# SANDIA REPORT

SAND2011-4648
Unlimited Release
Printed July 2011

# Current Parallel I/O Limitations to Scalable Data Analysis

Philippe Pébay and Ajith Mascarenhas

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.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

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

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

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

SAND2011-4648
Unlimited Release
Printed July 2011

# Current Parallel I/O Limitations to Scalable Data Analysis

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

Ajith Mascarenhas
Sandia National Laboratories
M.S. 9051, P.O. Box 969
Livermore, CA 94551, U.S.A.

**Abstract**

This reports describes the limitations to parallel scalability which we have encountered when applying our otherwise optimally scalable parallel statistical analysis tool kit to large data sets distributed across the parallel file system of the current premier DOE computational facility.

# Acknowledgments

The authors would like to thank:

- Robert Clay, for supporting this work;
- Jackie Chen, who graciously offered the Lifted Ethylene Jet dataset, and computational time on the `Jaguar-pf` DOE/ORNL peta-scale platform; and
- Hongfeng Yu, for generating the visualizations of the Lifted Ethylene Jet dataset.

# Contents


1 Introduction . . . 7
1.1 Background . . . 7
1.2 Summary . . . 8
2 The Learn-Derive-Assess Pattern: Design and Performance . . . 9
2.1 Operations . . . 9
2.2 Parallelism . . . 11
2.3 Code Structure . . . 14
3 Petascale Data Analysis with Parallel I/O . . . 17
3.1 Parallel Data Set . . . 17
3.2 Evaluation Criterion . . . 18
3.3 Results with Parallel I/O . . . 19
4 Conclusion and Perspectives . . . 23
References . . . 26

This page intentionally left blank

# 1 Introduction

This report describes our study to evaluate the effect of parallel I/O on the overall scalability of a parallel data analysis pipeline using our scalable parallel statistics tool kit [PTBM11]. In this goal, we tested it using the `Jaguar-pf` DOE/ORNL peta-scale platform on a large combustion simulation data under a variety of process counts and domain decompositions scenarios.

## 1.1 Background

In earlier work [BPRT09a, PTB10, PTBM11], we have presented a consistent design pattern, called *Learn-Derive-Assess*, and a programmatic interface for a wide range of data analysis methods, specifically aimed at data sets distributed on parallel computational platforms at peta-scale and beyond. We have described how each of the available algorithm fits the pattern and how its implementation scales in a distributed-memory parallelism context. The design decisions made during development were motivated by two primary factors:

(i) We wanted to mimic the predominant types of data analysis workflows, so that a data analyst using our framework would find it natural and intuitive to use.

(ii) The design had to be conducive to embarrassingly parallel implementations when possible.

This was accomplished by isolating those parts of the analysis which by construction are not embarrassingly parallel (due to the mathematics of the statistical analysis itself, not due to our design) so that parallel design trade-offs are limited to those components where embarrassingly parallel implementations are not viable.

In addition, we have provided production level implementation of this design pattern as C++ classes in VTK [PÍ0], itself part of the Titan Informatics Toolkit [tit]; specifically the following 7 parallel statistics workflows are implemented at the time of writing:

- descriptive statistics,
- histograms and order statistics,
- bivariate linear correlation and regression,
- contingency statistics and information entropy,
- multi-variate linear correlation,
- principal component analysis,
- $k$-means clustering.

This list will most likely be expanded in the near future.

The source code is freely and publicly available as part of VTK, along with its test harness covering most of its features, which can be found in the `VTK/Infovis/Testing/Cxx/` sub-directory of VTK, version 5, with the additional benefit that all performance and reliability results which we claim can be independently verified. For additional usage details, the reader is invited to refer to the *VTK User's Guide* [PÍ10], specifically §8.6 (*"Statistics"*), pp. 192–198.

Regarding performance, we have provided detailed parametric studies conducted on several parallel computational clusters, and in particulier at DOE's premier computing facility, with up to 10,000 processes.

## 1.2 Summary

In Section §2, we briefly recall the key performance findings of the parallel data analysis algorithms which are implemented under our Learn-Derive-Assess design pattern. In Section §3, we first describe the data and the test platform used in our experiments, followed by results and conclusions.

# 2 The Learn-Derive-Assess Pattern: Design and Performance

In this section, we briefly recall how our design constraints have resulted in the splitting of the statistical analysis workflow into 4 distinct operations. In this report we only explain the generic principles of each operation, illustrating it with the case of descriptive statistics for ease of understanding. For details as to how the operations are articulated for each of the 7 currently available parallel statistical engines, we refer to [PTBM11]. In addition, we explain how we implemented these algorithms in the context of data parallelism to achieve as much parallel speed-up and scalability as possible.

## 2.1 Operations

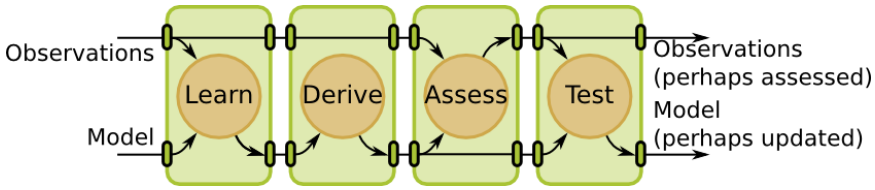


**Figure 1.** The 4 operations of statistical analysis and their interactions with input observations and models. When an operation is not requested, it is eliminated by connecting input to output ports.

In order to meet the two overlapping but not exactly congruent design requirements outlined in § 1, we partition the statistical analysis workflow 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 1. However, it is 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, PÍ0]) only the first 3 operations are mentioned; the Test operation, which we initially saw as a part of Derive, was separated out for reasons we discuss in § 2.2. These operations, performed on a request comprising a set of columns of the input observations table, are further explicated as follows:

### Learn

Calculate a primary statistical model from an input data set. By primary, we mean the minimal (in particular, non-redundant) representation of the desired model for a given statistical technique.

### Derive

Calculate a more detailed statistical model from a primary model. By ”more detailed”, we mean a model which includes the primary model along with additional, derived or redundant statistics, generally more commonly used for analysis purposes than the primary statistics from which they derive.

### Assess

Annotate each observation with a number of quantities relative to a given a statistical model. These quantities generally measure how well the particular observation coincides with the model. Note that the model used to annotate an observation need not have been calculated from the same data.

### Test

Given a statistical model, and possibly a data set, calculate at least one test statistic so at least one hypothesis can be tested. For descriptive statistics, a Jarque-Bera test of goodness of fit [JB87] is performed, calculating the test statistic with the skewness and kurtosis, and then retrieving the corresponding $p$-value from the $\chi^2$ distribution with 2 degrees of freedom by a single call to R for all variables at once; therefore, a second pass through the data is not needed.

**Table 1.** Left: A table of observations used as input data by a statistics algorithm. Right: the same observations, annotated by the Assess operation of descriptive statistics engine with the model obtained with the Derive and Assess operations.

| row | A |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 5 |
| 5 | 7 |
| 6 | 11 |

| row | A | $d$(A) |
|---|---|---|
| 1 | 1 | 1.03315 |
| 2 | 2 | 0.76363 |
| 3 | 3 | 0.49411 |
| 4 | 5 | 0.04492 |
| 5 | 7 | 0.58395 |
| 6 | 11 | 1.66202 |

*Example* 2.1. In order to illustrate this general design pattern for data analysis methods, consider what is probably the most widely used, at least for the Learn and Derive operations, namely, descriptive statistics. Specifically, consider the univariate observations given in Table 1, left, which has 6 observations of variable (column) A.

Our primary model for descriptive statistics, calculated by the Learn operation, comprises the sample size, minimum, maximum, mean, and centered $M_2$, $M_3$ and $M_4$ aggregates (*cf.* [BPRT09a]) of the data. In our case of variable B in Table 1, these statistics are 6, 1, 11, $4.8\bar{3}$, $68.8\bar{3}$, $159.\bar{4}$, and $1759.819\bar{4}$, respectively.

The additional statistics calculated from the primary model by the Derive operation are: variance, standard deviation, skewness, kurtosis, and sum, with various estimators being offered for these statistics, save for the sum. In our running example, the respective values of these derived statistics are $13.7\bar{6}$, 3.7103, 0.52025, $-1.4524$, and 29. Note that there is now redundancy in the derived model, meaning that there is potential for inconsistency if one decides to modify some entries of the derived model, while this cannot happen with the primary model. For instance, the standard deviation, variance, and $M_2$ aggregate can be mutually inconsistent.

The Assess operation of the descriptive statistics engine marks each datum with its relative deviation with respect to the model mean and standard deviation, i.e., the one-dimensional Mahalanobis distance [Mah36]. In addition, our implementation allows for the use of signed deviations in this one-dimensional case, should the user prefer this over the positive-definite Mahalanobis distance. The result of this operation can be seen in Table 1, right, where a new column $d(\mathrm{A})$ had been appended with the corresponding assessment for each input datum.

Last, the Test operation for the observations of variable A yields a test statistic of 0.79803 for a corresponding $p$-value of 0.67098, meaning, e.g., that the null hypothesis of normality cannot be rejected at the significance level of, e.g., 5%.

## 2.2 Parallelism

While the first of our design goals for partitioning the workflow into 4 operations was to mimic the typical use patterns of statistical analysis, the second design goal was the enabling of scalable parallelization on distributed data. And indeed, from the parallelism standpoint, the partition reduces two operations to the map-reduce pattern [DG04] and the remaining two are embarrassingly parallel.

Specifically, Learn is essentially a special case of the map-reduce pattern [DG04], a framework within which the map function generates a set (key,value) pairs in parallel. All intermediate values associated with the same intermediate key are then merged by the reduce function to compute the final solution of interest. In some of our statistical algorithms, it is not necessary to communicate the keys for there is a fixed number of them, identical across all processes, and these keys may be ordered uniquely, so sending values alone is unambiguous. However, for other algorithms, tables with an arbitrary number of key-value pairs must be communicated and different keys may be

present on each process.

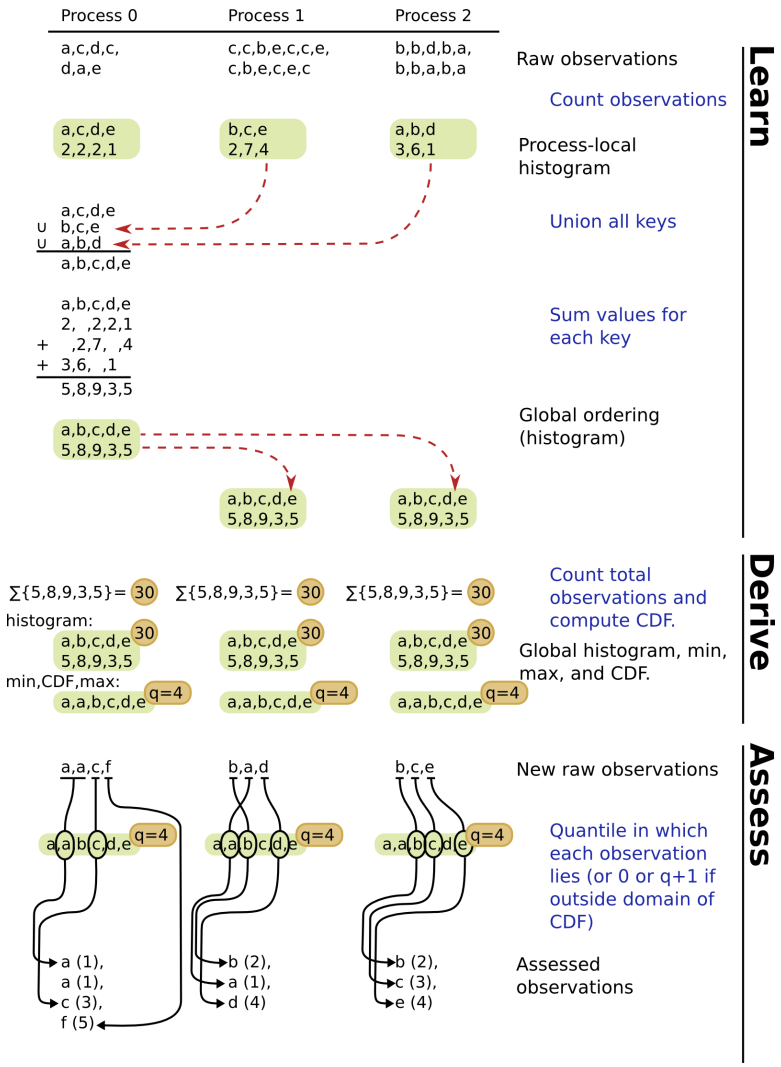


**Figure 2.** Example showing parallel execution of the Learn, Derive, and Assess operations of order statistics on 3 processes.

By construction, Learn is the only operation which always requires inter-process communication; for instance, in the case of descriptive statistics, cardinality, extremal values, and centered aggregates up to the fourth order must be exchanged and updated to assemble a global model. The Test operation may, for some types of analyses, also require inter-process communications when the type of statistical test being performed necessitates a second pass through the data. As mentioned in § 2.1, we separated Test from our original design of Derive because:

(i) Test tends to be more computationally intensive and in our current implementation relies in some cases on calls to the R library [R] for $p$-value retrieval (but in an embarrassingly parallel way as explained below),

(ii) For some statistics algorithms it requires a second pass through the data in order to compute per-observation quantities involving derived model information that cannot be obtained in an on-line fashion[1], henceforth requiring a second step of parallel updates, and

(iii) it rests on assumptions which are not accepted by all statisticians, namely statistical significance.

Figure 2 shows the communication patterns and parallel computations performed when calculating order statistics on simple character data distributed across 3 processes; note that the Test operation is not depicted here as its implementation is not complete yet for all statistical engines and some changes are to be expected in the near future.

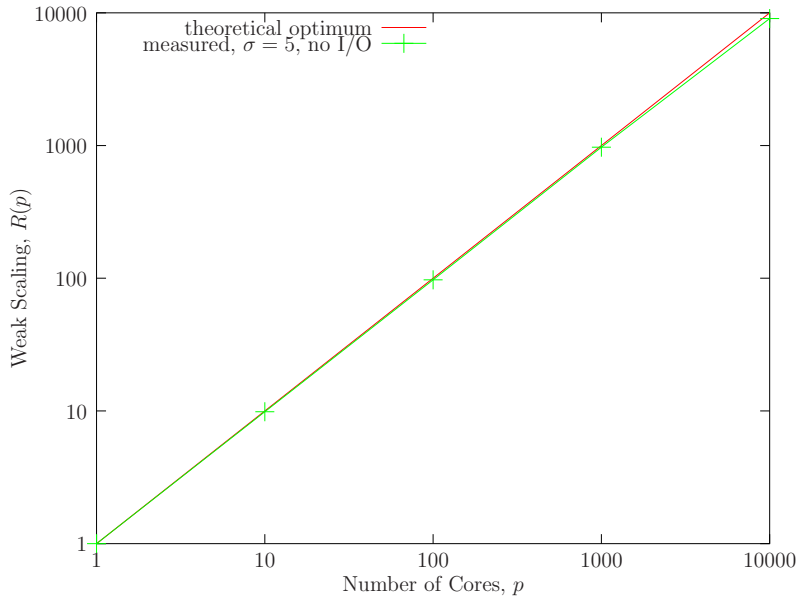


**Figure 3.** Parallel order statistics: weak scaling (at constant work per core), synthetic data of size $2 \times 10^7$, no I/O calls.

We note that the Learn (and Test) operations of most of the currently implemented statistical algorithms demonstrate optimal parallel scaling properties. Specifically, in [BPRT09a] we have shown how to perform these updates in a numerically stable – yet single-pass – way for all statistics algorithms which make use of moments and co-moments, hence resulting in optimal parallel scaling for the Learn and Test operations for such statistics.

On the other hand, in [PTB10], we discussed the design trade-offs and limitations encountered when computing contingency tables in parallel, which resulted in our choosing of what we called the *Full-reduce+broadcast* approach; in this case, the Learn operation of the algorithm is difficult

[1]For example, the multi-variate normality test requires that multi-skewness and multi-kurtosis be computed, but those estimators in turn need each observation to be projected to the eigenvectors of the data set, which cannot be computed incrementally and thus are not available until after all observations have been processed.

to characterize because its parallel scaling properties range from embarrassingly parallel to serial, depending on the distribution of the input data. Nevertheless, we established showed that the case where our parallel design does not scale well exactly corresponds to the class of problems for which contingency tables are not suited for analysis, hence validating our design pattern even for non moment-based algorithms. In [PTBM11] we extended and generalized these results for another case of quanta-based statistics, namely, order statistics, where the primary model is represented by histogram, whereby the input data is also quantized. In particular, we establish that when used in their proper context, that is, when the input data is honestly discrete (as opposed to quasi-diffuse), then our parallel design trade-offs allow for near-optimal scalability, demonstrated with up to $10,000$ processes of `Jaguar-pf`, the premier computing facility of DOE, as illustrated in Figure 3.

For the Test operation of those engines which currently rely on R for the retrieval of the $p$-values corresponding to calculated $\chi^2$ statistics, it is important to note that the invokation of R is done by each process independently from all other processes, each one retrieving the same $p$-value from a previously globally calculated $\chi^2$ statistic. Therefore, the use of the (serial) R package does not invalidate the embarrassingly parallel nature of this operation.

On the other hand, the Derive and Assess operations are always embarrassingly parallel, by construction: deriving additional statistics from a primary statistical model always need only be executed locally, without communication, after all parallel updates of the primary model. Likewise, the Assess operation annotates only the observations in the portion of the data residing on the local process, and thus no communication is required. This results in a new data set distributed in the same way across processes but with additional columns for the annotations.

## 2.3 Code Structure

Each statistics algorithm is first implemented in C++ in a serial fashion as a subclass of a base statistics algorithm class that provides virtual methods for each of the 4 phases (Learn, Derive, Assess, and Test). Each serial implementation then serves as a base class for a parallel implementation which overrides Learn and, optionally, Test as well, as shown in Figure 4. These methods invoke the implementation of the serial superclass and then use an abstract communication-class instance to perform the inter-process communication required for the reduction operation. This separation between serial and parallel implementations also allows for experiments with different programming models: e.g., one where a full reduction operation might not be necessary to sufficiently, in some sense, approximate the global model.

The descriptive, correlative, multi-correlative, PCA, and $k$-means statistics algorithms perform their respective parallel aggregations using the update formulas presented in [BPRT09a], cf. [PT08, BPRT09b, BPT09] for implementation details. These moment-based parallel statistics classes thus all have optimal parallel scalability properties. Similarly, the order and contingency statistics classes respectively derive from the univariate and bivariate statistics classes and implement their own aggregation mechanisms for the Learn operation. However, unlike moment-based statistics (descriptive, correlative, multi-correlative, PCA, and $k$-means), these aggregation operations are

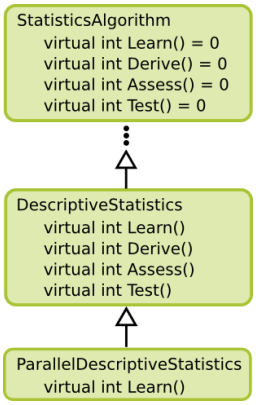


**Figure 4.** The inheritance pattern of the descriptive statistics implementation. This serves to illustrate how parallel communication is separated into a subclass which may use its superclass's implementation for all but the reduction operation.

not, in general, embarrassingly parallel; as a result, these statistical methods scale sub-optimally when used outside of their intended domain of applicability, as explained in [PTB10] for contingency statistics, and in [PTBM11] for order statistics. However in this report it is assumed that quantization-based statistics be used in their appropriate context, i.e., with honestly discrete data sets, and as such do not exhibit any loss of parallel scalability due to misuse.

# 3 Petascale Data Analysis with Parallel I/O

In this section, we first describe the parallel Lifted Ethylene data set used in our data analysis pipeline scalability tests, followed by a brief description of the computation platform and our evaluation criterion, and finally present the results of the tests.

## 3.1 Parallel Data Set

In turbulent non-premixed combustion, where the fuel and oxidizer reactant streams are segregated, the reactant streams must be molecularly mixed before reaction can occur. Therefore, the turbulent mixing rate is a key quantity in determining the overall burning rate and efficiency. Turbulent mixing is characterized locally by the scalar dissipation rate, generally denoted $\chi$.

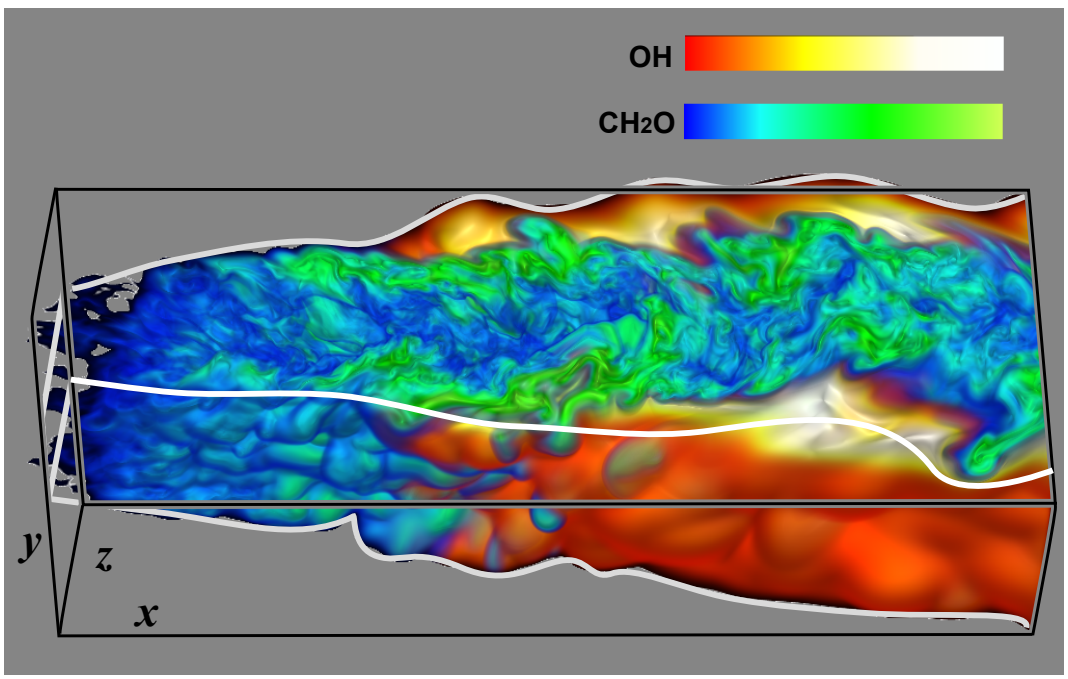


**Figure 5.** Lifted Ethylene flame shown with hydroxyl ($OH$) flame marker, and formaldehyde ($CH_2O$) upstream ignition marker. The fuel is injected into the left plane and the flame develops along the $x$-axis. The envelope lines are shown (white) to provide better perspective.

The chosen test data set comes from the Lifted Ethylene Jet [YRSC09] simulation calculated on `Jaguar-pf`, of which a volume rendering of two participating variables at one time-step is shown in Figure 5. Specifically for this study, one time-step of the scalar dissipation rate field, denoted $\chi$, will be used; this field is further illustrated in Figures 6 and 7. From this field, two distinct test

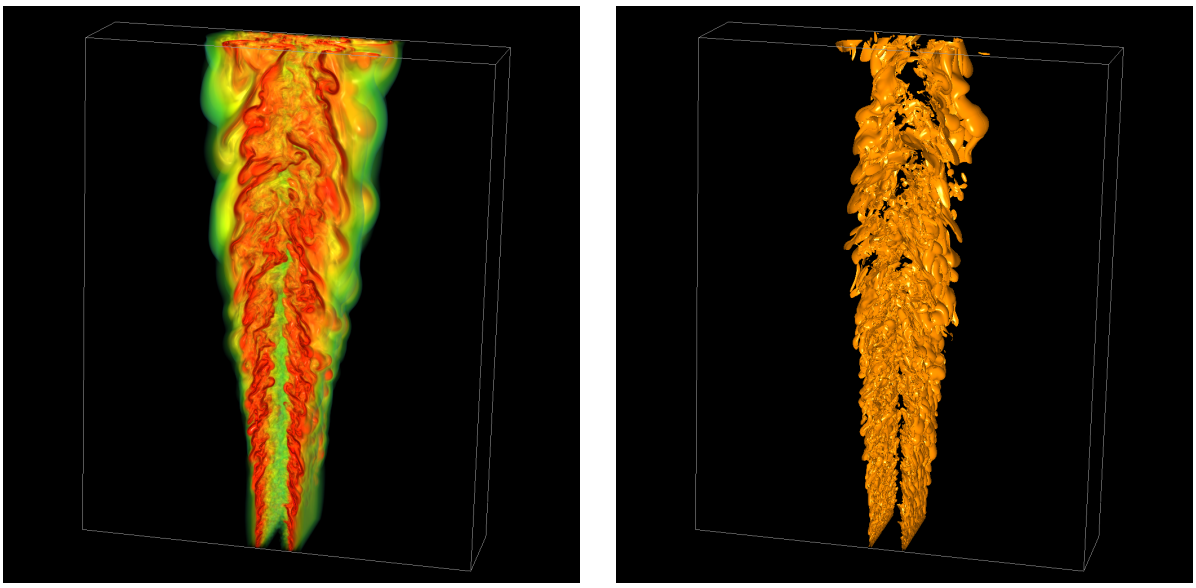

**Figure 6.** Scalar dissipation rate field ($\chi$) of the Lifted Ethylene Jet: volume rendering with transfer function chosen to emphasize the high $\chi$ values (left) and an iso-surface showing the structures in the field (right).

cases are extracted:

- The full data set, distributed across the system by the parallel file system, with dimensions of $2025 \times 1600 \times 400$ per time step.
- For baseline comparison with a case which can be run on a single core (which is not the case on the premier DOE test platform we use for the full data set), a subset of the above with size $200^3$ per time step.

We store the $\chi$ field in a single file per time-step. In each file, the data consists of one 32-bit floating point value per data point written in sequence such that the x-coordinates of the points vary the fastest, the y-coordinates the next, and the z-coordinates vary the slowest. In other words, the data organization corresponds to marching along the z-plane, and along the y-plane within each z-plane. Because this organization impacts the read time during our analysis, we will test a variety of processor configurations and domain decompositions in our experiments. These data sets are computed and stored on the petascale test bed, which we are now briefly describing.

## 3.2 Evaluation Criterion

In order to assess the effect of the parallel I/O system when moving real data to the parallel analysis engines, we build an analysis pipeline comprising:

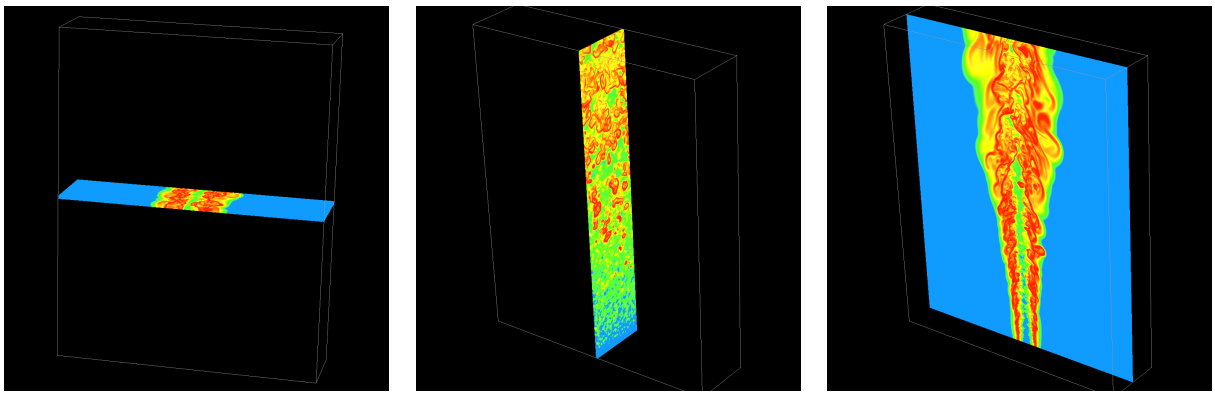

**Figure 7.** Scalar dissipation rate field ($\chi$) of the Lifted Ethylene Jet: Slices normal to the three grid axes.

1. the parallel data reader required to access the distributed test data chosen for these tests and
2. a statistics engine from the parallel tool kit,

so that parallel I/O calls are included in the measured wall times, together with the analysis itself.

We run our tests on `Jaguar-pf` [Jag], the primary system of the National Center for Computational Sciences at Oak Ridge National Laboratory. `Jaguar-pf` is a Cray XT5 system containing 224,256 compute cores in addition to dedicated login/service nodes. Each compute node contains two hex-core AMD Opteron processors, 16GB memory, and a SeaStar 2+ router.

With this setting, we use *strong scaling*, i.e., scaling at constant total work, as an evaluation criterion. Hereafter $p$ will denote the number of processes and $N$ the size of the problem, which in our case is the data set cardinality and remains constant for a given case of a strong scaling study. The wall clock time measured to execute the calculation is denoted $T_N(p)$. Then, strong scaling is defined as:

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

Some authors prefer to write the numerator as $T_s$ rather than $T_N(1)$ to make it clear that the parallel algorithm should be compared to the most efficient serial implementation available and not just the parallel algorithm run on a single process. Evidently, optimal (linear) scaling is attained when $S_N(p) = p$ and, therefore, strong scaling results can be visually inspected by plotting $S_N$ versus the number of processes: optimal scaling is revealed by a line, the angle bisector of the first quadrant.

## 3.3 Results with Parallel I/O

In this study, only Learn, Derive, and Assess operations are invoked, as Test relies on the external calls to R, whose execution we want to keep aside from our study. As the main purpose is to

evaluate the effect of parallel I/O on overall scalability, our timings naturally include execution times of the entire pipeline including parallel read routines and statistical analyses themselves. However, in order to be able to isolate the effect of the former, we also provide results obtained by executing only the latter with perfectly load-balanced synthetic data, whose creation time on the compute nodes is not included in the timings. Last, we make use of the type of statistical analysis which probably appears as the simplest and most natural by data analysts in general, namely, descriptive statistics, cf. [PT08] for details.

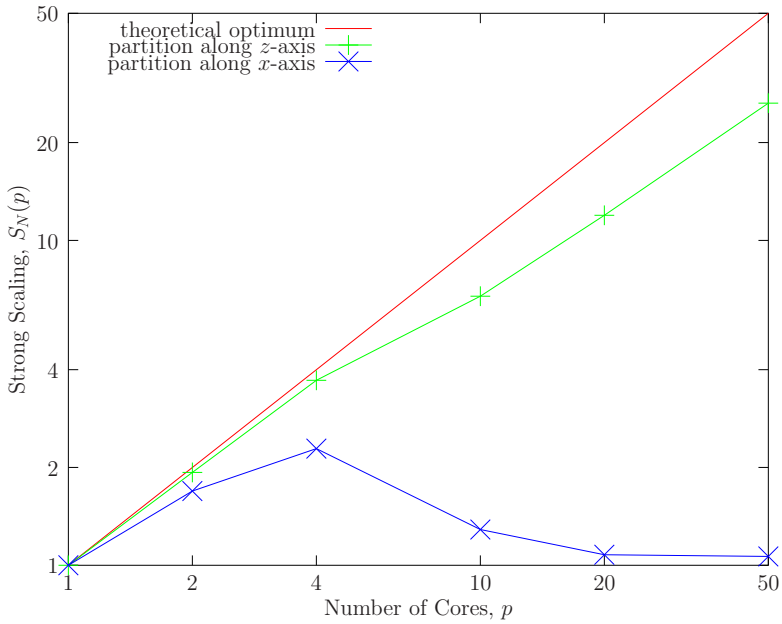


**Figure 8.** Parallel descriptive statistics: strong scaling (at constant total work) on `Jaguar-pf`, 200x200x200 case, including parallel I/O calls.

As illustrated in Figure 8 with the subset of size $200^3$, and even further with the full data set in Figure 9, parallel (strong) scaling degrades rapidly, even though some improvements can be made by ad-hoc data read strategies: specifically, it is possible to take advantage of the data layout when it is known of time, as shown with the smaller subset. However, the effects of such ad hoc approach quickly vanish as data set size increases, as the test with the full data set clearly reveals.

This is a clearly problematic finding regarding parallel I/O efficiency, especially when contrasted with the results obtained with the same statistical engine but with synthetic data, generated on each core and thus in absence of any I/O consideration, as shown for instance in Figure 10 with up to 64 processes: near-optimal strong scaling is maintained past 64 cores whereas by that point, when parallel I/O calls to access a real data set were included, performance degradation was already obvious.

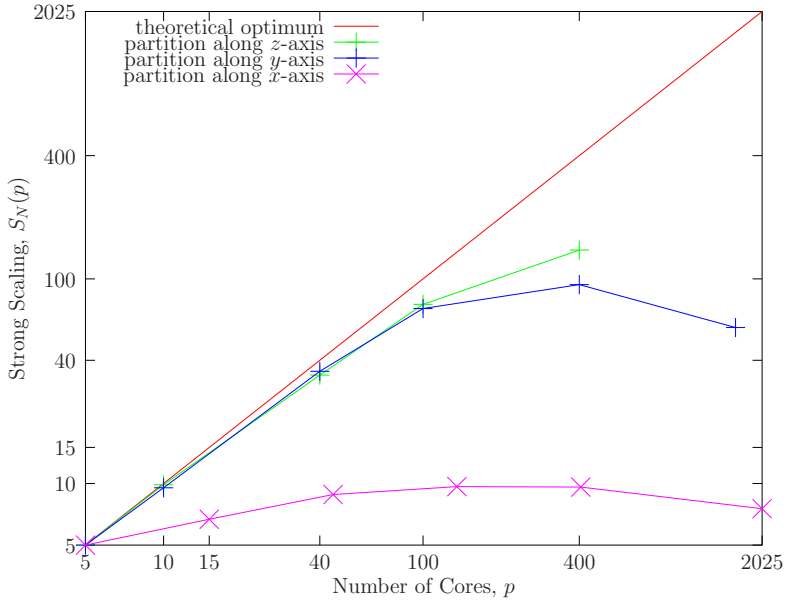


**Figure 9.** Parallel descriptive statistics: strong scaling (at constant total work) on `Jaguar-pf`, 2025x1600x400 case, including parallel I/O calls.

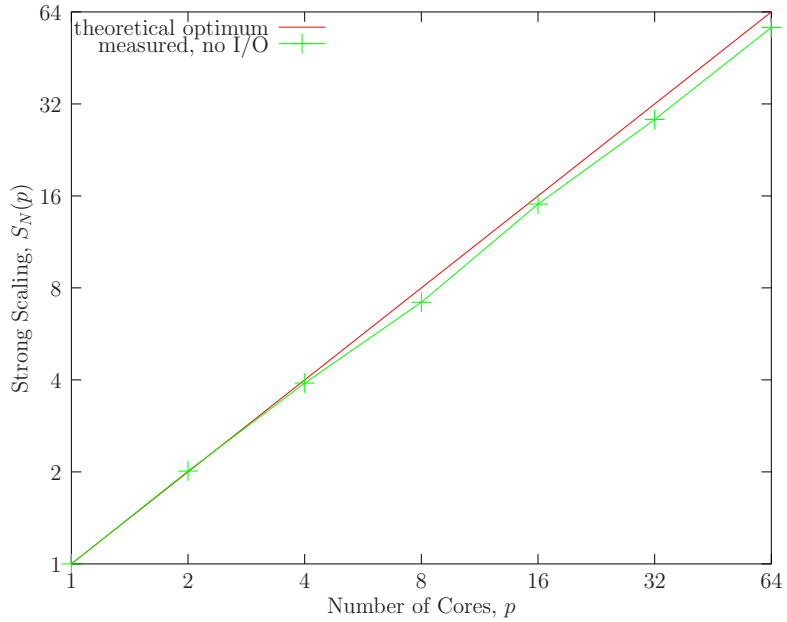


**Figure 10.** Parallel descriptive statistics: strong scaling (at constant total work), synthetic data of size $1.024 \times 10^8$, no I/O calls.

# 4 Conclusion and Perspectives

In this report we have recalled the foundations of the parallel statistical analysis tool kit which we have designed and implemented, with the specific double intent of reproducing typical data analysis workflows, and achieving optimal design for scalable parallel implementations. We have briefly reviewed those earlier results and publications which allow us to conclude that we have achieved both goals.

However, in this report we have further established that, when used in conjuction with a state-of-the-art parallel I/O system, as can be found on the premier DOE peta-scale platform, the scaling properties of the overall analysis pipeline comprising parallel data access routines degrade rapidly. This finding is problematic and must be addressed if peta-scale data analysis is to be made scalable, or even possible.

In order to attempt to address these parallel I/O limitations, we will investigate the use the Adaptable IO System (ADIOS) [LZL+10] to improve I/O performance, while maintaining flexibility for a variety of IO options, such MPI IO, POSIX IO. This system is developed at ORNL and other collaborating institutions, and is being tested extensively on `Jaguar-pf`. Simulation code being developed on these systems will also use ADIOS to output the data thereby making it easier for other systems, such as ours, to process that data.

# References


[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, U.S.A., 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.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, December 2004.

[Jag] Jaguar. http://www.nccs.gov/computing-resources/jaguar.

[JB87] Carlos M. Jarque and Anil K. Bera. A test for normality of observations and regression residuals. *Revue Internationale de Statistique*, 55(2):163–172, 1987.

[LZL+10] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf. Managing variability in the io performance of petascale storage systems. In *SC'10: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, New York, NY, U.S.A., 2010. ACM.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. *Proc. of the National Institute of Science of India*, (12):49–55, 1936.

[PÍ0] P. Pébay. Statistics. In *The VTK User's Guide, version 5.4*, pages 192–198. Kitware, Inc., eleventh edition, 2010.

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

[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. 25$^{th}$ IEEE International Parallel & Distributed Processing Symposium, 12$^{th}$ International Workshop on Parallel and Distributed Scientific and Engineering Computing*, Anchorage, AK, U.S.A., May 2011.

[R] The R project for statistical computing. http://www.r-project.org/.

[tit] Titan informatics toolkit. http://titan.sandia.gov/.

[YRSC09] C. S. Yoo, E. S. Richardson, R. Sankaran, and J. H. Chen. A DNS study of the stabilization mechanism of a turbulent lifted ethylene jet flame in highly-heated coflow. *submitted to Proc. Combust. Inst.*, 2009.

# DISTRIBUTION:

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

Sandia National Laboratories