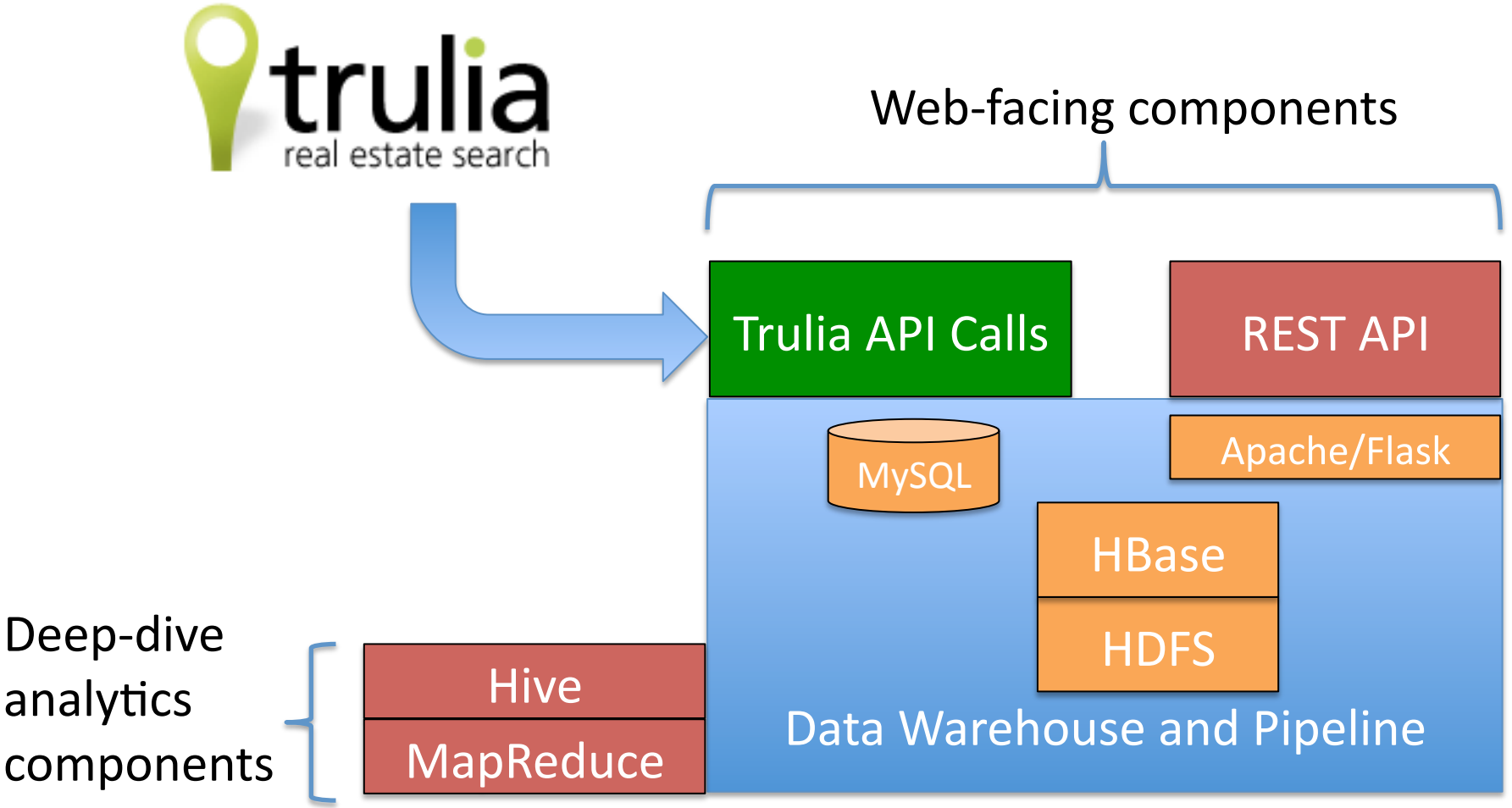
Grand Theft Market

Organizing historical real estate listing data using scalable components

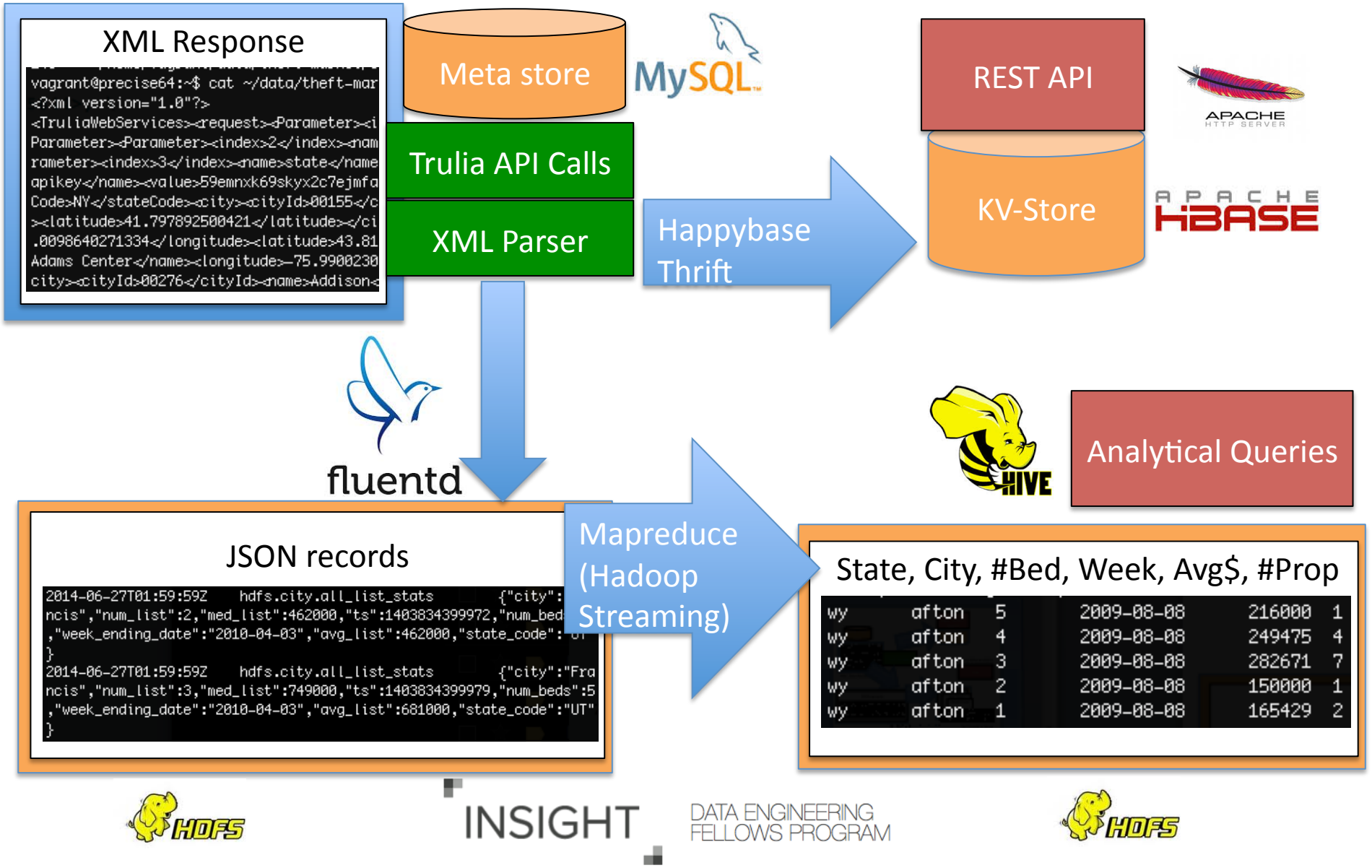
Ryan Irwin

High-level Idea



INSIGHT DATA ENGINEERING FELLOWS PROGRAM

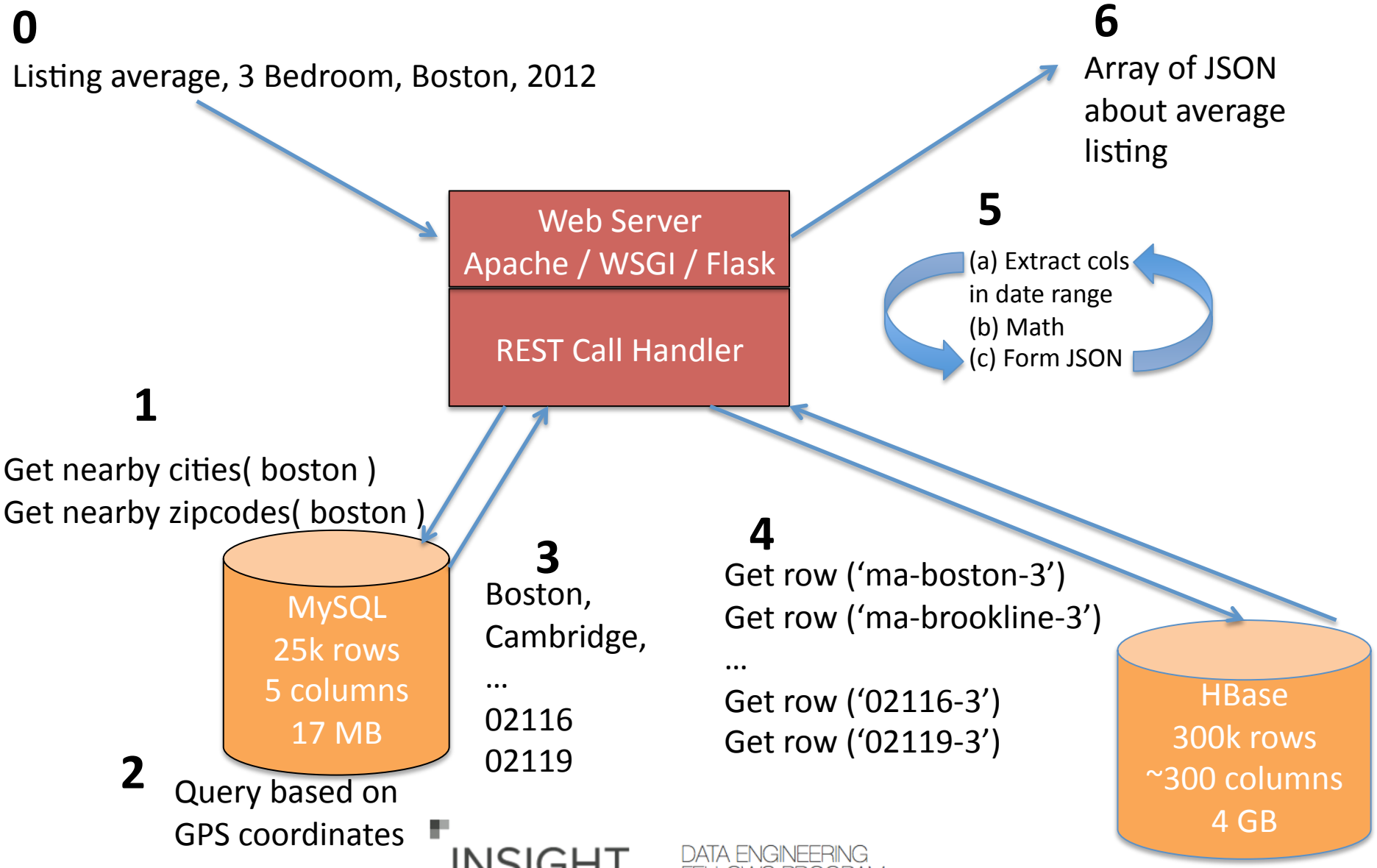
Pipeline Details



Hive Questions

- Which *large* city had the most expensive 1 bedrooms?
  - 2013: NYC $861k, SF $691k, Boston $458k
  - 2010: NYC $780k, Honolulu $629k, SF $573k
- Which city had the highest 8-bedroom list volume in the last 4 years?
  - Brooklyn, NY (but average is $1.5 M)
  - Listing average < $500k - Chicago, IL
- Which *large* city has the cheapest *k*-bedroom listings in last 2 years?
  - 2-bedroom: Tulsa, OK $96k
  - 3-bedroom: Detroit, MI $128k
  - 4-bedroom: El Paso, TX $147k

REST API Response 0 to 6 in < 0.5 s



INSIGHT DATA ENGINEERING FELLOWS PROGRAM

Issues & Next Steps

- Issues:
  - Useful insights from aggregating already-aggregated data
  - HBase thrift server
    - Overloaded easily, runs out of memory on malformed input
    - HBase lookup time tripled while inserting zipcode data
    - Web attacks by w00tw00t
- Next steps:
  - Make installation/configuration easier
  - Finish scripting & coding for pulling recent data only
  - REST API for batch queries

About Me – Ryan Irwin

- BBN as Software Engineer / Scientist
  - Led development of monitoring architecture for NSF's nationwide research network, GENI
  - Help build first 100-node multi-radio ad hoc network (DARPA project)

Raytheon
BBN Technologies

- Virginia Tech, Computer Engineer PhD
  - Graph theory and operations research

Wireless @ Virginia Tech

- Personal
  - Avid runner and volunteer guide
  - Github: https://github.com/rirwin/

INSIGHT DATA ENGINEERING FELLOWS PROGRAM

Example call

http://54.193.52.251/data/city/average?q={"state_code":"MA","city":"Boston","num_bedrooms":3,"start_date":"2013-01-01","end_date":"2014-01-01"}

Total server-side time is 433 ms

Lookup time decreases if window is shorter because of fewer python dictionary evals


```
[
  - {
      city: "Boston",
      end_date: "2014-01-01",
      num_bedrooms: 3,
      list_average: 1014409,
      state_code: "MA",
      start_date: "2008-01-01"
    },
  - {
      city: "Brookline",
      end_date: "2014-01-01",
      num_bedrooms: 3,
      list_average: 1012853,
      state_code: "MA",
      start_date: "2008-01-01"
    },
  - {
      city: "Cambridge",
      end_date: "2014-01-01",
      num_bedrooms: 3,
      list_average: 827104,
      state_code: "MA",
      start_date: "2008-01-01"
```


```
INFO:root:------------- Start Data Warehouse Access -----------------
INFO:root:get nearby cities from MySQL (GPS calculations) took: 0.00100111961365s
INFO:root:get nearby zipcodes from MySQL (GPS calculations) took: 0.00074291229248s
INFO:root:get city average listings from HBase (10 row lookup and manipulation) took: 0.232179164886s
INFO:root:get zipcode average listings from HBase (10 row lookup and manipulation) took: 0.199162960052s
INFO:root:Total time took: 0.433792114258s
INFO:root:------------- Done with Data Warehouse Access -----------------
```

```
@wrappers.general_function_handler
def get_city_list_average(self, state_code, city, num_bedrooms, start_date, end_date):

    row_key = '-'.join([state_code, city, str(num_bedrooms)])

    data =  self.city_stats_table.row(row_key)

    start_datetime = datetime.datetime.strptime(start_date, '%Y-%m-%d')
    end_datetime = datetime.datetime.strptime(end_date, '%Y-%m-%d')

    # Pythonic way to extract relevant columns of row
    filtered_keys = [k for k in data.keys() if k.startswith('cf:') \
                        and datetime.datetime.strptime(k[3:],"%Y-%m-%d") < end_datetime \
                        and datetime.datetime.strptime(k[3:],"%Y-%m-%d") > start_datetime]

    num = 0
    denom = 0
    for key in filtered_keys:
        num += int(eval(data[key])['a'])*int(eval(data[key])['n'])
        denom += int(eval(data[key])['n'])
    if denom == 0:
        return 0
    else:
        return round(float(num)/float(denom))
```