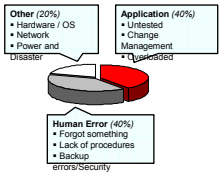
e-Business Threshold Management for Proactive Problem Determination

Anca Sailer and **Gautam Kar,** IBM T. J . Watson Research Center
Soila Pertet and **Priya Narasimhan,** Carnegie Mellon University

Motivation

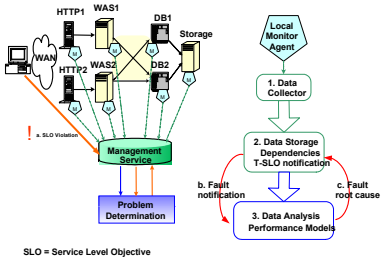


Causes of site failure
Source: Gartner Group - 2001

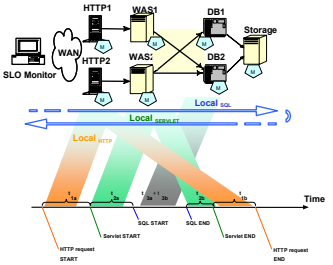
- Traditionally, problem determination is related to the state of components at the system level (e.g., CPU, memory)
- If problems manifest at the transaction level, the management service has no knowledge on the cause of the problem

Autonomic Problem Determination (IBM)

We build on work done by Sailer et al that pinpoints the root cause of performance degradation in multi-tier distributed systems



Monitoring architecture for e ommerce system



Determine time locally spent at component

Problem Determination Steps

1. Monitor transaction-related resource dependencies and resource behavior performance models
2. Use monitored data to decompose transaction-SLOs (T-SLOs) into component-SLOs (c-SLO)
3. Localize the root-cause of problem based on the degree to which components violate their constructed c-SLO

A. Decompose Transaction-SLO into Component-SLO

NO SLO VIOLATIONS

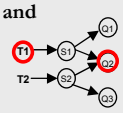
1. Read response time RT for component C
2. Lookup graph to create T, set {T1, T2} of transactions that depend on C

$$T_{Local}(C) = RT(C) - \sum_{k=1}^{M} \frac{\#Occurrences_k}{\#Occurrences_c} \times RT_k$$

$$P(C) = T_{Local}(C) / RT_{End\text{-}to\text{-}end}$$

$$c\text{-}SLO = T\_SLO * avgP(C)$$

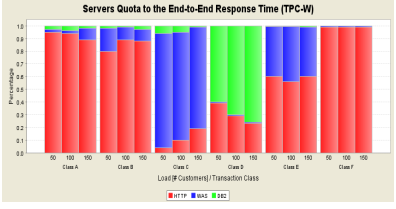
B. Detect c-SLO Violations and Localize Root Cause

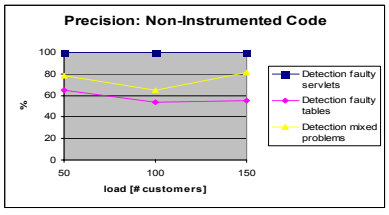


SLO VIOLATION
(i.e., if any transaction of T exceeds SLO)

1. If (TLocal > c-SLO) Then
   bad_avg(C) = αbad_avg(C) + (1-α)TLocal
   bad_N(C) = bad_N(C) + 1
2. If (bad_N(C) > β) Then
   Compute severity(C) = bad_avg(C)/ c-SLO
3. Sorts nodes by severity value
4. When problem is diagnosed
   Reset bad_N(C) = bad_avg(C) = 0

Experimental Observations





Published in the Ninth IFIP/IEEE International Symposium on Integrated Network Management 2005

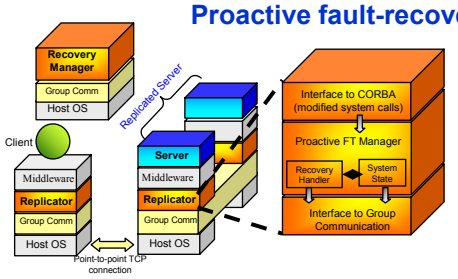
- The c-SLO does not need to be recomputed if the system operates within the linear part of the response time curve
- Precision of problem determination in instrumented code≈100%
- Precision of non-instrumented code ≈ 50% -100%

Proactive Fault-Tolerance (CMU)

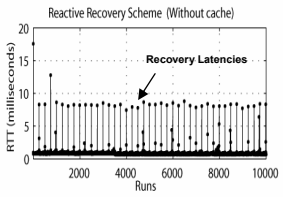
Determine if we can exploit performance monitoring to identify pre-fault symptoms and initiate proactive (rather than reactive) actions for faster fault-recovery

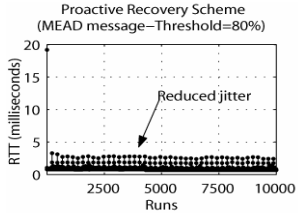
Pertet and Narasimhan developed a proactive fault-recovery strategy for distributed middleware systems in the presence of resource-exhaustion faults.

Proactive fault-recovery architecture



1. Transparently modify behavior of function calls associated with TCP middleware communication
2. Detect resource exhaustion and initiate proactive fault recovery
3. Migrate client connections away from failing server replica on receipt of proactive fault notification





Proactive fault-recovery significantly reduces recovery latencies in 2-tier middleware application [IEEE DSN 2004]

Idea: Increase the benefit of proactive fault-tolerance through topology awareness

Use knowledge of application dependencies to curb error propagation in multi-tier distributed systems

New Collaborative Approach

***Research Question**: What is the minimal level of system monitoring required to proactively diagnose performance problems in transaction-oriented distributed applications?*

- How do we detect SLO violations based on metrics other than response time? *(e.g., resource usage, throughput violations)*
- How much monitoring and instrumentation is required to effectively detect SLO violations of various kinds?
  Should we monitor everything or get partial data and extrapolate values through statistical inference or machine learning?
- What granularity of monitoring is most appropriate to detect SLO violations for different metrics?
- Does looking at more metrics provide higher accuracy in problem determination, and potentially more focused recovery?
- Can system monitoring help us identify patterns of abnormal behavior to enable proactive fault-tolerance?