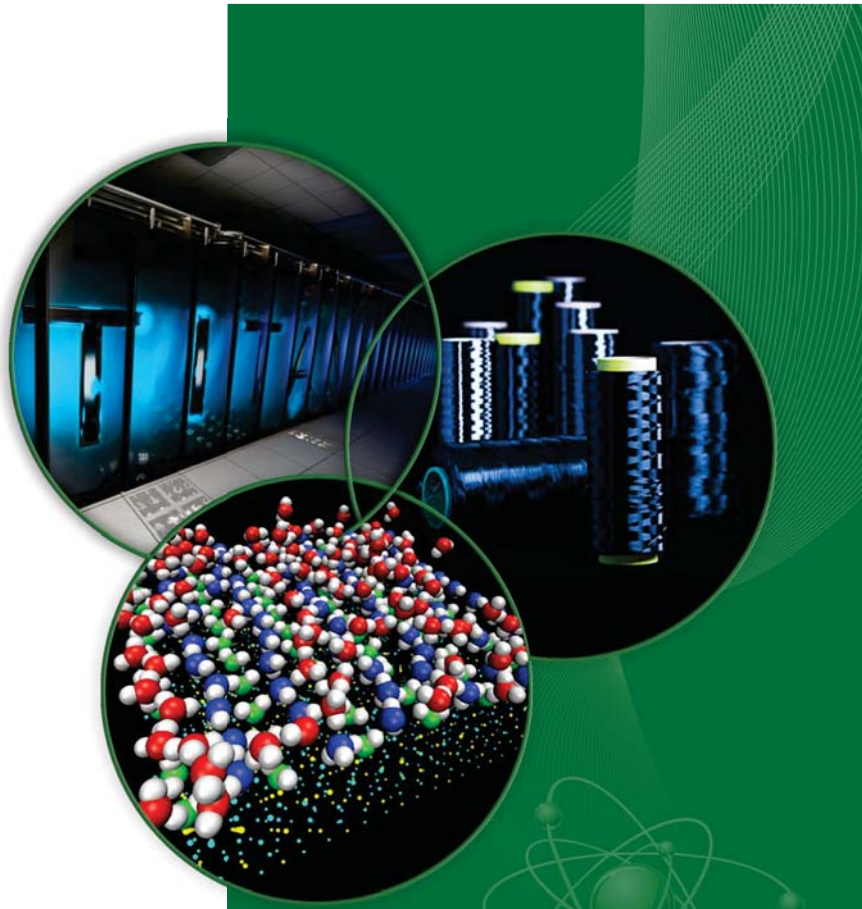
Exploring Emerging Technologies in the Extreme Scale HPC Co-Design Space with Aspen

Jeffrey S. Vetter

Jeremy Meredith

MODSIM Workshop 2015
Seattle
13 Aug 2015



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

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

Georgia Tech College of Computing
Computational Science and Engineering

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

OAK RIDGE National Laboratory

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

Model Creation

- Static analysis via compiler, tools
- Empirical, Historical
- Manual (for future applications)

Representation in Aspen

- **Modular**
- **Sharable**
- **Composable**
- **Reflects prog structure**

E.g., MD, UHPC CP 1, Lulesh, 3D FFT, CoMD, VPFFT, …

Model Uses

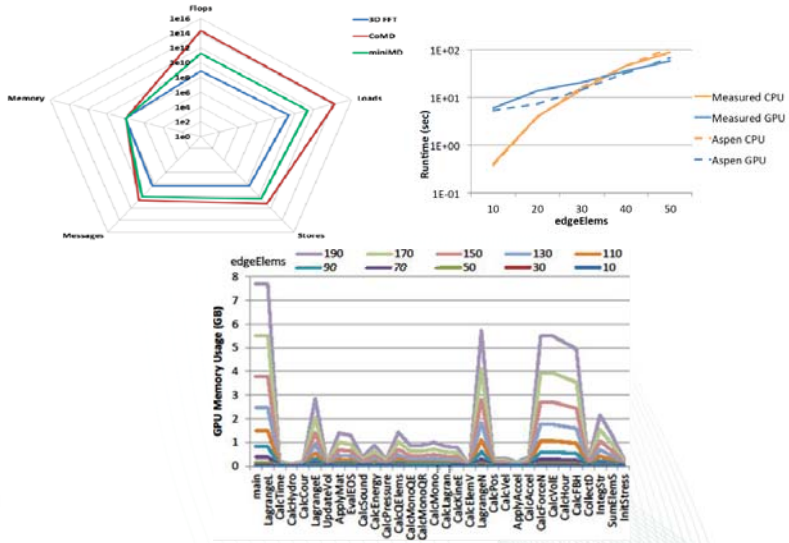
- Interactive tools for graphs, queries
- Design space exploration
- Workload Generation
- Feedback to Runtime Systems

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
  }
}
```

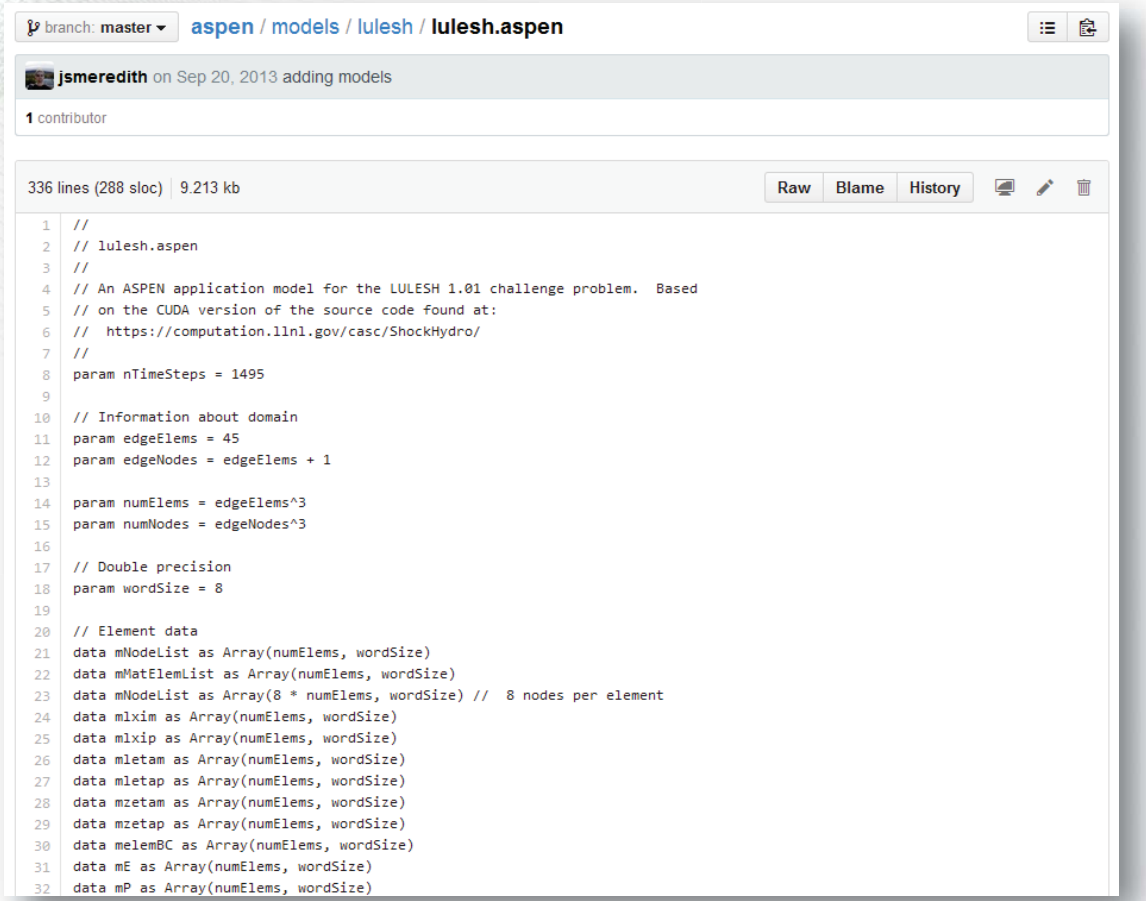


Researchers are using Aspen for parallel applications, scientific workflows, capacity planning, power, quantum computing, etc

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*

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

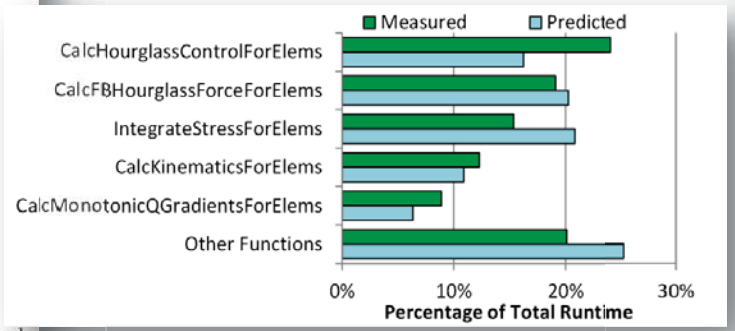
```
//
// lulesh.aspen
//
// An ASPEN application model for the LULESH 1.01 challenge problem.  Based
// on the CUDA version of the source code found at:
//  https://computation.llnl.gov/casc/ShockHydro/
//
param nTimeSteps = 1495

// 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)
```

```
147 kernel CalcMonotonicQGradients {
148   execute [numElems]
149   {
150     loads [8 * indexWordSize] from nodelist
151     // Load and cache position and velocity.
152     loads/caching [8 * wordSize] from x
153     loads/caching [8 * wordSize] from y
154     loads/caching [8 * wordSize] from z
155
156     loads/caching [8 * wordSize] from xvel
157     loads/caching [8 * wordSize] from yvel
158     loads/caching [8 * wordSize] from zvel
159
160     loads [wordSize] from volo
161     loads [wordSize] from vnew
162     // dx, dy, etc.
163     flops [90] as dp, simd
164     // delvk delxk
165     flops [9 + 8 + 3 + 30 + 5] as dp, simd
166     stores [wordSize] to delv_xeta
167     // delxi delvi
168     flops [9 + 8 + 3 + 30 + 5] as dp, simd
169     stores [wordSize] to delx_xi
170     // delxj and delvj
171     flops [9 + 8 + 3 + 30 + 5] as dp, simd
172     stores [wordSize] to delv_eta
173   }
174 }
```

Example Uses: Resource Exploration

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.



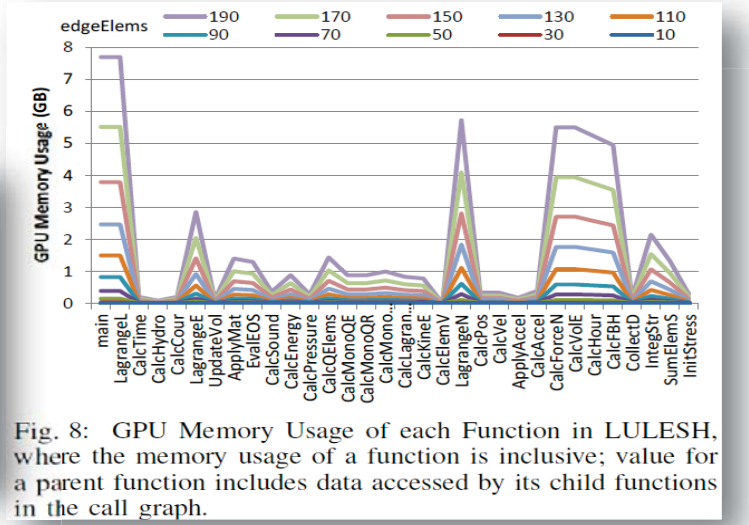


Fig. 8: GPU Memory Usage of each Function in LULESH, where the memory usage of a function is inclusive; value for a parent function includes data accessed by its child functions in the call graph.

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

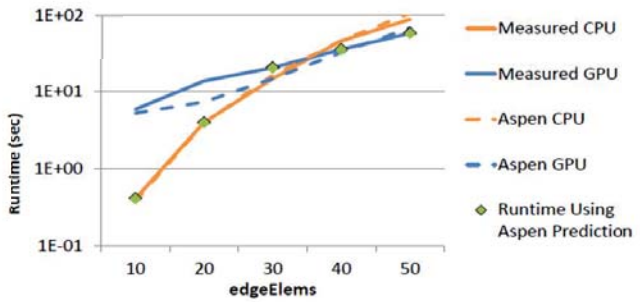


Fig. 7: Measured and predicted runtime of the entire LULESH program on CPU and GPU, including measured runtimes using the automatically predicted optimal target device at each size.

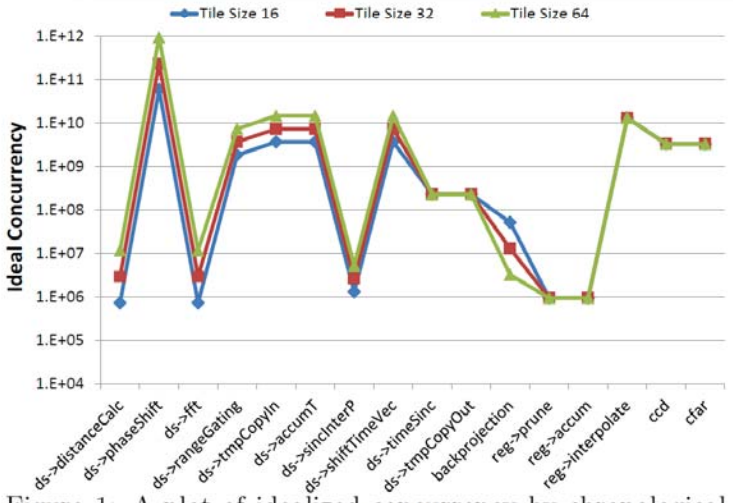
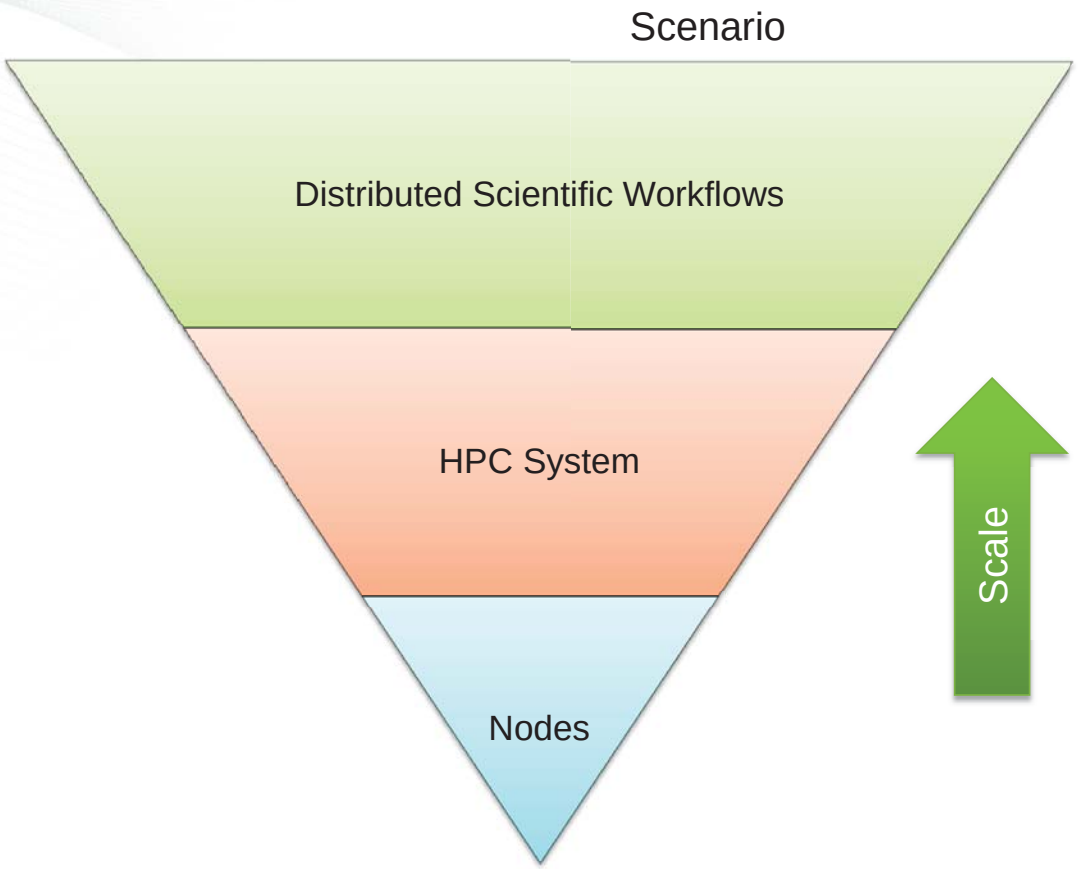


Figure 1: A plot of idealized concurrency by chronological phase in the digital spotlighting application model.

Aspen allows Multiresolution Modeling



Scope

Wide-Area Networking, Files, Many HPC systems, and Archives

Computation, Memory, Communication, IO

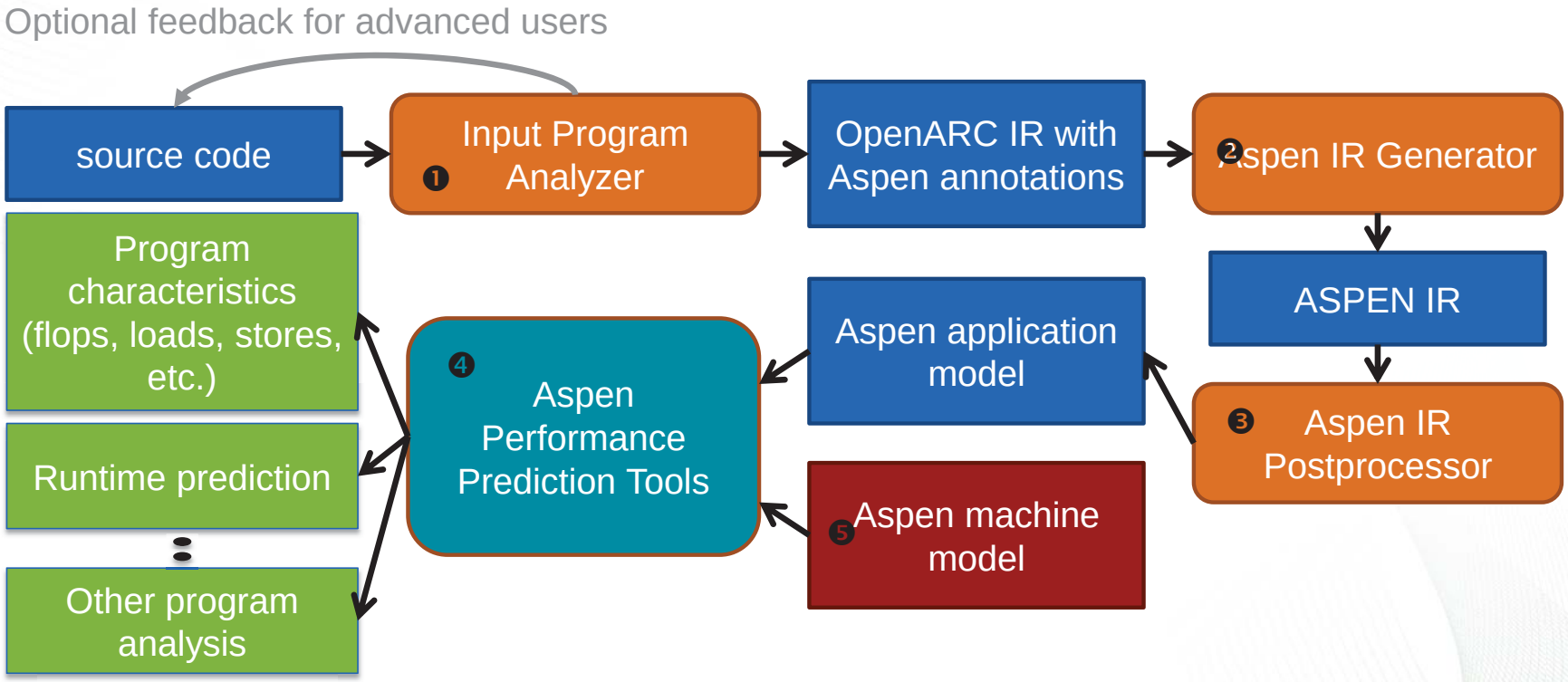
Computation, Memory, Threads

Node Scale Modeling with COMPASS

OAK RIDGE National Laboratory

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

```
1  int N = 1024;
2  void matmul(float *a, float *b, float *c){ int i, j, k ;
3  #pragma acc kernels loop gang copyout(a[0:(N*N)]) \
4  copyin(b[0:(N*N)],c[0:(N*N)])
5    for (i=0; i<N; i++){
6  #pragma acc loop worker
7      for (j=0; j<N; j++) { float sum = 0.0 ;
8        for (k=0; k<N; k++) {sum+=b[i*N+k]*c[k*N+j];}
9        a[i*N+j] = sum; }
10   } //end of i loop
11 } //end of matmul()
12 int main() {
13   int i; float *A = (float*) malloc(N*N*sizeof(float));
14   float *B = (float*) malloc(N*N*sizeof(float));
15   float *C = (float*) malloc(N*N*sizeof(float));
16   for (i = 0; i <  N*N; i++)
17   {  A[i] = 0.0F; B[i] = (float) i; C[i] = 1.0F; }
18 #pragma aspen modelregion label(MM)
19   matmul(A,B,C);
20   free(A); free(B); free(C); return 0;
21 } //end of main()
```

```
1  model MM {
2    param floatS = 4; param N = 1024
3    data A as Array((N*N), floatS)
4    data B as Array((N*N), floatS)
5    data C as Array((N*N), floatS)
6    kernel matmul {
7      execute matmul2_intracommIN
8      {  intracomm [floatS*(N*N)] to C as copyin
9        intracomm [floatS*(N*N)] to B as copyin }
10     map matmul2 [N] {
11       map matmul3 [N] {
12         iterate [N] {
13           execute matmul5
14           {  loads [floatS] from B as stride(1)
15             loads [floatS] from C; flops [2] as sp, simd }
16         } //end of iterate
17         execute matmul6 { stores [floatS] to A as stride(1) }
18       } // end of map matmul3
19     } //end of map matmul2
20     execute matmul2_intracommOUT
21     { intracomm [floatS*(N*N)] to A as copyout }
22   } //end of kernel matmul
23   kernel main { matmul() }
24 } //end of model MM
```

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)
    . . . . . .
```

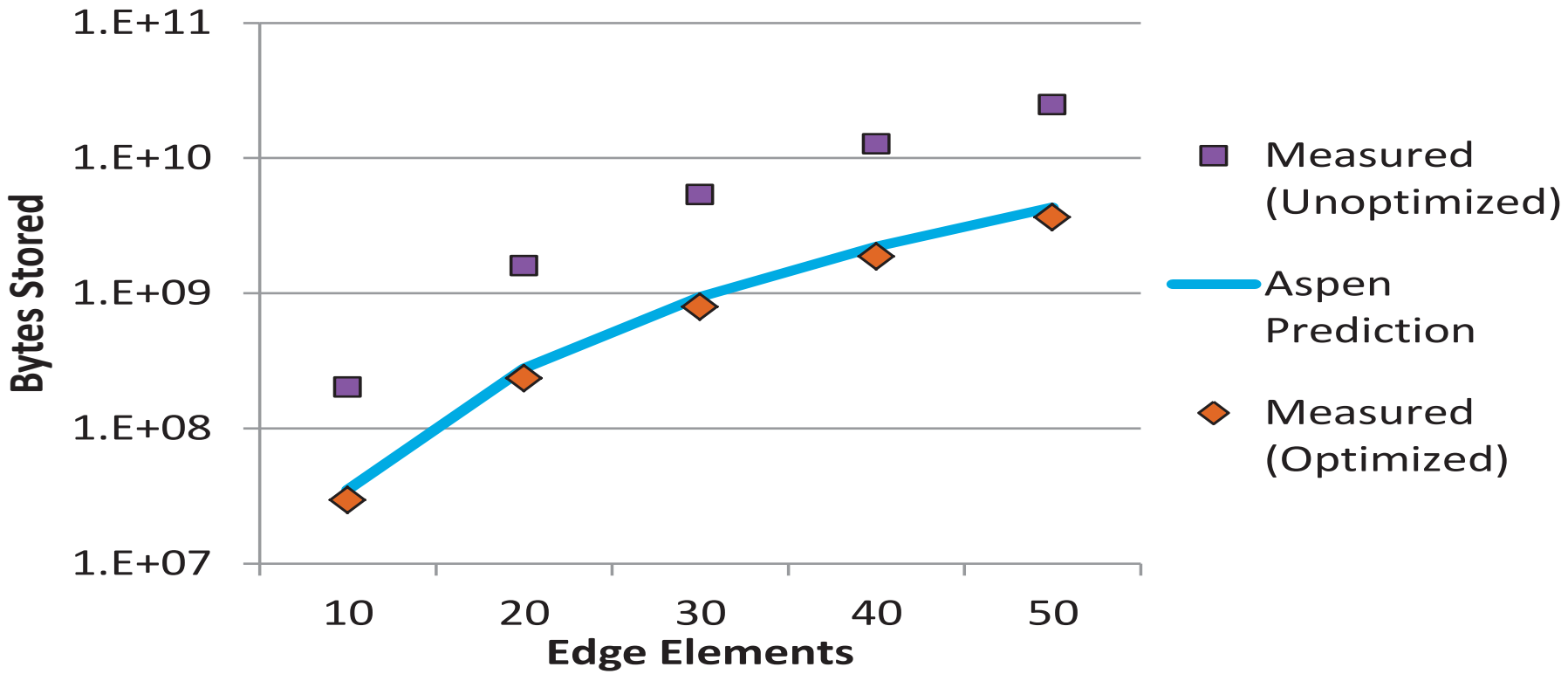
- Input LULESH program: 3700 lines of C codes
- Output Aspen model: 2300 lines of Aspen codes

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%

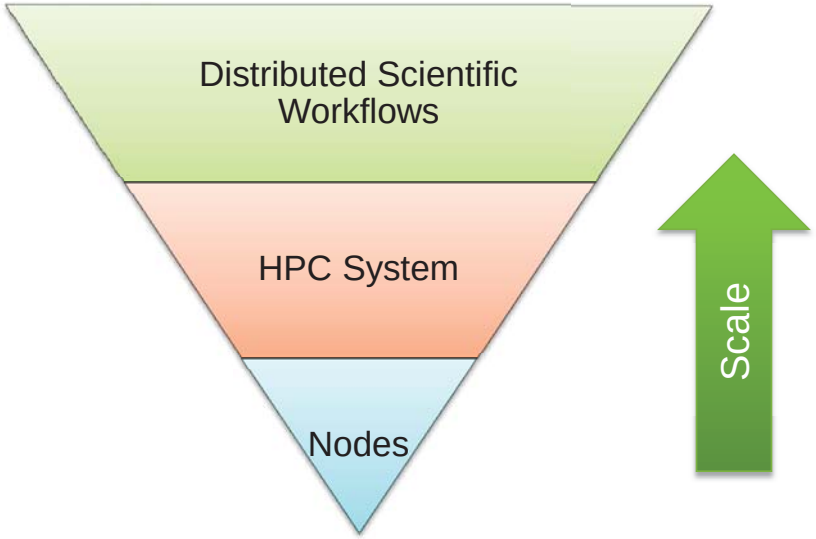
0% means that prediction fell between measurements from optimized and unoptimized runs of the code.

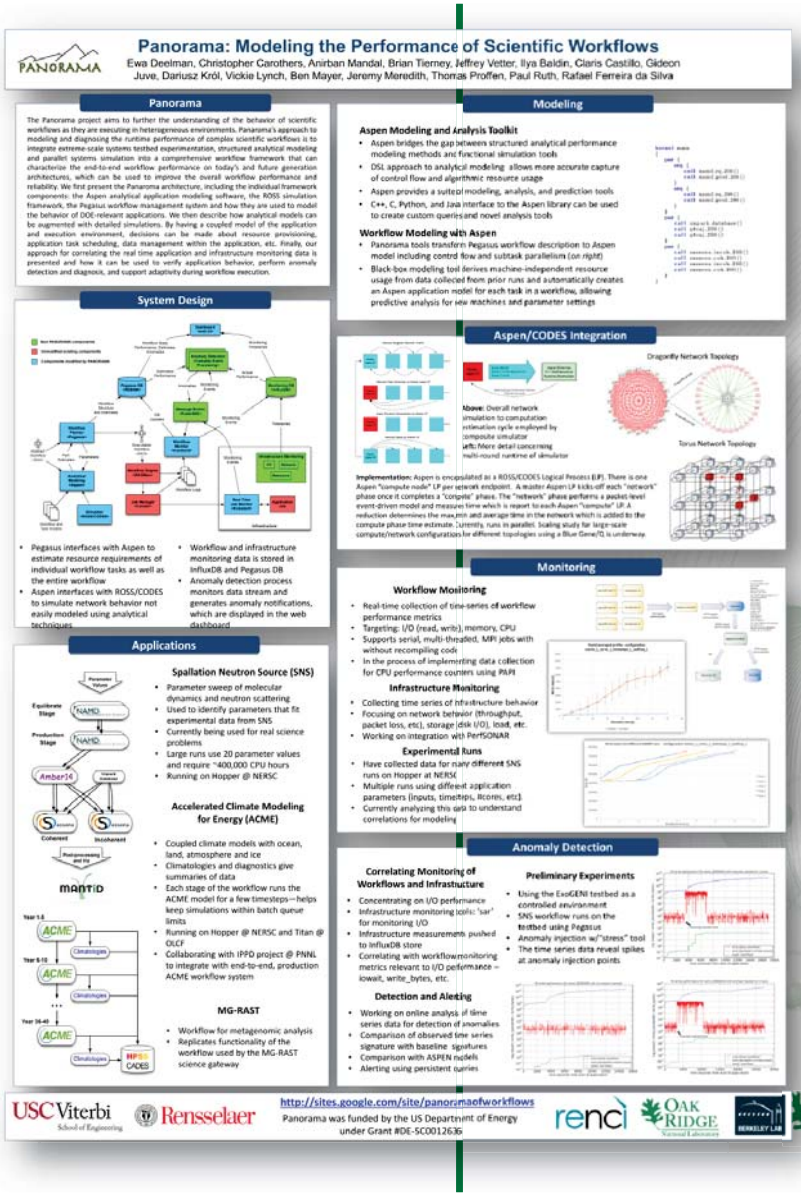
Model Scaling Validation (LULESH)



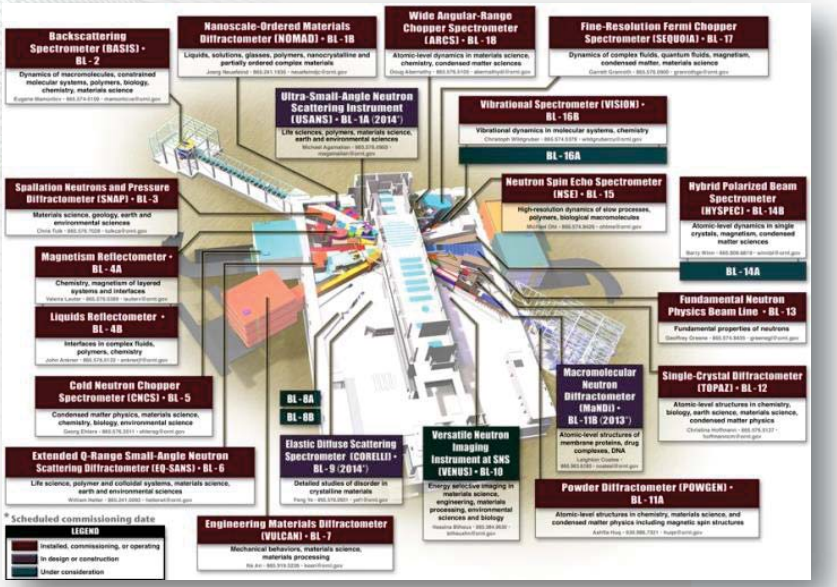
Performance Modeling for Distributed Scientific Workflows

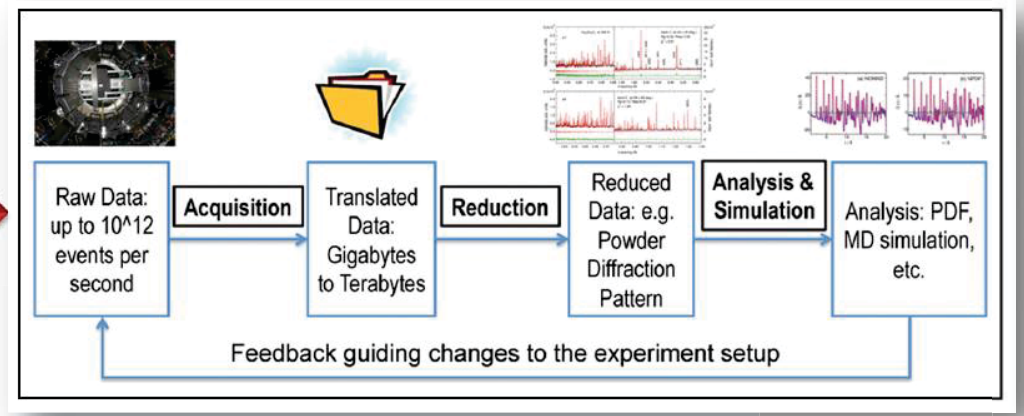
(see our Panorama poster)





Workflow: SNS





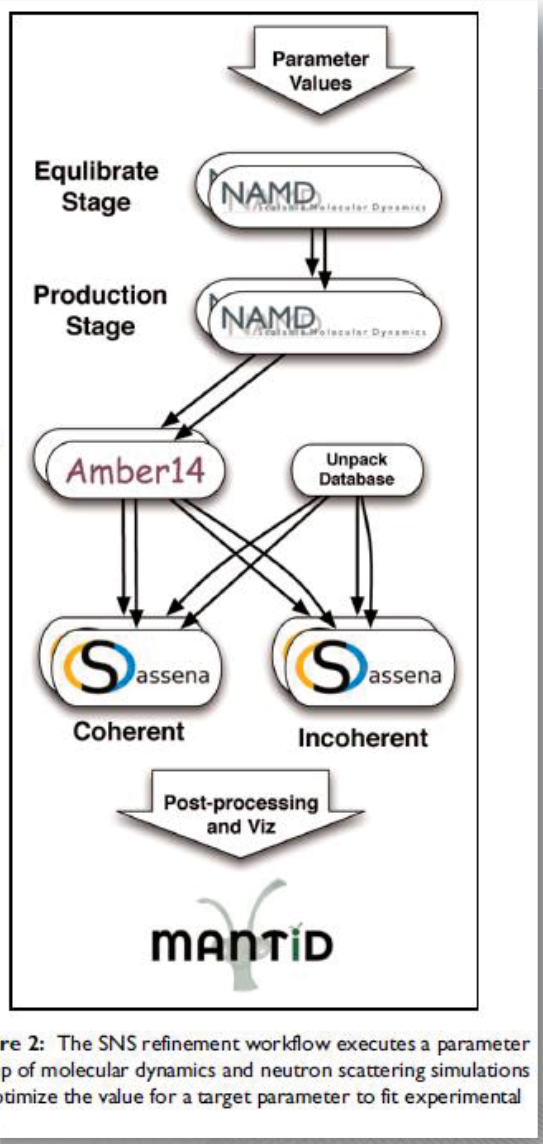


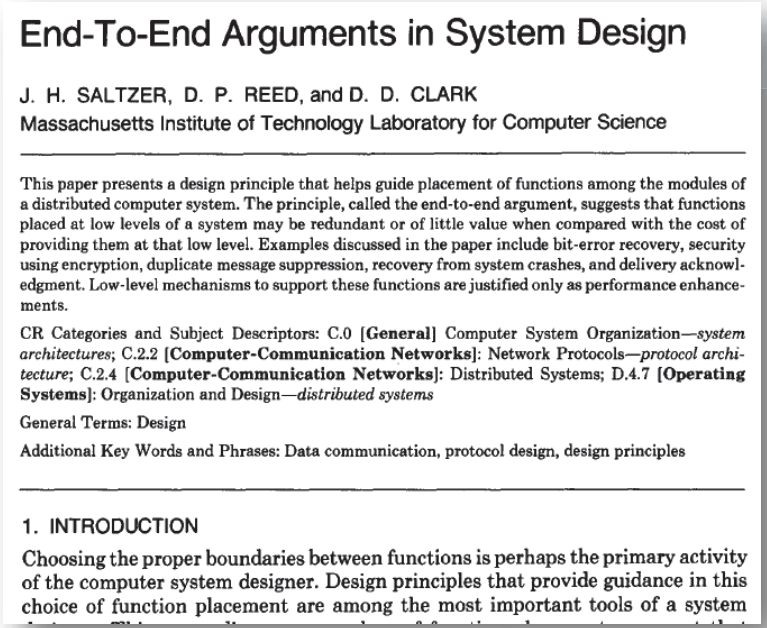
Figure 2: The SNS refinement workflow executes a parameter sweep of molecular dynamics and neutron scattering simulations to optimize the value for a target parameter to fit experimental data.

End-to-end Resiliency Design using Aspen

OAK RIDGE National Laboratory

Resiliency Modeling with Aspen

- End-to-End system design for Extreme-scale HPC
  - Why pay redundant costs for power, performance, etc?
- We introduce a new metric, the data vulnerability factor (DVF)
  - Quantifying vulnerability of data structures
  - Avoiding the isolation between application and hardware
- We measure DVF based on Aspen, a domain specific language for system modeling
- We categorize memory access patterns of scientific applications from a spectrum of computational domains
  - Dense linear algebra, Sparse linear algebra, N-body method, Structured grids, Spectral methods, and Monte Carlo
- We demonstrate the significance of DVF by two case studies
  - Algorithm optimization
  - Data protection quantification



End-To-End Arguments in System Design

J. H. SALTZER, D. P. REED, and D. D. CLARK
Massachusetts Institute of Technology Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. Examples discussed in the paper include bit-error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgment. Low-level mechanisms to support these functions are justified only as performance enhancements.

CR Categories and Subject Descriptors: C.0 [General] Computer System Organization—*system architectures*; C.2.2 [Computer-Communication Networks]: Network Protocols—*protocol architecture*; C.2.4 [Computer-Communication Networks]: Distributed Systems; D.4.7 [Operating Systems]: Organization and Design—*distributed systems*

General Terms: Design

Additional Key Words and Phrases: Data communication, protocol design, design principles

1. INTRODUCTION

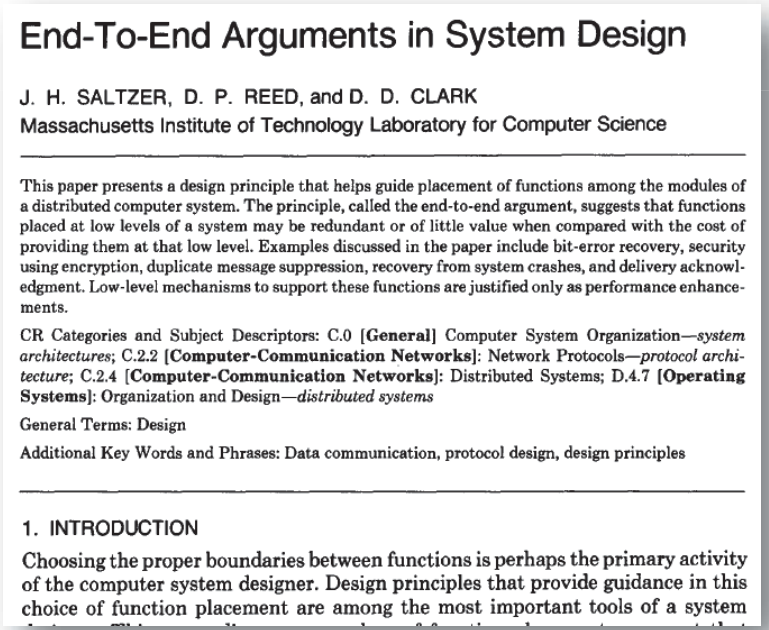
Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer. Design principles that provide guidance in this choice of function placement are among the most important tools of a system

DVF_d	DVF for a specific data structure
FIT	Failure rate (i.e., failures per billion hours per Mbit)
T	Application execution time
S_d	Size of data structure
N_{error}	Number of errors that could occur to a specific data structure during application execution
N_{ha}	Number of accesses to hardware (the main memory in this work)
n	Number of major data structures in an application
DVF_a	DVF for the application

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.*

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



End-To-End Arguments in System Design

J. H. SALTZER, D. P. REED, and D. D. CLARK
Massachusetts Institute of Technology Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. Examples discussed in the paper include bit-error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgment. Low-level mechanisms to support these functions are justified only as performance enhancements.

CR Categories and Subject Descriptors: C.0 [General] Computer System Organization—*system architectures*; C.2.2 [Computer-Communication Networks]: Network Protocols—*protocol architecture*; C.2.4 [Computer-Communication Networks]: Distributed Systems; D.4.7 [Operating Systems]: Organization and Design—*distributed systems*

General Terms: Design

Additional Key Words and Phrases: Data communication, protocol design, design principles

1. INTRODUCTION

Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer. Design principles that provide guidance in this choice of function placement are among the most important tools of a system

DVF_d	DVF for a specific data structure
FIT	Failure rate (i.e., failures per billion hours per Mbit)
T	Application execution time
S_d	Size of data structure
N_{error}	Number of errors that could occur to a specific data structure during application execution
N_{ha}	Number of accesses to hardware (the main memory in this work)
n	Number of major data structures in an application
DVF_a	DVF for the application

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.*

Workflow to calculate Data Vulnerability Factor

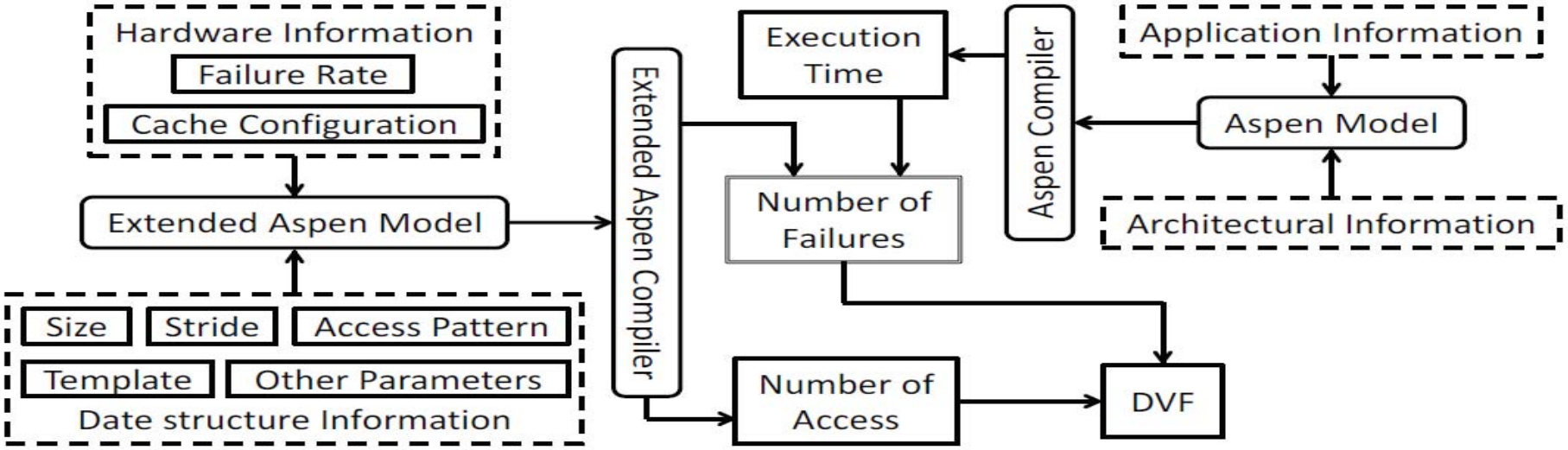


Fig. 3. The workflow to calculate DVF.

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

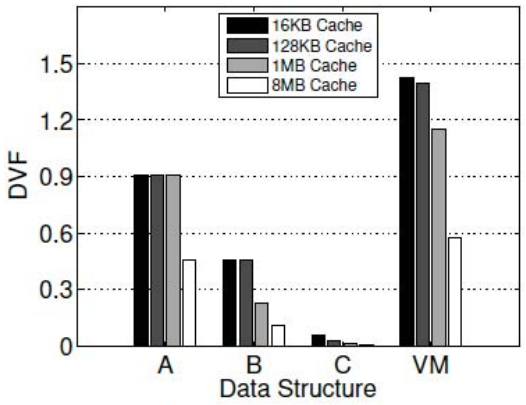
Extended Complier

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

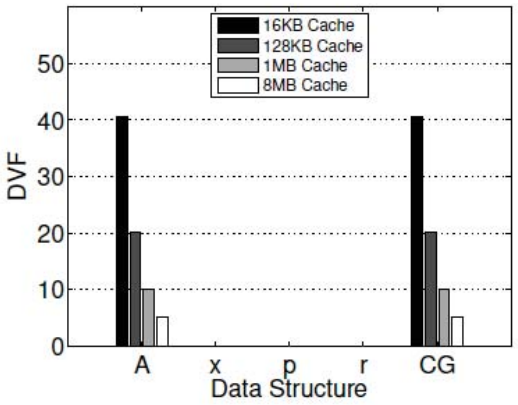
Resilience Modeling Results

DVF Results

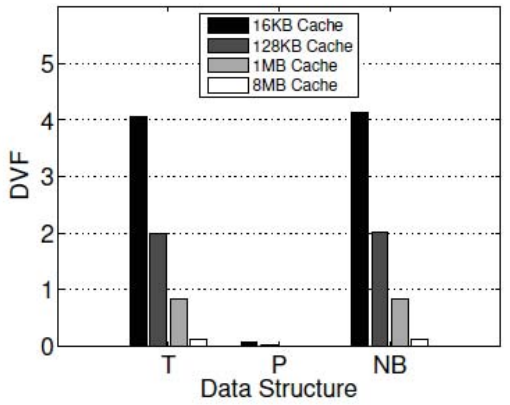
Provides insight for balancing interacting factors



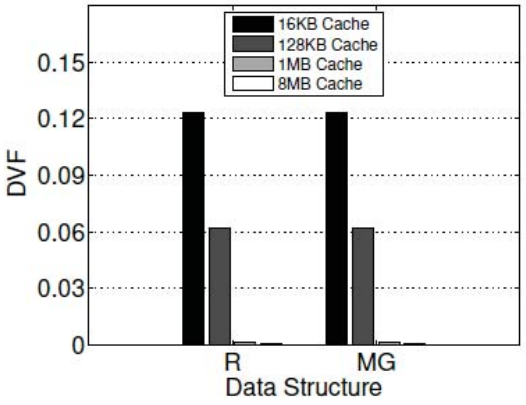
(a) Vector Multiplication



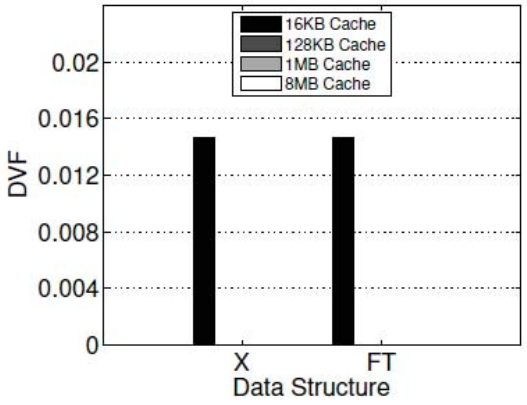
(b) Conjugate Gradient



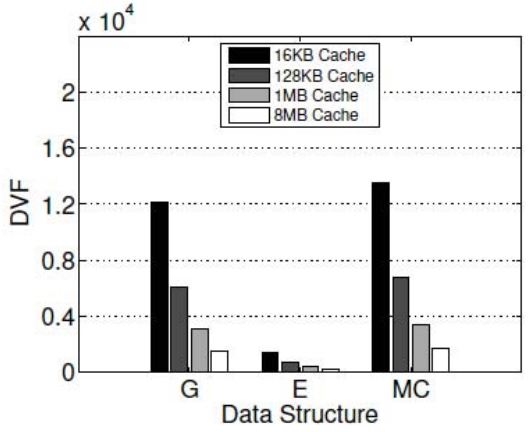
(c) Nbody (Barnes-hut)



(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!

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