Profiling Methodology and Performance Tuning of the Met Office Unified Model for Weather and Climate Simulations

Peter Strazdins
Computer Systems Group,
School of Computer Science,
The Australian National University

collaborators: Margaret Kahn (NCI NF), Joerg Henrichs (Oracle), Tim Pugh (BoM), Mike Rezny (Monash)

Workshop on Parallel and Distributed Scientific and Engineering Computing, Anchorage, 20 May 2011

(slides available from http://cs.anu.edu.au/~Peter.Strazdins/seminars)

THE AUSTRALIAN NATIONAL UNIVERSITY

1 Overview

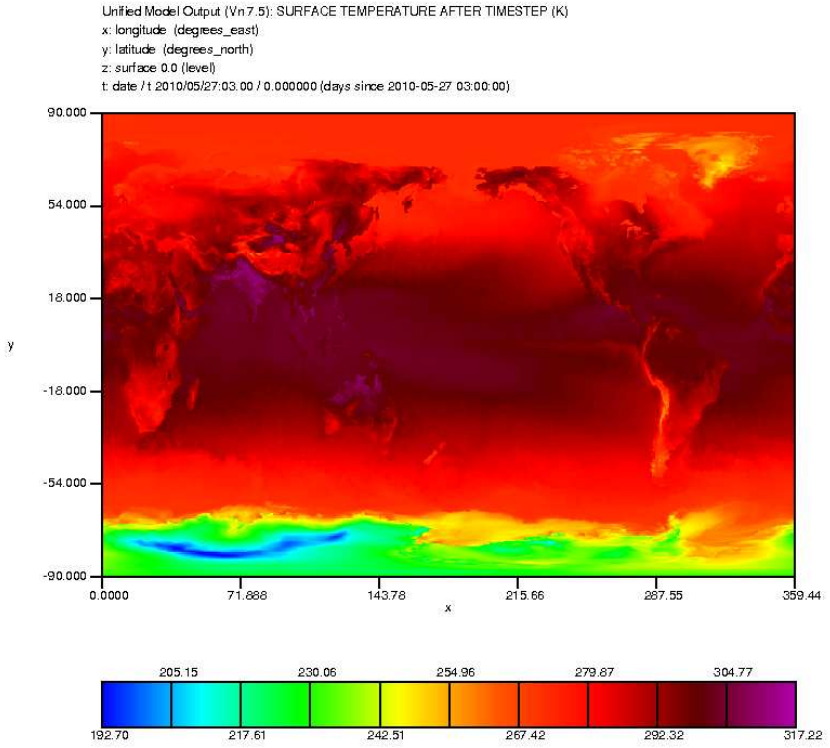
- the Met Office Unified Model for weather and climate simulations
  - initiatives in Australia: BoM and ACCESS
  - overview of the software, including internal profiler module
- the vayu cluster at the NCI National Facility
- preliminary experiences with the UM_N320L70 benchmark
  - variability analysis
- profiling methodology: aim to discover bottlenecks (where & why) with:
  - efficiency (CPU hours) and low-overhead (time, space)
- scalability analysis & validation of methodology
- load imbalance and affinity issues: variability and performance
- performance tuning
- memory-intensive parallel applications: tricks of the trade
- conclusions and future work

2 The Unified Model in Aust. Weather and Climate Simulations

- the Met Office Unified Model (MetUM, or just UM) is a (global) atmospheric model developed by the UK Met Office from early '90s
- for weather, BoM currently uses a N144L50 atmosphere grid
  - wish to scale up to a N320L70 ($640 \times 481 \times 70$) then a N512L70 ($1024 \times 769 \times 70$) grid
  - operational target: 24 hr simulation in 500s on $<$ 1K cores (10-day 'ensemble' forecasts)
  - doubling the grid resolution increases 'skill' but is $\leq 8\times$ the work!
- climate simulations currently use a N96L38 ($192 \times 145 \times 38$) grid
  - ACCESS project to run many (long) runs for IPCC 2011
    - common infrastructure: atmosphere: UM (96 cores); ocean: NEMO, sea ice: CICE, coupler: OASIS (25 cores)
  - next-generation medium-term models to use N216L85 then N320L70
- note: (warped) 'cylindrical' grids are easier to code but problematic ...

3 The MetOffice Unified Model

- configuration via UMUI tool creates a directory with (conditionally-compiled) source codes + data files (for a particular grid)
  - main input file is a ‘dump’ of initial atmospheric state (1.5GB for N320L70)
  - ‘namelist’ files for ≈ 1000 run-time settable parameters
  - in operational runs, periodically records statistics via the STASH sub-system
- partition evenly the EW & NS dimensions of the atmosphere grid on a $P \times Q$ (MPI) process grid



4 Unified Model Code Structure and Internal Profiler

- codes in Fortran-90 (mostly F77; ≈ 900 KLOC) with cpp (include common blocks, commonly used parameter sub-lists, etc)
- main routine `u_model()`, reads dump file & repeatedly calls `atm_step()`
  - dominated by Helmholtz P-T solver (GCR on a tridia. linear system)
- internal profiling module can be activated via 'namelist' parameters
  - has 'non-inclusive' +'inclusive' timers (≈ 100 of each)
  - the top-level non-inclusive timer is for `u_model()`; sum of all non-inclusive timers is time for `u_model()`
  - reports number of calls and totals across all processes, e.g.

```
    ROUTINE              MEAN   MEDIAN       SD   % of mean      MAX   ...
1 PE_Helmholtz         206.97   206.98     0.05       0.02%   207.02   ...
3 ATM_STEP              36.39    38.53     9.46      25.99%    44.60   ...
4 SL_Thermo             25.38    26.60     3.45      13.58%    31.15   ...
5 READDUMP              24.18    24.36     1.12       4.62%    24.37   ...
...
```

  - due to global sync. when a timer starts, can estimate load imbalance

5 The Vayu Cluster at the NCI National Facility

- 1492 nodes: two quad-core 2.93 GHz X5570 quad-core Nehalems (commissioned Mar 2010)
- memory hierarchy: 32KB (per core) / 256KB (per core) / 8MB (per socket); 24 GB RAM
- single plane QDR Infiniband: latency of 2.0μs latency & 2600 MB/s (uni-) bandwidth per node

- jobs (parallel) I/O via Lustre filesystem
- jobs submitted via locally modified PBS; (by default) allocates 8 consecutively numbered MPI processes to each node
  - typical snapshot:

```
1216 running jobs (465 suspended), 280 queued jobs, 11776 cpus in use
```

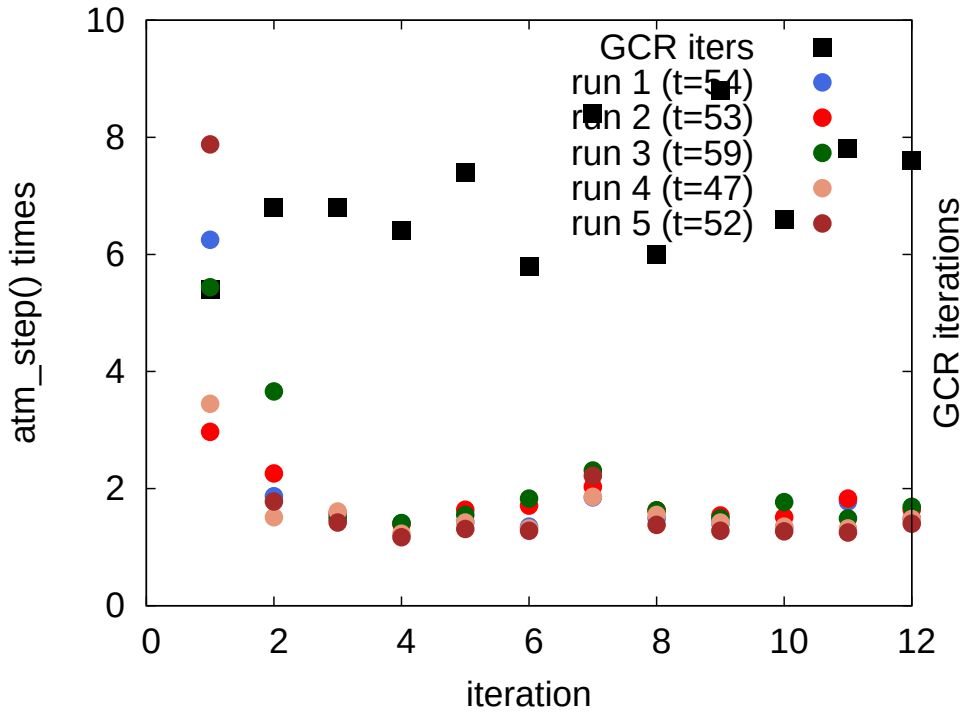
  - estimating time and memory resources accurately is important!
- the UM profiling project was (in 2010) allocated a few thousand CPU hours, max. core count 2048 . . .

6 Preliminary Experiences on the N320L70 Benchmark

- used OpenMPI 1.4.1; STASH output was disabled (to simplify)
- memory-related difficulties in running on < 8 or > 1536 cores, and on process grids like 60×24
- Sunstudio `collect` (with MPI & hardware perf. counters) would have been ideal, but only worked on the N96L43 grid
- the more lightweight IPM only worked to 256 processes
  - here, 44% of time spent in MPI, mainly barriers (load imbalance)
- variability in repeated run times of $\leq 50\%$ (bimodal; 'fast' runs $\leq 10\%$)
  - was worse (and slower) without OpenMPI process affinity flags
- preliminary subroutine profiling revealed 'inverse' scaling on:
  - `read_dump()` (25s plus, but is called once only)
  - `q_pos_ctl()`: many gather and scatter communications
    - $\Rightarrow$ Met Office subsequently supplied the PS24 patches

7 Preliminary Experiences: Inter-Iteration Variability

- inter-iteration variability within `atm_step()`
  - 1st step took 2–5× longer; every 6th step after slightly longer
    - partially alleviated by 'flushing' (entire!) physical memory 1st
    - much was in `q_pos_ctl()`; mostly due to setting up MPI connections
  - variability is not due to the number of iterations in the Helmholtz solver



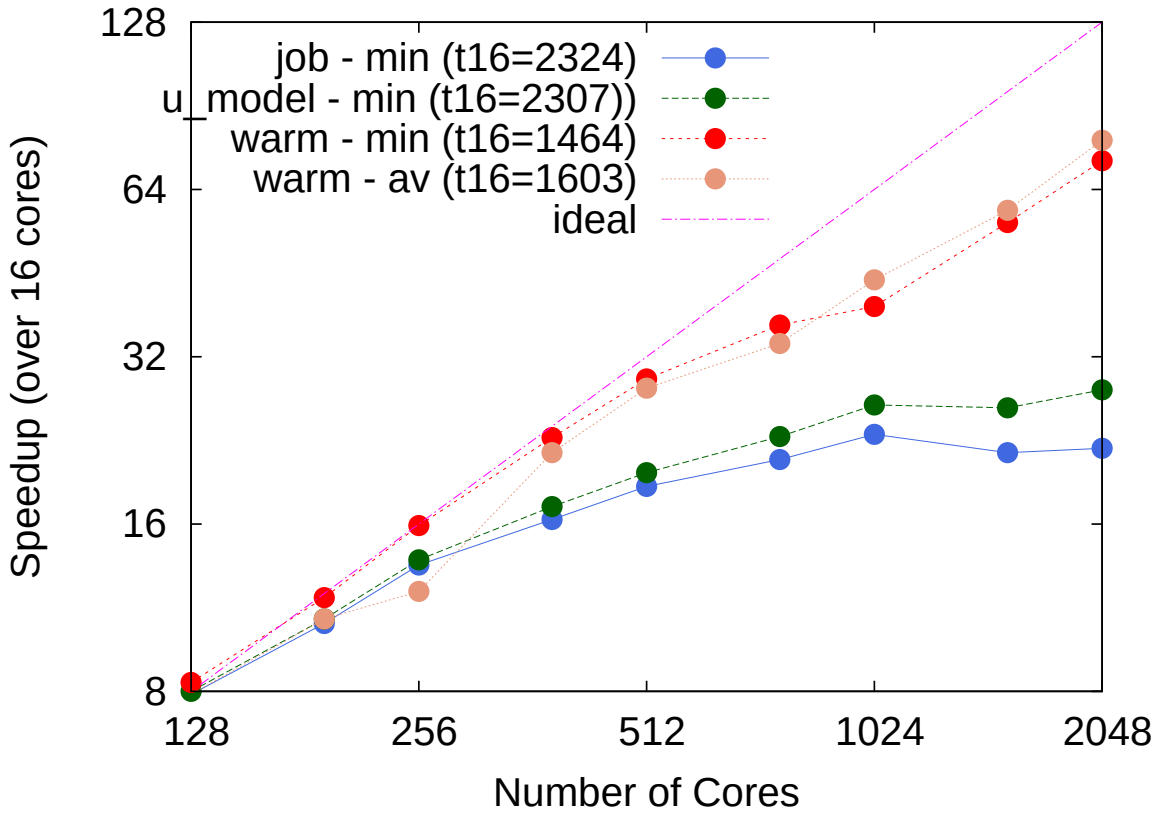
times & GCR iters. (scale: 0 to 50) per time step (each 12 mins sim. time)

8 Profiling Methodology

- based on these experiences, the following principles are desired:
  - infrastructure should have minimal effect on runtime (or memory)
  - accurately predict long-term simulation time
  - take into account all variabilities (as far as possible)
- resulting in the following methodology:
  - take at least 5 timings for each configuration (use min., or avg.?)
  - run for the minimal number of timesteps for accurate projections (chose 3 hours)
    - need to profile the ‘warmed period’ (hours 2–3) separately
  - reduce the overhead of profiling, and measure (or estimate) its extent
    - use 1st hour to determine which timers were ‘important’ and only do global syncs for these (hard!)
  - time each iteration of `atm_step()` for later analysis

9 Scalability Analysis: Which Time to Take? (N512L70 + PS24)

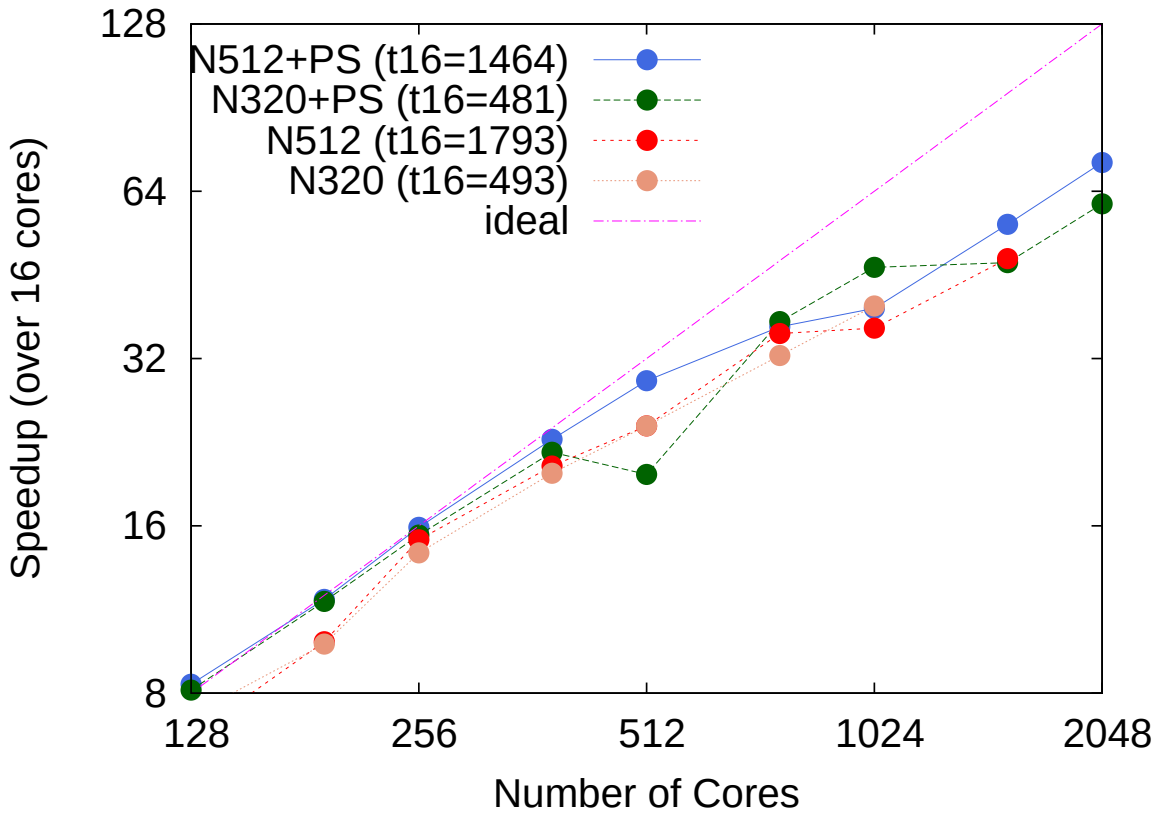
- process grids aspects between 1:1 and 1:2 chosen
- 't16' is time for 16 cores
- essentially linear scaling from 16 to 64 cores (slightly super-)
- surprisingly, average & minimum times show similar curves



- job time @ 1024 cores includes: pre-launch: 2s, launch processes: 4s, read 'namelist' files: 6s, `read_dump()`: 27s, cleanup: 1s

10 Results: Scalability of the N320/N512 Benchmarks with/out PS24

- 't16' is time for 16 cores (N512L70 $\approx 4\times$ more – uses the same timestep)
  - really ate up our Vayu quotas!
- N512L70 scaled better ($1.6\times$ higher volume:surface)
- bus errors occurred for non-PS24 for $>$ 1024 cores
- PS24 also scaled better



11 Individual Subroutine Scalability (N512L70, 1536 cores)

function	UM7.5			UM7.5+PS24		
	s	% t_w	% im.	s	% t_w	% im.
PE_Helmholtz	59.4	51	0	61.6	48	0
atm_step	15.1	11	27	11.4	25	20
q_pos_ctl	0.8	12	0	5.4	0	10
SL_Full_wind	50.8	10	47	49.1	13	46
SL_Thermo	52.7	6	20	53.3	9	19
Convect	28.0	9	54	66.2	5	37
SF_EXPL	6.7	6	69	35.4	1	75
Atmos_Physics2	12.9	5	71	52.9	1	28
total:	52.1	100	17	55.8	100	30

- s is the speedup (over 16 cores – ideally $s = 96$), % t_w is % of warmed period time, % im. is fraction of time due to load imbalance
- none scale perfectly; scope for worthwhile optimization in all
- PS24 patches have solved `q_pos_ctl()` problems
- the (underestimated) load imbalance is significant in most!

12 Validation of Profiling Methodology

- projected run time for 24hr operational job (960 cores) is

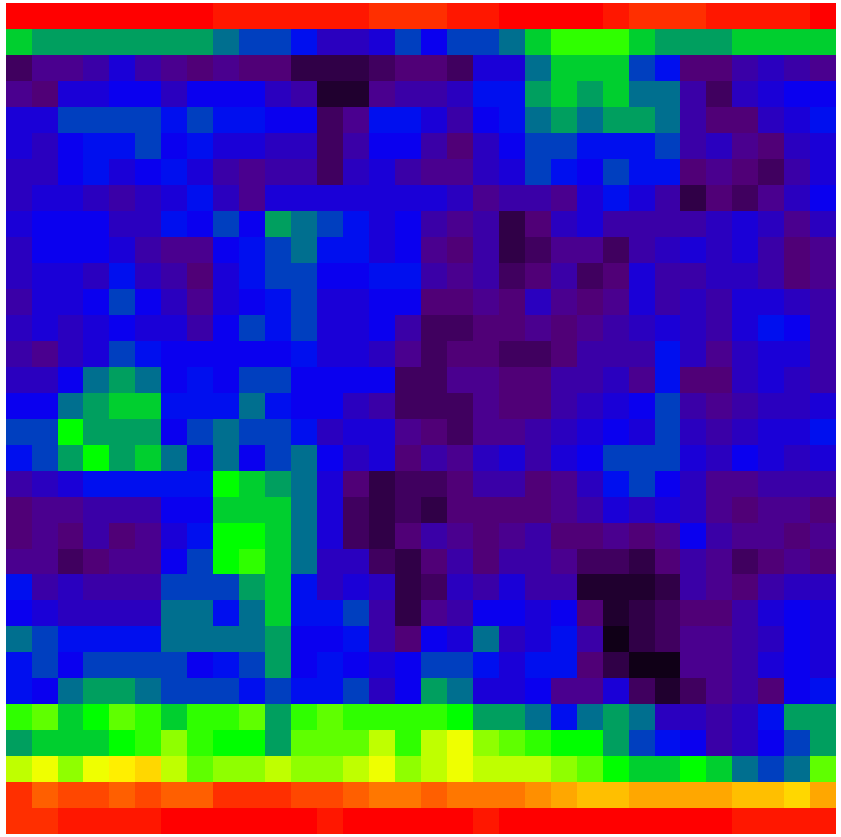
  $t' = (t - t_{2:24}) + 11.5 t_{2:3}$ $\qquad (t_{1:1} = t - t_{2:24})$

run	t	$t_{2:24}$	$t_{2:3}$	t'	anomalies
N512L70	527	39.12	6.5	524.3	–
N320L701	224	163.8	13.7	214.8	iter. 59 $(7.9s$– av.$1.1s)$
N320L702	237	174.9	14.9	233.6	iter. 134 $(4.2s$– av.$1.1s)$

- defensive programming check: sum of 'non-inclusive' timers matched total to less than 0.1%
- methodology reduced number of barriers within the profiler by factor of 10
- measured profiler overhead (gather data, barriers) of $< 1\%$ of 'warmed period' times
  - overhead of printing output needs further investigation for large core counts

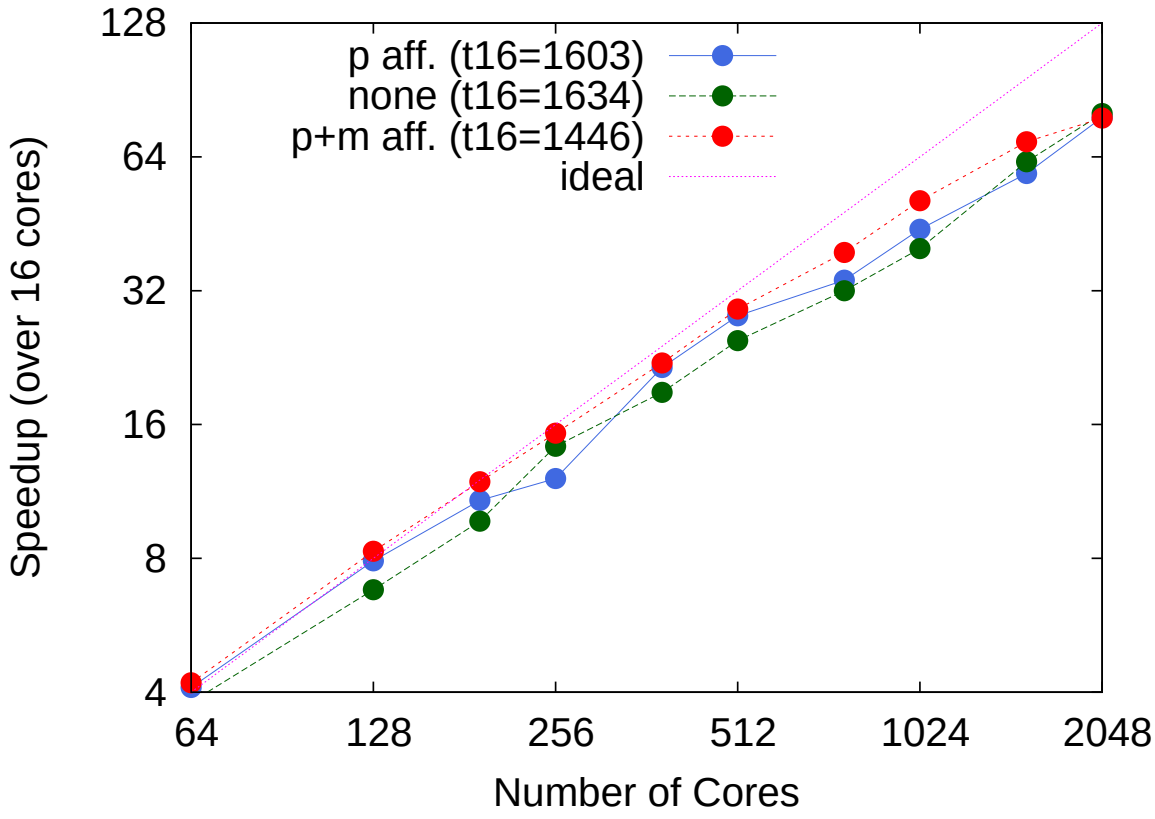
13 Load Imbalance Across Cores (32×32 process grid)

- over 'warmed period'; Red = 0%, Black = 15% of run-time
- from time spent in barriers within UM internal profiler
  - large value indicates a lighter load
- clear indication of latitudinal variations
- difficult to avoid with fixed, regular data distribution



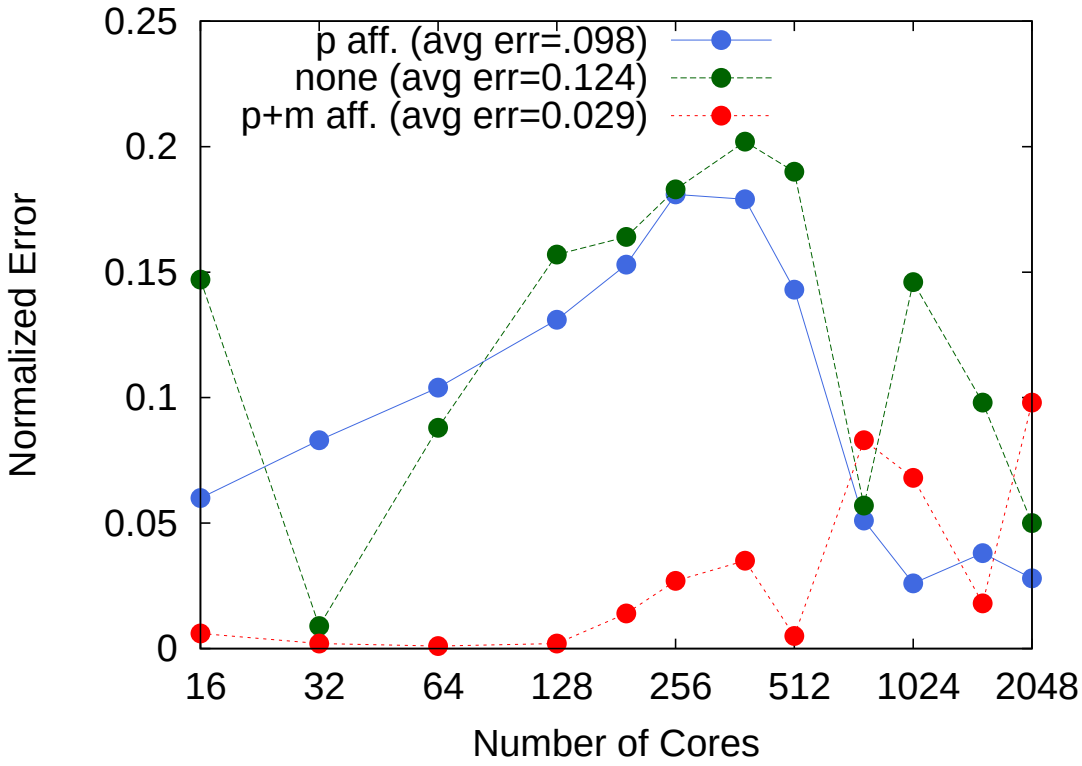
14 Effect of Process and NUMA Affinity on Scaling

- (local) OpenMPI 1.4.3 has process and NUMA affinity available
- compared with OpenMPI 1.4.1 with/out former
- note differing values for t16!
- on the X5570, local:remote memory access is 65:105 cycles
- indicates a significant amount of L3$ misses



15 Effect of Affinity on Variability

- use the normalized error from the average $\frac{\Sigma_{i=1}^{n}|t_i-\bar{t}|}{n\bar{t}}$
- noting the number of measurements ($n = 5$) only sufficient to observe general trends:
  - no clear correlation of error and number of cores
  - process affinity reduces variability by 20%
  - NUMA affinity reduces this further by a factor of 4!



(for 'warmed time' of PS24/N512L70)

16 Tuning: Segment Size Parameters

- tuning segment sizes in the short-wave / long-wave radiation & convection routines:
  - N96L38 benchmark (climate): $\approx 20\%$ improvement in each routine, $\approx 4\%$ overall
  - N320L70: negligible – as predicted by the profiling
- process grid aspect ratios: results so far are near-square
  - for the N512L70 with PS24, near-square indeed optimal:

grid:	10×24	12×20	15×16	20×12	24×10
time :	100.1	97.0	97.4	98.0	98.2

grid:	15×64	20×48	30×32
time (s):	33.3	31.1	28.1

- recall that `read_dump()` takes significant time
  - investigation showed that much of this spent in C byte-swapping code (not active on Power 6 based systems)
  - default UM config. compiled this code without optimization!
  - simply using `icc -O3` reduced time by 42% (16 cores)
  - will have large impact when STASH output on (operational runs)

17 Tricks of the Trade: Memory-intensive Applications on the Vayu Cluster

- PBS issues:
  - maximum time estimate needs to be tight (job gets killed vs very long wait times or wasted CPU hours if job 'hangs')
  - similarly for maximum memory (N512L70 close to limit at small & large core counts)
- memory-related problems: solution:
  - job crashed relatively early: `ulimit -s unlimited`
  - internal comm. error < 16 cores: redefine internal message buffer size
  - crashes in OpenMPI for > 900 cores: `ulimit -l 1048576`
  - crashes in OpenMPI for 12×80: add PBS flag `-lother=physmem`

18 Conclusions

- working with such codes and systems is hard!
  - 'bleeding edge' technology, variability effects, pushing memory limits, cumbersome and huge legacy code systems . . .
- efficient & accurate performance analysis methodology was developed
  - does need to be lightweight for accuracy (and also not to crash)
- a limited effect of tuning is possible with UM parameters
  - plenty of scope for worthwhile optimization, but only by the hard way!
  - load imbalance issues are particularly significant for the UM (may need a major overhaul!)
- significant effect of process and NUMA affinity on performance and its variability
- from this year on, a UM will consume a huge amount of compute cycles in Australia & elsewhere
  - and by people who don't (want to) know about performance!

19 Future Work

- analyze major subroutines' MPI and hardware event counter profiles
  - IPM 'hooks' have just been added to the UM internal profiler
  - preliminary analysis indicates loads/stores & L1$/L2$/L3$ misses completely accounts for computation time in most routines (0.5 – 2 GLOPs)
  - more accurately estimate fraction of MPI time due to load imbalance (especially in reductions and barriers)
- use this to estimate benefit of using hybrid OpenMP / MPI models
  - potential benefit in reducing nearest-neighbor 'HALO' communications and slightly faster collectives ($4\times$ fewer processes)
  - comparison of hybrid and pure MPI profiles should indicate routines with poor OpenMP parallelization
  - if worthwhile, could attempt to implement this

Thank You!

Questions? . . .