4th Grenoble Workshop on Autonomic Computing and Control

Bogdan ROBU (Gipsa-lab) & Eric RUTTEN (Ctrl-A)

supported by HPES project of ANR Persyval-lab
Outline

• Intro : Autonomic Computing, feedback loops and control

• Some past and ongoing loops

• Programme
Ctrl-A team: Control for Autonomic Computing

Automated administration & regulation in reaction to variations in load, resources,… in large (Big Data) or embedded (IoT) systems

promising, but challenge in developing applications need for automation & separation of concerns

Understand and design control for problems in efficiency (e.g.; energy) & assurances (e.g.crash avoidance)
Motivation

• Our approach: Software Engineering:
  • Middleware-level instrumentation and architectures,
  • Model-based control (Discrete Event Systems),
  • Programming support (reactive, components)

• Targets: HPC, IoT coarse grain, mid-size, heterogeneity
  problems: navigation in configurations space

• Method: propose generic models and validate them
  to attack lack of models & wide range of problems

• Multidisciplinarity: autonomic computing, languages
  + control theory, target platforms (HW/SW)

• Our goal: languages & model-based Softw. Eng. methods
  validated in target domains: HPC, reconfigurable arch., IoT
Autonomic Computing feedback loop

**interfaces** between managed element and manager

**objective** to be enforced by the manager, using **Control Theory**

**self-configuration, self-optimization, self-healing, self-protection**

**knowledge/model** of the managed element used by the manager

**pace** at which decisions are made (frequency, event-based)

---


Outline

• Intro: Autonomic Computing, feedback loops and control
• Some past and ongoing loops
• Programme
Parallelism/synchronization trade-off in STM

**intuition** lower parallelism when synchronization cost increases

**interfaces** sensors: commits, aborts, time; actuator: nb. threads

**objective** maximize throughput (commits/time)

**knowledge/model** commit ratio \( CR = \frac{\text{commits}}{\text{commits} + \text{aborts}} \)

probabilistic model: \( n_{opt} = -\frac{n-1}{\ln CR} \)

**pace** when \( CR \) changes by more than \( X\% \)

---

DPR FPGA reconfiguration control

intuition change bitstreams according to application/environment interfaces sensors: progress and environment events; actuator: choice of bitstreams to execute, tile on/off

objective maintain QoS while minimizing resources knowledge/model configuration states space (FSM): tasks versions w. different characteristics/costs, application DAG pace when event

reconfiguration controller

DPR FPGA

start task version, tile on/off

task end, value, tile fault, battery level, req appli

new project ANR HPeC

Software components reconfiguration

**intuition** reconfigure component assemblies (instances, bindings)

**interfaces** sensors: applicative, FIFO levels, ...

actuators: actions on components/composites

**objective** for Comanche server: avoid overload, logging/caching

**knowledge/model** components versions/implems, DSL & contracts incl., composites assembly configurations (FSM)

**pace** when event

controller in component membrane

req, load, FIFO levels

add/rm, bind/unbind

Fractal/Frascati MW

T. Bouhadiba, Q. Sabah, G. Delaval, E. Rutten. *Synchronous Control of Reconfiguration in Fractal Component-based Systems – a Case Study* EMSOFT’11

Overload avoidance in a HPC cluster

**intuition** regulate jobs sent according to load

**interfaces** sensor: current number of jobs processed;
actuator: max. number of new jobs to be sent to the cluster

**objective** maximize cluster utilization while avoiding overload

**knowledge/model** identification; PI controller

**pace** periodic

---

Coordinating sizing & repair in Multi-tiers

intuition avoid interference/redundancies between loops

interfaces sensor: load, faults; manager activity
actuator: add/rm server; suspend manager action

objective suspend downstream mgrs when upstream busy

knowledge/model activity state of mgrs (FSM)

pace when event

Conclusion

- **feedback loops** are all around
- **control techniques** are there to design efficient and well-mastered autonomic managers
- **a variety of loops**: differences and commonalities, some know-how, ...
Outline

• Intro : Autonomic Computing, feedback loops and control
• Some past and ongoing loops
• Programme