INFN - Padova

michele.michelotto at pd.infn.it

Worker Node per HEP – Technology update

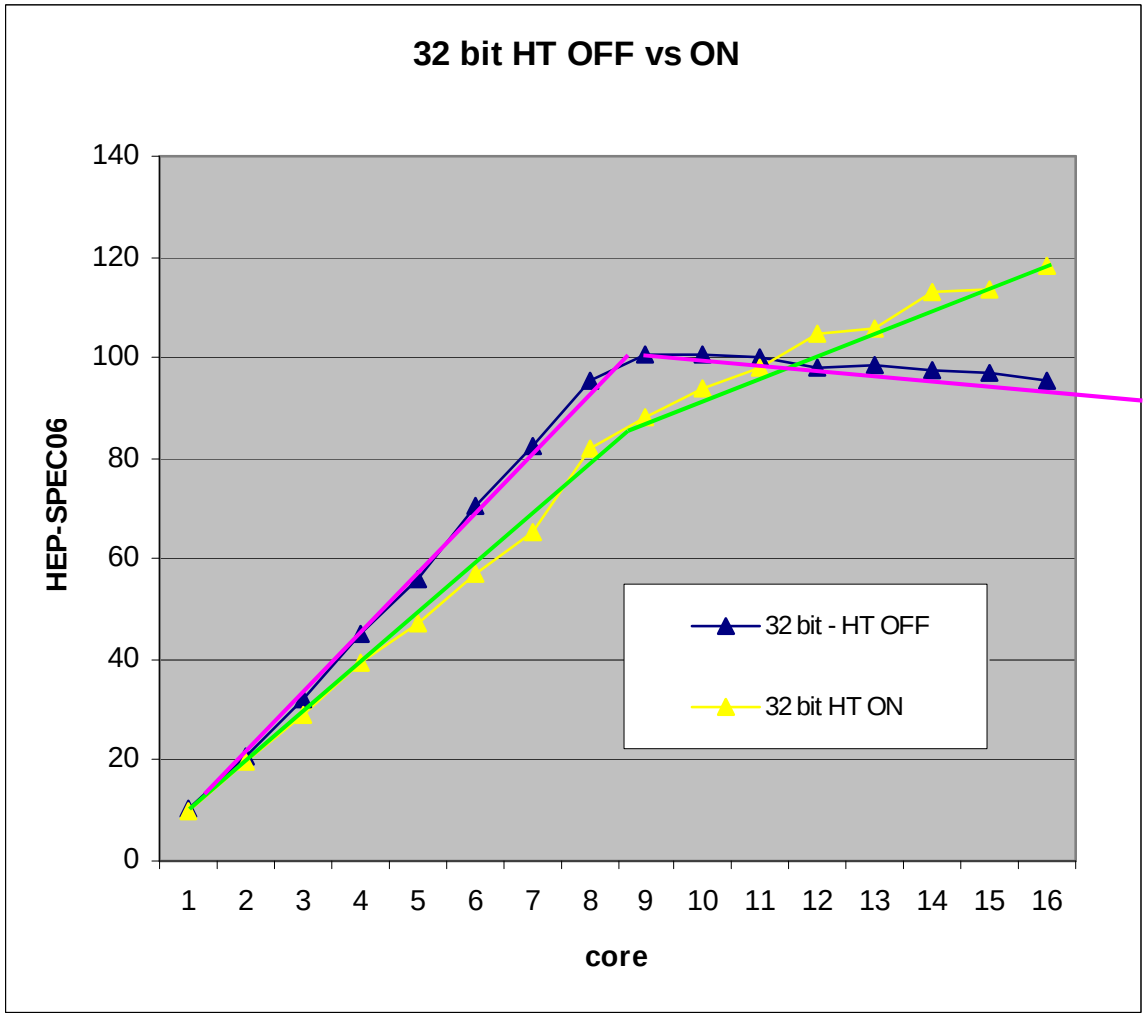
Processor Available

- Magny Cours and Westmere now available
- I'll show some preliminary result

CPU	Westmere Gulftown	Nehalem Beckton	Magny Cours	Lisbon
process	32 nm	45 nm	45 nm	45 nm
est GHz at launch	3.33++	2.26++	2.2	2.8
Cores & caches	6 cores 12 MB L3	8 cores 24 MB L3	12 cores 12 MB L3	6 cores 6 MB L3
Net DDR3 ch / socket	3	8	4	2
market	Mid-to-high end	Ultra high end	High end	Mid-to-high end
Perf per core est avg	1 x	0.7 x	0.55 x	0.75x

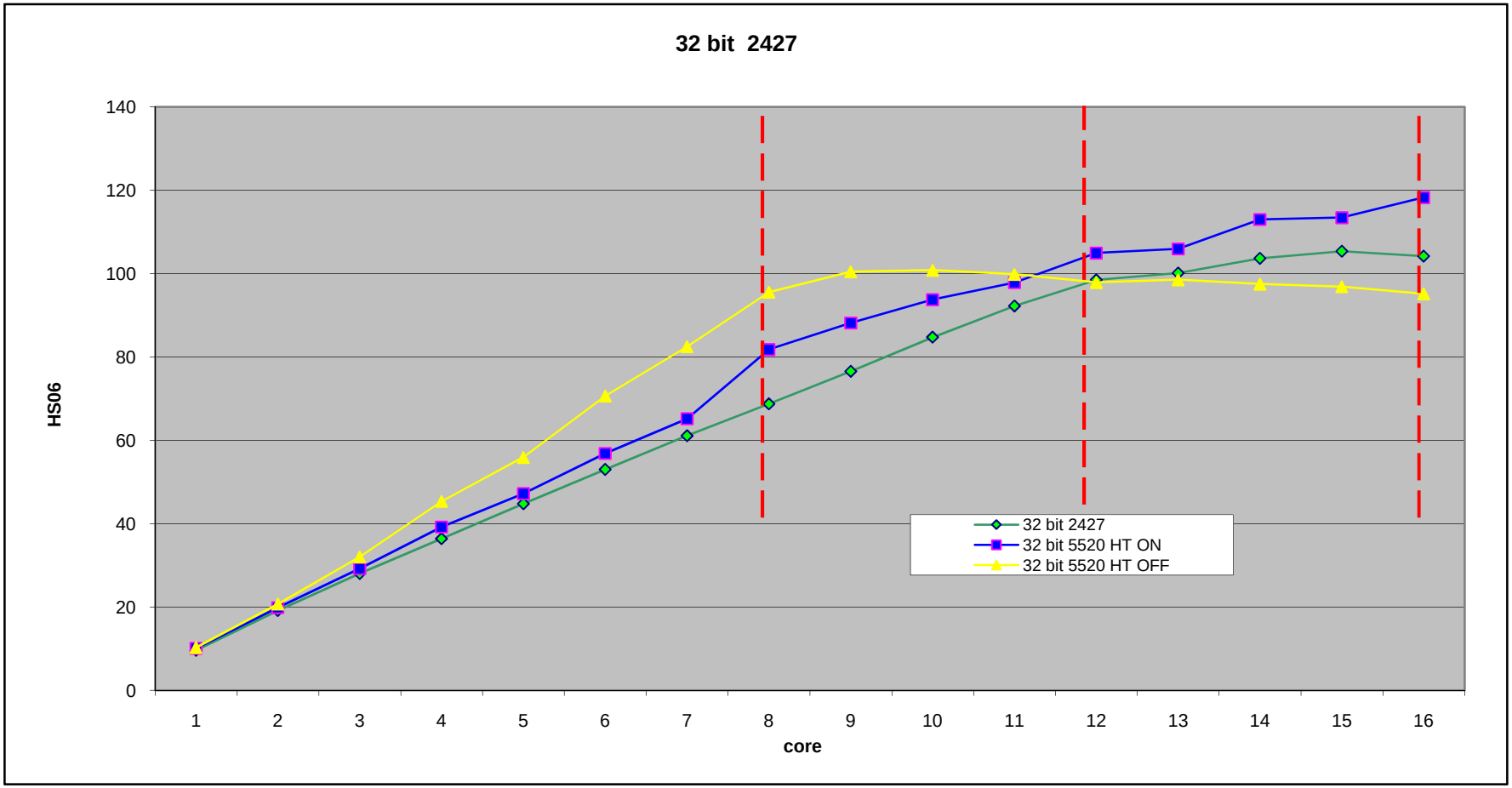
Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. Martin Golding

5520 32 bit - HT ON/OFF



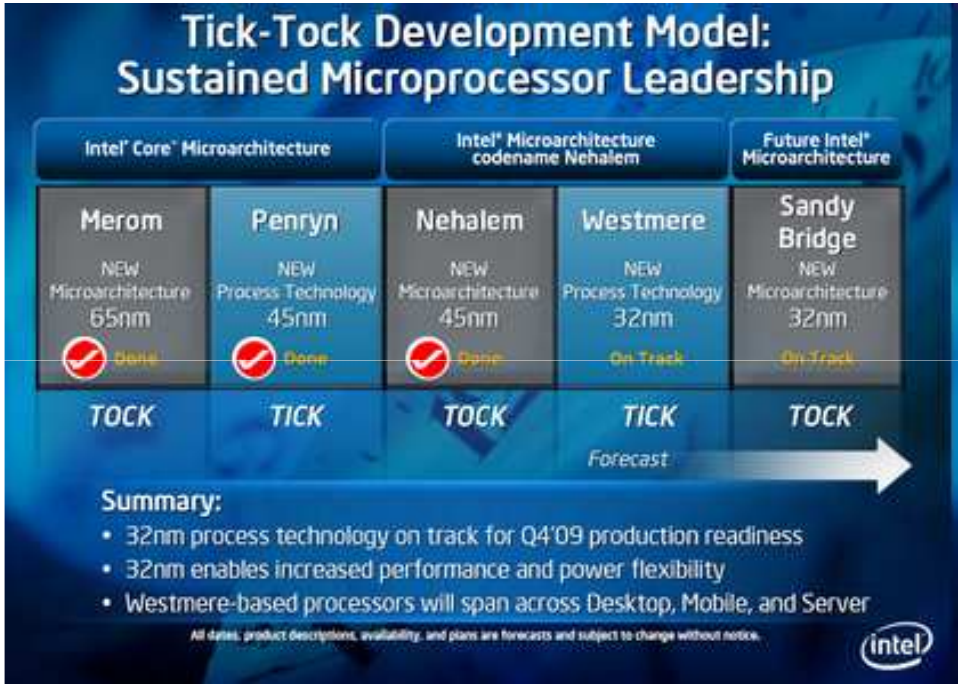
HT ON:81.81
HT OFF: 95.96
HT OFF is better up to 11concurrent run

5520(2266) vs 2427(2200)



There are two major products that come out of Berkeley: LSD an UNIX. We don't believe this to be a coincidence. *(J. S. Anderson)*

Tick Tock



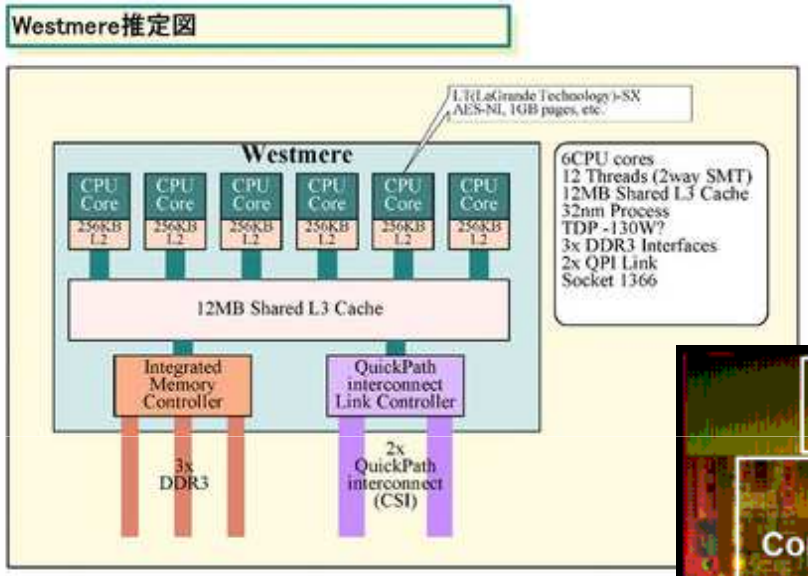
Programmer - an organism that turns coffee into software. Author Unknown

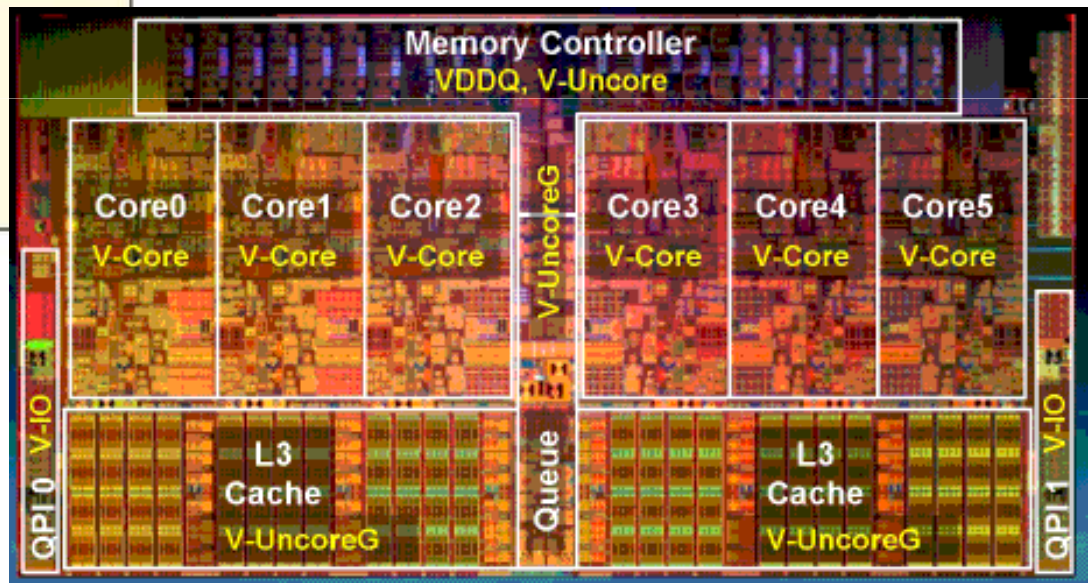
Westmere DP

- 16 March 2010 first release of Xeon DP on Westmere 32nm

model	Cores/threads	L3 cache	Clock with Turbo Mode Off	TDP
X5680	6/12	12 MB	3.33 GHz	130 W
X5670	6/12	12 MB	2.93 GHz	95 W
X5660	6/12	12 MB	2.80 GHz	95 W
X5650	6/12	12 MB	2.66 GHz	95 W
L5640	6/12	12 MB	2.26 GHz	60 W
X5677	4/8	12 MB	3.46 GHz	130 W
X5667	4/8	12 MB	3.06 GHz	95 W
E5640	4/8	12 MB	2.66 GHz	80 W
E5630	4/8	12 MB	2.53 GHz	80 W
E5620	4/8	12 MB	2.40 GHz	80 W
L5630	4/8	12 MB	2.26 GHz	40 W

Westmere 56xx





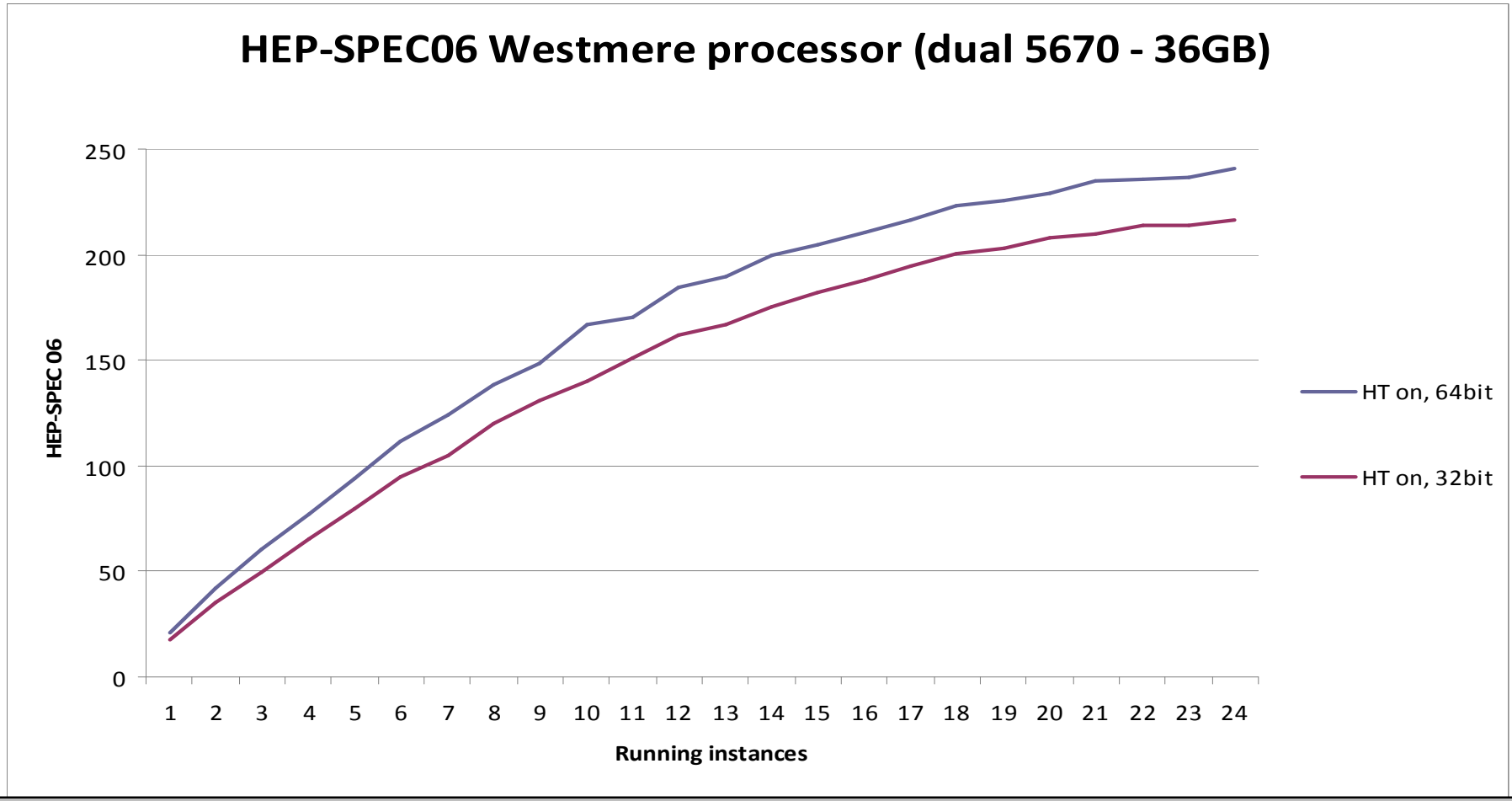
Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the universe trying to build bigger and better idiots. So far, the universe is winning.

Thanks

- Thanks to Andrea Chierici from the Farming Group at CNAF – INFN Tier1 in Bologna for measurements on Intel Westmere
- Thanks to AMD for the access to a 2x6174 in Munich
- Thanks to Alberto Crescente for scripting and measuraments submissions
- HEPMARK Experiment is financed by CSN5 of INFN

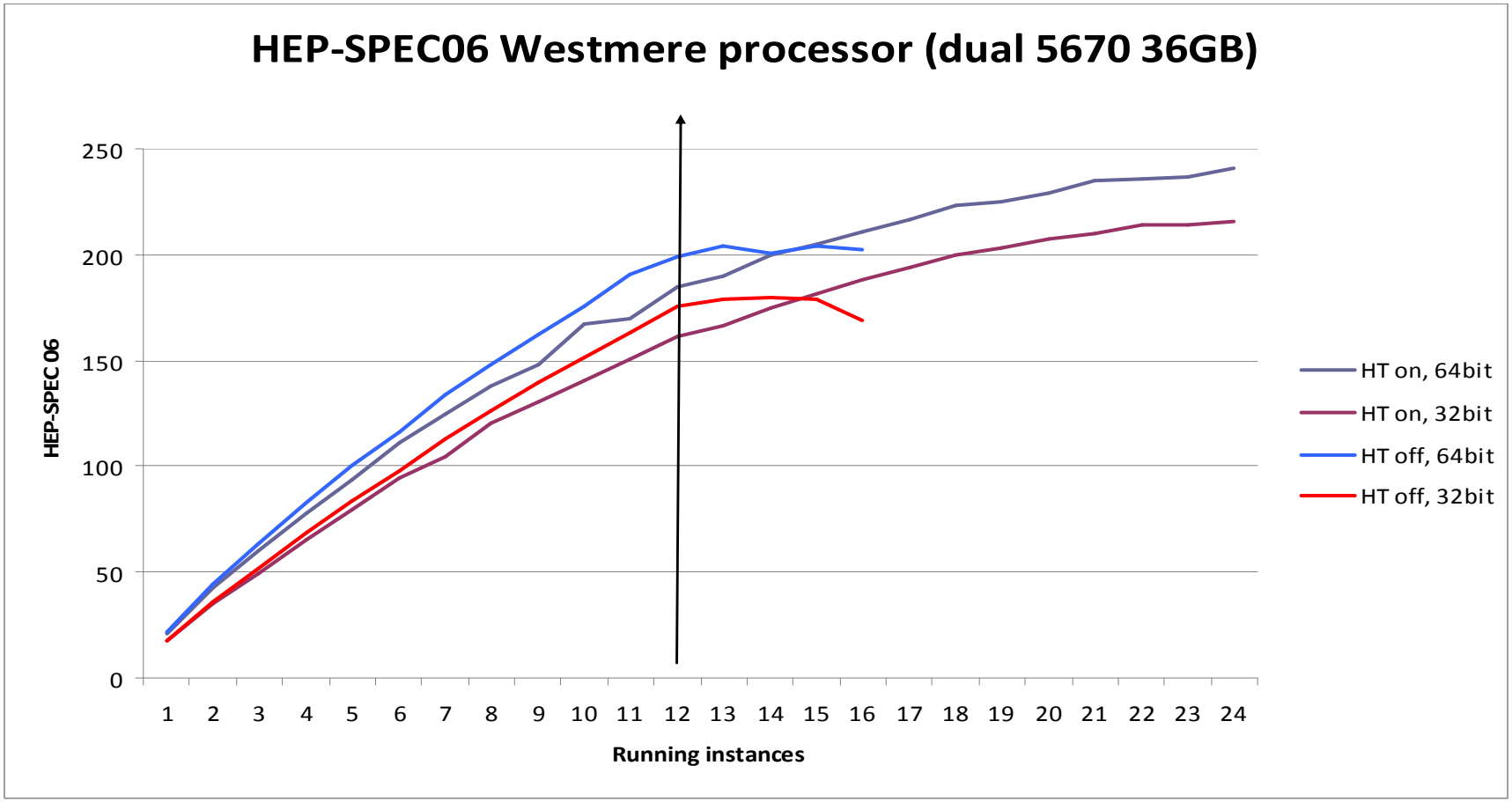
“I am not out to destroy Microsoft, that would be a completely unintended side effect.” – *Linus Torvalds*

5670 64 bit vs 32 bit



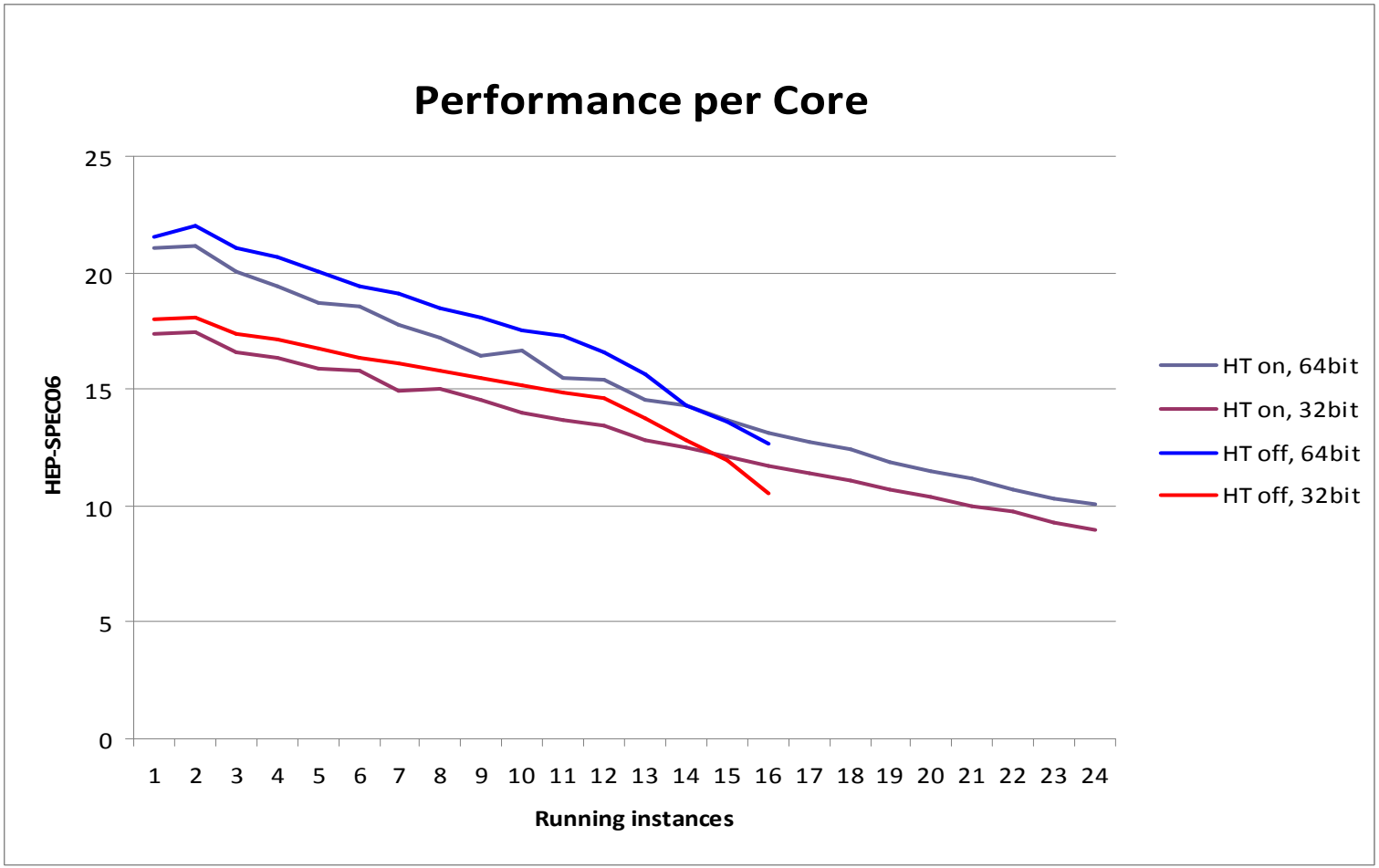
Better train people and risk they leave – than do nothing and risk they stay. Anonymous Technical Trainer

5670 HT-ON or OFF



"Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems."– *J. Zawinski*

HS06 per core



"Perl: The only language that looks the same before and after RSA encryption. –

Keith Bostic

AMD Magny-Cours 61xx

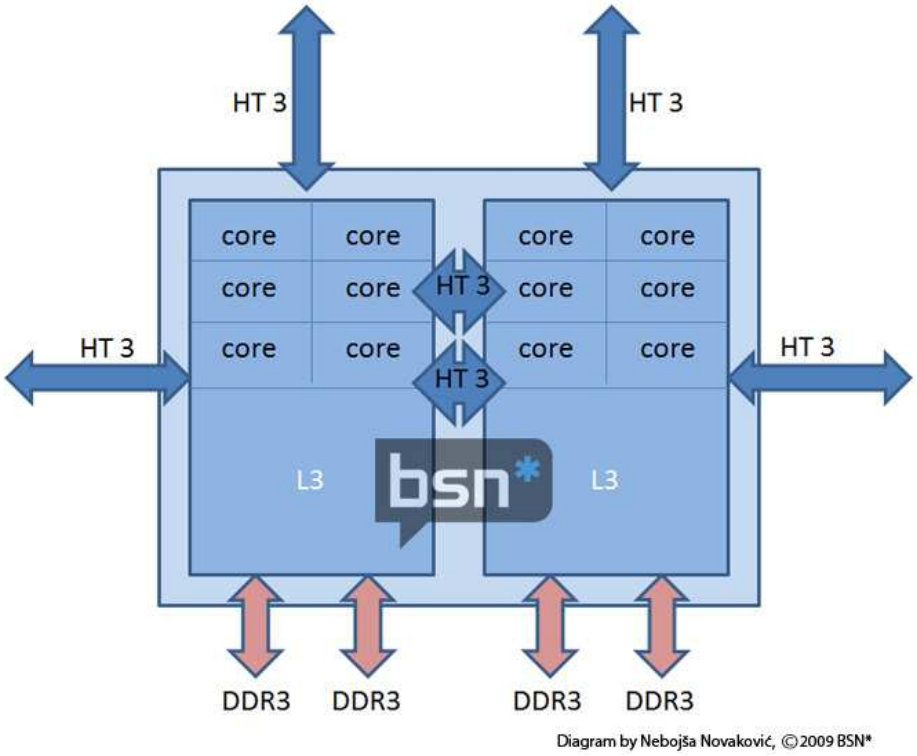
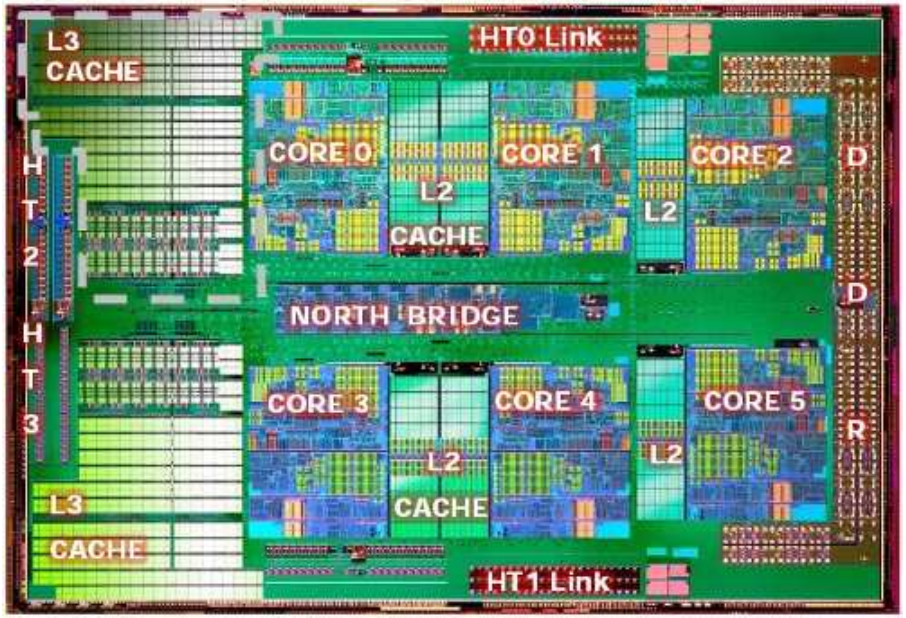
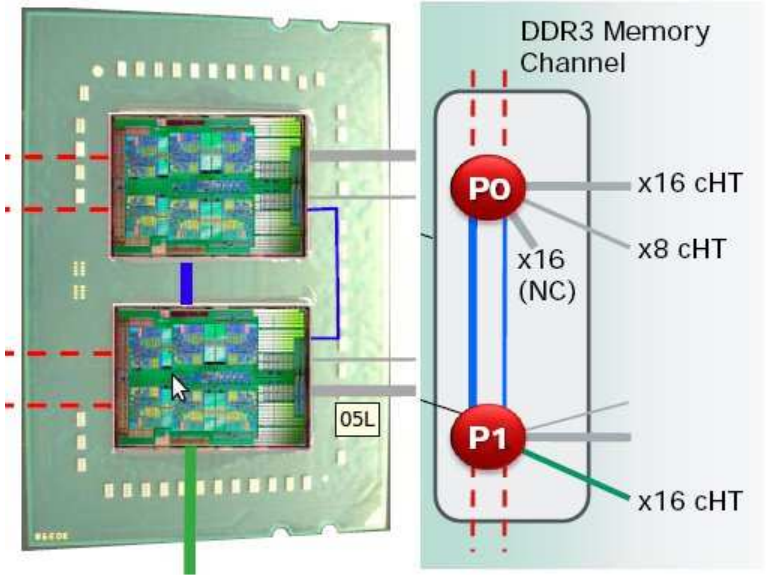


Diagram by Nebojša Novaković, ©2009 BSN*



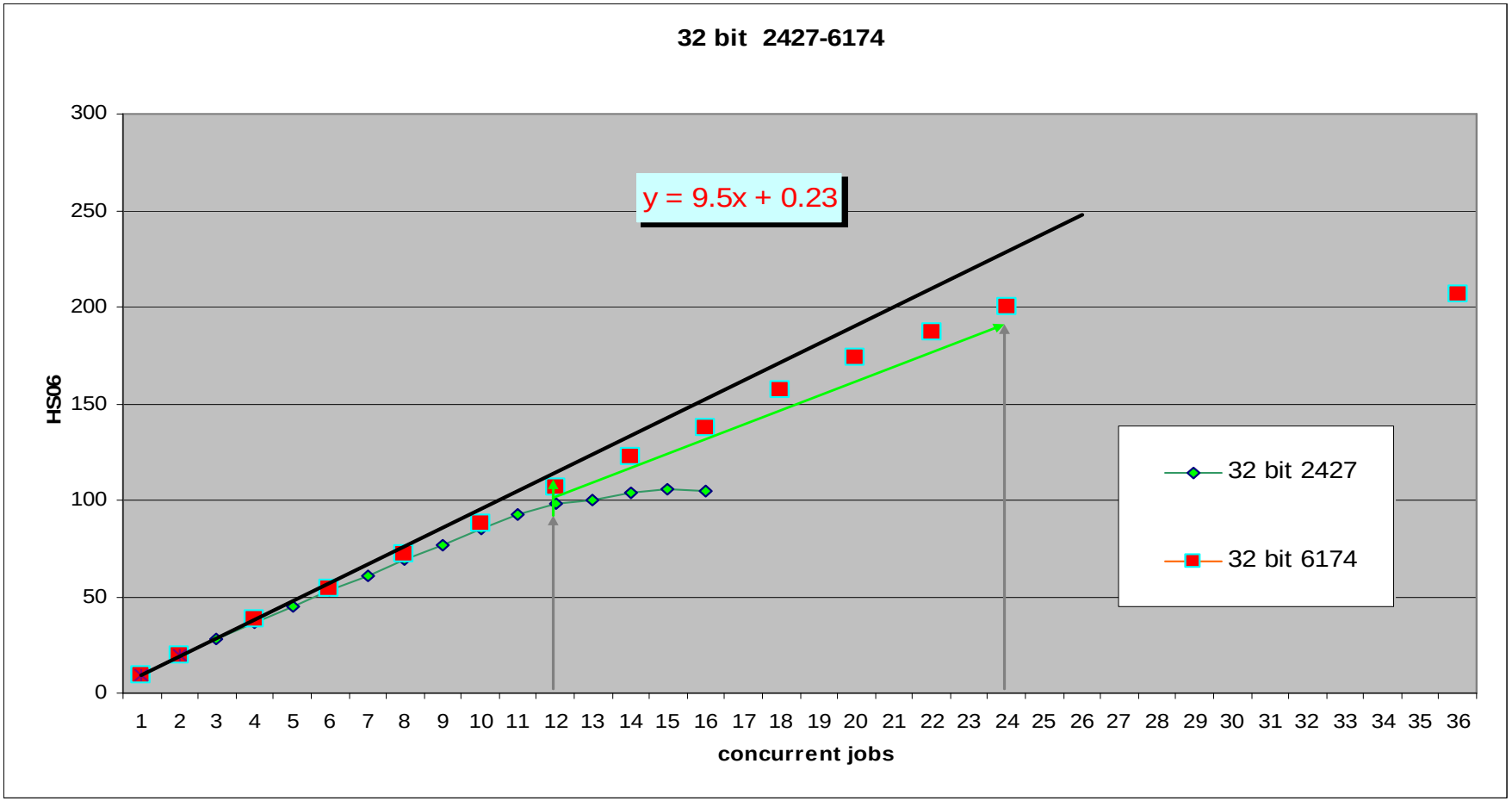


Magny-Cours

model	Cores/threads	L3 cache	Clock	ACP
Xeon 5670	6/12	12 MB	2.93 GHz	95 W
6124 HE	8/8	12 MB	1.8 GHz	65 W
6128	8/8	12 MB	1.8 GHz	80 W
6128 HE	8/8	12 MB	2.0 GHz	65 W
6134	8/8	12 MB	2.3 GHz	80 W
6136	12/12	12 MB	2.4 GHz	80 W
6164 HE	12/12	12 MB	1.7 GHz	65 W
6168	12/12	12 MB	1.9 GHz	80 W
6172	12/12	12 MB	2.1 GHz	80 W
6174	12/12	12 MB	2.2 GHz	80 W
6176 SE	12/12	12 MB	2.3 GHz	105 W

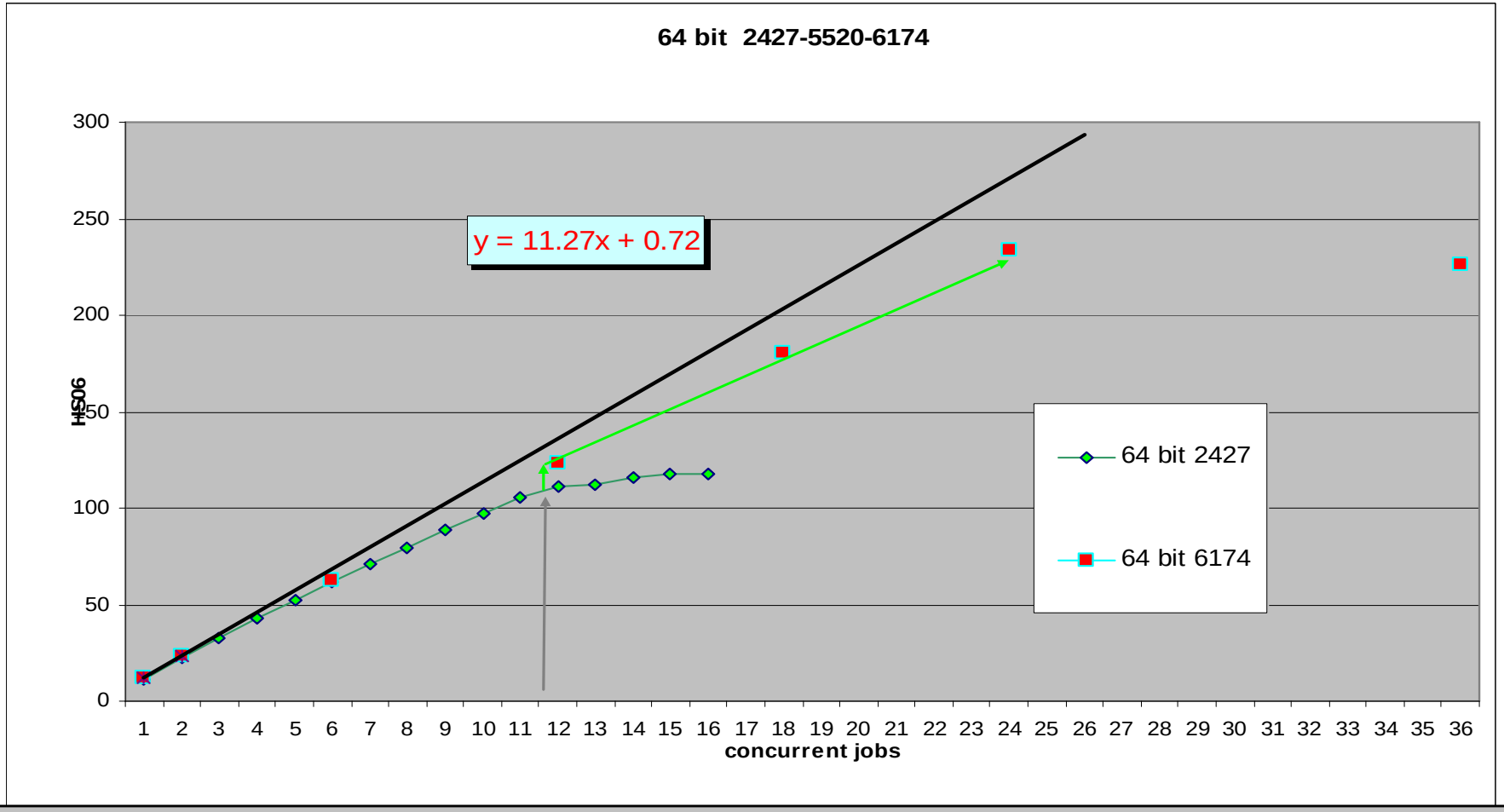
A known bug is better than an unknown feature.

6174 (2.2 GHz) 64 GB 32bit



It is easier to measure something than to understand what you have measured.

6174 64 bit



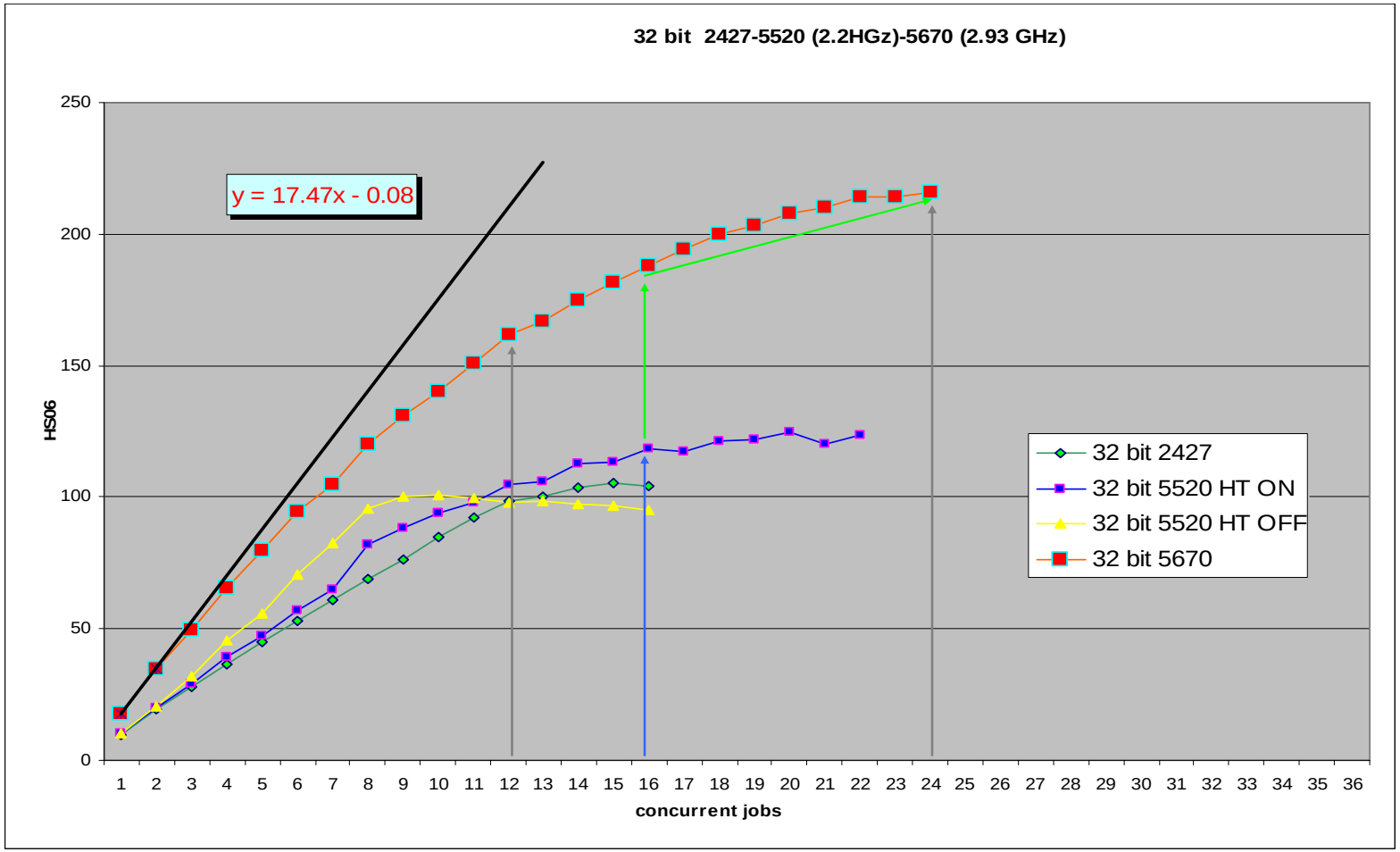
"Most of you are familiar with the virtues of a programmer. There are three, of course: laziness, impatience, and hubris."

From 2009 to 2010

- Intel moved Nehalem to Westmere
  - From 45 nm to 32 nm
  - From 4-core/8 logical cpu to 6core/12cpu
- Amd moved from Instambul to Magny-Cours
  - From 6 cores to 12 cores

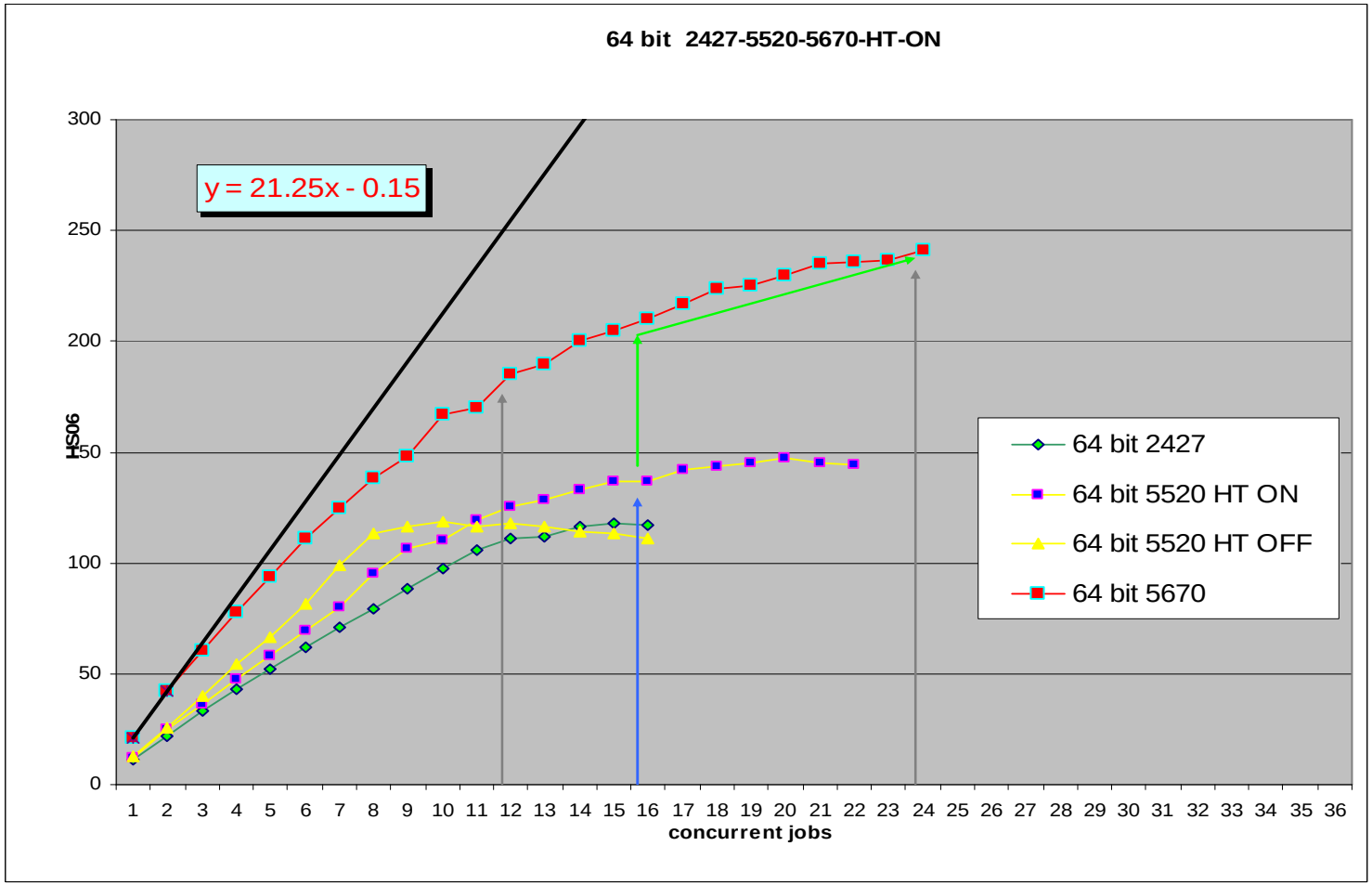
"There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies." *(C.A.R. Hoare)*

32 bit - 5520 (2.2) - 5670 (2.93)



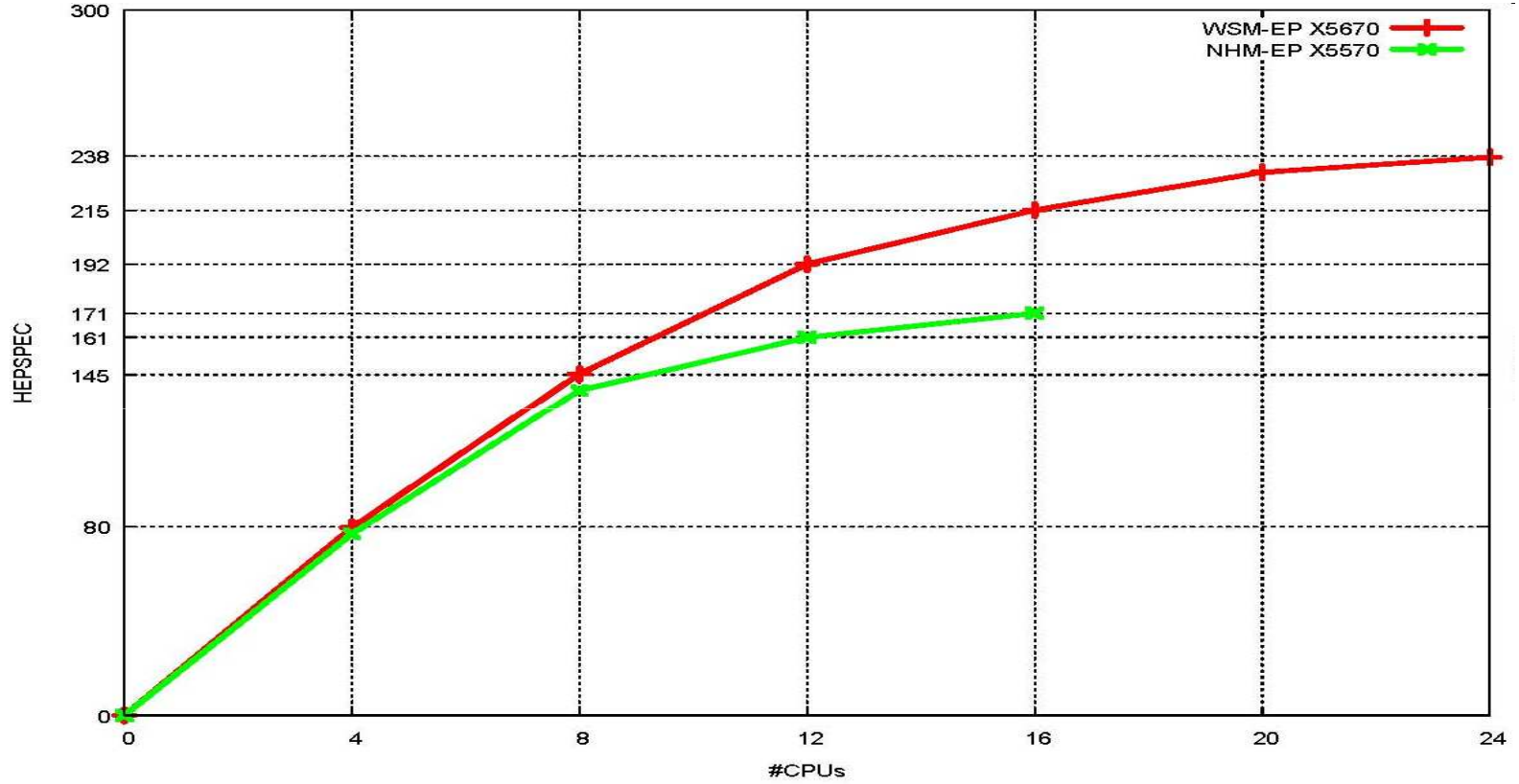
"In C++ it's harder to shoot yourself in the foot, but when you do, you blow off your whole leg. *(Bjarne Stroustrup)*

64bit - 5520 (2.2) - 5670 (2.93)



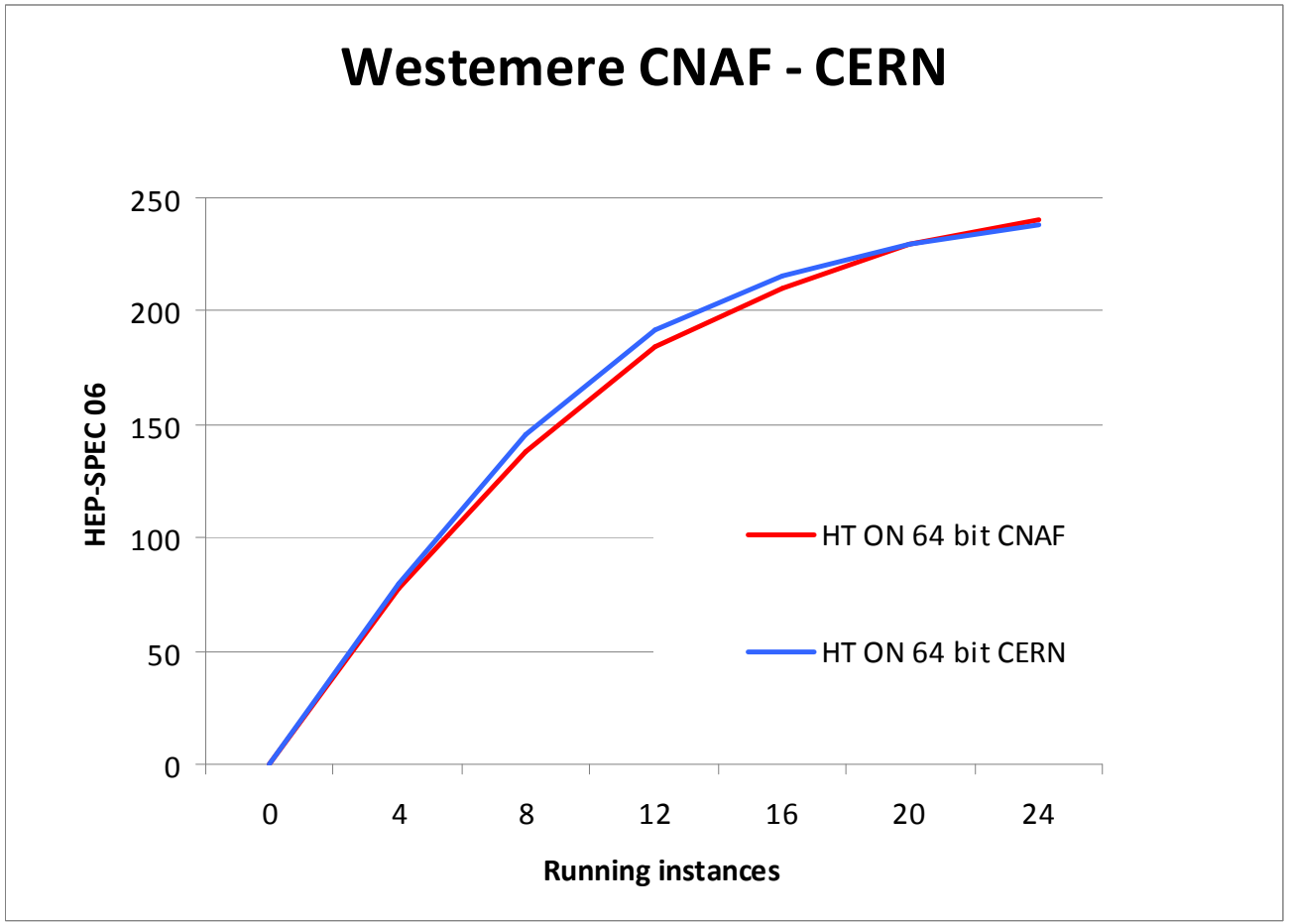
“If Java had true garbage collection, most programs would delete themselves upon execution.”

Westmere Nehalem same clock - CERN



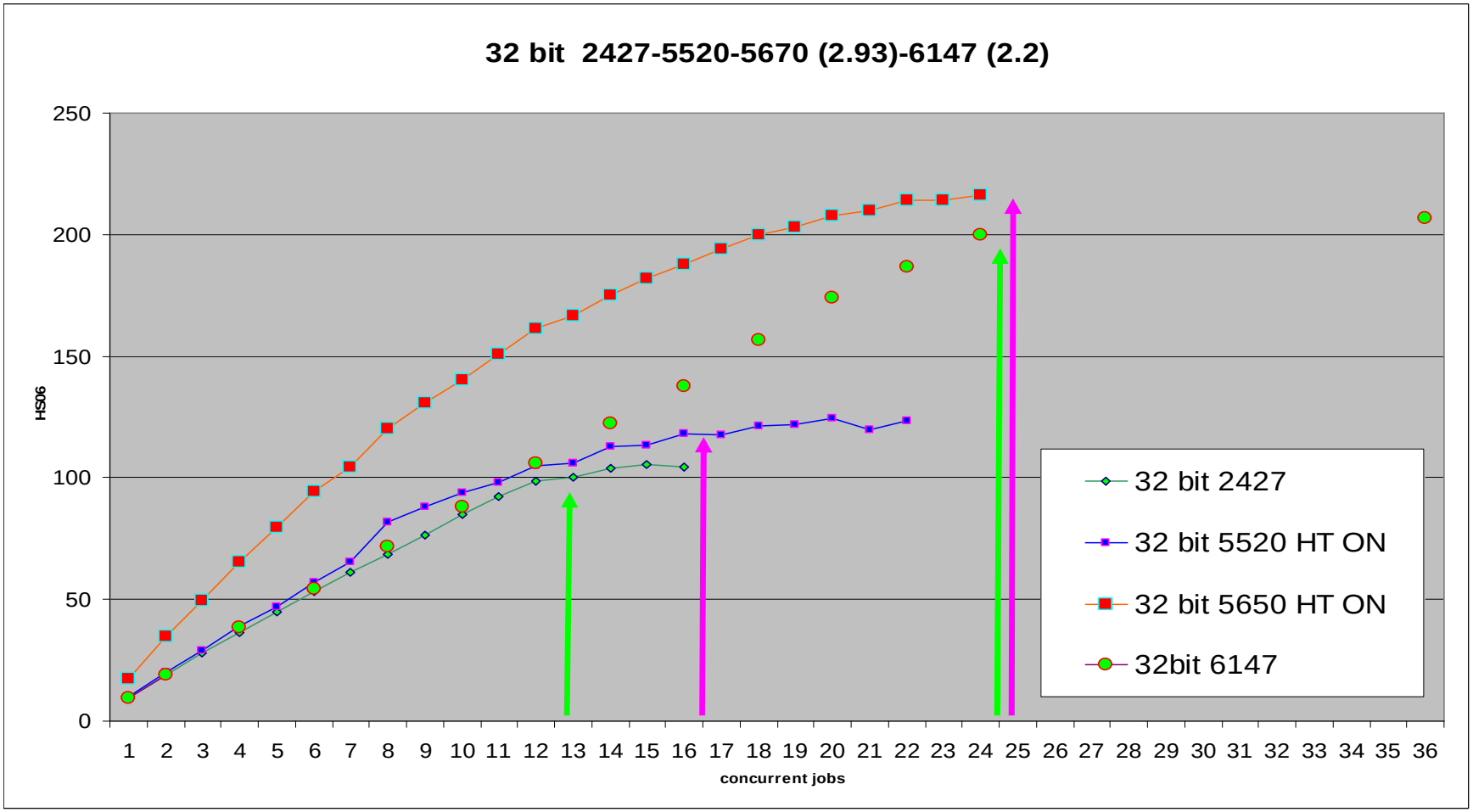
"Debugging is twice as hard as writing the code in the firstplace. Therefore, if you write the code as cleverly as possible, you are–by definition–not smart enough to debug it."*(Brian Kernighan)*

CERN vs CNAF



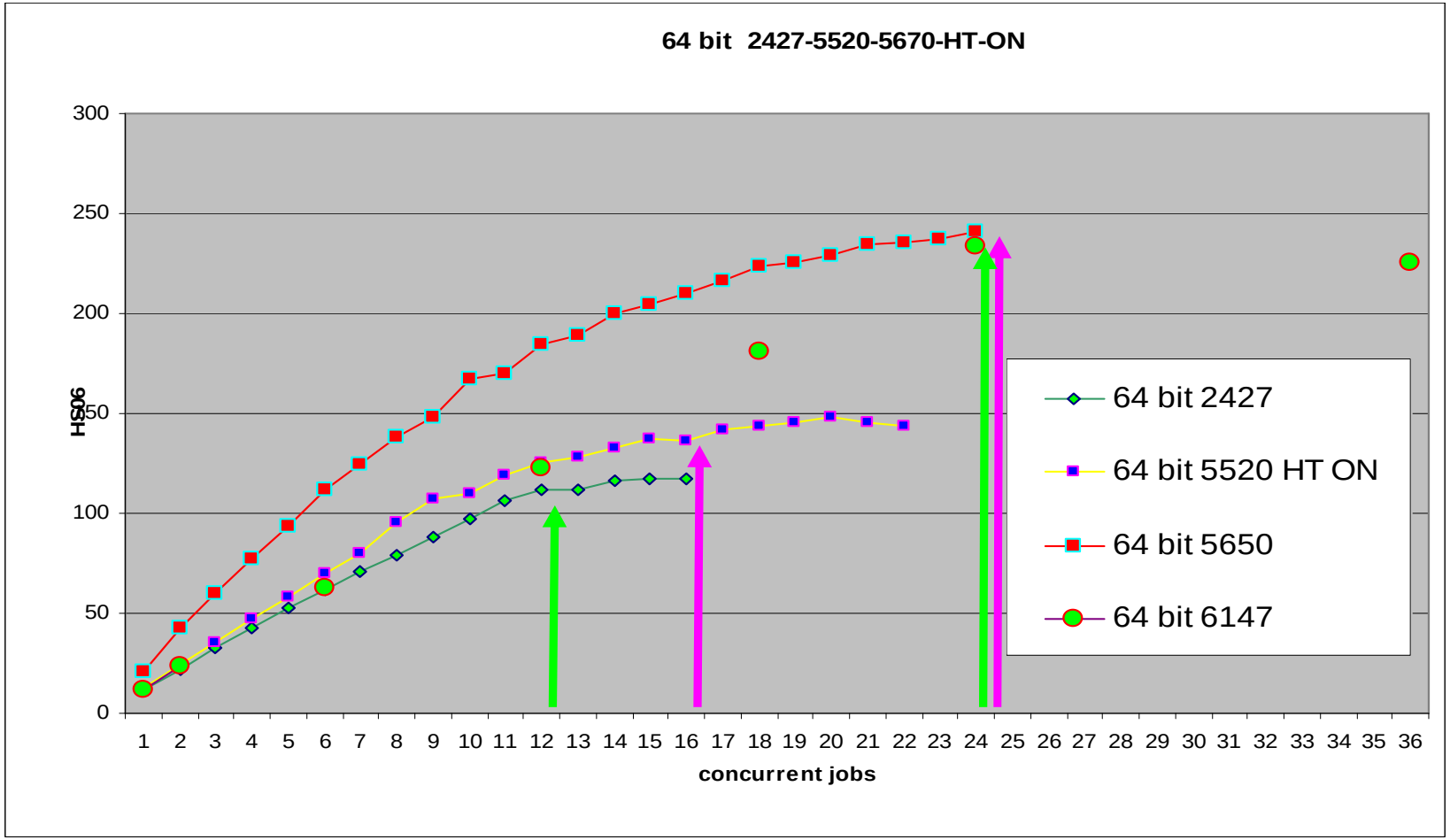
"I don't care if it works on your machine! We are not shipping your machine!" (*Vidiu Platon*)

From 12-16 cpu to 24 cpu



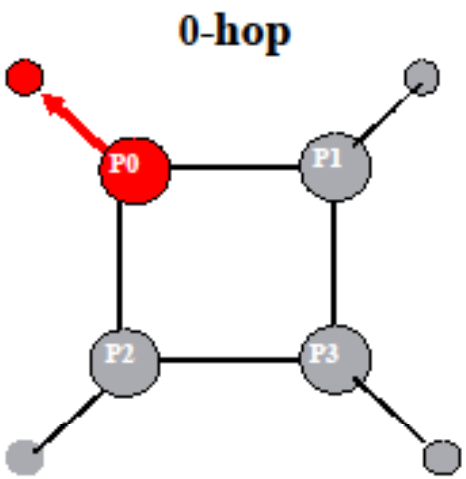
"A computer lets you make more mistakes faster than any invention in human history–with the possible exceptions of handguns and tequila." *(Mitch Radcliffe)*

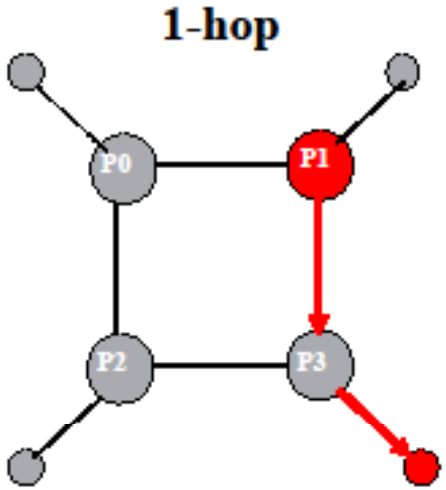
From 12-16 cpu to 24 cpu

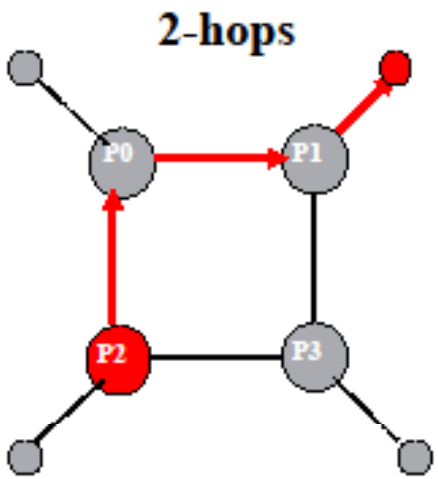


"Passwords are like underwear: you don't let people see it, you should change it very often, and you shouldn't share it with strangers."
– Chris Pirillo

NUMA







- 0 Hop: Local memory access
- 1 Hop: Remote 1 memory access
- 2 Hop: Remote 2 memory access

"Physics is the universe's operating system."
– Steven R Garman

NUMA and Linux

- If the OS is **not** NUMA aware then
  - Memory allocation is random mixing local and remote
  - Performance is unpredictable and below optimal
- NUMA awareness is the OS means that
  - The OS is able to distinguish and select the local memory pools of each processor
  - The OS provides control to the user regarding memory pool selecection and process binding
- BIOS switch allows to interleave memory across processors, good for multithread applications with huge memory requirements, but HT links becomes bottlenecks

Memory Latency	Interleaving OFF	Interleaving ON
LOCAL	~ 100 ns	Average for 4P system is ~ 120 ns
1 hop	~ 115 ns	
2 hops	~ 150 ns	

NUMA and SL5

- SL5 kernel should be NUMA aware

```
[root@hep-wn-004 ~]# numactl --hardware
available: 2 nodes (0-1)
node 0 size: 16145 MB
node 0 free: 12935 MB
node 1 size: 16160 MB
node 1 free: 11659 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
[root@hep-wn-004 ~]# numactl --show
policy: default
preferred node: current
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11
cpubind: 0 1
nodebind: 0 1
membind: 0 1
```

How to bind

- **Normal run**

runspec --config=cern --nobuild --noreportable $BENCHMARK

- **Bind Node and memory**

```
if (( $i < 9 )); then
                k=0;
        else
                k=1;
        fi
        echo "Node bind: cpunodebind=$k membind=$k"
        numactl --cpunodebind=$k --membind=$k runspec --
    config=cern --nobuild --noreportable $BENCHMARK
```

“It’s hardware that makes a machine fast. It’s software that makes a fast machine slow.” – *Craig Bruce*

How to bind

- **Normal run**

runspec --config=cern --nobuild --noreportable $BENCHMARK

- **Bind to physical Node and memory**

```
for i in `seq $COUNT`;
        do
            j=`expr $i - 1`
            if (( $i < 9 )); then
                    k=0;
            else
                    k=1;
            fi
            echo "Physical Bind: physcpubind=$j membind=$k"
            numactl --physcpubind=$j --membind=$k runspec --
    config=cern --nobuild --noreportable $BENCHMARK
```

“If the code and the comments do not match, possibly both are incorrect.”– *Norm Schryer*

Deliberate wrong binding

- **Normal run**

```
runspec --config=cern --nobuild --noreportable $BENCHMARK
```

- **Bind Node to REMOTE memory**

```
for i in `seq $COUNT`;
        do
           j=`expr $i - 1`
           if (( $i < 9 )); then
                  k=0;
                  z=1;
           else
                  k=1;
                  z=0;
           fi
           echo "Node UnBind: cpunodebind=$k membind=$z"
           numactl --cpunodebind=$k --membind=$z runspec --config=cern --
    nobuild --noreportable $BENCHMARK
```

When debugging, novices insert corrective code; experts remove defective code. – Richard Pattis

Results on HEPSPEC

32/64	64	64	32	32
Threads	12	12	12	16
Processor	AMD 2374	Xeon 5520	AMD 2374	Xeon 5520
Standard run	100.00%	100.00%	100.00%	100.00%
Node to Mem binding	90.49%	92.71%	90.82%	98.68%
Core to Mem binding	101.37%	100.32%	103.20%	99.12%
Antibinding	71.90%	77.61%	81.90%	86.34%

Theory is when you know something, but it doesn't work. Practice is when something works, but you don't know why. Programmers combine theory and practice: Nothing works and they don't know why.

Effect on numastat

	node0	node1
numa_hit	132002	84038
numa_miss	0	0
numa_foreign	0	0
interleave_hit	4821	4915
local_node	131118	77434
other_node	884	6604

Benchmark S2k6 all_cpp 64bit (12 thread node bind)

	node0	node1
numa_hit	55478019	51197936
numa_miss	0	0
numa_foreign	0	0
interleave_hit	9131	9219
local_node	55471736	51133252
other_node	6283	64684

Node memory bind

	Node0 before	Node1 Before	Node 0 after	Node 1 after
numa_hit	55,478,019	51,197,936	+69,579,633	+36,699,269
numa_miss	0	0	+0	+0
numa_foreign	0	0	+0	+0
interleave_hit	9,131	9,219	+5,206	+5,244
local_node	55,471,736	51,133,252	+69,542,836	+36,697,287
other_node	6,283	64,684	+36,797	+1,982

- local_node+other_node = numa_hit
- local_node >> other_node
- never seen a numa_miss numa_foreign
- Why local_node for node0 and node1 so different?

wrong bind

	Node0 before	Node1 Before	Node 0 after	Node 1 after
numa_hit	194,398,361	124,919,692	+38,375,457	+67,941,685
numa_miss	0	0	+0	+0
numa_foreign	0	0	+0	+0
interleave_hit	18,082	18,278	+4,501	+4,522
local_node	178,766,891	124,823,723	+7,202,636	+5,649,346
other_node	15,631,470	95,969	+31,172,821	+62,292,339

- local_node << other_node
- Local0 and local1 different but now inverted

Questions?

numastat

- **numa_hit** is the number of allocations where an allocation was intended for that node and succeeded there.
- **numa_miss** shows how often an allocation was intended for this node, but ended up on another node due to low memory.
- **numa_foreign** is the number of allocations that were intended for another node, but ended up on this node. Each *numa_foreign* event has a *numa_miss* on another node.
- **interleave_hit** is the number of interleave policy allocations that were intended for a specific node and succeeded there.
- **local_node** is incremented when a process running on the node allocated memory on the same node.
- **other_node** is incremented when a process running on another node allocated memory on that node.