

Hadoop, Pig, and Twitter

Kevin Weil -- @kevinweil
Analytics Lead, Twitter



Introduction

- ▶ Hadoop Overview
- ▶ Why Pig?
- ▶ Evolution of Data Processing at Twitter
- ▶ Pig for Counting
- ▶ Pig for Correlating
- ▶ Pig for Research and Data Mining
- ▶ Conclusions and Next Steps

My Background

- ▶ Studied Mathematics and Physics at Harvard, Physics at Stanford
- ▶ **Tropos Networks** (city-wide wireless): mesh routing algorithms, GBs of data
- ▶ **Cooliris** (web media): Hadoop and Pig for analytics, TBs of data
- ▶ **Twitter**: Hadoop, Pig, machine learning, visualization, social graph analysis, ??? of data

Introduction

- ▶ **Hadoop Overview**
- ▶ Why Pig?
- ▶ Evolution of Data Processing at Twitter
- ▶ Pig for Counting
- ▶ Pig for Correlating
- ▶ Pig for Research and Data Mining
- ▶ Conclusions and Next Steps

Data is Getting Big

- ▶ NYSE: 1 TB/day
- ▶ Facebook: 20+ TB compressed/day
- ▶ CERN/LHC: 40 TB/day (15 PB/year!)
- ▶ And growth is accelerating
- ▶ Need multiple machines, horizontal scalability

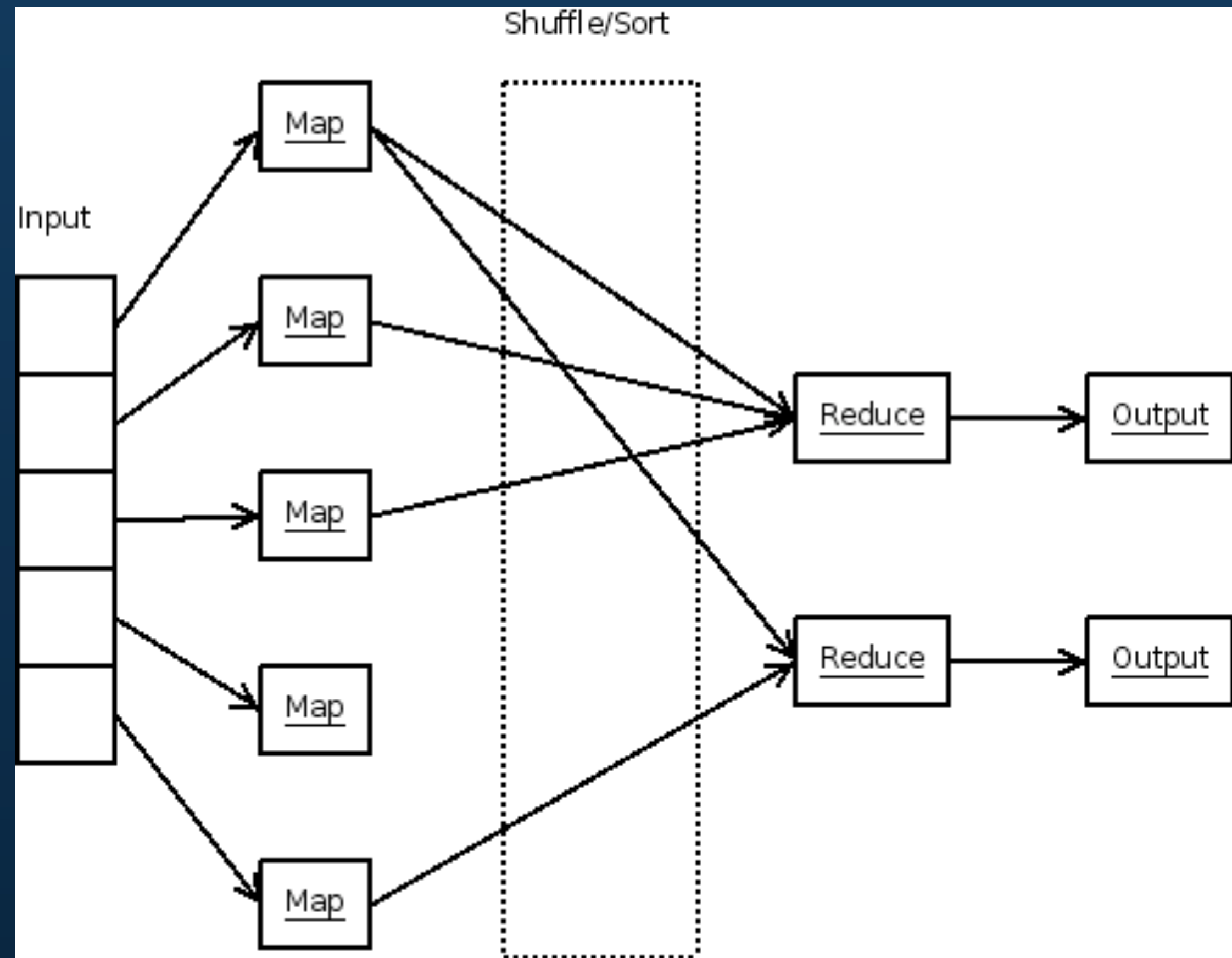


Hadoop



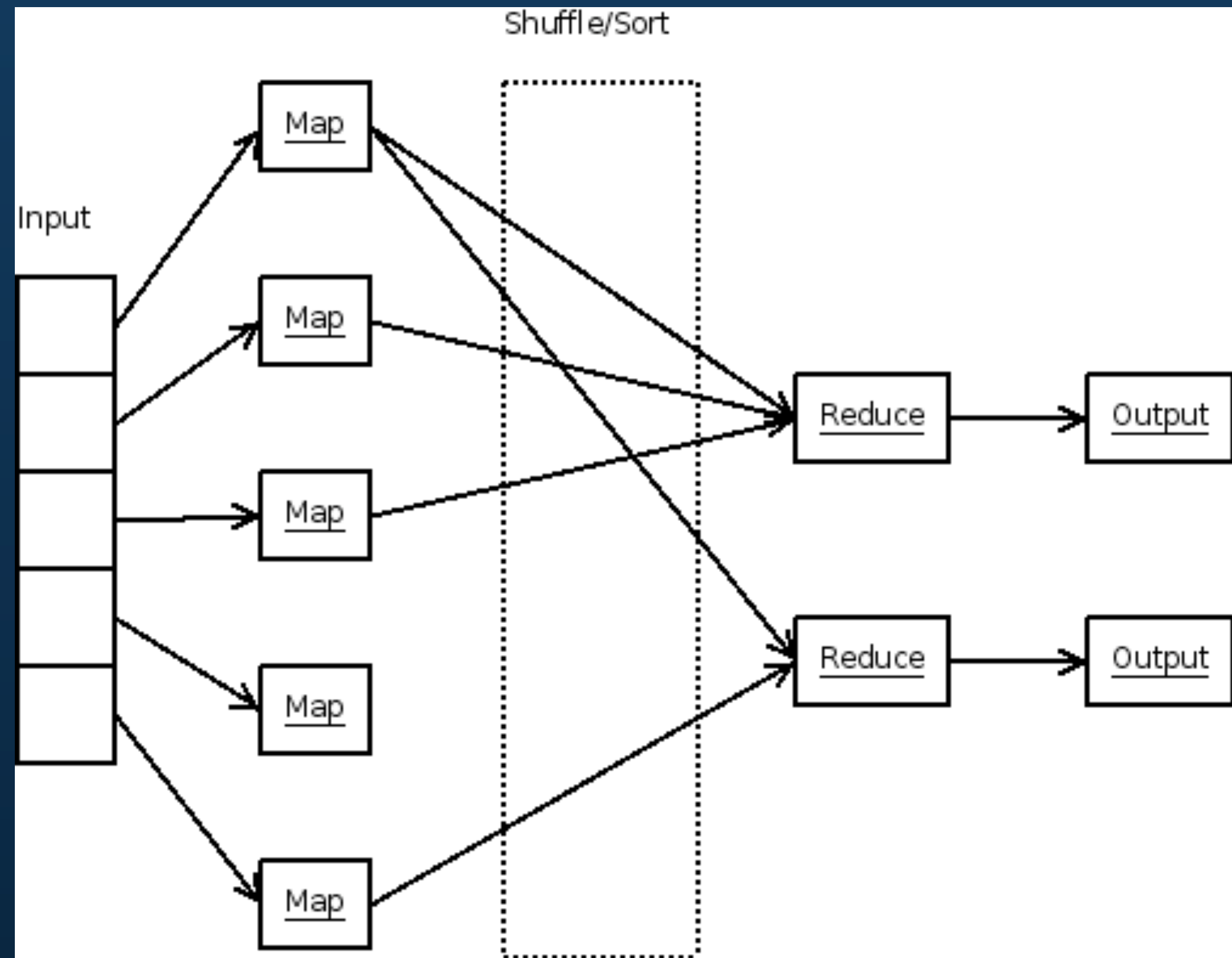
- ▶ Distributed file system (hard to store a PB)
- ▶ Fault-tolerant, handles replication, node failure, etc
- ▶ MapReduce-based parallel computation (even harder to process a PB)
- ▶ Generic key-value based computation interface allows for wide applicability
- ▶ Open source, top-level Apache project
- ▶ Scalable: Y! has a 4000-node cluster
- ▶ Powerful: sorted a TB of random integers in 62 seconds

MapReduce?



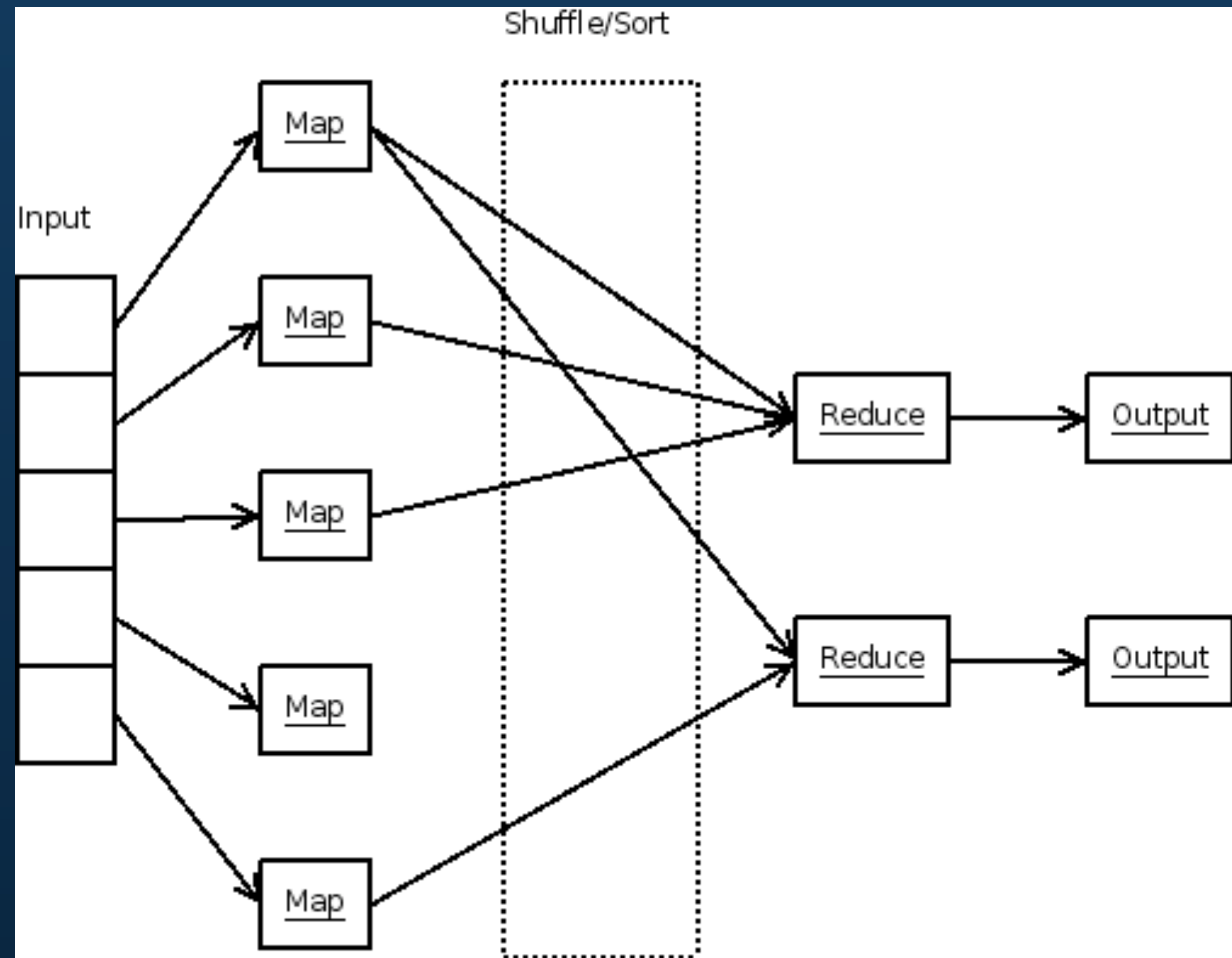
- ▶ **Challenge:** how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

MapReduce?



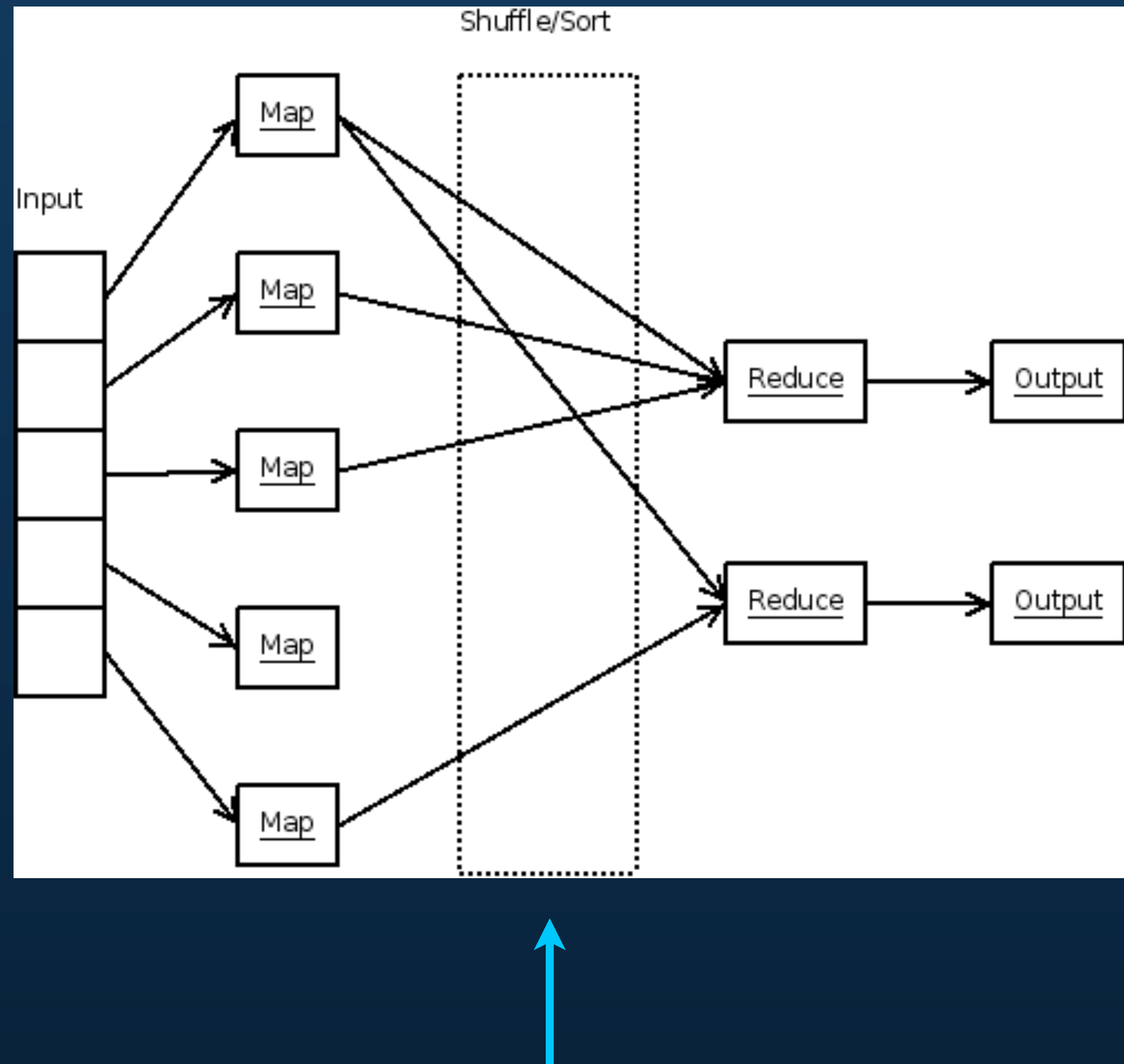
- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

MapReduce?



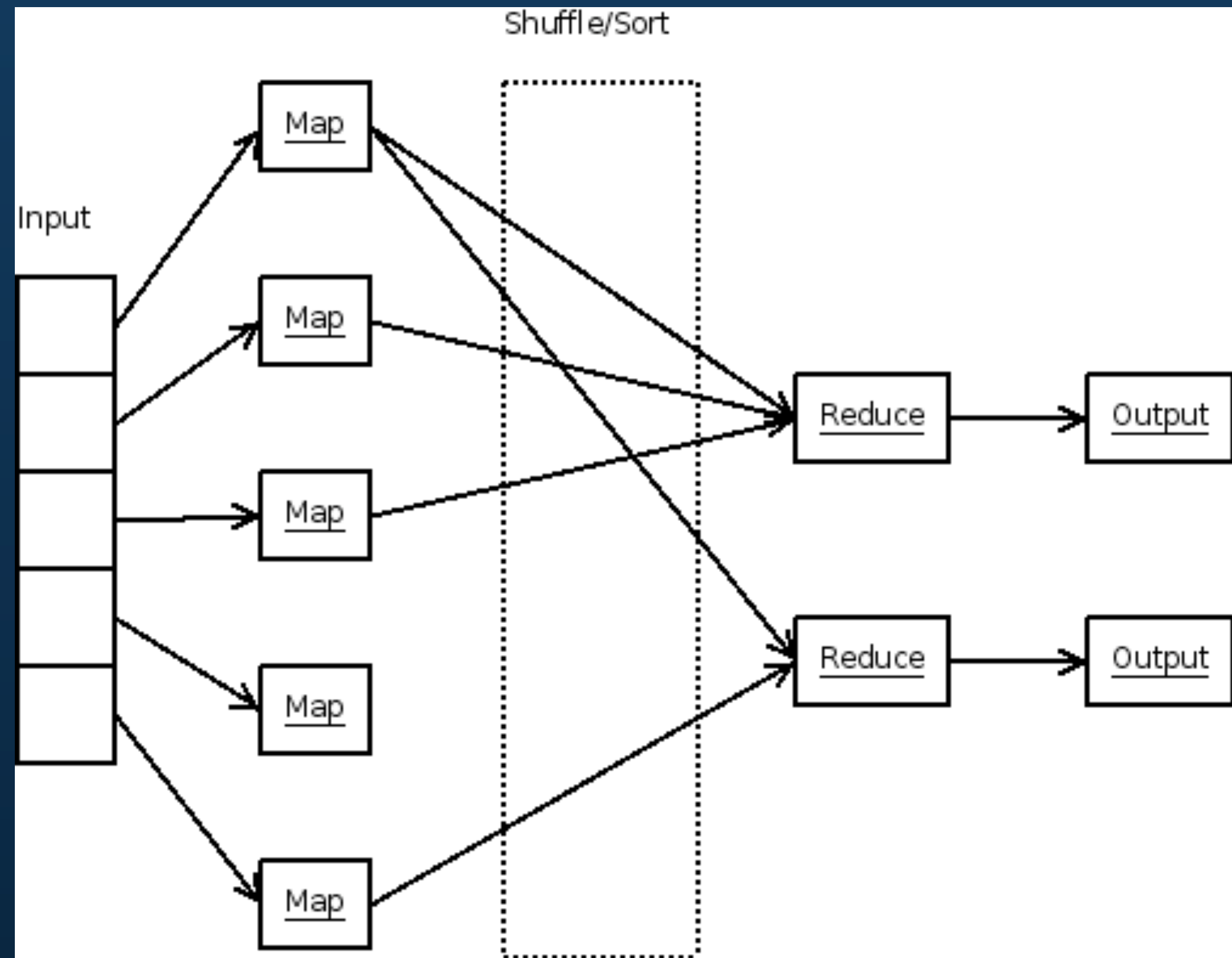
- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

MapReduce?



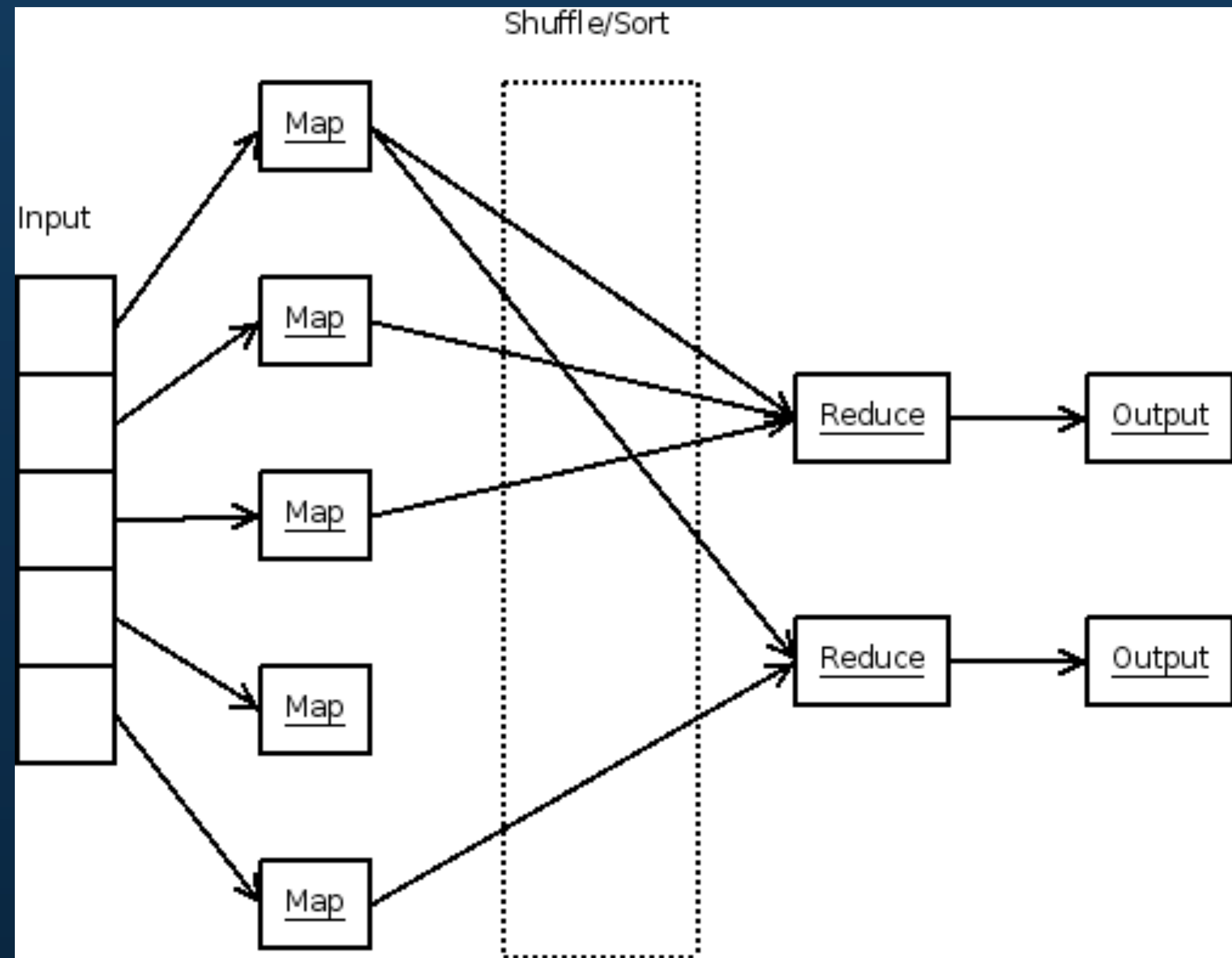
- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

MapReduce?



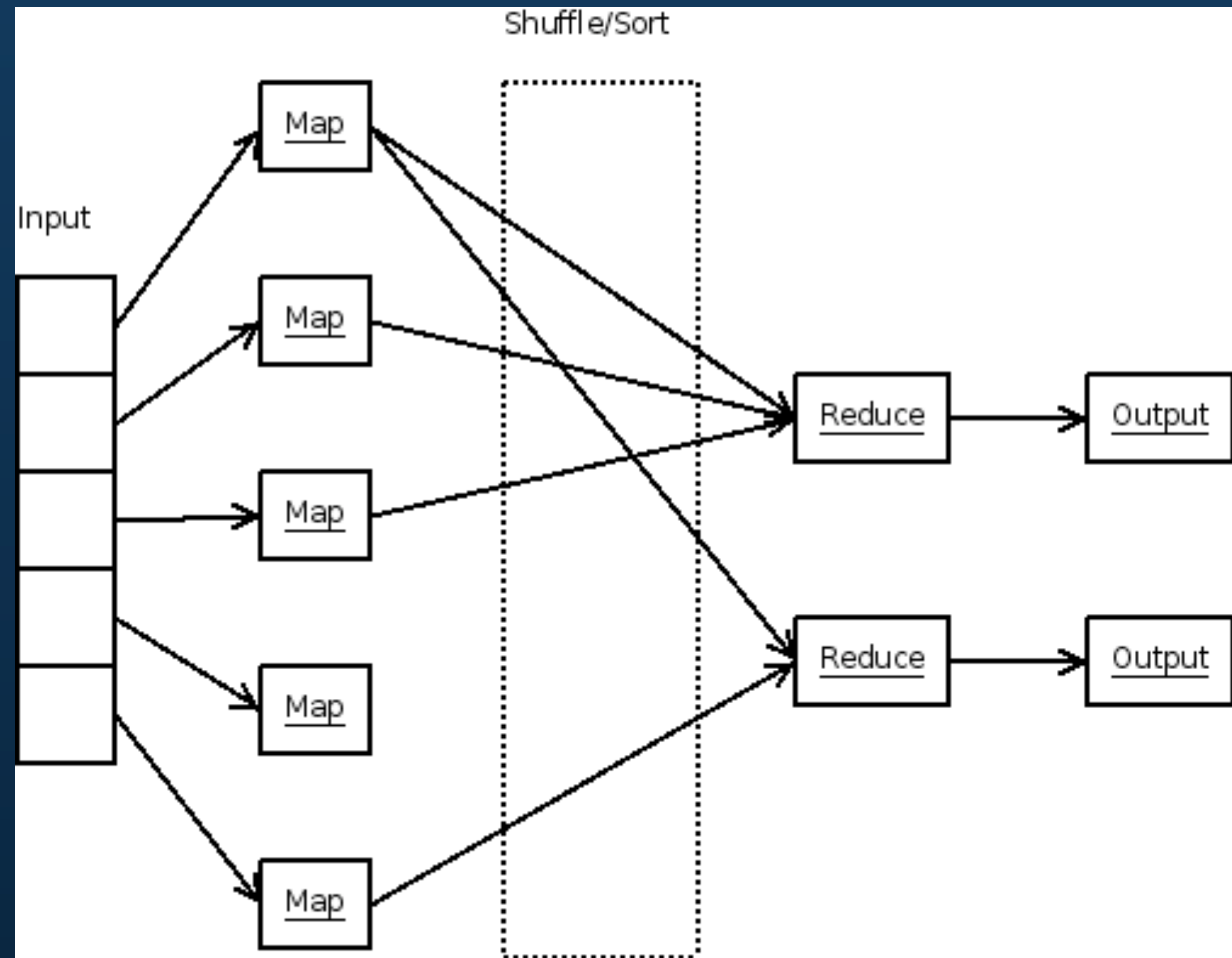
- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

MapReduce?



- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ With 2x machines, runs close to 2x faster.

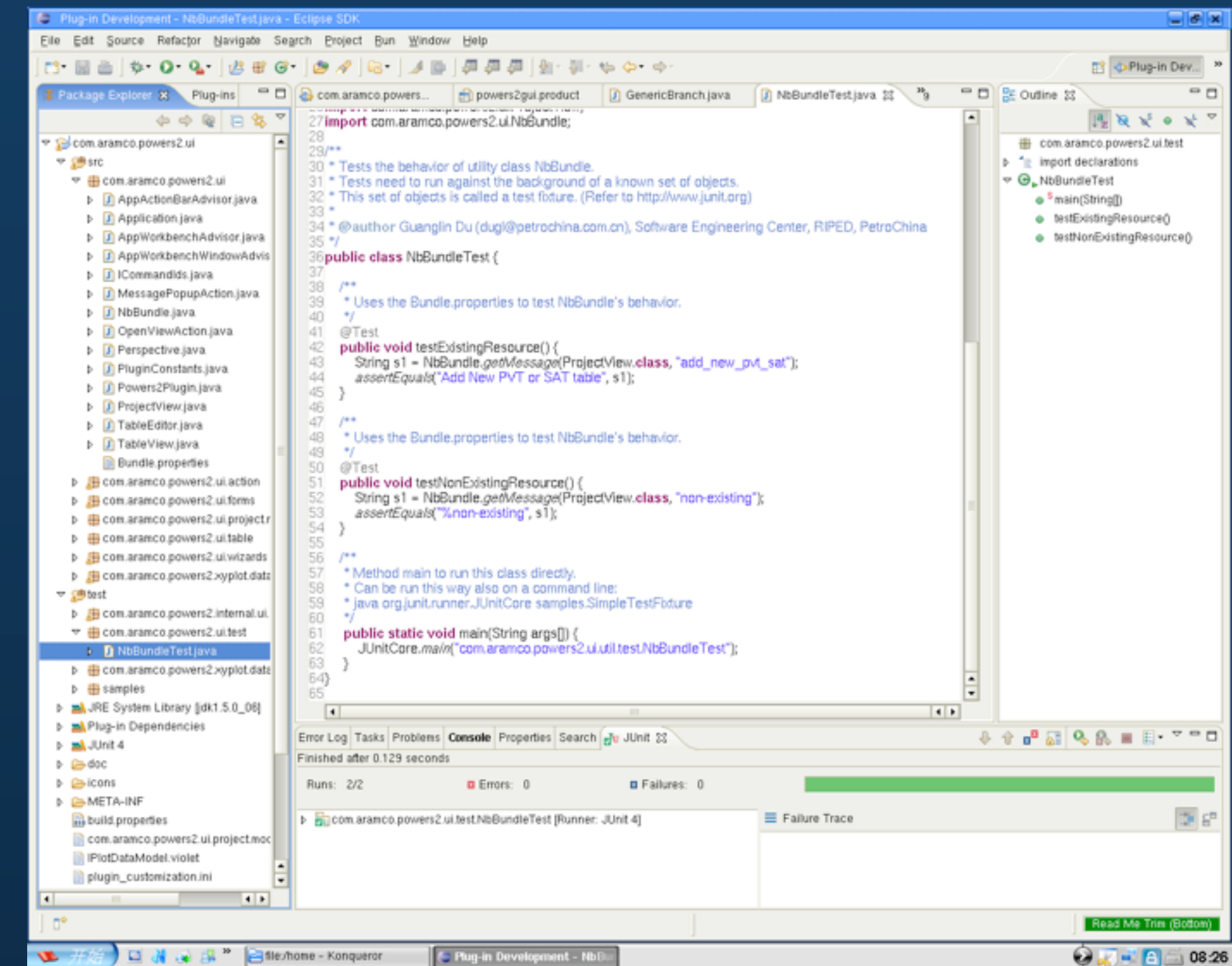
MapReduce?



- ▶ Challenge: how many tweets per user, given tweets table?
- ▶ Input: key=row, value=tweet info
- ▶ Map: output key=user_id, value=1
- ▶ Shuffle: sort by user_id
- ▶ Reduce: for each user_id, sum
- ▶ Output: user_id, tweet count
- ▶ **With 2x machines, runs close to 2x faster.**

But...

- ▶ Analysis typically done in Java
- ▶ Single-input, two-stage data flow is rigid
- ▶ Projections, filters: custom code
- ▶ Joins: lengthy, error-prone
- ▶ n-stage jobs: Hard to manage
- ▶ Prototyping/exploration requires compilation



- ▶ analytics in Eclipse?
ur doin it wrong...

Enter Pig

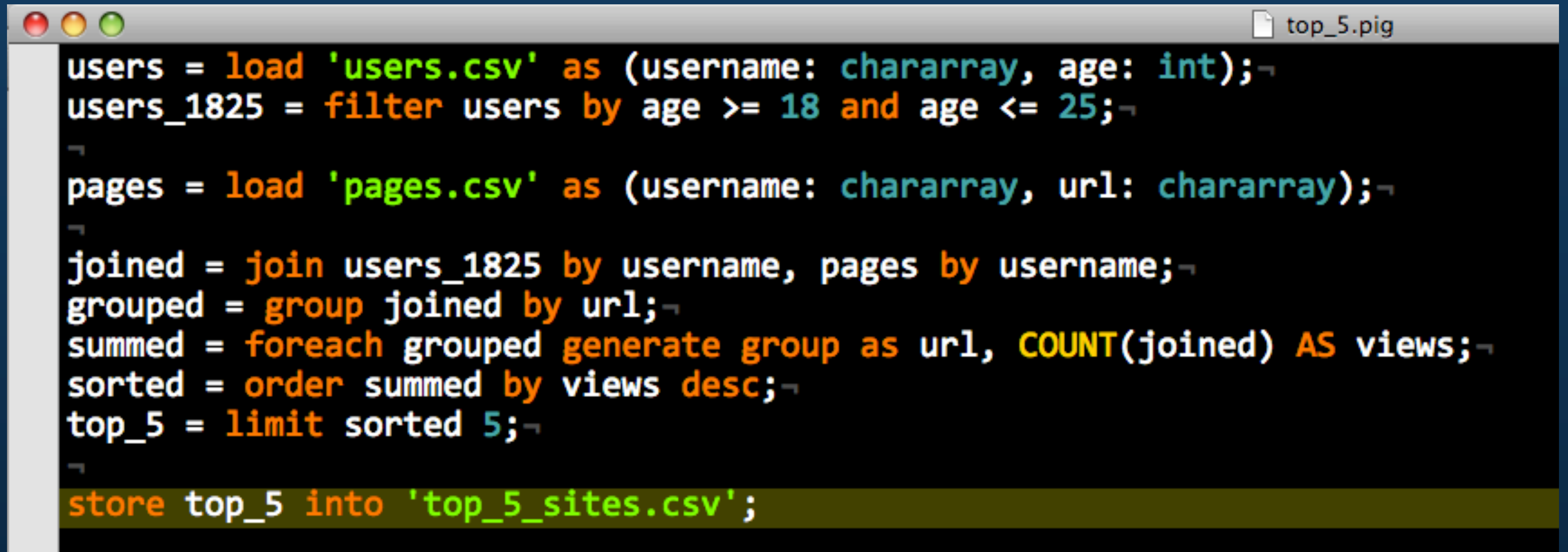


- ▶ High level language
- ▶ Transformations on sets of records
- ▶ Process data one step at a time
- ▶ Easier than SQL?

Why Pig?

- ▶ Because I bet you can read the following script.

A Real Pig Script



```
users = load 'users.csv' as (username: chararray, age: int);  
users_1825 = filter users by age >= 18 and age <= 25;  
  
pages = load 'pages.csv' as (username: chararray, url: chararray);  
  
joined = join users_1825 by username, pages by username;  
grouped = group joined by url;  
summed = foreach grouped generate group as url, COUNT(joined) AS views;  
sorted = order summed by views desc;  
top_5 = limit sorted 5;  
  
store top_5 into 'top_5_sites.csv';
```

- ▶ Now, just for fun... the same calculation in vanilla Hadoop MapReduce.

No, seriously.

[illegible]

Pig Democratizes Large-scale Data Analysis

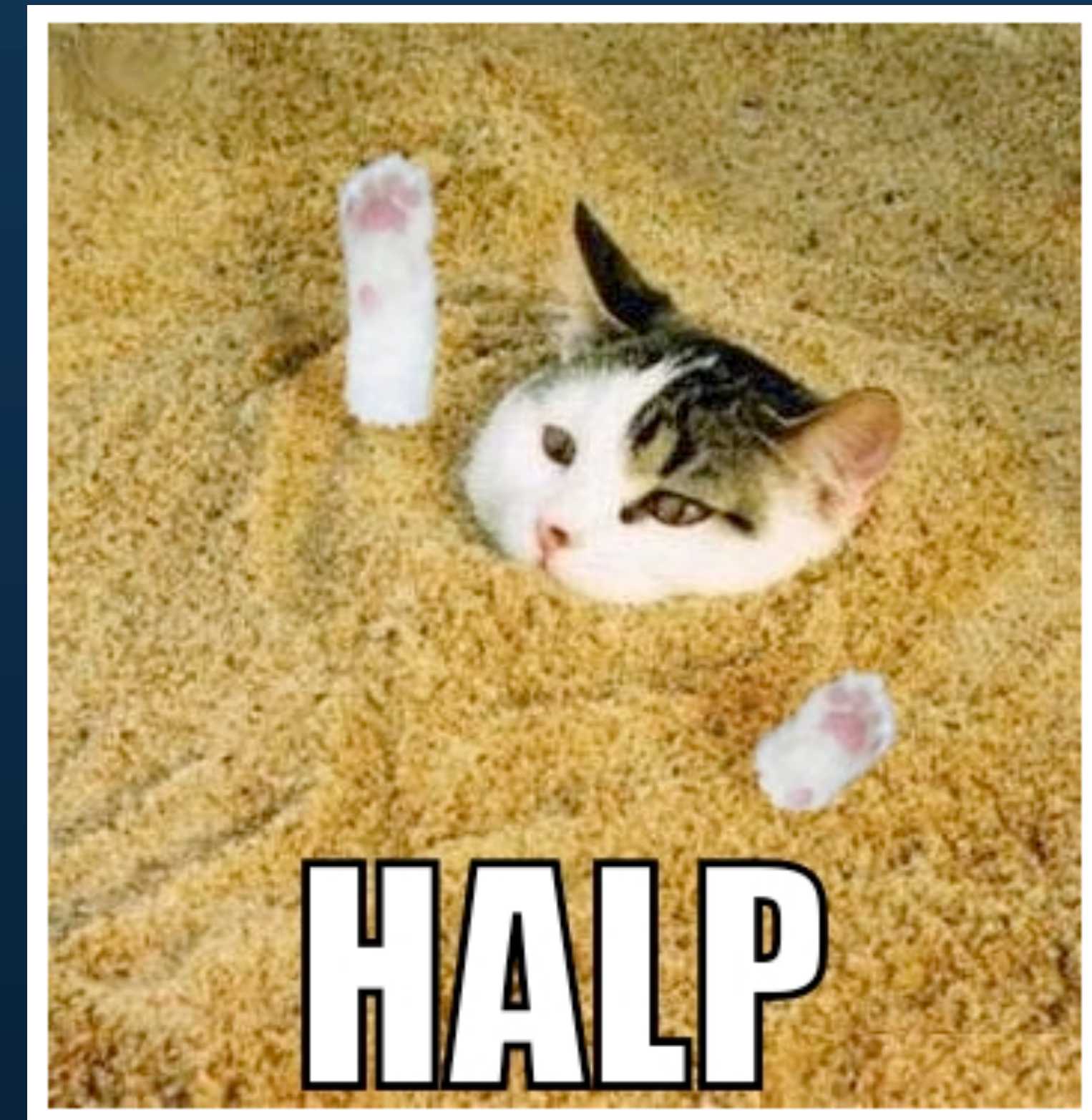
- ▶ The Pig version is:
 - ▶ **5% of the code**
 - ▶ **5% of the time**
 - ▶ Within 50% of the execution time.
- ▶ Innovation increasingly driven from large-scale data analysis
 - ▶ Need fast iteration to understand the *right questions*
 - ▶ More minds contributing = more value from your data

Introduction

- ▶ Hadoop Overview
- ▶ Why Pig?
- ▶ **Evolution of Data Processing at Twitter**
- ▶ Pig for Counting
- ▶ Pig for Correlating
- ▶ Pig for Research and Data Mining
- ▶ Conclusions and Next Steps

MySQL, MySQL, MySQL

- ▶ We all start there.
- ▶ But MySQL is not built for analysis.
- ▶ `select count(*) from users?` Maybe.
- ▶ `select count(*) from tweets?` Uh...
- ▶ Imagine joining them.
- ▶ And grouping.
- ▶ Then sorting.



The Hadoop Ecosystem at Twitter

- ▶ Cloudera's free distribution, running Hadoop 0.20.1
- ▶ Heavily modified Facebook Scribe for log collection -> HDFS*
- ▶ Heavily modified LZO code for fast, splittable data compression**
- ▶ Data stored either as LZO-compressed flat files (logs, etc) or serialized, LZO-compressed protocol buffers (structured data).
- ▶ Custom InputFormats, Pig LoadFuncs for the above*
- ▶ Some Java-based MapReduce, some Hadoop Streaming
- ▶ **Most analysis, and most interesting analyses, done in Pig.**

▶ * Open sourced, or on the way. Please come talk afterwards if you're interested.

▶ ** <http://www.github.com/kevinweil/hadoop-lzo>

Data?



- ▶ **Semi-structured:** apache logs, search logs, RoR logs, mysql query logs, rate limiter logs, per-application logs
- ▶ **Structured:** tweets, users, block notifications, phones, favorites, saved searches, retweets, authentications, sms usage, third party clients, followings
- ▶ **Entangled:** the social graph

Introduction

- ▶ Hadoop Overview
- ▶ Why Pig?
- ▶ Evolution of Data Processing at Twitter
- ▶ **Pig for Counting**
- ▶ Pig for Correlating
- ▶ Pig for Research and Data Mining
- ▶ Conclusions and Next Steps

Counting Big Data

- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶



Counting Big Data

- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶ What is the average latency? 95% latency?
- ▶



Counting Big Data



- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶ What is the average latency? 95% latency?
- ▶ Group by response code. What is the hourly distribution?
- ▶

Counting Big Data



- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶ What is the average latency? 95% latency?
- ▶ Group by response code. What is the hourly distribution?
- ▶ How many searches happen each day on Twitter?
- ▶

Counting Big Data



- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶ What is the average latency? 95% latency?
- ▶ Group by response code. What is the hourly distribution?
- ▶ How many searches happen each day on Twitter?
- ▶ How many unique queries, how many unique users?
- ▶

Counting Big Data



- ▶ **standard counts, min, max, std dev**
- ▶ How many requests do we serve in a day?
- ▶ What is the average latency? 95% latency?
- ▶ Group by response code. What is the hourly distribution?
- ▶ How many searches happen each day on Twitter?
- ▶ How many unique queries, how many unique users?
- ▶ What is their geographic distribution?

Counting Big Data

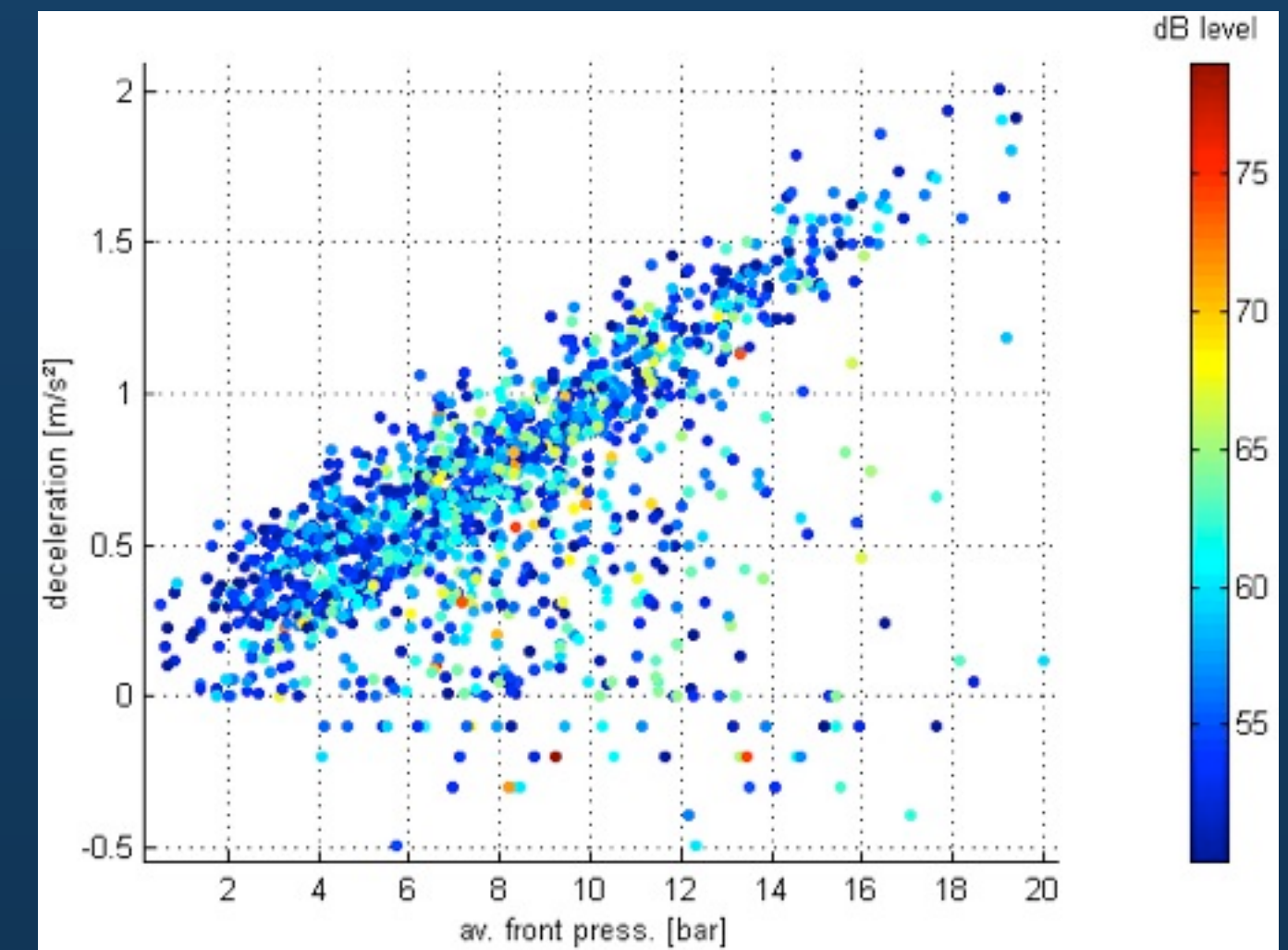
- Where are users querying from? The API, the front page, their profile page, etc?



```
search_origin.pig
raw_data = load '$INPUT_FILES' using com.twitter.twadoop.pig.storage.LzoTwitterApacheLogLoader() as
  (... , virtual_host: chararray, apache_time: chararray, request_method: chararray, request_url: chararray,
  request_protocol: chararray, response_code: chararray, response_size: int, referrer: chararray,
  user_agent: chararray, response_microseconds: int, ...);
searches_only = filter raw_data by com.twitter.twadoop.pig.piggybank.IsSearchUrl(request_url);
searches_with_type = foreach searches_only generate
  com.twitter.twadoop.pig.piggybank.ExtractSearchOrigin(virtual_host, request_url) as origin;
grouped = group searches_with_type by origin parallel $PARALLEL;
counted = foreach grouped generate group, COUNT(searches_with_type);
store counted into 'searches_by_type.tsv' using PigStorage('\t');
```

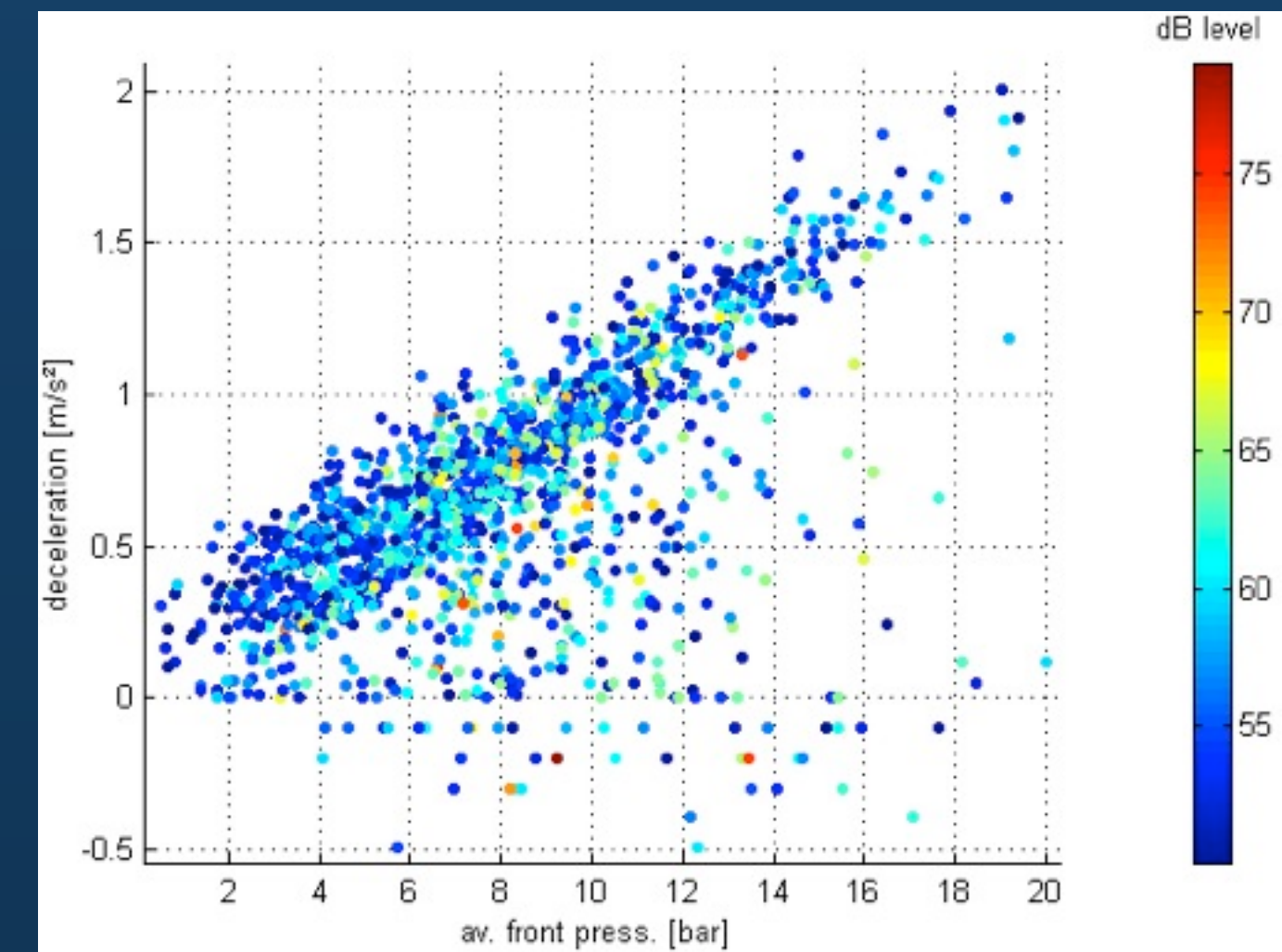
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?



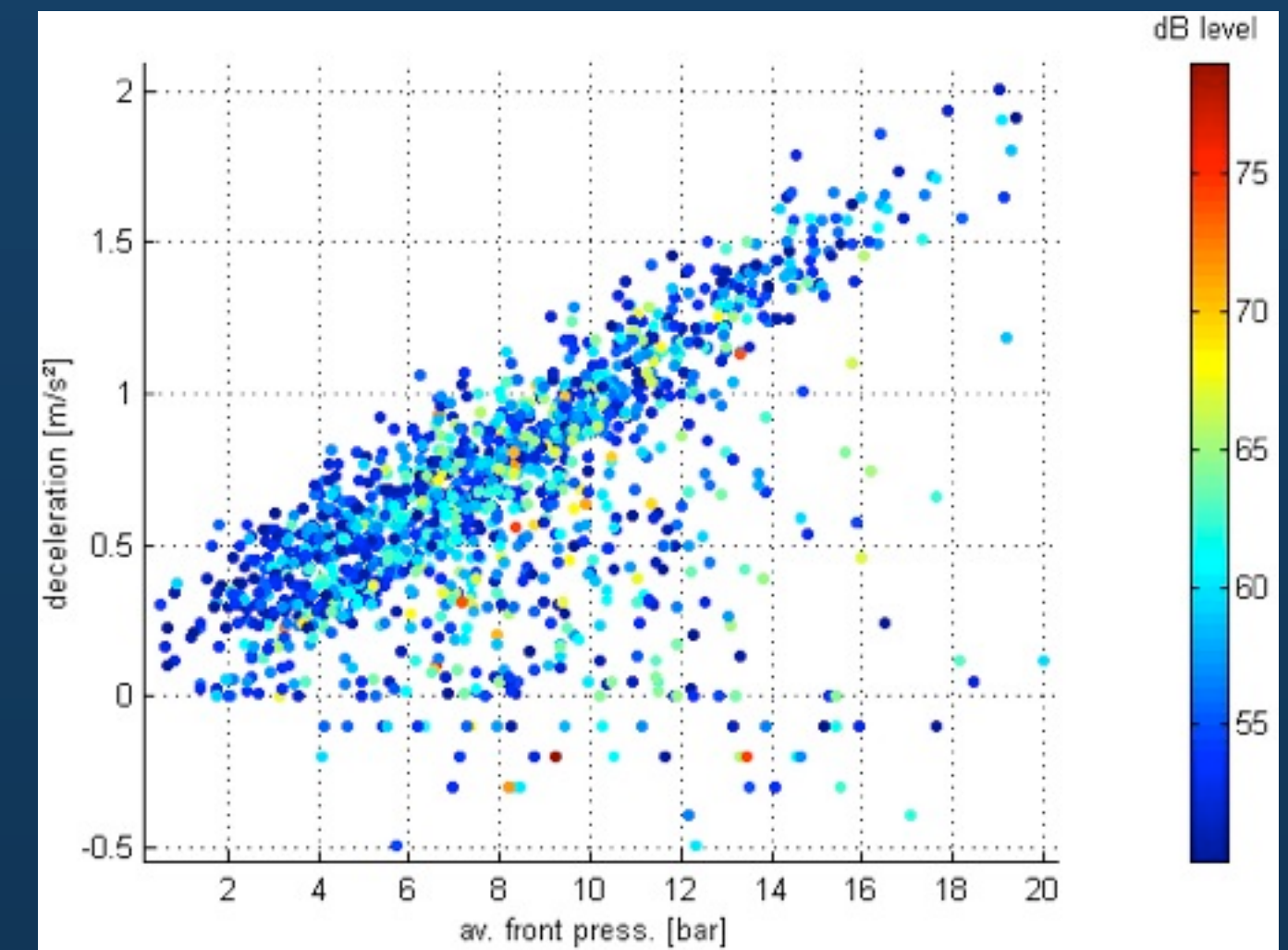
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?



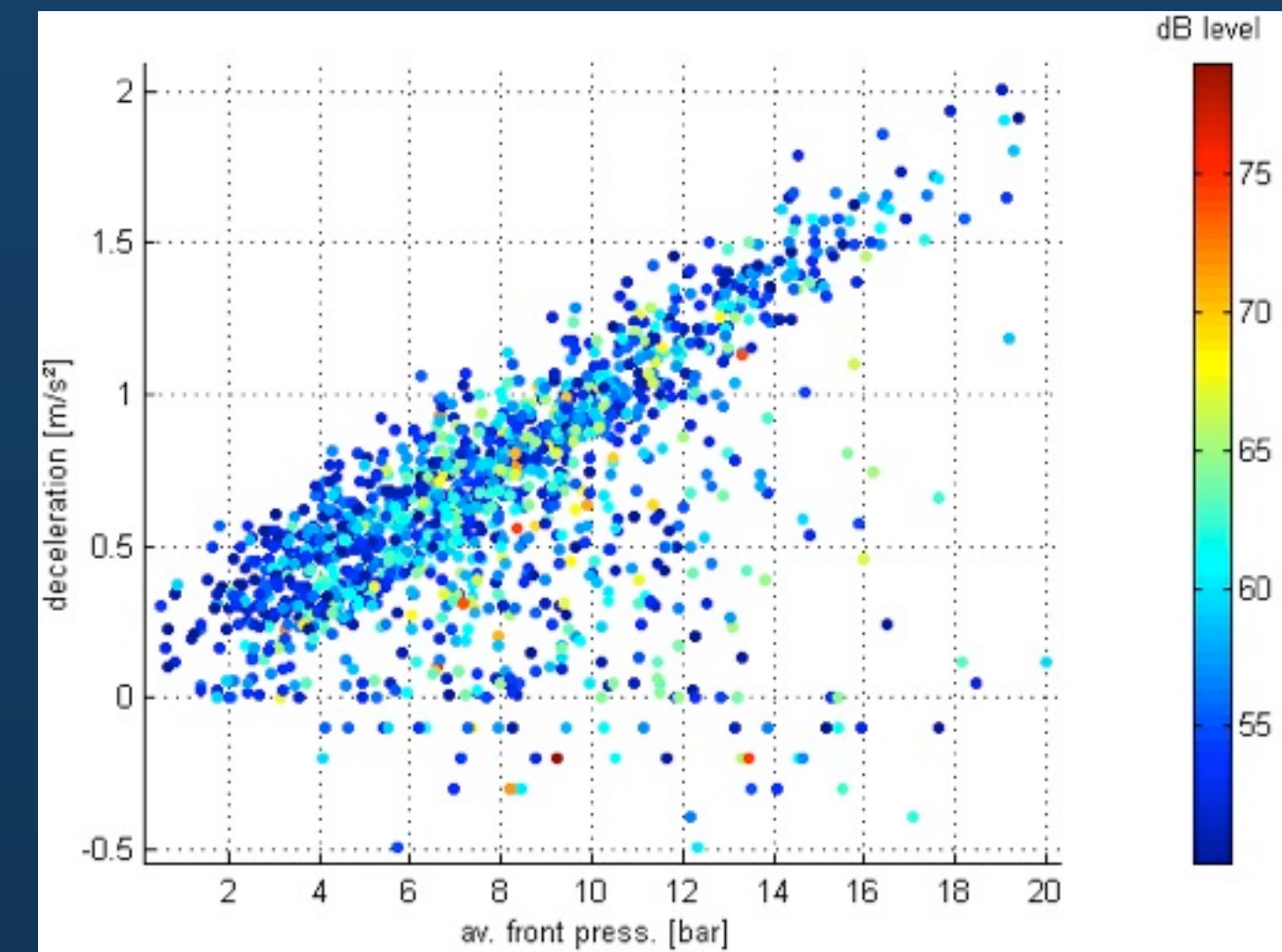
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses



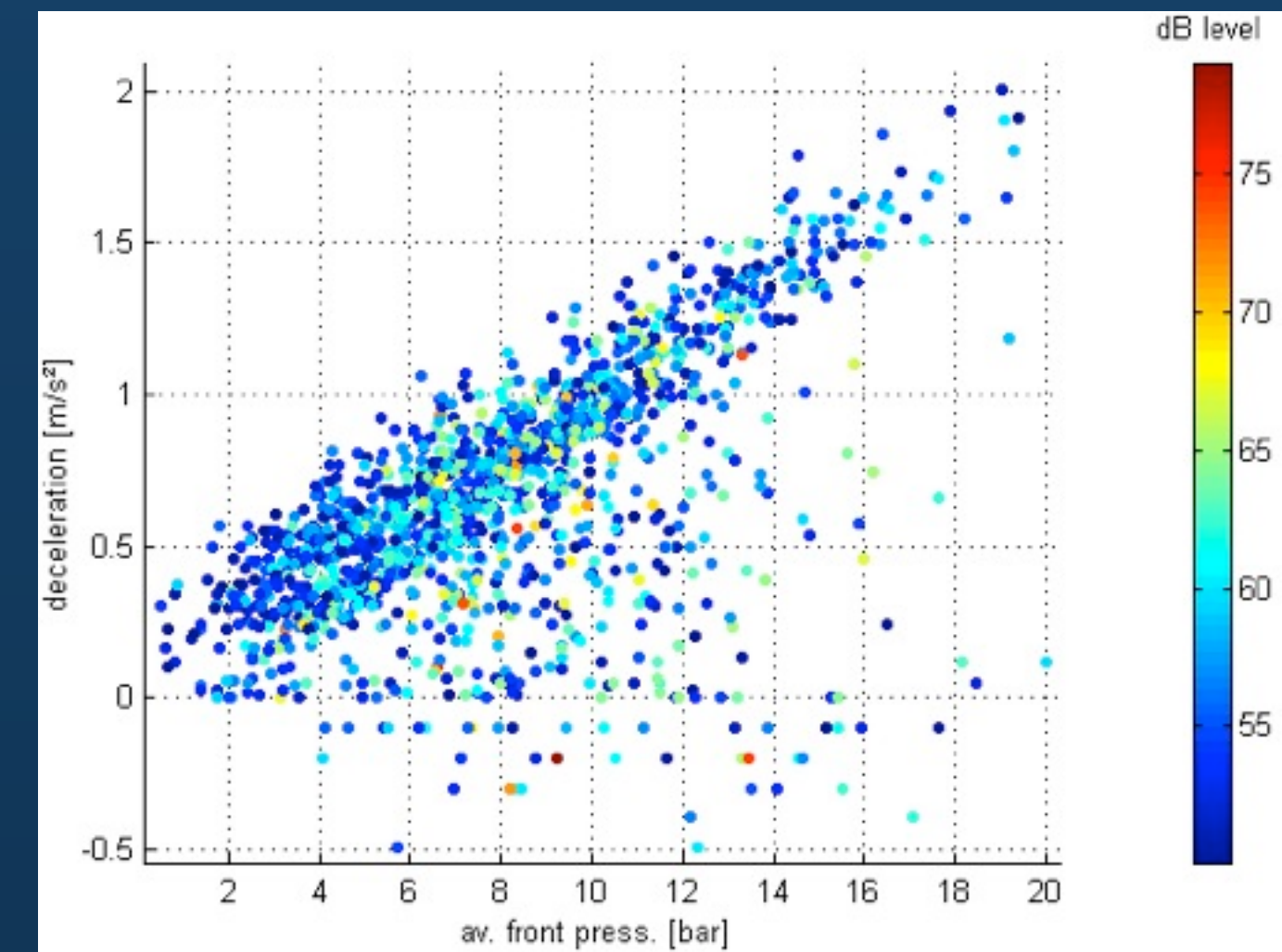
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses
- ▶ Site problems: what goes wrong at the same time?



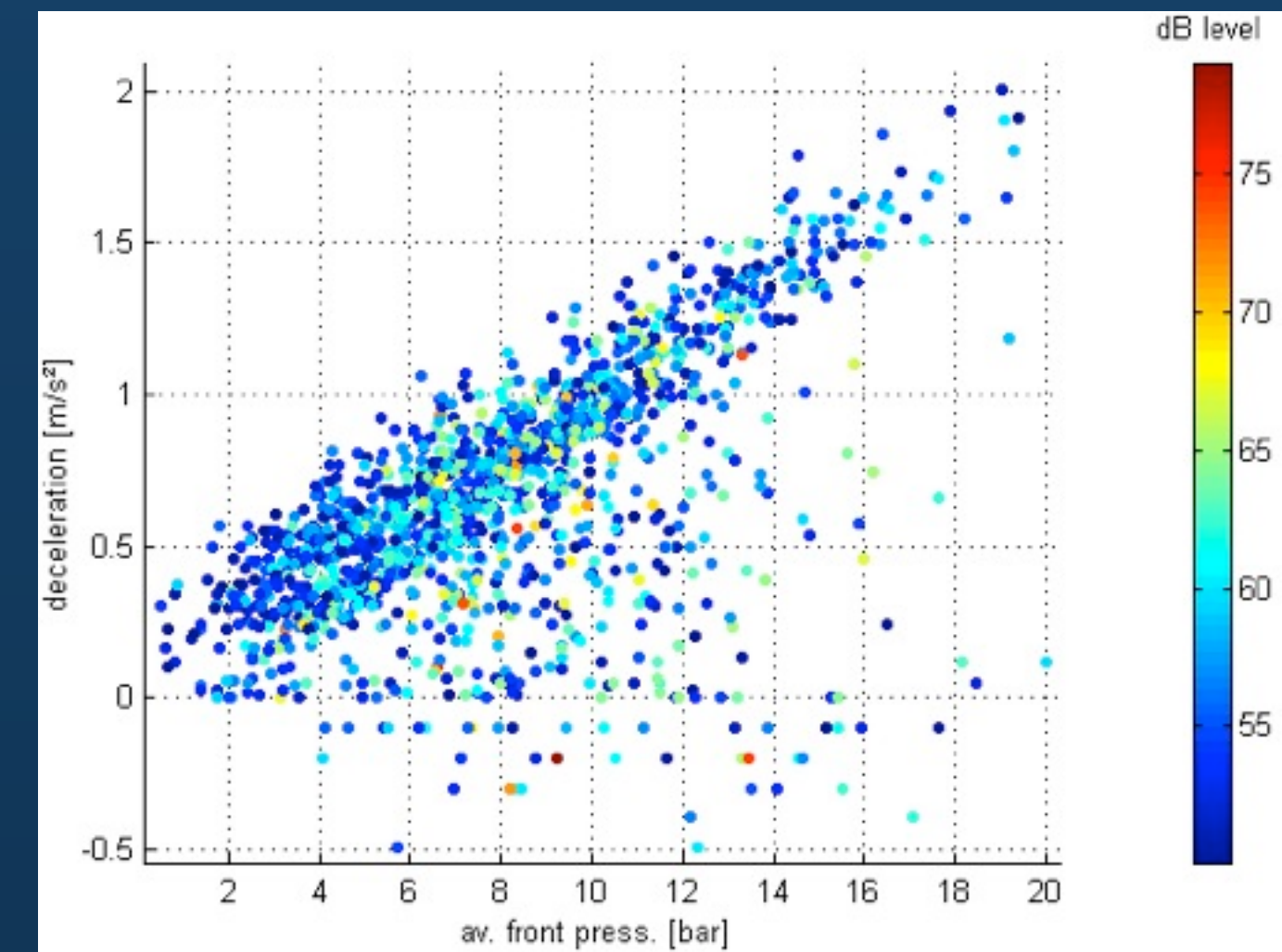
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses
- ▶ Site problems: what goes wrong at the same time?
- ▶ Which features get users hooked?



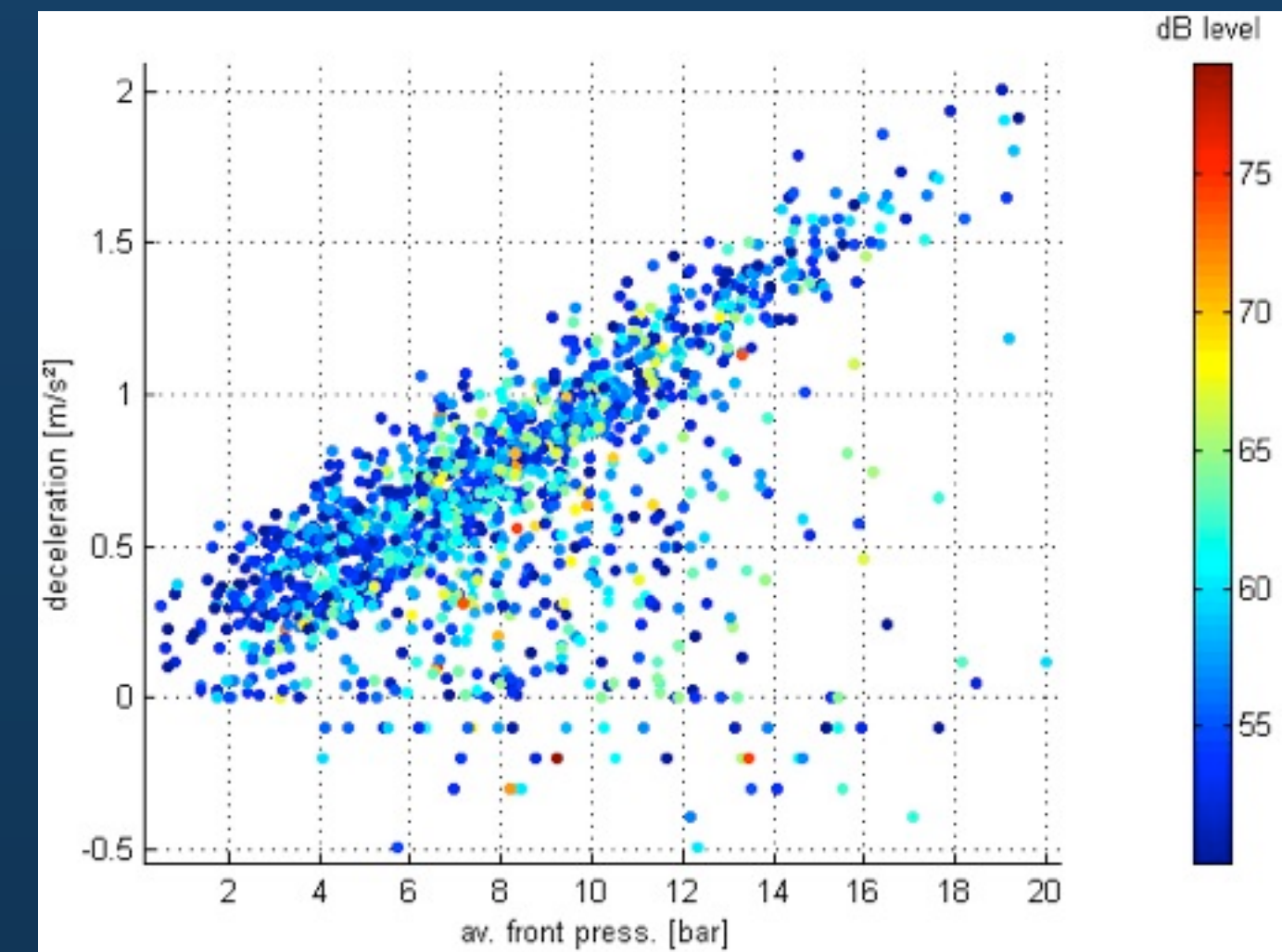
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses
- ▶ Site problems: what goes wrong at the same time?
- ▶ Which features get users hooked?
- ▶ Which features do successful users use often?



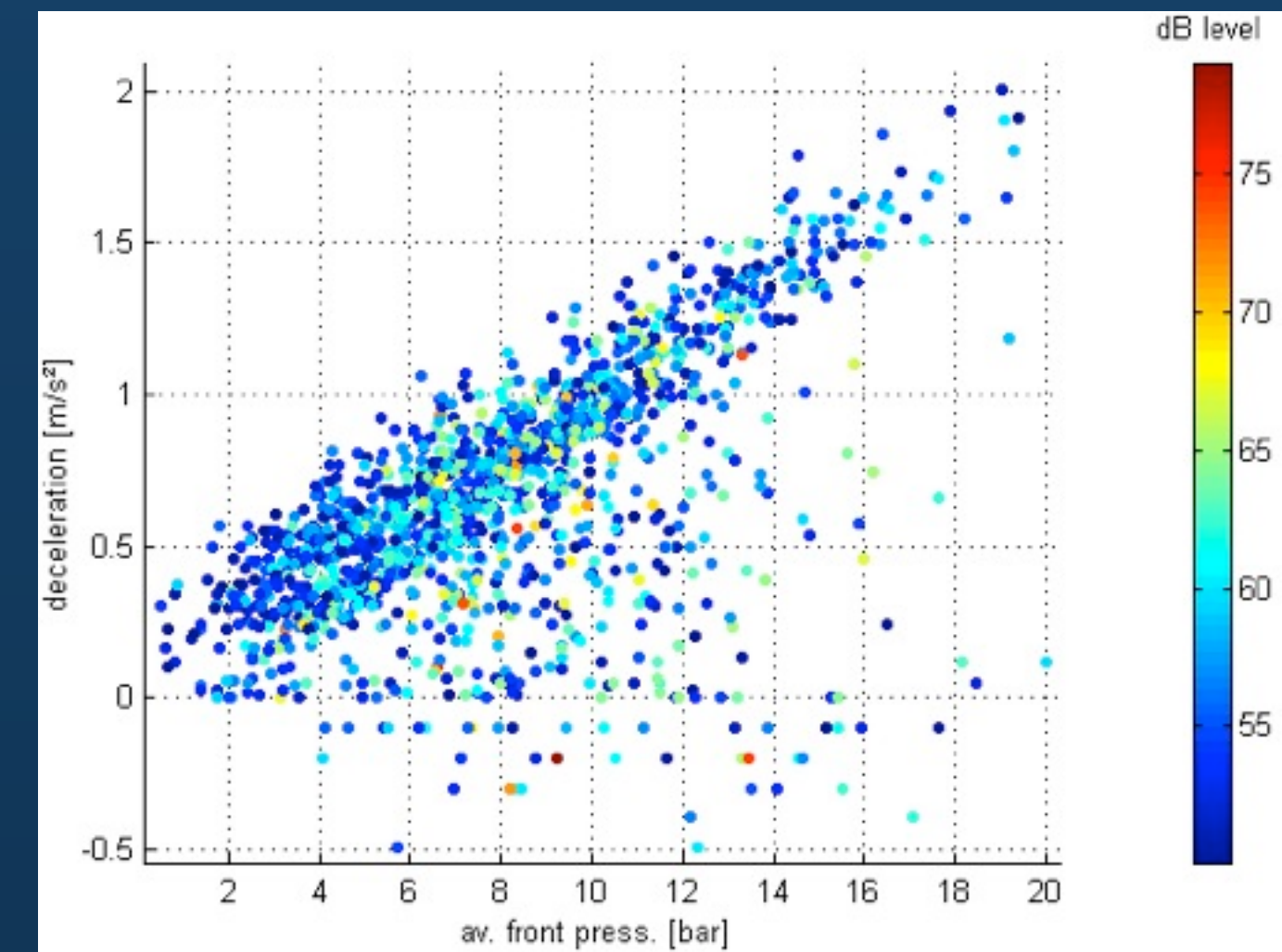
Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses
- ▶ Site problems: what goes wrong at the same time?
- ▶ Which features get users hooked?
- ▶ Which features do successful users use often?
- ▶ Search corrections, search suggestions



Correlating Big Data

- ▶ **probabilities, covariance, influence**
- ▶ How does usage differ for mobile users?
- ▶ How about for users with 3rd party desktop clients?
- ▶ Cohort analyses
- ▶ Site problems: what goes wrong at the same time?
- ▶ Which features get users hooked?
- ▶ Which features do successful users use often?
- ▶ Search corrections, search suggestions
- ▶ A/B testing



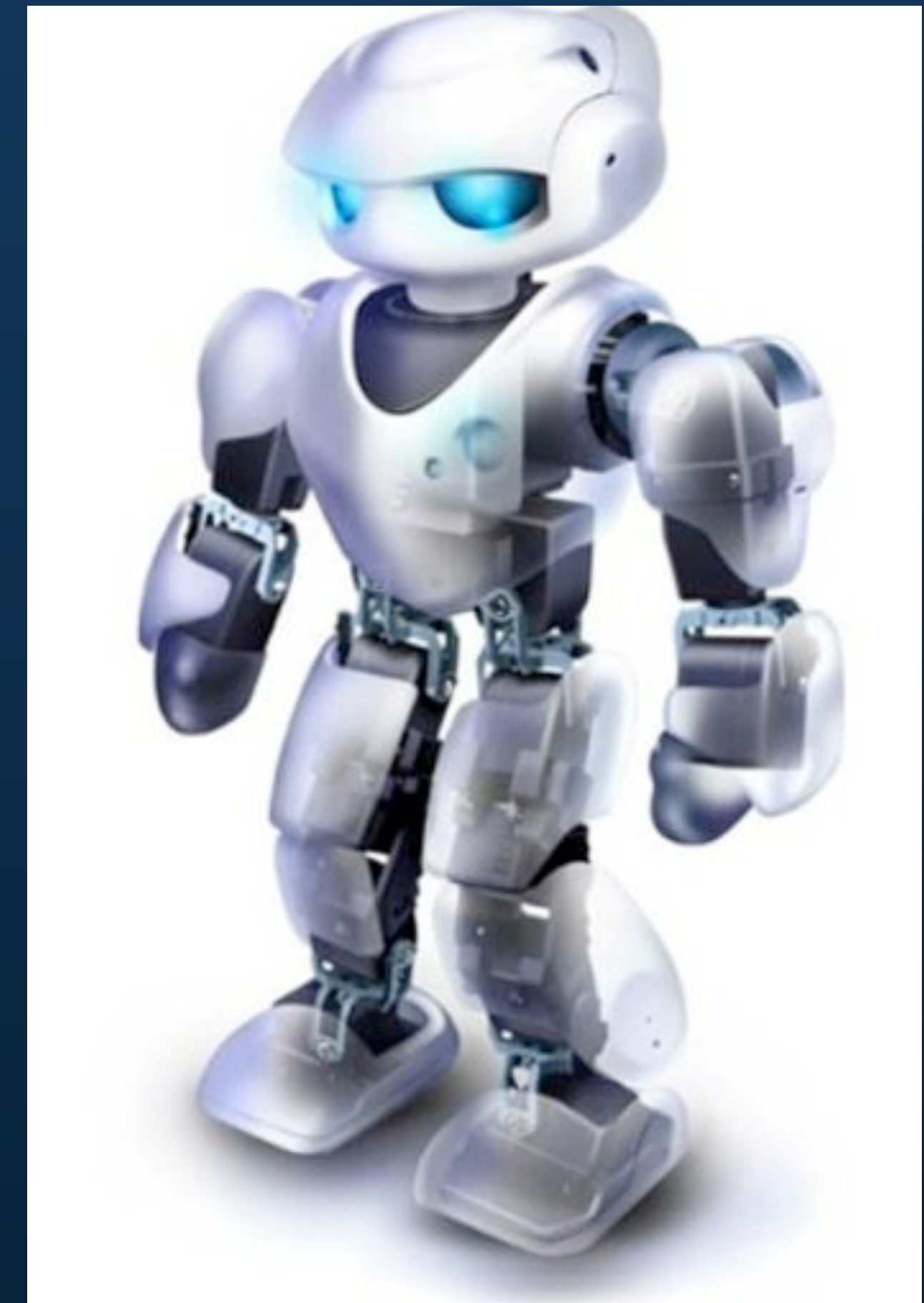
Correlating Big Data

- ▶ What is the correlation between users with registered phones and users that tweet?

```
correlation.pig
register piggybank.jar;
devices = load '$DEVICES' using com.twitter.twadood.pig.storage.LzoDevicesLoader() as
(device_id: long, user_id: long, ...);
tweets = load '$TWEETS' using com.twitter.twadood.pig.storage.LzoStatusLoader() as
(tweet_id: long, user_id: long, text: chararray, ...);
tweet_user_ids = foreach tweets generate user_id;
tweets_grouped = group tweet_user_ids by user_id;
tweets_by_user = foreach tweets_grouped generate group as user_id, COUNT(tweet_user_ids) as count;
combined = cogroup devices by user_id, tweets_by_user by user_id;
summed = foreach combined generate user_id, SIZE(devices) as has_device,
(SIZE(tweets_by_user) == 0 ? 0 : SUM(tweets_by_user.count)) as tweet_count;
summed_grouped = group summed all;
covariance = foreach summed_grouped generate group,
COR(summed_grouped.has_device, summed_grouped.tweet_count);
```

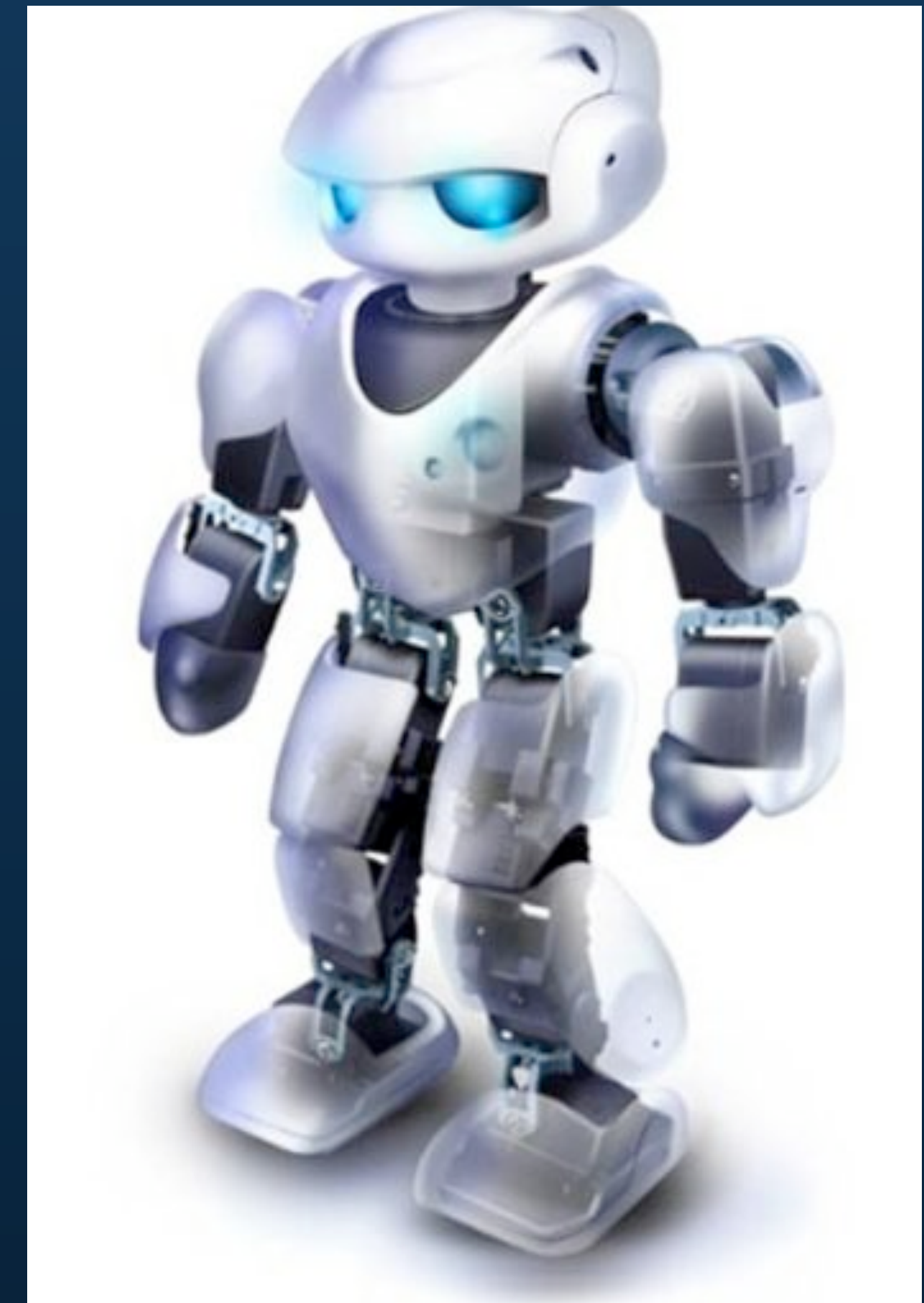
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?



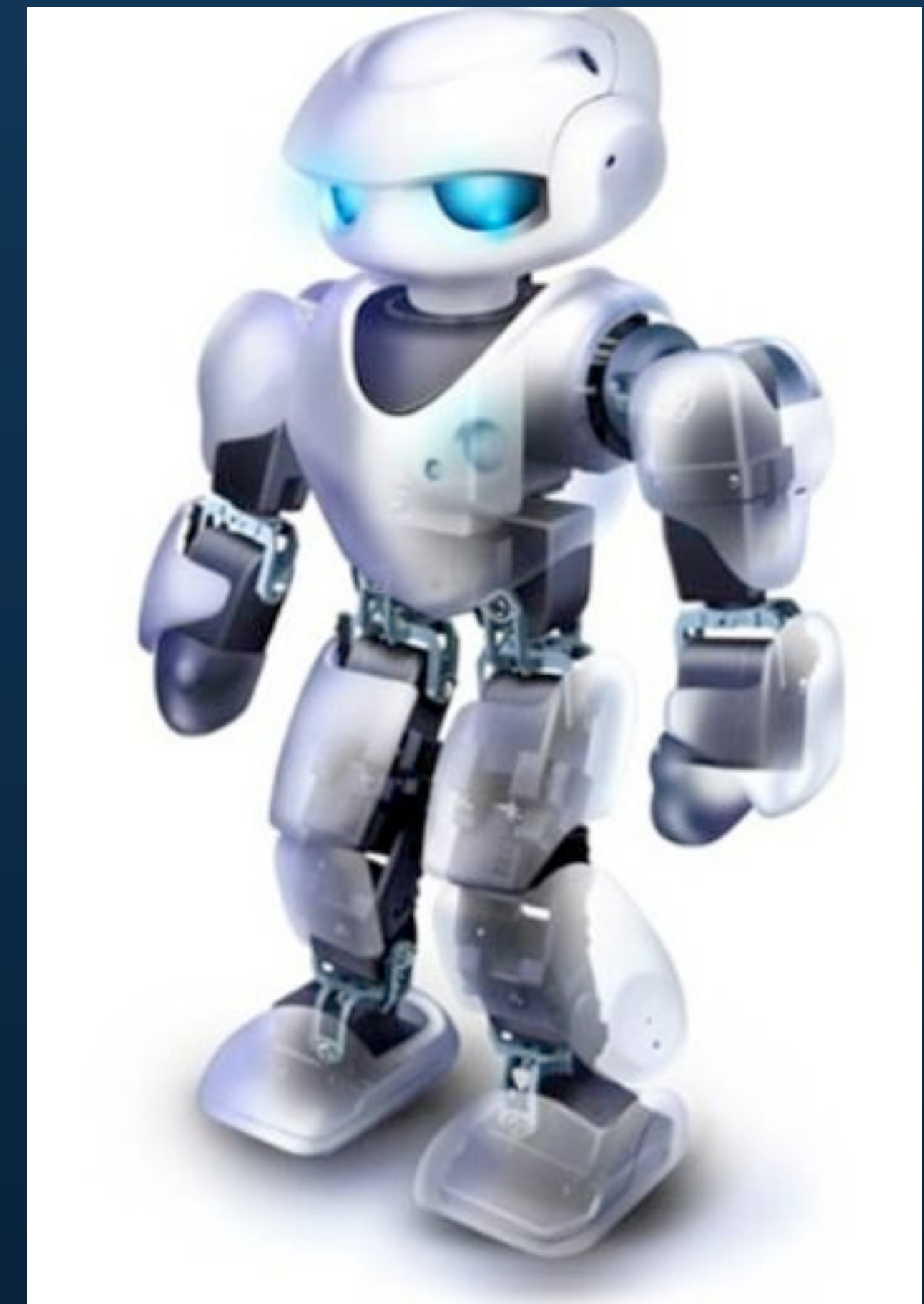
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?
- ▶ From the tweets of those they follow?



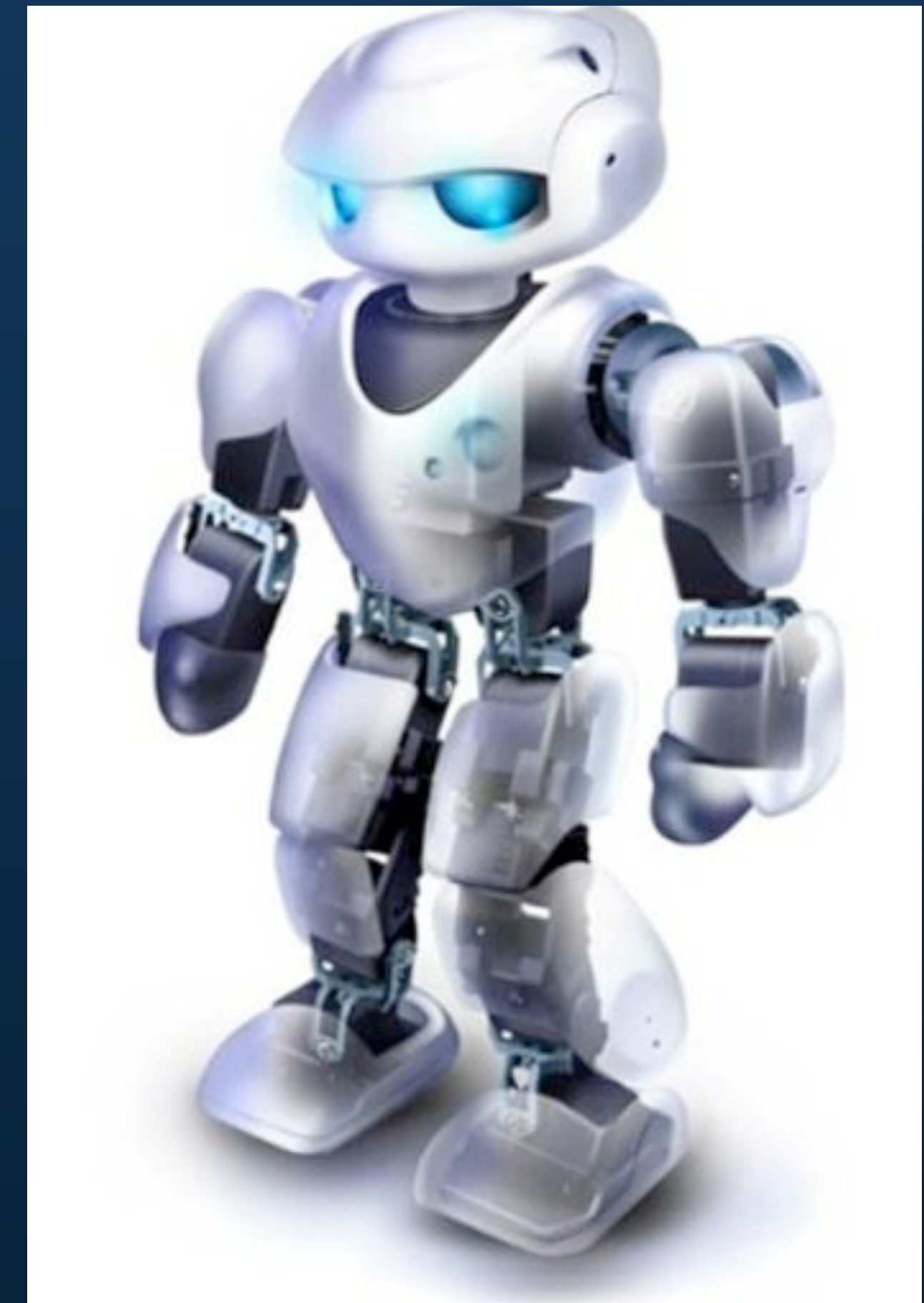
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?
- ▶ From the tweets of those they follow?
- ▶ From the tweets of their followers?



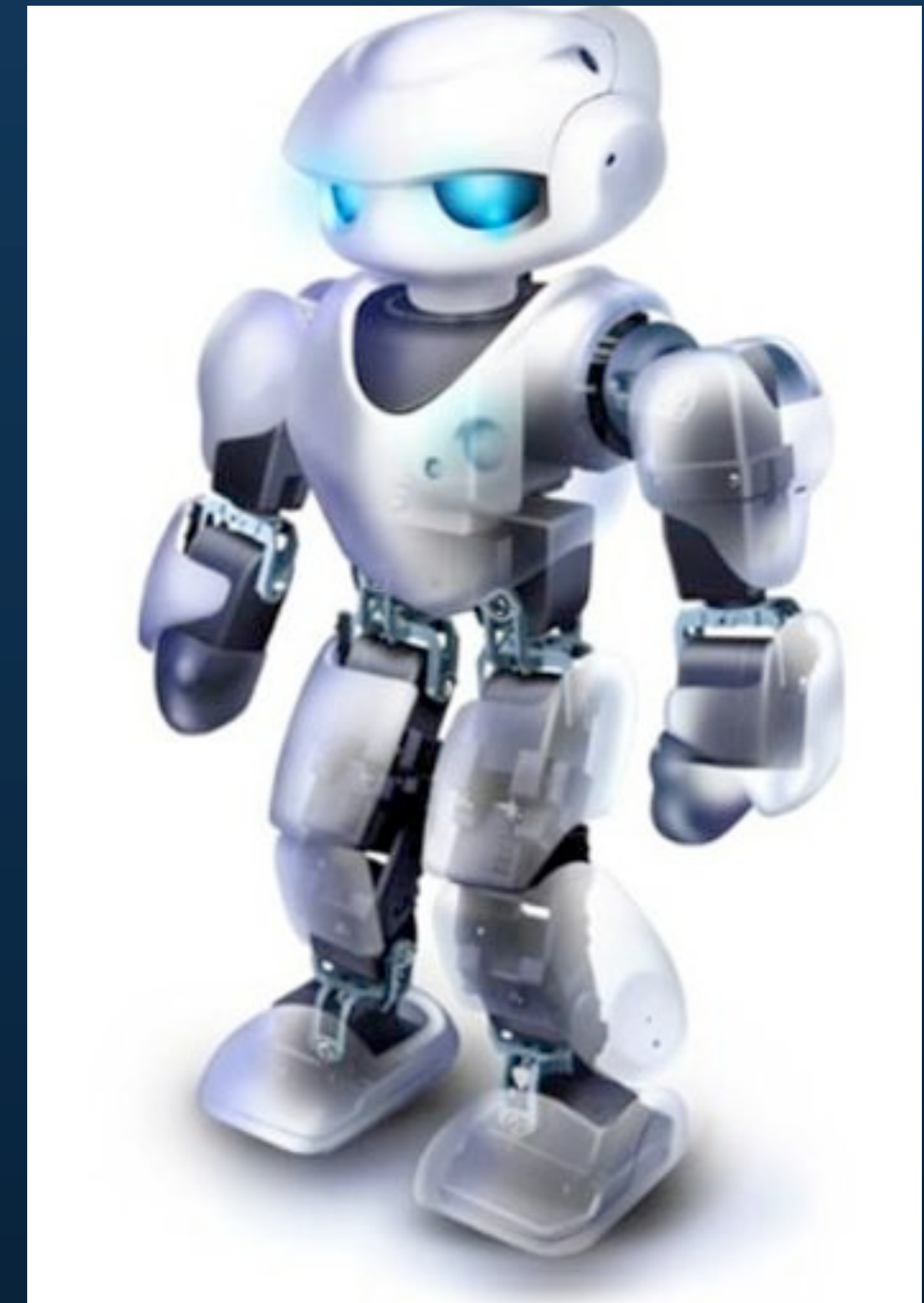
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?
- ▶ From the tweets of those they follow?
- ▶ From the tweets of their followers?
- ▶ From the ratio of followers/following?



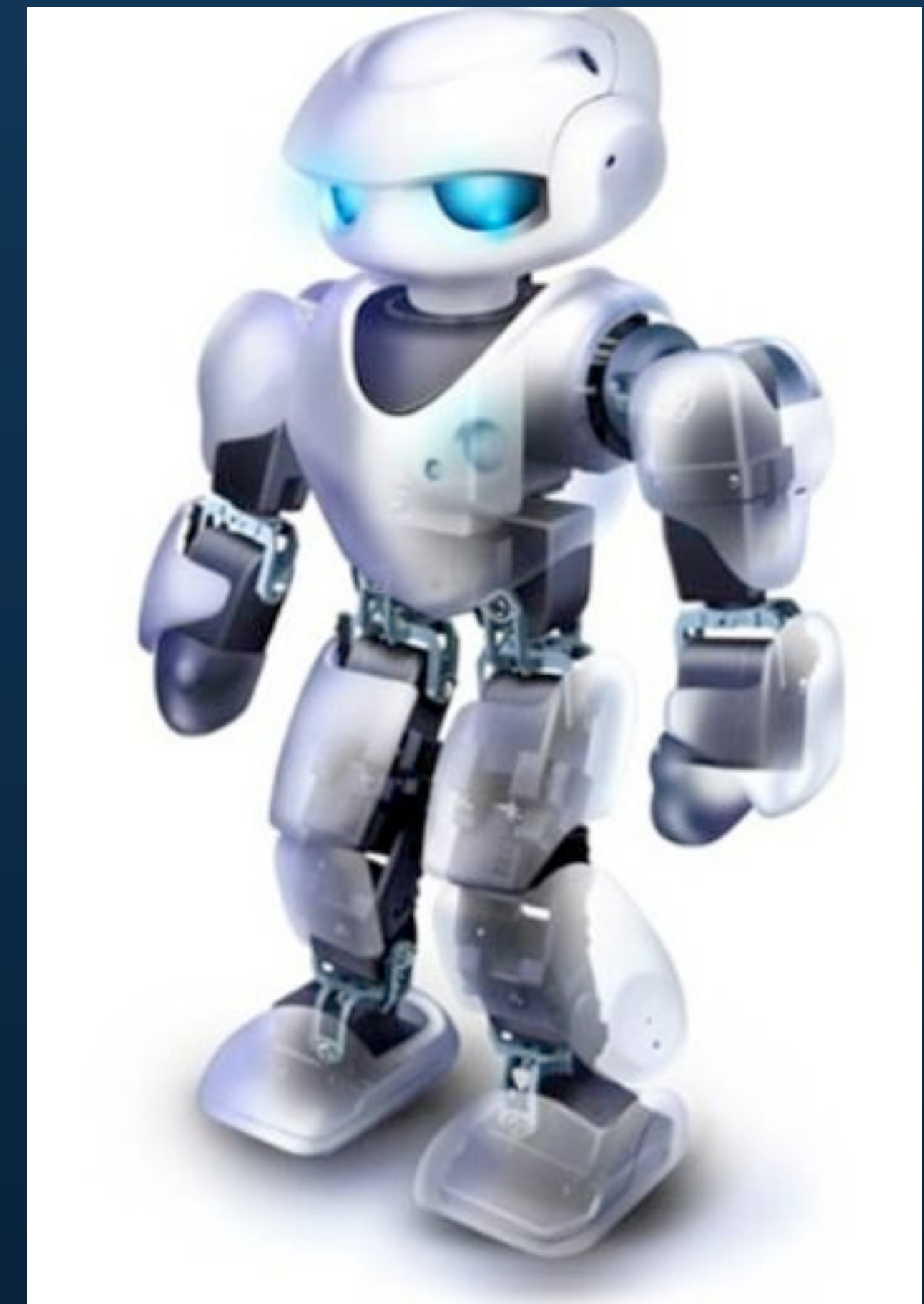
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?
- ▶ From the tweets of those they follow?
- ▶ From the tweets of their followers?
- ▶ From the ratio of followers/following?
- ▶ What graph structures lead to successful networks?



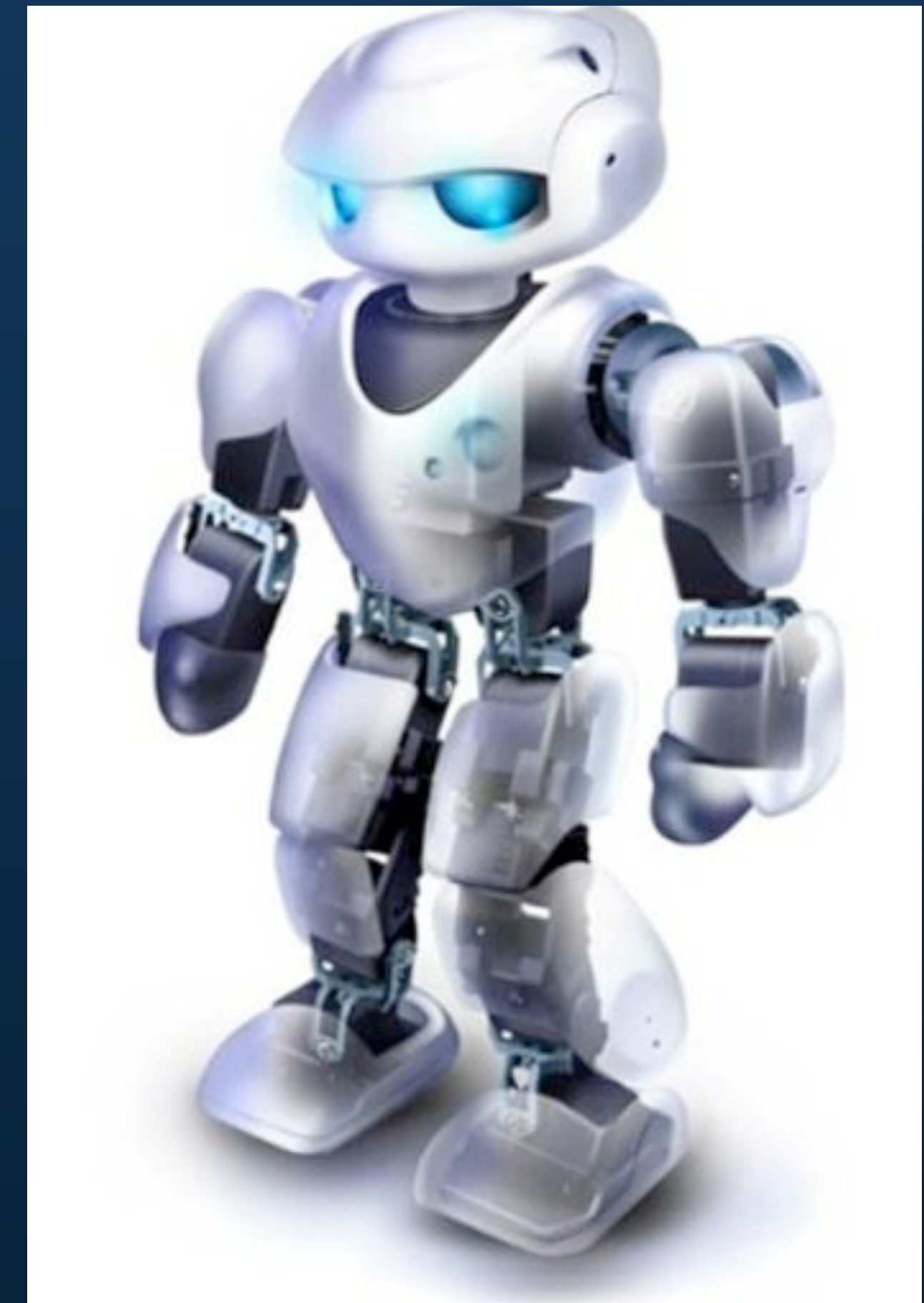
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ What can we tell about a user from their tweets?
- ▶ From the tweets of those they follow?
- ▶ From the tweets of their followers?
- ▶ From the ratio of followers/following?
- ▶ What graph structures lead to successful networks?
- ▶ User reputation



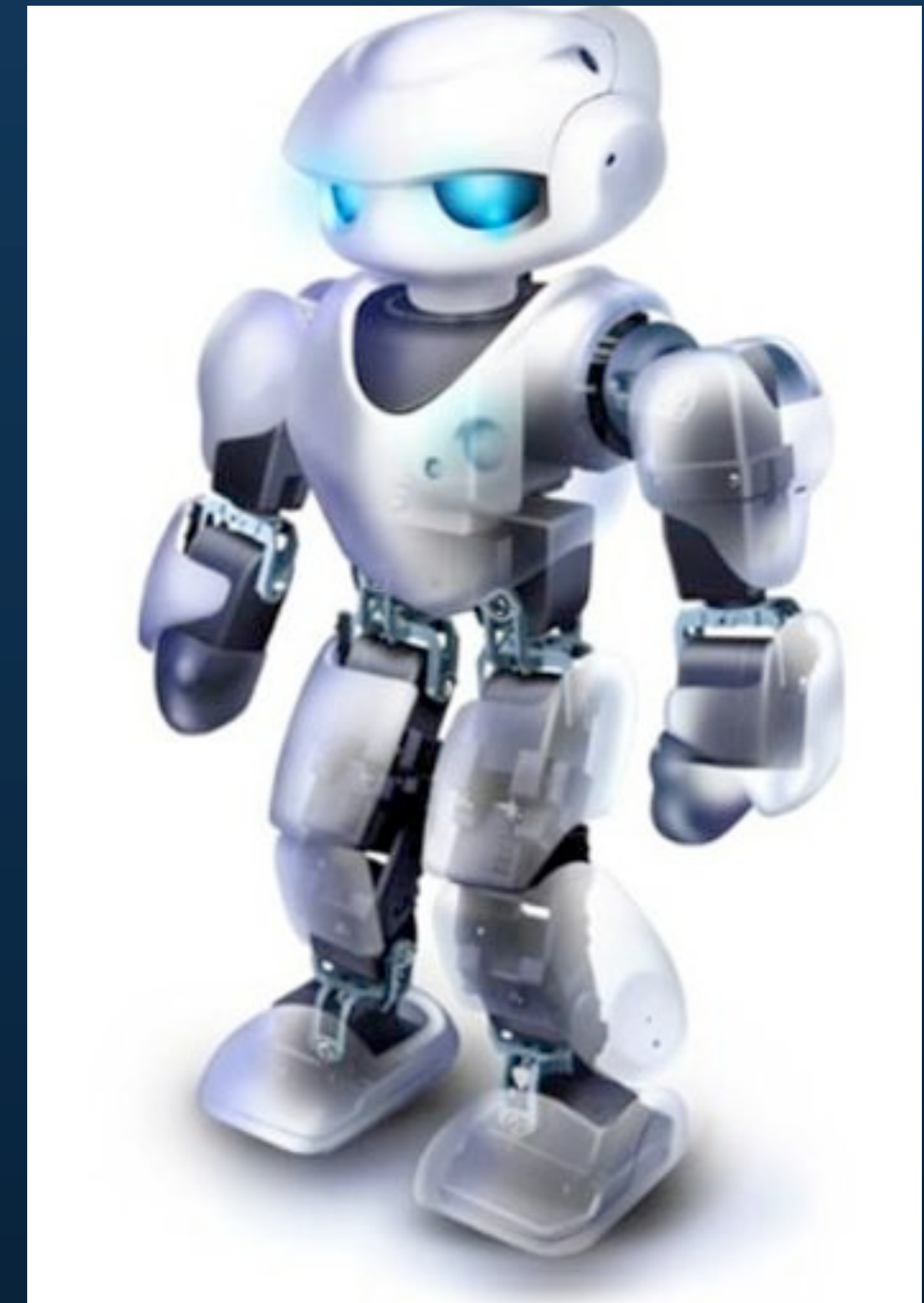
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis



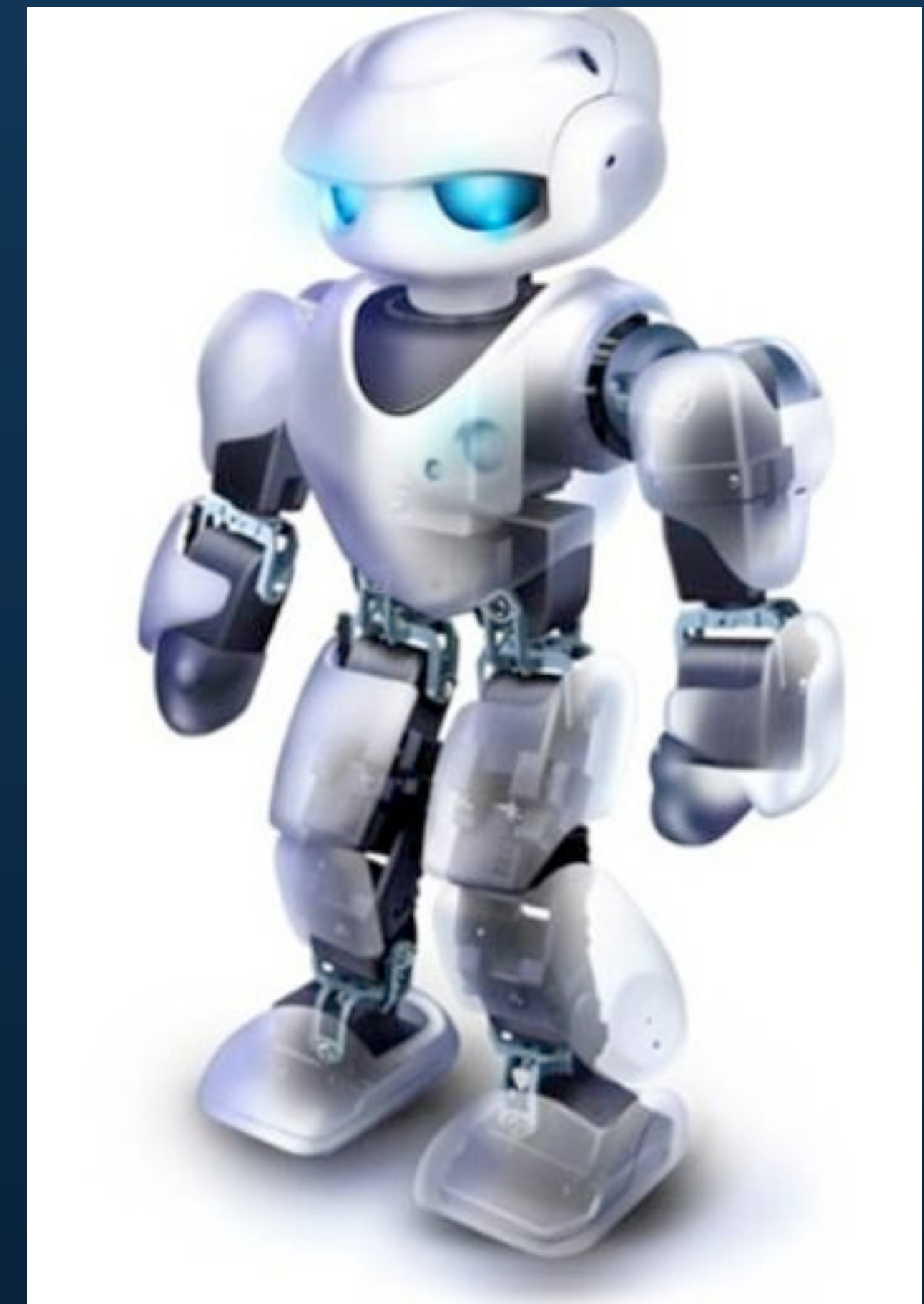
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?



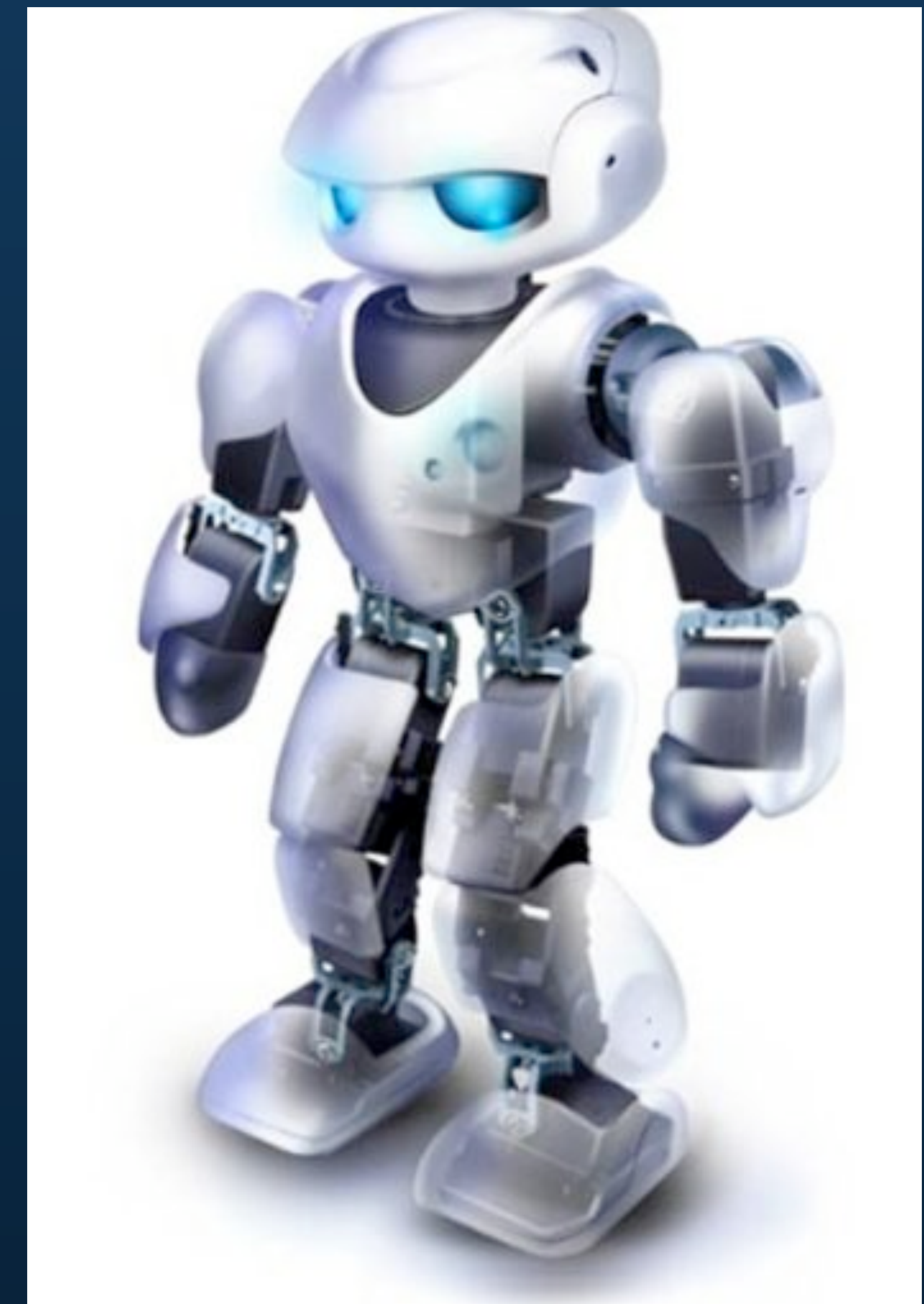
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?
- ▶ How deep is the corresponding retweet tree?



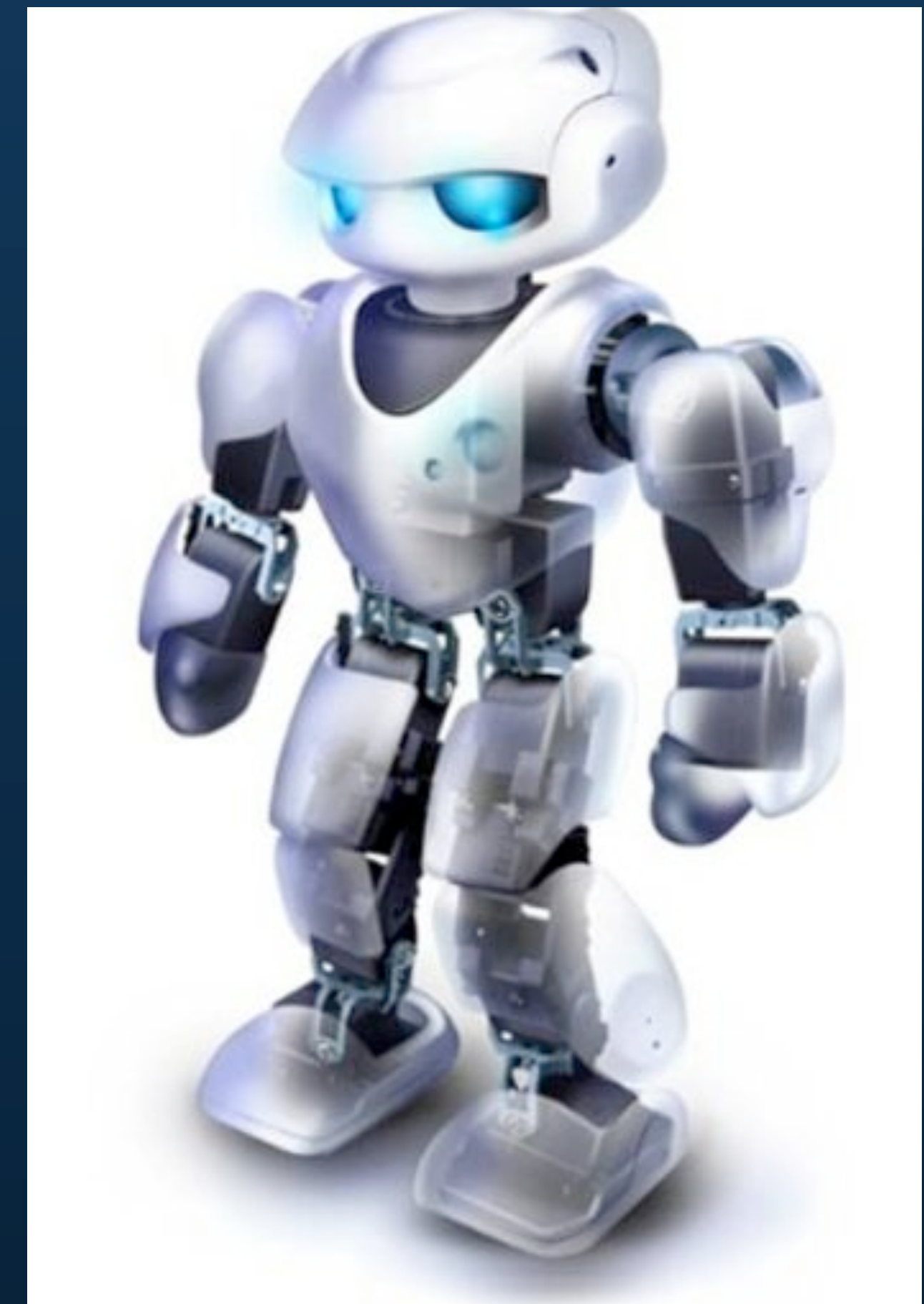
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?
- ▶ How deep is the corresponding retweet tree?
- ▶ Long-term duplicate detection



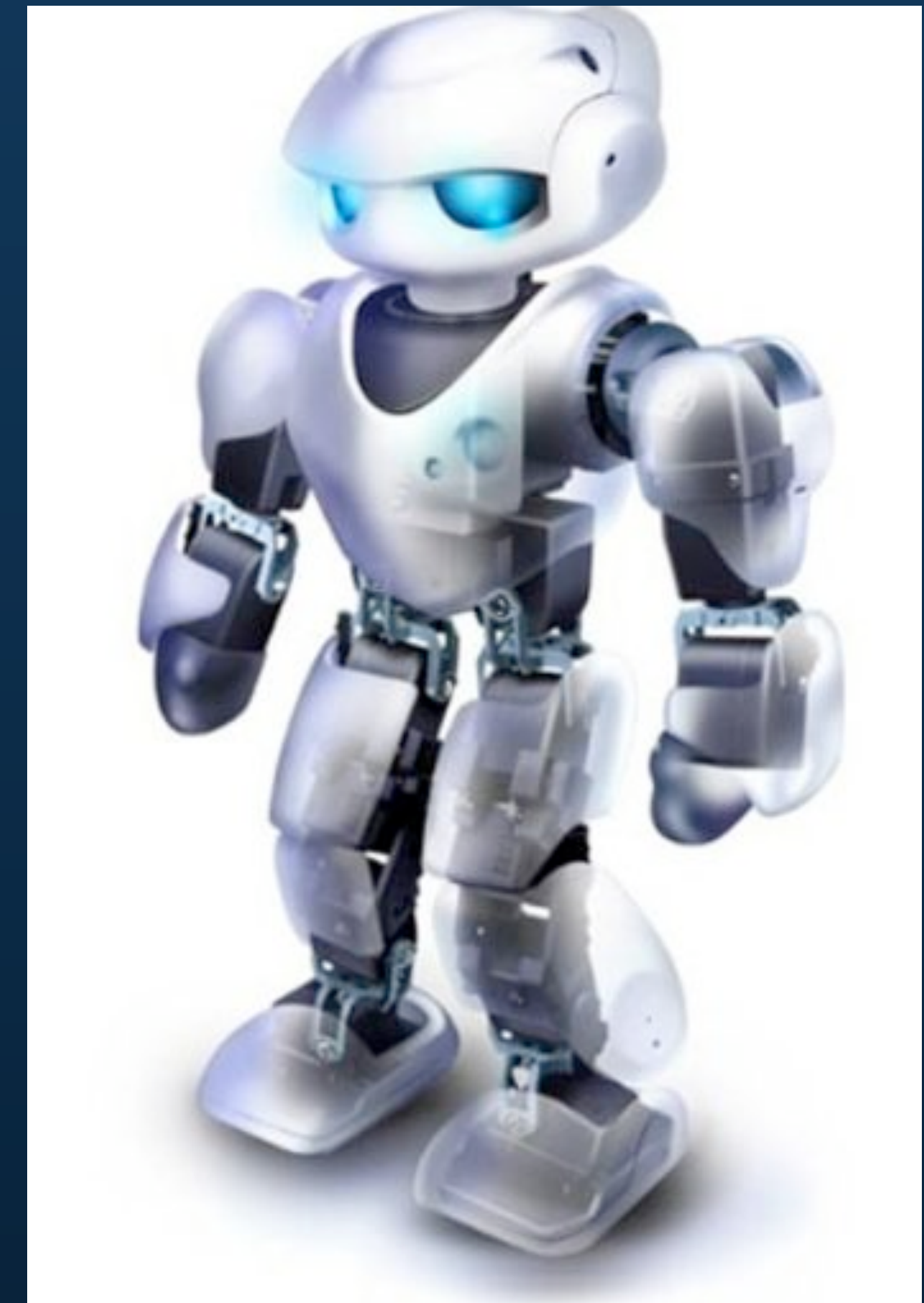
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?
- ▶ How deep is the corresponding retweet tree?
- ▶ Long-term duplicate detection
- ▶ Machine learning



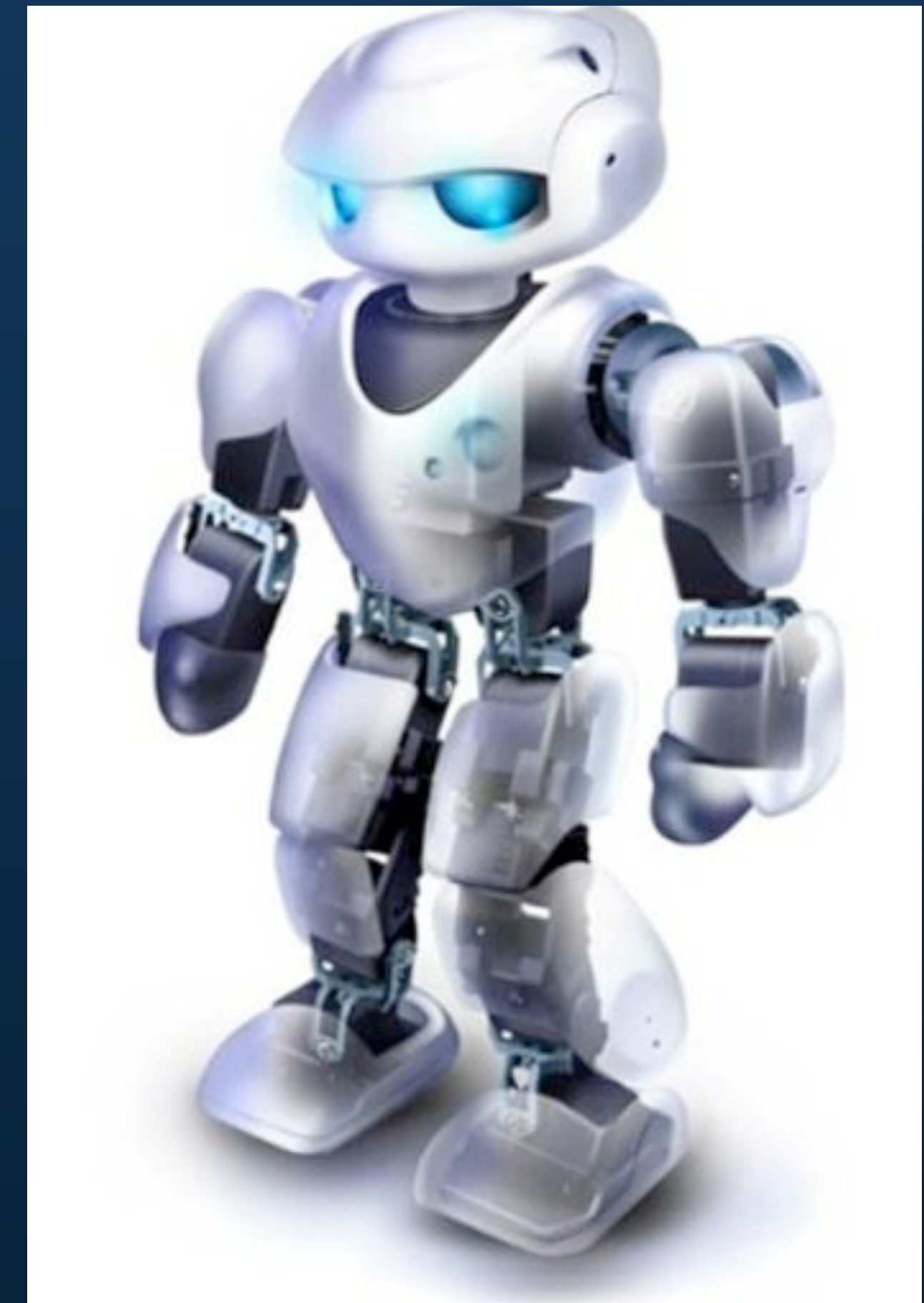
Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?
- ▶ How deep is the corresponding retweet tree?
- ▶ Long-term duplicate detection
- ▶ Machine learning
- ▶ Language detection



Research on Big Data

- ▶ **prediction, graph analysis, natural language**
- ▶ Sentiment analysis
- ▶ What features get a tweet retweeted?
- ▶ How deep is the corresponding retweet tree?
- ▶ Long-term duplicate detection
- ▶ Machine learning
- ▶ Language detection
- ▶ ... the list goes on.



Research on Big Data

- ▶ How well can we detect bots and other non-human tweeters?

```
bot_probability.pig

raw_data = load '$INPUT_FILES' using com.twitter.twadood.pig.storage.LzoStatusLoader() as
(id: long,
 user_id: long,
 text: chararray,
 created_at: chararray,
 ...);

projected = foreach raw_data generate user_id, text;

grouped_by_user = group projected by user_id parallel $PARALLEL;

bot_probabilities = foreach grouped_by_user generate group as user,
com.twitter.twadood.pig.piggybank.UserBotProbability(projected.text) as bot_probability;

store bot_probabilities into 'bot_test.tsv' using PigStorage('\t');
```


Introduction

- ▶ Hadoop Overview
- ▶ Why Pig?
- ▶ Evolution of Data Processing at Twitter
- ▶ Pig for Counting
- ▶ Pig for Correlating
- ▶ Pig for Research and Data Mining
- ▶ **Conclusions and Next Steps**

Why Hadoop?

- ▶ Data is growing rapidly; need horizontally scalable computation
- ▶ Fault tolerant; gracefully handles machine failure
- ▶ ... but writing MapReduce jobs in Java is harder than it should be.

Why Pig?

- ▶ Pig makes Hadoop accessible
- ▶ Pig chains together complex job flows
- ▶ User-defined functions are first class citizens
- ▶ Vibrant OS community, dedicated team at Y! improving it daily
- ▶ **At Twitter, Pig helps us understand our business faster.**

Questions?

Follow me at
twitter.com/kevinweil

- ▶ If this sounded interesting to you -- that's because it is. And we're hiring.

twitterTM