High Level methodology for applications characterization and extrapolation

*Philippe Thierry**
Asma Farjallah and Amine Mrabet***
**Intel corp*
*** ENS Cachan*

Software and Services Group | Energy Engineering Team

High level roadmap and (customer) questions

		Past (was good)	Present (excellent)	Future (even better)
Multi Cores	4S			
	2S			
	1S			
	1S+Gen			
Many Cores	Boot			
	Copro			

Standard questions:

- Use of new instructions set
- Impact of nb of cores
- Impact of memory hierarchies, bw and latency

New questions:

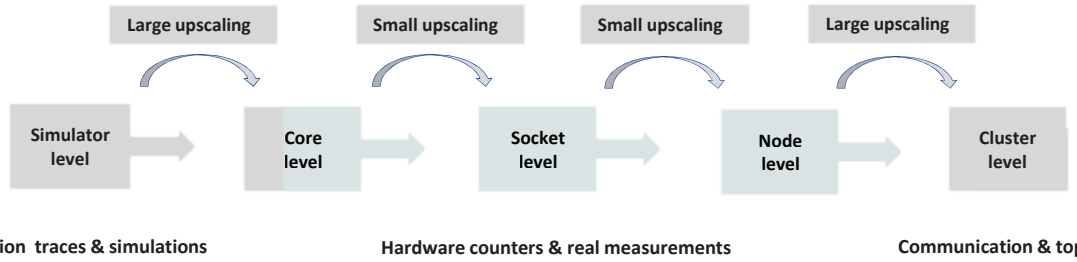
- New cores, new instructions set
- Impact of nb of thread: Amdahl is back !
- Is MPI+OMP always needed, What about GB per core
- Affinity, First touch, « Cluster on die »

- How my datacenter will look like in 2y from now

Answers : applications will tell

Model application's behavior at several levels:

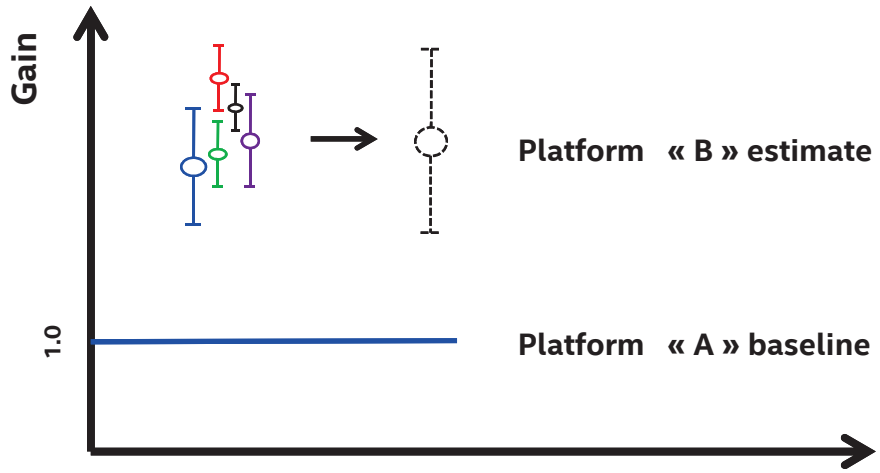
- Determine current performance (« characterization »)
- Formalize an extrapolation model or use simulators
- Extrapolate performance on future hardware
- Size the future machine that will best match one or more applications
- Influence micro-u designs (intel internal)



Longer term objectives

Platform « A » → **Platform « B »**

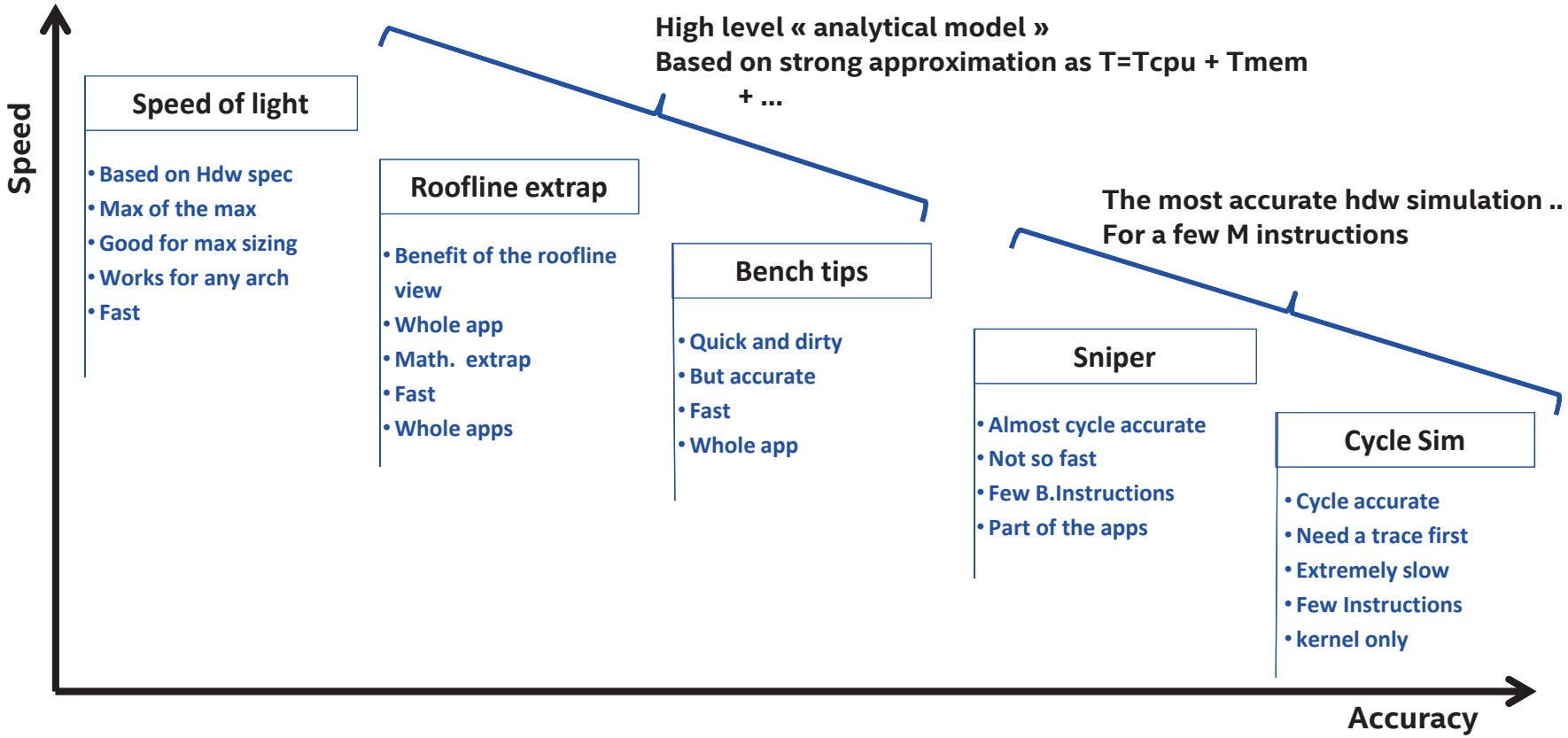
- **There is NO single answer to this problem**
- **Results must come from different views AND include uncertainties**



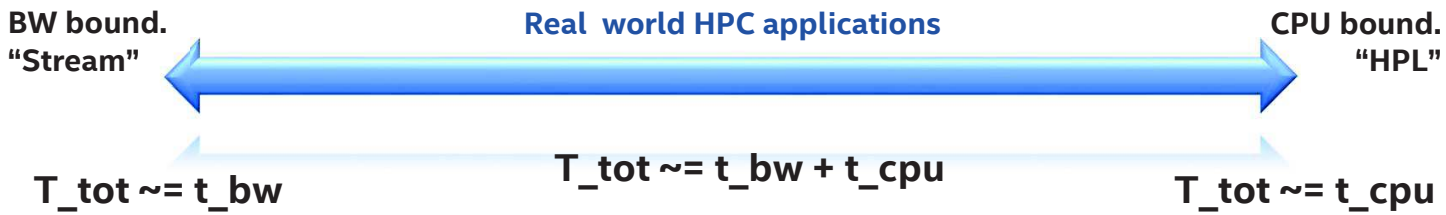
Agenda

- Goals
- **Extension of the roofline analysis**
- High level extrapolations

Try to be pragmatic first ..



First order approximation & apps classification



BW demanding applications are bounded by BW ratio of platforms A & B

Flops/s demanding applications are bounded by FP ratio platforms A & B

Analyzing this ratio between 2 computers will give a first guess defined as « speed of light»

Hypothesis: *Same efficiency on both sides (implementation, compiler, OS)*

Problem: *how much to remove from this limit to account for efficiency, OS, Compiler effects ..*

Hypothesis (related to $t_{tot} = t_{mem} + t_{cpu}$)

- Does not account for the effect of cache size.
  - If CacheSize(B) >> CacheSize(A) , the run on B will be less sensitive to bandwidth
  - It will produce conservative estimates
- Does split Memory and CPU contributions while in reality they can overlap completely.
- Does not account for DRAM latency (won't work for latency bounded code)
- Does not account for inter-node communication effect nor I/O
  - those are implicitly accounted for

Several solutions exist to go 1 level down (caches & latency) and 1 level up (cluster)

Extended Roofline Model

$$GFlop/s(AI) = min \begin{cases} p_f \\ AI \times p_b \end{cases} or \ \min \begin{cases} xGEMM \\ AI \times StreamBWD \end{cases}$$

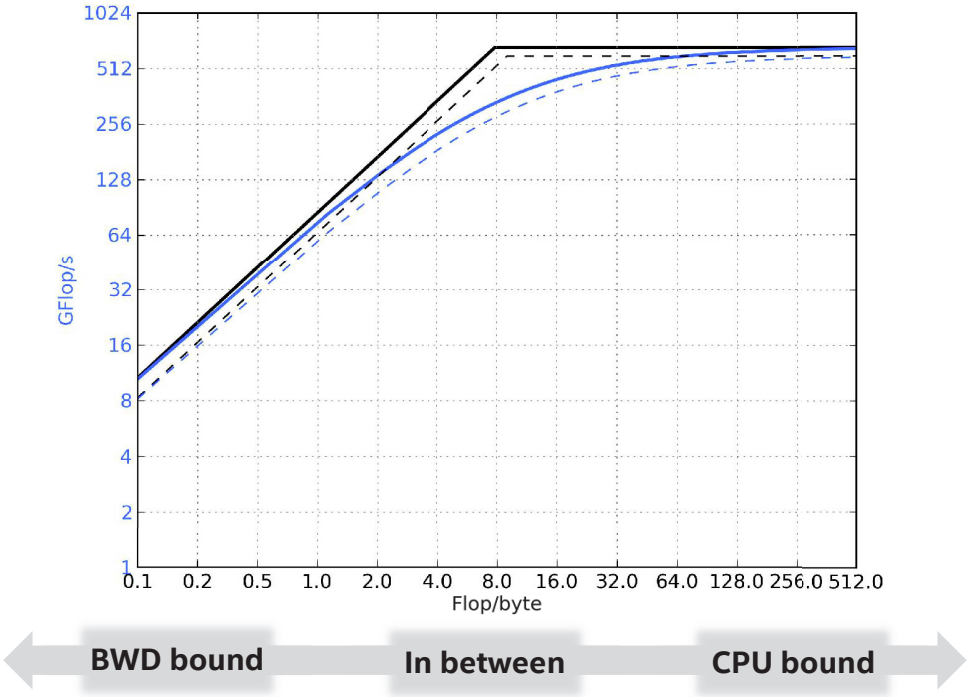
Extended roofline derived from the simple model

$$t_{tot} = t_{mem} + t_{cpu}$$

$$GFlop/s = \frac{AI\ p_b\ p_f}{p_f + AI\ p_b}$$

- Will impact the “in between applications” characterization
- Explicitly handling caches in the model would help (WIP)

AI: arithmetic intensity
p_f: peak FP
p_b: peak bandwidth



How to collect Flops and Bytes (AI def)

$$AI = \frac{Flop}{Byte}$$

SDE

- ✓ Possible for future architectures
- ✓ Sum over execution time. (standard usage)

Hardware counters

- ✓ Not always available
- ✓ Vtune, PCM, LIKWID, PAPI
- ✓ As f(t)

By hands

- ✓ Not always possible
- ✓ Scalar

Hardware counters

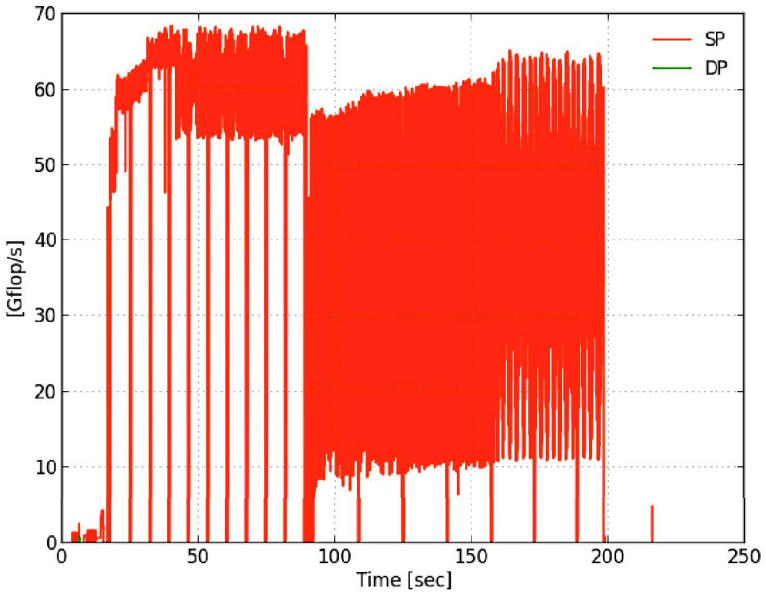
- ✓ Vtune, PCM, LIKWID, PAPI
- ✓ As f(t)

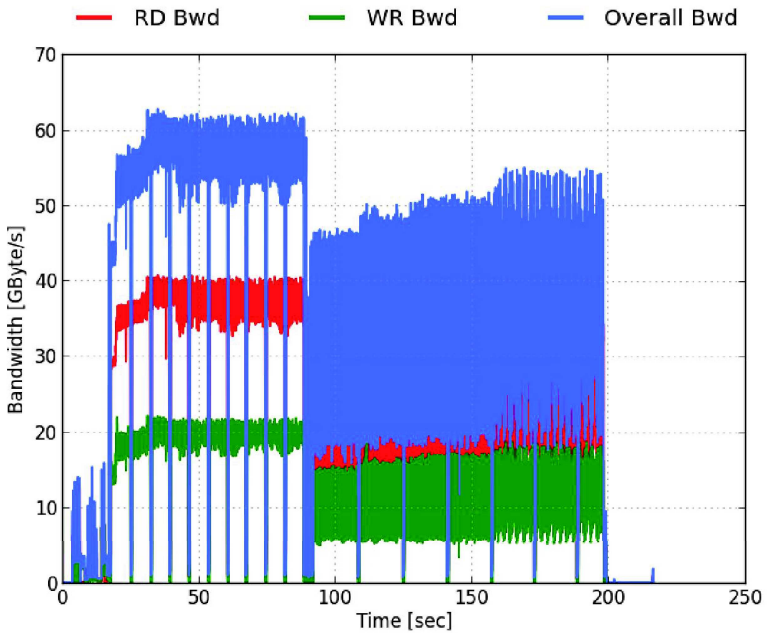
By hands

- ✓ Not always possible
- ✓ Scalar

How to define memory demand without cache impact ?

Tools allow counter collection = f(t)





Seismic Imaging on IVB E5-2697. FP ops and Memory bandwidth collected as a function of time for the whole application

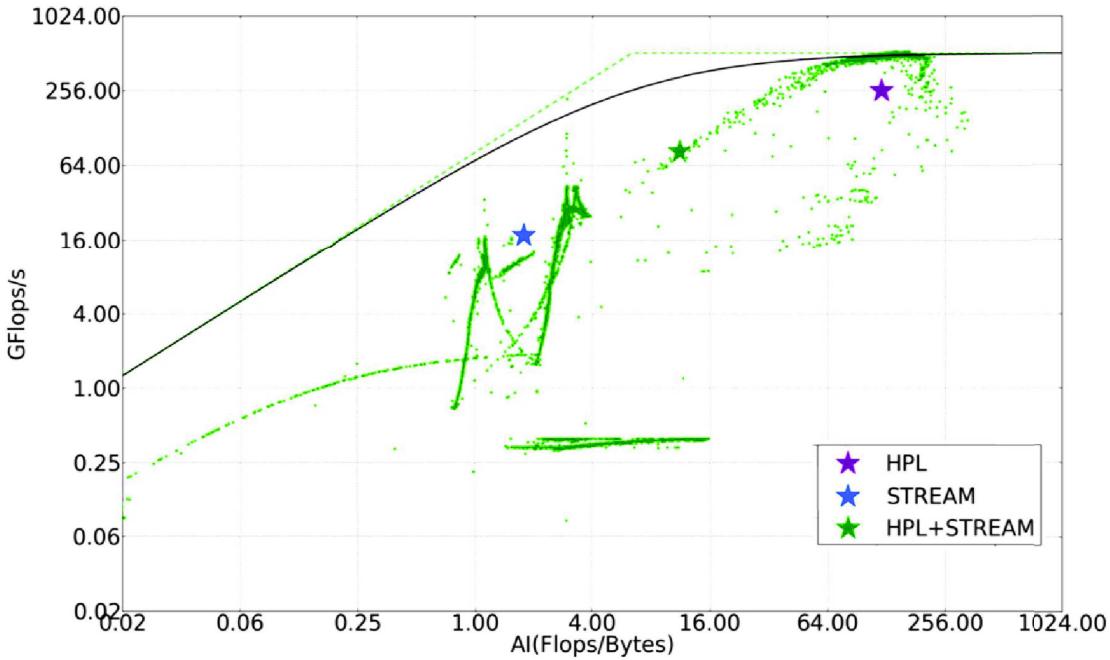
Temporal roofline: Application phases identification

Let us consider an application doing

- ✓ 50 % of Stream Triad
- ✓ 50 % of DGEMM

Temporal roofline identifies these phases distinctively.

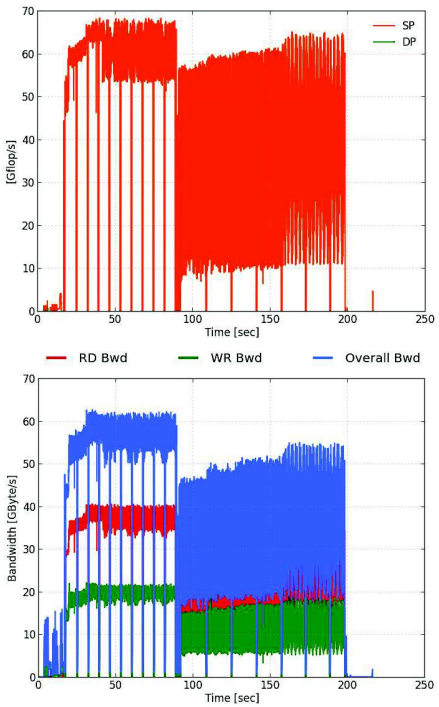
Scalar AI just gives an average value

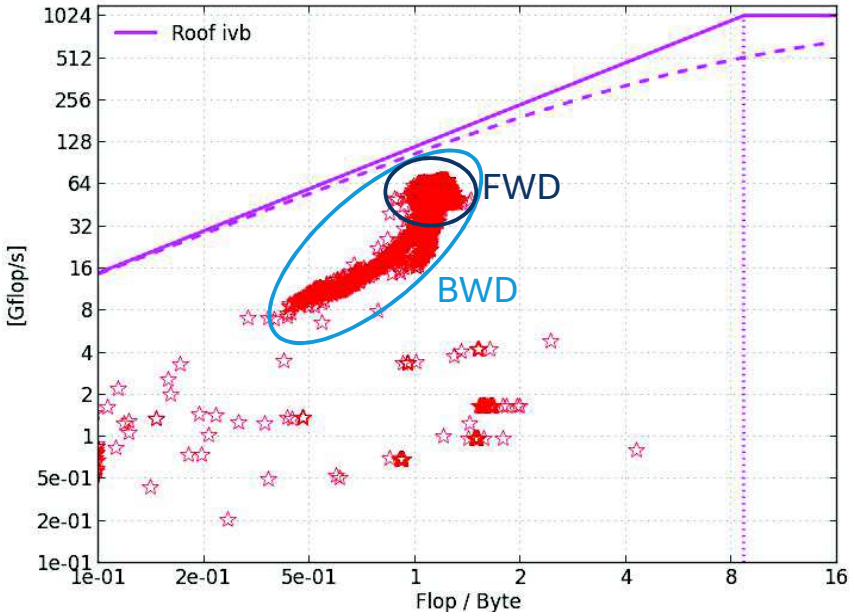


Scalar AI is not representative for application including different kernels

Note that FP counters are wrong on E5-2697v2 (over estimated here)

Temporal roofline on a real world application

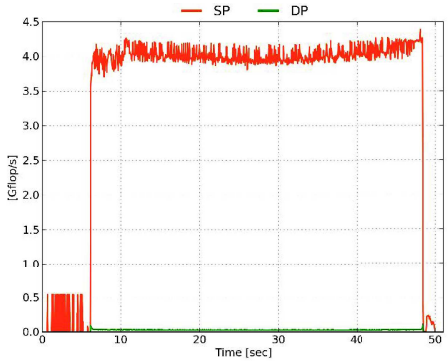




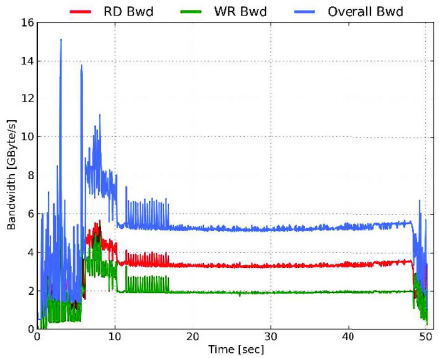
Seismic Imaging on IVB E5-2697. FP ops and Memory bandwidth collected as a function of time for the whole application.

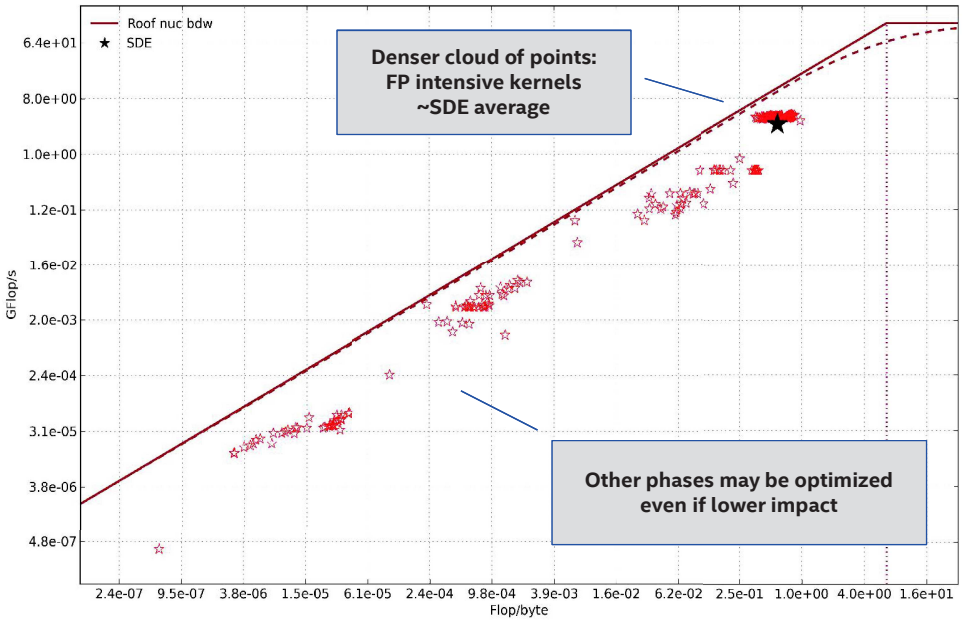
Next Steps : being able to link any single point with corresponding source code and assembly.

Example on Broadwell i3. FPs are back..



Different phases, different FP and BW behaviors



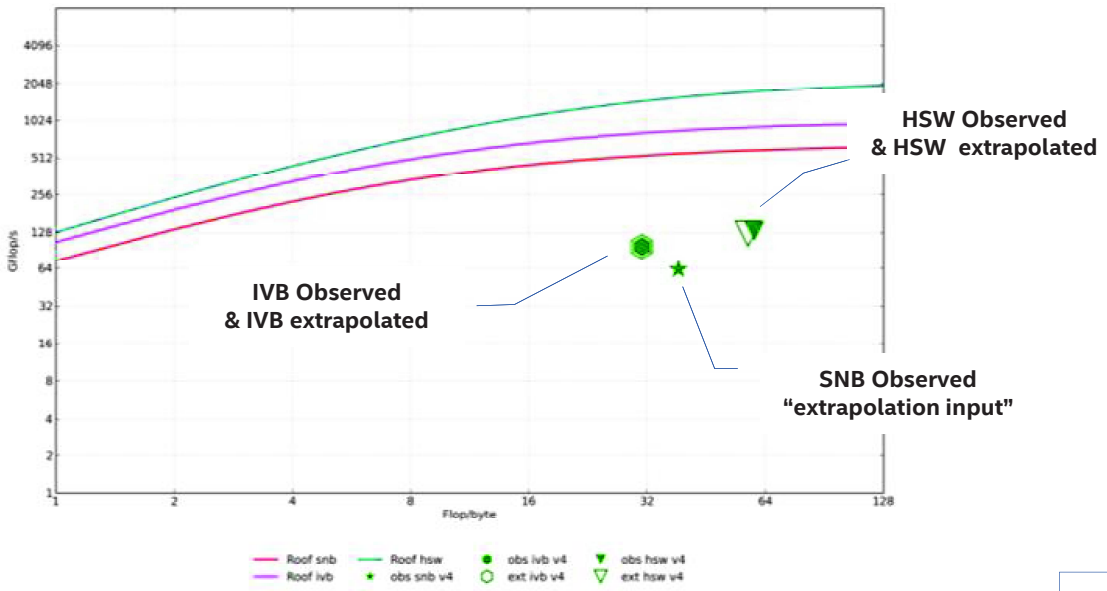


When FP counters are correct, both SDE (sum over time) and AI(t) are coherent

Agenda

- Goals
- Extension of the roofline analysis
- **High level extrapolations**

WIP. From extended and temporal Roofline: Extrapolation



- ✓ Works fine using 2 tuning parameters that must be related to hardware and applications.
- ✓ Handle AI changes
- ✓ Effective AI is not a constant !

$$x_B = \varepsilon_x \left[(\varepsilon_y - 1) \frac{1}{p_{bB}} + \varepsilon_y \frac{p_{bA}}{p_{bB} \times p_{fA}} x_A \right] p_{fB}$$

$$GF_B = \varepsilon_y GF_A \frac{S_B\left(\frac{x_B}{\varepsilon_x}\right)}{S_A(x_A)}$$

« In between applications » : Bench tips.

Let's make 2 runs of the applications on platform A

- with the max of cores / node
- with half nb of cores / node:
  - 2x more nodes needed in scatter mode
  - But provides 2x more BW per node
  - Then $t_{\_bw}$ should be 2x smaller at max

$$T_{tot\_1} = t_{\_cpu\_A} + t_{\_bw\_A}$$

$$T_{tot\_2} = t_{\_cpu\_A} + (t_{\_bw\_A}) / 2$$

Solving this linear system gives $t_{\_cpu\_A}$ and $t_{\_bw\_A}$

« In between applications » : Bench tips

From the "CPU Freq scaling"" and "BW scaling" equality between Platforms A & B

We can easily obtain the *„cpu"* and *„mem"* contribution on platform B, $t_{\_cpu\_B}$ and $t_{\_bw\_B}$

$$t_{\_cpu\_B} = t_{\_cpu\_A} * (GF_A / GF_B)$$

$$t_{\_bw\_B} = t_{\_bw\_A} * (BW_A / BW_B)$$

where we considered *the „cpu" part non correlated with the „mem" part*
(which is an extremely strong hypothesis)

But this doesn't work if we consider a simple frequency Scaling *(GF_= Frequency in the above equation)*

Extrapolation on the same micro-u (IVY / SNB) : 0. 84 %

Extrapolation on different micro-u (HSW / SNB) : - 40%

Extended frequency scaling for different arch.

Comparison of Real measurements (FWI) on SNB (e5-2670) , IVY (e5-2697v2), HSW (e5-2697v3)

We Need to extend the « CPU » contribution from simple frequency scaling

Let us define GF as:

GF = (FP_ops/FP_inst)_theo * % vecto * (Inst/cyc) * (Cyc/sec) * eff * nbc

hdw SDE on App hdw hdw App hdw

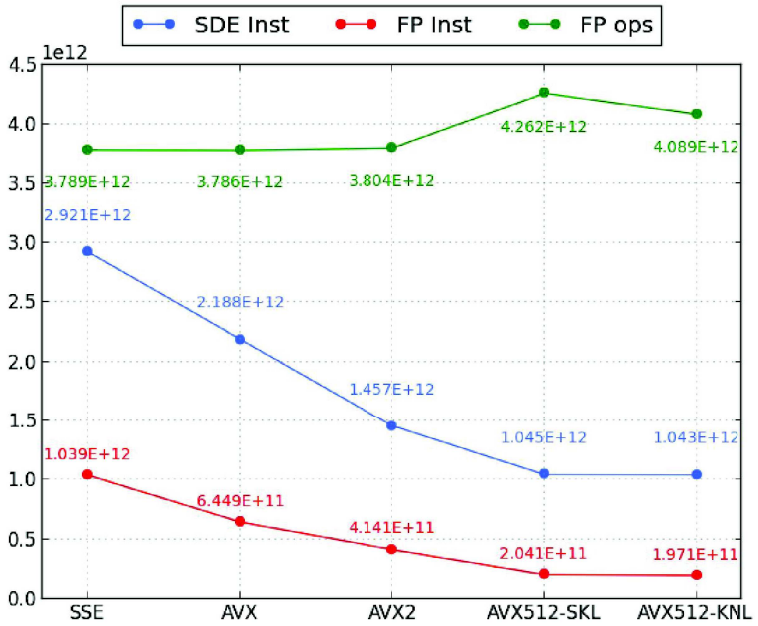
nbc : denotes the nb of cores

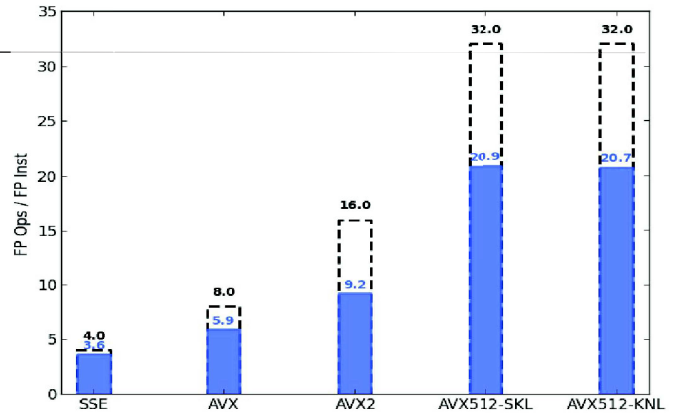
eff : denotes HPL efficiency * Amdahl scalability

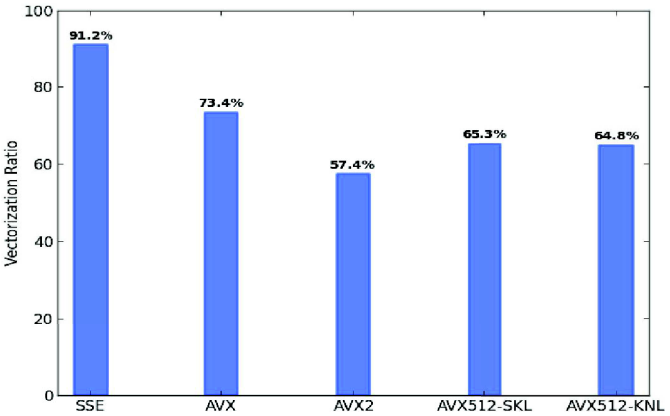
Extrapolation on the same micro-u (IVY / SNB) : 0. 13 %

Extrapolation on different micro-u (HSW / SNB) : < - 5 %

SDE Instruction Mix



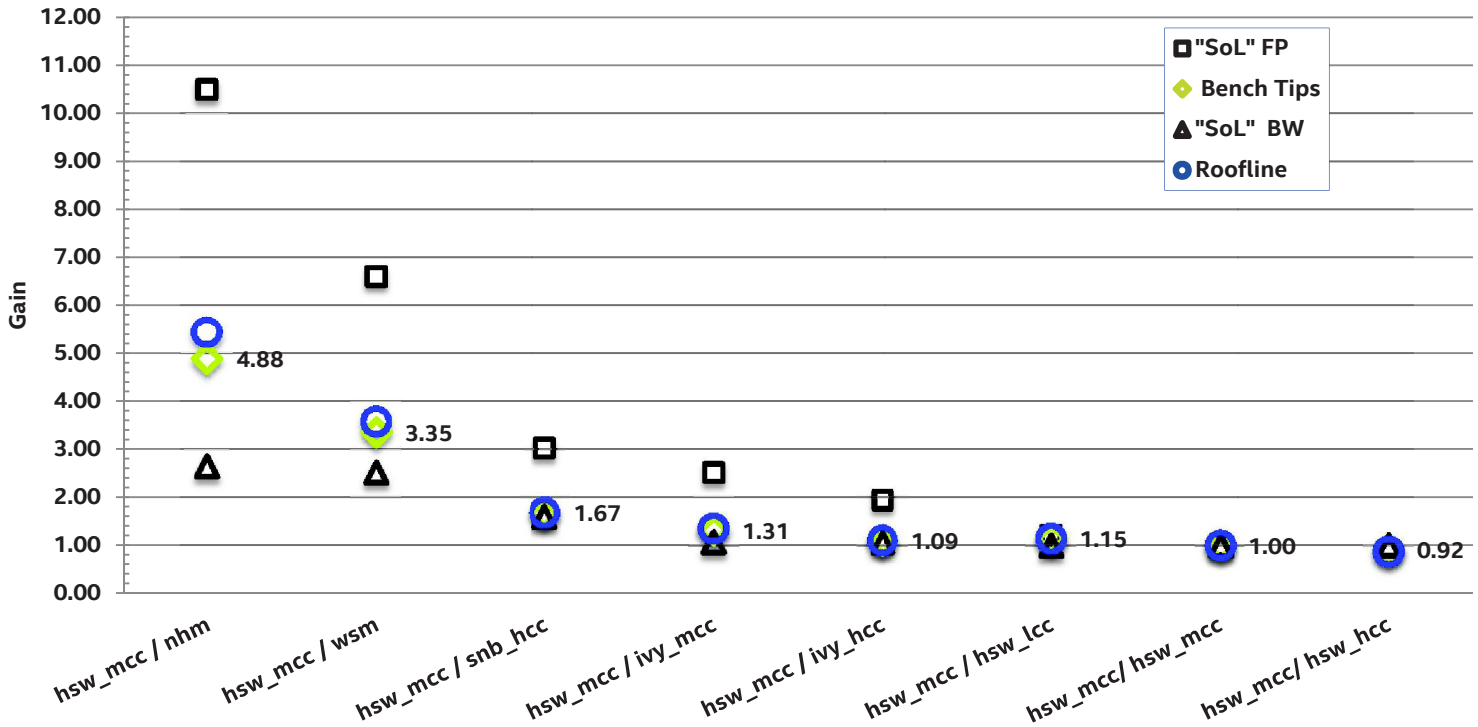




Current Intel Compiler. Current SDE. All available tools

https://software.intel.com/en-us/articles/intel-software-development-emulator

Example for Extrapolated « gain » for seismic imaging (fwi2d)



Can be expressed as time, Flops, Ops/Joule, Shot per Watt , Shot per day, … for a few years ahead

Prediction vs measurements (“seismic imaging”)

Once measurements have been done

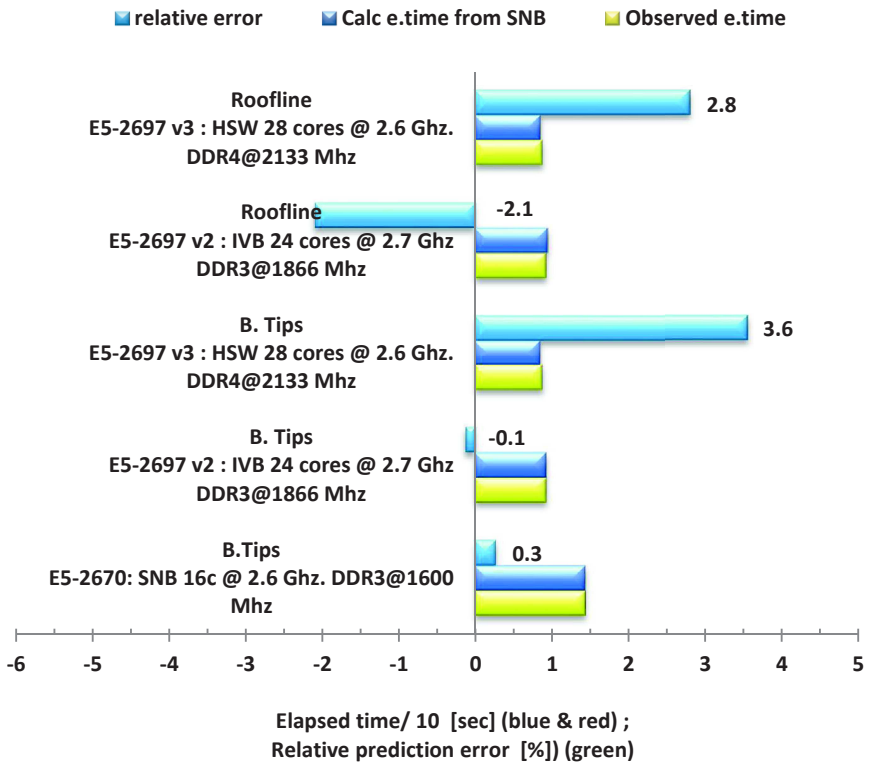
(%vecto, double runs),

-> simulating any CPU is instantaneous

Where accuracy is coming from

- Knowledge of future hdw specs.
- Knowledge of “Amdahl” scalability
- Potential affinity impact
  - COD with standart compact/scatter

- Since all specs can’t be known at time of prediction
  - Uncertainties are mandatory for all apps
  - Caches effect may help for “InBetween” apps



B. Tips on SNB E5-2670 are done from Ivy E5-2697v2: we can even predict the past !

WIP: Where to use uncertainties

$t_{\_cpu\_A}$ and $t_{\_bw\_A}$ can directly include measurement uncertainties

GF = (FP_ops/FP_inst)_theo : fixed, at least for the coming 5y

- % vecto : output of Compiler + SDE : need uncertainties since that tools are always under dev.
- * (Inst/cyc) : fixed, at least for the coming 5y
- * (Cyc/sec) : the unknown part of the CPU Frequency: need uncertainties
- * eff_amdahl: extracted from the applications: need uncertainties
- * eff_hpl : FP efficiency of the CPU derived from HPL: need uncertainties
- * nbc: cannot be sure 5y ahead: need uncertainties
- $BW_X = BW_{theo}$ * Stream efficiency: need uncertainties

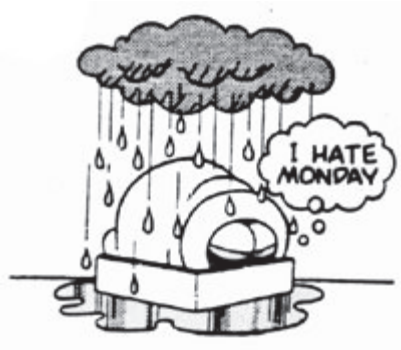
Conclusions

- High level extrapolation works fine for several arch ahead
  - Extremely quick and pragmatic way.
  - “SoL” is simple but nice for safety : validation of other techniques
  - New Roofline extrap and temporal roofline brings added values
  - “bench tips”: application level prediction within [0 : 10 %]

Next:

- Improve precision and range of apps, by including
  - Simulator data as Sniper
  - uncertainties on frequencies,
  - hits, misses , latency for all levels : caches, MCDRAM, DDR, NVM
  - Interconnection impacts for comms and io (Amdahl pitfalls)

Questions ?



Legal Disclaimers

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Legal Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.

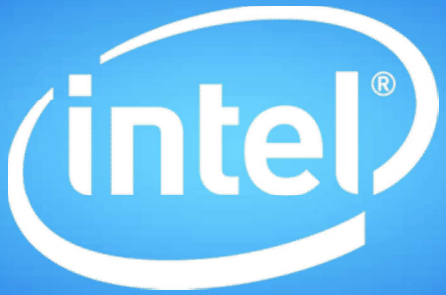
Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Estimated Results Benchmark Disclaimer:
Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Software Source Code Disclaimer:
Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



Software and Services Group | Energy Engineering Team