VisWeek 09
VIS • INFOVIS • VAST

ParaView

Statistics Engines

Philippe Pébay
David Thompson
Nathan Fabian
Diana Roe
Janine Bennett

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-AC04-94-AL85000

Outline

- Motivation
- Terminology
- Class Hierarchy; Available Engines
- Python Example (Serial)
- C++ Example (Parallel)
- OverView Example

Motivation

Motivation

- Want to add statistics functionalities to VTK/ ParaView to allow for better analysis of simulation results
- Want to mimic typical analyst's approach (learn, derive, assess) in a flexible way
- Want parallel versions of these engines which scale as well as theoretically possible (in particular, linearly for several engines)

Terminology

- Observation:
  - Simultaneous measurement of all columns
  - One row of a table is one observation
- Statistic:
  - Image of a function on a set of observations

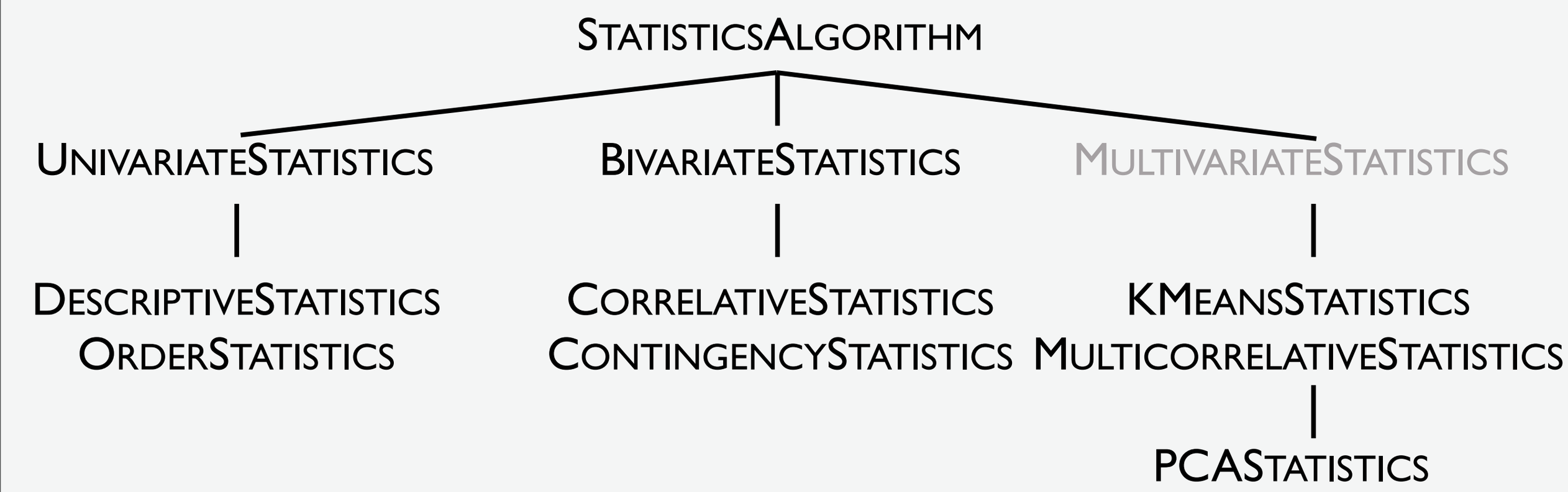
Terminology: Phases

- *Learn*:
  - Computes raw, *minimal* summary statistics
  - Parallel classes add a reduce operation
- *Derive*:
  - Computes values of interest from learned data
  - All to date trivial, so replicated on all processes
- *Assess*:
  - Derived model used to annotate observations
  - May show error, new projection, cluster center, ...

Statistics Filters: I/O

- First input is a table of observations (raw data)
- Second input (optional) is for model/summary
  - May be table or
  - Multiblock of tables
- First output is a table of *possibly*-annotated data
  - How well does it fit the model? or
  - What is its value using the new model?
- Second output is for model/summary (table/MB)

Class Hierarchy



- Univariate: AddColumn, RemoveColumn, SetColumnStatus
- Bivariate: AddColumnPair, RemoveColumnPair, SetColumnStatus (all pairs added)
- Multivariate: SetColumnStatus, RequestSelectedColumns

Descriptive

- Learn: Compute extrema, means, M2, M3 and M4 aggregates
- Derive: Calculate unbiased variance estimator, standard deviation, skewness (g1 and G1), kurtosis (g2 and G2)
- Assess:
  - Calculate 1-D Mahalanobis (relative deviation)
  - One column per request

Order

- Learn: Calculate histogram
- Derive: Calculate arbitrary quartiles (e.g., “5”-point statistics, deciles, percentiles, etc.)
- Assess:
  - Mark with quartile index
  - One column per request

Correlative

- Learn: Compute means and M2 aggregates
- Derive:
  - Compute covariance matrix
    - Compute linear regressions and Pearson r
- Assess:
  - Calculate 2-D Mahalanobis distance from model of each observation
  - One column per request

Contingency

- Learn: Compute contingency table
- Derive:
  - Calculate joint, conditional, marginal PDFs and information entropies.
  - Calculate pointwise mutual information
- Assess:
  - Mark with joint and conditional PDFs and PMIs.
  - One column per request

Multicorrelative

- Learn: Compute means & M2 aggregates
- Derive:
  - Assemble covariance matrix
  - Compute Cholesky decomposition
- Assess:
  - Calculate n-D Mahalanobis distance from model of each observation
  - One column per request

PCA

- Learn: Multicorrelative (covariances)
- Derive:
  - Multicorrelative plus computes eigenvals/vectors of covariance matrix
  - Optional normalization before SVD
- Assess:
  - Project to new basis (eigenvectors)
  - May project to fixed subspace or fixed energy representation

K-Means

- Learn: Calculate new cluster centers using initial cluster centers
- Derive:
  - Calculate global and local rankings amongst sets of clusters.
  - Report total error.
- Assess:
  - Mark with closest cluster ID and associated distance for each set of cluster centers.

Example, Python (serial PCA)

Python Example

```
from vtk import *
exr = vtkExodusIIReader()
exr.SetFileName( '/PV/Data/disk_out_ref.ex2' )
exr.SetPointResultArrayStatus( 'Pres', 1 )
exr.SetPointResultArrayStatus( 'Temp', 1 )
exr.SetPointResultArrayStatus( 'AsH3', 1 )
exr.SetPointResultArrayStatus( 'GaMe3', 1 )
exr.SetPointResultArrayStatus( 'CH4', 1 )
exr.SetPointResultArrayStatus( 'H2', 1 )
exr.Update()
eblks = exr.GetOutput().GetBlock(0)
eblk = eblks.GetBlock(0)
```

Python Example

```
dot = vtkDataObjectToTable()
dot.SetInput( eblk )
dot.Update()
tab = dot.GetOutput()
pca = vtkPCAStatistics()
pca.SetInput(tab)
pca.SetColumnStatus( 'Pres', 1 )
pca.SetColumnStatus( 'Temp', 1 )
pca.SetColumnStatus( 'CH4', 1 )
pca.RequestSelectedColumns()
```

Python Example

```
pca.SetColumnStatus( 'Pres', 0 )
pca.SetColumnStatus( 'Temp', 0 )
pca.SetColumnStatus( 'AsH3', 1 )
pca.SetColumnStatus( 'GaMe3', 1 )
pca.SetColumnStatus( 'H2', 1 )
pca.RequestSelectedColumns()
pca.Update()
covs = pca.GetOutputDataObject(1).GetBlock(0)
req1 = pca.GetOutputDataObject(1).GetBlock(1)
req2 = pca.GetOutputDataObject(1).GetBlock(2)
```

Python Example

`req2.Dump(12)`

Column	Mean	CH4	Pres	Temp
CH4	0.000515	1.71E-07	-6.70E-07	0.03359
Pres	0.020881	0.000413	4.77E-05	0.42490
Temp	425.2	-0.001623	0.006713	29040
Cholesky	8499	81.2963	82.9427	124.704
PCA 0	29040	-1.16E-06	-1.46E-05	-1
PCA 1	4.15E-05	0.02807	-0.9996	1.46E-05
PCA 2	9.92E-08	-0.9996	-0.02807	1.57E-06

Column means
Covariance matrix
Cholesky decomposition
Number of samples
PCA Eigenvalues
PCA Eigenvectors (rows)

Note that without normalization, temperature dominates the results.

Example, C++ (parallel descriptive & multicorrelative)

C++ Example

```
void Foo( vtkMultiProcessController* controller, void* arg )
{
  vtkMultiProcessController::SetGlobalController( controller );
  // Assume the input dataset is passed to us:
  vtkTable* inputData = static_cast<vtkTable*>( arg );
  vtkPDescriptiveStatistics* pds = vtkPDescriptiveStatistics::New();
  // Set input data port
  pds->SetInput( 0, inputData );
  // Select all columns in inputData
  for ( int c = 0; c < inputData->GetNumberOfColumns(); ++ c )
    {
    pds->AddColumn( inputData->GetColumnName[c] );
    }
  // Calculate statistics with Learn and Derive phases only
  pds->SetLearn( true );
  pds->SetDerive( true );
  pds->SetAssess( false );
  pds->Update();
}
```

C++ Example

```
vtkPMultiCorrelativeStatistics* pms
  = vtkPMultiCorrelativeStatistics::New();

  // Turn on columns of interest
  pms->SetColumnStatus( "A", 1 );
  pms->SetColumnStatus( "B", 1 );
  pms->SetColumnStatus( "C", 1 );
  pms->RequestSelectedColumns();

  // Columns A, B, and C are still selected, so first we turn off
  // column A so it will not appear in the next request.
  pms->SetColumnStatus( "A", 0 );
  pms->SetColumnStatus( "D", 1 );
  pms->RequestSelectedColumns();
```

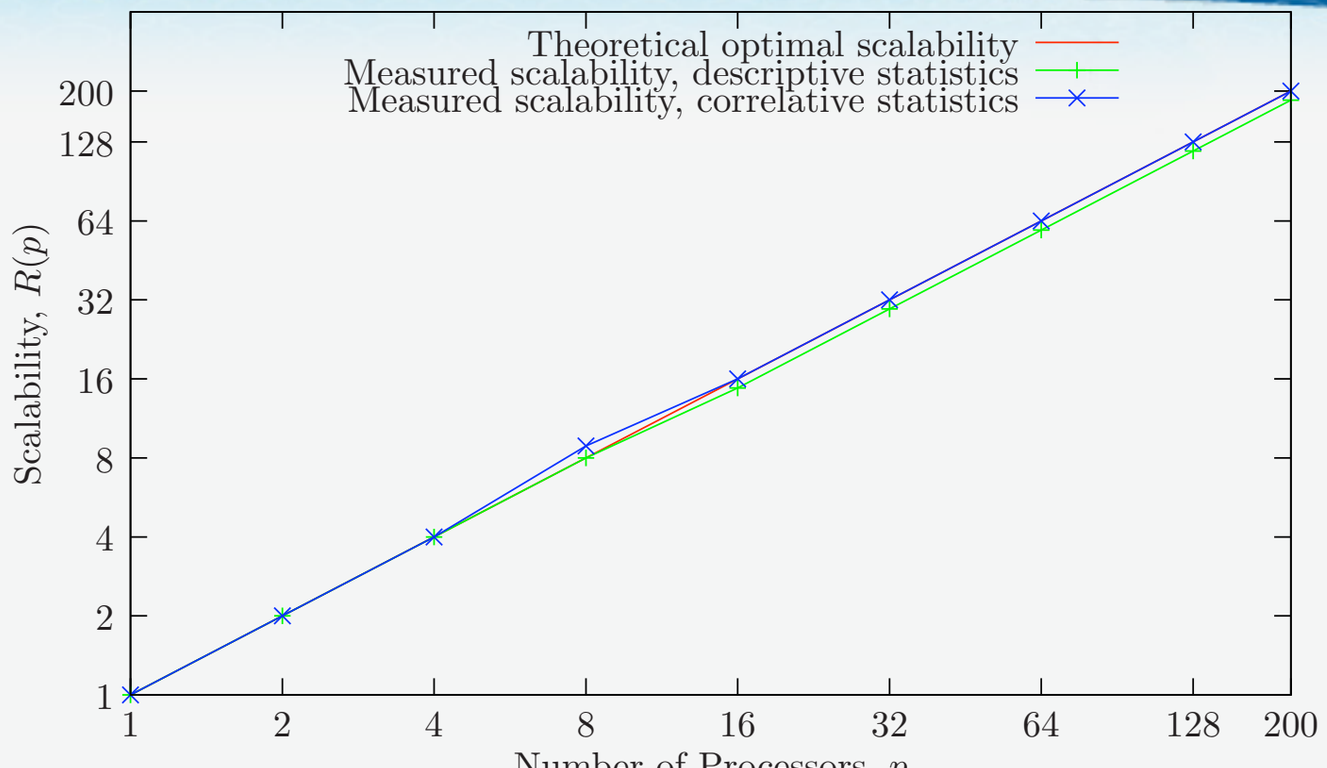
C++ Example

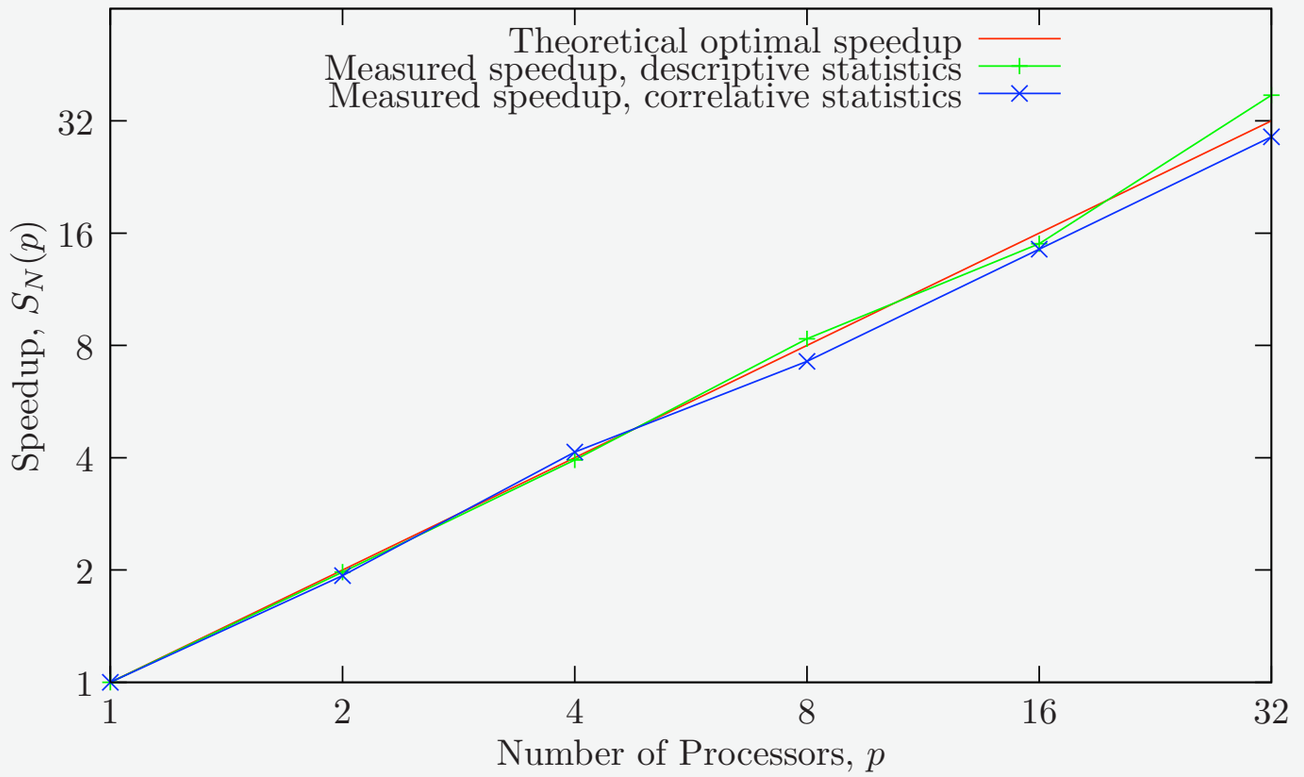
```
vtkTable* inputData;
  vtkMPIController* controller = vtkMPIController::New();
  controller->Initialize( &argc, &argv );

  // Execute the function named Foo on all processes
  controller->SetSingleMethod( Foo, &inputData );
  controller->SingleMethodExecute();

  // Clean up
  controller->Finalize();
  controller->Delete();
```

C++ Example


Theoretical optimal scalability
Measured scalability, descriptive statistics
Measured scalability, correlative statistics
Scalability, $R(p)$
200
128
64
32
16
8
4
2
1
1
2
4
8
16
32
64
128
200
Number of Processors, p


Theoretical optimal speedup
Measured speedup, descriptive statistics
Measured speedup, correlative statistics
Speedup, $S_N(p)$
32
16
8
4
2
1
1
2
4
8
16
32
Number of Processors, p

OverView Example