

Using Application-Specific Performance Models to Inform Dynamic Scheduling

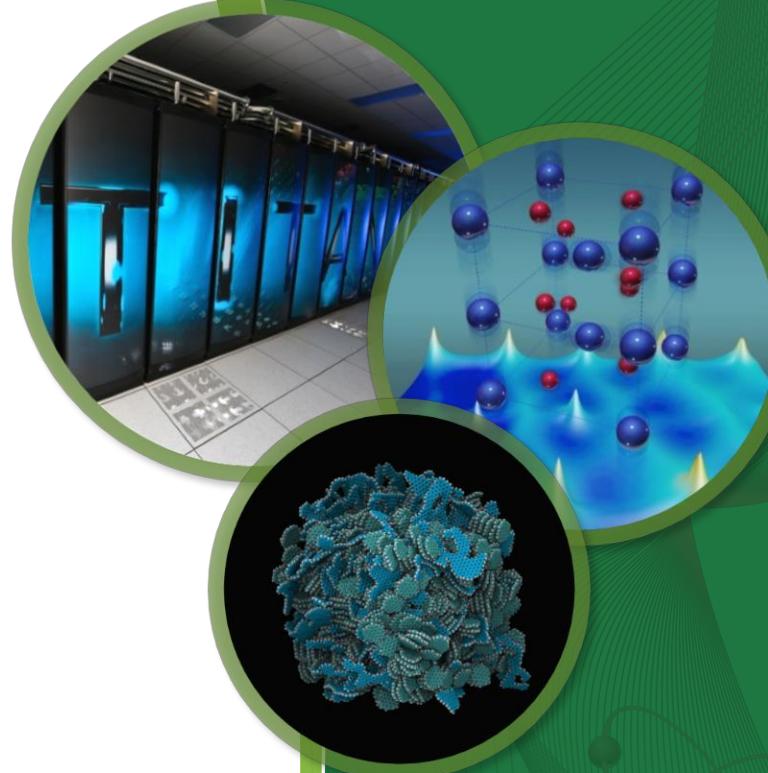
Jeffrey S. Vetter

Jeremy Meredith

and many collaborators

Presented to
11th Scheduling for Large Scale Systems Workshop

18-21 May 2016
Vanderbilt University
Nashville



Executive Summary

- Resource scheduling is a seminal problem for computing ... and it is becoming much more difficult
 - Scheduling has many potential solutions
 - Algorithmic, Historical, application specific, etc
- Both architectures and applications are growing more complex
 - Trends dictate that this will get worse; not better
 - This complexity creates irregularity in computation, communication, and data movement
- Posit that we can use application-specific performance models to inform scheduling decisions
 - Aspen performance modeling language helps create models
 - Two recent experiments
 - GPU offload
 - Distributed scientific workflows

Scheduling is critical to CS ...



11th Scheduling for Large Scale Systems Workshop

18-21 May 2016 Vanderbilt University, Nashville, TN (United States)

Login ▾

MAIN MENU

Home

Program

List of participants

Talks sorted by speakers

Talks sorted by themes

Venue

Hotels

Sponsors

HELP

@ Contact

THE 11TH SCHEDULING FOR LARGE SCALE SYSTEMS WORKSHOP

The 11th Scheduling for Large Scale Systems Workshop will be held at **Vanderbilt University in Nashville Tennessee**, May 18-May 20. This will be the eleventh edition of this workshop series after Aussois (2004), San Diego (2005), Aussois (2008), Knoxville (2009), Aussois (2011), Pittsburgh (2012), Dagstuhl (2013), Lyon (2014) and Dagstuhl (2015).

As in the past, the workshop will be structured as a set of **thematic** half-day sessions, mainly focused on scheduling and algorithms for large-scale systems. In addition to the talks (about 20 minutes each), plenty of time will be left for informal discussion and exchanges.

The workshop is by invitation only and there will be no registration fee.

NEWS:

Fill this poll to register for meals and the social event (that should take place in Mammoth Cave National Park):

<https://framadate.org/Us5qiTviro47HrlW>

ORGANIZING COMMITTEE

Guillaume Aupy, George Bosilca, Henri Casanova, Julien Langou, Padma Raghavan and Yves Robert

Trends toward Exascale



Exascale architecture targets circa 2009

2009 Exascale Challenges Workshop in San Diego

Attendees envisioned two possible architectural swim lanes:

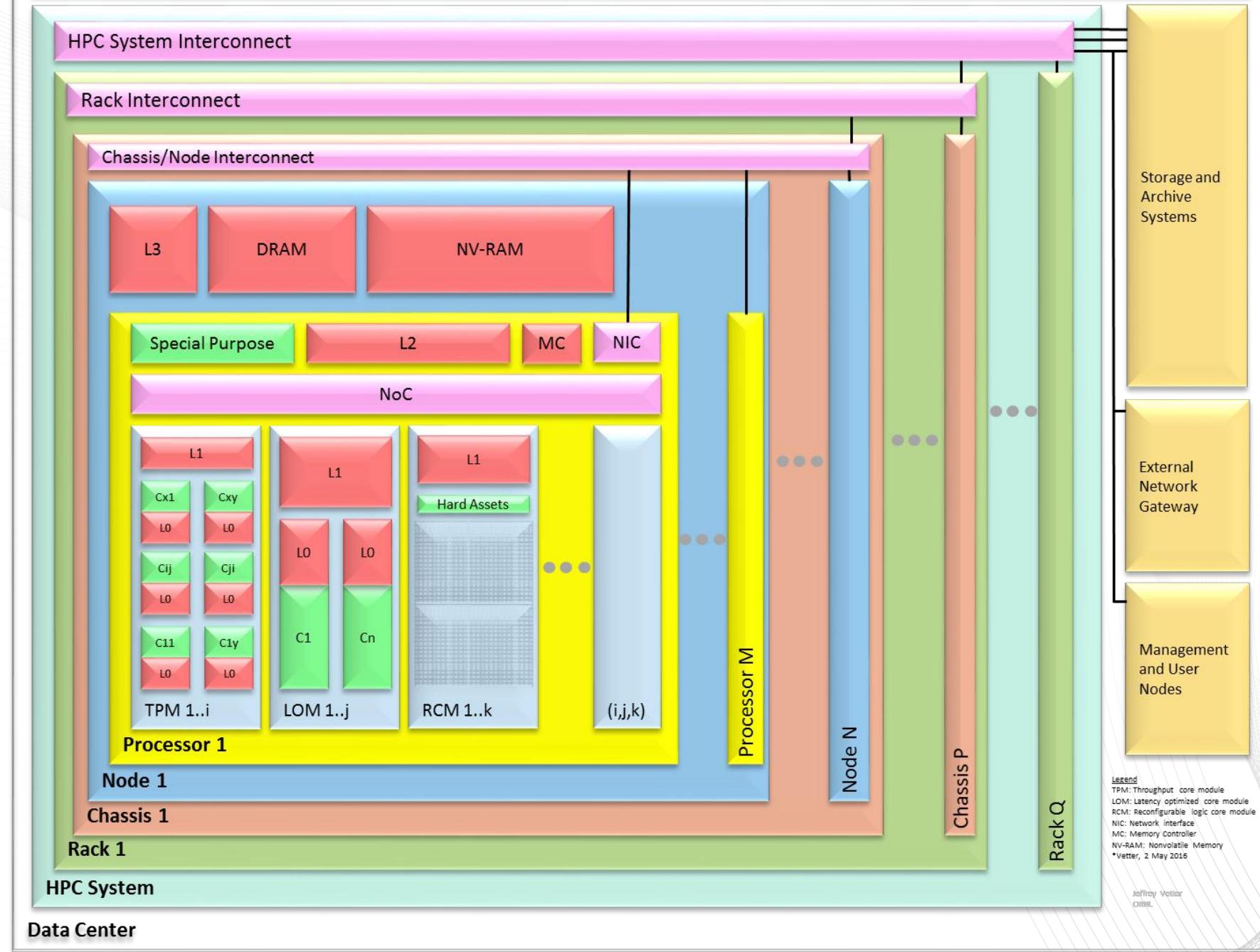
1. Homogeneous many-core thin-node system
2. Heterogeneous (accelerator + CPU) fat-node system

System attributes	2009	“Pre-Exascale”		“Exascale”	
System peak	2 PF	100-200 PF/s		1 Exaflop/s	
Power	6 MW	15 MW		20 MW	
System memory	0.3 PB	5 PB		32–64 PB	
Storage	15 PB	150 PB		500 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1 TB/s	1 TB/s	0.4 TB/s	4 TB/s
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	500,000	50,000	1,000,000	100,000
Node interconnect BW	1.5 GB/s	150 GB/s	1 TB/s	250 GB/s	2 TB/s
IO Bandwidth	0.2 TB/s	10 TB/s		30-60 TB/s	
MTTI	day	O(1 day)		O(0.1 day)	

Contemporary ASCR Computing At a Glance

System attributes	NERSC Now	OLCF Now	ALCF Now	NERSC Upgrade	OLCF Upgrade	ALCF Upgrades	
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 Volta 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	2 nd 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®

Complexity $\propto T$



Complexity is the next major challenge!

- “Exciting” times in computer architecture
 - Heterogeneous cores
 - Multimode memory systems
 - Fused memory systems
 - I/O architectures
 - Error correction
 - Changing system balance
- Uncertainty, Ambiguity
 - How do we design future systems so that they are faster than current systems on mission applications?
 - Entirely possible that the new system will be slower than the old system!
 - How do we provide some level of performance portability for applications teams?
 - How do we understand reliability and performance problems?
- Managing complexity is our main challenge!

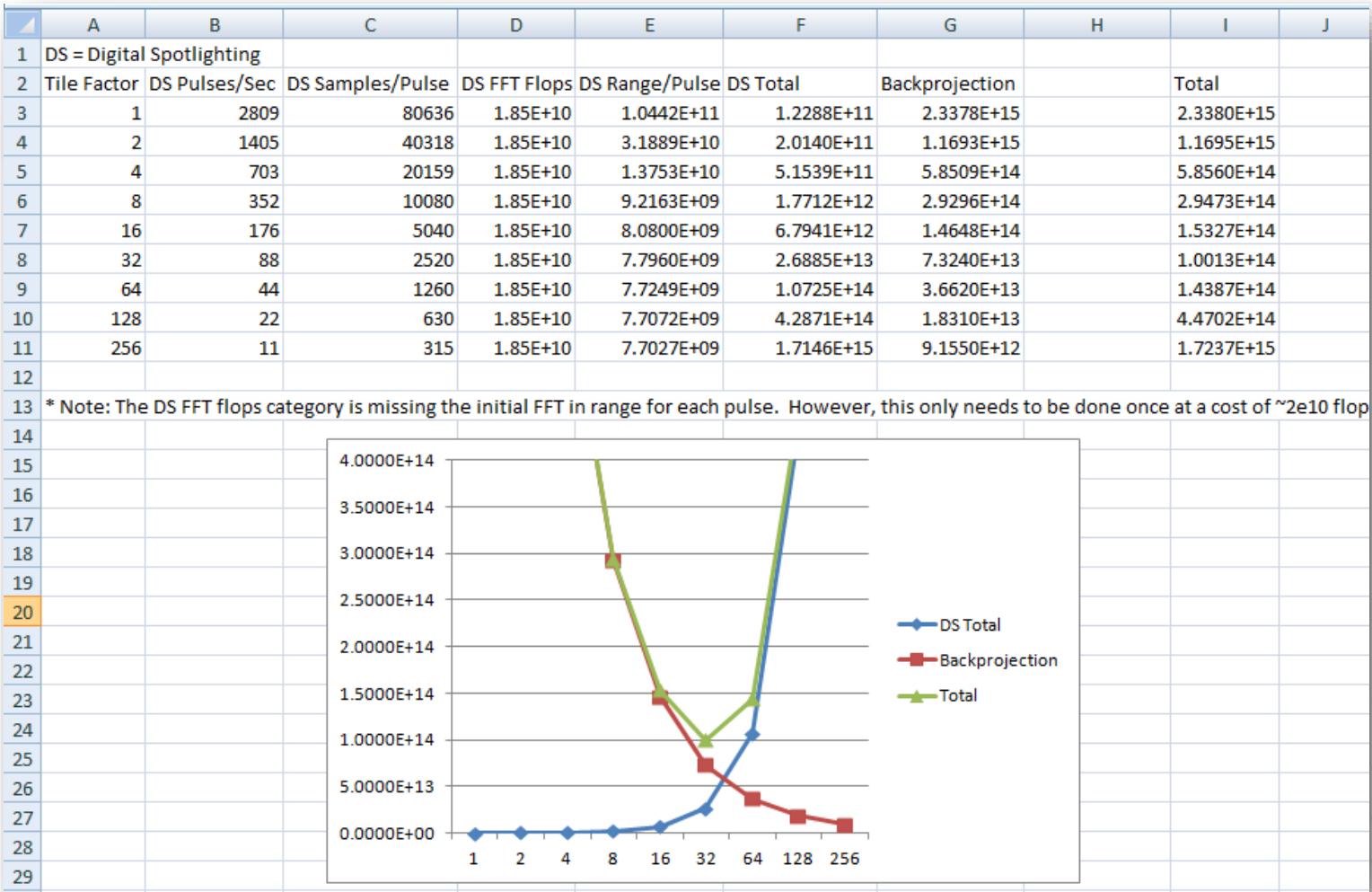
Performance Prediction with Aspen

Example Ad Hoc Model: Latex Equations

the communication and computation of two sheets. The expression we get for the runtime is

$$\begin{aligned} T = & 2 \left[t_c \frac{n}{p} n \log_2 n + (p-1)o + (p-2)g + \frac{n}{p} nG \right. \\ & + L + \left(\frac{n}{p} - 1 \right) \max \left\{ (p-1)o + t_c \frac{n}{p} n \log_2 n, \right. \\ & \left. \left. (p-1)g + \frac{n}{p} nG + L \right\} \right] + t_c \frac{n^2}{p^2} n \log_2 n \end{aligned}$$

Example: Ad-Hoc Excel Files



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
<i>Aspen</i>	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

Model Uses

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

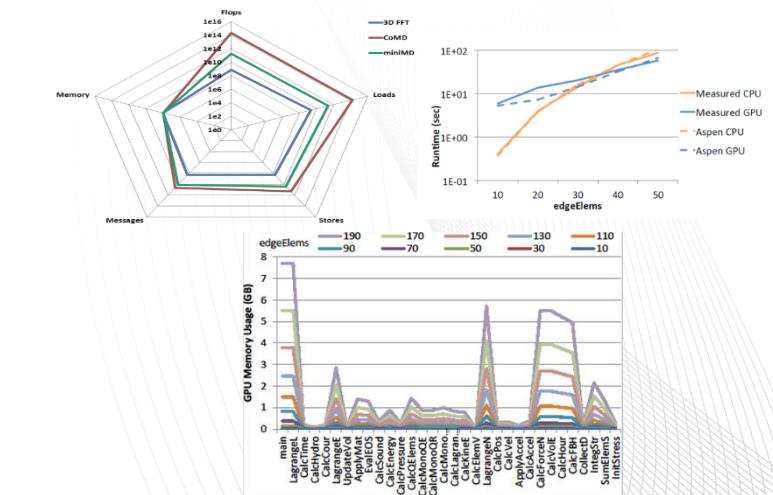
E.g., MD, UHPC CP 1, Lulesh,
3D FFT, CoMD, VPFFT, ...

Source code

```
2324 static inline
2325 void CalcMonotonicQGradientsForElems(Index_t p_nodelist[T_NUMELEM8],
2326     Real_t p_x[T_NUMNODE], Real_t p_y[T_NUMNODE], Real_t p_z[T_NUMNODE],
2327     Real_t p_xd[T_NUMNODE], Real_t p_yd[T_NUMNODE],Real_t p_zd[T_NUMNODE],
2328     Real_t p_volo[T_NUMELEM], Real_t p_vnew[T_NUMELEM],
2329     Real_t p_delix_zeta[T_NUMELEM], Real_t p_dely_zeta[T_NUMELEM],
2330     Real_t p_delix_xi[T_NUMELEM], Real_t p_dely_xi[T_NUMELEM],
2331     Real_t p_dely_eta[T_NUMELEM], Real_t p_dely_uta[T_NUMELEM])
2332 {
2333     Index_t i;
2334     Index_t numElem = m_numElem;
2335     #pragma acc parallel loop independent present(p_vnew, p_nodelist, p_x, p_y, p_z, p_xd,
2336     p_yd, p_zd, p_volo, p_delix_xi, p_dely_eta, p_delix_zeta, p_dely_xi, p_dely_eta,\n
2337     p_dely_zeta)
2338     for (i = 0 ; i < numElem ; ++i) {
2339         const Real_t ptiny = 1.e-36 ;
2340         Real_t ax,ay,az ;
2341         Real_t ddx,dyy,dzz ;
2342
2343         const Index_t *elemToNode = &p_nodelist[8*i];
2344         Index_t n0 = elemToNode[0] ;
2345         Index_t n1 = elemToNode[1] ;
2346         Index_t n2 = elemToNode[2] ;
2347         Index_t n3 = elemToNode[3] ;
2348         Index_t n4 = elemToNode[4] ;
2349         Index_t n5 = elemToNode[5] ;
2350         Index_t n6 = elemToNode[6] ;
2351         Index_t n7 = elemToNode[7] ;
2352
2353         Real_t x0 = p_x[n0] ;
```

Aspen code

```
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 xvvel
157         loads/caching [8 * wordSize] from yvvel
158         loads/caching [8 * wordSize] from zvvel
159
160         loads [wordSize] from volo
161         loads [wordSize] from vnew
162         // dx, dy, etc.
163         flops [90] as dp, simd
164         // delvx/delx
165         flops [9 * 8 + 3 + 30 + 5] as dp, simd
166         stores [wordSize] to delv_xeta
167         // delxi/deli
168         flops [9 * 8 + 3 + 30 + 5] as dp, simd
169         stores [wordSize] to delv_xi
170         // delxj/delvj
171         flops [9 * 8 + 3 + 30 + 5] as dp, simd
172         stores [wordSize] to delv_eta
173     }
174 }
```



Creating an Aspen Model

Manual Example of LULESH

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

jsmeredith on Sep 20, 2013 adding models

1 contributor

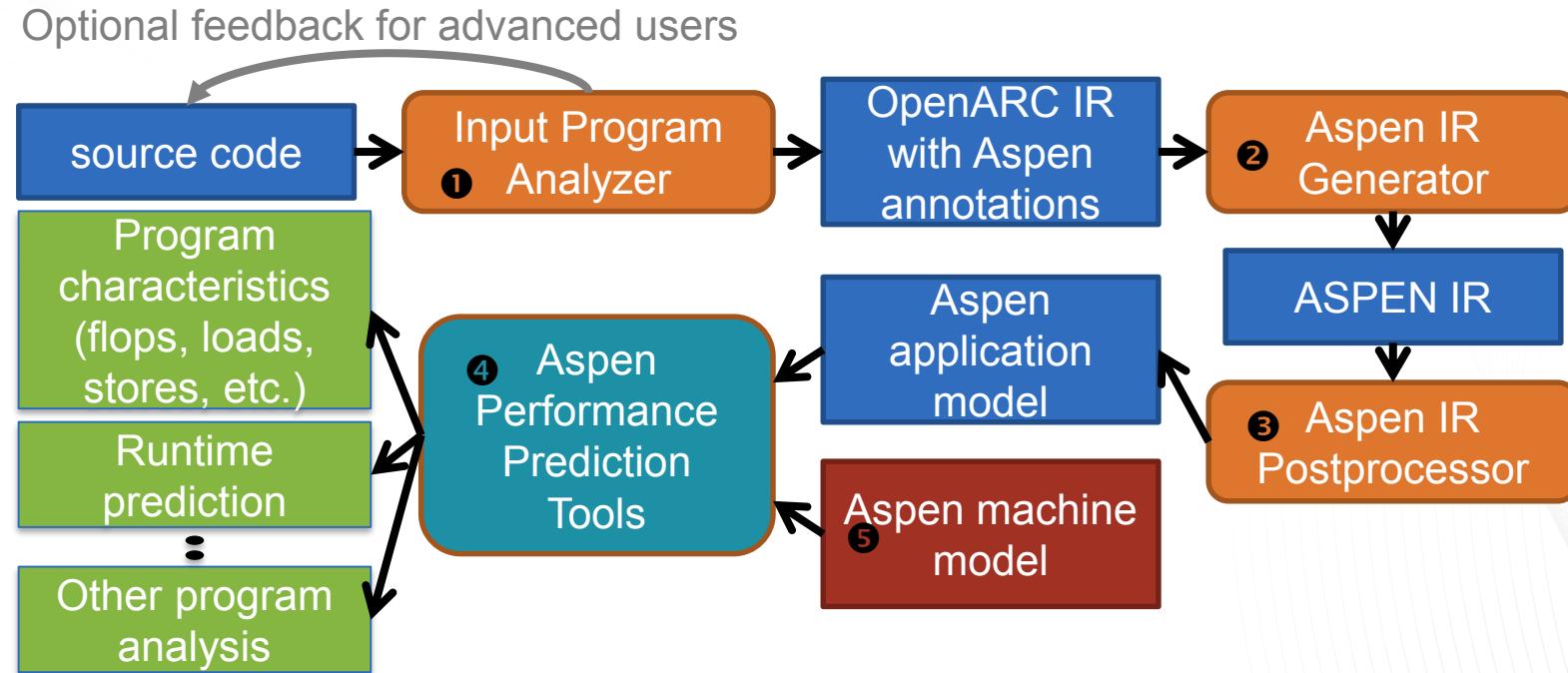
336 lines (288 sloc) | 9.213 kb

Raw Blame History

```
1 //  
2 // lulesh.aspen  
3 //  
4 // An ASPECT application model for the LULESH 1.01 challenge problem. Based  
5 // on the CUDA version of the source code found at:  
6 // https://computation.llnl.gov/casc/ShockHydro/  
7 //  
8 param nTimeSteps = 1495  
9  
10 // Information about domain  
11 param edgeElems = 45  
12 param edgeNodes = edgeElems + 1  
13  
14 param numElems = edgeElems^3  
15 param numNodes = edgeNodes^3  
16  
17 // Double precision  
18 param wordSize = 8  
19  
20 // Element data  
21 data mNodeList as Array(numElems, wordSize)  
22 data mAtEleList as Array(numElems, wordSize)  
23 data mNodeList as Array(8 * numElems, wordSize) // 8 nodes per element  
24 data mixim as Array(numElems, wordSize)  
25 data mixip as Array(numElems, wordSize)  
26 data mletam as Array(numElems, wordSize)  
27 data mletap as Array(numElems, wordSize)  
28 data mzeta as Array(numElems, wordSize)  
29 data mzetap as Array(numElems, wordSize)  
30 data melemBC as Array(numElems, wordSize)  
31 data mE as Array(numElems, wordSize)  
32 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 }

COMPASS System Overview



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
```

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 [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.

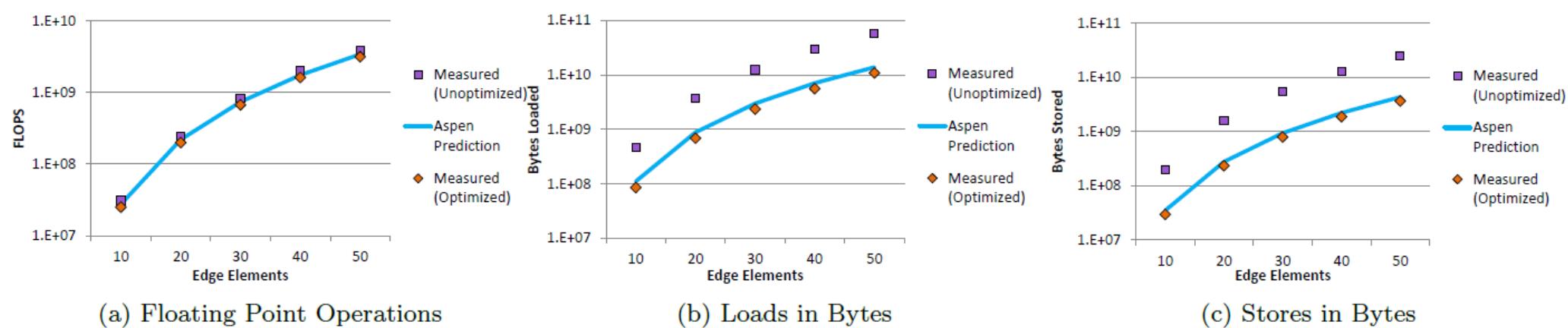
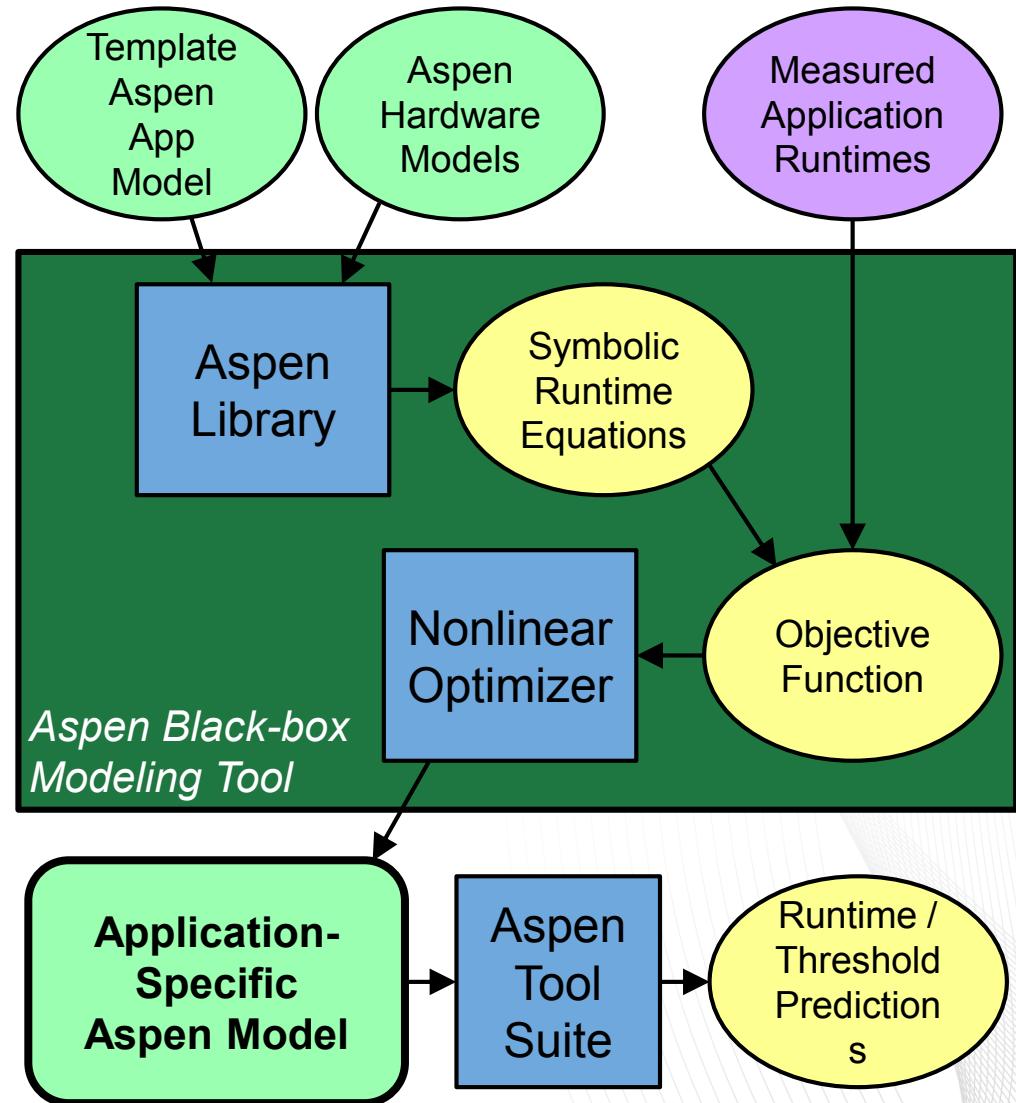


Figure 2: Predicted Resource Usage of LULESH versus Measured (with and without compiler optimization)

Black Box Analytical Modeling

- In some cases, we do not have access to a white box Aspen performance model
- Using input vector and empirical results, we can develop Aspen Black Box model
- User provides
 - measured runtimes with app/machine parameters
 - e.g. nAtoms, nCores
 - template Aspen model with
 - application parameters
 - unknowns to solve for
 - new machine models (if necessary)
- Modeling tool
 - Generates symbolic predictions
 - Combines with measurements to generate objective function
 - Solves for unknowns in template
 - Output: completed app model usable for predictive behavior



Black Box Modeling Example

MD template model

```
model NAMD_Template {
    // application parameters
    // (defined in the input file)
    param nAtoms      = 1e6
    param nTimeSteps = 100
    // solve for these parameters
    // (within the given ranges)
    param c = 1 in 1 .. 1e18
    param d = 1 in 1 .. 1e18
    // application behavior:
    // execution and control flow
    kernel main
    {
        iterate [nTimeSteps] {
            execute {
                loads [c * nAtoms^2]
                flops [d * nAtoms]
            }
        }
    }
}
```

CSV data file with parameters
and runtimes

nAtoms	nTimeStep s	nCores	machine	runtime
1e6	100	144	exogeni	384.2
1e6	100	144	hopper	340.1
1e6	150	144	hopper	482.9

Concrete NAMD model

```
model NAMD_Equilibrate {

    // NAMD input parameters
    param nAtoms      = 1e6
    param nTimeSteps = 100

    // calculation-specific constants
    param c = 402.1
    param d = 10.95

    // NAMD application behavior
    kernel main
    {
        iterate [nTimeSteps] {
            execute {
                loads [c * nAtoms^2]
                flops [d * nAtoms]
            }
        }
    }
}
```

- nAtoms and nTimeSteps defined in template application model and CSV input data
- nCores defined in machine models and CSV input data
- solves for c and d, filling out a concrete application model for that problem
- new predictions can still vary nAtoms, nTimeSteps, and nCores

Using an Aspen Model

Aspen: Abstract Scalable Performance Engineering Notation

Model Creation

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

- Modular
- Sharable
- Composable
- Reflects prog structure

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

Source code

```

3234 static inline
3235 void CalcMonotonicQGGradientsForElements(Index_t p_nodelist[T_NUMELEM],
3236     Real_t p_x[T_NUMNODE], Real_t p_y[T_NUMNODE], Real_t p_z[T_NUMNODE],
3237     Real_t p_xd[T_NUMNODE], Real_t p_yd[T_NUMNODE], Real_t p_zd[T_NUMNODE],
3238     Real_t p_volo[T_NUMELEM], Real_t p_vnew[T_NUMELEM],
3239     Real_t p_delx_zeta[T_NUMELEM], Real_t p_delv_zeta[T_NUMELEM],
3240     Real_t p_delx_eta[T_NUMELEM], Real_t p_delv_eta[T_NUMELEM],
3241     Real_t p_delx_xi[T_NUMELEM], Real_t p_delv_xi[T_NUMELEM],
3242     Real_t p_delx_eta[T_NUMELEM], Real_t p_delv_eta[T_NUMELEM])
3243 {
3244     Index_t i;
3245     Index_t numElem = m_numElem;
3246 #pragma acc parallel loop independent present(p_vnew, p_nodelist, p_x, p_y, p_z, p_xd,
3247     p_yd, p_zd, p_volo, p_vnew, p_delx_xi, p_delv_eta, p_delx_eta, p_delv_xi, p_delv_eta,\n
3248     p_delt_zeta)
3249     for (i = 0 ; i < numElem ; ++i) {
3250         const Real_t ptiny = 1.e-36 ;
3251         Real_t ax,ay,az ;
3252         Real_t dxv,dyv,dzv ;
3253
3254         const Index_t *elemToNode = 6p_nodelist[8*i];
3255         Index_t n0 = elemToNode[0] ;
3256         Index_t n1 = elemToNode[1] ;
3257         Index_t n2 = elemToNode[2] ;
3258         Index_t n3 = elemToNode[3] ;
3259         Index_t n4 = elemToNode[4] ;
3260         Index_t n5 = elemToNode[5] ;
3261         Index_t n6 = elemToNode[6] ;
3262         Index_t n7 = elemToNode[7] ;
3263
3264         Real_t x0 = p_x[n0] ;
3265         Real_t y0 = p_y[n0] ;
3266         Real_t z0 = p_z[n0] ;
3267
3268         Real_t x1 = p_x[n1] ;
3269         Real_t y1 = p_y[n1] ;
3270         Real_t z1 = p_z[n1] ;
3271
3272         Real_t x2 = p_x[n2] ;
3273         Real_t y2 = p_y[n2] ;
3274         Real_t z2 = p_z[n2] ;
3275
3276         Real_t x3 = p_x[n3] ;
3277         Real_t y3 = p_y[n3] ;
3278         Real_t z3 = p_z[n3] ;
3279
3280         Real_t x4 = p_x[n4] ;
3281         Real_t y4 = p_y[n4] ;
3282         Real_t z4 = p_z[n4] ;
3283
3284         Real_t x5 = p_x[n5] ;
3285         Real_t y5 = p_y[n5] ;
3286         Real_t z5 = p_z[n5] ;
3287
3288         Real_t x6 = p_x[n6] ;
3289         Real_t y6 = p_y[n6] ;
3290         Real_t z6 = p_z[n6] ;
3291
3292         Real_t x7 = p_x[n7] ;
3293         Real_t y7 = p_y[n7] ;
3294         Real_t z7 = p_z[n7] ;
3295
3296         Real_t dx = x1 - x0 ;
3297         Real_t dy = y1 - y0 ;
3298         Real_t dz = z1 - z0 ;
3299
3300         Real_t dx2 = x2 - x0 ;
3301         Real_t dy2 = y2 - y0 ;
3302         Real_t dz2 = z2 - z0 ;
3303
3304         Real_t dx3 = x3 - x0 ;
3305         Real_t dy3 = y3 - y0 ;
3306         Real_t dz3 = z3 - z0 ;
3307
3308         Real_t dx4 = x4 - x0 ;
3309         Real_t dy4 = y4 - y0 ;
3310         Real_t dz4 = z4 - z0 ;
3311
3312         Real_t dx5 = x5 - x0 ;
3313         Real_t dy5 = y5 - y0 ;
3314         Real_t dz5 = z5 - z0 ;
3315
3316         Real_t dx6 = x6 - x0 ;
3317         Real_t dy6 = y6 - y0 ;
3318         Real_t dz6 = z6 - z0 ;
3319
3320         Real_t dx7 = x7 - x0 ;
3321         Real_t dy7 = y7 - y0 ;
3322         Real_t dz7 = z7 - z0 ;
3323
3324         Real_t dxv1 = dx + dx2 ;
3325         Real_t dyv1 = dy + dy2 ;
3326         Real_t dzv1 = dz + dz2 ;
3327
3328         Real_t dxv2 = dx3 + dx4 ;
3329         Real_t dyv2 = dy3 + dy4 ;
3330         Real_t dzv2 = dz3 + dz4 ;
3331
3332         Real_t dxv3 = dx5 + dx6 ;
3333         Real_t dyv3 = dy5 + dy6 ;
3334         Real_t dzv3 = dz5 + dz6 ;
3335
3336         Real_t dxv4 = dx7 ;
3337         Real_t dyv4 = dy7 ;
3338         Real_t dzv4 = dz7 ;
3339
3340         Real_t dxv = dxv1 + dxv2 + dxv3 + dxv4 ;
3341         Real_t dyv = dyv1 + dyv2 + dyv3 + dyv4 ;
3342         Real_t dzv = dzv1 + dzv2 + dzv3 + dzv4 ;
3343
3344         Real_t dxv5 = dxv / 4.0 ;
3345         Real_t dyv5 = dyv / 4.0 ;
3346         Real_t dzv5 = dzv / 4.0 ;
3347
3348         Real_t dxv6 = dxv5 ;
3349         Real_t dyv6 = dyv5 ;
3350         Real_t dzv6 = dzv5 ;
3351
3352         Real_t dxv7 = dxv6 ;
3353         Real_t dyv7 = dyv6 ;
3354         Real_t dzv7 = dzv6 ;
3355
3356         Real_t dxv8 = dxv7 ;
3357         Real_t dyv8 = dyv7 ;
3358         Real_t dzv8 = dzv7 ;
3359
3360         Real_t dxv9 = dxv8 ;
3361         Real_t dyv9 = dyv8 ;
3362         Real_t dzv9 = dzv8 ;
3363
3364         Real_t dxv10 = dxv9 ;
3365         Real_t dyv10 = dyv9 ;
3366         Real_t dzv10 = dzv9 ;
3367
3368         Real_t dxv11 = dxv10 ;
3369         Real_t dyv11 = dyv10 ;
3370         Real_t dzv11 = dzv10 ;
3371
3372         Real_t dxv12 = dxv11 ;
3373         Real_t dyv12 = dyv11 ;
3374         Real_t dzv12 = dzv11 ;
3375
3376         Real_t dxv13 = dxv12 ;
3377         Real_t dyv13 = dyv12 ;
3378         Real_t dzv13 = dzv12 ;
3379
3380         Real_t dxv14 = dxv13 ;
3381         Real_t dyv14 = dyv13 ;
3382         Real_t dzv14 = dzv13 ;
3383
3384         Real_t dxv15 = dxv14 ;
3385         Real_t dyv15 = dyv14 ;
3386         Real_t dzv15 = dzv14 ;
3387
3388         Real_t dxv16 = dxv15 ;
3389         Real_t dyv16 = dyv15 ;
3390         Real_t dzv16 = dzv15 ;
3391
3392         Real_t dxv17 = dxv16 ;
3393         Real_t dyv17 = dyv16 ;
3394         Real_t dzv17 = dzv16 ;
3395
3396         Real_t dxv18 = dxv17 ;
3397         Real_t dyv18 = dyv17 ;
3398         Real_t dzv18 = dzv17 ;
3399
3400         Real_t dxv19 = dxv18 ;
3401         Real_t dyv19 = dyv18 ;
3402         Real_t dzv19 = dzv18 ;
3403
3404         Real_t dxv20 = dxv19 ;
3405         Real_t dyv20 = dyv19 ;
3406         Real_t dzv20 = dzv19 ;
3407
3408         Real_t dxv21 = dxv20 ;
3409         Real_t dyv21 = dyv20 ;
3410         Real_t dzv21 = dzv19 ;
3411
3412         Real_t dxv22 = dxv21 ;
3413         Real_t dyv22 = dyv21 ;
3414         Real_t dzv22 = dzv21 ;
3415
3416         Real_t dxv23 = dxv22 ;
3417         Real_t dyv23 = dyv22 ;
3418         Real_t dzv23 = dzv22 ;
3419
3420         Real_t dxv24 = dxv23 ;
3421         Real_t dyv24 = dyv23 ;
3422         Real_t dzv24 = dzv23 ;
3423
3424         Real_t dxv25 = dxv24 ;
3425         Real_t dyv25 = dyv24 ;
3426         Real_t dzv25 = dzv24 ;
3427
3428         Real_t dxv26 = dxv25 ;
3429         Real_t dyv26 = dyv25 ;
3430         Real_t dzv26 = dzv25 ;
3431
3432         Real_t dxv27 = dxv26 ;
3433         Real_t dyv27 = dyv26 ;
3434         Real_t dzv27 = dzv26 ;
3435
3436         Real_t dxv28 = dxv27 ;
3437         Real_t dyv28 = dyv27 ;
3438         Real_t dzv28 = dzv27 ;
3439
3440         Real_t dxv29 = dxv28 ;
3441         Real_t dyv29 = dyv28 ;
3442         Real_t dzv29 = dzv28 ;
3443
3444         Real_t dxv30 = dxv29 ;
3445         Real_t dyv30 = dyv29 ;
3446         Real_t dzv30 = dzv28 ;
3447
3448         Real_t dxv31 = dxv30 ;
3449         Real_t dyv31 = dyv29 ;
3450         Real_t dzv31 = dzv28 ;
3451
3452         Real_t dxv32 = dxv31 ;
3453         Real_t dyv32 = dyv29 ;
3454         Real_t dzv32 = dzv28 ;
3455
3456         Real_t dxv33 = dxv32 ;
3457         Real_t dyv33 = dyv29 ;
3458         Real_t dzv33 = dzv28 ;
3459
3460         Real_t dxv34 = dxv33 ;
3461         Real_t dyv34 = dyv29 ;
3462         Real_t dzv34 = dzv28 ;
3463
3464         Real_t dxv35 = dxv34 ;
3465         Real_t dyv35 = dyv29 ;
3466         Real_t dzv35 = dzv28 ;
3467
3468         Real_t dxv36 = dxv35 ;
3469         Real_t dyv36 = dyv29 ;
3470         Real_t dzv36 = dzv28 ;
3471
3472         Real_t dxv37 = dxv36 ;
3473         Real_t dyv37 = dyv29 ;
3474         Real_t dzv37 = dzv28 ;
3475
3476         Real_t dxv38 = dxv37 ;
3477         Real_t dyv38 = dyv29 ;
3478         Real_t dzv38 = dzv28 ;
3479
3480         Real_t dxv39 = dxv38 ;
3481         Real_t dyv39 = dyv29 ;
3482         Real_t dzv39 = dzv28 ;
3483
3484         Real_t dxv40 = dxv39 ;
3485         Real_t dyv40 = dyv29 ;
3486         Real_t dzv40 = dzv28 ;
3487
3488         Real_t dxv41 = dxv40 ;
3489         Real_t dyv41 = dyv29 ;
3490         Real_t dzv41 = dzv28 ;
3491
3492         Real_t dxv42 = dxv41 ;
3493         Real_t dyv42 = dyv29 ;
3494         Real_t dzv42 = dzv28 ;
3495
3496         Real_t dxv43 = dxv42 ;
3497         Real_t dyv43 = dyv29 ;
3498         Real_t dzv43 = dzv28 ;
3499
3500         Real_t dxv44 = dxv43 ;
3501         Real_t dyv44 = dyv29 ;
3502         Real_t dzv44 = dzv28 ;
3503
3504         Real_t dxv45 = dxv44 ;
3505         Real_t dyv45 = dyv29 ;
3506         Real_t dzv45 = dzv28 ;
3507
3508         Real_t dxv46 = dxv45 ;
3509         Real_t dyv46 = dyv29 ;
3510         Real_t dzv46 = dzv28 ;
3511
3512         Real_t dxv47 = dxv46 ;
3513         Real_t dyv47 = dyv29 ;
3514         Real_t dzv47 = dzv28 ;
3515
3516         Real_t dxv48 = dxv47 ;
3517         Real_t dyv48 = dyv29 ;
3518         Real_t dzv48 = dzv28 ;
3519
3520         Real_t dxv49 = dxv48 ;
3521         Real_t dyv49 = dyv29 ;
3522         Real_t dzv49 = dzv28 ;
3523
3524         Real_t dxv50 = dxv49 ;
3525         Real_t dyv50 = dyv29 ;
3526         Real_t dzv50 = dzv28 ;
3527
3528         Real_t dxv51 = dxv50 ;
3529         Real_t dyv51 = dyv29 ;
3530         Real_t dzv51 = dzv28 ;
3531
3532         Real_t dxv52 = dxv51 ;
3533         Real_t dyv52 = dyv29 ;
3534         Real_t dzv52 = dzv28 ;
3535
3536         Real_t dxv53 = dxv52 ;
3537         Real_t dyv53 = dyv29 ;
3538         Real_t dzv53 = dzv28 ;
3539
3540         Real_t dxv54 = dxv53 ;
3541         Real_t dyv54 = dyv29 ;
3542         Real_t dzv54 = dzv28 ;
3543
3544         Real_t dxv55 = dxv54 ;
3545         Real_t dyv55 = dyv29 ;
3546         Real_t dzv55 = dzv28 ;
3547
3548         Real_t dxv56 = dxv55 ;
3549         Real_t dyv56 = dyv29 ;
3550         Real_t dzv56 = dzv28 ;
3551
3552         Real_t dxv57 = dxv56 ;
3553         Real_t dyv57 = dyv29 ;
3554         Real_t dzv57 = dzv28 ;
3555
3556         Real_t dxv58 = dxv57 ;
3557         Real_t dyv58 = dyv29 ;
3558         Real_t dzv58 = dzv28 ;
3559
3560         Real_t dxv59 = dxv58 ;
3561         Real_t dyv59 = dyv29 ;
3562         Real_t dzv59 = dzv28 ;
3563
3564         Real_t dxv60 = dxv59 ;
3565         Real_t dyv60 = dyv29 ;
3566         Real_t dzv60 = dzv28 ;
3567
3568         Real_t dxv61 = dxv60 ;
3569         Real_t dyv61 = dyv29 ;
3570         Real_t dzv61 = dzv28 ;
3571
3572         Real_t dxv62 = dxv61 ;
3573         Real_t dyv62 = dyv29 ;
3574         Real_t dzv62 = dzv28 ;
3575
3576         Real_t dxv63 = dxv62 ;
3577         Real_t dyv63 = dyv29 ;
3578         Real_t dzv63 = dzv28 ;
3579
3580         Real_t dxv64 = dxv63 ;
3581         Real_t dyv64 = dyv29 ;
3582         Real_t dzv64 = dzv28 ;
3583
3584         Real_t dxv65 = dxv64 ;
3585         Real_t dyv65 = dyv29 ;
3586         Real_t dzv65 = dzv28 ;
3587
3588         Real_t dxv66 = dxv65 ;
3589         Real_t dyv66 = dyv29 ;
3590         Real_t dzv66 = dzv28 ;
3591
3592         Real_t dxv67 = dxv66 ;
3593         Real_t dyv67 = dyv29 ;
3594         Real_t dzv67 = dzv28 ;
3595
3596         Real_t dxv68 = dxv67 ;
3597         Real_t dyv68 = dyv29 ;
3598         Real_t dzv68 = dzv28 ;
3599
3600         Real_t dxv69 = dxv68 ;
3601         Real_t dyv69 = dyv29 ;
3602         Real_t dzv69 = dzv28 ;
3603
3604         Real_t dxv70 = dxv69 ;
3605         Real_t dyv70 = dyv29 ;
3606         Real_t dzv70 = dzv28 ;
3607
3608         Real_t dxv71 = dxv70 ;
3609         Real_t dyv71 = dyv29 ;
3610         Real_t dzv71 = dzv28 ;
3611
3612         Real_t dxv72 = dxv71 ;
3613         Real_t dyv72 = dyv29 ;
3614         Real_t dzv72 = dzv28 ;
3615
3616         Real_t dxv73 = dxv72 ;
3617         Real_t dyv73 = dyv29 ;
3618         Real_t dzv73 = dzv28 ;
3619
3620         Real_t dxv74 = dxv73 ;
3621         Real_t dyv74 = dyv29 ;
3622         Real_t dzv74 = dzv28 ;
3623
3624         Real_t dxv75 = dxv74 ;
3625         Real_t dyv75 = dyv29 ;
3626         Real_t dzv75 = dzv28 ;
3627
3628         Real_t dxv76 = dxv75 ;
3629         Real_t dyv76 = dyv29 ;
3630         Real_t dzv76 = dzv28 ;
3631
3632         Real_t dxv77 = dxv76 ;
3633         Real_t dyv77 = dyv29 ;
3634         Real_t dzv77 = dzv28 ;
3635
3636         Real_t dxv78 = dxv77 ;
3637         Real_t dyv78 = dyv29 ;
3638         Real_t dzv78 = dzv28 ;
3639
3640         Real_t dxv79 = dxv78 ;
3641         Real_t dyv79 = dyv29 ;
3642         Real_t dzv79 = dzv28 ;
3643
3644         Real_t dxv80 = dxv79 ;
3645         Real_t dyv80 = dyv29 ;
3646         Real_t dzv80 = dzv28 ;
3647
3648         Real_t dxv81 = dxv80 ;
3649         Real_t dyv81 = dyv29 ;
3650         Real_t dzv81 = dzv28 ;
3651
3652         Real_t dxv82 = dxv81 ;
3653         Real_t dyv82 = dyv29 ;
3654         Real_t dzv82 = dzv28 ;
3655
3656         Real_t dxv83 = dxv82 ;
3657         Real_t dyv83 = dyv29 ;
3658         Real_t dzv83 = dzv28 ;
3659
3660         Real_t dxv84 = dxv83 ;
3661         Real_t dyv84 = dyv29 ;
3662         Real_t dzv84 = dzv28 ;
3663
3664         Real_t dxv85 = dxv84 ;
3665         Real_t dyv85 = dyv29 ;
3666         Real_t dzv85 = dzv28 ;
3667
3668         Real_t dxv86 = dxv85 ;
3669         Real_t dyv86 = dyv29 ;
3670         Real_t dzv86 = dzv28 ;
3671
3672         Real_t dxv87 = dxv86 ;
3673         Real_t dyv87 = dyv29 ;
3674         Real_t dzv87 = dzv28 ;
3675
3676         Real_t dxv88 = dxv87 ;
3677         Real_t dyv88 = dyv29 ;
3678         Real_t dzv88 = dzv28 ;
3679
3680         Real_t dxv89 = dxv88 ;
3681         Real_t dyv89 = dyv29 ;
3682         Real_t dzv89 = dzv28 ;
3683
3684         Real_t dxv90 = dxv89 ;
3685         Real_t dyv90 = dyv29 ;
3686         Real_t dzv90 = dzv28 ;
3687
3688         Real_t dxv91 = dxv90 ;
3689         Real_t dyv91 = dyv29 ;
3690         Real_t dzv91 = dzv28 ;
3691
3692         Real_t dxv92 = dxv91 ;
3693         Real_t dyv92 = dyv29 ;
3694         Real_t dzv92 = dzv28 ;
3695
3696         Real_t dxv93 = dxv92 ;
3697         Real_t dyv93 = dyv29 ;
3698         Real_t dzv93 = dzv28 ;
3699
3700         Real_t dxv94 = dxv93 ;
3701         Real_t dyv94 = dyv29 ;
3702         Real_t dzv94 = dzv28 ;
3703
3704         Real_t dxv95 = dxv94 ;
3705         Real_t dyv95 = dyv29 ;
3706         Real_t dzv95 = dzv28 ;
3707
3708         Real_t dxv96 = dxv95 ;
3709         Real_t dyv96 = dyv29 ;
3710         Real_t dzv96 = dzv28 ;
3711
3712         Real_t dxv97 = dxv96 ;
3713         Real_t dyv97 = dyv29 ;
3714         Real_t dzv97 = dzv28 ;
3715
3716         Real_t dxv98 = dxv97 ;
3717         Real_t dyv98 = dyv29 ;
3718         Real_t dzv98 = dzv28 ;
3719
3720         Real_t dxv99 = dxv98 ;
3721         Real_t dyv99 = dyv29 ;
3722         Real_t dzv99 = dzv28 ;
3723
3724         Real_t dxv100 = dxv99 ;
3725         Real_t dyv100 = dyv29 ;
3726         Real_t dzv100 = dzv28 ;
3727
3728         Real_t dxv101 = dxv100 ;
3729         Real_t dyv101 = dyv29 ;
3730         Real_t dzv101 = dzv28 ;
3731
3732         Real_t dxv102 = dxv101 ;
3733         Real_t dyv102 = dyv29 ;
3734         Real_t dzv102 = dzv28 ;
3735
3736         Real_t dxv103 = dxv102 ;
3737         Real_t dyv103 = dyv29 ;
3738         Real_t dzv103 = dzv28 ;
3739
3740         Real_t dxv104 = dxv103 ;
3741         Real_t dyv104 = dyv29 ;
3742         Real_t dzv104 = dzv28 ;
3743
3744         Real_t dxv105 = dxv104 ;
3745         Real_t dyv105 = dyv29 ;
3746         Real_t dzv105 = dzv28 ;
3747
3748         Real_t dxv106 = dxv105 ;
3749         Real_t dyv106 = dyv29 ;
3750         Real_t dzv106 = dzv28 ;
3751
3752         Real_t dxv107 = dxv106 ;
3753         Real_t dyv107 = dyv29 ;
3754         Real_t dzv107 = dzv28 ;
3755
3756         Real_t dxv108 = dxv107 ;
3757         Real_t dyv108 = dyv29 ;
3758         Real_t dzv108 = dzv28 ;
3759
3760         Real_t dxv109 = dxv108 ;
3761         Real_t dyv109 = dyv29 ;
3762         Real_t dzv109 = dzv28 ;
3763
3764         Real_t dxv110 = dxv109 ;
3765         Real_t dyv110 = dyv29 ;
3766         Real_t dzv110 = dzv28 ;
3767
3768         Real_t dxv111 = dxv110 ;
3769         Real_t dyv111 = dyv29 ;
3770         Real_t dzv111 = dzv28 ;
3771
3772         Real_t dxv112 = dxv111 ;
3773         Real_t dyv112 = dyv29 ;
3774         Real_t dzv112 = dzv28 ;
3775
3776         Real_t dxv113 = dxv112 ;
3777         Real_t dyv113 = dyv29 ;
3778         Real_t dzv113 = dzv28 ;
3779
3780         Real_t dxv114 = dxv113 ;
3781         Real_t dyv114 = dyv29 ;
3782         Real_t dzv114 = dzv28 ;
3783
3784         Real_t dxv115 = dxv114 ;
3785         Real_t dyv115 = dyv29 ;
3786         Real_t dzv115 = dzv28 ;
3787
3788         Real_t dxv116 = dxv115 ;
3789         Real_t dyv116 = dyv29 ;
3790         Real_t dzv116 = dzv28 ;
3791
3792         Real_t dxv117 = dxv116 ;
3793         Real_t dyv117 = dyv29 ;
3794         Real_t dzv117 = dzv28 ;
3795
3796         Real_t dxv118 = dxv117 ;
3797         Real_t dyv118 = dyv29 ;
3798         Real_t dzv118 = dzv28 ;
3799
3800         Real_t dxv119 = dxv118 ;
3801         Real_t dyv119 = dyv29 ;
3802         Real_t dzv119 = dzv28 ;
3803
3804         Real_t dxv120 = dxv119 ;
3805         Real_t dyv120 = dyv29 ;
3806         Real_t dzv120 = dzv28 ;
3807
3808         Real_t dxv121 = dxv120 ;
3809         Real_t dyv121 = dyv29 ;
3810         Real_t dzv121 = dzv28 ;
3811
3812         Real_t dxv122 = dxv121 ;
3813         Real_t dyv122 = dyv29 ;
3814         Real_t dzv122 = dzv28 ;
3815
3816         Real_t dxv123 = dxv122 ;
3817         Real_t dyv123 = dyv29 ;
3818         Real_t dzv123 = dzv28 ;
3819
3820         Real_t dxv124 = dxv123 ;
3821         Real_t dyv124 = dyv29 ;
3822         Real_t dzv124 = dzv28 ;
3823
3824         Real_t dxv125 = dxv124 ;
3825         Real_t dyv125 = dyv29 ;
3826         Real_t dzv125 = dzv28 ;
3827
3828         Real_t dxv126 = dxv125 ;
3829         Real_t dyv126 = dyv29 ;
3830         Real_t dzv126 = dzv28 ;
3831
3832         Real_t dxv127 = dxv126 ;
3833         Real_t dyv127 = dyv29 ;
3834         Real_t dzv127 = dzv28 ;
3835
3836         Real_t dxv128 = dxv127 ;
3837         Real_t dyv128 = dyv29 ;
3838         Real_t dzv128 = dzv28 ;
3839
3840         Real_t dxv129 = dxv128 ;
3841         Real_t dyv129 = dyv29 ;
3842         Real_t dzv129 = dzv28 ;
3843
3844         Real_t dxv130 = dxv129 ;
3845         Real_t dyv130 = dyv29 ;
3846         Real_t dzv130 = dzv28 ;
3847
3848         Real_t dxv131 = dxv130 ;
3849         Real_t dyv131 = dyv29 ;
3850         Real_t dzv131 = dzv28 ;
3851
3852         Real_t dxv132 = dxv131 ;
3853         Real_t dyv132 = dyv29 ;
3854         Real_t dzv132 = dzv28 ;
3855
3856         Real_t dxv133 = dxv132 ;
3857         Real_t dyv133 = dyv29 ;
3858         Real_t dzv133 = dzv28 ;
3859
3860         Real_t dxv134 = dxv133 ;
3861         Real_t dyv134 = dyv29 ;
3862         Real_t dzv134 = dzv28 ;
3863
3864         Real_t dxv135 = dxv134 ;
3865         Real_t dyv135 = dyv29 ;
3866         Real_t dzv135 = dzv28 ;
3867
3868         Real_t dxv136 = dxv135 ;
3869         Real_t dyv136 = dyv29 ;
3870         Real_t dzv136 = dzv28 ;
3871
3872         Real_t dxv137 = dxv136 ;
3873         Real_t dyv137 = dyv29 ;
3874         Real_t dzv137 = dzv28 ;
3875
3876         Real_t dxv138 = dxv137 ;
3877         Real_t dyv138 = dyv29 ;
3878         Real_t dzv138 = dzv28 ;
3879
3880         Real_t dxv139 = dxv138 ;
3881         Real_t dyv139 = dyv29 ;
3882         Real_t dzv139 = dzv28 ;
3883
3884         Real_t dxv140 = dxv139 ;
3885         Real_t dyv140 = dyv29 ;
3886         Real_t dzv140 = dzv28 ;
3887
3888         Real_t dxv141 = dxv140 ;
3889         Real_t dyv141 = dyv29 ;
3890         Real_t dzv141 = dzv28 ;
3891
3892         Real_t dxv142 = dxv141 ;
3893         Real_t dyv142 = dyv29 ;
3894         Real_t dzv142 = dzv28 ;
3895
3896         Real_t dxv143 = dxv142 ;
3897         Real_t dyv143 = dyv29 ;
3898         Real_t dzv143 = dzv28 ;
3899
3900         Real_t dxv144 = dxv143 ;
3901         Real_t dyv144 = dyv29 ;
3902         Real_t dzv144 = dzv28 ;
3903
3904         Real_t dxv145 = dxv144 ;
3905         Real_t dyv145 = dyv29 ;
3906         Real_t dzv145 = dzv28 ;
3907
3908         Real_t dxv146 = dxv145 ;
3909         Real_t dyv146 = dyv29 ;
3910         Real_t dzv146 = dzv28 ;
3911
3912         Real_t dxv147 = dxv146 ;
3913         Real_t dyv147 = dyv29 ;
3914         Real_t dzv147 = dzv28 ;
3915
3916         Real_t dxv148 = dxv147 ;
3917         Real_t dyv148 = dyv29 ;
3918         Real_t dzv148 = dzv28 ;
3919
3920         Real_t dxv149 = dxv148 ;
3921         Real_t dyv149 = dyv29 ;
3922         Real_t dzv149 = dzv28 ;
3923
3924         Real_t dxv150 = dxv149 ;
3925         Real_t dyv150 = dyv29 ;
3926         Real_t dzv150 = dzv28 ;
3927
3928         Real_t dxv151 = dxv150 ;
3929         Real_t dyv151 = dyv29 ;
3930         Real_t dzv151 = dzv28 ;
3931
3932         Real_t dxv152 = dxv151 ;
3933         Real_t dyv152 = dyv29 ;
3934         Real_t dzv152 = dzv28 ;
3935
3936         Real_t dxv153 = dxv152 ;
3937         Real_t dyv153 = dyv29 ;
3938         Real_t dzv153 = dzv28 ;
3939
3940         Real_t dxv154 = dxv153 ;
3941         Real_t dyv154 = dyv29 ;
3942         Real_t dzv154 = dzv28 ;
3943
3944         Real_t dxv155 = dxv154 ;
3945         Real_t dyv155 = dyv29 ;
3946         Real_t dzv155 = dzv28 ;
3947
3948         Real_t dxv156 = dxv155 ;
3949         Real_t dyv156 = dyv29 ;
3950         Real_t dzv156 = dzv28 ;
3951
3952         Real_t dxv157 = dxv156 ;
3953         Real_t dyv157 = dyv29 ;
3954         Real_t dzv157 = dzv28 ;
3955
3956         Real_t dxv158 = dxv157 ;
3957         Real_t dyv158 = dyv29 ;
3958         Real_t dzv158 = dzv28 ;
3959
3960         Real_t dxv159 = dxv158 ;
3961         Real_t dyv159 = dyv29 ;
3962         Real_t dzv159 = dzv28 ;
3963
3964         Real_t dxv160 = dxv159 ;
3965         Real_t dyv160 = dyv29 ;
3966         Real_t dzv160 = dzv28 ;
3967
3968         Real_t dxv161 = dxv160 ;
3969         Real_t dyv161 = dyv29 ;
3970         Real_t dzv161 = dzv28 ;
3971
3972         Real_t dxv162 = dxv161 ;
3973         Real_t dyv162 = dyv29 ;
3974         Real_t dzv162 = dzv28 ;
3975
3976         Real_t dxv163 = dxv162 ;
3977         Real_t dyv163 = dyv29 ;
3978         Real_t dzv163 = dzv28 ;
3979
3980         Real_t dxv164 = dxv163 ;
3981         Real_t dyv164 = dyv29 ;
3982         Real_t dzv164 = dzv28 ;
3983
3984         Real_t dxv165 = dxv164 ;
3985         Real_t dyv165 = dyv29 ;
3986         Real_t dzv165 = dzv28 ;
3987
3988         Real_t dxv166 = dxv165 ;
3989         Real_t dyv166 = dyv29 ;
3990         Real_t dzv166 = dzv28 ;
3991
3992         Real_t dxv167 = dxv166 ;
3993         Real_t dyv167 = dyv29 ;
3994         Real_t dzv167 = dzv28 ;
3995
3996         Real_t dxv168 = dxv167 ;
3997         Real_t dyv168 = dyv29 ;
3998         Real_t dzv168 = dzv28 ;
3999
4000         Real_t dxv169 = dxv168 ;
4001         Real_t dyv169 = dyv29 ;
4002         Real_t dzv169 = dzv28 ;
4003
4004         Real_t dxv170 = dxv169 ;
4005         Real_t dyv170 = dyv29 ;
4006         Real_t dzv170 = dzv28 ;
4007
4008         Real_t dxv171 = dxv170 ;
4009         Real_t dyv171 = dyv29 ;
4010         Real_t dzv171 = dzv28 ;
4011
4012         Real_t dxv172 = dxv171 ;
4013         Real_t dyv172 = dyv29 ;
4014         Real_t dzv172 = dzv28 ;
4015
4016         Real_t dxv173 = dxv172 ;
4017         Real_t dyv173 = dyv29 ;
4018         Real_t dzv173 = dzv28 ;
4019
4020         Real_t dxv174 = dxv173 ;
4021         Real_t dyv174 = dyv29 ;
4022         Real_t dzv174 = dzv28 ;
4023
4024         Real_t dxv175 = dxv174 ;
4025         Real_t dyv175 = dyv29 ;
4026         Real_t dzv175 = dzv28 ;
4027
4028         Real_t dxv176 = dxv175 ;
4029         Real_t dyv176 = dyv29 ;
4030         Real_t dzv176 = dzv28 ;
4031
4032         Real_t dxv177 = dxv176 ;
4033         Real_t dyv177 = dyv29 ;
4034         Real_t dzv177 = dzv28 ;
4035
4036         Real_t dxv178 = dxv177 ;
4037         Real_t dyv178 = dyv29 ;
4038         Real_t dzv178 = dzv28 ;
4039
4040         Real_t dxv179 = dxv178 ;
4041         Real_t dyv179 = dyv29 ;
4042         Real_t dzv179 = dzv28 ;
4043
4044         Real_t dxv180 = dxv179 ;
4045         Real_t dyv180 = dyv29 ;
4046         Real_t dzv180 = dzv28 ;
4047
4048         Real_t dxv181 = dxv180 ;
4049         Real_t dyv181 = dyv29 ;
4050         Real_t dzv181 = dzv28 ;
4051
4052         Real_t dxv182 = dxv181 ;
4053         Real_t dyv182 = dyv29 ;
4054         Real_t dzv182 = dzv28 ;
4055
4056         Real_t dxv183 = dxv182 ;
4057         Real_t dyv183 = dyv29 ;
4058         Real_t dzv183 = dzv28 ;
4059
4060         Real_t dxv184 = dxv183 ;
4061         Real_t dyv184 = dyv29 ;
4062         Real_t dzv184 = dzv28 ;
4063
4064         Real_t dxv185 = dxv184 ;
4065         Real_t dyv185 = dyv29 ;
4066         Real_t dzv185 = dzv28 ;
4067
4068         Real_t dxv186 = dxv185 ;
4069         Real_t dyv186 = dyv29 ;
4070         Real_t dzv186 = dzv28 ;
4071
4072         Real_t dxv187 = dxv186 ;
4073         Real_t dyv187 = dyv29 ;
4074         Real_t dzv187 = dzv28 ;
4075
4076         Real_t dxv188 = dxv187 ;
4077         Real_t dyv188 = dyv29 ;
4078         Real_t dzv188 = dzv28 ;
4079
4080         Real_t dxv189 = dxv188 ;
4081         Real_t dyv189 =
```

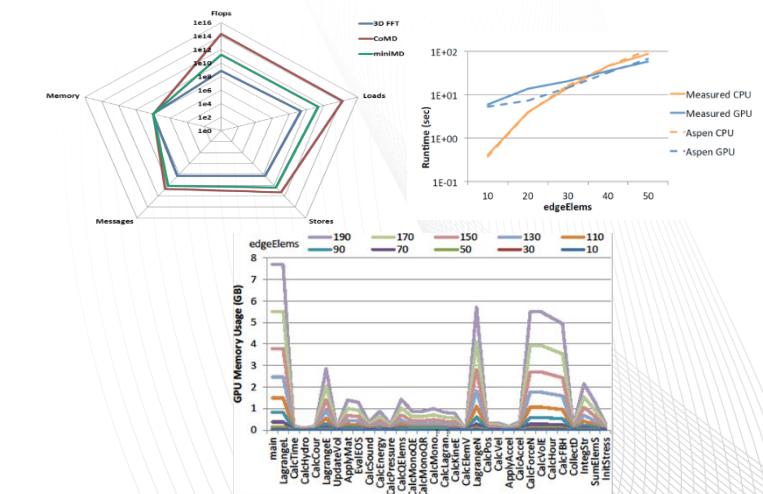
Aspen code

```

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 [98] as dp, simd
164     // delvk delvkx
165     flops [9 + 8 + 3 + 30 + 5] as dp, simd
166     stores [wordSize] to delvk_xeta
167     // delxi delvi
168     flops [9 + 8 + 3 + 30 + 5] as dp, simd
169     stores [wordSize] to delxi_xi
170     // deljk and delvjk
171     flops [9 + 8 + 3 + 30 + 5] as dp, simd
172     stores [wordSize] to delv_eta
173   }
174 }
```

Model Uses

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

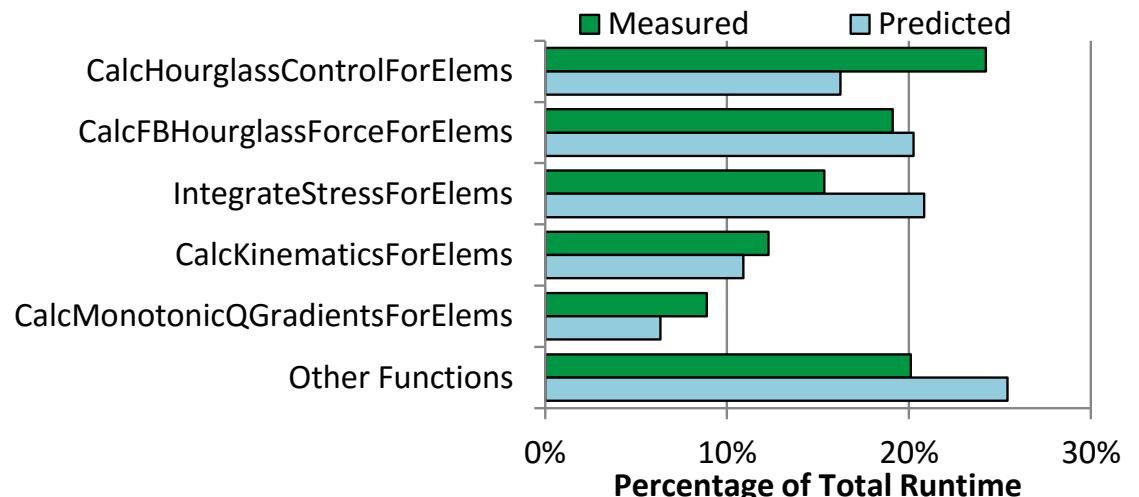


View Aspen performance models as normal performance analysis output with Gprof

Flat profile:						
% time	cum sec	self sec	calls	self ms/call	total ms/call	name
86.91	370.76	370.76	30	12358.52	12358.52	fft3d.localFFT
10.03	413.54	42.78	20	2139.09	2139.09	fft3d.exchange
3.06	426.57	13.03	20	651.71	651.71	fft3d.shuffle
0.00	426.57	0.00	10	0.03	0.03	exchange
0.00	426.57	0.00	10	0.03	0.03	buildNLlist
0.00	426.57	0.00	10	0.01	0.01	ljForce
0.00	426.57	0.00	30	0.00	0.00	integrate
0.00	426.57	0.00	10	0.00	42657.18	fft
0.00	426.57	0.00	10	0.00	42657.18	fft3d.main
0.00	426.57	0.00	1	0.00	426572.70	main

Call graph:		
index	%time	self children name
[1]	100.0	0.00 426.57 main [1]
		0.00 0.00 buildNLlist [8]
		0.00 0.00 exchange [7]
		0.00 426.57 fft [2]
		0.00 0.00 integrate [10]
		0.00 0.00 ljForce [9]
[2]	100.0	0.00 426.57 main [1]
		0.00 426.57 fft [2]
		0.00 426.57 fft3d.main [3]
[3]	100.0	0.00 426.57 fft [2]
		0.00 426.57 fft3d.main [3]
		42.78 0.00 fft3d.exchange [5]
		370.76 0.00 fft3d.localFFT [4]
		13.03 0.00 fft3d.shuffle [6]
[4]	86.9	370.76 0.00 fft3d.main [3]
		0.00 426.57 fft3d.localFFT [4]
[5]	10.0	42.78 0.00 fft3d.main [3]
		0.00 426.57 fft3d.exchange [5]
[6]	3.1	13.03 0.00 fft3d.main [3]
		0.00 426.57 fft3d.shuffle [6]
[7]	0.0	0.00 426.57 main [1]
		0.00 0.00 exchange [7]
[8]	0.0	0.00 426.57 main [1]
		0.00 0.00 buildNLlist [8]
[9]	0.0	0.00 426.57 main [1]
		0.00 0.00 ljForce [9]
[10]	0.0	0.00 426.57 main [1]
		0.00 0.00 integrate [10]

Not from gprof
but rather aspen
performance
model



Aspen Model User Queries

Benchmark	Runtime Order
BACKPROP	$H * O + H * I$
BFS	$nodes + edges$
CFD	$nelt * ndim$
CG	$nrow + ncol$
HOTSPOT	$simTime * rows * cols$
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.

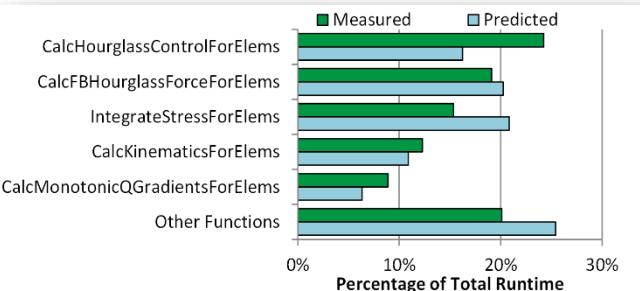


Fig. 8: GPU Memory Usage of each Function in LULESH, where the memory usage of a function is inclusive; value for a parent function includes data accessed by its child functions in the call graph.

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
VolDiver	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
CalcElemVelocityGradient	0.13
CalcKinematicsForElems	0.24
CalcLagrangeElements	0.24
CalcMonotonicOGradientsForElems	0.46

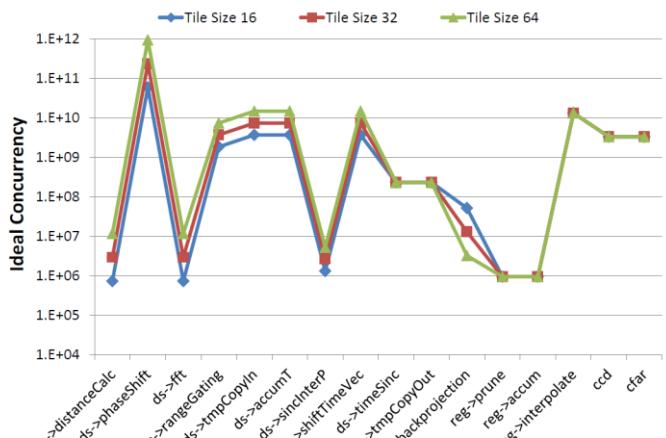


Figure 1: A plot of idealized concurrency by chronological phase in the digital spotlighting application model.

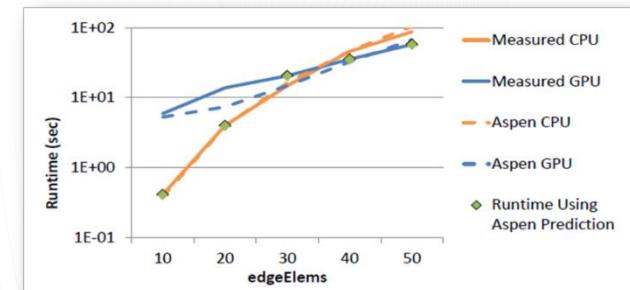


Fig. 7: Measured and predicted runtime of the entire LULESH program on CPU and GPU, including measured runtimes using the automatically predicted optimal target device at each size.

Scheduling GPU Offloads with Aspen Performance Models

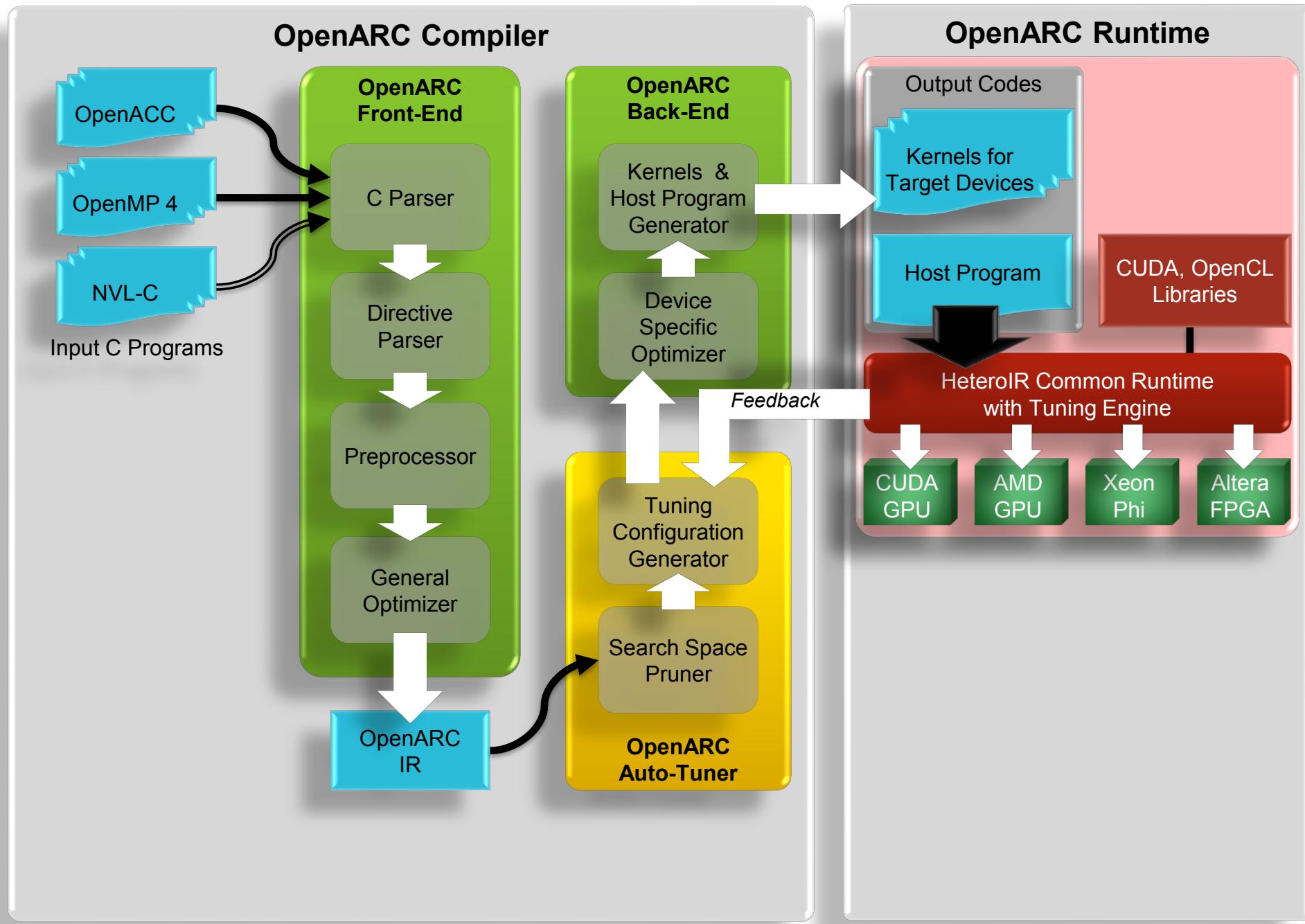


Should the application offload kernel to GPU or not?

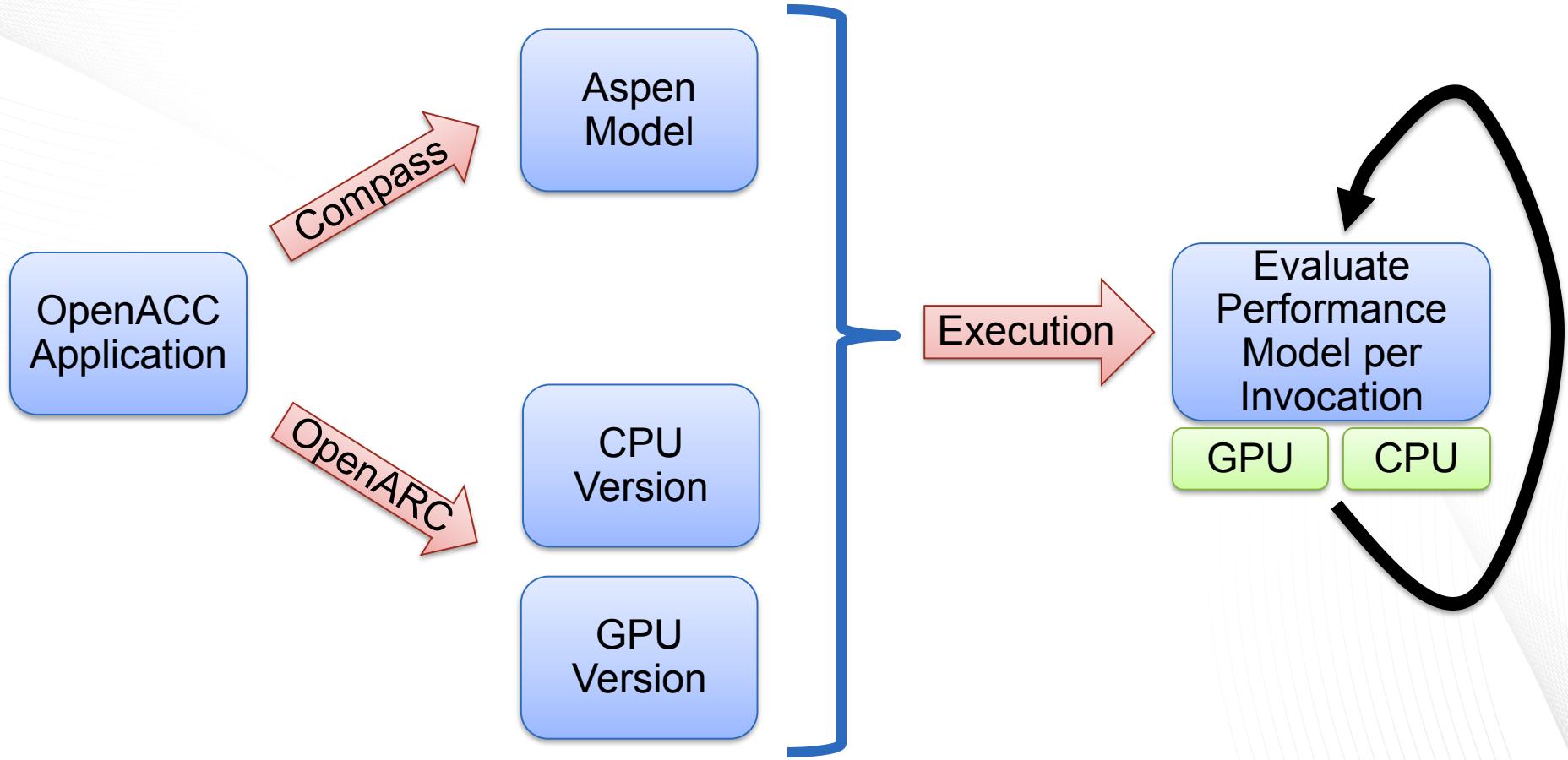
- Simply offloading all computation is not smart
- Depends
 - When it is ‘small’, run the computation on the host CPU,
 - Otherwise, send it to the GPU
 - Expense of data movement over PCIe (twice) and launch GPU kernels?
- Portability?
 - Need to account for performance, working set size, data transfer costs, ...

Listing 1: Input OpenACC Matrix Multiplication Code

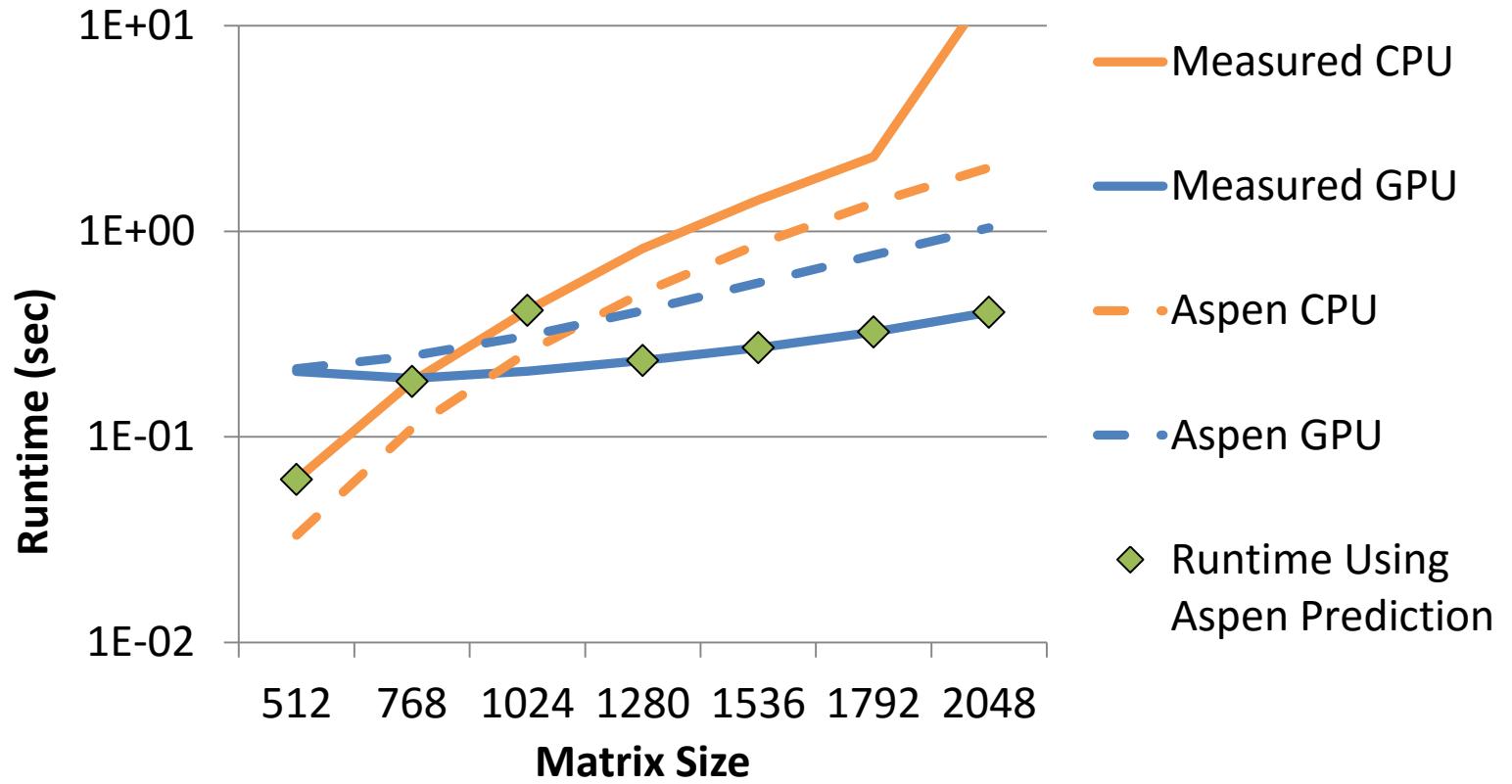
```
1 int N = 1024;
2 void matmul(float *a, float *b, float *c){ int i, j, k ;
3 #pragma acc kernels loop gang copyout(a[0:(N*N)]) \
4 copyin(b[0:(N*N)],c[0:(N*N)])
5 for (i=0; i<N; i++){
6 #pragma acc loop worker
7     for (j=0; j<N; j++) { float sum = 0.0 ;
8         for (k=0; k<N; k++) {sum+=b[i*N+k]*c[k*N+j];}
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()
```



Process



Matrix Multiply



LULESH: Runtime and Working Set Size Predictions

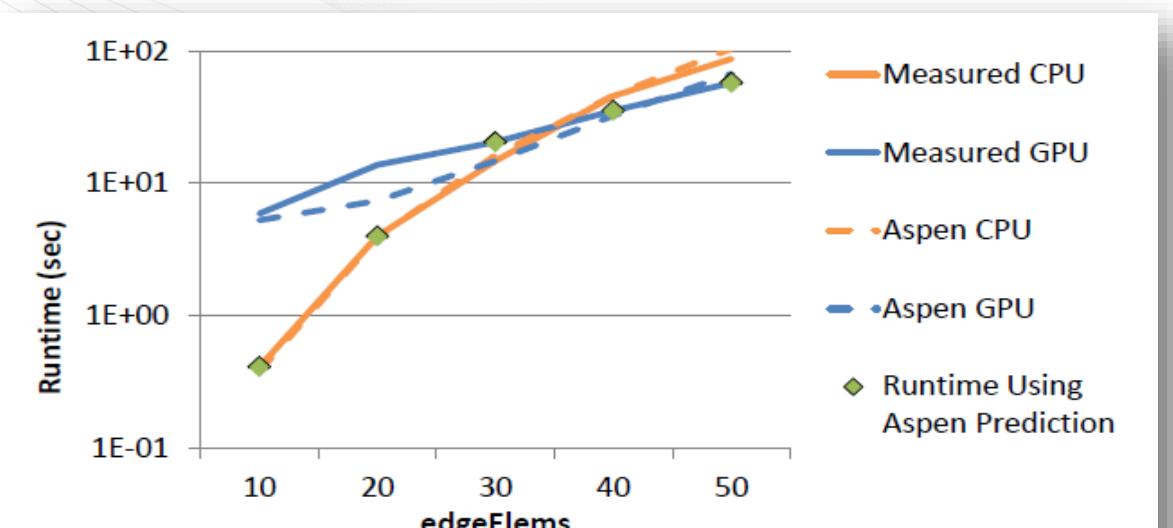


Fig. 7: Measured and predicted runtime of the entire LULESH program on CPU and GPU, including measured runtimes using the automatically predicted optimal target device at each size.

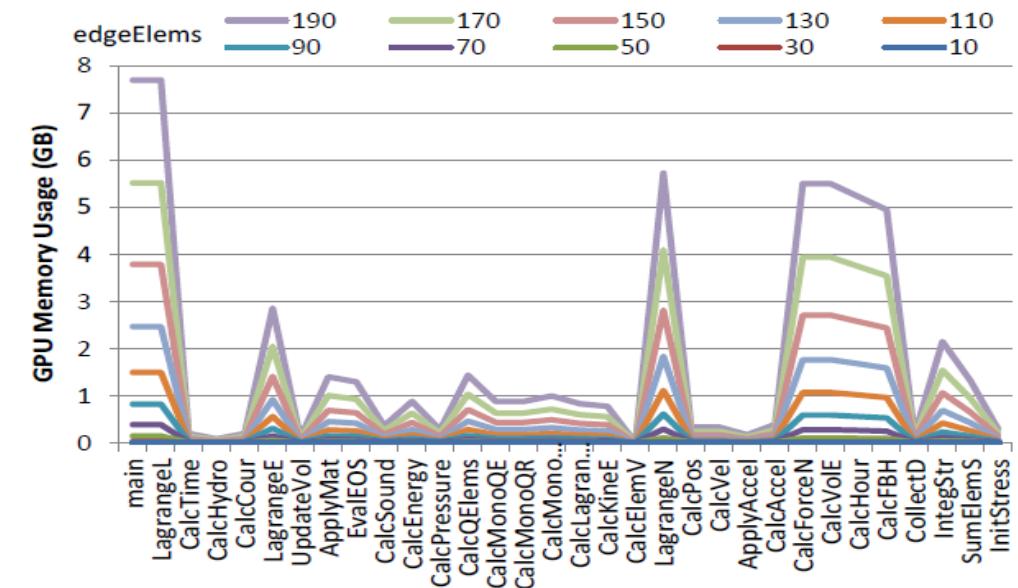
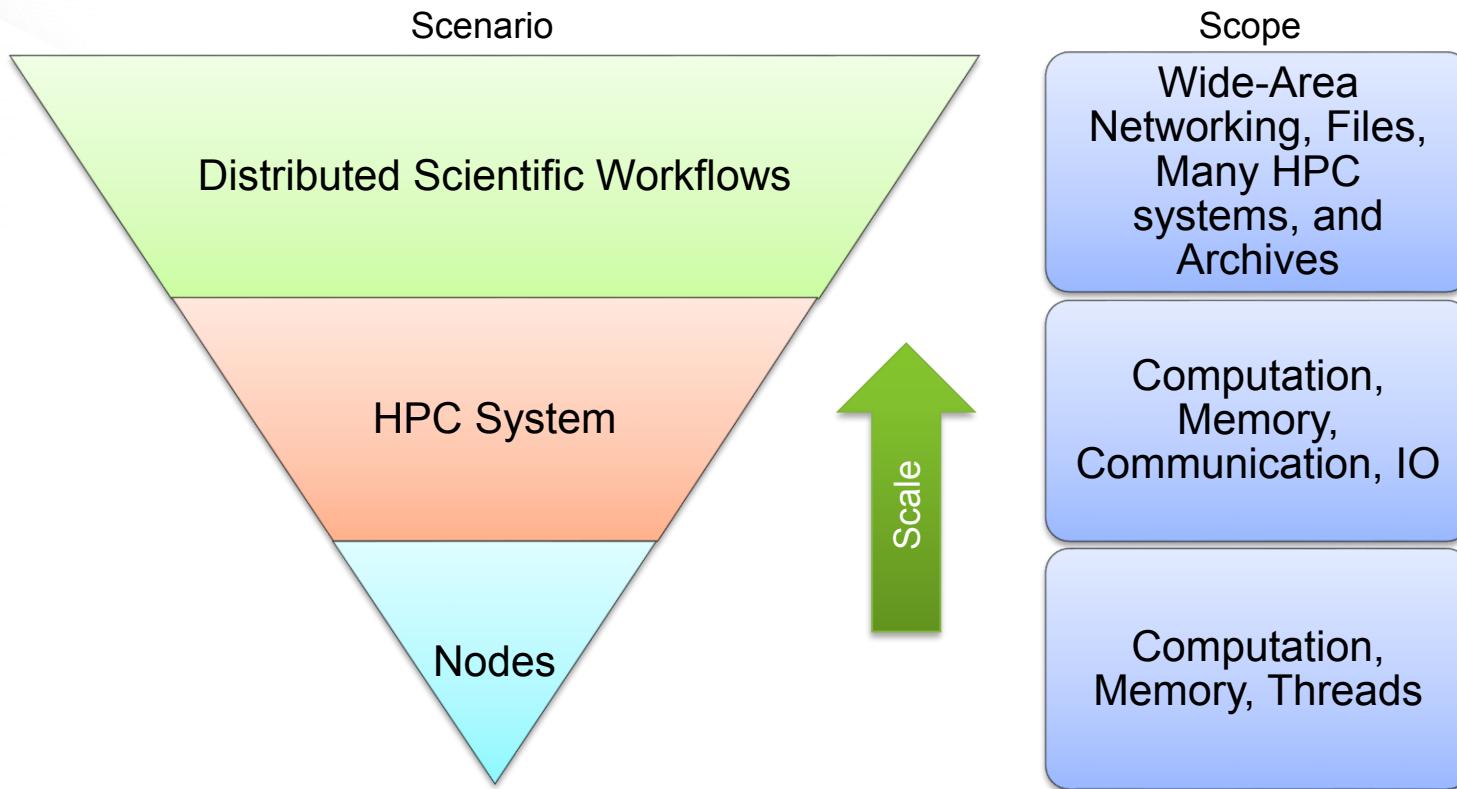


Fig. 8: GPU Memory Usage of each Function in LULESH, where the memory usage of a function is inclusive; value for a parent function includes data accessed by its child functions in the call graph.

Using Aspen for Distributed Workflows

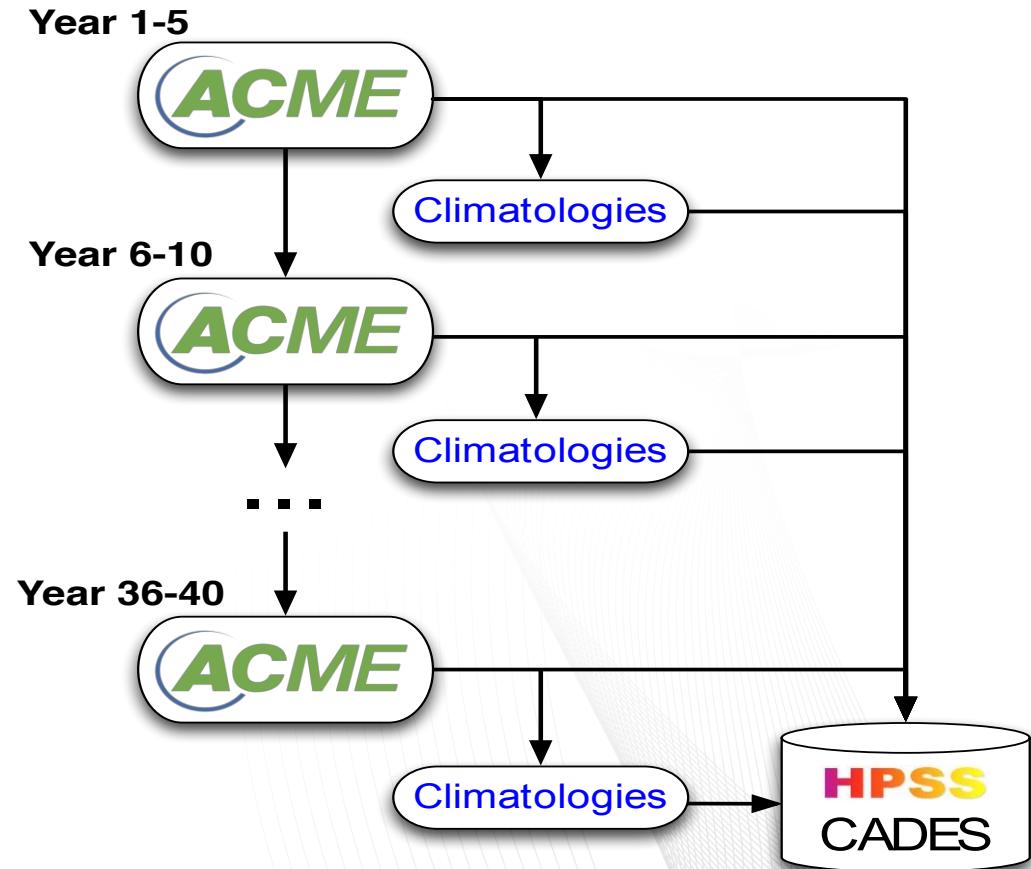


Aspen allows Multiresolution Modeling



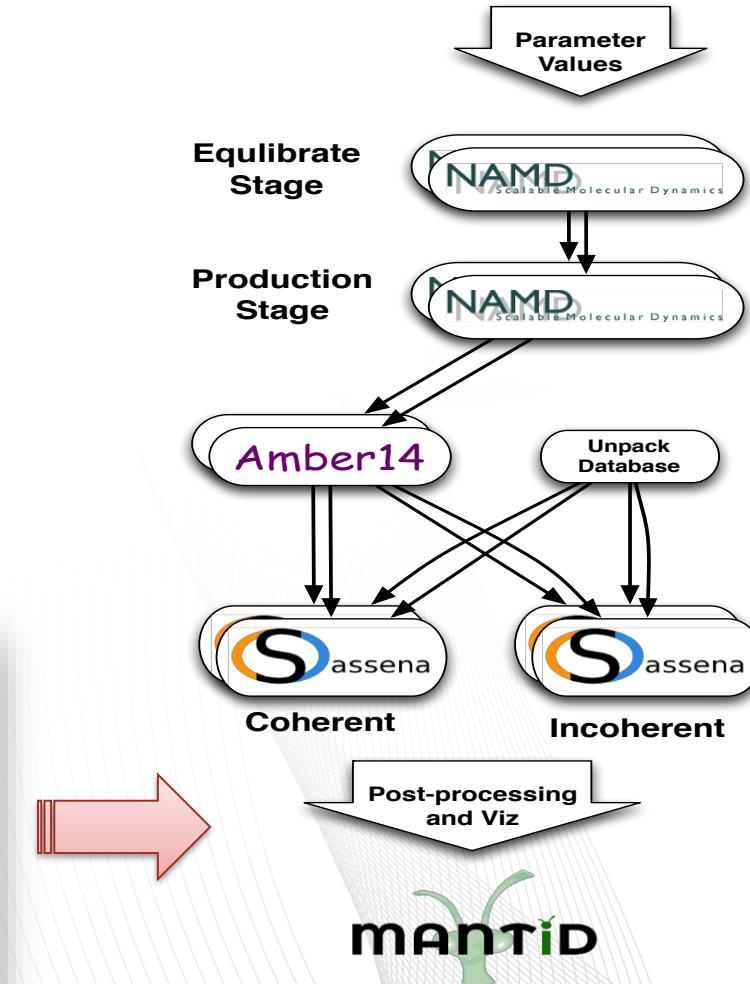
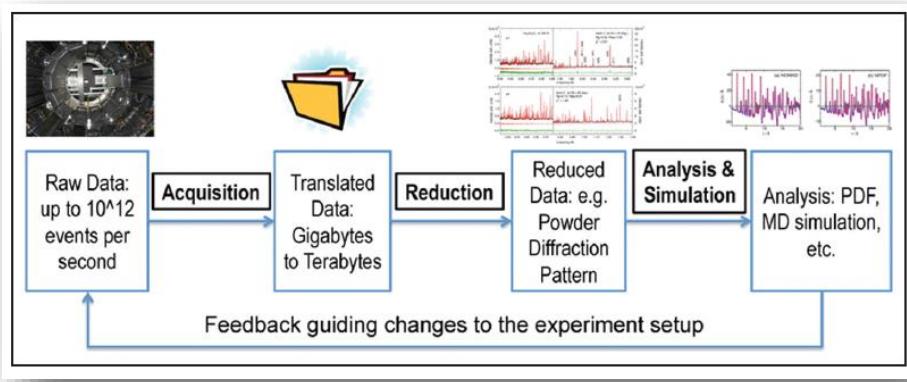
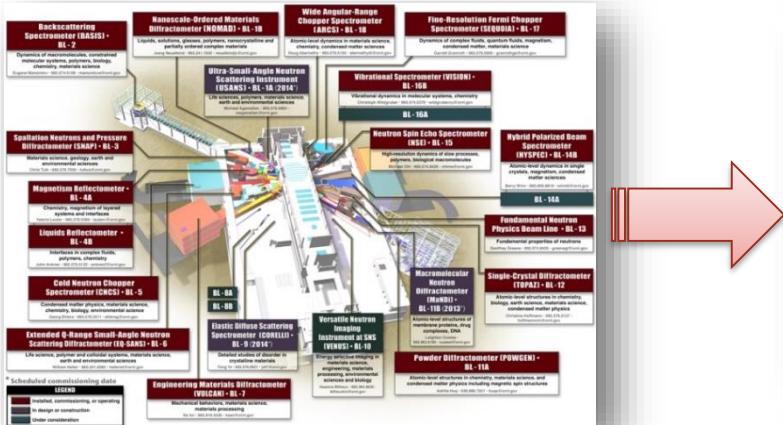
Simulation: ACME Workflows

- Accelerated Climate Modeling for Energy (ACME)
- Coupled climate models with ocean, land, atmosphere and ice
- Climatologies and diagnostics give summaries of data
- Each stage of the workflow runs the ACME model for a few timesteps—helps keep simulations within batch queue limits
- Running on Hopper @ NERSC and Titan @ OLCF

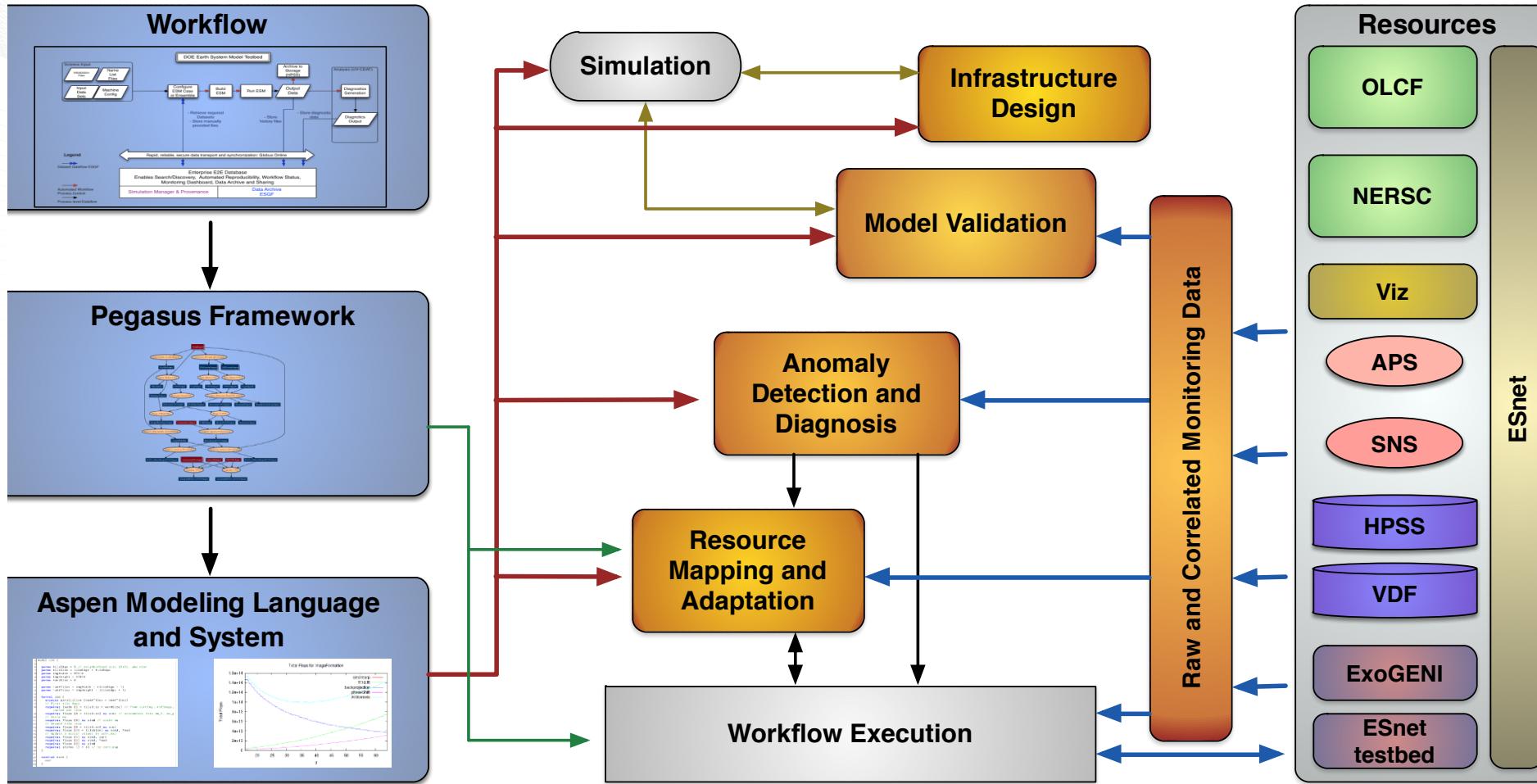


Experimental Data: SNS Workflows

- Spallation Neutron Source
- Parameter sweep of molecular dynamics and neutron scattering
- Used to identify parameters that fit experimental data from SNS
- Currently being used for real science problems
- Large runs use 20 parameter values and require ~400,000 CPU hours
- Running on Hopper @ NERSC and coming to Titan @ ORNL



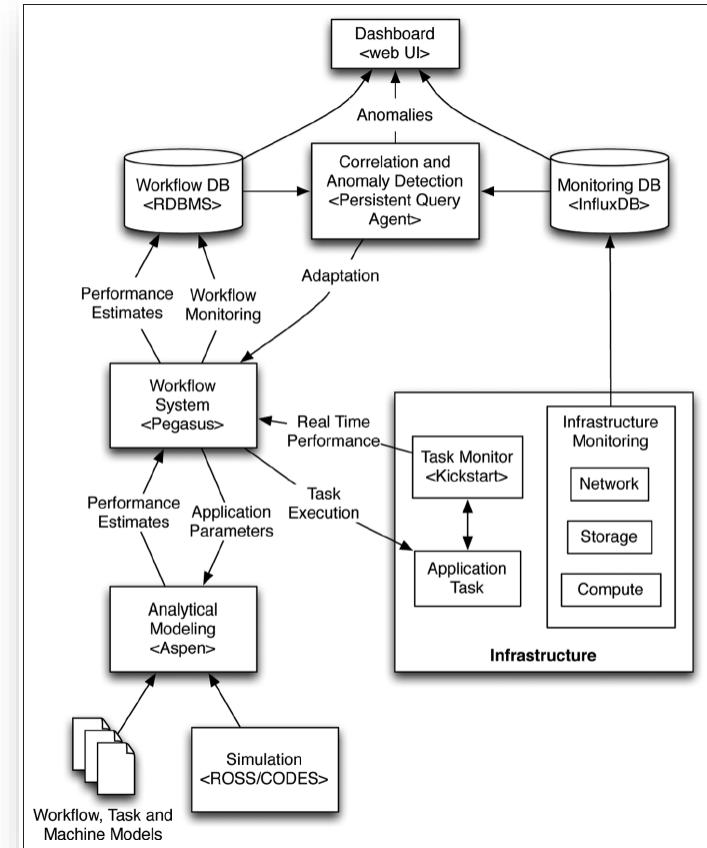
PANORAMA Overview



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

```
1 kernel main
2 {
3     par {
4         seq {
5             call namd.eq_200()
6             call namd.prod_200()
7         }
8         seq {
9             call namd.eq_290()
10            call namd.prod_290()
11        }
12    }
13    par {
14        call unpack_database()
15        call ptraj_200()
16        call ptraj_290()
17    }
18    par {
19        call sassena.incoh_200()
20        call sassena.coh_200()
21        call sassena.incoh_290()
22        call sassena.coh_290()
23    }
24 }
```

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



Black Box Modeling of NAMD for SNS Workflow

MD template model

```
model NAMD_Template {
    // application parameters
    // (defined in the input file)
    param nAtoms      = 1e6
    param nTimeSteps = 100
    // solve for these parameters
    // (within the given ranges)
    param c = 1 in 1 .. 1e18
    param d = 1 in 1 .. 1e18
    // application behavior:
    // execution and control flow
    kernel main
    {
        iterate [nTimeSteps] {
            execute {
                loads [c * nAtoms^2]
                flops [d * nAtoms]
            }
        }
    }
}
```



CSV data file with parameters
and runtimes

nAtoms	nTimeSteps	nCores	machine	runtime
1e6	100	144	exogeni	384.2
1e6	100	144	hopper	340.1
1e6	150	144	hopper	482.9



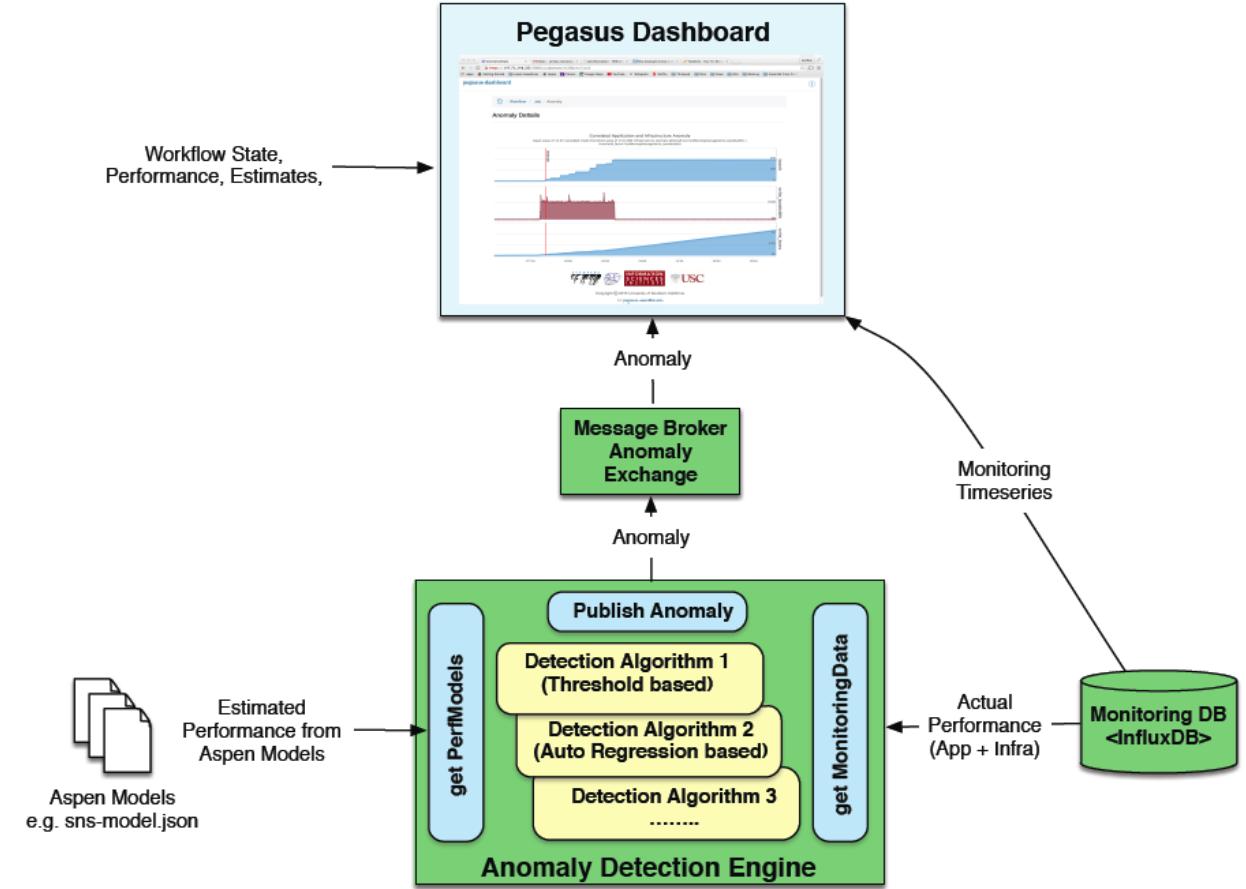
Concrete NAMD model

```
model NAMD_Equilibrat {
    // NAMD input parameters
    param nAtoms      = 1e6
    param nTimeSteps = 100
    // calculation-specific constants
    param c = 402.1
    param d = 10.95
    // NAMD application behavior
    kernel main
    {
        iterate [nTimeSteps] {
            execute {
                loads [c * nAtoms^2]
                flops [d * nAtoms]
            }
        }
    }
}
```

- nAtoms and nTimeSteps defined in template application model and CSV input data
- nCores defined in machine models and CSV input data
- solves for c and d, filling out a concrete application model for that problem
- new predictions can still vary nAtoms, nTimeSteps, and nCores

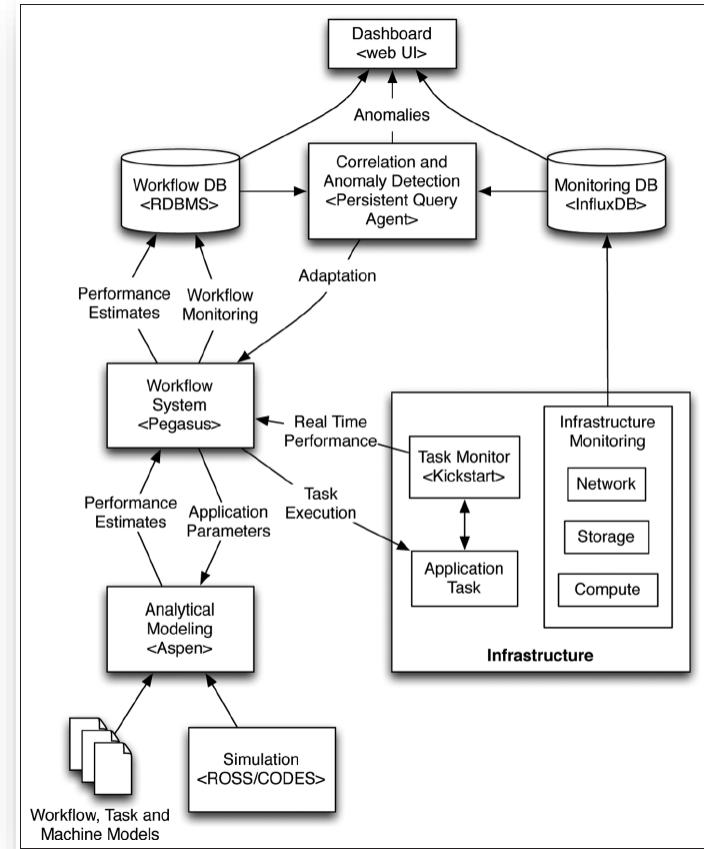
Aspen Model Use Case: Anomaly Detection

- Aspen models embedded into workflow
Pegasus DAX model
 - Evaluated in real time to compare with
Pegasus monitoring data
 - Highlight differences on Pegasus
Dashboard
-
- Status:
 - Works well for individual components
 - Understanding global predictions
 - Several components not modeled: queue times, I/O subsystem



Aspen Model Use Case: Resource Planning

- Augment Pegasus resource planner to include Aspen performance predictions
 - Extend Pegasus DAX with Aspen hooks
 - Catalog of relevant application models
 - Catalog of relevant resources
- During planning and mapping phases, optimize schedule with ‘best’ global resources for TTS, Cost, Power, etc
- Status
 - Aspen predictors completed
 - Currently augmenting Pegasus planner/mapper with callouts to Aspen predictors



Summary

- Resource scheduling is a seminal problem for computing ... and it is becoming much more difficult
 - Scheduling has many potential solutions
 - Algorithmic, Historical, application specific, etc
- Both architectures and applications are growing more complex
 - Trends dictate that this will get worse; not better
 - This complexity creates irregularity in computation, communication, and data movement
- Posit that we can use application-specific performance models to inform scheduling decisions
 - Aspen performance modeling language helps create models
 - Two recent experiments
 - GPU offload
 - Distributed scientific workflows



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



PMES Workshop @ SC16

- <https://j.mp/pmes2016>
- @SC16
- Position papers due June 17



Search this site

2016 Post-Moore's Era Supercomputing (PMES) Workshop Home

News
[Call For Position Papers - Submission Deadline - June 17](#)
[Invited Speakers](#)
[Photos](#)
[Program](#)
[Resources](#)
[Workshop Venue](#)
[Sitemap](#)

230
days until
PMES Workshop @ SC16

Co-located with [SC16](#) in Salt Lake City
Monday, 14 November 2016

Workshop URL: <http://j.mp/pmes2016>
CFP URL: <http://j.mp/pmes2016cfp>
Submission URL (EasyChair): <http://j.mp/pmes2016submissions>
Submission questions: pmes16@easychair.org

This interdisciplinary workshop is organized to explore the scientific issues, challenges, and opportunities for supercomputing beyond the scaling limits of Moore's Law, with the ultimate goal of keeping supercomputing at the forefront of computing technologies beyond the physical and conceptual limits of current systems. Continuing progress of supercomputing beyond the scaling limits of Moore's Law is likely to require a comprehensive re-thinking of technologies, ranging from innovative materials and devices, circuits, system architectures, programming systems, system software, and applications.

The workshop is designed to foster interdisciplinary dialog across the necessary spectrum of stakeholders: applications, algorithms, software, and hardware. Motivating workshop questions will include the following. "What technologies might prevail in the Post Moore's

News
[PMES Workshop Confirmed for SC16!](#)
[Submissions open for PMES Position Papers on April 17](#)

Important Dates

- Submission Site Opens: 17 April 2016
- Submission Deadline: 17 June 2016
- Notification Deadline: 17 August 2016
- Workshop: 14 November 2016

In cooperation with IEEE Computer Society
IEEE computer society

