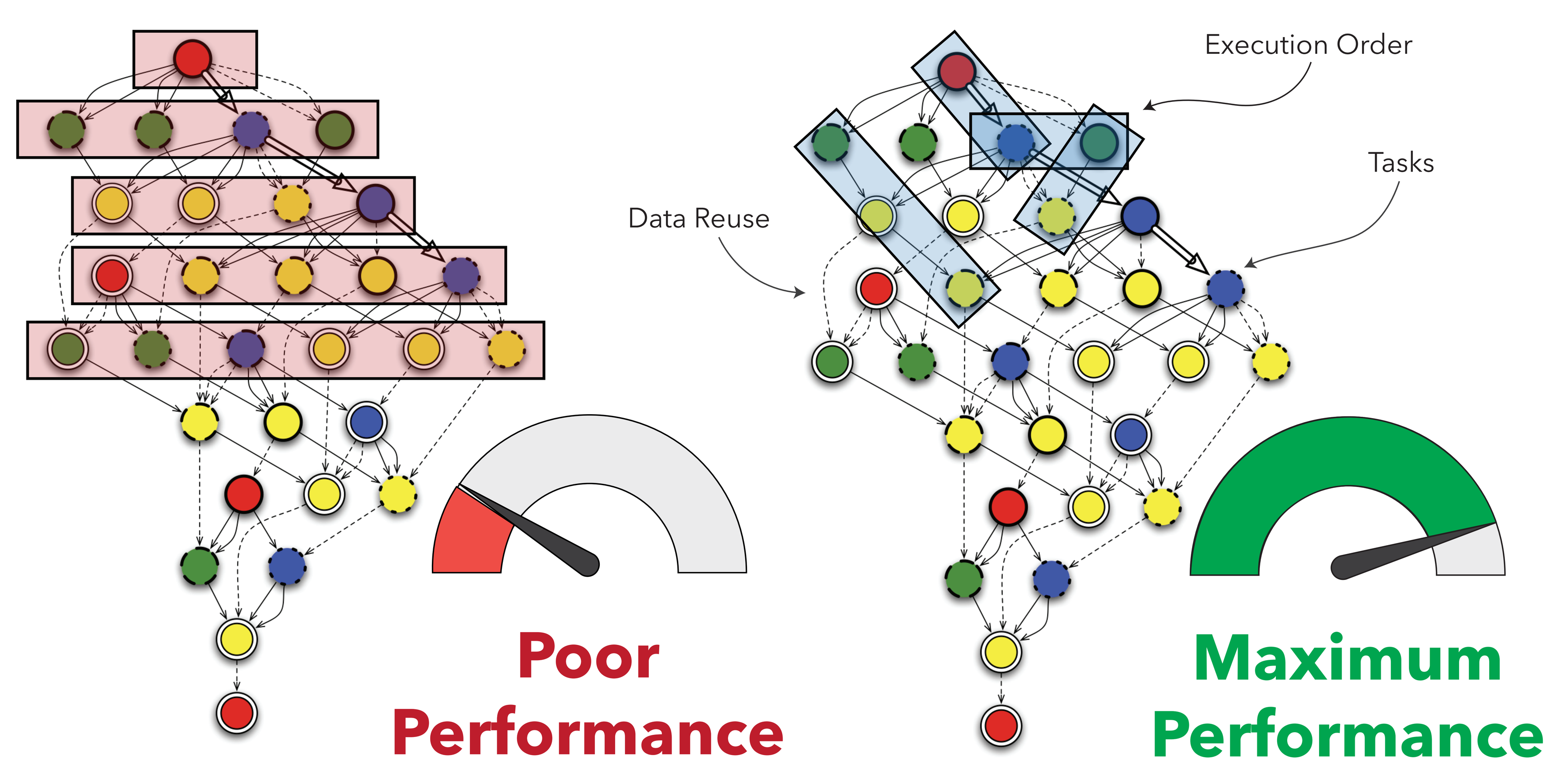
UPPSALA UNIVERSITET

How does **memory** affect **performance** of tasks?

Germán **Ceballos**, Andra **Hugo**, David **Black-Schaffer**

firstname.lastname@it.uu.se

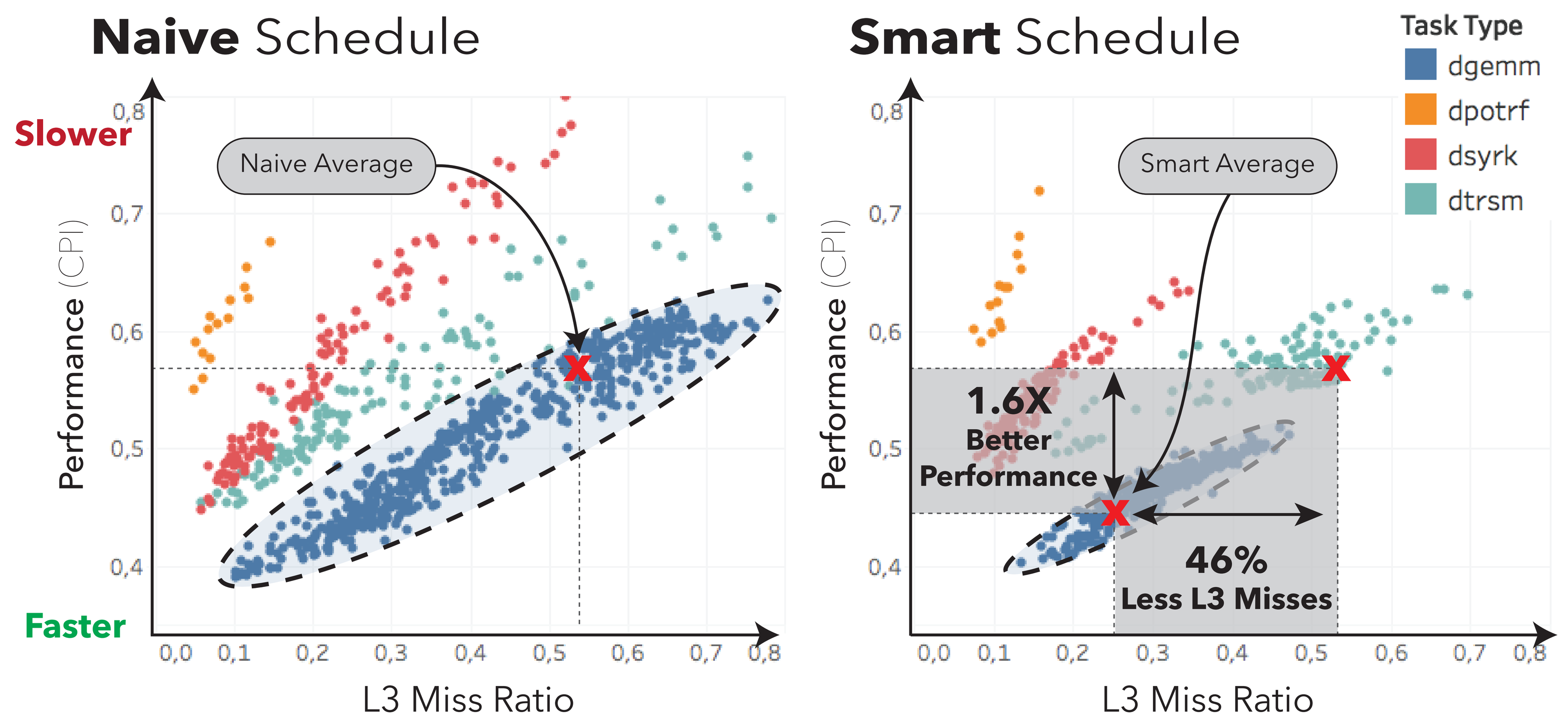
1 **Problem**: Different schedules, different performance



- Different schedules for the **same task-based application** (e.g. Cholesky Factorization)
- Executions show up to **30% performance difference!**
- Scheduling affects **memory behaviour** of the application.

WHY?

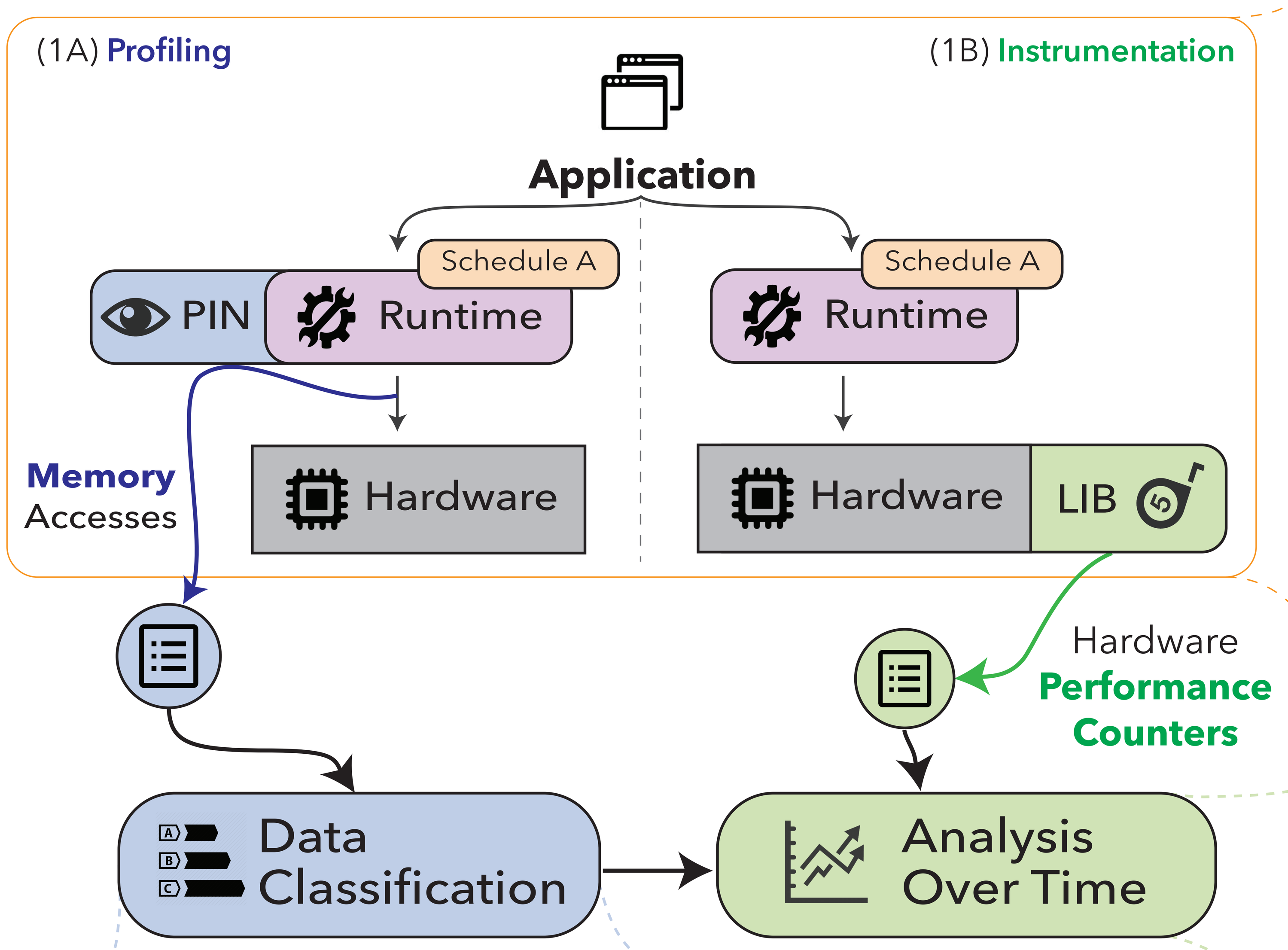
What is the difference?



- Tasks in Naive Schedule miss 46% more in the last level cache.
- Smart Schedule has 1.6x better performance due to **better cache reuse**.
- Different schedules result on **different memory behaviour**.

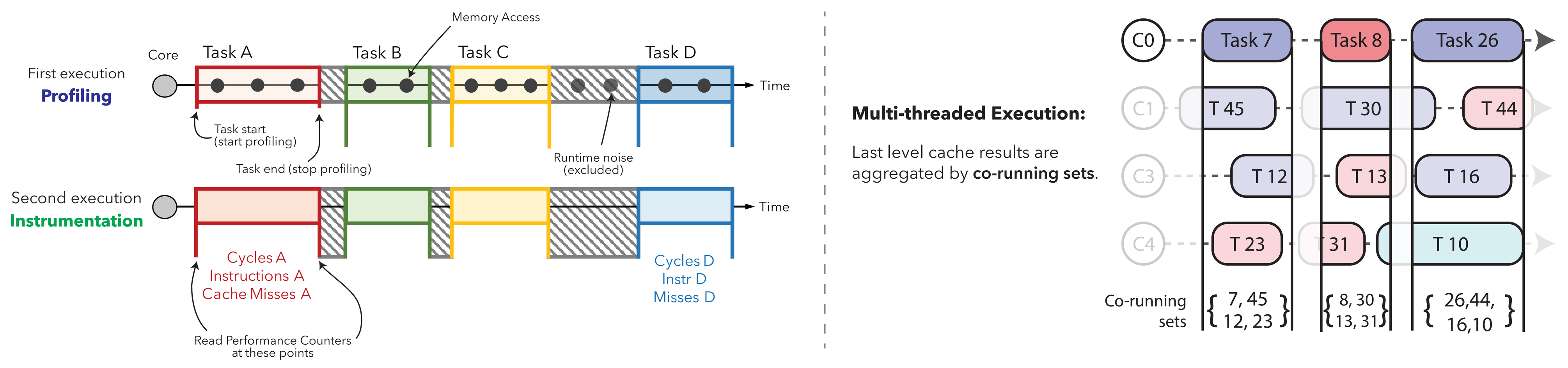
How can we understand why memory affected performance?

2 Task**Insight**: What, When and Why?



1 Profiling and Instrumentation

Step 1: The application is executed twice with **the same schedule**.
(1A) In the first execution memory accesses are saved using a PIN-based tool.
(1B) In the second execution, hardware performance counters are read using library calls.



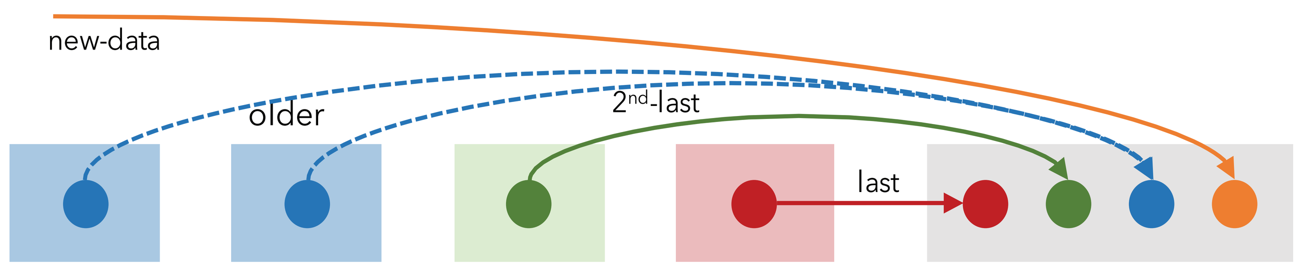
For multi-threaded executions, results are aggregated by **co-running sets:** the set of tasks running at the same time.
We ignore runtime noise by guiding the profiling/instrumentation with the start and end of each task.

2 Data Classification

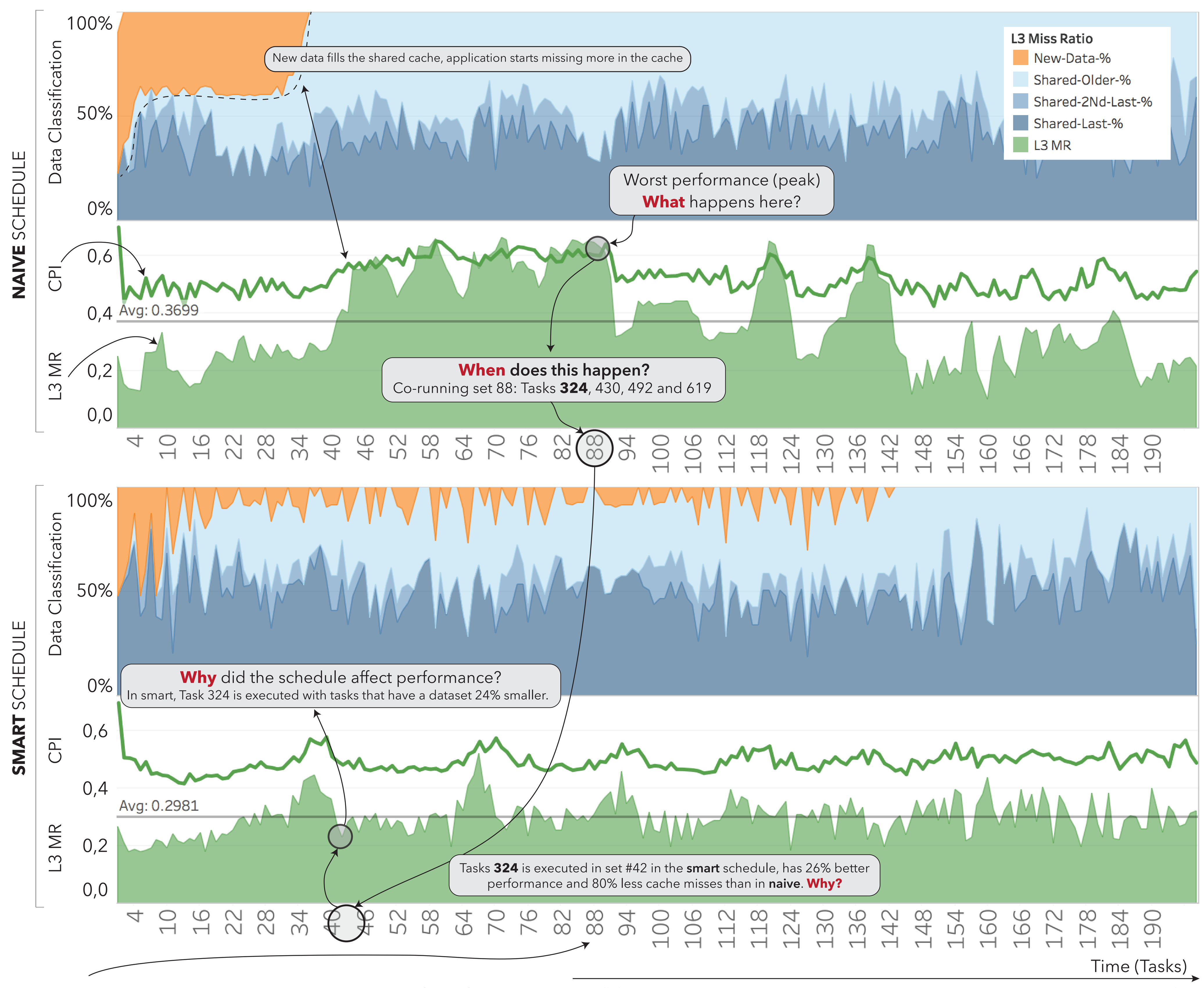
Step 2: After running the application, memory reuses are identified across tasks. Later, each memory access collected is classified depending on the type of reuse:

- **New data:** first time the memory address is accessed.
- **Last reuse:** The memory address was used by the previously executed task.
- **2nd-last reuse:** The address was used by the second-to-last task.
- **Older reuse:** The memory address was used before.

This classification is displayed over time during the data analysis step, and connected with performance information captured from hw peformance counters.



3 Analysis Over Time



Co-running sets: The execution is represented as a sequences of sets of tasks running in parallel. E.g. The 86th co-running set comprises tasks 324, 430, 492 and 619 in the naive schedule.

Step 3: Data classification (Step 2) is correlated with information from hardware performance counters and displayed over time. By doing this, Task**Insight** can expose if changes in **memory** behaviour had an impact in performance, **when** and **why**.

3 Conclusion

- Schedules affect **memory behaviour** and **performance** up to 30%.
- Current tools rely on programmers intuition, don't help to understand.
- Task**Insight** exposes effects of **scheduling** on **memory** and **performance** and can be used to produce better schedules.
- Task**Insight** shows performance differences (**what**) on the same tasks across schedules, **when** this happened, and **why**.