Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, Neil S. O'Brien,
Iridis-pi: a low-cost, compact demonstration cluster, Cluster Computing, June 2013. DOI: 10.1007/s10586-013-0282-7
Full text: http://link.springer.com/article/10.1007%2Fs10586-013-0282-7

Iridis-pi: a low-cost, compact demonstration cluster

Simon J. Cox · **James T. Cox** · **Richard P. Boardman** · **Steven J. Johnston** · **Mark Scott** · **Neil S. O'Brien**

Received: 1 February 2013 / Accepted: 29 May 2013

Abstract In this paper, we report on our "Iridis-Pi" cluster, which consists of 64 Raspberry Pi Model B nodes each equipped with a 700 MHz ARM processor, 256 MiB of RAM and a 16 GiB SD card for local storage. The cluster has a number of advantages which are not shared with conventional data-centre based cluster, including its low total power consumption, easy portability due to its small size and weight, affordability, and passive, ambient cooling. We propose that these attributes make Iridis-Pi ideally suited to educational applications, where it provides a low-cost starting point to inspire and enable students to understand and apply high-performance computing and data handling to tackle complex engineering and scientific challenges. We present the results of benchmarking both the computational power and network performance of the "Iridis-Pi." We also argue that such systems should be considered in some additional specialist application areas where these unique attributes may prove advantageous. We believe that the choice of an ARM CPU foreshadows a trend towards the increasing adoption of low-power, non-PC-compatible architectures in high performance clusters.

Keywords Low-power cluster · MPI · ARM · Low cost · Education · Hadoop · HDFS · HPL

1 Introduction

In this paper, we introduce our "Iridis-Pi" cluster, which was assembled using 64 Raspberry Pi nodes interconnected with 100 $\mathrm{Mbit\,s^{-1}}$ Ethernet links, two of which are shown in Figure 1. The name is derived from the names of the Raspberry Pi, and of our main institutional cluster, Iridis 3, which ranked 74$^{\mathrm{th}}$ in the TOP500 list and was the greenest machine in the UK when it was launched in 2010. The "Iridis-Pi" cluster has a low total cost, comparable to that of a single workstation, and consumes relatively little energy. These qualities, along with its light weight, small volume and passive, ambient cooling render it eminently suitable for a number of applications that a to which a conventional cluster with its high attendant costs and special infrastructure requirements is ill-suited.

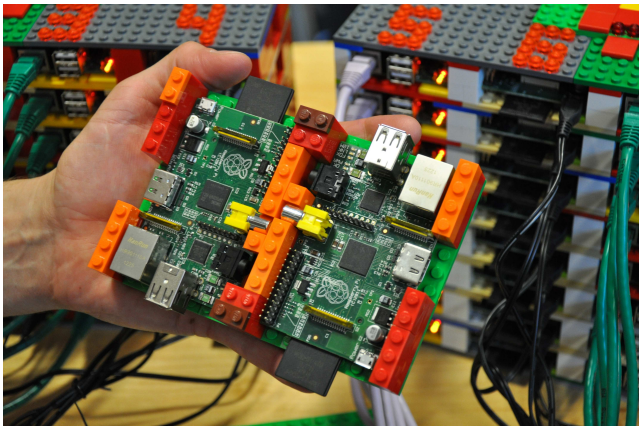

Fig. 1 The "Iridis-Pi" cluster

1.1 Context and related work

Low-power, data intensive computing is coming to be an area of great interest, both academically and in industry. As large scale, data-rich applications become increasingly mainstream, infrastructure providers are paying considerable attention to both the energy consumed by the computational hardware and the cooling burden that this imposes. The hardware industry is offering more energy-efficient servers,

Prof. S. J. Cox
Faculty of Engineering and the Environment,
University of Southampton, Southampton, UK. Tel.: +44-2380-593116
Fax: +44-2380-597082
E-mail: sjc@soton.ac.uk

often using low-power CPUs, and there is growing interest in using low-power chips of alternative architectures such as ARM in the data centre; companies such as Calxeda[1] have been founded around these developments. Moreover, Adapteva's "Parallela: A Supercomputer for Everyone" project[2], recently succeeded in gathering sufficient community funding towards the aim of democratising high-performance computing by producing compact boards based upon a scalable array of simple RISC processors accompanied by an ARM CPU. Traditional spinning disks deliver relatively poor seek performance, which decreases their attractiveness for random access to small data, whilst high-performance RAM consumes significant amounts of energy. Partially in response to these shortcomings, solid state flash storage is seeing a rise in popularity, offering low power consumption coupled with fast, efficient random access to data.

There is an increasingly active research effort into low-power cluster systems, such as the Fast Array of Wimpy Nodes (FAWN) project [3], which develops low-power systems for key-value storage systems using a cluster of low-power, PC-compatible systems based on AMD Geode LX or Intel Atom processors and flash based storage. Others have also proposed coupling solid state storage and low-power CPUs, with a goal of increasing systems' I/O performance without incurring increased power consumption [14]. Earlier work also argued that low-cost, low-power servers may return better price-to-performance and power-to-performance ratios than purpose-built servers [5, 8, 10].

1.2 The present work

The cluster that we present in this work combines the unconventional elements of utilising low-cost and low-power ARM processors, commodity Ethernet interconnects, and low-power flash based local storage, whilst supporting traditional technologies such as MPI upon which many supercomputing applications are built. With a very compact overall size, light weight, and passive, ambient cooling, our cluster is ideal for demonstration and educational purposes. We also propose that similar architectures may be configured to achieve high levels of fault tolerance at modest cost, which may find applications in *e.g.* military and in-vehicle systems.

The structure of the paper is as follows: in Section 2 we give the specification of the cluster and its nodes, describing its hardware, software, cost and power consumption. In Section 3 we then turn our attention to the cluster's performance, giving results from various benchmarks. With these results in mind, we examine applications to which this kind of architecture might be well-suited in Section 4. Finally, we conclude the paper in Section 5 with an outlook in which we foresee an increasing use of low-power, ARM based computing in high-performance and scientific applications.

2 System description

In this section, we provide a description of the architecture of the "Iridis-Pi" cluster, both in terms of its hardware components[3] and its software environment. We also detail the low cost of the system and provide measurements of the power consumption. The section finishes after considering some of the potential shortcomings of building a cluster with such low-cost hardware.

2.1 Hardware

The "Iridis-Pi" system consists of 64 Raspberry Pi Model B nodes[4], which are credit card sized single board computers. They feature a Broadcom BCM2835 system-on-chip, which integrates a 700 MHz ARM1176JZF-S RISC processor, 256 MiB RAM (current Model B production models have been upgraded to 512 MiB), a Broadcom VideoCore IV GPU with HDMI and composite output, a DSP for audio, and a single USB controller. This is wired up to an SMSC 9512 USB 2.0 hub and $10/100\,\mathrm{Mbit\,s^{-1}}$ Ethernet adaptor, which exposes two USB 2.0 and one Ethernet ports externally. Local storage is provided via an SD (secure digital) card slot on board, and there are various low-level interfaces including I^2C and SPI busses, a UART and eight GPIO pins. The device was designed to be cheap to manufacture, and costs $35 at the time of writing [12].

The "Iridis-Pi" nodes are housed in a chassis built from Lego, and interconnected via commodity Ethernet, as illustrated in Figure 2. We installed a Kingston 16 GB SDHC flash memory card in each node, with a specified write speed of up to $15\,\mathrm{MB\,s^{-1}}$. Therefore, the cluster has a total RAM of 16 GiB and a flash capacity of 1 TB. We powered the nodes using individual commodity, off the shelf, 5 V DC power supplies connected to the micro USB power sockets on the nodes.

The Raspberry Pi is not currently Open Source Hardware, but the project has openly published schematics and many components of the required device drivers. The Raspberry Pi uses the first ARM-based multimedia system-on-a-chip (SoC) with fully-functional, vendor-provided, fully open-source drivers [1], although some code that runs on the

[1] `http://www.calxeda.com/`

[2] see `http://www.kickstarter.com/projects/adapteva/parallella-a-supercomputer-for-everyone`, accessed 10 Dec 2012.

[3] also shown in a video available online, `https://www.youtube.com/watch?v=Jq5nrHz9I94`, assembled according to the guide at `http://www.southampton.ac.uk/~sjc/raspberrypi/pi_supercomputer_southampton_web.pdf`

[4] See `http://www.raspberrypi.org/`

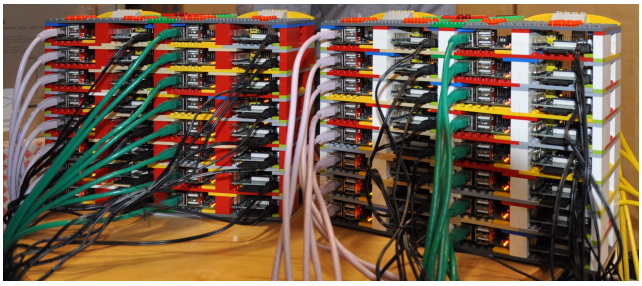

Fig. 2 The "Iridis-Pi" cluster

GPU remains closed-source. This emphasises an increasing trend towards open-source hardware as well as software entering mainstream computing.

2.2 Software

We run the Raspbian operating system[5], an optimised version of the Debian GNU/Linux distribution for the Raspberry Pi. We prepared one SD card using an image downloaded from the Raspberry Pi Foundation, and customised this by compiling our benchmarks and associated software. We cloned this, our 'master' image, to the SD cards of the other nodes.

The results we report on here were, except where stated otherwise, were generated while the cluster was running the 16th August 2012 release of the operating system and device firmware. We have subsequently upgraded to the latest release (at the time of writing, is dated 16th December 2012) and noted a small performance increase on LINPACK performance when using the default system clock speed settings.

The benchmarks in this paper required additional software to be installed. To assess the numerical compute power of the nodes and the cluster, we employed the well-known LINPACK benchmark [6] for single-node performance, and the HPL (High-Performance LINPACK) benchmark to measure the throughput of the cluster as a whole. Both of these benchmarks time the machine solving a dense $n \times n$ linear system, $\mathbf{A}\boldsymbol{x} = \mathbf{B}$, and since they are often CPU-bound, we enabled several compiler optimisations supported by the `gcc` compilers on the ARM platform. The HPL benchmark also requires the BLAS (basic linear algebra subroutines) and a message passing library such as MPICH2 [4] to be available on the system. We chose to use the ATLAS BLAS package [15], employing the same compiler optimisations that were used for LINPACK and HPL. For our LINPACK benchmark we picked an implementation in C which is available through Netlib[6]. More efficient implementations may exist but the general availability and lack of specialised library dependencies of this implementation makes it appealing. Network bandwidth and latency measurements were performed using iperf [7] and the NetPIPE performance evaluator [13]. We tested the raw IO throughput using DD and the cluster throughput using Hadoop [18].

[5] `http://www.raspbian.org/`

[6] `http://www.netlib.org/benchmark/linpackc.new`

2.3 Cost

The total cost of the system excluding network switches was below £2500. Three 24-port $10/100\,\mathrm{Mbit\,s^{-1}}$ switches could be procured for less than £110 in total, so a complete "Iridis-Pi" system could cost as little as £2610. We opted for more capable $1\,\mathrm{Gbit\,s^{-1}}$ switches with additional management features and power over Ethernet, bringing the total cost to around £3400. Thus, the overall cost is comparable to that of a single, multi-core workstation.

2.4 Power

The power consumption of the system is very modest. We measured the current drawn from the mains with the cluster idle (the current drawn by the switches was not included in this measurement) as 810 mA, which corresponds to 194 V A with a 240 V mains supply. With the cluster busy (all nodes working on a CPU-intensive benchmark test) the current rose to 900 mA, corresponding to 216 V A.

2.5 Physical enclosure

The Lego design for the rack consisted of 2 blocks of 8 layers each containing 4 Raspberry Pi computers to give a total of 64 housed together. A single layer is shown in figure Figure 3. This design was optimised to get to a high packing density along with allowing access to the HDMI ports on the outer nodes. By placing the nodes in alternating directions it allowed separation of the power and network connections into bundles of 8 and helped with both the physical system stability and cable strain relief. No additional external cooling was needed and the system has been stable for over 3 months with no heat-related issues for the processors and only a very minor discoloration of a few of the single white Lego bricks which are touching the (yellow) video out leads. We are currently developing a 3D Rapid Prototyped case for the system. However, we found that Lego was a convenient way to develop a quick, easy, and stable housing system for the computers. It has also proved a compelling way to demonstrate how this low-cost system can be easily built with parts around the home and has also created further interest and outreach to demonstrate this sort of unconventional architecture to a wider audience.

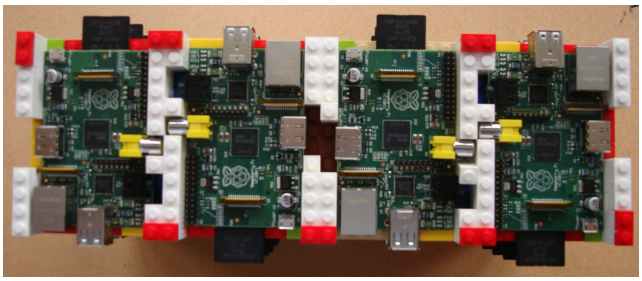

Fig. 3 Layout of a layer of 4 Raspberry Pi computers showing back to back arrangement

2.6 Management features

The Raspberry Pi boards lack many of the management features available on current server-class motherboards for the microprocessors more commonly deployed in servers and clusters, such as the ability to boot the system from the network for rapid deployment of operating systems, as well as various remote management and system health monitoring technologies which reduce the need for physical hands-on processes during system administration. These shortcomings will not be a problem for individual users who wish to use single Raspberry Pi boards, and such users make up the majority of the Raspberry Pi's target market. However, the lack of a network boot facility for cluster deployment meant that the 64 SD cards had to be manually imaged—a time-consuming task. The lack of remote out-of-band management facilities would be a considerable inconvenience if the cluster were to be deployed in a data centre, as local access to each board may be necessary for any troubleshooting or upgrading process; however, the compact, lightweight, quiet, passively-cooled nature of the "Iridis-Pi" cluster could mitigate these issues, since it is feasible to have the cluster located in close proximity to its users and administrators rather than in a data centre.

3 Performance results

In this section we present some results obtained in benchmarking the performance of the cluster and of the nodes. We begin with some computational benchmark results obtained by running the LINPACK and HPL (high-performance LINPACK) benchmark suite [2] on the cluster. We then give additional performance results for the network and storage subsystems.

3.1 Single node compute

We benchmarked the nodes of the cluster by running the LINPACK benchmark, both in single and double precision

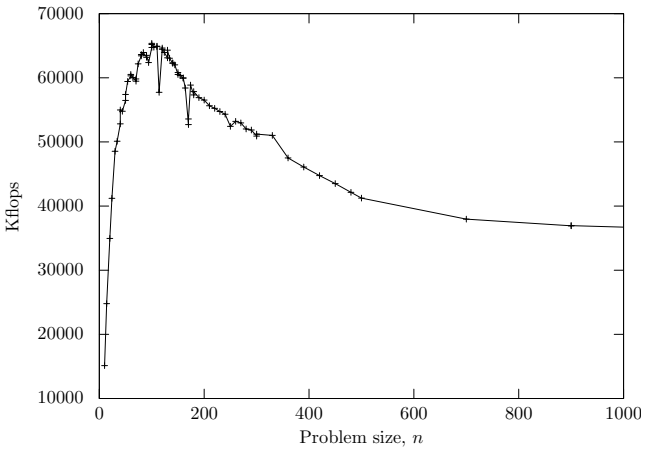

Fig. 4 Single precision performance of a single node on various problem sizes n using the LINPACK benchmark (tests carried out on Raspbian 16th Dec 2012 release with the Linux 3.6.11+ kernel).

modes. This benchmark aims to assess the computer's ability to solve real-world problems by solving a dense $n \times n$ system of linear equations, $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, on a single node.

We compiled the benchmark with optimisations, and ran it for $n = 200$ in both double and single precision on the individual nodes of the "Iridis-Pi" system. Across the nodes, the mean single precision performance was 55 571 kflop s^{-1}, with a standard deviation of 304 kflop s^{-1}; the mean double precision performance was 41 078 kflop s^{-1} with standard deviation 1077 kflop s^{-1}.

On a single node, running a newer release of the operating system, compiler and firmware (dated 16th December 2012), we investigated a range of different LINPACK problem sizes using the single precision benchmark. Figure 4 illustrates our results, which indicate a peak single-node performance of around 65 000 kflop s^{-1} for a problem size $n = 100$.

By means of a comparison, we also ran the LINPACK benchmark on a workstation based on an Intel Xeon E5320 clocked at 1.86 GHz, and achieved a peak single-precision performance (using a single core, of which this chip has four) of 1.18 Gflop s^{-1} with $n = 2000$ and 1.06 Gflop s^{-1} for $n = 200$.

3.2 Cluster compute

Here we present results for the compute performance of the cluster running the HPL suite. This benchmark aims to assess the computer's ability to solve real-world problems by solving a dense $n \times n$ system of linear equations, $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, in parallel across a chosen number of nodes, a common requirement in engineering applications.

We benchmarked using various problem sizes n and numbers of nodes N, and for each combination, we tried multiple process grid dimensions. In these results we quote the

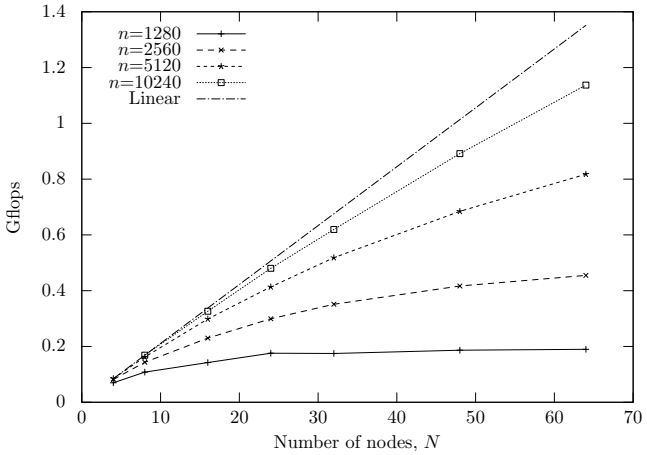

Fig. 5 Computational performance measured using HPL benchmark, as a function of number of nodes, N, for various orders n of the matrix **A**.

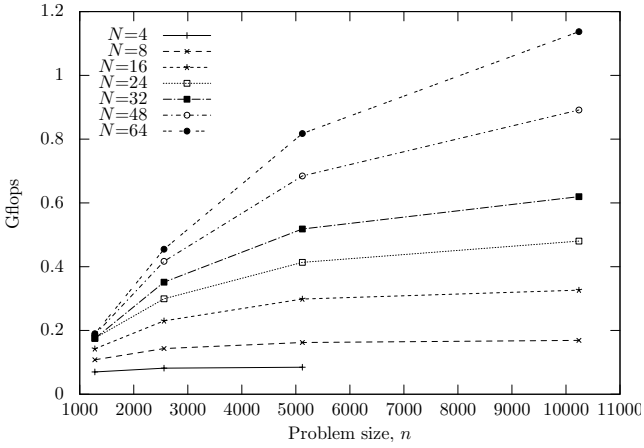

Fig. 6 Computational performance measured using HPL benchmark, as a function of the various order n of the matrix **A**, for various numbers N of nodes.

best achieved performance for a given n–N combination. The performance scaling is presented in Figure 5 and Figure 6.

In Figure 5, we show how the computational performance increases for various problem sizes as more nodes are introduced. The linear scaling line is fitted through the origin and the first data point for the $n = 10240$ problem. It can be seen that for the smallest problem ($n = 1280$), using more than 24 nodes gives no advantage, with the increase in communications overhead overwhelming the larger processing capacity available. For $n = 10240$, the system continues to demonstrate good scaling up to the full 64 nodes, achieving $1.14\,\mathrm{Gflop\,s^{-1}}$ throughput. Larger problem sizes (constrained by the available memory) would achieve even closer to the asymptotic linear scaling.

The results are plotted as a function of the problem size, n, for each of the numbers N of nodes in Figure 6. It was not possible to run the largest problem size on only four nodes because the available memory was insufficient. It can be seen that the system performance scaling is closer to linear for the larger problem sizes, where communication overheads are amortised over larger, longer-running calculations.

3.3 Network

We used the nodes' on-board $100\,\mathrm{Mbit\,s^{-1}}$ network interface (which is part of the SMSC 9512 USB hub and Ethernet adaptor, and therefore communicates to the processor over USB 2.0) for our cluster's inter-node communications. We now present the results we obtained from assessing the available bandwidth and latency using these interfaces. We considered three scenarios: firstly, picking two physical nodes, we measured the bandwidth between them as a function of the size of the message passed between the nodes in a "ping pong" style test, and secondly as a function of the number of TCP connections made in parallel. Finally, we nominated one physical node a 'server' and measured the bandwidth that it afforded to various numbers N of 'client' nodes that connected to it. These tests were carried out with the iperf [7] and NetPIPE [13] software tools.

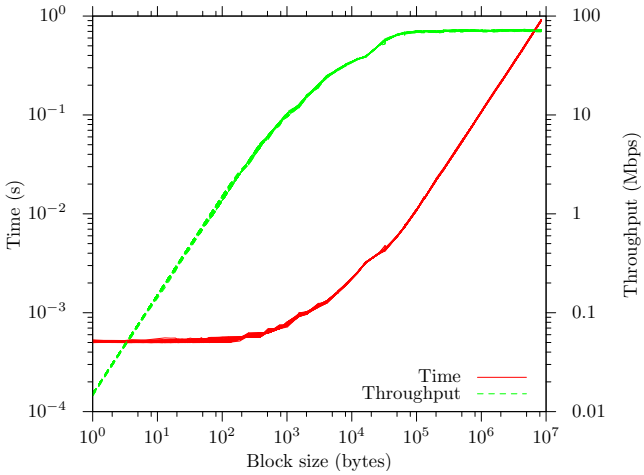

Fig. 7 Throughput and half-round-trip time plotted against block size.

3.3.1 Effects of data block size

We nominated one node as our 'reference' node, and used the NetPIPE software to investigate the effect of block size on throughput and round-trip time between this reference node and the other nodes in the system. A round trip consists of the 'reference' node sending, then receiving, a block of some size c, whilst the 'other' node receives, then sends blocks of size c. Our recorded times are the half-round-trip times, *i*.e. they represent the time taken to send or receive a block of a given size.

In Figure 7, network throughput and half-round-trip time are plotted against the size, c, of the blocks in use. Lines

representing the measurements carried out between the 'reference' and each 'other' node are superimposed upon each other, accounting for the thickness of the plotted lines and illustrating that the inter-node variability is relatively low. It can be seen from the throughput curve that using small block sizes will limit the network's throughput severely. This observation is also borne out by the time curve for small block sizes; there is virtually no time penalty for increasing the block size from 1 B to a little above 100 B, and the throughput grows approximately linearly with block size in this regime, indicating that the performance here is mostly limited by per-packet overheads (*i.e.* latency). For block sizes of approximately 10^5 B and above, peak throughput is achieved and in this regime, the time increases linearly with block size, indicating that the transfers are now limited by the maximum sustained bandwidth of the network link.

Latency The latency can be inferred from Figure 7 as the time taken for a block size of zero, and is around 0.5 ms between the 'reference' and any of the 'other' nodes.

3.3.2 Multiple connections between one client and one server

We now turn our attention to multiple simultaneous network connections between a single client and a single server, as may arise *e.g.*, when distributing multiple files to a node. The results of the first such investigation, using various numbers of parallel connections between one client and one server node, are shown in Figure 8. It can be seen that the aggregate bandwidth reported by the iperf server decreased with increasing numbers of connections, which is as expected, since each connection has associated overheads. Modelling the overheads incurred as a linear function of the number of connections used, we fitted the curve $B(N_p) = a/N_p + b \cdot (N_p - 1)$, where a is the bandwidth of a single connection and b is a coefficient expressing a per-connection overhead, to the data using a nonlinear least-squares (NLLS) Marquardt-Levenberg algorithm [9,11]. This resulted in the single-connection bandwidth $a = 74.1 \pm 0.3\,\mathrm{Mbit/s}$ and a per-connection decrement of $b = -0.014 \pm 0.0099\,\mathrm{Mbit/s}$. This is the curve fit shown in the figure and gives some insight into the relatively small additional overhead incurred when many parallel connections are opened between two nodes.

We noted that with larger numbers of parallel connections, not all connections were reported by the server as having been open for identical time periods. In these cases, the reported bandwidth sum is an estimate, obtained from the total amounts of data transferred and time passed.

Connections from multiple clients to one server An alternative case is the one where a particular node is designated the

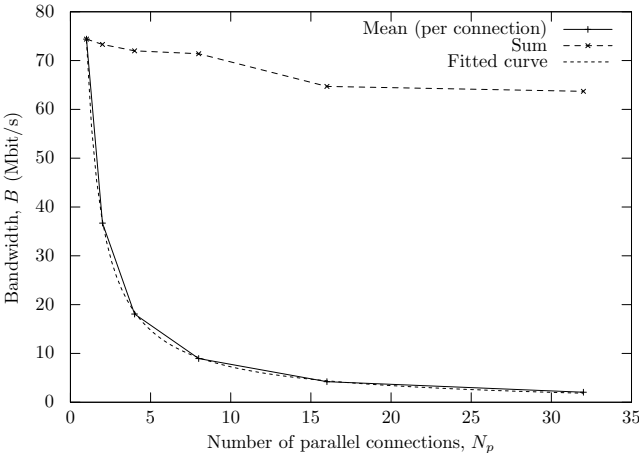

Fig. 8 Bandwidth achieved between two Iridis-Pi nodes, as a function of number of parallel connections.

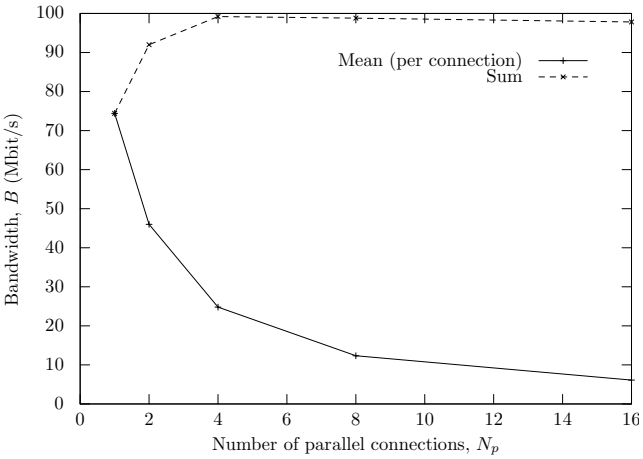

Fig. 9 Bandwidth achieved between multiple Iridis-Pi nodes as clients of one server node, shown as a function of number of connections.

`iperf` server, and we make multiple connections to it from various numbers of other nodes, measuring the bandwidth achieved in each situation, as might be the case when all the nodes involved in an MPI job communicate their results to a "master" (rank 0) node. To synchronise the connections to the server, we ensured that all the nodes kept an accurate local clock by running `ntpd`, and scheduled the multiple connections to begin synchronously by using the `at` scheduler on each node.

Our results are presented in Figure 9. It can clearly be seen that the sum of the bandwidths across all client nodes increases to a peak for four concurrent connections, and decreases only slightly as the number of clients increases. At peak, we achieved 99.2% of the $100\,\mathrm{Mbit\,s^{-1}}$ theoretical peak throughput.

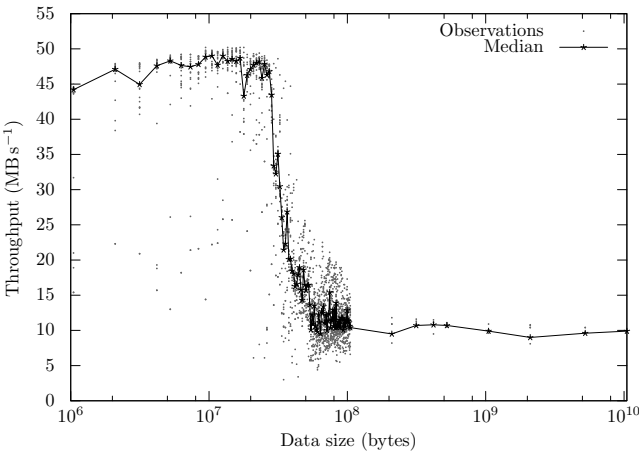

Fig. 10 SD card write performance results for files of various sizes.

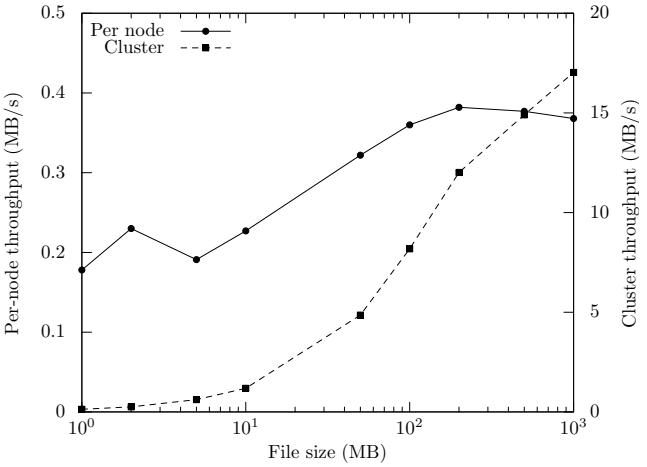

Fig. 11 Cluster and per node write throughput for files of various sizes.

3.4 Storage

The "Iridis-Pi" uses 16 GiB Class 10 Kingston SD cards with an expected 15 MB s^{-1} write throughput. The expected I/O performance will vary between brands and classes. We tested the performance using `dd` to write various file sizes, keeping a constant 1 MiB block size, and plotted the results in Figure 10. The data were piped from `/dev/zero` to reduce the CPU load. For large file sizes, the write rate converges to approximately 10 MB s^{-1}.

3.5 Distributed file system

Each node in the cluster has an SD card for storage, giving the cluster a total storage capacity of 1 TiB. As this storage is distributed across all the nodes it is difficult to utilise efficiently, for example the MPI tasks use local storage as required but then pool results on a single node. In order to utilise the storage efficiently we need to use a distributed file system, such as Hadoop DFS [16, 17]. Hadoop consists of a distributed file system as well as a distributed computational system — map-reduce [18] — and it is supported across a wide range of hardware, operating systems and cloud providers.

We used Hadoop version 1.1.1 with default settings with the exception of the following: the number of replications was set to one, resulting in each file existing in a total of two locations; and each node was limited to a maximum of one map and one reduce task to avoid over committing the hardware. With the exception of the buffer tests, all tests were carried out with a memory buffer size of 1024 B.

The Hadoop cluster setup uses a dedicated node for the distributed file system service, and a dedicated node for the job tracker service. Each of the remaining nodes are configured with both a data tracker and a task tracker service. Counting the cluster and Hadoop headnodes, this reserves a total of three nodes for dedicated functions plus two kept as spare, leaving 59 nodes for the Hadoop testing.

Since each node does not have much RAM, we had to increase the swap file size to run the tests. Each node was allocated a 400 MB swap file. The current production version of the Model B Raspberry Pi , which has double the RAM would be better suited to running Hadoop as the tests quickly run out of memory and using a swap file has a serious performance impact.

Hadoop ships with a series of benchmarking and testing tools. We selected the TestDFSIO [17] benchmark to perform a series of read and write tests on the cluster. These also test the compute functionality as TestDFSIO uses map-reduce to distribute tests and collate results on the cluster. A series of tests were run for both read and write with a constant 1024 B memory buffer, and a third set of tests were run with a variable memory buffer size. Varying the buffer size up to 1.3 MB did not have an impact on the throughput.

The per node throughput shown in Figure 11 peaks at 0.38 MB s^{-1} for file sizes of 200 MB and only degrades slightly as the file size grows to 1 GB. The throughput for smaller files is significantly worse. If we look at the throughput for the cluster as a whole, Figure 11 shows that the throughput improves with larger file sizes, and indicates that files greater than 1 GB can expect to see slightly better throughput than the 17 MB s^{-1} shown.

Although the Hadoop IO results are very slow, the cluster executed map-reduce tasks and replicated data as expected. The setup and configuration is similar to that implemented on larger, more traditional clusters, providing an excellent learning and testing tool.

4 Potential Applications

The small size, low power usage, low cost and portability of the "Iridis-Pi" cluster must be contrasted against its relatively low compute power and limited communications bandwidth (compared to a contemporary, traditional HPC

cluster), making this architecture most appropriate as a teaching cluster. In this role, it could be used to help students to understand the building blocks of parallel and high performance computation. It is also a valuable alternative to using virtualization to demonstrate the principles of HPC, since the latter tends to hide various "real-world" aspects of HPC such as interconnects between nodes, power, cooling, file systems, etc. Due to its low cost, it may also bring cluster computing to institutions which lack the space, funds, and administrative staff to run a conventional cluster. Even in institutions which have clusters available, "Iridis-Pi" style clusters could be made available to students as a rapid means of testing the functionality and scaling of parallel codes without the long job queues often associated with institutional systems. It would also encourage developers to work with other architectures, enhancing code portability, which may be of increasing importance as low-power ARM chips begin to enjoy a larger role in the data centre, as we suggest in Section 5.

A reasonable comparison of computing power at this cost would be a single multi-core workstation, which may provide more capacity and expandability in terms of conventional SATA storage and PCI Express peripherals. However, a portable cluster can be designed to continue to run even after failures of a number of its nodes as single points of failure may be removed. In harsh environments or where repair would be difficult or impossible—such as military use, or in the ocean or space—this could be attractive if raw computing power is not the only requirement, raising several interesting possibilities.

As the computational power of low-power processors improves, driven largely by portable consumer devices such as mobile telephones and tablets, the amount of useful work that can be done on a cluster like this will inevitably increase. Embedded clusters could become convenient for building management systems, where they could be used for running prediction models based on current sensor readings to optimise the energy consumption and inhabitant comfort of a building. Light-weight clusters could also become useful in aeroplanes, trains and cars. Even in the consumer market, small groups of embedded computers networked together in future generations of smart houses could allow more intelligent energy management and improved reliability.

Our "Iridis-Pi" cluster includes 64 GPUs, giving rise to the possibility of a portable GPU cluster which may provide something that a single high-powered workstation cannot. Its 64 video outputs would allow it to drive video walls, and the GPUs could perform image reconstruction, perhaps in medical or similar applications. The network bandwidth may be the main hindrance here. Again, a portable GPU cluster may be designed so that even if some nodes fail, the cluster is able to continue at a reduced capacity (*e.g.*, at a lower resolution in the case of image reconstruction). These GPUs have a specified performance of $24\,\mathrm{Gflop\,s^{-1}}$ each and therefore afford the cluster, in theory, an additional $1536\,\mathrm{Gflop\,s^{-1}}$ peak performance.

As clustered file systems and map-reduce functionality become more common place it is advantageous to demonstrate such concepts for educational proposes. We installed Hadoop on the cluster and configured both the distributed file system and the map-reduce features to provide over 700 GiB of usable storage as well as a distributed computational resource. The performance was usable but limited to simple examples, with RAM being the key limitation. Current production versions of the Raspberry Pi Model B have more RAM and would be better suited to use with Hadoop, but still biased towards education.

Overall, a low cost cluster-in-a-box has the potential to broaden access to cluster computing and move clusters out of the data centre and into everyday use.

5 Outlook

With the increasing relevance of energy-aware computing we foresee that ARM-based processors are likely to become ever more popular as time goes on. AMD has recently announced[7] that it will introduce a multi-core system-on-a-chip processor based on 64-bit ARM technology, targeted for production in 2014. The company foresees the transition in which 64-bit ARM chips become mainstream as analogous to the earlier transition in which x86-64 architectures became popular in the data centre. The performance per watt offered by these ARM parts should be higher than that of x86-64 competition, and their energy efficiency will increase the achievable server density. Further evidence for this view is the recent success of Adapteva's "Parallela: A Supercomputer for Everyone" project[8], which aims to democratise high-performance computing by producing compact boards based upon a scalable array of simple RISC processors accompanied by an ARM CPU. Moreover, the increasing trend towards open computational environments is echoed in the Parallela project, which will support the OpenCL standard, and ship with free, open source development tools.

6 Conclusions

We have described our 64-node Raspberry Pi cluster and presented some performance benchmarks. A single node exhibited a peak computational performance of around $65\,000\,\mathrm{kflop\,s^{-1}}$ in single precision for a problem size of $n = 100$, and the

[7] `http://www.amd.com/us/press-releases/Pages/press-release-2012Oct29.aspx` accessed 10 Dec 2012.

[8] see `http://www.kickstarter.com/projects/adapteva/parallella-a-supercomputer-for-everyone`, accessed 10 Dec 2012.

mean double precision performance for $n = 200$ was 41 078 kflop s^{-1}. Large problem sizes in the high performance LINPACK (HPL) benchmark scaled well with increasing numbers of nodes but network overheads limited the scaling for smaller problems. The cluster delivered a peak performance of 1.14 Gflop s^{-1} using 64 nodes with a problem size of $n = 10240$. The memory capacity of the nodes is the main limit on the maximum problem size addressable.

At peak, a node was found to achieve 99.2% of its nominal 100 Mbit s^{-1} bandwidth, and the typical latency for small packets was around 0.5 ms.

Benchmarking Hadoop running on the cluster showed that the IO performance scaled across the cluster, although the peak per node throughput was a rather poor 0.38 MB s^{-1}. For file sizes of 1 GB we saw a total cluster throughput of 17 MB s^{-1}.

We have described how the unconventional architecture of this cluster reduces its total cost and makes it an ideal resource for educational use in inspiring students who are learning the fundamentals of high-performance and scientific computing. We also explored additional application areas where similar architectures might offer advantages over traditional clusters. We foresee that, although our cluster architecture is unconventional by today's standards, many aspects of its design—the use of open source hardware and software, the adoption of low-power processors, and the wider application of flash based storage—will become increasingly mainstream into the future.

References

1. Open source ARM userland. URL `http://www.raspberrypi.org/archives/2221`
2. A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary: HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. URL `http://www.netlib.org/benchmark/hpl/`. Accessed Jan 2013
3. Andersen, D.G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., Vasudevan, V.: Fawn: a fast array of wimpy nodes. Commun. ACM **54**(7), 101–109 (2011). DOI 10.1145/1965724.1965747
4. Argonne National Laboratory: MPICH2. URL `http://www-unix.mcs.anl.gov/mpi/mpich2/`
5. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The landscape of parallel computing research: A view from berkeley. Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley (2006). URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html`
6. Dongarra, J.J., Luszczek, P., Petitet, A.: The linpack benchmark: past, present and future. Concurrency and Computation: Practice and Experience **15**(9), 803–820 (2003). DOI 10.1002/cpe.728
7. Gates, M., Tirumala, A., Ferguson, J., Dugan, J., Qin, F., Gibbs, K., Estabrook, J.: Iperf - The TCP/UDP bandwidth measurement tool. URL `http://sourceforge.net/projects/iperf/`. [Online]
8. Hamilton, J.: Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services (2009). URL `www.mvdirona.com/jrh/talksandpapers/jameshamilton_cems.pdf`
9. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics **2**, 164–168 (1944)
10. Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T., Reinhardt, S.: Understanding and designing new server architectures for emerging warehouse-computing environments. SIGARCH Comput. Archit. News **36**(3), 315–326 (2008). DOI 10.1145/1394608.1382148
11. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics **11**(2), 431–441 (1963). DOI 10.1137/0111030. URL `http://epubs.siam.org/doi/abs/10.1137/0111030`
12. Raspberry Pi Foundation: Raspberry Pi FAQs. URL `http://www.raspberrypi.org/faqs`. Accessed Dec 2012
13. Snell, Q.O., Mikler, A.R., Gustafson, J.L.: Netpipe: A network protocol independent performance evaluator. In: in IASTED International Conference on Intelligent Information Management and Systems (1996)
14. Szalay, A.S., Bell, G.C., Huang, H.H., Terzis, A., White, A.: Low-power amdahl-balanced blades for data intensive computing. SIGOPS Oper. Syst. Rev. **44**(1), 71–75 (2010). DOI 10.1145/1740390.1740407
15. Whaley, R.C., Petitet, A.: Minimizing development and maintenance costs in supporting persistently optimized BLAS. Software: Practice and Experience **35**(2), 101–121 (2005). URL `http://www.cs.utsa.edu/~whaley/papers/spercw04.ps`
16. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, *The Google file system*, ACM Symposium on Operating Systems Principles, 2003, pp. 29–43.
17. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler, *The Hadoop Distributed File System*, Symposium on Mass Storage Systems, 2010.
18. Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplied Data Processing on Large Clusters*, Operating Systems Design and Implementation, 2004, pp. 137–150.

Simon Cox is Professor of Computational Methods and Head of the Computational Engineering and Design Research group in the Faculty of Engineering and Environment at the University of Southampton. His research interests are in the application and development of computational tools, technologies and platforms to enable science and engineering research.

James Cox (aged 6) is at West Hill Park School, Titchfield, UK. He worked on this project at the University of Southampton over summer 2012 and is interested in computing and gaming using the Raspberry Pi.

Neil O'Brien holds an MPhys (Hons) degree from the University of Southampton and has recently completed his PhD in Algorithms for Scientific Computing at Southampton's Faculty of Engineering and the Environment. His research interests include numerical methods, computational technologies, and smart home networks.

Mark Scott joined the University of Southampton in 2001 as an I.T. Systems Architect and Developer. After working for several groups in the School of Engineering Sciences, he moved to the central I.T. department and is currently working on the development of data management projects for the University such as the Materials Data Centre. He is also studying for a PhD at the Microsoft Institute for High Performance Computing in the Faculty of Engineering and the Environment.

Richard Boardman completed a PhD in Computational Physics in 2005. Since then, he has worked in computational chemistry and computer science, and currently works in X-ray computed tomography.

Steven Johnston completed an MEng in Software engineering and a PhD in data management. He is currently a senior research fellow at the University of Southampton and a program manager for the .NET Gadgeteer team at Microsoft Research. His research interests include embedded hardware, data management, the internet of things and cloud computing.