

Performance Evaluation / Modelling and Infrastructure Support in Heterogeneous Clusters

Peter Strazdins,
(and the Jabberwocky Project team),
Department of Computer Science,
The Australian National University

seminar at School of IT, Deakin, 08 November 2006

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



1 Overview

- the Jabberwocky Heterogeneous Cluster Project
 - overview
 - sub-projects:
 - Grid-Enabled Clusters
 - Performance Modelling and Prediction
- communication performance of SMP clusters with multiple GigE interfaces
 - motivation
 - background: virtualization under Xen and TCP/IP under Linux SMP
 - configurations using channel bonding and independent pathways
 - bypass of the Xen virtual machine monitor
 - results
 - conclusions and future work

2 The Jabberwocky Heterogeneous Cluster Project

- an umbrella project between Alexander Technology and CS dept. at ANU (also Adelaide), began March 2006
- people:
 - Richard Alexander, David Barr, Brendan Howe (AT)
 - Paul Coddington, (Adelaide), Alistair Rendell, Peter Strazdins
 - 2 APAI_IT PhD students & 1 postdoc (under recruitment!)
 - formerly: Tony Breeds (Masters student - Performance Evaluation Methodologies)
- hardware: heterogeneous cluster (cluster of clusters) consisting of:
 - BanderSnatch: head node for SlithyToves mini-cluster
 - VorpallBlades: 4 (\times 2-4 CPU) Opterons with GigaE and Pathscale Infiniband
 - SlithyToves: 4 (\times 1-2 CPU) Opterons with GigE and Quadrics
 - TulgeyWood: remote node imaging file repository

and its growing like the mythical Jabberwocky . . .

3 The Grid-Enabled Clusters Project

- part of Linkage project *Next Generation Grid Enabled Cluster Computers: Performance Optimisation for e-Science*
(Alistair Rendell & Peter Strazdins, 12/06 – 12/09)
- scenario: a ‘compute center’ serving several users (with scientific applications)
 - heterogeneity is desirable, inevitable!
- aim: develop intelligent runtime environments
 - select where to execute a job, based on cost / benefit, current load
- approach:
 - develop performance-monitoring and characterization infrastructure into runtime environments
 - sophisticated scheduling strategies
 - use of virtualization by Xen to aid migration and operating system customization

4 The Performance Modelling and Prediction Project

- part of Linkage project *Efficient and Accurate Performance Modelling and Prediction of Cluster Computers*
(Peter Strazdins & Paul Coddington, 12/06 – 12/09)
- scenario: cluster company wishing to make the most cost-effective design for medium to large- scale clusters for a client
 - client has particular set of applications (available as source or object)
 - huge range of possibilities: nodes (types and # CPUs), network interfaces (types and # / node), switches
 - can experiment with only small-scale clusters (e.g. the Jabberwocky)

5 Performance Modelling and Prediction: Background

- the MPIBench / PEVPM methodology (Grove and Coddington)
- MPIBench: measure execution times of large samples of MPI calls
 - variations in times captures effects such as contention
 - fit times to probability distribution (e.g. Weibull)
 - probabilistic modelling claimed to be more accurate than using the averaged time
- PEVPM: Performance Emulating Parallel Virtual Machine
 - software emulator for MPI programs
 - 'estimates' time of computational sections
 - uses probabilistic functions to model MPI call times
 - ✓ made highly accurate predictions
 - ✗ only on very simple applications, times for computational sections added by hand using annotations

6 Performance Modelling and Prediction: New Work

- *efficient* derivation of complex application performance models (ANU):
 - utilize the ‘trace’ of MPI calls (call type, data size, and timestamp)
 - (virtualized) timestamps give times of computational sections
 - MPIBench-derived models give the MPI call times
 - challenge: extend the model derived from a cluster of size p to a larger cluster of size $p' \dots$
- develop more advanced models for the communication network (Adelaide)
 - again, use probabilistic modelling
 - capture more subtle effects such as the degree of CPU involvement
 - challenge: as above
- validation: using the large clusters: Bunyip and Hydra

7 Communication Performance of SMP Clusters with Multiple GigE Interfaces

- motivation:
 - low-end SMPs (esp. with multi-cores) are highly cost-effective for computational power
 - communication performance of COTS clusters has not kept up
 - many COTS motherboards support multiple I/O connections
 - e.g. IWILL DK8-HTX (Opteron) has three PCI-X / PCI buses and 14 device slots
 - adding network interface cards is a t small relative cost
 - could these be used to balance a cluster's communication performance
 - (Gigabit) Ethernet (GigE) is the most widely used interface:
highly cost-effective
- more details on this work in a paper (with Richard Alexander and David Barr) for the Xen in HPC Workshop in Dec

8 Background: Virtualization under Xen

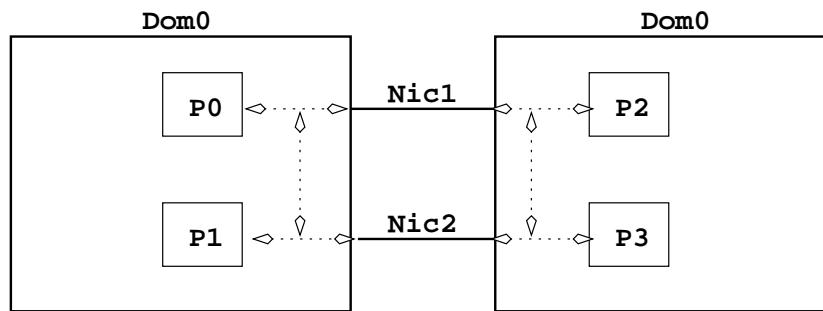
- virtualization offers many advantages (greater encapsulation), but typically comes at a significant performance penalty
- (OS) virtualization: a hypervisor (or virtual machine monitor, VMM) sits at a higher privilege level to the virtualized ('guest') OS
 - direct access to devices, page tables, privileged registers is by calls to the VMM
- Xen uses para-virtualization: the parts of the (Linux) kernel dealing with the above must be modified ('XenoLinux')
 - offers both potentially high performance and functionality
- requires one special guest OS (domain0, the 'driver domain'):
 - manages (configures, creates, destroys) a number of guest OSs (guest VMs)
 - external communication occurs through a virtual interface:
 - data is transferred by pseudo device drivers to domain0

9 Background: TCP/IP Stack Processing under Linux SMP

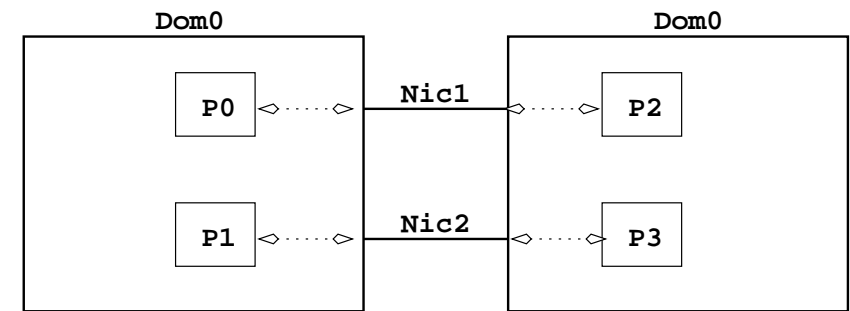
- Linux SMP 2.6 has sophisticated TCP/IP stack processing
- uses 2 kinds of locks: connection-related and socket-related (more frequently accessed)
- 2 broad strategies:
 - message-parallel: ||ize the processing (of different segments) of a single transmission
 - connection-parallel: ||ize processing of messages using different sockets (generally superior performance)
- recent studies have shown parallelization incurs some overhead (lock manipulation and cache pollution)
 - explains the generally poor performance of channel bonding

10 Communication Configurations with Multiple Interfaces

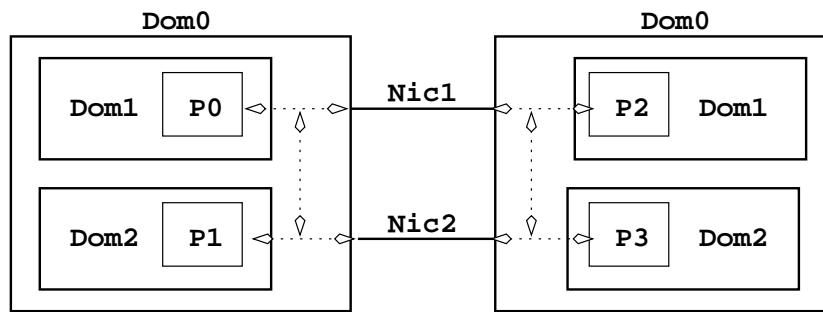
- consider aggregate bandwidth between MPI process pairs 0 & 2, 1 & 3:



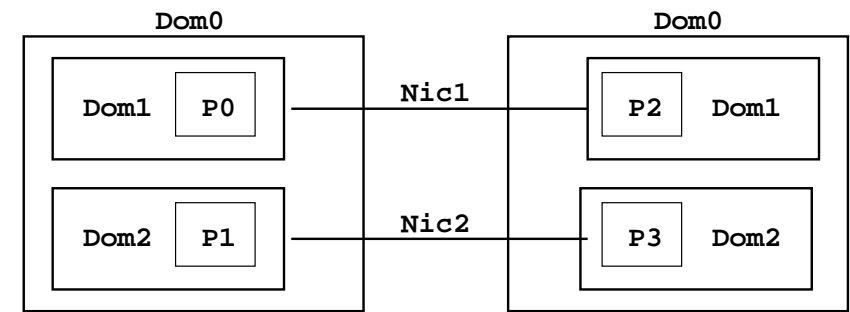
(a) driver.bond.2p



(b) driver.indep.2p



(c) guest.bond.2p



(d) guest-byp.indep.2p

- similarly have 1-pair (.1p - baseline) and 4-pair (.1p) configurations

11 Communication Configs: Characteristics & Implementation

- parallelization of TCP/IP stack: `*.bond: socket-level, driver.indep.*: connection level, guest-byp.indep: full`
- communication bypass of the Xen VMM is achieved by modifying the guest VM's configuration file:
 - unset the `vif` variable (no virtual interfaces)
 - setting the `pci` variable to the desired bus/slot, e.g. `pci = ['03,04,0']`
 - the guest VM is bound to a particular CPU
- in `driver.indep`, the `route add` command is used to send messages to a particular process through the respective NIC
 - in the test programs, `sched_setaffinity()` used to bind processes to their respective CPUs
- in `indep.*`, interrupt requests for each NIC are bound to each CPU
 - with 4 NICs, found there were an insufficient IRQs #;
had to implement filesystems on a network bootable RAM disk!

12 Experimental Setup

- hardware:
 - 2 node cluster of dual SMP dual-core 2.2 GHz AMD Opterons
 - IWILL DK8-HTX_815 motherboard with 800 MB/s HyperTransport links
 - eth1: one of the in-built dual Intel GigE interfaces (PCI bus 3)
 - eth2, eth3: 82546GB dual Pro/1000 MT NIC (some TCP/IP of-fload, PCI bus 3)
 - eth4: 82541PI Pro/1000 MT NIC (PCI bus 1)
 - from `lspci`: all buses in 32-bit mode @ 66 MHz (264 MB/s)
- software:
 - use MPICH-2 MPI from OSUbench; MPI configured to use only eth1 to eth4
 - use multiple-pair version of OSUbench's latency (ping-pong), uni-directional and bi-directional bandwidth benchmarks

13 Results for Communication Configurations: Single Pair

- for `eth1`, uni-b/w 110 MB/s (c.f. GigE max 125 1MB/s),
except `guest.bond`: only 50MB/s !
- bi-b/w: 135 MB/s; `eth2/eth3`: 155 MB/s, `eth4`: 90 MB/s
- inter-node latency tests:

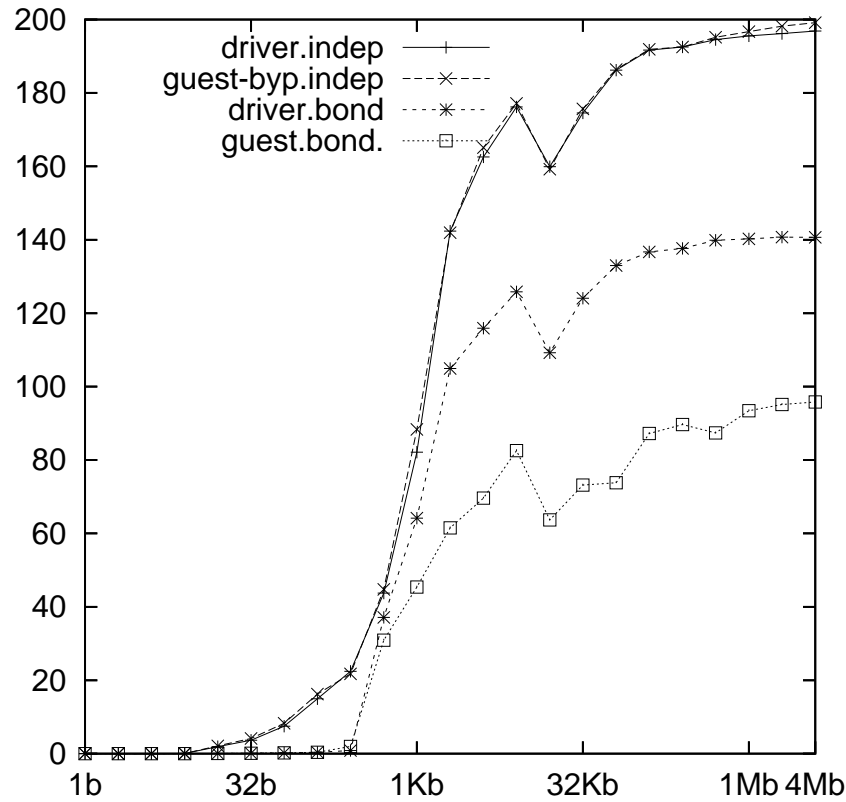
	<code>*.indep</code>	<code>driver.bond</code>	<code>guest.bond</code>
time at 1 B (μs)	87	91	106
B/W at 4 MB (MB/s)	107	105	51

- intra-node latency tests indicate `guest*.*` are using virtual interfaces ...

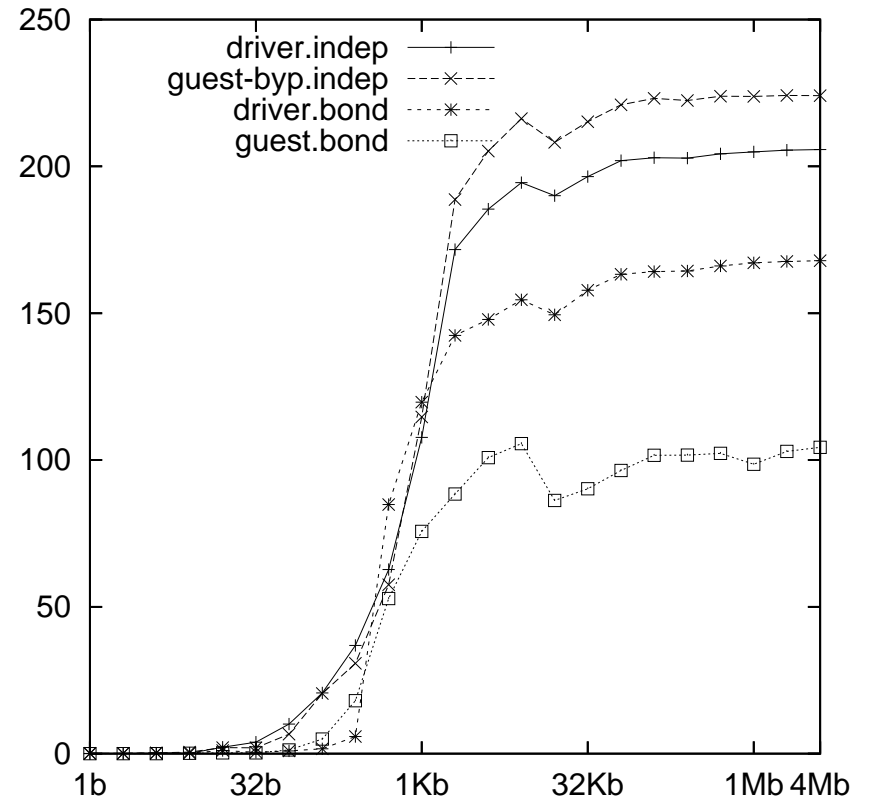
	<code>driver.*</code>	<code>guest*.*</code>
time at 1 B (μs)	18	530
B/W (MB/s) at 4 MB	30	50

- native Linux performance essentially the same as `driver.indep`

14 Results: Bandwidth for 2-pair Configurations

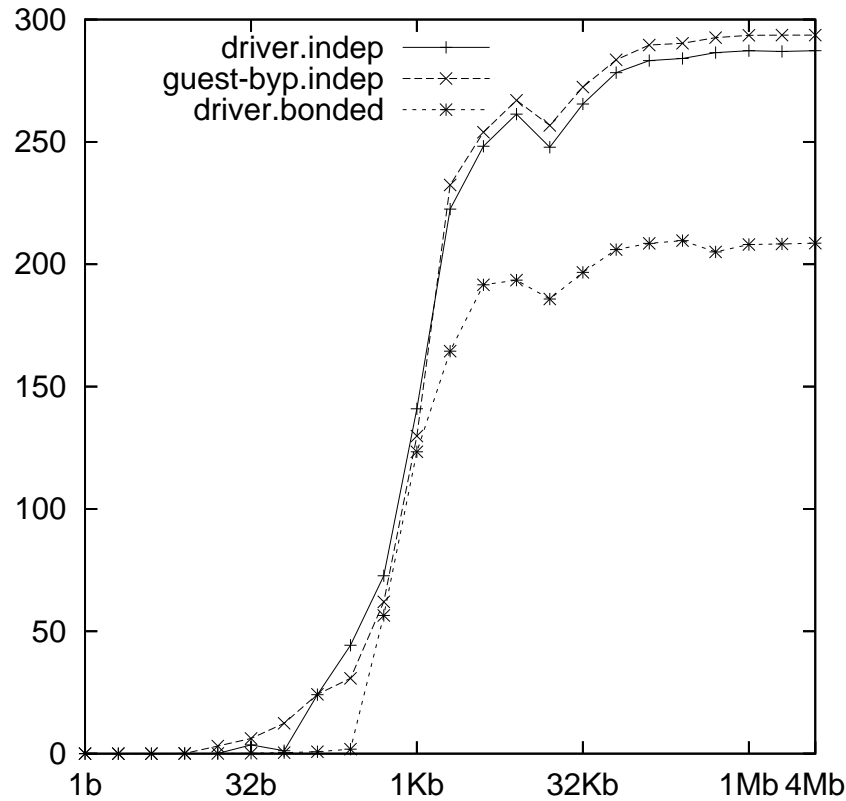


(a) uni-directional

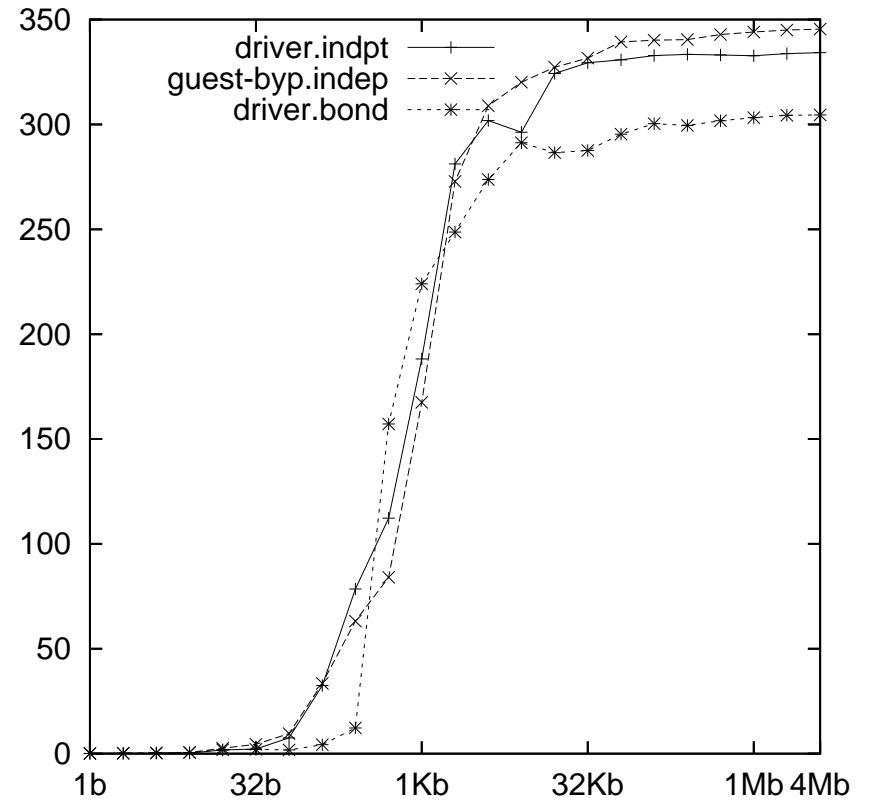


(b) bi-directional

15 Results: Bandwidth for 4-pair Configurations



(a) uni-directional



(b) bi-directional

16 Results for Multiple-Pair Configurations

- latency (at 4MB), in MB/s:

	driver.indep	guest-byp.indep	driver.bond	guest.bond
2-pairs	172	200	144	86
4-pairs	296	296	256	—

- native Linux (independent channels) behaved like driver.indep except for 4 pairs (latency slower, but uni-b/w up to 335 MB/s!)
- general observations:
 - large degree of variability in individual driver.indep.4p measurements; very small in guest-byp.indep
 - clear (although sub-linear) increase in bandwidth as network interfaces are added
 - channel bonding performance increases with multiple communication streams
 - likely diminishing return from adding yet more interfaces

17 Conclusions and Future Work

- worthwhile improvements in communication bandwidth can be achieved by using multiple network interfaces
- independent streams were generally faster significantly than channel bonding
 - best performance using Xen virtualized hosts with VMM-bypass due to full parallelization of TCP/IP stack processing
 - while not decisive, showed least variability in performance
- future work:
 - VMM-bypass needed for intra-node communication (non-trivial)
 - test application level performance
 - test with other motherboards and NICS; 8 CPU case?
 - develop virtual clusters: customization, migration
- virtualization offers many advantages for HPC; where VMM-bypass is acceptable, may even have benefits in performance !