Assessing Data Architectures – Assignment 1.1

Filipp Krasovsky

University of San Diego

ADS 507 – Practical Data Engineering

January 17, 2022

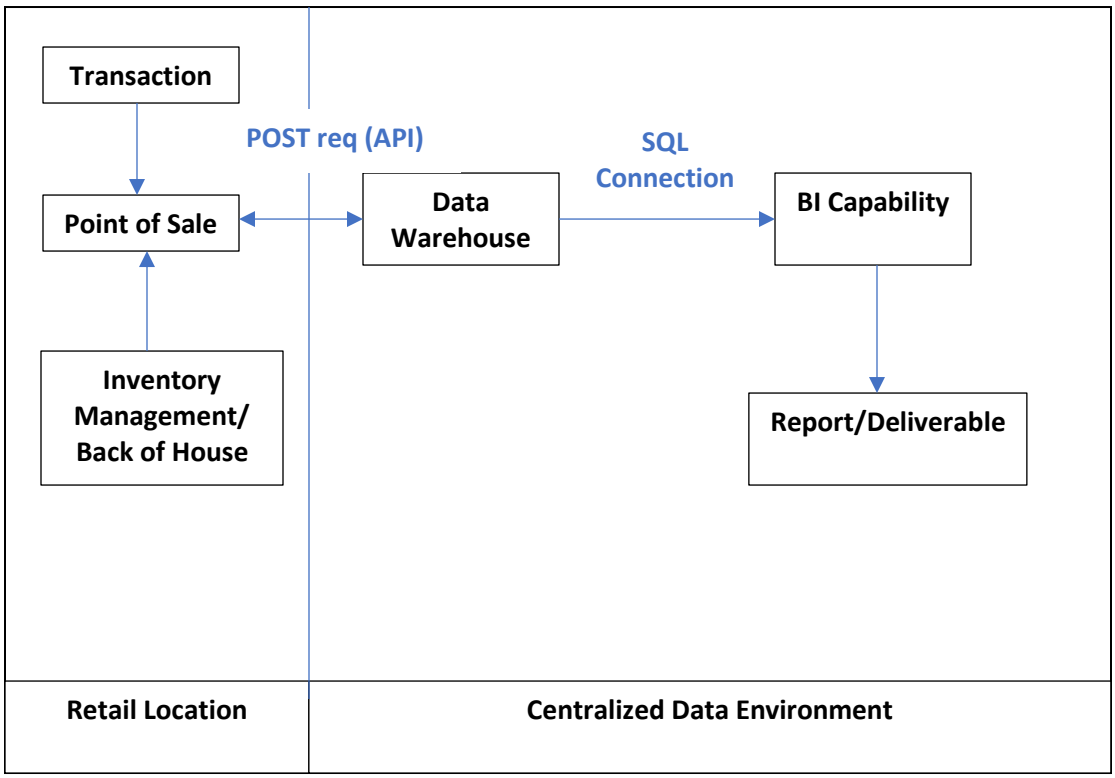
Case Study 1: Retail Chain

This company is a national retailer. They load data from their stores into a centralized data warehouse and run their business intelligence applications on top of the warehouse. The issue that they are facing is that bad data from some of their stores is entering the warehouse regularly and causing incorrect reports to be generated by the business intelligence applications. How can you ensure that reports received by business users are accurate?

We can begin by identifying that this is an issue of **reliability** – that is, the system isn't failing when incorrect reports are generated, but a component in the system is deviating from the norm. The result of component deviation is that bad data is entering the warehouse, but the underlying cause can originate from several locations. To explore these, we make some assumptions about the architecture:

1. The retail locations all use some sort of POS system (Point of Sale) to record and conduct transactions. Some examples included Shopify, Square, and Toast. It is also likely that this POS software is cloud-based and stores data in a remote server similarly to Square. (Square, 2017)
    a. A corollary of this is that the POS hardware itself is the middleman layer between the transaction itself and the centralized data warehouse run by the company.
2. We can also assume that data from our warehouse can be accessed through some variant of SQL (TSQL, Postgres, MySQL, etc.) and passed on to a Business Intelligence platform such as Power BI.
3. The type of data collected by the POS includes, but is not limited to, customer details, payment methods, employee productivity, tax details, and inventory details. (Retailminded, 2017)

A visual representation of this architecture is also feasible and helps with our analysis:



Therefore, it's likely that bad data entering the warehouse environment is the result of one of several possibilities and can be handled as follows:

1. The API call between the POS and the DW is faulty, duplicates data, overwrites it due to deprecated business logic, or is otherwise interrupted during execution.
    a. Any server-side stored procedures that validate incoming data (if they exist) might be causing the issue.
    b. **Recommendation:** Rigorous QA testing to evaluate whether the data retail management believes is being sent from their location corresponds to the packet received, and then subsequent validation that the packet received maintains the same state after every middle layer before insertion into the SQL database.

2. As many POS systems synchronize with the back of house inventory tracking system, inventory shrinkage might be contributing to a disparity in real and reported inventory numbers.
    a. **Recommendation:** conduct onsite validation to ensure inventory data lines up across systems – if not, consider turning inventory updates into a nightly scheduled task with validation procedures to ensure that inventory numbers line up, or otherwise include variance in the report.
3. The Point of Sale software itself might be a dependency issue that suffers from software problems, API calls failing to execute, etc.
    a. **Recommendation:** This is a case where fault tolerance might be worth inspecting – data might be misreported because some processes at the point of sale are failing to execute. Either manual validation of transactions or implementing a tool like Chaos Monkey to inspect resilience may provide insights.

Case Study 2: Mobile App Startup

This company is an early startup, and you were hired to start their data team. The company's DevOps engineer gave you credentials for their production database on your first day, and you started to build metrics dashboards that pull data directly from the database. It has been a few weeks now, more users are downloading the app, and that same DevOps engineer lets you know that your user's long-running queries are slowing response times. How can you ensure that your work does not affect end users of the app?

This challenge is more direct – this is an issue of **scalability.** That is, some component of the system (in this case, a query used to create a metric dashboard) is slowing down performance as the number of inputs increases. In this instance, there are two solutions:

1. Refactoring Code – if we assume the queries can be viewed inside a development environment such as SQL Server Management Studio (SSMS), extensions such as a TSQL Linter can be used to inspect the code for violation of best practices, inefficient joins, or operations that could be replaced with more efficient variants (ie a JOIN or UPDATE instead of a cursor). This should provide some meaningful increase in speed. Furthermore, a dirty read that queries tables with NOLOCK included in the syntax can ensure that other users can still access the database while the dashboard is being generated.
2. Provided that the cadence of the metric dashboard generation is slow enough – for example, the deliverable is required on a monthly or quarterly basis, and if the database itself has backups that are updated at the same or at a higher frequency, the metrics dashboard can operate by creating a connection to the backup or historical warehouse instead of the working database to ensure that other users aren't affected as much.

Case Study 3: Steady SaaS Company

This company is a SaaS (software as a service) company. They have been steadily growing over the years. Throughout this time they have used a data warehouse to generate reports, build machine learning models, and improve their internal processes. Recently they have noticed that they have scaled up their data warehouse to handle continued data growth, but have not had a similar increase in their compute needs. Because compute and storage are purchased together for their current data warehouse, the warehouse has a lot of processing power that sits idle most days. Is there anything you can do to reduce the costs of this infrastructure?

If we assume that the vendor that provides the company with their technical solutions (ie computational capacity and database architecture) is not open to creating an exception in the contract, we could still advocate that the company ought to make a change to its architecture. For instance, as the company continues to grow, they could create an in-house solution that allows them to selectively active and deactivate computational resources, like the number of CPU cores being used, based on some pattern that can be derived from changes in load. For example, if most processing power is idle except for Friday, when most clients begin creating large volumes of reports, a **Distributed Network** (Hazelcast, 2017) can handle requests by linking nodes to a task server. This task server can concurrently change the number of nodes that are turned on in response to the load it receives, saving the company in operational costs.

Case Study 4: Big Data Bank

This company is a bank that were early adopters in the big data movement. They have a centralized data lake and business units manage their own Hadoop clusters to process data from the lake. Several business units have recently had their Hadoop experts leave the company. They are now having difficulties keeping clusters healthy to manage their current workload and they do not know whether they can schedule additional jobs. How can you solve this growing knowledge gap?

This is an issue of **Operability** and **Scalability** – that is, because Hadoop experts are leaving the company, and interest in Hadoop is also empirically waning (Hemsoth, 2021), we can consider Hadoop to be a legacy system in this business challenge as a key assumption. Furthermore, the business challenge implies that the company's scale is increasing over time while the Hadoop environment has remained static because of a knowledge gap in the company that precludes Hadoop resources from adjusting the new load. We also assume that the company cannot switch to some other data processing framework from Hadoop in the short run.

There are several solutions at our disposal:

1. Assuming that only *some* business units are suffering from this knowledge gap and not others, knowledge sharing across departments is a feasible short-term solution, as well as having Hadoop experts in non-affected departments commit more of their bandwidth towards managing failing Hadoop clusters. Finally, if feasible, departments can merge Hadoop clusters with other teams with clusters healthy enough to take on additional load in the form of tasks.
2. Hire a team of software consultants to diagnose what steps can be taken to support the health of the Hadoop environment over a long enough horizon for the company to implement a data processing solution that handles the increasing load moving forward.

   a. This solution allows the Hadoop environment to remain in place and preserve its functionality while delegating any load beyond a certain threshold to a different processing service such as Spark or Hive.
   b. We also assume this new framework will scale over time, and that there are subject matter experts within the company who can drive implementation, while Hadoop slowly deprecates.

3. Concurrently with Solution 1, we can also artificially restrict the cluster bandwidth so that tasks/jobs are processed through a queue that takes the cluster health into consideration as to avoid major system failures that cascade through the rest of the production environment.
4. Given that Hadoop is relatively well documented, we can expend resources on a training program for employees and dedicate a small team specifically towards maintaining Hadoop while the knowledge gap fills over time. This solution assumes that the exodus of Hadoop experts is idiosyncratic in nature and not part of a market trend, meaning that it will eventually equilibrate and the team composition will include enough Hadoop users as the recruitment cycle continues. Concurrently, we may also entertain a proposal to change the company recruiting practices to give higher priority to individuals with expertise in Hadoop.

References

11 Reasons You Need a Web Based POS. Square. (2017, Jan. 28th). Retrieved January 17, 2022, from https://squareup.com/us/en/townsquare/cloud-pos

What data is captured by POS software? can it be used to improve business? Retail Minded. (2017, May 15). Retrieved January 17, 2022, from https://retailminded.com/what-data-are-captured-by-pos-software-can-they-be-used-to-improve-business/

What is Distributed Computing? Hazelcast. Retrieved January 17, 2022 from https://hazelcast.com/glossary/distributed-computing/

Why The Fortune 500 Is (Just) Finally Dumping Hadoop. Hemsoth, Nicole. Nextplatform.Com. (2021, Feb. 17) Retrieved January 17, 2022 from https://www.nextplatform.com/2021/02/17/why-the-fortune-500-is-finally-dumping-hadoop/