2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops

The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures

Fung Po Tso, David R. White, Simon Jouet, Jeremy Singer, Dimitrios P. Pezaros
School of Computing Science, University of Glasgow, G12 8QQ, UK
Email: {posco.tso, david.r.white, jeremy.singer, dimitrios.pezaros}@glasgow.ac.uk, s.jouet.1@research.gla.ac.uk

***Abstract*—Data Centers (DC) used to support Cloud services often consist of tens of thousands of networked machines under a single roof. The significant capital outlay required to replicate such infrastructures constitutes a major obstacle to practical implementation and evaluation of research in this domain. Currently, most research into Cloud computing relies on either limited software simulation, or the use of a testbed environments with a handful of machines. The recent introduction of the Raspberry Pi, a low-cost, low-power single-board computer, has made the construction of a miniature Cloud DCs more affordable.**

In this paper, we present the Glasgow Raspberry Pi Cloud (PiCloud), a scale model of a DC composed of clusters of Raspberry Pi devices. The PiCloud emulates every layer of a Cloud stack, ranging from resource virtualisation to network behaviour, providing a full-featured Cloud Computing research and educational environment.

***Index Terms*—Raspberry Pi Cloud; Data Centre; ARM-based Data Centre; Cloud Emulator;**

I. Introduction

Cloud Computing is emerging as a new processing paradigm based on outsourcing infrastructure on a "pay-as-you-go" basis, accommodating services ranging from private data processing to public website hosting. Despite much attention from the research community, research and development of Cloud Computing systems, applications, services and resource management are still in their infancy.

There are a number of important issues that need detailed investigation throughout the Cloud "stack". For example, topics of interest to Cloud providers include: economic strategies for provisioning virtualised resources to incoming user requests, application scheduling, and resource discovery. The extent to which most researchers can adequately address such problems is limited by the inaccessibility of Data Centre (DC) infrastructures for cost-effective large-scale experimentation. It is therefore important that suitable tools for exploring and evaluating ideas are made available to researchers.

A typical Cloud DC usually contains tens of thousands of servers [1], making it prohibitively expensive for an educational or research institution to construct one. Even a practical testbed consisting of a reasonable number of servers (say, 40 machines) can still be out of the reach of most researchers when one needs to consider space, power and cooling infrastructure.

Some researchers have instead focused on developing Cloud Computing simulators. While simulation has in the past been used to successfully model some of the underlying state of a target system (such as within network simulation), it fails to capture essential Cloud Computing properties in a number of ways.

Traffic patterns in operational Cloud DC networks constantly change over time and are generally unpredictable [2], [3] in the long term. The realism of simulated traffic patterns is questionable, since traffic dynamism is difficult to model. Obtaining realistic DC traffic pattern statistics is of paramount importance, as it can have a crucial impact on server and network utilisation, which ultimately drives Cloud service management.

Often, we also find that simulation does not model cross-layer (e.g., application, network, virtualisation) correlations and interaction. For example, iCanCloud [4] is a simulator of Cloud infrastructures aimed at simulating instance types provided by Amazon without considering the underlying network behaviour. A more sophisticated and extensible simulator, CloudSim [5], supports the modelling of essential DC resources such as virtualisation and network layers. However, it does not model popular Cloud applications, such as data mining or web services. Furthermore, its constrained development environment usually requires extensively tweaking existing models, and the resulting software packages cannot be readily ported to an operational environment.

With Cloud simulation tools being still at early development and physical x86 Cloud testbeds being prohibitively expensive, we are exploring a novel alternative: the construction of a cutting-edge "scale model" of a Cloud system to create a high fidelity and affordable testbed. To the best of our knowledge, we have built the first cost-effective scale model of a Cloud DC – the Glasgow Raspberry Pi Cloud (PiCloud) – composed of 56 Raspberry Pi devices. Following a typical densely interconnected network topology, the PiCloud can accurately replicate a Cloud DC's functionalities with scaled components. As a development environment, it permits reproduction of actual traffic patterns with realistic Cloud applications. Applications and technologies developed on top of the PiCloud can be readily adapted to real Cloud environments.

We believe that scale modelling of Cloud computing services with ARM-based devices is an important line of research that has yet to be exploited by the research community. We trust CCRM will be an ideal forum to elaborate on and promote this approach.

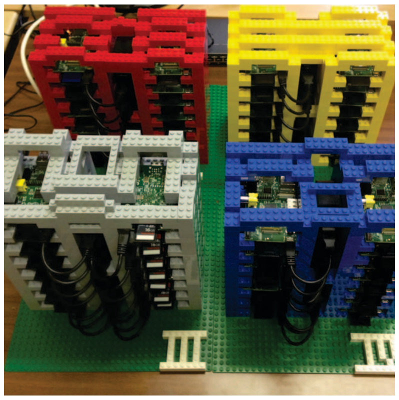


Fig. 1: Four PiCloud racks

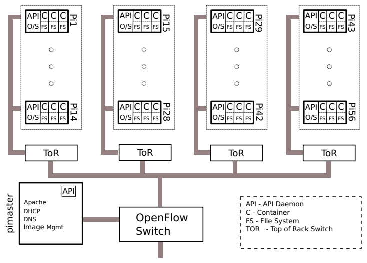


Fig. 2: System architecture

II. System Design

In this section we discuss design and implementation of the PiCloud, highlighting the rationale as well as the operational and design details of the individual components.

A. System Architecture

We have constructed a prototype scale model of a Cloud Data Centre (DC) cluster, called the Glasgow Raspberry Pi Cloud (abbreviated to PiCloud). It contains 56 Model B version Raspberry Pi devices, housed in racks constructed using Lego bricks, as shown in Fig. 1. The system's architecture is shown in Fig. 2. The 56 Raspberry Pi's are divided into 4 racks with 14 Raspberry Pis each.

In order to reflect a typical DC network architecture, the Raspberry Pi devices are interconnected through a canonical multi-root tree topology. Machines in the same rack are connected to the same Top of Rack (ToR) switch, which in turn connect to the rest of the topology through an OpenFlow-enabled aggregation switch. The benefit of using OpenFlow is to make the topology fully programmable and compatible with the leading-edge Software Define Networking (SDN) research. SDN is a fairly recent concept of logically centralising the network's control plane so that network-wide management can be programmed in software and subsequently enforced through the centrally-controlled installation of rules on the switches along the path. All Raspberry Pi devices are eventually connected to the Internet through the School's university gateway, which functions as a core or border router. The accessibility and design of the prototype means the PiCloud clusters can easily be re-cabled to form a fat-tree topology.

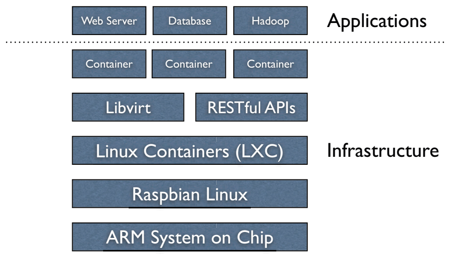


Fig. 3: PiCloud software stack

As the key building blocks of the PiCloud, each of these microcomputers runs Linux from a Sandisk 16GB SD card storage, supporting up to 3 co-located concurrent virtualised hosts with isolated filesystems, which are realised through Linux Containers and the supporting LXC suite. There is an API daemon on each Pi providing a RESTful management interface for facilitating virtual host management and interacting with a head node (the *pimaster*). A system administrator can implement customised IP and naming policies through DHCP and DNS services running on the *pimaster*, which also hosts image management tools providing image upgrading, patching, and spawning.

While our system architecture replicates the overall structure of a Cloud DC, we believe that the emulation at an individual server level is equally important. The software stack for an individual Pi is shown in Fig 3. Thanks to the effort of the Raspberry Pi community, there is a distribution of Debian optimised for the Raspberry Pi hardware known as Raspbian [6]. Unlike other ARM-based operating systems which usually contain only a subset of basic programs and utilities, the Raspbian comes with over 35,000 pre-compiled software packages, built with support for the Pi's hardware floating point capabilities.

Immediately above the Raspbian operating system, Linux Containers are used to provide a type of virtualisation, playing a role similar to the hypervisor in x86 virtualisation technologies. Coupled with the LXC toolset, the libvirt [7] library and some other customized APIs are used to facilitate resource management through interaction with *pimaster*. On top of the management layer are the virtual hosts. Currently, we are able to comfortably support three containers concurrently on a Raspberry Pi. Within each of the containers, any user application can be executed as if it was running on normal servers.

B. System Virtualisation

Given that resource virtualisation is one of the key enabling technologies for Cloud computing, it is imperative that the PiCloud can suitably emulate certain levels of system virtualisation. Hardware virtualisation architectures such as Xen and VMWare are used to create virtual hosts that observe shared hardware as if they solely own them; these technologies are the most prevalent virtualisation tools in use today. However, there are obstacles to running the same software on the PiCloud: full virtualisation technologies such as Xen are memory-intensive when compared to the 256MB RAM capacity of the original Raspberry Pi devices. In addition, many CPUs provide architectural support for virtualisation in order to efficiently isolate the guest operating systems. This support is usually provided by x86 processors, and while there is an ongoing effort trying to enable Xen on the ARM platform [8], this project is still under development at the time of writing.

As an alternative, we use a lightweight operating system-level virtualisation method for running multiple isolated servers on a single control host – the Linux Container, which is supported by the Linux kernel's CGROUPS functionality. Linux containers do not provide a full virtual machine, but rather a virtual environment that has its own process and network space; they are informally referred to as an enhanced version of "chroot". Linux Containers are a simple yet efficient solution for resource-constrained systems, although they do not yet provide the level of isolation that full virtualisation achieves. For example, we can run three containers on a single Pi, each consuming 30MB RAM when idle. In terms of network virtualisation, we can easily provide native networking to each container by bridging or NATing the virtual hosts to the physical network, allowing them to communicate over the DC infrastructure.

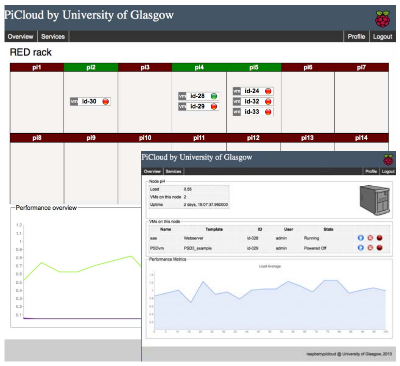
C. Management API

The LXC package provides a set of tools that can be used to control Linux Containers, using functionality provided by the Linux kernel. For example, the script `lxc-start` spawns a container. Above these basic tools, a management layer is needed to administer the PiCloud. We intend to adapt the libvirt framework, however this is currently not fully functional on the Pi platform (without significant workarounds). Instead, for the moment we rely upon a bespoke administration API supported by daemons on the *pimaster* and on individual Pi devices. An outward-facing webserver on *pimaster* provides a web-based control panel to users and administrators as shown in Fig 4. This website interacts with the local daemons, and controls workloads running on the Pi devices using RESTful interfaces. Typical use-case scenarios include remote monitoring of the CPU load on some/all Pi nodes, spawning new VM instances and specifying (soft) per-VM resource utilisation limits.

III. Research Directions

The PiCloud enables us to conduct research on any aspect of Cloud Computing and unified ICT with a level of fidelity and confidence that would otherwise be beyond our reach. In this section, we provide some examples of research directions for our work on the PiCloud so far.

Virtual Machine (VM) management is an important aspect of Cloud Computing, since it allows for consolidation to reduce power consumption, and oversubscription to improve cost efficiency. The way in which VMs are allocated is crucial;



Node pi4

Load	0.55
VMs on this node	2
Uptime	2 days, 18:07:37.960000

VMs on this node

Name	Template	ID	User	State
aas	Webserver	id-028	admin	Running
PSDvm	PSD3_example	id-029	admin	Powered Off

Fig. 4: PiCloud management web interface on pimaster node

we can experiment with new algorithms on the PiCloud, while directly observing the resulting behaviour on all layers of the Cloud architecture. Similarly, by operating an actual infrastructure, we can empirically evaluate improvements to file management and migration techniques. Whilst such tasks could also be carried out in a simulator, the PiCloud allows us to consider the impact of changes on all aspects of operation and on all layers of the infrastructure. By not isolating any single aspect and by not employing over-simplifying assumptions, we can observe the interactions as well as the effects of specific optimisations on all different layers, something that gives us confidence in the feasibility of any new approach by demonstrating that it works, at least in miniature.

In terms of network management, we can examine ways of reducing congestion through improved resource allocation, as well as looking at novel network architectures and technologies that require significant changes to the infrastructure. For example, we are researching IP-less routing in order to support more flexible and efficient migration. This is a good example of designing synergistic optimisation between different control loops of the Cloud (i.e., networking and virtualisation) that to date operate mostly in isolation.

As with a real DC, the PiCloud requires administration. Thus, we are forced to deal with the mundane yet crucial aspects of image management, API design, UI design etc., that may otherwise be forgotten. We are experimenting with new UIs for control of the Cloud, and the flexibility of owning our own testbed allows us to consider radical departures to the norm, such as a peer-to-peer Cloud management system.

One potential scenario in the future development of Cloud Computing is the removal of virtualisation. This could be in terms of switching to lighter mechanisms, such as the Linux Containers used in the PiCloud, or else removing virtualisation completely and renting out physical nodes rather than virtual ones. Such a 'fine-grained' approach to Cloud Computing would be well-supported by smaller, power-efficient processors - such as the ARMv6 ISA chips found on the Pi. The PiCloud gives us a platform to examine the potential of this idea at an early stage.

When considering power efficiency, it is difficult to provide accurate power consumption data through simulation or other 'closed' evaluation platforms. The PiCloud allows us to both isolate individual components to measure their power consumption characteristics, or instrument directly across the whole Cloud: we can run the PiCloud from a single trailing power socket board.

IV. Discussion

Why the Raspberry Pi? The Raspberry Pi project [9] has great potential to revolutionise Computer Science education and is currently garnering support from industry and the education sector, as well as widespread media coverage. The project and the resulting infrastructure enable Computing Science students not only to use-case ICT but also to get hands-on experience on building and programming full distributed systems. To further this goal, the Pi is available for as little as \$25. The machines are currently available in Model A & B, featuring small differences in RAM size and number of I/O ports. These machines, however, share many of the properties of Cloud-based servers, such as limited storage and peripheral capability - albeit at a smaller scale. The use of an educational computing platform to construct a Cloud environment offers a promising combination of learning and research.

At the same time, ARM-based architectures are currently attracting significant attention as a cost-effective alternative for large-scale compute environments. Even though current versions of the Raspberry Pi have not been designed specifically for this purpose, the potential for further decreasing the cost of individual boards makes the architecture particularly attractive. The Bill of Materials (BoM) for the Raspberry Pi is currently under Non-Disclosure Agreement (NDA), yet we can try to infer it based on previous products that use similar ARM cores and devices, such as the Allwinner A10 processor or the slightly older BeagleBone. Estimations place the processor as the most expensive component for around 10\$, followed by the cost of Printed Circuit Board (PCB), RAM, the Ethernet connector and the rest of the components. The Raspberry Pi is built around a BCM2835 processor that has been primarily designed for multimedia-capable embedded devices. The integrated peripherals of such processor include a dual core multimedia Co-processor, HD video encoding and decoding, image sensing pipeline, a Graphic Processing Unit (GPU) and a video display unit [10] It becomes evident that a significant cost for this System on Chip (SoC) can be cut for a Data Centre-tuned ARM chip, by removing most of the multimedia-related external peripherals while adding an Ethernet PHY. This can be motivated by companies processing large amounts of data that would require more servers that can handle simple tasks than raw single unit processing power. Due to the low power dissipation, the elimination of a north and south bridge, and the fact that most ARM SoCs leverage the use of their BGA pads to stack the SoC and RAM, more processors can be fitted in a small space. Calxeda is an example of a small start-up that expect to fit 288 quad core cortex A9 in a 4U (redstone) server rack. In 2012, 8.7 billion ARM processor-based chips reported as sold, representing a 32% share of the total available market [11].

What is the cost of the PiCloud? Table I illustrates the cost breakdown for constructing a 56-machine testbed with commodity x86 servers and Raspberry Pi devices, respectively. The cost of the PiCloud is several orders of magnitude smaller than that of commercial servers. If the goal is to carry out intensive computation, then clearly the PiCloud is no substitute for x86 boxes. However, if our goal is to replicate the architecture of a Cloud DC, then the PiCloud offers a very economic alternative. Furthermore, due to the lightweight nature of the Raspberry Pi both in size and in power consumption, it can be hosted without extensive physical footprint or expensive power and cooling management, which reportedly accounts for 33% of the total power consumption in Cloud DCs.

TABLE I: Cost breakdown of a testbed consisting 56 servers

	Server	Power	Needs Cooling?
Testbed	\$112,000 (@\$2,000)	10,080W/h (@180W/h)	Yes
PiCloud	\$1,960 (@\$35)	196W/h (@3.5W/h)	No

Isn't the Raspberry Pi just a "toy" device? It is true that, while hardware capacity can be linearly scaled down to a certain ratio (say 1:10), software cannot similarly be scaled directly. The current version of the Raspberry Pi cannot perform the compute-intensive operations per unit that one may find in large-scale, x86-based DCs. We are therefore currently limited to a subset of software (lightweight httpd servers, hadoop etc.) at the application layer that can be used to emulate current DC workloads. However, the hardware itself is evolving fast and prices drop quickly. Recently, the Raspberry Pi foundation doubled the RAM size on every Raspberry Pi while keeping the same price. We anticipate that this trend will continue, for example with regards to its processing power. Furthermore, the onboard GPU can also be exploited for general computation.

Cloud computing typically serves different needs from those of supercomputing. Efficient Cloud DC design advocates the use of COTS (Commodity Off-The-Shelf) servers and encourages distributed computation. Computation-intensive jobs are often divided into several small tasks which are in turn distributed over many servers. The different economic and technical models that Intel and ARM operate, favour ARM for the development of myriad-core compute Clouds. While Intel's revenue is based on semiconductor unit price, ARM covers an increasingly expanding market based on selling intellectual property as widely as possible. ARM's extremely large device base, lead to an overall license cost per device below \$0.1. At the same time, ARM processors are inherently less expensive due to the native RISC instruction set that allows for a simpler processor architecture. On the other hand, Intel's hybrid instruction set with performance optimisation have led to an architecture with an inherently fixed transistor cost for decoding the x86 instruction set into RISC [12] Complete with the Raspberry Pi's development environment, the PiCloud is thus not a "toy" approach but rather a cost-effective scale model of a Cloud evironment with excellent potential to evolve as a fully-fledged next generation custom-compute distributed platform.

The PiCloud for resource management in Cloud Computing? The virtualisation and cohabiting nature of Cloud Computing makes dynamic resource provisioning an important research challenge. In the layered design of service-oriented Cloud architectures, changes to resources provided in one layer (e.g., the VM pool) will often have ripple-effects on other layers (e.g., network traffic dynamics). For example, imperfect VM migration or a naive consolidation algorithm may improve server resource usage at the expense of frequent episodes of network congestion. Simulators focusing only on modelling system VM instances such as iCanCloud [4] will be unable to accurately reveal such side effects. This makes the PiCloud particularly relevant to CCRM, since it provides a real testbed with all the 'murky details' of practical DC management. With a physical testbed such as the PiCloud, complex interactions and conflicting requirements are not abstracted away, enabling (or, indeed, motivating) the design of a resource management schemes which synergistically manage resources across different layers. We also anticipate that results from testbed experiments can be fed back into the improvement of Cloud simulation and modelling processes.

Furthermore, the PiCloud is SDN-ready with OpenFlow switches forming the aggregation layer. SDN-based network logical centralisation abstracts complexity from hardware to controller software, allowing control logic to be dynamically defined and programmed in software. Such a global view of the network will enhance overall resource management for Cloud computing, with finer granularity management policies and more efficient policy reinforcement.

V. Related Work

Building a Cloud infrastructure can costs millions of dollars [1]. In order to support the research community, Yahoo, HP and Intel have together established the Open Cirrus, a global Cloud Computing testbed that supports a federation of DCs located in 10 organisations [13]. Building such experimental environments is expensive and still hard to access due to its globally shared nature.

Limited accessibility to actual Cloud testbeds has made researchers to resolve to simulated environments. Cloud Computing simulators include CloudSim [5], GreenCloud [14], iCanCloud [4] and MDCSim [15]. GreenCloud [14] is an extension of the NS2 network simulator for evaluation of energy-aware Cloud DCs. The main strength of GreenCloud is the detailed modelling of communication within a DC network. MDCSim [15] is a commercial discrete event simulator that models specific hardware characteristics of different DC components such as servers, communication links, and switches. On the contrary, iCanCloud [4] is a hypervisor simulator specifically aimed at simulating instance types provided by Amazon. Thus far, the only tool that can simulate an entire Cloud stack is CloudSim [5]. However, among these simulators, there is limited or no support for more realistic and complex applications composed of communicating tasks and workflows, enabling no or limited cross-layer interaction.

As a practical balance between a prohibitively expensive testbed and limited simulation, the PiCloud offers an inexpensive means of building a physical Cloud Computing testbed with a full software stack and complete virtualisation management tools.

VI. Conclusion and Future Work

In this paper, we have presented the design and implementation of the PiCloud, a scale model that emulates a Cloud DC, from its overall architecture to the software stack on each individual machine. It allows us to carry out practical research into Cloud computing without the limitations of simulation or the expenditure of full-size x86 testbeds.

We have built an initial 56-node PiCloud arranged in a DC Clos network topology. Nevertheless, there are still many issues to be addressed: Immediate effort will be focused on adapting the *libvirt* framework to facilitate more secure and generic virtual resource management. Subsequently, we will implement sophisticated live migration within the PiCloud, to enable the study of important Cloud resource management aspects in depth.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, January 2009.

[2] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *ACM SIGCOMM*, 2011.

[3] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," *ACM SIGCOMM*, 2009.

[4] A. Nuez, J. Vzquez-Poletti, A. Caminero, J. Carretero, and I. Llorente, "Design of a new cloud computing simulation platform," in *Computational Science and Its Applications - ICCSA 2011*, ser. Lecture Notes in Computer Science, 2011, vol. 6784, pp. 582–593.

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[6] "Raspbian." [Online]. Available: http://www.raspbian.org/

[7] "The virtualization api." [Online]. Available: http://libvirt.org/

[8] "Xen arm." [Online]. Available: http://xen.org/products/xen_arm.html

[9] "Raspberry pi." [Online]. Available: http://www.raspberrypi.org

[10] "Raspberry pis secret: sell out a little to sell a lot." [Online]. Available: http://www.wired.com/opinion/2012/09/raspberry-pi-insider-exclusive-sellout-to-sell-out

[11] "Arm annual report & accounts 2012," p. 2.

[12] "The battle between arm and intel gets real." [Online]. Available: http://spectrum.ieee.org/semiconductors/processors/the-battle-between-arm-and-intel-gets-real

[13] A. Avetisyan, R. Campbell, I. Gupta, M. Heath, S. Ko, G. Ganger, M. Kozuch, D. O"Hallaron, M. Kunze, T. Kwan, K. Lai, M. Lyons, D. Milojicic, H. Y. Lee, Y. C. Soh, N. K. Ming, J.-Y. Luke, and H. Namgoong, "Open cirrus: A global cloud computing testbed," *Computer*, vol. 43, no. 4, pp. 35–43, April.

[14] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec., pp. 1–5.

[15] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. Das, "Mdcsim: A multi-tier data center simulation, platform," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 31 2009-Sept. 4, pp. 1–9.