Paraver Quick Guide

Content

1 Introduction

The Paraver Quick Guide introduces Paraver's main concepts and analysis views to allow understanding performance reports using Paraver without in-depth knowledge of the Paraver interface and functionality. To analyze the behavior of an application, Paraver reads and displays application traces usually produces with the according trace recorder Extrae.

2 Displays

Paraver presents performance information with two main displays that provide qualitatively different types of information. The timeline display represents the behavior of the application along time and processes or threads. It provides the user a general understanding of the application behavior and simple identification of phases and patterns. The statistics display allows a numerical analysis of the data that can be applied to any user selected region, helping to draw conclusions on where and how to focus the optimization effort.

2.1 Timeline Display

Timelines show the evolving program behavior over time in a two-dimensional chart. The vertical axis includes the executing processes and threads; the horizontal axis represents the runtime of the recorded application. The current state of each executed process or thread is marked by a colored rectangle. These colored rectangles highlight the evolution of a selected metric of the recorded application.
Thereby, the metrics are distinguished in metrics with numerical values and categorical values. Figure 1 shows Paraver's timeline view exemplary for the numerical value 'Useful Duration' that displays the duration of computing phases. Numerical values, such as in Figure 1, are represented with a gradient color ranging from green to blue. Whereas, green shows the minimum and blue the maximum value of the gradient (see bottom of Figure 1). In the case, where minimum and maximum of the color gradient differ from the actual minimum and maximum of the observed value, values below the gradient minimum are marked in yellow and values above the maximum are marked in orange. However, typically the gradient is adjusted to match the actual minimum and maximum of the value.

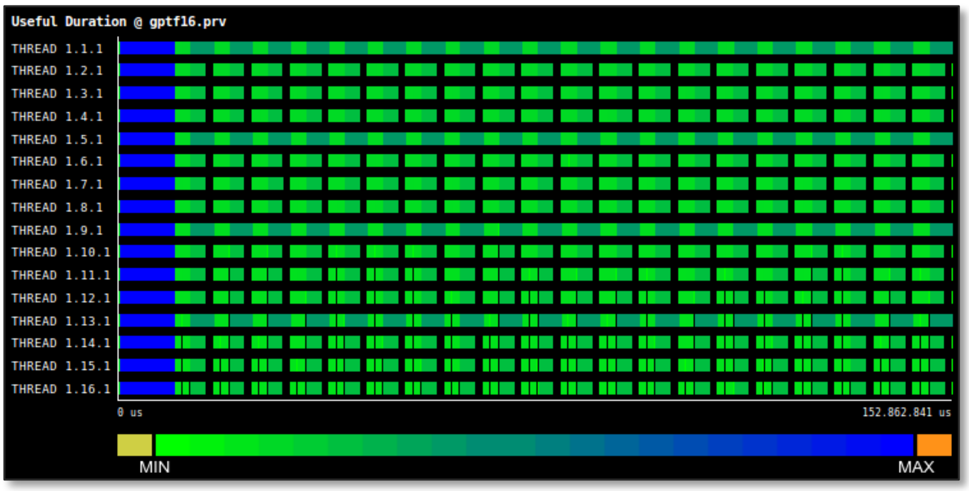


Figure 1: Paraver timeline displaying the numerical value 'Useful Duration' as gradient color from green (minimum) to blue (maximum) for the different states of the application. Yellow and orange mark values below minimum or above maximum gradient value, respectively.

For categorical values, a color is assigned to each recorded value of the parameter. Figure 2 displays Paraver's timeline view for the active MPI call, where each color represents a different MPI call as shown on the right of Figure 2.

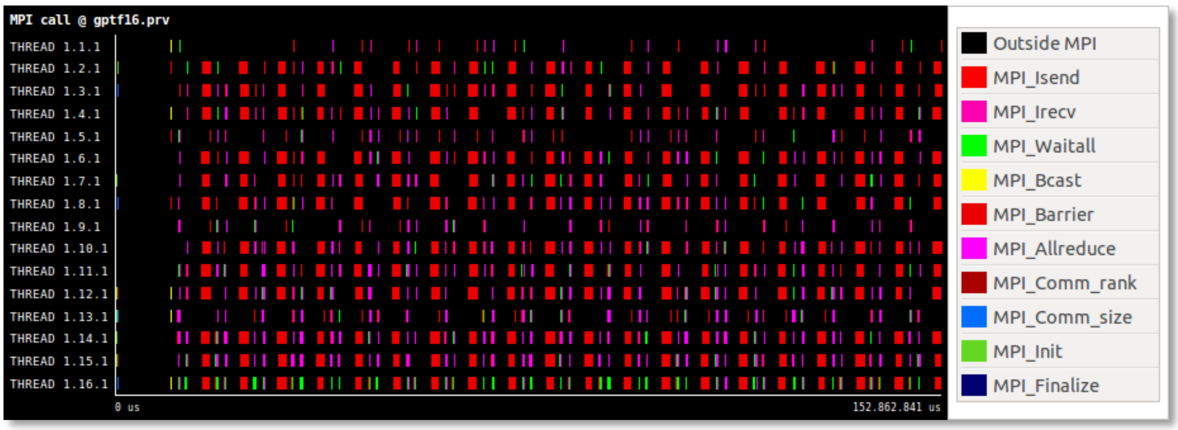
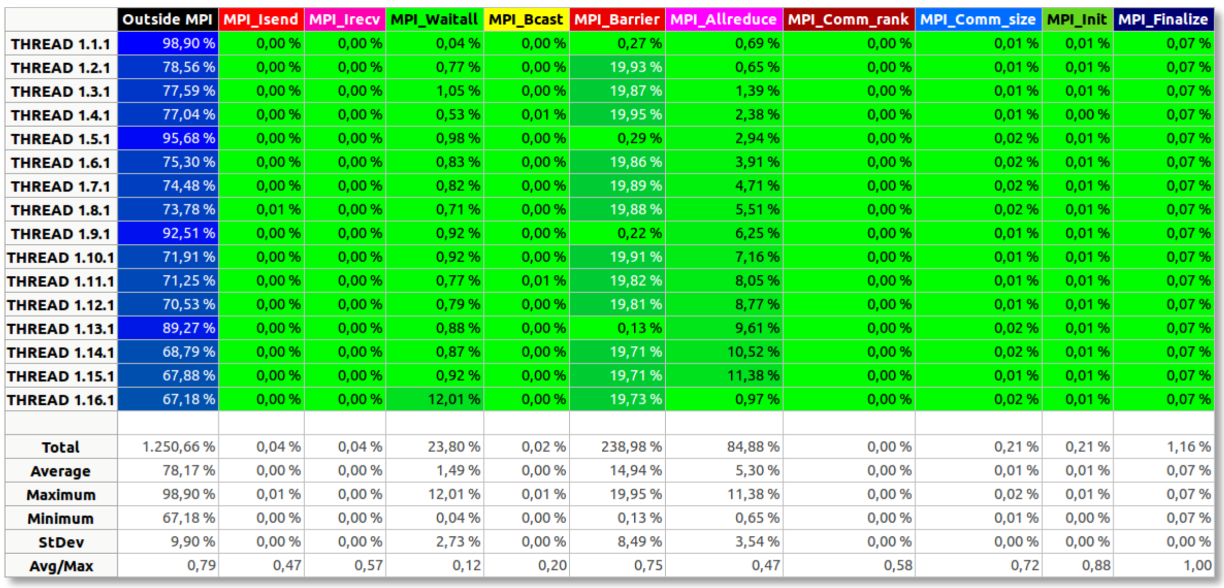


Figure 2: Paraver timeline displaying the categorical value 'MPI call' with each color representing one value of the state, in this case, the active MPI call.

2.2 Statistics Display

Next to timelines that present qualitative behavior, Paraver's statistics displays allow a quantitative analysis. The two-dimensional analysis computes different statistics for metrics with numerical or categorical values. It includes typical statistics, such as distribution, minimum, maximum, or average for a state's runtime or number of occurrences. The statistics are presented in the form of tables with the state's value constituting the columns and the executing processes or threads the rows. Figure 3 shows an analysis of the runtime spent within the different MPI calls and the time spent outside MPI calls, i.e. time spent in computation. For categorical values, such as type of MPI call, each value is represented by one column in the table. Each cell contains the statistics value, in this case the percentage of runtime, which is also color-coded with the green-blue gradient. In addition, the accumulated values (bottom of Figure 3) allow an overview of the different executed processes or threads including the balancing between the processes or threads (last row).



	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Barrier	MPI_Allreduce	MPI_Comm_rank	MPI_Comm_size	MPI_Init	MPI_Finalize
THREAD 1.1.1	98,90 %	0,00 %	0,00 %	0,04 %	0,00 %	0,27 %	0,69 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.2.1	78,56 %	0,00 %	0,00 %	0,77 %	0,00 %	19,93 %	0,65 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.3.1	77,59 %	0,00 %	0,00 %	1,05 %	0,00 %	19,87 %	1,39 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.4.1	77,04 %	0,00 %	0,00 %	0,53 %	0,01 %	19,95 %	2,38 %	0,00 %	0,01 %	0,00 %	0,07 %
THREAD 1.5.1	95,68 %	0,00 %	0,00 %	0,98 %	0,00 %	0,29 %	2,94 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.6.1	75,30 %	0,00 %	0,00 %	0,83 %	0,00 %	19,86 %	3,91 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.7.1	74,48 %	0,00 %	0,00 %	0,82 %	0,00 %	19,89 %	4,71 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.8.1	73,78 %	0,01 %	0,00 %	0,71 %	0,00 %	19,88 %	5,51 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.9.1	92,51 %	0,00 %	0,00 %	0,92 %	0,00 %	0,22 %	6,25 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.10.1	71,91 %	0,00 %	0,00 %	0,92 %	0,00 %	19,91 %	7,16 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.11.1	71,25 %	0,00 %	0,00 %	0,77 %	0,01 %	19,82 %	8,05 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.12.1	70,53 %	0,00 %	0,00 %	0,79 %	0,00 %	19,81 %	8,77 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.13.1	89,27 %	0,00 %	0,00 %	0,88 %	0,00 %	0,13 %	9,61 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.14.1	68,79 %	0,00 %	0,00 %	0,87 %	0,00 %	19,71 %	10,52 %	0,00 %	0,02 %	0,01 %	0,07 %
THREAD 1.15.1	67,88 %	0,00 %	0,00 %	0,92 %	0,00 %	19,71 %	11,38 %	0,00 %	0,01 %	0,01 %	0,07 %
THREAD 1.16.1	67,18 %	0,00 %	0,00 %	12,01 %	0,00 %	19,73 %	0,97 %	0,00 %	0,02 %	0,01 %	0,07 %
Total	1.250,66 %	0,04 %	0,04 %	23,80 %	0,02 %	238,98 %	84,88 %	0,00 %	0,21 %	0,21 %	1,16 %
Average	78,17 %	0,00 %	0,00 %	1,49 %	0,00 %	14,94 %	5,30 %	0,00 %	0,01 %	0,01 %	0,07 %
Maximum	98,90 %	0,01 %	0,00 %	12,01 %	0,01 %	19,95 %	11,38 %	0,00 %	0,02 %	0,01 %	0,07 %
Minimum	67,18 %	0,00 %	0,00 %	0,04 %	0,00 %	0,13 %	0,65 %	0,00 %	0,01 %	0,00 %	0,07 %
StDev	9,90 %	0,00 %	0,00 %	2,73 %	0,00 %	8,49 %	3,54 %	0,00 %	0,00 %	0,00 %	0,00 %
Avg/Max	0,79	0,47	0,57	0,12	0,20	0,75	0,47	0,58	0,72	0,88	1,00

Figure 3: Paraver statistics display showing the runtime distribution of MPI calls (exemplary for categorical values) and the time spent outside MPI calls, i.e. time spent in computation. The percentage values are color-coded with the green-blue gradient.

For statistics based on numerical values, such as the duration of individual states, columns contain value ranges. This representation is also called histogram. Figure 4 shows the statistics display for the duration of states outside MPI, called ‘Useful Duration’. Again the rows contain the executing threads but the columns represent ranges of state durations. In this case, the color gradient represents the instructions per cycle (IPC) for each state to distinguish between different states, i.e. entries of similar color and duration represent most likely the same application phase. This way, histograms are an effective tool to identify imbalances between processes or threads. For instance, the histogram in Figure 4 shows two imbalances marked with violet boxes: in the first, the duration is constantly decreasing for increasing process numbers; in the second, there are two groups of processes, where process 1,5,9, and 13 show a significantly longer duration than the remaining processes. The origins of the imbalance are detailed in **Error! Reference source not found.**.
Furthermore, the statistics can be extended to three-dimensional statistics with an additional metric to further categorize the displayed values, e.g. filter values by the third metric.

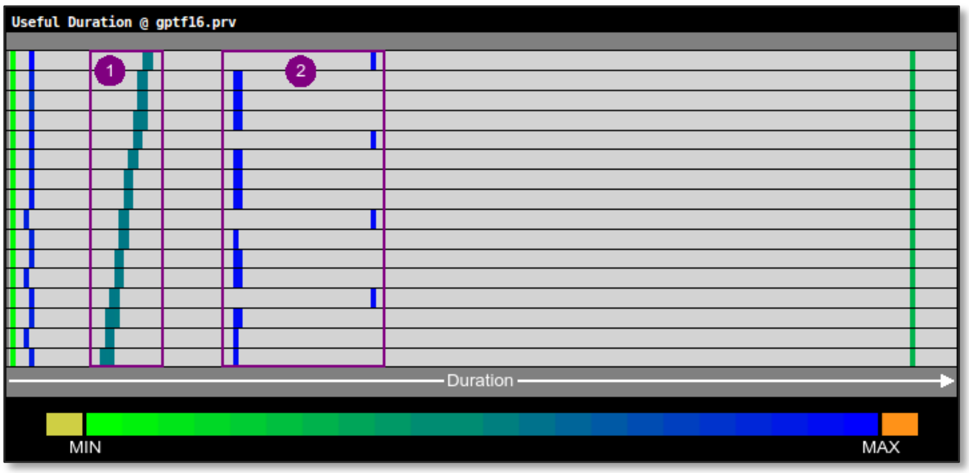


Figure 4: Paraver histogram displaying the distribution of two values of the states of the application: The horizontal axis represents the first value, in this case, the ‘Useful Duration’. The color gradient represents the second value, in this case, the ‘instructions per cycle (IPC)’.

3 Clustering

Clustering, or cluster analysis, is applied to detect different trends in an application's computation phases with minimum user intervention. This detection provides an insight into the application behavior and is capable to reveal the computation structure of an application. The result of a cluster analysis usually consists of two main displays: first, a timeline display that shows the compute regions colored according to the cluster they belong and a two-dimensional plot representing the single values contained in the clusters.
Figure 5 shows the timeline display with four different clusters (dark green, light green, yellow, and red). It reveals the structure of the application, in this case, an initialization phase (dark green), followed by 20 iterations. The iterations consist of three phases: two phases where the duration decreases with increasing process number (light green) and a third phase where processes 1,5,9, and 13 run significantly longer than the rest (red and yellow).

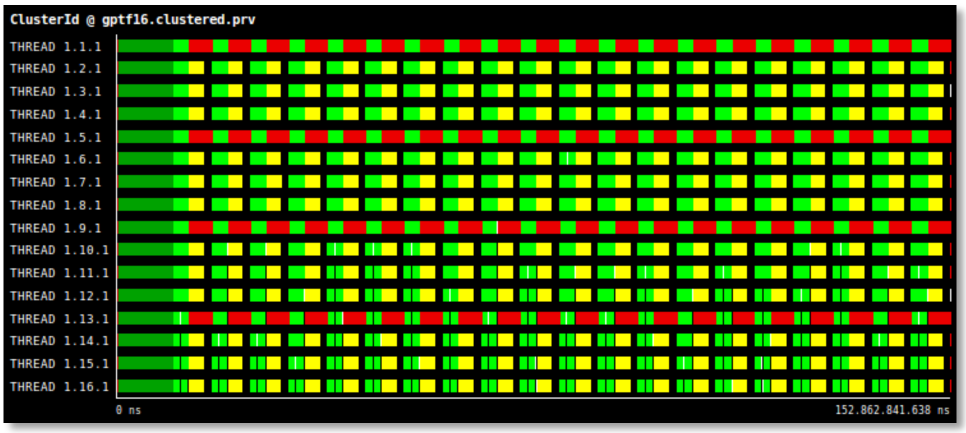


Figure 5: Timeline display highlighting the application structure: an initialization phase (dark green), followed by 20 iterations.

The two-dimensional plot that contains the single values that form each cluster allows reviewing the structure and quality of the clustering. Figure 6 shows the plot for the above cluster analysis, where the clustering is based on the number of instructions of each phase and their IPC. In this case, the four different clusters are clearly separated: The dark green cluster executes the fewest instructions and achieves the lowest IPC. The light green cluster completes about 5 billion instructions with approx. 1.3 instructions per cycle. The red and yellow clusters show a similar IPC value but the red cluster completes significantly more instructions.

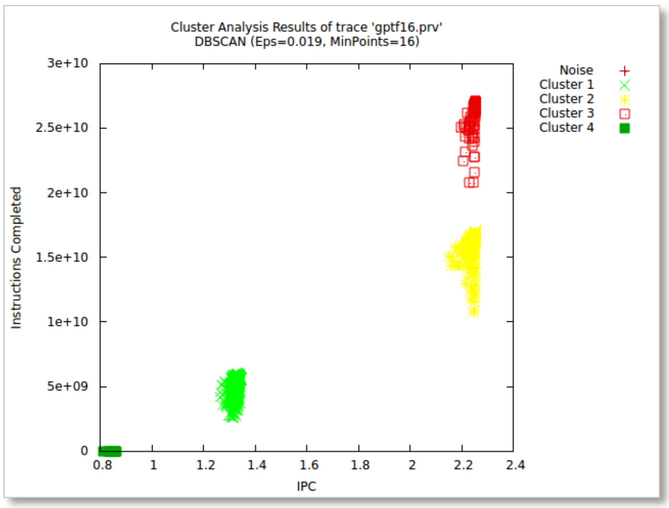


Figure 6: Two-dimensional plot displaying the composition of the four detected clusters.

4 Efficiency Indicators

Paraver provides efficiency indicators that combine different performance metrics to quantify parallel efficiency and scalability with a single percentage value. They allow an easy high-level comparison of different executions, e.g., different application parameters or core counts. There are six performance indicators: load balance, serialization efficiency, transfer efficiency, communication efficiency, parallel efficiency, and global efficiency.

- **Load balance** quantifies the potential efficiency loss caused by imbalances in the computing time by each process or thread. It is computed as the ration between average and maximum computing time.
- **Serialization efficiency** reflects the loss of efficiency caused by dependencies between different processes or threads leading to a serialization of parallel computation. It is computed as the maximum efficiency achieved with instantaneous communication, i.e., zero communication overhead.
- **Transfer efficiency** quantifies the efficiency loss due to actual data transfers.
- **Communication efficiency** reflects the total loss of efficiency due to communication. It combines serialization and transfer efficiency.
- **Parallel efficiency** indicates the overall parallel performance gain by combining load balance and communication efficiency.
- **Computation Scalability** reflects how well the distribution of computational load scales for an increasing number of cores. It compares the time spent in computation, aggregated for all processes, for multiple runs with increasing core counts.
- **Global efficiency** indicates the overall performance of the application by combining parallel and computation efficiency.
- **IPC Scalability** reflects how well computational intensity (IPC = Instructions per cycle) scales for an increasing number of cores.
- **Instructions Scalability** reflects how well the computational load (Number of instructions) scales for an increasing number of cores.

For the above example the load balance is 79%, serialization efficiency is 100%, and transfer efficiency is 98.9% (see also Figure 3). Consequently, the communication efficiency is 98.9% and the combined parallel efficiency is 78.1%. Since the example surveys only a single run, computation scalability, IPC/Instruction scalability, and global efficiency are not relevant.