Fault Tolerance Interfaces

When discussing fault tolerance, it is necessary to define the dividing lines between aspects handled by the hardware infrastructure, the middleware, and the application code. This document presents a strawman proposal for where those lines should begin. Inevitably, negotiation will be needed over specific trade-offs in order to find the most cost-effective solution that meets the requirements.

Middleware/Infrastructure? Interface

This section attempts to define the roles of the middleware and infrastructure teams with regard to fault tolerance planning. The chosen model is for middleware to define an overall architecture for handling fault tolerance and develop hardware reliability requirements from that architecture; infrastructure then transforms those requirements into particular hardware designs and specifications.

Computation/Storage? Nodes

The hardware infrastructure should provide adequately reliable computation and storage nodes. Even architectures that presume only "commodity" levels of reliability still cannot provide the needed quality of service on top of nodes with high failure rates. The exact levels of reliability required will depend on the overall middleware fault tolerance architecture chosen. The amount of redundant hardware including spares will also depend on the architecture.

Middleware should provide an estimate for the overall level of node reliability needed; the infrastructure group should then determine more detailed specifications for components such as:

- Disk
- CPU
- Power supplies
- Network interfaces
- Memory

Infrastructure should also be in charge of specifying protocols such as insisting on choosing components from multiple production batches or even multiple vendors to prevent common-mode failures.

Network

Any distributed processing system is also highly dependent on its network infrastructure. The network must provide adequate bandwidth for both normal and failure recovery modes of operation.

Middleware will define reasonable requirements for network reliability, including the frequency, duration, and extent (node, rack, center; outage vs. partition) of acceptable network faults, for intra-cluster LANs, intra-center LANs (within the Base Camp data center, the Archive data center, or a Data Access Center), and inter-center WANs. Infrastructure will then be responsible for converting those requirements into specifications for such items as physically separate redundant WAN paths, switch provisioning and reliability, cabling and connector reliability, etc.

Power and Cooling

All components of the data management system will rely on the provision of adequate power and its partner cooling.

Middleware will again define reasonable requirements for the frequency, duration, and extent of acceptable outages. Infrastructure will need to provide suitable systems, perhaps including multiple power feeds and backup generators, to meet those requirements.

Sample Requirement

The middleware-provided requirement might be something like this, with numbers chosen arbitrarily:

- No more than 3% of the nodes in any cluster will be down (for any unscheduled reason including disk failure) at the same time, unless the entire center is down. Those 3% may include one or more entire racks.
- Certain cluster nodes will be designated critical; they will not all be down more than 0.1% of the time, unless the entire center is down.
- No center shall be down (unscheduled) for more than 1% of the time.
- Sufficient storage will be provided for two copies of all disk-based data. The probability of loss or corruption of both copies of any disk-based data megabyte must be less than 1 in 10**10 per year.**
- Network partitions in which nodes remain up but communications between them are inoperative must not occur more than 0.01% of the time. Shutting down partitioned nodes is acceptable if the above constraints are also met.

Middleware/Applications? Interface

This section attempts to define the roles of the middleware and application software teams with regard to fault tolerance planning. The proposal is for middleware to define the fundamental architecture for handling fault tolerance. This architecture in turn will specify the capabilities that the application layer can use and will control which faults middleware will handle and which must be dealt with at the upper layer.

Application Services

The middleware architecture will define the services provided to applications and thus the supportable cluster computation models, including the quantum of restartability and the inter-node communications mechanisms. These may in turn constrain the types of application algorithms that can be implemented easily (or at all).

Application Faults

Certain faults are best detected by the application code. These include permanent application faults (usually due to algorithmic conditions) and some types of resource faults. Middleware will define classes of faults that the application code may report. Each class of faults will typically have a different recovery strategy.

Sample Requirement

The list of capabilities provided by the middleware to the applications might look like this obviously ridiculous set:

- All stages must 1) operate on local data which has a well-defined partitioning, 2) perform commutative, associative operations on sets of data to produce a result, or 3) persist data to files or the database.
- The only non-persistence communications provided are to combine the results of the second type of operation across all nodes, ensuring that each piece of data is included only once, and to distribute a set of data to all nodes according to some partitioning scheme.
- Each stage may be re-run or restarted whenever necessary, perhaps on a different machine, so all operations must be idempotent.
- All faults reported by the application are treated as permanently fatal and cause unavailability of the system. In particular, the middleware makes no provisions for re-running a stage due to an application-reported fault, with either the same or a different Policy configuration.