Ray's Notes from the Fault Tolerance Workshop

Discussion of FaultToleranceInterfaces

Discussion of sched. vs unsched. outages:

- page mainly addresses unsched.
- scheduled addressed by
    - ♦ telescope stream not always on
    - ♦ rundundancy can allow scheduled maintanence of parts of system

Infrastructure
: develop requirements on hardware

Application Layer
: specify the services and mechanisms available to application layer

Isolation of systems

Failure Modes (FaultToleranceUseCases)

characteristics:

- repeatability

- location
- stateless vs. stateful
- scale (one node vs. systems)
- causes slow execution
- noisiness: silent, corrupting, or run-away
- correlated failures

What is the scope of this document?

How is the role of SDQA analysis in overall fault-tolerance? What is the impact on middleware?

> Ease of implementation at each level: you prefer to push problem solving as low down as possible where it is easiest. Higher levels you may have more information, but it's more complicated to address.

Requirements

Science requirements: section 3.5, DP and mgtmt reqs

OTT1
: Not mentioned: How often must we meet the design/minimum
Gregory: use these 2 to define a heuristic (e.g. 1 sigma within design spec, 3 sigma within minimum).

DMFR

Section 2:
OTT1 drives sizing model:
0% data loss (information associations part of this)
0.1% alert publication failure (1 minute or 2 minute goal?)
what does 98% availability mean? Section 4:
App Layer:
reprocessing scopes capacity
24 hour downtime needs clarification
detection lists update latency is still TDB; what is detection list?
metadata summary updated every 6 months (goal/rationale?)
most stringent DM-APP-DP-AL-2: alerts 60 s after 2nd image
DQ report available w/in 4 hours of end of obs. night.
performance report w/in 4 hours ": has AC info, bu AC processing stretched over day
What is the catch-up capacity? if there is a DQA detected problem, how long do we have available to correct it?

Human Error

treat like any other error
best practices for human-based control/configuration

know how to buy back extra capacity.

tapes

read back processes; all data or sampling

If we can meet deadline

- Redo entire visit
- redo since checkpoint
- full redundant option

If not,

- redo portion of visit

triply redundant processing of a few amplifiers?

every day re-run the same test pipeline on the production system, run to detect changes in result.

Goals for Wed.

Working List of common practices.

Common Practices:

- test runs prior to production runs
- backups/duplication of data to enable swap in response
- mechanisms for detecting failure (heartbeats, checksums, ...)
    - ♦ on-the-fly checksum checking during read-in
    - ♦ sampled redundant processing

        - ♦ independent and dependent heartbeats
    - rescheduling of failed processes, allow for automated re-configuration
    - redundant execution of processes, fail-over to back up processes (db)
    - prevention techniques:
        - ♦ choice of fault-tolerant hardware
    - limit need for roll-backs:
        - ♦ limit over-writes
        - ♦ separate read-only vs. over-write

recovering patterns

- doubling cpu, allowing one to fail
- check-pointing with SAN with fail-over
- drop portions, use spare capacity to redo & deliver late
- drop entire frame

ramifications:

- dropping portions: minimize the scale of correlated processing;

    full focal plane correlations means losing whole visits

- how long does it take to reconfigure system?

maintanence practices:

- operations manual
- allow time for development of maintanence scripts/documentation
- any hardware health maintenance
- documenting fault-tolerance patterns

DC3:

- Can we evaluate the feasability of check-pointing?