Precisely – Enterworks Workload

AWS Ideas and Recommendations

April 26th 2024

aws

Components

1/ Review of current challenges

2/ Potential Paths to Consider (short/mid/long term)

3/ Options Value Matrix

4/ Next Steps & Open Questions

aws

Challenges

Performance at Scale	Application is experiencing performance issues for larger customers. Bigger infrastructure alone does not likely solve the issue.
Performance (general)	Metrics for both transactional and analytical operations fall below desired performance goals.
UI Performance	The main function dashboard presented by the system is often slow to respond or refresh for users, diminishing user experience.
Insufficient Resiliency	Application resiliency is impacted by existing design. When certain failures occur, a monolithic approach is required to bring the services back online, which results in increased customer downtime and impact.
Prohibitive Cost Profile	Costs of the product will not scale ideally with existing design and prevent delivery of Enterworks at acceptable gross margin.

aws

Summary

Core Recommendation:

Implement architectural separation between transactional and analytical workloads as the underlying principle on which to evolve the Enterworks stack. This approach can offer both short-term relief and creates optionality for implementing technology changes over time that could reduce risk and unlock significant performance and economic benefits.

Short Term	Medium Term	Long Term
Now; next 2-3 months	*4-6 month time frame*	*6+ months to evaluate & implement*
Goal: Relieve immediate customer-facing pain points with low-medium effort and low time-to-impact.	**Goal:** Incrementally improve price-performance, resiliency, and scalability.	**Goal:** Achieve major improvements in price-performance, resiliency, scalability, maintenance burden, and velocity of delivering features.
Minimize changes to technologies in the Enterworks stack	Exploring specific changes to technologies in the stack.	Exploring specific changes to technologies in the stack.

aws

Short Term Options

Initiatives:

Easiest:

- Create a read replica using 3rd party tool like Cloudbasix. Point all export work and the main customer dashboard to this replica.
- Create table partitions.
- Investigate instance type.

Medium Effort:

- Create separate tables for ~~categories~~ repositories.
- Shift from XML to JSON

Benefits:

- Should significantly reduce blocking in the database.
- Performance for both transactional and analytical operations will see improvement.
- Will incrementally improve ability to scale.

aws

Mid Term Options

Initiative:

Explore technology alternatives for each of the analytical and transactional workloads:

- Transactional: consider alternatives to MSSQL such as Aurora Postgres
- Analytical: workloads (eg. count(1)) move to an OLAP data warehouse (e.g. Redshift/Snowflake)
- Analytical: For imports, consider data pre-processing service (e.g. EMR/Glue)

Benefit:

- Incrementally improves performance and/or price-performance.
- Incrementally improves load times for import operations.
- Improvement in resiliency and failover capabilities.

aws

Long Term Options

Initiative:	Benefit:
• Explore DynamoDB for transactional workloads. • Explore DynamoDB as commercially supported alternative to BoltDB if security updates are a concern. • Explore Quicksight embedded as a low maintenance drop in for user dashboards.	• Significantly improves scalability. • Significantly improves price/performance. • Reduces administrative overhead. • Improves security posture.
• Move to microservices for remaining J2EE components.	• Improves delivery velocity. • Allows for less user impact and quicker return to service during failure events.

aws

Matrix – Areas Impacted per Option

	Performance	Scalability	Resiliency	Failover	Maintenance Improvement	Price/ Performance	Update Velocity	Upfront Costs	Effort
Read Replica	Med	Low	Low					Med	Low
Table Partitions	Low	Low							Low
Instance Type	Low	Low						Low	Low
Separate category tables	Low								Med
XML -> JSON	Med	Med				Low			Med
Transactions on Aurora	Med	Med	Med			Med			
Analytics on Redshift	Med	Med	Med			Med			
Imports use EMR	Med	High	Med						
Transactions on DynamoDB	High	V. High	High		Low	High		V High	V High
BoltDB to DynamoDB		High	High		Med				Med
Dashboards to Quicksight		High	High		Med		High		Med
J2EE to Microservices		High	High	High		High	High		V High

Outstanding Questions to Help Validate Short/Mid Term

Query Complexity:

- What is the typical complexity of the queries that will be executed (simple selects, complex joins, window functions, etc.)?
- Will queries involve processing large volumes of data?
- Are there any performance-critical queries that need to run extremely fast?
- Percentage of simple/moderate/complex queries in a given workload in a given timeframe

Outstanding Questions to Help Validate Short/Mid Term

Data Volume and Growth:

- What is the current data volume that needs to be processed?
- What is the projected data growth rate over time?
- Is the data mostly structured or are there semi-structured/unstructured components?

Concurrency and Workload:

- How many concurrent queries/users need to be supported?
- Are there periodic peak loads or is the workload more consistent?

aws

Proposed Next Steps

- Discuss short-term recommendations in-depth with Srikar to align on a plan
- POC for migrating the transactional workload to Aurora Postgres
- POC for migrating the analytical workload to Redshift and compare to Snowflake results
- Discussion with DynamoDB expert to introduce how Dynamo might apply and begin qualifying transactional workload feasibility

aws