# LXCloud-CR: Towards LinuX Containers Distributed Hash Table Based Checkpoint-Restart

**Doctorante: Thouraya LOUATI [a]**

**Directeurs de thèse: Heithem Abbes [a], Christophe Cérin [b] , Mohamed Jemni[a]**

**a. Laboratoire de Recherche en Technologies de l'Information et de la Communication et Génie Électrique LaTICE.**

**b. Laboratoire d'Informatique de Paris Nord**

**Mercredi 5 Juillet 2017**

# Plan

- Introduction
- Approche proposée : LXCloud-CR
- Conclusion & Travaux Futurs

## Motivation

- **Objectif de la thèse**
  - **Proposer une approche permettant de garantir la qualité de services QoS pour les plateformes Cloud en utilisant les mécanismes de Tolérance aux fautes.**

# Plan

- Introduction
- Approche proposée : LXCloud-CR
- Conclusion & Travaux Futurs

# Approche proposée(1): LXCloud-CR

## Motivation

- **Objectif de la thèse**
  - **Proposer une approche permettant de garantir la qualité de services QoS pour les plateformes Cloud <u>en utilisant les mécanismes de Tolérance aux fautes</u>**
    - **Explosion d'échelle**
    - **Fréquence de fautes (Prédiction: 10 minutes)**
      - ***Modern datacenters (DCs) host hundreds of thousands of nodes (Hardware commodity), dynamic***
    - **Le modèle *pay-as-you-go cloud model***
    - ***IaaS based containers* (déploiement simple des applications)**

**Replication des conteneurs**

→ **LXCloud-CR: Towards LXCs Distributed Hash Table Based CR**

## Introduction

- **Approche proposée**

**Un modèle décentralisé de sauvegarde des points de reprise des conteneurs LXC à base des tables de hachage distribuées DHT**
***« Key-value store »*** ***snapshots of the whole Linux Container (LXC) instances***

- **Passage à l'échelle**
- **Un stockage persistent des points de reprise**
- **Fournit la TF et assure la bonne terminaison des applications**
- **Performances**

**Nous visons la tolérance aux pannes des conteneurs LXC et les nœuds hébergeant ces conteneurs.**

## Introduction

- **Définition**

**Le recouvrement arrière se fonde sur la sauvegarde des points de reprise**

- **Un ensemble de données décrivant l'état de l'application.**
  - **Mémoire**
  - **Registre**
  - **etc...**

➡ **Un point de reprise d'un conteneur = l'état du conteneur + rootfs**

## Introduction

- **Checkpoint-Restart on IaaS Clouds based containers**
  - **Cas d'utilisation (*Live Migration*, Tolérance aux fautes)**
  - **Modèle de fautes (Par arrêt total: défaillance des nœuds=défaillance de tous les conteneurs ; défaillance de quelques conteneurs)**
  - **Modèle d'applications: Un ensemble de processus, chacun est lancé dans un conteneur.**
  - **État de calcul: État d'un conteneur**
  - **Objectifs (Transparence: blcr, CRIU, Performances: Éviter les surcoûts, Passage à l'échelle: Architecture décentralisée )**

## Solution proposée (1/8)

### 1 Sauvegarde des points de reprise

- Sauvegarder un point de reprise des applications distribuées lancées dans des conteneurs nécessite 2 opérations fondamentales qui doivent être orchestrées avec soin:

1. Sauvegarder localement et périodiquement l'état de chaque conteneur

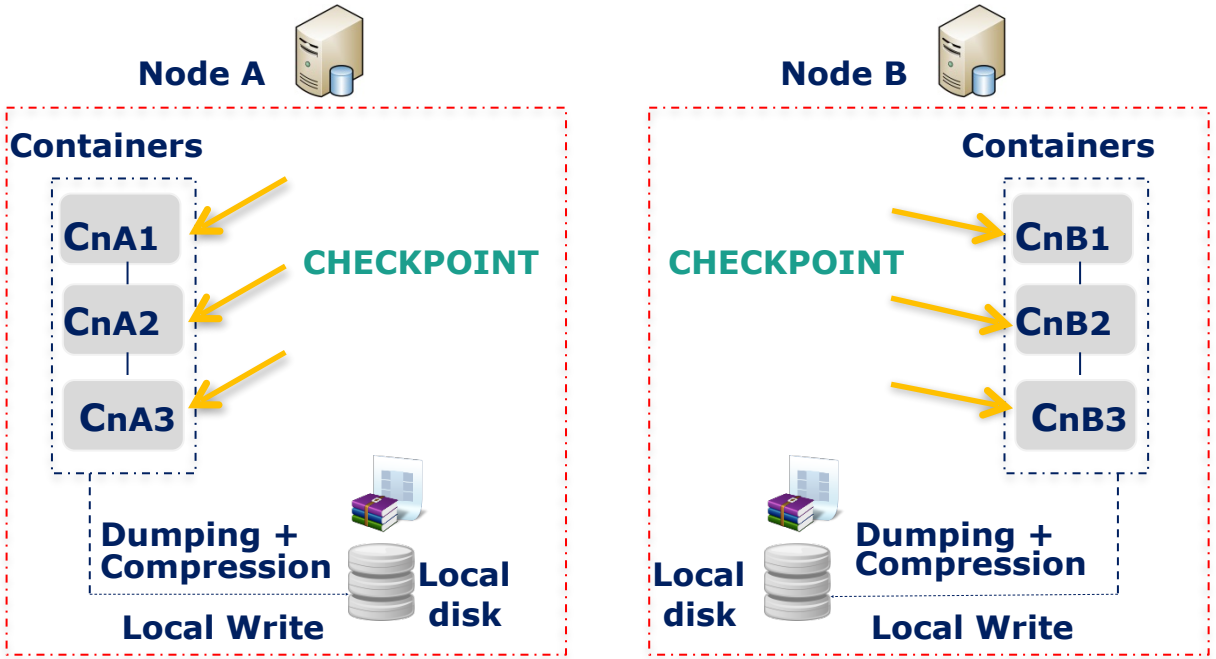
2. Sauvegarder, éventuellement, les points de reprise sur un support de stockage stable (d'une manière distribuée et persistente)

- Suite à une défaillance, les conteneurs défaillants redémarrent à partir d'un point de reprise « Snapshot »

## Solution proposée (2/8)

### 1 Sauvegarde locale des points de reprise

- Sauvegarder localement et périodiquement l'état des conteneurs




The CHECKPOINT function is presented in Algorithm 1.

**Algorithm 1** Initiate a dumping process

```
1: function CHECKPOINT(container: worker w_i)
2:     version ← 1
3:     Snapshot-Directory <− concat(worker w_i,version)
4:     Create_Snapshot_Directory(Snapshot-Directory)
5:     while worker w_i is RUNNING do
6:         lxc-checkpoint (worker w_i)
7:         Storing Dumping files in the Snapshot-Directory
8:         j ← version − 1
9:         Delete_Old_Snapshot(concat(worker w_i, j))
10:        Compression of the recent Snapshot-Directory
11:        Saving Dumping files in a distributed and
           persistent manner (See Sect. IV-B)
12:        Garbage Collection, Delete obsolete distributed
           snapshots
13:        Wait a period of time
14:    end while
15: end function
```

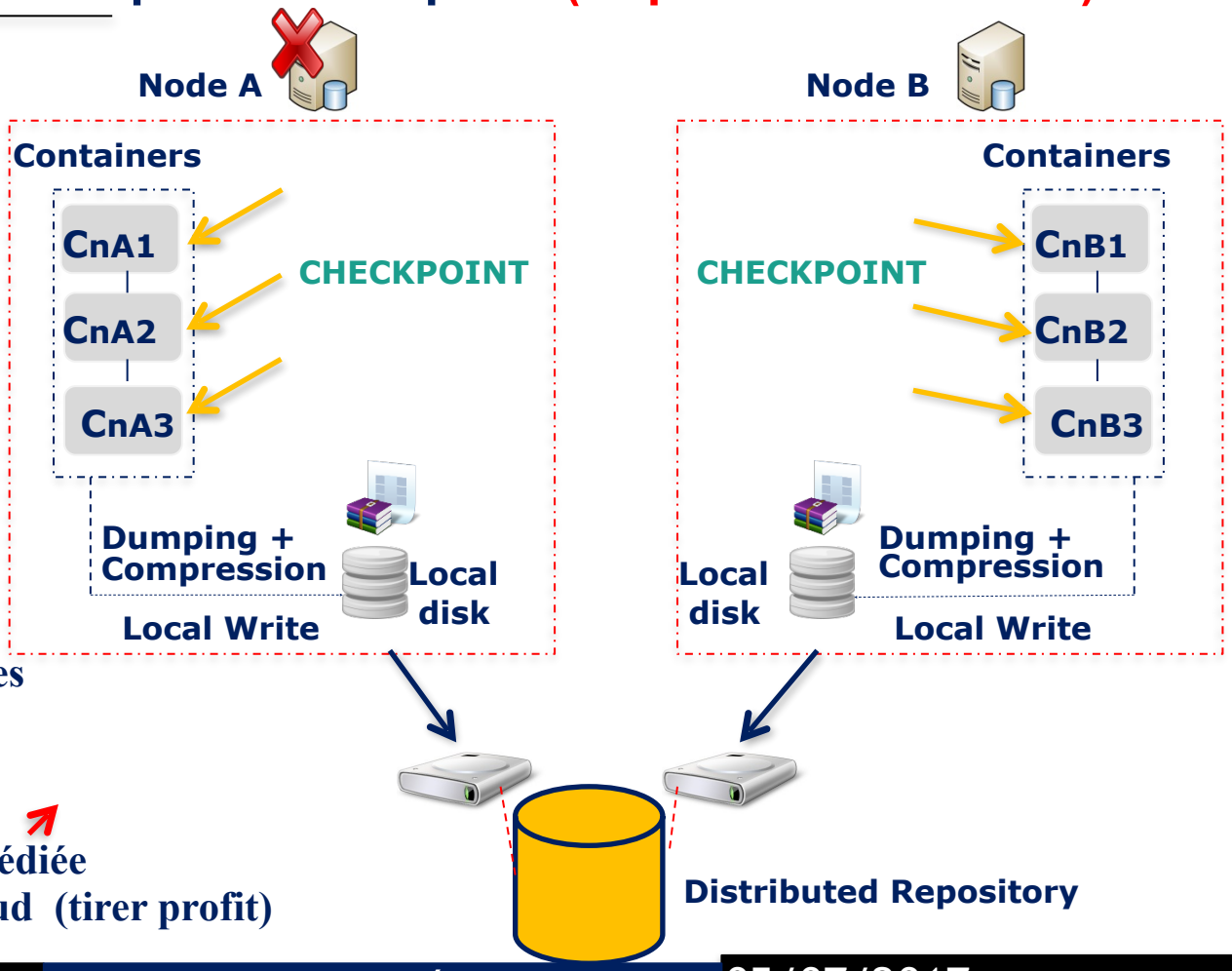
- Surcoût en termes de stockage
- Nombre de conteneurs augmentent ↗

- Un conteneur quitte et se reconnecte dans un délai «*time-out*»
- Le cas du *Crash* d'un conteneur LXC

## Solution proposée (3/8)

**2** Sauvegarde distribuée des points de reprise (Répertoire Distribué)

- Espace de stockage Local non persistent
- Snapshots perdus
- Stockage persistent?



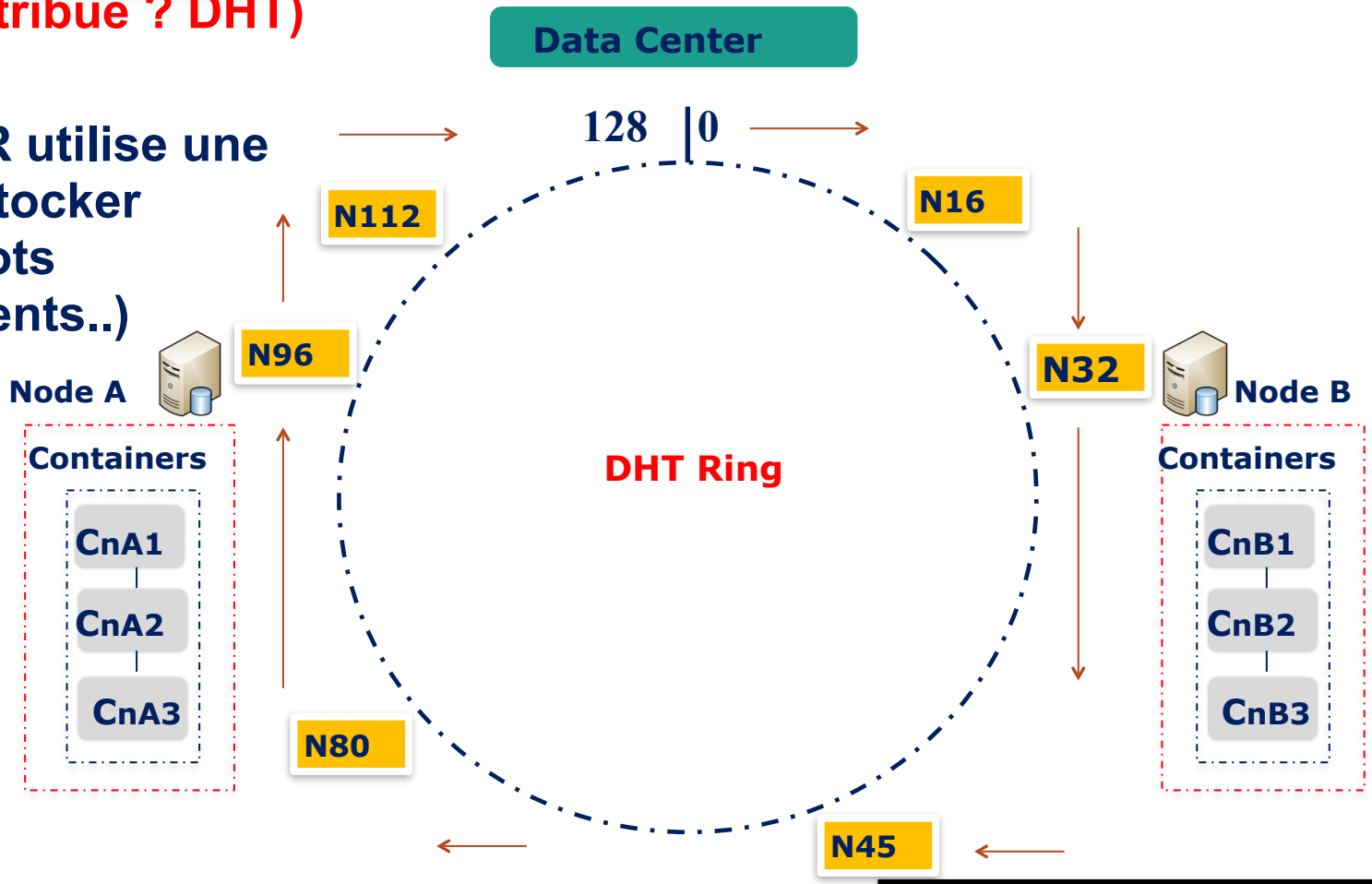

Répertoire distribué de stockage des checkpoints (LXC *Snapshots*)

- Passage à l'échelle
  - Nombre de Host ↗ Stockage ↗
  - C'est différent à une ressource dédiée
  - On utilise que les ressources du Cloud (tirer profit)

## Solution proposée (4/8)

2 **Sauvegarde distribuée des points de reprise (Comment on va gérer le répertoire distribué ? DHT)**

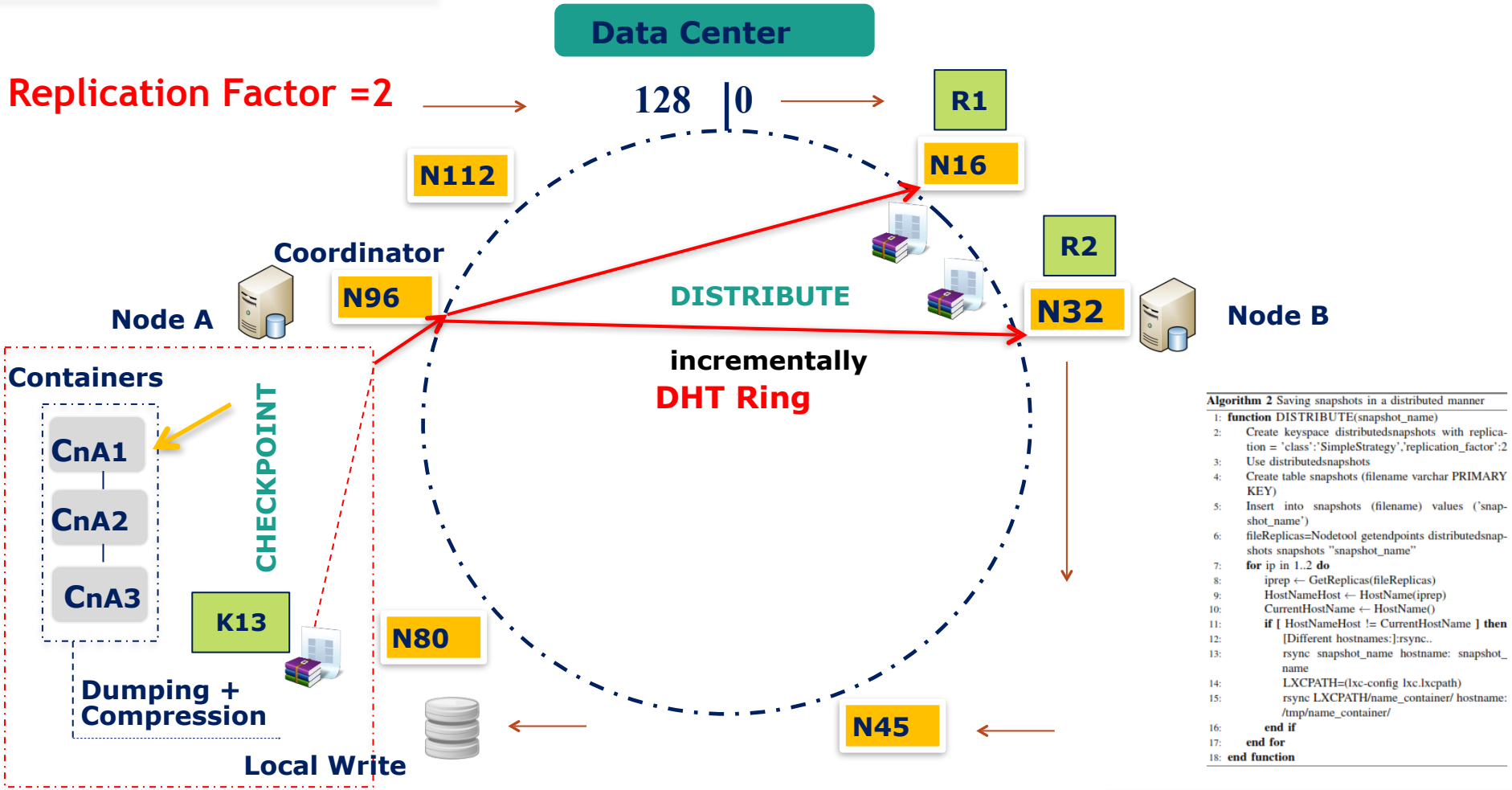
- LxCloud-CR utilise une DHT pour stocker les snaphsots (emplacements..)




Insert
Lookup
Delete
Objet = fichier (unique key)

## Solution proposée (6/8)

### 2 Sauvegarde distribuée des points de reprise (DISTRIBUTE)




**Algorithm 2** Saving snapshots in a distributed manner

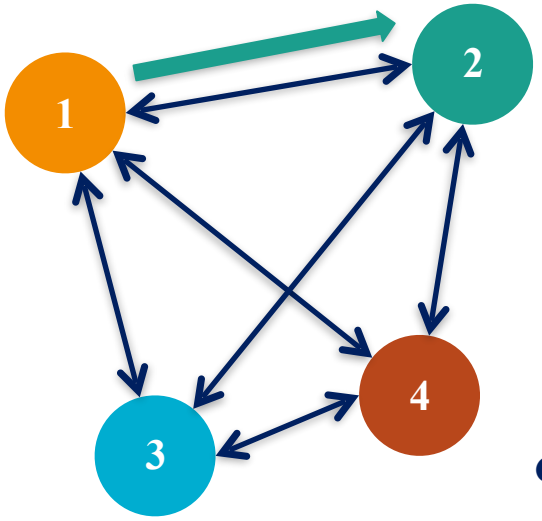
```
1:  function DISTRIBUTE(snapshot_name)
2:      Create keyspace distributedsnapshots with replication = 'class':'SimpleStrategy','replication_factor':2
3:      Use distributedsnapshots
4:      Create table snapshots (filename varchar PRIMARY KEY)
5:      Insert into snapshots (filename) values ('snapshot_name')
6:      fileReplicas=Nodetool getendpoints distributedsnapshots snapshots "snapshot_name"
7:      for ip in 1..2 do
8:          iprep ← GetReplicas(fileReplicas)
9:          HostNameHost ← HostName(iprep)
10:         CurrentHostName ← HostName()
11:         if [ HostNameHost != CurrentHostName ] then
12:             [Different hostnames:]:rsync..
13:             rsync snapshot_name hostname: snapshot_name
14:             LXCPATH=(lxc-config lxc.lxcpath)
15:             rsync LXCPATH/name_container/ hostname: /tmp/name_container/
16:         end if
17:     end for
18: end function
```

## Solution proposée (7/8)

### 3 Détection de fautes (Gossip Protocol)

| | | |
|---|---|---|
| 1 | 10121 | 67 |
| 2 | 10104 | 63 |
| 3 | 10099 | 64 |
| 4 | 10112 | 66 |

Address — Heartbeat Counter — Time (local)




| | | |
|---|---|---|
| 1 | 10119 | 65 |
| 2 | 10111 | 65 |
| 3 | 10091 | 59 |
| 4 | 10112 | 66 |

| | | |
|---|---|---|
| 1 | 10121 | 71 |
| 2 | 10111 | 63 |
| 3 | 10099 | 71 |
| 4 | 10112 | 66 |

Current time: 40 at node 2
(asynchronous clocks)

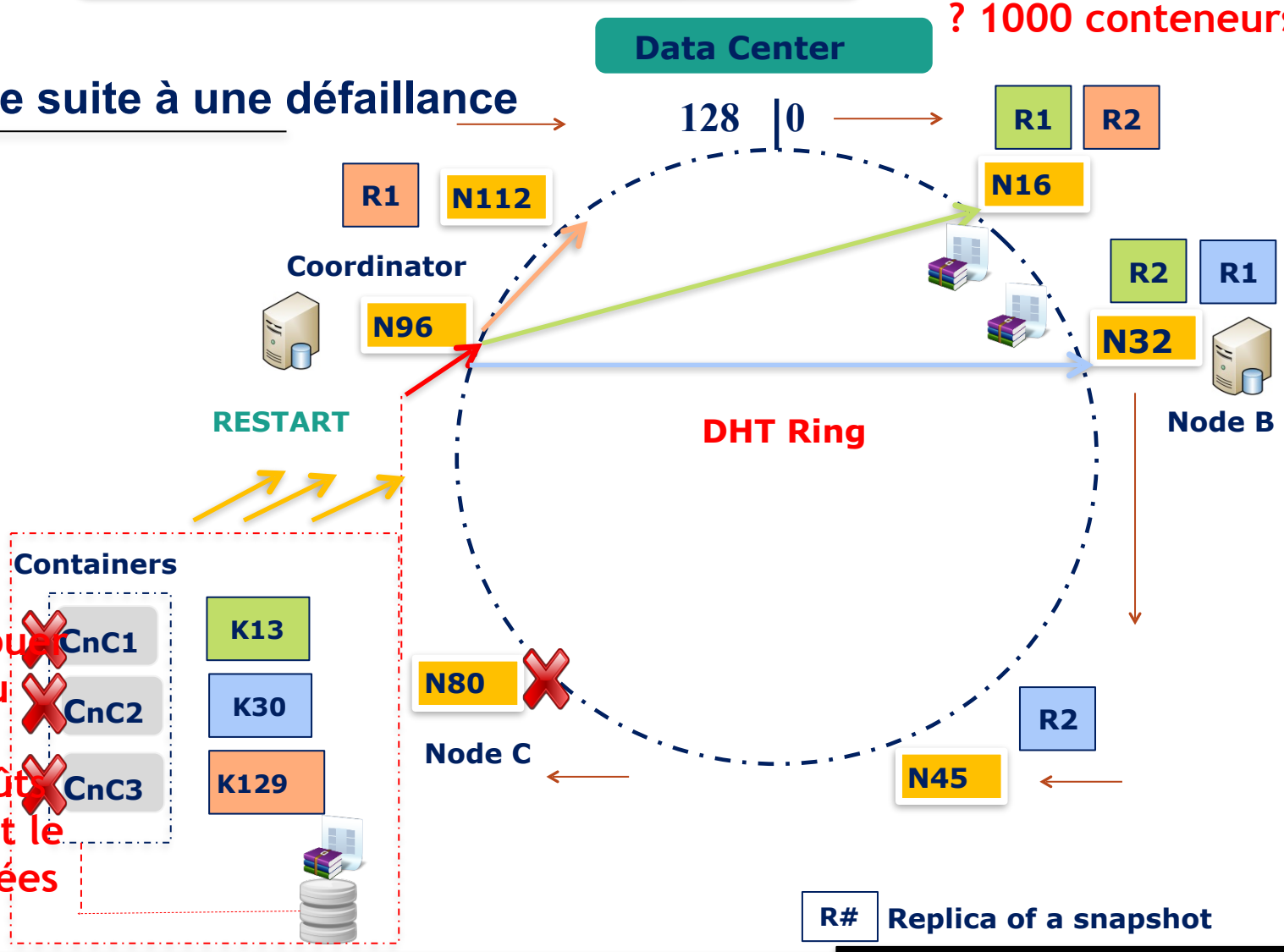
## Solution proposée (8/8)

Conteneurs supplémentaires ? 1000 conteneurs

### 4 Redémarrage suite à une défaillance

**Algorithm 3** Initiate the restart process following a Host failure

```
1:  function RESTART(file of failed containers at time 't')
2:      Extract ip address of failed containers DN: down
        normal
3:      for ip in ipaddr[] do
4:          HostOfFailedContainer ← GetHost(ip)
5:          HostNameContainer ← HostName(ip)
6:          fileReplicas ← GetFile(FailedContainer)
7:          iprep ← GetFirstReplicas(fileReplicas)
8:          fileStateHosts ← GetFile(States)
9:          state ← GetState(iprep, fileStateHosts)
10:         find=true
11:         if [ state = "DN" ] then
12:             "The first host hosting replicas is Down.."
13:             iprep ← GetSecondReplicas(fileReplicas)
14:             state1 ← GetState(ipaddrreplicas, fileStateHos
15:             if [ state1 = "DN" ] then
16:                 "The second host is also Down.."
17:                 find=false
18:             end if
19:         end if
20:         if [ find = True ] then
21:             Get hostname of host hosting replicas
22:             Connect to the corresponding host and:
23:             Copy configuration directory to /var/lib/lxc/
24:             Decompression of the snapshot
25:             Restart using lxc-checkpoint -r command
26:         else
27:             if [ find = False ] then
28:                 echo "Replicas not found, uncomplet..."
29:             end if
30:         end if
31:         Calling CHECKPOINT function (see Algo.1)
32:     end for
33: end function
```




- On a évité d'allouer des ressources au préalable
- Réduire les surcoûts de performance et le transfert de données

## Expérimentations

### 1 VM vs Container based Approaches

Table 1: Snapshot/Migration Comparison VM and Container (The service inside is MYSQL)

| Snapshot / Migration cost | VM | Container |
|---|---|---|
| Image size | 530 MB | 107MB |
| Snapshot duration | 7s | < 1s |
| Restore duration | 2s | < 1s |
| Image transfer duration (via scp) | 5s | < 1s |

Table 2: Snapshot of a Container

| Snapshot cost / Containers DGEMM Size | Image size(Mega) | Snapshot duration (second) | Restore duration (second) |
|---|---|---|---|
| 4000 | 411 M | < 1s | < 1s |
| 4500 | 513 M | < 1s | < 1s |
| 5000 | 626 M | < 1s | < 1s |
| 5500 | 751 M | < 1s | < 1s |
| 6000 | 887 M | =1s | < 1s |
| 6500 | 1100 M | =2 s | < 1s |
| 7000 | 1200 M | =2 s 52 | < 1s |
| 7500 | 1300 M | =2 s 74 | < 1s |
| 8000 | 1500 M | =2 s 93 | < 1s |
| 8500 | 1700 M | =3 s 22 | < 1s |

## Expérimentations

### 1 VM vs Container based Approaches

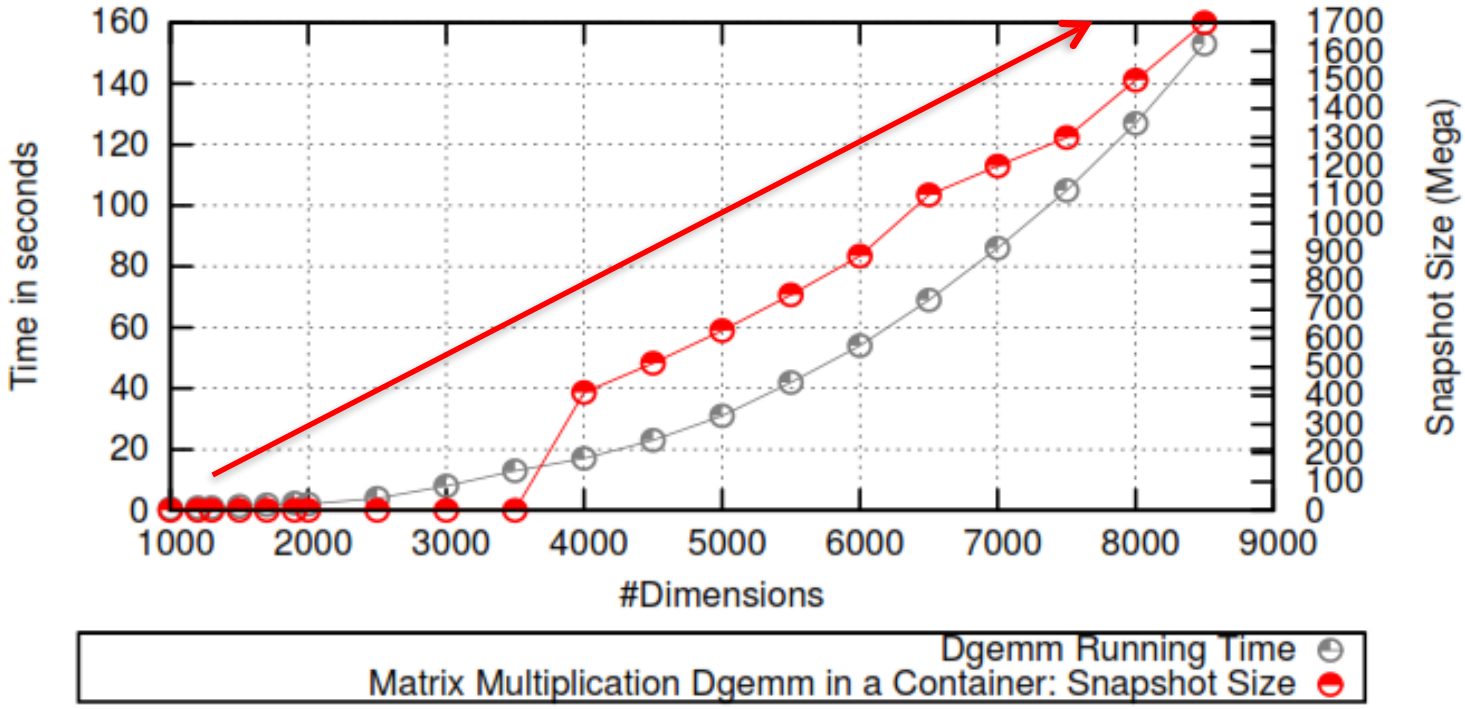



Figure 10: Snapshot of a Container

## Expérimentations

### 1 CRIU Vs BLCR Experimentally

| Snapshot cost / Containers DGEMM Size | Runtime (second) | Image size CRIU (Mega) | Snapshot duration (second) | Image size BLCR (Mega) | Snapshot duration BLCR (second) |
|---|---|---|---|---|---|
| 4000 | 17 | 411 M | < 1s | 402 M | 5 s 38 |
| 4500 | 23 | 513 M | < 1s | 504 M | 6 s 59 |
| 5000 | 31 | 626 M | < 1s | 617 M | 8 s 45 |
| 5500 | 42 | 751 M | < 1s | 742 M | 9 s 90 |
| 6000 | 54 | 887 M | =1s | 878 M | 11 s 39 |
| 6500 | 69 | 1100 M | =2s | 1100 M | 12 s 95 |
| 7000 | 86 | 1200 M | =2 s 52 | 1200 M | 13 s 66 |
| 7500 | 105 | 1300 M | =2 s 74 | 1300 M | 15 s 44 |
| 8000 | 129 | 1500 M | =2 s 93 | 1500 M | 18 s 71 |
| 8500 | 153 | 1700 M | =3 s 22 | 1700 M | 20 s 88 |

## Expérimentations

### 2 Détection de fautes

Table 3: Time to Detect Failure

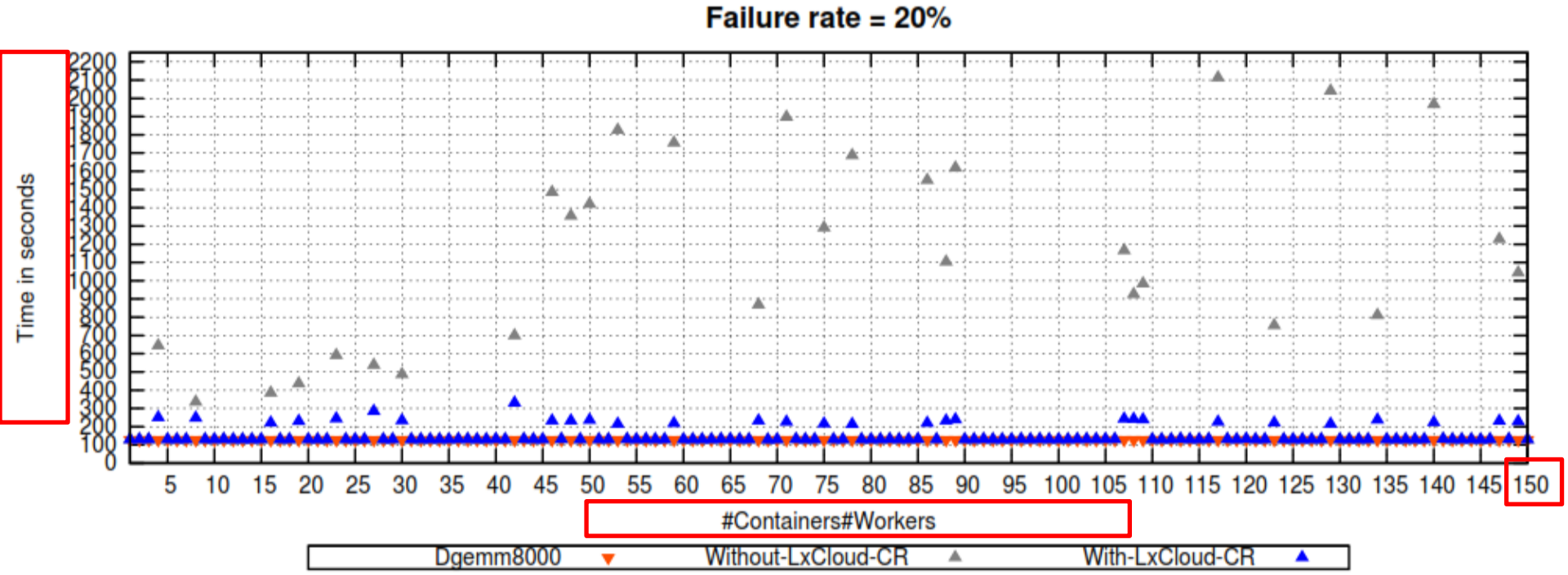
| State of: | Cost (s: seconds) |
|---|---|
| 9 containers | 2 s 84 |
| 15 containers | 3 s 64 |
| 24 containers | 5 s 37 |
| 36 containers | 7 s 03 |
| 45 containers | 8 s 78 |
| 51 containers | 9 s 58 |
| 55 containers | 10 s 57 |
| 65 containers | 13 s 01 |
| 70 containers | 14 s 47 |
| 76 containers | 16 s 12 |
| 81 containers | 16 s 48 |
| 84 containers | 19 s 55 |

## Expérimentations

Application : 129 secondes
With-LxCloud-CR: 330 secondes
Without LxCloud-CR: 1984 secondes
1

### 3 Évaluation expérimentale ( Taux de pannes des conteneurs= 20%)

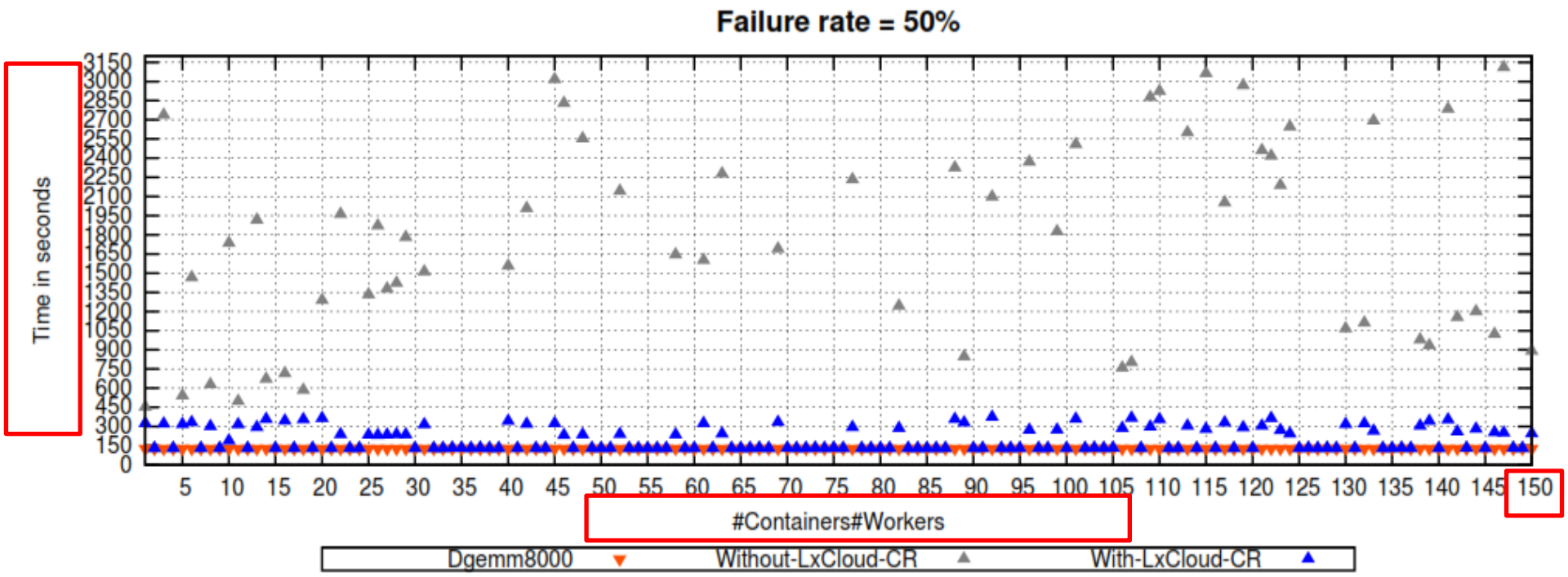
**Grid'5000, 50 nodes (i.e. 150 conteneurs)**
**CRIU version 1.6.1; Apache Cassandra 2.1.6**



## Expérimentations

Application : 129 secondes
With-LxCloud-CR: 372 secondes
Without LxCloud-CR: 3109 secondes

### 3 Évaluation expérimentale ( Taux de panne 2 conteneurs= 50%)



## Expérimentations

### 2 Analyses des résultats

- Le surcoût correspond au temps nécessaire pour le redémarrage suite aux défaillances. Pourquoi ? Pourquoi il est en cours d'augmentation ?

**Sans LxCloud-CR:**

- Défaillance = L'exécution du benchmark est arrêtée
  Le conteneur quitte le réseau (ring)
  Sans supervision
  Redémarrage = Redémarrage du conteneur défaillant *lxc-start*
  Le conteneur rejoint le réseau
  Démarrage du protocole gossip (Supervision pour une autre éventuelle défaillance)
  Démarrage du benchmark depuis le début.

→ Les conteneurs défaillants ne peuvent pas rejoindre le ring (nœud boots simultanément → Redémarrage séquentiel

## Expérimentations

### 2 Analyses des résultats

- Le surcoût correspond au temps nécessaire pour le redémarrage suite aux défaillances. (Un surcoût pas très important, constant presque! )

  **Avec LxCloud-CR:**

  - Défaillance = L'exécution du benchmark est arrêtée
    Le conteneur quitte le réseau (ring)
    Sans supervision

    Redémarrage = Redémarrage en parallèle (lxc-checkpoint –r) des conteneurs défaillants (ceci restore automatiquement l'état du réseau ring et le benchmark redémarre à partir de son dernier état )

**LxCloud-CR améliore les performances.**

## Expérimentations

### 3 Évaluation expérimentale ( Taux de panne des Hosts= 20%)

Table X
STATE HOSTS

| – | Address | Host ID | Rack |
|---|---|---|---|
| UN | 172.16.65.72 | 9557ecc8-a615-4c72-9df1 | 65 |
| UN | 172.16.65.91 | f3ecb30d-50b7-491c-a723 | 65 |
| UN | 172.16.65.8 | 5c6203fc-7427-4b02-af19 | 65 |
| UN | 172.16.65.77 | a4337e4c-f94b-494f-a4f6 | 65 |
| UN | 172.16.65.92 | e26b26e0-34b9-4436-9f91 | 65 |
| UN | 172.16.65.9 | c08de9c8-866c-494b-a8e6 | 65 |

.....

Table XI
STATE HOSTS

| – | Address | Host ID | Rack |
|---|---|---|---|
| UN | 172.16.65.72 | 9557ecc8-a615-4c72-9df1 | 65 |
| UN | 172.16.65.91 | f3ecb30d-50b7-491c-a723 | 65 |
| DN | 172.16.65.8 | 5c6203fc-7427-4b02-af19 | 65 |
| UN | 172.16.65.77 | a4337e4c-f94b-494f-a4f6 | 65 |
| UN | 172.16.65.92 | e26b26e0-34b9-4436-9f91 | 65 |
| DN | 172.16.65.9 | c08de9c8-866c-494b-a8e6 | 65 |

## Expérimentations

### 3 Évaluation expérimentale ( Taux

Table XII
FIRST FAILED HOST

| NAME | STATE | IPV4 |
|---|---|---|
| griffon-8.nancy.grid5000.fr-worker1 | STOPPED | - |
| griffon-8.nancy.grid5000.fr-worker2 | STOPPED | - |
| griffon-8.nancy.grid5000.fr-worker3 | STOPPED | - |
| worker | STOPPED | - |

Table XIII
SECOND FAILED HOST

| NAME | STATE | IPV4 |
|---|---|---|
| griffon-9.nancy.grid5000.fr-worker1 | STOPPED | - |
| griffon-9.nancy.grid5000.fr-worker2 | STOPPED | - |
| griffon-9.nancy.grid5000.fr-worker3 | STOPPED | - |
| worker | STOPPED | - |

Table XIV
DISTRIBUTED REPLICAS

| Failed Containers | Replicas 1 | Replicas 2 |
|---|---|---|
| griffon-8.nancy.grid5000.fr-worker1 | 172.16.65.91 | 172.16.65.72 |
| griffon-8.nancy.grid5000.fr-worker2 | 72.16.65.66 | 172.16.65.9 |
| griffon-8.nancy.grid5000.fr-worker3 | 72.16.65.66 | 172.16.65.85 |
| griffon-9.nancy.grid5000.fr-worker1 | 72.16.65.92 | 172.16.65.71 |
| griffon-9.nancy.grid5000.fr-worker2 | 172.16.65.67 | 172.16.65.71 |
| griffon-9.nancy.grid5000.fr-worker3 | 172.16.65.67 | 172.16.65.85 |

Table XV
LISTING CONTAINERS

| NAME | STATE | IPV4 |
|---|---|---|
| griffon-8.nancy.grid5000.fr-worker1 | RUNNING | 10.147.239.144 |
| griffon-91.nancy.grid5000.fr-worker1 | RUNNING | 10.147.239.136 |
| griffon-91.nancy.grid5000.fr-worker2 | RUNNING | 10.147.239.135 |
| griffon-91.nancy.grid5000.fr-worker3 | RUNNING | 10.147.239.134 |
| worker | STOPPED | - |

## Expérimentations

Application : 129 secondes
Stratégie de réplication + Without-CR: 1540 secondes
Stratégie de replication + With-CR: 2546 secondes

1

### 3 Évaluation expérimentale ( Taux de pann[illegible]s Hosts= 20%)

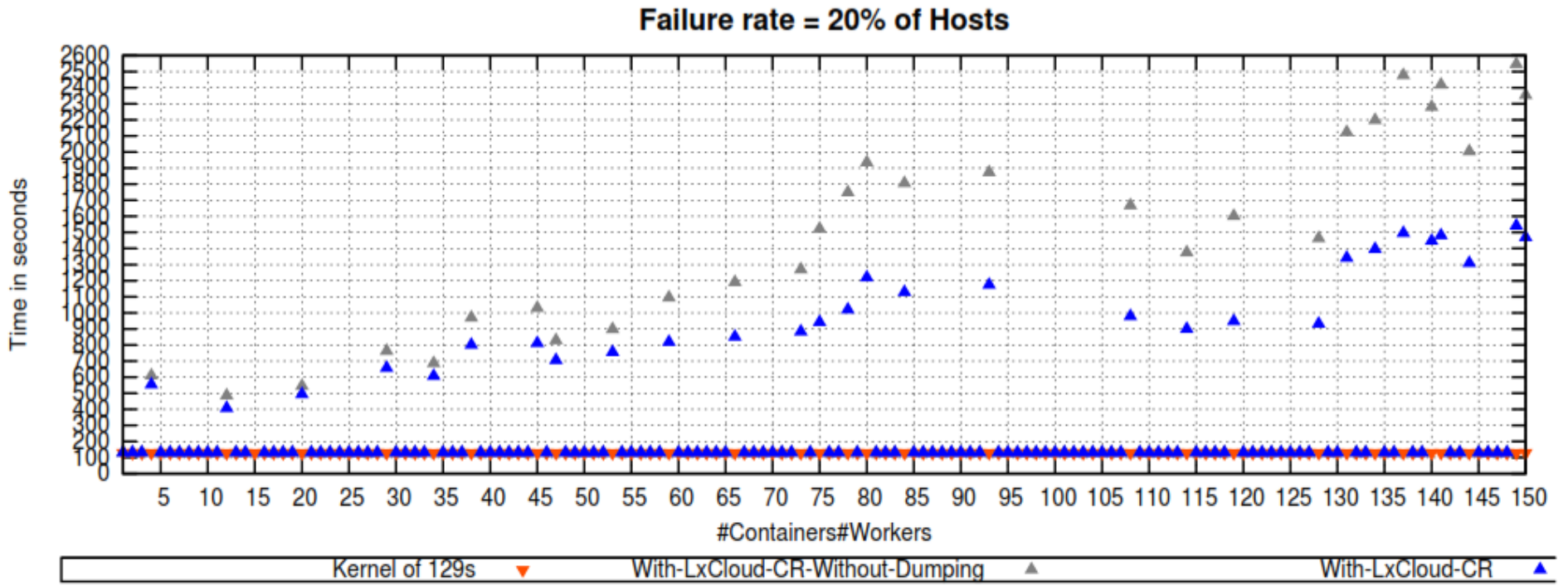



Figure 13: Injecting Fault on 20% of Hosts

## Expérimentations

### 3 Évaluation expérimentale ( Taux de panne des Hosts= 50%)

Table XVI
DISTRIBUTED REPLICAS

| Failed containers | Replicas 1 | Replicas 2 |
|---|---|---|
| griffon-61.nancy.grid5000.fr-worker1 | 172.16.65.85 | 172.16.65.9 |
| griffon-61.nancy.grid5000.fr-worker2 | 172.16.65.85 | 172.16.65.41 |
| griffon-61.nancy.grid5000.fr-worker3 | 172.16.65.85 | 172.16.65.71 |
| griffon-63.nancy.grid5000.fr-worker1 | 172.16.65.67 | 172.16.65.38 |
| griffon-63.nancy.grid5000.fr-worker2 | Same node | 172.16.65.62 |
| griffon-63.nancy.grid5000.fr-worker3 | 172.16.65.9 | 172.16.65.58 |
| griffon-66.nancy.grid5000.fr-worker1 | 172.16.65.67 | 172.16.65.9 |
| griffon-66.nancy.grid5000.fr-worker2 | 172.16.65.61 | 172.16.65.38 |
| griffon-66.nancy.grid5000.fr-worker3 | 172.16.65.62 | 172.16.65.85 |
| griffon-71.nancy.grid5000.fr-worker1 | 172.16.65.38 | 172.16.65.9 |
| griffon-71.nancy.grid5000.fr-worker2 | 172.16.65.67 | 172.16.65.58 |
| griffon-71.nancy.grid5000.fr-worker3 | Same Node | 172.16.65.85 |
| griffon-9.nancy.grid5000.fr-worker1 | 172.16.65.62 | 172.16.65.38 |
| griffon-9.nancy.grid5000.fr-worker2 | Same Node | 172.16.65.66 |
| griffon-9.nancy.grid5000.fr-worker3 | 172.16.65.62 | 172.16.65.58 |

Down

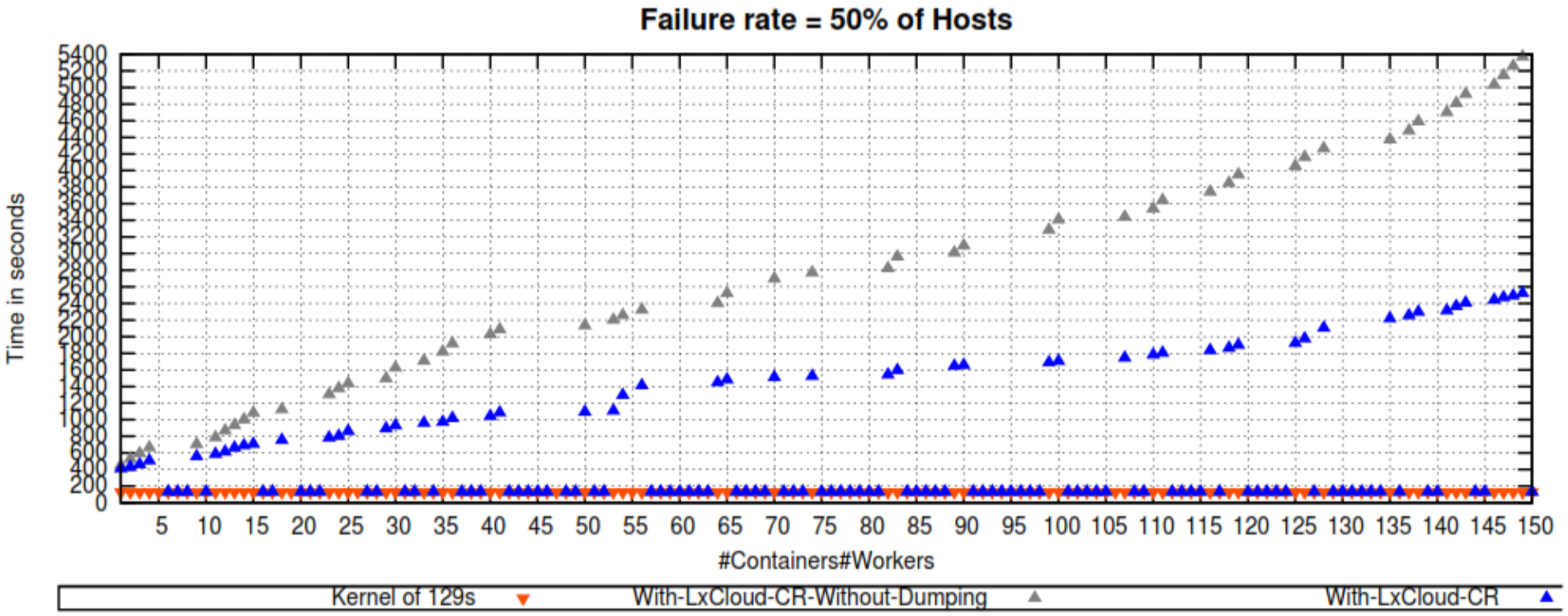
Uncomplete

Table XVII
LISTING CONTAINERS

| NAME | STATE | IPV4 |
|---|---|---|
| griffon-38.nancy.grid5000.fr-worker1 | RUNNING | 10.147.25.216 |
| griffon-38.nancy.grid5000.fr-worker2 | RUNNING | 10.147.239.130 |
| griffon-38.nancy.grid5000.fr-worker3 | RUNNING | 10.147.239.129 |
| griffon-66.nancy.grid5000.fr-worker2 | RUNNING | 10.147.239.112 |
| griffon-71.nancy.grid5000.fr-worker1 | RUNNING | 10.147.239.107 |
| worker | STOPPED | - |

## Expérimentations

### 3 Évaluation expérimentale ( Taux de panne des Hosts= 50%)



## Travaux Existants

Table 17: Comparing Checkpoint-Restart Approaches

| Approach | Virtualization Technology | Checkpoint-Restart | Checkpointing Approach | Managing Check-points | Storage |
|---|---|---|---|---|---|
| Tutum | IaaS clouds based Containers (Docker) | No | No | No | No |
| OpenQRM | IaaS clouds based Containers (lxc) | No | No | No | No |
| dotCloud | IaaS clouds based Containers (Docker) | No | No | No | No |
| Docker | IaaS clouds based Containers | Yes | System Level (CRIU) | No | No |
| Nicolae and Cappello | IaaS clouds based VMs | Yes | Virtual disk image | Yes | Decentralized |
| Our Approach LXCloud-CR | IaaS clouds based Containers (lxc) | Yes | System Level (CRIU) | Yes | Decentralized |

## Travaux Existants

### 1 Travaux connexes

- **Application-level-checkpointing**

**Le code permettant la sauvegarde de points de reprise est inséré directement dans l'application par le programmeur.**

**Réduire la taille d'un *snapshot*. On déclenche le *dumping process* quand on a peu de données.**

**L'utilisation d'une telle approche complexifie la tâche des développeurs.**

**N'est pas transparent ni pour les applications ni pour les utilisateurs.**

## Travaux Existants

### 2 Travaux connexes

- **System-level-checkpointing**

**C'est à travers l'utilisation des bibliothèques niveau système.**

- **Transparence pour les applications et pour les utilisateurs.**
- **La fréquence. On peut faire le *dumping process* à n'importe quel moment Exemple: Gaussian APP qui traite des données de taille gigantesque pour des heures.**

## Travaux Existants

### 3 Travaux connexes

- **BLCR**
  - **Sauvegarde que l'état des processus et écarte l'état du conteneur.**
  - **Ceci engendre un surcoût lors du redémarrage suite à une défaillance (création et initialisation du conteneur).**

- **CRIU (sauvegarde l'état de tout un conteneur : rootFS, RAM etc,,)**
- **Approach *that snapshot only the layered file system of the container and discard the memory* (e.g: BlobCR for HPC applications), Sous le prétexte que la taille d'un snapshot est plus petite avec un redémarrage rapide.**
  - **Offline Migration. Avec les plateformes Cloud, on vise Live migration.**

## Travaux Existants

### 4 Travaux connexes

- **Sauvegarde distribuée des points de reprise (Répertoire Distribué)**

**DHT vs (parallel and distributed) File Systems qui supportent la réplication.**
***With Fault Tolerance and reliability, DHT gives a faster lookup mechanism accross various nodes in a cluster.***

**HPC: Multi-level Checkpoint Restart which is aiming at using local disks**
**SCR library *Scalable Checkpoint/Restart Library.* Cette approche utilise un système de fichiers distribué et ne garde pas des réplicas du snapshot.**

**FTI: high performance Fault Tolerance Interface for hybrid systems (*application level checkpointing*)**

## Publication

### 1 Papier Journal


LXCloud-CR: Towards LinuX Containers Distributed Hash Table Based Checkpoint-Restart

Thouraya Louati[a,*], Heithem Abbes[a,*], Christophe Cérin[b], Mohamed Jemni[a]

[a] *LaTICE Research Lab., ENSIT, University of Tunis, Tunisia*
*5 Avenue Taha Hussein, BP, 56, Bâb Manara, Tunis, Tunisia*
[b] *Université de Paris 13, LIPN, UMR CNRS 7030, 99, avenue Jean-Baptiste Clément,*
*93430 Villetaneuse, France*

**Abstract**

Infrastructure-as-a-Service (IaaS) container-based virtualization technology is gaining, over these years, significant interest in industry as an alternative platform for running distributed applications. They present an interesting alternative to virtual machines in the Cloud due to newer advances in container-based virtualization as a new technology that simplify the deployment of applications. As the cloud architectures continue to grow in scale and complexity, faults become very recurrent which make reliability a true challenge. Given the dynamic nature of IaaS clouds and the pay-as-you-go cloud model where the costs are directly proportional to the resource usage, a Checkpoint Restart mechanism is essential in this context. We propose LXCloud-CR our new decentralized Checkpoint-Restart model based on a distributed checkpoints repository using key-value store on a Distributed Hash Table (DHT). It is able to take snapshots of the whole Linux Container (LXC) instances. LXCloud-CR aims to reduce the runtime and storage overheads of checkpointing. Large scale experiments on the Grid'5000 testbed demonstrate the benefits of our proposal. Obtained results validate our model and improve the performance of applications.

*Keywords:*
Cloud computing; IaaS; container; virtualization; fault tolerance; checkpoint-restart; migration; DHT; Grid'5000.

## Motivation

### Objectif

- **Checkpoint-Restart est une technique fondamentale de tolérance aux fautes.**

⚠ **Surcoût en termes de stockage.**

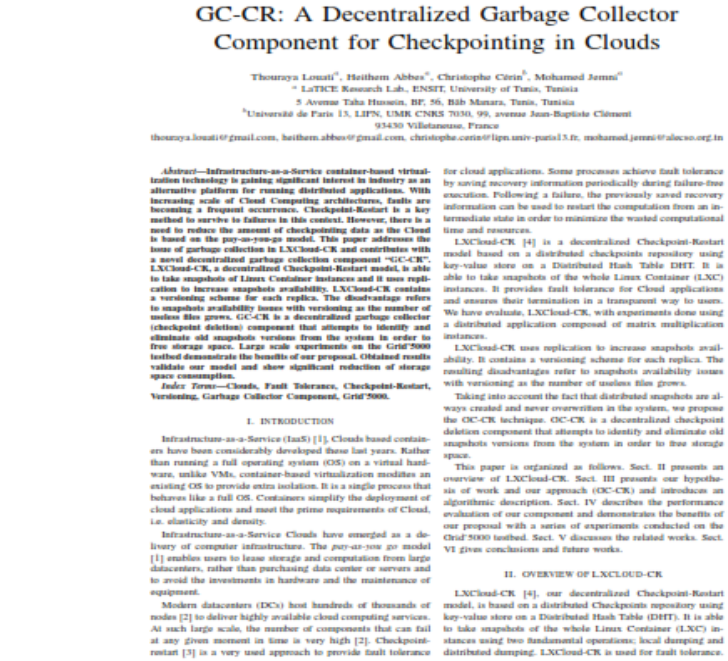
➡ **Il est nécessaire de réduire les données de *checkpointing* comme le Cloud est basé sur le modèle *pay-as-you-go.***

- **LXCloud-CR est un modèle décentralisé de sauvegarde des points de reprise des conteneurs LXC à base des tables de hachage distribuées DHT*« Key-value store »* *snapshots of the whole Linux Container (LXC) instances***
- **LXCloud-CR contient un schéma de *versioning* pour chaque réplica.**

⚠ **Le nombre de checkpoints obsolètes (inutiles) augmente considérablement.**

## Motivation

- **Objectif**
  - **GC-CR: A Decentralized Garbage Collector Component for Checkpointing in Clouds**



GC-CR: A Decentralized Garbage Collector Component for Checkpointing in Clouds

Thouraya Louati[a], Heithem Abbes[a], Christophe Cérin[b], Mohamed Jemni[a]
[a] LaTICE Research Lab., ENSIT, University of Tunis, Tunisia
5 Avenue Taha Hussein, BP, 56, Bâb Manara, Tunis, Tunisia
[b]Université de Paris 13, LIPN, UMR CNRS 7030, 99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
thouraya.louati@gmail.com, heithem.abbes@gmail.com, christophe.cerin@lipn.univ-paris13.fr, mohamed.jemni@alecso.org.tn

**Abstract—Infrastructure-as-a-Service container-based virtualization technology is gaining significant interest in industry as an alternative platform for running distributed applications. With increasing scale of Cloud Computing architectures, faults are becoming a frequent occurrence. Checkpoint-Restart is a key method to survive to failures in this context. However, there is a need to reduce the amount of checkpointing data as the Cloud is based on the pay-as-you-go model. This paper addresses the issue of garbage collection in LXCloud-CR and contributes with a novel decentralized garbage collection component "GC-CR". LXCloud-CR, a decentralized Checkpoint-Restart model, is able to take snapshots of Linux Container instances and it uses replication to increase snapshots availability. LXCloud-CR contains a versioning scheme for each replica. The disadvantage refers to snapshots availability issues with versioning as the number of useless files grows. GC-CR is a decentralized garbage collector (checkpoint deletion) component that attempts to identify and eliminate old snapshots versions from the system in order to free storage space. Large scale experiments on the Grid'5000 testbed demonstrate the benefits of our proposal. Obtained results validate our model and show significant reduction of storage space consumption.**

***Index Terms*—Clouds, Fault Tolerance, Checkpoint-Restart, Versioning, Garbage Collector Component, Grid'5000.**

I. INTRODUCTION

Infrastructure-as-a-Service (IaaS) [1], Clouds based containers have been considerably developed these last years. Rather than running a full operating system (OS) on a virtual hardware, unlike VMs, container-based virtualization modifies an existing OS to provide extra isolation. It is a single process that behaves like a full OS. Containers simplify the deployment of cloud applications and meet the prime requirements of Cloud, i.e. elasticity and density.

Infrastructure-as-a-Service Clouds have emerged as a delivery of computer infrastructure. The *pay-as-you go* model [1] enables users to lease storage and computation from large datacenters, rather than purchasing data center or servers and to avoid the investments in hardware and the maintenance of equipment.

Modern datacenters (DCs) host hundreds of thousands of nodes [2] to deliver highly available cloud computing services. At such large scale, the number of components that can fail at any given moment in time is very high [2]. Checkpoint-restart [3] is a very used approach to provide fault tolerance for cloud applications. Some processes achieve fault tolerance by saving recovery information periodically during failure-free execution. Following a failure, the previously saved recovery information can be used to restart the computation from an intermediate state in order to minimize the wasted computational time and resources.

LXCloud-CR [4] is a decentralized Checkpoint-Restart model based on a distributed checkpoints repository using key-value store on a Distributed Hash Table DHT. It is able to take snapshots of the whole Linux Container (LXC) instances. It provides fault tolerance for Cloud applications and ensures their termination in a transparent way to users. We have evaluate, LXCloud-CR, with experiments done using a distributed application composed of matrix multiplication instances.

LXCloud-CR uses replication to increase snapshots availability. It contains a versioning scheme for each replica. The resulting disadvantages refer to snapshots availability issues with versioning as the number of useless files grows.

Taking into account the fact that distributed snapshots are always created and never overwritten in the system, we propose the GC-CR technique. GC-CR is a decentralized checkpoint deletion component that attempts to identify and eliminate old snapshots versions from the system in order to free storage space.

This paper is organized as follows. Sect. II presents an overview of LXCloud-CR. Sect. III presents our hypothesis of work and our approach (GC-CR) and introduces an algorithmic description. Sect. IV describes the performance evaluation of our component and demonstrates the benefits of our proposal with a series of experiments conducted on the Grid'5000 testbed. Sect. V discusses the related works. Sect. VI gives conclusions and future works.

II. OVERVIEW OF LXCLOUD-CR

LXCloud-CR [4], our decentralized Checkpoint-Restart model, is based on a distributed Checkpoints repository using key-value store on a Distributed Hash Table (DHT). It is able to take snapshots of the whole Linux Container (LXC) instances using two fundamental operations: local dumping and distributed dumping. LXCloud-CK is used for fault tolerance.


**Soumis à SBAC-PAD 2017**

# Plan

- Introduction
- Approche proposée : LXCloud-CR
- Conclusion & Travaux Futurs

# Conclusion & Travaux Futurs

## 1 Conclusion

- **Approches proposées**

  - **Une approche permettant de garantir la qualité de services QoS pour les plateformes Cloud en utilisant les mécanismes de Tolérance aux fautes.**

    **→ LXCloud-CR**

## Conclusion & Travaux Futurs

### 2 Travaux Futurs

- **LXCloud-CR:**
  - **Gérer d'une manière dynamique la fréquence des points de reprise.**

# MERCI