How performance tools are essential for designing next generation architectures

W. Jalby, C. Valensi, E. Oseret, M. Popov, M. Tribalat
ECR (Exascale Computing Research)
CEA INTEL UVSQ

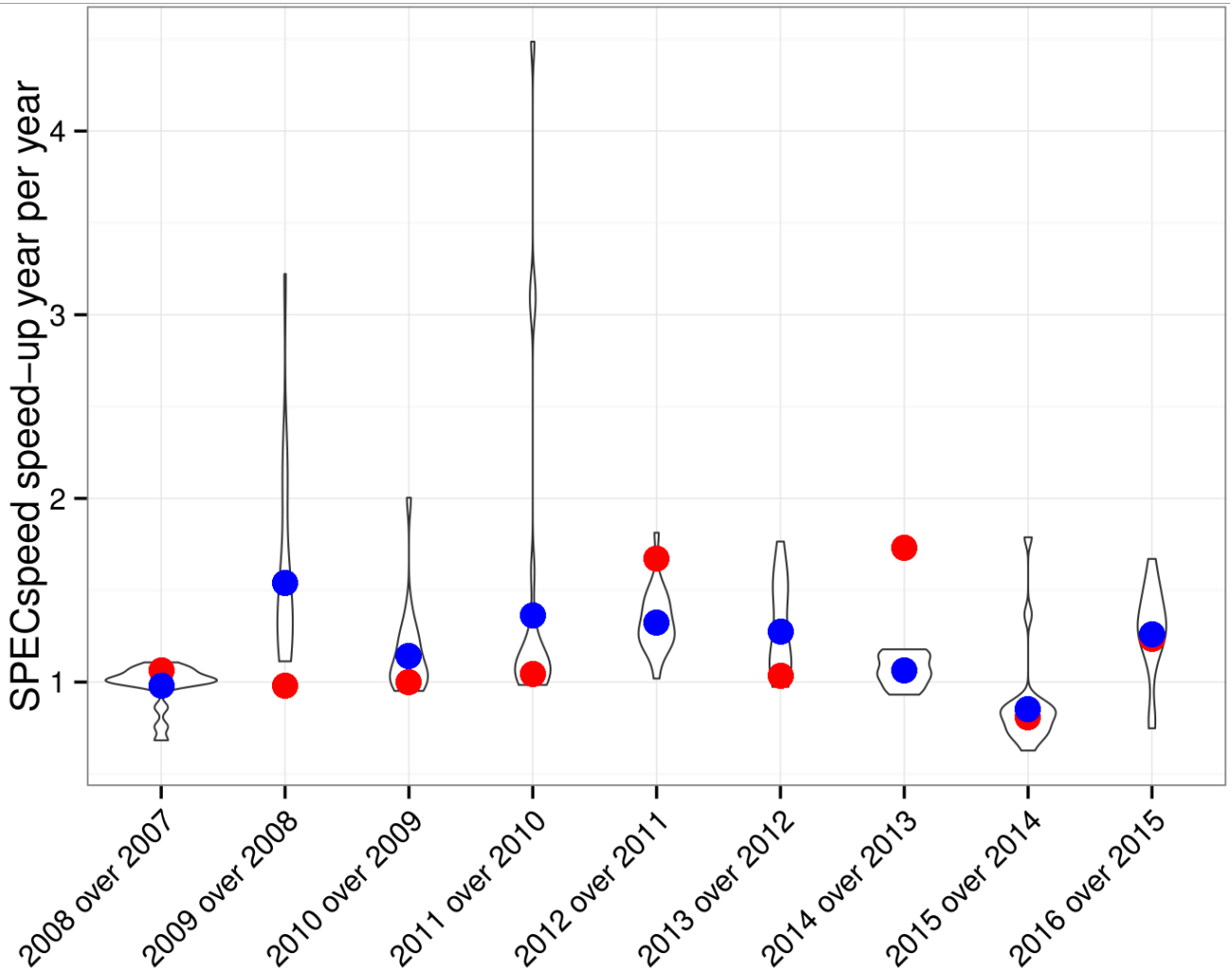
A LOOK AT THE PAST DECADE

System	CPU	Year	Core DP (Gflops)	Freq (GHz)	# Cores	L3 (MB)	RAM (GB)
Harpertown (45 nm)	X5482	2007	12.80	3.20	8	0	16
Harpertown (45 nm)	X5492	2008	13.60	3.40	8	0	16
Gainestown (45 nm)	W5590	2009	13.32	3.33	8	8	48
Westmere-EP (32 nm)	X5680	2010	13.32	3.33	12	12	48
Westmere-EP (32 nm)	X5690	2011	13.88	3.47	12	12	96
Sandy Bridge-EP (32 nm)	E5-2690	2012	23.20	2.90	16	20	64
Ivy Bridge-EP (22 nm)	E5-2690 v2	2013	24.00	3.00	20	25	128
Haswell-EP (22 nm)	E5-2690 v3	2014	41.60	2.60	24	30	256
Haswell-EP (22 nm)	E5-4650	2015	33.60	2.10	48	30	512
Broadwell-EP (14 nm)	E5-2690 v4	2016	41.60	2.60	28	35	256

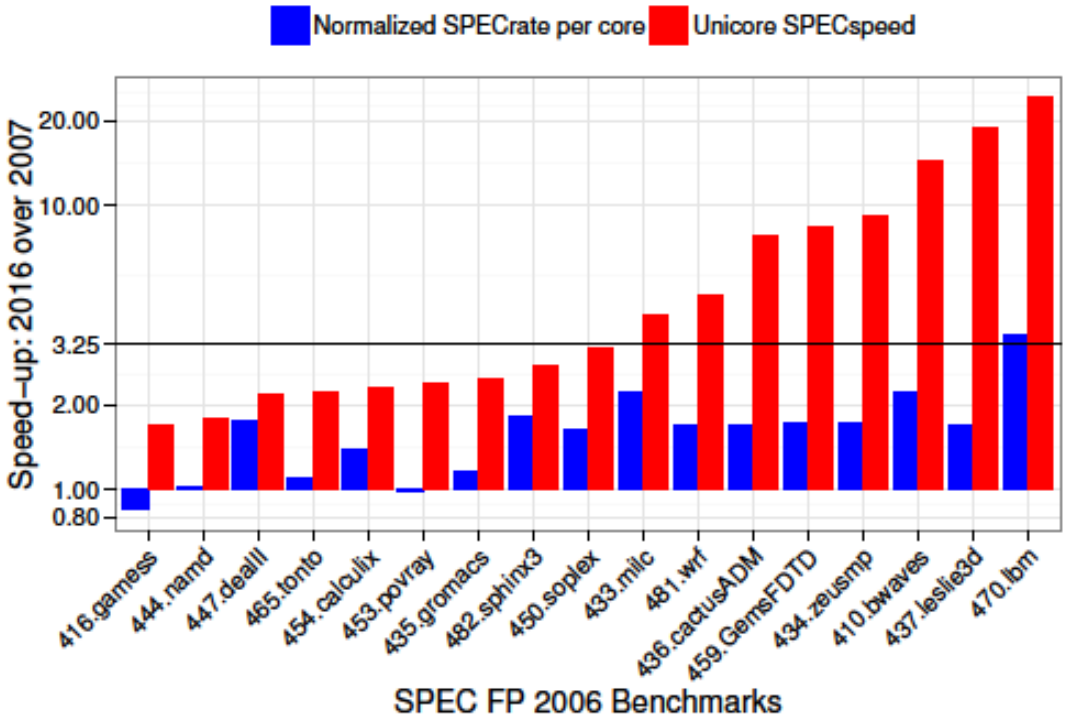
- SPEC FP 2006 : Full set of “real” scientific applications
- SPEC a well known and thorough performance measurement methodology, a large (unique ??) amount of results
- 10 INTEL reference architectures from the past decade
- Architecture/compilers vary from year to year BUT Source code are invariant

PERFORMANCE PROGRESS DURING THIS PAST DECADE

- Speedup from one year to the next
- Violin: distribution for the full set of SPECFP
- Red Dots: speedup of peak FP performance
- Blue Dots: geometric mean of Speedups
- Unicore measurements
- Baseline

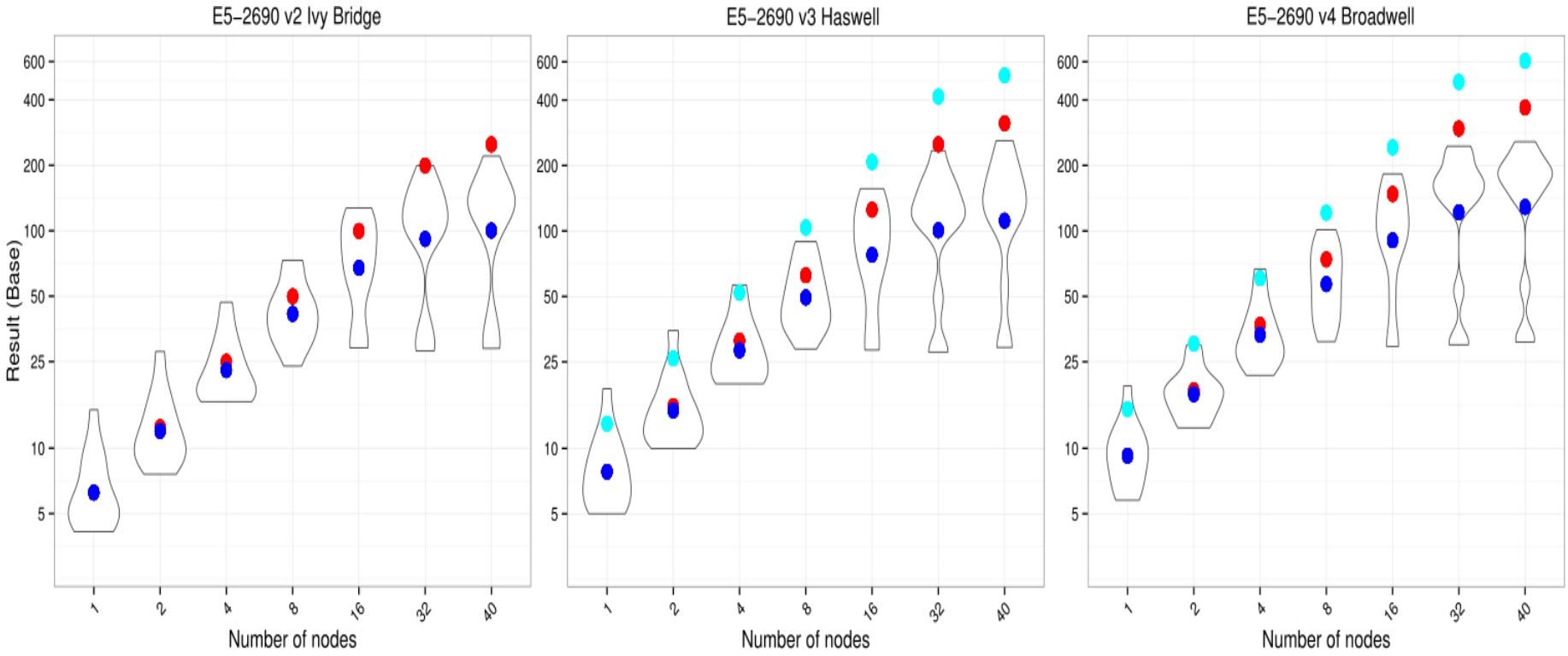


A MORE REALISTIC VIEW ON PERFORMANCE PROGRESS



- Unicore measurements (red bars) are optimistic: one core has full access to the whole memory hierarchy.
- SPECrate (blue bars) divided by number of cores is a more realistic view
- Baseline numbers

SPEC MPI PERFORMANCE PROGRESS: LAST 4 YEARS



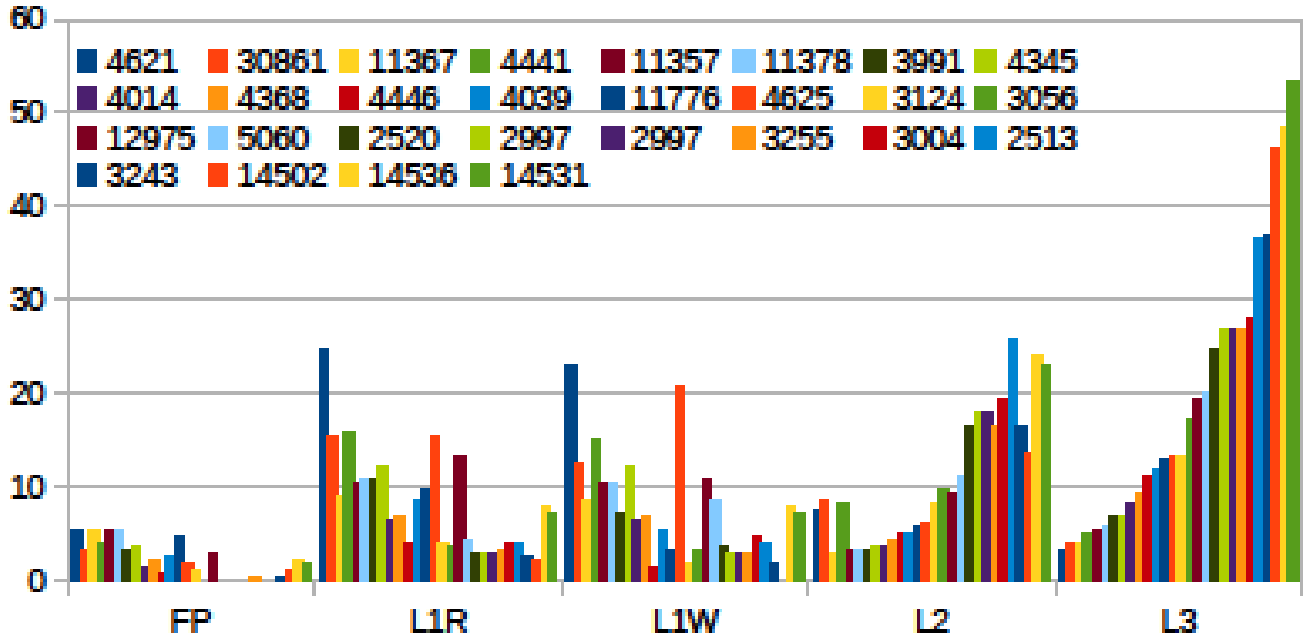
CPU	cores per node	peak (TFlops)	SPEC result	result vs peak
E5-2690 v2	20	19.2	100.26	5.22
E5-2690 v3	24	39.94	111.61	2.79
E5-2690 v4	28	46.59	128.94	2.77

GORDON BELL AWARD OVER THE LAST DECADE

Year	Machine	Cores	Tflops obtained	Tflops peak	Obtained over Peak (%)	Computations
2007	BlueGene/L	131K	0,11	0,28	39	Micron-Scale Atomistic Simulation of Kelvin-Helmholtz Instability
2008	Cray XT4	31K	0,2	0,26	77	Simulations of disorder effects in high-Tc superconductors
2009	Cray XT5	147K	1,03	1,36	76	Ab initio computation of free energies
2010	Cray XT5-HE	200K	0,7	2,3	30	Direct numerical simulation of blood flow
2011	K computer	442K	3,08	7,07	44	First-principles calculations of electron states of a silicon nanowire
2012	K computer	663K	4,45	10,6	42	Astrophysical N -Body Simulation
2013	Sequoia	1.6M	11	20,1	55	Cloud Cavitation Collapse
2014	Anton 2	NA	NA	NA	NA	Molecular dynamics
2015	Sequoia	1.6M	0,69	20,1	3	Implicit Solver for Complex PDEs
2016	Sunway TaihuLight	10.56M	7,85	125	6	Fully-Implicit Solver for Nonhydrostatic Atmospheric Dynamics

- Very impressive results: high efficiency except for the last 2 years
- Very different results from SPEC evolution
- Codes have been fully customized: using (??) top of the line performance evaluation tools ☺
- Are these codes more than Proof of Concept ?? Impact on standard apps ??

RESOURCE USAGE: YALES2 LOOPPS



- 28 loops among the hottest for YALES2: combustion CFD application used in academia and industry
- FP (Floating Point Units), L1R (L1 Read access), L1 (L1 Write access), L2 access, L3 access
- Per “node” (FP, L1R, L1W, L2, L3) and then per loop, percentage of peak bandwidth used: “saturation ratio”. Loops sorted by increasing L3 usage

A FEW LESSONS

SPEC numbers over the last decade are completely depressing.

Hardware resource usage is poorly controlled. Low saturation ratio means bad investment also useless power consumption.

Several possible choices:

1. Improve Compiler Autotuning: OK but will provide at best 10 to 20% perf improvement
2. Fully rewrite applications (cf Gordon Bell): OK but very costly and not practical for most application fields

A FEW DIRECTIONS

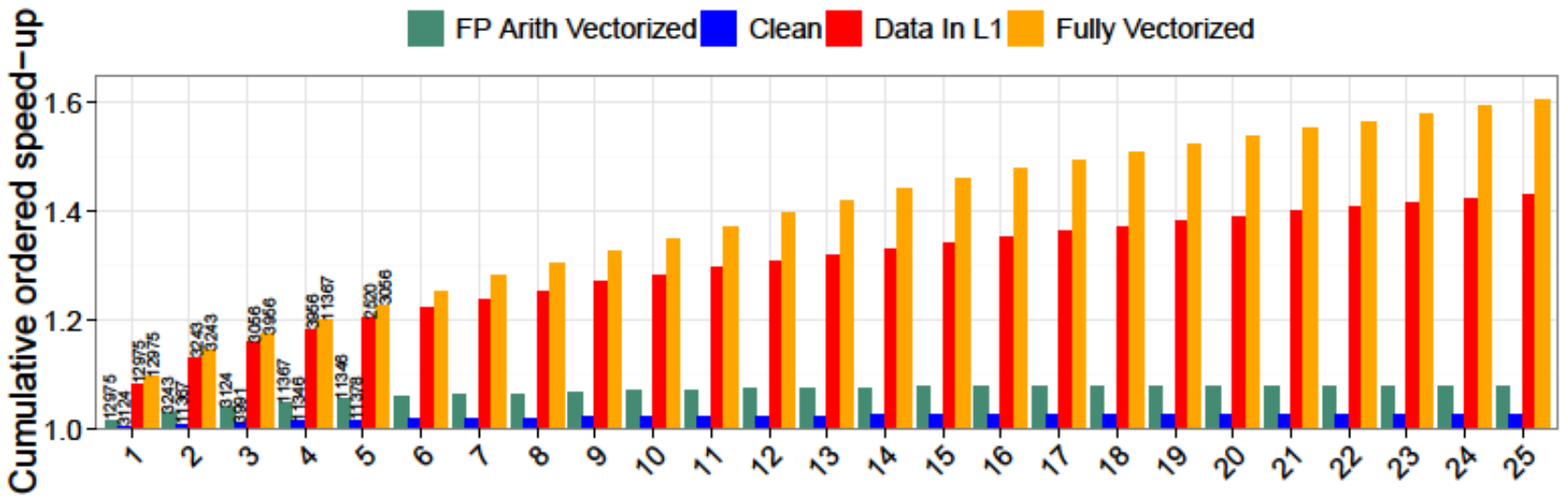
More powerful approaches

- Use performance tools to guide application restructuring. Stop giving detailed diagnostics that a standard user cannot understand or lead to the wrong path. Instead of pointing to problems, suggest and evaluate potential solutions. THINK AS A DOCTOR ☺
- Use performance tools to conduct application performance characterization and drive hardware/software co design

FOUR OPTIMIZATIONS TO BE EVALUATED

- "Clean Code": checks compiler deficiencies. Supress non FP instructions and evaluate performance potential gains.
- "FP Arith Vectorization " : evaluate performance gains due to partial vectorization
- "Fully Vectorized": evaluate performance due to full vectorization (Loads/Stores + arithmetic)
- "L1 Blocking": evaluate performance impact of perfect blocking (all data in L1).

Ecuptae nobis et vel hillitas dolupta turiat.



"ONE View" optimization report on Yales 2 CFD Combustion code.
X axis: the most fruitful loops to be optimized. Loop identifiers given for the 5 most fruitful loops.
Y axis: cumulative performance gain

A QUICK LOOK AT ENERGY

- 2 Haswell architecture: client and server
- Different behavior across instructions
- Different behavior between two "similar" architectures
- Power varies during computation

Energy (nJ)/ HW node	HSWE3 (3.5GHz)	HSWEP (2.4GHz)
FE (nJ/Inst)	0.11	0.08
INT (nJ/Inst)	0.14	0.13
ADD (SS/SD) 32/64 (nJ/Inst)	0.33	0.33
ADD (PS/PD) 128 (nJ/Inst)	0.33	0.33
ADD (PS/PD) 256 (nJ/Inst)	0.60	0.45
MUL (SS/SD) 32/64 (nJ/Inst)	0.16	0.13
MUL (PS/PD) 128 (nJ/Inst)	0.30	0.15
MUL (PS/PD) 256 (nJ/Inst)	0.33	0.25
DIV (SS) 32 (nJ/Inst)	3.76	4.19
DIV (SD) 64 (nJ/Inst)	5.59	5.46
DIV (PS) 128 (nJ/Inst)	5.21	4.80
DIV (PD) 128 (nJ/Inst)	6.32	6.15
DIV (PS) 256 (nJ/Inst)	11.83	11.20
DIV (PD) 256 (nJ/Inst)	14.09	17.05
FMA (PS/PD) 256 (nJ/Inst)	0.70	0.63
LD/L1 (SS/SD) 32/64 (nJ/Inst)	0.30	0.15
LD/L1 (PS/PD) 128 (nJ/Inst)	0.33	0.18
LD/L1 (PS/PD) 256 (nJ/Inst)	0.45	0.34
ST/L1 (SS/SD) 32/64 (nJ/Inst)	0.69	0.69
ST/L1 (PS/PD) 128 (nJ/Inst)	0.69	0.69
ST/L1 (PS/PD) 256 (nJ/Inst)	0.78	0.69
L2 (Read/Write) (nJ/64 bytes)	2.96	2.12
L3 (Read/Write) (nJ/64 bytes)	4.59	4.70

Table 2: Energy coefficients expressed in nano-Joule

ENERGY: NEW AND MANDATORY CHALLENGES

C: Capacity. Performance metric (Flops per cycle, Transactions per cycle, etc….)

E: Energy consumed by a computation

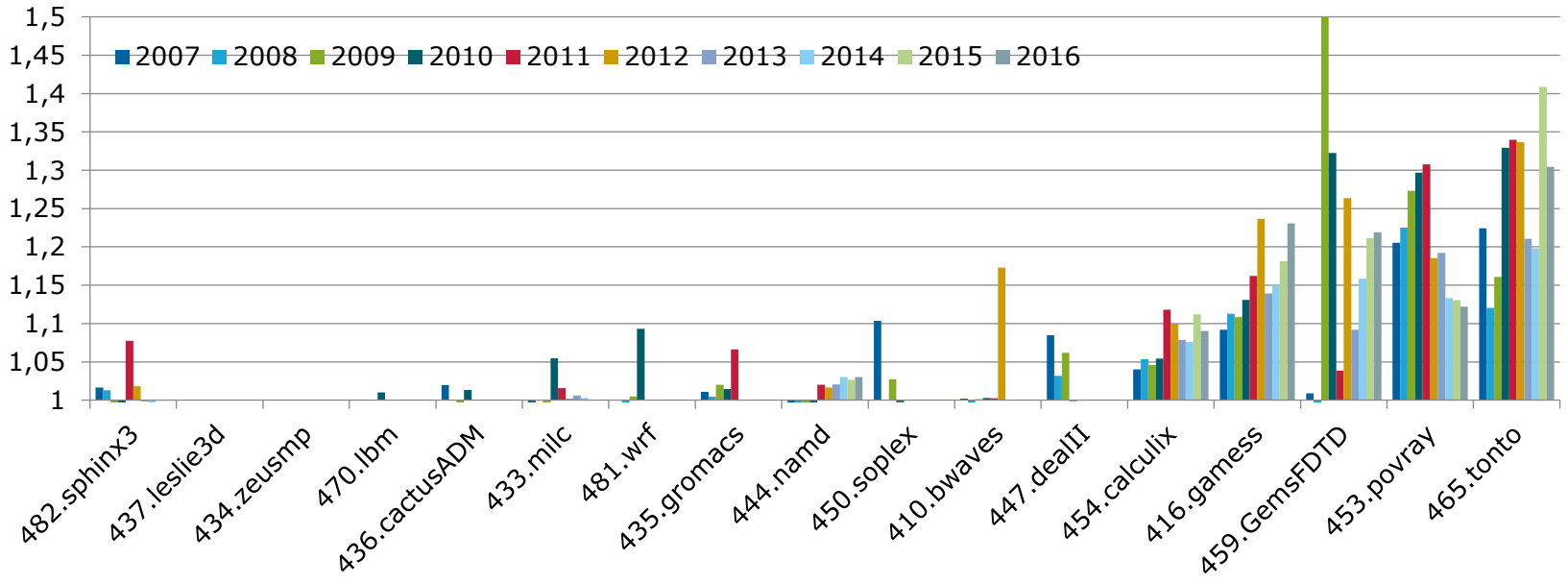
- Only maximizing C is no longer a correct objective because it might lead to unacceptable power/energy costs
- Only minimizing E is not a correct objective either because it leads to low capacities.
- Race to Halt strategies are also too short minded because they essentially assume constant power

PERFORMANCE AND ENERGY MODELS ARE NEEDED

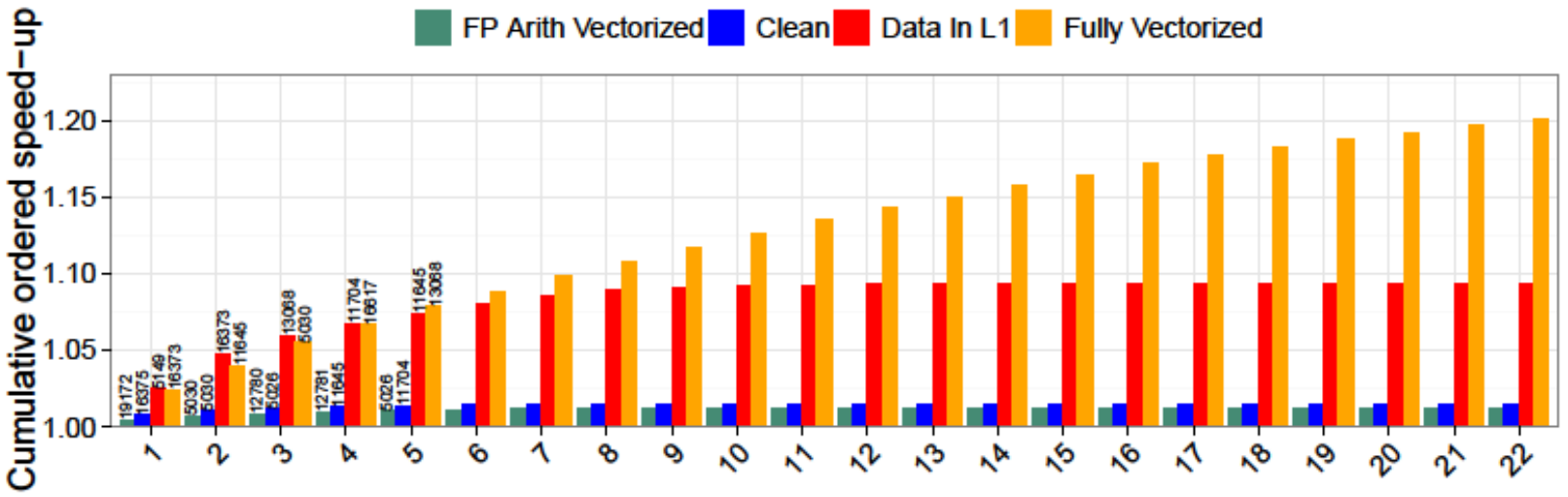
Real Objectives

- Maximising a Quality metric (C, C/E) under constraints (Constant Power, Constant Capacity)
- To be correctly addressed, such objectives needs performance models which will use as an essential component "measurements"
- Performance tools needs to add predictive power to predict power behavior, performance behavior.

Ecuptae nobis et vel hillitas dolupta turiat.



Ecuptae nobis et vel hillitas dolupta turiat.



Thank You