

Self-adaptive Distributed Systems

Corso di Sistemi Distribuiti e Cloud Computing A.A. 2024/25

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

Self-adaptive software systems

"Intelligence is the ability to adapt to changes" S. Hawking

- We aim to a software system capable of adapting its operations at run-time with respect to itself and the environment: a self-adaptive (or autonomic) software system
 - Applications of self-adaptive systems in many computing environments
 - Cloud computing
 - Edge/fog computing
 - Compute continuum
 - HPC
 - Cyber-physical systems

- Autonomic computing: computing paradigm able of responding to the need of managing IT systems complexity and heterogeneity through automatic adaptations
 - Inspired by human autonomic nervous system, able to control some vital functions (heart rate, digestion, temperature, ...) masking their complexity to humans
- A self-adaptive (or autonomic) software system can: ٠
 - Manage its functionalities and goals autonomously (i.e., without or with minimal human intervention)
 - Handle changes and uncertainty in its environment and system itself

Kephart and Chess, The vision of Autonomic Computing, IEEE Computer, 2003 Valeria Cardellini - SDCC 2024/25





Self-healing networks, www.juniper.net

Autonomous Vision: Effortless, Limitless, Unbreakable Data Cloud





Autonomous database,

www.oracle.com/autonomous-database/whatis-autonomous-database/



Goals of self-adaptive systems

- A self-adaptive (self-*) software system is able to selfmanage, pursuing the following goals:
- Self-optimize
 - Capability of system to optimize its resource usage or performance while providing its required quality goals
 - E.g., change placement of application components onto system nodes to satisfy application response time
- Self-heal
 - Capability of system to discover, diagnose and recover from faults to provide its required quality goals or degrade gracefully otherwise
 - E.g., detect crashed nodes and exclude them from serving requests

Valeria Cardellini - SDCC 2024/25

Goals of self-adaptive systems

Self-configure

- Capability of system to automatically integrate new elements, without interrupting the system's normal operation, or tune some configuration parameters
- E.g., discover new nodes and add them to serve requests

• Self-protect

- Capability of system to detect anomalies (i.e., intrusion detection) and react to intrusion and attack actions and its consequences (i.e., intrusion response) so to protect from security threats
- E.g., in a network of IoT devices detect a jamming attack that corrupts network traffic and adapt packets schedule

How to achieve the goals of a self-* system?

- The system should know its internal state (selfawareness) and the current external operating conditions (self-situation)
- Should identify changes regarding its state and the surrounding environment (self-monitoring)
- And should adapt consequently (self-adjustment)
- These attributes are the implementation mechanisms

Valeria Cardellini - SDCC 2024/25

Conceptual model of self-adaptive system



Conceptual model of self-adaptive system



Valeria Cardellini - SDCC 2024/25

Conceptual model of self-adaptive system



Conceptual model of self-adaptive system



Valeria Cardellini - SDCC 2024/25

You are already familiar with this model

The control-theory perspective of a self-adaptive system



• MAPE (Monitor, Analyze, Plan, Execute) loop



Valeria Cardellini - SDCC 2024/25

12

MAPE: building blocks (or phases)

- Monitor
 - Collects data from the managed system and execution environment through sensors; aggregates, filters and correlates these data into symptoms that can be analyzed
- Analyze
 - Observes and analyzes situations to determine need for adaptation
 - If adaptation is required, it triggers Plan
- Plan
 - Determines which mitigation actions need to be performed so to enact a desired alteration in the managed system
- Execute
 - Enacts the change plan by carrying out the actions determined by Plan through effectors so to adapt the managed system
- Plus Knowledge (MAPE-K)
 - Stores shared knowledge regarding relevant aspects of the managed system, environment, and the administrator's goals

- Main design options for Monitor:
 - When to monitor: continuously, on demand
 - What to monitor: resources, workload, performance, ...
 - How to monitor: architecture (centralized vs. decentralized), methodology (active vs. passive)
 - Where to store monitored data (e.g., time-series database) and how (e.g., some pre-processing)

Valeria Cardellini - SDCC 2024/25

14

MAPE: Analyze

- Main design options for Analyze:
 - When to analyze: event- or time-triggered
 - Reactive vs. proactive adaptation
 - *Reactive*: *in reaction to* events that have already occurred (e.g., increase number of resources *after* workload increase)
 - Proactive: based on prediction so to plan adaptation actions in advance (e.g., increase number of resources before workload increase occurs)

- The most challenging and studied MAPE phase
- A variety of methodologies and techniques can be used to plan adaptation, including
 - Optimization theory
 - Optimal adaptation actions
 - X Con: formulation can be NP-hard, too expensive to solve at runtime
 - Heuristics
 - ✓ Faster
 - X Sub-optimal adaptation actions
 - Machine learning, including reinforcement learning
 - Control theory
 - Queueing theory
- Example: optimal bin packing and heuristic policies to dynamically place virtual machines or containers on servers

Valeria Cardellini - SDCC 2024/25

16

Example: VM/container placement

- VM/container placement problem can be modeled as bin packing optimization problem
- <u>Bin packing</u>: pack items of different sizes (VMs/containers) into a minimum number of bins (server nodes), each of a given capacity (amount of resources), such that total size of items in each bin does not exceed bin capacity



 Integer optimization problem ⇒ NP-hard ⇒ we need efficient heuristics to find a new placement when some change occurs

Example: VM/container placement

- Some examples of baseline heuristic policies
- *Round robin:* organize bins in a *circular* list, saving the latest bin used for placement; allocate each item on next bin with enough capacity, starting from current position on list
 - Does not minimize number of used bins
- *First fit:* organize bins in a list and place each item into *the first bin* in which it fits, restarting for each item at the beginning of list
 - First fit decreasing: variant in which items are sorted in decreasing order
- Many other heuristics, e.g.,
 - Best fit: place item into the bin with the minimum amount of capacity into which the item can fit
 - Worst fit: similar to best fit, but maximum

Valeria Cardellini - SDCC 2024/25

18

Alternative architectures for MAPE

- How to design the managing system?
 - Centralized MAPE: all MAPE components on same node, simpler but lack of scalability in geodistributed environments
 - Decentralized MAPE: MAPE components are distributed; many architectural patterns, each one with pros and cons
 - No clear winner, it depends on system and application features and requirements



How to decentralize the adaptation control

· Main architectural patterns for decentralized MAPE



Weyns et al., On patterns for decentralized control in self-adaptive systems. In SEfSAS II, 2013

Valeria Cardellini - SDCC 2024/25

20

How to decentralize the adaptation control

- First design choice: hierarchical vs. flat
 - Hierarchical: easier to design, but risk of bottleneck in top level of hierarchy
 - Flat: more difficult to coordinate, but can scale better
- Hierarchical MAPE patterns: multiple MAPE loops organized in a hierarchy, where a higher-level control loop manages subordinated control loops
 - Master-worker
 - Hierarchical control
- Flat MAPE patterns: multiple MAPE loops cooperate as peers
 - Coordinated control
 - Information sharing

Hierarchical MAPE: master-worker pattern

- Decentralize M and E on workers, keep A and P centralized on master
- Global view on master who can achieve global adaptation goals
- X Communication overhead and risk of performance bottleneck and SPOF on master



Valeria Cardellini - SDCC 2024/25

22

Hierarchical MAPE: hierarchical control pattern

- Multiple MAPE loops, which can operate at different time scales and with separation of concerns
- Top-level MAPE can achieve global goals, increased flexibility
- X Can be non-trivial to identify different levels of control, depends on managed system characteristics



Flat MAPE: coordinated control pattern

- Multiple control loops, each one in charge of some part of the managed system but coordinated through interaction
- ✓ Better scalability
- X More difficult to take joint adaptation decisions



Valeria Cardellini - SDCC 2024/25

24

Flat MAPE: information sharing pattern

- Special case of coordinated control pattern: interaction only among M components
- ✓ Better scalability
- X Lack of coordination on planning, conflicting or suboptimal adaptation actions can be enacted



Examples of self-adaptive systems

- Let's analyze 3 examples of self-adaptive systems for resource management
 - 1. Auto-scaling EC2 instances
 - 2. Selecting services of composite applications
 - 3. Auto-scaling microservice-based applications
- Common ground
 - Applications face unexpected events (e.g., workload surge and spikes, node crashes)
 - Adaptation goal: satisfy some SLO (e.g., based on application response time, application availability)
 - Examples differ in planning methodologies and control architectures

Valeria Cardellini - SDCC 2024/25

26

Example 1: Amazon EC2 Auto Scaling

AWS service to automatically add or remove EC2
 instances according to user-defined conditions and

health checks https://aws.amazon.com/ec2/autoscaling/





- Which scaling metrics?
 - CPU utilization, network I/O, Application Load Balancer request count, …

Example 1: Amazon EC2 Auto Scaling

- MAPE Plan: we study 2 policies implemented in Auto Scaling for determining scale-out/in decisions https://docs.aws.amazon.com/autoscaling/ec2/userguide/scale-your-group.html
 - Dynamic simple scaling
 - Predictive scaling
- In addition to auto-scaling, Auto Scaling is a selfhealing system
 - Can detect when an EC2 instance is unhealthy, terminate it, and launch a new instance to replace it

Example 1: Amazon EC2 Auto Scaling

- Dynamic simple scaling: user-defined scaling plan to decide when and how to scale (*reactive*)
- Based on a threshold-based heuristic policy
 - Set upper and lower thresholds on some scaling metric(s)
 - if (metric > upper_thr) scale-out
 - else if (metric < lower_thr) scale-in</pre>
 - Example of scale-out rule: if average CPU utilization of all instances > 70% in last 1 minute, then add 1 new instance
 - Example of *scale-in* rule: if average CPU utilization of all instances is <35% in last 5 min, then remove 1 instance
 - CloudWatch monitors and sends alarms, one for scaling out (upper_thr) and the other for scaling in (lower_thr)
 - Cooldown period between each scaling activity

Threshold-based policy: pros & cons

✓ Simple and easy-to-understand: select scaling metric(s), period and thresholds for alarms X Not easy to choose metrics and thresholds Metric can be application-dependent: application components can be CPU/memory/IO-intensive or a mix Thresholds value can be either too aggressive or conservative, some example Slow scale-out, e.g., not enough instances added because upper thr is high \Rightarrow SLO violations occur > Rapid scale-out, e.g., too many instances added because upper thr is low \Rightarrow large underutilization and high cost Slow scale-in, e.g., not enough instances removed because scaling period is long \Rightarrow large underutilization and high cost Rapid scale-in, e.g., too many instances removed because lower thr is high \Rightarrow SLO violations occur No application-specific metric (e.g., response time) X Not robust against varying load patterns

Valeria Cardellini - SDCC 2024/25



Threshold-based policy: cons



Strasser et al., Autoscaler Evaluation and Configuration: A Practitioner's Guideline, ICPE 2023 https://research.spec.org/icpe_proceedings/2023/proceedings/p31.pdf

Example 1: Amazon EC2 Auto Scaling

- Predictive scaling: based on ML (proactive)
 - Trained ML model to predict application expected traffic and EC2 usage, including daily and weekly patterns
 - Requires historical data collected from CloudWatch
 - Model needs at least one day's of historical data to start making predictions
 - Re-evaluated every 24 hours to forecast for the next 48 hours



- ✓ Proactive
- X Requires training: the more the historical data, the more accurate the forecast
- X Choice of scaling metric is core

Valeria Cardellini - SDCC 2024/25

32

Example 2: Service selection

- QoS-driven self-adaptation of SOA applications
 - Multiple concrete services for each abstract service: how to select which concrete services to use so to satisfy application SLAs?



Cardellini et al., MOSES: a framework for QoS driven runtime adaptation of service-oriented systems, *IEEE Trans. Soft. Eng.*, 2012 <u>http://www.ce.uniroma2.it/publications/tse2012.pdf</u>

MAPE Plan

 Centralized policy: select optimal set of concrete services (and their coordination) by means of linear programming optimization



Valeria Cardellini - SDCC 2024/25

Example 3: Hierarchical scaling of microservices

- Hierarchical control pattern to elastically scale a microservices-based application
 - Goal: keep response time below maximum
 - Each Microservice Mngr. determines scale-out/in needs through local policy based on queueing theory (and workload prediction) and sends proposal to Application Mngr.
 - Application Mngr. coordinates scaling proposals by accepting or not them and sends decision to local Execute components



Rossi et al., Hierarchical scaling of microservices in Kubernetes, *Proc. IEEE ACSOS 2020* http://www.ce.uniroma2.it/publications/acsos2020.pdf 35

- The vision of Autonomic Computing
 <u>https://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf</u>
- An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective, chapter 1 https://introsas.cs.kuleuven.be/2020ExcerptBook.pdf
- On patterns for decentralized control in self-adaptive systems https://www.ics.uci.edu/~seal/publications/2012aSefSAS.pdf

Valeria Cardellini - SDCC 2024/25