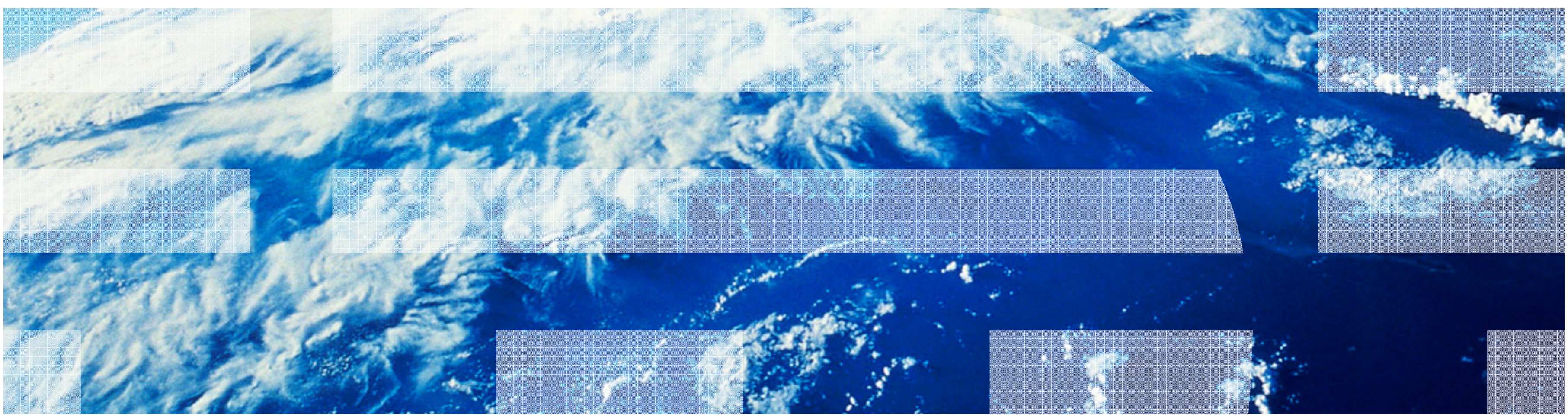
Scalable and Numerically Stable Descriptive Statistics in SystemML

Yuanyuan Tian, Shirish Tatikonda, Berthold Reinwald
IBM Almaden Research Center

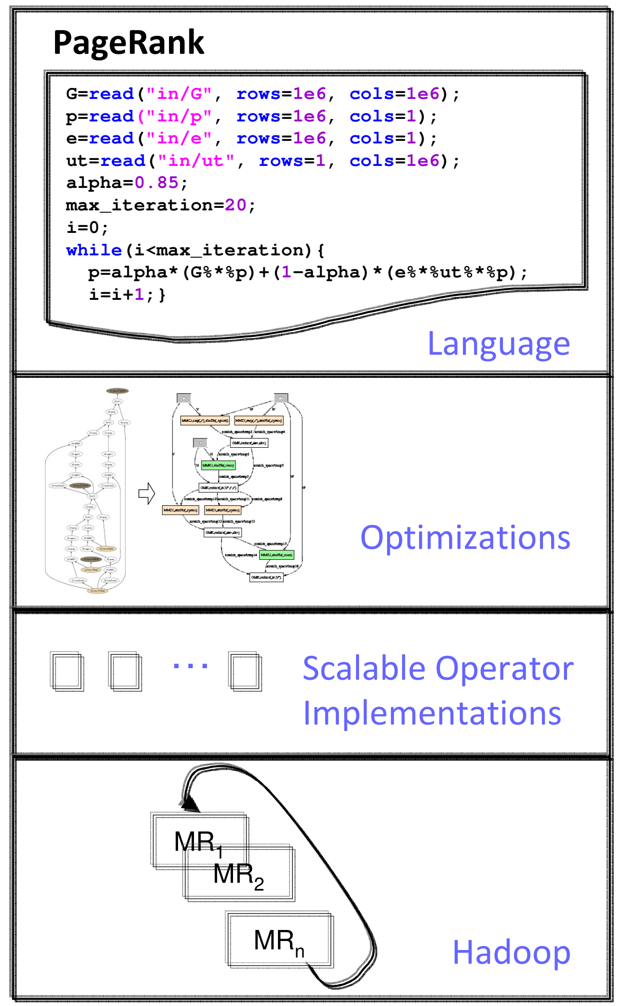


SystemML

- Pervasive need to enable machine learning (ML) on massive datasets
- Increasing interest in implementing a ML algorithms on MapReduce
- Directly implementing ML algorithms on MapReduce is challenging
- Solution: **SystemML** – A Declarative Approach to Machine Learning on MapReduce

SystemML Overview

- **High-level language** with ML specific constructs
  - Syntax is similar to R and Matlab
    - Matrix, vector and scalar data types
    - Linear algebra and mathematics operators
- **Optimizations** based on data and system characteristics
  - Cost-based and rule-based optimization
  - Job generation heuristics
- **Scalable** implementations on Hadoop
  - Handle large sparse data sets
  - Parallelization is *transparent* to end users



ICDE 2011

Example ML Algorithms Supported in SystemML

- Classification: linear SVMs, logistic regression
- Regression: linear regression
- Matrix Factorization: NMF, SVD, PCA
- Clustering: k-means
- Ranking: PageRank, HITS
- Data Exploration: **Descriptive Statistics** ⬅ **focus of this talk**
  - **Univariate Statistics**:
    - **Scale**: Sum, Mean, Harmonic mean, Geometric mean, Min, Max, Range, Median, Quantile, Inter-quartile-mean, Variance, Standard deviation, Coefficient of variance, Central moment, Skewness, Kurtosis, Standard error of mean, Standard error of skewness, Standard error of kurtosis
    - **Categorical**: Mode, Per-category frequencies
  - **Bivariate Statistics**:
    - **Scale-scale**: Covariance, Pearson correlation
    - **Scale-categorical**: Eta, ANOVA F measure
    - **Categorical-categorical**: Chi-square coefficient, Cramer's V, Spearman correlation

How hard is it?

- Seemingly trivial to implement in MapReduce
  - Most descriptive statistics can be written in certain summation form

Sum	$s = \sum x_i$
Mean	$\bar{x} = \frac{1}{n} \sum x_i$
Variance	$s^2 = \frac{1}{n-1} \sum (x_i - \bar{x})^2 = \frac{1}{n-1} \sum x_i^2 - \frac{1}{n(n-1)} (\sum x_i)^2$

- **Pitfall**: these straight forward implementations can lead to disasters in *numerical accuracy*

- Problem gets **worse** with increasing volumes of data

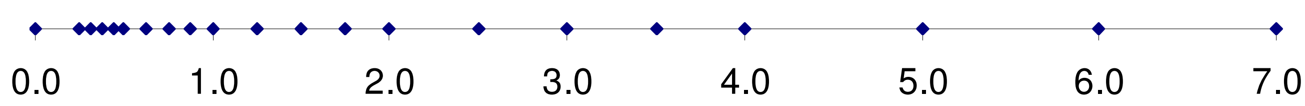
Background: Floating Point Numbers

- Source of Inaccuracy: finite precision arithmetic for floating point numbers

- Floating point number system $F \subset$ R: $y = \pm\beta^e \times .d_1d_2 \ldots d_t$ $(0 \le d_i \le \beta - 1)$
  - *base β,*
  - *precision t ,*
  - *exponent range* $e_{min} \le e \le e_{max}$
  - IEEE double: $\beta=2$, $t=53$, $e_{min}=-1021$, $e_{max}=1024$

- Floating point numbers are **not** equally spaced.
  - Example number system: $\beta = 2$, $t = 3$, $e_{min} = -1$, and $e_{max} = 3$

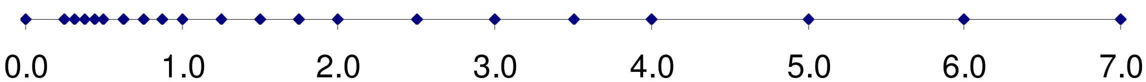
    F={ 0, ±0.25, ±0.3125, ±0.375, ±0.4375, ±0.5, ±0.625, ±0.75, ±0.875, ±1.0, ±1.25, ±1.5, ±1.75, ±2.0, ±2.5, ±3.0, ±3.5, ±4.0, ±5.0, ±6.0, ±7.0 }



Background: Numerical Accuracy

- Example number system: $\beta = 2$, $t = 3$, $e_{min} = -1$, and $e_{max} = 3$

  F={0, ±0.25, ±0.3125, ±0.375, ±0.4375, ±0.5, ±0.625, ±0.75, ±0.875, ±1.0, ±1.25, ±1.5, ±1.75, ±2.0, ±2.5, ±3.0, ±3.5, ±4.0, ±5.0, ±6.0, ±7.0 }



relative error = |x-x'|/|x|

- x: true value
- x': value in float point

- **round off:** 6.375 and 5.625

true value*: 6.375 and 5.625* ***computed****: 6.0 and 6.0* ***relative error****: 0.059 and 0.067*

- **big number + many small numbers:** 5.0 + 0.25 + 0.25 + … + 0.25 (eight 0.25) → ***pitfall for summation***

true value*: 7.0* ***computed****: 5.0* ***relative error****: 0.286*

5.0 + 0.25 = 5.25 round to 5.0

- **subtraction of 2 big and similar numbers:** 6.375 – 5.625 → ***catastrophic cancellation for subtraction***

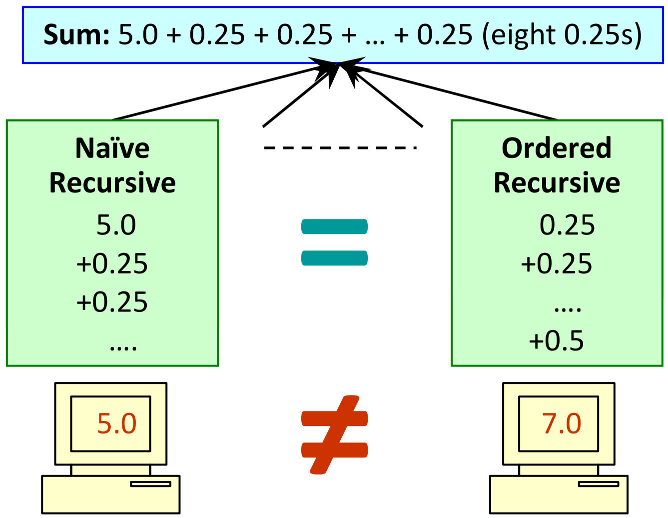
true value*: 0.75* ***computed****: 0* ***relative error****: 1.0*

6.375 round to 6.0

5.625 round to 6.0

Background: Numerical Stability

- Algebraically equivalent algorithms for the same calculation produce different results on digital computers
  - Some damp out errors
  - Some magnify errors



- *Numerical stability* is a desirable property of numerical algorithms
  - Algorithms that can be proven not to magnify approximation errors are called *numerically stable*

Importance of Numerical Stability

- Numerical stability issue has been largely ignored in big data processing
  - e.g. PIG and HIVE, are using well-known unstable algorithms for computing some basic statistics
- How about software floating point packages, e.g. BigDecimal?
  - Arbitrary precision, but *very* slow
    - +, -, *: **2 orders of magnitude** slower
    - /: **5 orders of magnitude** slower
- **Goal of this Talk**: share our experience on descriptive statistics algorithms for big data
  - **Scalable** – database people already understand
  - **Numerically Stable** – need more attention!!!!!

Numerically Stable Summation

- **Naïve Recursive**: unstable
- **Sorted Recursive**: better for nonnegative but needs an expensive sort
- **Kahan**: efficient and stable *[Kahan 1965]*
  - Recursive summation with a **correction term** to compensate rounding error

    (s', c') = ***KahanAdd*** (s, c , a)

s/s': old/new sum
c/c': old/new correct
a: number to add

a' = a + c
s' = s + a'
c' = a' - (s' - s)

Stability Property: relative error bound independent of problem size n, when n is less than $O(10^{16})$ for IEEE doubles

Kahan: 5.0 + 0.25 + 0.25 + … + 0.25

	sum	correction
	5.0	**0**
+	0.25	
	5.0	**0.25**
+	0.25	
	6.0	**-0.5**
+	0.25	
	6.0	**-0.25**
+	0.25	
	6.0	**0**
	…….	

- **MR Kahan in SystemML**:
  - Mapper: apply Kahan to compute partial sum and correction
  - Reducer: apply Kahan on partial results to compute sum

    (s, c) = ***KahanAdd*** (s_1, c_1+c_2, s_2)

s_1/s_2: partial sums
c_1/c_2: partial corrects
s: total sum
c: total correct

Our Proof: relative error bound independent of problem size n, as long as each mapper processes less than $O(10^{16})$ numbers

MR Kahan scales to larger problem size with numerical accuracy.

Numerically Stable Mean

- **Naïve sum/count**: unstable
- **Incremental**: stable *[Chan et al 1979]*

  n_1, n_2: partial counts, μ_1, μ_2: partial means

  $n = n_1 + n_2$

  $\delta = \mu_2 - \mu_1$

  $\mu = \mu_1 + \delta n_2/n$
- **MR Incremental**: adapt Incremental to MapReduce

  $n = n_1 + n_2$

  $\delta = \mu_2 - \mu_1$

  $\mu = \mu_1$ ¤ $\delta n_2/n$ (¤ -- *KahanAdd*, maintain a correction term for μ)

Numerically Stable Higher-Order Statistics

- **Higher order statistics**: central moment, variance, standard deviation, skewness, kurtosis (core: central moment $m_p = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^p$)
- **2-Pass**: 1st pass to compute mean, 2nd pass to compute m_p
  - Stable, but needs 2 scans of data
- **Textbook 1-Pass**: textbook rewrites $m_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \frac{1}{n^2}(\sum_{i=1}^{n}x_i)^2$
  - **Notoriously unstable** (due to *catastrophic cancellation*), can even produce negative result for m_p when p%2=0
  - Unfortunately, widely used in practice
- **Incremental**: stable* & 1 pass *[Bennett et al 2009]*

$$n = n_a + n_b, \delta = \mu_b - \mu_a, \mu = \mu_a \text{ ¤ } n_b \frac{\delta}{n}$$

$$M_p = M_{p,a} \text{ ¤ } M_{p,b} \text{ ¤ } \{\sum_{j=1}^{p-2}\binom{p}{j}[(-\frac{n_b}{n})^j M_{p-j,a} + (\frac{n_a}{n})^j M_{p-j,b}]\delta^j + (\frac{n_a n_b}{n}\delta)^p[\frac{1}{n_b^{p-1}} - (\frac{-1}{n_a})^{p-1}]\}$$

- **MR Incremental**: adapt Incremental to MapReduce
  - Use *KahanAdd*

Numerically Stable Covariance

- **2-Pass**: 1st pass to compute means, 2nd pass to compute covariance
  - Stable, but needs 2 scans of data
- **Textbook 1-Pass**: textbook rewrites $c_2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{n-1}\sum_{i=1}^{n} x_i y_i - \frac{1}{n(n-1)}(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)$
  - Notoriously unstable (catastrophic cancellation)
  - Unfortunately, widely used in practice
- **Incremental**: practically stable & 1 pass *[Bennett et al 2009]*

$$n = n_a + n_b,\ \delta_x = \mu_{x,b} - \mu_{x,a},\ \mu_x = \mu_{x,a} \text{ ¤ } n_b \frac{\delta_x}{n},\ \delta_y = \mu_{y,b} - \mu_{y,a},\ \mu_y = \mu_{y,a} \text{ ¤ } n_b \frac{\delta_y}{n}$$

$$C = C_a \text{ ¤ } C_b \text{ ¤ } \frac{n_a n_b}{n} \delta_x \delta_y$$

- **MR Incremental**: adapt Incremental to MapReduce
  - Use *KahanAdd*

Experiment Results

Example Univariate Statistics

Ranges : R1= [1.0 – 1.5), R2= [1000.0 – 1000.5), R3= [1000000.0 – 1000000.5)

	Size (million)	Sum		Mean		Variance		Standard Deviation	
Range		SML	Naïve	SML	Naïve	SML	Textbook	SML	Textbook
R2	10	**16.8**	14.4	**16.5**	14.4	**15.4**	5.9	**15.9**	6.2
	100	**16.1**	13.4	**16.9**	13.4	**15.6**	5.3	**15.8**	5.6
	1000	**16.6**	13.1	**16.4**	13.1	**16.2**	4.9	**16.4**	5.2
R3	10	**15.9**	14.0	**16.3**	14.0	**14.4**	**0**	**14.7**	**0**
	100	**16.0**	13.1	**16.9**	13.1	**12.9**	**Negative**	**13.2**	**NA**
	1000	**16.3**	12.9	**16.5**	12.9	**13.2**	**Negative**	**13.5**	**NA**

Example Bivariate Statistics

	Size (million)	CoVariance		Pearson-R	
Range		SML	Textbook	SML	Textbook
R1 vs R2	10	**15.0**	8.4	**15.1**	6.2
	100	**15.6**	8.5	**15.4**	6.4
	1000	**16.0**	8.7	**15.7**	6.2
R2 vs R3	10	**13.5**	3.0	**13.5**	3.0
	100	**12.8**	2.8	**12.7**	**NA**
	1000	**13.6**	3.9	**13.8**	**NA**

Significantly better accuracy!
No sacrifice to performance!

- **Data Sets**: uniform distribution in 3 ranges
  - R1= [1.0 – 1.5), R2= [1000.0 – 1000.5), R3= [1000000.0 – 1000000.5)
  - modeled after NIST StRD benchmark
- **Accuracy Measure**:
  - ***LRE*** = *log*(relative error)
  - # significant digits that match between the computed value and the true value.
  - true value: produced by BigDecimal with precision 1000.

Lessons Learned (1/2)

- Many existing numerical stable techniques can be adapted to the distributed environment
- Divide-and-conquer design helps in scaling to larger data sets while achieving good numerical accuracy
  - e.g. MR Kahan can handle more data than Kahan with numerical accuracy
- Kahan technique is useful beyond simple summation

R1 10 million points

Variance		Std	
¤	+	¤	+
16.0	13.5	**15.9**	13.8

R1 vs R2 10 million points

Covariance		Pearson-R	
¤	+	¤	+
15.0	14.2	**15.1**	13.0

Lessons Learned (2/2)

- Shifting can be used for improved accuracy
  - Accuracy degrades as magnitude of values increases
  - Achieve better accuracy by shifting the data points by a constant prior to computation

10 million points

	Sum		Mean		Variance		Standard Deviation	
Range	SML	Naïve	SML	Naïve	SML	Textbook	SML	Textbook
1000 – 1000.5	**16.8**	14.4	**16.5**	14.4	**15.4**	5.9	**15.9**	6.2
1000,000 – 1000,000.5	**15.9**	14.0	**16.3**	14.0	**14.4**	**0**	**14.7**	**0**

- Performance need not be sacrificed for accuracy

- Recommended Reading:
  - Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2nd edition, 2002.
- **Thanks! and Questions?**