Performance and Power Modeling using ASPEN

Shirley Moore[1], Marco Lopez[1], Jeremy Meredith[2], and Jeffrey Vetter[2]

[1]Computer Science Department
The University of Texas at El Paso
svmoore@utep.edu; malopez25@miners.utep.edu

[2]Future Technologies Group
Oak Ridge National Laboratory
jsmeredith@ornl.gov; vetter@ornl.gov

Introduction

The Department of Energy (DOE) has the goal of the limiting its first exascale supercomputer to 20 MegaWatts of electricity. Specifications for new petascale systems, such as the DOE CORAL systems, also include stringent power bounds. Designing and optimizing code that makes the best use of power-limited resources will require tools that support integrated power and performance modeling. The ASPEN performance modeling tool predicts application performance, based on system and application characteristics. We have recently extended ASPEN to include power consumption modeling. We use this capability to model and predict application power and energy consumption on a given architecture.

ASPEN Modeling Approach

ASPEN uses a semi-analytical modeling approach based on theoretical or measured machine and application parameters [1]. A machine model is hierarchical and models a computer system in a top-down manner, breaking the overall system into nodes connected by a network, and breaking nodes down into their constituent parts. An application model specifies the workload, memory, and communication requirements of the application. These requirements can be specified for different application phases and control constructs.

```
// Machine Model for TACC Stampede
param nnodes = 6400

include memory/ddr3_1600.aspen
include interconnects/fdrInfiniband.aspen
include sockets/intel_xeon_e5_2680.aspen

machine Stampede {
  // Properties
  node [nnodes] stampedeNode
  interconnect fdrInfiniband // as fatTree
}

node stampedeNode {
   socket [2] intel_xeon_e5_2680
}

param ddr3MemClock = 1600 * mega
param ddr3Cap = 12 * giga
param ddr3Channels = 3
param ddr3BW  = ddr3MemClock * ddr3Channels * 8
param ddr3Lat = 10 * nano / sbSIMD

memory ddr3 {
   property capacity  [ddr3Cap]
   resource loads(numBytes) [ddr3Lat + numBytes / ddr3BW]
     with stride [base*2], random[base*8]
   resource stores(numBytes) [ddr3Lat +numBytes / ddr3BW]
     with stride [base*2], random[base*8]
   conflict loads, stores
}
```

```
// Model for a Xeon E5-2680

// SB Core Parameters
param sbNumCores = 8
param sbCoreClock = 2.7 * giga // turbo up to 3.5
param sbIssueRate = sbCoreClock * 2
param sbIMD = 4

// Cache
param sbCacheCap = 20 * mega
param sbCacheLat = (1 / sbCoreClock) * 40
param sbCacheBW = sbNumCores * 96 * giga

socket intel_xeon_e5_2680 {
   core [sbNumCores] sandybridgeCore
   memory ddr3
   cache sbCache
   link qpi
}

core sandybridgeCore {
  resource flops(number) [number / sbIssueRate ]
    with dp   [base * 2],
         simd [base / sbSIMD],
         fmad [base],
         sin  [base * 18]
}

cache sbCache {
   property capacity  [sbCacheCap]
}

link qpi {
   resource intracomm(bytes) [ 0.2*micro + (bytes/(giga))]
}
```

Fig. 1. ASPEN machine model for Stampede

```
model matvec {

// Matrix size
param n = 32000

// Number of processors (n should be a multiple of sqrt(p)
param procs = 4

// How many times to repeat the mutliplication
param nTimes = 1

// Word Size (precision) in bytes
param wordSize = 8

// Data structures
data matA as Matrix(n, n, wordSize)
data vecX as Matrix(n, 1, wordSize)
data matY as Matrix(n, 1, wordSize)

// The matrix multiply kernel
kernel matvec {
  execute [1] "mainblock"
  {
    flops [(2*(n^2))/procs] as dp, fmad, simd
    loads [n^2/procs] of size [wordSize] from matA
    loads [n/sqrt(procs)] of size [wordSize] from matX
    stores [n/sqrt(procs)] of size [wordSize] to matY
    sharedmem [log(procs)+1] of size [n/sqrt(procs)]
  }
}
// Control Flow (in this case a single kernel)
kernel main {
    iterate [nTimes] { call matvec }
}
}
```

```
//
// CoMD.aspen
// Aspen model for CoMD Kernels, LJ\EAM force

model coMD {

// Double Precision
param wordSize = 8
param nTimesteps = 100

// Input Parameters
// Number of atoms
param nAtom = 32000

// Spatial Decomposition Info

param nCellsX = 20
param nCellsY = 20
param nCellsZ = 20

param cellsX = nCellsX / P_x
param cellsY = nCellsY / P_y
param cellsZ = nCellsZ / P_z

// Cutoff Distance
param cutoff = 4.5

// Box factor
param boxFactor = 1.0

// Cell size parameterized by box factor
param cellSize = boxFactor * cutoff

// Cells
param cellVolume = cellSize^3 // Could add epsilon here.

param boxX = cellSize * nCellsX // Length of sim box in x
param boxY = cellSize * nCellsY
param boxZ = cellSize * nCellsZ

// Assume average atom density (atoms per cell)
param atomsPerCell = nAtom / (nCellsX * nCellsY * nCellsZ)

// --> 27 neighbors
param avgNeighbors = 27 * atomsPerCell
// Neighbors within cutoff (15.5%)
param neighFactor = 0.155 / boxFactor^3
param avgNeighborsInCutoff = avgNeighbors * neighFactor
```

Fig. 2. Application models: a) parallel matrix-vector multiplication and b) a portion of the CoMD model

Performance Modeling Results

Parallel Matvec Runtime (in seconds)

p	n	Stampede	ASPEN
4	8000	0.005971	0.004650
4	16000	0.02318	0.01858
4	32000	0.09269	0.07425
16	8000	0.01402	0.01209
16	16000	0.05419	0.04833
16	32000	0.2062	0.1933

Stampede
- code tdmvm.c run with p MPI processes on one compute node
- communication: 1 pt2pt, log(sqrt(p)) bcasts, log(sqrt(p)) reduces, all of size n/sqrt(p)
- time measured with MPI_Wtime (computational part only)

Aspen model (using examples/basic_runtime)
- machine: stampede-1node.aspen (single node)
- socket: intel_xeon_e5_2680.aspen
- memory: ddr3_1600.aspen (using memory bandwidth measured by STREAM)
- app: matvec-par.aspen

Parallel CoMD Runtime (in seconds)

(8 processes on 1 node, 256000 atoms)

CoMD Configuration	ASPEN	Stampede
Canonical (Lennard Jones)	8.501	10.3241
AdamsEAM	13.3709	16.5013
MishinEAM	16.775	20.2981

Stampede
CoMD 1.1 release

ASPEN model
- machine: stampede-1node.aspen
- socket: intel_xeon_e5_2680
  - memory: ddr3.aspen (bw measured using STREAM)
- app: comd/CoMD-mpi-papi-mpip.aspen

Fig. 3. Measured versus predicted runtimes on Stampede

Extensions for Power Modeling

To extend ASPEN for power modeling, we need machine parameters for power and energy consumption. We obtain these by measuring runtime and energy consumption for a microbenchmark of known arithmetic intensity. We run the benchmark with different arithmetic intensities and fit the data to the equation below using a linear regression model, as in [2]. We use the kernel from the Roofline Toolkit [3] for the microbenchmark. We use the PAPI RAPL component [4] on the Catalyst cluster at LLNL, with Turbo boost turned off, to measure energy consumption. Catalyst has two Intel Xeon E5-2695 6-core sockets per node.

Machine Parameters		Application Parameters	
τ_{flop}	time per flop	W	workload
τ_{mem}	time per byte	Q	bytes moved
ε_{flop}	energy per flop	$I = W/Q$	arithmetic intensity
ε_{mem}	energy per byte		
$\pi_{flop} = \varepsilon_{flop}/\tau_{flop}$	power per flop	T	runtime
$\pi_{mem} = \varepsilon_{mem}/\tau_{mem}$	power per byte	E	energy consumption
π_1	idle power		
$\Delta\pi$	usable power		
$\beta_\tau = \tau_{mem}/\tau_{flop}$	machine time balance		
$\beta_\varepsilon = \varepsilon_{mem}/\varepsilon_{flop}$	machine energy balance		

Regression equation:

$$E = W\varepsilon_{flop} + Q\varepsilon_{mem} + \pi_1 T$$

After fitting the data to the regression model, we derive the remaining power and energy related machine parameters. We use these parameters to model power consumption versus arithmetic intensity, as in [2]. We have extended ASPEN with a powerline tool and to predict energy consumption for a given application on a given architecture.

$$\overline{P} = \overline{P}(I) = \pi_1 + \begin{cases} \pi_{flop} + \pi_{mem}\dfrac{\beta_\tau}{I} \; if \; I \geq \beta_\tau^+ \\ \pi_{flop}\dfrac{I}{\beta_\tau} + \pi_{mem} \; if \; I \leq \beta_\tau^- \\ \Delta\pi \quad otherwise \end{cases}$$

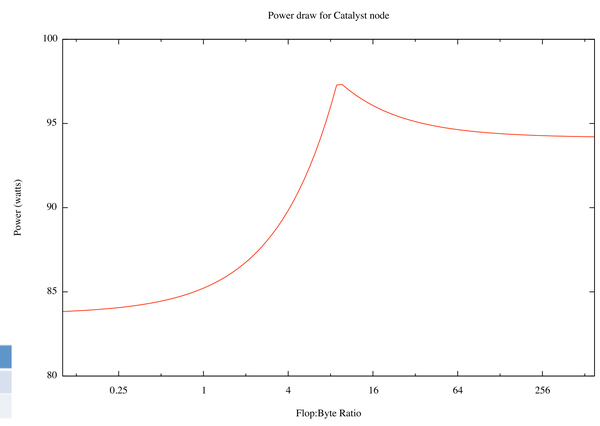
where

$$\beta_\tau^+ = \beta_\tau \max\left(1, \frac{\pi_{mem}}{\Delta\pi - \pi_{flop}}\right)$$

$$\beta_\tau^- = \beta_\tau \min\left(1, \frac{\Delta\pi - \pi_{mem}}{\pi_{flop}}\right)$$

Energy Consumption Modeling of Parallel Matvec on Catalyst

n	RAPL (measured)	ASPEN (predicted)
7200	.76475 Joules	.73104 Joules
36000	20.0716 Joules	19.8853 Joules



References

1. Seyong Lee, Jeremy S. Meredith, Jeffrey S. Vetter. COMPASS: A Framework for Automated Performance Modeling and Prediction. ICS 2015: 405-414
2. Jee Choi, Marat Dukhan, Xing Liu, Richard W. Vuduc. Algorithmic Time, Energy, and Power on Candidate HPC Compute Building Blocks. IPDPS 2014: 447-457
3. Yu Jung Lo, Samuel Williams, Brian Van Straalen, Terry J. Ligocki, Matthew J. Cordery, Leonid Oliker, Mary W. Hall. Roofline Model Toolkit: A Practical Tool for Architectural and Program Analysis. PMBS 2014
4. Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Daniel Terpstra, Shirley Moore: Measuring Energy and Power with PAPI. ICPP Workshops 2012: 262-268

This work is partially supported by the Air Force Office of Scientific Research under Award No. FA9550-12-1- 0476 and by the Department of Energy under Award Nos. DE-SC00006722 and DE-AC05-00OR22725. We would like to thank Texas Advanced Computing Center for use of Stampede and Lawrence Livermore National Laboratory for use of Catalyst.