Autonomic Self-Optimization According to Business Objectives

Natalia Razinkov natali@il.ibm.com, Sarel Aiber sarel@il.ibm.com, Dagan Gilat dagang@il.ibm.com, Ariel Landau ariel@il.ibm.com, Aviad Sela sela@il.ibm.com, Segev Wasserkrug segevw@il.ibm.com
Outline

- **Introduction**
- **Case study**
- **Business objective driven optimization**
- **Implementation technologies & methodologies**
- **Modeling & validation**
- **Conclusions, summary & future work**
Scenario: ROI - The IT view

The CIO:

- I have prepared a marvelous plan to increase our servers availability... it requires of course more servers, more disks, more people, more routers, more....
The CEO:

Nice work… but - where is the ROI? Do we need more computing power? Maybe we can get more revenues from using our current resources better? Remember – our business goal is to increase revenue, and not to own more MIPS. Look for a solution…
Problem Definition

- Current IT related optimization efforts focus on optimizing IT level metrics such as response times, availability, etc.

- What the business requires is that such IT optimization be carried out so as to optimize business objectives

- Such optimization is not a one-time effort as there may be significant changes, (e.g. server failures, sudden increase in the number of users) that may render any existing policy sub-optimal
ARAD Purpose

On demand optimization of IT and business process parameters and resources according to business objectives
Challenges:

- Modeling (how IT settings affect the business objectives)
- Initial optimization
- Continuous optimization (due to changes in hardware, software, mix & load of users)
- Solution requires self-optimization:
  - An automatic mechanism for carrying out IT optimization according to business objectives
  - An automatic mechanism for recognizing significant changes and re-optimizing
- A set of technologies and methodologies supporting the implementation of such a process
- Validation
Case Study – Site Architecture
Case Study – Business Objectives

- Business rules:
  - A commission is paid for each purchase/sale of stock
  - SLAs:
    - Flat fee is paid by customer for ensuring a certain service level
    - Penalties are paid to customers for SLA violations
  - Profit = commission + flat fee – penalties

- Customers have two important attributes (four types of customers)
  - Spending Amount – High/Medium
  - SLA type – Gold/Platinum

- Bandwidth must be allocated between the customers using the TQoS component of edge server so as to optimize profit
Case Study - AnyLogic 4.5 Simulation Model
Case Study - Optimization Results

- Improved income from -$475,376 to $179,088 !!
Implementing autonomic loop based on business objectives, business rules & SLAs drives IT policy optimization.
Business rules definitions

- General economic models
- Rule engines
- AMIT – a rule language & event correlation engine
- Examples of AMIT rules:
  - For each buying or selling of stock 4% commission is earned
  - For each service level agreement violation a 50$ penalty is paid
ARAD Modeling Concept

- **Business model**
  - Calculation of business metrics
    - Commissions, penalties, fees, etc.
    - Customers deserting due to poor service
    - Gaining & losing customers due to reputation
  - OBM – overall business metric – “end result”, objective function for optimizer

- **IT model**
  - System model
    - Hardware configuration (number of servers, number of CPUs, network configuration, etc);
    - Software (applications & their behavior, resources requirements,…)
  - User behavior model (mix, load, sessions, attributes such as spending amounts,…)

- **IT-to-business impact analysis model**
  - Poor response time impacts penalty & customer dissatisfaction indicator
ARAD Modeling Concept

Model creation

- **The IT architecture** is mapped using a set of building blocks which describe the set of standard components (hardware, server and middleware)
  - Queuing model
  - “Blackbox” building block
- **The customer specific** information required to simulate the customer environment is automatically derived (machine learning & statistical techniques)
  - User behavior model
  - User attribute model
  - Tier level message breakdown & resource requirements

Model updates

- In response to a significant change

Validation
While (true) //or time constraint, or optimization quality constraint
{
  1. Optimizer generates policy A
  2. Simulate model to calculate OBM for A
  3. If new OBM significantly better than OBM for existing policy, set real system policy to A
  4. Provide OBM result as input to optimizer
}
ARAD Optimization Architecture

- System Traffic Model
- System & Effector Simulation Model
- Current Actions Policy
- AMIT / ADI
- Economic value computation (AMIT situations)
- Optimizer
Real environment – Policy Example

```plaintext
policyAction PlatinumHigh
{
    PolicyScope  DataTraffic
    DiffServExcessTrafficTreatment drop
    #     DiffServInProfileRate          500
    DiffServInProfileTokenBucket  1000
}

policyAction GoldHigh
{
    PolicyScope  DataTraffic
    DiffServExcessTrafficTreatment drop
    DiffServInProfileRate          300
    DiffServInProfileTokenBucket  1000
}

policyAction PlatinumMedium
{
    PolicyScope  DataTraffic
    DiffServExcessTrafficTreatment drop
    #     DiffServInProfileRate          200
    DiffServInProfileTokenBucket  1000
}

policyAction GoldMedium
{
    PolicyScope  DataTraffic
    DiffServExcessTrafficTreatment drop
    DiffServInProfileRate          300
    DiffServInProfileTokenBucket  1000
}
```
IT Policy Enforcement

- An effector – component that affects the production environment, can manage its IT resources according to a specified policy
- Different settings of this policy must have a significant impact on business objectives and the OBM
The environment is constantly monitored measuring business objectives achieved so far
Existing monitoring techniques can be bound with AMIT rule engine to calculate business objectives
Significant Change Detection

1. A **significant change** is defined as an event that signals when significant change occurs (alert)
   - The failure of a server
2. A **significant change** is defined as a significant deviation of the monitored business objectives from the business objectives predicted by the model
   - Statistical tests

At the end of a period, for each business metric to be tested:

\[\begin{align*}
1. & \text{ Calculate the sample points for the business metric from the real environment (} n \text{ samples for each of the last } k \text{ periods)} \\
2. & \text{ Do the same for the simulation runs} \\
3. & \text{ Perform the } \chi^2 \text{ test for the real environments vs. the simulated runs} \\
4. & \text{ If the test returns } false, \text{ signal a significant change}
\end{align*}\]
Conclusions, Summary & Future work

- Given a valid model of the system, very significant improvements in business objectives can be obtained using this methodology, by setting the policy of currently available tools such as IBM’s Edge Server.
- In order to keep the site optimized according to business objectives, it is important to recognize significant changes and respond to them.
- A process, architecture, and a set of technologies for autonomic, on-demand optimization of an IT business infrastructure according to high-level business objectives, rather than IT level metrics are presented.
- This approach has benefits of constantly keeping the infrastructure aligned with business objectives, and results in a clear connection of IT related policy decisions to business level metrics such as profit or ROI.
- The architecture of the optimization component is very general, and may apply to various IT scenarios, such as e-Commerce sites and messaging infrastructure.
- Working towards automatic model validation, automatic deployment, other optimization algorithms, other types of models.
References


http://www.xjtek.com/anylogic

http://www.ibm.com/software/webservers/edgeserver/