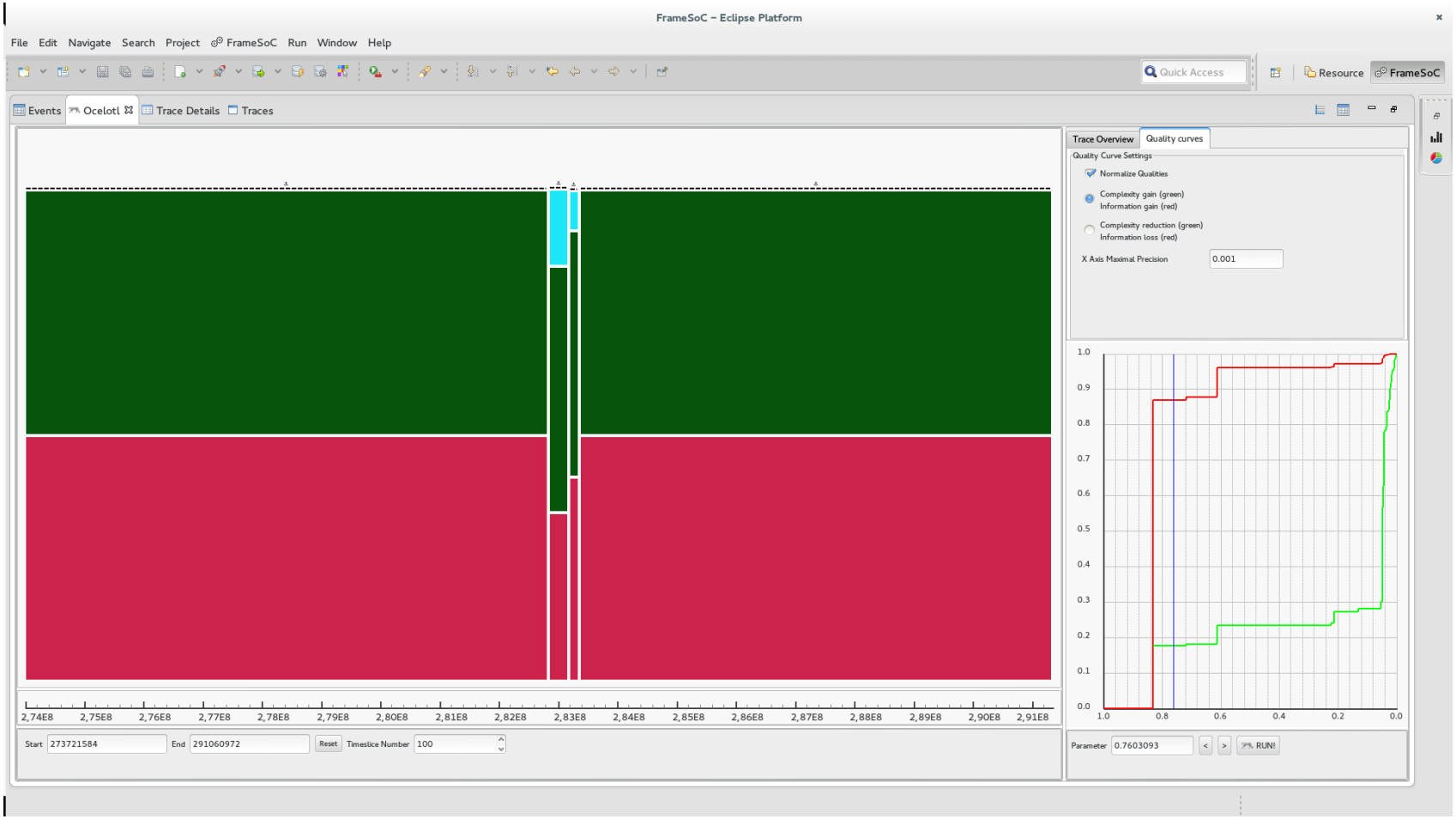
→ The user chooses the **aggregation strength**

→ He is aware of the **information loss**

→ He can **interact**

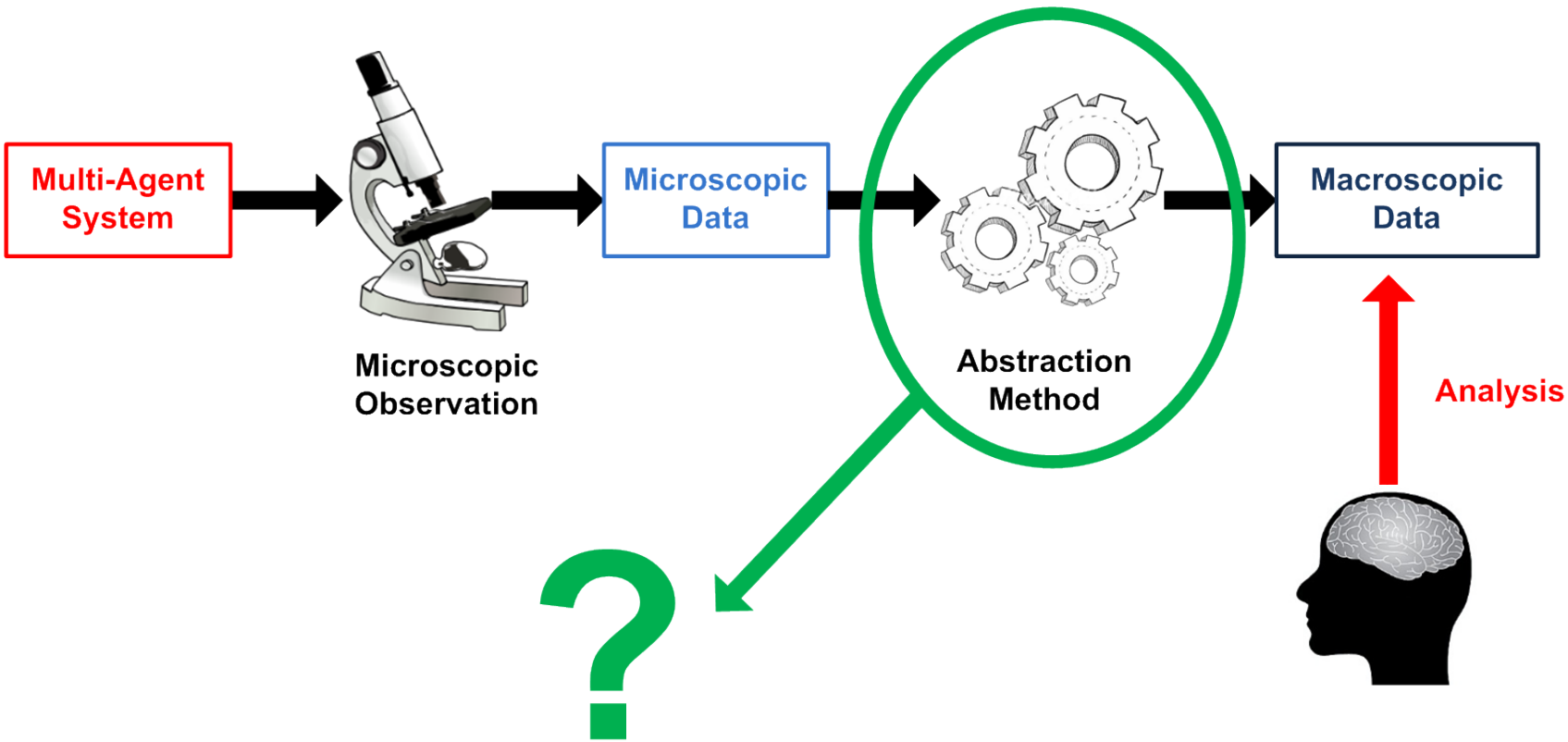
→ We manage up to 200 million events

# EXAMPLE OF OCELOTL VISUALIZATION

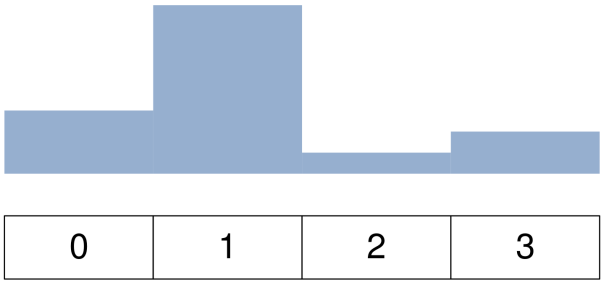


# Temporal Aggregation Technique

# LAMARCHE-PERRIN: BUILD A MACROSCOPIC MODEL



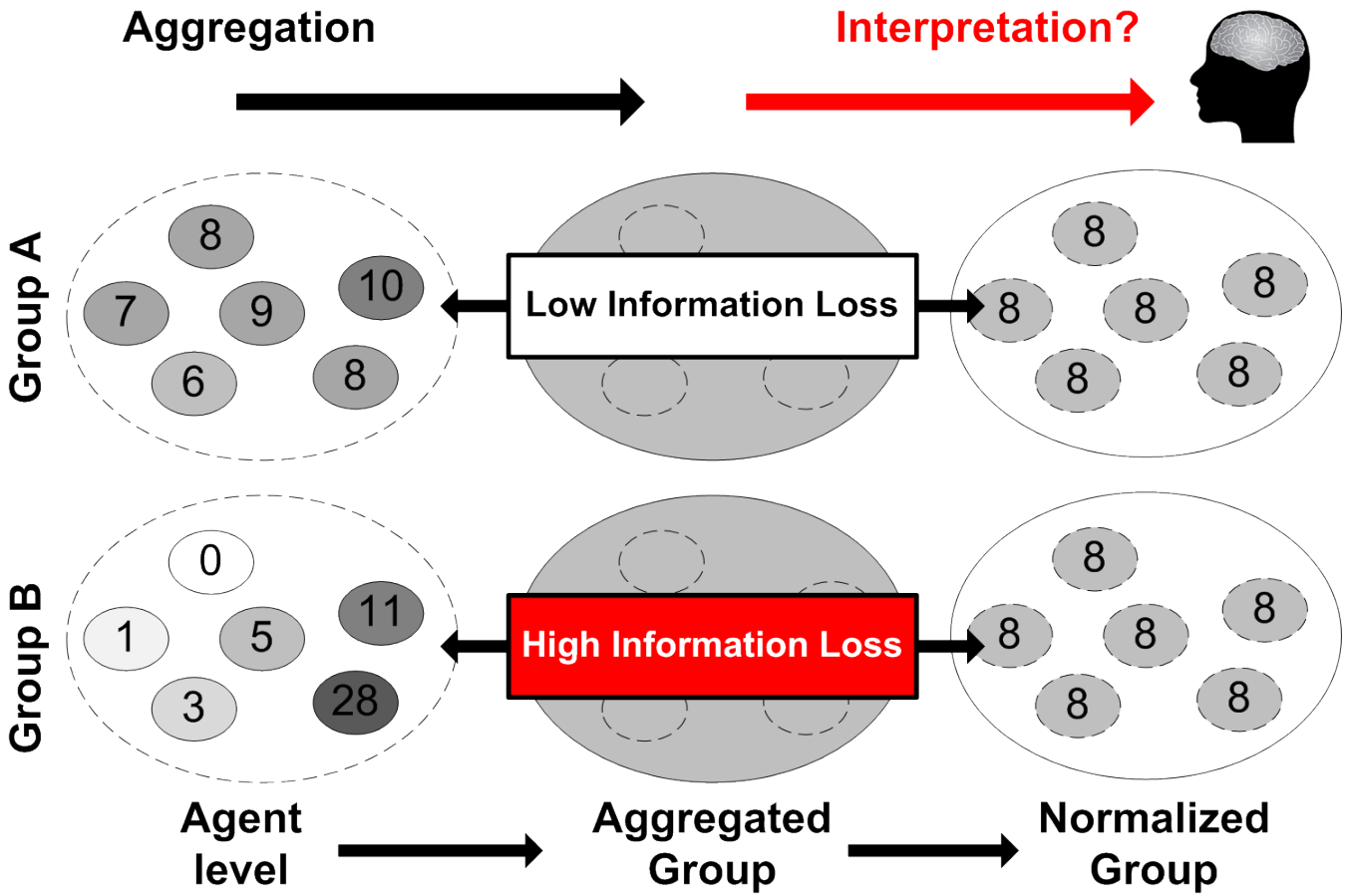
# AGGREGATION METHODOLOGY




**Microscopic Model**

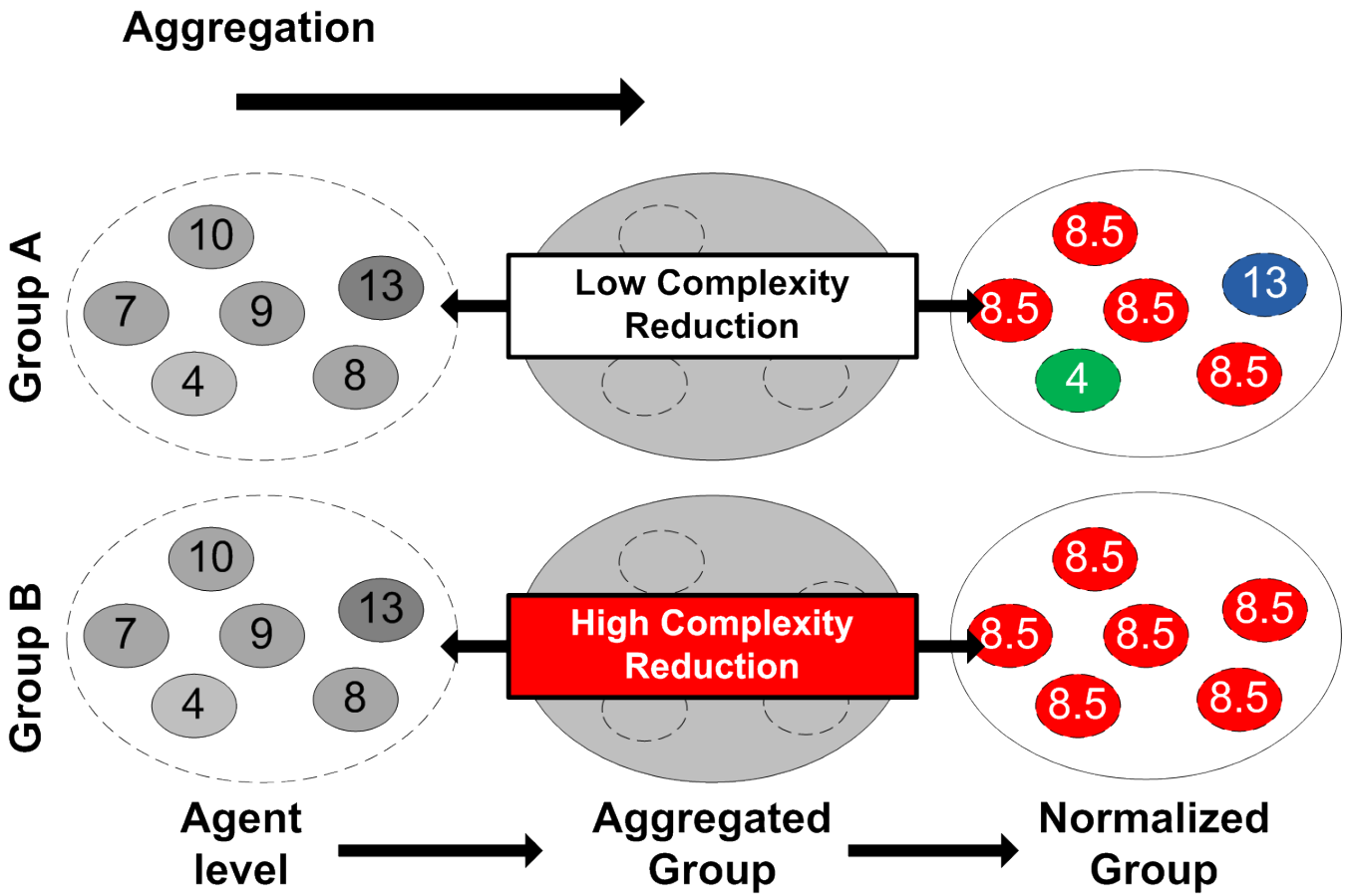
→ Determine **the dimension(s)** to aggregate:
*time*
→ Define a **microscopic model**:
*time slices with associated values*
→ **Constrain** the aggregation:
*only consecutive parts*
→ Choose a **criterion** that **triggers** the aggregation:
*information and complexity*
→ Build an **algorithm** to provide a partition of the system

# INFORMATION LOSS: KL DIVERGENCE



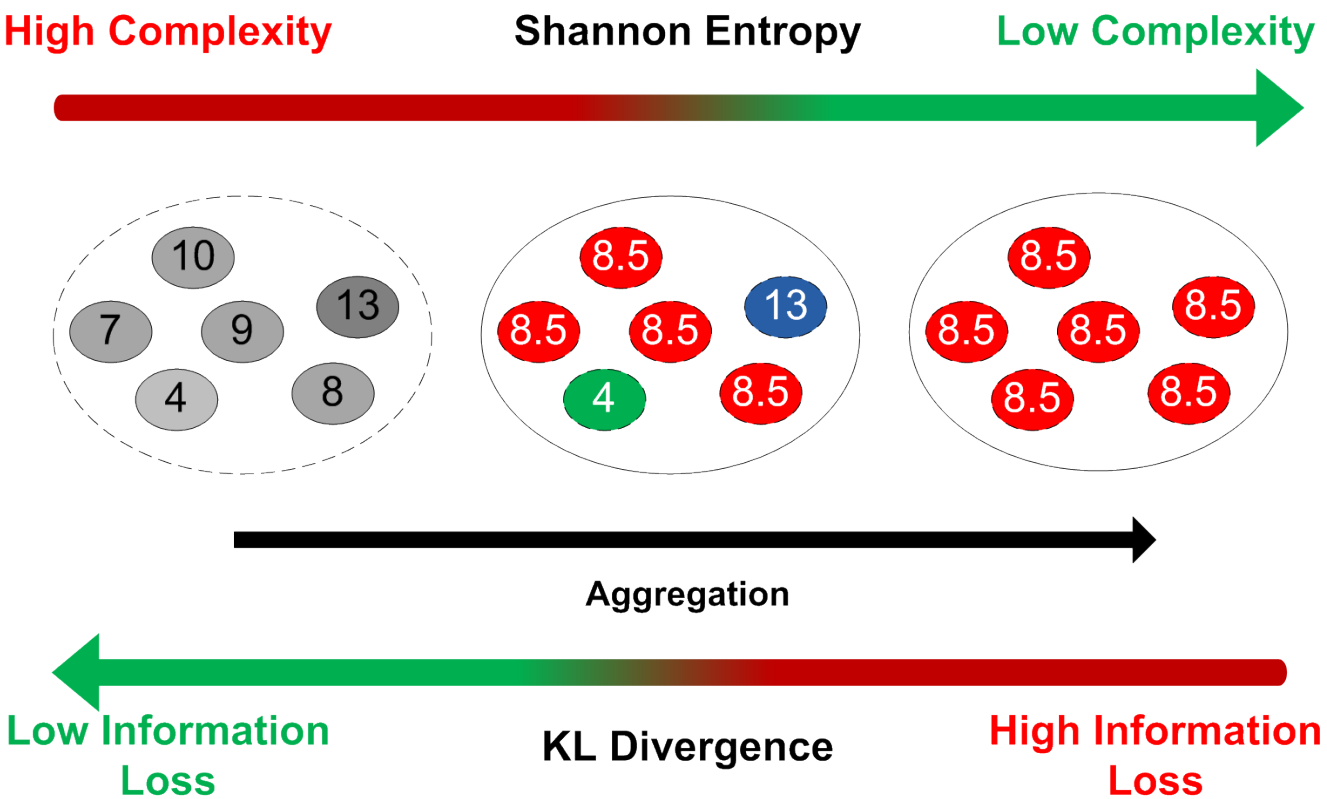

$$\text{loss}_X = \sum_{x \in X} \rho_x \log_2 \left( \frac{\rho_x}{\rho_X} \right)$$

# COMPLEXITY REDUCTION: SHANNON ENTROPY




$$\text{gain}_X = \rho_X \log_2 -\rho_X \sum_{x \in X} \rho_x \log_2 \rho_x$$

# COMPROMISE: PIC




$$\mathrm{pIC}_X = p\,\mathrm{gain}_X - (1-p)\,\mathrm{loss}_X$$

# AGGREGATION ALGORITHM

| 0 | 1 | 2 | 3 |
|---|---|---|---|

**Microscopic Model**

| 0123 | | | |
|---|---|---|---|
| 012 | 123 | | |
| 01 | 12 | 23 | |
| 0 | 1 | 2 | 3 |

**Quality Measures** **X2 (Loss, Gain)**

| 0123 | | | |
|---|---|---|---|
| 012 | 123 | | |
| 01 | 12 | 23 | |
| 0 | 1 | 2 | 3 |

**pIC for a given p**

**Best Cuts: select the highest pIC**

Best 0 = 0

Best 01 = Best 0 | 1 **or** 01

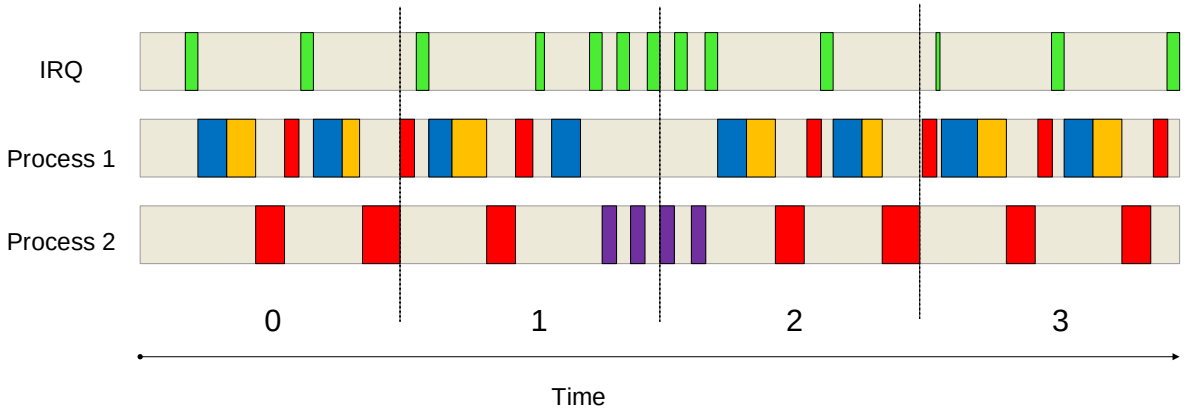
Best 012 = Best 01 | 2 **or** 012 **or** Best 0 | 12

Best 0123 = Best 012 | 3 **or** 0123 **or** Best 0 | 123 **or** Best 01 | 23

**Result:**

Best 01 | 23 = Best 0 | 1 | 23 = 0 | 1 | 23

# BUILD THE TRACE MICROSCOPIC MODEL

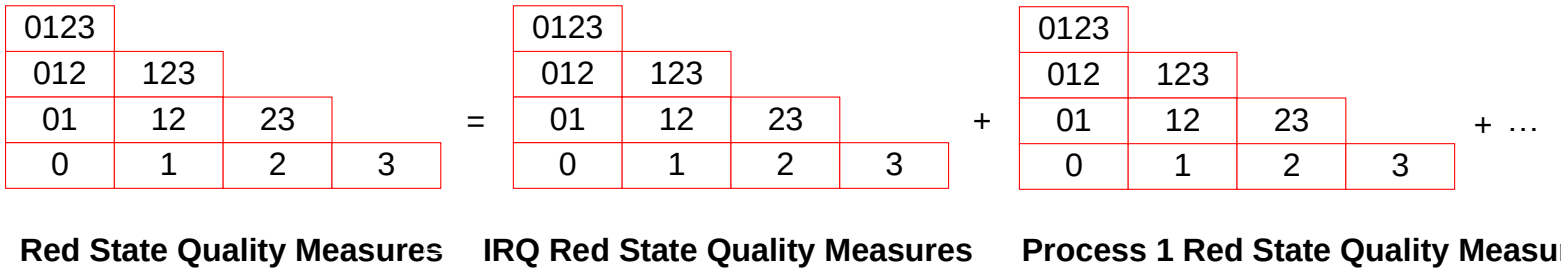



| | | | | |
|---|---|---|---|---|
| IRQ | 0 | 0 | 0 | 0 |
| Process 1 | 1 | 2.1 | 1 | 3 |
| Process 2 | 4.1 | 2 | 4.1 | 4 |

| | | | | |
|---|---|---|---|---|
| IRQ | 2 | 4.9 | 3 | 2.4 |
| Process 1 | 0 | 0 | 0 | 0 |
| Process 2 | 0 | 0 | 0 | 0 |

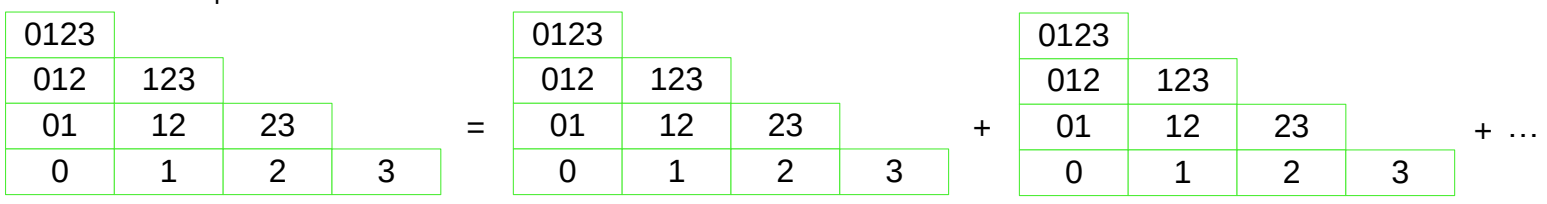
And so on...

# COMPUTE THE QUALITIES




**Red State Quality Measures** **IRQ Red State Quality Measures** **Process 1 Red State Quality Measures**

+

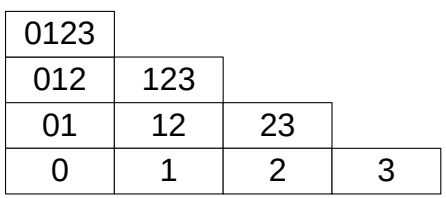



**Green State Quality Measures** **IRQ Green State Quality Measures** **Process 1 Green State Quality Measures**

+

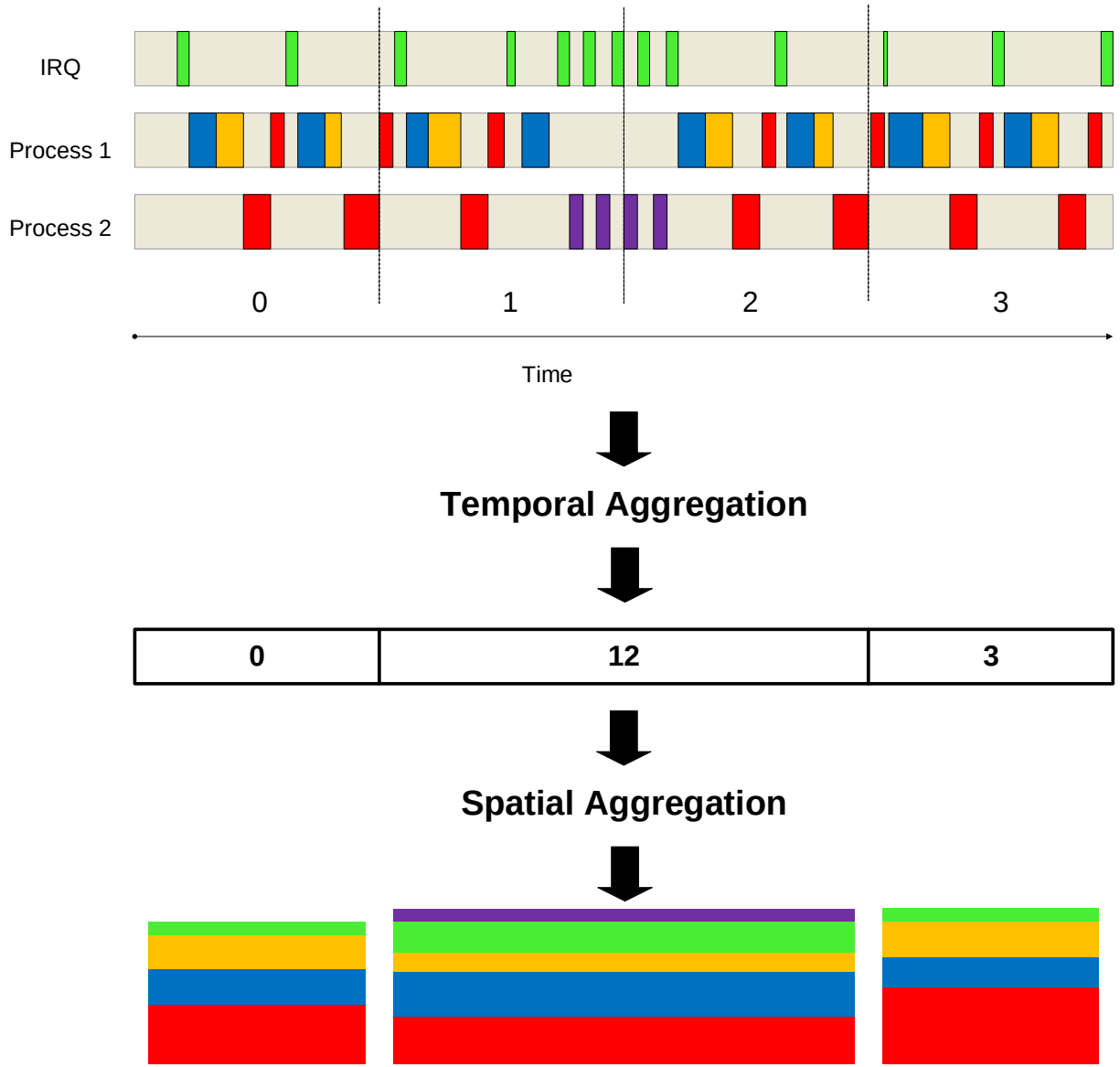
…

=



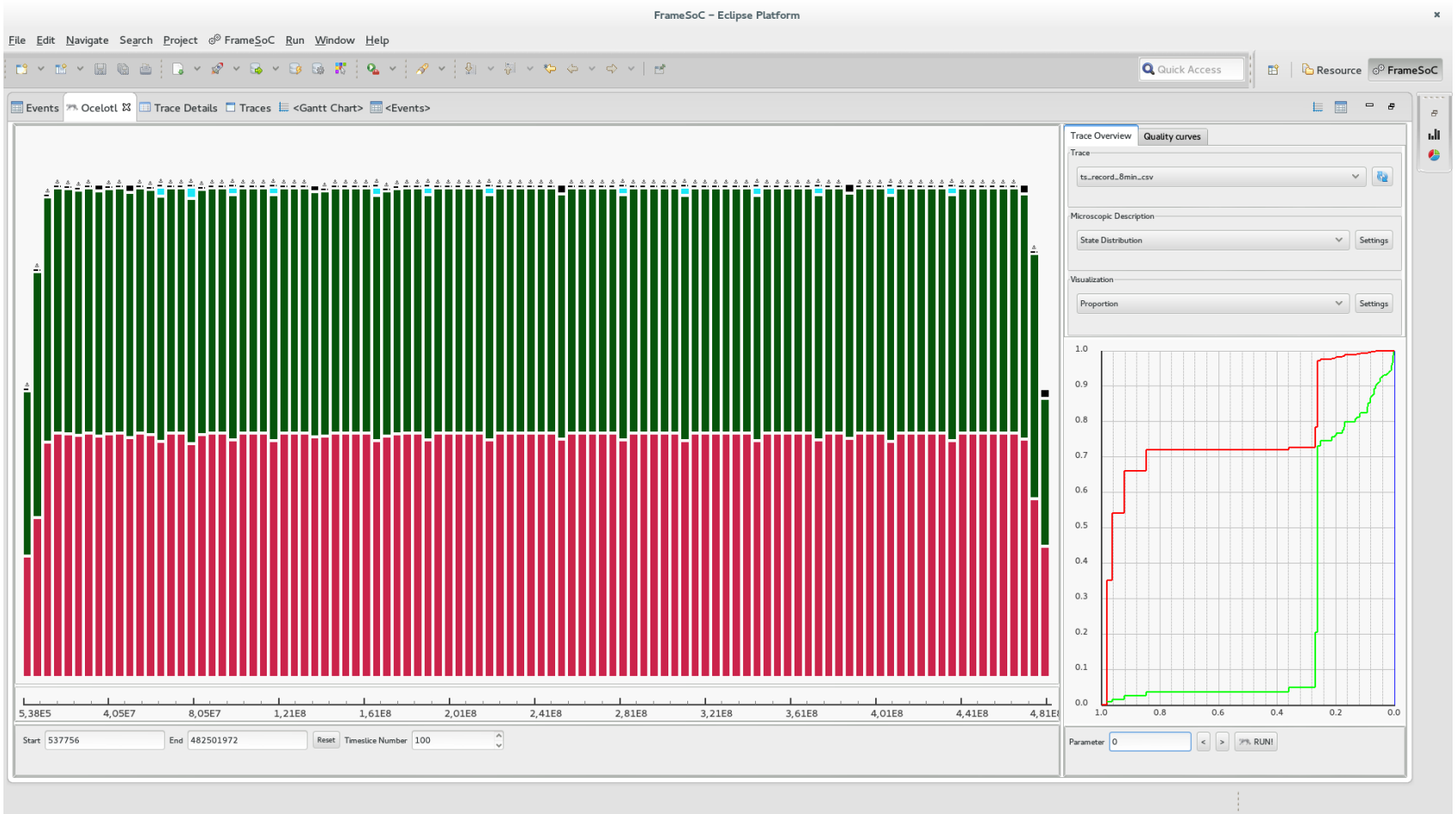

**Quality Measures**

# REPRESENT THE AGGREGATES

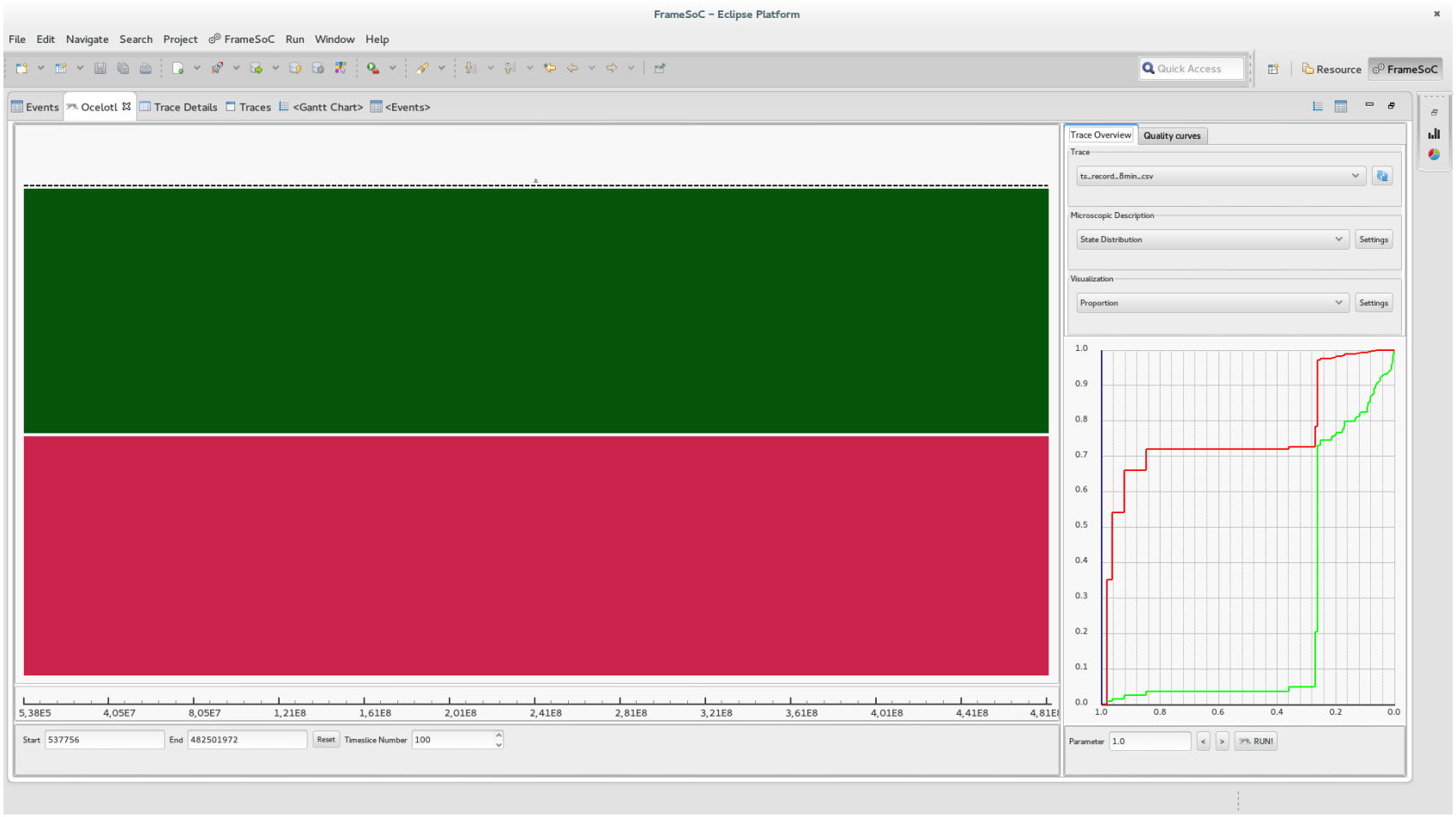


# TS Record Analysis

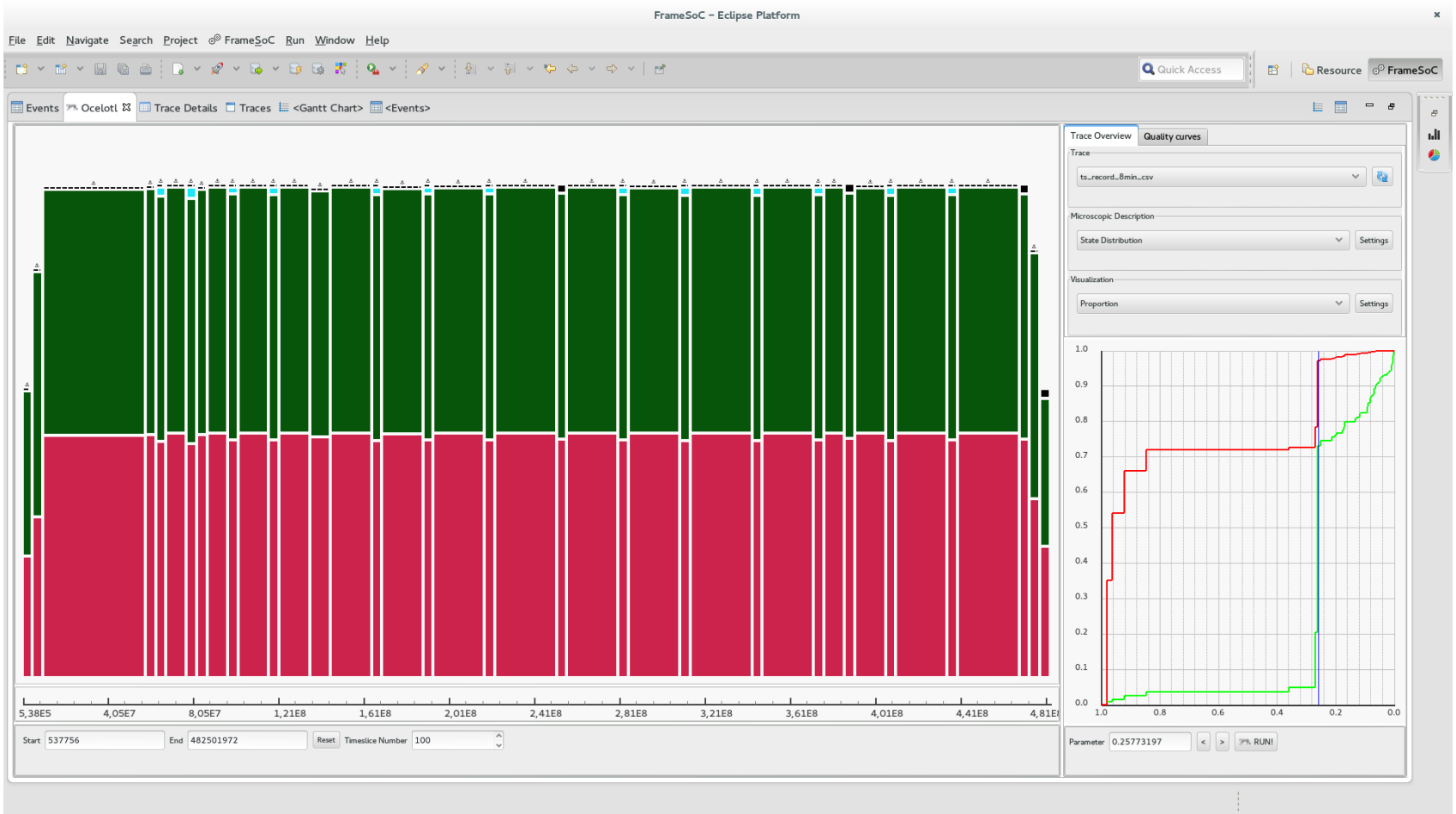
# MICROSCOPIC DESCRIPTION



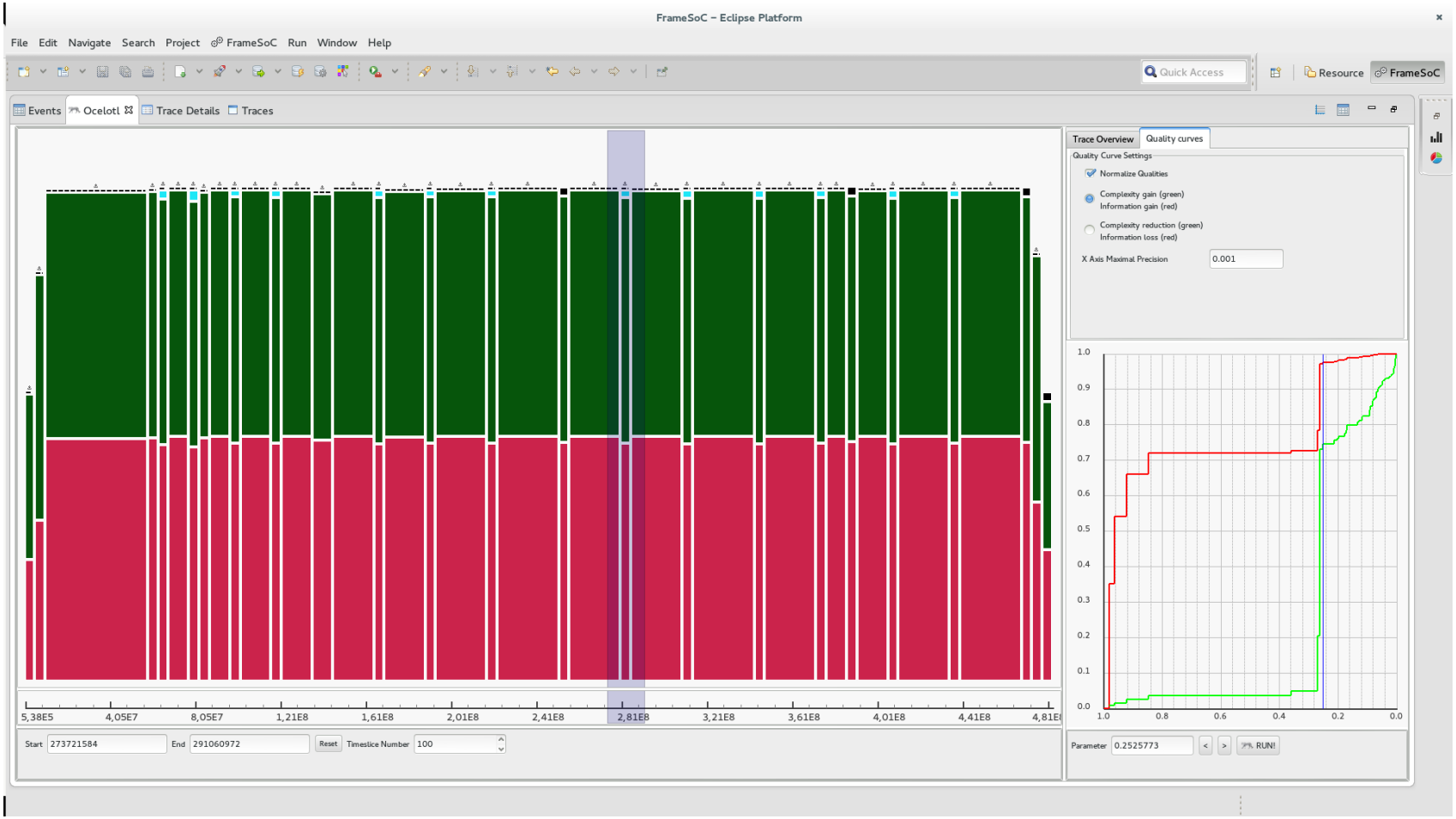
# MACROSCOPIC DESCRIPTION



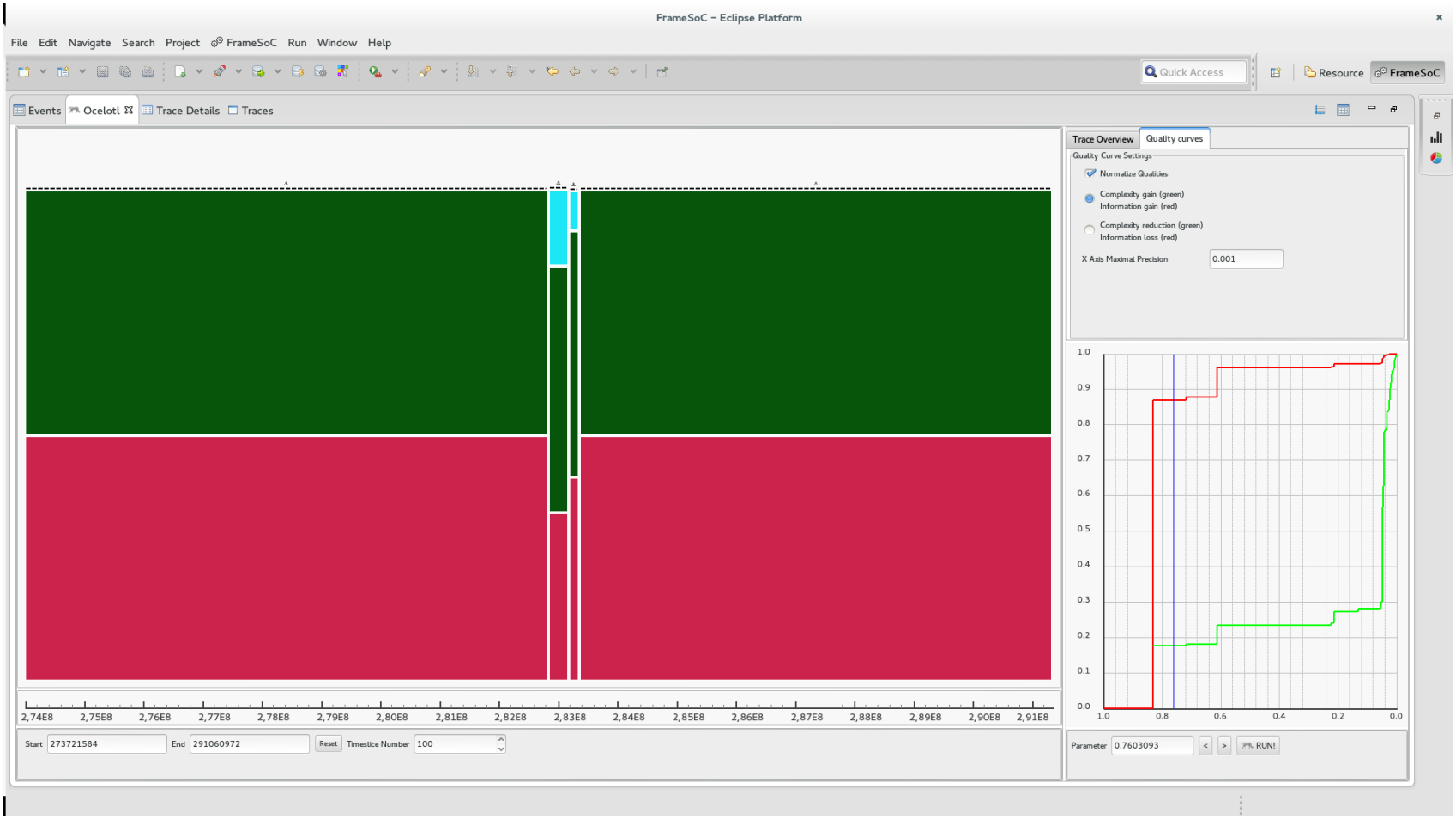
# MESOSCOPIC DESCRIPTION



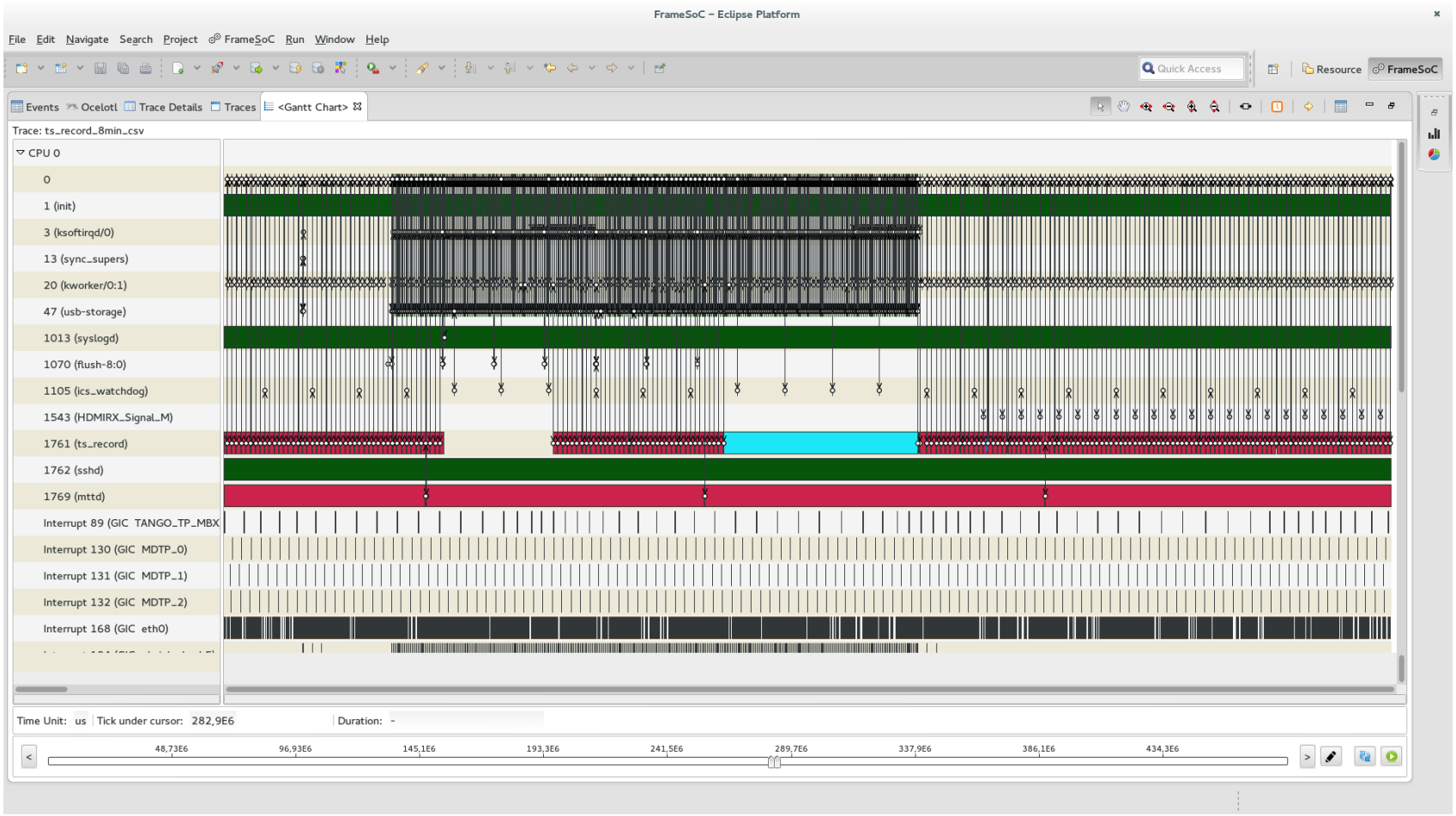
# FOCUS ON AN AREA



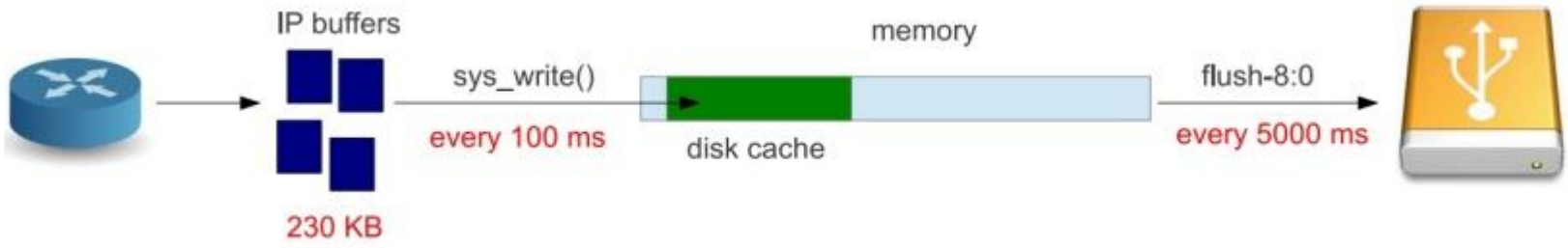
# REPEAT THE AGGREGATION PROCESS



# FIND THE BUG WITH THE GANTT CHART



# ANALYSIS CONCLUSION




→ During the flush, the cache is **locked**, and sys_write() is **blocked**

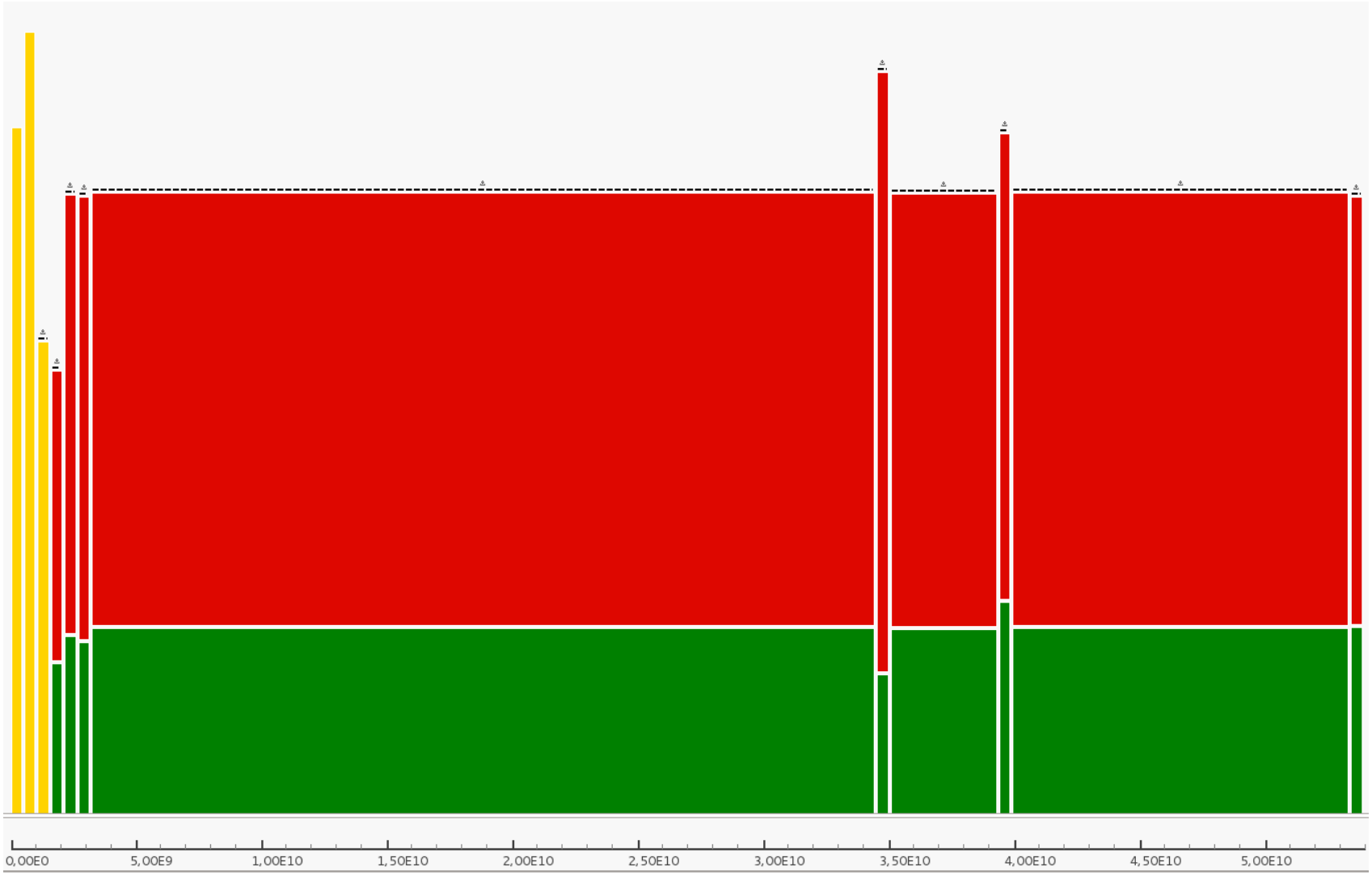
## Problem

Writing on a USB disk is **slow**, IP buffers are often **overflowed**

## Solution

Linux configuration: Wake up the flush according to the **data quantity**

# OCELOTL APPLIED ON A PARALLEL APPLICATION TRACE




Grid'5000 Rennes, 64 Cores, NASBP CG, class C

# CONCLUSION

→ **Shneiderman** methodology respected

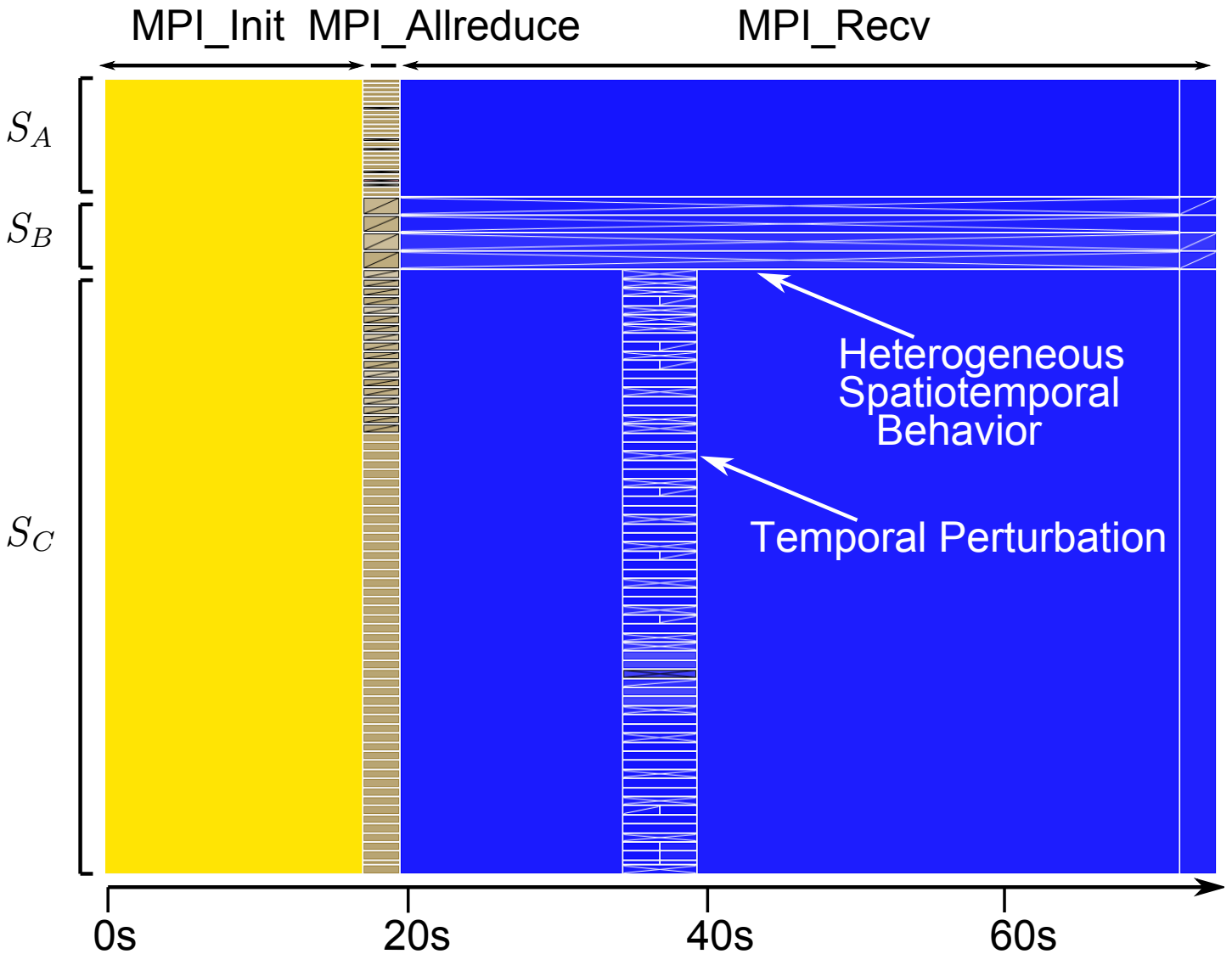
→ Experiences show the ability to understand the **temporal behavior**

→ We found why the episode of Game of Throne was not recorded correctly ;-)

→ Able to work with **200 million events** (theoretically, no limit)

→ Satisfying algorithm and interaction performance

# CURRENT WORK : SPATIO TEMPORAL AGGREGATION




Grid'5000 Nancy, 700 Cores, NASBP LU, class C