DAAD Summerschool Curitiba 2011

Aspects of Large Scale High Speed Computing Building Blocks of a Cloud

Storage Networks

4: Distributed Heterogeneous Hash Tables

Christian Schindelhauer

Technical Faculty

Computer-Networks and Telematics

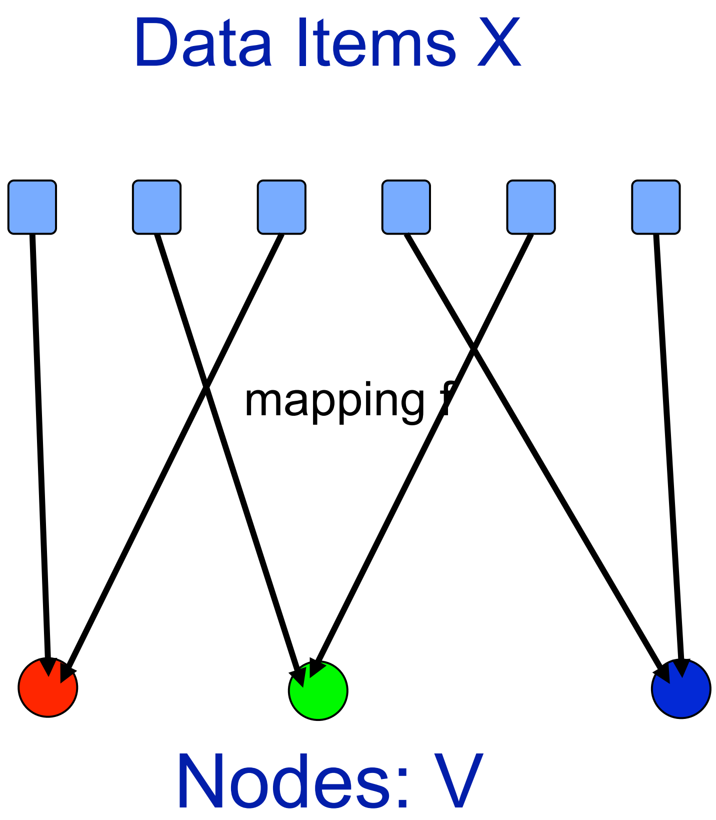
University of Freiburg

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

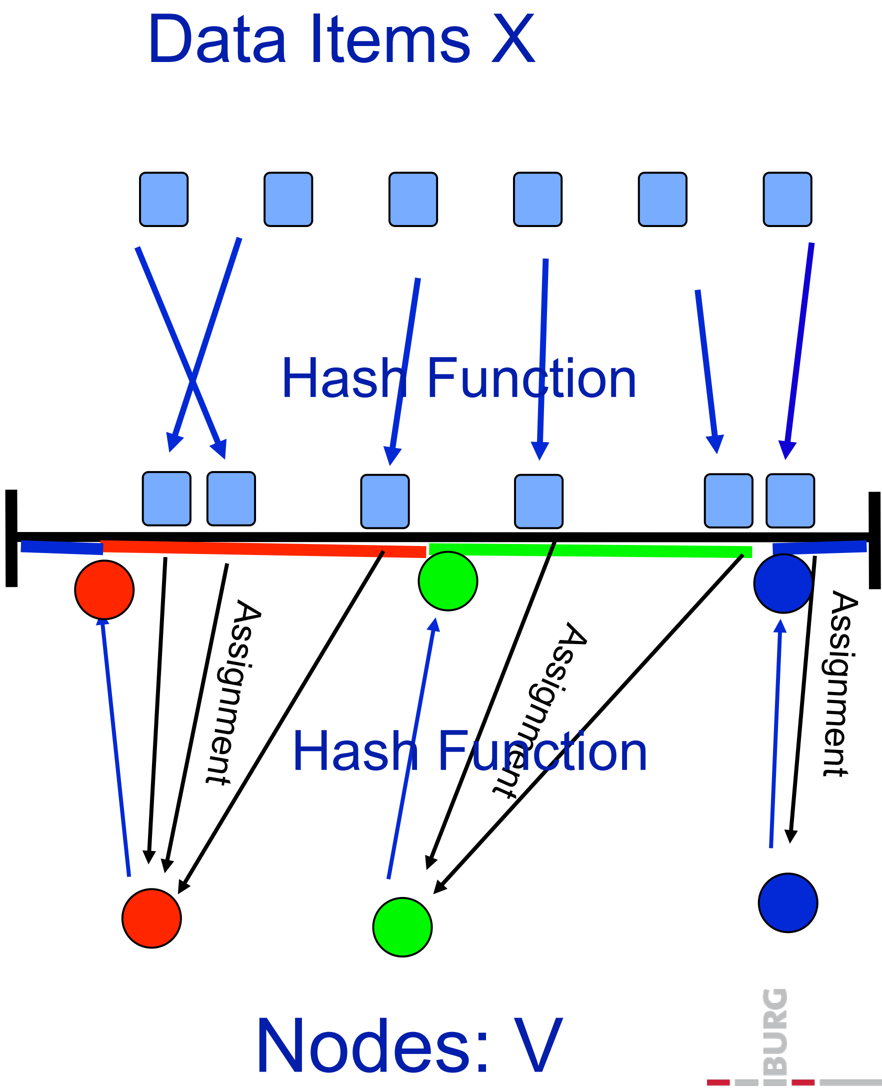
The Uniform Problem

- **Given**
  - a dynamic set of n nodes $V = \{v_1, \ldots, v_n\}$
  - data elements $X = \{x_1, \ldots, x_m\}$
- **Find**
  - a mapping $f_V : X \rightarrow V$
- **With the following properties**
  - The mapping is simple
    - $fV(x)$ be computed using V and x
    - without the knowledge of $X \backslash \{x\}$
  - Fairness:
    - $|f_V^{-1}(v)| \approx |f_V^{-1}(v)|$
  - Monotony: Let $V \subset W$
    - For all $v \in V$: $f_V^{-1}(v) \supseteq f_W^{-1}(v)$
- **where $f_V^{-1}(v) := \{x \in X : f_V(x) = v\}$**



Distributed Hash Tables
THE Solution for the Uniform case

- **“Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web”,**
  - David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, Rina Panigrahy, STOC 1997
  - Present a simple solution
- **Distributed Hash Table**
  - Chooose a space M = [0,1[
  - Map nodes v to M via hash function
    - $h : V \to M$
  - Map documents and servers to an interval
    - $h : X \to M$
  - Assign a document to the server which minimizes the distance in the interval
  - $f_V(x) = \text{argmin}\{v \in V: (h(x)-h(v)) \bmod 1\}$
    - where $x \bmod 1 := x - \lfloor x \rfloor$

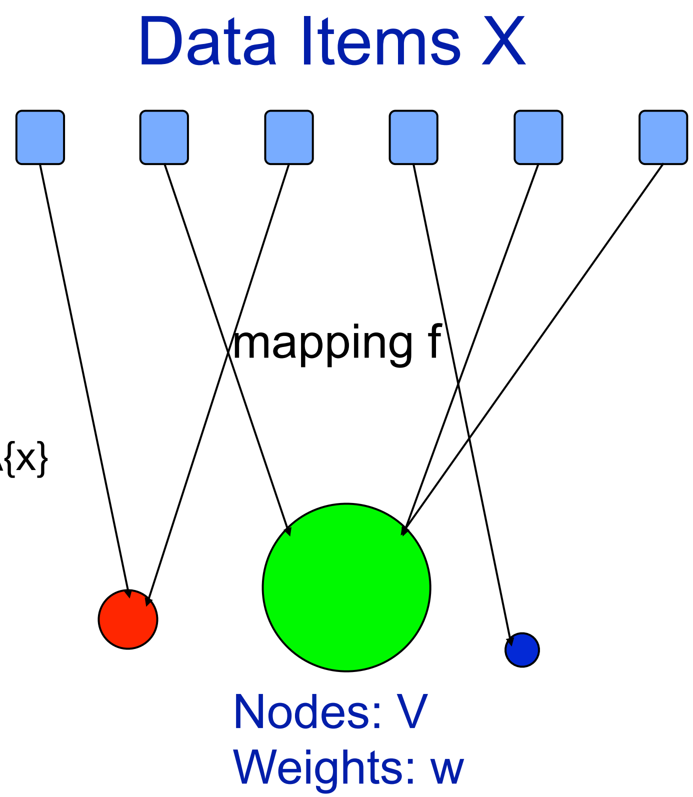


The Performance of Distributed Hash Tables

- **Theorem**
  - Data elements are mapped to node i with probability $p_i = 1/|V|$, if the hash functions behave like perfect random experiments
- Balls into bins problem
  - Expected ratio $\max(p_i)/\min(p_i) = \Omega(\log n)$
- **Solutions**:
  - Use O(log n) **copies** of a node
  - **Principle of multiple choices**
    - check at some O(log n) positions and choose the largest empty interval for placing a node,
  - **Cookoo-Hashing**
    - every node chooses among two possible position

The Heterogeneous Case

- **Given**
  - a dynamic set of n nodes $V = \{v_1, \ldots, v_n\}$
  - dynamic weights $w : V \rightarrow R+$
  - dynamic set of data elements $X = \{x_1,\ldots,x_m\}$
- **Find a mapping $f_{w,V} : X \rightarrow V$**
- **With the following properties**
  - The mapping is simple
    - $f_{w,V}(x)$ be computed using V, x, w without the knowledge of $X \setminus \{x\}$
  - Fairness: for all $u,v \in V$:
    - $| f_{w,V}^{-1}(u)|/w(u) \approx | f_{w,V}^{-1}(v)|/w(v)$
  - Consistency:
    - Let $V \subset W$: For all $v \in V$:
      - $f_{w,V}^{-1}(v) \supseteq f_{w,W}^{-1}(v)$
    - Let for all $v \in V \setminus \{u\}$: $w(v) = w'(v)$ and $w'(u)>w(u)$:
      - for all $v \in V \setminus \{u\}$: $f_{w,V}^{-1}(v) \supseteq f_{w',V}^{-1}(v)$ and $f_{w,V}^{-1}(u) \subseteq f_{w',V}^{-1}(u)$
- **where $f_{w,V}^{-1}(v) := \{ x \in X : f_{w,V}(x) = v \}$**



Some Application Areas

- Proxy Caching
  - Relieving hot spots in the Internet
- Mobile Ad Hoc Networks
  - Relating ID and routing information
- Peer-to-Peer Networks
  - Finding the index data efficiently
- Storage Area Networks
  - Distributing the data on a set of servers

Application
Peer-to-Peer Networks

- Peer-to-Peer Network:
  - decentralized overlay network delivering services over the Internet
  - no client-server structure
    - example: Gnutella
- Problem: Lookup in first generation networks very slow
- Solution:
  - Use an efficient data structure for the links and
  - map the keys to a hash space
- Examples:
  - **CAN**
    - maps keys to a d-dimensional array
    - builds a toroidal connection network,
      - where each peer is assigned to rectangular areas
  - **Chord**
    - maps keys and peers to a ring via DHT
    - establishes binary search like pointers on the ring

Application
Storage Networks

- Distribute data over a set of hard disks
  - Nodes = hard disks
  - Data items = blocks
- Problem
  - Place copies of blocks for redundancy
  - If a hard disk fails other hard disk carry the information
  - Add or remove hard disks without unnecessary data movement
  - Hard disks may have different sizes

Storage Network Architecture

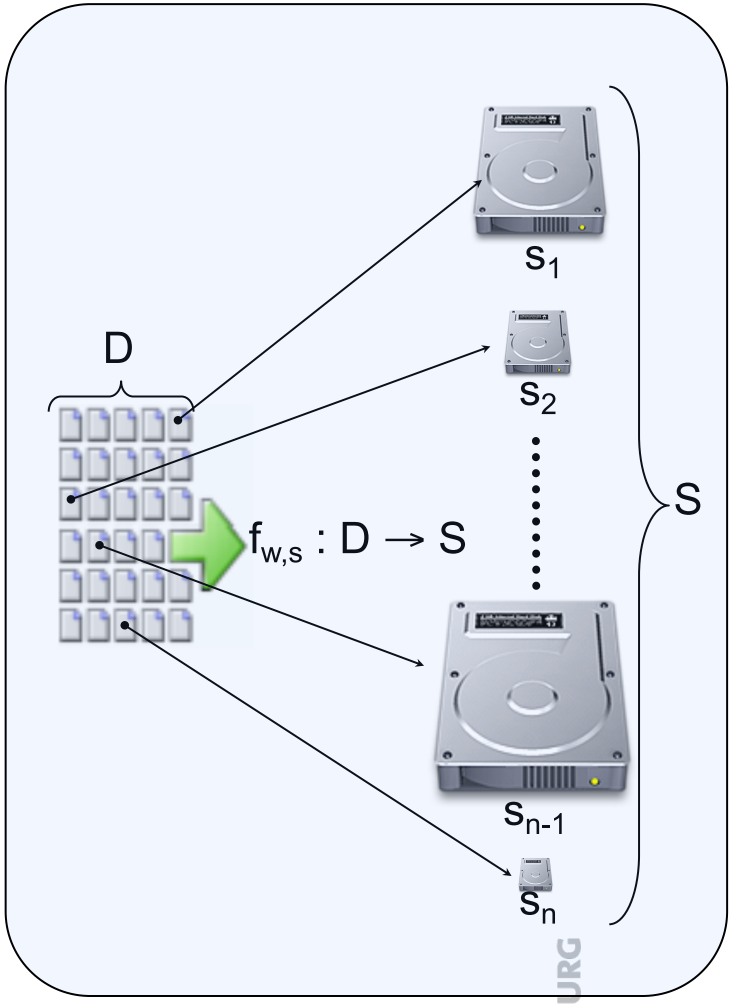
- Avoid server based architectures
  - Assignment of data is not flexible enough
  - High local storage concentration (for LAN traffic reduction)
  - Low availability of free capacity
- Basic distributed storage network concept
  - Combine all available disks into a single virtual one
  - Server independent existence of storage

Challenges in Storage Networks

- Heterogeneity
  - hard disks typically differ in capacity and speed
- Popularity
  - some data is popular and other not (e.g. movies, music :-)
  - their popularity rank varies over time
- Consistency
  - system changes by adding or re-placing/moving
  - preserving a fair share rate
  - only necessary data replacements must be done
- Availability
  - hard disks may fail, but data should not!
- Performance

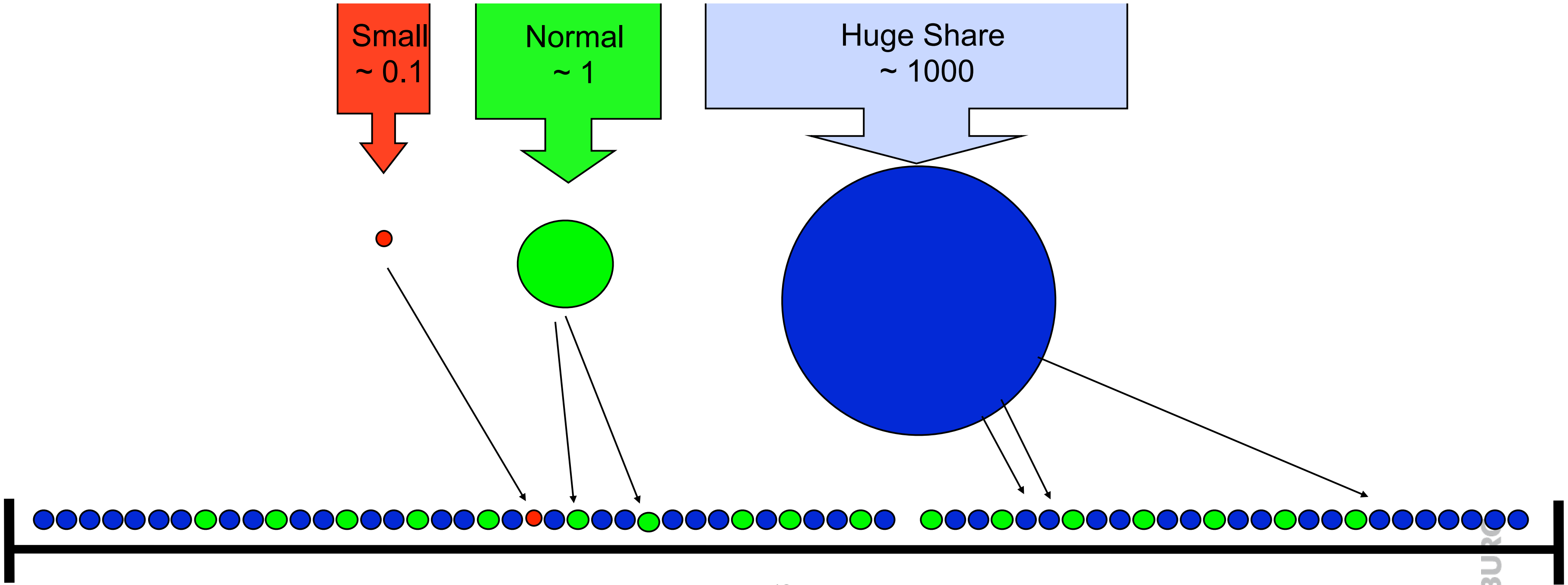
The Heterogeneous Case

- **Given**
  - a dynamic set of n nodes $V = \{v_1, \dots, v_n\}$
  - dynamic weights $w : V \to \mathbf{R}^+$
  - dynamic set of data elements $X = \{x_1,\dots,x_m\}$
- **Find a mapping $f_{w,V} : X \to V$**
- **With the following properties**
  - The mapping is **simple**
    - $f_{w,V}(x)$ be computed using V, x, w
    - without the knowledge of $X \setminus \{x\}$
  - **Fairness**: for all $u,v \in V$:
    - $|f_{w,V}^{-1}(u)|/w(u) \approx |f_{w,V}^{-1}(v)|/w(v)$
  - **Consistency**:
    - minimal replacements to preserve the data distribution
- **where $f_{w,V}^{-1}(v) := \{ x \in X : f_{w,V}(x) = v \}$**



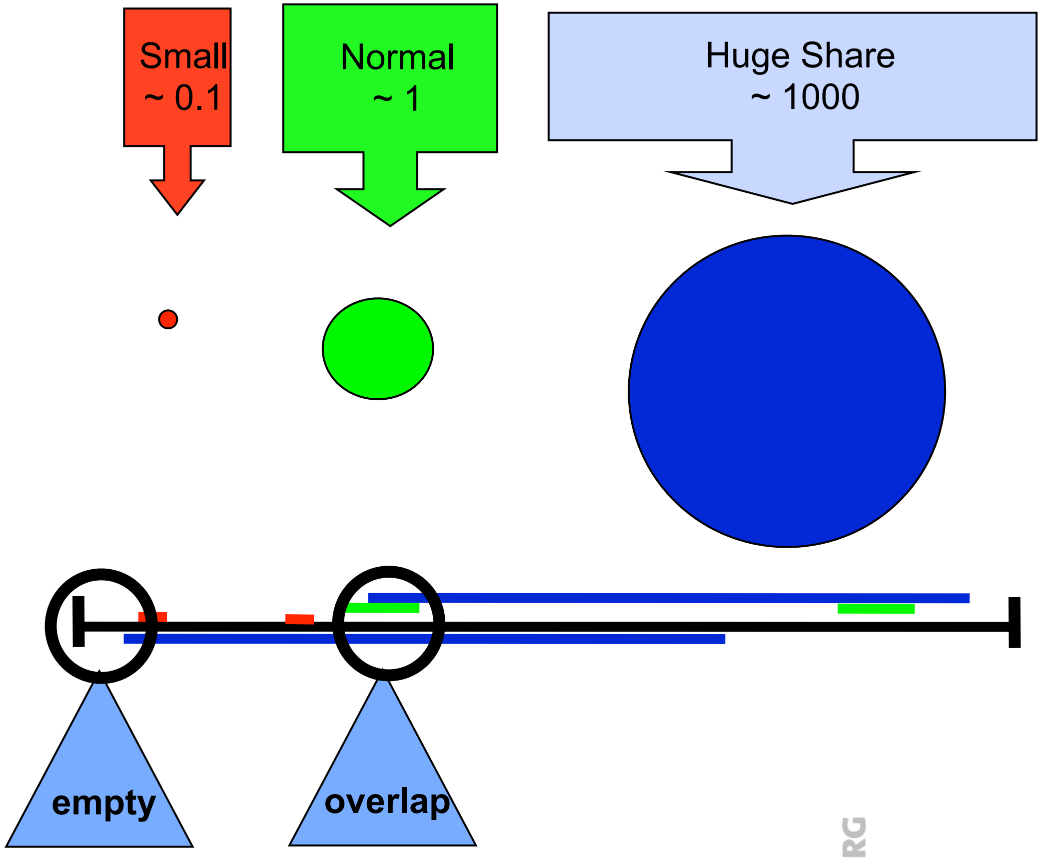
The Naive Approach to DHT

- Use $\left\lceil \frac{w_i}{\min_{j \in V}\{w_j\}} \right\rceil$ copies for each node w_i
- This is not feasible, if $\max_{j \in V}\{w_j\} / \min_{j \in V}\{w_j\}$ is too large
- Furthermore, inserting nodes with small weights increases the number of copies of all nodes.



SIEVE: Interval based consistent hashing

- **Interval based approach**
  - Brinkmann, Salzwedel, and Scheideler, SPAA 2000
- **Map nodes to random intervals (via hash function)**
  - interval length proportional to weight
- **Map data items to random positions (via hash function)**
- **Two problems**
  - What to do if intervals overlap?
  - What to do if the unions of intervals do not overlap the hash space M?

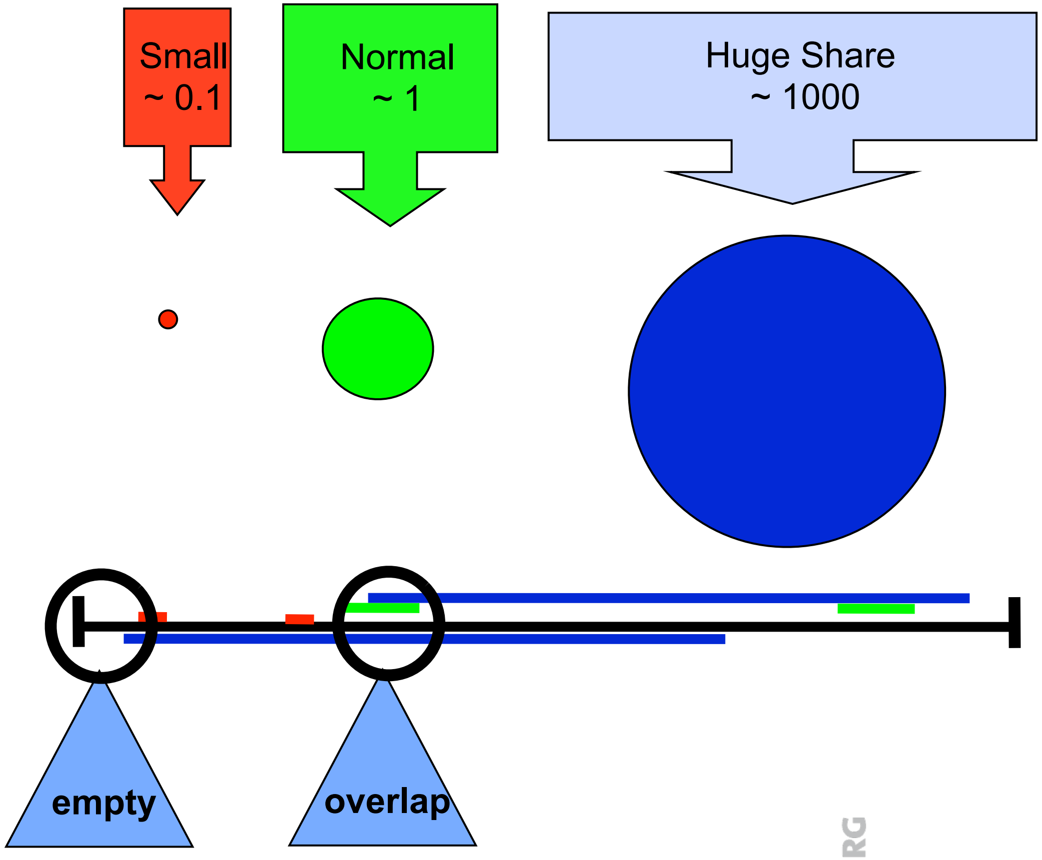


SIEVE: Interval based consistent hashing

1. **What to do if intervals overlap?**
   - Uniformly choose random candidate from the overlapping intervals
2. **What to do if the unions of intervals do not overlap the hash space M?**
   - Increase all intervals by a constant factor (stretch factor)
   - Use O(log n) copies of all nodes
     - resulting in O(n log n) intervals

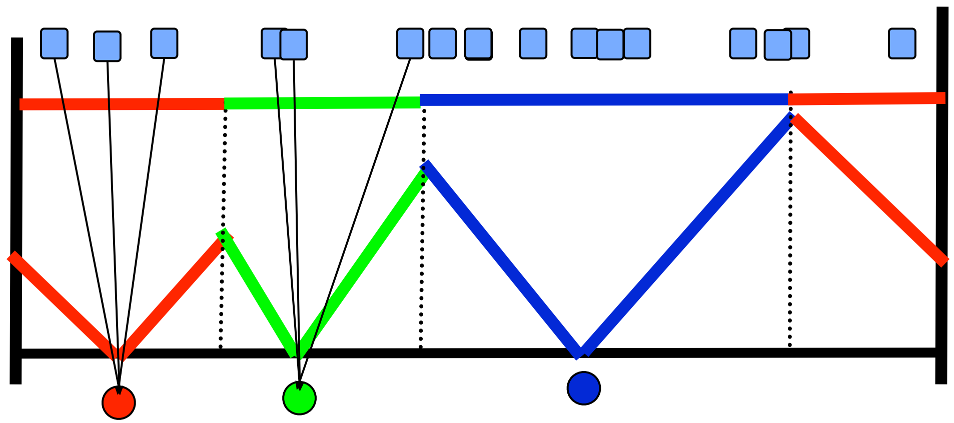
➢ **If more nodes appear**
   - then decrease all intervals by a constant factor

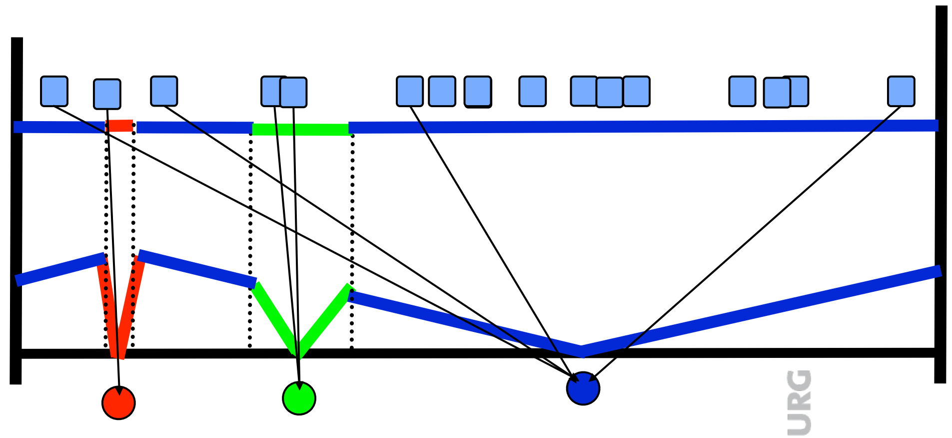
➢ **SIEVE is not providing monotony**
   - Re-stretching leads to unnecessary re-assignments



The Linear Method

- **Alternative presentation of (uniform) Consistent Hashing**
- **After “randomly” placing nodes into M**
  - Add cones pointing to the node’s location in M
- **Compute for each data element x the height of the cones**
  - Choose the cone with smallest height
- **For the Linear Method**
  - Choose for each node i a cone stretched by the factor wi
- **Compute for each data element x the height of the cones**
  - Choose the cone with smallest height



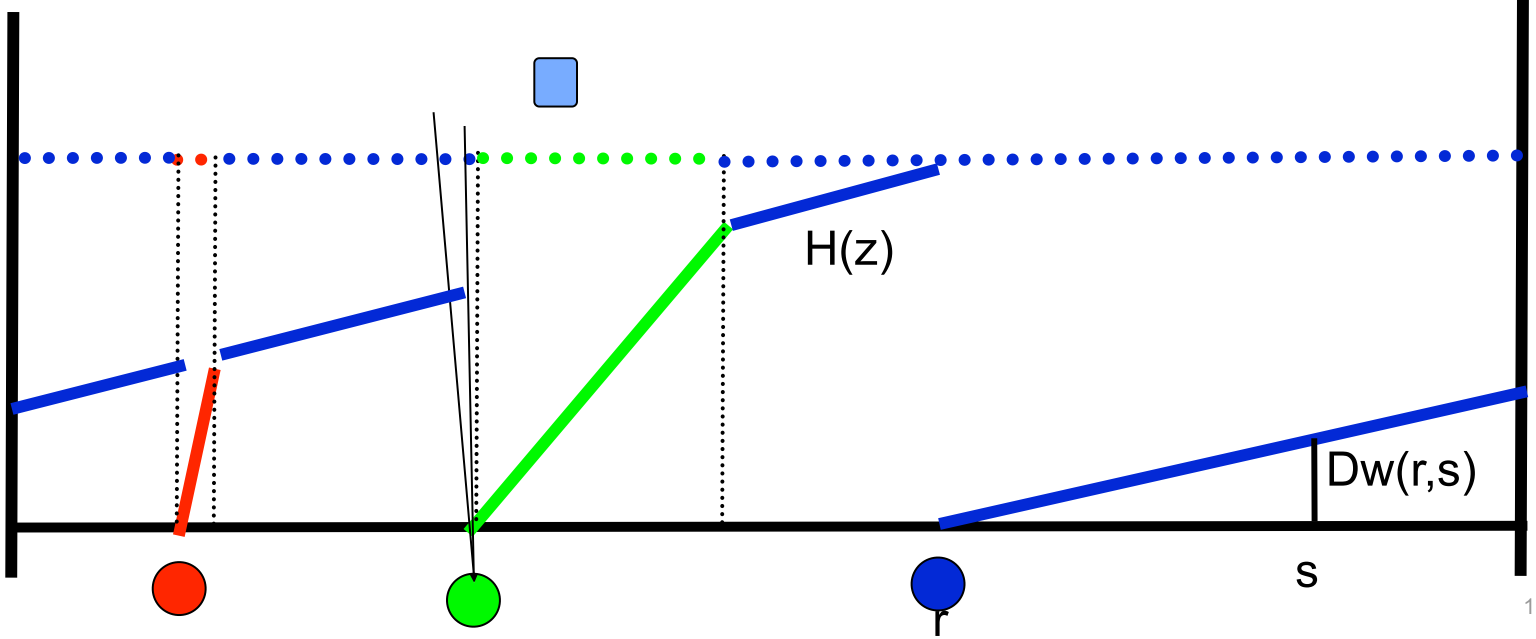


The Linear Method: Basics

- For easier description we use half-cones,
  - the weighted distance is
    - where x mod 1 := x - ⌊x⌋

$$D_w(r,s) := \frac{((s-r) \bmod 1)}{w}$$

- Analyzing heights is easier as analyzing interval lengths!
- Define: $H(z) := \min_{u \in V} D_{w_u}(z, s_u)$
  - Consider a data element and n randomly hashed nodes



The Linear Method: Basics

LEMMA 1. *Given n nodes with weights $w_1, \dots, w_n$. Then the height $H(r)$ assigned to a position r in M is distributed as follows:*

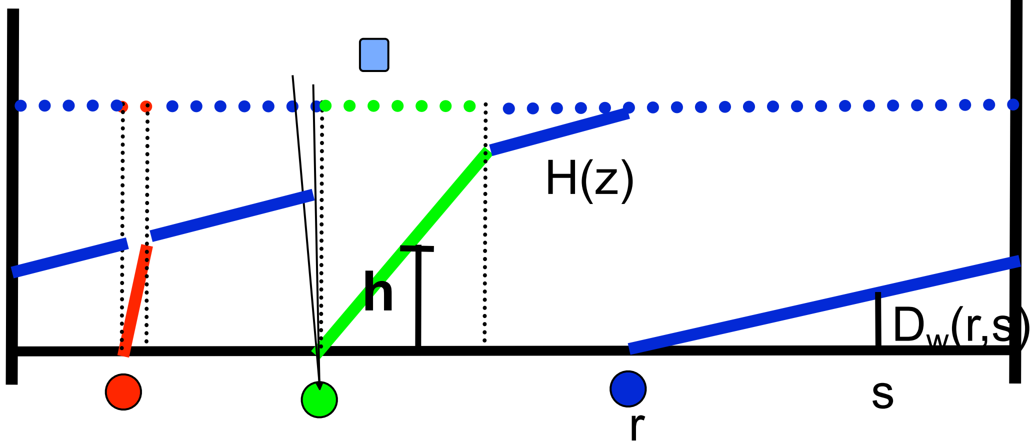
$$P[H(r) > h] = \begin{cases} \prod_{i\in[n]}(1 - hw_i)\,, & \text{if } h \le \min_i\{\frac{1}{w_i}\} \\ 0\,, & \text{else} \end{cases}$$

- Proof:
  - The probability of to receive height of at least h with respect to a node i is

    $1 - h\, w_i$
  - Since

$$\mathrm{P}[H_i \le h] = \begin{cases} 1\,, & h \ge \frac{1}{w_i} \\ h \cdot w_i & \text{else.} \end{cases}$$

$$H(z) := \min_{u\in V} D_{w_u}(z, s_u)$$



∎

An Upper Bound for Fairness

THEOREM 1. *The Linear Method stores with probability of at most* $\frac{w_i}{W-w_i}$ *a data element at a node* i*, where* $W := \sum_{i=1}^{|V|} w_i$.

Proof:
From Lemma 1 follows

$$\mathrm{P}[H_i \in [h, h+\delta] \wedge \forall j \neq i : \; H_j > h] = \begin{cases} 0\,, & \exists j \; : \; h \geq \frac{1}{w_j} \\ \delta w_i \prod_{j \neq i}(1 - h w_j) & \text{else}\,. \end{cases}$$

We define $P_{i,h,\delta} := \delta w_i \prod_{j \neq i}(1-h w_j)$

and the following term describes an upper bound

$$\sum_{m=1}^{\infty} P_{i,\delta m,\delta} \qquad \text{where} \qquad h = m\delta$$

An Upper Bound for Fairness (II)

THEOREM 1. *The Linear Method stores with probability of at most* $\frac{w_i}{W-w_i}$ *a data element at a node* i*, where* $W := \sum_{i=1}^{|V|} w_i$.

Proof (continued):

$$\begin{aligned}
\lim_{\delta\to 0}\sum_{m=1}^{\infty} P_{i,\delta m,\delta} &\leq \lim_{\delta\to 0}\sum_{m=1}^{\infty} w_i \delta e^{-a\delta m} \\
&= \int_{x=0}^{\infty} w_i e^{-ax} dx = \frac{w_i}{a} \\
&= \frac{w_i}{\sum_{j\neq i} w_j}
\end{aligned}$$

The Limits of the Linear Method

THEOREM 5. *The Linear Method (without copies) for n nodes with weights $w_1 = 1$ and $w_2, \ldots, w_{n-1} = \frac{1}{n-1}$ assigns a data element with probability $1 - e^{-1} \approx 0.632$ to node 0 when n tends to infinity.*

PROOF. We use Lemma 1 and reduce the probability to the following term.

$$\lim_{n\to\infty} \int_{x=0}^{1} x \left(1 - \frac{x}{n-1}\right)^{n-1} dx =$$

$$\int_{x=0}^{1} x e^{-x} dx = \left[-e^{-x}\right]_0^1 = 1 - e^{-1} \,.$$

Why does the biggest node win?

The small ones are competing against each other

The big one has no competitor in his league

The solution:

Use copies of each node

The Linear Method with Copies

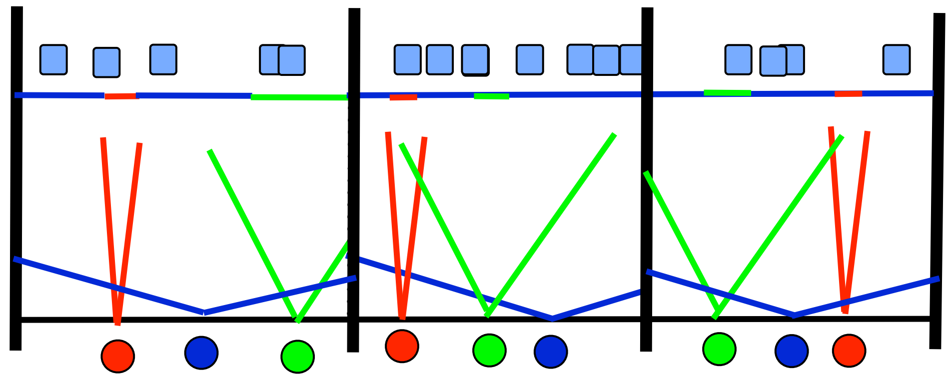
THEOREM 2. *Let $\epsilon > 0$. Then, the Linear Method using $\lceil \frac{2}{\epsilon} + 1 \rceil$ copies assigns one data element to node i with probability p_i where*

$$(1 - \sqrt{\epsilon}) \cdot \frac{w_i}{W} \leq p_i \leq (1 + \epsilon) \cdot \frac{w_i}{W} .$$

- A constant number of copies suffice to "repair" the linear function
- This theorem works only for one data item
  - If many data items are inserted, then the original bias towards some nodes is reproduced:
    - "Lucky" nodes receive more data items
- Solution
  - Independently repeat the game at least O(log n) times

Partitioning and the Linear Method

- Partitions:
  - Partition the hash range into sub-intervals
  - Map each data element into the whole interval
  - Map for each node 2/ε+1 copies into each sub-interval



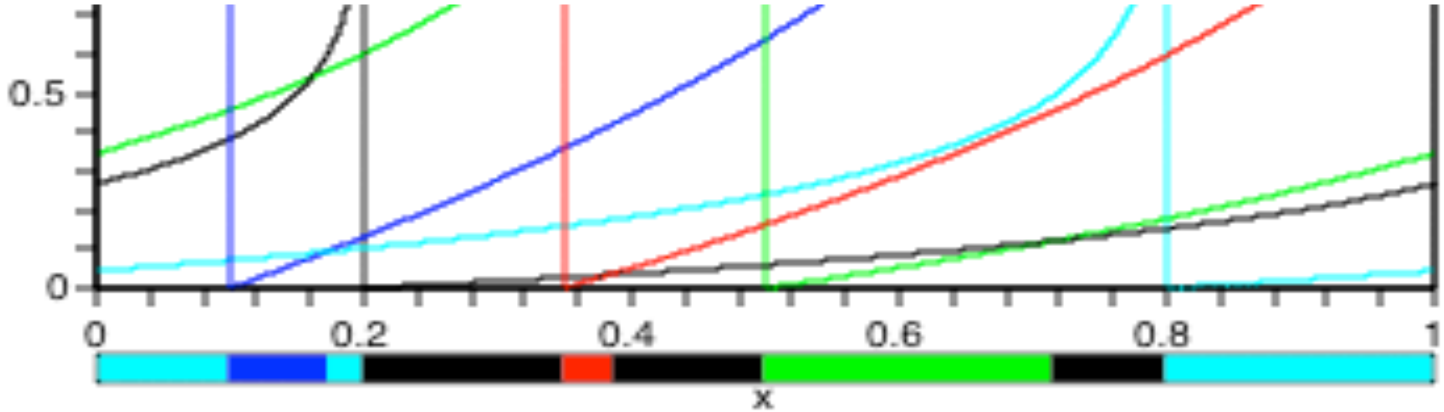
Theorem 3 *For all $\epsilon, \epsilon' > 0$ and $c > 0$ there exists $c' > 0$ such that when we apply the Linear Method to n nodes using $\lceil \frac{2}{\epsilon} + 1 \rceil$ copies and $c' \log n$ partitions, the following holds with high probability, i.e. $1 - n^{-c}$.*

Every node $i \in V$ receives all data elements with probability p_i such that

$$(1 - \sqrt{\epsilon} - \epsilon') \cdot \frac{w_i}{W} \le p_i \le (1 + \epsilon + \epsilon') \cdot \frac{w_i}{W} .$$

The Logarithmic Method

- Replacing the linear function by $L_w(r,s) := \frac{-\ln((1-(r-s)) \bmod 1)}{w}$
- improves the accuracy



FACT 2. *If in the Logarithmic Method (without copies and without partitions) a node arrives with weight w then the probability that data element x with previous height H_x is assigned to the new node is $1 - e^{-wH_x}$.*

THEOREM 6. Given n nodes with positive weights $w_1, \ldots, w_n$ the Logarithmic Method assigns a data element to node i with probability $\frac{w_i}{W}$, where $W := \sum_{i=1}^{|V|} w_i$.

The Logarithmic Method

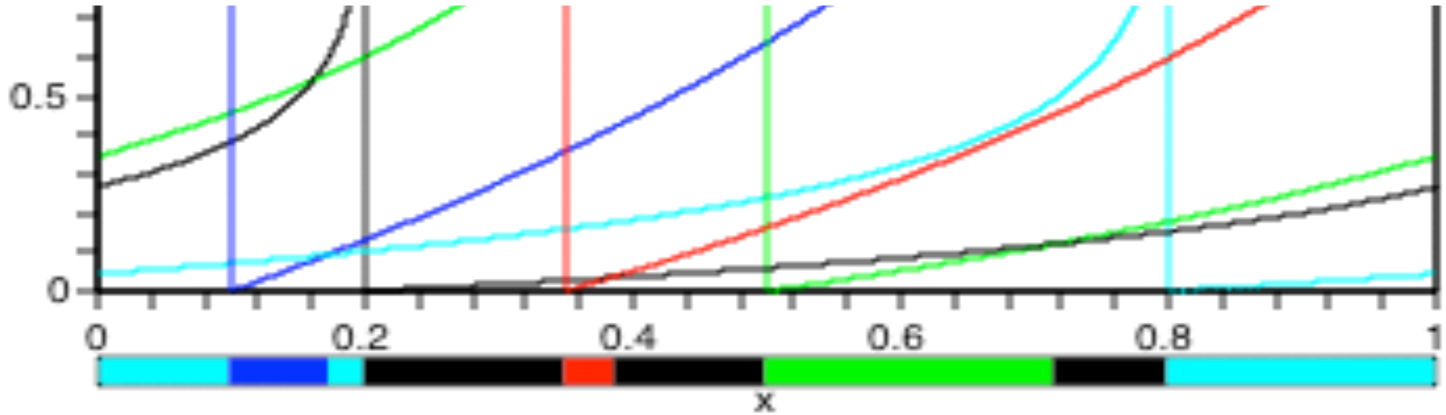
CoNe Freiburg

- Replacing the linear function with -ln((1-d_i(x)) mod 1)/w_i improves the accuracy of the probability distribution

Theorem 7 *For all $\epsilon > 0$ and $c > 0$ there exists $c' > 0$, where we apply the Logarithmic Method with $c' \log n$ partitions. Then, the following holds with high probability, i.e. $1 - n^{-c}$.*

Every node $i \in V$ receives data elements with probability p_i such that

$$(1-\epsilon) \cdot \frac{w_i}{W} \leq p_i \leq (1+\epsilon) \cdot \frac{w_i}{W} .$$



Further Features

- Efficient data structure for the linear and logarithmic method
  - can be implemented within O(n) space
  - Assigning elements can be done in O(log n) expected time
  - Inserting/deleting new nodes can be done in amortized time O(1)
- Predicting Migration
  - The height of a data element correlates with the probability that this data element is the next to migrate to a different server
- Fading in and out
  - Since the consistency works also for the weights:
  - Nodes can be inserted by slowly increasing the weight
  - No additional overhead
  - Node weight represents the transient download state
  - Vice versa for leaving nodes

Double Hashing

- If every node uses a different hashing, then the logarithmic method can be chose without any copies

For this, we apply for each node an individual hash function $h : V \times [0,1) \to [0,1)$. So, we start mapping the data element x to $r_x \in [0,1)$ as above and then for every node we compute $r_{i,x} = h(i, r_x)$. Now x is assigned to a node i which minimizes $r_{i,x}/w_i$ according the Linear Method. In the Logarithmic Method x is assigned to the node minimizing $-\ln(1 - r_{i,x})/w_i$.

- Advantage:
  - Perfect probability distribution
- Disadvantage:
  - Intrinsic linear time w.r.t. the number of servers

- This is the method of choice for Storage Area Networks

DAAD Summerschool Curitiba 2011

Aspects of Large Scale High Speed Computing Building Blocks of a Cloud

Storage Networks

4: Distributed Heterogeneous Hash Tables

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