



Operationalizing Machine Learning Models

Tom Stern



The next section of the exam guide is about Analyzing and Modeling.

Analyzing is looking for patterns and gaining insight from data.

Modeling is about identifying patterns that can be used for categorizing or recognizing new data or predicting values or states in advance.



Analyzing Data and Enabling Machine Learning

Tom Stern



In the first section, we will look at analyzing data and enabling machine learning. The way these two are related is that a lot of businesses begin by analyzing data.

After they get value from the analysis, they start to look deeper and often realize that they have unstructured data that could provide business insights.

And that leads them to want to enable machine learning. So this is a natural adoption path.

Pretrained models are a fast way to magical experiences

Pre-trained ML models

Ready to go

(accessed through REST APIs;
no machine learning knowledge is required.)



Cloud Natural
Language
API

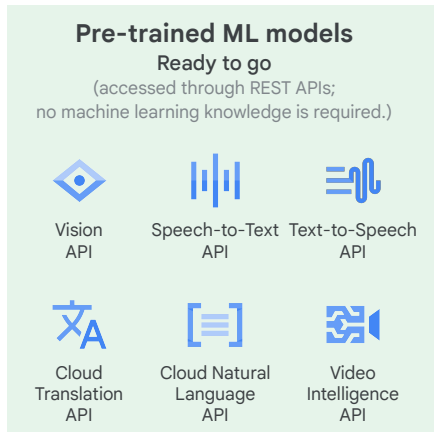
Make sure you are familiar with each pre-trained ML model. For example, what are the three modes of the Natural Language API?

The answer is Sentiment analysis, Entities, and Syntax.

Would Natural Language API be an appropriate tool for identifying all of the locations mentioned in a document?

Yes. It might be useful for that purpose.

Pretrained models are a fast way to magical experiences



Pre-trained models can turn apparently meaningless data into meaningful data. I think the translation API is a good example.

If you have text in a language you don't understand, the meaning contained in the data is not available to you.

Use the translation API to convert it to a language you DO understand, and suddenly there is meaning and value in it.

Pre-trained models create value from spoken word, from text, and from images common sources of unstructured data.

Pretrained models are a fast way to magical experiences

Pre-trained ML models

Ready to go

(accessed through REST APIs;
no machine learning knowledge is required.)



Vision
API



Speech-to-Text
API



Text-to-Speech
API



Cloud
Translation
API



Cloud Natural
Language
API



Video
Intelligence
API

Vertex AI (Previously AI Platform)



AutoML

Bring Your Own Data

Image
Video
Text
Translation
Tabular

ML frameworks

For Advanced Use Cases

TensorFlow
PyTorch
scikit-learn
and more

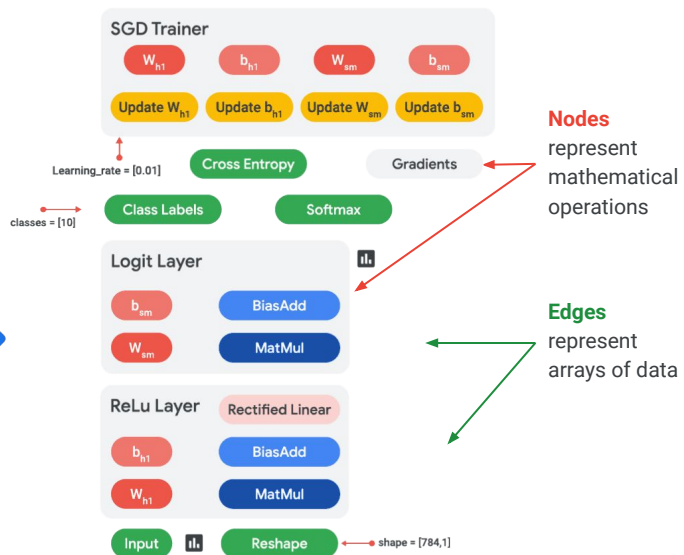
If none of the pre-trained models will work for you, you can use TensorFlow and AI Platform to create your own models.

TensorFlow

TensorFlow is an open-source high-performance library for numerical computation that uses directed graphs.

TIP

Directed Graphs are mentioned in multiple technologies in Google Cloud.



TensorFlow is an open-source high-performance library for numerical computation. It is not just for machine learning. It can work with any numeric computation. In fact, people have used TensorFlow for all kinds of GPU computing; for example, you can use TensorFlow to solve partial differential equations -- these are useful in domains like fluid dynamics. TensorFlow as a numeric programming library is appealing because you can write your computation code in a high-level language -- like Python -- and have it be executed in a fast way.

The way TensorFlow works is that you create a directed graph (a DG) to represent your computation. For example, the nodes could represent mathematical operations -- things like adding, subtracting, and multiplying. and also more complex functions. Neural Network training and evaluation can be represented as data flow graphs. The tensor data representation is passed from node to node where it is processed. It is analogous to Dataflow and a pipeline, but the input and output is mathematical operations. TensorFlow was developed at Google and is portable across GPUs, CPUs, and special hardware called TPUs, which are TensorFlow Processing Units.



Deploying an ML Pipeline

Tom Stern



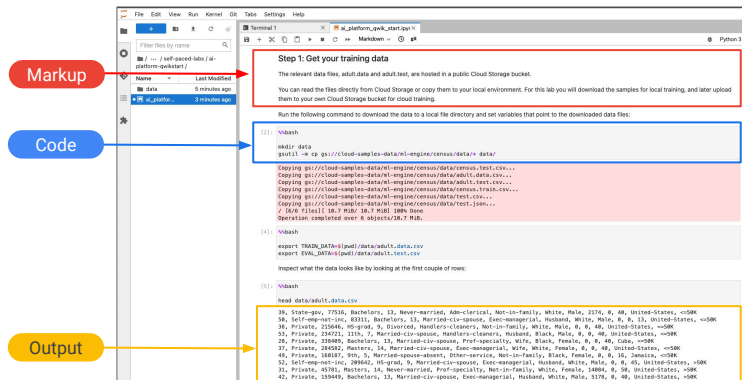
If you are a Data Analyst, you might use SQL to analyze data. This is a special area of strength for BigQuery.

As a Data Engineer you are probably more interested in setting up the framework for data analysis that would be use by a Data Analyst than actually deriving insights from the data.

Does that make sense?

Analyzing Data in a Data Engineer context is about the systems you might put into place to make analysis possible for your users or clients.

Increasingly, data analysis and ML are carried out in self-descriptive, shareable, executable notebooks



A typical notebook contains code, charts, and explanations

TIP Analysis and development can be done in notebooks.

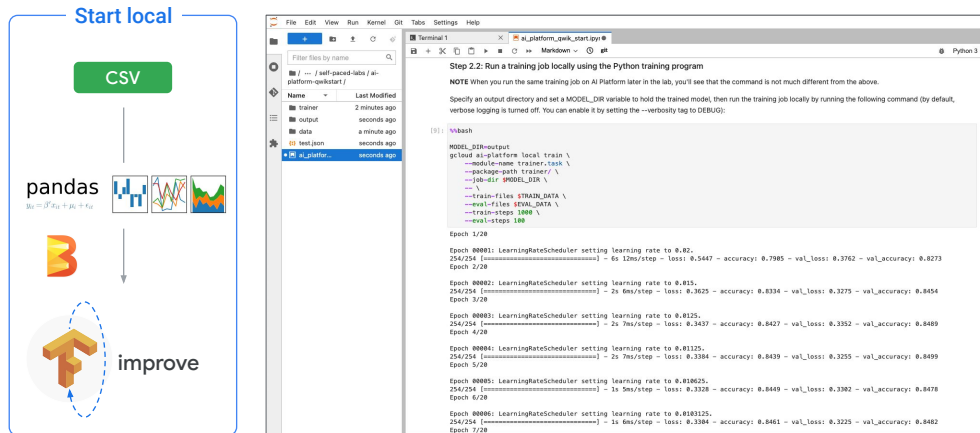
If you aren't running queries, then most likely you are running programs to analyze the data.

This is where notebooks shine.

Notebooks are a self-contained development environment and are often used in modern data processing and machine learning development because it combines code management, source code, control, visualization, and step-by-step execution for gradual development and debugging. A notebook is a great framework for experimenting in a programming environment.

There are a number of popular notebook frameworks in use today including Colab and Datalab.

In a notebook, start locally on sampled dataset



Let's talk about analyzing data when the data is unstructured or not organized in a way that is suitable to your purpose.

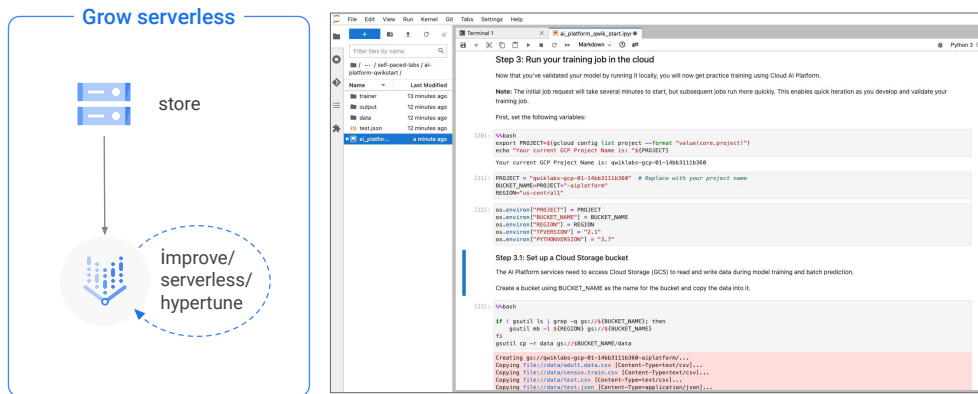
If you can use a pre-trained ML model, it can quickly transform that data into something useful.

But if you don't have an appropriate model, you might need to develop one.

One of the basic concepts of Machine Learning is correctable error. If you can make a guess about something, like a value or a state, and if you know whether that guess was right or not, and especially if you know how far off the guess was, you can correct it. Repeat that hundreds and thousands of times and it becomes possible to improve the guessing algorithm until the error is acceptable for your application. Concepts like "fast failure", lifecycle, and iterations become important in developing and refining a model.

In this example, development of the model is started in a notebook on a simple subset of data.

Then, scale it out to Google Cloud using serverless technology



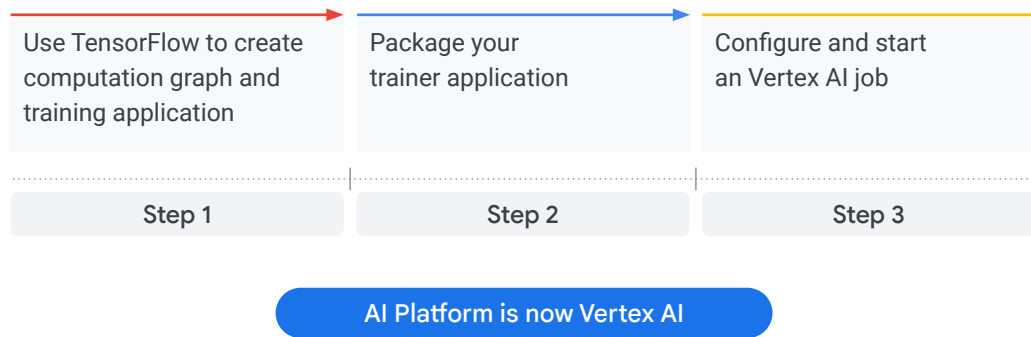
After you have a model that is working locally, when you have the parts setup and tested, you can scale it up using Vertex AI serverless technology.

That's when Big Data is processed and the model starts to become accurate enough for your purposes.

Each time you run through the training data is called an Epoch. And you would change some parameters to help the model develop more predictive accuracy.

As in this example, you can neatly connect and grow from a sample application in a notebook to Vertex AI.

Training your model with Vertex AI



This is the pattern for developing your own Machine Learning models.

- First, prepare the data
- That means you gather the training data, clean the data, split it into pools or groups for different purposes.
- Then you select features and improve them with feature engineering.
- Next, store the training data in an online location that Vertex AI can access such as in Cloud Storage.

Then you follow these steps. You use TensorFlow to create the training application. You package it. And then you configure and start a Vertex AI job.



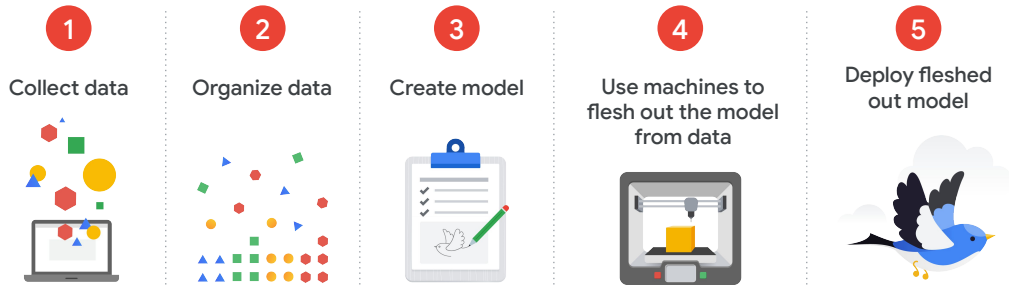
Machine Learning Terminology Review

Tom Stern



Let's talk more about Machine Learning. Machine Learning is composed of an orderly set of processes.

In reality, machine learning is...



TIP

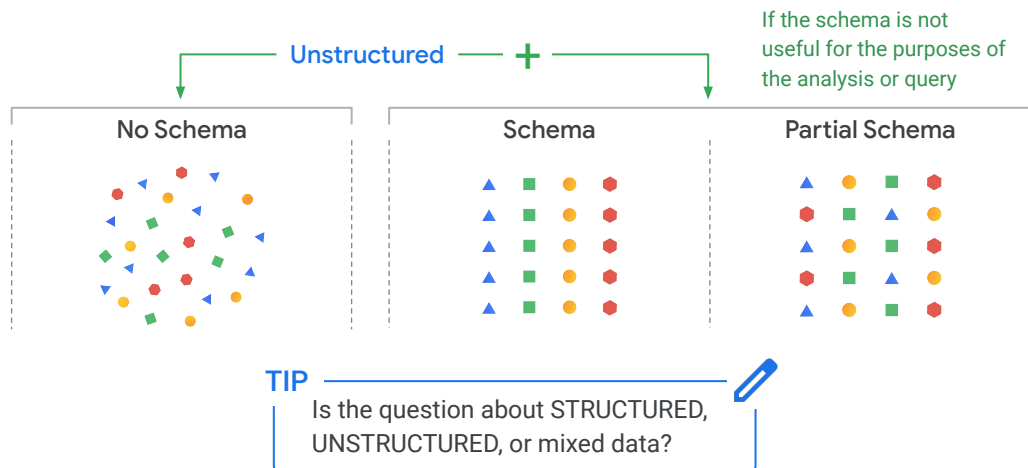
Which step or steps is the question asking about?

In questions about Machine Learning, make sure you identify the step. Some steps involve similar or related actions

On Google Cloud, we can use:

- Logging APIs, Pub/Sub, etc. and other real-time streaming to collect the data.
- BigQuery, Dataflow and ML preprocessing SDK to organize the data using different types of organization.
- Use TensorFlow to create the model.
- And use Cloud ML to train and deploy the model.

What qualifies as unstructured data?



Even data that has a schema might still be unstructured if it is not useful for your intended purpose.

Here is an example. Imagine that you are selling products online. After the product is delivered, an email is sent out asking for feedback about the experience. Upon reviewing the first dozen or emails you begin to regret not sending some kind of survey. Because compiling the results of the text from each email is going to be impossible. For the purpose of identifying best practices and worst practices, the email text data is unstructured.

However, you could use sentiment analysis to tag the emails and to group them. Let the machine learning do the reading for you, and sort the emails into representative groups.

Now you can look at the most positive and most negative emails to identify what behaviors to enforce or avoid. The Machine Learning process turned the unstructured data into structured data for your purposes.

Working with unstructured data

Human Reasoning

Real-time insight into supply chain operations. Which partner is causing issues?

Drive product decisions. How do people really use feature X?

Easy counting problems = big data

Did error rates decrease after the bug fix was applied?

Which stores are experiencing long delays in payment processing?

Harder counting problems = ML

Are programmers checking in low-quality code?

Which stores are experiencing a lack of parking space?

TIP

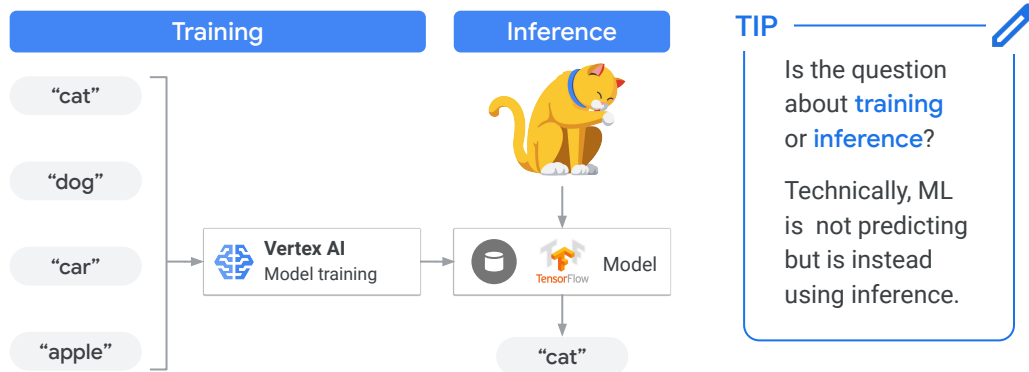
Is the problem suited for BIG DATA, MACHINE LEARNING, or human decision-making?



Distinguish between one-off (reasoning) problems that are best solved by humans, and Big Data problems that can be solved by crunching a lot of data, and Machine Learning problems that are best solved using modeling.

I was once asked if a Machine Learning model could distinguish upside-down images from right-side-up images. Could you train a model to do that? I suppose so. But most modern cameras add metadata into the image header about the orientation of the camera at the time the image was taken. That data is accurate and easily accessed. So in this case reading the metadata would be a better solution than training a Machine Learning model.

Data Engineers must focus on both the training and inference stages of ML



It is important to recognize that machine learning has two stages: training and inference.

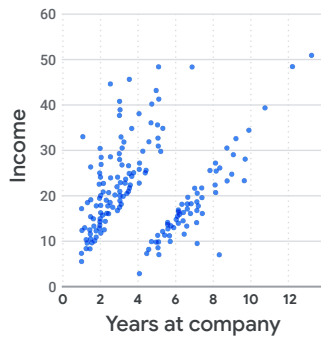
Sometimes the term "prediction" is preferred over "inference" because it implies a future state. For example, recognizing the image of the cat is not really "predicting" it to be a cat, it is really inferring from pixel data that a cat is represented in the image.

Data Engineers often focus on training the model, and minimize or forget about inference. It is not enough to build a model. You need to operationalize it. You need to put it into production so that it can run inferences.

In supervised learning, you have labels

Unsupervised Learning

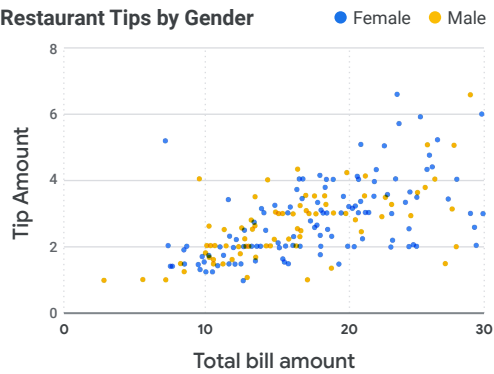
Income vs. Job Tenure



In **unsupervised** learning, the data is not labeled

Supervised Learning

Restaurant Tips by Gender



In **supervised** learning, we are learning from past examples to predict future values

If you have an ML Question that refers to LABELS it is a question about SUPERVISED LEARNING.

Regression and classification are supervised ML model types

1	total_bill	tip	sex	smoker	day	time
2	16.99	1.01	Female	No	Sun	Dinner
3	10.34	1.66	Male	No	Sun	Dinner
4	21.01	3.5	Male	No	Sun	Dinner
5	23.68	3.31	Male	No	Sun	Dinner
6	24.59	3.61	Female	No	Sun	Dinner
7	25.29	4.71	Male	No	Sun	Dinner
8	8.77	2	Male	No	Sun	Dinner
9	26.88	3.12	Male	No	Sun	Dinner

Option 1

Regression Model

Predict the tip amount

Option 2

Classification Model

Predict the sex of the customer

If the question is about REGRESSION or CLASSIFICATION it is using SUPERVISED ML.

A data warehouse can be a source of structured data training examples for your ML model

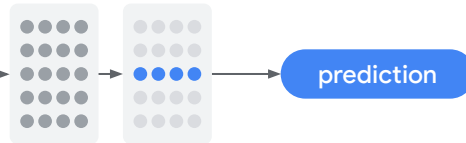
```
SELECT
  gestation_weeks,
  mother_age,
  cigarette_use,
  alcohol_use,
  weight_gain_pounds
FROM
  `bigquery-public-data.samples.natality`
WHERE cigarette_use is not null AND alcohol_use
```

Data on births is sourced from our BigQuery Data Warehouse using SQL.

TIP

Existing data warehouse may contain structured data (and maybe labeled data) that you can use to train your ML model.

weight	year	mother_age	gestation_weeks	cigarette_use	alcohol_use
7.86	2003	25	39	false	false
7.5	2003	21	39	false	false
8.06	2004	29	40	false	false
7.56	2004	38	37	false	false
7.06	2003	22	38	false	false



A very common source of structured data for machine learning is your data warehouse. Unstructured data includes things like pictures, audio, or video, and free-form text.

People sometimes forget that structured data might make great training data because it is already pre-tagged.

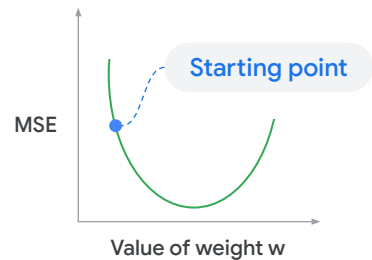
This example shows that birth data can be used to train a model to predict births.

Another example I like to use is real estate data. There is a ton of information online about houses, how big they are, how many bedrooms, and so forth. And also the history of when houses sold and how much was paid for them. This is great training data for building a home pricing evaluation model. In other words, the goal would be to describe the house to the Machine Learning model and have it return a price of what the house might be worth.

Mean squared error is a measure of loss

\hat{Y} -cap is the model estimate
 Y is the labeled value
Mean squared error (MSE) is:

$$\frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$



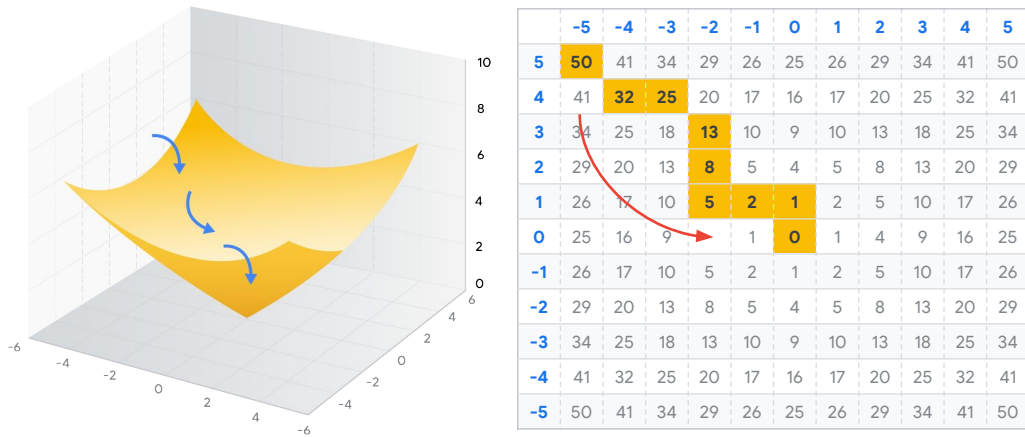
TIP

Mean Square Error (MSE)

If you don't define a metric or measure for how well your model works, how will you know it is working sufficiently to be useful for your business purpose?


You should be familiar with Mean Square Error or MSE.

Gradient descent is used to find the best parameters



Gradient descent is an important method to understand. It is HOW an ML problem is turned into a search problem.

Recompute error after each batch of examples (not full dataset)

TIP Can you describe RMSE? 



MSE and RMSE are measures of how well the model fits reality, how well the model works to categorize or predict.

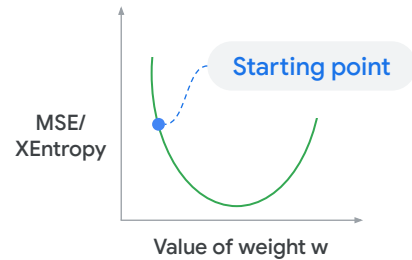
The Root of the Mean Square Error.

One reason for using the Root of the Mean Square Error rather than the Mean Square Error, is because the RMSE is in the units of the measurement, making it easier to read and understand the significance of the value.

For classification problems, we use cross entropy

For classification problems, the most commonly used error measure is cross entropy—because it is differentiable:

$$-\frac{1}{N} \sum_{n=1}^N \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right]$$



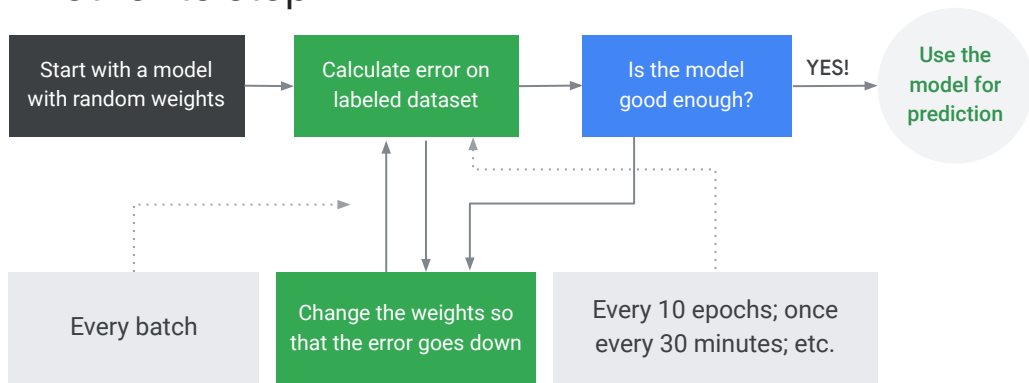
Categorizing produces discrete values and regression produces continuous values. Each uses different methods.

Is the result you are looking for like deciding whether an instance is in category "A" or category "B"? If so, it is a discrete value and therefore uses classification.

Is the result you are looking for more like a number. like the current value of a house? If so, it is a continuous value and therefore uses regression.

If the question describes cross-entropy... it is a classification ML problem.

Occasionally, evaluate the model to decide whether to stop

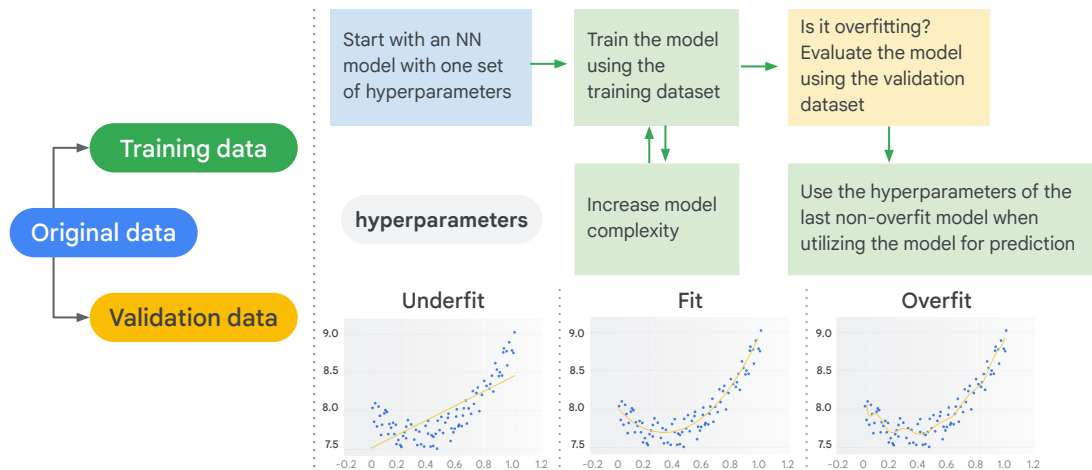


This is the implementation of the principal of "Practical, not perfect." When will your model be usable? When should you stop improving it?

If this step is missing, you could have run-away costs, poor performance, or a model that doesn't work sufficiently and is misleading.

Note that after we calculate error on batch, we can either keep going or we can evaluate the model. Evaluating the model needs to happen on full dataset, not just a small batch!

Split data, experiment with models



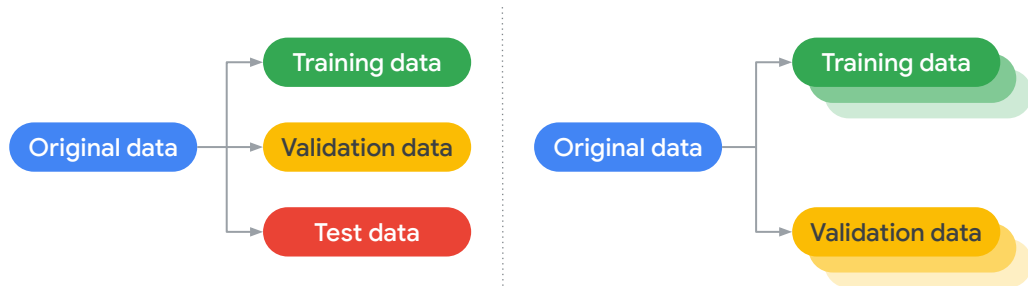
If you have one pool of data, then you will need training data and validation data. You can't use it all in both places or you won't get measurable error.

Training and evaluating an ML model is an experiment with finding the right generalizable model that fits your training dataset but doesn't memorize it. As you see here, we have an overly simplistic linear model that doesn't fit the relationships in the data. You'll be able to see how bad this is immediately by looking at your loss metric during training and visually on this graph here as there are quite a few points outside the shape of trend line. This is called underfitting.

On the opposite end of the spectrum is overfitting, as shown on the right extreme. Here we greatly increased the complexity of our linear model and turned it into an n -th order polynomial which seems to model the training dataset really well -- almost too well. This is where the evaluation dataset comes in -- you can use the evaluation dataset to determine if the model parameters are leading to overfitting. Overfitting or memorizing your training dataset can be far worse than having a model that only adequately fits your data.

If someone said they had a Machine Learning model that recognizes new instances and categorizes them correctly 100% of the time -- it would be an indicator that the validation data somehow got mixed up with the training data and that data is no longer a good measure of how well the model is working.

Use independent test data, or cross-validate
if data is scarce



Read these slides backwards. If the question says "data is scarce"... then you should be thinking "independent test data" or "cross-validate" are candidate answers. Be familiar with the various methods of cross validation, including training, validation and test, and cross validation.

Scenario #1

Question

Quickly and inexpensively develop an application that sorts product reviews by most favorable to least favorable.

- A. Train an entity classification model with TensorFlow. Deploy the model using Vertex AI. Use the entity to sort the reviews.
- B. Build an application that performs entity analysis using the Natural Language API. Use the entity to sort the reviews.
- C. Build an application that performs sentiment analysis using the Natural Language API. Use the score and magnitude to sort the reviews.
- D. Train a sentiment regression model with TensorFlow. Deploy the model using Vertex AI. Use the magnitude to sort the reviews.

Let's try some sample exam questions.

Your goal is to quickly and inexpensively develop an application that sorts product reviews by most favorable to least favorable.

Choose from the following answer choices.

- A. Train an entity classification model with TensorFlow. Deploy the model using Vertex AI. Use the entity to sort the reviews.
- B. Build an application that performs entity analysis using the Natural Language API. Use the entity to sort the reviews.
- C. Build an application that performs sentiment analysis using the Natural Language API. Use the score and magnitude to sort the reviews.
- D. Train a sentiment regression model with TensorFlow. Deploy the model using Vertex AI. Use the magnitude to sort the reviews.

Have your answer?

Scenario #1

Answer

Quickly and inexpensively develop an application that sorts product reviews by most favorable to least favorable.

- A. Train an entity classification model with TensorFlow. Deploy the model using Vertex AI. Use the entity to sort the reviews.
- B. Build an application that performs entity analysis using the Natural Language API. Use the entity to sort the reviews.
- C. Build an application that performs sentiment analysis using the Natural Language API. Use the score and magnitude to sort the reviews.
- D. Train a sentiment regression model with TensorFlow. Deploy the model using Vertex AI. Use the magnitude to sort the reviews.

C is the correct answer. Build an application that performs sentiment analysis using the Natural Language API. Use the score and magnitude to sort the reviews.

Scenario #1

Rationale

C is correct. Use a pre-trained model whenever possible. In this case the Natural Language API with sentiment analysis returns score and magnitude of sentiment.

A and D are incorrect because they require creating a model instead of using an existing one.

B is incorrect because entity analysis will not determine sentiment: it recognizes objects, not opinions.

<https://cloud.google.com/natural-language/docs/sentiment-tutorial>

<https://cloud.google.com/natural-language/docs/analyzing-entities>

<https://cloud.google.com/natural-language/docs/analyzing-sentiment>

The story here is to use a pre-trained model if it will do. Creating models is expensive and time-consuming.

Use a pre-trained model whenever possible. In this case, the Natural Language API with sentiment analysis returns a score and magnitude of sentiment that you can use to sort the reviews.

A and D are not correct because they require creating a model instead of using an existing one. And B is not correct because it is using the wrong Natural Language feature. Entity analysis will not determine sentiment; it recognizes objects, not opinions.

Scenario #2

Question

Maximize speed and minimize cost of deploying a TensorFlow machine-learning model on Google Cloud.

- A. Export your trained model to a SavedModel format. Deploy and run your model on Vertex AI.
- B. Export your trained model to a SavedModel format. Deploy and run your model from a Google Kubernetes Engine cluster.
- C. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on GPUs.
- D. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on TPUs.

How about this scenario? Your goal is to maximize speed and minimize cost of deploying a TensorFlow machine-learning model on Google Cloud.

Choose from the following answer choices.

- A. Export your trained model to a SavedModel format. Deploy and run your model on Vertex AI.
- B. Export your trained model to a SavedModel format. Deploy and run your model from a Google Kubernetes Engine cluster.
- C. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on GPUs.
- D. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on TPUs.

Have your answer?

Scenario #2

Answer

Maximize speed and minimize cost of deploying a TensorFlow machine-learning model on Google Cloud.

- A. Export your trained model to a SavedModel format. Deploy and run your model on Vertex AI.
- B. Export your trained model to a SavedModel format. Deploy and run your model from a Google Kubernetes Engine cluster.
- C. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on GPUs.
- D. Export 2 copies of your trained model to a SavedModel format. Store artifacts in Cloud Storage. Run 1 version on CPUs and another version on TPUs.

The answer is A, Export the trained model and deploy and run on Vertex AI.

Scenario #2

Rationale

A is correct because of Google recommended practices; that is, "just deploy it."

B is not correct because you should not run your model from Google Kubernetes Engine.

C and D are not correct because you should not export 2 copies of your trained model, etc., for this scenario.

<https://cloud.google.com/ml-engine/docs/deploying-models>

<https://cloud.google.com/ml-engine/docs/prediction-overview>

A is correct because it follows Google's recommended practices. It is a best practice to use each tool for the purpose for which it was designed and built.

So a tip here is to note when recommended best practices are called out, because those might be on the exam.

B is not correct because Google Kubernetes Engine is not the right tool for this circumstance. And C and D are not correct because in this situation you don't need to export two copies of the trained model.

Data Engineer Case Study 02:

A customer had this interesting business requirement...

Capture data reading and updates events to know who, what, when, and where.

Separation of who manages the data and who can read the data.

Allocate costs appropriately; costs to read/process vs. costs to store.

Prevent exfiltration of data to other Google Cloud projects and to external systems.

Case Study 02

This case involves a media company that has decided to move their in-house data processing into BigQuery. This example is focused on security and compliance.

As part of the migration, they have been moving their data centers from on-prem to BigQuery in the cloud. They have a lot of concerns about security. Who has access to the data they are migrating into the cloud? How is access audited and logged? What kind of controls can be placed on top of that? And they are very concerned about data exfiltration. They are worried about potential bad actors within the company, who, as part of their role have access to certain data. They want to make sure that employees who have access to that data cannot then take the data, load it onto their own computer, or load it onto another cloud project, and from there, perhaps, take that data somewhere else.

A customer had this interesting business requirement...

- Capture data reading and updates events to know who, what, when, and where.
- Separation of who manages the data and who can read the data.
- Allocate costs appropriately; costs to read/process vs. costs to store.
- Prevent exfiltration of data to other Google Cloud projects and to external systems.

We worked together to understand these business requirements and to help turn them into more technical requirements. We wanted to focus the technologies on the capabilities already available in BigQuery. So we introduced them to the concept of audit logs on Google Cloud, and specifically the default logs available from BigQuery. We presented them with Admin Logs that record creating and deleting datasets, and then the more detailed Access Logs that identify when people are reading datasets or perhaps even reading or accessing parts of the BigQuery UI.

We encouraged them to have everything managed by IAM. We developed groups based on role. Then assigned members to groups. And established permissions and applied those to the groups based on role.

Data Engineer Case Study 02:

We mapped that to technical requirements like this...

All access to data should be captured in audit logs.

All access to data should be managed via IAM.

Configure service perimeters with VPC service controls.

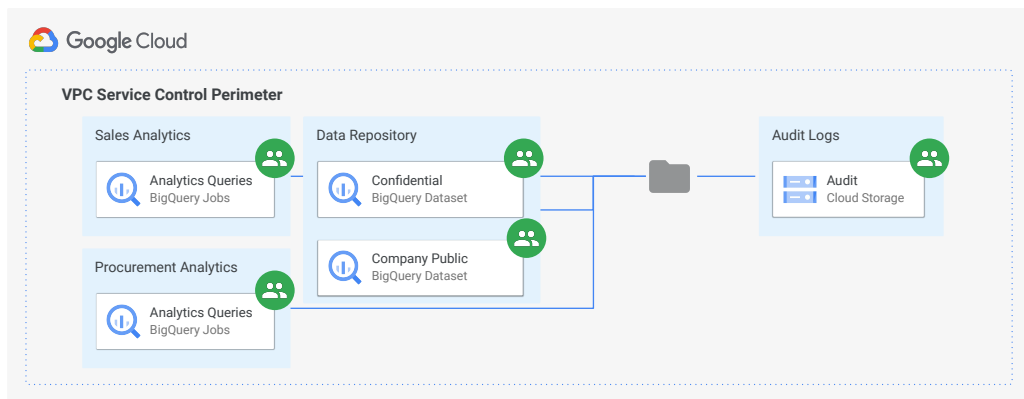
We mapped that to technical requirements like this...

Requirements

- All access to data should be captured in audit logs.
- All access to data should be managed via IAM.
- Configure service perimeters with VPC service controls.

Data Engineer Case Study 02:

And this is how we implemented that technical requirement



And this is how we implemented that technical requirement.

Each group was isolated in separate projects and allowed limited access between them using VPC Service Controls. BigQuery allows separation of access by role, so we were able to limit some roles to only loading data and other to only running queries. Some groups were able to run queries in their own project using datasets for which they only had read access, and the data was stored in a separate repository. We made sure that at the folder level of the resource hierarchy, we had aggregated log exports enabled. That ensured that even if you were the owner of a project and had the ability to redirect exports, you wouldn't be able to do so with specific exports, because those rights were set at the folder level, where most team members didn't have access. So by using aggregated log exports we were able to scoop up all the logs, store them in cloud storage, and create a record of who is running what query at what time against what dataset.

The VPC perimeter enabled us to allow APIs within the perimeter to run and only talk with other APIs belonging to other projects within the same perimeter.

So if someone had a separate project and started a BigQuery job that was to read from a dataset within the perimeter, even though they have credentials and access to the dataset, they would not be able to use or run the queries, because the APIs would not allow it at the perimeter.



Modeling Business Processes for Analysis and Optimization

Tom Stern



Modeling business process for analysis and optimization.

Modeling processes means thinking about them in some formal context or paradigm.

Some of these skills are not specific to Google or to Google Cloud Technologies. However, we will discuss a few common frameworks.

What happens if your model is wrong?

Confusion Matrix

		Reference	
		Positive	Negative
Prediction or Classification	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Recall

= True Positive rate

Accuracy

= Fraction correct

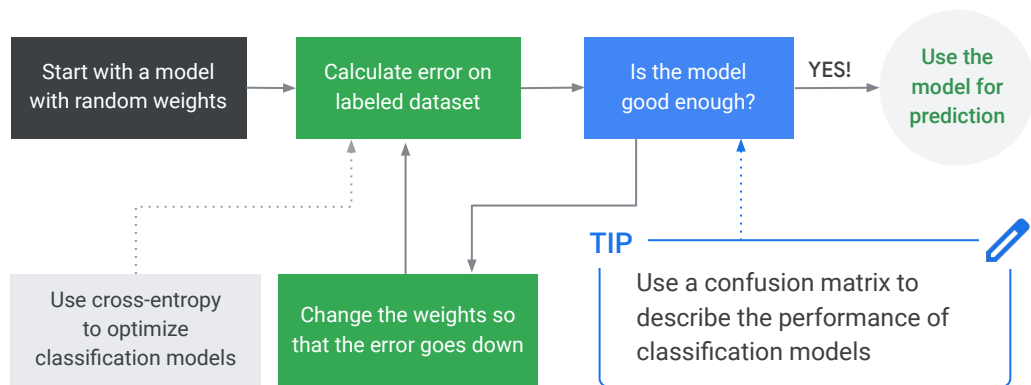
Accuracy fails if dataset is unbalanced

A confusion matrix classifies kinds of errors. There are two kinds of mistakes your model can make. The consequences are not the same.

Here is an example. Imagine you are predicting a dangerous condition in an automobile part. Positive means that the part is hazardous and dangerous. Negative means the part is safe. A False Positive means that your model predicted that the part was dangerous when it wasn't. So you removed a part that you didn't need to eliminate. A False Negative means that your model predicted that the part was safe when it was actually dangerous. So the part was used in an automobile because it was thought to be safe, and as a result, accidents occurred.

The logic could be reversed. If you were predicting that a part was safe, the False Positive would be that it is actually dangerous. So you have to think through this kind of logic problem to understand what business decision, procedure, or action should be taken as a result of the ML model.

Communicating to the business users



You may have to plan to explain your reasoning and design. In this example, a confusion matrix is used to make the data engineering choices understandable to the business users.

Your exam tip is: Use a confusion matrix to describe the performance of classification models.

Build, buy, or modify

Good

Train your own ML models

Build

TIP

What are the business priorities?

Inexpensive

Use AutoML to augment models

Modify

Buy

Fast

Use pre-trained models

What is the of business priorities?

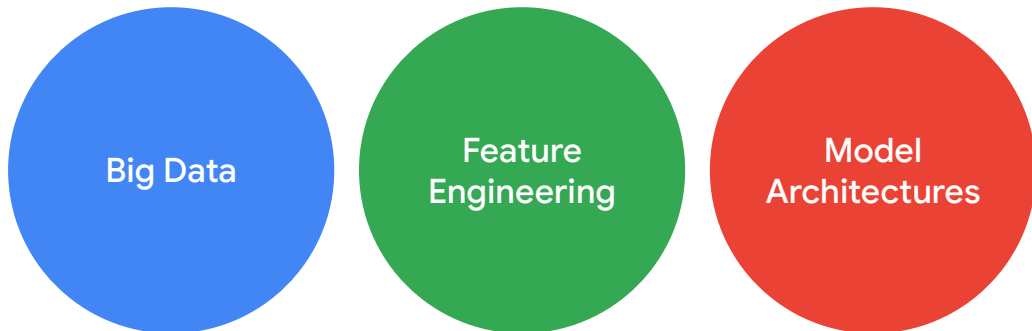
A high quality model takes time or money or both to develop.

Using pre-trained models is fast and inexpensive, but it may not be tailored to your needs.

A compromise is to use an existing trained model, but to build additional capability in the model. And this is the solution provided by AutoML.

You exam tip: Think about scenarios in terms of good, fast, and inexpensive. And identify what the question indicates the customer cares about.

Ways to build effective ML



One of the themes in ML is to start simply and build to production. Shown is the general progression of building an ML solution; start with Big Data. Go through Feature Engineering. Then create the Model and deploy it.

Feature engineering is a complicated process

Choosing good features

1. Is the feature you are considering related to the result you are trying to predict?
2. Is the predictive value known?
3. Is the feature a numeric value with meaningful magnitude?
4. Are there enough examples?

Feature engineering process

- Pre-processing
- Feature creation
- Hyperparameter tuning



Good features
bring human insight
to a problem

Other important concepts and considerations

- Feature crosses
- Discretize floats that are not meaningful
- Bucketize features
- Dense and sparse features
 - DNNs for dense, highly correlated
 - Linear for sparse, independent

Feature Engineering is a unique discipline. Selecting which feature or features to use in a model is critical. And there are a lot of things to consider, including whether the data of the feature is dense or sparse, and if the value is numeric, whether the magnitude is meaningful or abstract. Also, a good feature needs to have enough examples available to train, validate, and evaluate the model.

Learning rate is an important hyperparameter

Small step sizes can take a very long time to converge



Hyperparameters can determine whether your model converges on the truth quickly or not at all!

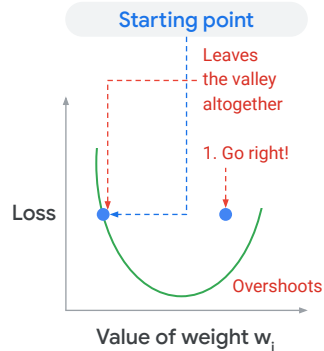
Small step sizes: Putting aside direction for the moment, if your step size is too small, your training might take forever. You are guaranteed to find the minimum though so long as there is only one minimum like this linear regression loss curve here if you use a small enough step size.

Learning rate is an important hyperparameter

Small step sizes can take a very long time to converge



Large step sizes may never converge to the true minimum



Larger step sizes: If your step size is too big, you might either bounce from wall to wall or bounce out of your valley entirely, and into an entirely new part of the loss surface. Because of this, when the step size is too big, the process is not guaranteed to converge.

Learning rate is an important hyperparameter

Small step sizes can take a very long time to converge



Large step sizes may never converge to the true minimum



A correct and constant step size can be difficult to find

Step size or "learning rate" is a hyper-parameter that is set before training



Correct step size: If your step size is just right Well, then, you're set. But whatever this value is, it's unlikely to be just as good on a different problem.

You exam tip: Better know about Learning Rate and Hyperparameter tuning in Machine Learning.

BigQuery performance

TIP

Performance is critical to practical solutions.



Performance is critical to practical solutions. In a case study there are often requirements that help define the level of performance required.

Here are some questions to consider:

BigQuery performance

Input data and data sources (I/O): How many bytes does your query read?

TIP

Performance is critical to practical solutions.



Input data and data sources (I/O): How many bytes does your query read?

BigQuery performance

Input data and data sources (I/O): How many bytes does your query read?

Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?

TIP

Performance is critical to practical solutions.



Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?

BigQuery performance

Input data and data sources (I/O): How many bytes does your query read?

Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?

Computation: How much CPU work does your query require?

TIP

Performance is critical to practical solutions.



Computation: How much CPU work does your query require?

BigQuery performance

Input data and data sources (I/O): How many bytes does your query read?

Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?

Computation: How much CPU work does your query require?

Outputs (materialization): How many bytes does your query write?

TIP

Performance is critical to practical solutions.



Outputs, also called materialization: How many bytes does your query write?

BigQuery performance

Input data and data sources (I/O): How many bytes does your query read?

Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?

Computation: How much CPU work does your query require?

Outputs (materialization): How many bytes does your query write?

Query anti-patterns: Are your queries following SQL best practices?

TIP

Performance is critical to practical solutions.



Query anti-patterns: Are your queries following SQL best practices?

Tax schema from the perspective of a data architect

Each of these data fields needs to
be stored in a structured way.

Form 990 (2018)

Part IX Statement of Functional Expenses

Section 501(c)(3) and 501(c)(4) organizations must complete all columns.
Check if Schedule O contains a response or note to any

Do not include amounts reported on lines 6b, 7a, 8b, 9b, and 10b of Part VIII.

	(a)	Total expenses
1 Grants and other assistance to domestic organizations and domestic governments. See Part IV, line 21 . . .		
2 Grants and other assistance to domestic individuals. See Part IV, line 22 . . .		
3 Grants and other assistance to foreign organizations, foreign governments, and foreign individuals. See Part IV, lines 15 and 16 . . .		
4 Benefits paid to or for members . . .		
5 Compensation of current officers, directors, trustees, and key employees . . .		
6 Compensation not included above, to disqualified persons (as defined under section 4958(f)(1)) and persons described in section 4958(c)(3)(B) . . .		
7 Other salaries and wages . . .		
8 Pension plan accruals and contributions (include section 401(k) and 403(b) employer contributions) . . .		
9 Other employee benefits . . .		
10 Payroll taxes . . .		
11 Fees for services (non-employees):		
a Management . . .		
b Legal . . .		
c Accounting . . .		
d Lobbying . . .		
e Professional fundraising services. See Part IV, line 17 . . .		
f Investment management fees . . .		
g Other. (If line 11g amount exceeds 10% of line 25, column (A) amount, list line 11g expenses on Schedule O.) . . .		
12 Advertising and promotion . . .		
13 Office expenses . . .		
14 Information technology . . .		
15 Royalties . . .		
16 Occupancy . . .		
17 Travel . . .		

Page 10 of the IRS Form 990 lists 24 different expense types.

What happens if I need to add another expense type? I need to change the schema and provide NULLs historically? Ugh..

Adding each expense field as a New Column

Table details: irs_990_2015

Refresh

Query Table

Copy Table

Export Table

Delete Table

Schema

Details

Preview

payrolltx	feesforsvcgmt	legalfees	feesforsvclobby	profndraising	advrtpromo	officepromo	officexpns	infotech	royaltsexpns	occupancy	travel
0	0	0	0	0	0	0	0	0	0	0	0
847695	0	0	28654	0	0	0	155715	43796	0	1156758	0
539651	127071	34165	44264	0	0	567392	732920	875416	0	887599	33446
209707	0	120	22000	0	0	0	165306	11391	0	231092	0

... results in a really WIDE table that is **not scalable**...

If we add each expense field as a new column, the table becomes very wide. And processing that wide table is not scalable.

Break out expenses into another Lookup Table

Organization Details

Company ID	Company Name
161218560	NY Association Inc.

Historical Transactions

Company ID	Expense Code	Amount
161218560	1	\$10,000

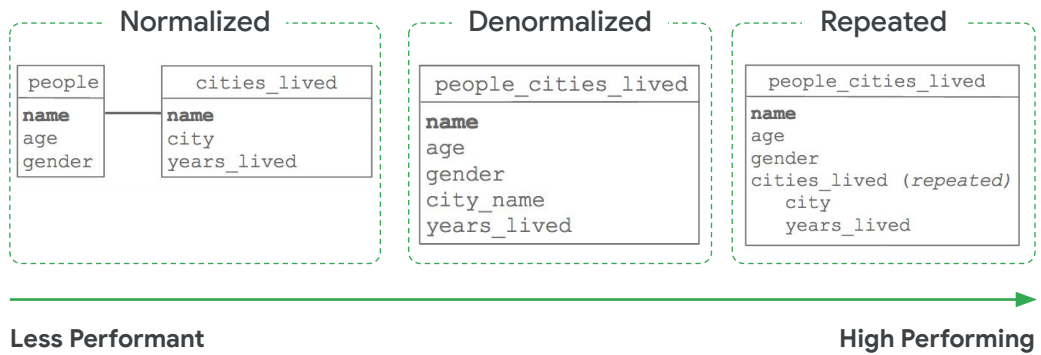
Code Lookup Tables

Expense Code	Expense Type
1	Lobbying
2	Legal
3	Insurance

... this breaking-apart process is called **normalization**

The tradeoff between relationalism and flat structure is called normalization. This process of breaking out fields into another lookup table, increasing the relations between tables, is called normalization.

BigQuery architecture introduces repeated fields



Normalization represent relations between tables.

Denormalized data presents information in a flat format.

Repeated fields enables related data to be handled in a loop, making it more efficient to process.

Normalize for efficiency

Original data

ID	Name	Transaction		
1	Bob			
2	Doug			
		OrderID	Date	Quantity
		11537	10/29/2018	11
		78244	11/05/2018	14
		32367	09/09/2018	18
3	Tom			
		OrderID	Date	Quantity
		13323	04/09/2018	6
		45452	11/10/2018	13
		17671	08/20/2018	15

Normalized data

Customer Table	
ID	Name
1	Bob
2	Doug
3	Tom

Transaction Table			
ID	OrderID	Date	Quantity
2	11537	10/29/2018	11
2	78244	11/05/2018	14
2	32367	09/09/2018	18
3	13323	04/09/2018	6
3	45452	11/10/2018	13
3	17671	08/20/2018	15

The trade-off is performance versus efficiency. Normalized is more efficient. Denormalized is more performant.

The original data is organized visually, but if you had to write an algorithm to process the data, how might you approach it? Could be by rows, by columns, by rows-then-fields. And the different approaches would perform differently based on the query. Also, your method might not be parallelizable.

The original data can be interpreted and stored in many ways in a database. Normalizing the data means turning it into a relational system. This stores the data efficiently and makes query processing a clear and direct task. Normalizing increases the orderliness of the data.

Denormalize for performance

Normalized data

Customer Table	
ID	Name
1	Bob
2	Doug
3	Tom

Transaction Table				
ID	OrderID	Date	Quantity	
2	11537	10/29/2018	11	
2	78244	11/05/2018	14	
2	32367	09/09/2018	18	
3	13323	04/09/2018	6	
3	45452	11/10/2018	13	
3	17671	08/20/2018	15	

Denormalized data

Denormalized Table				
ID	Name	OrderID	Date	Quantity
2	Doug	11537	10/29/2018	11
2	Doug	78244	11/05/2018	14
2	Doug	32367	09/09/2018	18
3	Tom	13323	04/09/2018	6
3	Tom	45452	11/10/2018	13
3	Tom	17671	08/20/2018	15

↑ Repeated field

Denormalizing is the strategy of accepting repeated fields in the data to gain processing performance.

Data must first be normalized before it can be denormalized. Denormalization is another increase in the orderliness of the data. Because of the repeated fields (in the example, the Name field is repeated), the denormalized form takes more storage. However, because it is no longer relational, queries can be processed more efficiently and in parallel using columnar processing.

Your exam tip: know and understand normalization and denormalization and when to apply each to your data representation design.

BigQuery can use nested schemas for highly scalable queries

Organization Details with Nested Historical Transactions

NESTED	Company ID	Company Name	Transactions.Amount	Code.Expense
	161218560	NY Association Inc.	\$10,000	Lobbying
			\$5,000	Legal
			\$1,000	Insurance
	123435560	ACME Co.	\$7,000	Travel

BigQuery can use nested schemas for highly scalable queries.

In the example shown, the company field has multiple (nested) transactions.

Four key elements of work

- **I/O:** How many bytes did you read?
- **Shuffle:** How many bytes did you pass to the next stage?
 - Grouping: How many bytes do you pass to each group?
- **Materialization:** How many bytes did you write to storage?
- **CPU work:** User-defined functions (UDFs), functions

Here are some items to consider when thinking about efficiency. Speed of processing, cost, and efficiency are related.

I would just point out that shuffling the data from one stage to another for the purposes of grouping can be a source of inefficiency that is not as easy to see or detect as slow input or output.

Long running jobs are a symptom. So you might want to measure resources and time between stages or to run tests on successively larger samples to verify how the pipeline is scaling.

Avoid input/output wastefulness

- Do not SELECT *, use only the columns you need.
- Filter using WHERE as early as possible in your queries.

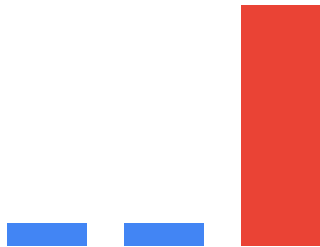
Remember to avoid using SELECT wildcard in SQL statements.

Filter with WHERE clause in SQL not LIMIT.

LIMIT only limits the output, not the work it took to get there.

WHERE filters on input.

Shuffle wisely: Be aware of data skew in your dataset



Skewed data creates an imbalance between BigQuery worker slots (uneven data partition sizes)

- **Filter your dataset** as early as possible (this avoids overloading workers on JOINS).
- Hint: Use the Query Explanation map and compare the Max vs. the Avg times to highlight skew.
- BigQuery will automatically attempt to reshuffle workers that are overloaded with data.

We already discussed shuffling. However, a hidden cause of inefficiency can be data skew.

The skewed data causes most of the work to be allocated to one worker and the rest of the workers sit and wait for that worker to complete.

Careful use of GROUP BY

- Best when the number of distinct groups is small (fewer shuffles of data).
- Grouping by a high-cardinality unique ID is a bad idea.

Row	contributor_id	LogEdits
1	2221364	4
2	104574	4
3	73576	4
4	311307	4
5	291919	4
6	140178	4
7	181636	4
8	3661553	4
9	3600820	4
10	4737290	4
11	938404	4
12	295955	4
13	183812	4
14	1811786	4
15	8918196	4
16	561624	4
17	5338406	4

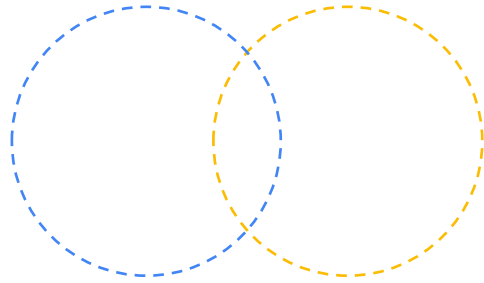
Do not Group
on an ID

The GROUP BY clause works best when the number of groups is small and the data is easily divided among them.

A large number of groups won't scale well. For example, a cardinality sort on an ID could cause increasingly poorer results as the data grows and the number of groups possible changes.

Joins and unions

- Know your join keys and whether they're unique: no accidental cross joins.
- LIMIT Wildcard UNIONS with `_TABLE_SUFFIX` filter.



Understand what fields you are using for keys when using JOIN.

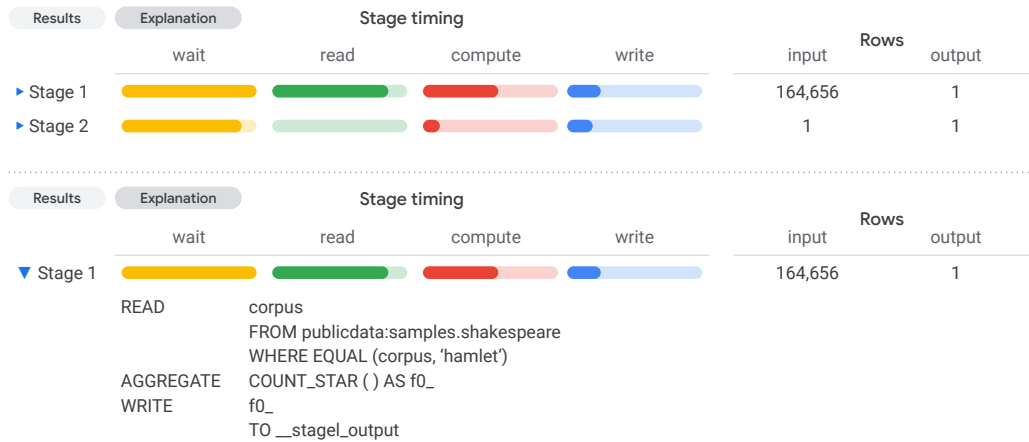
Limit UDFs to reduce computational load

- Use native SQL functions whenever possible.
- Concurrent rate limits:
 - for non-UDF queries: 100
 - for UDF-queries: **6**

```
CREATE TEMP FUNCTION SumFieldsNamedFoo(json_row STRING)
  RETURNS FLOAT64
  LANGUAGE js AS """
function SumFoo(obj) {
  var sum = 0;
  for (var field in obj) {
    if (obj.hasOwnProperty(field) && obj[field] != null) {
      if (typeof obj[field] == "object") {
        sum += SumFoo(obj[field]);
      } else if (field == "foo") {
        sum += obj[field];
      }
    }
  }
  return sum;
}
var row = JSON.parse(json_row);
return SumFoo(row);
""";
```

Limit the use of User Defined Functions. Use native SQL whenever possible.

Diagnose performance issues with Query Explanation map






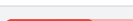

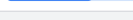


There are tools available such as the Query Explanation Map which shows how processing occurred at each stage.

This is a great way to diagnose performance issues and narrow down to specific parts of the query that might be the cause.

Diagnose performance issues with the Query Explanation map

The following ratios are also available for each stage in the query plan.

API JSON Name	Web UI*	Ratio Numerator **
waitRatioAvg		Time the average worker spent waiting to be scheduled.
waitRatioMax		Time the slowest worker spent waiting to be scheduled.
readRatioAvg		Time the average worker spent reading input data.
readRatioMax		Time the slowest worker spent reading input data.
computeRatioAvg		Time the average worker spent CPU-bound.
computeRatioMax		Time the slowest worker spent CPU-bound.
writeRatioAvg		Time the average worker spent writing output data.
writeRatioMax		Time the slowest worker spent writing output data.

* The labels 'AVG' and 'MAX' are for illustration only and do not appear in the web UI.

** All of the ratios share a common denominator that represents the longest time spent by any worker in any segment.

You will also find overall statistics and ratios that can be instructive. For example, the time the slowest worker spent reading input data, CPU-bound, or writing output data, which you can compare to the average.

Table partitioning

Time-partitioned tables are a cost-effective way to manage data.

Queries spanning time periods are straightforward to write.

When you create tables with time-based partitions, BigQuery automatically loads data in the correct partition.

To declare the table as partitioned at creation time, use this flag:

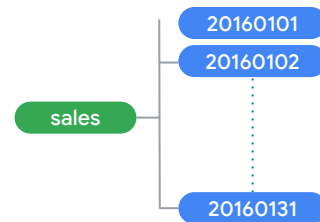
```
--time_partitioning_type
```

To create a partitioned table with expiration time for data, use this flag:

```
--time_partitioning_expiration
```

Example

```
SELECT ...  
FROM `sales`  
WHERE _PARTITIONTIME  
BETWEEN TIMESTAMP("20160101")  
AND TIMESTAMP("20160131")
```



Partitioning by time is often a way to evenly distribute work and avoid hot-spots in processing data.

In this example a partitioned table includes a pseudo column named `_PARTITIONTIME` that contains a date-based timestamp for data loaded into the table.

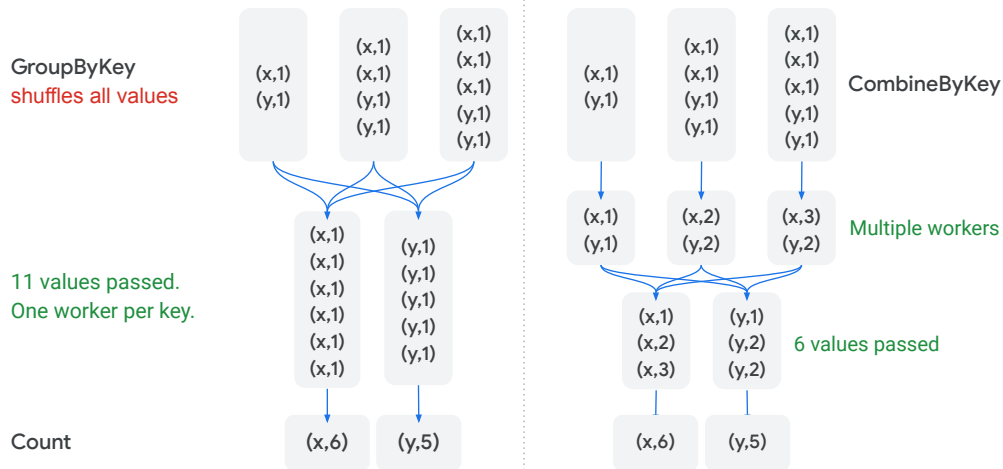
This improves efficiency for BigQuery.

There is more useful information available in the class and online.

But the opposite advice is recommended for selecting training data for Machine Learning.

When you identify data for training ML, be careful to randomize rather than organize by time, because, you might train the model on the first part, for example, on summer data, and test using the second part which might be winter data. And it will appear that the model isn't working.

Order of operations can influence shuffling overhead



Consider also the need to transfer data from one location to another over the network. Data transfer can be costly compared with a change in approach that might produce the same result with less data transfer.

In this example GroupByKey can use no more than one worker per key which uses all the values to be shuffled so they are all transmitted over the network. And then there is one worker for the 'x' key and one worker for the 'y' key, creating a bottleneck.

Combine allows Dataflow to distribute a key to multiple workers and process it in parallel. In this example, CombineByKey first aggregates values and then processes the aggregates with multiple workers. Also, only 6 aggregate values need to be passed over the network.

Can also group by time (Windowing)

- For batch inputs, explicitly emit a timestamp in your pipeline:
 - Instead of `c.output()`

```
c.outputWithTimestamp(f, Instant.parse(fields[2]));
```

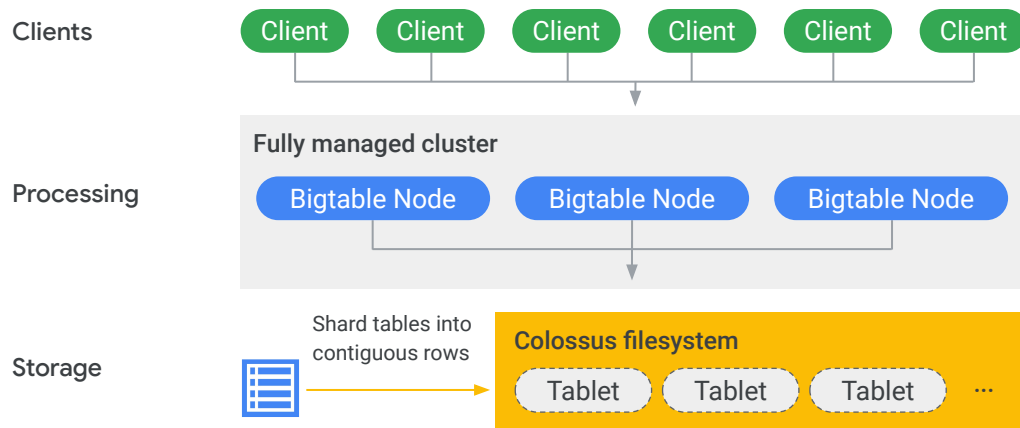
- Then use windows to aggregate by time:

```
PCollection<KV<String, Integer>> scores = input  
    .apply(Window.into(FixedWindows.of(Minutes(2))))  
    .apply(Sum.integersPerKey());
```

Subsequent Groups, aggregations, etc. are computed only within time window.

Consider when data is unbounded or streaming, that using windows to divide the data into groups can make the processing of the data much more manageable. Of course, then you have to consider the size of the window and whether the windows are overlapping.

Cloud Bigtable separates processing and storage



This slide goes into detail about how Bigtable stores data internally. A Bigtable table is sharded into blocks of contiguous rows, called tablets, to help balance the workload of queries.

Data is never stored in Cloud Bigtable nodes themselves; each node has pointers to a set of tablets that are stored in a storage service. Rebalancing tablets from one node to another is very fast, because the actual data is not copied. Recovery from the failure of a Cloud Bigtable node is very fast, because only metadata needs to be migrated to the replacement node.

When a Cloud Bigtable node fails, no data is lost.

Bigtable has efficiency when it comes to fast streaming ingestion -- just storing data....whereas BigQuery is efficient for queries because it is optimized for SQL support.

A table can have only one index (the row key)

Row Key	user_information				
	followers	birthdate	age	gender	messageCount
andrew	34	06_04_1986	34	F	1
brianna	31	07_24_1993	23	F	12
caitlyn	55	03_22_1952	51	F	15

Rows are stored in ascending order of the row key.



Entries in the birthdate column cannot be indexed.



In Bigtable, each table has only one index, the row key. There are no secondary indices. And the data is stored in the row key in ascending order. All operations are atomic at the row level.

Row key design



What is the most common query you need to support?

Make query parameter the row key
e.g. highway-MILEMARKER
(135-34)

This slide discusses how to use the row key.

The example is traffic data that arrives with a milemarker code and a timestamp -- indicating where vehicles were located at specific times. The question we are trying to answer is "Which value should be used as the index?"

If you store the data in timestamp order, the newest data will be at the bottom of the table. If you are running queries that are based on most recent data and only use older data for some queries, this organization is the worst case.

Row key design



What is the most common query you need to support?

Will you often need to retrieve records within a certain time range?

Make query parameter the row key
e.g. highway-MILEMARKER (I35-34)

Add reverse timestamp to the row key
e.g. highway-MILEMARKER-RTS (I35-347-123456789)

Instead, to optimize the query for the use case, you might want to consider using a reverse time stamp, so the newest data is always on top. But this doesn't work well either... it introduces a new problem.

Row key design



What is the most common query you need to support?

Make query parameter the row key
e.g. highway-MILEMARKER (I35-34)

Will you often need to retrieve records within a certain time range?

Add reverse timestamp to the row key
e.g. highway-MILEMARKER-RTS (I35-347-123456789)

Can you have both distributed writes and block reads?

Bad:
TS-highway-MILEMARKER (20170531T102354-I35-347)
MILEMARKER-highway-RTS (347-I35-1234567)

Because data is stored sequentially in Bigtable. Events starting with the same timestamp will all be stored on the same tablet. That means the processing isn't distributed.

Row key design



Good:

highway-MILEMARKER-RTS
(135-347-123456789)

What is the
most common
query you need
to support?

Will you often
need to retrieve
records within
a certain time
range?

Can you have
both distributed
writes and
block reads?

Make query parameter
the row key
e.g. highway-MILEMARKER
(135-34)

Add reverse timestamp
to the row key
e.g. highway-MILEMARKER-RTS
(135-347-123456789)

Bad:
TS-highway-MILEMARKER
(20170531T102354-I35-347)
MILEMARKER-highway-RTS
(347-I35-1234567)

In the end, this example used the milemarker code as the row key. It didn't necessarily make the data easy to read or faster for a specific kind of query, but it randomized the access so the query was distributed.

You exam tip: Know how indexes and key values influence performance and possibly introduce bias. Another example is time series, where your baseline data comes from Winter and your validation comes from Summer, introducing a seasonal bias into your analysis.

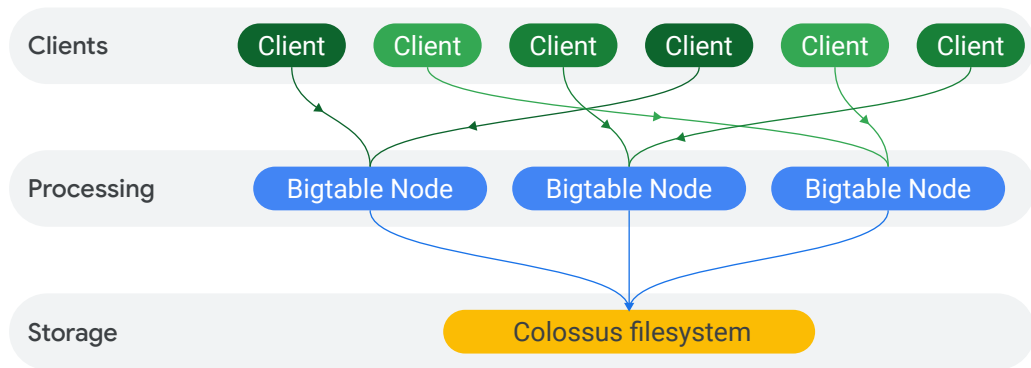
Years of engineering to...

- Teach Cloud Bigtable to configure itself.
- Isolate performance from “noisy neighbors.”
- React automatically to new patterns, splitting, and balancing.



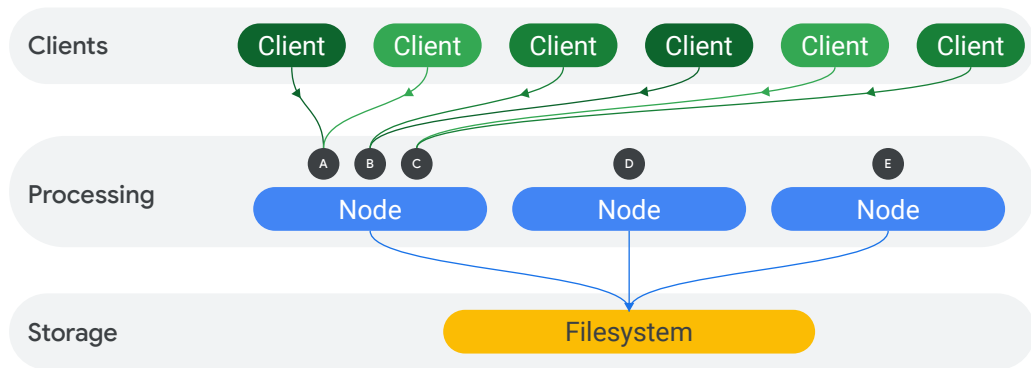
Cloud Bigtable is more sophisticated than we have so far described. It actually learns about your usage patterns and adjusts the way that it works to balance and optimize its performance.

Cloud Bigtable looks at access patterns and improves itself



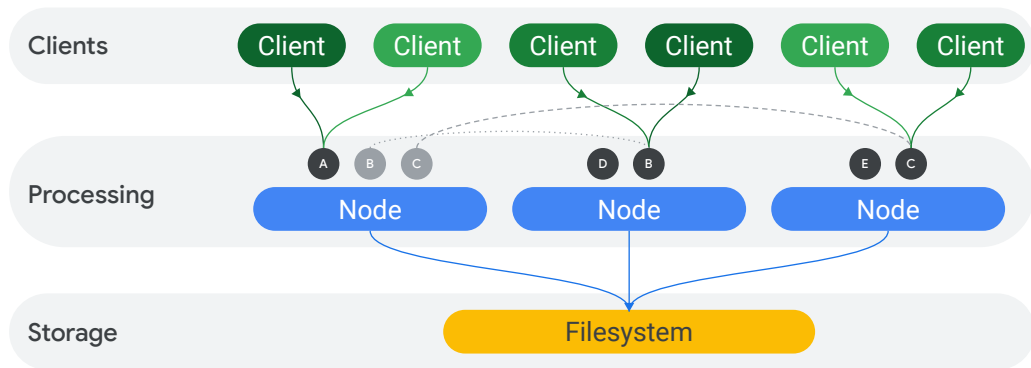
Cloud Bigtable looks at access patterns to improve itself.

Cloud Bigtable learns access patterns...



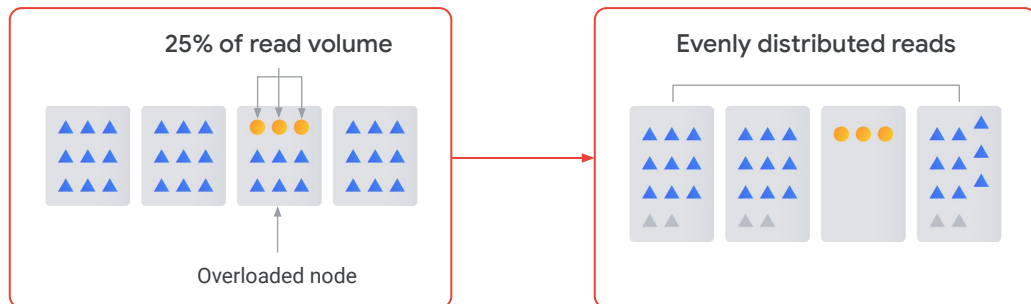
This example illustrates that because A,B,C,D,E are not data but rather pointers and references and cache rebalancing is not time consuming.

...and rebalances data accordingly



Moving the pointers rebalances which nodes do the work, but the data isn't copied or transferred.

Rebalance strategy: distribute reads



And this is an example when 25% of the read volume hits a single node. The overload condition is detected, and the pointers are shuffled to distribute the reads.

There are multiple copies of the data in the file service. For resiliency, the data needs to exist on different nodes. So Bigtable knows to use copies on other nodes in the file system to improve performance.

Growing a Cloud Bigtable cluster

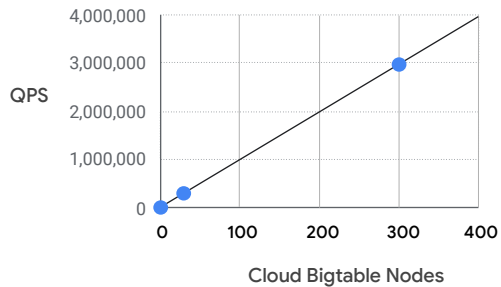
Make sure clients and Cloud Bigtable are in the same zone.

Change schema to minimize data skew.

Performance increases linearly with the number of nodes.

Disk speed: SSD faster than HDD

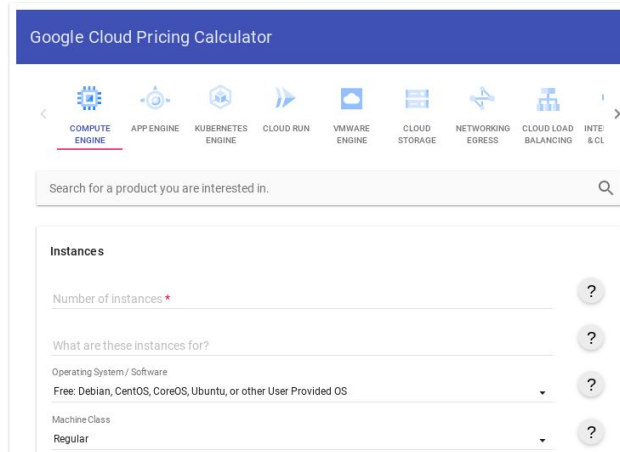
Takes some time after scaling up nodes for performance improvement to be seen.



Here are some tips to growing a Bigtable cluster. There are a number of steps you can take to increase the performance.

One item I would highlight is that there can be a delay of several minutes to hours when you add nodes to a cluster before you see the performance improvement. That makes sense, right? Because the data and pointers have to be reshuffled, and it takes Bigtable some time to figure out how to adjust the usage patterns to the new resources.

Pricing Calculator



The screenshot shows the Google Cloud Pricing Calculator interface. At the top, there's a blue header with the text "Google Cloud Pricing Calculator". Below the header is a navigation bar with icons and labels for various Google Cloud services: COMPUTE ENGINE, APP ENGINE, KUBERNETES ENGINE, CLOUD RUN, VMWARE ENGINE, CLOUD STORAGE, NETWORKING EGRESS, CLOUD LOAD BALANCING, and INTELLIGENT SCL. The "COMPUTE ENGINE" tab is currently selected. Below the navigation bar is a search bar with the placeholder text "Search for a product you are interested in." and a magnifying glass icon. Under the search bar, there's a section titled "Instances". This section contains four input fields, each with a question mark icon to its right: "Number of instances*", "What are these instances for?", "Operating System / Software" (with a dropdown menu showing "Free: Debian, CentOS, CoreOS, Ubuntu, or other User Provided OS"), and "Machine Class" (with a dropdown menu showing "Regular").

TIP

Performance and cost are related.

<https://cloud.google.com/products/calculator/>

The pricing calculator can be used with BigQuery to estimate the cost of a query before you submit it.

Three categories of BigQuery pricing

Storage	Processing	Free
<ul style="list-style-type: none">• Amount of data in table• Ingest rate of streaming data• Automatic discount for old data	<ul style="list-style-type: none">• On-demand OR Flat-rate plans• On-demand based on amount of data processed• 1 TB/month free• Have to opt in to run high-compute queries	<ul style="list-style-type: none">• Loading• Exporting• Queries on metadata• Cached queries• Queries with errors

TIP

Avoid SELECT *
Query only the columns that you need.

There are three elements of BigQuery pricing. Storage, Processing, and free activities, such as loading data.

So you don't pay to load the data, for ingest, but you do pay to store the data once it is loaded.

Exam tip is to be suspicious of anything with SELECT all. You need to understand the use of wildcards.

Use query validator with Pricing Calculator for estimates

1. Click on validator.
2. Shows data estimate.
3. Plug in to calculator.

TIP

Price your queries
before running them.

Valid: This query will process 6.36 GB when run.

Standard SQL Dialect

RUN QUERY

Save Query

Save View

Format Query

Show Options

Google Cloud Pricing Calculator

INTERCONNECT

BIQUERY

BIQUERYML

DATATUNE

FIRESTORE

DATAPROC

DATAFLOW

CLOUD SQL

SQL

Search for a product you are interested in.

BigQuery

ON-DEMAND

FLAT-RATE

Table Name

Name

Flood Zone Data

Location

US (multi-regional) (us)

Storage Pricing

Storage

1000

GB

Streaming Inserts

100

MB

Query Pricing

Queries

2

TB

ADD TO ESTIMATE

Estimate

BigQuery

Flood Zone Data

Location: United States

Storage: 1,000 GB

Streaming Inserts: 0 MB

Queries: 0.002 TB

USD 38.52

Flood Zone Data

Location: United States

Storage: 1,000 GB

Streaming Inserts: 100 MB

Queries: 2 TB

USD 37.53

Total Estimated Cost: USD 68.05 per 1 month

Estimate Currency

USD - US Dollar

EMAIL ESTIMATE

SAVE ESTIMATE

The query validator also estimates how much data will be used by the query before you run it. You can plug this into the pricing calculator to get an estimate of how much you will spend before you run the query.

Data Engineer Case Study 03:

A customer had this interesting business requirement...

The overall business requirement was to migrate to Cloud a on-premise reporting solution aimed to produce daily reports to regulators.

The on-premises solution was coded using a 'SQL-like language' and it was run on a Hadoop cluster using Spark/MapReduce (leveraging proprietary third-party software).

The client wanted to optimize the target cloud solution to:

- Minimize changes to their processes and codebase.
- Leverage PaaS and native cloud solution (i.e., avoid third-party software).
- Significantly improve performance (from hours to minutes).

Case Study 03

A customer had this interesting business requirement...

The overall business requirement was to migrate to Cloud a on-premise reporting solution aimed to produce daily reports to regulators.

The on-premises solution was coded using a 'SQL-like language' and it was run on a Hadoop cluster using Spark/MapReduce (leveraging proprietary third-party software).

The client wanted to optimize the target cloud solution to:

- Minimize changes to their processes and codebase.
- Leverage PaaS and native cloud solution (i.e., avoid third-party software).
- Significantly improve performance (from hours to minutes).

Data Engineer Case Study 03:

We mapped that to technical requirements like this...

Business Requirement	Technical Requirement
Minimum changes to their processes and codebase	<ul style="list-style-type: none">• Programmatically convert 'SQL-like' into ANSI SQL• No changes to source/target input structures• Automated 'black box' regression testing
Leverage PaaS and native Cloud Solution (i.e., avoid third-party software)	<ul style="list-style-type: none">• Minimize number of systems/interfaces, aim for full execution in BigQuery (i.e., remove the need for a Hadoop cluster)
Significantly improve performance (from hours to minutes)	<ul style="list-style-type: none">• Analyze and (manually) optimize the SQL code in BigQuery for performance

We mapped that to technical requirements like this...

Business Requirements

- Minimum changes to their processes and codebase
- Leverage PaaS and native Cloud Solution (i.e., avoid third-party software)
- Significantly improve performance (from hours to minutes)

Technical Requirements

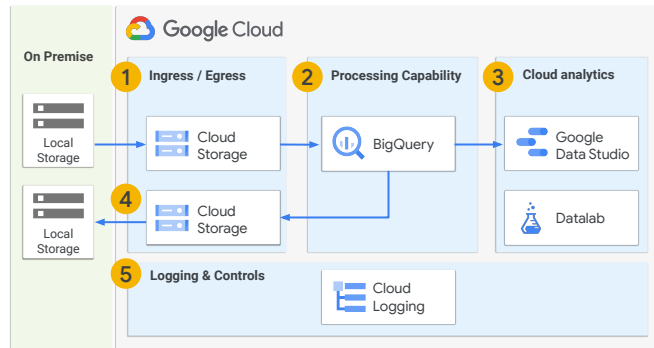
- Programmatically convert 'SQL-like' into ANSI SQL
- No changes to source/target input structures
- Automated 'black box' regression testing
- Minimize number of systems/interfaces, aim for full execution in BigQuery (i.e., remove the need for a Hadoop cluster)
- Analyze and (manually) optimize the SQL code in BigQuery for performance

The financial reporting applications run either daily or monthly. So we knew that all we needed to do was set up a pipeline to handle their requirements and then run the pipeline as required by their applications.

Data Engineer Case Study 03:

And this is how we implemented that technical requirement

1. Source data is ingested in Cloud Storage (no changes to sourcing).
2. financial reports are generated entirely in BigQuery.
3. Cloud Analytics allow users to review the reports.
4. Data is egressed on-premise (no changes to target structures -> no changes downstream).
5. Logs and controls out of the box in Cloud Logging.



And this is how we implemented that technical requirement.

- Source data is ingested in Cloud Storage (no changes to sourcing)
- Output reports are generated entirely in BigQuery
- Cloud Analytics allow users to review the reports
- Data is egressed on-premises (no changes to target structures → no changes downstream)
- Logs and controls out of the box in Stackdriver

We ended up with a fairly simple solution. The most notable part is what is not in the picture -- and that is Hadoop. Recall that the customer's application that was performing the processing was on Hadoop and some MapReduce. We were able to port that data out of the Hadoop cluster to Cloud Storage. Once it was in Cloud Storage, we found that the processing that was being done was simple enough that we were able to implement it in the processing front-end of BigQuery. This created the Liquidity report they wanted. And we were able to use this for analytics in Google Data Studio and for some analytics processing in Cloud Datalab.

The final reports are pushed first into Cloud Storage and then back into the on premise storage. This was important because it meant no changes in their business process were needed.

Challenge Lab 02

PDE Prep—Dataproc cluster
operations and maintenance:
Challenge Lab



A Challenge Lab has minimal instructions. It explains the circumstance and the expected results -- you have to figure out how to implement them.

This is a timed lab. It has activity tracking, which means when you complete a step the lab system will evaluate if the step has been performed correctly and will give you credit for it.

The lab will expire after 60 minutes.

The lab can be completed in about 45 minutes.

