Visualizing Physical Query Execution in a Distributed Big Data Management System

Umar Javed, Thierry Moreau, Dominik Moritz, Adriana Szekeres

Large-scale distributed databases are facing many challenges: dealing with high communication costs, scheduling computation on available resources, process huge amounts of data, etc. For such systems, the design of the **physical query plan** and the **partitioning of data** are critical to **query performance**. Profiling and identifying potential performance bottlenecks in such systems can be a daunting task.

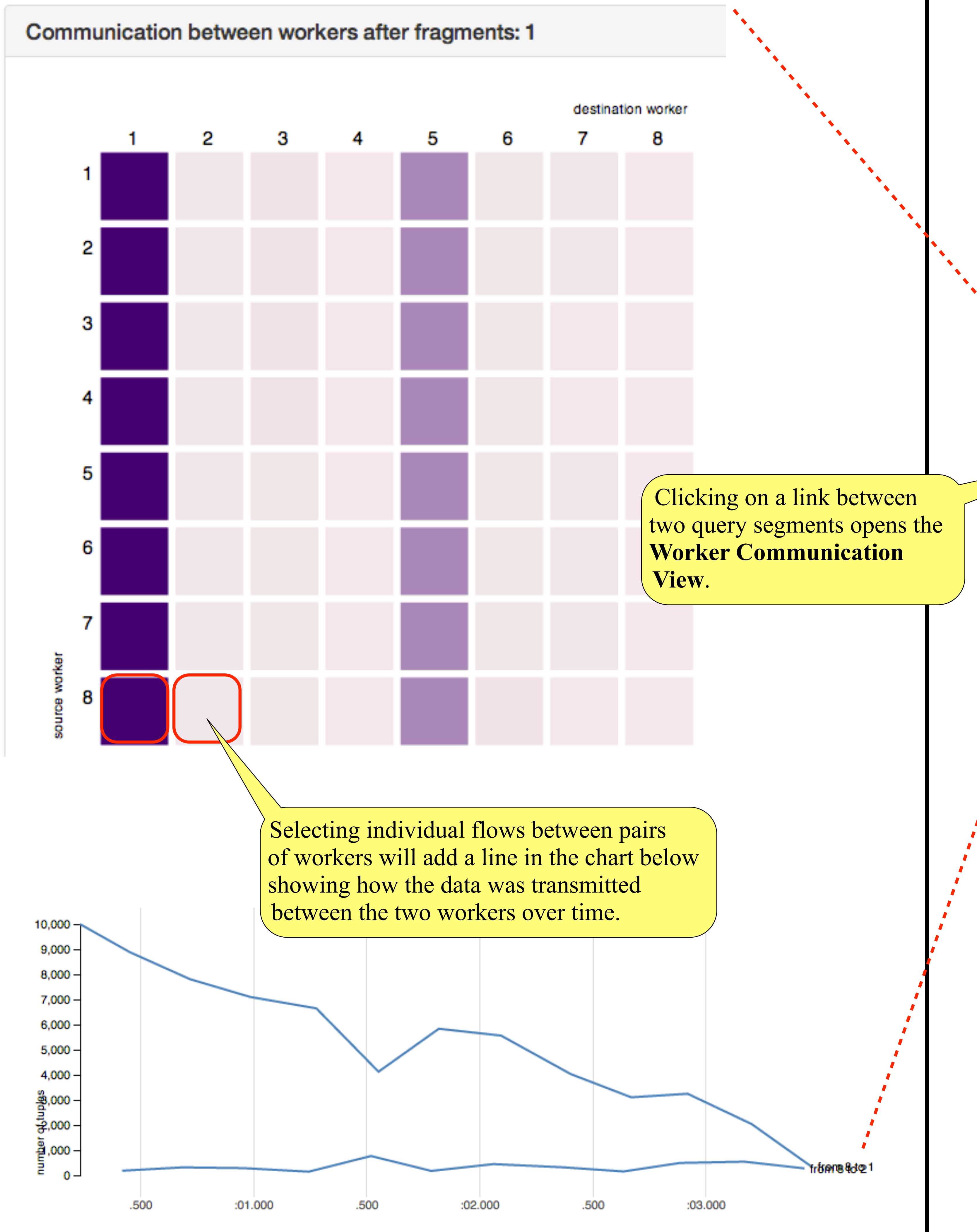
Our visualization tool helps developers and users gain **quick and effortless insight** into the execution of a query. Our tool is used by **MyriaDB,** an online Big Data Management System. [http://db.cs.washington.edu/myria/]

Goal: Help the developers and the programmers make **queries run faster** by addressing the following questions:

- Which worker/node/operator causes performance **bottlenecks?**
- How skewed is execution or data?
- How does **data flow** through the distributed system?

Check our tool here! https://myria-vis.appspot.com/

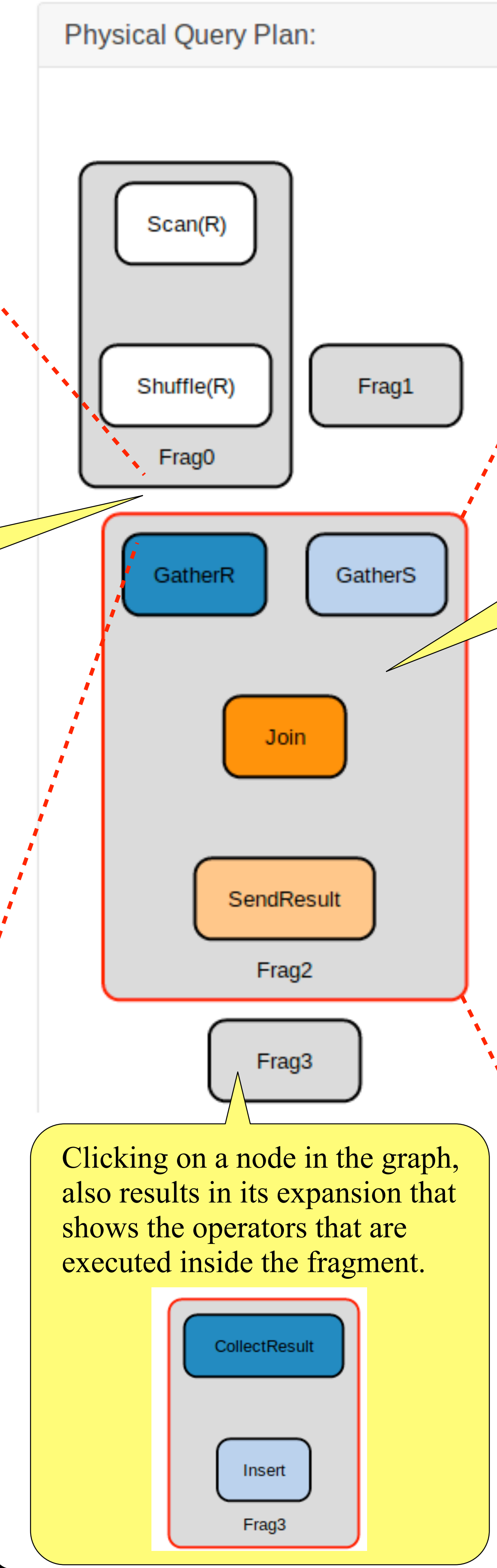
Worker Communication View: visualizes the amount of tuples sent between two query segments, per worker basis.



Clicking on a link between two query segments opens the **Worker Communication View.**

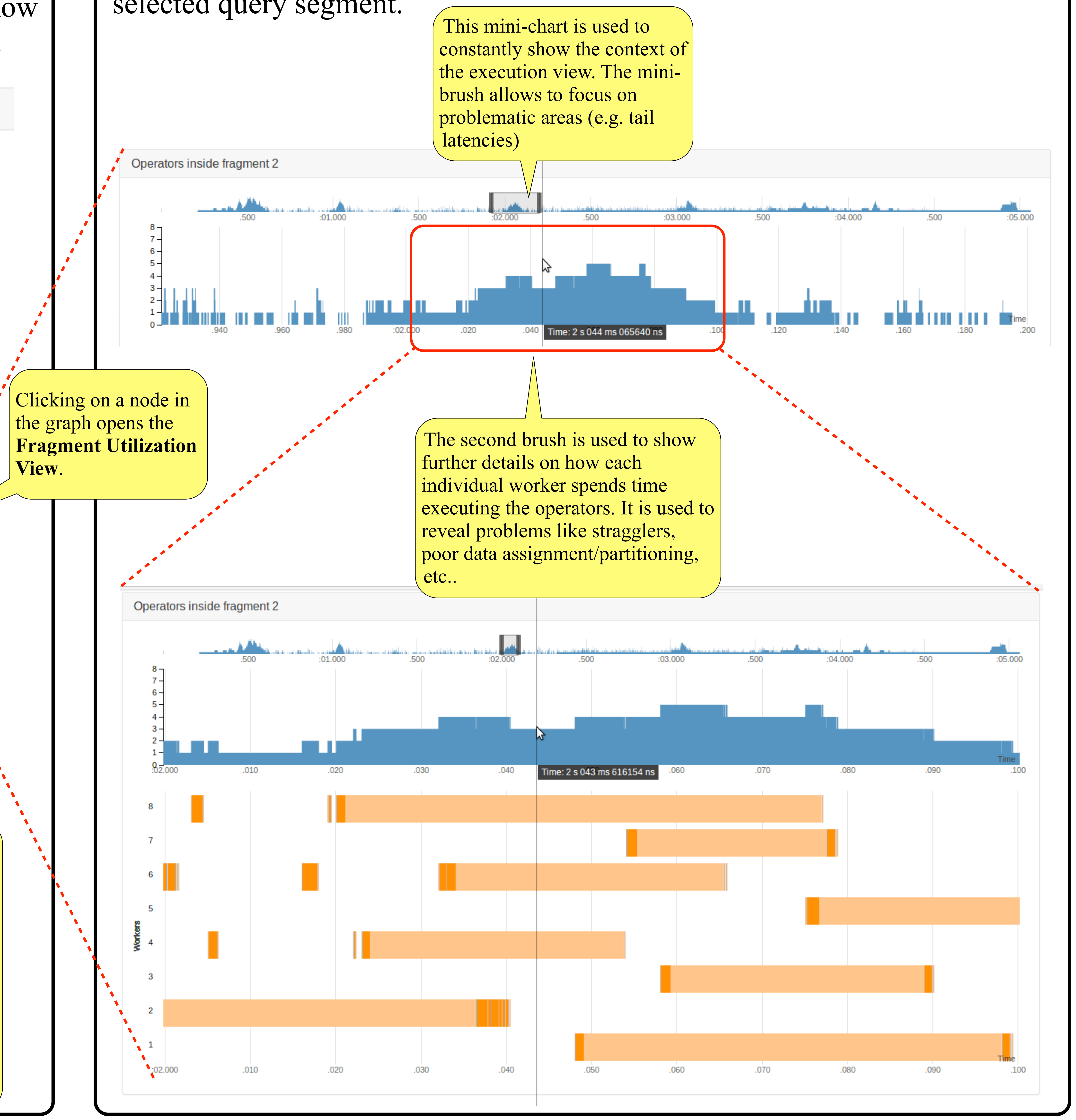
Selecting individual flows between pairs of workers will add a line in the chart below showing how the data was transmitted between the two workers over time.

Query Plan View: visualizes the query segments and how data flows between them.



Clicking on a node in the graph, also results in its expansion that shows the operators that are executed inside the fragment.

Segment Execution View: visualizes the utilization of the cluster for the selected query segment.



This mini-chart is used to constantly show the context of the execution view. The mini-brush allows to focus on problematic areas (e.g. tail latencies)

Clicking on a node in the graph opens the **Fragment Utilization View.**

The second brush is used to show further details on how each individual worker spends time executing the operators. It is used to reveal problems like stragglers, poor data assignment/partitioning, etc..