Software Frameworks for Fault Detection

High Availability Services and Watchdogs

Services, managers, and processes in the data management middleware will be monitored for activity and proper operation through a set of established practices and strategies. Important services, daemons, and threads will have associated `high availability services` and `watchdogs` to detect failures and start the system on a path to recovery.

High availability services will serve as external monitors of the state of critical services; they will detect errors of various types and provide the framework for failovers, thus enabling recovery from even catastrophic errors that terminate the service instances. The high availability services will replicate the state of the critical services across multiple auxiliary resources to allow for failover to a backup service or daemon. Hence, redundancy will be a key aspect of the use of high availability services. The monitoring service for a given system service will be configured with detailed knowledge of the target service so as to identify quickly the occurence of a failed state or degraded process. Upon detection of a problem, the monitoring service will take the system out of the standard operating state and place it into a class of recovery state with a prescribed strategy to take the system back to operation. One of several distinct recovery states may be selected, depending on detected conditions. For example, the service may repair the existing instance of the target daemon if this is feasible; if not, it may initiate a failover to a backup instance where the state is rolled back to a recent checkpoint. Examples of services that may be protected with failover mechanisms are pipeline managers, event messaging brokers, data access services, and databases.

Long-lived processes that execute application codes and perform computation in the pipeline framework will be monitored by watchdog daemons that check on their activity and state. The watchdog daemons will listen on messages from the running processes (for example, through subscription to appropriate event topics or channels), and will also employ heartbeat monitoring to periodically register if a process is still active or not. In addition to checking if the process is alive, the heartbeat monitor will examine the state of the outputs and properties of the process to assess whether it is performing adequately or has started operating at a level that is inordinately subpar. Processes that the watchdog daemon has determined 1) to have exited abnormally, 2) to have started operating slowly or inefficiently, 3) to be hanging, or 4) to be exhibiting runaway behavior that consumes exorbitant resources will be halted as needed and restarted by the watchdog. Repairs or transitioning of parallel communicators will be an important part of the recovery of the system when failed processes are removed and/or restarted. Communicators might proceed with a subset of running processes after failed threads have been pared away (a case of degraded service or capability), or they may be fully reconstituted with fresh processes added to replace failed ones. The decision on which of these paths to select will be made based on context, e.g., for the real time processing communicators will not be reconstituted if the time required will cause the system to miss a deadline and fall unacceptably behind, whereas large scale archive center reprocessing may operate in a more thorough manner and rebuild the communicators.

Checksum Validation within a Data Access Framework

Detection of errors in the transfer of image files across networks and into archive spaces and file systems will be accompished using checksum validation after each operation. The process of ensuring the integrity of the raw data will begin with the creation of a checksum at an early juncture, perhaps before the arrival of the image into the context of the LSST DM system proper. Redundant data will be generated at a very early stage as well.

A data access framework will manage the transfer of image files from site to site within the distributed LSST DM archive and guarantee reliable transfers through the use of the checksum validation. This verification will occur after long haul transfers between geographically disparate sites, and also as part of pipeline processing when data is staged to disk on computing platforms. As data arrives at each site, new checksums will be calculated and compared to the preexisting checksums for the data, with comparisons done at multiple levels to ensure correctness. To try to localize errors and optimize overall performance, the system will instruct the checksum algorithm to attempt the recovery of errors by calculating where the expected error might be, and if possible, perform the repair without the extra overhead of invoking or scheduling a retransmission of the data. If a repair is not successful, the framework will initiate a retransmission of the image associated with the checksum error.

Beyond the checks on individual transfers, data scrubbing of image file collections on devices in the final archive storage on a seasonal basis will be crucial to maintaining data integrity.