Storage Networks
Optimizing Heterogeneous Data Distribution

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

Literature

- André Brinkmann, Kay Salzwedel, Christian Scheideler, Compact, Adaptive Placement Schemes for Non-Uniform Capacities, 14th ACM Symposium on Parallelism in Algorithms and Architectures 2002 (SPAA 2002)
- Christian Schindelhauer, Gunnar Schomaker, Weighted Distributed Hash Tables, 17th ACM Symposium on Parallelism in Algorithms and Architectures 2005 (SPAA 2005)
- Christian Schindelhauer, Gunnar Schomaker, SAN Optimal Multi Parameter Access Scheme, ICN 2006, International Conference on Networking, Mauritius, April 23-26, 2006

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 x n) distribution matrix A where
    $\forall s : \sum_d A_{d,s} \leq |s|$ and $\forall d : \sum_s A_{d,s} = |d|$
- DHHT
  - assigns $A_{d,s}(1 \pm \varepsilon)$ elements of d ∈ D to s ∈ S
  - Information needed: File-IDs, Server-IDs, and matrix A
  - If matrix A changes to A´ $(1+\varepsilon)\sum_{d,s} \left| A_{d,s} - A'_{d,s} \right|$
    data reassignments are needed

How to Balance

- 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) = bandwidth of server s
  - b(s) = number of bytes per second
- p(d) = popularity of document d
  - p(d) = number of read/write accesses
- Sequential time for a document d and an assignment A

$$\mathrm{SeqTime}_A(d) := \sum_{s \in S} \frac{A_{d,s}}{b(s)}$$

- Parallel time for a document d and an assignment A

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

- 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

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

- Worst case sequential time

$$\text{WSeqTime} := \max_d \{\text{SeqTime}_A(d)\}$$

- Average sequential time

$$\text{AvSeqTime} := \sum_{d \in \mathcal{D}} p(d) \ \text{SeqTime}_A(d)$$

- 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

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

- Worst case parallel time

$$\text{WParTime} := \max_d \{\text{ParTime}_A(d)\}$$

- Average parallel time

$$\text{AvParTime} := \sum_{d \in \mathcal{D}} p(d) \ \text{ParTime}_A(d)$$

- 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

$$\mathrm{SeqTime}_A(d) := \sum_{s \in \mathcal{S}} \frac{A_{d,s}}{b(s)}$$

- Sequential bandwidth
  - download speed of a document d

$$\mathrm{SeqBandwidth}_A(d) := \frac{|d|}{\mathrm{SeqTime}_A(d)}$$

- Worst case sequential bandwidth
  WBandwidth := mind {SeqBandwidthA(d)}
- Average sequential bandwidth
  AvBandwidth := $\sum_{d \in \mathcal{D}} p(d)$ 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

CoNe Freiburg

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

$$\mathrm{ParTime}_A(d) := \max_{s \in \mathcal{S}} \left\{ \frac{A_{d,s}}{b(s)} \right\}$$

- Parallel bandwidth
  - download speed of a datum d

$$\mathrm{ParBandwidth}_A(d) := \frac{|d|}{\mathrm{ParTime}_A(d)}$$

- Worst case parallel bandwidth

$$\mathrm{WParBandwidth} := \min_d \{\mathrm{ParBandwidth}_A(d)\}$$

- Average parallel bandwidth time

$$\mathrm{AvParBandwidth} := \sum_{d \in \mathcal{D}} p(d) \; \mathrm{ParBandwidth}_A(d)$$

- 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:

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

- 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

$$\forall s : \sum_d A_{d,s} \le |s| \qquad \forall d : \sum_s A_{d,s} = |d|$$

Measure	Linear programm	Add. variables	Additional restraint	Optimize
AvSeqTime	yes	—	—	$\min \sum_{s\in\mathcal{S}} \sum_{d\in\mathcal{D}} p(d) \frac{A_{d,s}}{b(s)}$
WSeqTime	yes	m	$\forall d \in \mathcal{D} : \sum_{s\in\mathcal{S}} \frac{A_{d,s}}{b(s)} \le m$	$\min m$
AvParTime	yes	$(m_d)_{d\in\mathcal{D}}$	$\forall s \in \mathcal{S}, \forall d \in \mathcal{D} : \frac{A_{d,s}}{b(s)} \le m_d$	$\min \sum_{d\in\mathcal{D}} p(d) m_d$
WParTime	yes	m	$\forall s \in \mathcal{S}, \forall d \in \mathcal{D} : \frac{A_{d,s}}{b(s)} \le m$	$\min M$
AvSeqBandwidth	no	—	—	$\max \sum_{d\in\mathcal{D}} \frac{p(d)\lvert d\rvert}{\sum_{s\in\mathcal{S}} \frac{A_{d,s}}{b(s)}}$
WSeqBandwidth	yes	m	$\forall d \in \mathcal{D} : \sum_{s\in\mathcal{S}} \frac{A_{d,s}}{\lvert d\rvert b(s)} \le m$	$\min m$
AvParBandwidth	no	$(m_d)_{d\in\mathcal{D}}$	$\forall d \in \mathcal{D} : \sum_{s\in\mathcal{S}} \frac{A_{d,s}}{b(s)\lvert d\rvert} \le m_d$	$\max \sum_{d\in\mathcal{D}} \frac{p(d)}{m_d}$
WParBandwidth	yes	m	$\forall s \in \mathcal{S}, \forall d \in \mathcal{D} : \frac{A_{d,s}}{\lvert d\rvert b(s)} \le m$	$\min m$

How to Describe AvParTime as a LP

AvParTime

$$= \sum_{d \in D} p(d) \cdot \underbrace{\max_{s \in S} \frac{A_{d,s}}{b(s)}}_{m_d}$$

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

Variables: $A_{d,s}, m_d$

Restraints: $\sum_s A_{d,s} = |d|$

$$\sum_d A_{d,s} \le |s|$$

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

Additional Restraints

$$\left\{ \begin{array}{l} m_d \ge \frac{1}{b(s_1)} \cdot A_{d,s_1} \\ m_d \ge \frac{1}{b(s_2)} \cdot A_{d,s_2} \\ \vdots \end{array} \right.$$

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

$A_{d,s}$	s_1	s_2	s_3	Σ
d_1	100	0	0	100
d_2	2	2	1	5
d_3	2	98	0	100
Σ	≤ 500	≤ 100	≤ 1	

	SeqTime	SeqBand width	ParTime	ParBand width
d_1	1000	100	1000	100
d_2	61	82	40	125
d_3	1980	51	1960	51
Av	1864	121	1827	160
Worst case	1980	51	1960	51

Excursion: Linear Programming

- Linear Program (Linear Optimization)
- Given: m × 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 × n matrix A

  m-dimensional vector b

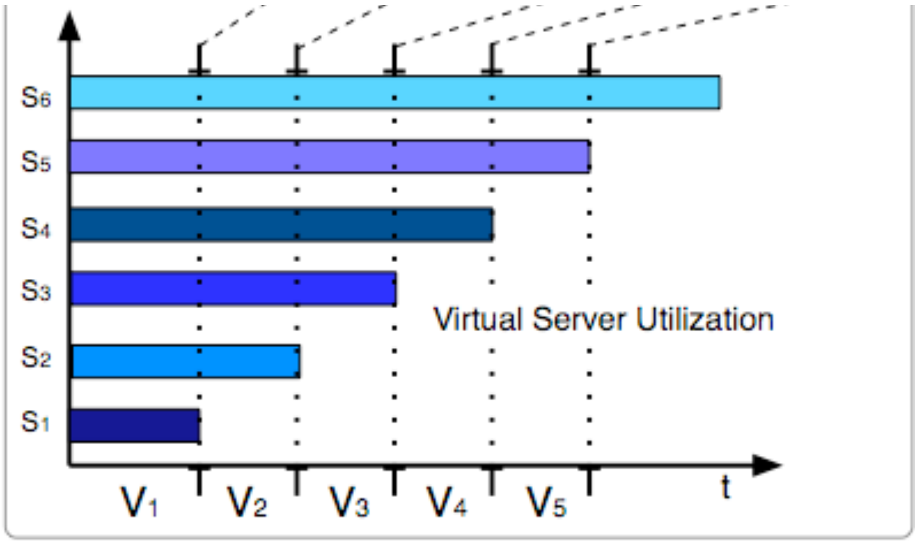
  n-dimensional vector c
- Find: n-dimensional vector $x=(x_1, ..., x_n)$
- such that
  - $x \geq 0$
  - $A x \leq b$
  - $z = c^T x$ is maximal

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

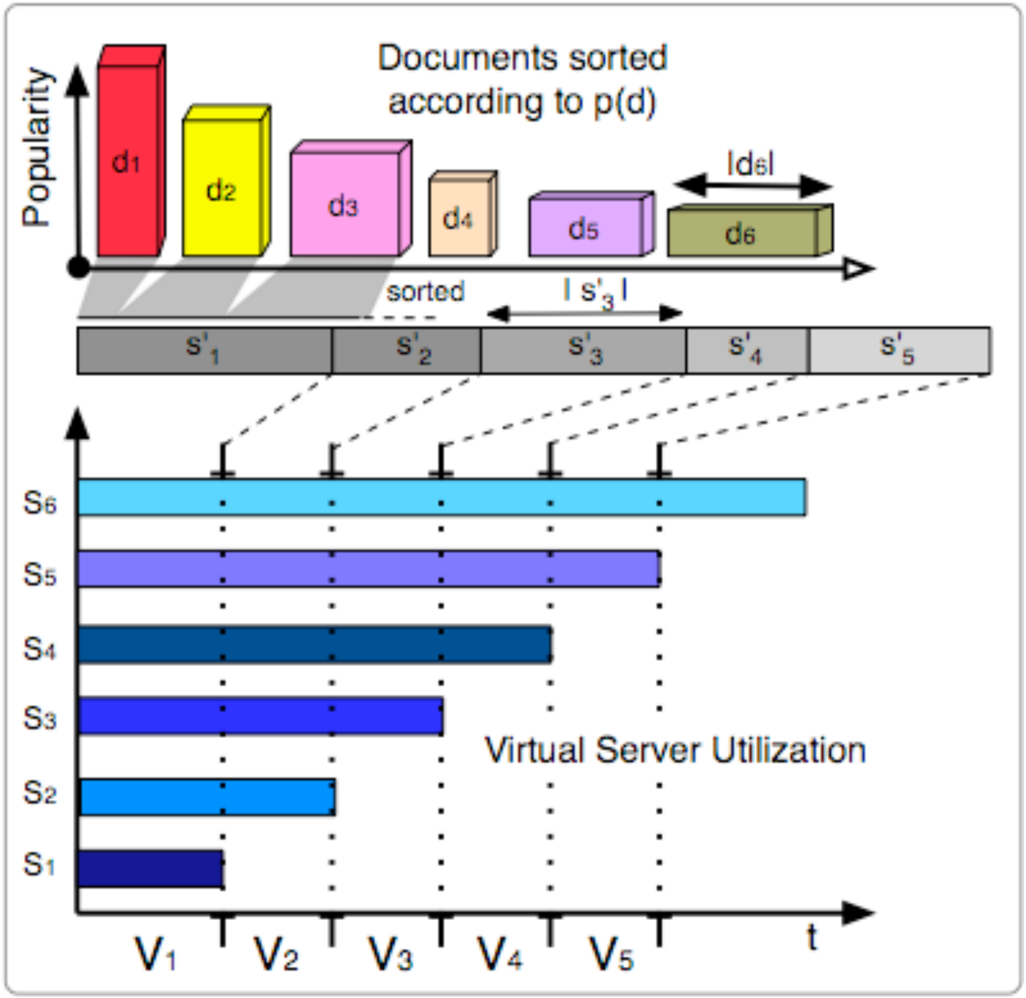
- **Reduce optimal solution for LP of ParTime to the optimal solution of LP of SeqTime**
  - Combining capacity of many disks in parallel
- **Define new sequential virtual servers $s'_1, \ldots, s'_m$**
  - Sort s_i such that $\frac{|s_j|}{b(s_j)} \leq \frac{|s_{j+1}|}{b(s_{j+1})}$
  - Server s'_j parallelizes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



$$t_j = \frac{|s_j|}{b(s_j)} - \sum_{i=1}^{j-1} t_i \qquad s'_j = b(s'_j) \cdot t_j$$

Solve the LP of AvSeqTime

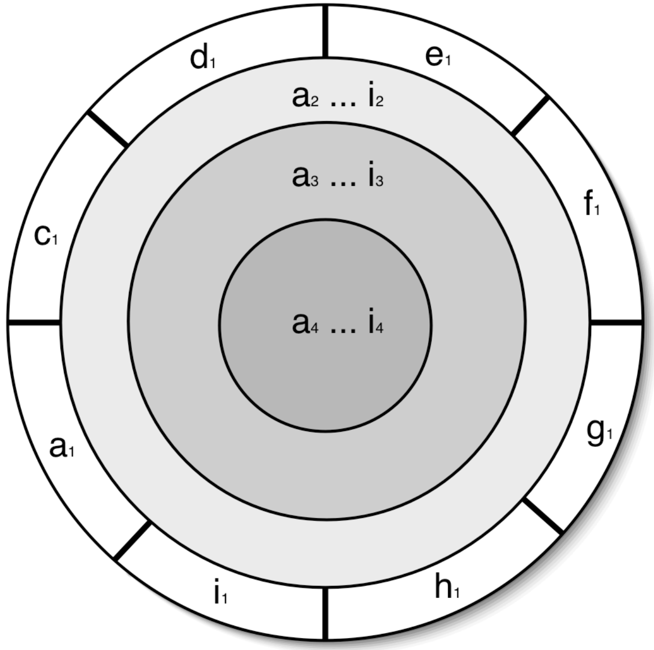
- **Simple optimal greedy solution**
- **Repeat until all documents are assinged:**
  - Assign most popular document on fastest sequential (virtual) server
  - Reduce the storage of the server by the document size and remove the document



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

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

Distributed Storage is Ubiquitous

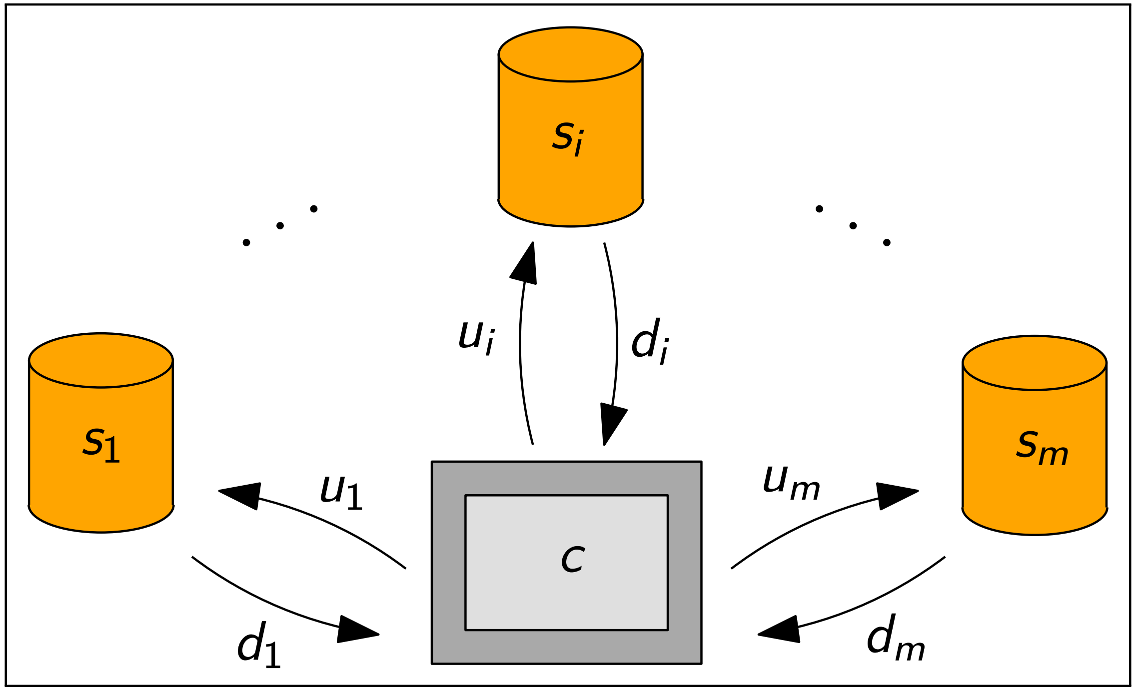
Motivation

- 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)
- 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

Distribution Network

Problem Setting

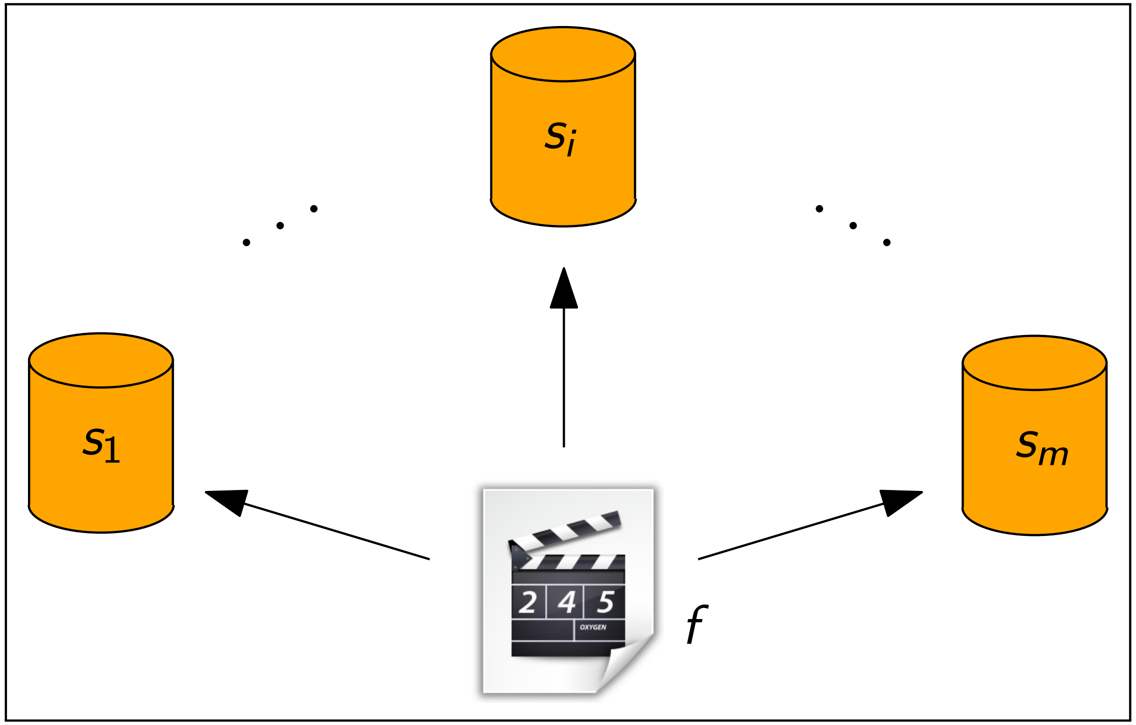


- Servers s_i with $i \in S = \{1, \ldots, m\}$, and one client c
- Each server s_i is characterized by its upload bandwidth u_i and download bandwidth d_i

Distribution Problem

Problem Setting

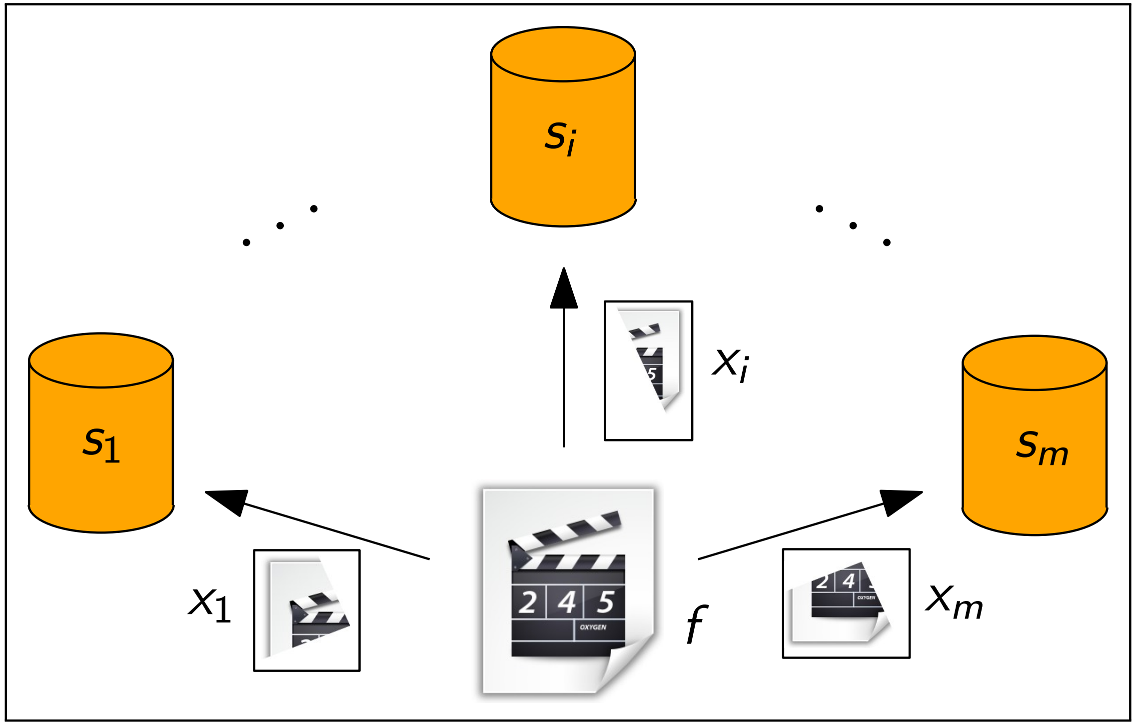
- How do we split a file f of size $|f|$ into fragments for the servers to minimize the time for one upload and n downloads?



Distribution 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?

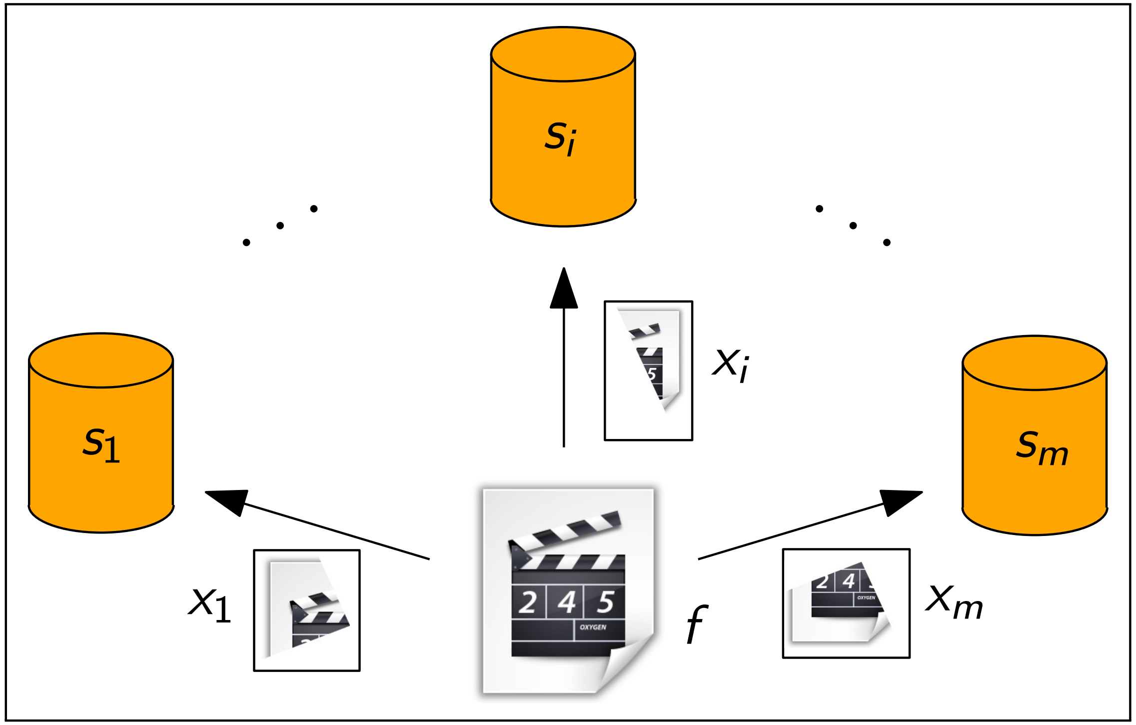


- Distribution vector $\mathbf{x} = (x_1, \ldots, x_m)$ with $\sum x_i = |f|$. x_i is the size of the fragment of server s_i.

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 $|f| = n = 1$.

Transfer Times

Problem Setting

The quality of a distribution $\mathbf{x}$ is given by the transfer times:

- Upload time:

$$t_u(\mathbf{x}) = \max_{i \in S} \left\{ \frac{x_i}{u_i} \right\}$$

Transfer Times

Problem Setting

The quality of a distribution **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:

$$t_d(\mathbf{x}) = \max_{i \in S} \left\{ \frac{x_i}{d_i} \right\}$$

Transfer Times

Problem Setting

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:

$$t_d(\mathbf{x}) = \max_{i \in S} \left\{ \frac{x_i}{d_i} \right\}$$

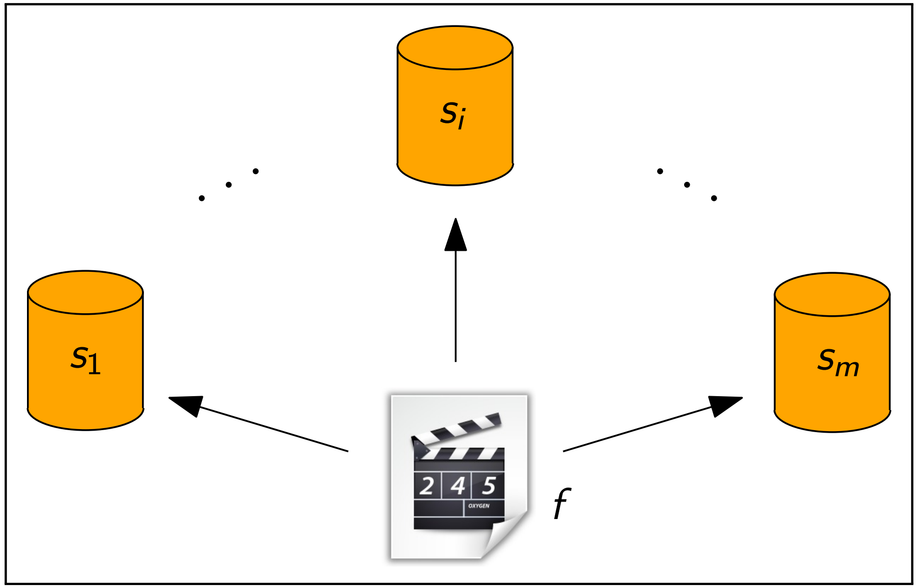
- Total time:

$$\mathrm{val}(\mathbf{x}) = t_u(\mathbf{x}) + (n \cdot)\, t_d(\mathbf{x})$$

Objective

Problem Setting

We are given a distribution network and a file f with size 1.



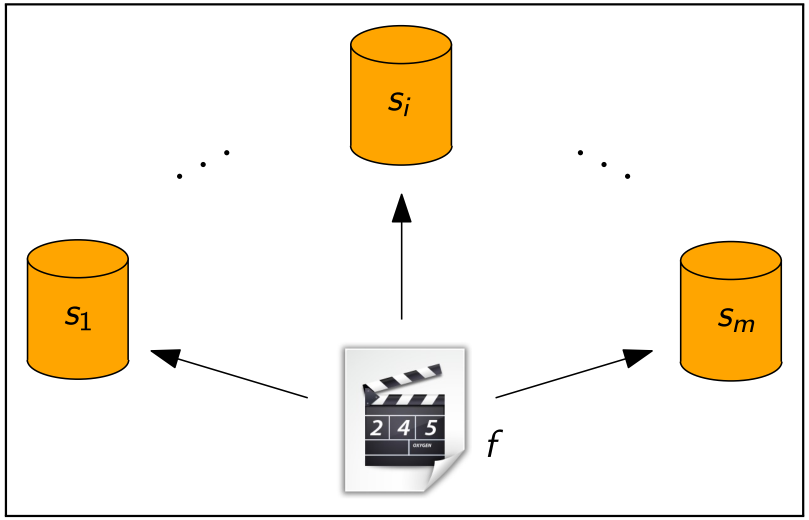
Objective

Find the optimal distribution $\mathbf{x}^*$ that minimizes the total time

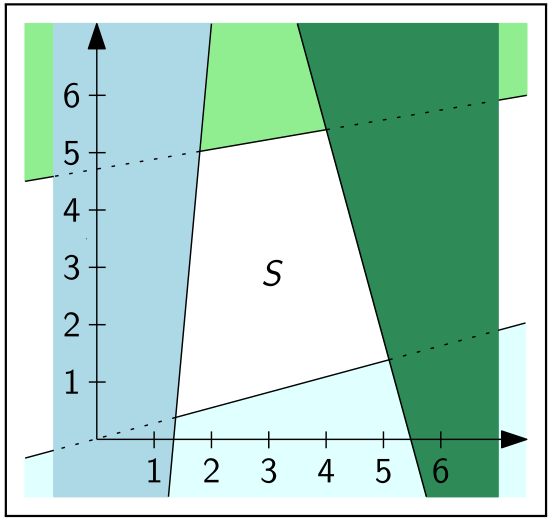
$$\mathrm{val}(\mathbf{x}) = t_u(\mathbf{x}) + t_d(\mathbf{x}) \quad .$$

Important Properties

Problem Setting



$\Rightarrow$



- The distribution problem can be formulated as linear program.
  $\Rightarrow$ Its solution space is a convex region.
  $\Rightarrow$ Every local minimum is global.

Non-Linear Optimization Problem

Linear Program

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

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

Linear Optimization Problem

Linear Program

Linear Program

$$
\begin{aligned}
\text{minimize} \quad & t_u + t_d \\
\text{subject to} \quad & \sum_{i=1}^{m} x_i = 1 \\
& \frac{x_i}{u_i} \leq t_u \qquad \text{for all } i \in \{1, \ldots, m\} \\
& \frac{x_i}{d_i} \leq t_d \qquad \text{for all } i \in \{1, \ldots, m\} \\
& x_i \geq 0 \qquad \text{for all } i \in \{1, \ldots, m\}
\end{aligned}
$$

$\boxed{\textbf{Dimension: } m + 2}$

Synopsis

The Distribution Problem as Flow Problem

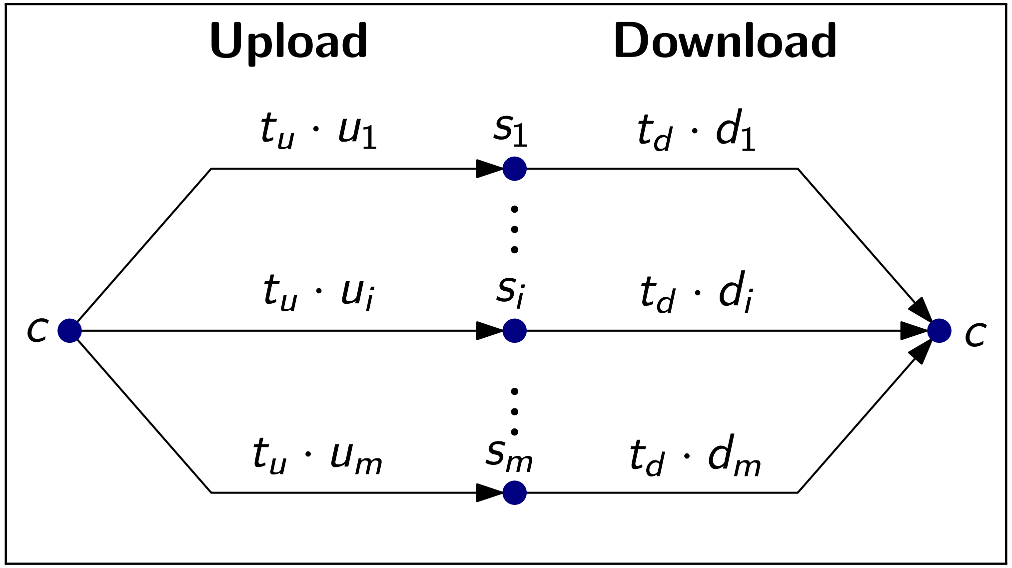
Maximum Flow in Distribution Problems

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

The Distribution Problem as Flow Problem

Maximum Flow in Distribution Problems

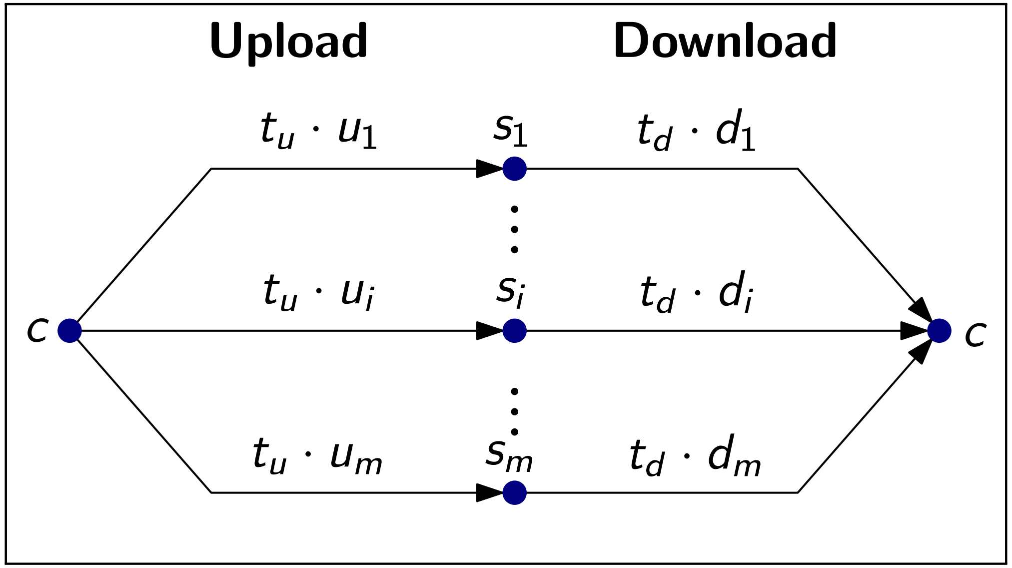
- Assume we are given upload time t_u and download time t_d



The Distribution Problem as Flow Problem

Maximum Flow in Distribution Problems

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

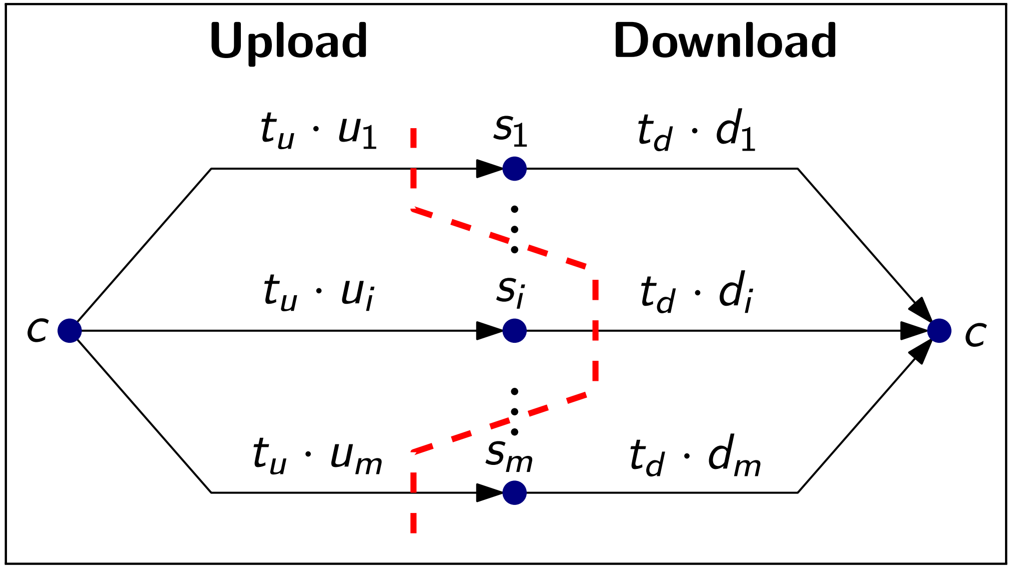


- The maximal amount of data that can be uploaded within t_u and downloaded within t_d corresponds to a maximum flow.

The Distribution Problem as Flow Problem

Maximum Flow in Distribution Problems

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

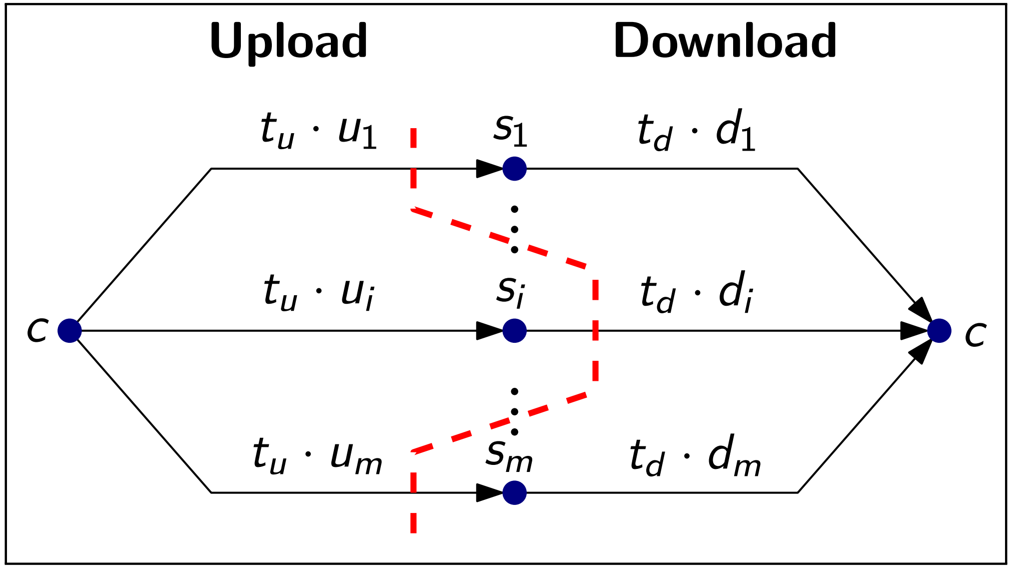


- The maximal amount of data that can be uploaded within t_u and downloaded within t_d corresponds to a maximum flow.
- Value of maximum flow f^* equals capacity of minimal cut

The Distribution Problem as Flow Problem

Maximum Flow in Distribution Problems

- 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

$$\mathrm{val}(f^*) = \sum_{i=1}^{m} \min\{t_u\, u_i,\ t_d\, d_i\}$$

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

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

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

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

- 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

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

- 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.

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

Total Data Function

Total Data Function

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

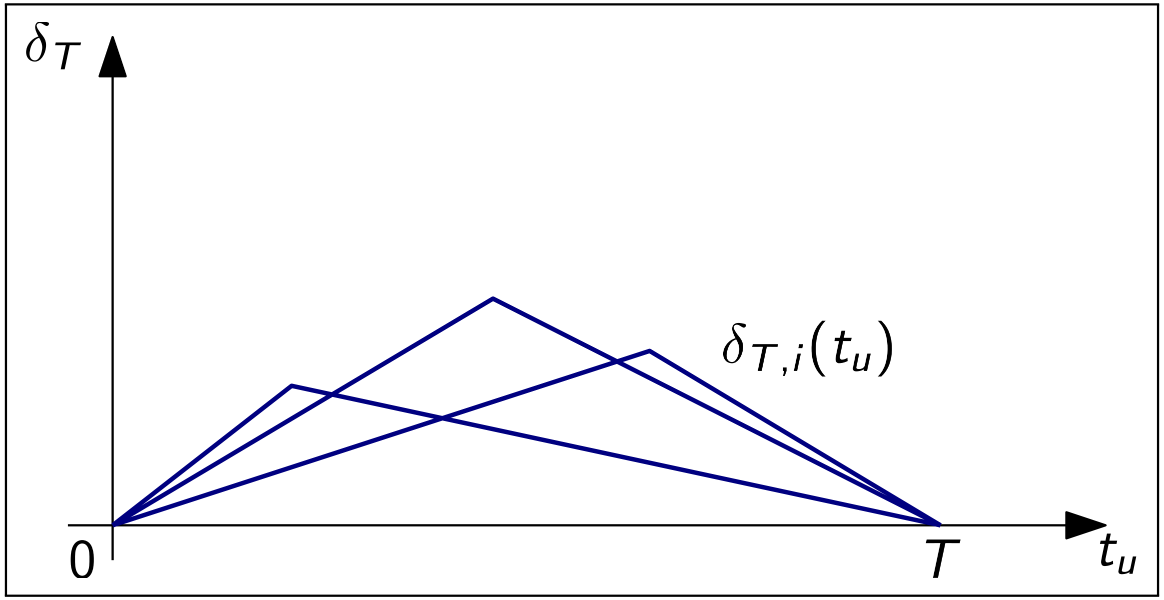
$$\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u\, u_i,\ (T - t_u) \cdot d_i\}$$

Total Data Function

Total Data Function

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

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

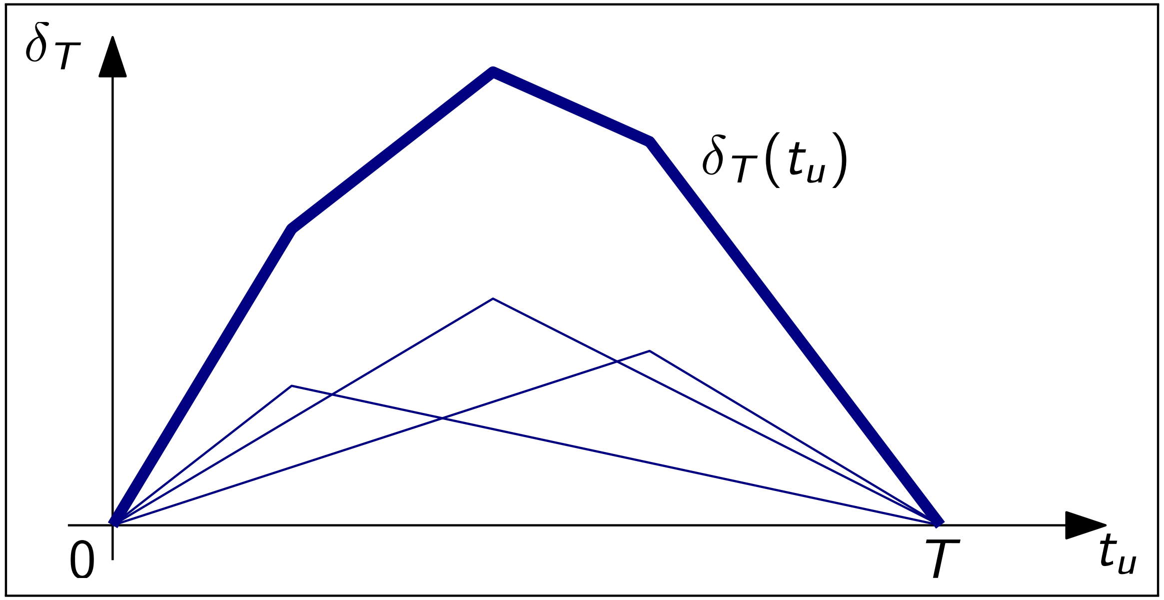


Total Data Function

Total Data Function

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

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

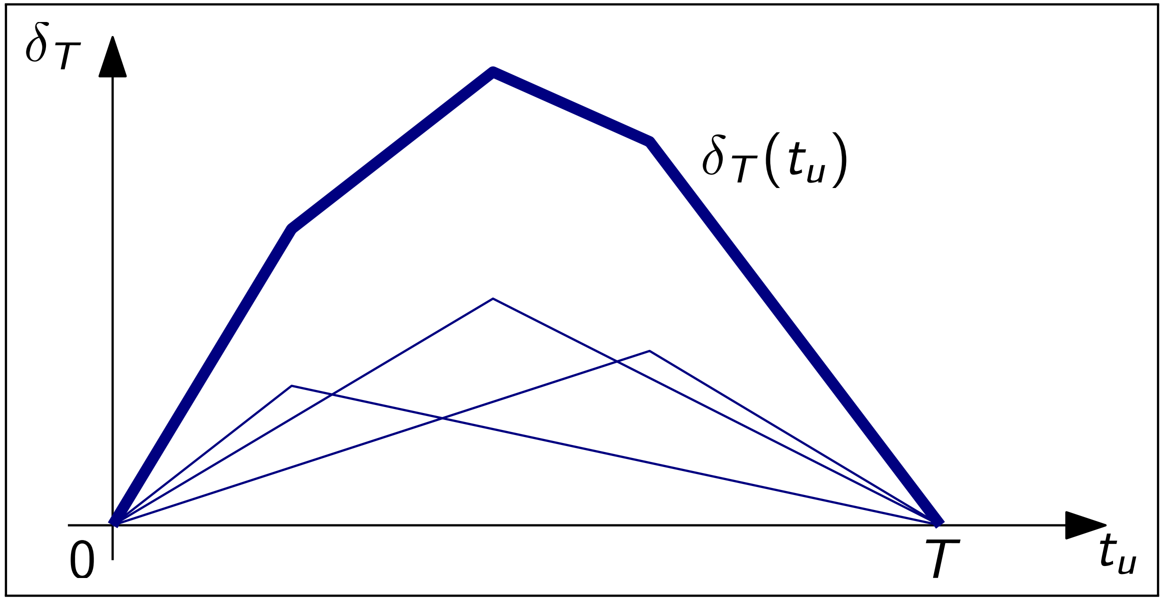


Total Data Function

Total Data Function

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

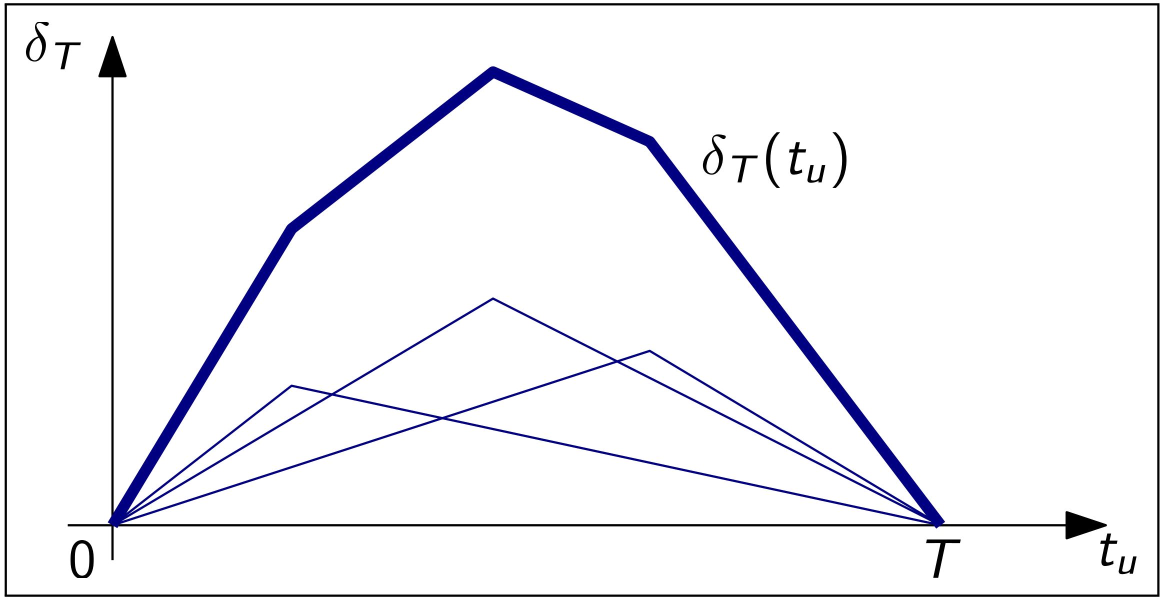
$$\delta_T(t_u) = \sum_{i=1}^{m} \min\{t_u\, u_i,\ (T - t_u)\cdot d_i\}$$



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

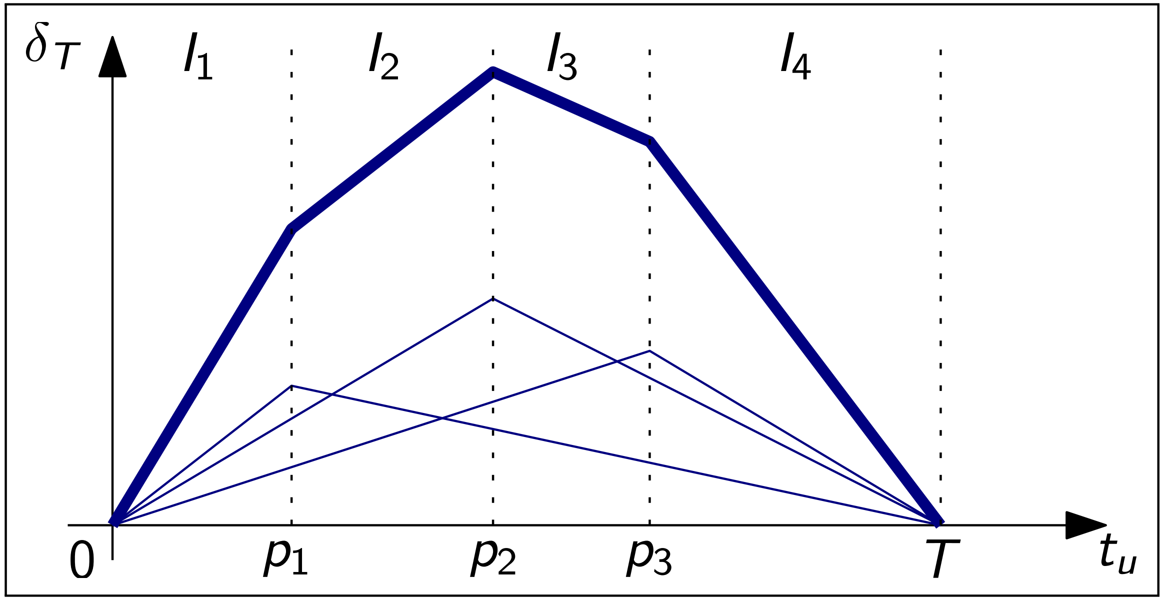
Evaluation

The Total Data Function



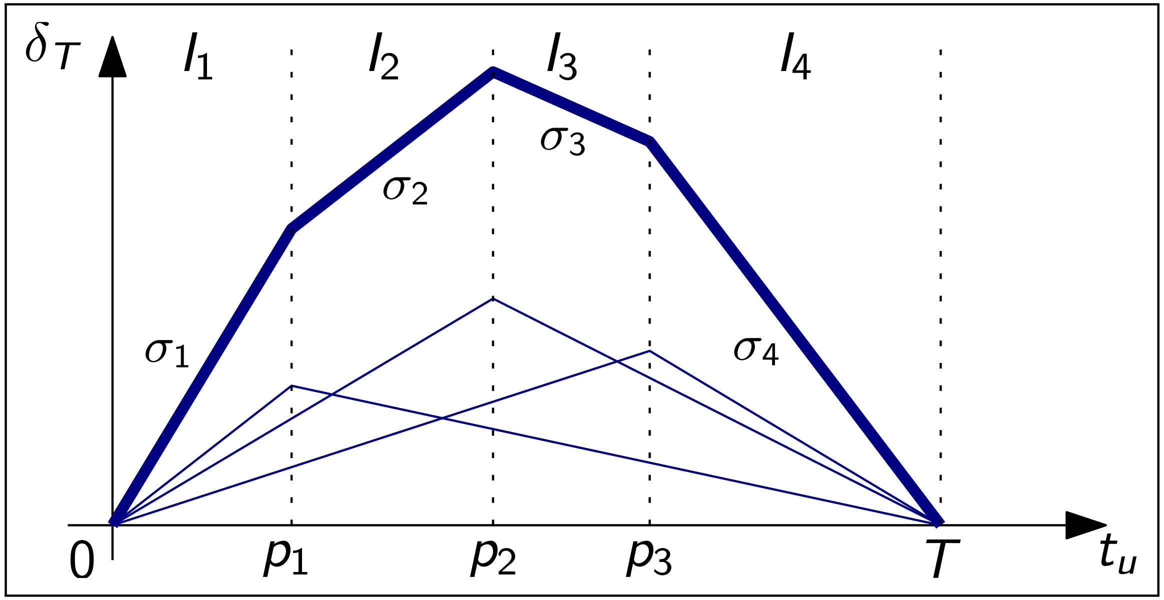
Evaluation

The Total Data Function



Evaluation

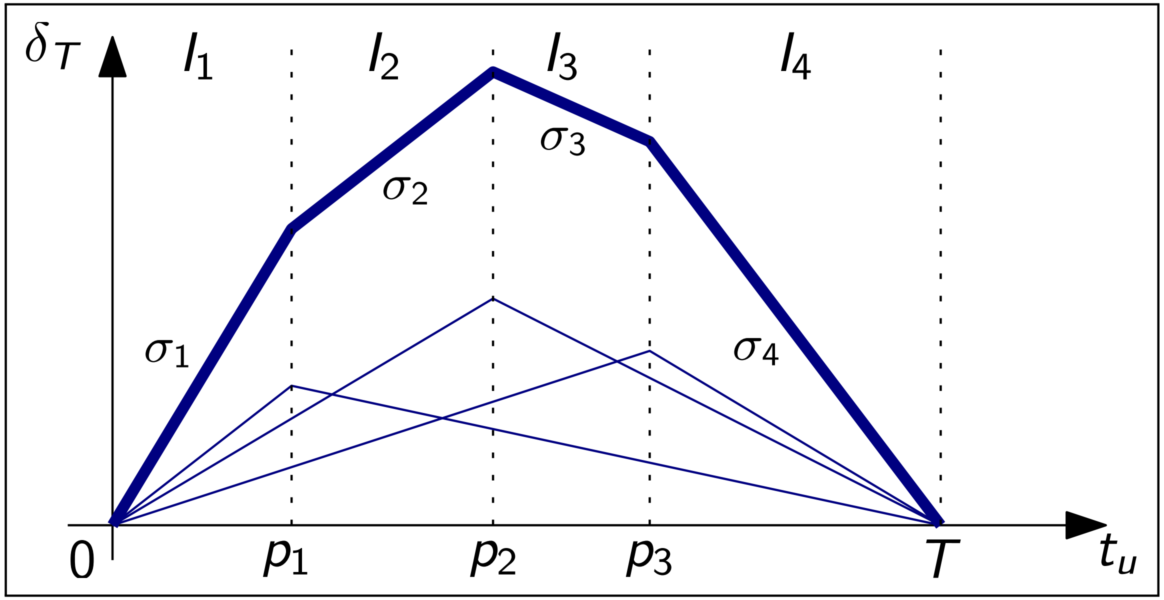
The Total Data Function



σ_i is slope in interval I_i.

Evaluation

The Total Data Function



σ_i is slope in interval I_i.

Recursion formula

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

Scaling the Total Data Function

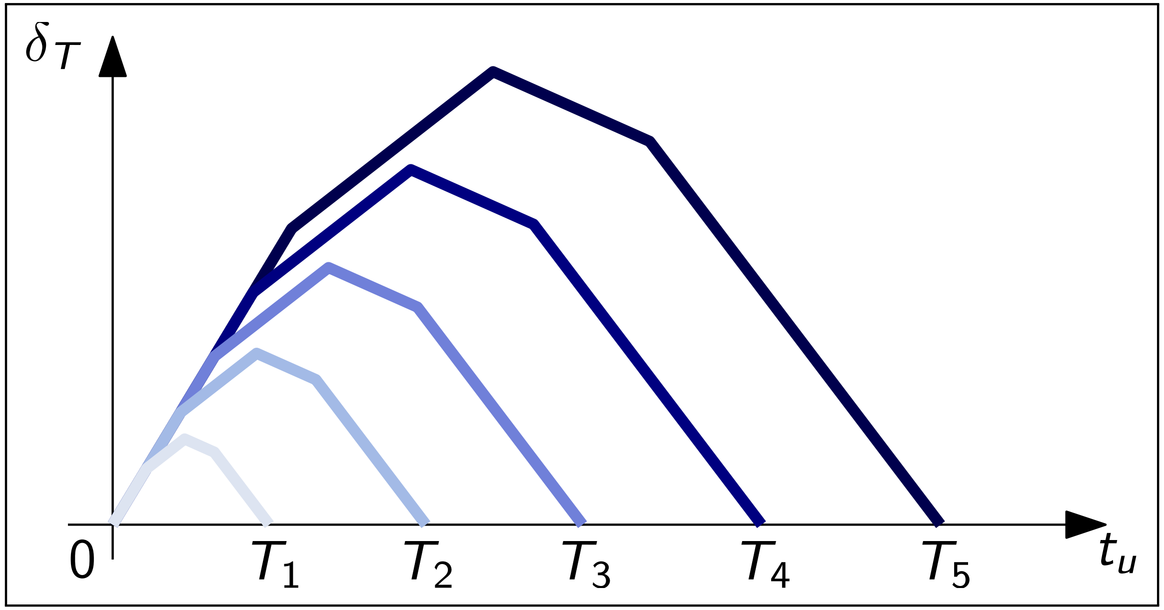
Total Data Function in 3D

- 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.

Scaling the Total Data Function

Total Data Function in 3D

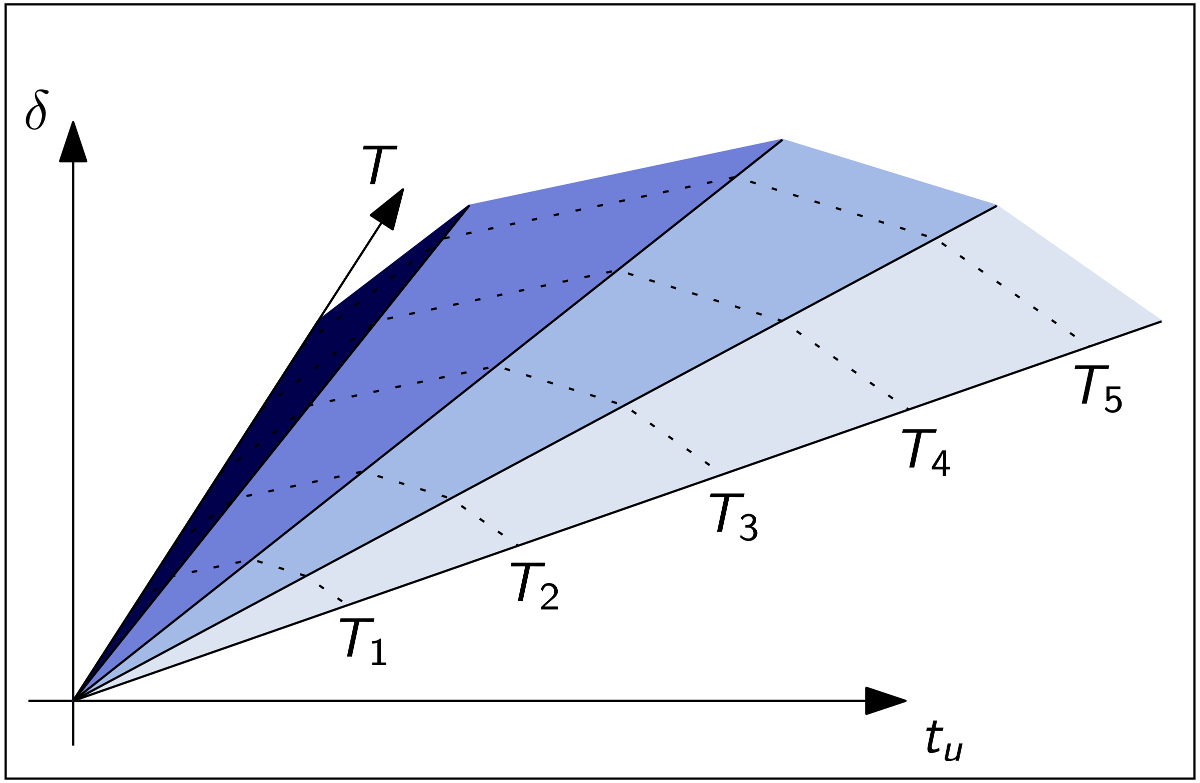
- The total data function $\delta_T(t_u)$ determines how much data can be uploaded to and downloaded again from the servers within time T for different values of t_u.
- Varying values of T only scale the total data function



Two-Variable Total Data Function

Total Data Function in 3D

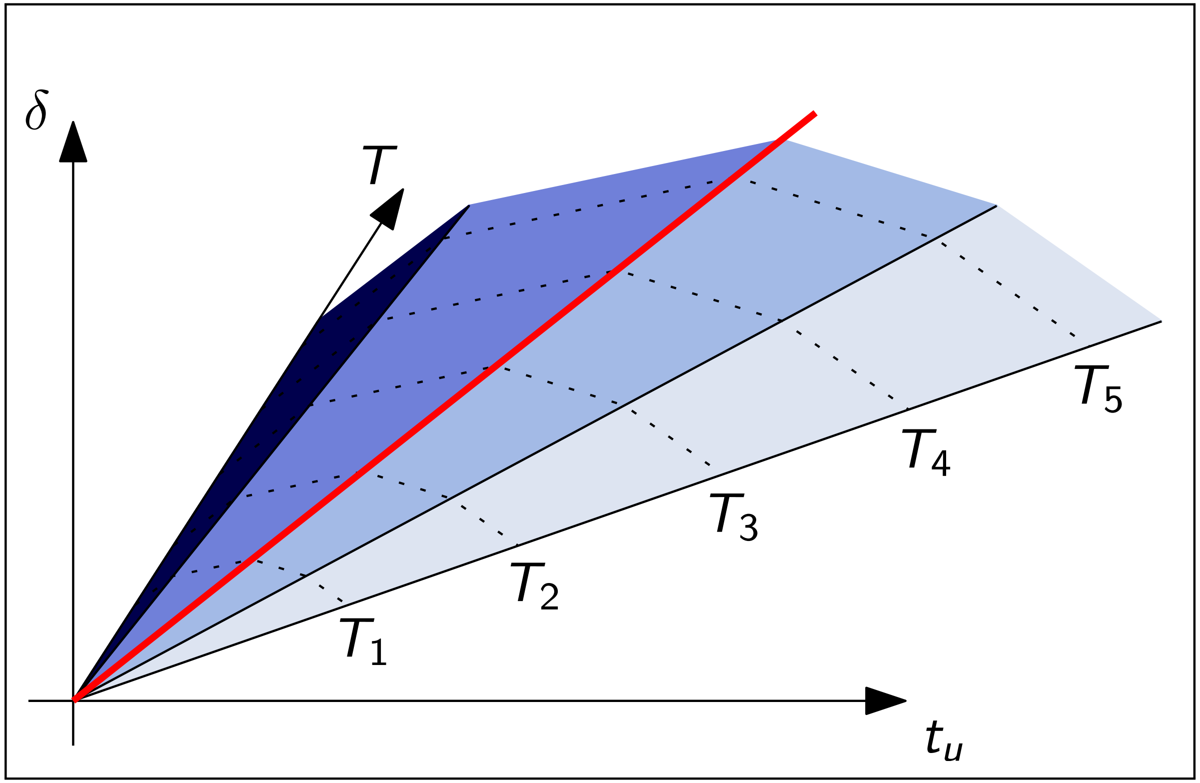
- Interpret $\delta_T(t_u)$ as two-variable function $\delta(T, t_u)$ now



Two-Variable Total Data Function

Total Data Function in 3D

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

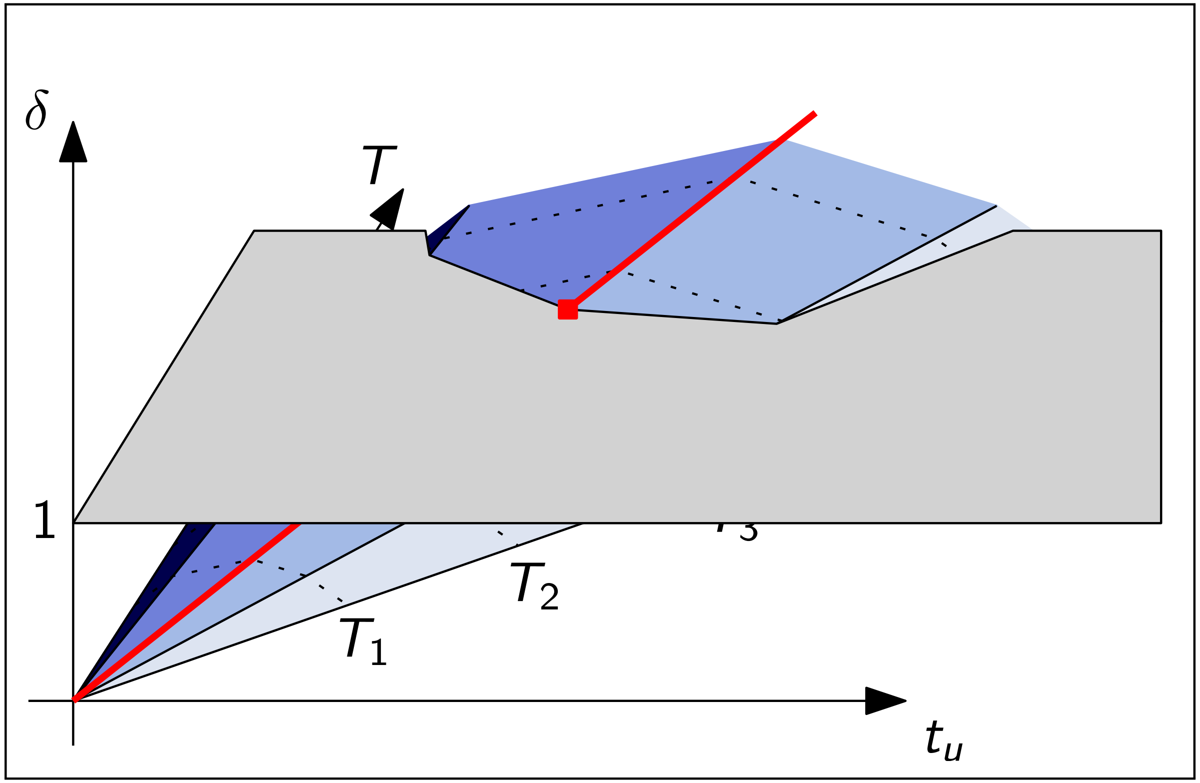


- To find the minimal value of T with $\delta(T, t_u) \geq 1$ for some t_u, establish straight line through the highest point of $\delta(T, t_u)$

Two-Variable Total Data Function

Total Data Function in 3D

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

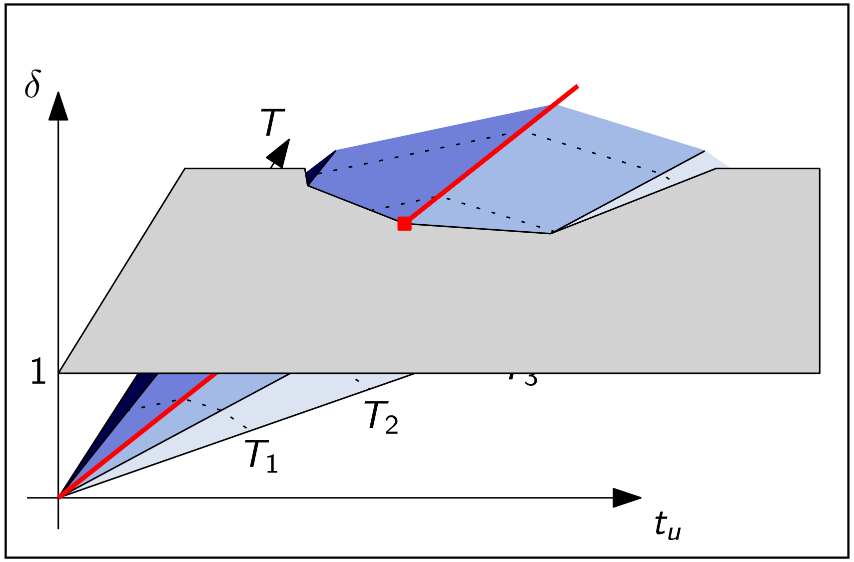


- To find the minimal value of T with $\delta(T, t_u) \geq 1$ for some t_u, establish straight line through the highest point of $\delta(T, t_u)$
- Determine where it intersects the plane $\delta = 1$

Determining the Actual Distribution

The Algorithm

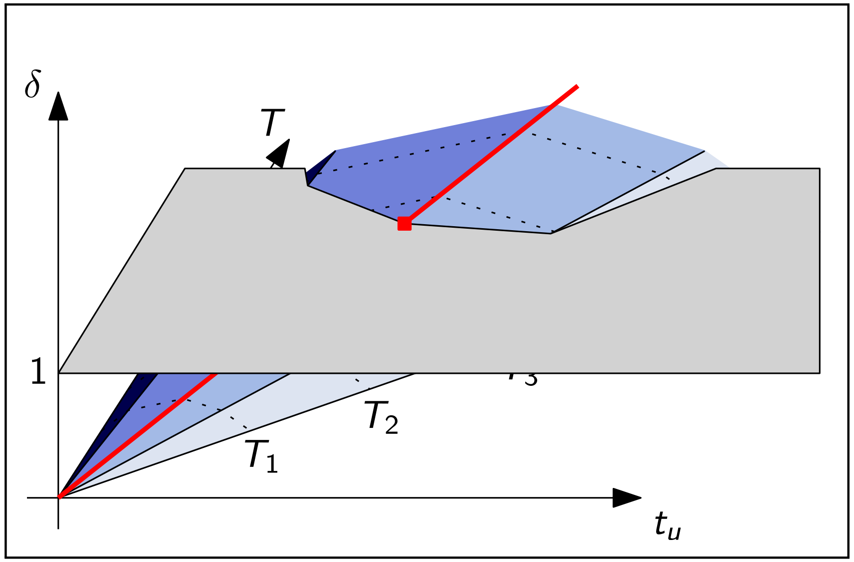
- We determine the minimal value of T as depicted.



Determining the Actual Distribution

The Algorithm

- We determine the minimal value of T as depicted.

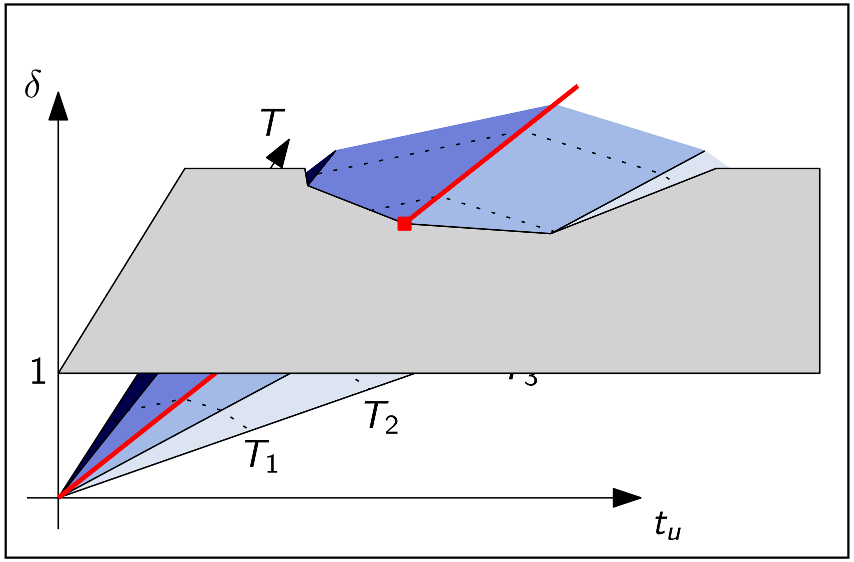


- The decomposition of T into t_u and t_d is given by the coordinates of the intersection point.

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.

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

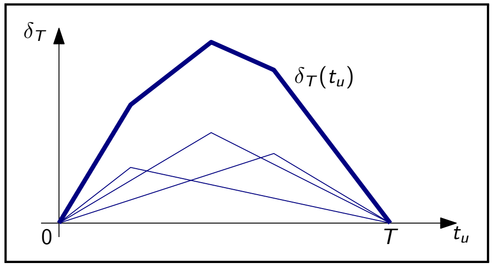
$$\sum_{i=1}^{m} \min\{t_u\, u_i,\ t_d\, d_i\} = 1$$

we know that $\sum x_i = 1$ and thus $\mathbf{x}$ is a valid distribution.

Runtime Complexity

The Algorithm

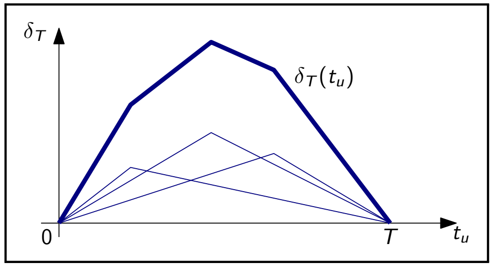
- Evaluating the total data function for a fixed value of T runs in $\mathcal{O}(m \log m)$ steps.



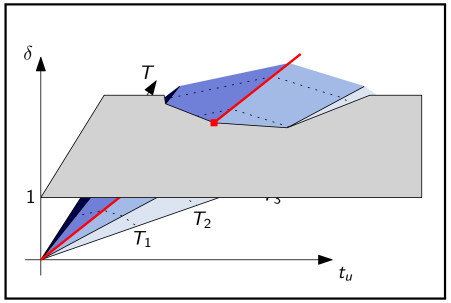
Runtime Complexity

The Algorithm

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



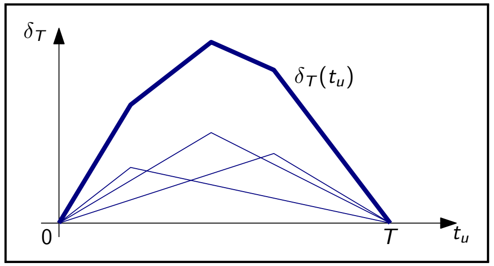
- Intersecting the straight line with the plane $\delta = 1$ is possible in constant time.



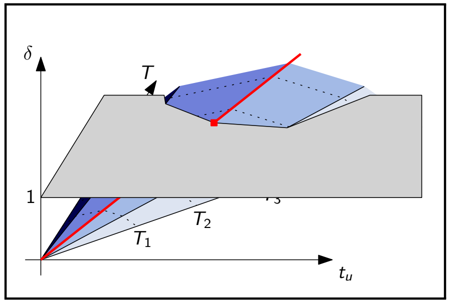
Runtime Complexity

The Algorithm

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



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



Runtime complexity: $\mathcal{O}(m \log m)$

Synopsis

Summary

- Formal introduction of a new distribution problem

Summary

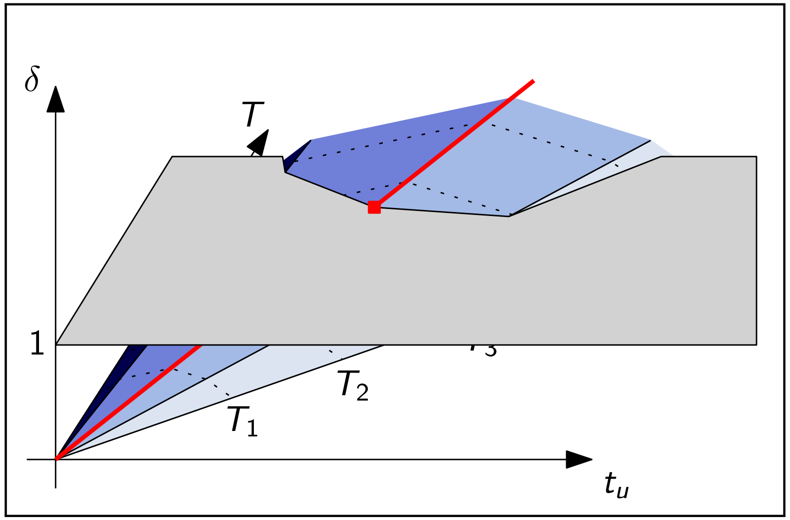
- Formal introduction of a new distribution problem
- Formulation as linear program

Summary

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

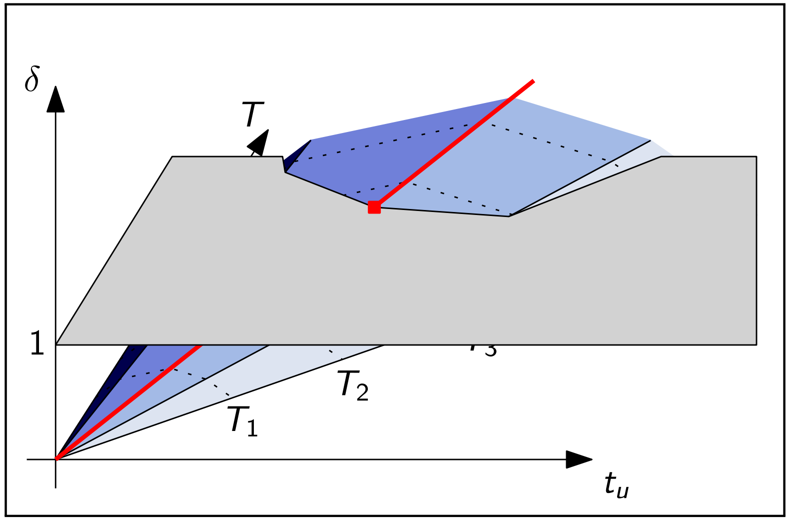
Summary

- 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

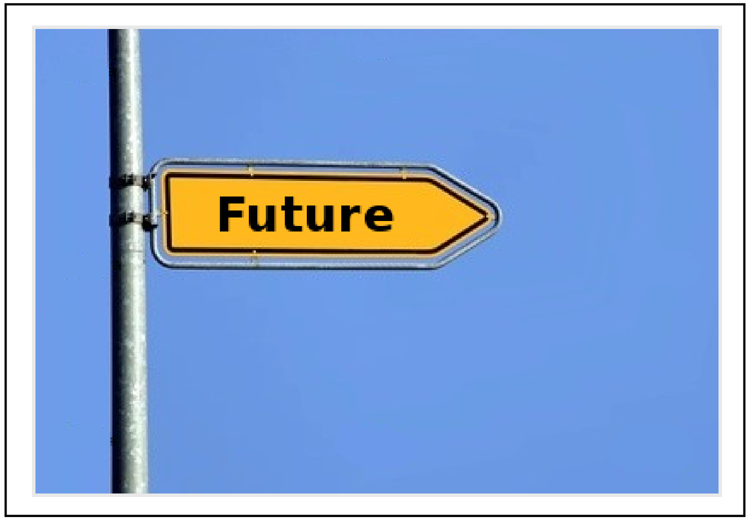


Summary

- 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

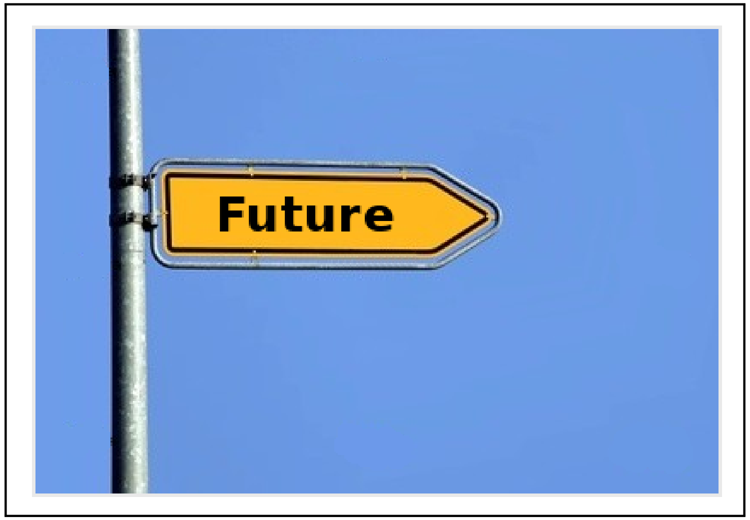


- Implementation of our ideas into real-world applications

Future Directions

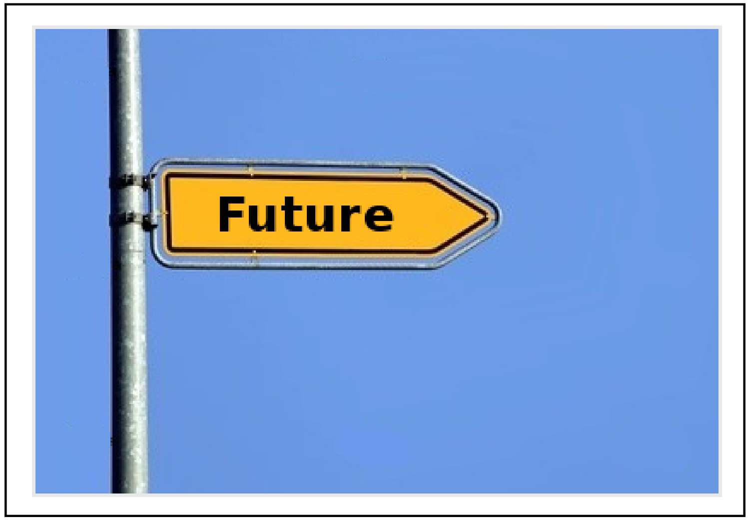
- Implementation of our ideas into real-world applications
- Cope with non-static, fluctuating bandwidths

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?