A Closer Look at Variance Implementations In Modern Database Systems

Niranjan Kamat  Arnab Nandi
Computer Science & Engineering
The Ohio State University
{kamatn,arnab}@cse.osu.edu

ABSTRACT
Variance is a popular and often necessary component of sampled aggregation queries. It is typically used as a secondary measure to ascertain statistical properties of the result such as its error. Yet, it is more expensive to compute than simple, primary measures such as \text{SUM}, \text{MEAN}, and \text{COUNT}.

There exist numerous techniques to compute variance. While the definition of variance is considered to require multiple passes on the data, other mathematical representations can compute the value in a single pass. Some single-pass representations, however, can suffer from severe precision loss, especially for large number of data points.

In this paper, we study variance implementations in various real-world systems and find that major database systems such as PostgreSQL 9.4 and most likely System X, a major commercially used closed-source database, use a representation that is efficient, but suffers from floating point precision loss resulting from catastrophic cancellation. We note deficiencies in another popular representation, used by databases such as MySQL and Impala, that suffers from not being distributive and therefore cannot take advantage of modern parallel computational resources. We review literature over the past five decades on variance calculation in both the statistics and database communities, and summarize recommendations on implementing variance functions in various settings, such as approximate query processing and large-scale distributed aggregation.

1. INTRODUCTION
Variance is an important aggregate function supported in most major modern databases and is an essential tool in sampling-based aggregation queries. Typically used as a secondary measure, it augments measures such as \text{AVERAGE} and provides an insight into the distribution of the data beyond the primary measure. Computation of variance, however, can suffer from precision loss: it is particularly susceptible when the variance is much smaller than the mean \cite{1}.

There exist several techniques to compute variance. The standard variance formula uses two passes to provide an accurate estimate (Two Pass). However, as is apparent, this can be prohibitively expensive for large datasets. Some other techniques using a single pass over the data store basic statistics such as the count, sum, and sum of squares. One such formula, although fast, is known to suffer from precision loss (Textbook One Pass) due to catastrophic cancellation \cite{4}, an undesirable effect of a floating point operation that causes the relative error to increase far more than the absolute error. Figure 2 demonstrates this problem.

Another formula (Updating), which has been recommended by Knuth \cite{6}, has found a strong foothold in the database community, with numerous implementations citing Knuth in their documentation. However, this formula is constrained by the fact that it can only incorporate a single data point to the current running estimates. It is unable to combine the estimates of different subsets of data.

Given the rise of large-scale data processing, with massive multi-core support and availability of GPUs, it is prudent to consider using representations that can be combined at a larger scale instead of incrementally adding a single data point, such as Pairwise Updating. Further, Pairwise Updat-
ing is also known to have a better precision, as shown by Chan et al. [1] for single precision input (and, as verified in the following sections, for double precision as well.)

**Contributions & Outline:**

- We analyze source code for various open source database systems to catalog usage of different variance formulas (Table 2).
- We experiment with different closed source and open source databases to investigate precision loss issues. We find that precision of PostgreSQL and System X deteriorates the most.
- Based on our experiments, we experimentally study representations used by two closed source implementations (System X and System Y). After looking at the PostgreSQL source code, we can verify that it uses *Textbook One Pass*, and hypothesize that System X does so as well (or uses a similar variant).

In the next subsection, we look at the adverse effects of imprecise variance calculation. Section 2 presents the different variance representations and their properties. We then detail the representations used by modern databases in Section 3. Section 4 lists our analysis of the behavior of the different formulas (double precision input compared with single precision in Chan et al. [1]) as well as study the databases empirically as the input dataset is made more prone to precision loss. Finally, we conclude with our recommendations for variance representation in current environments.

### 1.1 Impact of Variance Calculations

Due to the pervasive use of variance, a loss of precision can have an impact in a variety of different domains. In the following paragraphs, we look at some use cases where the lack of precision in variance calculation can have adverse consequences.

**Incorrect Output:** In its simplest form, it is possible to experimentally observe the loss of precision as incorrect output. In order to illustrate the pitfalls in using *Textbook One Pass*, data points were generated from a Uniform(0,1) distribution and shifted by $10^\text{Shift Exponent}$ for $\text{Shift Exponent}$ varying from 1 to 14. Thus, the variance obtained by using a shift exponent should be similar to the one without any exponent shift. We verify this by adding and subtracting the shift exponent and note that the variance of the resultant dataset was close to the true sample variance.

Figure 2 shows that PostgreSQL 9.3 and System X suffer from variance calculations being susceptible to precision loss since the variance should not show such an increase. We know that PostgreSQL uses *Textbook One Pass* and the pattern of the erroneous calculations displayed by both of them hints towards System X using the same variance calculation as PostgreSQL.

In contrast, other database systems suffered minor precision loss, as expected: these results are not shown since they do not add any additional information to the figure. It should be noted that System Y was found to be highly immune to precision loss.

**Visualization:** Erroneous variance calculation, however small, can have a notable impact on visualizations. As a demonstration, we show the results of a repetition of the above experiment in Figure 1 and depict the sample mean and the confidence interval. Due to precision loss, we observe inaccurate results for higher shift values for PostgreSQL 9.3 and System X. While the error bars should be similar throughout, they vary widely and inaccurately instead. Error bars for System Y are (correctly) low throughout.

**Negative variance:** It is possible for the variance to be negative while using *Textbook One Pass* – an impossible result (Table 3). We observed in the PostgreSQL source code that the implementation checks if the variance is negative and sets it to zero, if true. Figure 1 shows numerous values of 0 for PostgreSQL (shift exponent 8, 9, and 12) and also for System X (shift exponents 10 and 11), providing evidence of System X employing a similar strategy for handling negative variance values and using *Textbook One Pass*.

**Decision support systems:** As a building block in popular algorithms, flaws in variance implementations can have far-reaching impacts, e.g., in hypothesis testing, which is an integral part of numerous decision support systems. Having imprecise or incorrect variance estimates can greatly change the result of hypothesis testing.

- **Load Failure:** Consider the case of a sample 2ailed t-test with the shift exponent of 8 using the output of PostgreSQL as given in Figure 1. Let the null hypothesis be as follows:
  \[ H_0: \mu = 10^8 + 0.483594 \] (sample mean)
  And the alternate hypothesis as:
  \[ H_a: \mu \neq 10^8 + 0.483594 \]
  The t-statistic can be given by \( \frac{\bar{x} - \mu}{s/n} \), where \( \mu \) is the hypothesized mean estimate, \( \bar{x} \) is the sample mean, \( s \) is the sample standard deviation, and \( n \) is the sample size. In this case, since \( s = 0 \), the t-test will fail by reporting an error.

- **Silent Failure:** We now look at the more harmful error of silent failures. Let us consider the sample with shift exponent of 12 and use the output of System X. Again, let the hypotheses be as follows:
  \[ H_0: \mu = X \]
  \[ H_a: \mu \neq X \]
  where \( X \) is the hypothesized population mean. Let \( \alpha \) (con-
The t-statistic will be equal to \( \frac{\text{sample mean} - \text{mean}}{\text{standard error}} \) instead of being \( \frac{\text{sample mean} - \text{mean}}{\text{standard error}} \). If the variance calculation were correct, the null hypothesis will end up not being rejected.

Data Mining: Variance is an important tool in statistical analysis and machine learning algorithms such as Gaussian Naive Bayes, or Mixture-of-Gaussians-based algorithms such as background modeling, clustering, or topic modeling. For example, we found usage of Textbook One Pass within a graphics library of the R language [5]. Similarly, MADlib [3] was also found to have a call to the PostgreSQL variance function: thus, an erroneous calculation of variance can extend from the underlying databases to the systems built on top of them.

### 2. DIFFERENT WAYS TO CALCULATE VARIANCE

Table 1 presents the common variance representations [1]. We have also described Total Variance, for which we could not find a reference in accessible literature. We use a similar naming convention to that used by Chan et al. [1]: \( S \) stands for the sum of squares. The sample variance can be given by \( \frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N-1} \), where \( N \) is the sample size. \( x_i \) is the \( i \)th data point. \( \bar{x} \) is the sample mean. \( m_{n,m} \) is the mean of the data points with index \( m \) to \( n \) (both inclusive). \( T_{n,m} \) is the total of the data points with index \( m \) to \( n \) (both inclusive). Finally, \( k \) represents a constant.

Textbook One Pass can be computationally dangerous as the quantities \( \sum_{i=1}^{N} x_i^2 \) and \( \frac{1}{N} (\sum_{i=1}^{N} x_i)^2 \) can nearly cancel each other out. The Updating formula refers to the nearly identical formulas for unweighted sum of squares used by Welford et al. [7], West et al. [8], and, and, Hanson et al. [2]. The Pairwise Updating formula uses \( O(\log(N)) \) storage and reduce the relative errors from \( O(N) \) to \( O(\log(N)) \) [1]. Shifting the data by an exact or approximate value of \( \bar{x} \) can also result in substantial accuracy gains [1].

#### 2.1 Properties

While Chan et al. [1] note the accuracy, passes and storage required for most of the formulas given in Table 1 (other than Total Variance), their classification as being distributive and thus the ability to be parallelized, has not been explicitly listed before, which we do.

The accuracy of Shifted One Pass depends on the accuracy of the estimate of the mean. The second and third passes required by Total Variance are over the groups obtained as a result of the first pass, and not over the number of tuples and thus can vary widely based on the group sizes. As we can see Pairwise Updating is the only representation giving accurate results which is highly parallelizable, and yet uses a single pass.

Additionally, as we will see in Section 4, the precision of Total Variance (which has not been studied before) is slightly better than that of Updating Pairwise, which typically has the best precision amongst all single pass algorithms. We note two major differences between the two formulas. First, Total Variance calculates the overall mean before combining the variances of the different groups whereas Updating Pairwise keeps combining two groups and hence Total Variance has slightly better precision. Second, Updating Pairwise computation is performed in a tree-like fashion in a single pass over the data whereas Total Variance uses 1-3 passes. As a side note, amongst the different representations, Two Pass, Total Variance and Textbook One Pass are the only ones that can be represented using a standard SQL query.

#### 2.2 Current Recommendation Guidelines

Chan et al. [1] provide detailed recommendation guidelines on the use of different variance formulas. We note that they recommend usage of Pairwise Updating for combining variances across multiple processors since it reduces the errors and is massively parallelizable if extra \( O(\log(N)) \) space is available. Further, it is also the safest (least precision loss) algorithm to use within each processor.

#### 2.3 Extensibility to Other Measures

Standard deviation, standard error and co-efficient of variation are important statistical measures and need variance estimation. Given the variety of variance representations, these measures too will have the properties that the corresponding formulas hold. Similarly, the properties also extend to any user-defined measure whose variance can be expressed in a closed form as a function of the variance of one of the measure dimensions. For example, for a user-defined measure given by \( a * \text{VAR}(\text{Agg}_1) + b \), where \( a \) and \( b \) are constants and \( \text{Agg}_1 \) is a measure dimension, the variance of the measure can be given in closed form as \( a^2 * \text{VAR} (\text{Agg}_1) \).
We note that obtaining a closed form solution to the variance of measures such as holistic measures or complex measures is, however, not always possible, with bootstrapping being a popular choice for estimating the variance.

3. VARIANCE IMPLEMENTATIONS IN MODERN DATABASE SYSTEMS

Given the variety of variance formulas, we now survey various open source databases to find out which formulas are used by them to compute variance. We also conjecture about the closed source implementations using experimental validation. Table 2 lists the formula used in each database system.

<table>
<thead>
<tr>
<th>Database</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL 9.4.4</td>
<td>Textbook One Pass</td>
</tr>
<tr>
<td>MySQL 5.7</td>
<td>Updating</td>
</tr>
<tr>
<td>Impala 2.1.5</td>
<td>Updating</td>
</tr>
<tr>
<td>Hive 1.2.1</td>
<td>Updating Pairwise</td>
</tr>
<tr>
<td>Spark 1.4.1</td>
<td>Updating, Updating Pairwise</td>
</tr>
<tr>
<td>SQLite</td>
<td>No Variance Support</td>
</tr>
<tr>
<td>System X</td>
<td>Textbook One-pass (Conjecture)</td>
</tr>
<tr>
<td>System Y</td>
<td>Higher precision variables (Guess) Cannot conjecture about formula</td>
</tr>
</tbody>
</table>

Table 2: Variance Implementations in Modern Databases

PostgreSQL uses Textbook One Pass and is thus susceptible to precision loss, as confirmed in our empirical evaluation. MySQL and Impala use Knuth’s modification [6] of Welford’s updating formula. Thus, MySQL and Impala can only process a single additional data point and thus cannot avail of the parallelization presented by Pairwise Updating. On the other hand, Spark 1.4.1 uses a modified version of Updating Pairwise for combining sum of squares. Even though the source code for System X is not available, its behavior was similar to that shown by PostgreSQL in our experiments. Amongst all the systems, System Y was found to have the best precision. We hypothesize that it uses higher precision variables, but cannot make any conjecture about the exact representation.

Thus, we conclude that PostgreSQL and System X use representations likely to suffer from precision loss. Further, other than Spark and Hive, the other database engines lack the ability to combine variance of different groups and are inherently non-parallelizable.

4. EXPERIMENTAL ANALYSIS

Chan et al. [1] have looked at the precision of different algorithms given single precision input. We present the precision values for double precision input. Further, we could not find any discussion on the precision of the Total Variance formula, which we cover. We also look at the precision in the variance calculation offered by different databases.

4.1 Impact of Shift on Precision

We implemented formulas in the previous section using Java 7, and evaluated their precision for varying shift exponents. The formulas were run over a synthetic dataset of 10000 samples with double precision from Uniform(0, 1) with the resulting variance of \( \frac{1}{12} \). As before, these samples were shifted by adding values ranging from \( 10^1 \) to \( 10^{15} \). For each variance calculation, the number of correct decimal places was recorded. The results were averaged over 100 runs. The initial group size was set at 10 for VarianceCommon. We present our findings in Figure 3, where the Y-axis represents the number of correct decimal digits (non-fractional part of the result was 0). The results are as expected [1], and Textbook One Pass is clearly impacted by the shift exponent.

![Figure 3: Precision in Variance Calculations: With increasing shift exponent, all representations experience precision loss – though some experience it more severely.](image)

4.2 Impact of Shift on Database Implementations

We now look at the variance results of different databases in the presence of similar datasets that may cause precision issues. 100 points were chosen from the aforementioned Uniform(0, 1) distribution with exponential decimal shifts in the same fashion as in the motivating example\(^1\). Each experiment was repeated 10 times. We noted the number of correct decimal digits in Figure 4 as a measure of precision of the variance calculation. As we can see, the precision

\(^1\) Care was taken for different systems to have the same comparable configuration in consideration with the data types.
loss follows a similar pattern in both System X and PostgresQL. Impala and MySQL which use the same variance representation have similar error pattern as well.

### 4.3 Textbook One Pass in Action

To further illustrate the catastrophic cancellation occurring as a result of Textbook One Pass, we present below the corresponding mantissa of the two expressions that compose it, in double precision. We consider a random sample of size 10000 generated from Uniform(0,1) distribution that has been shifted by exponents ranging from 1 to 7. Note that Textbook One Pass calculates the sum of squares as \( S = S_1 - S_2 \) where \( S_1 = \sum_{i=1}^{N} x_i^2 \) and \( S_2 = \frac{1}{2} \left( \sum_{i=1}^{N} x_i \right)^2 \). Thus, we can see that increasing number of bits start becoming the same in the mantissa of \( S_1 \) and \( S_2 \), until the all precision is lost for the shift exponent of 6.

<table>
<thead>
<tr>
<th>Shift</th>
<th>mantissa((S_1))</th>
<th>mantissa((S_2))</th>
<th>(S = S_1 - S_2)</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0xa7677ed386b82</td>
<td>0x3f74ce8319d49</td>
<td>831.584227247941</td>
<td>0.08316671895836578</td>
</tr>
<tr>
<td>1</td>
<td>0xda6e1lcd8b24</td>
<td>0x3f74ce8319d49</td>
<td>831.58422725085</td>
<td>0.08316671894437384</td>
</tr>
<tr>
<td>2</td>
<td>0x815fe01176cb</td>
<td>0x815fe01176cb</td>
<td>831.584228646994</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>3</td>
<td>0xe2a531c0e87f3</td>
<td>0xe2a531c0e87f3</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>4</td>
<td>0x2a531c0d87f3f</td>
<td>0x2a531c0d87f3f</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>5</td>
<td>0x2a531a6dbd3c9</td>
<td>0x2a531a6dbd3c9</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>6</td>
<td>0x1c37a6532f25c</td>
<td>0x1c37a6532f25c</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>7</td>
<td>0x1c37a6532f25c</td>
<td>0x1c37a6532f25c</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
<tr>
<td>8</td>
<td>0x5af1d7c632dd</td>
<td>0x5af1d7c632dd</td>
<td>831.5840644836426</td>
<td>0.08316671894404526</td>
</tr>
</tbody>
</table>

Table 3: Example of Application of Textbook One Pass

5. CONCLUSION & RECOMMENDATIONS

Floating point precision can cause information loss in both data measurement as well as data storage. This problem is further exacerbated to varying degrees by different variance calculation formulas. 

Pairwise Updating has been recommended from the perspective of precision as well as speed in the presence of multiple processors or cores [1]. Further, the precision issues associated with Textbook One Pass have been well documented. However, we have seen that PostgreSQL and likely System X still use it; this implementation should either be changed or documented readily for end users. Although Textbook One Pass suffers from catastrophic precision loss only in the presence of some specific datasets, it is not entirely implausible to encounter such values, especially with growing data volume and variability. Thus, we recommend for safety and therefore to not use it as a general purpose method. However, we note that its computation is extremely fast and parallelizable and may be practical in fast-but-inaccurate-style computations. Further, in the case that \(O(\log(N))\) memory is available, it would be wise to use Pairwise Updating instead of Updating considering the tremendous speedups and increase in precision provided by it. Additionally, this representation is also highly suitable for GPUs.

Finally, due to variability in the datasets, in contrast to [1], we suggest the database providers allow the user to choose between the different formulas for their applications based on their need for speed and accuracy since the user might have a better understanding of the data and can choose wisely. This argument is further reinforced by the fact that PostgreSQL and System X have used Textbook One Pass for a few years and have been generally accepted by real-world users. Further, in the use case of approximate querying, it is assumed that the results will not be accurate and in that case due to the speedups offered by Textbook One Pass, it might be more worthwhile to consider it.

6. REFERENCES