RESOURCE MANAGEMENT

Chapter 11: Parallel and Distributed Software Systems

OUTLINE

11.1 Definition

11.2 Resource allocation algorithms

11.3 Autonomic resource allocation

11.4 Control theory

11.5 Performance models

11.6 Virtualized environments

DEFINITION

- Allocation of resources:
  - Crucial in distributed systems
  - To make sure applications run properly
  - Feedback to applications required

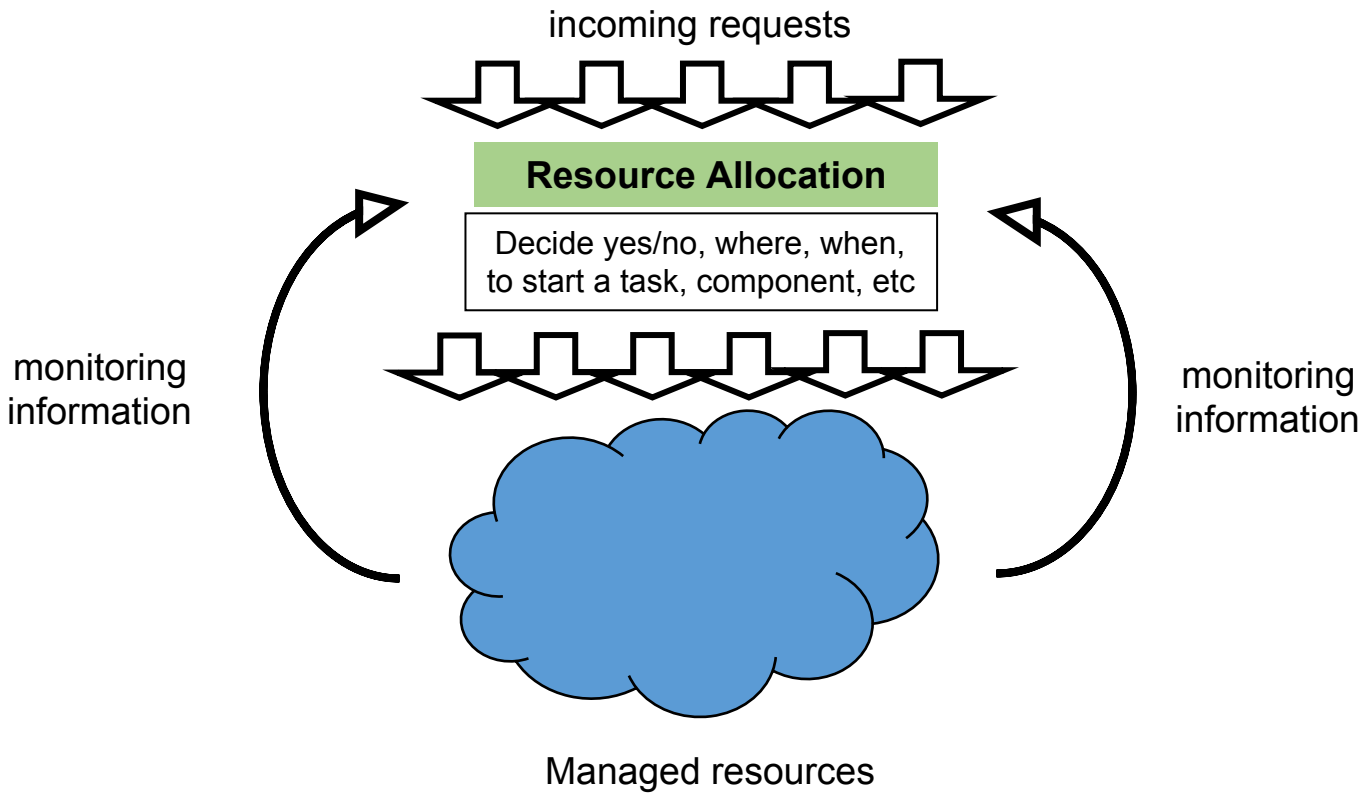
- Different type of resources:
  - CPU, memory, bandwidth and storage

- Objectives:
  - Minimize response times of applications

GHENT UNIVERSITY

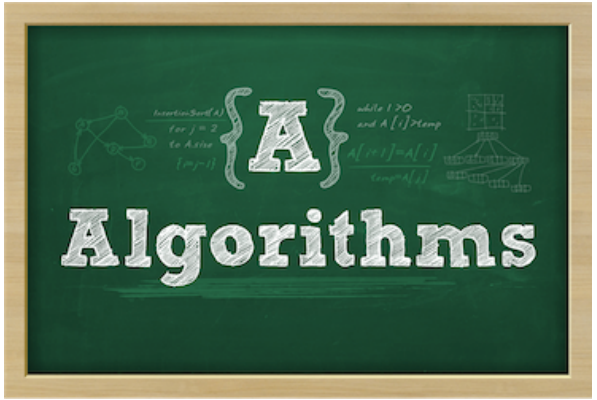
RESOURCE ALLOCATION

1. Definition



GHENT UNIVERSITY

RESOURCE ALLOCATION ALGORITHMS



Different types:

- Single request versus batch
- Optimal versus heuristical
- Online versus off-line
- Static versus dynamic

Models required:

- Resource models
- Request models

next:

- Integer Linear programming
- Metaheuristics
- Greedy algorithms

GHENT UNIVERSITY

INTEGER LINEAR PROGRAMMING

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \\ \text{and} \quad & \mathbf{x} \in \mathbb{Z}, \end{aligned}$$

Optimal solution

Scalability ?!

Example:

$$\begin{cases} x_1+x_2+x_3+x_4+x_5 \leq 8 \\ 6x_1-x_3+x_5 \leq 25 \\ x_1-x_2-2x_3 \leq 4 \\ 3x_4-x_5 \leq 3 \\ \\ x_1,x_2,x_3,x_4,x_5 \geq 0 \\ x_1,x_2,x_3,x_4,x_5 \leq 100 \end{cases}$$

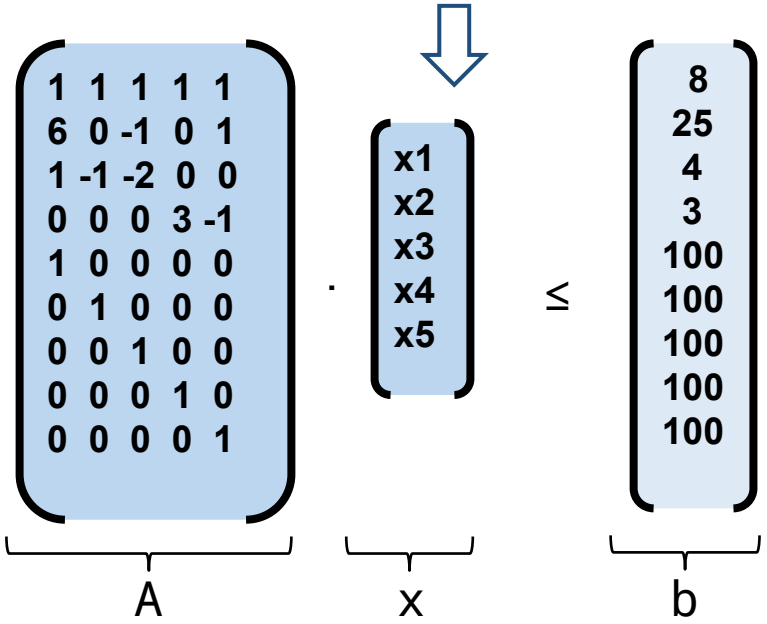
constraints

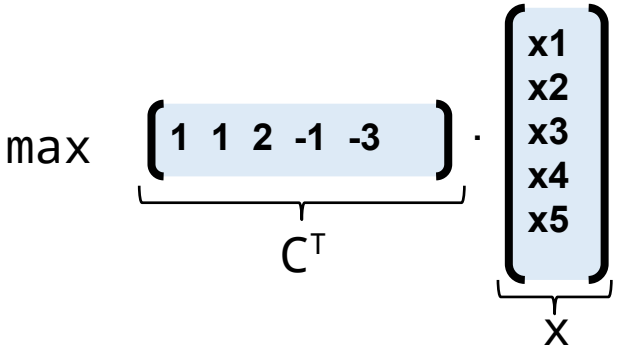
$$\max(x_1+x_2+2x_3-x_4-3x_5)$$

objective function

GHENT UNIVERSITY

INTEGER LINEAR PROGRAMMING





GHENT UNIVERSITY

INTEGER LINEAR PROGRAMMING

ILP methodology:

1. determine the decision variables and constraints
2. this provides the description of the vector x,
   and the values for the matrix A, vector b and c
3. a solver package then determines based on A, b and c:
   a. the values of the optimal x vector
   b. the feasibility of the solution

e.g. iLog CPLEX solver

- Simplex algorithm
- Branch and Bound algorithm

INTEGER LINEAR PROGRAMMING

Important :

1. solution process can be stopped after certain amount of time
2. deviation (in %) from optimal solution is available!

Decision variables in x:

can be:

1. resource utilization
2. resource allocation (which job on which machine)
3. binary decision which machine is switched on
4. binary decision whether a software component is assigned to a machine or not

INTEGER LINEAR PROGRAMMING

Example 1:

$$
\begin{array}{lllll}
\min & \displaystyle\sum_{i\in L}\sum_{j\in F} c_{ij}x_{ij} + \sum_{j\in F} f_j y_j & & & \\
\text{s.t.} & \displaystyle\sum_{j\in F} x_{ij} & = 1 & \forall i \in L & \text{(assign\_def)} \\
 & x_{ij} & \leq y_j & \forall i \in L, j \in F & \text{(link)} \\
 & \displaystyle\sum_{i\in L} d_i x_{ij} & \leq C y_j & \forall j \in F & \text{(capacity)} \\
 & x_{ij} \in \{0,1\} & & \forall i \in L, j \in F & \\
 & y_j \in \{0,1\} & & \forall j \in F &
\end{array}
$$

Example 2:

$$
\begin{array}{ll}
\min \displaystyle\sum_{i=0}^{n}\sum_{j\neq i, j=0}^{n} c_{ij}x_{ij} & \\
0 \leq x_{ij} \leq 1 & i,j = 0,\cdots,n \\
u_i \in \mathbf{Z} & i = 0,\cdots,n \\
\displaystyle\sum_{i=0, i\neq j}^{n} x_{ij} = 1 & j = 0,\cdots,n \\
\displaystyle\sum_{j=0, j\neq i}^{n} x_{ij} = 1 & i = 0,\cdots,n \\
u_i - u_j + n x_{ij} \leq n-1 & 1 \leq i \neq j \leq n
\end{array}
$$

Additional example in appendix!

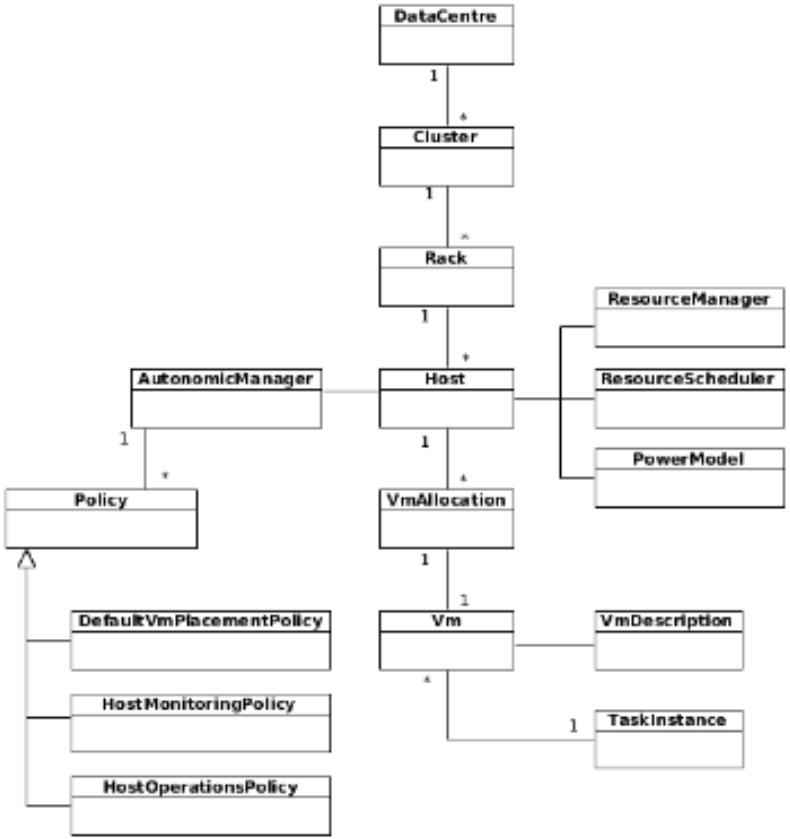
INPUT PARAMETERS

TABLE I: The model input variables.

Symbol	Description
G	The graph $G = (N, E)$ representing the network.
N	The nodes within the network. This collection can be partitioned into a set of computational nodes N^c and a set of service nodes N^s.
E	The edges within the network.
E_n^{in}	The edges that are incoming in node $n \in N$.
E_n^{out}	The edges that are outgoing from node $n \in N$.
$C(e)$	The bandwidth capacity of an edge $e \in E$.
$L(e)$	The network latency of an edge $e \in E$.
$\mathcal{C}$	The service chains that must be allocated on the network.
$R^{VM}(C)$	The VM requests that are part of service chain $C \in \mathcal{C}$.
$R^s(C)$	The service requests that are part of service chain $C \in \mathcal{C}$.
$R(C)$	The resource requests that are part of service chain $C \in \mathcal{C}$. $R(C) = R^{VM}(C) \cup R^s(C)$.
S	The collection of all services that exist within the model.
$D^\gamma(r)$	The resources of type γ needed for a request r. This request can either be a service request or a VM request.
Γ^c	The resource types that are available on a physical computational node. Typically these resources are CPU and Memory.
Γ^s	The resource types that are provided by a service $s \in S$. Typically this is a service-specific resource such as requests per second.
$\mathcal{R}$	A collection containing all of the resource requests of all service chains in $\mathcal{C}$.
C_n^γ	The resource capacity of a node or service n. For computational nodes, $\gamma \in \Gamma^{VM}$. For service nodes offering service s, $\gamma \in \Gamma^s$. A service VM of a service s provides C_s^γ resources of types $\gamma \in \Gamma^s$.
$D(r_1, r_2)$	The network demand between requested VMs or services r_1 and r_2.
$L(r_1, r_2)$	The maximum network latency between requested VMs or services r_1 and r_2.

GHENT UNIVERSITY

RESOURCE MODEL



GHENT UNIVERSITY

ILP SOLVERS

- e.g. iLog CPLEX solver

- Simplex algorithm
- Branch and Bound algorithm

- Execution times can be large
- Usage of HPC - after debugging!
- Stop criteria!

METAHEURISTICS

- Tabu search
- Simulated annealing
- Genetic programming (evolutionary algorithms)
- Ant-based algorithms or particle swarm algorithms (nature-inspired algorithms)

– Can serve as benchmarks (lower bound) for the exact solutions and the greedy algorithms.

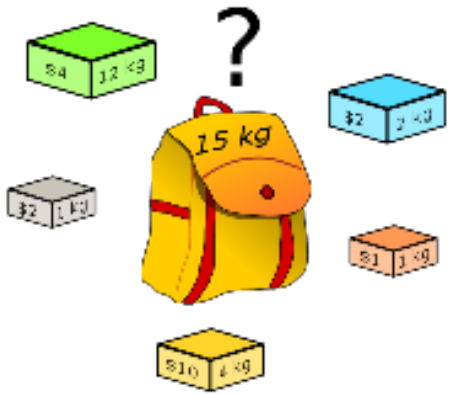
BIN PACKING ALGORITHMS

- objects of different volumes must be packed into a finite number of bins or containers, with a predetermined capacity C.
- goal: minimize the number of bins used.
- combinatorial NP-hard problem.
- greedy algorithm: **first fit algorithm**
  - fast but often non-optimal
  - placing each item into the first bin in which it will fit
  - can be made much more effective by first sorting the list of elements into decreasing order (also called first-fit decreasing algorithm)

KNAPSACK ALGORITHMS

Special case of bin packing algorithm:

- number of bins is restricted to 1
- each item is characterised by both a volume and a value
- the problem of maximizing the value of items that can fit in the bin



AUTONOMIC RESOURCE ALLOCATION

Structure:

3.1 Autonomic Systems

- Goal
- Adaptive vs. Autonomic
- Use cases

3.2 Control Loops

- MAPE
- OODA
- FOCALE

3.3 Architecture for distributed autonomic systems

3.4 Knowledge representation

3.5 Reasoning

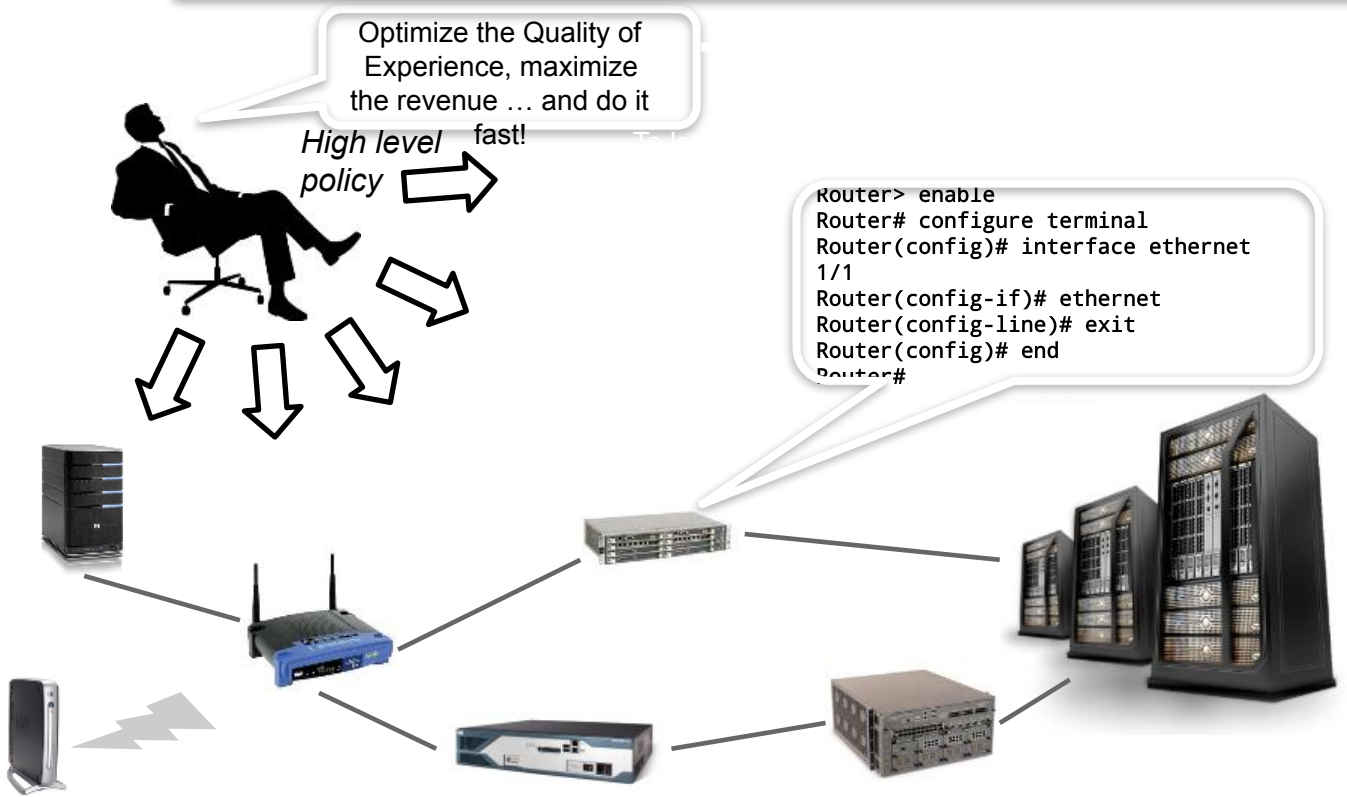
AUTONOMIC SYSTEMS

The system adapts its behavior to changes in
The environment
End-user needs
Service requirements

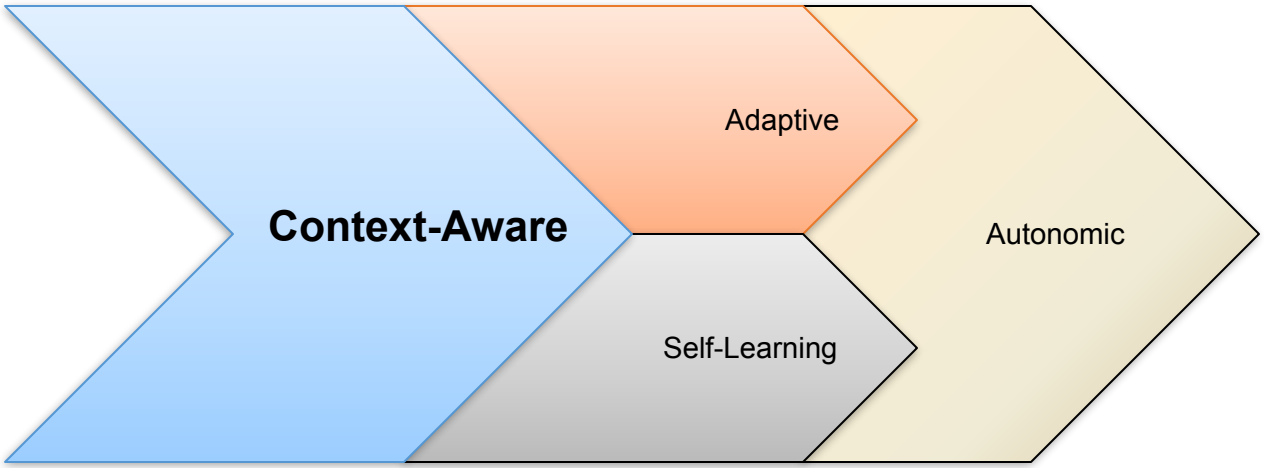
It is governed by high-level policies
Representing business goals
Defined and managed by human operators

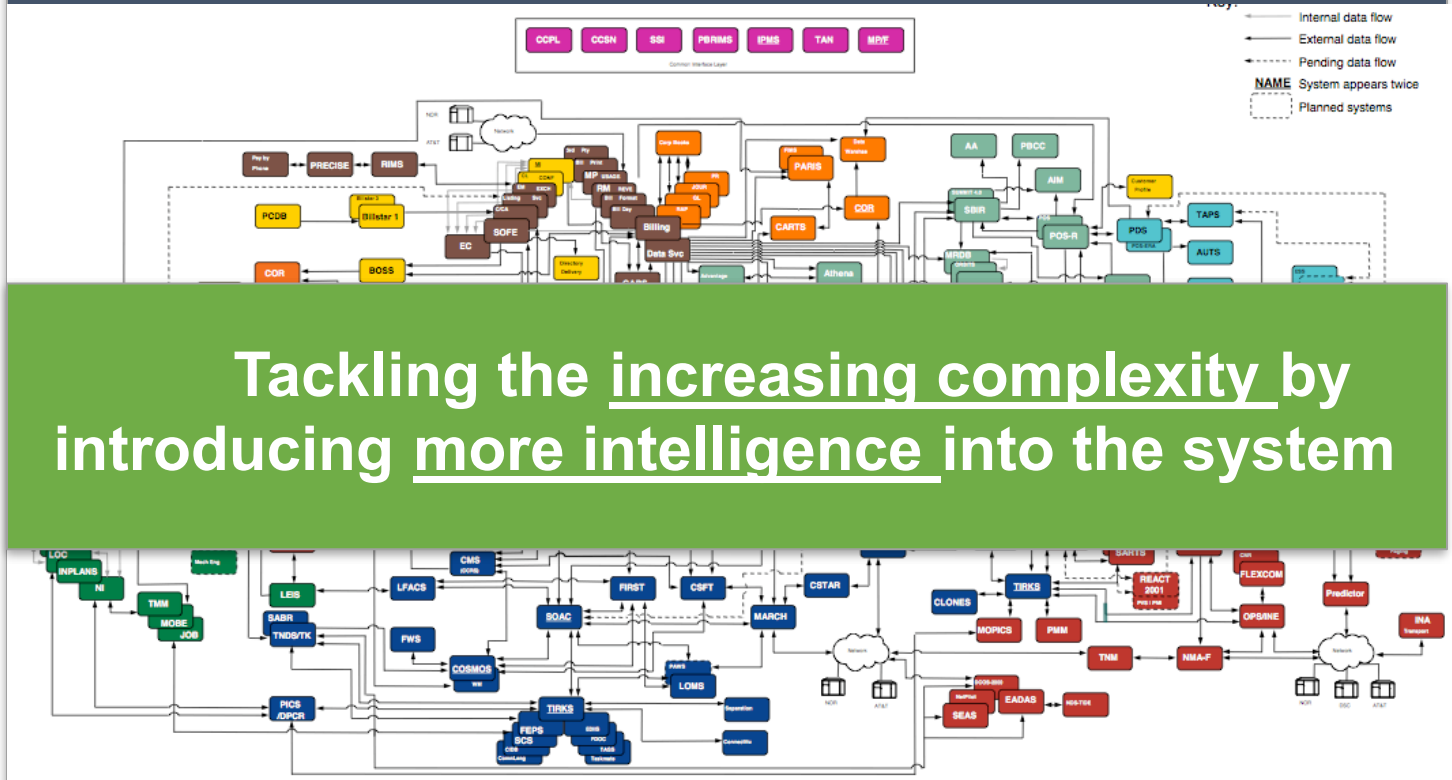
THE GOAL OF AUTONOMIC SYSTEMS

Autonomic systems decrease management complexity by performing low-level configurations themselves



ADAPTIVE VS. AUTONOMIC





Tackling the increasing complexity by introducing more intelligence into the system

GHENT UNIVERSITY

USE CASES

Cloud Computing

eHealth

Intelligent Transportation

Robotics

How do we design and implement an autonomic system?

DEFINITION – 4 CHARACTERISTICS

- Self chop characteristics:

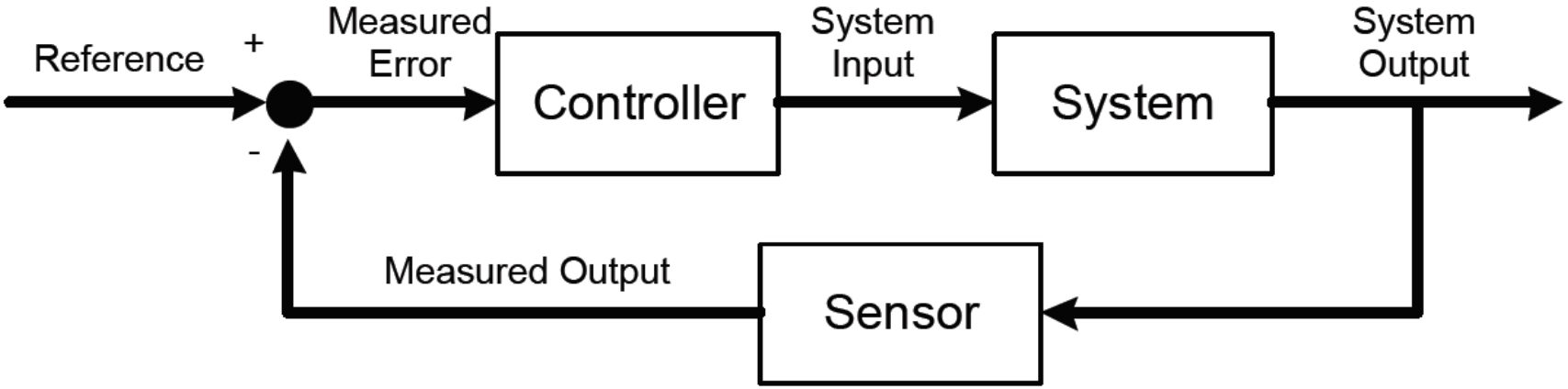
Self-configuration

Self-healing

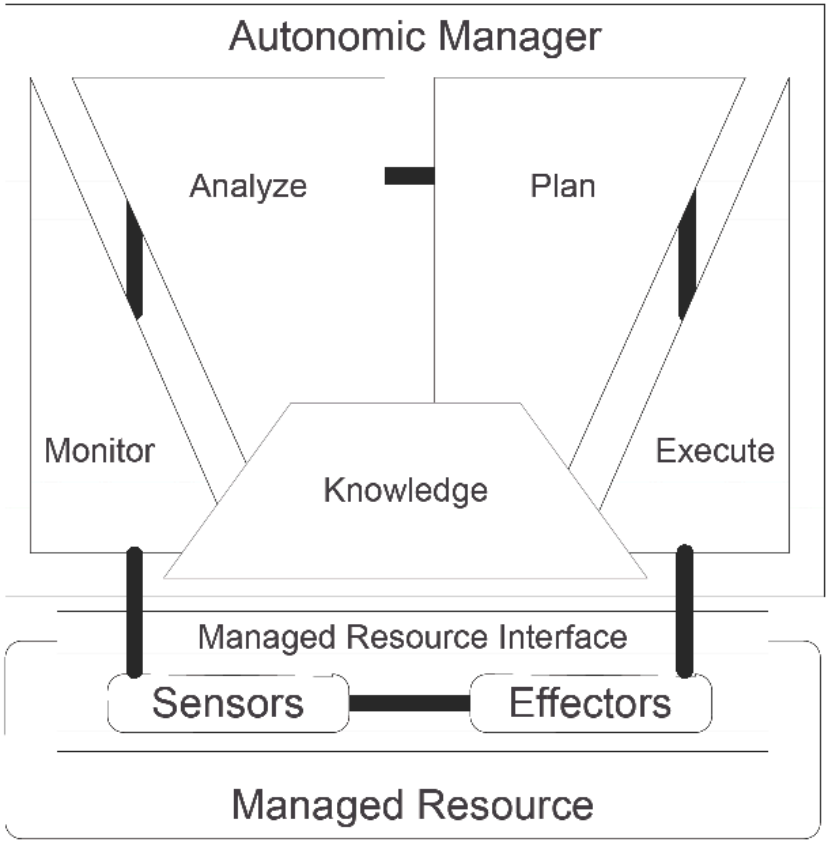
Self-optimization

Self-protection

CONTROL LOOPS

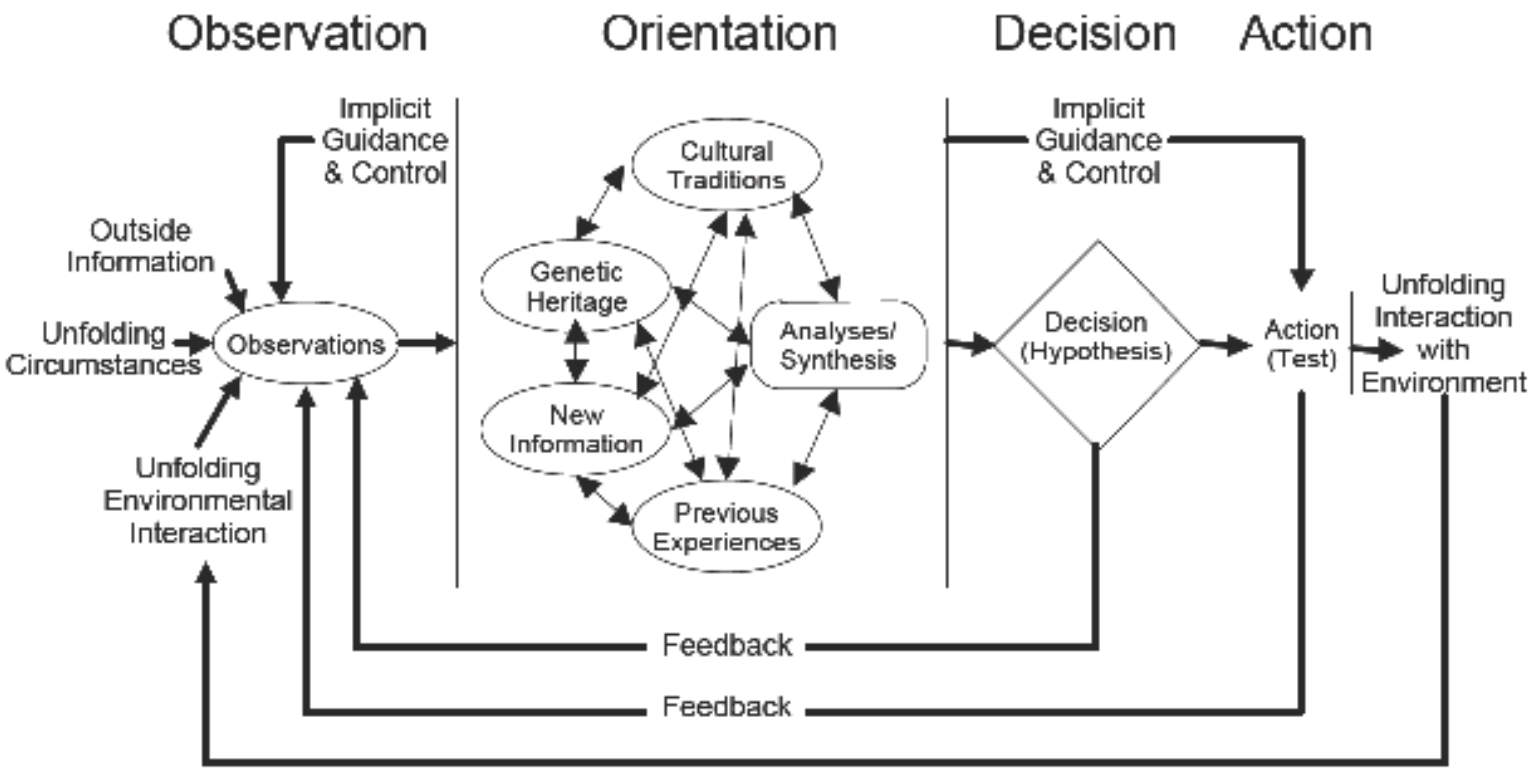


MAPE CONTROL LOOP

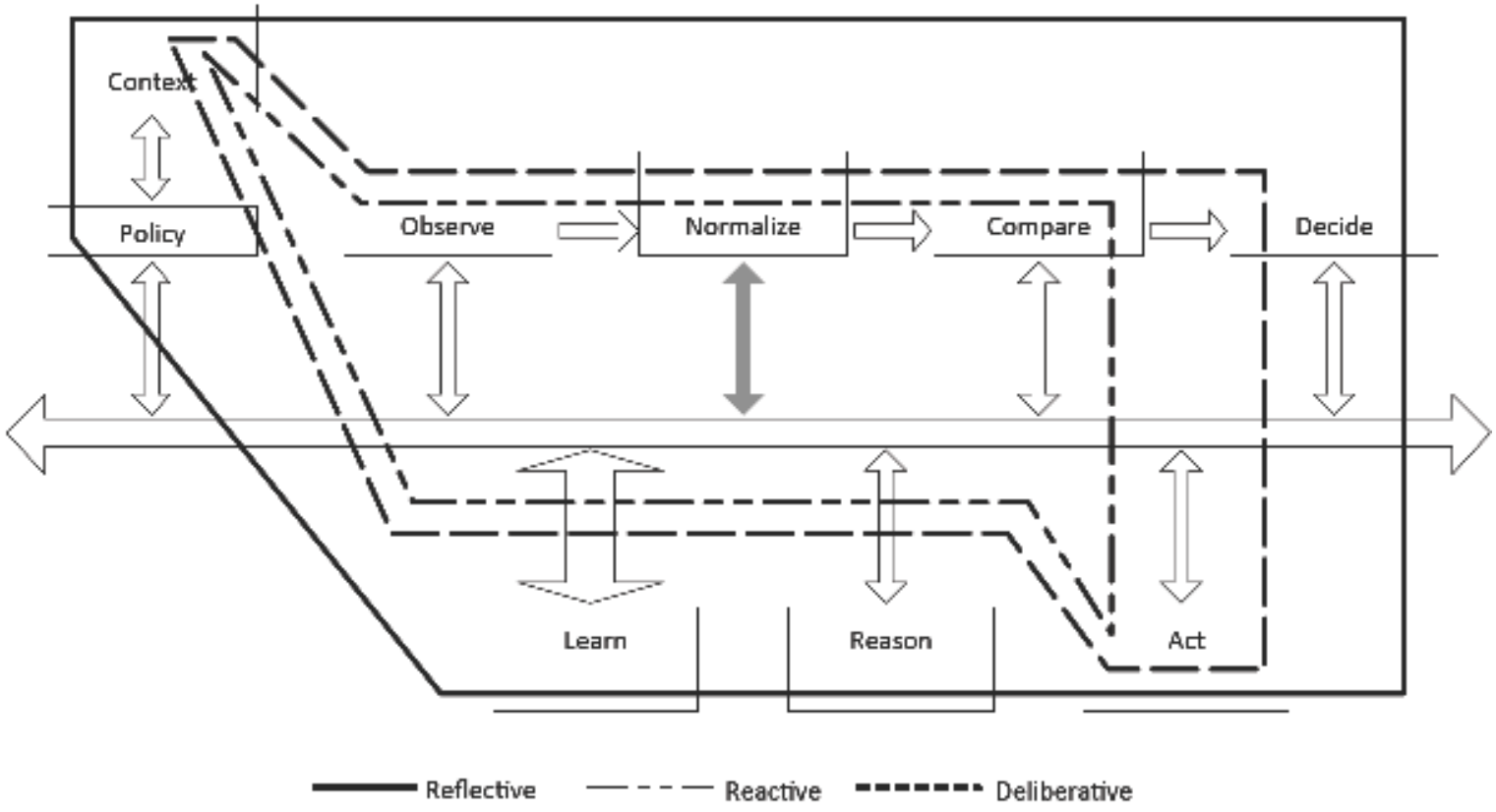


GHENT UNIVERSITY

OODA CONTROL LOOP



FOCALE CONTROL LOOP



CONTROL LOOP EXAMPLE