ecProf meets the High Resolution IFS Forecast model at ECMWF

Deborah Salmond* and Sami Saarinen**
*Research Department
**Computer Department
ECMWF

17th HPC Workshop in Meteorology
26 Oct 2016

Agenda

- Upgrades at ECMWF
  - HPC configurations
  - Model resolution for Operational forecasts
- Profiling → Optimisation
  - Introduction to ecProf
  - Optimisation using ecProf profiles for IFS

Agenda

- Upgrades at ECMWF
  - HPC configurations
  - Model resolution for Operational forecasts

- Profiling → Optimisation
  - Introduction to ecProf
  - Optimisation using ecProf profiles for IFS

ECMWF's HPC upgrade in 2016

Date of move of operational forecast	**Phase1** **Cray XC-30** Sept 2014	**Phase2** **Cray XC-40** June 2016
Processor	Intel Xeon E5-2697v2 'IvyBridge'	Intel Xeon E5-2695v4 'Broadwell'
Clock Speed	2.7 GHz	2.1 GHz
Switch	Cray Aries	Cray Aries
Nodes	3505*2	3610*2
Cores per Node	24	36
Cores	84,120*2	129,960*2
Peak per Node (Tflops)	0.512	1.209
Peak bi-directional B/W per Node (GB/s)	14	14
Memory per Node (GB)	64	128

ECMWF's HPC upgrade in 2016

Date of move of operational forecast	Cray XC-30 Sept 2014	Cray XC-40 June 2016	Cray KNL Test cluster
Processor	Intel Xeon E5-2697v2 'IvyBridge'	Intel Xeon E5-2695v4 'Broadwell'	Intel XeonPhi 7210 'KNL'
Clock Speed	2.7 GHz	2.1 GHz	1.3 GHz
Switch	Cray Aries	Cray Aries	Cray Aries
Nodes	3505*2	3610*2	32
Cores per Node	24	36	64
Cores	84,120*2	129,960*2	2048
Peak per Node (Tflops)	0.512	1.209	2.661
Peak bi-directional B/W per Node (GB/s)	14	14	14
Memory per Node (GB)	64	128	96 + 16 MCDRAM

Migration from IvyBridge to Broadwell

- June 2016: Operational forecast switched from Ivybridge to Broadwell
- Run in ‘IvyBridge compatibility mode’ on same number of nodes.
  - Bit-identical Broadwell (18 threads) vs. IvyBridge (12 threads)
- HRES Operational runs are scheduled on 350 Nodes in so called ‘electrical group’ which gives less variation in runtimes
- Research Department experiments (run at lower resolution) are run on fewer nodes on Broadwell and make full use of extra memory
- November 2016 : Next Operational Cycle 43r1 run in Broadwell mode

ECMWF's resolution upgrade in March 2016
HRES: TL1279 → TCo1279

Spectral Truncation	TL1279	TCo1279
Horizontal Grid-pts	$2x10^6$	$6x10^6$
Grid spacing	16km	9km
Vertical Levels	137	137
Timestep	600 Seconds	450 Seconds
Floating-point ops	$12x10^{15}$	$30x10^{15}$
MPI Comms	150 TB	436 TB
SL Halo-width	18	26
Nodes used for operations	120	350
Sustained Tflops	4	15
Wall time (research expt)	3000 Seconds	2000 Seconds

IFS & HRES and ENS forecast characteristics

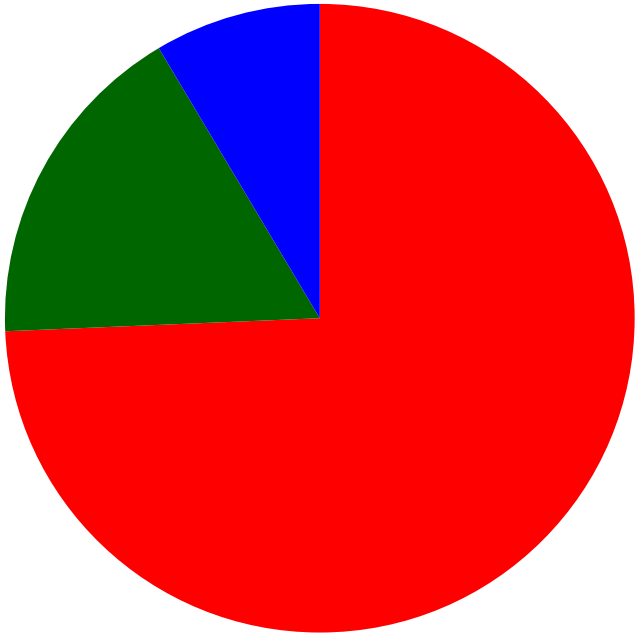
- IFS-Arpège code shared with Météo-France
- Hybrid MPI and OpenMP + I/O-server
- Bit-reproducibility: (Repeatable, OpenMP, MPI)
- Loop length: NPROMA=16
- Single Precision version 'in test'

	HRES TCo1279 L137	ENS TCo639 L91
Horiz Grid-points	6×10^6 (9km)	1.6×10^6 (18km)
Timestep	450 Seconds	720 Seconds
Forecast Length	10 days	15 Days
Nodes	350	51*20
Floating-Point Ops	30×10^{15}	$51 * 4.5 \times 10^{15}$
Time for 1 step	1 Second	2 Seconds
Total Wall time	2000 seconds	(Day 1-6) 2000 secs + (Day 6-15) 3000 secs

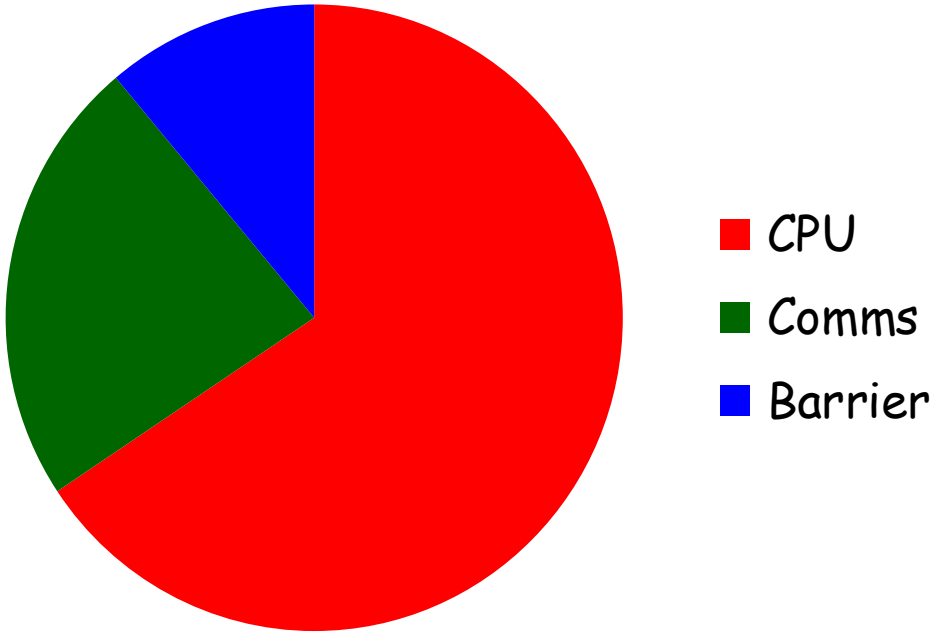
- Planned future ENS resolution → TCo1279

Profile of IFS 10-day forecast :TCo1279

IvyBridge TL1279 100 Nodes
HT:400 MPI x 12 OMP



Broadwell TCo1279 350 Nodes
No HT:1380 MPI x 9 OMP+5 IO



- Phase1 → Phase2 Aries same speed – but we have more cores
- TL1279 → TCo1279 Spectral resolution same – but bigger SL Halo

Agenda

- Upgrades at ECMWF
  - HPC configurations
  - Model resolution for Operational forecasts

- Profiling → Optimisation
  - Introduction to ecProf
  - Optimisation using ecProf profiles for IFS

2004 Dr.Hook → 2016 ecProf

2004 : Dr. Hook profile for TL511 forecast - hpca

#	%Time (self)	Cumul (sec)	Self (sec)	Total (sec)	#calls	MIPS	MFlops	%Div	Routine
1	7.43	35.027	35.027	40.573	49	961	273	2.9	WVCOUPLE@1 [567,1]
2	3.67	52.349	17.322	17.367	5824	1113	546	3.6	*CLOUDSC@1 [5,4]
3	3.65	52.349	17.204	17.287	5791	1116	548	3.6	CLOUDSC@4 [5,4]
4	3.64	52.349	17.181	17.289	5769	1118	549	3.6	CLOUDSC@2 [5,4]
5	3.63	52.349	17.138	17.202	5770	1117	549	3.6	CLOUDSC@3 [5,4]

2016 : Dr. Hook profile for TCo1279 forecast - cca

#	% Time (self)	Cumul (sec)	Self (sec)	Total (sec)	# of calls	Self ms/call	Total ms/call	Routine
1	5.40	128.736	128.736	168.385	1921	67.02	87.65	SLCOMM2A
2	5.13	251.139	122.403	138.361	126892	0.96	1.09	*CLOUDSC@2
3	5.13	251.139	122.200	138.480	128114	0.95	1.08	CLOUDSC@4
4	5.11	251.139	121.898	137.850	129372	0.94	1.07	CLOUDSC@3
5	5.11	251.139	121.854	138.091	127770	0.95	1.08	CLOUDSC@9
6	5.07	251.139	120.789	136.823	127060	0.95	1.08	CLOUDSC@6
7	5.06	251.139	120.571	136.793	128252	0.94	1.07	CLOUDSC@8
8	5.06	251.139	120.558	136.620	127062	0.95	1.08	CLOUDSC@1
9	5.04	251.139	120.217	136.400	126964	0.95	1.07	CLOUDSC@7
10	5.04	251.139	120.139	136.430	127272	0.94	1.07	CLOUDSC@5

ecProf: Motivation & background

- To optimise IFS we need information beyond what is available from in-house DrHook & GSTATS

  * These show only the partial story
  * Intrusive : need to insert function calls for invocation
  * Source line or instruction level details are missing

- CrayPAT, Allinea/MAP, Intel Vtune, …
  * Excellent tools, but also commercial (£££)
  * Not available on all platforms we liked
  * Cannot be tuned for our purposes

We had to dig deeper …

- More specialised tool was required : **ecProf** was born
- Frequent access to Linux ***/proc/<tid>/stat***
- No need for explicit profiler calls – unlike DrHook
- Uses LD_PRELOAD method for initialization
- Works with or without MPI + OpenMP
- Also access to non-IFS code performance
- Run-time instruction ptrs are mapped back to src code

Marginal gains guaranteed

- Now we discover how well/badly we vectorise
  * Often just memory bound (nearly 50%)
  * Often using SSE rather than AVX instructions
  * Hardly any FMA instructions issued
- We can see how much memory we consume
  * Even for crashed jobs – post-mortem
- We can discover (and plot) the actual thread affinities
  * These maybe wrong or incorrectly set
- Many different output profile modes available

IFS TCo1279 timings (~ 50% of total)

```
=============================================
   # %Time WallTime   #Hits : Function (file)
=============================================
   1   4.6  120.81s    2016 : cloudsc_ (cloudsc.F90)
   2   3.9  101.59s    1677 : cpg_ (cpg.F90)
   3   3.0  77.849s    1223 : lascaw_ (lascaw.F90)
   4   1.4  37.352s     600 : laitli_ (laitli.F90)
   5   1.1  29.924s     478 : laitri_ (laitri.F90)
   6   1.1  29.115s     488 : radlswr_ (radlswr.F90)
   7  0.95  24.855s     435 : cuadjtq_ (cuadjtq.F90)
   8  0.88  22.973s     402 : wvxf2gb_ (wvxf2gb.F90)
   9  0.86  22.318s     377 : cloudvar_ (cloudvar.F90)
  10  0.73  19.008s     315 : cpdyddh_ (cpdyddh.F90)
      ...

 571  0.00   0.003s       1 : suswn_ (suswn.F90)
 572  0.00   0.003s       1 : susrad$susrad_mod_ (susrad_mod.F90)
=============================================
 =>>  49.6% 1294.4s   22035 :
=============================================
```

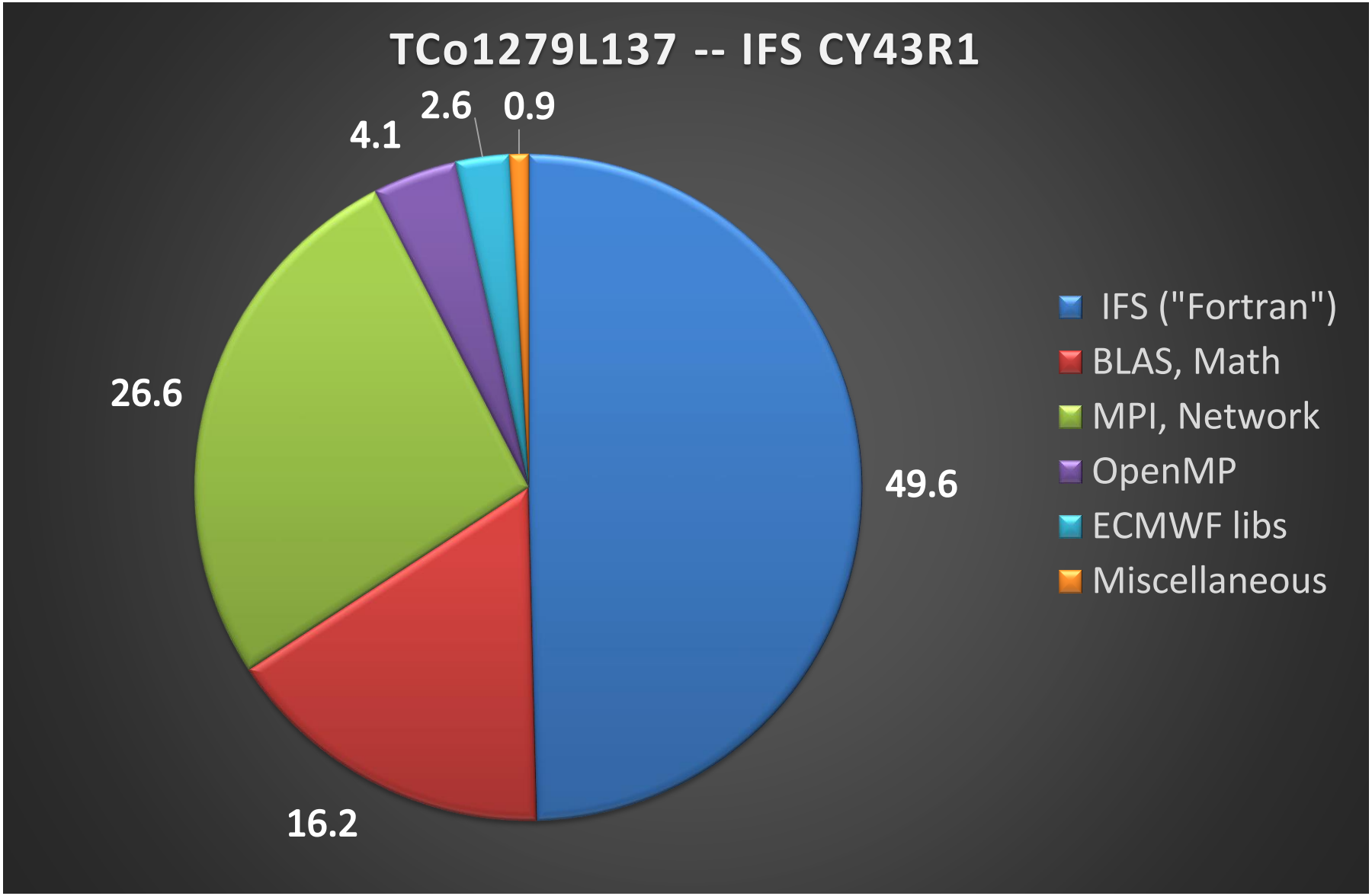
Libraries (non-IFS) : the other ~ 50%

```
=============================================
   # %Time WallTime   #Hits : Function (file)
=============================================
   1   6.3  164.31s    3061 : GNI_CqGetEvent (??)
   2   4.7  122.31s    2336 : GNII_DlaProgress (??)
   3   3.7  95.842s    1785 : MPIDI_CH3I_Progress (??)
   4   3.5  91.642s    1602 : __GI___sched_yield (??)
   5   3.3  85.149s    1593 : MPID_nem_gni_poll (??)
   6   3.0  77.418s    1442 : MPID_nem_gni_check_localCQ (gni_poll.c)
   7   2.6  67.951s    1182 : fullscan_barrier_list (omp_barrier.c)
   8   1.5  40.014s     778 : MPID_nem_gni_datagram_poll (??)
   9   1.5  39.607s     742 : MPID_nem_gni_smsg_cm_poll (??)
  10   1.5  37.827s     670 : _F90_COPY_FROM_DV (??)
       ...

 441  0.00   0.009s       1 : atlas::array::ArrayView<long, 1>::shape
=============================================
 =>>  50.4% 1313.8s   28891 :
=============================================
```

OpenMP thread ranking

```
=================================================================================
   # %Time WallTime UserCPU  SysCPU   #Hits : Function@<tid> (file)
=================================================================================
   1   4.7  122.13s 106.50s  0.474s   17436 : cloudsc_@4 (cloudsc.F90)
       4.6  120.81s 104.39s  0.492s    2016 : cloudsc_@0 (cloudsc.F90)
       4.6  119.54s 103.84s  0.462s   21110 : cloudsc_@5 (cloudsc.F90)
       4.5  117.24s 101.62s  0.487s   28478 : cloudsc_@7 (cloudsc.F90)
       4.4  115.34s 99.947s  0.506s   31837 : cloudsc_@8 (cloudsc.F90)
       4.4  115.23s 99.798s  0.446s   24767 : cloudsc_@6 (cloudsc.F90)
       4.4  114.70s 99.846s  0.467s   13516 : cloudsc_@3 (cloudsc.F90)
       4.4  114.34s 99.277s  0.479s    6107 : cloudsc_@1 (cloudsc.F90)
       4.3  112.83s 98.431s  0.486s    9780 : cloudsc_@2 (cloudsc.F90)
   2   3.9  101.59s 75.732s  0.784s    1677 : cpg_@0 (cpg.F90)
       3.0  77.976s 62.613s  0.562s   10496 : cpg_@3 (cpg.F90)
       2.9  76.656s 61.603s  0.556s    4512 : cpg_@1 (cpg.F90)
       2.9  76.621s 63.400s  0.420s   13350 : cpg_@4 (cpg.F90)
       2.9  76.603s 61.804s  0.487s   22076 : cpg_@7 (cpg.F90)
       2.9  76.147s 61.219s  0.572s    7467 : cpg_@2 (cpg.F90)
       2.9  74.743s 60.252s  0.484s   24955 : cpg_@8 (cpg.F90)
       2.9  74.694s 60.271s  0.466s   19108 : cpg_@6 (cpg.F90)
       2.8  74.135s 59.856s  0.471s   16220 : cpg_@5 (cpg.F90)
```

MPI-task#0 profile

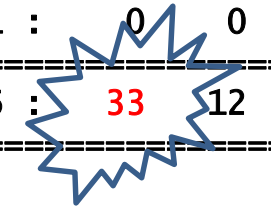


We are 46% memory bound !!!

#	%Time	WallTime	#Hits	%MEM	%VEC	%FMA	%CMP	%JMP	%OTH	Function [LIB]
1	6.3	164.31s	3061	35	0	0	0	4	61	GNI_CqGetEvent [ARIES]
2	4.7	122.31s	2336	29	0	0	6	3	63	GNII_DlaProgress [ARIES]
3	4.6	120.81s	2016	51	16	1	3	5	24	cloudsc_ [USER]
4	3.9	101.59s	1677	48	3	0	1	0	49	cpg_ [USER] (cpg.F90)
5	3.7	95.842s	1785	55	0	0	6	13	26	MPIDI_CH3I_Progress [MPI]
6	3.5	91.642s	1602	0	0	0	100	0	0	__GI___sched_yield [LIBC]
7	3.3	85.149s	1593	34	0	0	12	12	41	MPID_nem_gni_poll [MPI]
8	3.0	77.849s	1223	64	24	0	1	1	10	lascaw_ [USER]
9	3.0	77.418s	1442	35	0	0	7	5	53	MPID_nem_gni_check_localCQ [MPI]
10	2.6	67.951s	1182	25	0	0	25	0	50	fullscan_barrier_list [OMP]
	...									
1014	0.00	0.003s	1	100	0	0	0	0	0	suswn_ [USER]
1015	0.00	0.003s	1	100	0	0	0	0	0	susrad$susrad_mod_ [USER]
=>>	100%	2608.2s	50926	46	8	2	8	5	31	

BLAS etc. 33% memory bound

#	%Time	WallTime	#Hits	%MEM	%VEC	%FMA	%CMP	%JMP	%OTH	Function
1	3.5	91.642s	1602	0	0	0	100	0	0	__GI___sched_yield
2	1.5	37.827s	670	30	0	0	11	19	41	_F90_COPY_FROM_DV
3	1.1	27.592s	474	79	0	0	7	0	14	__cray_dcopy_HSW
4	1.1	27.581s	477	8	88	0	0	0	4	_RTOR_00
5	0.85	22.190s	360	26	25	39	0	0	10	dgemm_kernel_a1b0_haswell
6	0.76	19.724s	659	100	0	0	0	0	0	__read_nocancel
7	0.74	19.382s	333	53	13	0	7	0	27	__cray_dset_HSW
8	0.71	18.514s	313	42	15	35	0	0	9	dgemm_kernel_nocopy_nn_a1b1_haswell
9	0.64	16.786s	288	29	46	12	0	0	12	__cray2_EXP_W_01
10	0.50	13.061s	216	39	6	0	0	0	56	dgemm_tcopy_4_avx_autogen
	...									
187	0.00	0.013s	1	0	0	0	0	0	100	__rawmemchr_sse2
=>>	16.2%	422.75s	7606	33	12	6	26	4	19	



LASCAW – per line profile

```
=============================================================
   # %Time WallTime   #Hits : [OpCode] Function (file:lineno)
=============================================================
   1  0.21   5.455s      82 : [mov] lascaw_ (lascaw.F90:557)
   2  0.17   4.492s      68 : [mov] lascaw_ (lascaw.F90:560)
   3  0.16   4.137s      62 : [mov] lascaw_ (lascaw.F90:565)
      ...
  18  0.04   1.052s      18 : [vcvttsd2si:xmm] lascaw_ (lascaw.F90:525)
      ...
  22  0.03   0.812s      13 : [vsubpd:ymm] lascaw_ (lascaw.F90:523)
      ...
 236  0.00   0.010s       1 : [vsubpd:ymm] lascaw_ (lascaw.F90:672)
=============================================================
 =>>   3.0% 77.855s    1223 :
=============================================================
```

SSE

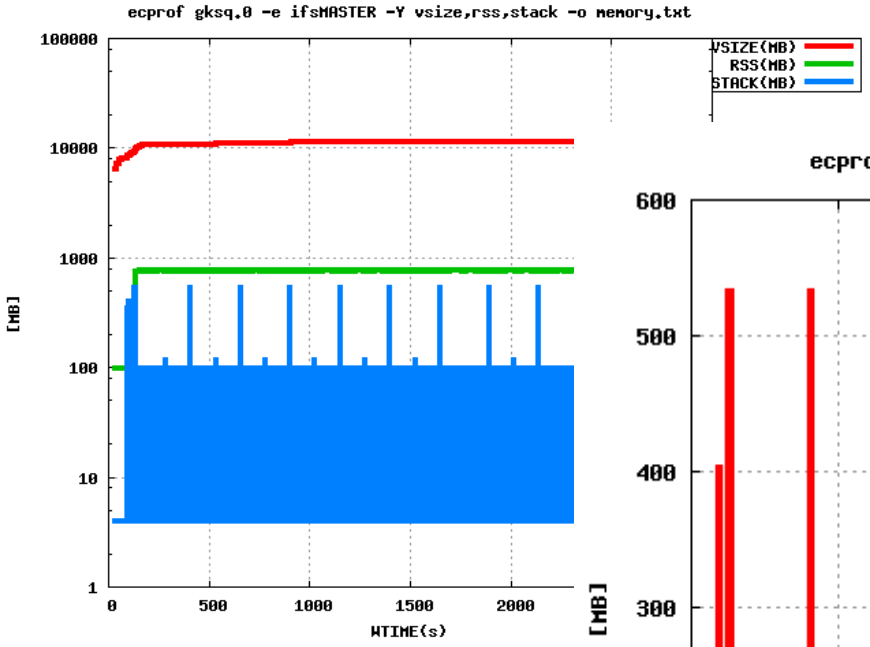
AVX

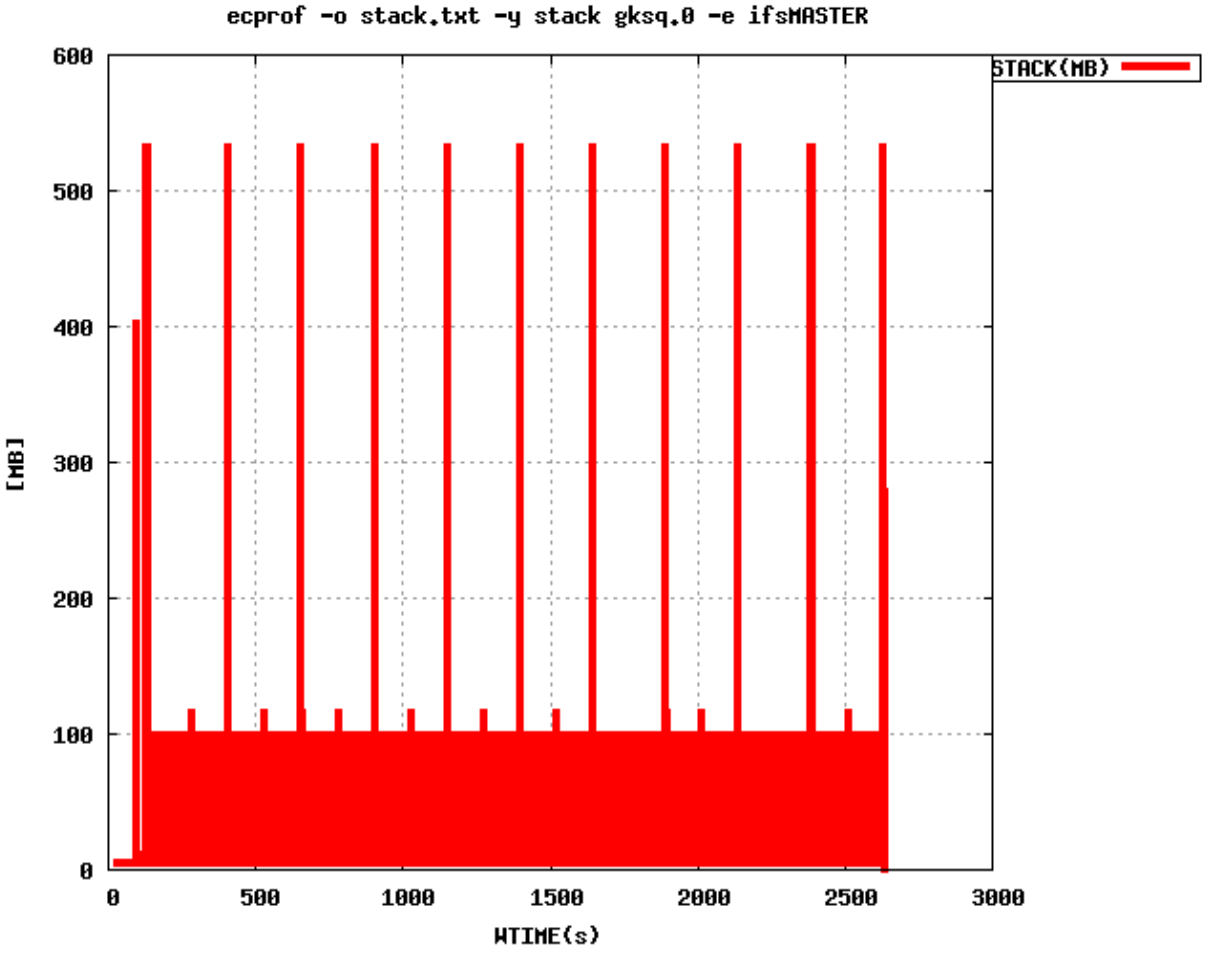
Embedding profiles with source code

```
================================================================================
   # %Time WallTime   #Hits : %MEM %VEC %FMA %CMP %JMP %OTH : [OpCode] Function
================================================================================
   1  0.21   5.455s      82 :  100    0    0    0    0    0 : [mov] lascaw_
          /home/rd/rdx/rd_source/43r1/ifs/interpol/lascaw.F90

       555 |          YDSL%MASK_SL2(ILO (JROF)-IDIF+3)=1
       556 |          YDSL%MASK_SL2(ILO1(JROF)-IDIF  )=1
    000557 >          YDSL%MASK_SL2(ILO1(JROF)-IDIF+1)=1
       558 |          YDSL%MASK_SL2(ILO1(JROF)-IDIF+2)=1
       559 |          YDSL%MASK_SL2(ILO1(JROF)-IDIF+3)=1

  22  0.03   0.812s      13 :    0  100    0    0    0    0 : [vsubpd:ymm] lascaw_
          /home/rd/rdx/rd_source/43r1/ifs/interpol/lascaw.F90

       521 |        ZLO3  =PLON(JROF,JLEV)*ZLSDEPI_(IJ_,ILA+3)
       522 |        ILO3(JROF)  =INT(ZLO3)
    000523 >        PDLO(JROF,JLEV,3)=ZLO3-REAL(ILO3(JROF),JPRB)
       524 |
       525 |        ILEV  =KVAUT(INT(PLEV(JROF,JLEV)*ZFAC))-1
```

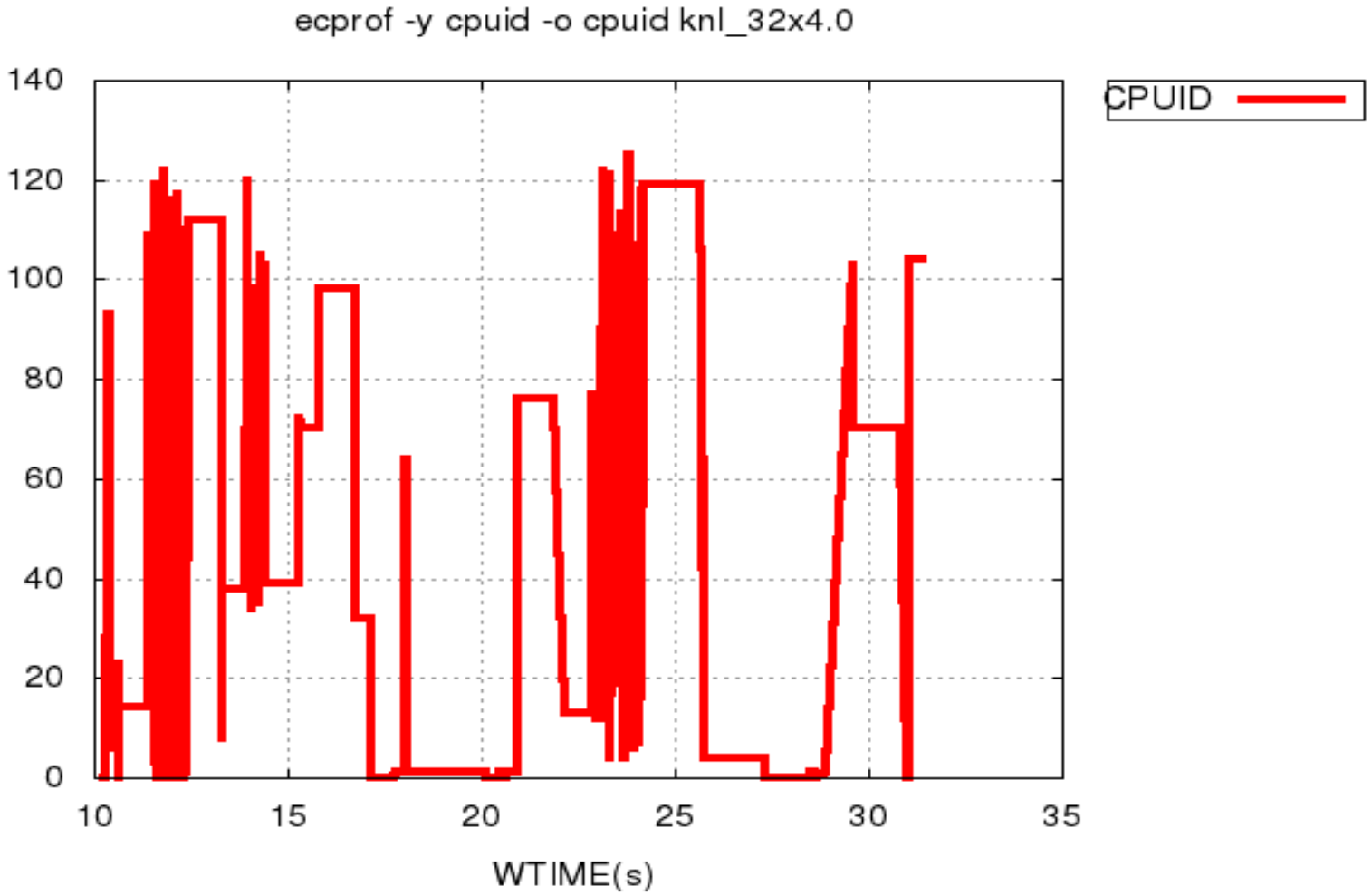
AVX

Stack usage unexpectedly high
→ the culprit quickly identified in IFS





Thread affinity issue on KNL? - One of our early runs issue



Optimisation of IFS for Broadwell

- With help of **ecProf** and looking at vectorisation reports
  - Reduced memory traffic (array copies & initialisations)
  - Improved vectorisation
- Example of **LASCAW*** with **ecProf : 1.6X** speed-up

```
%Time WallTime UserCPU  SysCPU  : %MEM %VEC %FMA %CMP %JMP %OTH : Function
=================================================================================
 2.9  73.554s 52.884s  0.234s  :   67   24    0    2    1    6 : lascaw_ (Orig)
 1.9  45.744s 33.363s  0.157s  :   48   38    0    0    1   13 : lascaw_ (Opt)
```

IFS CY43R1	Secs	Tflops	Speedup
Original	**2351**	**12.76**	
With various optimisations	**1950**	**15.38**	**1.2X**

*** Thanks to suggestions from Ilias Katsardis, John Hague and Ryad El Khatib**

Thank You !

Any questions?