White Paper

How Happy Is London?

An analytical approach to creating a point-in-time view of the happiness of London.

David Fearne - Technical Director, Arrow, UK and Ireland

Abstract

How Happy Is London? was devised as a way to demonstrate the power of large-scale analytics. By looking at multiple, freely available data sources and applying simplistic weighting algorithms, it is possible to get a point-in-time consensus as to the happiness of London. This demonstration is designed to give the observer an idea of how to take the methods and theories employed and apply them within their own business.

The project is a demonstration of both net result and visual analytics. It leverages both structured and semi-structured data sources to process 2.6 billion data points a day to answer a single point-in-time question accurately: Is London happy?

Overview

Whenever faced with an analytics problem, it is important to define first what the required outcome will be. Using this approach, one can work backwards from the business need rather than up to an outcome that may not match what the business requires.

This is critical of all major technology projects undertaken today, but few have such a large potential business impact as analytics. Its sole purpose is to aid and inform better business decision-making. It's also important to define the final purpose of the analytics process. Will the result drive automated changes, display greater details or provide point-in-time or predictive decision assistance?

The project goal was to design a system that could show the power of analytics in such a way that its message could be easily grasped by board room executives, operational executives and middle management.

This required an approach that would engage a user, helping them to understand how to gain a point-in-time snapshot of their business or organisation, but also enable them to delve deeper into the data to identify opportunities and gain a business advantage over the competition.

The demonstration had to show how this is achieved using large sets of data that were unrelated in themselves, but once used to enrich the initial data streams became valuable insights that were simply unachievable via conventional methods.

The project uses data sources found in London to create a point-in-time predictive analysis of whether London is generally 'Business as usual', 'Happy', 'Life's good' or 'On top of the world'. This is then used to control a video representation of the mood of London. Around the outside of the video, we then display the various data feeds as we process them to create the indexes.

Note: *How Happy Is London?* is designed to be a demonstration of different methodologies and techniques that could be implemented to help businesses gain commercial or intellectual advantage. The goal of the system, and this paper, is to act as a reference architecture to be extended and consolidated to provide a more focused analytical solution.

Analytics architecture overview

Analytics systems follow a common foundation architecture that is transferable between different vendors, data sources and open source systems. Please see Table 1 on next page. This model helps us to provide an overall plug-and-play framework. *How Happy Is London?* follows both structured and semi-structured analytical frameworks.

Data flow through analytics system				
Data type	Ingestion	Extract, transform and load	Store	Explore
Structured	Database, APIs	Basic conversions between string to int, Removal of fields etc.	Enterprise data warehouse (EDW), Database	Visualistion, BI/BA, APIs
Semi-structured	APIs, Web, Text	Regular Expression, Hadoop	HDFS, No SQL, Graph Database	Visualistion, BI/BA, APIs, EDW
Unstructured	Images, Audio, Video	Machine Learning, Artificial Neural Networks	HDFS, Memory,	APIs, EDW

Table 1 - Data analytics model

System architecture

The technical system architecture is split into five layers. This allows for horizontal scaling at any layer of the system to cope with varying loads. These layers also closely mirror the data analytics model from Table 1, with the exception of security.

Those layers are:

- Security
- Data Intergration
- Data Processing
- Information Storage
- Information Presentation

This section will describe the function requirements for an analytics platform as well as the choices made.

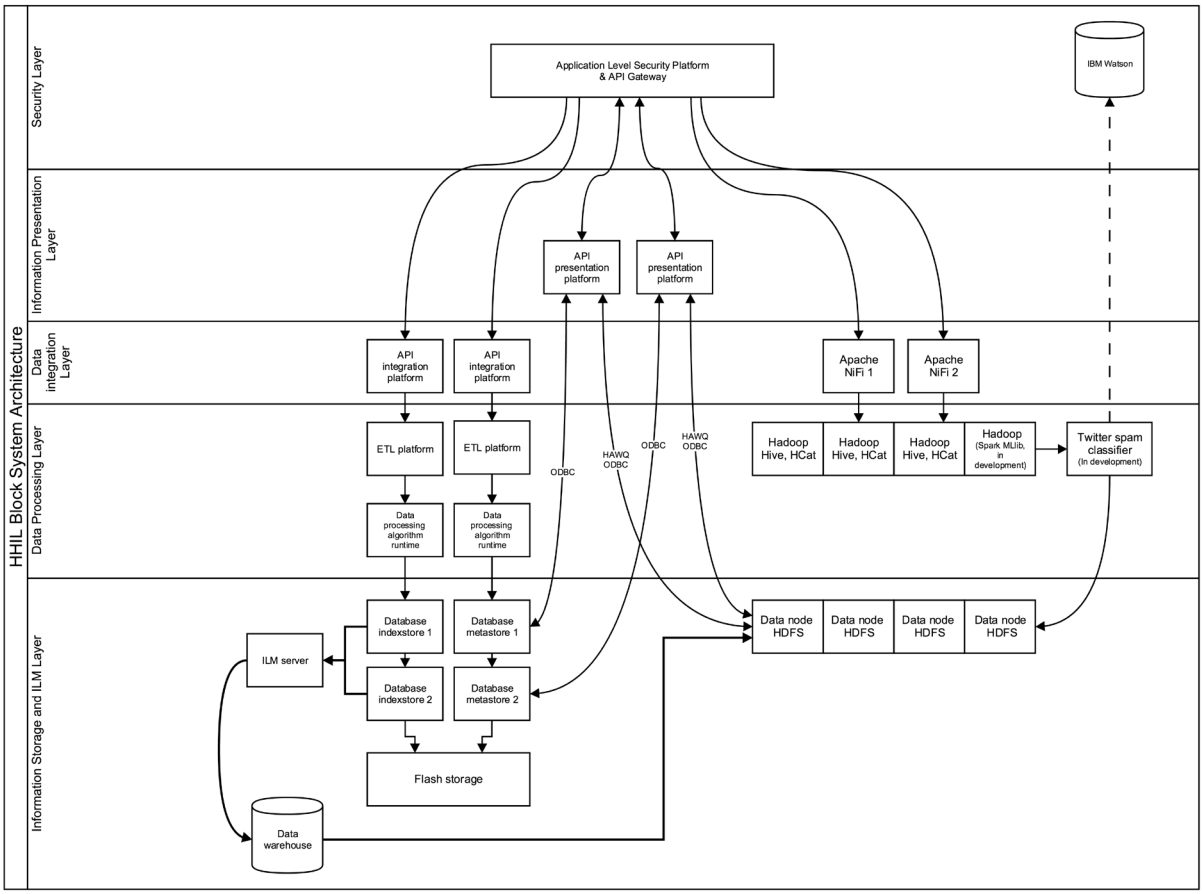

Figure 1 - Block architecture of the HHIL platform

Security

When developing a Service Oriented Architecture (SOA), it is important to identify and understand the unique security challenges you will face and be able to determine the difference between these and more common web or internal data security systems.

This section will focus on the external security factors and not the data at rest and data access control factors: these should be taken into account for a production system. The *How Happy Is London?* architecture focuses on mitigating three areas; frequency of requests and responses, variety of requestors and volume of subsequent data after requestor has been authenticated.

These map closely to the three Vs of big data Volume, Variety and Velocity.

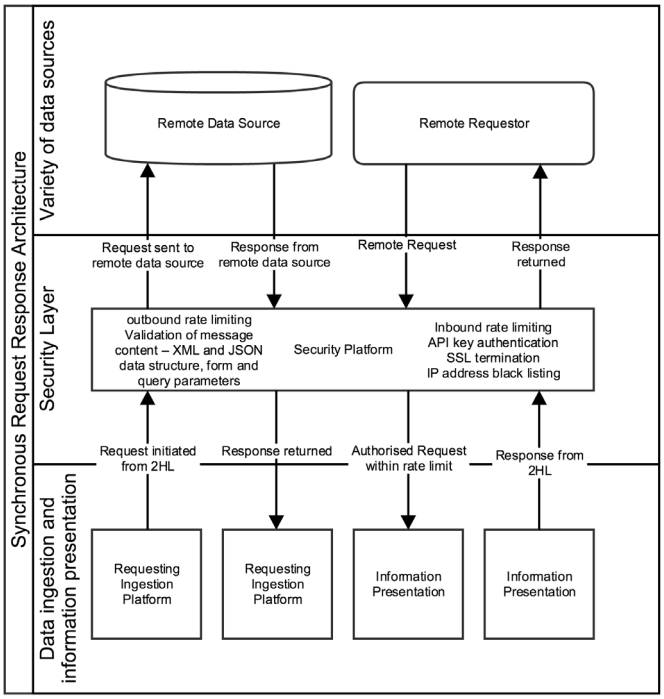

Figure 2 - HHIL's synchronous request response security architecture

Publicly accessible APIs, database links or application protocols all have very different security requirements. In *How Happy Is London?* we use publicly available REST APIs for both inbound and outbound data. This allowed HHIL to focus on a security layer that was appropriate to its needs. API rate limiting was implemented based on an API key to allow inbound user identification and limit the quantity of query responses for any given time period. Outbound query rate limiting was also introduced to ensure our systems didn't cause an unwanted denial of service attack on our data source providers.

As previously mentioned, the project uses REST APIs for information presentation. This helped us to focus on access, authentication and audit technologies that were appropriate to our needs. Based on the type of information being presented, *How Happy Is London?* implements SSL encryption and an API key based access and authentication using a 64-character alphanumeric string along with automated key blacklisting and IP address blacklisting. Other technologies reviewed were:

- Basic auth
- OAuth 1.0
- OAuth 2.0
- IP Address whitelisting
- Geographic blacklisting

These were not chosen, despite offering greater security, as they were thought to be over kill for version 1.

Request size limiting was implemented via the key rate limiting and is currently set at 1,440 requests per day or 1 per minute per API key.

Query response size limiting is implemented using query request inspection in the API engine. This limits what could be a critical vulnerability, whereby users of the API could request years of results at any one time for multiple data sets, putting strain on the database servers and lowering the response SLA to other users. Currently we have limited the request sizes to a 1-month range of historiographic data. This will return 4,460 as an upper maximum of data points.

Data ingestion

Data ingestion, with the ability to quickly integrate many varied data sources, is critical success factor for Big Data to break out of the confines of what's supported inside a single layer and start to access what's available. It doesn't matter whether that is online services, FTP sites, databases or good old-fashioned flat files.

To this end, an SOA integration platform was required. This once again links back to the three Vs of Big Data; Volume, Variety and Velocity.

The design approach to the ingestion layer for volume and velocity could be regarded as a compute or node-sizing factor, but to truly understand the best solution it's important to understand the requirements of the data you are trying to ingest and whether that data is best ingested via a single request-response for a specific resource or via long-lived connections with a low latency, sending new information as it occurs, such as streaming APIs 2.

Both have their pros and cons and were designed for different use cases. The streaming API was designed predominantly to run uninterrupted, capturing everything from a continuous stream of data in real time. The REST API was designed to take a number of requests and to perform a number of tasks, ranging from querying data, adding data, putting data, editing data or deleting data.

One of the key advantages of receiving a stream with the streaming API rather than with the REST API is that the Streaming API receives all its data in near real time.

The architecture of near real-time systems grows because processing is done in stream, which increases the management and complexity overheads. With this in mind, the majority of data sources are based on a RESTful request response architecture.

A non-functional requirement was the ability to allow integration to become a line of business (LOB) task by utilising codeless drag-and-drop systems. Creating a complex code-heavy data ingestion layer will lead to limited adoption and business units' time to productivity being hampered by the bottleneck this will become. This allows integration systems to be LOB friendly and empower LOB to make more interesting discoveries in the data, more quickly and with lower overhead on the IT function. To this end, self-service is also a key non-functional feature, creating a number of simple-to-use, quick-start templates.

Codeless integration building blocks won't cover 100% of all use cases: therefore some lifting may need to be undertaken to work with obscure sources or transformations. This is often a JavaScript custom function block that allows you to create reusable code segments to deal with the unexpected.

How Happy Is London? utilises a codeless environment, leveraging a simple left-to-right, block-based visual development environment for ingestion of structured data sources. See figure 3.

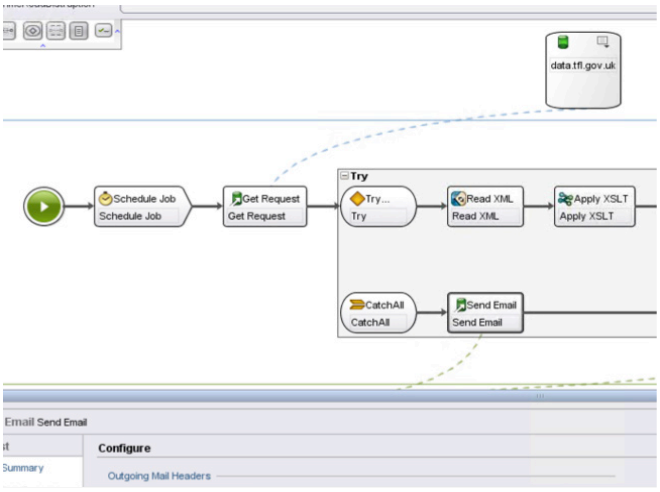

Figure 3 - showing data ingestion process and try catch error control

The project uses mostly REST-based APIs, allowing a very diverse data set to be ingested quickly to determine its value to our happiness equations.

However, for near real-time data needs (Twitter tweet sentiment analysis), *How Happy Is London?* had to embrace a different type of integration system. It uses Hadoop as its ETL platform for semi-structured data. The aim was to introduce a system with the correct feature set to address large volumes of streaming data that needed large data interrogation requirements.

For ingesting data into Hadoop, we initially looked at Apache Flume. Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating and moving large amounts of data. However, as the project matured, a more feature-rich integration platform was required. This is when NiFi was introduced. NiFi provides a reliable, scalable, manageable and accountable platform for developers and technical staff to create and evolve powerful data flows.

Such a system is useful in many contexts, including large-scale enterprise integration, interaction with cloud services and frameworks, business-to-business, intra-departmental and inter-departmental flows. NiFi fits well within the Apache Software Foundation (ASF) family as it depends on numerous ASF projects and integrates with several others. The NiFi project graduated in July 2015 to a top-level Project Apache project, demonstrating a level of maturity which was appropriate for this implementation. NiFi also meets the non-functional requirement to allow integration to become a line of business task by utilising codeless drag-and-drop development environment.

A structured data ingestion model

The structured data processing model is used specifically to process structured data and is designed to suit Representational State Transfer (REST) Application Programming Interfaces (APIs) which are stateless, client-server, cacheable communications protocols in which the HTTP protocol is used. REST APIs power a large majority of the world's networked applications, closely followed by the Simple Object Access Protocol (SOAP), for which this methodology would also be applicable.

The internet is awash with APIs. They represent every type of data imaginable, from weather and economic to knowledge and gambling.

Large-scale directories have grown over the last few years to represent APIs. One such directory (Programmable web7) has listed over 13,600 available APIs, some free to use and some commercially modelled.

We will be using four openly available APIs during the phase 1 release.

Structured data APIs

Structured data refers to information with a high degree of organisation, such that inclusion in a relational database is seamless and readily searchable by simple, straightforward search algorithms or other search operations.

This definition also includes a limited or predefined variation in the key value pairs returned. This allows for limiting of processing performed per data point and therefore processes more results per second.

Structured data APIs used in the project

Transport for London (TfL) Unified API

TfL's unified API brings together data across all modes of transport into a single RESTful API. This API provides access to the most highly requested real-time and status information across all the modes of transport, in a single and consistent way, including:

- Tube disruption
- Bus disruption
- Road disruption

This API allows *How Happy Is London?* to query a list of current disruptions for all given corridors: this includes the polygon of the disrupted area, along with a textural description and severity assessment. Future disruption information is also available through the API: this includes planned works and modernisation plans.

National Rail

National Rail Enquiries (NRE) supports the principle of transparency and contributes to the wider industry agenda by making data openly available in the public domain.

NRE has a selection of XML API feeds that are available for use by third-party developers to create their own applications. The data feeds are derived from three primary engines: Darwin, KB and OJP. *How Happy Is London?* leverages National Rail's API to provide real-time structured data on overground rail service disruption.

The Met Office

DataPoint is the Met Office service to access freely available Met Office data feeds in a format that is suitable for application developers. It is aimed at anyone looking to re-use Met Office data within their own innovative applications - professionals, the scientific community, student and amateur developers.

In *How Happy Is London?*, DataPoint is used for structured data on London's weather, providing real-time data on:

- Temperature
- Weather type
- Precipitation potential (Chance of rain)

Semi-structured data ingest model used in the project

Semi-structured data is a form of structured data that does conform to the formal structure. The data model associates with relational a database via tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data, but nonetheless contains fields that have no structure to them.

Semi-structured and unstructured data is often misinterpreted. To ensure completeness, unstructured data refers to information that either does not have a pre-defined data model or is not organised in a pre-defined manner. Unstructured information is typically seen in books, journals, documents, audio, audio and video content, analogue data and images.

Semi-structured APIs

Twitter

Twitter's streaming APIs give developers low latency-access to Twitter's global stream of tweet data. A proper implementation of a streaming client will be pushed messages indicating tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

For the sentiment of London, we use Twitter's streaming API with the search terms that relate to London. This provides human curated, near real-time feed on events, comments and the overall mood of London.

Data processing

The data processing layer in a Big Data system varies widely between use cases but it is commonly a pre-storage extract, transform and load (ETL) stage. This is designed to refine the data from its raw sources into an appropriate structured data or information. Tooling for this layer, as above in the data ingestion layer, is best in the hands of line of business and often goes hand in hand with the data ingestion layer.

Functional requirements for the processing layer include having a wide range of data manipulation techniques and reusable templates for common tasks allowing LoB to define the business rules or data mappings required for their purposes. Meanwhile, non-functional requirements include ensuring that these are as simple as possible to use via drag-and-drop data mappings as shown in Figure 4.

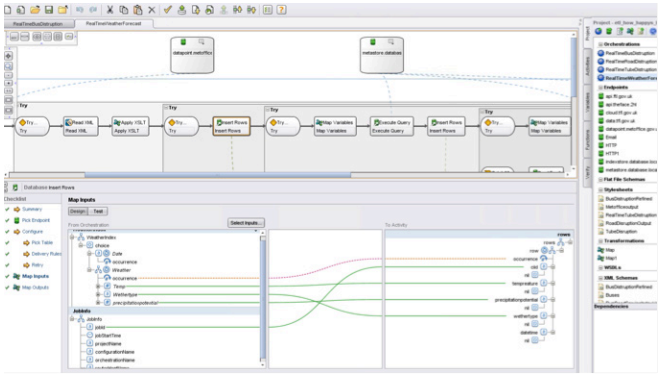

Figure 4 - Showing block-based development of data manipulation.

Once again, in practice codeless processing building blocks won't cover 100% of all use cases: therefore some heavy lifting may be needed to work with obscure transformations. This is often a JavaScript custom function block that allows you to create reusable code segments to deal with the unexpected.

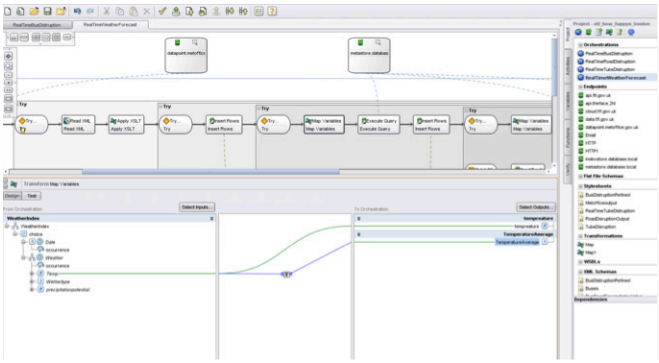

Figure 5 - Showing custom java script function

How Happy Is London? uses a combination of custom mathematical functions and includes functions to create a set of ETL processes that refine the raw data down to the happiness index.

Table 2 describes the ingestion rate and the subsequent data refinement of the various ETL processes that *How Happy Is London?* undertakes to create the happiness index.

Per Second Frequency	Data points remaining
Raw ingest	30340
Inline Filtration and Map Reduce	1368
Indexation	9
Resultant Single index	1

Table 2 - Showing the data refinement rate during different ETL processes

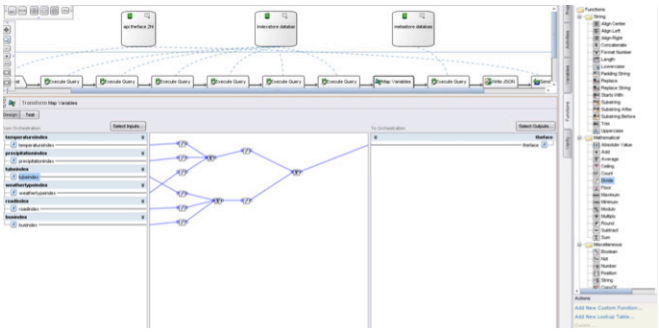

Figure 6 - Showing custom mathematical functions applying the weighting algorithm during a variable map

Structured data flow

The project's structured data processing model as shown in figure 7 follows the typical data processing model as detailed in table 1 for structured data.

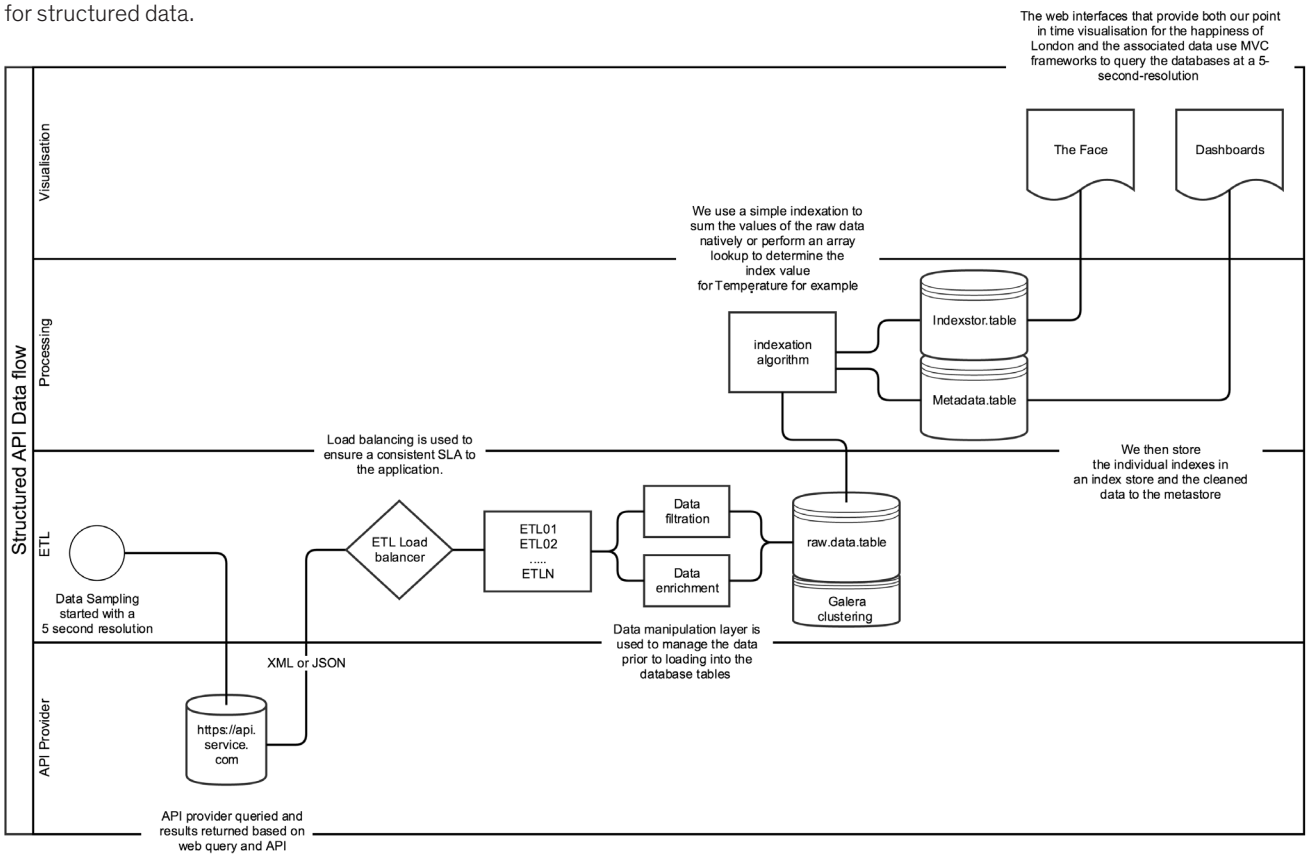

Figure 7 - HHIL's Structured Data Flow

Points to note from figure 7

The start point is time activated and limited to a 5-second resolution by the upstream processing overhead and subsequently the downstream load placed on individual web services.

ETL load balancing is used to ensure the SLA of writing data to the storage subsystem is maintained so the resolution of the data is always consistent.

In practice, we discovered that the data filtration and enrichment can cause an undesired random variation in overhead when the system is running at full sampling rate, as these processes may rely on external systems over which *How Happy Is London?* has no control. This problem was solved by simply running multiple ETL processes asynchronously and then distributing the load across a calculated number ETL systems. A number of different systems could have been implemented for this purpose, but due to the nature of the integration system and the data sources, load balancers were chosen. This also introduces a level of ETL service availability.

Data filtration refers to the removal of irrelevant metadata, replicated data and obsolete data prior to storing to database. This process is eloquently described as the "burn the hay" methodology by CERN's Data wizards15. It allows the system to store only what we need, but the trade off is that the data pre-processing overhead is higher.

Other data requires enrichment, whereby data from external or internal sources is added to the data points in transit. This could be as simple as a timestamp or as complex as utilising external services to augment the data to provide future insight prior to storage.

In phase 1 of the project, we are using MySQL databases with Galera clustering software to provide synchronisation between our databases. Galera is implemented to underpin the resolution SLA guaranteed to the application.

The indexation algorithm is the same for both structured and semi-structured data workflows and will be described in the data processing area of this paper.

Semi-structured data flow

HHIL's semi-structured data processing model as shown in figure 8 follows the data processing model.

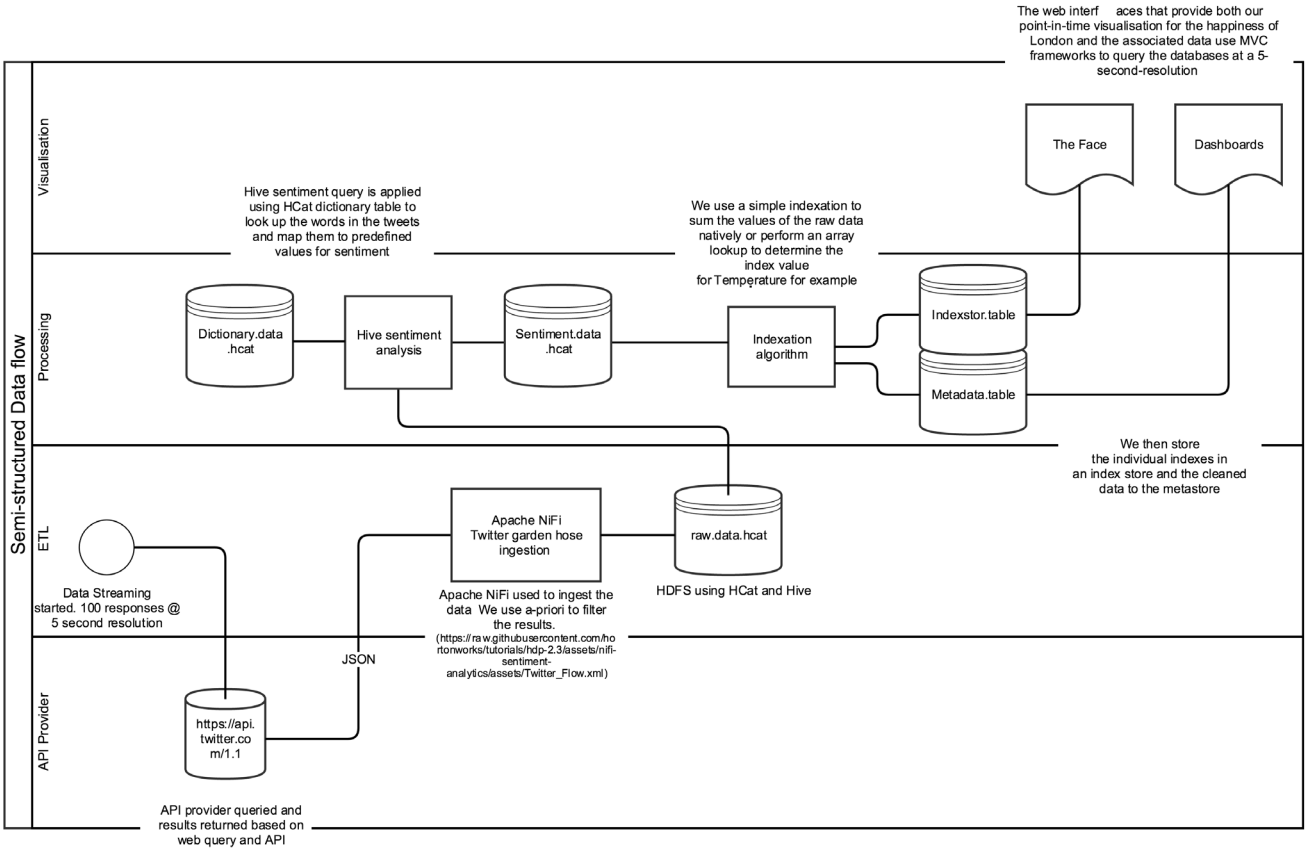

Figure 8 - HHIL's Semi-structured Data Flow

Points to note from figure 8

The start point for Twitter is used once to open the stream and then maintain the HTTPS connection between Twitter and Apache NiFi.

NiFi provides a reliable, scalable, manageable and accountable platform for developers and technical staff to create and evolve powerful data flows. Such a system is useful in many contexts including: large-scale enterprise integration, interaction with cloud services and frameworks, business-to-business, intra-departmental, and inter-departmental flows. NiFi fits well within the Apache Software Foundation (ASF) family as it depends on numerous ASF projects and integrates with several others.

This data is written to HDFS in an HCatalog (HCat) table. HCat is a table and storage management layer for Hadoop that enables users with different data processing tools — Pig, MapReduce, Hive — to more easily read and write data on the grid. HCatalog's table abstraction presents users with a relational view of data in the Hadoop distributed file system (HDFS) and ensures that users need not worry about where or in what format their data is stored. As Twitter data has a schema, this allows for easier querying later in the workflow.

How the sentiment query works for HHIL

For phase 1 of the project, we use a very basic sentiment analysis algorithm.

Hive is used to analyse the tweet bodies for sentiment. Initially developed by Facebook, Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarisation, query, and analysis.

In this query, we use the raw tweet bodies and then a second HCat table that's used for dictionary lookup. Each word is then compared to a sentiment lexicon to test if it appears. If it does appear, it then has a sentiment value associated with it that is then used to contribute to the overall index of sentiment for London.

Example below:

Tweet Body Twitter Sentiment Analysis Process

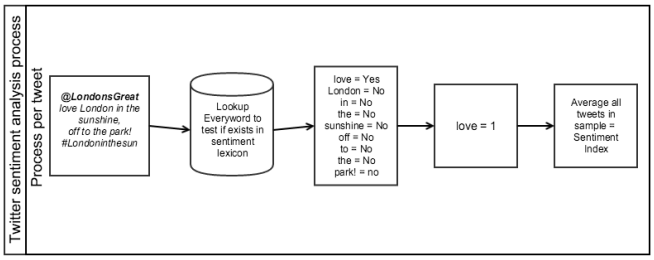

Figure 9 - Block diagram of sentiment query

In the above example, the word 'love' would be identified as being in the sentiment lexicon and rated based on its predetermined value.

Example key value dictionary HCat table

word	index
loathe	0.2
Love	1
Lust	0.9

Table 2 - Showing lexicon scoring table in HCat

Once all of the tweets have been analysed, their keywords looked up and their values averaged, the remaining value is then the index for Twitter sentiment for that period in time.

Data storage

Data type from table 1 plays a critical part in determining the data storage layer's functional requirements.

Structured data works well when stored in a traditional relational database as its predetermined structure can fit to a data schema well. Typically, this type of database is an OLTP (OnLine Transaction Processing) database.

However, a transactional database doesn't lend itself well to analytics, due to the speed at which data needs to be written and interrogated. To effectively perform structured analytics, you need a data warehouse.

A data warehouse is a database of a different kind: an OLAP (OnLine Analytical Processing) database. A data warehouse exists as a layer on top of another database or databases, usually OLTP databases. The data warehouse takes the data from all of these databases and creates a layer optimised for and dedicated to analytics. This is an important point to take into account when designing a Big Data solution.

Other considerations to take into account are that enterprise data warehouses are expensive and can add a layer of complexity to an analytics architecture. Making sure your applications will take full advantage of an EDW is important to ensure the solution isn't over-engineered.

For structured data processing, *How Happy Is London?* uses MySQL. This decision was made due to the ease of implementation and the relatively low number of data points being written to disk, combined with the predictability of queries.

To improve performance, further query caching is enabled. *How Happy Is London?* implements two virtual database servers: one for indexes, which is used by the presentation APIs, and a second for metadata, which is used to create the indexes from the ingested API data. This approach, combined with use of indexed columns and writing SQL queries to take advantage of these columns, gives the required balanced read / write performance. Further performance will be gained by horizontally scaling the MySQL platform, whether based on specific workload or using Galera Clustering.

As the complexity of the system increases and if the use case of the system changes, an EDW will be implemented to provide query flexibility while maintaining the lowest latency results.

For semi-structured and unstructured data, as well as data that is streaming in nature, another approach is required. Solutions include Not Only SQL (NoSQL) databases, Object stores such as Hadoop Distributed File System (HDFS) and in-memory databases.

The benefit of these database architectures is that they can scale transactions and analytics linearly across nodes while maintaining data consistency and with high availability. Their distributed nature means that tasks can be spread across many different nodes using an active-active data access mechanism. This allows for greater processing throughput on tasks that require intensive data interrogation.

The project uses HDFS for its semi-structured storage stage. This decision was made based on two factors. The data processing layer was Hive, which is part of the Hadoop ecosystem, and HDFS is a core element. Its ability to work in unison with Yet Another Resource Schedular (YARN) and HDFS's parallel data access means Hive can distribute tasks over the entire Hadoop cluster, helping to keep the processing time within the SLA for computing the happiness index.

Data presentation

As the last stage of the data analytics architecture, it is often assumed that data presentation is the last to be designed. However, data presentation is the business outcome and in a system that is nearly always delivering a business outcome, this stage is the most important.

Understanding what the data presentation layer must do will help to determine the downstream architecture. Examples of data presentation are:

- Dashboards used to aid decision making processes
- Self-service data exploration tools to help users explore data
- APIs to interface with external systems
- Automated outcomes or feeding the data into other analytics models for further or future processing

Once the outcome has been determined, an analytics project would need to work out what data was needed to deliver the outcome, and how it will be processed into information and then stored ready for the presentation stage.

How Happy Is London? uses an API as its presentation layer. This decision was taken to enable flexibility and reach from the system. The API has been designed to feed our website at HowHappyIs.London it has also been designed to support our developer community at: api.howhappys.london. Our API is openly available and designed to be used under an open licence GNU General Public Licence v321.

How Happy Is London? API Spec

The happiness of London project presents its data via Open RESTful APIs. These APIs are used to access the indexes for either the point-in-time values or a range of values based on a URL encoded time start and time end stamp.

Authorisation

An API key is required to access the API. This must be presented as an API key header.

apiKey: jd1384y13ruc93480ry3498ry329t8y2

Request size limiting was implemented via the key rate limiting and is currently set at 1,440 requests per day or 1 per minute per API key. Query limiting is implemented using query request inspection in the API engine. Currently we have limited the request sizes to one month of histographic data. This will return 44,640 data points.

All API calls are GET requests.

API calls

Happiness of London now

Request

curl --header "apiKey: jd1384y13ruc93480ry3498ry329t8y2" http://api.arrowdemo.center/v1/theface/now

Response

{"value":"indifferent","index":"0.530952380952381"}

Happiness of London over time

This API call requests for a range of indexes to represent the happiness of London over time The range of time is determined by time start 'ts' and time end 'te'. The time and data stamps are formatted as YYYY-MM-DD HH-MM-SS and URL encoded

Request

curl --header "apiKey: jd1384y13ruc93480ry-3498ry329t8y2" http://api.arrowdemo.center/v1/theface/range?ts=2015-10-23%2000%3A50%3A48&te=2015-10-23%2000%3A59%3A48

Response

{ "value": [{ "value": "happy", "index": "0.873886269070735", "datetime": "2015-10-23T00:50:48.000" }, { "value": "happy", "index": "0.873886269070735", "datetime": "2015-10-23T00:51:48.000" }, { "value": "happy", "index": "0.873886269070735", "datetime": "2015-10-23T00:52:48.000" }, { "value": "happy", "index": "0.873886269070735", "datetime": "2015-10-23T00:53:48.000" }, { "value": "happy", "index": "0.873886269070735", "datetime": "2015-10-23T00:54:48.000" }, { "value": "happy", "index": "0.8738815331010453", "datetime": "2015-10-23T00:55:48.000" }, { "value": "happy", "index": "0.8738815331010453", "datetime": "2015-10-23T00:56:48.000" }, { "value": "happy", "index": "0.8738815331010453", "datetime": "2015-10-23T00:57:48.000" }, { "value": "happy", "index": "0.8738815331010453", "datetime": "2015-10-23T00:58:48.000" }, { "value": "happy", "index": "0.8738815331010453", "datetime": "2015-10-23T00:59:48.000" }] }

Recalling individual indexes over time

As part of the API set, we have also exposed access to the individual indexes that go towards creating the overall happiness index. The range of time is determined by time start 'ts' and time end 'te'. The time and data stamps are formatted as YYYY-MM-DD HH-MM-SS and URL encoded The item required is determined by the i parameter in the request string. Currently we support:

Weather data courtesy of the Met Office
Current temperature in London determined by the i=temperature
Chance of rainfall determined by the i=precipitation
Predicted type of weather i=weathertype
Travel data courtesy of Transport for London (TfL)
Underground tube line status determined by the i=tube
Road congestion status determined by the i=road
Bus line status determined by the i=bus
Social data courtesy of Twitter
Social sentiment index of London determined by the i=social

Request

curl --header "apiKey: jd1384y13ruc93480ry-3498ry329t8y2" http://api.arrowdemo.center/v1/items/range?i=precipitation&ts=2016-2-23%2000%3A50%3A48&te=2016-2-23%2000%3A59%3A48

Response

{ “item”: [{ “index”: “0.97”, “datetime”: “2016-02-23T00:53:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:57:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:52:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:56:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:59:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:54:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:58:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:51:11.000” }, { “index”: “0.97”, “datetime”: “2016-02-23T00:55:11.000” }]}

Footnotes

http://www.forbes.com/sites/oreillymedia/2012/01/19/volume-velocity-variety-what-you-need-to-know-about-big-data/

For clarification, the term API is used in this context to denote any remote connectivity to any data source.

http://dev.datasift.com/blog/introduction-streaming-api

https://en.wikipedia.org/wiki/Apache_Flume

https://wiki.apache.org/incubator/NiFiProposal

http://incubator.apache.org/projects/nifi.html

http://www.programmableweb.com

http://www.brightplanet.com/2012/06/structured-vs-unstructured-data/

https://api-portal.tfl.gov.uk/docs

http://www.nationalrail.co.uk/46391.aspx

http://www.metoffice.gov.uk/datapoint

https://en.wikipedia.org/wiki/Semi-structured_data

https://en.wikipedia.org/wiki/Unstructured_data

https://dev.twitter.com/overview/api

http://www.theregister.co.uk/2015/03/23/cerns_atom_big_data_f500/

https://wiki.apache.org/incubator/NiFiProposal

https://cwiki.apache.org/confluence/display/Hive/HCatalog+UsingHCat

https://en.wikipedia.org/wiki/Apache_Hive

https://www.healthcatalyst.com/database-vs-data-warehouse-a-comparative-review

http://galeracluster.com/

https://opensource.org/licenses/GPL-3.0

References

Scalable sentiment analytics
http://online.journals.tubitak.gov.tr/openAcceptedDocument.htm?fileID=405808&no=86444

Sentiment strength detection in short informal text
Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai
www.scit.wlv.ac.uk/~cm1993/papers/SentiStrengthPreprint.doc

Knowledge vault: A web-scale approach to probabilistic knowledge fusion
http://www.cs.ubc.ca/~murphyk/Papers/kv-kdd14.pdf

DBpedia: A nucleus for a web of open data
Soren Auerl, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak and Zachary Ives
http://richard.cyganiak.de/2008/papers/dbpedia-iswc2007.pdf

Analyzing Twitter data with Hadoop
https://blog.cloudera.com/blog/2012/10/analyzing-twitter-data-with-hadoop-part-2-gathering-data-with-flume/

TensorFlow: Large-scale machine learning on heterogeneous distributed systems
Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, et al
http://download.tensorflow.org/paper/whitepaper2015.pdf

Data integration platforms for Big Data and the enterprise
http://www.oracle.com/us/products/middleware/data-integration/di-oracle-informatica-ibm-wp-2438402.pdf

Architectural patterns for near real-time data processing with Apache Hadoop
http://blog.cloudera.com/blog/2015/06/architectural-patterns-for-near-real-time-data-processing-with-apache-hadoop/

Flume 1.6.0 user guide
https://flume.apache.org/FlumeUserGuide.html

Designing scalable data warehouse using MySQL
http://www.slideshare.net/vanuganti/designing-scalable-data-warehouse-using-mysql

Arrow Electronics, Inc.
Enterprise Computing Solutions
Nidderdale House
Beckwith Knowle
Otley Road
Harrogate
HG3 1SA

In Person

020 7786 3400
Call to talk or set up a face-to-face meeting with one of our knowledgeable representatives.

Via Email

HHIL.ecs.uk@arrow.com
Email us for answers to questions or to start a conversation.

Online

HowHappyIs.London