6. Differences between data processing services

Dataflow	DataProc	Amazon Kinesis
- Stream and batch processing - Scalable processing (can process on several nodes at once) - Autoscaling, horizontal and vertical - Job visualization tools (Smart Diagnostics) - Templates to allow for easy sharing of pipelines - Supports Realtime ingestion from MySQL, PostgreSQL, SQL Server and Db2 - Support for customer managed encryption keys for data security	- Runs Apache spark, Apache Flink, Presto and other data processing tools - Allows for both managed and serverless processing - For non-serverless the system can scale from 1 node to hundreds - Integrates into BigQuery, Bigtable, Google Cloud Storage, Stackdriver Monitoring, Stackdriver Logging. - Offers highly availability - Processing nodes can run custom os images - Can use workflow templates	- Used for stream processing of data - No batch processing - Scalable processing (can process on several nodes at once) - Integrates with Amazon S3, DynamoDB, Amazon Redshift, Amazon EMR, Kinesis firehose - Support for customer managed encryption for data security

Dataflow reference - https://cloud.google.com/dataflow#section-10

Dataproc reference - https://cloud.google.com/dataproc#section-9

Kinesis reference - https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html

7. Using the gps_rtk.csv sensor_data from http://robots.engin.umich.edu/nclt/ in dataflow

The gps_rtk.csv has the following schema

Field	Description	Units
1	UTIME of the GPS fix	µs
2	Fix mode, as reported by the GPS unit: 0 means that the mode update is not yet seen, 1 means that there is no GPS fix, 2 means that the fix is good for longitude and latitude, 3 means that the fix is also good for altitude.	–
3	The number of satellites used in the fix	–
4	Latitude	rad
5	Longitude	rad
6	Altitude	m
7	Track	m
8	Speed	m/s

Source: http://robots.engin.umich.edu/nclt/nclt.pdf

A practical application for this data could be autonomous robot monitoring. One could use the data to monitor if the robot is moving. In the case of an autonomous food delivery robot, it should almost always be on the move. If it has stopped for more then a few minutes likely something is wrong. Below I propose a two dataflow pipelines that could be used to monitor a fleet of delivery robots.

1. Robot continuously publishes GPS data to Kafka on where it is
2. In Dataflow a streaming pipeline could be
    a. Consume Kafka message of delivery robot GPS data
    b. Deserialize message
    c. Calculate its trip stats (distance, time) based on the current position and last reported position
    d. Serialize the status message
    e. Publish the message in Kafka under a topic for processed streaming data
3. In Dataflow a batch pipeline could be
    a. Consume last 30m of Kafka message of GPS data
    b. Deserialize messages
    c. Calculate overall trip stats (distance, time) over the past 30 minutes
    d. Determine health of robot based on above stats
    e. Serialize the status message
    f. Publish the message in Kafka under a topic for processed batch data
4. Then some program subscribes to the processed Kafka topics and uses it for a status page to display delivery robot health for example.

Why use batch and streaming for this you may ask. Well, a streaming process may have a lot of false positives of non-movement being an issue, e.g., waiting for an obstruction to clear. This would be especially true with very frequent reporting time internals (<1m). Real-time status allows those monitoring the machine to see potential problems early. The batch job on the other hand will have way less false positives of non-movement being an issue as it's monitored over a longer time span. Though with the batch job issues may not be caught right away but will be more definite if there is an issue.