Performance Measurement and Analysis of Real Programs:

Case Study of a Large-Scale Atmosphere Simulation

by Peter Strazdins, Computer Systems Group

Overview:

- running time of real programs and parallel computing
- the Met Office Unified Model for weather and climate simulations
  - initiatives in Australia: BoM and ACCESS
  - overview of the software, including internal profiler module
- preliminary experiences with the UM_N320L70 benchmark
  - variability analysis
- profiling methodology
- scalability analysis & validation of methodology
- load imbalance and affinity issues
- conclusions

Running Time of Real Programs

- on a *real* computer, the running time of a simple computation like computing n =`length xs` could be modelled by $t(n) = a_0 + a_1 n$
  - computers have H/W support for measuring time ($\equiv$ no. of clock cycles) and other events; access these in a program via system calls
  - can determine by experiment: $a_0 = t(0)$, $a_1 = (t(1000) - a_0)/1000$
  - in practice, this might not always give an accurate prediction of $t(n)$. Why?
- large-scale simulations (e.g. 10^7 GFLOPs) on huge data sets (e.g. 10^2 GB) is an increasingly important use of computers
  - typically, very large and complex software is required
  - what kind of computer can run such a job in reasonable time?
  - writing efficient software is important - but first, how do we (accurately and efficiently) measure and understand its performance characteristics?

Parallel Processing with Clusters of Multicore Processors

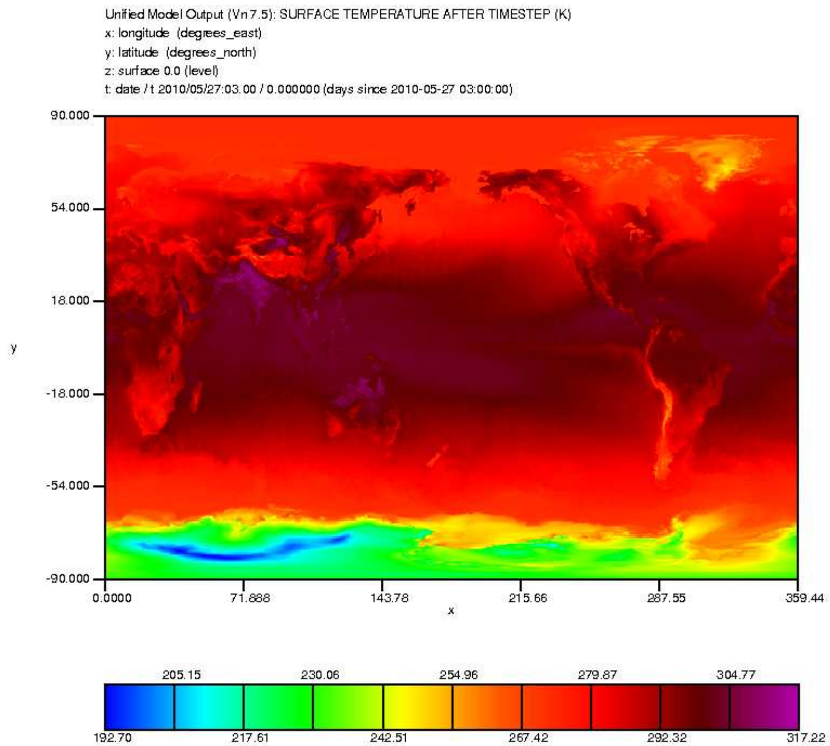
- with 10^3 modern processing units (cores), we can run an application with 10^7 GFLOPs in around 2 hours (assuming 1 GFLOPs/sec processing speed)
  - $\approx$ 200 memory systems of 1 GB could hold a 10^2 GB data set
  - e.g. Intel Nehalem computing 'node' has 4×2 cores;
    100 such nodes connected by a (high-speed) network would suffice
  - accessing large amounts of data presents a particular challenge: hence the memory hierarchy in each node
- the above assumes we can divide up the *whole* calculation *evenly* over all cores with *no overhead*!
- parallel processing however presents 3 main difficulties
  - load imbalance: hard to divide a computation evenly across each core
  - overhead: cores have to spend time communicating data to each other
  - start-up effects: certain parts of the calculation cannot be parallelized
- concept of parallel speedup: if $T(p)$ is execution time using p cores, we would ideally like a speedup of $S(p) = T(1)/T(p) \approx p$ (called 'good scalability')

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 ($192 \times 145 \times 3$) 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 . . .

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 $\approx$ 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 $P \times Q$ process grid



Unified Model Code Structure and Internal Profiler

- codes in Fortran-90 (mostly F77; $\approx$ 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 ($\approx$ 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

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
```

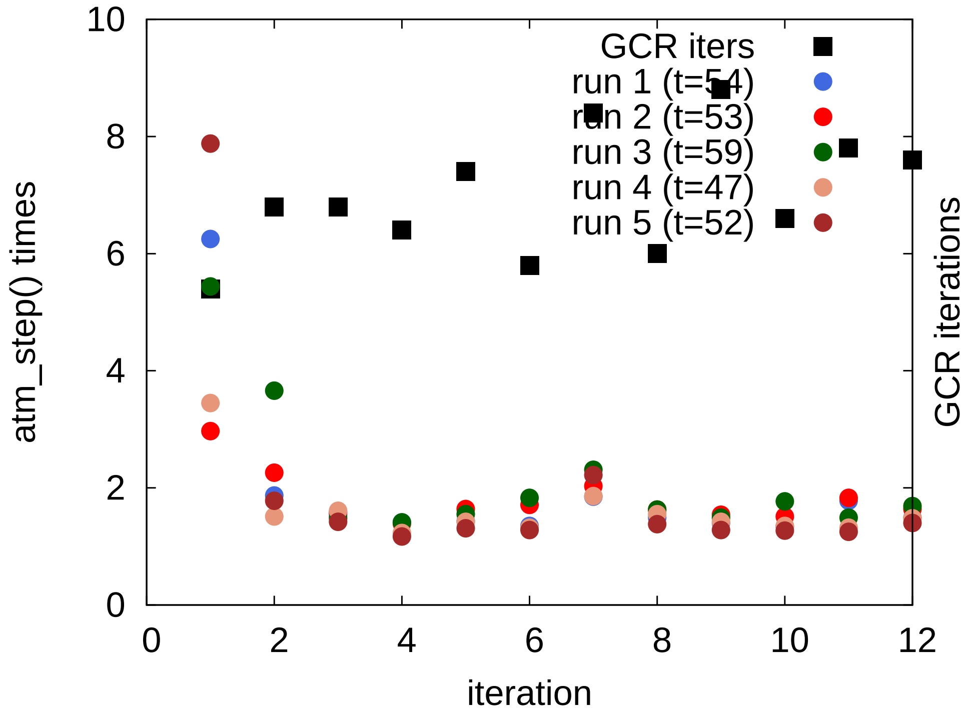
  - estimating time and memory resources accurately is important!
- the UM profiling project was (in 2010) allocated a few thousand CPU hours, max. core count 2048 . . .

Preliminary Experiences on the N320L70 Benchmark

- STASH output was disabled (to simplify)
- memory-related difficulties in running on <8 or >1536 cores, and on process grids like 60×24
- could not run benchmark under performance profiling tools like Sunstudio `collect` (memory?); had to rely on internal UM profiler
- variability in repeated run times of $\leq 50\%$ (bimodal; 'fast' runs $\leq 10\%$)
- preliminary subroutine profiling revealed 'inverse' scaling on:
  - `read_dump()` (25s plus)
  - `q_pos_ctl()`: many 'gather' and 'scatter' communications
    - $\Rightarrow$ Met Office subsequently supplied the PS24 patches

Preliminary Experiences: Inter-Iteration Variability

- inter-iteration variability within `atm_step()`
  - 1st step took 2–5$\times$ 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 ‘connections’



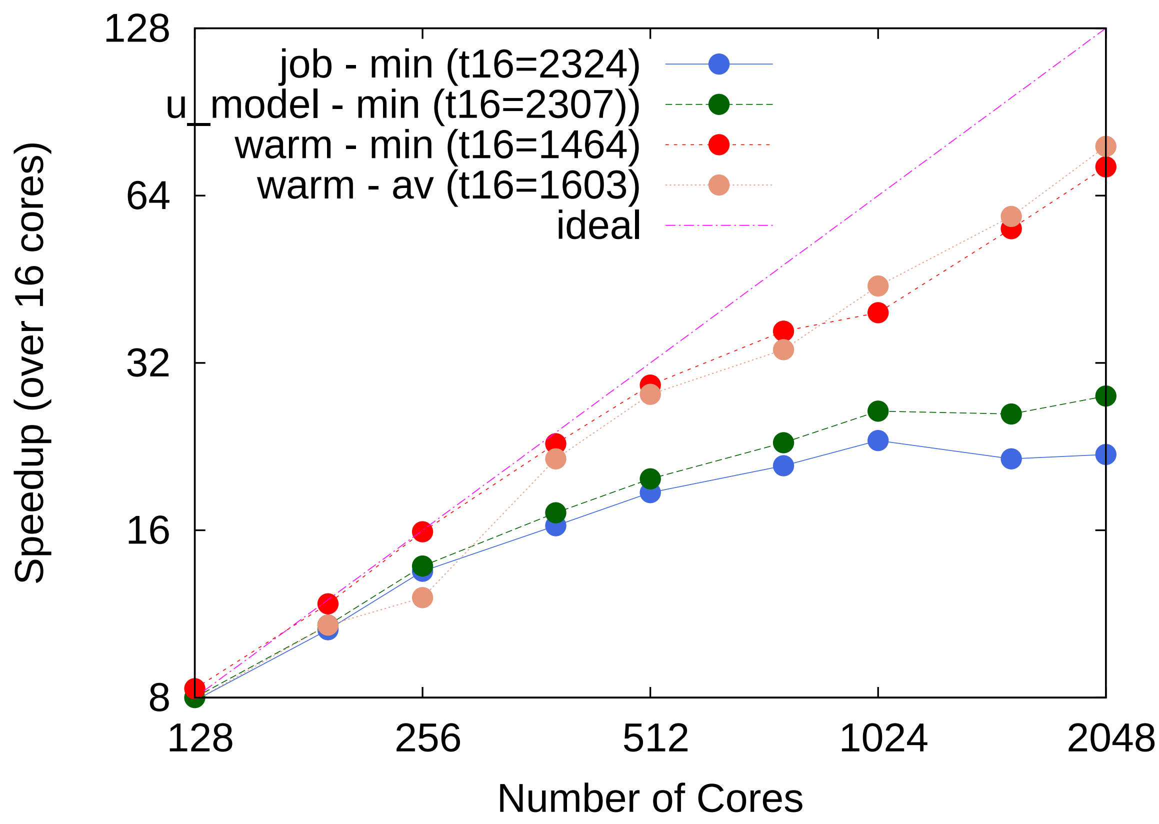
times & GCR iters. (scale: 0 to 50) per time step (each 12 mins sim. time)

Profiling Methodology

- based on these experiences, the following principles are desired:
  - infrastructure should have minimal effect on runtime (or memory)
  - consume minimal resources to accurately predict long-term sim.
  - 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.?)
  - when varying parameters, run each config. once before next rep.
  - 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

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

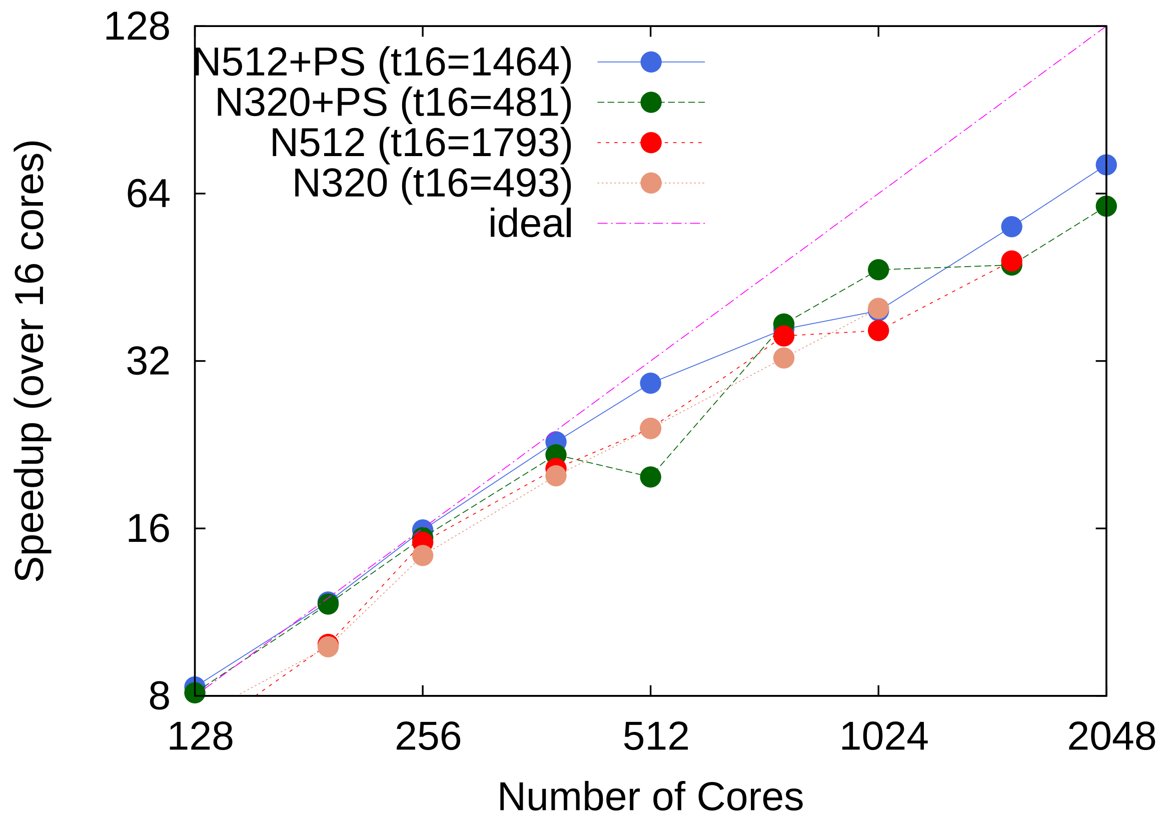
- process grids aspects between 1:1 and 1:2 chosen
- 't16' is time for 16 cores
- essentially linear scaling from 16 to 64 cores (slightly super-)
- surprisingly, average & minimum times show similar curves



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

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 eats 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



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), % 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

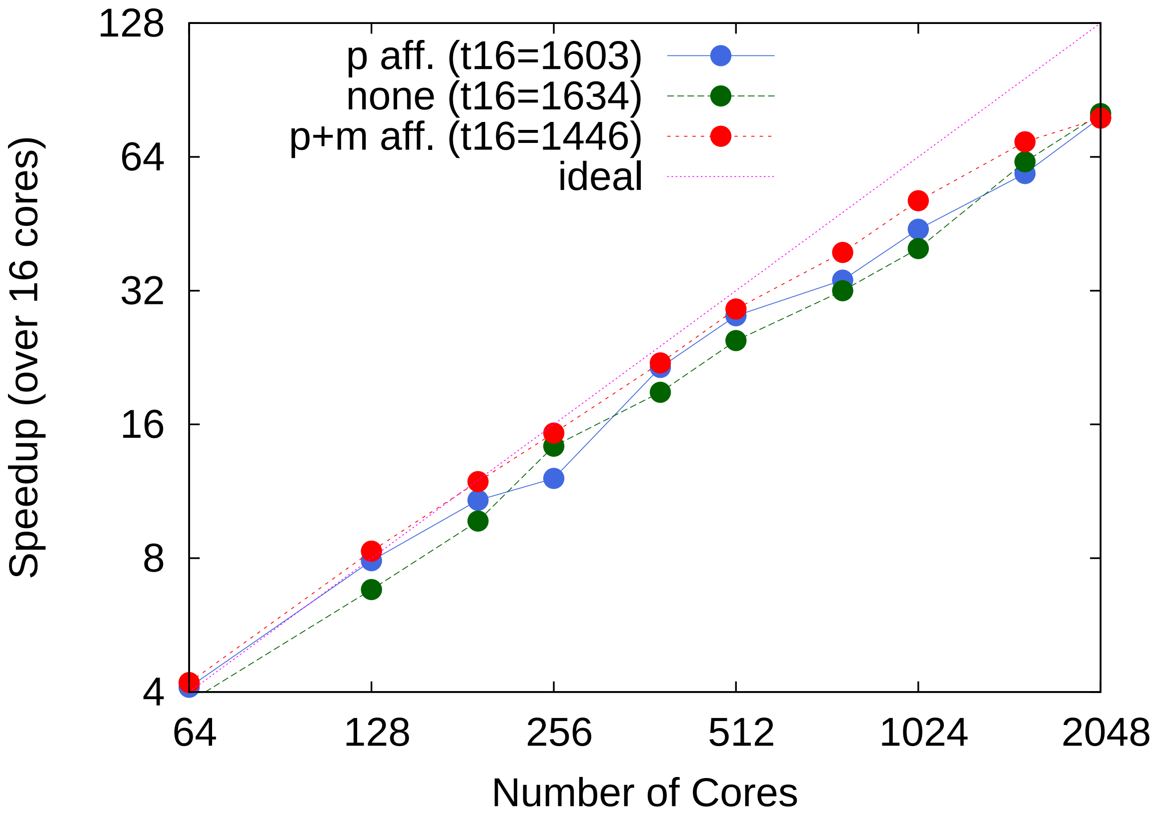
Load Imbalance Across Cores (16 × 24 process grid)

```
666666666666666666666666
999999999999999999999999
888999999999999999999999
888988889999999888999999
888888888889887777778888
887777778888777777777788
887777777788777766677778
666666666677666555556667
222222223333322200002222
333322223344222233333344
555555556666666555556655
111122221222222200112222
000011223333333223333322
222223334444444444444443
555555666666666666666666
555555555555555555555555
```

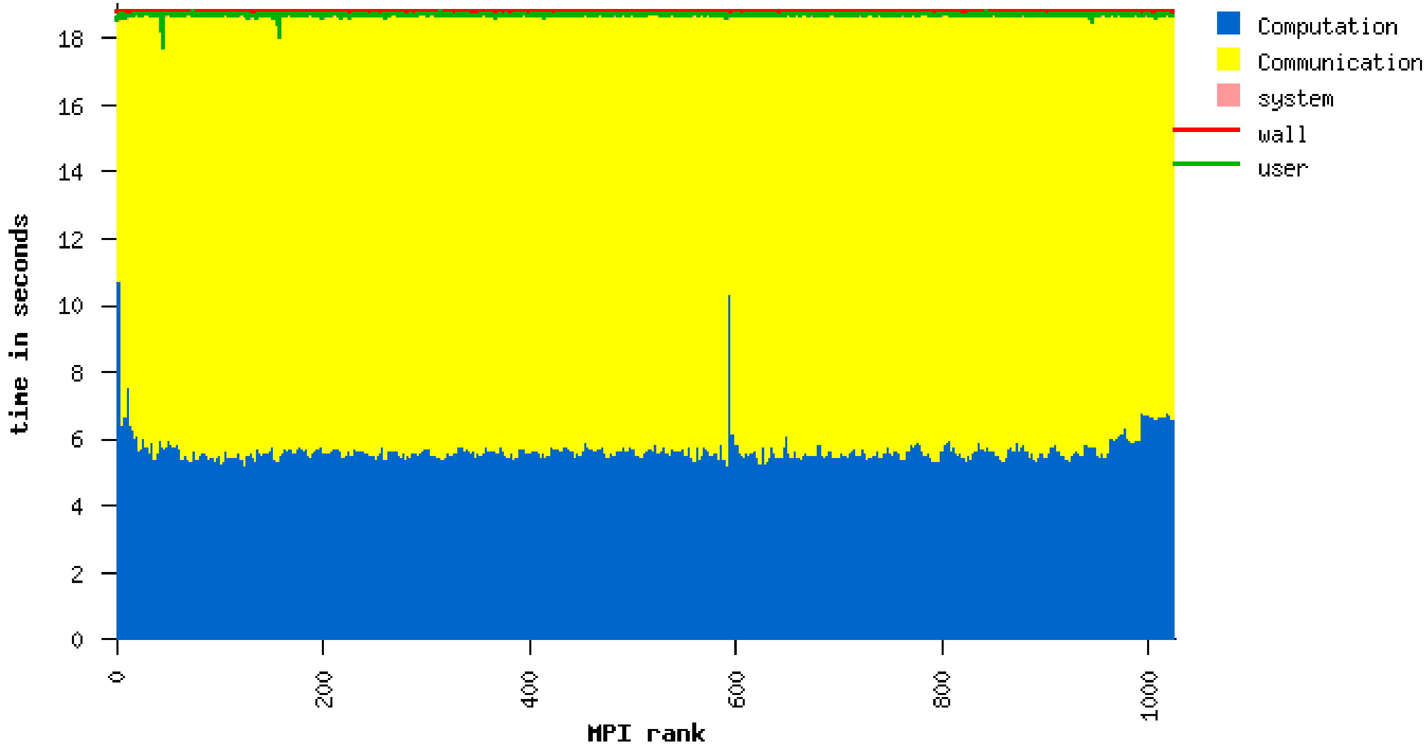
- over ‘warmed period’; ‘0’ = 0%, ‘9’ = 15% of run-time
- from time spent in barriers within UM internal profiler
  - large value indicates a lighter load
- clear indication of latitudinal and longitudinal variations
- difficult to avoid with fixed, regular data distribution

Results: Effect of Affinity on Scaling

- we can enforce process affinity (each of job's process is 'pinned' to a core on a node) and NUMA affinity (each process only access memory on nearest DRAM module)
- note differing values for t16!
- on X5570, local:remote memory access is 65:105 cycles
- indicates a significant amount of L3$ misses



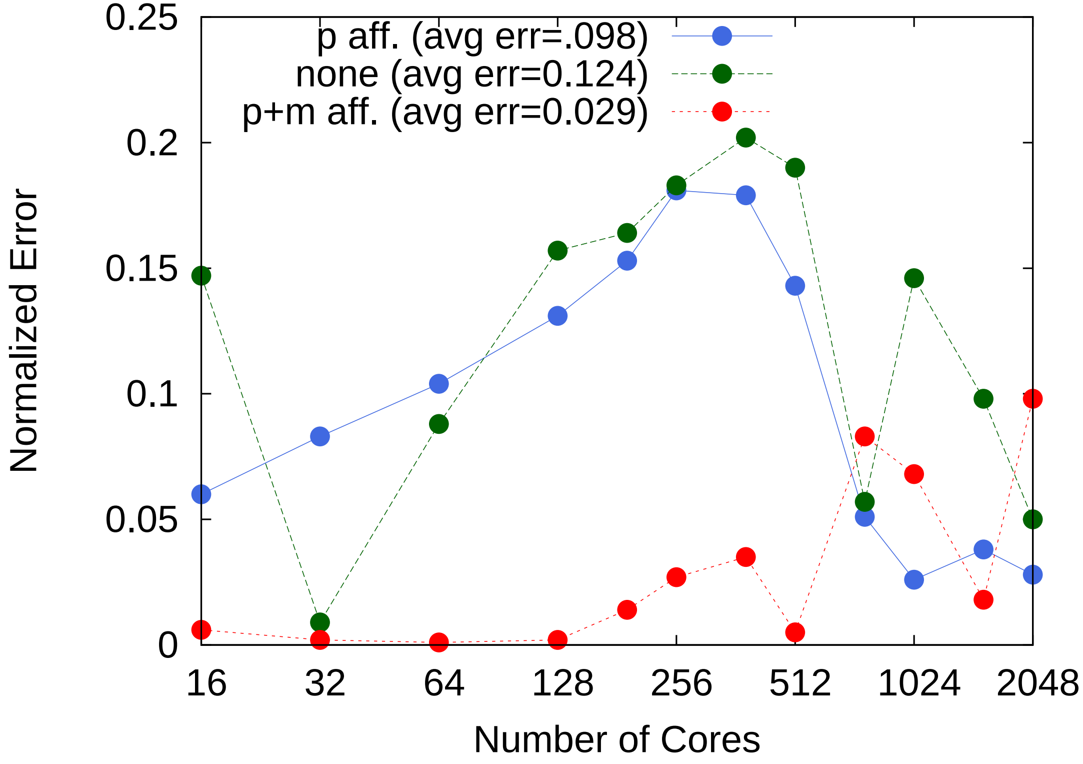
Effect of NUMA Affinity on Scaling (II)



- a performance analysis tool (IPM) gives a breakup of time
- without affinity, groups of 4 processes show spikes in computation times
  - i.e. always on the 1st ‘socket’ of a core!

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 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)

Evaluation of Profiling Methodology

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

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

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)
N320L702	237	174.9	14.9	233.6	iter. 134 (4.2s)

- 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

Conclusions

- working with such codes and systems is hard!
  - ‘bleeding edge’ technology, variability effects, pushing memory limits, cumbersome and huge legacy code systems . . .
- an efficient but accurate performance analysis methodology was developed
  - does need to be lightweight for accuracy (and also not to crash)
  - use of internal profiler gave useful information
  - scalability was reasonable (with PS24 patch), provided an appropriate timing was chosen
  - 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!