

参考資料

DHTの基礎知識

※発表内容ではありません。 2009.12.11 VIOPS事務局

DHT(Distributed Hash Table)分散ハッシュテーブル

- P2Pオーバーレイネットワークを構成
- Key-Valueペア分散探索技術
- IDにConsistent-Hashを利用
- スケールアウトしやすい

代表的なDHTアルゴリズム

- CAN (2001) N次トーラス
- Tapestry, Pastry (2001) Plaxton
- Kademlia (2002) 二分木
- Chord (2001) 環状
- Koorde (2004) Chord改
- Broose (2004) Koorde+Kademlia改

CAN Content Addressable Network

論文 A Scalable Content Addressable Network (2001 UCB & ATT)

N次元トーラス空間にノードIDとコンテンツIDをマッピング

探索ホップ数= $O(dN^{1/d})$

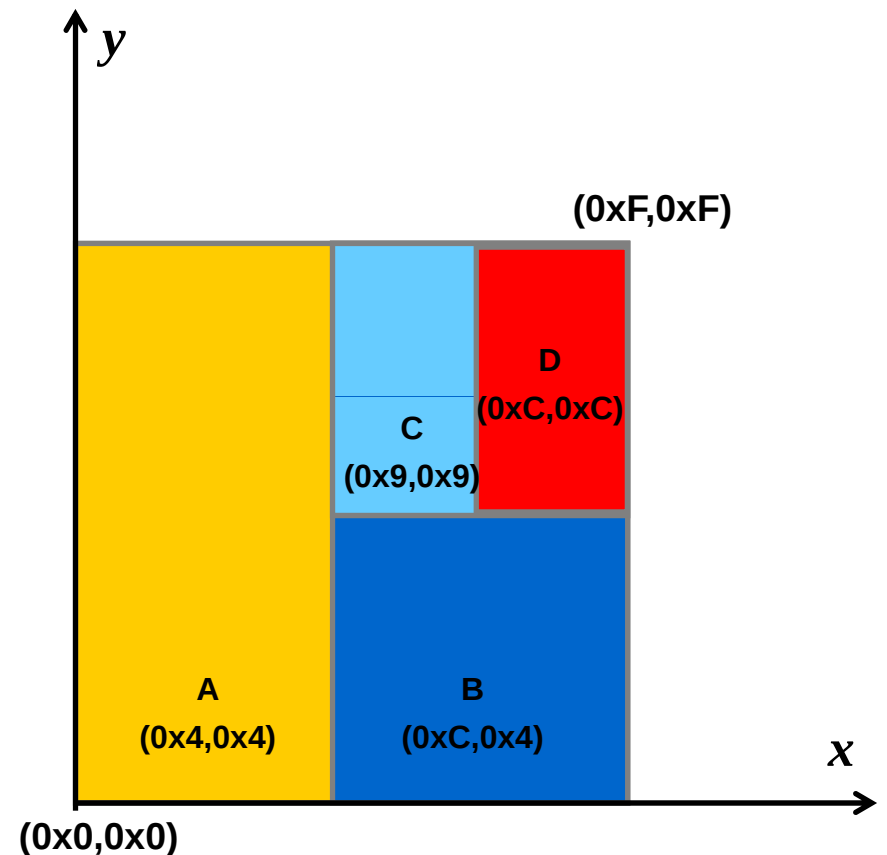
$X = \text{Hash1}(\text{'Hello'}) = 0xE$

$Y = \text{Hash2}(\text{'Hello'}) = 0xC$

$\text{Put}(\text{'Hello'}, \text{'World'}) \rightarrow \text{ノードD}$

$(0xE, 0xC)$ のコンテンツをAから探索

$A \rightarrow C \rightarrow D \rightarrow (0xE, 0xC) \rightarrow \text{'World'}$

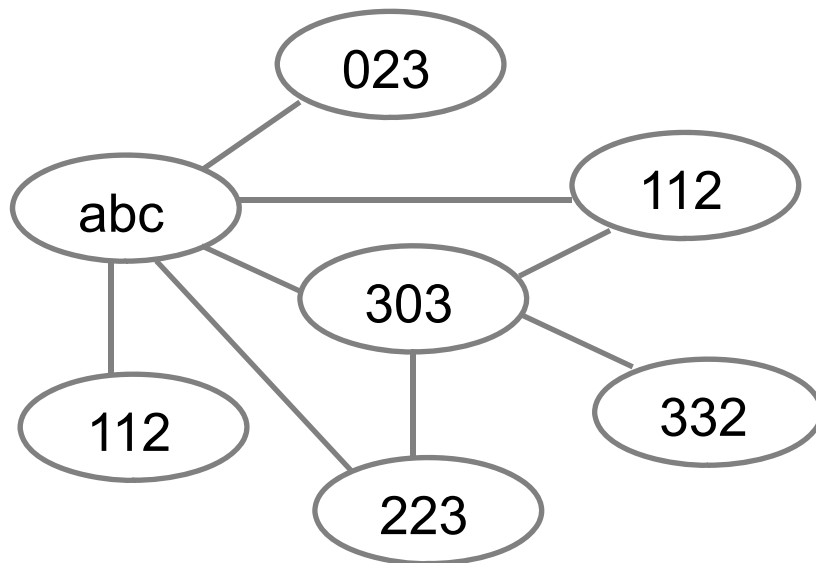


Pastry PlaxtonアルゴリズムによるDHT

論文 Scalable distributed object location and routing

for large-scale peer-to-peer systems (2001 MS & Rice)

- ・ Plaxtonアルゴリズム
- ・ IDは基数(b)と桁数(n)により構成 (例 3桁4進数)



ノードabcの経路表

0-23	1-12	2-23	3-03
a0-1	a1-0	a2-1	a3-2
ab0	ab1	ab2	ab3

Pastryアルゴリズム $O(b \log N)$

Plaxtonテーブルとリーフセット、ネイバーフッドセットの組み合わせ

Kademlia(カデムリア)

論文 A Peer-to-peer information System

Based on the XOR Metric (2002 NYU)

二分木構造のID空間

XORを空間の距離として定義

距離 = $\text{XOR}(\text{ノードID}, \text{Key})$

ノードA = 001

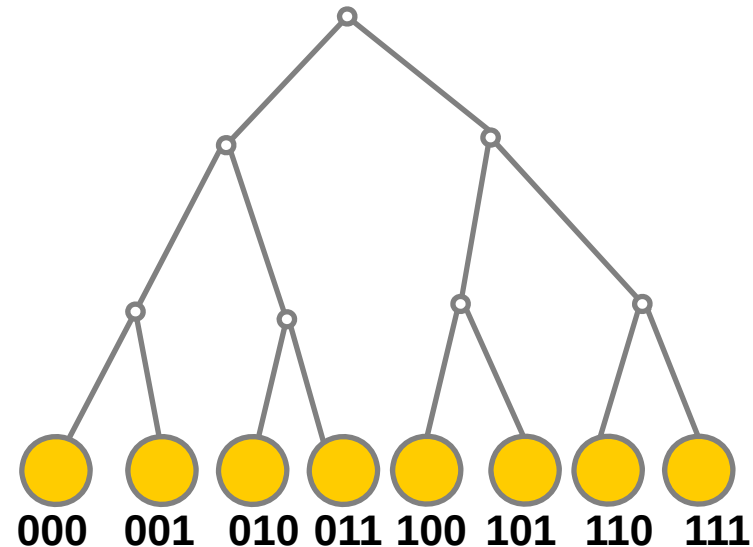
ノードB = 010

Key = 100

$\text{XOR}(001, 100) = 010 = 2$

$\text{XOR}(010, 100) = 001 = 1 \rightarrow$ ノードBにValueをストア

プレフィックス一致長によるルーティングテーブル



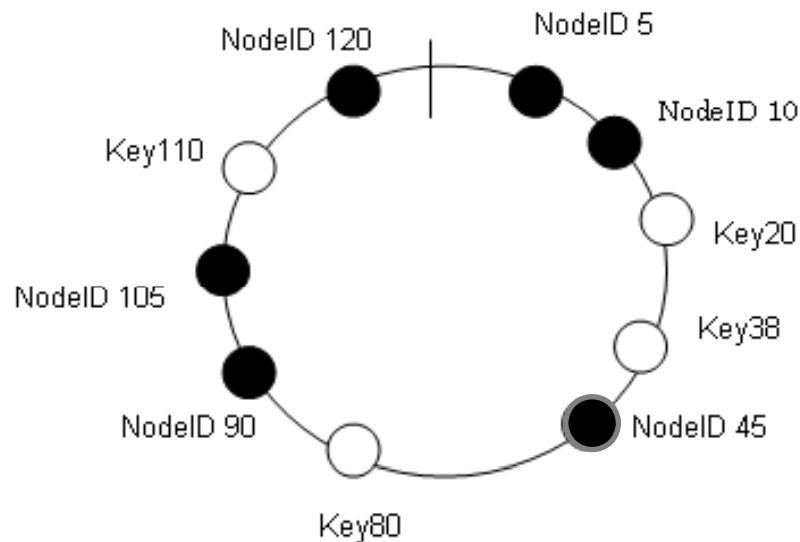
Chord

論文 A Scalable Peer-to-Peer Lookup Service for
Internet Applications (2001 MIT)

160bit SHA-1によるハッシュ化

1次元環状空間 にノードIDをマッピング

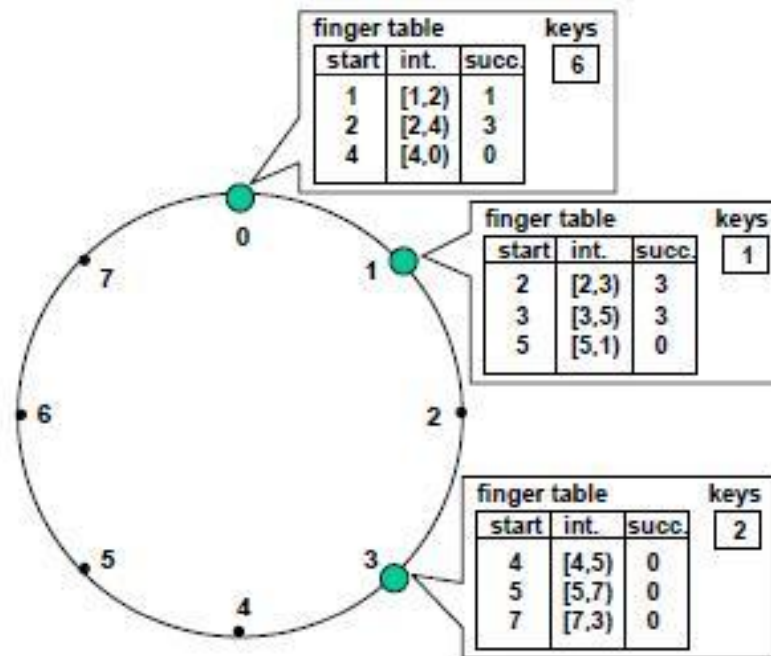
スキップリストによる探索(Finger Table) $O(\log N)$



Chord

Finger Table (mビット空間の場合、m個のエントリ)

1. $\text{finger}[i].\text{start} = (n + 2^{i-1}) \bmod 2^m$
2. Interval (範囲) = $\text{finger}[i].\text{start} \leq \text{interval} < \text{finger}[i+1].\text{start}$
3. $\text{successor}(\text{id}) = \text{interval内の一}\text{番近いノードID}$

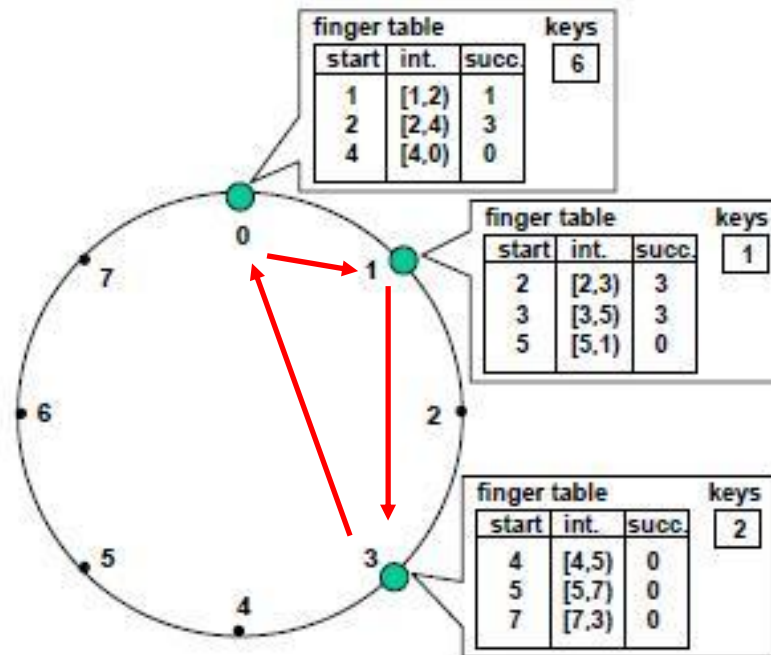


3ビット空間での例(原論文より抜粋)

Chord

探索「ノード3がキー1を探す」

1. ノード3のFinger Tableから $[7, 3)$ のsuccessorを見つける
2. ノード0のFinger Tableから $[1, 2)$ のsuccessorを知りノード3に返す
3. ノード3はノード1にキー1をリクエストする



3ビット空間での例(原論文より抜粋)

探索シーケンスの擬似コード

// ノードn上で、id のsuccecssorを探す

n.find_successor(id) { // ノード3上のプロシーダを実行

n' = find_predecessor(id); // idのpredecessorを探す、この場合 n' はノード0。

 return n'.successor; // ノード n' = 0 の finger table から id の successorノードIDを返す

}

// ノード n で id の predecessorを探す

n.find_predecessor(id) {

 n' = n; // まず、自分から

 while(id ! $\in$ (n', n'.successor]) { // finger table内のエントリで、id が intervalに入るまで

n' = n'.closest_preceding_finger(id); // id に最も近いノードIDを n' とする

 }

 return n'; // finger table内で最もidに近いpredecessorを見つけた！

}

// id が interval 内に含まれるエントリを探す

n.closest_preceding_finger(id) { // 自分のfinger tableをみるよ！

 for i = m downto 1 { // finger table の下の方から順番に

 if(finger[i].node $\in$ (n, id)) { // successorノードが 自分 と id との間にあれば、

 return finger[i].node; // そのsuccessorノードIDを返す

 }

 return n; // 無ければ、それは自分だよ

}

}