SMP Trace Display Options

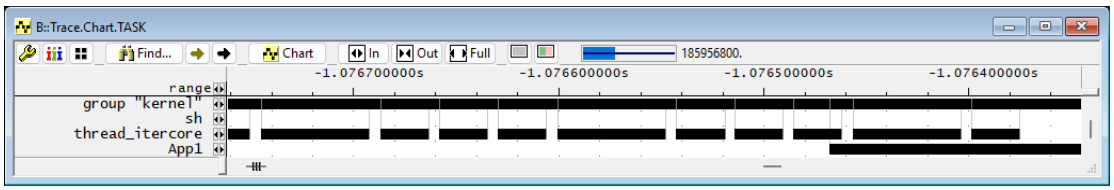
2025-10-15 - Comments (0) - Trace

TRACE32 PowerView provides different display options for analyzing trace results on Symmetric Multi-Processing (SMP) systems. The most suitable option depends on the target software and the user's analysis goals.

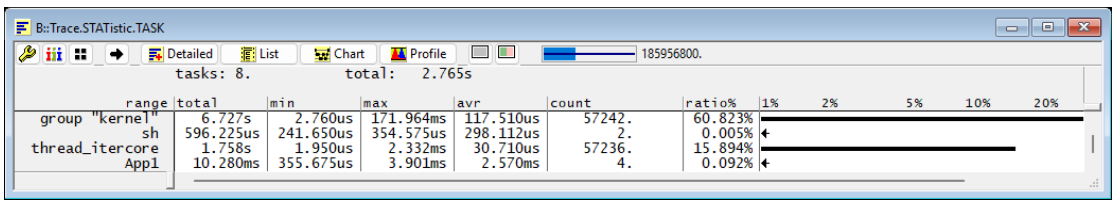
The default display option is **MergeCORE**, where timing is calculated separately for each core, but the results are presented as a combined summary across all cores.

To illustrate this, consider a Linux system running on a target with four cores. For demonstration purposes, we created an artificial example where a task is forced to switch execution in a loop from one core to the next. This setup makes the behavior clearly visible without depending on naturally occurring core switches.

In this scenario, the timing diagram of the task run-times is displayed in the default **MergeCORE** view, where the execution times are aggregated across all cores:



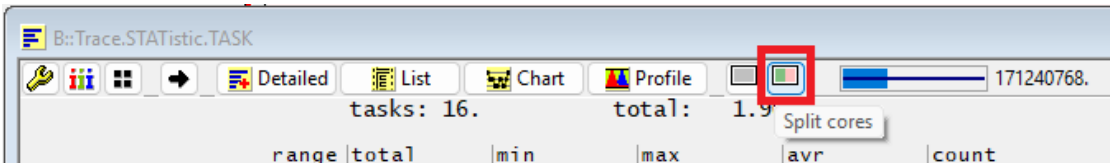
The corresponding numerical view is shown below. It indicates that the task *thread_itercore* executed for 1.758 seconds, which corresponds to 15.894% of the total system run-time. This percentage reflects the share of overall CPU time consumed by the task.



An alternative display mode is **SplitCORE**, where timing is calculated and displayed separately for each core.

You can switch to this mode by selecting the **"Split cores"** button in the

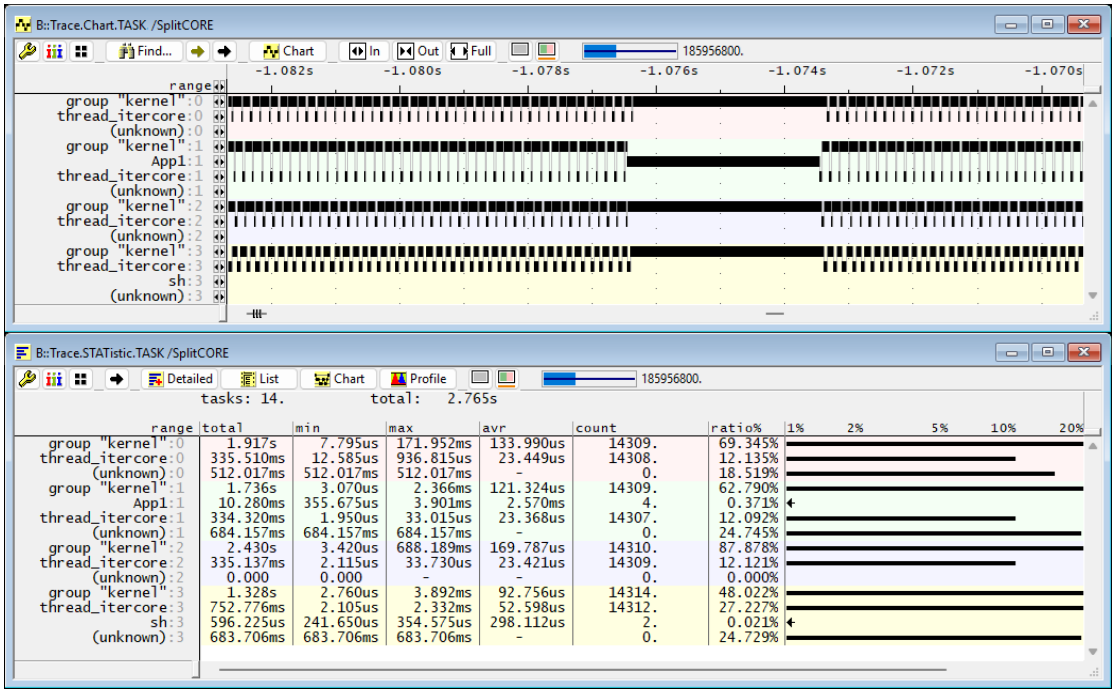
trace display window.



The display mode can also be selected using the **/SplitCORE** option, which is supported by all trace display commands:

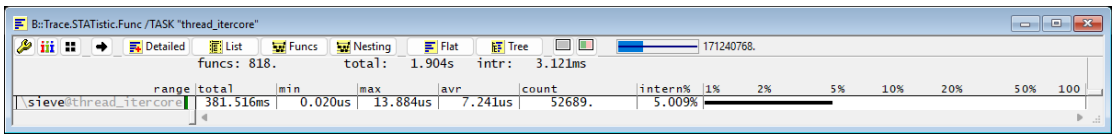
```
Trace.Chart.TASK /SplitCORE
```

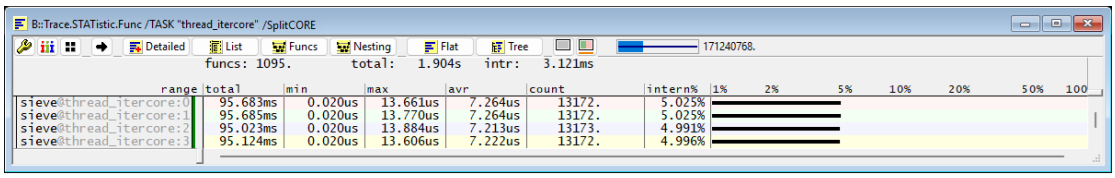
Switching to the **SplitCORE** view in the previous example produces the following results:



In this view, the run-time of the task *thread_itercore* is displayed separately for each core. The total run-time of 1.758 seconds corresponds to the sum of the task's execution times across all four cores.

Function run-times can be analyzed in the same way as task run-times, using either display mode. For example, the run-time of the *sieve* function within the *thread_itercore* task is shown below in both the **MergeCORE** and **SplitCORE** views, illustrating how the execution time is aggregated across all cores or displayed separately for each core:

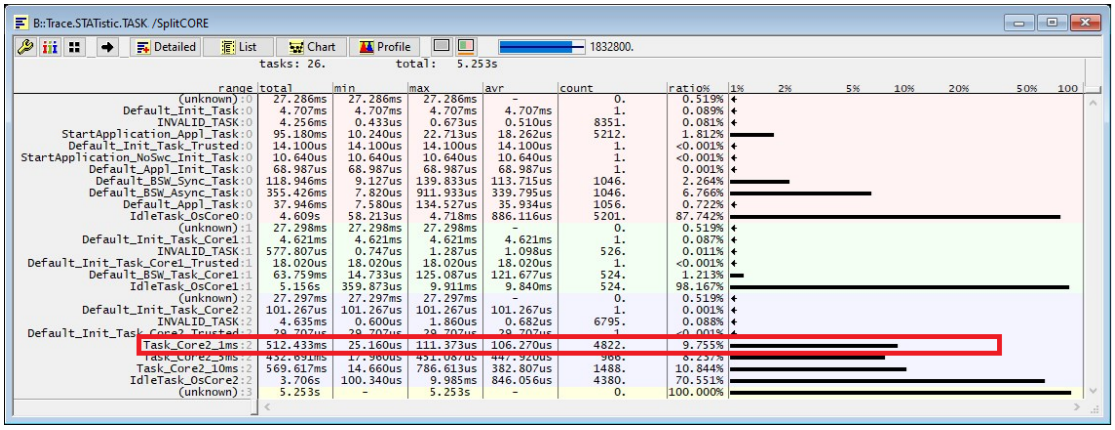




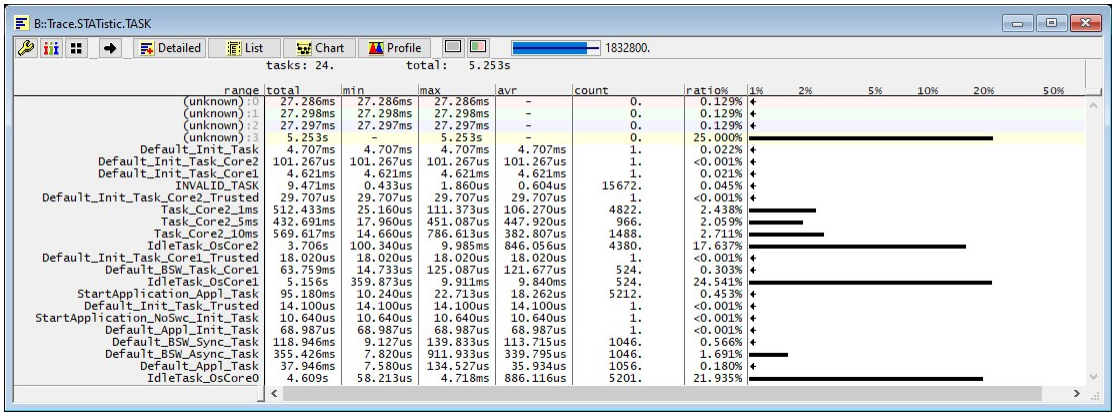
For dynamic SMP systems such as Linux, where tasks and functions may be executed across multiple cores, the default **MergeCORE** view is generally more suitable. It allows the user to focus on the total execution time of a task or function, independent of the kernel's core scheduling decisions.

In contrast, the **SplitCORE** view is useful when execution is strictly bound to specific cores, as is often the case in AUTOSAR Classic Platform systems. This mode enables detailed analysis of per-core behavior and load distribution.

The following screenshot shows an example of an AUTOSAR Classic Platform system running on 4 cores. It clearly illustrates for instance that the task *Task_Core2_1ms* is running on core 2 with its corresponding runtimes.



The default **MergeCORE** view shows in this case the same run-time results, however without the core information:

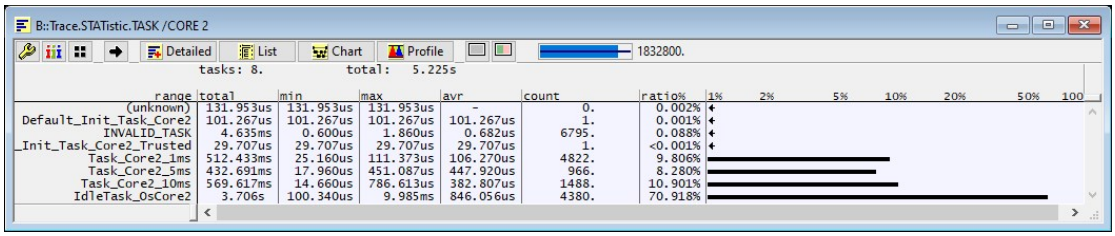


Note

It is also possible to display the trace results for each core in one separate

window using the **/CORE < number >** option, e.g.:

```
Trace.STATtistic.TASK /CORE 2
```

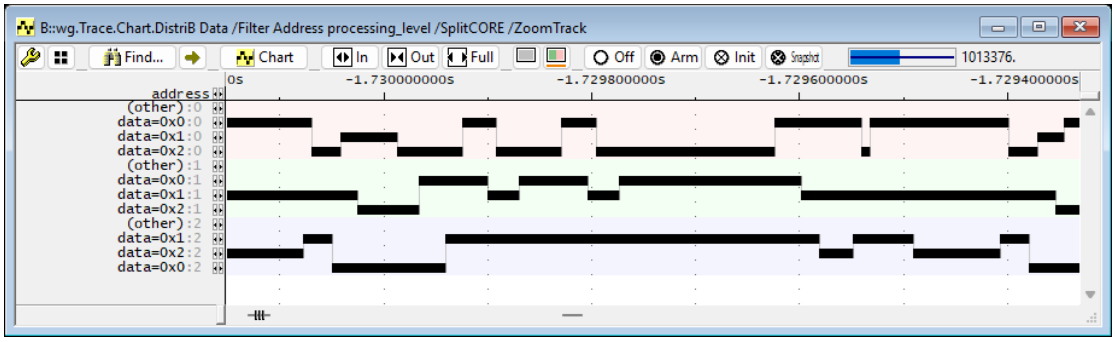


A third display option, which can be useful in specific scenarios, is **JoinCORE**. When this option is enabled, the analysis treats all cores as a single unified entity, effectively ignoring the core information.

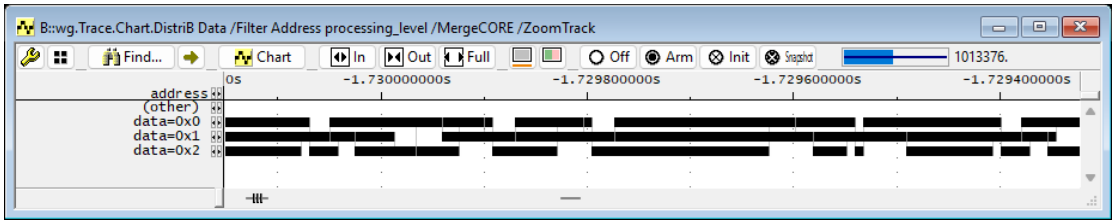
This option is particularly valuable when you want to measure the timing of individual events that can occur on different cores.

In the following example, we perform a distribution analysis on a global variable named *processing_level*, which represents the state of a state machine. This variable is written by all three cores and can take the values 0, 1, or 2.

The **SplitCORE** display shows the values written by each core separately, as illustrated in the screenshot below:

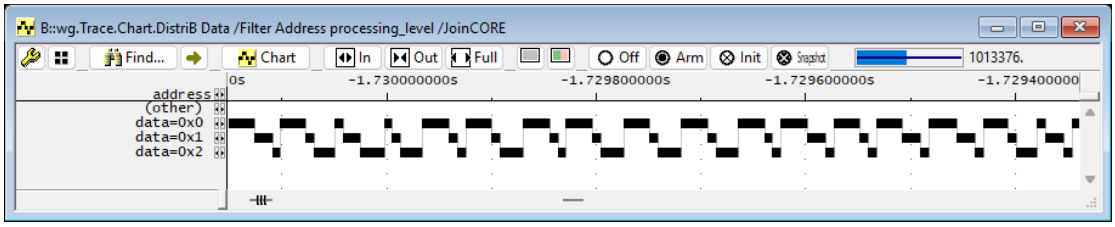


The **MergeCORE** option, on the other hand, merges the data from all three cores into a single timeline but does not provide a true system-level view.



By contrast, **JoinCORE** provides the desired view for this use case. It allows you to observe the variable values independently of which core

performed the write access. In this example, the result clearly shows for instance that the transitions consistently follow the sequence 0 → 1 → 2.



Note

Keep in mind that **JoinCORE** is not suitable for measuring runtimes that can overlap across multiple cores.

For more information about the various trace display commands and options, refer to General Commands Reference Guide T