Optimizing Failure Prediction to Maximize Availability

Igor Kaitovic and Miroslaw Malek
Advanced Learning and Research Institute (ALaRI)
Faculty of Informatics, Università della Svizzera italiana
Lugano, Switzerland

Presented by: Muhammad Salman Aslam (G-00763290)
Department of Computer Science George Mason University
Properties of Autonomic Systems

- Self Configuration
- Self Optimization
- Self Healing
- Self Protection

Requires Fault Tolerance
Fault Monitoring

Ref: Daniel A. Menasce “An Introduction to Autonomic Computing” Slide No 8, Department of Computer Science George Mason University
Types of Approaches for FT

- Reactive
  - Wait for the failure to manifest
  - Interrupt the process
  - Take corrective actions
  - Restore the System State
  - Periodic Check Pointing

- Proactive
  - Detect Signs of failure
  - Take early steps to avoid failure
  - Take corrective action to recover
  - Add to long term knowledge to revise fault repair procedures
  - Periodic and Predictive Check Pointing
Fault Monitoring / Prediction

- **Advantage**
  - Early detection
  - Prevention and Correction methodology implementation
  - Long term strategic course correction
  - Planned or auto repair options

- **Disadvantage**
  - Performance overhead
  - Wrong Detections (False Alarms)
    - Additional downtime
    - Interruption of the process
Run up to the Current study

- Periodic and Predictive Check pointing \[2\]
  - Improved efficiency by 30% with 6% overhead (then Periodic only)
- Health indicators (CPU Temperature, Fan Speed) Used to Migrate VMs \[3\]
  - Improved execution time by 30%
- Adaptive FT-Pro –using Reactive CP + Future Failure Probability \[4\]
  - Improved execution time by 2% to 43%
- Generic Framework for FT (Vallée et al) \[5\]
  - Policy Daemon, Failure predictor and FT Modules
- Failure Prediction + Migration Plan + Time Impact \[6\]
- Availability determinants studied (Salfner and Malek -- ) \[7\]
  - Used Precision, Recall, Prevention Prob, Repair time, Risk

**Current Study**

- Challenges Earlier Assumptions of CORRECT PREDICTIONs
- Optimizing Failure Prediction
Contributions

- Predictive Fault Tolerance in HPC Effects
- Study effect of Prediction Quality on availability
- Build a Model to Quantify the effect of Precision and Recall on Availability
  - Using Steady State Availability Equation
  - Introduces A-Measure
- Analysis of Availability Precision Recall trade off
  - Find the Optimal on the Adaptive precision recall graph
- Comparison of A-Measure with F-Measure and Recall
- Sensitivity Analysis against measures effecting availability
- Prediction Quality improvement Vs MTTR/MTTF improvement comparison
Different Approaches of FT

- **Reactive**
  - Wait for the failure to manifest
  - Detect Failure
  - Take corrective action and Clean up

- **Proactive (Predictive)**
  - Preventive
    - Detect Signs of failure
    - Take early steps to avoid failure
  - Repair Time Minimize
    - Predicative Check Point
    - Take corrective action if failure occurs
Model for Fault Tolerance (FT Model)

- Predictive Fault Tolerance
  - Failure Prediction
  - Corrective actions
  - Improved efficiency by 30% with 6% overhead (then Periodic only)

- Prediction
  - Lead Time - How Early
    - Goal = Lead Time < MTTR
  - Precision – How correctly predictions were made
  - Recall - Efficiency of prediction; how many of the failures predicted
Fault Tolerant Policies

- Policies
  - Failure Preventive FT Policy (Run time Migration)
  - Repair Time Minimization FT Policy (Predictive Check Pointing)

- New Parameters
  - Reward
    - Decrease of MTTR
  - Penalty
    - Overhead due to incorrect prediction

- Penalty = Overhead

- Reward (failure Prediction) = MTTR – Overhead
- Reward (Repair time minimization) = MTTR – (Overhead + MTTR(p))
Model (Quantitative setup)

- **Availability** = \( \frac{System\ Up\ Time}{System\ Life\ Time} = \frac{MTTF}{MTTF+MTTR} \)

- **True positive** - When a failure is predicted and it also occurs \( n_{tp} \)

- **False positive** - When a failure is predicted but none occurs \( n_{fp} \)

- **True negative** - When no failure is predicted and none occurs \( n_{tn} \)

- **False negative** - When no failure is predicted but it occurs \( n_{fn} \)

- **Total Failures** - True Positive + False Negative = \( n_{tp} + n_{fn} = n_f \)

- **Total Predictions** - True Positive + False Positive = \( n_{tp} + n_{fp} = n_a \)

- **Precision** \((P)\) = \( \frac{n_{tp}}{n_a} \)  \( \text{Recall} \ (R)\) = \( \frac{n_{tp}}{n_f} \)

- A measure - Measure for Availability \((\text{Coming Up})\)
Precision – Recall Trade off

Optimize for best tradeoff and implement procedures to achieve the threshold

Approximation $\sim P = 1 - R^3$
Model Equations

- MTTF Predicted
- Down Time
- Criterion for Reducing Downtime
- Availability (steady state)
  - Cond. P > 0 i.e. $n_{tp}$ non zero
  - No advantage if $P > 0$ or $R > 0$
- Availability Score
- Criterion For predictive
Insights

- Breakeven depends on Precision not Recall
- Can we improve availability?
  - Yes If we improve Precision
  - Ensure that we make good prediction
- How fast we can improve
  - We can improve rate if we improve recall
  - Ensure that we do not miss to predict any failure

- Let us optimize
System Parameters

- Referenced Parameters
  - Prevention
  - Model Parameters
    - Repair time minimize

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTTF</td>
<td>Mean-Time-To-Failure</td>
<td>30 h</td>
</tr>
<tr>
<td>MTTFD</td>
<td>Mean-Time-To-Failure-Detection</td>
<td>20 s</td>
</tr>
<tr>
<td>MML</td>
<td>Mean-Migration-Latency</td>
<td>50 s</td>
</tr>
<tr>
<td>MMO</td>
<td>Mean-Migration-Overhead</td>
<td>20 s</td>
</tr>
<tr>
<td>MRT\text{pep}</td>
<td>Mean-Recovery-Time (periodic checkpoint)</td>
<td>50 s</td>
</tr>
<tr>
<td>MRT\text{odep}</td>
<td>Mean-Recovery-Time (on-demand checkpoint)</td>
<td>20 s</td>
</tr>
<tr>
<td>MCPO</td>
<td>Mean-Checkpoint-Overhead</td>
<td>10 s</td>
</tr>
<tr>
<td>MTTR</td>
<td>$= MTTFD + MML + MRT\text{pep}$</td>
<td>120 s</td>
</tr>
<tr>
<td>MTTR\text{rim}</td>
<td>$= MTTFD + MML + MRT\text{odep}$</td>
<td>90 s</td>
</tr>
<tr>
<td>penalty\text{fp}</td>
<td>$= MMO$</td>
<td>20 s</td>
</tr>
<tr>
<td>reward\text{fp}</td>
<td>$= MTTR - MMO$</td>
<td>100 s</td>
</tr>
<tr>
<td>P\text{breakeven-fp}</td>
<td>see Equation (8)</td>
<td>0.1667</td>
</tr>
<tr>
<td>penalty\text{rim}</td>
<td>$= MCPO$</td>
<td>10 s</td>
</tr>
<tr>
<td>reward\text{rim}</td>
<td>$= MTTR - (MTTR\text{rim} + MCPO)$</td>
<td>20 s</td>
</tr>
<tr>
<td>P\text{breakeven-frm}</td>
<td>see Equation (8)</td>
<td>0.3333</td>
</tr>
</tbody>
</table>
Result for Failure Prevention Policy

- Availability drops sharply for lower precision.
- Break Even BE at intersection
- Best Case
  - Availability = 0.9998
  - Downtime Decrease by 83%
  - 8 hours improvement per year
- Bounding the contour by the Precision Recall tradeoff gives the line which allows to select optimal values
  - $P = 0.5598$, $R = 0.7607$
  - $A = 0.99948$, improves by 53.4%
Result for Runtime Minimize Policy

- Availability contours less aggressively above the breakeven than the FP policy
- Break Even BE at intersection
  - Break even at higher precision
- Availability under perfect conditions reaches 0.99908
- Downtime Decrease by 16.7%
  - 2 hours reduction over a year
- Bounding the contour by the Precision Recall tradeoff – Optimum values
  - $P = 0.7208$, $R = 0.6536$
  - $A = 0.9989$, improves by 8.83%
Result of Different Optimization Metrics

- Compared to earlier belief
- Recall is not as important as Precision
- Compare availability with both policies
- F-Measure lays stress on Recall and not on precision
- A-Measure weighs heavily on Precision
- A-Measure is better predictor
Result of Sensitivity Analysis

Sensitivity analysis is a technique used to determine how different values of an independent variable impact a particular dependent variable. [7]

+ve - increases function value  -ve - decreases the function value  0 – No effect

- {MTTF and MTTR improvements = Perdition Quality Improvement} -> Higher availability

- Penalty and Reward has a significant effect on availability
Some Things are Bothering

- Derivation of the optimum through analytical means not explained fully
- F-Measure calculation not clear
- Constraints of eq 5 and eq 8 not really utilized
- Single precision recall trade off utilized – It might vary under varying circumstances

- Explains the effect but addressing cause is important
  - HOW Do we do it ??
  - Is optimal Precision even achievable ??
  - Decisions on system health depend on VMM – Different VMM make huge difference, the bias needs to be taken care of.
Important Take away

- Fault Prediction is important
- Fault Tolerant Policy based in Prediction enhances availability
  - Preventive prediction better than Repair time minimize
    - Former is hard to manage and implement later is readily implementable
- We can quantify and optimize
  - Different Metrics can help us analyze and optimize
- Precision is More important than Recall
  - We need to accurately catch the faults - More False Positive better we are
  - Recall is still important
  - Precision is decisive if penalty is high enough
- Downtime can be minimized by improving prediction quality
References


7. F. Salfner, and M. Malek, “Proactive fault handling for system availability enhancement”, 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Denver, Colorado, USA, 2005


Q & A