

Big Data ecosystem

Worldline

M. Fanilo Andrianasolo

UNIVERSITÉ
LUMIÈRE
LYON 2
UNIVERSITÉ DE LYON



Data Analytics Tech Lead & Product Manager Worldline

2013 - 2016

Big Data
Engineer

2016 - 2017

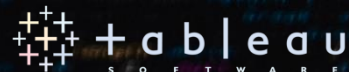
Data Analytics
Evangelist

2017-2019

Data Science
Tech Lead

2019-current

Data Analytics
Product Manager

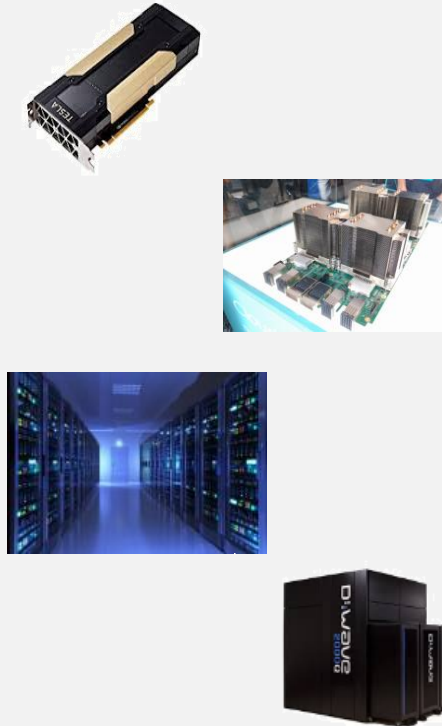


Data is at the center of all IT activities

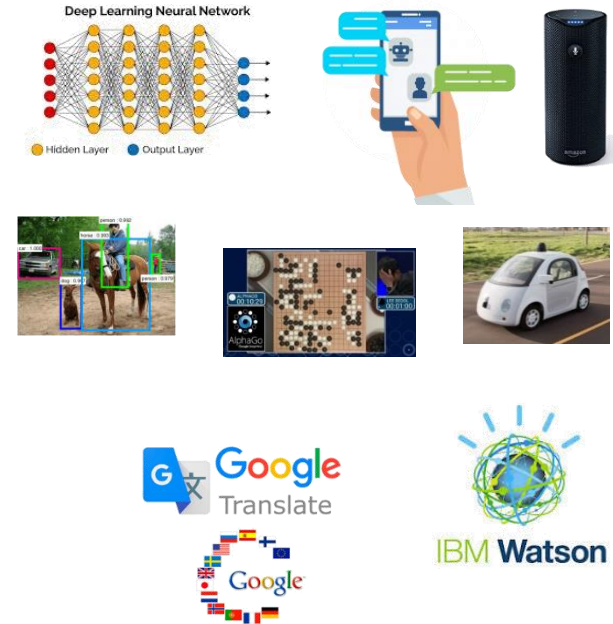
Data



Hardware

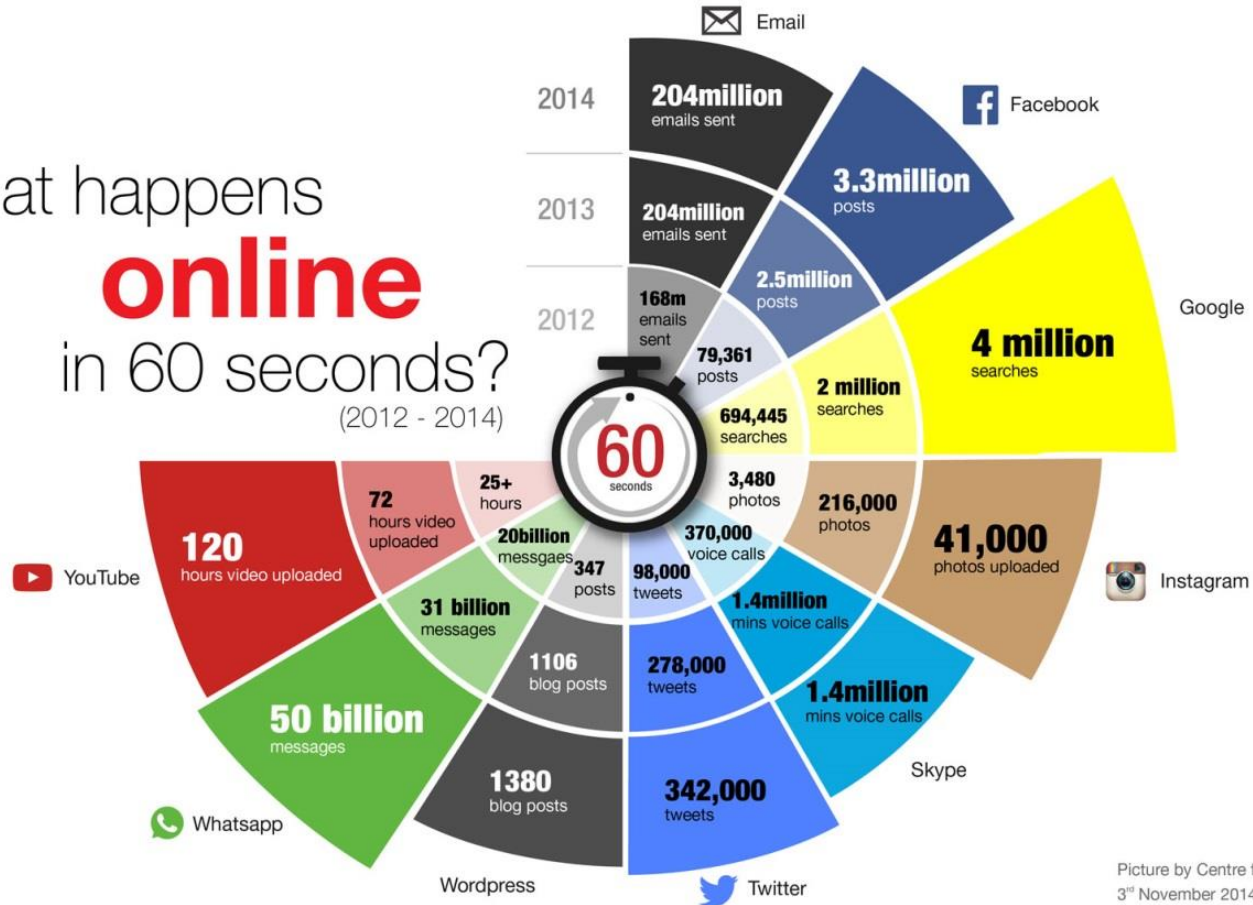


Innovations

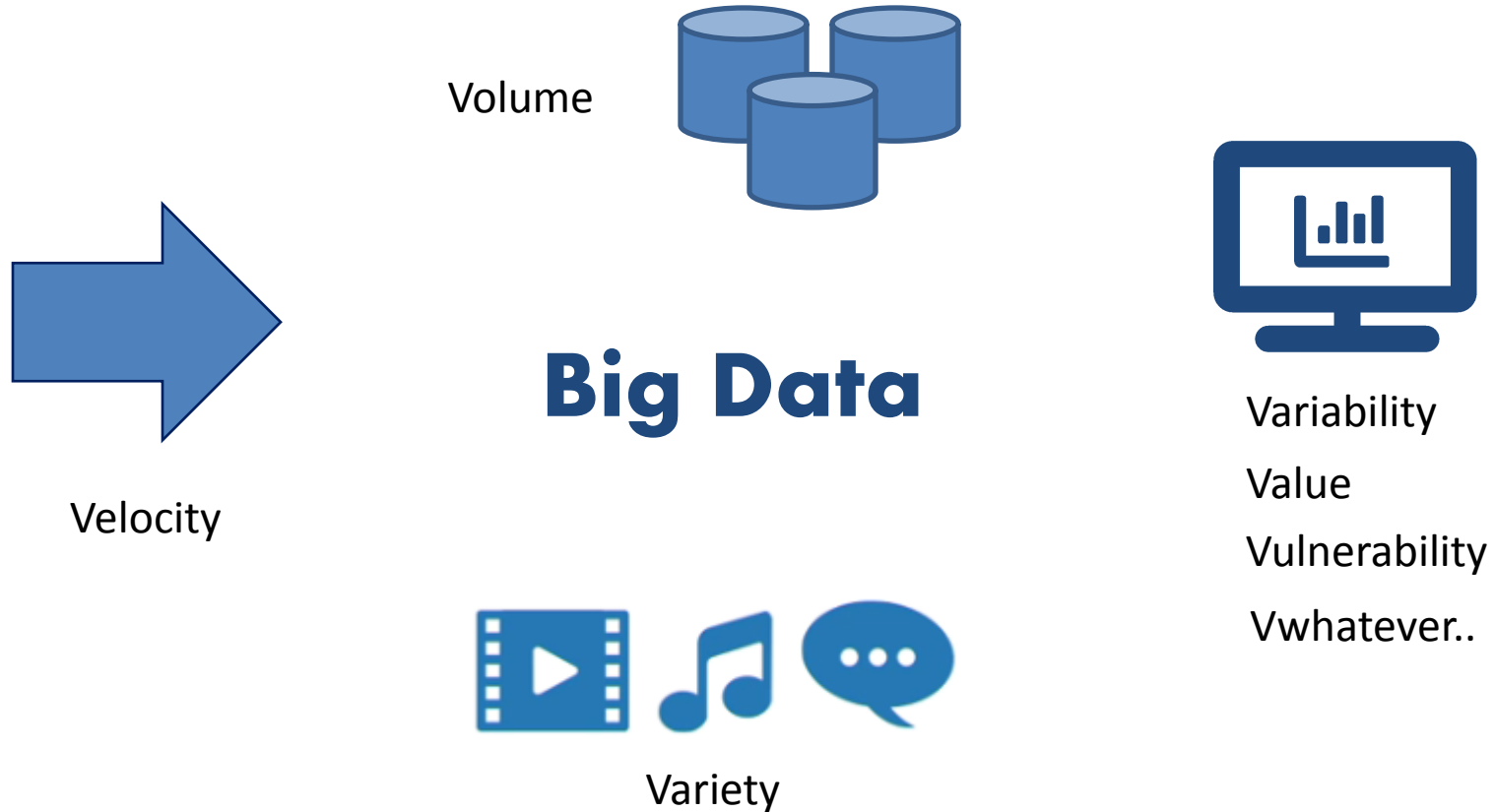


Explosion of data

What happens
online
in 60 seconds?
(2012 - 2014)



3Vs of Big Data



Problem

f Daily rate in 2014



How should we store and query such data ?

Scaling ?

Vertical scaling



Less power consumption, cooling costs

Less challenging to implement

Less licencing costs

(Sometimes) less network hardware

PRICE

Hardware failure causes bigger outages

Vendor lock-in

Limited upgradeability

Horizontal scaling



Much cheaper

Easier fault-tolerance

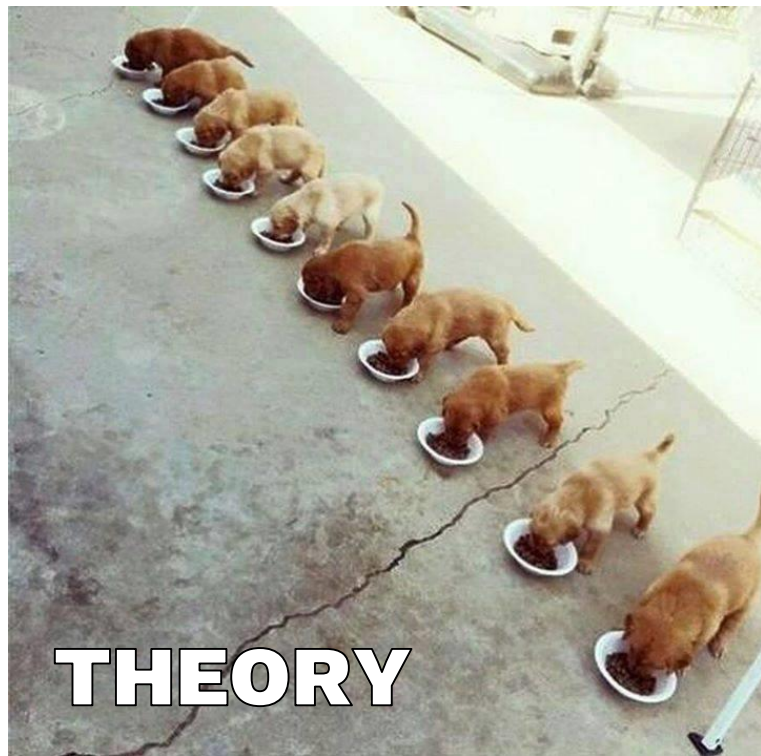
“Easier” upgrade by adding new machines

Bigger energy footprint

Higher utility cost (electricity, cooling)

More networking equipment

Scaling is hard



Big Data ecosystem



Apache Hadoop



Open-source software for reliable, scalable, distributed computing

Apache Hadoop ecosystem



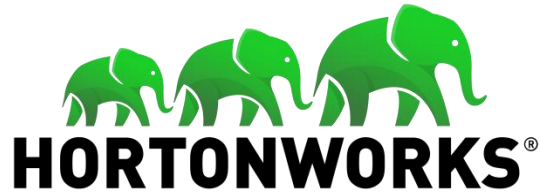
More than 30 open source projects for managing and analyzing Big Data



...

Hadoop distributions

CLOUDERA



Hadoop distributions vs Cloud providers



CLouDERA



 **Hewlett Packard**
Enterprise



Hadoop ecosystem use cases



Web indexing from web crawlers



Playlist generation from every listens



Log analysis



Product recommendation from purchases

A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing



Security



Orchestration

A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing

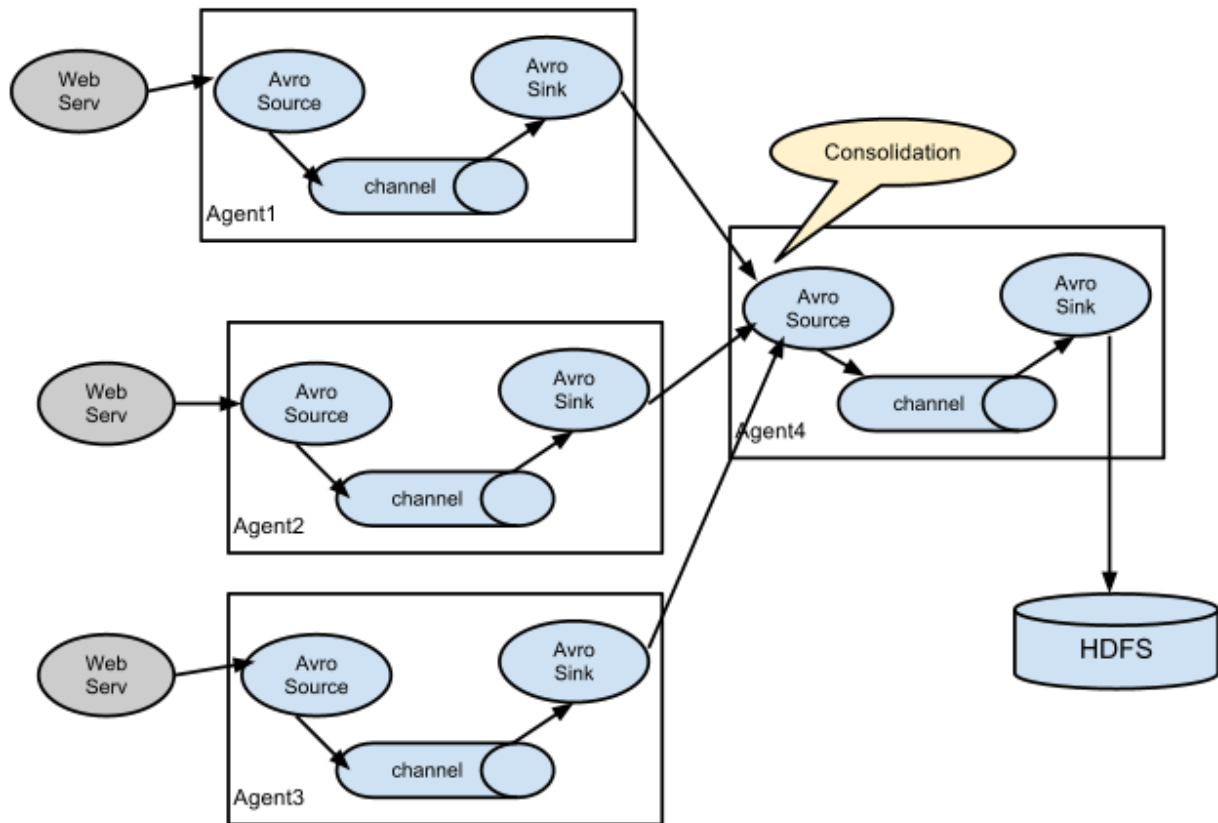


Security

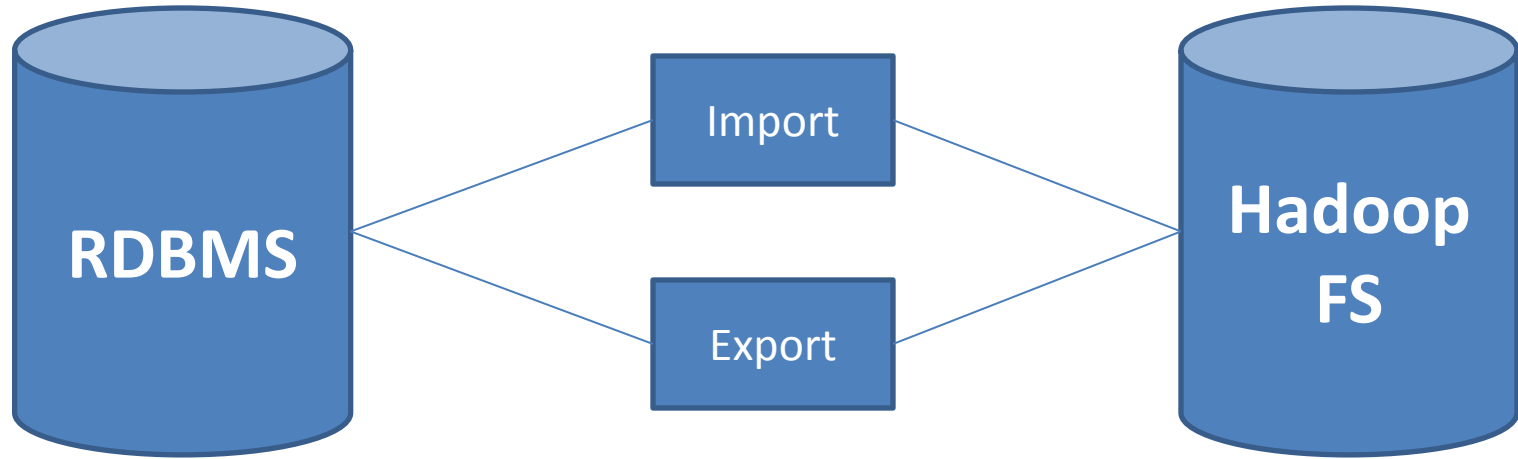


Orchestration

Acquisition



Acquisition



A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing

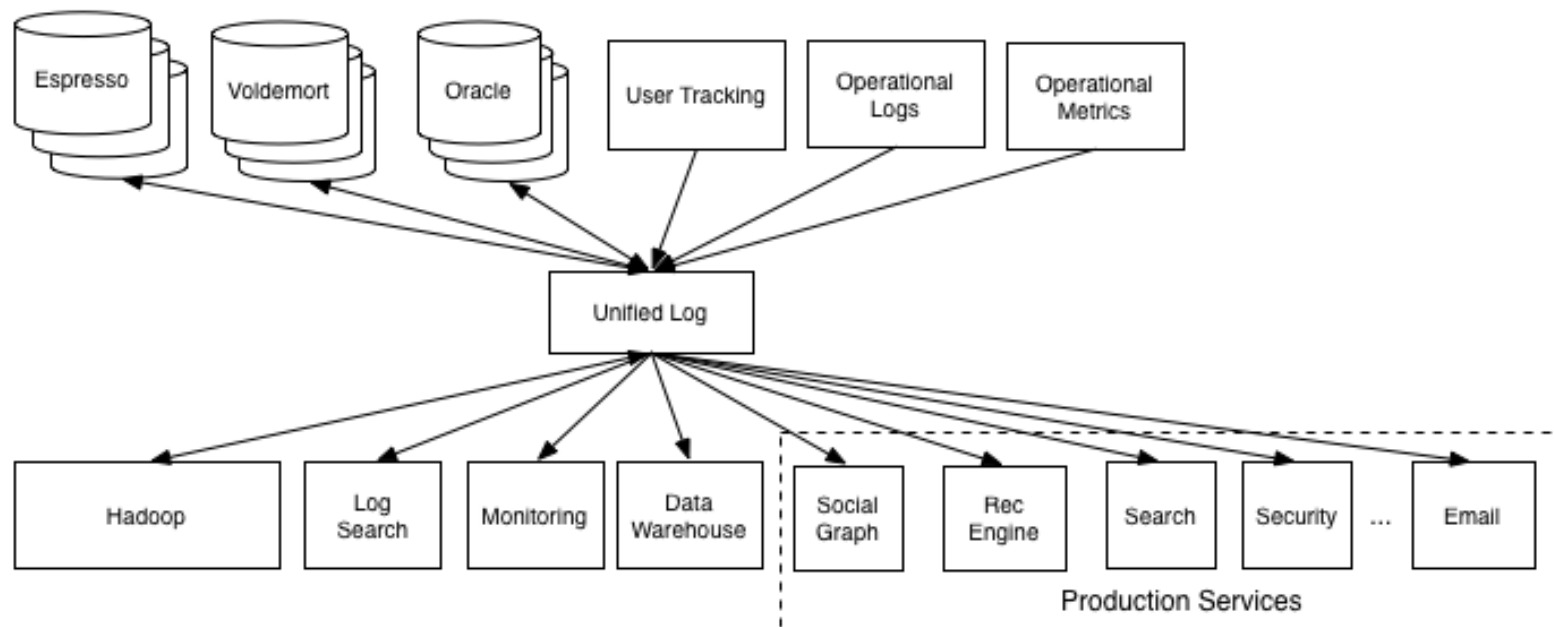


Security



Orchestration

Transport

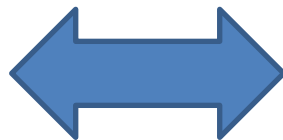


Transport



```
{  
  "type" : "record",  
  "namespace" : "test",  
  "name" : "Employee",  
  "fields" : [  
    { "name" : "Name" , "type" : "string" },  
    { "name" : "Age" , "type" : "int" }  
  ]  
}
```

.ascv



```
emp e1=new emp( );  
e1.setName("omar");  
e1.setAge(21);
```

.java

A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing

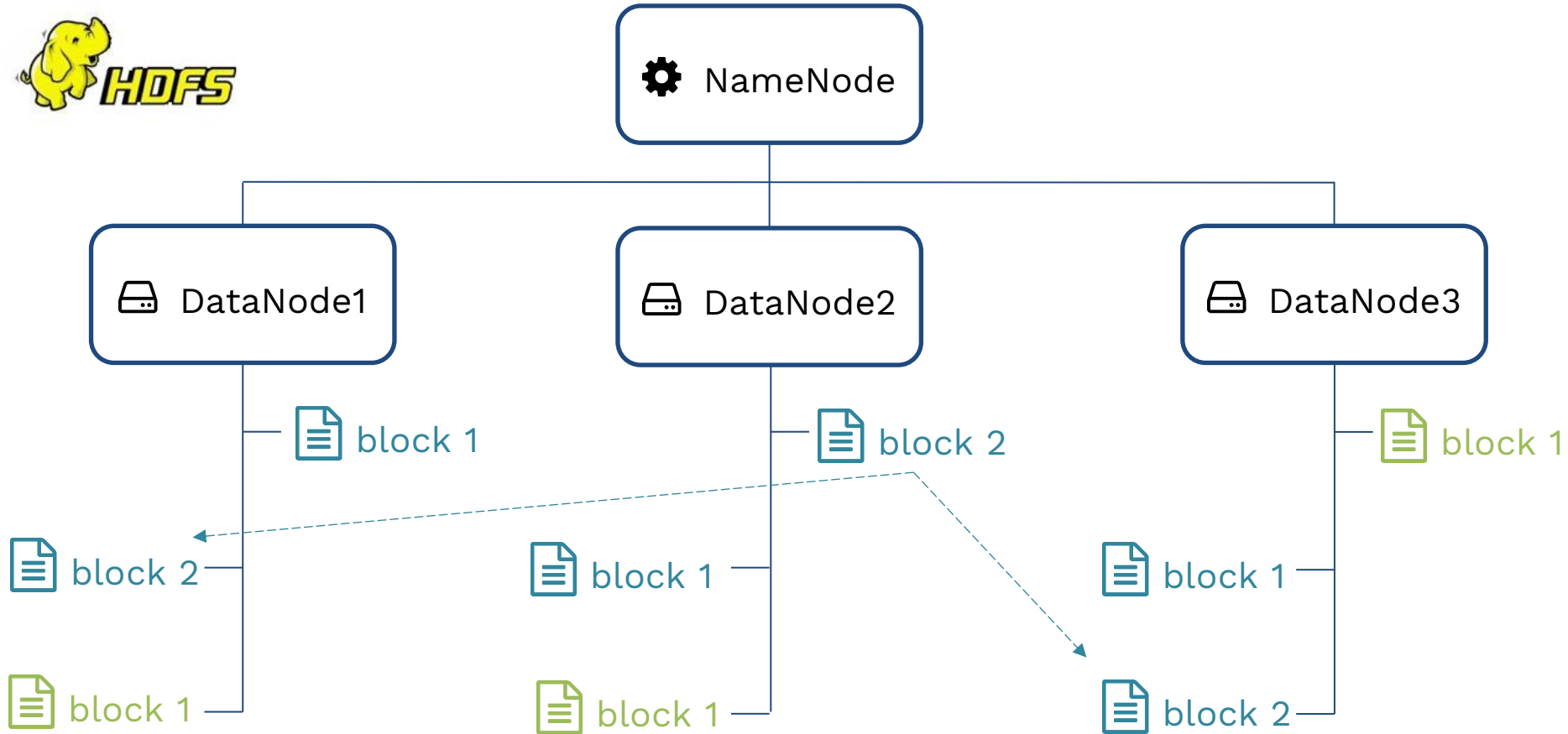


Security



Orchestration

Hadoop Distributed File System



HBase



Key	U:cookie	U:is_auth	U:has_t	P:Product1	P:Product2	P:Product3
1960:Fanilo	c13e	1				3
2001:Fanilo	c13e		1			
1990:Omar	d45				1	

A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing



Security



Orchestration

YARN – Yet Another Resource Negotiator

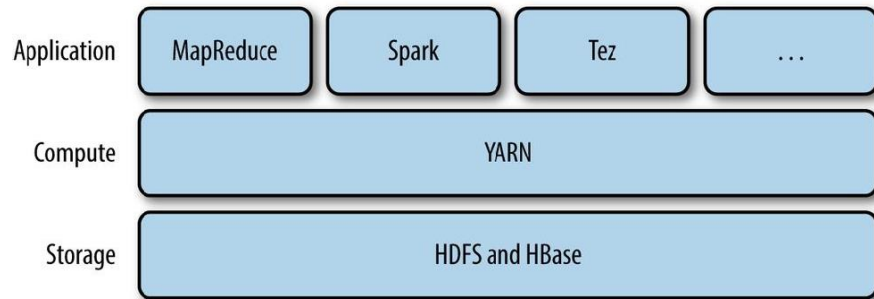
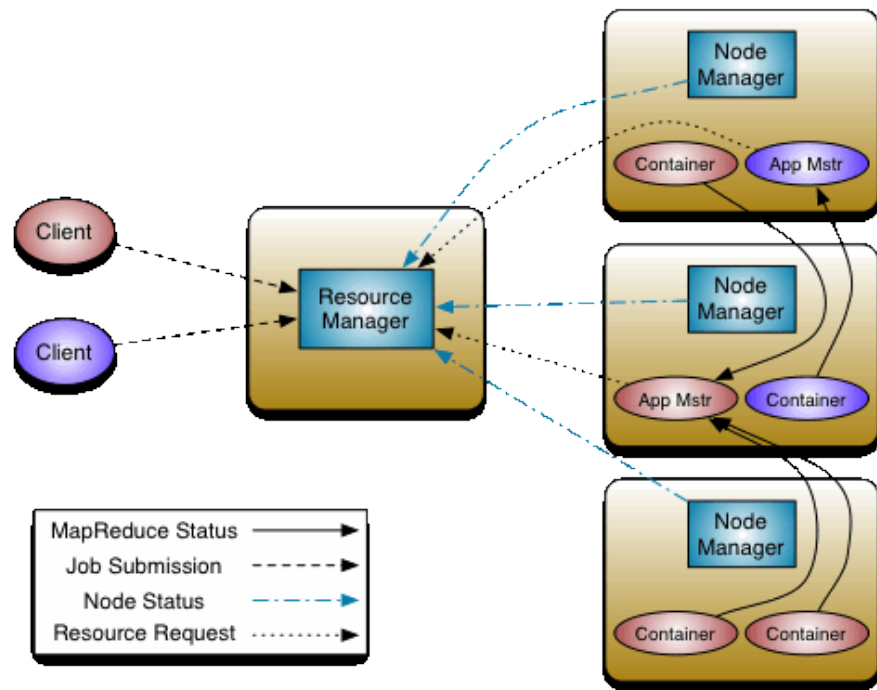
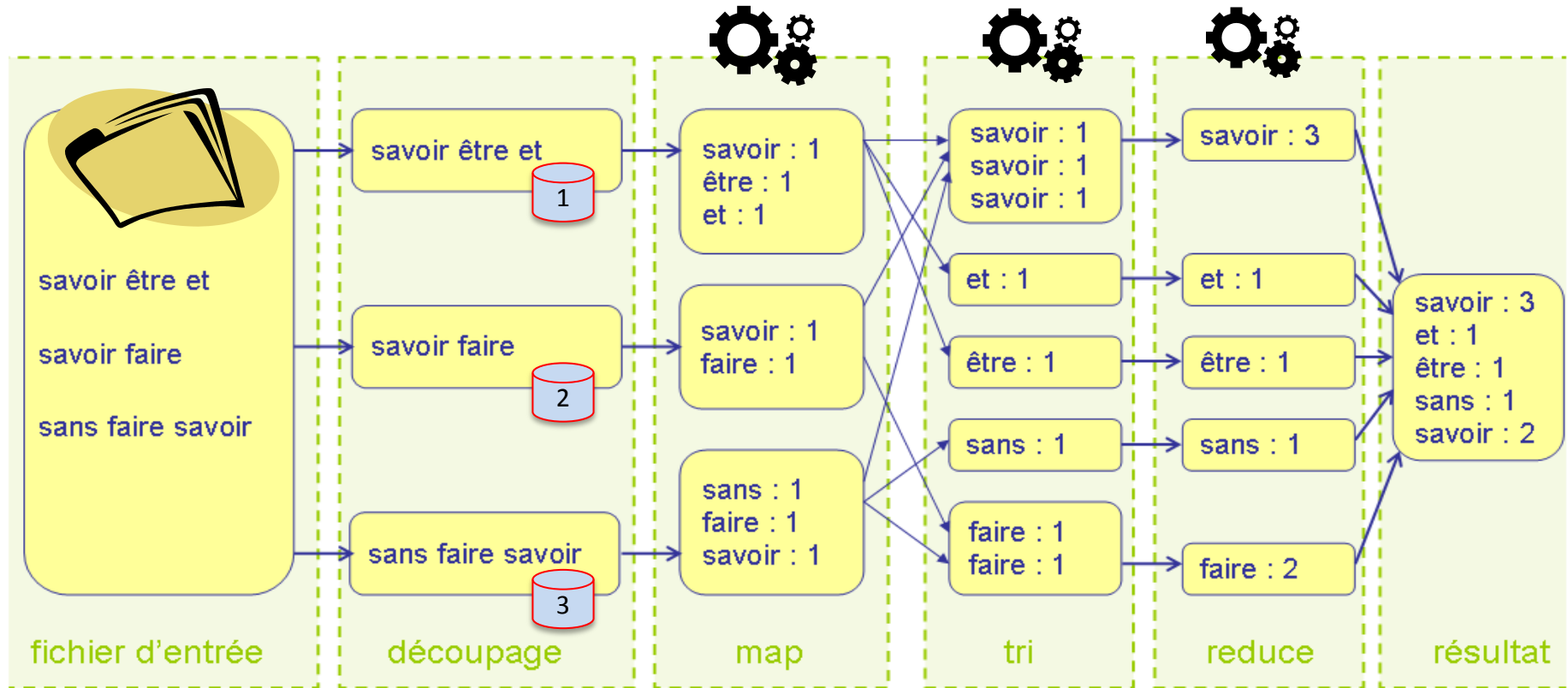


Figure 4-1. YARN applications

YARN hides the resource management details from the user to facilitate the management of parallel applications.

Batch processing - Map Reduce



Data locality : Moving Computation is Cheaper than Moving Data

Batch processing



HiveQL →

MapReduce



music_sales.csv

```
1, « Let it go », 4.99€, 5
2, « Snow », 7.99€, 1
3, « Lion King », 0.99€, 1
4, « SISE », 1.99€, 2
5, « Lyon is great », 2.99€, 3
```

Batch processing



```
recordings = LOAD '$file' USING PigStorage(',') AS  
              (id, price, artist, title,  
duration, year);  
limit = LIMIT recordings $size;  
DUMP limit;
```

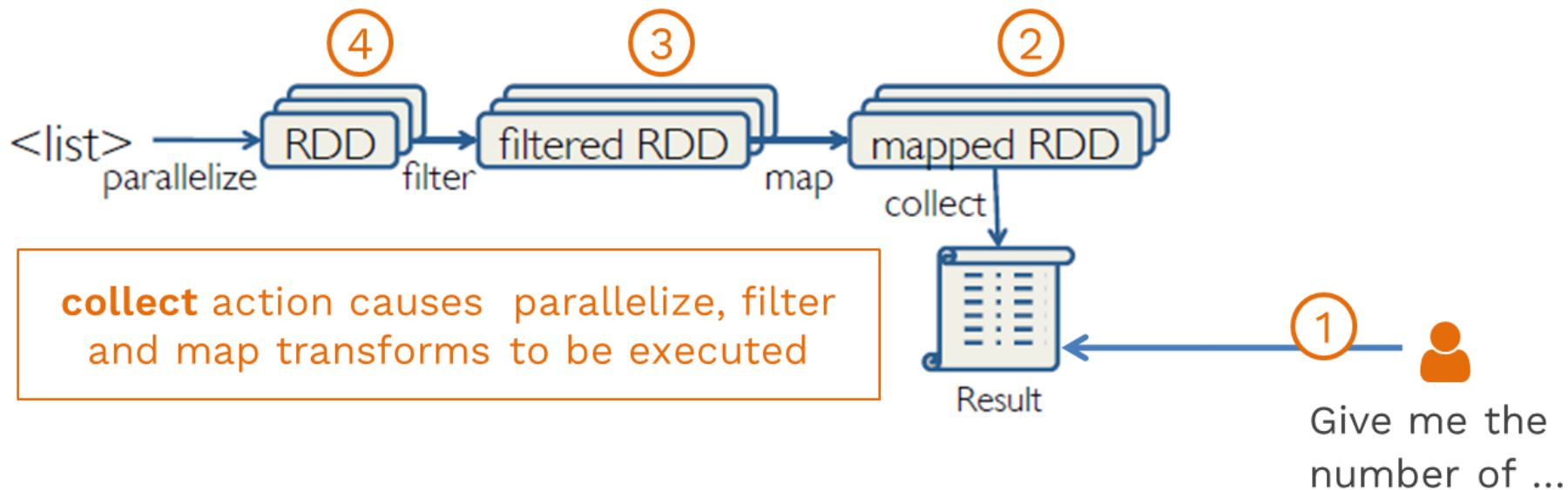
MapReduce



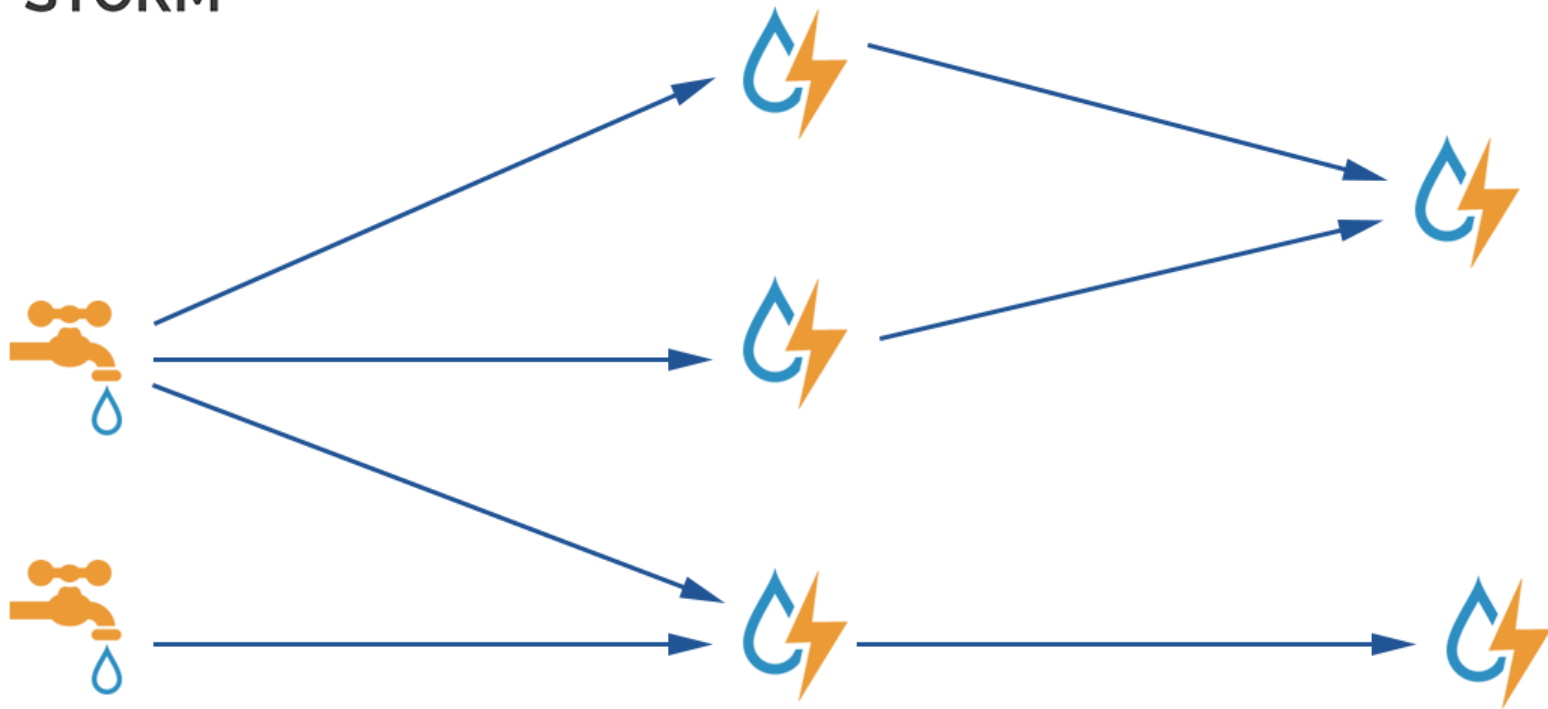
music_sales.csv

```
1, « Let it go », 4.99€, 5  
2, « Snow », 7.99€, 1  
3, « Lion King », 0.99€, 1  
4, « SISE », 1.99€, 2  
5, « Lyon is great », 2.99€, 3
```

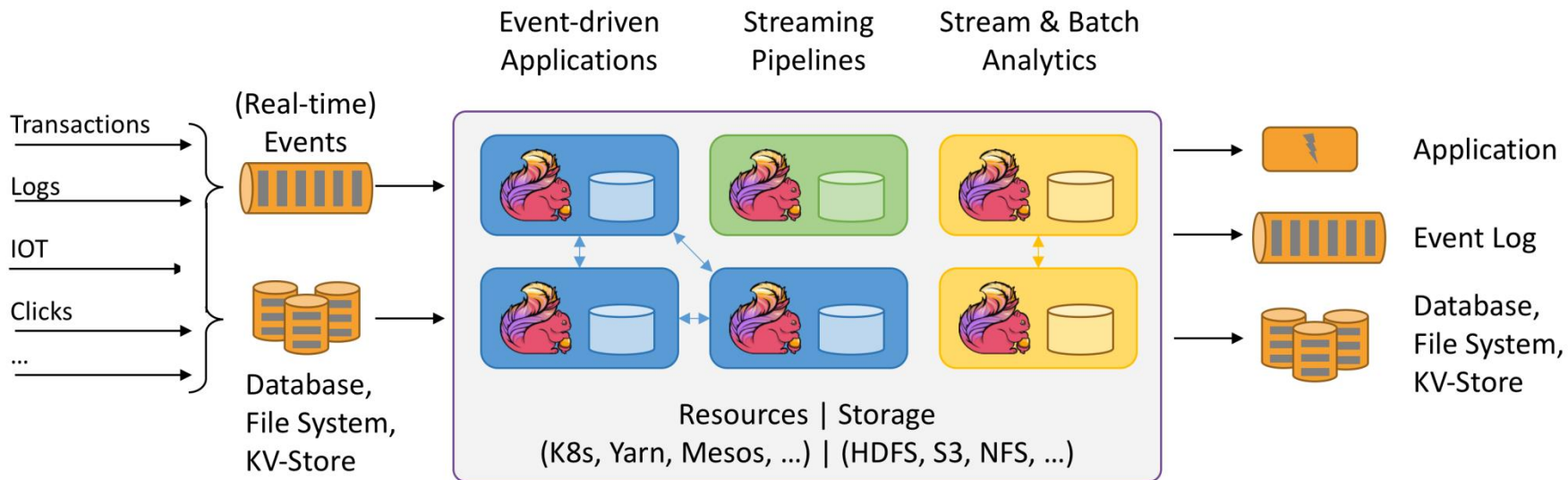
Batch processing



Realtime processing



Realtime processing



A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing



Security



Orchestration

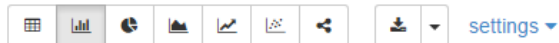
Visualizing



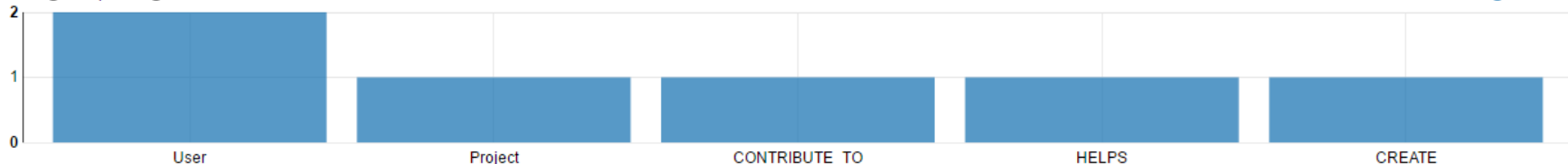
Apache Zeppelin

```
%spark
print(s"""
%network {
  "nodes": [{ "id": 1, "label": "User", "data": { "fullName": "Andrea Santurbano" } }, { "id": 2, "label": "User", "data": { "fullName": "Lee Moon Soo" } }, { "id": 3, "label": "Project", "data": { "name": "Zeppelin" } } ],
  "edges": [{ "source": 2, "target": 1, "id": 1, "label": "HELPS" }, { "source": 2, "target": 3, "id": 2, "label": "CREATE" }, { "source": 1, "target": 3, "id": 3, "label": "CONTRIBUTE_TO", "data": { "oldPR": "https://github.com/apache/zeppelin/pull/1582" } } ],
  "labels": { "User": "#8BC34A", "Project": "#3071A9" },
  "directed": true,
  "types": [ "HELPS", "CREATE", "CONTRIBUTE_TO" ]
}
""")
```

FINISHED ▶ ⌂ ⚙



● Grouped ○ Stacked



Took 1 sec. Last updated by anonymous at March 06 2017, 10:35:46 PM. (outdated)

Visualizing

</> Editor

Dashboard

Scheduler

Documents

Files

S3

Tables

Indexes

Jobs

Streams

HBase

Security

Importer

Support

romain@gethue.com

Query

Impala

Add a name...

Add a description...

0.92s

Database default

WHERE a.key = 'shipping' and a.zip_code = '76710';

-- Compute total amount per order for all customers

SELECT

c.id AS customer_id,

c.name AS customer_name,

o.order_id,

v.total

FROM

customers c,

c.orders o,

(SELECT SUM(price * qty) total FROM o.items) v;

Query 834d413ec7474ed5:4249de1000000000 100% Complete (1034d413ec7474ed5:4249de1000000000)

Query 834d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)

Query 834d413ec7474ed5:4249de1000000000 100% Complete (1 out of 1)

Query History

Saved Queries

Results (106)

Execution Analysis

	customer_id	customer_name	order_id	total
1	75012	Dorothy Wilk	4056711	918
2	75012	Dorothy Wilk	J882C2	96
3	17254	Martin Johnson	I72T39	18
4	12532	Melvin Garcia	PB6268	68
5	12532	Melvin Garcia	B8623C	2507
6	12532	Melvin Garcia	R9S838	1278
7	42632	Raymond S. Vestal	HS3124	1944
8	42632	Raymond S. Vestal	BS5902	2798
9	77913	Betty J. Giambrone	DN8815	1320
10	77913	Betty J. Giambrone	XR2771	4315

Tables (5) +

Filter...

customers

id (int)

name (string)

email_preferences (struct)

addresses (map)

orders (array)

k8s_logs

sample_07

sample_08

web_logs

version (bigint)

app (string)

bytes (smallint)

city (string)

client_ip (string)

code (tinyint)

country_code (string)

country_code3 (string)

country_name (string)

device_family (string)

extension (string)

latitude (float)

longitude (float)

method (string)

os_family (string)

os_major (string)

protocol (string)

record (string)

referer (string)

region_code (bigint)

request (string)

subapp (string)

time (string)

url (string)

Tables

Statement 3/3

Filter...

default.customers

id int

name string

email_preferences struct

addresses map

orders array

A data platform canvas



Acquisition



Transport



Storage



Processing



Servicing



Security

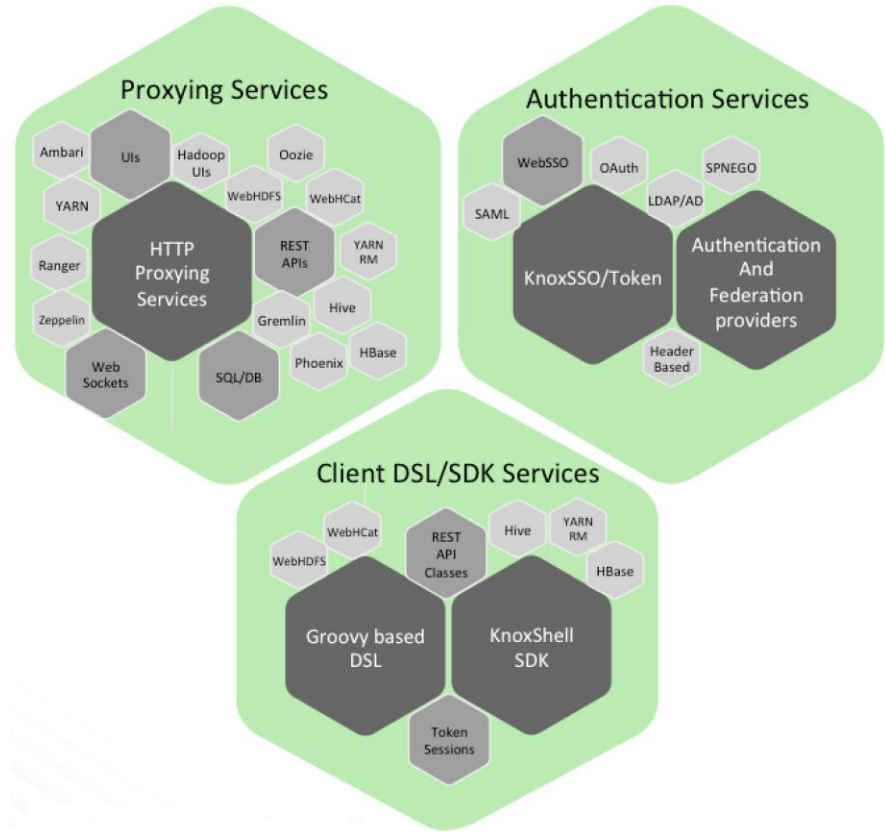


Orchestration

Security



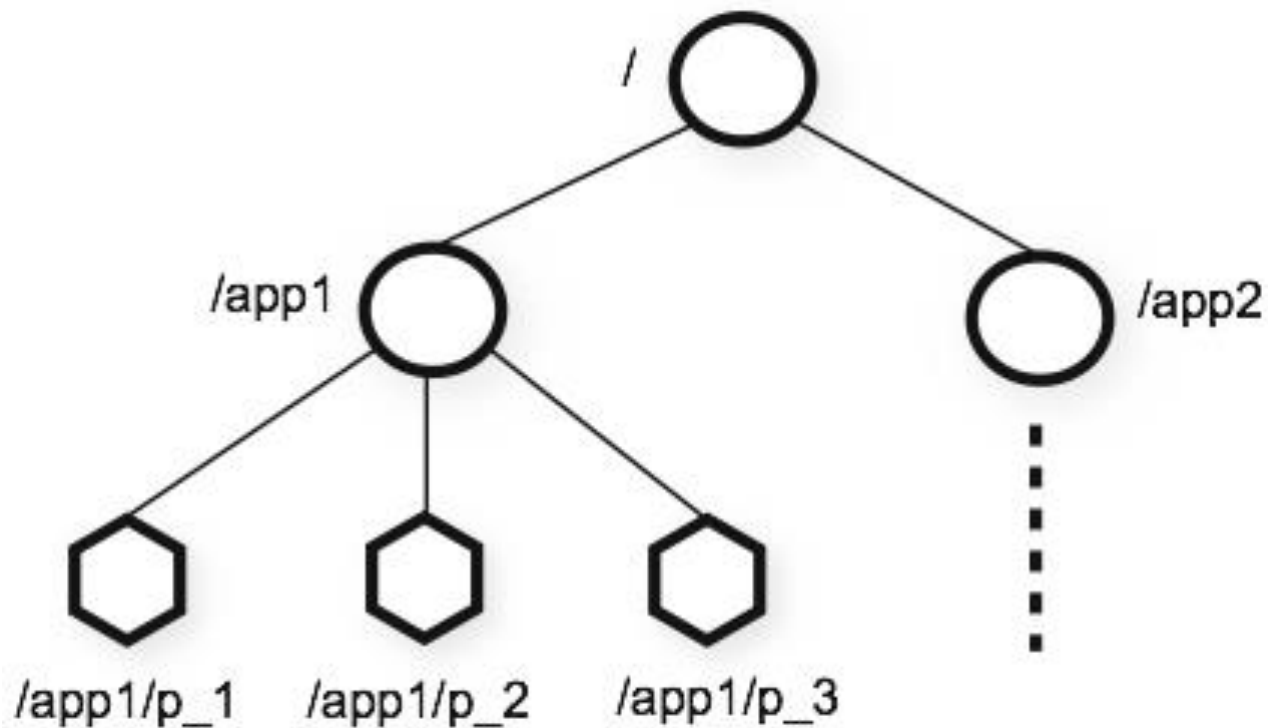
Kerberos



Orchestration



Apache ZooKeeper™



Orchestration



Airflow

DAGsData ProfilingBrowseAdminDocsAbout

On DAG: example_bash_operator

Graph ViewTree ViewTask DurationTask TriesLanding TimesGanttDetailsCodeRefreshDelete

example_bash_operator

```
1  # -*- coding: utf-8 -*-
2  #
3  # Licensed to the Apache Software Foundation (ASF) under one
4  # or more contributor license agreements. See the NOTICE file
5  # distributed with this work for additional information
6  # regarding copyright ownership. The ASF licenses this file
7  # to you under the Apache License, Version 2.0 (the
8  # "License"); you may not use this file except in compliance
9  # with the License. You may obtain a copy of the License at
10 #
11 # http://www.apache.org/licenses/LICENSE-2.0
12 #
13 # Unless required by applicable law or agreed to in writing,
14 # software distributed under the License is distributed on an
15 # "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
16 # KIND, either express or implied. See the License for the
17 # specific language governing permissions and limitations
18 # under the License.
19
20 import airflow
21 from builtins import range
22 from airflow.operators.bash_operator import BashOperator
23 from airflow.operators.dummy_operator import DummyOperator
24 from airflow.models import DAG
25 from datetime import timedelta
26
27
28 args = {
29     'owner': 'airflow',
30     'start_date': airflow.utils.dates.days_ago(2)
31 }
32
33 dag = DAG(
34     dag_id='example_bash_operator', default_args=args,
35     schedule_interval='0 0 * * *',
36     dagrun_timeout=timedelta(minutes=60))
37
38 cmd = 'ls -l'
39 run_this_last = DummyOperator(task_id='run_this_last', dag=dag)
40
41 # [START howto_operator_bash]
42 run_this = BashOperator(
43     task_id='run_after_loop', bash_command='echo 1', dag=dag)
44 # [END howto_operator_bash]
45 run_this.set_downstream(run_this_last)
46
```

Overview



Acquisition



Transport



Storage



Processing



Servicing



Security

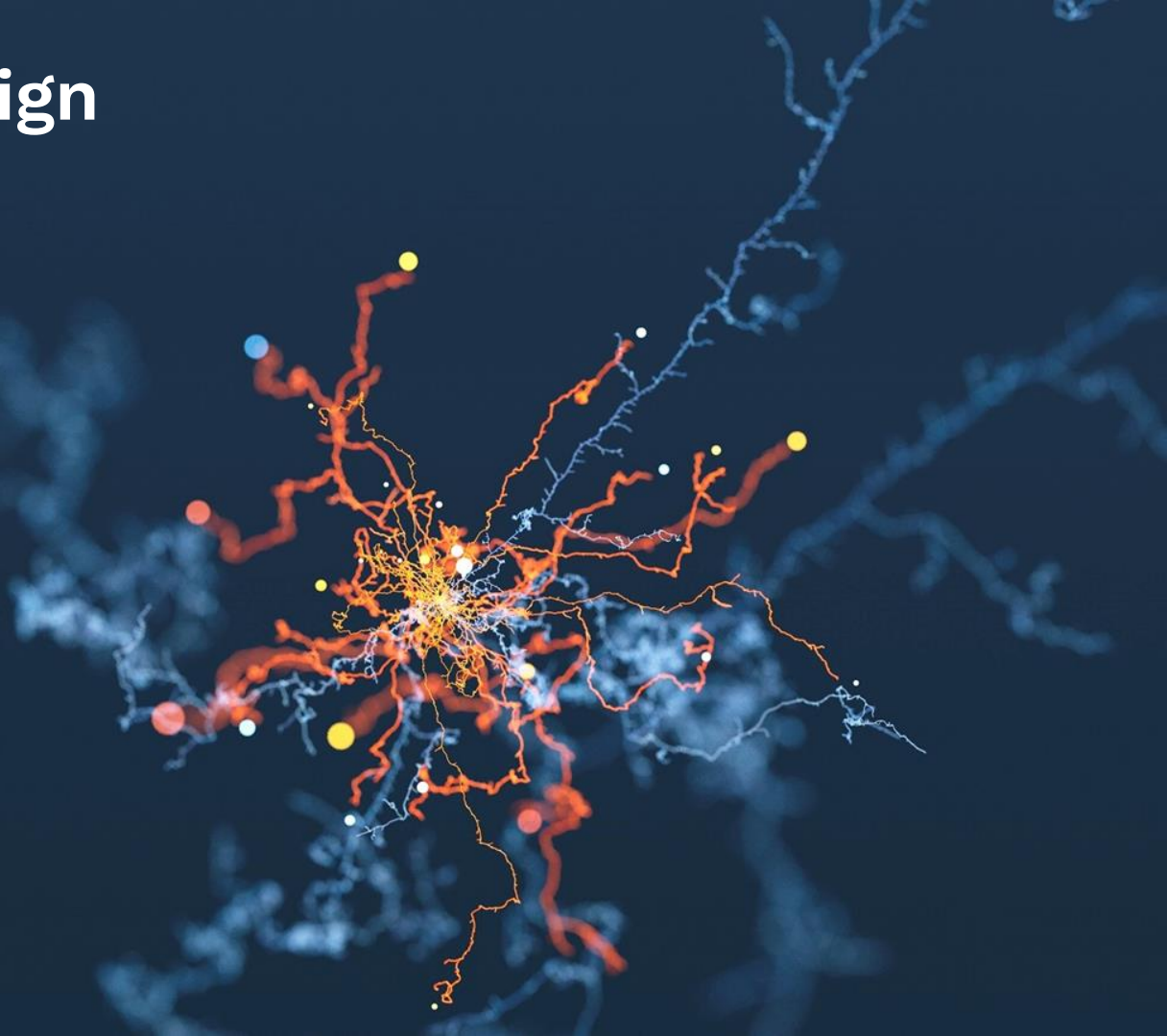


Orchestration

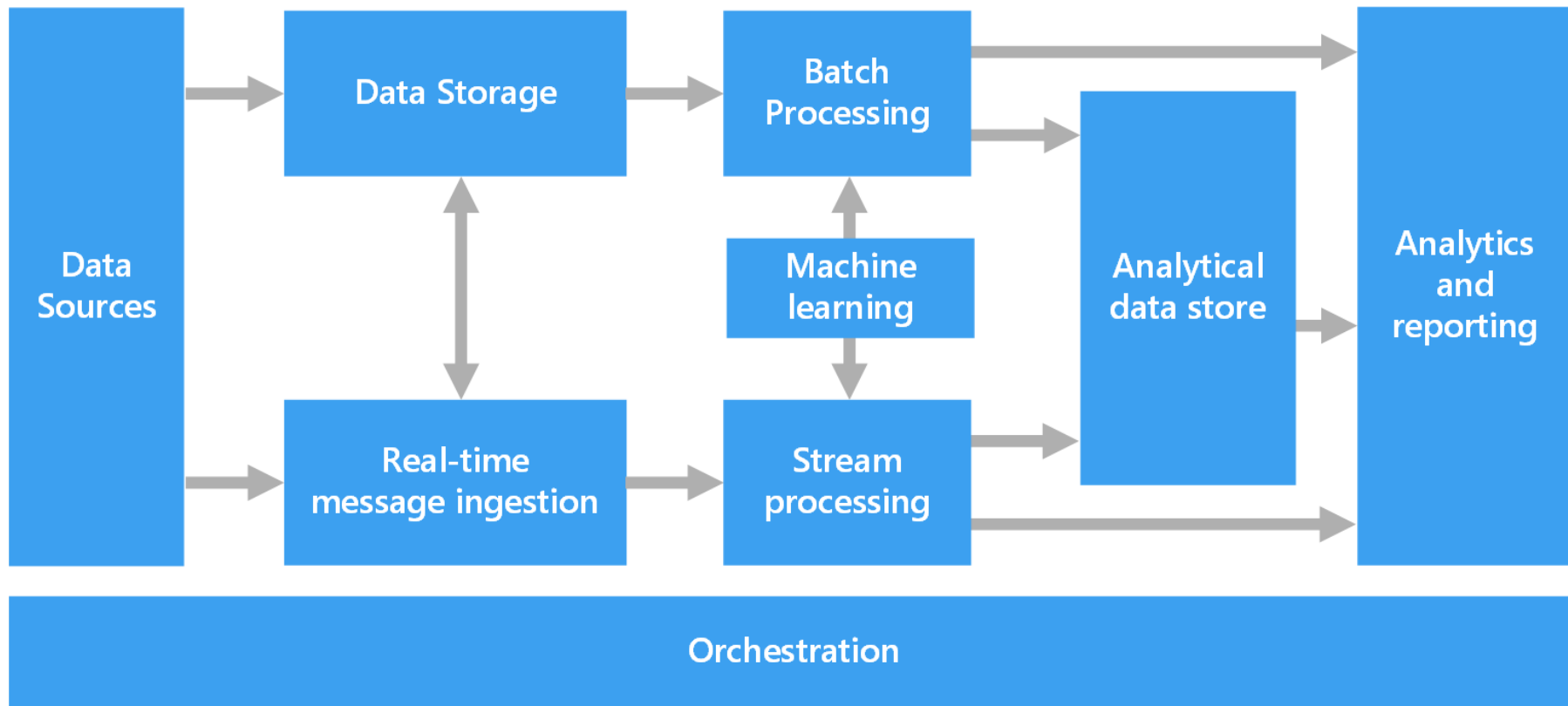


RELAX

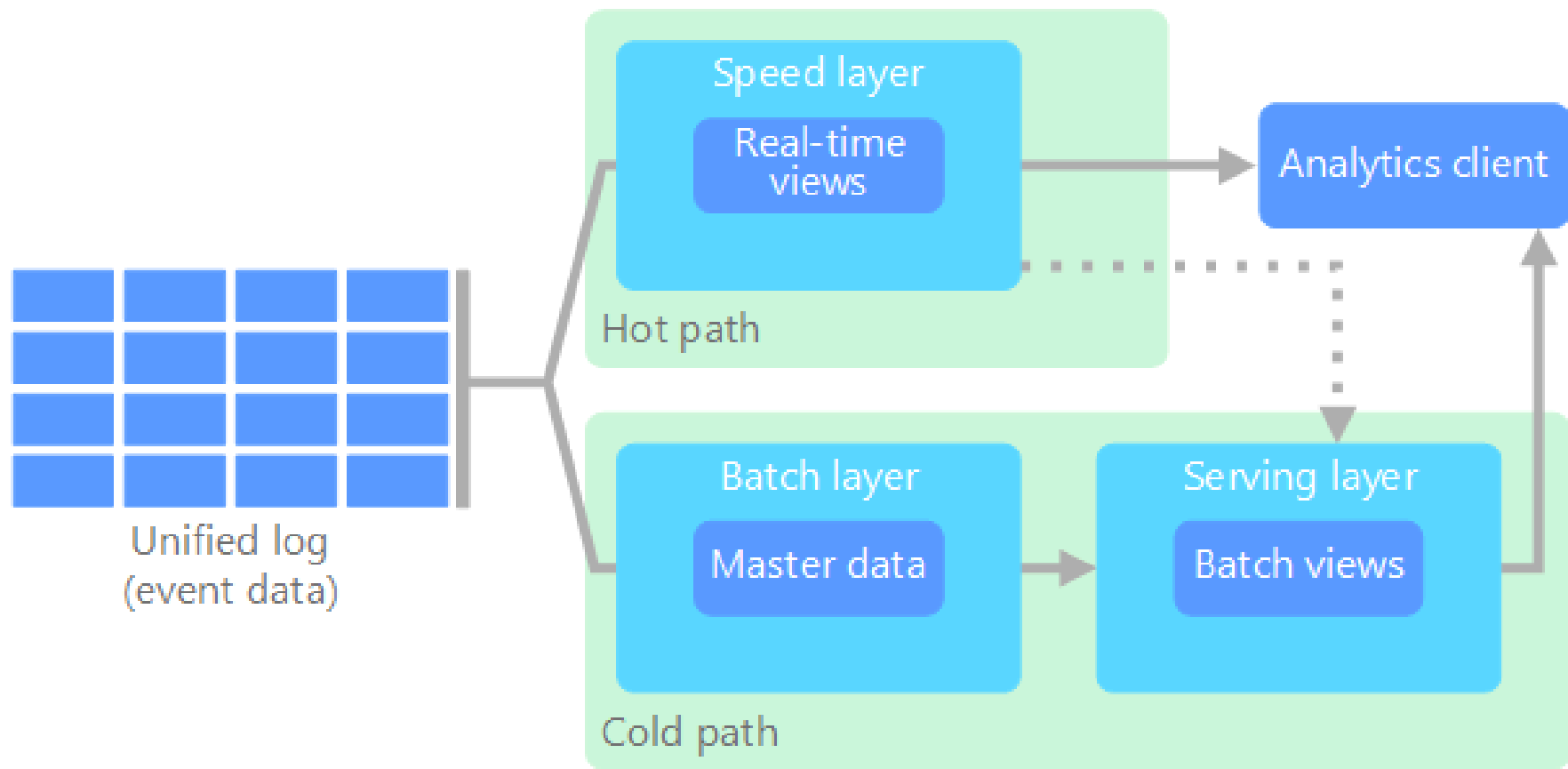
Architecture design



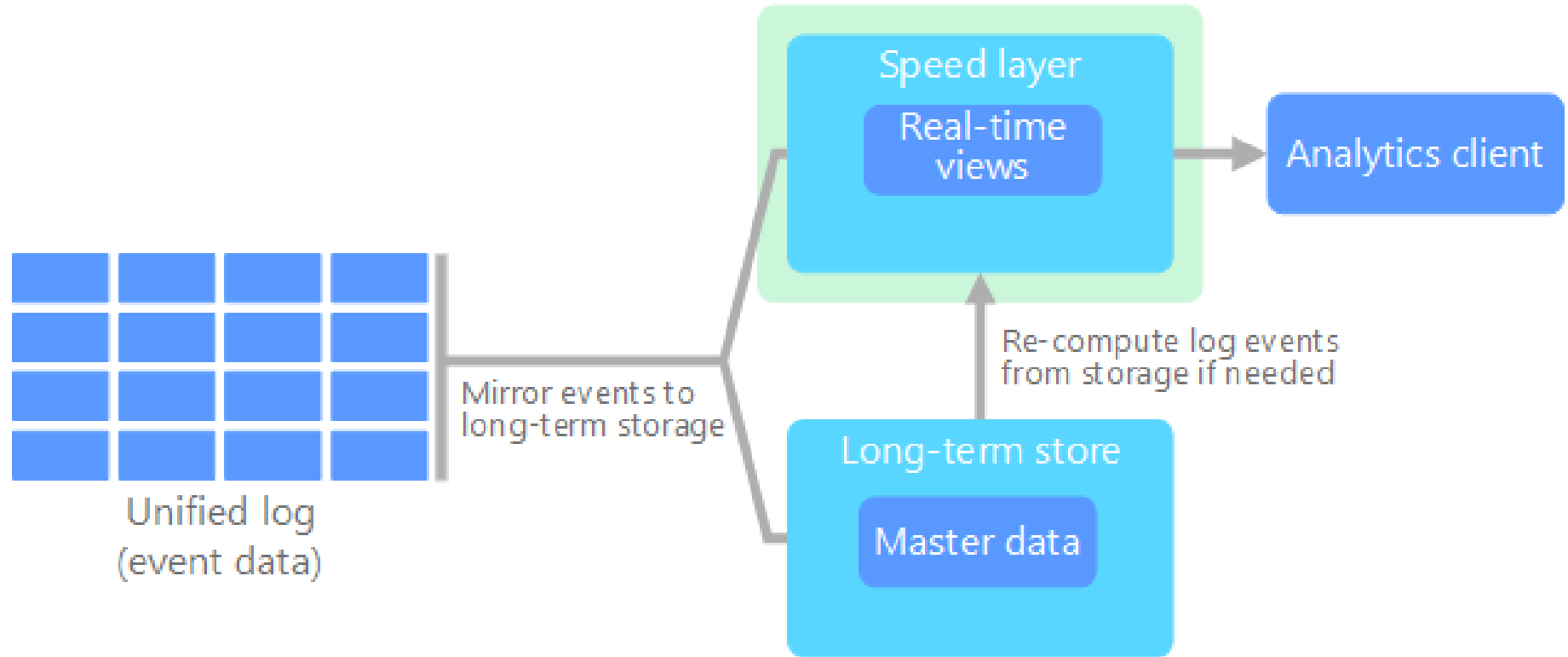
Multiple architectures



Lambda architecture



Kappa architecture





CONCLUSION

THANKS



@andfanilo



@andfanilo



andfanilo@gmail.com