Stencil

A Declarative System
for Visualizing Dynamic Data

Joseph A. Cottam

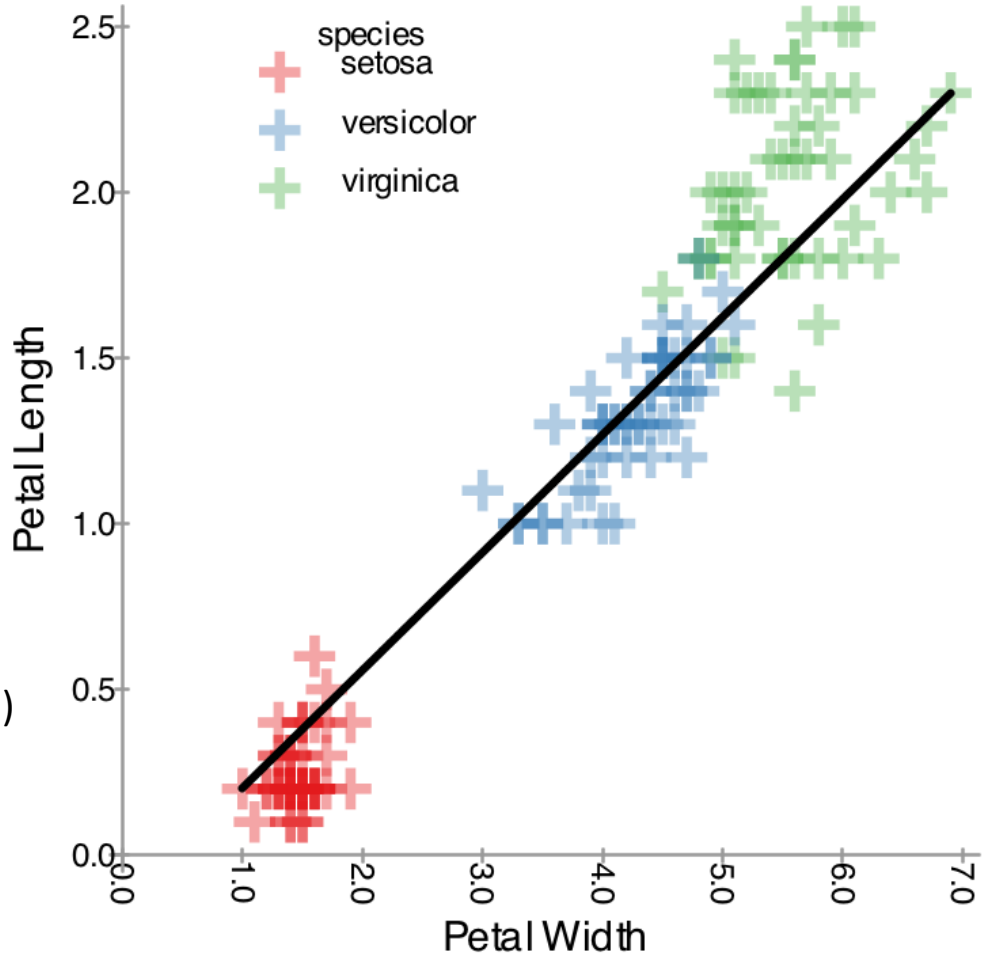
CREST – Indiana University

Objective

Construct a system for working with dynamic data. In that systems, provide abstractions for **analysis**, abstractions for **representation** and an **efficient runtime**. Make this system accessible for integration.

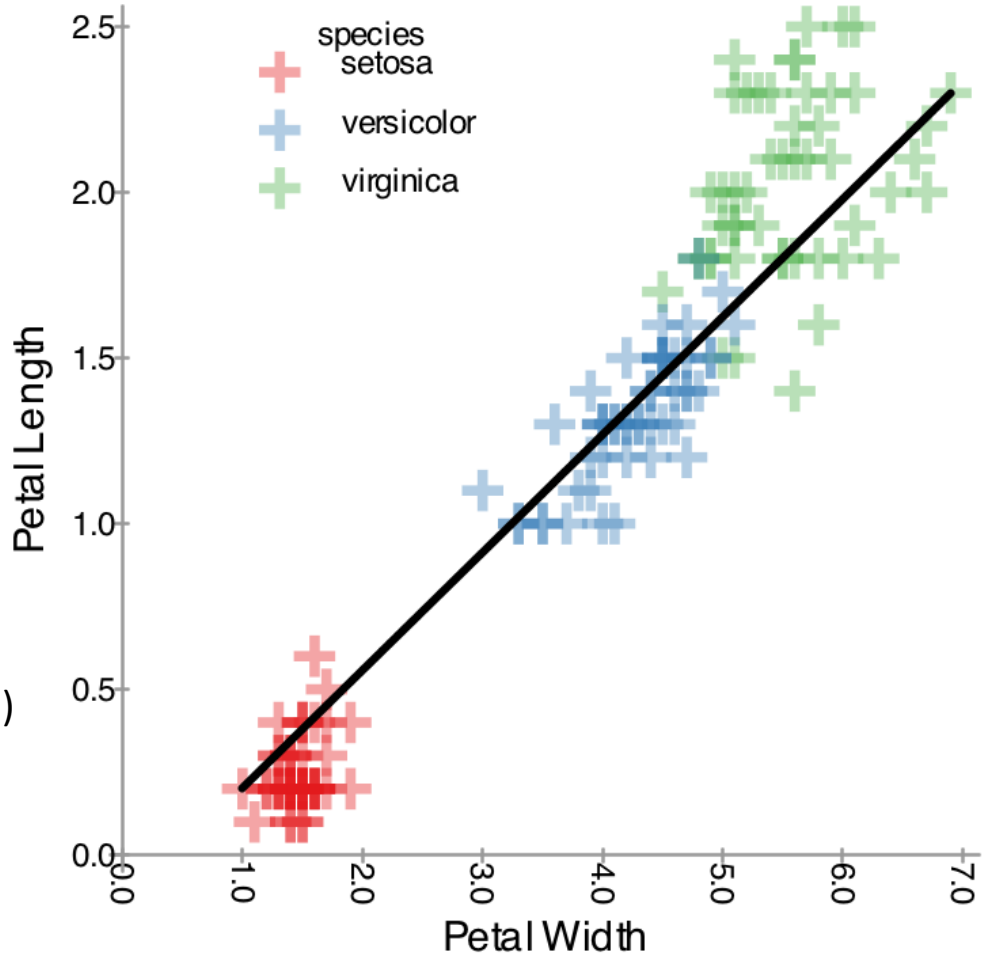
Reading a Stencil Program

```
stream flowers(sepalL, petalW, sepalW,
                        petalL, species, obs)

layer FlowerPlot
guide
    legend[X: 0, Y: 100] from FILL_COLOR
    axis[guideLabel: "Petal Length"] from Y
    axis[guideLabel: "Petal Width"] from X
from flowers
    ID: obs
    X:* Scale[0,100](petalW)
    Y:* Scale[0,100](petalL)
    FILL_COLOR: BrewerColors(species)
                    -> SetAlpha(50,BrewerColors)

    REGISTRATION: "CENTER"
    SHAPE : "CROSS"
```



Dynamic Binding

```
stream flowers(sepalL, petalW, sepalW,
                        petalL, species, obs)

layer FlowerPlot
guide
    legend[X: 0, Y: 100] from FILL_COLOR
    axis[guideLabel: "Petal Length"] from Y
    axis[guideLabel: "Petal Width"] from X
from flowers
    ID: obs
    X:* Scale[0,100](petalW)
    Y:* Scale[0,100](petalL)
    FILL_COLOR: BrewerColors(species)
                    -> SetAlpha(50,BrewerColors)

    REGISTRATION: "CENTER"
    SHAPE : "CROSS"
```

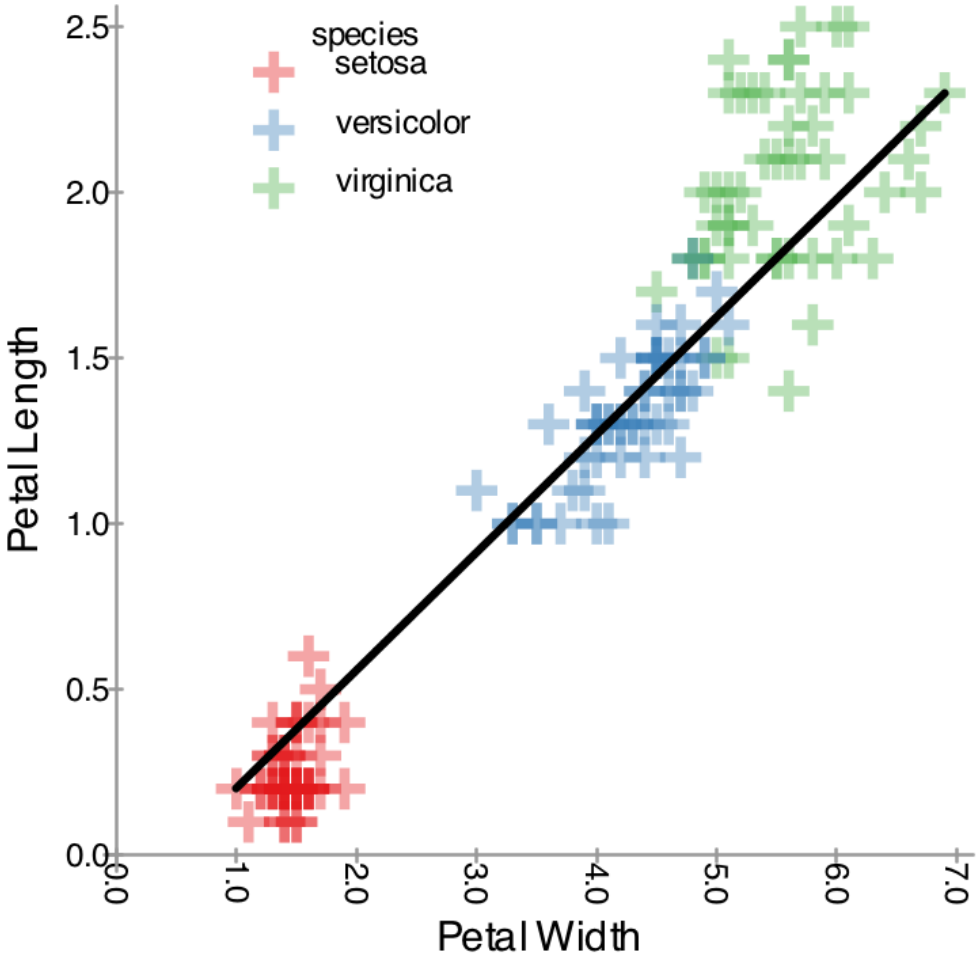


Dynamic Binding

```
layer FlowerPlot
from flowers
  X:* Scale[0,100](petalW)
```



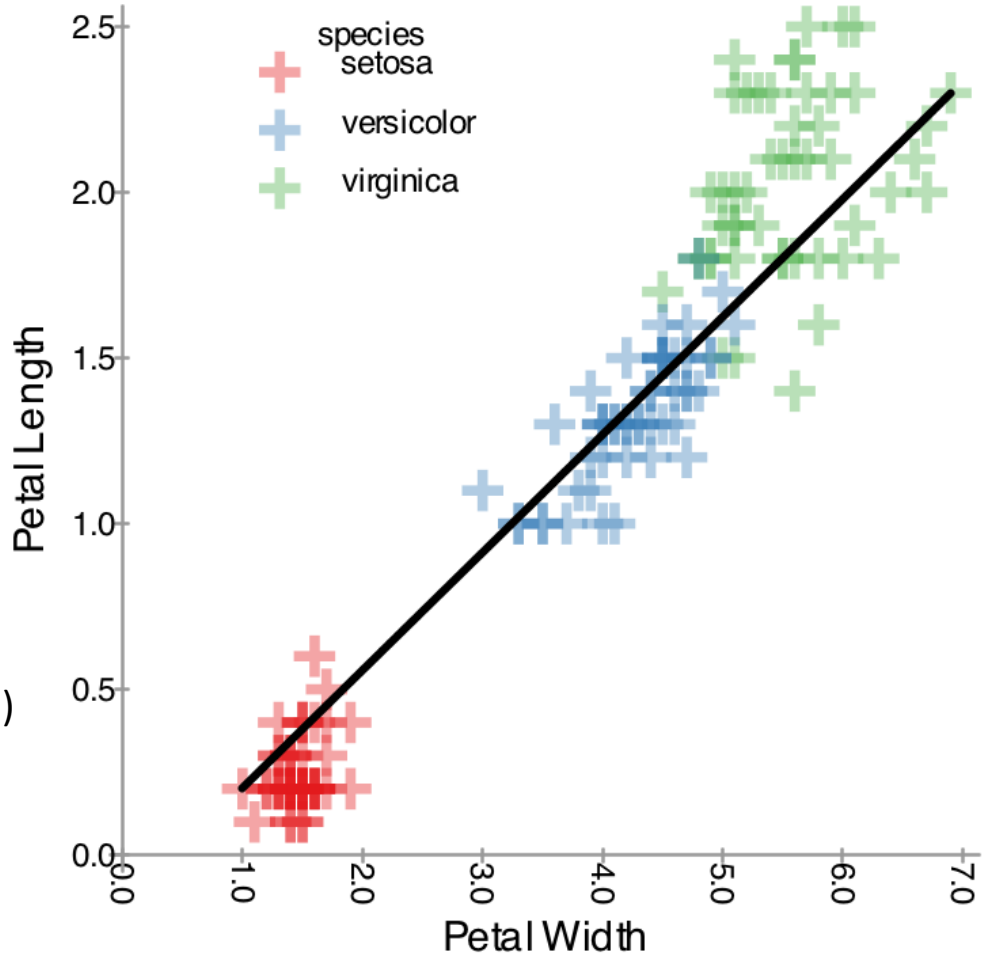
```
layer FlowerPlot
from flowers
  X: Scale.map(petalW)
  #data: petalL
from #Render
  X: Scale.query(#data.0)
```

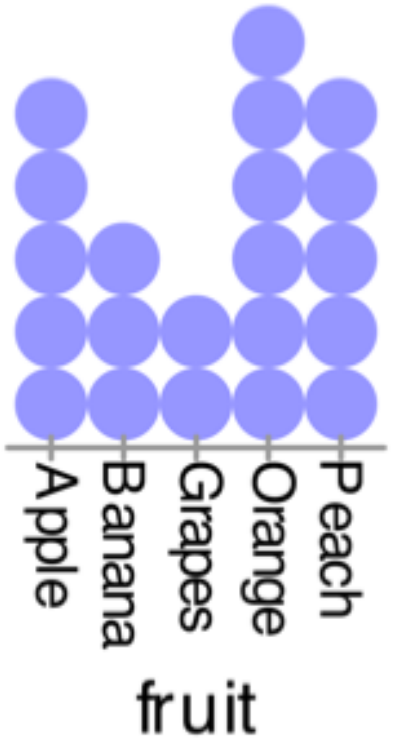


#Render is the stream that controls rendering

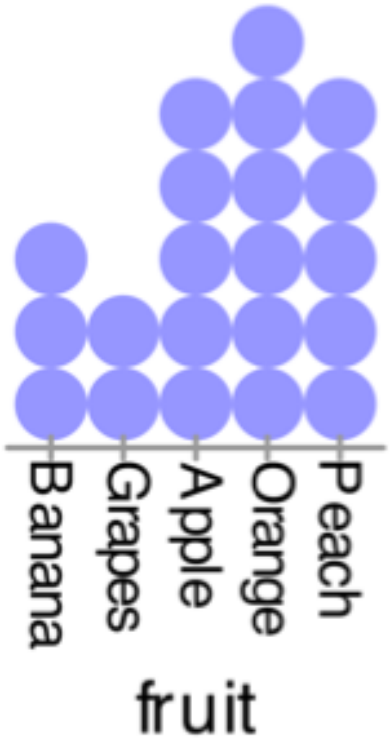
Guide Creation

```
stream flowers(sepalL, petalW, sepalW,
                        petalL, species, obs)

layer FlowerPlot
guide
    legend[X: 0, Y: 100] from FILL_COLOR
    axis[guideLabel: "Petal Length"] from Y
    axis[guideLabel: "Petal Width"] from X
from flowers
    ID: obs
    X:* Scale[0,100](petalW)
    Y:* Scale[0,100](petalL)
    FILL_COLOR: BrewerColors(species)
                    -> SetAlpha(50,BrewerColors)

    REGISTRATION: "CENTER"
    SHAPE : "CROSS"
```

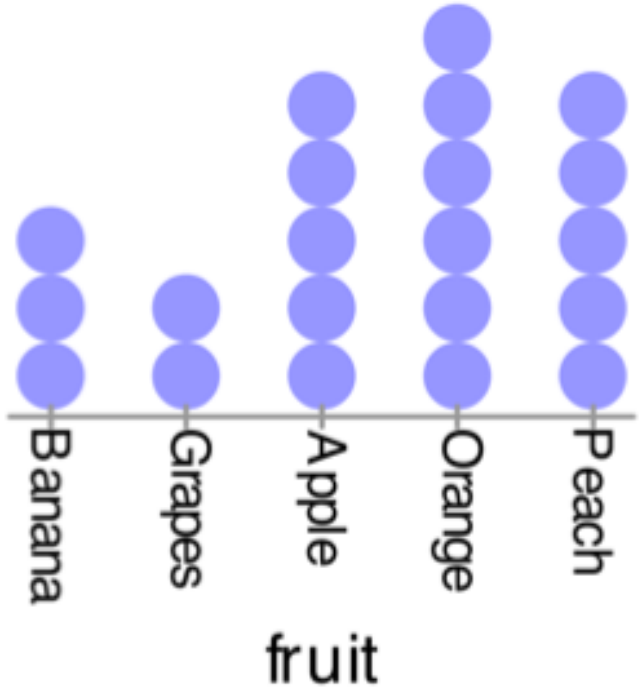




(a) Rank



(b) Index



(c) Wider Index

```
1  stream survey(fruit)
2
3  layer plot
4  guide
5    axis from X
6  from survey
7      ID: Count()
8      X:* Rank(fruit) -> Mult(5, Rank)
9  /* X:* Index(fruit) -> Mult(5, index)*/
10 /* X:* Index(fruit) -> Mult(10, index)*/
11     Y: Count(fruit) -> Mult(5, Count)
12     REGISTRATION: "CENTER"
13     FILL_COLOR: Color{150,150,255}
```

Counterpart

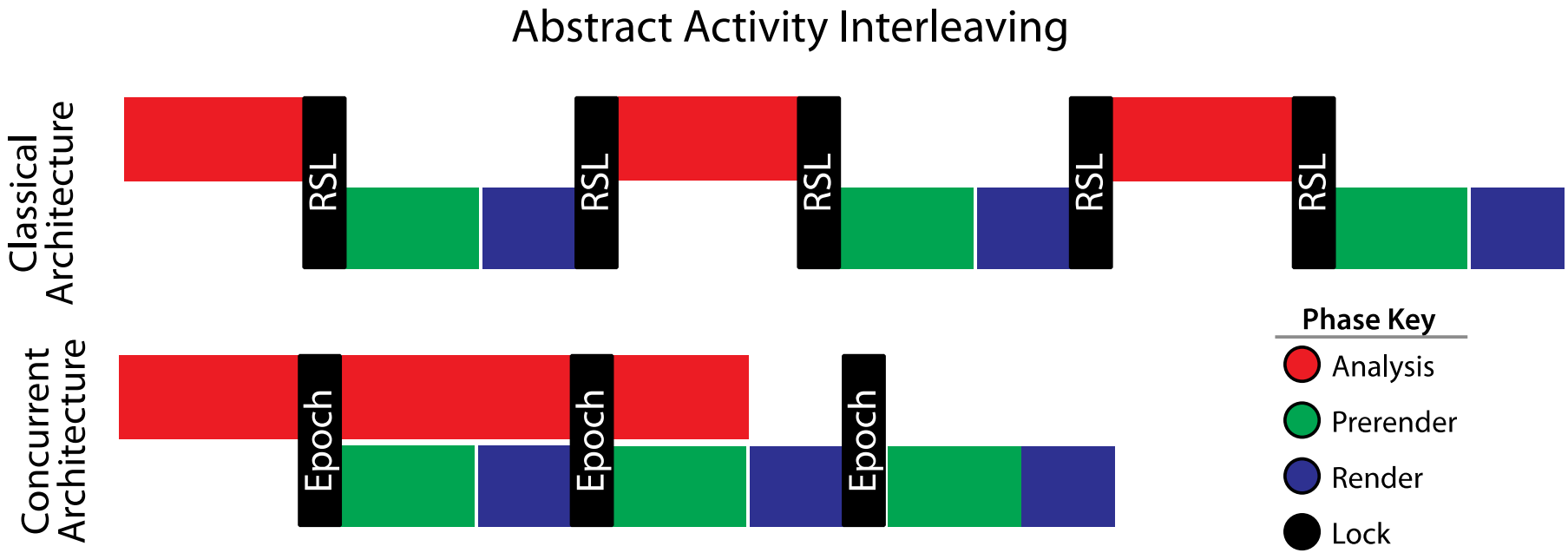
Scale between 0 and 1

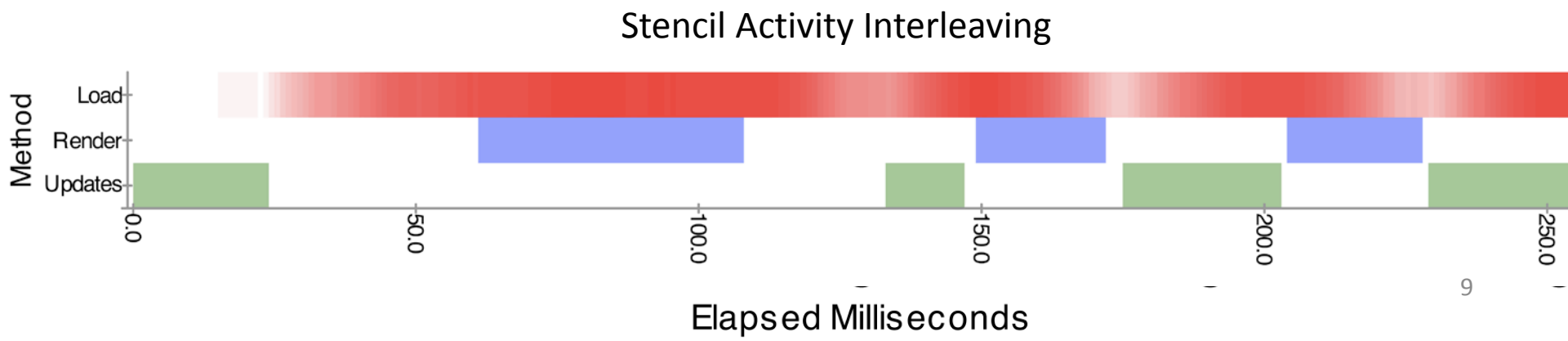
Scale.map(0)	→ 0
Scale.map(10)	→ 1
Scale.map(9)	→ .9
Scale.map(10)	→ 1
Scale.query(.8)	→ .8
Scale.map(20)	→ 1
Scale.query(10)	→ .5
Scale.map(10)	→ .5
Scale.query(100)	→ -1
Scale.query(-1)	→ -1

Count occurrences

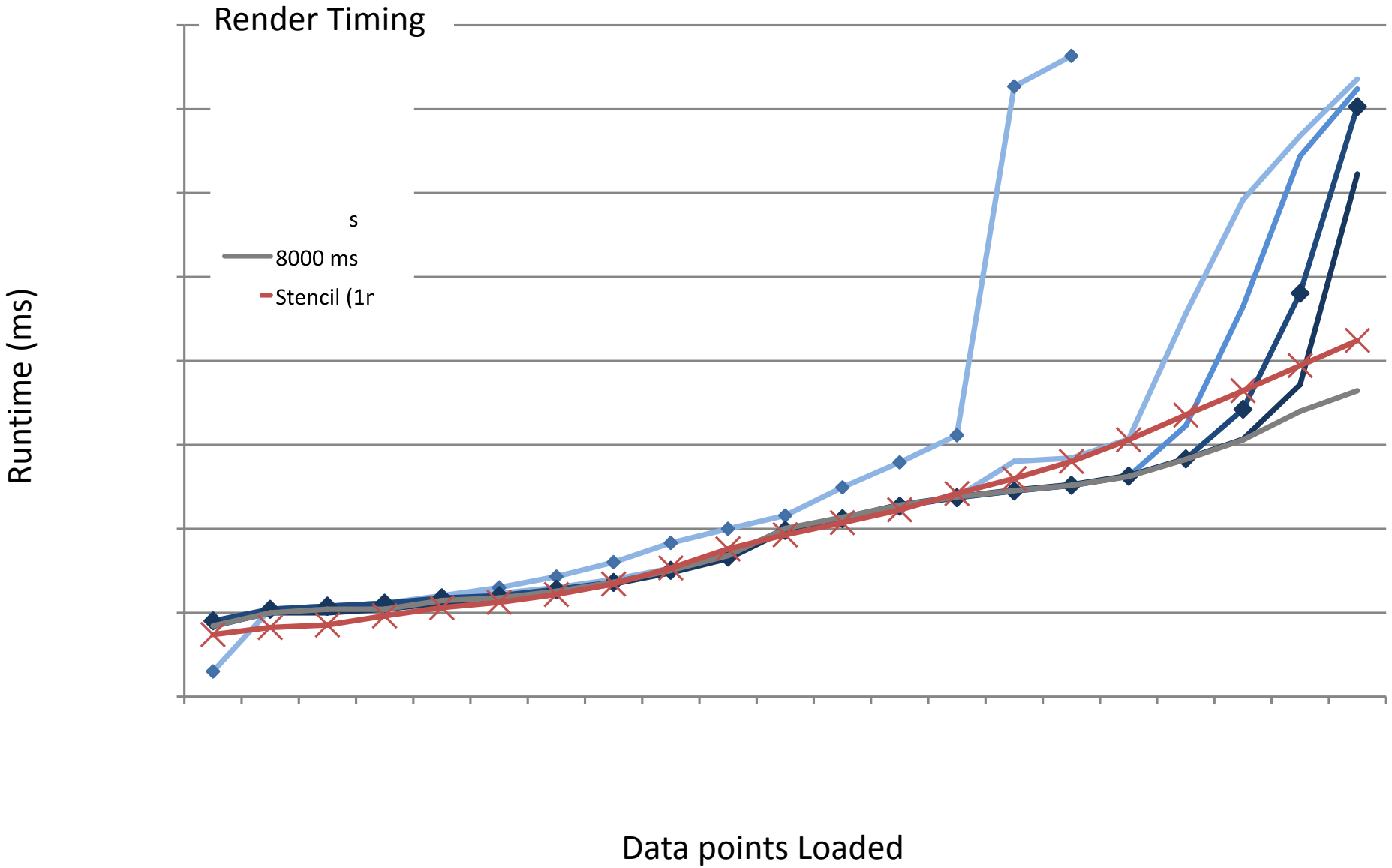
Count.map("page")	→ 1
Count.map("page")	→ 2
Count.map("page")	→ 3
Count.query("page")	→ 3
Count.query("page")	→ 3
Count.map("pic")	→ 1
Count.query("pic")	→ 1
Count.map("pic")	→ 2
Count.query("pic")	→ 2
Count.query("grape")	→ -1

Task Parallelism Interleaving

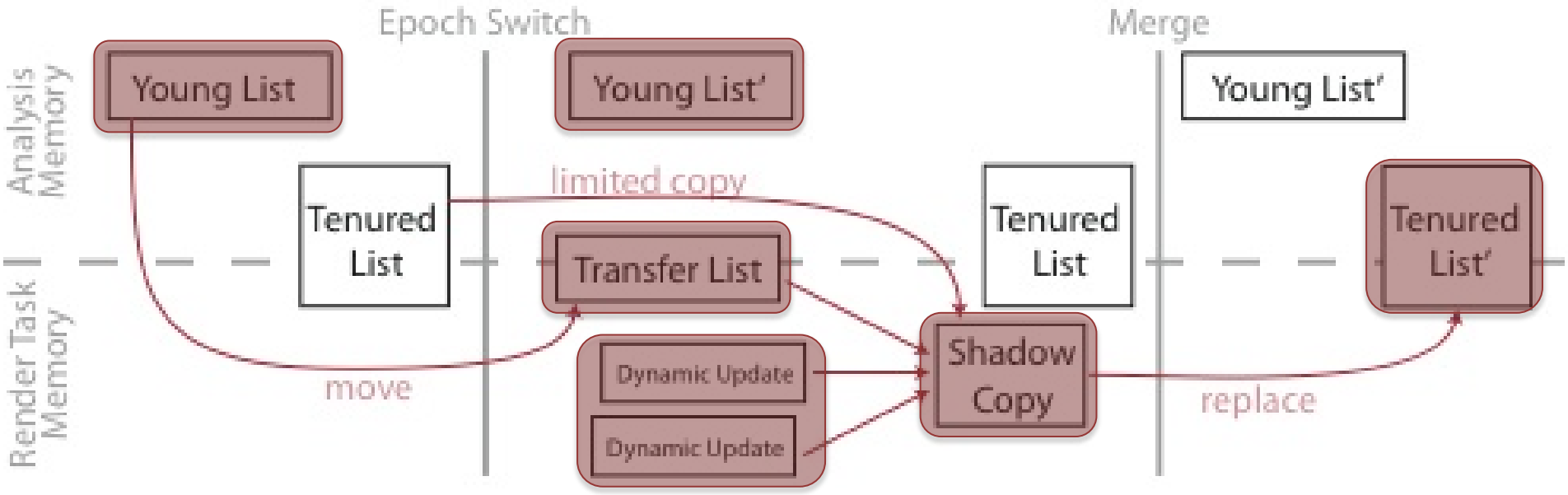




Task Runtime



Persistent Architecture Workflow



Current Projects

- Simplified Runtime
  - Multiple runtimes
- In-situ visualization
  - Directly Shared data
  - Code generation (reducing abstraction costs)
- Simplify custom operator integration