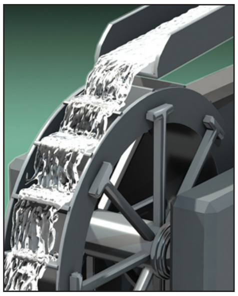

Principles of Elastic Processes

Schahram Dustdar • *Vienna University of Technology*

Yike Guo • *Imperial College London*

Benjamin Satzger and Hong-Linh Truong • *Vienna University of Technology*

Cloud computing's success has made on-demand computing with a pay-as-you-go pricing model popular. However, cloud computing's focus on resources and costs limits progress in realizing more flexible, adaptive processes. The authors introduce elastic processes, which are based on explicitly modeling resources, cost, and quality, and show how they improve on the state of the art.

Process automation and workflows are familiar concepts in modern computer science. Increasingly, data-intensive applications play a crucial role in this domain — our online and interconnected society produces massive amounts of data. Sources include sensor-equipped environments, such as smart buildings, social media, and financial markets. To harvest the valuable information hidden in these "data blobs," we can often apply the concept of processes to streamline data processing and analytical steps. Currently, we can apply such processes for both static and real-time data from different sources and deliver the analytical results within a structured enterprise computing environment. However, we argue that such a computing paradigm lacks some necessary features for modern Internet-scale information processing, where both cloud and human computing[1] are heavily employed.

Cloud computing and human computing have the following common features that we must address for process automation:

- *Dynamic resource requirement and provision*. Both cloud and human computing environments are based on the concept of provisioning adequate resources as services in a demand-driven fashion based on a price model. Such a service economy mechanism should be an integrated part of process models.[1]
- *Quality of service (QoS) within processes*. Because services realize each process in a workflow, QoS becomes an important notion for two reasons. First, when we uniformly regard computation as service, we can view a workflow as a compositional service. Thus, its quality must be well defined by the quality of its component services. Second, QoS is related to the resources services require and thus the cost of those resources.

We propose the concept of *elastic processes* (EPs), precisely defining the various facets of elasticity that capture process dynamics in cloud and human computing. The main properties for modeling EPs' economic and physical dynamics are *resource elasticity*, *cost elasticity*, and *quality elasticity* (the "Elasticity in Related Disciplines" sidebar provides the general definitions for elasticity that we consider in our work).

Elasticity captures one essence of cloud computing: when limited resources are offered for potentially unlimited use, providers must manage them elastically by scaling up and down, as needed. However, as is common today, understanding and supporting elasticity purely from

1089-7801/11/$26.00 © 2011 IEEE

a resource-management viewpoint is rather restrictive. Resources' requirements aren't determined only by the application using them. If we really treat computation as a service, then we must consider all aspects of a service that might impact the demands on a resource.

Elasticity in Related Disciplines

In computer science, the term *elastic computing* has recently been used as the academic synonym of *cloud computing*, thanks to Amazon's premier cloud service offering, the Elastic Compute Cloud (EC2).

The current Wikipedia definition of elasticity in physics states that "elasticity is the physical property of a material when it deforms under stress (for example, external forces) but returns to its original shape when the stress is removed. The relative amount of deformation is called the strain." When applied to computing, elasticity naturally reflects the on-demand nature of cloud service provisioning: it states that the amount of resources an application uses or a provider offers can expand or contract based on influences such as demand.

Another related definition of elasticity is found in economics, which describes it as "the ratio of the percent change in one variable to the percent change in another variable."[1] That is, elasticity measures a function's responsiveness or sensitivity to changes in parameters in a relative way. In general, the formula for the elasticity of *Y* with respect to *X* is

$$e(Y,X)=\frac{dy}{dx}\frac{X}{Y},$$

where e(*Y*, *X*) is short for "the elasticity of *Y* with respect to *X*," and *dY*/*dX* is the derivative of *Y* with respect to *X*. In economics, elasticity is an effective way to measure demand and supply responsiveness. This notion of elasticity should be adequate to apply to the resource, quality, and cost dynamics in service-oriented computing, especially in the context of cloud computing.

Reference

1. E. Dowling, *Introduction to Mathematical Economics*, 3rd ed., McGraw-Hill, 1980.

The proposed EP is a novel concept that significantly enriches computational processes' properties in the context of cloud computing and service-oriented computing in general. Existing workflows are limited to resource elasticity by adjusting machine power, while cost and quality are barely considered. However, these three main properties are interdependent, and we must study them based on a uniform foundation. Our aim is to build a proper modeling, reasoning, and execution framework in which we can specify and monitor these properties to build a quantifiable, proactive, and predictive resource-capacity-management system for Internet-scale process automation that integrates multiple clouds and various forms of human computing.

Elasticity Properties

We've identified elasticity considering resources, cost, and quality as crucial for future processes in the context of service-based computing. Let's look more closely at cost and quality elasticity, which are discussed much more rarely than is resource elasticity.

Cost Elasticity

Cost elasticity describes a resource provision's responsiveness to changes in cost. Service providers apply it when defining price models for cloud computing systems. In this context, cost elasticity is also referred to as *utility computing*, in which resources such as computational services provided by virtual machines, data transmission on the network, and storage services provided on different storage hierarchies are charged based on a pay-as-you-go pricing mechanism. In defining a price model for utility computing, the cost incurred to support the computing capacity level is the baseline for the design. These cost items include the investment, provisioning, and maintenance of processor, memory, hard disk, and network with, respectively, desired clock frequency, memory size, size of disk space used, and data transmission cost. Based on these factors, providers can develop dynamic pricing models based on the cost elasticity concept. Taking Amazon as an example, the following price models are based on cost elasticity estimation:

- *On-demand instances* are a pure pay per use-on-demand model, in which customers don't have long-term commitments and are free from planning.
- *Spot instances* occur when spot prices fluctuate over time according to supply-demand status and other factors Amazon considers. Users bid a maximum price they're willing to pay for these instances and run them as long as the spot price $\leq$ bidding price, until the instance is explicitly terminated, or the price rises above users' bidding price.

With the spot price option, Amazon can use higher spot prices during peak times and lower prices during off-peak times to shape customer behaviors such that flexible users would tend to consume more during off-peak times and avoid purchases during peak times. This would flatten aggregate usage over time, which, in turn, would decrease Amazon's maintenance costs. In this sense, price is intuitively

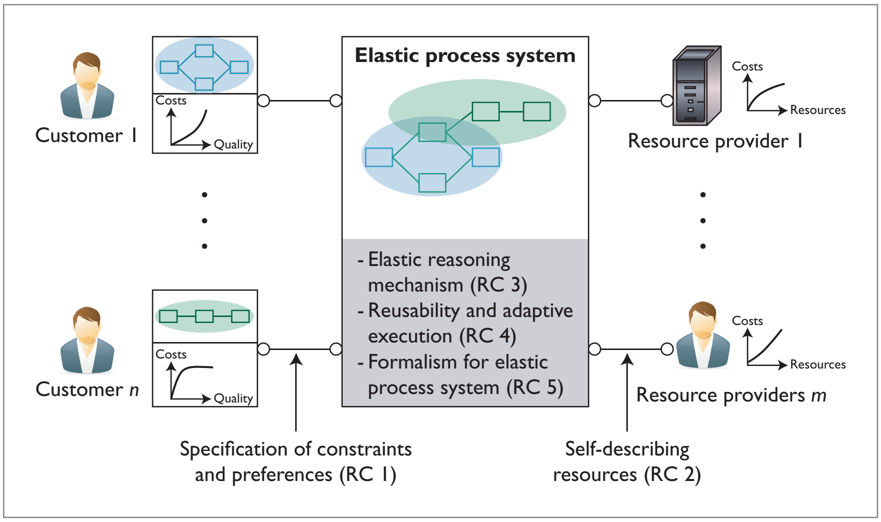
Customer 1
Costs
Quality
Customer *n*
Costs
Quality
Elastic process system
- Elastic reasoning mechanism (RC 3)
- Reusability and adaptive execution (RC 4)
- Formalism for elastic process system (RC 5)
Costs
Resources
Resource provider 1
Costs
Resources
Resource providers *m*
Specification of constraints and preferences (RC 1)
Self-describing resources (RC 2)

Figure 1. Conceptual architecture of elastic process environment. We can see the five main research challenges (RCs) in designing an elastic process system.

controlled not only by cost elasticity but also by the incentive effect on customers.

Quality Elasticity

Quality elasticity measures how responsive quality is to a change in resource usage. The elasticity comes from a feature inherent to cloud applications — that is, to have a well-defined quality elasticity measurement, an application service's underlying algorithm requires that the service's quality improvement be monotonic to the consumption of the resource needed. In other words, the more resources consumed, the better the achievable quality. The main issue here is to associate a service with a measurable quality and the cost function, which computes the resource requirement for a given quality, such as execution speed. In this case, a service's result is deterministic, but its execution speed is scaled based on the required resource. In cloud computing, some computational forms have this desired property. For example, MapReduce is a scalable programming framework that lets users process data elastically.[2] It has a desired quality elasticity that states that execution speed is scalable to the increase of servers in a distributed file system.

Response time isn't the only quality criteria used. Other quality measurements such as the result quality in an approximation-based computing process can help provide a new class of cloud algorithms. The Aqua approximate query answering system developed at Bell Labs is an example of a system that makes trade-offs considering quality aspects in query processing.[3] Traditional query processing focuses on generating exact answers. However, when huge data stores are involved, providing an exact result might take an unacceptably long time. In many cases, exact answers aren't required, and approximate or quick results are preferred. Aqua is a system for quickly executing queries by providing approximate answers tailored to data warehousing environments. When we couple such an approximation process with a monotonic resource consumption model, we can build an elastic querying system based on the notion of quality elasticity. Recent research in data space as an approximation-based type of search computing is an important attempt toward an elastic search paradigm.[4]

Conceptual Model

To realize EPs, we propose a conceptual architecture of an EP environment, as Figure 1 illustrates. We identified five primary research challenges that informed our model's design, and discuss these in detail later. First, let's look at EPs' physical and economic properties.

Physical Elasticity Properties

An EP must decide how to use existing resources in its environment in an optimal way (one that can meet multidimensional demands but with a maximum benefit). The EP environment is dynamic, with diverse resource types (computational, data, and network resources). These resources are also dynamic, as are their quality and cost models. Based on quality and cost, an EP might use different sets of resources as well as its processing activities to produce multiple outputs. On the other hand, some demands might have similar requirements, so the same resources and processing elements in the EP can produce multiple outputs. Such behaviors reflect an EP's internal physical elasticity properties.

Economic Elasticity Properties

First, let's distinguish between an EP and resources for building EPs, which can be any kind of machine or human computation and network resource; machine computation can come from (virtual) computational machines or software services atop machines. Providers make resources available, and each resource has certain properties, such as quality and cost. An EP's *function* (for example, translation) is a static property that accepts certain input data sources and produces some results. The function is modeled and implemented as a set of interdependent activities. It's built from existing components but differently than are static processes.

As with its physical elasticity properties, an EPs' economic elasticity properties include resource, cost, and quality elasticity. An EP uses resources provisioned by any provider at any place and used at any time, as long as their capabilities meet the constraints the processes require, such as minimum spending costs. Essentially, resource elasticity is an internal property that isn't exposed to consumers. For quality elasticity, however, an EP can offer different models, which are accessible to the users. They depend on functions, costs, and resources used. Similarly, an EP considers different cost models and presents those models to consumers.

Partially Elastic Processes

Systems considering quality or cost when deciding on resource usage are not novel. The novelty is in explicitly modeling quality, cost, and resources allowing for reasoning and making trade-offs. We call processes considering only parts of these aspects "partially elastic processes." One example can be found in the integration of machine and human capabilities for processing. Recently, we've moved from pure machine computation processes (such as traditional, compute-intensive workflows) to a combination of machine and human computation. We've seen that people and software services can participate in processes to perform certain tasks, such as image evaluation. Given that people have heterogeneous skills and interests, human processing systems start to explicitly consider quality for "resource allocation" — that is, for assigning a task to a suitable worker. This can lead to results that meet predefined quality requirements.[1]

A further example of partially elastic processes can be found in data analysis in sustainable facilities and smart cities. Current facility-management techniques have enabled sensor infrastructures that can collect different types of facility information. Furthermore, data resources available on the Internet, such as weather information and maps, can be combined with facility data to support complex data analysis processes. In sensor networks, energy awareness is an essential property, and indeed a large body of research on energy-efficient sensor networks exists, mostly with a focus on routing, but also on energy-aware resource allocation for process-oriented tasks.[2] Because energy consumption generates costs, this can be seen as a partially elastic process as defined previously.

References

1. B. Satzger et al., "Stimulating Skill Evolution in Market-Based Crowdsourcing," *Proc. 9th Int'l Conf. Business Process Management* (BPM 11), to appear, 2011.
2. K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," *Ad Hoc Networks*, vol. 3, no. 3, 2005, pp. 325–349.

Operation and Modeling Principles

In our view, an EP's basic operation principles are its ability to monitor, manage, and describe dynamic properties; the dynamic refinement of process functions based on quality (that is, new functions such as data enrichment or data cleaning can be added to improve quality); the ability to determine cost based on multiple resource cost models; and the ability to provide elasticity across providers – that is, an EP could spread and combine components from different providers, as long as it satisfies its requirements. Ultimately, an EP can deal with multiple service objectives. In the simplest case, the EP would serve one consumer (as with an analysis of Facebook activities) and utilize one provider (such as Amazon). In the most extreme case, an EP will have N concurrent consumers and access to a market of M providers. N consumers would give K requirements (input data, cost, quality), and $K \leq N$. So, EPs must be able to deal with trade-offs between requirements.

EPs have several properties that enable them to compose modeling principles, including overlaying EPs, function composition, and dynamic property composition. We can outline modeling principles as follows. An EP must model its function as a static property. The EP's results are based on requirements concerning cost and quality, modeled as a set of constraints; this model influences the resource elasticity. Furthermore, modeling can also describe how an EP can communicate with other EPs. This communication can be based on the abstraction of a service interface such as REST or SOAP. We can apply the refinement and composition of the EP's resource, cost, and quality to different levels – activities within an EP, fragments within an EP, and the whole EP – and also apply the different operation and modeling principles at these levels.

Research Challenges

Existing solutions haven't been able to deal with all the properties we've mentioned (the "Partially Elastic Processes" sidebar provides examples for existing solutions). To build real systems with these properties, we must address several research challenges for interfaces between EPs, consumer demands and environments, and elastic properties.

Specification of Constraints and Preferences

Compared to traditional process execution, elasticity requires giving more autonomy to the infrastructure and the processes themselves. Each process consumer or user who wants to utilize the EP system (EPS) defines a process enriched with constraints and preferences specifying cost and quality trade-offs. The EPS takes this tuple and will eventually present the result to the user. However, users must still be able to control the system behavior with simple and intuitive interfaces. They need a means to express their constraints and preferences in a human-centered way. They should make statements about cost and quality rather than resources. Intuitive human-centered models need a mechanism for translation into computer-readable formats and vice versa if the system is to interact with users about constraints and preferences (for instance, by recommending removing a constraint, resulting in high costs and low quality gains).

Self-Describing Resources

For the actual processing, the EPS maps parts of the processes onto resources (machines or humans), taking into account the specified requirements. Thus, it must know about available resources' existence and capabilities. To that end, resources must provide a description containing information about their availability and corresponding costs.

The challenge here is that we envision EPs "living" in heterogeneous environments with different hardware resources, load characteristics, administration, ownership, laws, and privacy policies. Each resource must deal with this degree of heterogeneity to describe itself. Different levels of detail are possible, and some information will be optional, but the description should be comprehensible to anyone.

To improve scalability, we propose a hierarchical description methodology: a cloud could, for instance, have its own description that's an aggregation of the "sub-cloud" description, which, in turn, comprises numerous single machines, each with its own description, too. Resources might also be humans (or social compute units[1]), whose description might be based on a skill profile, track record, or whether the human is available to process some task.

Elastic Reasoning Mechanism

With multidimensional dynamic demands, an EP must be equipped with an *elastic reasoning mechanism* (ERM) to decide how to utilize resources in an optimal way. We can regard an ERM as an optimization system that takes dynamic resource and cost information from the environment to maintain a cloud's dynamically generated capacity and price information (computational, data, and network resources). Such an environment is usually available as part of a cloud management platform, such as Eucalyptus.[5]

Reusability and Adaptive Execution

Executing processes in an elastic way, in compliance with user-defined constraints and preferences, can be highly challenging. While several related works on adaptive process execution exist, they generally don't consider combined resources, costs, and quality. Existing refinement techniques for process structures, for instance, focus on performance-related quality (such as service availability) but not on result quality (better images). Runtime refinements are basic — for instance, component replacement — while complex refinements such as fragment replacement are supported only in offline (not continuous and elastic) processes. To achieve a trade-off between these aspects in a large-scale heterogeneous environment requires additional research efforts.

Because the environments we're considering are highly dynamic, process execution can't be sluggish or even static. It must focus on continuous monitoring and re-planning. In such large, complex environments, exact algorithms drop out, but approximate decision approaches based on heuristics and partial information are needed. Techniques such as prediction, optimization, auctions, and virtual markets are candidate ingredients for the final adaptive execution recipe.

The EPS allows for adaptive process execution and can react to changes in the environment and partially merge processes for optimized execution. In Figure 1, for instance, the blue and green processes share a common computation, which we can reuse for efficient execution.

Formalism for Elastic Process Systems

A formal system for studying elastic computing can contribute to modeling and understanding EPs. As in any process calculus, such a system must be built on a well-defined set of operators over processes. Different from traditional communicating process calculi, the system's operators should mainly focus on modeling processes' elastic features and their composition.

We've identified cost and quality as main facets to consider for process execution. We argue that future processes should be able to take a description of quality and cost requirements. The execution environment needs the intelligence to determine the actual resource usage based on that description. This leads to elastic processes.

References

1. S. Dustdar and K. Bhattacharya, "The Social Compute Unit," *IEEE Internet Computing*, vol. 15, no. 3, 2011, pp. 64–69.
2. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Comm. ACM*, vol. 51, no. 1, 2008, pp. 107–113; http://doi.acm.org/10.1145/1327452.1327492.
3. S. Acharya et al., "The Aqua Approximate Query Answering System," *Proc. ACM SIGMOD Int'l Conf. Management*

of Data (SIGMOD 99), ACM Press, 1999, pp. 574–576; http://doi.acm.org/10.1145/304182.304581.

4. K. Belhajjame et al., "Feedback-Based Annotation, Selection, and Refinement of Schema Mappings for Dataspaces," *Proc. 13th Int'l Conf. Extending Database Technology*, ACM Press, 2010, pp. 573–584.
5. D. Nurmi et al., "The Eucalyptus Open-Source Cloud-Computing System," *Proc. 9th IEEE/ACM Int'l Symp. Cluster Computing and the Grid* (CCGRID 09), IEEE CS Press, 2009, pp. 124–131; http://dx.doi.org/10.1109/CCGRID.2009.93.

Schahram Dustdar is a full professor of computer science (informatics) with a focus on Internet technologies and heads the Distributed Systems Group, Institute of Information Systems, at the *Vienna University of Technology* (TU Wien). Dustdar is an ACM Distinguished Scientist. Contact him at dustdar@infosys.tuwien.ac.at; www.infosys.tuwien.ac.at/.

Yike Guo is a computing science professor in the Department of Computing, *Imperial College London*. His research is in large-scale scientific data analysis, data mining algorithms and applications, parallel algorithms, and cloud computing. Contact him at yg@doc.ic.ac.uk; www.doc.ic.ac.uk/~yg/.

Benjamin Satzger is an assistant professor of computer science in the Distributed Systems Group, Institute of Information Systems, at TU Wien. Contact him at satzger@infosys.tuwien.ac.at; www.infosys.tuwien.ac.at/staff/bsatzger/.

Hong-Linh Truong is a post-doctoral scientist in the Distributed Systems Group, Institute of Information Systems, at TU Wien. Contact him at truong@infosys.tuwien.ac.at; www.infosys.tuwien.ac.at/staff/truong/.