PCOS : Prescient Cloud I/O Scheduler for Workload Consolidation and Performance

Nitisha Jain
Research Guide: Dr J. Lakshmi

Cloud Systems Lab, SERC
Indian Institute of Science
Bangalore, India

November 4, 2015
Outline

1. Overview
2. Need for Meta-scheduling
3. PCOS Framework
4. Conclusions
1. Overview

2. Need for Meta-scheduling

3. PCOS Framework

4. Conclusions
Cloud computing enabled by virtualization:
- Better utilization of physical resources.
- Energy savings.
Cloud computing enabled by virtualization:
- Better utilization of physical resources.
- Energy savings.

But..
- Sharing of resources $\rightarrow$ performance interference.
- Multiple VMs on 1 physical machine $\rightarrow$ unpredictable delays, degradation of performance.
Cloud computing enabled by virtualization:

- Better utilization of physical resources.
- Energy savings.

But..

- Sharing of resources $\Rightarrow$ performance interference.
- Multiple VMs on 1 physical machine $\Rightarrow$ unpredictable delays, degradation of performance.

Trade-off between Application Performance and Workload Consolidation!
- Focus on I/O workloads.
  - Different latency and throughput requirements.
- Fair and equal allocation $\rightarrow$ Latency sensitive applications may suffer undesirable delays.
- Need for differentiated services.

---

Focus on I/O workloads.
  ▶ Different latency and throughput requirements.
  ▶ Fair and equal allocation → Latency sensitive applications may suffer undesirable delays.
  ▶ Need for differentiated services.

**PriDyn (Dynamic Priority) Scheduler**

- Performance-driven latency-aware application scheduler.
- Dynamically computes latency estimates for all concurrent I/O applications.
- Determines priority assignment for underlying disk scheduler.

---

▶ At Cloud data center level, need for intelligent scheduling of I/O workloads.
▶ Optimal combination of I/O applications \(\rightarrow\) max resource utilization with good performance.
At Cloud data center level, need for intelligent scheduling of I/O workloads.

Optimal combination of I/O applications → max resource utilization with good performance.

**PCOS (Prescient Cloud I/O Scheduler) Framework**

- Proactive meta-scheduling framework for Cloud storage.
- Admission control for selecting suitable workload mix.
- Enables server consolidation with guaranteed performance.
1. Overview

2. Need for Meta-scheduling

3. PCOS Framework

4. Conclusions
Different Workload Combinations

<table>
<thead>
<tr>
<th>Application features</th>
<th>Application A</th>
<th>Application B</th>
<th>Application C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency Sensitive?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Disk Priority</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
</tr>
<tr>
<td>Case 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency Sensitive?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Disk Priority</td>
<td>Default</td>
<td>Default</td>
<td>Low</td>
</tr>
</tbody>
</table>

Response Time for Application A in Case 1
Response Time for Application A in Case 2

Deadline Violations for Applications
1. Overview

2. Need for Meta-scheduling

3. PCOS Framework
   Design and Implementation
   Experimental Validation

4. Conclusions
Prescient Cloud I/O Scheduler

PRESCIENT CLOUD I/O SCHEDULER (PCOS)

HYPervisor
ADMISSION CONTROLLER [PROACTIVE]

PRIDYN

DISK

---

Overview
Need for Meta-scheduling
PCOS Framework
Design and Implementation
Experimental Validation
Conclusions
Features

- Proactive approach for meta-scheduling.

- *PCOS* ensures optimal workloads on all servers with admission controller.

- Assigns suitable server for all new I/O requests.

- Gives higher priority to scheduled applications, avoid migration overheads.

- Two main components – *AdCon* module and *PriDyn* scheduler working together.
PCOS Design

- **PRESIENT CLOUD I/O SCHEDULER**
  - New I/O Request [Size, Deadline]
  - Priority Database

- **I/O Admission Controller**
  - Proactive Agent
    - Accept (Priority)
    - New Priorities

- **PRIDYN**
  - Disk Scheduler

- **Overview**
  - Need for Meta-scheduling

- **PCOS Framework**
  - Design and Implementation
  - Experimental Validation

- **Conclusions**
Admission Controller (AdCon)

Input: Size, deadline of new I/O application request.

- Collect information about current resource allocation, priorities of applications using PriDyn.
- **Proactive Agent** - Anticipate system behavior if new request is scheduled using Priority Database.
- If deadline violations expected, search suitable priorities using PriDyn.

Output: **Accept** or **Reject** new I/O request.
**Priority Database**

- Stores expected disk bandwidth allocation based on system history, number and priorities of the applications.
- Iterative learning database, continuously updated for different set of I/O applications.

**PriDyn Scheduler**

- Assist AdCon to find suitable priority combination for given application set.
- Implement the disk allocation if new request accepted by AdCon.
**PCOS Algorithm**

**Require:** DataSize $R_{new}$, Deadline $D_{new}$

**Ensure:** Server $S_r$ for scheduling

1: for each server do
2: Call AdCon($R_{new}, D_{new}$)
3: if Accept new then
4: Schedule new request
5: else
6: Continue
7: end if
8: end for
Current I/O applications $N$, request for $N+1$ ...

**Case 1**

Deadline violated for one or more applications in $<1...N>$, deadline satisfied for $N+1$.

- Priority of the new request decreased if possible.
- Potential latencies recalculated, start over.
Current I/O applications $N$, request for $N+1$.

**Case 1**

Deadline violated for one or more applications in $<1...N>$, deadline satisfied for $N+1$.

- Priority of the new request decreased if possible.
- Potential latencies recalculated, start over.

**Case 2**

Deadline violated for one or more applications in $<1...N>$, deadline violated for $N+1$.

- New request rejected for the system at present state.
- Considered again when system state changes.
Case 3

Deadline satisfied for all applications in $<1...N>$, deadline satisfied for $N+1$.

- New request accepted on the system with the assigned priority.
Case 3
Deadline satisfied for all applications in $< 1...N >$, deadline satisfied for $N + 1$.

- New request accepted on the system with the assigned priority.

Case 4
Deadline satisfied for all applications in $< 1...N >$, deadline violated for $N + 1$.

- Attempt to adjust priorities of applications to get suitable combination to achieve performance, call Priority Manager module of PriDyn scheduler.
Results

Two media applications executing concurrently on VMs, sharing disk bandwidth

- Case 1: Web server application scheduled, latency sensitive.

Performance of web server requests with media applications
Case 2: Research application scheduled, latency insensitive.

Performance of research requests with media applications
Comparison of number of requests scheduled
Total Disk Bandwidth Utilization with *PCOS* framework

Application A, B : Media Requests
Application C : Research Requests
1. Overview

2. Need for Meta-scheduling

3. PCOS Framework

4. Conclusions
Summary

**PriDyn scheduler**
- Dynamic scheduling framework, cognizant of the latency requirements of applications to enable differentiated I/O services.

**PCOS framework**
- Proactive scheduling to achieve the balance between resource consolidation and application performance guarantees in Cloud environments.
Limitations

- Proposed framework - extract good disk resource utilization but not guarantee all deadlines.
- Participation of physical device is necessary in resource allocation, placement strategies.
- Significant changes to the architecture, hardware support for virtualization required for fine grained performance control, QoS guarantees.
Limitations ..

- Proposed framework - extract good disk resource utilization but not guarantee all deadlines.
- Participation of physical device is necessary in resource allocation, placement strategies.
- Significant changes to the architecture, hardware support for virtualization required for fine grained performance control, QoS guarantees.

Future work ..

- Demonstrate performance of proposed frameworks for environments having virtualization-enabled hardware.


Thank You

For questions, please contact authors at nitishajain15@gmail.com or sercnitisha@ssl.serc.iisc.in
Overview
Need for Meta-scheduling
PCOS Framework
  Design and Implementation
  Experimental Validation
Conclusions
Require: DataSize $R_{N+1}$, Deadline $D_{N+1}$
Ensure: Accept $N + 1$ ($Pr_{N+1}$) or Reject $N + 1$

1: Find Current State ($N, < R, D, B, S, Pr >$), default $Pr_{N+1}$
2: Call PROACTIVE AGENT($N + 1, Pr_{<1...N+1>}$)
3: while (1) do
4:   Find $i$ s.t. $L_i > (D_i - (T - S_i))$ [ $i$ in $< 1...N >$]
5:   if (exists $i$) then
6:     if ($L_{N+1} < (D_{N+1})$) & ($Pr_{N+1}$ > lowest) then
7:       Decrease $Pr_{N+1}$
8:       Call PROACTIVE AGENT($N+1, Pr_{<1...N+1>}$)
9:     else
10:       Reject $N + 1$
11:   end if
12:   else  \> deadlines met for all $i$ in $< 1...N >$
13:     if ($L_{N+1} < (D_{N+1})$) then
14:       Accept $N + 1$, ($Pr_{N+1}$)
15:     else
16:       Call PRIORITY MANAGER($L_{<1...N+1>}, D_{<1...N+1>}$)
17:     end if
18:   end if
Proactive Agent

\[
\text{PROACTIVE AGENT}(N+1, \text{Pr}_{1...N+1})
\]
1: Search Priority Database
2: Update Bandwidth \( B_{1...N+1} \)
3: Execute LATENCY PREDICTOR(\( R_{1...N+1}, B_{1...N+1} \))
4: for all \( i \) in \( 1...N+1 \) do
5: \( \text{RemainingData}_i = R_i - \text{DataProcessed}_i \)
6: \( L_i = \text{RemainingData}_i / B_i \)
7: end for
8: return Latency \( L_{1...N+1} \)
PriDyn Algorithm

**Require:** Deadline $D$, TotalDataSize $R$

**Ensure:** Priority $Pr$

1. **LATENCY PREDICTOR($R, B$)**
   a. `for every process $P_i$ do`
   b. $RemainingData_i = R_i - DataProcessed_i$
   c. $L_i = RemainingData_i / B_i$
   d. `end for`

2. **return Latency $L$**

3. **PRIORITY MANAGER($L, D$)**
   a. Find $P_i$ s.t. $(L_i > (D_i - T_i))$ & $D_i$ is minimum
   b. if (exists $P_i$) then
      c. Find all $P_j, (j \neq i)$ s.t. $(D_j > D_i)$ & $(L_j < (D_j - T_j))$
      d. Select $P_j$ s.t. $(Pr_j > lowest)$ & $(P_j < highest)$ is maximum
   e. if (exists $P_j$) then
      f. Decrease $Pr_j$
   g. else
      h. if ($Pr_i < highest$) then
         i. Increase $Pr_i$
      j. else
         k. Set $Pr_i$ to lowest
      l. Restore $Pr_j$
   m. end if
   n. end if
   o. `end if`
PriDyn Algorithm

- Feedback based design.
- If latency of critical process P expected to be violated,
  - Case 1: Increase disk priority of P if possible, else,
  - Case 2: Decrease priority of other non-critical processes if possible, else,
  - Case 3: If deadlines cannot be satisfied, give lowest priority to P, identify process for migration.
- Critical process gets respectable performance even in worst case, finish execution earlier than estimated latency value.
- Acceptable services ensured for the non-critical processes.
To be noted:

- Complexity of algorithm is \( N \), where \( N \) is the number of active concurrent processes.

- It is able to meet desired deadlines for latency sensitive applications for all values within the performance bounds of the system.
Cloud based storage environments host a wide range of heterogeneous I/O intensive applications.

Varied latency bounds and bandwidth requirements.

Co-located applications get shared disk bandwidth, may affect SLAs.

Scheduling plays an important role in ensuring performance with resource consolidation.
Deadline Assignment for I/O Requests

- **Makespan** - Min time for completing I/O request.
- **BWLoss** - Loss of disk bandwidth due to contention for resources, proportional to number of VMs.
- **Makespan** = \( \frac{IOSize}{((MaxBW-BWLoss)/N)} \)
- **Delay Tolerance Parameter** \( \delta \) - Based on latency characteristics of application.
- **Deadline** = **Makespan** + (**Makespan** * \( \delta \))

Calculation of **BWLoss** Parameter
Priority Manager

\[ \text{PRIORITY \ MANAGER}(L^{<1\ldots N+1>}, D^{<1\ldots N+1>}) \]

1: \textbf{for} \( j \) in \( 1 \ldots N \) \textbf{do}
2: \hspace{1em} \text{Find all} \ j \ \text{s.t.} \ (Pr_j > \text{lowest})
3: \hspace{1em} \textbf{end for}
4: \textbf{if} (exists \( j \)) \textbf{then}
5: \hspace{1em} \text{Select} \ j \ \text{s.t.} \ ((D_j - (T - S_j)) - L_j) \text{ is maximum}
6: \hspace{1em} \text{Decrease} \ Pr_j
7: \hspace{1em} \text{Call} \ \text{PROACTIVE AGENT}(N + 1, Pr^{<1\ldots N+1>})
8: \hspace{1em} \textbf{else}
9: \hspace{2em} \textbf{if} (Pr_{N+1} < \text{highest}) \textbf{then}
10: \hspace{3em} \text{Increase} \ Pr_{N+1}
11: \hspace{3em} \text{Call} \ \text{PROACTIVE AGENT}(N + 1, Pr^{<1\ldots N+1>})
12: \hspace{2em} \textbf{else}
13: \hspace{3em} \text{Reject} \ N + 1
14: \hspace{2em} \textbf{end if}
15: \hspace{1em} \textbf{end if}
16: \textbf{return}