Release Notes

- Changes in 2.4 2091325
- Changes in 2.4 2069290
- Changes in 2.3 1923604
- Changes in 2.2 1715407
- Changes in 2.2 1677499
- Changes in 2.2 1641321
- Changes in 2.2 1625934
- Changes in 2.2 1614082
- Changes in 2.2 1593580
- Changes in 2.2 1590950
- Changes in 2.2 1588479
- Changes in 2.2 1581500
- Changes in 2.2 1572682
- Changes in 2.2 1564321
- Changes in 2.2 1557505
- Changes in 2.2 1551997
- Changes in 2.1 1505289
- Changes in 2.1 1496411
- Changes in 2.0 1434105
- Changes in 2.0 1428659
- Changes in 2.0 1415017
- Changes in 2.0 1393388

Tractor 2.x Features

New features in Tractor 2 extend the core Tractor system established in the 1.x family of releases. There are a broad range of new additions and improvements, from productive new command line and scripting interfaces for wranglers, to simple user interface changes. Internal upgrades range from a new high-performance, high-capability job database, to new studio-wide resource sharing and allocation controls. Please refer to the guidelines described in Upgrading.

Here are some highlights:

- Tractor Product Layout -- *Single Release Directory, Single Download per platform, Bundled Subsystem Updates* -- The Tractor 2.x packaging and installation layout includes a matched set of Tractor components all in one download: engine, blade, spooler, user interfaces, and scripting tools. They are all installed together in one versioned area, along with only one copy of matched shared resources including pre-built versions of several third-party subsystems.
- Tractor Query Tools -- Introducing tq the *tractor query* command line tool and modules. Based on proven Pixar studio tools, tq is the best way to query live or historical Tractor data from your terminal shell, from your Python scripts, or from a new tab in the Dashboard.
- Adaptive Farm Allocations -- A way to dynamically allocate abstract resources between people or projects using Tractor's flexible *Limits* system. If two films are in production, 60% of the farm can be allocated to one of them, 30% to the other, leaving the remaining 10% for other projects. If one show is idle the others can temporarily expand their shares, then shrink back to the nominal levels when all projects are active.
- Dispatching Tiers -- A simple way to organize broad sets of jobs into a descending set of site-defined priority groups. The default tiers are named: **admin, rush, default, batch.** Create your own!
- Custom Menu Actions -- Add site-defined Dashboard menu items that can invoke your own centralized scripts, parameterized by the user's current list selection.
- Job Authoring API -- A new tractor.api.author module allows your Python scripts to easily create Job, Task, and Command objects linked together according to your dependency requirements. The Job.spool() method then sends the resulting job to the tractor-engine job queue.
- Simple Engine Discovery -- A simple "zero-config" announcement capability for small studios allows tractor-blades and other Tractor tools to find tractor-engine on the local network without requiring manual nameserver (DNS/LDAP) configuration changes. Tractor-engine will automatically disable this SSDP-style traffic at studios where the hostname alias "tractor-engine" has already been created by an administrator in the site nameserver database.
- Checkpoint-Resume Support -- Extensions to job scripting, dispatching, and the Dashboard add interesting new capabilities related to incremental computation. Tractor also supports a general "retry from last checkpoint" scheme. Both features integrate with the new RenderMan 19 RIS checkpoint and incremental rendering features.
- Blade Auto-Update -- A simple tractor-blade patch management system allows administrators to "pin" the farm to a particular blade patch version, and automatically push out a new version to the entire farm. Out of date blades restart themselves using the new module version.
- Pluggable Authentication Module (PAM) support -- The engine's optional new built-in PAM support delegates password validation directly to the operating system on the engine host. This alternative makes it simple to enable password support at studios where the LAN already provides adequate credential transport security.
- Privilege Insulation -- The *EngineOwner* setting in tractor.config specifies the login identity under which tractor-engine should operate. This setting is important because it allows the engine to drop root privileges immediately after it has acquired any protected network ports that it may need. The engine's normal day-after-day operations will then occur under the restrictions associated with the specified login name.
- Dynamic Service Key Advertisement -- Several blade profile **"Provides"** modes have been added to support some advanced service key use cases. For example, blades can dynamically advertise a different set of service key capabilities depending on which keys have already been "consumed" by previously launched commands.
- Resource Usage Tracking -- The operating system *rusage* process metrics CPU, RSS, and VSZ are now recorded into the job database for each launched command. Currently supported on Linux and OSX tractor-blades.
- Command Retry History -- A unique tracking record is now created in the job database for every command launch attempt. So the history of retries on a given task can be reviewed using the tq tool, for example.
- Configuration File Loading -- A streamlined override system can help to reduce clutter and improve clarity about which files have been modified from their original "factory settings" at your studio.
- Task Concurrency Throttle -- Each job can specify a "maxActive" attribute to constrain the number of concurrently running tasks from that job. This optional control over the This quick wrangling control over a job's "footprint" size on the farm can be useful when changing the full site-wide Limits settings is not appropriate.

- Automatic Blade Error Throttle -- This blade profile setting will prevent blades from picking up new work if they encounter too many errors within a given time interval.
- Job Spooling Improvements -- Job processing upgrades include faster processing, better error checking, and bundling of required subsystems. A parallelized job intake and database staging scheme can dramatically reduce backlogs when many jobs are spooled simultaneously, or when many "expand" tasks are running in parallel. A self-contained Tcl interpreter bundled with the spooler simplifies site install requirements and can perform client-side error checking prior to job delivery to the engine. A new JSON job spooing format is also supported (but not available prior to beta-1 pending changes).

Changes in 2.4 2091325

Bug fixes and improvements

- Fixed a regression in the prior 2.4 release that meant service keys might be reported incorrectly
- The launchd plist for running the macOS services had several outdated keys and caused errors to be reported and the service not to be restarted under certain conditions
- On Windows, AMD CPUs like the 3990x would report 64 instead of the expected 128 cores

Changes in 2.4 2069290

Tractor 2.4 is an update focused on overall performance, and handling large scale farm size, job size, and number of concurrently connected user sessions. Changes include:

- A significant code refactoring effort addressed several internal thread contention bottlenecks, including those related to frequent password checks and identity management. The new logic results in improved throughput, especially on very large farms (5000+ blades) where many dashboard sessions and automated scripts (1000+) are accessing job status.
- A more robust job-global system for sorting newly ready commands produced by "expand" tasks. This change addresses the "Cmd not Ready?" error problem - which was due to sorting key collisions (precision) on large recursively expanded jobs.
- Optimized limit checking operations, especially around repeated tokenization, trace message construction, and efficient handling of limits at their maximum capacity that temporarily throttle new dispatching.
- Addition of a new "connecting" entry in the queue/backlog diagnostic (the "status&qlen=1" query). This count gives distinct visibility to the number of connected but incomplete inbound requests, those still awaiting network i/o for their complete http request body contents.
- The number handler threads for running custom menu item backend scripts now scales up along with other thread pools based on tractor.config settings.
- Fix for a string handling issue that could result in intermittent loss of access to task log output.
- Fix for a tractor-engine crash during error message construction in response to unparseable (control) characters in "expand task" job graph extensions.
- Fixed Dashboard custom job filter matching on array types such as job Project names. Previously only exact matches were working for these entries, but not "starts with" or "contains" comparisons.
- Updated command-line options for the tractor-spool job submission utility. Several new formatting options such as zero padding are now supported on "range" values used to construct new jobs directly from command line parameters (as distinct from submitting previously constructed complete job files).
- Fixed a problem with "tractor-spool --ribs *.rib" wildcard expansion handling on Windows.
- Fixed mismatched version strings in some of the system service start up scripts.

Changes in 2.3 1923604

- Updated environment handlers and paths to accommodate batch processing of RenderMan 22 scenes from Maya and Katana.
- Addressed a Dashboard selection and copy issue related to a change of Firefox webkit css settings.
- Updated Dashboard links to the newest Tractor documentation.
- Address path separator issue on Windows in the Python Job Authoring module.
- Address a task state transition race condition in some "expand chunk" use cases.
- Fixed full job restart pruning of previously expanded tasks that were created by the "expand chunk" mechanism.
- Addressed potential security issues related to malicious interface use by on-site Tractor users.

Changes in 2.2 1715407

- RenderMan (prman) progress messages are now detected correctly by tractor-blade when Katana (renderboot) wraps them in additional logging text.

- Jobs submitted to tractor-engine from clients using the Python job authoring API method EngineClient.spool() now correctly abide by the current tractor.config setting AllowJobOwnerOverride. Note that this fix requires clients to re-import the new tractor.api.author module.
- Address a tractor.engine threading issue that could cause large “config” backlogs and slow dispatching in the unusual case where tq scripting clients make requests to the engine as usual, but site network routing issues prevented engine reply buffers from being delivered back to a few of those clients. Now those stalled deliveries time-out without affecting other transactions.
- Additional internal handling of the "assigner Cmd not Ready” state inconsistency condition that can arise in some cases of simultaneous task retry and job restart.
- Fix negative elapsed times that were sometimes displayed in the Dashboard as tasks finished, prior to a display refresh.

Changes in 2.2 1677499

- Custom menu items will cause a new window to be opened if there is any script output, even if the menu item is configured to normally suppress a new window. This enables problematic scripts to be more easily detected and debugged.

- Custom menu items support "login" as a special entry in the "values" list, which will cause the Dashboard user name to be a part of the payload sent to the menu item's script.
- Custom menu items will now observe user names and "@owner" in the crews attribute forjoband task custom menu items.
- Fixed bug in which selecting next task bystatein Dashboard task list was not automatically scrolling totask.
- Fixed the calculation of the elapsed time of a skipped task in the rollover of the Dashboard job graph.
- Sorting of tasks in the Dashboard task list has been corrected.
- Dashboard preview commands can now be displayed and run for archived jobs.
- The kill operation is now supported intqand the query API. It is used to kill a running command and leaves the task in an error state.
- tqhas a newreloadconfigcommand to enable the triggering of configuration file reloading from the command line.
- The query API will only return non-registered blades when the archive flag is set to True.
- tractor-dbctl--exec-sqlnow emits error messages.
- The blade caches user database entries in order to be more resilient against transient LDAP server outages.
- The systemd configuration directory defaults to the Tractor installation config/ directory, consistent with the sysvinit setting.
- systemd now starts tractor blade as root by default, consistent with the sysvinit setting.

Changes in 2.2 1641321

- Fixed the default MAYA_LOCATION on Mac OSX for renders using the recently introduced "rfm" environment key patterns like "rfm-RRR-maya-MMMM" (as distinct from similar "rms-" keys). The environment key causes tractor-blade to set up various enviroment variables before it launches each command. These can include things like MAYA_LOCATION and PATH. There are built-in key handlers as well as custom handlers defined by each studio. In this case, a key like "rfm-21.0-maya-2016.5" should have caused shared.macosx.envkeys to extend PATH with

  ```
  /Applications/Autodesk/maya2016.5/Maya.app/Contents/bin
  ```

- Fixed a blade status probe problem in some cases where a single blade was running a large number of commands concurrently.
- Fixed a tractor-blade issue that caused task launches to fail for jobs submitted by users with the '@' character in their login name. (Not their email address, but their unix login user name.)
- Adjusted the log severity level of the 'limit max set to zero?' diagnostic. It now only occurs at TRACE level as a debugging hint. It is acceptable for limits to have a maximum sometimes set to zero.

Changes in 2.2 1625934

- Added job service key expression support for blade selection based on "total physical RAM". For example the expression

  ```
  RemoteCmd {prman my.rib} -service "PixarRender && @.totalmem > 24"
  ```

  selects blades that provide the "PixarRender" service (blade.config) and which have at least 24 gigabytes of RAM installed. The previously supported "@.mem" key for "available free RAM" is also still available.
- Enabled access to BladeUse attributes in 'tq blades' queries: taskcount, slotsinuse, and owners.
- Added --user option to the logcleaner utility script so that a different user to query jobs can be used from the process owner which performs the file removal.
- Fixed the --add and --remove operations in the tq jattr and cattr commands for making relative changes to job and command attributes that are lists.
- Addressed a tractor-engine socket exception handling issue on Linux for cases where a tractor-blade host (operating system) has become unresponsive, such as in cases of GPU driver or OOM issues or a kernel panic. The tractor-engine process would sometimes exhibit high cpu load in these cases, spinning in the socket handler.
- Fixed the access-denied advisory text in JSON responses to retry, skip, and job interrupt URL requets.
- Suggested workaround for RHEL6 PAM-related file descriptor leak:

  ```
  On Linux RHEL 6.x era releases, the pam_fprintd.so module
  contains a bug causing it to leak file descriptors on every call from
  tractor-engine.  Since PAM modules are loaded into the tractor-engine
  process, and it performs many authentications over time, the unclosed
  "pipe" descriptors will accumulate, unknown to the main tractor-engine
  code and will eventually exhaust the available file descriptor limit
  for that engine process.  While many studios do not depend on
  fingerprint validation, especially for scripted API access to a system
  service, the "fprint" module is called indirectly from many common
  RHEL6 PAM policies, including "login" and "su".  It has been removed
  from the common policies in RHEL 7 era distributions.  A workaround
  for RHEL6 is to create your own "tractor" policy that doesn't include
  system-auth, or perhaps to specify a less general policy in crews.config,
  such as password-auth.
  ```

Changes in 2.2 1614082

- Added better user attributions in log messages related to job deletes and task task actions such as skip, kill, recall, and retry.
- Improved tractor-engine self recovery from unexpected command signature changes that could result in dispatching stalls in some jobs, and an "assigner Cmd not Ready?" warning in the logs.
- Fixed an engine problem that would sometimes update task records too early on active tasks that were in the midst of being swept from active blades during a manual retry of a predecessor tasks. The failed update could lead to an incorrect report of the number of active slots on the given blade.
- Fixed enforcement of job attribute edit policies such that dashboard edits cannot be applied to a job that is being moved to the archives (aka deleted).
- Fixed a tractor-blade state checkpoint problem that could sometimes cause state reporting delays when a blade was rebooted.
- Fixed an engine crash related to handling a deliberately malformed URL query.
- Added additional exception handler protection in tractor-blade to guard against errors in user-provided custom TractorSiteStatusFilter extensions.
- Fixed a problem handling very long limit tag names.

- Improved the dashboard efficiency related to automatic refreshes of job and blade lists, reducing load on the engine and database when many users are connected.
- Fixed an RPM specification issue that could result in RPM install error messages like "Transaction check error: file /opt from install ..."
- Fixed tractor init.d scripts to return a non-zero status code (3) when the "status" query is used to whether the service is running or not.
- Worked around an init.d built-in function issue that could sometimes result in a "dirname" error in blade service start up.
- Introduced optional new systemd start up scripts for tractor-engine and tractor-blade, for use on RHEL7 systems for example. The scripts are shipped in /opt/pixar/Tractor-2.2/lib/SystemServices/systemd/. See the documentation section Linux - with systemd

---

Changes in 2.2 1593580

- Fixed an issue where a task retry after job pause could allow that task and its successor to both become runnable concurrently after the job was unpaused.
- Fixed engine logic that could generate spurious "retry successor task loop detected" log messages in some complex tractor Instance use cases.
- The "rfm--*maya*-" environment block in the new stock shared.*.envkeys configuration files now includes additions that allow xgen procedurals to load correctly when batch rendering from Tractor. Copy the "shared.*.envkeys" files to your tractor config directory, or integrate similar changes into your "rfm" handler block if you have customized files.
- Changed the shipped example task custom menu item to avoid confusion with a similar example in documentation.

Changes in 2.2 1590950

- Fixed bug that prevented engine from communicating with blades, such as to perform NIMBY operations or probe the blade for its status.
- Jobs submitted to non-existent tiers are now prioritized as though they were submitted to the default tier.
- The Dashboard distinguishes between ready and blocked tasks when a job has been selected; real-time updates of task state transitions from blocked to ready are not yet supported.

Changes in 2.2 1588479

- Fix the "TR_EXIT_STATUS NN" handler in tractor-blade to restore corrected matching logic for negative exit code values. Negative values cause tractor-blade to treat the process exit as a termination due to a signal rather than as normal exit.
- Add a more aggressive connection timout for sockets originating inside the engine and connecting to blades that may be unreachable due to network outages or firewall blocks.

Changes in 2.2 1581500

- On the first start of the new engine, a database upgrade will be applied to fix the state of some expand tasks in still-running jobs. This fix prevents some out of order execution cases that could occur otherwise.
- Fix database consistency following engine auto-retries of tasks on unresponsive blades.
- Fix tractor-blade free disk space reports on terabyte+ OSX partitions.
- Improve Windows operating system version detection for some Windows 10 upgrade cases.
- Additional improvements to tractor-blade auto-update restart diagnostics and pidfile management.
- Fixes to avoid lengthy engine i/o thread stalls on unresponsive "full" client sockets.
- Fix an engine crash that could occur when a command's entire "argv" parameter string was deleted during a Dashboard task edit.
- Fix nimby state updates that could fail in some cases immediately following a blade.config reload on the engine.
- Fix a dispatchability problem affecting some tasks following a slot count edit.
- Fix host address partial match filters created in the Dashboard's filter editor.
- Fix emulation of legacy Iterate directives when the "-by" parameter is unspecified.
- Fixed dashboard graphical representation of archived jobs to ensure that jobs in the "trashcan" are visually distinct from those still on the active queue.
- Tractor-blade now adds TR_ENV_TRACTOR_ROOT to the environment of each launched command. It is the path to the top of the Tractor install tree for the running blade, for example '/opt/pixar/Tractor-2.2'

Changes in 2.2 1572682

- The new "Serial Elapsed" column in the dashboard job list displays the aggregate wall-clock runtimes for tasks in each job. This value will be longer than the real time that the job as a whole spends on the farm when tasks are running concurrently. For example, if a job has 10 tasks that run for 2 minutes, and all 10 tasks launch simultaneously on different blades, then the job will finish in 2 minutes and "Serial Elapsed" will display 20 minutes.
- The prior job "Elapsed" column in the dashboard has been renamed "Activity Span" to better characterize the value it displays. This is the span of wall-clock time between the first task dispatch from that job until the last task activity or user action on that job. It does not necessarily represent the time spent actively executing on the farm since there may have been scheduling gaps between tasks in a given job (other jobs with higher priority, etc), or user actions like attribute edits or retries might occur well after the last prior task activity.
- Renamed the "Elapsed", "App Secs", and "Sys Secs" fields in the dashboard command info view to "Real Elapsed", "Thread Elapsed", and "Sys Elapsed" to better describe the values. Values are expressed in both H:M:S and number of seconds.
- Fixed a dashboard problem that could cause the job elapsed time column (now Activity Span) to be stuck at 0:00 until the job was manually selected.
- Fixed the disappearance of the preview indicators when the job graph is scrolled.
- Fixed "tq blades" behavior to consistently list and operate on "registered" blades by default. Those are the blades that have registered with the engine more recently than their visibility was last "cleared" from the dashboard or tq. Use "tq blades up" to list blades that have sent a heartbeat recently. Some blades will register but then subsequently go offline; they will not have recent heartbeats but will remain registered so that admins can see that they have gone offline.
- Fixed the Python job authoring API to correctly handle the Command(local=true, ...) keyword when creating a "local" command rather than a more usual, default, RemoteCmd.
- Fixed several state update issues related task retries.
- Improved job elapsed time forecast estimates, especially following task retries.
- Fixed blade auto-update restarts on Windows related to variant install locations.

- Fixed the engine to send job project values to tractor-blade so that the variable TR_ENV_JOB_PROJECT will be correctly set in the environments of launched commands.

Changes in 2.2 1564321

Features

- GPU Exclusion Patterns can now be specified in the blade.config "ProfileDefaults" block. This setting is used to exclude certain GPU types from being used during service matching and counting consideration. Note that this keyword is only valid in the ProfileDefaults dictionary, and GPU filtering is performed **prior** to any per-profile match testing. Background: A given profile can match specific hosts based on several criteria in the "Hosts" clause, these can include the GPU type and count available on that host. Some hosts contain multiple GPUs including "uninteresting" virtual or underpowered GPUs that should always be excluded from consideration, *prior* to the profile matching pass. Use "GPUExclusionPatterns" to enumerate the makes/models of GPUs to be skipped in counts and matches. Each item in the list is a simple "glob-style" wildcard pattern, and patterns without '*' or '?' will be treated as if "*TEXT*" was given. See the new stock blade.config for an example:

```
"GPUExclusionPatterns": ["QX?", "paravirtual", "RV3*"]
```

- Added a new --allow parameter to "tq" to control custom nimby modes on selected blades. Usage is like tq nimby --allowMODE BLADE_SELECTION, for example:

```
tq nimby --allow santa,yoda name like rack42
```

  In that example, blades running on machines with hostnames containing "rack42" (like "rack42-01" or "rack42-g11") will only accept tasks from jobs owned by either santa or yoda.
- Added a new tq blade query modifier called "registered" to select blades that have re-registered with the engine since last having been "cleared" from the blade list display. Now, by default, only registered blades are shown in tq commands, matching the same blades that are visible in the dashboard. Older cleared blades (not re-registered) can still be listed with the tq -a/--archives option.
- Tractor-blade now adds its current path to command output logs in the case where the command exec itself fails due to executable not found.
- A new blade.config setting "SubstituteJobCWD" can mandate that a single working directory should be used for all command launches from all jobs, on blades using a given profile.
- Added a new administrative tractor-dbctl "--exec-sql" option for diagnostic and maintenance use; it uses db.config for connection details.

Fixes and Optimizations

- Fix performance issues related to dashboard update traffic following bulk edits, such as attribute changes on many tasks at once.
- Reduce dashboard refresh traffic to the engine in other common job update scenarios such as priority changes or pause/unpause.
- Fix tractor-blade SIGINT-to-SIGTERM escalation in some task interrupt situations. The previous behavior would sometimes leave processes running incorrectly on the blade if the top-level launched app did not respond to SIGINT by exiting. (For example, when a python script does not handle KeyboardInterrupt exceptions in a boilerplate try-except block.) Now tractor-blade, on Linux and OSX, will escalate to sending SIGTERM, and eventually SIGKILL, to the entire launched process tree for the given task if it does not exit promptly after receiving SIGINT.
- Improve speed of job database record insertion related to expand tasks.
- Fix out of order processing on expand tasks after 'retry errors' resets tasks within the expanded subtree.
- Fix engine updates to task "retry" state in the job database, causing various dashboard display inconsistencies.
- Fix handling of prior job elapsed time when a job is reloaded into the engine. The time estimation summary was sometimes zeroed, leading to incorrect values displayed in tq and the dashboard.
- Fix summary of task counts reported by the engine to the job database, for display in tq and the dashboard.
- Fix a tractor-blade race condition causing oversubscribed blades when several event queue actions occur between receiving a new command launch directive and actually launching the command.
- Fix dashboard task graph "instance" node rendering to apply task state change colors immediately and to avoid unnecessary decorations.
- Updated dashboard query pane to support display of recently added database fields.
- Update RMS/RFM environment handlers to include the bundled "rmantree" directory in the path form RMS, or the matching RPS tree for the upcoming RfM-21 release.
- Tractor-blade no longer resets the HOME and USER environment variables prior to command launch, in setuid mode, in situations where a site-defined tractor environment handler has already changed them to some new value.
- Fixed text formatting in the dashboard's "About" pane.
- Updated several built-in tq help descriptions.
- At smaller studios using the default python-based "cmdlogger" to collect command output logs from blades, idle blades will periodically close their open socket connections to the logging server and then reopen them again as needed. This behavior can avoid issues with stale socket handles, such as when some firewalls silently "kill" idle sockets.
- Update wrapper scripts for Linux and OSX to invoke /bin/bash explicitly rather than /bin/sh.

Changes in 2.2 1557505

Features

- The Python job authoring API now supports setting of job-level serialsubtasks and spoolcwd attribute.
- The "Clear earlier blade data" operation in the Dashboard's blade list context menu causes the selected blades to be hidden from view until they register again. This operation now also causes an internal cache to recalculate the active task count and number of slots in use for the selected blades. This is useful should sites observe invalid values in the blade list.
- Pressing 'Z' in the dashboard causes the view to scroll to the currently selected item.
- A new command line operation, tq queuestats, displays internal queue information from the engine. This is useful in debugging engine backlogs under unusual load.
- A new command line operation, tq dbreconnect, causes the engine to reestablish its database connections. This administrative operation may be useful in a several unusual situations. For example, dbreconnect can reclaim accumulated system memory consumed by a bug in PostgreSQL when new large jobs are submitted.

Fixes

- Fixed bug in which Dashboard would display incorrect task counts in job list.
- Fixed bug in which the stoptime and process metrics of a command invocation may not be updated if the engine was restarted while the command was running.
- Fixed bug in which a command invocation's current flag was not getting cleared if its task was retried while the command was running. This addresses multiple problems reported in the Dashboard, such as multiple blades reported for a command and unnecessary vertical spaces appearing in the job graph.
- Fixed bug in tractor-spool in which using the --engine option with the default engine, namely tractor-engine:80, was not being observed if the TRACTOR_ENGINE environment variable was set.
- Fixed a bug causing "linked.joblist" messages to appear in the engine log.
- Fixed blade list and blade activity views so that selecting a blade in one view will cause the selected blade to become visible in the other view.
- Fixed item lists so that when an out-of-viewport item is selected with the up or down arrow keys, the selected item will automatically be scrolled into view.
- Fixed broken client-side search box by adding a check for null values.

Optimizations

- Improved tq responsiveness through additional threads to handle query execution.
- Optimized task skip operation, reducing database I/O and message payload to Dashboard.

Changes in 2.2 1551997

Tractor 2.2 includes significant updates to several internal systems to improve user experience, correctness, and overall performance. There have been a variety of other features and updates as well.

A complete shutdown of the engine and job database will be required for the transition to 2.2. Simply starting the 2.2 engine will cause the old database to be upgraded to the new format automatically. Note that this is a one-way data migration, please backup your data area prior to restarting with 2.2 in order to retain 2.1 format data, should you need to revert to the older engine. Any tasks left running on the farm when the older engine is shutdown will continue to execute, and older tractor-blade processes will re-register with the new engine when it is started. New 2.2 tractor-blade servers can be started later, all at once or on a rolling basis; also the blade.config VersionPin mechanism can be used to automatically upgrade blades as they finish their current tasks.

Features

- The existing Dashboard provides different types of job data views through the "Job List" and "Query" tabs. Now in 2.2, double-clicking a job, task, command, or invocation in the query pane will cause the associated job and its graph to be displayed in the job list pane. The specific task will also be selected in the job list view when a task, command, or invocation is double-clicked in the query view.
- The task list view now contains a column showing the blade on which tasks ran.
- The task list can be filtered locally by blade name.
- The Dashboard "Blade List" tab contains search box to limit the displayed list of blades. In 2.2, several additional blade data fields are now searched for matches including GPU name, OS name, blade notes, and others.
- Several additional blade data fields, including GPU name, OS name, blade notes, and others, are now included in search box results on the Dashboard "Blade List" tab.
- Tractor-blade will add the job "project" name, if given, to the environment of each launched command, in the variable TR_ENV_JOB_PROJECT.
- Added built-in tractor-blade support for handling the changes to progress notification format in the upcoming PRMan 21 release.
- Blade profile definitions can now contain a list of acceptable GPU labels, allowing profiles based on GPU type to match several host configuration variants.
- Menu item entries menus.config now support an "enabled" option allowing administrators to remove custom menu items from view, without having to remove the entries from the file entirely.
- Output logs from job "postscript" commands are now accessible in the Dashboard.
- The new blade.config setting "TaskBidTuning" can be used to aggressively increase the rate at which very short running tasks are scheduled onto blades that have just finished a prior task or that still have free slots after an earlier assignment.
- A new metric representing the current job database commit backlog has been added to Dashboard's diagnostic graphs.
- Job attribute values can now be transmitted from the current dashboard task selection to custom task menu item handlers.
- Python job authoring API module updates: The Job class now accepts a "spoolcwd" keyword to set the working directory for the job. It also accepts a "serialsubtasks" keyword to serialize the top-level tasks in the job, if desired. The job.spool method now accepts hostname and port keywords to specify an alternate engine host and port destination for delivering the job, overriding the default "tractor-engine:80". Several module interdependencies were also removed, making the job authoring module more portable to other python installations.
- New options to the tractor-dbctl command line utility: - "tractor-dbctl --get-log" displays the entirety of the current day's postgresql log file. - "tractor-dbctl --log-filename" displays the path of the current day's postgresql log file. - "tractor-dbctl --log-day" [0-6|sun-sat] sets the day for postgresl log file options.
- New subcommand for the tq command line utility: - "tq blades up" will show all blades that are considered responsive. - "tq ping" tests connectivity to engine.

Fixes

- Improved job time-to-completion (TTC) estimates.
- Improved engine handling of Dashboard session-id cookies for auto-login, resulting in fewer reprompts for passwords for reloaded dashboard windows.
- Improved session id handling for multiple dashboards in different tabs of same browser by ensuring distinct session ids, even when re-login occurs.
- The tq command line utility now maintains distinct credentials sets for each --user parameter setting used by the same logged in user.
- Fixed a caching problem that could prevent commands from being considered for scheduling until the cache was periodically cleared.
- Fixed dashboard so that it doesn't force a re-scroll to the selected job or blade after a list refresh.
- Fixed dashboard to correctly allow existing saved filters to be overwritten.
- Fixed blade name hyperlink in the command info view to select the matching blade in the blade info view.
- Fixed docs link in dashboard.
- Added backwards compatility support for "/monitor?q=jobs&owner=*"
- In limits.config sharing definitions, the "nominal" and "reserved" values are now clamped to the cap, if given. Also a cap of 0 (zero) denies access to the given project.
- Fixed blade OS name reporting for some newer platforms.

Optimizations

- Improved dashboard interactivity with a 5x speed up of retrieving a selected job.
- Improved dashboard interactivity when a new job is selected after a large one has been selected.
- Improved blade's handling of large log output and long output lines.
- Removed redundant "ready task classification" logic from spooler and job restart, which can speed the latter operation up to 2x.
- Improved spooling speed by 2x through bulk SQL operations.
- Searching for jobs by user is more efficient with modification of underlying SQL query.
- Reduced limit sharing payload required for dashboard graphs.
- Improved limit data transactions handling to improve dashboard interactivity.
- Increased the postgresql.conf setting for checkpoint_segments to 64 to follow best practices for OLTP with PostgreSQL and avoid checkpoint segment warning in psql log files.

Changes in 2.1 1505289

- Improved engine and dashboard responsiveness when many users are fetching large job list results.
- Engine and job database performance improvements when loading and restarting jobs.
- Improved engine throughput and latency when delivering limits data to dashboards, either when there are many dashboards requesting the data, or when some of clients are connected through very slow networks.
- Improved tracking of still-running commands on engine restart.
- The dashboard job list quick type-in filter now also searches the job "projects" column for matches.
- Rule-based filters for the blade list can now match patterns in new GPU field. Also fixed a recently introduced problem matching the blade "Note" field.
- Improved the frequency and accuracy of updates to the job database fields "elapsedsecs" and "esttotalsecs" -- especially when the first tasks of a job are still running, but prior to the first task exit.
- Added new optional "metadata" fields to Tasks and Commands, created analogously to the existing Job metadata option.
- A new "metadata=0" modifier to the "/Tractor/monitor?q=jobs" query will remove potentially large metadata blocks from the returned list of job data. The metadata strings are not usually needed for display purposes. The dashboard now uses this option to improve responsiveness when building the basic job list display.
- Fixed simple job owner filtering in URL requests like "q=jobs&owner=somename", the owner restriction had been ignored in several recent releases.
- Add "LogLongRunningRequests" as a new tractor.config diagnostic option. The pair of values specify time thresholds (float secs) that trigger special logging from the engine when an inbound request has been waiting on a queue for too long (LAG), or when the actual processing of a request is took too long (RUN).
- Adjusted the tractor-engine --httpdebug option to emit simpler debug log entries for each inbound http request to the engine, analogous to entries in a web server access_log.
- Added an internal engine throttle on new task dispatching when the database backlog of pending unsaved prior job state changes becomes too large. Task dispatching resumes when these temporary backlogs have cleared. The recorded state changes are necessary for data durability, as well as dashboard consistency.
- Fixed graph cycle detection for jobs authored with incorrect Instance reference loops. Restarts of tasks within the loop could sometimes cause unresponsive engine handler threads leading to many time-out retry attempts from blades or dashboards. The engine side of these connections could enter a CLOSE_WAIT state, consuming process file descriptors.
- Adjusted the Tractor Python API modules for compatibility with both Python 2.6 and 2.7.
- Fixed a tq bug in which job operations failed when no jobs matched the search clause.
- Better diagnostics are reported database upgrade problems are detected.
- Fixed an engine crash related to unusual cases where the psql client library allocates large error message blocks temporarily.

Changes in 2.1 1496411

- Dashboard Job Notes -- A new Notes field has been added to the Dashboard job details pane, allowing text annotations to be added to any job. Notes are visible to other users, and the presence of a note is indicated with a small "chat bubble" icon in the job list. These notes can be used to describe a problem to wranglers, or to explain why a job needs, or is getting, special handling. The engine will automatically add a note to a job when an attribute is changed through some user action, such as altering priority, so the notes become a history of changes to the job.
- Dashboard Blade Notes -- A new Notes field has been added to the Dashboard blade details pane, allowing text annotations to be attached to a blade entry. These notes can be used by system administrators to describe known issues or to discuss ongoing admin work on a machine.
- Dashboard Job Pins -- Individual jobs in each user's job list can now be "pinned" to the top of the list, independent of the global list sorting mode. Jobs might be pinned because they are important to track or just because they represent a current "working set" of jobs. The group of pinned jobs float at the top of the list, and they are sorted according to the overall list sorting mode, within the pinned group.
- Dashboard Job Locks -- A single user can now "lock" a job from the Dashboard. A locked job can only be modified by the user who locked it. Locks are typically only used by wranglers who are investigating a problem and who want to prevent other users from changing, restarting, or deleting a job while the investigation is proceeding. The lock owner can unlock the job when done. Permission to apply a lock is controlled by the JobEditAccessPolicies "lock" attribute in crews.config.
- Task Logs 'L' Hotkey -- When navigating the tasks within a job, the logs for the currently selected task can be display by pressing the 'L' key. The key is a toggle, so pressing 'L' again will close the currently open log.
- User-centric Job Shuffle - Individual users can re-order their own jobs on the queue without disrupting global priority settings. The dashboard job list option "Shuffle Job To Top" essentially exchanges the "place in line" of the selected job with a job submitted earlier from the same user, causing the selected job to run sooner than it would in the default submission order. This swap does not affect the ordering of other jobs on the queue, relative to the submission slots already held by that user. This slightly unusual feature is a simplified re-implementation of the old per-user dispatching order controls in Alfred, as requested by several customers. Permission to perform this kind of reordering is controlled by the JobEditAccessPolicies "jshuffle" attribute in crews.config.
- The "project" affiliations for each job are now displayed in the job list view.
- "Delete Job" action is now called "Archive Job" -- The former "Delete Job" menu item has been changed to "Archive Job" to better reflect its actual function: when the db.config setting "DBArchiving" is enabled, jobs that are removed from the active queue are transfered to an archive database where they can still be inspected and searched in tq queries. If DBArchiving is False, then "deleted" jobs are actually deleted and their database entries are removed -- in this case the dashboard menu item still says "Delete Job".

- Archived Jobs View -- A Dashboard view of previously "deleted" (archived) jobs is now available. This view is analogous to a "trash can view" in some file browsers or e-mail clients. Jobs listed in the archive view can be browsed, and can also be restored to the main job queue where they can again be considered for dispatching. Note that jobs can sometimes contain "clean-up" commands that execute when they finish executing. These clean-ups may remove important temporary files that can make it impossible to re-execute that job.
- Task progress bars for Nuke renders -- Tractor-blade now triggers a Dashboard progress bar update when it encounters a multi-frame progress message from Nuke, of the form "Frame 42 (7 of 9)".
- Task Elapsed Time Bounds -- Job authors can now specify an acceptable elapsed time range for a given launched command. Commands whose elapsed time is outside the acceptable range will be marked as an error. Commands that run past the maximum time boundary will be killed. Example job script syntax:

```
RemoteCmd {sleep 15} -service PixarRender -minrunsecs 5 -maxrunsecs 20
```

- Per-Tier Scheduling -- A new extension to the DispatchTiers specification in tractor.config allows each defined tier to have its own scheduling mode. For example, the "rush" tier might be schedule in a strict FIFO order, whereas the default mode might be one of the modes that favors shared-access (like P+ATCL+RR). Tiers can be assigned the new "P+CHKPT" mode to take advantage of partial-graph looping feature in Tractor 2.0; and tiers using that mode should be placed before tiers receiving "classic" non-checkpoint jobs.
- Site-define Task Log Filters -- A new FilterSubprocessOutputLine() method is now available as an advanced customization feature in the TractorSiteStatusFilter module. This method provides python access to every line of task output. The site-written code can perform arbitrary actions in response to task output, and built-in Tractor-specific actions are also available. These include marking the task as an error, generating percent-done progress updates, initiating a task graph "expand" action, and stripping the output line from the logs.
- GPU Detection -- On start-up, tractor-blade now makes an attempt to enumerate any GPU devices installed on the blade host. The device model and vendor name "labels" are made available during the profile selection process so that groups of blades can be categorized by the presence or type of GPU, if desired. The "Hosts" dictionary in a blade.config profile definition defines the matching criteria for that profile. Two new optional keys are now available: the "MinNGPU" entry specifies minimum number of GPU devices required for a match; and "GPU.label" specifies a wildcard-style matching string for a particular vendor/model. This label string also now appears in the Dashboard blade list, if a GPU device is found.
- The new tractor.config setting "CmdAutoRetryStopCodes" specifies a list of exit codes that will be considered "terminal" -- automatic retries will NOT be considered for commands that exit with these codes, unless the -retryrc list for a specific command requests it. Negative numbers represent unix signal values, and the codes 10110 and 10111 are generated when a command's elapsed time falls outside the new run-time bounds options, when given. The default setting for the no-retry stop codes are the values for SIGTERM, SIGKILL, and the two time-bounds codes:

```
"CmdAutoRetryStopCodes": [-9, -15, 10110, 10111],
```

- Engine statistics query -- A new URL request (Tractor/monitor?q=statistics) has been added to help integrate tractor-engine performance metrics with other site-wide monitoring systems. The returned JSON object contains the most recent sample of several statics that the engine collects about itself. This data might be used, for example, to populate an external site monitoring system. Some monitoring systems are able to make this URL request for data directly, while others may require a small data source script to be written that requests the JSON statistics report and then forwards each value of interest to the monitoring system separately.
- Concurrent Expand Chunks -- This advanced expand task variant provides one approach to avoiding serial delays in jobs containing long-running single commands that produce a sequence of results needed by other tasks in the job. This new extension enables pipeline integrators to construct jobs that launch a long running command, such as a fluid simulation, and then concurrently launch another command, such as a render, when each sequential output file is generated by the first command. Thus rendering can proceed without waiting for all of the simulation steps to complete. This particular approach is well suited to cases where the simulation app is creating output files whose filenames are not known ahead of time, and thus the subsequent render command line arguments must be generated dynamically. The simulation, or a wrapper script, detects when the next step is complete, then it writes the appropriate rendering Task description into a temporary file, and then notifies tractor-blade by emitting the new 'TR_EXPAND_CHUNK "filename"n' line on stdout. Tractor-blade will detect that directive in the application stdout stream and deliver the file contents to the engine. The new render task is inserted into the running job and can be dispatched immediately elsewhere on the farm. The blade will automatically remove the temporary file once it has been delivered.
- TR_EXIT_STATUS auto-terminate policy change -- the default behavior for the TR_EXIT_STATUS handler has now reverted to the 1.x and earlier 2.x behavior in which the status value is simply recorded and then reported later when the command actually exits. The more recent behavior in which the blade actively kills the app upon receipt of TR_EXIT_STATUS is still available, but it must be explicitly enabled in blade.config using the profile setting:

```
"TR_EXIT_STATUS_terminate": 1,
```

- Blade record visibility flag -- The Dashboard blade list display is created from database records describing each tractor-blade instance that has connected to the engine in the past. These records are retained, even when a blade host is no longer deployed, in order to correlate previously executed commands with the machine they ran on. The dashboard blade list menu item "Clear prior blade data" no longer removes the actual database record for the given blade. Instead it simply sets a flag that hides the record from display in the dashboard. The record (and its new unique id field) are now retained for correlation with old task records. The blade data items can be completely removed manually if they are truly unneeded.
- Cookie-based Dashboard relogin -- A new policy allows auto-relogin to new Dashboard windows based on a saved session cookie, even when site passwords are enabled. The cookie contains only a session ID that is validated by the engine, it does not contain any password data itself. The older policy that denied auto-login when passwords are required can be restored by adding a "_nocookie" modifier to the crews.config SitePasswordValidator setting.
- Added a new tractor-dbctl --set-job-counter option that sets the initial job ID value in a new job database. Job IDs start 1 by default, so this ability to specify a different starting value can be helpful when starting from a fresh Tractor install in order to prevent overlaps between the job IDs from the new install and older jobs. Tractor upgrade installs that reuse the prior job database will continue to see job ID continuity.
- Several internal improvements have been made to the job database upgrade proceedure. Many code-related changes in new releases can now be applied without a significant database alteration, needing only an engine restart. Changes involving new database schema definitions are now applied with a system that better handles upgrades across multiple versions.
- Overall throughput optimizations -- Various performance improvements have been made in the this release, especially with regards to handling large numbers of simultaneous updates as many jobs complete or are deleted at the same time.

Changes in 2.0 1434105

- Fixed a "pending retry" issue that would sometimes cause tasks to remain in the pending state and also block ready tasks with nearly identical command signatures from being dispatched.
- Fixed several "nimby app" startup problems that could arise when local hostname lookups fail or are inconsistent.

- Addressed job list ordering and truncation issues for some cases where filter return large numbers of records.
- Add tractor-spool option "--alf-argv-subst" to address a special backward compatibility case involving Assign statements and RemoteCmd argument lists.
- Improved upgrade checks applied to an existing job database to ensure consistency with new releases.
- Addressed several Tractor Blade "VersionPin" restart issues for blades running as Windows Services, particularly for cases where the service logon username had been altered.
- Addressed Windows Tractor installer "VC10 runtime" dependencies.
- Fixed blade reporting of numeric OS version on Windows.
- Updated installer internals for Mac OS X.

Changes in 2.0 1428659

- Fixed service-key selection errors in some cases where "RemoteCmd -refersto" was relying on Task level -id and -service.
- Addressed a limit sharing issue regarding allocation counts attributed to a share that is later removed in a limits.config reload. These counts are now attributed to the "default" share for as intended, for the purposes of limit enforcement and display. Similarly, if new share names are added to a limit while previously attributed commands are still running, ensure that the decrement matches the original increment.
- Fix a renormalization issue that could occur when a limits.config definition included several shares whose nominal allocations summed to greater than 1.0.
- Fixed assignment criteria handling that would sometimes prevent ready tasks from being dispatched if they closely matched other tasks that were known to be waiting for limits.
- Fixed blade handling of the "RecentErrorThrottle" settings in blade.config. Earlier 2.0 releases were effectively ignoring attempts to modify these parameters.
- Added dirmap support to the job authoring python API.
- Fixed job submission confirmation output from tractor-spool to include an entry for each submitted job, when several are submitted at once.
- Add --task-title and --limit-tags command line options to tractor-spool.
- Fixed tractor-blade command logger set-up for some cases where TRACTOR_ENGINE was not resolved correctly in default configurations.
- Fixed the "Log" view menu item on the dashboard task list to only be selectable when the task is known to have emitted output.
- Fixed the operating system name reported by tractor-blade when running on Windows 8.x.
- Address job delete problems that could occur in some cases where an engine was restarted while long-running commands were still active, and blade hostnames were not resolvable by the nameserver used by the engine host.
- Reduced nimby app start-up dependency on a resolvable local hostname. Also addressed a problem setting nimby at all in a preview version of this fix.

Changes in 2.0 1415017

- Updated the Dashboard task-details paging controls for tasks with multiple commands.
- Fixed "frozen" job time-to-completion (TTC) estimates.
- Custom menu item handlers now receive job ID in the event details.
- Blade now adds TR_ENV_JID, TR_ENV_TID, TR_ENV_CID environment variables prior to the launch of each command.
- New tractor.config setting AllowJobOwnerOverride can be set to 0 (zero) to disallow unauthenticated job ownership attribution changes. Full support requires job submission to use the updated tractor-spool from this release.
- Improved tq session ID handling and password prompts.
- Fixed time column display in the dashboard data query pane.
- Improved job script "Instance" handling in expand output, and in "forward reference" cases.
- Fix order of operations problems for tasks with multiple commands and the first command is an expand.
- Task "-refersto" and "-id" values are now shown in the Dashboard task details.
- Fix "return to previous blade" via RemoteCmd -id and -refersto in cases where prior blade affiliation is reloaded from the job database, such as at engine restart.
- Fixed -refersto caching on task retry and other rebinding cases.
- Added Serial Elapsed as distinct from Real Elapsed display in task details.
- Blade "delist" and "eject" directives are now accessible from tq and the query API.
- Extended the engine's password challenge length to accommodate longer client passwords.
- Fix invalid default rss, vsz, systime in Windows command exit reports.
- Improved a job database start-up validation test.

Changes in 2.0 1393388

- Added Dashboard task graph visualization of Instance nodes.
- Add a "quick job syntax check" option: tractor-spool --parse-debug (job.file)
- Supplement the tractor-blade TR_EXIT_STATUS handler such that it will now actively kill running applications that emit TR_EXIT_STATUS directives if they do not exit on their own in a timely manner. This behavior can be useful for simple wrapper scripts that cannot implement the full process-group shutdown that tractor-blade already provides. The new behavior can be disabled in blade.config by adding "TR_EXIT_STATUS_terminate": 0,
- Fix storage of afterJids attribute edits on previously spooled jobs.
- Fix a dispatching problem that resulted in "no dispatchable tasks" in some cases after task retries and "afterJids" delays.
- Address unicode handling issues for non-ascii characters in RemoteCmd application parameter lists (aka comman-line argv).
- Fixed Dashboard display of elapsed time for still-active tasks to avoid issues caused by clock differences between the engine and user hosts.
- Fix job ready-task counts reported by the Dashboard and tq in some cases following retries.
- Removed engine start-up usage of a platform-dependent external python module, psycopg2.
- Updated the Tractor Query API to improve its python module conformity.
- Changed the way that tractor-blade reloads site-defined TractorSiteStatusFilter modules on profile refresh to improve predictability.
- Extended the engine's expand-node output handling to tolerate some older Alfred-compatible no-op constructs.
- Allow TractorSiteStatusFilter to set an advisory status message (aka "excuse") that is visible in the Dashboard blade list, even when the site-specific callback is allowing a request for work to proceed.