Real -Time Order Promising and the Predictive Supply Chain

Manufacturers have been implementing supply chain optimization systems since the early 1990's. However these systems often have challenges dealing with the inevitable uncertainties of the real world. Here we describe a new approach that uses a new kind of data platform we call Online Predictive Processing (OLPP). We start out by highlighting the top three problems plaguing manufacturers. Then we present the shortcomings of the current solutions to these issues. We then describe a new approach to one of these problems that avoids the pitfalls of current solutions, and provide a technical explanation why this approach is superior. Finally, we provide a glimpse of the solutions to the other two problems introduced which will be covered in future articles.

One customer adopting this new solution is Infinera, a provider of Intelligent Transport Networks that enable carriers, cloud operators, governments and enterprises to scale network bandwidth. Infinera is deploying a cloud-based order promising and scheduling service that enables automated lookups of order promising and scheduling dates across multi-line order quotes. The solution built with Intrigo Systems, a leading supply chain consultancy, includes a web-based user interface for performing the order promising and scheduling queries using available-to-promise (ATP) logic, as well as a scalable backend infrastructure to synchronize inventory reservations with ERP order data.

"This order promising and scheduling solution gives our salespeople a real -time reservation tool to better serve our customers and make reliable commitments on behalf of our enterprise," said Todd Toumala, CIO, Infinera. Additionally, we will get an early warning system of orders threatened by unexpected shortages and tools to re-optimize."

What are three of the biggest pain points plaguing sales and operations today?

1. Sales people cannot in real time give accurate order promise dates to customers;
2. Planners cannot optimally allocate inventory to orders in order to prioritize shipments. Planners want to apply different business rules in the beginning of the quarter, like prioritizing on-time delivery, versus the end of the quarter, where orders with more revenue are more important;

3. Sales and operations people would love to have a crystal ball that could predict what will go wrong in the supply chain. Now, they can use artificial intelligence to see around corners by developing machine learning models that predict changes to the plan and then use those predictions in order to proactively promise orders and schedule shipments against the predictions.

Why are today's supply chain optimization systems not the answer?

Instead of addressing these major pain points, today's supply chain optimization systems focus instead on forecasting and demand planning, supply planning, capacity planning, and sales and operations planning. They are not architected to effectively address the pain points listed above.

These systems try to boil the ocean and therefore have seven-digit total cost of ownerships, huge implementation timelines, and serious scalability constraints.

Reason 1: High cost
Supply chain optimization systems are expensive because they rely on proprietary in-memory compute engines to perform tasks that traditional ERP systems, powered by traditional databases, could not handle. These scale-up engines typically store all the data in memory and max out server memory capacity, making the solutions prohibitively expensive.

Reason 2: Implementation complexity
These systems are typically implemented as large "big bang" projects that try to transform all of sales and operations planning at once. It takes time to implement many new planning systems and integrate them into enterprise transactional systems. Change management also takes time to design new business processes and to train people to perform those processes.

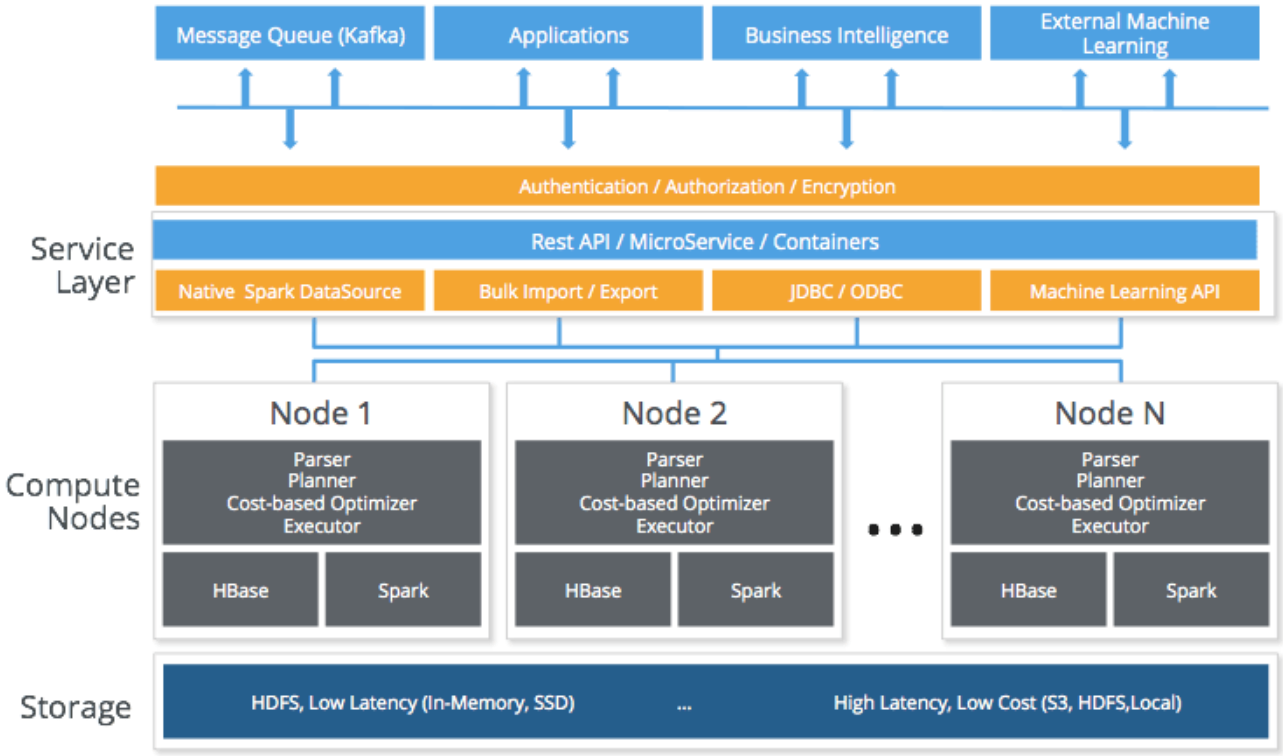
Reason 3: Lack of scalability
Supply chain systems are typically implemented on large, centralized servers with maximum limits on memory. Once you hit this wall, there is no way to solve the problems.

But there is a better way:

1. Use a scale-out architecture versus scale-up memory intensive systems to deal with the granular data requirements for fine grained scheduling
2. Use a traditional SQL system so that you don't need to be trained on proprietary scale-out technology
3. Implement point solutions in 30 day iterations in the cloud

What is a scale-out architecture?

Scaling out using a cluster of commodity machines is better for supply chain workloads than scaling up by adding more resources, such as memory or CPUs to a single server for both performance and scalability reasons. Scale-out is more performant because many servers can operate in parallel on a computation versus one. It is more scalable because as the system starts to reach capacity, you can add more servers to elastically extend capacity without shutting down. Popular scale-out architectures such as Hadoop, HBase and Spark were designed for this purpose and form the basis of Splice Machine's OLPP architecture.



Why SQL in the cloud?

Now the implementation of scale-out architectures are easier than ever. It used to be that IT professionals would have to learn new distributed scale-out systems like Hadoop and HBase and use low-level APIs for key-value stores that did not support indexes and joins. But now they can use the full power of SQL which is the data query language they have been using for years. In addition, implementing scale out systems used to be complex with significant devops skills. Now you can use cloud services and never have to procure, configure, or operate the platform. You just connect to the data platform and use it. Even better, provisioning a cloud

OLPP service is easier than ever. You can use your cloud budget and billing method. For example on AWS, you can go to their AWS Marketplace and literally provision an OLPP platform using your AWS account.

A three-step implementation plan

We propose a three step deployment plan corresponding to the supply-chain problems above:

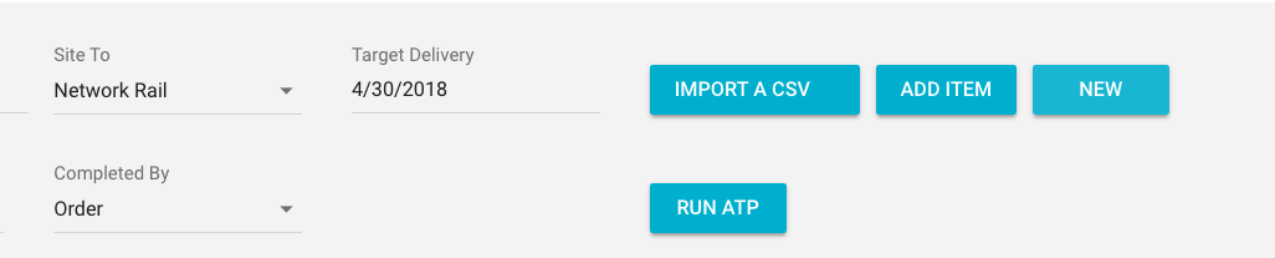
1. Implement order promising to plan
2. Implement order scheduling and optimization
3. Develop machine learning models to better predict inventory availability for order promising and order scheduling

Step One: Order Promising

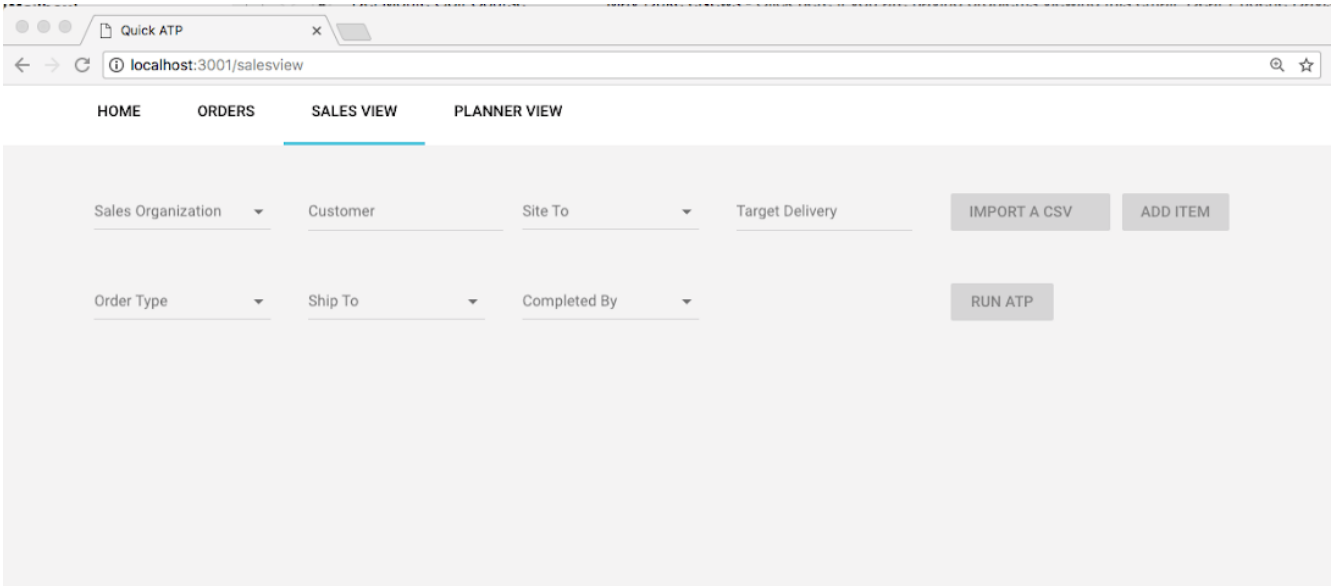
If you're a large manufacturer of network equipment, and you have sales people out trying to sell large systems, invariably what happens is the salespeople are asked, 'Can you get the order to me by this date?' And in most companies – even today with the best ERP system out there – salespeople are relegated to saying, 'I will go check and get back to you.' That's an opportunity for the customer to go somewhere else and get a competing quote on the order — after all, they have to wait, anyway.

This order promising problem involves rapid order entry with a request date (or more than one request date for more complex orders), as well as shipping rules for how to group the items. Then, the system calculates whole order available-to-promise (ATP) as well as ATP by line item. This gives the sales person instant insight and negotiating room to commit a large portion of the order to meet the customer's request date(s). Here's a screen walk through demo of the order promising system delivered to Infinera.

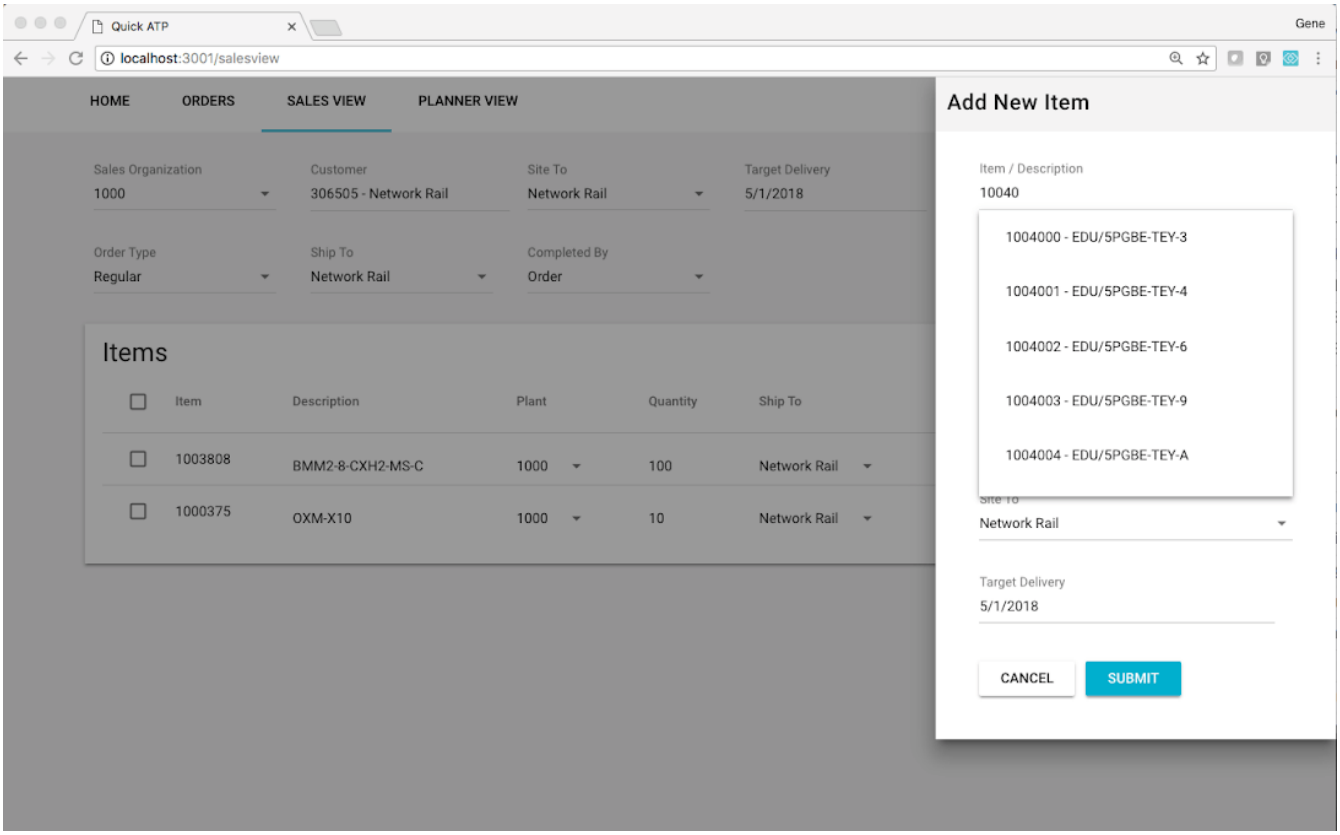
ER VIEW



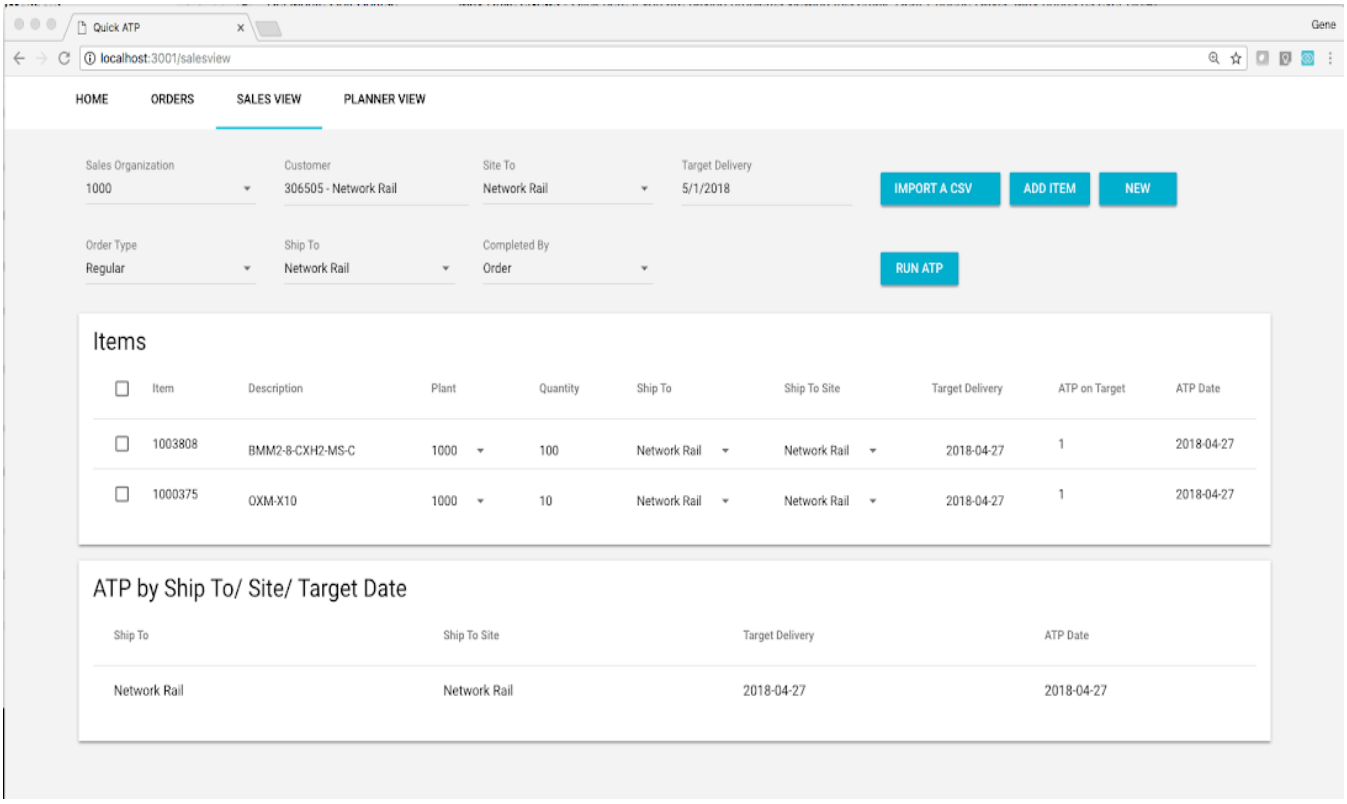
Here is the main screen where you can choose to make a new order in this display or import an entire order via a CSV file.



Now, we need to specify the customer, a target delivery date, and the site the customer is designating for shipment.



The new line item button allows a fast search with auto-completion for the item and its description. Line items can also have custom target delivery dates.



Then you hit the ATP button and the system returns promise dates in seconds for the whole order as well as each line item.

The enabling data schema: Timelines

Timelines are a relational representation of temporal data for AI applications that record historical, present and future values.

A timeline table contains a collection of timelines, each with a unique id. In this case, each id represents an inventory item at a location. Here is one simplified implementation:

```
create table SUPPLYCHAIN.TIMELINE_INT(Timeline_Id BIGINT, ST
TIMESTAMP, ET TIMESTAMP, VAL BIGINT, primary key (Timeline_Id, ET)
);
```

Tracking Inventory As Timelines

FINISHED

```
%splicemachine
select * from supplychain.timeline_int
where TIMELINE_ID = ${inv=200}
order by SUPPLYCHAIN.TIMELINE_INT.ST;
```

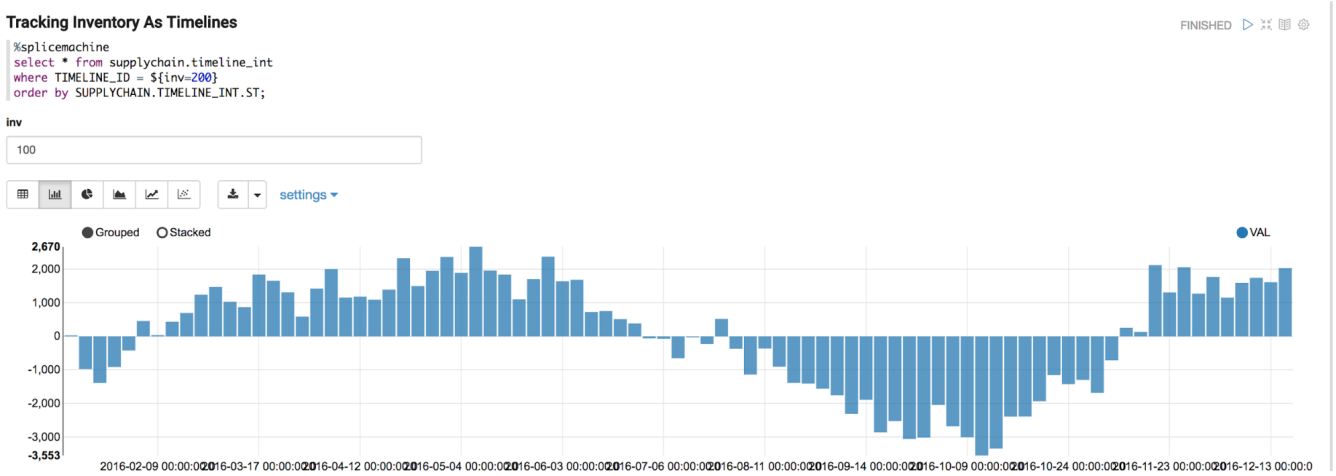
inv

100

TIMELINE_ID	ST	ET	VAL
100	2016-08-15 00:00:00.0	2016-08-20 00:00:00.0	-905
100	2016-08-20 00:00:00.0	2016-08-25 00:00:00.0	-1388
100	2016-08-25 00:00:00.0	2016-08-30 00:00:00.0	-1409
100	2016-08-30 00:00:00.0	2016-09-04 00:00:00.0	-1560
100	2016-09-04 00:00:00.0	2016-09-09 00:00:00.0	-1755
100	2016-09-09 00:00:00.0	2016-09-14 00:00:00.0	-2307
100	2016-09-14 00:00:00.0	2016-09-19 00:00:00.0	-1891
100	2016-09-19 00:00:00.0	2016-09-21 00:00:00.0	-2863
100	2016-09-21 00:00:00.0	2016-09-24 00:00:00.0	-2526

Took 0 sec. Last updated by splice at April 18 2018, 10:55:58 AM. (outdated)

Every row represents: `TIMELINE_ID = VAL @ [ST ET]` meaning the variable denoted by the id has the value over that time interval. The values in this example represent the amount of that item available at that location over a period of time. This is shown above in a Zeppelin Notebook paragraph. Zeppelin comes pre-integrated with the Splice Machine Cloud Service. Zeppelin also allows lightweight data visualizations such as histograms.



Most ERP systems do not materialize – or in other words explicitly calculate and store – the net of demand and supply over every time interval for every part. That would be too computationally expensive for them. With scale-out, this is not a problem, and as you will see below, it enables fast ATP queries.

Under the hood: An Available-to-Promise (ATP) Query

Timelines require indexed row-based storage and an OLTP compute engine to quickly look up values associated at times. Here is a typical ATP query for a simplistic supply chain model:

Here is the explain plan for this query:

```
Plan

Cursor(n=6,rows=1,updateMode=,engine=control)

 ->  ScrollInsensitive(n=5,totalCost=8.437,outputRows=1,outputHeapSize=0 B,partitions
=1)

   ->  ProjectRestrict(n=4,totalCost=4.034,outputRows=17,outputHeapSize=0 B,partition
s=1)

     ->  GroupBy(n=3,totalCost=4.034,outputRows=17,outputHeapSize=66 B,partitions=1)

       ->  ProjectRestrict(n=2,totalCost=4.034,outputRows=17,outputHeapSize=66 B,part
itions=1)

         ->  TableScan[TIMELINE_INT(5376)](n=1,totalCost=4.034,scannedRows=17,outputR
ows=17,outputHeapSize=66 B,partitions=1,preds=[(ST[0:2] >= 2017-01-01 00:00:00.0),(TI
MELINE_ID[0:1] = 100),(ET[0:3] < 2017-05-05 00:00:00.0)])
```

The reason why this query is fast is that in Splice Machine, short range scans are performed on the Apache HBase engine. In the table's definition above, we see

the primary key is the combination of the timeline id and the end time of an interval. HBase is blazingly fast to look up records based on the primary keys it uses to sort the records, even at Petabyte scale. In the ATP query plan above, you can see the predicates in the Table Scan that are pushed down to Apache Hbase restrict both the ID and and ET fields, so this range scan will be very fast.

The application stack we used for this application we playfully call BREWS:
(B) Babel - Javascript/Typescript compiler
(R) React - UI Framework
(E) Express - Web application framework for Node.js designed for building web applications and RESTful APIs
(W) Webpack - module bundler.
(S) Splice Machine - OLPP RDBMS.

Step Two: Order Scheduling

In a follow-on article, we will describe the integration to SAP and will show how planners can use this system to:

- Remove orders from the plan
- Pick business priority rules
- Place rules on the plan
- Reconcile reservations and actual orders in the SAP ERP system.

Step Three: The Supply Chain Crystal Ball: Learning to predict supply chain events

We will also publish an article on how we use machine learning to predict what orders are going to be late and how we incorporate these predictions into the supply chain plans. See the previous blog on the native Spark DataSource to get a sneak-peak on how we do in-database machine learning.

Summary

Here we described a new approach supply-chain optimization powered by a new scale-out Online Predictive Processing (OLPP) engine. We demonstrated a fast ATP application and explained how we represent temporal data such as inventory

availability. Coming soon, we will show how planners optimize inventory and learn to predict supply chain events.