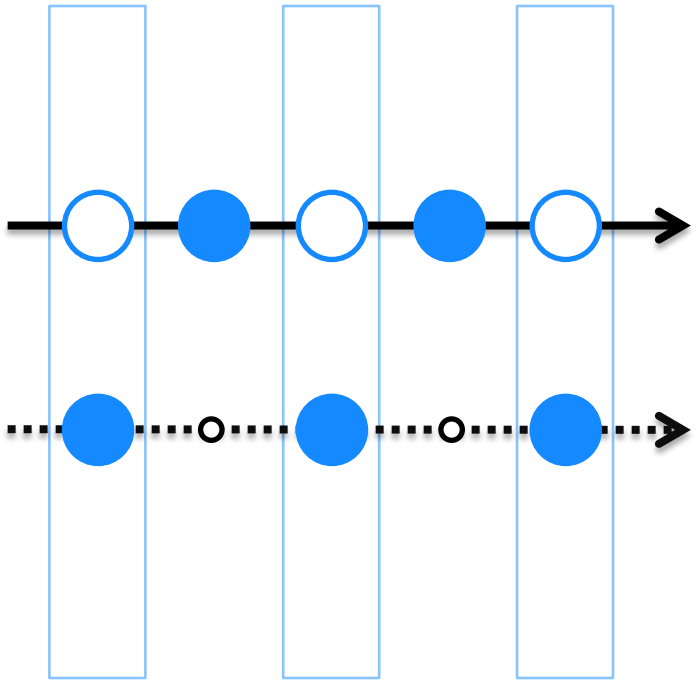
Spreads

Series and Panels for Real-time and Exploratory Analysis of Data Streams



Contents

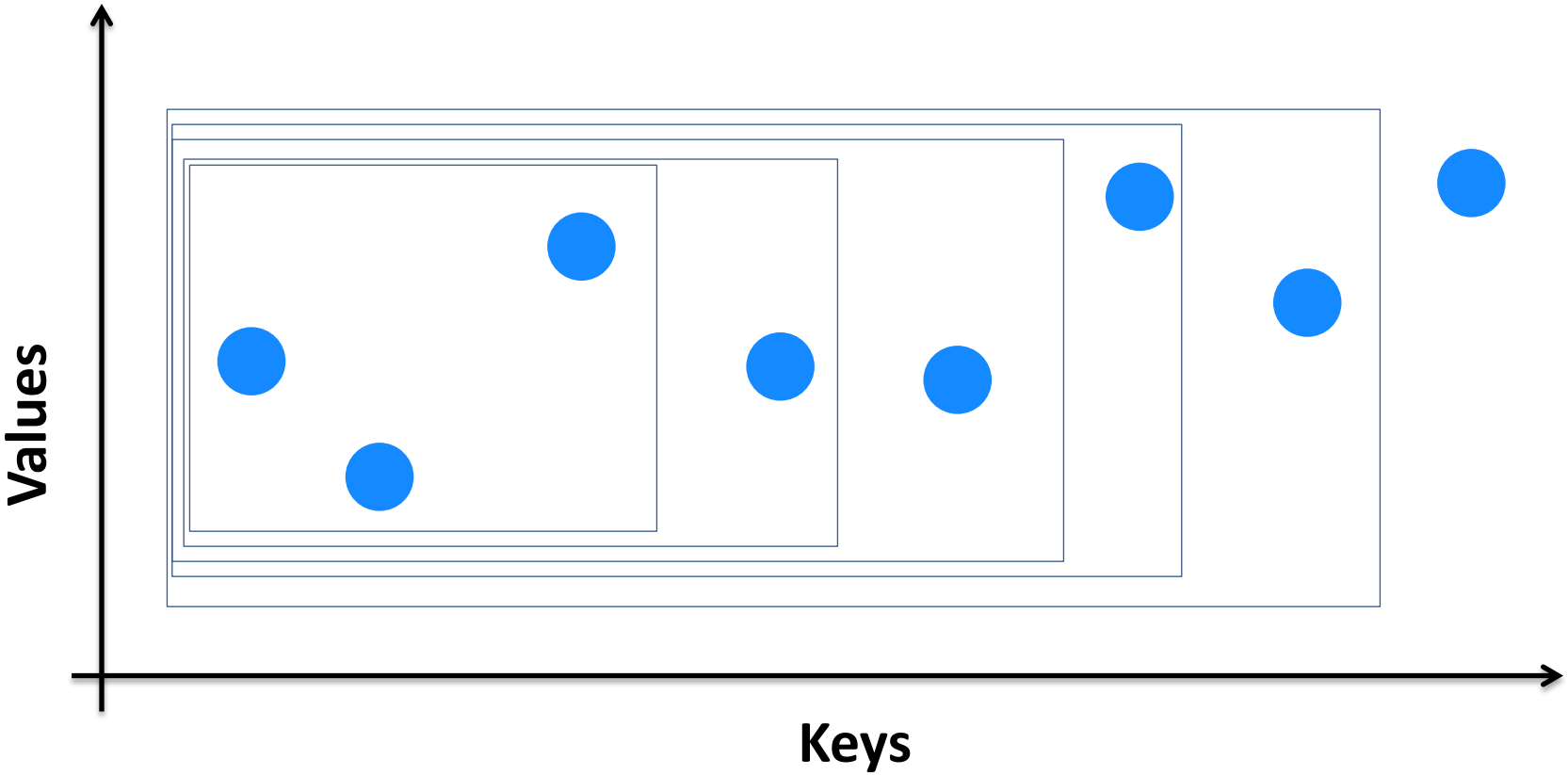
- **Declarative series manipulation**
- **Zip N**
- **Live repository**
- **Integration with R**
- **Extensions**

- **Declarative series manipulation**
- **Zip N**
- **Live repository**
- **Integration with R**
- **Extensions**

What is series

- A sequence of keyed **values** which is:
  - **Unbounded** or **complete**;
  - **Navigable** and **ordered** by **sorted keys** or **index**;
  - Exists **regardless of storage space**: not a column on a spreadsheet or arrays in memory, but points in key-value dimensions. Physical representation is an implementation details;
  - Usually is an **attribute** of an **identity**;
  - Mutable as object, **immutable** as data.

What is series



Series data is usually immutable and append only. However, we do not restrict history rewrites at mutable containers level.

What is series

- Mathematical Function
  - Wiki: In *mathematics*, a *function* is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.
  - Series is a getter of a *cursor* that acts as a function:

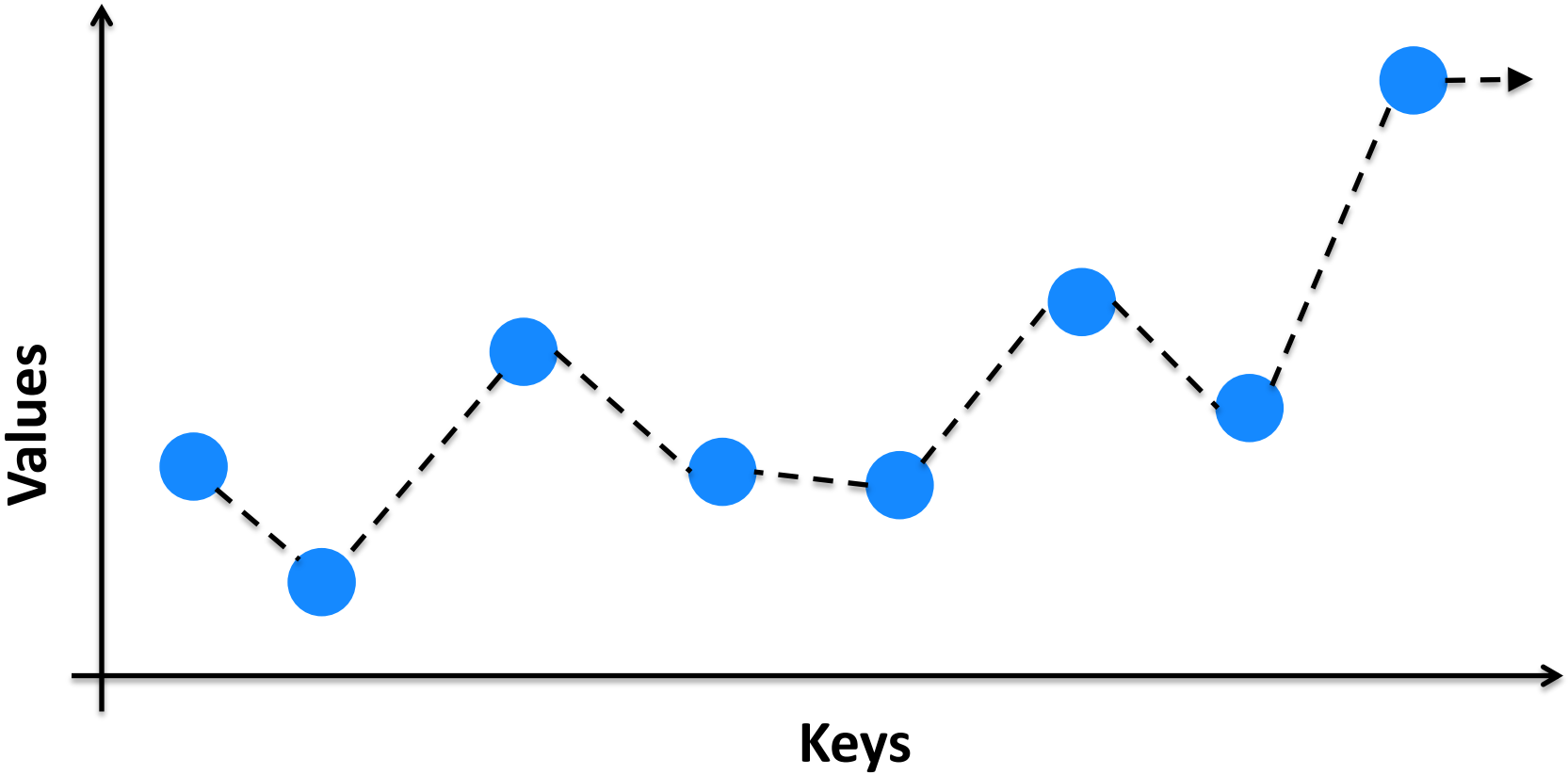
    Series<K,V> = unit -> (K -> (K*V) opt)
  - Series is a *functor*:

    Series<K,V> -> ((K*V) opt -> (K*U) opt) -> Series<K,U>

What is series

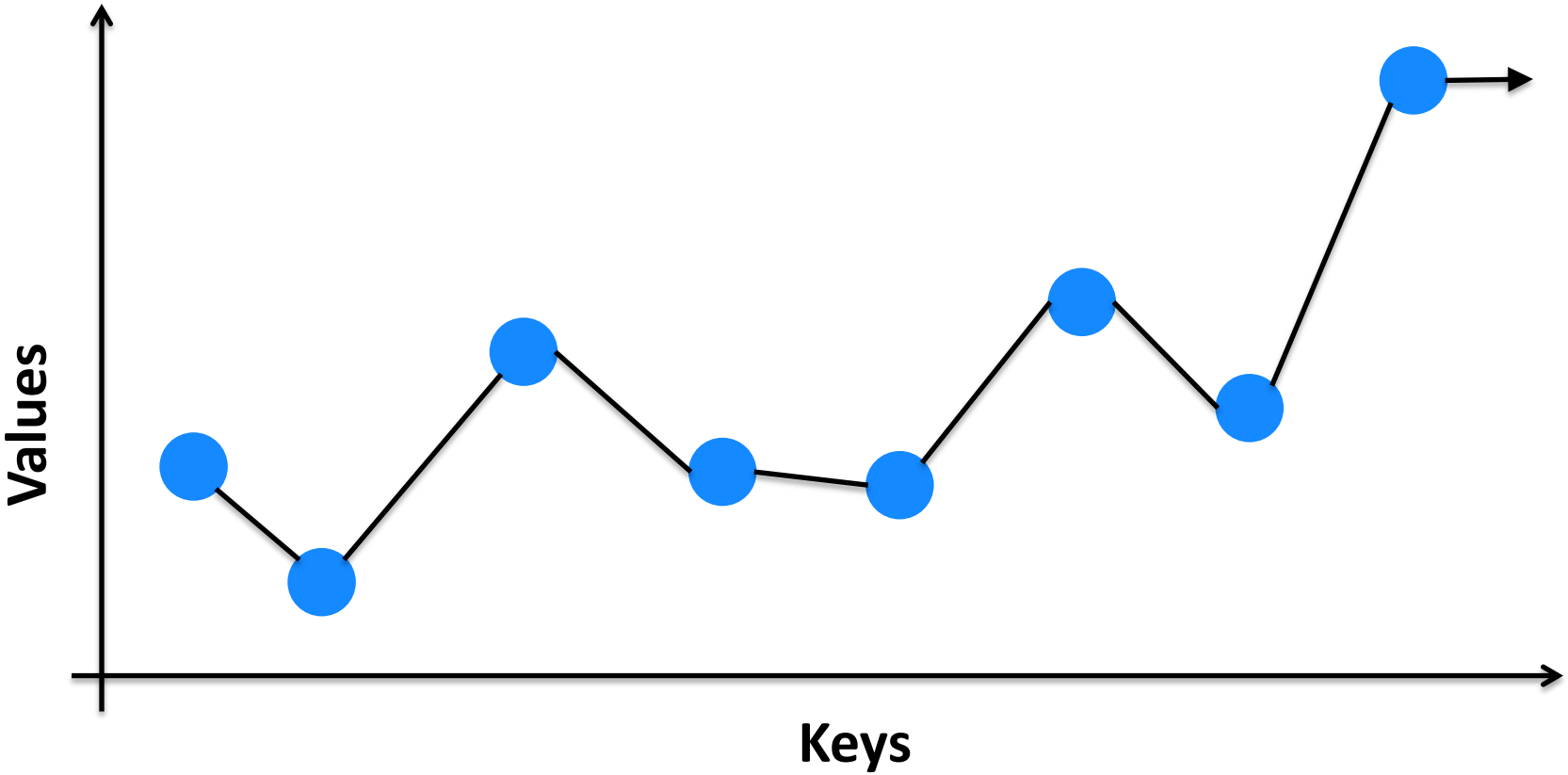
- IDictionary<TKey, TValue>
  - For a drop-in replacement of many .NET SCG interfaces with additional functionality such as persistence
- IObservable<KeyValuePair<TKey, TValue>>
  - For pushing data to consumers and integrating with Reactive Extensions
- IAsyncEnumerable<KeyValuePair<TKey, TValue>>
  - For pulling data by consumers and integrating with Interactive Extensions
- Lazy Evaluation
  - Any complex calculation tree is evaluated on-demand and could be consumed via push or pull interface or evaluated into a container (e.g. ToSortedMap()).

Discrete Series



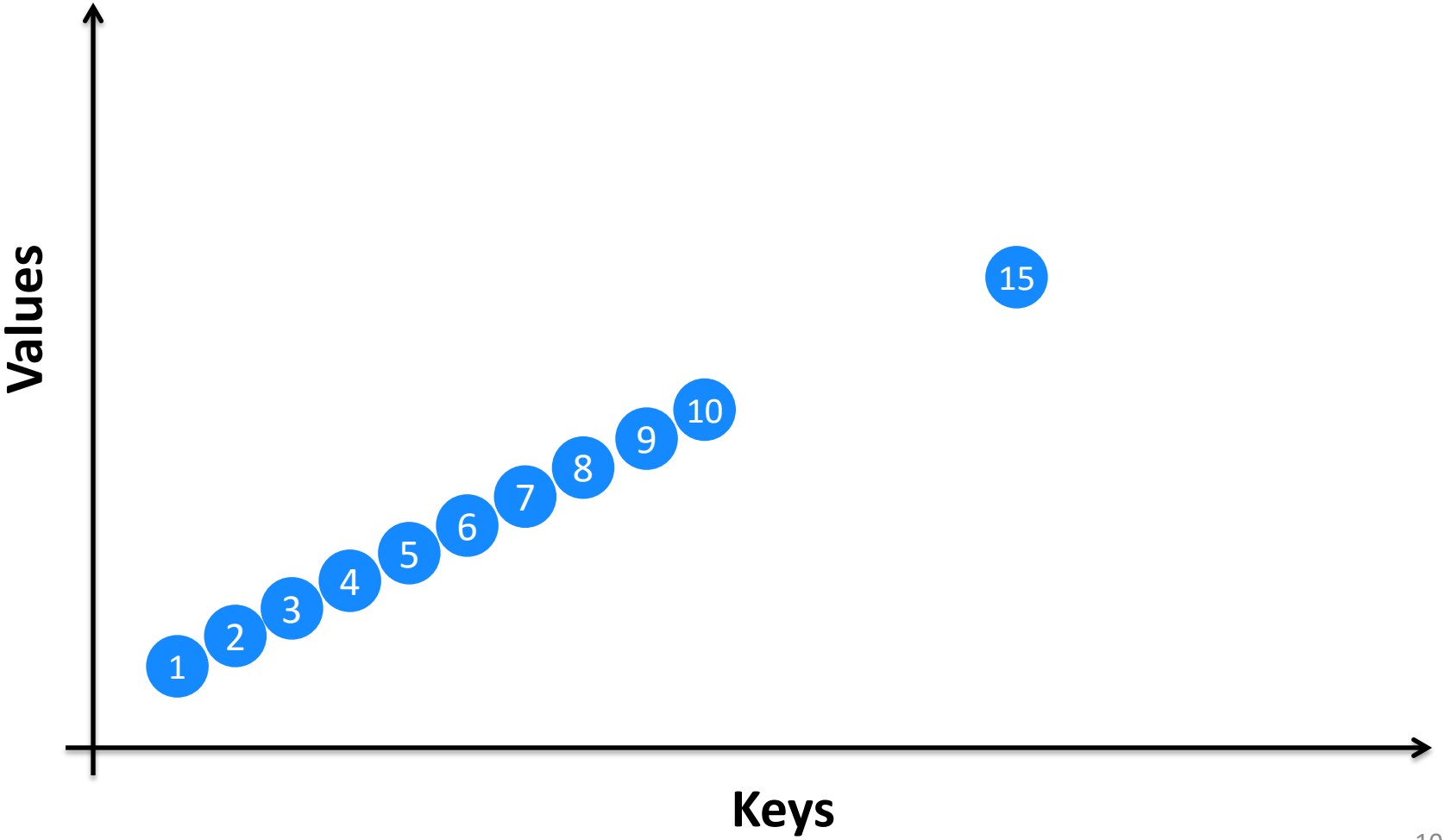
Discrete series have values defined only at observations (e.g. trade volume)

Continuous Series

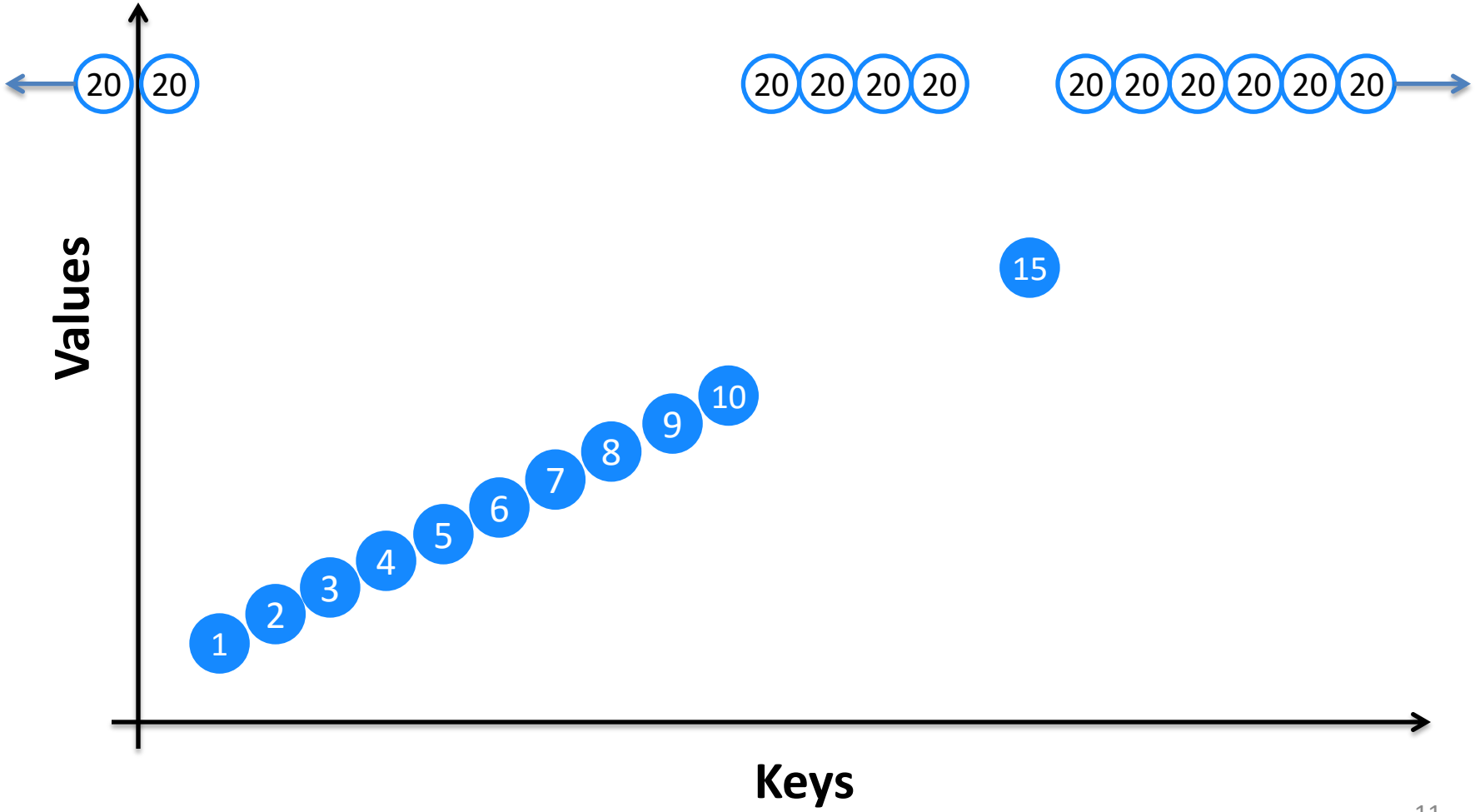


Continuous series have values defined at any key, even between observed keys (e.g. last price or cubic splines).

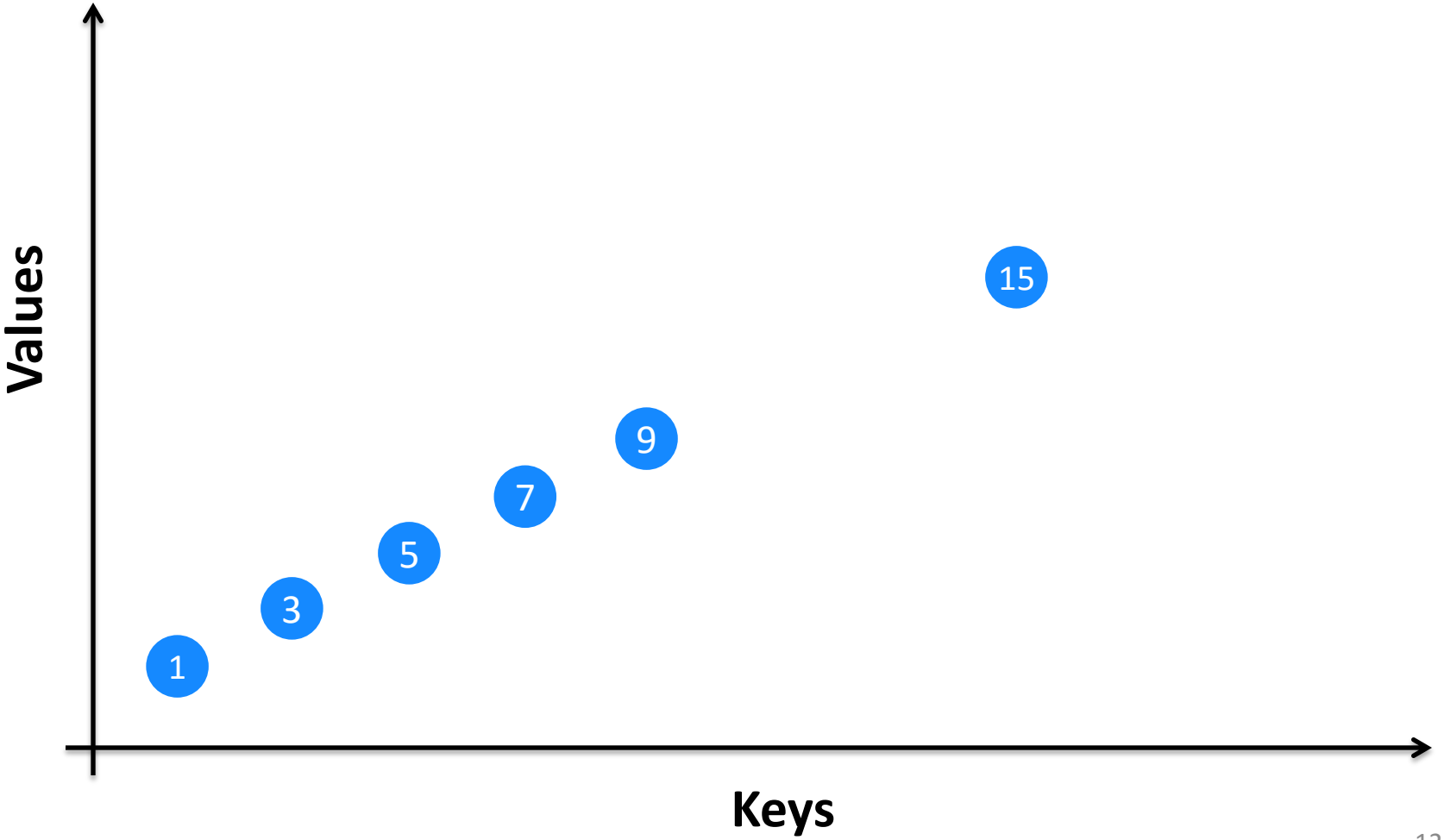
Original



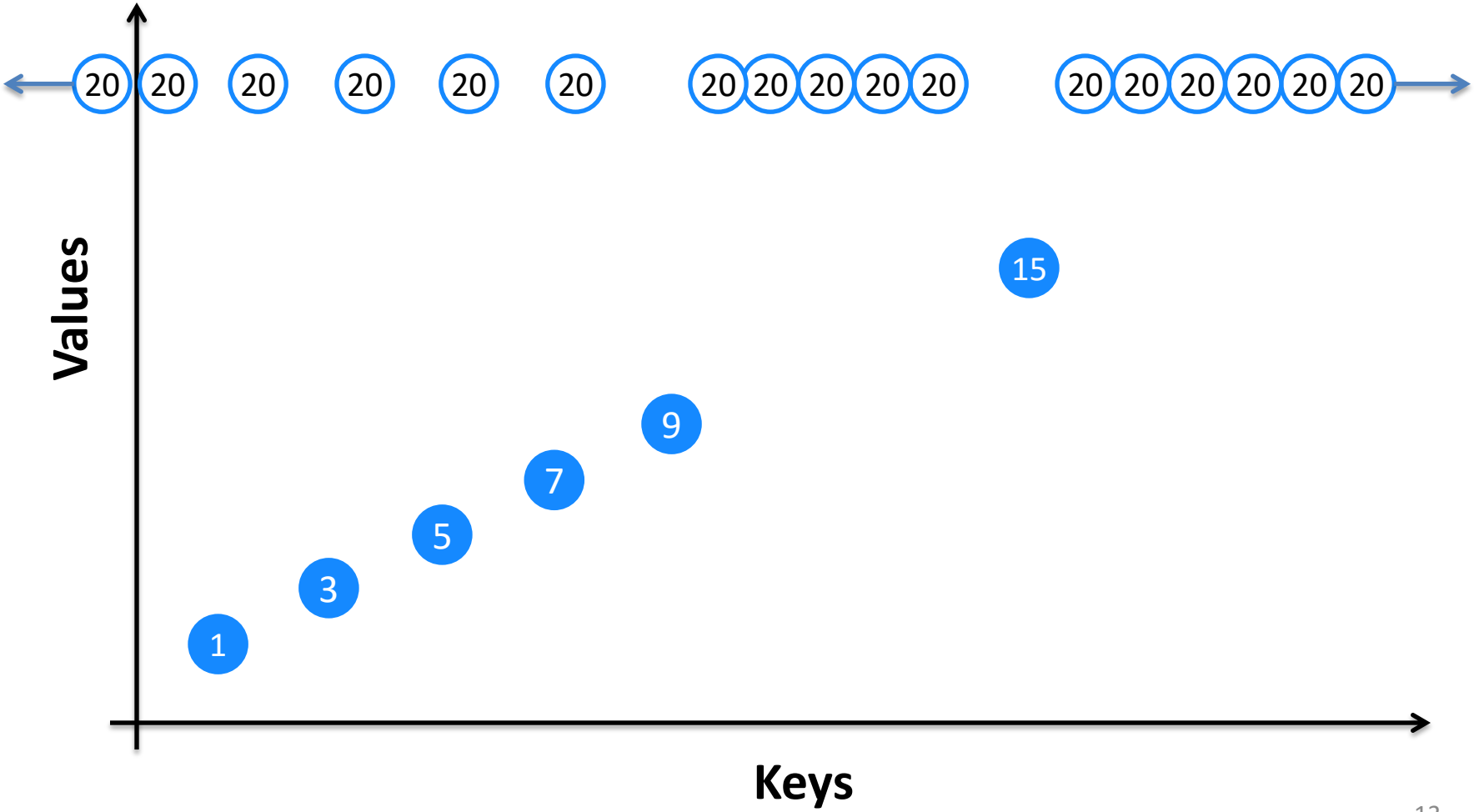
Fill 20



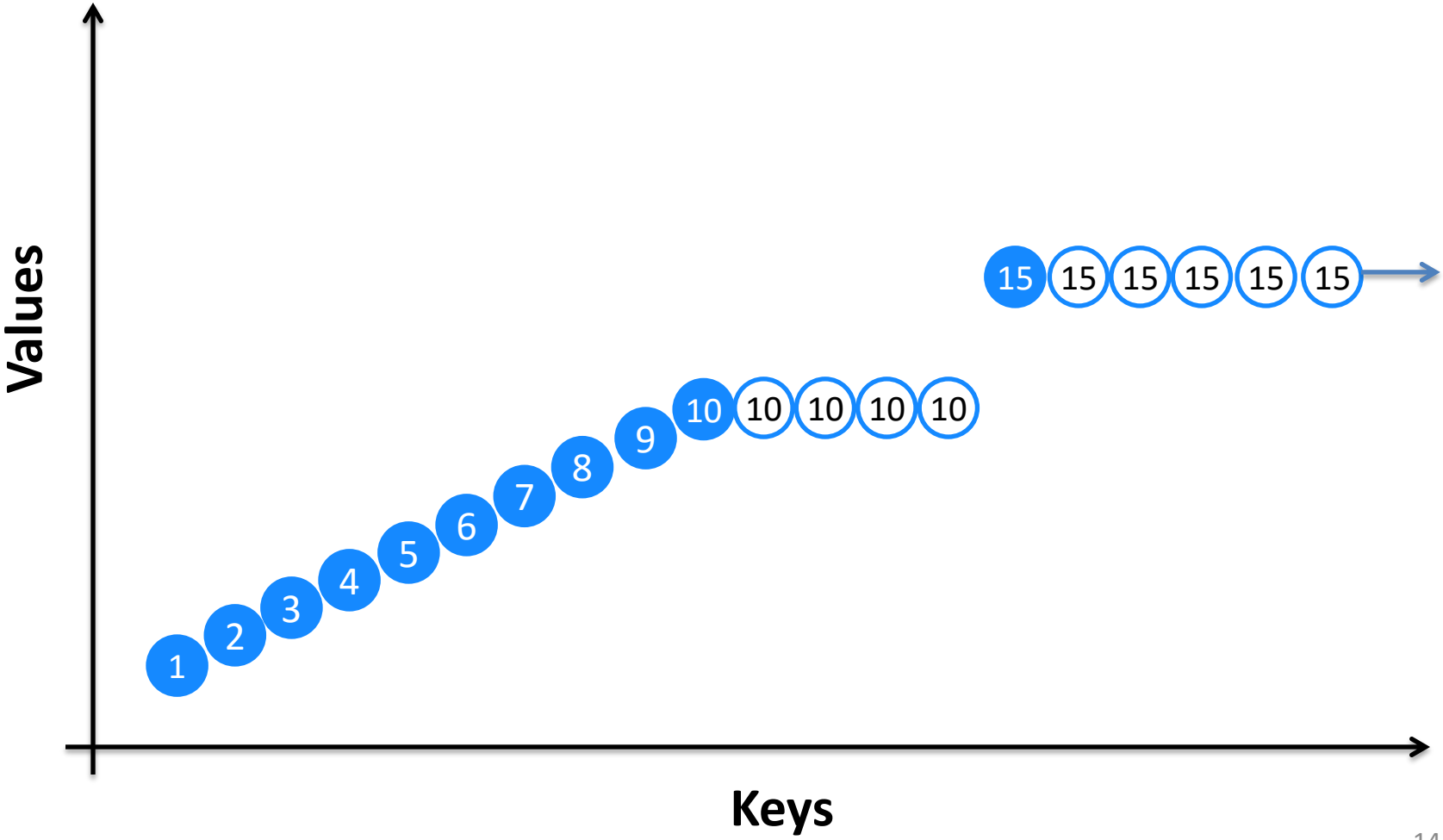
Odd



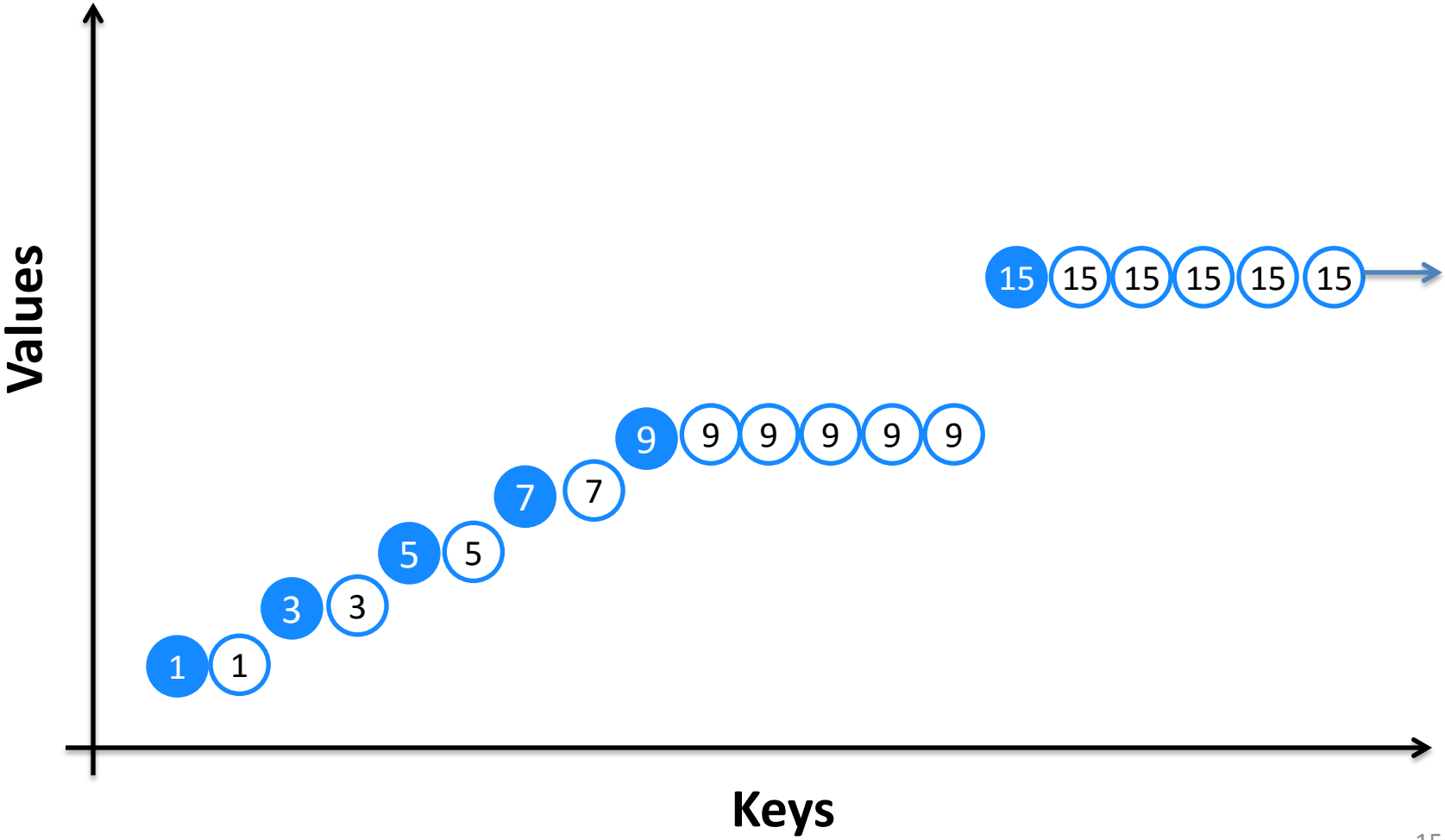
Odd |> Fill


20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
Values
15
9
7
5
3
1
Keys

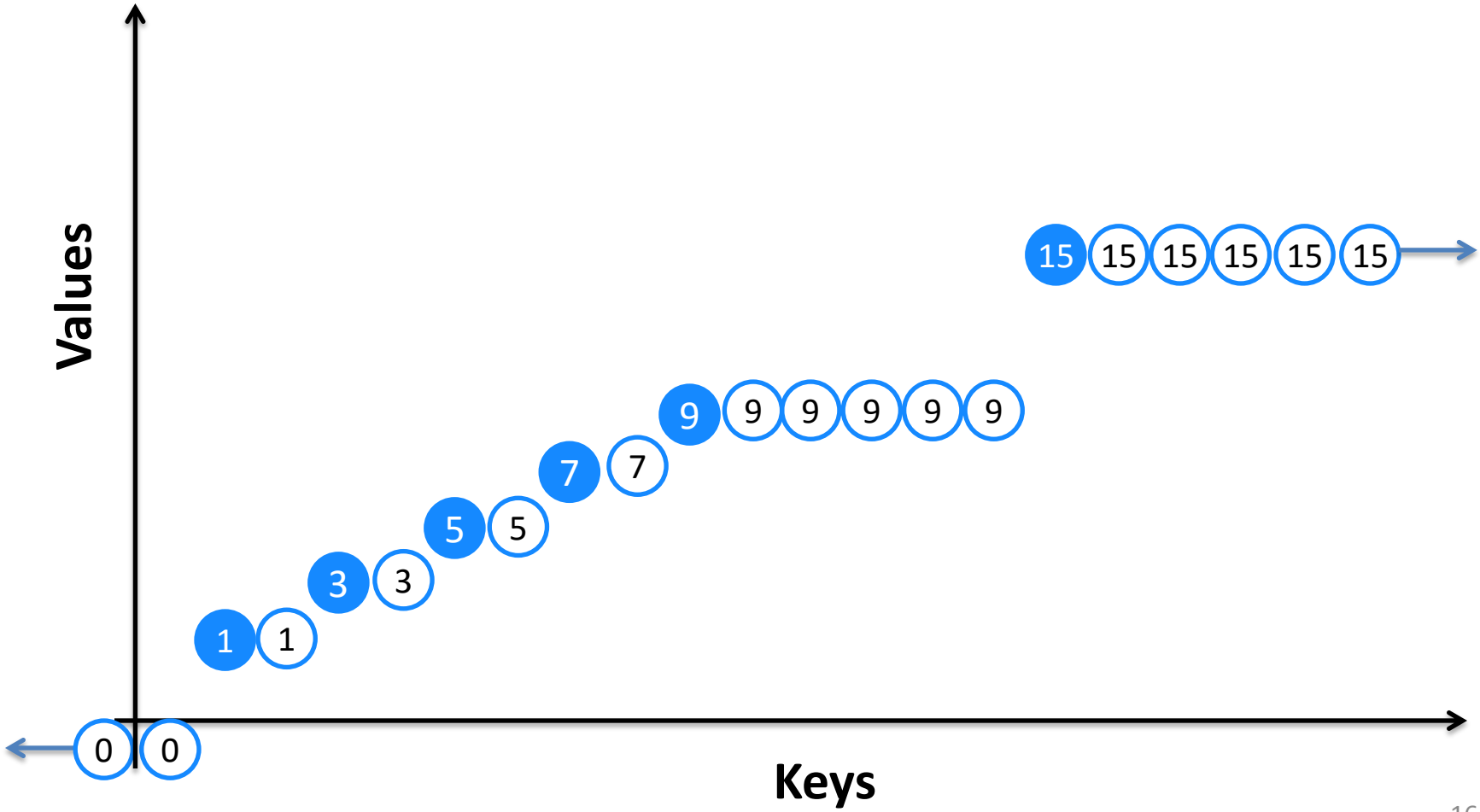
Repeat



Odd |> Repeat



Odd |> Repeat |> Fill -1



Series Containers

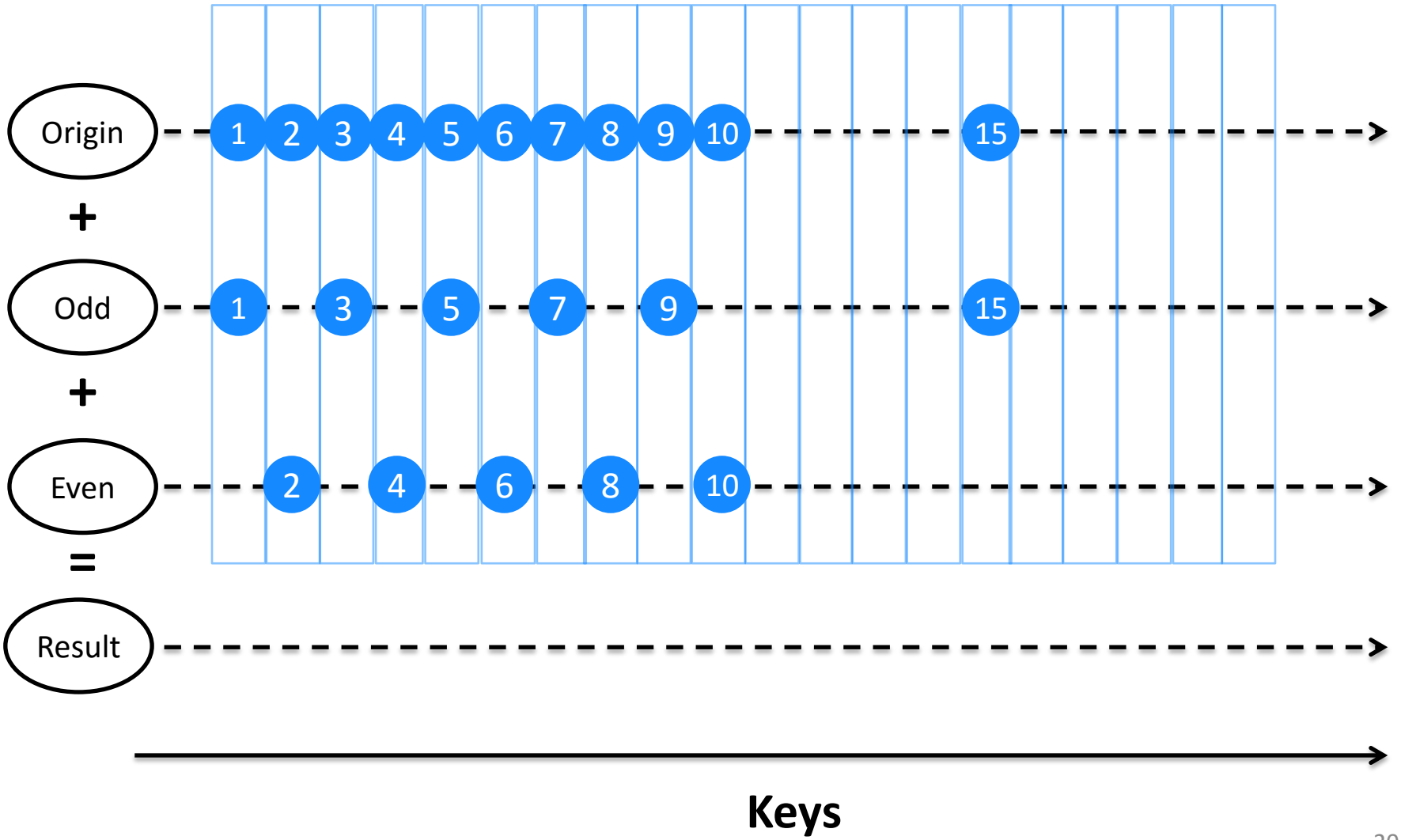
- Containers for series implement:
  - **IReadOnlyOrderedMap** – for read;
  - **IOrderedMap/IImmutableOrderedMap** – for write.
- Mutable and immutable maps:
  - Real-time fast highly optimized imperative containers for series as mutable objects: **SortedMap<K,V>** and **SortedChunkedMapM<K,V>**;
  - Pure-functional slow tree-based persistent maps.

- Declarative series manipulation
- **Zip N**
- Live repository
- Integration with R
- Extensions

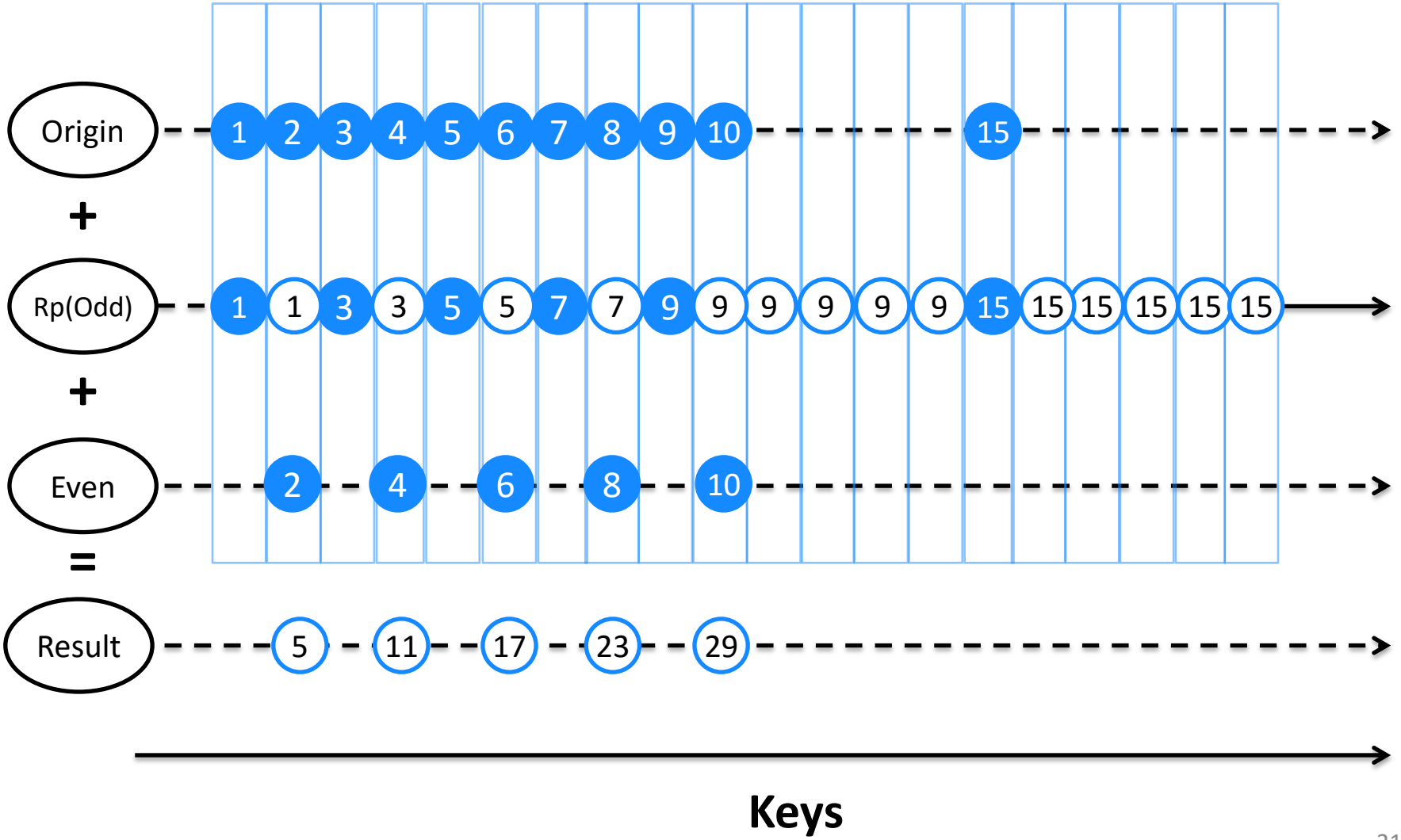
ZipN

- ZipN is **inner join** of N series. Nothing more.
- Instead of complex join rules, we declaratively turn discrete series into continuous series with *.Repeat(), .Fill(), .Interpolate(), .Forecast()*, etc.
- ZipN is a *Series<K,V[]>* with all series properties and behavior.
- All binary operators with series (e.g. +) are implemented as Zip2.

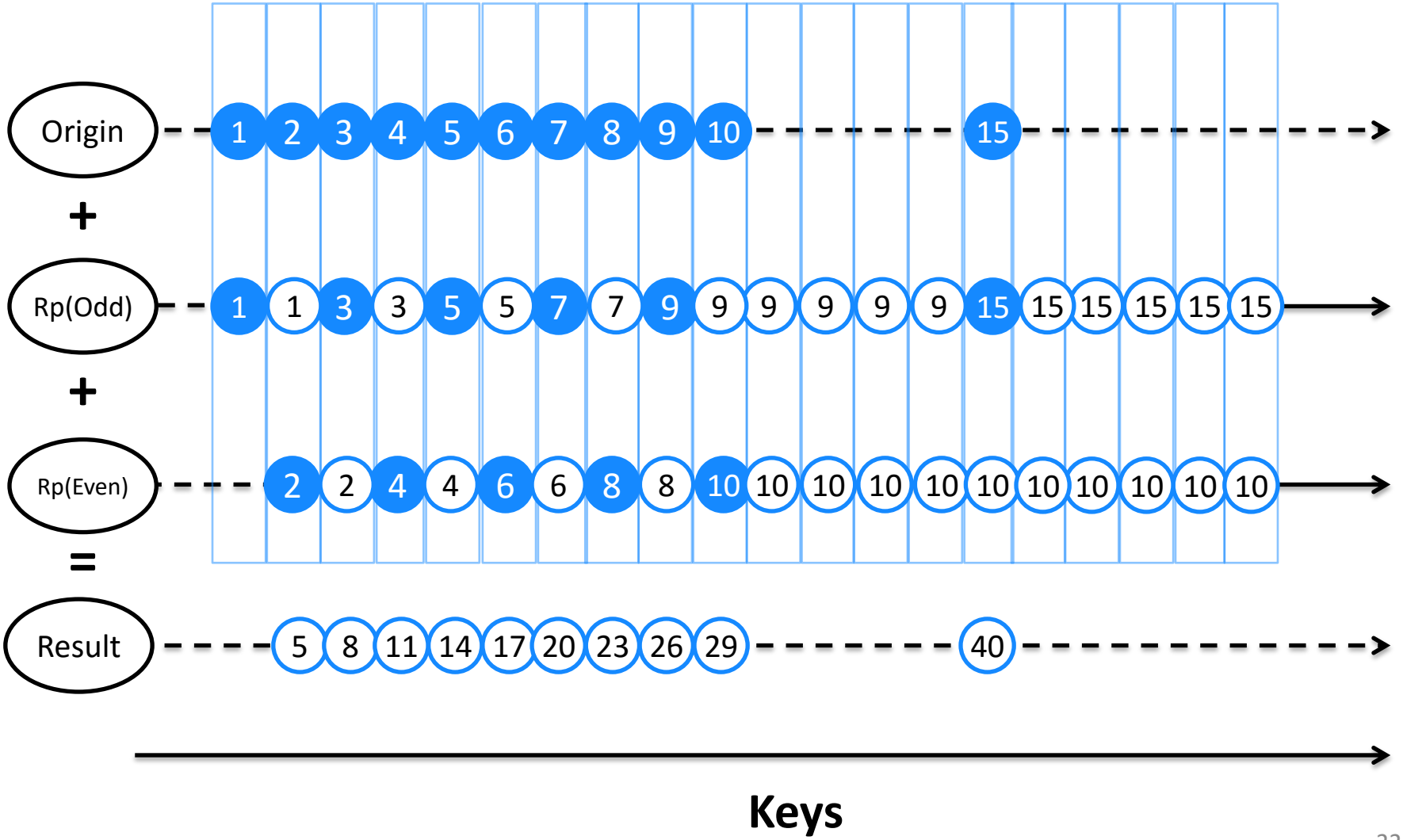
Empty ZipN



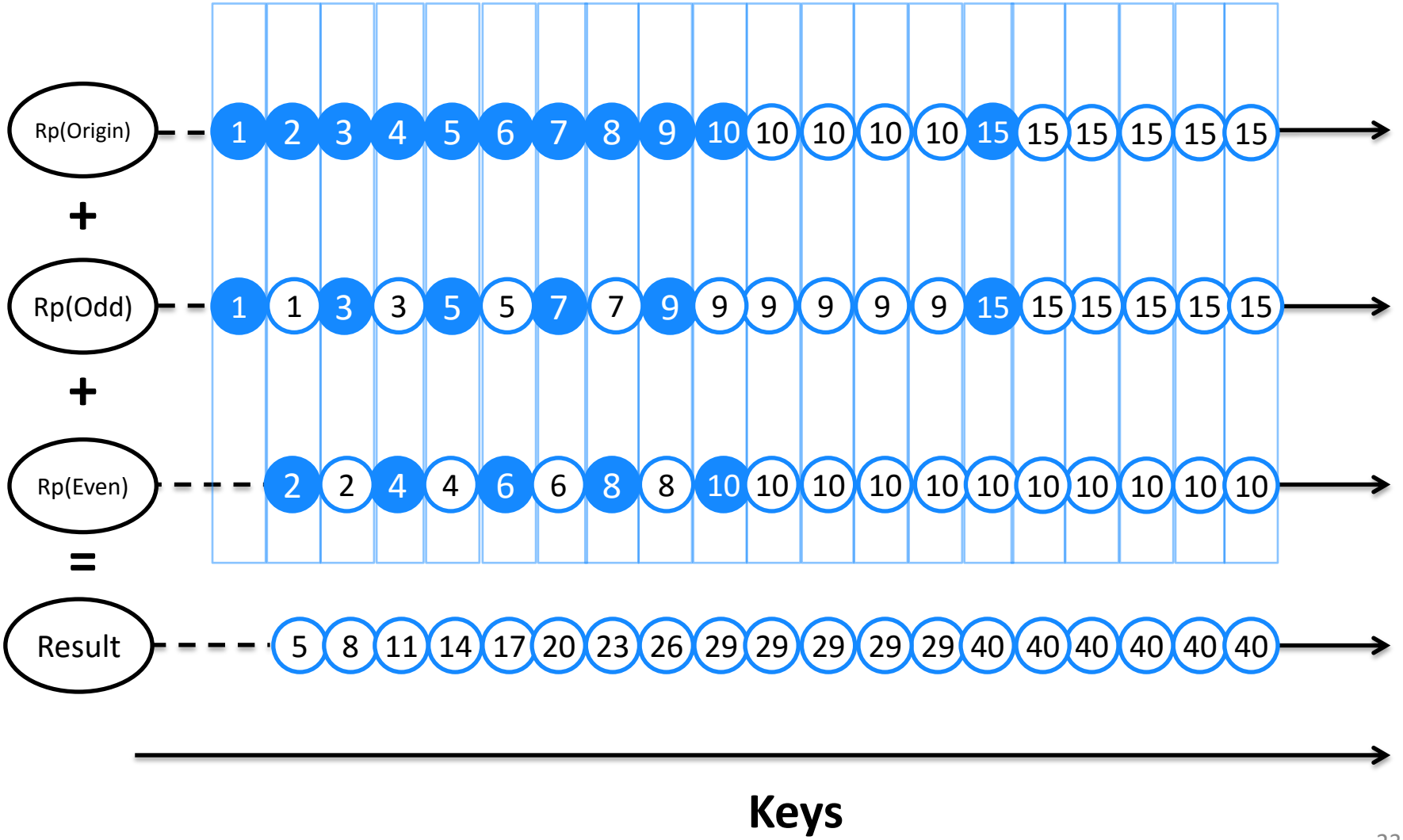
Repeat Odd


Origin
1 2 3 4 5 6 7 8 9 10 15
+
Rp(Odd)
1 1 3 3 5 5 7 7 9 9 9 9 9 15 15 15 15 15 15
+
Even
2 4 6 8 10
=
Result
5 11 17 23 29
Keys

Repeat Odd & Even



Repeat All


Rp(Origin)
1 2 3 4 5 6 7 8 9 10 10 10 10 10 15 15 15 15 15 15
+
Rp(Odd)
1 1 3 3 5 5 7 7 9 9 9 9 9 15 15 15 15 15 15
+
Rp(Even)
2 2 4 4 6 6 8 8 10 10 10 10 10 10 10 10 10 10 10
=
Result
5 8 11 14 17 20 23 26 29 29 29 29 29 40 40 40 40 40 40
Keys

- Declarative series manipulation
- Zip N
- **Live repository**
- Integration with R
- Extensions

Persistent Series

- Series are stored in a named repository on disk via DataRepository object.
- Single-writer principle – a series could be openned for writes only once.
- Ultra-fast inter-process communication via memory-mapped files (based on Aeron LogBuffers scheme).
- Single data space with real-time updates for all processes on a machine.

Persistent Map

- Concurrent ultra-fast lock-free memory-mapped implementation of *IDictionary<Tkey,Tvalue>* interface.
- 99%-ile latency is below 1 microsecond, median latency is less than timer resolution.
- Multiple concurrent writers and readers.
- Crash recovery: all writes are atomic, if a writer dies during an update, next read/write operation will recover data state.

Broadcast Observable

- Ultra-fast messaging and events broadcast.
- *BroadcastObservable* object implements *IObservable* interface for receiving and *IObserver* interfaces for sending messages.
- Could be opened by name on any number of clients (instances of DataRepository) in any process.
- Sender does not receive its own message.

- Declarative series manipulation
- Zip N
- Live repository
- **Integration with R**
- Extensions

Integration with R

- R package for exporting R functionality instead of programming R from .NET.
- Optimized data transfer: direct evaluation of lazy series and panels into R vector without intermediate buffers.
- Access to series in local repositories.

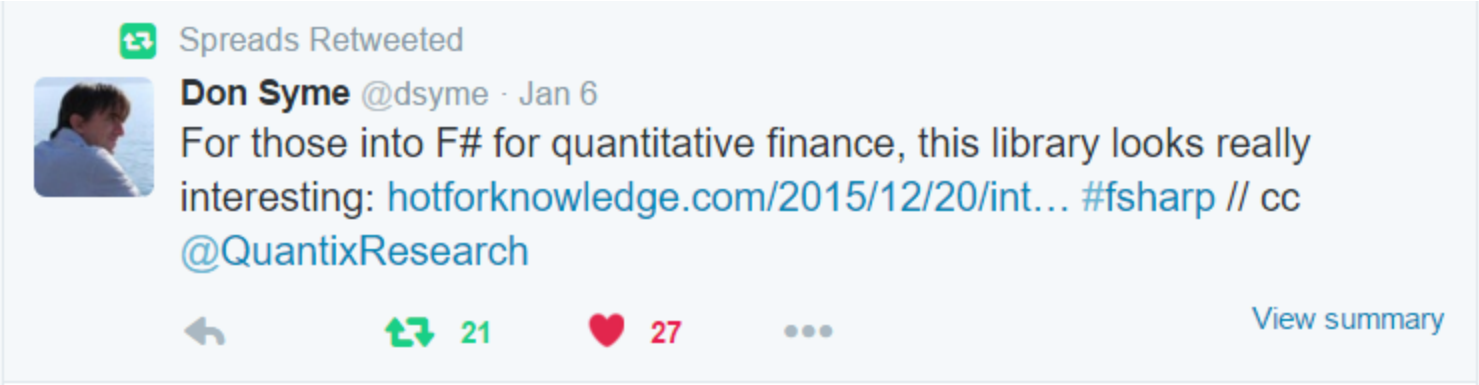
- Declarative series manipulation
- Zip N
- Live repository
- Integration with R
- **Extensions**

Extensions

- Very efficient binary serializer and compressor of arrays and series based on Blosc.
- Interactive Extensions adapted for Spreads;
- Yeppp math library for SIMD calculations.
- Generic array pool that gives very visible performance gain on some benchmarks.
- Blittable data types optimized for storage and compression, such as Price, Tick, Candle.
- Concrete optimized implementations of series calculations, such as SMA, StDev.
- Useful utils for historical time zones conversions using NodaTime.

Data Spreads!

DataSpreads.com



https://github.com/Spreads

http://hotforknowledge.com/tag/spreads/

@DataSpreads

https://github.com/buybackoff

@buybackoff

vb@fi.im