Storage Networks
Optimizing Heterogeneous Data Distribution

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
Literature

The Problem in Storage Networks

- $A_{d,s}$: Number of bytes of document $d$ assigned to storage $s$

Distributed Algorithm:
- Use DHHT to split each document into $|S|$ parts
- Store corresponding blocks on the server

Can be also achieved by a centralized algorithm

Straight forward generalization of fair balance
- Distribute data according to a $(m \times n)$ distribution matrix $A$ where
  \[ \forall s \colon \sum_d A_{d,s} \leq |s| \text{ and } \forall d \colon \sum_s A_{d,s} = |d| \]

DHHT
- assigns $A_{d,s}(1 \pm \varepsilon)$ elements of $d \in D$ to $s \in S$
- Information needed: File-IDs, Server-IDs, and matrix $A$
- If matrix $A$ changes to $A'$, $(1 + \varepsilon) \sum_{d,s} |A_{d,s} - A'_{d,s}|$
data reassignments are needed
A fair balance like $A_{d,s} = |d| \cdot \frac{|s|}{\sum_{s' \in S} |s'|}$ is not always the best to do.

Servers are different in capacity and bandwidth.

Documents are different in size and popularity.

Goal: Optimize Time.

Assumption
- All sizes can be modeled as real numbers.
Which Time?

- $b(s) = \text{bandwidth of server } s$
  - $b(s) = \text{number of bytes per second}$
- $p(d) = \text{popularity of document } d$
  - $p(d) = \text{number of read/write accesses}$
- **Sequential time for a document } d \text{ and an assignment } A**
  \[
  \text{SeqTime}_A(d) := \sum_{s \in S} \frac{A_{d,s}}{b(s)}
  \]
- **Parallel time for a document } d \text{ and an assignment } A**
  \[
  \text{ParTime}_A(d) := \max_{s \in S} \left\{ \frac{A_{d,s}}{b(s)} \right\}
  \]
- **Observation**
  - Popular bytes cause more traffic than less popular once
  - Costs are defined by the traffic per byte
Sequential Time

- Sequential time
  - load all parts of a document from all servers sequentially

\[
\text{SeqTime}_A(d) := \sum_{s \in S} \frac{A_{d,s}}{b(s)}
\]

- Worst case sequential time
  \[
  \text{WSeqTime} := \max_d \{\text{SeqTime}_A(d)\}
  \]

- Average sequential time
  \[
  \text{AvSeqTime} := \sum_{d \in D} p(d) \cdot \text{SeqTime}_A(d)
  \]

where
- \( S \): set of servers with bandwidth \( b(s) \) and capacity \(|s|\) for each server \( s \)
- \( D \): set of documents with size \(|d|\) and popularity \( p(d) \) for each document
Parallel Time

- Parallel time
  - load all parts of a document from all servers simultaneously

\[
\text{ParTime}_A(d) := \max_{s \in S} \left\{ \frac{A_{d,s}}{b(s)} \right\}
\]

- Worst case parallel time
  \[
  \text{WParTime} := \max_d \{\text{ParTime}_A(d)\}
  \]

- Average parallel time
  \[
  \text{AvParTime} := \sum_{d \in D} p(d) \cdot \text{ParTime}_A(d)
  \]

where
- \( S \): set of servers with bandwidth \( b(s) \) and capacity \( |s| \) for each server \( s \)
- \( D \): set of documents with size \( |d| \) and popularity \( p(d) \) for each document
Sequential Bandwidth

- **Sequential time**
  - load all parts of a document from all servers sequentially
  \[
  \text{SeqTime}_A(d) := \sum_{s \in S} \frac{A_{d,s}}{b(s)}
  \]

- **Sequential bandwidth**
  - download speed of a document d
  \[
  \text{SeqBandwidth}_A(d) := \frac{|d|}{\text{SeqTime}_A(d)}
  \]

- **Worst case sequential bandwidth**
  \[
  \text{WBandwidth} := \min \{ \text{SeqBandwidth}_A(d) \}
  \]

- **Average sequential bandwidth**
  \[
  \text{AvBandwidth} := \sum_{d \in D} p(d) \text{SeqBandwidth}(d)
  \]

  where
  - S: set of servers with bandwidth b(s) and capacity |s| for each server s
  - D: set of documents with size |d| and popularity p(d) for each document
Parallel Bandwidth

- **Parallel time**
  - load all parts of a document from all servers in parallel

- **Parallel bandwidth**
  - download speed of a datum $d$

\[
\text{ParBandwidth}_A(d) := \frac{|d|}{\text{ParTime}_A(d)}
\]

- **Worst case parallel bandwidth**
  \[
  \text{WParBandwidth} := \min_d \{\text{ParBandwidth}_A(d)\}
  \]

- **Average parallel bandwidth time**
  \[
  \text{AvParBandwidth} := \sum_{d \in D} p(d) \text{ ParBandwidth}_A(d)
  \]

- where
  - $S$: set of servers with bandwidth $b(s)$ and capacity $|s|$ for each server $s$
  - $D$: set of documents with size $|d|$ and popularity $p(d)$ for each document
Most Reasonable Time Measures

- Minimize the expected sequential time based on popularity of the document:

\[
\text{AvSeqTime}(p, A) = \sum_{d \in D} \sum_{s \in S} p(d) \frac{A_{d,s}}{b(s)}
\]

- Minimize the expected parallel time based on the popularity of the document:

\[
\text{AvParTime}(p, A) = \sum_{d \in D} \max_{s \in S} \frac{A_{d,s}}{b(s)} p(d)
\]
Solution by Linear Program

∀s: \sum_d A_{d,s} \leq |s|

∀d: \sum_s A_{d,s} = |d|

<table>
<thead>
<tr>
<th>Measure</th>
<th>Linear program</th>
<th>Add. variables</th>
<th>Additional restraint</th>
<th>Optimize</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvSeqTime</td>
<td>yes</td>
<td>—</td>
<td>—</td>
<td>min \sum_{s \in S} \sum_{d \in D} p(d) \frac{A_{d,s}}{b(s)}</td>
</tr>
<tr>
<td>WSeqTime</td>
<td>yes</td>
<td>m</td>
<td>\forall d \in D: \sum_{s \in S} \frac{A_{d,s}}{b(s)} \leq m</td>
<td>min m</td>
</tr>
<tr>
<td>AvParTime</td>
<td>yes</td>
<td>(m_{d})_{d \in D}</td>
<td>\forall s \in S, \forall d \in D: \frac{A_{d,s}}{b(s)} \leq m_{d}</td>
<td>min \sum_{d \in D} p(d) m_{d}</td>
</tr>
<tr>
<td>WParTime</td>
<td>yes</td>
<td>m</td>
<td>\forall s \in S, \forall d \in D: \frac{A_{d,s}}{b(s)} \leq m</td>
<td>min M</td>
</tr>
<tr>
<td>AvSeqBandwidth</td>
<td>no</td>
<td>—</td>
<td>—</td>
<td>max \sum_{d \in D} \frac{p(d)</td>
</tr>
<tr>
<td>WSeqBandwidth</td>
<td>yes</td>
<td>m</td>
<td>\forall d \in D: \sum_{s \in S} \frac{A_{d,s}}{b(s)} \leq m</td>
<td>min m</td>
</tr>
<tr>
<td>AvParBandwidth</td>
<td>no</td>
<td>(m_{d})_{d \in D}</td>
<td>\forall d \in D: \sum_{s \in S} \frac{A_{d,s}}{b(s)</td>
<td>d</td>
</tr>
<tr>
<td>WParBandwidth</td>
<td>yes</td>
<td>m</td>
<td>\forall s \in S, \forall d \in D: \frac{A_{d,s}}{</td>
<td>d</td>
</tr>
</tbody>
</table>
How to Describe AvParTime as a LP

AvParTime

\[ = \sum_{d \in D} p(d) \cdot m_d \]

Variables: \( A_{d,i} \), \( m_d \)

Restraints:

\[ \sum_{s \in S} A_{d,i,s} = |d| \]

\[ \sum_{d \in D} \frac{A_{d,i,s}}{b(s)} \leq |S| \]

\[ m_d = \max_{s \in S} \frac{A_{d,i,s}}{b(s)} \]

Additional Restraints

\[ m_d \geq \frac{1}{b(S_i)} \cdot A_{d,i,s_1} \]

\[ m_d \geq \frac{1}{b(S_i)} \cdot A_{d,i,s_2} \]
Example

- **Storage device**
  - $s_1$: 500 GB, 100 MB/s
  - $s_2$: 100 GB, 50 MB/s
  - $s_3$: 1 GB 1000 MB/s

- **Documents**
  - $d_1$: 100 GB, popularity 1/111
  - $d_2$: 5 GB, popularity 100/111
  - $d_3$: 100 GB, popularity 10/111

<table>
<thead>
<tr>
<th>$A_{d,s}$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$\Sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>$d_2$</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>$d_3$</td>
<td>2</td>
<td>98</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>$\leq$ 500</td>
<td>$\leq$ 100</td>
<td>$\leq$ 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SeqTime</th>
<th>SeqBand width</th>
<th>ParTime</th>
<th>ParBand width</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>1000</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>$d_2$</td>
<td>61</td>
<td>82</td>
<td>40</td>
<td>125</td>
</tr>
<tr>
<td>$d_3$</td>
<td>1980</td>
<td>51</td>
<td>1960</td>
<td>51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SeqTime</th>
<th>SeqBand width</th>
<th>ParTime</th>
<th>ParBand width</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>1000</td>
<td>100</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>$d_2$</td>
<td>61</td>
<td>82</td>
<td>40</td>
<td>125</td>
</tr>
<tr>
<td>$d_3$</td>
<td>1980</td>
<td>51</td>
<td>1960</td>
<td>51</td>
</tr>
<tr>
<td>$\text{Av}$</td>
<td>1864</td>
<td>121</td>
<td>1827</td>
<td>160</td>
</tr>
<tr>
<td>$\text{Worst case}$</td>
<td>1980</td>
<td>51</td>
<td>1960</td>
<td>51</td>
</tr>
</tbody>
</table>
Excursion: Linear Programming

- Linear Program (Linear Optimization)
- Given: $m \times n$ matrix $A$
  - $m$-dimensional vector $b$
  - $n$-dimensional vector $c$
- Find: $n$-dimensional vector $x = (x_1, ..., x_n)$
- such that
  - $x \geq 0$, i.e. for all $j$: $x_j \geq 0$
  - $A x = b$, i.e. $\sum_{j=1}^{n} \sum_{i=1}^{m} A_{ij} x_j = b_j$
  - $z = c^T x$ is minimized, i.e. $z = \sum_{j=1}^{n} c_j x_j$ is minimal
Linear Programming 2

- Linear Programming (LP2)
- Given: $m \times n$ matrix $A$
  - $m$-dimensional vector $b$
  - $n$-dimensional vector $c$
- Find: $n$-dimensional vector $x=(x_1, \ldots, x_n)$
- such that
  - $x \geq 0$
  - $A \cdot x \leq b$
  - $z = c^T \cdot x$ is maximal
Performance of Linear Programming

- Worst case time behavior of the Simplex algorithm is exponential
  - A simplex can have an exponential number of edges
- For randomized inputs, the running time of Simplex is polynomial on the expectation
- The Ellipsoid algorithm is a different method with polynomial worst case behavior
  - In practice it is usually outperformed by the Simplex algorithm
ParTime = SeqTime with virtual servers

- Reduce optimal solution for LP of ParTime to the optimal solution of LP of SeqTime
  - Combining capacity of many disks in parallel
- Define new sequential virtual servers $s'_1, ..., s'_m$
  - Sort $s_i$ such that $\frac{|s_j|}{b(s_j)} \leq \frac{|s_{j+1}|}{b(s_{j+1})}$
  - Server $s'_j$ parallelizes servers $s_j,...,s_{|S|}$
  - Virtual servers $s'_i$ are then sorted such that $b(s'_i) > b(s'_{i+1})$
  - Size of $s'_i$:
    \[
    t_j = \frac{|s_j|}{b(s_j)} - \sum_{i=1}^{j-1} t_i
    \]
    \[
    s'_j = b(s'_j) \cdot t_j
    \]
Solve the LP of $\text{AvSeqTime}$

- Simple optimal greedy solution

- Repeat until all documents are assigned:
  - Assign most popular document on fastest sequential (virtual) server
  - Reduce the storage of the server by the document size and remove the document
Applications in Storage Networks

- **Object storage with different popularity zones**
  - e.g. movies with varying popularities over time
  - Fragmentation is done automatically
  - Includes dynamics for adding and removing documents
  - The same for servers

- **Use different bandwidth**
  - Each disk has different bandwidths
  - Exporting different zone classes as sequential servers
Storage Networks
Optimizing Heterogeneous Data Distribution

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
Optimal File-Distribution in Heterogeneous and Asymmetric Storage Networks

Tobias Langner, Christian Schindelhauer, Alexander Souza

Computer Engineering and Networks Laboratory (TIK)
Department for Information Technology and Electrical Engineering
ETH Zurich, Switzerland

SOFSEM ’11
Nový Smokovec
25 January, 2011
Distributed Storage is Ubiquitous

Motivation

- Storing files on multiple machines is a **common habit**
  - P2P overlays basically represent storage networks
  - File-hosting services like Amazon S3, Rapidshare, etc.
Distributed Storage is Ubiquitous

Motivation

- Storing files on multiple machines is a common habit
  - P2P overlays basically represent storage networks
  - File-hosting services like Amazon S3, Rapidshare, etc.
- Gigantic storage requirements by special applications (Google, Amazon, CERN)
Distributed Storage is Ubiquitous

- Storing files on multiple machines is a **common habit**
  - P2P overlays basically represent storage networks
  - File-hosting services like Amazon S3, Rapidshare, etc.
- **Gigantic storage requirements** by special applications (Google, Amazon, CERN)
- Most existing approaches specifically tailored for data centers with **homogeneous and symmetric network connections**.
Motivation

Distributed Storage is Ubiquitous

Storing files on multiple machines is a common habit
- P2P overlays basically represent storage networks
- File-hosting services like Amazon S3, Rapidshare, etc.

Gigantic storage requirements by special applications (Google, Amazon, CERN)

Most existing approaches specifically tailored for data centers with homogeneous and symmetric network connections.

Little attention to asymmetric bandwidths typical for end-user connections.
Synopsis

1 Motivation

2 Problem Setting

3 Analytical Scaling
   - Maximum Flow in Distribution Problems
   - Total Data Function in 3D
   - The Algorithm

4 Conclusion
Servers \( s_i \) with \( i \in S = \{1, \ldots, m\} \), and one client \( c \).

Each server \( s_i \) is characterized by its upload bandwidth \( u_i \) and download bandwidth \( d_i \).
Problem Setting

Distribution Problem

Problem Setting

- How do we split a file \( f \) of size \(|f|\) into fragments for the servers to minimize the time for one upload and \( n \) downloads?

\[ x_i = |f| \]

\( x_i \) is the size of the fragment of server \( s_i \).

Equivalent simplified problem with \(|f| = n = 1\).
Problem Setting

Distribution Problem

Problem Setting

- How do we split a file $f$ of size $|f|$ into fragments for the servers to minimize the time for one upload and $n$ downloads?

- Distribution vector $\mathbf{x} = (x_1, \ldots, x_m)$ with $\sum x_i = |f|$. $x_i$ is the size of the fragment of server $s_i$. 
Problem Setting

Distribution Problem

Problem Setting

- How do we split a file $f$ of size $|f|$ into fragments for the servers to minimize the time for one upload and $n$ downloads?

- Distribution vector $\mathbf{x} = (x_1, \ldots, x_m)$ with $\sum x_i = |f|$. $x_i$ is the size of the fragment of server $s_i$.
- Equivalent simplified problem with $|f| = n = 1$. 

T. Langner et al. Optimal File-Distribution in Heterogeneous and Asymmetric Storage Networks
The quality of a distribution $x$ is given by the transfer times:

- **Upload time:**
  
  $$ t_u(x) = \max_{i \in S} \left\{ \frac{x_i}{u_i} \right\} $$
The quality of a distribution $\mathbf{x}$ is given by the transfer times:

- **Upload time:**
  \[
  t_u(\mathbf{x}) = \max_{i \in S} \left\{ \frac{x_i}{u_i} \right\}
  \]

- **Download time:**
  \[
  t_d(\mathbf{x}) = \max_{i \in S} \left\{ \frac{x_i}{d_i} \right\}
  \]
The quality of a distribution $x$ is given by the transfer times:

- **Upload time:**
  \[
  t_u(x) = \max_{i \in S} \left\{ \frac{x_i}{u_i} \right\}
  \]

- **Download time:**
  \[
  t_d(x) = \max_{i \in S} \left\{ \frac{x_i}{d_i} \right\}
  \]

- **Total time:**
  \[
  \text{val}(x) = t_u(x) + (n \cdot) t_d(x)
  \]
We are given a distribution network and a file $f$ with size 1.

Find the optimal distribution $x^*$ that minimizes the total time

$$\text{val}(x) = t_u(x) + t_d(x).$$
The distribution problem can be formulated as a linear program. 
⇒ Its solution space is a convex region.
⇒ Every local minimum is global.
Non-Linear Optimization Problem

Upload/download bandwidths $u_1$ to $u_m$ and $d_1$ to $d_m$

File distribution $x = (x_1, \ldots, x_m)$

Non-Linear Program

$$\begin{align*}
\text{minimize} & \quad \max \left\{ \frac{x_1}{u_1}, \frac{x_2}{u_2}, \ldots, \frac{x_m}{u_m} \right\} + \max \left\{ \frac{x_1}{d_1}, \frac{x_2}{d_2}, \ldots, \frac{x_m}{d_m} \right\} \\
\text{subject to} & \quad \sum_{i=1}^{m} x_i = 1 \\
& \quad x_i \geq 0 \quad \text{for all } i \in \{1, \ldots, m\}
\end{align*}$$
Linear Optimization Problem

Linear Program

minimize \( t_u + t_d \)

subject to

\[
\sum_{i=1}^{m} x_i = 1
\]

\[
\frac{x_i}{u_i} \leq t_u \quad \text{for all } i \in \{1, \ldots, m\}
\]

\[
\frac{x_i}{d_i} \leq t_d \quad \text{for all } i \in \{1, \ldots, m\}
\]

\( x_i \geq 0 \quad \text{for all } i \in \{1, \ldots, m\} \)

Dimension: \( m + 2 \)
Synopsis

1 Motivation

2 Problem Setting

3 Analytical Scaling
   - Maximum Flow in Distribution Problems
   - Total Data Function in 3D
   - The Algorithm

4 Conclusion
The Distribution Problem as Flow Problem

Assume we are given upload time $t_u$ and download time $t_d$. 
Assume we are given upload time $t_u$ and download time $t_d$. The maximal amount of data that can be uploaded within $t_u$ and downloaded within $t_d$ corresponds to a maximum flow. The value of the maximum flow $f^*$ equals the capacity of the minimal cut $\text{val}(f^*) = \min_{i=1}^{m} \{ t_u \cdot u_i, t_d \cdot d_i \}$. 

![Diagram showing the distribution problem as a flow problem with nodes and edges labeled with upload and download times.](image)
The Distribution Problem as Flow Problem

Assume we are given upload time $t_u$ and download time $t_d$.

The maximal amount of data that can be uploaded within $t_u$ and downloaded within $t_d$ corresponds to a maximum flow.
The Distribution Problem as Flow Problem

Assume we are given upload time $t_u$ and download time $t_d$

Upload

\[ t_u \cdot u_1 \quad s_1 \quad t_d \cdot d_1 \]

\[ t_u \cdot u_i \quad s_i \quad t_d \cdot d_i \]

\[ t_u \cdot u_m \quad s_m \quad t_d \cdot d_m \]

Download

The maximal amount of data that can be uploaded within $t_u$ and downloaded within $t_d$ corresponds to a maximum flow.

Value of maximum flow $f^*$ equals capacity of minimal cut
Assume we are given upload time $t_u$ and download time $t_d$.

The maximal amount of data that can be uploaded within $t_u$ and downloaded within $t_d$ corresponds to a maximum flow.

Value of maximum flow $f^*$ equals capacity of minimal cut

$$\text{val}(f^*) = \sum_{i=1}^{m} \min\{t_u u_i, t_d d_i\}$$
There exists a distribution such that $f$ with $|f| = 1$ can be uploaded and downloaded in time $t_u$ and $t_d$, respectively, if

$$
\sum_{i=1}^{m} \min\{t_{ui}, t_{di}\} \geq 1
$$
There exists a distribution such that $f$ with $|f| = 1$ can be uploaded and downloaded in time $t_u$ and $t_d$, respectively, if

$$
\sum_{i=1}^{m} \min\{t_u u_i, t_d d_i\} \geq 1
$$

We want to find the minimum values for $t_u$ and $t_d$ for which the above predicate holds.
Decision Predicate

Maximum Flow in Distribution Problems

- There exists a distribution such that $f$ with $|f| = 1$ can be uploaded and downloaded in time $t_u$ and $t_d$, respectively, if

$$\sum_{i=1}^{m} \min\{t_u u_i, t_d d_i\} \geq 1$$

- We want to find the minimum values for $t_u$ and $t_d$ for which the above predicate holds.

- Substitute $t_d$ by $T - t_u$ to obtain the total data function.

$$\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u u_i, (T - t_u) \cdot d_i\}$$
The total data function for a fixed value of $T$:

$$
\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u u_i, (T - t_u) \cdot d_i\}
$$
Total Data Function

The total data function for a fixed value of $T$:

$$\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u \cdot u_i, (T - t_u) \cdot d_i\}$$

Can be evaluated in $O(m \log m)$ steps.
The total data function for a fixed value of $T$:

$$
\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u u_i, (T - t_u) \cdot d_i\}
$$
The total data function for a fixed value of $T$:

$$
\delta_T(t_u) = \sum_{i=1}^{m} \min \{ t_u u_i, (T - t_u) \cdot d_i \}
$$

Can be evaluated in $\mathcal{O}(m \log m)$ steps.
Evaluation

The Total Data Function

\[ \delta_T(t_u) \]

\[ \sigma_i \text{ is slope in interval } I_i. \]
Evaluation

The Total Data Function
Evaluation
The Total Data Function

\[ \delta_T \]

\[ l_1 \quad l_2 \quad l_3 \quad l_4 \]

\[ \sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \sigma_4 \]

\[ 0 \quad p_1 \quad p_2 \quad p_3 \quad T \]

\[ t_u \]

\( \sigma_i \) is slope in interval \( l_i \).
Evaluation
The Total Data Function

\[ \delta_T(p_i) = \delta_T(p_{i-1}) + \sigma_i \cdot (p_i - p_{i-1}) \]

\( \sigma_i \) is slope in interval \( I_i \).

Recursion formula
The total data function $\delta_T(t_u)$ determines how much data can be uploaded to and downloaded again from the servers within time $T$ for different values of $t_u$. 
Scaling the Total Data Function

Total Data Function in 3D

- The total data function $\delta_T(t_u)$ determines how much data can be uploaded to and downloaded again from the servers within time $T$ for different values of $t_u$.
- Varying values of $T$ only scale the total data function.
Interpret $\delta_T(t_u)$ as **two-variable** function $\delta(T, t_u)$ now.
Two-Variable Total Data Function

Total Data Function in 3D

- Interpret $\delta_T(t_u)$ as two-variable function $\delta(T, t_u)$ now

- To find the minimal value of $T$ with $\delta(T, t_u) \geq 1$ for some $t_u$, establish straight line through the highest point of $\delta(T, t_u)$
Two-Variable Total Data Function

Total Data Function in 3D

- Interpret $\delta_T(t_u)$ as two-variable function $\delta(T, t_u)$ now

- To find the minimal value of $T$ with $\delta(T, t_u) \geq 1$ for some $t_u$, establish straight line through the highest point of $\delta(T, t_u)$
- Determine where it intersects the plane $\delta = 1$
Determining the Actual Distribution

The Algorithm

- We determine the minimal value of $T$ as depicted.
Determining the Actual Distribution

The Algorithm

- We determine the minimal value of $T$ as depicted.

- The decomposition of $T$ into $t_u$ and $t_d$ is given by the coordinates of the intersection point.
We determine the minimal value of $T$ as depicted.

The decomposition of $T$ into $t_u$ and $t_d$ is given by the coordinates of the intersection point.
Determining the Actual Distribution

The Algorithm

- Algorithm to determine a solution with these time bounds:
  - Iterate through all servers $s_i \in S$
  - Assign to $s_i$ the data amount $x_i = \min\{t_u u_i, t_d d_i\}$
  - Return the resulting distribution
Determining the Actual Distribution

The Algorithm

- Algorithm to determine a solution with these time bounds:
  - Iterate through all servers \( s_i \in S \)
  - Assign to \( s_i \) the data amount \( x_i = \min\{t_u u_i, t_d d_i\} \)
  - Return the resulting distribution

- Through the feasibility predicate

\[
\sum_{i=1}^{m} \min\{t_u u_i, t_d d_i\} = 1
\]

we know that \( \sum x_i = 1 \) and thus \( x \) is a valid distribution.
Evaluating the total data function for a fixed value of $T$ runs in $O(m \log m)$ steps.
Runtime Complexity

The Algorithm

- Evaluating the total data function for a fixed value of $T$ runs in $O(m \log m)$ steps.

- Intersecting the straight line with the plane $\delta = 1$ is possible in constant time.
Runtime Complexity

The Algorithm

- Evaluating the total data function for a fixed value of $T$ runs in $O(m \log m)$ steps.
- Intersecting the straight line with the plane $\delta = 1$ is possible in constant time.

Runtime complexity: $O(m \log m)$
Synopsis

1 Motivation

2 Problem Setting

3 Analytical Scaling
   - Maximum Flow in Distribution Problems
   - Total Data Function in 3D
   - The Algorithm

4 Conclusion
Summary

- Formal introduction of a new distribution problem
Summary

- Formal introduction of a new distribution problem
- Formulation as linear program
Summary

- Formal introduction of a new distribution problem
- Formulation as linear program
- Derived total data function from flow formulation
Summary

- Formal introduction of a new distribution problem
- Formulation as linear program
- Derived total data function from flow formulation
- Presented Analytical Scaling algorithm to find optimal solutions
Summary

- Formal introduction of a new distribution problem
- Formulation as linear program
- Derived total data function from flow formulation
- Presented Analytical Scaling algorithm to find optimal solutions
Future Directions

- Implementation of our ideas into real-world applications
Future Directions

- Implementation of our ideas into real-world applications
- Cope with non-static, fluctuating bandwidths
Future Directions

- Implementation of our ideas into real-world applications
- Cope with non-static, fluctuating bandwidths
- Respect limited storage capacity of servers
Future Directions

- Implementation of our ideas into real-world applications
- Cope with non-static, fluctuating bandwidths
- Respect limited storage capacity of servers
- Incorporation of redundancy to allow for failure tolerance
The End

Thank you for your attention!

Questions?