History-Based Harvesting of Spare Cycles and Storage in Large-Scale Datacenters

Yunqi Zhang, George Prekas, Giovanni Matteo Fumarola, Marcus Fontoura, Inigo Goiri, Ricardo Bianchini.
University of Michigan and Microsoft Research

12th USENIX symposium on Operating System Design and Implementation

Presenter: Yi-Ning Chang

July 27, 2017

Outline

1. Introduction
2. Behavior Patterns
3. Co-location Techniques
4. Experiment
5. Conclusion

1. Introduction
2. Behavior Patterns
3. Co-location Techniques
4. Experiment
5. Conclusion

Introduction

- Overprovision resources in datacenters
  - Low tail latency requirement
  - Provisioned for peak load
  - Unexpected load spikes and failures
- A way to increase utilization and reduce costs in datacenters is to co-locate their latency-critical services and batch workloads.
- Harvest spare compute cycles and storage space for co-location purpose.

Challenges

- Interactive services “own” the servers
- Resource availability
  - Interactive services performance
  - Resource availability dynamics - task killing
- Data storage co-location
  - Data availability
  - Data durability

1 Introduction

2 Behavior Patterns

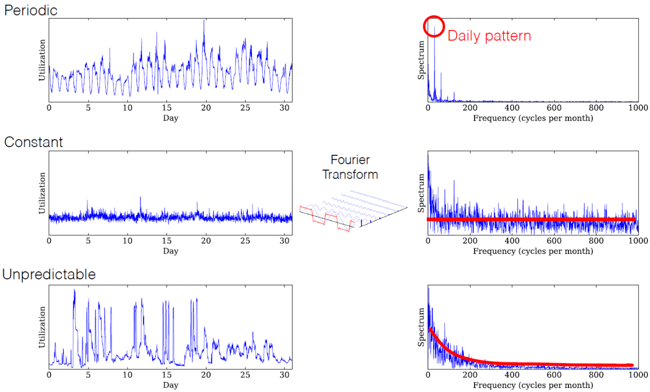
3 Co-location Techniques

4 Experiment

5 Conclusion

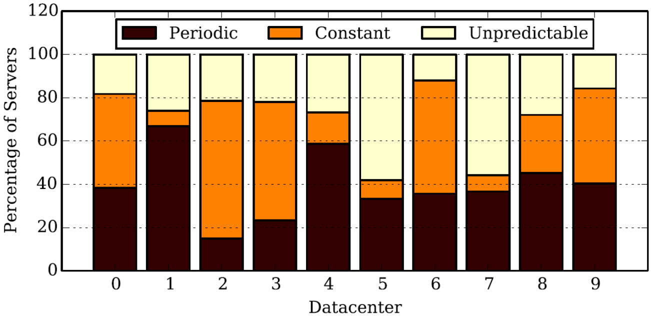
Resource Utilization

- Identify three main classes of primary tenants.



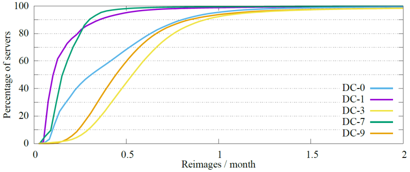
Resource Utilization

- Percentages of servers per class.

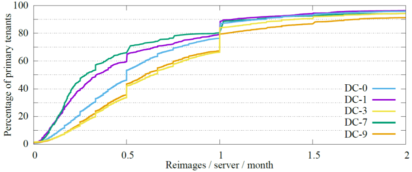


Disk Reimaging

- Per-server number of reimages in three years.



- Per-tenant number of reimages in three years.



1 Introduction

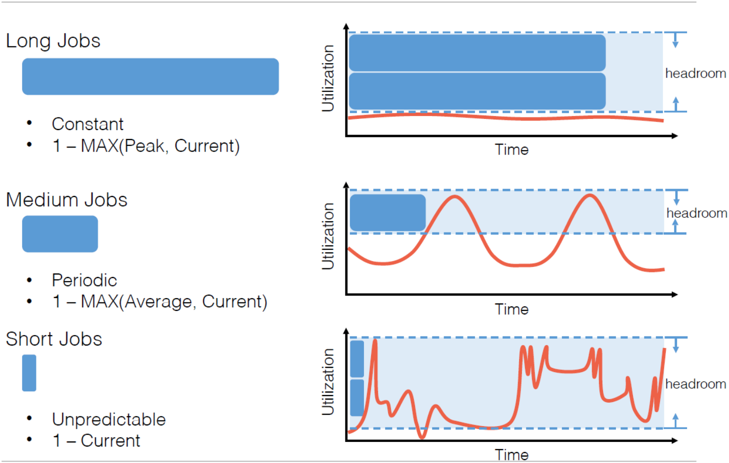
2 Behavior Patterns

3 Co-location Techniques

4 Experiment

5 Conclusion

History-Based Task Scheduling



Task Scheduling

- Fast Fourier Transform (FFT)
  - Get 3 patterns.
- Clustering algorithm
  - K-means algorithm
  - Average and peak utilizations
- Class selection algorithm

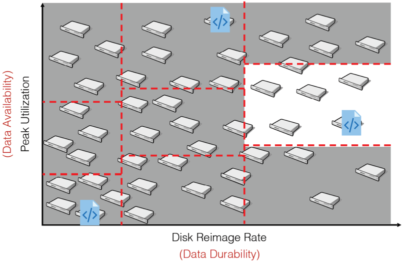
Class Selection Algorithm

Algorithm 1 Class selection algorithm.

Given: Classes C, Headroom($type$,c), Ranking Weights W
function SCHEDULE(Batch job J)
    J.type = Length (short, medium, or long) from its last run
    J.req = Max amount of concurrent resources from DAG
    for each $c \in C$ **do**
        c.weightedroom=Headroom(J.type,c) $\times$ W[J.type,c.class]
    end for
    $F = \{\forall c \in C |$ Headroom(J.type,c) $\geq J$.req$\}$
    if $F \neq \emptyset$ **then**
        Pick 1 class $c \in F$ probabilistically $\propto c$.weightedroom
        return $\{c\}$
    else if Job J can fit in multiple classes combined **then**
        Pick $\{c_0, \ldots, c_k\} \subseteq C$ probabilistically $\propto c$.weightedroom
        return $\{c_0, \ldots, c_k\}$
    else
        Do not pick classes
        return $\{\emptyset\}$
    end if
end function

History-Based Data Placement

- Data availability
  - Diverse in utilization pattern.
- Data durability
  - Diverse in reimaging pattern.



Replica Placement Algorithm

Algorithm 2 Replica placement algorithm.

```
Given: Storage space available in each server, Primary reimaging
       stats, Primary peak CPU util stats, Desired replication R
function PLACE REPLICAS(Block B)
    Cluster primary tenants wrt reimaging and peak CPU util
        into 9 classes, each with the same total space
    Select the class of the server creating the block
    Select the server creating the block for one replica
    for r = 2; r ≤ R; r = r + 1 do
        Select the next class randomly under two constraints:
            No class in the same row has been picked
            No class in the same column has been picked
        Pick a random primary tenant of this class as long as
            its environment has not received a replica
        Pick a server in this primary tenant for the next replica
        if (r mod 3) == 0 then
            Forget rows and columns that have been selected so far
        end if
    end for
end function
```

1 Introduction

2 Behavior Patterns

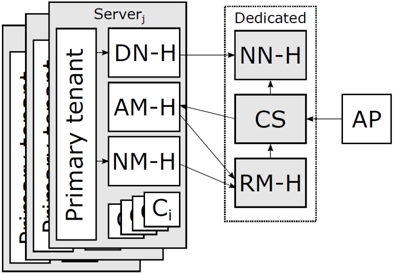
3 Co-location Techniques

4 Experiment

5 Conclusion

System Implementation

- Overview of YARN-H, Tez-H, and HDFS-H in a co-location scenario.

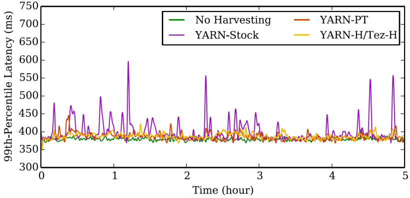


Evaluation

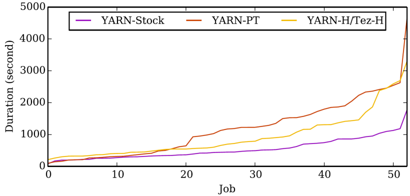
- Environment:
  - 102-server cluster
- Primary tenant (interactive service):
  - Apache Lucene search engine with utilization trace
- Batch task:
  - TPC-DS benchmark

Batch Task Scheduling

- Interactive services performance

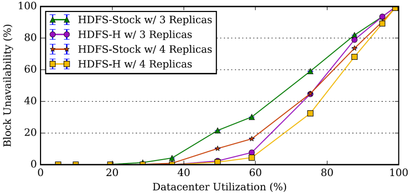


- Job duration - reducing task killing

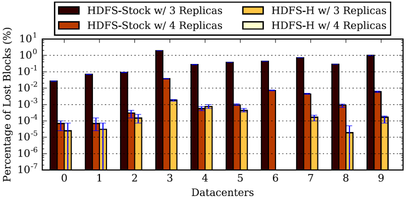


Data Placement

- Data availability



- Data durability



1 Introduction

2 Behavior Patterns

3 Co-location Techniques

4 Experiment

5 Conclusion

Conclusion

- Embody knowledge of the existing primary workloads, and leverage historical utilization.
- Improve batch job performance while protecting primary workloads.
- Eliminate data loss and unavailability in many scenarios.

Future Work

- Paper
  - Isolation and security in public cloud.
- Project
  - Network utilization.
  - Constraint of Ethernet bandwidth.