EXECUTION ANALYSIS OF DSM APPLICATIONS: A DISTRIBUTED AND SCALABLE APPROACH

Lionel Brunie, Laurent Lefèvre et Olivier Reymann
Laboratoire de l'Informatique du Parallélisme
Ecole Normale Supérieure de Lyon
69364 LYON Cedex 07 , France
(lbrunie, llefevre, oreymann)@lip.ens-lyon.fr

SPDT 96 - 22th May 1996

Plan

- What is a DSM system ?
- The DOSMOS system
- Why monitoring a DSM application ?
- A model for DSM application monitoring
  - Trace detection and collection : ***Event Manager Process***
  - Trace management : ***Meta-Object concept***
  - Analysis and visualization of execution traces: ***Visualization process***
- Implementation and architecture
  - Meta-Objects
  - System architecture
- Protocols : write, read, acquire, release operation protocols
- Experiments
- Conclusion and futur works

What is a DSM system ?

- Designed for distributed memory architectures
- Allows transparent accesses to shared data
  - Virtual shared memory systems
  - Object based distributed shared memory systems
- Hides the inter-process communication

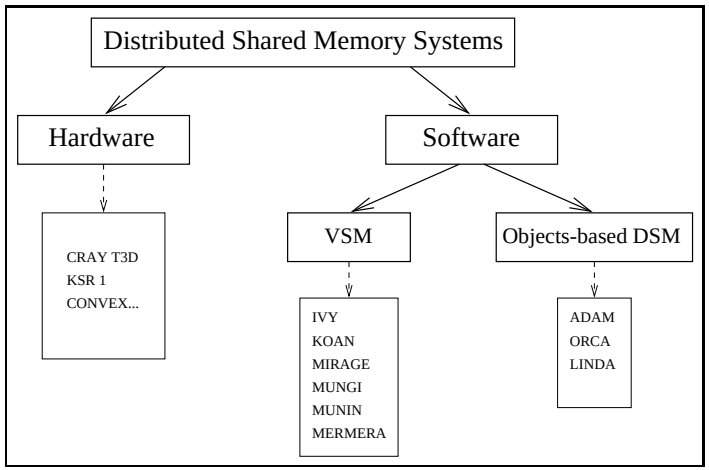


Figure 1: A taxonomy of DSM systems

The designing goals of DOSMOS

- Providing an easy-to-use programming environment
- Breaking the centralization of the memory accesses
- Scalability
- Portability

Basics of DOSMOS

- Dedicated processes (AP, MP...)
- Array allocation
- Hierarchical structuring of application processes
- Optimized weak consistency protocols
- Mixing PVM and DOSMOS code
- From distributed systems to parallel machines

About Array splitting up

```
shared float A(14,20)[3,5];
```

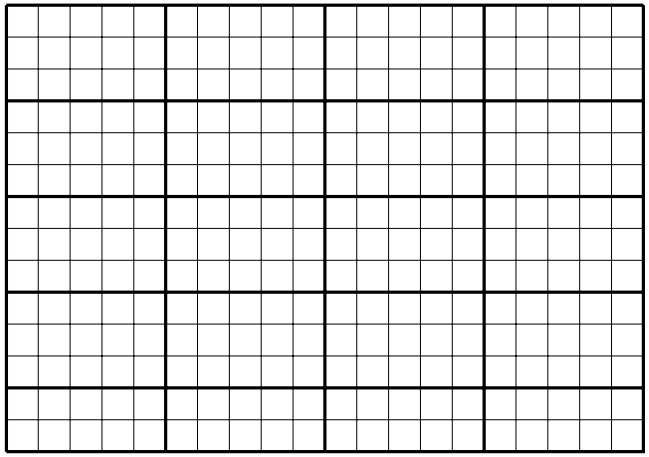


Figure 2: Example of a split matrix

- Increase of the number of objects
- Improve the object distribution over the processors
- reduce the false sharing and the access bottleneck

Clustering using hierarchic groups

Processes that share a common set of variables are gathered.

$\Longrightarrow$ Limit the number of variable copies in the system

$\Longrightarrow$ Reduce the cost of the consistency maintenance

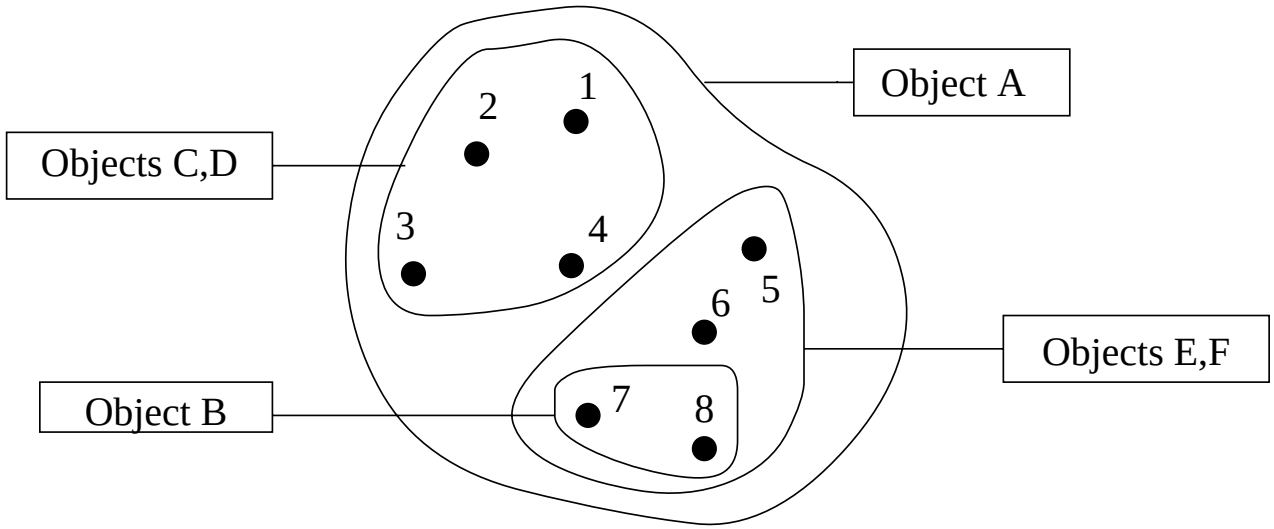


Figure 3: Example of a hierarchic group structure

DOSMOS architecture

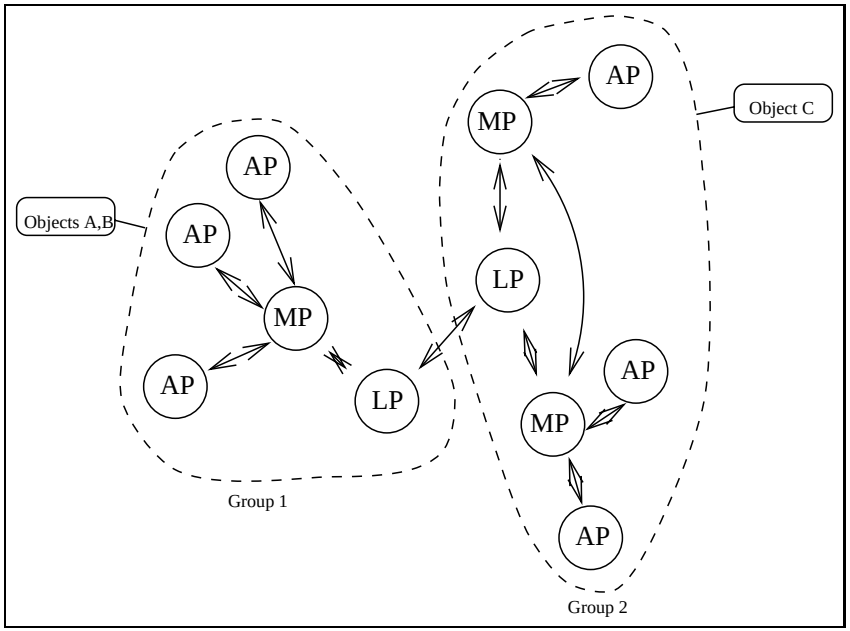


Figure 4: DOSMOS system: an example of software configuration with two groups and three objects A, B and C

Why monitoring a DSM application ?

- Informations :
  - DSM system administration
  - Shared data
- Detections :
  - Bottlenecks
  - Ping-Pong effects
  - No-sharing
  - Specific features

A model for DSM application monitoring

Trace detection and collection : ***Event Manager Process***

This process takes charge of the collection of the information concerning traces and its management.

- Do not overload the DSM system with the monitoring task
- Generate distributed trace files
- Easy to implement
- Definition of a communication protocol between the system and the monitoring tool

Trace management

- To store the events in a file on disk
  - For a post-mortem use
- Use a dedicated structure : the **Meta-Object structure**

  This is a data structure that contains trace information about the operations performed on a variable.
  - Such a structure is linked to each variable
  - Managed by an Event Manager Process
  - Data stored in a judicious way
  - Adapted for on-line analysis and optimization

DOSMOS-Trace architecture

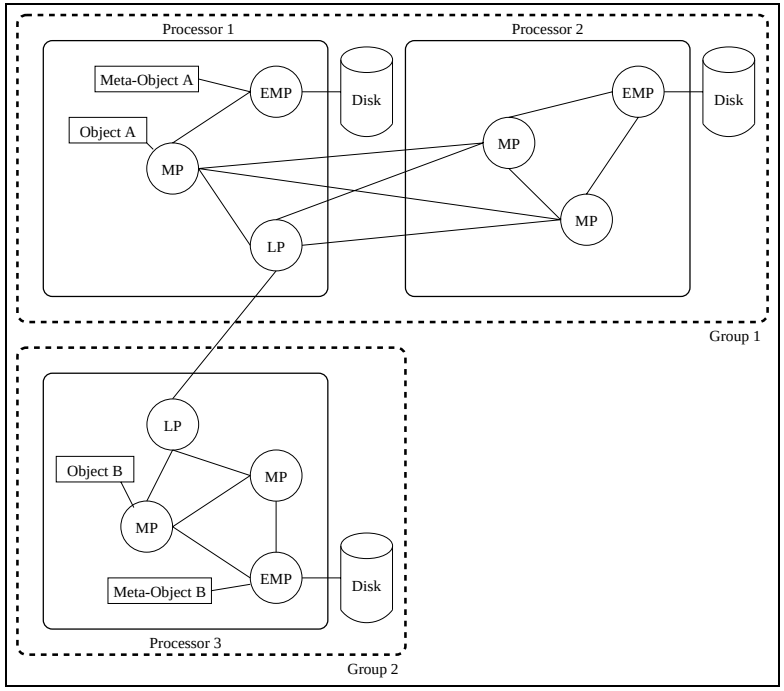


Figure 5: DOSMOS-Trace: example of monitoring environment

Protocols

Write operations

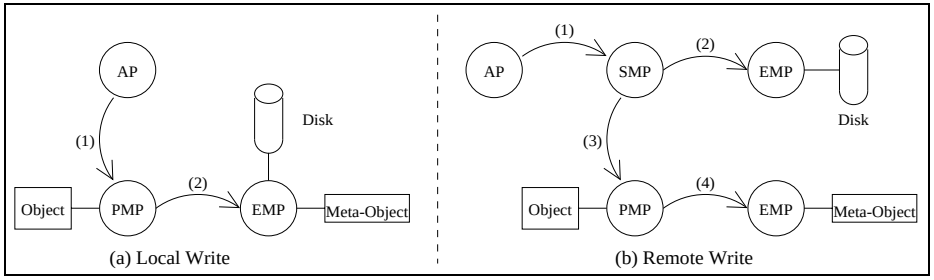


Figure 6: Protocol implemented to collect the trace information about a write operation.

Read operations

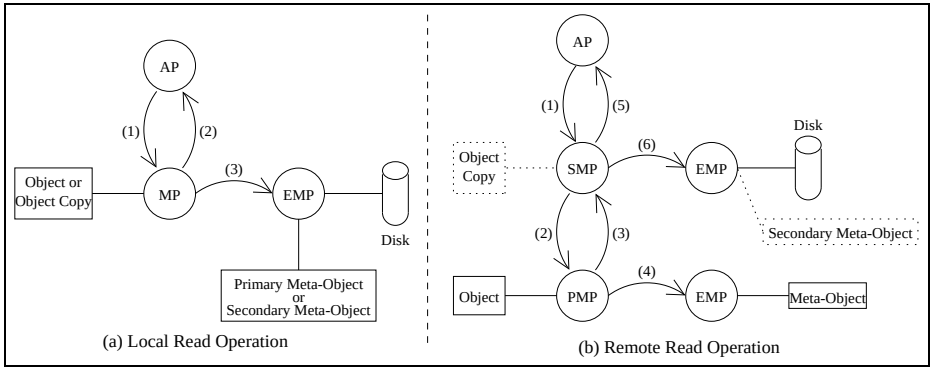


Figure 7: Protocol implemented to collect the trace information about a read operation.

Acquire operations

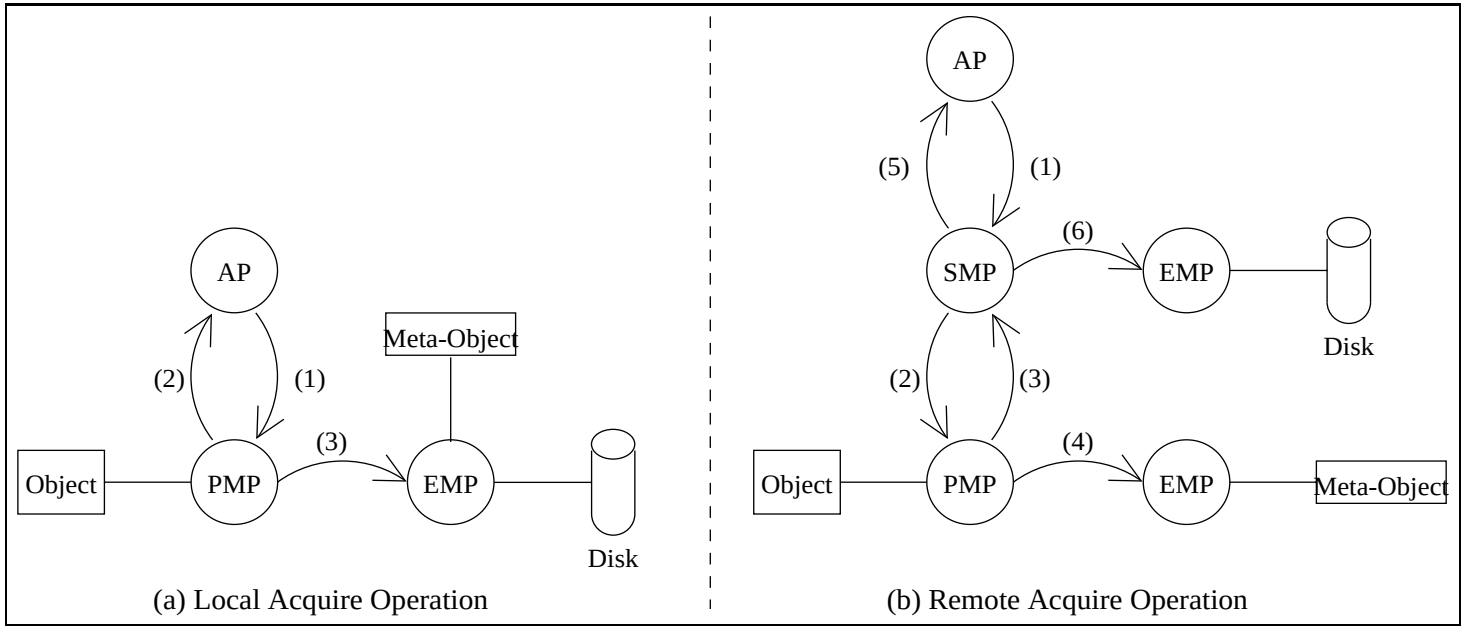


Figure 8: Protocol implemented to collect the trace information about an Acquire operation.

Release operations

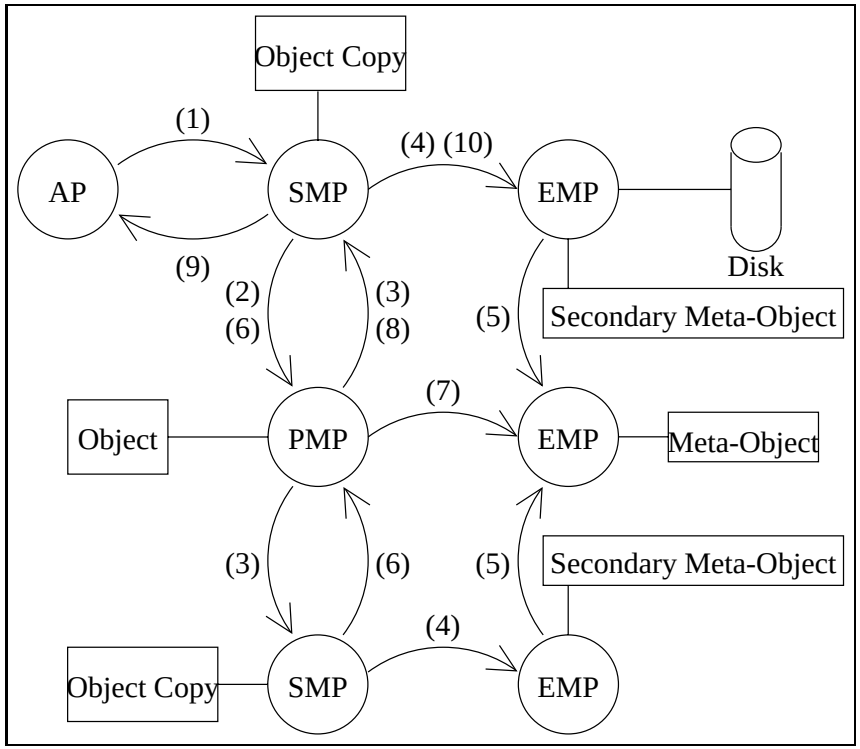


Figure 9: Protocol implemented to collect the trace information about a Release operation.

Analysis and visualization of execution traces

The ***Visualization process*** provides two kinds of diagrams

- Diagrams showing variable accesses according to time
- Diagrams showing statistical results on variable accesses

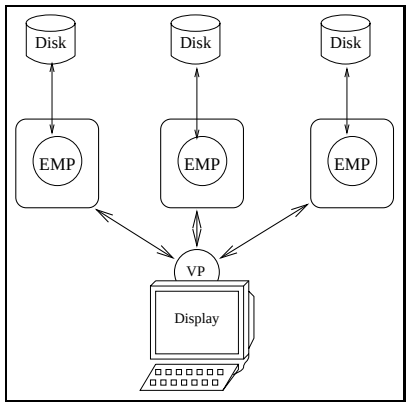


Figure 10: DOSMOS-Trace: the Visualization Process (VP) $\Longrightarrow$ E.M.P.

Experiments

Access to shared objects

Diagrams showing statistical results on variable accesses

- Number of readings during a time interval
- Number of writings during a time interval

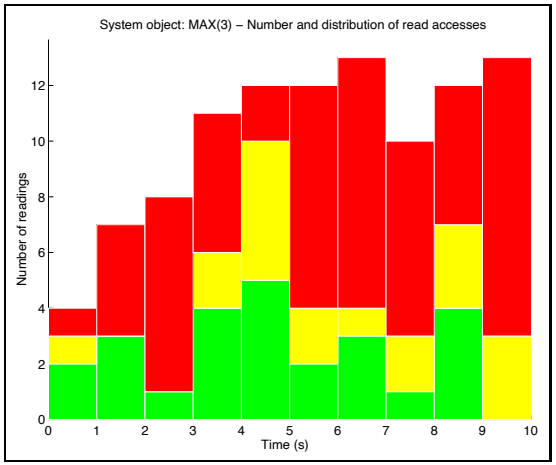


Figure 11: Number and origin of the read accesses performed on an object vs execution time (in black: inter-group accesses)

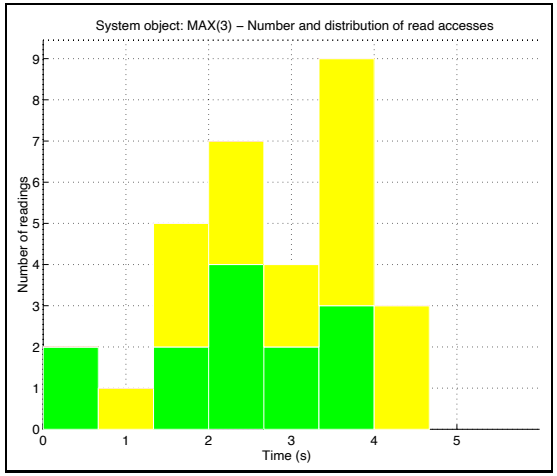


Figure 12: Number and origin of the read accesses performed on an object vs execution time (note this execution does not include inter-group accesses)

Histories

Diagrams showing variable accesses according to time

- History of accesses performed by the processes
- History of accesses performed on the variables

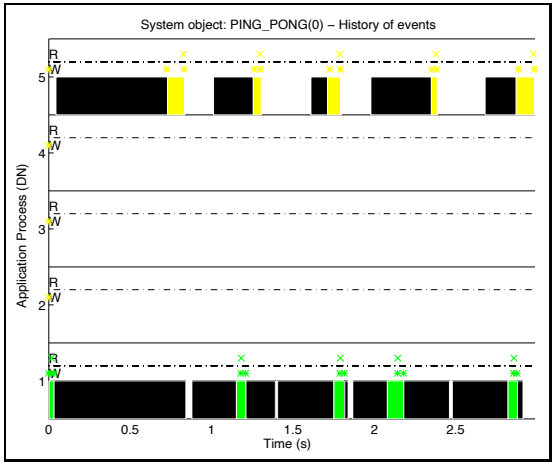


Figure 13: Object activity vs execution time

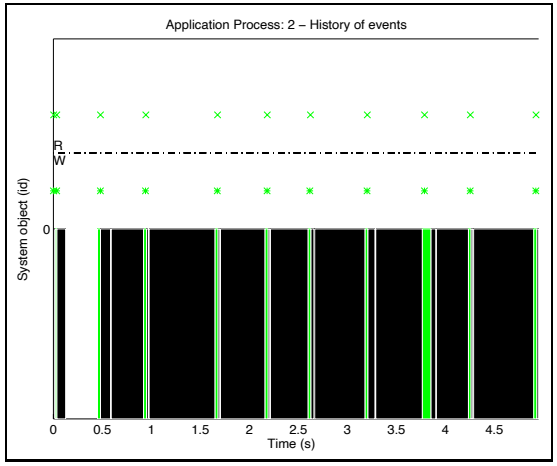
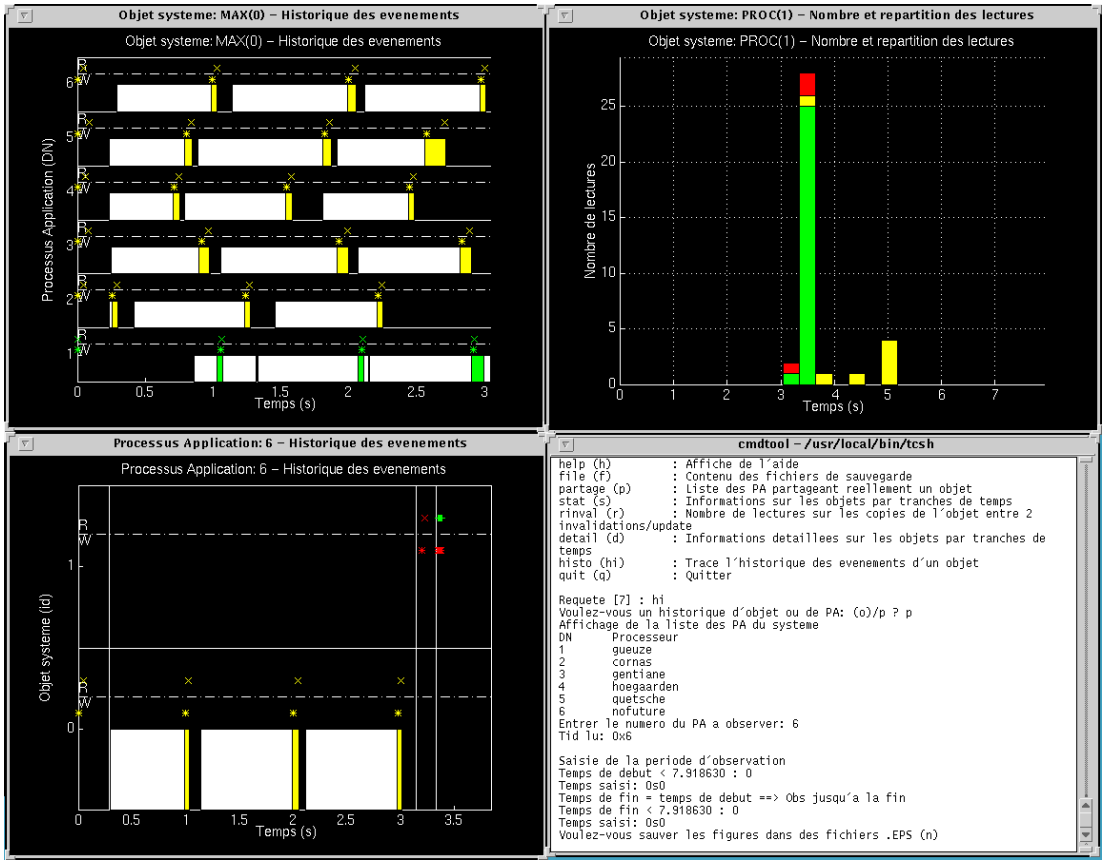


Figure 14: Process activity vs execution time

DOSMOS-Trace environnement



Intrusion

Configuration	1	2	3	4
Execution time	21.90	40.00 (+83%)	48.20 (+120%)	28.30 (+29%)

Table 1: Execution time (in seconds) for several configurations

Ratio \ Configuration	1	2	3	4
R=1	43.80	61.90 (41%)	70.10 (60%)	50.20 (15%)
R=2	65.70	83.80 (28%)	92.00 (40%)	72.10 (10%)
R=3	87.60	105.70 (21%)	113.90 (30%)	94.00 (7%)

Table 2: Calculated execution time (in seconds) for different (computation/shared data access) ratios

Conclusion

- Weak intrusion
- Flexibility
- System scalability
- Meta-object : good storage structure for on-line analysis
- User-orientation
- Independence

$\Longrightarrow$ *A distributed debugger for DSM systems*

Adress: ***http://www.ens-lyon.fr/ llefevre/DOSMOS/Dosmos.html***