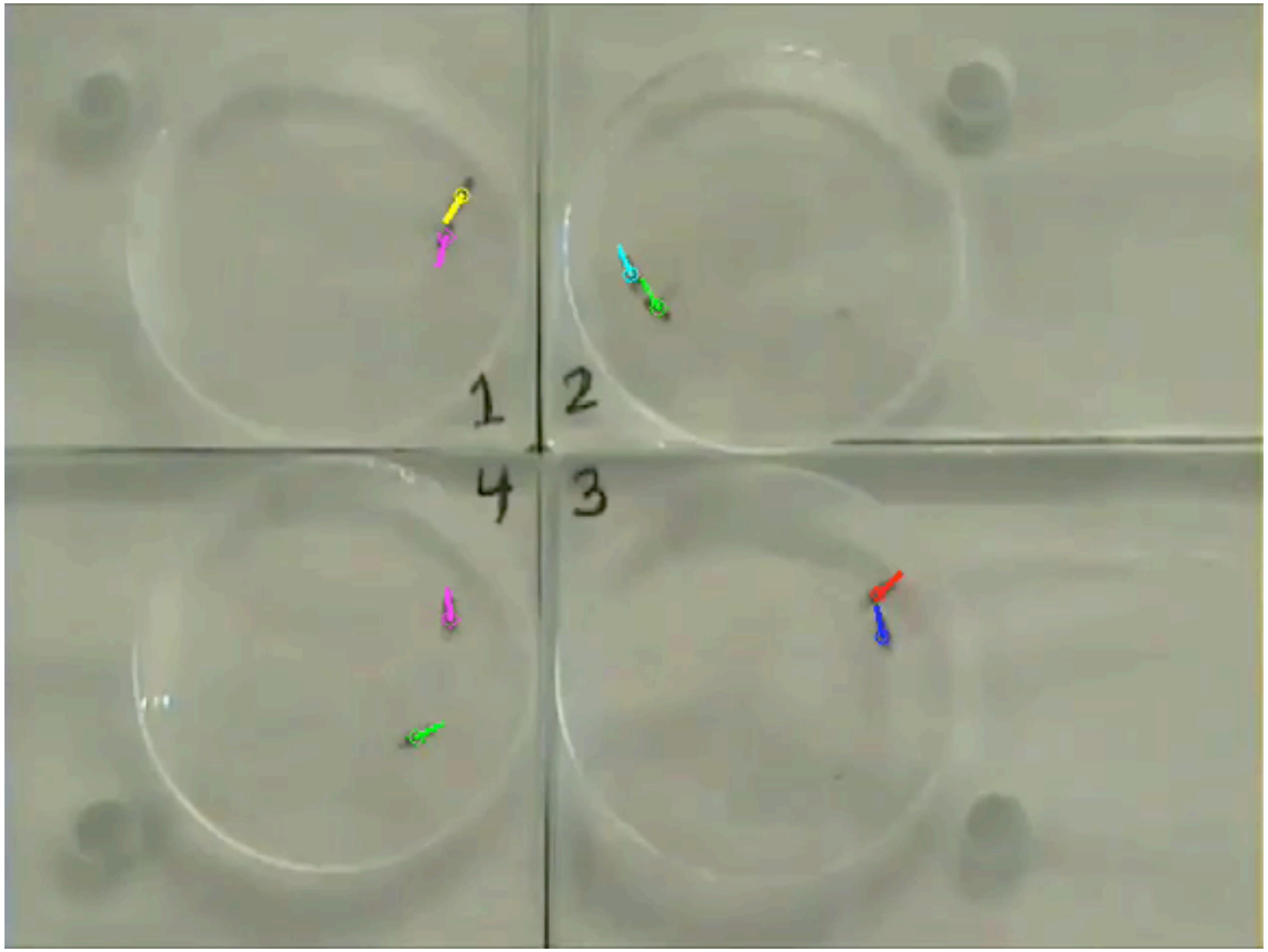
# Cloudy Vision

## Demystifying On-Demand Scalable Image Processing

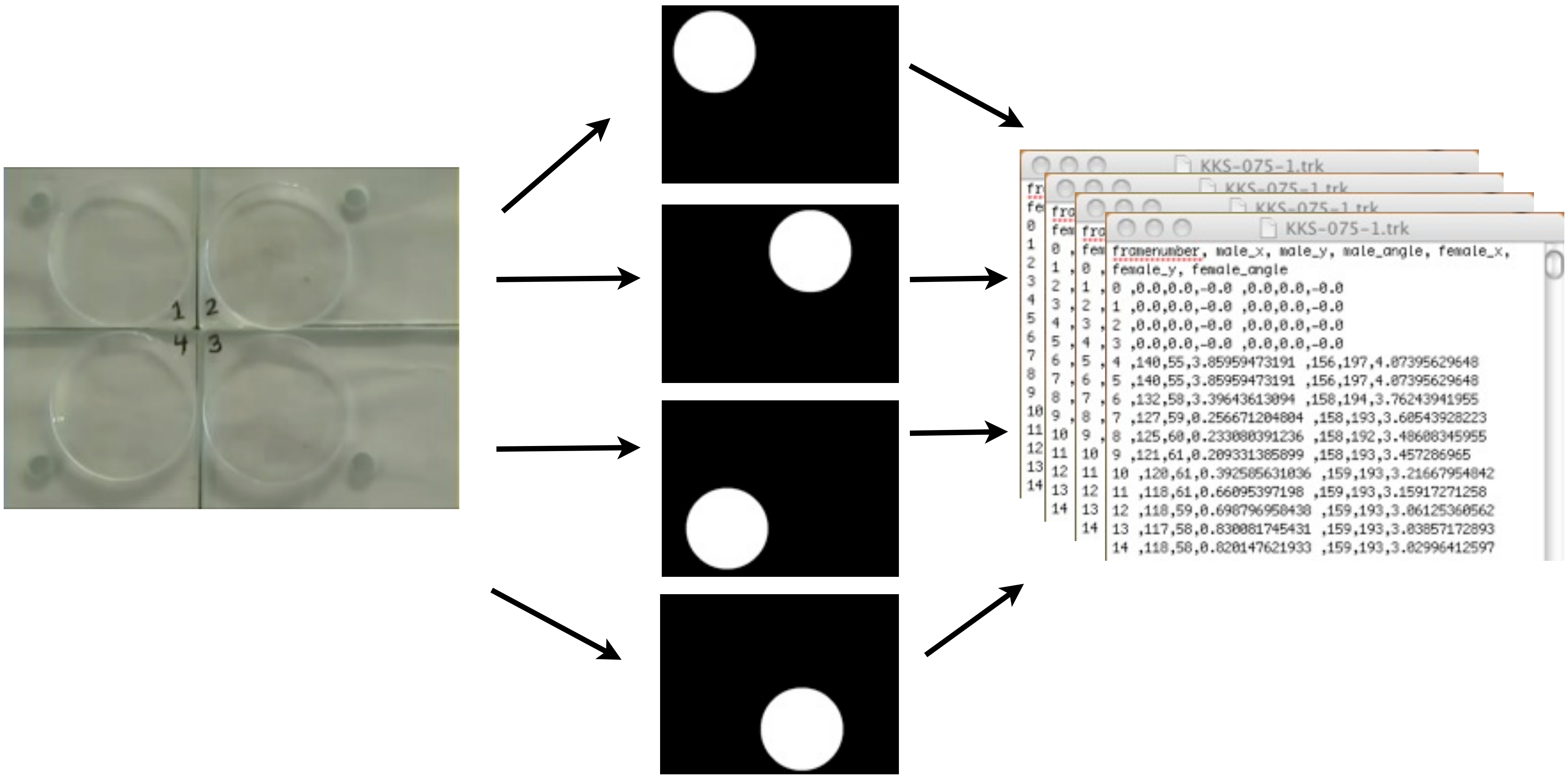
Tim Lukins
Institute for Adaptive and Neural Computation
School of Informatics, University of Edinburgh

# A problem for us...




ibehave®

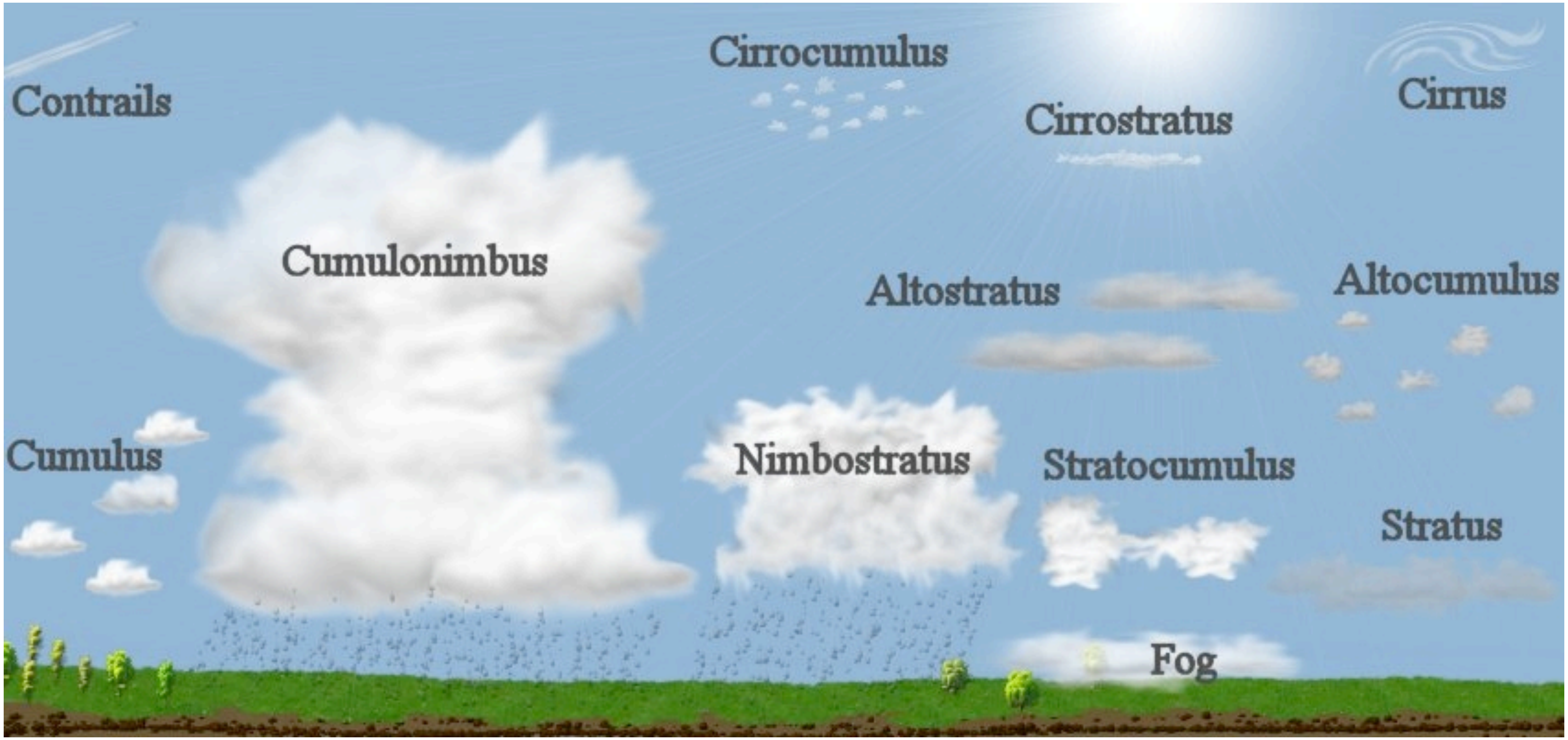
# Smarter = Faster



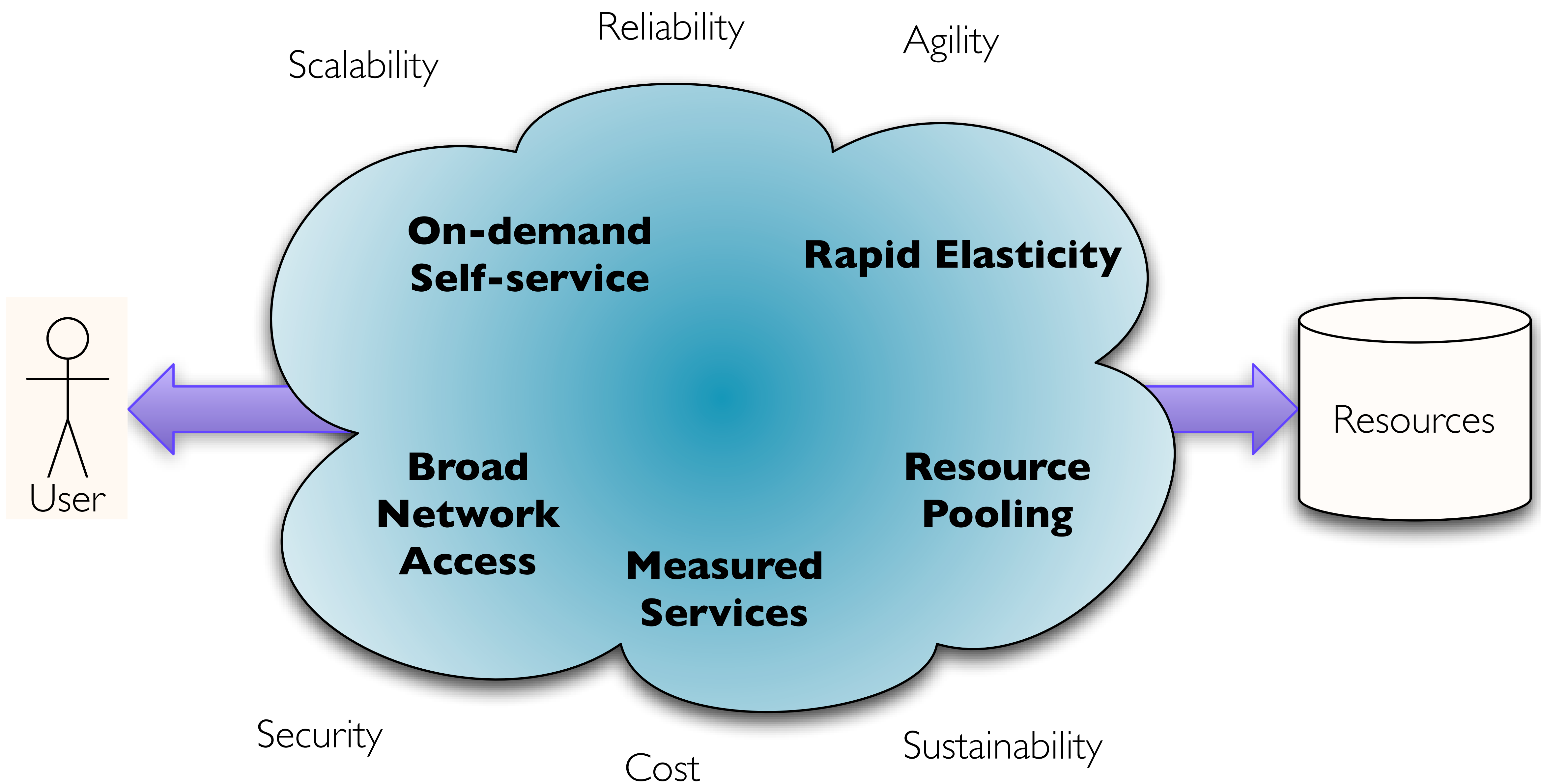

- If 4 trials x 20 videos x 1 hour each at 15fps = 4,320,000 images to process...

# Hence this talk...

- The reality of cloud computing for solving complex, scientific, image-based problems.



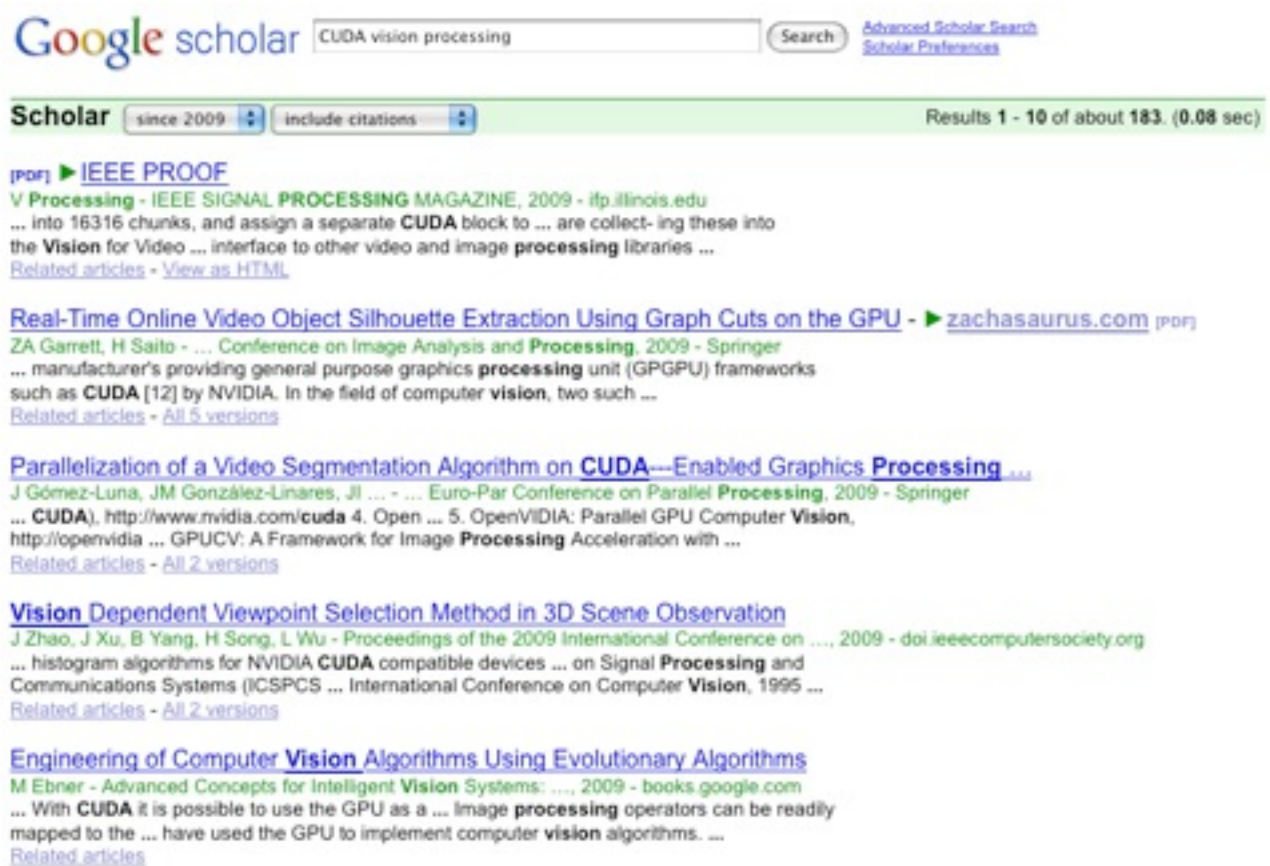
# The Cloud




[NIST Working Definition of Cloud Computing (draft)]

# Déjà vu?

- Real time parallelism: CUDA, OpenCL, Erlang, Occam, etc.
- Distributed processing: HPC, Beowulf clustering, The Grid...




[Bob Jones, Comparitive Study: Grids and Clouds, Evolution or Revolution?, CERN EGEE Technical Report, 2008]

# What's so different?

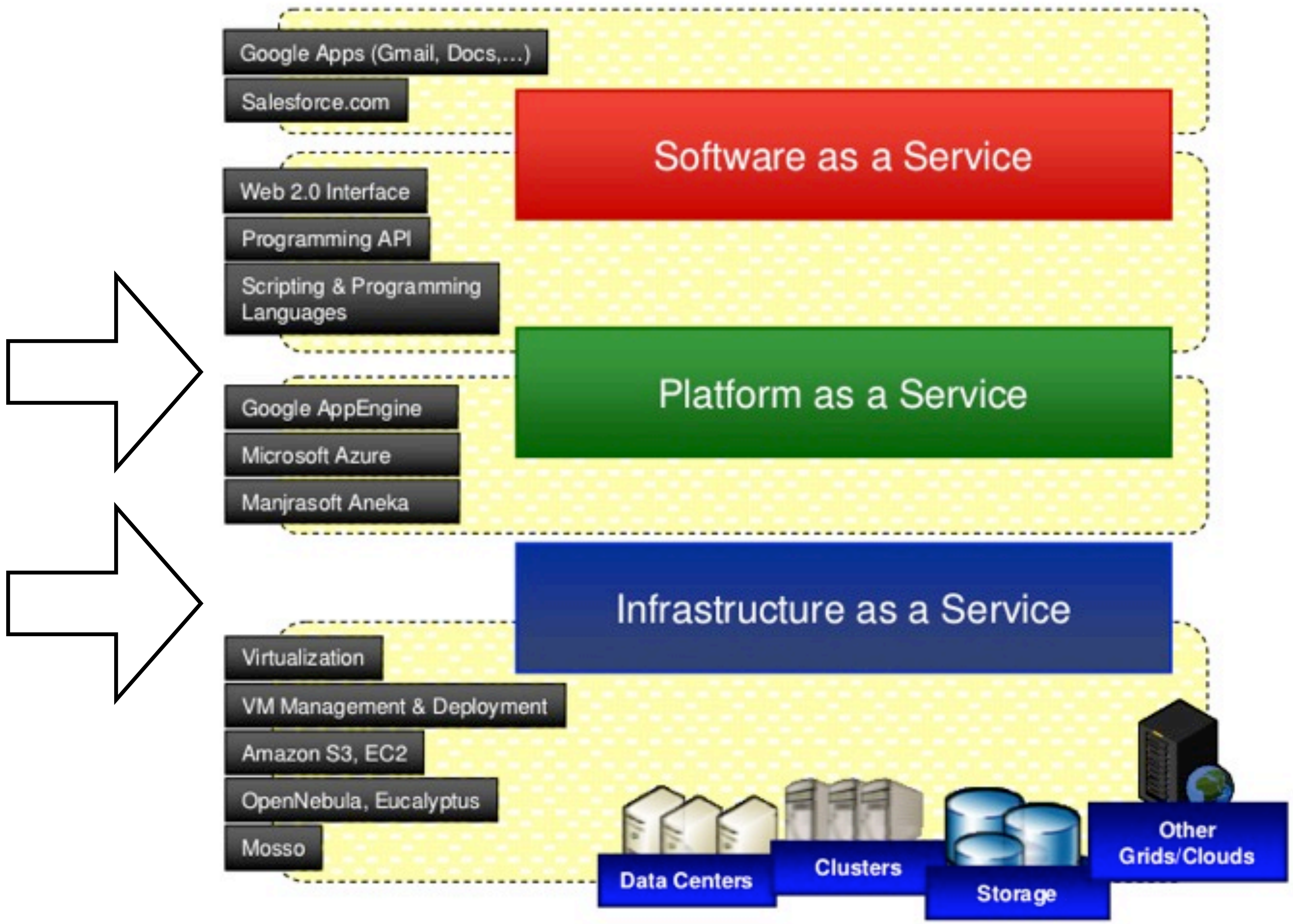
Big Data - Simple Processing

Very Scalable - Robust Distributed File System

New model - Dynamic Provisioning, Easy Access

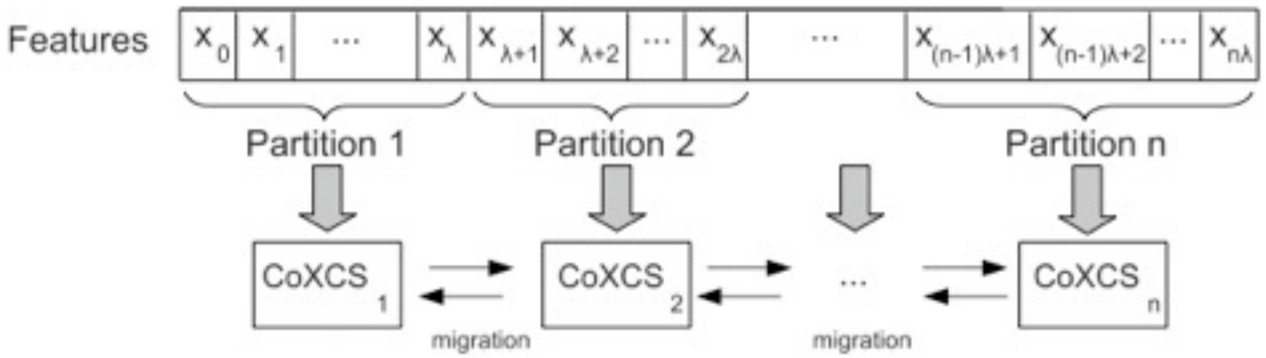
# The Reality

Insert Science Here

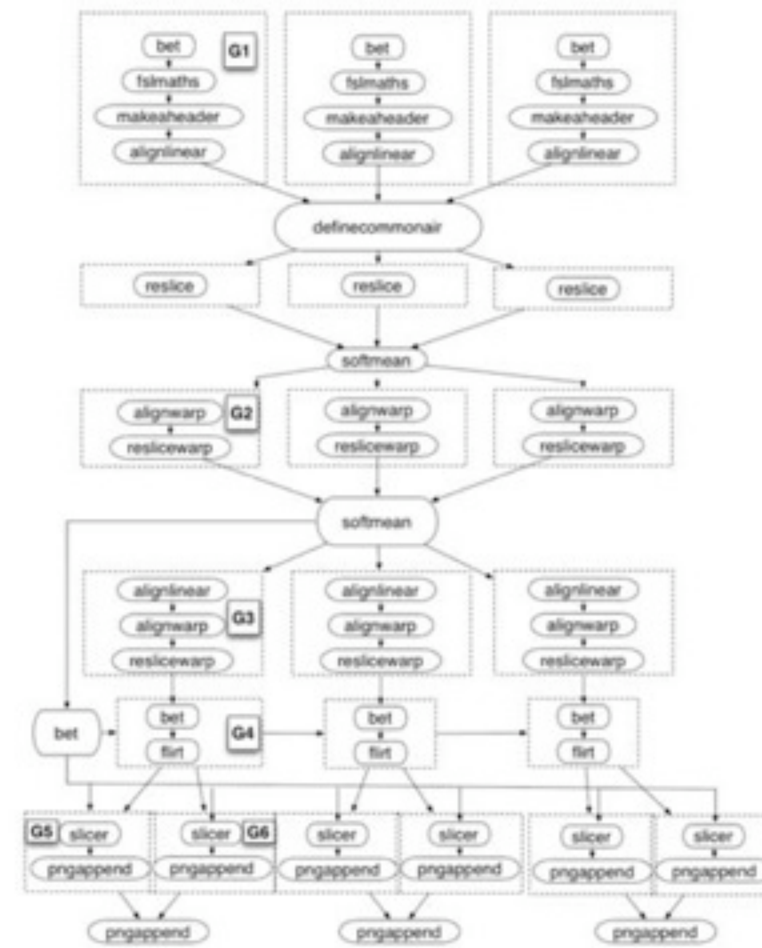



[Christian Vecchiola, Suraj Pandey, Rajkumar Buyya, High-Performance Cloud Computing: A View of Scientific Applications, Keynote, I-SPAN 2009]

# To do what?




| Classifier | Mode | BRCA | Prostate |
|---|---|---|---|
| J48 | Train | 0.92 ± 0.06 | 1.00 |
| | Test | 0.35 ± 0.01 | 0.60 ± 0.10 |
| NBTree | Train | 1.00 | 1.00 |
| | Test | 0.65 ± 0.12 | 0.46 ± 0.04 |
| Random Forest | Train | 1.00 | 1.00 |
| | Test | 0.51 ± 0.01 | 0.60 ± 0.09 |
| Logistic Regression | Train | 1.00 | 0.50 |
| | Test | 0.85 ± 0.17 | 0.50 |
| Naïve Bayes Classifier | Train | 0.99 ± 0.01 | 1.00 |
| | Test | 0.90 ± 0.05 | 0.35 ± 0.04 |
| SVM | Train | 1.00 | 1.00 |
| | Test | 0.53 ± 0.04 | 0.51 ± 0.07 |
| XCS | Train | 0.50 | 0.50 |
| | Test | 0.50 | 0.50 |
| CoXCS | Train | 1.00 | 1.00 |
| | Test | **0.98 ± 0.02** | **0.70 ± 0.02** |




Cloud CoXCS for classifying gene expression data.

fMRI image registration workflows.

[M. Abedini and M. Kirley, "CoXCS: A Coevolutionary Learning Classifier Based on Feature Space Partitioning," Proc. The 22nd Australasian Joint Conference on Artificial Intelligence (AI'09), Melbourne, Australia, December 1-4, 2009. ]

[Second IEEE International Scalable Computing Challenge (SCALE 2009)]

# MapReduce

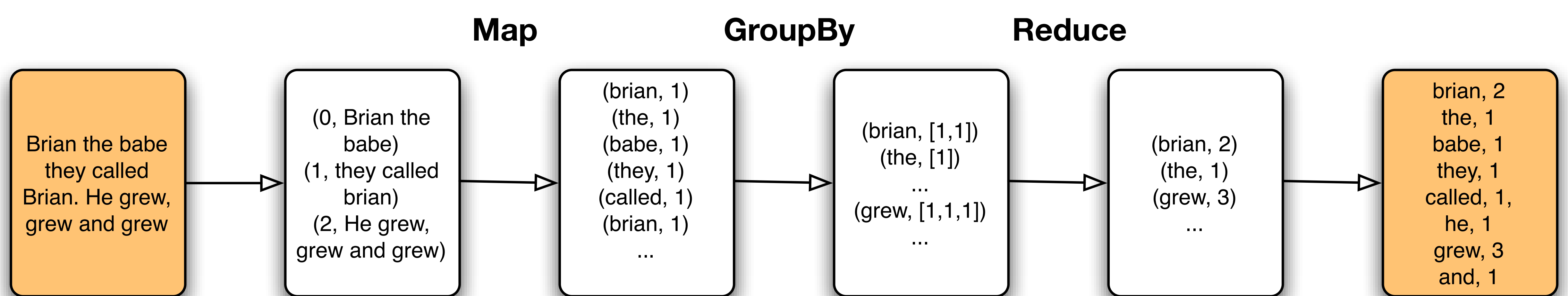
- A software framework introduced by Google
- Actually, Map-Groupby-Reduce
- Re-implemented by Apache Hadoop
- In Java, but with support for "Streaming"

```
    map: (k1,v1) -> list(k2,v2)
reduce: (k2, list (v2)) -> list(v3)
```

[http://wiki.apache.org/hadoop/HadoopPresentations]

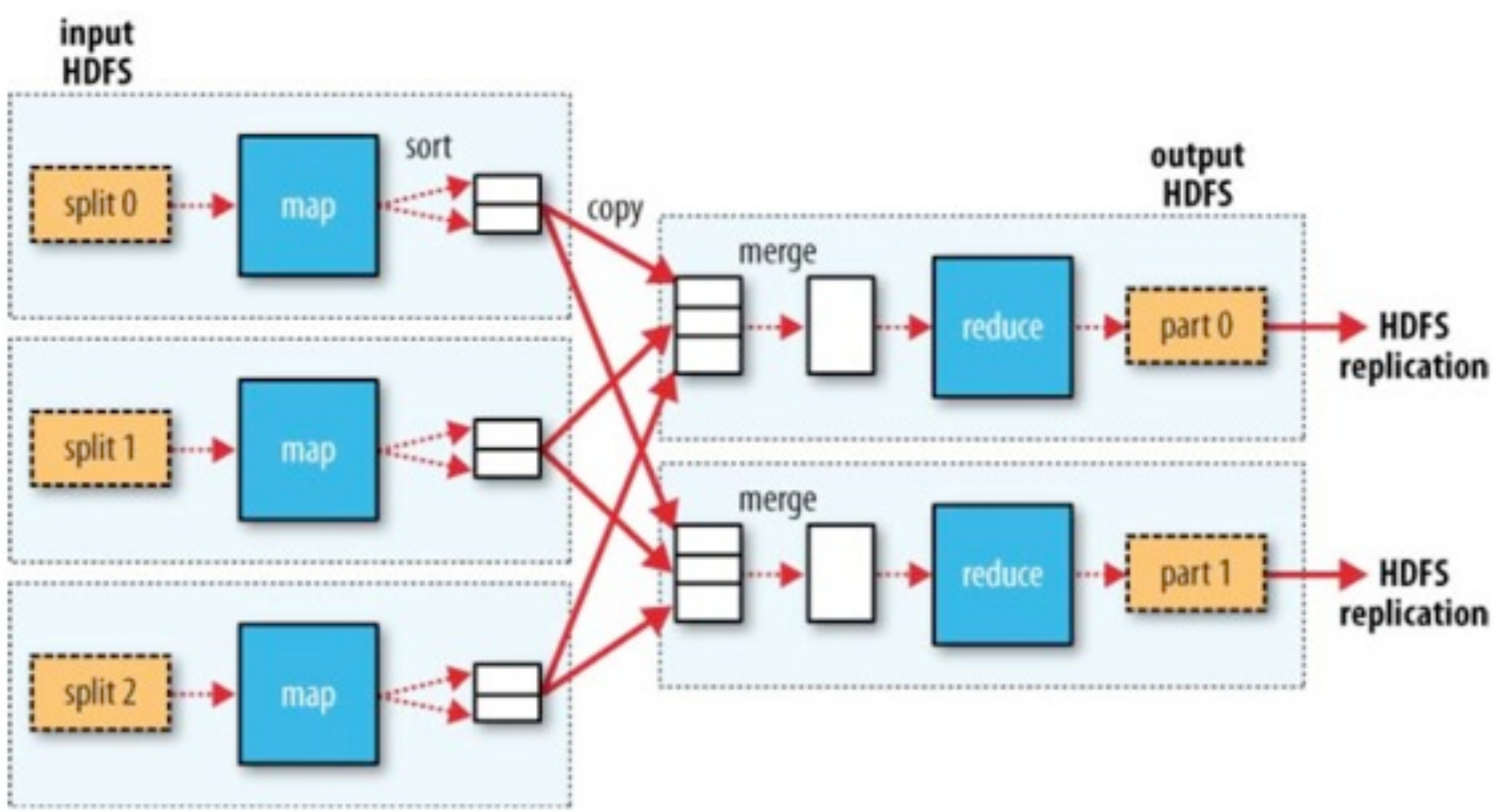
# Counting words

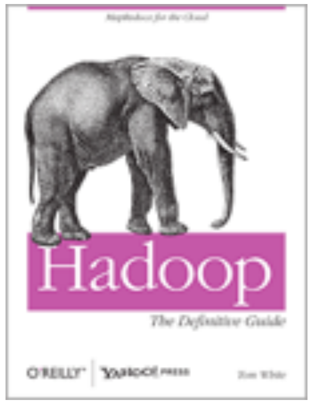
```python
import dumbo

def mapper(key,value):
    for word in value.split(): yield word,1

def reducer(key,values):
    yield key,sum(values)

if __name__ == "__main__":
    dumbo.run(mapper,reducer,combiner=reducer)
```



# Generating the result...

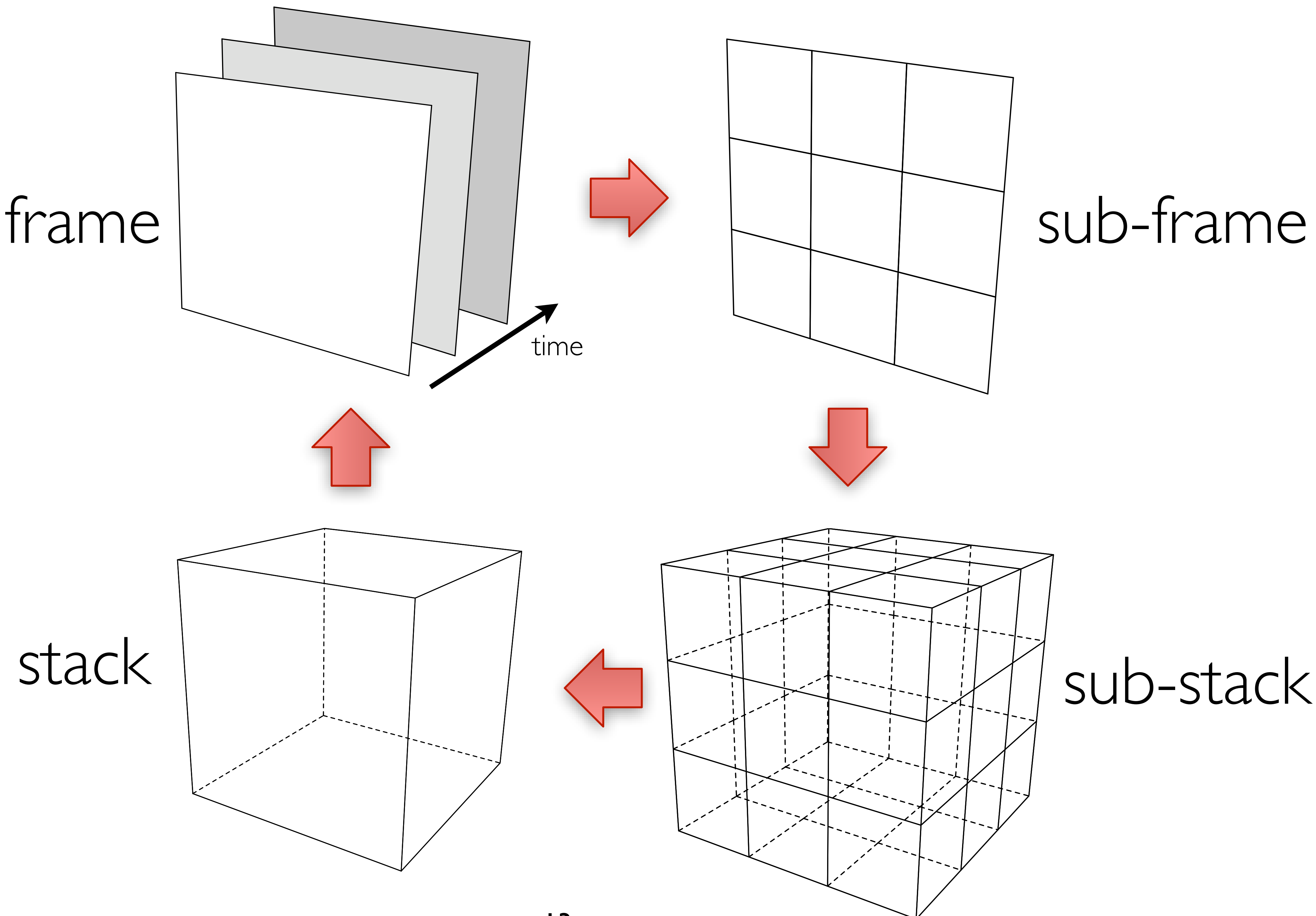
```
> dumbo put brian.txt brian.txt -hadoop /usr/lib/hadoop/
> dumbo start wordcount.py -input brian.txt -output brianwc -hadoop /usr/lib/hadoop/
> dumbo cat brianwc -hadoop /usr/lib/hadoop/
brian, 2
the, 1
babe, 1
they, 1
called, 1,
he, 1
grew, 3
and, 1
```




[Hadoop: The Definitive Guide, Tom White, 2009]

# Image/Video splitting



# Input and process

- Overide InputFormat and RecordReader
- Map function accepts stack/image
- Use OpenCV methods + Python to process
- Reduce function collates/sums

```
public class ImageFileInputFormat extends
FileInputFormat<Text, TypedBytesWritable>

public class ImageFileRecordReader implements
RecordReader<Text, TypedBytesWritable>
```

[http://opencv.willowgarage.com]

# An example

```
import dumbo
from opencv import *

def mapper(key,value):
  data = cvInitMatHeader(1, key, CV_8UC3, value)
  size = cvGetSize(data)
  gray = cvCreateImage(size,8,1)
  cvConvertImage(data,gray)
  hist = cvCreateHist([255], CV_HIST_ARRAY, [[0,256]], 1)
  cvCalcHist([gray], hist, 0, None)
  for i in range(255):
    yield i,cvRound(cvGetReal1D(hist.bins[0],i))


def reducer(key,values):
    yield key,sum(values)

if __name__ == "__main__":
    dumbo.run(mapper,reducer,combiner=reducer)
```

```
> dumbo cat lena -hadoop /usr/lib/hadoop/
...
23 0
24 1
25 7
26 22
27 28
28 63
29 93
30 135
...
```

# Another example


```
import dumbo
from opencv import *

# key is frame number, width & height are options
def mapper(key,value):
   data = cvInitMatHeader(width,height, CV_8UC3, value)
   size = cvGetSize(data)
   gray = cvCreateImage(size,8,1)
   cvConvertImage(data,gray)
   x,y = find(gray,background,mask) # NOTE: from cache file
   yield key,[x,y]

def reducer(key,values):
    yield key,collapse(values) # collapse separates list

if __name__ == "__main__":
    dumbo.run(mapper,reducer,combiner=reducer)
```

```
> dumbo cat
mousemove -
hadoop /usr/lib/
hadoop/
...
134 560 450
135 560 453
136 559 454
138 557 552
139 557 550
140 557 550
...
```

# Infrastructure & platforms

Pig, Hive, HBase, Cascading

cloudera

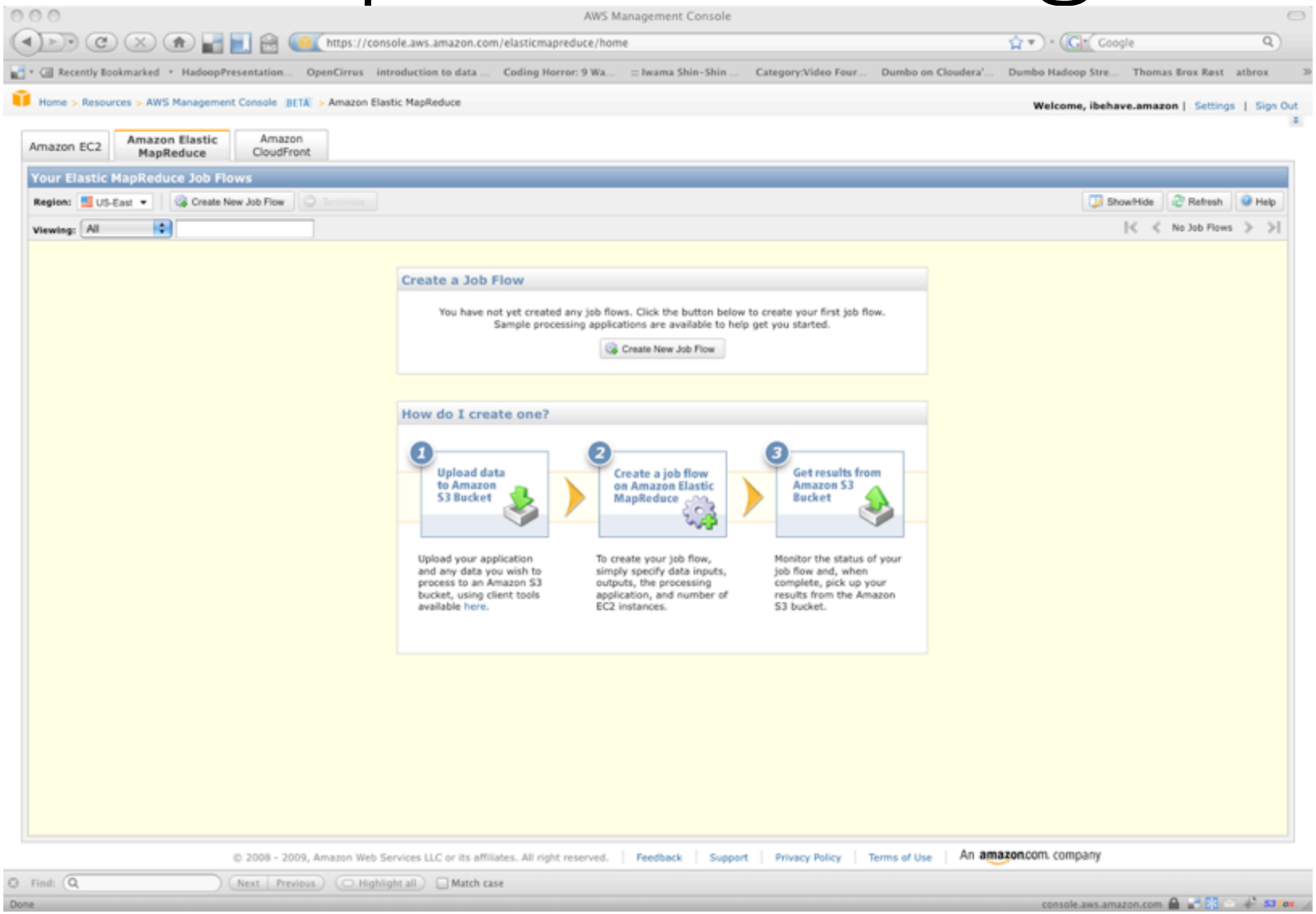
disco
massive data - minimal code

amazon webservices™

the rackspace cloud

Google™

# Up and running

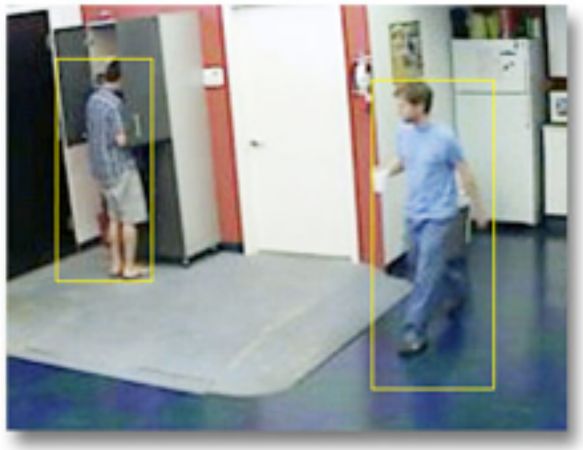



[aws.amazon.com]

# Apps: Machine vision

vitamin d

Numenta®



- The use of a Hierarchical Temporal Memory (HTM) “web-service”.
- Adding “meta-data” layers ontop of video.

[Jeff Hawkins, On Intelligence: How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines, 2004]

# Apps: Machine learning

| | single | multi |
|---|---|---|
| LWLR | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| LR | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| NB | $O(mn + nc)$ | $O(\frac{mn}{P} + nc\log(P))$ |
| NN | $O(mn + nc)$ | $O(\frac{mn}{P} + nc\log(P))$ |
| GDA | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| PCA | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| ICA | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| k-means | $O(mnc)$ | $O(\frac{mnc}{P} + mn\log(P))$ |
| EM | $O(mn^2 + n^3)$ | $O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2\log(P))$ |
| SVM | $O(m^2 n)$ | $O(\frac{m^2 n}{P} + n\log(P))$ |

- Algorithms that fit the Statistical Query model can be written in "summation form".
- E.g. for Taste recommendation filter.

[Chu *et al*, Map-Reduce for Machine Learning on Multicore, NIPS, 2006]

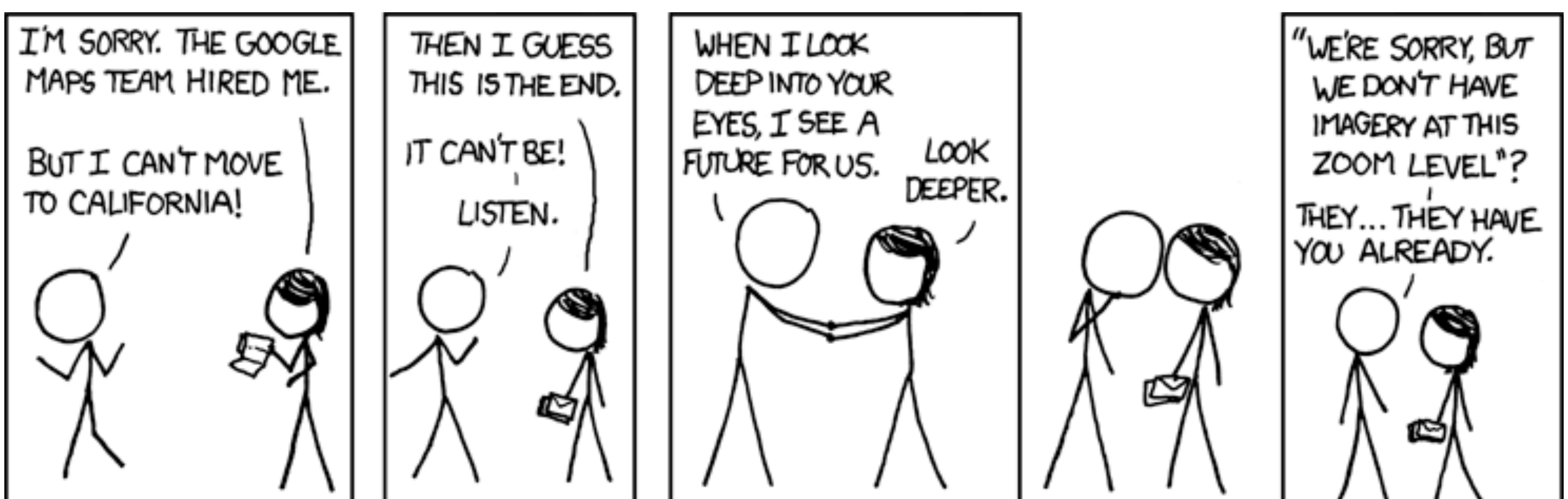
http://lucene.apache.org/mahout/

# Apps: Next?

- Registration
- Convolution
- Filtering
- = Framework
  (e.g. Yahoo Pipes)

# Conclusions

- Basic intro to cloud computing.
- Writing and deploying scientific code.
- The peculiarities of handling images.
- Where next?




[xkcd.com]

# Clear Skies Beyond!