Implementation

Implementation

The engine (*tractor-engine*) is a high performance, multi-threaded C++ application. It responds to HTTP transaction requests, and maintains a database of spooled jobs.

The Blades are a straightforward execution server, written in Python, and providing site-extensible modules and role configuration.

The Dashboard provides job tracking, visualization, and control widgets developed on a modern HTML5/CSS3/JavaScript framework. Each user's Dashboard layout and preferences are maintained in a per-user section of the engine's data area.

New jobs are typically spooled into the queue as Alfred-format job scripts that are automatically processed by the engine and stored as job, task, and command records in a database (currently PostgreSQL). Jobs are usually spooled to the engine by running the supplied tractor-spool command-line script. There is also a supplied Python job authoring module that provides a scriptable API for creating jobs and submitting them.

Release Download Contents

The Tractor-2.0 download contains all of the system services, user interfaces, scripting tools, and other resources needed to deploy Tractor on your farm. However, each download is platform specific, so if your studio will be running Tractor components on several operating systems, then you will need to download one installer for each platform.

The Tractor system consists of these high-level parts:

- *tractor-engine* -- the main job queue manager executable
- *tv* -- the "tractor viewer" web-app implementation of the Tractor Dashboard
- *tractor-blade* -- the remote execution server that launches commands on each farm host
- *tractor-spool* -- sends job description files into the queue
- *tq* -- the job data query command-line interface and scripting module
- *rmanpy* -- a full Python 2.7.1 distribution, with extra built-in modules
- *job database* -- a PostgreSQL 9.3 database server distribution, located within the Tractor install area.
- *nimby* -- a small Qt-based user interface that helps control the use of workstations as auxilliary farm nodes

NOTE! Not every Tractor component is needed on every host, but all of the components are installed together below a single versioned Tractor directory because they share so many resources. The actual Tractor applications themselves are nearly the smallest part of the install package.

Many studios will choose to install once on shared disk, and run it on each participating host via network shares. Others will install the package on each host separately, but only run the pieces that are needed specifically.

Tractor-2.0 Bundle Details

The new Tractor 2.0 packaging and installation layout places a matched set engine, blade, spooler, user interaces, and scripting components all in one place, along with only one copy of shared resources including pre-built versions of several third-party subsystems. Tractor should be ready to go, with no additional external downloads, builds, or installs required.

The third-party components include a full Python 2.7.1 with built-in extension modules such as Qt and database support; a Tcl interpreter; and a full PostgreSQL 9.3 server to replace the prior job database. These components are provided as a convenience to our customers, and could be replaced on-site builds from source, should a studio wish to do so.

- A *PostgreSQL database* is now bundled with the release. It replaces the prior job database and is used to store all spooled jobs as well as all collected data about command launch and disposition. The database underlies several important 2.0 features, most notably the tq query variations. It also stores job data in a more compact footprint than before, and in some circumstances may permit a useful form of load-balancing. Operationally, the main tractor-engine process typically manages the database server as a subprocess and without the need for administrator assistance. That includes start-up and shutdown, table creation, and versioned schema updates. However, there are also advanced options for controlling specific aspects the database operation and administration. Only tractor-engine connects to the database, and so the default configuration only allows client connections from the localhost interface.
- *Python Update* -- many of the Tractor 2.0 tools, including the engine, now depend on access to a compliant Python interpreter. Tractor now takes advantage of the same *rmanpy* bundling scheme used by other Pixar RenderMan products. The version bundled into this Tractor release has been updated to Python 2.7.1, and includes several important built-in extension modules.
    - The updated version Python by itself includes several important features such as improved unicode support and bug fixes.
    - All of the python-based Tractor tools and applications are now shipped as *site-packages* modules in the bundled rmanpy Python distribution. The shared utility modules are also located in the same Python library area so they can be easily imported by rmanpy-based scripts.
    - The rmanpy package includes Qt support, used by Tractor in the reimplementation of the small stand-alone "nimby" desktop UI (thus also removing some old X11 and Tk platform requirements).
    - There is a PostgreSQL access module, used internally by Tractor to manage the backend data store. External scripted access to the database should always be mediated through the Engine's own exposed access protocols in order to protect developers and their pipelines from release-to-release changes to the internal database schema.
- *Job spooling support* --
    - Another extension includes a bundled Tcl interpreter used to parse Alfred-style job scripts. This extension is actually packaged as an rmanpy python module, allowing the Tractor python-based spooler and handler system to process Tcl-formatted alfscript jobs directly, without any additional requirements for separately installed Tcl packages, especially on platforms that do not (or no longer) have built-in

tclsh support. The Tcl extension also allows the *tractor-spool* script to sanity check new jobs directly on the user's spooling client system, before the job even leaves the their machine, flagging syntax errors and other problems before involving the engine or the job storage database.

- The Tractor *rmanpy* Python bundle and the PostgresSQL bundle described above are combined in Tractor-2.0 to improve the system's backend job spooling throughput. Parallel peristent intake channels with direct access to the job database can process new jobs and "expand" fragments at a significantly faster rate than prior releases.