



UFC

DEPARTAMENTO
DE COMPUTAÇÃO

MDCC

GR€at

Consistent Hashing

O problema se escalar um sistema...

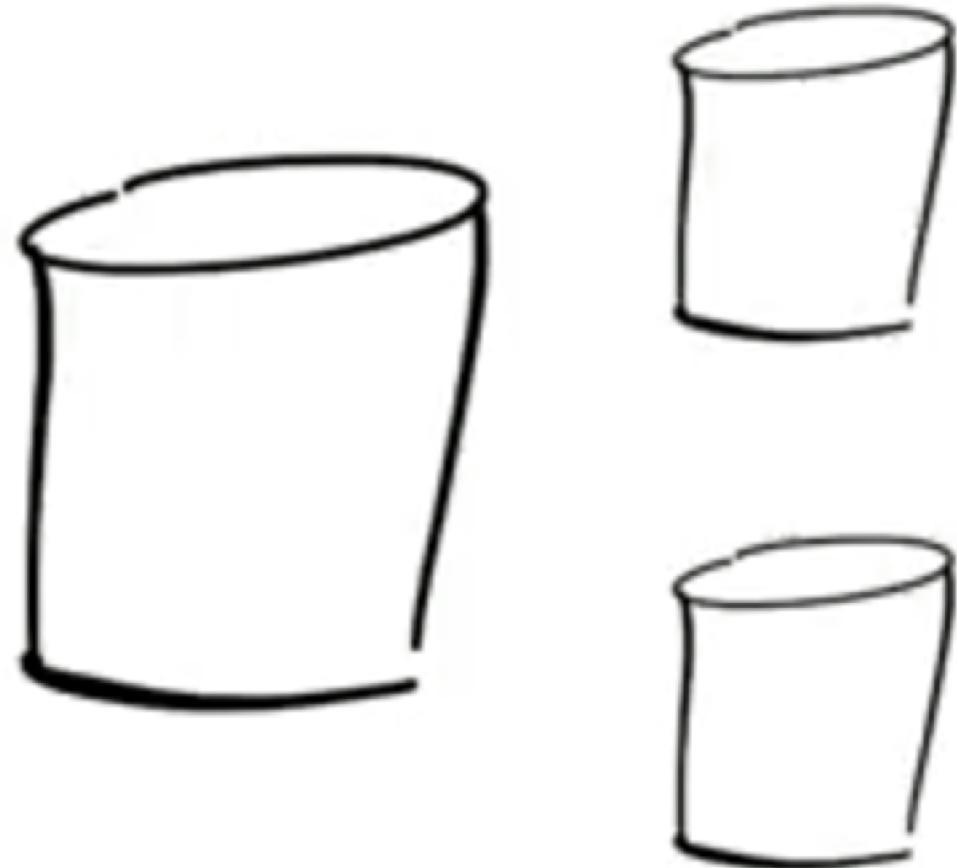
- Uma visão sobre um sistema distribuído...
- Tem-se a visão de um único sistema, porém
 - Há diferentes nós...
 - Nós que são independentes
 - Nós que não compartilham um relógio global
 - Necessidade de sincronização
- Aspectos da distribuição
 - Armazenamento
 - Computação
 - Comunicação

A visão mais simples



Escalando o sistema

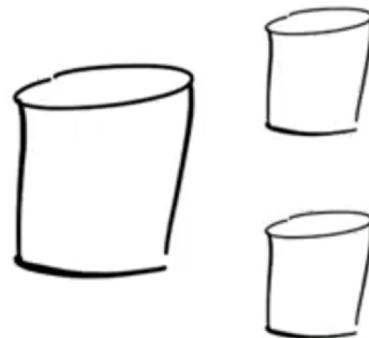
- Replicação



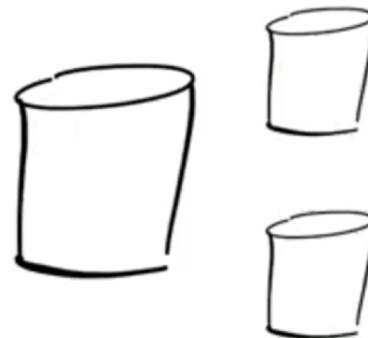
Escalando ainda mais

- Sharding

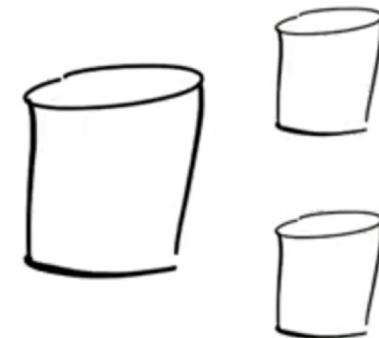
A - F



G - N



O - Z



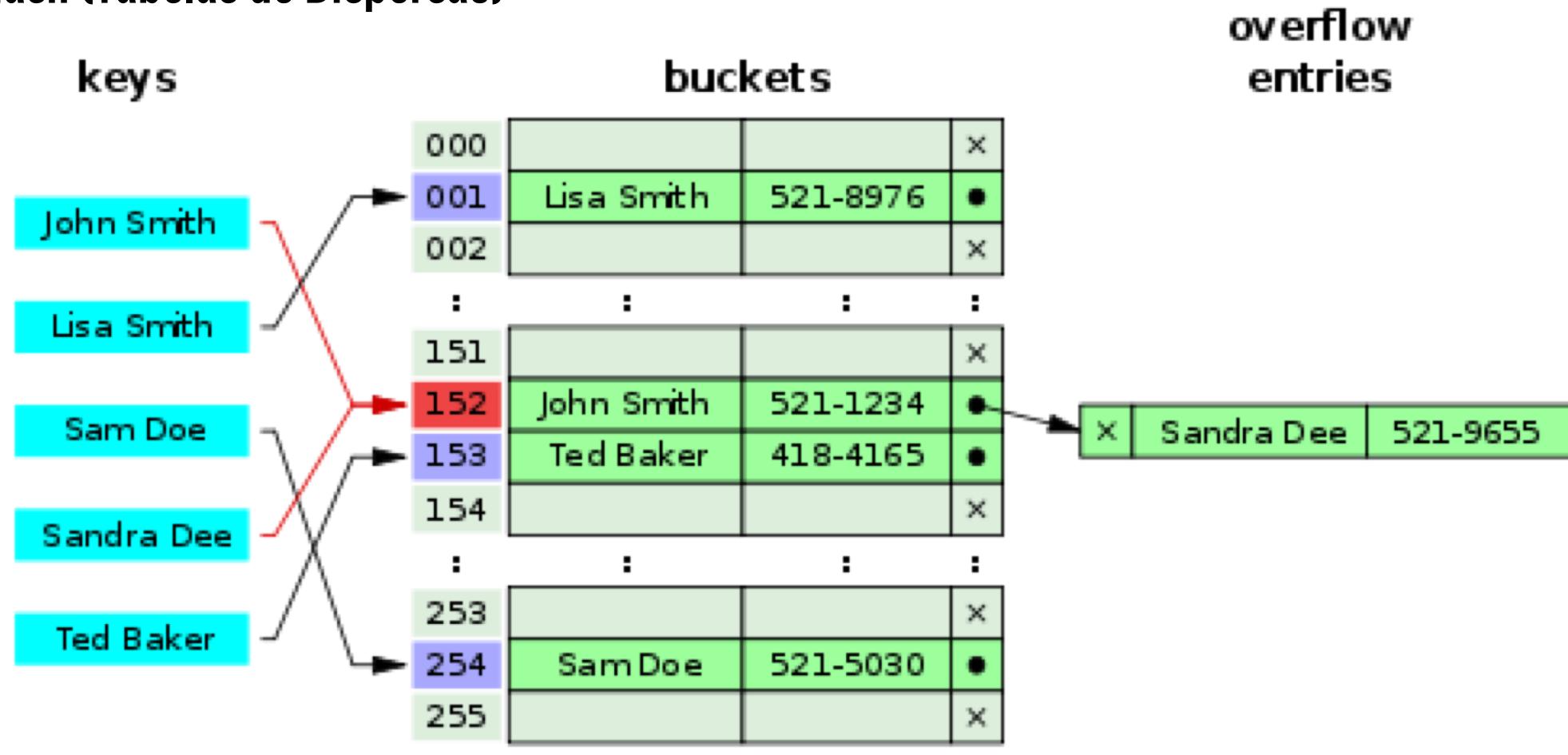
Consistent Hashing

- Karger et al (1997).
 - Artigo seminal
 - Como distribuir os pedidos dos clientes em um cluster de servidores Web com atualizações frequentes?
- Organização dos servidores e conteúdo em uma topologia de anel
 - Faixas de conteúdo são alocadas à um servidor específico
 - Adição e Remoção de Servidores demanda baixa atualização na disposição dos conteúdos

O que é uma Função Hash?

- Qualquer função que dados de tamanho qualquer para um conjunto de dados de tamanho fixo
- Entradas possíveis >> Saída Possíveis
 - Problema: Colisão
- Boas funções hash
 - A partir de os dados de entrada -> saídas com diferentes valores
 - Distribuídas tão uniformemente quanto possível sobre a faixa de saída
- Tabelas Hash
 - Chaves/Valor
 - Buckets

Tabelas Hash (Tabelas de Dispersão)



Index = $\text{hash}(\text{chave}) \text{ MOD } \text{Num_buckets}$

Usando hash para distribuir conteúdo

- Usar caches para aliviar o acesso a um banco
 - Busca na cache
 - Se tiver, recupera e segue a vida
 - Se não tiver (*cache fault*), busca no banco, atualiza a cache e segue a vida
- Questões:
 - Como distribuir o conteúdo?
 - Solução mais simples: servidor = *hash* (chave) MOD N, onde N é quantidade de servidores

Exemplo: 3 servidores

Chave	Hash	HASH MOD 3
John	1633428562	2
Bill	7594634739	0
Jane	5000799124	1
Steve	9787173343	0
Kate	3421657995	2

Exemplo: 3 servidores (buscar por John)

A	B	C
		John

Exemplo: 3 servidores (buscar por Bill)

A	B	C
Bill		John

Exemplo: 3 servidores (todos na cache)

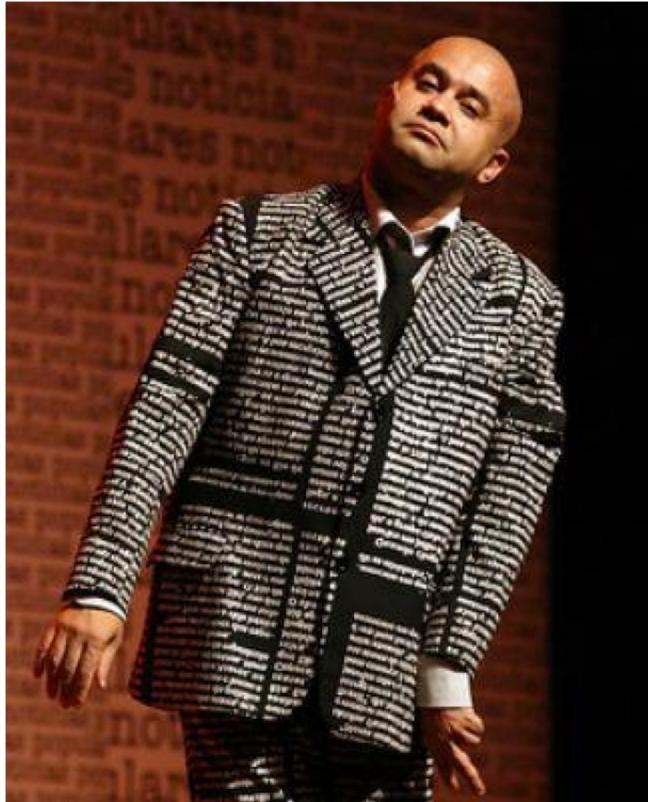
A	B	C
Bill	Jane	John
Steve		Kate

Distributed Hashing

- Funciona bem, Simples, Intuitivo...

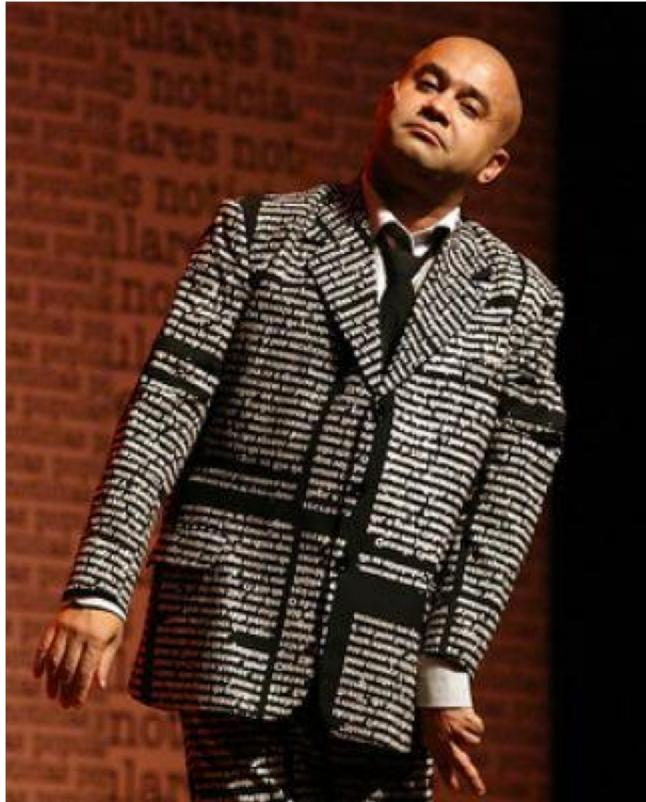
Distributed Hashing

- Funciona bem, Simples, Intuitivo...
- Porém...



Distributed Hashing

- Funciona bem, Simples, Intuitivo...
- Porém...



Se um servidor cair?
Se um novo servidor for adicionado?

Exemplo: Um servidor cai

- servidor = hash (chave) MOD 3 -> servidor = hash (chave) MOD 2

Chave	Hash	HASH MOD 2
John	1633428562	0
Bill	7594634739	1
Jane	5000799124	0
Steve	9787173343	1
Kate	3421657995	1

Exemplo: 3 servidores (todos na cache)

A	B
John	Bill
Jane	Steve
	Kate

Exemplo: 3 servidores (todos na cache)

A	B	C
Bill	Jane	John
Steve		Kate

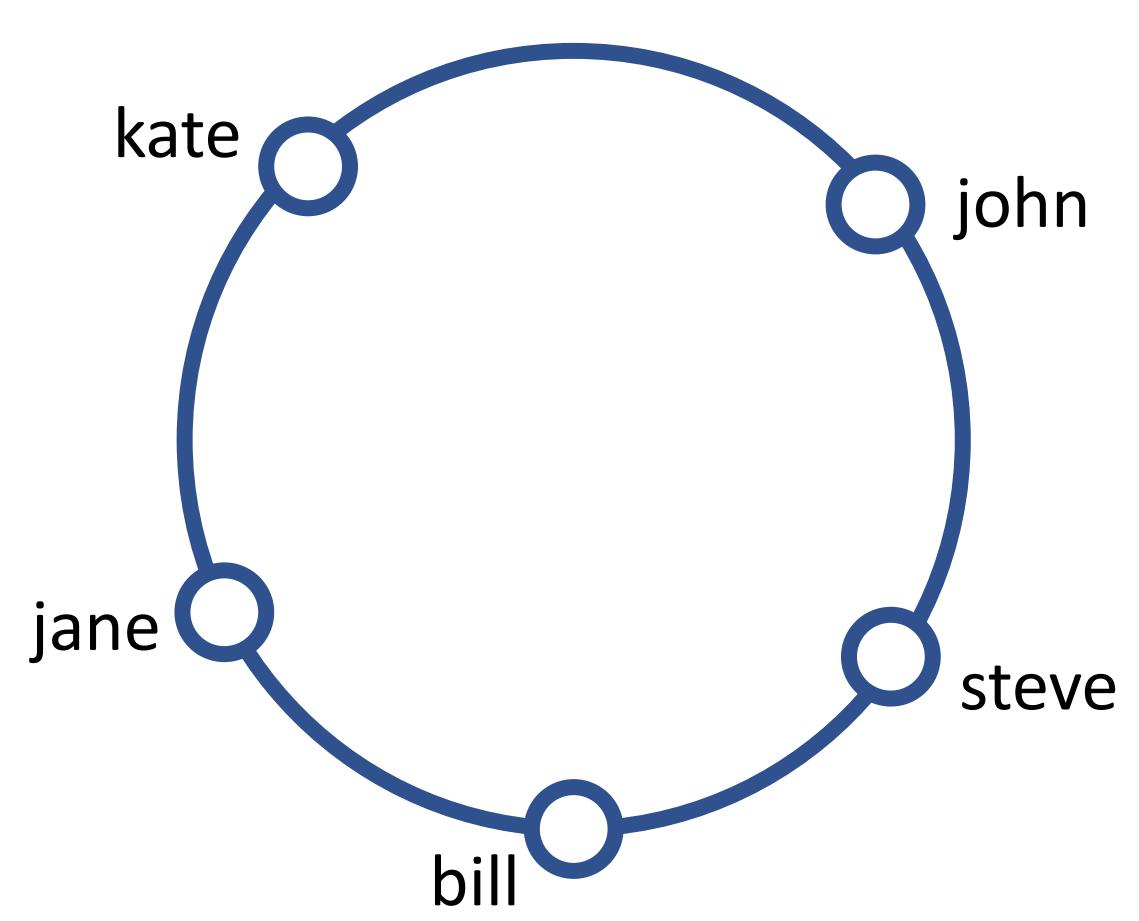
A	B
John	Bill
Jane	Steve
	Kate

Consistent Hashing

- Diminuir a dependência do número de servidores
- Mapeamento dos objetos em uma abstração de um círculo ou anel (*hash ring*)
- As posições no anel são marcadas de acordo com o valor da hash de uma chave
 - Valor inicial = ângulo ZERO
 - Valor final – ângulo 2π (360)

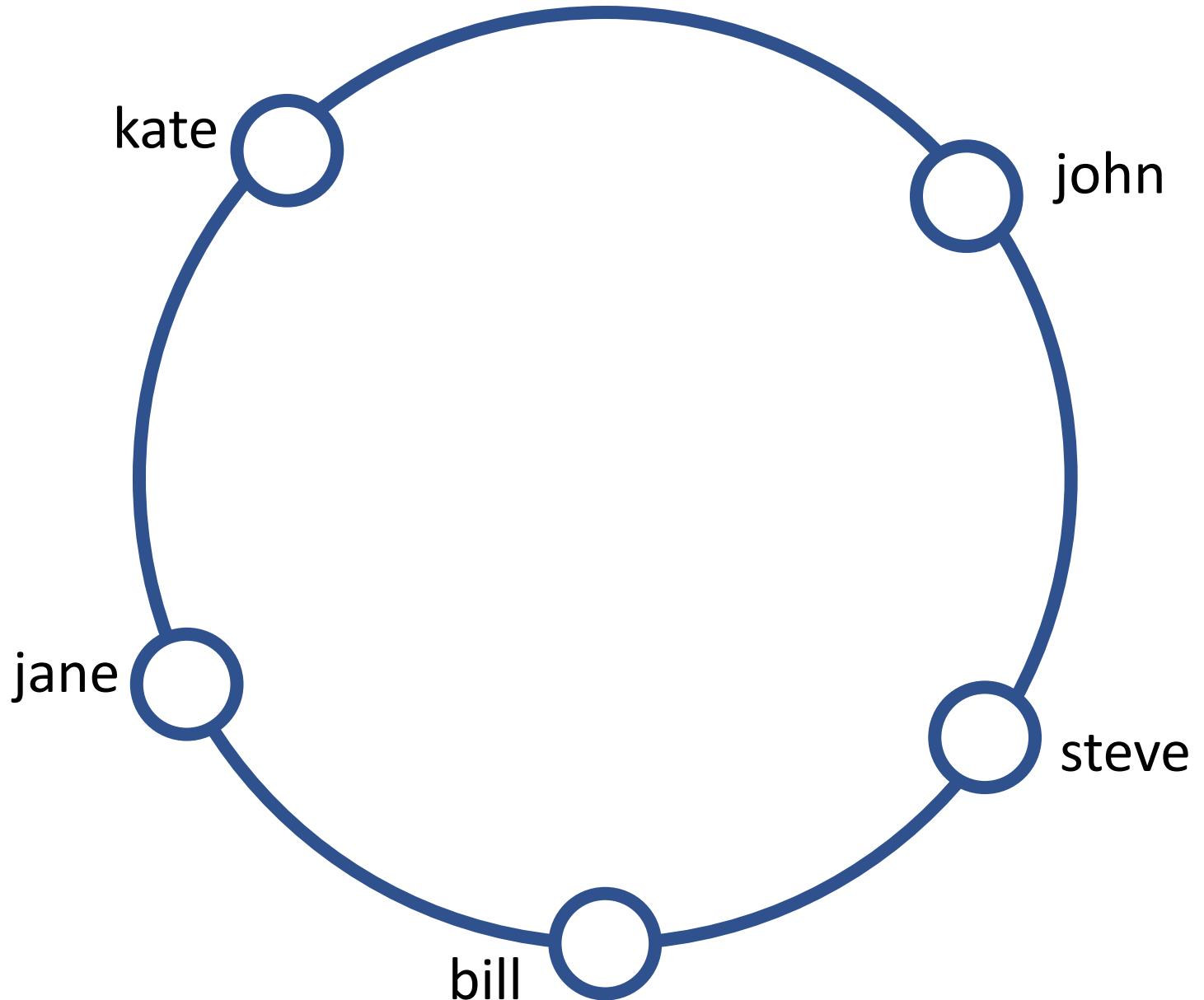
Distribuindo no Hash Ring

Chave	Hash	Angulo
John	1633428562	58.8
Bill	7594634739	273.4
Jane	5000799124	180
Steve	9787173343	352.3
Kate	3421657995	123.2



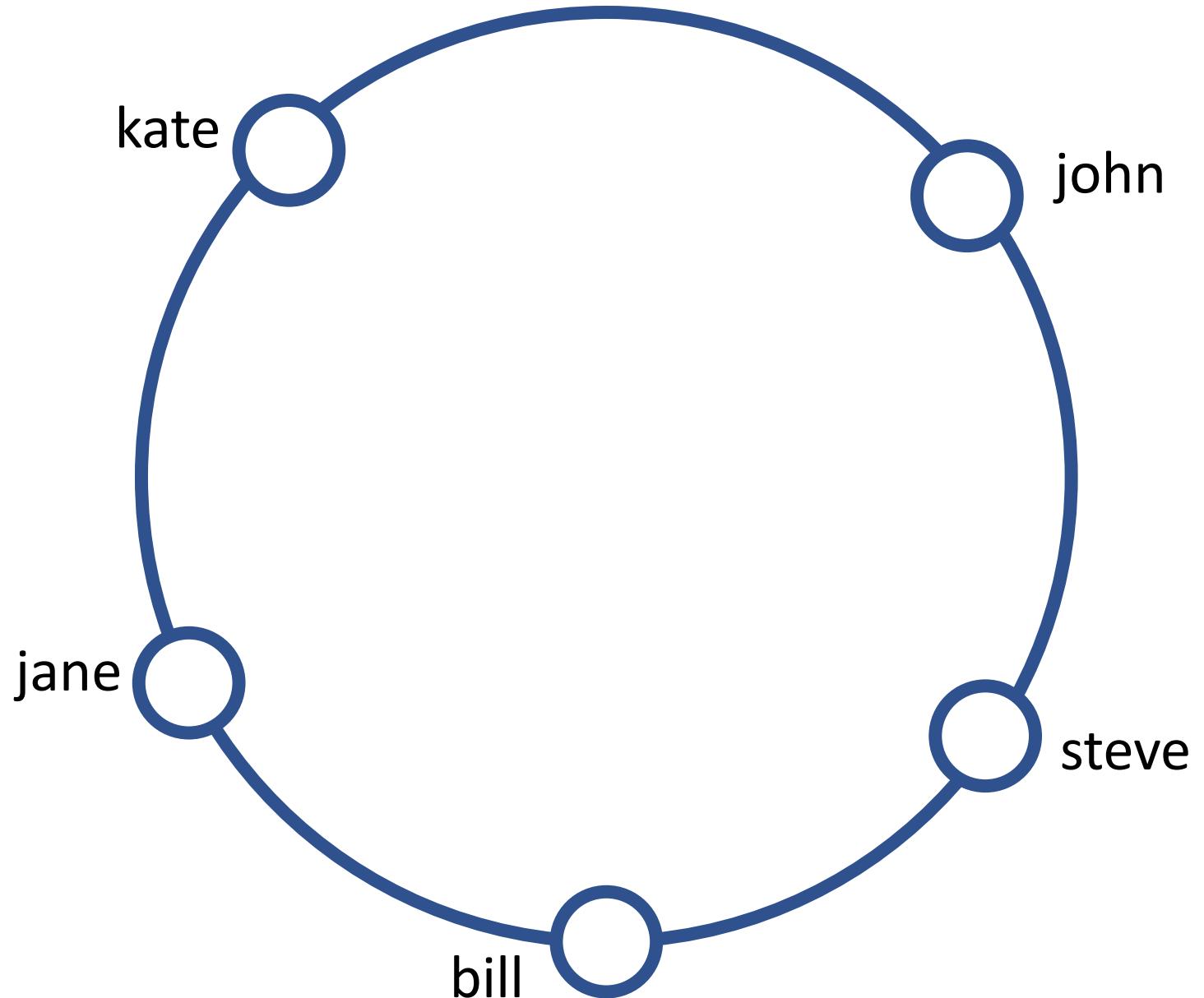
Adicionando Servidores

Chave	Hash	Angulo
John	1633428562	58.8
Bill	7594634739	273.4
Jane	5000799124	180
Steve	9787173343	352.3



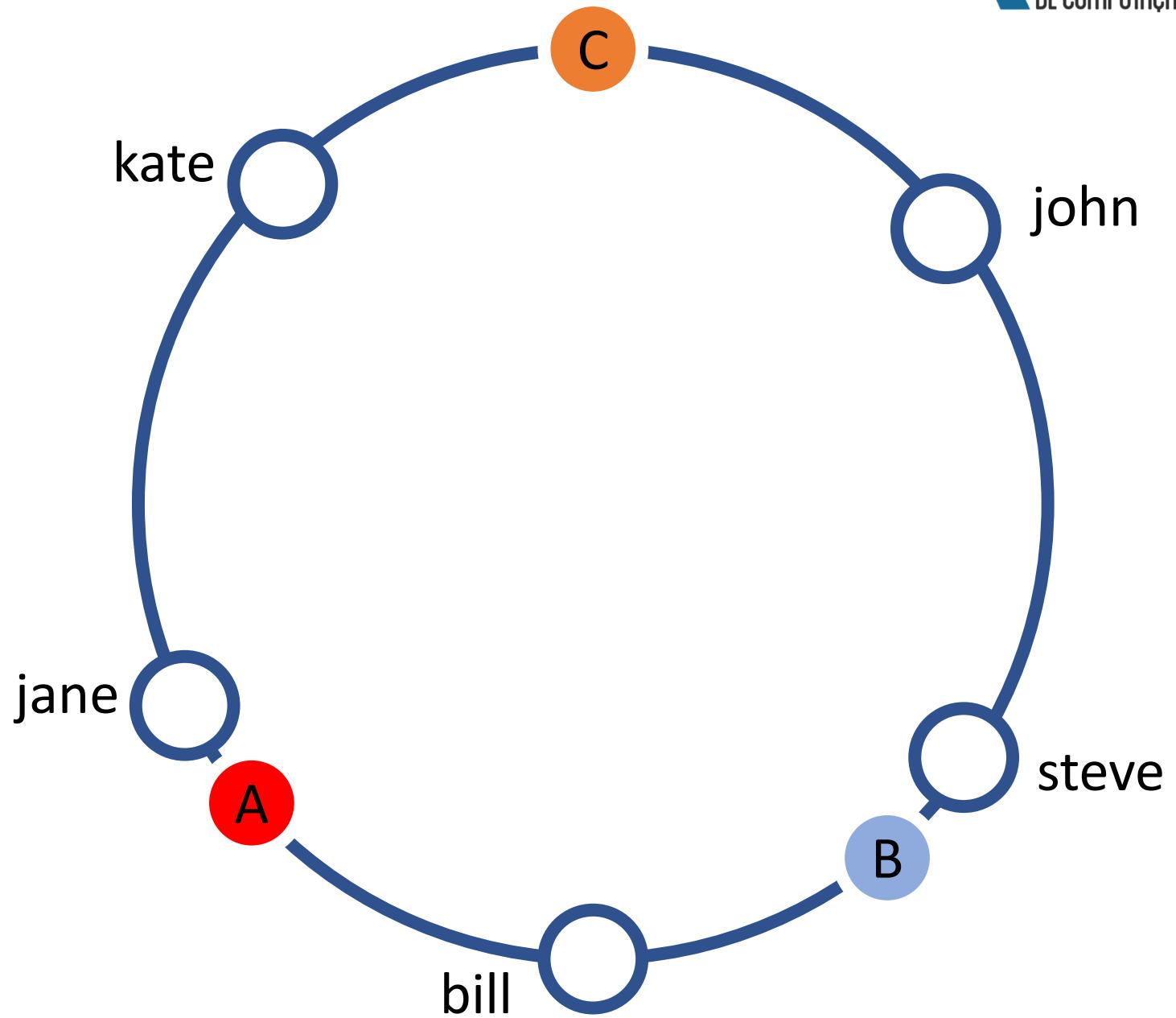
Adicionando Servidores

Chave	Hash	Angulo
John	1633428562	58.8
Bill	7594634739	273.4
Jane	5000799124	180
Steve	9787173343	352.3
Kate	3421657995	123.2
A	5572014558	200.6
B	8077113362	290.8
C	2269549488	81.7



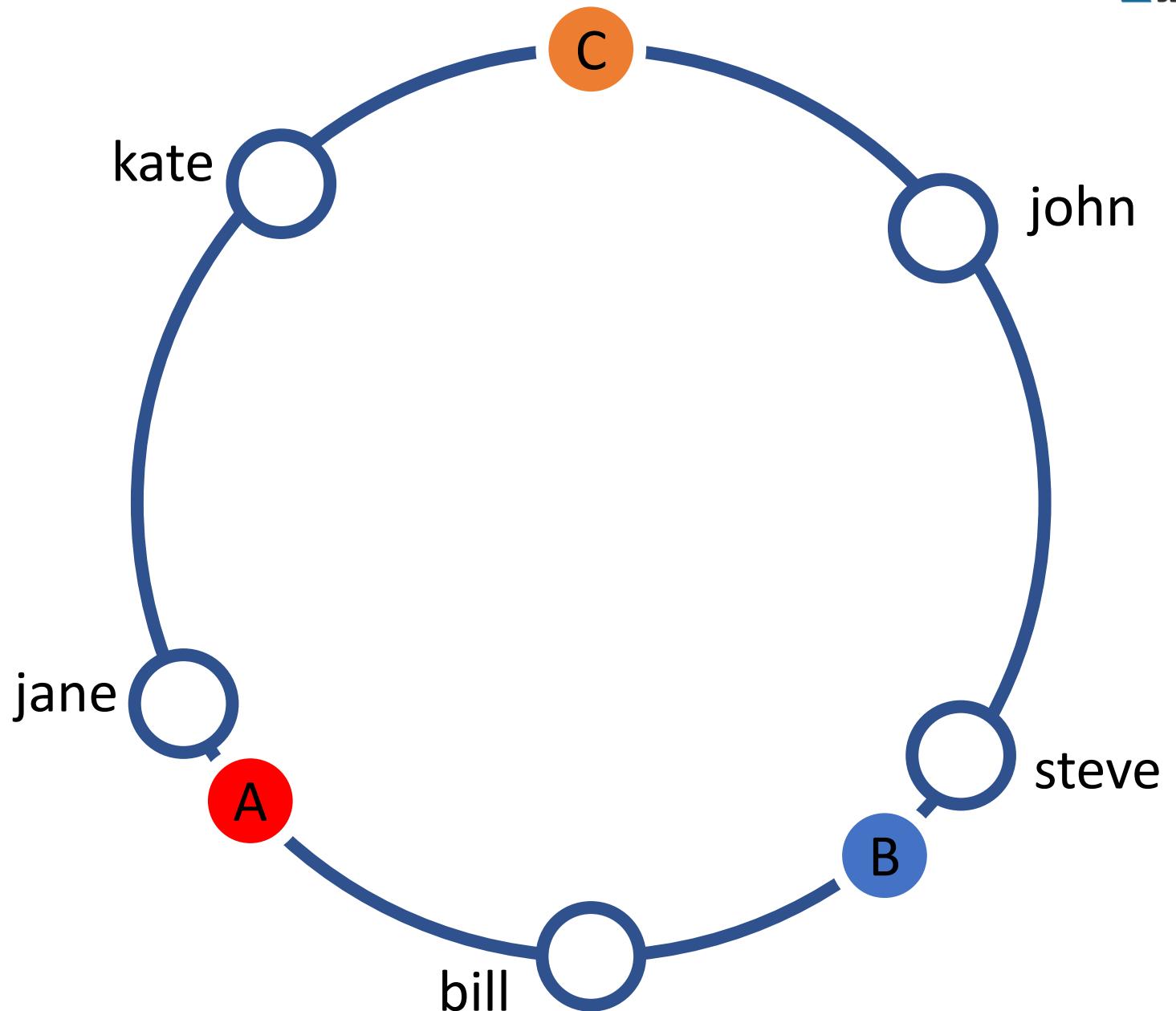
Adicionando Servidores

Chave	Hash	Angulo
John	1633428562	58.8
Bill	7594634739	273.4
Jane	5000799124	180
Steve	9787173343	352.3
Kate	3421657995	123.2
A	5572014558	200.6
B	8077113362	290.8
C	2269549488	81.7



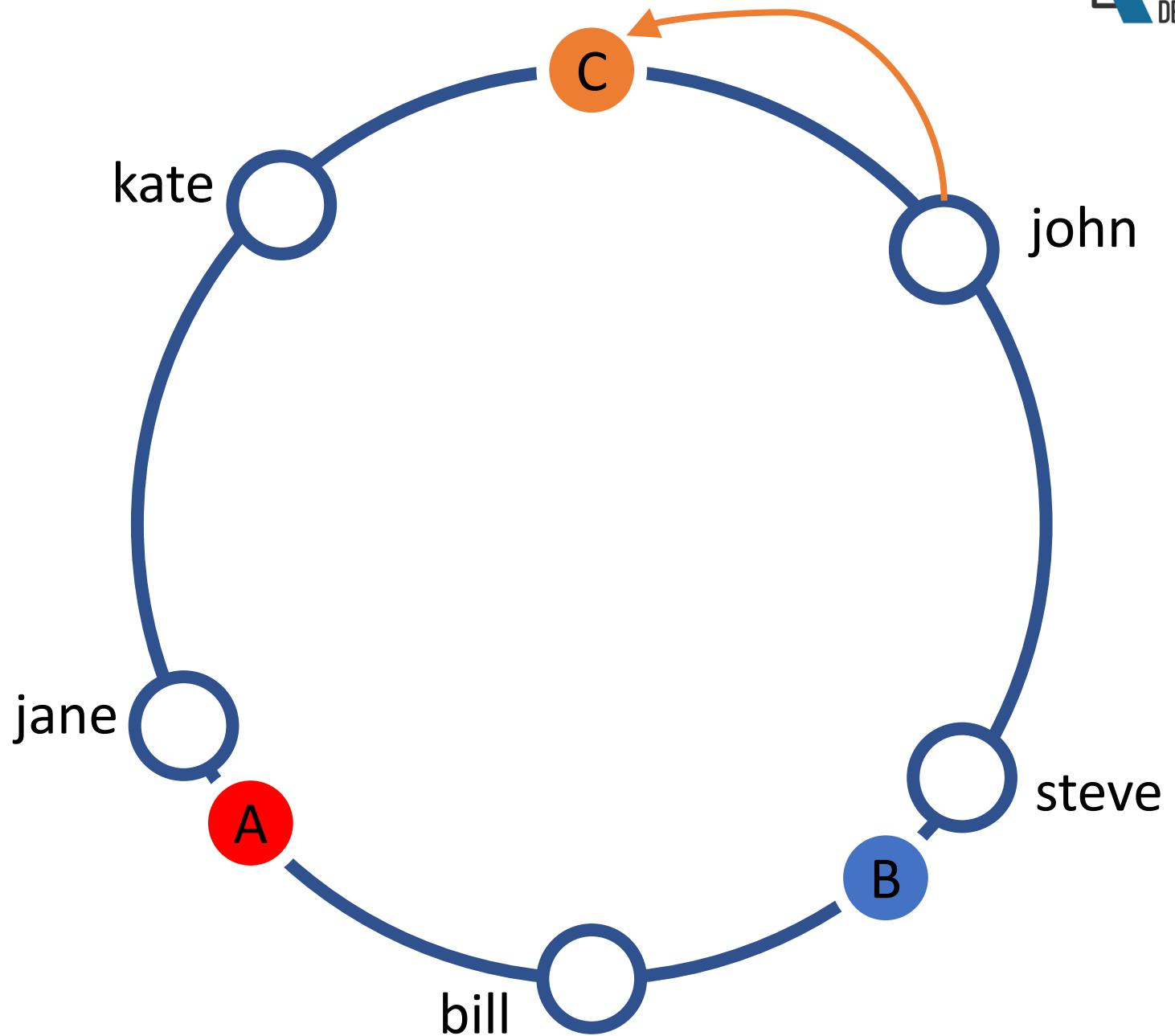
Marcando Dados

Chave	Hash	Angulo	Serv
John	1633428562	58.8	
kate	3421831276	123.2	
Jane	5000648311	180	
bill	7594873884	273,4	
Steve	9786437450	352.3	



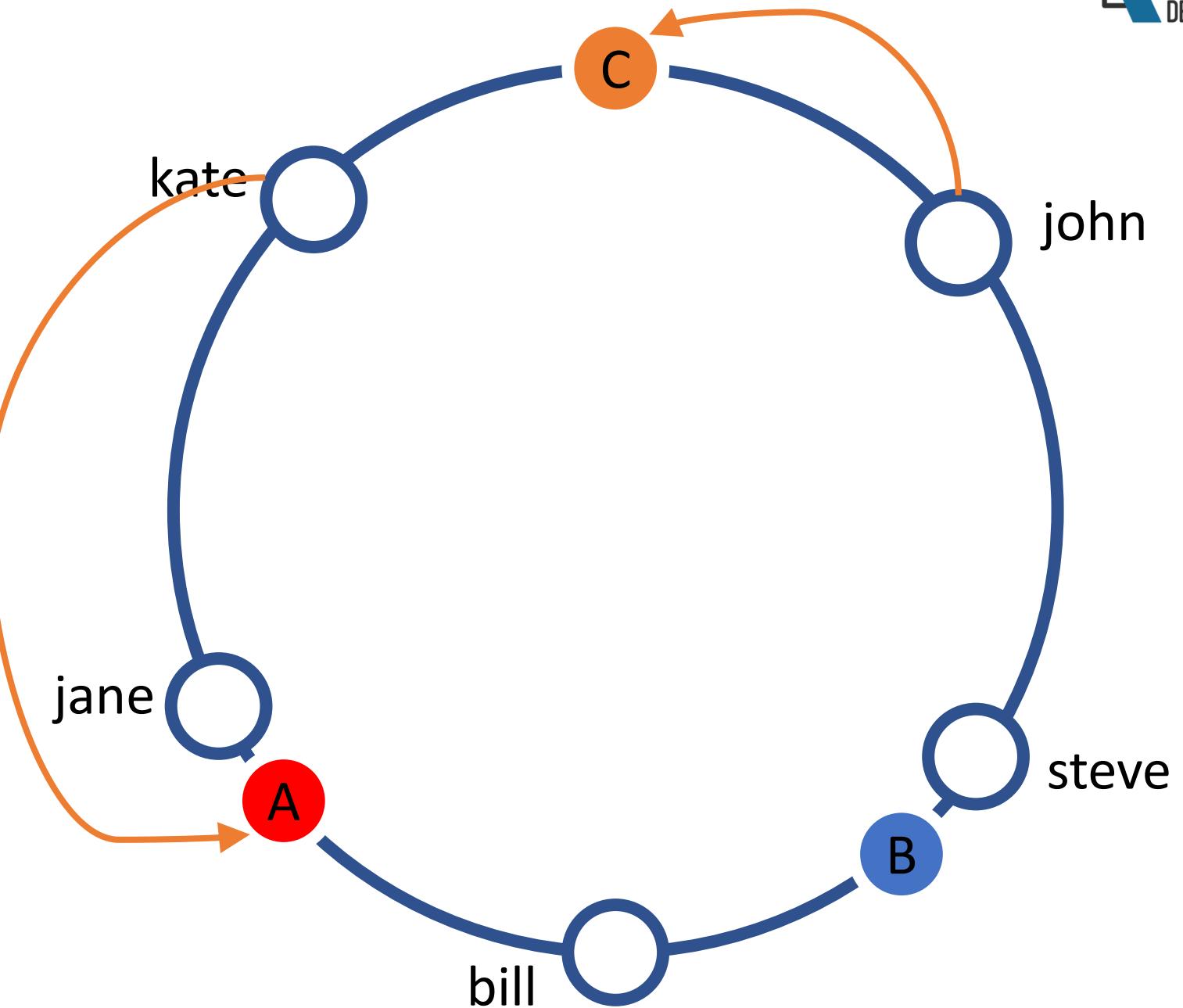
Marcando Dados

Chave	Hash	Angulo	Serv
John	1633428562	58.8	C
kate	3421831276	123.2	
Jane	5000648311	180	
bill	7594873884	273,4	
Steve	9786437450	352.3	



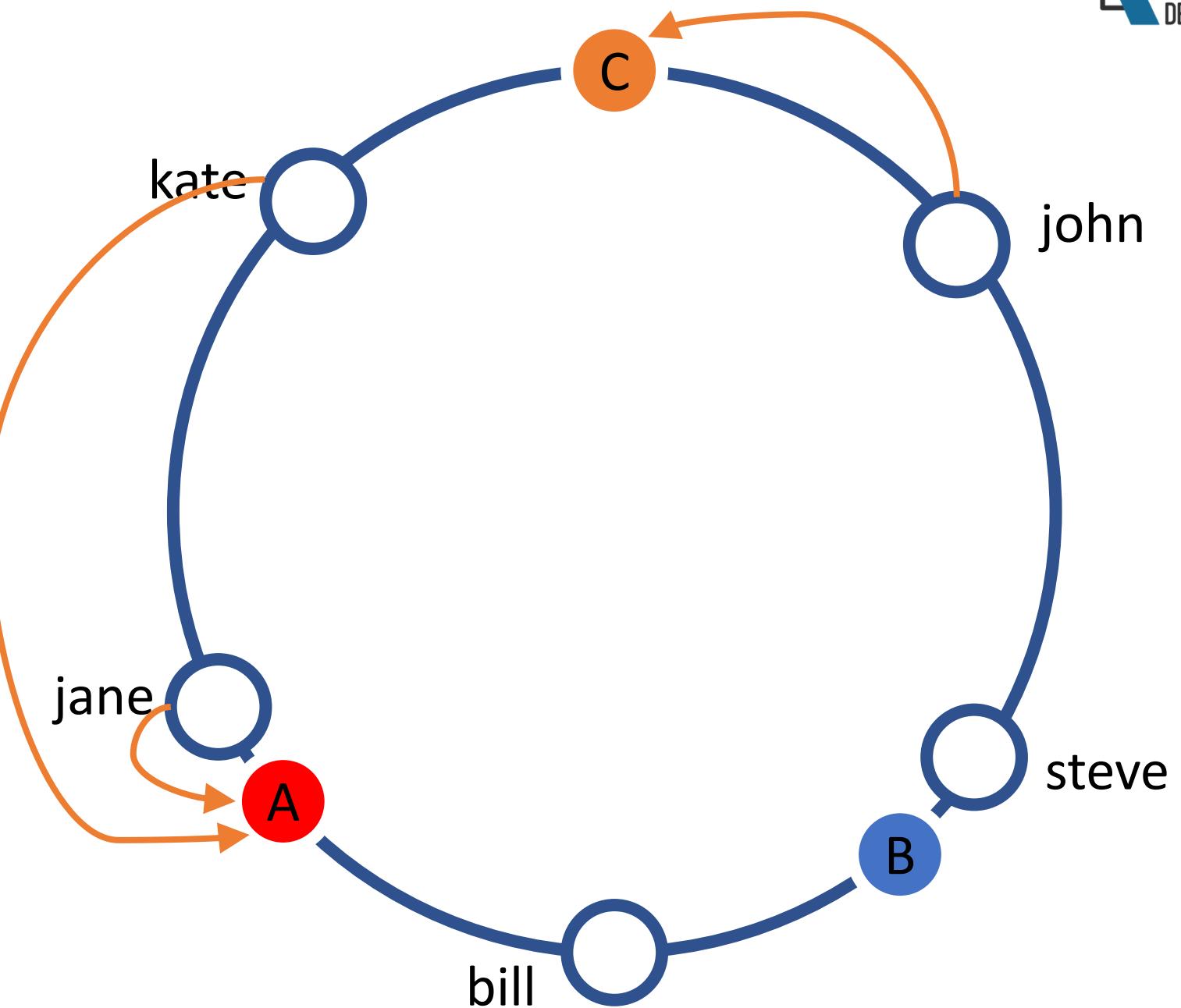
Marcando Dados

Chave	Hash	Angulo	Serv
John	1633428562	58.8	C
kate	3421831276	123.2	A
Jane	5000648311	180	
bill	7594873884	273,4	
Steve	9786437450	352.3	



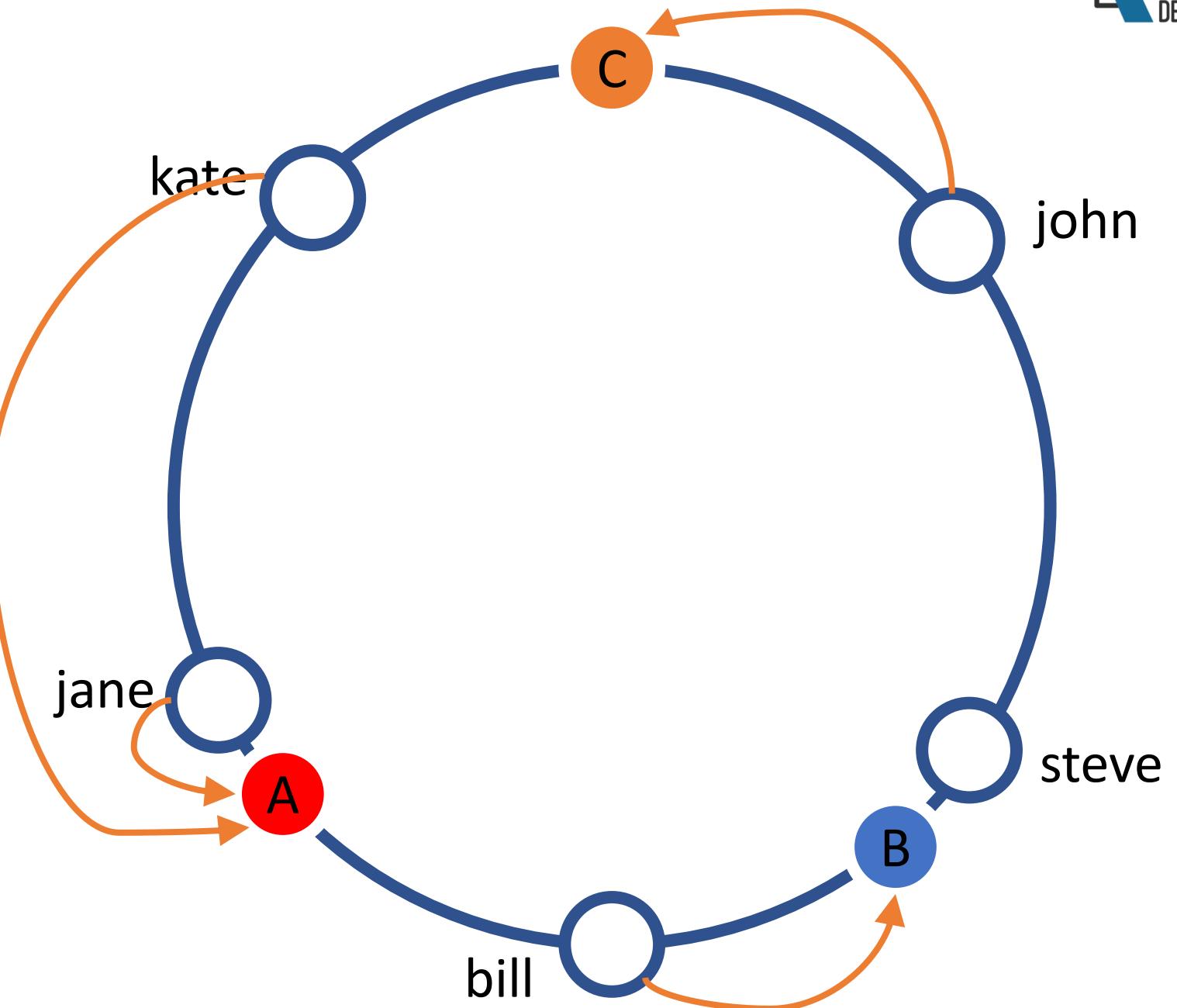
Marcando Dados

Chave	Hash	Angulo	Serv
John	1633428562	58.8	C
kate	3421831276	123.2	A
Jane	5000648311	180	A
bill	7594873884	273,4	
Steve	9786437450	352.3	



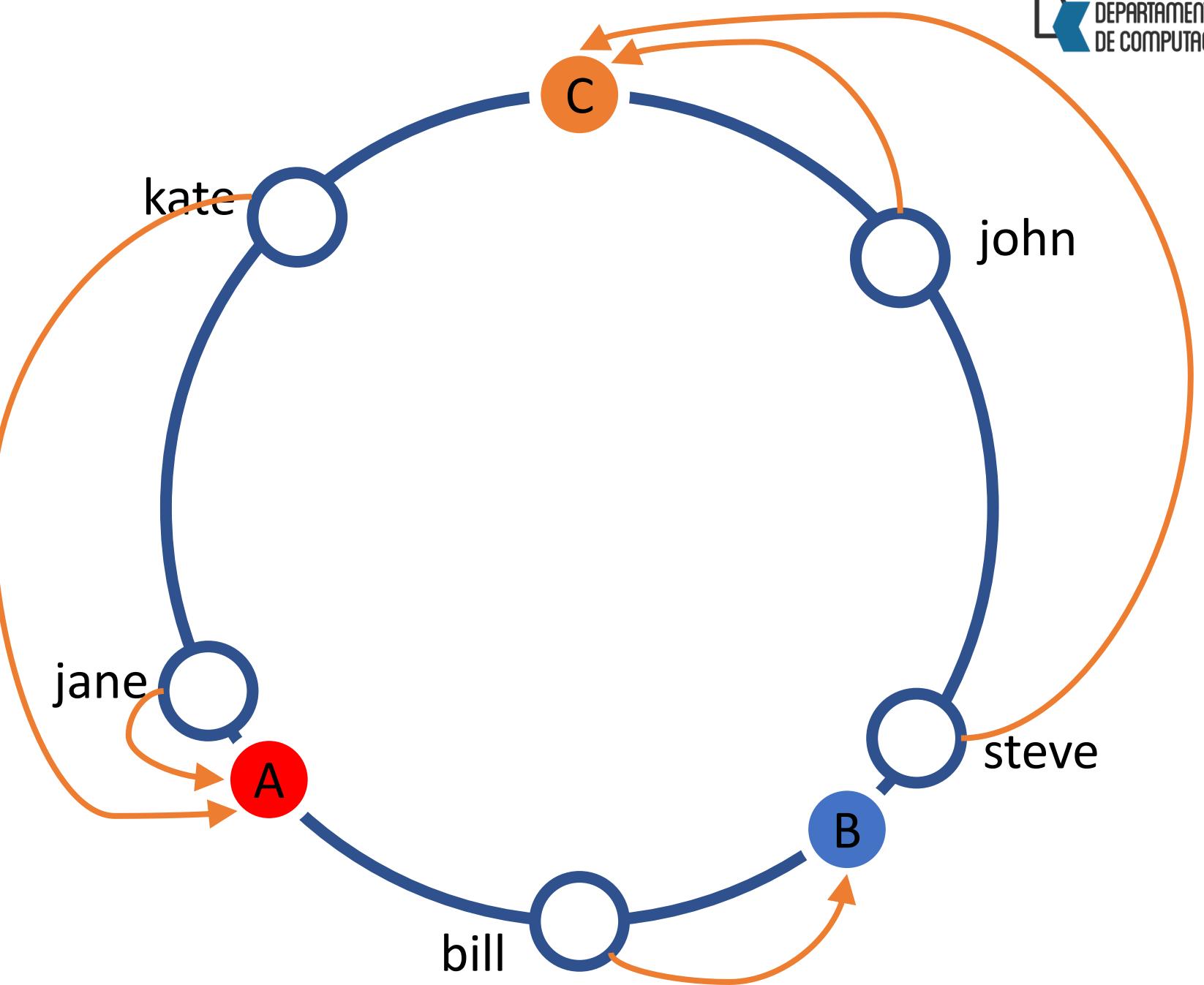
Marcando Dados

Chave	Hash	Angulo	Serv
John	1633428562	58.8	C
kate	3421831276	123.2	A
Jane	5000648311	180	A
bill	7594873884	273,4	B
Steve	9786437450	352.3	



Marcando Dados

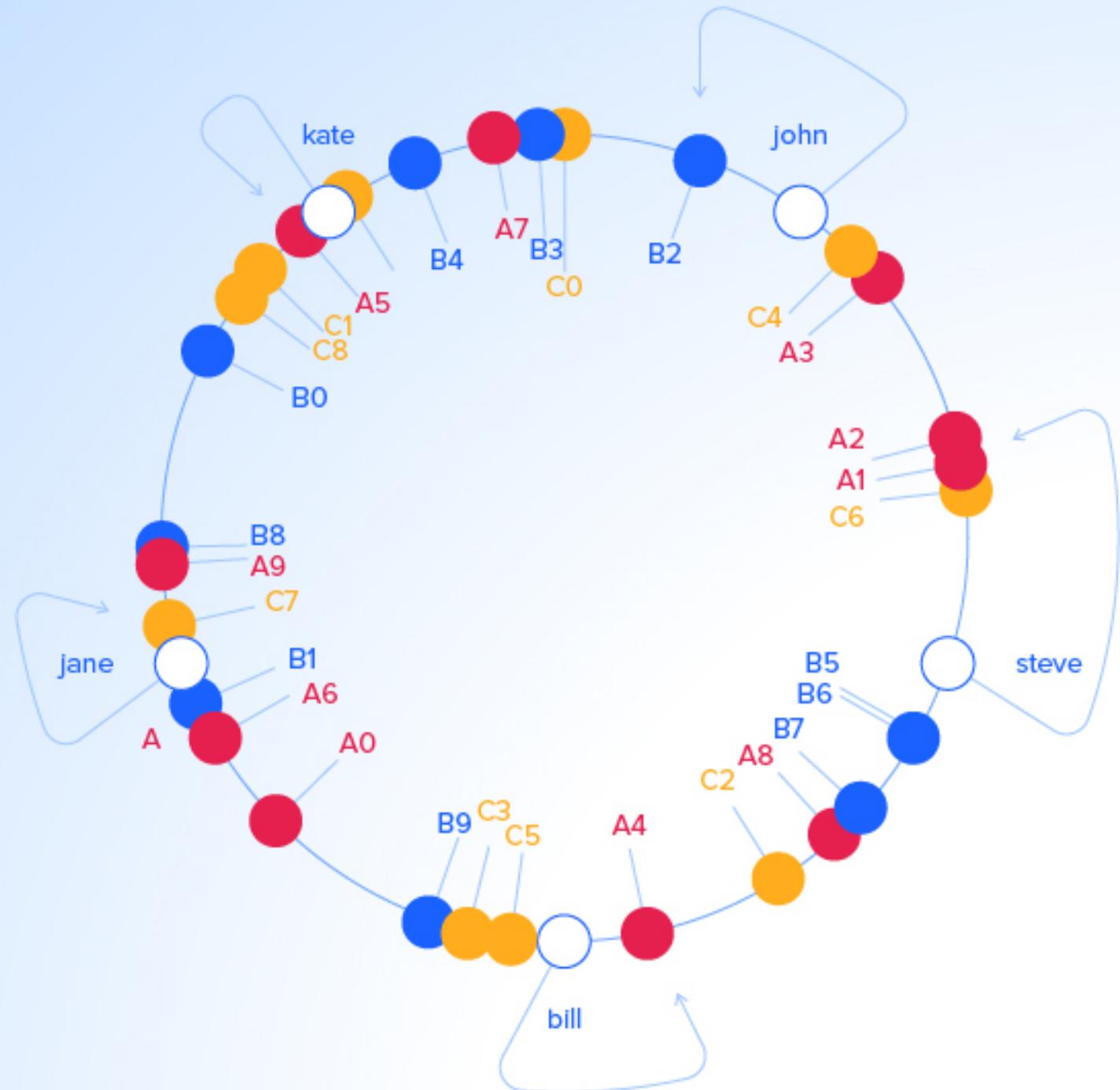
Chave	Hash	Angulo	Serv
John	1633428562	58.8	C
kate	3421831276	123.2	A
Jane	5000648311	180	A
bill	7594873884	273,4	B
Steve	9786437450	352.3	C



Como melhor distribuir?

- Determinado conteúdo pode ser mais acessado que outro...
- Um servidor pode ser mais robusto que outro...
- Solução criar *labels* como servidores fictícios que mapeiam para os servidores reais
- No nosso exemplo:
 - Ao..A9, Bo...B9, Co...C9

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	C6	C



Ok, mas qual o benefício?

- Imagine que o Servidor C, caiu. O que fazer?
- Retirar os labels associados ao servidor
- Uma possível ação: Substituir aleatoriamente por labels AX e BX

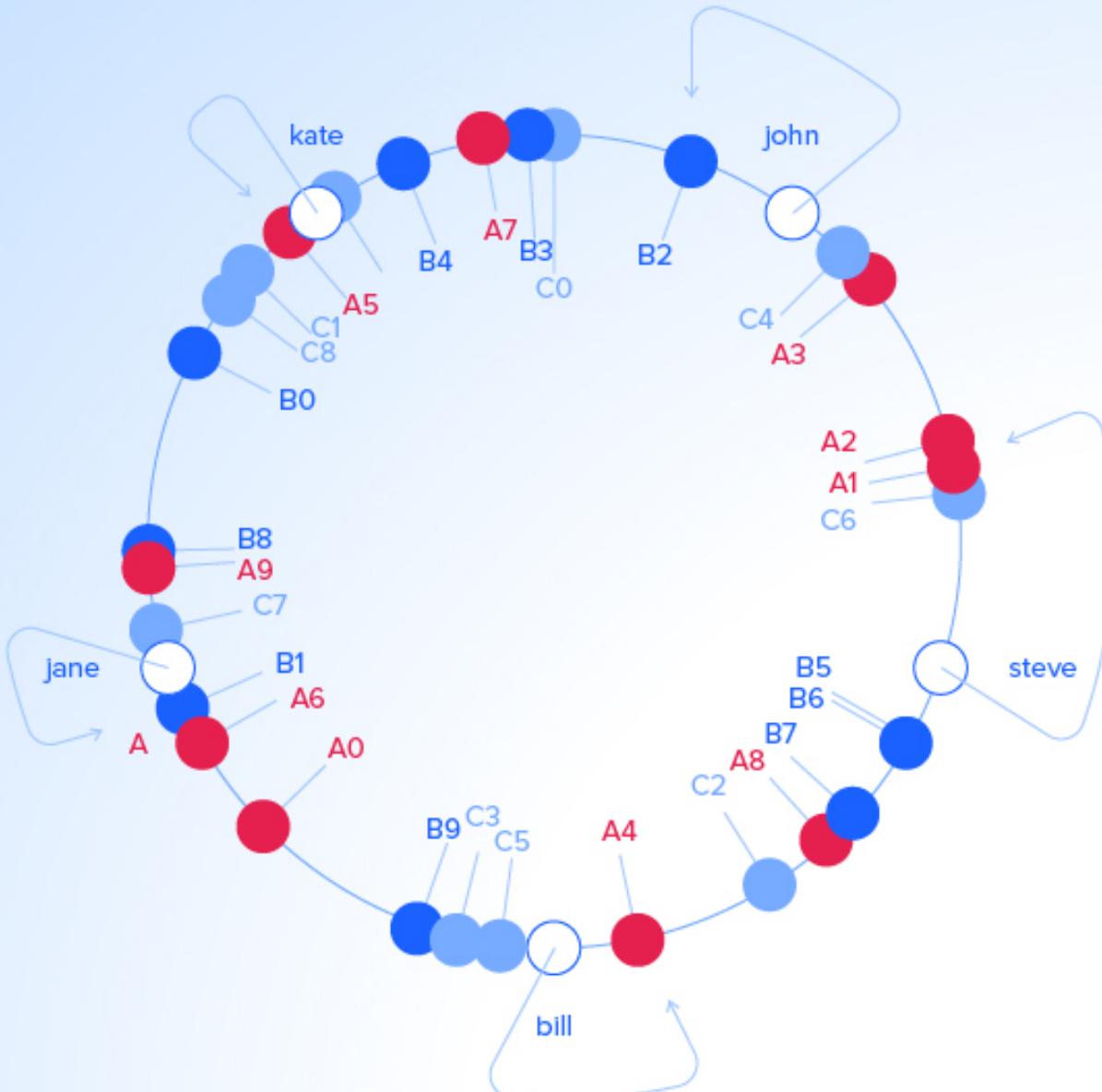


Tabela de Dispersão após a Queda do Servidor C

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	C6	C

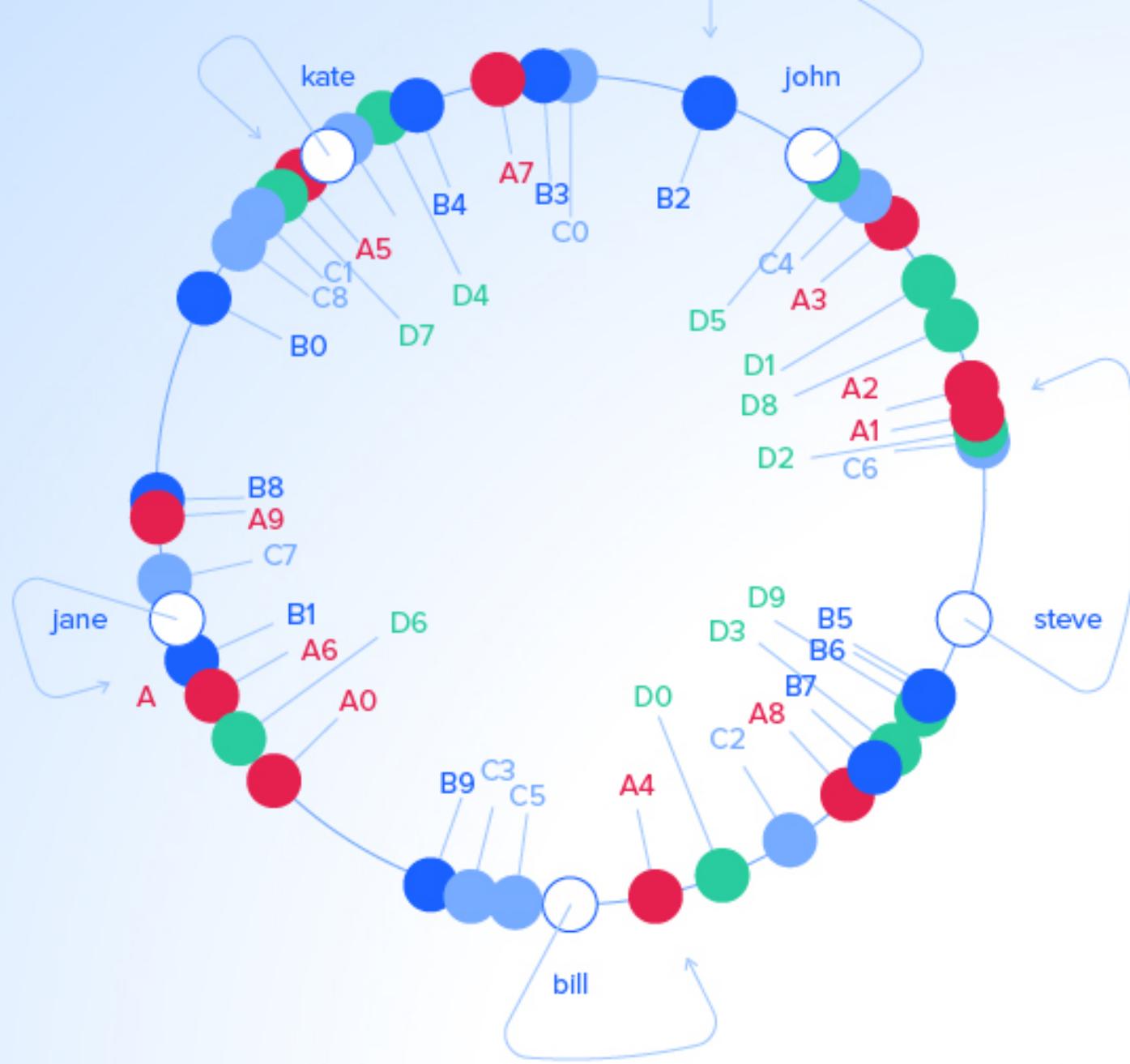
Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	A1	A

Tabela de Dispersão após a Queda do Servidor C

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	C6	C

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	A1	A

Inserir um Servidor



Após inserir um novo servidor

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	C6	C

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	D2	D

Após inserir um novo servidor

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	C6	C

Chave	Hash	Angulo	Label	Serv
John	1633428562	58.8	B2	B
kate	3421831276	123.2	A5	A
Jane	5000648311	180	B1	B
bill	7594873884	273,4	A4	A
Steve	9786437450	352.3	D2	D

Em geral, apenas k/N chaves precisam ser remapeadas

Quem usa Consistent Hashing?

- Key applications
 - Armazenamento Key-value
 - Busca de Conteúdo P2P
 - Deduplicação
- Produtos
 - Cassandra Database
 - Akamai Content Delivery
 - Couchbase data partitioning
 - Openstack Data Storage - Swift
 - AWS Dynamo

Bons links

- A Guide to Consistent Hashing
 - <https://www.toptal.com/big-data/consistent-hashing>
- Vídeos
 - Consistent Hashing (<https://www.youtube.com/watch?v=viaNG1zyx1g>)
 - Distributed Systems in one Lesson (<https://www.youtube.com/watch?v=Y6Ev8Gllbxc&t=1442s>)