PriDynSim: A Simulator for Dynamic Priority Based I/O Scheduling for Cloud Applications

Nitisha Jain*, J. Lakshmi*
Nikolay Grozev‘, Rajkumar Buyya‘

*Indian Institute of Science, Bangalore, India
‘The University of Melbourne, Parkville, Australia

November 26, 2015

IEEE CCEM 2015, Bangalore
Outline

1. Overview
2. PriDynSim
3. Architecture
4. Implementation
5. Performance Evaluation
6. Conclusions
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
Cloud computing - popular model for renting computational resources as per need.

- On-demand availability, scalability of IT resources, lower costs.
- This flexibility is important for enterprises in emerging markets, provides competitive advantages.
- One of the major impediments for the research in the area -> lack of affordable testing environments.
Simulation tools

- Provide a good alternative to testing in large scale and expensive testbeds.
- Controlled environment for hypotheses evaluation.
- Allow emerging enterprises to quickly evaluate novel resource management techniques.
- Fostering innovation, putting them at par with competitors.
CloudSim Tookit

- Developed by Rajkumar Buyya et. al. at CLOUDS lab, *University of Melbourne*

- Most widely used tool for Cloud simulation.

- Provides flexible, customizable platform for modeling of Cloud data-centers, services, brokers, virtualized servers.
CloudSim Toolkit

- Developed by Rajkumar Buyya et. al. at CLOUDS lab, University of Melbourne
- Most widely used tool for Cloud simulation.
- Provides flexible, customizable platform for modeling of Cloud data-centers, services, brokers, virtualized servers.

Limitation
Lacks support for disk I/O operations and policies for allocation of I/O resources to concurrent I/O workloads.
Suitable I/O performance models extending the widely adopted CloudSim simulator can further research in I/O resource management.

Need for an I/O resource scheduler, to assign resources to applications based on their specific characteristics and requirements.

*PriDynSim* (**Dynamic Priority Simulator**)  
Generic *CloudSim* based simulator for priority based I/O resource scheduling.
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
PriDynSim Toolkit

- Facilitates evaluation of policies for dynamic I/O resource scheduling across co-located heterogeneous applications.

- Consider requirements of wide spectrum of I/O applications, guarantee performance QoS.

- Incorporates the representation of latencies, deadlines of applications in simulation environment.

- Facilitate future research efforts in development of time-constrained scheduling policies.
Related Work

- **CloudSim** - most closely related to our work, but lacks support for representation of deadline driven I/O workloads, priority based scheduling policies.

- Extensions of **CloudSim**:
  - **WorkflowSim** - Support for workflow simulations and scheduling algorithms.
  - **DynamicCloudSim** - Extends **CloudSim** to handle heterogeneity of applications, dynamic changes to the performance.
  - **CloudReports** - Provides GUI for simulating techniques for power optimization in Cloud computing environments.
Related Work

- **CloudSim** - most closely related to our work, but lacks support for representation of deadline driven I/O workloads, priority based scheduling policies.

- Extensions of **CloudSim**:
  - **WorkflowSim** - Support for workflow simulations and scheduling algorithms.
  - **DynamicCloudSim** - Extends CloudSim to handle heterogeneity of applications, dynamic changes to the performance.
  - CloudReports - Provides GUI for simulating techniques for power optimization in Cloud computing environments.

None of them address performance modeling of concurrently executing latency sensitive I/O workloads, common in real-life Cloud setups.
GreenCloud, MDCSim focus on energy consumption and power optimization for multi-tier Cloud data-centers, but no consideration to application performance.

ICanCloud enables simulation on larger scale spanning multiple machines, but lacks any support for I/O operations or priority scheduling policies.

None of the available simulation platforms have attempted to design priority based I/O scheduling policies to address the performance concerns for I/O intensive workloads in a Cloud environment like PriDynSim.
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
Proposed Extensions

- **HddCloudletEx**
  - *Deadline* - Time by which the cloudlet is expected to finish completion.
  - *Start Time* - Time at which the cloudlet is assigned the disk IOPS for executing I/O operations.
  - *IOPS* - Value of disk IOPS assigned to the cloudlet at a given time.
Proposed Extensions

- **HddCloudletEx**
  - *Deadline* - Time by which the cloudlet is expected to finish completion.
  - *Start Time* - Time at which the cloudlet is assigned the disk IOPS for executing I/O operations.
  - *IOPS* - Value of disk IOPS assigned to the cloudlet at a given time.

- **PriDynSim Scheduler**
  - Replaces existing time shared fair scheduler.
  - Measures requirements of cloudlets as per their I/O operations, deadline values, and assigns IOPS to satisfy deadlines.
Proposed Extensions

- **HddCloudletEx**
  - **Deadline** - Time by which the cloudlet is expected to finish completion.
  - **Start Time** - Time at which the cloudlet is assigned the disk IOPS for executing I/O operations.
  - **IOPS** - Value of disk IOPS assigned to the cloudlet at a given time.

- **PriDynSim Scheduler**
  - Replaces existing time shared fair scheduler.
  - Measures requirements of cloudlets as per their I/O operations, deadline values, and assigns IOPS to satisfy deadlines.

- **DatacenterBrokerEx**
  - Acts on behalf of the user for creation and destruction of VMs and submission of cloudlets to the VMs.
  - Can handle the submission of **HddCloudletEx** entities with I/O operations, deadlines.
PriDynSim Class Diagram
Functionality

Interaction Diagram
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
## PriDyn Algorithm Terminology

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloudlet</td>
<td>$C = \langle C(1), C(2) \ldots C(N) \rangle$</td>
</tr>
<tr>
<td>Deadline</td>
<td>$D = \langle D(1), D(2) \ldots D(N) \rangle$</td>
</tr>
<tr>
<td>Total Data Size</td>
<td>$R = [R_1, R_2 \ldots R_N]$</td>
</tr>
<tr>
<td>Number of IO Operations</td>
<td>$IO = \langle IO(1), IO(2) \ldots IO(N) \rangle$</td>
</tr>
<tr>
<td>Start Time</td>
<td>$ST = \langle ST(1), ST(2) \ldots ST(N) \rangle$</td>
</tr>
<tr>
<td>Disk IOPS</td>
<td>$IOPS = \langle IOPS(1), IOPS(2) \ldots IOPS(N) \rangle$</td>
</tr>
<tr>
<td>Latency</td>
<td>$L = \langle L(1), L(2) \ldots L(N) \rangle$</td>
</tr>
<tr>
<td>Priority Cloudlet</td>
<td>$C_{Priority}$</td>
</tr>
<tr>
<td>Priority IOPS</td>
<td>$IOPS_{Priority}$</td>
</tr>
<tr>
<td>Maximum IOPS</td>
<td>$IOPS_{max}$</td>
</tr>
</tbody>
</table>
Algorithm - Priority Scheduler I

**Require:** \( N_{\text{Total}}, C, \text{Total IO}, D, ST \)

**Ensure:** \( \text{IOPS} \)

1. Initialize \( \text{IOPS}_{\text{Priority}} \) as 0
2. for each \( C(i) \) in \( < C(1) \ldots C(N) > \) do
3. Call \( \text{IOPS}_\text{Manager}(N, \text{IOPS}_{\text{Priority}}) \)
4. end for
5. while \( (N > = 0) \) do
6. for each \( C(i) \) in \( < C(1) \ldots C(N) > \) do
7. Calculate \( L(i) = \text{IOPS}_{(i)}/\text{IO(i)} \)
8. end for
9. if (exists \( C(i) \) s. t. \( L(i) > D(i) \)) then
10. Find \( C(i) \) where \( D(i) \) is minimum
11. if \( ((\text{IO}_{(i)}/D(i)) - ST(i)) > \text{IOPS}_{\text{max}} \) then
12. Continue to next \( C(i) \)
13. else
14. \( C_{\text{Priority}} = C(i) \)
Algorithm - Priority Scheduler II

15: \[ IOPS_{\text{Priority}} = \frac{IO(i)}{(D(i) - ST(i))} \]
16: \hspace{10pt} \text{end if}
17: \hspace{10pt} \text{end if}
18: \textbf{for All } C(i) \textbf{ in } < C(1)...C(N) \textbf{ do}
19: \hspace{10pt} \textbf{if } C(i) = C_{\text{Priority}} \textbf{ then}
20: \hspace{20pt} IOPS(i) = IOPS_{\text{Priority}}
21: \hspace{10pt} \textbf{else}
22: \hspace{20pt} \textbf{Call } IOPS\_Manager(N, IOPS_{\text{Priority}})
23: \hspace{10pt} \textbf{end if}
24: \hspace{10pt} \textbf{end for}
25: \textbf{end while}
Algorithm - IOPS Manager

Require: $N$, $IOPS_{Priority}$
Ensure: $IOPS(i)$

1: if $(IOPS_{Priority} = 0)$ then
2: \hspace{1cm} $IOPS(i) = IOPS_{max}/N$
3: else
4: \hspace{1cm} $IOPS(i) = (IOPS_{max} - IOPS_{Priority})/(N - 1)$
5: end if
6: return $IOPS(i)$
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
Using Real I/O Workload Traces

- Real world I/O traces used to understand multi-tenanted Cloud setups.\(^1\)
- Block I/O traces from servers at Microsoft Cambridge.
- Consist of diverse applications like web server, media server, research projects etc.
- Trace requests were modelled to represent I/O application stream running on a virtual machine on the server.
- Deadline values assigned to requests based on their characteristics and I/O size.

\(^1\)http://www.iotta.snia.org/
Deadline Assignment for I/O Requests

- Makespan - Min time for completing I/O request.

\[ Makespan = \frac{IO}{IOPS(max)} \]

- Delay tolerance parameter \( \delta \), represents functional characteristics of request.

\[ \text{Deadline} = \text{rand} \left( \text{Makespan}, \text{Makespan} + (\text{Makespan} \times \delta) \right) \]
Case Study

- Experimental evaluation with varying number, combinations of cloudlets.
- Set of cloudlets with equal number of latency sensitive, delay tolerant jobs modeled by the I/O requests belonging to Media server and Research server.
Case Study

- Experimental evaluation with varying number, combinations of cloudlets.
- Set of cloudlets with equal number of latency sensitive, delay tolerant jobs modeled by the I/O requests belonging to Media server and Research server.

![Comparison of Missed Deadlines for 10 Cloudlets](image-url)

Comparison of Missed Deadlines for 10 Cloudlets
Comparison of Missed Deadlines for 15 Cloudlets
Measure of *deviations* of cloudlets from their deadlines, i.e. the difference between the values of response times (the time at which the cloudlet finished execution) and the value of deadline assigned to it.

Comparison of Deviations from Deadlines
1. Overview

2. PriDynSim

3. Architecture

4. Implementation

5. Performance Evaluation

6. Conclusions
PriDynSim Benefits

- **PriDynSim** -> Explore policies for dynamic allocation of disk resources to I/O bound applications.

- Gives prioritized disk access to latency-sensitive jobs.

- Guaranteed application performance for a wide variety of typical Cloud workloads modeled by real world I/O traces.

- Designed as a generic simulator -> can be used as a testbed by future researchers for evaluation of own specific scheduling policies.
Future Work...

Extension to data-center level, allocation of cloudlets or job to a VM based on the application requirements -> optimize overall efficacy of the resource allocation for Cloud data-center.

Acknowledgements

We thank Rodrigo Calheiros, Amir Vahid Dastjerdi, Yaser Mansouri, and Chenhao Qu for their comments on improving this work. This work is partially supported by the Melbourne-Chindia Cloud Computing (MC3) Research Network.
Publications


Thank You

Questions ?