Automatic Configuration of an Autonomic Controller - An Experimental Study with Zero-Configuration Policies

Thomas Heinis¹ and Cesare Pautasso²
heinist@inf.ethz.ch, cesare.pautasso@unisi.ch
¹Systems Group, Department of Computer Science, ETH Zurich
²Faculty of Informatics, University of Lugano

Motivation

- Autonomic controllers are added to systems to enable self-configuration
- Autonomic behavior often requires configuration
- Configuring an autonomic system is very difficult and requires expertise
- Our initial experiments show performance variation of up to 287%
Distributed Workflow Execution

![Diagram showing Distributed Workflow Execution]

**Autonomic Configuration**

- Autonomic Controller monitors performance and adjusts configuration:
  - Monitors performance
  - Calculates new configuration
  - Applies changes to configuration
  - Waits for changes to take effect

- Acts upon:
  - Selection policy: Which nodes are reconfigured?
  - Information policy: What parameters should be monitored?
  - Optimization policy: How should the system be reconfigured?
Configuration Problem

- Best-known (Growth policy) policy reconfigures system once growth in either queue exceeds configured threshold

Standard PID Controller

- System is balanced if:
  - \( Q_{\text{Process}} + Q_{\text{Event}} = Q_{\text{Task}} \)
  - Control Error = \( \frac{Q_{\text{Process}} + Q_{\text{Event}}}{Q_{\text{Task}}} \)

- Control Actions:
  - If \( Q_{\text{Process}} + Q_{\text{Event}} < Q_{\text{Task}} \) => more dispatchers need to be added and vice versa
  - Control error \([-\infty, \infty]\) is mapped to the number of required dispatchers \([0, a]\), with \(a\) the size of the cluster

- Still requires tuning of parameters
Balancing Zero-Configuration Policy

- Balance producers and consumers given the growth of the queues
- Formally express growth in each of the queues:
  \[
  \text{Growth Q}_{\text{event}} = \#\text{Dsps} \times \#\text{Msgs} \times \text{Production Rate} - \#\text{Navs} \times 1 \times \text{ Consumption Rate}
  \]
  \[
  \text{Growth Q}_{\text{task}} = \#\text{Navs} \times \#\text{Msgs} \times \text{Production Rate} - \#\text{Dsps} \times 1 \times \text{Consumption Rate}
  \]
  \[
  \#\text{Dsps} = \text{Size of Cluster} - \#\text{Navs}
  \]
- Production and consumption rates are measured at runtime
- \#Messages is determined analytically

Evaluation

- Evaluation of the policies for 3 different workloads:
  - Busy workload:
    - 500 WF executions, 10 parallel tasks, 10s each
  - Burst workload:
    - Sequential 1s, Parallel 0s, Sequential 1s, Parallel 0s with:
      - Sequential 1s: 500 WF executions, 10 sequential tasks, 1s each
      - Parallel 0s: 2000 WF executions, 10 parallel tasks, 0s each
  - fMRI workload:
    - Medical workflow used for the post processing of Functional Magnetic Resonance Imaging data
      - 10 fMRI workflows started 10s after each other
Busy Workload

PID Controller Policy

Balancing Policy

Burst Workload

PID Controller Policy

Balancing Policy
### fMRI Workload

#### PID Controller Policy vs. Balancing Policy

<table>
<thead>
<tr>
<th>Queued Events</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Queue</td>
<td>0</td>
</tr>
<tr>
<td>Task Queue</td>
<td>50</td>
</tr>
<tr>
<td>Process Queue</td>
<td>100</td>
</tr>
</tbody>
</table>

#### Comparison

- **Execution time per policy and workload**

<table>
<thead>
<tr>
<th>Workload</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buey Workload</td>
<td>2000</td>
</tr>
<tr>
<td>Burst Workload</td>
<td>1000</td>
</tr>
<tr>
<td>fMRI Workload</td>
<td>900</td>
</tr>
</tbody>
</table>
Conclusions

- Performance of an autonomic system is very sensitive to its configuration
- Difficult to set configuration parameters right
- We have experimentally studied two zero-configuration policies:
  - PID controller policy
  - Balancing policy
- Both policies provide a performance gain