

Cloud, Fog and Mist Computing Collaboration for Scheduling Time-Critical Applications

State of the Art and Research Challenges

Helen D. Karatza

Department of Informatics

Aristotle University of Thessaloniki

Greece

Telecom 2025

Sofia, Bulgaria

November 20-21, 2025

Scope

The scope of this talk is to present:

- state-of-the-art research covering a variety of concepts in cloud, fog and mist computing from the performance perspective,
- **resource management issues** that must be addressed in order to make clouds viable for HPC,
- **efficient scheduling techniques** for complex **time-critical** applications in cloud, fog and mist computing environments.
- to provide future trends and directions in the cloud computing area.

Presentation Structure

- Cloud Issues
- Resource Management and Scheduling in Clouds
- Complex Workloads – Real-Time Applications
- Fog and Mist Computing
- Conclusions and Future Directions

Cloud Issues – Big Data (1/2)

- The overwhelming flow of data of huge volume generated by a wide spectrum of sources, such as:
 - sensors,
 - mobile devices,
 - social media, and
 - the Internet of Things,

has led to the emergence of trends such as **big data** and **big data analytics**.

Cloud Issues – Big Data (2/2)

- **Computationally intensive applications** are employed in many domains such as science, engineering, enterprises, finance, healthcare, etc., in order **to exploit the power of big data**.
- **Big data analytics** employ computationally intensive algorithms in order to process big data and produce **meaningful results in a timely manner**.
- Consequently, applications operating on big data can be considered **real-time** with **firm deadlines**, since failing to meet their time constraints would make their results useless.

Resource Allocation and Scheduling (1/3)

Cloud computing offers cost-effective services to demanding applications.

However, cloud energy consumption is a concern with an impact to environment.

Resource allocation and scheduling are important issues **towards the direction for sustainable cloud.**

Scheduling manages:

- the **selection** of resources for a job,
- the **allocation** of jobs to resources and
- the **monitoring** of jobs execution.

Resource Allocation and Scheduling (2/3)

- Composite jobs may have end-to-end deadlines (***Real-Time Scheduling***).
- Software failures may occur during the execution of a composite job (***Fault-Tolerant Scheduling***).

Resource Allocation and Scheduling (3/3)

- A job may consist of **independent tasks** which can be processed in parallel
(***Bag-of-tasks Scheduling***).
- A job may consist of **frequently communicating tasks** which must be processed in parallel
(***Gang Scheduling***).
- A job may be decomposed into a collection of **tasks with precedence constraints** among them. These tasks may be scheduled on different nodes of the system
(***DAG or Workflow Scheduling***).

Real-Time Scheduling (1/6)

- Clouds are often used to run **time-critical (real-time) applications**.
- In ***real-time systems*** the correctness of the system does not depend only on the logical results of the computations, but also on the time at which the results are produced.
- Such systems are used for the control of nuclear power plants, financial markets, radar applications, wireless communications, transport systems, traffic control, etc. **contributing to the goal towards sustainable cities.**
- The jobs in a real-time system have ***deadlines*** which must be met. If a real-time job cannot meet its deadline, then its results will be useless, or even worse, catastrophic for the system and the environment that is under control.

Real-Time Scheduling (2/6)

▪ Real-time Jobs

Typical parameters that characterize a task of an application submitted for execution in a large-scale distributed system

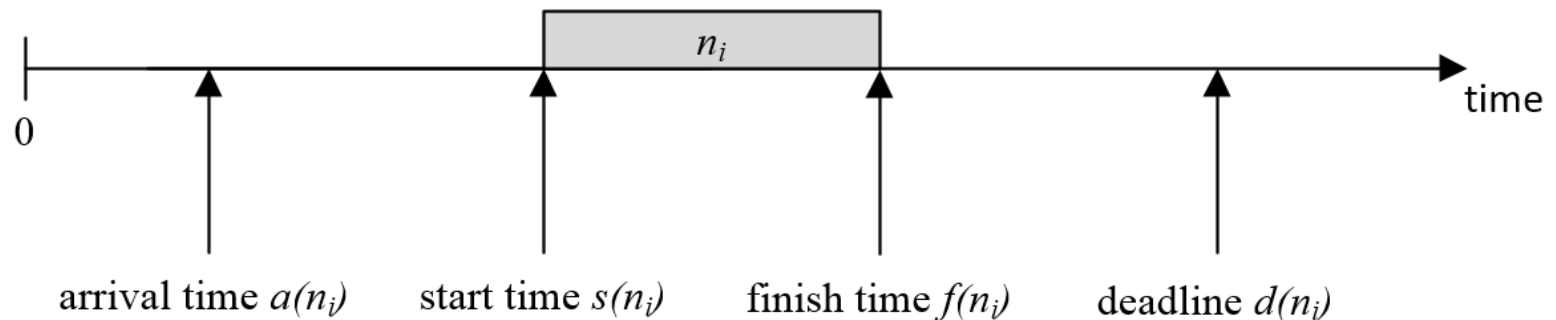


Fig. 1. An aperiodic job

Real-Time Scheduling (3/6)

▪ Real-time Periodic Jobs

A periodic job J_i is characterized by (P_i, C_i) , where P_i is the period of job J_i and C_i is the execution time of J_i . The deadline of the job is D_i , where $D_i \leq P_i$.

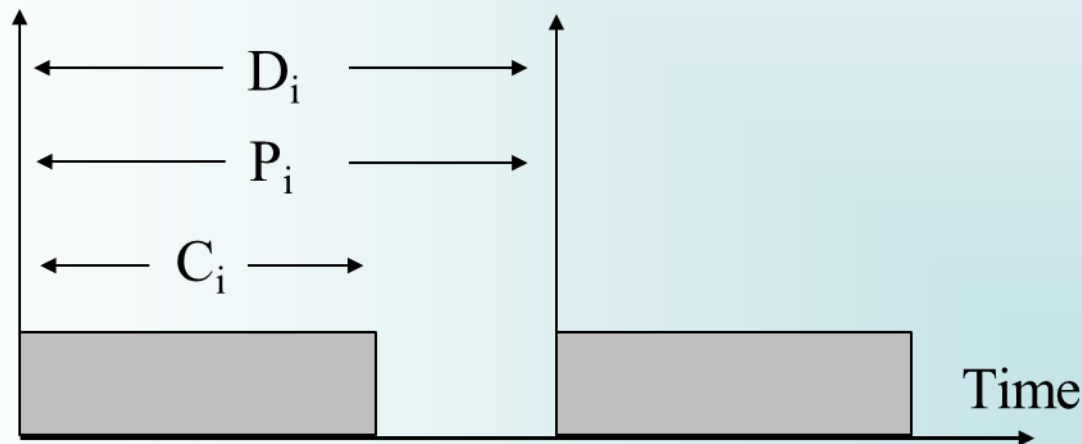


Fig. 2. A periodic job, $D_i = P_i$.

Real-Time Scheduling (4/6)

- In real-time systems it is often more desirable for a job to produce an **approximate result by its deadline**, than to produce an **exact result late**.
- ***Imprecise (Approximate) Computations*** can achieve that. It is a technique according to which the execution of a real-time job is allowed to return ***intermediate*** (imprecise) results of **poorer**, but still ***acceptable quality***, when the deadline of the job cannot be met.

Real-Time Scheduling (5/6)

- It is assumed that every job is ***monotone***, that is the accuracy of its intermediate results is increased as more time is spent to produce them.
- If the execution of a monotone job is fully completed, then the results are precise.
- Typically, a monotone job consists of a ***mandatory part MP***, followed by an ***optional part OP***.
- In order for a job to be completed, it must complete at least its mandatory part before its deadline.

Real-Time Scheduling (6/6)

The **notification time NT** of a job is the difference between the absolute deadline of the job and the job's mandatory part ($NT = D - MP$).

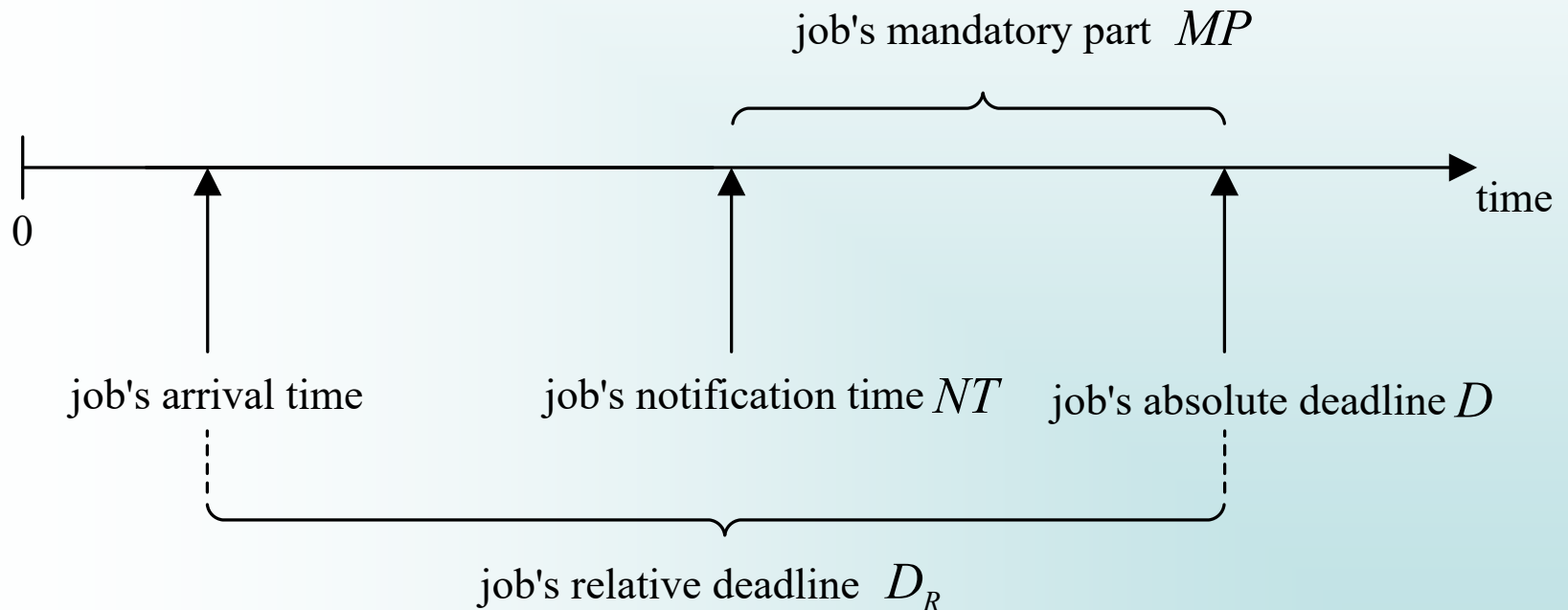


Fig. 3. A job's associated times in the Imprecise Computations case

Fault-Tolerant Scheduling (1/3)

- Fault tolerance is an important issue in Cloud Computing.
- Real-time systems in particular, need to tolerate possible ***software faults*** that may cause failures during the execution of a job.
- **Imprecise computations** combined with **checkpointing** can provide fault-tolerance in large-scale distributed real-time systems such as clouds.
- This is achieved with ***application-directed checkpoints***:
 - each job is responsible for checkpointing its own progress periodically (by saving its intermediate results) at regular intervals during its execution, so that a checkpoint takes place when the job completes its mandatory part.

Fault-Tolerant Scheduling (2/3)

Example: Checkpoints occur when the 20%, 40%, 60% and 80% of the job's service time is completed. The mandatory part of the job constitutes the 60% of the job's service time. The third checkpoint takes place when the mandatory part of the job is completed.

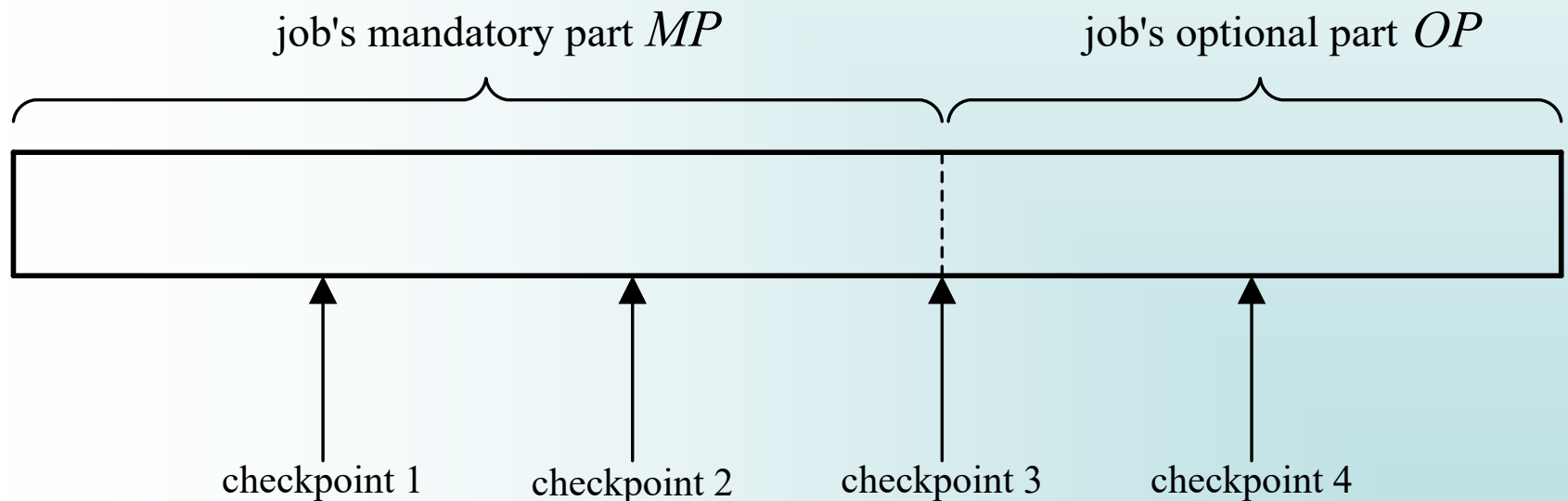


Fig. 4. Checkpoints

Fault-Tolerant Scheduling (3/3)

G.L. Stavrinides, H.D. Karatza, “Scheduling real-time parallel applications in SaaS clouds in the presence of transient software failures”, in Proceedings of the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'16), Montreal, Canada, Jul. 2016.

- **Application-directed Checkpointing and Approximate Computations**
- **Objectives:**
 - (a) provide resilience against temporary software failures,
 - (b) guarantee that all applications will meet their deadline,
 - (c) provide application results of high quality,
 - (d) minimize the monetary cost charged to the end-users.

Bag of Tasks Scheduling (BoT) (1/5)

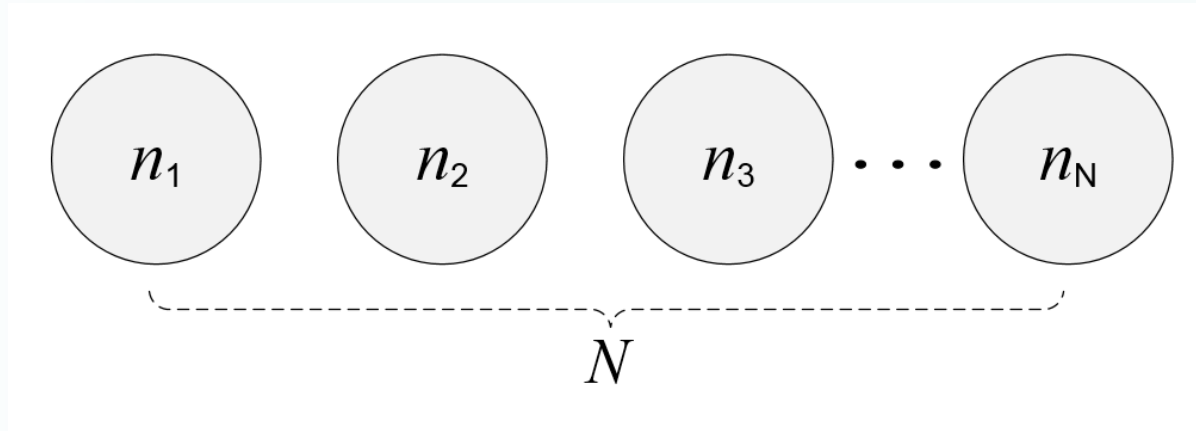


Fig. 5. A BoT consisting of N parallel independent tasks.

- A BoT is a job which consists of simple independent tasks which arrive to system at the same time.
- Execution of a BoT is completed when all of the tasks which belong to the same job are executed.

Bag of Tasks Scheduling (BoT) (2/5)

- **G. L. Stavrinides and H. D. Karatza**, “Performance evaluation of a SaaS cloud under different levels of workload computational demand variability and tardiness bounds”, Simulation Modelling Practice and Theory, Elsevier, Vol. 91, pp. 1-12, Febr. 2019.
- The performance of a SaaS cloud under various tardiness bounds and different levels of workload computational demand variability is evaluated.
- The workload consists of bag-of-tasks jobs which have soft deadlines and different levels of variability in their computational demands.

Bag of Tasks Scheduling (BoT) (3/5)

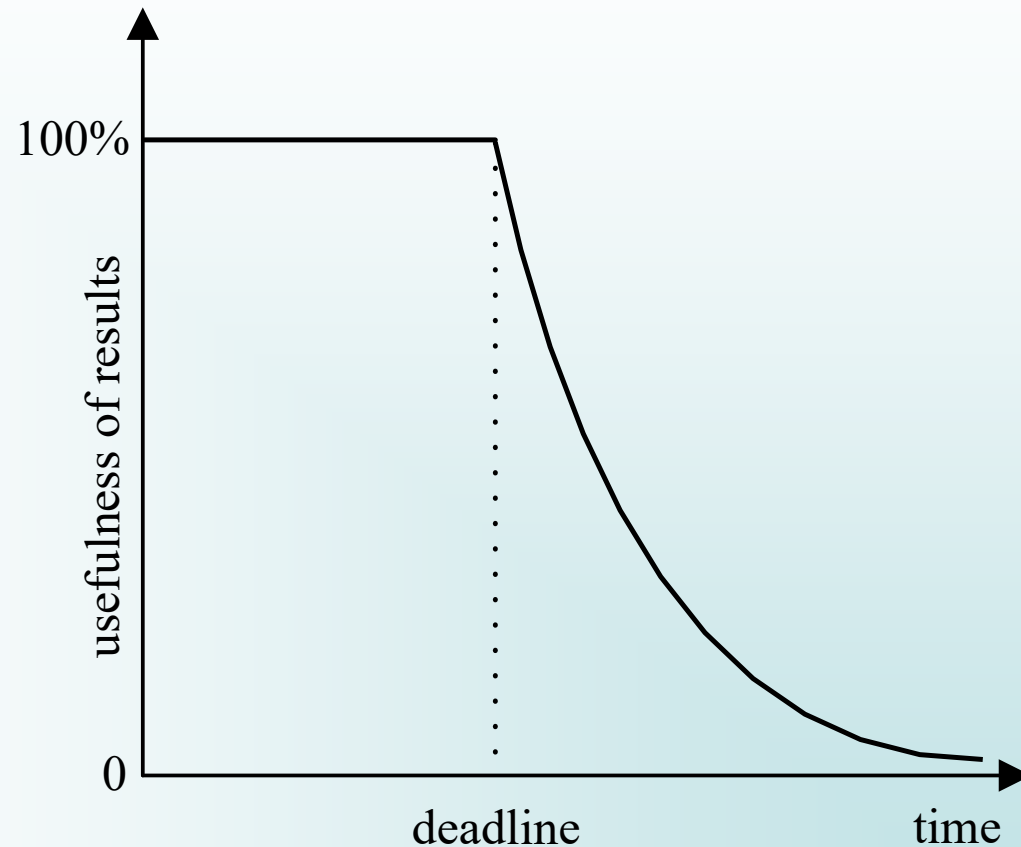


Fig. 6. The usefulness of the results of a job with a soft deadline over time.

Bag of Tasks Scheduling (BoT) (4/5)

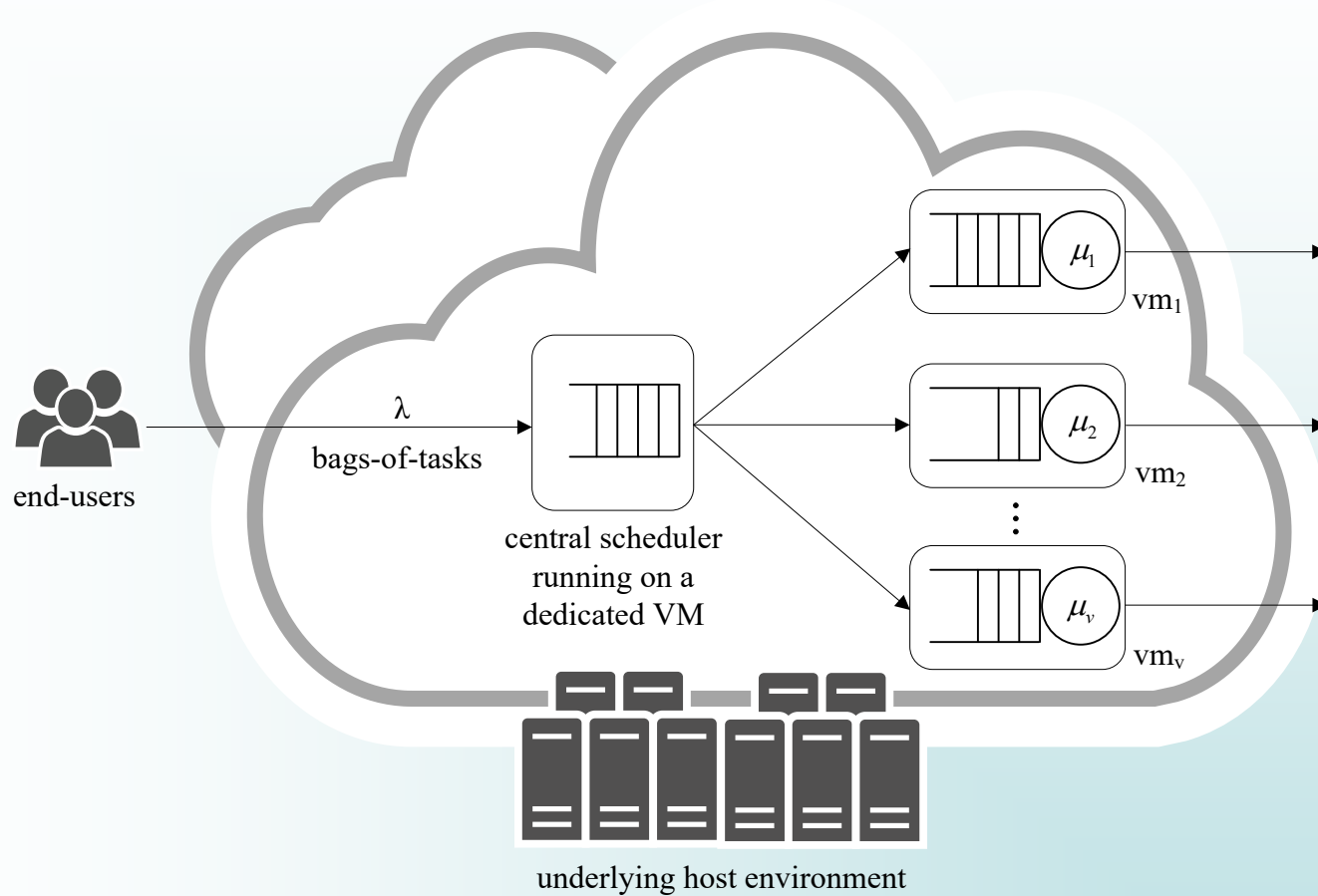


Fig. 7. The queueing model of the SaaS cloud.

Bag of Tasks Scheduling (BoT) (5/5)

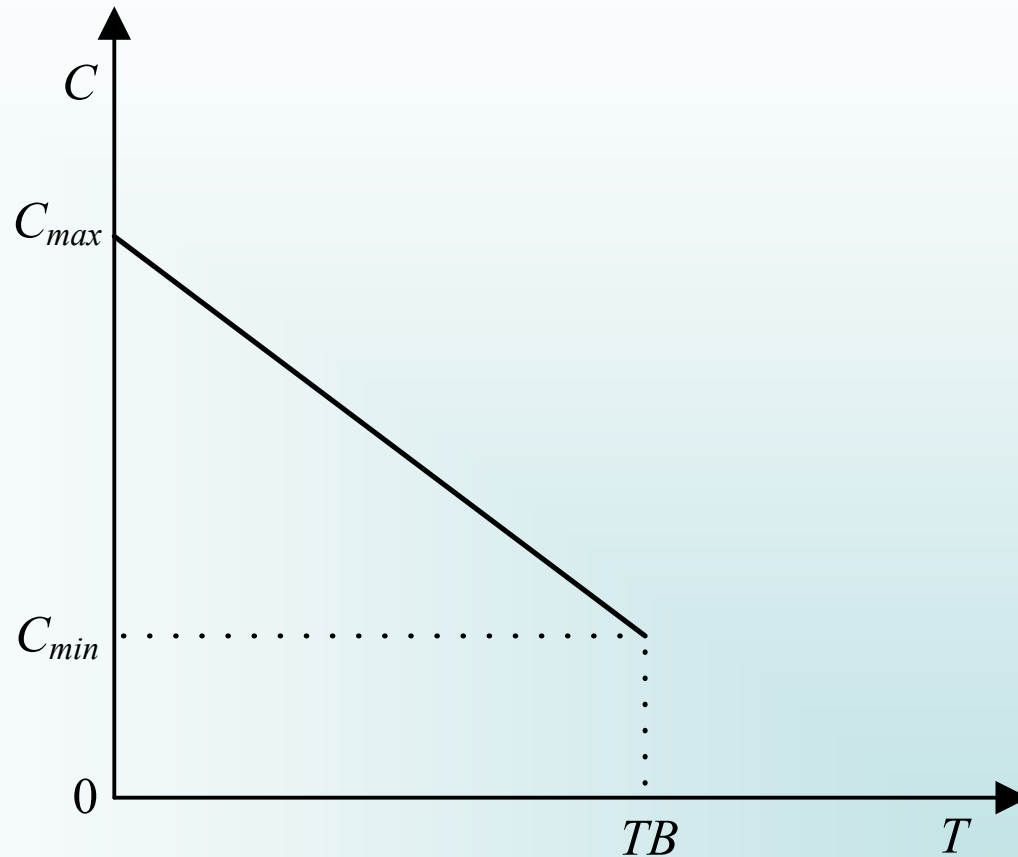


Fig. 8. The monetary cost per time unit charged for the execution of each job according to the provided level of QoS, as defined in the multi-tier SLA.

Gang Scheduling (1/4)

- In distributed systems jobs often consist of frequently communicating tasks which can be processed in parallel.
- An efficient way to schedule this kind of jobs is ***Gang Scheduling***, which is a combination of time and space sharing.
- According to this technique, a job is decomposed into tasks that are grouped together into a gang and scheduled and executed simultaneously on different processors.

Gang Scheduling (2/4)

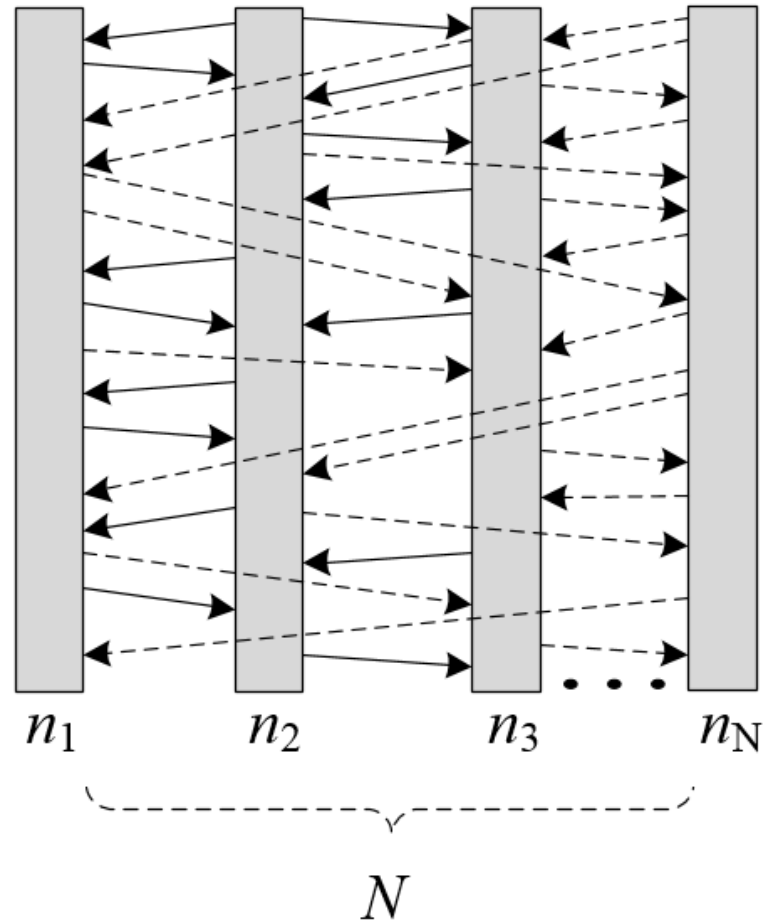


Fig. 9. A gang with N parallel frequently communicating tasks.

Gang Scheduling (3/4)

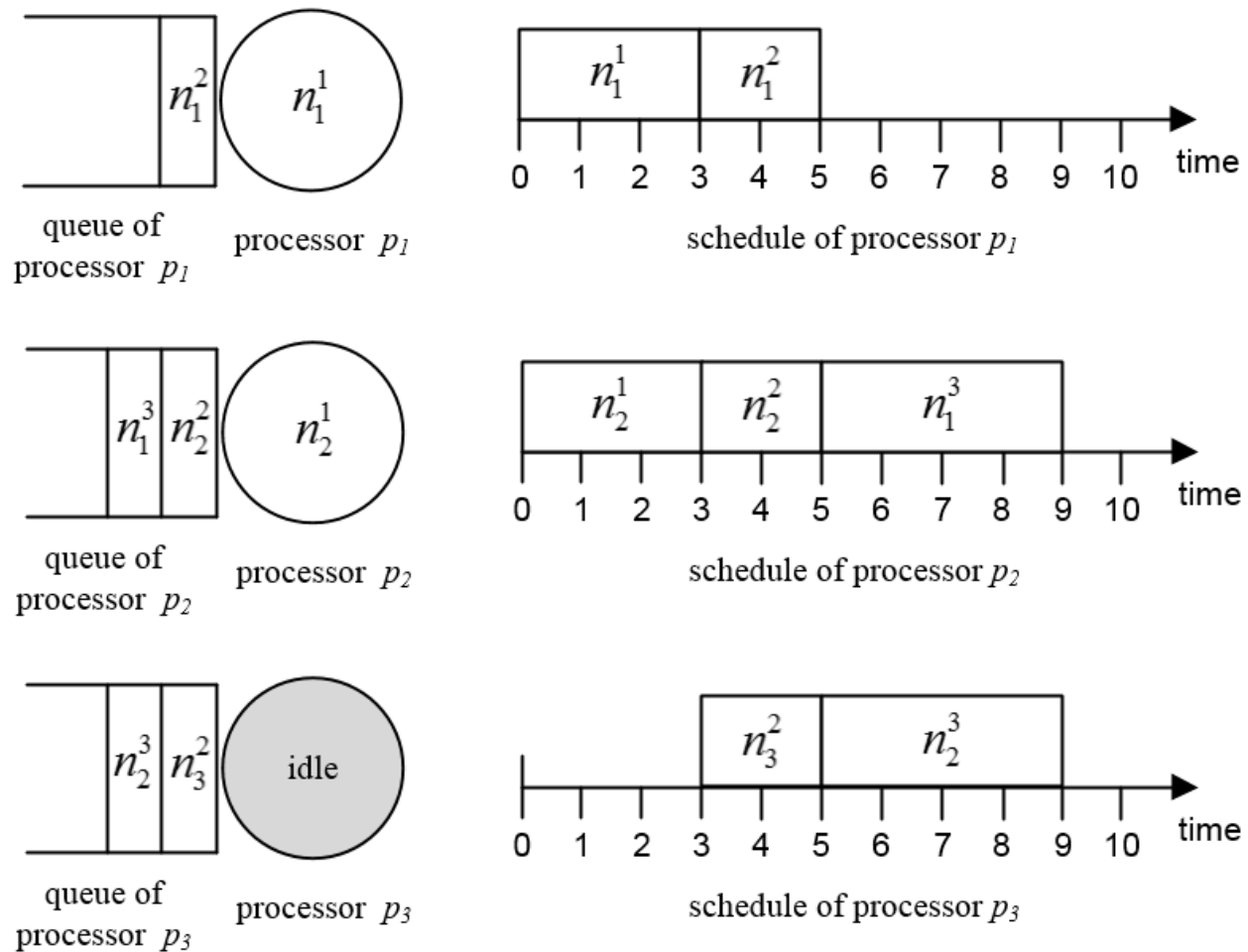


Fig. 10. Example of gang scheduling.

Gang Scheduling (4/4)

- **G. L. Stavrinides and H. D. Karatza**, “The impact of checkpointing interval selection on the scheduling performance of real-time fine-grained parallel applications in SaaS clouds under various failure probabilities”, Concurrency and Computation: Practice and Experience, Wiley, 30(12), 2018.

The **impact of checkpointing interval selection** on the performance of a SaaS cloud is studied, where

- fine-grained parallel applications with firm deadlines, and approximate computations are scheduled for execution, under various failure probabilities.

The **relation between the checkpointing interval and failure probability** is studied and analyzed.

DAG (Workflow) Scheduling (1/3)

A different workload model, is the following:

- A job may be decomposed into a collection of tasks with ***precedence constraints*** among them, so that a task's output may be used as input by other tasks of the job.
- That is, a job is a ***Directed Acyclic Graph (DAG) or Workflow***.
- In order for a task to start execution, all of its predecessor tasks must have been completed.
- **G.L. Stavrinides, H.D. Karatza**, “A cost-effective and QoS-aware approach to scheduling real-time workflow applications in PaaS and SaaS clouds”, In Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud'15), Rome, Italy, Aug. 2015, IEEE, pp. 231-239.

DAG (Workflow) Scheduling (2/3)

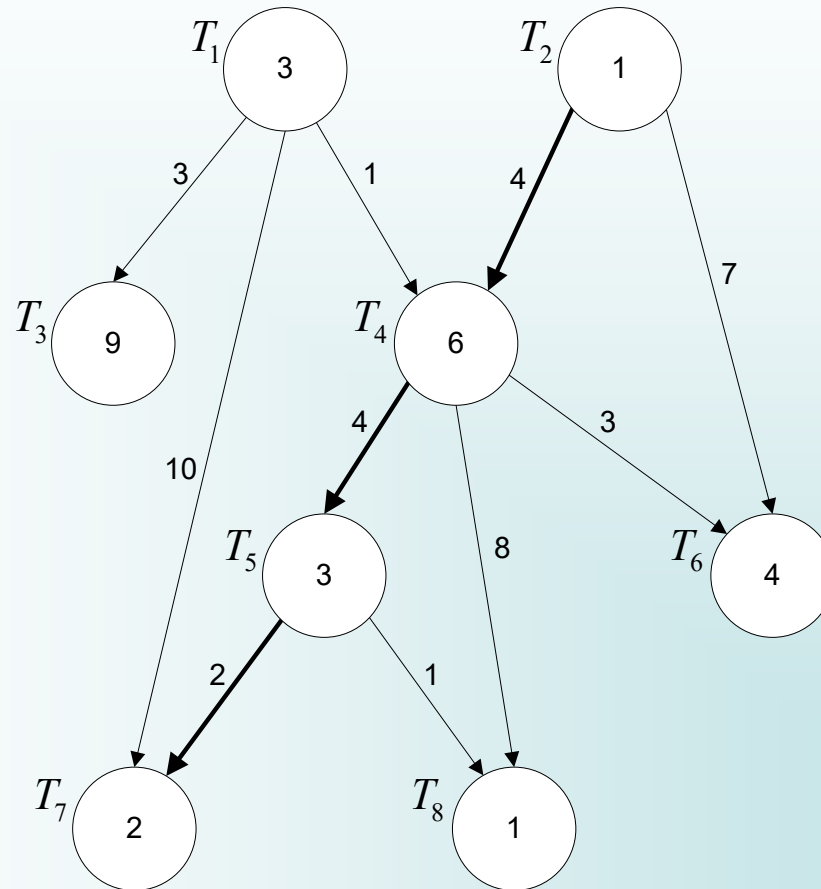


Fig. 11. A Directed Acyclic Graph (DAG)

DAG (Workflow) Scheduling (3/3)

- One of the main characteristics of cloud environments is the elasticity of the virtualized resources, which allows them to be dynamically scaled in or out.
- This research investigates the effect of laxity of real-time workflow jobs on the performance of elastic cloud resources.
- It evaluates through simulation the impact of laxity on the deadline miss ratio, energy consumption and monetary cost associated with the resources, considering workloads with different slack times.

G. L. Stavrinides and H. D. Karatza, “The Effect of Laxity of Real-Time Workflow Applications on the Performance of Elastic Cloud Resources”, in Proceedings of the 9th International Conference on Future Internet of Things and Cloud (FiCloud'22), Rome, Italy, 22-24 Aug., 2022.

Linear Workflow Scheduling (1/3)

H. D. Karatza and G. L. Stavrinides,
“Resource allocation and aging priority-based scheduling of linear workflow applications with transient failures and selective imprecise computations”, Cluster Computing, Springer, vol. 27, Issue 4, pp. 5473–5488, Jul. 2024.

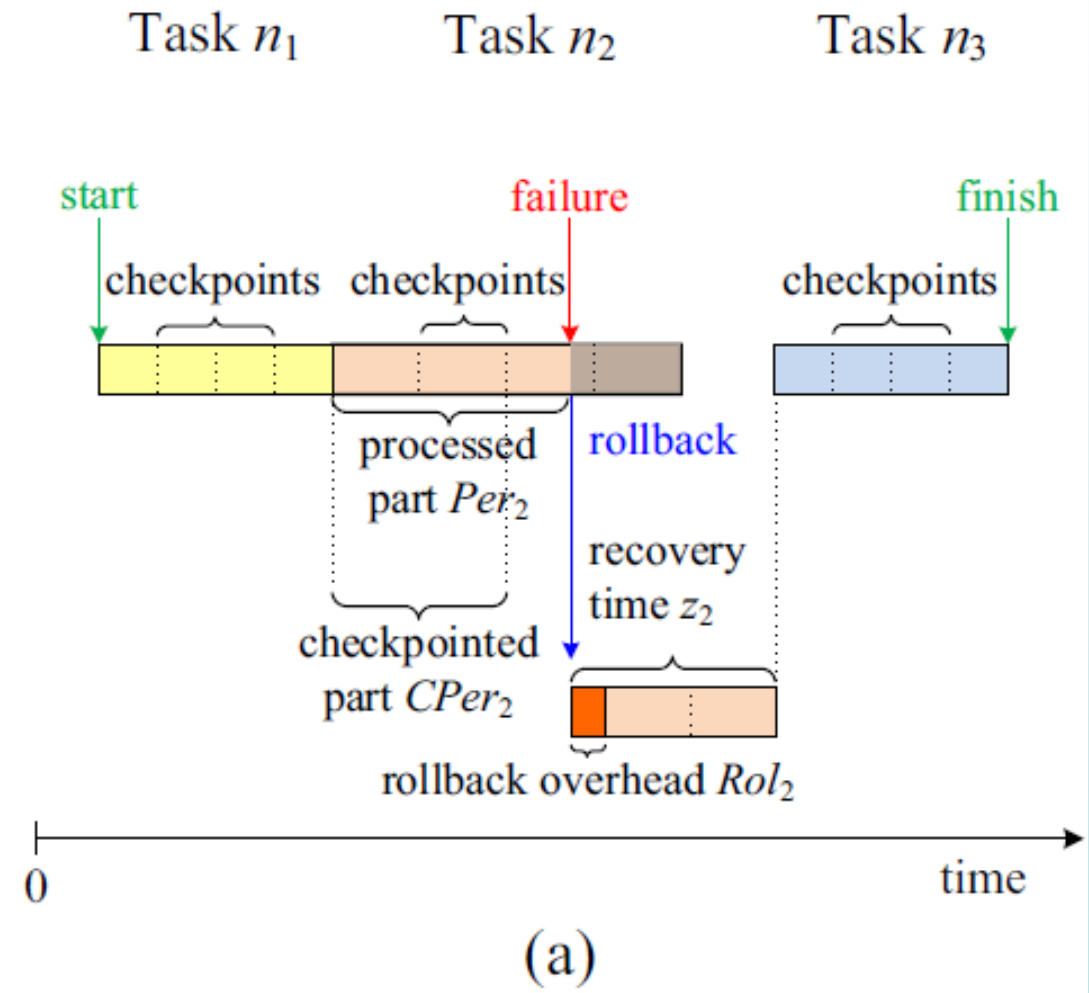


Fig. 12. A Linear Workflow – a failure occurs after the second checkpoint of a non-exit task (n_2).

Linear Workflow Scheduling (2/3)

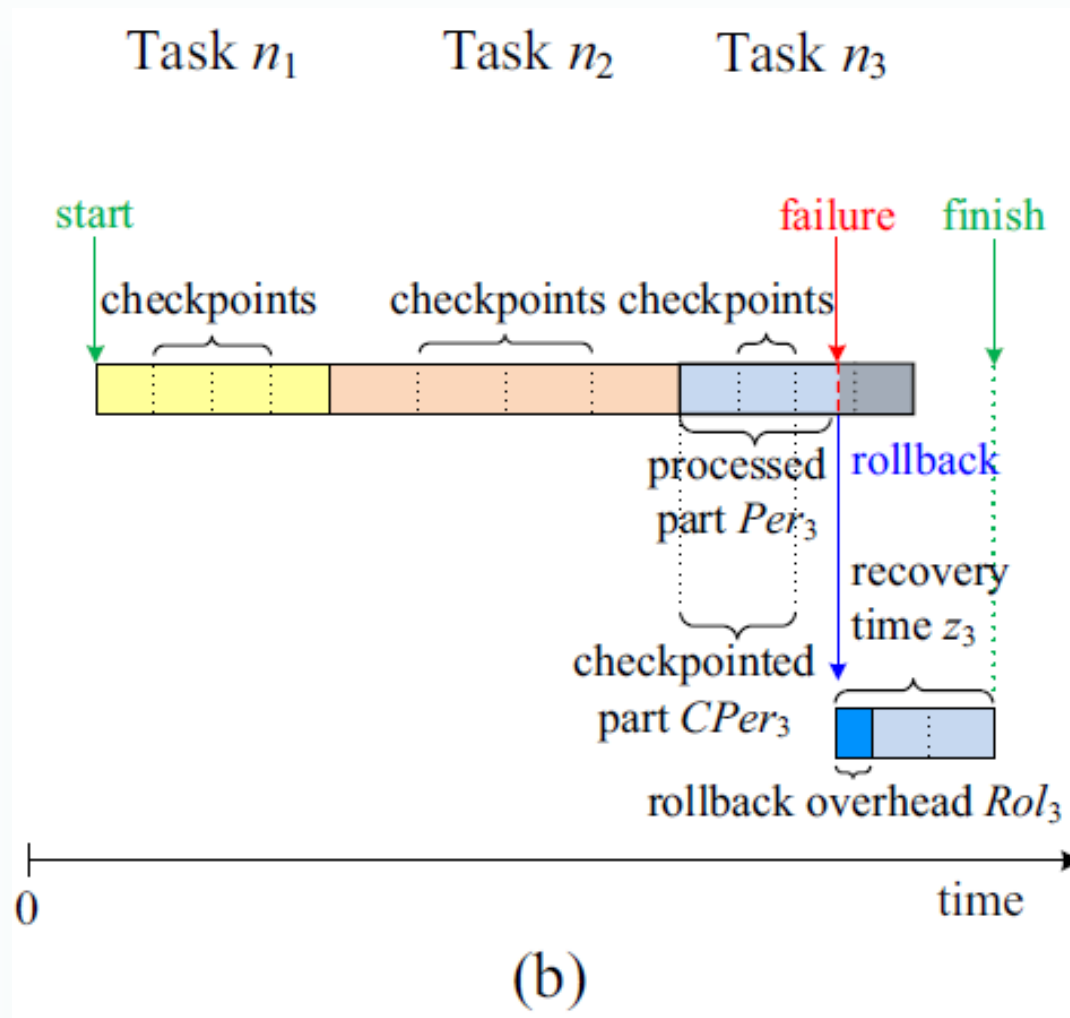


Fig. 13. A Linear Workflow - a failure occurs after the second checkpoint of the exit task (n_3) – no imprecise computations.

Linear Workflow Scheduling (3/3)

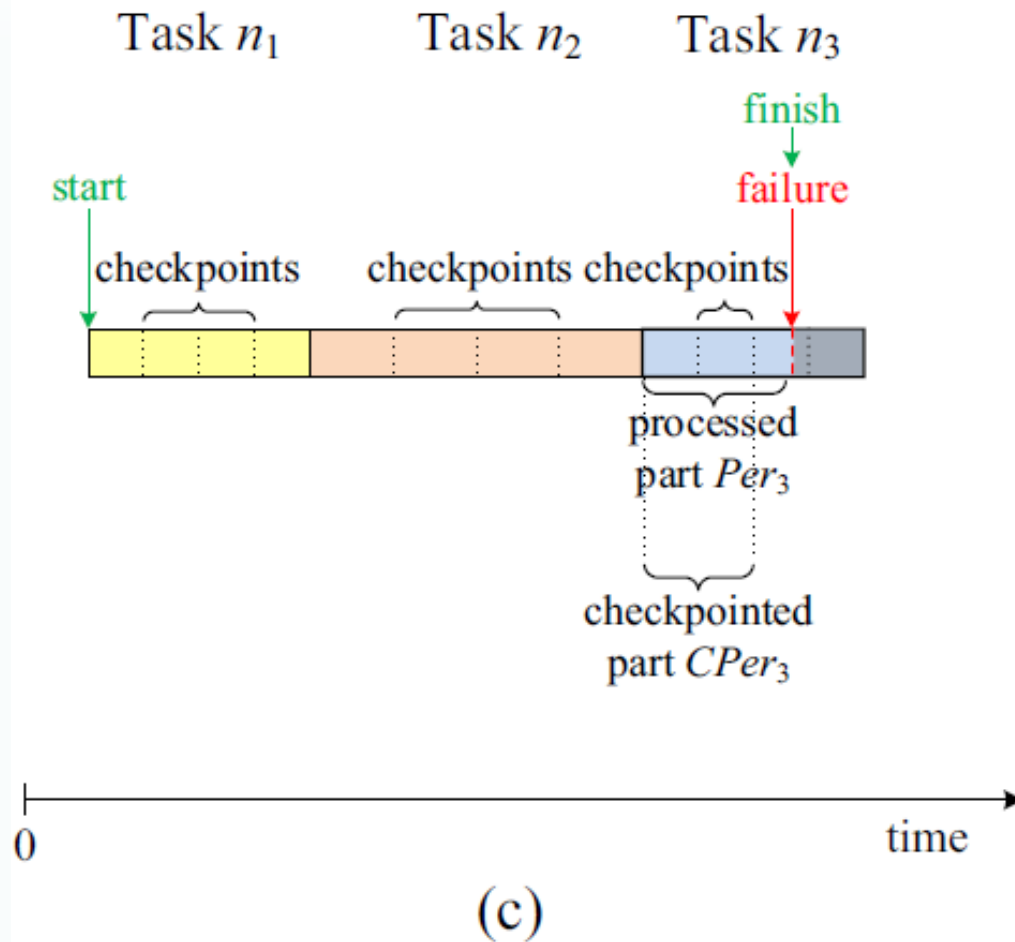


Fig. 14. A Linear Workflow - a failure occurs after the second checkpoint of the exit task (n_3) – imprecise computations.

Energy Efficiency (1/3)

- Energy efficiency in cloud systems reduces energy consumption and operational costs.
- There is a growing focus on the **minimization of the carbon footprint** of the computational resources, especially through the efficient scheduling of the workload towards the goal of **sustainable cloud**.
- However, energy conservation should be considered together with users' satisfaction regarding QoS.
- Complex multiple-task applications may have precedence constraints and specific deadlines and may impose several restrictions and QoS requirements.

Energy Efficiency (2/3)

- An **energy-aware heuristic** for the scheduling of **real-time workflow applications** in cloud environments has been proposed by Stavrinides and Karatza 2018, 2019.
- Their approach utilizes **per-core DVFS** on the underlying heterogeneous multi-core processors and **approximate computations**, in order to fill in schedule gaps.

G. L. Stavrinides and H. D. Karatza, “Energy-Aware Scheduling of Real-Time Workflow Applications in Clouds Utilizing DVFS and Approximate Computations”, in Proceedings of the 6th International Conference on Future Internet of Things and Cloud (FiCloud'18), Barcelona, Spain, 6-8 Aug, 2018.

G. L. Stavrinides and H. D. Karatza, “An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations”, Future Generation Computer Systems, Elsevier, Vol. 96, pp. 216-226, July 2019.

Energy Efficiency – Cloud (3/3)

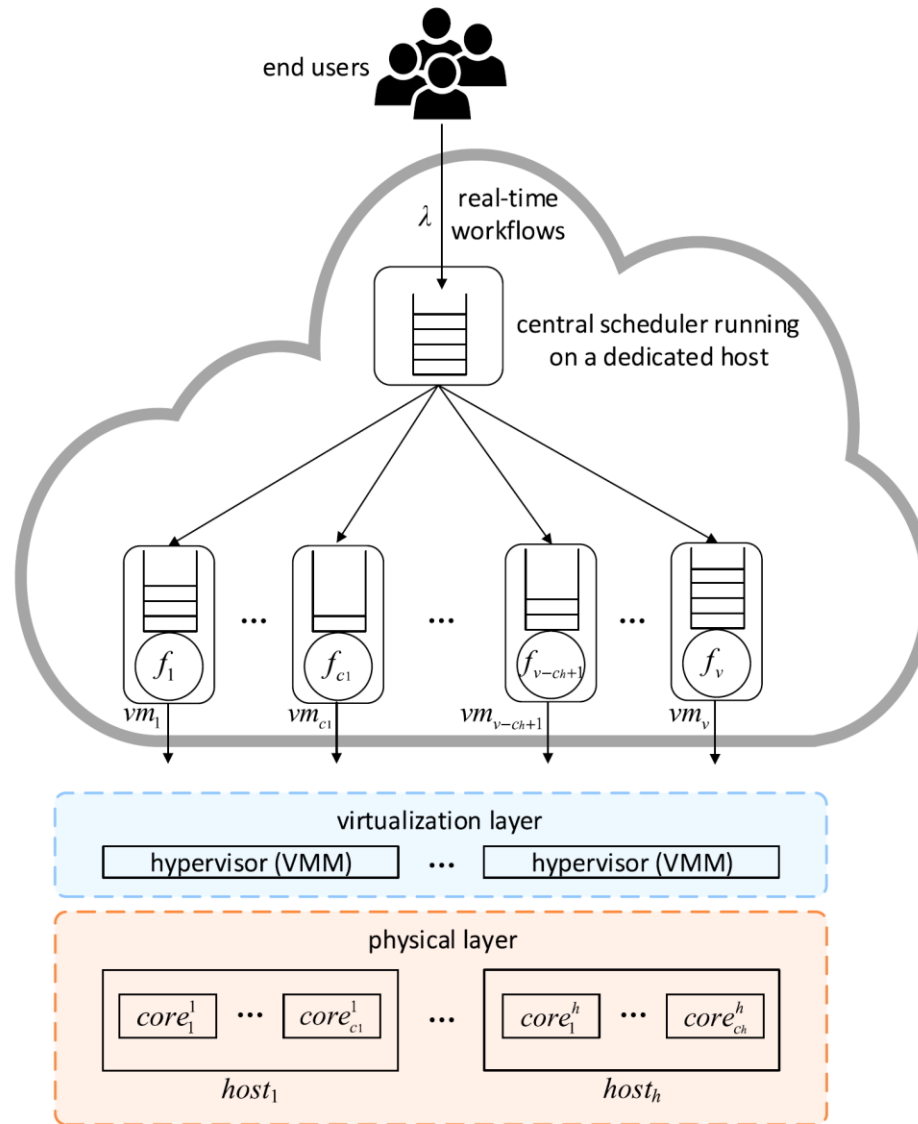


Fig. 15. The cloud computing environment

Fog Computing (1/4)

Fog Computing extends the Cloud Computing paradigm to **the edge of the network**, thus enabling a new breed of applications and services.

- The approach aims to schedule **computationally intensive tasks** with low communication demand to be executed on the **cloud**.
- Other tasks with low computational requirements and **high communication cost** are processed on the **fog**.

G. L. Stavrinides and H. D. Karatza, “A Hybrid Approach to Scheduling Real-Time IoT Workflows in Fog and Cloud Environments”, Multimedia Tools and Applications, Springer, vol. 78, no. 17, 2019.

Fog Computing (2/4)

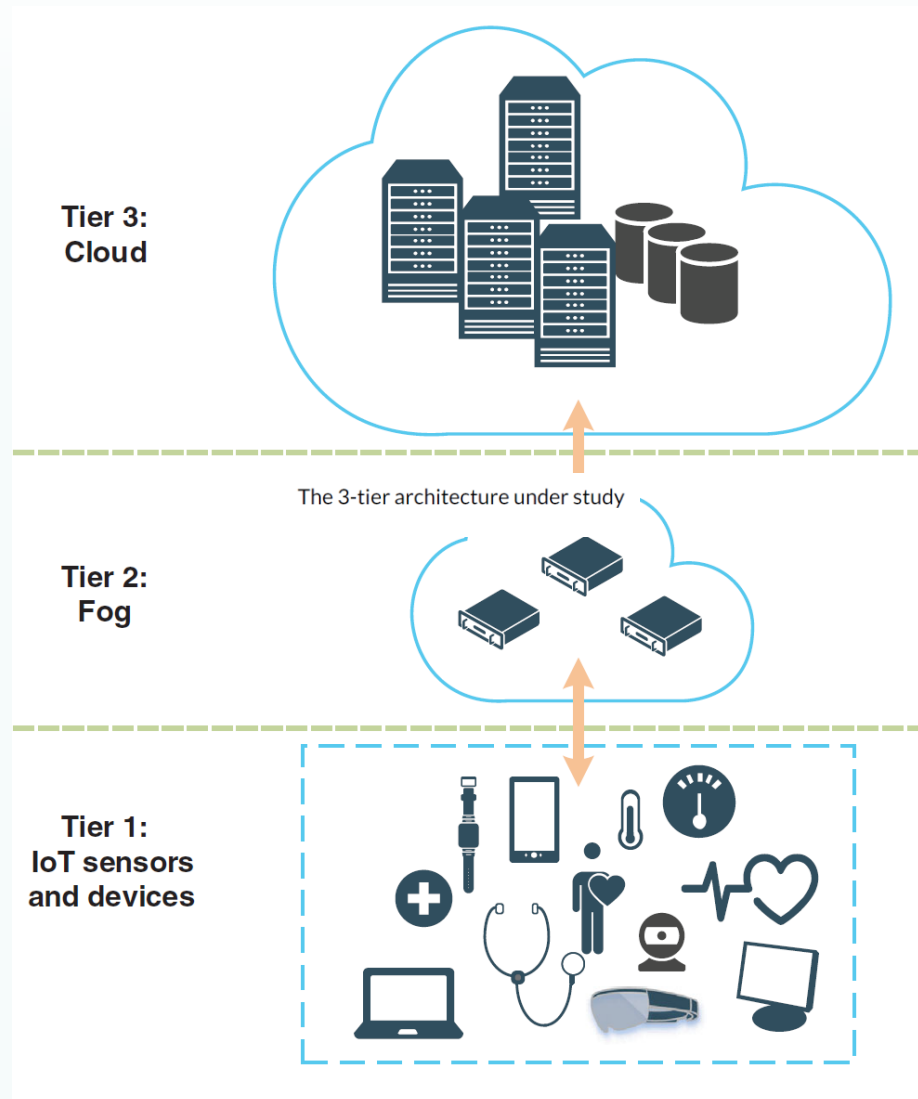


Fig. 16. The 3-tier architecture

Fog Computing (3/4)

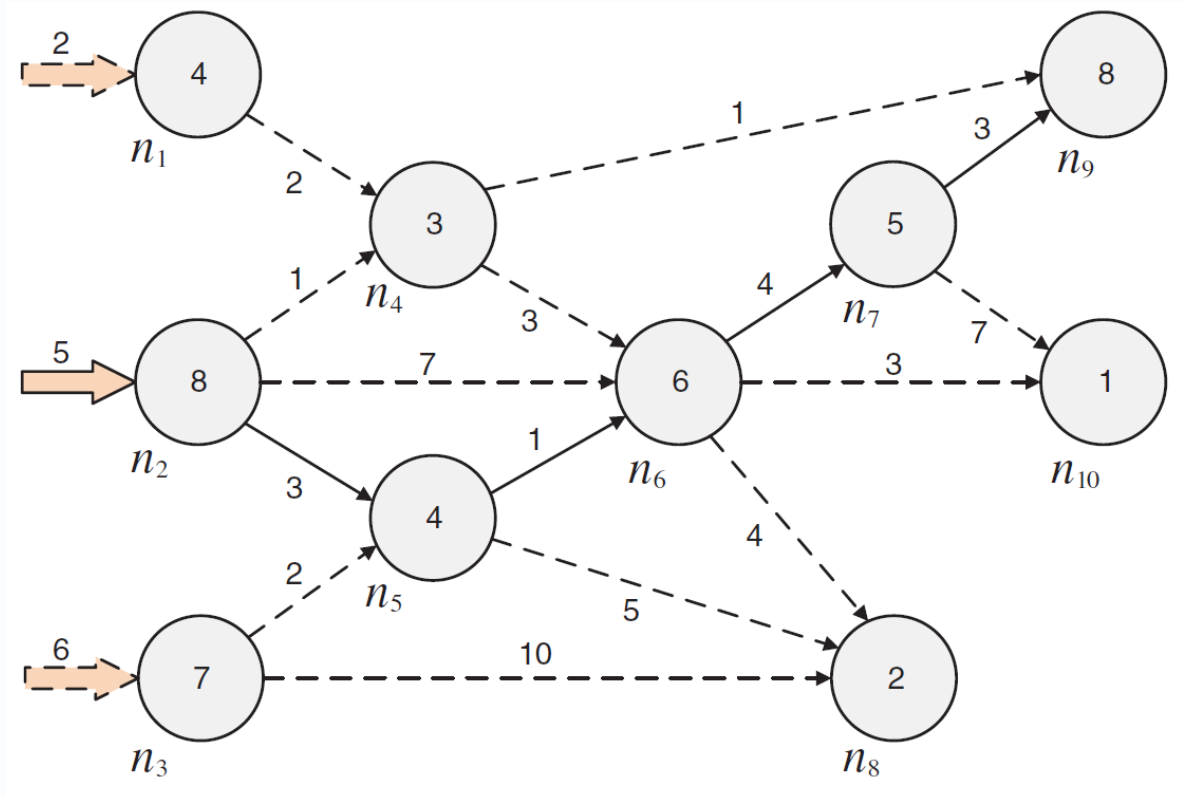


Fig. 17. A workflow job represented by a directed acyclic graph with three entry tasks and three exit tasks. **The arrows pointing to the entry tasks of the graph denote the initial input data of the job transferred from the IoT layer.**

Fog Computing (4/4)

G. L. Stavrinides and H. D. Karatza, “Orchestration of Real-Time Workflows with Varying Input Data Locality in a Heterogeneous Fog Environment”, in Proceedings of the 5th IEEE Intern. Conf. on Fog and Mobile Edge Computing (FMEC 2020), Paris, 30 June – 3 July, 2020.

- This research evaluates the performance of a **heterogeneous fog platform** where multiple data-intensive workflow jobs with deadline constraints arrive dynamically.
- Each entry component task of a workflow may require input data either from the IoT layer or from local fog resources.
- The impact of **workflow entry task input data locality** on the performance of the system, under different data location probabilities is investigated.

Mist Computing (1/5)

“Mist computing is the extreme edge of a network, typically **consisting of micro-controllers and sensors**. Mist computing uses microcomputers and microcontrollers **to feed into fog computing nodes** and potentially onward towards the centralised (cloud) computing services.

The 2 main goals of Mist computing include:

- Enabling resource harvesting by computation and communication capabilities available on the sensor itself.
- Allowing arbitrary computations to be provisioned, deployed, managed, and monitored on the sensor itself.”

Source: <https://radiocrafts.com/cloud-vs-fog-vs-mist-computing-which-one-should-you-use/>

Mist Computing (2/5)

- Mist computing, a lightweight form of fog computing, extends the fog layer even closer to the IoT sensors and devices.
- The collaboration of mist, fog and cloud resources for the processing of time-critical applications involves many challenges.
- Security aware heuristics for the scheduling of workflow jobs that process IoT data are required.

G. L. Stavrinides and H. D. Karatza, “Security and Cost Aware Scheduling of Real-Time IoT Workflows in a Mist Computing Environment”, 8th International Conference on Future Internet of Things and Cloud (FiCloud'21), 23-25 Aug, 2021.

Mist Computing (3/5)

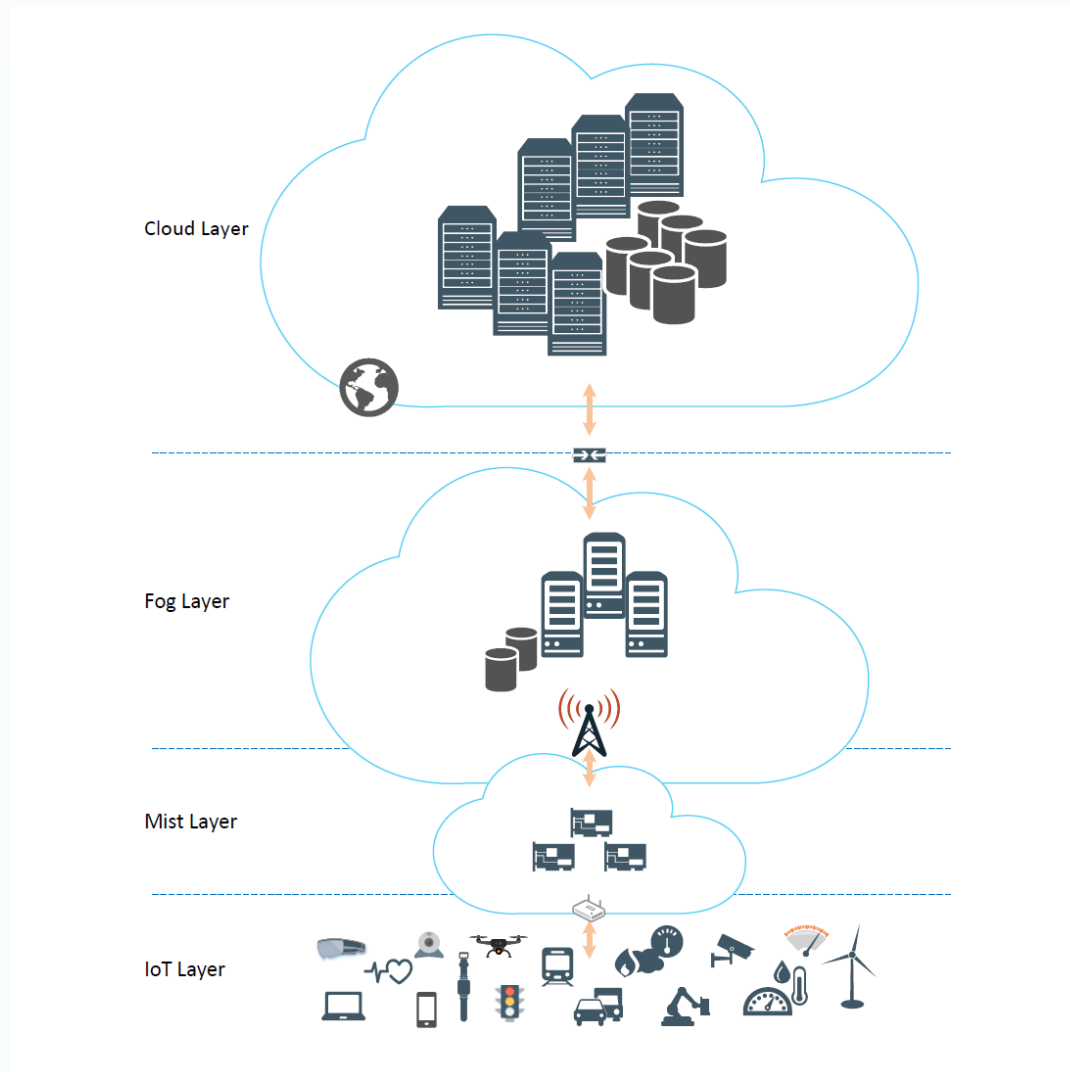


Fig. 18. The 4-tier architecture

Mist Computing (4/5)

- A security, cost and energy aware heuristic was proposed for the dynamic scheduling of multiple real-time workflow jobs that processed IoT data with various security requirements.
- The environment under study was a four-tier architecture, consisting of IoT, mist, fog and cloud layers.
- The proposed scheduling technique, was compared to a baseline policy, which was security aware, but not cost and energy aware.
- The simulation results revealed the superiority of the proposed policy in all of the examined aspects.

G. L. Stavrinides and H. D. Karatza, “Security, Cost and Energy Aware Scheduling of Real-Time IoT Workflows in a Mist Computing Environment”, Information Systems Frontiers, Springer, 2024.

Mist Computing (5/5)

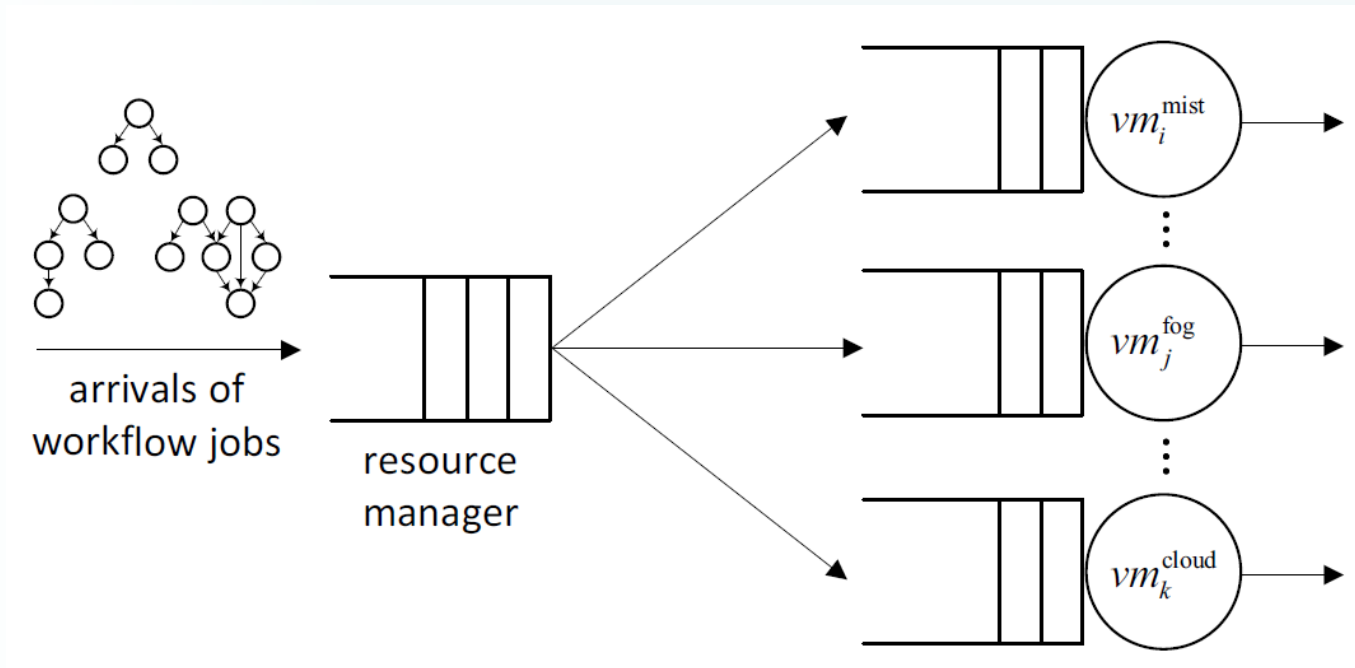


Fig. 19. The queueing model of the computational resources

Cloud-Fog-Mist Computing (1/2)

- **H. D. Karatza**, “Scheduling Linear Workflows with Multicriteria based Partial Computations in a Cloud–Fog–Mist Environment”, in Proceedings of the 9th International Conference on Fog and Mobile Edge Computing (FMEC 2024), Malmö, Sweden. 2-5 Sept. 2024, pp. 107-113.
- A mist LW job should be assigned exclusively to a mist VM, while a fog job can be directed to either a fog or a mist VM.
- Cloud LW jobs, however, may be allocated to either a cloud or a fog VM.
- The proportion of service demand omitted from **the final task of a cloud job**, due to partial computations, is contingent upon factors such as the maximum security level of fog jobs in the same VM queue, the waiting time limit associated with each security level, and the number of tasks within the cloud job.

Cloud-Fog-Mist Computing (2/2)

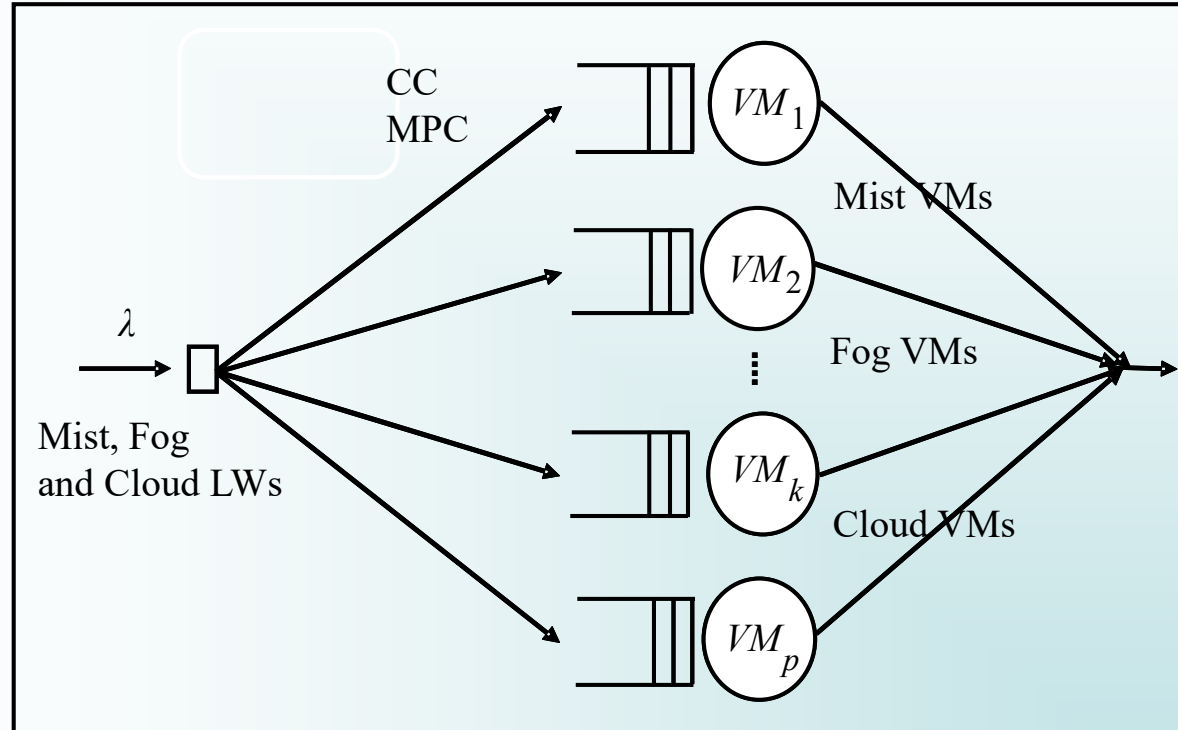


Fig. 20. The queueing model of the system under study.

Fluid Computing (1/1)

“Fluid computing is an architectural principle based on the abstraction of the topological details of the computational infrastructure. Fluid architectures provide an end-to-end fabric that can be used to seamlessly provision, deploy, manage and monitor applications, regardless of whether the underlying resource is provided by the cloud infrastructure, the fog infrastructure, or by things.

Fluid computing unifies under a single abstraction cloud, fog, and mist computing. *In other words, cloud, fog, and mist computing can be seen as application of fluid computing in a specific, bounded context.”*

Source: Angelo Corsaro, <https://www.embedded-computing.com/embedded-computing-design/fluid-computing-unifying-cloud-fog-and-mist-computing/>

Conclusions and Future Directions (1/2)

- Our perception of computing is changing constantly (**Cloud, Fog, Mist, ...**).
- The rise of Cloud computing presents a new opportunity for the evolution of computing.
- **Maybe**, in few years computers will be nothing more than thin-clients, and **all our processing will be done on the Clouds**.

Conclusions and Future Directions (2/2)

- However, **multiple issues have to be addressed** before Clouds become viable for large scale real-time distributed processing.
- **Security, availability** and **energy efficiency** will need the improvement of existing technologies, or the introduction of new ones, in order to achieve scalability that spans a very large number of nodes.

Thank you !

`karatza@csd.auth.gr`