Addressing the Complexity of HPC in the Cloud: Emergence, Self-Organisation, Self-Management, and the Separation of Concerns

Theo Lynn

Abstract New use scenarios, workloads, and increased heterogeneity combined with rapid growth in adoption are increasing the management complexity of cloud computing at all levels. High performance computing (HPC) is a particular segment of the IT market that provides significant technical challenges for cloud service providers and exemplifies many of the challenges facing cloud service providers as they conceptualise the next generation of cloud architectures. This chapter introduces cloud computing, HPC, and the challenges of supporting HPC in the cloud. It discusses how heterogeneous computing and the concepts of self-organisation, self-management, and separation of concerns can be used to inform novel cloud architecture designs and support HPC in the cloud at hyperscale.

T. Lynn (✉)
Irish Centre for Cloud Computing (IC4), Dublin City University,
Dublin, Ireland
e-mail: theo.lynn@dcu.ie

© The Author(s) 2018

T. Lynn et al. (eds.), *Heterogeneity, High Performance Computing, Self-Organization and the Cloud*, Palgrave Studies in Digital Business & Enabling Technologies,
https://doi.org/10.1007/978-3-319-76038-4_1

Three illustrative application scenarios for HPC in the cloud—(i) oil and gas exploration, (ii) ray tracing, and (iii) genomics—are discussed.

Keywords Cloud computing • High performance computing • Emergent systems • Self-organising systems • Self-managing systems • Heterogeneous computing

1.1 INTRODUCTION

The objective of this book is to introduce readers to CloudLightning, an architectural innovation in cloud computing based on the concepts of self-organisation, self-management, and separation of concerns, showing how it can be used to support high performance computing (HPC) in the cloud at hyperscale. The remainder of this chapter provides a brief overview of cloud computing and HPC, and the challenges of using the cloud for HPC workloads. This book introduces some of the major design concepts informing the CloudLightning architectural design and discusses three challenging HPC applications—(i) oil and gas exploration, (ii) ray tracing, and (iii) genomics.

1.2 CLOUD COMPUTING

Since the 1960s, computer scientists have envisioned global networks delivering computing services as a utility (Garfinkel 1999; Licklider 1963). The translation of these overarching concepts materialised in the form of the Internet, its precursor ARPANET, and more recently cloud computing. The National Institute of Standards and Technology (NIST) defines cloud computing as:

...a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

(Mell and Grance 2011, p. 2)

NIST defines cloud computing as having five essential characteristics, three service models, and four deployment models as per Table 1.1.

Since the turn of the decade, the number and complexity of cloud providers offering one or more of the primary cloud service models—

Table 1.1 Cloud computing essential characteristics, service models, and deployment models (adapted from Mell and Grance 2011)

Essential characteristics	
On-demand self-service	Consumers can unilaterally provision computing capabilities as needed automatically without requiring human interaction with the cloud provider.
Broad network access	Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms and interfaces (e.g. devices).
Resource pooling	The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
Rapid elasticity	Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outwards and inwards to meet demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
Measured service	Cloud systems automatically control and optimise resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported, providing transparency to the service provider and the consumer.
Service models	
Software as a Service	The capability provided to a consumer to use a provider's applications running on a cloud infrastructure and accessible by client interface.
Platform as a Service	The capability provided to a consumer to deploy onto the cloud infrastructure consumer-created or acquired applications created using development technologies provided by the provider.
Infrastructure as a Service	The capability provided to a consumer to provision computing resources to deploy and run arbitrary software such as operating systems and applications.
Deployment models	
Private Cloud	The cloud infrastructure is provisioned for exclusive use by a single organisation comprising multiple consumers. Ownership, management, and operation of the infrastructure may be done by one or more of the organisations in the community, by a third party, or a combination of both, and it may exist on or off premise.
Community Cloud	The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organisations that have shared concerns. Ownership, management, and operation of the infrastructure may be done by one or more of the organisations in the community, by a third party, or a combination of both, and it may exist on or off premise.

(continued)

Table 1.1 (continued)

Public Cloud	The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organisation, or some combination of them. It exists on the premises of the cloud provider.
Hybrid Cloud	The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardised or proprietary technology that enables data and application portability.

Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service—as private, public, community, and hybrid clouds has increased. Cloud computing is now considered to be the dominant computing paradigm in enterprise Information Technology (IT) and the backbone of many software services used by the general public, including search, email, social media, messaging, and storage. Enterprises are attracted by the convergence of two major trends in IT—IT efficiencies and business agility, enabled by scalability, rapid deployment, and parallelisation (Kim 2009). Figure 1.1 summarises the strategic motivations for cloud adoption.

Despite its ubiquity, cloud computing is dominated by a small number of so-called hyperscale cloud providers, companies whose underlying cloud infrastructure and revenues from cloud services are at a different order of magnitude to all the others. These include companies who offer a wide range of cloud services such as Microsoft, Google, Amazon Web Services (AWS), IBM, Huawei and Salesforce.com, as well as companies whose core businesses leverage the power of cloud to manage the scale of their, typically online, operations such as Facebook, Baidu, Alibaba and eBay. Estimates suggest that such companies operate one to three million or more servers worldwide (Data Center Knowledge 2017; Clark 2014). Research by Cisco (2016) suggests that these hyperscale operators number as little as 24 companies operating approximately 259 data centres in 2016. By 2020, these companies will account for 47% of all installed data centre servers and 83% of the public cloud server installed base (86% of public cloud workloads) serving billions of users worldwide (Cisco 2016).

The data centres operated by hyperscale cloud service providers are sometimes referred to as Warehouse Scale Computers (WSCs) to differentiate them from other data centres. The data centre(s) hosting WSCs are typically not shared. They are operated by one organisation to run a small number

Fig. 1.1 IC4 cloud computing strategic alignment model

of high-use applications or services, and are optimised for those applications and services. They are characterised by hardware and system software platform homogeneity, a common systems management layer, a greater degree of proprietary software use, single organisation control, and a focus on cost efficiency (Barroso and Hölzle 2007). It is important also to note that for these hyperscale clouds, the clouds, per se, sit on top of the physical data centre infrastructure and are abstracted from end-user applications, end users, and software developers exploiting the cloud. Indeed, hyperscale clouds operate across multiple data centres typically organised by geographic region. This abstraction, combined with homogeneity, provides cloud service providers with cost efficiencies and deployment flexibility allowing cloud service providers to maintain, enhance, and expand the underlying cloud infrastructure without requiring changes to software (Crago and Walters 2015). Conventionally, cloud computing infrastructure performance is improved through a combination of scale-out and natural improvements in microprocessor capability, while service availability is assured through over-provisioning. As a result, hyperscale data centres are high-density facilities utilising tens of thousands of servers and often measure hundreds of thousands of square feet in size. For example, the Microsoft data centre in Des Moines, Iowa, is planned to occupy over 1.2 million square feet in size when it opens in 2019. While this high-density homogeneous scale-out strategy is effective, it results in significant energy costs. Servers may be underutilised relative to their peak load capability, with frequent idle times resulting in disproportionate energy consumption (Barroso and Hölzle 2007; Awada et al. 2014). Furthermore, the scale of data centre operations results in substantial cooling-related costs, with significant cost and energy impacts (Awada et al. 2014). Unsurprisingly, given their focus on cost effectiveness, power optimisation is a priority for WSC operators.

From a research perspective, WSCs introduce an additional layer of complexity over and above smaller-scale computing platforms due to the larger scale of the application domain (including an associated deeper and less homogeneous storage hierarchy), higher fault rates, and possibly higher performance variability (Barroso and Hölzle 2007). This complexity is further exacerbated by the dilution of homogeneity through technological evolution and an associated evolving set of use cases and workloads. More specifically, the emergence of new specialised hardware devices that can accelerate the completion of specific tasks and networking infrastructure that can support higher throughput and lower latency is enabling

support for workloads that traditionally would be considered HPC (Yeo and Lee 2011). The introduction of heterogeneity combined with new workloads, such as those classified as HPC, will further introduce greater system performance variability, including response times and, as a result, will impact the quality of service. As such, new approaches to provisioning are required. Despite these challenges, cloud service providers have sought to enter the HPC market catering largely for batch processing workloads that are perfectly or pleasingly parallelisable. Examples include AWS Batch, Microsoft Azure Batch, and Google Zync Render. Notwithstanding the entry of these major cloud players, cloud is one of the smallest segments in the HPC market and vice versa (Intersect360 Research 2014)

1.3 HIGH PERFORMANCE COMPUTING

HPC typically refers to computer systems that through a combination of processing capability and storage capacity rapidly solve difficult computational problems (Ezell and Atkinson 2016). Here, performance is governed by the (effective) processing speed of the individual processors and the time spent in inter-processor communications (Ray et al. 2004). As technology has evolved, processors have become faster, can be accelerated, and can be exploited by new techniques. Today, HPC systems use parallel processing achieved by deploying grids or clusters of servers and processors in a scale-out manner or by designing specialised systems with high numbers of cores, large amounts of total memory, and high-throughput network connectivity (Amazon Web Services 2015). The top tier of these specialised HPC systems are supercomputers whose cost can reach up to US\$100 million. Such supercomputers are measured in floating-point operations per second (FLOPS) rather than millions of instructions per second, the measurement of processing capacity in general-purpose computing. At the time of writing, the world's fastest supercomputer, the Chinese Sunway TaihuLight, has over 10 million cores and a LINPACK benchmark rating of 93 petaflops (Feldman 2016; Trader 2017) and a peak performance of 125 petaflops (National Supercomputing Centre, WuXi n.d.). It is estimated to have cost US\$273 million (Dongarra 2016).

Traditionally, HPC systems are typically of two types—Message passing (MP)-based systems and Non-uniform Memory Access (NUMA)-based systems. MP-based systems are connected using scalable, high-bandwidth, low-latency inter-node communications (interconnect) (Severance and

Dowd 2010). Instead of using the interconnect to pass messages, NUMA systems are large parallel processing systems that use the interconnect to implement a distributed shared memory that can be accessed from any processor using a load/store paradigm (Severance and Dowd 2010). In addition to HPC systems, HPC applications can be organised into three categories—tightly coupled, loosely coupled, and data intensive. The stereotypical HPC applications run on supercomputers are typically tightly coupled and written using the messaging passing interface (MPI) or shared memory programming models to support high levels of inter-node communication and high performance storage (Amazon Web Services 2015). Weather and climate simulations or modelling for oil and gas exploration are good examples of tightly coupled applications. Loosely coupled applications are designed to be fault tolerant and parallelisable across multiple nodes without significant dependencies on inter-node communication or high performance storage (Amazon Web Services 2015). Three-dimensional (3D) image rendering and Monte Carlo simulations for financial risk analysis are examples of loosely coupled applications. A third category of HPC application is data-intensive applications. These applications may seem similar to the loosely coupled category but are dependent on fast reliable access to large volumes of well-structured data (Amazon Web Services 2015). More complex 3D-animation rendering, genomics, and seismic processing are exemplar applications.

HPC plays an important role in society as it is a cornerstone of scientific and technical computing including biological sciences, weather and climate modelling, computer-aided engineering, and geosciences. By reducing the time to complete the calculations to solve a complex problem and by enabling the simulation of complex phenomenon, rather than relying on physical models or testbeds, HPC both reduces costs and accelerates innovation. Demand and interest in HPC remain high because problems of increasing complexity continue to be identified. Society values solving these problems, and the economics of simulation and modelling is believed to surpass other methods (Intersect360 Research 2014). As such, it is recognised as playing a pivotal role in both science discovery and national competitiveness (Ezell and Atkinson 2016). International Data Corporation (IDC), in a report commissioned for the European Commission, highlights the importance of HPC:

The use of high performance computing (HPC) has contributed significantly and increasingly to scientific progress, industrial competitiveness, national and

regional security, and the quality of human life. HPC-enabled simulation is widely recognized as the third branch of the scientific method, complementing traditional theory and experimentation. HPC is important for national and regional economies—and for global ICT collaborations in which Europe participates—because HPC, also called supercomputing, has been linked to accelerating innovation.

(IDC 2015, p. 20)

Despite the benefits of HPC, widespread use of HPC has been hampered by the significant upfront investment and indirect operational expenditure associated with running and maintaining HPC infrastructures. The larger supercomputer installations require an investment of up to US\$1 billion to operate and maintain. As discussed, performance is the overriding concern for HPC users. HPC machines consume a substantial amount of energy directly and indirectly to cool the processors. Unsurprisingly, heat density and energy efficiency remain a major issue and has a direct dependence on processor type. Increasingly, the HPC community is focusing beyond mere performance to performance per watt. This is particularly evident in the Green500 ranking of supercomputers.¹ cursory analysis of the most energy efficient supercomputers suggests that the use of new technologies such as Graphical Processing Units (GPUs) results in significant energy efficiencies (Feldman 2016). Other barriers to greater HPC use include recruitment and retention of suitably qualified HPC staff. HPC applications often require configuration and optimisation to run on specialised infrastructure; thus, staff are required not only to maintain the infrastructure but to optimise software for a specific domain area or use case.

1.4 HPC AND THE CLOUD

At first glance, one might be forgiven for thinking that HPC and cloud infrastructures are of a similar hue. Their infrastructure, particularly at Warehouse Scale, is distinct from the general enterprise, and both parallelisation and scalability are important architectural considerations. There are high degrees of homogeneity and tight control. However, the primary emphasis is very different in each case. The overriding focus in HPC is performance and typically optimising systems for a small number of large workloads. Tightly coupled applications, such as those in scientific computing, require parallelism and fast connections between processors to

meet performance requirements (Eijkhout et al. 2016). Performance is improved through vertical scaling. Where workloads are data intensive, data locality also becomes an issue, and therefore, HPC systems often require any given server in its system to be not only available and operative but connected via high-speed, high-throughput, and low-latency network interconnects. The advantages of virtualisation, and particularly space and time multiplexing, are of no particular interest to the HPC user (Mergen et al. 2006). Similarly, cost effectiveness is a much lower consideration.

In contrast, the primary focus in cloud computing is scalability and not performance. In general, systems are optimised to cater for multiple tenants and a large number of small workloads. In cloud computing, servers also must be available and operational, but due to virtualisation, the precise physical server that executes a request is not important, nor is the speed of the connections between processors provided the resource database remains coherent (Eijkhout et al. 2016). As mentioned earlier, unlike HPC, the cloud is designed to scale quickly for perfectly or pleasingly parallel problems. Cloud service providers, such as AWS, are increasingly referring to these types of workloads as High Throughput Computing (HTC) to distinguish them from traditional HPC on supercomputers. Tasks within these workloads can be parallelised easily, and as such, multiple machines and applications (or copies of applications) can be used to support a single task. Scalability is achieved through horizontal scaling—the ability to increase the number of machines or virtual machine instances. Cost effectiveness is a key consideration in cloud computing.

So, while there are technical similarities between the hyperscale cloud service providers operating their own Warehouse Scale Computing systems and HPC end users operating their own supercomputer systems, the commercial reality is the needs of HPC end users are not aligned with the traditional operating model of cloud service providers, particularly for tightly coupled use cases. Why? HPC end users, driven by performance, want access to heterogeneous resources including different accelerators, machine architectures, and network interconnects that may be unavailable from cloud service providers, obscured through virtualisation technologies, and/or impeded by multi-locality (Crago et al. 2011). The general cloud business model assumes minimal capacity for the end user to interfere in the physical infrastructure underlying its cloud and to exploit space and time multiplexing through virtualisation to achieve utilisation and efficiency gains. The challenge for service providers and HPC end users is one of balancing the need for (i) performance and scalability and

(ii) maximum performance and minimal interference. CloudLightning argues that this can be achieved through architectural innovation and the exploitation of heterogeneity, self-organisation, self-management, and separation of concerns.

1.5 HETEROGENEOUS COMPUTING

As discussed earlier, cloud computing data centres traditionally leverage homogeneous hardware and software platforms to support cost-effective high-density scale-out strategies. The advantages of this approach include uniformity in system development, programming practices, and overall system capability, resulting in cost benefits to the cloud service provider. In the case of cloud computing, homogeneity typically refers to a single type of commodity processor. However, there is a significant cost to this strategy in terms of energy efficiency. While transistors continued to shrink, it has not been possible to lower the processor core voltage levels to similar degrees. As a result, cloud service providers have significant energy costs associated not only with over-provisioning but with cooling systems. As such, limitations on power density, heat removal, and related considerations require a different architecture strategy for improved processor performance than adding identical, general-purpose cores (Esmailzadeh et al. 2011; Crago and Walters 2015).

Heterogeneous computing refers to architectures that allow the use of processors or cores, of different types, to work efficiently and cooperatively together using shared memory (Shan 2006; Rogers and Fellow 2013). Unlike traditional cloud infrastructure built on the same processor architecture, heterogeneity assumes use of different or dissimilar processors or cores that incorporate specialised processing capabilities to handle specific tasks (Scogland et al. 2014; Shan 2006). Such processors, due to their specialised capabilities, may be more energy efficient for specific tasks than general-purpose processors and/or can be put in a state where less power is used (or indeed deactivated if possible) when not required, thus, maximising both performance and energy efficiency (Scogland et al. 2014). GPUs, many integrated cores (MICs), and data flow engines (DFEs) are examples of co-processor architectures with relatively positive computation/power consumption ratios.² These architectures support heterogeneous computing because they are typically not standalone devices but are rather considered as co-processors to a host processor. As mentioned previously, the host processor can complete one instruction

stream, while the co-processor can complete a different instruction stream or type of stream (Eijkhout et al. 2016).

Modern GPUs are highly parallel programmable processors with high computation power. As can be derived from their name, GPUs were originally designed to help render images faster; however, wider adoption was hindered by the need for specialised programming knowledge. GPUs have a stream processing architecture fundamentally different than the widely known Intel general-purpose Central Processing Unit (CPU) programming models, tools, and techniques. As general-purpose GPU programming environments matured, GPUs were used for a wider set of specialist processing tasks including HPC workloads (Owens et al. 2007; Shi et al. 2012). Intel's Many-Integrated Core (MIC) architecture seeks to combine the compute density and energy efficiency of GPUs for parallel workloads without the need for a specialised programming architecture; MICs make use of the same programming models, tools, and techniques as those for Intel's general-purpose CPUs (Elgar 2010). DFEs are fundamentally different to GPUs and MICs in that they are designed to efficiently process large volumes of data (Pell and Mencer 2011). A DFE system typically contains, but is not restricted to, a field-programmable gate array (FPGA) as the computation fabric and provides the logic to connect an FPGA to the host, Random Access Memory for bulk storage, interfaces to other buses and interconnects, and circuitry to service the device (Pell et al. 2013). FPGAs are optimised processors for non-floating-point operations and provide better performance and energy efficiency for processing large volumes of integer, character, binary, and fixed point data (Proaño et al. 2014). Indeed, DFEs may be very inefficient for processing single values (Pell and Mencer 2011). A commonly cited use case for DFEs is high-performance data analytics for financial services. In addition to their performance, GPUs, MICs, and DFEs/FPGAs are attractive to HPC end users as they are programmable and therefore can be reconfigured for different use cases and applications. For example, as mentioned earlier, GPUs are now prevalent in many of the world's most powerful supercomputers.

It should be noted that while heterogeneity may provide higher computation/power consumption ratios, there are some significant implementation and optimisation challenges given the variance in operation and performance characteristics between co-processors (Teodoro et al. 2014). Similarly, application operation will depend on data access and the pro-

cessing patterns of the co-processors, which may also vary by application and co-processor type (Teodoro et al. 2014). For multi-tenant cloud computing, these challenges add to an already complex feature space where processors may not easily support virtualisation or where customers may require bare-metal provisioning thereby restricting resource pooling (Crago et al. 2011). For data-intensive application, data transmission to the cloud remains a significant barrier to adoption. Notwithstanding these challenges, cloud service providers have entered the HPC space with specialised processor offerings. For example, AWS now offers CPUs, GPUs, and DFEs/FPGAs, and has announced support for Intel Xeon Phi processors (Chow 2017).

1.6 ADDRESSING COMPLEXITY IN THE CLOUD THROUGH SELF-* DESIGN PRINCIPLES

This chapter previously discussed two computing paradigms—cloud computing and HPC—being driven by end-user demand for greater scale and performance. To achieve these requirements, heterogeneous resources, typically in the form of novel processor architectures, are being integrated into both cloud platforms and HPC systems. A side effect, however, is greater complexity—particularly in the case of hyperscale cloud services where the scale of infrastructure, applications, and number of end users is several orders of magnitude higher than general-purpose computing and HPC. This complexity in such large-scale systems results in significant management, reliability, maintenance, and security challenges (Marinescu 2017). Emergence and the related concept of self-organisation, self-management, and the separation of concerns are design principles that have been proposed as potential solutions for managing complexity in large-scale distributed information systems (Heylighen and Gershenson 2003; Schmeck 2005; Herrmann et al. 2005; Branke et al. 2006; Serugendo et al. 2011; Papazoglou 2012; Marinescu 2017).

The complexity of hyperscale cloud systems is such that it is effectively infeasible for cloud service providers to foresee and manage manually (let alone cost effectively) all possible configurations, component interactions, and end-user operations on a detailed level due to high levels of dynamism in the system. Self-organisation has its roots in the natural sciences and the study of natural systems where it has been long recognised that higher-level outputs in dynamic systems can be an emergent effect of lower-level

inputs (Lewes 1875). This is echoed in the field of Computer Science and through Alan Turing’s observation that “global order arises from local interactions” (Turing 1952). De Wolf and Holvoet (2004) define emergence as follows:

A system exhibits emergence when there are coherent emergent at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergent are novel with regards to the individual parts of the system.

(De Wolf and Holvoet 2004, p. 3)

Based on their review of the literature, De Wolf and Holvoet (2004) identify eight characteristics of emergent systems:

1. *Micro-macro effect*—the properties, behaviour, structure, and patterns situated at a higher macro-level that arise from the (inter)actions at the lower micro-level of the systems (so-called emergents).
2. *Radical novelty*—the global (macro-level) behaviour is novel with regard to the individual behaviours at the micro-level.
3. *Coherence*—there must be a logical and consistent correlation of parts to enable emergence to maintain some sense of identity over time.
4. *Interacting parts*—parts within an emergent system must interact as novel behaviour arises from interaction.
5. *Dynamical*—emergents arise as the system evolves over time; new attractors within the system appear over time and as a result new behaviours manifest.
6. *Decentralised control*—no central control directs the macro-level behaviour; local mechanism influences global behaviour.
7. *Two-way link*—there is a bidirectional link between the upper (macro-) and lower (micro-) levels. The micro-level parts interact and give rise to the emergent structure. Similarly, macro-level properties have causal effects on the micro-level.
8. *Robustness and flexibility*—no single entity can have a representation of the global emergent combined with decentralised control implies that no single entity can be a single point of failure. This introduces greater robustness, flexibility, and resilience. Failure is likely to be gradual rather than sudden in emergent systems.

Self-organising systems are similar in nature to emergent systems. Ashby (1947) defined a system as being self-organising where it is “at the same time (a) strictly determinate in its actions, and (b) yet demonstrates a self-induced change of organisation.” Heylighen and Gershenson (2003) define organisations as “structure with function” and self-organisation as a functional structure that appears and maintains spontaneously. Again, based on an extensive review of the literature, De Wolf and Holvoet (2004) offer a more precise definition of self-organisation as “a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control.” This definition is consistent with Heylighen and Gershenson (2003) while at the same time giving greater insight. De Wolf and Holvoet (2004) synthesise the essential characteristics of self-organising systems as:

1. *Increase in order*—an increase in order (or statistical complexity), through organisation, is required from some form of semi-organised or random initial conditions to promote a specific function.
2. *Autonomy*—this implies the absence of external control or interference from outside the boundaries of the system.
3. *Adaptability or robustness with respect to changes*—a self-organising system must be capable of maintaining its organisation autonomously in the presence of changes in its environment. It may generate different tasks but maintain the behavioural characteristics of its constituent parts.
4. *Dynamical*—self-organisation is a process from dynamism towards order.

The concept of self-organisation is often conflated with emergence, particularly in Computer Science due to the dynamism and robustness inherent in the systems and, frankly, historical similarity of language. While both emergent systems and self-organising systems are dynamic over time, they differ in how robustness is achieved. They can exist in isolation or in combination with each other. For example, Heylighen (1989) and Mamei and Zambonelli (2003) see emergent systems arising as a result of a self-organising process thus implying self-organisation occurs at the micro-level. In contrast, Parunak and Brueckner (2004) consider self-organisation as an effect at the macro-level of emergence as a result of increased order. Sudeikat et al. (2009) note that the systematic design of self-organising

systems is scarcely supported and therefore presents a number of challenges to developers including:

- Architectural design including providing self-organising dynamics as software components and application integration
- Methodological challenges including conceptual but practical means for designing self-organising dynamics by refining coordination strategies and supporting validation of explicit models for self-organised applications

Despite these challenges, De Wolf and Holvoet (2004) conclude for hugely complex systems “...we need to keep the individuals rather simple and let the complex behaviour self-organise as an emergent behaviour from the interactions between these simple entities.”

The concept of self-management is much more well defined in the Computer Science literature and has its roots in autonomic computing (Zhang et al. 2010). The concept of autonomic computing was popularised by IBM in a series of articles starting in 2001 with Horn’s “Autonomic Computing: IBM’s Perspective on the State of Information Technology.” These ideas were further elaborated by Kephart and Chess (2003) and Ganek and Corbi (2003) amongst others. For IBM, autonomic computing was conceptualised as “computing systems that can manage themselves given high-level objectives from administrators” (Kephart and Chess 2003). Kephart and Chess (2003) further elaborated the essence of autonomic computing systems through four aspects of self-management—self-configuration, self-optimisation, self-healing, and self-protection. In line with autonomic computing, the function of any self-management is the use of control or feedback loops, such as Monitor-Analyse-Plan-Execute-Knowledge (MAPE-K), that collect details from the system and act accordingly, anticipating system requirements and resolving problems with minimal human intervention (Table 1.2) (IBM 2005).

The so-called self-* aspects of IBM’s vision of autonomic computing are used in a wide range of related advanced technology initiatives and have been extended to include self-awareness, self-monitoring, and self-adjustment (Dobson et al. 2010). Despite the significant volume of research on self-management, like self-organisation, implementation of self-management presents significant challenges. These include issues related to the application of the agent-oriented paradigm, designing a component-based approach (including composition formalisms) for supporting self-management, managing relationships between autonomic

elements, distribution and decentralisation at the change management layer, design and implementation of robust learning and optimisation techniques, and robustness in a changing environment (Kramer and Magee 2007; Nami et al. 2006).

Research on the application of the principles of emergence, self-organisation, and self-management is widely referenced in Computer Science literature, typically discretely. There are few significant studies on architectures combining such principles. One such example is that of the Organic Computing project funded by the German Research Foundation (DFG). This research programme focused on understanding emergent global behaviour in “controlled” self-organising systems with an emphasis on distributed embedded systems (Müller-Schloer et al. 2011). However, for cloud computing architectures, there are relatively few examples. This is not to say that there is a dearth of applications of these concepts for specific cloud computing functions. There are numerous examples of bio-inspired algorithms for task scheduling (e.g. Li et al. 2011; Pandey et al. 2010), load balancing (Nishant et al. 2012), and other cloud-related functions. Similarly, Gutierrez and Sim (2010) describe a self-organising agent system for service composition in the cloud. However, these are all at the sub-system level. The relatively few cloud architectural studies, other than those relating to CloudLightning, are all the more surprising given that some commentators, notably, Zhang et al. (2010), posit that cloud computing systems are inherently self-organising. Such a proposition is not to

Table 1.2 Self-management aspects of autonomic computing (adapted from Kephart and Chess 2003)

<i>Concept</i>	<i>Description</i>	<i>Benefit</i>
Self-configuration	Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly.	Increased responsiveness
Self-optimization	Components and systems continually seek opportunities to improve their own performance and efficiency.	Increased operational efficiencies
Self-healing	System automatically detects, diagnoses, and repairs localised software and hardware problems.	Increased resilience
Self-protection	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent system-wide failures.	Increased security

dismiss self-management in cloud computing outright. Indeed, Zhang et al. (2010) admit that cloud computing systems exhibit autonomic features. However, a more purist interpretation suggests that these are not self-managing and do not explicitly aim to reduce complexity. Marinescu et al. (2013) emphasises the suitability of self-organisation as a design principle for cloud computing systems proposing an auction-driven self-organising cloud delivery model based on the tenets of autonomy of individual components, self-awareness, and intelligent behaviour of individual components including heterogeneous resources. Similarly, while self-management has been applied at a sub-system or node level (e.g. Brandic 2009), there are few studies on large-scale self-managing cloud architectures. One such system-level study is Puviani and Frei (2013) who, building on Brandic (2009), propose a catalogue of adaptation patterns based on requirements, context, and expected behaviour. These patterns are classified according to the service components and autonomic managers. Control loops following the MAPE-K approach enact adaptation. In their approach, each service component is autonomous and autonomic and has its own autonomic manager that monitors itself and the environment. The service is aware of changes in the environment including new and disappearing components and adapts on a negotiated basis with other components to meet system objectives. While Puviani and Frei (2013) and Marinescu et al. (2013) propose promising approaches, they are largely theoretical and their conclusions lack the data from real implementations.

While emergence, self-organisation, and self-management may prove to be principles for reducing overall system complexity, for a HPC use case, the issue of minimal interference remains. At the same time, surveys of the HPC end-user community emphasise the need for “ease of everything” in the management of HPC (IDC 2014). To create a service-oriented architecture that can cater for heterogeneous resources while at the same time shielding deployment and optimisation effort from the end user is not insignificant. As discussed, it is counter-intuitive to the conventional general-purpose model, which, in effect, is one-size-fits-all for end users. Separation of concerns is a concept that implements a “what-how” approach cloud architectures separating application lifecycle management and resource management. The end user, HPC, or otherwise, focuses its effort on what needs to be done, while the cloud service provider concentrates on how it should be done. In this way, the technical details for interacting with cloud infrastructure are abstracted away and instead the

end user or enterprise application developer provides (or selects) a detailed deployment plan including constraints and quality of service parameters using a service description language and service delivery model provided by the cloud service provider, a process known as blueprinting. Blueprinting empowers an “end-user-centric view” by enabling end users to use highly configurable service specification templates as building blocks to (re) assemble cloud applications quickly while at the same time maintain minimal interference with the underlying infrastructure (Papazoglou 2012). While there are a number of existing application lifecycle frameworks for PaaS (e.g. Apache Brooklyn and OpenStack Solum) and resource frameworks for IaaS (OpenStack Heat) that support blueprints, neither the blueprints nor the service delivery models have been designed to accommodate emergence, self-organisation, or self-management.

1.7 APPLICATION SCENARIOS

It is useful when reading further, to have one or more use cases in mind that might benefit from HPC in the cloud and more specifically a novel cloud computing architecture to exploit heterogeneity and self-* principles. Three motivating use cases are presented: (i) oil and case exploration, (ii) ray tracing, and (iii) genomics. These fall into the three HPC application categories discussed earlier, that is, tightly coupled applications, loosely coupled applications, and data-intensive applications. In each case, an architecture exploiting heterogeneous resources and built on the principles of self-organisation, self-management, and separation of concerns is anticipated to offer greater energy efficiency. By exploiting heterogeneous computing technologies, the performance/cost and performance/watt are anticipated to improve significantly. In addition, heterogeneous resources will enable computation to be hosted at hyperscale in the cloud, making large-scale compute-intensive applications and by-products accessible and practical from a cost and time perspective for a wider group of stakeholders. In each use case, even relatively small efficiency and accuracy gains can result in competitive advantage for industry.

1.7.1 *Oil and Gas Exploration*

The oil and gas industry makes extensive use of HPC to generate images of earth’s subsurface from data collected from seismic surveys as well as compute-intensive reservoir modelling and simulations. Seismic surveys

are performed by sending sound pulses into the earth or ocean, and recording the reflection. This process is referred to as a “shot”. To generate images in the presence of complex geologies, a computationally intensive process called Real-Time Migration (RTM) can be used. RTM operates on shots, and for each shot, it runs a computationally and data-expensive wave propagation calculation and a cross-correlation of the resulting data to generate an image. The images from each shot are summed to create an overall image. Similarly, the Open Porous Media (OPM) framework is used for simulating the flow and transport of fluids in porous media and makes use of numerical methods such as Finite Elements, Finite Volumes, Finite Differences, amongst others. These processes and simulations typically have not been operated in the cloud because of (a) data security, (b) data movement, and (c) poor performance. At the same time, on-site in-house HPC resources are often inadequate due to the “bursty” nature of processes where peak demand often exceeds compute resources. RTM and OPM are exemplars of tightly coupled applications.

One solution to address challenges and objections related to poor performance is to use a self-organising, self-managing cloud infrastructure to harness larger compute resources efficiently to deliver more energy and cost-efficient simulations of complex physics using OPM/Distributed and Unified Numeric Environment (DUNE). As well as supporting greater cloud adoption for HPC in the oil and gas sector, the development of a convenient scalable cloud solution in this space can reduce the risk and costs of dry exploratory wells. Relatively small efficiency and accuracy gains in simulations in the oil and gas industry can result in disproportionately large benefits in terms of European employment and Gross Domestic Product (GDP).

1.7.2 *Ray Tracing*

Ray tracing is widely used in image processing applications, such as those used in digital animation productions where the development of an image from a 3D scene is achieved by tracing the trajectories of light rays through pixels in a view plane. In recent years, the advancement of HPC and new algorithms has enabled the processing of large numbers of computational tasks in a much smaller time. Consequently, ray tracing has become a potential application for interactive visualisations. Ray tracing is commonly referred to as an “embarrassingly parallelisable algorithm” and is naturally

implemented in multicore shared memory systems and distributed systems. It is an example of a loosely coupled application.

Ray tracing has applications in a wide variety of industries including:

- Image rendering for high resolution and 3D images for the animation and gaming industry
- Human blockage modelling in radio wave propagation studies and for general indoor radio signal prediction
- Atmospheric radio wave propagation
- Modelling solar concentrator designs to investigate performance and efficiency
- Modelling laser ablation profiles in the treatment of high myopic astigmatism to assess the efficacy, safety, and predictability
- Development of improved ultrasonic array imaging techniques in anisotropic materials
- Ultrasonic imaging commonly used in inspection regimes, for example, weld inspections
- Modelling Light-emitting diode (LED) illumination systems

These industries have significant scale, and they increasingly rely on computationally intensive image processing, accelerated by innovations in consumer electronics, for example, HDTV and 3D TV. A variety of ray tracing libraries exist that are optimised for MIC and GPU platforms, for example, Intel Embree and NVIDIA Optix.

1.7.3 *Genomics*

Genomics is the study of all of a person's genes (the genome), including interactions of those genes with each other and with the person's environment. Since the late 1990s, academic and industry analysts have identified the potential of genomics to realise significant gains in development time and reduced investment, largely attached to realising efficiency gains. Genomics provides pharmaceutical companies with long-term upside and competitive advantage through savings right along the Research and Development (R&D) value chain (including more efficient target discovery, lead discovery, and development) but also in better decision-making accuracy resulting from more, better, and earlier information which ultimately results in higher drug success rates (Boston Consulting Group 2001). The net impact is that genomics can result in more successful drug

discovery. Relatively small efficiency and accuracy gains in the pharmaceutical industry can result in disproportionately large benefits in terms of employment and GDP. However, genome processing requires substantial computational power and storage requiring significant infrastructure and specialist IT expertise. While larger organisations can afford such infrastructure, it is a significant cost burden for smaller pharmaceutical companies, hospitals and health centres, and researchers. Even when such an infrastructure is in place, researchers may be stymied by inadequate offsite access.

Genomics has two core activities:

- *Sequencing*: a laboratory-based process involving “reading” deoxyribonucleic acid (DNA) from the cells of an organism and digitising the results
- *Computation*: the processing, sequence alignment, compression, and analysis of the digitised sequence

Historically, the cost of sequencing has represented the most significant percentage of the total. However, this cost has decreased dramatically over the past decade due to breakthroughs in research and innovation in that area. As the cost of sequencing has dropped, the cost of computation (alignment, compression, and analysis) has formed a greater proportion of the total. The biggest consumer of compute runtime is sequence alignment—assembling the large number of individual short “reads” which come out of the sequencer (typically, a few hundred bases long) into a single complete genome. This can be split into many processing jobs, each processing batches of reads and aligning against a reference genome, and run in parallel. Significant input data is required, but there is little or no inter-node communication needed. The most computationally intensive kernel in the overall process is local sequence alignment, using algorithms such as Smith Waterman, which is very well suited to being optimised through the use of heterogeneous compute technologies such as DFEs.

Genome processing is an exemplar of a data-intensive application. Greater energy efficiency is anticipated from using heterogeneous computing resulting in lower costs. As the cost of the raw sequencing technology drops, the computing challenge becomes the final significant technology bottleneck preventing the routine use of genomics data in clinical settings. Not only can the use of heterogeneous computing technologies offer significantly improved performance/cost and performance/

watt, but enabling this computation to be hosted at large-scale in the cloud makes it practical for wide-scale use. In addition to realigning the computation cost factors in genome processing with sequencing costs, a HPC solution can significantly improve the genome processing throughput and speed of genome sequence computation thereby reducing the wider cycle time thus increasing the volume and quality of related research. The benefits of such a cloud solution for genome processing are obvious. Researchers, whether in large pharmaceutical companies, genomics research centres, or health centres, can invest their energy and time in R&D and not managing and deploying complex on-site infrastructure.

1.8 CONCLUSION

This chapter introduces two computing paradigms—cloud computing and HPC, both of which are being impacted by technological advances in heterogeneous computing but also hampered by energy inefficiencies and increasing complexity. A combination of self-organisation, self-management, and the separation of concerns is proposed as design principles for a new hyperscale cloud architecture that can exploit the opportunities presented by heterogeneity to deliver more energy-efficient cloud computing and, in particular, support HPC in the cloud.

This book presents CloudLightning, a new way to provision heterogeneous cloud resources to deliver services, specified by the user, using a bespoke service description language. As noted, self-organising and self-managing systems present significant architecture design, methodological, and development challenges. These challenges are exacerbated when combined and considered at hyperscale. The remainder of this book presents CloudLightning's response to these challenges illustrating the utilisation of concepts in emergence, self-organisation, self-management, and the separation of concerns in a reference architecture for hyperscale cloud computing (Chap. 2).

Chapter 3 describes the self-organising and self-management formalisms designed to support coordination mechanisms within the CloudLightning architecture. As discussed earlier, stakeholders in cloud computing, and specifically HPC end users, have different concerns, for example, enterprise application developers and end users may want greater control over application lifecycle management, and cloud service providers want greater control over resource management. To support the separation of concerns and ease of use, a minimal-intrusive service delivery

model is presented in Chap. 4. This model uses a CloudLightning-specific service description language, blueprinting, and gateway service to enable enterprise application developers to specify comprehensive constraints and quality of service parameters for services and/or resources and, based on the specified constraints and parameters, provide an optimal deployment of the resources.

Finally, Chap. 5 addresses the issue of validation of such a novel architecture. As per Sudeikat et al. (2009), the validation of self-organising models summatively and formatively presents significant challenges that are further complicated at hyperscale. Chapter 5 presents CloudLightning's work on the design and implementation of a Warehouse-Scale cloud simulator for validating the performance of CloudLightning.

1.9 CHAPTER 1 RELATED CLOUDLIGHTNING READINGS

1. Lynn, T., Xiong, H., Dong, D., Momani, B., Gravvanis, G. A., Filelis-Papadopoulos, et al. (2016, April). CLOUDLIGHTNING: A framework for a self-organising and self-managing heterogeneous Cloud. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), 1 and 2* (pp. 333–338). SCITEPRESS-Science and Technology Publications, Lda.
2. Lynn, T., Kenny, D., & Gourinovitch, A. (2015). Global HPC market. Retrieved November 6, 2017, from <https://cloudlightning.eu/?ddownload=2446>
3. Lynn, T., Kenny, D., Gourinovitch, A., Persehais, A., Tierney, G., Duignam, M., et al. (2015). 3D image rendering. Retrieved November 6, 2017, from <https://cloudlightning.eu/?ddownload=2435>
4. Lynn, T., & Gourinovitch, A. (2016). Overview of the HPC market for genome sequence. Retrieved November 6, 2017, from <https://cloudlightning.eu/?ddownload=2443>
5. Lynn, T., Gourinovitch, A., Kenny, D., & Liang, X. (2016). Drivers and barriers to using high performance computing in the cloud. Retrieved November 6, 2017, from <https://cloudlightning.eu/?ddownload=2904>
6. Callan, M., Gourinovitch, A., & Lynn, T. (2016). The Global Data Center market. Retrieved November 6, 2017, from <https://cloudlightning.eu/?ddownload=3588>

NOTES

1. <https://www.top500.org/green500/>
2. There are other niche processor solutions worth exploring including Automata Processors for graph analysis, pattern matching, and data analytics; Digital Signal Processor for processing real-world analogue signals; Application-Specific Integrated Circuits (ASICs) for use cases such as bitcoin mining; and neuromorphic chips for cognitive computing. For more discussion, see Zahran (2017).

REFERENCES

- Amazon Web Services. (2015). An introduction to high performance computing on AWS [White Paper]. Seattle, WA: Amazon Web Services. Retrieved October 23, 2017, from https://d0.awsstatic.com/whitepapers/Intro_to_HPC_on_AWS.pdf
- Ashby, W. R. (1947). Principles of the self-organizing dynamic system. *The Journal of General Psychology*, 37(2), 125–128.
- Awada, U., Li, K., & Shen, Y. (2014). Energy consumption in cloud computing data centres. *International Journal of Cloud Computing and Services Science*, 3(3), 145.
- Barroso, L. A., & Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12), 33–37.
- Boston Consulting Group (2001). *A revolution in R&D—How genomics and genetics are transforming the bio-pharmaceutical industry*. Boston, MA: Boston Consulting Group. Retrieved October 23, 2017, from <https://www.bcg.com/documents/file13745.pdf>
- Brandic, I. (2009, July). Towards self-manageable cloud services. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, Vol. 2 (pp. 128–133). IEEE.
- Branke, J., Mnif, M., Muller-Schloer, C., & Prothmann, H. (2006, November). Organic computing—Addressing complexity by controlled self-organization. In *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006 (ISoLA 2006)* (pp. 185–191). IEEE.
- Chow, O. (2017). AWS & Intel: A partnership dedicated to cloud innovations [PowerPoint slides]. *SlideShare*. Retrieved October 23, 2017, from <https://www.slideshare.net/AmazonWebServices/aws-intel-a-partnership-dedicated-to-cloud-innovations-77355517>
- Cisco. (2016). Cisco Global Cloud index: Forecast and methodology, 2015–2020 [White Paper]. *Cisco*. Retrieved October 23, 2017, from <https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf>

- Clark, J. (2014, November 11). 5 numbers that illustrate the mind-bending size of Amazon's cloud. *Bloomberg*. Retrieved October 23, 2017, from <https://www.bloomberg.com/news/2014-11-14/5-numbers-that-illustrate-the-mind-bending-size-of-amazon-s-cloud.html>
- Crago, S., Dunn, K., Eads, P., Hochstein, L., Kang, D. I., Kang, M., Modium, D., Singh, K., Suh, J., & Walters, J. P. (2011, September). Heterogeneous cloud computing. In *2011 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 378–385). IEEE.
- Crago, S. P., & Walters, J. P. (2015). Heterogeneous cloud computing: The way forward. *Computer*, 48(1), 59–61.
- Data Center Knowledge. (2017, March 16). Google Data Center FAQ. *Data Center Knowledge*. Retrieved October 23, 2017, from <http://www.datacenterknowledge.com/archives/2017/03/16/google-data-center-faq>
- De Wolf, T., & Holvoet, T. (2004, July). Emergence versus self-organisation: Different concepts but promising when combined. In *International Workshop on Engineering Self-organising Applications* (pp. 1–15). Berlin: Springer.
- Dobson, S., Sterritt, R., Nixon, P., & Hinchey, M. (2010). Fulfilling the vision of autonomic computing. *Computer*, 43(1), 35–41.
- Dongarra, J. (2016, June 24). Report on the Sunway TaihuLight System. Retrieved November 7, 2017, from <http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf>
- Eijkhout, V., van de Geijn, R., & Chow, E. (2016). Introduction to high performance scientific computing. *Zenodo*. <https://doi.org/10.5281/zenodo.49897>
- Elgar, T. (2010, December). Intel Many Integrated Core (MIC) architecture [PowerPoint Slides]. In *2nd UK GPU Computing Conference*, December 2010. Retrieved October 23, 2017, from <http://www.many-core.group.cam.ac.uk/ukgpucc2/talks/Elgar.pdf>
- Esmailzadeh, H., Blem, E., St Amant, R., Sankaralingam, K., & Burger, D. (2011, June). Dark silicon and the end of multicore scaling. *ACM SIGARCH Computer Architecture News*, 39(3), 365–376. ACM.
- Ezell, S. J., & Atkinson, R. D. (2016, April). *The vital importance of high-performance computing to US competitiveness*. Washington, DC: Information Technology and Innovation Foundation. Retrieved October 23, 2017, from <http://www2.itif.org/2016-high-performance-computing.pdf>
- Feldman, M. (2016, June 20). China tops supercomputer rankings with new 93-Petaflop Machine. *TOP500.org*. Retrieved October 23, 2017, from <https://www.top500.org/news/china-tops-supercomputer-rankings-with-new-93-petaflop-machine/>
- Ganek, A. G., & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1), 5–18.
- Garfinkel, S. (1999). *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. MIT Press.

- Gutierrez-Garcia, J. O., & Sim, K. M. (2010, November). Self-organizing agents for service composition in cloud computing. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 59–66). IEEE.
- Herrmann, K., Muhl, G., & Geihs, K. (2005). Self management: The solution to complexity or just another problem? *IEEE Distributed Systems Online*, 6(1), 1.
- Heylighen, F. (1989). Self-organization, emergence and the architecture of complexity. In *Proceedings of the 1st European Conference on System Science* (pp. 18, 23–32). Paris: AFCET.
- Heylighen, F., & Gershenson, C. (2003). The meaning of self-organization in computing. *IEEE Intelligent Systems*, 18(4), 72–75.
- Horn, P. (2001). Autonomic computing: IBM's perspective on the state of information technology. *IBM*. Retrieved October 23, 2017, from http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf
- IBM. (2005). An architectural blueprint for autonomic computing [White Paper]. *IBM*. Retrieved October 23, 2017, from <https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>
- IDC. (2014). *Market analysis perspective: Worldwide HPC, 2014—Directions, trends, and customer requirements*. Framingham, MA.
- IDC. (2015). *High performance computing in the EU: Progress on the implementation of the European HPC strategy*. Brussels, Belgium: European Commission.
- Intersect360 Research. (2014). *Worldwide high performance computing 2013: Total market model and 2014–18 forecast*. Sunnyvale, CA.
- Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1), 41–50.
- Kim, W. (2009). Cloud computing: Today and tomorrow. *Journal of Object Technology*, 8(1), 65–72.
- Kramer, J., & Magee, J. (2007, May). Self-managed systems: An architectural challenge. In *2007 Future of Software Engineering* (pp. 259–268). IEEE Computer Society.
- Lewes, G. (1875). *Problems of life and mind* (Vol. 2). London, UK: Kegan, Paul, Trench, Turbner.
- Li, J. F., Peng, J., Cao, X., & Li, H. Y. (2011). A task scheduling algorithm based on improved ant colony optimization in cloud computing environment. *Energy Procedia*, 13, 6833–6840.
- Licklider, J. C. (1963). Memorandum for members and affiliates of the intergalactic computer network. *Advanced Research Projects Agency*. Washington, DC. Retrieved October 23, 2017, from <http://www.kurzweilai.net/memorandum-for-members-and-affiliates-of-the-intergalactic-computer-network>

- Mamei, M., & Zambonelli, F. (2003, July). Self-organization in multi agent systems: A middleware approach. In *International Workshop on Engineering Self-organising Applications* (pp. 233–248). Berlin: Springer.
- Marinescu, D. (2017). *Complex systems and clouds—A self-organization and self-management perspective*. Cambridge, MA: Elsevier.
- Marinescu, D. C., Paya, A., Morrison, J. P., & Healy, P. (2013, December 12). An auction-driven self-organizing cloud delivery model. Retrieved October 23, 2017, from arXiv preprint <https://arxiv.org/abs/1312.2998> [cs.DC].
- Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing. Special Publication 800–145*. Gaithersburg, MD: National Institute of Standards and Technology.
- Mergen, M. F., Uhlig, V., Krieger, O., & Xenidis, J. (2006). Virtualization for high-performance computing. *ACM SIGOPS Operating Systems Review*, 40(2), 8–11.
- Müller-Schloer, C., Schmeck, H., & Ungerer, T. (Eds.). (2011). *Organic computing—A paradigm shift for complex systems*. Springer Science & Business Media.
- Nami, M. R., Bertels, K., & Vassiliadis, S. (2006, November). Autonomic computing systems: Issues and challenges. In *17th Annual Workshop on Circuits, Systems and Signal Processing*.
- National Supercomputing Centre, WuXi. (n.d.). Hardware. Retrieved October 23, 2017, from <http://www.nscwx.cn/wxcyw/soft1.php?word=soft&ci=46>
- Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., & Rastogi, R. (2012, March). Load balancing of nodes in cloud using ant colony optimization. In *2012 UKSim 14th International Conference on Computer Modelling and Simulation (UKSim)* (pp. 3–8). IEEE.
- Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., et al. (2007, March). A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1), 80–113. Blackwell Publishing.
- Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)* (pp. 400–407). IEEE.
- Papazoglou, M. P. (2012). Cloud blueprints for integrating and managing cloud federations. In *Software service and application engineering* (pp. 102–119). Berlin: Springer.
- Parunak, H. V. D., & Brueckner, S. A. (2004). Engineering swarming systems. In *Methodologies and Software Engineering for Agent Systems* (pp. 341–376). Springer US.
- Pell, O., & Mencer, O. (2011). Surviving the end of frequency scaling with reconfigurable dataflow computing. *ACM SIGARCH Computer Architecture News*, 39(4), 60–65.

- Pell, O., Mencer, O., Tsoi, K. H., & Luk, W. (2013). Maximum performance computing with dataflow engines. In *High-performance computing using FPGAs* (pp. 747–774). New York: Springer.
- Proaño, J., Carrión, C., & Caminero, M. B. (2014, April). An open-source framework for integrating heterogeneous resources in Private Clouds. In *4th International Conference on Cloud Computing and Services Science (CLOSER 2014)* (pp. 129–134). INSTICC.
- Puviani, M., & Frei, R. (2013, October). Self-management for cloud computing. In *Science and Information Conference (SAI)* (pp. 940–946). IEEE.
- Ray, J., Trebon, N., Armstrong, R. C., Shende, S., & Malony, A. (2004, April). Performance measurement and modeling of component applications in a high performance computing environment: A case study. In *Proceedings of 18th International Conference on Parallel and Distributed Processing Symposium, 2004* (p. 95), IEEE.
- Rogers, P., & Fellow, A. C. (2013, August). Heterogeneous system architecture overview. *Hot Chips*, 25.
- Schmeck, H. (2005, May). Organic computing—A new vision for distributed embedded systems. In *Eighth IEEE International Symposium on Object-oriented Real-time Distributed Computing, 2005 (ISORC 2005)* (pp. 201–203). IEEE.
- Scogland, T. R., Steffen, C. P., Wilde, T., Parent, F., Coghlan, S., Bates, N., et al. (2014, March). A power-measurement methodology for large-scale, high-performance computing. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering* (pp. 149–159). ACM.
- Serugendo, G. D. M., Gleizes, M. P., & Karageorgos, A. (2011). Self-organising systems. In *Self-organising software* (pp. 7–32). Berlin: Springer.
- Severance, C., & Dowd, K. (2010). *High performance computing*. Houston, TX: Connexions.
- Shan, A. (2006). Heterogeneous processing: A strategy for augmenting Moore's law. *Linux Journal*, 142, 7.
- Shi, L., Chen, H., Sun, J., & Li, K. (2012). vCUDA: GPU-accelerated high-performance computing in virtual machines. *IEEE Transactions on Computers*, 61(6), 804–816.
- Sudeikat, J., Braubach, L., Pokahr, A., Renz, W., & Lamersdorf, W. (2009). Systematically engineering self-organizing systems: The SodekoVS approach. *Electronic Communications of the EASST*, 17.
- Teodoro, G., Kurc, T., Kong, J., Cooper, L., & Saltz, J. (2014, May). Comparative performance analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: A case study from microscopy image analysis. In *Proceedings of 28th International Conference on Parallel and Distributed Processing Symposium, 2014 IEEE* (pp. 1063–1072). IEEE.

- Trader, T. (2017, June 19). Top500 results: Latest list trends and what's in store. Retrieved November 7, 2017, from <https://www.hpcwire.com/2017/06/19/49th-top500-list-announced-isc/>
- Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 237(641), 37–72.
- Yeo, S., & Lee, H. H. (2011). Using mathematical modeling in provisioning a heterogeneous cloud computing environment. *Computer*, 44(8), 55–62.
- Zahran, M. (2017). Heterogeneous computing: Here to stay. *Communications of the ACM*, 60(3), 42–45.
- Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1), 7–18.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this book or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.