An Open Dynamic Big Data Driven Application System Toolkit

Craig C. Douglas

University of Wyoming and

KAUST

This research is supported in part by the National Science Foundation and King Abdullah University of Science & Technology (KAUST).

Computational Sciences Morphs into Data Intensive Scientific Discovery (DISD)

- Big Data has become a superset of computational sciences with applications in all walks of life with the overriding question, "What if you had all of the data?"
- Technology limits to some extent how big the Big Data can be, but over time the size has increased by 1,000 fold every few years since the 1950's.

What Is Big Data?... It Depends

Unit	Approximately	10^n	Related to
Kilobyte (KB)	1,000 bytes	3	Circa 1952 computer memory
Megabyte (MB)	1,000 KB	6	Circa 1976 supercomputer memory
Gigabyte (GB)	1,000 MB	9	Mid 1980's disk controller memory attached to a mainframe (with 128 MB memory)
10 Gigabytes (GB)	10,000 MB	10	2013 typical memory stick (16 GB)
Terabyte (TB)	1,000 GB	12	2012 largest SSD in a laptop
Petabyte (PB)	1,000 TB	15	250,000 DVD's or the entire digital library of all known books written in all known languages
Exabyte (EB)	1,000 PB	18	175 EB copied to disk in 2010 (est.)
Zettabyte (ZB)	1,000 EB	21	2 ZB copied to disk in 2011 (est.)
10 Zettabytes (ZB)	10,000 EB	22	Single NSA Big Dataset in 2013 (est.)

Dynamic Big Data Driven Application Systems (DBDDAS)

- DBDDAS research is motivated from the following difficult key points:
  - Obtaining and communicating out of sequence remote data of unknown quality to remote computers and using it to steer simulations.
  - The obvious societal gains from more accurate, long term forecasts from nonlinear, time dependent, complex scientific and engineering models.
  - Updating the entire DDDAS paradigm using recent Big Data research concepts.

DDDAS and BD

- All DDDAS have been dealing with the Big Data issue as long as there have been DDDAS.
- Each project has individually invented Big Data concepts with no real field wide consensus on how to handle Big Data.
- Combining the two paradigms not only makes sense, but is essential to creating new DBDDAS quickly and efficiently.
- *Re-inventing the wheel is bad science once the wheel has been identified*.

DDDAS Paradigm

- Extends conventional applications by adding the following:
  - Model state save, modification, and restoration.
  - Ensemble data assimilation algorithms to modify ensemble member states by comparing the data with synthetic data of the same kind created from the simulation state.
  - Multiscale algorithms for interpolation within models.
  - Retrieve, filter, and ingest data from intelligent sensors with potentially significant processing capabilities.
  - Visualize computed results on a wide variety of devices including mobile ones such as a tablet or smart phone.

Big Data Paradigm

- BD adds
  - Structured query language (SQL) and Not only SQL (NoSQL) capabilities for storing and retrieving data from dynamically growing databases or data streams.
  - Flexible methods for handling large quantities of data in highly parallel computing environments, e.g., MapReduce, a parallel merge-sort algorithm.
  - Fast, parallel read-write capabilities, e.g., NetCDF or HDF5.
  - Extremely large, robust, distributed file systems.
  - Time dependent data retrieval based on a dynamically chosen time windows so that automatic weighting of data based on how recent it was acquired can be easily adopted.
  - Big Data visualization.

DBDDAS Commonalities

- Sensors should provide data with known error bounds within time constraints.
- Correct sensors need to be chosen to provide good data from specific locations.
- Data is not guaranteed to be accurate, complete, nor timely, but will arrive anyway and must be sensibly processed, including discarding.
- Sensors should be dynamically reprogrammable during simulations.

DBDDAS Commonalities

- Rapid error growth in new data when steering is unavailable frequently occurs, causing simulations to be restarted. Automatic new methods to recognize and curtail rapid error growth are essential. New data intensive methods are helpful.
- An application's models, numerical methods, and most significant items that are tracked may need to be changed as a result of the incoming sensor data or a human in the loop.
- Extremely large quantities of input and/or output data that is difficult to analyze.
- Inadequate visualization of output data common.

DBDDAS Commonalities

- Using Big Data analytic techniques,
  - Much longer running simulations are possible instead of running a large sequence of short running simulations and then combining each result into a video of predictions.
  - Additionally, data can be better analyzed, visualized, managed, and accessed faster using Big Data methods than ad hoc data management methods created by nonexperts who excel otherwise in creating application simulators.

OpenDBDDAS Framework

- Much of the framework exists, but is not tied together.
- An open source sensor language standard that stays open.
- An open source MapReduce suitable for medical records that
  - Must be HIPPA compliant.
  - Must scale well for very large databases.
  - Must have individual/group access capabilities (ala Linux files) for records.
  - Must not have complexity O(disk access) on a DFS.
    - Should use OpenMP and MPI.
    - Should use (disk) cache aware hashing methods.
  - *Will be useful well beyond medical records*.

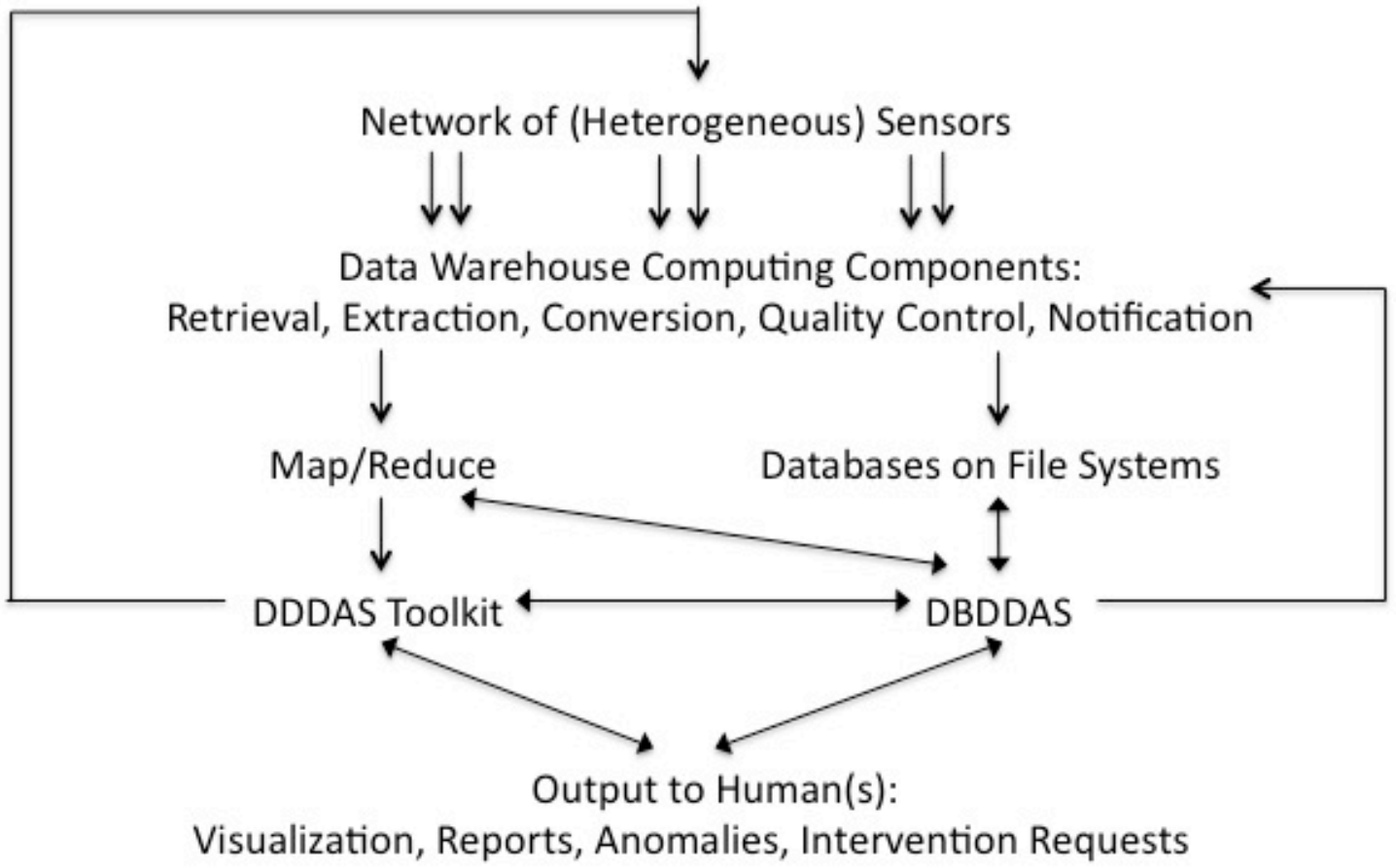
OpenDBDDAS Framework

- Cache aware, fast hash tables (not textbook variety).
- High performance disk I/O assuming there is already a (distributed) file system.
- SQL and NoSQL databases.
- Data warehouse with six point functionality
  - retrieval, extraction, conversion, quality control, store, notification
  - Intelligent sensing and processing (ISP) capable.

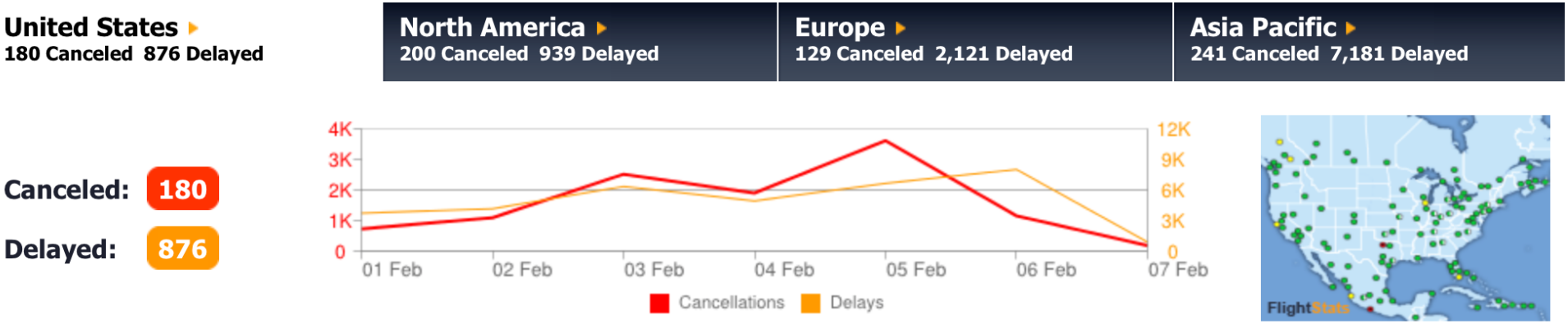
OpenDBDDAS Framework

- DDDAS toolkit that allows
  - Uncertainty quantification and statistical analysis.
  - Numerical solvers for standard (nonlinear time dependent coupled) PDEs.
  - Optimization solvers.
  - Data assimilation using multiscale interpolation methods.
  - Item identification, tracking, and steering.
  - Anomaly tracking and verification.
  - Human notification methods.
- *Must be usable by non-computer scientists*.

Typical Use of OpenDBDDAS Framework



Travel App: FlightCast

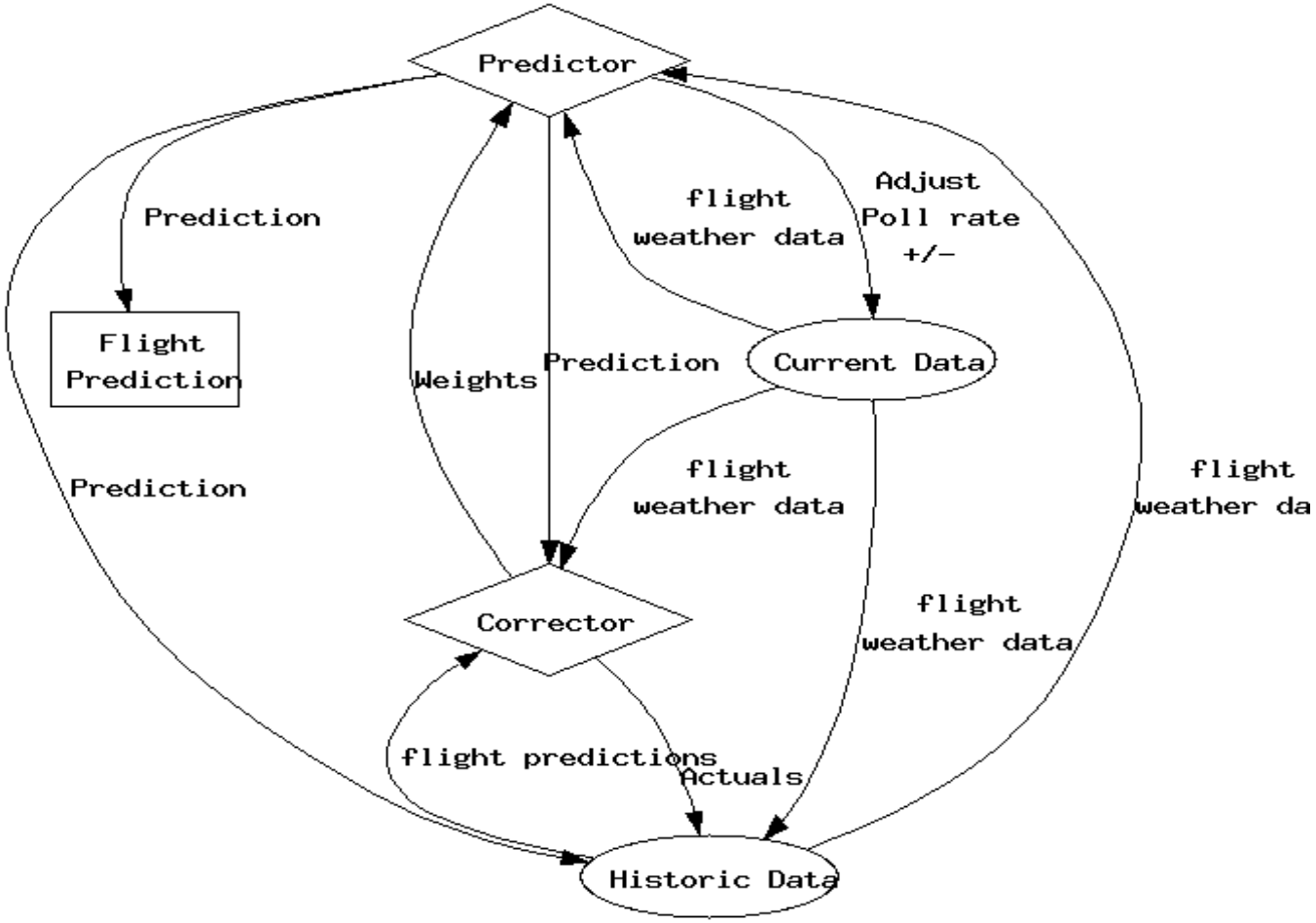


- A DBDDAS for flyers. Aimed primarily at:
  - Frequent flyers,
  - Particularly flying commuters.

FlightCast

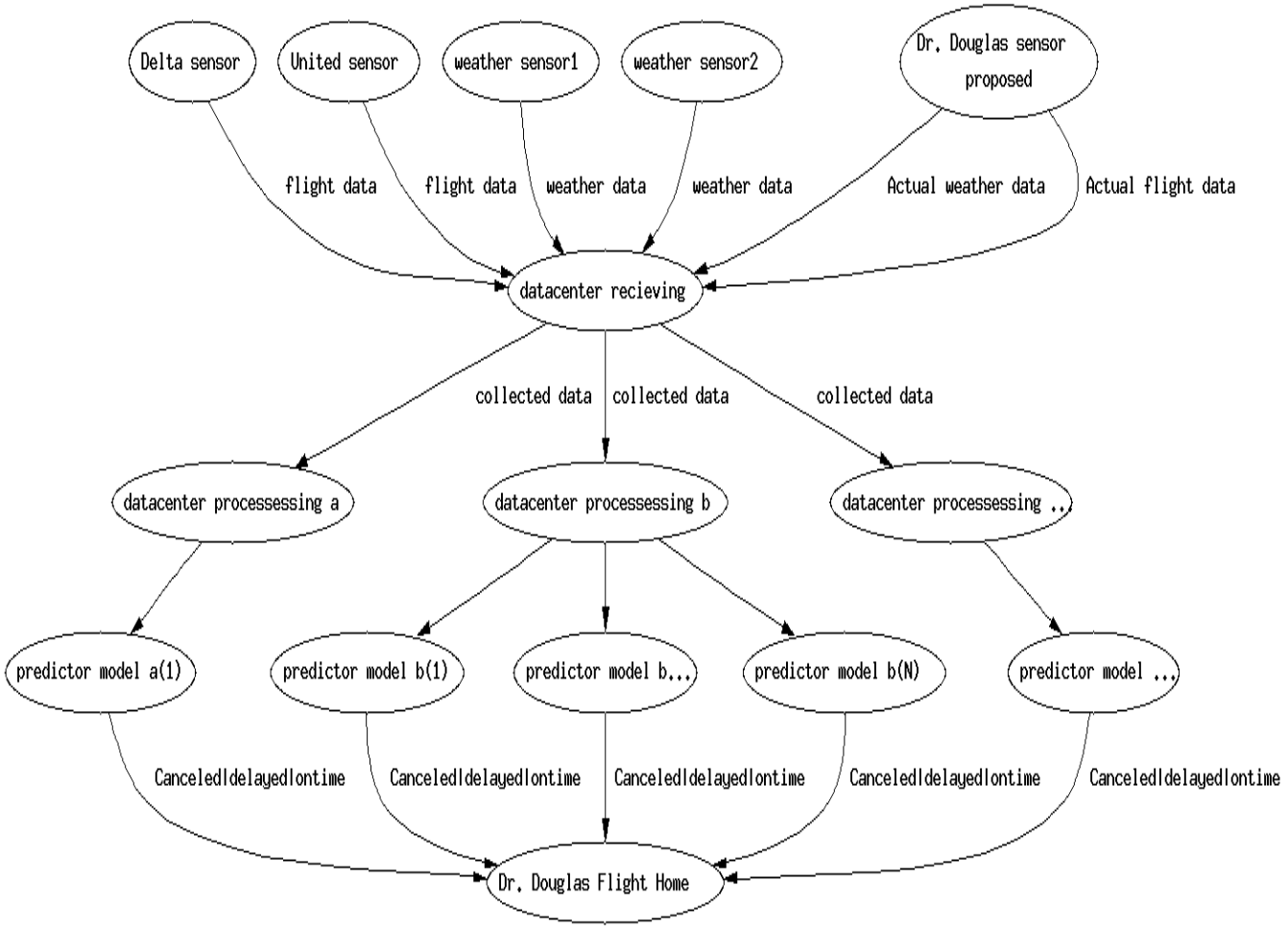
- Model airplane connections for a given route and day. Project is broken into multiple functional units:
  - Sensors: poll and collect data (software agents)
  - Transport: move data inside the DDDAS.
  - Data store: repository for Big Data.
  - Predictor: makes delay predictions.
  - Corrector: analyze accuracy of predictions and modify parameters (uses BD to solve an inverse problem cheaply).

FlightCast



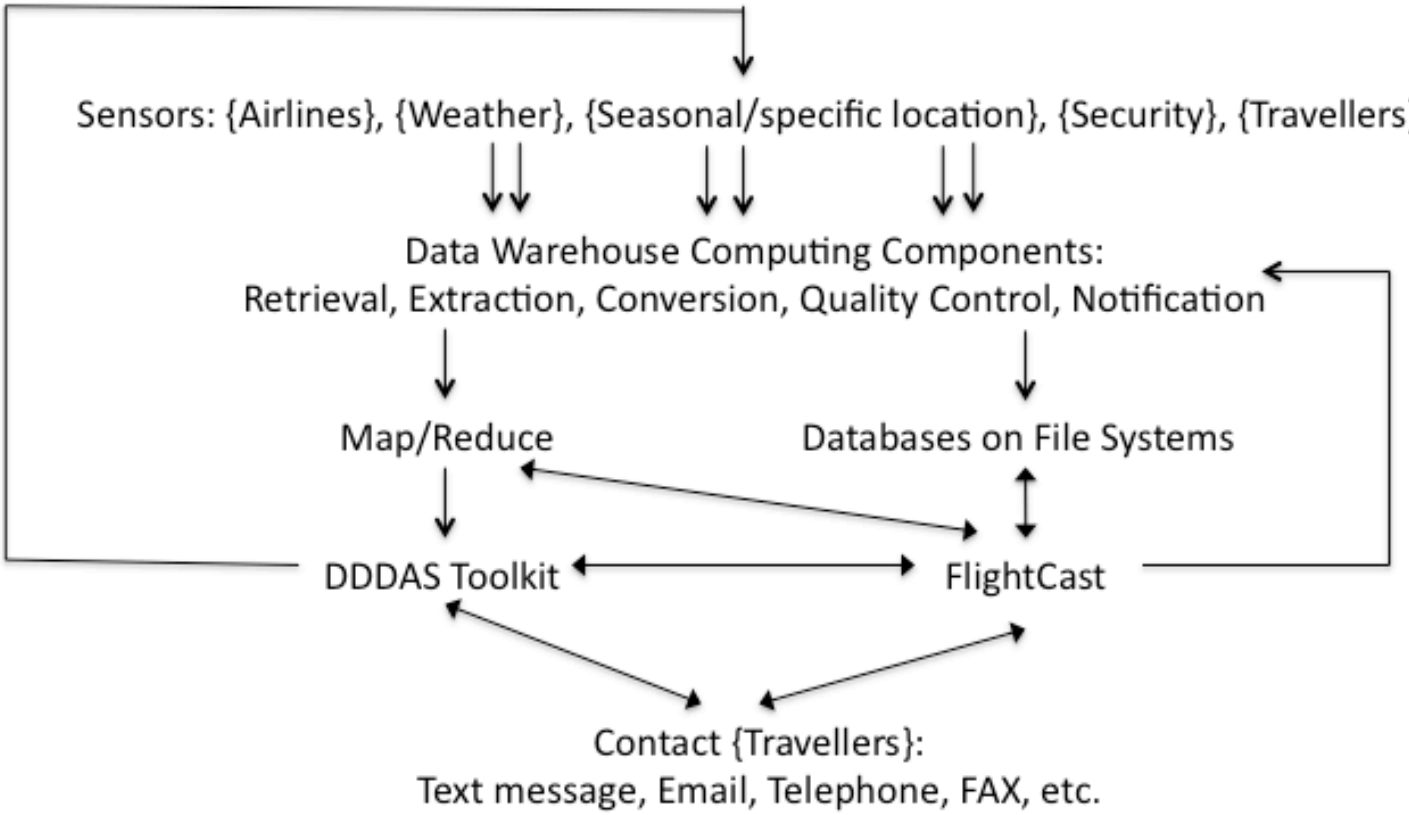
DDDAS PredictorA model

FlightCast



DDDAS flight prediction outline

Putting It Together



DBDDAS

- Five major components
  1. *Sensors*: collect data for the DDDAS – software agents.
  2. *Transport*: transfer data between the sensors and the other parts of the DDDAS.
  3. *Data store*: central repository for collected data.
  4. *Predictor*: predict flight delay and cancellation based on historical data and current conditions.
  5. *Corrector*: analyze the accuracy of predictions and make corrections to the predictor to improve accuracy over time.

Sensors

- Travellers
  - Tracks traveller location before, during, after flight(s)
  - Monitors preferences (want on time, want misses, etc.)
  - Maximize frequent flyer miles given conditions
- Airlines, Threat level, and Weather (similar)
  - Low level web requests to avoid graphics, ads, etc. – cookies helped.
  - Used tools for constructing simple compilers.
  - Weather: wind speed + direction, visibility, temperature, type of precipitation (could have added more)
  - Lots of data from airlines and weather. Less so from TSA.

Sensors and Transport

- Seasonal and Location specific trends
  - This required a training period of at least one year.
  - We used a number of years worth of weather data specific to a set of airports to build a database and correlated flight delays and cancellations to it.
- Transport
  - Java based
  - GRID computing enabled
  - Stored data in a SQL database.

Predictor-Corrector

- Predictor
  - Combined data from the relevant sensors (airline, weather, threat level, seasonal, and a specific traveller),
  - Weighted the data,
  - Produced a probability (from 0 to 1) of either a significant delay or cancellation.
  - Each factor in the probability was produced using fuzzy logic. The factors used in the predictions were stored in the database for use by the corrector component later.

Predictor-Corrector

- Corrector
  - Component reviewed the results of the predictor on a regular basis
  - It is inexpensive computationally due to the information saved in the prediction component.
  - The main purpose of the corrector is to adjust weights used in the predictor component.
- Predictor/corrector components ran multiple times before the actual flight(s) and the corrector ran once more after the flight(s).

FlightCast Most Like Future DBDDAS

- Building up a database of flight information for just Delta and United airlines over one year generated several disk drives worth of data, even with relatively low sampling rates and used all standard BD techniques known.
- After several months of operating FlightCast tracking one particular frequent flyer with a regular commute between New York's Laguardia and Lexington's airports, FlightCast had a better than 95% correct prediction rate.
- The two airlines tracked had nothing similar to FlightCast and could only react to aircraft delays and maintenance issues on the day of flights.
- To this day there is no tool similar to FlightCast available to the general public.