Summary of Intent

Plainly stated, we have outgrown our existing data tier. We are non-scalable with respect to writes to the database, and business intelligence personnel must split their queries between the actual production monolith and Redshift, which is connected to our raw production data by the tenuous and overworked Matillion instance.

The goal of this proposal is to demonstrate that with reasonable capital expenditure the efficiency and horizontal/vertical scalability of our data layer can be accomplished.

Impact: Benefits and Opportunities

The overriding concern is that we are unable to scale at present. Our production data tier has exactly ***one*** entrypoint for data, the production master. This already has led to deadlocks, resource contention, and failed operations, not only for our backend processes, but for the customer-facing website. This must be solved.

The guiding principle in creating any serviceable data architecture is “divide and conquer.” Our current control and tuning variables are very coarse. By intelligently apportioning our resources within the data layer by task type and usage, not only will pressure be relieved on existing services, but we will gain the ability to tune in a very granular manner each segment to align with the class of functionality being required. Therefore, the objectives of this refactoring are thus:

- Segment the data layer by purpose;
- Tune each sub-layer based on usage;
- Take a giant leap towards moving from a data lake to a true data warehouse;
- Pre-summarizing our big data into the format needed for BI without belaboring our customer-facing databases as well as not asking more of our ETL processes than can reasonably be expected;
- Amplify security practices surrounding our data into a more protected-by-design design.

Conservatively stated, this new blueprint would increase performance by at least an order of magnitude.

Impact: Costs, Risks, and Constraints

The obvious pain point in opening this approach is the overlap cost. At this point, approval has not been granted for a proof of concept, which could be done with a minimum of a 3-node Aurora cluster and a single EC2 instance with EBS as the storage medium. During this POC timeframe, however, the expenditure will qualify as sunk cost. Appendix A lists the cost per month for a minimal POC environment; Appendix B is a rough estimate of yearly costs using various combinations of reserved and on-demand instances.

With respect to risks, as mentioned above, the greater risk to attempt to inveigle and cajole our current systems as we go forward. The truism of any data migration procedure is that the larger the datastore, the more difficult and time-consuming the maneuver becomes.

The cold fact is that we must do something to remain nimble and scalable. What is outlined in this document is a proven method to accomplish this with a minimum of expenditure and utilizing a stepwise process to ameliorate the impact to production systems.

Evaluation and Measurement

Given that the proof-of-concept environment is available, testing and evaluation would progress in the following manner:

- Capture a time period's worth of production queries while executing a JetProfiler analysis on the production master.
- Replay these queries on the POC environment segmented according to the categories outlined by purpose.
- Compare runtimes, bottlenecks, lock wait times, number of timeouts, number of resource contention events.

With even this minimal amount of analytic data, the relative benefit or non-benefit will be immediately apparent.

Out of Scope activities

- This proposal deals exclusively with the overall architecture; individual schemata, tables, columns, optimization, etc., are not covered within this scope.
- While connections from external applications will be created/tested, adding new applications to the BI or ETL list is not.
- Added security will be a value-added part of this project; however, this does not entail a full-scale security audit of users/permissions.

Project Overview

The current simplified view of our persistence layer can be visualized thus:

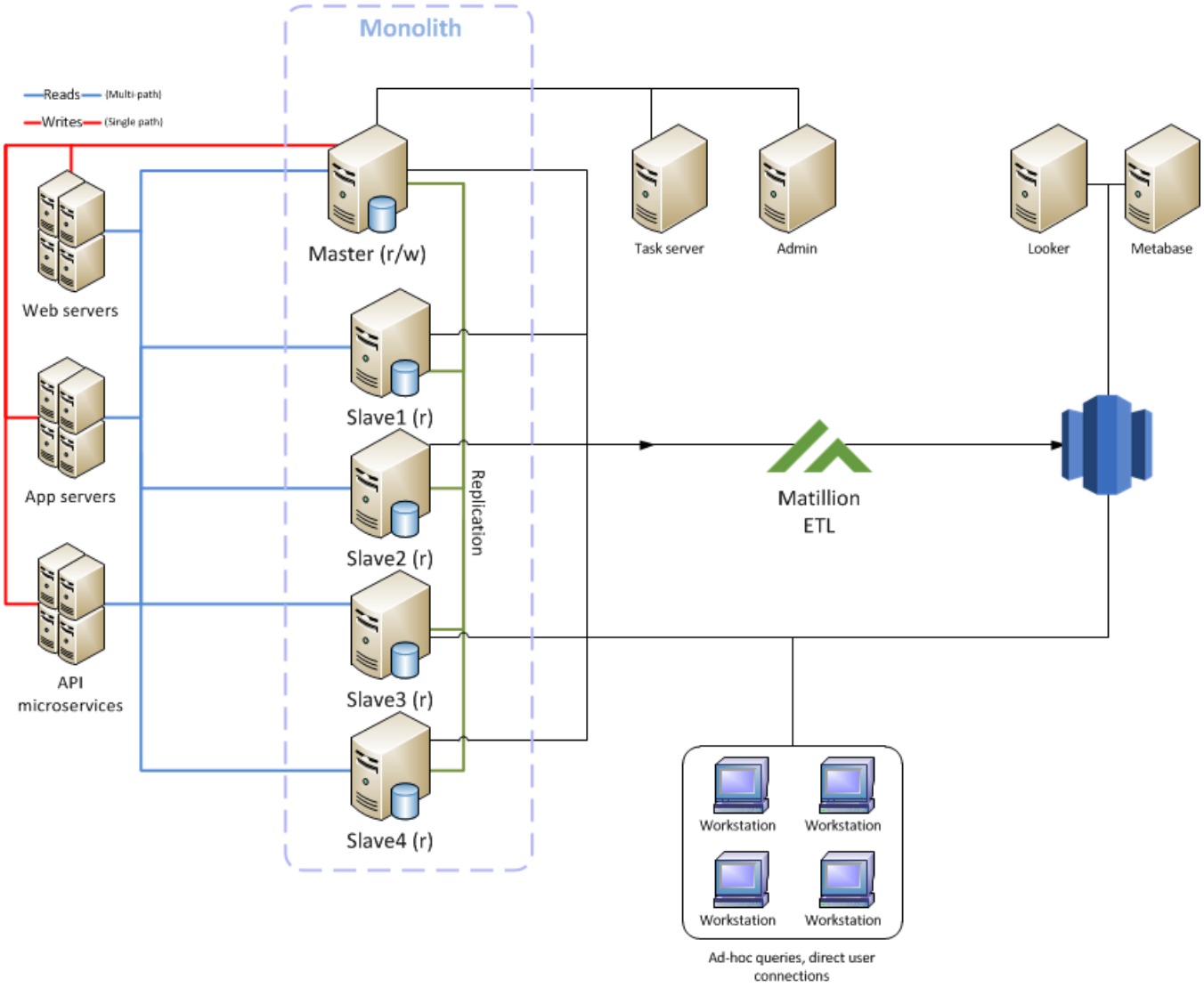

Figure 1. Current persistence layer architecture

If this seems a bit busy, that would be because it *is*. As things stand, the production monolith—our revenue-bearing database—supports all the following, *directly connected* to the same system which supports our business:

- Production websites
- API endpoints
- Microservice endpoints
- The Admin management tools
- Ad-hoc user queries
- Business intelligence queries
- Directly-connected ETL

While this structure served well early in the company's existence, the number of properties under management, datapoints collected for each property, customer base, and dozens of other discrete points are now stored in our database. We do not yet approach the industry buzzword of "big data," but we are close, and as the enterprise continues to expand, our necessary data, even for bare-bones business intelligence, will continue to grow.

It is therefore common sense to ensure that as much of our data is prepped for business consumption in a straightforward, modular, scalable, and tunable way. The current configuration simply isn't built in this fashion. However, with a rational, lucid, and stepwise progression, this can be accomplished.

Here is how the new architecture can be visualized:

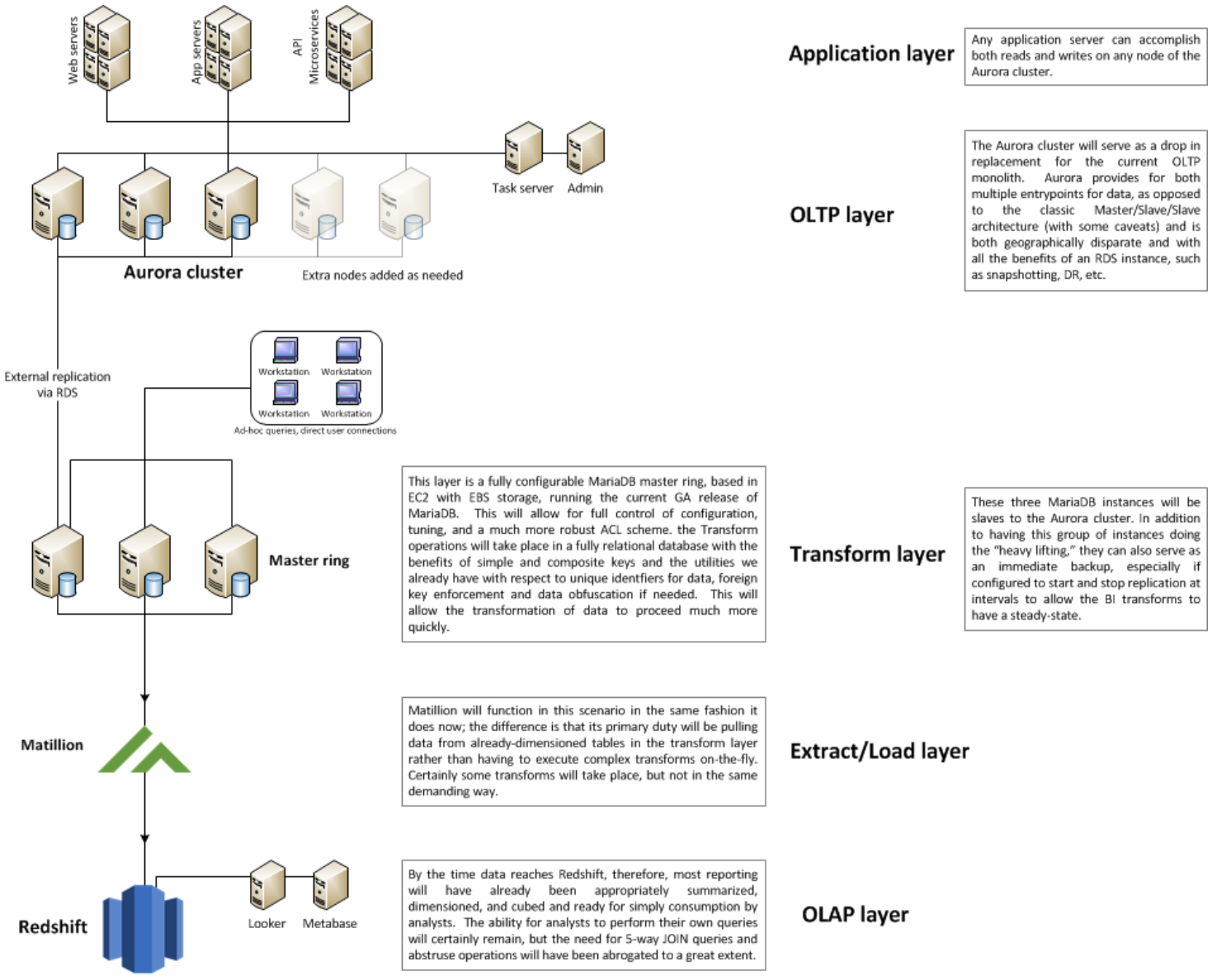

Figure 2. Projected persistence layer segmentation by purpose

Recommendations

My findings indicate that a POC is needed without delay to begin true evaluation of this blueprint. To begin, a three-node Aurora cluster and a suitably sized EC2 server with an EBS storage backend would be sufficient. This will allow initial load testing to execute as well as further demonstrating that offloading the highly-relational Matillion jobs which cause Redshift to churn for very long runtimes will execute much more efficiently in a relational database.

(Note: One particular BI job that had been taking between 15 and 20 minutes on Redshift was converted to a Jenkins periodic job on the production master earlier this week. On MariaDB, it takes between 6 and 8 seconds. All the Matillion job now has to do is pull the resultant summary table straight into Redshift, which occurs in 3 seconds or less.)

Once results from this first step have been appraised, the next step would be to add two more servers to the Transform layer, mimicing the configuration of the existing EC2/EBS node. At this point, we have the working bones of the new production system, and we can begin discussing timelines.

Conclusion

As stated above, the most hazardous path for us to take at this point in Vacasa's lifecycle is to do nothing. The proposal outlined above, based on my 30+ years in information technology, with 25 of them as a production database architect and administrator, merely delineates the standard business strategy in tackling scaling concerns, molded to fit our particular technology stack.

Appendix A: POC environment cost

EC2 Transform instance	
db.r4.2xlarge	$389.43
3-node Aurora Cluster	
db.r4.2xlarge	$899.13
Total:	$1,288.56

Appendix B: Estimated Production CapEx

Aurora-based monolith with EC2 Transform layer
Replaces current production master and five slaves
Possible to collapse rates, finance, and other ancillary instances into the cluster

Instance Type	Aurora r4.8xlarge for monolith; r4.4xlarge for Transform layer			
	Aurora	**EC2**		
Nodes	4	3		
Projected totals			**Upfront**	**Annualized monthly**
1yr all upfront RI monthly	$800.00	$2,700.00	$100,684.00	$11,890.33
1yr partial upfront RI monthly	$5,004.80	$3,212.46	$46,141.00	$12,062.34
1yr no upfront monthly	$9,618.44	$4,386.54	$0.00	$14,004.98
3yr all upfront RI monthly	$800.00	$2,700.00	$193,301.00	$8,869.47
3yr partial upfront RI monthly	$3,603.20	$3,043.83	$92,363.00	$9,212.67
1yr all on demand monthly	$9,618.44	$4,386.54	$0.00	$14,004.98

Aurora 4-node Cluster: db.r4.8xlarge

1 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
All Upfront	$88,648.00	$800.00	$98,248.00	$98,248.00
Partial Upfront	$40,000.00	$5,004.80	$100,057.60	$100,057.60
No Upfront	$0.00	$9,618.44	$0.00	$0.00

3 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
All Upfront	$170,060.00	$800.00	$66,286.67	$170,060.00
Partial Upfront	$80,000.00	$3,603.20	$69,905.07	$209,715.20
No Upfront		Not available		

EC2-based Transform layer: 3 nodes

1 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
RI All Upfront	$12,036.00	$2,700.00	$44,436.00	$44,436.00
RI Partial Upfront	$6,141.00	$3,212.46	$44,690.52	$44,690.52
On-demand	$0.00	$4,386.54	$52,638.48	$52,638.48

3 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
RI All Upfront	$23,241.00	$2,700.00	$7,747.00	$23,241.00
RI Partial Upfront	$12,363.00	$3,043.83	$40,646.96	$121,940.88
On-demand	$0.00	4386.54	$52,638.48	$157,915.44

Totals

1 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
All Upfront	$100,684.00	$3,500.00	$142,684.00	$142,684.00
Partial Upfront	$46,141.00	$8,217.26	$144,748.12	$144,748.12
No Upfront	$0.00	$14,004.98	$52,638.48	$52,638.48

3 Year

	Upfront Cost	Monthly Cost	Annual Cost	RI Life Cost
All Upfront	$193,301.00	$3,500.00	$74,033.67	$193,301.00
Partial Upfront	$92,363.00	$6,647.03	$110,552.03	$331,656.08