Chapter 1

Basics

Upon completion of this chapter the reader will understand:

1. The standard definition, characteristics, and benefits of cloud computing
2. The key roles in the cloud computing ecosystem
3. Key concepts of application, demand, supply, quality, and fungibility

This section reviews the following key concepts that are used extensively in this work:

- Cloud computing fundamentals (Section 1.1)
- Roles in cloud computing (Section 1.2)
- Applications (Section 1.3)
- Demand, supply, capacity, and fungibility (Section 1.4)
- Demand variability (Section 1.5)

1.1 CLOUD COMPUTING FUNDAMENTALS

Cloud computing is defined by ISO/IEC 17788[1] as a "*paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with*

[1] In 2011, the US Government National Institute of Standards and Technology (NIST) published the seminal "*NIST Definition of Cloud Computing*" (NIST SP 800-145) and "*NIST Cloud Computing Reference Architecture*" (NIST SP 500-292). In 2014, the International Standards Organization (ISO) published "*Cloud Computing Overview and Vocabulary*" (ISO/IEC 17788) and "*Cloud Computing Reference Architecture*" (ISO/IEC 17789), which are fundamentally consistent with the 2011 NIST documents. This work is consistent with both of these suites of standards.

Lean Computing for the Cloud, First Edition. Eric Bauer.

self-service provisioning and administration on-demand." ISO/IEC 17788 specifies the following six key characteristics of cloud computing[2]:

1. **Broad network access** – "*A feature where the physical and virtual resources are available over a network and accessed through standard mechanisms that promote use by heterogeneous client platforms. The focus of this key characteristic is that cloud computing offers an increased level of convenience in that* ***users can access physical and virtual resources from wherever they need to work****, as long as it is network accessible, using a wide variety of clients including devices such as mobile phones, tablets, laptops, and workstations*" (ISO/IEC 17788). Operationally, this means that end users access cloud-based application services via commonly available wireless and wireline IP networks.
2. **Measured service** – "*A feature where the metered delivery of cloud services is such that usage can be monitored, controlled, reported, and billed… The focus of this key characteristic is that* ***the customer may only pay for the resources that they use****. From the customers' perspective, cloud computing offers the users value by enabling a switch from a low efficiency and asset utilization business model to a high efficiency one*" (ISO/IEC 17788). When cloud customers pay only for resources that are used, application services that are engineered so cloud resource usage tracks with application service usage which tracks with application revenue can reduce business risk by better linking application service provider's costs with application service revenues.
3. **Multi-tenancy** – "*A feature where physical or virtual resources are allocated in such a way that multiple tenants and their computations and data are isolated from and inaccessible to one another*" (ISO/IEC 17788). Multi-tenancy enables infrastructure service providers to maximize resource sharing and boost their capacity utilization.
4. **On-demand self-service** – "*A feature where a cloud service customer can provision computing capabilities, as needed, automatically or with minimal interaction with the cloud service provider. The focus of this key characteristic is that* ***cloud computing offers users a relative reduction in costs, time, and effort needed to take an action, since it grants the user the ability to do what they need, when they need it, without requiring additional human user interactions or overhead***" (ISO/IEC 17788). This means that application service providers and/or automated systems working on behalf of those application operators can install, configure, and provision cloud resources to serve their applications in real time. On-demand self-service of capacity planning and fulfillment actions, coupled with rapid elasticity, enables significant reductions in fulfillment times for capacity change actions compared to traditional deployments.

[2] The six ISO/IEC 17788 key characteristics of cloud computing are fundamentally the five essential characteristics of cloud computing offered by NIST in SP 800-145, plus *multi-tenancy*.

5. **Rapid elasticity and scalability** – "*A feature where physical or virtual resources can be rapidly and elastically adjusted, in some cases automatically, to quickly increase or decrease resources. For the cloud service customer, the physical or virtual resources available for provisioning often appear to be unlimited and can be purchased in any quantity at any time automatically, subject to constraints of service agreements. Therefore, the focus of this key characteristic is that cloud computing means that the* ***customers no longer need to worry about limited resources and might not need to worry about capacity planning***" (ISO/IEC 17788). Application service providers and/or automated systems working on their behalf can allocate and release infrastructure resources on-the-fly, thereby enabling applications to transform from allocating and configuring capacity based on peak forecast demand (which may never even be approached) to just-in-time, demand-driven capacity configuration.
6. **Resource pooling** – "*A feature where a cloud service provider's physical or virtual resources can be aggregated in order to serve one or more cloud service customers… From the customer's perspective, all they know is that the service works, while they generally have no control or knowledge over how the resources are being provided or where the resources are located. This offloads some of the customer's original workload, such as maintenance requirements, to the provider*" (ISO/IEC 17788). Resource pooling, coupled with multi-tenancy, enables cloud service providers (CSPs) to leverage economies of scale to boost operational efficiencies beyond what has traditionally been feasible.

Figure 1.1 offers a simplified view of cloud computing: many different applications simultaneously share a large and flexibly configured pool of compute, memory,

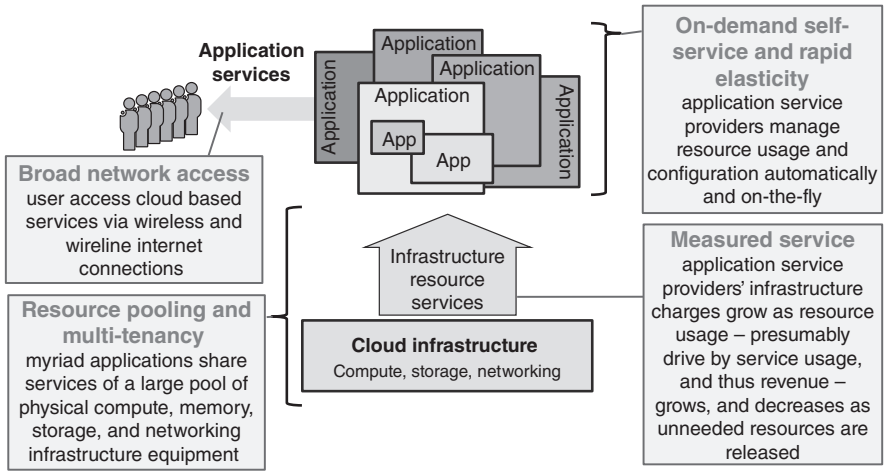

Figure 1.1 Cloud Computing in a Nutshell

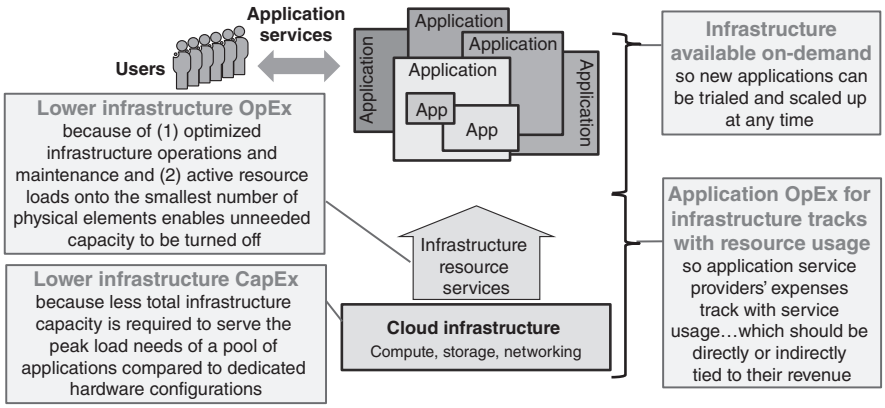

Figure 1.2 High-Level Benefits of Cloud Computing

storage, networking, and functional component resources offered by a CSP. Coupling on-demand self-service and rapid elasticity enables application service providers to fundamentally transform their businesses from configuring application capacity to meet peak forecast demand to just-in-time, demand-driven capacity management. Using measured service as the basis of charging application service providers for their resource usage enables at least a portion of the application provider's costs to shift from being fixed and independent of usage to variable so that resource charges are tied to resource usage, which is tied to application service usage, which should be tied to application revenue.

Figure 1.2 visualizes key high-level business benefits of cloud computing:

1. **Lower infrastructure CapEx** – the peak *aggregate* demand for compute, storage, and networking resources is smaller than the sum of peak demand of each individual application. Infrastructure resource sharing enables less total hardware to be deployed than with traditional operation models, and thus capital expense (CapEx) can be saved. Also, consolidating operation of that shared infrastructure enables efficiencies that can reduce operating expense (OpEx).
2. **Infrastructure available on demand** – the ability of application service providers to allocate and release infrastructure resources on-the-fly enables agile, just-in-time application capacity management to minimize the twin business risks of over investing in application capacity that is never used (if demand forecasts are too optimistic) or being unable to serve customer demand (if demand forecasts are too conservative).
3. **Lower infrastructure OpEx** because a single infrastructure service provider can operate and maintain shared infrastructure cheaper than each application service provider can operate and maintain their own private infrastructure. In addition, consolidating active resource loads onto the smallest number of physical elements enables unneeded capacity to be turned off.

4. **Application OpEx for infrastructure tracks with resource usage** thereby derisking application service provider businesses by having their OpEXs for resource capacity track with service usage, and thus hopefully with revenue.

Cloud enables disruption of IT service management in several ways:

- **Encourage agile, incremental development and deployment models** – these practices can reduce the organization's business risks around development activities by enabling applications and services to be developed and deployed via smaller project tasks which can generally be better managed rather than relying on massive software projects that are harder to manage.
- **Elastic capacity enables new operational models for application release management** – infrastructure available on-demand means that instead of relying on complex software upgrade, update, or retrofit procedures that must execute "in-place" on traditional hardware, an independent set of infrastructure resources can be allocated to host the new software upgrade, update, or retrofit and that release maintenance action can largely proceed without impacting the production instance thereby both reducing service risk for users and operational complexity (and presumably costs) for application service providers.
- **Fungibility of virtualized resources enables workload aggregation** – virtualization enables commodity physical compute, memory, and storage resources to be configured on-the-fly to serve a huge range of applications. Thus, rather than planning, installing, and allocating explicit physical resources to dedicate to each application in advance, a pool of commodity hardware can be created, with virtual resources allocated to applications on-the-fly.
- **Virtual resources are ephemeral** – application service providers can allocate and deallocate virtual resources as needed to better align their resource usage – and thus presumably their resource costs – with service usage which is presumably linked to their revenue. Leveraging ephemeral resources and usage-based resource pricing to better align costs with revenue is fundamentally different from the traditional model where resource (i.e., infrastructure equipment) costs are a sunk cost for the application service provider to manage.
- **Virtual compute, memory, storage, and networking resources released by applications can easily be reallocated to other applications** – virtualization increases the fungibility of physical infrastructure resources by enabling physical resources to emulate a range of resource configurations that might be desired by application software. Greater fungibility means that each type of physical resource can serve a broader range of application needs, thereby permitting more efficient resource utilization.
- **Automation of virtual resource management and orchestration can:**
  - yield short and consistent lead times for application capacity management changes;
  - yield more reliable application capacity management changes;
  - lower fulfillment costs per capacity management change action.

- **On-the-fly virtual resource allocation and operations enables aggressive power management** – this topic is detailed in Chapter 9: Lean Infrastructure Commitment.

1.2 ROLES IN CLOUD COMPUTING

Figure 1.3 visualizes the primary ISO/IEC 17788 roles in cloud computing:

- **Cloud Service User** is an end user, or an application operating on their behalf, who enjoy cloud services such as social networking applications or watching streaming movies.
- **Cloud Service Customers** (CSCs) are organizations that operate cloud services for cloud service users, such as an organization that offers streaming entertainment or real-time communications services to end users via cloud-based applications. For example, a company that offers streaming movie services to end users by deploying their application software onto some other organization's infrastructure-as-a-service (IaaS) offering is a CSC. This role is analogous to NIST SP 500-292 *Cloud Consumer*. As the notion of customer or consumer often creates confusion, this book will refer to this role with the more commonly used term *application service provider*.
- **Cloud Service Provider** (CSP) is broadly defined by ISO/IEC 17788 as a "*party which makes cloud services available*"; this role is analogous to NIST SP 500-292 *Cloud Provider*. Some CSPs offer virtualized compute, networking, storage, and memory resources to application service providers (CSCs) as IaaS offerings. Virtual compute, memory, storage, and networking can be offered by infrastructure service providers to application service providers via technologies such as hypervisors, Linux containers, and other virtualization

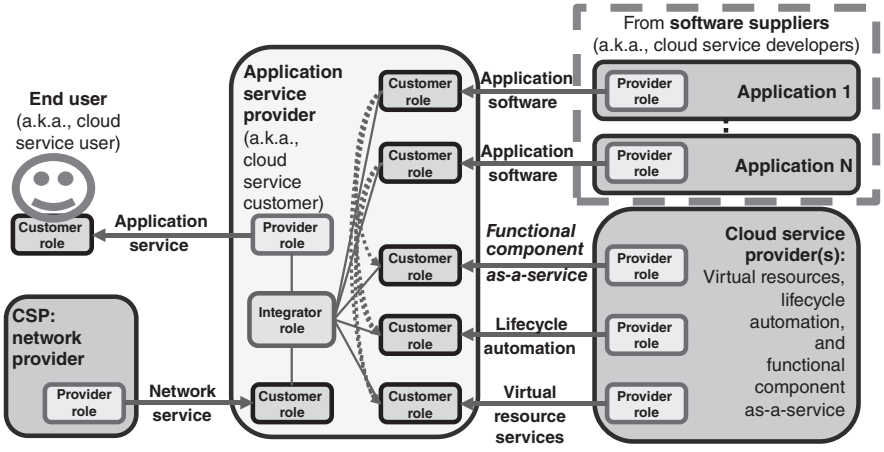

Figure 1.3 Cloud Service Delivery Relationships

mechanisms. For convenience, this paper will refer simply to virtual machines (VMs) as the primary unit of infrastructure capacity, so this should be understood to cover Linux containers and other implementation options as well. ISO/IEC 17789 defines *functional component* as "*A functional building block needed to engage in an activity, backed by an implementation*"; some CSPs will offer functional components such as databases or load balancers to consumers via platform-as-a-service offerings. While there is some disagreement across the industry regarding exactly what functionality offered as-a-service should be classified as platform-as-a-service versus software-as-a-service (or even IaaS), the exact taxonomy is unimportant for this analysis. For simplicity, this book will refer to this role as *infrastructure service provider*. Note that there may actually be multiple service provider organizations, such as one organization operating virtualized compute, memory, storage, and networking infrastructure and another organization operating higher-level functional components like database-as-a-service, but that distinction is also unimportant for this analysis.

- **CSP: Network Provider** is defined by ISO/IEC 17789 as a party which "*may provide network connectivity between systems within the cloud service provider's data centre, or provide network connectivity between the cloud service provider's systems and systems outside the provider's data centre, for example, cloud service customer systems or systems belonging to other cloud service providers.*" This role is analogous to NIST SP 500-292 *Cloud Carrier*.
- **Cloud Service Partners** (CSN) are defined by ISO/IEC 17788 as a "*party which is engaged in support of, or auxiliary to, activities of either the cloud service provider or the cloud service customer, or both*." The three partner roles enumerated in ISO/IEC 17789 are:
  - **Cloud Service Developers** are "*responsible for designing, developing, testing and maintaining the implementation of a cloud service*" (ISO/IEC 17789). This book will generally refer to this role as *software suppliers*.
  - **Cloud Auditor** defined by ISO/IEC 17788 as a "*Cloud service partner with the responsibility to conduct an audit of the provision and use of cloud services*." This role is analogous to NIST SP 500-292 *Cloud Auditor*. This role is not shown in Figure 1.3 and this book will not consider this role.
  - **Cloud Service Broker** defined by ISO/IEC 17788 as a "*cloud service partner that negotiates relationships between cloud service customers and cloud service providers.*" This role is analogous to NIST SP 500-292 *Cloud Broker*. This role is not shown in Figure 1.3 and this book will not consider this role.

Note that the primary service relationships among roles visualized in Figure 1.3 are shown in the style of TMF GB917 and TMF TR178 in Figure 1.3. The application service provider offers valuable services to end users which are delivered across one or more CSP network providers' (or cloud carriers') networks. The application service provider's service offering is the result of integrating one or more applications from

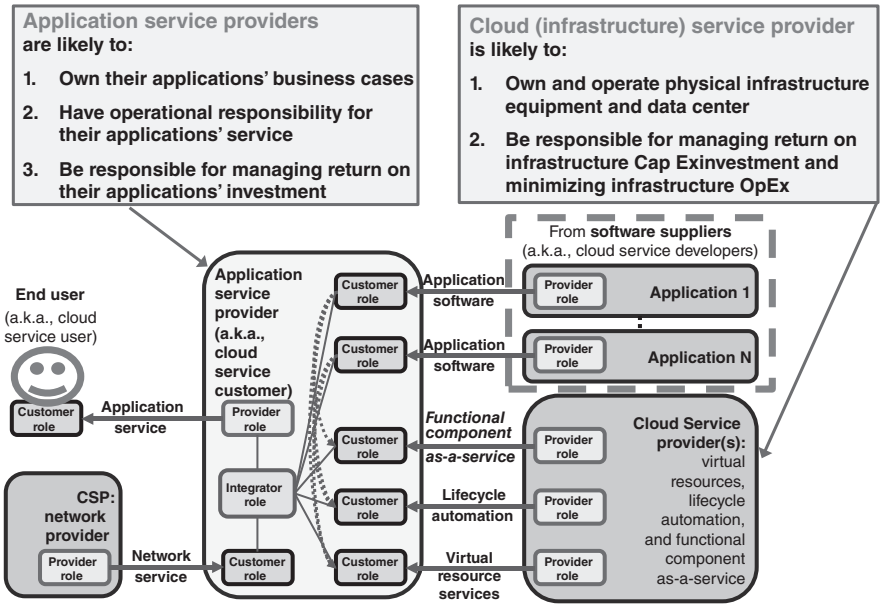

Figure 1.4 Responsibilities in the Context of Cloud Service Delivery Relationships

software suppliers with virtual compute, networking, storage and memory, along with automated lifecycle management services to instantiate, operate, and manage the application instances, and perhaps some functional components offered as-a-service by one or more CSPs.

Figure 1.4 visualizes the simplified responsibilities of the primary actors of Figure 1.3. The cloud (infrastructure) service provider organization owns and operates the pool of physical and virtual resources that are offered to various application service provider (CSC) organizations. Each application service provider organization is responsible for engineering, integration, and operation of their user facing service offering. Figure 1.5 overlays simplified responsibilities onto Figure 1.1.

The two most common standard cloud deployment models discussed in this work are:

- **Public cloud** is defined by ISO/IEC 17788 as "*Cloud deployment model where cloud services are potentially available to any cloud service customer and resources are controlled by the cloud service provider. A public cloud may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud service provider.*"
- **Private cloud** is defined by ISO/IEC 17788 as "*Cloud deployment model where cloud services are used exclusively by a single cloud service customer and resources are controlled by that cloud service customer. A private cloud may be owned, managed, and operated by the organization itself or a third party and may exist on premises or off premises.*"

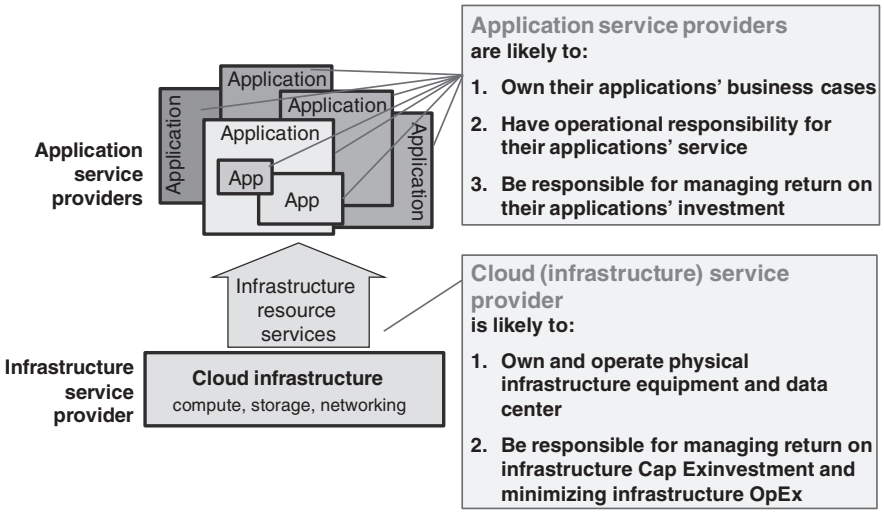

Figure 1.5 Simplified Responsibilities

Cloud service users do not care – and often do not even know – if the application that they enjoy is hosted on a public or private cloud. The distinction of public or private cloud is of practical importance for both the application service provider and the organization that owns and operates the shared cloud infrastructure as it drives the governance and business model for the relationship between those organizations.

1.3 APPLICATIONS

Application is defined by ITIL as software that provides Functions that are required by an IT Service. Applications are composed of software components that execute on physical compute, networking, memory, and storage equipment. An application instance is an operational configuration of application software components on appropriate compute, networking, memory, and storage infrastructure elements that can offer service to users.

Figure 1.6 shows a logical topology diagram of a sample application. As will be discussed in Section 2.5.3.1: Scale Capacity of a Single Application Instance, online application capacity can be adjusted by independently configuring the number of online application logic (AppLogic) instances and application storage (AppStorage) instances. Note that the sample application is architected to have only a single pair of frontend components to distribute user workload across the pool of AppLogic component instances, and a single pair of operations, administration, maintenance, and provisioning (OAMP) components manages all of the application's component instances.

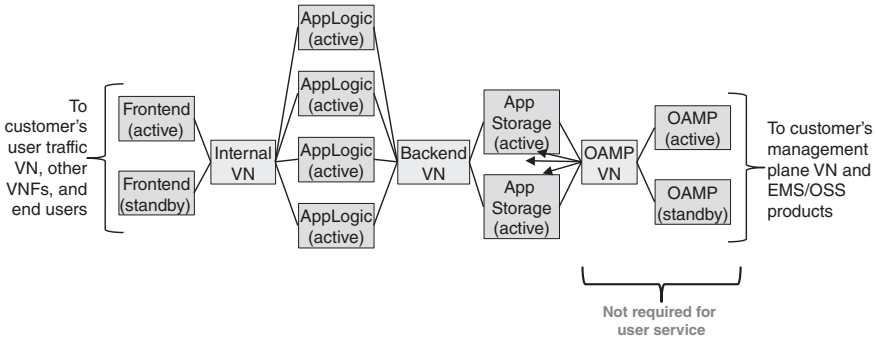

Figure 1.6 Topology of Simple, Sample Application

Workloads within an application instance can often be distributed across pools of component instances, which this paper will call *load balancing*. Our sample application supports two levels of *intra*-application load balancing: the active Frontend component balances user workload across the pool of available AppLogic components (illustrated in Figure 1.7); and each AppLogic component balances their database workload across the pool of available AppStorage components. Intra-application workload balancing is typically managed by the load balancing/distribution component itself without direct decisions, support, or coordination from other elements like element management systems, operations support systems, or business support systems. Note that application protocols and architectures may constrain workload distribution strategies to maximize user quality of experience or for other reasons. For example, once a user's session has been established on a particular AppLogic instance all operations associated with that session will normally be served by that instance to minimize service latency and disruption by leveraging user session state information that has been cached by the particular AppLogic instance.

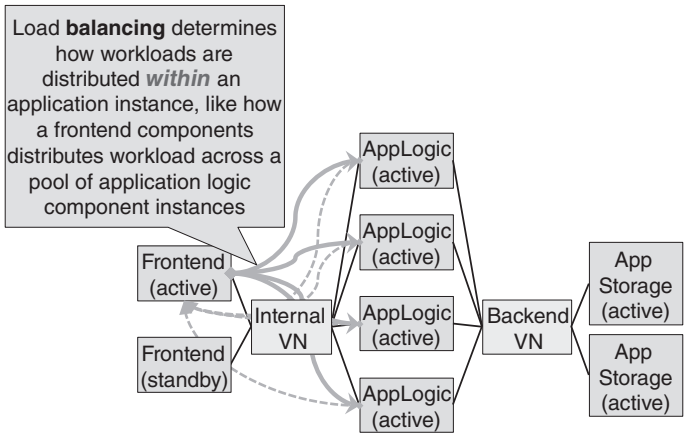

Figure 1.7 Intra-Application Workload Distribution via Load Balancing

Users' client application instances typically find an instance of the sample application via DNS, so altering the IP addresses of sample application instances returned for DNS queries is the primary mechanism for distributing user workload across application instances; discrete load balancer elements are also used to distribute workloads across multiple application instances. The application server and client also support a redirect mechanism (e.g., return code 302 Moved Temporarily) by which a frontend application component instance can redirect a particular client to another application instance, such as when the application instance is offline for maintenance. User workload can also be distributed across multiple application instances, as discussed in Section 2.5.3.3: Redistribute Workload between Application Instances.

1.3.1 Application Service Quality

Application service delivered to individual end users experiences four fundamental service qualities:

- **Accessibility** – service *accessibility* is defined by IEC 60050:191 as "*the probability that a service can be obtained within specified tolerances and other given operating conditions when requested by the user.*"[3] For example, was the end user able to establish a service session within an acceptable time (typically seconds)? Thus a particular service access attempt is either successful or unsuccessful (including unacceptably slow).
- **Retainability** – service *retainability* is defined by IEC 60050:191 as "*the probability that a service, once obtained, will continue to be provided under given conditions for a given time duration.*" For example, did the end user's service session continue to deliver acceptable service quality until session was terminated either on end user demand (e.g., one party hung-up a telephone call) or for normal service session termination (e.g., inactivity timeout)? A dropped telephone call is a well-known service retainability failure. Thus, a particular user's session was either retained with acceptable service quality until normal service termination or it was prematurely terminated for some reason, including unacceptable service quality which prompts the end user to abandon a call or streaming video when the media quality is unacceptably poor.
- **Reliability** – reliability is defined by ISO/IEC/IEEE 24765 (3.2467) as "*1. the ability of a system or component to perform its required functions under stated conditions for a specified period of time. 2. capability of the software product to maintain a specified level of performance when used under specified conditions.*" For example, the reliability of a wireless service provider to seamlessly hand off calls from one wireless basestation to another as a user moves.

[3] Per ISO/IEC/IEEE 24765, "*Although 'accessibility' typically addresses users who have disabilities, the concept is not limited to disability issues.*"

- **Latency** – the elapsed time to serve a user's request, such as the time between when an user clicks "send" or "play" on their device and when they hear ringback for a telephone call or see an image for a streaming media service. Thus, each user transaction or service request has a latency value. Typically latency measurements for successful transactions are tracked separately from latency for unsuccessful transactions so "lost" transactions (nominally "infinite" latency) can be managed as reliability concerns rather than compromising performance measurements of successful transactions. Service latency experienced by end users fundamentally has two components:
  - **Application latency** – the time it takes application software and supporting components to execute users' requests. This latency is driven by the application architecture and performance of the underlying compute, memory, storage, as-a-service platform components (e.g., database-as-a-service) and the virtual networking between those components. Pure application latency is typically measured within the data center hosting application instances to eliminate access and wide area transport latency to form the application latency measurement.
  - **Transport latency** – time consumed transporting packets between an end user's device and the data center hosting the application instance serving the end user's device. One way uplink (from end user's device to data center) and downlink (from data center to end user's device) are not necessarily the same, and will vary based on network technology (e.g., 3G wireless vs. 4G/LTE vs. fiber to the home) and other factors. Serving users from data centers that are physically closer (so there are fewer miles or kilometers of transmission facilities for packets to traverse) with fewer intermediate systems (because each router, firewall, and other intermediate system adds a bit of latency) can reduce end-to-end transport latency. Application protocol design drives the number of packets that must travel to and from the user's device to complete a user-visible transaction, and thus the number of one way transmission latency values that accrue to complete each user operation.

Note that these four fundamental service quality characteristics can overlap, such as when excessive service latency on a call setup or video playback request will cause the user to abandon the service request and thus perceive the impairment as a service accessibility failure, or when an unreliable wireless handover event appears to the end user as a service retainability failure when their call drops.

General service quality measurements are sometimes used, such as mean opinion score (MOS) which are typically expressed via the five point scale in Table 1.1. Application specific qualities are sometimes used like Audio-visual synchronization (a.k.a., "lip sync" for stream video services). Application specific service qualities are beyond the scope of this paper, but are considered at length in Bauer and Adams (2013).

As application instances often serve hundreds, thousands, or more end users, the service quality experiences by all end users are generally aggregated to statistically

Table 1.1 Mean Opinion Scores (ITU P.800)

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

characterize overall performance. Thus, accessibility, retainability, and reliability are often reported as "defects per million" (DPM), such as:

- Accessibility – X failed calls/sessions per million valid attempts
- Retainability – Y dropped calls/sessions per million (or Y' dropped calls/sessions per million minutes)
- Reliability – Z failed transactions per million valid requests

Service latency can be statistically aggregated as a cumulative distribution function (CDF). Most people find complimentary cumulative distribution functions (CCDF), like Figure 1.8, more intuitive and comprehensible than traditional CDFs.

An application's configured online capacity is primarily driven by the volume of user service that can continuously be delivered with acceptable service quality.

1.4 DEMAND, SUPPLY, CAPACITY, AND FUNGIBILITY

Figure 1.9 shows the highest-level application model: a human (or nonhuman) user consumes some service offered by an application instance. The application instance is owned and operated by an application service provider. From an economics perspective, users offer demand and application service providers offer supply; consumption

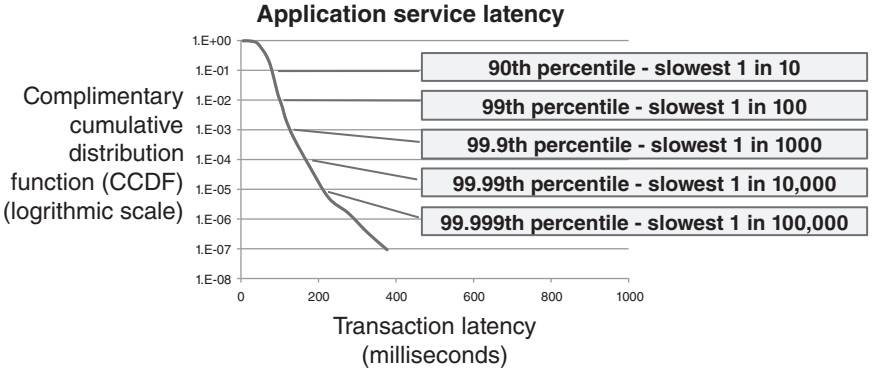

Figure 1.8 Sample Service Latency Complementary Cumulative Distribution Function (CCDF)

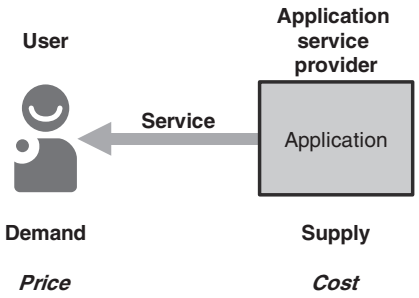

Figure 1.9 Canonical Application Service Model

is the amount of service that users enjoy in an interval of time. User (or some organization on the user's behalf) pays some price to enjoy service, and service providers must cover their costs of delivering that service.

Cloud-based applications differ from traditional applications in that software component instances execute on virtual compute, memory, storage, and networking resources that are offered by some cloud infrastructure service provider and shared by multiple application instances, as shown in Figure 1.10.

Figure 1.11 highlights a key fact: each application instance has a specific, and hence finite, configuration at each point in time, thereby offering a specific, and hence finite, application capacity at each point in time. The application's online service capacity can be changed by executing specific configuration change actions, but those actions are neither instantaneous nor flawless.

Figure 1.12 visualizes how demand, capacity, and consumption fit together. The X-axis is the instantaneous service demand, such as how many users want to watch videos on demand at a particular moment in time; the Y-axis is the number of users served at that moment in time, or instantaneous consumption. The horizontal dotted line shows the effective instantaneous supply of service capacity available to serve user demand. Instantaneous service capacity is fundamentally limited by the throughput of the underlying compute, networking, memory, and storage infrastructure configured to support the service, as well as configured limits (e.g., buffer or

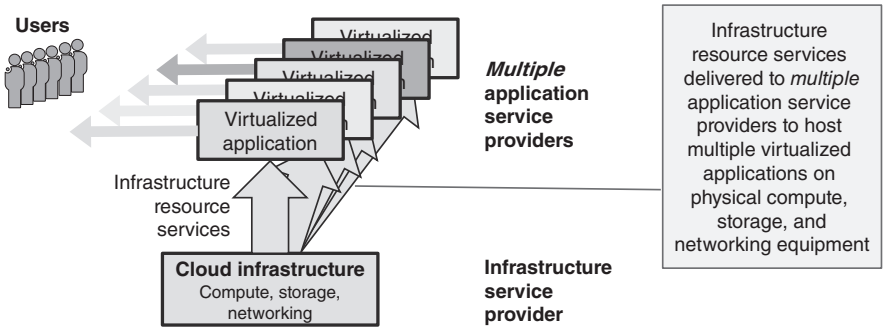

Figure 1.10 Canonical Shared Infrastructure Service Model

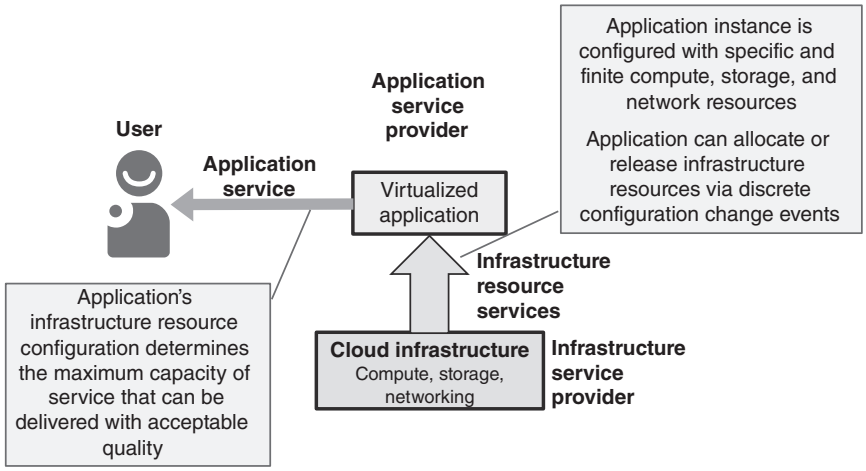

Figure 1.11 Cloud-Based Application Service Model

queue sizes, software licensing) and architectural constraints. Demand below the effective capacity limit is served with acceptable service quality; thus the solid line has a 1:1 slope (45 degrees) until service consumption reaches the limit of current online capacity. If user demand exceeds the application's online capacity, then some demand may not be served at all (because of overload or congestion controls) or it may be served with poor or unacceptable service quality because there is insufficient online resource capacity instantaneously available to the application software instance. Note that the upper right triangle of "unserved demand" of Figure 1.12 often manifests as congestion or degraded service quality when demand outstrips the physical capacity to serve the offered workload. This applies to applications just as it applies to congested roadways at rush hour.

Fungible is defined by Merriam-Webster as "*being of such a nature that one part or quantity may be replaced by another equal part or quantity in the satisfaction*

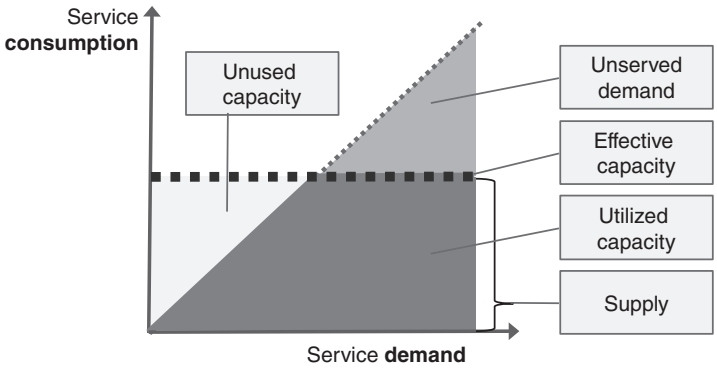

Figure 1.12 Supply, Demand, Consumption, and Capacity

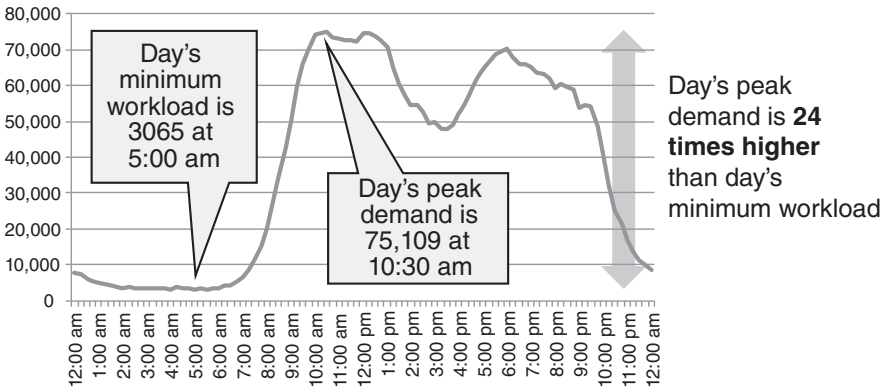

Figure 1.13 Sample Daily Application Demand Pattern

of an obligation."[4] For example, oil, wheat and lumber are fungible commodities. Individual application instances are implemented via various software components supported by various hardware components. Some components are engineered to be fungible, like workers in a load shared pool (e.g., AppLogic instances in Figure 1.6). When identical components hold no user or session state or context information, then they may be fully fungible; when user or session information is cached, additional time will be required for another component instance to load that information to fully serve users; when nonreplicated user session information is held by a component, then shifting the user to another component may require additional time or cause loss of session state and thus the user might experience loss of context which degrades their quality of experience. Sometimes nominally identical components are not fungible; for instance a particular AppLogic component executing in a data center in California, USA, is probably not be fungible with an AppLogic component executing in Bangalore, India, because end users in California would likely experience higher service latency – and thus degraded service quality – if they were served from a data center half way around the world compared to being served from a local data center. Thus, resource fungibility is an important practical consideration for configuration and capacity management.

1.5 DEMAND VARIABILITY

Humans have daily patterns, with nominally 8 hours of sleep, 8 hours of work, 4 hours of travel/transit, and 4 hours of leisure on weekdays, and hopefully more leisure time on weekends. Demand by humans for applications and other services generally have a pattern that fits into the daily pattern of their lives. Figure 1.13 illustrates a daily demand pattern for a sample application. This application exhibits clear day/night demand patterns with the workload minimum at 5 am (local time)

[4] http://www.merriam-webster.com/dictionary/fungible, retrieved January 5, 2015.

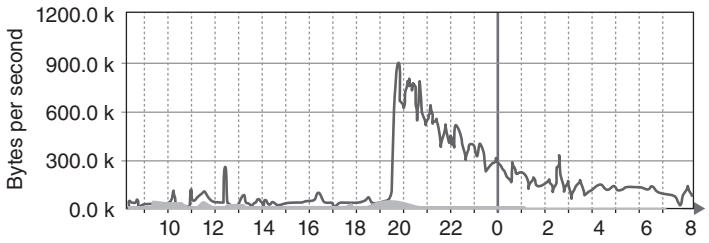

Figure 1.14 Sample Demand Shock or Slashdot Event (Courtesy Wikipedia)

and the daily workload maximum at 10:30 am; the day's maximum workload is 24 times greater than the day's minimum workload. Daily workloads often vary with the regularity of tides on the ocean. The "high" and "low" tide levels for the application routinely vary with the days of the week. The example of Figure 1.13 is from a Thursday; the typical Wednesday pattern is similar, but the typical Sunday usage pattern is materially different. Day-of-the-week variations in usage are common for many applications. Workloads often have seasonality in which the daily demand peaks may be much higher. In addition, the mean high and low tide levels may vary over time as the popularity and user base of an application grows and shrinks.

Superimposed on cyclical demand patterns like Figure 1.13 are inevitably random demand variations from minute to minute, second to second, millisecond to millisecond. Internal buffering or queuing mechanisms will smooth random demand variations on the tiniest time scales (e.g., microseconds), but the particular characteristics of application service qualities and implementation architectures will determine the shortest window of random demand variability that should be considered for the application.

Demand can surge as a positive demand shock, such as the user demand surge of the Slashdot event[5] of Figure 1.14. Demand surges can also come from maintenance actions like pushing an emergency software update, autonomous or automatic recovery actions by impacted systems, or some other synchronized or correlated trigger. The exact timing of many demand shock events cannot be predicted. Even when the time of the event can be predicted (e.g., during an entertainment event, following a commercial or promotional event), the magnitude of the demand shock is often difficult to accurately predict.

Force majeure events like an earthquake may impact the ability of some users to access a service (e.g., if their internet access connection is impacted), but myriad other users may temporarily shift their application service demands to learn, comprehend, and appropriately react to the event.

[5] "*The Slashdot effect, also known as slashdotting, occurs when a popular website links to a smaller site, causing a massive increase in traffic*," from http://en.wikipedia.org/wiki/Slashdot_effect, retrieved March 26, 2015.

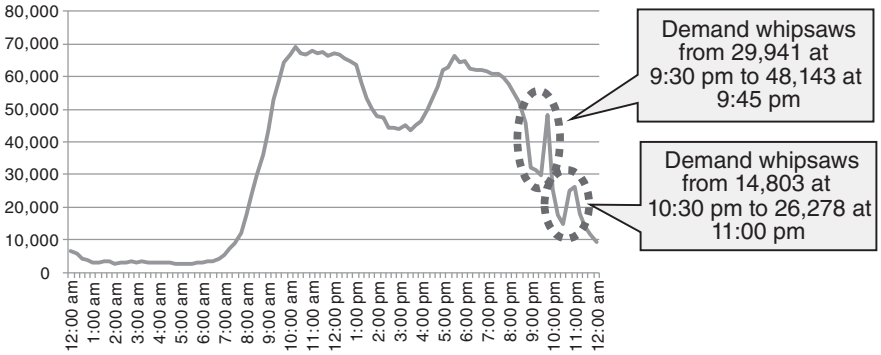

Figure 1.15 Sample of Application Demand Whipsaw

Another demand shock is demand whipsaw[6] events. Figure 1.15 illustrates two workload whipsaw incidents in a single evening; in the midst of the evening's traffic roll-off was a 60% traffic surge from 9:30 pm to 9:45 pm, followed by further traffic roll-off for 45 minutes, and then a 70% traffic surge from 10:30 pm to 11:00 pm.

1.6 CHAPTER REVIEW

- ✓ Cloud computing is a *paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand* (ISO/IEC 17788).
- ✓ CSPs offer pools of virtual resources to multiple CSCs simultaneously. These CSCs (e.g., application service providers) offer valuable services to end users via the internet.
- ✓ CSCs enjoy scalable, elastic resource capacity on-demand which is generally charged based on actual usage. CSCs can thus focus on serving their end users rather than on planning, procuring, installing, operating, maintaining, and upgrading compute, memory, storage, and networking infrastructure.
- ✓ CSPs focus on delivering high-quality virtual compute, memory, storage, and networking resource capacity on-demand and with the highest efficiency.
- ✓ Effective online capacity of an application is primarily determined by the level of user demand that can be instantaneously served with acceptable service quality.
- ✓ User demand for application service varies across time and space, so the optimal online application capacity – and associated compute, memory, storage, and networking infrastructure – will also vary across time and space.

[6] Defined as "*to beset or victimize in two opposite ways at once, by a two-phase operation, or by the collusive action of two opponents <wage earners were whipsawed by inflation and high taxes>*," http://www.merriam-webster.com/dictionary/whipsaw, retrieved January 5, 2015.