are we there yet?

What is the state of a distributed application? Or even betterer, the state of a set of distributed applications? How can we measure the state, or progress, of a computation?

Per process

On a process by process basis, we can measure completeness by

- consumption of inputs (we're better off having done 60% of inputs than 40%). There may be several simultaneously active measures of inputs.
- consumption of constraints. Typically, these are system constraints like memory but limit the progress of a single process.

Per data

Sometimes, a system is constrained by access to data. The obvious cases are

- singular sources, such as a database or a set of source files
- live feeds, where the measure is whether the computation can keep up with the input rate of the data (or in some cases, catch up with the data flow)

Per communication leg

Often, the system is constrained by the inter-process links. This can appear in several ways:

- file creation/access speeds (for the typical case of dumping output into a file, and then the next step reads that file as input)
- network speeds (when the link is a TCP stream or somesuch)
- in extreme circumstances, memory speed (where the processes are threads or linked through shared memory or similar).

Other issues

The above is a natural consequence of considering a computation to be a graph of three sets of first-class things: processes, data and the communication links between them. Other issues include:

- overall system-wide constraints and sequencing. Sometimes we can make the system go faster by adjusting the sequence of processes.
- rework. What happens if some data is bad? How much of the system needs to be recomputed? This relies on both provenance and processing dependence.
- critical path analysis. Often, the overall system "speed" depends on a single processing pipeline; it would be nice to identify that pipeline, especially in an emergent dynamic system.

Common practise

Common practise is typically a combination of

- measurement of input files (or such like)
- measurement of overall system resources (in order to identify unexpected constraints)
- whisper tests (simple "are you processing" tests)
- heartbeat tests

Currently, *crux* supports heartbeat tests, and some processes (like *reeve*) support ping tests.

Operator alerts

Sometimes, the cluster needs to talk to an operator. For example, if there is some persistent problem in running cluster wide services, such as steward or healthcheck, How do we do that?

References

Doodle3 talks about overall system considerations.
Doodle38 expands on the notion of communication legs
Doodle42xx talks a little about alembic, which covers a bunch of this.
Doodle40 talks about how to think about a single application.