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 –> performance interference.
- Multiple VMs on 1 physical machine –> unpredictable delays, degradation of performance.

Cloud computing enabled by virtualization :

- Better utilization of physical resources.
- Energy savings.

But..

- Sharing of resources –> performance interference.
- Multiple VMs on 1 physical machine –> 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 –> Latency sensitive applications may suffer undesirable delays.
- Need for differentiated services.

[1] "*PriDyn : Framework for Performance Specific QoS in Cloud Storage*", Proceedings of IEEE CLOUD 2014, June 27 - July 2, 2014, Alaska, USA.

- 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.

[1]

[1] "*PriDyn : Framework for Performance Specific QoS in Cloud Storage*", Proceedings of IEEE CLOUD 2014, June 27 - July 2, 2014, Alaska, USA.

- 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.

- 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 (*P*rescient *C*loud *I/O* *S*cheduler) Framework

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

1. Overview

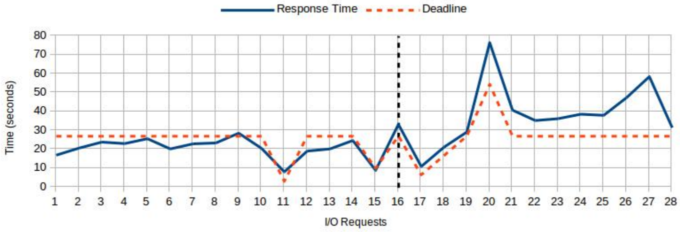
2. Need for Meta-scheduling

3. PCOS Framework

4. Conclusions

Different Workload Combinations

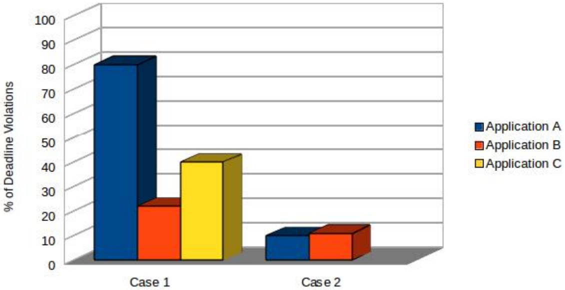
	Application features	Application A	Application B	Application C
Case 1	Latency Sensitive? Disk Priority	Yes Default	Yes Default	Yes Default
Case 2	Latency Sensitive? Disk Priority	Yes Default	Yes Default	No Low



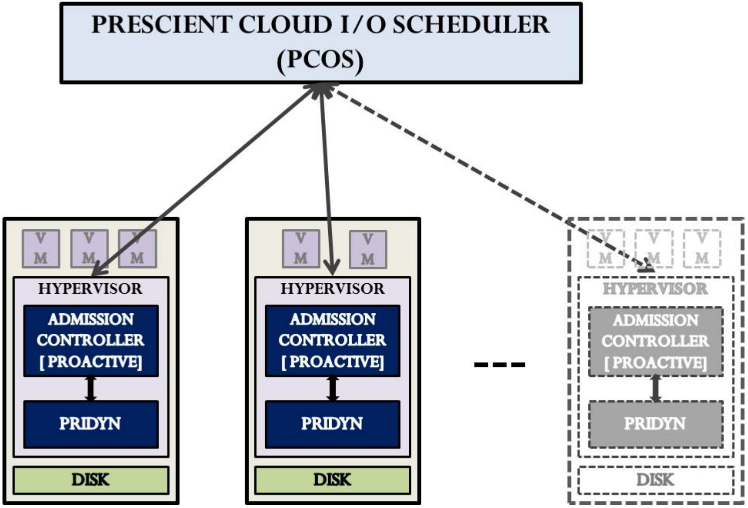
Response Time for Application A in Case 1

Response Time for Application A in Case 2

Deadline Violations for Applications



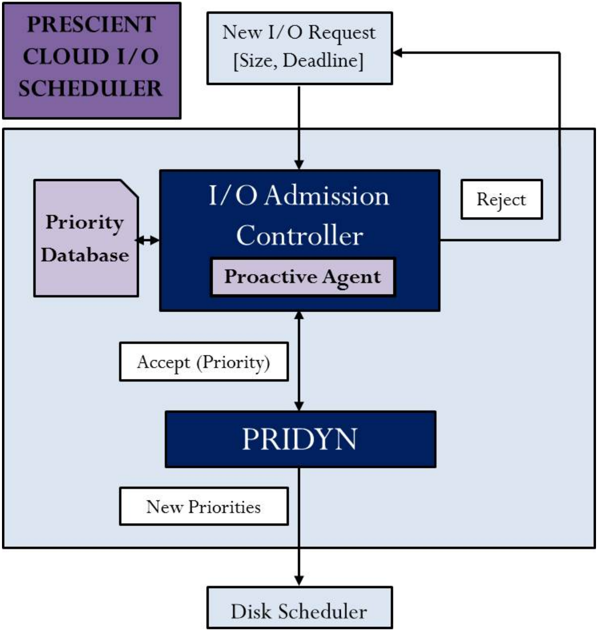
Prescient Cloud I/O Scheduler



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


PRESCIENT CLOUD I/O SCHEDULER
New I/O Request [Size, Deadline]
Priority Database
I/O Admission Controller
Proactive Agent
Reject
Accept (Priority)
PRIDYN
New Priorities
Disk Scheduler

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*

for *each server* **do**
    Call AdCon(R_{new}, D_{new})
    if *Accept new* **then**
        Schedule new request
    else
        Continue
    end if
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 $< 1 \ldots N >$, 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.

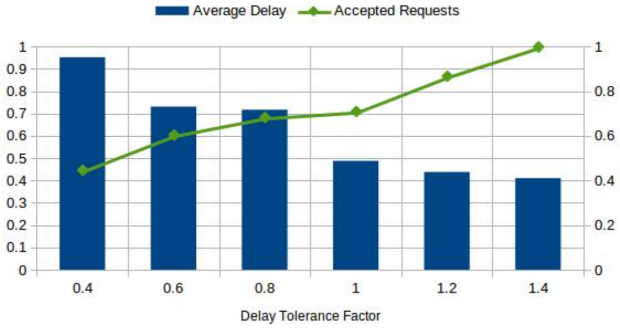
Results

Two media applications executing concurrently on VMs, sharing disk bandwidth

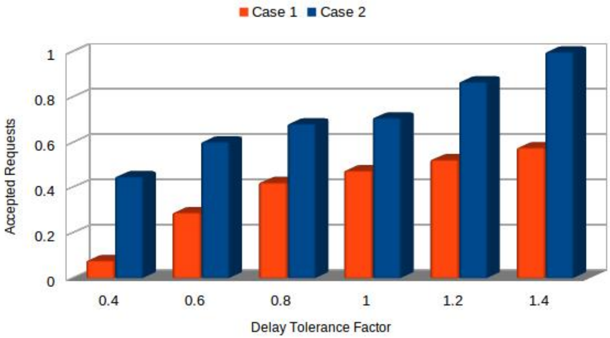
- Case 1: Web server application scheduled, latency sensitive.

Performance of web server requests with media applications

- Case 2: Research application scheduled, latency insensitive.



Performance of research requests with media applications

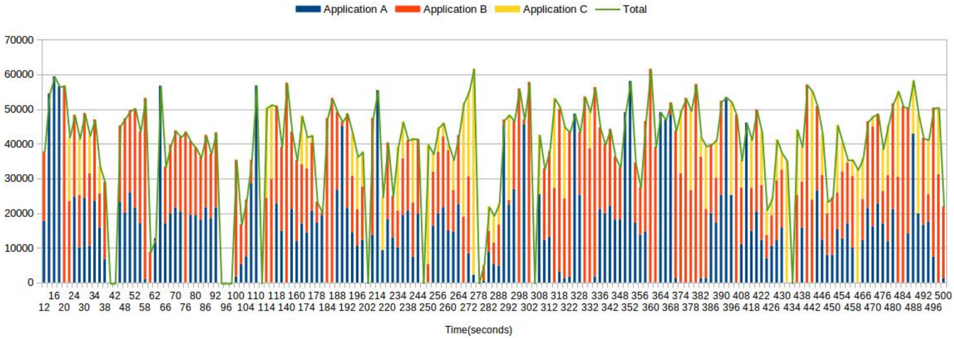


Comparison of number of requests scheduled

Total Disk Bandwidth Utilization with *PCOS* framework

Application A, B : Media Requests

Application C : Research Requests



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.

Publications

1. Nitisha Jain, J. Lakshmi, "PriDyn : Enabling Differentiated I/O Services in Cloud using Dynamic Priorities", *IEEE Transactions on Services Computing (Special Issue on Cloud Computing)*, vol. PP, no. 99, 2014.

2. Nitisha Jain, J. Lakshmi, "PriDyn : Framework for Performance Specific QoS in Cloud Storage", *Proceedings of the 7th IEEE International Conference on Cloud Computing (IEEE CLOUD 2014)*, June 27 - July 2, 2014, Alaska, USA.

3. Nitisha Jain, Nikolay Grozev, Rajkumar Buyya, J. Lakshmi, "PriDynSim : A Simulator for Dynamic Priority Based I/O Scheduling", accepted at the *3rd IEEE International Conference on Cloud Computing in Emerging Markets (CCEM 2015)*, November 25 - 27, 2015, Bangalore, India.

Thank You

For questions, please contact authors at
nitishajain15@gmail.com or sercnitisha@ssl.serc.iisc.in

Require: *DataSize* R_{N+1}, *Deadline* D_{N+1}

Ensure: *Accept* $N+1$ (Pr_{N+1}) *or Reject* $N+1$

Find Current State $(N, < R, D, B, S, Pr >)$, *default* Pr_{N+1}

Call PROACTIVE AGENT$(N+1, Pr_{<1...N+1>})$

while (1) **do**

    Find i s.t. $L_i > (Di - (T - S_i))$ $[i$ *in* $< 1...N >]$

    if (*exists* i) **then**

        if $(L_{N+1} < (D_{N+1}))$ & $(Pr_{N+1} >$ *lowest*$)$ **then**

            Decrease Pr_{N+1}

            Call PROACTIVE AGENT$(N+1, Pr_{<1...N+1>})$

        else

            Reject $N+1$

        end if

    else ▷ *deadlines met for all i in* $< 1...N >$

        if $(L_{N+1} < (D_{N+1}))$ **then**

            Accept $N+1$, (Pr_{N+1})

        else

            Call PRIORITY MANAGER$(L_{<1...N+1>}, D_{<1...N+1>})$

        end if

    end if

Proactive Agent

$PROACTIVE\ AGENT(N+1, Pr_{<1...N+1>})$

$Search\ Priority\ Database$
$Update\ Bandwidth\ B_{<1...N+1>}$
$Execute\ LATENCY\ PREDICTOR(R_{<1...N+1>}, B_{<1...N+1>})$
for $all\ i\ in < 1...N+1 >$ **do**
    $RemainingData_i = R_i - DataProcessed_i$
    $L_i = RemainingData_i / B_i$
end for
return $Latency\ L_{<1...N+1>}$

PriDyn Algorithm

Require: $Deadline\ D, TotalDataSize\ R$
Ensure: $Priority\ Pr$

$LATENCY\ PREDICTOR(R, B)$

for $every\ process\ P_i$ **do**
  $RemainingData_i = R_i - DataProcessed_i$
  $L_i = RemainingData_i / B_i$
end for
return $Latency\ L$

$PRIORITY\ MANAGER(L, D)$

$Find\ P_i$ s.t. $(L_i > (D_i - T_i))$ & $D_i\ is\ minimum$
if $(exists\ P_i)$ **then**
  $Find\ all\ P_{j,(j!=i)}$ s.t. $(D_j > D_i)$ & $(L_j < (D_j - T_j))$
  $Select\ P_j$ s.t. $(Pr_j > lowest)$ & $((D_j - T_j) - L_j)\ is\ maximum$
  if $(exists\ P_j)$ **then**
    $Decrease\ Pr_j$
  else ▷ $If\ no\ such\ P_j\ exists$
    if $(Pr_i < highest)$ **then**
      $Increase\ Pr_i$
    else
      $Set\ Pr_i\ to\ lowest$
      $Restore\ Pr_j$
    end if
  end if
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.

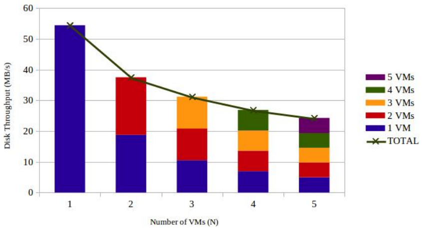
To be noted..

- 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* = *IOSize* / ((*MaxBW*-*BWLoss*)/*N*)
- *Delay Tolerance Parameter* δ - Based on latency characteristics of application.
- *Deadline* = *Makespan* + (*Makespan* * δ)



Calculation of *BWLoss* Parameter

Priority Manager

```
PRIORITY MANAGER(L<1...N+1>, D<1...N+1>)
for j in <1...N> do
    Find all j s.t. (Pr_j > lowest)
end for
if (exists j) then
    Select j s.t. ((D_j − (T − S_j)) − L_j) is maximum
    Decrease Pr_j
    Call PROACTIVE AGENT(N + 1, Pr<1...N+1>)
else
    if (Pr_{N+1} < highest) then
        Increase Pr_{N+1}
        Call PROACTIVE AGENT(N + 1, Pr<1...N+1>)
    else
        Reject N + 1
    end if
end if
return
```