SUMMARY:
Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly.
Dryad: distributed data-parallel programs from sequential building blocks.
In *Proc. EuroSys Conference*, pages 59-72, 2007.

DATE: 22 February 2010

Dryad is an execution engine that facilitates the development of parallel and distributed applications by providing core services such as scheduling, concurrency and fault tolerance. An execution task is translated into a Dryad *job*, which is modelled by a *directed acyclic graph* (DAG). A job can be divided over smaller modular programs, represented by the vertices of the (job) DAG. Data channels connect the inputs and outputs of these programs and hence are represented by edges of the (job) DAG (connecting the vertices). Independent of the channel implementation, each channel produces and consumes heap objects utilized by the programs (vertices) in a well defined manner. The application code defines and controls serialization and deserialization of the heap objects. A *job manager* (process) constructs the communication graph from the application code, and then schedules the work across the available resources. A *name server* lists all available resources and their positions within the network, allowing for locality based scheduling decisions. *Daemon* processes (on worker machines) act as proxies for the job manager, accepting and executing vertex programs, and allowing for channel state monitoring (by the job manager).

The job manager plays a key role, and a job is terminated in the event of job manager failure. Concurrency is achieved via running multiple instances of a vertex. A failed vertex run can be re-tried (for a specific number of times) to provide fault tolerance. An *execution record* is created for a vertex execution when all the input channels become available. Each execution of the vertex is identified via a version number and a corresponding *execution record*. The output channel is uniquely defined for each run as well. When a job is successfully completed, each vertex identifies its successful execution run and the corresponding output channel.

A Dryad DAG is statically defined by identifying a sequence of vertices and a set of directed edges. Initially a graph is either created or cloned, and graph edges are added. Multiple graphs can also be merged together to produce a single (acyclic) graph. Runtime optimizations are possible by inserting layers of internal vertices.

SUMMARIZED BY: Atif Khan (a78khan)