

An economical model for storage management

Andrii Baranovski¹ on behalf of the CEDPS data group
¹Fermi National Accelerator Laboratory

Abstract

A corner stone requirement for services deployed as part of the computing grid is support for functions that allow sharing of the service resource in a manageable way. There has been a lot of progress made towards that goal in area of computer CPU scheduling (job brokering, pilot job factories, job forwarding, etc) yet data storage aspects have been largely omitted. With the exceptions of space reservation, to date, we don't have means to plan requests to data storage in a predictable and transparent manner. This document attempts to fill that gap.

Introduction

As grid paradigm has shifted high performance computing to implementations which use globally distributed "shared" resources, the limitation pertaining to a mere availability of CPU cycles became less critical. At the same time, requirements to storage systems that participate in data exchange between distributed computing centers grew in aspects related to performance and service quality. Those are the aspects important for data analysis or otherwise data intensive jobs. Thus in the setting where data is a prerequisite to computation, it is fundamental to be able to co allocate different instances of storage resources participating in globally distributed computing request.

Historically though the development requirements for storage systems have evolved the concept into somewhat of a black box. Today, storage software black box typically implements interfaces allowing retrieval of data stored or declared to it at some point in past. We think that the "black box" storage environment is simply not sufficient to implement the complexity of coordination between data and execution workflows required for optimum utilization of affected computing resources.

Data placement challenge

One of the challenges in planning data access and movement using storage as "black box" is lack of sufficient and credible information about storage state ahead of the time. The uncertainty with storage state is a direct result of private cost model in effect and unpredictable utilization of storage service in future. For example, without understanding storage cost model, clients risk incurring large wait time penalties in cases where requests hit data with large variance in cost of access. Particular cases of storage implementations where that variance is common are implementations of short term cache over long term tape storage system. Random influx of spontaneous user activity is the other example where it may difficult to set projections on future use of the storage. Under the above circumstances, the service time for any requests may be difficult to estimate at the moment data scheduling decision should be made.

In this document we attempt to introduce a degree of freedom – an interface that will allow clients manage expectation of storage service quality as well as leverage its internal cost model. We will attempt to evaluate whether the presented approach allows co-scheduling algorithms that reduce variance in waiting times for process or data placement job. We will also study the impact of accuracy in storage accounting on variance value. It

is believed that this above is important for opportunistic storage users who do not have other non automated means to pre-establish desired level of service.

Contract aware storage

Most grid storage systems are implemented by combining esoteric management policies over disk, network, tape and rarely CPU resources. These policies define internal storage cost model and deliver storage semantics to user through usually minimalistic file system like interface. The details of the implementation are not always known or can be counted on for the purpose of independent and efficient inter storage data placement. We claim that obscurity in storage resource management policies is one of the limiting factors for storage clients to efficiently co schedule use of different systems that are not or cannot share the same internal context.

That's said, the main idea behind this paper is to formalize presentation of basic resources managed by a storage system and build a model of how storage state and its resources management policies can be made more available for advanced planning by the data placement programs. In order to establish that, we introduce concepts of resource "Contract", resource exchange, and contract aware storage.

"Contract" (reservation unit,lot) is a description of primitive resource that storage service agrees to allocate to contract holder in a particular context of storage interface invocation. The description of the resource must include quantity of that resource, its type, and time duration for which resource should be allocated. Depending on the immediate need in the respective service dimension, client can request one or several contracts at a time in a transaction like manner. Storage, in turn, is responsible for accounting and enforcing all contract agreements. In doing so, thus, it may "default" the agreement or limit resource usage if client exceeded contract allocated quota. Within allocated limit, contract offers to a client exclusive and real time opportunity to access service resource.

Within this definition, we use notion of allocation to underscore two major requirements. First, it is resource accountability. The accountability requirement implies the resource been quantifiable by some numerical metric which can be maintained (updated) in a reasonable manner during the life time of the storage service. Second, allocated resource can be dispensed in a controllable fashion to requests that by their nature must utilize that resource in order to complete.

For example: even in presence of free capacity, rate of the data transfer for each contract aware request must be moderated to ensure that all cumulative transfers referring the same CPU(or equivalently bandwidth) contract do not exceed specified value. Similar logic applies to metadata lookups. The rate of requests to external name space maintaining database system must be moderated with respect to other activities performed on behalf of the same metadata access rate contract.

Given the above, not only storage should understand how its resources are used but it also must apply resource usage accounting and utilization enforcement to all of its contract aware interface invocations. In this sense, the accountability requirement of the contract is similar to service billing. The fundamental difference however is in that the contact always refers to external resources managed by the storage system. The cost of accessing

such resources is usually fixed and is invariant to implementation of storage management layer. That is the pivotal point of this proposal which allows a client transparent access to cost model of actual resources involved in the process of data storage or retrieval.

Some similarities however do help storage systems which already support billing capabilities become more accommodative to the new semantics.

Storage will periodically evaluate its state, known performance limitations and resource properties in order to export list of open (available) contracts. Storage will also maintain accounting of all contracts that are been fulfilled. In our model, number of open contracts defines storage commitment to provide service in the respective area. In other words, the number and size of all contracts in selected resource domain must reflect the cost of utilization of that resource. Such cost model can be leveraged when a contract is “bought” by the client. This action will automatically reduce storage commitment for other clients for the chosen resource. Because number of contracts at any time is fixed and publicly available, the paradigm will allow any client (or broker on its behalf) to make judgments on storage utilization at any time. Finally, since the contract transfer does not commit actual resource, the capability to inspect storage state and lock the share of the storage resource must allow for greater flexibility for algorithms that plan co-use of multiple storage deployments in a less resource wasteful way. That is fundamental difference of the contract based workflow from schedule-ahead data movement (or “pre-staging”).

For simplicity we suggest contracts on the most basics resources used in known SS implementations: CPU/bandwidth, rate of metadata requests, and storage space.

Resource exchange/broker

Resource exchange is a module responsible for “acquiring” and then transferring service contracts to clients. This module maps client requests to storage into a sequence (or just one) of contract acquisitions necessary to fulfill user’s call to storage in an optimal way. This principal feature allows independent implementations of different and possible VO specific models of how to queue and prioritize user workflows given common set of contractual constraints published by storage service. Storage side, on the other hand, may be free to implement any internal queuing and prioritization policies as long as the final outcome falls into contract agreement within which requests were submitted.

A simple model of the resource exchange may support limited number different types of contracts and storage calls. The primary role of such model would be to differentiate between uses of CPU, Network or disk IO based on their role in the context of a particular request to storage system.

For example: client requests to calculate CRC on a file will not utilize network capacity but will require substantial CPU and disk IO throughput. In such case, resource exchange may want to wait for availability of CPU and disk contracts before letting users dispatch their requests to storage to compute file checksum. Other examples may include: large files write/read or frequent access to small files. The former may need contracts on bandwidth and disk space while latter requires contracts on rate of bandwidth and metadata lookup, etc. In all cases, resource exchange will request set of contracts which it considers necessary to accomplish planned tasks. Upon success of that phase, clients

should then proceed with using storage service utilizing list of contracts acquired by the exchange.

The utilitarian purpose of the resource exchange is to reduce complicity of clients which would otherwise need to manage individual contract resources on a fine grain level. In this setting the exchange is clearly an optional component that defines VOs policy on how storage resources should be utilized by VO members. In the most trivial cases however the direct use of contracts can also be helpful when establishing simple long term service periods for routine yet important and time constrained computing activities (i.e. computing production).

Simulation/Field test work (todo)

In contrast to averaging common waiting time metric, we define service quality through waiting time “*variance*”. This metric estimates the unavoidable risk of user not getting the data within expected time frame. Our goal is to show that use of contract based reservation semantics improves the parameter value allowing for more predictable and thus efficient data scheduling.

Our principal assumption is “substantial” variability in use of storage resources. The extent of that variability must be such that it can adversely affect existing long term data placement algorithms.

The details assumptions list is the following:

- Network and disk IO are limited (for simplicity we may pair the two constraints into one).
- jobs are scheduled randomly
- data is requested randomly from 2 independent storage instances.
- data is moved to a common local location
- local cost of access is free.
- storage space is infinite.

Interface

interface Contract:

ResourceType getType()

TimeDiff getDuration()

URL getDestination()

String getId()

Float getQuantity()

interface ContractAwareStorage:

Contract []

listAvailableContracts(ResourceType type,URL destination) throws Exception

void acquire(Contract []contracts, TimeDiff duration) throws Exception

void release(Contract contract)

Adendum

