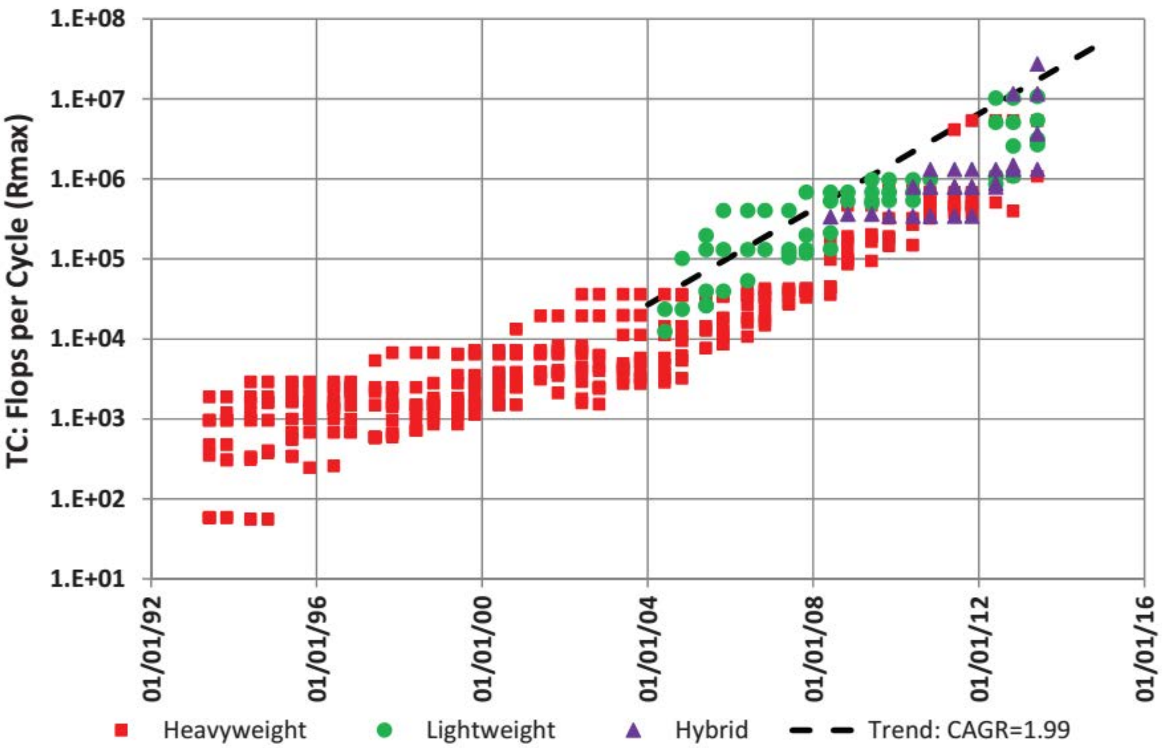
Understanding Performance—Energy Tradeoffs: When Is Energy ≠ Time?

[1]Eric Anger, [2]Jeremiah Wilke, [1]Sudhakar Yalamanchili

[1]School of Electrical and Computer Engineering
Georgia Institute of Technology

[2]Sandia National Laboratories

Scaling Performance and Power



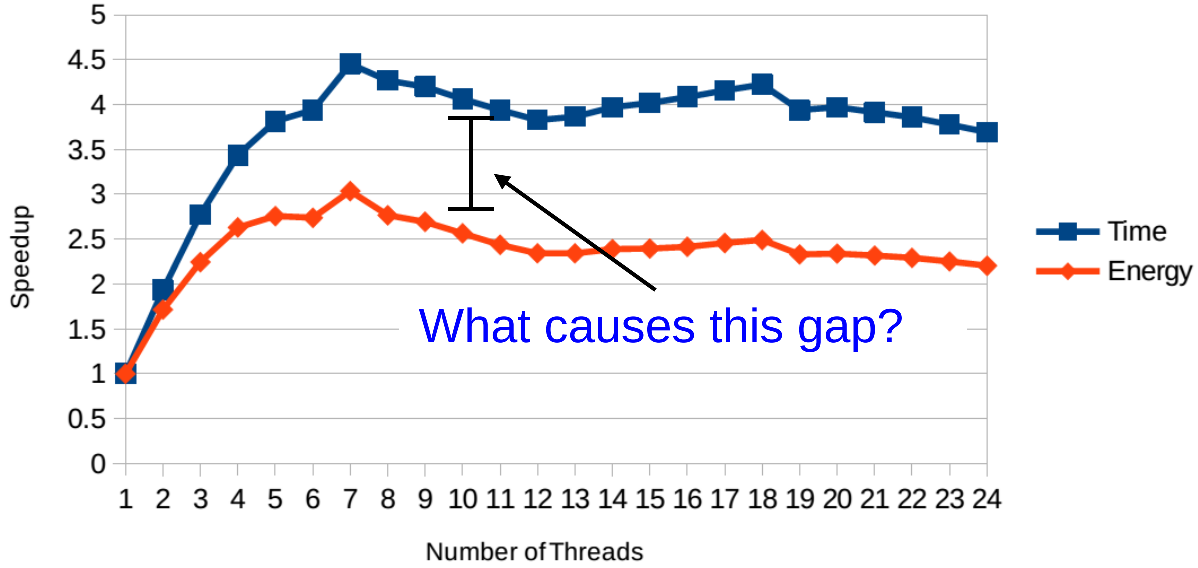
P. Kogge, D. Resnick. Yearly Update: Exascale Projections for 2013. Sandia National Laboratories.

- Increasing performance requires increasing system scale
- System power caps limit scaling
- Energy scales with number of nodes → How?

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

Energy Scaling vs. Time Scaling

HPCCG Mini-app on 32nm Sandy Bridge EP



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

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

- Developers must 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

Energy Components

- Static Energy
  - Scales with execution time
  - E.g., leakage energy
    - Technology dependent

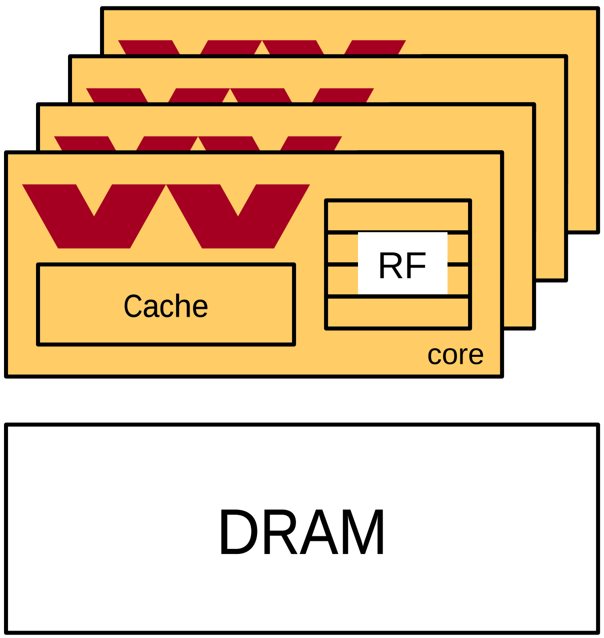
} Energy overhead

- Dynamic Energy
  - Scales with work
  - Independent of time (strong scaling)
  - 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 Law for Energy Scaling



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

f → *serial fraction*

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

s → *static fraction*

Base case is a single core executing a serial algorithm

$$\eta_t = \frac{S_t}{p}$$

η_t → *Time efficiency*

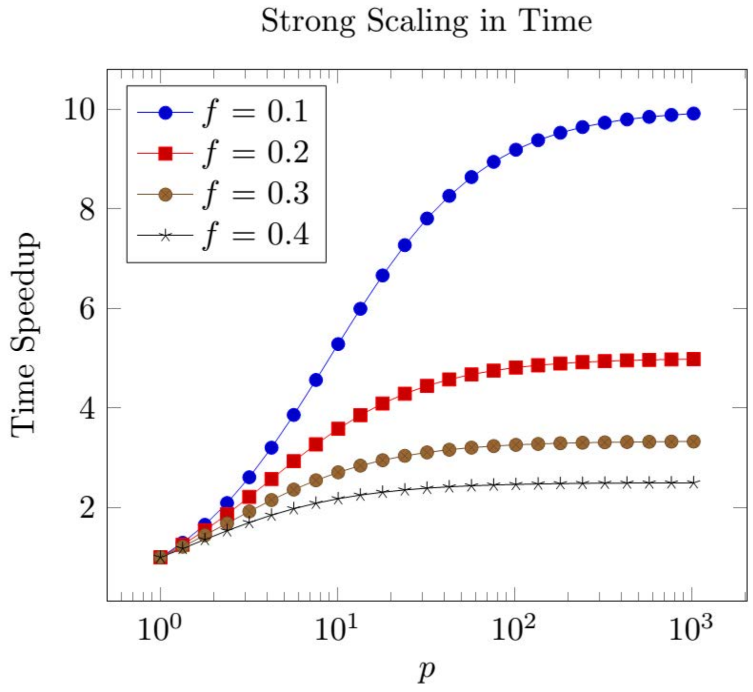
$$s = \frac{E_s}{E_s + E_d}$$

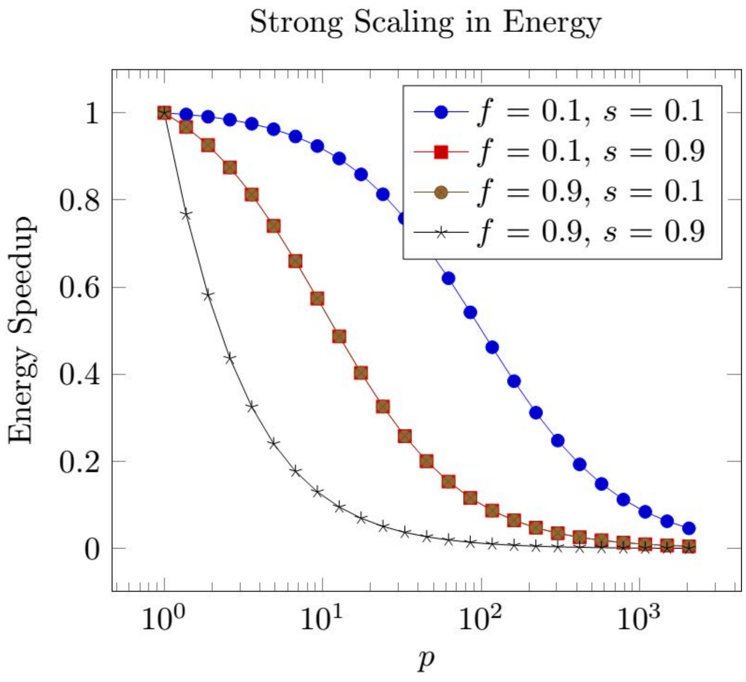
E_s → *static energy*

E_d → *dynamic energy*

- Ideal energy speedup never exceeds 1.0
- Energy speedup < 1 implies *energy penalty* for concurrency

Impact of Concurrency



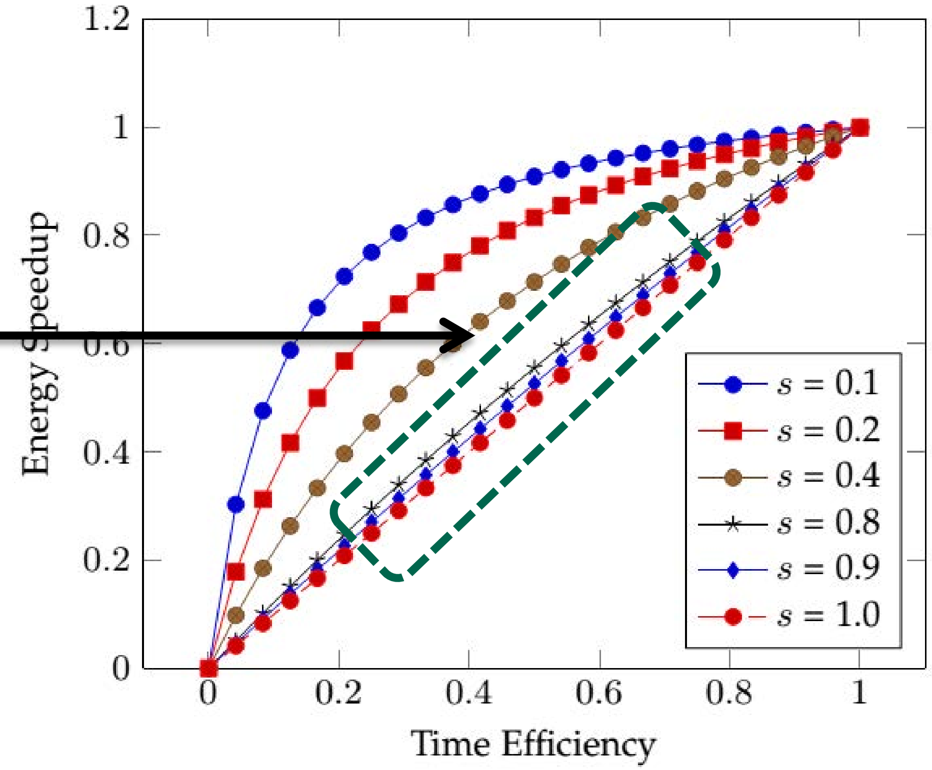


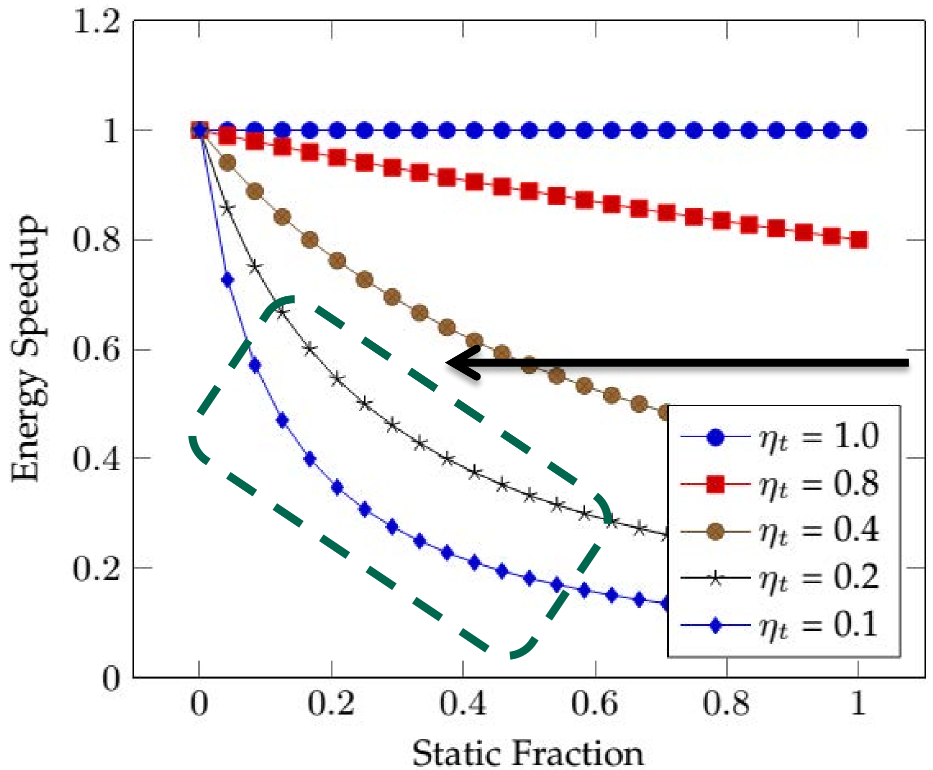
$$S_e = \frac{1}{(1 - f \times s) + p \times f \times s}$$

- 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 Behaviors

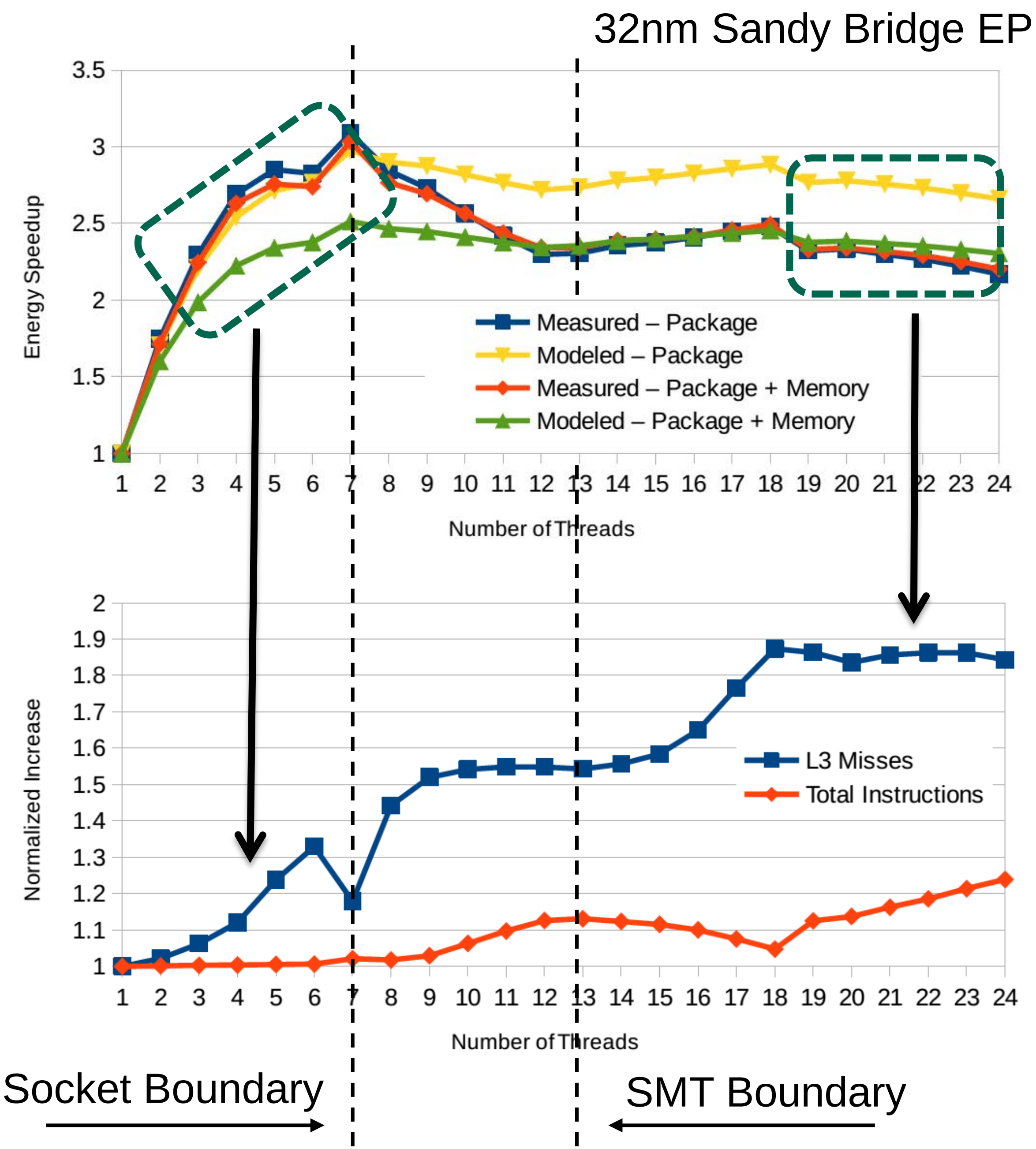
When the static fraction is high,
time speedup → energy speedup





When the serial fraction is high,
energy efficiency → energy speedup

Measurements: HPCCG

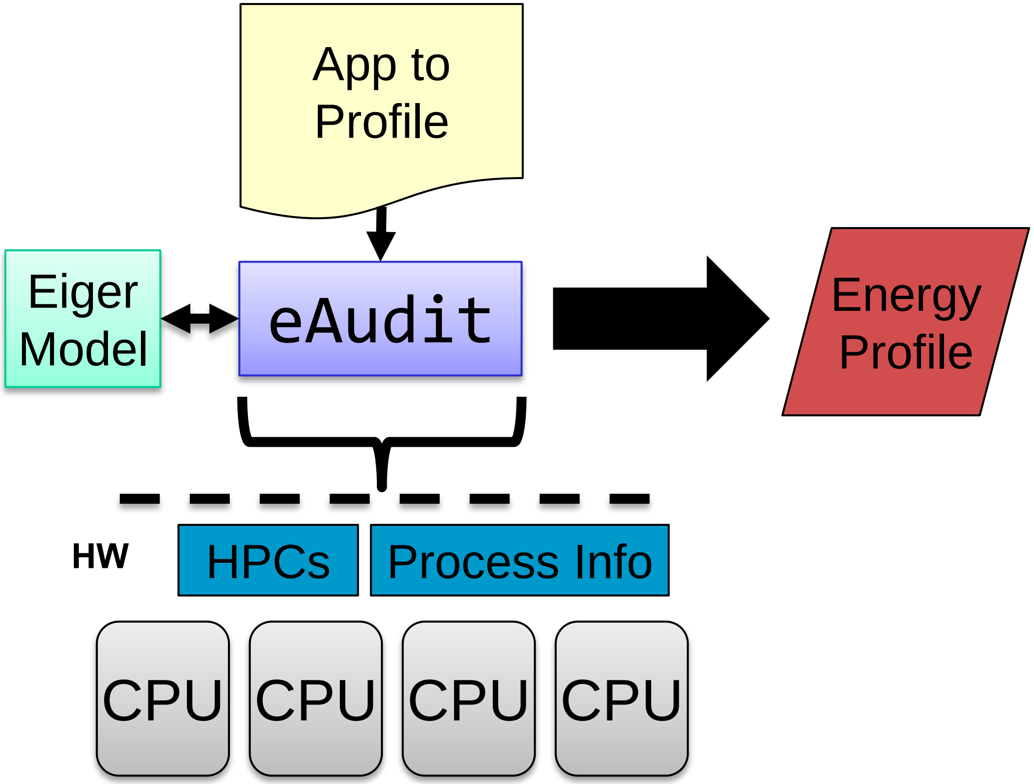


$$S_p = \frac{1}{(1-s) + \frac{s}{\eta_t \times p}}$$

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

Energy Auditing: eAudit

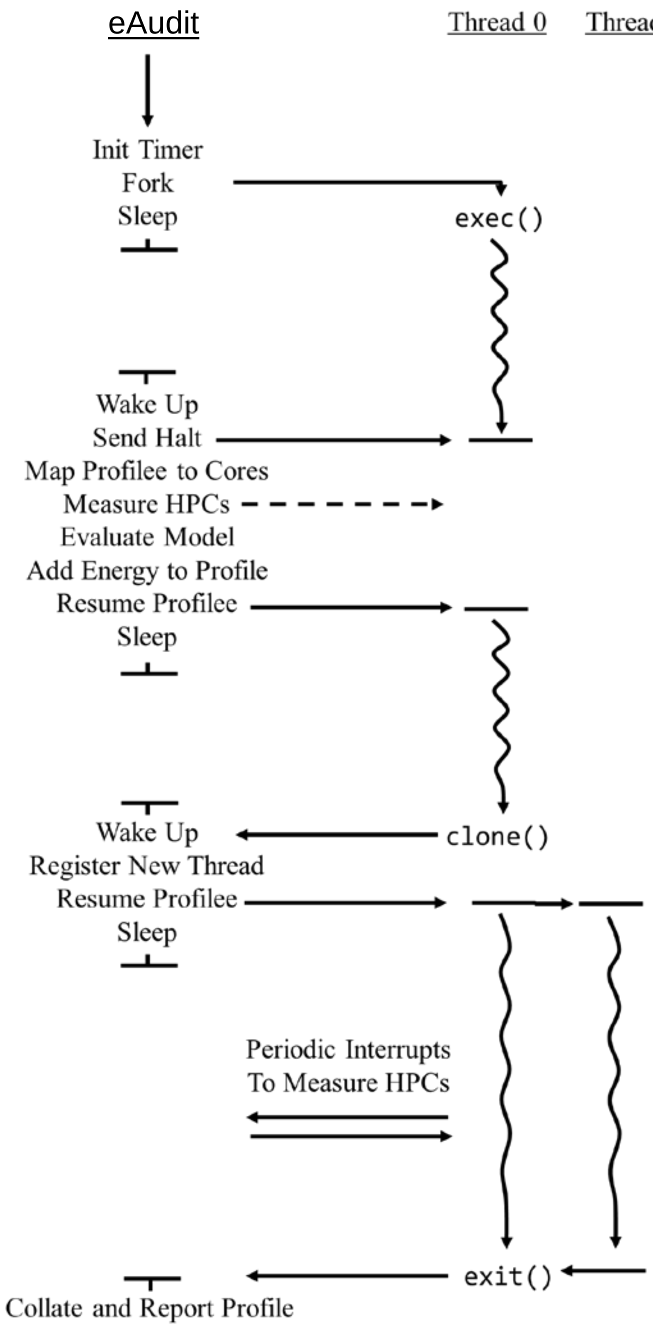
- Application energy auditing tool
  - *Function-level* attribution
- Diagnose application energy consumption behavior
- Provide actionable information to steer energy optimization

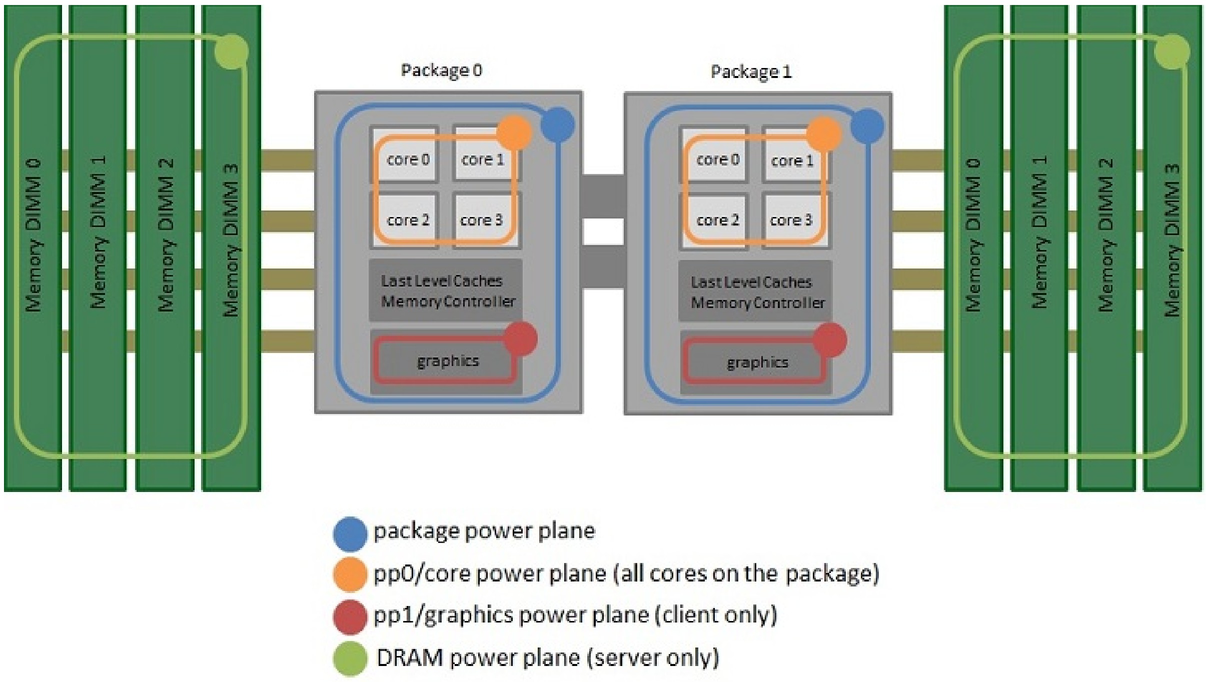


Example output

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



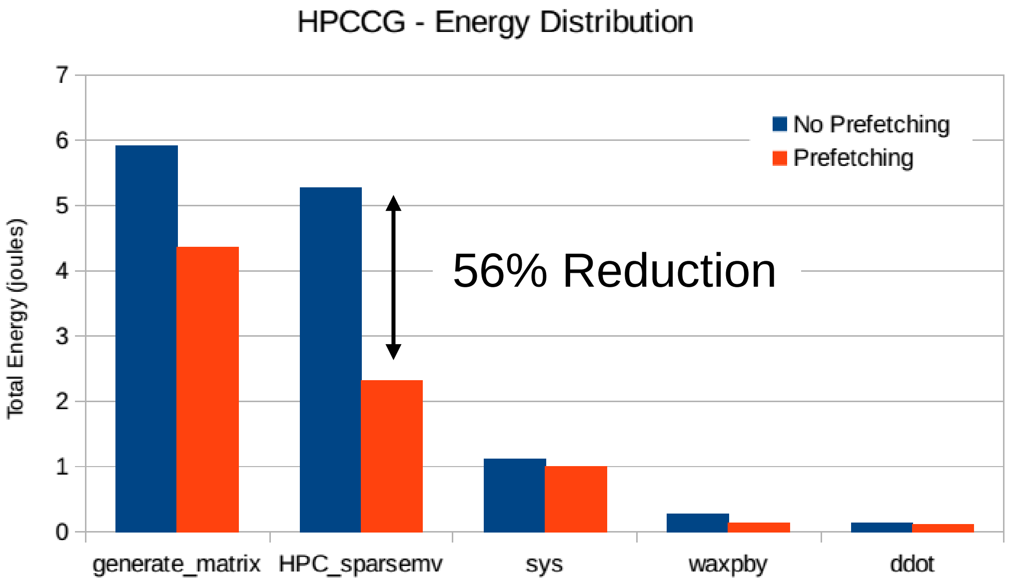


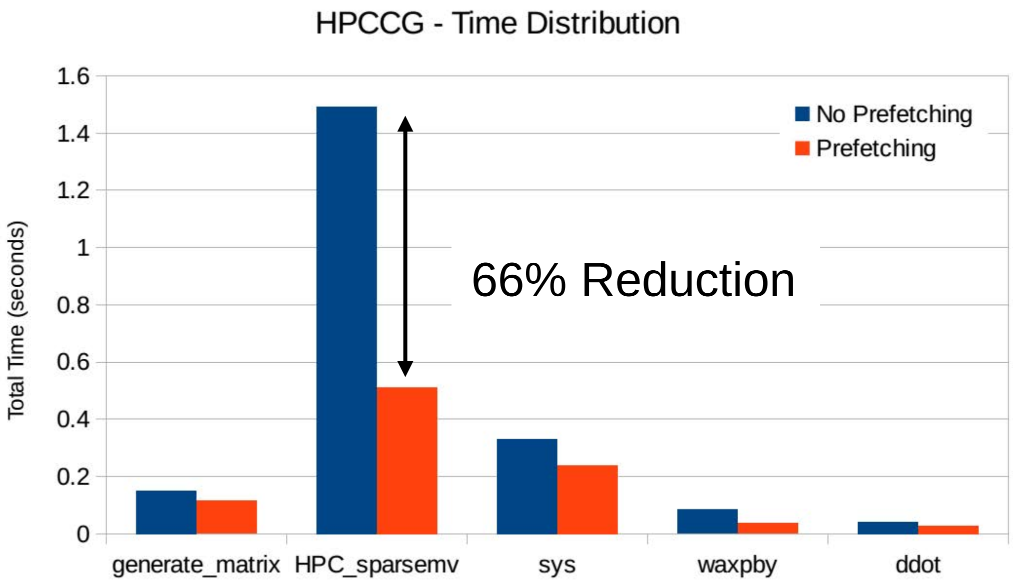
Martin Dimitrov. https://software.intel.com/en-us/articles/intel-power-governor

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

eAudit In Action: Effect of Prefetching

32nm Sandy Bridge EP

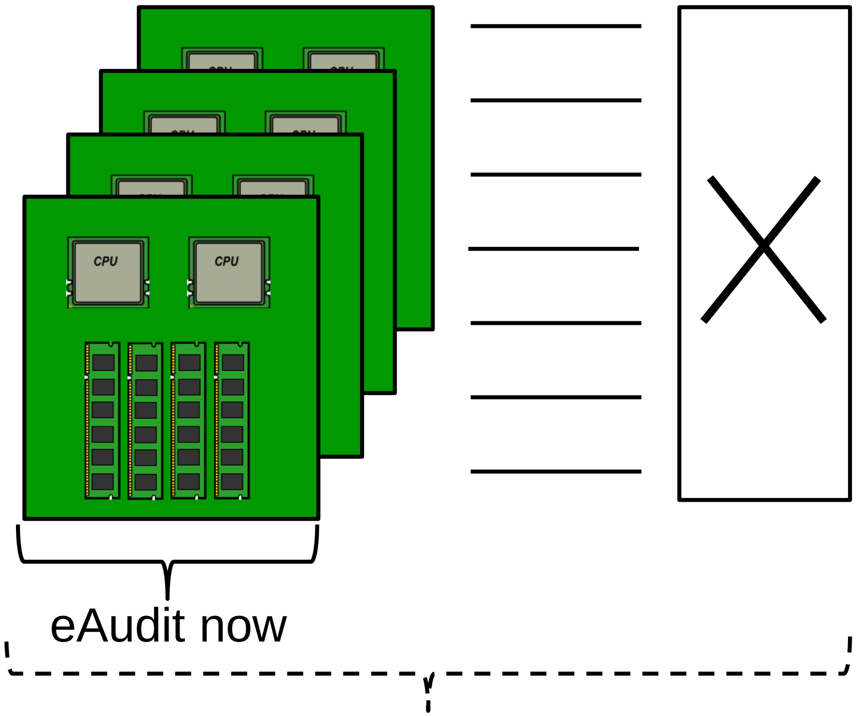




- Prefetching decreases time and energy, but not my same degree
  - Reduction in time → reduction in static energy
  - Speculative memory traffic → increase in dynamic energy

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)

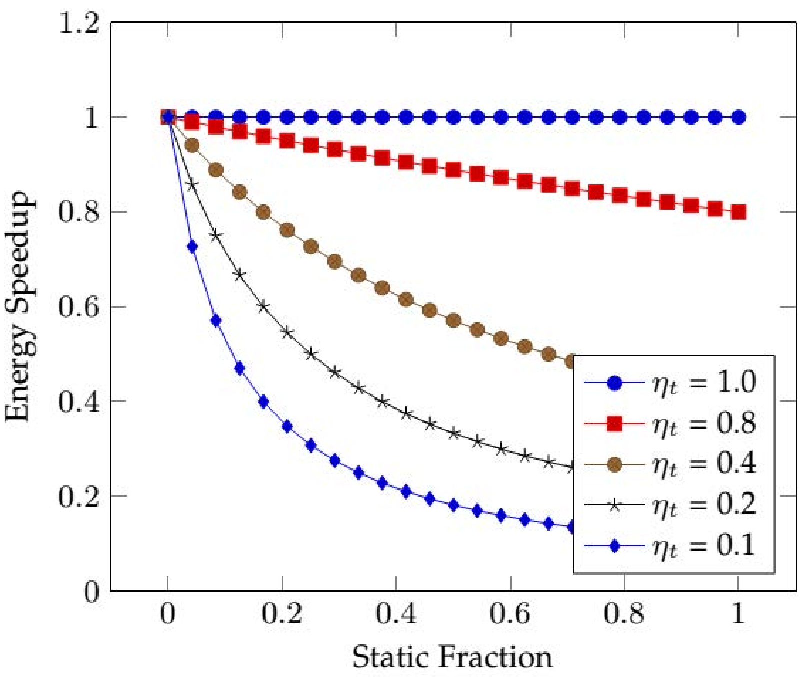
Name	Package Energy Speedup	Package + Memory Energy Speedup	Time Speedup
HPC_sparsemv	1.03	1.10	1.94
waxpby	1.32	1.39	2.41
ddot	1.46	1.48	2.30
generate_matrix	0.59	0.62	1.00
sys	0.14	0.15	0.24

Summary

- Application design must 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*



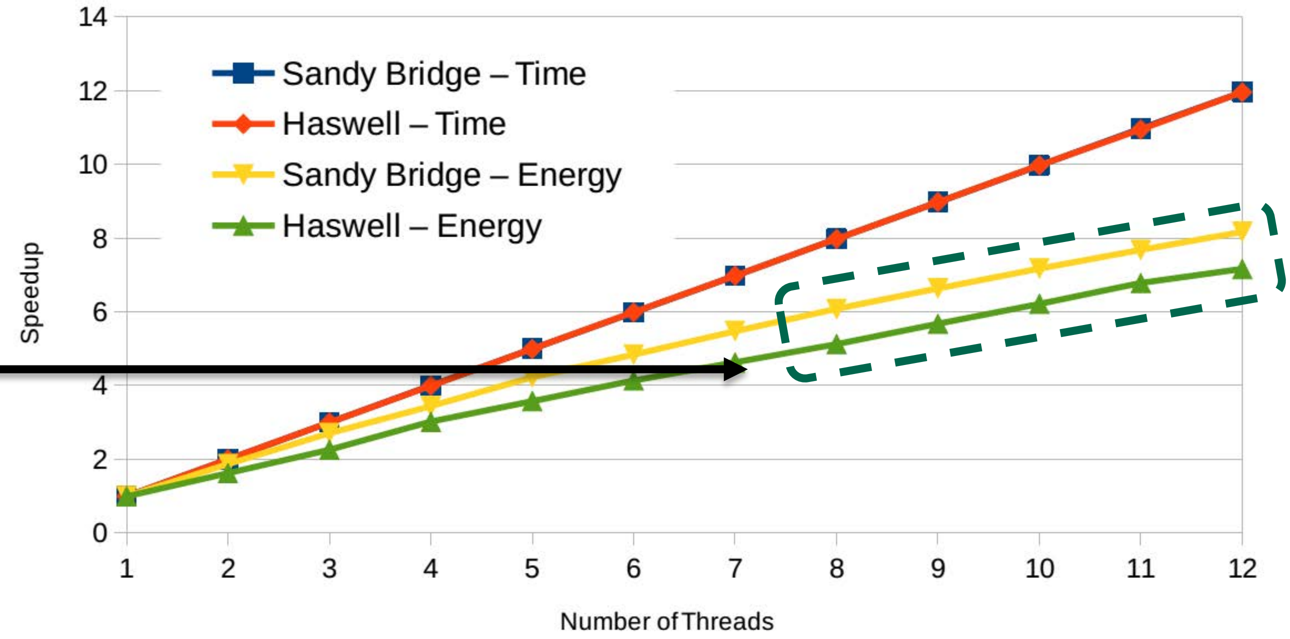
eAudit available now: github.com/gtcasl/eaudit

Backup Slides

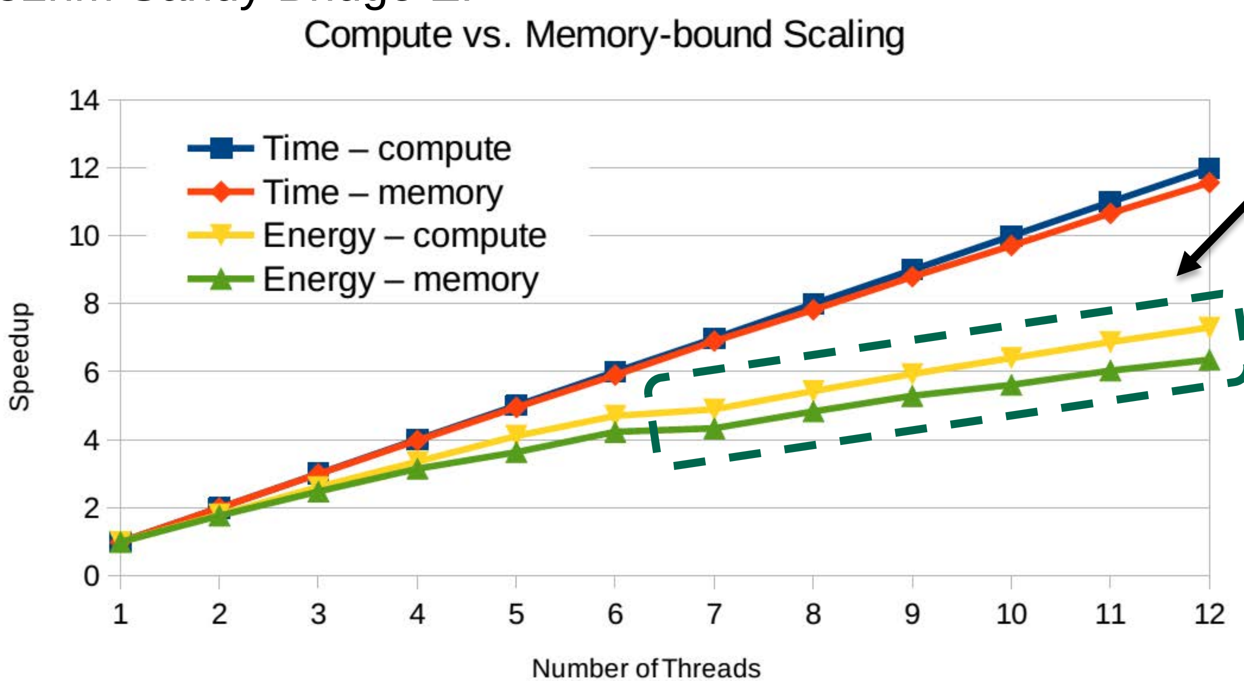
Case Studies

- Similar time speedup
- Lower energy speedup on more efficient platform

32nm Sandy Bridge EP vs. 22nm Haswell EP



32nm Sandy Bridge EP



- Similar time speedup, but different energy speedup
- Competing for memory bandwidth → increase latency → increase static energy → lower energy speedup

Energy Scaling Model

Energy with *p* cores

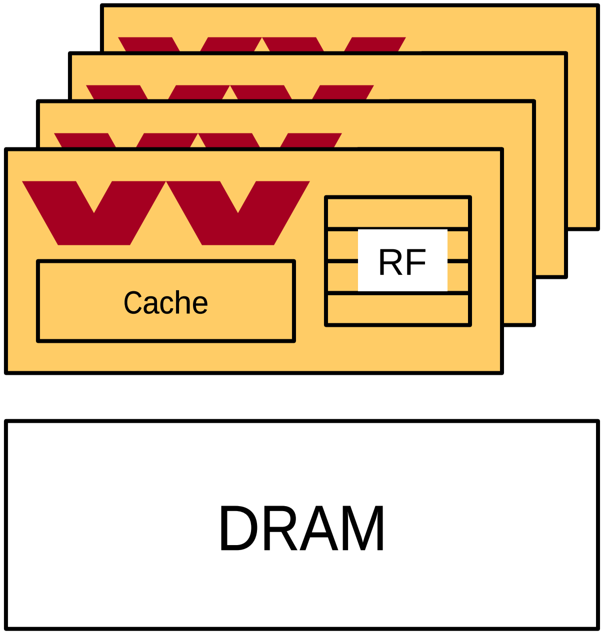
$$E_p = (E_s \times p) \times \frac{T_p}{T_1} + E_d$$

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

Scaling static power with time

Energy Speedup

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



Time and Energy Efficiency

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

Base case is a single core executing a serial algorithm