Computing (2021) 103:1859–1878
https://doi.org/10.1007/s00607-020-00900-y

REGULAR PAPER

The evolution of distributed computing systems: from fundamental to new frontiers

Dominic Lindsay[1] · **Sukhpal Singh Gill**[2] · **Daria Smirnova**[1] · **Peter Garraghan**[1]

Received: 6 February 2020 / Accepted: 28 December 2020 / Published online: 30 January 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH, AT part of Springer Nature 2021

Abstract

Distributed systems have been an active field of research for over 60 years, and has played a crucial role in computer science, enabling the invention of the Internet that underpins all facets of modern life. Through technological advancements and their changing role in society, distributed systems have undergone a perpetual evolution, with each change resulting in the formation of a new paradigm. Each new distributed system paradigm—of which modern prominence include cloud computing, Fog computing, and the Internet of Things (IoT)—allows for new forms of commercial and artistic value, yet also ushers in new research challenges that must be addressed in order to realize and enhance their operation. However, it is necessary to precisely identify what factors drive the formation and growth of a paradigm, and how unique are the research challenges within modern distributed systems in comparison to prior generations of systems. The objective of this work is to study and evaluate the key factors that have influenced and driven the evolution of distributed system paradigms, from early mainframes, inception of the global inter-network, and to present contemporary systems such as edge computing, Fog computing and IoT. Our analysis highlights assumptions that have driven distributed systems appear to be changing, including (1) an accelerated fragmentation of paradigms driven by commercial interests and physical limitations imposed by the end of Moore's law, (2) a transition away from generalized architectures and frameworks towards increasing specialization, and (3) each paradigm architecture results in some form of pivoting between centralization and decentralization coordination. Finally, we discuss present day and future challenges of distributed research pertaining to studying complex phenomena at scale and the role of distributed systems research in the context of climate change.

Keywords Distributed computing · Computing systems · Evolution · Green computing

✉ Sukhpal Singh Gill
s.s.gill@qmul.ac.uk

Extended author information available on the last page of the article

Mathematics subject classification 68M14 · 68U35 · 86A08 · 01-02

1 Introduction

Societal prosperity of the latter half of the twentyfirst century has been underpinned by the Internet, formed by large-scale computing infrastructure composed of distributed systems which have accelerated economic, social and scientific advancement [1]. The complexity and scale of such systems have been driven by increased societal demand and dependence on such computing infrastructure, which in turn has resulted in the formation of new distributed system paradigms. In fact, these paradigms have evolved in response to technological changes and usage, resulting in alterations to the operational characteristics and assumptions of the underlying computing infrastructure. For example, early mainframe systems provided centralised computing and storage interfaced by teletype terminals. Clustering and packet switching advancement in microprocessor technology and GUIs transferred computing from large mainframes operated remotely to home PCs [2, 3]. Standardisation of network protocols enabled global networks-of-networks to exchange messages for global applications [1]. Organisations developed frameworks and protocols capable of offloading computation to remote machine pools of computing resources such as processing, storage and memory [4, 5], eventually incorporating sensing and actuator objectives with embedded network capabilities [6]. Thus, distributed systems paradigms have evolved to distribute and facilitate service from centralised clusters, extending infrastructure beyond the boundaries of central networks forming paradigms such as IoT and Fog computing [7, 8].

For the past 60 years distributed system paradigms have conceptually evolved to meet challenges introduced by an ever-changing computing infrastructure and society [9]. From mainframes to clusters, clusters to Cloud, and Cloud to distributed and decentralised infrastructures encompassing the IoT to Edge Infrastructure [10]. Yet paradigms still retain the same underlying characteristics and elements that define their operation [11]. Each is defined by persistent research activities and are often driven by the development of new capabilities, such as security [12], hardware accelerators [13], edge computing [14] and power efficiency [15]. Whilst application framework have evolved to meet challenges presented by integrating with wider ecosystems, ranging from distributed clouds to highly specialised application specific infrastructures [16–18]. As such distributed paradigms require constantly evolving middleware's, communication protocols, and secure isolation mechanisms [19].

This work focuses on ascertaining the key characteristics and elements of distributed and networked systems, critically appraise the historical technologies and social behaviour that drove their paradigm formation, whilst identifying key trends across the paradigms including system architecture fragmentation, centralisation and decentralisation pivoting, and delays in paradigm conceptualisation to creation by tracing the impact of networked systems on society. From these findings we discuss how future distributed systems will support decentralisation of computation services through composition of decentralised computation platforms specialised to meet workload specific performance goals, forming exponentially larger systems capable

of holistic operational requirements including capability and energy availability. Finally we summarise how a dynamic centralised/decentralised distributed paradigms may form and will shape the direction of future computer science research as well as their potential impact within greater society.

The rest of the article is structured as follows: Sect. 2 presents the background of distributed systems. Section 3 the evolution of the distributed system paradigms. Section 4 analyses trends and observations across all paradigms. Section 5 discusses future challenges facing distributed systems, and Sect. 6 presents our conclusions.

2 Background

Distributed systems describe a class of computing system in which hardware and software components are connected by means of a network, and coordinate their actions via message passing in order to meet a shared objective [20, 21]. Whilst paradigms exhibit differing operational behaviour and leverage various technologies, these systems are defined by their underlying core characteristics and elements that facilitate their operation.

2.1 Characteristics

2.1.1 Transparent concurrency

Distributed Systems are inherently concurrent, with any participating resource accessible via any number of local or remote processes. The capacity and availability of such a system can be increased by adding resources that require mechanisms for accounting and identification. Such a system is vulnerable to volatile inter-actor behaviours and must be resilient to node failure as well as lost and delayed messages [22]. The management and access of objects, hardware or data in a distributed networked environment is also of particular importance due to potential for physical resource contention [3, 4, 23, 24].

2.1.2 Lack of shared clock

Systems maintain their own independent time, interpreted from a variety of sources, and as such Operating Systems (OSs) are susceptible from clock skew and drift. Furthermore, detecting when a message was sent or received is important for ensuring correct system behaviour. Therefore, events are tracked by means of conceptual Logical and Vector clocks; by sequencing messages, processes distributed across a network are able to ensure total event ordering [25–27].

2.1.3 Dependable and secure operation

Components of a distributed system are autonomous, and service requests are dependent on correct transaction of operation between sub-systems. Failure of any

subsystem may affect the result of service requests and may manifest in ways that are difficult to effectively mitigate. Fault tolerance and dependability are key characteristics towards ensuring the survivability of distributed systems and allow services to recover from faults and whilst maintaining correct service [22].

2.2 Elements

2.2.1 Physical system architecture

Physical system architecture identifies physical devices that exchange messages in a distributed system and what medium they communicate over. Early distributed systems such as mainframes were physically connected to clients. Later packet switching enabled long-haul multi-hop communication. Cellular networks incorporate mobile computing systems, whilst modern systems host services at specialised hardware between services providers and consumers. Initial designs of distributed systems aimed to provide service across local or campus wide networks of tens to hundreds of machines, and were focused on the development of operating systems and remote storage [1, 4]. Early efforts were designed to explore potential challenges and demonstrate their feasibility [8] and to enhance their functional and non-functional properties (performance, security, dependability, etc).

2.2.2 Entities

A logical perspective of a distributed system describes several process exchanging messages in order to achieve a common goal [28, 29]. Contemporary systems extend this definition by considering logical and aggregate entities, such as Objects and Components, used for abstracting resource and functionality [30]. Here systems are exposed as well-defined interfaces capable of describing natural decomposition of functional software requirements, and enabled exploring the loose-coupling between interchangeable components for domain specific problems found in distributed computing [31]. More recent systems leverage web services and micro-services, that consider their deployment to physical hardware as well as constraints including locality, utilization and stakeholders' policies [32]. Grid and Cloud computing enable distributed computing by abstracting processing, memory and disk space aggregation [33] whereas Fog and Edge computing emphasize integrating mobile and embedded devices [34, 35].

2.2.3 Communication models

Several communication models support distributed systems [36–38] including (1) *Inter-process Communication:* Enabling two different processes to communicate with each other by means of operating system primitives such as pipes, streams, and datagrams in a client—server architecture; (2) *Remote Invocation:* Mechanisms and concepts enabling a process in one address space to affect execution of operations, procedures and methods in another address space; and (3) *Indirect Communication:*

Mechanisms enabling message exchanges between one to many processes via an intermediary. In contrast with previous communication models, senders and receiving processes are decoupled, and responsible for facilitating message exchange is passed to the intermediary [39, 40].

2.2.4 Consensus and consistency

Distributed systems make decisions amongst groups of cooperating processes each possessing possibly inconsistent states. Consensus algorithms are a mechanism in which a majority subset of nodes or 'quorum' can fulfil a client request negotiate a truth and fulfil a client request. Replication and partitioning are common techniques used to improve system scalability, reliability and availability [22] when exposed to volatile environments. Consistency is a challenge to both replicated, partitioned storage and consensus algorithms [22, 25]. Consistency in distributed systems can be defined as strong consistency, where any update to a partition of a data set is immediately reflect in any subsequent accesses, or weak consistency in which updates may experience delay before they are propagated through the system and are reflected in subsequent access's.

3 The evolution of distributed systems

Distributed systems have continued to evolve in response to various scientific, technological and societal factors. This has given rise to new forms of computer systems, as well as adaptation of paradigms from Client-Server through to IoT and Fog Computing [38]. However, the core characteristics and model elements discussed in Sect. 2 have remained relatively constant, with the precipitating paradigm augmenting (or re-engineering) technology from prior paradigms. Table 1 provides a detailed a timeline of key distributed paradigm formation, technologies that enabled their realisation, and a description of their respective elements. The formation of distributed systems does not occur in a vacuum, and is influenced by factors spanning other computer science disciplines (e.g. HCI, security), societal exposure, education, and business strategy [36, 37].

Due to the sheer volume of potential influences, we have focused our discussions pertaining to major technological advances and impact upon distributed system elements.

3.1 The mainframe (1960–1967)

Mainframes machines of the early 1960's provided time sharing service to local clients that interacted with teletype terminals [41]. Such system conceptualised the client-server architecture, prevalent in present day distributed systems design [42]. The client process connects and requests server processes, enabling a single time-sharing system to multiplex resources amongst clients [43]. Mainframes remained

Table 1 Timeline of distributed system paradigms formation and key technological drivers

Year	Driver	Technology and paradigm	Model elements			
			Physical	Conceptual	Entities	Communication
1960–1970	Clustering and packet switching (1967–1977)	Inter-process Communication (IPC) Client-server Supercomputer ARPANET and early Internet	Mainframe and telnet clients Local networks interconnected over packet switching infrastructure primarily for research activity	Client terminal connections share mainframe resources Networks, provide specific services to private networks, accessible clients across geographic and organisational boundaries	Clients (teletype terminals) and servers Hosts (servers), switches, routers and mainframes	Datagram transport (ATM, X.25)
1970–1980	GUI (WIMP), x86 architecture Internet protocols (1974–1984)	ARPANET Unix Initial conception of TCP/IP and UDP protocols Distributed operating systems	Local networks interconnected over packet switching infrastructure Home teletype computers and home video games. Early GUI based home computer based systems. Increased memory	Private networks provide services across geographic and organisational boundaries Domains translated to IP addresses for identifying networked hosts DNS system created	Hosts (servers), switches, routers and mainframes Mainframes provide specialised co-processors enabling parallel request processing from clients at scale	IP addressable hosts are able to communicate by means of datagrams

Table 1 (continued)

Year	Driver	Technology and paradigm	Model elements			
			Physical	Conceptual	Entities	Communication
1980–1990	POSIX.1	Home Computer (Apple LISA, ZX Spectrum, etc)	Mainframe terminals replaced with 8086 microprocessor architectures	Hosts interconnected by IP addresses and switches; DNS provides address translation; networks remain centralised Networks ARPANET, NSFNET, DECNET made obsolete by WAN infrastructure via TCP/IP	Clusters of microprocessor machines displace monolithic mainframes	TCP/IP becomes standard internet protocol of internet
	Remote Procedure Call (RPC)	Standardized TCP/IP and Initial internet	BBS boards begin to appear hosted and & operated by consumers			
	HTTP and HTML. (1985-1990)	Generalised OS (drivers)	Move from centralised mainframes to decentralised computers outside of research			
1990–2000	Middleware Peer to peer protocols	HTTP (TBL) HTML WWW P2P computing Mobile Computing	WWW leads to form geographic internet, services now provided by home servers and clustered machines Home systems connected by dial up modems	DNS, WWW, and TCP/IP enable decentralised internet Peer to Peer architecture enables highly decentralised file sharing, parallel processing, and online gaming applications	Most services now provisioned via off-the-shelf-machines organised into clusters Servers provide resources described by Uniform Resource Locators	Remote objects and procedures, enabled development of early middleware HTTP over TCP/IP popularise internet P2P protocols, group communication

Table 1 (continued)

Year	Driver	Technology and paradigm	Model elements			
			Physical	Conceptual	Entities	Communication
2000–2010	High speed broadband x86 Virtualization Hypervisors	Web Services Grid computing Community Computing Virtualized Commodity Clusters Cloud computing	Educational organizations form Grids for scientific goals VM para-virtualization application mobility Services and resource consolidation to datacenter Rise of smart phone adoption and mobile computing	Grids computing provides orchestration across organizational boundaries VMs enables resource isolation between applications on shared hardware Web services allow further service abstraction from physical hardware	Most services provisioned via off-the-shelf-machines organised into clusters described by Uniform Resource Locators Grids and Cloud provide resource pooling (CPU, memory, storage)	Cluster middleware REST, WSDL, XML, JSON, MQTT, XMPP (application layer group comm) Xen and KVM hypervisor
2010–2020	Software Defined Networks Containerization	IoT Edge Computing Fog Computing	Fog nodes Smart objects and edge infrastructure Edge datacenters	Specialization of computing tasks and hardware (GPU, NPU, smart phones, sensors) Remote resources (Storage, processing)	Containers become increasingly prominent Cloudlets	P4, Openflow Open SVN

prohibitively expensive and were the focus of supercomputing engineers that lead to the innovation of early disk-based storage and transistor memory [44].

3.2 Cluster networks (1967–1974)

The late 1960s and early 1970s saw the development of packet switching, and clusters of off-the-shelf computing components were identified as a cheaper alternative to more powerful yet more expensive supercomputer and mainframes [45]. New programming environments and resource abstractions were developed abstracting resource across local networks of machines [1, 4]. This time period also saw the creation of ARPANET and early networks that enabled global message exchange [5], allowing for services hostable on remote machines across geographic bounds decoupled from a fixed programming model. Cerf and Karn [5, 6] defined the TCP/IP protocol that facilitated datagram and stream orientated communication over a packet switched autonomous network of networks [46].

3.3 Internet and home PCs (1974–1985)

During this era, the Internet was created. Whilst early NCP-based ARPANET systems were characterised by powerful timesharing systems serving multiple clients over networks, new technologies such as TCP/IP had begun to transform the Internet into a network of several backbones, linking local networks to the wider Internet [5]. Thus, the number of hosts connected to the network began to grow rapidly, and centralised naming systems such as HOSTS.TXT could not scale sufficiently [2]. Domain Name Systems (DNSs) were formalised in 1985 and were able to transform hosts domain names to IP addresses; the Unix BIND system was the first public implementation of the DNS. Computers such as Xerox Star and Apple LISA utilizing early WIMP based GUIs demonstrated the feasibility of computing within the home, providing applications such as video games and web browsing to consumers.

3.4 World wide web (1985-1996)

During the late 1980s and early 1990s, the creation of HyperText Transport Protocol (HTTP) and HyperText Markup Language (HTML) [3] resulted in the first web browsers, website, and web-server[1]. Standardisation of TCP/IP provided infrastructure for interconnected network of networks known as the World Wide Web (WWW). This enables explosive growth of the number of hosts connected to the Internet, and was the public's first large societal exposure to Information Technology [3, 5]. Mechanisms such as Remote Procedure Calls (RPCs) were invented, allowing for the first time applications interfaced with procedure, functions and method across address spaces and networks [23].

[1] The first webpage—http://info.cern.ch/hypertext/WWW/TheProject.html

3.5 P2P, grids and web services (1994–2000)

Peer to Peer (P2P) applications such as Napster and Seti@Home demonstrated it was feasible for a global networks of decentralised cooperating processes to perform large-scale processing and storage. P2P enabled a division of workload amongst different peers/computing nodes whereby other peers could communicate with each other directly from the application layer [7][2] without the requirement of central coordinator. The creation of Web Services enables further abstraction of the system interface from implementation in the Web [11]. Rather than facilitate direct communication between clients and servers, Web Services mediated communication via brokerage services [47]. Scientific communities identified that creating federations for large pools of computing resources from commodity hardware could achieve capability comparable to that of large supercomputing systems [48]. Beowulf enables resource sharing amongst process by means of software libraries and middle-wares, conceptualising clustered infrastructure as a single system [49]. Grid computing enabled open access to computing resources and storage by means of open-protocols and middleware. This time period also saw the creation of effective x86 virtualization [50], which became a driving force for subsequent paradigms.

3.6 Cloud, Mobile and IoT (2000–2010)

A convergence of cluster technology, virtualization, and middleware resulted in the formation of the Cloud computing that enabled creating service models for provision application and computing resource as a service [51]. Driven primarily by large technology organization whom constructed large-scale datacenter facilities, computation and storage began a transition from the client-side to the provider side more similar to that of mainframes in the 1960s and 1970s [32, 52]. Mobile computing enabled access to remote resources from resource constrained devices with limited network access [50, 53] IoT also began to emerge from the mobile computing and sensor network communities providing common objects with sensing, actuating and networking capabilities, contributing towards building a globally connected network of ‘things’ [54].

3.7 Fog and edge computing (2010-present)

Whilst data produced by IoT and Mobile computing platforms continued to increase rapidly, collecting and processing the data in real-time was, and still remains an unsolved issue [55]. This resulted in forming Edge computing whereby computing infrastructure such as power efficient processors, and workload specific accelerators are placed between consumer devices and datacenter providers [53]. Fog computing provides mechanisms that allow for provisioning applications upon edge devices

[2] History of Distributed Systems—https://medium.com/microservices-learning/the-evolution-of-distributed-systems-fec4d35beffd

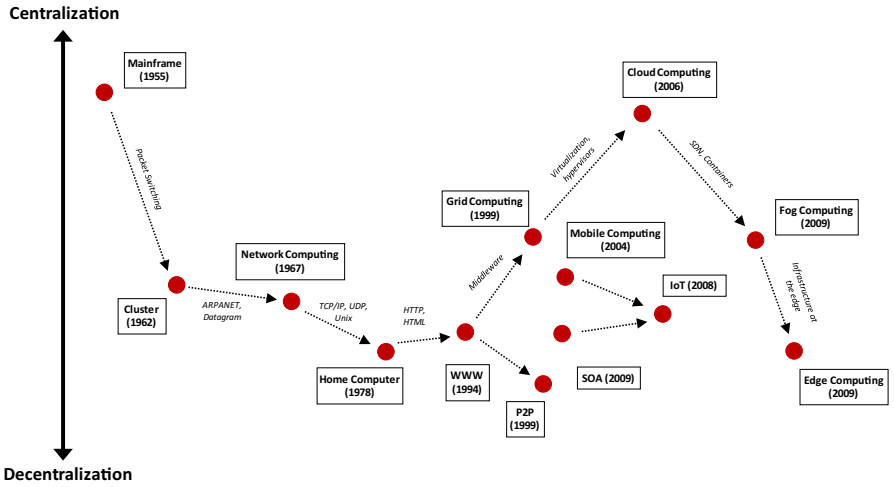

Fig. 1 Depiction of distributed system paradigm evolution

[56, 57], capable of coordinating and executing dynamic workflows across decentralised computing systems. The composition of Fog and Edge computing paradigms further extended the Cloud computing model away from centralised stakeholders to decentralized multi-stakeholder systems [56] capable of providing ultra-low service response times, increased aggregate bandwidths and geo-aware provisioning [14, 55]. Such a system may comprise of one-off federations or clusters, realised to meet single application workflows or act as intermediate service brokers, and provide common abstractions such as utility and elastic computing across heterogeneous, decentralised networks of specialised embedded devices, contrasting with centralised networks found in clouds [34].

4 Trends and observations

By appraising the evolution of the past six decades of distributed system paradigms shown in Table 1, it is apparent that a variety of technological advancements within computer science have driven the formation of new distributed paradigms. It is thus now possible to observe longer-term trends and characteristics of particular interest within distributed systems research.

4.1 Diversification of paradigms

There appears to be an increased diversification of distributed system paradigms as the research area has matured as shown in Fig. 1. This is predominantly driven by two factors: first, it is observable that the acceleration of paradigm formation was precipitated by the invention of the WWW in 1999. This is intuitive as this event enabled distributed systems to transition away from specialized research focused

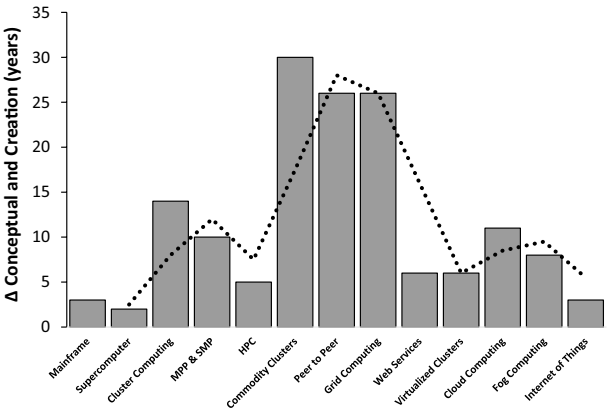

Fig. 2 Time gap between paradigm conception and creation

activities into greater society, with each sector requiring specific requirements from entertainment to commercial use. The second reason is that the maturity of fundamental technologies (TCP/IP, HTTP, Unix) created a platform that heavily emphasised abstraction to interconnect heterogeneous platforms in an effective manner, hence future paradigms were able to build upon these concepts. Figure 2 also demonstrates how distributed system paradigms transitioned from a potentially 'niche' research with a development singular track within the computer science community towards an area spanning a wide variety of paradigms coinciding with the time the Internet and WWW gained traction.

4.2 Architecture pivoting from centralization to decentralization

The creation of a new technology appears to drive the next distributed system paradigm, and respectively alter its respective degree of centralization as shown in Fig. 1. The creation of a new paradigm results in researchers revisiting fundamental mechanisms (schedulers, fault tolerance, monitoring) to ensure that they are capable of effectively operating within the new set of system assumptions. This is exemplified when considering responsibilities frequently carried out by scientists; a principle purpose of peer-review within the research community is ascertaining whether proposed approaches exhibit suitable differences from previous paradigms to determine their novelty (or whether it is a 'reinvention of the wheel'). This is apparent when considering a number of papers created that attempt to clearly distinguish between paradigms that leverage shared technologies [33]. We observe that the majority of paradigms predominantly are decentralized in nature, with the exception of Cloud computing which follows many similarities with the centralized mainframe in terms of the coordination of computational resources within a datacenter facility which users access via web APIs.

4.3 Time between system conception and creation

The delay between the description of a potential paradigm and actual successful implementation in recent years appears to be shorter in contrast to previous decades as shown in Figure 2. It is worth noting that ascertaining the precise publication fully credited in accurately describing the full realization of a paradigm due to a single individual or group is not necessarily feasible. Thus, we have attempted to seek papers which first define the appropriate terminology and paradigm description that were later adopted. As shown Fig 2, the formative years of distributed systems between 1960 and 1996 saw an average delay of 13 years and after the adoption of the WWW saw an average 8.8-years delay. It is observable that most paradigm are conceived and created sometime within 3–10 years, with the exception between 1960 and 1990 which is likely due to insufficient technologies when first envisioned, Later paradigms again appear to be relatively short in duration to create, and is likely a by-product of increased maturity of the research area, combined with its pervasiveness within society and growth of research activity within each respective paradigm (i.e. there are a sizable proportion of distributed researchers whom focus on a particular paradigm).

5 Future of large-scale computing infrastructure

5.1 Accelerated paradigm specialization

It is observable that specific distributed system paradigms have a particular affinity for tackling different objectives; whilst Cloud computing is capable of handling generalized application workload, paradigms such as edge computing and fog computing have been envisioned to be particularly effective for sensor actuation and increasingly important latency requirements. A growing number of microprocessors are being designed to accelerate specific tasks (such as graphics and machine learning using GPUs and NPUs, respectively). In tandem, the end of Moore's law indicates that by 2025 chip density will reach a scale where heat dissipation and quantum uncertainty make transistors unreliable [58]. When combining all of these factors together, it is apparent computing systems are in the process of undergoing massive diversification. This diversification is not solely limited to hardware but can also be observed in software.

For example, the last decade has seen resource management undergo a transition from centralized monolithic scheduling to decentralized model architecture [9, 16, 53]. Centralized schedulers maintain a global view of cluster state and are therefore able to make high quality placement decisions at the cost of latency [5, 6, 59, 60]. However, decentralized schedulers maintain only partial state about the cluster, and so they are able to make low latency decision at the cost placement quality [61]. As a result, we envision that further diversification and fragmentation of the distributed paradigm will continue to accelerate and affect all of its respective elements. For example, it is not hard to envision that the system that enables an infrastructure autonomous vehicle operation being substantially different to that of remote sensor

networks and smart phones; we are already seeing such diversification with making custom OSs and applications for these scenarios. In the case of cluster resource management there have been an increased research activity in hybrid schedulers, capable of multiplexing centralized and decentralized architectures [10, 19], and we expect that future distributed systems must be capable of architectural adaptivity in response to changes to operation.

5.2 Generalization against specialization

Related to paradigm specialization discussed in Section 5.1, the distributed systems research area appears to be at a particularly interesting cross-roads; ensure that system paradigms are designed to be generalizable to handle a wide variety of operational conditions and scenarios (at the cost of performance and efficiency), or alternatively focus on creating more specialized and bespoke distributed system more suitable to a particular task at the expense of generalization and portability. While the wide-spread adoption of the x86 architecture, middleware, and virtualization have reinforced that historically the community has championed generalizable and portability, continued diversification of paradigms and technological limitations have begun discussion whether the axis is pivoting in the other direction [62]. This is further reinforced by increasing customization of microprocessors, OSs, and power management techniques for particular use case scenarios. For example, the increased uptake of deep learning has resulted in further increase into research into GPU and NPUs inside and outside of the datacentre, as well as creation of cluster resource schedulers specifically for deep learning [63].

5.3 Complexity at scale and the role of academic research

An area of potential future research challenge moving forward is how to understand these future distributed systems at scale. For many years Computer scientists have leveraged well-structured system abstractions in order to reduce the complexity to understand component interactions and assumptions. However increasingly there have been difficulties in handling unseen emergent behaviour within massive-scale distributed systems [64] that require rethinking well-established assumptions for system mechanisms [65]. Moreover, with the rapid uptake of new technologies such as deep learning and reinforcement learning to conduct decision making of system operation [66], whilst introduction of temporal applications and mobile compute will likely lead to increased complexity of distributed system operation at scale. In relation to the academic research community, where there is a substantive reliance on simulation or small to medium-scale distributed systems, it will continue to become increasingly difficult to evaluate effectiveness of their approaches when exposed to emergent behaviour within systems at scale. Whilst production systems from industry can greatly support understanding of distributed systems at scale, it does not provide an avenue to conduct experiments within a controller environment to test hypothesis effectively.

5.4 The green agenda

Growing end-use demand for applications and subsequent data generation in the regions of Exabyte will usher in the first system at Exascale by 2020, and eventually Zettascale by 2035 [67]. Whilst an achievement in itself, it also brings a variety of associated challenges. One challenge which is particularly problematic is the enormous power requirements that will be necessary for operation. ICT presently consumes more than 10% of the global electricity annually [68]. The creation of ever larger systems through efficiency improvements is in fact detrimental due to the Rebound Effect [69] that causes even greater demand and consumption. At a time where climate change and a 1.5°C increase in global temperatures by 2100 due to Greenhouse Gas Emissions [70], we foresee energy and GHG emissions being increasingly important for future distributed system paradigms. This is not solely increasing energy-efficiency as we see today, but more fundamental concerns related to systems assuming operate constant stable power sources, integration with renewable energy sources, and alternative methods for reducing energy consumption but also computation itself. An area of particular interest is that of holistic coordination of energy management (asynchronous computing, voltage scaling, Wake-on-LAN, cooling, etc.) [15] towards studying and treating systems as living eco-systems, as opposed to individual components in isolation.

5.5 Shifting from centralised systems to decentralised edge

The evolution of centralised systems towards decentralised system transformed many industries and organisations which have resulted in significant contributions towards economic growth worldwide [71]. With the emergence of Big Data, centralised cloud systems have played an important role to process both structured and unstructured data in an efficient manner [72]. With the rapid adoption of IoT technology, these systems are able to process large amount data using various machine learning algorithms. It is difficult to process real-time jobs on centralised cloud systems due to increases in latency and response time, and incurs various complexities: New distributed applications (cryptocurrencies, the machine economy etc.) require computing models which are not compatible with existing centralised cloud systems [53]. As the adoption of edge computing is increasing, decentralised edge systems have been positioned to be particularly effective at processing user workloads immediately on powerful edge devices without the reliance upon large cloud datacenters [73], thus reducing round trip communication times at the cost of reduced computational performance. The evolution from centralised cloud systems to decentralised edge is growing among various industries while executing IoT based decentralised applications [74]. It is likely that given sufficient time and technological innovations, this pendulum will swing in the opposite direction, whereby computationally powerful decentralised systems will in turn form centralised architectures (and possibly an assortment of centralised systems coordinated via federation).

5.6 Distributed green computing

Rapid growth in large scale distributed application servicing paradigms ranging from Big Data and Machine Learning to the Internet of Things, are increasingly responsible for the world's energy consumption and as such a major contributor to environmental pollution [75]. One such example includes distributed Machine Learning systems [76]—comprising of clusters of GPUs dedicated to Deep Learning applications; require effective energy management aware scheduling policies [77]. As such new orchestration mechanism capable of capturing GPU, CPU, and memory energy characteristics [78] informing new scheduling algorithms prioritising energy consumption in contrast with traditional performance and fairness scheduling objectives [13, 15, 79]. Such scheduler should holistically consider energy consumption and account for out of band costs including impact of workload consolidation on cooling systems [15, 79]. Furthermore, exergy and energy source can be utilised to further inform datacentre operators about the carbon impact of their infrastructure. Whilst, hybrid energy grids utilizing green intermittent decentralised energy sources including solar and wind can provide clean energy whilst brown energy source can be utilized at peak time, minimized reliance of fossil fuels energy sources, and achieve new sustainable computing standards [80].

6 Conclusions

In this paper, we have discussed and evaluated the evolution of the distributed paradigm over the past six decades by focussing on the development and decentralised pivoting of networked computing systems. We have identified core elements of distributed systems by describing their physical infrastructure, logical entities and communication models. We examine how cross cutting factors such conceptual and physical models influence centralisation and decentralisation across various paradigms. We observe long term trends in distributed systems research, by identifying influential links between system paradigms, and technological breakthroughs. Of particular interest, we have observed that distributed system paradigms have undergone a long history of decentralisation up until the inception of the World Wide Web. In the following years, pervasive computing paradigms—such as the Internet of Things—brought about by advancements and specialisation of microprocessor architecture, operating systems designs, and networking infrastructure further diversified both infrastructure and conceptual systems. Furthermore, it is apparent that the diversification of distributed systems paradigms that begun at conception of the World Wide Web is likely to further accelerate due to increased emphasis on decentralisation and prioritization of specialized hardware and software for particular problems within domains such as machine learning and robotics. This is somewhat removed from the past few decades which has emphasized generality and portability of distributed system operation and as such will be the focus of research efforts over the coming years. Moreover, there are potentially difficult challenges on the horizon related to the upfront cost of operating large systems testbeds out of reach for most

academic laboratories, and the impact of climate change and how it shapes future system design.

Acknowledgements This work is supported by the UK Engineering and Physical Sciences Research Council (EP/P031617/1).

References

1. Armbrust M et al (2009) Above the clouds: A Berkeley view of cloud computing. EECS Department, University of California, Berkeley, no. January, pp 1–25, 2009
2. Lamport L (1978) Time, clocks, and the ordering of events in a distributed system. Commun ACM 21(7):558–565
3. Chow Y-C (1979) Models for dynamic load balancing in a heterogeneous multiple processor system. IEEE Trans Comput 10(5):354–361
4. Botta A, De Donato W, Persico V, Pescap A (2016) Integration of cloud computing and internet of things: a survey. Future Gen Comput Syst 56:684–700
5. Yu X, MI Fellow IEEE, Xue Y (2016) Smart grids: a cyber–physical systems perspective. Proc IEEE 104(5):1058–1070
6. Cisco Systems (2016) Fog computing and the internet of things: extend the cloud to where the things are, p 6. www.Cisco.com
7. Walker Bruce TG, Popek G, English R, Kline C (1983) The LOCUS distributed operating system. ACM SIGOPS Oper Syst Rev 17:49–70
8. Birrell AD, Levin R, Schroeder MD, Needham RM (1982) Grapevine: an exercise in distributed computing. Commun. ACM 25(4):260–274
9. Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz RH, Shenker S, Stoica I (2011) Mesos: a platform for fine-grained resource sharing in the data center. NSDI 11:22–22
10. . Delgado P, Dinu F, Kermarrec A-M, Zwaenepoel W (2015) Hawk: hybrid datacenter scheduling. In: USENIX ATC, 2015, pp 499–510
11. Peltz C (2003) Web services orchestration and choreography. IEEE Internet Comput 36(10):46–52
12. Arnautov S et al (2016) SCONE: Secure Linux containers with Intel SGX. In: Proceedings of 12th USENIX symposium on operating systems design and implementation, OSDI 2016, pp 689–703
13. I. R. Z. Michael Kaufmann, IBM Research Zurich, Karlsruhe Institute of Technology; Kornilios Kourtis (2017) The HCl scheduler: going all-in on heterogeneity. In: 9th {USENIX} workshop on hot topics in cloud computing (HotCloud 17), pp 1–7
14. Naha RK et al (2018) Fog computing: survey of trends, architectures, requirements, and research directions, vol 6, pp 47980–48009
15. Li X et al (2018) Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. IEEE Trans Parallel Distrib Syst 29:1317–1331
16. Vavilapallih V, Murthyh A, Douglasm C, Konarh M, Evansy R, Gravesy T, Lowey J, Sethh S, Sahah B, Curinom C, O'Malleyh O, Agarwali S, Shahh H, Radiah S, Reed B, Baldeschwieler E (2013) Apache Hadoop YARN. In: SoCC, 2013, pp 1–16
17. Burns B, Grant B, Oppenheimer D, Brewer E, Wilkes J (2016) Borg, omega, and kubernetes. Commun. ACM 59(5):50–57
18. Zaharia M, Das T, Li H, Hunter T, Shenker S, Stoica I (2013) Discretized streams: fault-tolerant streaming computation at scale. In: SOSP 2013—proceedings of the 24th ACM symposium on operating systems principles, no. 1, pp 423–438
19. Karanasos K, Rao S, Curino C, Douglas C, Chaliparambil K, Fumarola GM, Heddaya S, Ramakrishnan R, Sakalanaga S (2015) Mercury: hybrid centralized and distributed scheduling in large shared clusters. In: USENIX ATC, 2015, pp 485–497
20. Enslow PH (1978) What is a distributed data processing system? Computer 11(1):13–21
21. Gerard L (1977) Distributed systems—towards a formal approach. In: IFIP Congress, 1977
22. Algirdas Avižienis LC, Laprie J-C, Randell B (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secur Comput 1(1):11–33
23. Birrell AD, Nelson BJAY (1984) Implementing remote procedure calls. ACM Trans Comput Syst 2(1):39–59

24. Thain D, Tannenbaum T, Livny M (2005) Distributed computing in practice: the Condor experience. Concurr Comput Pract Exp 17(2–4):323–356
25. Lamport L, Shostak R, Pease M (1982) The Byzantine Generals Problem. ACM Trans Program Lang Syst 4(3):382–401
26. Figde C (1991) Logical time in distributed computing systems. Computer (Long Beach CA) 24:28–33
27. Friedemann M (1999) Virtual time and global states of distributed systems. SIAM J Comput 28(5):1829–1847
28. Sunderam VS, Geist GA, Dongarra J, Manchek R (1994) The PVM concurrent computing system: evolution, experiences, and trends. Parallel Comput 20(4):531–545
29. Gropp W (1998) An introduction to MPI parallel programming with the message passing interface, pp 1–48s
30. Gummadi PK, Saroiu S, Gribble SD (2002) A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems. ACM SIGCOMM Comput Commun Rev 32(1):82–82
31. Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D (2002) Seti@home an experiment in public-resource computing. Commun ACM 45(11):56–61
32. Fazio M, Celesti A, Ranjan R, Liu C, Chen L, Villari M (2016) (2016) Open issues in scheduling microservices in the cloud the types of devices that might. IEEE Cloud Comput 3(5):81–88
33. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: Grid computing environ work GCE 2008, pp 1–10
34. Mell P, Grance T (2011) The NIST definition of cloud computing recommendations of the National Institute of Standards and Technology. Nist Spec Publ 145:7
35. Singh S, Chana I (2016) A survey on resource scheduling in cloud computing: issues and challenges. J Grid Comput 14(2):217–264
36. Baheti R, Gill H (2011) Cyber-physical systems. Impact Control Technol 1:161–166
37. Karnouskos S (2011) Cyber-physical systems in the SmartGrid. In: 2011 9th international conference on industrial informatics, vol 1 VN-re, 2011
38. Evans D (2011) The internet of things—how the next evolution of the internet is changing everything. In: CISCO white paper, no. April, pp 1–11
39. Cerf VG, RE Icahn (1974) A protocol for packet network intercommunication. In: ACM SIGCOMM computer communication review 71 vol 35, number 2, April 2005, pp 71–82
40. Mockapetris Paul DK (1988) Development of the domain name system. In: SIGCOMM '88 Symposium, Communication, Architectures and Protocols, 1988
41. Flynn MJ (1966) Very high-speed computing systems. Proc IEEE 54(12):1901–1909
42. Singh S, Chana I, Singh M (2017) The journey of QoS-aware autonomic cloud computing. IT Prof 19(2):42–49
43. Casavant TL, Kuhl JG (1988) A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Trans Soft Eng 14(2):141–154
44. Compton K, Hauck S (2002) Reconfigurable computing : a survey of systems and software. 34(2):171–210
45. Amdahl GM (1967) Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS spring joint computer conference, pp 1–4
46. Lindsay D, Gill SS, Garraghan P (2019) PRISM: an experiment framework for straggler analytics in containerized clusters. In: Proceedings of the 5th international workshop on container technologies and container clouds, pp 13–18
47. Yu J, Buyya R A taxonomy of workflow management systems for grid computing, pp 1–31
48. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid. Hand Clin 17(4):525–532
49. Sterling T, Becker DJ, Savarase D, Dorband JE, Ranawake UA, Packer CV (1995) BEOWULF: a parallel workstation for scientific computation. In: Proceedings of the 24th international conference on parallel processing, pp 2–5
50. Gill SS, Ouyang X, Garraghan P (2020) Tails in the cloud: a survey and taxonomy of straggler management within large-scale cloud data centres. J Supercomput 50:10050–10089
51. Singh S, Chana I (2015) QoS-aware autonomic resource management in cloud computing: a systematic review. 48(3)
52. Leiner BM et al (2000s) Internet society (ISOC) all about the internet : a brief history of the internet internet society (ISOC) all about the internet : a brief history of the internet, pp 1–18
53. Gill SS et al (2019) Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: evolution, vision, trends and open challenges. Internet Things 8:100118

54. Whitmore A, Agarwal A, Da Xu L (2015) The internet of things—a survey of topics and trends. no. March 2014, pp 261–274
55. Gill SS, Garraghan P, Buyya R (2019) ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices. J Syst Softw 154:125–138
56. Brogi A, Forti S, Guerrero C, Lera I (2019) How to place your apps in the fog—state of the art and open challenges
57. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge Computing: Vision and Challenges. IEEE Internet Things J. 3(5):637–646
58. Waldrop M (2016) The chips are down for Moore's law. Nature 530:144
59. Verma A, Pedrosa L, Korupolu M, Oppenheimer D, Tune E, Wilkes J (2015) Large-scale cluster management at google with Borg. In: Proceedings of the tenth European conference on computer systems, EuroSys '15. ACM, New York, pp 18:1–18:17
60. Gog I, Schwarzkopf M, Gleave A, Watson RMN, Hand S (201) Firmament: fast, centralized cluster scheduling at scale. In: Proceedings of 12th USENIX symposium on operating systems design and implementation, 2016, pp 99–115
61. Ousterhout K, Wendell P, Zaharia M, Stoica I (2013) Sparrow: distributed, low latency scheduling. In: Proceedings of the 24th ACM symposium on operating systems principles, 2013, pp 69–84
62. Blair G (2018) Complex distributed systems: the need for fresh perspectives. In: IEEE ICDCS, pp 1410–1421
63. Xiao W et al (2018) Gandiva, introspective cluster scheduling for deep learning. In: OSDI, 2018
64. Gill SS, Shaghaghi A (2020) Security-aware autonomic allocation of cloud resources: a model, research trends, and future directions. J Organ End User Comput (JOEUC) 32(3):15–22
65. Garraghan P et al (2018) Emergent failures: rethinking cloud reliability at scale. IEEE Cloud Comput 5:12–21
66. Gao J (2014) Machine learning applications for data center optimization. Google White Paper, 2014
67. Liao X (2018) Moving from Exascale to Zettascale computing: challenges and techniques. Front Inf Technol Electron Eng 19:1236–1244
68. Van Heddeghem W, Lambert S, Lannoo B, Colle D, Pickavet M, Demeester P (2014) Trends in worldwide ICT electricity consumption from 2007 to 2012. Comput Commun 50:64–76
69. Gossart C (2014) Rebound effects and ICT: a review of the literature. In: ICT innovations for sustainability, pp 435–448
70. IPCC (2018) Global warming of 1.5 °C. Intergovernmental Panel on Climate Change, 2018
71. Chandra A, Weissman J, Heintz B (2013) Decentralized edge clouds. IEEE Internet Computing 17(5):70–73
72. Ferrer AJ, Manuel Marquès J, Jorba J (2019) Towards the decentralised cloud: survey on approaches and challenges for mobile, ad hoc, and edge computing. ACM Comput Surv 51(6):1–36
73. Khan MA, Algarni F, Quasim MT (2020) Decentralised internet of things. In: Decentralised internet of things. Springer, Cham, pp 3–20
74. Psaras I (2018) Decentralised edge-computing and IoT through distributed trust. In: Proceedings of the 16th annual international conference on mobile systems, applications, and services, pp 505–507
75. Alqahtani A, Solaiman E, Patel P, Dustdar S, Ranjan R (2019) Service level agreement specification for end-to-end IoT application ecosystems. Softw Pract Exp 49(12):1689–1711
76. Xiao W, Bhardwaj R, Ramjee R, Sivathanu M, Kwatra N, Han Z, Patel P, Peng X, Zhao H, Zhang Q, Yang F, Zhou L (2018) Gandiva: introspective cluster scheduling for deep learning. In: Proceedings of the 13th USENIX conference on operating systems design and implementation (OSDI'18). USENIX Association, USA, pp 595–610
77. Gill SS, Garraghan P, Stankovski V, Casale G, Thulasiram RK, Ghosh SK, Ramamohanarao K, Buyya R (2019) Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge. J Syst Softw 155:104–129
78. Yang R, Hu C, Sun X, Garraghan P, Wo T, Wen Z, Peng H, Xu J, Li C (2020) Performance-aware speculative resource oversubscription for large-scale clusters. IEEE Trans Parallel Distrib Syst 31(7):1499–1517
79. Ma K, Li X, Chen W, Zhang C, Wang X (2012) GreenGPU: a holistic approach to energy efficiency in GPU-CPU heterogeneous architectures. In: Proceedings of international conference on parallel processing, pp 48–57
80. Gill SS, Tuli S, Toosi AN, Cuadrado F, Garraghan P, Bahsoon R, Lutfiyya H et al (2020) ThermoSim: deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. J Syst Softw 164:110596

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Dominic Lindsay[1] · Sukhpal Singh Gill[2] · Daria Smirnova[1] · Peter Garraghan[1]

Dominic Lindsay
d.lindsay4@lancaster.ac.uk

Daria Smirnova
d.smirnova@lancaster.ac.uk

Peter Garraghan
p.garraghan@lancaster.ac.uk

[1] School of Computing and Communication, Lancaster University, Lancaster, UK

[2] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK