SANDIA REPORT

SAND2015-1152
Unlimited Release
Printed February 2015

A Divergence Statistics Extension to VTK for Quantitative Performance Analysis

Philippe Pébay, Janine Bennett

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-1152
Unlimited Release
Printed February 2015

A Divergence Statistics Extension to VTK for Quantitative Performance Analysis

Philippe Pébay, Janine Bennett
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551, U.S.A.
pppebay,jcbenne@sandia.gov

Abstract

This report follows the series of previous documents ([PT08, BPRT09b, PT09, BPT09, PT10, PB13], where we presented the parallel descriptive, correlative, multi-correlative, principal component analysis, contingency, k-means, order and auto-correlative statistics engines which we developed within the Visualization Tool Kit (VTK) as a scalable, parallel and versatile statistics package. We now report on a new engine which we developed for the calculation of divergence statistics, a concept which we hereafter explain and whose main goal is to quantify the discrepancy, in a stasticial manner akin to measuring a distance, between an observed empirical distribution and a theoretical, "ideal" one. The ease of use of the new diverence statistics engine is illustrated by the means of C++ code snippets. Although this new engine does not yet have a parallel implementation, it has already been applied to HPC performance analysis, of which we provide an example.

Acknowledgments

The authors would like to thank who Jeremy Wilke (Sandia National Laboratories) who encouraged this work in the context of performance analysis for extreme-scale computing.

Contents

This page intentionally left blank

1 Introduction

This report is a sequel to [PT08, BPRT09b, PT09, BPT09, PT10, PB13], which respectively focused on the parallel descriptive, correlative, multi-correlative, principal component analysis, contingency, k-means, order, and auto-correlative engines which we designed and implemented as VTK parallel filters; please refer to these references for a detailed presentation of these engines as well as an assessment of their scalability and speed-up properties.

1.1 Initial Motivation: The Titan Informatics Toolkit

The addition of a parallel, scalable statistics module to VTK was motivated by the Titan Informatics Toolkit [WBS08], a collaborative effort between Sandia National Laboratories and Kitware. This effort significantly expanded the Visualization ToolKit (VTK) to support the ingestion, processing, and display of informatics data. By leveraging the VTK data and execution models, Titan provides a flexible, component based, pipeline architecture for the integration and deployment of algorithms in the fields of intelligence, semantic graph and information analysis. A theoretical application

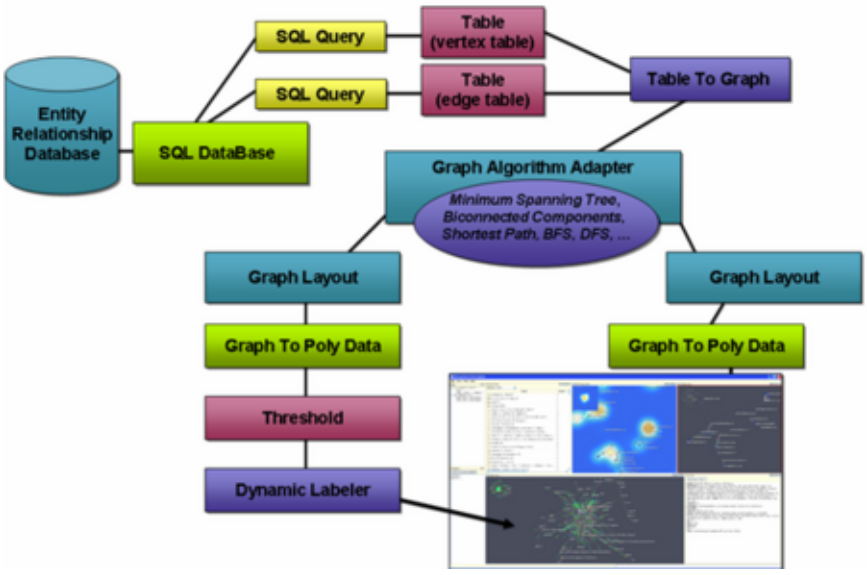

Figure 1. A theoretical application built with Titan.

built from Titan/VTK components is schematized in Figure 1. The flexibility of the pipeline architecture of VTK allows for effective utilization of the Titan components for different problem domains. For instance, an early implementation was OVIS, a generalization of the ParaView scientific visualization application dedicated to information visualization, leveraging the ParaView client-server architecture to perform scalable analysis on distributed memory platforms.

In 2008, the parallel statistical engines were integrated into VTK, and the module has continued

to grow as new engines have been developed in the context of the "Network Grand Challenge" LDRD project at Sandia, and under a 3-year grant from the DOE/ASCR program to conduct broad research in the topological and statistical analysis of petascale data. Now, the main thrust for the current work is for the sake of large-scale high perfomance architecture simulation, where it desired to quantify how a number of variables of interest behave under various simulation inputs, as compared to a model "ideal" behavior.

This report thus presents the statistical engine which we developed in this context, using statistical divergences and distances.

1.2 Statistics Functionality in VTK

A number of univariate, bivariate, and multivariate statistical tools have been implemented in VTK. Each tool acts upon data stored in one or more tables; the first table serves as observations and further tables serve as model data. Each row of the first table is an observation, while the form of further tables depends on the type of statistical analysis. Each column of the first table is a variable.

Operations

In order to meet the two overlapping but not exactly congruent design requirements of matching typical data analysis workflows and being conducive to scalable parallel implementation, our design partitions the statistical analysis algorithms into 4 disjoint operations: learn a model from observations; derive statistics from a model; assess observations with a model, and test a hypothesis. These operations, when all are executed, occur in order as shown in Figure 2. However, it is

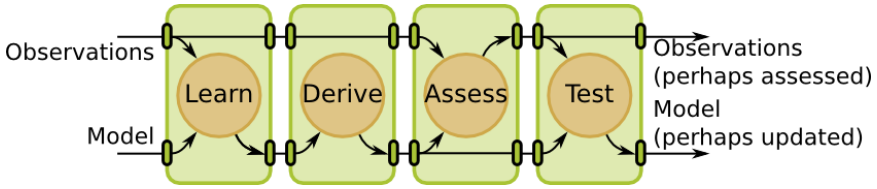

Figure 2. The 4 operations of statistical analysis and their interactions with input observations and models. When an operation is not requested, it is eliminated by connecting input to output ports.

also possible to execute only a subset of these, for example when it is desired that previously computed models, or models constructed with expert knowledge, be used in conjunction with existing data. Note that in earlier publications (e.g., [BPRT09a, PTB10, Inc10]) only the first 3 operations were mentioned; the Test operation, which we initially saw as a part of Derive, was separated out for reasons we explained in [PTBM11]. These operations, performed on a request comprising a set of columns of the input observations table, are further explained as follows:

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 (*cf.* [PÓ8]). For Table 1 with a request $R_1 = \{B\}$, these values are 6, 1, 11, $4.8\bar{3}$, $68.8\bar{3}$, $159.\bar{4}$, and $1759.819\bar{4}$, respectively.

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 Table 1 with a request $R_1 = \{B\}$, these additional values are $13.7\bar{6}$, 3.7103, 0.520253, 0.936456, -1.4524, and -1.73616 respectively.

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 its relative deviation with respect to the model mean and standard deviation (this amounts to the one-dimensional Mahalanobis distance). Table 1 shows this distance for $R_1 = \{B\}$ in column E.

Variables

A univariate statistics algorithm only uses information from a single column and, similarly, a bivariate from 2 columns. Because an input table may have many more columns than an algorithm can make use of, the API must provide a way for users to denote columns of interest. Because it may be more efficient to perform multiple analyses of the same type on different sets of columns at once as opposed to one after another, the VTK statistical engines provide a way for users to make multiple analysis requests of a single filter.

Table 1. A table of observations that might serve as input to a statistics algorithm.

row	A	B	C	D	E
1	0	1	0	1	1.03315
2	1	2	2	2	0.76363
3	0	3	4	6	0.49411
4	1	5	6	24	0.04492
5	0	7	8	120	0.58395
6	1	11	10	720	1.66202

As an example, consider Table 1. It has 6 observations of 5 variables. If univariate statistics of A, B, and C are desired then three univariate requests must be made, one for each column. On the other hand, if a multi-variate statistical analysis, such as PCA, is desired $\{A,B,C\}$ then a single request is sufficient.

Algorithms

At the time of writing, the following algorithms are available in VTK proper or as extensions to it (as is the case for the divergence statistics engine discussed in this report):

1. Univariate statistics:
   (a) Descriptive statistics:
      Learn: calculate minimum, maximum, mean, and centered M_2, M_3 and M_4 aggregates;
      Derive: calculate unbiased variance estimator, standard deviation, skewness (1_2 and G_1 estimators), kurtosis (g_2 and G_2 estimators);
      Assess: mark with relative deviations (one-dimensional Mahalanobis distance).
   (b) Order statistics:
      Learn: calculate histogram;
      Derive: calculate arbitrary quantiles, such as "5-point" statistics (quantiles) for box plots, deciles, percentiles, etc.;
      Assess: mark with quantile index.
2. Bivariate statistics:
   (a) Correlative statistics:
      Learn: calculate minima, maxima, means, and centered M_2 aggregates;
      Derive: calculate unbiased variance and covariance estimators, Pearson correlation coefficient, and linear regressions (both ways);
      Assess: mark with squared two-dimensional Mahalanobis distance.
   (b) Contingency statistics:
      Learn: calculate contingency table;
      Derive: calculate joint, conditional, and marginal probabilities, as well as information entropies;
      Assess: mark with joint and conditional PDF values, as well as pointwise mutual informations.
3. Multivariate statistics: These filters all accept requests containing n_i variables upon which simultaneous statistics should be computed.
   (a) Multi-Correlative statistics:
      Learn: calculate means and pairwise centered M_2 aggregates;
      Derive: calculate the upper triangular portion of the symmetric $n_i \times n_i$ covariance matrix and its (lower) Cholesky decomposition;
      Assess: mark with squared multi-dimensional Mahalanobis distance.

(b) PCA statistics:

Learn: identical to the multi-correlative filter;

Derive: everything the multi-correlative filter provides, plus the n_i eigenvalues and eigenvectors of the covariance matrix;

Assess: perform a change of basis to the principal components (eigenvectors), optionally projecting to the first m_i components, where $m_i \leq n_i$ is either some user-specified value or is determined by the fraction of maximal eigenvalues whose sum is above a user-specified threshold. This results in m_i additional columns of data for each request R_i.

(c) k-means statistics:

Learn: compute optimized set(s) of cluster centers from initial set(s) of cluster centers. In the default case, the initial set comprises the first k observations. However, the user can specify one or more sets of cluster centers (with possibly differing numbers of clusters in each set) via an optional input table, in which case an optimized set of cluster centers is computed for each of the input sets.

Derive: calculate the global and local rankings amongst the sets of clusters computed in the learn operation. The global ranking is the determined by the error amongst all new cluster centers, while the local rankings are computed amongst clusters sets with the same number of clusters. The total error is also reported;

Assess: mark wth closest cluster id and associated distance for each set of cluster centers.

4. Time-series statistics: In this case, requests contain a fixed number m of values per variable, measured across n time-steps; conceptually, the input is thus represented for each variable by a $n \times m$ block of data, where each $1 \times m$ row is often referred to as a *time-slab*. In other words, there are as many data points for each variable as there are time steps times the slab size. In addition, an extra parameter table containing the *time lags* of interest is passed, i.e., those time steps for which the statistics should be computed with respect to the initial time step.

   (a) Auto-correlative statistics:

   Learn: calculate minimum, maximum, mean, and centered M_2 aggregates for a variable with respect to itself for a set of specified time lags (i.e., time steps between data sets of equal cardinality assumed to represent the same variable distributed in space);

   Derive: calculate unbiased auto-covariance matrix estimator and its determinant, Pearson auto-correlation coefficient, linear regressions (both ways), and fast Fourier transform of the auto-correlation function, again for a set of specified time lags;

   Assess: mark with squared two-dimensional Mahalanobis distance.

A utilization example of the statistical engines of VTK in ParaView is shown in Figure 3. Specifically, a PCA analysis is performed on a quadruple of variables of interest in a 2D flame simulation, whereby the statistical model is calculated (learn and derive operations) on a randomly-sampled

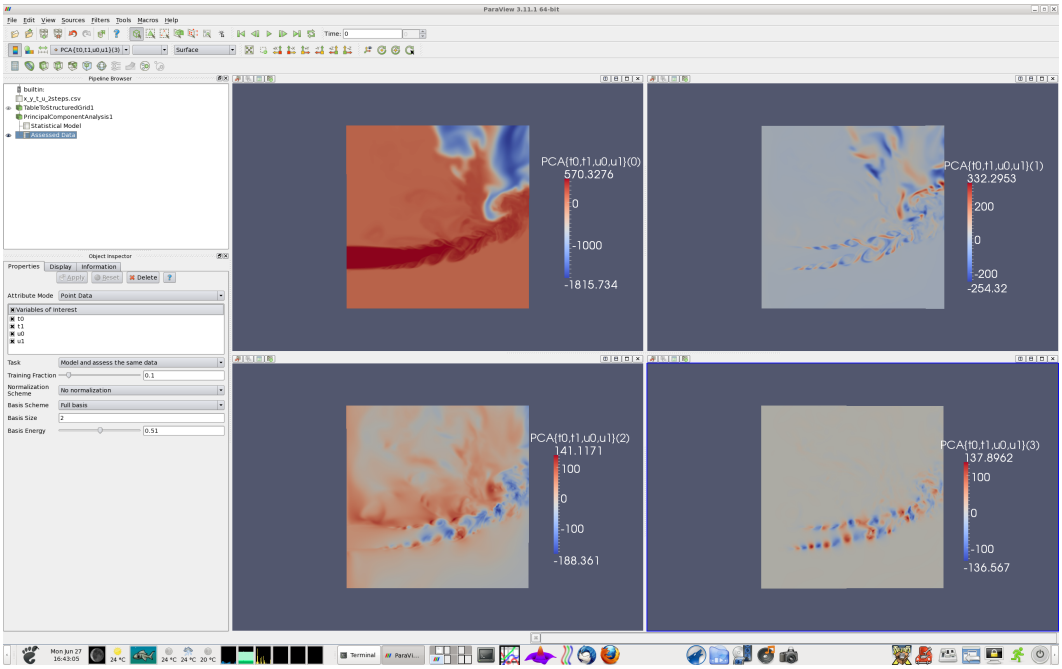

Figure 3. Several of the VTK parallel stastistics engines are integrated into ParaView. Example using PCA on 4 data set attributes.

subset of $\frac{1}{10}$-th of the entire data set, after which all points in the data set are marked with their respective relation deviations from this model.

1.3 Input and Output Ports

The statistics algorithms have by default 3 input ports (except in one case at the time of writing) and 3 output ports as follows:

Input Port 0: This port is identified as `vtkStatisticsAlgorithm::INPUT_DATA` and is used for learn data.

Input Port 1: This port is identified as `vtkStatisticsAlgorithm::LEARN_PARAMETERS` and is used for learn parameters (e.g., initial cluster centers for *k*-means clustering, time lags for auto-correlation).

Input Port 2: This port is identified as `vtkStatisticsAlgorithm::INPUT_MODEL` and is used for a priori models.

Input Port 3: At the time of writing, this port is only defined for the divergence statistics engine; it contains a reference to a `vtkTable` containing a set of per-variable normalization values,

i.e., values that are considered "ideal" for each variable (the default value for any variable, in the absence of a corresponding ideal value, being considered to be 1 for 100%.)

Output Port 0: This port is identified as `vtkStatisticsAlgorithm::OUTPUT_DATA` and mirrors the input data, plus optional assessment columns.

Output Port 1: This port is identified as `vtkStatisticsAlgorithm::OUTPUT_MODEL` and containts any generated model.

Output Port 2: This port is identified as `vtkStatisticsAlgorithm::OUTPUT_TEST` and is currently experimental and not used by all statistics algorithms[1].

All input and output ports are of type `vtkTable`, with the exception of both input and output ports 1 which are of type `vtkMultiBlockDataSet`. Note that in earlier implementations of these filters it was also possible for ports 1 to be of type `vtkTable`, however this is no longer the case.

In the following sections, we explain the concept of *divergence statistics* and how we used them to devise a new engine with the specific intent of providing quantitative performance analysis of large-scale computational clusters, either real or simulated.

[1] In earlier implementations and reports it was called `vtkStatisticsAlgorithm::ASSESSMENT`, a key which has been deprecated since.

2 Divergence Statistics

In this section, we present a summary on the notion of *divergence statistics*, along with a description of how we use them to quantitatively assess how observed data samples differ from an idealized model.

2.1 Statistical Divergences and Distances

The term *statistical divergence* is used to describe a series of statistical techniques to quantify the discrepancy between two discrete distributions. Such divergence functions are positive definite, but in general neither are symmetric nor satisfy the triangle inequality: as such, they do cannot be called *distances*. This is the reason why the more general term *divergence* is used.

In particular, one interesting family of statistical divergences is that of *f-divergences*, which are defined between distributions P and Q with respective probability densities p and q as:

$$(\cdot||\cdot) : (p,q) \longmapsto \int_{\mathbf{R}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

where f is a convex function such that $f(1) = 0$, cf. [Bas10] for more details. In the case of discrete probability distributions, which is that which is of interest to us here, where p and q instead refer to probability mass functions, the definition becomes:

$$(\cdot||\cdot) : (p,q) \longmapsto \sum_{x_i \in \mathscr{S}} q(i) f\left(\frac{p(x_i)}{q(x_i)}\right)$$

where $\mathscr{S}$ denotes the union of the sample spaces (i.e., where probability is nonzero) of P and Q. Note that in the case where those two sample spaces do not exactly overlap, some terms in the sum become degenerate and the divergence must be calculated as a limit.

2.2 Method

For the sake of HPC performance analysis, our approach is to compare an observed (i.e., empirical) distribution of values for some set of variables of interest (e.g., measurements of network traffic, CPU utilization, etc.) with respect to an ideal distribution, materialized by a probability mass function whose entire weight (1) is located at a user-defined, variable-specific "ideal" value. For instance, in the case of CPU utilization, our method is to quantify the discrepancy between the empirical distribution observed across a number of compute cores with respect to an ideal 100% CPU load for all cores.

With this goal in mind, we have implemented the following 5 statistical divergences and distances, which provide often qualitatively similar results, but sometimes reveal different details as our first analyses have shown. For this reason, we have not decided on a particular divergence against the others.

- The *total variation distance* $\delta(\cdot,\cdot)$, obtained when $f(u) = \frac{1}{2}|u-1|$, and which is a distance in the true sense of the term (being symmetric and satisfying the triangle inequality). Note that in our case, where we focus on discrete distributions, it is the same distance as the *1-distance*, up to a factor of 2:

$$\delta(P,Q) = \frac{1}{2}||P-Q||_1.$$

  We prefer to use the TVD for it has an upper bound of 1, which is convenient, but in statistical literature the 2 are sometimes confounded.

- The *Hellinger distance* $d_H(\cdot,\cdot)$, also symmetric and satisfying the triangle inequality, obtained by taking the square root of the f-divergence associated with $f(u) = (\sqrt{u}-1)^2$. Again in the case of discrete distribution, there is a relationship with a known norm, in this case the 2-distance (or Euclidean distance): specifically,

$$d_H(P,Q) = \frac{1}{\sqrt{2}}||\sqrt{P}-\sqrt{Q}||_2.$$

  In addition, d_H also has an upper bound of 1.

- The *Bhattacharyya coefficient* $b(\cdot,\cdot)$ is the f-divergence obtained when $f(u) =$. We then define the *Bhattacharyya semi-distance* $d_B = -\log b$. It is a semi-distance because it satisfies all the axioms of a distance, except for the triangle inequality. In particular, unlike divergences in general, it is symmetric. However it is not bounded and takes on an infinite value when the respective supports of P and Q are disjoint.

- The *Kullback-Leibler divergence* $\Delta(\cdot||\cdot)$, obtained with $f(u) = u\log_2 u$. Albeit not a statistical distance, it is useful in particular due to its non-symmetricity that allows for the giving two different meanings to the distributions P and Q: in our case, P will be viewed as the model distribution, representing a desired outcome. In this context, $\Delta(P||Q)$ quantifies the amount of information lost when Q is observed instead of the "ideal" P. Unlike the aforementioned distances, the Kullback-Leibler divergence is not bounded above and when P and Q do not have exactly the same support, it can takes its values in $[0,+\infty]$ by using the symbolic notations $0\log 0 = \lim_{x\to 0^+} x\log x = 0$ and $\frac{1}{\log 0} = \lim_{x\to 0^+} \frac{1}{\log x} = +\infty$.

- The χ^2 *divergence* $\chi^2(\cdot||\cdot)$, obtained with $f(u) = (u-1)^2$ which too is not a distance. Using the limits for degenerate cases at 0, $\chi^2(\cdot||\cdot)$ takes its values in $[0,+\infty]$. Also called *Pearson divergence*, it plays an important role in statistical literature in particular as result of its relationship with the homonymous χ^2 hypothesis testing.

Note that, in the context of HPC performance analysis, performed either experimentally (i.e., measured values on a real, live system performing an actual computation) or by simulation (for instance, using Sandia's Structural Simulation Toolkit SST/Macro), typical analyses will have divergence analyses repeated at regular time intervals, in order to obtain a time-series analysis which can be further processed to obtain a space-time quantitative perfomance aggregate value. This last step is not described here, as it is entirely independent of the per-step divergence analysis.

3 The Divergence Statistics Engine

3.1 Implementation Details

The divergence statistics engine is implemented as `vtkDivergenceStatistics`, a subclass of the VTK class `vtkOrderStatistics` which we had developed earlier [PT10], wherefrom the divergence statistics inherits the Learn, Derive, and Assess operations. We used this approach in order to leverage the fact that the order statistics engine, as part of its Learn and Derive operations, computes an empirical probablility mass function (EPMF) of the input set of observables, an EPMF which we then use in a Test operation specific to the divergence statistics engine, whose output is a `vtkTable` containing the 5 divergences outlined in §2.2.

Please note however that, although a subclass of a class which is part of the VTK library, `vtkDivergenceStatistics` is **not** released open-source and in particular is not part of the VTK package, as permitted by the BSD-like license of VTK which unlike other software license models does not force developers to release extensions to the original library.

3.2 Usage

It is fairly easy to use the serial statistics classes of VTK; it is therefore just as easy to use the new divergence statistics engine. All that is required is a build of VTK installed on your system. In addition, a separate build of the divergence statistics engine is necessary, as it is not part of VTK proper; however, as a convenience we provide as part of the tarball containing the source code a `CMakeLists.txt` file which allows you to do exactly that, using the `CMake` cross-platform build utility.

For example, Listing 1 demonstrates how to calculate divergence statistics on each column of an input set `inData` of type `vtkTable*` with an associated set of input parameters representing the ideal (or "peak") values for each variable of interest, and no subsequent data assessment. It is required that this input data have numeric type (i.e., double), although the order statistics engine allows for non-numeric types as well. This additional requirement which is not needed for the computation of the EPMF with the order statistics engine nonetheless allows for a simpler implementation of the divergence engine without, in our knowledge, resulting in a loss of generality for the sake of performance analysis: all application cases which came to our attention measure numeric quantities as part of the analysis process.

For univariate statistics algorithms, as is the case for the order and divergence statistics engines, calling `AddColumn()` for each column of interest is sufficient – each request R_i can by definition only reference a single column and so the filter automatically turns each column of interest into a separate request.

```
void Foo( void* arg1, void* arg2 )
{
  // Assume the input dataset is passed to us
  vtkTable* inputData = static_cast<vtkTable*>( arg1 );

  // Assume ideal values are passed to us as well
  vtkTable* idealVals = static_cast<vtkTable*>( arg2 );

  // Create divergence statistics class
  vtkDivergenceStatistics* ds = vtkDivergenceStatistics::New();

  // Set input data and additional parameter ports
  ds->SetInputData( vtkStatisticsAlgorithm::INPUT_DATA, inputData );
  ds->SetSpecifiedPeakValues( idealVals );

  // Select all columns in inputData
  for ( int c = 0; c < inputData->GetNumberOfColumns(); ++ c )
    {
    ds->AddColumn( inputData->GetColumnName[c] );
    }

  // Calculate statistics with all operations except Assess
  ds->SetLearnOption( true );
  ds->SetDeriveOption( true );
  ds->SetAssessOption( false );
  ds->SetTestOption( true );
  ds->Update();
```

Listing 1: A subroutine for calculating divergence statistics.

3.3 Application Example

We illustrate the use of the divergence statistics engine by incorportating into a the simulation of a computational cluster comprising 64 switches with one compute node per switch, using a 3-dimension torus (4x4x4) network topology, simulated using Sandia's Structural Simulation Toolkit (SST/Macro) using only one core of a Linux machine. It is outside of the scope of this report to discuss the SST/Macro application, cf. [WK15] for more detailed information in this regard.

The experimental set-up uses of a C++ skeleton of a parallel application with a quadratic inter-process communication pattern, sending data packets with 4096B fixed size, from all to all 64 compute nodes, and then emulating a fixed compute time of 10μs per process upon receive. The SST/Macro visualization engine allows for for qualitative analysis, but more thorough analysis

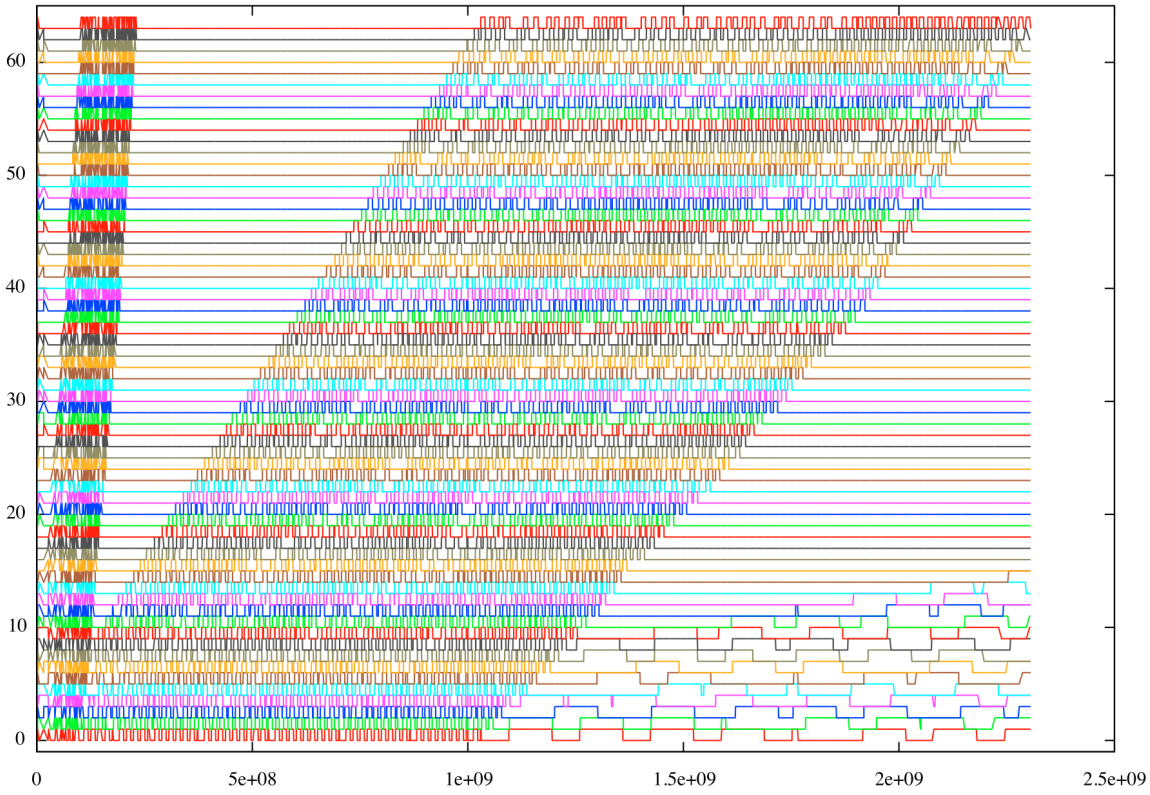

Figure 4. Qualitative analysis of CPU loads obtained with SST/Macro simulation: the x-axis represents time expressed in ns, and compute core loads for each of the 64 cores participating in the simulated run are represented along the y-axis, at scale but with an offset equal to the rank ID.

requires accurate quantification of resource utilization, allowing for comparison between different programming models, run parameters, etc.

For instance, if compute load is the variable of interest, which is often the case in first order analysis, one can simply plot the variation of the quantities of interest as shown in Figure 4: one readily observes that the quadratic communication pattern results in sub-optimal resource allocation. However, as the number of components (in this case, the number of compute cores) grows it becomes increasingly difficult to make sense of this raw data. Furthermore, this qualitative assessment cannot be used to make objective comparisons between various input parameters (which can represent both hardware and software design choices).

Meanwhile, using the divergence statistics engine embedded in a time-series analysis as outlined in §2.2, one readily obtains a quantitative measurement of the discrepancy between the simulation results and the ideal value, which is here chosen to correspond to a full utilization of every compute core at every time step, as shown in Figure 5 for the Hellinger divergence be-

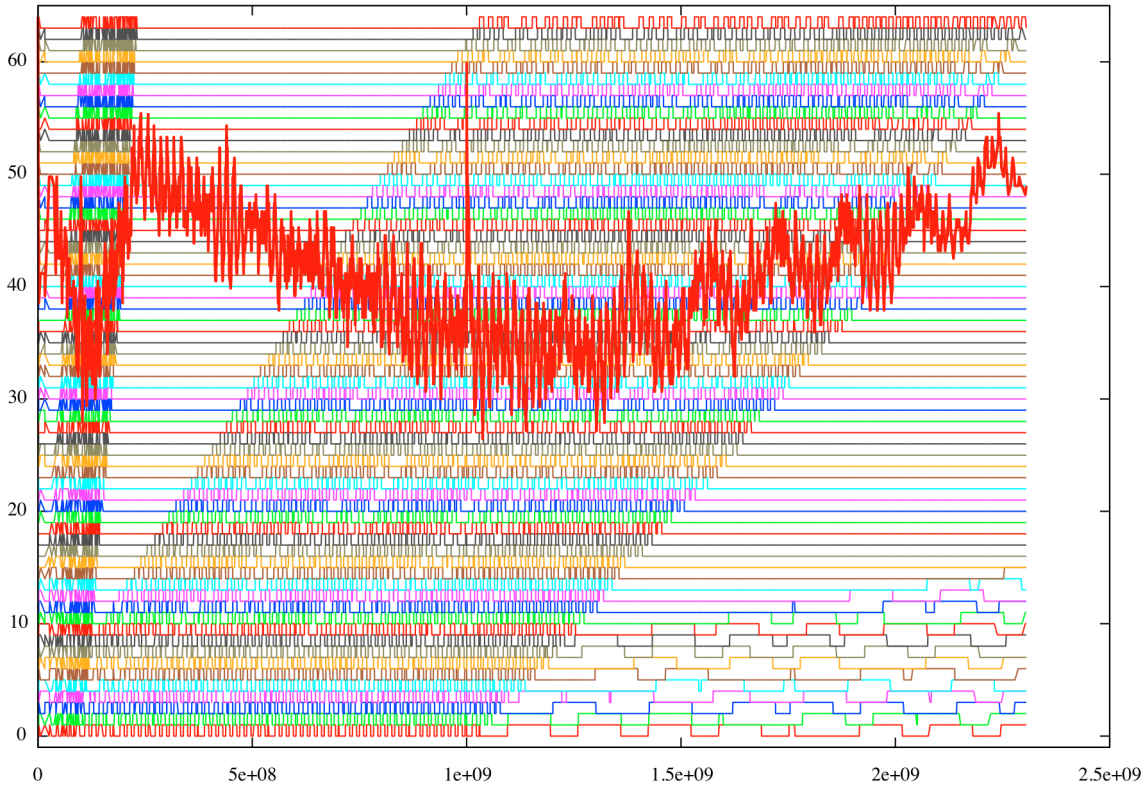

Figure 5. Quantitative analysis of CPU loads obtained with Hellinger divergence analysis of SST/Macro simulation: Figure 4 overlaid with Hellinger divergence, scaled by a factor of 64 of compute loads as they diverge from user-defined 100% peak value.

tween simulated compute core loads (equal to either 0 or 1) and the peak value of 1. Note that whether this particular choice of a desired, "ideal" peak value is good or even realistic is not part of the discussion here; moreover, our implementation allows the user to compare, *ad libitum*, various optimization strategies thanks to the `SetSpecifiedPeakValues()` function of the `vtkDivergenceStatistics` class. In any event in the case displayed here, using a divergence immediately allows for a global aggregation of all variable values into a single, time-varying quantity which can be further acted upon (e.g. for visualization, optimization, or resource re-allocation).

For instance, as shown in Figure 6 (which for the sake of legibility only displays 4 amongst the 5 available statistics), all divergences reveal qualitatively similar results, but also immediately reveal aggregate information that was not observable from the collection of raw plots in Figure 4: in particular, one can directly evince that after roughly 1s of simulated time, i.e., just before the middle of the run, an absolute low-point (i.e., a high divergence value) is reached. Of course, the Kullback-Leibler and Bhattacharyya divergences, being unbounded above, reveal this in a more obvious way that the Total Variation or the Hellinger distances whose range is $[0, 1]$. Meanwhile, the latter might make the distances (as opposed to the non-distance divergences) more apt for an optimization

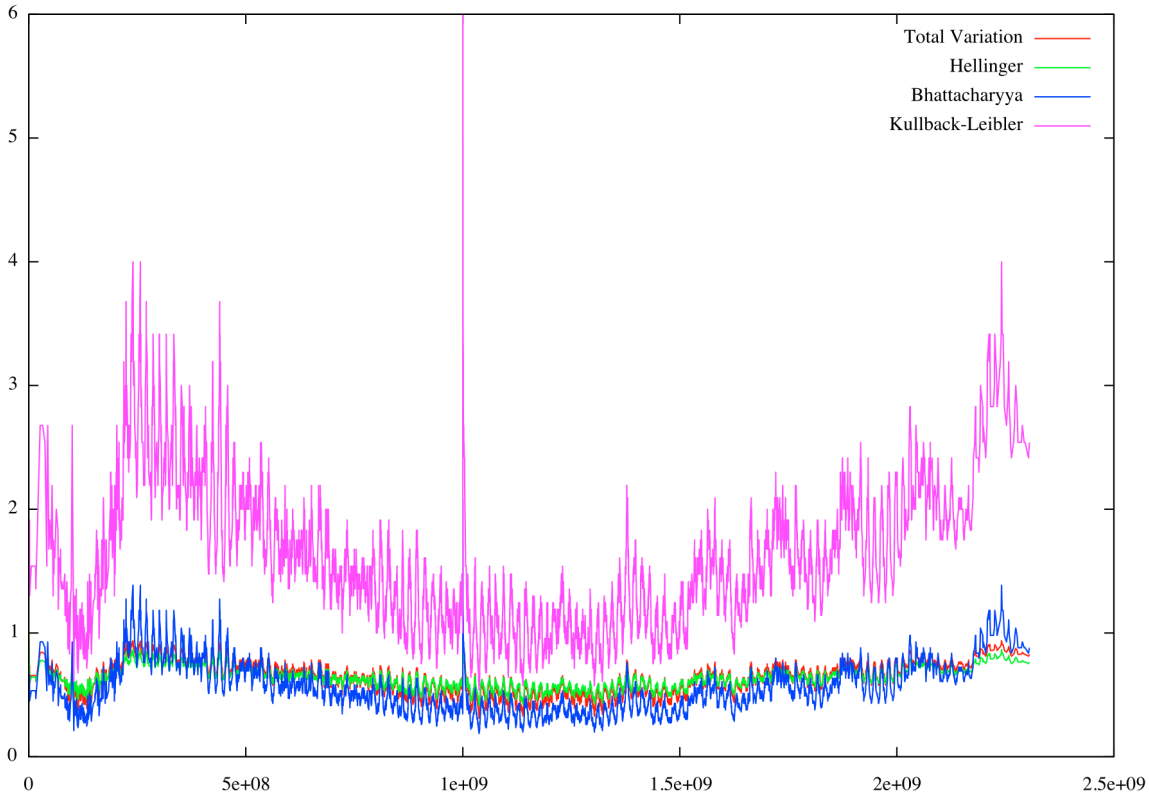

Figure 6. Quantitative analysis of CPU loads obtained with divergence statistics analysis of SST/Macro simulation: divergence values (y-axis) vs. time expressed in ns (x-axis), for 4 out of the 5 available divergences provided by `vtkDivergenceStatistics`.

loop as their values will not cause out-of-range numerical errors; for this reason, we decided to *not* decide as to whether true distances are more or less suited towards our performance analysis goals than unbounded divergences, but rather to provide a panel of choices of possible qualitative assessments. Also, it is worth acknowledging here that our choice of a base-2 implementation for the Kullback-Leibler divergence (which is not always defined as such, depending on the authors, with a natural logarithm also often used instead of $\log_2$) allows for a direct, intuitive understanding of the divergence value: for instance, the computed maximum of 6 corresponds to the fact that only $\frac{1}{2^6}$ of the entire collection of measured variables is at the desired peak value, i.e., a single compute core is computing at that point amongst all 64, a very poor resource allocation from this standpoint indeed.

References

[Bas10] M. Basseville. Divergence measures for statistical data processing. Publications Internes de l'IRISA PI-1961, IRISA, November 2010.

[BPRT09a] J. Bennett, P. Pébay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Proc. 2009 IEEE International Conference on Cluster Computing*, New Orleans, LA, August 2009.

[BPRT09b] J. Bennett, P. Pébay, D. Roe, and D. Thompson. Scalable multi-correlative statistics and principal component analysis with Titan. Sandia Report SAND2009-1687, Sandia National Laboratories, March 2009.

[BPT09] J. Bennett, P. Pébay, and D. Thompson. Scalable *k*-means statistics with Titan. Sandia Report SAND2009-7855, Sandia National Laboratories, November 2009.

[Inc10] Kitware Inc. *The VTK User's Guide, version 5.4*. Kitware, Inc., 2010.

[PÓ8] P. Pébay. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. Sandia Report SAND2008-6212, Sandia National Laboratories, September 2008.

[PB13] P. Pébay and J. Bennett. Parallel auto-correlation statistics with VTK. Sandia Report SAND2013-3435, Sandia National Laboratories, April 2013.

[PT08] P. Pébay and D. Thompson. Scalable descriptive and correlative statistics with Titan. Sandia Report SAND2008-8260, Sandia National Laboratories, December 2008.

[PT09] P. Pébay and D. Thompson. Parallel contingency statistics with Titan. Sandia Report SAND2009-6006, Sandia National Laboratories, September 2009.

[PT10] P. Pébay and D. Thompson. Parallel order statistics with Titan. Sandia Report SAND2010-6466, Sandia National Laboratories, September 2010.

[PTB10] P. Pébay, D. Thompson, and J. Bennett. Computing contingency statistics in parallel: Design trade-offs and limiting cases. In *Proc. 2010 IEEE International Conference on Cluster Computing*, Heraklion, Crete, Greece, September 2010.

[PTBM11] P. Pébay, D. Thompson, J. Bennett, and A. Mascarenhas. Design and performance of a scalable, parallel statistics toolkit. In *Proc. 25th IEEE International Parallel & Distributed Processing Symposium, 12th International Workshop on Parallel and Distributed Scientific and Engineering Computing*, Anchorage, AK, U.S.A., May 2011.

[WBS08] B. Wylie, J. Baumes, and T. Shead. Titan informatics toolkit. In *IEEE Visualization Tutorial*, Columbus, OH, October 2008.

[WK15] Jeremiah J. Wilke and Joseph P. Kenny. Using discrete event simulation for programming model exploration at extreme-scale: Macroscale components for the structural simulation toolkit (SST). Sandia Report SAND2015-1027, Sandia National Laboratories, February 2015.

DISTRIBUTION:

2	MS 9018	Central Technical Files, 8944
1	MS 0899	Technical Library, 9536

Sandia National Laboratories