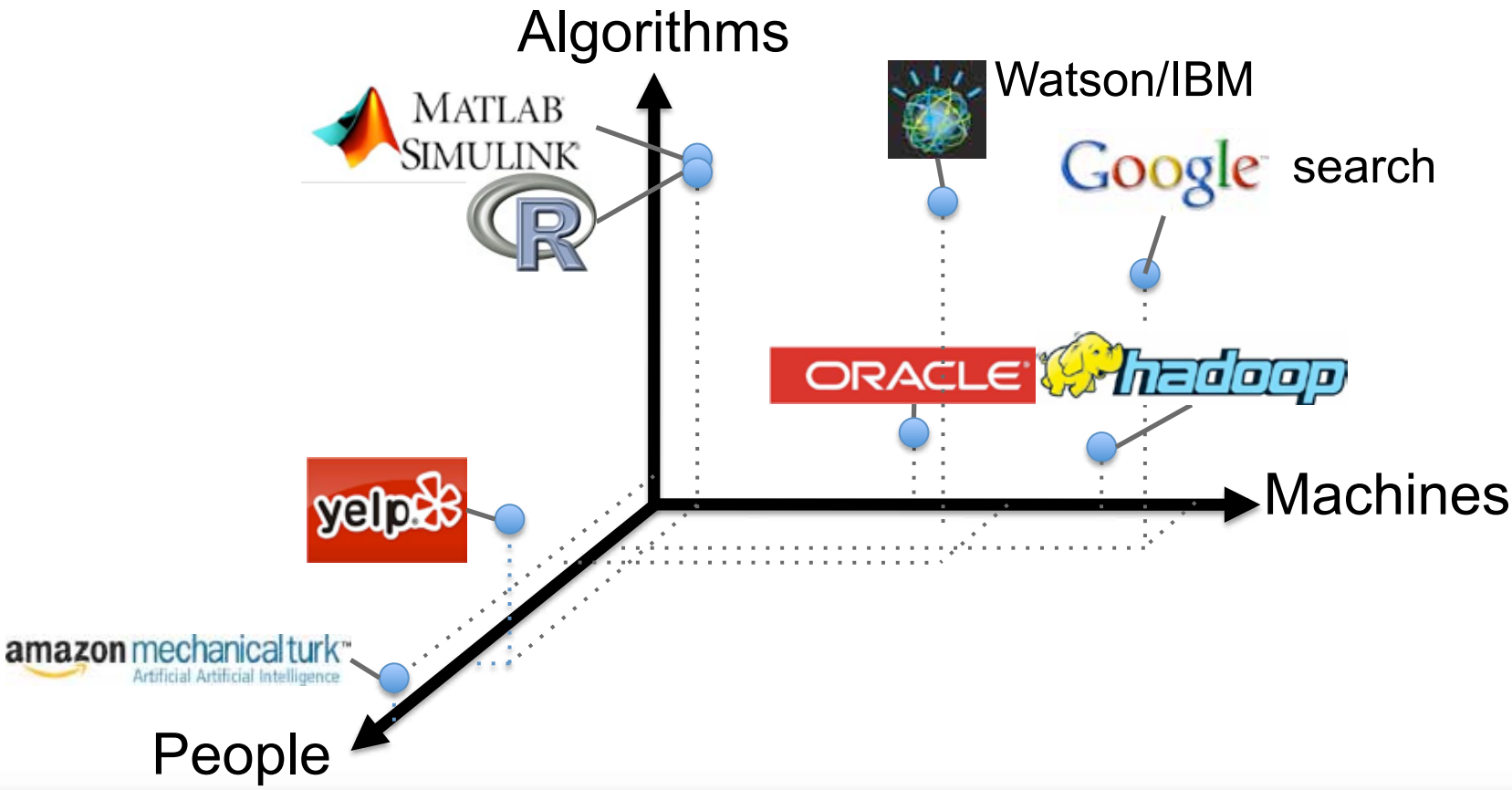
CrowdDB: Answering queries with crowdsourcing

Michael Franklin[+], Donald Kossmann*, **Tim Kraska[+]**
Sukriti Ramesh*, and Reynold Xin[+]
[+] UC Berkeley * ETH Zurich

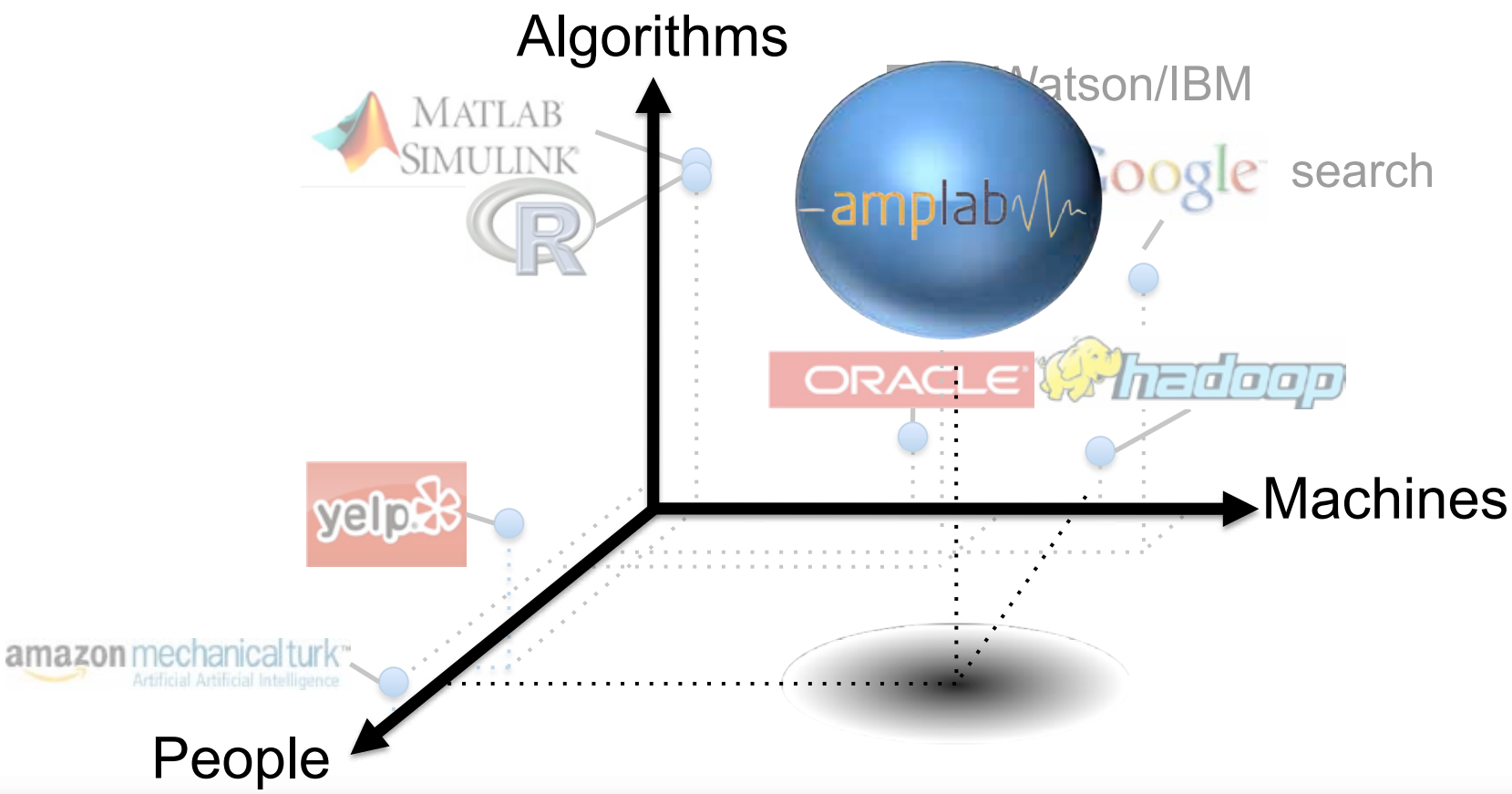
Algorithms, Machines, People

Today's apps: fixed point in solution space



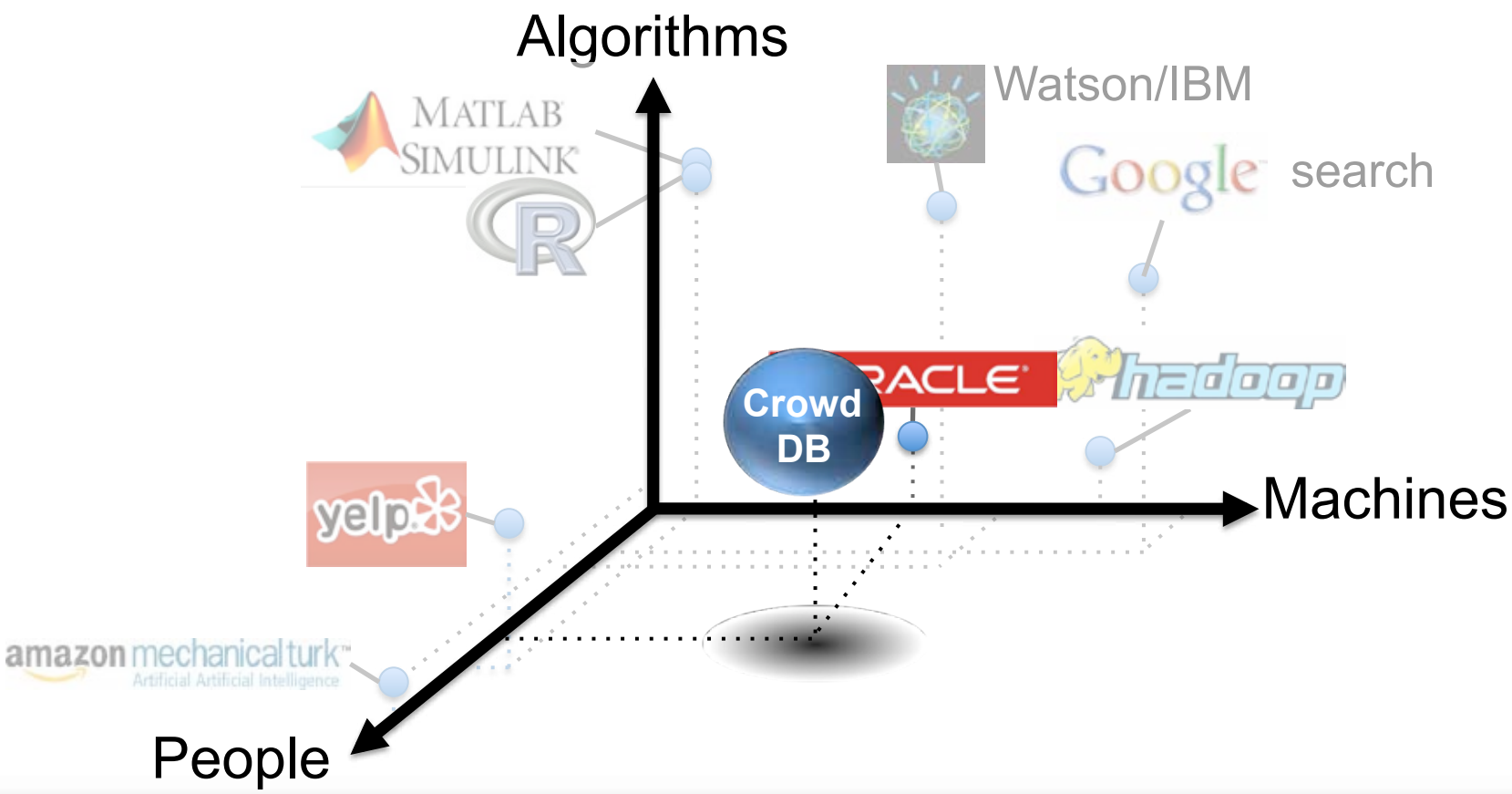
Need techniques to dynamically pick best operating point

The AMP Lab



Make sense of data at scale by tightly integrating algorithms, machines, and people

CrowdDB



Make sense of data at scale by tightly integrating algorithms, machines, and people

DB-hard Queries

Company_Name	Address	Market Cap
Google	Googleplex, Mtn. View CA	$210Bn
Intl. Business Machines	Armonk, NY	$200Bn
Microsoft	Redmond, WA	$250Bn

```
SELECT Market_Cap
From Companies
Where Company_Name = "IBM"
```

Number of Rows: 0

Problem:
Entity Resolution

DB-hard Queries

Company_Name	Address	Market Cap
Google	Googleplex, Mtn. View CA	$210Bn
Intl. Business Machines	Armonk, NY	$200Bn
Microsoft	Redmond, WA	$250Bn

```
SELECT Market_Cap
From Companies
Where Company_Name = "Apple"
```

Number of Rows: 0

Problem:
Closed World Assumption

amplab

DB-hard Queries

```
SELECT Top_1(Image)
From Pictures
Where Theme = “Business Success”
```

Number of Rows: 0

Problem:
Missing Intelligence

Easy Queries

Company_Name	Address	Market Cap
Google	Googleplex, Mtn. View CA	$210Bn
Intl. Business Machines	Armonk, NY	$200Bn
Microsoft	Redmond, WA	$250Bn

```
SELECT Market_Cap
From Companies
Where Company_Name = "IBM"
```

```
$200Bn
Number of Rows: 1
```


amplab

Pretty Easy Queries

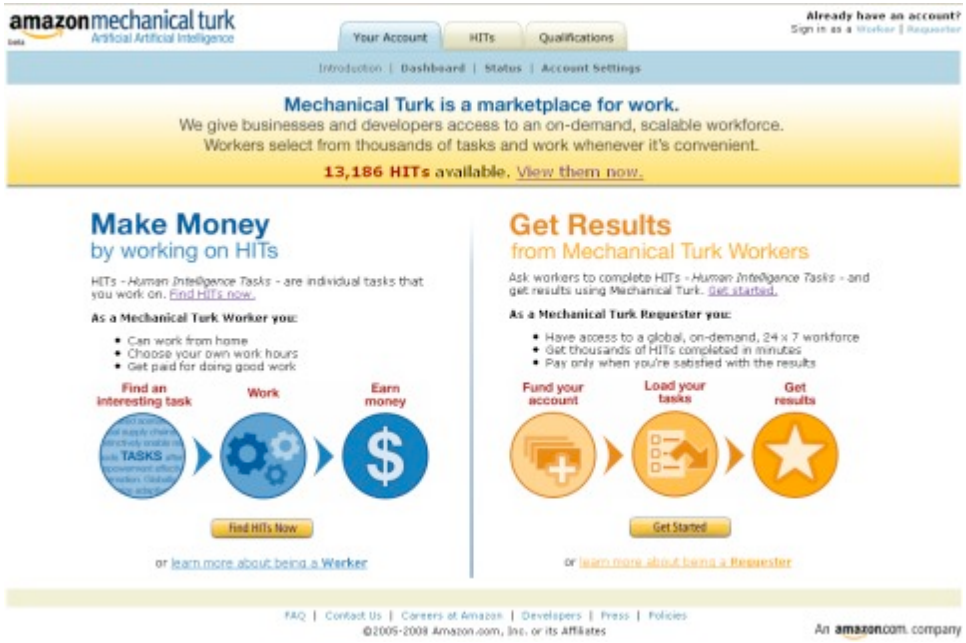
Company_Name	Address	Market Cap
Google	Googleplex, Mtn. View CA	$210Bn
Intl. Business Machines	Armonk, NY	$200Bn
Microsoft	Redmond, WA	$250Bn

```
SELECT Market_Cap
From Companies
Where Company_Name =
  "The Cool Software Company"
```

```
$2xxBn
Number of Rows: 1
```


amplab

Crowdsourcing



flickr

yelp

twitter

samasource

[TopCoder]

many eyes beta

for shared visualization and discovery

amplab

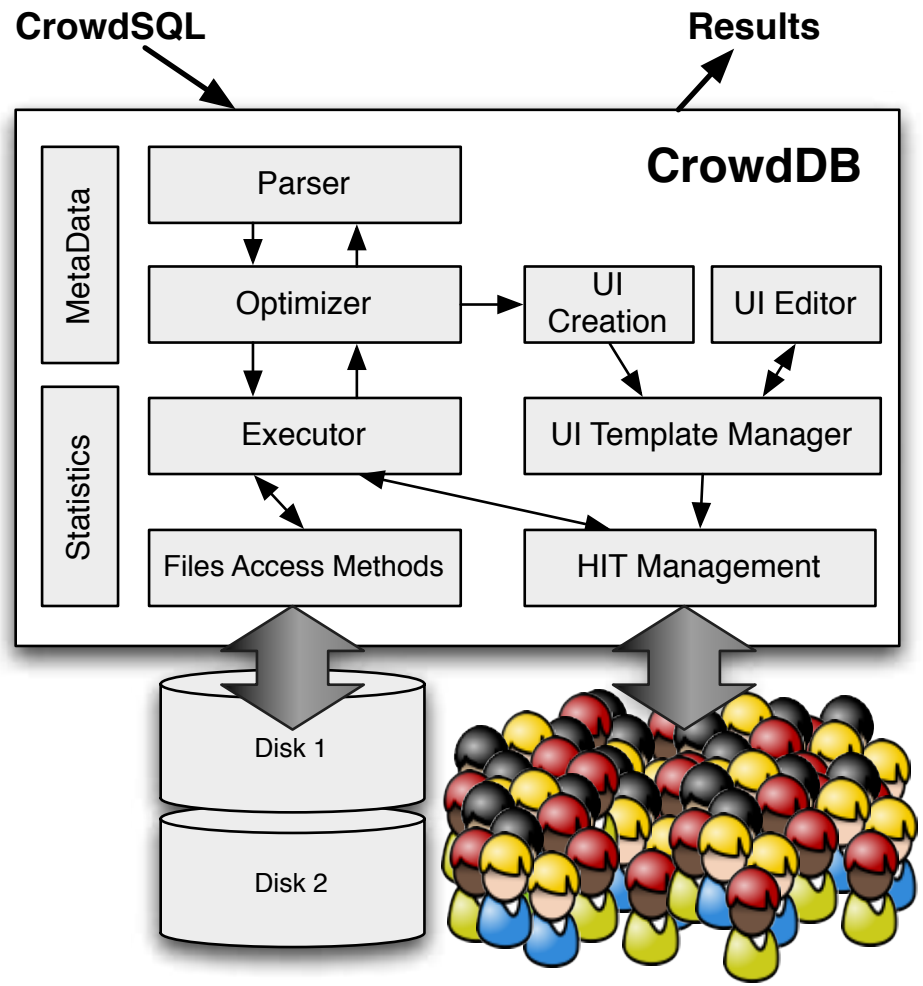
Microtasking – Virtualized Humans

- Current leader: Amazon Mechanical Turk
- Requestors place Human Intelligence Tasks (HITs)
  - Minimum price: $0.01
  - Other parameters: #of replicas (assignments), expiration, **User Interface**,…
  - API-based: "createHit()", "getAssignments()", "approveAssignments()", "forceExpire()"
  - Requestors approve jobs and payment
- Workers (a.k.a. "turkers") choose jobs, do them, get paid

CrowdDB

Use the crowd to answer DB-hard queries

- Where to use the crowd:
  - Have people find data
  - Have people do "fuzzy" comparisons
    - Equality (hash) and Ordering (sort)
- Where it doesn't make sense:
  - Have people do quick sort of sets (i.e., TurkIt)
  - Anything the computer already does well



amplab

CrowdSQL == SQL???

- **Closed-World → Open-World**
  - SQLs closed-world assumption is a lie
  - Influences query execution strategies/options
  - At the moment, we only allow a restricted set of queries against crowd-sourced tables
  - Goal: Explore the open-world as much as possible (e.g., answer a query as best as possible given a certain budget)
- **Caching**
  - Every result from the crowd is stored → Too expensive not to do so
  - Queries use stored results whenever possible
  - Answers change based on the query history and cache behavior
  - Goal: Offer more control over caching, TTL, …
- **Answer Quality**
  - Human-input-tolerant query processing
  - At the moment a simple set of heuristics
  - Goal: confidence intervals, iterative improvement,…

CrowdSQL

DDL Extensions:

Crowdsourced columns

```
CREATE TABLE company (
  name STRING PRIMARY KEY,
  hq_address CROWD STRING);
```

Crowdsourced tables

```
CREATE CROWD TABLE department (
  university STRING,
  department STRING,
  phone_no STRING)
PRIMARY KEY (university, department);
```

DML Extensions:

CrowdEqual:

```
SELECT *
FROM companies
WHERE Name ~ "Big Blue"
```

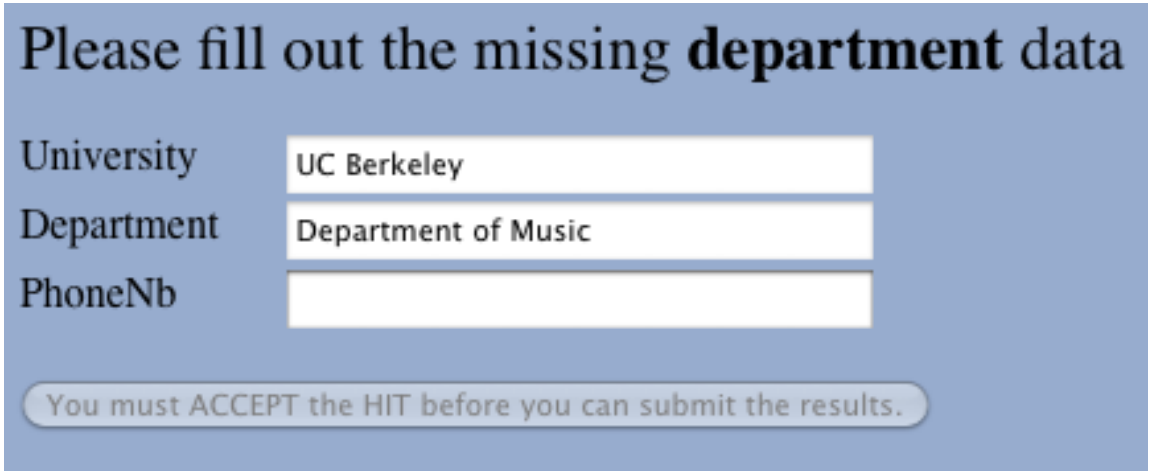
CROWDORDER operators (currently UDFs):

```
SELECT p FROM picture
WHERE subject =
    "Golden Gate Bridge"
ORDER BY CROWDORDER(p, "Which
pic shows better %subject");
```


amplab

User Interface Generation

- A clear UI is key to response time and answer quality.
- We can leverage the SQL Schema to auto-generate UI (e.g., Oracle Forms, etc.)



amplab

UI with Context

Find the missing information about the academic **department** at **UC Berkeley**

Department: Department of Music

PhoneNb:

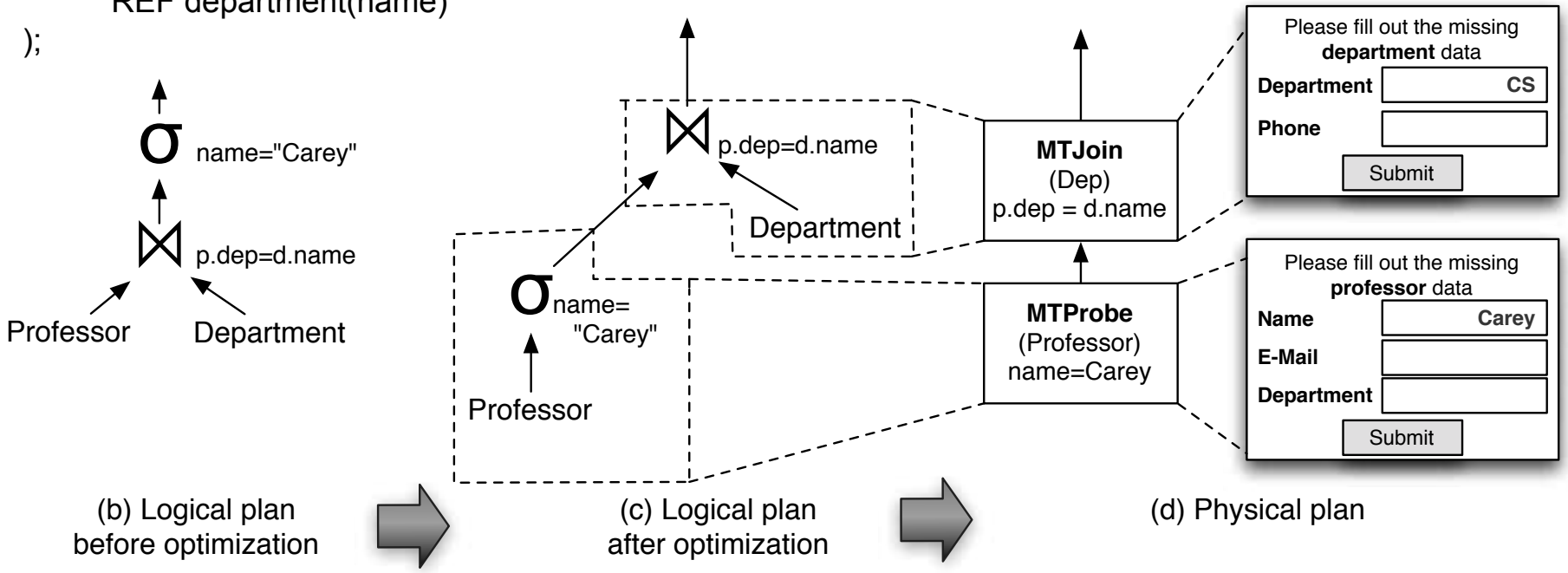
You must ACCEPT the HIT before you can submit the results.

amplab

Query Optimization and Execution

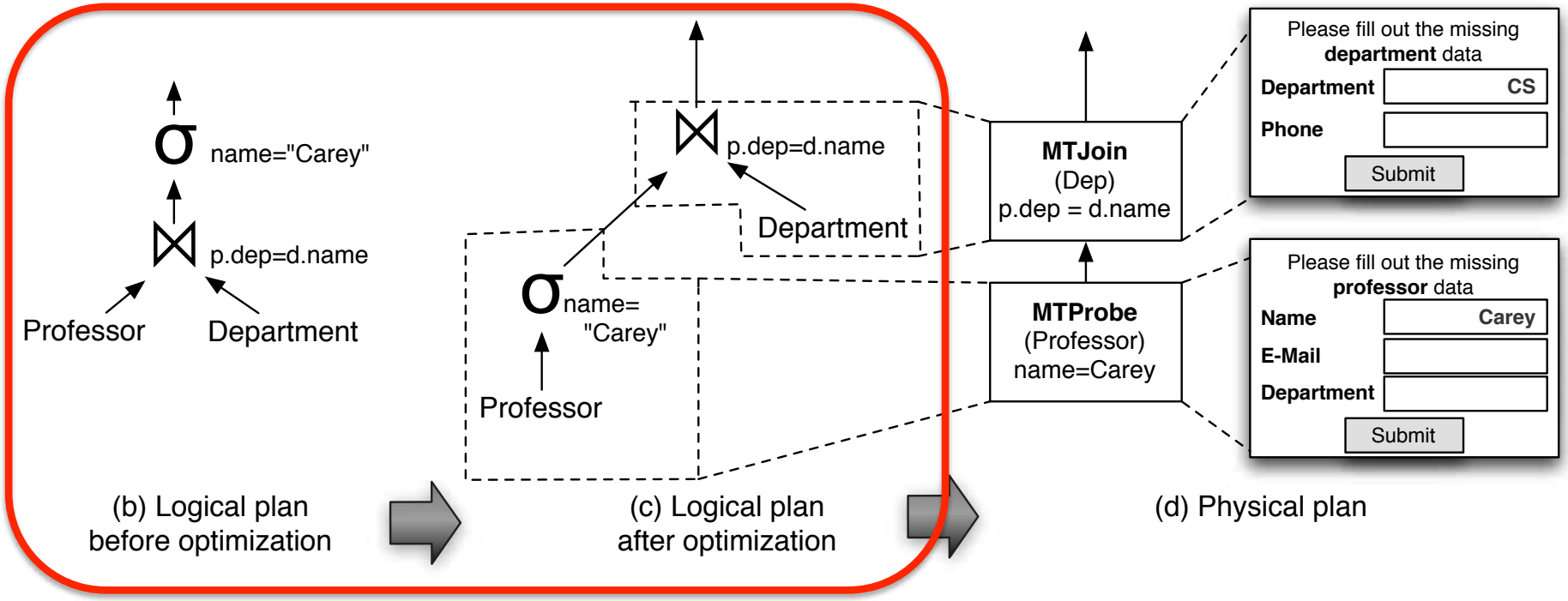
```
CREATE CROWD TABLE department (
    name STRING PRIMARY KEY
    phone_no STRING);

CREATE CROWD TABLE professor (
    name STRING PRIMARY KEY
    e-mail STRING
    dep STRING
      REF department(name)
);
```

SELECT *
FROM PROFESSOR p, DEPARTMENT d
WHERE d.name = p.dep
AND p.name ="Michael J. Carey"



amplab

Query Optimization



Rule based optimizer

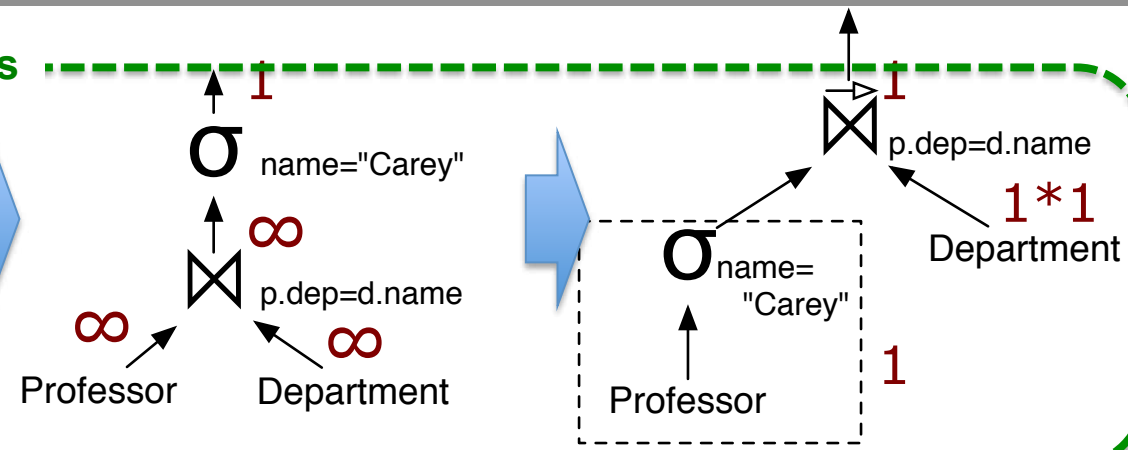
- Performance Insightful Query Language (PIQL) techniques to deal with open-world assumption
- Simple set of rules to pick the best plan
- Simple heuristics to set the crowd parameters (e.g., replication factor, price per HIT, etc.)

amplab

Dealing with the Open-World

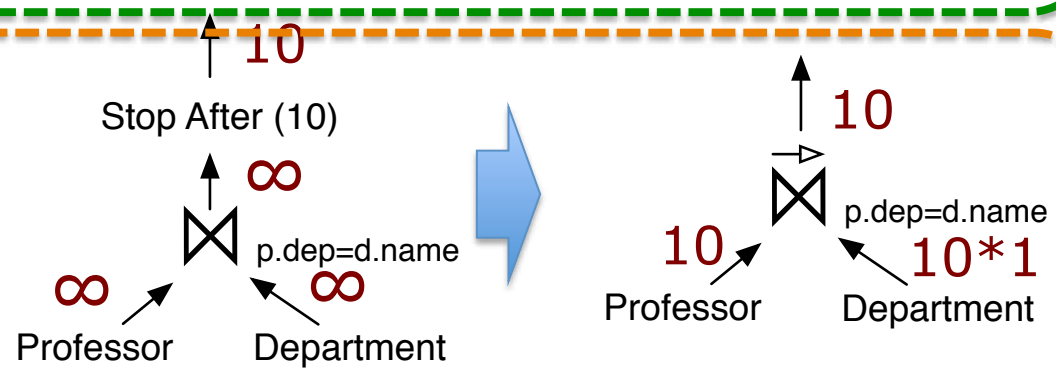
Focus Right Now: PK Queries

```
SELECT *
FROM PROFESSOR p,
  DEPARTMENT d
WHERE p.dep = d.name
AND p.name ="Carey"
```



Soon

```
SELECT *
FROM PROFESSOR p,
  DEPARTMENT d
WHERE p.dep = d.name
LIMIT 0, 10
```

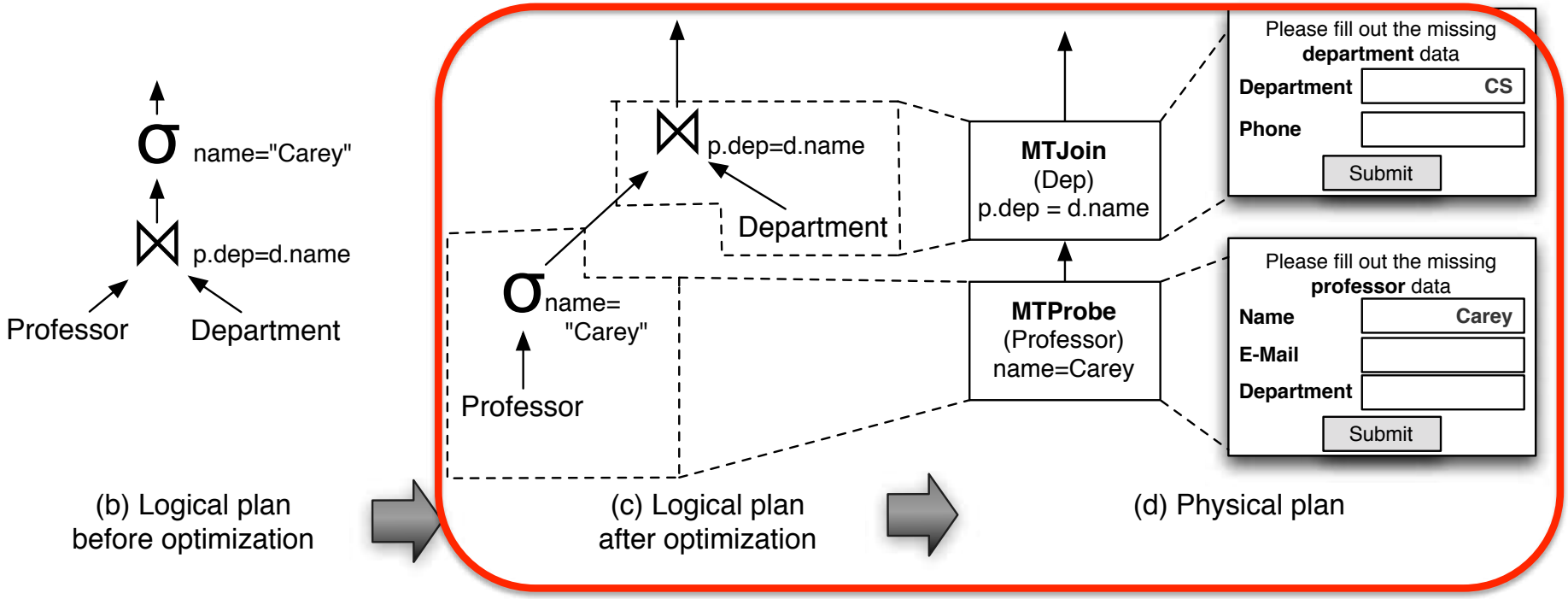


Never

```
SELECT *
FROM PROFESSOR p,
  DEPARTMENT d
WHERE p.dep = d.name
```

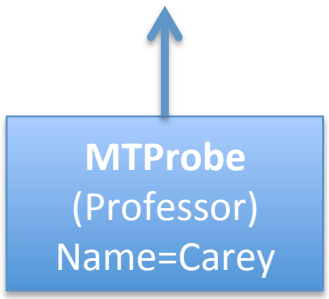
amplab

Query Optimization

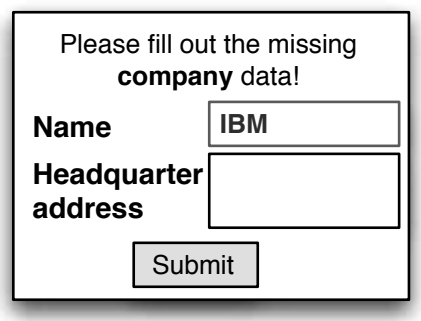


- Creates user interface templates
- Select physical operators
- New Query Operators:
  - Crowd Operators: MTProbe, MTJoin, MTFunction
  - Other: STOP AFTER (i.e., limit)

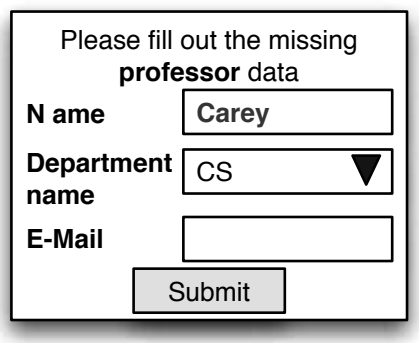
MTProbe



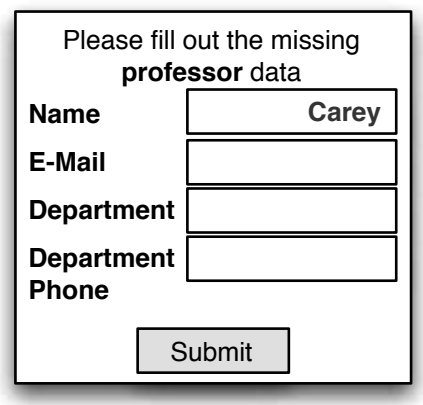
- Similar to a table-scan with predicate push-downs
- Batches several jobs into one HIT
- Issues as many requests in parallel as possible (based on the cardinality prediction)
- Does simple quality control (quorum votes)
- "Caches" the result inside the corresponding table → queries have side-effects



Crowd Column & Crowd Colums w/o foreign keys



Crowd Column & Crowd Colums with foreign keys



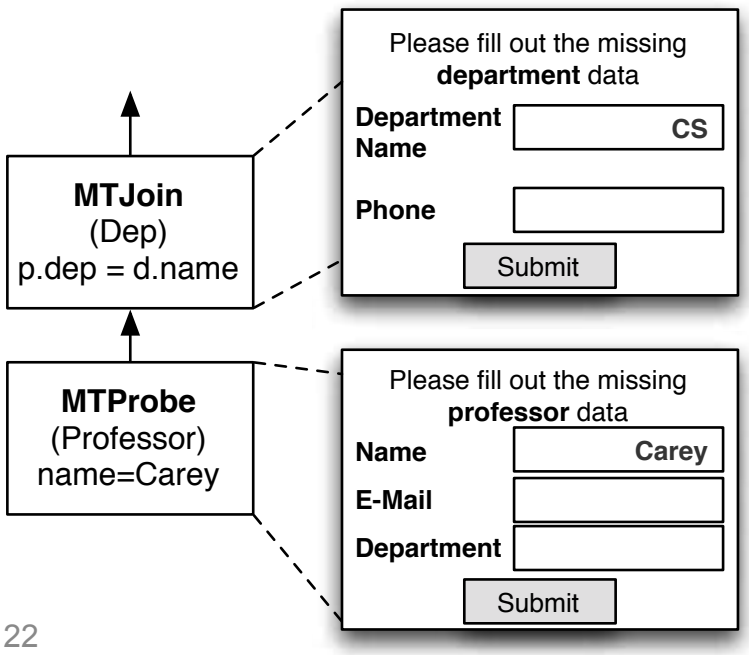
Denormalization

amplab

MTJoin & MTFunction

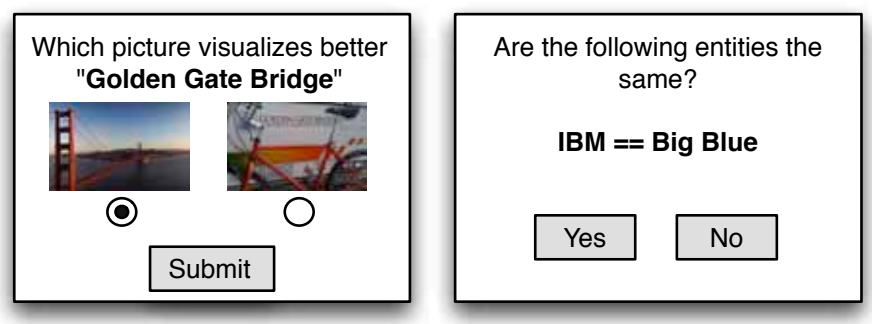
MTJoin

- Indexed nested-loop join
- Rest (quality control, HIT grouping) similar to MTProbe



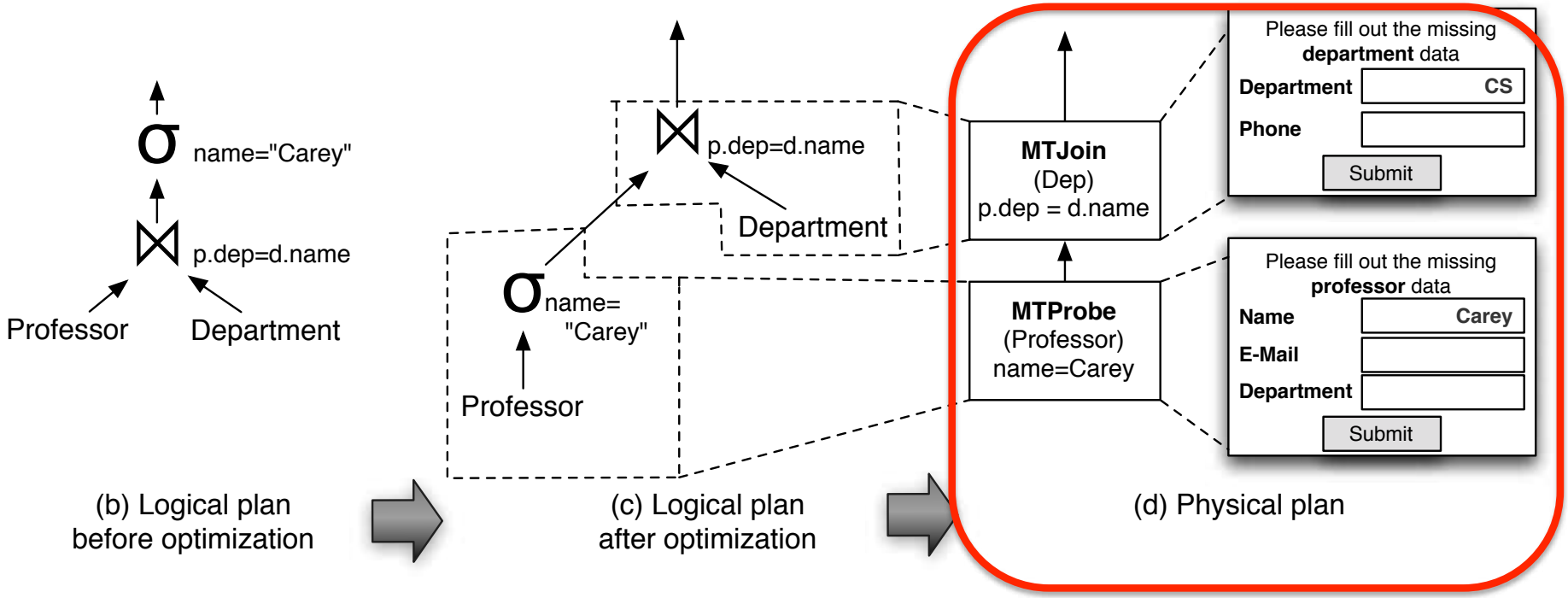
MTFunction

- implements the CROWDEQUAL and CROWDORDER comparison
- Takes some description and a type (equal, order) parameter
- Ordering can be further optimized (e.g., Three-way comparisions vs. Two-way comparisons)



amplab

Query Execution



Initiate user interface templates at run-time

amplab

Entity resolution

Schema:

```
CREATE TABLE company (
 name STRING,
 headquarter_address CROWD STRING
);
```

Query:

```
SELECT name
FROM company
WHERE name ~ [a non-uniform
    name of the company]
```

Which entities are the same as
Big BLUE?

- ☐ Google
- ☐ HP
- ☐ IBM
- ☐ Facebook
- ☐ NetApp
- ☐ CrowdFlower
- ☐ Yahoo
- ☐ Microsoft
- ☐ Salesforce
- ☐ SAP
- ☐ *None of the above*

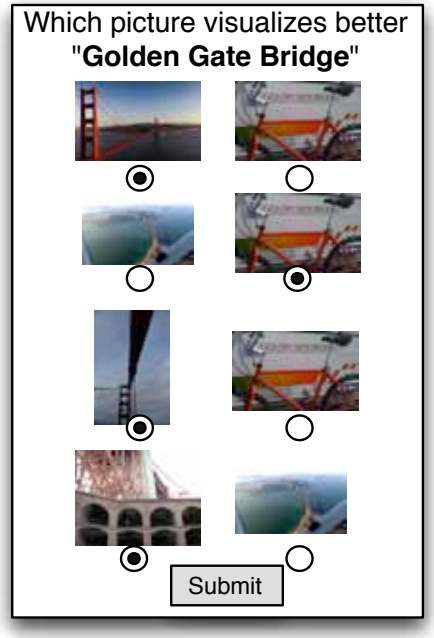
Data-Size: 100 company names
Batching: 10 comparisons per HIT
Replication: 3 Assignments per HIT
Price: 1 cent per HIT

Non Uniform Name	Query Result	Votes	Error Examples
Bayerische Motoren Werke	BMW	3	TATA Group, Gazprom, Boeing, Toyota
International Business Machines	IBM	2	Samsung, HP
Company of Gillette	P&G	2	Aviva, AIG, France Telecom
Big Blue	IBM	2	Microsoft

amplab

Picture ordering

```
Query:
SELECT p FROM picture
WHERE subject = "Golden Gate Bridge"
ORDER BY CROWDORDER(p, "Which pic shows
                        better %subject");
```



Data-Size: 30 subject areas, with 8 pictures each
Batching: 4 orderings per HIT
Replication: 3 Assignments per HIT
Price: 1 cent per HIT

(turker-votes, turker-ranking, expert-ranking)

Our (database) dream…

A cost-based optimizer for the crowd

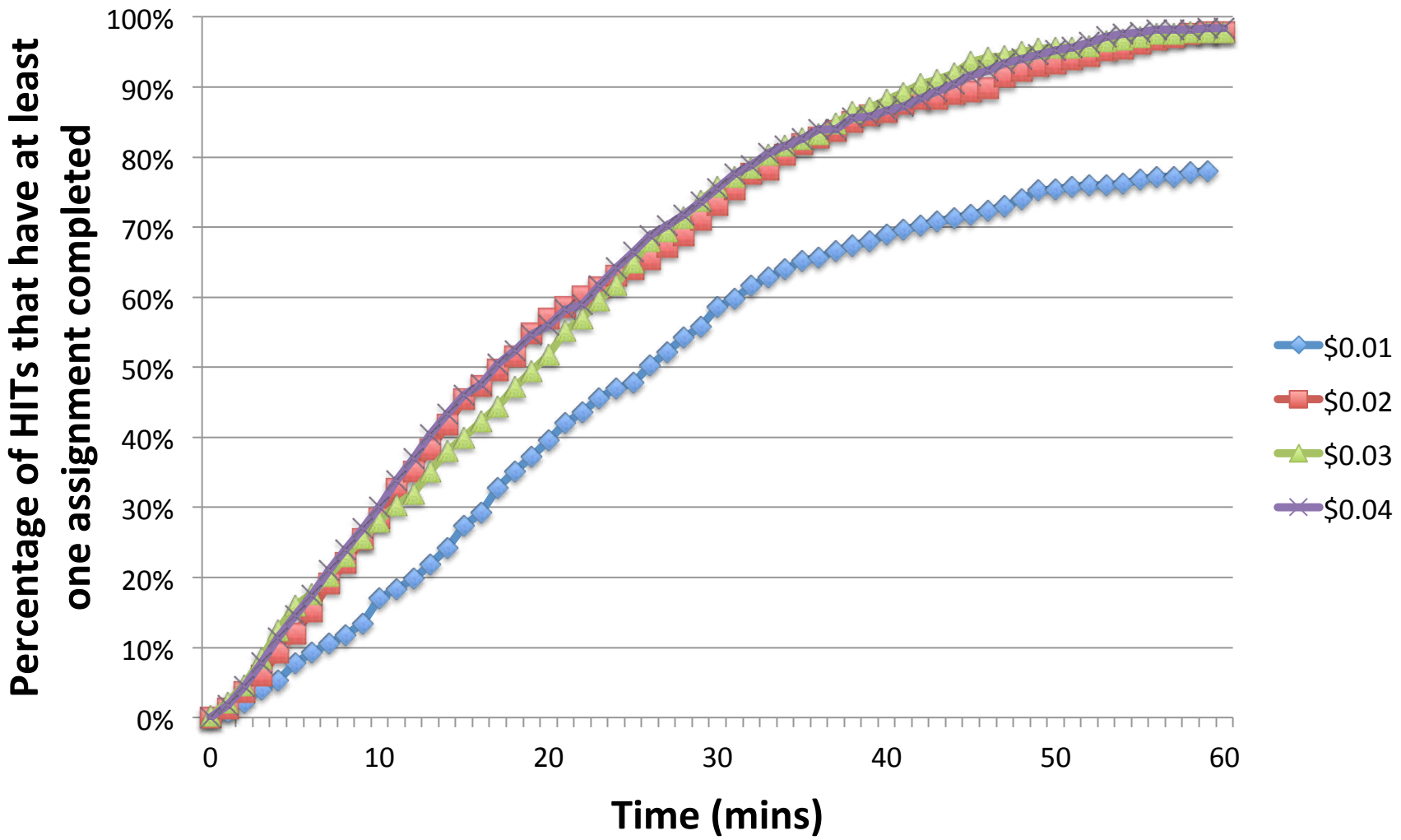
- SQL provides physical and logical data independence
- Crowd platforms have so many parameters; impossible for programmer to get them all right
  - Price per HIT
  - Complexity/Nb. questions per HIT
  - Nb. assignments per HIT → Replication
  - Type of User Interface
  - …
- Parameters change over time (e.g., turkers learn, prices increase, ...)

Idea: Create a <u>cost model</u> of crowd operators and plug them into the DB optimizer.

amplab

Price vs. Response Time

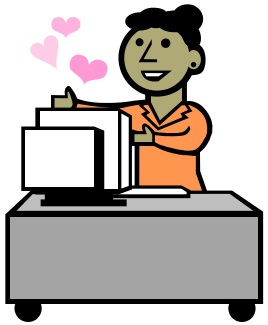
5 Assignments, 100 HITs



amplab

But…

Processor Relations?



AMPLab

HIT Group » Simple straight-forward HITs, find the address and phone number for a given business in a given city. All HITs completed were approved. Pay was decent for amount of time required, when compared to other available HITs. But not when looked at from an hourly wage perspective. I would do work for this requester again. posted by…

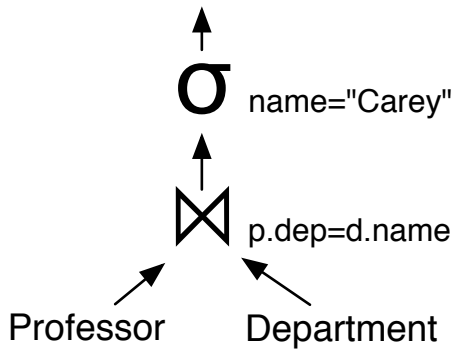
fair:5 / 5 fast:5 / 5 pay:4 / 5 comm:0 / 5

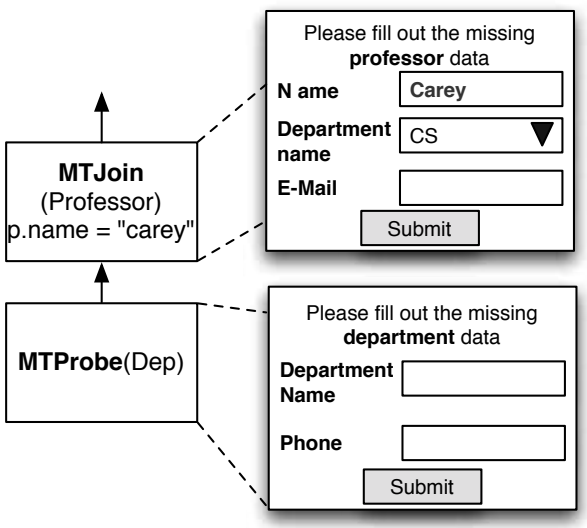
Tim Klas Kraska

HIT Group » I recently did 299 HITs for this requester.... Of the 299 HITs I completed, 11 of them were rejected without any reason being given. Prior to this I only had 14 rejections, a .2% rejection rate. I currently have 8522 submitted HITs, with a .3% rejection rate after the rejections from this requester (25 total rejections). I have attempted to contact the requester and will update if I receive a response. Until then be very wary of doing any work for this requester, as it appears that they are rejecting about 1 in every 27 HITs being submitted. posted by …

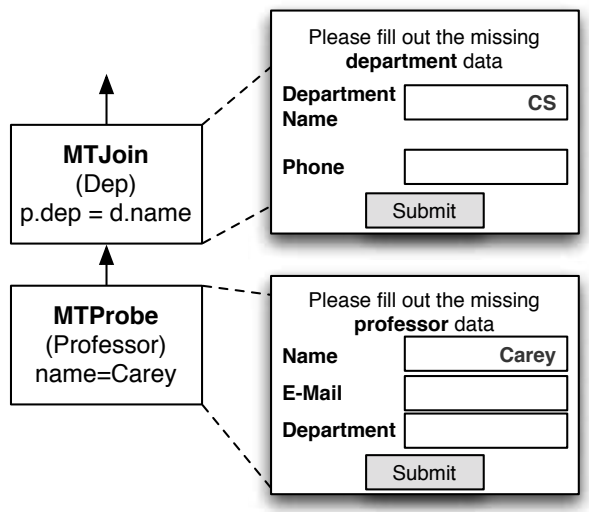
fair:2 / 5 fast:4 / 5 pay:2 / 5 comm:0 / 5

User Interface vs. Quality

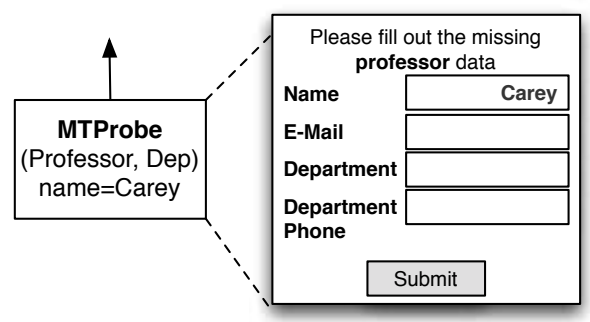




(Department first)



(Professor first)

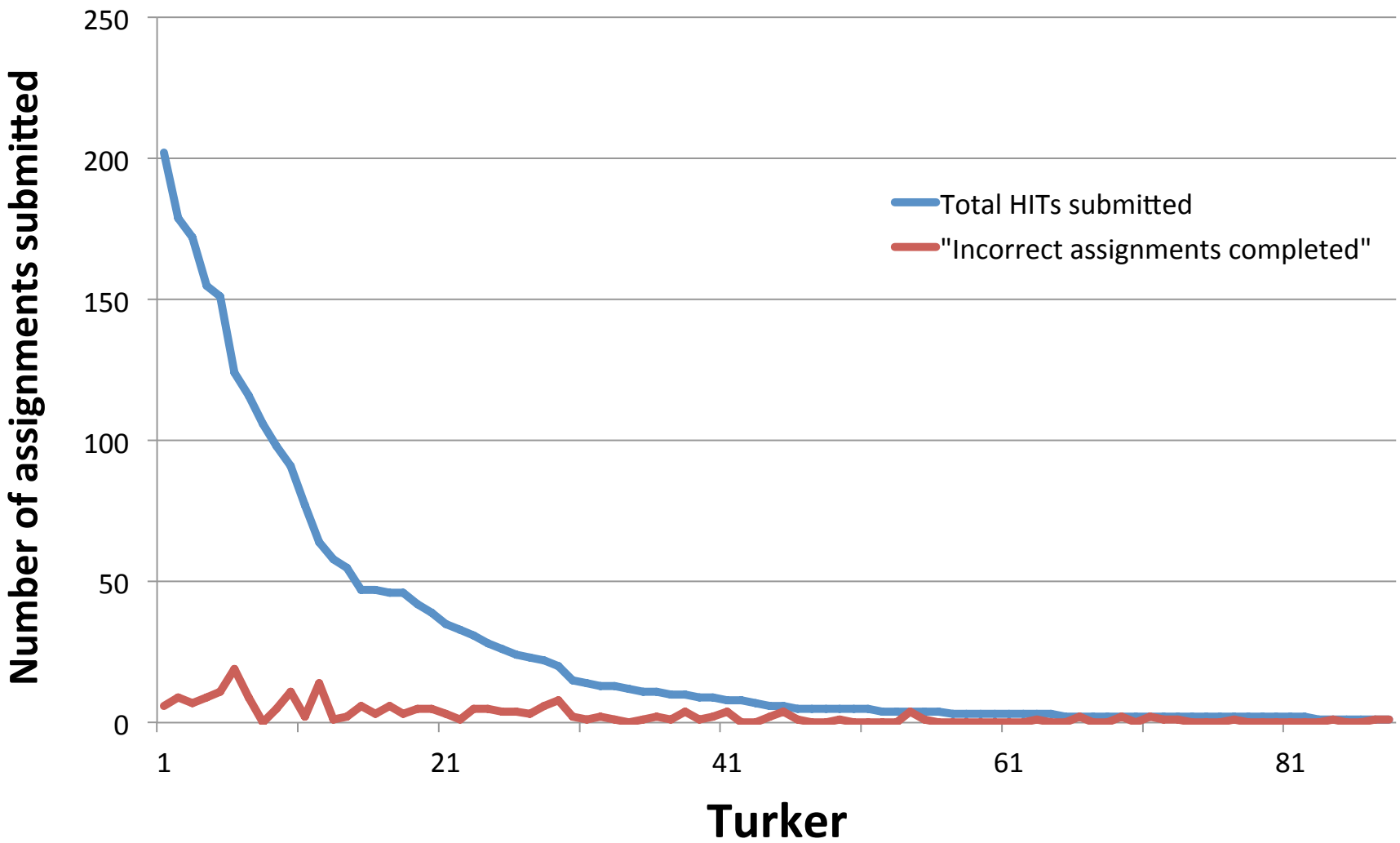


(De-normalized Probe)

amplab

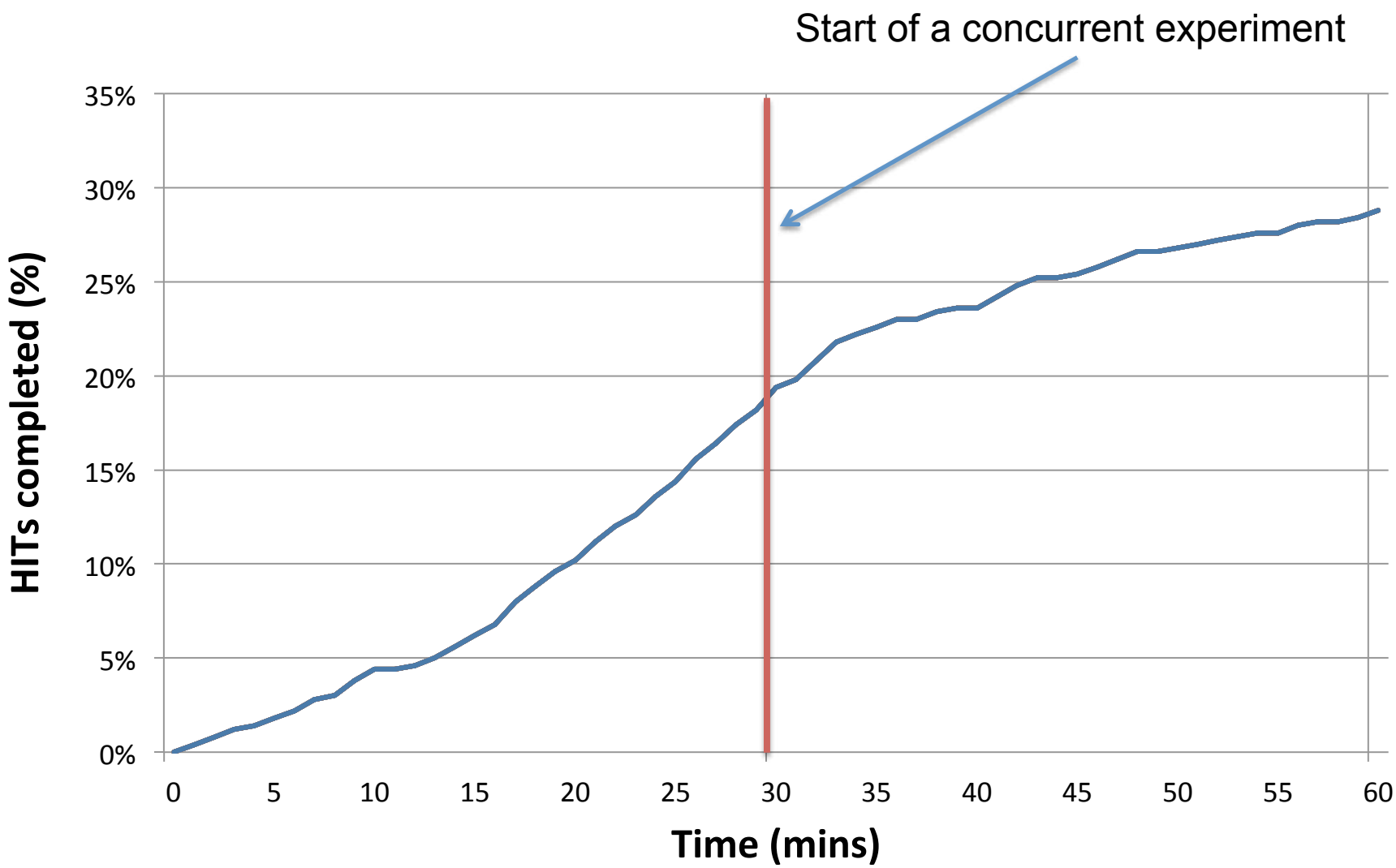
Turker Affinity and Errors

5 Assignments



amplab

The market is smaller than expected



amplab

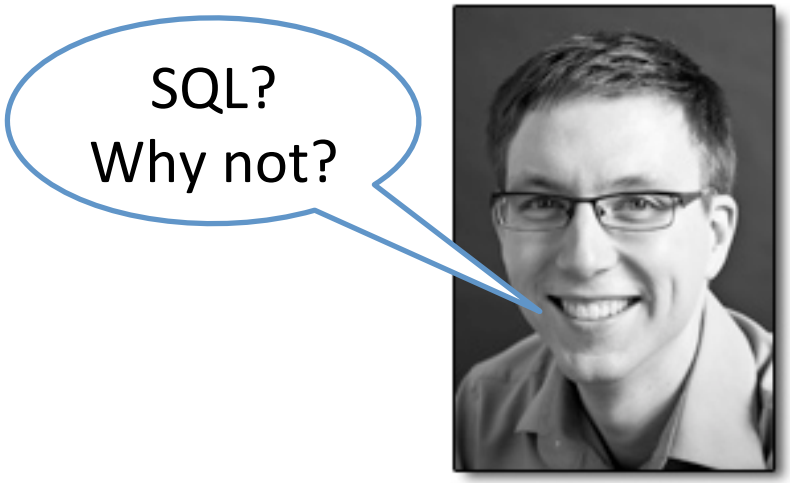
Future: Crowdsourcing → DB++?

- Cost Model for the Crowd
  - Latency (mins) vs. Cost ($) vs. Quality (%error)
- Adaptive Query Optimization
  - How to monitor crowd during query execution?
  - How to adapt the query plan?
- Caching / Materializing Crowd Results
  - E.g., maintaining the cached values
- Complexity Theory
  - Three-way comparisions vs. Two-way comparisons
- Privacy: Public vs. Private crowds
  - Flip-side of affinity of turkers
- Meta-crowds: Crowds help crowds (e.g. UI)

Future: DB → Crowdsourcing++

Crowd-Hard Problems:

- **Programming Language: GUI**
  Question design, ambiguities, granularity, …
- **Many, many knobs to turn**
  Price, replication factor, HIT group size, expiration time,….
- **Changing platform behavior**
  Increasing market size, new platform features,…
- **Jungle of different techniques**
  Rejection policy (Quorum-Vote, Test-Set,…), Quality control (Quorum, iterative models,…)
- **Learning effects / Community Management**
- …



The DB-Approach

- **Data independence**
  If HW changes, app need not change
- **DBMS optimizes queries**
  - Decide what to crowdsource
  - Statistics about the market place, question ordering,…

Related Work

- A. Marcus, E. Wu, S. Madden and R. Miller:
  Crowdsourced Databases: Query Processing with People. *CIDR, 2011*
- A. Parameswaran and N. Polyzotis:
  Answering Queries using Humans, Algorithms and Databases. *CIDR, 2011*
- A. Parameswaran, A. Das Sarma, H. Garcia-Molina, N. Polyzotis and J. Widom:
  Human-assisted Graph Seach: It's okay to ask questions!. VLDB. *2011*
- S. Amer-Yahia, A. Doan, J. M. Kleinberg, N. Koudas, M. J. Franklin:
  Crowds, clouds, and algorithms: exploring the human side of "big data" applications. *SIGMOD, 2010*
- P. DeRose, X. Chai, B. Gao, W. Shen, A. Doan, P. Bohannon and J. Zhu:
  Building Community Wikipedias: A Human-Machine Approach, *ICDE, 2008*
- G. Little, L. B. Chilton, R. Miller and M. Goldman:
  TurKit: Tools for Iterative Tasks on Mechanical Turk. *HCOMP´09*
- J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh:
  VizWiz: nearly real-time answers to visual questions. UIST, *2010*
- P. G. Ipeirotis:
  Analyzing the Mechanical Turk Marketplace, ACM XRDS, 2010

amplab

Summary

A first attempt towards the P in AMP

- CrowdDB is a hybrid Crowd/Cloud computing
  - Small set of SQL extensions allow to express how the crowd should be used
  - Special crowd operators encapsulate the input of the crowd
- Shows that people can help answer DB-hard Queries
- And, it raises lots of interesting and important research issues.

Tim Kraska
kraska@cs.berkeley.edu

amplab