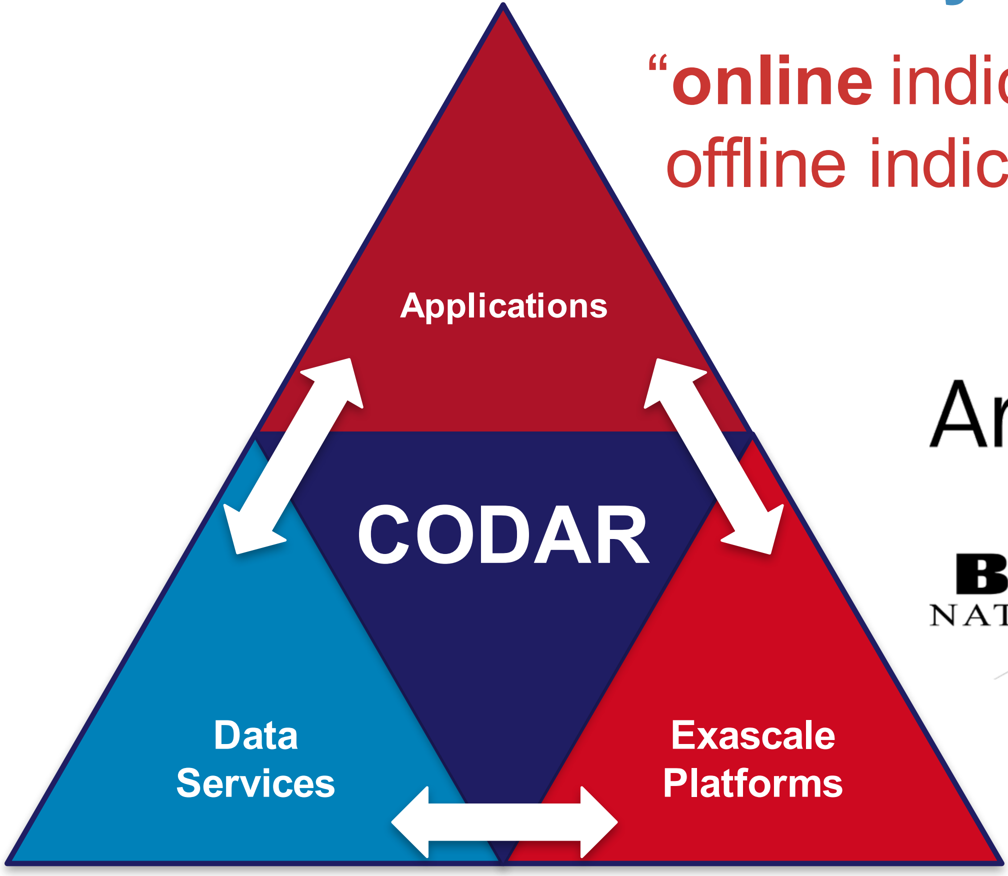
CODAR: Online Data Analysis and Reduction

“**online** indicates a state of connectivity … offline indicates a disconnected state” [wikipedia]



Argonne NATIONAL LABORATORY

BROOKHAVEN NATIONAL LABORATORY

OAK RIDGE National Laboratory

BROWN UNIVERSITY

UNIVERSITY OF OREGON

RUTGERS THE STATE UNIVERSITY OF NEW JERSEY

Stony Brook University

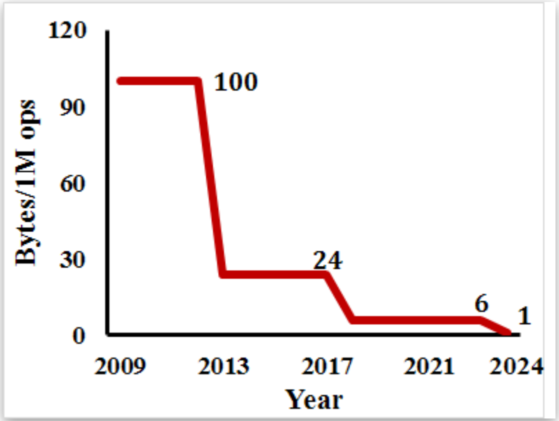
NNSA National Nuclear Security Administration

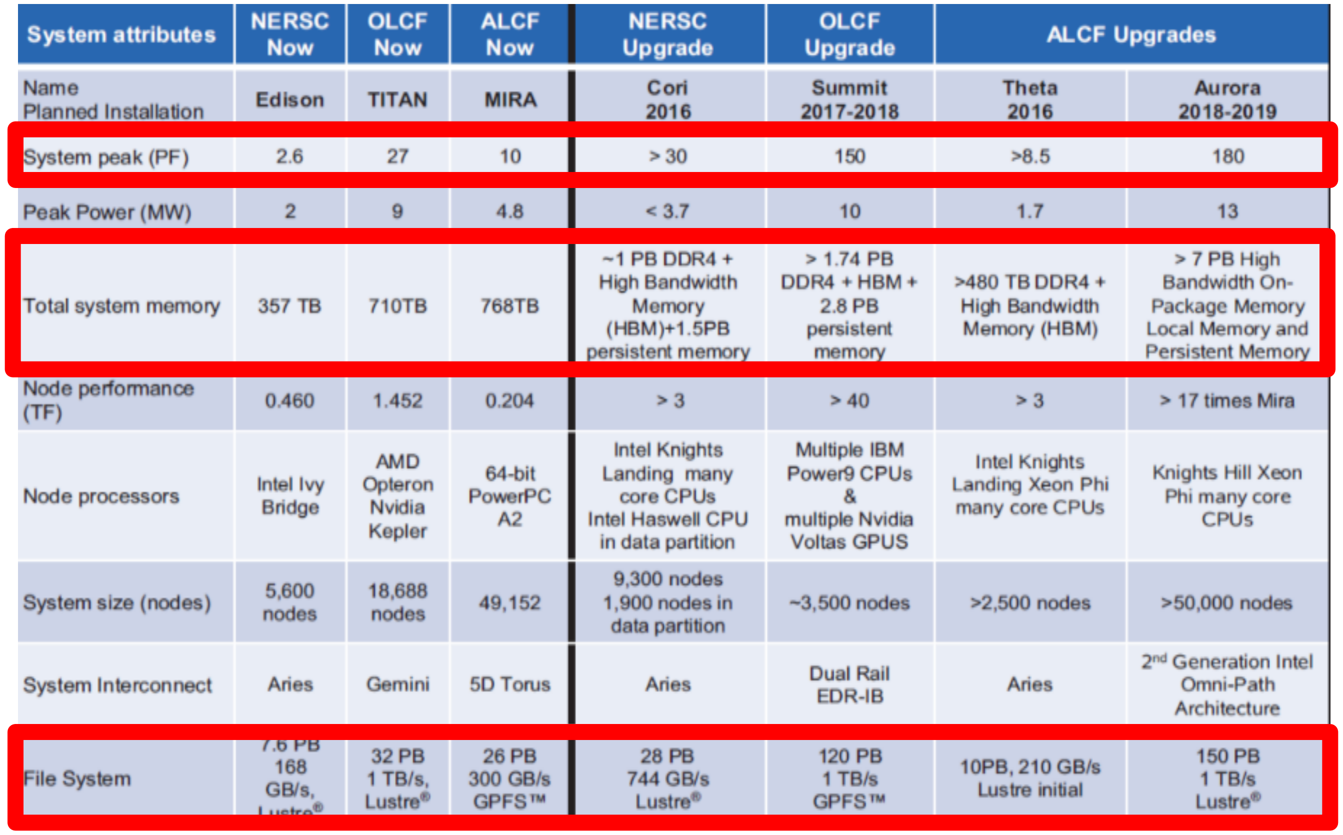
U.S. DEPARTMENT OF ENERGY | Office of Science

ECP EXASCALE COMPUTING PROJECT

Motivation: The widening compute-data gap

Filesystem/network bandwidth falls behind CPU/memory: Fewer bytes/operation





System attributes	NERSC Now	OLCF Now	ALCF Now	NERSC Upgrade	OLCF Upgrade	ALCF Upgrades	
Name Planned Installation	Edison	TITAN	MIRA	Cori 2016	Summit 2017-2018	Theta 2016	Aurora 2018-2019
System peak (PF)	2.6	27	10	> 30	150	>8.5	180
Peak Power (MW)	2	9	4.8	< 3.7	10	1.7	13
Total system memory	357 TB	710TB	768TB	~1 PB DDR4 + High Bandwidth Memory (HBM)+1.5PB persistent memory	> 1.74 PB DDR4 + HBM + 2.8 PB persistent memory	>480 TB DDR4 + High Bandwidth Memory (HBM)	> 7 PB High Bandwidth On-Package Memory Local Memory and Persistent Memory
Node performance (TF)	0.460	1.452	0.204	> 3	> 40	> 3	> 17 times Mira
Node processors	Intel Ivy Bridge	AMD Opteron Nvidia Kepler	64-bit PowerPC A2	Intel Knights Landing many core CPUs Intel Haswell CPU in data partition	Multiple IBM Power9 CPUs & multiple Nvidia Voltas GPUS	Intel Knights Landing Xeon Phi many core CPUs	Knights Hill Xeon Phi many core CPUs
System size (nodes)	5,600 nodes	18,688 nodes	49,152	9,300 nodes 1,900 nodes in data partition	~3,500 nodes	>2,500 nodes	>50,000 nodes
System Interconnect	Aries	Gemini	5D Torus	Aries	Dual Rail EDR-IB	Aries	2nd Generation Intel Omni-Path Architecture
File System	7.6 PB 168 GB/s, Lustre®	32 PB 1 TB/s, Lustre®	26 PB 300 GB/s GPFS™	28 PB 744 GB/s Lustre®	120 PB 1 TB/s GPFS™	10PB, 210 GB/s Lustre initial	150 PB 1 TB/s Lustre®

Couple application tasks **online** rather than via file system, to:

1) Reduce I/O costs
2) Reduce storage needs
3) Enable online response

Summit: 150 PF/sec, 1 TB/s to disk→ One million FLOPS per **double**
Exascale: 1 EF/sec, ~1 TB/s to disk → Ten million FLOPS per **double**!

Thus, our key motif: Online data analysis and reduction

Traditional approach:
Compute…output…analyze [offline]

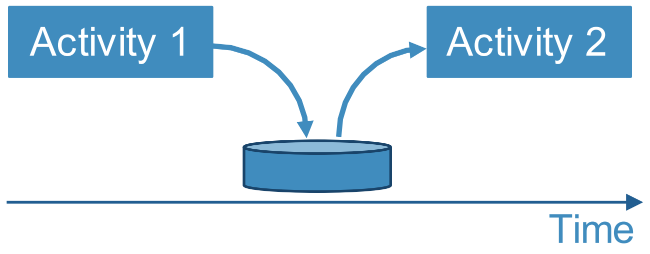
Write simulation output to secondary storage; read back for analysis

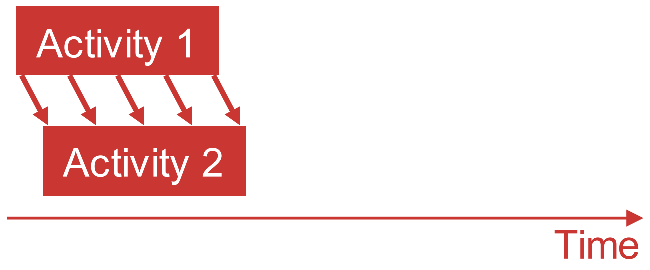
Decimate in time when simulation output rate exceeds output rate of computer

New approach:
Online data analysis and reduction

Co-optimize simulation, analysis, reduction for performance and information output

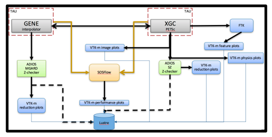
Substitute CPU cycles for I/O, via online data (de)compression and/or online data analysis

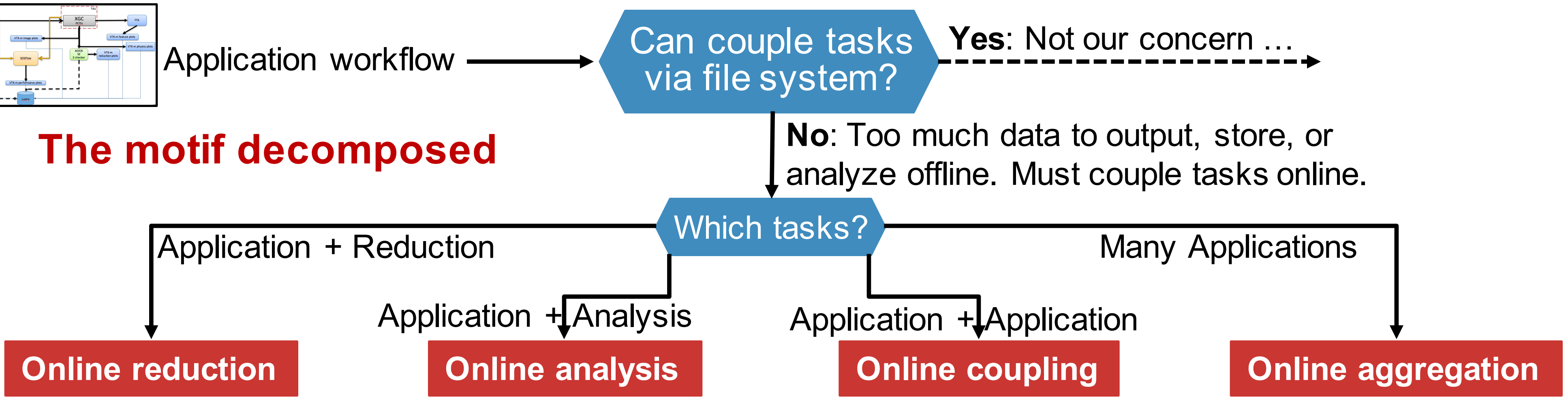


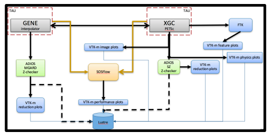


Provide the right information at the right time and place to accelerate discovery!

The motif decomposed




Application workflow
Can couple tasks via file system?
Yes: Not our concern …
No: Too much data to output, store, or analyze offline. Must couple tasks online.
Which tasks?
Application + Reduction
Application + Analysis
Application + Application
Many Applications
Online reduction
Online analysis
Online coupling
Online aggregation



Application workflow → **Can couple tasks via file system?**

Yes: Not our concern …

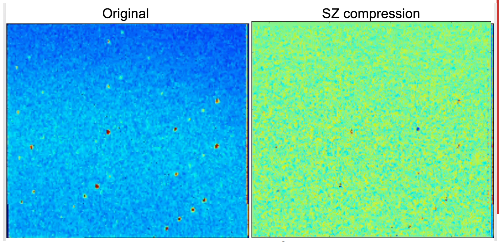
No: Too much data to output, store, or analyze offline. Must couple tasks online.

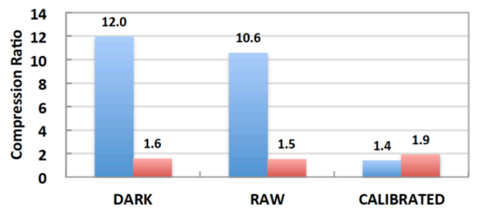
The motif decomposed

Which tasks?

- Application + Reduction → **Online reduction**
- Application + Analysis → **Online analysis**
- Application + Application → **Online coupling**
- Many Applications → **Online aggregation**

Online reduction

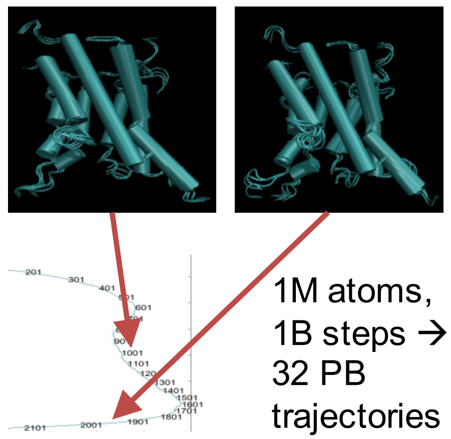




(a) Point-wise relative error bound = 10^{-1}

ExaFEL: X-ray laser imaging

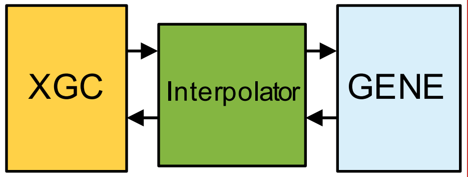
Online analysis

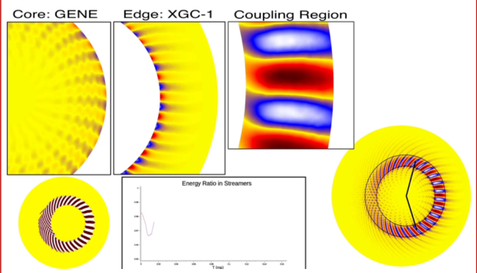


1M atoms, 1B steps → 32 PB trajectories

NWChemEx: Molecular dynamics

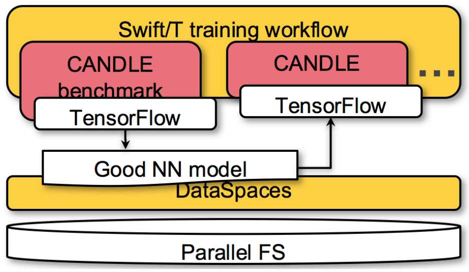
Online coupling





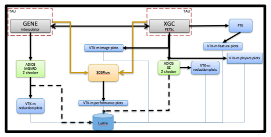
WDMApp: Fusion whole device model

Online aggregation

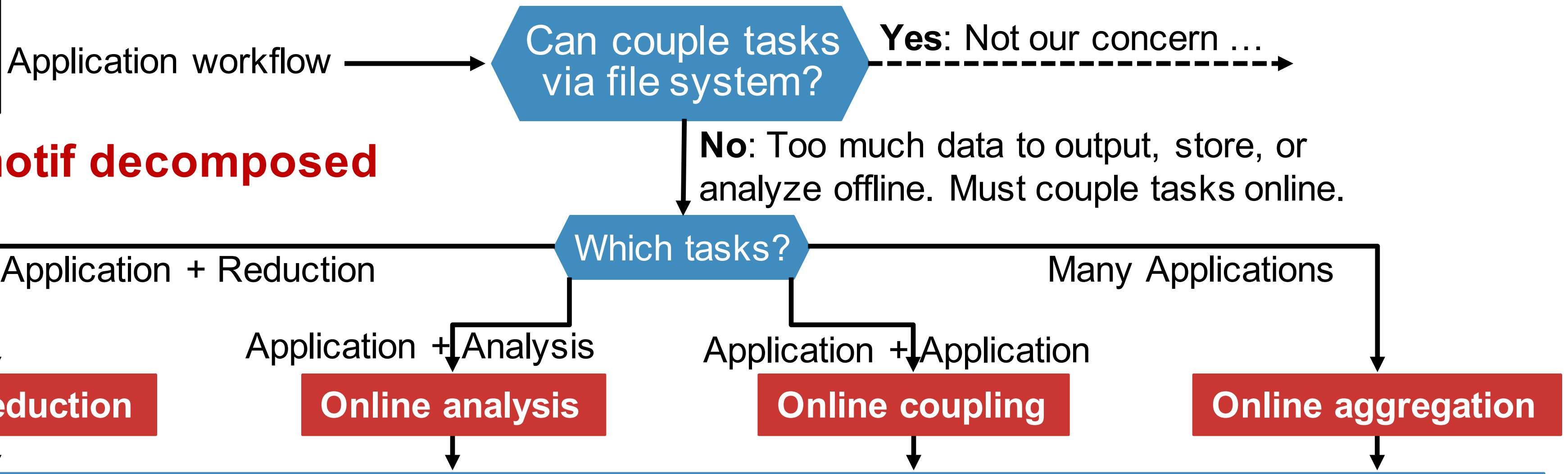


Hyperparam. optimization: 10^3–10^6 training runs, each fitting many parameters

CANDLE: Cancer deep learning



The motif decomposed



Cross-cutting co-design questions

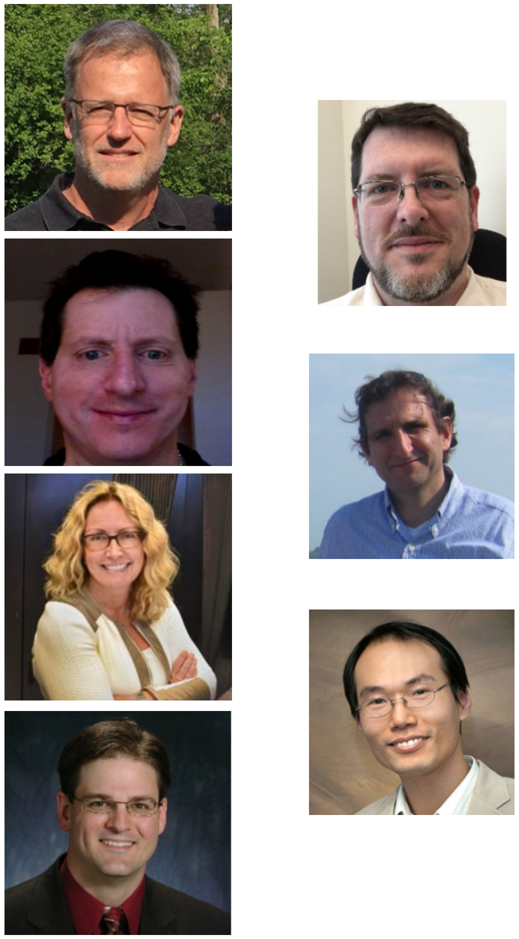
What are the best data analysis and reduction algorithms for different application classes and exascale systems, in terms of speed, accuracy, and resource requirements?

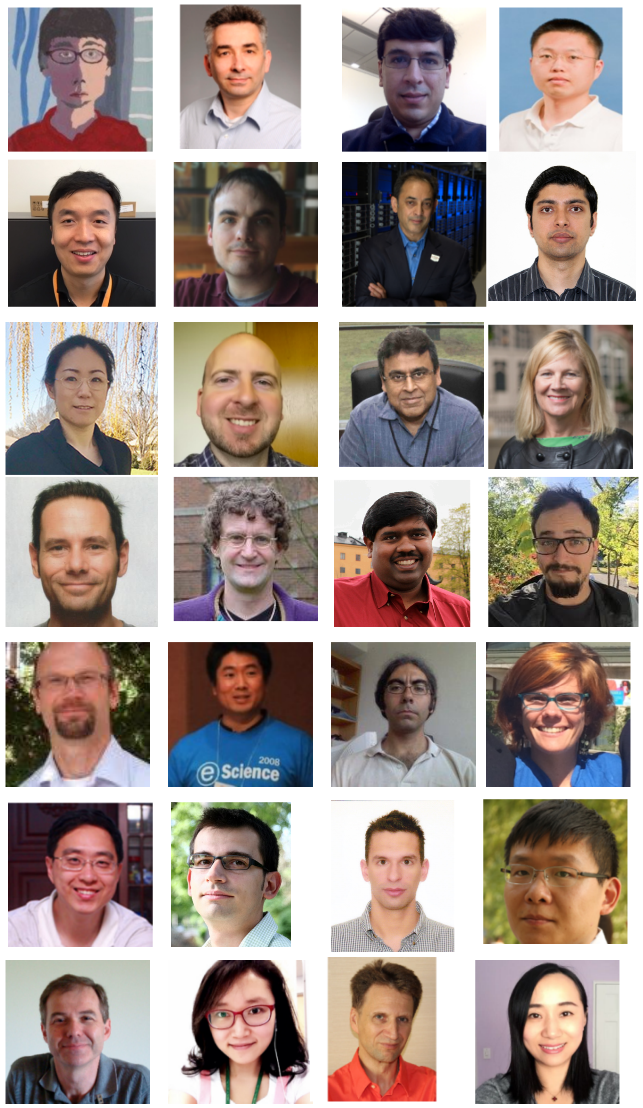
What are tradeoffs in data accuracy, resource needs, and application performance among analysis and reduction methods and for different placement strategies? How do tradeoffs vary with exascale hardware and software choices?

How can we effectively orchestrate online activities to simplify evaluation of tradeoffs and reduce associated overheads? How can exascale hardware and software help with orchestration?

Management – Ian Foster (Lead)

- Scott Klasky
- Kerstin Kleese van Dam
- Todd Munson (Deputy)





Infrastructure – Matthew Wolf (Lead)

- Cheetah: Bryce Allen, Tahsin Kurc, Kshitij Mehta, Li Tang, Lipeng Wan
- Savanna: Philip Davis, Manish Parashar, Pradeep Subedi, Tong Shu, Justin Wozniak
- Chimbuko: Abid Malik, Line Pouchard
- TAU: Kevin Huck, Allen Malony, Sameer Shende, Chad Wood

Data Reduction – Franck Cappello (Lead)

- MGARD: Mark Ainsworth, Jong Choi, Ozan Tugluk
- Z-checker: Julie Bessac, Sheng Di, Emil Constantinescu

Data Analysis – Shinjae Yoo (Lead)

- Anomalies: Gyorgy Matyasfalvi
- FTK: Hanqi Guo, Tom Peterka
- Hierarchical: Wendy Di
- Visual Analytics: Klaus Mueller, Wei Xu

Four topics that laddress here

1. The **software architecture** that factors the computational motif into kernels that can be implemented at the highest possible performance on the exascale architectures. Our strategyy for **addressing performance portability** across multiple exascale architectures
2. How we address **usability across multiple applications**, both the targeted ECP applications and the broader set of potential applications for which this motif appears prominently. The APIs and workflows that facilitate integration into such applications
3. The process by which we **engage with ECP applications stakeholders**

(1a) CODAR architecture enables five things

A. Select **analysis and reduction modules** to meet application needs

B. Configure **application** with multiple computation, analysis, reduction components

C. Perform a **run** with specified number of nodes and component mapping to nodes

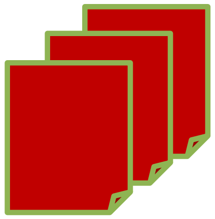
D. Collect **reduction quality** and **application/workflow performance** information

E. Perform a **set of runs**, collecting performance and reduction quality information and analyzing the results

A) Select analysis and reduction modules to meet application needs

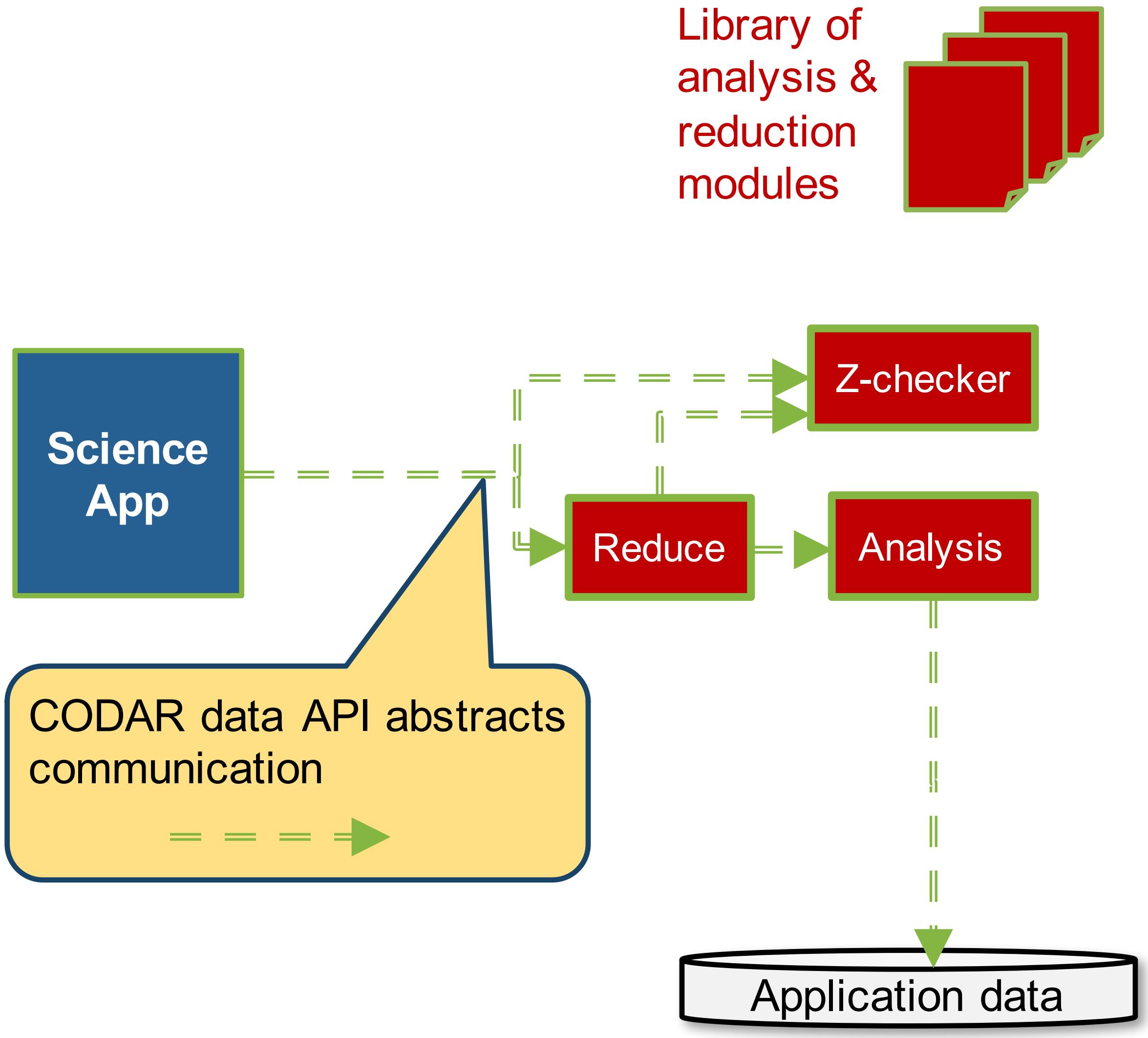
Each module addresses needs of specific class of data

Library of analysis & reduction modules

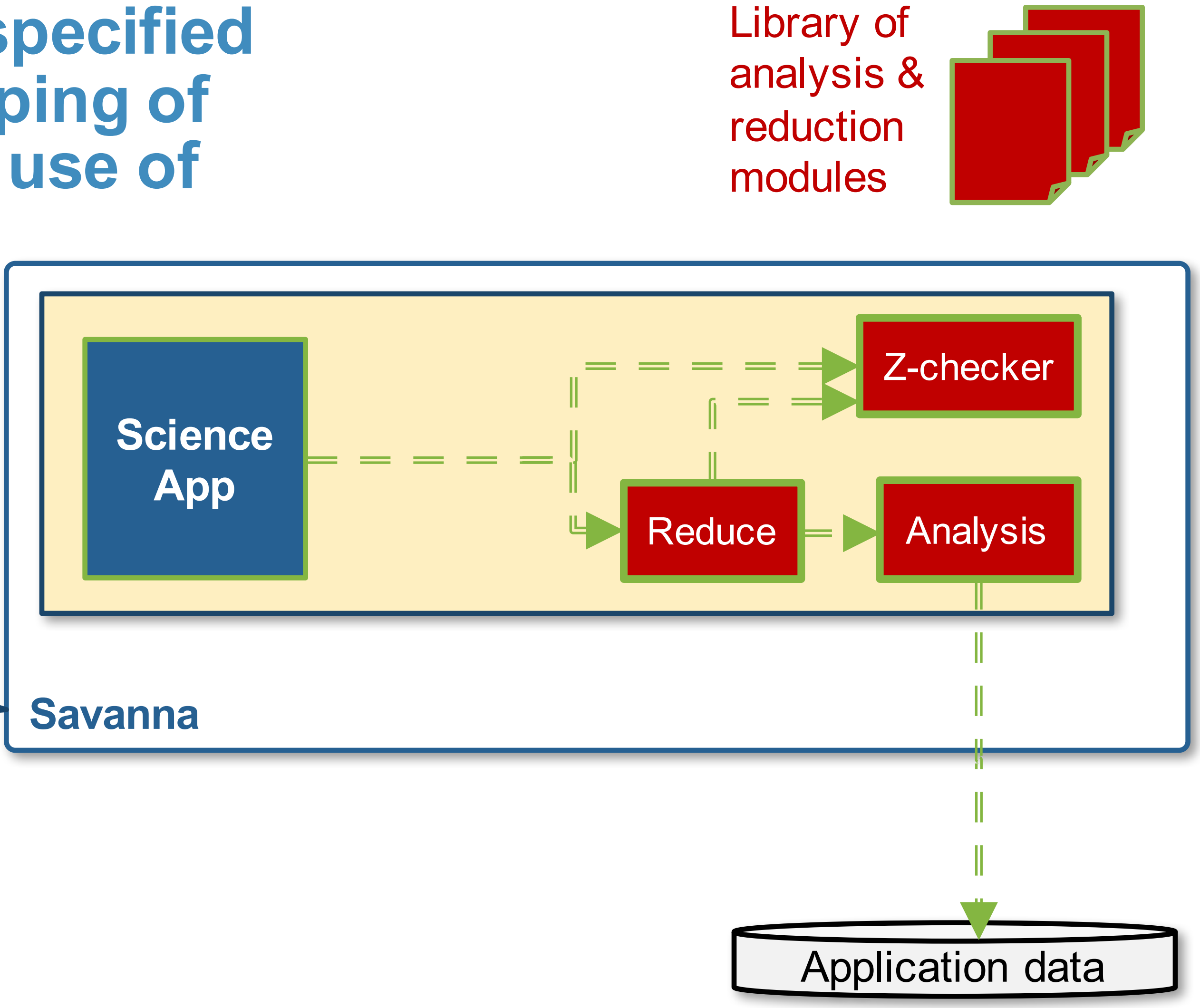


https://github.com/codarcode/TODD (soon)

B) Configure application involving multiple science, analysis, reduction components


Library of analysis & reduction modules
Science App
Z-checker
Reduce
Analysis
CODAR data API abstracts communication
Application data

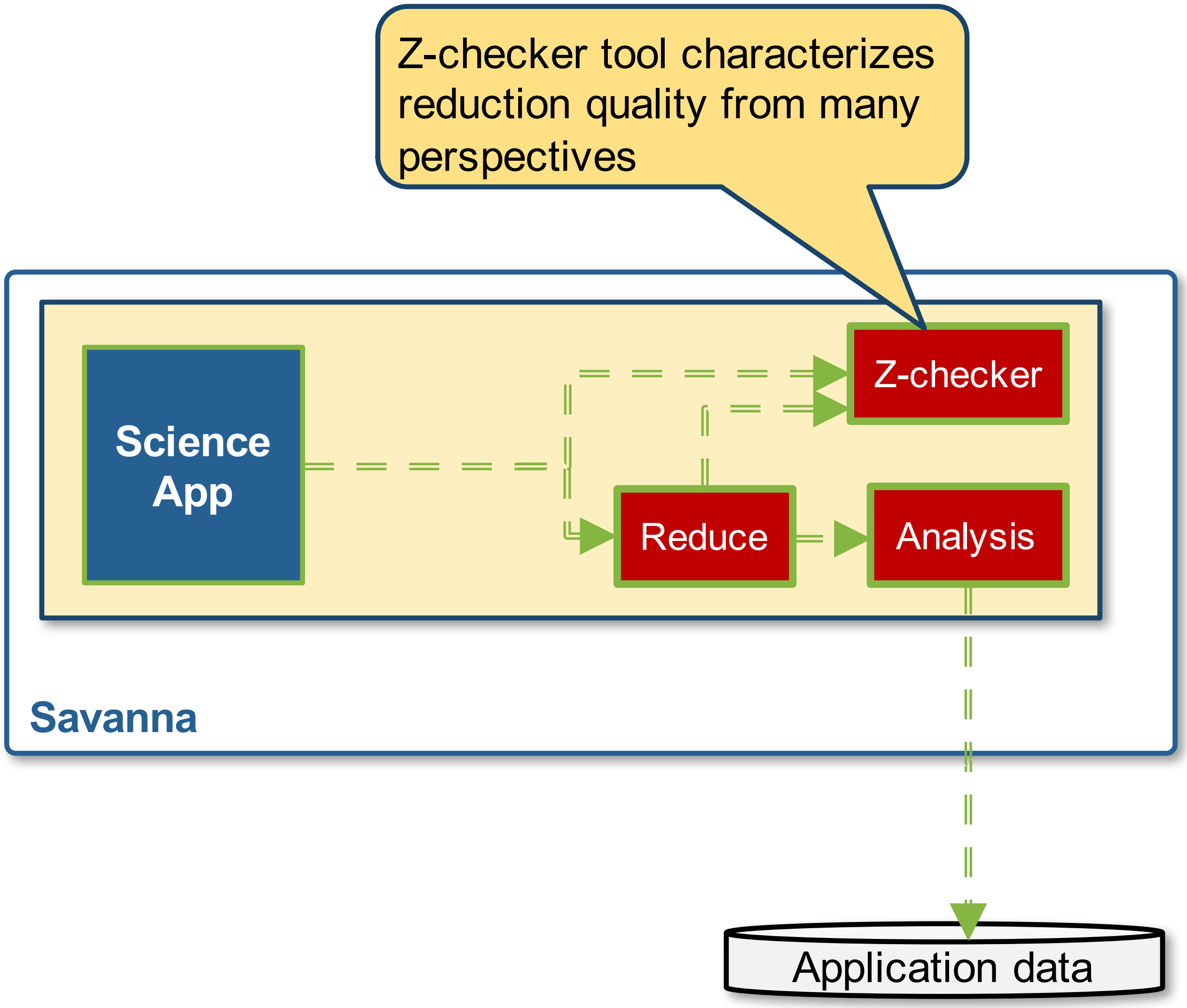
C) Perform a run with specified number of nodes, mapping of components to nodes, use of NVRAM, etc.



Savanna tool manages mapping and computation on different computers and using different mechanisms

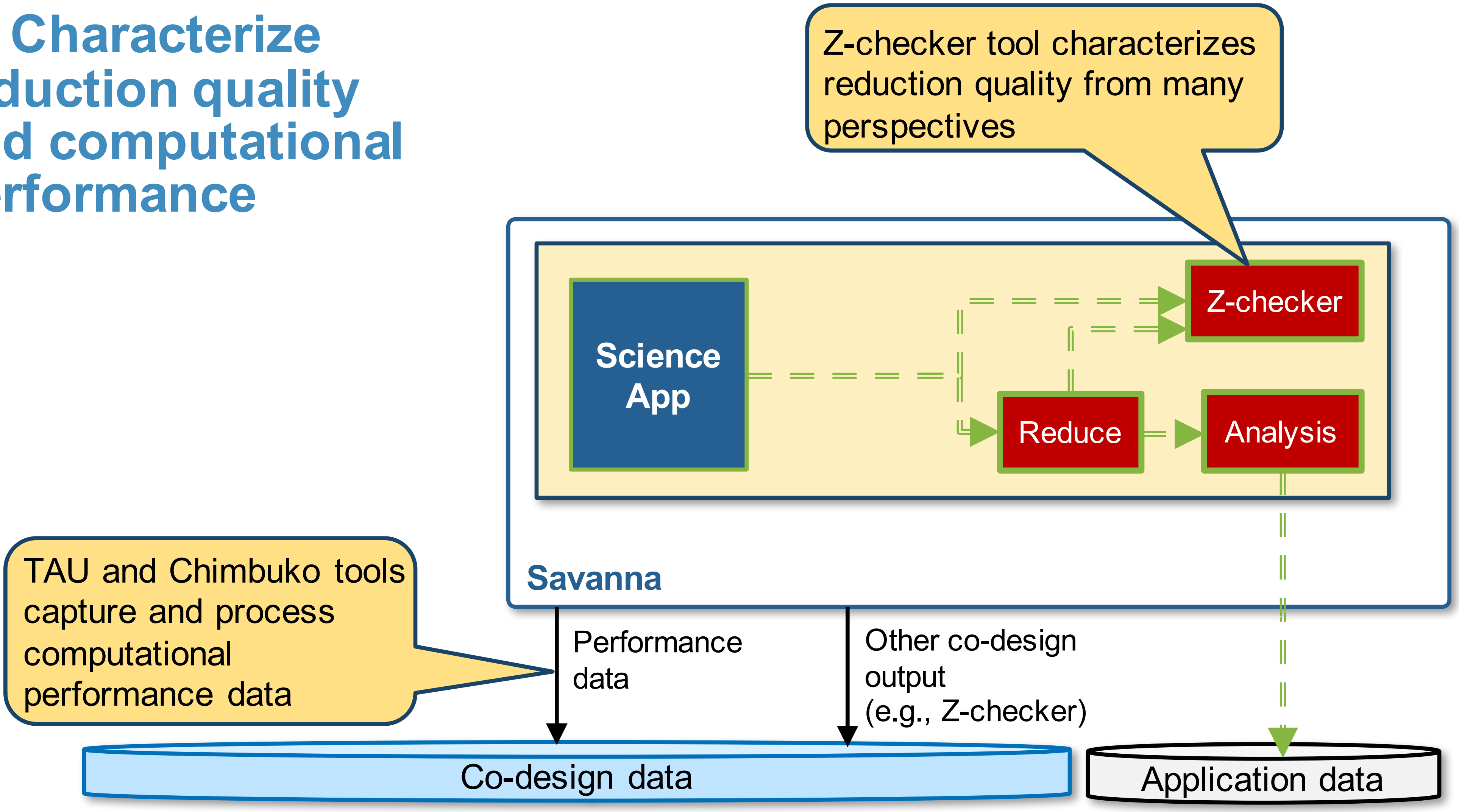
https://github.com/codarcode/savanna

D) Characterize reduction quality


Z-checker tool characterizes reduction quality from many perspectives
Science App
Z-checker
Reduce
Analysis
Savanna
Application data

https://github.com/codarcode/Z-Checker

D) Characterize reduction quality and computational performance


Z-checker tool characterizes reduction quality from many perspectives
Science App
Z-checker
Reduce
Analysis
Savanna
TAU and Chimbuko tools capture and process computational performance data
Performance data
Other co-design output (e.g., Z-checker)
Co-design data
Application data

https://github.com/codarcode/Chimbuko

Scalable Observation System for TAU performance data

Scope & Objectives

- How can ECP application developers get a holistic view of performance data and metadata from all components of a scientific workflow?
- How can ECP application developers monitor high-level performance data from workflows, and control its execution accordingly?
- Scalable Observation System (SOS) integrated with TAU to aggregate performance data for runtime monitoring and control.
- Collaborating within CODAR to integrate with Cheetah, Savanna, Chimbuko for parametric study design and provenance understanding.
- Collaborating with NWChemEx, WDMApp as science drivers.

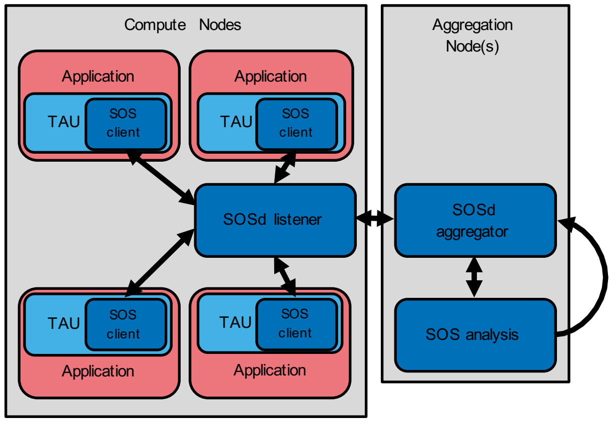


Figure 1: SOS topology in typical TAU deployment.

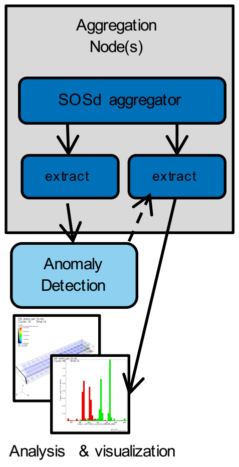


Figure 2: SOS / Chimbuko integration for anomaly detection.

Impact

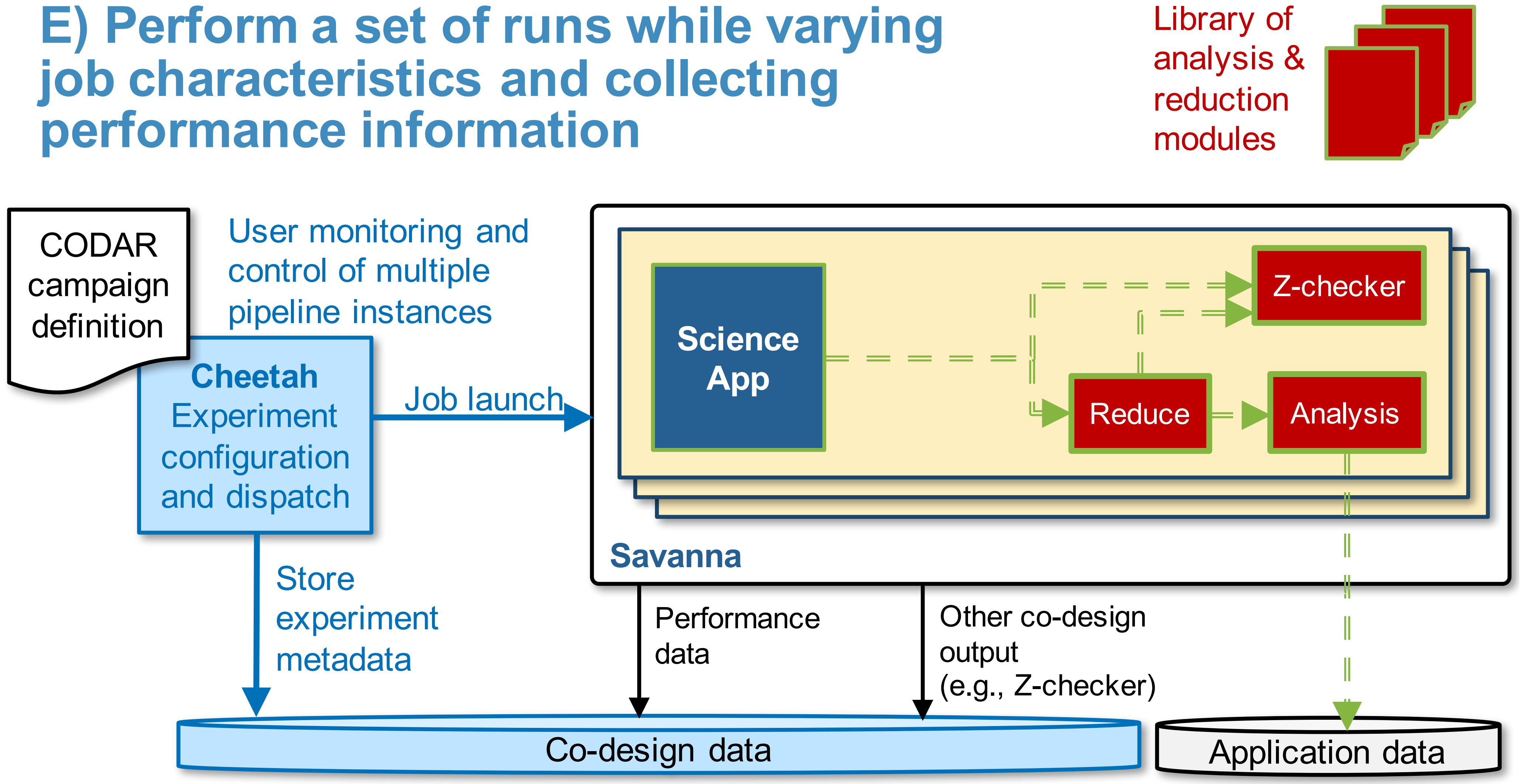
- SOS enables runtime monitoring and control of complex workflows with distributed data sources using data aggregation and/or distributed queries of time-series data.
- Integration with Chimbuko allows for targeted, detailed trace data collection limited to anomalous performance events.
- Exploring distributed autotuning, optimization for runtime frameworks such as RAJA (non-ECP collaboration).

Project Accomplishment

- Integrated TAU and SOSflow with coupled XGC/GENE Fusion application to provide runtime monitoring of high-level metrics such as per-iteration FLOPs and memory consumption (HWM, RSS) across all ranks, visualized in real time.
- Designed and implemented high-speed data caching infrastructure in SOSflow to handle high-volume, large scale data aggregation for Chimbuko.
- Integrated with Chimbuko framework to feed performance anomaly detection analysis and trigger export of preceding trace and provenance for post-mortem visualization and analysis.

Deliverables: Pouchard et al., "Prescriptive Provenance for Streaming Analysis of Workflows at Scale", NYSDS 2018 (to appear); Xie et al., "Performance Visualization for TAU Instrumented Scientific Workflows", IVAPP 2018; SOSflow: https://github.com/cdwdirect/sos_flow; TAU: http://tau.uoregon.edu

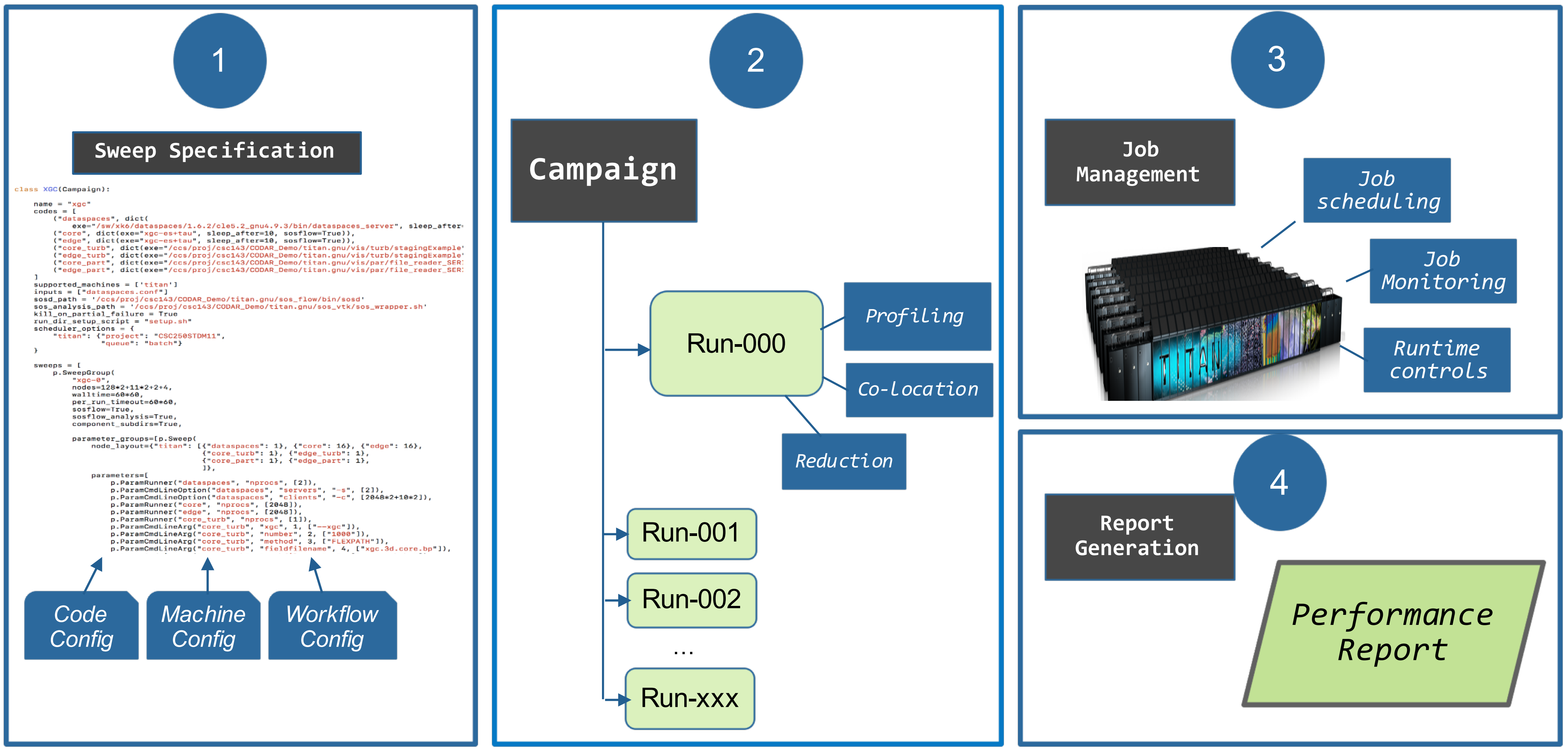
E) Perform a set of runs while varying job characteristics and collecting performance information



Library of analysis & reduction modules
CODAR campaign definition
User monitoring and control of multiple pipeline instances
Cheetah Experiment configuration and dispatch
Job launch
Science App
Z-checker
Reduce
Analysis
Savanna
Store experiment metadata
Performance data
Other co-design output (e.g., Z-checker)
Co-design data
Application data

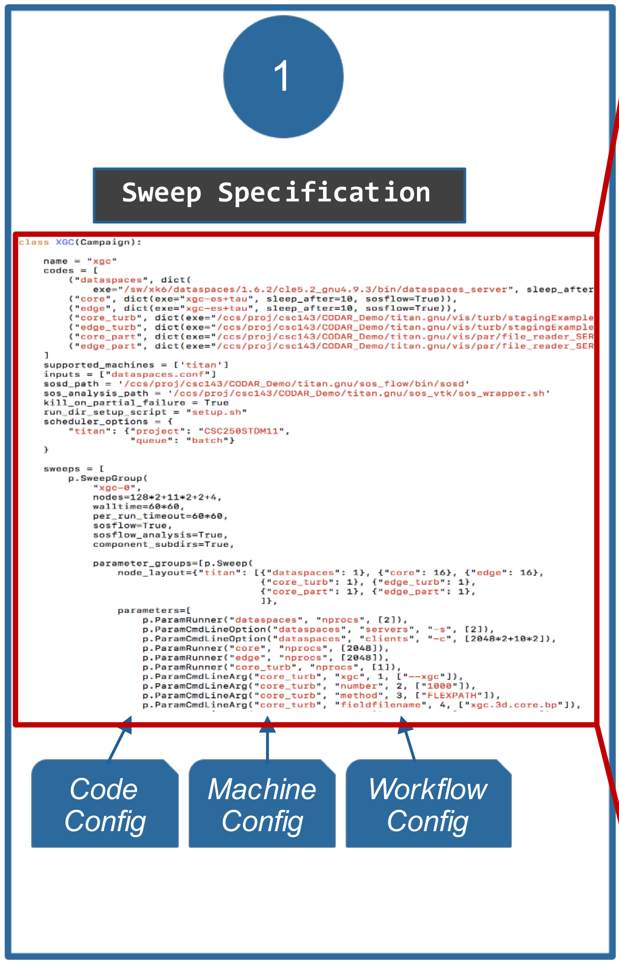
https://github.com/codarcode/cheetah

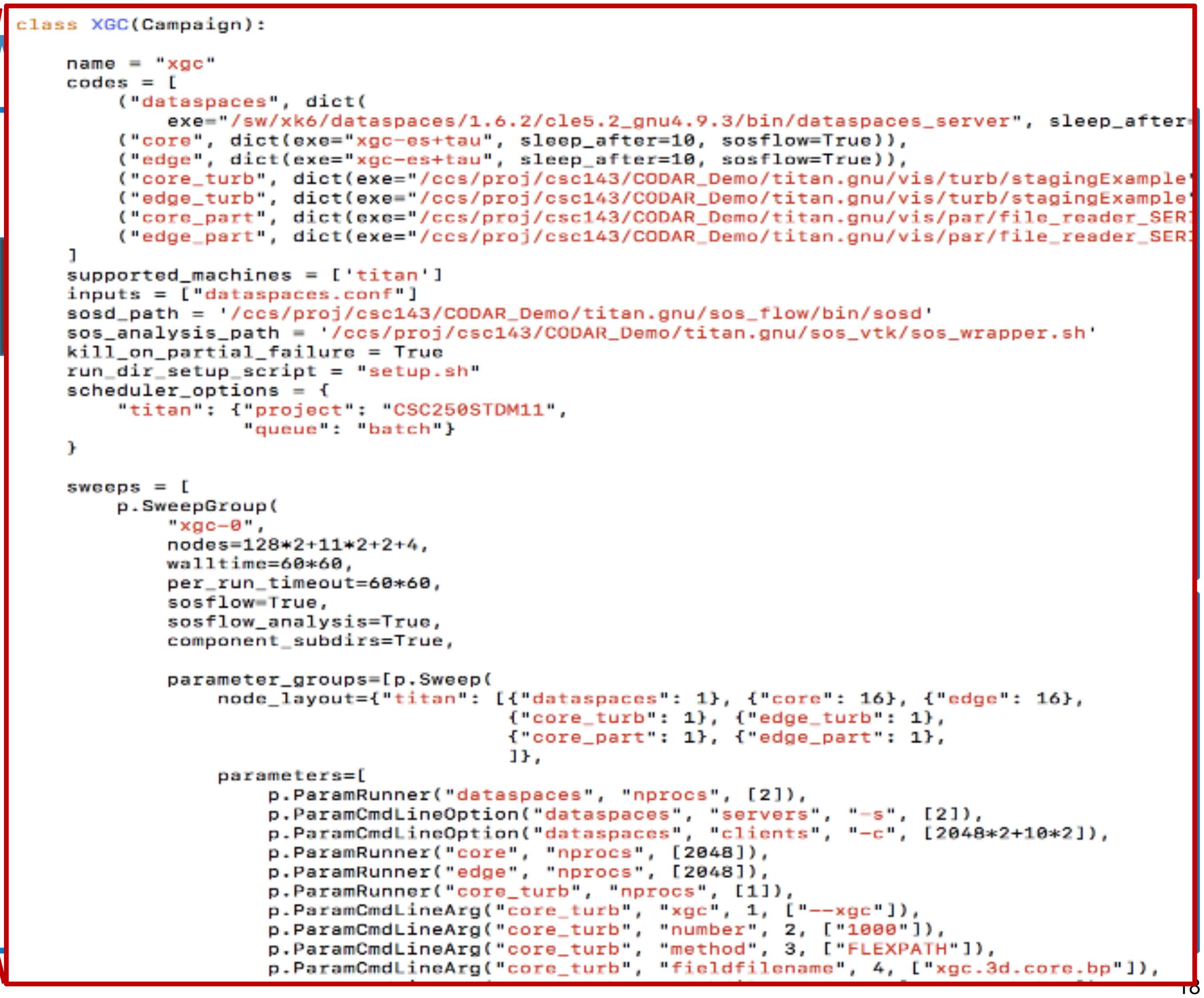
Cheetah example: WDM co-design study



```
class XGC(Campaign):

    name = "xgc"
    codes = [
        ("dataspaces", dict(
            exe="/sw/xk6/dataspaces/1.6.2/cle5.2_gnu4.9.3/bin/dataspaces_server", sleep_after
        ("core", dict(exe="xgc-es+tau", sleep_after=10, sosflow=True)),
        ("edge", dict(exe="xgc-es+tau", sleep_after=10, sosflow=True)),
        ("core_turb", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/turb/stagingExample'
        ("edge_turb", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/turb/stagingExample'
        ("core_part", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/par/file_reader_SER
        ("edge_part", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/par/file_reader_SER
    ]
    supported_machines = ['titan']
    inputs = ["dataspaces.conf"]
    sosd_path = '/ccs/proj/csc143/CODAR_Demo/titan.gnu/sos_flow/bin/sosd'
    sos_analysis_path = '/ccs/proj/csc143/CODAR_Demo/titan.gnu/sos_vtk/sos_wrapper.sh'
    kill_on_partial_failure = True
    run_dir_setup_script = "setup.sh"
    scheduler_options = {
        "titan": {"project": "CSC250STDM11",
                  "queue": "batch"}
    }

    sweeps = [
        p.SweepGroup(
            "xgc-0",
            nodes=128*2+11*2+2+4,
            walltime=60*60,
            per_run_timeout=60*60,
            sosflow=True,
            sosflow_analysis=True,
            component_subdirs=True,

            parameter_groups=[p.Sweep(
                node_layout={"titan": [{"dataspaces": 1}, {"core": 16}, {"edge": 16},
                                       {"core_turb": 1}, {"edge_turb": 1},
                                       {"core_part": 1}, {"edge_part": 1},
                                       ]},
                parameters=[
                    p.ParamRunner("dataspaces", "nprocs", [2]),
                    p.ParamCmdLineOption("dataspaces", "servers", "-s", [2]),
                    p.ParamCmdLineOption("dataspaces", "clients", "-c", [2048*2+10*2]),
                    p.ParamRunner("core", "nprocs", [2048]),
                    p.ParamRunner("edge", "nprocs", [2048]),
                    p.ParamRunner("core_turb", "nprocs", [1]),
                    p.ParamCmdLineArg("core_turb", "xgc", 1, ["--xgc"]),
                    p.ParamCmdLineArg("core_turb", "number", 2, ["1000"]),
                    p.ParamCmdLineArg("core_turb", "method", 3, ["FLEXPATH"]),
                    p.ParamCmdLineArg("core_turb", "fieldfilename", 4, ["xgc.3d.core.bp"]),
```

Cheetah example: V





```
class XGC(Campaign):

    name = "xgc"
    codes = [
        ("dataspaces", dict(
            exe="/sw/xk6/dataspaces/1.6.2/cle5.2_gnu4.9.3/bin/dataspaces_server", sleep_after
        ("core", dict(exe="xgc-es+tau", sleep_after=10, sosflow=True)),
        ("edge", dict(exe="xgc-es+tau", sleep_after=10, sosflow=True)),
        ("core_turb", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/turb/stagingExample
        ("edge_turb", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/turb/stagingExample
        ("core_part", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/par/file_reader_SER
        ("edge_part", dict(exe="/ccs/proj/csc143/CODAR_Demo/titan.gnu/vis/par/file_reader_SER
    ]
    supported_machines = ['titan']
    inputs = ["dataspaces.conf"]
    sosd_path = '/ccs/proj/csc143/CODAR_Demo/titan.gnu/sos_flow/bin/sosd'
    sos_analysis_path = '/ccs/proj/csc143/CODAR_Demo/titan.gnu/sos_vtk/sos_wrapper.sh'
    kill_on_partial_failure = True
    run_dir_setup_script = "setup.sh"
    scheduler_options = {
        "titan": {"project": "CSC250STDM11",
                  "queue": "batch"}
    }

    sweeps = [
        p.SweepGroup(
            "xgc-0",
            nodes=128*2+11*2+2+4,
            walltime=60*60,
            per_run_timeout=60*60,
            sosflow=True,
            sosflow_analysis=True,
            component_subdirs=True,

            parameter_groups=[p.Sweep(
                node_layout={"titan": [{"dataspaces": 1}, {"core": 16}, {"edge": 16},
                                       {"core_turb": 1}, {"edge_turb": 1},
                                       {"core_part": 1}, {"edge_part": 1},
                                       ]},
                parameters=[
                    p.ParamRunner("dataspaces", "nprocs", [2]),
                    p.ParamCmdLineOption("dataspaces", "servers", "-s", [2]),
                    p.ParamCmdLineOption("dataspaces", "clients", "-c", [2048*2+10*2]),
                    p.ParamRunner("core", "nprocs", [2048]),
                    p.ParamRunner("edge", "nprocs", [2048]),
                    p.ParamRunner("core_turb", "nprocs", [1]),
                    p.ParamCmdLineArg("core_turb", "xgc", 1, ["--xgc"]),
                    p.ParamCmdLineArg("core_turb", "number", 2, ["1000"]),
                    p.ParamCmdLineArg("core_turb", "method", 3, ["FLEXPATH"]),
                    p.ParamCmdLineArg("core_turb", "fieldfilename", 4, ["xgc.3d.core.bp"]),
```

TODD: Toolkit for Online Data Discovery

Analysis

- FTK: Feature detection and tracking using machine learning
- Z-checker: Compression quality analyzer
- Performance anomaly detection
- Adaptive sampling for particle trajectories
- Multiple VTK-m and Ascent modules for online analysis and visualization, e.g.:
  - Ray casting
  - Iso-surfaces
  - Others

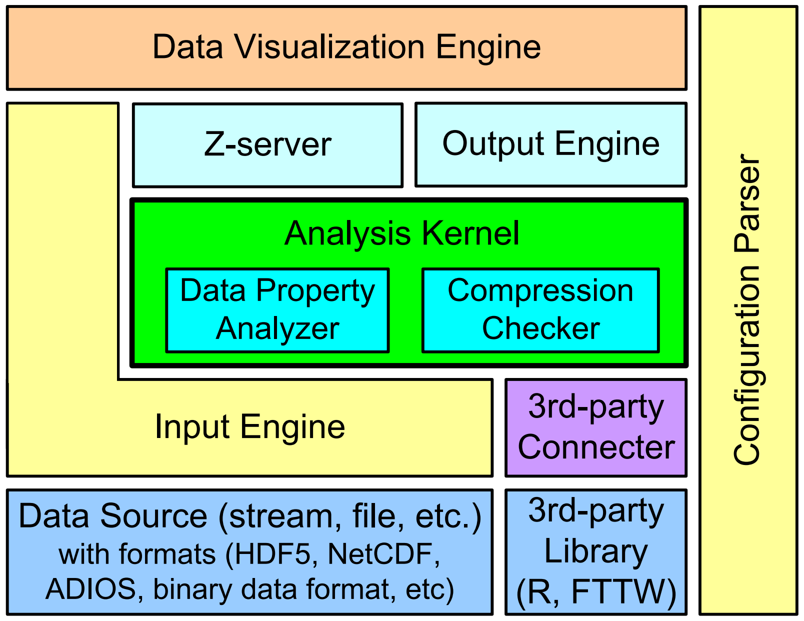
Reduction

- MGARD: Adaptive multilevel compression
- Numerical optimization-based compression
- SZ: Lossy compression
- ZFP: Lossy compression
- Various lossless compression methods

Key:
Developed by CODAR
Developed by others

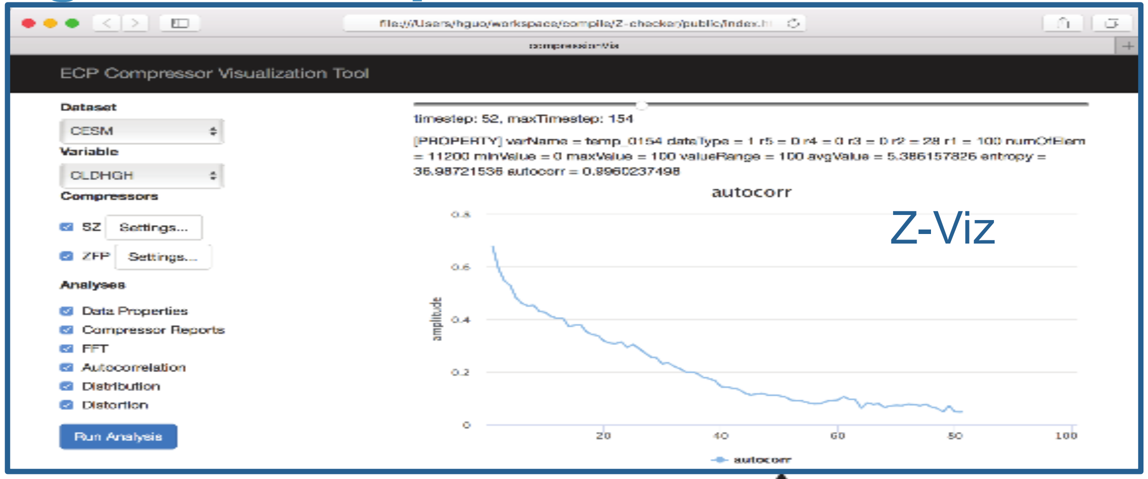
https://github.com/codarcode/TODD (soon)

Z-checker for characterizing reduction performance


Data Visualization Engine
Z-server
Output Engine
Analysis Kernel
Data Property Analyzer
Compression Checker
Input Engine
3rd-party Connecter
Data Source (stream, file, etc.) with formats (HDF5, NetCDF, ADIOS, binary data format, etc)
3rd-party Library (R, FTTW)
Configuration Parser

Z-checker uses **analysis modules** to compare the initial data set and the reduced dataset. The input engine features multiple **reader modules**.

https://github.com/codarcode/Z-checker

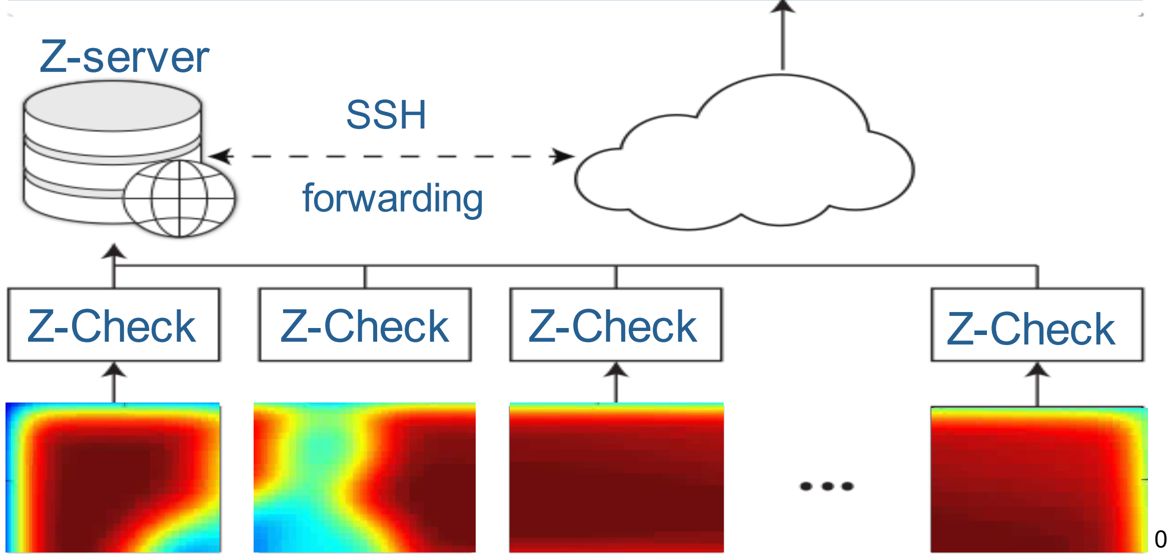

file:///Users/hguo/workspace/compile/Z-checker/public/index.h
compression-Viz
Run Analysis
autocorr
amplitude
Z-Viz

ECP Compressor Visualization Tool

Dataset: CESM
Variable: CLDHGH
Compressors: ☑ SZ Settings... ☑ ZFP Settings...
Analyses: ☑ Data Properties ☑ Compressor Reports ☑ FFT ☑ Autocorrelation ☑ Distribution ☑ Distortion

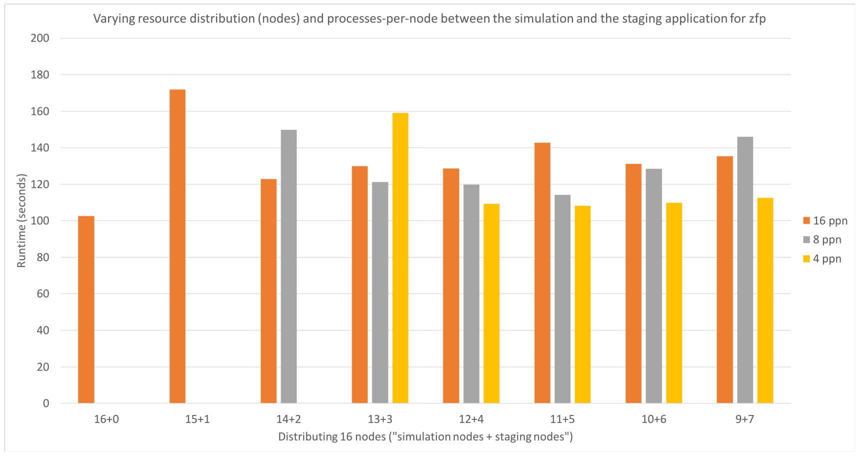
timestep: 52, maxTimestep: 154

[PROPERTY] varName = temp_0154 dataType = 1 r5 = 0 r4 = 0 r3 = 0 r2 = 28 r1 = 100 numOfElem = 11200 minValue = 0 maxValue = 100 valueRange = 100 avgValue = 5.386157826 entropy = 35.98721536 autocorr = 0.8960237498



(1b) Performance portability across architectures

Our performance portability goal is to enable identification of online coupling configurations that perform well on an architecture:



1. API, Savanna, Cheetah makes it easy to experiment with alternative mechanisms

   E.g., alternative communication layers, task placements, memory vs. NVRAM, compression params, etc, etc.

2. Z-Checker, TAU, and Chimbuko provide for performance monitoring and feedback (including at runtime)

   Online analysis and reduction of performance data, e.g. to adapt compression rates

3. We integrate high-performance implementations of methods (e.g., ADIOS, SZ, ZFP, Z-Checker)

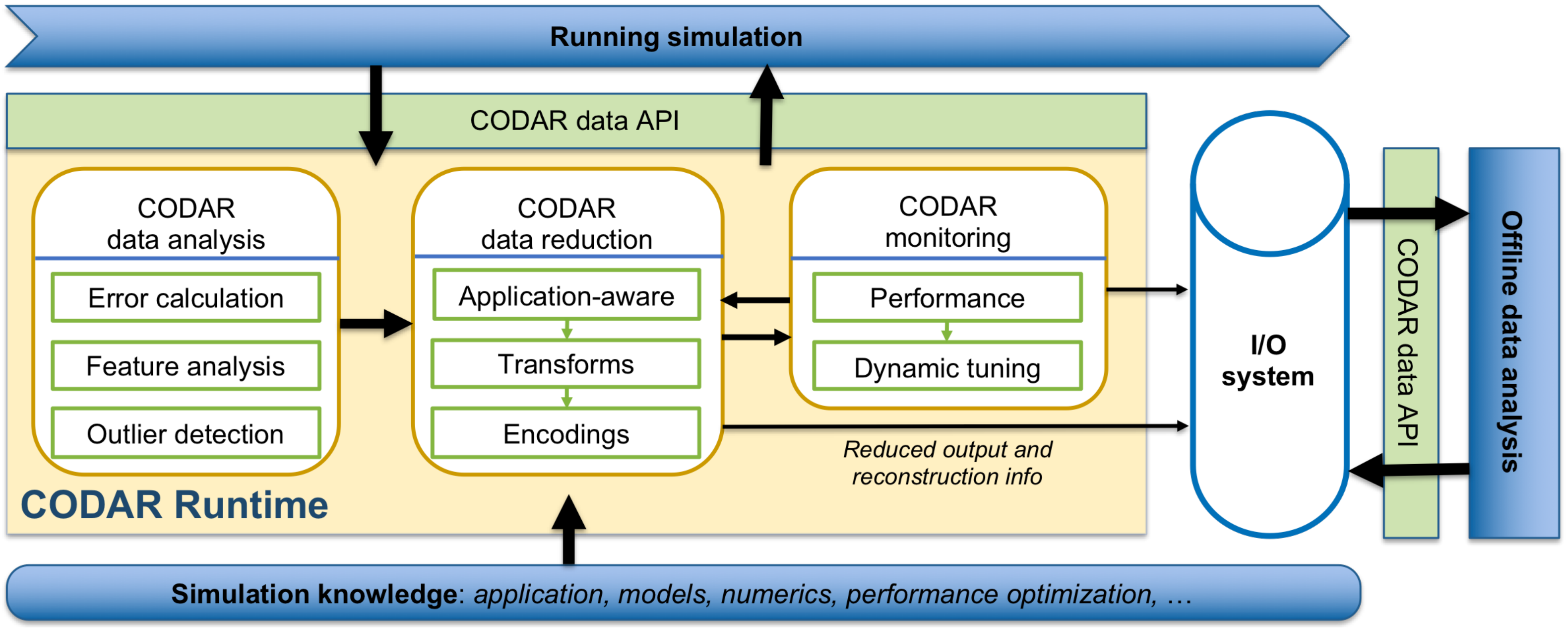
4. If the methods we develop (e.g., MGARD) require optimization on specific machines, then we can take on that task

(2) Usability across applications

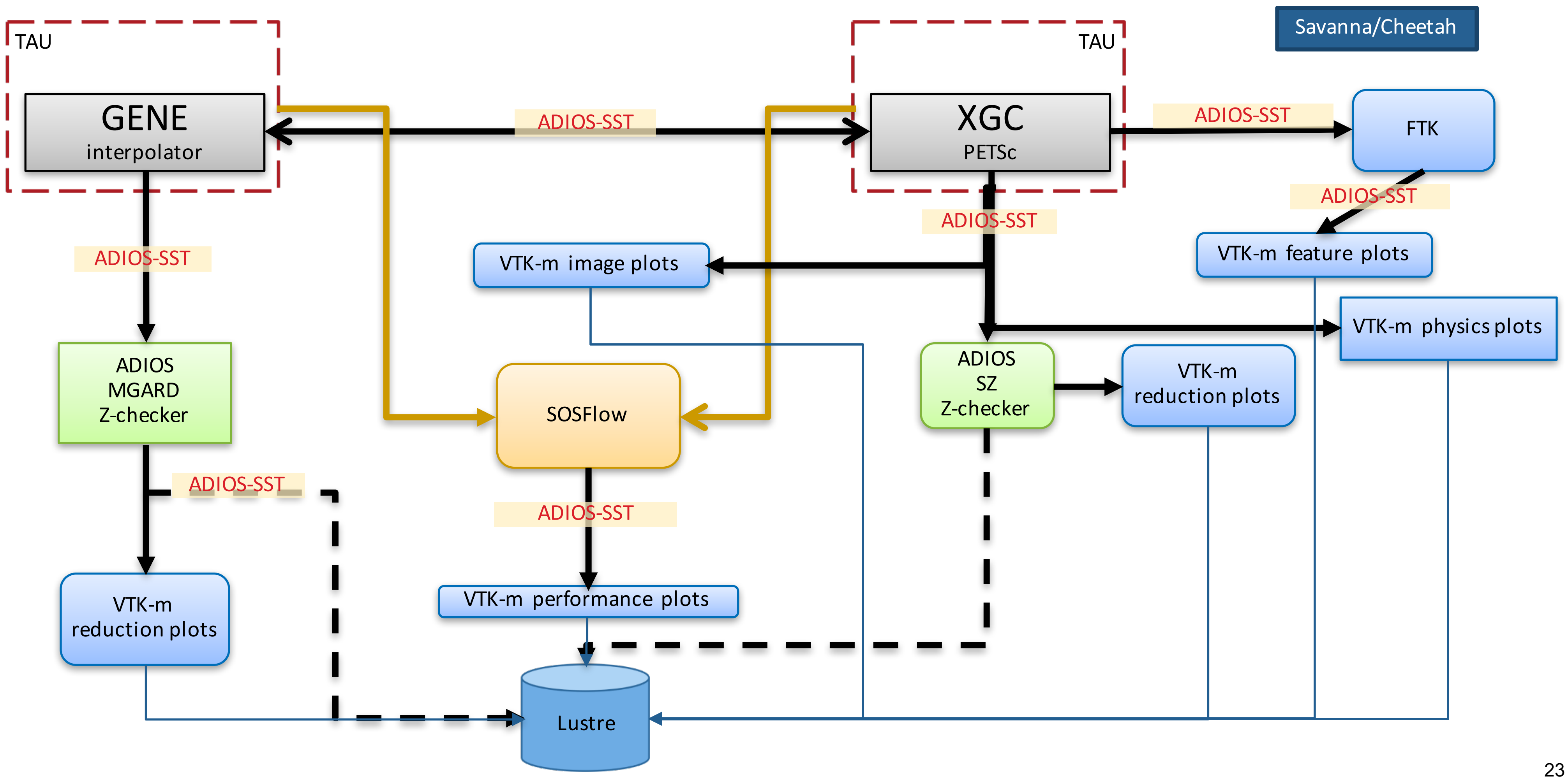
The “CODAR data API” provides a typed I/O abstraction, enabling plug-and-play integration of application, analysis, and reduction components

Analysis, reduction, and monitoring components provided by TODD can all be used in different applications.

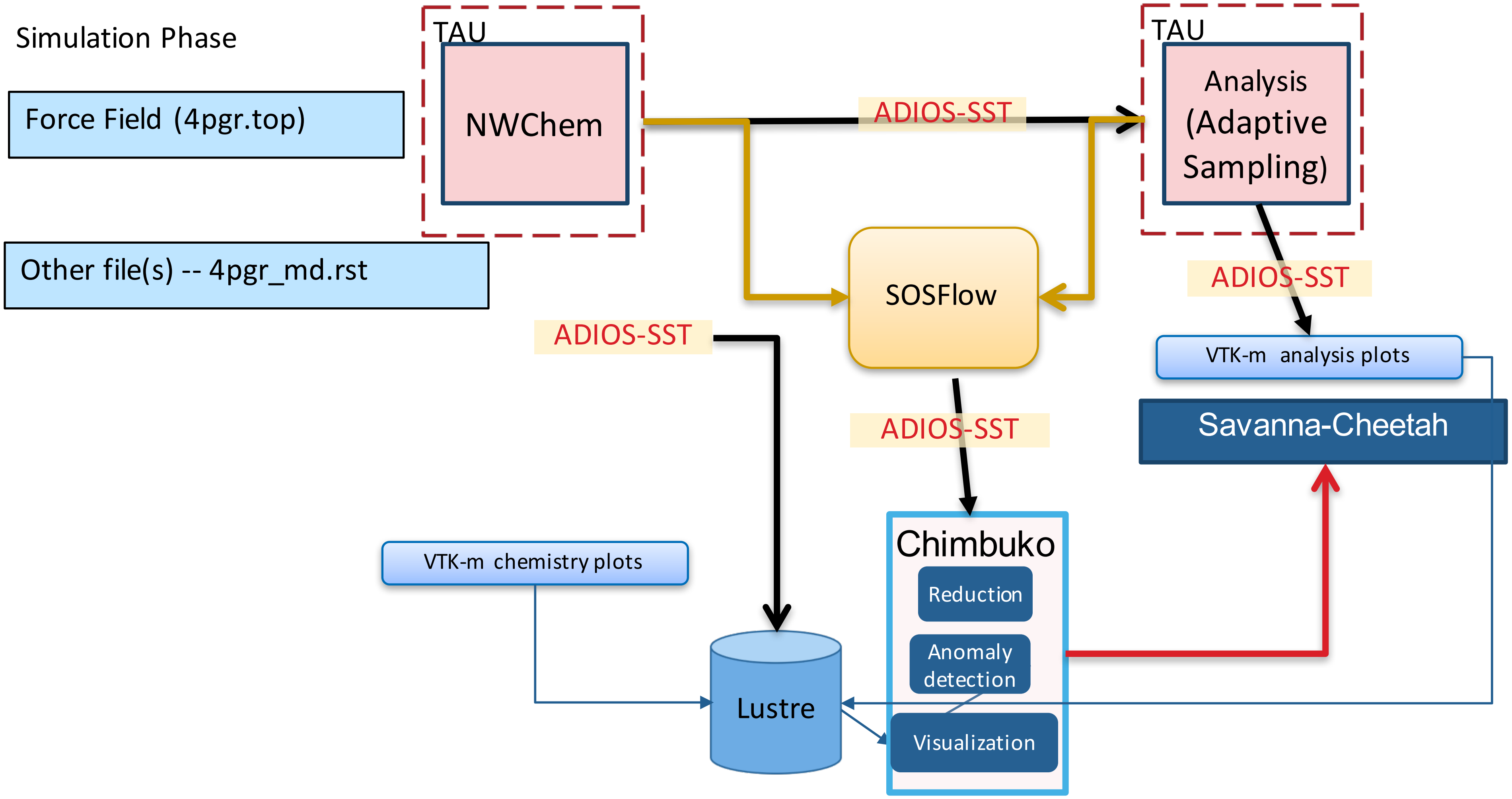
Cheetah, Savanna, Z-Checker, TAU, and Chimbuko are all application independent



Usability across applications



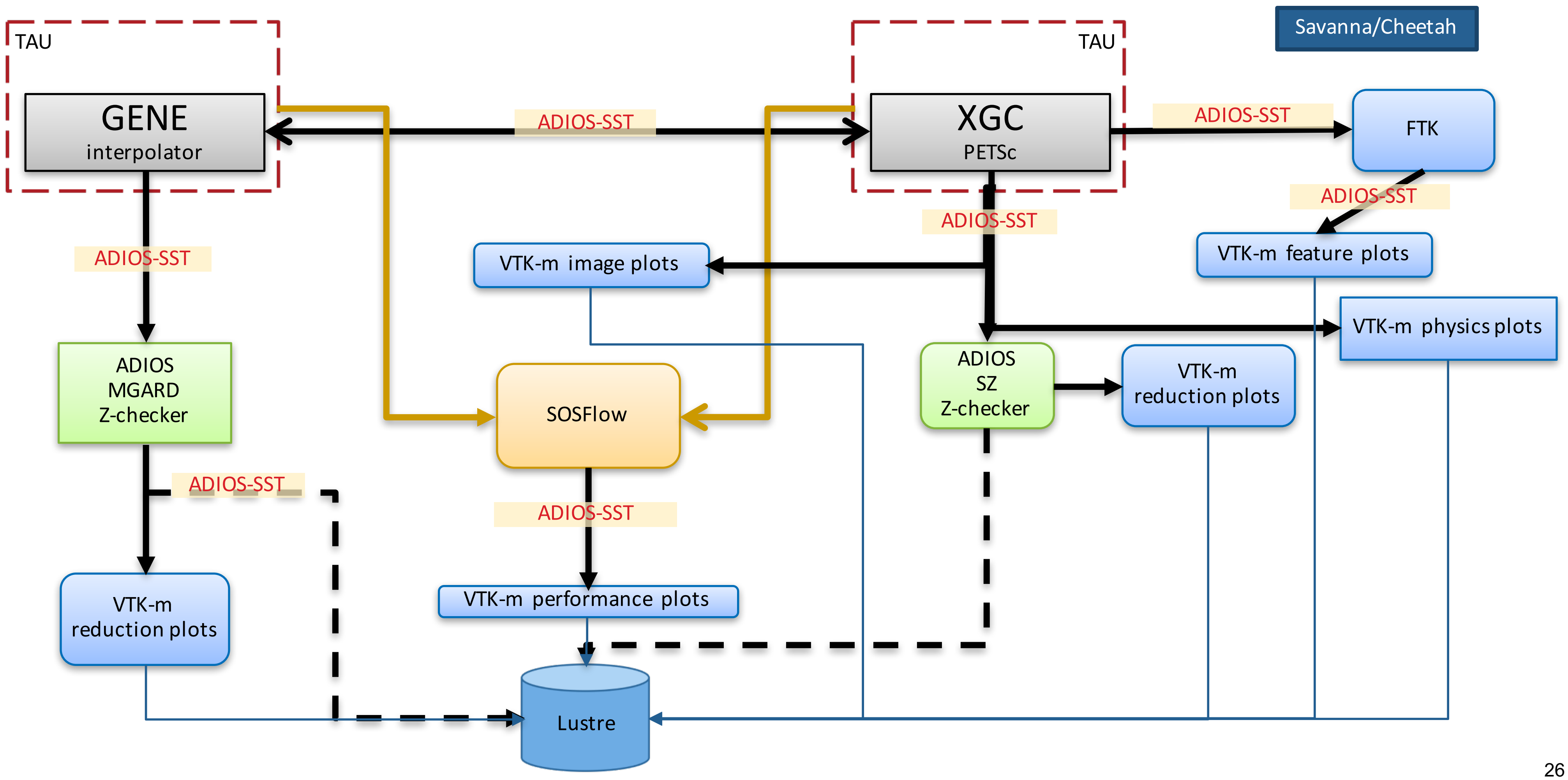
Usability across applications



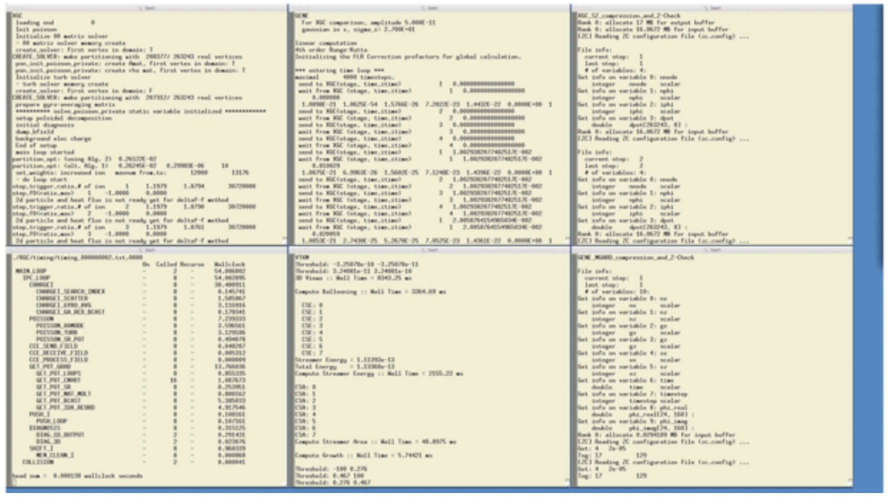
(3a) Broad engagements with applications: Infrastructure Integration and Co-design Studies

- Coordinated effort from application development, software technology, and co-design teams
  - Application workflow including online data analysis and reduction needs
  - Integration of required software technologies into infrastructure
  - Orchestration of workflow using developed co-design software
  - Run, collate, and analyze experimental results from study
- Enable new science and improved performance by application development teams
- Leverage experience from these engagements to produce miniapps that capture data analysis and reduction patterns
- Examples: WDMApp, NWChemEx, CANDLE
- Under discussion: Urban data, EQSIM (seismology)

WDMApp workflow



WDMApp demonstration



```
XGC
 loading end          0
 Init poisson
 Initialize 00 matrix solver
 - 00 matrix solver memory create
 create_solver: first vertex in domain: T
CREATE_SOLVER: make partitioning with  208377/ 263243 real vertices
 psn_init_poisson_private: create Amat, first vertex in domain: T
 psn_init_poisson_private: create rhs mat, first vertex in domain: T
 Initialize turb solver
 - turb solver memory create
 create_solver: first vertex in domain: F
CREATE_SOLVER: make partitioning with  207312/ 263243 real vertices
 prepare gyro-averaging matrix
 *********** solve_poisson_private static variable initialized ************
 setup poloidal decomposition
 initial diagnosis
 dump_bfield
 background elec charge
 End of setup
 main loop started
partition_opt: (using Alg. 2)  0.26122E-02
partition_opt: (alt. Alg. 1)   0.26245E-02   0.29983E-06     19
 set_weights: increased ion   maxnum from,to:       12000       13176
 - do loop start
step,trigger_ratio,# of ion      1     1.1979     1.0794       30720000
step,f0(ratio,max)    1    -1.0000     0.0000
 2d particle and heat flux is not ready yet for deltaf-f method
step,trigger_ratio,# of ion      2     1.1979     1.0790       30720000
step,f0(ratio,max)    2    -1.0000     0.0000
 2d particle and heat flux is not ready yet for deltaf-f method
step,trigger_ratio,# of ion      3     1.1979     1.0761       30720000
step,f0(ratio,max)    3    -1.0000     0.0000
 2d particle and heat flux is not ready yet for deltaf-f method
```

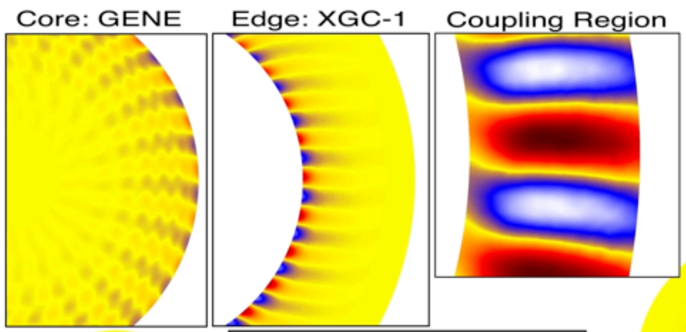
```
GENE
  for XGC comparison, amplitude 5.000E-11
  gaussian in x, sigma_i: 2.700E+01

linear computation
4th order Runge-Kutta
Initializing the FLR Correction prefactors for global calculation.

*** entering time loop ***
maximal       4000 timesteps.
 send to XGC(stage, time,itime)          1   0.0000000000000000
 wait from XGC (stage, time,itime)          1   0.0000000000000000
    0.000000
  1.0098E-21  1.0625E-54  1.5766E-26  7.2022E-23  1.4432E-22  0.0000E+00  1
 send to XGC(stage, time,itime)          2   0.0000000000000000
 wait from XGC (stage, time,itime)          2   0.0000000000000000
 send to XGC(stage, time,itime)          3   0.0000000000000000
 wait from XGC (stage, time,itime)          3   0.0000000000000000
 send to XGC(stage, time,itime)          4   0.0000000000000000
 wait from XGC (stage, time,itime)          4   0.0000000000000000
 send to XGC(stage, time,itime)          1   1.0029382877482517E-002
 wait from XGC (stage, time,itime)          1   1.0029382877482517E-002
    0.010029
  1.0075E-21  6.9963E-26  1.5602E-25  7.1240E-23  1.4396E-22  0.0000E+00  1
 send to XGC(stage, time,itime)          2   1.0029382877482517E-002
 wait from XGC (stage, time,itime)          2   1.0029382877482517E-002
 send to XGC(stage, time,itime)          3   1.0029382877482517E-002
 wait from XGC (stage, time,itime)          3   1.0029382877482517E-002
 send to XGC(stage, time,itime)          4   1.0029382877482517E-002
 wait from XGC (stage, time,itime)          4   1.0029382877482517E-002
 send to XGC(stage, time,itime)          1   2.0058764554965034E-002
 wait from XGC (stage, time,itime)          1   2.0058764554965034E-002
    0.020059
  1.0053E-21  2.7430E-25  5.2670E-25  7.0525E-23  1.4361E-22  0.0000E+00  1
```

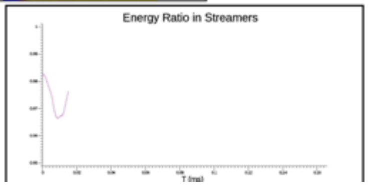
```
XGC_SZ_compression_and_Z-Check
Rank 0: allocate 17 MB for output buffer
Rank 0: allocate 16.0672 MB for input buffer
[ZC] Reading ZC configuration file (zc.config) ...

File info:
  current step:   1
  last step:      1
  # of variables: 4:
Get info on variable 0: nnode
    integer    nnode    scalar
Get info on variable 1: nphi
    integer    nphi     scalar
Get info on variable 2: iphi
    integer    iphi     scalar
Get info on variable 3: dpot
    double     dpot[263243, 8] :
Rank 0: allocate 16.0672 MB for input buffer
[ZC] Reading ZC configuration file (zc.config) ...

File info:
  current step:   2
  last step:      2
  # of variables: 4:
Get info on variable 0: nnode
    integer    nnode    scalar
Get info on variable 1: nphi
    integer    nphi     scalar
Get info on variable 2: iphi
    integer    iphi     scalar
Get info on variable 3: dpot
    double     dpot[263243, 8] :
Rank 0: allocate 16.0672 MB for input buffer
[ZC] Reading ZC configuration file (zc.config) ...
```

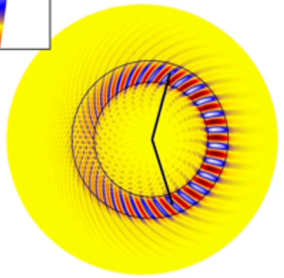
```
./XGC/timing/timing_000000002.txt.0000
                                       On  Called Recurse    Wallclock
 MAIN_LOOP                              -      2     -      54.006002
   IPC_LOOP                             -      8     -      54.002095
     CHARGEI                            -      8     -      30.400911
       CHARGEI_SEARCH_INDEX             -      8     -       0.145741
       CHARGEI_SCATTER                  -      8     -       1.505067
       CHARGEI_GYRO_AVG                 -      8     -       3.116916
       CHARGEI_GA_RED_BCAST             -      8     -       0.170341
     POISSON                            -      8     -       7.239333
       POISSON_00MODE                   -      8     -       3.590561
       POISSON_TURB                     -      8     -       3.129186
       POISSON_SR_POT                   -      8     -       0.494878
     CCE_SEND_FIELD                     -      8     -       0.040267
     CCE_RECEIVE_FIELD                  -      8     -       0.005312
     CCE_PROCESS_FIELD                  -      8     -       0.000009
     GET_POT_GRAD                       -      8     -      13.766836
       GET_POT_LOOPS                    -      8     -       0.055335
       GET_POT_CNVRT                    -     16     -       5.087673
       GET_POT_SR                       -      8     -       0.252951
       GET_POT_MAT_MULT                 -      8     -       0.000162
       GET_POT_BCAST                    -      8     -       5.385833
       GET_POT_ION_REORD                -      8     -       4.917546
     PUSH_I                             -      8     -       0.160161
       PUSH_LOOP                        -      8     -       0.147161
     DIAGNOSIS                          -      8     -       0.315175
       DIAG_1D_OUTPUT                   -      2     -       0.291431
       DIAG_3D                          -      2     -       0.023876
     SHIFT_I                            -      8     -       0.960339
       MEM_CLEAN_I                      -      8     -       0.000068
   COLLISION                            -      2     -       0.000041

head sum =  0.000130 wallclock seconds
```

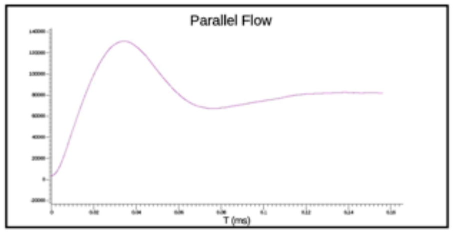
```
VTKM
Threshold: -3.25078e-10 -3.25078e-11
Threshold: 3.24081e-11 3.24081e-10
3D Views :: Wall Time = 8343.25 ms

Compute Ballooning :: Wall Time = 3364.69 ms

  CSE: 0
  CSE: 1
  CSE: 2
  CSE: 3
  CSE: 4
  CSE: 5
  CSE: 6
  CSE: 7
Streamer Energy = 1.11392e-13
Total Energy    = 1.13308e-13
Compute Streamer Energy :: Wall Time = 2155.22 ms

CSA: 0
CSA: 1
CSA: 2
CSA: 3
CSA: 4
CSA: 5
CSA: 6
CSA: 7
Compute Streamer Area :: Wall Time = 40.0975 ms

Compute Growth :: Wall Time = 5.74421 ms

Threshold: -100 0.276
Threshold: 0.467 100
Threshold: 0.276 0.467
```

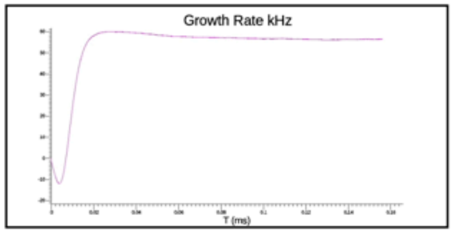
```
GENE_MGARD_compression_and_Z-Check

File info:
  current step:   1
  last step:      1
  # of variables: 10:
Get info on variable 0: nx
    integer    nx       scalar
Get info on variable 1: nz
    integer    nz       scalar
Get info on variable 2: gx
    integer    gx       scalar
Get info on variable 3: gz
    integer    gz       scalar
Get info on variable 4: ox
    integer    ox       scalar
Get info on variable 5: oz
    integer    oz       scalar
Get info on variable 6: time
    double     time     scalar
Get info on variable 7: timestep
    integer    timestep scalar
Get info on variable 8: phi_real
    double     phi_real[24, 160] :
Get info on variable 9: phi_imag
    double     phi_imag[24, 160] :
Rank 0: allocate 0.0294189 MB for input buffer
[ZC] Reading ZC configuration file (zc.config) ...
Get: 4   2e-05
Tag: 17         129
[ZC] Reading ZC configuration file (zc.config) ...
Get: 4   2e-05
Tag: 17         129
```

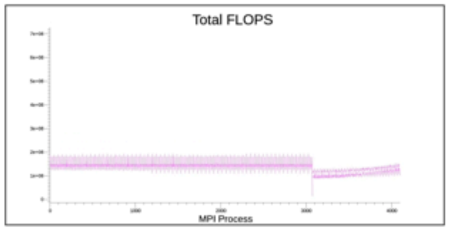


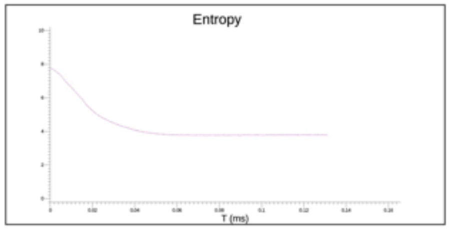


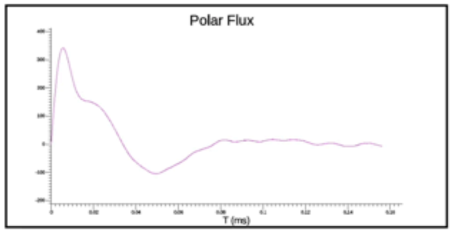


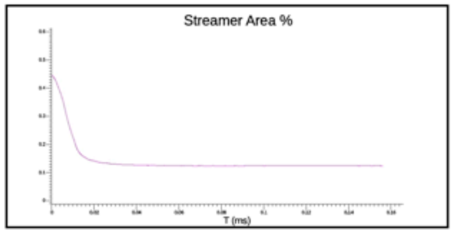


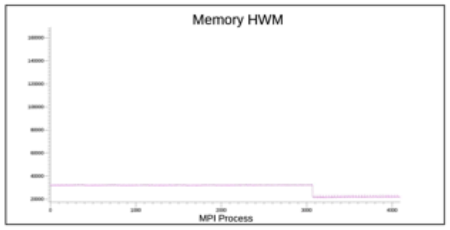


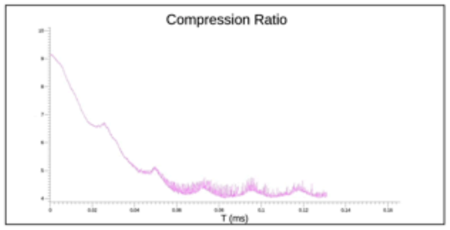






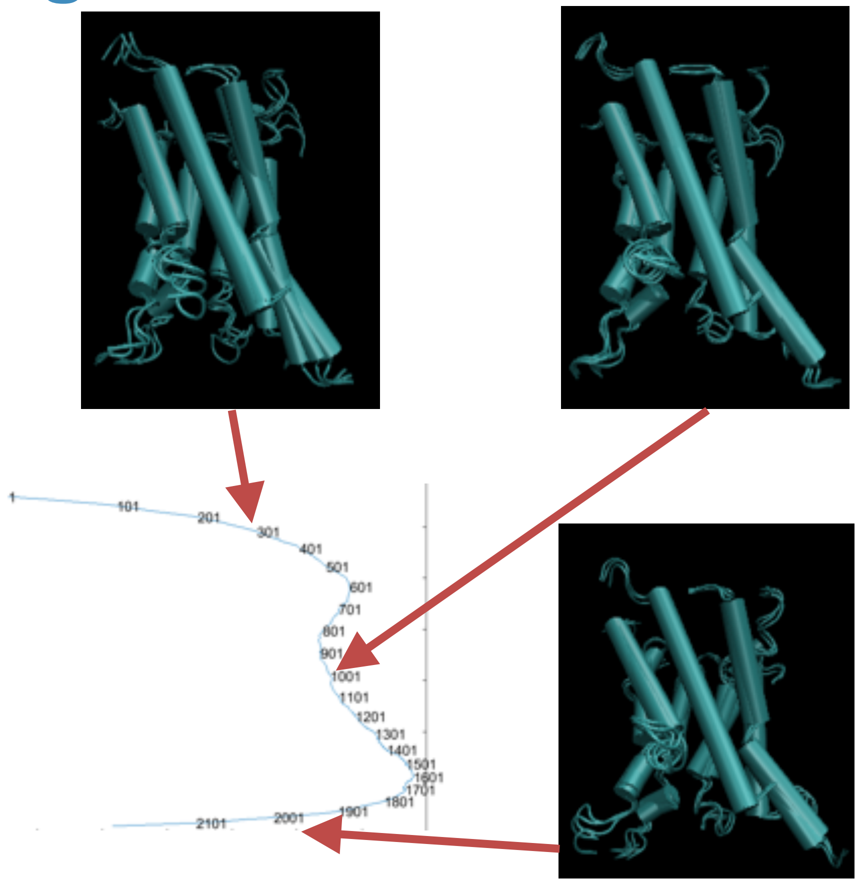






NWChemEx adaptive sampling

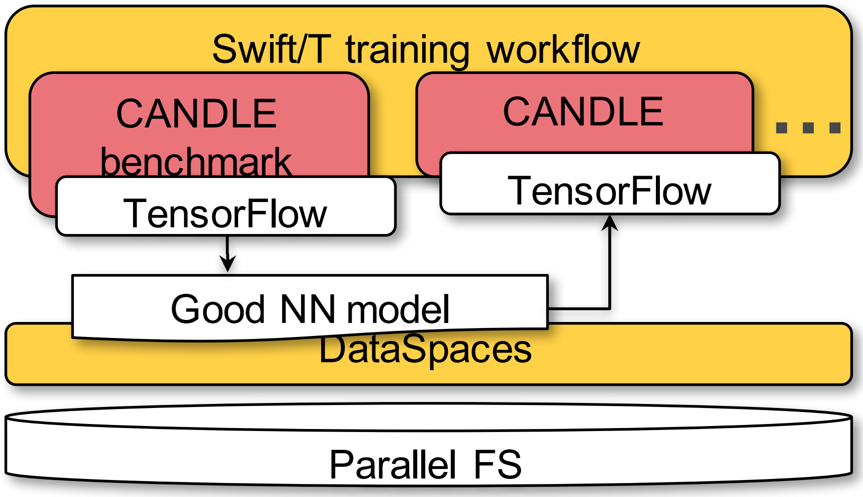
- Motivation
  - Not all trajectories are interesting
  - Sampling only significant phase change events
- Challenges
  - Atoms vibrate significantly
  - Difficult to compress
- Approaches
  - Manifold learning for trajectories
    - Learn the intrinsic relations among time points
    - Less vibrational noise than original space
    - Effectively compress high dimensional coordinates
    - Matrix sketch of trajectories
  - Important / Interesting event detection
    - L2 normalized gradient changes
    - Mean filter for smoothed trajectories
    - Weighted reservoir sampling of gradient changes

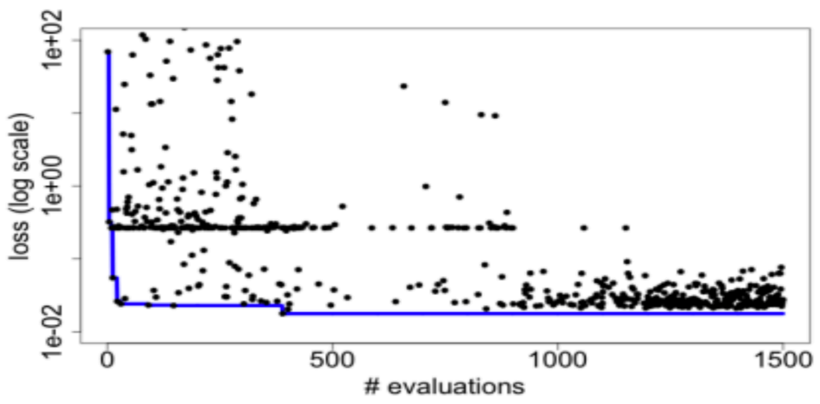


Low-dimensional manifold projection of different state of MD trajectories

CANDLE workflow and demonstration

- CANDLE workflows produce many medium-sized machine learning models
  - Cache these on compute node storage for possible later use
  - Flush to global filesystem before end of run
- This activity
  - Integrates Swift/T workflow system with DataSpaces client for caching
  - Accelerates workflow performance
  - Enables novel training strategies (parameter sharing)
  - Provides an opportunity for workflow-based data analysis and I/O reduction
  - Demonstrate the utility of node-local storage for complex workflows



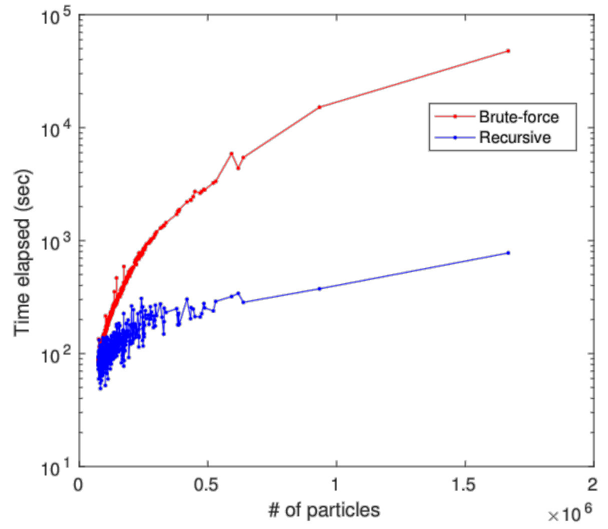


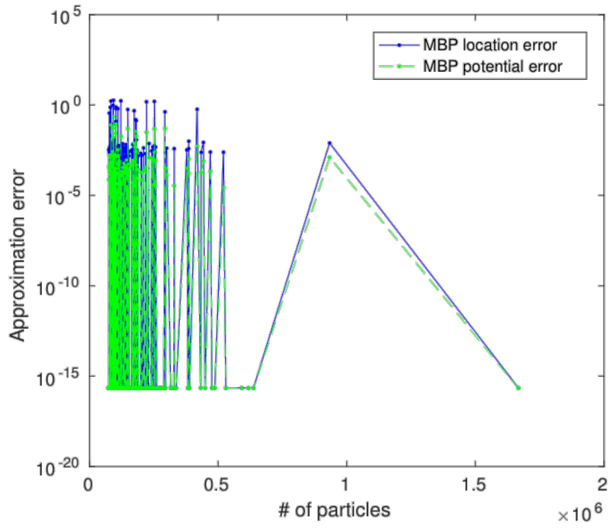
(3b) Focused engagements with applications: Data Analysis and Reduction Methods

- Targeted engagement with application team members to identify novel methods
  - Smaller engagements between a small number of team members
  - Fill gaps in available methods (e.g., feature tracking across timesteps)
  - Cover full range of data types (e.g., structured, unstructured, or particle data)
  - Exhibit unique data access patterns (e.g., multigrid or hierarchical access)
- Abstract and prototype general capability to address needs; demonstrate offline; evaluate
  - **Exploration** phase: improved statistical metrics for compression quality analysis (ExaSky, WDMApp, others), numerical optimization based compression (EXAALT)
  - **Development** phase: adaptive sampling (NWChemEx), feature tracking kit (WDMApp), multilevel data reduction (Combustion), performance anomaly detection, hierarchical data analysis (ExaSky implementing in CosmoTools)
  - **Completion** phase: functional data analysis (WDMApp)
- Provide artifacts to engage the community

Completed: Hierarchical Data Analysis (ExaSky)

- Many scientific data analysis problems require
  - Discovering latent/intrinsic structure within the data
  - Grouping the data into corresponding clusters
  - Performing operations on the respective groups
- Hierarchical data analysis methods exploit nested structure to improve performance
- Our hierarchical method to approximate the most bound point (MBP) shows dramatic speedup compared to other common approaches

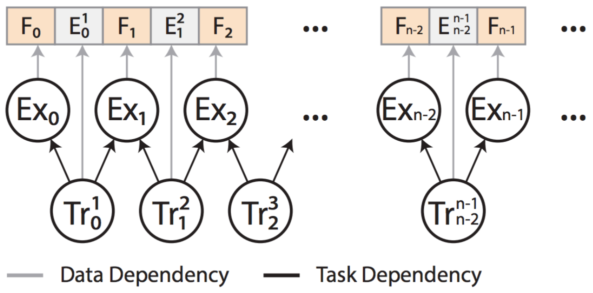




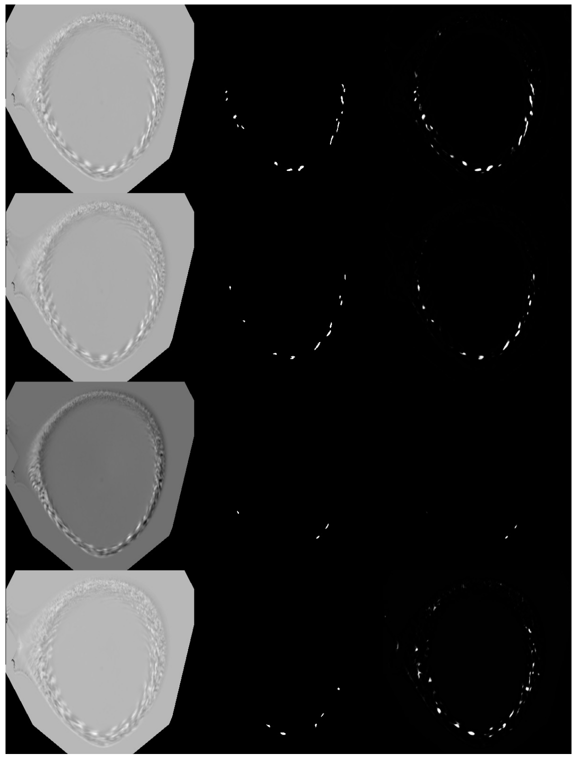
Result of applying hierarchical method for various sets of halos. Left plot shows the time comparison with brute force method, while the right plot shows the approximation error from both the location error and MBP value error.

Development: Feature Tracking Kit (WDMApp)

- Feature tracking is needed by all sciences. It can reduce the amount of data to store, while preserving insights
- Our online feature tracking kit contains
  - Core building blocks
    - Tracking graph for event characterization
    - Abstraction for a zoo of features, including points, lines, surfaces, and regions
  - In situ coupling
    - Integration with Decaf and ADIOS
    - Task-parallelism for in situ feature extraction and feature tracking
  - Algorithms
    - Deep learning/machine learning
    - Statistical approaches
    - Topological approaches



Blob tracking workflow

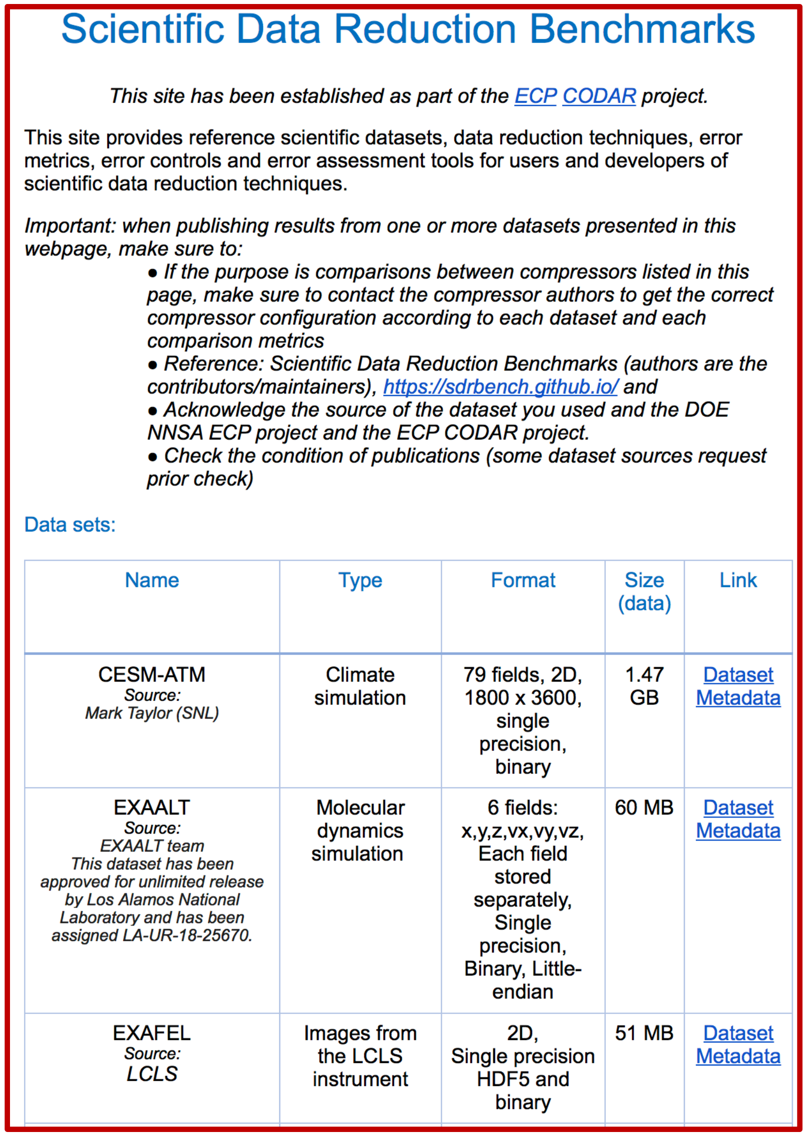


Preliminary results of blob detection with U-Net

Community engagement

- Scientific Data Reduction Benchmarks
  - Public collection of scientific data sets for performing comparisons
  - Community can contribute datasets, compression tools, metrics, analysis methods, etc.
- 14 datasets as of 8/22/18
  - Climate, cosmology, light sources, combustion, fusion, molecular dynamics, and others
  - Image, particle, array, and other types

https://sdrbench.github.io/



Scientific Data Reduction Benchmarks

This site has been established as part of the ECP CODAR project.

This site provides reference scientific datasets, data reduction techniques, error metrics, error controls and error assessment tools for users and developers of scientific data reduction techniques.

Important: when publishing results from one or more datasets presented in this webpage, make sure to:

- *If the purpose is comparisons between compressors listed in this page, make sure to contact the compressor authors to get the correct compressor configuration according to each dataset and each comparison metrics*
- *Reference: Scientific Data Reduction Benchmarks (authors are the contributors/maintainers), https://sdrbench.github.io/ and*
- *Acknowledge the source of the dataset you used and the DOE NNSA ECP project and the ECP CODAR project.*
- *Check the condition of publications (some dataset sources request prior check)*

Data sets:

Name	Type	Format	Size (data)	Link
CESM-ATM *Source: Mark Taylor (SNL)*	Climate simulation	79 fields, 2D, 1800 x 3600, single precision, binary	1.47 GB	Dataset Metadata
EXAALT *Source: EXAALT team This dataset has been approved for unlimited release by Los Alamos National Laboratory and has been assigned LA-UR-18-25670.*	Molecular dynamics simulation	6 fields: x,y,z,vx,vy,vz, Each field stored separately, Single precision, Binary, Little-endian	60 MB	Dataset Metadata
EXAFEL *Source: LCLS*	Images from the LCLS instrument	2D, Single precision HDF5 and binary	51 MB	Dataset Metadata

Some lessons learned

- Engagement with applications is essential but care is needed to maximize impact
  - Full-press demonstrations provide many benefits but are resource intensive
  - Focusing resources on development of good online coupling mini-apps for use in co-design studies can reduce resource needs, but impact may be more diffuse
  - Engage with other co-design centers as a route to greater impact?
- Novel data analysis methods provide another path towards development miniapps
  - Support application needs, while developing a general capability
- The continuous integration problem
  - CODAR must integrate N components, each on multiple computers and often with multiple compilers. Each component has reliability $R_i < 1$. $R_{total} = \prod_N R_i << 1$.
  - Cheetah is useful for uncovering problems, but not (yet) for diagnosing and correcting them
  - Needed: Continuous (or at least periodic) integration for all components
- Co-design engagements with exascale hardware and software are challenging due to time scales
  - For example, mechanisms are needed to launch, monitor, and manage sub computations
  - Solution: Use mini-apps to communicate needs
  - MPIX_Comm_launch as an exemplar

Recap

At **exascale,** you will need **ten million** FLOPS per **double** written to storage!!

The compute-data gap will continue to widen forcing a **tradeoff** between computing and I/O

Online coupling, including with analysis and reduction, is part of the answer to making this tradeoff

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

ECP EXASCALE COMPUTING PROJECT

RESEARCH SPONSORED BY

The Exascale Computing Project

A Collaborative effort of the U.S. Department of Energy Office of Science And the National Nuclear Security Administration

(17-SC-20-SC)

U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration