Improving Availability of Distributed Event-Based Systems via Run-Time Monitoring and Analysis

Sam Malek
and
Marija Mikic-Rakic
Nels Beckman
Nenad Medvidovic

University of Southern California
Outline

- **Motivation**
- Problem description
- Prism-MW
- DeSi
- Algorithms
- Concluding remarks
Motivation

Deployment architecture (i.e., assignment) of software components onto hardware nodes.

How good is this deployment architecture?

What are its properties?

How should it be modified to ensure higher availability?
Outline

- Motivation
- Problem description
- Prism-MW
- DeSi
- Algorithms
- Concluding remarks
Problem description

Given system model parameters:

- **Software component properties**
  - Memory requirements
  - Frequency of interaction
  - Size of the exchanged data

- **Hardware host properties**
  - Memory capacity
  - Network reliability
  - Network bandwidth

- **Constraints**
  - Location
  - Co-location
Problem description

Find a function $f : C \rightarrow H$ such that the system’s overall availability $A$ defined as

$$A = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (freq(c_i, c_j) \ast rel(f(c_i), f(c_j)))}{\sum_{i=1}^{n} \sum_{j=1}^{n} freq(c_i, c_j)}$$

is maximized, and the deployment is valid.

Note that the possible number of different functions $f$ is $k^n$
Problem breakdown

1) **Lack of knowledge about runtime system model parameters**
   - System model parameters not known at the time of initial deployment
   - System model parameters change at runtime
   - **Middleware with monitoring support**

2) **Exponentially complex problem**
   - $n$ components and $k$ hosts = $k^n$ possible deployments
   - **Polynomial time approximating algorithms**

3) **Environment for assessing deployments**
   - Comparison of different solutions and algorithms
   - performance vs. complexity, sensitivity analysis, etc
   - **Analysis and visualization utilities**

4) **Effecting the selected solution**
   - Redeploying components
   - Requires an automated solution
   - **Middleware with deployment support**
Outline

- Motivation
- Problem description
  - Prism-MW
- DeSi
- Algorithms
- Concluding remarks
Prism middleware

- An architectural middleware
- Enables implementation and deployment of distributed systems in terms of their architectural elements
- Support for monitoring and redeployment
Monitoring and redeploying

Legend:
- Architecture
- Event frequency monitor
- Skeleton configuration
- Network reliability monitor
- Pointer to architecture
- Component
DeSi

- Deployment simulation environment
  - Specification and generation of deployment architectures
  - Visualization and analysis of distributed system
  - Estimation of the quality of deployment
  - Facilitation of rapid development and comparison of algorithms
DeSi’s architecture

DeSi Model
- SystemData
- AlgoResultData
- GraphViewData

DeSi View
- TableView
- GraphView

DeSi Controller
- Generator
- Modifier
- Algorithm Container
- Middleware Adapter
  - Monitor
  - Effector

Legend:
- Data flow
- Control flow

Middleware Platform
Suite of algorithms

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

![Bar chart showing achieved availability and time taken for different algorithms and numbers of components and hosts.](chart.png)

- **Achieved availability**
  - 10 comps 4 hosts
  - 100 comps 10 hosts
  - 200 comps 20 hosts
  - 1000 comps 100 hosts
  - 1000000 comps 100 hosts

- **Time taken (in ms)**
  - 40 hosts
  - 7 hosts
  - 300 hosts

- **Colors in the chart**:
  - Original Availability
  - Stochastic Algorithm
  - Avala Algorithm
  - DecAp Algorithm
  - Exact Algorithm
Integration

1) Monitor

2) Monitoring data

3) Analyze

4) Redeployment data

PrismMW

DeSi
Conclusion and future work

- Quality of deployment architectures
- Techniques/tools for improving availability

On-going/future work:
- Modeling other system properties
- Integrating DeSi with other platforms
- Decentralization and trust
Questions?
Approach - overview

Enabling the system to:

- Monitor its operation
- Estimate its new deployment architecture
- Effect the estimated architecture

\[ \text{Availability} \]

\[ \text{Time} \]

\[ \text{T}_M, \text{T}_E, \text{T}_R \]

\[ \text{T}_O \]

\[ \text{T}'_M, \text{T}'_E, \text{T}'_R \]

\[ \text{T}'_O \]
Automatic algorithm selection

\[ T_E \cdot A_C + (T_{AVG} - T_E) \cdot A_{EXP} \]
Using Prism-MW

```java
class DemoArch {
    static public void main(String argv[]) {
        Architecture arch = new Architecture("DEMO ");
        // create components
        ComponentA a = new ComponentA("A");
        ComponentB b = new ComponentB("B");
        ComponentD d = new ComponentD("D");
        // create connectors
        Connector conn = new Connector("Conn");
        // add components and connectors
        arch.addComponent(a);
        arch.addComponent(b);
        arch.addComponent(d);
        arch.addConnector(conn);
        // establish the interconnections
        arch.weld(a, conn);
        arch.weld(b, conn);
        arch.weld(conn, d);
    }
}
```
Component D sends an event

```
Event e = new Event("Event_D");
e.addParameter("param_1", p1);
send(e);
```

Component B handles the event and sends a response

```
public void handle(Event e) {
    if (e.equals("Event_D")) {
        ...
        Event e1 = new Event("Response_to_D");
e1.addParameter("response", resp);
send(e1);
    }
}
```