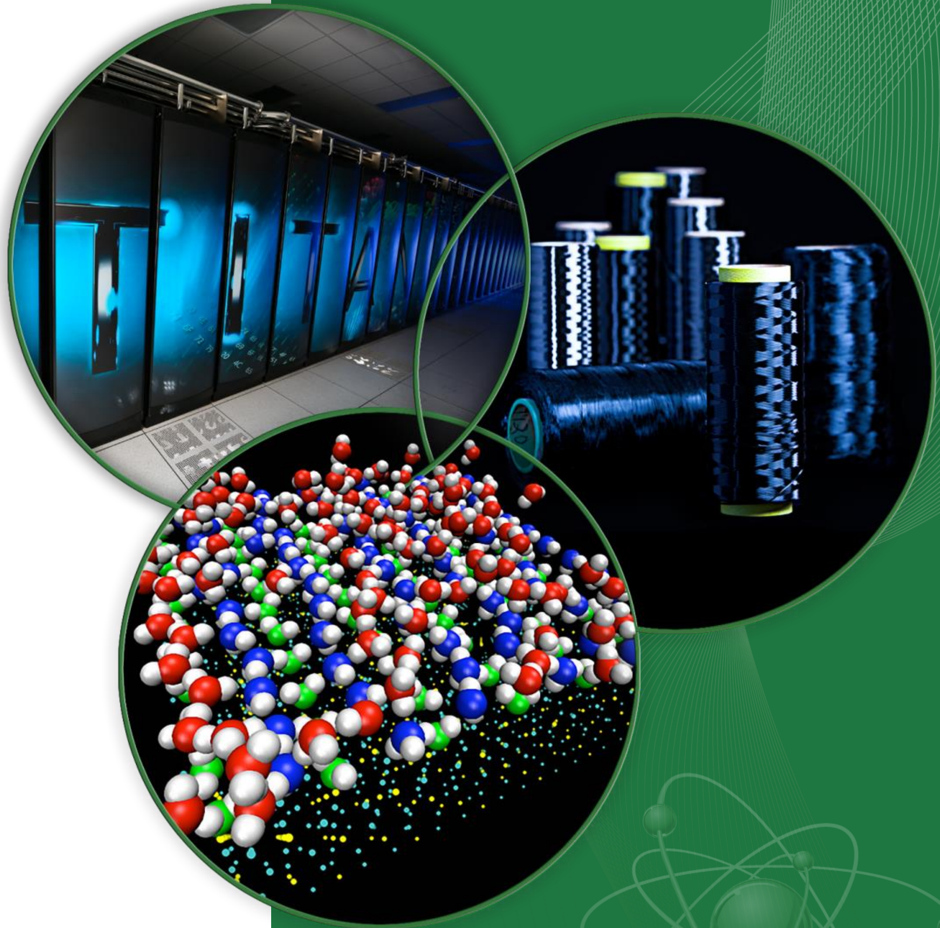
Exploring Emerging Technologies in the Extreme Scale HPC Co-Design Space with Holistic Performance Prediction

Jeffrey S. Vetter

Jeremy Meredith



ISC Workshop: Performance Modeling: Methods and Applications

Frankfurt

16 Jul 2015

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Georgia Tech College of Computing
Computational Science and Engineering

ORNL is managed by UT-Battelle
for the US Department of Energy

http://ft.ornl.gov ♦ vetter@computer.org

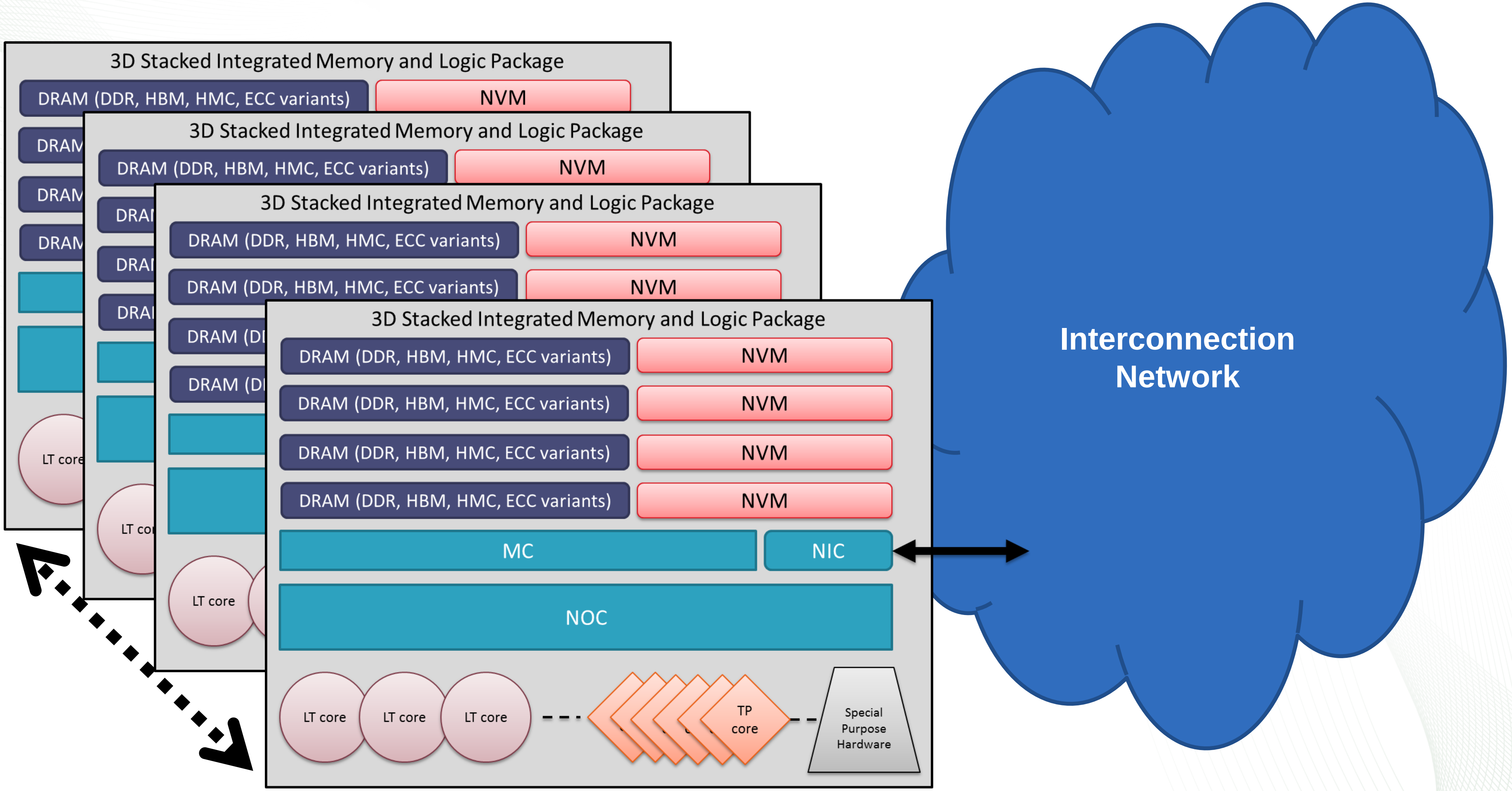
OAK RIDGE National Laboratory

Overview

- Our community has major challenges in HPC as we move to extreme scale
  - Power, Performance, Resilience, Productivity
  - New technologies emerging to address some of these challenges
    - Heterogeneous computing
    - Nonvolatile memory
  - Not just HPC: Most uncertainty in at least two decades
- We need performance prediction and engineering tools now more than ever!
- Aspen is a tool for structured design and analysis
  - Co-design applications and architectures for performance, power, resiliency
  - Automatic model generation
  - Scalable to distributed scientific workflows
  - DVF – a new twist on resiliency modeling

Notional Future Architecture

See ISC30 talks



OAK RIDGE National Laboratory

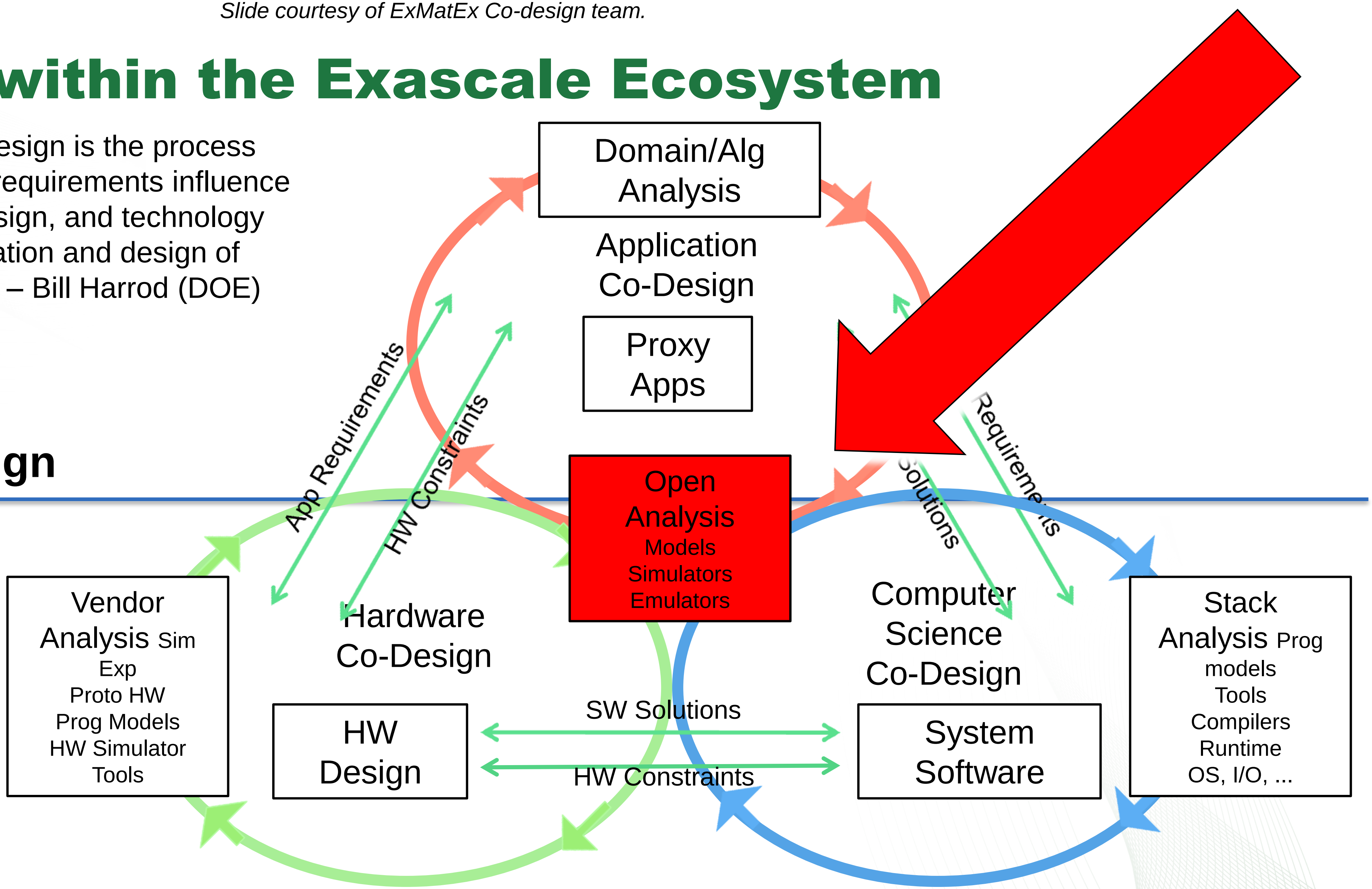
Slide courtesy of ExMatEx Co-design team.

Workflow within the Exascale Ecosystem

"(Application driven) co-design is the process where scientific problem requirements influence computer architecture design, and technology constraints inform formulation and design of algorithms and software." – Bill Harrod (DOE)

Application Design

System Design



Prediction Techniques Ranked

	Speed	Ease	Flexibility	Accuracy	Scalability
Ad-hoc Analytical Models	1	3	2	4	1
Structured Analytical Models	1	2	1	4	1
Simulation – Functional	3	2	2	3	3
Simulation – Cycle Accurate	4	2	2	2	4
Hardware Emulation (FPGA)	3	3	3	2	3
Similar hardware measurement	2	1	4	2	2
Node Prototype	2	1	4	1	4
Prototype at Scale	2	1	4	1	2
Final System	-	-	-	-	-

Prediction Techniques Ranked

	Speed	Ease	Flexibility	Accuracy	Scalability
Ad-hoc Analytical Models	1	3	2	4	1
Structured Analytical Models	1	2	1	4	1
Aspen	1	1	1	4	1
Simulation – Functional	3	2	2	3	3
Simulation – Cycle Accurate	4	2	2	2	4
Hardware Emulation (FPGA)	3	3	3	2	3
Similar hardware measurement	2	1	4	2	2
Node Prototype	2	1	4	1	4
Prototype at Scale	2	1	4	1	2
Final System	-	-	-	-	-

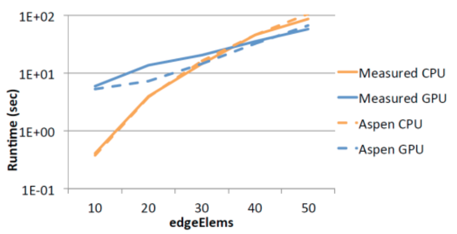
Aspen: Abstract Scalable Performance Engineering Notation

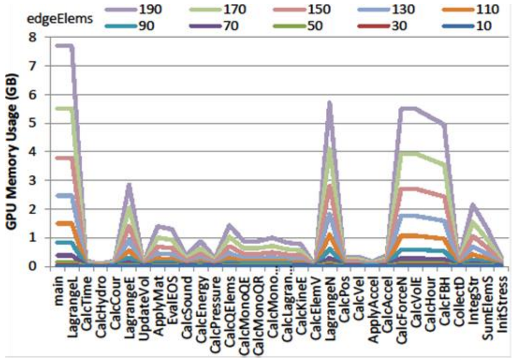
Source code

```
static inline
void CalcMonotonicQGradientsForElems(Index_t p_nodelist[T_NUMELEM8],
        Real_t p_x[T_NUMNODE], Real_t p_y[T_NUMNODE], Real_t p_z[T_NUMNODE],
        Real_t p_xd[T_NUMNODE], Real_t p_yd[T_NUMNODE],Real_t p_zd[T_NUMNODE],
        Real_t p_volo[T_NUMELEM], Real_t p_vnew[T_NUMELEM],
        Real_t p_delx_zeta[T_NUMELEM], Real_t p_delv_zeta[T_NUMELEM],
        Real_t p_delx_xi[T_NUMELEM], Real_t p_delv_xi[T_NUMELEM],
        Real_t p_delx_eta[T_NUMELEM], Real_t p_delv_eta[T_NUMELEM])
{
   Index_t i;
   Index_t numElem = m_numElem;
#pragma acc parallel loop independent present(p_vnew, p_nodelist, p_x, p_y, p_z, p_xd, \
p_yd, p_zd, p_volo, p_delx_xi, p_delx_eta, p_delx_zeta, p_delv_xi, p_delv_eta,\
p_delv_zeta)
   for (i = 0 ; i < numElem ; ++i ) {
      const Real_t ptiny = 1.e-36 ;
      Real_t ax,ay,az ;
      Real_t dxv,dyv,dzv ;

      const Index_t *elemToNode = &p_nodelist[8*i];
      Index_t n0 = elemToNode[0] ;
      Index_t n1 = elemToNode[1] ;
      Index_t n2 = elemToNode[2] ;
      Index_t n3 = elemToNode[3] ;
      Index_t n4 = elemToNode[4] ;
      Index_t n5 = elemToNode[5] ;
      Index_t n6 = elemToNode[6] ;
      Index_t n7 = elemToNode[7] ;

      Real_t x0 = p_x[n0] ;
```

Aspen code

```
kernel CalcMonotonicQGradients {
  execute [numElems]
  {
    loads [8 * indexWordSize] from nodelist
    // Load and cache position and velocity.
    loads/caching [8 * wordSize] from x
    loads/caching [8 * wordSize] from y
    loads/caching [8 * wordSize] from z

    loads/caching [8 * wordSize] from xvel
    loads/caching [8 * wordSize] from yvel
    loads/caching [8 * wordSize] from zvel

    loads [wordSize] from volo
    loads [wordSize] from vnew
    // dx, dy, etc.
    flops [90] as dp, simd
    // delvk delxk
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delv_xeta
    // delxi delvi
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delx_xi
    // delxj and delvj
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delv_eta
  }
}
```





Creation

- Static analysis via compilers
- Empirical, Historical
- Manual for future applications

Representation in Aspen
- **Modular**
- **Sharable**
- **Composable**
- **Reflects prog structure**

Existing models for MD, UHPC CP 1, Lulesh, 3D FFT, CoMD, VPFFT, …

Use

- Interactive tools for graphs, queries
- Design space optimization
- Drive simulators
- Feedback to runtime systems

Researchers are using Aspen for parallel applications, scientific workflows, capacity planning, quantum computing, etc

K. Spafford and J.S. Vetter, "Aspen: A Domain Specific Language for Performance Modeling," in *SC12: ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis, 2012*

Oak Ridge National Laboratory

Manual Example of LULESH

branch: master aspen / models / lulesh / **lulesh.aspen**

jsmeredith on Sep 20, 2013 adding models

1 contributor

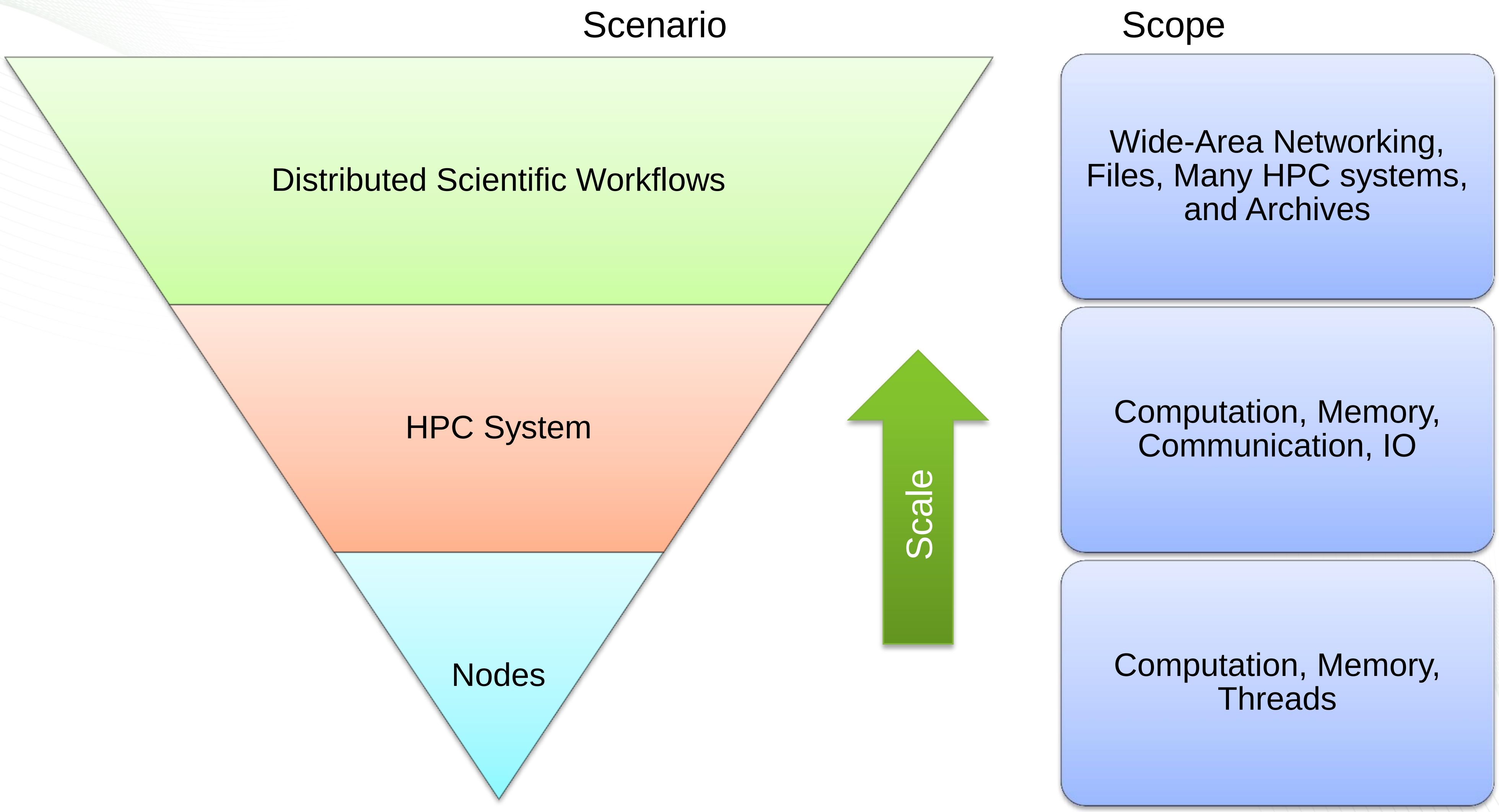
336 lines (288 sloc) | 9.213 kb

Raw Blame History

```
//
// lulesh.aspen
//
// An ASPEN application model for the LULESH 1.01 challenge problem.  Based
// on the CUDA version of the source code found at:
//  https://computation.llnl.gov/casc/ShockHydro/
//
param nTimeSteps = 1495

// Information about domain
param edgeElems = 45
param edgeNodes = edgeElems + 1

param numElems = edgeElems^3
param numNodes = edgeNodes^3

// Double precision
param wordSize = 8

// Element data
data mNodeList as Array(numElems, wordSize)
data mMatElemList as Array(numElems, wordSize)
data mNodeList as Array(8 * numElems, wordSize) //  8 nodes per element
data mlxim as Array(numElems, wordSize)
data mlxip as Array(numElems, wordSize)
data mletam as Array(numElems, wordSize)
data mletap as Array(numElems, wordSize)
data mzetam as Array(numElems, wordSize)
data mzetap as Array(numElems, wordSize)
data melemBC as Array(numElems, wordSize)
data mE as Array(numElems, wordSize)
data mP as Array(numElems, wordSize)
```

```
kernel CalcMonotonicQGradients {
  execute [numElems]
  {
    loads [8 * indexWordSize] from nodelist
    // Load and cache position and velocity.
    loads/caching [8 * wordSize] from x
    loads/caching [8 * wordSize] from y
    loads/caching [8 * wordSize] from z

    loads/caching [8 * wordSize] from xvel
    loads/caching [8 * wordSize] from yvel
    loads/caching [8 * wordSize] from zvel

    loads [wordSize] from volo
    loads [wordSize] from vnew
    // dx, dy, etc.
    flops [90] as dp, simd
    // delvk delxk
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delv_xeta
    // delxi delvi
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delx_xi
    // delxj and delvj
    flops [9 + 8 + 3 + 30 + 5] as dp, simd
    stores [wordSize] to delv_eta
  }
}
```

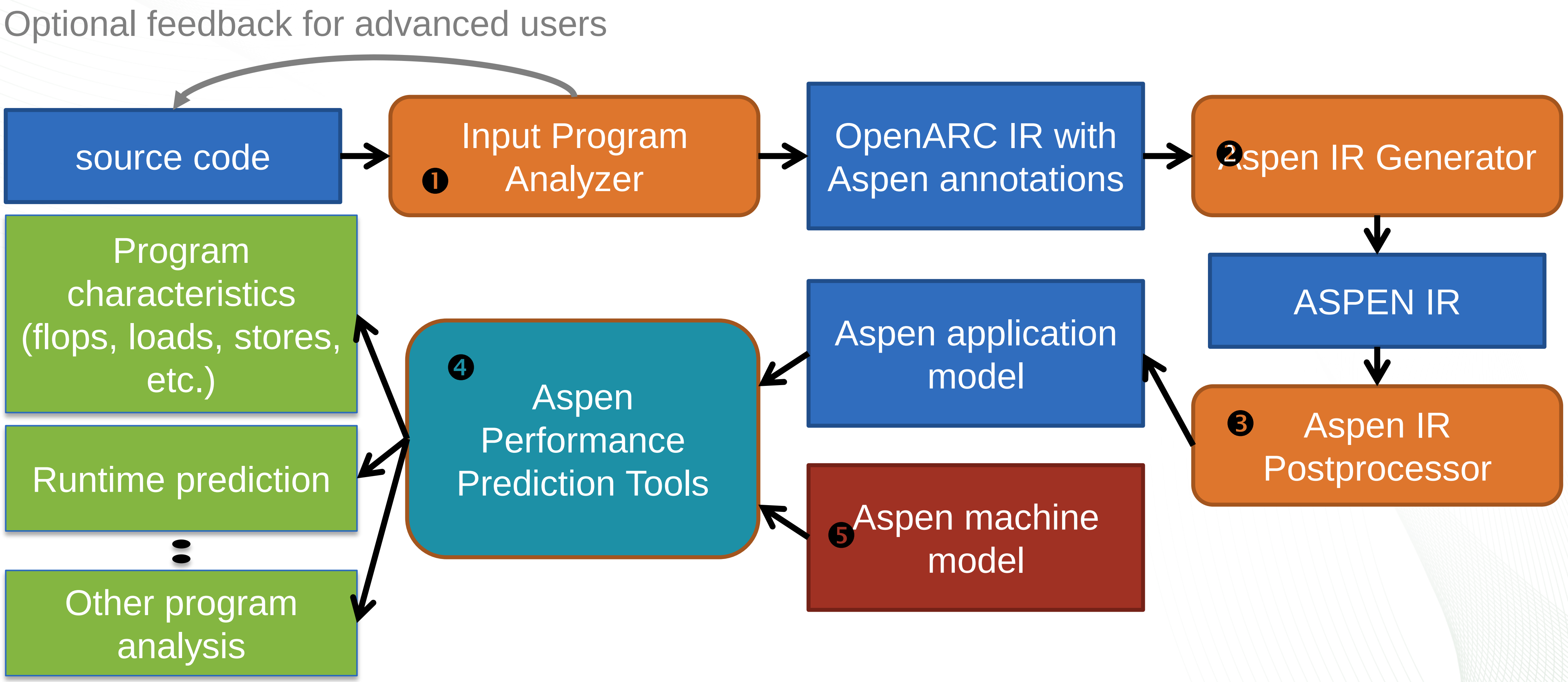
Aspen allows Multiresolution Modeling



Node Scale Modeling with COMPASS

COMPASS System Overview

- Detailed Workflow of the COMPASS Modeling Framework



S. Lee, J.S. Meredith, and J.S. Vetter, "COMPASS: A Framework for Automated Performance Modeling and Prediction," in *ACM International Conference on Supercomputing (ICS)*. *Newport Beach, California: ACM, 2015, 10.1145/2751205.2751220.*

MM example generated from COMPASS

```
int N = 1024;
void matmul(float *a, float *b, float *c){ int i, j, k ;
#pragma acc kernels loop gang copyout(a[0:(N*N)]) \
copyin(b[0:(N*N)],c[0:(N*N)])
  for (i=0; i<N; i++){
#pragma acc loop worker
    for (j=0; j<N; j++) { float sum = 0.0 ;
      for (k=0; k<N; k++) {sum+=b[i*N+k]*c[k*N+j];}
      a[i*N+j] = sum; }
  } //end of i loop
} //end of matmul()
int main() {
  int i; float *A = (float*) malloc(N*N*sizeof(float));
  float *B = (float*) malloc(N*N*sizeof(float));
  float *C = (float*) malloc(N*N*sizeof(float));
  for (i = 0; i <  N*N; i++)
  {  A[i] = 0.0F; B[i] = (float) i; C[i] = 1.0F; }
#pragma aspen modelregion label(MM)
  matmul(A,B,C);
  free(A); free(B); free(C); return 0;
} //end of main()
```

```
model MM {
 param floatS = 4; param N = 1024
 data A as Array((N*N), floatS)
 data B as Array((N*N), floatS)
 data C as Array((N*N), floatS)
 kernel matmul {
  execute matmul2_intracommIN
  {  intracomm [floatS*(N*N)] to C as copyin
   intracomm [floatS*(N*N)] to B as copyin }
  map matmul2 [N] {
   map matmul3 [N] {
    iterate [N] {
     execute matmul5
     {  loads [floatS] from B as stride(1)
      loads [floatS] from C; flops [2] as sp, simd }
    } //end of iterate
    execute matmul6 { stores [floatS] to A as stride(1) }
   } // end of map matmul3
  } //end of map matmul2
  execute matmul2_intracommOUT
  { intracomm [floatS*(N*N)] to A as copyout }
 } //end of kernel matmul
 kernel main { matmul() }
} //end of model MM
```

Input MatMul Code Annotated to Use an Alternative Algorithm

```
int N = 1024;
#pragma aspen control execute flops(N^2.372, traits(sp)) \
  stores(N*N*floatS:to(A):traits(stride(1))) \
  loads(N*N*floatS:from(B):traits(stride(1)), ...) …
void matmul(float * A, float * B, float * C) {
    ... //the original function body is here.
} //end of matmul()

int main()
{
  ... //the original main code is here.
}
```

- The original MatMul code uses a simple algorithm with $O(N^3)$ load operations.
- The new Aspen directive overrides the result produced by the analysis framework for the matmul() function to use the *Coppersmith-Winograd* algorithm that requires only $O(N^{2.372})$ operations, generating a new Aspen application model without rewriting the input program.

Annotation Overhead

Benchmark Name	Lines of Code	Lines of Annotation	Annotation Overhead (%)
JACOBI	241	2	0.8
MATMUL	128	1	0.7
SPMUL	423	10	2.3
LAPLACE2D	210	7	3.3
CG	1511	10	0.6
EP	759	9	1.1
BACKPROP	1074	4	0.3
BFS	435	16	3.6
CFD	752	9	1.1
HOTSPOT	525	11	2.0
KMEANS	1822	11	0.6
LUD	421	6	1.4
NW	478	8	1.7
SRAD	550	12	2.1
LULESH	3743	125	3.3

Example: LULESH (10% of 1 kernel)

```
kernel IntegrateStressForElems
{
   execute [numElem_CalcVolumeForceForElems]
   {
       loads [((1*aspen_param_int)*8)] from elemNodes as stride(1)
       loads [((1*aspen_param_double)*8)] from m_x
       loads [((1*aspen_param_double)*8)] from m_y
       loads [((1*aspen_param_double)*8)] from m_z
       loads [(1*aspen_param_double)] from determ as stride(1)
       flops [8] as dp, simd
       flops [8] as dp, simd
       flops [8] as dp, simd
       flops [8] as dp, simd
       flops [3] as dp, simd
       flops [3] as dp, simd
       flops [3] as dp, simd
       flops [3] as dp, simd
       stores [(1*aspen_param_double)] as stride(0)
       flops [2] as dp, simd
       stores [(1*aspen_param_double)] as stride(0)
       flops [2] as dp, simd
       stores [(1*aspen_param_double)] as stride(0)
       flops [2] as dp, simd
       loads [(1*aspen_param_double)] as stride(0)
       stores [(1*aspen_param_double)] as stride(0)
       loads [(1*aspen_param_double)] as stride(0)
       stores [(1*aspen_param_double)] as stride(0)
       loads [(1*aspen_param_double)] as stride(0)
       . . . . . .
```

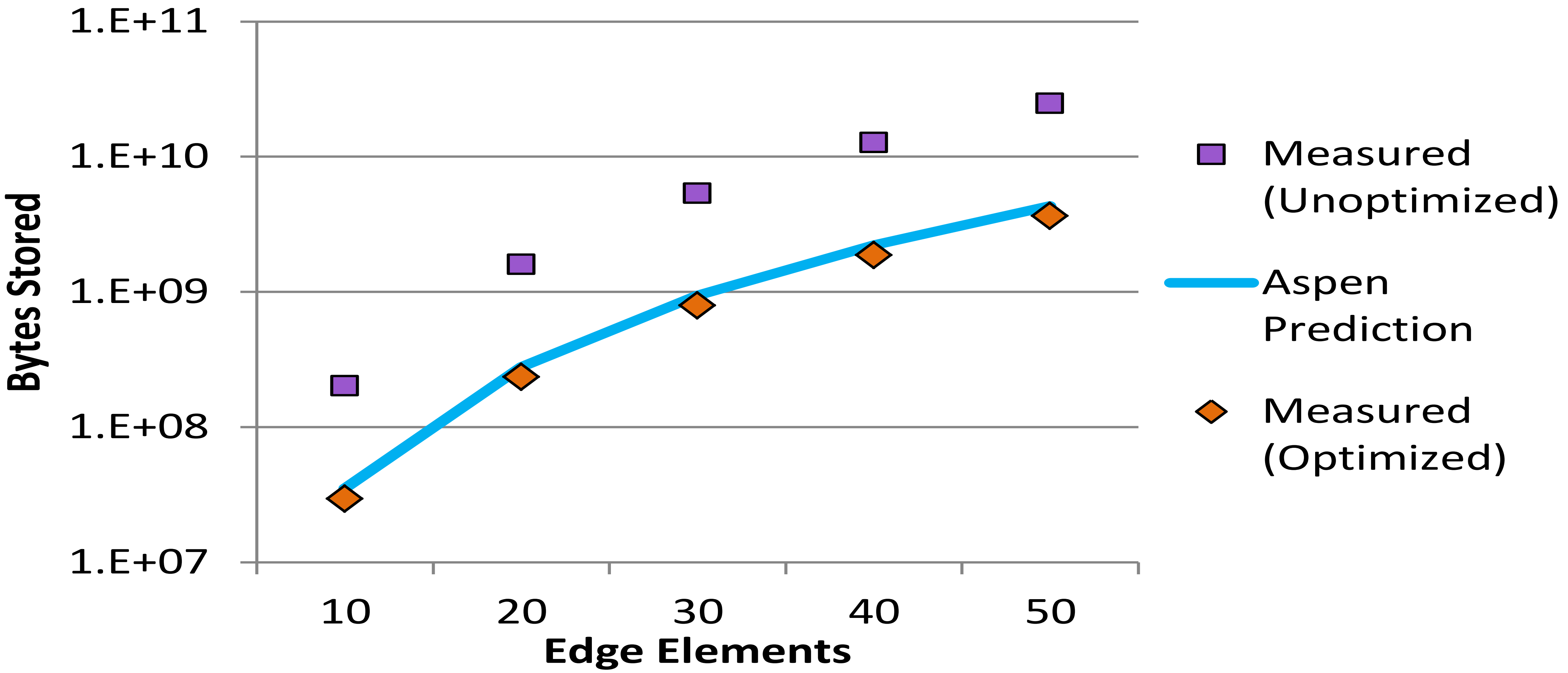
- Input LULESH program: 3700 lines of C codes
- Output Aspen model: 2300 lines of Aspen codes

Model Validation

	FLOPS	LOADS	STORES
MATMUL	15%	<1%	1%
LAPLACE2D	7%	0%	<1%
SRAD	17%	0%	0%
JACOBI	6%	<1%	<1%
KMEANS	0%	0%	8%
LUD	5%	0%	2%
BFS	<1%	11%	0%
HOTSPOT	0%	0%	0%
LULESH	0%	0%	0%

0% means that prediction fell between measurements from optimized and unoptimized runs of the code.

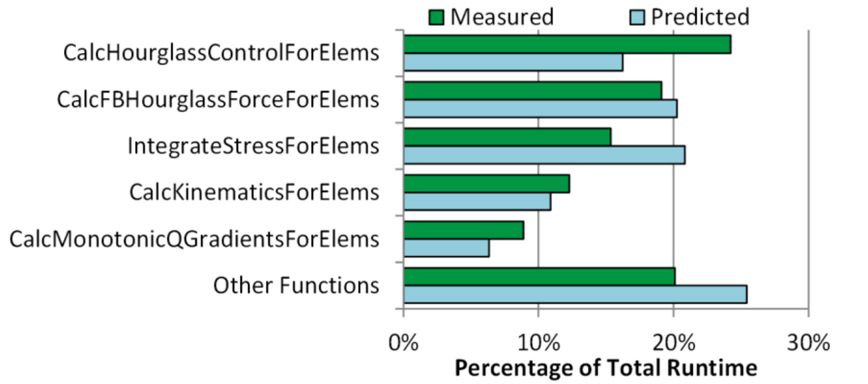
Model Scaling Validation (LULESH)



Example Queries

Benchmark	Runtime Order
BACKPROP	$H * O + H * I$
BFS	$nodes + edges$
CFD	$nelr * ndim$
CG	$nrow + ncol$
HOTSPOT	$sim_time * rows * cols$
JACOBI	$m\_size * m\_size$
KMEANS	$nAttr * nClusters$
LAPLACE2D	n^2
LUD	$matrix\_dim^3$
MATMUL	$N * M * P$
NW	max_cols^2
SPMUL	$size + nonzero$
SRAD	$niter * rows * cols$

Table 2: Order analysis, showing Big O runtime for each benchmark in terms of its key parameters.



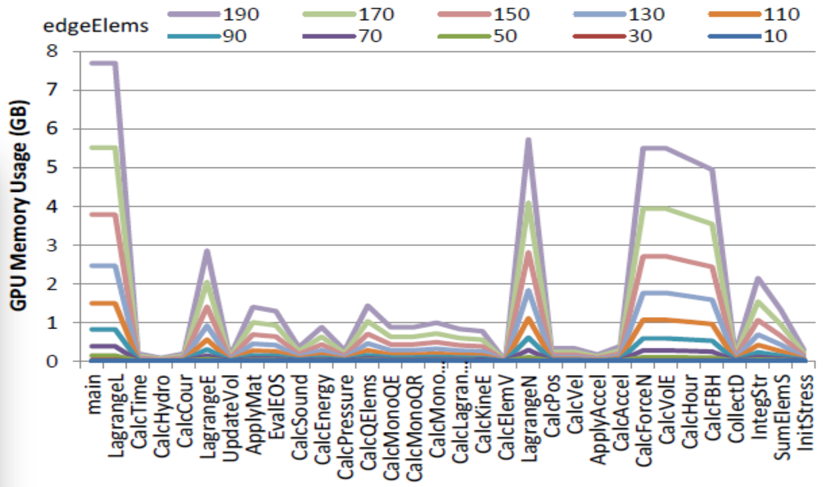


Fig. 8: GPU Memory Usage of each Function in LULESH, where the memory usage of a function is inclusive; value for a parent function includes data accessed by its child functions in the call graph.

Method Name	FLOPS/byte
InitStressTermsForElems	0.03
CalcElemShapeFunctionDerivatives	0.44
SumElemFaceNormal	0.50
CalcElemNodeNormals	0.15
SumElemStressesToNodeForces	0.06
IntegrateStressForElems	0.15
CollectDomainNodesToElemNodes	0.00
VoluDer	1.50
CalcElemVolumeDerivative	0.33
CalcElemFBHourglassForce	0.15
CalcFBHourglassForceForElems	0.17
CalcHourglassControlForElems	0.19
CalcVolumeForceForElems	0.18
CalcForceForNodes	0.18
CalcAccelerationForNodes	0.04
ApplyAccelerationBoundaryCond	0.00
CalcVelocityForNodes	0.13
CalcPositionForNodes	0.13
LagrangeNodal	0.18
AreaFace	10.25
CalcElemCharacteristicLength	0.44
CalcElemVelocityGrandient	0.13
CalcKinematicsForElems	0.24
CalcLagrangeElements	0.24
CalcMonotonicQGradientsForElems	0.46

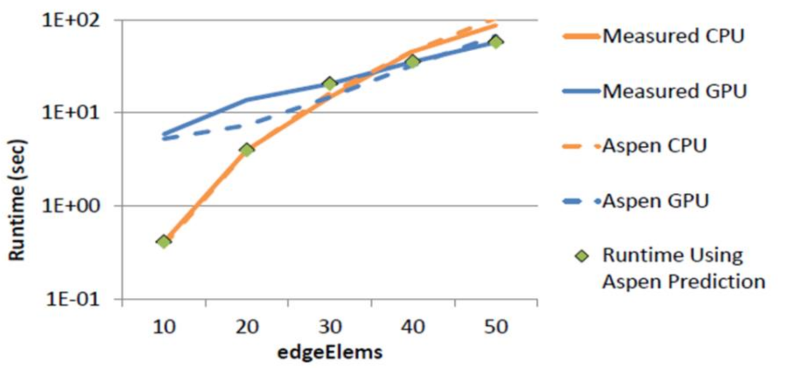


Fig. 7: Measured and predicted runtime of the entire LULESH program on CPU and GPU, including measured runtimes using the automatically predicted optimal target device at each size.

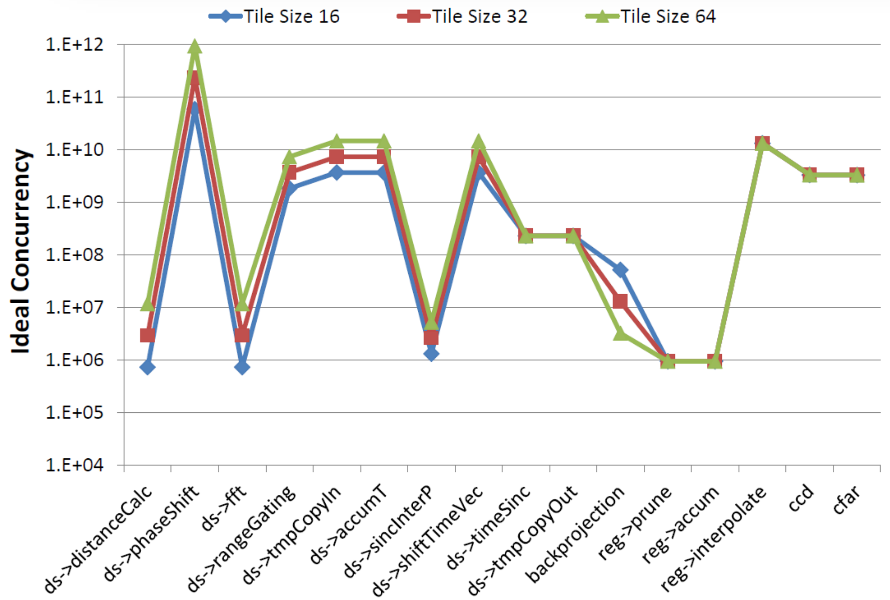
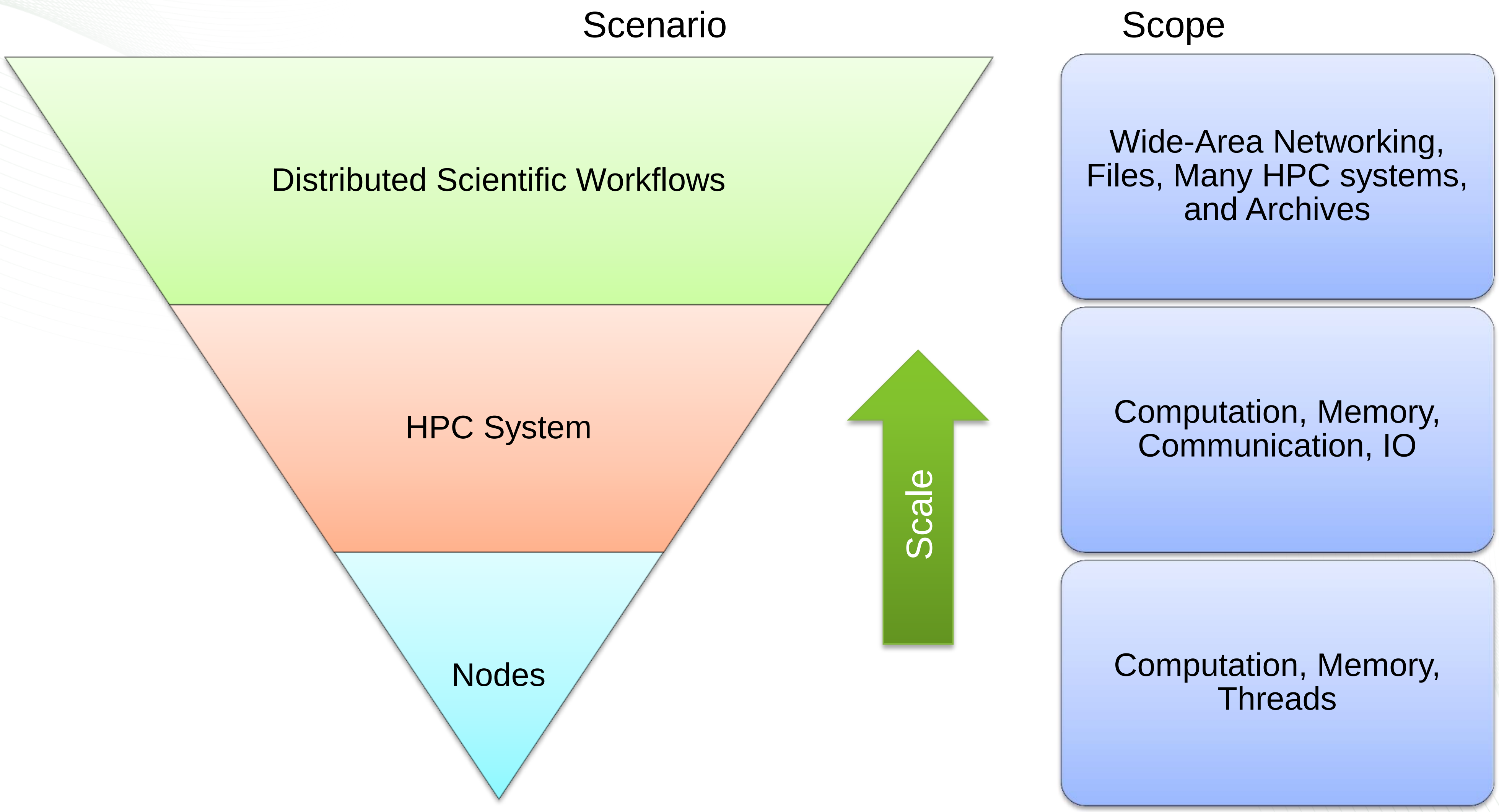


Figure 1: A plot of idealized concurrency by chronological phase in the digital spotlighting application model.

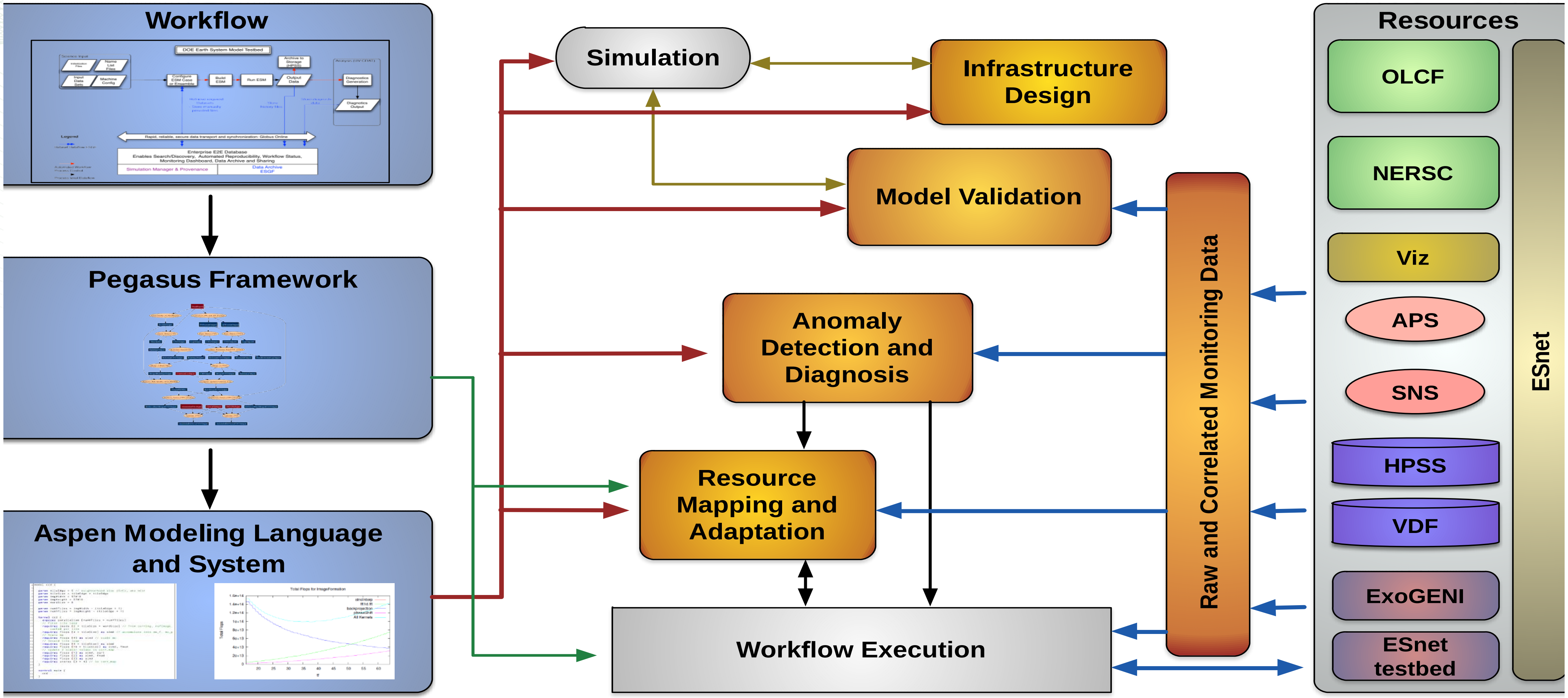
Performance Modeling for Distributed Scientific Workflows

OAK RIDGE National Laboratory

Aspen allows Multiresolution Modeling



PANORAMA Overview



E. Deelman, C. Carothers *et al., "PANORAMA: An Approach to Performance Modeling and Diagnosis of Extreme Scale Workflows," International Journal of High Performance Computing Applications, (to appear), 2015,*

Workflo
ACME
Climate
Modelin

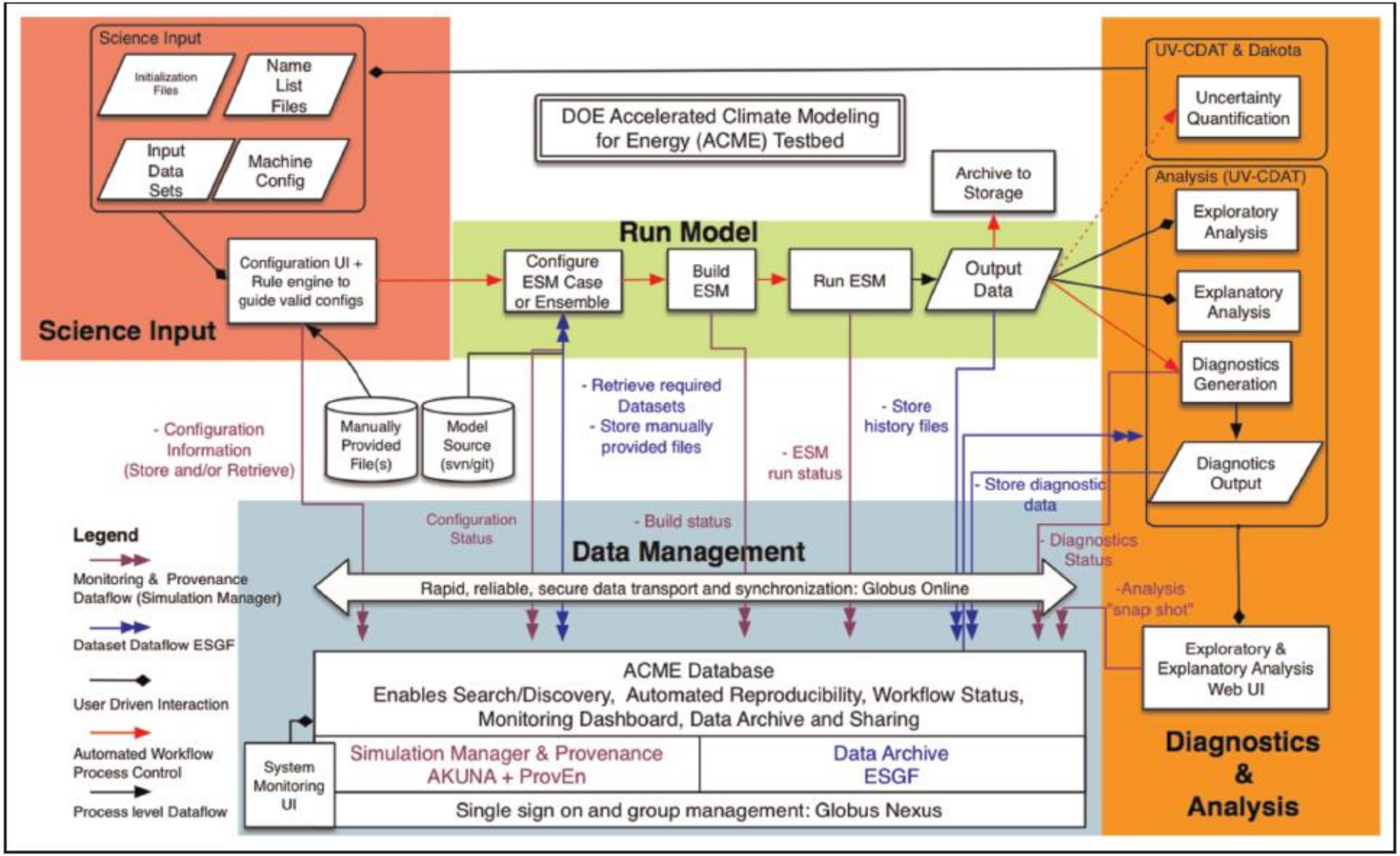
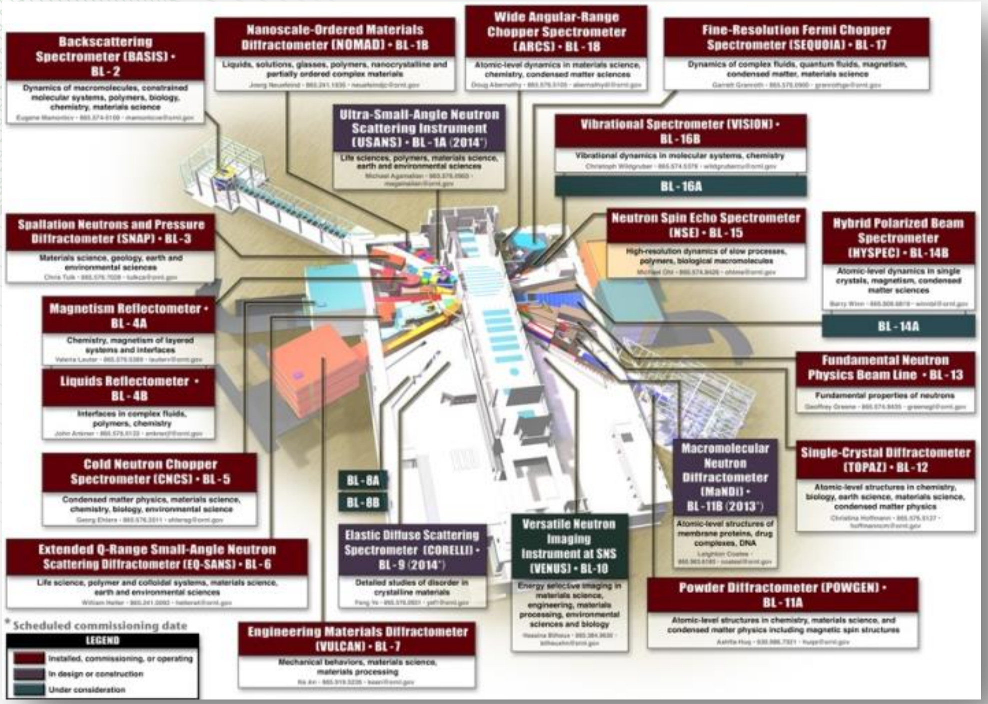
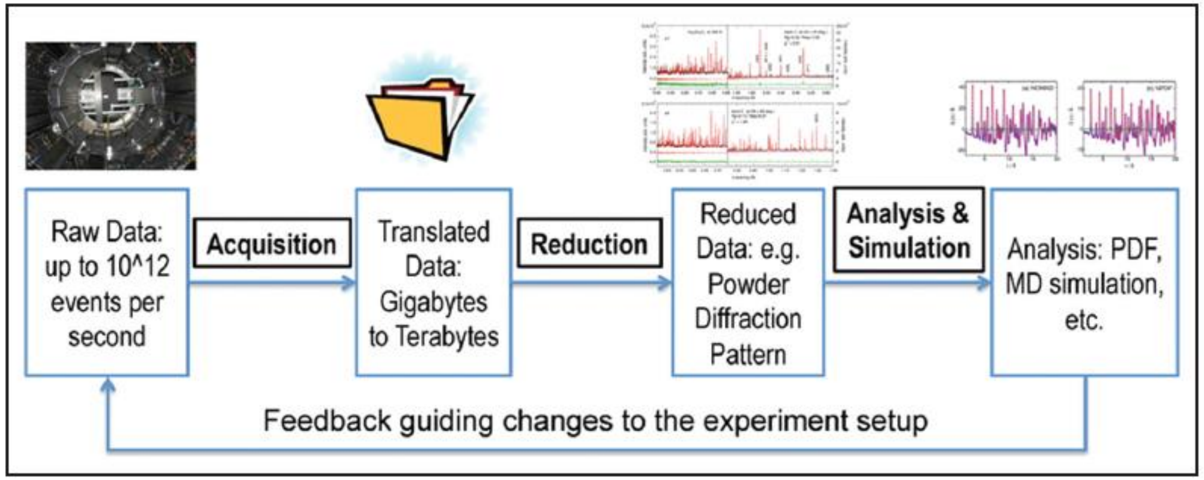


Figure 3: The complete Accelerated Climate Modeling for Energy (ACME) includes many interacting components distributed across DOE labs.

Workflow: SNS





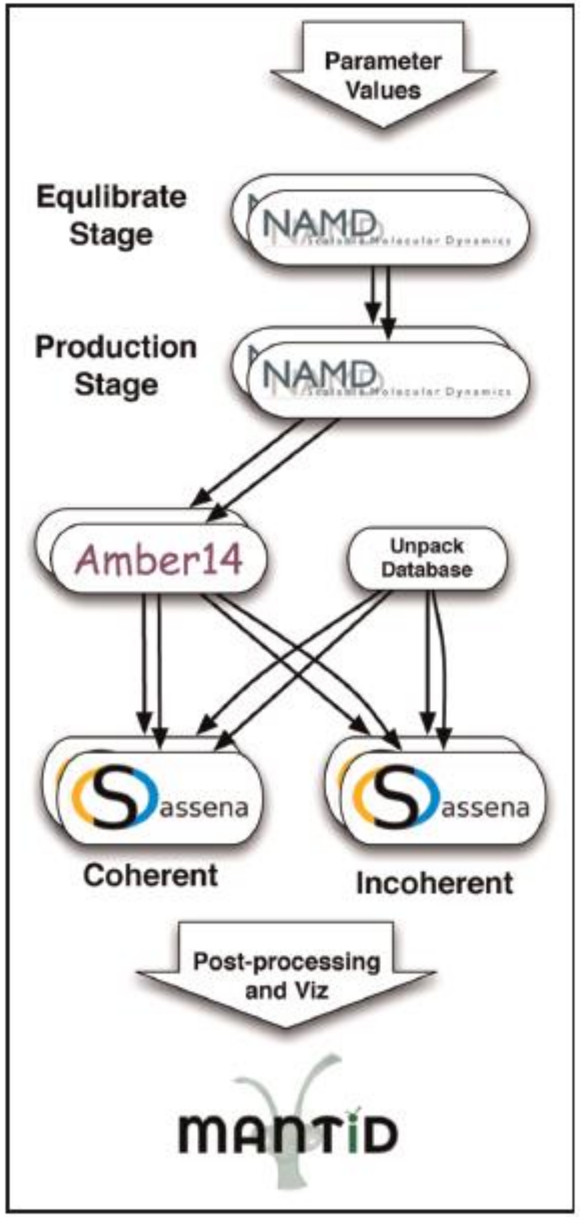
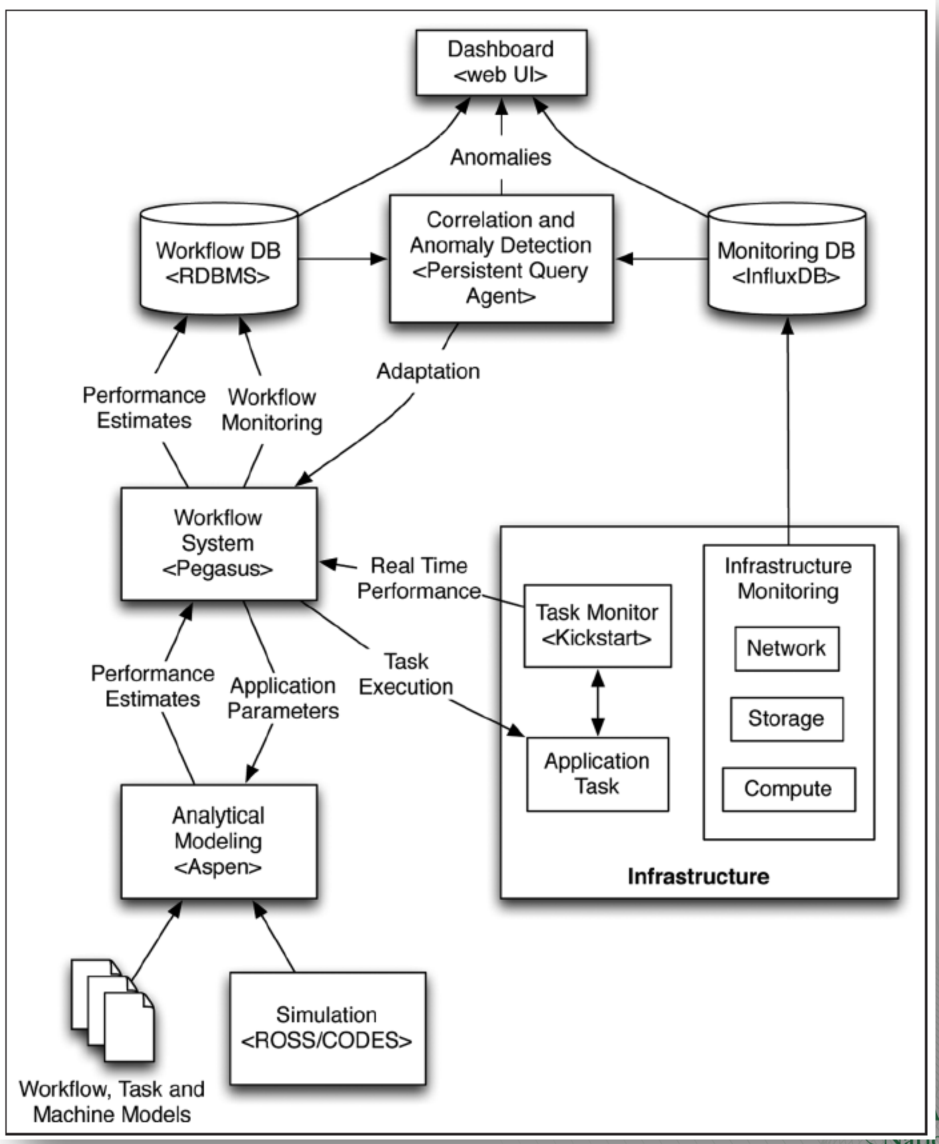


Figure 2: The SNS refinement workflow executes a parameter sweep of molecular dynamics and neutron scattering simulations to optimize the value for a target parameter to fit experimental data.

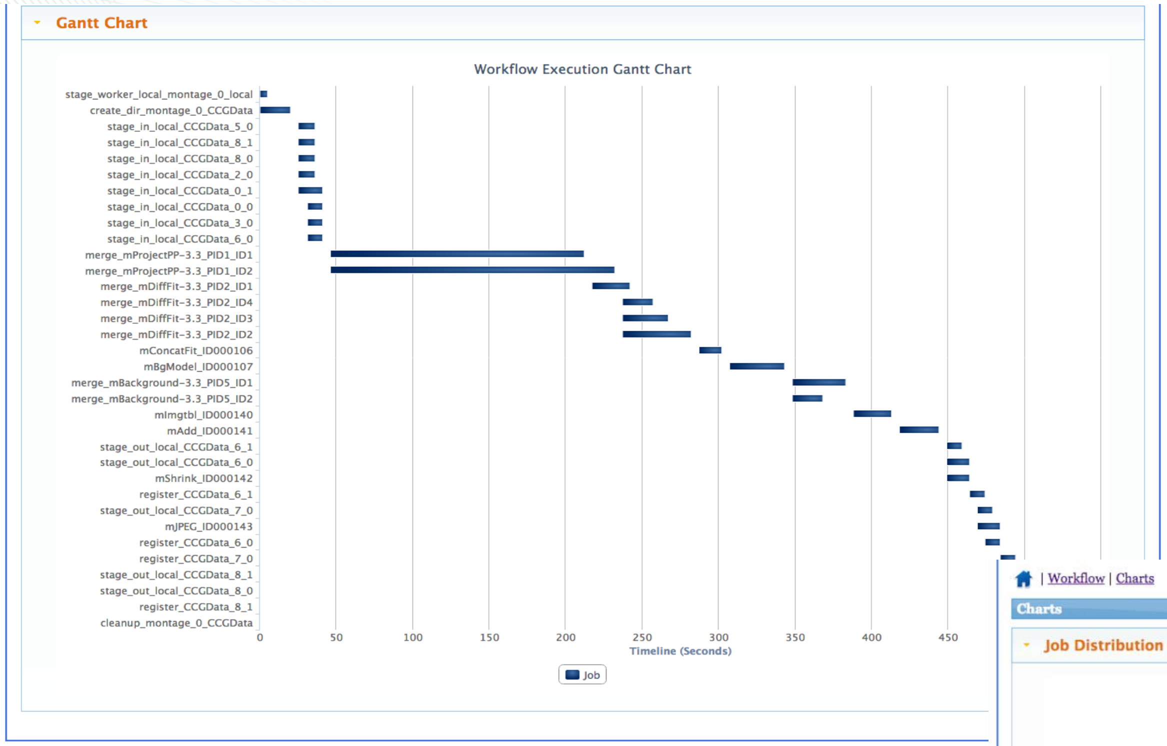
Automatically Generate Aspen from Pegasus DAX; Use Aspen Predictions to Inform/Monitor Decisions

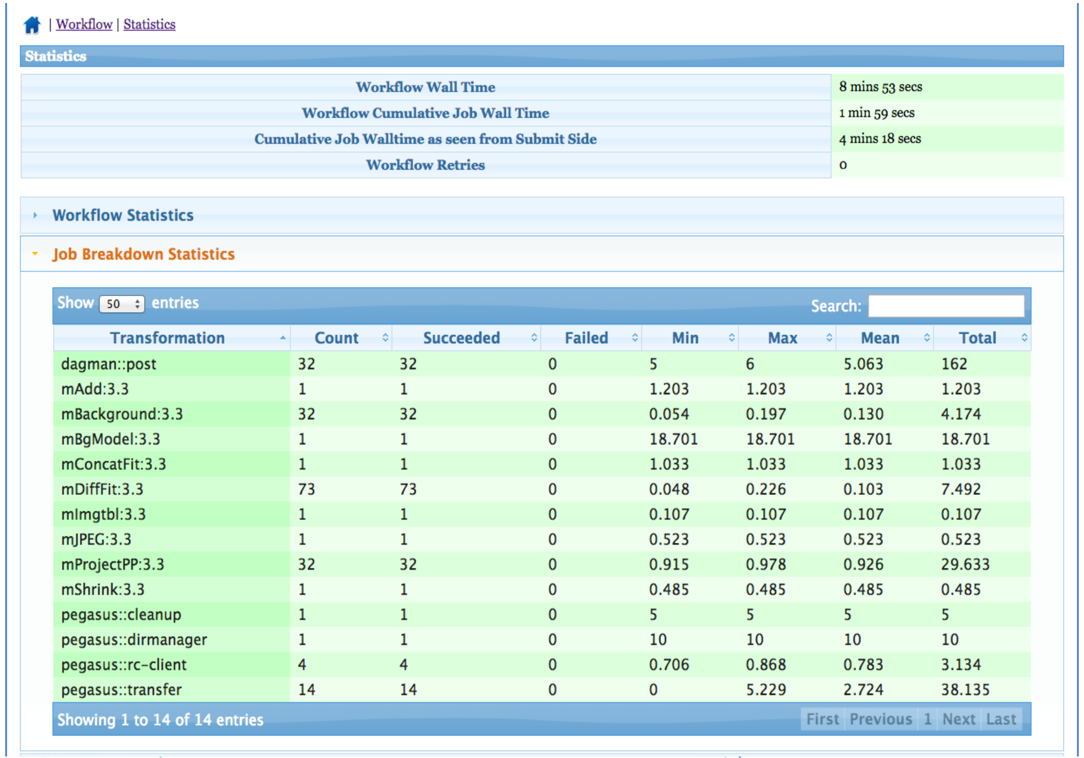
```
kernel main
{
    par {
        seq {
            call namd_eq_200()
            call namd_prod_200()
        }
        seq {
            call namd_eq_290()
            call namd_prod_290()
        }
    }
    par {
        call unpack_database()
        call ptraj_200()
        call ptraj_290()
    }
    par {
        call sassena_incoh_200()
        call sassena_coh_200()
        call sassena_incoh_290()
        call sassena_coh_290()
    }
}
```

Listing 1: Automatically generated Aspen model for
xample SNS workflow.



Workflow Monitoring Dashboard – *pegasus-dashboard*




| Workflow | Statistics

Statistics

Workflow Wall Time	8 mins 53 secs
Workflow Cumulative Job Wall Time	1 min 59 secs
Cumulative Job Walltime as seen from Submit Side	4 mins 18 secs
Workflow Retries	0

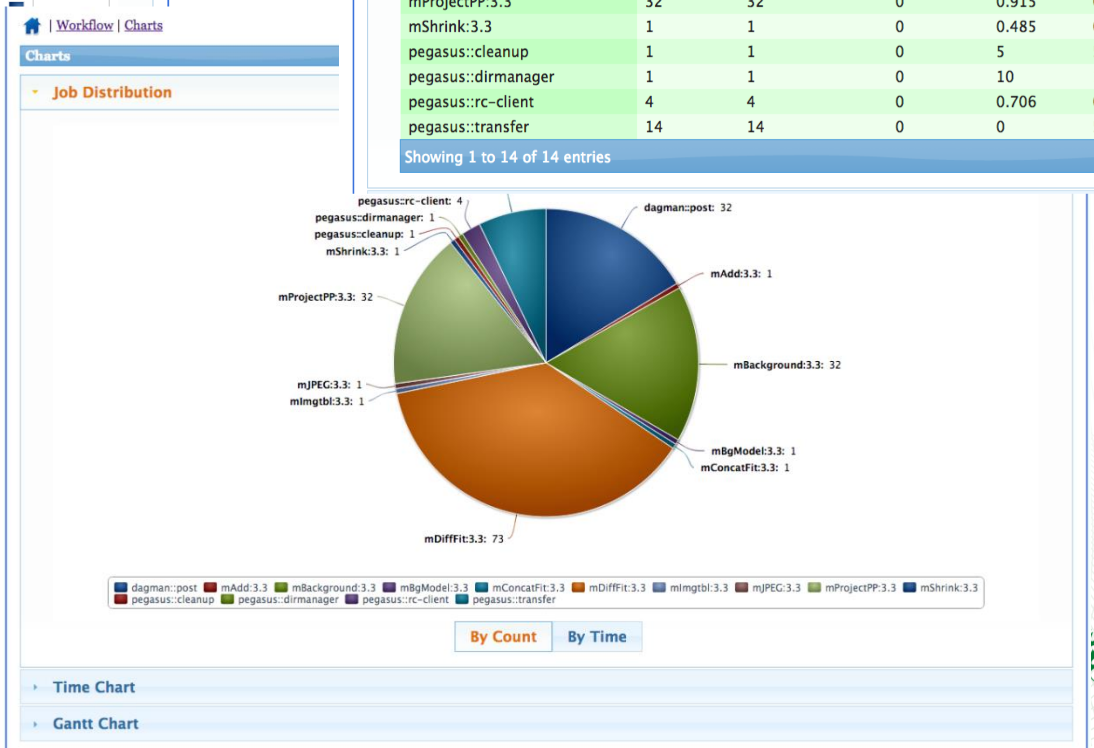
Workflow Statistics

Job Breakdown Statistics

Show 50 entries Search:

Transformation	Count	Succeeded	Failed	Min	Max	Mean	Total
dagman::post	32	32	0	5	6	5.063	162
mAdd:3.3	1	1	0	1.203	1.203	1.203	1.203
mBackground:3.3	32	32	0	0.054	0.197	0.130	4.174
mBgModel:3.3	1	1	0	18.701	18.701	18.701	18.701
mConcatFit:3.3	1	1	0	1.033	1.033	1.033	1.033
mDiffFit:3.3	73	73	0	0.048	0.226	0.103	7.492
mImgtbl:3.3	1	1	0	0.107	0.107	0.107	0.107
mJPEG:3.3	1	1	0	0.523	0.523	0.523	0.523
mProjectPP:3.3	32	32	0	0.915	0.978	0.926	29.633
mShrink:3.3	1	1	0	0.485	0.485	0.485	0.485
pegasus::cleanup	1	1	0	5	5	5	5
pegasus::dirmanager	1	1	0	10	10	10	10
pegasus::rc-client	4	4	0	0.706	0.868	0.783	3.134
pegasus::transfer	14	14	0	0	5.229	2.724	38.135

Showing 1 to 14 of 14 entries First Previous 1 Next Last



Status, statistics, timeline of jobs

Helps pinpoint errors

OAK RIDGE National Laboratory

End-to-end Resiliency Design using Aspen

Data Vulnerability Factor: Why a new metric and methodology?

- Analytical model of resiliency that includes important features of architecture and application
  - Fast
  - Flexible
- Balance multiple design dimensions
  - Application requirements
  - Architecture (memory capacity and type)
- Focus on main memory initially
- Prioritize vulnerabilities of application data

L. Yu, D. Li *et al.*, *"Quantitatively modeling application resilience with the data vulnerability factor (Best Student Paper Finalist)," in SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans, Louisiana: IEEE Press, 2014, pp. 695-706, 10.1109/sc.2014.62.*

DVF Defined

Data Structure Vulnerability $\rightarrow$ $DVF_d = \boxed{N_{error}} * \boxed{N_{ha}}$

Application Vulnerability $\rightarrow$ $DVF_a = \sum_{i=1}^{n} DVF_{d_i}$

Larger DVF indicates higher vulnerability, and vice versa

$N_{error} = FIT * T * S_d$

We focus on a specific hardware component, the main memory, in this work

$N_{ha} \leftarrow$ Hardware Access Pattern

Implementing DVF

- Extend Aspen performance modeling language
- Specify memory access patterns
- Combine error rates with memory regions and performance
- Assign DVF to each application memory region, Sum for application

Workflow to calculate Data Vulnerability Factor

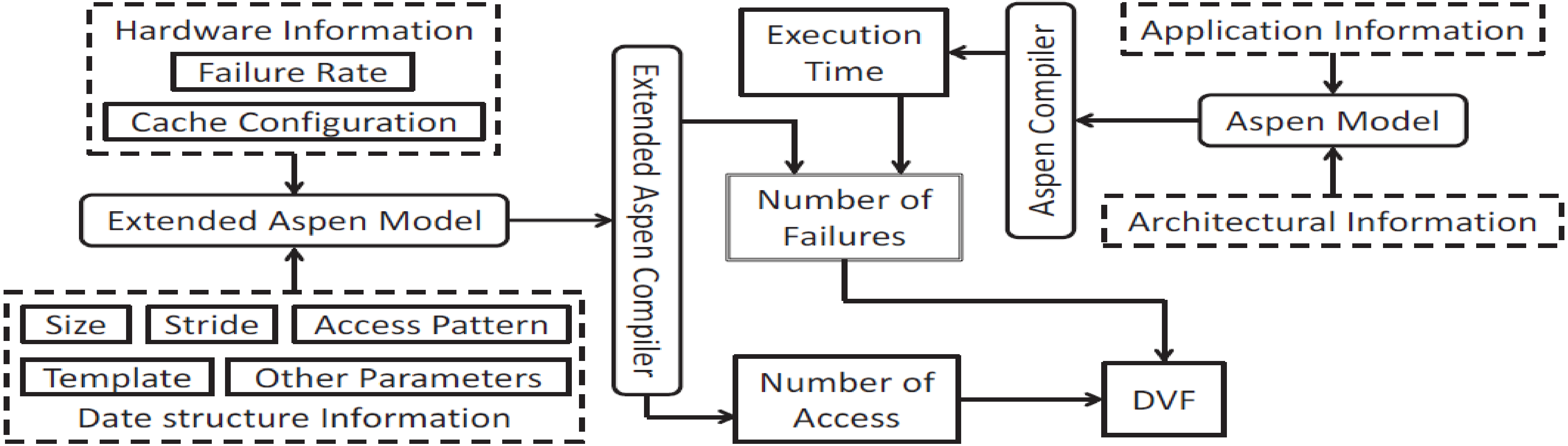


Fig. 3. The workflow to calculate DVF.

An Example of Aspen Program for DVF

procedure VM(A,B,C)
    for i ← 1, 1000 **do**
        C[i] ← C[i] + A[i*4] * B[i*8]
    end for
end procedure

Pseudocode

```
kernel vecmul {
    execute mainblock2 [1]
    {
    flops [2*(n^3)] as sp, fmad, simd
    access {1000} from {matA} as stream(4,16)
    access {4000} from {matB} as stream(4,32)
    access {8000} from {matC} as stream(4,4)
    }
}
```

Extended Aspen Statements

Extended Parser

```
Resilience Statements:
    Footprint Sizes:
        Int: 16,000
    Data Structures:
        Ident: matA
    Access Pattern: Stream
        Int: 4
        Int: 16
Resilience Statements:
    Footprint Sizes:
        Int: 16,000
    Data Structures:
        Ident: matA
    Access Pattern: Stream
        Int: 4
        Int: 16
Resilience Statements:
    Footprint Sizes:
        Int: 16,000
    Data Structures:
        Ident: matA
    Access Pattern: Stream
        Int: 4
        Int: 16
```

Syntax Tree

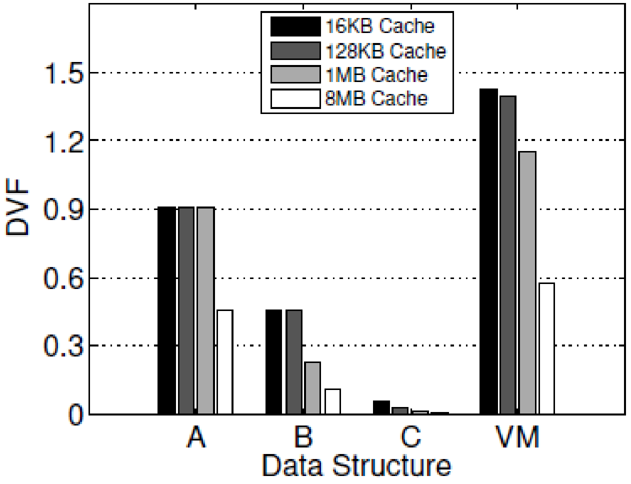
Extended Complier

Data structure A:
Number of errors: 30,400
Number of memory accesses: 51
DVF: 105504e+06
…

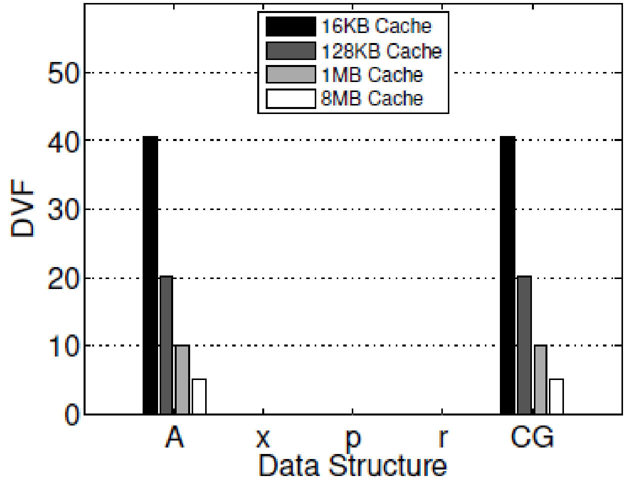
Resilience Modeling Results

DVF Results

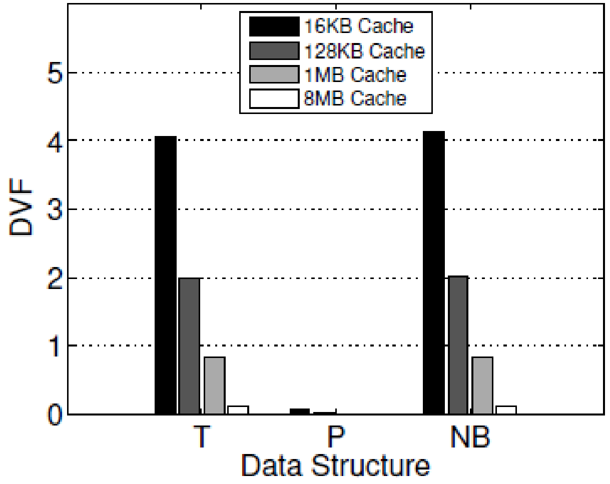
Provides insight for balancing interacting factors



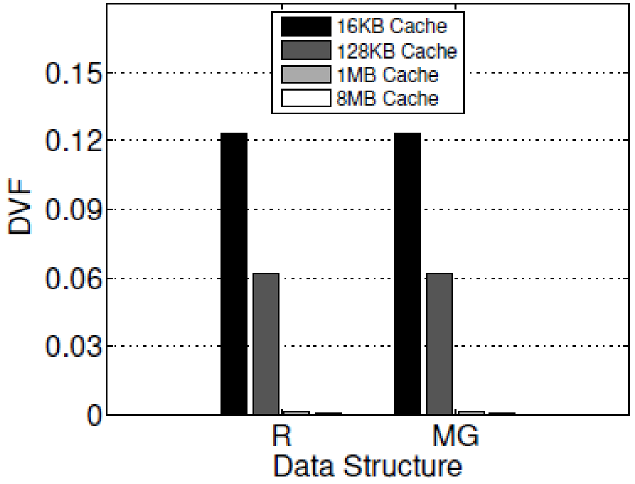
(a) Vector Multiplication



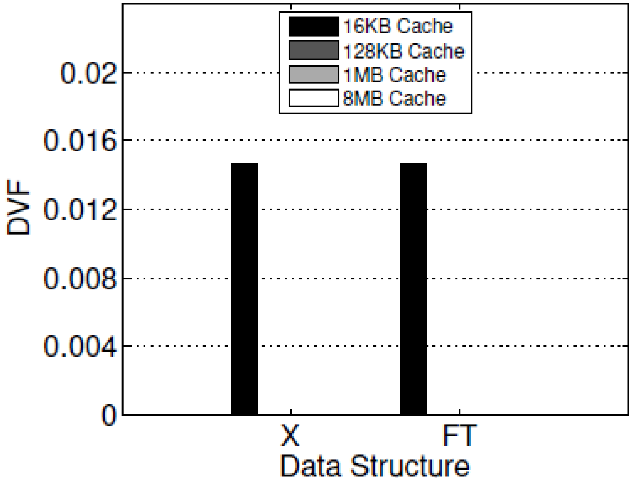
(b) Conjugate Gradient



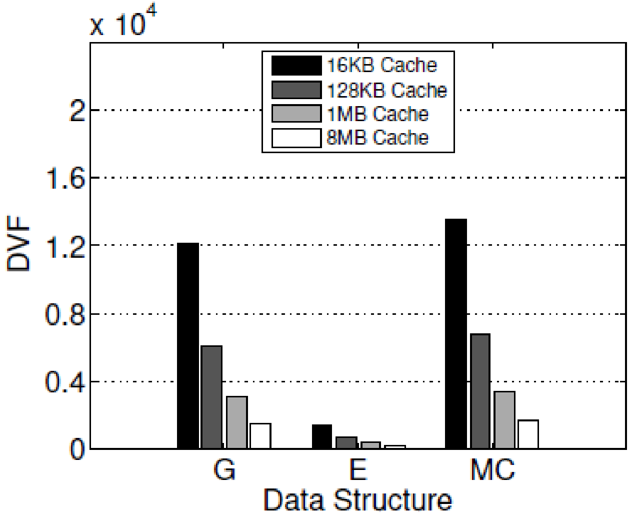
(c) Nbody (Barnes-hut)



(d) Multi-grid



(e) 1D FFT



(f) Monte Carlo

DVF: next steps

- Evaluated different architectures
  – How much no-ECC, ECC, NVM?
- Evaluate software and applications
  – ABFT
  – C/R
  – TMR
  – Containment domains
  – Fault tolerant MPI
- End-to-End analysis
  – Where should we bear the cost for resiliency?
    - Not everwhere!

OAK RIDGE National Laboratory

37

Summary

- Our community has major challenges in HPC as we move to extreme scale
  - Power, Performance, Resilience, Productivity
  - New technologies emerging to address some of these challenges
    - Heterogeneous computing
    - Nonvolatile memory
  - Not just HPC: Most uncertainty in at least two decades
- We need performance prediction and engineering tools now more than ever!
- Aspen is a tool for structured design and analysis
  - Co-design applications and architectures for performance, power, resiliency
  - Automatic model generation
  - Scalable to distributed scientific workflows
  - DVF – a new twist on resiliency modeling

Acknowledgements

- Contributors and Sponsors
  - Future Technologies Group: http://ft.ornl.gov
  - US Department of Energy Office of Science
    - DOE Vancouver Project: https://ft.ornl.gov/trac/vancouver
    - DOE Blackcomb Project: https://ft.ornl.gov/trac/blackcomb
    - DOE ExMatEx Codesign Center: http://codesign.lanl.gov
    - DOE Cesar Codesign Center: http://cesar.mcs.anl.gov/
    - DOE Exascale Efforts: http://science.energy.gov/ascr/research/computer-science/
  - Scalable Heterogeneous Computing Benchmark team: http://bit.ly/shocmarx
  - US National Science Foundation Keeneland Project: http://keeneland.gatech.edu
  - US DARPA
  - NVIDIA CUDA Center of Excellence

OAK RIDGE
National Laboratory

Notional Exascale Architecture Targets
(From Exascale Arch Report 2009)

System attributes	2001	2010	"2015"		"2018"	
System peak	10 Tera	2 Peta	200 Petaflop/sec		1 Exaflop/sec	
Power	~0.8 MW	6 MW	15 MW		20 MW	
System memory	0.006 PB	0.3 PB	5 PB		32-64 PB	
Node performance	0.024 TF	0.125 TF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW		25 GB/s	0.1 TB/sec	1 TB/sec	0.4 TB/sec	4 TB/sec
Node concurrency	16	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	416	18,700	50,000	5,000	1,000,000	100,000
Total Node Interconnect BW		1.5 GB/s	150 GB/sec	1 TB/sec	250 GB/sec	2 TB/sec
MTTI		day	O(1 day)		O(1 day)	

Parallel I/O ??

http://science.energy.gov/ascr/news-and-resources/workshops-and-conferences/grand-challenges/

Today's Status

System attributes	Today		CORAL	
Name	**TITAN**	**MIRA**	**Summit**	**Aurora**
System peak (PF)	27	10	150	180
Peak Power (MW)	9	4.8	10	13
Total system memory	710TB	768TB	2 PB DDR4 + HBM + 2.7 PB persistent memory	>7 PB High Bandwidth On-Package Memory, local Memory and Persistent Memory
Node performance (TF)	1.452	0.204	> 40	> 17 times Mira
Node processors	AMD Opteron Nvidia Kepler	64-bit PowerPC A2	Multiple IBM Power9 CPUs & multiple Nvidia Voltas GPUS	Intel Xeon Phi processors (codenamed Knights Hill)
System size (nodes)	18,688 nodes	49,152	>3,400 nodes	>50,000 nodes
System Interconnect	Gemini	5D Torus	Dual Rail EDR-IB	2nd generation Intel Omni-Path Architecture
File System	32 PB 1 TB/s, Lustre®	26 PB 300 GB/s GPFS™	120 PB 1 TB/s GPFS™	150 PB >1 TB/s Lustre®

(Un-)Balanced Systems ??

System attributes	2001	2010	2014	"2015"		est 2018	Summit/Titan	"2018"	
Name	Seaborg3	Jaguar	Titan			SUMMIT			
System peak	10 Tera	2	27	200		136	5.0	1 Exaflop/sec	
Power (MW)	0.8	6	9	15		10	1.1	20	
Node main memory (GB)		16	38			512	13.5		
System memory (PB)	0.006	0.3	0.7106	5		1.7408	2.4	32-64	
Node Persistent Memory (GB)						800	inf		
System Persistent Memory (PB)						2.72	inf		
Node performance (TF)	0.024	0.125	1.4	0.5	7	40	28.6	1	10
Node memory BW		25 GB/s		0.1 TB/sec	1 TB/sec			0.4 TB/sec	4 TB/sec
Node concurrency	16	12		O(100)	O(1,000)	*POWER9s + *VOLTAs		O(1,000)	O(10,000)
System size (nodes)	416	18700	18700	50000	5000	3400	0.2	1000000	100000
Total Node Interconnect BW (GB/s)		1.5 GB/s		150 GB/sec	1 TB/sec			250 GB/sec	2 TB/sec
injection bandwidth per node (GB/s)		7.6	20			23	1.2		
File system capacity (PB)		6	32			120	3.8		
File system bandwidth (TB/s)		0.3	1			1	1.0		
MTTI		day		O(1 day)				O(1 day)	

- Power is constant
- 1/5 of the node count
- Heterogeneous
- I/O and NIC bandwidth has plateaued
- NVM is new!

Oak Ridge National Laboratory