ProRenaTa: Proactive and Reactive Tuning to Scale a Distributed Storage System

Ying Liu, Navaneeth Rameshan, Enric Monte, Vladimir Vlassov and Leandro Navarro


Summarized by Anush Neelakantam
OUTLINE

- ABSTRACT
- INTRODUCTION
- KEY OBSERVATIONS & BACKGROUND
- SYSTEM DESIGN
- ALGORITHM
- WORKLOAD PREDICTION
- CONTROLLER MODELS
- EVALUATION
ABSTRACT

- Provisioning stateful services in the Cloud at high QoS and low hosting cost is a challenge.
- Two approaches generally used:
  - Prediction based controller, which looks ahead of time and gives the system time to adjust.
  - Feedback based controller, which bases the scaling on previous metrics and achieves high accuracy.
- The paper shows the limitations of using the above mentioned approaches in isolation and presents ProRenaTa, which combines both the proactive and reactive approaches to scale resources for distributed storage systems.
- It also includes a data migration model to manage the resource scaling overhead that is incurred.
- Evaluations against state of the art methods show improvements in resource utilization.
INTRODUCTION

- Elastic provisioning is a lucrative approach to hosting services in the Cloud as instances are only spawned when needed and then removed when the workload drops. (Pay as you use)
- Elastic provisioning is done through elasticity controllers which helps reduce the hosting cost and maintain the SLAs that were negotiated. SLA maintenance is a key issue as it could lead to penalties and loss of profits.
- The paper studies the elastic scaling of distributed storage systems which are generally state based. This also involves considering the scaling overhead involved when scaling the resources.
An efficient controller must do the following:
- Constraint satisfaction regarding the SLA and cost.
- Adaptation to workload changes without oscillations.
- Must be aware of scaling overheads
- Must use resources efficiently.

As mentioned before, it is done in 2 ways:
- Reactive control: Uses a feedback loop to react to the system metric values.
  Pros: Accurate
  Cons: Delayed system adaptation. Initial phases of SLA violations are experienced.
- Proactive control: Uses a workload prediction approach to prepare the system in advance for the workload changes.
  Pros: SLA violations in initial phase are not experienced as the workload change is already being predicted.
  Cons: May lead to inaccuracies as the prediction highly depends on the application specific access patterns.
The authors present ProRenaTa, which combines the advantages of the reactive and proactive scaling techniques and achieves resource provisioning without any resource oscillations by making accurate workload predictions.

The controller also includes a data migration model which considers the scaling overhead involved (the cost for data migration).

Finally, it uses the reactive model to guarantee the further accuracy of resource allocations.
KEY OBSERVATIONS & BACKGROUND

- Distributed storage services:
  - Provide unified storage services to clients by combining and integrating large number of storage instances.

- Advantageous because of high scalability and availability.

- Examples: Hadoop, Cassandra, Dynamo, Spanner, PNUT.

- The authors use GlobLease as the underlying storage system for ProRenaTa.

- GlobLease is a key-value store that uses Distributed Hash Tables. It is setup using a read/write consistency of “ONE”. This ensures that the request is processed by any node in the cluster than can process it.
KEY OBSERVATIONS & BACKGROUND

- Distributed storage services:
  - Provide unified storage services to clients by combining and integrating large number of storage instances.
  - Advantageous because of high scalability and availability.
  - Examples: Hadoop, Cassandra, Dynamo, Spanner, PNUT.

- The authors use GlobLease as the underlying storage system for ProRenaTa
- GlobLease is a key – value store that uses Distributed Hash Tables. It is setup using a read/write consistency of “ONE”. This ensures that the request is processed by any node in the cluster than can process it.
KEY OBSERVATIONS

- The authors setup experiments to study the scaling of GlobLease using a proactive and reactive approach.

- The prediction based approach looks to add instances to answer the workload change. However, this still leads to SLA violation because of the extra overhead of data redistribution. The data migration model of ProRenaTa is specifically intended to solve this problem.

- We still run into the problem of inaccuracy.

- Even when using the feedback approach, the system observed SLA violations because the instances added cannot serve the workload immediately. The data copying takes time and the system observes SLA violations.

- Ultimately, we note that using both the approaches together yields much better resource utilization.

Fig. 1: Observation of SLA violations during scaling up. (a) denotes a simple increasing workload pattern; (b) scales up the system using a proactive approach; (c) scales up the system using a reactive approach.
The design of the ProRenaTa follows the MAPE – K control loop with certain modifications.

The input workload is the arrival rate of reads and writes on each node. This is fed to the Workload pre-process module and the ProRenaTa scheduler.

**Monitor**

- **Workload pre-process module:**
  - Sums up the workload that is being monitored over a smoothing window.
  - The size of the smoothing window is application specific as if it is too small, causes resource oscillations and if it is too large, miss out on sudden workload changes.

- **The ProRenaTa Scheduler** estimates the system utilization and calculates the spare capacity to handle scaling overheads.
SYSTEM DESIGN

ANALYSIS:

- The analysis phase consists of the **Workload Prediction module**

- **Workload Prediction: Associates a timestamp to the workload.**
  
  The pre-processed workload is sent to this module.
  
  The workload is predicted over every prediction window (PW). At the beginning of each PW, this module basically predicts the workload intensity at the end of the current PW.
  
  The output after this module is the aggregated workload intensity along with the timestamp that indicates the time to finish the scaling of resources for the predicted workload.
SYSTEM DESIGN

PLAN:

- The total workload is sent forward to the **Proactive and Reactive Schedulers**.

  The proactive scheduler decides the number of instances to be added in the next prediction window (PW).

  The reactive scheduler calculates the scaling inaccuracy as the difference between the predicted workload at the start of PW and the observed workload at the end of the PW.

  If this value is greater than a pre defined threshold given by the throughput performance model, then the number of instances to be added or removed are sent to the ProRenaTa scheduler.
SYSTEM DESIGN

PLAN:

ProRenaTa scheduler:

- Gathers instructions as to how many instances are to added or removed from the two previously mentioned schedulers.

- Looks to efficiently conduct the scaling adjustment immediately and then looks to conduct the scaling plan for the future next. If the two plans contradict or interfere with each other, the scheduler consults the data migration model which quantifies the spare capacity needed to handle the scaling overhead.

- The individual plans on their own can easily be executed in a single PW. However, there may be a case where we might observe interference and both controller plans may overlap each other.
SYSTEM DESIGN

PLAN:

ProRenaTa scheduler:

- **When no interference occurs**, both the scheduler plans are implemented with the reactive plan being implemented first.

- **When interference occurs**, the difference between the two plans is computed and implemented as a proactive plan at the end of the PW.

EXECUTE:

- Uses the scaling actuator module: to interact with the GlobLease storage and adds and removes instances through the add or remove server APIs.

- Also controls the data migration by using the BwMan, which allocates quota limits for the migration on each storage node.
SYSTEM DESIGN

KNOWLEDGE:
ProRenaTa uses three knowledge bases:

▶ Throughput model
  Correlates the read and write requests under SLA constraints.
▶ The migration overhead model
  System capacity to perform data migration while system reconfiguration
▶ Monitoring
  Provides workload intensity details for assist the scheduler in accurate decisions.
**ALGORITHM FOR ProRenaTa**

**Proactive control:**
Predicts workload for next PW.
Consults the throughput model and then instructs the scheduler to add or remove instances.

**Reactive Control:**
Computes the difference between the observed workload intensity and the predicted workload intensity.
Determines if any adjustments need to be made to the scaling of the previous PW.

**ProRenaTa Scheduler:**
If the plans for both schedulers are not executable in a single window, then the difference is computed and executed as a proactive plan at the end of the PW.
If the plans can be executed, then the reactive plan is executed first and then the proactive plan follows.

---

**Algorithm 1 ProRenaTa Control Flow**

```plaintext
1: procedure PROACTIVECONTROL() > Program starts at time $T_i$
   2: $PW.T_{i+1} \leftarrow \text{workloadPrediction(Trace)}$
   3: $VM.s.T_{i+1} \leftarrow \text{throughputModel}(PW.T_{i+1})$
   4: $\Delta VM.s.T_{i+1} \leftarrow V M.s.T_{i+1} - V M.s.T_i$

6: procedure REACTIVECONTROL() > Program starts at time $T_i$
   7: $\Delta W.T_i \leftarrow W.T_i - PW.T_i$
   8: $\delta VM.s.T_i \leftarrow \text{throughputModel}(\Delta W.T_i)$

10: procedure PRORENA TaSCHEDULER() > Program starts at $T_i$
   11: $R.S.T_i \leftarrow \text{dataMigrationModel}(T_i)$
   12: $T.p \leftarrow \text{analyticalModel}(\Delta VM.s.T_{i+1}, R.S.T_i)$
   13: $T.r \leftarrow \text{analyticalModel}(\delta VM.s.T_i, R.S.T_i)$
   14: if $T.p + T.r > T_{i+1} - T_i$ then
       15: $VM.s.toChange \leftarrow \Delta VM.s.T_{i+1} + \delta VM.s.T_i$
       16: $t \leftarrow \text{analyticalModel}(VMsToChange, R.S.T_i)$
       17: $TimeToAct \leftarrow T_{i+1} - t$
       18: WaitUntil $TimeToAct$
       19: ConductSystemResize($VMsToChange$)
   20: else
       21: ConductSystemResize($\delta VM.s.T_i$)
       22: $TimeToAct \leftarrow T_{i+1} - T.p$
       23: WaitUntil $TimeToAct$
       24: ConductSystemResize($\Delta VM.s.T_{i+1}$)
```
WORKLOAD PREDICTION

Uses two modules for prediction. The Short term module for predicting the workload in stable load and cyclic behaviour and random noises.

Uses the Wiener filter, which linearly combines past samples of the workload to predict the future workload.

The coefficients of this linear prediction is computed by minimizing the Mean Square Error (MSE) between the predicted and a reference sample.

If this error is increasing for a certain length of time, the filter weights are recomputed.

11: procedure SHORTTIMEPREDICTION\(\omega_l, N_{FrHr}, x\)
12: \[\hat{x}_{Shrt}[n + N_{FrHr}] = \sum_{i=0}^{L_{Shrt}} w_i x[n - i]\]
13: > Compute \(\hat{x}_{Shrt}[n + N_{FrHr}]\) from a linear combination of the data in a window of length \(L_{Shrt}\).
WORKLOAD PREDICTION

The long term prediction module is used to predict the behaviour for workloads having periodic but sudden peaks.

The rises seen here also depend on past values. However, they depend on values much more in the past than the other module.

A list defining the periodicities for the sampling frequencies are set and the decision to forecast the workload is based on testing if the final estimation \( n + N_{FHR} \) is a multiple of the list of the frequencies.

\begin{verbatim}
21: procedure LONGTIMEPREDICTION(h_i, j, P_i, L_i, x)
22: \( \hat{x}_{L_{no}}[n + N_{FHR}] = \sum_{i=0}^{N_p} \sum_{j=0}^{L} h_{i,j} x[n - jP_i] \)
\end{verbatim}

\( \hat{x}_{L_{no}}[n + N_{FHR}] \) from a linear combination of the data in a window of length corresponding to the periods \( P_i \).
CONTROLLER MODELS

**Throughput Performance Model**

Determines the number of servers needed to meet the SLA requirements for a specific workload.

The key step is to build a performance model is to see how the workload is being distributed.

**Load Balance:** Enabling virtual tokens was seen as a good approach as it helped the servers host discrete virtual tokens and store the corresponding data.

It was noted that assigning more number of virtual tokens to a server helped uniformly distribute the workload among the servers. (The standard deviation was observed to decrease).

This helped study the system by studying the individual servers and it was seen that if the sample server did not violate SLA, the system did not violate SLA.

![Graph showing standard deviation of load on 20 servers running a 24 hours Wikipedia workload trace. With larger number of virtual tokens assigned to each server, the standard deviation of load among servers decreases.](image)
CONTROLLER MODELS

Throughput Performance Model

The workload is represented by the request rate of read and write operations.

Under the specified SLAs, the server can either be violating or satisfying the SLA.

The model reads in the workload intensity and tells the number of instances to be added to bring the system under the SLA.

The idea is to keep the server just under the SLA so that the resource utilization is maximum.

Fig. 5: Throughput performance model
CONTROLLER MODELS

- **Data migration model**
  - Studies the workload and determines the maximum rate at which the data can be transferred while scaling so that the SLAs are not violated.
  - It helps ProRenaTa determine the time taken to add/remove the said instances.

- **Statistical Model**: The spare capacity that can be used for a data migration can be determined in the following way.
  - The workload is mapped to each server and is expressed as a data point. The closest border below this point reflects the data transfer speed for data migration on that server.

![Fig. 6: Data migration model under throughput and SLA constraints](image-url)
EVALUATION

- **Experimental Setup:**
  - The workload used was synthesised from the Wikipedia access logs over two weeks.
  - Workload generated in JAVA to emulate workloads of different granularities of rps.
  - Study 5% of the most accessed pages for reading.

- Configured the system to handle 50 concurrent requests.
- 3000-7000 requests per second.
EVALUATION

- Compare ProRenaTa with baseline approaches: feedback control and prediction – based control.

- Also compare ProRenaTa with an ideal case which is a theoretical controller that predicts the future workload perfectly.

- Experiment 1:

  - Compare the 4 approaches with the Wikipedia workload trace over two weeks. The focus is on the 95th percentile latency of requests calculated from each hour

  - It is seen that the feedback approach is the worst as it violates SLA the most. This is because the reaction to changing workload is too late.

  - ProRenaTa and proactive approach similar because of the accurate predictions.
EVALUATION

Experiment 2:

Compare the 4 approaches with the Wikipedia workload trace over two weeks. The focus is on the CPU utilization.

The feedback approach experiences resource under utilization and also high saturation. Leads to high cost and SLA violation respectively.

The prediction approach also experienced under utilization because, when the workload decreased, the instances were removed in the next control period. This lead to resource under utilization during the current control period.

The proposed approach had stable resource utilization between 50 - 80%. The working of the two approaches simultaneously helped maintain the CPU utilization at a respectable level.

Fig. 8: Aggregated CDF of CPU utilization for different approaches
EVALUATION

- **Experiment 3:**
  - Study the approaches by collecting the data during 48 hours.
  - The figure shows that the workload predicted during the 48 hours by the prediction module is highly accurate.

- **Total number of VMs used during the 48 hours:**
  - The prediction approach uses more VMs than ideal and the feedback approach uses fewer VMs than ideal. ProRenaTa is very close to the ideal number of VMs used.

Fig. 9: Actual workload and predicted workload and aggregated VM hours used corresponding to the workload
EVALUATION

- **Experiment 4:**
  - SLA commitment:
  - Feedback controller violates the SLA the most as it fails to respond to every workload increase in time.
  - The prediction approach and ProRenaTa are almost similar but it is seen that ProRenaTa is better and it is also better in terms of resource utilization as seen earlier.

Fig. 10: SLA commitment comparing ideal, feedback and predict approaches with ProRenaTa
Utility Measure: The utility measure of the system is the ability of the system to achieve low latency and high CPU utilization simultaneously.

The utility in this case is given by:

\[ U = \text{VM} \text{ hours} + \text{Penalty} \]

Penalty = DurationOfSLAViolations \* penalty_factor.

4 different penalty factors were used and it was seen that as the penalty grew higher, ProRenaTa achieved lower utility, which in turn meant lower penalty.