Cloud Performance – Resource Allocation and Scheduling Issues

Eleni D. Karatza

Department of Informatics

Aristotle University of Thessaloniki

Greece

cHiPSet Training School

Aristotle University of Thessaloniki

19-21 September 2018

Scope

The scope of this lecture is to present:

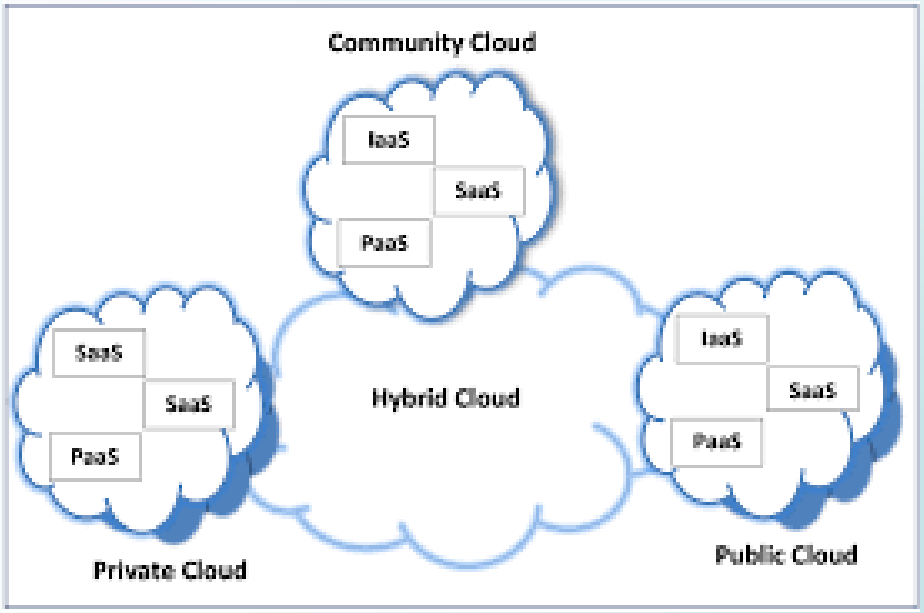
- state-of-the-art research covering a variety of concepts in cloud computing from the performance perspective,
- resource management issues that must be addressed in order to make clouds viable for HPC,
- efficient scheduling techniques for complex real-time applications
- to provide future trends and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

Presentation Structure

- Cloud Issues
- Performance Evaluation
- Resource Management and Scheduling in Clouds
- Complex Workloads – Real-Time Applications
- Mobile Cloud, Sky, Fog, Edge, Dew, Jungle and Dust Computing
- Conclusions and Future Direction

Cloud Issues (1/12)

- Cloud computing provides users the ability **to lease** computational resources from its virtually infinite pool for commercial, business, and scientific applications.



Cloud Issues (2/12)

- If cloud computing is going to be used for HPC, sophisticated methods must be considered for both real-time parallel job scheduling and VM scalability.

- Furthermore, high-speed, scalable, reliable networking is required for transferring data within the cloud and between the cloud and external clients.

Cloud Issues (3/12)

- Clouds were mostly used for simple **sequential applications**. However, recent evolutions enables the HPC community to run **parallel applications** in the Cloud.

- Good resource management policies can provide great improvements on different metrics:

  - maximum utilization of the resources,
  - faster execution times, and
  - better user's satisfaction (QoS guarantees).

Cloud Issues (4/12)

- Users can have access to a large number of computational resources at a fraction of the cost of maintaining a supercomputer center.

- A user can receive a service from the cloud without ever knowing **which** machines rendered the service, **where** it was located, or how many redundant copies of its data there are.

- The term “cloud” appears to have originated with **depiction of the Internet as a cloud hiding many servers and connections**.

Cloud Issues (5/12)

Cloud computing is a paradigm in which computing is moving from personal computers to large, centrally managed datacenters – **Questions:**

- What **new functionalities** are available to application developers and service providers?
- How do such applications and services leverage pay-as-you-go pricing models and rapid provisioning to meet **elastic demands** ?

Cloud Issues (6/12)

- The cloud model utilizes the concept of **Virtual Machines** (or VMs) which act as the computational units of the system.

- Depending on the computational needs of the jobs being serviced, new VMs can be **leased** and later **released** dynamically.

- It is important to study, analyze and evaluate both the performance and the overall cost of different scheduling algorithms.

Cloud Issues – Scheduling (7/12)

- The scheduling algorithms must seek a way to maintain **a good response time to leasing cost ratio.**

- Users requirements for **quality of service** (QoS) and specific system level objectives such as **high utilization**, **cost**, etc. have to be satisfied.

- Furthermore, **data security** and **availability** are critical issues that have to be considered as well.

Cloud Issues – Big Data (8/12)

- 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 (9/12)

- **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.

Cloud Issues – Big Data (10/12)

- A large body of work has been devoted to developing various **data-aware techniques** for the scheduling of data intensive applications.

- In this context, the **MapReduce** programming paradigm has been proposed by Google.

- This programming model is designed to **process large volumes of data in parallel** and it is inspired by the map and reduce functions commonly used in functional programming.

Cloud Issues – Big Data (11/12)

- The most popular implementation of the MapReduce model is the **Apache Hadoop** framework, which adopts a master slave architecture, in order to **process big data**, **exploiting data locality**.

- However, due to the fact that Hadoop considers only one slave node at a time in order to schedule the tasks, there are cases where it does not exploit data locality effectively. Furthermore, it does not take into account other characteristics of the workload, such as deadlines and resource usage fairness.

I. Mavridis and H. Karatza, "Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark", Journal of Systems and Software, Vol. 125, March 2017, pp. 133–151.

Cloud Issues – Privacy and Trust (12/12)

- A significant barrier to the adoption of cloud services is that users fear **data leakage** and **loss of privacy** if their sensitive data is processed in the cloud.

- **The privacy of data** has to be ensured - Users have to be reassured that their data will not be inadvertently released to others.

- Cryptographic techniques for enforcing the **integrity** and **consistency** of data stored in the cloud have to be studied.

Performance Evaluation – Simulation (1/3)

- The performance evaluation of clouds is often possible only **by simulation** rather than by **analytical techniques**, due to the complexity of the systems.

- Analytical modeling is difficult and often requires simplifying assumptions that may have an unpredictable impact on the results.

Performance Evaluation – Simulation (2/3)

- Advanced **modelling and simulation techniques** are a basic aspect of performance evaluation that is needed **before the costly prototyping actions** required for complex large scale distributed systems.

- Traces from real systems – Synthetic workloads.

Performance Evaluation – Workloads (3/3)

- Real workloads are representative of real systems.
  - However they are inflexible in the sense that they cannot be modified easily to answer **“what if”** questions.

- Synthetic workloads, allow researchers to directly vary the different parameters that can affect performance.
  - Thereby they permit the investigation of the impact of varying a given parameter on system performance.

Resource Allocation and Scheduling (1/3)

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 Scheduling***).

Real-Time Scheduling (1/8)

- Clouds are often used to run **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 and wireless communications.
- 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 (2/8)

- **Real-time Jobs**

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

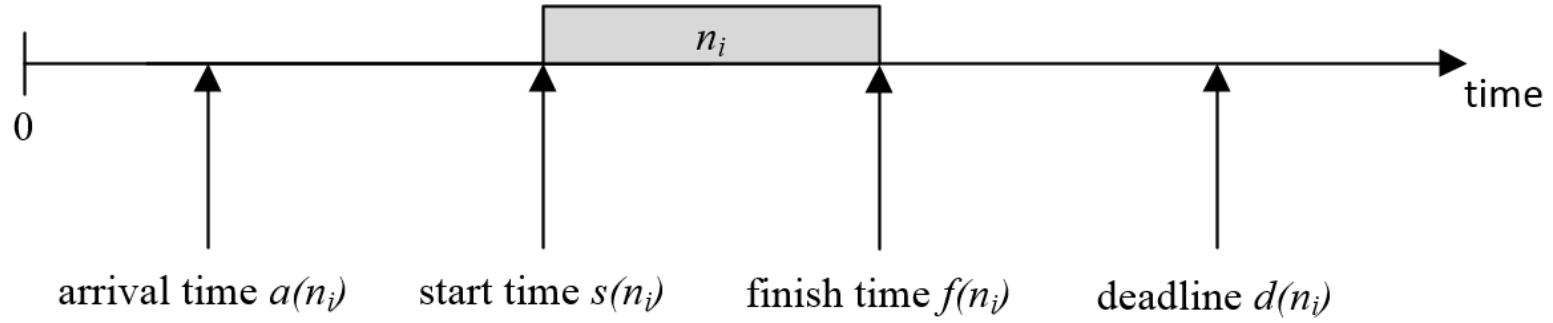


Fig. 1. An aperiodic job

Real-Time Scheduling (3/8)

- **Periodic jobs 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 $\boldsymbol{D_i \leq P_i}$.

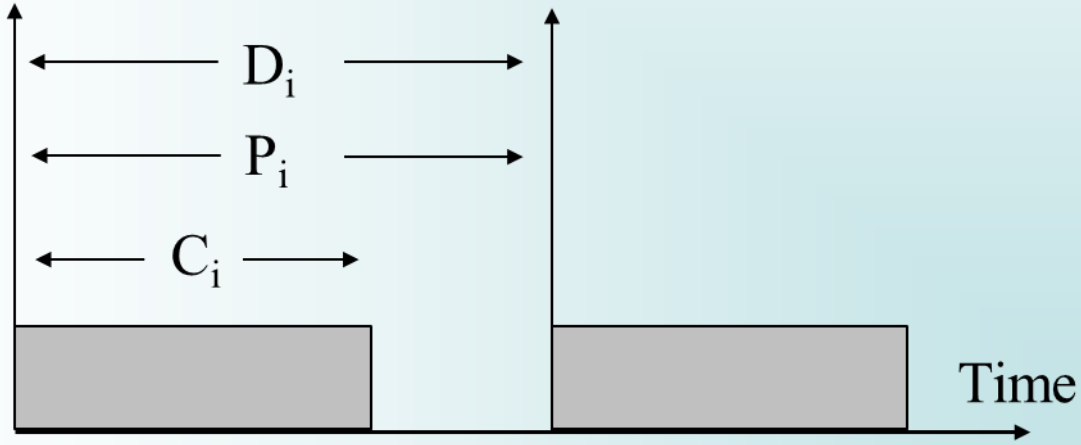


Fig. 2. A periodic job, $D_i = P_i$.

Real-Time Scheduling (4/8)

- 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 (5/8)

- 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 (6/8)

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)*.

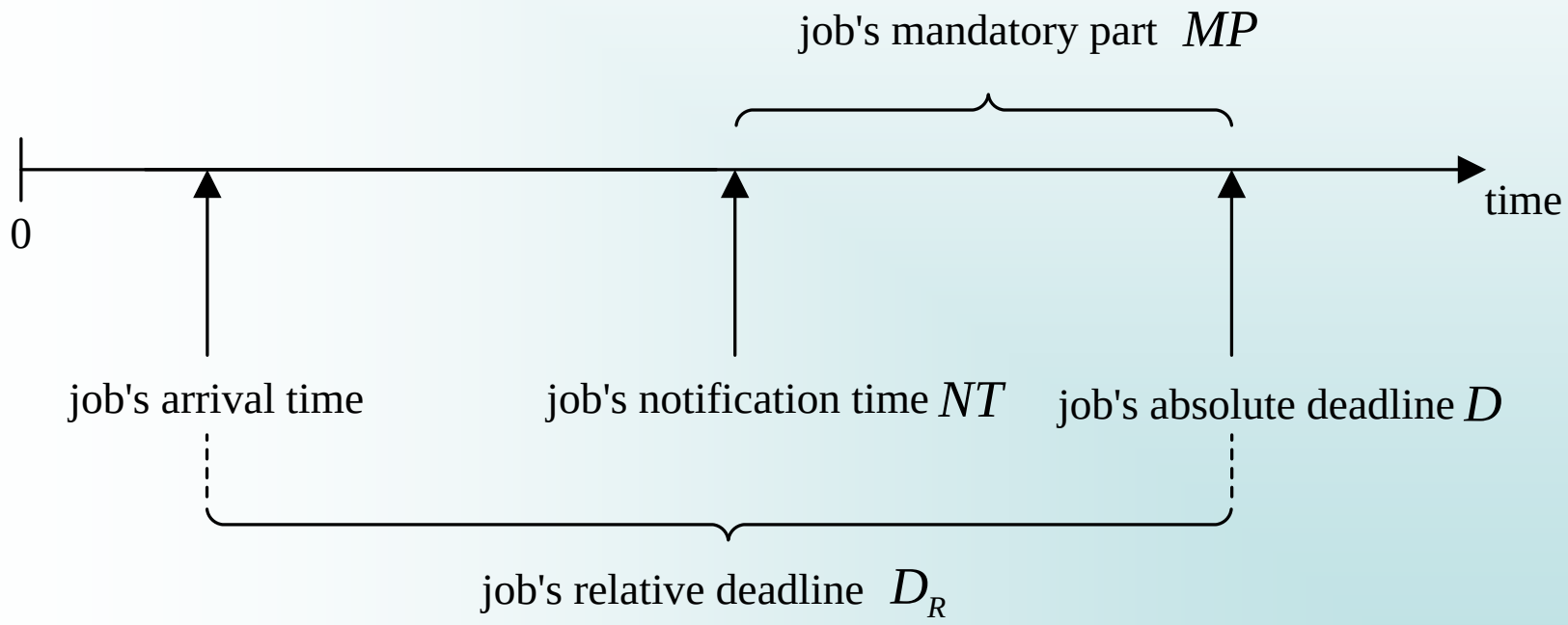


Fig. 3. A job's associated times in the Imprecise Computations case

Real-Time Scheduling (7/8)

- If a job *J* is waiting for service and its notification time is reached, then it can start execution if:
  - its assigned processor is idle or
  - the job in service on *J*'s assigned processor has completed its mandatory part. In this case, the job in service is aborted and job *J* occupies the processor.
- If job *J* cannot start execution, it is considered lost, because it will definitely miss its deadline.

Real-Time Scheduling (8/8)

- In the Imprecise Computations case, the output of a parent task in a DAG may be imprecise.
- Therefore, the child tasks that use as input the result of the particular parent task may have **input error**.
- Input error may cause an **increase in the execution time of the mandatory part of a child task**, since more time may be required by the child task to correct the error and produce an acceptable result.
- The quality of a DAG's results ultimately depends on the **result precision of the DAG's exit tasks**. Therefore, all exit tasks of a graph should be allowed to complete their entire optional part.

Fault-Tolerant Scheduling (1/4)

- 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 (2/4)

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.

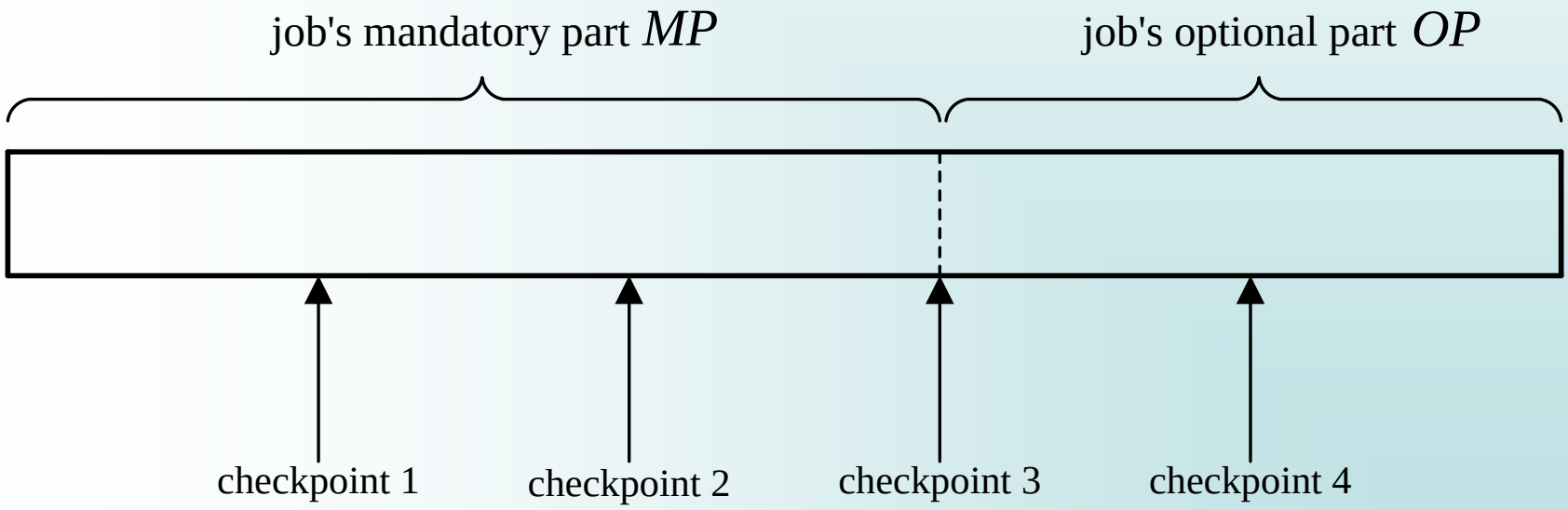


Fig. 4. Checkpoints

Fault-Tolerant Scheduling (3/4)

- When a failure occurs, the interrupted job is rolled back and resumes execution from its last generated checkpoint.

- If the last generated checkpoint of the interrupted job occurred after the completion of the job's mandatory part, then there is no need for rollback. The job is aborted and we accept the imprecise results saved by the job's last checkpoint.

Fault-Tolerant Scheduling (4/4)

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) (1/13)

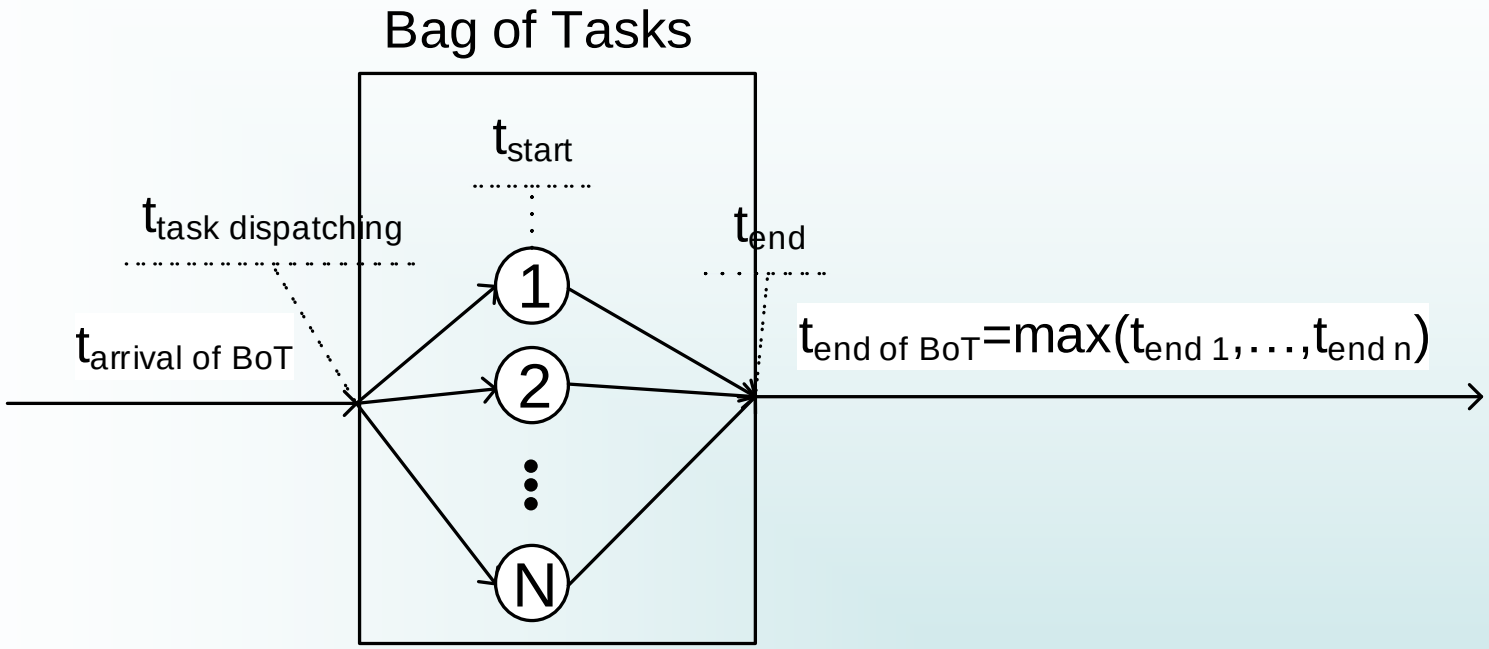


Fig. 5. A BoT

- 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) (2/13)

- **G. L. Stavrinides and H. D. Karatza**, “The Effect of Workload Computational Demand Variability on the Performance of a SaaS Cloud with a Multi-Tier SLA”, in Proceedings of the 5rd International Conference on Future Internet of Things and Cloud (FiCloud'17), Prague, Czech Republic, Aug. 2017, IEEE.

- A SaaS cloud with a multi-tier SLA that focuses on the fair billing of the end-users, according to the provided level of QoS is studied.

- The workload consists of bags-of-tasks with soft deadlines and different levels of variability in their computational demands.

Bag of Tasks Scheduling (BoT) (3/13)

- The bags-of-tasks are scheduled on the VMs of the underlying host environment.
- The impact of the workload computational demand variability on the system performance is investigated via simulation.
- The simulation results show that the computational demand variability has a different impact on the various performance metrics, depending on the employed scheduling strategy.

Bag of Tasks Scheduling (BoT) (4/13)

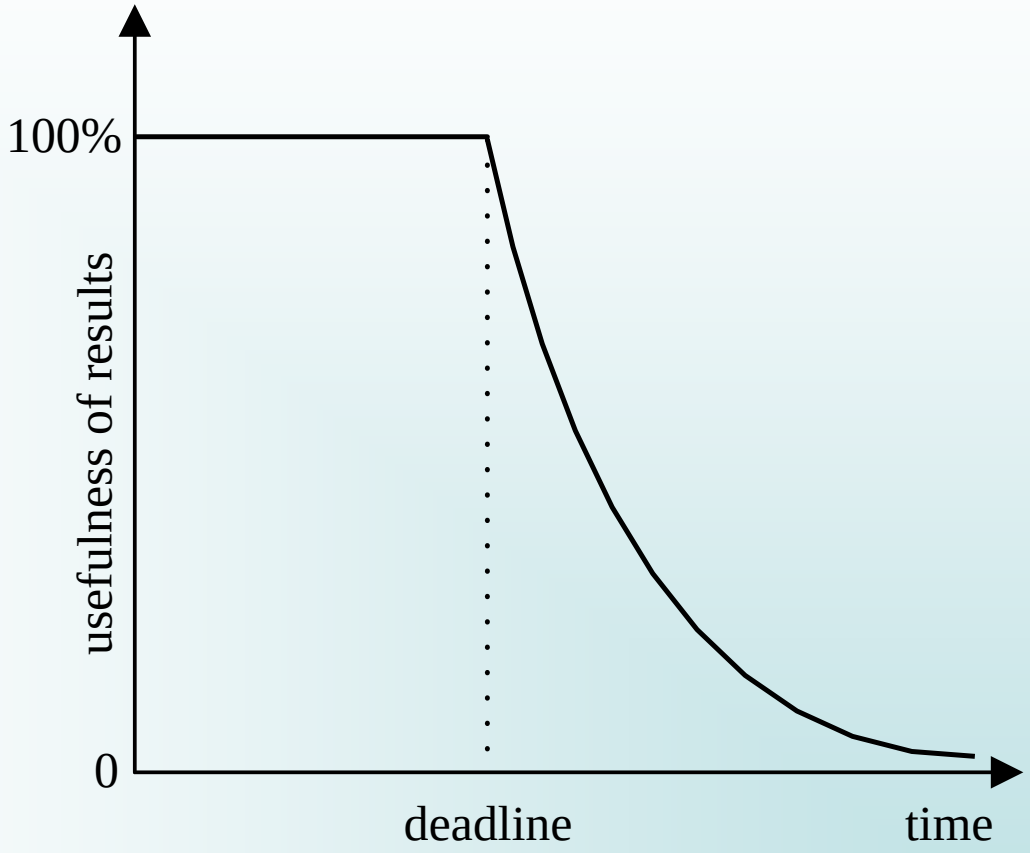


Fig. 6. The usefulness of the results of a job with a soft deadline over time.

Bag of Tasks Scheduling (BoT) (5/13)

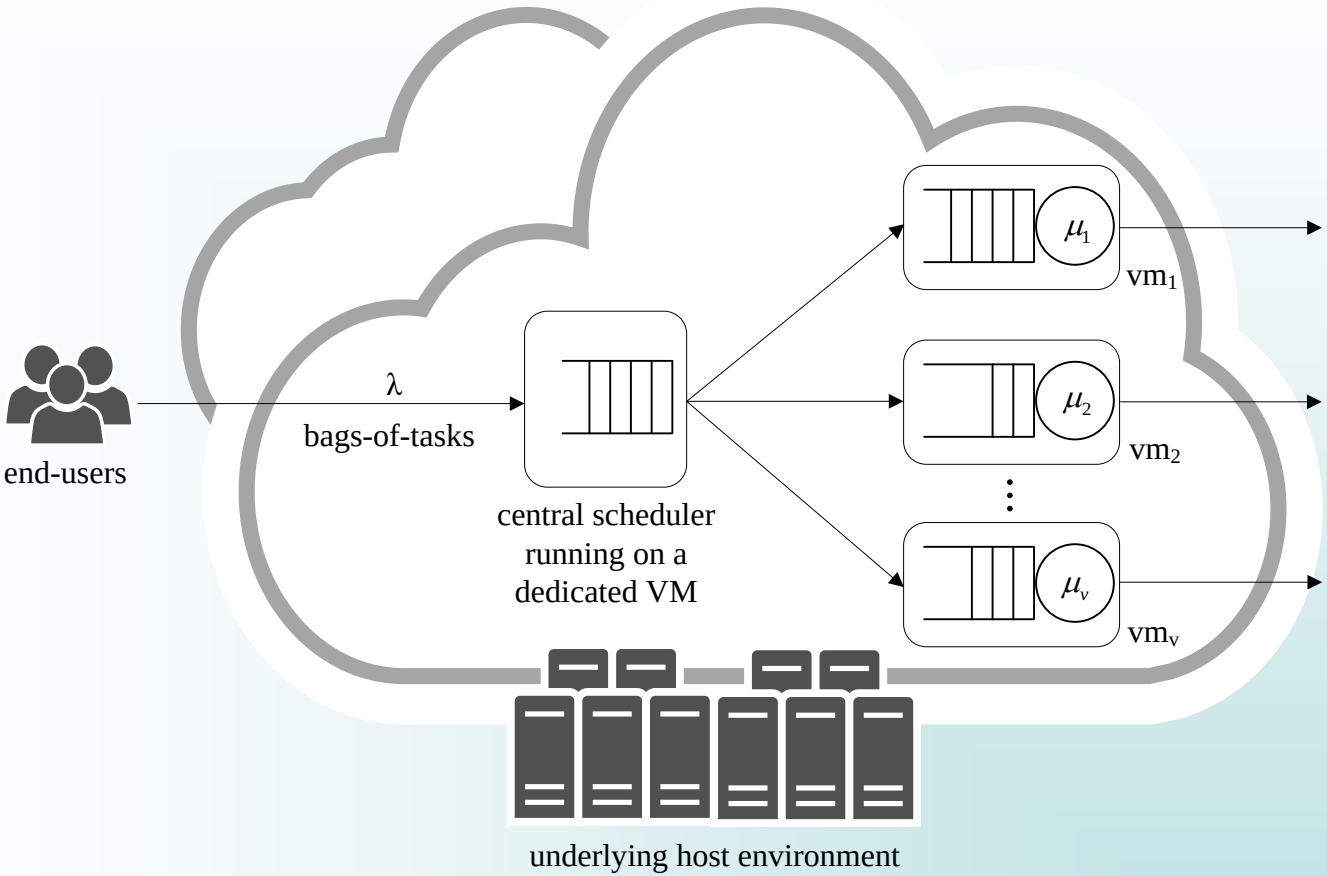


Fig. 7. The queueing model of the SaaS cloud.

Bag of Tasks Scheduling (BoT) (6/13)

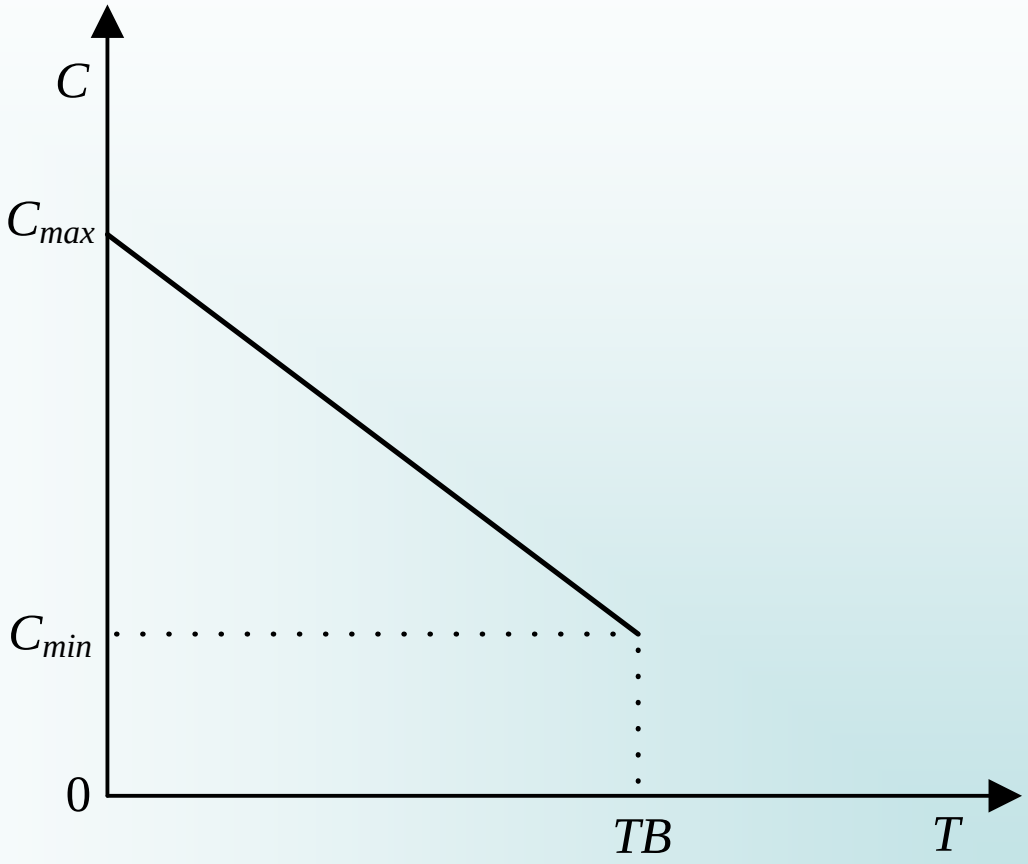


Fig. 8. The monetary cost per time unit charged for the execution of each job according to the provided level of QoS, as defined in the multi-tier SLA.

Bag of Tasks Scheduling (BoT) (7/13)

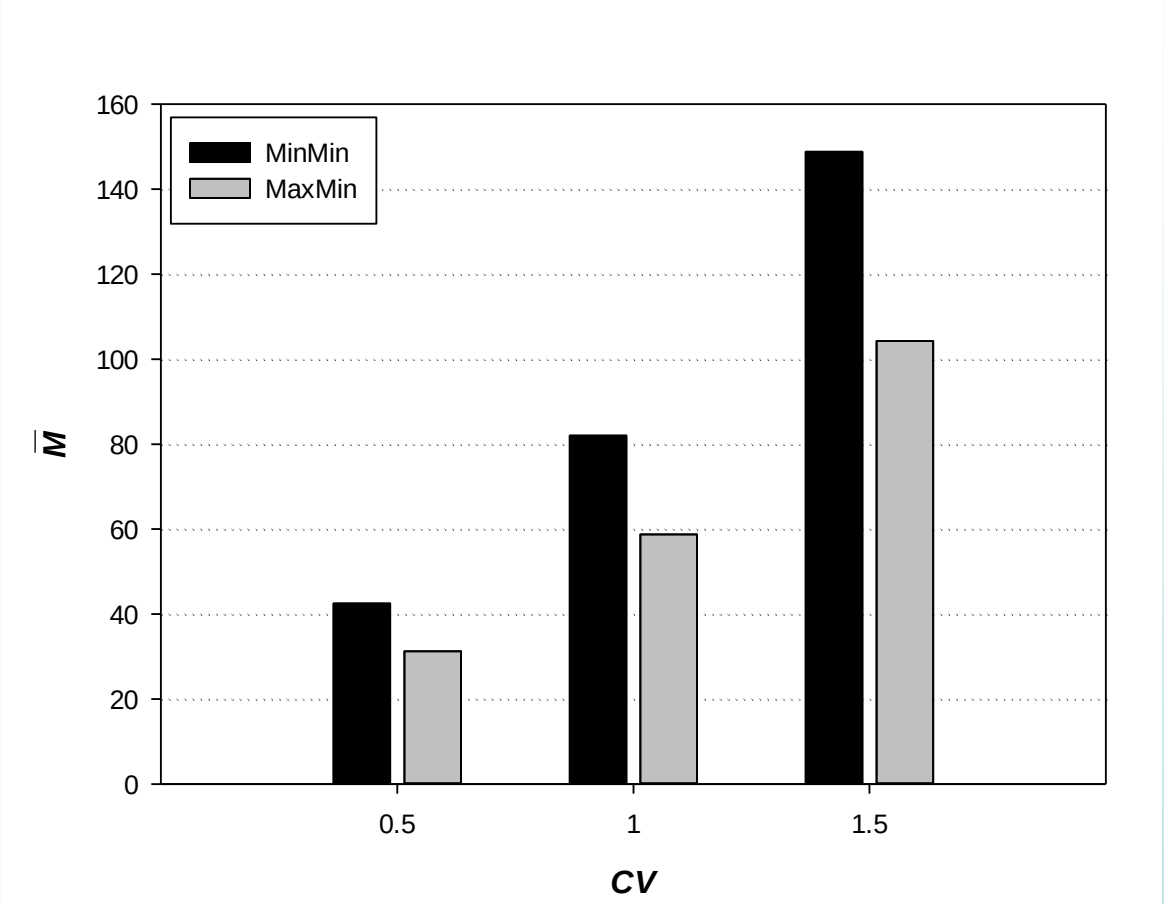


Fig. 9. Average makespan per completed job (M) vs. task computational volume coefficient of variation (CV).

Bag of Tasks Scheduling (BoT) (8/13)

G. L. Stavrinides and H. D. Karatza, “The impact of data locality on the performance of a SaaS cloud with real-time data-intensive applications”, in Proceedings of the 21st IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT'17), Rome, Italy, October 18-20, 2017.

- The impact of **data locality** on the performance of a SaaS cloud, where real-time, data-intensive bags-of-tasks are scheduled dynamically, under various data availability conditions is investigated.

- The simulation results show that among the other characteristics of the workload, data locality should be taken into account during scheduling, particularly **in the cases were the input data are not replicated** on all of the VMs in the cloud.

Bag of Tasks Scheduling (BoT) (9/13)

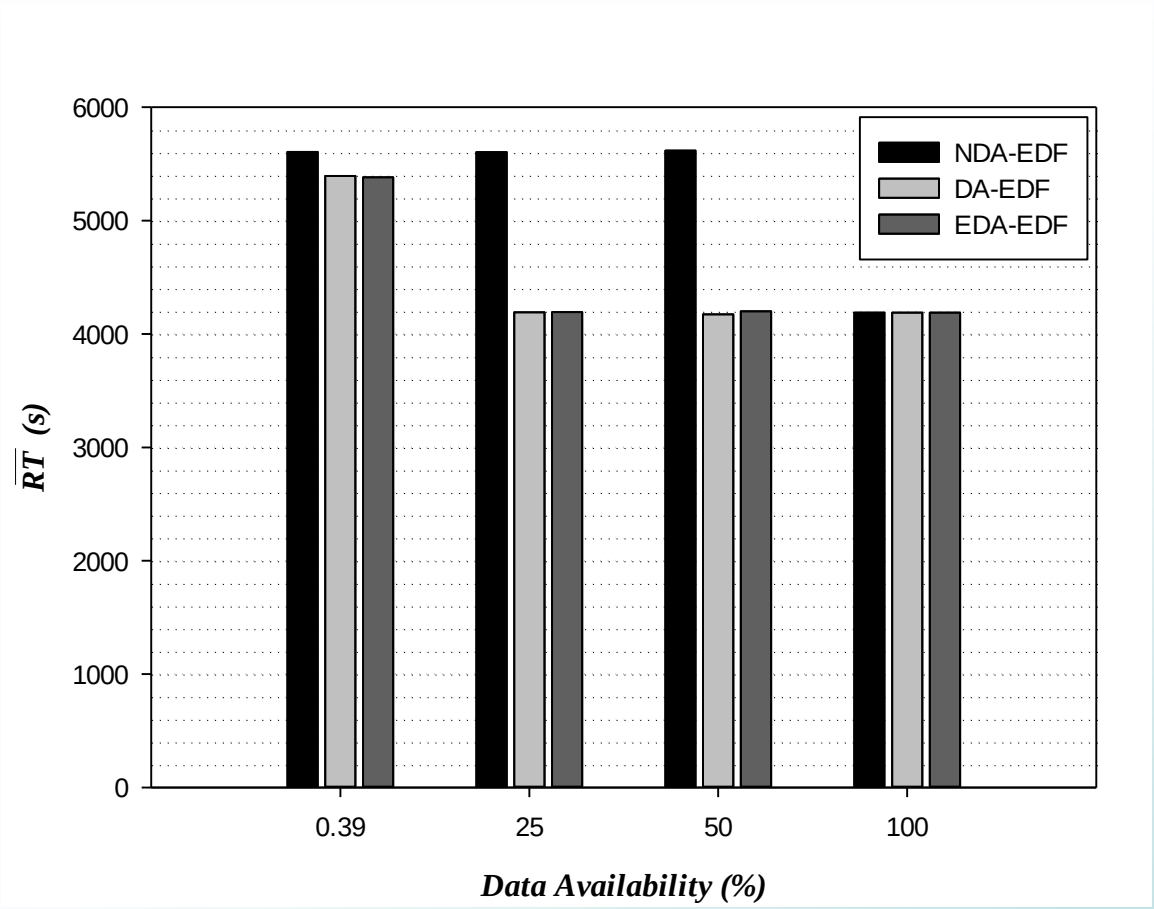


Fig. 10. Average response time per completed job *RT* (s) vs. Data Availability (%).

Bag of Tasks Scheduling (BoT) (10/13)

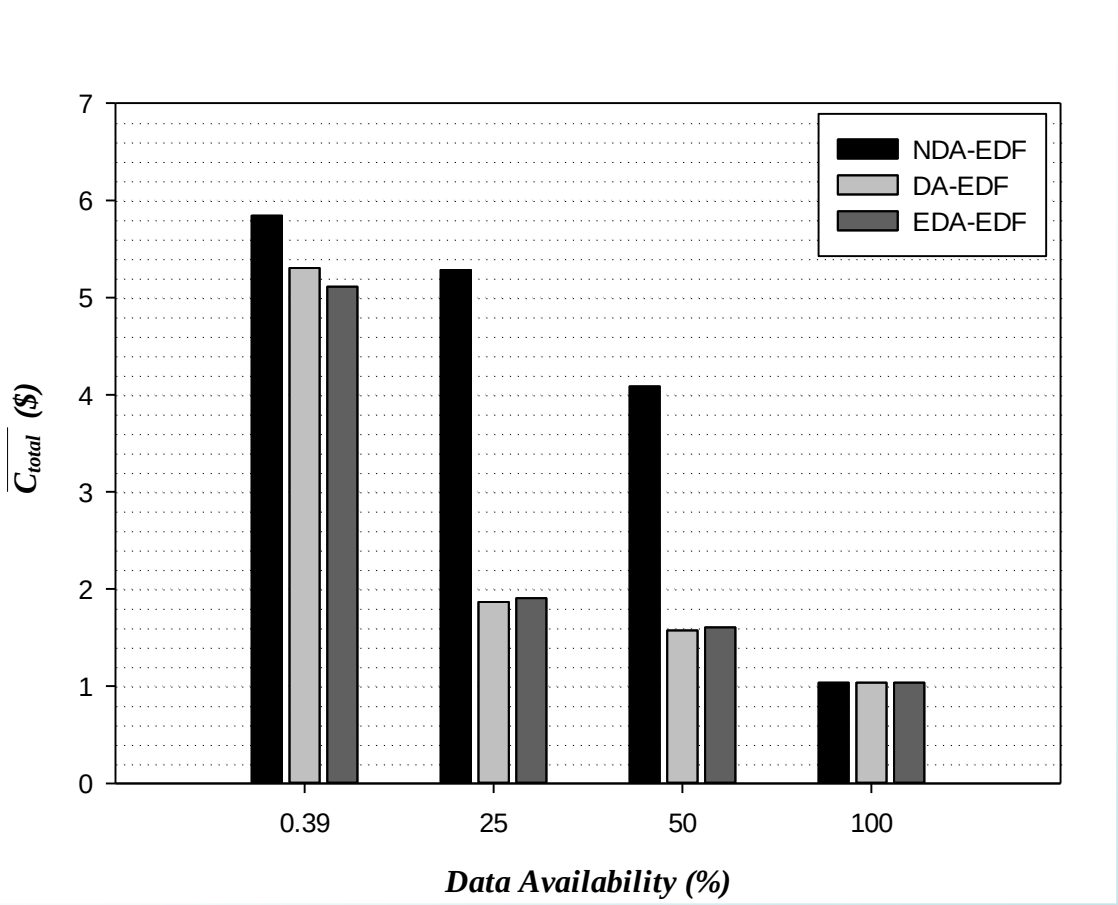


Fig. 11. Average total monetary cost per completed job C_{total} ($) vs. Data Availability (%).

Bag of Tasks Scheduling (BoT) (11/13)

G. L. Stavrinides and H. D. Karatza, Scheduling real-time bag-of-tasks applications with approximate computations in SaaS clouds, Concurrency and Computation: Practice and Experience, Wiley, First published online: 20 June 2017.

- Some of the most commonly used scheduling algorithms for bag-of-tasks applications are enhanced by utilizing approximate computations.

- The impact of different levels of variability in the computational demands of the applications on the performance of the examined heuristics is investigated.

Bag of Tasks Scheduling (BoT) (12/13)

- **Ioannis A. Moschakis and Helen D. Karatza**, “Multi-criteria scheduling of Bag-of-Tasks applications on heterogeneous interlinked Clouds with Simulated Annealing, Journal of Systems and Software, Elsevier, Vol. 101, March 2015.

- While the use of the **meta-heuristics** does impose a performance overhead due to their complexity in comparison to simpler heuristics,

  the experimental analysis shows that only a relatively small number of steps is required in order to achieve an optimized schedule.

Bag of Tasks Scheduling (BoT) (13/13)

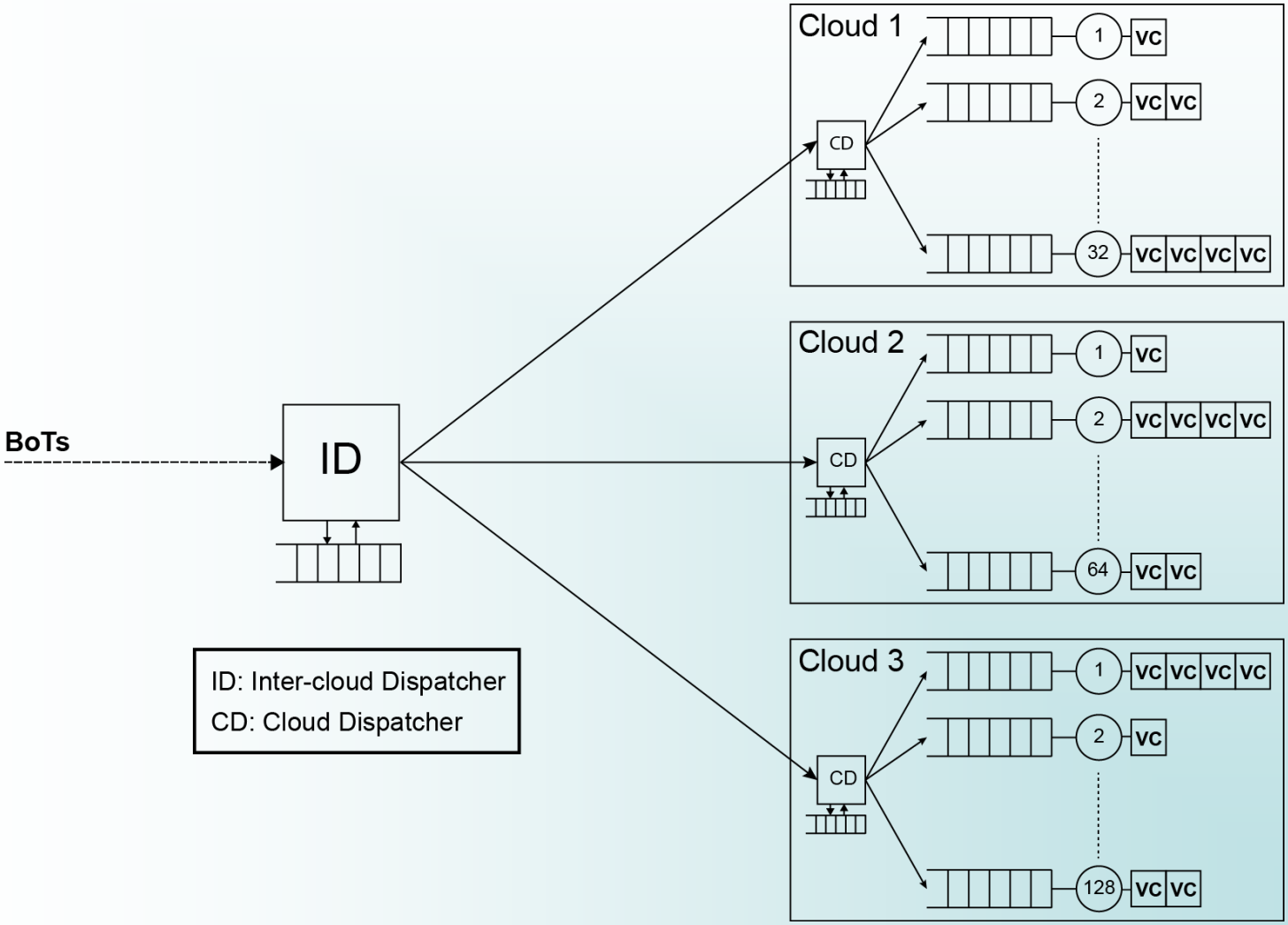


Fig. 12. Interlinked Clouds

Gang Scheduling (1/8)

- 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 (2/8)

- The number of tasks in a gang must be less or equal to the number of available processors.

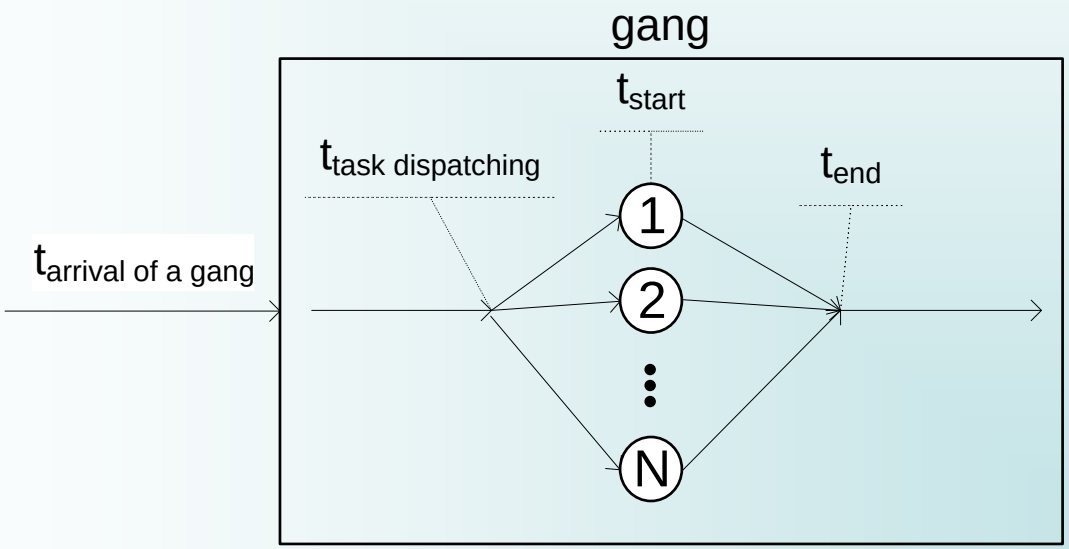


Fig. 13. Model of a gang job with N tasks

Gang Scheduling (3/8)

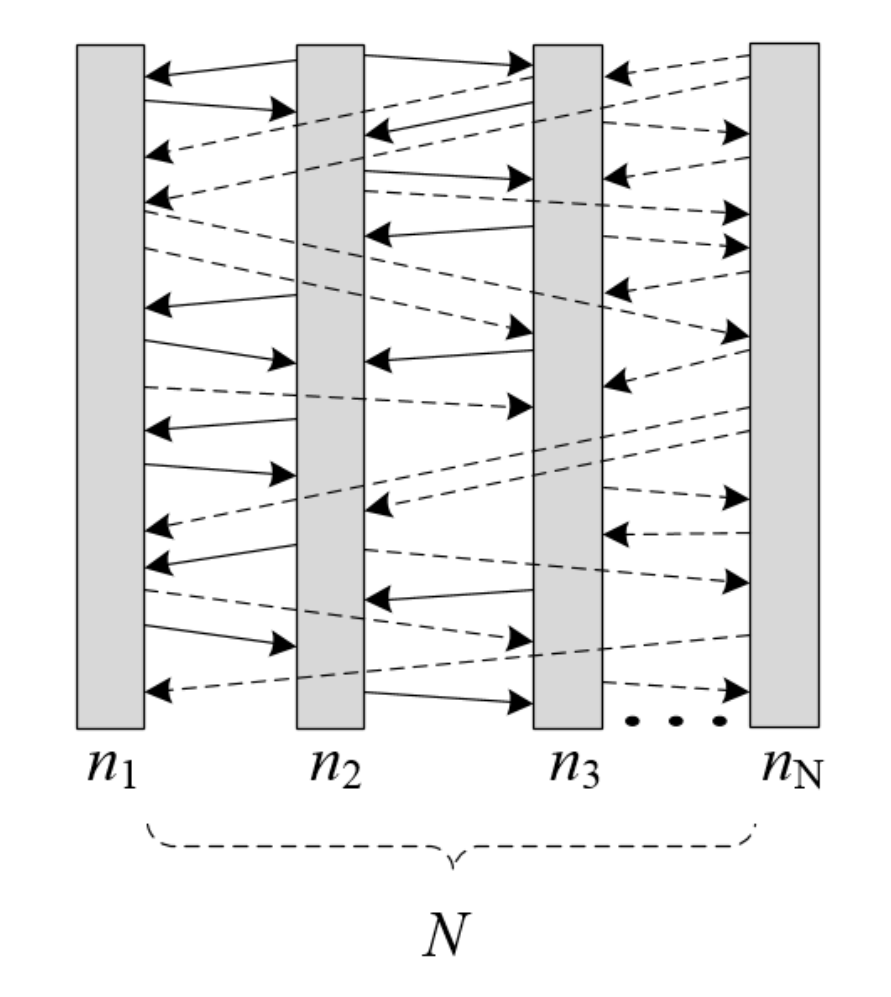


Fig. 14. A gang with *N* parallel frequently communicating tasks.

Gang Scheduling (4/8)

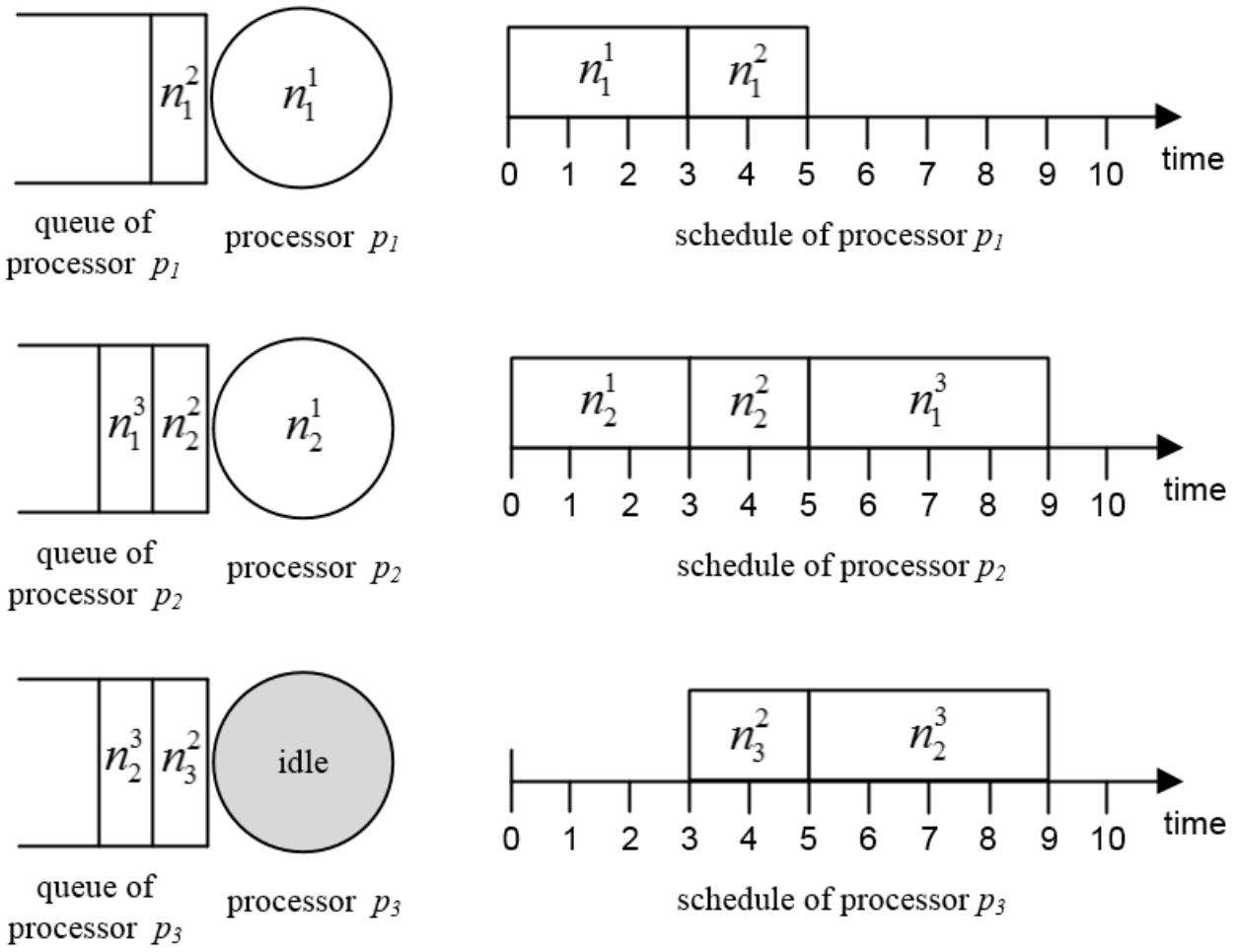


Fig. 15. Example of gang scheduling.

Gang Scheduling (5/8)

- In Gang Scheduling, the tasks of a job need to start execution simultaneously, because in this way the risk of a task waiting to communicate with another task that is currently not running is avoided.
- Without Gang Scheduling, the synchronization of a job's tasks would require more context switches and thus additional overhead.
- In Gang Scheduling, in order for a job with N tasks to be completed, N processors must execute the tasks concurrently.

Gang Scheduling (6/8)

- **G.L. Stavrinides, H.D. Karatza**, “Scheduling different types of applications in a SaaS cloud”, in Proceedings of the 6th International Symposium on Business Modeling and Software Design (BMSD'16), Rhodes, Greece, Jun. 2016, pp.144-151.

- One of the major challenges is to cope with the case where high-priority real-time single-task applications arrive and have to interrupt other non-real-time parallel applications in order to meet their deadlines.

- In this case, it is required to effectively deal with the real-time applications, at the smallest resulting degradation of parallel job performance.

Gang Scheduling (7/8)

The workload consists of

- non-real-time gangs, and
- periodic high-priority soft real-time single-task applications that can tolerate deadline misses by bounded amount.

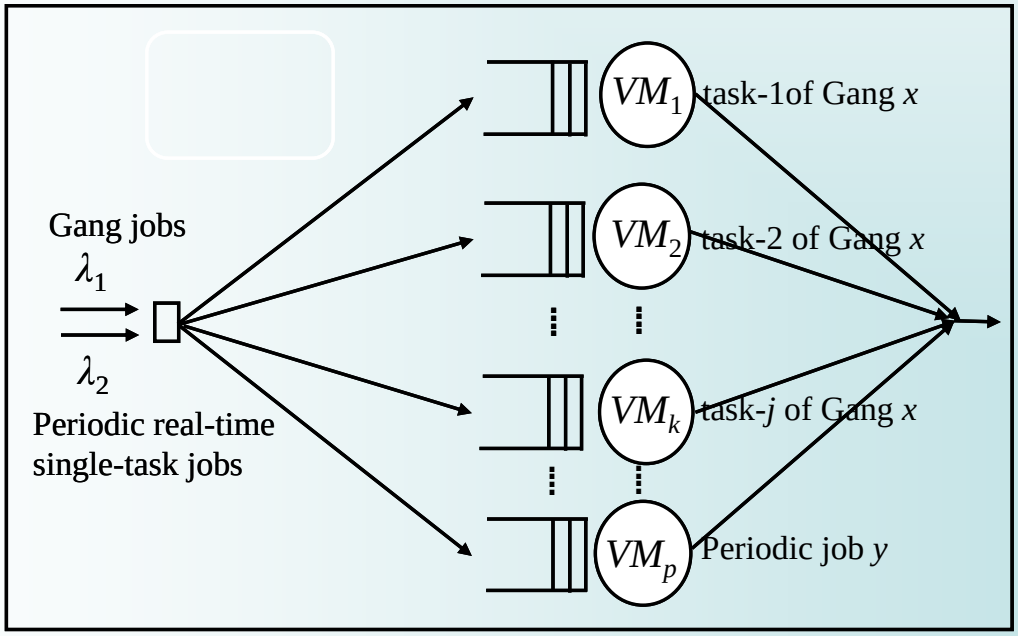


Fig. 16. The system model

Gang Scheduling (8/8)

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 Scheduling (1/4)

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).***
- In order for a task to start execution, all of its predecessor tasks must have been completed.

DAG Scheduling (2/4)

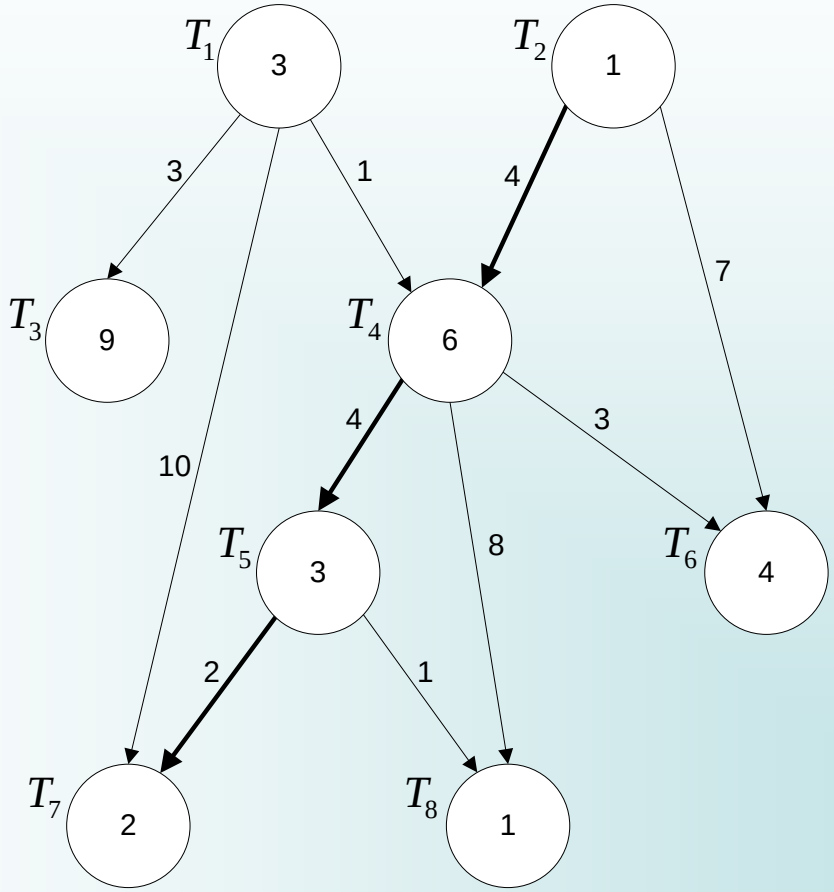


Fig. 17. A Directed Acyclic Graph (DAG)

DAG Scheduling (3/4)

- **G.L. Stavrinides and H.D. Karatza,** “Scheduling Real-Time DAGs in Heterogeneous Clusters by Combining Imprecise Computations and Bin Packing Techniques for the Exploitation of Schedule Holes”, Future Generation Computer Systems, Elsevier, Vol. 28, No. 7, pp. 977-988, July 2012.

- The improvement that can be gained in the performance of a heterogeneous cluster dedicated to real-time DAG jobs, by exploiting schedule holes with an approach that combines imprecise computations and bin packing strategies is investigated.

DAG Scheduling (4/4)

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.

- Scheduling heuristic for **real-time workflow applications** in a heterogeneous PaaS (or SaaS) cloud.

- **Objectives:**

  (a) to guarantee that all applications will meet their deadline, providing high quality results, and

  (b) to minimize the execution time of each workflow application and thus the cost charged to the user.

Scheduling Data-Intensive Workloads (1/2)

G. L. Stavrinides and H. D. Karatza, “Scheduling Data-Intensive Workloads in Large-Scale Distributed Systems: Trends and Challenges”, in Modeling and Simulation in HPC and Cloud Systems, Springer’s Studies in Big Data, Springer, pp. 19-43, 2018.

- With the growth of **big data**, workloads tend to get more complex and computationally demanding.
- **Data-intensive applications** are typically processed on interconnected computing resources that are geographically distributed. Computational grids and clouds are examples of such platforms.

Scheduling Data-Intensive Workloads (2/2)

- Data-intensive applications may have different degrees of parallelism and **must effectively exploit data locality**.
- Furthermore, they may impose several Quality of Service requirements.
- These features of the workloads present major challenges that require the employment of effective scheduling techniques.

Energy Efficiency (1/2)

- Energy efficiency in large scale distributed systems reduces energy consumption and operational costs.
- 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.
- There is a growing focus on the **minimization of the carbon footprint** of the computational resources, especially through the efficient scheduling of the workload.

Energy Efficiency (2/2)

- In Stavrinides and Karatza ACM ICPE 2017 a technique is proposed for the energy-aware scheduling of **bag-of-tasks applications with time constraints** in a large-scale heterogeneous distributed system.
- The simulation results show that the proposed heuristic not only reduces the energy consumption of the system, but also improves its performance.

G. L. Stavrinides and H. D. Karatza, “Simulation-Based Performance Evaluation of an Energy-Aware Heuristic for the Scheduling of HPC Applications in Large-Scale Distributed Systems”, in Proceedings of ENERGY-SIM 2017, 23rd April 2017, L’Aquila, Italy, in conjunction with the 8th ACM International Conference on Performance Engineering (ACM ICPE) 2017.

Mobile Cloud Computing (1/2)

- The enormous growth of cloud computing, together with the advance in mobile technology have led to the new era of **Mobile Cloud Computing (MCC)**.
- Efficient and reliable management of distributed resources in Mobile Clouds became more important due to the increase of users and applications.
- However, **adapting the cloud paradigm for mobile devices** is still in its early stage and several issues are yet to be answered.

Tundong Liu, Fufeng Chen, Yingran Ma, Yi Xie, An energy-efficient task scheduling for mobile devices based on cloud assistant, Future Generation Computer Systems, Vol. 61, 2016, pp. 1–12.

Mobile Cloud Computing (2/2) - Security issues

- All the security issues of cloud computing plus the extra limitation of resource constraint need to be studied.
- Therefore, **the security algorithms** proposed for the cloud computing environment **cannot be directly run on a mobile device**.
- There is a need for **a lightweight secure framework** that provides security with **minimum communication and processing overhead** on mobile devices.

Abdul Nasir Khan, M.L. Mat Kiah, Samee U. Khan, Sajjad A. Madani, Towards secure mobile cloud computing: A survey, Vol. 29, Issue 5, 2013, pp. 1278–1299.

From Cloud to Sky Computing (1/2)

Sky with Clouds !

Sky Computing:
An aggregation of several heterogeneous Clouds.

A Monteiro, C. Teixeira, J.S. Pinto, Sky Computing: exploring the aggregated Cloud resources, Cluster Computing, 2017, pp. 20:621–631.

From Cloud to Sky Computing (2/2)

- In order to have **different Clouds compatible together**, **standards** are being developed and also users develop software **compatible** with multiple Cloud platforms.

R. Buyya, R. Ranjan, and R. Calheiros. InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. LNCS, Vol. 6081 pp. 13–31, Springer Berlin / Heidelberg, 2010.

Fog Computing (1/3)

While the need for scalability and speed is increasing, the **resources available to the end-users** are often **more diverse** than those contained in a single cluster, grid, or Cloud System.

Moreover, more and more applications, e.g. IoT applications, are producing a significantly **huge amount of data** and **it is not sensible to upload all of them on the Cloud.**

As a result, Fog Computing Systems are proposed so that all the available computational power be combined and be **closer to the application**.

Fog Computing (2/3)

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

a) Low latency and location awareness,
b) Wide-spread geographical distribution,
c) Mobility,
d) Very large number of nodes,
e) Predominant role of wireless access,
f) Strong presence of streaming and **real time applications**,
g) Heterogeneity

Flavio Bonomi et als., Fog Computing and Its Role in the Internet of Things, MCC'12, August 17, 2012, Helsinki, Finland

Fog Computing (3/3)

Virtual Machines,
Raspberry PIs,
Local PCs Cluster
and Smartphones.

D. Tychalas and H. Karatza, “Simulation and Performance Evaluation of a Fog System”, The Third IEEE International Conference on Fog and Mobile Edge Computing (FMEC 2018), Barcelona, Spain, April 23-26, 2018.

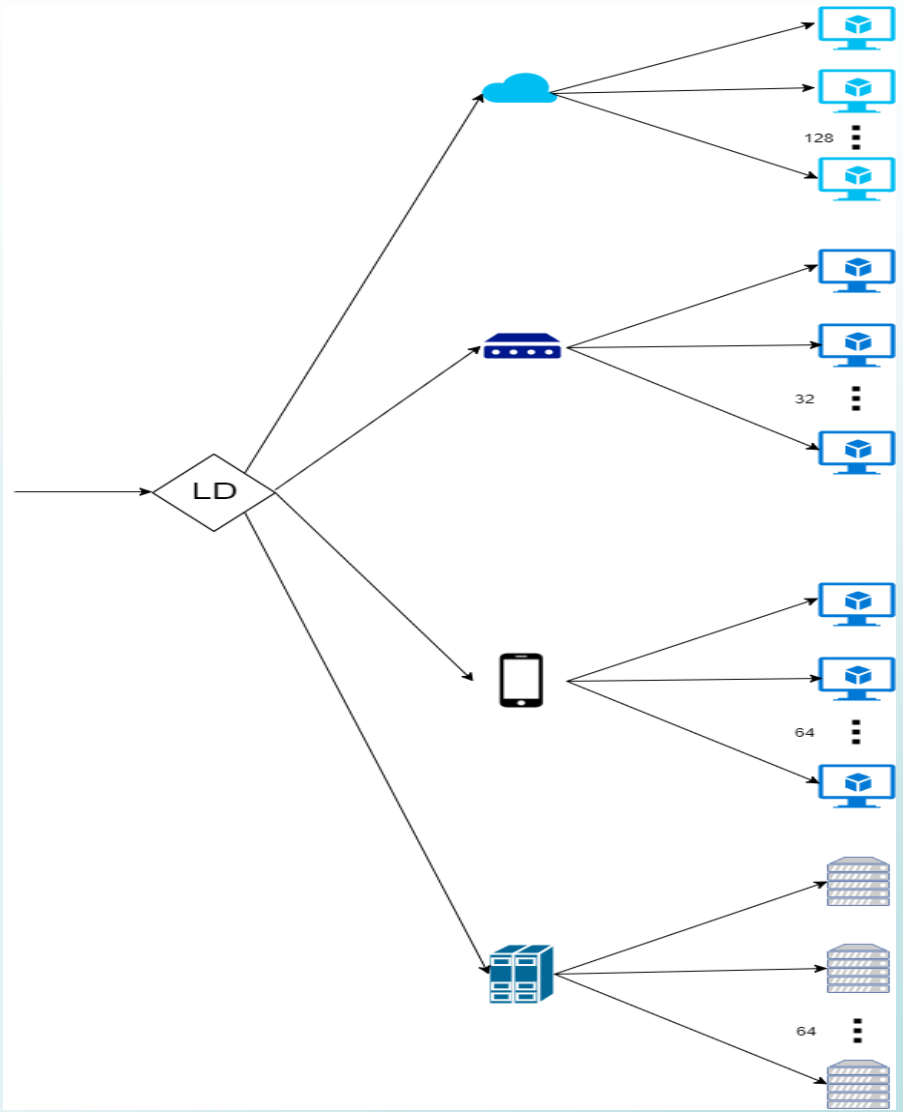


Fig. 18. A Fog system model

Edge Computing (1/1)

H. Chang; Hari, A.; Mukherjee, S.; Lakshman, T.V., "Bringing the cloud to the edge," *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on* , pp.346,351, April 27 2014-May 2 2014.

- **The Edge Cloud** addresses edge computing specific issues by augmenting the traditional data center cloud model with service nodes placed **at the network edges**.

- Architecture of the Edge Cloud and its implementation as an overlay hybrid cloud using the industry standard OpenStack cloud management framework is studied.

Dew Computing (1/1)

- **Dew computing** is a new computing paradigm appeared after the widely acceptance of **cloud computing**.
- Dew computing key features:

  1) local computers provide rich micro-services independent of cloud services;

  2) these micro services inherently collaborate with cloud services.

- **Wang, Y.** (2015) “Cloud-dew architecture”, *Int. J. Cloud Computing*, Vol. 4, No. 3, pp.199-210.
- **Wang, Y.** “The Initial Definition of Dew Computing”. Dew Computing Research.

  http://dewcomputing.org/index.php/2015/11/10/the-initial-definition-of-dew-computing/

Jungle Computing (1/2)

- **Jungle computing** is a form of high performance computing that distributes computational work across cluster, grid and cloud computing.

D. Tychalas and H. Karatza, “High Performance System based on Cloud and beyond: Jungle Computing”, Journal of Computational Science, Elsevier, 22, pp. 131-147, 2017.

Jason Maassen, et al, “Towards jungle computing with Ibis/Constellation”, in Proceedings of the 2011 workshop on Dynamic distributed data-intensive applications, programming abstractions, and systems, ACM New York, 2011.

Frank Seinstra et al, “Jungle Computing: Distributed Supercomputing Beyond Clusters, Grids, and Clouds”, in Grids, Clouds and Virtualization, Computer Communications and Networks", Springer-Verlag London Limited, 2011.

Jungle Computing (2/2)

S. Zikos and H. D. Karatza, “Allocating jobs of different priorities to a distributed system with heterogeneous resources”, in Proceedings of the 2018 International Conference on Computer, Information and Telecommunication Systems (CITS 2018), Colmar, France, 11-13 July 2018, pp. 60-64.

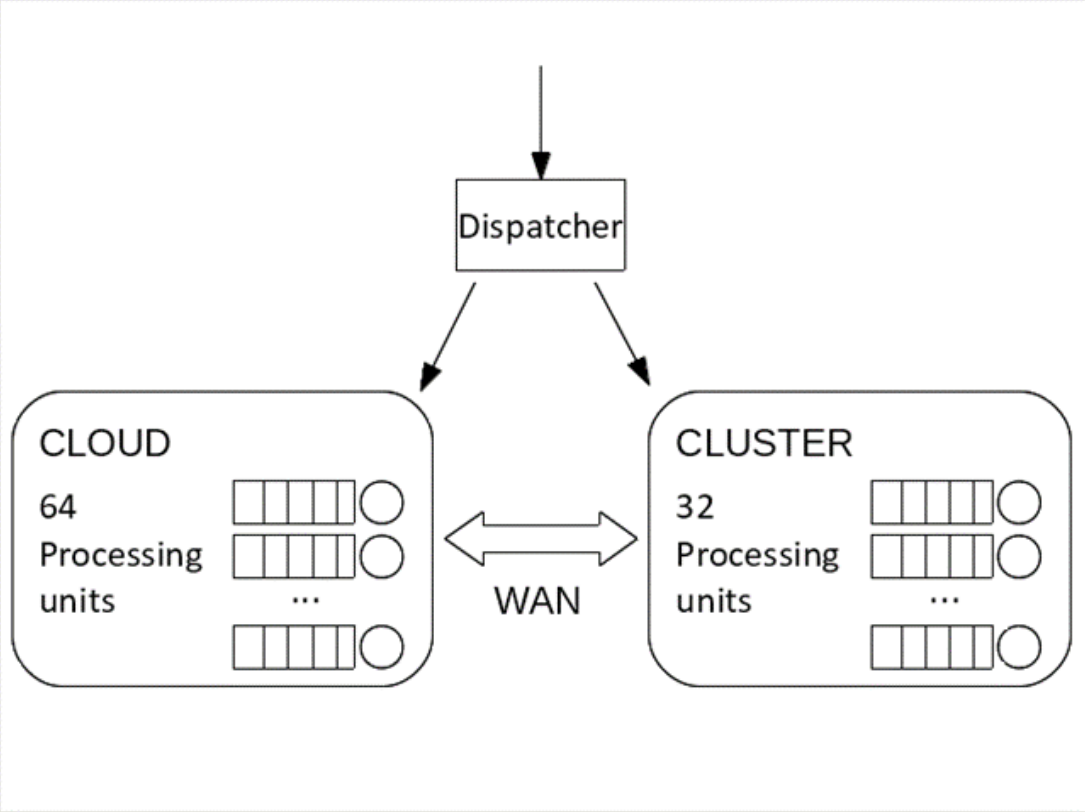


Fig. 19. A jungle computing system model.

Dust Computing (1/1) - Smart Dust

- *“Smart Dust are tiny computers that are designed to function together as a wireless sensor network. Currently, Smart Dust particles are quite small - about the size of a grain of rice. But, in the near future, it's expected that the technology will advance so that each sensor is as small as a dust particle or a grain of sand.*
- *The basic idea behind Smart Dust is that you could drop thousands of tiny sensors over a landscape and create an ad hoc wireless sensor network where there isn't one already”.*

Source:
PennState
https://www.e-education.psu.edu/geog583/node/77

Conclusions and Future Directions (1/4)

- Advances in **processing**, **communication** and **systems/middleware** technologies had as a result:

  -- new paradigms and platforms for computing.

- The **Cloud computing paradigm** promises:

  -- on-demand scalability, reliability, and cost-effective high-performance.

Conclusions and Future Directions (2/4)

- Our perception of computing is changing constantly (**Mobile Cloud Computing, Fog, Edge, Dew Computing**).
- 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 (3/4)

- **Cloud computing** offers great opportunities for scientists, organizations and enterprises.
- **Simulation modeling** is a valuable cost effective tool to efficiently examine **the costs and risks** associated with **moving real-time applications to the Cloud**.
- By using simulation, **risks can be avoided** and the possible benefits of moving applications to the Cloud **can be in advance estimated.**

Conclusions and Future Directions (4/4)

- However, **multiple issues have to be addressed** before Clouds become viable for large scale real-time distributed processing.

- **Security** and **availability** 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 !

We need secure, available and energy efficient clouds !