Scalable, Reliable Marshalling and Organization of Distributed Large Scale Data Onto Enterprise Storage Environments[*]

Joesph JaJa
joseph@umiacs.umd.edu

Mike Smorul
toaster@umiacs.umd.edu

Fritz McCall
fmccall@umiacs.umd.edu

Yang Wang
wpwy@umiacs.umd.edu

Institute for Advanced Computer Studies
University of Maryland, College Park

Abstract

Emerging technologies in high speed NAS, hierarchical storage management systems, and networked systems that virtualize interconnected storage over IP and fiber-channel networks, promise to consolidate distributed data stores onto large-scale professionally managed enterprise storage environments. We describe the software architecture of the PAWN (Producer – Archive Workflow Network) environment that enables scalable, reliable marshalling and organization of distributed data into such enterprise storage environments. PAWN was initially developed to capture the core elements required for long term preservation of digital objects as identified by researchers in the digital library and archiving communities. In this paper, we show how PAWN can be extended to enable multiple clients at a number of distributed sites to prepare, organize, and bulk transfer large scale data onto clusters of servers that securely verify the integrity of the data, register the metadata, and store the data into an enterprise storage environment. PAWN allows detailed description, auditing, and organization of the data, and hence will allow for efficient management, access, and disaster recovery. The basic software components are based on open standards and web technologies, and hence are platform independent.

1. Introduction

We are pursuing, in collaboration with the San Diego Supercomputer Center (SDSC) and the National Archives and Records Administration (NARA), a broad research program to address major components required to build a computing infrastructure for enabling long term archiving and preservation of digital assets. It is well known that digital preservation is substantially more challenging than the traditional problem of archiving and preserving physical objects. This collaboration has resulted in the establishment of a pilot persistent archive currently consisting of three node servers at SDSC, University of Maryland, and NARA, linked through the SDSC Storage Request Broker (SRB) data grid, and managing several terabytes of significant NARA selected collections. We have outlined in [1] the software architecture of the Producer – Archive Workflow Network (PAWN), which provides secure ingestion of digital objects into the persistent archive. PAWN captures the essential features of the producer-archive interface methodology articulated in [2], which covers the first stage of the Ingest Process as defined by the Open Archival Information System (OAIS) reference model [3]. We make use of METS (Metadata Encoding and Transmission Standard) schema [4] to encapsulate content, structural, descriptive, and preservation metadata, leading to the specification of the digital object model as described in the OAIS model.

In this paper, we show how the PAWN architecture can be extended to a general-purpose scalable software that can efficiently marshal and organize large scale distributed data into emerging mass storage systems. PAWN will allow efficient staging, arrangement, and assembly of the data at various sites, followed by parallel bulk transfers into receiving servers whose loads are managed by a scheduler. The receiving servers will validate data and metadata, organize the data into collections, and register the metadata into a unified metadata database before storing the data into the enterprise storage management environment. We provide a brief description of the overall architecture and some of the software components in the next two sections. We are currently conducting experimental results to illustrate the scalability of our software using the storage systems available through the Global Land Cover Facility (GLCF) at Maryland.

[*] Sponsored by the National Archives and Records Administration and the National Science Foundation under the PACI Program.

0-7695-2318-8/05 $20.00 © 2005 IEEE

2. Overview of our Architecture

Our overall architecture is shown in Figure 1. PAWN realizes this architecture through four software components: a management server and clients at each producer; verification services and receiving servers connected to mass storage systems.

We assume that, in general, a number of people at each producer will be engaged in preparing and transferring data to the receiving servers. The management server at that producer will act as a central point for the initial organization of the data, and for tracking bitstreams and metadata functionality. More specifically, this server performs the following functions:

- It provides the necessary security infrastructure to allow secure transfer of bitstreams between the producer and the receiving site.
- It assigns a unique identifier for each bitstream to be transferred.
- It provides an interface for bitstream organization and metadata editing.
- It accepts checksums/digital signature, system metadata and other client supplied metadata.
- It tracks which bitstreams have been transferred to the receiving servers.

A client will run on each machine to automatically register metadata information and bulk transfer its data to the data center. The client will be responsible for:

- Bulk registration of bitstreams, checksums and system metadata;
- Assembly of a valid data stream;
- Transmission of the data stream to the center either directly or through a third party proxy server.

The center is assumed to have a cluster of servers setup to receive data transferred from the producers, managed through a scheduler. Each receiving server will accept data and initiate verification/validation processes on the bitstream. Each receiving server will do the following:

- Securely accept a data stream from clients at a producer site as assigned by the load balancer;
- Process data streams and initiate verification/validation processes;
- Coordinate authentication with the management server at the corresponding producer site;
- Verify with the management server that all data streams have arrived intact; and
- Provide enough temporary storage for incoming data streams until they can be validated and then transferred to mass storage systems.

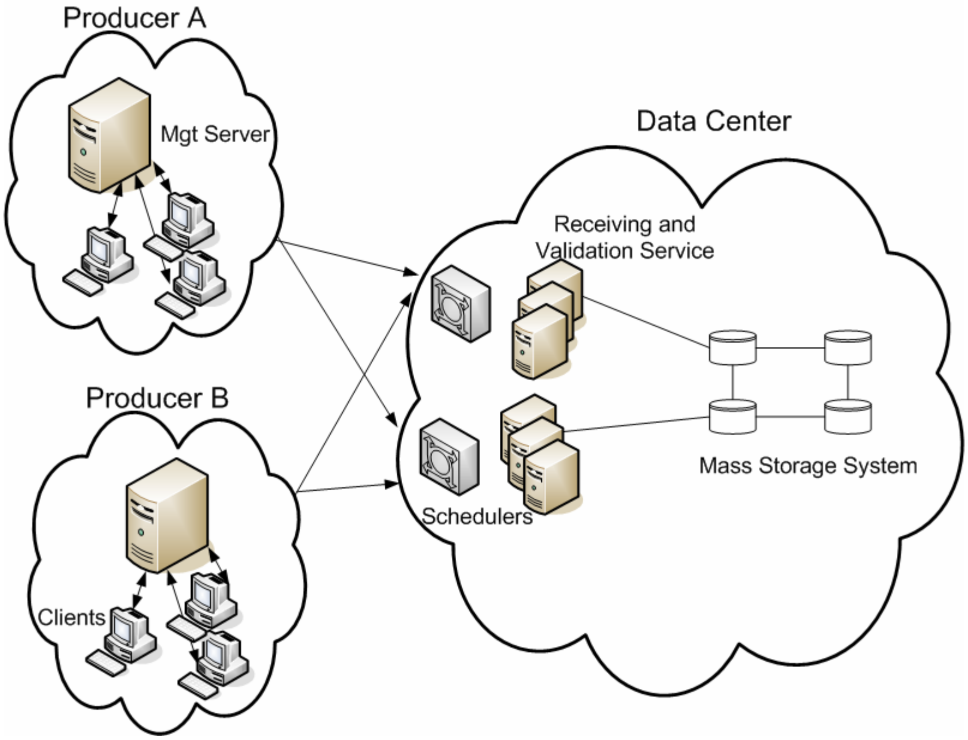

Figure 1. Architectural overview of PAWN.

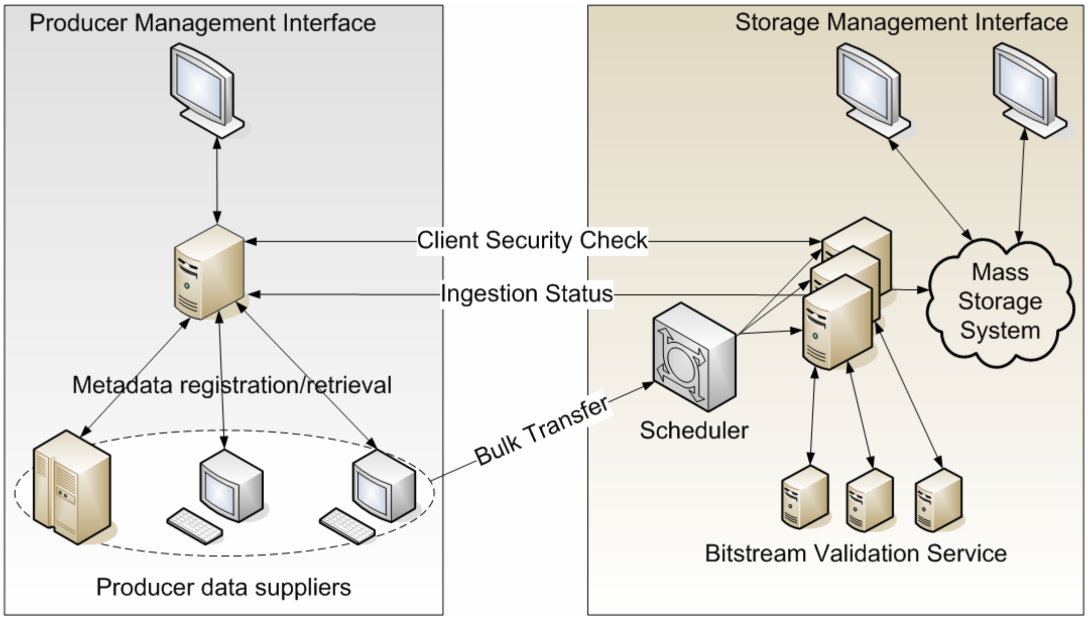
Producer Management Interface
Storage Management Interface
Client Security Check
Ingestion Status
Mass Storage System
Metadata registration/retrieval
Bulk Transfer
Scheduler
Producer data suppliers
Bitstream Validation Service

Figure 2. Software Components in PAWN.

3. Some Implementation Details

PAWN was implemented using open standards and protocols and web technologies. Its software components include the XML schema and editing tools, metadata storage and arrangement at the producer, client uploads, servers at the data center to receive data, and bitstream verification services (Figure 2).

3.1. The Management server at the producer site

As stated earlier, the management server serves as a central point for organizing the data and tracking bitstreams and metadata at its producer site. It plays two roles, one to store and manage submissions from various clients, and the other to manage security issues between clients, itself, and the data center. All communication between clients, a data center, and management is performed through a common web service layer. This allows for a variety of clients from various platforms and languages to connect.

3.1.1. Security Management in PAWN. Our overall security architecture is based on open standards (PKI, X.509, and GSI) and distributed trust management. It enables mutual authentication, confidential communication, and requires no or minimum user intervention. Since we assume minimal operational trust between the data center and producer, we allow for each party to manage security locally.

The producer will set up a Certificate Authority (CA), or use a pre-existing one that is trusted by the receiving servers at the archive. The producer's CA will track current valid certificates and assign a unique certificate for each client that will supply data. At the data center, a receiving server will verify connecting clients by checking their certificates against the producer CA certificate in the same way a web browser verifies a bank's certificate against a commercial CA such as Verisign.

Aside from setting up trust, the validity of connecting clients needs to be determined at both the producer and the data center. The producer site may revoke certificates of clients at any point due to security concerns or other reasons. A SOAP call to the security component of the management server will determine whether the presented certificate is on a revocation list, or if it is valid.

We have developed a Web-based CA for sites that do not have a pre-existing CA infrastructure. This CA will generate and manage standard X.509 certificates. Such a CA draws on the advantages of Java and some J2EE features. The software requires minimum knowledge and instructions to set up and configure, and can be repackaged and redeployed at any time for easier back-ups or system migration, and is completely self-contained to ensure minimum external dependencies.

Certificate management within PAWN uses the BouncyCastle API and java keystore technology. While certificates are stored in java keystore format for internal usage, they can be exported as grid compatible certificates.

3.1.2. Bistream and Metadata Management. The management interface performs two functions, editing metadata and allowing organization of registered bitstreams. An arrangement interface enables the producer to organize and provide context to the bitstreams. This component allows for different individuals within the producer organization to have different roles in arranging and editing attributes associated with a bitstream. In our current version, we provide the ability to hierarchically arrange bitstreams.

As our model is inherently distributed with multiple producers, the roles of individuals vary significantly from other systems that assume ingestion through a centralized pipeline. As the content ingested from one producer is anticipated to be under one administrative domain, the roles should be considerable simplified.

3.2. The Client at the Producer

A stand-alone client was developed to run on each data source at the producer. This client communicates with the management server using X.509 certificates. The current version was tested on Linux and Windows. It offers file browsing/selection options, perform registration of a bitstream and system metadata to the management server, and transfer data to a receiving server. The client is also capable of harvesting metadata information such as local file attributes (checksum, size, type, etc.) and original host and file location, and can arrange files into an abstract hierarchy.

To implement the client, we chose to create a standalone Java application run from CDROM to be executed on each workstation. The client does not store any data locally on the machine. We anticipate that in many cases a user will not have appropriate permissions to install software on a machine. Running a standalone application from a CDROM will allow someone to transfer their own files without compromising their systems integrity.

3.3. The Receiving Server

The receiving server coordinates with the management server of the producer to authenticate a connecting client, provides temporary storage for incoming bitstreams, triggers various validation services, and publishes bitstreams into a mass storage system. The receiving server is designed to be a standalone server with enough storage for processing and verifying data transferred. It is a lightweight service that can be easily duplicated behind load balancing technologies to achieve high throughput.

Data is transferred to a receiving server by various clients. When a client requests a transmission, the load balancer assigns it to a receiving server. The assigned receiving server will first check with the producer's revocation list to ensure a client is valid before accepting client data transmission. Once data arrives, the server will verify metadata and bistream.

We have created a Web Services Definition Language (WSDL) interface file that will specify the communication between the receiving server and various validation services. Using an http-based communication path allows us to scale the validation service using traditional web load balancing techniques. We have developed a Web service framework that is accessible via SOAP to provide the validation service. The framework should be extensible through add-on modules to validate bitstreams in different file formats. A few sample plug-in modules for validating major file formats, such as JPEG, TIFF, and PDF have already been demonstrated in our framework.

3.4. Scheduler

PAWN optimizes performance between producers and receivers using standard Internet Protocol traffic management techniques. The main interaction between producer client and receiving server is a single http call and as such requires no state tracking beyond a single TCP stream. However, as the submission requirements will vary from client to client, some level of scheduling is necessary to ensure intelligent use of available resources, and as transfer requests approach multi-gigabyte or terabyte sizes some level of checkpointing will become necessary.

A scheduler can be used to allocate the processing of the data streams from the different clients to a cluster of receiving servers residing at the data center. Unlike the case of handling similar (small) requests from a very large number of users in a typical commercial enterprise environment, which can be handled using proven load balancing technologies, the requests in our environment can vary substantially in size and processing times. In particular, each of our receiving servers has to invoke validation and verification services and provide temporary storage for incoming data streams that are potentially of widely different sizes and types.

There are several well-known schedulers that optimize the utilization of available resources while scheduling jobs based on priority, resource requirement, fairness, etc. Among the most widely used schedulers in high performance computing environments are PBS (Portable Batch System) with the Maui Scheduler as plug-in, and Condor. PBS is based on the concept of resource reservation that enables policy-based scheduling environment, and attempts to maximize the utilization of resources and hence minimize the turnaround time. On the other hand, Condor is used as a high-throughput scheduler to leverage free cycles available on large collections of privately owned resources.

Since our environment is simpler than the general scheduling environment, we have opted to build our own scheduler rather than using PBS or Condor. The most important consideration when matching clients to receiving servers is to assure each receiving server will have adequate space to handle incoming transmissions.

This will involve more coordination between receiving server and scheduler because of the unpredictable nature of client transmissions.

We expect our receiving servers to be partitioned into different groups, each responsible for handling data submissions of certain sizes. For example, we expect one group to be reserved to handle very large submissions (say 100GB or more), and hence each server in this group needs to have appropriately-sized storage attached to it. Our scheduler will use backfill strategy to allocate receiving servers within each group. With such a strategy, we will be able to make efficient utilization of resources while optimizing the processing of different data streams.

References

[1] PAWN: Producer – Archive Workflow Network in Support of Digital Preservation, M. Smorul, J. JaJa, Y. Wang, and F. McCall, UMIACS Technical Report TR-2004-49, University of Maryland, 2004.

[2] Producer-Archive Interface Methodology: Abstract Standard, Consultative Committee for Space Data Systems, CCSDS- 651.0-R-1, Red Book, December 2002.

[3] Reference Model for an Open Archival Information System (OAIS), CCSDS 650.0-B-1, Blue Book, Issue 1, January 2002 [Equivalent to ISO 14721:2002].

[4] METS – Metadata Encoding and Transmission Standard.

[5] http://www.loc.gov/standards/mets/

[6] NARA Persistent Archives: NPACI Collaboration Project, R. Moore et al., SDSC Technical Report, 2003.