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
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
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

Vendor Analysis Sim Exp Proto HW Prog Models HW Simulator Tools

Hardware Co-Design

HW Design

Computer Science Co-Design

Open Analysis Models Simulators Emulators

Domain/Alg Analysis

Proxy Apps

SW Solutions

HW Constraints

App Requirements

HW Constraints

Stack Analysis Prog models Tools Compilers Runtime OS, I/O, ...

System Software
# Prediction Techniques Ranked

<table>
<thead>
<tr>
<th>Technique</th>
<th>Speed</th>
<th>Ease</th>
<th>Flexibility</th>
<th>Accuracy</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc Analytical Models</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Structured Analytical Models</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Simulation – Functional</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Simulation – Cycle Accurate</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Hardware Emulation (FPGA)</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Similar hardware measurement</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Node Prototype</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Prototype at Scale</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Final System</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
# Prediction Techniques Ranked

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>Ease</th>
<th>Flexibility</th>
<th>Accuracy</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc Analytical Models</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Structured Analytical Models</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td><em>Aspen</em></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Simulation – Functional</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Simulation – Cycle Accurate</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Hardware Emulation (FPGA)</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Similar hardware measurement</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Node Prototype</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Prototype at Scale</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Final System</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Aspen: Abstract Scalable Performance Engineering Notation

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

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

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

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

---

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

Manual Example of LULESH

```c
kernel CalcMonotonicQGradients {
  execute [numelems] {
    loads [8 * indexWordSize] from nodalist
    // 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 vox0
    loads [wordSize] from vnew
    // dx, dy, etc.
    flops [90] as dp, sdp
    // delwv delxk
    flops [9 + 8 + 3 + 30 + 5] as dp, sdp
    stores [wordSize] to delw_xeta
    // delxi delvi
    flops [9 + 8 + 3 + 30 + 5] as dp, sdp
    stores [wordSize] to delxi_xi
    // delw] and delv]
    flops [9 + 8 + 3 + 30 + 5] as dp, sdp
    stores [wordSize] to delw_xet}
  }
}
```
Aspen allows Multiresolution Modeling

Scenario

- Distributed Scientific Workflows
- HPC System
- Nodes

Scope

- Wide-Area Networking, Files, Many HPC systems, and Archives
- Computation, Memory, Communication, IO
- Computation, Memory, Threads

Scale
Node Scale Modeling with COMPASS
COMPASS System Overview

- Detailed Workflow of the COMPASS Modeling Framework

Optional feedback for advanced users

MM example generated from COMPASS

```c
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()
```

```c
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
        } //end of kernel matmul
    kernel main { matmul() }
} //end of model MM
```
Input MatMul Code Annotated to Use an Alternative Algorithm

```c
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

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

```c
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
        stores [((1*aspen_param_double))] as stride(0)
        stores [2] as dp, simd
        stores [((1*aspen_param_double))] as stride(0)
        stores [2] as dp, simd
        stores [((1*aspen_param_double))] as stride(0)
        stores [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

<table>
<thead>
<tr>
<th>Code</th>
<th>FLOPS</th>
<th>LOADS</th>
<th>STORES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATMUL</td>
<td>15%</td>
<td>&lt;1%</td>
<td>1%</td>
</tr>
<tr>
<td>LAPLACE2D</td>
<td>7%</td>
<td>0%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>SRAD</td>
<td>17%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>JACOBI</td>
<td>6%</td>
<td>&lt;1%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>KMEANS</td>
<td>0%</td>
<td>0%</td>
<td>8%</td>
</tr>
<tr>
<td>LUD</td>
<td>5%</td>
<td>0%</td>
<td>2%</td>
</tr>
<tr>
<td>BFS</td>
<td>&lt;1%</td>
<td>11%</td>
<td>0%</td>
</tr>
<tr>
<td>HOTSPOT</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>LULESH</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

0% means that prediction fell between measurements from optimized and unoptimized runs of the code.
Model Scaling Validation (LULESH)

Bytes Stored vs. Edge Elements

- Measured (Unoptimized)
- Aspen Prediction
- Measured (Optimized)
Example Queries

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Runtime Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKPROP</td>
<td>$H \times O + H \times I$</td>
</tr>
<tr>
<td>BFS</td>
<td>$\text{nodes} \times \text{edges}$</td>
</tr>
<tr>
<td>CFD</td>
<td>$\text{nefl} \times \text{nldim}$</td>
</tr>
<tr>
<td>CG</td>
<td>$\text{nrow} \times \text{ncol}$</td>
</tr>
<tr>
<td>HOTSPOT</td>
<td>$\text{simtime} \times \text{rows} \times \text{cols}$</td>
</tr>
<tr>
<td>JACOBI</td>
<td>$m \times m \times \text{size}$</td>
</tr>
<tr>
<td>KMEANS</td>
<td>$\text{nAttr} \times \text{nClusters}$</td>
</tr>
<tr>
<td>LAPLACE2D</td>
<td>$n^2$</td>
</tr>
<tr>
<td>LUD</td>
<td>$\text{matrix}_\text{dim}^3$</td>
</tr>
<tr>
<td>MATMUL</td>
<td>$N \times M \times P$</td>
</tr>
<tr>
<td>NW</td>
<td>$\text{max}_\text{cols}^2$</td>
</tr>
<tr>
<td>SPMUL</td>
<td>$\text{size} + \text{nonzero}$</td>
</tr>
<tr>
<td>SRAD</td>
<td>$\text{niter} \times \text{rows} \times \text{cols}$</td>
</tr>
</tbody>
</table>

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

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        
VolumDer               | 1.50        
CalcElemVolumeDerivative | 0.33        
CalcFBHourglassForceForElems | 0.15        
CalcHourglassForceForElems | 0.17        
CalcHourglassControlForElems | 0.19        
CalcVolumeForceForElems | 0.18        
CalcForceForNodes      | 0.18        
CalcAccelerationForNodes | 0.04        
ApplyAccelerationBoundaryCond | 0.00        
CalcVelocityForNodes   | 0.15        
CalcPositionForNodes   | 0.13        
LogranteNode           | 0.18        
AreasFace              | 10.25       
CalcElemCharacteristicLength | 0.44        
CalcElemVelocityGradient | 0.13        
CalcKinematicsForElems | 0.24        
CalcLagrangeElements   | 0.24        
CalcMonotonicQGradientsForElems | 0.46        

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.

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
Aspen allows Multiresolution Modeling

Scenario

Distributed Scientific Workflows

HPC System

Nodes

Scope

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

Computation, Memory, Communication, IO

Computation, Memory, Threads

Scale
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

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

```plaintext
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()
  }
}
```
Status, statistics, timeline of jobs

Helps pinpoint errors
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

**DVF Defined**

Data Structure Vulnerability → \( DVF_d = N_{\text{error}} \times N_{\text{ha}} \)

Application Vulnerability → \( DVF_a = \sum_{i=1}^{n} DVF_{d_i} \)

Larger DVF indicates higher vulnerability, and vice versa.

\( N_{\text{error}} = FIT \times T \times S_d \)

\( N_{\text{ha}} \leftarrow \text{Hardware Access Pattern} \)

We focus on a specific hardware component, the main memory, in this work.
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
    end for
end procedure
```

Pseudocode

```
kernl 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

```
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
```

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

Resilience Modeling Results

```
Syntax Tree
```

```
Extended Parser
```

```
Extended Complier
```

```
35
```
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 everywhere!
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
  – 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/
  – US National Science Foundation Keeneland Project: http://keeneland.gatech.edu
  – US DARPA
  – NVIDIA CUDA Center of Excellence
# Notional Exascale Architecture Targets

(From Exascale Arch Report 2009)

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

Parallel I/O ??

# Today’s Status

<table>
<thead>
<tr>
<th>System attributes</th>
<th>Today</th>
<th>MIRA</th>
<th>Summit</th>
<th>CORAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>TITAN</td>
<td>MIRA</td>
<td>Summit</td>
<td>Aurora</td>
</tr>
<tr>
<td>System peak (PF)</td>
<td>27</td>
<td>10</td>
<td>150</td>
<td>180</td>
</tr>
<tr>
<td>Peak Power (MW)</td>
<td>9</td>
<td>4.8</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>Total system memory</td>
<td>710TB</td>
<td>768TB</td>
<td>2 PB DDR4 + HBM + 2.7 PB persistent memory</td>
<td>&gt;7 PB High Bandwidth On-Package Memory, local Memory and Persistent Memory</td>
</tr>
<tr>
<td>Node performance (TF)</td>
<td>1.452</td>
<td>0.204</td>
<td>&gt; 40</td>
<td>&gt; 17 times Mira</td>
</tr>
<tr>
<td>Node processors</td>
<td>AMD Opteron Nvidia Kepler</td>
<td>64-bit PowerPC A2</td>
<td>Multiple IBM Power9 CPUs &amp; multiple Nvidia Volta GPUs</td>
<td>Intel Xeon Phi processors (codenamed Knights Hill)</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>18,888 nodes</td>
<td>49,152</td>
<td>&gt;3,400 nodes</td>
<td>&gt;50,000 nodes</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Gemini</td>
<td>5D Torus</td>
<td>Dual Rail EDR-IB</td>
<td>2nd generation Intel Omni-Path Architecture</td>
</tr>
<tr>
<td>File System</td>
<td>32 PB 1 TB/s, Lustre®</td>
<td>26 PB 300 GB/s GPFS™</td>
<td>120 PB 1 TB/s GPFS™</td>
<td>150 PB &gt;1 TB/s Lustre®</td>
</tr>
</tbody>
</table>
## (Un-)Balanced Systems

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

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