Elastic Application Platforms for Cloud Computing

Introduction

Statisticians, engineers, scientists, and analysts in performance-intensive domains, such as financial services, multimedia systems, and bioinformatics, are trying to harness the processing power available in cloud computing environments to analyze financial trends, create test simulations, model climate, compile code, render video, decode genomes and other complex processing tasks. The proliferation of cloud environments during the past several years has contributed to explosive growth in management and provisioning tools that enable unobtrusive access to—and control over—elastic hardware and the migration of software applications onto this hardware. Elastic hardware can substantially accelerate the performance of applications that can distribute and/or parallelize their internal workloads on co-located or distributed cores. Elastic software can accelerate and scale up data processing automatically by utilizing more hardware without changing application business logic or configurations.

Achievements in the area of automatic provisioning of hardware and software were sufficient for the first wave of "best-effort" application deployments in cloud environments. The potential for making mission-critical applications elastic remains largely untapped, however, due to the lack of software infrastructure and tool support that can simplify and automate the following three characteristics of elastic software applications:

- They are **parallelized**, i.e., their computations can run concurrently on multiple processors and/or cores,
- They are **scaled dynamically**, i.e., the number of processors and/or cores used by an application can expand or contract during runtime, and
- They are **deployment agnostic**, i.e., they can seamlessly deploy on a single co-located multicore processor or distributed multicore processors.

Despite the price/performance advantages of clouds it remains hard to accelerate and scale the performance of applications, which are often written for a single computer and thus cannot leverage the parallelism inherent in cloud environments without extensive refactoring and reprogramming. What is needed, therefore, are *elastic application platforms* that simplify the development and deployment of ultra-high-performance elastic applications.

Elastic application platforms provide tools to help developers rapidly transform sequential applications into powerful elastic applications that can be distributed and/or parallelized on a wide range of hardware, from multicore processors to clusters to private or public clouds. These tools help eliminate tradeoffs between co-located vs. distributed parallelism patterns via common "program once, run anywhere" APIs that enable software to run in deployments where the choice of co-location vs. distribution is hidden from application software on both clients and servers. These platforms also solve distribution challenges by transparently deploying applications on co-located and/or distributed processing cores via straightforward changes to declarative configuration parameters, rather than requiring tedious and error-prone imperative programming.

The Elastic Application Software Development Process

The process of developing elastic applications—or retrofitting legacy applications to scale elastically—involves mapping application elasticity technology onto the challenges of particular business logic. This process pinpoints potential parallelism in software and identifies the best approach to exploit it on available hardware, ranging from co-located cores to interconnected clouds. This effort can be as easy as flattening existing loops or as complex as refactoring software architectures to add new abstractions that parallelize critical program logic. Enabling such adjustments requires elastic application platforms to seamlessly integrate powerful programming patterns (such as task and data parallelism) with reusable services (such as intelligent load equalizers that identify routing destination dynamically based on real-time feedback from distributed services) to shield application developers from complex concurrency and networking mechanisms, while ensuring substantially higher performance on commodity cloud environments.

Refactoring applications to run on co-located or distributed cores incur various challenges that have historically been addressed by unrelated and non-compatible APIs and tools. In particular, co-location strives to support short tasks, where potential network latency might rival the time needed to run a single task. As tasks get shorter, therefore, co-location must provide auto-batching and zero-copy techniques that balance workload among the cores. Distribution, on the other hand, makes application elastic, offering potentially unlimited scaling, while incurring inevitable latency costs, e.g., from data copy and network (de)marshaling.

There are a few tools on the market that offer application transformation toolkits, ranging from fully automatic "analysis and reconstruction" to complex SDKs that introduce completely new architectural paradigms requiring dedicated programming and domain knowledge expertise. Both ends of this spectrum suffer from substantial disadvantages in either limited end result performance or complex undertaking, often yielding unrealistic and cost prohibitive solutions for existing applications. A more effective approach, therefore, involves rearranging the workload of existing business logic so that it can be executed in parallel and then drop-in replacing it with another layer of intelligence that can distribute workload across available co-located cores and/or distributing it over the network, while providing necessary quality attributes, such as fault tolerance, automatic service discovery, smart load balancing, and request recovery. There are two steps to this approach:

- Step one preserves the application's business logic while grouping and reformatting its workload into independent steps for optimum parallelism. At this stage applications remain non-elastic, but are prepared for drop-in replacement of their workloads.
- Step two replaces workload execution in the application to another layer that looks similar, but executes the business logic outside of the application onto elastic hardware that can be replaced and managed independently and transparently to the application. This step is mostly mechanical; the business logic still remains.

This two-step approach successfully applies distribution and parallelization technology to new or existing applications with minimum invasion into business logic, leaving application developers in command of their domain expertise and intellectual property.

Evaluation Criteria

Not every application can benefit from elastic application platforms, so an appropriate assessment should answer the following questions:

- Is there independent processing inside the application that can be repeated multiple times?
- What is the data model and can the data be encapsulated into requests or treated as a cache and distributed to servers?
- What is duration of the processing that can be exported to the elastic hardware and how will the computation speedup correlate to the latency of available networks?

Answers to these questions help systems engineers, software architects, and programmers to fit elastic applications into one of following template solutions that are capable of being deployed to many cloud providers, as well as clusters, public and private data centers:

- **Collocated on an n-way multicore machine**
  - Pros: Serves best the class of applications that require substantial memory footprint, have massive parameter lists and relatively short duration of their processing.
  - Cons: Scalability is restricted by the number of available computer cores on the machine.
- **Distributed to a set of engines utilizing 1 worker thread each.**
  - Pros: A single processing thread eliminates the need of the "thread safety" for the code running remotely. A cloud will typically run as many single thread worker engines as there are accessible cores on the correspondent hardware platform.

- Cons: Increased number of engines heightens operational control and monitoring.

- **Distributed to a set of engines utilizing multiple worker threads.**
  - Pros: A single engine per hardware platform substantially reduces the operational and monitoring load. Usually these engines will utilize as many worker threads as there are accessible cores on the correspondent hardware platform.
  - Cons: The code running in multiple worker thread engine must be thread safe.
- **Mixed**.
  - Pros: The most flexible model that benefits complex environments.
  - Cons: Complex operational model.

Concluding Remarks

While application elasticity enables efficient use of hardware and massive workload parallelization, it cannot be harnessed effectively without support from the underlying elastic application platform. Elastic applications use these platforms to intelligently equalize load, identify routing destination dynamically based on real-time feedback from distributed services, recover missing requests, tolerate network or service failure, automatically discover services, and monitor the health and status of application and system services.

The next-generation of elastic application platforms are thus increasing the utilization of the previously underutilized resources and exploiting the processing power available to them, including newer technologies, such as multicore processors and cloud computing environments, as well as traditional desktops, clusters, and servers. In addition, giving developers a set of tools to quickly deploy ultra-high-performance elastic applications in cloud environments that can scale from 10's to 1,000's of processors without code revisions will be a key factor in deciding which platform to select.