Online Model-Based Adaptation for Optimizing Performance and Dependability

Kaustubh Joshi
William Sanders
Adnan Agbaria
University of Illinois at Urbana-Champaign

Matti Hiltunen
Rick Schlichting
AT&T Labs Research

ACM SIGSOFT

Workshop on Self-Managed Systems (WOSS’04)
31 Oct – 1 Nov 2004, Newport Beach, CA
Motivation

• Lots of work on adaptive architectures and mechanisms
• But, when and how should we use these mechanisms?
• We want to develop well-founded quantitative techniques to help decide how and when an adaptation mechanisms should be used

• Goals
  – Generate policies based on models of environmental and system behavior to govern adaptation mechanisms
  – Policies optimize important metrics that are defined on the system
  – Should not depend on specific detection or adaptation mechanisms
  – Develop tools to allow system designers to specify the models

• Non-Goals
  – Inventing new adaptation architectures or mechanisms
  – Learning unknown system behaviors
  – Inventing new failure or event detection mechanisms
Scope of Work

• Domains
  – Distributed Systems
  – Server farms
  – Scientific Computation Pipelines

• Metrics/Adaptations
  – Currently considering system level adaptations
  – Performance: response time, throughput
    • Adaptive resource management
  – Dependability: availability, MTTF
    • Selection and management of fault tolerance/recovery mechanisms

• Events
  – Faults/failures sensed through failure detector output
  – Workload changes, overloads (e.g. flash crowds)

• Assumptions
  – Stochastic behavior
  – Finite state systems
  – Finite number of adaptation actions (unlike linear feedback control)
  – Predictive control
Overview

- Offline solution yields initial policy table
- Online refinement of model parameters
- Periodic re-computation of the policy table
- Might not need to compute entire strategy table in some cases
Example: A Replicated Service

- Active replication is a technique to build highly available servers that maintain tight consistency amongst replicas.

- Number of replicas increase! Cost", Performance " (due to total ordering)
- But, too few replicas may lead to unavailability
- When a new node is added into the system, there is a period of unavailability while the application state is transferred to it
- To achieve the best availability, best performance, or maximum profit when should we add replicas to the system, and how many?
- Adaptation strategy in terms of feedback on number of servers currently up
## Elements of the System Model

<table>
<thead>
<tr>
<th>World States</th>
<th>Specify the possible states of the system and the environment variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>e.g., number of replicas currently operational</td>
</tr>
<tr>
<td>Adaptation Modes</td>
<td>Specify “soft” state of the controller. System behavior is typically</td>
</tr>
<tr>
<td></td>
<td>parameterized by the adaptation mode</td>
</tr>
<tr>
<td></td>
<td>e.g., number of replicas currently being added to the service</td>
</tr>
<tr>
<td>Events</td>
<td>Cause a change in the world state</td>
</tr>
<tr>
<td></td>
<td>Can be enabled only in specific world state, adaptation mode combinations</td>
</tr>
<tr>
<td></td>
<td>Multiple events can be enabled in each world state, adaptation mode</td>
</tr>
<tr>
<td></td>
<td>Can occur a random amount of time after being enabled – allow modeling of stochastic behavior</td>
</tr>
<tr>
<td></td>
<td>e.g., server failures, server additions</td>
</tr>
<tr>
<td>Adaptation Actions</td>
<td>Occur instantaneously</td>
</tr>
<tr>
<td></td>
<td>Change the adaptation mode (“soft state” of the controller)</td>
</tr>
<tr>
<td></td>
<td>e.g., number of replicas to add to the service</td>
</tr>
<tr>
<td>Rewards</td>
<td>The metric that is to be optimized</td>
</tr>
<tr>
<td></td>
<td>Defined for each world state, adaptation mode pair</td>
</tr>
<tr>
<td></td>
<td>Could be the solution of another model</td>
</tr>
<tr>
<td></td>
<td>e.g., response time of each state</td>
</tr>
</tbody>
</table>
Optimal Policies

- System evolves as:

  ![Diagram](image)

- A policy is a mapping of the World state $w,m$ to the Environment Controller, where the system evolves as $w',m'$ and $w'',m''$.

- Individual state rewards are aggregated over the lifetime of the system using an infinite or exponentially windowed summation:

  $$\sum_t R^t, R^tW_t; M_t \circ \sum_t R^t, -tR^tW_t; M_t \circ$$

- Optimal policies maximize the average aggregate starting from the state the system is currently in.

- System model can be converted into a Markov Decision Process and the corresponding sequential decision problem is solved (currently using standard techniques) to yield optimal policy.
Active Replication Model

- World State: # of servers, whether current view is consistent
- Adaptation state, actions: add n servers
- Events: server failure at rate $\lambda$, member removed from group with average removal time $1/\mu_r$, n members added with average time $1/\mu_a$
- Removal time is smaller than addition time (due to state transfer)
- Rewards
  - Maximize availability: 1 if state is 1, 2, or 3. 0 otherwise.
  - Minimize response time: solution of queuing model
  - Maximize profit: throughput*profit per customer - # servers*cost per server
Conclusions and Ongoing Work

- Use of stochastic models of a system and its environment to generate optimal policies for selecting adaptation actions
- Formulation of the adaptation problem as a Markov Decision Process
- Dealing with state-space explosion
  - State aggregation by binning
  - Model decomposition

- Currently applying the approach to a mobile messaging system developed at AT&T
  - Paper contains system and model description
  - Preliminary experiments indicate that models with state-spaces up to 1,000,000 states require a few seconds to solve

- Developing a modeling tool to allow designers to specify the models in a high-level languages such as Stochastic Petri-Nets
Extra Slides
Markov Decision Processes

- States, stochastic transitions, non-deterministic transitions
- Non-deterministic transitions: choices (adaptation actions) a controller can make
- Stochastic transitions: how the system and the environment evolve as a result of an adaptation action
- Reward is summed over entire lifetime
- Several algorithms (e.g. DP) exist to compute choice of actions (policy) that will maximize the reward

Markov Decision Process

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Policy

Markov Chain
Rewards

• Maximize availability regardless of performance: reward of 0 if all nodes have failed, or nodes are being added and 1 otherwise

• Minimize average response time over lifetime: response time computed using the queuing model in states where system is operational. In unavailable states, response time equals expected holding time in unavailable state + response time in subsequent state

• Minimize cost (where the cost of each replica and the cost of downtime is specified in $)

• Minimize average response time while maintaining a certain level of availability: constrained optimization that uses both the performance and availability rewards specified above
Adaptation Model Tool Chain

Model Solution using Möbius
- Model: States, Stochastic or Deterministic Actions, Measures
  - Möbius State-space generator
  - Markov Chain
  - Möbius Markov Chain Solver
  - Measure Value

Proposed Framework for Adaptation
- Model: States, Stochastic, Deterministic, and Non-deterministic (adaptation) actions, Measures
  - State-space generator
  - Markov Decision Process
  - Markov Decision Process Solver
  - Adaptation Policy: Optimal State Action mapping
Adaptation in a Mobile Messaging System based on iMobile

- Adaptation policy generation can be part of ADE Log/Monitor Component
Performance Adaptations

• Goal: maintain the average response time below a certain value in face of changing workload
• Response Time = time between message submission and delivery to the user
• However, response time cannot be directly measured. Models are needed to predict it in terms of the observables of the system

• We introduce adaptation actions that change acknowledgment processing modes
  – Complete – handle acknowledgments as usual
  – Delayed – gateways receive acknowledgments but do not send them to the servers (they are stored in the JMS)
  – Denied – acknowledgments are denied completely
  – Transfer – perform full processing of acknowledgments and send stored acknowledgments from the gateways to the servers
Performance Model

- Simple model structure, complexity in reward definition
- “Learn” the workload model. Ability to make decisions on anticipated workload changes (using forecasting)
- State: current ack processing mode, current workload, current acknowledgment load, outstanding acks?
- State space too large – aggregate by “binning”
- Events: any change in state
- Adaptation actions have no costs
- Cost: \([\text{Average response time} - \text{Target response time}]^+ + \text{Cost of delayed/rejected acknowledgments}\)
- Costs are computed by queuing models for each acknowledgment processing mode
Reward Queuing Models

- **Complex in full generality – several simplifying assumptions**
  - Convert system to flow balanced system
  - Assume gateways, servers, and DB on distinct hosts
  - Treat gateways of multiple types uniformly
  - Model using a BCMP Open Queuing Network

- **Full Processing**

\[ \lambda_{\text{alert}} = \lambda_r \times \text{Alerts/Request} \]

- **Delayed Processing**

Servers are M/M/n queues, where \( n = \# \text{number of hosts running the component} \times \text{level of parallelism in the server} \) obtained by benchmarking.
Infinite Horizon Decision Making

- Goal: generate policy that maximizes accumulated reward
- Well-studied problem (for Markov Decision Processes)
- Step 1: Use uniformization to convert into a discrete time model
- Step 2: use a stochastic dynamic programming recursion
  - Define a “value” for each world state and configuration mode that specifies the maximum average accumulated reward obtainable when starting from it
  - Value is then the solution of
    \[
    v(w; m) \quad \sum_{a \in A_w; m} \quad p_e(w; m^0) \quad \sum_{e \in E_w; m^0} \quad r(w; m; \hat{a}) \quad w_e(w; m^0) \quad m^0 \quad m^0;
    \]
  - Solution to the value function is obtained using a combination of linear system solving and dynamic programming updates
  - Choice of linear solution technique can have a substantial effect on the speed of convergence
  - The optimal policy chooses the maximizing action for each state
- Multiple reward functions can be handled using a linear programming formulation