Maximizing Efficiency By Trading Storage for Computation[†]

Ian F. Adams Darrell D. E. Long Ethan L. Miller
University of California, Santa Cruz
Shankar Pasupathy Mark W. Storer
NetApp *Pergamum Systems*

Abstract

Traditionally, computing has meant calculating results and then storing those results for later use. Unfortunately, committing large volumes of rarely used data to storage wastes space and energy, making it a very expensive strategy. Cloud computing, with its readily available and flexibly allocatable computing resources, suggests an alternative: storing the provenance data, and means to recomputing results as needed.

While computation and storage are equivalent, finding the balance between the two that maximizes efficiency is difficult. One of the fundamental challenges of this issue is rooted in the knowledge gap separating the users and the cloud administrators—neither has a completely informed view. Users have a semantic understanding of their data, while administrators have an understanding of the cloud's underlying structure. We detail the user knowledge and system knowledge needed to construct a comprehensive cost model for analyzing the trade-off between storing a result and regenerating a result, allowing users and administrators to make an informed cost-benefit analysis.

1 Introduction

In traditional computing, storage is used to hold the results of computation. In this simple model where the final computational state is preserved, results are simply read from storage each time they are needed. Cloud computing, with its promise of readily available, flexibly allocated computational resources, calls into question the relationship between processing and storage. Instead of storing a result, it may be more efficient to store the provenance data and inputs to a process along with the means to recalculate a result if it is ever needed again.

For example, climate models use relatively small inputs such as sensor data and historic records, but generate enormous amounts of output data. When the model becomes obsolete, these old results are often accessed very infrequently. As an alternative to using costly storage for such rarely used data, it may be cheaper to store the provenance information and recomputation means needed to regenerate the data. In this way, storage costs are reduced, while still preserving access to the result should the need arise. A more common application of this tradeoff is already realized in video-on-demand services. Rather than store every possible resolution and format, video providers store only the highest-resolution video and utilize a cloud-based service to transcode on-demand to the desired output format [13].

Recomputation as a replacement for storage fits well into the holistic model of computing described by the cloud architecture [3]. With its dynamically scalable, and virtualized architecture, cloud computing aims to abstract away the details of underlying infrastructure. In both *public* and *private* clouds, the user is encouraged to think in terms of services, not structure.

While the decision between storage and computing superficially appears to be a simple cost-benefit tradeoff, there are several issues to consider. First, we must examine what can actually be stored and or feasibly computed. Is the goal to recompute an exact result, or merely an acceptable one? Are there legal requirements? Security issues? Second, a system that aims to enable recomputation will need additional metadata and provenance facilities in order to ensure that re-computation methods are known, and that result regenerations are successful. Third, we must understand the factors that determine when it is more efficient to recompute a re-

[†]Supported in part by the Petascale Data Storage Institute under Department of Energy award DE-FC02-06ER25768 and by the industrial sponsors of the Storage Systems Research Center.

sult, as opposed to storing it, such as the likelihood of reuse and the potential penalties if the data is unavailable when needed. Because this includes both user and system knowledge, understanding these factors is an important step in ensuring that the decision maker has all of the information required to make an informed decision.

Though complicated, the decision is in essence a comparison of two costs: The cost of storage (C_s) and the cost of recomputation (C_r).

$$C_s = \textit{cost per bit} \times \textit{bits stored}$$
$$C_r = (\textit{computation cost} + \textit{miss penalty}) \times \textit{likelihood of reuse}$$

The cost of storage—assuming a linear cost model—is straightforward; it is simply the cost per bit multiplied by the number stored and the duration they must be stored for. The cost of recomputation is more complex as it involves more factors, such as the cost of adapting a process to a new cloud provider. Similarly, the miss penalty may be difficult to estimate accurately, and is highly application-specific. Finally, the likelihood of reuse can be very difficult to determine, yet its value is perhaps the most critical to determining the expected cost to recompute a result.

As an example, consider an organization using Amazon's AWS cloud services at current prices that *might* need access to a 5 TB data set over 10 years. If the data is unavailable when needed they lose $20,000. To store for 10 years will cost $90,000 and regenerating the data requires 5 days and 100 machines costing a total of $5,000. With these numbers alone it is unclear whether it is better to store persistently, or recompute at a later date. With recomputation, even a 50% chance per year of reuse may yield net savings provided there is sufficient lead time to regenerate results, while a lower chance of reuse may still dictate we store persistently if there insufficient time to regenerate the data before its needed. Furthermore, market factors may significantly raise or lower costs in the long term. These are just a few of the factors that should be examined when considering such a tradeoff.

This leads to a significantly more complicated cost-benefit analysis than would be assumed at first glance. The rest of the discussion revolves around issues that are relevant to understanding the tradeoff between storage and computation. First, we examine the requirements for storing the components needed to recompute results. We next discuss the factors involved in a cost-benefit model comparing storage with computation. We end with related works discussing other views of costs in cloud computing, as well potentially beneficial technologies that could aid in our cost-benefit analysis.

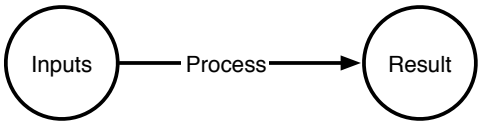


Figure 1: The three components involved with generating computed data: one or more *inputs*, a *process* that transforms those inputs, and the ensuing *result*.

2 Discussion

The basic relationship governing results computation can be expressed with a simple, three entity model. As Figure 1 illustrates, the computation is rooted in one or more *inputs*, which can, in turn, be the result of previous computation. These inputs are acted upon, and transformed by a *process*. The output of that process is a *result*.

In a traditional model of computing, this result is stored for future use. In contrast, cloud computing's holistic view of storage and computation is well suited to intelligently choosing which results to store and which to recompute as needed. The efficiency gains of this approach can be likened to file compression, which trades some computation costs for storage efficiency.

2.1 Requirements

As a first step in determining when it is desirable to recompute results as opposed to storing them, it is important to understand the conditions that make recomputation possible. A primary concern is the integrity constraint of the result. If there is a *strict* integrity constraint, then the goal is to always regenerate the same result. Some results, however, may only carry *loose* integrity constraints, in which a merely "acceptable" result is required. For example, a simulation process might generate different output each time it is run, but each of the different results is admissible.

If inputs are being stored with an eye towards regenerating results, then the corresponding process is also required. This is especially true for results with a strict integrity constraint; for loose integrity results, an alternate, slightly different, process might suffice.

If the process is to be stored and reused, there are a number of requirements which must be met. First, the process must be known, and it must be describable in some manner that renders it reusable. Second, for strict integrity, the process must be deterministic. This does not, however, preclude the storage of pseudo-random processes, so long as all the necessary input seed values have also been stored. Third, the process must be re-executable over the desired lifetime of the result, a par-

Result Specific	Odds and frequency of reuse Reuse lead time Rebuild time Miss penalty
Marginal Costs	Storage Network Computation
Trends	Technology trends Market trends Volatility and forecast confidence

Table 1: Summation of factors to include in a cost benefit model covering the computation and storage tradeoff.

ticularly difficult challenge for long-term scenarios [6]. Cloud computing increases the challenge, since different clouds may have different computation models, and even a single cloud's engine may change over time. For example, Amazon's EC2 service offers root access on machines, while Google's App Engine imposes a custom version of Python, and database restrictions [15]. Finally, even if the result is not stored, but rather recalculated, there is pertinent metadata that should still be considered for storage. In a strict integrity scenario, a hash of the result is useful for confirming successful computation. Additionally, a measure of time or resources required to rebuild can assist in scheduling reconstruction.

2.2 Cost Analysis Model

Once it has been determined that an end product can be recomputed, a cost model is invaluable in choosing the most efficient strategy for dealing with computed results. Such a cost model has three primary facets, summarized in Table 1. First, there are several *result specific* aspects to consider. Second, *marginal costs* describe how much an additional unit of cloud service would cost at the present moment. Third, the cost *trends* attempt to predict where the costs might be at a point in the future.

One of the key challenges in balancing storage and computation is deciding who makes the decision regarding what is to be stored and what is to be recomputed; both end user and system knowledge are required for an informed decision. For example, if the cloud provider is making the decision in order to maximize their system efficiency, they may not fully understand the miss penalty associated with a time sensitive result. In contrast, a user or application-centric decision may lack significant infrastructure knowledge. To this end, a fully informed decision will involve knowledge from both parties. The answer may lie in the form of user generated metadata associated with a result, and used in conjunction with a cloud provider's cost model.

2.2.1 Result Specific Issues

There are a number of factors, intrinsic to the result itself, that must be taken into account when choosing wether to store or recompute. These result-specific issues can be divided into *low-level*, and *high-level* factors. Low-level factors describe a system level view of how results are created, while high-level factors are based on the the meaning of the result, and are often highly subjective.

Provenance aware systems track a number of low-level factors by constructing dependency graphs to capture the relationship between inputs, processes and results. These graphs can be extended to record factors such as resource requirements and hashes of computed results. By recording resource requirements, it is possible to estimate how much lead time is required to recalculate a result. Hashes can be used to confirm reconstruction. As dependency chains grow longer, such data is increasingly important; the longer the chain of data products that must be regenerated, the greater the inclination to store at least some of the key intermediate values.

There are a number of high-level factors that require either an application or user level understanding of a result. First, the likelihood and frequency that a result will be used at a later time is highly variable. Files are not accessed uniformly; some files are very "hot", though the vast majority of storage is rarely accessed [9]. Second, there is a potential concern over the time needed to recompute results—while retrieved results can be accessed nearly immediately, recomputing results can be time-intensive, particularly if inputs earlier in the dependency graph must be recomputed as well. Moreover, computation time can be greatly effected by the amount of parallelism available in the cloud and the degree to which the process can be parallelized to take advantage of it. The miss penalty is also a critical factor: does it incur a small financial cost or is the miss penalty very high? If miss penalties are high, a strategy that provides the lowest possible access time will likely be optimal.

Miss penalties are especially important to consider, since overbooking is a standard practice in many service-oriented businesses. One way that miss penalties could be mitigated is with the use of insurance: a service contract with a cloud provider could include a standard SLA (Service Level Agreement), detailing assured levels of service, along with an insurance clause offering monetary compensation to the user if the provider fails to provide service up to the SLA. Cloud providers like Amazon's EC2 [1] and RackSpace [11] already provide basic

SLA's to compensate users for lost cycles and downtime. However, this must be balanced against the miss penalty from the user's perspective; a small monetary concession may be sufficient compensation for a few missed frames in a video playback, but not for a loss of life.

Finally, different data has different security needs. Storing the information needed to regenerate a result may introduce new avenues for malicious exploits. While a cloud administrator can measure the risks associated with different strategies, they cannot always measure the impact of a security breach without an understanding of the result itself. Similarly, while public clouds are often sold as abstract computing resources, seemingly banal details—such as where the infrastructure is located, and who owns the hardware—can have a direct impact on legal jurisdiction and liability [5]. Currently, some cloud computing solutions, such as Amazon's Web Service, explicitly state in their user policies that, while they will try to keep data reliably accessible and secure, they are not responsible for leaked or lost data, and it is the user's responsibility to provide adequate security and backup [2].

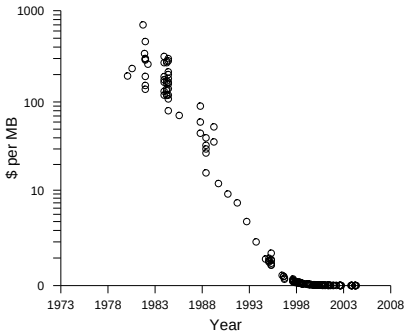
2.2.2 Marginal Costs

There are three primary marginal utility costs to consider when maximizing efficiency: storage, computation and network costs. Each measures the cost of one additional unit of service. In anything other than a simple linear cost model where X costs Y and $10X$ costs $10Y$, marginal costs become the relevant figure. While underlying costs such as energy and floor space could be specifically itemized, we assume that the marginal costs of storage, computation and transport can be formulated to encompass both capital and operating expenditures.

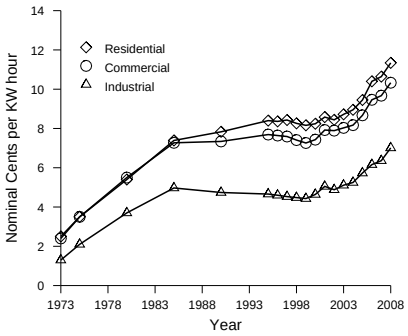
The first of the three marginal costs is *storage*, which, in a cloud environment, can be measured in dollars per unit per year. The second direct cost, *network*, describes the costs to transport and move data within the cloud. This may be measured using both the amount of data moved, as well as the speed at which the data is moved. The third cost, *computation*, which may be highly dependent on the process itself. For example, depending on how computation is billed, the parallelism of the process may affect the cost to the user.

2.2.3 Forecasts

While marginal costs describe the financial state of the cloud at the present, a number of indirect factors describe potential future state. For example, advances in storage capacity and processing capacity occur at different rates; Kryder's law states that hard drive areal density doubles annually [14], while Moore's law states that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years. Further, market volatility can also play into a cost efficiency model, making it difficult to predict the future state of a volatile market. As Figure 2 shows, while the price of energy has trended upwards, it has not followed as predictable a growth rate as hard drive costs [4]. Similarly, while DRAM prices have trended lower, market forces such as supply and demand, and extra-market forces such as collusion and price fixing, conspire against prediction confidence [10]. Furthering the difficulty of prediction are technological plateaus and disruptive technologies.



(a) Price per megabyte of commercially available hard drives.



(b) Nominal price of energy for three markets.

Figure 2: Predicting costs can be difficult; some factors such as price per megabyte of hard drive storage are fairly predictable, others such as energy prices are more volatile.

3 Related Work

The equivalence of storage and computation is well established, and several applications utilize storage as a replacement for repeated computation. For example, dynamic programming techniques exploit storage to solve

problems that exhibit overlapping subproblems. More concretely, some cryptanalysis techniques utilize storage by precomputing and storing tables to aid in key recovery [8]. The larger the table size, the less time is spent on computation during the key search. Cloud computing allows such a space-time trade off to be done on a larger scale, and as such demands a unique analysis to understand its potential benefits and detriments.

There have already been a few general analyses of the potential applications, concerns and costs in cloud computing. Balani *et al.* [3] examine many of the issues in cloud computing. In contrast to our discussion, however, their focus is primarily on the direct costs and challenges of computation in a cloud—such as adaptation of programs to the cloud —and they do not examine the potential of trading computation for storage.

Gray examined economic issues [7], but focused more on the physical transmission and coordination costs involved in computation. Gray looked at cost and argued that "...computations should be near the data", and stating that in computation as a utility, the key factor is the amount of processing done per bit. Again, however, the potential tradeoffs in storage and computation are not explicitly examined.

Storage systems with an awareness of how data is generated, such as Provenance Aware Storage Systems (PASS) [12], provide a number of low level facilities needed to trade computation for storage space. PASS preserves the provenance of data by recording the inputs, commands, and processes used to create a file. By extending this information with additional metadata—such as a hash of the result, and the time and resources needed to regenerate the result—systems such as PASS can provide much of the underlying support for analyzing the feasibility and cost of re-computation.

4 Conclusions

The presence of readily available, and easily allocated computational utility promised by cloud computing calls into the question the traditional role of storage as a way to preserve past computations. Instead of storing a result, it may be most cost efficient to store the inputs, processes and provenance data needed to regenerate a result, and then regenerate results on-demand. In essence, computation can be used in place of storage. If data is unlikely to be reused, as is often the case, this approach may yield significant cost savings.

Deciding when to store a result or when to rely instead on computation comes down to a cost-benefit analysis. We have discussed the constraints and requirements for storing the input, the process, and the results. Furthermore, we presented the factors involved in a cost model covering three key areas. First, the semantic meaning of the data, such as miss penalties, and the odds and frequency of reuse. Second, marginal costs describing the costs for additional units of cloud utility. Third, forecasting to predict where prices will be in the future. These factors span the knowledge of both users, and cloud administrators, motivating the need for methods of interaction between the two. By combining information from both users and cloud administrators, users can make informed decisions between storage and recomputation to minimize costs.

References

[1] Amazon. Amazon EC2 service level agreement. `http://aws.amazon.com/ec2-sla`, Oct. 2008.

[2] Amazon. AWS customer agreement. `http://aws.amazon.com/agreement/`, Apr. 2009.

[3] R. Balani *et al*. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, UCB, Feb. 2009.

[4] Energy Information Administration. Monthly energy review January 2009. `http://www.eia.doe.gov/emeu/mer/contents.html`, Jan. 2009.

[5] Fort Worth Star-Telegram. FBI raids Dallas computer firm. *`http://www.star-telegram.com`*, Apr. 2009.

[6] H. M. Gladney and R. A. Lorie. Trustworthy 100-year digital objects: Durable encoding for when it's too late to ask. *ACM Transactions on Information Systems*, July 2005.

[7] J. Gray. Distributed computing economics. *ACM Queue*, May 2008.

[8] M. Hellman. A cryptanalytic time-memory trade-off. In *IEEE Transactions on Information Theory*, volume 26, July 1980.

[9] A. W. Leung *et al*. Measurement and analysis of large-scale network file system workloads. In *USENIX 2008 Technical Conference*.

[10] S. K. Moore. Price fixing in the memory market. *IEEE Spectrum*, Dec. 2004.

[11] Mosso. Rackspace SLA. `http://www.mosso.com/downloads/sla_cloud_servers.pdf`, Mar. 2009.

[12] K.-K. Muniswamy-Reddy *et al*. Provenance-aware storage systems. In *USENIX 2006 Technical Conference*.

[13] L. Rao. HD cloud puts video formatting in the cloud. `http://www.techcrunch.com/2009/04/14/hd-cloud-puts-video-formatting-in-the-cloud/`, Apr. 2009.

[14] C. Walter. Kryder's law. *Scientific American*, July 2005.

[15] P. Wayner. Cloud versus cloud: A guided tour of Amazon, Google, AppNexus, and GoGrid. `http://www.infoworld.com/print/37122`, July 2008.