Data Platforms for the Cloud +Big Data World: What's New?

Surajit Chaudhuri

surajitc@microsoft.com

Microsoft Research

A Few Implications of Cloud and Big Data

- Invariant: SQL or Relational Algebra variants rule as primary workload over all Data Analytic Systems
  - Even for MR, Spark, or NoSQL ecosystems
  - Why? Programmer productivity is paramount
  - Challenge: Query Optimization Technology stuck in 1980s with poor support for user-defined functions
- New: Elasticity is the defining attraction of Cloud
  - Challenge: Compute-Storage separation (across the network) for data analytic systems
  - Challenge: Enable "On-the-fly" light predicate evaluation for Storage Service
  - Challenge: Resource Governance for Multi-Tenant Data Services
  - Challenge: Provisioning to support Serverless for Data Services (aka "Extreme Elasticity")

Two Opportunities

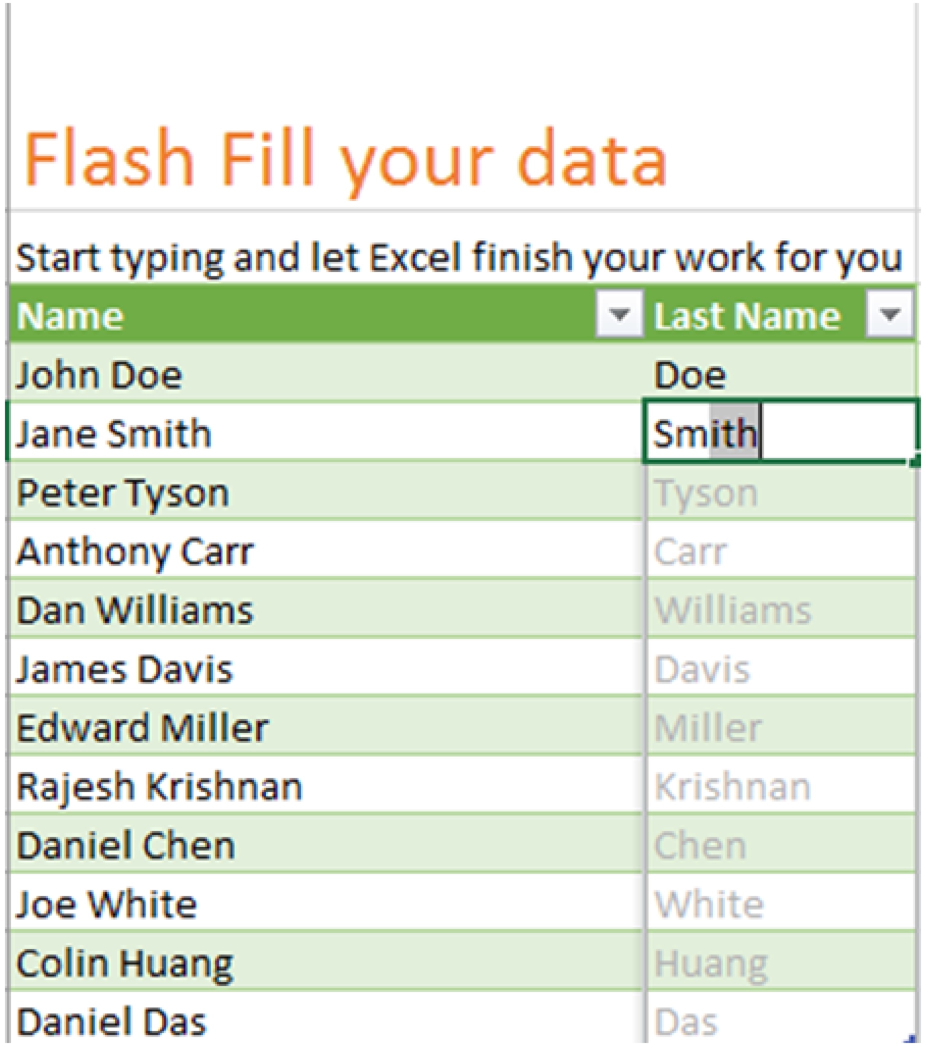
- Data Transformation: Crucial for data analytics (OLAP or ML)
  - A field that has not advanced as much
  - Can we take a leap in reducing the cost to programmers?
- Approximate Query processing: Potential to lower Cost
  - Why run analytics on Petabytes/Exabytes of data all the time?
  - Can we support approximation for SQL queries?
- *Language abstractions are vital for both Data Transformation DSL and injection of Approximation in Query processing and we need the help of the PL community*
- More on these two now …

Transform-Data-By-Example

Yeye He (Lead) and Collaborators

By-Example Transformation

- Users only need to give a few input-output examples
- Algorithm searches programs in pre-defined string operators:
  - E.g. substring, concat, split
- But, falls short in many domain-specific transformation tasks …


Flash Fill your data

Start typing and let Excel finish your work for you

Name	Last Name
John Doe	Doe
Jane Smith	Smith
Peter Tyson	Tyson
Anthony Carr	Carr
Dan Williams	Williams
James Davis	Davis
Edward Miller	Miller
Rajesh Krishnan	Krishnan
Daniel Chen	Chen
Joe White	White
Colin Huang	Huang
Daniel Das	Das

Domain-Specific Transformation: Names

Input	Output
John K. **Doe** Jr.	Doe, John
Mr. **Doe**, **John**	Doe, John
Jane A. **Smith**	Smith, Jane
MS. **Jane Smith**	
Smith, **Jane**	
Dr **Anthony** R **Von Fange** III	
Peter Tyson	
Dan E. **Williams**	
James Davis Sr.	
James J. **Davis**	
Mr. **Donald** Edward **Miller**	
Miller, **Donald**	
Rajesh Krishnan	
Daniel Chen	

Many More Domain-Specific Transformations ...

- Many head and tail domains:
  - *Name, date, phone, email, url, address, unit, ip, number-encoding, color, string-casing, math-expressions, html, isbn, …*
- Proprietary, enterprise domains
- FlashFill-like approach: one-domain-at-a-time, would not scale

Using Code for Domain-Specific Transformations

- Rich transformation logic locked up in code repositories
- Systematically **collect**, and **compose** them for user tasks
- **Extensible** to many domains

Leveraging Functions

Input	Output
John K. **Doe** Jr.	Doe, John
Mr. **Doe, John**	Doe, John
Jane A. **Smith**	Smith, Jane
MS. **Jane Smith**	
Smith, Jane	
Dr **Anthony** R **Von Fange** III	
Peter Tyson	
Dan E. **Williams**	
James Davis Sr.	
James J. **Davis**	
Mr. **Donald** Edward **Miller**	
Miller, Donald	
Rajesh Krishnan	
Daniel Chen	

Github: NameParser Function (Apache-license)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace CSharpNameParser
{
    public class NameParser
    {
        public Name Parse(string fullName)
        {
            string salutation = null;
            string firstName = null;
            string initials = null;
            string lastName = null;
            string suffix = null;
            // Parse the name
            // The logic to parse the name
            //  ...
            //  ...

            return new Name()
            {
                Salutation = salutation,
                FirstName = firstName,
                MiddleInitials = initials,
                LastName = lastName,
                Suffix = suffix
            };
        }

    }
}
```

Synthesis using Existing Function

Github: NameParser Function (Apache-license)

Invoke Functions

Program Synthesis

Input	Sal.	FN	M.	LN	Suffix	Output
John K. Doe Jr.		John	K.	Doe	Jr.	Doe, John
Mr. Doe, John	Mr.	John		Doe		Doe, John
MS. Jane A. Lee	MS.	Jane	Alice	Lee		Lee, Jane
Sean Smith Sr.		Sean		Smith	Sr.	Smith, Sean
Dr. Alan Turing	Dr.	Alan		Turing		Turing, Alan

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace CSharpNameParser
{
    public class NameParser
    {
        public Name Parse(string fullName)
        {
            string salutation = null;
            string firstName = null;
            string initials = null;
            string lastName = null;
            string suffix = null;
            // Parse the name
            // The logic to parse the name
            //  ...
            //  ...
            return new Name()
            {
                Salutation = salutation,
                FirstName = firstName,
                MiddleInitials = initials,
                LastName = lastName,
                Suffix = suffix
            };
        }

    }
}
```

Input

Initialize

Domain-specific parsing logic

Return

Transform-Data-by-Example (TDE): Highlights

- **Search by-example** interactively

- **Synthesized** complex programs on-the-fly

- **Out-of-box** support for many head and tail domains

- **Extensible** to new code, DLL, services and tables

TDE: Product Impact and Technical Reference

- Technical reference
  - Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek R. Narasayya, Surajit Chaudhuri: Transform-Data-by-Example (TDE): An Extensible Search Engine for Data Transformations. PVLDB 11(10): 1165-1177 (2018)
- June 2017: Released as Excel Add-in in Office Store (via Garage)
  - Since release: 455K user transformation queries [as of 6 months ago]

Download from the Office Store

- May 2018: Synthesis technology ships in Microsoft Power Query

PQ Power Query

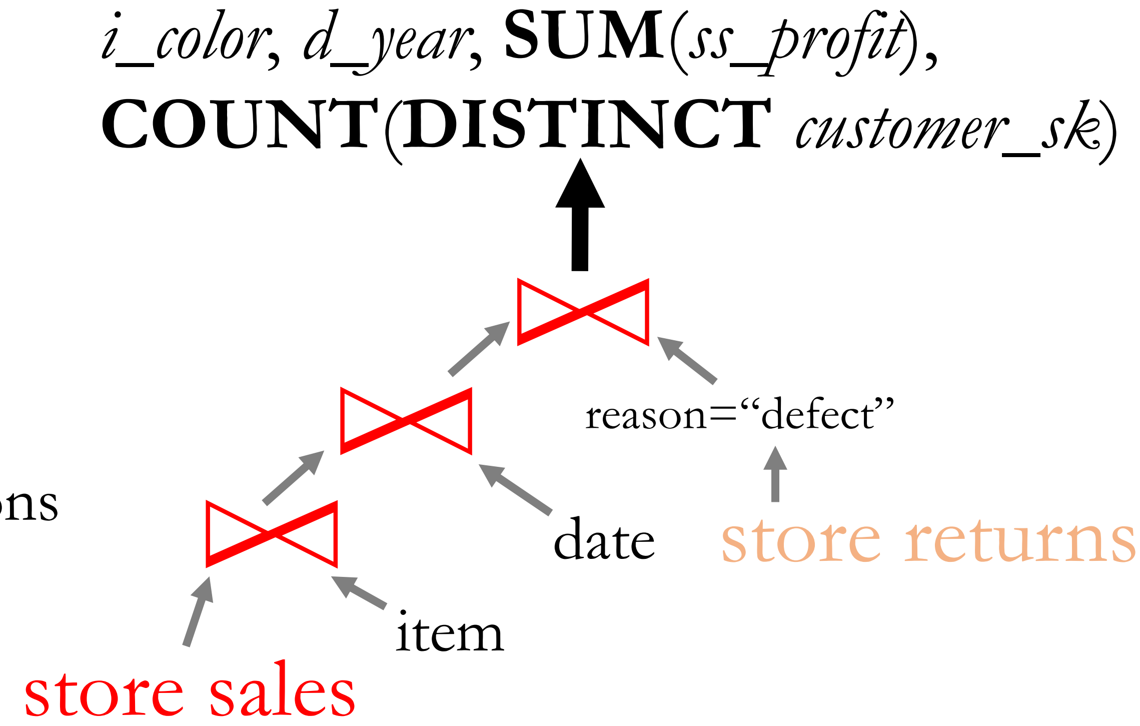
Approximating Queries in Microsoft's Big-Data Clusters

Srikanth Kandula (Lead)

Approximating complex queries is challenging

If not careful, an approximate answer may
- miss rows in answer ("groups")
- mis-estimate aggregates

Gains are large only if "work" is skipped
⇒ Must reduce input ***before*** costly operations



Solution space for AQP

Low Coverage / High Efficiency

Efficiency

Coverage

High Coverage / Low Efficiency

Simple queries

Predictable queries
(i.e., with few, known, columns in {selects, aggregates, …}), only simple joins

Complex, ad-hoc queries

Precomputed Sketches and Synopses

E.g., HLL, T-Digest

Precomputed input samples and/or indices

E.g., PowerDrill, SnappyData

Metric in SCOPE clusters	**Value @ Percentile**		
	50th	**75th**	**95th**
#Columns	8	24	104
#Joins	3	9	112
#user-defined functions, aggregates	27	46	265
# of "passes" over data	2.45	3.63	9.78

Query-time Sampling

Samplers are streaming operators

1. <u>Uniform *(p)*</u> — Each row passes with probability *p*.
2. <u>Distinct (*C*, *f*, *p*)</u> — When used before group-by, can ensure no group miss

> Per unique value of columns in C, pass *f* rows and the rest with probability p.

item_sk, **SUM**(*ss_profit*)

↑

store sales

item_sk, **SUM**(*ss_profit*w*)

↑

$Distinct(\{item\_sk\}, p, 3)$

↑

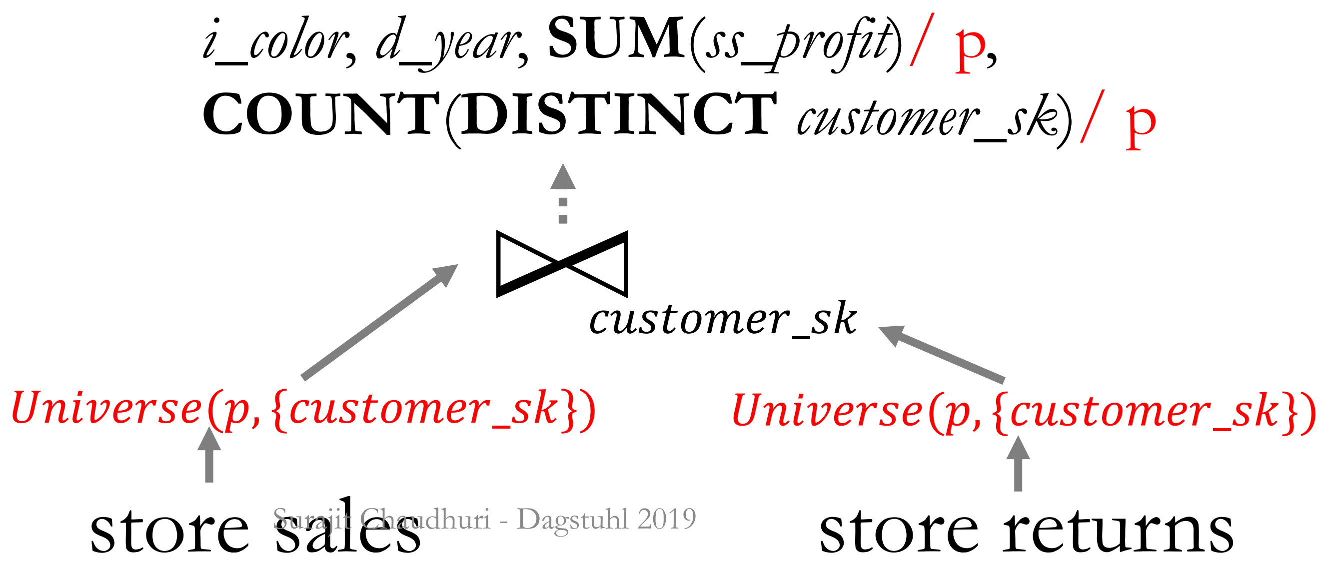
store sales

Samplers are streaming operators

1. Uniform (p) — Each row passes with probability p.
2. Distinct (C, f, p) — When used before group-by, can ensure no group miss
3. Universe (C, p) — When used before join, can ensure

$$\text{join} \leftarrow \{\text{sample}\} \equiv \text{sample} \leftarrow \text{join}$$

Pick *a random p fraction* of values of C



Sampler pushdown rules in production

		Transformation	Condition
Sampler below project	Rule-U1	$\Gamma^{\mathsf{U}}_{p}(\pi(R)) \overset{*}{\Leftrightarrow} \pi(\Gamma^{\mathsf{U}}_{p}(R))$	–
	Rule-V1	$\Gamma^{\mathsf{V}}_{p,\mathcal{D}}(\pi(R)) \overset{*}{\Leftrightarrow} \pi(\Gamma^{\mathsf{V}}_{p,\mathcal{D}_{\mathcal{C}_b\to\mathcal{C}_a}}(R))$	if $c \notin \mathcal{D}$
	Rule-D1	$\Gamma^{\mathsf{D}}_{p,\mathcal{D},f}(\pi(R)) \overset{*}{\Leftrightarrow} \pi(\Gamma^{\mathsf{D}}_{p,\mathcal{D}_{\mathcal{C}_b\to\mathcal{C}_a},f}(R))$	if $c \notin \mathcal{D}$ or $\mathcal{C}_c \subseteq \mathcal{D}$
Sampler below select	Rule-U2	$\Gamma^{\mathsf{U}}_{p}(\sigma_{\mathcal{C}}(R)) \overset{*}{\Leftrightarrow} \sigma_{\mathcal{C}}(\Gamma^{\mathsf{U}}_{p}(R))$	–
	Rule-V2	$\Gamma^{\mathsf{V}}_{p,\mathcal{D}}(\sigma_{\mathcal{C}}(R)) \overset{*}{\Leftrightarrow} \sigma_{\mathcal{C}}(\Gamma^{\mathsf{V}}_{p,\mathcal{D}}(R))$	–
	Rule-D2	$\Gamma^{\mathsf{D}}_{p,\mathcal{D},f}(\sigma_{\mathcal{C}}(R)) \overset{*}{\Leftrightarrow} \sigma_{\mathcal{C}}(\Gamma^{\mathsf{D}}_{p,\mathcal{D},f}(R))$	if $\mathcal{C} \subseteq \mathcal{D}$
Sampler below join	Rule-U3	$\Gamma^{\mathsf{U}}_{p}(R \bowtie_{\mathcal{C}} S) \overset{*}{\Leftrightarrow} \Gamma^{\mathsf{U}}_{p}(R) \bowtie_{\mathcal{C}} S$	if $\mathcal{C}$ is a primary-key in S
	Rule-V3a	$\Gamma^{\mathsf{V}}_{p,\mathcal{D}}(R \bowtie_{\mathcal{C}} S) \overset{*}{\Leftrightarrow} \Gamma^{\mathsf{V}}_{p,\mathcal{D}_{S\to R}}(R) \bowtie_{\mathcal{C}} S$	if $\mathcal{D}_{S\to R} \subseteq R_c$ and $\mathcal{C}$ is a primary-key in S
	Rule-V3b	$\Gamma^{\mathsf{V}}_{p,\mathcal{D}}(R \bowtie_{\mathcal{C}} S) \overset{*}{\Leftrightarrow} \Gamma^{\mathsf{V}}_{p,\mathcal{C}}(R) \bowtie_{\mathcal{C}} \Gamma^{\mathsf{V}}_{p,\mathcal{C}}(S)$	if $\mathcal{C} = \mathcal{D}$
	Rule-D3	$\Gamma^{\mathsf{D}}_{p,\mathcal{D},f}(R \bowtie_{\mathcal{C}} S) \overset{*}{\Leftrightarrow} \Gamma^{\mathsf{D}}_{p,\mathcal{D}_{S\to R},f}(R) \bowtie_{\mathcal{C}} S$	if $\mathcal{C} \subseteq \mathcal{D}_{S\to R} \subseteq R_c$ and $\mathcal{C}$ is a primary-key in S.

All sampler types *move*

Production exclusively uses substitution rules

- For faster QO
- Exploration rules need statistics (e.g., dv estimates) that are not broadly available

An example recurring job that switched to using samplers

metric	Orig.	Sampled	Change%
Duration (hours)	10.2	6.9	-32
Tot. compute hours	5118	3183	-37
Bytes read (TB)	340	340	0
Bytes written (TB)	172	170	-1.0
# Tasks (x 10^3)	85.5	85.4	-0.1

# Groups	650583
`MissedGroupFract.`	0.00
# Aggregates	24
`AvgAvgAbsOverAvgTrueV`	1.30%
`AvgAvgRelErr`	17.7%

More examples of recurring jobs that now use samplers

	Runtime reduction	**Resource Usage reduction**	**Accuracy: aggr. error μ_{RelE} (median), $\frac{\mu_{AbsE}}{\mu_{TrueV}}$**	**Accuracy: row miss**
Job1	68% (7Hr→ 2Hr)	67% (4907Hr→ 1594Hr)	1.7% (0.6%), 0.5% (well within natural variation)	0
Job2	59% (18Hr→ 7Hr)	62% (7595Hr→ 2912Hr)	8% (5%), 2%	0
Job3	40% (11Hr→ 6Hr)	41% (4237Hr→ 2516Hr)	19% (7%), 0.2% (small values)	0

Barriers to Adoption

- “Cannot tolerate any error.”
  - Each log entry can be thought of as a measurement sample; analytic answers inherently variable
  - Sampling okay iff additional variability due to sampling ≈ natural variation
- “Why change?”
  - Inertia and risk to change queries esp: legacy complex queries
  - Top-down push to reduce cost can help
- No guarantee on error for unseen inputs: open issue
- Cognitive overhead in deciding how to sample

Summary and References

- Approximate query processing can be a very effective tool in cost-accuracy tradeoff in Big data systems
- Automated approximation and guarantees are hard, if not impossible, for the complete surface of SQL
- Programmer-guided approximation for SQL as well as pre-computation based approximation for narrow DSL-s are viable
- References

  - Surajit Chaudhuri, Bolin Ding, Srikanth Kandula:Approximate Query Processing: No Silver Bullet. SIGMOD Conference 2017: 511-519
  - Srikanth Kandula, Kukjin Lee, Surajit Chaudhuri, Marc : with Approximating Queries in Microsoft's Production Big-Data Clusters. PVLDB 12(12): 2131-2142 (2019)
  - Srikanth Kandula, Anil Shanbhag, Aleksandar Vitorovic, Matthaios Olma, Robert Grandl, Surajit Chaudhuri, Bolin Ding: Quickr: Lazily Approximating Complex AdHoc Queries in BigData Clusters. SIGMOD Conference 2016: 631-646