Conceptual Background for Modeling the Data Universe

Brian McMillin

Modeling the performance of the Data Universe requires a certain background in the capacities of the storage and communication resources to be used. This discussion breaks each aspect of the Data Universe down into simple elements that build progressively toward the ultimate goal. With luck and patience, the performance of each level of sophistication can be modeled and some idea of the performance of the overall concept can be achieved.

The Data Universe is a conceptual subset distributed across the current Internet. Communication links exist between Host computers and join the elements within the Data Universe. Data also flows between the Data Universe and the Outside World, by which I mean the rest of the Internet.

Conceptual Steps for Building the Data Universe

1. **Define a Host** - Storage, Bandwidth and Processing capabilities
    1. Each Host has a unique address H, perhaps consisting of IP:Port
    2. The repository of a given host H may store at most S_H blocks.
    3. Each Host can attempt to send data blocks to any other host, but there is no guarantee of success
    4. Each Host is interconnected with bandwidth capable of sending and receiving R_H blocks per second.
    5. Host processors are capable of some level of processing based on the content of their repositories.

2. **Holographic Diffusion**
    1. Normal (gaseous) Diffusion **moves** particles randomly to "nearby" positions in space
    2. Holographic Diffusion, for want of a better term, **copies** data blocks randomly to connected positions in a physical address space
    3. At a rate R_H, each host will select a random block from S_H and attempt to send it to another host address.
    4. Fundamental Communication Features
        1. Random Blocks sent to random Hosts
        2. One-way datagrams
        3. Point-to-point: No broadcast or multicast needed
            1. On the other hand...couple it with USENET storage and broadcast with NNTP
        4. Latency Independent
        5. Connectionless
        6. No Confirmation
    5. A data block D_N, received at host H will overwrite a randomly selected block in S_H.
    6. A particular data block D_N will occur at most one time in S_H.
    7. Assume communication between Hosts is a connected graph, like the Internet.
        1. Explore the ramifications of Connected Subnets (like computers behind firewalls)
        2. Extend addressability to cover such situations
        3. Explore the tradeoffs involved in making the communication two-way
    8. Tends (very slowly) toward static equilibrium
    9. Not very efficient

3. **Dynamically Add and Remove Hosts**
    1. Adding and Removing Hosts corresponds to Computers coming online or going offline or changing IPs
    2. H_{total} is now a function of time
    3. New Hosts per second given by $\partial H_{total} / \partial t$

4. **Dynamically Add and Remove Data Blocks**
    1. Adding block D_N to a host is equivalent to receiving the block from another Host or the Outside World
    2. Removing block D_N from a host is what happens when a block is overwritten by a different received block
    3. Removing all copies of block D_N from *all* Hosts is what happens when a block expires
        1. Expiration simply means the block will be preferentially overwritten by received blocks

4. B_{total} is now a function of time
5. New Data Blocks per second given by $\partial B_{total} / \partial t$
6. Consider the ramifications of the *same* block D_N being added at multiple hosts (i.e. popular .MP3 files).

5. **Directed Replication** - Make the Holographic Diffusion process more efficient
   1. Add the concept of a Host List Block which can propagate from Host to Host like any other Block
   2. Hosts periodically build new Host List Blocks based on knowledge of successful communication and the contents of other Host List blocks that may have been received.
   3. Gives the transmissions a vastly better chance of succeeding
   4. Requires Hosts to examine the content of the data Blocks
   5. Implies some form of time synchronization and expiration of obsolete data

6. **Group Related Data Blocks** - Create the traditional concept of a File.
   1. Allows meaningful data to be exchanged between the Data Universe and the Outside World
   2. Add the concept of a File Description Block which can propagate like any other Block
   3. Describes the file by name, date, author, contents, etc.
   4. Lists the additional data Blocks required to reassemble the original data.
   5. Include a signature for the data in the entire file, so that
      1. Successful reconstruction of the entire file can be verified.
      2. Multiple copies of the same file with different names or descriptions can be grouped.
      3. Different files (versions?) with the same name or description can be kept separate.
      4. Annotations referring to a specific file can be added in the form of supplemental File Descriptions
   6. Here we should discuss the ramifications of the choice of Block size
      1. Initial suggestion is 65500 bytes, leaving room for the Block ID to be stored with the Block
      2. Fits in a single IPv4 packet (probably). Does this really matter?
      3. Implies a chaining or "include" requirement for File Description Blocks
      4. 8192 byte file system "clusters" are way too small. 20th century design constraints in the 21st century.

7. **Queries** - Provide a method to select particular data to be extracted from the Data Universe
   1. Allow searching File Description Blocks on a host.
   2. Cause a host to search for the required data Blocks (by ID) to rebuild a file in the Outside World.
   3. May be unsuccessful if the necessary Blocks are not present in the Host's repository.
   4. Queries are non-deterministic and must be designed to expire.

8. **Query Propagation** - Provide a method to get the data Blocks I want onto my specific Host
   1. Query Blocks have four states: Pending, Successful, Unsuccessful, and Expired.
   2. Successful Queries generate one or more result Blocks directed back to a Host specified by the originator.
   3. Unsuccessful Queries propagate intact to one or more Hosts (like ripples in a pond) to be tried again.
   4. Queries for "popular" data are more likely to be successful because more copies of the target exist.

9. **Query Optimization** - Make it efficient enough to be useful
   1. Add the concept of a Directory List Block.
      1. List of Block Ids known to reside on a particular Host (at some point in time, maybe not now...)
      2. Variant has data for multiple Hosts, which could give a choice of Hosts for the same Block.
   2. Add the concept of an Index Block.
      1. List of File Description Block Ids known to contain a particular keyword or feature
      2. Expand it to include keywords in file *contents*, not just descriptions
   3. Consider the effects of a Fan Out parameter and Queries returning multiple Blocks
      1. Fan Out > 1 propagates unsuccessful Queries exponentially until they expire
      2. Successful Queries may return multiple Blocks to the designated "originating" Host
      3. Explore ways to prevent pathological or malicious behavior.
   4. Consider the effects of Universe Spiders

1. Web crawlers adapted to life in the Data Universe.
2. Automated systems that rebuild data files and generate indexes of the content.
3. Dredge up rarely accessed data and keep multiple copies in existence
4. Use error recovery mechanisms to recreate blocks that are not readily available

10. **Intelligent Insertion** - Data transport from the Outside World
    1. Automate useful Data, File Description, and Index Block creation from data in the Outside World.
    2. Extract metadata from image files
    3. Extract individual files from archives (.ZIP or .TAR) and add both the files and the complete archive.

11. **Fault Recovery** - Make it robust enough to be useful
    1. Implement Error Correction Algorithms.
    2. Recover missing Blocks.
    3. Correct the (vanishingly small) chance of duplicate Block IDs

12. **Security** - Answer privacy concerns
    1. Encrypt the data as it is added to prevent eavesdropping
    2. Add cryptographic signatures to ensure authenticity
    3. Manage Cryptographic Keys
    4. No Key Revocation
        1. You *can* say that documents signed with this key after a certain time are not valid
        2. If the encryption key is compromised an adversary could still forge documents
        3. If the decryption key is compromised *ALL* documents using it become vulnerable
        4. This is not a new or unique problem, just important since the Data Universe is, by design, an archive of all such documents.

13. **Computational Queries** - Massively Parallel Distributed Computing
    1. Perhaps Javascript in Query Blocks
    2. Collect input data from the Data Universe, process it, and return Results to the Universe.
    3. Effects of possible pathological behavior
    4. Using the Block “include” feature would allow large code libraries, etc.
        1. Would never suffer from the “wrong version of the .DLL” problem
        2. Ensures the code always runs as the author wrote it. Good or Bad.

Parameters Required for Modeling

Data Block contents are denoted by D_N, $N \in \{ 0 .. 2^{128}-1 \}$ where N is the Block ID. This is for the MD5 version.

N is divided into two distinct subsets: "defined" and "undefined". The "defined" subset is the set for which D_N is known.

Total Actual Unique Data Blocks B_{total} in the Data Universe is a the number of elements in the "defined" subset of N.

Physical Host addresses $H \in \{ 0 .. 2^{48}-1 \}$ (assuming naming using IPv4 address space of the form IP:Port).

H is divided into two distinct subsets: "implemented" and "not implemented" depending on whether a host with host address H exists and can communicate with the network.

Total number of real Hosts H_{total} is a the number of elements in the implemented subset of H.

Storage per Host S_H = Number of Blocks that can be stored on Host H. Hosts that are "not implemented" have zero space.

Total Physical Storage in the Data Universe is $S_{total} = \sum S_H$.

Diffusion Rate R_H is the number of transmissions per second attempted by Host H. Hosts may have different rates based on available bandwidth.

Suggested Models

Start with an even distribution of B_{total} / H_{total} Blocks stored on each Host.
Compute the storage utilization on each Host after blocks diffuse for time T.
Prove that the equilibrium number of Block copies is S_{total} / B_{total}, in the limit as time $T \Rightarrow \infty$

What if the initial condition includes a distribution with more than B_{total} / H_{total} Blocks stored on each Host. In this case, "popular" data Blocks will already exist on multiple hosts. Note the restriction that a particular Block D_N can never be duplicated on the same host.

With each different level of sophistication, design models that allow answers to the following questions to be derived.

1. How many copies of a specific data block D_N will exist in the Data Universe?
2. How many Hosts will be involved in a Query for a particular data block D_N? Typical and worst case.
3. What is the optimum timeout for Query Blocks?
4. From any Host in the Universe, how long should it take to rebuild a file of a particular size?
5. Is there an optimum bandwidth for a given repository size? Is there any correlation at all?
6. What are the effects of changing the rate at which new data is introduced from the Outside World?
7. What are the effects of changing the rate at which physical storage is added to the Universe?
8. How much network traffic will be devoted to Query processing and how much to permanent data "diffusion"?
9. How much storage will be devoted to permanent data, and how much to blocks that will expire?
10. How much storage will be devoted to File Descriptions and how much to content Blocks?

Consider the effects of a technological adversary capable of fabricating gibberish data Blocks with duplicate hash Ids. This is considered computationally infeasible now, but Quantum Computing or other "outside the box" developments might make understanding the ramifications important. Is the minimum 128-bit Block ID appropriate?