MTTR “>>” MTTF

Armando Fox, June 2002 ROC Retreat

Low MTTR Beats High MTTF

- Previous ROC gospel:
  - A = MTTF / (MTTF+MTTR)
  - 10x decrease MTTR just as good as 10x increase MTTF
- New ROC gospel?:
  - 10x decrease MTTR *better than* 10x increase MTTF
  - In fact, decreasing MTTR may even beat a *proportionally larger* increase in MTTF (ie *less* improvement in A)

Why Focus on MTTR?

1. Today’s MTTF’s cannot be directly verified by most customers. MTTR’s can, thus MTTR claims are verifiable.
   - “For better or worse, benchmarks shape a field”

2. For end-user-interactive services, lowering MTTR directly improves user experience of a specific outage, and directly reduces impact to operator ($$ and customer loyalty). Increasing MTTF does neither, as long as MTTF is greater than the length of one user session.

MTTF Can't Be Directly Verified

- Today's availabilities for data-center-based Internet sites: between 0.99 and 0.999 [Gray and others, 2001]
  - Recall A is defined as MTTF/(MTTF+MTTR)
  - A=0.99 to 0.999 implies MTTF is 100x to 1000x MTTR
  - Hardware: Today's disk MTTF's >100 years, but MTTR's for complex software ~ hours or tens of hours
  - Software: ~30-year MTTF, based on latent software bugs [Gray, HDCC01]
- Result: verifying MTTF requires observing many system-years of operation; beyond the reach of most customers

MTTF Can't Be Directly Verified (cont.)

- Vendor MTTF's don't capture environmental/operator errors
  - MS's 2001 Web properties outage was due to operator error
  - "Five nines" as advertised implies sites will be up for next 250yrs
  - Result: high MTTF can't guarantee a failure-free interval - only tells you the chance something will happen (under best circumstances)
  - But downtime cost is incurred by impact of specific outages - not by the likelihood of outages
- So what are the costs of outages?
  - (Direct) dollar cost in lost revenue during downtime?
  - (Indirect) temporary/permanent loss of customers?
  - (Indirect?) effect on company's credibility -> investor confidence

A Motivational Anecdote about Ebay

- Recent software-related outages: 4.5 hours in Apr02, 22 hours in Jun99, 7 hours in May99, 9 hours in Dec98
- Assume two 4-hour ("newsworthy") outages/year
  - A=(182*24 hours)/(182*24 + 4 hours) = **99.9%**
  - Dollar cost: Ebay policy for >2 hour outage, fees credited to all affected users (US$3-5M for Jun99)
  - Customer loyalty: after Jun99 outage, Yahoo Auctions reported statistically significant increase in users
  - Stock: Ebay's market cap dropped US$4B after Jun99 outage
- What about a 10-minute outage once per week?
  - A=(7*24 hours)/(7*24 + 1/6 hours) = **99.9% - the same**
  - Can we quantify "savings" over the previous scenario?

End-user Impact of MTTR

- Thresholds from HCI on user impatience (Miller, 1968)
  - Miller, 1968: >1sec “sluggish”, >10sec “distracted” (user moves on to another task)
  - 2001 Web user study: T_{ok}~5 sec “acceptable”, T_{stop} ~10 sec “excessively slow”
  - much more forgiving on both if incremental page views used
  - Note, the above thresholds appear to be technology-independent
- If S is steady-state latency of site response, then:
  - $MTTR \leq T_{ok} - S$: failure effectively masked (weak motivation to reduce MTTR further)
  - $T_{ok} - S \leq MTTR \leq T_{stop} - S$: user annoyed but unlikely to give up (individual judgment of users will prevail)
  - $MTTR \geq T_{stop} - S$: most users will likely give up, maybe click over to competitor

Outages: how long is too long?

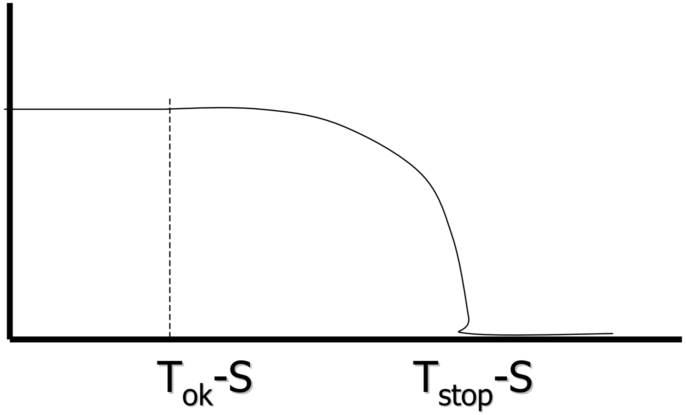
- Ebay user tasks = auction browsing and bidding
  - Number of auctions affected is proportional to duration of outage
  - Assuming auction end-times are approx. uniformly distributed
  - Assuming # of active auctions is correlated with # of active users, duration of a single outage is proportional to # affected users
- another (fictitious) example: failure of dynamic content generation for a news site. What is critical outage duration?
  - Fallback = serve cached (stale) content
  - $T_{headline}$: how quickly updates to “headline” news must be visible
  - T_{other}: same, for “second claass” news
  - Suggests different MTTR requirements for front-ends (T_{stop}), small content-gen for headline news ($T_{headline}$), larger content-gen for “old” news (T_{other})

MTTR as a utility function

- When an outage occurs during normal operation, what is "usefulness" to each affected end-user of application as a function of MTTR?

- We can consider 2 things:
  - Length of recovery time
  - Level of service available during recovery

- A generic utility curve for recovery time
  - Threshold points and shape of curved part may differ widely for different apps
  - Interactive vs. noninteractive may be a key distinction



Level of service during recovery

- Many “server farm” systems allow a subset of nodes to fail and redistribute work among remaining good nodes
  - Assume N nodes, k simultaneous failures, similar offered load
- Option 1 - k/N spare capacity on each node, or k standbys
  - no perceptible performance degradation, but cost of idle resources
- Option 2 - turn away k/N work using admission control
  - Will those users come back? What’s their “utility threshold” for suffering inconvenience? (eg Ebay example)
  - If cost of admission control is reflected in latency of requests that are served, must ensure $S+f(k/N) < T_{stop}$ (or admission control is for naught)

Level of service during recovery, cont.

- Option 3 - keep latency *and* throughput, degrade quality of service
  - E.g. harvest/yield - can trade data per query vs. number of queries
  - E.g. CNN.com front page - can adopt "above-the-fold" format to reduce amount of work per user (also "minimal" format)
  - E.g. dynamic content service - use caching and regenerate less content (more staleness)
- In all cases, can use technology-independent thresholds for length of the degraded service

Some questions that arise

- If users are accustomed to some steady-state latency…
  - *for how long* will they tolerate temporary degradation?
  - *how much* degradation?
  - Do they show a preference for increased latency vs. worse QOS vs. being turned away and incentivized to return?
- For a given app, which tradeoffs are proportionally better than others?
  - Ebay: can't afford to show "stale" auction prices
  - vs CNN: "above-the-fold" lead story may be better than all stories slowly

Motivation to focus on reducing MTTR

- Stateful components often have long recovery times
  - Database: minutes to hours
  - Oracle “fast recovery” trades frequency of checkpointing (hence steady-state throughput) for fast recovery
- What about building state from multiple redundant copies of stateless components?
  - Can we reduce recovery time by settling for probabilistic (bounded-lifetime) durability and probabilistic consistency (with detectable inconsistency)? (RAINS)
  - For what limited-lifetime state is this a good idea? “Shopping cart”? Session? User profile?

Summary

- MTTR can be directly measured, verified
- Costs of downtime often arise not from too low Availability (whatever that is…) but too high MTTR
- Technology-independent thresholds for user satisfaction can be used as a guideline for system response time and target for MTTR