ENSIAS, Mohammed V University of Rabat

Doctoral Thesis Defense

Efficient Management of Big Data Applications Deployed in the Cloud Computing

February 06, 2024

Presented by BOUHOUCH Laila

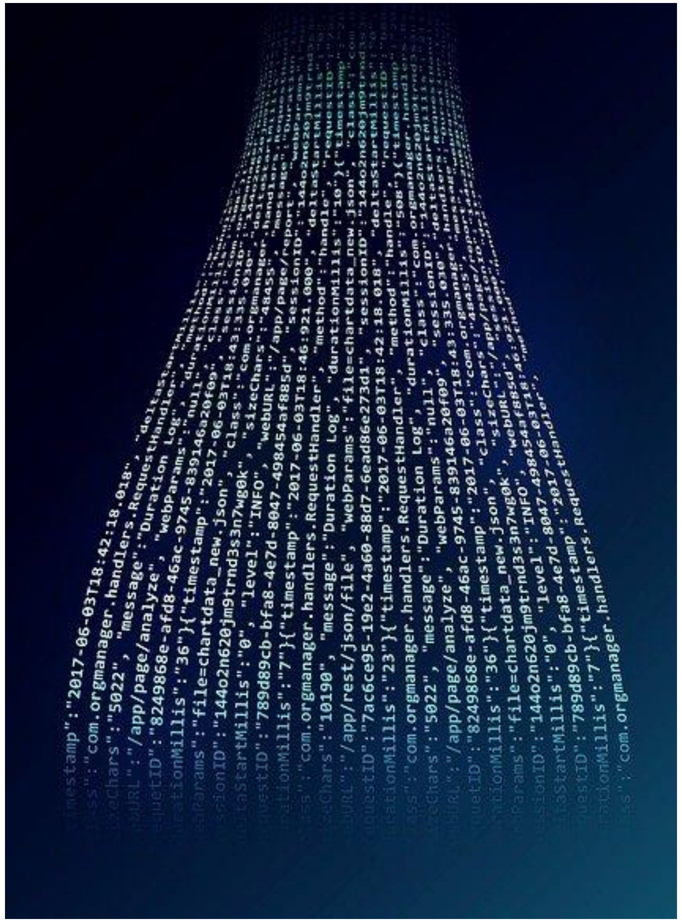
Jury Members

Pr. Mohamed Dafir ECH-CHERIF El KETTANI	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Président**
Pr. Mostapha ZBAKH	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Thesis supervisor**
Pr. Claude TADONKI	Professeur des universités, Mines ParisTech, Ecole de Mines de Paris, France	**Thesis co-supervisor**
Pr. Christophe CERIN	Professeur des universités, Université Sorbonne Paris Nord, France	**Dissertation reporter**
Pr. Mohsine ELEULDJ	PES, EMI, Université Mohammed V de Rabat, Maroc	**Dissertation reporter**
Pr. Mahmoud NASSAR	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation reporter**
Pr. Faissal EL BOUANANI	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation examiner**
Pr. Houda BENBRAHIM	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation examiner**

Outline of the talk

- Introduction
- Context and motivation
- Problematic of the thesis
- Thesis contributions and results analysis
- Conclusion and perspectives

Big Data



- Rapid increase of data collected by high-performance heterogeneous applications, commonly known as Big Data applications.
- Processing and analyzing Big Data pose challenges for traditional techniques.
- New techniques and technologies are required to deal with Big Data.

Introduction

Big Data

Scale of Data

The amount of data generated grows exponentially, ranging from terabytes to zettabytes.

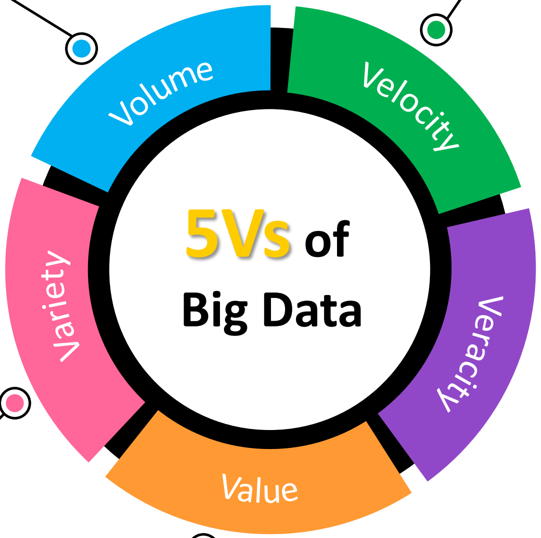
2012	2013	2015	2020	2025
2.5 Exa	4.4 Zeta	8 Zeta	40 Zeta	163 Zeta

Analysis of Streaming Data

The rate at which data is generated and delivered.

In April 2023, there were **5,18 billion** internet users worldwide.

Modern cars have around **100 sensors** that monitor things like fuel level and tire pressure.



Uncertainty of Data

- Quality
- Accuracy
- Trustworthiness
- Consistency
- Reliability

Different Forms of Data

- Structured Data
- Semi-Structured Data (JSON, XML)
- Unstructured Data (TXT)

Value of Data

Extract value and insights from available data.

Insights — Decision-Making — Efficiency

Cloud computing

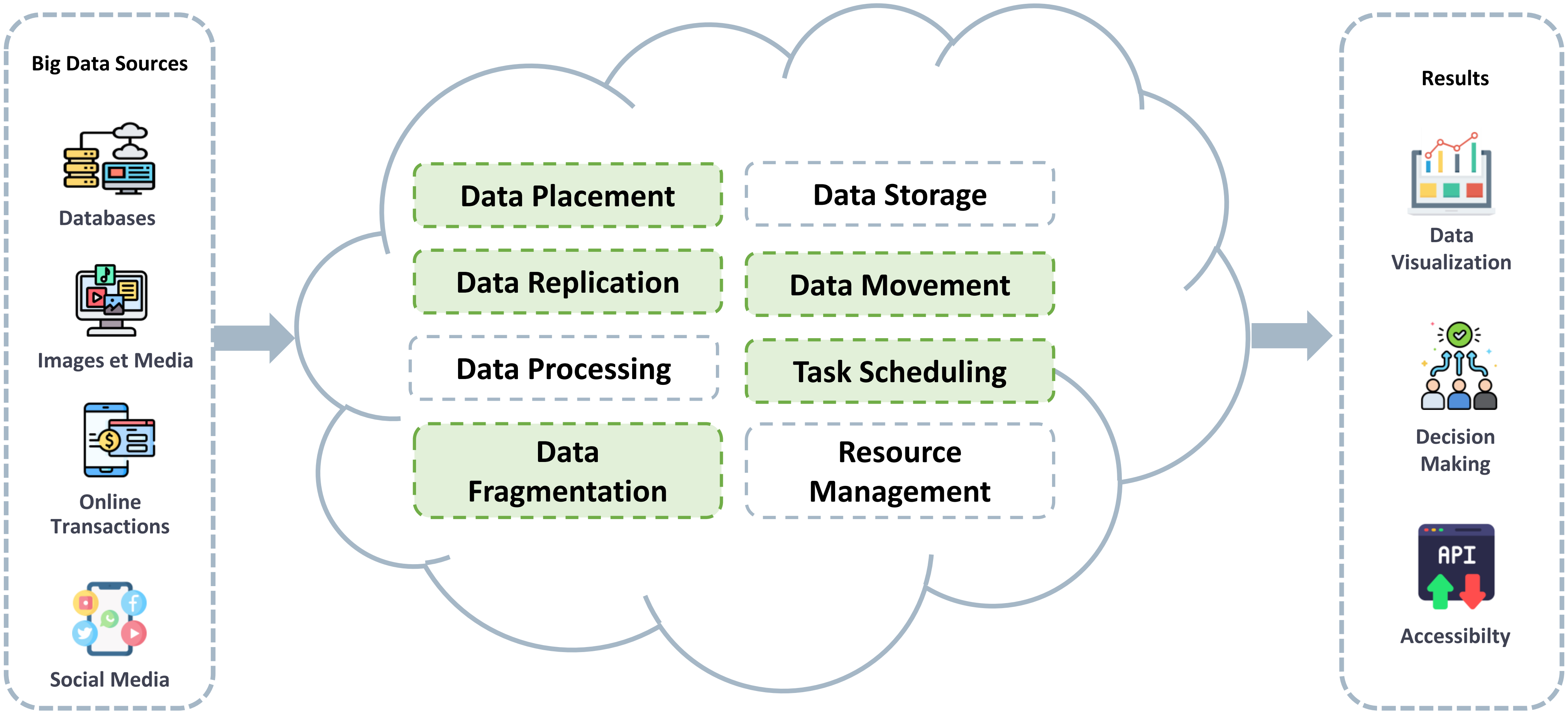
- Resources like servers, storage, databases, networks, software are provided to users as services.
- Users can access and utilize them on-demand and pay only for their usage.
- Users can store and access data and applications over the internet instead of on local hardware which reduces the need for physical infrastructure.

Cloud Computing

Actors	Service Models	Deployement Models	Characteristics	Performance Metrics
Provider	IaaS	Private	On-demand self-service	Response Time
Customer	PaaS	Public	Broad network access	Throughput
	SaaS	Hybrid	Geo-Distribution	Availability
			Rapid Elasticity	Cost
			Measured Services	Energy
			Virtualization	Security

⇒ **Cloud computing** is key to efficiently storing, processing, and analyzing **Big Data**, offering scalability, flexibility, and cost-effectiveness.

Context and motivation



- Save costs/energy
- Optimize resources
- Provide scalability
- Ensure fault tolerance
- Ensure data security
- Improve system performance

Simulation tools

- Find cloud simulators able to simulate data-related aspects.
- This significant challenge restricts the development and optimization of data management algorithms.

Data Placement

- Efficient data placement in cloud computing is challenging, especially with geographically dispersed data and tasks.
- The heterogeneity of the system intensifies the problem.
- Inefficient data placement can lead to increased execution time and higher monetary costs.

Problematic of the thesis

Data Replication

- Balance redundancy necessity with the costs of managing multiple copies is challenging.
- Managing data copies becomes complex with rising storage and bandwidth costs.

Task scheduling in the context of data

- Handle data locality, remote access, and dynamic resource allocation are the challenges.
- Frequent changes in resources availability make it challenging to ensure data proximity to tasks.

Contribution 1

Cloudsim Extensions: Modeling and Simulation of Data Migration in Distributed Data Centers

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki. 2020. “**Data Migration: Cloudsim Extension**”, In Proceedings of the 3rd International Conference on Big Data Research (ICBDR ’19). Association for Computing Machinery, New York, NY, USA, 177–181. https://doi.org/10.1145/3372454.3372472

- **Laila Bouhouch**, Mostapha Zbakh and Claude Tadonki. 2023. “**DFMCloudsim: an extension of cloudsim for modeling and simulation of data fragments migration over distributed data centers**”, International Journal of Computers and Applications. https://doi.org/10.1080/1206212X.2023.2277554

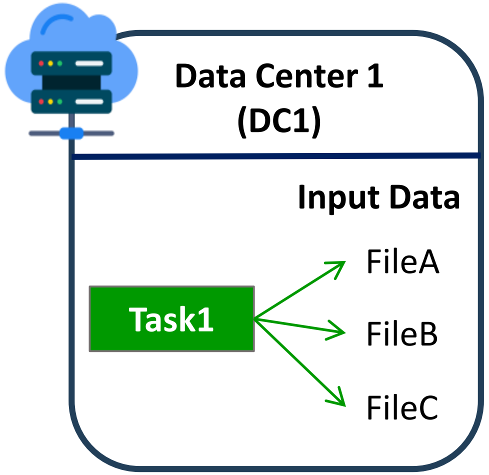
Objective

Introduce Two Extensions for Cloudsim: DMCloudsim and DFMCloudsim.

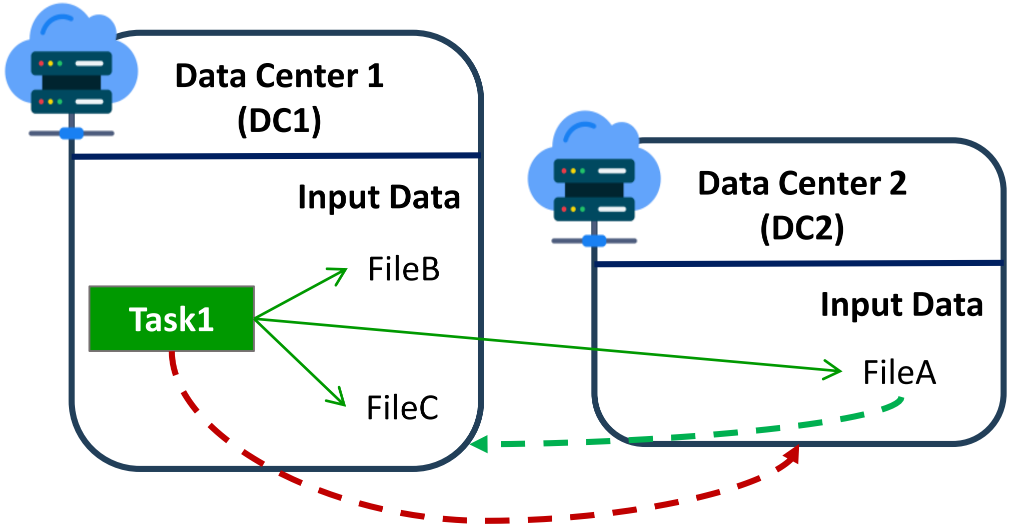
- Simulate and evaluate the processes of Data Migration and Data Fragmentation before moving to real Clouds.
- Users can easily implement their own Data Migration and Data Fragmentation algorithms.
- Enhance Data Management in Geographically Distributed Cloud Systems.

Problematic

Ideal scenario for big data applications in cloud



Realistic scenario for big data applications in cloud



- ***Hardware constraints***
- ***Software constraints***

Migrate data towards tasks to accomplish their execution.

Motivation

Cloudsim

- Open Source
- Well-known
- Stable Code
- Dynamic Cloud model

Lack of various aspects (edge, workflows, data, …)

What is it?

- An open source framework coded and designed in Java language.

For what?

- A powerful simulator for modeling and analyzing cloud computing environments before moving on to real clouds.
- Experiment and evaluate new algorithms.

How?

- Configure your own infrastructure: multiple data centers, hosts, virtual machines, task and more.
- Communication among these components is through events.

Benefits?

- Not costly such as real infrastructure.
- Repeat the evaluation, especially, with specific conditions each time.

Motivation

Cloudsim

- Open Source
- Well-known
- Stable Code
- Dynamic Cloud model

Lack of various aspects (edge, workflows, data, …)

Extensions

- EdgeCloudSim
- WorkflowSim
- CloudReports
- CloudsimDisk

Lack of data-related aspects and data migration simulation

EdgeCloudSim

- For edge computing environments.
- Modeling and simulating edge nodes, IoT devices, and their interactions.

WorkflowSim

- Creation and execution of workflows in a Cloud environment.
- Offers various task scheduling algorithms.

CloudReports

- Provides a graphical interface.
- Generates detailed reports (resource usage costs, execution time, and energy consumption).

CloudsimDisk

- Models and simulates energy consumption during the interaction of tasks with storages.

DMCloudsim - Motivation

Cloudsim

- Open Source
- Well-known
- Stable Code
- Dynamic Cloud model

Lack of various aspects (edge, workflows, data, …)

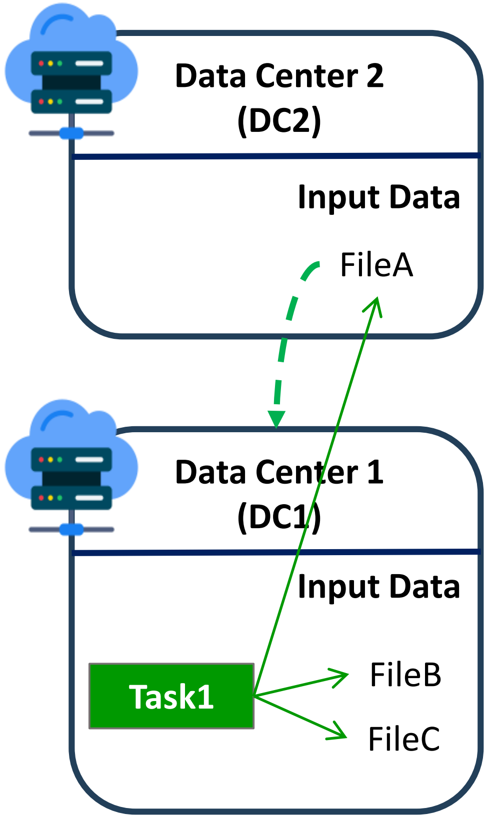
Extensions

- EdgeCloudSim
- WorkflowSim
- CloudReports
- CloudsimDisk

Lack of data-related aspects and data migration simulation

DMCloudsim*

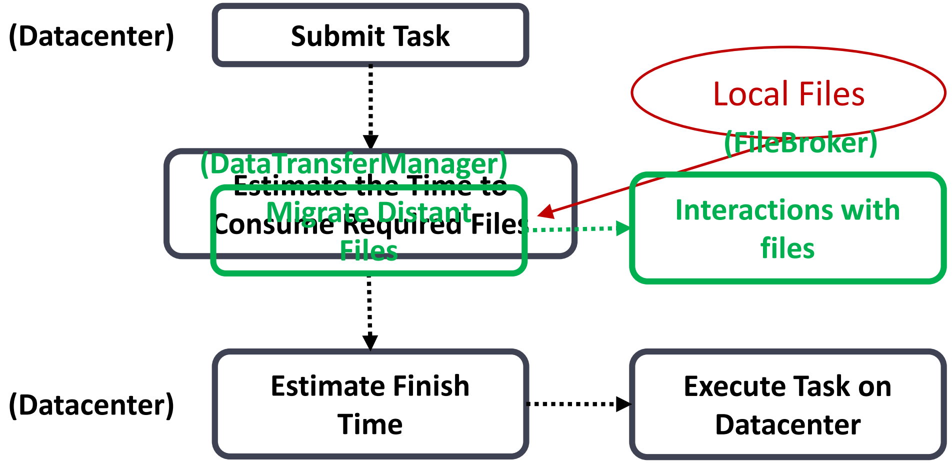
- File Migration
- Evaluate file migration algorithms



Ensure that required files are available locally.
Evaluate different data migration algorithms, thereby optimizing task execution.
Provide detailed informations : number of migrated files, estimated migration time, execution time.

* « *DMCloudsim* » extension refers to « *Data Migration Cloudsim* » extension

DMCloudsim - Extension Design



.core
- ExtDatacenter.java
- ExtHdd.java
- ExtCloudlet.java
- ExtFile.java
- ExtCloudsimTags.java

.examples
- Example.java
- Simulator.java
- Helper.java

.dataMigration
- DataTransferManager.java
- FileBroker.java
- DataMigrationStrategy.java

.distribution
- ExtCloudletDistribution.java

.util
- WriteToLogFile.java
- WriteToResultFile.java

New Packages

Extended Packages

***Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki. 2020. “**Data Migration: Cloudsim Extension**”, In Proceedings of the 3rd International Conference on Big Data Research (ICBDR ’19). Association for Computing Machinery, New York, NY, USA, 177–181.* https://doi.org/10.1145/3372454.3372472

DFMCloudsim - Motivation

Cloudsim

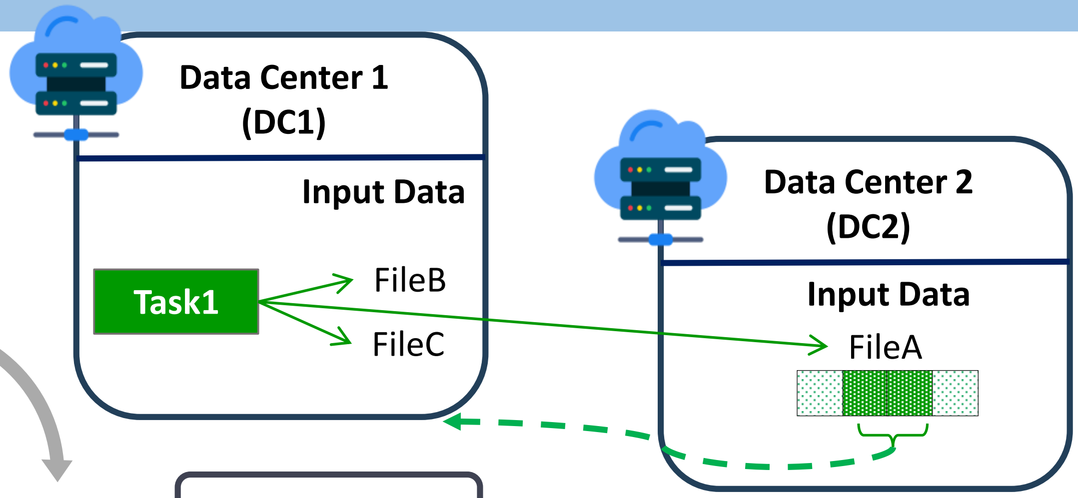
- Open Source
- Well-known
- Stable Code
- Dynamic Cloud model

Lack of various aspects (edge, workflows, data, …)

Extensions

- EdgeCloudSim
- WorkflowSim
- CloudReports
- CloudsimDisk

Lack of data-related aspects and data operation simulation



DMCloudsim

- File Migration

Process entire file, increasing data transfer overhead

DFMCloudsim*

- File Fragmentation
- Fragment Migration

Fragment files using fragmentation strategy
Ensure that required fragement are available locally using migration strategy.
Evaluate different data fragmentation and migration algorithms.
Provide detailed informations : fragments, estimated migration time, execution time.

* « *DFMCloudsim* » extension refers to « *Data Fragments Migration Cloudsim* » extension

Cloudsim Extensions: Modeling and Simulation of Data Migration in Distributed Data Centers

DFMCloudsim - Extension Design

.core
- MyDatacenter.java
- MyCloudlet.java
- MyCloudsimTags.java

.examples
- MyFragmentationTest.java
- Simulator.java
- Helper.java

.distribution
- ExtCloudletDistribution.java

.util
- WriteToLogFile.java
- WriteToResultFile.java

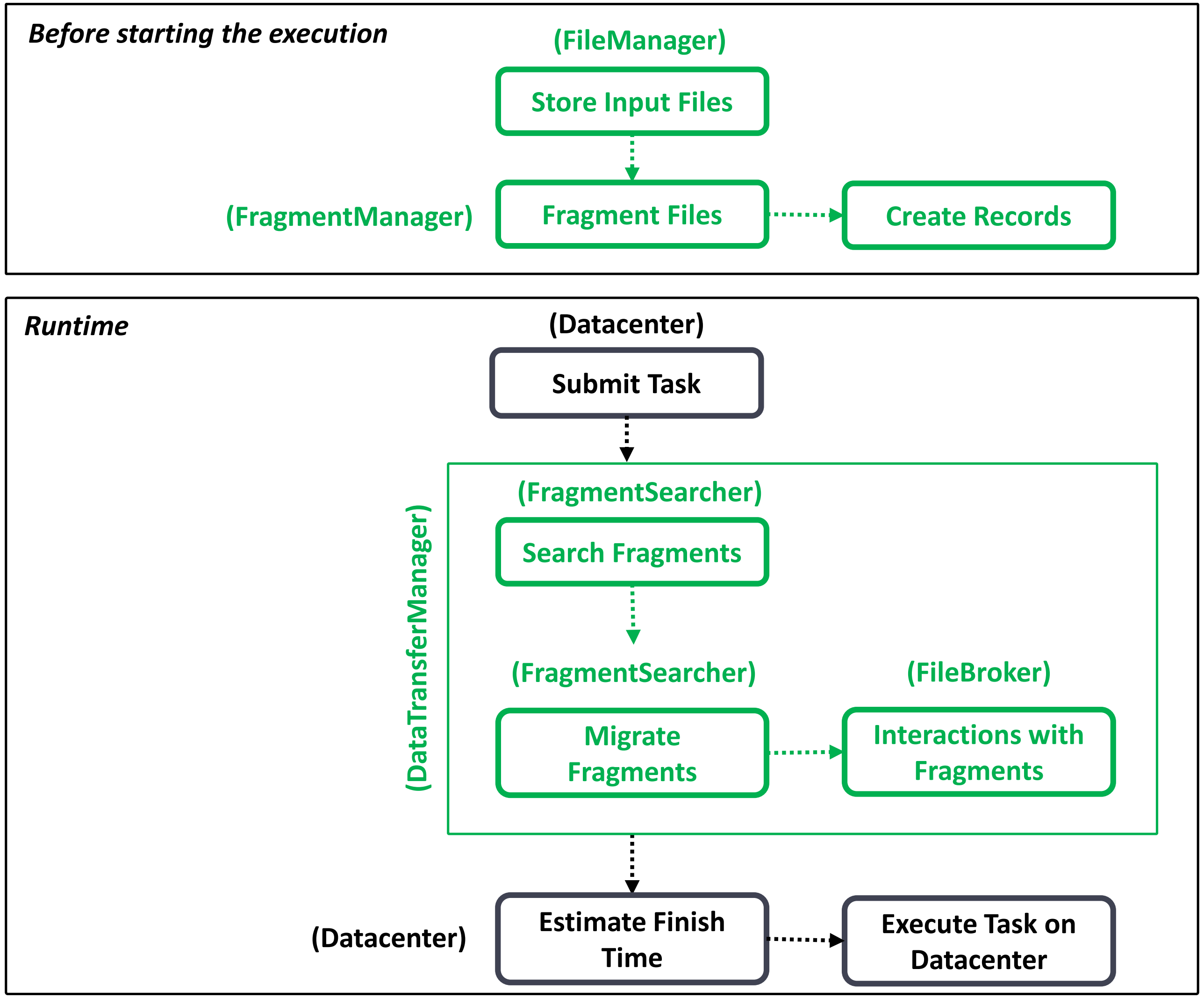
.fileManager
- MyHdd.java
- MyFile.java
- InputFileAdder.java

.fragmentManager
- InputFormat.java
- TextFileFormat.java
- InputBlock.java
- FileBlock.java
- Record.java
- TextRecord.java
- MyBlock.java

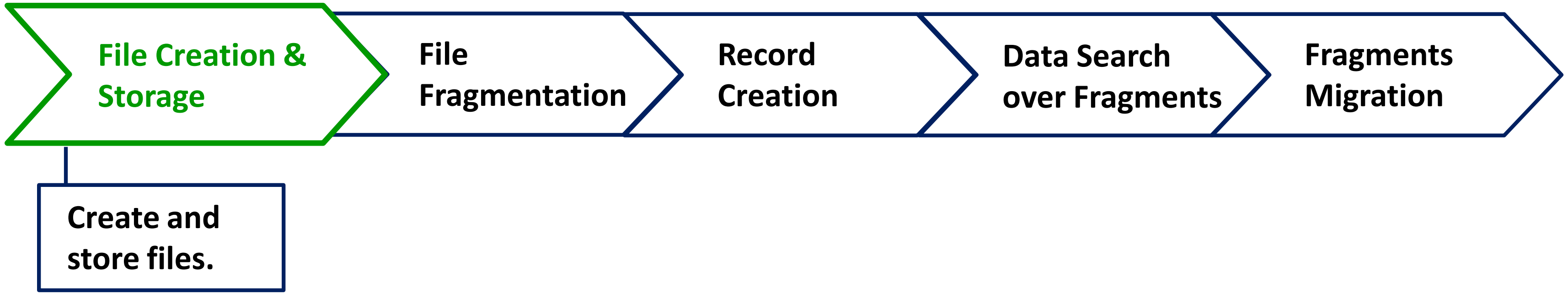
.dataTransfer
- FragmentSearcher.java
- DataTransferManager.java
- FileBroker.java

Extended Packages

New Packages



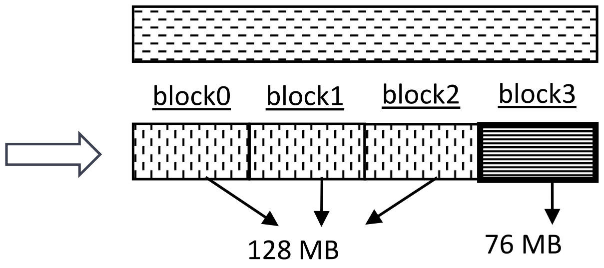
DFMCloudsim - Module Processing



DFMCloudsim - Module Processing

Split TextFileFormat files using a block size.

fileName = ‘‘fileA’’, fileSize = 460 MB, blockSize = 128 MB



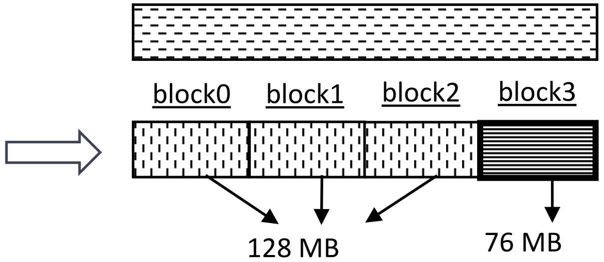
File Creation & Storage → **File Fragmentation** → **Record Creation** → **Data Search over Fragments** → **Fragments Migration**

Create and store files.

DFMCloudsim - Module Processing

Split TextFileFormat files using a block size.

fileName = ‘‘fileA’’, fileSize = 460 MB, blockSize = 128 MB



File Creation & Storage → **File Fragmentation** → **Record Creation** → **Data Search over Fragments** → **Fragments Migration**

Create and store files.

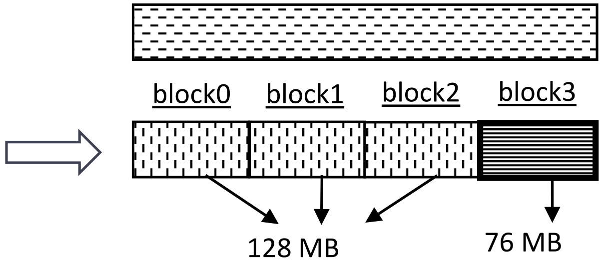
Create records from the generated blocks to facilitate data search.

block0	This is a simple example \n This is an example of record,
block1	splitted into three blocks three blocks to better understand the
block2	creation of record process \n But not the second record \n
block3	…

DFMCloudsim - Module Processing

Split TextFileFormat files using a block size.

fileName = ''fileA'', fileSize = 460 MB, blockSize = 128 MB



Find the required task datasets within the produced blocks.

But not the second record \n => block2

This is an example of record, splitted into three blocks three blocks to better understand the creation of record process \n => block0, block1 and block2

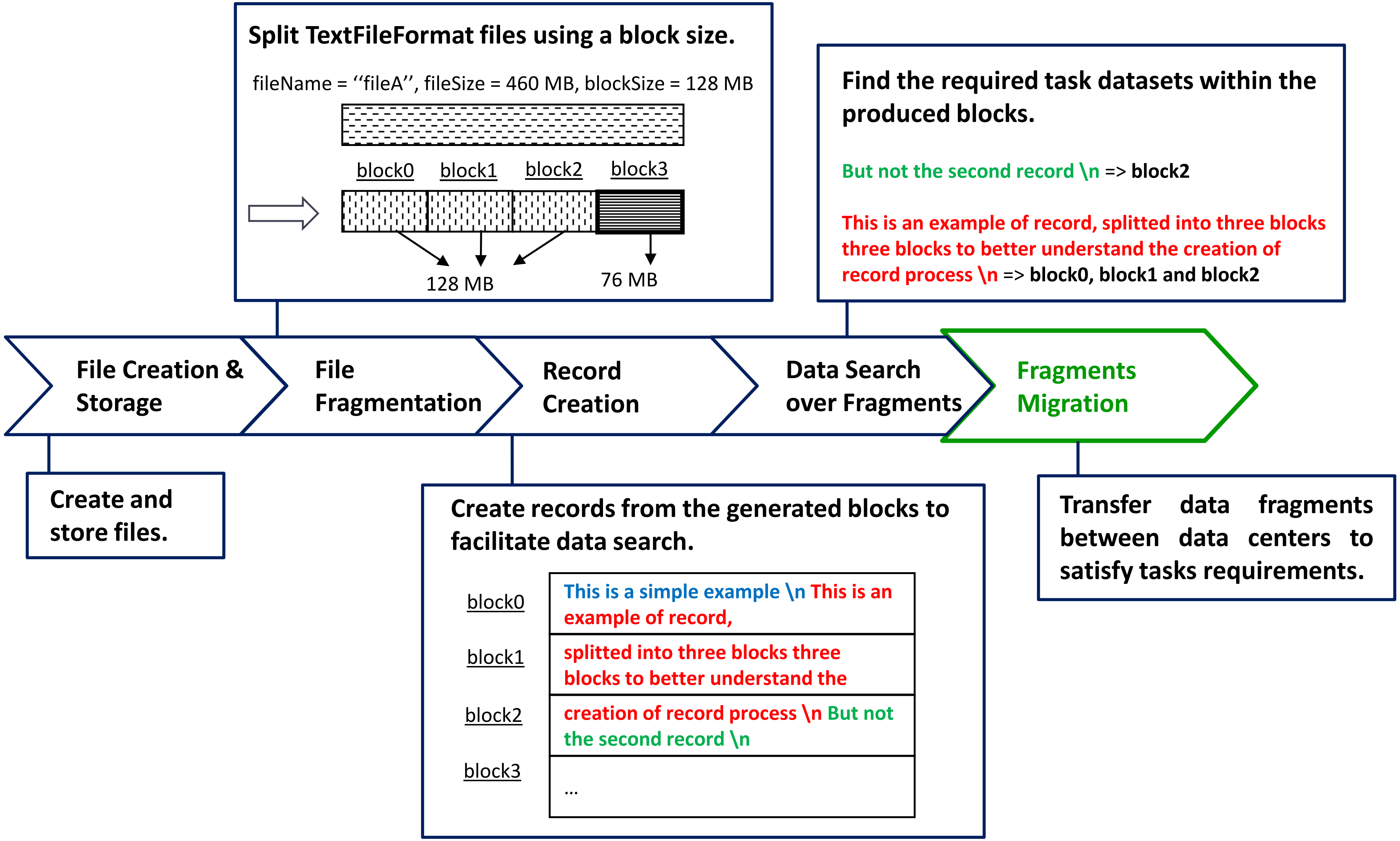
File Creation & Storage → **File Fragmentation** → **Record Creation** → **Data Search over Fragments** → **Fragments Migration**

Create and store files.

Create records from the generated blocks to facilitate data search.

block0	**This is a simple example \n This is an example of record,**
block1	**splitted into three blocks three blocks to better understand the**
block2	**creation of record process \n But not the second record \n**
block3	…

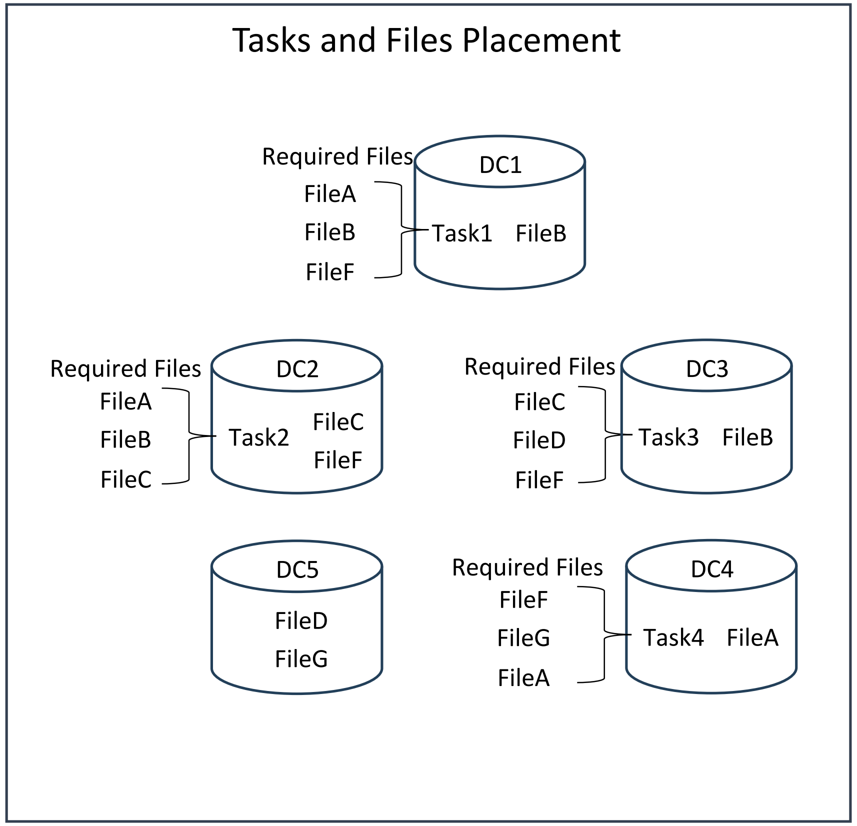
DFMCloudsim - Module Processing



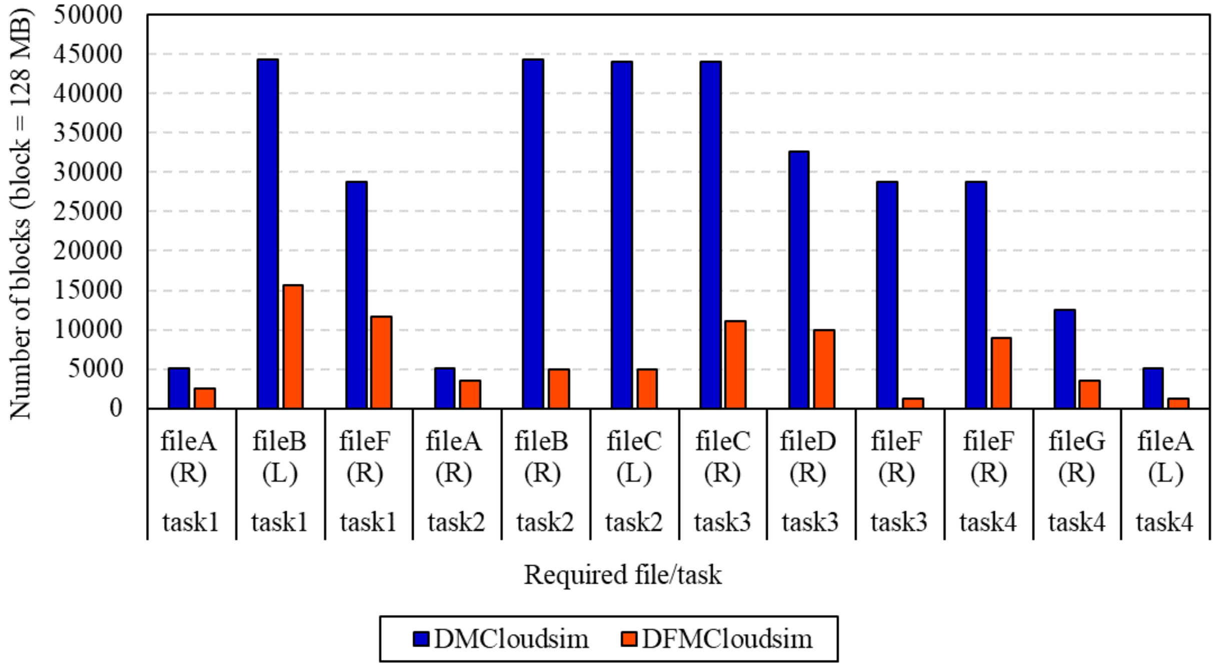
Experimental Setup

- Validate the proposed features.
- Show how our extensions effectively provide detailed insights into each mechanism.
- Compare between the two extensions : DMCloudsim et DFMCloudsim.

Parameter	Value
Number of data centers	5
Number of HDD/DC	2
Read/Write speed	[160 Mb/s, 216 Mb/s]
Intra-datacenter bandwidth	[5 Mb/s, 60 Mb/s]
Avg. Intra-datacenter delay	1,1 s
Number of tasks	4
Avg. Task's length	2500
Number of files	26
File size	[600 GB, 6000 GB]
Required files/task	3
Block size	128 MB

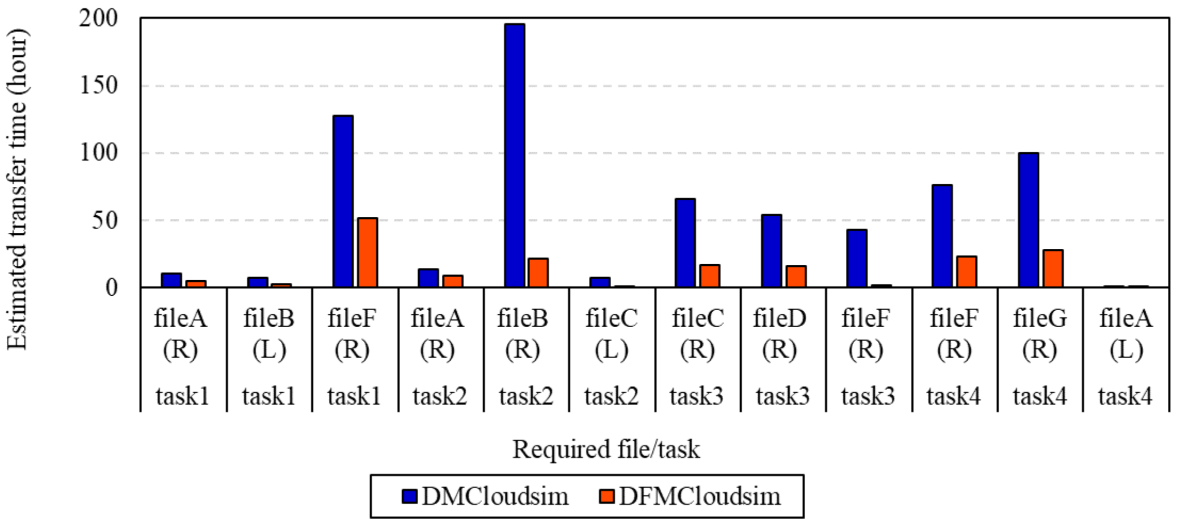


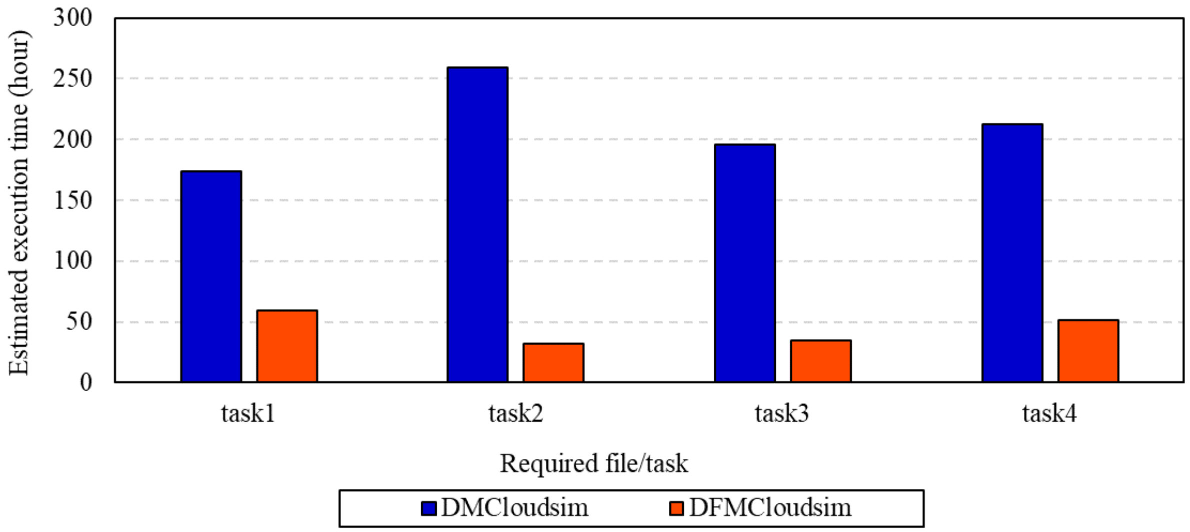
Experimental Results



- Detailed informations on the number and size of blocks/files manipulated for each task.
- DFMCloudsim reduces the number of blocks by 58%.

Experimental Results



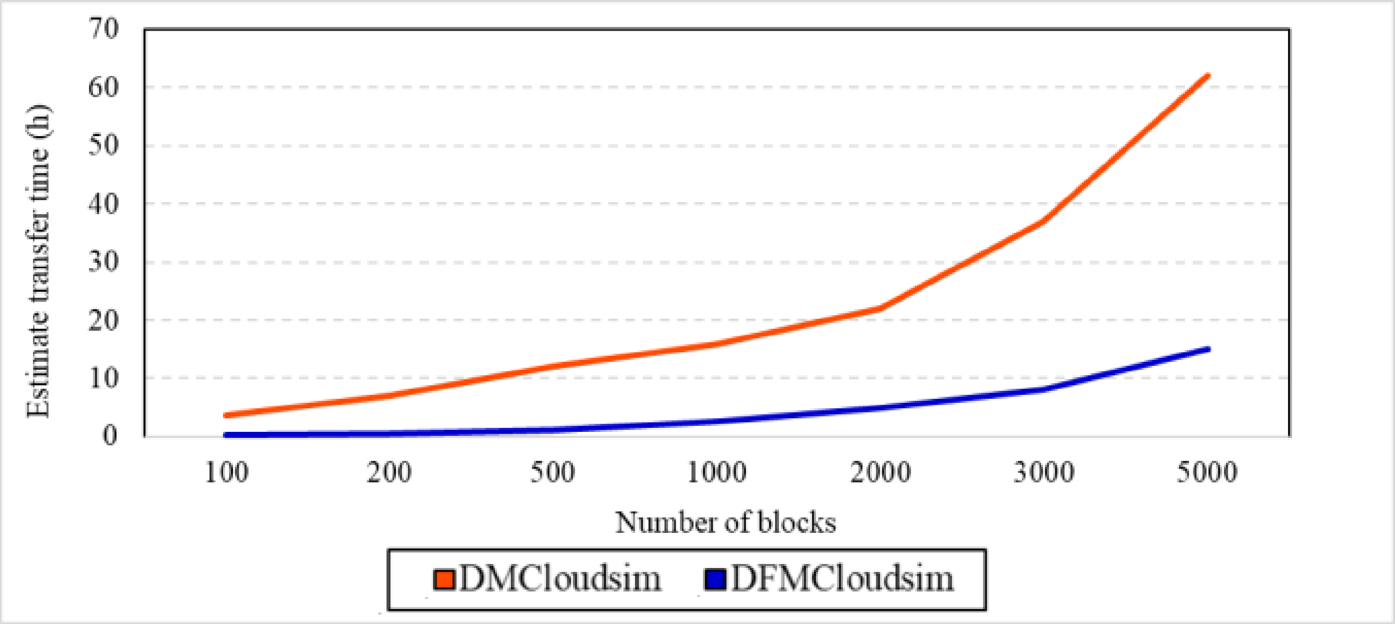


- Detailed informations on the estimated transfer time of each file per task.
- DFMCloudsim reduces the transfer time.

- Detailed informations on the estimated execution time of each task.
- DFMCloudsim reduces the overall execution time by 74%.

- Tasks don't require entire files for execution.
- Block management reduces data transfer, so leads to minimized execution times.
- DFMCloudsim optimizes execution time by handling only necessary data.

Experimental Results



- Vary the total number of required blocks for tasks.
- Investigate the impact of the number of block on the transfer time.
- DFMCloudsim outperforms due to tasks not needing the entire file.
- As the number of blocks increases, the transfer time grows.
  - ⇒ Importance of efficient data fragmentation and migration strategies that can be implemented using DMCloudsim/DFMCloudsim

Conclusion

- DMCloudsim and DFMCloudsim simulate fragmentation and migration mechanisms.
- Provide insights of each mechanism.
- Allow users to implement and evaluate fragmentation and migration algorithms for better efficiency.
- Evaluation shows fragmentation minimizes data transfer and improves execution times.
- Some limitations of our work:
    - Static fragmentation may not suit dynamic scenarios.
    - Need for more efficient fragmentation and migration algorithms.

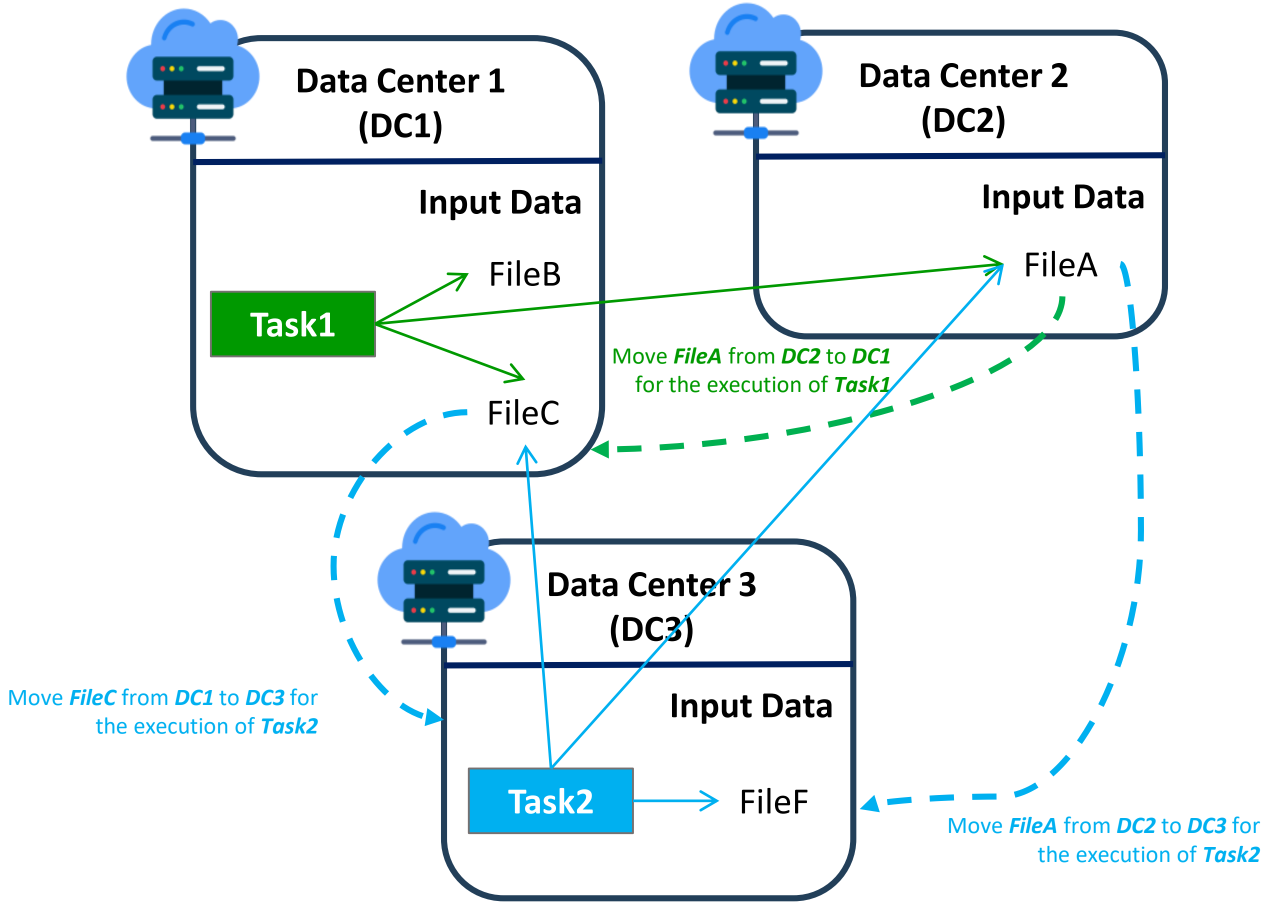
Dynamic Data Replication and Placement Strategy in Geo-graphically Distributed Data Centers

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki. "**A Big Data Placement Strategy in Geographically Distributed Datacenters**" 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakesh, Morocco, 2020, pp. 1-9. https://doi.org/10.1109/CloudTech49835.2020.9365881

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki. "**Dynamic data replication and placement strategy in geographically distributed data centers**". Concurrency Computat Pract Exper. 2023; 35(14):e6858. https://doi.org/10.1002/cpe.6858

Problematic

Realistic scenario for data-intensive applications in cloud



- Task execution may require multiple remote data.
- Necessity to migrate remote data.
- Migration of large data impacts execution time.
- Inefficient management can lead to additional costs.

Motivation & Objective

Existing strategies

Initial Data Placement

Aim to place data in the appropriate data centers before starting the execution

Reduce only the number of data movements does not necessarily reduce the time data movement.

Data Replication

Distribute multiple copies of data across data centers to enhance data availability and fault-tolerance.

Replicas generated from data movement during task execution is not considered

Proposed strategy : Combine & Improve

Reduce data migrations time

Minimize total task execution time and financial costs

Build Time Phase - Data Placement Strategy - Step 1. Compute Transfer Time Matrix

Build-time: Initial data placement

Step 1. Compute Transfer Time Matrix

Step 2. Apply Greedy Algorithm

- Compute the transfer time matrix **τ**.
- τ_{ij} is the total time to transfer data d_i from location m_j to all data centers hosting tasks that require d_i.
- τ_{ij} is computed based on the volume of the datasets, the read/write speeds of the disks, and the network bandwidth.

$$\tau(i,j) \leftarrow \tau(i,j) + \left(\frac{1}{r_j} + \frac{1}{w_{\alpha_k}} + \frac{1}{b_{j\alpha_k}}\right) \times v_i$$

- Sort elements of each row **τ**(i,:) in ascending order.

$$[\sigma(i,:)] \leftarrow sort(\tau(i,:))$$

$$\boldsymbol{\tau_i \Rightarrow \sigma_i}$$

where $\boldsymbol{\sigma_i(j) = k}$: index of the data center corresponding to the k^{th} element of $\boldsymbol{\tau(i,:)}$

- The goal is to get a better data center for each dataset, starting with the one that yields the smallest migration time.

Algorithm 3 Data Placement - Calculate data tra

Input:

$M = (m_1, m_2, \ldots, m_p)$: Set of data centers
$T = (t_1, t_2, \ldots, t_n)$: Set of tasks
$D = (d_1, d_2, \ldots, d_m)$: Set of datasets
$V = (v_1, v_2, \ldots, v_m)$: Volumes of the datasets
$F = (f_{ij})$: Matrix of relationship between datasets and tasks
$R = (r_1, r_2, \ldots, r_p)$: Read speed of data centers
$W = (w_1, w_2, \ldots, w_p)$: Write speed of data centers
$B = (b_{ij})$: Matrix of bandwidth between data centers m_i and m_j
$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$: Index of the data center where task t_i is assigned to (i.e. m_{α_i})

Output:

$\tau = (\tau_{ij})$: Matrix of datasets transfer time from a fixed source to all target data centers
$\sigma = (\sigma_{ij})$: Matrix τ sorted row by row (the aim to facilitate the selection of the best values at the row level)

// Computation of τ matrix
for $i \in D$ **do**
  for $j \in M$ **do**
    $\tau(i,j) \leftarrow 0$
    for $k \in T$ **do**
      if $f(i,k) == 1$ **then**
        if $j \neq \alpha_k$ **then**
          $\tau(i,j) \leftarrow \tau(i,j) + (\frac{1}{r_j} + \frac{1}{w_{\alpha_k}} + \frac{1}{b_{j\alpha_k}}) \times v_i$
        end if *// Otherwise, the dataset and the task are within the same data center*
      end if
    end for
  end for
  // Sort elements of each row $\tau(i,:)$ in ascending order
  $[\sigma(i,:)] \leftarrow sort(\tau(i,:))$ *// $\sigma_i(j) = k$, the k^{th} element of $\tau(i,:)$*
end for

Build Time Phase - Data Placement Strategy - Step 2. Apply Greedy Algorithm

Build-time: Initial data placement

Step 1. Compute Transfer Time Matrix

↓

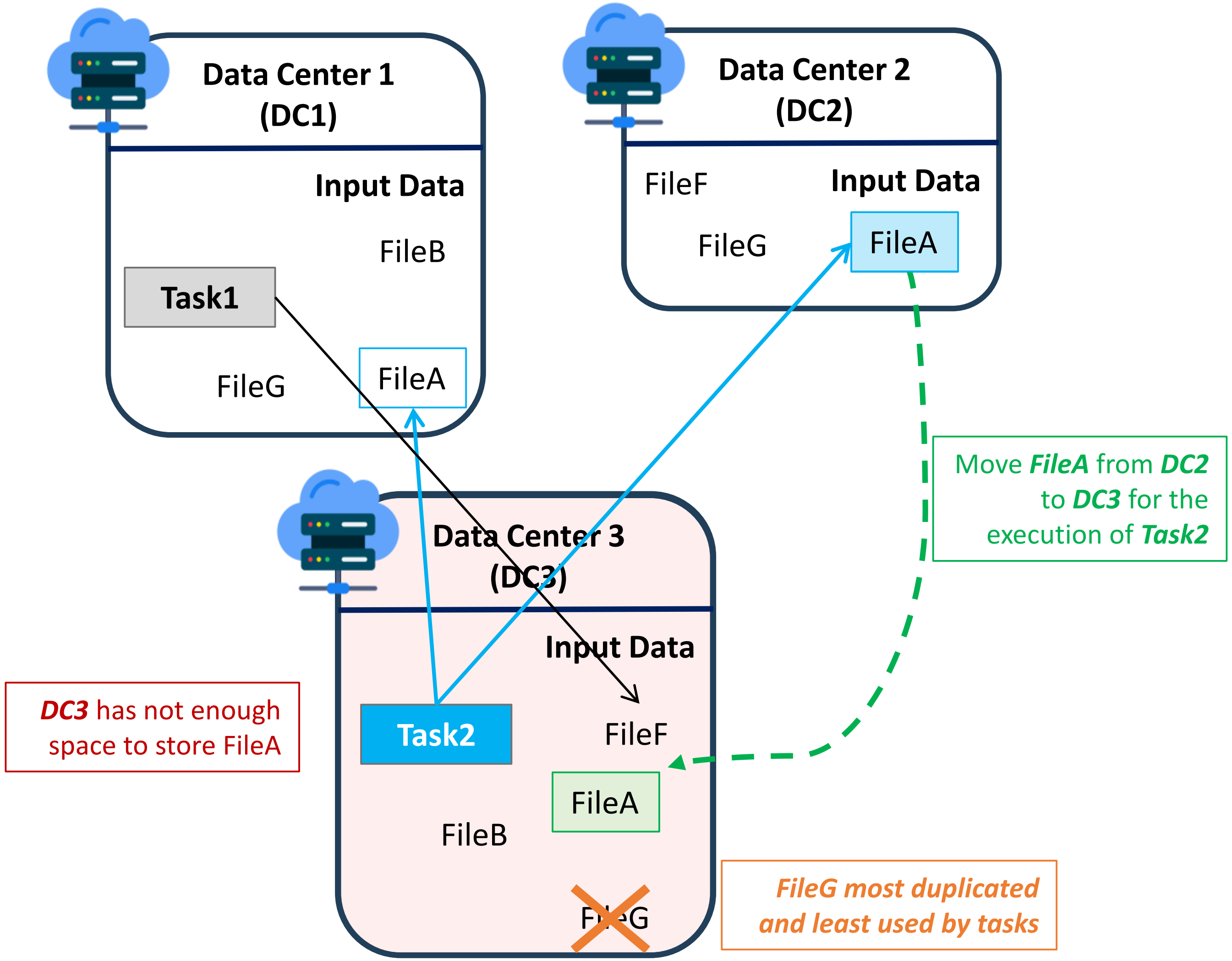
Step 2. Apply Greedy Algorithm

- A greedy algorithm is applied based on the previously calculated transfer time to assign each dataset to the optimal data center.
- The goal is to achieve an efficient data placement that will reduce the total migration time.

$h \leftarrow 1$
$s \leftarrow 0$
for $(i \leftarrow 1$ *to* $m)$ **do**
  $placed[i] \leftarrow 0$
end for
while $(s < m$ && $h \leq p)$ ***do***
  $max \leftarrow -1$
  $k \leftarrow -1$
  for $(i \leftarrow 1$ *to* $m)$ ***do***
    if $(placed[i] \neq 1)$ ***then***
      $\ell \leftarrow \sigma[i, h]$ // id of our h^{th} acceptable data center choice for dataset d_i
      if $((\tau(i,\ell) > max)$ && $(c_\ell - v_i > 0))$ ***then***
        $max \leftarrow \tau(i,\ell)$
        $k \leftarrow i$ // if needed because of space constraint, we will relocate d_k
      end if
    end if
  end for
  if $(k \neq -1)$ ***then***
    $\ell \leftarrow \sigma[k, h]$ // id of the acceptable data center for dataset d_k
    $placed[k] \leftarrow 1$
    $\phi[k] \leftarrow \ell$ // d_k will be stored in data center m_ℓ
    $c_\ell \leftarrow c_\ell - v_k$ // update of the capacity of data center m_ℓ as it receives dataset k
    $s{+}{+}$
  else
    $h{+}{+}$
  end if
end while
if $(s = m)$ ***then***
  Data placed successfully!
else
  Problem with data placement!
end if

Runtime Phase - Data Replication Strategy

- Replication strategy is performed iteratively at runtime phase.
- Takes advantage of the data movement that occurs during tasks execution.
- Aims to manage multiple copies of the datasets.



Decide which replicas to delete from ***DC3*** considering:

a. Minimum number of replicas that should exist in the entire system to ensure availability
b. Dependency between datasets and tasks not yet executed.

Experimental Setup

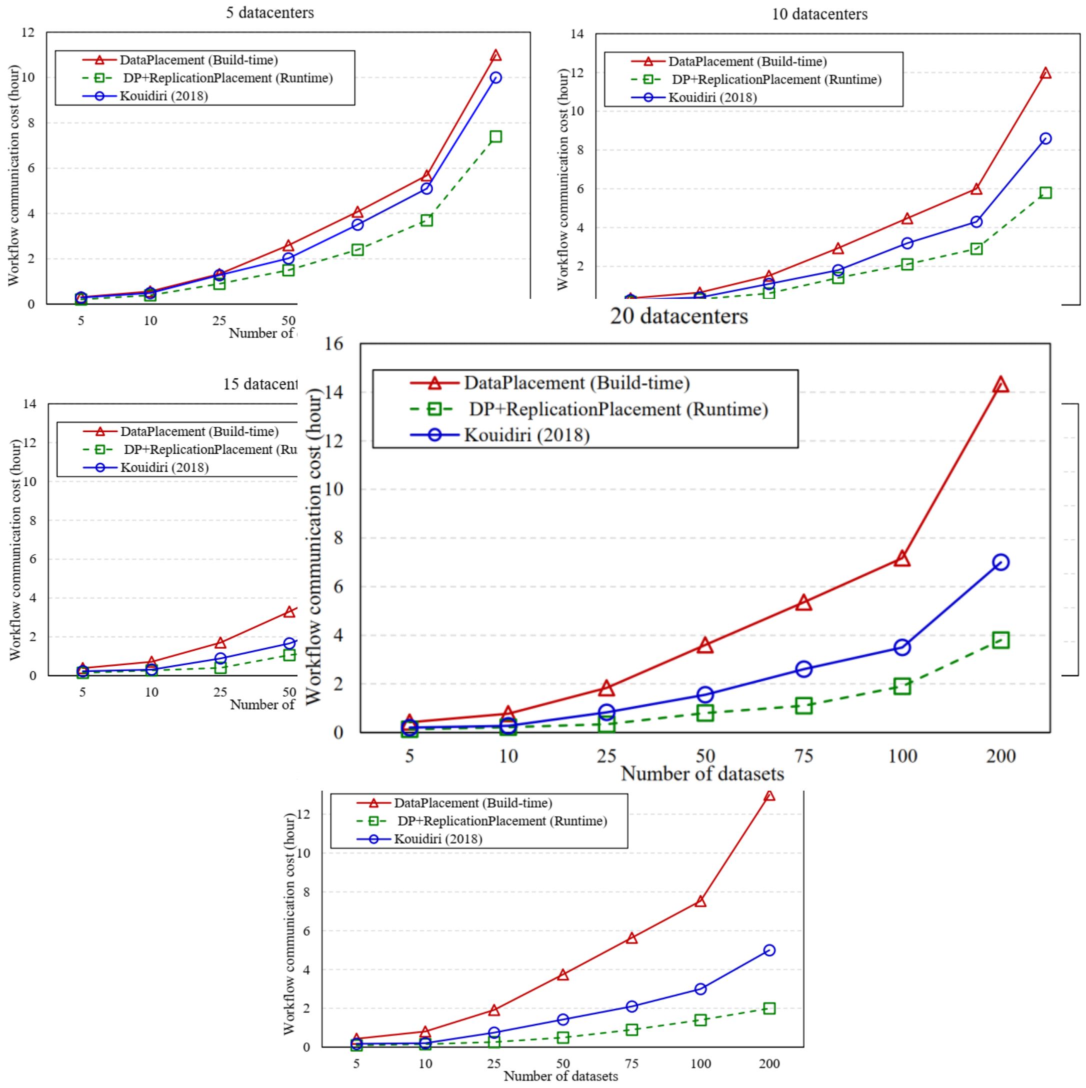
- **Build-time strategy:** the datasets are (efficiently) stored before runtime and immediately deleted once consumed.
- **Kouidri's strategy:** suggests keeping the most used replicas, storing them where there is enough space.

- Simulate with DMCloudsim.

Components	Values
# of tasks	1000
# of datasets	[5, 10, 25, 50, 75, 100]
Dataset size	[1TB - 100TB]
# of data centers	[5, 10, 15, 20, 25]
data center capacity	[1PB - 25PB]
Storage cost	\$0.1 per GB
Transfer cost	\$0.05 per GB
Penalty cost	\$0.01 per violation

- Evaluation metric : Total execution time of all tasks (WCC)

Experimental Results



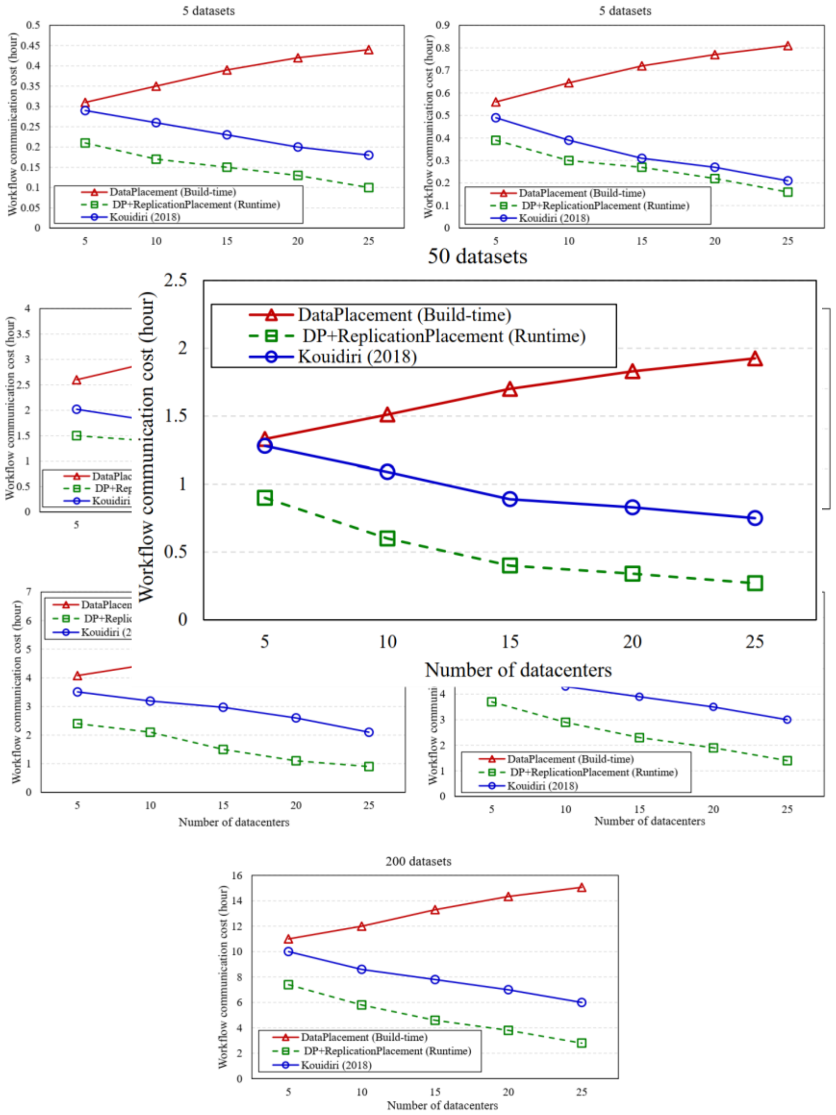
- Fix the number of data centers on 5, 10, 15, 20, 25.
- Vary the number of datasets from 5 to 200 while estimating WCC.

- WCC keeps increasing with the number of datasets.
- Our combined strategy outperforms both other strategies.

- For example, in the case of 20 data centers:
    - Reduction in WCC of 48.02%/Kouidri and 75.31%/no-replication.
    - For the timings with 5 to 200 datasets, our strategy yields a variance of 3.67 hours, while the Kouidri gives 6.8 and no-replication give 13.92 hours.

⟹ Our proposed strategy is always better regarding the reduction of migration cost and execution time.

Experimental Results



- Fix the number of datasets on 5, 10, 25, 50, 75, 100, and 200.
- Vary the number of data centers from 5 to 25 while estimating WCC.

- With the number of data, WCC increases for the no-replication strategy while it decreases with Kouidri and our approach.

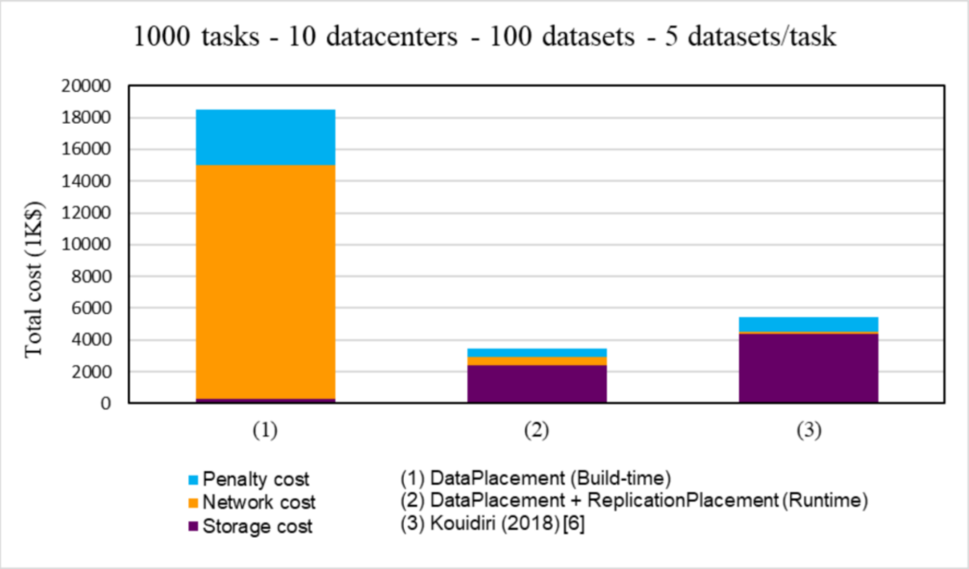
- For example, in the case of 50 datasets:
  - Our approach gives a mean of 1.05 hours **vs** 1.7 and 3.23 hours / the Koudiri's and no-replication algorithm respectively.
  - Reduction of 37.82% and 67.51% respectively.

- The results also show an increasing improvement with more and more data centers.

Monetary Cost

- The Cloud service provider has expenses.
- During task execution, three types of costs arise when data is migrated from remote data centers:
  - P: Network usage cost during data migrations.
  - Q: Data storage cost.
  - Penalty: Penalty paid by the provider to the user for any response time delay violation.
- The sum of these three costs constitutes the economic cost $FinancialCost = Q + P + Penalty$
- Our goal is to verify whether our strategy balances the monetary cost and performance improvement.

Experimental Results



- For no-replication strategy:
  - Huge network cost
  - High SLA violations => High penalty
- Our strategy reduces the storage cost by 45% compared to Kouidri's strategy.
- Our strategy generates a very high network cost compared to Kouidri's work. **But** this can considered as a compromise to save on the storage cost.
- Our strategy reduces the total financial costs by 81.33% and 36.5% compared to no-replication and Kouidri's strategies respectively .

⇒ This cost reduction shows the efficiency of our work in improving the performance of Cloud systems.

Conclusion

- A combination of ***data placement*** and ***data replication***.
- We demonstrate better performance in minimizing time and monetary costs by reducing the total migration time of datasets between data centers during execution.
- We show benefits in re-using replicas when executing tasks.

- Some limitations of our work:
    - Static Placement: Needs of dynamic data placement techniques.
    - Data Migration: Should be enhanced.

Contribution 3

Online Task Scheduling of Big Data Applications in the Cloud Environment

- **Laila Bouhouch,** Mostapha Zbakh, and Claude Tadonki. “**Online Task Scheduling of Big Data Applications in the Cloud Environment.**” Information 2023, 14, 292. https://doi.org/10.3390/info14050292

Task Scheduling Definition

- Scheduling algorithms are a set of policies, procedures, and rules to assign resources to the tasks.
- Scheduling algorithms are available in a variety of types: Static or dynamic, online or batch, preemptive or non-preemptive.
- Scheduling methods take into consideration several performance metrics. The most common metrics are mentioned below:

  - Execution time
  - Response time
  - Throughput
  - Execution cost
  - Load balancing
  - Fault tolerance
  - SLA violation
  - Energy consumption
  - Data transfer.

$\Rightarrow$ Effective task scheduling is important and many scheduling techniques have been developed to address this challenge.

Existing Task Scheduling Techniques

Single-objective scheduling techniques:

Method	Technique	Advantages	Limitations	Parameters
First Come First Served [79]	The last arrived tasks must wa... of the ex... ones.	Simple to implement and ...	Increases the waiting time ...	-
Shortest Job First [82]	Chooses ... to be ex...		...n	Execution time. Response time.
Round Robin [81]	Circular... tasks.	... time is given to every task.	... time.	-

- Aim to minimize one parameter.
- Ineffective in multi-dimensional scheduling.
- Cannot optimize multiple parameters simultaneously.

Multi-objective scheduling techniques :

Method	Technique	Advantages	Limitations	Parameters
Shyam and Manvi [213]	VM Migration.	Maximizes resource usage while minimizing time and budget.	Needed more agents for searching the best resource.	Execution time. Makespan time. Response time. Resource utiliza-...
Wang et al. [196]	D... si...			...cost. ...
Zhao et al. [214]	E... w... ar... tr... ti...			...cost. ...tiliza- ... ...sump-
Dubey et al. [215]	Sc... ba... B...			...ime. ...tiliza-
Reddy et al. [86]	M... ti...	... processing speed, and makespan in the fitness function.		...tiliza- ... Makespan. Load balance.
Biswas et al. [83]	Dynamic round robin.	Dynamically determines Time Quantum.	Starvation not handled.	Turnaround. Waiting time.
Soltani et al.[217]	Genetic meta-heuristic.	Multi-purposed weighted genetic algorithm to enhance performances.	Data-intensive online tasks not addressed.	Response time. Waiting time. Makespan.

- Focus on resources.
- Ignore the importance of data locality.
- Effective data management becomes increasingly critical.

Data Location based scheduling techniques:

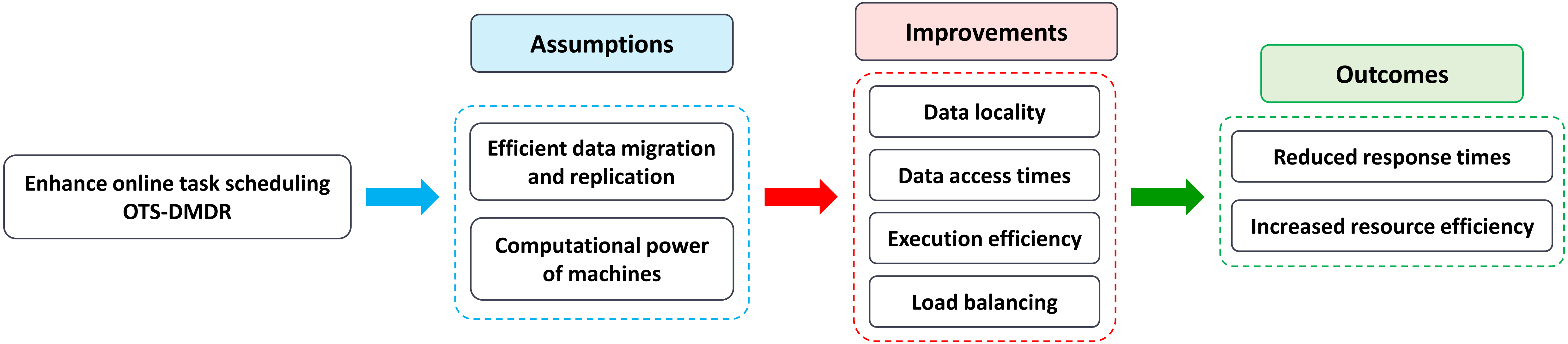
Method	Technique	Advantages	Limitations	Parameters
Delay algorithm [218]	Assign task... put data lo... lays tasks u... data is avai...			...n time.
Matchmaking algorithm [219]	Before assi... task to nod... has a fair ch... its local tasks.			...utiliza- ... ...ity.

- Tasks are allocated to data centers with most of their input data.
- Few servers are used leading to overloading.
- Longer task execution and lower throughput.

Method	Technique	Advantages	Limitations	Parameters
Li et al. [194]	Online job ... based on dat... based on a ... tween data ... and the waiti...			

- Schedules tasks sequentially one task after the other.
- Data center and data characteristics not considered when migrating data.
- Handling data replication not considered.

Objective

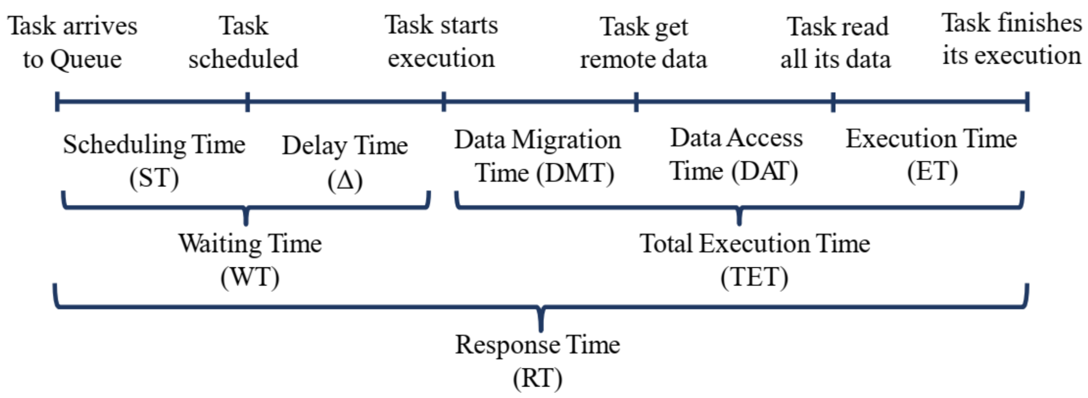

Enhance online task scheduling OTS-DMDR
Assumptions
Efficient data migration and replication
Computational power of machines
Improvements
Data locality
Data access times
Execution efficiency
Load balancing
Outcomes
Reduced response times
Increased resource efficiency

Objective Formulation

- Our objective function seeks an efficient online task scheduling that minimizes the task response time while maintaining a balanced load among the machines.
- The main idea is to select a set of tasks from the queue and schedule them to the most appropriate machine.
- Objective function of task if scheduled in machine:

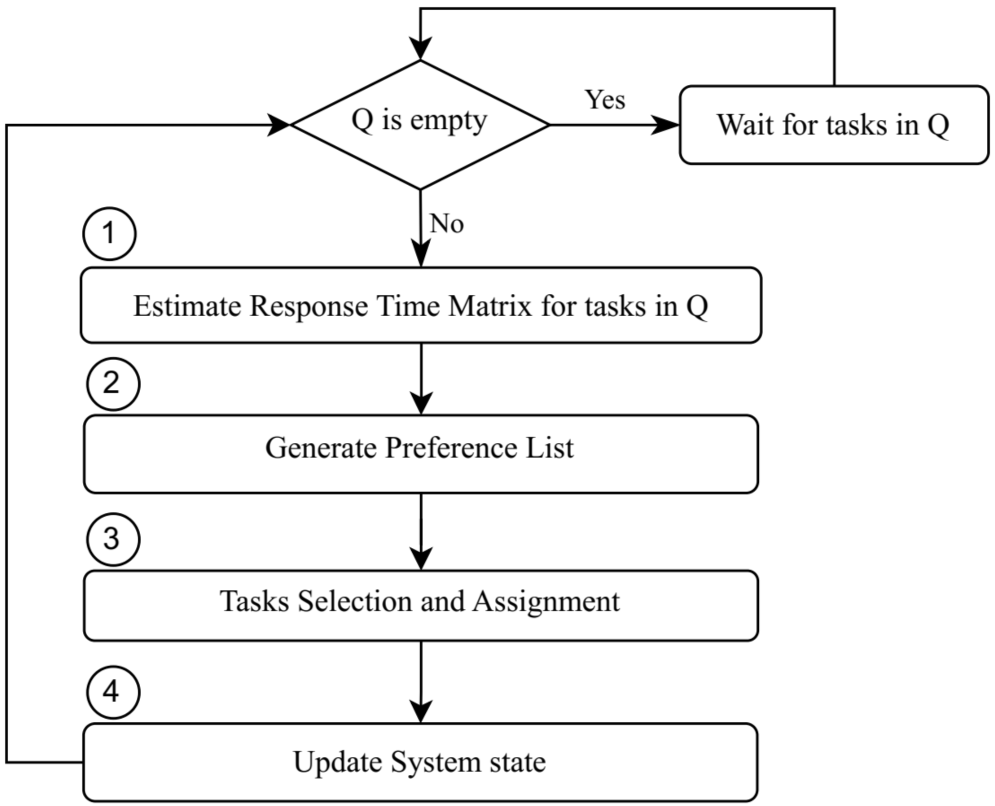
$$\min RT_{ij} = \min\,(WT_{ij} + TET_{ij})$$
$$= \min\,(ST_{ij} + \Delta_{ij} + DMT_{ij} + DAT_{ij} + ET_{ij})$$

- **Response time (RT)** is the time required for each task to complete from the time it arrives into the queue.



Proposed Approach

- The main steps of our suggested task scheduling strategy OTS-DMDR are:

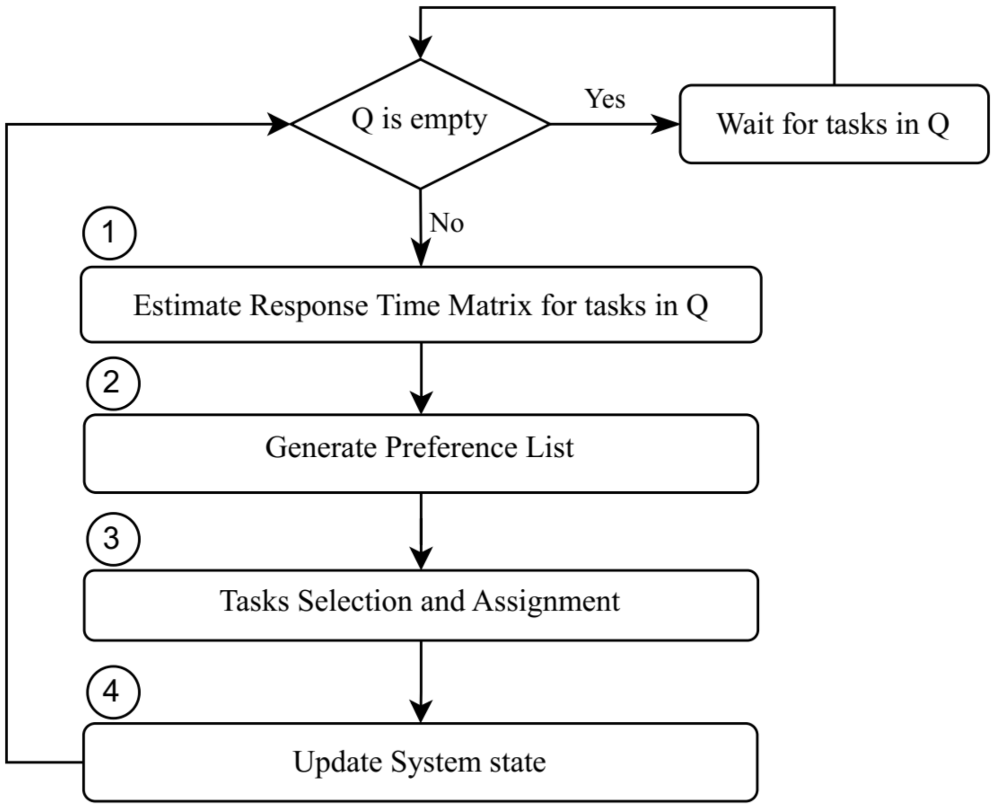


Proposed Approach

- The main steps of our suggested task scheduling strategy OTS-DMDR are:

- **Fitness**: Check if the machine can host a task.
- **Delay Time**: Waiting time of the task for the machine to be available.
- **Data Migration Time**: Time to migrate the required data of the task to the machine.
- **Local Data Access Time**: Time to consume the required data of the task in the machine.

$\Rightarrow$ Generate **Response Time (RT) matrix**, of all the tasks in the Queue for all the machines.

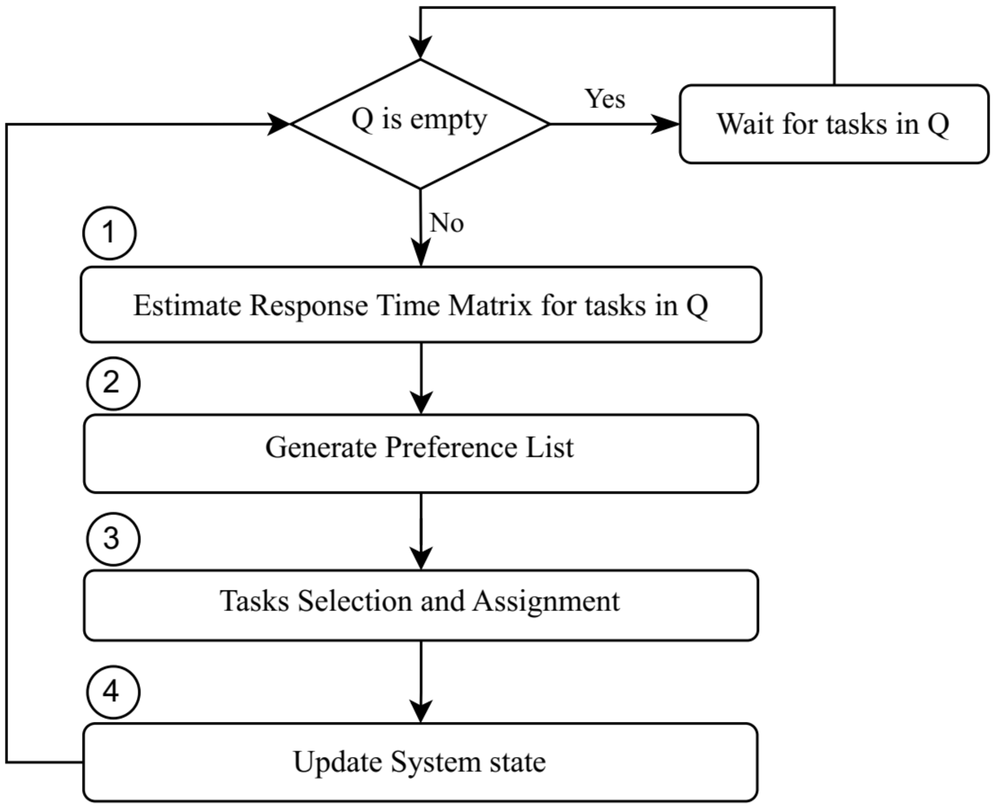


Proposed Approach

- The main steps of our suggested task scheduling strategy OTS-DMDR are:

- Sort all the element of the RT matrix in ascending order to generate Preference List (PL).
- PL provides the most suited machine for a task.

$$PL = \{pl_k\} = [(t_i, m_j, RT_{ij})]$$



Proposed Approach

- The main steps of our suggested task scheduling strategy OTS-DMDR are:

Select the set of tasks that must be scheduled in each of the machines.

<u>Iteration 1:</u>

a. Select the first element of PL, which is the lowest response time so we **assign t_2 to m_3**.

b. Mark m_3 as used and t_2 as selected.

c. Update PL.

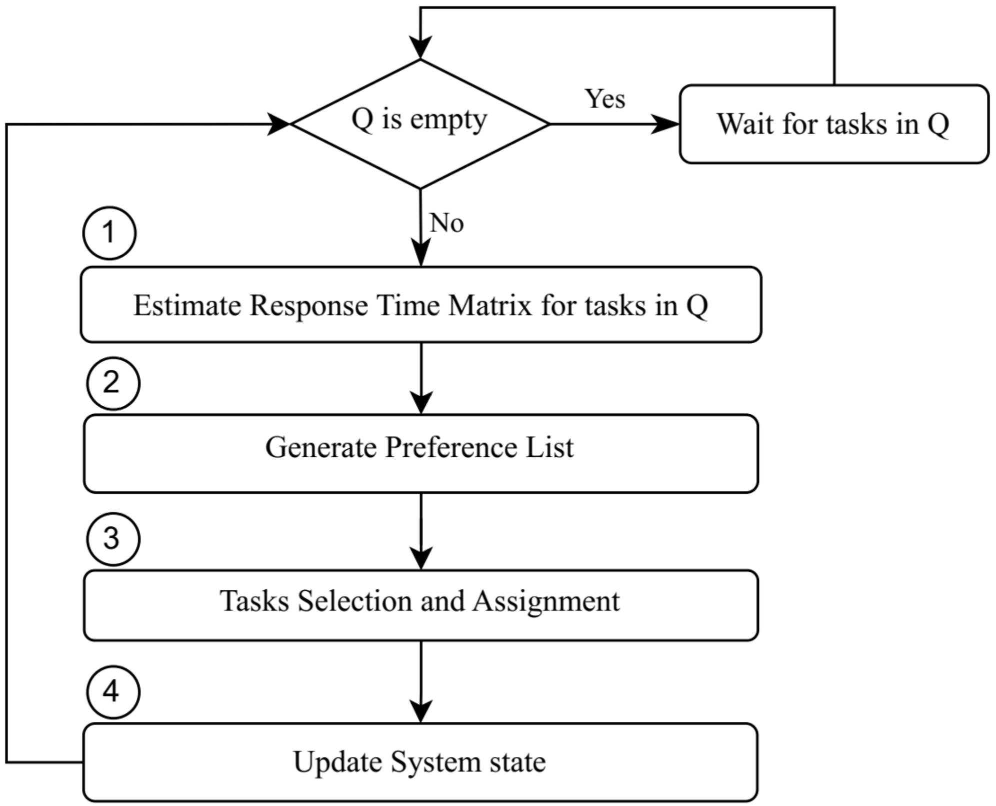
$$RT = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{bmatrix} 5 & 3 & 4 & 6 \\ 4 & 2 & 1 & 2 \\ 3 & 6 & 9 & 5 \\ 7 & 5 & 8 & 3 \\ 8 & 4 & 5 & 1 \end{bmatrix} \end{array} \rightarrow \textit{$RT_{23} = 1$ is the response time of executing t_2 in m_3}$$

$$\Longrightarrow PL = [\boxed{(t_2,m_3,1)}, (t_5,m_4,1), (t_2,\cancel{m_2},2), (t_2,\cancel{m_4},2), (t_1,m_2,3), (t_3,m_1,3), (t_4,m_4,3), (t_1,\cancel{m_3},4), (t_2,\cancel{m_1},4), (t_5,m_2,4), (t_1,m_1,5), (t_3,m_4,5), (t_4,m_2,5), (t_5,\cancel{m_3},5), (t_1,m_4,6), (t_3,m_2,6), (t_4,m_1,7), (t_4,\cancel{m_3},8), (t_5,m_1,8), (t_3,\cancel{m_3},9)]$$

<u>Iteration 2:</u> Assign t_5 to m_4

$$PL = [\boxed{(t_5,m_4,1)}, (t_1,m_2,3), (t_3,m_1,3), (t_4,\cancel{m_4},3), (t_5,\cancel{m_2},4), (t_1,m_1,5), (t_3,\cancel{m_4},5), (t_4,m_2,5), (t_1,m_4,6), (t_3,m_2,6), (t_4,m_1,7), (t_5,\cancel{m_1},8)]$$

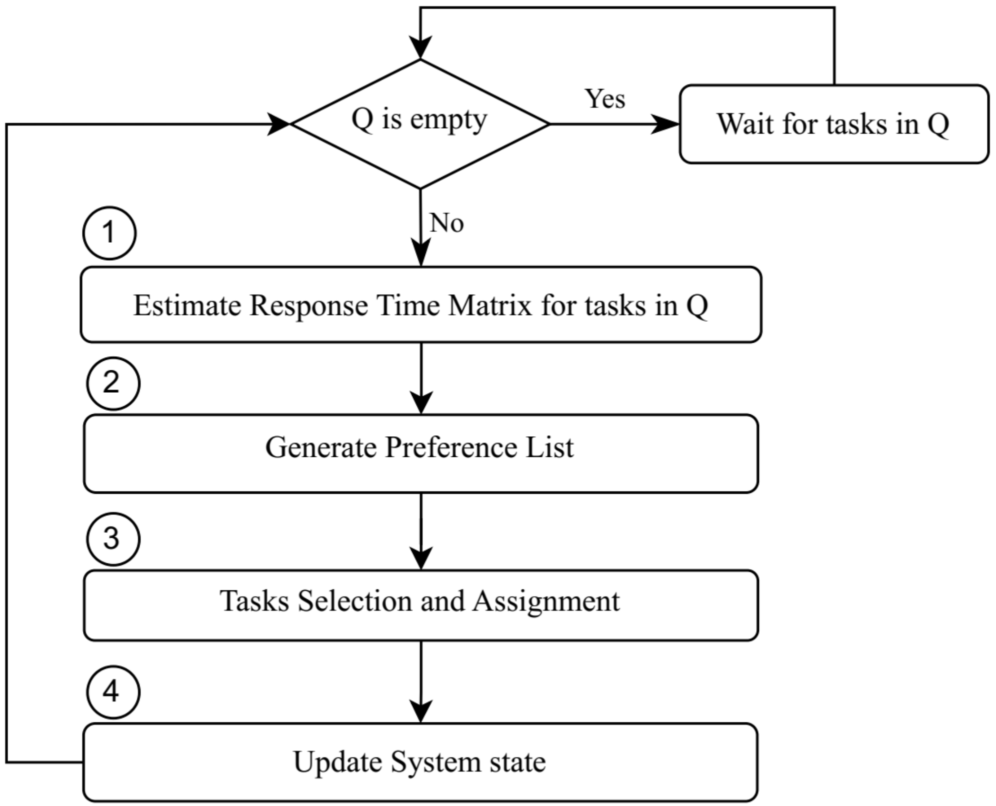
The process is repeated until PL is empty i.e. set of tasks can be scheduled in all machines.



Proposed Approach

- The main steps of our suggested task scheduling strategy OTS-DMDR are:

- Delete selected tasks from Q.
- Start executing tasks.
- Update system state.



Simulation Setup

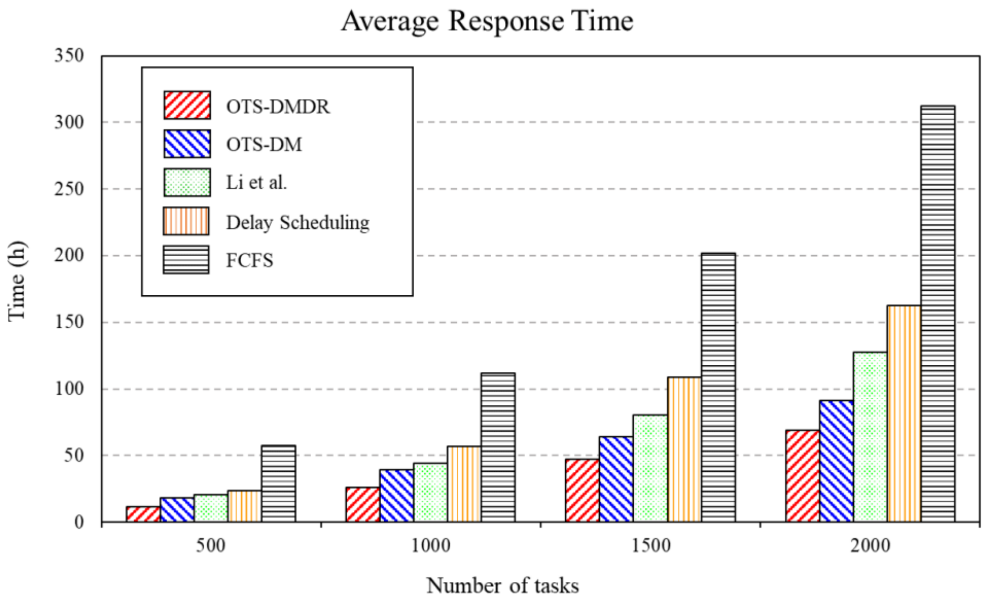
- Simulate with DMCloudsim.

Characteristic	Value
Number of machines	[5–100]
P_CPU (MIPS)	[1000–5000]
RAM (GB)	[64–2048]
Storage capacity (TB)	[1–25]
Number of tasks	[30–2000]
Size of tasks (MI)	[1000–4000]
Number of datasets	300
Size of datasets (GB)	[1–100]
Number of required datasets	[1–10]

- **FCFS**: the first arrived is the first to be scheduled.
- **Delay Scheduling**: delays the execution of a task in order to assign it to the server achieving the data locality.
- **Li et al. method**: compromises between waiting time and data migration costs.
- **Online Task Scheduling based on Data Migration (OTS-DM)**: our proposed algorithm without considering data replication.

- Evualtion metrics:
  - **Response Time (RT):** Task execution time from queue arrival to completion.
  - **Throughput**: Tasks processed per time slot.
  - **Degree of Imbalance (DI)**: Imbalance across all machines.

Simulation Results: Task Variation

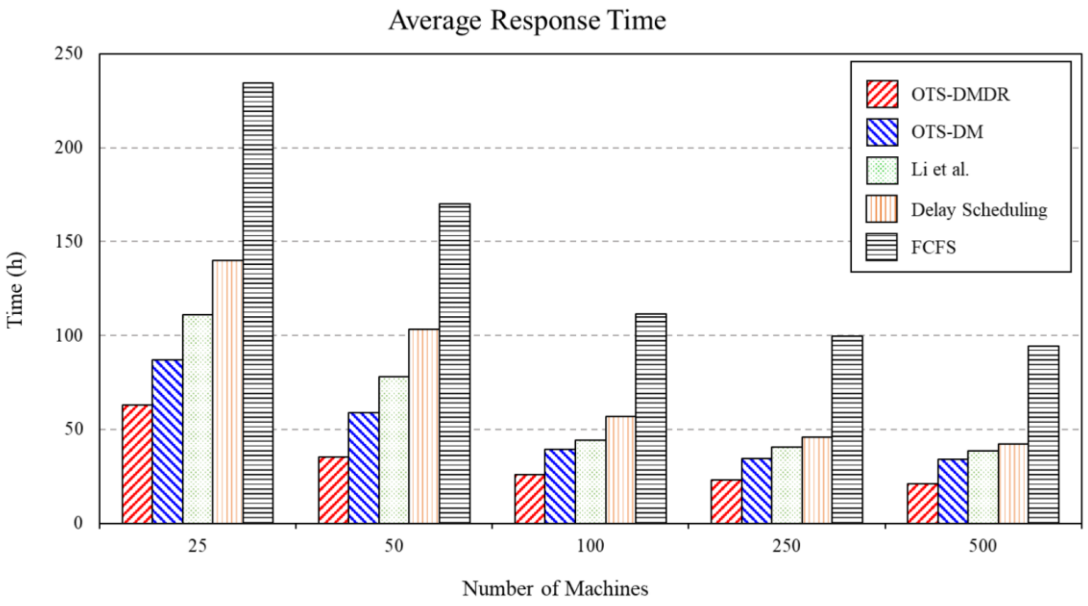


- Fix the number of machines and vary the number of incoming tasks to the queue.

- **FCFS** method exhibits poor performance for all the cases.
- **Other** methods have a competitive performance only for 500 and 1000 tasks.
- **OTS-DMDR** performed better than other algorithms, significantly reducing average response times, especially with higher number of tasks (1500 and 2000 tasks).

⇒ The trade-off between achieving *data locality*, minimizing *data migration cost*, considering *data replication*, *delay time* and *machine characteristics*, yields a better response time with lower data transfer time.

Simulation Results: Machine Variation

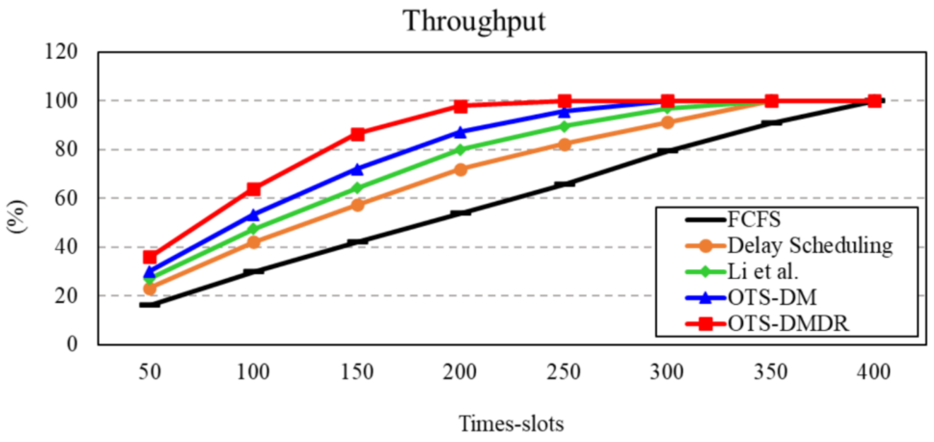


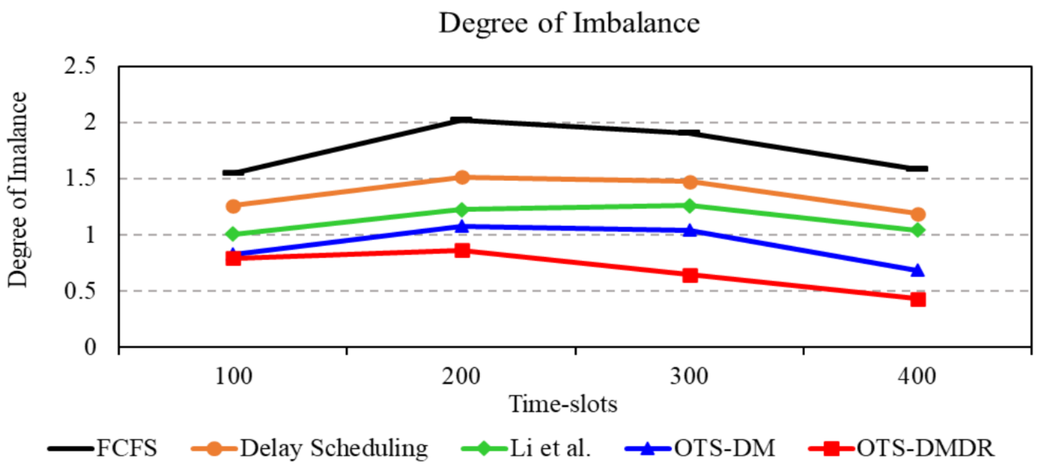
- Fix the number of tasks and vary the number of machines.

- As the number of machines increases the average response time decreases.
- OTS-DMDR gives significantly better results than all of the existing algorithms, particularly due to its efficient data migration, data availability and the small waiting time of tasks.

Simulation Results: Tasks arrive in 100 time-slot

- 2000 tasks to execute in 100 machines.
- Tasks arrive every 100-time slot.





- OTS-DMDR achieved the best performance through the efficient management of data and the consideration of machines characteristics during task scheduling.

- OTS-DMDR has the lowest degree of imbalance.
- OTS-DMDR considers the load of each machine when assigning tasks.

Conclusion

- OTS-DMDR enhances task scheduling by combining data migration and replication while considering machine's characteristics.
- OTS-DMDR achieves better data locality with efficient data migration and access times.
- OTS-DMDR outperforms existing techniques, significantly reducing response time, improving throughput and ensuring balanced machine loads.
- Some limitations of our work:
  - Static Data Placement: Needs of dynamic data placement techniques.
  - Ability to adapt to Workload Changes.

A New Classification for Data Placement Techniques in Cloud Computing

- **Laila Bouhouch**, Mostapha Zbakh and Claude Tadonki. "**A New Classification for Data Placement Techniques in Cloud Computing**," 2023 IEEE 6th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakech, Morocco, 2023, IEEE Xplore Digital Library, pp. 1-9. https://10.1109/CloudTech58737.2023.10366156

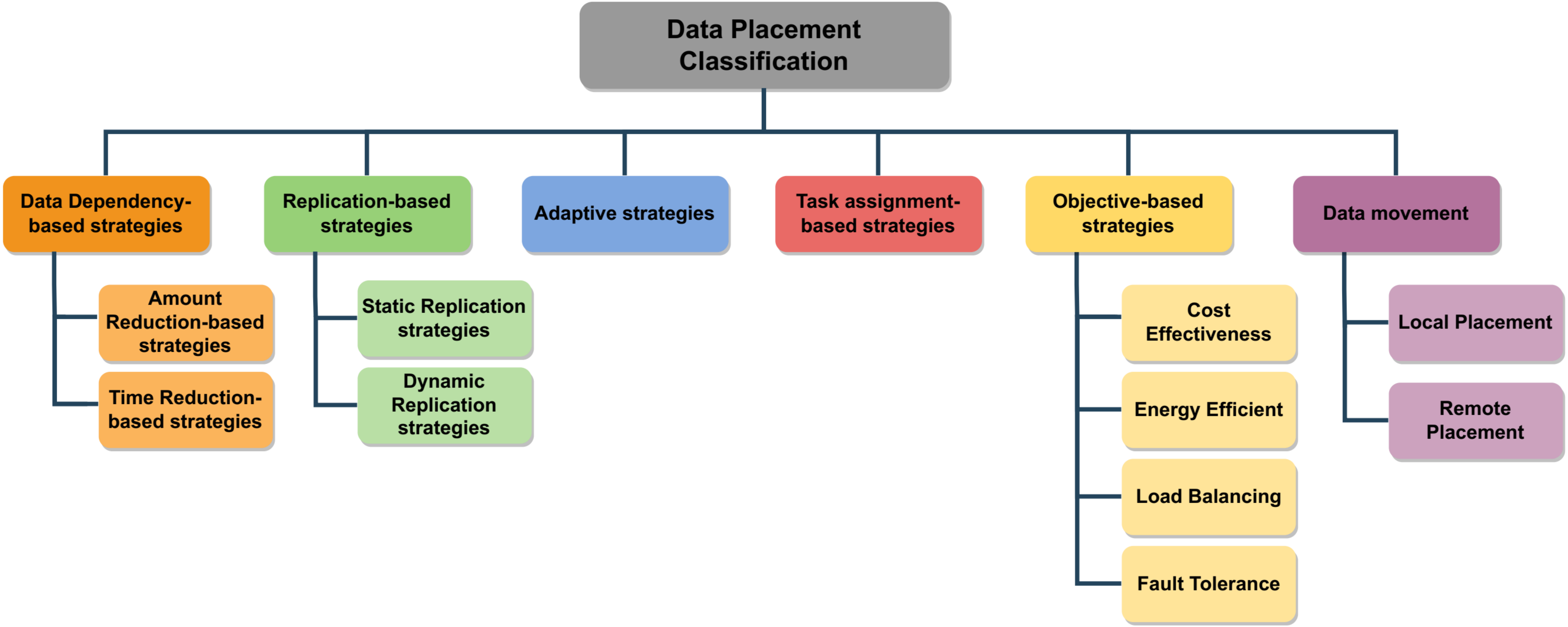
Motivation

There is a lack of comprehensive and organized classification for data placement techniques.

There is a need to

- Develop a New Classification for Data Placement Techniques in Cloud Computing.
- Categorize the existing strategies based on data dependencies, data movement, replication, task scheduling, and objectives such as *cost-effectiveness*, *energy consumption*, *fault-tolerance*, and *load balance*.

Proposed Classification



Conclusions

Cloudsim Extensions: Modeling and Simulation of Data Migration in Distributed Data Centers

Introduces DMCloudsim and DFMClousim modules to simulate data management strategies, using Cloudsim simulator.

Dynamic Data Replication and Placement Strategy in Geo-graphically Distributed Data Centers

Minimizes total execution time and financial costs by reducing the total migration time of datasets between data centers, using an efficient combination of data placement and data replication.

Online Task Scheduling of Big Data Applications in the Cloud Environment

Reduces the response time, improves the throughput and balances the loads between machines by achieving better data locality with access time while considering the heterogeneity of the system.

Future Work

- Explore dynamic data placement to adapt dynamic changes of cloud environment.
- Develop adaptive fragmentation for complex cloud systems.
- Apply Game Theory for decision-making improvements.
- Study data migration over data centers to boost the performance of the system.
- Implement our solutions in real platforms for realistic results.

Publications

International Conferences

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki. 2020.
  "**Data Migration: Cloudsim Extension**", In Proceedings of the 3rd International Conference on Big Data Research (ICBDR '19). Association for Computing Machinery, New York, NY, USA, 177–181.
  https://doi.org/10.1145/3372454.3372472

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki.
  "**A Big Data Placement Strategy in Geographically Distributed Datacenters**" 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakesh, Morocco, 2020, pp. 1-9.
  https://doi.org/10.1109/CloudTech49835.2020.9365881

- **Laila Bouhouch**, Mostapha Zbakh and Claude Tadonki.
  "**A New Classification for Data Placement Techniques in Cloud Computing**," 2023 IEEE 6th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech), Marrakech, Morocco, 2023, pp. 1-9.
  https://10.1109/CloudTech58737.2023.10366156

Publications

International Journals

- **Laila Bouhouch**, Mostapha Zbakh, and Claude Tadonki.
  "**Dynamic data replication and placement strategy in geographically distributed data centers**". Concurrency Computat Pract Exper. 2023; 35(14):e6858.
  https://doi.org/10.1002/cpe.6858

- **Laila Bouhouch**, Mostapha Zbakh and Claude Tadonki. 2023.
  "**DFMCloudsim: an extension of cloudsim for modeling and simulation of data fragments migration over distributed data centers**", International Journal of Computers and Applications.
  https://doi.org/10.1080/1206212X.2023.2277554

- **Laila Bouhouch,** Mostapha Zbakh, and Claude Tadonki.
  "**Online Task Scheduling of Big Data Applications in the Cloud Environment.**" Information 2023, 14, 292.
  https://doi.org/10.3390/info14050292

Thank you for your attention

ENSIAS, Mohammed V University of Rabat

Doctoral Thesis Defense

Efficient Management of Big Data Applications Deployed in the Cloud Computing

February 06, 2024

Presented by BOUHOUCH Laila

Jury Members

Pr. Mohamed Dafir ECH-CHERIF El KETTANI	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Président**
Pr. Mostapha ZBAKH	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Thesis supervisor**
Pr. Claude TADONKI	PES, Mines ParisTech, Ecole de Mines de Paris, France	**Thesis co-supervisor**
Pr. Christophe CERIN	Professeur, Université Sorbonne Paris Nord, France	**Dissertation reporter**
Pr. Mohsine ELEULDJ	PES, EMI, Université Mohammed V de Rabat, Maroc	**Dissertation reporter**
Pr. Mahmoud NASSAR	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation reporter**
Pr. Faissal EL BOUANANI	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation examiner**
Pr. Houda BENBRAHIM	PES, ENSIAS, Université Mohammed V de Rabat, Maroc	**Dissertation examiner**