Data Markets in the Cloud: An Opportunity for the Database Community

Magdalena Balazinska, Bill Howe, and Dan Suciu
University of Washington, Seattle, USA
{magda,billhowe,suciu}@cs.washington.edu

ABSTRACT

Cloud-computing is transforming many aspects of data management. Most recently, the cloud is seeing the emergence of digital markets for data and associated services. We observe that our community has a lot to offer in building successful cloud-based data markets. We outline some of the key challenges that such markets face and discuss the associated research problems that our community can help solve.

1. INTRODUCTION

New types of data markets are emerging. Facilitated by cloud-computing, these data markets offer a convenient single, logically centralized point for buying and selling data [4, 12]. Close behind are data "after markets", enabled by value-added services that derive *data products* (visualizations [16], dashboards [19]). These markets, however, are still in their infancy. The economic and algorithmic principles guiding the pricing of data, data products, and the services that deliver them are largely unexplored. Existing pricing frameworks are simplistic and can exhibit unexpected and undesirable properties leading to, for example, arbitrage situations, fairness violations, and unpredictability. Further, the technology to facilitate these cloud-based data markets and enforce pricing policies is underdeveloped.

There are two types of challenges in building a successful cloud-based data market. One is related to the behavior of agents (sellers and buyers) and the rules for successfully selling and buying data. This challenge belongs to our colleagues in economics departments. There is, however, a second challenge related to (1) deeply understanding how the value of data is modified during data transformations, integration, and usage, and (2) developing pricing models, supporting tools, and services for facilitating a cloud-based data market. This second challenge is of the competence of the database community and is the challenge that we discuss in this paper. Our conjecture is that the lessons of data modeling, management, and query processing developed by the database community over the last 40 years are necessary and sufficient for overcoming this challenge.

It is important for the database community to be involved because a cloud-based data market can have a significant economic effect by incentivizing investment in high-risk research and development. Indeed, such investments frequently produce valuable information, but less frequently proven, tangible products. A cloud-based data market facilitates monetization of such experimental data, benefiting academic research and encouraging federal research funding.

A cloud-based data market can also democratize and streamline the existing unmanaged data market. Most data products today are purchased through offline negotiations between providers and consumers, with only a small fraction of data being sold online (*e.g.*, [2, 7]). A cloud-based data market provides access to "one stop shopping" for companies, end-users, and application developers. Systems such as Google Fusion Tables [9] and Many Eyes [16] have demonstrated that ordinary users can take advantage of accessing, correlating, and analyzing each other's data. A cloud-based data market can help these users find and acquire data. It can also simplify the creation of value-added services by application developers. Consider the market for weather forecast data products — all such websites use the same handful of sources for weather forecast simulations, yet collectively constitute a $1.5 billion industry [14].

In this paper, we set a research agenda for the data management community to help in the creation of a successful cloud-based data market.

2. MOTIVATION AND RELATED WORK

There already exist independent vendors selling data online [2, 7]. Similarly, Amazon cloud users can already sell their S3 data for a profit [3]. These autonomous vendor approaches give most flexibility to sellers in terms of pricing models, but an organized market can facilitate data discovery and the logistics of selling and buying data.

Multiple digital markets for data have recently emerged in the cloud [4, 12]. These data markets enable content providers to upload their data and make it available either freely or for a fee through a query interface. In the case of Infochimps [12], the cloud provider sets prices: data consumers pay monthly subscriptions that enable a maximum number of queries (*i.e.*, API calls) per month. Alternatively, data providers set prices for users to download entire datasets. The Azure DataMarket [4] also uses data subscriptions with query limits: *i.e.*, a group of records returned by a query and that can fit on a page (currently 100) is called a *trans-*

Proceedings of the VLDB Endowment, Vol. 4, No. 12
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

action and each subscription is associated with a maximum number of transactions per month. In this market, however, content providers set the prices and query limits for subscriptions. In both cases, the cloud provider takes a percent cut of the content provider's income.

There are several weaknesses with these existing market pricing models:

First, per-query (or per-transaction) costs are irrelevant when charging for the data itself; this model provides no easy method to purchase data updates and forces data consumers to cache the purchased data as they will be charged repeatedly for accessing the same data.

Second, the pricing model can inadvertently lead to arbitrage situations, where using multiple smaller subscriptions through different accounts is less expensive than using one larger account. A class project at the University of Washington [1] found multiple arbitrage opportunities in the current offerings in the Azure Data Market.

Third, the model assumes all tuples are of equal value. When this assumption does not hold, savvy users can game the query interface to earn more value than "fair" users.

Fourth, a provider can emulate different prices by partitioning one logical dataset into multiple physical datasets, each with different pricing tiers. However, data providers have no principled way to set the pricing tiers, and the systems provides no guidance.

Today's markets' pricing models are thus inflexible, hard to use correctly, and have undesirable properties. Existing data markets also lack services such as an advisor for setting prices, a market-enabled query processing engine that could correlate independently-owned datasets, and more.

While the interaction between data management and economics has been studied in the database research community in occasional papers [8, 18], our community is currently not involved in the creation of cloud-based data markets.

There is a rich literature on pricing information products [13, 17], which will certainly guide the pricing of data in the cloud, and is complementary to our goal of providing data management tools in support of a cloud data market.

3. MODELING DATA MARKETS

A variety of different candidate data models and pricing models exist that may be suitable for reasoning about a data market. Consider the following:

Simple Digital media is often sold under a model where the market consists of a set of independent digital artifacts D, where prices are assigned by a pricing function $f : D \rightarrow \mathbb{R}$. This model is used, for example, by iTunes or the app stores offered by mobile phone providers. This model does not distinguish derived data products from "raw" data, which prevents the assertion of properties about their relative value.

Derived Data Product To capture the dependency relationship between digital artifacts, we can adopt a model (D, S, f), where D is a set of digital artifacts, S is a set of services $D \times D \times \cdots \times D \rightarrow D$, and f is a pricing function $D \rightarrow \mathbb{R}$. With this model, we might assert the condition that buying "raw materials" is always less expensive than buying the derived data product — that services always add value. That is, given $s \in S$, if $z = s(x_0, x_1, ..., x_n)$, then $f(z) > f(x_0) + f(x_1) + \cdots + f(x_n)$. With this model, we can, for example, express pricing schemes that recover the cost of the computing resources required to execute the service (*e.g.*, using Amazon EC2). We can also express the value-added visualizations and graphics used by the weather forecasting industry. In that case, the raw data is free, but the derived data products sometimes require a cost. This model, however, includes no concept of time, and cannot therefore express subscription-based pricing.

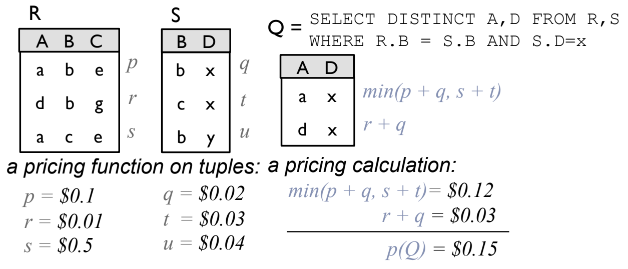

Figure 1: Query price derivation for a tuple-based pricing scheme using the pricing semiring formalism.

Subscription Users purchase N digital artifacts (tuples, blog posts, updates, images) per unit time. The number N may be infinite to model unlimited access (*e.g.*, the Hulu Plus television service [11]). The time period might typically be a month, but might be as short as minutes to model cloud computing services (*e.g.* CloudSigma [6]). The Azure Data Market uses this model with the extension of tiered pricing — different values of N result in different prices per unit.

Relational If the digital artifacts are tuples arranged into relations, we might wish to reason about the prices of query results given the prices of the base data. No current data markets offer such a capability.

4. RESEARCH OPPORTUNITIES

We present some of the key challenges for the database community related to a relational cloud data market.

4.1 Enabling Fine-grained Data Pricing

An initial challenge for the database community is to derive a pricing model that can capture all of the above models. By doing so, one gives content providers maximum flexibility to explore pricing strategies for the cloud data market. The goal is not to impose how pricing should be done but rather to develop tools that will help content providers price their data in a principled fashion with properties and side-effects that can be verified. For example, consider buying a subscription to a real estate map showing commercial properties that were renovated. The price could reflect the visualization service that created the map (Derived Data Product model), the value-added from joining disparate sources to create the map (Relational model), and a monthly fee to access the site (Subscription model).

One possible pricing model is for the content provider to specify four items: (1) the *structural granularity* at which prices are attached to data (cell, tuple, relation, schema, collection), (2) a *base data pricing function* assigning a price to each element at the chosen granularity in the base data, (3) a *derived data pricing method* specifying how query result prices are calculated from base data and propagated through query operations, and, optionally, (4) a *subscription model* specifying how updates to query results are priced (*e.g.*, does the user receive free continuous updates or not).

Based on this pricing scheme, the system will objectively compute the value of derived data. A *data consumer* reads data by issuing SQL queries, and is charged for the result.

Figure 1 illustrates an example pricing calculation (we describe the figure further below).

For example, a microbiologist charges for access to her genomic data for a particular organism. The value of this data corresponds to the value of the genes found within it, so the provider sets prices at the structural granularity of the tuple, and assigns a pricing function that determines price based on the existence of the sequence in a public database. Tuples not present in the database command a higher price, corresponding to their higher utility. The subscription model is one-time access, meaning that the charges are incurred on a per-query basis. The derived pricing method assigns a fixed cost to all COUNT results, and an average cost for all other aggregates. Further, use of a custom sequence similarity function incurs a charge for each invocation, since it incorporates IP-protected knowledge about the quality of the reads produced by the sequencer.

We still must consider how to define the pricing functions. We assume each tuple is assigned an individual price. The task is to automatically derive the price of any query result.

Pricing as provenance One approach is to use *lineage* (i.e., *provenance*) of data to compute the price of each output tuple. First, we model the price of a base tuple as a provenance annotation. Then, we can compute the price of the result of each operator: The join of two tuples is the sum of their two prices, and we take the minimum price during duplicate elimination. For example, the price of each item returned by the query `select distinct lon, lat from sensor` is the smallest price of all `sensor` records at a given longitude and latitude. This pricing scheme is captured elegantly in the framework of *provenance semirings* [10]: the *price semiring* is $(\mathbf{R}^+, \min, +, \infty, 0)$. Other pricing semirings are possible. An example price calculation using the pricing semiring is illustrated in Figure 1.

Submodular pricing The semiring formalism does not capture all pricing functions we would like to support. One example is a *submodular* pricing function: Given two output results D_1 and D_2, a function price : $D \rightarrow \mathcal{R}$ is submodular if when $D_1 \subseteq D_2$ then $\text{price}(D_1 \cup \{t\}) - \text{price}(D_1) \geq \text{price}(D_2 \cup \{t\}) - \text{price}(D_2)$.

In summary, several options are possible for Fine-grained Data Pricing that require a systematic investigation by the data management community.

4.2 Making Fine-Grained Pricing Efficient

Supporting the above model in the cloud raises multiple systems challenges. The basic challenge is how to efficiently compute the price of a query result. One approach is to compute the provenance expressions for all result tuples. The benefit of this approach is that the system can combine these expressions with additional pricing functions if needed before computing the final query result price. For example, an additional pricing function could give a discount on the base data prices when a query touches more than a certain number of base tuples. A challenge, however, is to achieve high efficiency as the provenance expressions grow. An alternate, more efficient approach, is to develop an extended relational algebra that is price-aware. With this approach, all base tuples are first annotated with their individual per-tuple prices. Second, each operator combines the price of its input tuples to derive the price of the corresponding output tuples. For example, using the pricing semiring above, a join operator adds the prices of the two joined input tuples to produce the price of the output tuple. With this extended relational algebra approach, we are incrementally evaluating the provenance expression for each result tuple. This second approach, however, works well only when prices are set at the granularity of tuples or cells and does not allow for complex pricing schemes such as submodular pricing.

4.3 Reasoning about Pricing Properties

Another contribution that the database community can make is to study the properties of various pricing models such as *efficiency* and *fairness* [15] in the context of a relational data market. Consider fairness. Economically, the fair value of a product is the amount at which it could be bought or sold in a current transaction between willing parties, or transferred to an equivalent party, other than in a liquidation sale.[1] A fair pricing model should price all query outputs at their fair values, but designing a fair pricing scheme for data is far from trivial: *e.g.*, if an owner sells her data in bulk, but the buyer can resell it item by item at a higher profit, then the prices are unfair, and will destabilize the market in the long term. Even the current pricing of Web services (in particular Amazon's pricing scheme) has been shown in a recent study to be unfair [20]. The database community can help in studying the inter-play of pricing and query processing: perhaps the base-data prices do not allow for arbitrage but the latter can occur with savvy purchase of specific query outputs.

4.4 Making Pricing Models Usable

Models that are general and expressive may have good market properties but they may be difficult to understand. Complex pricing models negatively impact content providers who must figure out how to price their data using these models and consumers who want to purchase that data. There is evidence that simple pricing models are more attractive to buyers (*e.g.*, the iTunes flat price per song model).

We identify two properties that pricing models must have in order to be *usable*. Other properties may also be needed The first property is *comprehensibility*: a content provider must understand the income that he is making from selling his data and a data consumer must understand the payments she is asked to make. To ensure comprehensibility, just as cell phone companies send users detailed listings of the calls they made and text messages they sent, the cloud provider will have to send detailed usage reports to both data consumers and providers. Data lineage is the natural answer to explain the price paid for each query result tuple. The challenge will be in summarizing and presenting lineage information to non-database experts in the context of possibly large numbers of data purchase operations.

The second property is *predictability.* Content providers want the ability to estimate monthly incomes. Content consumers want to estimate monthly charges. Predictability is at odds with fine-grained pricing: A flat subscription rate (*e.g.*, unlimited calling for \$25/month) leads to much more predictable charges than a pay-as-you-go (*e.g.*, \$0.01 per minute) approach. Unpredictability can be especially high in a data market where consumers are not only charged for the number of records that they purchase but for the record values and how these values were derived. One approach to addressing this challenge is to develop *price-tuning advisors* similar in spirit to physical-tuning advisors [5]. For a data

[1]`http://en.wikipedia.org/wiki/Fair_value`

provider, a price-tuning advisor can help determine what pricing model to adopt to maximize profit given a workload description in the form of queries and associated expected frequencies. The tuning advisor can also produce income estimates. For a consumer, a price-tuning advisor can compute estimated charges based on given data pricing models and a predicted workload on that data. In case of parameterized queries, the advisor could provide error bars to capture price uncertainties due to different query parameter values.

Another usability challenge is to help a content provider specify the functions for pricing base and derived data. One approach to facilitate this task is a statistical one. Let the content provider specify a number of concrete query templates and the prices of their results, and have the system automatically compute the cost of other query outputs: *e.g.*, the owner specifies \$0.50 for the output to a query template `select * from OceanProbe where depth = $d` and another price \$0.15 for `select * from OceanProbe where long = $lon and lat = $lat`. If the user issues a query that matches these templates, then the system will charge accordingly. If the user issues a different query, say `select * from OceanProbe where month = 'May'`, then the system will have to compute the price through *interpolation*. One approach is to reverse-engineer base-data prices and the functions for computing derived data prices from the query templates (through curve fitting), and use these reverse-engineered prices to price other query results.

4.5 Building Data Market Services

A data market will require a battery of services to support content providers and consumers. Due to space constraints, we only sketch four such services here as examples:

A price-aware query optimizer. As the number of content providers and the overlap in available data will grow, *price-aware* query optimization will become possible. Unlike the Mariposa [18] system, which put a price on resources and query latencies, here the price is on the data, which raises several new challenges. First, query prices need to be estimated with the potential for errors. These estimates may involve manipulating complex lineage expressions, yet they must be performed fast when they are on the critical path of a query optimizer. An interesting related challenge will be to study the *robustness* of different pricing schemes to estimation errors and the possibility to associate error bars to capture price uncertainty for different query results.

A property checker. As content providers price their base and derived data products, they may benefit from tools that can check important properties of their pricing scheme, such as whether the scheme is arbitrage-free or not.

A price-tuning advisor. As described above.

Data transforming services. In addition to data, users may be interested in purchasing services that transform the data in various ways: *e.g.*, a content provider could use a *data anonymization* service before making her data available. A data consumer could use a *data cleaning* service after purchasing and integrating data from multiple sources. In general, many data management services that our community has already developed or has the ability to develop can be useful in the context of cloud-based data markets.

4.6 Additional Challenges

There are other challenges that we do not have space to discuss. For instance, while a cloud-based data market is well-suited for selling data *en masse*, there will always be a need for negotiating custom data products. How can such private negotiations co-exist with the data market? As developers build new services (*e.g.*, data anonymization services), what pricing models are most appropriate for such services? If content providers elect to use simple, coarse-grained pricing models (in the spirit of the iTunes pricing model), can our price-tuning advisor help convert fine-grained prices with good efficiency, fairness, and other properties into coarser-grained ones with similar properties? Can we develop techniques to account for properties such as cleanliness or degree of anonymization when pricing data?

5. CONCLUSION

We discussed the implications of the emerging cloud-based data markets on the database research community. Our community has a great opportunity in making a significant impact on these data markets, while solving exciting data management research challenges.

6. ACKNOWLEDGMENTS

This work is partially supported by NSF and Microsoft through NSF grant CCF-1047815.

7. REFERENCES

[1] Cse544 - Principles of Database Systems - Winter 2011 projects. `http://www.cs.washington.edu/education/courses/cse544/11wi/project.html%`.

[2] Aggdata. `http://www.aggdata.com/`.

[3] Using Amazon S3 Requester Pays with DevPay. `http://docs.amazonwebservices.com/AmazonDevPay/latest/DevPayDeveloperGu%ide/index.html?S3RequesterPays.html`.

[4] Azure data market. `https://datamarket.azure.com/`.

[5] S. Chaudhuri and V. R. Narasayya. Self-tuning database systems: A decade of progress. In *Proc. of the 33rd VLDB Conf.*, pages 3–14, 2007.

[6] CloudSigma. `http://www.cloudsigma.com/`.

[7] CustomLists.net. `http://www.customlists.net/home`.

[8] D. Dash, V. Kantere, and A. Ailamaki. An economic model for self-tuned cloud caching. In *Proc. of the 25th ICDE Conf.*, pages 1687–1693, 2009.

[9] Google Fusion Tables. `http://tables.googlelabs.com/public/tour/tour1.html`.

[10] T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[11] Hulu plus. `http://www.hulu.com/plus?a`.

[12] Infochimps. `http://www.infochimps.com/`.

[13] S. Jain and P. K. Kannan. Pricing of information products on online servers: Issues, models, and analysis. *Management Science*, 48(9):1123–1142, 2002.

[14] C. Johnson. How did weather data get opened? `http://infovegan.com/2010/08/09/how-did-weather-data-get-opened`.

[15] N. Mankiw. *Principles of economics*. South-Western Pub, 2008.

[16] Many eyes. `http://www-958.ibm.com/software/data/cognos/manyeyes/`. `http://services.alphaworks.ibm.com/manyeyes/home`.

[17] C. Shapiro and H. R. Varian. Versioning: The smart way to sell information. *Harvard Business Review*, 1998.

[18] Stonebraker et al. Mariposa: a wide-area distributed database system. *VLDB Journal*, 5(1):048–063, 1996.

[19] Tableau. `http://www.tableausoftware.com/`.

[20] Wang et al. Distributed systems meet economics: Pricing in the cloud. In *Proc. of HotCloud'10*, pages 6–6, 2010.