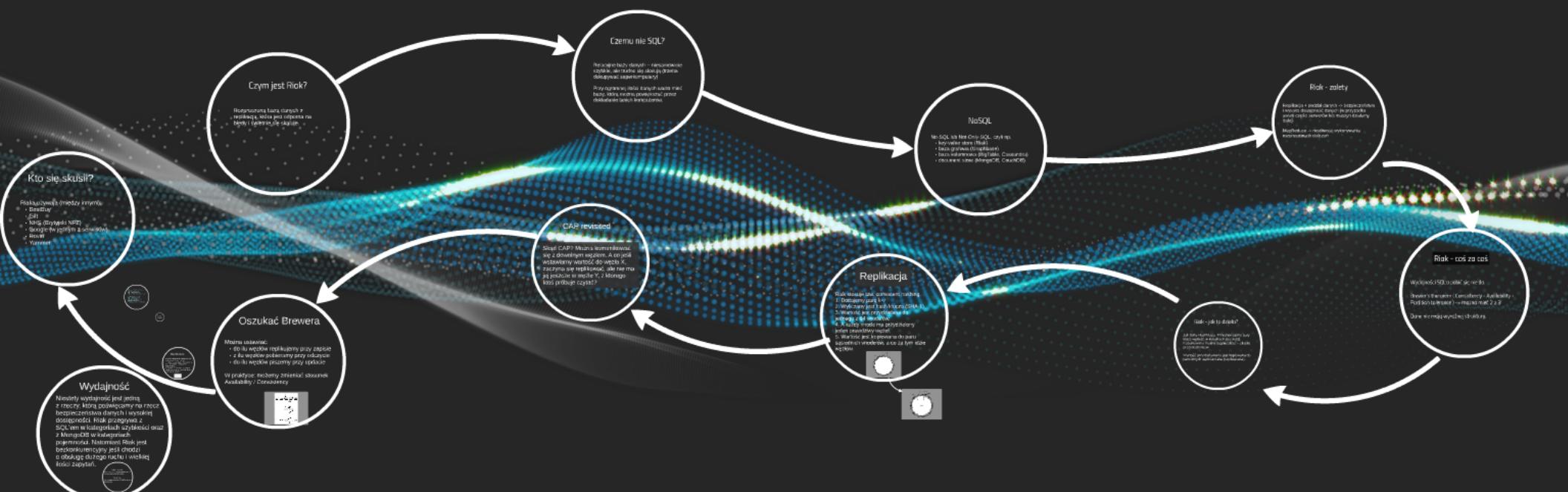
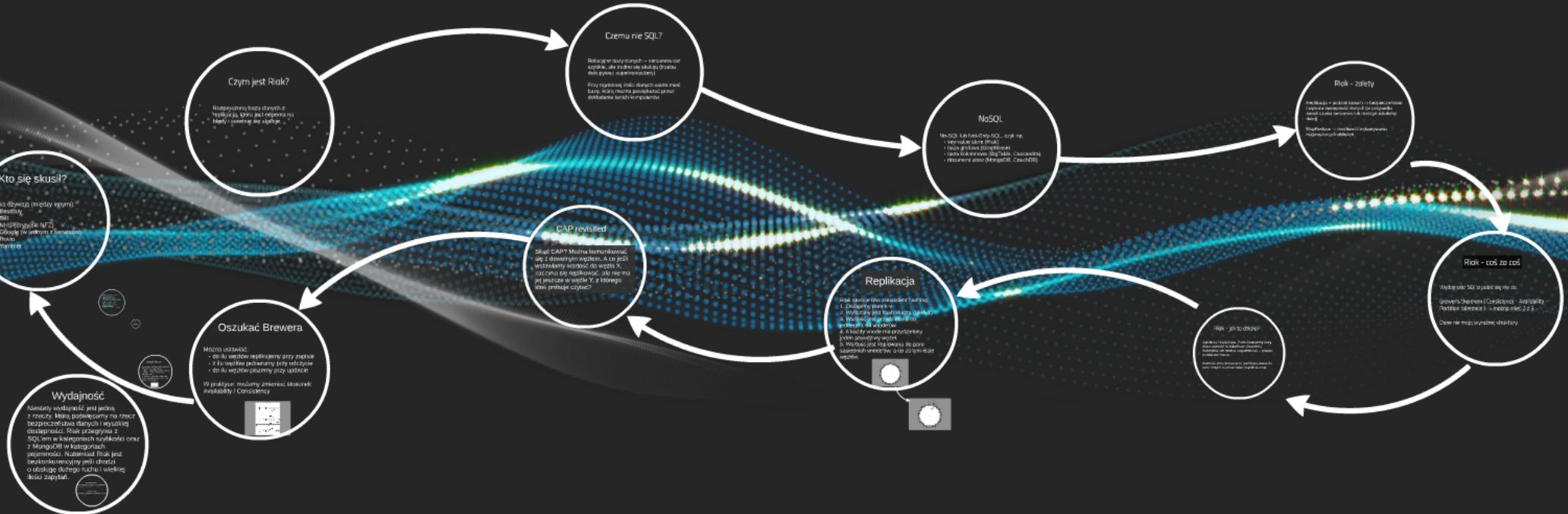


Riak



Riak



Czym jest Riak?

Rozproszoną bazą danych z replikacją, która jest odporna na błędy i świetnie się skaluje.

Czemu nie SQL?

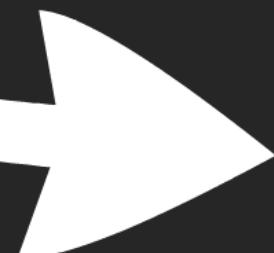
Relacyjne bazy danych -- niesamowicie szybkie, ale trudno się skalują (trzeba dokupywać superkomputery)

Przy ogromnej ilości danych warto mieć bazę, którą można powiększać przez dokładanie tanich komputerów.

NoSQL

No-SQL lub Not-Only-SQL, czyli np.

- key-value store (Riak)
- baza grafowa (Graphbase)
- baza kolumnowa (BigTable, Cassandra)
- document store (MongoDB, CouchDB)



Riak - zalety

Replikacja + podział danych -> bezpieczeństwo i wysoka dostępność danych (w przypadku awarii części serwerów lub maszyn działamy dalej)

MapReduce -> możliwość wykonywania rozproszonych obliczeń



Riak - coś za coś

Wydajności SQL'a pobić się nie da.

Brewer's theorem (Consistency - Availability - Partition tolerance) -> można mieć 2 z 3

Dane nie mają wyraźnej struktury.

Riak - jak to działa?

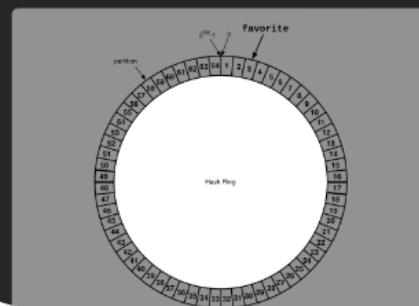
Jak duża HashMapa. Przechowujemy pary klucz-wartość w kubełkach (buckets). Kubełków nie można zagnieździć -- płaska przestrzeń nazw.

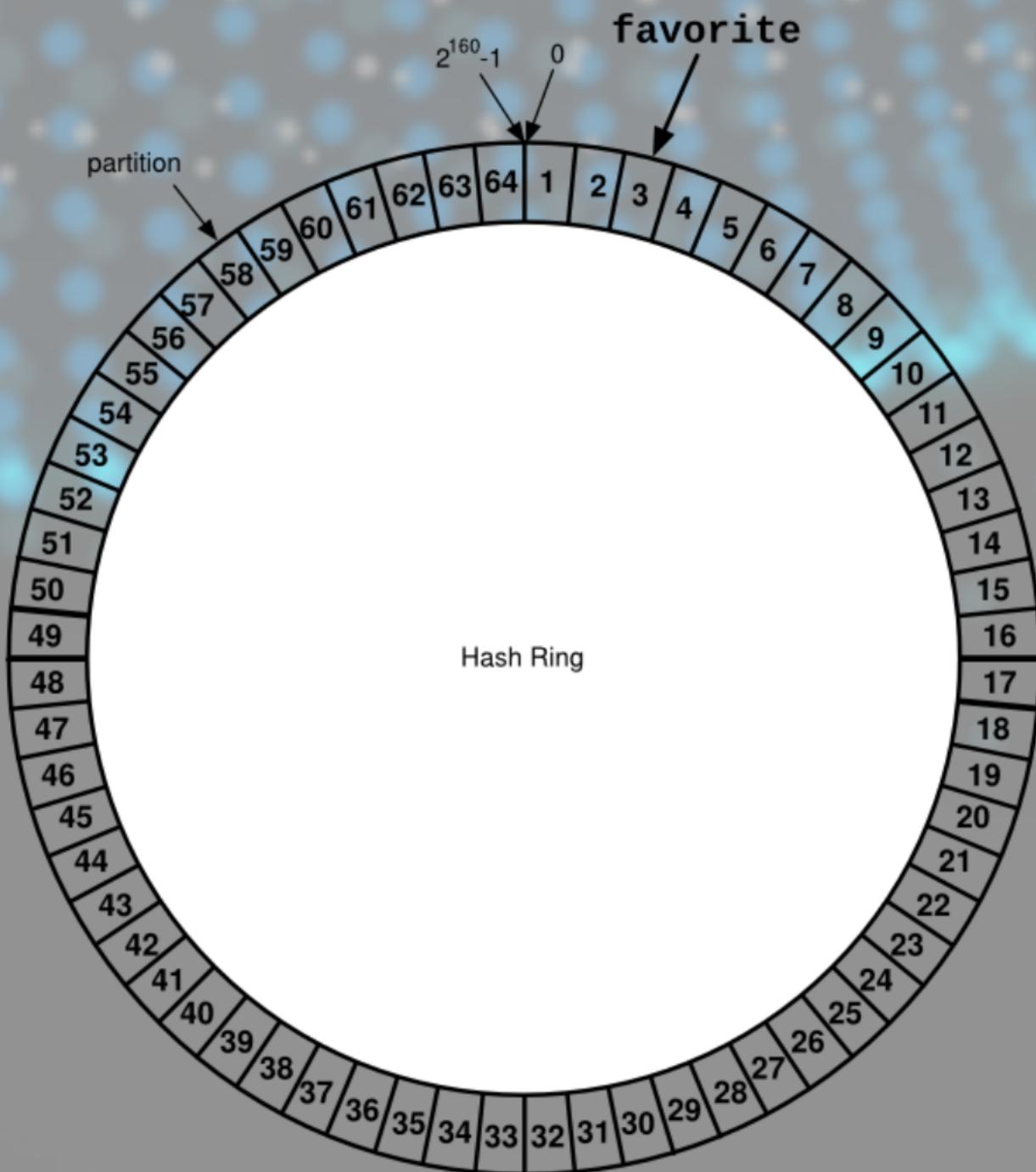
Wartość przy dodawaniu jest kopiowana do paru innych węzłów riaka (replikowana).

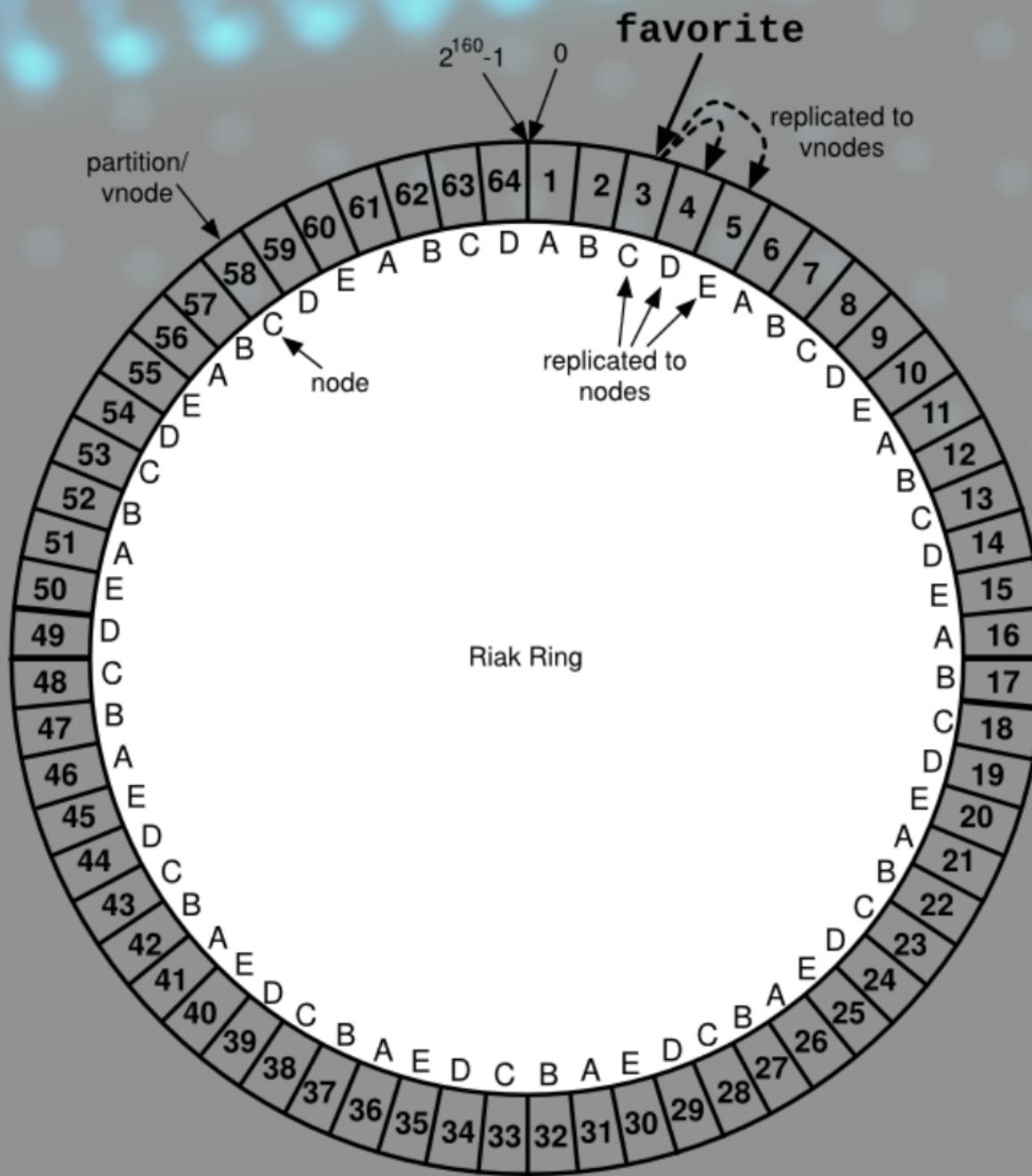
Replikacja

Riak stosuje tzw. consistent hashing.

1. Dodajemy parę k-v
2. Wyliczany jest hash klucza (SHA-1)
3. Wartość jest przydzielana do jednego z 64 vnode'ów.
4. A każdy vnode ma przydzielony jeden prawdziwy węzeł.
5. Wartość jest kopowana do paru sąsiednich vnode'ów, a co za tym idzie węzłów.







CAP revisited

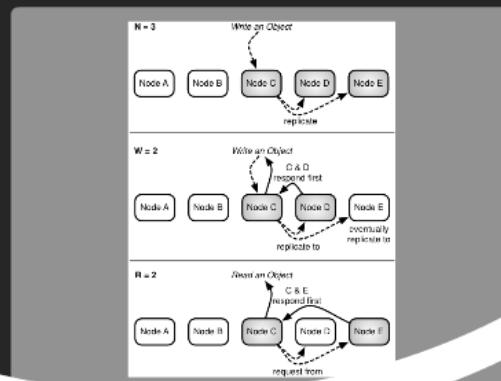
Skąd CAP? Można komunikować się z dowolnym węzłem. A co jeśli wstawiamy wartość do węzła X, zaczyna się replikować, ale nie ma jej jeszcze w węźle Y, z którego ktoś próbuje czytać?

Oszukać Brewera

Można ustawać:

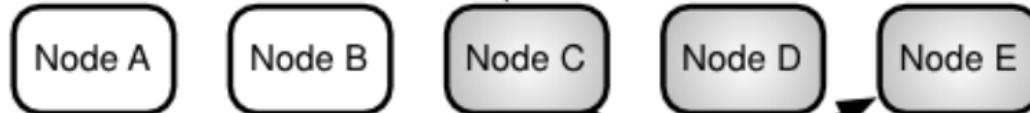
- do ilu węzłów replikujemy przy zapisie
- z ilu węzłów pobieramy przy odczycie
- do ilu węzłów piszemy przy updacie

W praktyce: możemy zmieniać stosunek
Availability / Consistency



N = 3

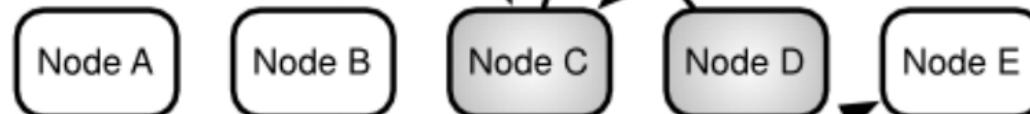
Write an Object



replicate

W = 2

Write an Object



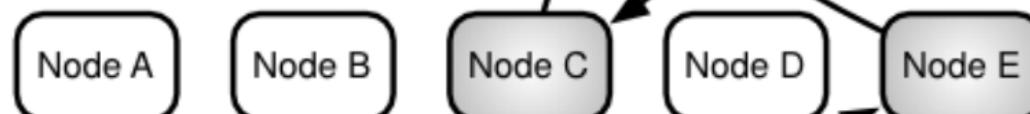
C & D
respond first

eventually
replicate to

replicate to

R = 2

Read an Object



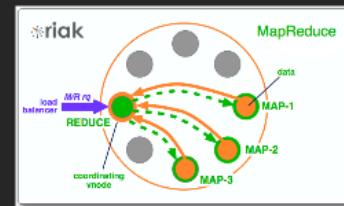
C & E
respond first

request from

MapReduce

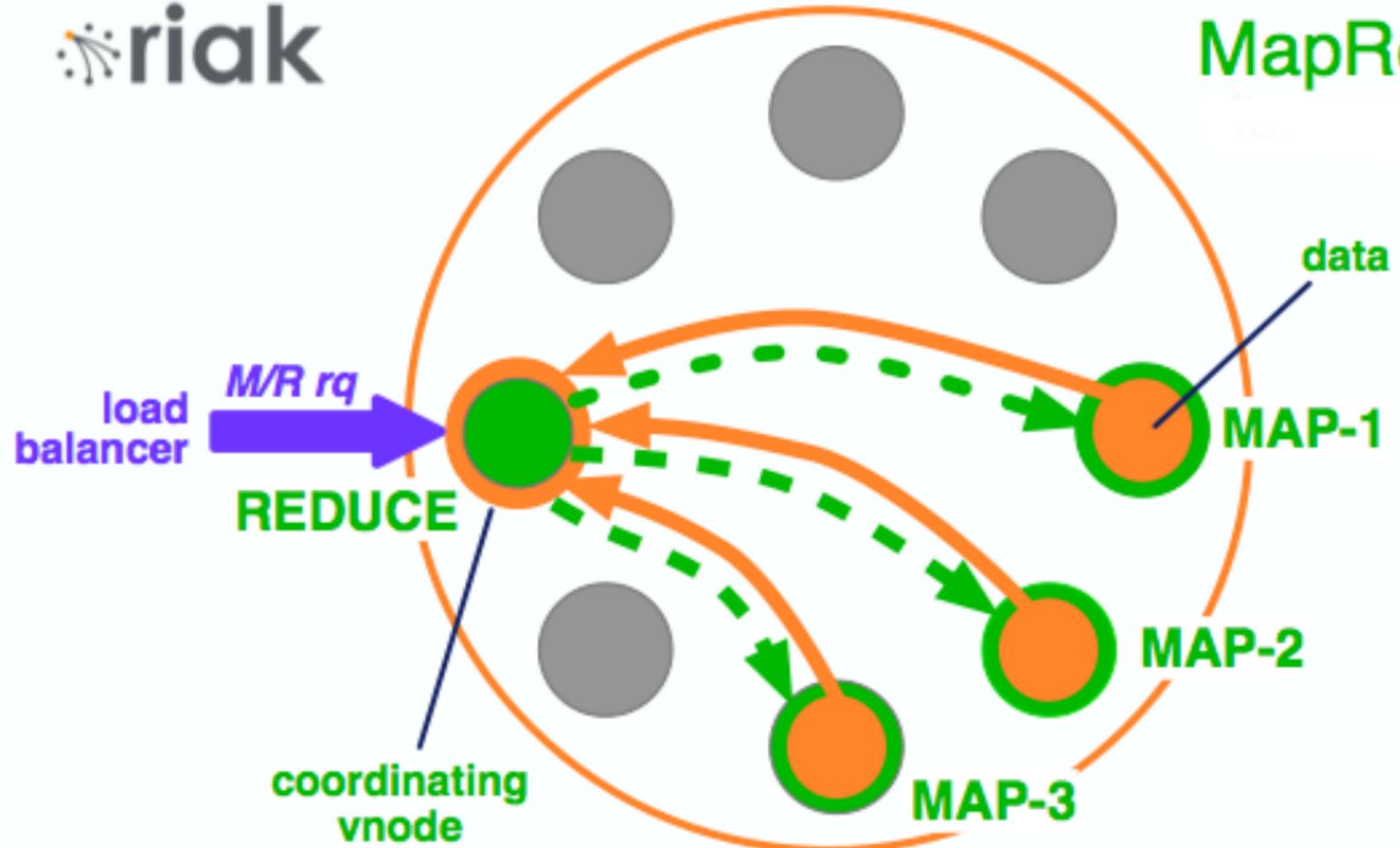
Sposób prowadzenia rozproszonych obliczeń na dużych danych.

1. Map -> węzeł-matka dzieli zadanie na wiele części i rozsyła je do węzłów-dzieci
2. Reduce -> dzieci po wykonaniu zadania odsyłają je do matki, która scalą wyniki w jedno.





MapReduce





Wydajność

Niestety wydajność jest jedną z rzeczy, którą poświęcamy na rzecz bezpieczeństwa danych i wysokiej dostępności. Riak przegrywa z SQL'em w kategoriach szybkości oraz z MongoDB w kategoriach pojemności. Natomiast Riak jest bezkonkurencyjny jeśli chodzi o obsługę dużego ruchu i wielkiej ilości zapytań.

Kubei zimnej wody:

<http://mdshannan1.blogspot.com/2012/05/riak-vs-mysql-performance-tests-1.html>

Success story:

<http://tech.gilt.com/post/55529520304/riak-passes-the-stress-test>

Kubel zimnej wody:

<http://mdshannan1.blogspot.com/2012/05/riak-vs-mysql-performance-tests-1.html>

Success story:

<http://tech.gilt.com/post/55529520304/riak-passes-the-stress-test>

Kto się skusił?

Riaka używają (między innymi):

- BestBuy
- Gilt
- NHS (Brytyjski NFZ)
- Google (w jednym z serwisów)
- Rovio
- Yammer

Informacje pochodzą ze stron:

<http://docs.basho.com/>

<http://littleriakbook.com/>

<http://mdshannan1.blogspot.com/>

Grafiki pochodzą ze stron:

<http://littleriakbook.com/>

[http://www.cbsolution.net/
techniques/ontarget/](http://www.cbsolution.net/techniques/ontarget/)

[mapreduce_vs_data_warehouse](http://www.cbsolution.net/techniques/ontarget/mapreduce_vs_data_warehouse)

Dziękujemy!

Hubert Słojewski
Piotr Moczurad