ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik Computer Science

Università della Svizzera italiana | Facoltà di scienze informatiche

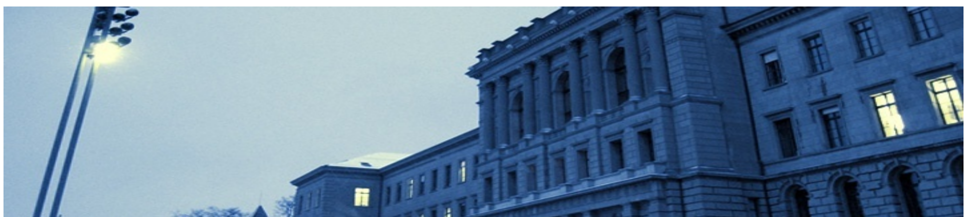
Automatic Configuration of an Autonomic Controller - An Experimental Study with Zero-Configuration Policies

Thomas Heinis[1] and Cesare Pautasso[2]

heinist@inf.ethz.ch, cesare.pautasso@unisi.ch

[1]Systems Group, Department of Computer Science, ETH Zurich

[2]Faculty of Informatics, University of Lugano



June 4th, 2008

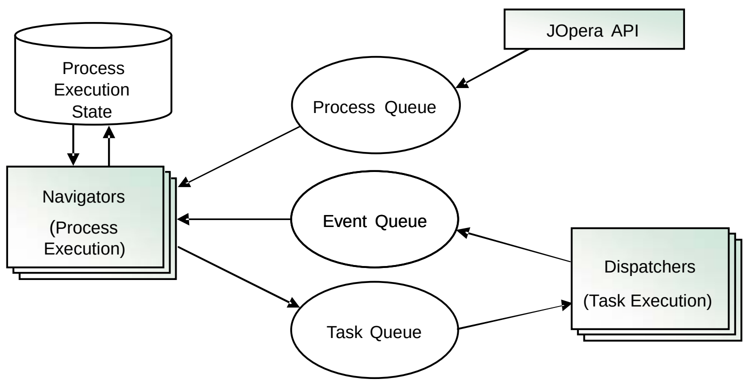
ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

JOpera Process Support for Web Services

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

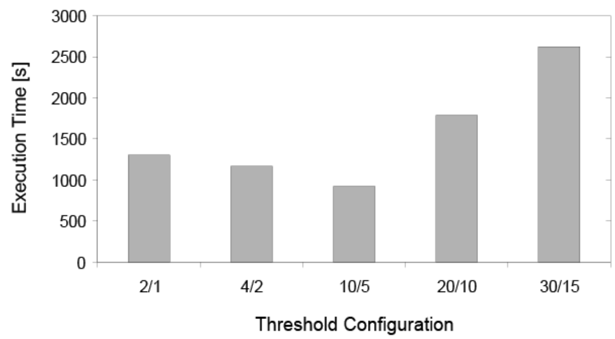


Autonomic Configuration

- 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:
  - $Q_{Process} + Q_{Event} = Q_{Task}$
  - Control Error = $(Q_{Process} + Q_{Event}) / Q_{Task}$
- Control Actions:
  - If $Q_{Process} + Q_{Event} < Q_{Task}$ => 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:

  Growth Q_{Event}= #Dsps * #Msgs * Production Rate – #Navs * 1 * Consumption Rate

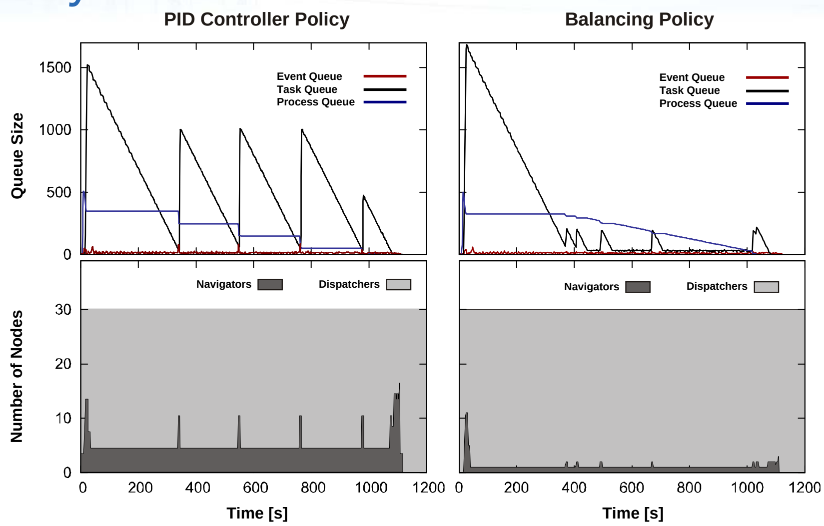
  Growth Q_{Task} = #Navs * #Msgs * Production Rate – #Dsps * 1 * Consumption Rate

  #Dsps = Size of Cluster - #Navs
- Production and consumption rates are measured at runtime
- #Messages is determined analytically

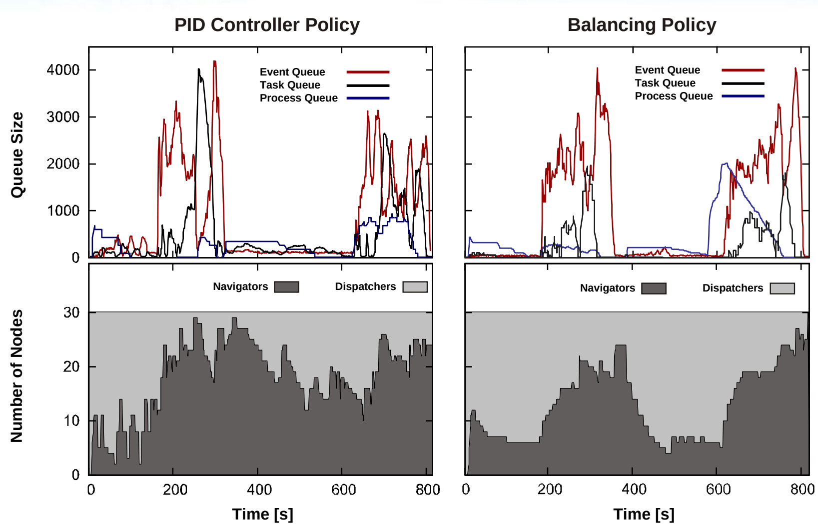
Evaluation

- 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

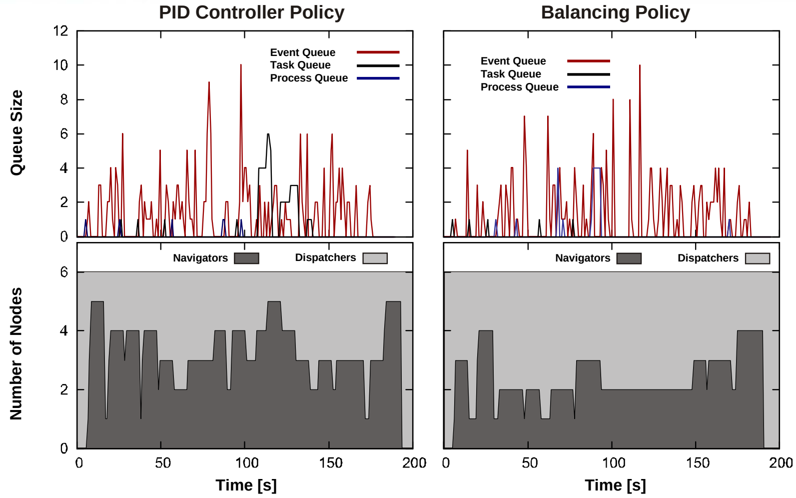
Busy Workload



Burst Workload



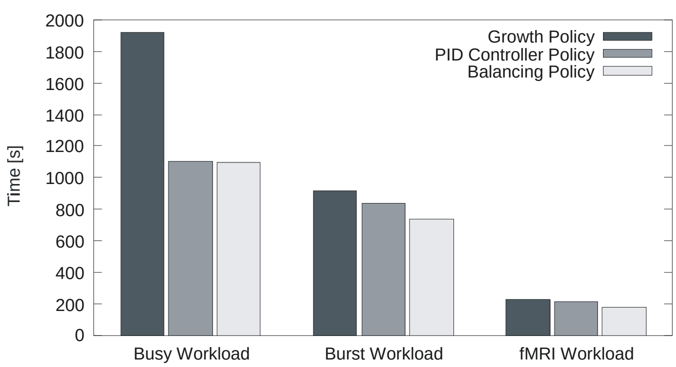
fMRI Workload



ETH
Jopera
Process Support for Web Services

Comparison

- Execution time per policy and workload



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