RAD Lab
UC Berkeley

Statistics Driven Workload Modeling for the Cloud

Archana Ganapathi, Yanpei Chen
Armando Fox, Randy Katz, David Patterson

SMDB 2010

Data analytics are moving to the cloud

Cloud computing ➔ economy of scale

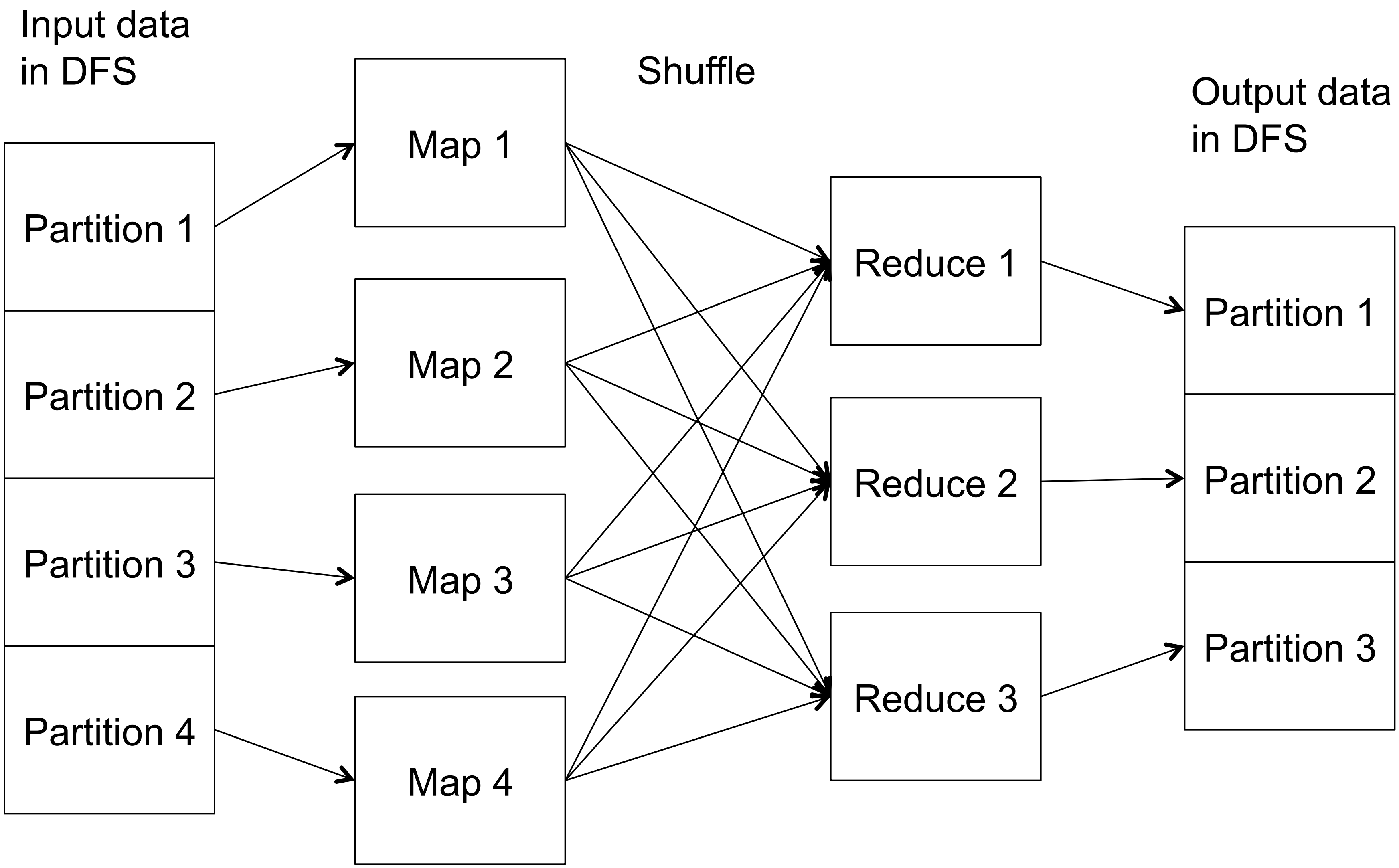
MapReduce for warehousing/analytics in the cloud

New challenges:

– Heterogeneous HW/SW/configuration/infrastructure
– Large variation in workload/software
– Ability to change resource consumption elastically

We present a partial solution

Brief MapReduce overview



Scheduling? Design trade-offs? Plan for the future?

Existing approaches:

- System simulation
- Benchmarks
- Trace replay
- Hardware/VM statistics

Our approach:

1. Predict performance on a fixed configuration
2. Workload synthesis & evaluation across configurations

Performance prediction via SML

Desirable properties:

✓ Predict job execution time and resource requirements in a single model

✓ Works equally well for SQL-like and traditional MapReduce jobs

✓ Generalizable across different hardware, software, configurations, etc.

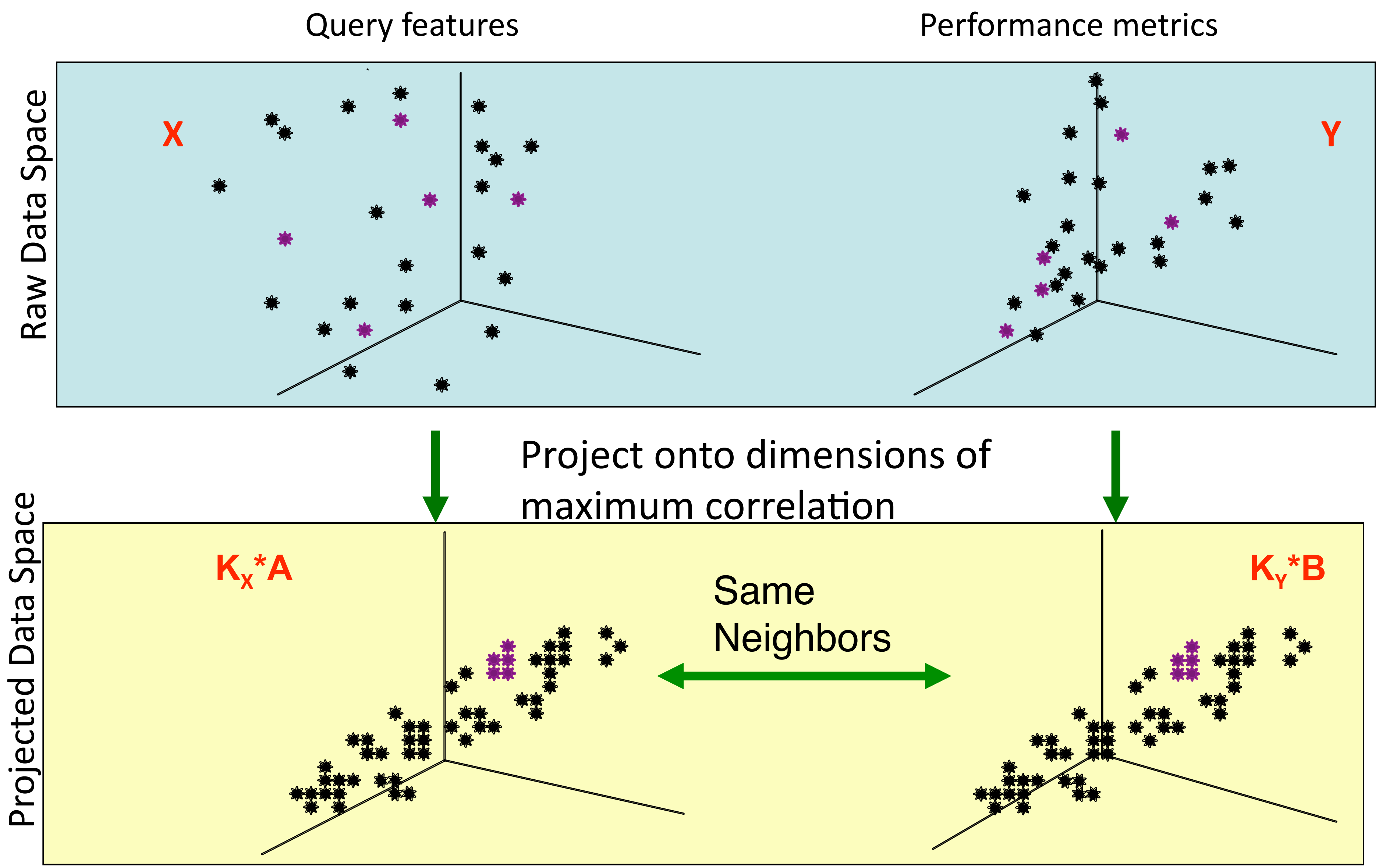
We've done this before …

A. Ganapathi, H. Kuno, U. Dayal , J. Wiener, A. Fox , M. Jordan , D. Patterson, **Predicting Multiple Performance Metrics for Queries: Better Decisions Enabled by Machine Learning,** ICDE 2009.

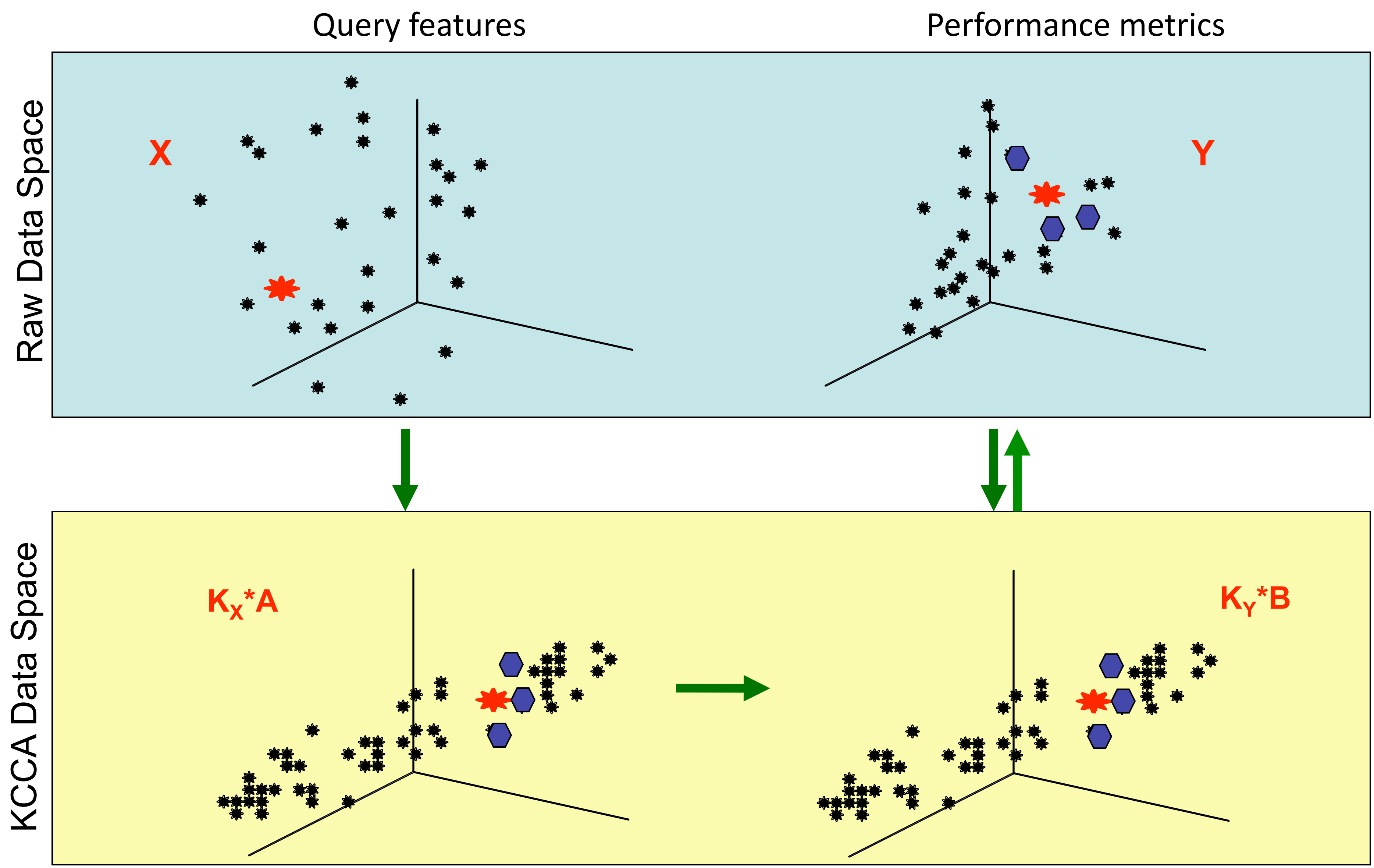
Silver bullet:

Kernel Canonical Correlation Analysis (KCCA)

High level description of KCCA



Prediction using KCCA model



Our particular experiment

Hadoop & Hive: open-source data analytics SQL interface

Production Hadoop Deployment at well-known social network

- Multi-user environment
- Nodes 1-300: 16 GB memory, 5 map slots, 5 reduce slots
- Nodes 301-600: 8 GB memory, 5 map slots, 0 reduce slots

Training Set of 5000 Hive queries

Test Set of 1000 *unseen* Hive queries

SQL features predict poorly

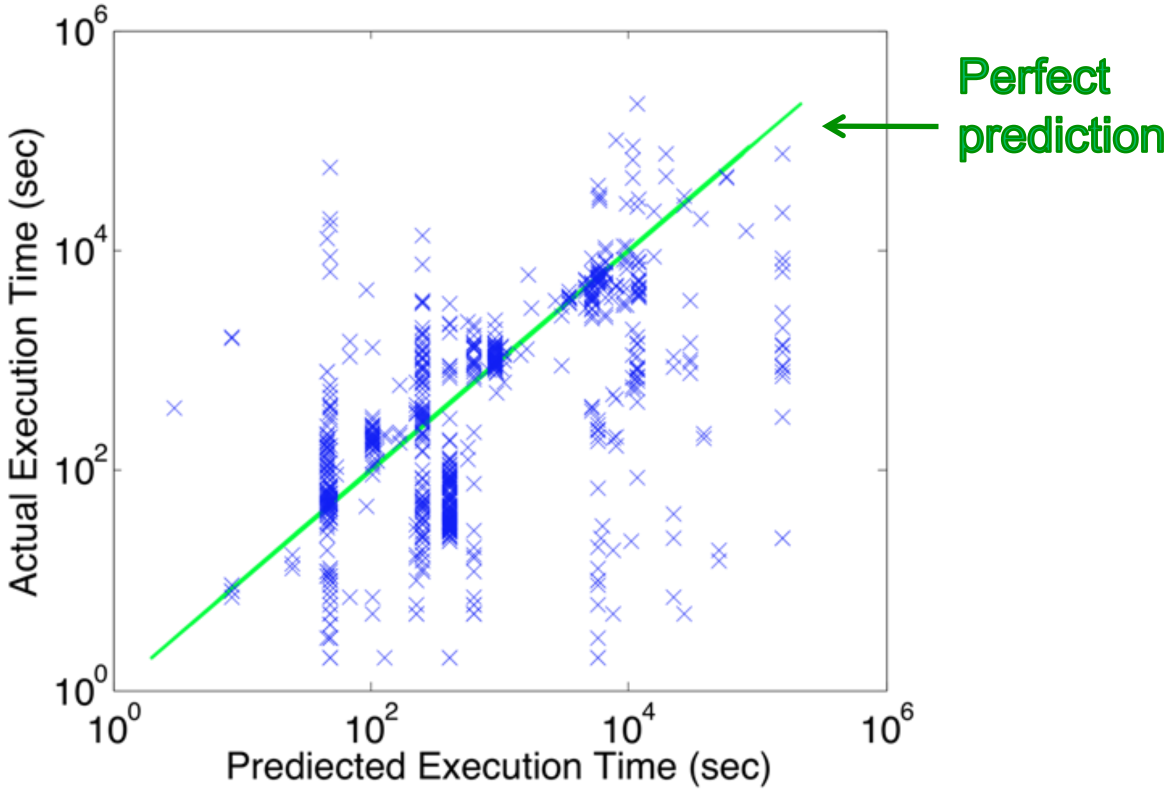
Query plan features:

Count of

- Create Table
- Filter
- Forward
- Group By
- Join
- Move
- Reduce Output

System behavior metrics:

- Total Execution Time
- Map Time
- Reduce Time
- Map Output Bytes
- Local/HDFS Bytes Written



MapReduce features predict very well

Query plan features:

Job config parameters

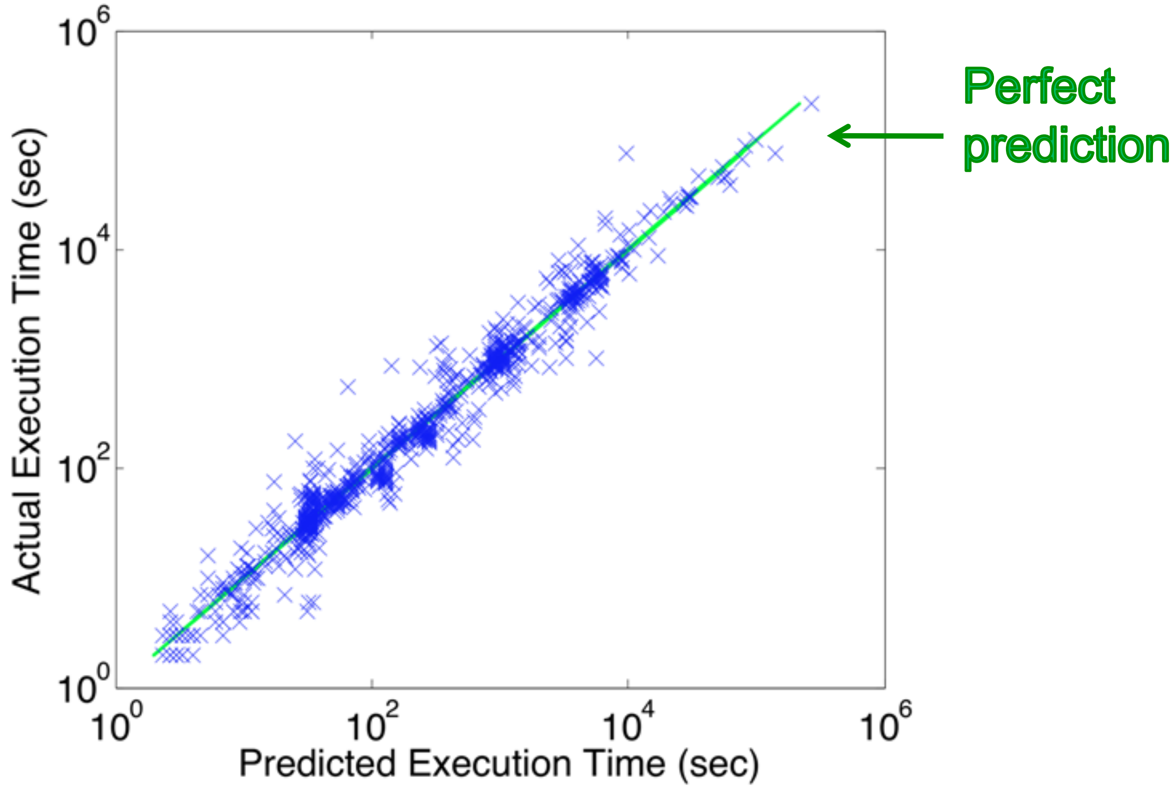
- Number of Maps
- Number of Reduces

Data characteristics

- Map Input Bytes
- Local/HDFS Bytes Read

System behavior metrics:

- Total Execution Time
- Map Time
- Reduce Time
- Map Output Bytes
- Local/HDFS Bytes Written



Generalize to other MapReduce jobs

Query plan features:

Job config parameters

- Number of Maps
- Number of Reduces

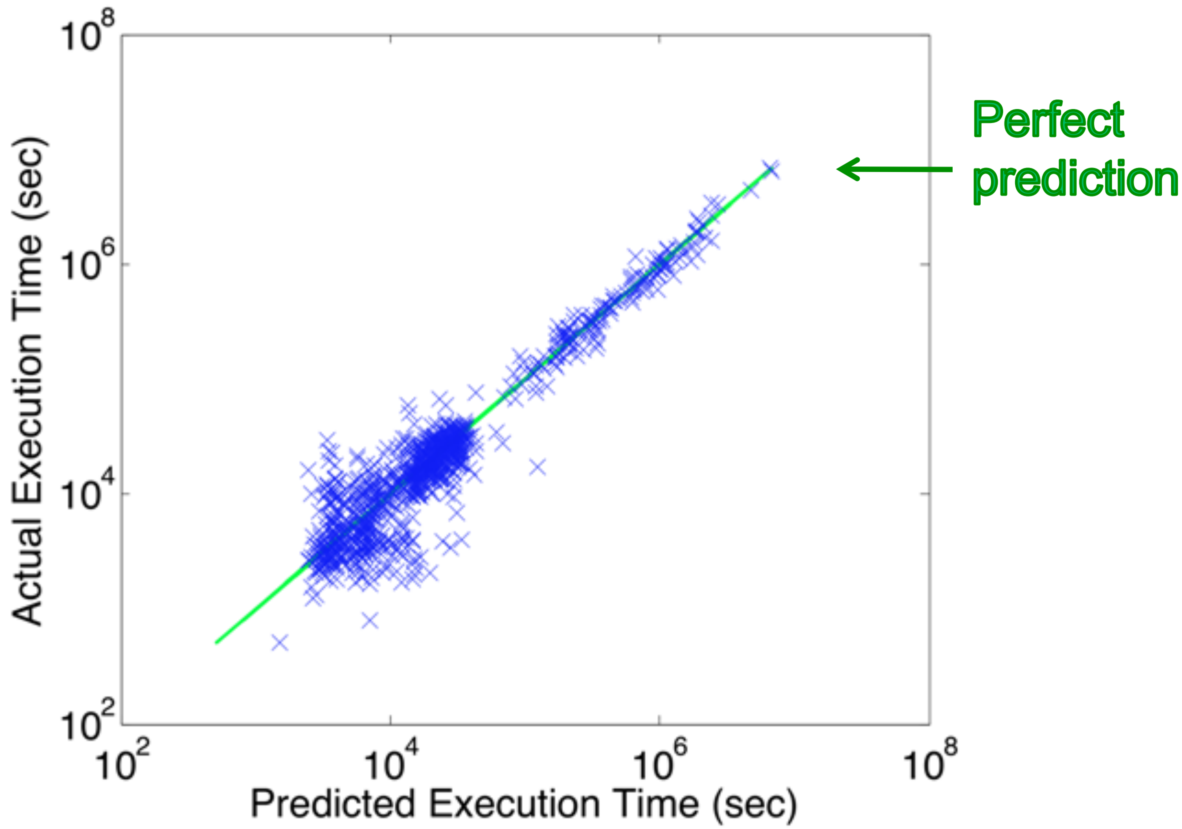
Data characteristics

- Map Input Bytes
- Local/HDFS Bytes Read

System behavior metrics:

- Total Execution Time
- Map Time
- Reduce Time
- Map Output Bytes
- Local/HDFS Bytes Written

E.g. Prediction for Extract-Transform-Load (ETL) jobs:



How do we use this tool?

- Given a particular setup, KCCA can predict performance
  - What if set up changes?
- Workload generator to ask “what-if” questions
  - Ideally, generalizes across hw/sw/config.
- KCCA identifies workload features that affect performance
  - Genesis for the MapReduce workload generator

Workload Generator Design

1. Collect statistics from real traces for these jobs features:
   – Inter-job arrival time
   – Per-job input data size
   – Per-job input:shuffle:output data ratio

2. Create synthetic workload that mimics real job stream
   – Compute approximate distributions
   – Sample from distribution to construct workload

3. Replay with low overhead

Workload Generator Design

1. Collect statistics from real traces for these jobs features:
   - Inter-job arrival time
   - Per-job input data size
   - Per-job input:shuffle:output data ratio

2. Create synthetic workload that mimics real job stream
   - Compute approximate distributions
   - Sample from distribution to construct workload

3. Replay with low overhead

Design trade-offs:

- Why not direct replay? Burdened with design defects in original system
- Why ignore locality & data skew? Test varied placement/data schemes
- Why ignore compute? IO is usually the bottleneck, confidentiality issues

We have been using our prototype!

MapReduce energy efficiency

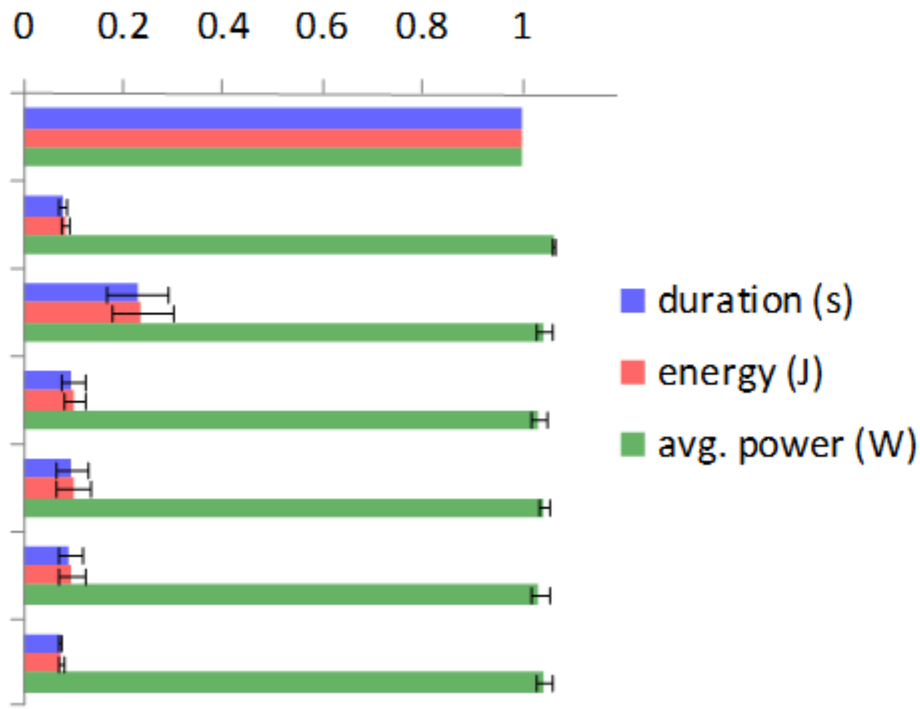
MapReduce schedulers

Multi-tenant resource provisioning and capacity planning

Preliminary look at latest results

Comparisons between different energy efficiency mechanisms

Batch?	Staggered launch	HDFS blocks	Tasks per node
No	n/a	64MB	4
Yes	No	64MB	4
Yes	No	64MB	8
Yes	No	80MB	4
Yes	No	80MB	6
Yes	No	80MB	8
Yes	Yes	80MB	8



Most of the improvement comes from batching

Reverse existing design priorities – idle energy > active energy

Are all traces like this?

Useful beyond MapReduce

Same approach works in systems other than MapReduce

- Bootstrap model with production traces
- Predictive framework to identify workload characteristics
- Workload description focused on application semantics
- Workload synthesis by directly sampling empirical traces

Only the workload and performance feature vectors are specific to MapReduce

Ongoing work: network storage, wind power, etc.

Lessons learned

- Performance predictions need to be multi-dimensional
- Effective prediction features focus on application semantics
- Workload characterization should also focus on application semantics
- Workload synthesis by sampling production traces is effective

RAD Lab
UC Berkeley

Questions?

archanag@eecs.berkeley.edu

ychen2@eecs.berkeley.edu