A User-Centric Approach for Improving a Distributed Software System's Deployment Architecture

Sam Malek
2005 Annual Research Review

SoftArch
USC-CSE
**Deployment Architecture**

- **Motivation**
  - **Users**
  - **Services**

- **Deployment architecture**: distribution (i.e., assignment) of software components onto hardware nodes.

- **Service** is a user-level functionality provided via the collaboration of a subset of deployed software components.

- User uses a subset of services and has varying QoS preferences for each service.
Motivation

- Change the deployment architecture such that the users’ QoS requirements are satisfied (or maximized)

Approach
- Monitor the system parameters
- Input QoS preferences
- Estimate an improved deployment architecture
- Redeploy the system
Challenges in Developing a Solution

- Many QoS dimensions
- Many system parameters
- Fluctuations of system parameters
- Large space of possible deployment architectures
- Conflicting QoS dimensions

- Not feasible to have a “one size fits all” solution
- Need a framework that simplifies reuse of the common aspects of different problems
- Complexity of problem and fluctuation of system parameters require a self-managed autonomic solution
Model

- Representation of system’s deployment architecture
  - Components, hosts, physical links, logical links
- Quantification of system parameters
  - Frequency of interaction, bandwidth, etc
- Representation of system’s usage
  - Services, users
- Representation of concerns
  - QoS: latency, availability, security
  - User preferences (utility)
Algorithm

- Two types of objective functions
  - Optimization
    - Maximize availability, minimize latency
  - Constraint satisfaction
    - Avoid overloading devices or network links

- Two types of precision
  - Exact
    - Optimal result
    - Exponential complexity
  - Approximative
    - Suboptimal result
    - Polynomial complexity

- Centralized vs. decentralized
**Algorithm**

Exact – finds optimal solution $O(k^n)$

Biased/Unbiased stochastic – random selection $O(n^2)$

Avala – greedy approximation $O(n^3)$

DecAp – decentralized auction based $O(n^3)$

Clustering – decreases complexity

<table>
<thead>
<tr>
<th>Achieved availability</th>
<th>Time taken (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 comps 4 hosts</td>
<td>100 comps 10 hosts</td>
</tr>
<tr>
<td>200 comps 20 hosts</td>
<td>1000 comps 100 hosts</td>
</tr>
<tr>
<td>1000 comps 40 hosts</td>
<td>30 comps 7 hosts</td>
</tr>
<tr>
<td>300 comps 70 hosts</td>
<td></td>
</tr>
</tbody>
</table>
Analyzer

- Meta-level algorithm
  - Ensures satisfaction of the overall objective
    - Utility computing
  - Conflict resolution
    - Resolve conflicting results from several algorithms
  - Reflection
    - Adding/removing low-level components
  - Execution Profile
    - Fine-tune the framework
    - Considers fluctuations in the system
Monitor/Effector

- Implementation platform dependent vs. independent
  - Dependent: monitoring system parameters, redeployment of components
  - Independent: determining stability of monitored data, coordination of redeployment process
- PrismMW supports monitoring and redeployment at the architecture level
Monitoring/Redeployment – Prism-MW
User Input

- Parameters than can not be monitored
- Parameters that do not need to be monitored
  - Consistently stable
- Constraints on the system
  - Location constraints
  - Parameter constraints
User Input - DeSi
Centralized Instantiation

Slave Host
Framework
Slave Effector
Slave Monitor
Slave Host Implementation Platform

Master Host
Framework
Centralized Analyzer
Centralized Model
Centralized Algorithm
Centralized User Input
System Architect

Master Effector
Master Monitor
Master Host Implementation Platform
Decentralized Instantiation
Conclusion

- Quality of deployment architectures
- A framework for analyzing and improving distributed deployment architectures
- Tool support for the implementation of the framework
- Increasing potential for creating pluggable, extensible, reusable components that could be leveraged in different scenarios