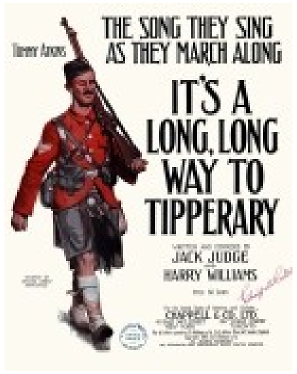
NTNU – Trondheim
Norwegian University of
Science and Technology

Energy Efficiency on the Vilje Supercomputer

Jan Christian Meyer
NTNU-IT, HPC section

PP4EE seminar
Trondheim, October 3. 2013

Outline

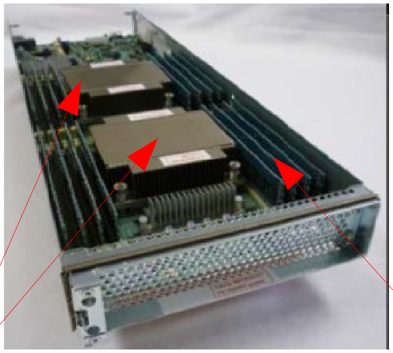

THE SONG THEY SING AS THEY MARCH ALONG
IT'S A LONG, LONG WAY TO TIPPERARY
JACK JUDGE
HARRY WILLIAMS
CHAPPELL & CO. LTD.

1. Energy on a chip
2. Power at the wall
3. Power on a chip & recognizing the application
4. Power spent on DRAM
5. Scaling up
6. ...and then?

Vilje

1404 of these make this

(Network)



2 x E5-2670 (Sandy Bridge)

Memory

Sandy Bridge MSRs

- All 2808 chips have Model Specific Registers to count the Running Average Power Limit (RAPL)
- That is, ***all these CPUs track their own energy use***

  ...
- Hooray!
- This should be fun, let us read where the Joules go.

Speed Bump #1
Privileged instructions

- Intel processors provide an instruction called RDMSR which reads model-specific registers
- It will only work with “ring 0 privileges”
  - Read: code that uses it must be part of the operating system kernel
- That would be a showstopper, we can't tweak kernels at will on this machine...

- Luckily, Linux puts the MSR values in a virtual file, so we can read it from there

Speed Bump #2
Privileged files

- The virtual file that contains these values belongs to the system administrator account, and reserves reading rights.
- Although that could be changed, there have already been lengthy debates between kernel developers and the authors of PAPI on whether to expose these values
- It may not be clever to invent novel security models on a million dollar contraption of national interest

First steps...

- It is still possible to start in the small, on systems that can be recklessly modified to our hearts' content, such as a 4-core Sandy Bridge desktop.
- Running application codes using the administrator account, the MSRs become visible...

Speed Bump #3:
The numbers pile up quickly

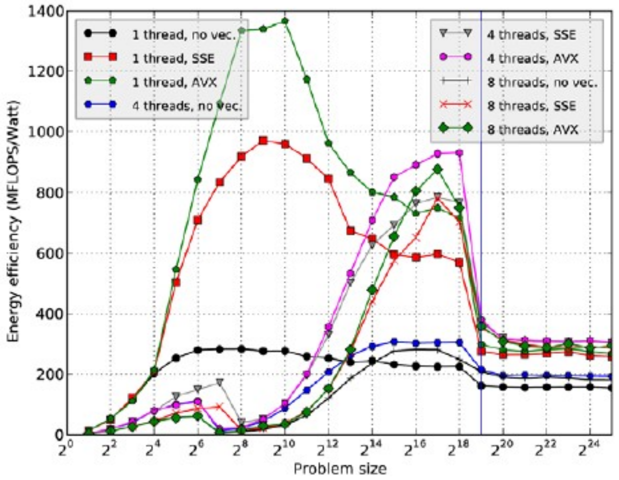
- The RAPL energy estimates come in multiples of 15.3 microJoules
- The RAPL energy registers aren't infinitely large
- Intel's documentation suggest a wraparound time on the order of minutes
- Still, with carefully selected
  - Benchmarks (BlackScholes, FFT, tiled matrix product)
  - Problem sizes (Stay within on-chip cache)

  we get:

End-to-end energy studies

"Case studies of Multi-Core Energy Efficiency in Task-based programs"
(ICT-GLOW 2012)

Full application runs on-chip in <1min, variable optimizations, thread counts:

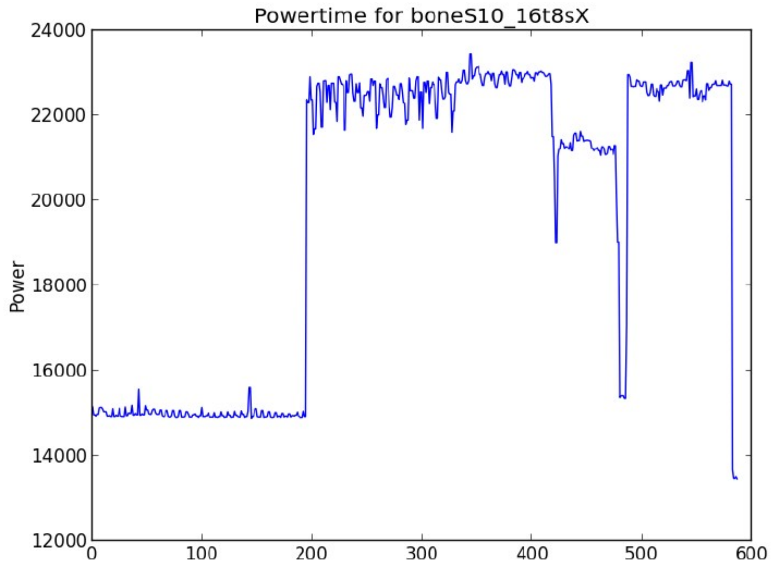


"Power instrumentation of task-based applications using model specific registers on the Sandy Bridge architecture"
(PRACE Whitepaper)

NTNU – Trondheim
Norwegian University of
Science and Technology

Energy at the wall

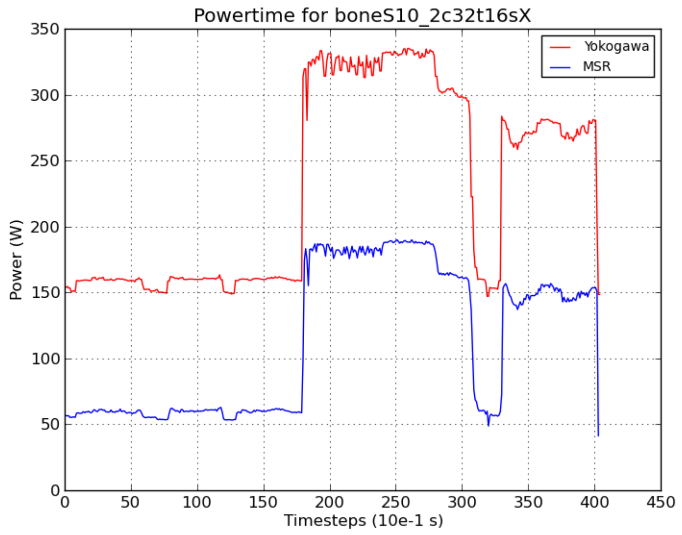
- Contact with CSLAB at NTU Athens through PRACE initiates study of CSX sparse matrix format
- External Yokogawa WT210 unit permits dedicated student to capture full-system power/time plots of CSX conjugate gradient solver:



NTNU – Trondheim
Norwegian University of
Science and Technology

Continuous RAPL numbers

Signal/callback implementation of MSR reading enables similar readouts from chip, for comparison:



That takes care of the register wraparound issue, and produces...

NTNU – Trondheim
Norwegian University of
Science and Technology

Increasing accuracy on a single machine

Continuous RAPL results in workshop paper with CARD, CSLAB

“Energy-efficient Sparse Matrix Autotuning with CSX - A Trade-off Study”

(IPDPSW 2013)

and 3 more PRACE Whitepapers:

“Energy-efficient Sparse Matrix Auto-tuning with CSX”

“An Energy-centric Study of Conjugate Gradient Method”

“Implementation of an Energy-Aware OmpSs Task Scheduling Policy”

...so, there were papers.

- In partial completion of today's topic, they have
  - taken steps towards capturing detailed application power profiles
  - utilized machines that are similar to the nodes of our supercomputer, more or less
- It's been an interesting road so far.

NTNU – Trondheim
Norwegian University of
Science and Technology

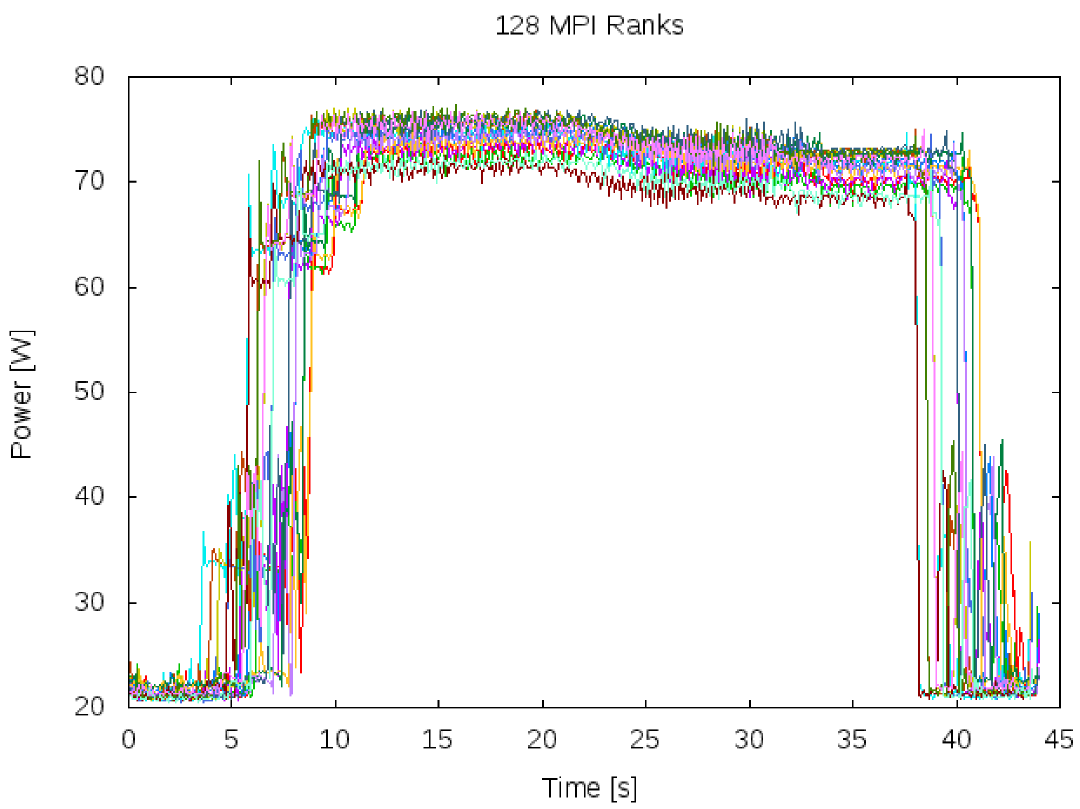
Reaching for Vilje

- In a sense, we have still not reached the goal of observing HPC applications in the wild.
- Speed bump #2 remains, and there are more:
  - Administrator privileges can't be free-for-all on a multiuser system
  - Integrating instrumentation code with the application is not a realistic ambition for a large number of diverse codes, with and without developer involvement
  - External instrumentation code with selectively raised priviliges isn't trivial to implement over remote file systems
  - (...further tedious details are available on demand...)
- Most of these have been at least partially addressed as they appeared, so we are close enough to do this:

A vulgar display of power/time

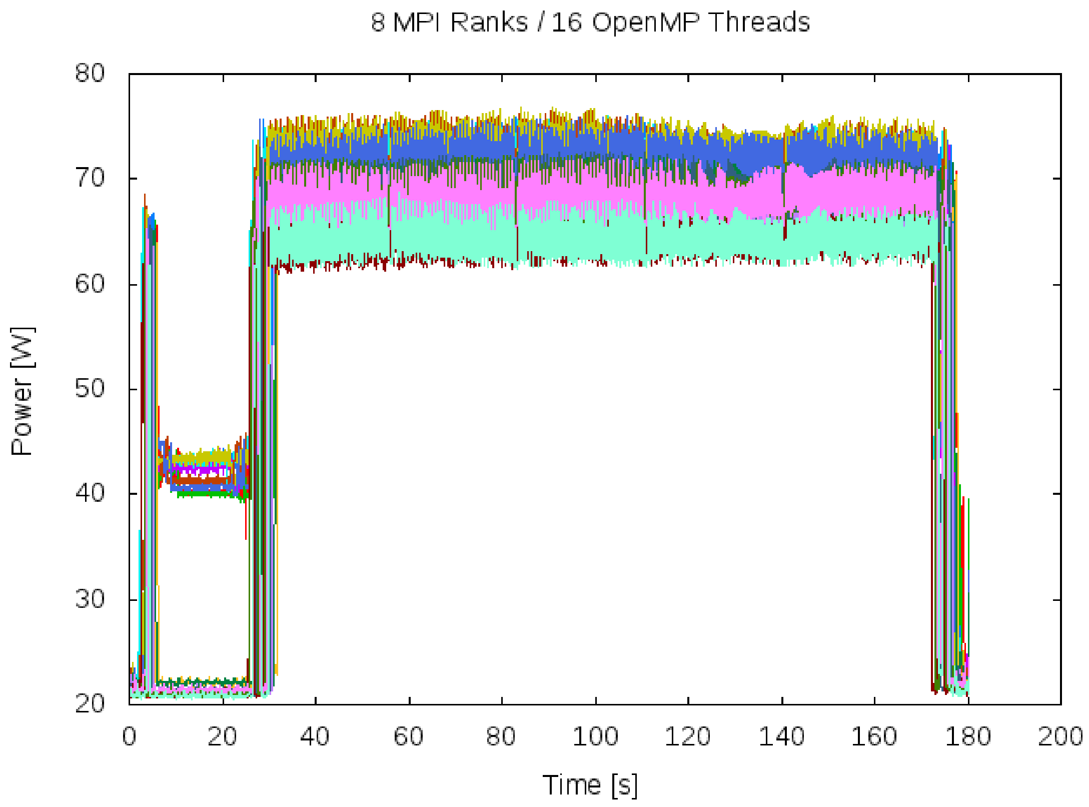
- Case study:
  - LAMMPS classical molecular dynamics code
  - 10718750 simulated iron atoms on 128 cores
  - Straightforward configuration runs 128 MPI processes, one per core
  - Hybrid configuration runs 8 MPI processes with one per node, and 16 OpenMP threads per process
- Per-chip instrumentation processes carefully started by use of (administrator) hand on 16 chips
- Parallel job launched in the regular batch-system way, carelessly starting on *those* 16 chips by use of educated guess.

Pure MPI run



NTNU – Trondheim
Norwegian University of
Science and Technology

Hybrid MPI/OpenMP run



What we can tell

- Startup / compute / finalize phases
  - MPI totals are 6.7, 21.3 and 3.5 kJ, respectively
  - Hybrid totals are 14.2, 156.3 and 5.6 kJ, respectively
- Hybrid startup clearly employs exactly 1/2 of the cpus
  - MPI ranks launch on core 0
- Standard deviation of per-processor consumption differs by an order of magnitude
  - Fork/join parallelism in Hybrid

What we can not tell

- It is not a thorough study of LAMMPS
  - the benchmark case is short, and artificially pleasant
- It is not a thorough study of MPI vs. Hybrid execution
  - the benchmark case is small
- It is not a thorough analysis of the result material
  - I have no clue what the spike at the beginning of the Hybrid run was
- Significant digits are scattered in all directions
  - Synchronization between the application run and the instrumentation is tenuous at best, eye measure at worst

Future work

- DRAM energy is actually possible on Vilje, but it is presently disabled by a default BIOS setting
  - A tedious direction for future work is rebooting 1404 nodes just to press 'delete' and 'enable' when they come up again
- Letting the user job trigger the instrumentation is within reach
  - The software is ready, but can not run until it is installed in each node's system image
- Producing clever visualizations from many, many trace files is a challenge in itself
  - The graphs we saw won't win any prizes for clarity, and they aren't even from large runs
- ...and obviously,

  Finally characterize the power profiles of production applications

Credit where credit is due

Almost all of the above are the contributions of

CARD group at NTNU

CSLAB at NTUA

PRACE

NTNU – Trondheim
Norwegian University of
Science and Technology

References

Case Studies of Multi-core Energy Efficiency in Task Based Programs

Hallgeir Lien, Lasse Natvig, Abdullah Al Hasib, and Jan Christian Meyer,

ICT-GLOW'12, LNCS 7453, pp. 44–54. Springer, September 2012

Energy-efficient Sparse Matrix Autotuning with CSX - A Trade-off Study

Jan Christian Meyer, Juan Manuel Cebrian, Lasse Natvig, Vasileios Karakasis, Dimitris Siakavaras, and Konstantinos Nikas, Proceedings of IPDPS-2013, IEEE CPS, May 2013

Whitepapers are all online at

http://www.prace-ri.eu/Power-efficiency

Thank you for your attention!

...are there any questions?