DOI:10.1145/2985784

To view the accompanying paper,
visit doi.acm.org/10.1145/2983551 rh

Technical Perspective Naiad

By Johannes Gehrke

THE NAIADS IN Greek mythology are the nymphs of fresh water. They are unpredictable and a bit scary, like big data, whose size has been exploding and continues to double every two years. Novel systems that process this data tsunami have been the focus of much research and development over the last decade. Many such big data processing systems are programmed through a workflow, where smaller programs with local state (nodes) are composed into bigger workflows through well-defined interfaces (edges). The resulting dataflows are then scaled to huge inputs through data parallelism (the execution of one node in the dataflow is scaled out across many servers), task parallelism (independent nodes in the dataflow are executed at the same time), and pipelining (a node later in the dataflow can already start its work based on partial output from its predecessors).

The most well-known class of such dataflow systems is based on the map-reduce pattern, enabling large-scale batch processing. These systems can process terabytes of data for preprocessing and cleaning, data transformation, model training and evaluation, and report generation, achieving high throughput while making the computation fault tolerant across hundreds of machines.

The second class of big data processing systems is stream-processing systems, which define dataflows that are optimized to react quickly to incoming data. They maintain state in their nodes for running aggregates or recent windows of data to watch out for patterns in the data stream; the occurrence of such patterns then triggers output for the next node. Stream processing systems are designed for low latency responses to new data while scaling with the arrival rate of records in the input streams. Example applications include monitoring data streams from the Internet of Things, high-performance trading, and maintenance of statistics for service analytics.

A third class of systems are graph processing systems optimized for dataflows with loops where it is important to efficiently handle state that is iteratively processed until a fixpoint for a computation is reached. Developers program graph algorithms by "thinking like a node" and writing the logic for a single node, and the platform then scales this to billions of nodes. Example applications include pagerank computation over large graphs and iterative solvers of linear systems such as Jacobi or Gaussian Belief Propagation.

The following paper describes the Naiad Dataflow System, which combines all these three classes in a single system, supporting high-throughput batch processing queries, low-latency data stream queries, and iterative programs in a single framework. The beauty of this work is that it does not create a Swiss Army Knife with different components for each capability, but that it unites them throughout the concept of *timely dataflow*, a simple, but expressive computational model that allows users to easily express all three of these concepts in the same platform. In timely dataflows, a record is associated with a structured timestamp that indicates where in the dataflow the record belongs. Thus, even though a huge number of records may flow concurrently through the system at any point in time with data and task parallelism and pipelining between nodes, it is easy to understand for a node which data it should process and when it should generate output. The result is a system that can have millisecond response times for low-latency queries, but also scales linearly for high-throughput applications.

In the stories about the Naiads, they are both the seduced and the seducers. May the following paper equally enchant you about the beauty of processing big data! C

Johannes Gehrke (johannes@acm.org) is a Distinguished Engineer in Office 365 at Microsoft, Bellevue, WA.