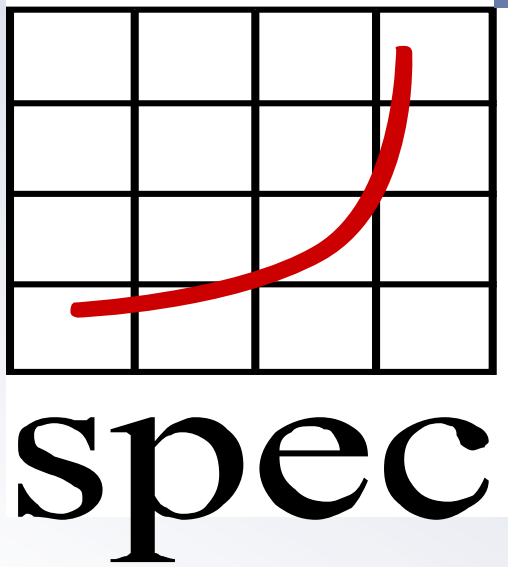
Advancing SPEC benchmarks

Oscar Hernandez, Wayne Joubert, ORNL

Guido Juckeland, Technische Universität Dresden



Contents

- Using power measurements
- Porting to new platforms (SPEC ACCEL OpenMP 4.0 Target Directives)

Contents

- Using power measurements
- Porting to new platforms (SPEC ACCEL OpenMP 4.0 Target Directives)

SPEC Power

- SPEC provides a standard methodology to measure and report power usage which can be incorporated into a SPEC benchmark.
- Normalizes the power usage across the full run of the suite


```
Time,03-10-2014 14:14:25.304,Temperature,25.187500,Humidity,29.000000
Time,03-10-2014 14:14:30.303,Temperature,25.187500,Humidity,29.000000
03-10-2014 14:14:52.251: Go with mark '304.olbm'
03-10-2014 14:14:52.251: Response to client sent: Starting untimed measure
ss at 5000ms with 0 rampup samples
03-10-2014 14:14:54.948,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:14:59.948,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:04.949,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:09.948,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:14.949,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:19.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:24.949,Temp,25.125000,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:29.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:34.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:39.949,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:44.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:49.949,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:54.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
03-10-2014 14:15:59.948,Temp,25.187500,Humidity,29.000000,Mark,304.olbm
```



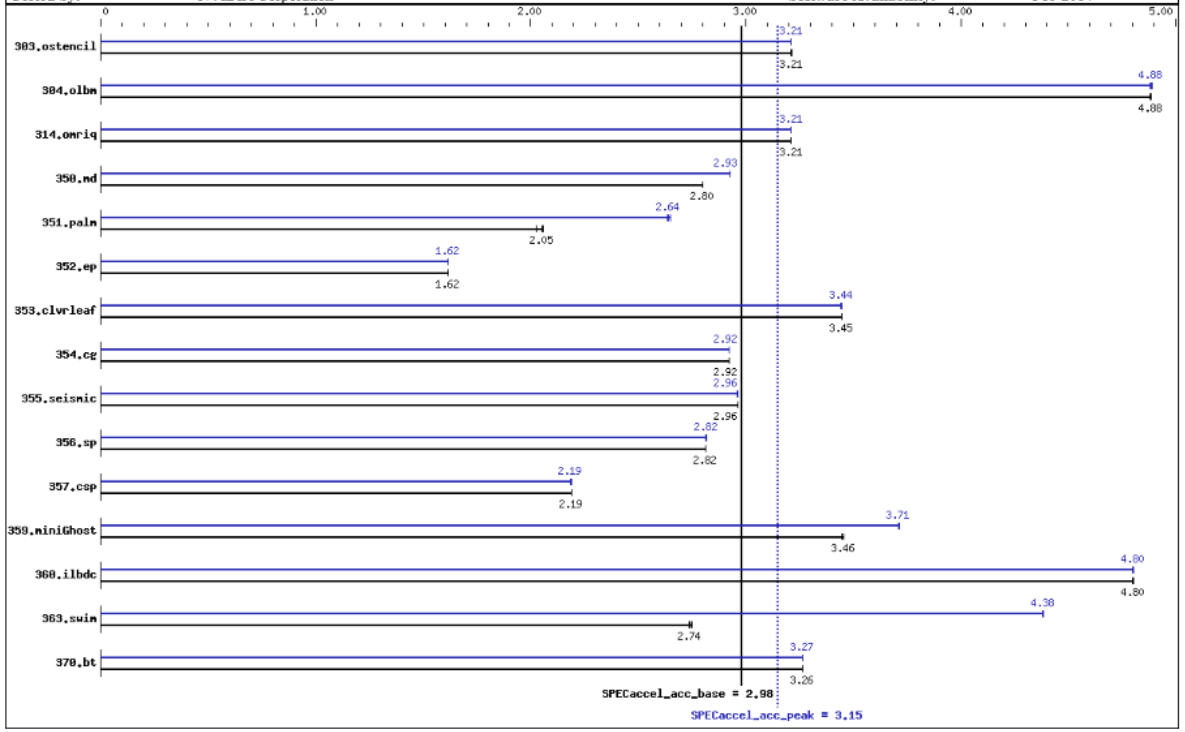
```
Time,03-10-2014 14:14:20.458,Watts,308.159000,Volts,120.442000,Amps,2.633368,PF,0.971632
Time,03-10-2014 14:14:21.457,Watts,308.156000,Volts,120.513000,Amps,2.632228,PF,0.971568
Time,03-10-2014 14:14:22.458,Watts,308.445000,Volts,120.528000,Amps,2.634028,PF,0.971622
Time,03-10-2014 14:14:23.458,Watts,308.457000,Volts,120.538000,Amps,2.634318,PF,0.971632
Time,03-10-2014 14:14:24.457,Watts,308.448000,Volts,120.521000,Amps,2.634158,PF,0.971614
03-10-2014 14:14:29.833: Response to client sent: Time,03-10-2014 14:14:25.458,Watts,308.751000,Volts,120.
531000,Amps,2.636380,PF,0.971645
Time,03-10-2014 14:14:26.458,Watts,308.834000,Volts,120.518000,Amps,2.637218,PF,0.971678
Time,03-10-2014 14:14:27.457,Watts,308.958000,Volts,120.524000,Amps,2.638168,PF,0.971677
Time,03-10-2014 14:14:28.458,Watts,308.862000,Volts,120.528000,Amps,2.638168,PF,0.971690
03-10-2014 14:14:52.250: Go with mark '304.olbm'
03-10-2014 14:14:52.250: Response to client sent: Starting untimed measurement, maximum 50000 samples at
1000ms with 0 rampup samples
03-10-2014 14:14:52.747,Watts,163.814000,Volts,120.827000,Amps,1.417630,PF,0.961701,Mark,304.olbm
03-10-2014 14:14:53.746,Watts,153.447000,Volts,120.836000,Amps,1.336560,PF,0.949928,Mark,304.olbm
03-10-2014 14:14:54.747,Watts,166.403000,Volts,120.850000,Amps,1.457490,PF,0.944537,Mark,304.olbm
03-10-2014 14:14:55.747,Watts,199.812000,Volts,120.779000,Amps,1.736290,PF,0.956115,Mark,304.olbm
03-10-2014 14:14:56.746,Watts,253.367000,Volts,120.582000,Amps,2.188830,PF,0.959402,Mark,304.olbm
03-10-2014 14:14:57.747,Watts,285.315000,Volts,120.565000,Amps,2.476210,PF,0.966108,Mark,304.olbm
03-10-2014 14:14:58.747,Watts,285.822000,Volts,120.550000,Amps,2.480240,PF,0.966163,Mark,304.olbm
03-10-2014 14:14:59.746,Watts,261.949000,Volts,120.697000,Amps,2.253870,PF,0.963508,Mark,304.olbm
03-10-2014 14:15:00.747,Watts,265.465000,Volts,120.558000,Amps,2.297140,PF,0.964333,Mark,304.olbm
03-10-2014 14:15:01.747,Watts,261.673000,Volts,120.550000,Amps,2.252590,PF,0.963824,Mark,304.olbm
03-10-2014 14:15:02.746,Watts,265.597000,Volts,120.577000,Amps,2.296130,PF,0.964259,Mark,304.olbm
```

SPEC® ACCEL_ACC Result

ASUS (Test Sponsor: NVIDIA Corporation)
NVIDIA Tesla K40c
ASUS P9X79 Motherboard

SPECaccel_acc_base =	2.98
SPECaccel_acc_energy_base =	3.49
SPECaccel_acc_peak =	3.15
SPECaccel_acc_energy_peak =	3.63

ACCEL license:	019	**Test date:**	Feb-2014
Test sponsor:	NVIDIA Corporation	**Hardware Availability:**	Nov-2013
Tested by:	NVIDIA Corporation	**Software Availability:**	Feb-2014



Hardware

CPU Name:	Intel Core i7-3930K
CPU Characteristics:	
CPU MHz:	3200
CPU MHz Maximum:	3800
FPU:	Integrated
CPU(s) enabled:	6 cores, 1 chip, 6 cores/chip, 2 threads/core
CPU(s) orderable:	1 chip
Primary Cache:	32 KB I + 32 KB D on chip per core
Secondary Cache:	256 KB I+D on chip per core
L3 Cache:	12 MB I+D on chip per chip
Other Cache:	None
Memory:	8 GB (2 x 4 GB 2Rx4 PC3-14900R-9, running at 1600 MHz)
Disk Subsystem:	1000 GB Seagate ST1000DM003 7200 RPM SATA
Other Hardware:	None

Accelerator

Accel Model Name:	Tesla K40c
Accel Vendor:	NVIDIA
Accel Name:	NVIDIA Tesla K40c
Type of Accel:	GPU
Accel Connection:	PCIe 3.0 16x
Does Accel Use ECC:	No
Accel Description:	GPU Boost set to maximum graphic clock frequency of 875 MHz. See notes below.
Accel Driver:	NVIDIA UNIX x86_64 Kernel Module 319.60

Software

Operating System:	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.el6.x86_64
Compiler:	PGI Accelerator Server Complete, Release 14.2
File System:	ext4
System State:	Run level 3 (multi-user)
Other Software:	FFTW 3.3.3

Power Analyzer

Power Analyzer:	Power Analyzer
Hardware Vendor:	Xitron Technologies, Inc.
Model:	2801
Serial Number:	28011109005
Input Connection:	RS232 via USB-adapter
Metrology Institute:	NIST
Calibration By:	Micro Precision Calibration, Inc.
Calibration Label:	220081222038459
Calibration Date:	02.20.2014
PTDaemon Version:	1.6.2 (372e138a; 2013-12-04)
Setup Description:	connected to the single power supply that powers the system
Current Ranges Used:	2.0A
Voltage Range Used:	135V

Temperature Meter

Temperature Meter:	Temperature Meter
Hardware Vendor:	Digi
Model:	DigiWATCHPORT_H
Serial Number:	WS34682143
Input Connection:	USB
PTDaemon Version:	1.6.2 (372e138a; 2013-12-04)
Setup Description:	Position 5mm above intake fan

Base Results Table

Benchmark	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio
303.ostencil	**45.2**	**3.21**	**16.0**	**366**	**354**	**3.31**	45.2	3.21	16.2	366	359	3.26	45.1	3.21	16.4	366	364	3.22
304.olbm	**93.2**	**4.88**	**27.1**	**301**	**291**	**5.75**	93.1	4.89	27.3	308	293	5.72	93.2	4.88	27.1	301	291	5.76
314.omriq	298	3.21	108	412	362	3.54	298	3.21	108	389	362	3.54	**298**	**3.21**	**108**	**388**	**362**	**3.55**
350.md	90.0	2.80	31.2	352	347	3.01	90.1	2.80	31.1	352	346	3.02	**90.0**	**2.80**	**31.2**	**353**	**347**	**3.01**
351.palm	183	2.03	41.0	235	224	2.61	180	2.06	40.4	242	225	2.64	**180**	**2.05**	**40.3**	**235**	**224**	**2.65**
352.ep	328	1.62	83.4	274	254	1.97	**328**	**1.62**	**83.1**	**256**	**253**	**1.98**	328	1.62	82.9	255	253	1.98
353.clvrleaf	**129**	**3.45**	**41.5**	**326**	**321**	**3.80**	129	3.45	41.3	326	320	3.82	129	3.45	41.3	326	320	3.82
354.cg	**140**	**2.92**	**39.4**	**320**	**283**	**3.55**	140	2.92	39.4	328	282	3.55	140	2.92	39.3	319	282	3.56
355.seismic	125	2.96	35.6	296	285	3.62	125	2.96	35.5	293	284	3.64	**125**	**2.96**	**35.5**	**293**	**284**	**3.64**
356.sp	**98.0**	**2.82**	**27.8**	**288**	**284**	**3.34**	98.0	2.82	27.8	288	284	3.34	98.0	2.82	27.8	288	283	3.34
357.csp	123	2.19	33.4	281	270	2.64	123	2.19	33.3	273	269	2.65	**123**	**2.19**	**33.3**	**285**	**270**	**2.64**
359.miniGhost	107	3.45	30.2	307	282	3.94	**107**	**3.46**	**30.1**	**307**	**282**	**3.95**	107	3.46	30.0	307	281	3.95
360.ilbdc	76.4	4.80	23.8	323	312	5.45	76.5	4.80	23.8	323	312	5.45	**76.4**	**4.80**	**23.8**	**322**	**312**	**5.45**
363.swim	83.7	2.75	21.9	263	262	3.46	**84.0**	**2.74**	**22.0**	**270**	**262**	**3.46**	84.1	2.74	21.9	263	261	3.46
370.bt	68.3	3.26	18.8	277	276	4.02	**68.3**	**3.26**	**18.8**	**277**	**275**	**4.02**	68.4	3.26	18.8	276	275	4.02

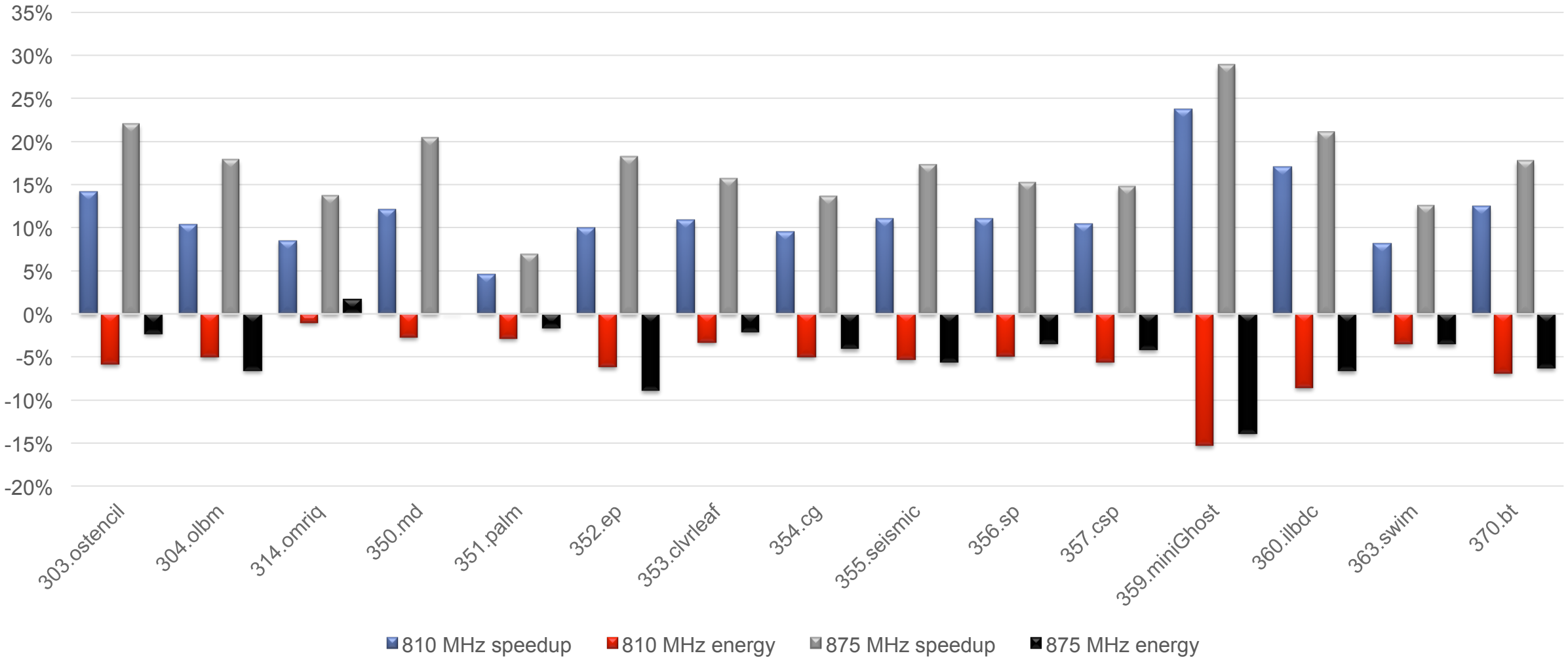
Results appear in the order in which they were run. Bold underlined text indicates a median measurement.

Peak Results Table

Benchmark	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio	Seconds	Ratio	Energy (kJ)	Maximum Power	Average Power	Energy Ratio
303.ostencil	**45.2**	**3.21**	**16.3**	**366**	**360**	**3.25**	45.2	3.21	16.4	379	364	3.22	45.2	3.21	16.2	366	359	3.26
304.olbm	93.2	4.88	27.1	301	291	5.76	93.1	4.89	27.2	300	292	5.75	**93.2**	**4.88**	**27.2**	**300**	**292**	**5.74**
314.omriq	**298**	**3.21**	**108**	**405**	**362**	**3.55**	298	3.21	108	389	362	3.55	298	3.21	108	389	361	3.55
350.md	**86.1**	**2.93**	**30.6**	**361**	**355**	**3.07**	86.1	2.93	30.7	361	356	3.06	86.1	2.93	30.8	361	357	3.06
351.palm	140	2.64	33.1	246	236	3.23	**140**	**2.64**	**32.6**	**251**	**233**	**3.28**	140	2.65	32.5	245	233	3.29
352.ep	**328**	**1.62**	**83.1**	**272**	**253**	**1.98**	328	1.62	82.9	255	253	1.98	328	1.62	83.1	255	253	1.98
353.clvrleaf	129	3.45	41.2	326	319	3.82	129	3.44	41.3	326	319	3.82	**129**	**3.44**	**41.3**	**326**	**320**	**3.81**
354.cg	140	2.92	39.3	319	281	3.56	**140**	**2.92**	**39.6**	**323**	**283**	**3.54**	140	2.92	39.4	319	283	3.55
355.seismic	125	2.96	35.5	301	284	3.63	**125**	**2.96**	**35.5**	**293**	**284**	**3.64**	125	2.96	35.5	295	284	3.64
356.sp	98.0	2.82	27.9	288	284	3.33	**98.0**	**2.82**	**27.8**	**288**	**283**	**3.34**	98.0	2.82	27.8	289	283	3.34
357.csp	**123**	**2.19**	**33.2**	**273**	**269**	**2.65**	124	2.19	33.2	273	268	2.65	123	2.19	33.2	273	269	2.66
359.miniGhost	**99.4**	**3.71**	**28.9**	**343**	**291**	**4.11**	99.4	3.71	28.9	341	291	4.11	99.4	3.71	29.0	339	292	4.10
360.ilbdc	76.5	4.80	23.9	321	313	5.43	**76.4**	**4.80**	**23.8**	**330**	**312**	**5.45**	76.4	4.80	23.8	321	312	5.46
363.swim	**52.5**	**4.38**	**15.7**	**299**	**298**	**4.85**	52.5	4.38	15.6	299	298	4.86	52.5	4.38	15.7	299	298	4.85
370.bt	68.3	3.27	18.7	277	274	4.04	68.3	3.26	18.8	277	274	4.03	**68.3**	**3.27**	**18.8**	**277**	**275**	**4.03**

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.

Impact of Clock Boost (result #13, #14, and #15, K40c, base, ECC on)



Contents

- Using power measurements
- Porting to new platforms (SPEC ACCEL OpenMP 4.0 Target Directives)

SPEC ACCEL

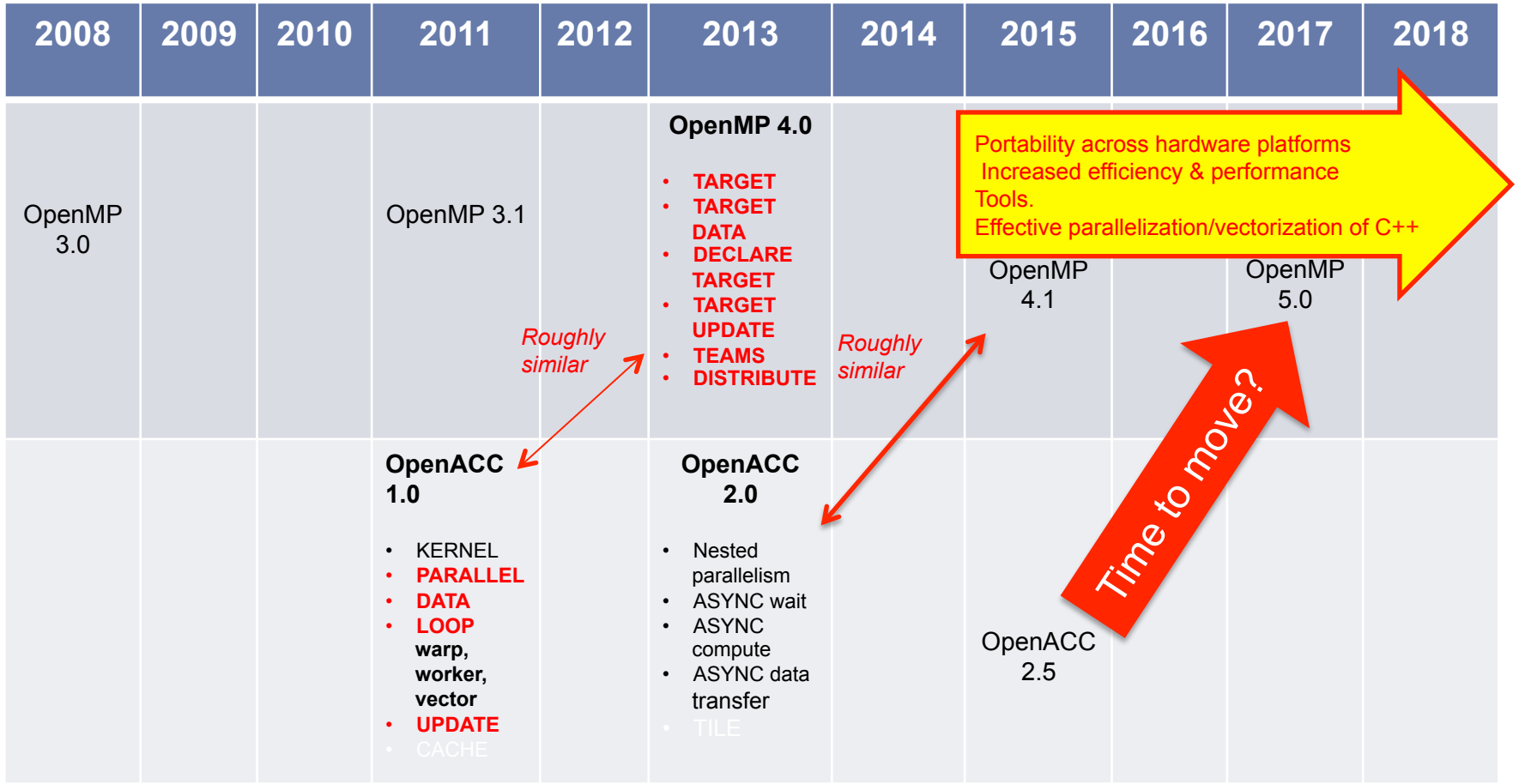
- Currently SPEC ACCEL supports OpenACC
- We are working on porting SPEC to other programming models such as:
  - OpenMP 4.1
- SPEC ACCEL can be used for research to explore new programming models:
  - Extensions for OpenACC and OpenMP 4.1
    - Flexible directives to improve portability
  - Exascale runtimes: OCR, ParalleX, Parsec with OpenACC/OpenMP
  - Hybrid programming: MPI + OpenMP 4.0, etc

OpenACC @ SPEC

- OpenACC has been a key strategy for portable programming of accelerators since it's the first working directive based solution
- Several current implementations of OpenACC (PGI, Cray)
- Growing breadth of support for OpenACC
  - GCC, Pathscale, several research compilers
- Plan to port benchmarks to OpenMP 4.1 will help our users
  - This transition is likely to occur in the next year or so.
  - There are some technical challenges that OpenMP 4.1 implementers are solving
    - E.g. Simulating SIMD on GPUs

OpenMP and OpenACC progress

- OpenACC innovation continues, OpenMP adopts relevant features



Programming models must evolve before they can stabilize

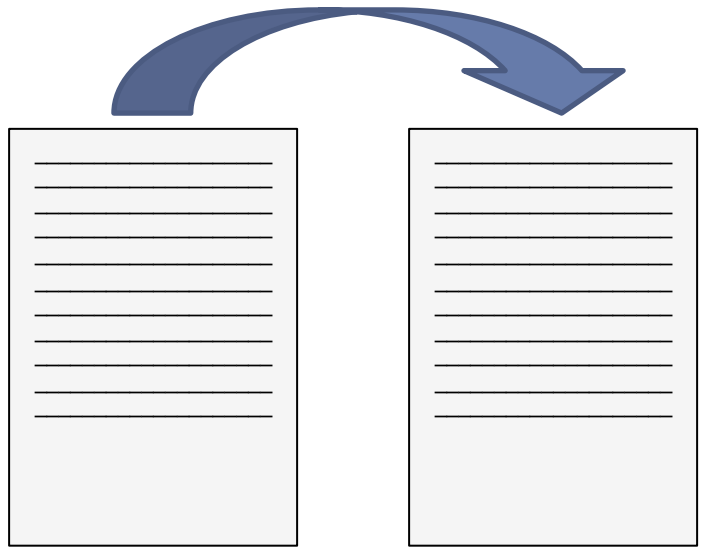
Porting OpenACC to OpenMP 4

- Compare OpenMP 4.1 accelerator extension with OpenACC
  - Understand mapping
  - Understand impact of newer OpenACC features
- OpenACC is evolving with new features which may impact OpenMP 4.1 or 5.
- OpenACC interoperability with OpenMP is important for the transition
- Current application investments in OpenACC porting are preserved when porting to OpenMP

OpenACC 2.0	OpenMP 4.0
parallel	target
parallel/gang/workers/vector	target teams/parallel/simd
data	target data
parallel loop	teams/distribute/parallel for
update	target update
cache	
wait	OpenMP 4.1 draft
declare	declare target
data enter/exit	OpenMP 4.1 draft
routine	declare target
async wait	OpenMP 4.1 draft
device type	OpenMP 4.1 draft
tile	
host data	OpenMP 4.1 draft

Converting OpenACC to OpenMP 4

- Many constructs currently translate directly from OpenACC to OpenMP
- Some constructs are present in one but not the other
- Also, at some points there are subtle differences, e.g., OpenACC allows the compiler more discretion regarding how loops are mapped to hierarchical parallelism



Translating OpenACC to OpenMP 4: Procedure (1)

1. The user must modify any OpenACC constructs for which no equivalent counterpart exists in OpenMP:
   - Explicit data regions
   - **`kernels`** directive
   - **`device_type`** clause
   - **`host_data`** and **`link`** clauses
   - **`cache`** directive
   - Complex use of asynchronous streams

Translating OpenACC to OpenMP 4:Procedure (2)

2. Translate data regions:
   - **`acc data` → `omp target data`**
   - **`create(...)` → `map(alloc:...)`**
   - **`pcopy(...)` → `map(tofrom:...)`**
   - **`pcopyin(...)` → `map(to:...)`**
   - **`pcopyout(...)` → `map(from:...)`**
   - use of **`async(…)`** can be replaced by use of OpenMP CPU threads or tasks to handle transfers
   - if multiple devices used, replace calls to **`acc_set_device_num(…)`** with **`device(…)`** clause

Translating OpenACC to OpenMP 4: Procedure (3)

3. Translate data update operations:
   - `acc update` → `omp target update`
   - `host(...)` → `from(...)`
   - `self(...)` → `from(:...)`
   - `device(...)` → `to(...)`
   - use of `async(…)` can be replaced by use of OpenMP CPU threads or tasks to handle transfers

Translating OpenACC to OpenMP 4: Procedure (4)

4. Translate accelerator parallel regions: generally,
   - `acc parallel` → `omp target teams`
   - `acc loop gang` → `omp distribute`
   - `acc loop worker` → `omp parallel for [simd]`
   - `acc loop vector` → `omp simd`
   - `acc loop independent -> teams|parallel for|simd`
   - `gang(...)` → `dist_schedule(...)`
   - `num_gangs(...)` → `num_teams(...)`
   - `num_workers(...)` → `thread_limit(...)`
   - `vector_length(...)` → `safelen(...)`
   - use of `async(…)` can be replaced by use of OpenMP CPU threads or tasks to handle device execution

Translating OpenACC to OpenMP 4: Procedure (5)

5. Adjust function attribute specifiers:
   - **`acc routine` → `omp declare target / end declare target`**
   - OpenACC **`gang`**, **`worker`**, **`vector`**, **`seq`** clauses have no exact counterpart in OpenMP

Example: 303.ostenci

OpenACC

```
void cpu_stencil(float c0,float c1, float *A0,float * Anext,const int nx, const int ny, const int nz)
{

  int i, j, k;
#pragma acc kernels  pcopyin(A0[0:nx*ny*nz]), pcopyout(Anext[0:nx*ny*nz])
  {
#pragma acc loop independent vector
      for(i=1;i<nx-1;i++)
      {
#pragma acc loop independent gang vector
          for(j=1;j<ny-1;j++)
          {
#pragma acc loop independent gang vector
              for(k=1;k<nz-1;k++)
              {
                  Anext[Index3D (nx, ny, i, j, k)] =
                  (A0[Index3D (nx, ny, i, j, k + 1)] +
                  A0[Index3D (nx, ny, i, j, k - 1)] +
                  A0[Index3D (nx, ny, i, j + 1, k)] +
                  A0[Index3D (nx, ny, i, j - 1, k)] +
                  A0[Index3D (nx, ny, i + 1, j, k)] +
                  A0[Index3D (nx, ny, i - 1, j, k)])*c1
                  - A0[Index3D (nx, ny, i, j, k)]*c0;
              }
          }
      }
  }
}
```

OpenMP 4.0

```
void cpu_stencil(float c0,float c1, float *A0,float *Anext,const int nx, const int ny, const int
nz)
{
  int i, j, k;
  int size=nx*ny*nz;
#pragma omp target map(alloc:A0[0:size], Anext[0:size])
#pragma omp teams distribute parallel for collapse(2)
      for(k=1;k<nz-1;k++)
      {
          for(j=1;j<ny-1;j++)
          {
#pragma omp simd
              for(i=1;i<nx-1;i++)
              {
                  Anext[Index3D (nx, ny, i, j, k)] =
                  (A0[Index3D (nx, ny, i, j, k + 1)] +
                  A0[Index3D (nx, ny, i, j, k - 1)] +
                  A0[Index3D (nx, ny, i, j + 1, k)] +
                  A0[Index3D (nx, ny, i, j - 1, k)] +
                  A0[Index3D (nx, ny, i + 1, j, k)] +
                  A0[Index3D (nx, ny, i - 1, j, k)])*c1
                  - A0[Index3D (nx, ny, i, j, k)]*c0;
              }
          }
  }
}
```

Summary

- Power measurements are a useful addition for energy-to-solution comparisons

- OpenMP 4 and OpenACC support for accelerator devices is similar in many respects
- Porting SPEC ACCEL to OpenMP 4 is straight forward
  - Challenges: maturity of compilers
- Application developers should be aware of similarities and differences in order to develop future-proofed code that can run well under either API

ACK

- Guido Juckeland, guido.Juckeland@tu-dresden.de
- Wayne Joubert, joubert@ornl.gov
- Oscar Hernandez, oscar@ornl.gov

We acknowledge the assistance of Wei Ding, Markus Eisenbach, Christos Kartsaklis, David E. Bernholdt and Daniel Tian.

This research used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

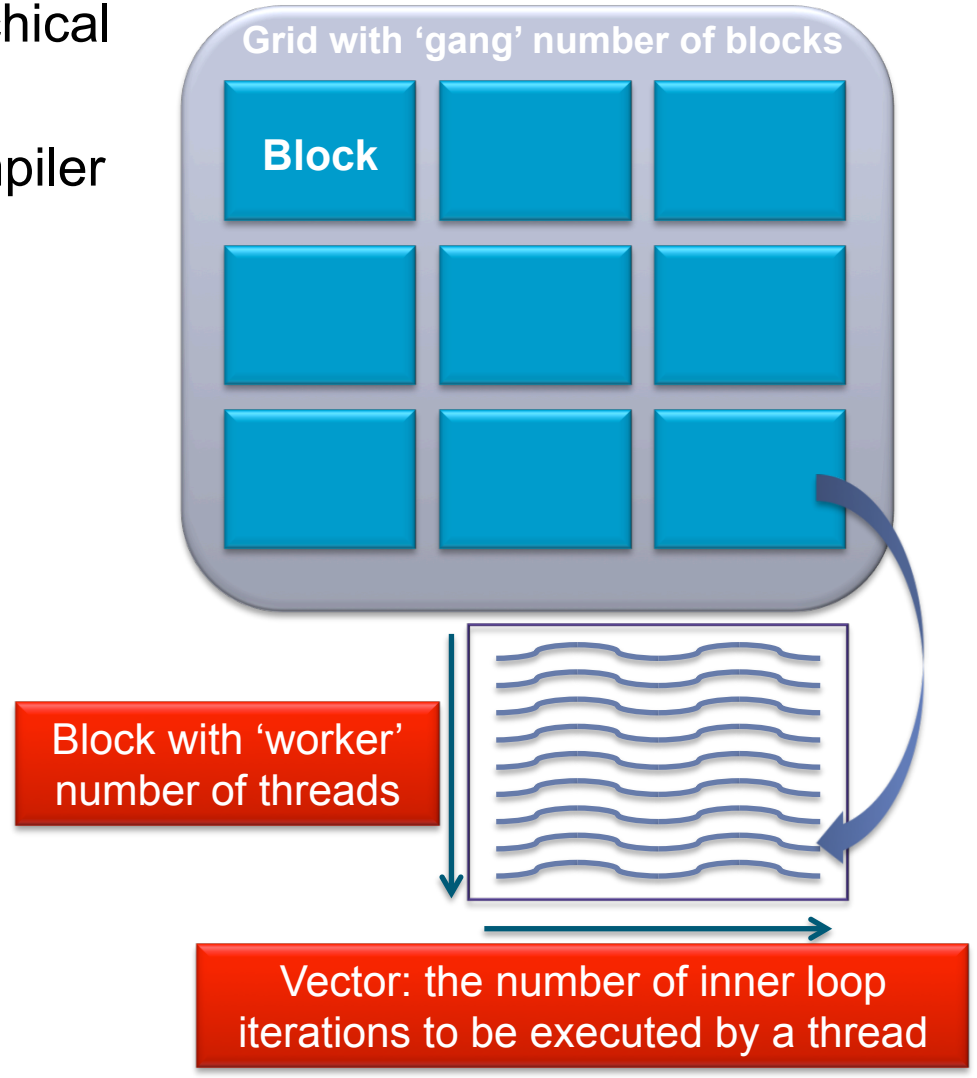
This material is in part based upon work supported by the National Science Foundation under Grant Number 1137097 and by the University of Tennessee through the Beacon Project. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the University of Tennessee.

Supplementary slides

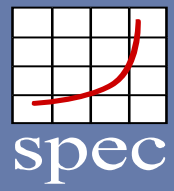
OpenACC Example: Gang, Workers and Vector in a Grid

- Directives used to map parallel loops to hierarchical parallelism on accelerator
- Execution configuration determined by the compiler or configured manually

```
#pragma acc kernels
{
#pragma acc loop independent
  for (int i = 0; i < n; ++i){
    for (int j = 0; j < n; ++j){
      for (int k = 0; k < n; ++k){
        B[i][j*k%n] = A[i][j*k%n];
      }
    }
  }
#pragma acc loop gang(NB) worker(NT)
  for (int i = 0; i < n; ++i){
    #pragma acc loop vector(NI)
    for (int j = 0; j < m; ++j){
      B[i][j] = i * j * A[i][j];
    }
  }
}
```



Challenges with directive-based programming models



- How to specify the in-node parallelism in the application
  - Loop based parallelism is not enough for future systems
- How to efficiently map the parallelism of the application to the hardware
  - How to schedule work to multiple accelerators within the node?
  - How to schedule work to within accelerators while being portable
- How to transfer data across different types of memory
  - Problem may go away but is important for data locality
- How to specify different memory hierarchies in the programming model
  - Shared memory within GPU, etc
- Tools APIs