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

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


Summarized by Kevin Andrea
Overview

- Introduction
- Background
  - Chord
  - GlobLease
- System Design
- Control Flow
- Controller Models
- Evaluation

Image: Chord Diagram by Seth Terashima, distributed under a CC-BY 3.0 License
Introduction

• Problem Statement

  • Auto-Scaling Distributed Storage Systems to Handle Demand

• Two General Approaches
  • Proactive
    • React to observed metrics, such as incoming read and write requests and CPU utilization.
    • Scale the system in response to workload changes.
  • Reactive
    • Explore historic access patterns to predict future workload levels.
    • Scale the system ahead of predicted workload changes.
Introduction – Auto-Scaling

• Elastic Provisioning
  • Use resources on demand to save hosting costs.
    • Add instances on demand to handle increases in workloads.
    • Remove instances when workloads decrease, based on utilization analysis.
  • Assumption that the customer only pays for used resources.

• Elastic Controller
  • Goal is to reduce costs without compromising Service Level Agreements (SLA)
    • Violating a SLA incurs penalty costs for both the provider and the customer.
  • Swiftly react to workloads without oscillation.
  • Utilize resources efficiently.
Introduction – Approaches to Auto-Scaling

• Reactive Control
  • Scales well and accurately, as it is based on observed workloads.
  • Problem:
    • Reacts to workload changes only after they occur, leading to SLA violations until the new instances are online to handle demand.

• Predictive Control
  • Prepares instances in advance to allow scaling without service disruptions.
  • Problem:
    • If the predicted model is not accurate, this will result in both under and over-utilization of resources and violations of the SLA.
Introduction – ProRenaTa

• ProRenaTa
  • Auto-scaling system for distributed storage systems.
  • Implements both Reactive and Proactive modules.
    • Improves resource utilization and reduces SLA violations.

• Achieves four properties:
  • Constraint satisfaction for minimizing SLA violations and provisioning costs.
  • Swift adaptation to workload changes without oscillations.
  • Uses awareness of scaling overheads to minimize SLA violations.
  • Efficient use of resources during scaling.
Background – Distributed Storage Systems

• Distributed Storage Systems provide unified storage to its clients.
  • These systems integrate and manage a large number of backend storage instances.
  • Typical Implementations:
    • Hadoop, Cassandra, Dynamo, Spanner, PNUT

• GlobLease (Similar to Cassandra)
  • GlobLease is the underlying storage system for ProRenaTa
    • Distributed Storage System created by the same author of ProRenaTa (Ying Liu)
  • Uses a Distributed Hash Table
  • Provides key-value storage for request routing.
  • Implemented using a Read/Write Consistency Level of “ONE”
    • This means requests for either action are sent to the closest node that can service the request.
Background – Distributed Hash Table (DHT)

- Distributed Hash Table (DHT)
  - Provides a scalable, distributed means of locating a Value using a Key.
  - Uses consistent hashing to assign keys.
    - All nodes receive roughly the same number of keys.
  - Uses an identifier circle for routing.
    - Removing a node only affects $O(1/N)$ keys.

- Figure shown is for Chord
  - Chord is a DHT implementation referenced by ProRenaTa.
  - Values associated with keys may be full files.
    - (Wikipedia Page Entries)

Background – GlobLease

• GlobLease
  • This is the backend storage system for ProRenaTa
  • Replication is achieved by multiple DHTs
    • Concentric Rings
  • Assumption is that most clients are readers.

• Provides low-latency read-access at a global scale.
  • Efficient write-access at a regional level.
• Multiple DHTs for geo-replication.
• Routing links add over time from request echos.
  • Echos contain keys and IP address that the serving node is responsible for.
  • Over time, all nodes create links to all other nodes.

Background – GlobLease Leases (Virtual Tokens)

• GlobLease
  • ProRenaTa references Virtual Tokens.
    • Leases in GlobLease.

• Lease
  • Token for serving read accesses.
  • Only valid for a specified time interval.
  • Allows an affiliated node to be added with little overhead.
    • Only the master node is involved with the addition
    • Can be easily added to meet increase workload demands
  • Write requests can only be handled by the master node.
    • Solves the consistency problem. All affiliated nodes with a lease are invalidated or updated by the master node.

Background – Observations

• Even with perfect prediction, SLA can still be violated.
  • There is overhead involved with adding new instances.
    • Data migration of the key and values (data) to the new leased instance.

• Each traditional approach must address this problem:
  • Predictive – Data transfers ahead of the predicted increase may still break the SLA (b)
    • Requires a data migration model.
  • Reactive – Problem with feedback. In addition to increased workload breaking the SLA, adding new instances add to the network traffic for data migration, exacerbating the problem. (c)
    • Solution is to scale the system prior to workload changing.
System Design

- ProRenaTa follows MAPE-K model

- M – Monitor
  - **Workload Pre-Process Module**
  - **ProRenaTa Scheduler** Module

- Workload Pre-Process Module
  - Records the monitored workload over a Smoothing Window (SW) interval.
  - The size of the SW is adjusted to prevent oscillations.
    - Too small and it will lead to frequent changes.
    - Too large and it will be unresponsive.
System Design

• ProRenaTa follows MAPE-K model

• M – Monitor
  • Workload Pre-Process Module
  • ProRenaTa Scheduler Module

• ProRenaTa Scheduler Module
  • Estimates the system utilization and calculates the spare capacity to handle scaling overhead.
  • Conducts the scaling plan (using data from the planning step)
System Design

• ProRenaTa follows MAPE-K model

• A – Analysis
  • Workload Prediction Module

• Workload Prediction Module
  • Takes the pre-processed input and performs workload forecasting using a Prediction Window (PW)
    • Forecasts the workload intensity at the end of the current PW
    • PW and SW are both the same size in ProRenaTa
  • Outputs a predicted workload intensity with a timestamp.
    • The timestamp is the deadline to finish scaling for the predicted intensity.
  • The total workload is sent to both Predictive and Reactive schedulers.
System Design

• ProRenaTa follows MAPE-K model

• P – Plan
  • Proactive Scheduler Module
  • Reactive Scheduler Module

• Proactive Scheduler
  • Calculates the number of instances needed in the next Prediction Window (PW)
  • Sends this number to the ProRenaTa Scheduler Module
System Design

• ProRenaTa follows MAPE-K model

• P – Plan
  • Proactive Scheduler Module
  • Reactive Scheduler Module

• Reactive Scheduler
  • Corrects inaccurate scaling caused by inaccurate prediction
  • Takes the predicted workload at the beginning of the PW and compares it with the observed workload at the end of the PW.
  • If the scaling inaccuracy reaches a threshold level for a specified duration, then a schedule to add or remove instances is sent to the ProRenaTa Scheduler.
System Design

• ProRenaTa follows MAPE-K model

• ProRenaTa Scheduler
  • Receives both Scheduler plans.
  • Consults with the Data Migration Model, which quantifies the spare capacity needed to handle the scaling overhead introduced by the plans.
    • This migration model estimates the time needed to handle the scaling overhead.

Tr – Time to finish the Reactive Plan
Tp – Time to finish the Predictive Plan
System Design

- ProRenaTa follows MAPE-K model

- ProRenaTa Scheduler
  - If there is sufficient time, both schedules are implemented.
    - Each adds or removes instances.
  - Otherwise, the difference between the two plans is calculated and is implemented at the end of the PW, as if it were the proactive plan.
System Design

• ProRenaTa follows MAPE-K model

• E – Execute
  • **Scaling Actuator** Module

• Scaling Actuator
  • Interacts with the underlying storage system.
  • Calls the add or remove instance APIs
  • Controls data migration using a quota system.
System Design

• ProRenaTa follows MAPE-K model

• K – Knowledge
  • Throughput Model
    • Correlates read and write requests under SLA constraints.
  • Migration Overhead Model
    • Quantifies the spare capacity to handle the migration overhead.
  • Monitoring
    • Provides the real-time workload information.
Control Flow

- **Proactive Control**
  - Provides the number of instances to either add or remove.
    - $PW.T_{i+1}$ is the predicted workload for the next control interval.
    - The change in $VMs.T_{i+1}$ shows the number of instances to scale.
      - Negative to remove instances.
      - Positive to add instances.

```plaintext
Algorithm 1 ProRenaTa Control Flow

1: \textbf{procedure} PROACTIVECONTROL()  
   \hspace{1em} \triangleright \text{Program starts at time } T_i
2: \hspace{2em} PW.T_{i+1} \leftarrow \text{workloadPrediction(Trace)}
3: \hspace{2em} VMs.T_{i+1} \leftarrow \text{throughputModel}(PW.T_{i+1})
4: \hspace{2em} \Delta VMs.T_{i+1} \leftarrow VMs.T_{i+1} - VMs.T_i

6: \textbf{procedure} REACTIVECONTROL()  
   \hspace{1em} \triangleright \text{Program starts at time } T_i
7: \hspace{2em} \Delta W.T_i \leftarrow W.T_i - PW.T_i
8: \hspace{2em} \delta VMs.T_i \leftarrow \text{throughputModel}(\Delta W.T_i)

10: \textbf{procedure} PRORENAKASCHEDULER()  
    \hspace{1em} \triangleright \text{Program starts at } T_i
11: \hspace{2em} RS.T_i \leftarrow \text{dataMigrationModel}(T_i)
12: \hspace{2em} T.p \leftarrow \text{analyticalModel}(\Delta VMs.T_{i+1}, RS.T_i)
13: \hspace{2em} T.r \leftarrow \text{analyticalModel}(\delta VMs.T_i, RS.T_i)
14: \hspace{2em} \textbf{if} T.p + T.r > T_{i+1} - T_i \hspace{1em} \textbf{then}
15: \hspace{3em} VMsToChange \leftarrow \Delta VMs.T_{i+1} + \delta VMs.T_i
16: \hspace{3em} t \leftarrow \text{analyticalModel}(\text{VMsToChange}, RS.T_i)
17: \hspace{3em} TimeToAct \leftarrow T_{i+1} - t
18: \hspace{3em} \text{WaitUntil TimeToAct}
19: \hspace{3em} \text{ConductSystemResize(VMsToChange)}
20: \hspace{2em} \hspace{2em} \textbf{else}
21: \hspace{3em} \text{ConductSystemResize(}\delta VMs.T_i\text{)}
22: \hspace{3em} TimeToAct \leftarrow T_{i+1} - T.p
23: \hspace{3em} \text{WaitUntil TimeToAct}
24: \hspace{3em} \text{ConductSystemResize(}\Delta VMs.T_{i+1}\text{)}
```


Control Flow

• Reactive Control
  • Provides the number of instances to either add or remove.
    • The change in W.T shows the difference between the predicted workload and the observed workload.
    • This is used to determine if any scaling adjustments are required for the next cycle.

Algorithm 1 ProRenaTa Control Flow

1: procedure PROACTIVECONTROL()
   ▷ Program starts at time $T_i$
2:   $PW.T_{i+1} \leftarrow \text{workloadPrediction}(\text{Trace})$
3:   $VMs.T_{i+1} \leftarrow \text{throughputModel}(PW.T_{i+1})$
4:   $\Delta VMs.T_{i+1} \leftarrow VMs.T_{i+1} - VMs.T_i$
5:   $\Delta W.T_i \leftarrow W.T_i - PW.T_i$
6:   $\delta VMs.T_i \leftarrow \text{throughputModel}(\Delta W.T_i)$

6: procedure REACTIVECONTROL()
   ▷ Program starts at time $T_i$
7: $RS.T_i \leftarrow \text{dataMigrationModel}(T_i)$
8: $T.p \leftarrow \text{analyticalModel}(\Delta VMs.T_{i+1}, RS.T_i)$
9: $T.r \leftarrow \text{analyticalModel}(\delta VMs.T_i, RS.T_i)$
10: if $T.p + T.r > T_{i+1} - T_i$ then
11:   $VMsToChange \leftarrow \Delta VMs.T_{i+1} + \delta VMs.T_i$
12:   $t \leftarrow \text{analyticalModel}(VMsToChange, RS.T_i)$
13:   $TimeToAct \leftarrow T_{i+1} - t$
14:   WaitUntil TimeToAct
15:   ConductSystemResize(VMsToChange)
16: else
17:   ConductSystemResize($\delta VMs.T_i$)
18:   $TimeToAct \leftarrow T_{i+1} - T.p$
19:   WaitUntil TimeToAct
20:   ConductSystemResize($\Delta VMs.T_{i+1}$)
Control Flow

- **ProRenaTa Scheduler**
  - RS.T\(_i\) is the maximum data rebalance speed achievable at time T\(_i\).
  - T.p and T.r are the times to finish the data rebalance plans from the Proactive and Reactive controllers.
    - If there is not sufficient time for rebalance, then the difference between the two is computed and the resize is scheduled for the end of the PW.
    - Otherwise, the Reactive plan is immediately scheduled and the Predictive plan is scheduled for the end of the PW.
Prediction

• Short-term Prediction
  • Methodology is to make an estimate of the actual value using a Wiener filter.
  • This is a linear combination of past samples to estimate a forecast.
  • Uses a window of several hours.
  • Computes the Mean Squared Error (MSE) difference between the forecast and the reference samples.
  • If the decision value (ratio of MSE between long and short-term schedulers) is greater than a threshold, then the Wiener filter weights are recomputed.
    • Indicates the model is no longer accurate.

```
Prediction

• Long-term Prediction
  • Accounts for the sudden rise in workloads.
    • The rises depend on the past values by a number of past samples from the short-term predictor.
    • The rises in demand have an amplitude higher than the rest of the series, and take a random value with a variance that has empirically been found.
      • The periodicities are a set \( P \), where \( P_i \) indicates the \( i \)-th periodicity in the sampling frequency.
      • \( N_p \) is the total number of periodicities.
      • The structure of the filter [line 22] uses:
        • \( N_{FrHr} \) – Forecasting Horizon
        • \( L_j \) – Number of weighted samples of the \( i \)-th periodicity.
        • \( h_{i,j} \) – Weight of each sample
        • \( x[n] \) – \( n \)th sample of the time series
Controller Models

• There are two models that are constantly updated in every period
  • Throughput Performance Model
  • Data Migration Model

• Throughput Performance Model
  • Load-Balance (Figure at Right)
    • As the number of virtual tokens increases, the model become more evenly distributed. (StdDev decreases)
    • If the system evenly distributes the requests and a sample server did not violate SLA, then the system did not violate SLA.
Controller Models

• There are two models that are constantly updated in every period
  • Throughput Performance Model
  • Data Migration Model

• Throughput Performance Model
  • Workload is represented by read and write operations.
  • The throughput performance model (figure) shows that approaching the SLA border, but not crossing it, will keep the system within SLA and provide maximum utilization.
Controller Models

• There are two models that are constantly updated in every period
  • Throughput Performance Model
  • Data Migration Model

• Data Migration Model
  • Monitors the system load and outputs the maximum data transfer rate that can be used without violating SLA
  • ProRenaTa obtains the time to implement the scaling plan and can control the bandwidth to ensure SLA compliance during migration.
Evaluation

• Evaluation used Wikipedia access logs.
  • System implemented 5% of the most accessed pages for reading.
    • 5% of those were changed to writes.
  • 3000-7000 requests per second
  • 100,000 keys (Wikipedia Pages) in the data system.

• Evaluation was conducted to compare ProRenaTa with traditional approaches:
  • Reactive
  • Proactive
  • Ideal
Evaluation

• Criteria: SLA Commitment
  • Feedback was worst violator.
  • ProRenaTa and Predict similar.
    • Both near Ideal

• Criteria: CPU Utilization
  • Predict resulted in severe underutilization.
    • Servers removed only after they were no longer needed.
  • Feedback had both under and over-utilization.
  • ProRenaTa was near Ideal
Evaluation

• Criteria: VM Hours Used
  • Feedback allocated too few VMs
  • Predict allocated too many VMs
  • ProRenaTa was near Ideal

• Criteria: SLA Commitment
  • Feedback violated SLA on every workload increase.
  • Predictive and ProRenaTa only violated on sudden increases.
Evaluation

• **Criteria: Utility Function**
  • The utility function to objectively assess the three approaches measures penalty.
    \[ U = V_{\text{hours}} + \text{DurationOfSLA_Violations} \times \text{PenaltyFactor} \]
  • With no penalty factor, Feedback has the best utility.
    • As penalty for SLA violations increases, feedback quickly becomes the worst, leaving ProRenaTa close to Ideal as the overall winner.