

Numerically Stable, Single-Pass, Parallel Statistics Algorithms

Janine Bennett ¹, Ray Grout ², Philippe Pébay ³, Diana Roe ⁴, David Thompson ⁵

*Sandia National Laboratories
M.S. 9159, P.O. Box 969
Livermore, CA 94551, U.S.A.*

¹ jcbenne@sandia.gov ² rwgrout@sandia.gov ³ pppebay@sandia.gov
⁴ dcroe@sandia.gov ⁵ dcthomp@sandia.gov

Abstract—Statistical analysis is widely used for countless scientific applications in order to analyze and infer meaning from data. A key challenge of any statistical analysis package aimed at large-scale, distributed data is to address the orthogonal issues of parallel scalability and numerical stability. In this paper we derive a series of formulas that allow for single-pass, yet numerically robust, pairwise parallel and incremental updates of both arbitrary-order centered statistical moments and co-moments. Using these formulas, we have built an open source parallel statistics framework that performs principal component analysis (PCA) in addition to computing descriptive, correlative, and multi-correlative statistics. The results of a scalability study demonstrate numerically stable, near-optimal scalability on up to 128 processes and results are presented in which the statistical framework is used to process large-scale turbulent combustion simulation data with 1500 processes.

I. INTRODUCTION

Centered statistical moments are one of the most widely used tools in descriptive statistics. It is therefore essential for statistical analysis packages that robust and efficient algorithms be devised and implemented. However, in this context, robustness and speed of execution tend to be orthogonal. For instance, the naïve single-pass algorithm for calculating centered statistical moments utilizes the sum of powers, leading to unacceptable numerical instability. This can be easily verified, e.g., by computing the variance of $\{1 - \delta, 1 + \delta\}$, where δ is smaller than the square root of the machine’s epsilon but bigger than the machine epsilon itself. In this case, the naïve single-pass algorithm will calculate a sum of squares that is equal to $2(1 + \delta^2)$ as the terms linear in δ will cancel each other out. However, because δ^2 is smaller than the machine epsilon, this will result in a sum of squares that is indistinguishable from 1 and the final calculated variance will be 0. A more careful approach would have calculated, depending on whether the biased or the unbiased estimator were used, δ^2 or $2\delta^2$. Note that, contrary to what is frequently asserted due to a widespread misconception, both quantities are fully distinguishable from 0. This simple example demonstrates an effect that occurs frequently in datasets with many small deviations

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy’s National Nuclear Security Administration under Contract DE-AC04-94AL85000.

accompanied by several large observations that nearly cancel each other out.

Nonetheless, this numerically unstable approach is still used even in recent parallel statistics literature; for example [1] still use it for the calculation of the covariance matrix prior to Principal Component Analysis (PCA). More sophisticated on-line algorithms have been developed, however the update formulas used by these algorithms have only been derived for up to fourth-order centered moments.

Two-pass algorithms can be used to address the issue of numerical stability, at the cost of increased execution time. In two-pass algorithms the mean of the data set is calculated in the first pass. In the second pass, the required powers of the deviations to the mean are computed. Two-pass algorithms are, in general, much more stable than the naïve single-pass approach. However, execution speed is severely impaired as each data point must be accessed twice. In the context of the statistical analysis of large-scale, distributed data sets, two-pass algorithms become entirely impractical as costs of distributed memory access massively dominate computation costs. Moreover, two-pass algorithms are not amenable to streaming processing.

In this work we derive general update formulas that allow for pairwise and incremental updates of arbitrary-order centered statistical moments. Additionally, we introduce parallel update formulas for both pairwise and incremental updates of covariance. Using these single-pass formulas, we have built an open source statistical framework for large-scale, distributed data sets that exhibits near optimal parallel speed up properties. The rest of this paper is outlined as follows: In Section II we discuss related work. Single-pass general update formulas are derived in Section III and our statistical framework is introduced in Section IV. A scalability study is provided in Section V that verifies the correctness of our algorithm and demonstrates linear speed-up properties with up to 128 processes. Results are included in which the framework is used to compute principal component analysis on large-scale turbulent combustion simulation data on 1500 processes. Finally, conclusions and future work are discussed in Section VI.

II. RELATED WORK

In the case of the variance, single-pass algorithms have been in use for some time [2]. Given a data set \mathcal{S}_1 with finite and non-negative cardinality $n-1$ and mean μ_1 , the mean μ of $\mathcal{S} = \mathcal{S}_1 \cup \{y\}$, where y is an additional data point is:

$$\mu = \mu_1 + \frac{y - \mu_1}{n}, \quad (\text{II.1})$$

Using (II.1), the following recurrence formula for the sum $M_{2,\mathcal{S}} = \sum_{x \in \mathcal{S}} (x - \mu)^2$ can then be used:

$$M_{2,\mathcal{S}} = M_{2,\mathcal{S}_1} + (y - \mu_1)(y - \mu). \quad (\text{II.2})$$

from whence, the unbiased estimator for the variance of \mathcal{S} is readily obtained as $\sigma_{n-1,\mathcal{S}}^2 = \frac{1}{n-1} M_{2,\mathcal{S}}$.

A more general set of pairwise update formulas was introduced in [3]. Given two data sets \mathcal{S}_1 and \mathcal{S}_2 , with respective cardinalities n_1 and n_2 and respective means μ_1 and μ_2 , the mean μ of $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ is:

$$\mu = \mu_1 + n_2 \frac{\delta_{2,1}}{n}, \quad (\text{II.3})$$

with $\delta_{2,1} = \mu_2 - \mu_1$.

The recurrence formula for $M_{2,\mathcal{S}}$ can then be used directly:

$$M_{2,\mathcal{S}} = M_{2,\mathcal{S}_1} + M_{2,\mathcal{S}_2} + n_1 n_2 \frac{\delta_{2,1}^2}{n}, \quad (\text{II.4})$$

The incremental update formulas (II.1) and (II.2) are specific cases of the pairwise update formulas (II.3) and (II.4) derived by [3] (occurring when \mathcal{S}_2 is reduced to a singleton $\{y\}$).

The formulas for third- and fourth-order moments, which are needed to calculate skewness and kurtosis of the data set, were derived by [4], in the form of pairwise update formulas for $M_{3,\mathcal{S}} = \sum_{x \in \mathcal{S}} (x - \mu)^3$ and $M_{4,\mathcal{S}} = \sum_{x \in \mathcal{S}} (x - \mu)^4$. These pairwise update formulas can be readily specialized to the incremental update form where one of the two subsets is reduced to a singleton, in a similar fashion to the specialization of (II.4) to (II.2).

The results recalled thus far provide the necessary formulas for efficient robust parallel calculations of statistical moments up to the fourth order, thus providing robust single-pass methods to compute the mean, variance, skewness, and kurtosis of a large distributed data set. Although this probably covers the needs of the vast majority of traditional applications of descriptive statistics, more recent applications require that higher-order statistics be used. For example, in the context of signal processing, [5] makes direct use of the eighth-order centered moments for identifying cell phone modulation schemes, and [6] proposes using arbitrarily high (but in practice limited to the sixth), even-ordered centered statistical moments to identify chromatic dispersion in fiber optic network devices.

III. SINGLE-PASS UPDATE FORMULAS

In this section we introduce the single-pass (both pairwise and incremental) update formulas for arbitrary-order centered moments and for covariance.

A. Arbitrary-Order Centered Moments

The notation from Section II are retained.

Proposition III.1. *Let p be a natural number greater than 1. Then, the pairwise update formula for $M_{p,\mathcal{S}} = \sum_{x \in \mathcal{S}} (x - \mu)^p$ is:*

$$\begin{aligned} M_{p,\mathcal{S}} &= M_{p,\mathcal{S}_1} + M_{p,\mathcal{S}_2} \\ &+ \sum_{k=1}^{p-2} \binom{k}{p} \left[\left(-\frac{n_2}{n} \right)^k M_{p-k,\mathcal{S}_1} + \left(\frac{n_1}{n} \right)^k M_{p-k,\mathcal{S}_2} \right] \delta_{2,1}^k \\ &+ \left(\frac{n_1 n_2}{n} \delta_{2,1} \right)^p \left[\frac{1}{n_2^{p-1}} - \left(\frac{-1}{n_1} \right)^{p-1} \right]. \end{aligned} \quad (\text{III.1})$$

Proof: By definition of $M_{p,\mathcal{S}}$, and because $\{\mathcal{S}_1, \mathcal{S}_2\}$ is a partition of \mathcal{S} , one has

$$\begin{aligned} M_{p,\mathcal{S}} &= \sum_{x \in \mathcal{S}} (x - \mu)^p \quad (\text{III.2}) \\ &= \sum_{x \in \mathcal{S}_1} (x - \mu)^p + \sum_{x \in \mathcal{S}_2} (x - \mu)^p \\ &= \sum_{x \in \mathcal{S}_1} \left(x - \frac{n_1 \mu_1 + n_2 \mu_2}{n} \right)^p + \sum_{x \in \mathcal{S}_2} \left(x - \frac{n_1 \mu_1 + n_2 \mu_2}{n} \right)^p \\ &= \sum_{x \in \mathcal{S}_1} \left(x - \mu_1 - \frac{n_2}{n} \delta_{2,1} \right)^p + \sum_{x \in \mathcal{S}_2} \left(x - \mu_2 + \frac{n_1}{n} \delta_{2,1} \right)^p \\ &= \sum_{k=0}^p \binom{k}{p} \left[M_{p-k,\mathcal{S}_1} \left(-\frac{n_2}{n} \delta_{2,1} \right)^k + M_{p-k,\mathcal{S}_2} \left(\frac{n_1}{n} \delta_{2,1} \right)^k \right], \end{aligned}$$

thanks to the commutativity of summations over finite sets, which allows us to permute $\sum_{k=0}^p$ with $\sum_{x \in \mathcal{S}_1}$ and $\sum_{x \in \mathcal{S}_2}$. Now, a few simplifications are in order. First, the $k=0$ term of the above summation is simply $M_{p,\mathcal{S}_1} + M_{p,\mathcal{S}_2}$. Second, by definition, both M_{1,\mathcal{S}_1} and M_{1,\mathcal{S}_2} are zero, thus eliminating the $k=p-1$ term from the summation. Last, $M_{0,\mathcal{S}_1} = n_1$, $M_{0,\mathcal{S}_2} = n_2$, and

$$\begin{aligned} &n_1 \left(-\frac{n_2}{n} \delta_{2,1} \right)^p + n_2 \left(\frac{n_1}{n} \delta_{2,1} \right)^p \\ &= \left(\frac{n_1 n_2}{n} \delta_{2,1} \right)^p \left[\frac{(-1)^p}{n_1^{p-1}} + \frac{1}{n_2^{p-1}} \right] \\ &= \left(\frac{n_1 n_2}{n} \delta_{2,1} \right)^p \left[\frac{1}{n_2^{p-1}} - \left(\frac{-1}{n_1} \right)^{p-1} \right]. \end{aligned} \quad (\text{III.3})$$

Therefore, by substituting (III.3) for the $k=p$ term in (III.2), one finally obtains (III.1). ■

One can readily verify that the pairwise update formulas for M_3 and M_4 indicated in [4], are particular cases of Proposition III.1 when $p=3$ and $p=4$, respectively.

Corollary III.2. *In the case where \mathcal{S}_2 is reduced to a singleton $\{y\}$, and denoting $\delta = y - \mu_1$, Proposition III.1 reduces to the incremental update formula for $\mathcal{S} = \mathcal{S}_1 \cup \{y\}$ as follows:*

$$\begin{aligned} M_{p,\mathcal{S}} &= M_{p,\mathcal{S}_1} + \sum_{k=1}^{p-2} \binom{k}{p} M_{p-k,\mathcal{S}_1} \left(\frac{-\delta}{n} \right)^k \quad (\text{III.4}) \\ &+ \left(\frac{(n-1)}{n} \delta \right)^p \left[1 - \left(\frac{-1}{n-1} \right)^{p-1} \right] \end{aligned}$$

Proof: Corollary III.2 is a straightforward specialization of Proposition III.1, obtained when $n_1 = n - 1$ and $n_2 = 1$. In this case, $\delta_{2,1} = \delta$ and each M_{p,\mathcal{S}_2} vanishes since $\mu_2 = y$, and thus (III.1) is immediately simplified into (III.4). ■

Remark III.1. By noticing that (II.1) is equivalent to

$$y - \mu = \frac{n-1}{n}(y - \mu_1) \quad (\text{III.5})$$

one directly retrieves (II.2) from Corollary III.2 with $p = 2$.

B. Co-moments

In this section we derive the update formulas for the covariance, which is used within both our multi-correlative and PCA statistics algorithms. Update formulas for, e.g., the coskewness and cokurtosis (cf. [7] for an example use in the context of mathematical finance) can be derived by combining the approach we use below for the variance with the higher-order summations used in III-A. Much of the notation remains unchanged in this section; however, now \mathcal{S} denotes a set of doubles $x = (u, v)$ and $\mu_{u,1}, \mu_{v,1}, \mu_{u,2}, \mu_{v,2}$ denote the means of u and v on \mathcal{S}_1 and \mathcal{S}_2 , respectively. We also define $\delta_{u,2,1} = \mu_{u,2} - \mu_{u,1}$, and $\delta_{v,2,1} = \mu_{v,2} - \mu_{v,1}$.

Proposition III.3. *The pairwise update formula for $C_{2,\mathcal{S}} = \sum_{(u,v) \in \mathcal{S}} (u - \mu_u)(v - \mu_v)$ is:*

$$C_{2,\mathcal{S}} = C_{2,\mathcal{S}_1} + C_{2,\mathcal{S}_2} + \frac{n_1 n_2}{n} \delta_{u,2,1} \delta_{v,2,1}. \quad (\text{III.6})$$

Proof: By definition of $C_{2,\mathcal{S}}$, and because $\{\mathcal{S}_1, \mathcal{S}_2\}$ is a partition of \mathcal{S} , one has

$$\begin{aligned} C_{2,\mathcal{S}} &= \sum_{(u,v) \in \mathcal{S}} (u - \mu_u)(v - \mu_v) \\ &= \sum_{(u,v) \in \mathcal{S}_1} (u - \mu_u)(v - \mu_v) + \sum_{(u,v) \in \mathcal{S}_2} (u - \mu_u)(v - \mu_v). \end{aligned} \quad (\text{III.7})$$

We expand the means over \mathcal{S} into expressions that relate them to the means on \mathcal{S}_1 and \mathcal{S}_2 :

$$\begin{aligned} &\sum_{(u,v) \in \mathcal{S}_1} (u - \mu_u)(v - \mu_v) \\ &= \sum_{(u,v) \in \mathcal{S}_1} \left(u - \frac{n_1 \mu_{u,1} + n_2 \mu_{u,2}}{n} \right) \left(v - \frac{n_1 \mu_{v,1} + n_2 \mu_{v,2}}{n} \right) \\ &= \sum_{(u,v) \in \mathcal{S}_1} \left(u - \mu_{u,1} - \frac{n_2}{n} \delta_{u,2,1} \right) \left(v - \mu_{v,1} - \frac{n_2}{n} \delta_{v,2,1} \right) \\ &= \sum_{(u,v) \in \mathcal{S}_1} \left[\begin{aligned} &(u - \mu_{u,1})(v - \mu_{v,1}) - \frac{n_2}{n} \delta_{u,2,1} (u - \mu_{u,1}) \\ &- \frac{n_2}{n} \delta_{v,2,1} (v - \mu_{v,1}) + \frac{n_2^2}{n^2} \delta_{u,2,1} \delta_{v,2,1} \end{aligned} \right]. \end{aligned} \quad (\text{III.8})$$

Again using the commutativity of summations over finite sets and noticing that, by definition of $\mu_{u,1}$ and $\mu_{v,1}$,

$$\sum_{(u,v) \in \mathcal{S}_1} \frac{n_2}{n} \delta_{u,2,1} (u - \mu_{u,1}) = \frac{n_2}{n} \delta_{u,2,1} \sum_{u \in \mathcal{S}_1} (u - \mu_{u,1}) = 0$$

and

$$\sum_{(u,v) \in \mathcal{S}_1} \frac{n_2}{n} \delta_{v,2,1} (v - \mu_{v,1}) = \frac{n_2}{n} \delta_{v,2,1} \sum_{v \in \mathcal{S}_1} (v - \mu_{v,1}) = 0,$$

we simplify (III.8) into

$$\sum_{(u,v) \in \mathcal{S}_1} (u - \mu_u)(v - \mu_v) = C_{2,\mathcal{S}_1} + \frac{n_1 n_2^2}{n^2} \delta_{u,2,1} \delta_{v,2,1}.$$

Similarly, by interchanging the roles of \mathcal{S}_1 and \mathcal{S}_2 , we obtain

$$\sum_{(u,v) \in \mathcal{S}_2} (u - \mu_u)(v - \mu_v) = C_{2,\mathcal{S}_2} + \frac{n_2 n_1^2}{n^2} \delta_{u,2,1} \delta_{v,2,1}$$

because $(\mu_{u,2} - \mu_{u,1})(\mu_{v,2} - \mu_{v,1}) = (\mu_{u,1} - \mu_{u,2})(\mu_{v,1} - \mu_{v,2})$. Therefore, (III.7) becomes

$$C_{2,\mathcal{S}} = C_{2,\mathcal{S}_1} + C_{2,\mathcal{S}_2} + \frac{n_1 n_2^2 + n_2 n_1^2}{n^2} \delta_{u,2,1} \delta_{v,2,1},$$

from whence the result arises since $n_1 + n_2 = n$. ■

Pairwise update formulas for the estimators of the variance are now obtained immediately; for instance, the unbiased estimator of the covariance of \mathcal{S} is $\frac{1}{n-1} C_{2,\mathcal{S}}$.

Remark III.2. In passing, we note that in the case where \mathcal{S}_2 is reduced to a singleton $\{(s, t)\}$, Proposition III.3 reduces to the following incremental update formula for $\mathcal{S} = \mathcal{S}_1 \cup \{(s, t)\}$:

$$C_{2,\mathcal{S}} = C_{2,\mathcal{S}_1} + \frac{n-1}{n} (s - \mu_{s,1})(t - \mu_{t,1}). \quad (\text{III.9})$$

IV. PARALLEL STATISTICAL FRAMEWORK

Using the single-pass update formulas presented in Section III we have built an open source parallel statistical framework that computes descriptive, correlative, multi-correlative, and PCA statistics. It has been our experience that not only these, but many other types of statistical analysis, can be decomposed into three distinct phases which form an interesting design pattern [8]. Specifically, the first 2 phases are essentially a special case of the map-reduce pattern [9] where the keys need not be communicated since they are identical across all processes and may be ordered uniquely. The final phase is effectively a second map operation. These phases, which can (but need not) be performed in a strict sequence on a single set of observations, can be described as:

a) *Learn:* Calculate a “raw” statistical model from an input data set. By “raw”, we mean the minimal representation of the desired model, that contains only primary statistics. For example, in the case of descriptive statistics: sample size, minimum, maximum, mean, and centered M_2 , M_3 and M_4 aggregates as defined in Section II and Section III. This stage involves a reduction operation across processes participating in the run.

b) *Derive:* Calculate a “full” statistical model from a raw model. By “full”, we mean the complete representation of the desired model, that contains both primary and derived statistics. For example, in the case of descriptive statistics, the following derived statistics are calculated from the raw model: unbiased variance estimator, standard deviation, and two estimators (g and G) for both skewness and kurtosis. For

the analyses we have implemented, this phase is embarrassingly parallel, not involving any inter-process communication. Even so, it can be viewed as a continuation of the reduction operation in a map-reduce scheme since the derived output is a reduced model of the initial data set.

c) *Assess*: Given a statistical model – from the same or another data set – mark each datum of a given data set. For example, in the case of descriptive statistics, each datum is marked with (or mapped to, in the map-reduce parlance) its relative deviation with respect to the model mean and standard deviation (this amounts to the one-dimensional Mahalanobis distance). This phase also is embarrassingly parallel, not involving any inter-process communication.

The purpose of decomposing our algorithms into three phases is twofold: parallel efficiency and replication of the typical statistical analysis process. In our approach, inter-process communication and updates are performed only for primary statistics. The parallel update formulas of Section III are utilized in the Learn phase only, for the other phases only involve direct (i.e., with no updates needed) derivations of the secondary statistics from the primary ones. The calculations to obtain derived statistics from primary statistics are typically fast and simple and need only be calculated once, without communication, upon completion of all parallel updates of primary variables. In other words, the (parallel) reduction operation required at the end of the Learn phase will be negligible as compared to the serial processing, as only minuscule amounts of data need be exchanged between processes. Data to be assessed is assumed to be distributed in parallel across all processes participating in the computation, thus no communication is required as each process assesses its own resident data. The phase descriptions of each of the algorithms in our statistical framework are described in Table I.

V. RESULTS

The parallel statistical framework currently consists of four classes

- `vtkDescriptiveStatistics`
- `vtkCorrelativeStatistics`
- `vtkMultiCorrelativeStatistics`
- `vtkPCAStatistics`

and of their four respective subclasses that overwrite the corresponding Learn methods:

- `vtkPDescriptiveStatistics`
- `vtkPCorrelativeStatistics`
- `vtkPMultiCorrelativeStatistics`
- `vtkPPCAStatistics`.

that are all available for download at [10] as part of VTK.

In Sections V-A and V-B we present results which demonstrate the correctness and scalability of these algorithms respectively using a series of synthetic data sets. In Section V-C we describe the use of the framework to perform PCA on data generated from a turbulent, reacting flow simulation.

A. Algorithm Correctness

The parallel runs described in this Section and Section V-B were executed on Sandia National Laboratories’ `catalyst` computational cluster, which comprises 120 dual 3.06GHz Pentium Xeon compute nodes with 2GB of memory each. This cluster has a Gigabit Ethernet user network for job launch, I/O to storage, and user interaction with jobs, and a 4X Infiniband fabric high-speed network using a Voltaire 9288 InfiniBand switch. Its operating system has a Linux 2.6.17.11 kernel, and its batch scheduling system is the TORQUE resource manager [11].

A series of (pseudo-) randomly generated samples is used in order to assess algorithm correctness. We inspect the numerical results obtained by both the multi-correlative and PCA statistics algorithms. More precisely, we examine the statistical models obtained when both Learn and Derive options are turned on. Relatively large input sets are used ($n = 10^6$), in order to mitigate the risk of statistical bias due to insufficient sampling. In addition, the test case is run 100 times for each random variable, and we examine the statistical dispersion of the results of the ensemble of these runs. Since the statistical properties of the test cases are known, we can immediately compare them to the calculated results.

To validate the multi-correlative algorithm, we generated 8 separate samples of independent pseudo-random variables, the first 4 variables having a standard normal distribution and the last 4 variables having a standard uniform distribution. We compared the results obtained with the Learn and Derive option of the multi-correlative statistical engine to the theoretical values of the random variables which serve as models for the pseudo-random inputs, namely, $\mathcal{N}(0,1)$ and $\mathcal{U}(0,1)$. This comparison was done by simple visual inspection of the numerical results, by:

- 1) comparing the sample mean of the quantity of interest (e.g., mean) across the number n_r of runs to the corresponding theoretical quantity (e.g, expectation), and
- 2) examining the variability of the results by checking the standard deviation of the quantity of interest across the n_r runs.

TABLE II
MEANS OF 8 PSEUDO-RANDOM INDEPENDENT SAMPLES OF 4 STANDARD NORMAL AND 4 STANDARD UNIFORM DISTRIBUTIONS, AVERAGED ACROSS 100 RUNS, *versus* THEORETICAL VALUES. THE LAST COLUMN INDICATES THE STANDARD DEVIATION OF THE MEANS ACROSS THE 100 RUNS.

Sample	Expectation (theoretical mean)	Average of sample means	Stand. deviation of sample means
$\mathcal{N}(0,1)_0$	0	0.0000328	0.000194
$\mathcal{N}(0,1)_1$	0	0.0000181	0.000159
$\mathcal{N}(0,1)_2$	0	-0.0000278	0.000173
$\mathcal{N}(0,1)_3$	0	0.0000278	0.000178
$\mathcal{U}(0,1)_4$	0.5	0.4999987	$5.448978 \cdot 10^{-5}$
$\mathcal{U}(0,1)_5$	0.5	0.4999961	$6.079136 \cdot 10^{-5}$
$\mathcal{U}(0,1)_6$	0.5	0.5000044	$5.586889 \cdot 10^{-5}$
$\mathcal{U}(0,1)_7$	0.5	0.5000030	$5.306256 \cdot 10^{-5}$

TABLE I
PHASE DESCRIPTIONS OF THE STATISTICAL ALGORITHMS IN OUR DISTRIBUTED FRAMEWORK.

	Learn	Derive	Assess
Descriptive	Calculate minimum, maximum, mean, and centered M_2 , M_3 and M_4 aggregates	Calculate unbiased variance estimator, standard deviation, skewness (1_2 and G_1 estimators), kurtosis (g_2 and G_2 estimators)	Mark with relative deviation (one-dimensional Mahalanobis distance)
Correlative	Calculate minima, maxima, means, and centered M_2 aggregates	Calculate unbiased variance and covariance estimators, Pearson correlation coefficient, and linear regressions (both ways)	Mark with squared two-dimensional Mahalanobis distance
Multi-Correlative	Calculate means and pairwise centered M_2 aggregates	Calculate the upper triangular portion of the symmetric $n_i \times n_i$ covariance matrix and its (lower) Cholesky decomposition	Mark with squared multi-dimensional Mahalanobis distance
PCA	Identical to the multi-correlative algorithm	Everything the multi-correlative algorithm provides, plus the n_i eigenvalues and eigenvectors of the covariance matrix	Perform a change of basis to the principal components (eigenvectors), optionally projecting to the first m_i components (either user-specified or determined by the fraction of maximal eigenvalues whose sum is above a user-specified threshold)

Using this methodology with $n_r = 100$ runs over 32 processes, the results provided in Table II show that pseudo-random input samples possess the desired first-order statistics, with negligible variation across the 100 runs.

TABLE III
CHOLESKY DECOMPOSITION OF COVARIANCE MATRIX USING THE MULTI-CORRELATIVE STATISTICAL ENGINE FOR 4 INDEPENDENT STANDARD NORMAL (TOP) AND UNIFORM (BOTTOM) SAMPLES, AVERAGED ACROSS 100 RUNS. THEORETICAL VALUES ARE THOSE OF THE IDENTITY MATRIX \mathbb{I}_4 .

Standard Normal			
1.000005			
0.000009	0.999989		
0.000015	-0.000007	1.000006	
-0.000005	0.000028	0.000001	1.000013
Standard Uniform			
0.288670			
0.000003	0.288675		
-0.000008	-0.000003	0.288678	
-0.000003	-0.000011	-0.000005	0.288674

Table III shows the Cholesky decompositions of the covariance matrices that were calculated by the multi-correlative statistics algorithm with, respectively, the standard uniform and standard normal pseudo-random inputs. We observe that the numerical results are in statistical agreement with their theoretical counterparts, up to at least 5 significant digits, in a manner consistent with the fluctuations of the pseudo-random inputs. In particular, and as expected due to the mutual independence of the input samples, only diagonal terms are non-zero (up to the aforementioned precision) in the Cholesky decompositions of the covariance matrices.

To verify that the PCA decomposition itself is correct, we

are using a slightly modified input, comprising 4 pseudo-random samples of the following distributions:

- 1) 2 independent standard normal variables, denoted $X_0 = \mathcal{N}(0, 1)_0$ and $X_1 = \mathcal{N}(0, 1)_1$;
- 2) 2 variables obtained as the following linear combinations of the above: $X_2 = X_0 + X_1$, and $X_3 = 2X_0 - 3X_1$.

The statistical correlations of the input variables should be reflected in the output of the principal component analysis. In fact, it is trivial, based on elementary properties of Gaussian random variables, to demonstrate that the random vector $X = (X_1, X_2, X_3, X_4)$ is Gaussian, centered, with covariance matrix:

$$\text{cov}(X) = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & -3 \\ 1 & 1 & 2 & -1 \\ 2 & -3 & -1 & 13 \end{pmatrix}$$

for, if X_i and X_j are two random variables with respective distributions $\mathcal{N}(\mu_i, \sigma_i^2)$ and $\mathcal{N}(\mu_j, \sigma_j^2)$, then, for any arbitrary pair of reals (a, b) , the random variable $aX_i + bX_j$ is Gaussian with distribution $\mathcal{N}(a\mu_i + b\mu_j, a^2\sigma_i^2 + b^2\sigma_j^2)$. The off-diagonal terms of the covariance matrix are obtained by using the linearity of the expectation:

$$\mathbb{E}(X_i \times (aX_j + bX_k)) = a\mathbb{E}(X_i X_j) + b\mathbb{E}(X_i X_k).$$

As predicated by the definition of the components of X , its covariance matrix is singular, with rank 2. Accordingly, it has only two non-zero eigenvalues, which can be approximated as $\lambda_1 \approx 14.1$ and $\lambda_2 \approx 2.91$. We are not discussing the eigenvectors here, as evincing them once the eigenvalues are known is trivial, and left to the reader as an exercise.

The two zero eigenvalues are indeed retrieved with our PCA statistics algorithm, and the values for the two non-zero ones, averaged over $n_r = 100$ runs are shown in Table IV. We observe again agreement with the theoretical values, up

TABLE IV

MEANS OF THE NON-ZERO EIGENVALUES OF 4 PSEUDO-RANDOM INDEPENDENT SAMPLES OF 2 STANDARD NORMAL DISTRIBUTIONS, AND 2 LINEAR CORRELATIONS THEREOF, AVERAGED ACROSS 100 RUNS, *versus* APPROXIMATED THEORETICAL VALUES. THE LAST COLUMN INDICATES THE STANDARD DEVIATION OF THE MEANS ACROSS THE 100 RUNS.

Eigenvalue (approx. theoretical)	Average of eigenvalues	Standard deviation of eigenvalues
14.0902	14.0906	0.011465
2.90983	2.90958	0.0023212

to the fifth significant digit. This is a better agreement than what we expected, considering the fact that the successive truncation errors of the covariance, Cholesky, and eigenvalue calculations are compounded with the fact that the pseudo-random samples do not exhibit perfect statistical properties, as illustrated by their means in Table II and covariances in Table III. Jitter, as measured by standard deviation across all 100 runs, is essentially insignificant.

B. Algorithm Scalability

In order to assess speed-up independently of the load-balancing scheme, a series of (pseudo-) randomly-generated samples is used. Similar to Section V-A, input tables are created at run time by generating 8 separate samples of independent pseudo-random variables the first 4 variables respectively having a standard normal distribution and the last 4 variables respectively having a standard uniform distribution. Since our objective is to assess the scalability of the parallel statistics engines only, equally-sized slabs of data are created by each process in order to work with perfectly load-balanced cases. For the same reason, the amount of time needed to create the input data table is excluded from the analysis. Each algorithm is executed with Learn, Derive, and Asses modes on.

With these synthetic examples, we assess:

- 1) relative speed-up (at constant total work), and
- 2) scalability of the rate of computation (at constant work per process).

TABLE V

RELATIVE SPEED-UP (AT CONSTANT TOTAL WORK), WITH A TOTAL SAMPLE SIZE OF $N = 25,600,000$. THE UNITS FOR THE D (DESCRIPTIVE), C (CORRELATIVE), MC (MULTI-CORRELATIVE), AND PCA COLUMNS ARE SECONDS / $S_N(p)$.

N/p	p	D	C	MC	PCA
25,600,000	1	79.7 / 1.00	60.4 / 1.00	53.6 / 1.00	65.7 / 1.00
12,800,000	2	39.6 / 2.01	30.3 / 1.99	26.6 / 2.01	33.0 / 1.99
6,400,000	4	20.4 / 3.90	15.6 / 3.86	13.4 / 4.00	16.8 / 3.91
3,200,000	8	11.1 / 7.15	8.0 / 7.57	7.2 / 7.40	8.8 / 7.50
1,600,000	16	5.3 / 14.97	4.0 / 14.95	3.6 / 14.94	4.5 / 14.60
800,000	32	2.8 / 28.65	2.1 / 28.75	1.9 / 28.52	2.2 / 29.72
400,000	64	1.4 / 56.90	1.1 / 53.43	0.9 / 56.44	1.1 / 57.12

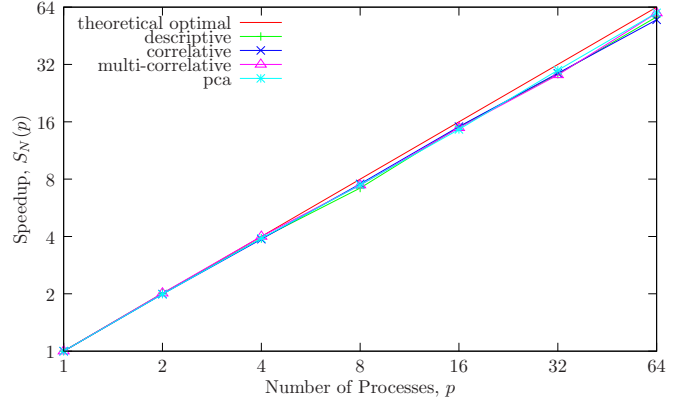


Fig. 1. Relative speed-up at constant total work with a total data size of $N = 102,400,000$.

1) *Relative Speed-Up*: Given a problem of size N (as measured in our case by sample size), the wall clock time measured to complete the work with p processes is denoted $T_N(p)$. Then, relative speed-up with p processes is

$$S_N(p) = \frac{T_N(1)}{T_N(p)}.$$

We achieve near optimal (linear) speedup with p processes when $S_N(p) = p$ and, therefore, relative speed-up results for $S_N(p)$ may be visually inspected by plotting $S_N(p)$ *versus* the number of processes: optimal speed-up is revealed by a line, the angle bisector of the first quadrant. The values of p were chosen to be increasing powers of 2, for convenience only, and making use of other values did not modify speed-up results. The results obtained on *catalyst* are provided in Tables V and plotted in Figure 1.

As expected based on the embarrassingly parallel nature of the algorithms, the measured relative speed-up is optimal (within $\pm 10\%$ fluctuations attributable to OS jitter and such), until total wall time measurements become too small to remain accurate (less than 1 sec.), and the decreasing amount of work per process ultimately results in a situation where overheads, even small in absolute terms, become dominant as compared to the amount of actual computational work. In this current example, it appears that with 32 processes, minimal reliably measurable wall clock time has been or is almost reached. Note that this corresponds to a per process load of $N/p = 800,000$ points per process.

2) *Rate of Computation Scalability*: The rate of computation is defined as

$$r(p) = \frac{N(p)}{T_{N(p)}(p)},$$

where $N(p)$, the sample size, now varies with the number of processes p . We then measure its scalability by normalizing it with respect to the rate of computation obtained with a single process, as follows:

$$R(p) = \frac{r(p)}{r(1)} = \frac{N(p)T_{N(1)}(1)}{N(1)T_{N(p)}(p)},$$

TABLE VI

RATE OF COMPUTATION SCALABILITY (AT CONSTANT LOAD PER PROCESS). THE UNITS FOR THE D (DESCRIPTIVE), C (CORRELATIVE), MC (MULTI-CORRELATIVE) AND PCA COLUMNS ARE SECONDS / $R(p)$.

$N(p)$	p	D	C	MC	PCA
8,000,000	1	12 / 1.00	9 / 1.00	17 / 1.00	21 / 1.00
16,000,000	2	12 / 2.00	9 / 2.01	17 / 2.02	21 / 2.02
32,000,000	4	12 / 4.00	9 / 4.11	17 / 3.89	22 / 3.78
64,000,000	8	12 / 8.00	9 / 8.09	18 / 7.48	21 / 7.95
128,000,000	16	13 / 14.8	10 / 13.7	18 / 14.76	21 / 15.86
256,000,000	32	13 / 29.5	9 / 31.0	20 / 27.05	24 / 28.36
512,000,000	64	13 / 59.1	9 / 61.2	20 / 53.21	24 / 54.94
1,024,000,000	128	13 / 118	10 / 117	19 / 112.60	24 / 111.77

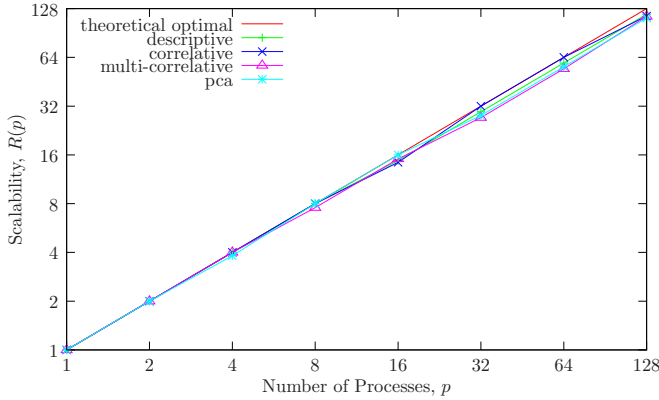


Fig. 2. Rate of computation scalability at constant work per process of $N(p)/p = 4,000,000$.

In particular, if the sample size is made to vary in proportion to the number of processes, i.e., if $N(p) = pN(1)$, then

$$R(p) = \frac{pT_{N(1)}(1)}{T_{pN(1)}(p)} = \frac{pT_{N(1)}(1)}{pT_{N(1)}(p)} = \frac{T_{N(1)}(1)}{T_{N(1)}(p)},$$

and thus, optimal (linear) scalability is also attained with p processes when $R(p) = p$. Note that without linear dependency between N and p , the latter equality no longer implies optimal scalability. Hence, under the above assumptions, scalability can also be visually inspected, with a plot of $R(p)$ versus the number of processes, where optimal scalability is also indicated by the angle bisector of the first quadrant.

In order to assess rate of computation scalability (at constant work per process), a series of increasingly large samples was generated. Each sample consisted of 8 variables, each with np observations, with $p \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ denoting the number of processes and $n = 10^6$ denoting the number of sample points per process. Note that the scheduler is left to decide whether one or two cores per node are occupied by the p processes. Forcing all cluster nodes to utilize either exactly one, or exactly two of their cores did not result in a measurable difference. Corresponding wall clock times measured on *catalyst* are given in Table VI and plotted in Figure 2. These clearly exhibit optimal scalability (again within $\pm 10\%$ fluctuations attributable to OS jitter and such),

thus experimentally verifying the embarrassingly parallel nature of these algorithms.

C. Parallel PCA of Turbulent Reacting Flow Data

In addition to the synthetic results which demonstrate correctness and scalability, our framework has been used to perform PCA on the output of S3D, a parallel application for simulating turbulent, reacting flows. S3D solves the differential equations governing the evolution of momentum and composition through a fully compressible solution to the Navier-Stokes equations. The solution is advanced on a finite difference grid using an 8th order approximation to the spatial derivatives. In the lifted jet flame simulation [12], evolution equations are also advanced for each of 21 species involved in a reduced chemical mechanism for ethylene-air combustion. An image from this simulation is shown in Figure 3.

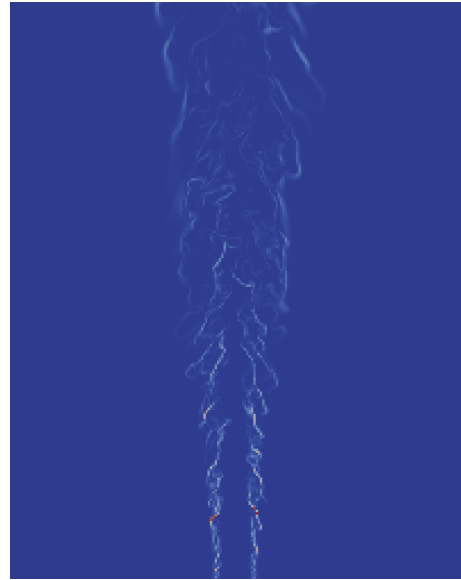


Fig. 3. A visualization of the lifted jet flame.

In this configuration, the conditions following mixing between the fuel and oxidizer are suitable for autoignition within the domain residence time, and autoignition is a significant factor in the stabilization of the flame. The solution is statistically stationary in time; although fluctuations about the mean quantities are unsteady, the time-averaged solution is steady. There is a spatial development in the solution from separate, non-burning, fuel and oxidizer at the domain inlet (bottom) to a steady flame near the outlet at the top of the domain. As this development occurs, different underlying physics control the composition. Before any reaction has taken place, the composition will be completely determined by the amount of fuel and oxidizer present — any of the three species (O_2, N_2, C_2H_4) is sufficient to determine the complete composition. After a steady flame has developed, it is known that the composition is largely determined by amount of the mixture which *originated* in the fuel stream - an indication of this can be obtained by a linear combination of all of the species present which contain

the elements found in the inlet streams. During the transition, the species which most naturally describe the thermodynamic state may have significance indicating the dominant physical processes.

In an effort to gain further insight into these physical processes, a principal component analysis of the simulation data was performed using our parallel statistics framework (specifically, the `vtkPPCAStatistics` algorithm). The full grid was 2025x1600x400 points in the x, y, and z direction with 21 species variables at each grid point. The analysis was carried out using 1,500 processes on the Cray XT4 at Oak Ridge National Labs, and thus each process was responsible for a subgrid of size 135x80x80 in the x, y, and z direction. It took 4.27 seconds to run the parallel PCA engine with the Learn and Derive options turned on, for the 1.296 billion data points with 21 variables each.

VI. CONCLUSION

In this work we derived general update formulas that allow for pairwise and incremental updates of both arbitrary-order centered statistical moments and covariance. These can be readily extended to arbitrary-order co-moments, such as cokurtosis and coskewness, which we have not derived here. Using these single-pass formulas, we have built an open source statistical framework for large-scale, distributed data sets available on line at [10]. The implementation of our framework proves to be numerically stable and achieves near-optimal linear scalability and speed-up properties. Moreover, our framework has been successfully used to gain insight into large-scale turbulent, reacting flow simulation data. The algorithms in the framework form a nice design pattern that is readily extensible to a variety of statistical algorithms. Future work includes implementation of additional statistical algorithms including those that utilize higher-order moments as well as the Learn-Derive-Assess pattern.

ACKNOWLEDGMENTS

The authors would like to thank Brian Wylie, for his comments on the integration of scalable statistical tools in `VTK/Titan`, Jackson Mayo for his precursor work on a serial version of the multi-correlative statistics algorithm, and Jacqueline Chen and Chun Sang Yoo for access to the S3D simulation code and data.

REFERENCES

- [1] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," in *NIPS*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2006, pp. 281–288. [Online]. Available: <http://dblp.uni-trier.de/rec/bibtex/conf/nips/ChuKLYBNO06>
- [2] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.
- [3] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances." Stanford University, Department of Computer Science, Technical Report STAN-CS-79-773, 1979.
- [4] T. B. Terriberry, "Computing higher-order moments online," 2008, <http://people.xiph.org/~terribe/notes/homs.html>.
- [5] P. Prakasam and M. Madheswaran, "M-ary shift keying modulation scheme identification algorithm using wavelet transform and higher order statistical moment," *Journal of Applied Science*, vol. 8, no. 1, pp. 112–119, 2008, [10.1155/2008/175236](https://doi.org/10.1155/2008/175236).
- [6] N. Kikuchi, S. Hayase, K. Sekine, and S. Sasaki, "Performance of chromatic dispersion monitoring using statistical moments of asynchronously sampled waveform histograms," *Photonics Technology Letters*, vol. 17, pp. 1103–1105, May 2005, [http://dx.doi.org/10.1109/LPT.2005.846752](https://doi.org/10.1109/LPT.2005.846752).
- [7] R. Christie-Davida and M. Chaudhryb, "Coskewness and cokurtosis in futures markets," *Journal of Empirical Finance*, vol. 8, no. 1, pp. 55–81, 2001, [10.1016/S0927-5398\(01\)00020-2](https://doi.org/10.1016/S0927-5398(01)00020-2).
- [8] R. J. Erich Gamma, Richard Helm and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
- [9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, Dec. 2004.
- [10] "VTK Doxygen documentation," <http://www.vtk.org/doc/nightly/html>.
- [11] "TORQUE Resource Manager," <http://www.clusterresources.com/pages/products/torque-resource-manager.php>.
- [12] C. S. Yoo, E. Richardson, R. Sankaran, and J. H. Chen, "Dns of a turbulent lifted ethylene/air jet flame in an auto-ignitive coflow - stabilization and flame structure," *Turbulent Non-premixed Flames Workshop (TNF'08)*, 2008.