Performance Analysis of Large-scale Simulations on Supercomputers and Clouds

Peter Strazdins
Computer Systems Group,
Research School of Computer Science

Institute of Advanced Studies,
Technical University of Munich,
13 July 2012

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

THE AUSTRALIAN NATIONAL UNIVERSITY

1 Overview

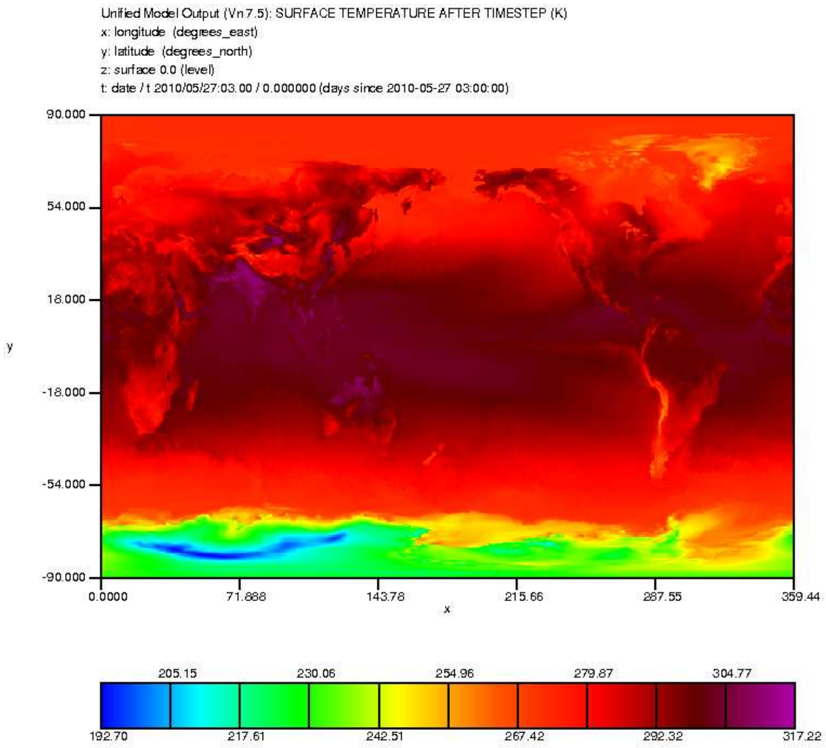
- the MetUM and Chaste projects
- supercomputers: the vayu cluster, the K supercomputer
- issues in large-scale, memory-intensive simulations
- techniques and tools for understanding scalability
  - identifying communication overhead & load imbalance, sections
  - tools: internal profilers, Integrated Performance Monitoring tool
- efficient performance analysis methodologies
  - accounting for variability of measurements; affinity effects
  - obtaining representative ‘sub-benchmarks’
- results on MetUM and Chaste on vayu
- comparison on the private and public clouds
  - motivations and setup
  - results: microbenchmarks and applications
- 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 recently used a N144L50 atmosphere grid
  - wished 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 ($192 \times 145 \times 38$)
- ACCESS project to run many (long) runs for IPCC
  - 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



THE AUSTRALIAN NATIONAL UNIVERSITY

4 Unified Model Code Structure and Internal Profiler

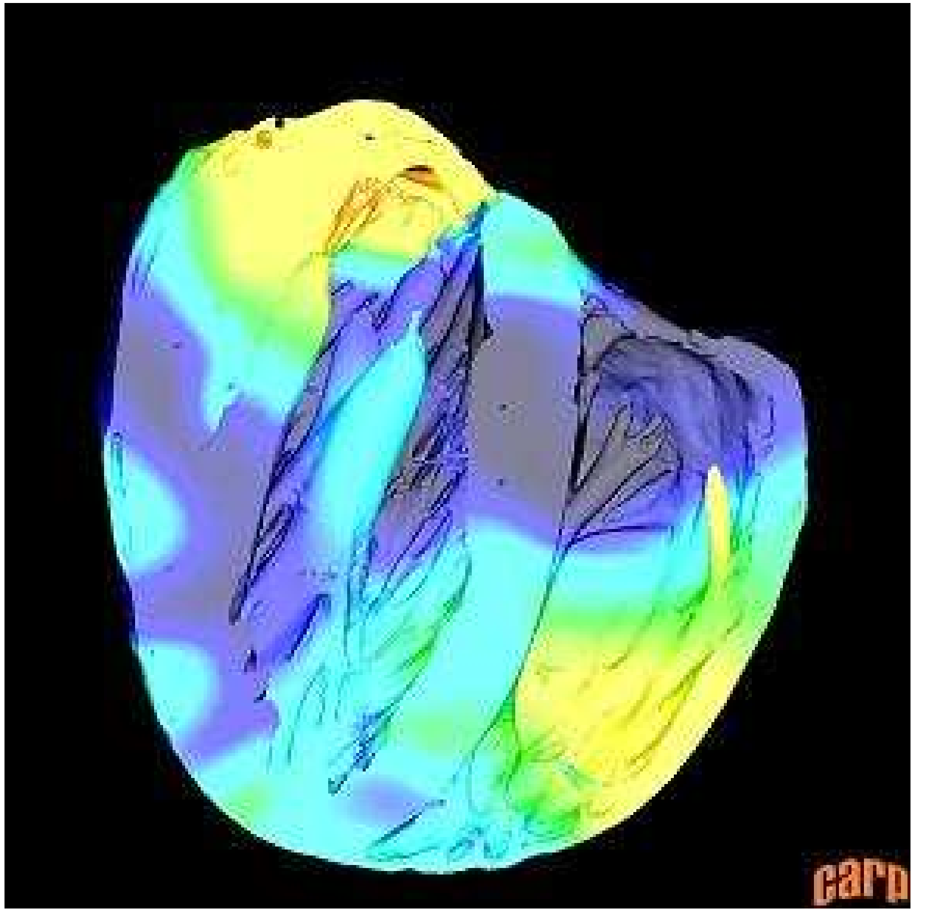
- codes in Fortran-90 (mostly F77; ≈ 900 KLOC) with cpp (include common blocks, commonly used parameter sub-lists, etc)
- main routine `u_model()`, reads dump file & repeatedly calls `atm_step()`
  - dominated by Helmholtz P-T solver (GCR on a tridia. linear system)
- internal profiling module can be 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 Chaste Cardiac Simulation Project

- Chaste: software infrastructure for modelling the electro-mechanical properties of the heart
- large system of C++ code, many dependencies
- also has internal profiler
- required resolution necessitates parallelization via MPI
- most computationally-intensive part is solution of a large sparse linear system once per timestep
- workload uses a high resolution rabbit heart (Oxford University) (2 $\times$ 1 GB files – 4 million nodes, 24 million elements)



6 The Vayu Cluster at the NCI National Facility

- 1492 nodes: two 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 & 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!
- allocation for our work was a few thousand CPU hours, max. core count 2048 . . .

7 Issues in Large-scale Memory-Intensive Simulations

- simulations of scientific interest run over many timesteps
  - ‘realistic’ benchmarks are resource-intensive: may be difficult on a ‘premiere facility’
  - variability of results problematic for accurate performance analysis
- resolution for state-of-the-art science pushes memory limits, even on a ‘premiere facility’
  - Chaste on 4M node mesh needs more memory than N320L70 atmosphere!
  - running MetUM on vayu required:
    - removing limit on stack size
    - redefining internal message buffer size (< 8 cores)
    - ‘pinning’ more (1 GB) physical memory for Infiniband (> 900 cores)
    - specifying memory limit to be physical (rather than virtual) (> 900 cores, wide process grid aspect ratios)

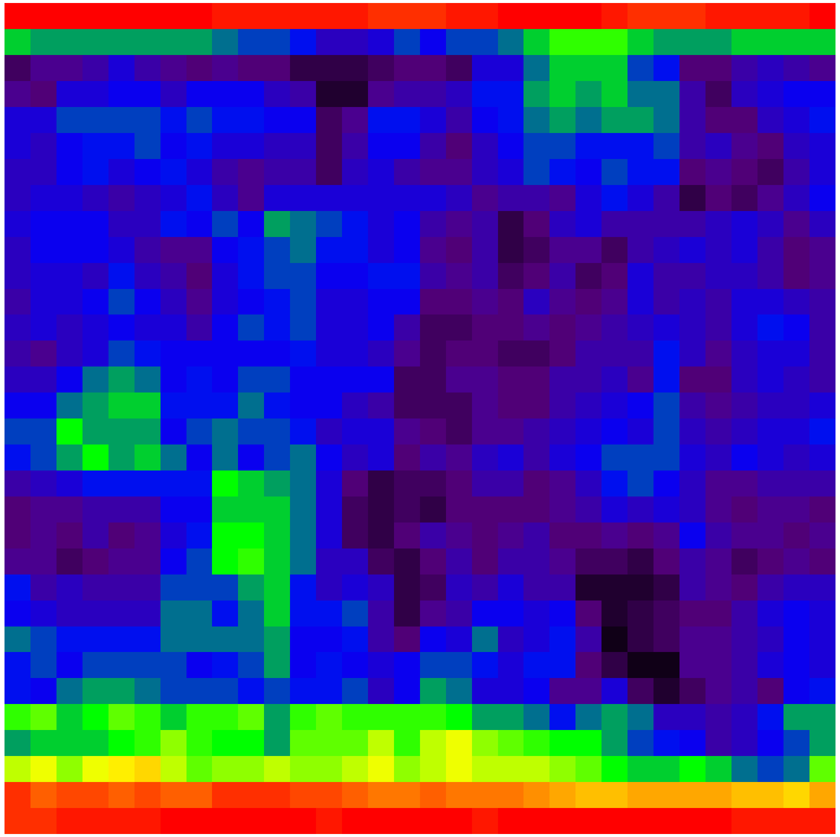
8 Techniques for Understanding Scalability: Communication Overhead and Load Imbalance

- measure time spent in communication library (MPI) separately
  - ideally, break-down per different communication operations (2 major categories: point-to-point and *collective*)
  - and (major categories of) buffer size
- load imbalance is more tricky to measure
  - differences in computation times across processes
  - *and/or* differences in times taken at barriers
    - i.e. the averaged time (over each process) spent in barriers, minus the estimated overhead of barriers (when perfectly balanced)
    - ideally, should do this for other collectives as well (e.g. small all-reduce operations)
  - note: problems are not solvable by a faster network! must be addressed at the application level

9 Case Sudy: Load Imbalance Across Cores in MetUM (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

  Note that this will be an underestimate!
- clear indication of especially latitudinal variations
- solution: less regular data distribution to take load off the polar areas



10 Tech. for Understanding Scalability: Section-Based Analysis

- consider latitudinal load imbalance in global atmosphere simulation
  - simulation at each timestep proceeds in a number of ‘sections’ polar filtering, thermal radiative transfers, convection, advection, etc
  - some require more work in high latitude, others in lower
- understanding of issues can be sharpened if considered separately
  - in particular, aggregate load imbalance is better estimated from weighted sum of per-section imbalances (triangle inequality)
- e.g. in the MetUM `atm_step()` routine:

```
If (Ltimer) Call timer ('PE_Helmholtz',3) ! 3: start non-inclusive timer
! code to call main PE_Helmholtz routine
...
If (Ltimer) Call timer ('PE_Helmholtz',4) ! 4: end non-inclusive timer
```

by calling at a barrier at the start of each (!) timer, can estimate per-section load imbalance by using variation in total times

11 Tools for Understanding Scalability

- desirable properties of any tool collecting scalability-related data
  - have minimum impact on computation time and memory footprint
  - communication vs. computation time breakup, load imbalance
    - provide further information indicating likely causes (i.e. hardware event counts: e.g. cache misses)
  - breakdown of these over component parts of the computation ,
- range from internal profilers to the heavy-weight SunStudio `collect`
  - the works! Pertinent sections derived automatically from the subroutine call-graph – combined with MPI & hardware event count profiling
- Integrated Performance Monitoring tool (IPM) in middle of the range
  - supports profiling of the MPI library and hardware event counters
  - support sections easily from internal profiler, e.g. from MetUM `timer()`:

```
if (timer_type == 3)  call mpi_pcontrol(+1, current_timer_name)
if (timer_type == 4)  call mpi_pcontrol(-1, current_timer_name)
```

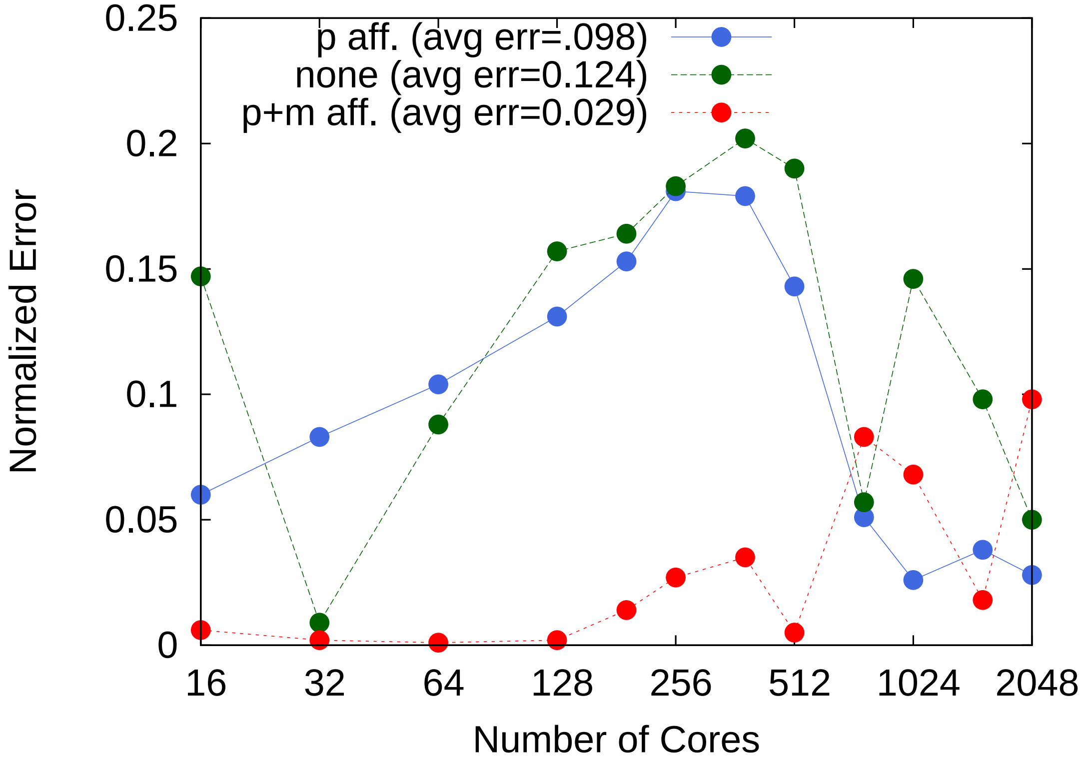
12 Methodologies: Minimizing Measurement Variability

- to analyze large-scale and long-running simulations over 1000's of cores, potentially need vast computing resources!
- compounded with fact that repeated experiments on a facility may give significant variability: many need to run many times!
- on a cluster such as vayu:
  - each node has 2 quad-core sockets; 8 processes given to each node

  the following effects were found to be important:
  - process affinity: once a node is assigned to a core, ensure that it stays there
  - NUMA affinity: memory used by a process must only be allocated on the socket of the core that it is bound to
- input/output requires (on vayu) access to the shared Lustre file system
  - exposes experiments to other users' using the file system
  - remains an open problem!

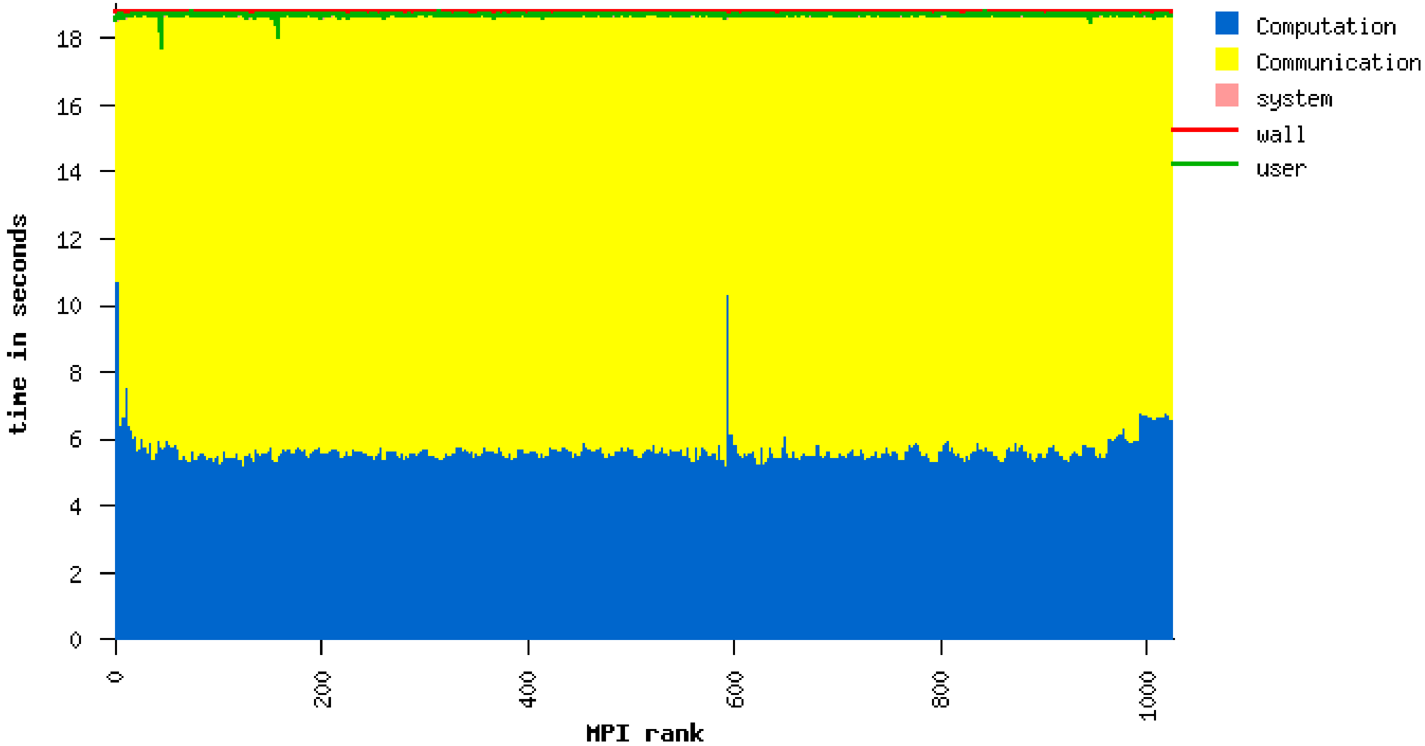
13 Case Study: Effect of Affinity on Variability of MetUM

- 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)

14 Effect of (no) NUMA Affinity – Exposed by IPM Profiles



- MetUM N320L70 (no STASH) output, 32×32 process grid
- no NUMA affinity: groups of 4 processes (e.g. socket 0) – spikes in compute times
- other runs: spikes occur on differing numbers & positions of nodes

15 Methodologies: Obtaining Representative Sub-Benchmarks

- standard 24 hour atmosphere benchmarks used by BoM are deemed to represent 10-day operational runs
- how much of this actually needs to be done for an accurate and representative performance analysis?
- basic idea: reduce number of iterations and select representative iterations for extrapolation for a larger simulation
  - works well when simulation's computational profile is *time-invariant*
- cardiac simulation is more problematic:
  - simulations of interest comprise applying an electric stimulation (e.g. 0.25 ms) and awaiting response over a longer interval (e.g. 30 ms)
  - depolarization are repolariztion wavefronts travel back and forth across the model over the response time
- in such cases, detailed performance analysis is required across all potentially different intervals to see if computational profiles (significantly) change

THE AUSTRALIAN NATIONAL UNIVERSITY

16 Validation of Sub-benchmark Methodology - MetUM

- methodology: run for 3 hours, taking the 12 iterations on hours 2–3 ('warmed period') as representative
- projected run time for 24 hour operational job (960 cores) is:

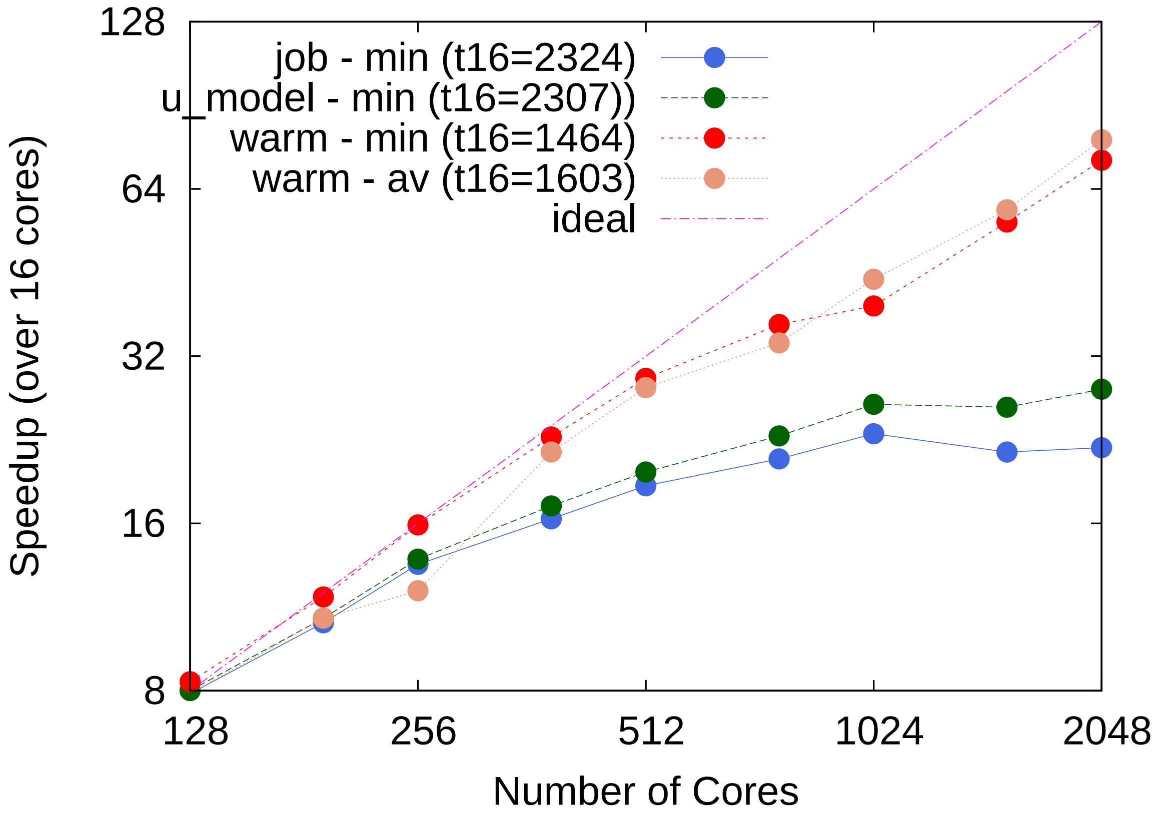
$$t' = (t - t_{2:24}) + 11.5 t_{2:3}$$

run	t	$t_{2:24}$	$t_{2:3}$	t'	anomalies
N512L70	527	39.12	6.5	524.3	–
N320L70 – 1	224	163.8	13.7	214.8	iter. 59 (7.9s)
N320L70 – 2	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%
- to reduce overhead, 1st hour was used to determine which sections were 'important' enough to perform barriers to estimate load imbalance
  - reduced number of such barriers by a factor of 10
  - measured profiler overhead (gather data, barriers) of $< 1\%$ of 'warmed period' times

17 Results: Which Time to Take? (N512L70 + PS24)

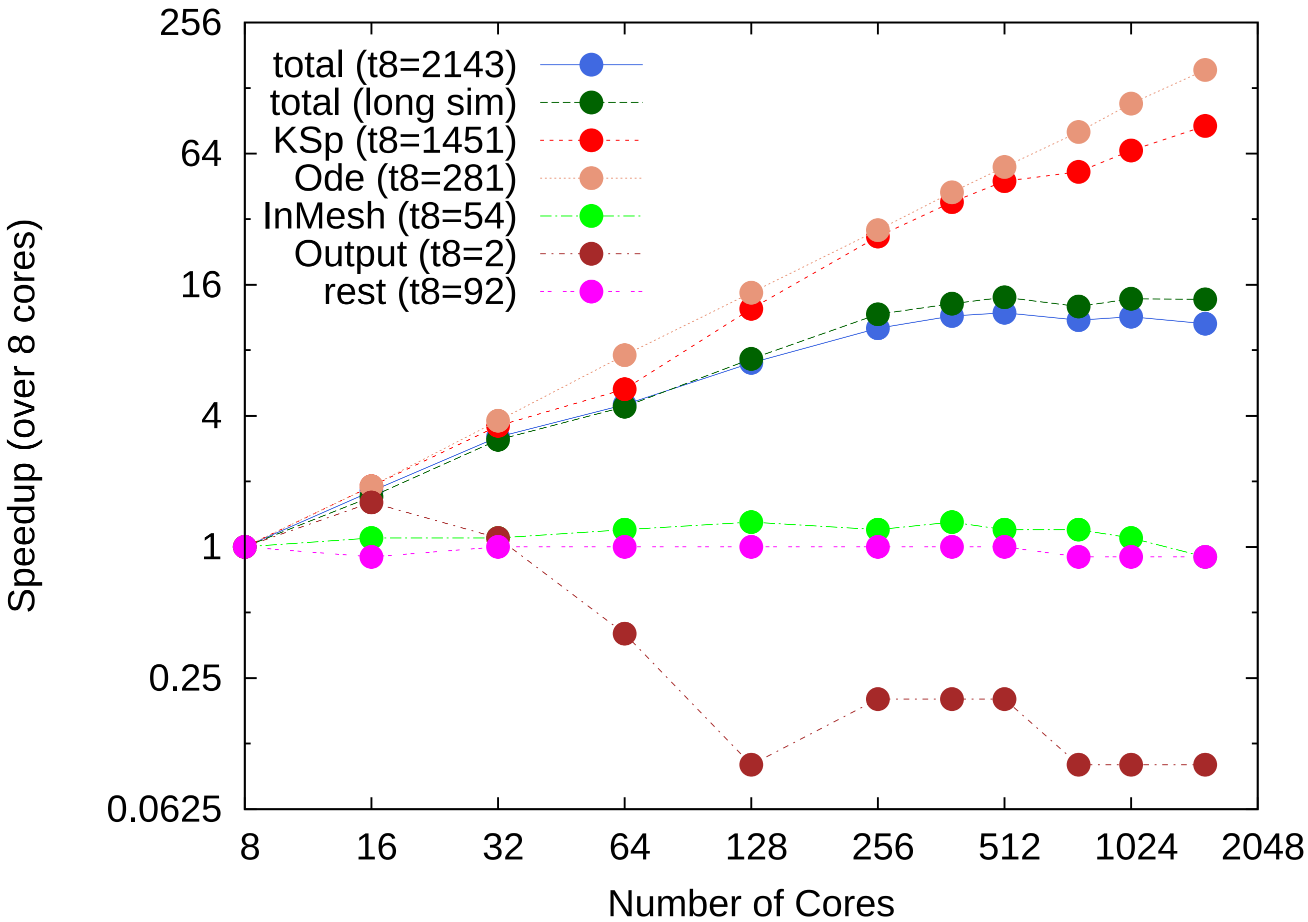
- process grids aspects between 1:1 and 1:2 chosen
- 't16' is time for 16 cores
- essentially linear scaling from 16 to 64 cores (slightly super-)
- surprisingly, average & minimum times show similar curves



- job time @ 1024 cores includes: pre-launch: 2s, launch processes: 4s, read 'namelist' files: 6s, `read_dump()`: 27s, cleanup: 1s
- message: 'warmed time' is a better predictor of the full simulation

18 Results: Section Analysis and Scaling Behavior (Chaste)

- 't8' is the time in seconds for 8 cores (due to memory constraints, could not run on less)



19 Results: Understanding Scalability Analysis via Profiling

- scalability of total time: max. 11.9 at 512 cores (from 8)
- scalability of ODE and KSp quite high; loss due to un-parallelized ‘rest’ and inversely scaling ’Output’ (using HDF5)
- execution time spent for 1024 cores

section	%t	%comm	main MPI	comments
rest	43	30	all-gather	25% time in I/O
Output	20	30	barrier	high load imbalance
InMesh	19	41	broadcast	
KSp	14	25	all-reduce (8b)	
Ode	1	18	all-reduce (4b)	
AssSys	0.8	0		slight imbalance; some I/O
AssRHS	0.5	7	waitany	high load imbalance

- note: IPM dilated the time spent in the KSp section by 50% and overall by 10%

20 Motivations for Using Clouds: a Cloud-bursting Supercomputer Facility

- supercomputing facilities provide access to state-of-the-art cluster computers
  - also provide comprehensive software stacks to support a diverse range of applications
- the supercomputing cluster is typically highly contended resource
  - users may be restricted to limited resources
  - may have long turnaround times
  - some workloads may not make good use of cluster
- $\Rightarrow$ may be better off using a private or even public cloud
  - requires easily replication of software stack on cloud resources
  - ideally, migration of jobs onto cloud would be transparent
  - recent frameworks can transparently profile HPC jobs for cloud suitability, e.g. ARRIVE-F, (and migrate VMs accordingly)

21 Experimental Cloud Setup: Systems

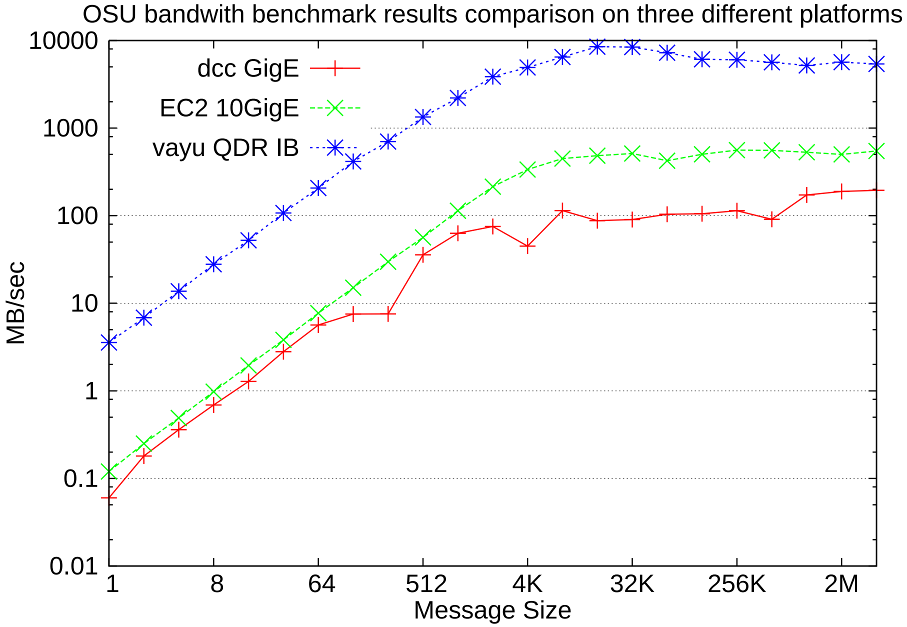
Platform		private cloud	public cloud	premiere cluster
Platform		DCC	EC2	Vayu
# of Nodes		8	4	1492
CPU	Model	Intel Xeon E5520	Intel Xeon X5570	Intel Xeon X5570
	Clock Spd	2.27GHz	2.93GHz	2.93GHz
	#Cores	8 (2 slots)	8×2	8 (2 slots)
	L2 Cache	8MB (shared)	8MB (shared)	8MB (shared)
Memory per node		40GB	20GB	24GB
Operating System		Centos 5.7	CentOS 5.7	CentOS 5.7
Virtualization		VMware ESX 4.0	Xen	–
File System		NFS	NFS	Lustre
Interconnect		1 GigE (dual)	10 GigE	QDR IB

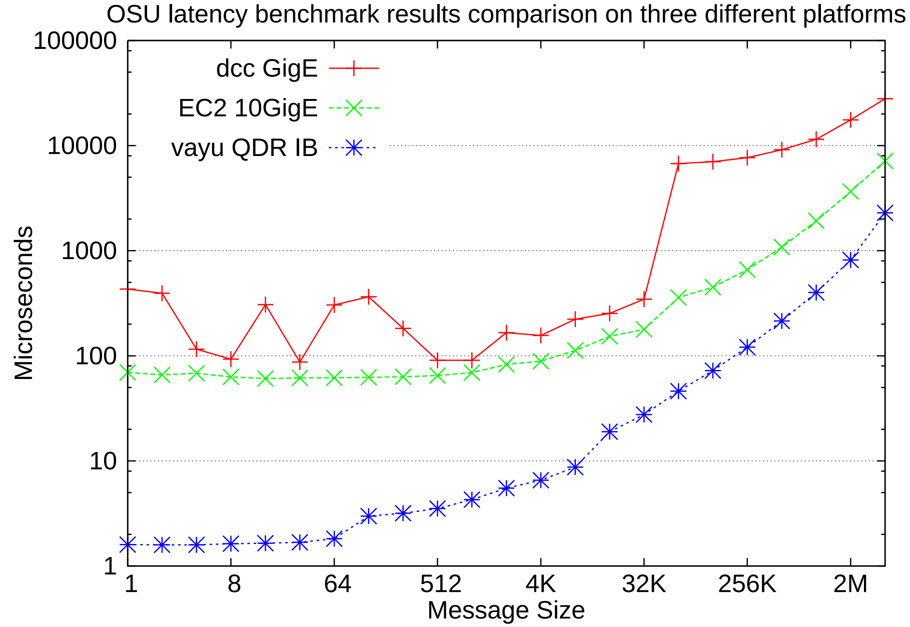
- DCC: 1 VM/node; filesystems mounted via external cluster via two QLogic channel fibre HBAs
- vayu: QDR IB used for both compute and storage

22 Cloud Setup: Software

- EC2: StarCluster instance to automate the build, configuration & management of HPC compute nodes
- vayu `/apps` directory: system-wide compilers, libraries, and application codes
  - user environment is configured via `module` package
- rsync `/apps` and user home/project directories onto the VM to replicate stack
  - minimizes interference of existing stack on the clouds
  - only occasionally needed to recompile for the clouds
- benchmarking software
  - OSU MPI communication micro-benchmarks: bandwidth and latency
  - NAS Parallel Benchmark MPI suite 3.3, class B
    - 5 kernels & 3 pseudo-applications derived from CFD applications

23 Cloud Results: Communication Micro-benchmarks



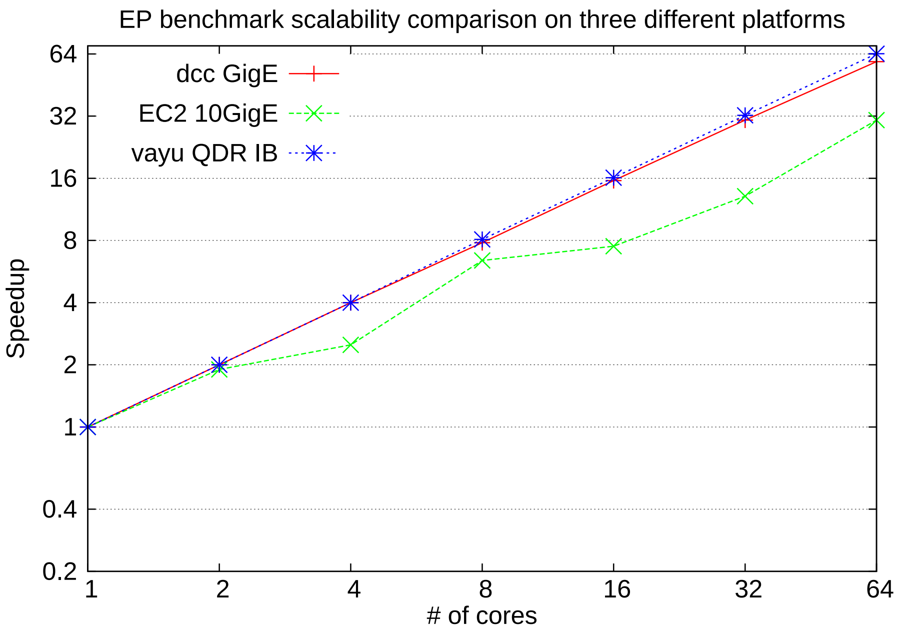


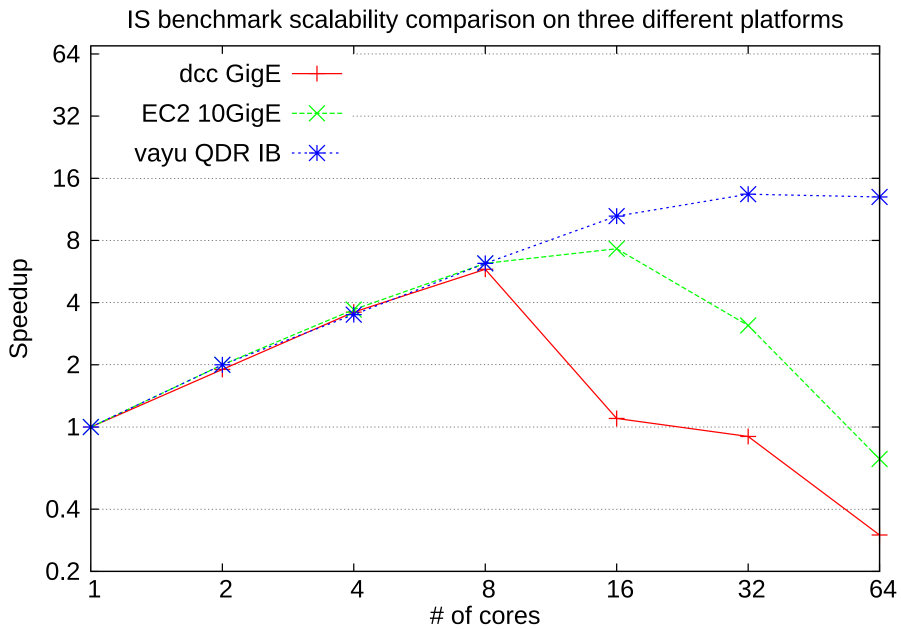
OSU MPI bandwidth tests (MB/s vs message size)

OSU MPI latency tests (time (μs) vs message size)

- trends as expected per theoretical specifications: more than one order of magnitude better performance on vayu
- fluctuations on DCC suspected from CPU scheduling by hypervisor

24 Cloud Results: NAS Parallel Benchmarks (I)



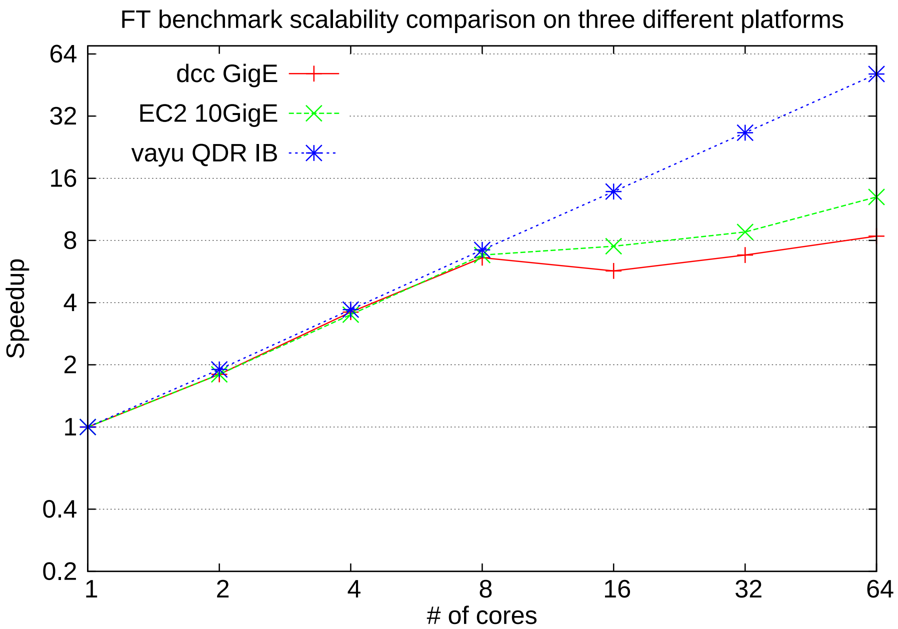


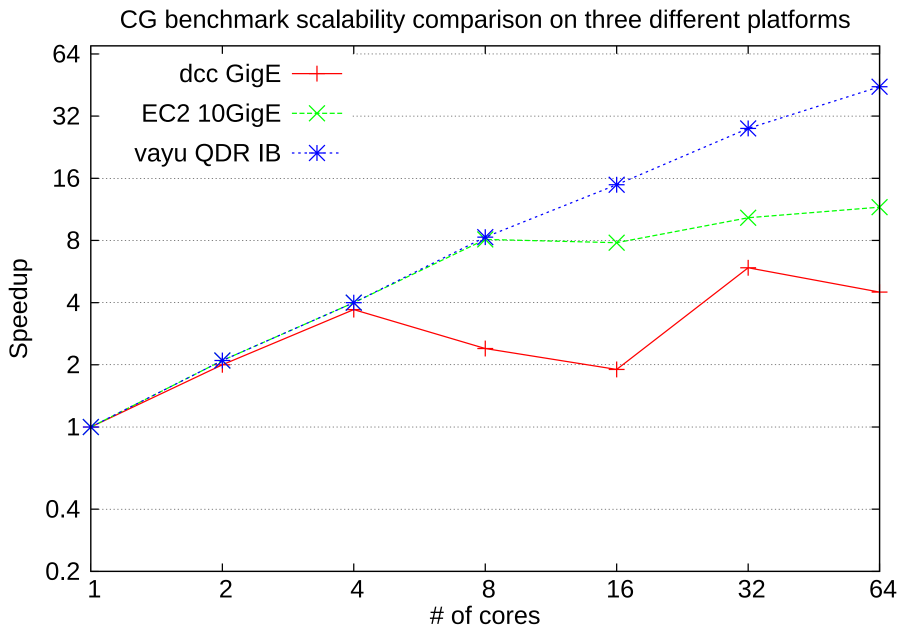
EP.B speedup, 1 to 64 cores IS.B speedup, 1 to 64 cores

- EC2 fluctuations for EP.B suspected due to jitter (CPU scheduling and hyperthreading
- IS shows the poorest scaling of all benchmarks:
  - IPM profiling shows % communication at 64 cores is 98% (DCC), 85% (DCC) and 68% (vayu)

THE AUSTRALIAN NATIONAL UNIVERSITY

25 Cloud Results: NAS Parallel Benchmarks (II)



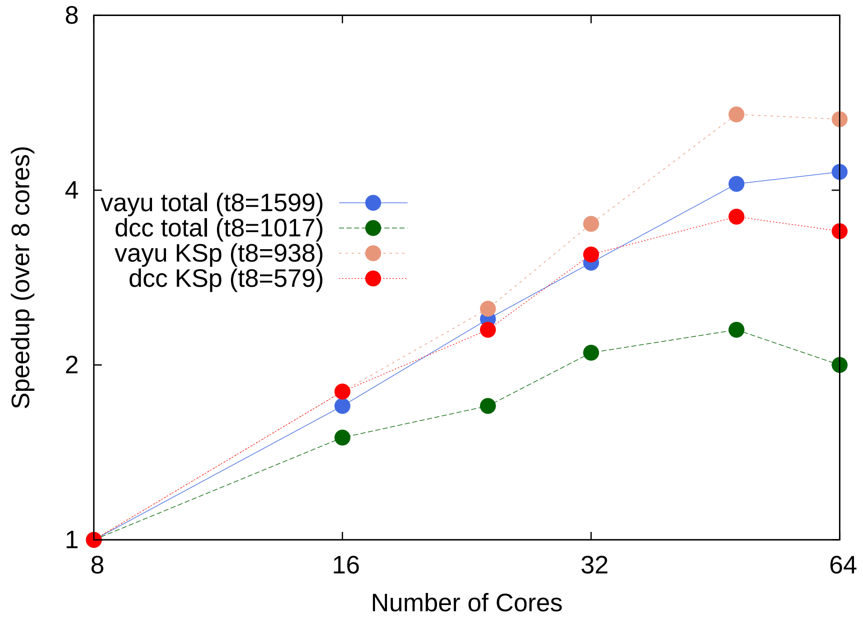


FT.B speedup, 1 to 64 cores CG.B speedup, 1 to 64 cores

- CG.B: IPM profiling shows % communication at 64 cores is 90% (DCC), 58% (DCC) and 22% (vayu)
- drop-offs on clouds occur when intra-node communication is required
- BT.B, MG.B, SP.B and LU.B showed similar scaling to FT.B
- single core performance consistently 20% (30%) faster on EC2 (vayu)

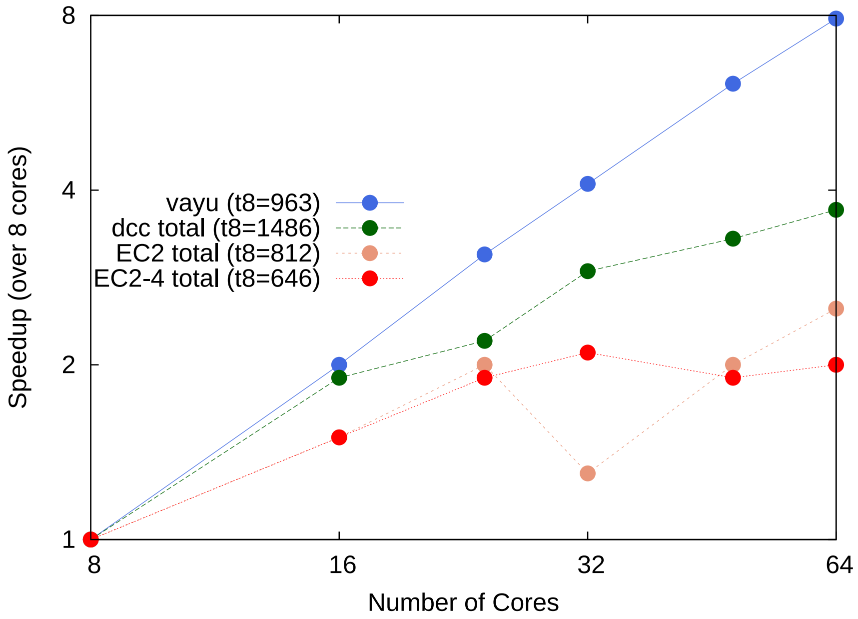
26 Cloud Results: Chaste Cardiac Simulation

- (results not available on EC2 due to complex dependencies)
- scaling of the KSp linear solver determines overall trends
  - note: benchmark scales to 1024 cores on vayu
- input mesh: 1.4× faster on vayu (8 cores), scaled the same on both
- output: 2.6× faster on vayu (8 cores) but scaled better on dcc



- @ 32 cores, 48% vs 11% of time spent in communication on dcc vs vayu
  - 13× more spent in KSp solver on dcc (large numbers of collectives)
- IPM profiles also indicated a greater degree and a higher irregularity of load imbalance on DCC
- ⇒ dcc performance hurt by high message latencies & jitter

27 Cloud Results: MetUM Global Atmosphere Simulation



	Vayu	DCC	EC2	EC2-4
time(s)	303	624	770	380
r_{comp}	1.0	1.37	2.39	1.17
r_{comm}	1.0	6.71	3.53	1.61
%comm	13	42	18	18
%imbal	13	4	18	19
I/O (s)	4.5	37.8	9.1	7.6

Details at 32 cores (EC2-4: 4 nodes used, uniformly 2$\times$ faster)

- overall load imbalance least on dcc, but generally higher & more irregular across individual sections (NUMA effects)
- EC-2 shows similar imbalance and communication profiles to vayu
- dcc spent most time in communication, particularly in sections where there were large numbers of collectives
- read-only I/O section: dcc much slower to vayu, EC2 similar, to vayu

THE AUSTRALIAN NATIONAL UNIVERSITY

28 Conclusions

- performance of large-scale memory-intensive simulations
  - working with such codes and systems is hard!
  - many techniques and suitably lightweight tools needs to be applied in order to understand it
    - need to understand *what* (is the issue), then *where*, and then *why*
    - it is however possible to get useful insights, even for complex applications
- efficient methodologies need to be developed – non-trivial unless computational profile is time-invariant!
- largely successful in creating x86-64 binaries on HPC system & replicating all software dependencies into the VMs on clouds
- communication bound applications were disadvantaged on the virtualized platforms
  - large numbers of short messages were especially problematic
- over-subscription & hidden effects (e.g. NUMA) also also affected clouds

Future Work

- extend performance analysis to other large-scale applications (e.g. ANUGA, GENE)
  - methods to reduce reductions and other global operations become more attractive as we scale to larger numbers of cores
- use metrics from the ARRIVE-F framework to assess candidate workloads for private/public science clouds
- using StarCluster, cloud burst onto OpenStack based resources locally & externally

Acknowledgements

- MetUM in collaboration with Tim Pugh (BoM) and others
- Chaste work in collaboration with Markus Hegland and James Southern (FLE)
- Fujitsu Laboratories Europe for supporting work on Chaste
- cloud work in collaboration with Jie Cai, Muhammad Atif and Joseph Antony (NCI NF)
- NCI NF staff technical support
- NCI NF for time on vayu cluster

Questions???