Scale-up is Dead, Long Live Scale-out!

Keith B. Evans Shadowbase Product Management, Gravic, Inc.
Paul J. Holenstein Executive Vice President, Gravic, Inc.

Introduction

A perennial problem in the IT world is how to handle the ebb and flow of user demand for IT resources. What is adequate to handle the demand at 3am on Sunday is not going to be sufficient to cope with the demand at noon on Black Friday or Cyber Monday. One way or another, IT resources must be sufficiently elastic to handle this range of user demand – they must be able to scale. If a company's IT resources are not able to sufficiently scale to keep up with demand at any given time, a service outage will most likely result, with major consequential impact to the business. As a result, the matter of IT resource scaling is of great concern for IT departments.

What is All this Scale-up/out/sideways/down Business Anyway?

How are IT processing resources scaled? The most obvious answer is to simply add more hardware to an existing system – more processors, more memory, more disk storage, more networking ports, etc. Alternatively, simply replace a system with one that is bigger and more powerful. This approach is known as scaling-UP, or vertical scaling.

The biggest problem with this approach is that of diminishing returns. Most scale-up systems use a hardware architecture known as symmetric multiprocessing (SMP). Simply put, in an SMP architecture, multiple processors share a single block of physical RAM. As more processors are added, contention for this shared memory and other shared resources becomes a significant bottleneck so that less and less actual performance benefit is realized for each processor added. Each additional processor yields less than 1x the power of that processor; the more processors, the more contention and the less incremental benefit. As more and more resources are added, eventually the system is simply unable to scale any further to meet user demand. The same restriction applies when a system is replaced; eventually there will not be a single SMP system powerful enough to meet peak capacity demands.

Besides the scalability limits, there are other issues with the scale-up approach:

High cost

- More and more hardware must be added to the system in order to meet user demand, because of the inefficiencies of scale (diminishing returns of additional processors, etc.). Or the system must be replaced with a larger, faster one. In either case, the hardware costs are significant.
- The system must be sized to serve the highest projected demand, which is a waste of expensive resources at times of lesser demand (probably the majority of time).

Large failure domain or poor fault-tolerance

- Loss of a single system will result in a service outage.
- Greater risk of outage after local incident, since systems are monolithic and cannot be geographically dispersed.
- Migration to a larger system is usually performed via the "big-bang" technique,[1] which has a high degree of risk/failure, is disruptive, and requires an outage. Such "rip and replace" migrations are difficult to test, as is fallback to an existing system.

Hardware vendor lock-in

- Additional hardware generally must come from the same vendor as existing components.
- It is costly to change vendors as the entire hardware and software stacks must be replaced, IT staff must be re-trained, etc.

Scale-out: the Elastic Solution

If vertical scaling has such significant issues, what are the alternatives? First, use a server with a different hardware architecture, a massively parallel processor (MPP). In an MPP architecture, each processor acts like a separate system, with its own memory, disks, and other hardware resources ("shared nothing"). Workload is distributed by the operating system and other software components across the processors, which communicate with each other via a high-speed message bus. Compared with an SMP, an MPP has no contention for shared resources (RAM, etc.); therefore, each processor delivers nearly 100% additional performance because MPP capacity scales linearly. The most well-known and successful MPP in the industry is the HPE NonStop server, which can scale linearly from 2-16 CPUs per system. However, what happens when a single system is not sufficient to handle the workload, or better availability is required?

Enter scale-OUT, or horizontal scaling. With scale-out, additional compute resources are provided by simply adding more servers, with the workload distributed between them (Figure 1). A scale-out architecture has the same characteristics and benefits as an MPP. In fact, an HPE NonStop server can be considered a scale-out system in a box. However, a scale-out architecture is unconstrained in terms of how many additional servers can be added, or the type of processors employed within each server (SMP or MPP). For example, a single HPE NonStop server can first scale-out by adding more CPUs, then by adding more NonStop servers to the network (up to a total of 255 servers). A scale-out architecture is able to meet much higher user demand levels than a scale-up architecture, because there essentially is no limit to the number of servers that can be incorporated.

[1] The "big-bang" technique of migration refers to the classic (and outdated) approach of requiring an outage of the primary environment in order to load, start, and cutover to the new environment. There are now newer techniques available that reduce the inherent risk of the big-bang approach, by allowing the new environment to be built, tested, validated, loaded and then synchronized before the cutover occurs. These techniques eliminate or at least dramatically reduce application outage time for the migration at substantially reduced risk. For more information, see Using HPE Shadowbase Software to Eliminate Planned Downtime via Zero Downtime Migration.

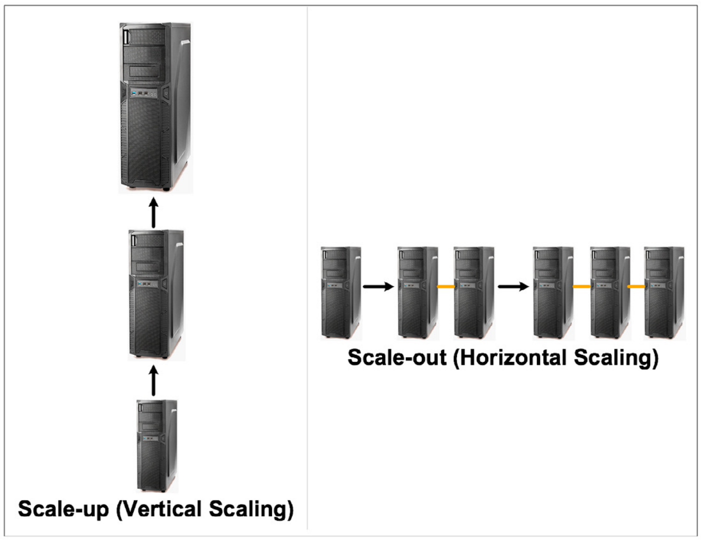

Figure 1: Scale-up vs Scale-out

Besides unlimited scalability, there are other benefits of the scale-out architecture over the scale-up architecture:

Better capacity utilization

- A scale-out configuration does not suffer the resource contention issues of an SMP; each additional processor delivers its full capacity. Hence, fewer system resources are required for a given workload than for a scale-up SMP system. A few smaller and cheaper servers can handle the same workload.
- It is easier and more cost effective to add (and remove) additional systems as load increases (or decreases). With a scale-up architecture, the extra capacity is wasted when not in use.
- Less overall capacity is lost when a failure occurs, because servers are smaller.

Lower cost

- It is incrementally much cheaper to add additional server capacity than to replace a single server with a larger, faster one; the existing hardware investment is preserved.
- Additional compute resources can be added via cloud service providers on demand as required, and released when no longer needed.

Excellent availability characteristics

- Since multiple systems are employed, the failure of any one does not result in a total service outage.
- Multiple servers can be geographically dispersed, which reduces outage risk from a localized incident.
- Zero downtime migration – hardware and software can be upgraded without service interruption and at a much lower risk. If necessary, upgrades can be incrementally performed while existing servers are maintained and leveraged as a fallback.

No hardware or software vendor lock-in

- Additional servers can be added to an existing scale-out architecture regardless of vendor.

But What About the Application?

Good question. Scale-up and scale-out architectures are very different from an application point-of-view. With a single large system (a vertical scaling model), like an SMP architecture, all applications run on that system and can access the same shared memory. This architecture tends to lead to monolithic application processes, which run multiple parallel threads and use shared memory as the primary method for sharing data and context/state between the threads/processes.

This type of application model is adequate for an SMP as the system is still able to handle user demand. But at some point, the system limits will be reached, and it will be necessary to move to a scale-out solution, migrate the application with much difficulty, and take advantage of the unlimited scalability provided. Therefore, from the outset, it is a best practice to write applications for a scale-out architecture and be prepared to scale (up or down) when needed.

Applications written for scale-out are easily spread across multiple systems, and workload can be distributed across any instance of the application process and on any system. As demand increases, it is simple to first instantiate more application processes across existing systems, and then meet demand across additional systems, if necessary.

This application scalability is primarily achieved by avoiding the use of shared resources (memory, etc.), and by not internally maintaining state (stateless servers). Ignoring either of these techniques limits the ability to distribute workload equally across all systems and application instances by forcing requests to be serviced by particular application instances/systems, which thereby limits scalability. Rather than offering all user services in a single monolithic application, scale-out applications provide them via many, smaller process instances. These instances are able to interoperate via inter-process communication (IPC), each offering a subset of the whole and grouping “like” services together (e.g., separating long-running requests from short-running requests), which enables the optimization of workload distribution, improving average response times, as well as scaling ability. Small footprint processes are also quick to spin up and down as user demand rises and falls in order to maintain desired throughput and application response times. Starting additional large processes to add capacity is not ideal if a system already is under heavy load.

The Elephant in the Room

Therefore, applications can be designed for scale-out, but there is an elephant in the room. As previously discussed, it is important that applications should be stateless and not use shared memory, but at some point, they have to access shared data. It does not significantly improve scalability/availability if workload can be distributed across multiple application server instances, yet still be forced to access a single database residing on a single server. Similarly, partitioning the data across multiple systems only provides partial relief. In order to maximize scalability/availability in a scale-out architecture, shared data must be locally available to all systems participating in the application. Each copy of the data must be kept consistent with all other copies as the data is being updated, regardless of on which system application updates are being executed. Enter real-time data replication.

With transactional real-time data replication implemented between all systems participating in the application, multiple copies of the database can be distributed across each system, which are kept consistent as data is changed on any system. This distribution optimizes scalability by, a) allowing user requests to be routed to any system based on load (the so-called “route anywhere” model), and b) by scaling the database and also the application (i.e., removing the database as a source of contention and hence a bottleneck). If any system fails, other systems have up-to-date copies of the database on which processing can continue, thereby maximizing application availability. This characteristic applies not only to unplanned outages, but also to planned system maintenance, which can be performed serially across systems so that no application outages ever need to occur. This characteristic even applies to system and software upgrades, allowing for zero downtime migrations (ZDM).

The highest levels of scalability (capacity utilization) and availability are obtained by using an active/active application architecture as described above, where user requests are distributed and executed on any system. The scale-out principle also may be applied to active/passive and sizzling-hot-takeover (SZT) configurations. In these configurations, all update transactions are executed on a single active system, but scalability can still be achieved via the use of data replication from the active system to multiple passive systems, which are then used for read-only or query type applications. A good example of such an architecture is a so-called "look-to-book" application. Multiple read-only nodes are used to look-up information (e.g., airline/hotel seat/room availability, or stock prices), while the active system is only used when an actual transaction is executed (e.g., an airline/hotel reservation, or a stock trade). It thereby offloads the active system and scales-out the workload across multiple systems without requiring the application to run fully active/active.[2]

Scale-out Example: Telco Phone Billing and Provisioning System

An example of a scale-out architecture is shown in Figure 2, demonstrating the use of both active/active systems and multiple read-only nodes to achieve continuous availability and horizontal scaling. A major international telco realized that its Home Locator Register (HLR) application could no longer support requirements to provision and manage smart phones, since the management of smart phone features is far more complex than for older, simpler cell phones. Therefore, the company implemented a new distributed active/active HPE NonStop server system to provision smart phones and to manage its more complex billing and service requirements. In order to handle the ever-increasing load, as well as the active/active pair that serve as the continuously available "master" system, multiple scale-out read-only query ("subordinate") nodes were also implemented, from which the HLRs obtain the smart phone provisioning information required to establish calls and verify/bill for services.

HPE Shadowbase technology provides the data replication infrastructure between these multiple nodes to support both the continuous availability of the active/active pair and to keep the data on the query nodes synchronized with the database of record. Both active NonStop nodes and all of the query nodes share exactly the same data. Though the master system load is relatively small, the query load is intensive as there must be an HLR query for each call being established. Since the master database is replicated to the query nodes, the master system is not burdened with query processing, and the architecture can easily scale to handle any load as the number of smart phones increases. The query nodes are distributed near population centers to improve query performance, which shortens call establishment time. In the initial deployment, the telco is using six query nodes. As activity increases, more query nodes can be easily added to scale-out the application without any interruption to existing service, which would not be possible with a scale-up architecture.

Summary

Keeping up with user demand is a significant challenge for IT departments. The traditional scale-up approach suffers from significant limitations and cost issues that prevent it from satisfying the ever-increasing workloads of a 24x7 online society. The use of MPP and scale-out architectures is the solution, since they can readily and non-disruptively apply additional compute resources to meet any demand, and at a much lower cost. The use of a data replication engine to share and maintain consistent data between multiple systems enables scale-out application and workload distributions across multiple compute nodes, which provides the necessary scalability and availability to meet the highest levels of user demand now and into the future.

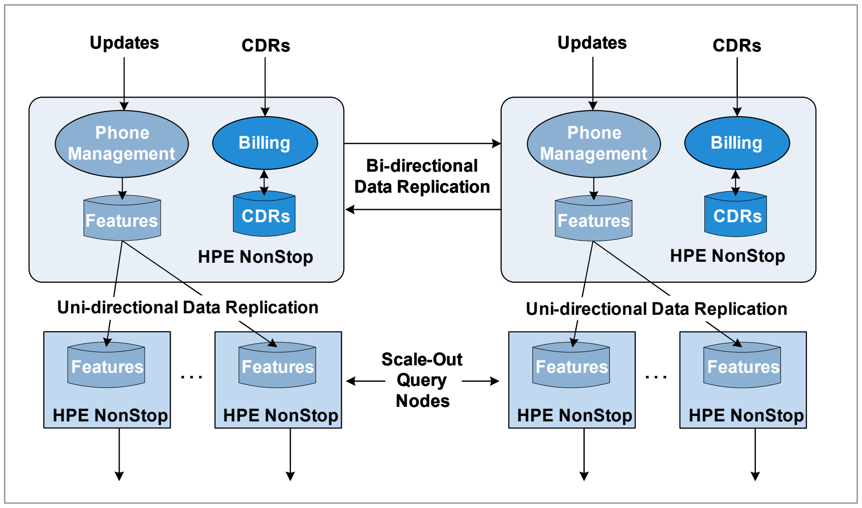

Figure 2: Telco HLR Scale-out Architecture

Keith B. Evans works in Shadowbase Product Management. Mr. Evans earned a BSc (Honors) in Combined Sciences from DeMontfort University, England. He began his professional life as a software engineer at IBM UK Laboratories, developing the CICS application server. He then moved to Digital Equipment Corporation as a pre-sales specialist. In 1988, he emigrated to the U.S. and took a position at Amdahl in Silicon Valley as a software architect, working on transaction processing middleware. In 1992, Mr. Evans joined Tandem and was the lead architect for its open TP application server program (NonStop Tuxedo). After the Tandem mergers, he became a Distinguished Technologist with HP NonStop Enterprise Division (NED) and was involved with the continuing development of middleware application infrastructures. In 2006, he moved into a Product Manager position at NED, responsible for middleware and business continuity software. Mr. Evans joined the Shadowbase Products Group in 2012, working to develop the HPE and Gravic partnership, internal processes, marketing communications, and the Shadowbase product roadmap (in response to business and customer requirements). A particular area of focus is the patented and newly released Shadowbase synchronous replication technology for zero data loss (ZDL) and data collision avoidance in active/active architectures.

Paul J. Holenstein is Executive Vice President of Gravic, Inc. He is responsible for the HPE Shadowbase suite of products. The HPE Shadowbase replication engine is a high-speed, uni-directional and bi-directional, homogeneous and heterogeneous data replication engine that provides advanced business continuity solutions as well as moves data updates between enterprise systems in fractions of a second. It also provides capabilities to integrate disparate operational application information into real-time business intelligence systems. Shadowbase Total Replication Solutions® provides products to leverage this technology with proven implementations. HPE Shadowbase software is built by Gravic, and globally sold and supported by HPE. Please contact your local HPE account team for more information, or visit https://www.ShadowbaseSoftware.com. To contact the authors, please email: SBProductManagement@gravic.com.

[2] For additional information on active/active, sizzling-hot-takeover, and active/passive business continuity architectures, see the white paper, Choosing a Business Continuity Solution to Match Your Business Availability Requirements.