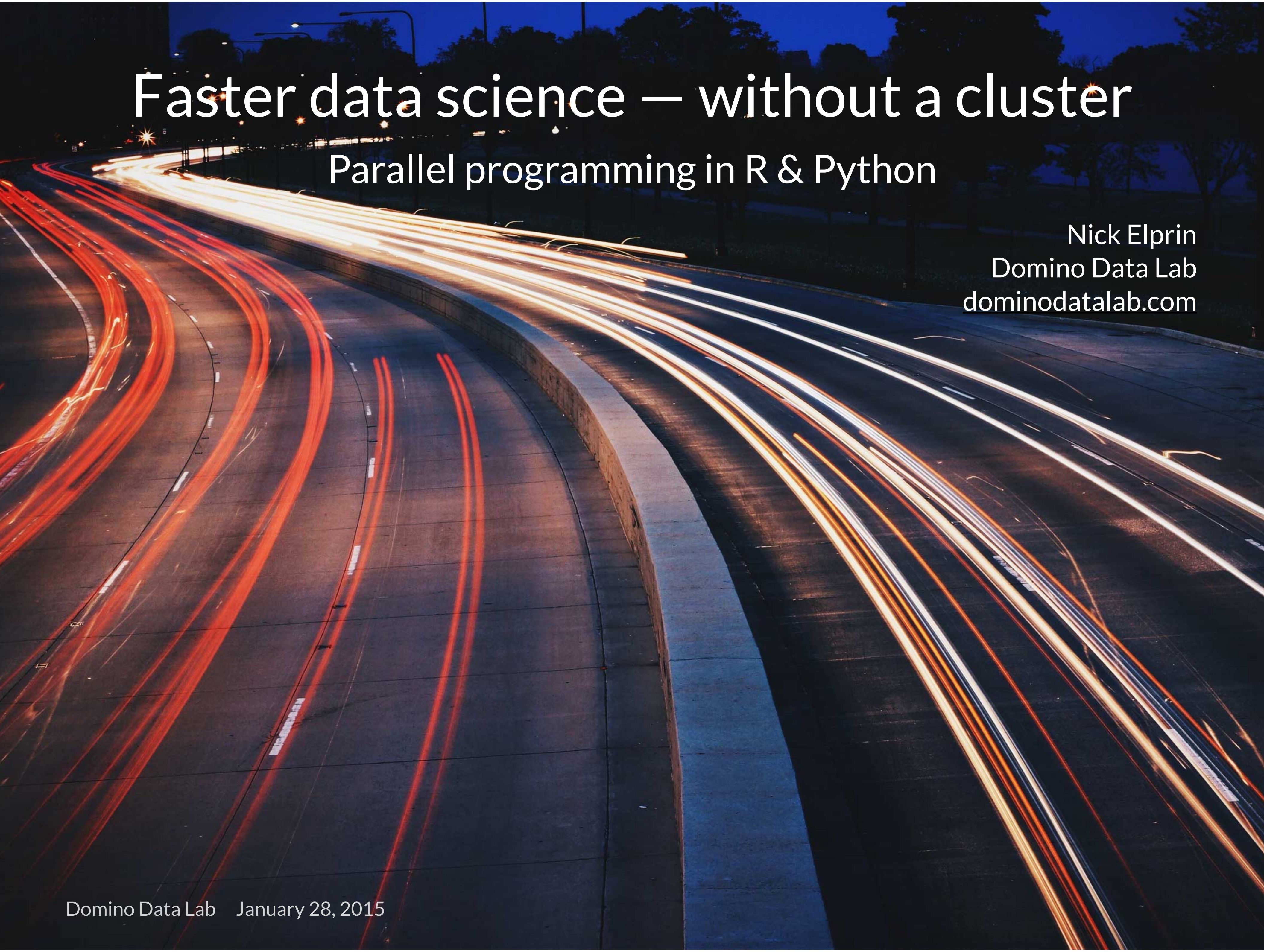


Faster data science — without a cluster

Parallel programming in R & Python

Nick Elprin
Domino Data Lab
dominodatalab.com

Who am I?

- Founder of Domino Data Lab, a software platform for enterprise data science
- Previously built analytical software at a big hedge fund
- BA, MS in computer science

Outline

- Motivation
- Basic conceptual intro to parallelism, general principles and pitfalls
- Machine learning applications
- Python examples (general and machine-learning focused)
- R examples (general and machine-learning focused)
- Questions

Motivation

"Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."

— *Dan Ariely*

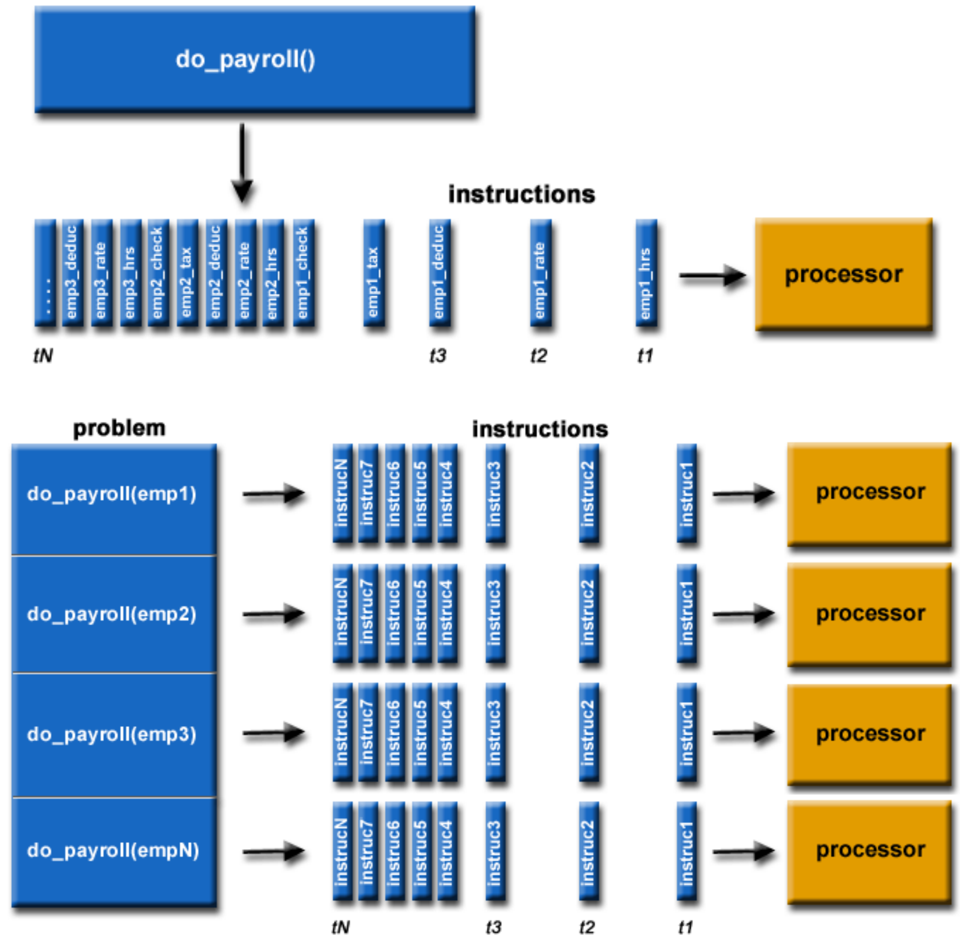
- Lots of "medium data" problems
  - Can fit in memory on one machine
- Lots of naturally parallel problems

- Easy to access large machines
- Clusters are hard
- Not everything fits map-reduce

Model	vCPU	Mem (GiB)	SSD Storage (GB)
r3.large	2	15.25	1 x 32
r3.xlarge	4	30.5	1 x 80
r3.2xlarge	8	61	1 x 160
r3.4xlarge	16	122	1 x 320
r3.8xlarge	32	244	2 x 320

Parallel programing 101

- Think about independent tasks (hint: “for” loops are a good place to start!)
  - Should be CPU-bound tasks
- Warning and pitfalls
  - Not a substitute for good code
  - Overhead
  - Shared resource contention
  - Thrashing



Source: Blaise Barney, Lawrence Livermore National Laboratory

Can parallelize at different "levels"

Experiments	Run different analyses at once
Algorithms	Write your code (or use a package) to parallelize functions or steps within your analysis
Math ops	Run against underlying libraries that parallelize low-level operations, e.g., openBLAS, ATLAS

Will focus on algorithms, with some brief comments on Experiments

Common Operation: Map

```
M = function(item) {
  manipulatedItem = ...
  manipulatedItem
}
```

So what's map-reduce?

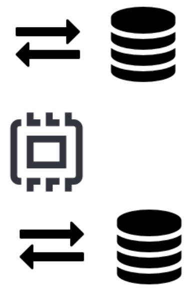
Parallelize tasks to match your resources

- Computing something (CPU)
- Reading from disk/database
- Writing to disk/database
- Network IO (e.g., web scraping)

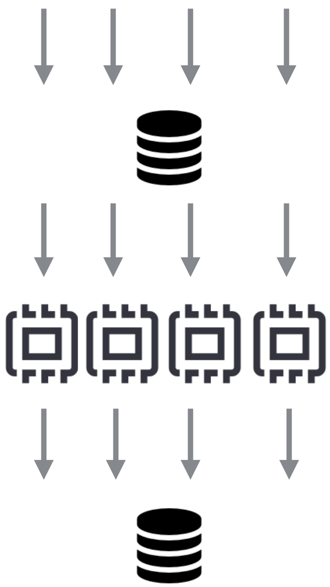
Saturating a resource will create a bottleneck

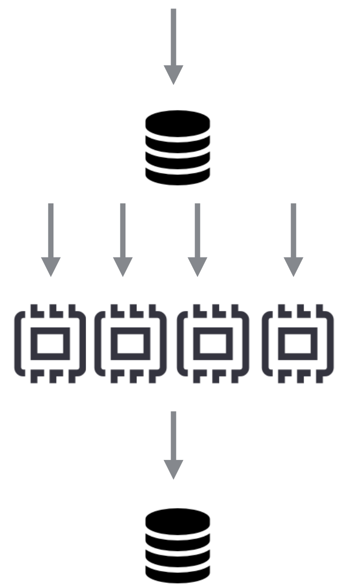
Parallelize tasks to match your resources

```
itemIDs = [1, 2, … , n]
parallel-for-each(i = itemIDs){
   item = fetchData(i)
   result = computeSomething(item)
   saveResult(result)
}
```



```
items = fetchData([1, 2, … , n])
results = parallel-for-each(i = items){
   computeSomething(item)
}
saveResult(results)
```

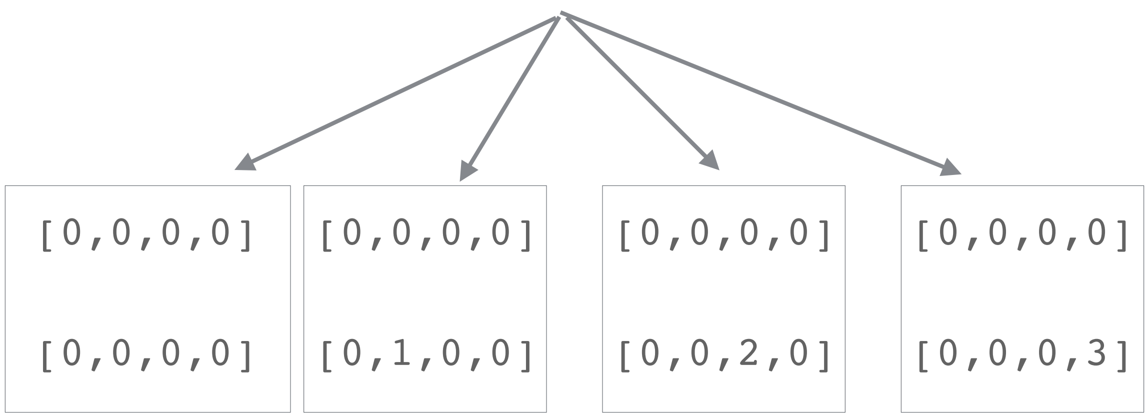




Avoid modifying global state

```
itemIDs = [0, 0, 0, 0]
parallel-for-each(i = 1:4) {
  itemIDs[i] = i
}
```

Array initialized in process 1   `A = [0,0,0,0]`



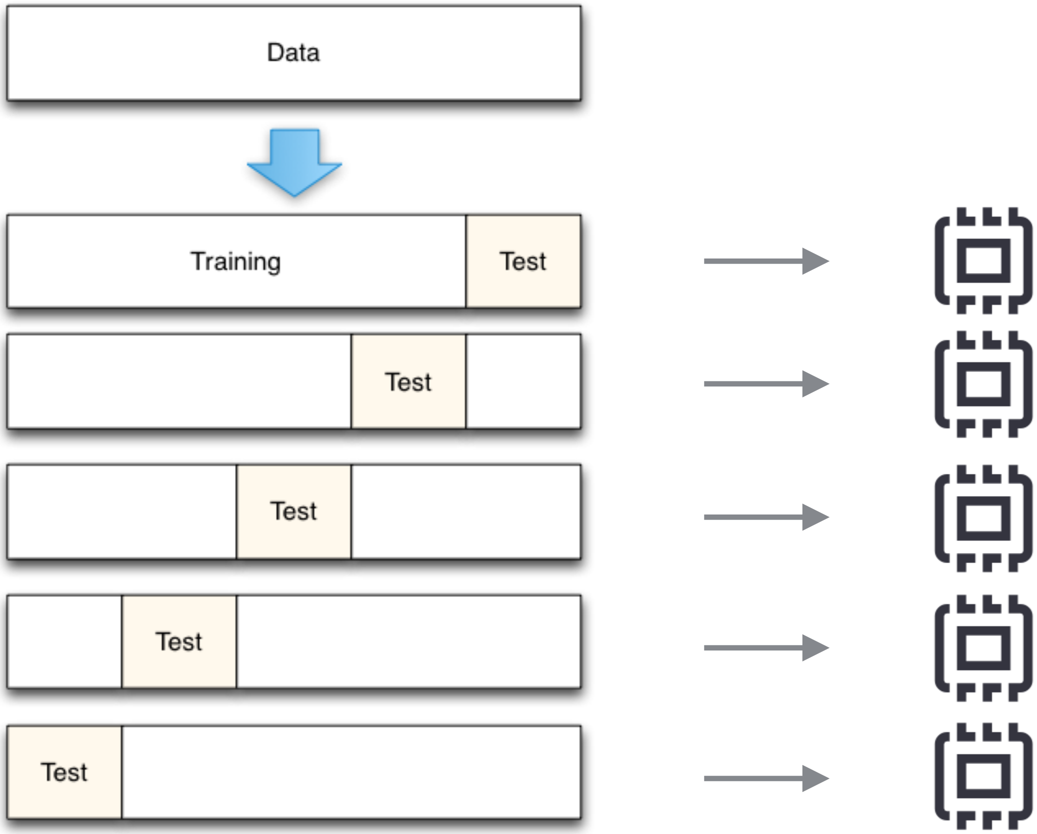
Array copied to each sub-process

The *copy* is modified

When all parallel tasks finish, array in original process remained unchanged   `[0,0,0,0]`

Many ML tasks are naturally parallelized

Cross validation



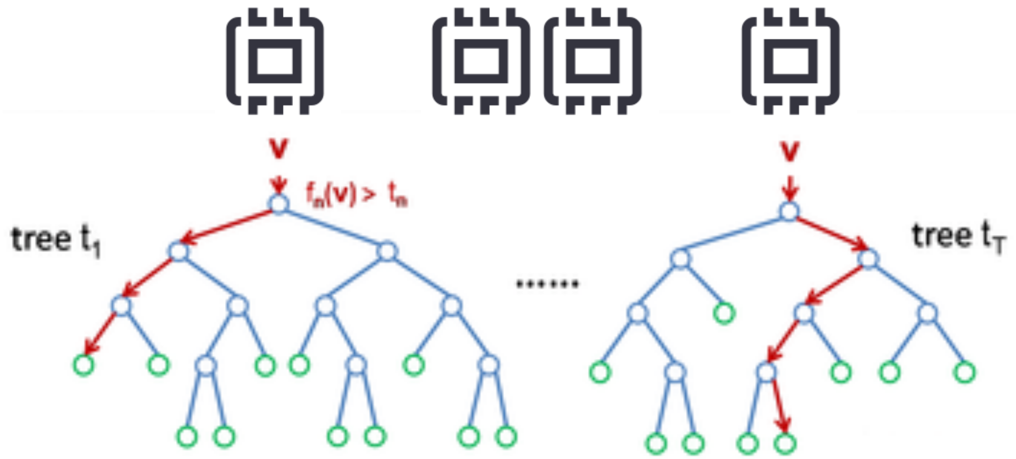
Grid search

C

Kernel

	1	10	100	1000
Linear				
RBF				

Random forest



More subtle

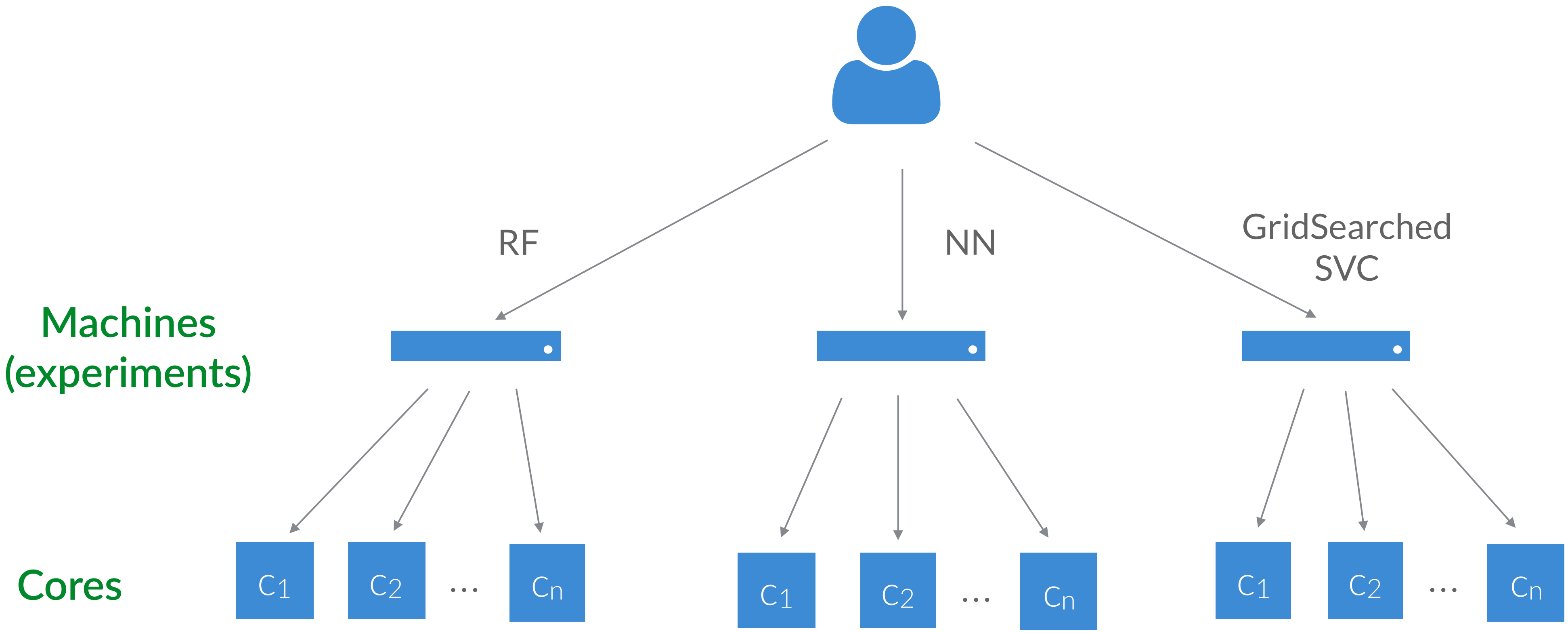
- KMeans
- Neural networks

Parallel programing in Python

- Joblib
  pythonhosted.org/joblib/parallel.html
- IPython Notebook clusters
  www.astro.washington.edu/users/vanderplas/Astr599/notebooks/21_IPythonParallel
- scikit-learn (*n_jobs*) scikit-learn.org
  - `GridSearchCV`
  - `RandomForest`
  - `KMeans`
  - `cross_val_score`

Demo

Can compose layers of parallelism



Demo

Parallel programing in R

- General purpose
  - `parallel`
  - `foreach`
    cran.r-project.org/web/packages/foreach
- More specialized
  - `randomForest`
    cran.r-project.org/web/packages/randomForest
  - `caret`
    topepo.github.io/caret
  - `plyr`
    cran.r-project.org/web/packages/plyr

Demo

Check us out

DOMINO

dominodatalab.com

blog.dominodatalab.com

@dominodatalab