PYTHIA: Improving Datacenter Utilization via Precise Contention Prediction for Multiple Co-located Workloads

Ran Xu (Purdue), Subrata Mitra (Adobe Research), Jason Rahman (Facebook),

Peter Bai (Purdue), Bowen Zhou (LinkedIn),

Greg Bronevetsky (Google), Saurabh Bagchi (Purdue)

CRISP

Outline

- Motivation and Background
- Observations on Prior Solutions for Contention Characterization
- Our Solution: PYTHIA
- Evaluation of PYTHIA
- Conclusion and Insights

Data Center Utilization is Low

- Total cost of ownership of datacenters is huge but utilization is low
  - 30-40% is common in the best managed data centers
- Multi-core processors enable multi-way co-location of applications on same server
  - Lots of processing power on one node
  - Lots of memory on one node
- Reason for low utilization?

Only a single application runs on one server

- Why?

Paranoia

Fear of delaying Latency Sensitive (LS) applications

PURDUE UNIVERSITY

Background: Contention

- Applications compete for underlying resources and cause resource contention
  - Compute cores
  - Memory (capacity plus transfer bandwidth)
  - Network
- Partitioning in hardware works well for *some* resources
  - Compute cores ✓
  - Memory (capacity plus transfer bandwidth) ✗
  - Network ✗
- Need to protect latency sensitive workloads from contentions to maintain Quality of Service (QoS)

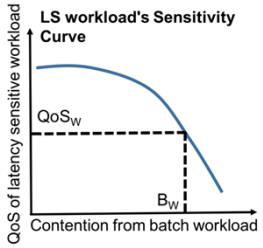
PURDUE UNIVERSITY

Outline

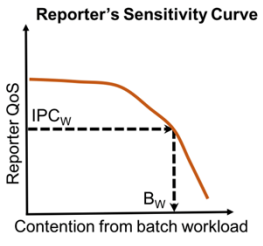
- Motivation and Background
- Observations on Prior Solutions for Contention Characterization
- Our Solution: PYTHIA
- Evaluation of PYTHIA
- Conclusion and Insights

Prior Art: Characterize Contention

- Characterize:
1. How much contention a batch application causes
2. How much contention a Latency Sensitive (LS) application can tolerate
- Assign a Bubble score to batch workloads depending on how much contention it creates for LS workloads
- Use a generic "Reporter" application as a stand-in for LS workloads
  - Reduces characterization overhead from M×N to M



(a) Co-location of batch workload and LS workload

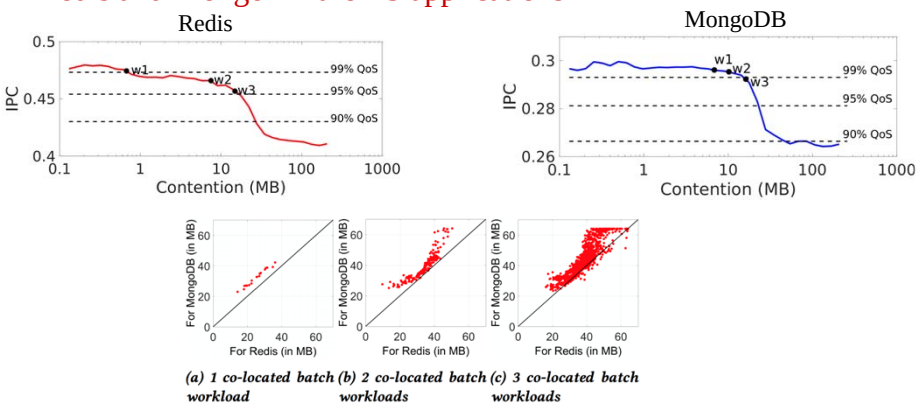


(b) Co-location of batch workload and Reporter

Bubble-Up: MICRO-2011

Our Observation #1: Contention Effect is LS workload specific

- Redis and MongoDB are LS applications



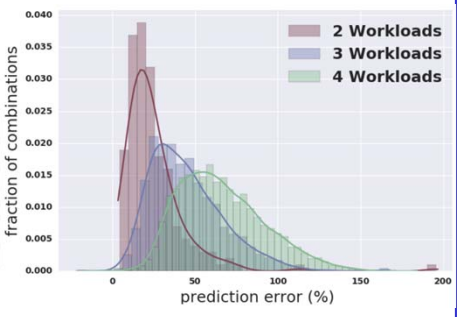
(a) 1 co-located batch workload *(b) 2 co-located batch workloads* *(c) 3 co-located batch workloads*

MongoDB sensitivity to contention > Redis sensitivity

PURDUE UNIVERSITY

Our Observation #2: Contention Effect Changes with degree of co-location

Batch workloads	Observed Contention (KB)	Estimated Contention (KB)	Error
SpecBwaves, SpecBzip	11694	14268	22%
ParsecStreamcluster, SpecBwaves, SpecZeusmp	12270	21630	76%
SpecTonto, SpecGcc	9704	14060	44%
ParsecFerret, SpecOmnetpp	5404	6596	22%
SpecBwaves, SpecH264, SpecOmnetpp	12533	17385	38%
SpecGromacs, SpecHmmer, SpecXalancbmk	7044	8226	16%
SpecMiLS, SpecBzip, SpecBzip	9127	11540	26%
ParsecBodytrack, ParsecCanneal, SpecMcf	8403	12895	53%



- Multiple co-locations make combined content prediction more challenging
  - Simple additive model does not work
  - Simple model overestimates contention caused by combination of batch workloads
- Root cause: Mutual contention – Multiple workloads interfere with each other
  - Error increases with degree of co-location

PURDUE UNIVERSITY

Outline

- Motivation and Background
- Observations on Prior Solutions for Contention Characterization
- Our Solution: PYTHIA
- Evaluation of PYTHIA
- Conclusion and Insights

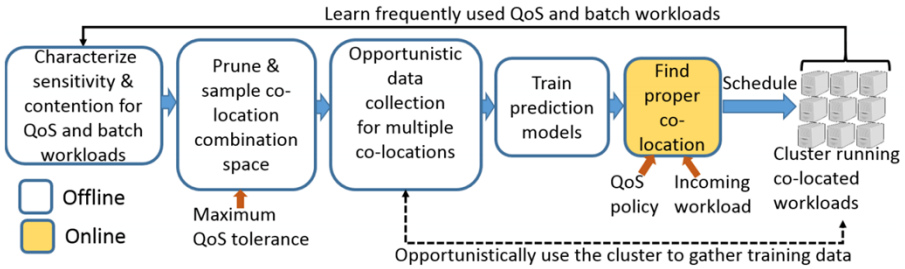
PURDUE UNIVERSITY

Our Solution: PYTHIA

- PYTHIA does workload-aware modeling of contention
  - Goal: A single LS application co-located with multiple batch applications
- It creates a model for multi-way co-locations
  - Simple linear regression model
  - Easily fit with limited amounts of training data
- PYTHIA can provide very accurate predictions for the contention induced by *multiple* co-located batch workloads
  - Can be used for initial placement of workloads when submitted to the cloud reducing the likelihood of contention
- PYTHIA incorporates dynamic monitoring and control schemes
  - Reduces priority of batch application
  - Needed, hopefully infrequently, to deal with unexpected changes in application behavior

PURDUE UNIVERSITY

PYTHIA Workflow



PURDUE UNIVERSITY

PYTHIA: Mathematical Basis

- Modeling Mutual Contention: $B_S = \sum_{W_i \in S} c_{W_i} B_{W_i}$

  c_{W_i} : Tendency of batch-workload W_i to suffer from other co-located workloads due to mutual contention

  B_{W_i} : Contention (bubble-size) created batch-workload W_i when run alone with the LS workload

- Learning coefficients from a system of equations by fitting observed data

$$\begin{pmatrix} \xi_{1,1}B_{W_1} & \xi_{1,2}B_{W_2} & \cdots & \xi_{1,M}B_{W_M} \\ \xi_{2,1}B_{W_1} & \xi_{2,2}B_{W_2} & \cdots & \xi_{2,M}B_{W_M} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{k,1}B_{W_1} & \xi_{k,2}B_{W_2} & \cdots & \xi_{k,M}B_{W_M} \end{pmatrix} \begin{pmatrix} c_{W_1} \\ c_{W_2} \\ \vdots \\ c_{W_M} \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_k \end{pmatrix}$$

Solve using least-square optimization

$\xi_{i,j}$: Number of times a batch workload W_i appeared in i^{th} workload combination

B_i : Observed combined contention for i^{th} workload combination

- The weights c_{W_i} in our model represent the tendency for the associated batch workload W_i to suffer interference from other batch workloads
  - Higher c_{W_i} $\Rightarrow$ Batch application is more resistant to mutual contention
  - Therefore it exerts higher cache and memory pressure on LS workload as a result of co-location

PURDUE UNIVERSITY

PYTHIA Placement Algorithm

- *x%* QoS policy ⇒ Minimum acceptable QoS is *x%* of the uninterfered performance

```
PYTHIA_Find_Best_Server {
W <= Incoming workload
P <= QoS policy
FOR Server in List of Servers:
    qos = PYTHIA_Predict_QoS (Server->Existing_Workload
+ W)
    IF qos >  P :
        IF Server->Available Resource <
Lowest_Available_Resource :
            Server_Chosen = Server
            Lowest_Available_Resource = Server->Available
Resource

return Server_Chosen
}
```

- Best Fit
- Worst Fit
- First Fit

Outline

- Motivation and Background
- Observations on Prior Solutions for Contention Characterization
- Our Solution: **PYTHIA**
- Evaluation of **PYTHIA**
- Conclusion and Insights

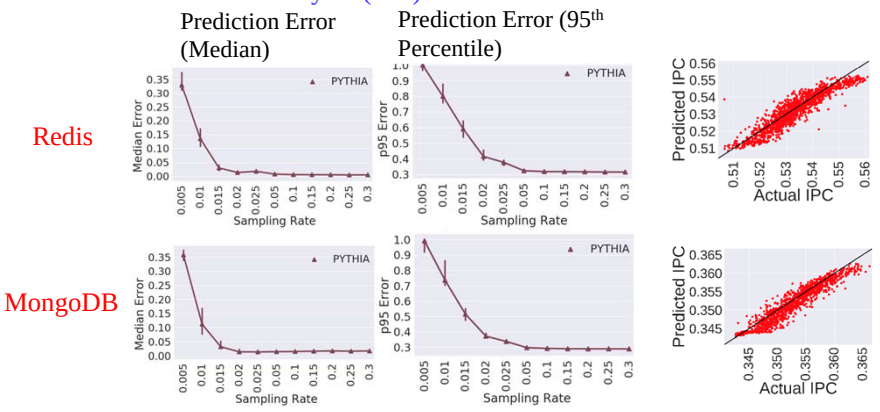
Evaluation Basics

- PYTHIA implemented using a combination of Python and shell scripts and consists of 5.5 KLOC
- LS and batch workloads run on *different* cores
  - Pin memory allocation on same NUMA socket as workload
- Applications:
  - Two popular LS applications: Redis and MongoDB driven using YCSB
  - Large variety of representative batch workloads drawn from the widely used PARSEC and SPEC2006 benchmark suites
  - 19 of 26 are co-locatable (4 PARSEC, 15 SPEC)
- Run on a 1,296-node homogeneous production cluster
- Each node: 2 sockets, 8 × 2 cores, 2.60 GHz, 64-bit Intel Xeon E5-2670 processor cores, 32 GB memory, 20 MB L3 cache (LLC) shared between cores on each socket
- 2 cores per workload ⇒ maximum degree of co-location = 3 (one socket dedicated to experimental monitoring)

PURDUE UNIVERSITY

Prediction Accuracy

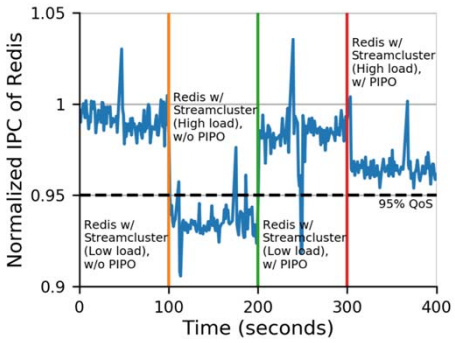
- Metric: Instructions Per Cycle (IPC). Lower is better.



- With just 5% sampling rate on the search space PYTHIA achieves decent accuracy in predicting IPC

PURDUE UNIVERSITY

Dynamic Throttling of Batch Workloads

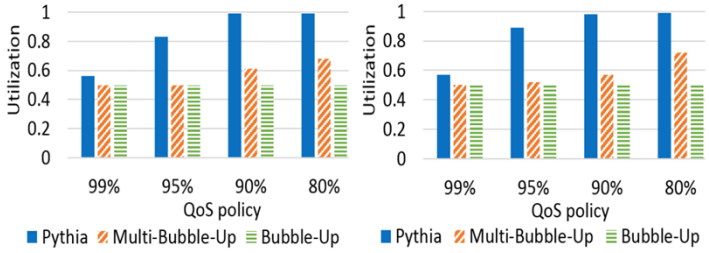


- QoS of LS application Redis always stays above the QoS threshold, even under high load

PURDUE UNIVERSITY

Improvement in Cluster Utilization

- Metric: Cluster Utilization = # Cores occupied / Total # cores in the cluster
  - Higher is better

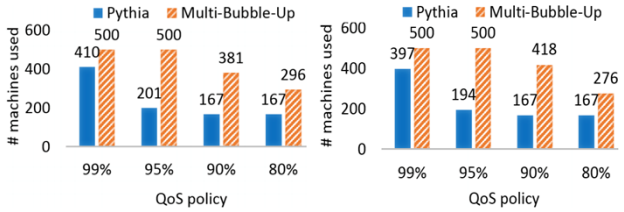


(a) Utilization with MongoDB *(b) Utilization with Redis*

- If QoS budget is tight, any solution will work as well
- For 95% QoS policy PYTHIA provides 72% improvement over Multi-Bubble-Up
- For 80% QoS policy PYTHIA provides close to 100% utilization

PURDUE UNIVERSITY

Reduction in # Active Machines



(a) # Servers for MongoDB *(b) # Servers for Redis*

- PYTHIA can provide much tighter packing, reducing number of active machines required for placement
- With 80% QoS policy, number of active machines required to handle the incoming task is reduced by 44%

PURDUE UNIVERSITY

Conclusions and Insights

- Multi-way co-location is crucial for increasing datacenter utilization
  - With multi-way co-location contention characteristics change
  - Simple additive contention prediction is erroneous
- Impact of contention is LS workload dependent
  - Increases offline profiling effort for characterizing contention
- PYTHIA can accurately predict contention for higher degree of co-location
  - Estimate contention due to multi-way co-locations using simple linear regression model
  - Use estimate for initial placement and dynamically throttle batch workload if needed
- PYTHIA can significantly improve cluster utilization
- Drawback: Does not work well for unseen workloads

PURDUE UNIVERSITY