Loughborough University

This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

Commodity single board computer clusters and their applications

PLEASE CITE THE PUBLISHED VERSION

https://doi.org/10.1016/j.future.2018.06.048

PUBLISHER

© the Authors. Published by Elsevier

VERSION

VoR (Version of Record)

PUBLISHER STATEMENT

This work is made available according to the conditions of the Creative Commons Attribution 4.0 International (CC BY 4.0) licence. Full details of this licence are available at: http://creativecommons.org/licenses/ by/4.0/

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Johnston, Steven J., Philip J. Basford, Colin S. Perkins, Herry Herry, Fung Po Tso, Dimitrios P. Pezaros, Robert D. Mullins, Eiko Yoneki, Simon J. Cox, and Jeremy Singer. 2019. “Commodity Single Board Computer Clusters and Their Applications”. figshare. https://hdl.handle.net/2134/34779.

Future Generation Computer Systems 89 (2018) 201–212

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Commodity single board computer clusters and their applications

Steven J. Johnston [a,*], Philip J. Basford [a], Colin S. Perkins [b], Herry Herry [b], Fung Po Tso [c], Dimitrios Pezaros [b], Robert D. Mullins [d], Eiko Yoneki [d], Simon J. Cox [a], Jeremy Singer [b]

[a] Faculty of Engineering & Physical Sciences, University of Southampton, Southampton, SO167 QF, UK
[b] School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, UK
[c] Department of Computer Science, Loughborough University, Loughborough, LE113 TU, UK
[d] Computer Laboratory, University of Cambridge, Cambridge, CB3 0FD, UK

HIGHLIGHTS

- Single Board Computers can run mainstream operating systems and workloads.
- Single Board Computers clusters replicate data center features.
- SBC clusters are a new and distinct computational deployment paradigm.
- SBC Clusters facilitate the Internet of Things, Smart Cities, Fog and Edge compute.
- SBC Clusters are a game changer in pushing application logic towards the network edge.

ARTICLE INFO

Article history:
Received 26 January 2018
Received in revised form 19 June 2018
Accepted 25 June 2018
Available online 30 June 2018

Keywords:
Raspberry Pi
Edge computing
Networks cloud computing
Centralization/decentralization
Distributed computing methodologies
Multicore architectures
Emerging architectures

ABSTRACT

Current commodity Single Board Computers (SBCs) are sufficiently powerful to run mainstream operating systems and workloads. Many of these boards may be linked together, to create small, low-cost clusters that replicate some features of large data center clusters. The Raspberry Pi Foundation produces a series of SBCs with a price/performance ratio that makes SBC clusters viable, perhaps even expendable. These clusters are an enabler for Edge/Fog Compute, where processing is pushed out towards data sources, reducing bandwidth requirements and decentralizing the architecture. In this paper we investigate use cases driving the growth of SBC clusters, we examine the trends in future hardware developments, and discuss the potential of SBC clusters as a disruptive technology. Compared to traditional clusters, SBC clusters have a reduced footprint, are low-cost, and have low power requirements. This enables different models of deployment—particularly outside traditional data center environments. We discuss the applicability of existing software and management infrastructure to support exotic deployment scenarios and anticipate the next generation of SBC.

We conclude that the SBC cluster is a new and distinct computational deployment paradigm, which is applicable to a wider range of scenarios than current clusters. It facilitates Internet of Things and Smart City systems and is potentially a game changer in pushing application logic out towards the network edge.

© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Commodity Single Board Computers (SBCs) are now sufficiently powerful that they can run standard operating systems and mainstream workloads. Many such boards may be linked together, to create small low-cost clusters that replicate features of large data centers, and that can enable new Fog and Edge Compute applications where computation is pushed out from the core of the network towards the data sources. This can reduce bandwidth requirements and latency, help improve privacy, and decentralize the architecture, but it comes at the cost of additional management complexity. In this paper, we investigate use cases driving the growth of SBC clusters, examine the trends in future hardware developments and cluster management, and discuss the potential of SBC clusters as a disruptive technology.

The introduction of the Raspberry Pi has led to a significant change in the SBC market. Similar products such as the Gumstix have been available since 2003 [1], however, the Raspberry Pi has sold in much higher volumes leading to the company behind it

* Corresponding author.
E-mail addresses: sjj698@zepler.org (S.J. Johnston), P.J.Basford@soton.ac.uk (P.J. Basford), csp@csperkins.org (C.S. Perkins), Herry.Herry@glasgow.ac.uk (H. Herry), p.tso@lboro.ac.uk (F.P. Tso), Dimitrios.Pezaros@glasgow.ac.uk (D. Pezaros), Robert.Mullins@cl.cam.ac.uk (R.D. Mullins), eiko.yoneki@cl.cam.ac.uk (E. Yoneki), sjc@soton.ac.uk (S.J. Cox), jeremy.singer@glasgow.ac.uk (J. Singer).

https://doi.org/10.1016/j.future.2018.06.048
0167-739X/© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Table 1
Comparison between Personal Computer (PC), Controller Board (CB), Smartphone, and Single Board Computer (SBC). (DB = daughter-board, N/A = not-available, ROM = read-only memory, RW-Ext = read–write external storage).

Component	PC	CB	Smartphone	SBC
CPU	DB	Yes	Yes	Yes
GPU	Yes, DB	Yes	Yes	Yes
Memory	DB	Yes	Yes	Yes
LAN	Yes, DB	N/A	N/A	Yes
Video Output	Yes, DB	N/A	N/A	Yes
Storage	ROM, RW-Ext	ROM	ROM, RW-Ext	ROM, RW-Ext
GPIO Header	Yes, (USB)	Yes	N/A	Yes

being the fastest growing computer company in the world [2]. This has led to a dramatic increase in the number of SBC manufacturers and available products, as described in Section 2. Each of these products has been subject to different design decisions leading to a large variation in the functions available on the SBC. The low price point of SBCs has enabled clusters to be created at a significantly lower cost than was previously possible. We review prototypical SBC clusters in Section 3.

SBC clusters can be created simply to gain an understanding of the challenges posed by such developments, but can also have practical uses where a traditional cluster would not be appropriate. The first SBC clusters, e.g., IridisPi [3] and the Glasgow Pi Cloud [4], were created primarily for education. Since then, SBC clusters have been created for many reasons including to manage art works [5,6], and to provide disposable compute power in extreme environments where node destruction is likely [7]. Section 4 highlights classes of use cases for this technology, including emergent fog and edge compute applications.

The purchasing of a multi-node cluster has been made significantly cheaper by the developments of SBCs, but the challenges of setup and ongoing maintenance remain. Some of these challenges are also experienced when running a standard cluster, but issues such as SD card duplication and low-voltage DC power distribution are unique to the creation of SBC clusters. Our contribution is to identify these differences and show where existing cluster management techniques can be used, and where new techniques are needed. Management tasks are further complicated by the fact that, unlike traditional clusters which are located within data centers due their high power demands, it is feasible for an SBC cluster to be geographically distributed. Section 5 discusses the major challenges.

This paper provides a survey of current achievements and outlines possible topics for future work. It concludes by looking forward, and discussing future applications for SBC systems in Section 6.

2. Single board computer overview

An Single Board Computer (SBC) has been described as "a complete computer built on a single circuit board, with microprocessor(s), memory, Input/Output (I/O) and other features required of a functional computer" [11]. This definition does not fully capture what it means to be an SBC, however, so we compared SBCs with other platforms to identify key differences. The results of this comparison are summarized in Table 1. Although the definition given above incorporates most of the factors, it ignores three major differences: the availability of built in general purpose I/O ports, power consumption, and cost. It is the inclusion of such ports and a low price that means SBCs fall into the gap between controller boards and PCs. The similarity between SBCs and smartphones is interesting to note, and the similarities continue as the majority of SBCs, and all current phones, use ARM processors as opposed to the Intel/AMD chips currently used in the PC market.

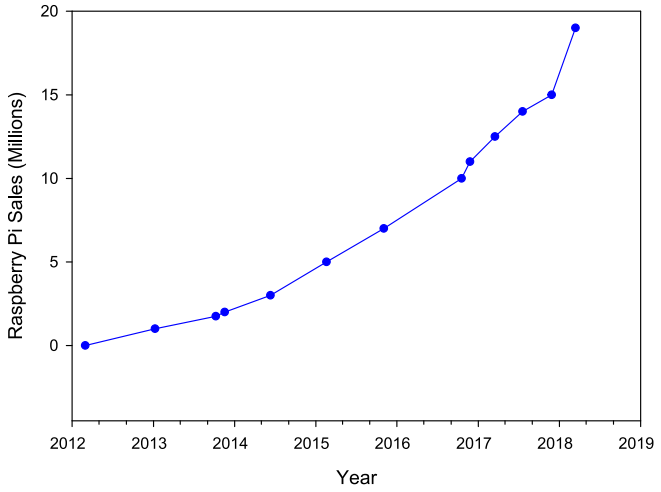


Fig. 1. Raspberry Pi units sold (all versions), according to statistics published by the official Raspberry Pi blog.

When the Raspberry Pi was released there were other SBCs, such as the Gumstix [1] and the BeagleBone, that had similar technical specifications (although they were more expensive) [8]. Despite this, it is the Raspberry Pi that has come to lead the market. Selling over a million units in the first year, the Raspberry Pi Foundation became the fastest growing computing company to date [2]. Fig. 1 shows the sales figures for Raspberry Pi units, based on statistics published by the official Raspberry Pi blog, and in March 2017 the Raspberry Pi became the third best-selling general purpose computer of all time [12]. From the early days of the Linux capable SBC, when the list of available boards was extremely limited, there is now a wide variety of different platforms available each with their own advantages and disadvantages. A very restricted list of these platforms is detailed in Table 3, with the System on Chip (SoC) used in each board further described in Table 2. Despite the fact that the SBC market is developing rapidly, manufacturers are aware that there is also demand for stability in product availability. For example, the Raspberry Pi 3B+ which was released in March 2018 has production guaranteed until January 2023 [13], and the Odroid-XU4 has guaranteed availability until the end of 2019, but is expected to be available longer [14].

One of the main advantages of an SBC, such as the Raspberry Pi, is its low-cost, which has been described as "a few weeks' pocket money" [2]. This enables children to buy one for themselves, and relaxes parents about replacement costs in the event of damage. Projects which require multiple SBCs are also within reach, and in some cases the SBC has become a standard building block for projects driven by the abundance of examples, documentation, and supporting software. Software faults on SBCs with removable storage can easily be rectified by wiping the storage; trivial in comparison to reinstalling a PC.

The low-cost and power consumption of SBCs has enabled them to be deployed into situations where a standard PC would not be suitable, but the processing requirements cannot be met by micro-controllers. Examples of such uses include collection of high resolution (12MP) images of rock-faces in the Swiss Alps [9], and controlling sensor networks on Icelandic glaciers [10].

Wireless sensor networks have benefited from the low power consumption of SBCs. This also brings advantages in traditional data centers. It has been shown that by using a cluster of Raspberry Pi nodes instead of several 'standard' servers, a reduction in power consumption of between 17x and 23x can be observed [15]. The figure is impressive despite only including power used directly by the servers, and not the reduction in associated costs due to

Table 2
Example System on a Chip (SoC) hardware used in Single Board Computers (SBC).

SoC	Cores / Clock	Architecture	GPU
Allwinner H3	4 × 1.6 GHz	32 bit	Mali-400 MP2
Allwinner R18	4 × 1.3 GHz	64 bit	Mali-400 MP2
Altera Max 10	2k logic cell FPGA	32 bit	–
Amlogic S905	4 × 1.5 GHz	32 bit	Mali-450
Broadcom BCM2835	1 × 700 MHz	32 bit	Broadcom VideoCore IV
Broadcom BCM2836	4 × 900 MHz	32 bit	Broadcom VideoCore IV
Broadcom BCM2837	4 × 1.2 GHz	64 bit	Broadcom VideoCore IV
Broadcom BCM2837B0	4 × 1.4 GHz	64 bit	Broadcom VideoCore IV
Exynos 5422	4 × 2.1 GHz & 4 × 1.5 GHz	32 bit	Mali-T628 MP6
Intel Pentium N4200	4 × 1.1 GHz	32 bit	Intel HD Graphics 505
Sitara AM3358	1 × 1 GHz	32 bit	SGX530
Xilinx Zynq-7010	2 × 667 MHz & 28k logic cell FPGA	32 bit	in FPGA

Table 3
Example Single Board Computer (SBC) platforms based on the SoCs described in Table 2. TF cards are fully compatible with micro SD cards. All prices as of April 2018.

Board	SoC	RAM	Price	I/O
Raspberry Pi 1 B+	BCM2835	512MB	$30	Audio, composite video, CSI, DSI, Ethernet, GPIO, HDMI, I2C, I2S,MicroSD, SPI, USB2
Raspberry Pi 2 B	BCM2836	1 GB	$40	Audio, composite video, CSI, DSI, Ethernet, GPIO, HDMI, I2C, I2S, MicroSD, SPI, USB2
Raspberry Pi 3 B	BCM2837	1 GB	$35	Audio, Bluetooth, composite video, CSI, DSI, Ethernet, GPIO, HDMI, I2C, I2S, MicroSD, SPI, USB2, WiFi
Raspberry Pi 3 B+	BCM2837B0	1 GB	$35	Audio, Bluetooth, composite video, CSI, DSI, Gigabit Ethernet, HDMI, I2C, I2C, MicroSD, PoE Header, SPI, USB2, WiFi
Raspberry Pi Zero W	BCM2835	512MB	$10	Bluetooth, composite video, CSI, GPIO, HDMI, I2C, I2S, MicroSD, SPI, USB2, WiFi
Odroid C2	S905	2 GB	$46	ADC, eMMC/MicroSD, Gigabit Ethernet, GPIO, HDMI, I2S, IR, UART, USB
Odroid XU4	5422	2 GB	$59	ADC, eMMC/MicroSD, Gigabit Ethernet, GPIO, HDMI, I2C, I2S, SPI, UART, USB, USB3.0
Pine A64	R18	≤2 GB	$32	CSI, DSI, Ethernet, Euler, EXP, GPIO, MicroSD, RTC, TP, USB
OrangePi Plus 2	H7	2 GB	$49	Audio, CSI, eMMC, Gigabit Ethernet, GPIO, HDMI, I2C, IR, SATA 2.0, SPI, TF, USB, WiFi
BeagleBone Black	AM335	512MB	$55	ADC, CANbus, Ethernet, GPIO, HDMI, I2C, eMMC/MicroSD, SPI, UART
UP Squared	N4200 & MAX 10	≤8 GB	$289	ADC, Gigabit Ethernet (x2), GPIO, HDMI, mini-PCIe/m-SATA, MIPI (x2), RTC, UART, USB2, USB3
Xilinx Z-turn	Zynq-7010	1 GB	$119	CANbus, Gigabit Ethernet, HDMI, TF Card, USB2-OTG, USB_UART

less demanding cooling requirements. Given the enhanced processor specification of more recent Raspberry Pi models, the power consumption improvements observed by Varghese et al. [15] may be even more dramatic on updated hardware. In order to take advantages of these savings, the SBCs have to be located in data centers with reliable connectivity—a service provided by at least two commercial hosting companies [16,17].

Recent developments in SBCs have led to the introduction of more powerful peripherals being included on the board. This is demonstrated in the Up Squared board, which as well as having a Pentium processor has an on-board FPGA [18]. This functionality currently comes at a higher purchase price, but given time such peripherals might trickle down the market into lower-cost boards. Having an FPGA attached to each node in the cluster opens up new possibilities for compute paradigms. One challenge currently faced by SBCs, including the Raspberry Pi, are storage limitations because the lack of a high speed interconnect prevents fast access to large storage volumes. This limitation has been removed in other SBCs by the provision of SATA ports, enabling standard HDDs and SSDs to be directly connected (see Table 3). As the capabilities of SBCs increase, so too does the functionality of clusters created from them.

3. Single board cluster implementations

A cluster can be defined as "a group of interconnected, whole computers working together as a unified computing resource that can create the illusion of being one machine" [19]. Previously clusters with many nodes were limited to large organizations that were able to afford the high purchase and running costs. Iridis 4, a cluster based at the University of Southampton, was bought in 2013 at a cost of £3.2M [20]. The same year the Iridis-Pi cluster was built for a total cost comparable to that of a single workstation [3]. Despite this being an extreme comparison, as when launched Iridis 4 was the most powerful supercomputer in a university in England and third largest UK based academic supercomputing facility, it puts the cost comparison into perspective. A more practical cluster would be made out of a few workstation/server nodes, which is still significantly more than the cost of a Raspberry Pi cluster. This massive reduction in the cost of creating a cluster has enabled many companies and individuals who otherwise would not be able to afford a cluster to experiment, making cluster technologies more available to the hobbyist market. Along with companies, these hobbyists have created a variety of different SBC clusters.

Table 4 shows details of published SBC clusters. It is noticeable that every cluster uses Raspberry Pi SBCs as the hardware platform. This is not to say that there are no clusters created using other platforms: clusters have been created using the NanoPC-T3 [23], Orange Pi [24], Pine A64+ [25], and the Beagle Board [26]. Despite these clusters being interesting prototypes, none of them have been scaled beyond 10 nodes, for reasons that are not clear.

One of the challenges when creating an SBC cluster is the physical arrangement, including power and network communications infrastructure. The clusters described in Table 4 use bespoke hardware solutions that vary from Lego [3] through to wooden panels [27] and laser-cut acrylic [28] as illustrated in Fig. 2. These arrangements solve the problem of mounting and holding the SBCs, but the challenge of providing power and network connectivity to the devices is not addressed. Arguably the neatest solution is that used by Mythic Beasts, which is to use Power Over Ethernet (PoE) to power the nodes [17] and reduce the amount of cabling required. This neatness comes at a cost, since each Raspberry Pi

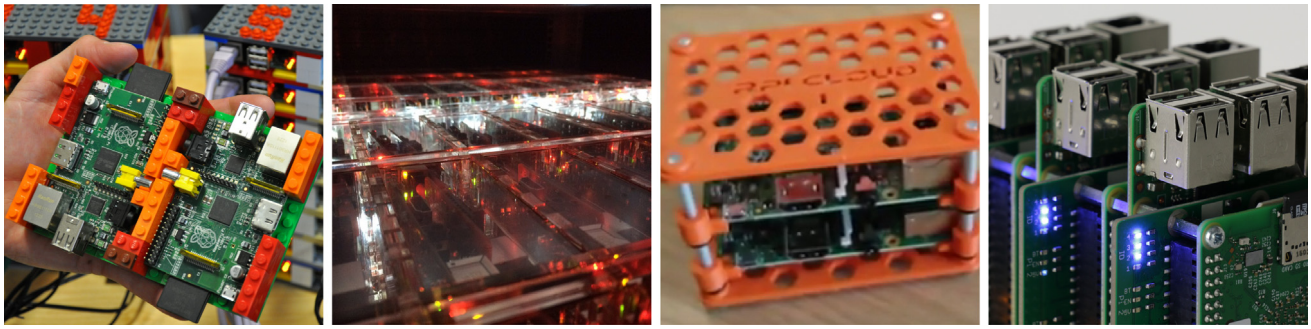


Fig. 2. A selection of SBC clusters. (a) Iridis Pi [3] built using Lego, (b) Mythic Beasts [21] mounted in a laser cut frame inside a 19inch rack, (c) Pi Cloud [4] built using 3D printed plastic and metal stand-offs, (d) Prototype FRμIT Cluster of 6 nodes built using the Pi-Stack interconnecting board [22].

needs a PoE breakout board and requires more expensive network switches. Fully managed PoE switches cost considerable more than unmanaged equivalents, but allow each node to be remotely power cycled.

Other solutions have used separate infrastructure for power and Ethernet. The approach first used in IridisPi was to use a separate power supply for each node. This has the advantage that a PSU failure will only affect a single board, however it is not a compact solution. An alternative used by some clusters, such as the Beast [27], is multiple-output DC power supplies that reduce the space requirements. The third approach, used by Beast 2, is to distribute DC power around the system and to have local power supplies for each node [28]. This reduces the required complexity of the power supply for each node, and by distributing a higher voltage (12V) the currents required are reduced, lowering cable losses. There are two main approaches to networking, using a few large switches centrally within the cluster, or using multiple small (circa 8 port) switches distributed around the cluster. In both cases, cable management can be a significant challenge.

The Federated Raspberry Pi μ-Infrastructure Testbed (FRμIT) project has developed an interconnect board to address some of these requirements when building a Raspberry Pi, or hardware form-factor compatible, SBC cluster. The Pi-Stack interconnect [22], shown in Fig. 2(d), adds independent hardware power management and monitoring to each SBC node in the stack and provides a dedicated RS-485 channel for infrastructure management-level communication. Each Pi-Stack board is allocated an unique address on the RS-485 bus, and a total of 16 Pi-Stack interconnect boards can be combined into a single stack, each supporting two Raspberry Pi compatible SBCs. This limits each stack to 32 nodes, providing a suitable power supply is available. This limit can be overcome by combining multiple stacks to form a single cluster. The RS-485 management interface provides instantaneous current and voltage monitoring for each node, as well a customizable heartbeat to ensure the node OS is functioning. Individual nodes can be notified of an impending reset or power-off through an interrupt, giving them sufficient notice to cleanly shutdown. Each node can be power cycled independently, so powering up the cluster can be staged to avoid peak current power supply load issues. This also facilitates disabling unwanted compute nodes and improves thermal management to avoid overheating issues. The SBC nodes in a stack are physically connected and supported by four conductive threaded stand-offs. Two of these supply power to the Pi-Stack interconnect, and hence to the compute nodes; the other two are used for the RS-485 management-level communication. Using the stack *chassis* for power and communication drastically reduces the cabling required, although the Pi-Stack has headers and isolation jumpers to support cabling if required. Since the Pi-Stack interconnect has on-board power regulators, we can power the stack through the chassis using a wide range of voltages (12V–24V). This is important to reduce the current as the number of nodes in the stack gets larger, and also makes connections to batteries or Photovoltaics (PVs) simple.

Commercial products to simplify the creation of Raspberry Pi clusters are now being produced. One example is the ClusterHAT [29] which enables 4 Raspberry Pi Zero (or Zero W) nodes to be mounted on top of a Raspberry Pi 2/3 that provides network (via USB gadget mode) and power. The central Raspberry Pi acts as a coordinator managing traffic, and can power down the Raspberry Pi Zero boards when they are not needed. Another product is the BitScope Blade [30]. The Blade allows either 1, 2 or 4 Raspberry Pi boards to be mounted on a stack-able back plane powered with 9–48 volts, and the provides local 5 volt power supplies, this significantly reduces the current needed through the back plane. This product was used in the creation of the cluster at Los Alamos [31]. As well as the development of products designed to make the creation of bespoke clusters easier it is possible to buy entire SBC clusters of up to 20 nodes off the shelf (not limited to just Raspberry Pi-based clusters) [32].

SBCs have facilitated the creation of multiple clusters at a variety of scales, but have some disadvantages compared to a traditional cluster. These include:

- limited computing resources (CPU, memory, storage) per node;
- increased hardware failure rate;
- high network latency;
- the need for architecture specific compilers and non-standard tooling; and
- potentially mixed hardware (where different SBCs are used)

The first two of these disadvantages can be partially mitigated by expanding the cluster, a proposition made possible by the low-cost and power consumption of the nodes. By increasing the size of the cluster the effect of a single node failing is reduced, and the low-cost of nodes means replacement of failed nodes is inexpensive. The high network latency is harder to mitigate against. This issue is particular prevalent on the earlier boards in the Raspberry Pi series of SBC, which use a USB2 100MBps network adapter rather than a direct connection to the processor. The Raspberry Pi 3B+ addresses this by using a gigabit adapter, but it is still limited by the USB2 connection. Other boards have chosen a USB3 gigabit network adaptor [33] which should reduce the network latency for the cluster.

The design and creation of the cluster is only the first step in using a SBC cluster. In addition to the relatively high failure rate of SBC hardware every cluster needs ongoing software maintenance, both to ensure security vulnerabilities are patched in a timely manner and to make sure that new packages are available. The differences in per node storage, RAM, and processing power in SBC clusters when compared to traditional high performance compute nodes means that different tools and techniques are needed to manage this.

Table 4
Example Raspberry Pi clusters.

Name	Year	Owner	Hardware	Use cases
Pi Co-location [16,34]	2013	PC Extreme (NL)	2500 Pi1	Raspberry Pi hosting provision (obsolete)
National Laboratory R&D Cluster [31]	2017	Los Alamos National Lab (US)	750 Pi3B	R & D prior to running on main cluster
Bolzano Cloud Cluster [35]	2013	Free University of Bolzano (IT)	300 Pi1	Education, research, & deployment in developing countries
SeeMore [6]	2015	Virginia Tech (US)	256 Pi2	Art installation, education
The Beast [27]	2014	Resin.io (UK)	120 Pi1	Demonstrating/testing distributed applications
The Beast 2.0 [28]	2017	Resin.io (UK)	144 Pi2	Demonstrating/testing distributed applications
Pi Hosting [17,21]	2016	Mythic Beasts (UK)	108 Pi3B per 4U rack	Raspberry Pi hosting provision
Raspberry Pi Cloud [4]	2013	University of Glasgow (UK)	56 Pi1 & 14 Pi2	Research and Teaching light-weight virtualization
Bramble [36]	2015	GCHQ (UK)	66 Pi1	Internal teaching tool
Iridis-Pi [3]	2013	University of Southampton (UK)	64 Pi1	Education
Wee Archie Green [37,38]	2015	University of Edinburgh (UK)	19 Pi2	Education Outreach

4. Use cases

This section outlines the broad domains in which SBC clusters might be deployed. In some cases, researchers have only identified a potential use case, in order to motivate the construction of SBC clusters. In other cases, researchers report prototype deployments and initial evaluation results. We classify use cases into various categories. This is not intended to be an exhaustive list for characterizing future use cases; it is simply a convenient means of grouping use cases we have identified from the current literature.

4.1. Education

Construction of the earliest Raspberry Pi clusters, for instance at Southampton [3], Glasgow [4] and Bolzano [35], was to provide a hands-on educational and learning experience. Educational exposure to *real* clusters is difficult and often limited to using a handful of PCs or simulated environments. SBC clusters are low-cost, quick to build, and offer challenges around the physical construction. They are typically under 100 nodes, are connected by Ethernet, powered via USB hubs, and run a custom Linux-based software stack.

These SBC clusters or micro-scale data centers [4] can provide many relevant experiences for students in terms of hardware installation (routers, racks, blades, power, and network cabling) and software frameworks (e.g., Linux, containers, MPI, Docker, Kubernetes). These are typical components in state-of-the-art data centers. Not all data center aspects are replicated in the SBC clusters, for example network bandwidth, computational power, memory, and storage capacities are considerably lower, limiting the cluster to small-scale jobs. Node power management and cluster network topologies are not replicated, although some work has investigated network topologies [35].

Such micro data centers have been used at various universities for undergraduate classes and projects. To date, the emphasis appears to be on the experience gained in building the cluster rather than subsequent operation [39,40], although the SeeMore cluster [6] is primarily intended to be a modern art installation. An informal comparative showcase of micro clusters took place at a recent Computing Education conference [41]. To the best of our knowledge, there have not yet been any systematic studies of the pedagogical benefits of hands-on experience with micro clusters.

The Edinburgh Parallel Computing Centre (EPCC) built Wee Archie [38], an 18-node Raspberry Pi 2 cluster, aiming to demonstrate applications and concepts relating to parallel systems. Several Beowolf clusters have targeted education, using a variety of SBC configurations [41] including 2-nodes ODROID, 6-nodes NVIDIA Jetson TK1, and 5-mixed-nodes of 1 NVIDIA Jetson TK1 and 4 Raspberry Pis. Introducing High Performance computing (HPC) and using the Message Passing Interface (MPI) library to develop the applications is addressed by several clusters [3,42]. Finally, we note that it is not just academic institutions creating Raspberry Pi clusters for educational purposes, for example GCHQ built a cluster of 66 nodes as a teaching tool for their internal software engineering community [36].

4.2. Edge compute

Many sensor network architectures form a star topology, with sensors at the edge and storage and compute power in the middle (often, with cloud-based storage and compute). The advantages of this are that the data is all stored and processed in one location, making management and post processing of the data less complex. The disadvantage is that all the data has to be transmitted from the data producers to the centralized storage and compute resources. On large scale deployments this means transmitting across bandwidth constrained wireless networks or expensive satellite uplinks. There is a need to be more efficient with data bandwidth, transmitting only the data that is required. Furthermore by processing the data at its origin we can reduce the bandwidth requirements by only transmitting processed data; for example threshold alerts or cumulative data. This requires computational power to be connected to the data producers, and is referred to as Edge Compute [43,44]. In cases where the compute is moved closer to the data producers, but is not attached to them, it is referred to as Fog Compute.

The system becomes more decentralized as the intelligence is pushed out towards the edge, improving latency as the system can react to local events [45]. Latency reduction is important for virtual/augmented reality and gaming applications, and is a key priority for 5G networks. As well as improving network utilization by reducing traffic, Edge Compute can potentially improve the system reliability, for example by enabling data producers to operate through connectivity outages. Further, minimizing data transfer can improve user privacy [46], for example by keeping personal or identifiable information local and transmitting only anonymized data.

The overhead of setting up a cluster is greater than that of using a single machine, and the use of more hardware increases the probability of hardware failures. There are two key reasons to consider SBC clusters for edge compute, rather than purchasing a single machine or traditional cluster. Firstly, the SBC cluster can be sized closer to the workload demands, keeping utilization high, reducing costs and potentially keeping power requirements lower. The cost increment per node to expand the cluster is also much

lower than adding nodes to a traditional cluster. This is particularity of benefit where a large number of edge compute clusters are deployed. Second, clusters can be configured to be more resilient to hardware failures by offering compute and storage redundancy.

One of the main drivers for edge computing is the ever increasing popularity of Cyber Physical Systems (CPS) and the Internet of Things (IoT), with some predictions stating that there will be 30 billion IoT devices by 2020 [47]. This may be an over estimate, but it demonstrates a huge demand and opportunity for embedded systems and hence SBC based devices. There is a need to decentralize the storage, compute, and intelligent of systems if they are to scale to such large volumes.

4.3. Expendable compute

Low-cost SBCs introduced the concepts of single-use, disposable, and expendable computing. This has the potential to extend compute resources into hostile, high risk environments, for example ad-hoc networks and systems *in the wild* on volcanoes [48] and in rainforests [49], but will require responsible deployment to avoid excessive pollution and waste.

We propose that SBC clusters might also be considered as expendable compute resources, providing further opportunities. These SBC clusters can provide significant computational power at the edges of IoT deployments, for example to drastically reduce bandwidth requirements [7] and promote autonomous, fault tolerant systems by pushing compute and hence logic out towards the edge. These expendable clusters are a new class of compute to facilitate IoT architectures.

4.4. Resource constrained compute

SBC clusters provide a new class of computational resource, distinguished by their low-cost. This provides an opportunity to place compute clusters in new and imaginative places, in particular where physical size and power consumption is restricted, for example backpack contained clusters, or solar powered deployments.

Many SBC clusters use low-power ARM based SoCs that lend themselves to high core densities and are generally considered low-power. Since many Edge Compute architectures require remote deployments powered via renewable energy there is a need to measured the trade-offs between performance and energy-consumption; often measured in MFLOPS/Watt. For example, currently the top of green supercomputer [50] is Shoubu system B, at the Advanced Center for Computing and Communication, RIKEN, which provides 17.009 GFLOPS/Watt. There have been numerous studies which benchmark SBC power consumption:

- A benchmark of ten types of single board computer [51] using Linpack [52] and STREAM [53] showed the Raspberry Pi 1 Model B and B+ produced 53.2 and 86.4 MFLOPS/Watt respectively.
- A benchmark of three different configurations of 8-nodes clusters using Linpack [52] showed the Raspberry Pi 1, Banana Pi, and Raspberry Pi 2 clusters provided 54.68, 106.15, and 284.04 MFLOPS/Watt respectively [54].
- A 4-node cluster [55] using Parallela boards [56], each with a single ARM-A9 CPU and a single 16-core Epiphany Coprocessor, was used to developed a floating-point multiplication application to measure the performance and energy consumption. Further power measurements or MFLOPS/Watt results were not presented.
- A comparative study of data analytics algorithms on an 8-node Raspberry Pi 2 cluster and an Intel Xeon Phi accelerator [57] showed the Raspberry Pi cluster achieves better energy efficiency for the Apriori kernel [58]. The opposite is true for *k*-means clustering, as the Xeon Phi is more energy efficient [59]. For both algorithms, the Xeon Phi gives better absolute performance metrics.

We can see from this that the MFLOPS/Watt still needs improvement and most of these benchmarks do not include the energy required to cool their clusters as they are passively cooled; larger deployments will probably require active cooling.

As the CPUs on the SBCs become more advanced there is a trend for the MFLOPS/Watt to increase. Better utilization of power will extend the reach of the SBC clusters, making them better candidates for renewable energy or battery powered deployments. We predict that key areas where SBC clusters can make an impact will rely on limited power sources, making this an important attribute. The current SBCs go some way towards efficient power usage, and in the future we can expect this to improve; mainly due to the inclusion of ARM based CPUs and ability to be passively cooled.

4.5. Next-Generation data centers

Traditionally SBCs use 32-bit processors, in contrast to most data centers that use 64-bit processors. Some more recent SBCs designs are based on 64-bit SoCs, as shown in Table 2. Data centers have traditionally used Intel/AMD based servers, but more recently some ARM servers exist such as HP Moonshot [60] servers. The Open Compute Community is also working on ARM based designs in conjunction with Microsoft and Cavium [61].

Projects such as Euroserver [62] are leading the initiative for ARM-based HPC clusters, with a key motivation being to improve data center power efficiency. ARM cores may provide one order of magnitude less compute resource than Intel [63], but it is possible to pack ARM cores much more densely [64], which means exascale sized platforms are potentially viable. We propose that ARM-based SBC clusters make reasonable low-cost testbeds to explore the next generation of ARM-based data centers.

The Mont Blanc project has investigated how the use of 64-bit ARM cores can be used in order to improve the efficiency in future HPC systems [65]. The SpiNNaker project has identified that the simulation of the human brain using traditional supercomputers places extremely large power demands on the electrical infrastructure, and so has developed SpiNNaker chips that require substantially less power per second per neuron simulated by using multiple low power processors instead of fewer more powerful processors [66].

Next-generation data centers may offer physical infrastructure as a service, rather than using virtualization to multiplex cloud users onto shared nodes. This reduces the software stack and potentially improves performance by providing cloud users with physical access to bare metal hardware. Some cloud hosting providers already sell data center hosted, bare metal, Raspberry Pi hardware as a service [21]. By using SBC hardware we can explore the capabilities of next generation physical infrastructure as a service, either as standalone single nodes or as complete cluster configurations.

4.6. Portable clusters

Portable clusters are not new. They tend to comprise specially ruggedized units, ranging in size from a standard server rack to a whole shipping container. They are often deployed into environments with either mains or generator power supplies. We believe that SBC clusters give rise to truly portable clusters, for example in a backpack. The reduced power requirements of SBC cluster mean that portable clusters can operate using batteries or renewable energy, and can be powered on-demand.

This is advantageous, for example, for first respondents to large scale disaster recovery, since the clusters can be carried by drones, aircraft, ground vehicles, or personnel, and can operate with minimal supporting infrastructure. It also gives rise to more mobile and dynamic Edge Compute topologies that do not rely on fixed location sensors. Some of the advantages of edge compute in emergency response scenarios are discussed in [67].

5. Single board cluster management

The purpose of cluster management is to transform a set of discrete computing resources into a holistic functional system according to particular requirement specifications, and then maintain its conformance for any specification or environment change [68]. These resources include, but are not limited to, machines, operating systems, networking, and workloads. There are at least four factors that complicate the problem: dependencies between resources, specification changes, scale, and failures.

Section 3 reviewed the hardware underlying SBC clusters and Section 4 outlined the range of applications running on SBC clusters, this section concentrates on the *software infrastructure* required to manage SBC clusters. Fig. 3 shows a schematic diagram of a cluster, with associated cross-cutting management tasks on the left hand side. The rest of this section will consider typical software deployed at each level of the infrastructure stack, along with management activity.

5.1. Per-node operating systems

Most SBCs are capable of running a range of mainstream operating system variants. Some support single-user, single-node instances, such as RISC OS. Others are IoT focused, and have limited traction in the broader community, such as Riot-OS [63], Contiki OS [69], and Windows 10 IoT Core [70]. The most popular SBC operating system is Linux, however it usually requires specific kernel patches for hardware device drivers that are only available in vendor repositories.

End-user targeted Linux distributions, such as Raspbian [71] or Ubuntu [72], are bloated with unnecessary software packages. These increase the size and complexity of cluster updates and can pose a security risk by expanding the potential attack surface.

In contrast, Alpine Linux [73] is a lightweight distribution. The disk image requires around 100MB rather than the multiple GBs of Raspbian. Alpine has a read-only root partition with dynamic overlays, which mitigates problems with SD card corruption. It supports security features like Position Independent Executables to prevent certain classes of exploits.

Customized OS generators such as Buildroot [74] and OpenEmbedded/Yocto [75,76] provide automated frameworks that build complete, customized, embedded Linux images, and can target multiple hardware architectures. They use different mechanisms to accomplish this [77]. Buildroot has a simple build mechanism using interactive menus, scripts, and existing tools such as `kconfig` and `make`. Although it is simple to build a small image, it is necessary to rebuild the complete image to update the OS since there is no package management. OpenEmbedded/Yocto addresses this problem by applying partial updates on an existing file system using a layer mechanism based on libOSTree [78]. This allows low bandwidth over-the-air updates [79,80] with shorter deployment times and support for rollback. The main drawback of these OS generators is their increased configuration complexity.

Another alternative is LinuxKit [81], a toolkit for building a lean OS that only provides core functionality, with other services deployed via containers. The project is still immature (released in April 2017) and only supports x86 and 64-bit ARM platforms.

In terms of OS image deployment on each node, early systems worked by manually flashing the selected image onto persistent memory such as an SD card. Cox et al. [3] note that this is time consuming and unsuitable for large scale clusters. Abrahamsson et al. [35] copy a standard OS image onto each SD card; the image includes boot scripts that automatically requests configuration files from a known master node, deploys these files to initialize per-node resources, and performs the registration. Once a node has joined the cluster, then the manager can control the node remotely to run workloads or un-register the node.

A more recent, superior, approach relies on network boot [82,83], allowing each node to be diskless. This simplifies OS updates and reduces the number of resources to be maintained. SBC hardware often does not support network boot, or only works with limited boot protocols at best, and network boot often suffers from significant security risks. iPXE [84] is a likely candidate for robust network boot, and supports ARM platforms.

5.2. Virtualization layer

Virtualization provides multi-tenancy, resource isolation, and hardware abstraction. These features are attractive for large scale data centers and utility computing providers. Micro data centers can also benefit from virtualization techniques.

Hypervisor based virtualization, such as Microsoft's Hyper-V [86], Xen [87], and VMware [88] tends to be heavy-weight and unsuitable for resource limited SBCs. Linux containers facilitate a more lightweight approach to virtualization. Tso et al. [4] run OS-level virtualized workloads based on Linux's `cgroups` functionality. This has much lower overhead than full virtualization, enabling a Raspberry Pi node to run several containerized workloads simultaneously. Popular lightweight virtualization frameworks include Docker [89] and Singularity [90]. These are suitable for deployment on resource constrained SBC clusters. Morabito [91] reports a comprehensive performance evaluation of containerized virtualization, and concludes this abstraction has minimal performance overhead relative to a bare-metal environment.

Singularity provides a *mobility of compute* solution by wrapping operating system files into a container image, so that the application runs as the user who invoked it, preventing a malicious application from obtaining root access from within a container. Unlike Docker, Singularity can run on kernels before Linux kernel version 3.10 without modification.

Some SBC clusters take advantage of unikernels, for example MirageOS [92], to enable a secure, minimal, application footprint on low power devices. Unikernels can run on hypervisors or directly on bare metal, and only support the subset of OS features required by the application, drastically reducing the host OS footprint.

There are many useful management tools for container based workloads that assist with the deployment and life-cycle of containers, for example Mesos [93], Docker Swarm [94], and Kubernetes [95,96].

5.3. Parallel framework

A parallel framework presents a cluster of nodes as a unified, logical entity for user-level applications. This is appropriate for scientific, big data, and utility computing domains. Such frameworks require each node to run a local management daemon or instance of a runtime system.

Many SBC clusters are designed to run HPC workloads based on MPI [3,38,85]. In order to improve performance, software libraries might need to be recompiled since distributed packages are generally not specialized for SBC nodes. Due to per-node memory restrictions, the MPI jobs are generally small scale example applications rather than calculations of scientific interest.

Some clusters run virtualized workloads, to provide utility computing services. The Bolzano cluster [35] hosts OpenStack, although the authors report poor performance because this complex framework requires more resources than SBCs can provide.

Schot [97] uses eight Raspberry Pi 2 boards to form a miniature Hadoop cluster, with one master and seven slave nodes. He uses standard tools including YARN for workload management and HDFS for distributed storage. This cluster uses DietPi, a slimmed down Debian OS variant. Test results show that the cluster can

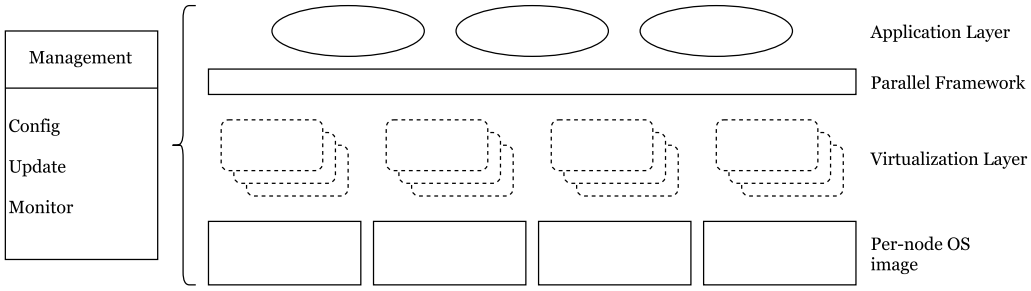


Fig. 3. Schematic diagram of typical Single Board Computer (SBC) cluster configuration.

utilize more than 90% of the 100 Mbps Ethernet bandwidth when transferring data from node to node. This is a single tenancy cluster, effectively a platform for performance tests. There is no capability for failure recovery, since a failing master node is not automatically replaced. Other SBC clusters run Hadoop or Spark workloads [3,98,99] with similar configurations. Major frustrations appear to be insufficient per-node RAM, high latency network links, frequent SD card corruption leading to HDFS failure, and poor performance of the underlying Java virtual machine.

Many of these difficulties are intrinsic to SBC clusters, suggesting that industry standard parallel frameworks will require significant re-engineering effort to make them effective on this novel target architecture.

5.4. Management

A cluster management system reduces repetitive tasks and improves scalability. Management tasks cut across the cluster stack, as shown in Fig. 3.

Configuration management allows cluster administrators to deploy software across the cluster, either for implementing new requirements, fixing system bugs, or applying security updates. Standard cluster management tools include Chef [100], Puppet [101], Ansible [102], and Salt [103]. These are standard tools, and do not require modifications for use in existing SBC contexts. They are generally useful for deploying application level software on nodes that are already running a base OS. Other management tools can handle provisioning at lower levels of the stack. For instance, the Metal as a Service (MAAS) framework handles automation for bare metal node instances [104].

A key problem is node reachability when SBCs are deployed behind NATs or firewalls. Existing configuration frameworks assume always-connected nodes in a local area network scenario. Section 6 discuss this further.

An update system is critical to any cluster because all nodes must be updated regularly. A traditional approach involves updating the root partition and then rebooting the machine when it is needed. If an error (for example an I/O error) occurs during update this can make the machine unable to boot properly due to a corrupted root partition. Several frameworks [105,106] address this problem by having two root partitions, one has the current root filesystem and the other has a newer version. Whenever an update error occurs, the machine can easily rollback by booting from an older version partition. This also allows Over-The-Air (OTA) updates since the target partition is not being used by the running system.

Relying on a single server to provide the updates is not an ideal solution, in particular for a large cluster, because all nodes are likely to download updates at the same time, potentially overloading the server with an overwhelming demand for bandwidth. One alternative approach [107] assigns a randomized delay for each node when it starts the download. By adjusting the randomization interval, we can adapt the update mechanism to the scale of the system. This approach does not solve a single point of failure problem. Another approach would be to balance download requests across multiple mirror servers, which has been the main solution of most Linux distributions. An alternative might be employing a peer-to-peer update mechanism to optimize network bandwidth efficiency by allowing a client to download from other clients [108,109].

Other cluster management facilities are important, including monitoring & analytics, resource scheduling, service discovery, and identity & access management. These standard cluster services must be configured appropriately for individual use cases. As far as we are aware, there are no special requirements for SBC clusters.

6. The future of the SBC cluster

We see a strong future for SBC clusters in two primary areas. First, and most importantly, SBC clusters are an ideal platform for Edge and Fog computing, CPS, and the IoT. These all require customizable, low-power, and ubiquitous deployment of compute nodes with the ability to interact with the environment. SBC clusters are well suited to this environment, with the Raspberry Pi being an exemplar of the type of devices considered, but we expect to see an increasing transition to SBC platforms that are engineered for robust long-term deployments, rather than hobbyist projects. For instance, certain types of neural networks [110,111] are being adapted to run effectively on SBC clusters, with low power constraints, limited memory usage, and minimal GPU processing.

Secondly, we expect continued use of SBC clusters to support educational activities and low-end hosting services. To maintain relevance, educators must teach cluster computing – the era of stand-alone machines is at an end – although building full-fledged data centers is beyond the means of most academic institutions. SBC clusters still have a powerful role to play in educating the next generation, as *scale model* data centers, but increasingly to teach Edge computing, CPS, and the IoT. Following from this, we expect to see a rise in low-end hosting services that allow hosting on a dedicated SBC rather than a virtual machine, catering to those educated on SBCs such as the Raspberry Pi.

A consequence of these developments will be increasing heterogeneity in terms of devices, deployment environments, and applications of SBC clusters, coupled with use in increasingly critical applications and infrastructure. This has implications for SBC hardware development and for the supporting software infrastructure.

On the hardware side, we expect to see divergence in new SBC designs with some becoming specialized for cluster deployments, with a focus on compute and network performance, while others support increasingly heterogeneous I/O interfaces and peripheral devices. SBC designs will become more robust and gain remote management features, for example, higher-quality flash storage, network boot, and PoE.

We expect the range of hardware configurations to increase as new vendors enter the market, and as devices are increasingly customized to particular applications. The implication of this is that

there will likely be no standard device configuration that can be assumed by SBC operating systems and management tools. Rather, there will be a common platform core, that is device independent, with a range of plug-ins and devices drivers for the different platforms. A key challenge will be providing stable and flexible device APIs so the core platform can evolve without breaking custom and/or unusual hardware devices and peripherals. Systems such as Linux provide a stable user-space API, but do not have a good track record of providing a stable in-kernel device driver API.

In terms of the software and management infrastructure, the heterogeneity of SBC cluster hardware, deployment environments, and applications forces us to consider issues that do not occur in traditional data center networks. Specifically, SBC cluster deployments may have no standard device hardware configuration; will run on devices that tend to be relatively underpowered, and may not be able to run heavy-weight management tools due to performance or power constraints; and will run on devices that have only limited network access due to the presence of firewalls, Network Address Translation (NAT), or intermittent connectivity and that cannot be assumed to be directly accessible by a management system.

The issue of limited network connectivity is a significant challenge for the management infrastructure. Traditional data centers are built around the assumption that hosts being managed are either directly accessible to the management system, or they can directly access the management system, and that the network is generally reliable. This will increasingly not be the case for many SBC cluster deployments. There are several reasons for this:

1. devices will be installed in independently operated residential or commercial networks at the edge of the Internet, and hence will be subject to the security policies of those networks and be protected by the firewalls enforcing those policies.
2. since they are in independently operated network devices may be in different addressing realms to the management system, and traffic may need to pass through a NAT between the device being managed and the management system.
3. devices may not always be connected to the Internet, perhaps because they are mobile, power constrained, or otherwise have limited access.

Traditional cluster management tools fail in these environments since they do not account for NAT devices and partial connectivity, and frequently do not consider intermittent connectivity. Tools need to evolve to allow node management via indirect, peer-to-peer, connections that traverse firewalls that prohibit direct connection to the system being managed. They must also incorporate automatic NAT traversal, building on protocols such as STUN and ICE [112,113] to allow NAT hole-punching and node access without manual NAT configuration (as needed by peer-to-peer management tools such as APT-P2P [114] and HashiCorp Serf today [115]). Existing cluster management tools scale by assuming a restricted, largely homogeneous, network environment. This is not the typical deployment environment for SBC clusters. They will increasingly be deployed in the wild, at the edge of the network, where the network performance and configuration is not predictable. Management tools must become smarter about maintaining connectivity in the face of these difficulties to manage nodes to which there is no direct access.

Finally, there are significant social and legal implications to the use of massively distributed SBC cluster platforms. As noted above, developing management tools to work within the constraints of different security and addressing policies is one aspect of this, but there are also legal implications of managing a device that may be in a different regulatory environment, on behalf of a user subject to different laws. The implications for trust, privacy, security, liability, and data protection are outside the scope of this paper, but are non-trivial. They will become increasingly critical as personal data migrates into Edge compute platforms based on SBC clusters, and as those platforms control increasingly critical infrastructure.

7. Conclusion

In this paper we have shown that SBCs are a computational game changer, providing a new class of low-cost computers. Due in part to their low-cost and size they are utilized for a wide variety of applications. This popularity has ensured that newer and more advanced SBCs are constantly being released. We have shown that often these SBCs are used to build clusters, mainly for educational purposes. These clusters are well suited to educational applications due to their low-cost, but they also make good testbeds for some newer data center technologies, for example high core-density and ARM based architectures. In IoT architectures there is a move away from a centralized compute resource, to an architecture where the computational power is pushed out closer to the edge, for example near data generating sensors. SBC clusters are an ideal candidate for Edge Compute because of their power requirements and size. It is also possible to power-off some, or all, of an SBC cluster, powering on nodes only when computational resources are required, thus saving power and coping with burst computational demands, for example audio, video or seismic data processing based on trigger events. We identify that the maturity and advancing features in SBCs will translate into more powerful, optimized, and feature rich SBC based clusters.

We have identified multiple use cases for SBC clusters and see a strong future, especially as more advanced SBCs become available. Current cluster management technology has limitations for Edge Compute and the physical cluster construction is currently rather bespoke. Through the FRμIT project we are currently working on tools and techniques to simplify the software stack, support containerization and facilitate the update and maintenance of large numbers of distributed Edge Compute clusters. The FRμIT project also manages the physical construction of Raspberry Pi compatible clusters, through the introduction of a Pi-Stack interconnect board which includes both power and OS management capability, as shown in Fig. 2(d); this reduces cabling, focuses air flow and adds independent power monitoring & isolation.

Small, portable, low-power clusters are the key to improving IoT architectures and overcoming limited or costly bandwidth restriction. As processor performance improves and SBC hardware advances, so will the clusters upon which they are based, unlocking new capabilities. If small, low cost, portable clusters are to become mainstream, we acknowledge that there is an overhead to convert applications to utilize clusters, rather than a single multi-core processor. We believe that as CPU speeds have plateaued and performance is achieved by increasing the number of cores, more applications will support multi-core and increasingly support clusters. SBC based clusters are a new and distinct class of computational power. Although in their infancy, they have the potential to revolutionize the next generation of sensor networks, and act as a fantastic exploratory tool for investigating the next generation of data centers.

Acknowledgment

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/P004024/1.

References

[1] E. Larson, Introduction to GumStix computers, Tech. Rep., University of West Florida, 2007.
[2] E. Upton, G. Halfacree, Raspberry Pi User Guide, fourth ed., Wiley, 2016.
[3] S.J. Cox, J.T. Cox, R.P. Boardman, S.J. Johnston, M. Scott, N.S. O'Brien, Iridis-pi: a low-cost, compact demonstration cluster, Cluster Comput. 17 (2) (2014) 349–358. http://dx.doi.org/10.1007/s10586-013-0282-7.
[4] F.P. Tso, D.R. White, S. Jouet, J. Singer, D.P. Pezaros, The Glasgow Raspberry Pi cloud: A scale model for cloud computing infrastructures, in: 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, IEEE, 2013, pp. 108–112. http://dx.doi.org/10.1109/ICDCSW.2013.25.
[5] P. Basford, G. Bragg, J. Hare, M. Jewell, K. Martinez, D. Newman, R. Pau, A. Smith, T. Ward, Erica the rhino: A case study in using Raspberry Pi single board computers for interactive art, Electronics 5 (3) (2016) 35. http://dx.doi.org/10.3390/electronics5030035.
[6] P. King, SeeMore - parallel computing sculpture, MagPi 40 (2015) 46–49.
[7] A. Sathiaseelan, A. Lertsinsrubtavee, A. Jagan, P. Baskaran, J. Crowcroft, Cloudrone: micro clouds in the sky, in: Proceedings of the 2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, 2016, pp. 41–44. http://dx.doi.org/10.1145/2935620.2935625.
[8] PR Newswire, Meet BeagleBone, the new $89 open source hardware platform, giving electronic enthusiasts a smaller, friendlier and more affordable treat, 2011. www.prnewswire.com/news-releases/meet-beaglebone-the-new-89-open-source-hardware-platform-giving-electronic-enthusiasts-a-smaller-friendlier-and-more-affordable-treat-132910373.html. (Accessed 13 June 2018).
[9] M. Keller, J. Beutel, L. Thiele, Demo Abstract: MountainviewPrecision Image Sensing on High-Alpine Locations, in: D. Pesch, S. Das (Eds.), Adjunct Proceedings of the 6th European Workshop on Sensor Networks, EWSN, Cork, 2009, pp. 15–16.
[10] K. Martinez, P.J. Basford, D. De Jager, J.K. Hart, Using a heterogeneous sensor network to monitor glacial movement, in: 10th European Conference on Wireless Sensor Networks, Ghent, Belgium, 2013.
[11] Single-board computer, 2017. https://en.wikipedia.org/wiki/Single-board_computer. (Accessed 13 June 2018).
[12] Sales soar above Commodore 64, Mag Pi 56 (2017) 8.
[13] Raspberry Pi Foundation, Raspberry Pi 3 Model B+, 2018. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. (Accessed 13 June 2018).
[14] Hard Kernel, Odroid-XU4, 2018. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825. (Accessed 13 June 2018).
[15] B. Varghese, N. Carlsson, G. Jourjon, A. Mahanti, P. Shenoy, Greening web servers: A case for ultra low-power web servers, in: International Green Computing Conference, IEEE, 2014, pp. 1–8. http://dx.doi.org/10.1109/IGCC.2014.7039157.
[16] PC Extreme, Raspberry Pi colocation, 2017. https://web.archive.org/web/20170217070549/https://www.pcextreme.com/colocation/raspberry-pi. (Accessed 13 June 2018).
[17] P. Stevens, IPv6 Only hosting, 2016. https://indico.uknof.org.uk/event/36/contribution/5/material/slides/0.pdf. (Accessed 13 June 2018).
[18] Up Board, UP2 Specification. 2016. http://www.up-board.org/wp-content/uploads/2016/05/UP-Square-DatasheetV0.5.pdf. (Accessed 13 June 2018).
[19] W. Stallings, Operating Systems, fourth ed., Prentice-Hall, 2000, p. 590.
[20] University of Southampton, Switching on one of the most powerful supercomputers, University of Southampton, 2013. www.southampton.ac.uk/news/2013/10/15-uos-one-of-the-most-powerful-supercomputers.page. (Accessed 13 June 2018).
[21] P. Stevens, RaspberryPiCloud, 2017. https://blog.mythic-beasts.com/wp-content/uploads/2017/03/raspberry-pi-cloud-final.pdf. (Accessed 13 June 2018).
[22] P. Basford, S. Johnston, Pi Stack PCB, 2017. http://dx.doi.org/10.5258/SOTON/D0379.
[23] N. Smith, 40-core ARM cluster using the NanoPC-T3, 2016. http://climbers.net/sbc/40-core-arm-cluster-nanopc-t3/. (Accessed 13 June 2018).
[24] N. Smith, 5 Node Cluster of Orange Pi Plus 2Es, 2016. http://climbers.net/sbc/orange-pi-plus-2e-cluster/. (Accessed 13 June 2018).
[25] N. Smith, Bargain 5 Node Cluster of PINE A64+, 2017. http://climbers.net/sbc/bargain-pine-a64-cluster/. (Accessed 13 June 2018).
[26] Antipasto Hardware, How to make a BeagleBoard Elastic R Beowulf Cluster in a Briefcase, 2010. http://antipastohw.blogspot.co.uk/2010/09/how-to-make-beagleboard-elastic-r.html. (Accessed 13 June 2018).
[27] A. Marinos, What would you do with a 120-Raspberry Pi Cluster? 2014. https://resin.io/blog/what-would-you-do-with-a-120-raspberry-pi-cluster/. (Accessed 13 June 2018).
[28] A. Davis, The evolution of the beast continues, 2017. https://resin.io/blog/the-evolution-of-the-beast-continues/. (Accessed 13 June 2018).
[29] C. Burton, ClusterHAT - Cluster HAT for Raspberry Pi Zero, 2016. https://clusterhat.com/. (Accessed 13 June 2018).
[30] Bitscope, BitScope Blade for Raspberry Pi, 2016. www.bitscope.com/product/blade. (Accessed 13 June 2018).
[31] N. Ambrosiano, C. Poling, Scalable clusters make HPC R&D easy as Raspberry Pi, 2017. www.lanl.gov/discover/news-release-archive/2017/November/1113-raspberry-pi.php. (Accessed 13 June 2018).
[32] PicoCluster, PicoCluster - Big Data in A Tiny Cube, 2017. https://www.picocluster.com/. (Accessed 13 June 2018).
[33] Globalscale Technologies Inc., ESPRESSObin, 2017. http://espressobin.net. (Accessed 13 June 2018).
[34] PCextreme, About us, 2017. https://www.pcextreme.com/about. (Accessed 13 June 2018).
[35] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkila, X. Wang, K. Hamily, S. Bugoloni, Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment, in: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, IEEE, 2013, pp. 170–175. http://dx.doi.org/10.1109/CloudCom.2013.121.
[36] GCHQ, GCHQ's Raspberry Pi 'Bramble' - exploring the future of computing, 2015. https://www.gchq.gov.uk/news-article/gchqs-raspberry-pi-bramble-exploring-future-computing. (Accessed 13 June 2018).
[37] A. Grant, N. Brown, Introducing Wee Archie, 2015. https://www.epcc.ed.ac.uk/blog/2015/11/26/wee-archie. (Accessed 13 June 2018).
[38] G. Gibb, Linpack and BLAS on Wee Archie, 2017. https://www.epcc.ed.ac.uk/blog/2017/06/07/linpack-and-blas-wee-archie. (Accessed 13 June 2018).
[39] P. Turton, T.F. Turton, Pibrain —a cost-effective supercomputer for educational use, in: 5th Brunei International Conference on Engineering and Technology, BICET 2014, 2014, pp. 1–4. http://dx.doi.org/10.1049/cp.2014.1121.
[40] K. Doucet, J. Zhang, Learning cluster computing by creating a Raspberry Pi cluster, in: Proceedings of the SouthEast Conference, in: ACM SE '17, 2017, pp. 191–194. http://dx.doi.org/10.1145/3077286.3077324.
[41] J.C. Adams, J. Caswell, S.J. Matthews, C. Peck, E. Shoop, D. Toth, J. Wolfer, The micro-cluster showcase: 7 inexpensive beowulf clusters for teaching PDC, in: Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16, ACM Press, New York, New York, USA, 2016, pp. 82–83. http://dx.doi.org/10.1145/2839509.2844670.
[42] A.M. Pfalzgraf, J.A. Driscoll, A low-cost computer cluster for high-performance computing education, in: Electro/Information Technology (EIT), 2014 IEEE International Conference on, IEEE, 2014, pp. 362–366. http://dx.doi.org/10.1109/EIT.2014.6871791.
[43] S. Helmer, S. Salam, A.M. Sharear, A. Ventura, P. Abrahamsson, T.D. Oyetoyan, C. Pahl, J. Sanin, L. Miori, S. Brocanelli, F. Cardano, D. Gadler, D. Morandini, A. Piccoli, Bringing the cloud to rural and remote areas via cloudlets, in: Proceedings of the 7th Annual Symposium on Computing for Development - ACM DEV '16, ACM Press, New York, New York, USA, 2016, pp. 1–10. http://dx.doi.org/10.1145/3001913.3001918.
[44] C. Pahl, S. Helmer, L. Miori, J. Sanin, B. Lee, A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops, FiCloudW, IEEE, 2016, pp. 117–124. http://dx.doi.org/10.1109/W-FiCloud.2016.36.
[45] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing - MCC '12, ACM Press, New York, New York, USA, 2012, p. 13. http://dx.doi.org/10.1145/2342509.2342513.
[46] C. Perera, S.Y. Wakenshaw, T. Baarslag, H. Haddadi, A.K. Bandara, R. Mortier, A. Crabtree, I.C. Ng, D. McAuley, J. Crowcroft, Valorising the iot databox: creating value for everyone, Transactions on Emerging Telecommunications Technologies 28 (1) (2017). http://dx.doi.org/10.1002/ett.3125.
[47] A. Nordrum, Popular internet of things forecast of 50 billion devices by 2020 is outdated, 2016. https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated. (Accessed 13 June 2018).
[48] D. Moure, P. Torres, B. Casas, D. Toma, M. Blanco, J. Del Río, A. Manuel, Use of low-cost acquisition systems with an embedded linux device for volcanic monitoring, Sensors 15 (8) (2015) 20436–20462. http://dx.doi.org/10.3390/s150820436.
[49] E. Yoneki, Demo: RasPiNET: decentralised communication and sensing platform with satellite connectivity, in: Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks - CHANTS '14, ACM Press, New York, New York, USA, 2014, pp. 81–84. http://dx.doi.org/10.1145/2645672.2645691.
[50] Green500 List - November2016, 2017. https://www.top500.org/green500/lists/2016/11/ (Accessed 13 June 2018).
[51] M.F. Cloutier, C. Paradis, V.M. Weaver, Design and analysis of a 32-bit embedded high-performance cluster optimized for energy and performance, in: 2014 Hardware-Software Co-Design for High Performance Computing, IEEE, 2014, pp. 1–8. http://dx.doi.org/10.1109/Co-HPC.2014.7.
[52] A. Petitet, R. Whaley, J. Dongarra, A. Cleary, HPL–A portable implementation of the high–performance Linpack benchmark for distributed–memory computers. 2004. www.netlib.org/benchmark/hpl. (Accessed 13 June 2018.

[53] J.D. McCalpin, Memory bandwidth and machine balance in current high performance computers, IEEE Comput. Soc. Tech. Committee Comput. Archit. Newslett. (1995) 19–25.
[54] C. Baun, Performance and energy-efficiency aspects of clusters of single board computers, Internat. J. Distrib. Parallel Syst. 7 (2/3/4) (2016) 4.
[55] M.J. Kruger, Building a Parallella Board Cluster, (Bachelor of Science Honours thesis), Rhodes University, Grahamstown, South Africa, 2015.
[56] ParallellaBoard, 2017. https://www.parallella.org/board/. (Accessed 13 June 2018).
[57] J. Saffran, G. Garcia, M.A. Souza, P.H. Penna, M. Castro, L.F.W. Góes, H.C. Freitas, A low-cost energy-efficient Raspberry Pi cluster for data mining algorithms, in: Euro-Par 2016: Parallel Processing Workshops, Springer International Publishing, 2017, pp. 788–799.
[58] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, 1994, pp. 487–499.
[59] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892. http://dx.doi.org/10.1109/TPAMI.2002.1017616.
[60] E. Turkel, Accelerating innovation in hpc, in: Proceedings of the ATIP/a*CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?, in: ATIP '12, A*STAR Computational Resource Centre, Singapore, Singapore, 2012, pp. 26:1–26:28. http://dl.acm.org/citation.cfm?id=2346696.2346727.
[61] Cavium Inc., Cavium collaborates with microsoft to Demonstrate ThunderX2 Platform Compliant with Microsoft's Project Olympus Specifications, 2018. https://www.prnewswire.com/news-releases/cavium-collaborates-with-microsoft-to-demonstrate-thunderx2-platform-compliant-with-microsofts-project-olympus-specifications-300616403.html. (Accessed 13 June 2018).
[62] Euro Server Project, Green computing node for European micro-servers, 2016. http://www.euroserver-project.eu/. (Accessed 13 June 2018).
[63] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, T. Schmidt, RIOT OS: Towards an OS for the Internet of Things, in: 2013 IEEE Conference on Computer Communications Workshops, IEEE, 2013, pp. 79–80. http://dx.doi.org/10.1109/INFCOMW.2013.6970748.
[64] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, A. Ramirez, Tibidabo: Making the case for an ARM-based HPC system, Future Gener. Comput. Syst. 36 (2014) 322–334. http://dx.doi.org/10.1016/J.FUTURE.2013.07.013.
[65] J. Wanza Weloli, S. Bilavarn, M. De Vries, S. Derradji, C. Belleudy, Efficiency modeling and exploration of 64-bit ARM compute nodes for exascale, Microprocess. Microsyst. 53 (2017) 68–80. http://dx.doi.org/10.1016/J.MICPRO.2017.06.019.
[66] T. Sharp, F. Galluppi, A. Rast, S. Furber, Power-efficient simulation of detailed cortical microcircuits on SpiNNaker, J. Neurosci. Methods 210 (1) (2012) 110–118. http://dx.doi.org/10.1016/J.JNEUMETH.2012.03.001.
[67] L. Chen, C. Englund, Every second counts: integrating edge computing and service oriented architecture for automatic emergency management, J. Adv. Transp. 2018 (2018). http://dx.doi.org/10.1155/2018/7592926.
[68] P. Anderson, System Configuration, in: Short Topics in System Administration, vol. 14, SAGE, 2006.
[69] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: 29th Annual IEEE International Conference on Local Computer Networks, IEEE (Computer Society), Tampa, FL, USA, 2004, pp. 455–462. http://dx.doi.org/10.1109/LCN.2004.38.
[70] I. Berry, N.R. Kedia, S. Huang, B. Banisadr, Windows 10 IoT Core, 2017. https://developer.microsoft.com/en-us/windows/iot. (Accessed 13 June 2018).
[71] Raspbian, 2017. https://www.raspbian.org. (Accessed 13 June 2018).
[72] Ubuntu MATE for Raspberry Pi, 2017. https://ubuntu-mate.org/raspberry-pi/. (Accessed 13 June 2018).
[73] Alpine linux, 2017. https://alpinelinux.org/. (Accessed 13 June 2018).
[74] Buildroot, 2017. https://buildroot.org. (Accessed 13 June 2018).
[75] OpenEmbedded project, 2017. http://openembedded.org. (Accessed 13 June 2018).
[76] Yocto project, 2017. https://yoctoproject.org. (Accessed 13 June 2018).
[77] A. Belloni, T. Petazzoni, Buildroot vs OpenEmbedded/Yocto project: A four hands discussion, 2016. https://events.linuxfoundation.org/sites/events/files/slides/belloni-petazzoni-buildroot-oe_0.pdf. (Accessed 13 June 2018).
[78] libOSTree, 2017. https://ostree.readthedocs.io/en/latest/. (Accessed 13 June 2018).
[79] Mender, 2017. http://mender.io. (Accessed 13 June 2018).
[80] meta-updater, 2017. https://github.com/advancedtelematic/meta-updater. (Accessed 13 June 2018).
[81] LinuxKit, 2017. https://github.com/linuxkit/linuxkit. (Accessed 13 June 2018).
[82] Raspberry Pi Foundation, Network Booting, 2017. https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net_tutorial.md. (Accessed 13 June 2018).
[83] P. Stevens, Sneak Preview From Mythic Beasts Labs Raspberry Pi Netboot, 2017. https://blog.mythic-beasts.com/2016/08/05/sneak-preview-from-mythic-labs-raspberry-pi-netboot. (Accessed 13 June 2018).
[84] iPXE –Open Source Boot Firmware, 2017. http://ipxe.org. (Accessed 13 June 2018).
[85] J.J. Cusick, W. Miller, N. Laurita, T. Pitt, Design, construction, and use of a single board computer beowulf cluster: Application of the small-footprint, low-cost, InSignal 5420 Octa Board, CoRR abs/1501.00039 (2014) arXiv:1501.00039.
[86] Hyper-V, 2016. https://technet.microsoft.com/en-us/library/mt169373(v=ws.11).aspx. (Accessed 13 June 2018).
[87] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles - SOSP '03, vol. 37, ACM Press, New York, New York, USA, 2003, p. 164. http://dx.doi.org/10.1145/945445.945462.
[88] M. Rosenblum, VMware's Virtual Platform A Virtual Machine Monitor for Commodity PCs, in: Hot Chips: A Symposium on High Performance Chips, Stanford, 1999.
[89] Docker Inc., 2017. https://docker.com. (Accessed 13 June 2018).
[90] G.M. Kurtzer, Singularity 2.1.2 - Linux application and environment containers for science, 2016. https://dx.doi.org/10.5281/zenodo.60736.
[91] R. Morabito, Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation, IEEE Access 5 (2017) 8835–8850. http://dx.doi.org/10.1109/ACCESS.2017.2704444.
[92] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, J. Crowcroft, Unikernels: library operating systems for the cloud, in: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, in: ASPLOS '13, ACM, New York, NY, USA, 2013, pp. 461–472. http://dx.doi.org/10.1145/2451116.2451167.
[93] Apache Mesos, 2017. http://mesos.apache.org/. (Accessed 13 June 2018).
[94] Swarm Mode Overview, 2017. https://docs.docker.com/engine/swarm/. (Accessed 13 June 2018).
[95] Kubernetes, 2017. https://kubernetes.io/. (Accessed 13 June 2018).
[96] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, Borg, Omega, and Kubernetes, Commun. ACM 59 (5) (2016) 50–57. http://dx.doi.org/10.1145/2890784.
[97] N. Schot, Feasibility of raspberry pi 2 based micro data centers in big data applications, in: 23rd Twente Student Conference on IT, vol. 23, University of Twente, 2015.
[98] W. Hajji, F.P. Tso, Understanding the performance of low power raspberry pi cloud for big data, Electronics 5 (2) (2016) 29. http://dx.doi.org/10.3390/electronics5020029.
[99] X. Fan, S. Chen, S. Qi, X. Luo, J. Zeng, H. Huang, C. Xie, An ARM-based hadoop performance evaluation platform: design and implementation, in: Collaborative Computing: Networking, Applications, and Worksharing: 11th International Conference, Springer, 2016, pp. 82–94. http://dx.doi.org/10.1007/978-3-319-28910-6_8.
[100] M. Marschall, Chef Infrastructure Automation Cookbook, Packt Publishing, 2013, p. 276.
[101] V. Hendrix, D. Benjamin, Y. Yao, Scientific cluster deployment and recovery - using puppet to simplify cluster management, J. Phys. Conf. Ser. 396 (4) (2012). http://dx.doi.org/10.1088/1742-6596/396/4/042027.
[102] M. Mohaan, R. Raithatha, Learning Ansible : Use Ansible to Configure Your Systems, Deploy Software, and Orchestrate Advanced IT Tasks, Packt Publishing, 2014, p. 308.
[103] C. Sebenik, T. Hatch, Salt Essentials, O'Reilly Media Inc, 2015, p. 178.
[104] Metal as a Service, 2017. https://maas.io. (Accessed 13 June 2018).
[105] B. Philips, Recoverable System Updates, 2017. https://coreos.com/blog/recoverable-system-upgrades.html. (Accessed 13 June 2018).
[106] Android –OTA Updates, 2017. https://source.android.com/devices/tech/ota. (Accessed 13 June 2018).
[107] G. Pollock, D. Thompson, J. Sventek, P. Goldsack, The Asymptotic Configuration of Application Components in a Distributed System, Citeseer, 1998.
[108] D. Halfin, N. Brower, B. Lich, L. Poggemeyer, rkopoku, Deploy updates using Windows Update for Business, 2017. https://docs.microsoft.com/en-gb/windows/deployment/update/waas-manage-updates-wufb. (Accessed 13 June 2018).
[109] H. Herry, E. Band, C. Perkins, J. Singer, Peer to peer secure update for heterogeneous edge devices, in: Proceedings of the IEEE/IFIP International Workshop on Decentralized Orchestration and Management of Distributed Heterogeneous Things (DOMINOS), 2018. https://csperkins.org/publications/2018/04/herry2018p2p-secure.pdf.
[110] TensorFlow Lite, 2017. https://www.tensorflow.org/mobile/tflite. (Accessed 13 June 2018).

[111] P.-H. Tsai, H.-J. Hong, A.-C. Cheng, C.-H. Hsu, Distributed analytics in fog computing platforms using TensorFlow and Kubernetes, in: Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific, IEEE, 2017, pp. 145–150. http://dx.doi.org/10.1109/APNOMS.2017.8094194.
[112] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, Session Traversal Utilities for NAT (STUN), 2008. http://dx.doi.org/10.17487/RFC5389.
[113] J. Rosenberg, ICE: A Protocol for NAT Traversal for Offer/Answer Protocols, 2010. http://dx.doi.org/10.17487/RFC5245.
[114] C. Dale, apt-p2p - apt helper for peer-to-peer downloads of Debian packages, 2017. http://manpages.ubuntu.com/manpages/zesty/man8/apt-p2p.8.html. (Accessed 13 June 2018).
[115] HashiCorp, Serf: decentralized cluster membership, failure detection, and orchestration, 2017. https://www.serf.io/. (Accessed 13 June 2018).



Steven J. Johnston is a Senior Research Fellow in the Faculty of Engineering & Physical Sciences at the University of Southampton. Steven completed a Ph.D. with the Computational Engineering and Design Group and he also received an M.Eng. degree in Software Engineering from the School of Electronics and Computer Science. Steven has participated in 40+ outreach and public engagement events as an outreach program manager for Microsoft Research. He currently operates the LoRaWAN wireless network for Southampton and his current research includes the large scale deployment of environmental sensors. He is a member of the IET.



Philip J. Basford is a Senior Research Fellow in Distributed Computing in the Faculty of Engineering & Physical Sciences at the University of Southampton. Prior to this he was an Enterprise Fellow and Research Fellow in Electronics and Computer Science at the same institution. He has previously worked in the areas of commercializing research and environmental sensor networks. Dr. Basford received his M.Eng. (2008) and Ph.D. (2015) from the University of Southampton, and is a member of the IET.



Colin Perkins is a Senior Lecturer (Associate Professor) in the School of Computing Science at the University of Glasgow. He works on networked multimedia transport protocols, network measurement, routing, and edge computing, and has published more than 60 papers in these areas. He is also a long time participant in the IETF, where he co-chairs the RTP Media Congestion Control working group. Dr. Perkins holds a D.Phil. in Electronics from the University of York, and is a senior member of the IEEE, and a member of the IET and ACM.



Herry Herry is a Research Associate in the School of Computing Science at the University of Glasgow. Prior to this he was a Research Scientist in Hewlett Packard Labs. He works on cloud computing, system configuration management, distributed system, and edge computing. Dr. Herry received his B.Sc. (2002) and M.Sc. (2004) from Universitas Indonesia, and Ph.D. (2014) from the University of Edinburgh. He is a member of the IEEE.



Fung Po Tso is a Lecturer (Assistant Professor) in the Department of Computer Science at Loughborough University. He was Lecturer in the Department of Computer Science at Liverpool John Moores University during 2014–2017, and was SICSA Next Generation Internet Fellow based at the School of Computing Science, University of Glasgow from 2011–2014. He received B.Eng., M.Phil., and Ph.D. degrees from City University of Hong Kong in 2005, 2007, and 2011 respectively. He is currently researching in the areas of cloud computing, data center networking, network policy/functions chaining and large scale distributed systems with a focus on big data systems.



Dimitrios Pezaros is a Senior Lecturer (Associate Professor) and director of the Networked Systems Research Laboratory (netlab) at the School of Computing Science, University of Glasgow. He has received funding in excess of £2.5m for his research, and has published widely in the areas of computer communications, network and service management, and resilience of future networked infrastructures. Dr. Pezaros holds B.Sc. (2000) and Ph.D. (2005) degrees in Computer Science from Lancaster University, UK, and has been a doctoral fellow of Agilent Technologies between 2000 and 2004. He is a Chartered Engineer, and a senior member of the IEEE and the ACM.



Robert Mullins is a Senior Lecturer in the Computer Laboratory at the University of Cambridge. He was a founder of the Raspberry Pi Foundation. His current research interests include computer architecture, open-source hardware and accelerators for machine learning. He is a founder and director of the lowRISC project. Dr. Mullins received his B.Eng., M.Sc. and Ph.D. degrees from the University of Edinburgh.



Eiko Yoneki is a Research Fellow in the Systems Research Group of the University of Cambridge Computer Laboratory and a Turing Fellow at the Alan Turing Institute. She received her Ph.D. in Computer Science from the University of Cambridge. Prior to academia, she worked for IBM US, where she received the highest Technical Award. Her research interests span distributed systems, networking and databases, including complex network analysis and parallel computing for large-scale data processing. Her current research focus is auto-tuning to deal with complex parameter spaces using machine-learning approaches.



Simon J. Cox is Professor of Computational Methods and Chief Information Officer at the University of Southampton. He has a doctorate in Electronics and Computer Science, first class degrees in Maths and Physics and has won over £30m in research & enterprise funding, and industrial sponsorship. He has published over 250 papers. He has co-founded two spin-out companies and, as Associate Dean for Enterprise, has most recently been responsible for a team of 100 staff with a £11m annual turnover providing industrial engineering consultancy, large-scale experimental facilities and healthcare services.



Jeremy Singer is a Senior Lecturer (Associate Professor) in the School of Computing Science at the University of Glasgow. He works on programming languages and runtime systems, with a particular focus on many core platforms. He has published more than 30 papers in these areas. Dr. Singer has BA, MA and Ph.D. degrees from the University of Cambridge. He is a member of the ACM.