universität innsbruck

Advanced Distributed Systems

Lecture 05-Resource management and scheduling

Dr. Zahra Najafabadi Samani

Agenda

- Informal definition of design objectives
- Timing as a design objective
- Monetary cost as a design objective
- Energy as a design objective
- Reliability as a design objective
- Security as a design objective

Informal definition of Design objective

General design/optimization situation:

- Design of a *product* for a *client*
  - Design refers to the process of making a series of decisions related to the creation of the product
  - The product has to be designed a set of *requirements*
- Requirements are classified as either:
  - *Functional*
  - *Non-functional*

Informal definition of Design objective

General design/optimization situation:

- Design of a *product* for a *client*
  - Design refers to the process of making a series of decisions related to the creation of the product
  - The product has to be designed a set of *requirements*
- Requirements are classified as either:
  - *Functional* :
    - Define specific behaviors or functions of the system
    - Describe **what** the system should do
  - *Non-functional* :
    - Specify criteria used to judge the operation of the system
    - Describe **how** the system will perform the activity specified by the functional requirements

Informal definition of Design objective

Example: designing a smartphone

- *Functional ?*

- *Non-functional ?*

Informal definition of Design objective

Example: designing a smartphone

- *Functional* :

  include features such as the ability to
  - make calls
  - send text messages
  - access the internet
  - take photos
  - and run various applications

- *Non-functional* :
  - response time (how quickly an application loads)
  - battery life (how long the phone can operate on a single charge)
  - data security (how well user data is protected)
  - and user interface usability (how intuitive and user-friendly the interface is)

Informal definition of Design objective

The definition of the *objective functions* is...

- part of the problem modeling process.
- often underestimated in its importance.
- has a massive influence on the...
  - choice of the appropriate design approach.
  - obtainable results and...
  - their ultimate usefulness for the client.

Design objective- Workflow scheduling

During the scheduling of workflows, following objectives usually play an important role:

- Timing
- Monetary costs
- Energy Consumption
- Reliability
- Security

Timing as a design objective

Motivation and metrics

There are three main reasons why minimizing the processing time of workflows is desirable:

- Efficiency (cost reduction)
- Application deadlines (real-time capability)
- Quality of service

Efficiency

The relevance of execution efficiency depends on the **business model** under which the computing resources are leased. From the monetary perspective, processing a given workload faster is especially relevant for parties leasing or buying hardware, e.g.

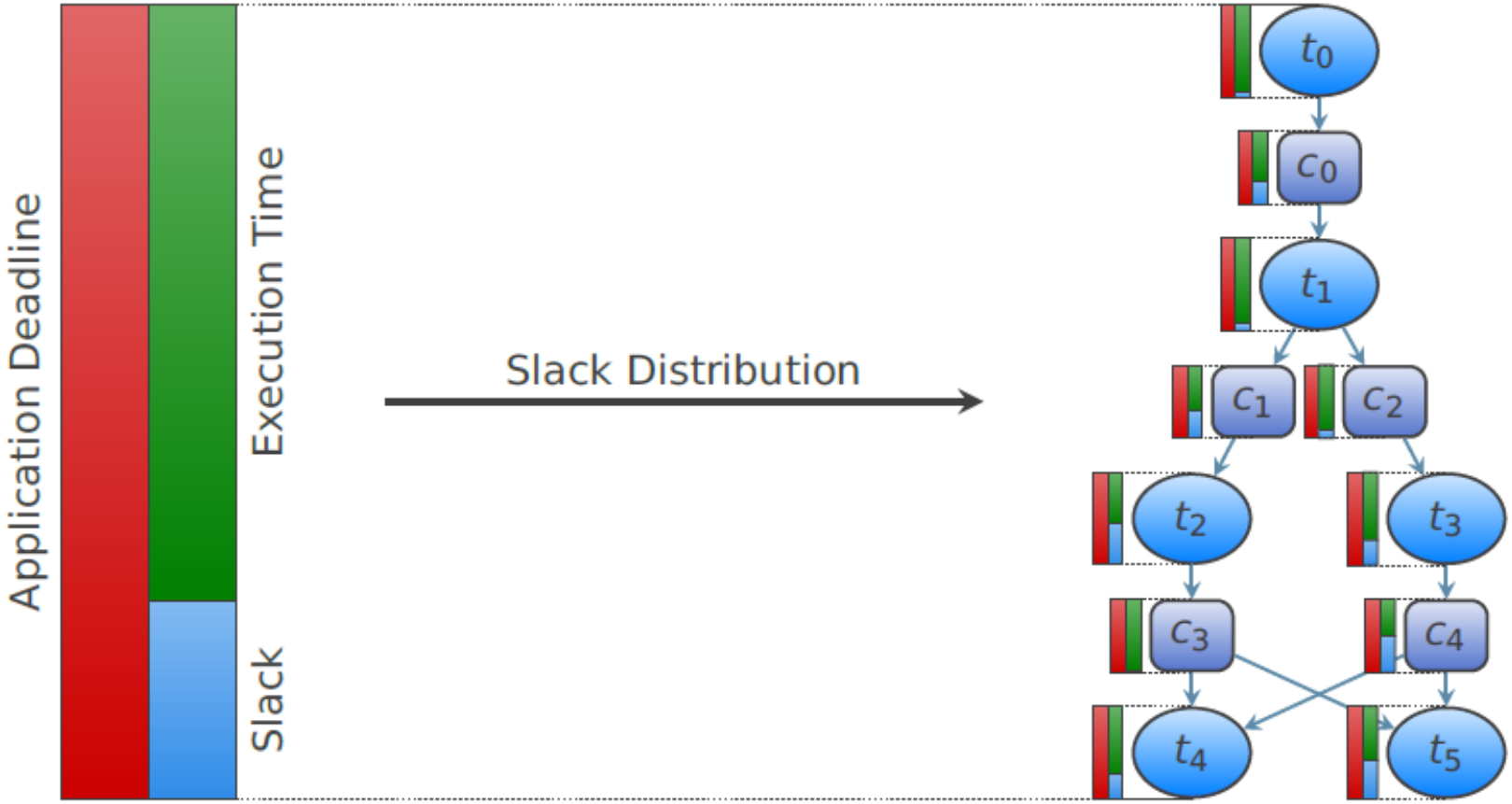
- clients using hardware-as-a-service business models.
- clients using cloud, fog, or edge solutions.
- cloud providers.

Application deadline

The requirements of *real-time* applications include *application deadlines*.

- **Deadline** is a time by which some task must be completed. Deadlines are typically defined as the maximum time between a triggering event and the time when a task, in this context, a task can refer to one or several nodes of the workflow graph or even the entire workflow, is finished.
- **Real-time system** is a system subject to real-time constraints, i.e., deadlines. Operating in real time **DOES NOT** mean operating "very very fast"!

Application deadline



Application deadline

Types of deadlines:

- *hard*: deadline misses result in a total system failures
- *firm*: infrequent deadline misses are tolerable, but degrade the quality of the system service
- *soft*: the usefulness of a result degrades after the deadline

Types of tasks/traffic:

- *Real-time* tasks/traffic: Tasks/traffic transmissions which are subject to deadlines
- *Safety-critical* real-time tasks/traffic: A special group of real-time tasks, where deadline misses potentially lead to catastrophic consequences, e.g., damage to the system or even loss of human life
- *Best effort* tasks/traffic: Tasks/transmissions for which the system does not provide any guarantees about the processing/transmission of the task/transmission, in particular w.r.t. quality-of-service metrics.

Application deadline

Deterministic execution vs. execution in real time:

- Execution in real-time: the application terminates **before** its deadline (the amount of slack may differ between executions).
- Deterministic execution: the application terminates at an **exact predefined** point in time.

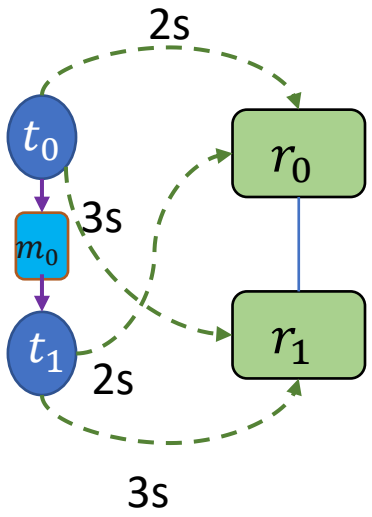
Most real-life applications fall into the category of real-time rather then deterministic applications, because it is important to have a result before a specified deadline (and having the result "too early" does not constitute a problem).

Objective type

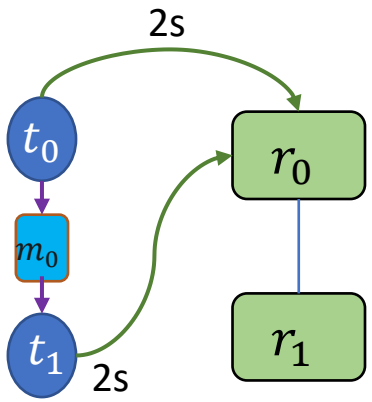
Makespan and ***throughput*** are two of the most common design objectives during the timing optimization of workflow schedules.

- **The makespan** of a workflow execution is defined as the time which elapses between the start of the processing of the first task and the end of the processing of the last task.
- **The throughput** of a system is defined as the amount of workload it can process within a given amount of time (or, analogously, as the amount of time which it requires for the processing of a given amount of workload).

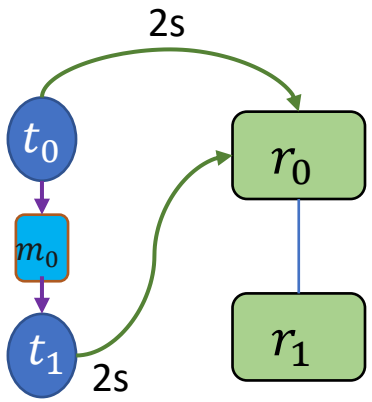
Objective type

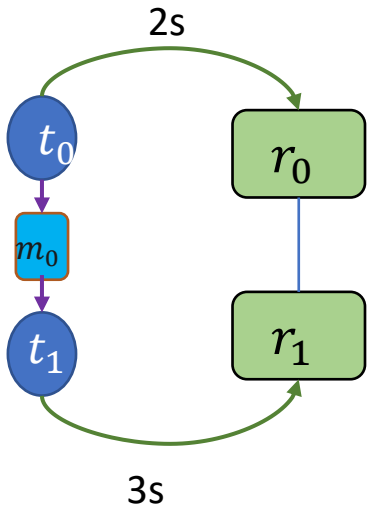


Objective type

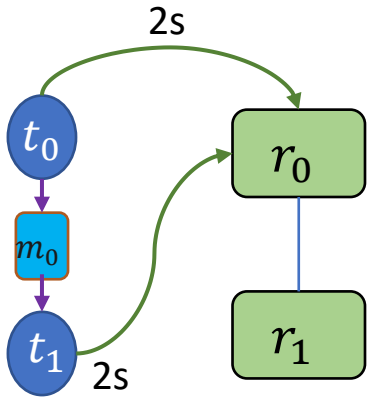


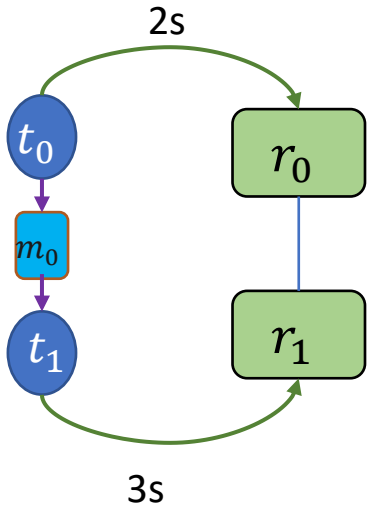
Objective type

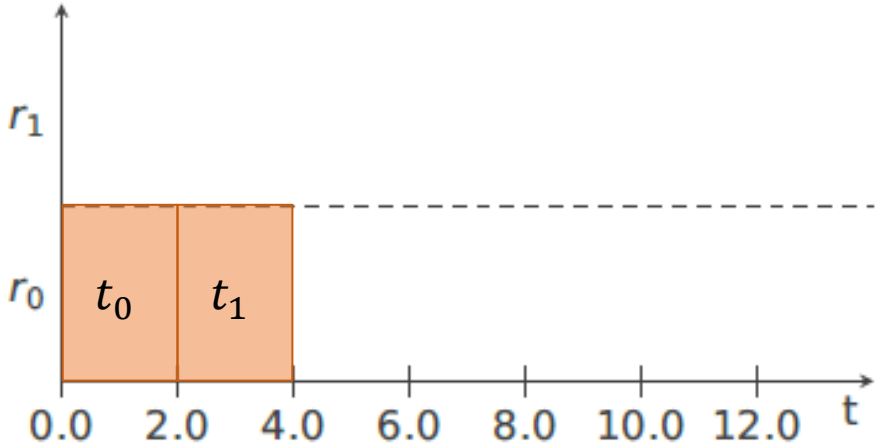


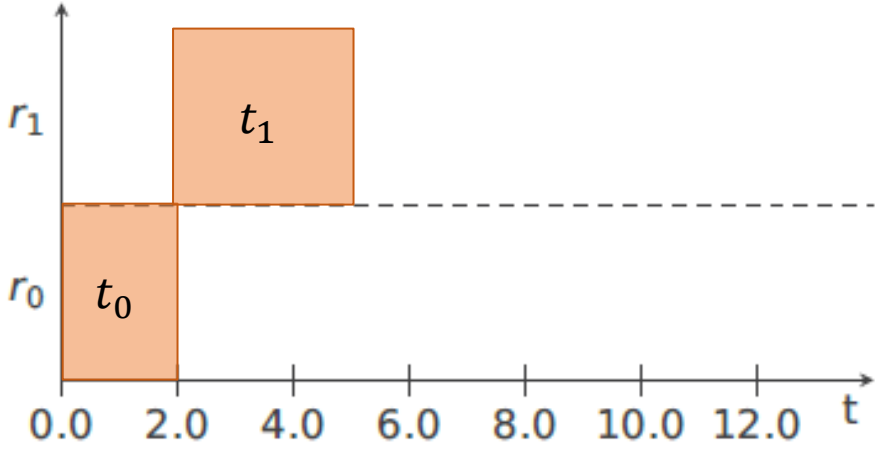


Objective type

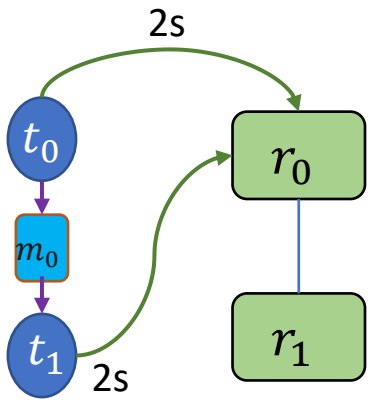


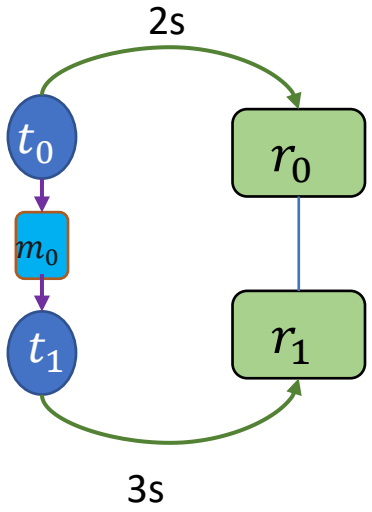


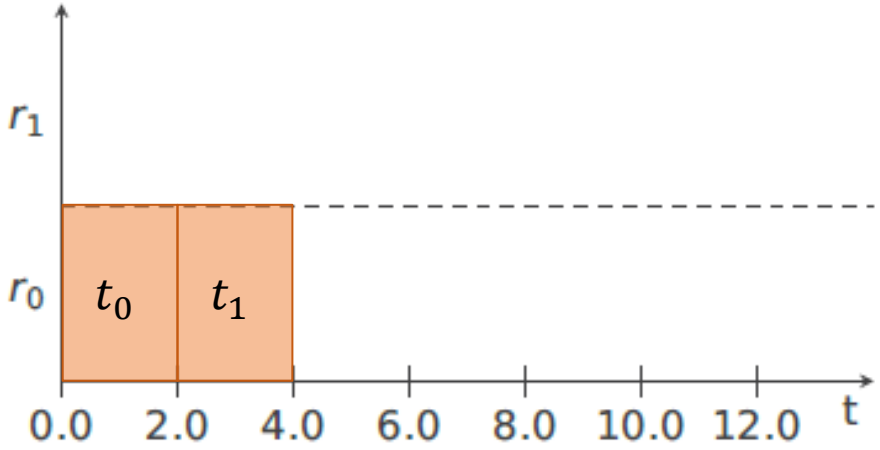




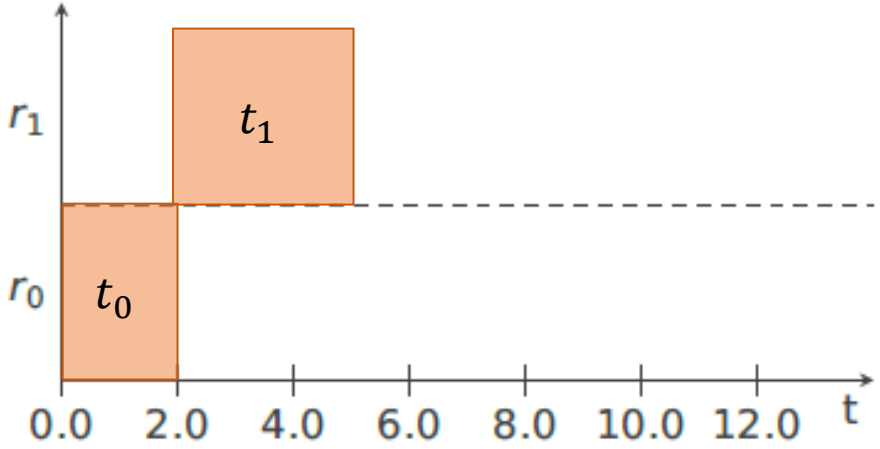
Objective type





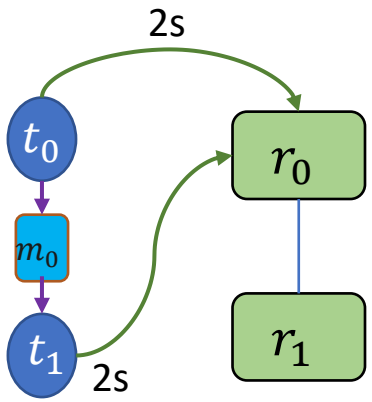


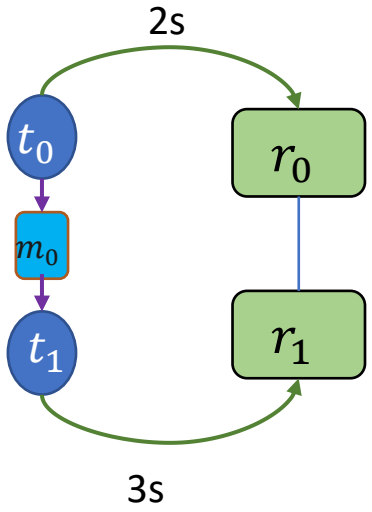
Makespan 4 s

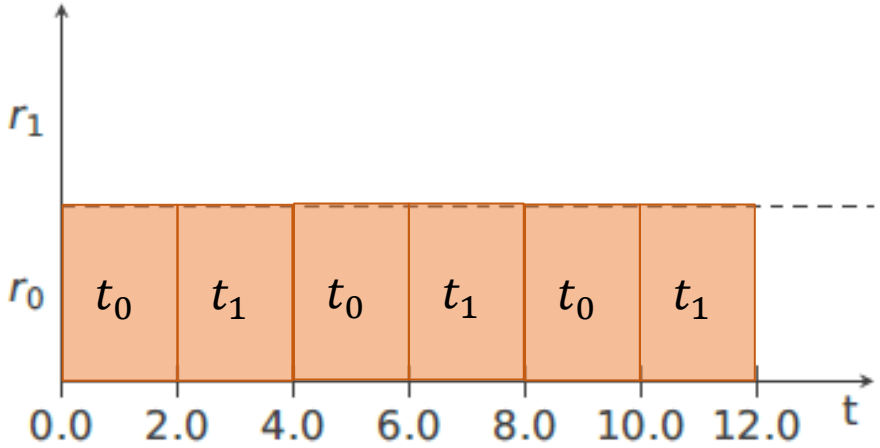


Makespan 5 s

Objective type

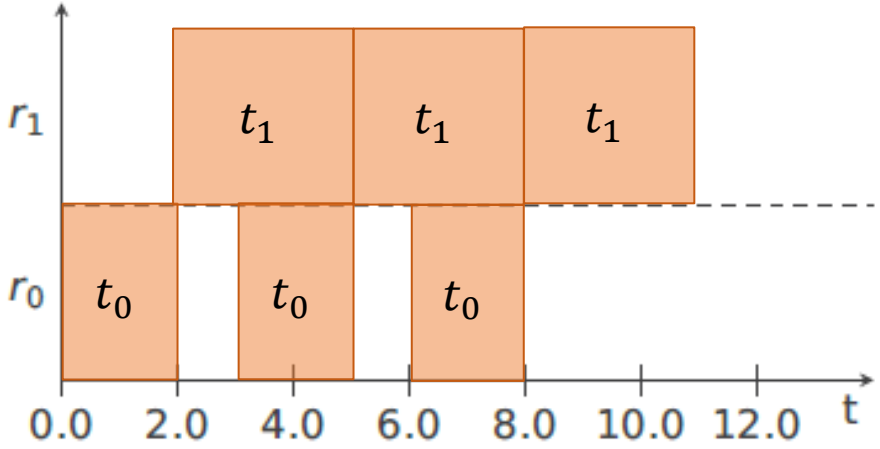






Makespan 4s

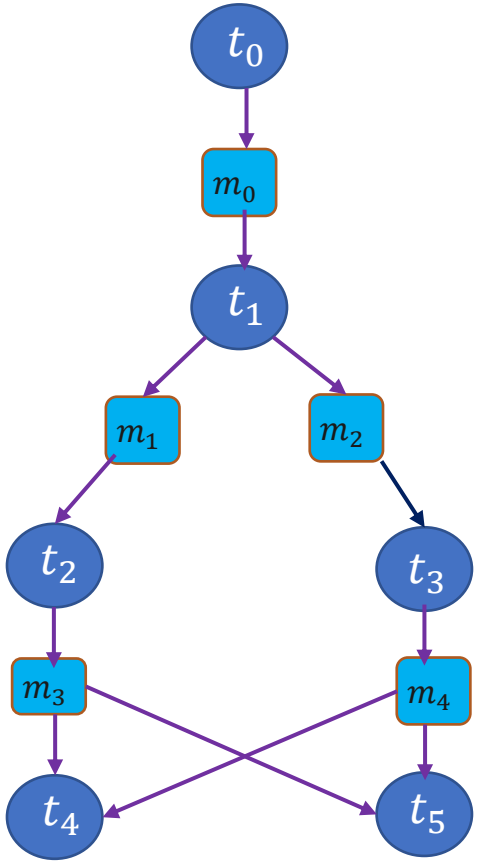
Throughput 4s



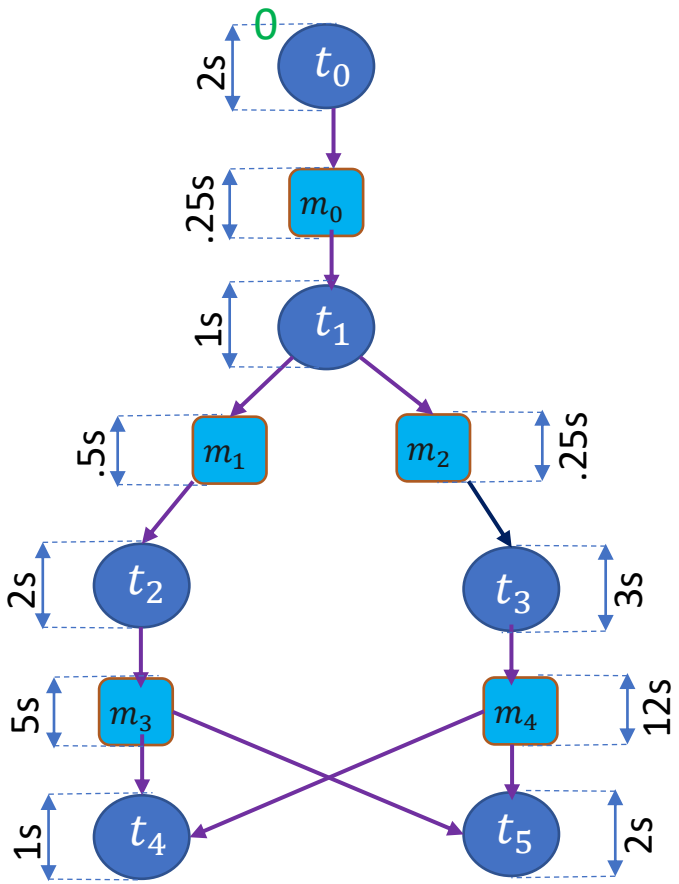
Makespan 5s

Throughput 3s

Makespan calculation



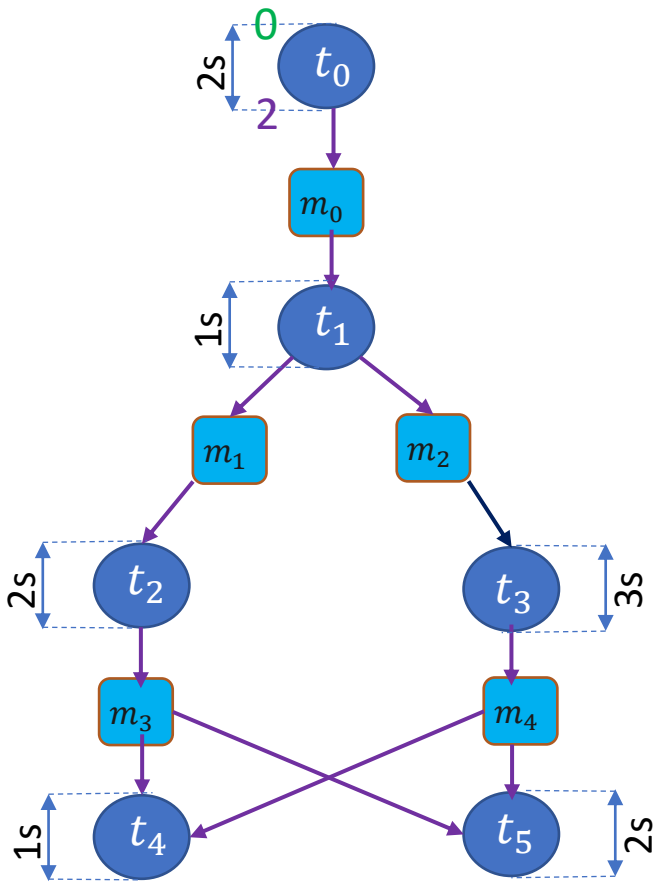
Makespan calculation



Makespan calculation:

Start time: The time that first task starts

Makespan calculation

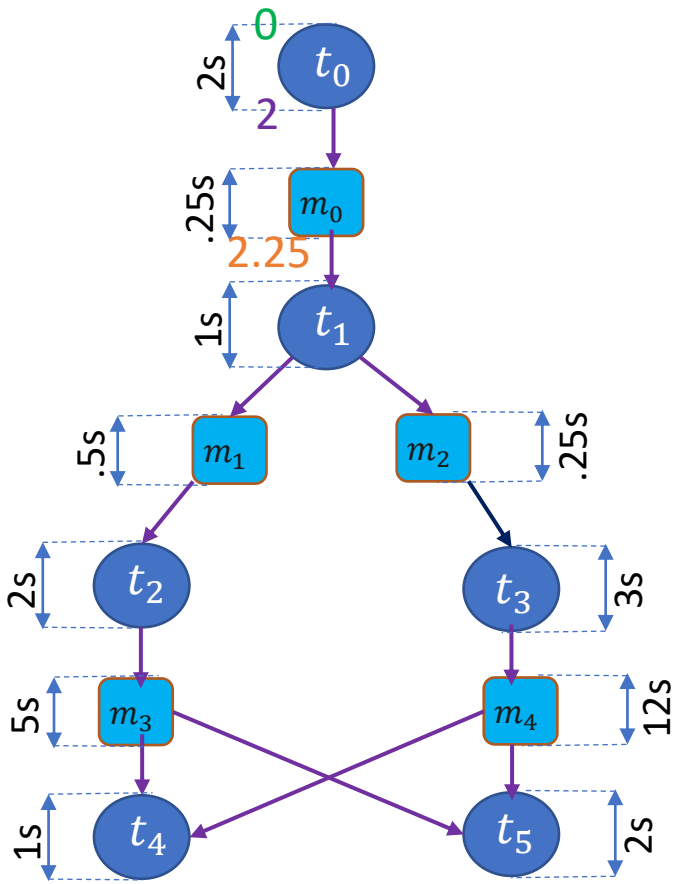


Makespan calculation:

Start time: The time that first task starts

Processing time: The time that takes to execute the task and set as **finish** time of the task

Makespan calculation



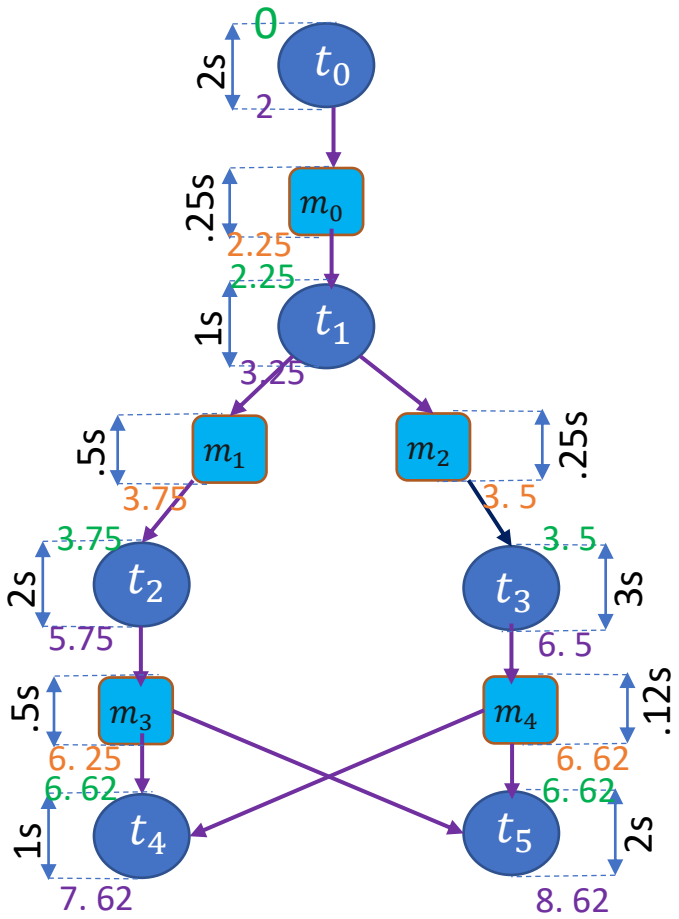
Makespan calculation:

Start time: The time that first task stats

Processing time: The time that takes to execute the task and set as **finish** time of the task

Transmission time: The time that takes to send message to the next task

Makespan calculation



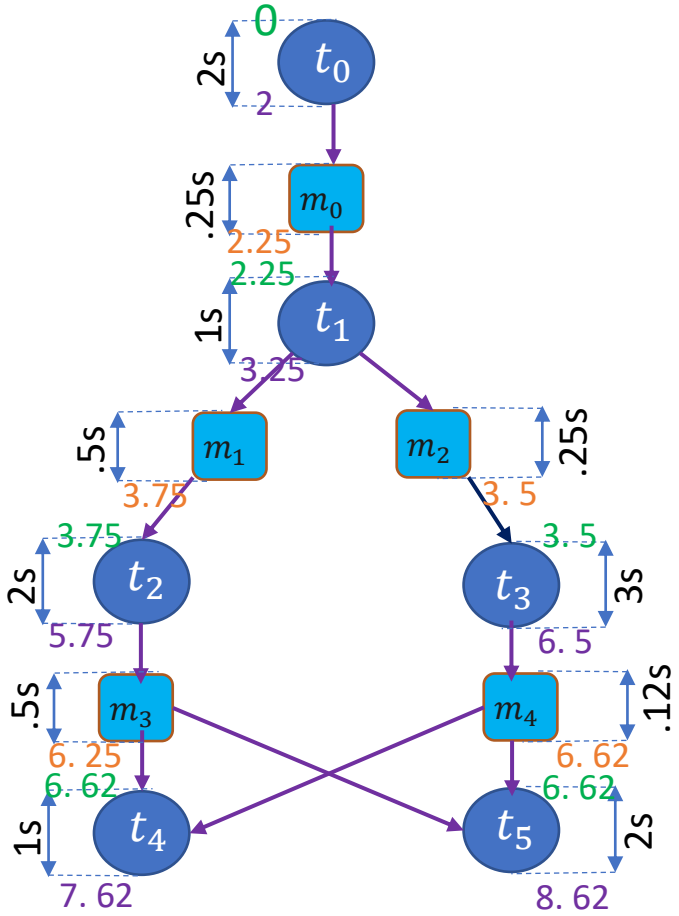
Makespan calculation:

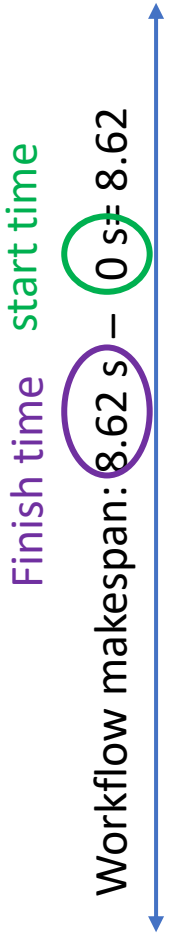
Start time: The time that first task stats

Processing time: The time that takes to execute the task and set as **finish** time the task

Transmission time: The time that takes to send message to the next task

Makespan calculation





Makespan calculation:

Start time: The time that first task stats

Processing time: The time that takes to execute the task and set as **finish** time the task

Transmission time: The time that takes to send message to the next task

Timing as design objective

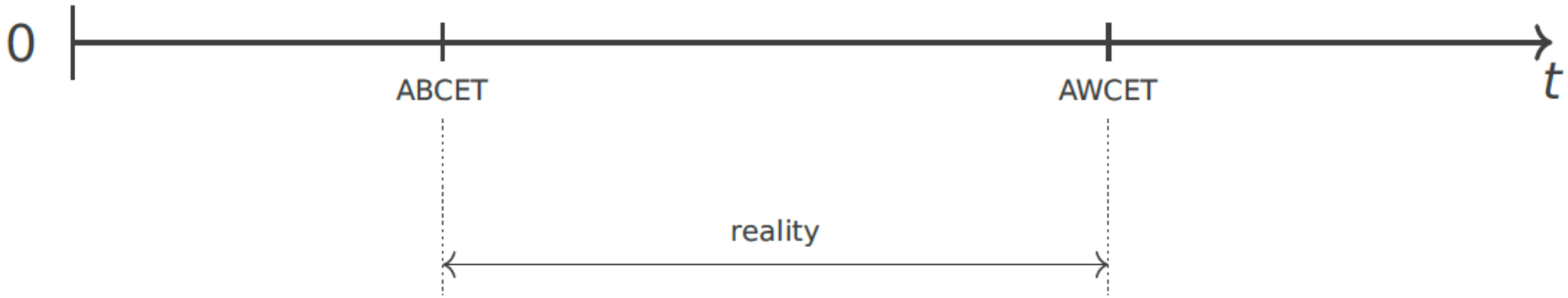
- The main sources of difficulties during the timing evaluation of realistic workflows are:
  - Significant size of realistic workflows
  - More complex graph topology constructs, e.g., conditional branches and loops
  - Parameter uncertainty due to the inherent dynamism of real-life systems
- The dynamism in the timing-relevant parameters is caused by:
  - Variations in the quantity of the input workflow.
  - Competition for resources with other applications.
  - Probabilistic behavior of the system model abstractions made during the system modeling.

Dynamism remedies

Depending on the use case and the design situation, parameter uncertainties are addressed by:

- **Formal Analysis:**
  - Formal calculation of a **safe** bound for the calculated parameter
  - Allows to **guarantee** (safety-critical) characteristics of the system
  - Comes with an over-/underestimation of the analyzed characteristic
- **System Approximation:** Usage of data from
  - Real executions
  - Historical executions
  - Simulated executions
- Create an **approximate** model of the parameter via
  - average values
  - probabilistic distributions
  - prediction models

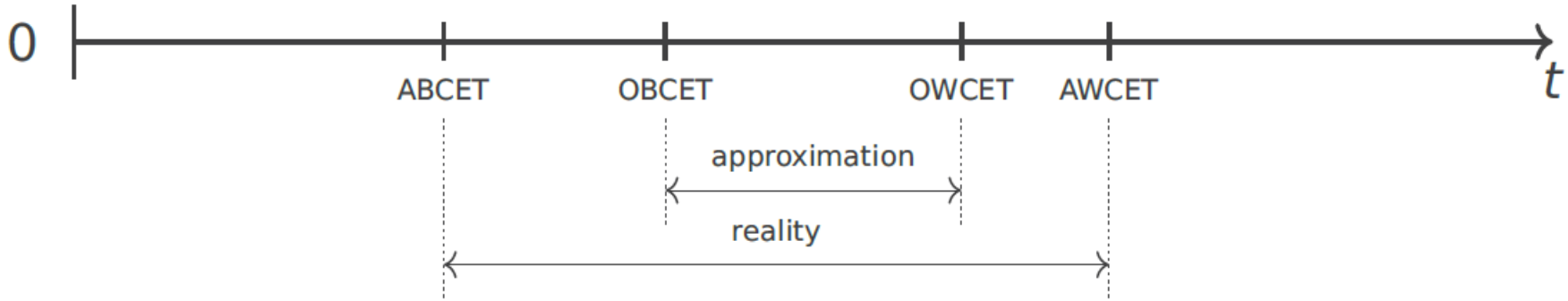
Dynamism remedies



ABCET: Actual best case execution time

AWCET: Actual worst case execution time

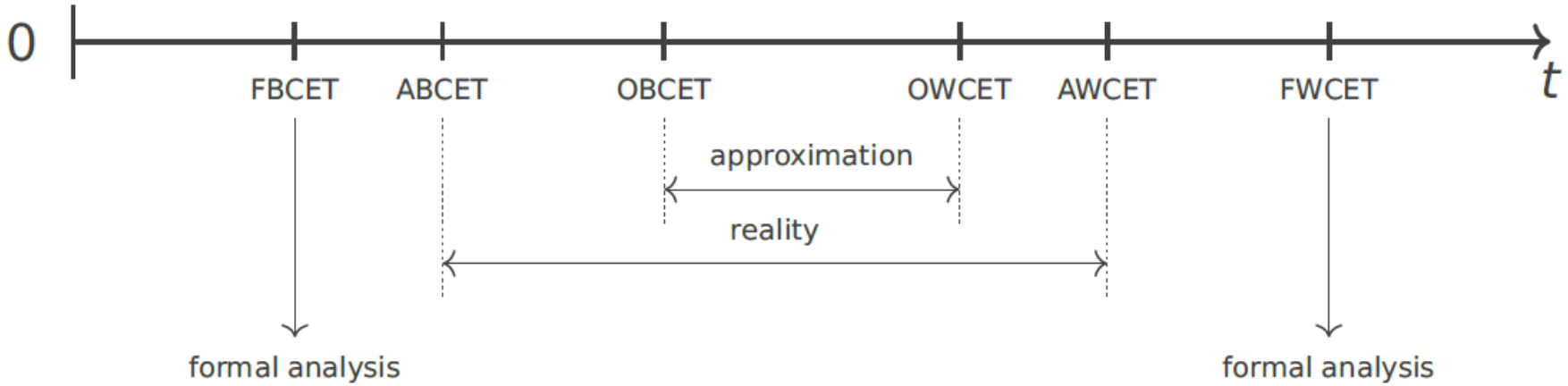
Dynamism remedies



OBCET: Observed best case execution time

OWCET: Observed worst case execution time

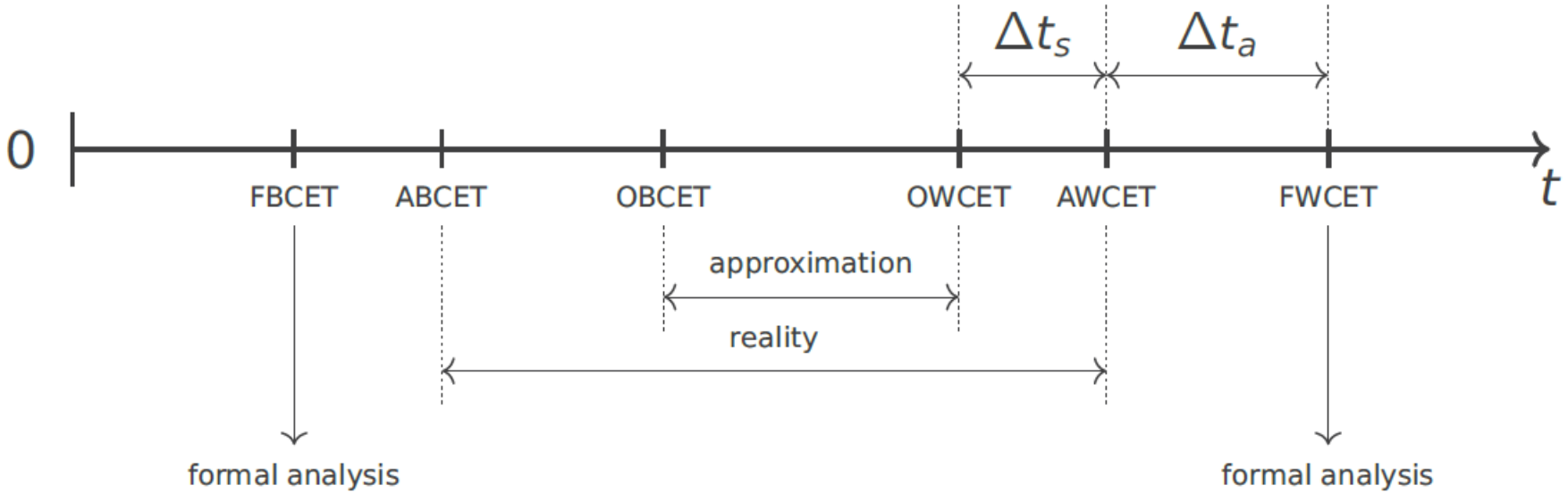
Dynamism remedies



FBCET: Formally calculated best case execution time

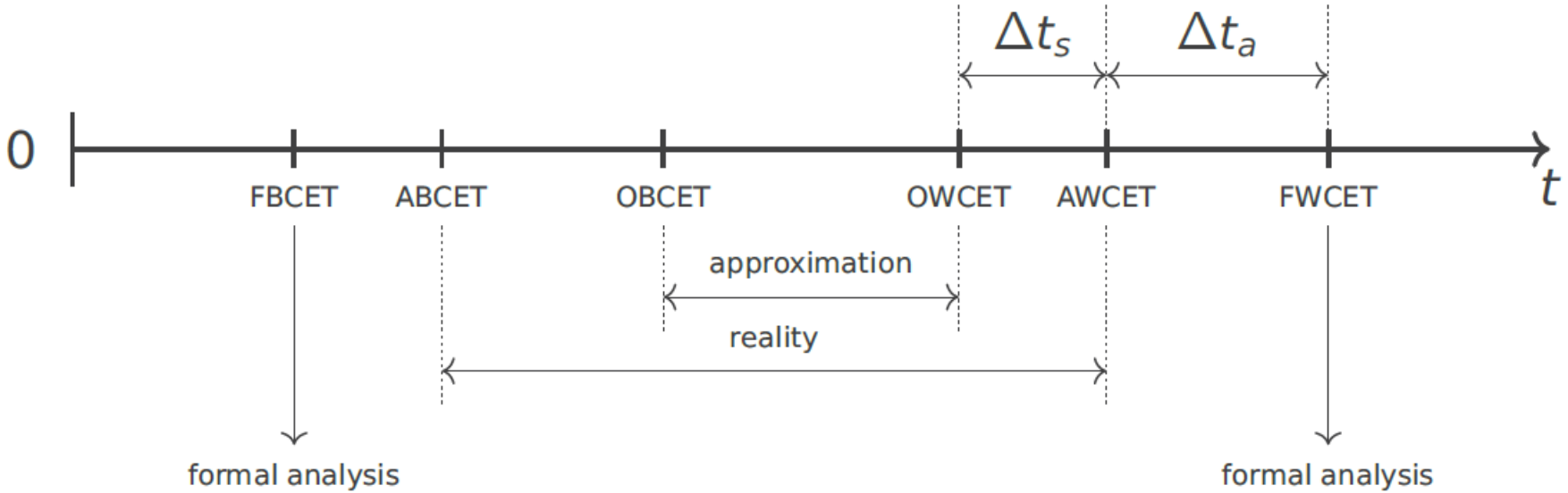
FWCET: Formally calculated worst case execution time

Dynamism remedies



- Since approximative approaches base their estimations on a finite number of experiments, they typically underestimate the actual worst case (interval Δt_s in the figure) and cannot be used to provide safety guarantees.

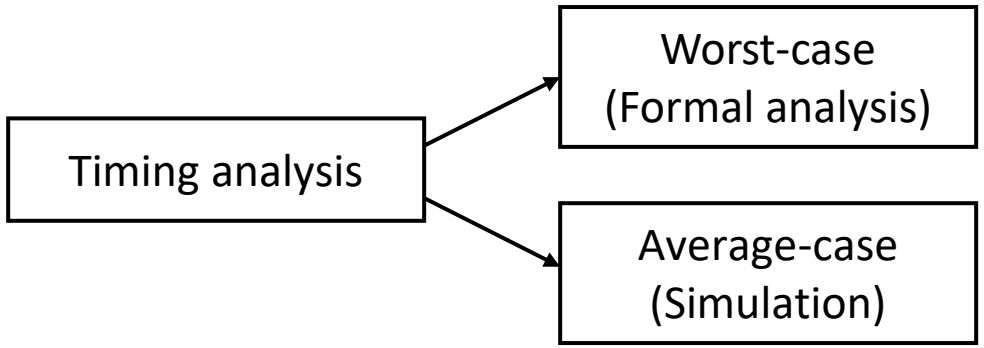
Dynamism remedies



- Even *tight* approaches for formal analysis, i.e., analysis approaches with a small overestimation of the actual worst case (interval Δt_a), address specific (and typically improbable) corner cases, so that they do not reflect the average-case behavior of the system.

Dynamism remedies

- When talking about timing values, there is a distinction between
  - the *best-case execution* time,...
  - the *average-case execution* time, and...
  - the *worst-case execution* time.
- From these three, the **worst-** and the **average-case** timing are the more relevant characteristics for scheduling decisions, since they are used to provide **real-time guarantees** and determine the **QoS** of the system, respectively.



Monetary cost as a design objective

Motivation

Benefit of Cost Minimization:

- Advantage for application users: end users gain the application benefit for a lower cost.
- Competitive for developers : Application developers can gain a competitive advantage by optimizing their applications, enhancing features, and maintaining high-quality services.
- Competitive for providers: By offering competitive pricing and cost-efficient services, providers can attract more customers and retain existing ones.

Challenges in the Cloud-Fog-Edge Continuum:

- The complex cost and business models.
- The inherent dynamism of the applications.
- The not yet fully mature business models, in particular for fog and edge applications.

Pricing models

Across the various providers and technologies, there is a large number of diverse pricing models. Nevertheless, there are certain prevailing trends:

- **Processing power:** The pricing is typically based on the amount of used processing power (given as the product of a processor-dependent cost rate and the **processing time** required by the application).
- **Leasing model:** Cloud providers offer various leasing models to accommodate different usage patterns and budgetary constraints.

Pricing models

Across the various providers and technologies, there is a large number of diverse pricing models. Nevertheless, there are certain prevailing trends:

- **Processing power:** The pricing is typically based on the amount of used processing power (given as the product of a processor-dependent cost rate and the **processing time** required by the application).
- **Leasing model:** Cloud providers offer various leasing models to accommodate different usage patterns and budgetary constraints.
  - Demand-based: users pay for resources on a pay-as-you-go basis
  - Fixed-time leasing: fixed-time leasing involves reserving resources for a specific duration

Pricing models

Across the various providers and technologies, there is a large number of diverse pricing models. Nevertheless, there are certain prevailing trends:

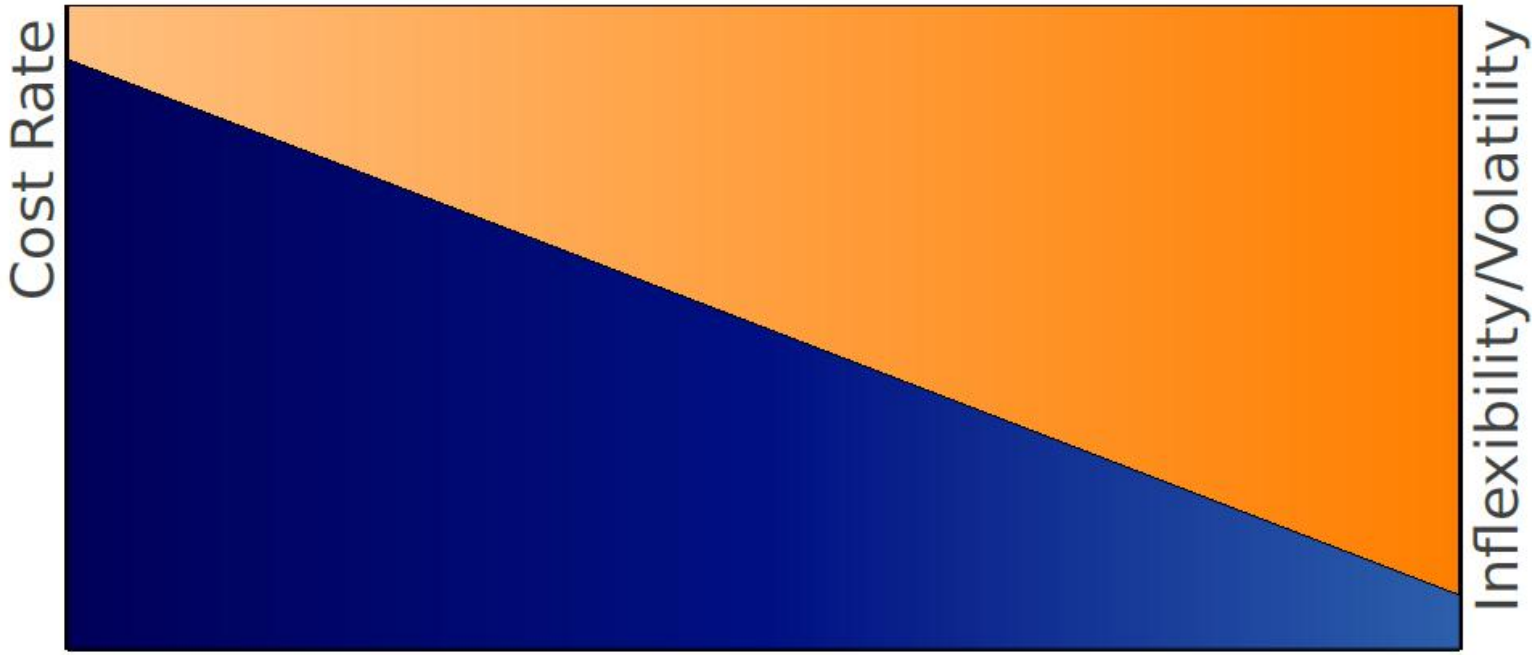
- **Processing power:** The pricing is typically based on the amount of used processing power (given as the product of a processor-dependent cost rate and the **processing time** required by the application).
- **Leasing model:** Cloud providers offer various leasing models to accommodate different usage patterns and budgetary constraints.
  - Demand-based: users pay for resources on a pay-as-you-go basis
  - Fixed-time leasing: fixed-time leasing involves reserving resources for a specific duration
- **Access Guarantees:** Another factor that influences pricing is availability and preemptability of resources:

Pricing models

Across the various providers and technologies, there is a large number of diverse pricing models. Nevertheless, there are certain prevailing trends:

- **Processing power:** The pricing is typically based on the amount of used processing power (given as the product of a processor-dependent cost rate and the **processing time** required by the application).
- **Leasing model:** Cloud providers offer various leasing models to accommodate different usage patterns and budgetary constraints.
  - Demand-based: users pay for resources on a pay-as-you-go basis
  - Fixed-time leasing: fixed-time leasing involves reserving resources for a specific duration
- **Access Guarantees:** Another factor that influences pricing is availability and preemptability of resources:
  - Unpreemptable Resources: some cloud instances come with guarantees of uninterrupted access, ensuring stable performance but potentially at a higher cost.
  - Preemptable Resources**:** it refers to spot instances (e.g., Amazon EC2 Spot Instances), come at a lower cost but they can be terminated by the provider when demand from on-demand users increases.

Pricing models



The (significantly) lower costs of less flexible or less reliable resources offer opportunities for cost savings by means of intelligent scheduling.

Pricing models

Several aspects of the cloud pricing model make the cost calculation more complex, compared to other domains:

- Discretized Time Intervals: Cloud resources are typically billed in discretized time intervals (e.g., a resource used for 125 seconds is billed for 3 minutes).
- Variable Network Costs: The costs of network usage (data transmissions) can significantly vary based on the concrete use case (e.g., the necessity of cross-provider data transmissions).
- Super-Linear Performance-to-Price Ratio: More powerful resources can offer a super-linear performance-to-price ratio, since they reduce the response time of the tasks.
- Emerging Energy Pricing Considerations: While energy pricing does not yet play a large role, it may become more important as fog and edge resources become more widespread.

These aspects render the calculation of costs **illinear** and make optimization significantly more challenging.

Pricing models in Fog and Edge

Compared to cloud resources, the pricing in the Fog and Edge is:

- Free or low cost usage:
  - Immaturity of Deployment: The primary reason for the often-free or low-cost nature of fog and edge resource usage is the early stage of deployment and development in these domains.
  - Volatility: The fog and edge landscape is characterized by rapid changes which makes it challenging to establish stable pricing models
- Expected Billing Models:
  - Consumption-Based Billing: This model charges users based on the actual resources consumption.
  - Flat Rate Pricing: Some providers may opt for flat rate pricing models, where users pay a fixed fee for a predetermined allocation of fog and edge resources.
- Focus on Energy Consumption:
  - Energy Efficiency Emphasis: Future pricing models are expected to incorporate energy consumption considerations, incentivizing users to optimize their applications for efficiency.

Energy consumption as a design objective

Motivation

For a long time, the energy consumption of applications in the cloud-edge continuum was viewed solely as a cost factor. More recently, it is becoming more and more important due to...

- Rapid spread of cloud technology: With more applications and services migrating to the cloud, data centers have multiplied in number and size. As a result, energy consumption within data centers has surged.
- Increasing environmental awareness: growing global emphasis on environmental sustainability and reducing carbon footprints.
- Emergence of mobile edge devices: These devices often operate on battery power and have limited energy resources.

From the orchestration perspective, the energy consumption (just like the timing or the monetary costs) is an additional design objective which is influenced mainly by the choice of the resources and the task-to-resource assignment.

Energy consumption calculation

The energy consumption W of a resource is typically calculated as a product of the **processing time** t and its power consumption P:

$$W = P.t$$

Hereby the power consumption depends mainly on the utilization U of the resource, its idle consumption P_{min}, and its maximal consumption P_{max} :

$$P = (P_{max} - P_{min}).U + P_{min}$$

Energy consumption as a design objective

During the scheduling of applications in the cloud-edge continuum, energy consumption is a design objective which...

- is becoming increasingly important.
- is correlated with the processing time of the application.
- can be difficult to calculate due to a lack of information about the resources.

Reliability as a design objective

Reliability

Two informal descriptions of reliability:

- "Reliability can be defined as the probability that a system will produce **correct outputs** up to **some given time**." – Computer Science Handbook by McClusky and Mitra
- "Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs **according to its specifications**." – Article at WhatIs.com (First google result when searching for "Reliability definition") by Margaret Rouse

In contrast to "self-contained" system qualities like application makespan or the energy consumption of the used resources, the reliability of a system is defined w.r.t. a certain ***failure model*** (also referred to as *fault model* or *error model*).

Reliability as a design objective

- The evaluation of the reliability of a system:
  - requires the definition of a *failure model*.
  - can be automated using the *structure function* of the system.
- The reliability of a system can be improved by means of *redundancy*.
- Redundancy mechanisms...
  - affect other design objectives, e.g., monetary cost or energy consumption.
  - can be either static or dynamic in nature.
  - may introduce the need for additional resources.

Reliability

In the context of reliability, an important distinction is made between 3 (relatively similar) terms:

- **Fault:** "Abnormal condition (or defect) that can cause an element or an item to fail." – Cause of the error
- **Error:** "Difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.“ – (Not necessarily perceptible) erroneous state of the system
- **Failure:** "Termination of the ability of an element or an item to perform a function as required." – The perceptible failure of the system

* Definitions taken from the *automotive.wiki* website

Motivation

Current state in the cloud-edge domain: Applications are mostly...

- executed in the cloud.
- are not safety-critical and have relatively long, soft deadlines.

Consequently...

- the occurring faults are mostly *transient*.
- failures of the system are associated with longer waiting times, i.e., a lower QoS.
- reliability is treated as a non-functional requirement, correlated with financial costs.

Motivation

Trends for the near future: Applications are increasingly...

- executed on fog and edge resources.
- are safety-critical and have hard deadlines.

Consequently...

- faults become more important.
- the consequences of system failures become more severe (property damage or even loss of human life).
- reliability is increasingly considered a functional requirement.

the reliability analysis in the cloud-edge domain will develop along a similar path as analyses in the classic safety-critical domains such as automotive or avionics.

Security as a design objective

Motivation

With the increasing spread of cloud-edge applications, security aspects are becoming more and more important, both for the industry and for consumers.

Companies in industry...

- appreciate the access to large amounts of processing power and data storage space...
- are anxious about the possibility of industrial spying and cyber attacks.

Consumers...

- profit from a combination of seamlessly available computing power with a high level of personalization...
- are anxious about a possible abuse of their personal data.

Orchestration perspective

In the context of scheduling, the extent to which security can be considered strongly depends on the information available concerning the security characteristics of...

- the tasks: Orchestrators need to match tasks with appropriate resources that meet these security requirements.
- the resources: Orchestrators should assess whether selected resources align with the security needs of the tasks they are orchestrating.
- the transmitted data: Orchestrators should ensure that data remains secure throughout its lifecycle, from processing to transmission and storage.
- the communication network: Orchestrators should route communication through secure channels to prevent data breaches or interception.

Orchestration rules

Security rules for cloud-edge applications can, e.g., look as follows:

- Depending on their security relevance (e.g., the degree to which they process personal information of the users), tasks are labeled as...
  - private
  - semi-private
  - public
- During the orchestration...
  - private tasks shall only be processed on resources owned by the user.
  - semi-private tasks shall be processed on resources owned by the user and on nearby edge/fog resources.
  - public tasks can be executed on any resource.

These rules can then be considered either by adjusting the system specification or as constraints for the optimizer.

Orchestration rules

The orchestration may also involve decisions whether or not to use security-specific services such as the en- and the decryption of messages. These services...

- are integrated into the application as additional tasks which...
- have to be mapped onto the appropriate resources...
- and impact other design objectives, e.g., the monetary costs or the energy consumption.

universität innsbruck

Thank you for your attention☺

Dr. Zahra Najafabadi Samani

Email: Zahra.Najafabadi-Samani@uibk.ac.at