

CFD Example Results

EPSRC

NERC SCIENCE OF THE ENVIRONMENT

archer

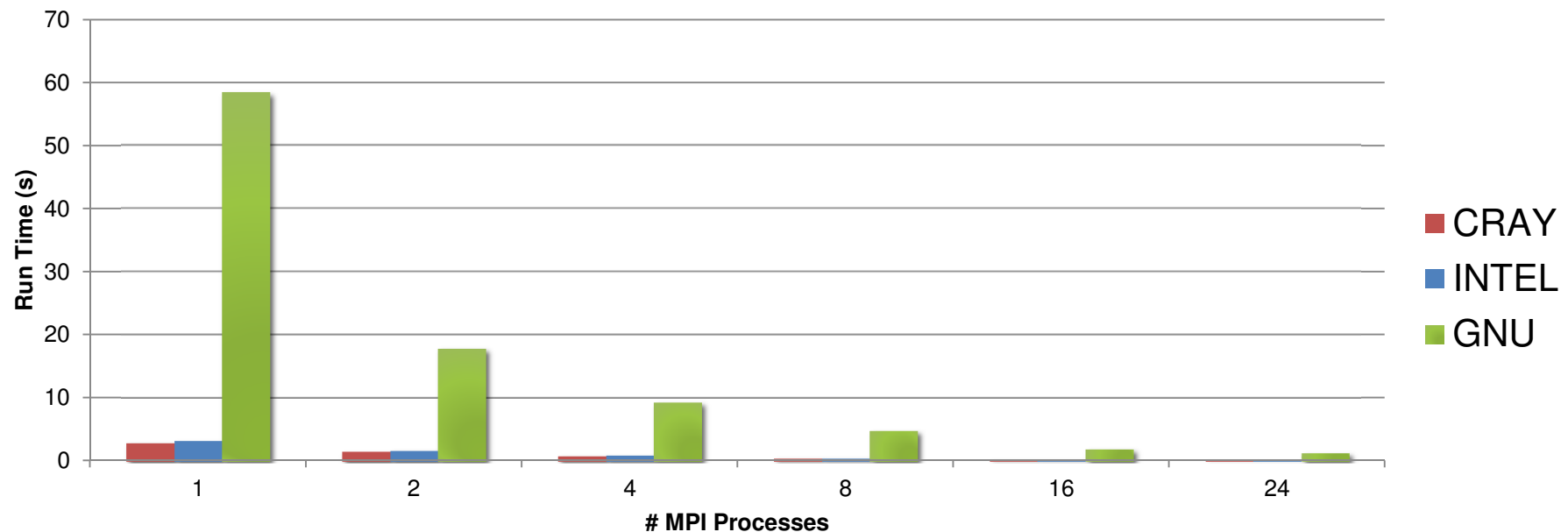
CRAY
THE SUPERCOMPUTER COMPANY

epcc



Compiler Implementation and Platform

- Three compilers on ARCHER: Cray, Intel and GNU.
- Cray and Intel: more optimisations on by default, likely to give more performance out-of-the-box.
- ARCHER is a Cray system using Intel processors. Cray compiler tuned for the platform, Intel compiler tuned for the hardware.



- GNU compiler likely to require additional compiler options...



Compiler Optimisation Options

- Flags for the compiler. Can be set on the command line or in the Makefile.
- Standard levels:
 - O3 Aggressive
 - O2 Suggested
 - O Conservative
 - O0 Off (for debugging)
- Finer tuning available. Details in compiler man pages.
- Higher levels aren't always better. Increased code size from some optimisations may negatively impact cache interactions.
- Can increase compilation time.



Hyper-Threading

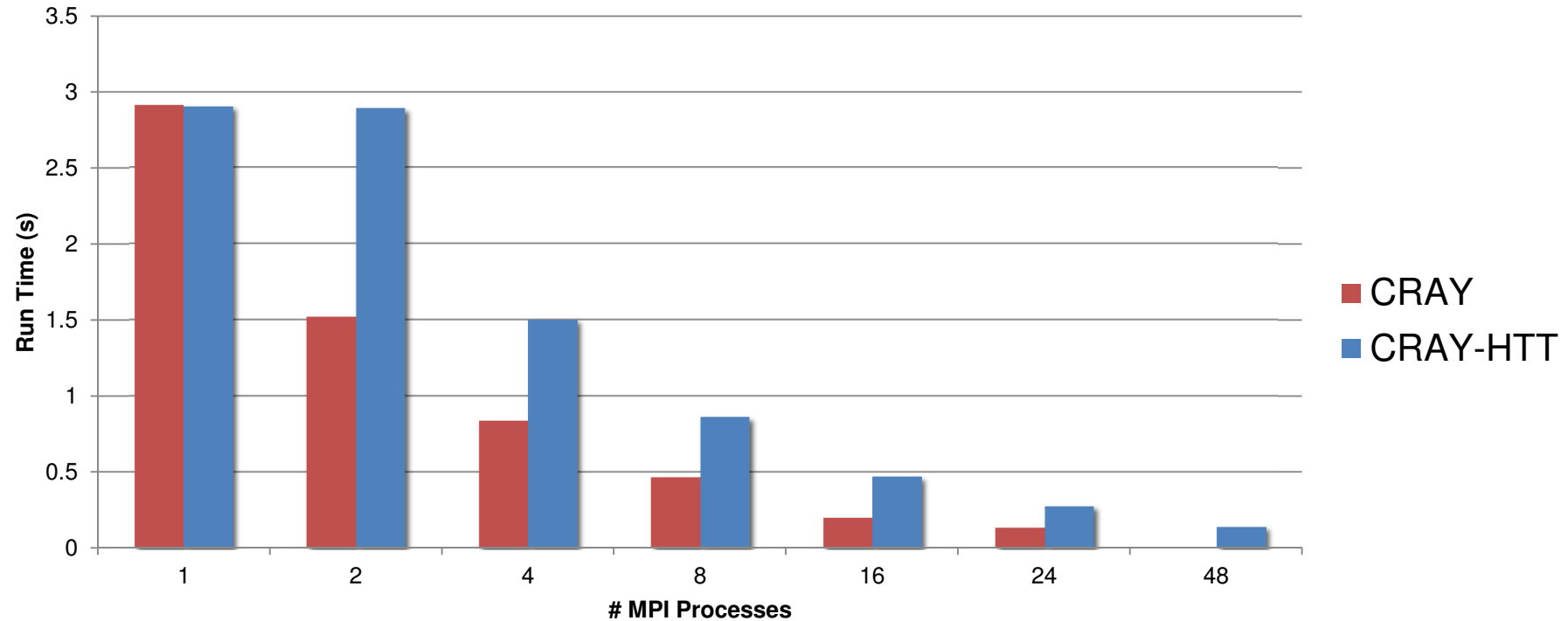
- Intel technology – designed to increase performance using simultaneous multi-threading (SMT) techniques.
- Presented as one additional *logical core* per physical one on the system.
- Each ARCHER node therefore reports a total of 48 available processors (can be confirmed by checking `/proc/cpuinfo`).
- Must be explicitly requested with the “-j 2” option:

```
#PBS -l select=1  
aprun -n 48 -j 2 ./myMPIProgram
```

- Hyper-Threading doubles the number of available parallel units per node at no additional resource cost.
- However, performance effects are highly dependent on the application...



Hyper-Threading Performance



- Can have a positive or negative effect on run times.
- Hyper-Threading is a bad idea for the CFD problem.
- Experimentation is key to determining if this technique would be suitable for your code.



Process Placement

- ARCHER is a NUMA system – processors access different regions of memory at different speeds.
- Compute nodes have two NUMA regions – one for each CPU. Hence 12 cores per region.
- It may be desirable to control which NUMA regions processes are assigned to.
- For example, with hybrid MPI and OpenMP jobs, it is suggested that processes are placed such that shared-memory threads in the same team access the same local memory.
- Can be controlled with *aprun* flags such as:
 - -N [parallel processes per node]
 - -S [parallel processes per NUMA region]
 - -d [threads per parallel process]



Parallel Scaling – Number of Processors

- Addition of parallel resources subject to diminishing returns.
- Depends on scalability of underlying algorithms.
- Any sources of inefficiency are compounded at higher numbers of processes.
- In the CFD example, run time can become dominated by MPI communications rather than actual processing work.

CFD Code	Iterations: 10,000	Scale Factor: 70		
MPI procs	Time	Speedup	Efficiency	
	1	331.34	1.00	1.00
	2	180.30	1.84	0.92
	4	132.16	2.51	0.63
	8	121.23	2.73	0.34
	16	89.02	3.72	0.23
	24	58.70	5.64	0.24



Parallel Scaling – Problem Size

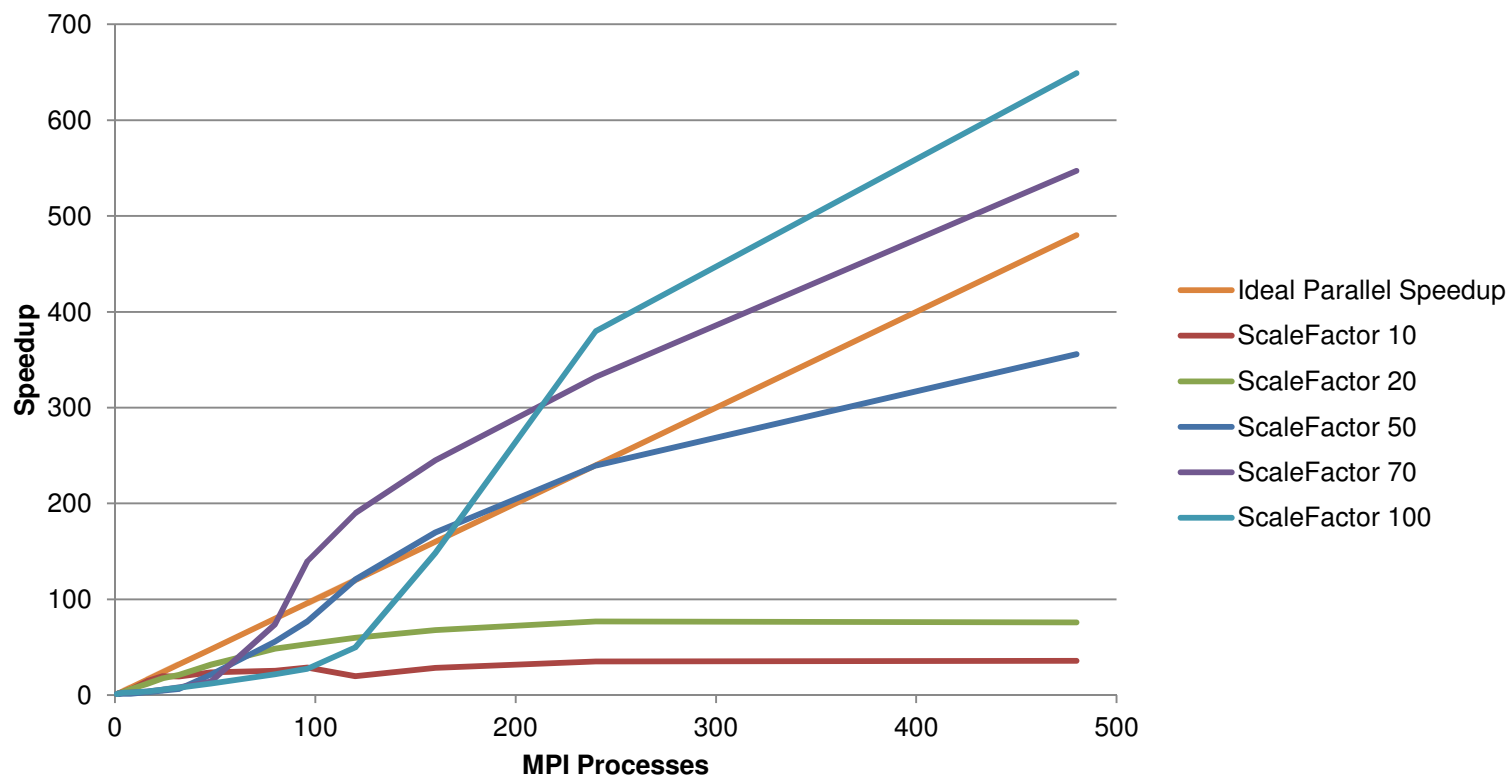
- Problem scale affects memory interactions – notably cache accesses.
- Additional processors provide additional cache space.
- Can lead to more, or even all, of a program's working set being available at the cache level.
- Configurations that achieve this will show a sudden efficiency “spike”.

CFD Code	Iterations: 10000	Scale Factor: 70		
MPI procs	Time	Speedup	Efficiency	
1	331.34	1.00	1.00	
48	23.27	14.24	0.30	
96	2.37	139.61	1.45	

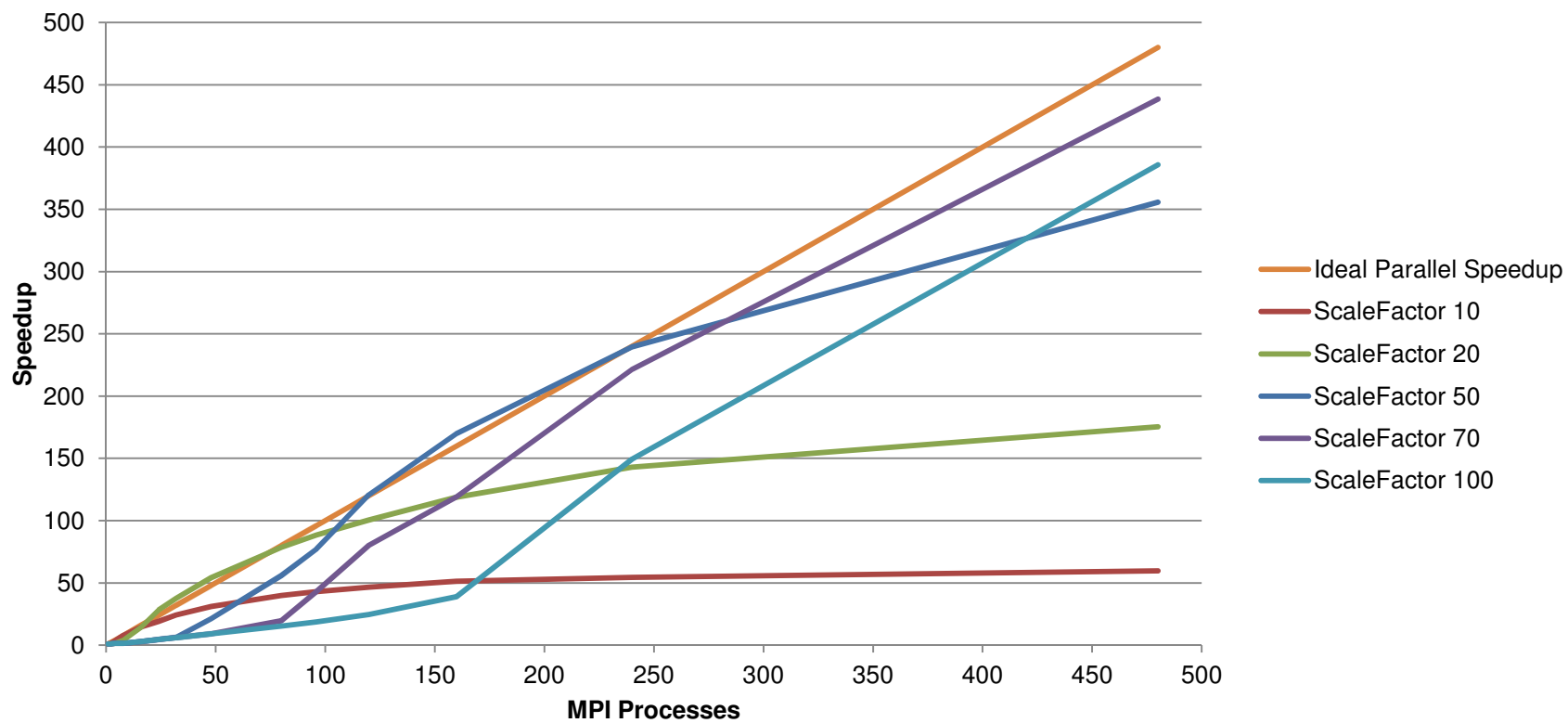
- 2x the number of MPI processes gives ~9.8x the speed up.



CFD Speedup on ARCHER



CFD Speedup on HECToR



ARCHER-ScaleFactor 10					ARCHER-ScaleFactor 20				
MPI procs	Time	Speedup	Efficiency		MPI procs	Time	Speedup	Efficiency	
1	2.91	1.00	1.00		1	11.92	1.00	1.00	
2	1.52	1.91	0.96		2	6.21	1.92	0.96	
4	0.84	3.47	0.87		4	3.38	3.52	0.88	
8	0.47	6.22	0.78		8	1.86	6.41	0.80	
16	0.20	14.46	0.90		16	1.00	11.91	0.74	
24	0.15	19.92	0.83		24	0.68	17.52	0.73	
32	0.15	19.45	0.61		32	0.57	21.03	0.66	
48	0.12	23.90	0.50		48	0.37	31.95	0.67	
80	0.11	25.63	0.32		80	0.25	48.43	0.61	
96	0.10	28.95	0.30		96	0.22	53.17	0.55	
120	0.15	19.78	0.16		120	0.20	59.86	0.50	
160	0.10	28.36	0.18		160	0.18	67.90	0.42	
240	0.08	35.14	0.15		240	0.16	76.77	0.32	
480	0.08	35.87	0.07		480	0.16	75.94	0.16	
HECToR-ScaleFactor 10					HECToR-ScaleFactor 20				
MPI procs	Time	Speedup	Efficiency		MPI procs	Time	Speedup	Efficiency	
1	8.91	1.00	1.00		1	48.42	1.00	1.00	
2	8.01	1.11	0.56		2	44.30	1.09	0.55	
4	2.77	3.21	0.80		4	30.68	1.58	0.39	
8	1.12	7.99	1.00		8	11.97	4.04	0.51	
16	0.61	14.56	0.91		16	3.34	14.49	0.91	
24	0.46	19.16	0.80		24	1.71	28.27	1.18	
32	0.37	24.28	0.76		32	1.29	37.59	1.17	
48	0.29	31.00	0.65		48	0.89	54.28	1.13	
80	0.22	39.80	0.50		80	0.62	78.63	0.98	
96	0.21	43.06	0.45		96	0.55	88.33	0.92	
120	0.19	46.47	0.39		120	0.48	100.57	0.84	
160	0.17	51.25	0.32		160	0.41	118.94	0.74	
240	0.16	54.58	0.23		240	0.34	143.04	0.60	
480	0.15	59.81	0.12		480	0.28	175.50	0.37	



ARCHER-ScaleFactor 100				ARCHER-ScaleFactor 150			
MPI procs	Time	Speedup	Efficiency	MPI procs	Time	Speedup	Efficiency
1	694.66	1.00	1.00	1	1577.00	1.00	1.00
2	378.47	1.84	0.92	2	856.87	1.84	0.92
4	272.62	2.55	0.64	4	617.34	2.55	0.64
8	250.92	2.77	0.35	8	569.49	2.77	0.35
16	184.39	3.77	0.24	16	423.34	3.73	0.23
24	121.45	5.72	0.24	24	280.15	5.63	0.23
32	88.64	7.84	0.24	32	207.53	7.60	0.24
48	56.98	12.19	0.25	48	134.89	11.69	0.24
80	31.66	21.94	0.27	80	77.95	20.23	0.25
96	25.26	27.50	0.29	96	69.59	22.66	0.24
120	13.89	50.02	0.42	120	53.61	29.42	0.25
160	4.68	148.34	0.93	160	37.43	42.14	0.26
240	1.83	379.89	1.58	240	19.89	79.30	0.33
480	1.07	648.81	1.35	480	4.96	317.79	0.66

HECToR-ScaleFactor 100				HECToR-ScaleFactor 150			
MPI procs	Time	Speedup	Efficiency	MPI procs	Time	Speedup	Efficiency
1	1229.85	1.00	1.00	1	2794.46	1.00	1.00
2	1135.95	1.08	0.54	2	2545.46	1.10	0.55
4	810.08	1.52	0.38	4	1823.64	1.53	0.38
8	803.56	1.53	0.19	8	1803.73	1.55	0.19
16	404.02	3.04	0.19	16	903.92	3.09	0.19
24	270.39	4.55	0.19	24	604.05	4.63	0.19
32	203.32	6.05	0.19	32	454.35	6.15	0.19
48	135.61	9.07	0.19	48	304.80	9.17	0.19
80	80.72	15.24	0.19	80	183.54	15.23	0.19
96	66.10	18.61	0.19	96	152.96	18.27	0.19
120	50.12	24.54	0.20	120	122.20	22.87	0.19
160	31.63	38.88	0.24	160	91.26	30.62	0.19
240	8.23	149.44	0.62	240	58.37	47.87	0.20
480	3.19	385.72	0.80	480	11.20	249.48	0.52

