When Is Energy ≠ Time?: Understanding Energy Scaling with eAudit

Sudhakar Yalamanchili and Eric Anger

School of Electrical and Computer Engineering
Georgia Institute of Technology

Scaling Performance and Power

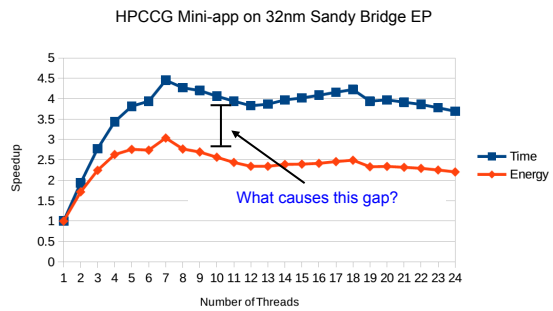
$$Perf\left(\frac{ops}{s}\right) = Power(W) \times Efficiency\left(\frac{ops}{joule}\right)$$

W. J. Dally, Keynote IITC 2012

- Increasing performance requires increasing system scale → parallelism
- Scaling parallelism does not effect energy and time in the same way!

	Current	Exascale	Growth
Cores	3.1M	1B	~300x
Power	17.8MW	20-40MW	~1.5-2.5x

Energy Scaling vs. Time Scaling



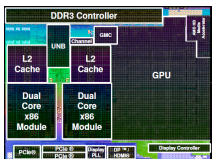
$$EnergySpeedup = \frac{E_1}{E_p}$$

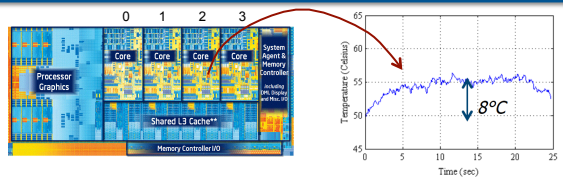
$$TimeSpeedup = \frac{T_1}{T_p}$$

- Developers should understand how both energy and time scale with parallelism
- Relationship between energy scaling and time scaling?
  - When are they not the same? Tradeoffs?
  - Drive the development of diagnostic tools

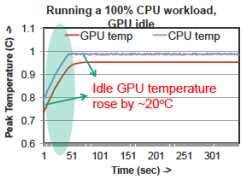
Why Is This an Application/Algorithm Problem?

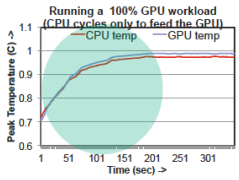
Managing Thermal Capacity: Thermal Coupling





Temperature on Core 2 when Core 3 is busy and remaining cores are idle

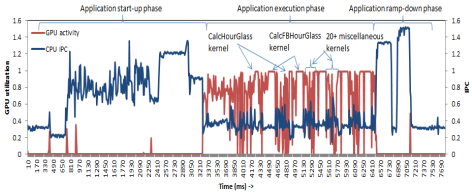




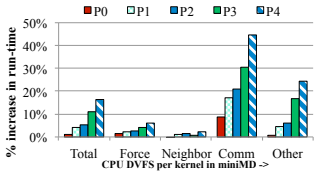
- Significant rise in temperature of the idle component due to thermal coupling and pollution
- CPU cores consume thermal headroom more rapidly (4X faster)
  - GPUs sustain power boosts longer!
- Better management for 10%-40% gains in measured energy efficiency are possible
- Power management ≠ thermal management

I. Paul, S. Manne, M. Arora, W.L. Bircher, S. Yalamanchili, "Cooperative boosting: needy versus greedy power management", *ISCA 2013.*

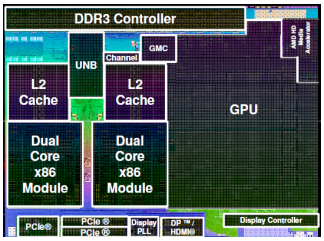
Managing Power Capacity



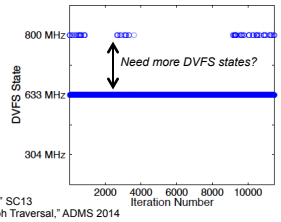
Dynamic Demand

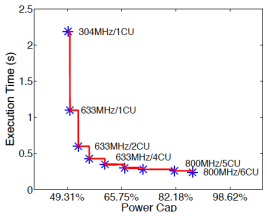


Power Sensitivity → Energy Efficiency[1]



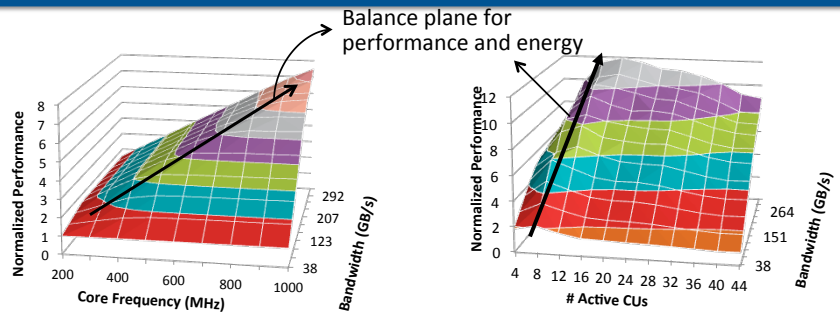
Design Space[2] (BFS)

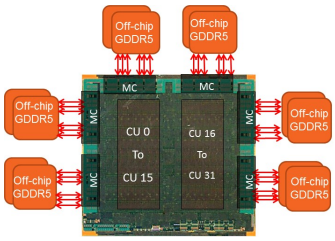




[1] I. Paul, et.al, "Coordinated Energy Management in Heterogeneous Processors" SC13
[2] A. McLaughlin et.al, "A Power Characterization and Management of GPU Graph Traversal," ADMS 2014

Shift in the Balance Point





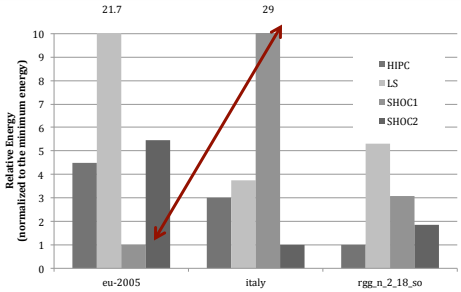
Hardware Balance

Relative energy costs of compute and memory access ⟷ *Relative ops/byte demand of application*

Up to 36% power savings with a maximum performance loss of 3.6%

I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing Compute and Memory Power in High Performance GPUs," *IEEE/ACM International Symposium on Computer Architecture (ISCA)*, June 2015.

Understanding Application-Level Energy Scaling



Courtesy H. Kim

Different implementations of BFS on different input data sets

- Note the variance of energy dissipation across different implementations of the same function
- Challenge: How do we understand the energy implications of our decisions? Algorithms, data structures, etc.

When Is Energy ≠ Time?

You can hide latency but not energy!

Energy Components

- Static Energy
  - Scales with execution time
  - E.g., leakage energy
    - Technology dependent
  - Idle energy consumption (OS, I/O, etc.)

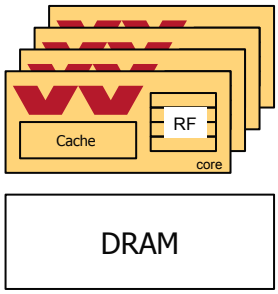
  } Energy overhead

- Dynamic Energy
  - Scales with work
  - Independent of time (strong scaling)
  - Energy required to solve problem

  } • Constant under ideal strong scaling
  • Grows under weak scaling

Energy scaling is limited by the fraction corresponding to static energy

Amdahl's View of Energy Scaling



$$S_t = \frac{1}{\frac{1-f}{p} + f}$$ → serial fraction

$$S_e = \frac{1}{(1-s) + \frac{s}{S_t}}$$ → static fraction

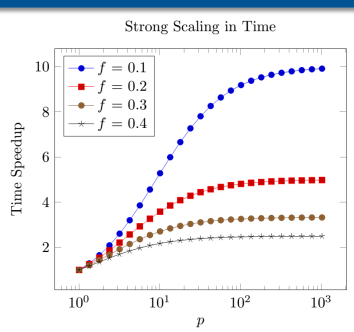
Base case is a single core executing a serial algorithm

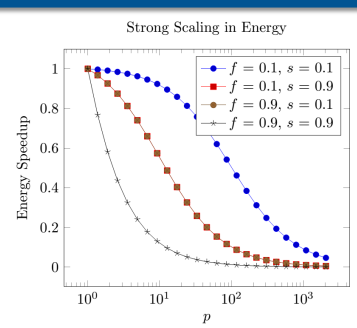
$$\eta_t = \frac{S_t}{p}$$ ↓ Time efficiency

$$s = \frac{E_s}{E_s + E_d}$$ → static energy, → dynamic energy

- Ideally energy speedup tracks time speedup (no idle energy)

Impact of Concurrency

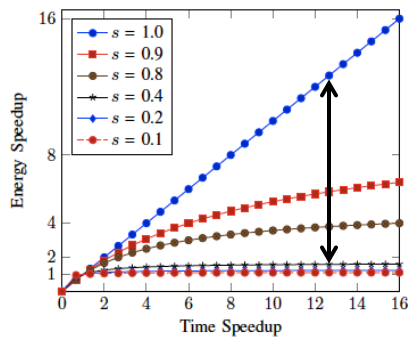


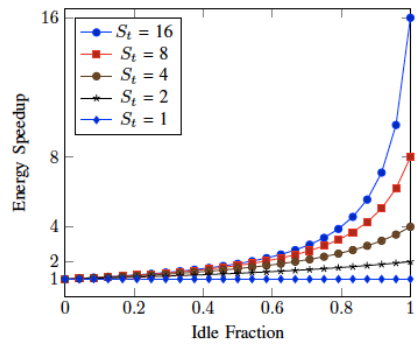


$$S_e = \frac{1}{(1 - f \times s) + p \times f \times s}$$

- In time, strong scaling is limited by the serial fraction
  - When it is small, large benefit from strong scaling
- In energy, strong scaling is limited by the static fraction
  - Static fraction is multiplicative penalty in addition of the serial fraction

Energy Scaling vs. Time Scaling





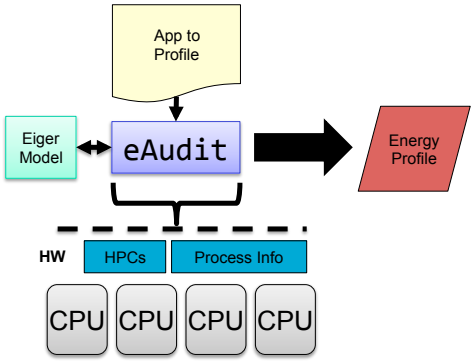
- Impact of time on energy scaling is a function of the static fraction

$$Time\_speedup \geq Energy\_speedup \geq 1.0$$

Runtime vs. Design Time?

Energy Auditing: eAudit

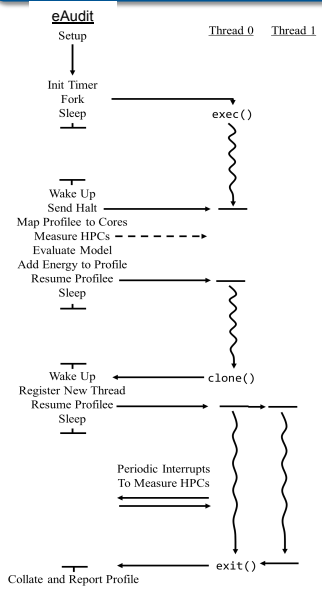
- Application energy auditing tool
  - *Function-level* attribution
- Diagnose application energy consumption behavior
- Provide actionable information to steer energy optimization

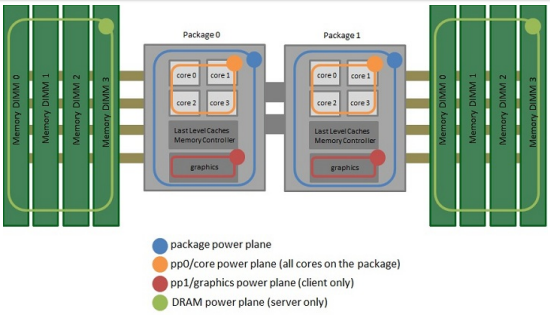


Example output

Name	Energy	Time	Instructions	% Energy	% Time
generate_matrix(int, int, int, HPC_Sparse_Matrix_STRUCT**, double**, double**, double**) at ??	4.3672	0.115	372291000	54.97805	12.3656
HPC_sparsemv(HPC_Sparse_Matrix_STRUCT*, double const*, double*) at HPC_sparsemv.cpp	2.3132	0.511	1.214E+09	29.12098	54.9462
?? at ??	1.0075	0.238	483396000	12.68337	25.5914
waxpby(int, double, double const*, double, double const*, double*) at waxpby.cpp	0.1348	0.037	67981800	1.69754	3.97849
ddot(int, double const*, double const*, double*, double&) at ddot.cpp	0.1207	0.029	62164100	1.520063	3.11828

eAudit Implementation

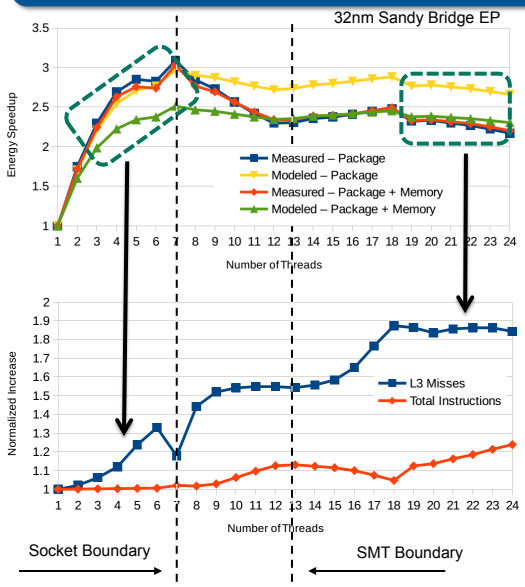




Martin Dimitrov. https://software.intel.com/en-us/articles/intel-power-governor

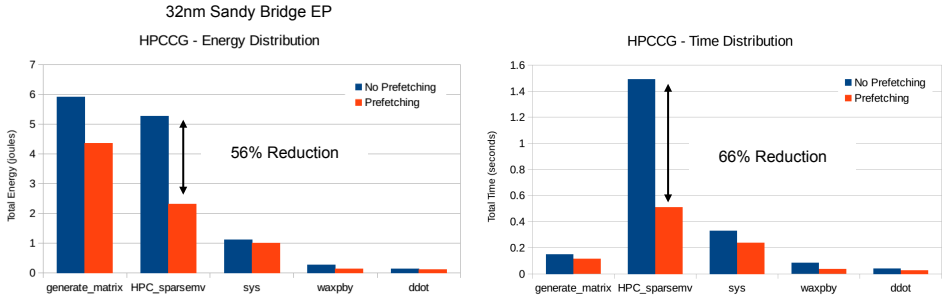
- Sampling-based measurements, similar to `gprof`
- RAPL limited to all cores on package: future versions should expose per-core counters

Measurements: HPCCG



- Model does **not** include variations to energy due to growth in work
  - Sharing, locking, barriers, etc.
  - Typically a function of # threads

eAudit In Action: Effect of Prefetching

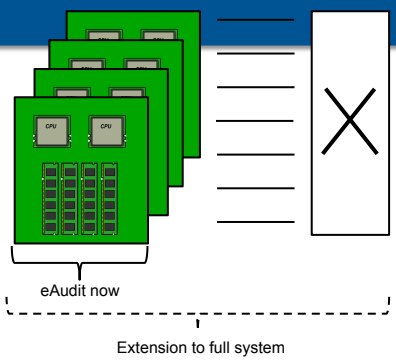


- Prefetching decreases time and energy, but not my same degree
  - Reduction in time → reduction in static energy
  - Speculative memory traffic → increase in dynamic energy

Need Better Energy Instrumentation!

Scaling Across Sockets

- eAudit demonstrated at board-level
- Next steps:
  - Add network energy models → system-level application energy audit



HPCCG on 32nm Sandy Bridge EP
Scaling from 1 socket (6 cores) to 2 sockets (12 cores)

Name	Package Energy Speedup	Package + Memory Energy Speedup	Time Speedup
HPC_sparsemv	1.03	1.10	1.94
waxpby	1.32	1.39	2.41
ddot	1.46	1.48	2.30
generate_matrix	0.59	0.62	1.00
sys	0.14	0.15	0.24

Extending eAudit: The lwperf Library

```
function foo(){
    lwperf_log("region1");
    // Computation of interest
    lwperf_stop("region1");

    //
    // Other work…
    //

    lwperf_log("region2");
    // Another computation
    lwperf_stop("region2");
    // …
}
```

Name	Energy (j)	Time (s)
region1	133.2	1.1
region2	552.7	3.3

- Enables measurements across regions of interest
- Integration with Eiger
  - For further analysis/modeling
- One measurement per `log/stop` pair

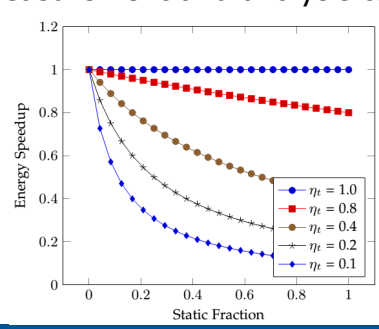
Summary

- Application design should take energy behavior into consideration to reach performance goals
- Characterize energy scaling as a function of *static* and *dynamic* energy
  - Time scaling only improves static energy
- Basis for eAudit, an energy measurement and analysis tool

$$S_e = \frac{1}{(1-s) + \frac{s}{\eta_t}}$$

$s \rightarrow$ *static fraction*

eAudit available : github.com/gtcasl/eaudit



Energy Scaling Model

Energy with *p* cores

$$E_p = (E_s \times p) \times \frac{T_p}{T_1} + E_d$$

$$E_p = \frac{E_s}{\eta_t} + E_d$$

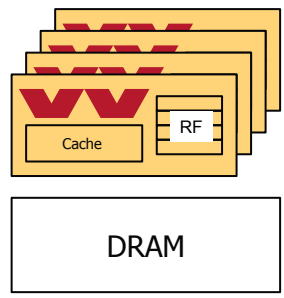
Scaling static power with time

Time and Energy Efficiency

$$\eta_t = \frac{T_1}{T_p \times p} \qquad \eta_e = \frac{E_d}{E_s + E_d}$$

Energy Speedup

$$S_e = \frac{1}{\frac{1-\eta_e}{\eta_t} + \eta_e} \qquad s = 1 - \eta_e$$



Base case is a single core executing a serial algorithm