Scaling Machine Learning

Models, Architectures and Algorithms

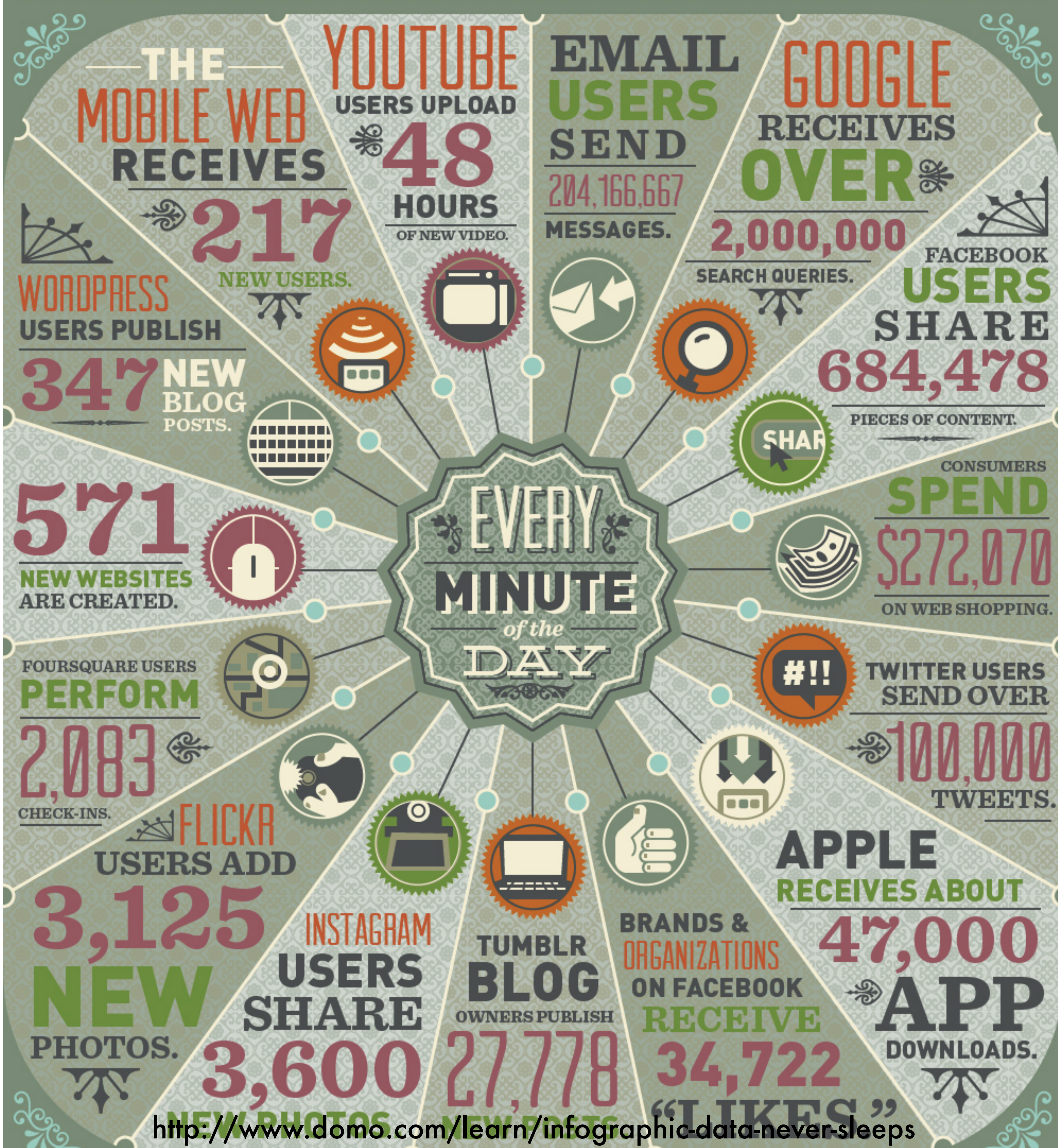
Alexander Smola & Amr Ahmed
Carnegie Mellon University & Google
alex.smola.org @smolix

Outline

- **Data**
  Actions, Interactions, User generated content
- **Architectures**
  MapReduce, Graphs, Streams, Parameterserver
- **Models and Algorithms**
  - Logistic regression (advertising, search)
    Distributed proximal gradient
  - Scaling Topic models (personalization, profiling)
  - Modeling (user generated) data and behavior


Data

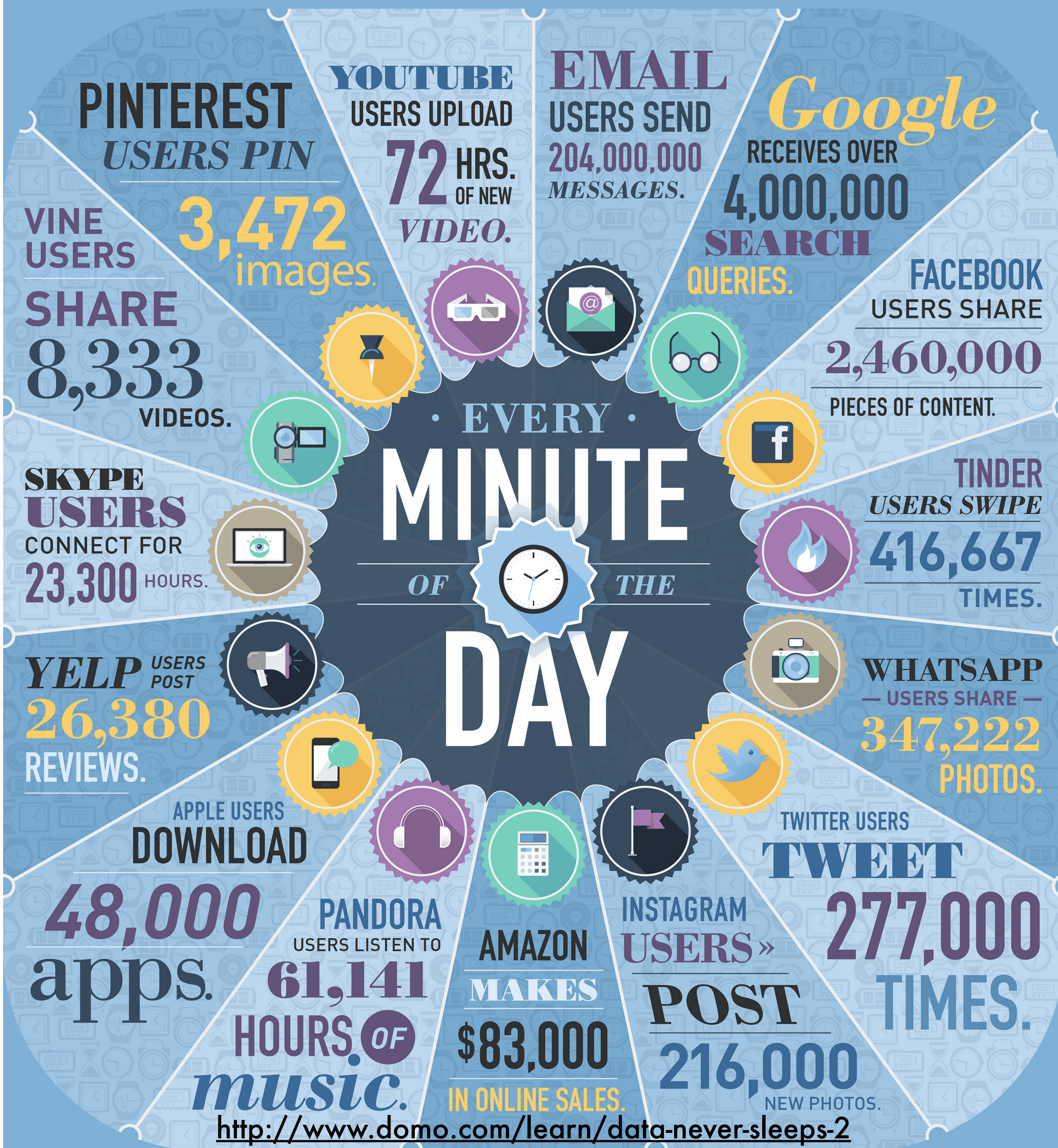
Data per minute 2012



http://www.domo.com/learn/infographic-data-never-sleeps

Google

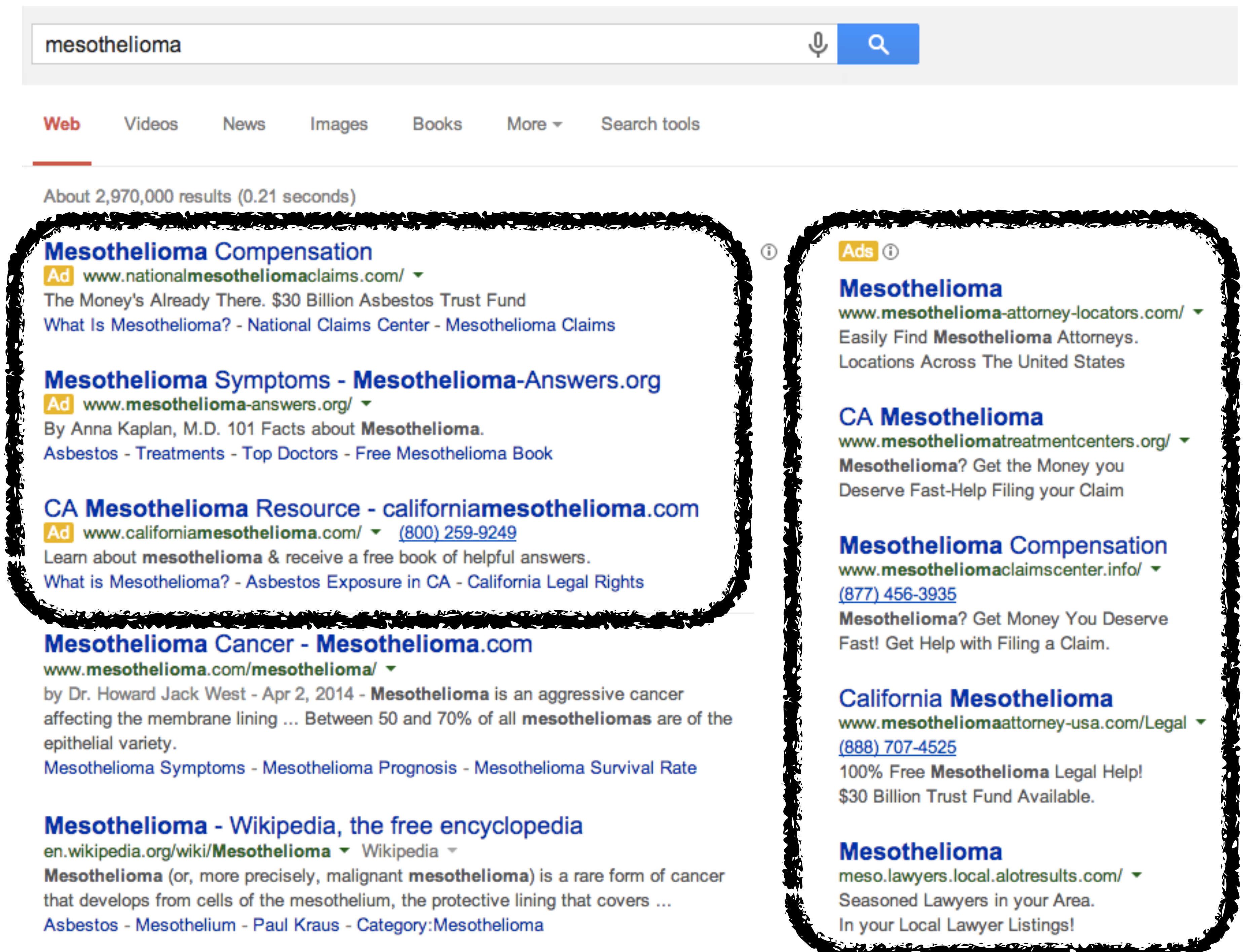
Data per minute 2014



http://www.domo.com/learn/data-never-sleeps-2

Google

Computational Advertising



sponsored search picks position of ad using

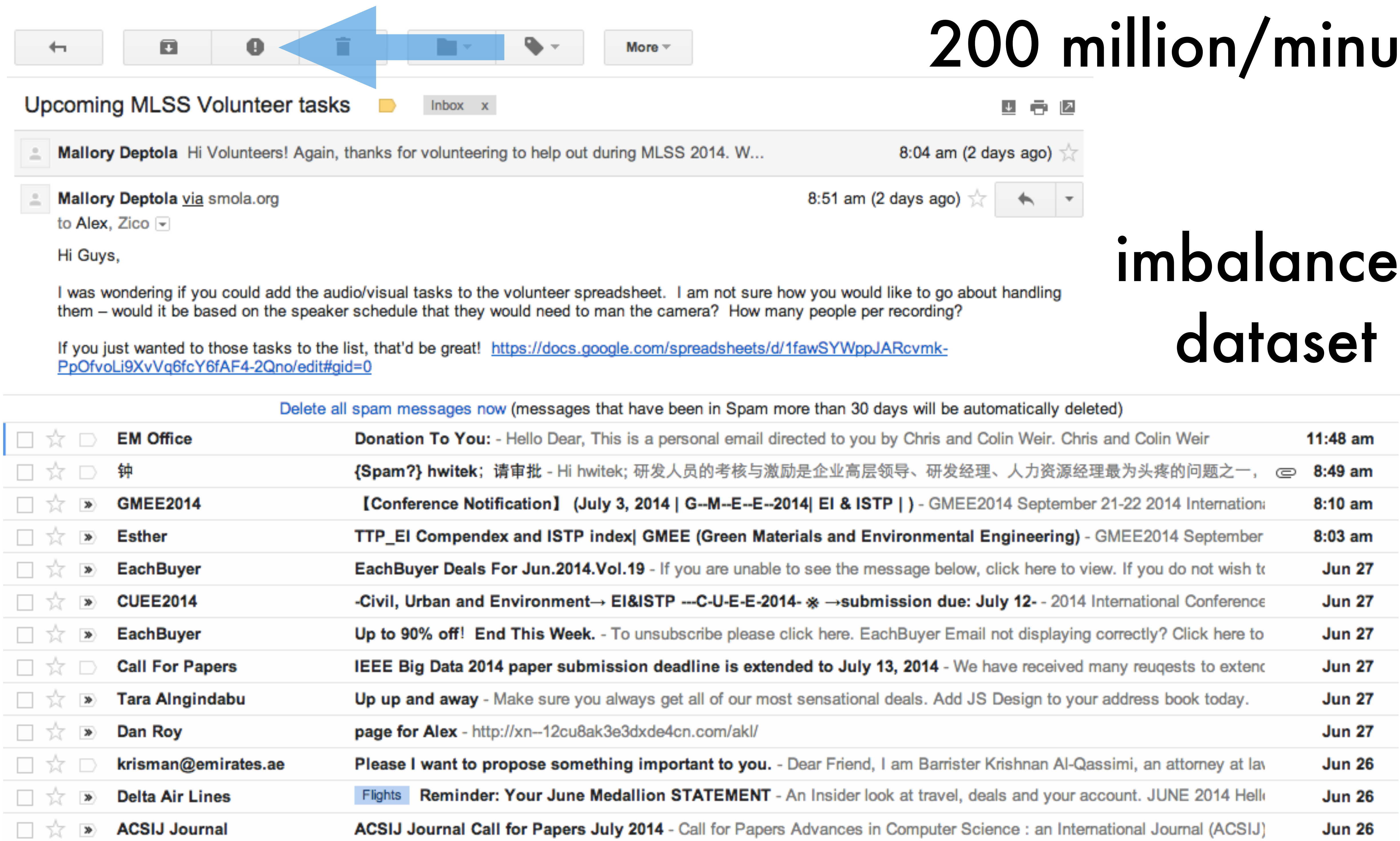
$$p(\text{click}|\text{ad}) \cdot \text{bid}(\text{ad})$$

estimate it

4 million/minute

Carnegie Mellon University

Spam filtering

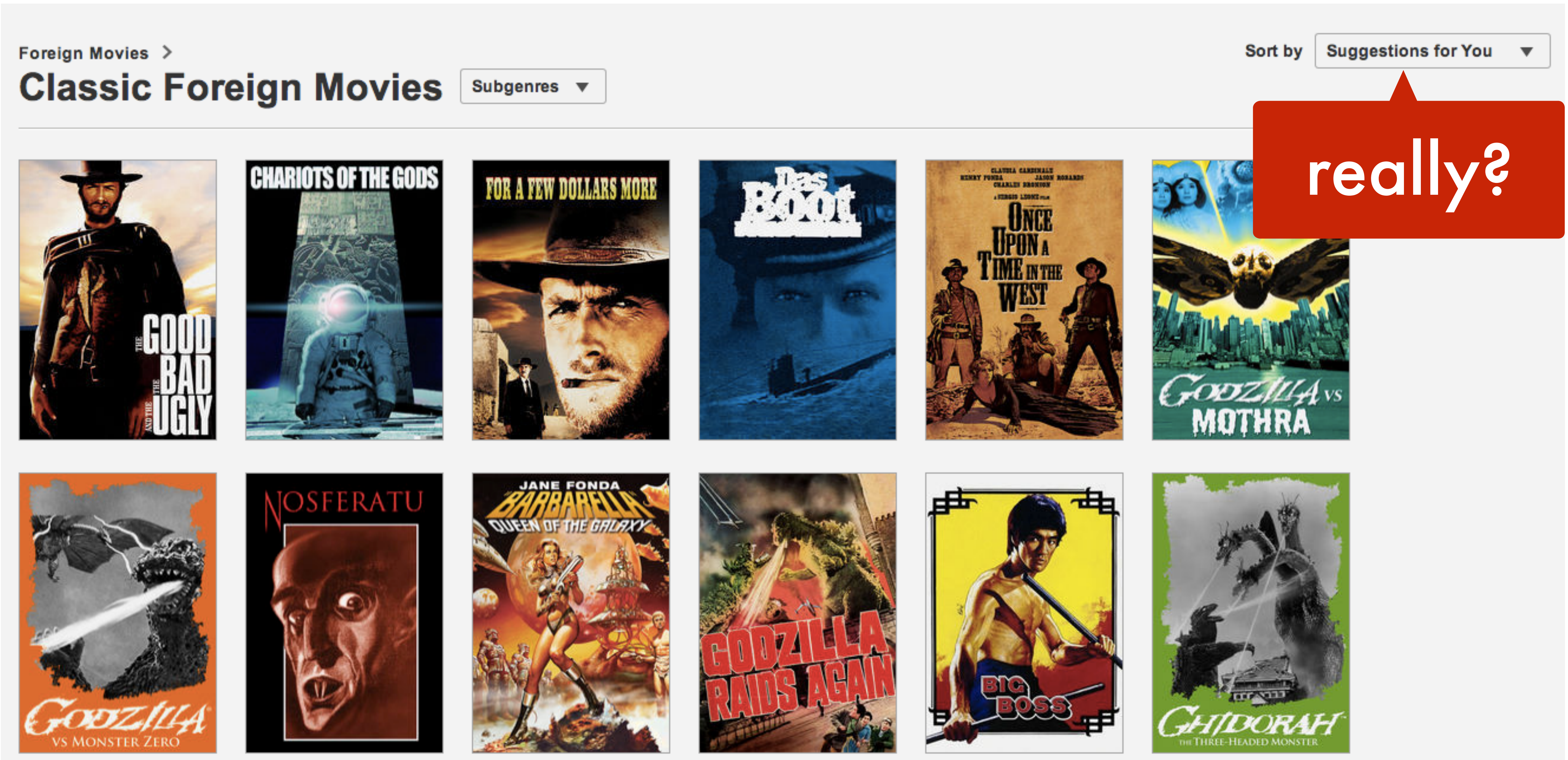


200 million/minute

imbalanced dataset

Recommendation & Ranking

maximize interaction probability for whole page



really?

Time series & trends

Topics

Subscribe

machine learning
Search term

data mining
Search term

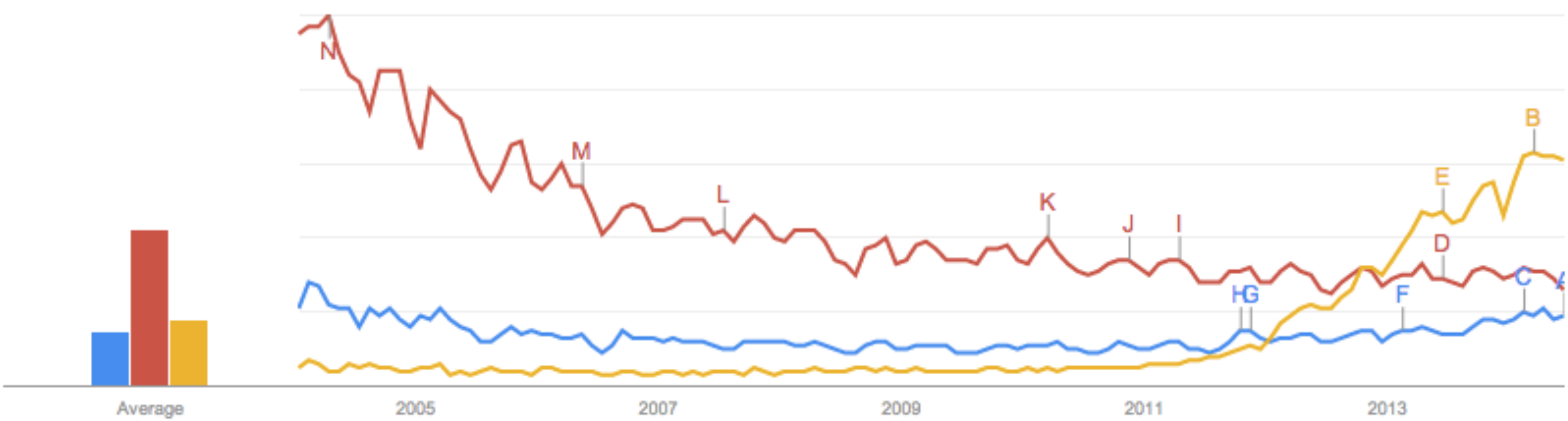
big data
Search term

+Add term

Interest over time

News headlines

Forecast



Google

Carnegie Mellon University

More data

- News articles & events (NY Times, GNews)
- Blogs / microblogs (Tumblr, Twitter, Weibo)
- Reviews (IMDB, Yelp, Amazon)
- Comments (YouTube, Reddit)
- Messages (Facebook, Hangouts, SMS)
- Graphs (Friends, Followers, Webpages)
- Information diffusion (Meme tracking)
- Spatiotemporal (GMaps, Foursquare, Twitter)

Lots more data

- Bioinformatics
  DNA Microarrays, High throughput sequencing
- Astronomy
  Square Kilometer Array, Radio telescopes
- Medicine
  MRI / MEG scans, Connectome, Health records
- Finance (e.g. high frequency trading)
- Geophysics (e.g. oil discovery)
- Industrial process monitoring

Summary

- Expensive data ≠ big data
  (1000 brain scans are expensive)
- Big data requires big models
  (1000 parameter model on TB of data)
- Big data needs systems built for it
  (don't ship data to computation)
- Vast range of problem domains
- Vast range of statistical models

Google

Carnegie Mellon University


Architecture


Real Hardware
google.com/datacenters
Google
tweden
049

Machines

Bulk transfer is at least 10x faster

- CPU
  - –8-64 cores (Intel/AMD servers)
  - –2-3 GHz (close to 1 IPC per core peak) - over 100 GFlops/socket
  - –8-32 MB Cache (essentially accessible at clock speed)
  - –Vectorized multimedia instructions (AVX 256bit wide, e.g. add, multiply, logical)

- RAM
  - –16-256 GB depending on use
  - –3-8 memory banks (each 32bit wide - atomic writes!)
  - –DDR3 (up to 100GB/s per board, random access 10x slower)

- Harddisk
  - –4 TB/disk
  - –100 MB/s sequential read from SATA2
  - –5ms latency for 10,000 RPM drive, i.e. random access is slow

- Solid State Drives
  - –500 MB/s sequential read
  - –Random writes are really expensive (read-erase-write cycle for a block)

The real joy of hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures

Jeff Dean's Stanford slides

slow disks, bad memory, misconfigured machines, flaky machines, etc.

Why a single machine is not enough

- Data (lower bounds)
  - 10-100 Billion documents (webpages, e-mails, ads, tweets)
  - 100-1000 Million users on Google, Facebook, Twitter, Hotmail
  - 1 Million days of video on YouTube
  - 100 Billion images on Facebook
- Processing capability for single machine 1TB/hour
  But we have much more data
- Parameter space for models is too big for a single machine
  Personalize content for many millions of users
- Process on many cores and many machines simultaneously

Cloud pricing

- Google Compute Engine and Amazon EC2

Instance type	Virtual Cores	Memory	Price (US$)/Hour (US hosted)
n1-standard-1	1	3.75GB	$0.070
n1-standard-2	2	7.5GB	$0.140
n1-standard-4	4	15GB	$0.280
n1-standard-8	8	30GB	$0.560
n1-standard-16	16	60GB	$1.120

$10,000/year

- Storage

Standard Provisioned Space	$0.04 GB / month
SSD Provisioned Space	$0.325 GB / month
Snapshot storage	$0.125 GB / month
IO operations	No additional charge

Spot instances much cheaper

Amazon EBS General Purpose (SSD) volumes
- $0.10 per GB-month of provisioned storage

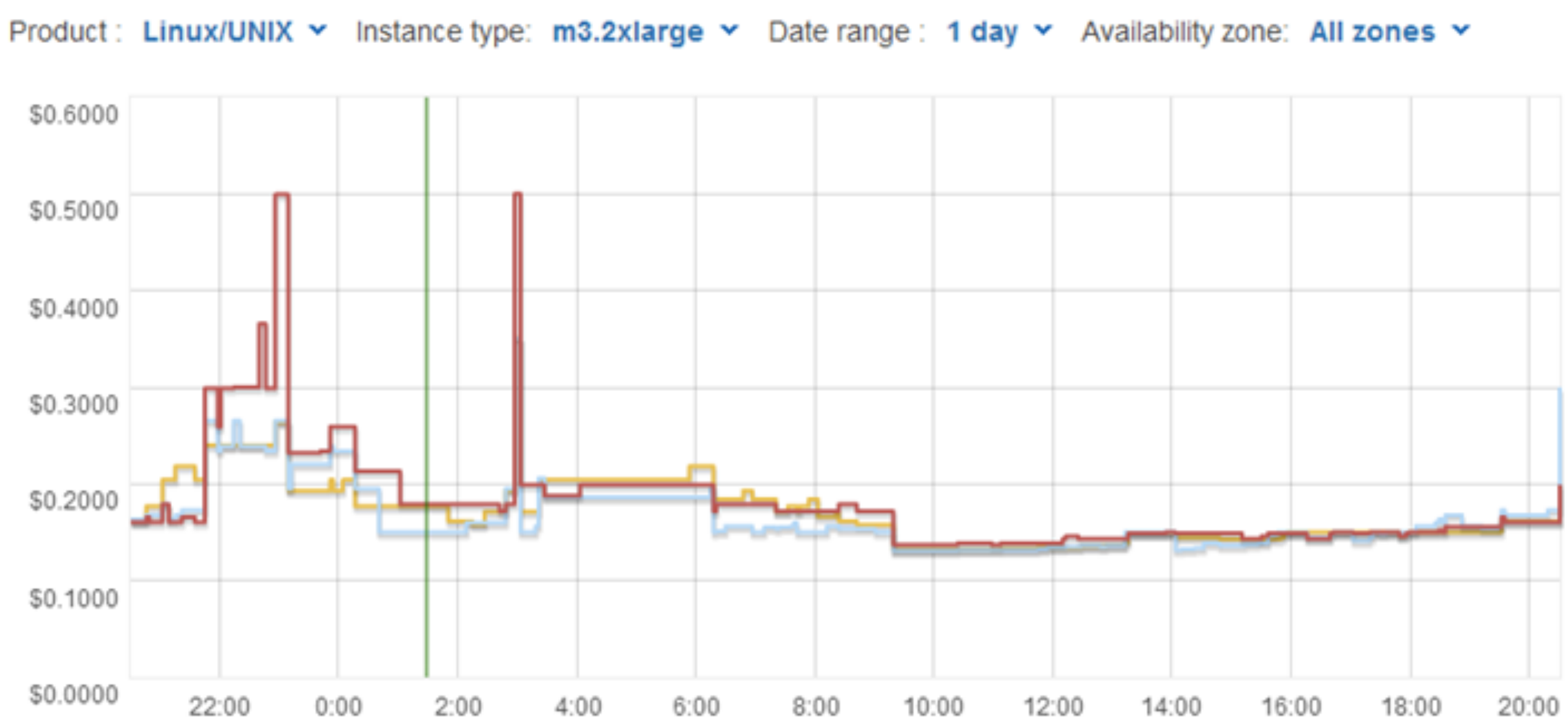
Amazon EBS Provisioned IOPS (SSD) volumes
- $0.125 per GB-month of provisioned storage
- $0.10 per provisioned IOPS-month

Amazon EBS Magnetic volumes
- $0.05 per GB-month of provisioned storage
- $0.05 per 1 million I/O requests

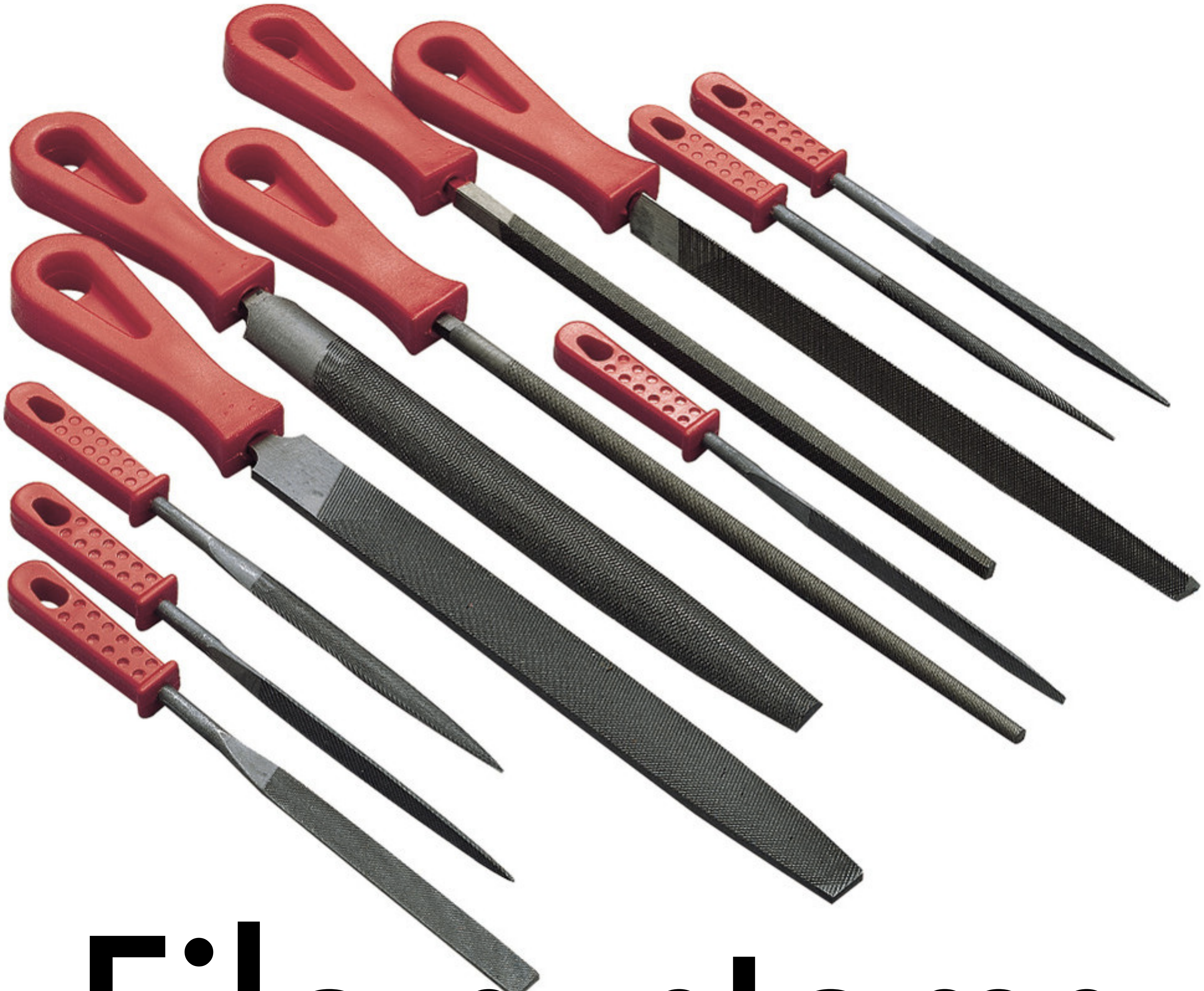
Amazon EBS Snapshots to Amazon S3
- $0.095 per GB-month of data stored

Real Hardware

- can and will fail
- Spot instances much cheaper (but can lead to preemption). Design algorithms for it!



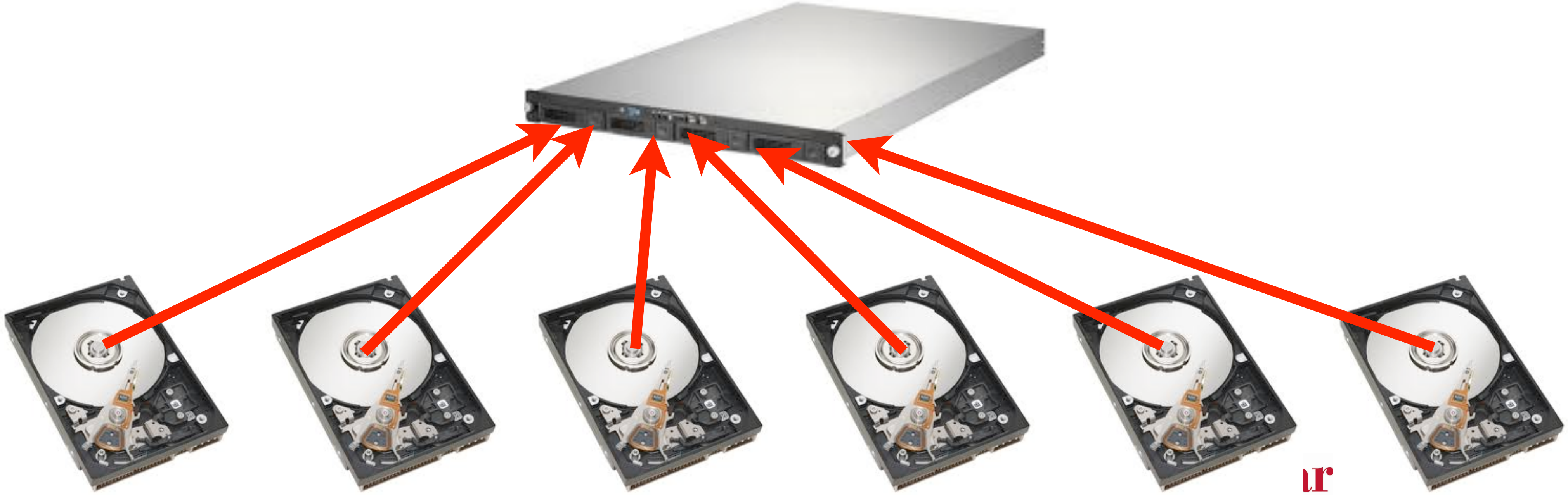

Work & storage
Waste
Location



File systems

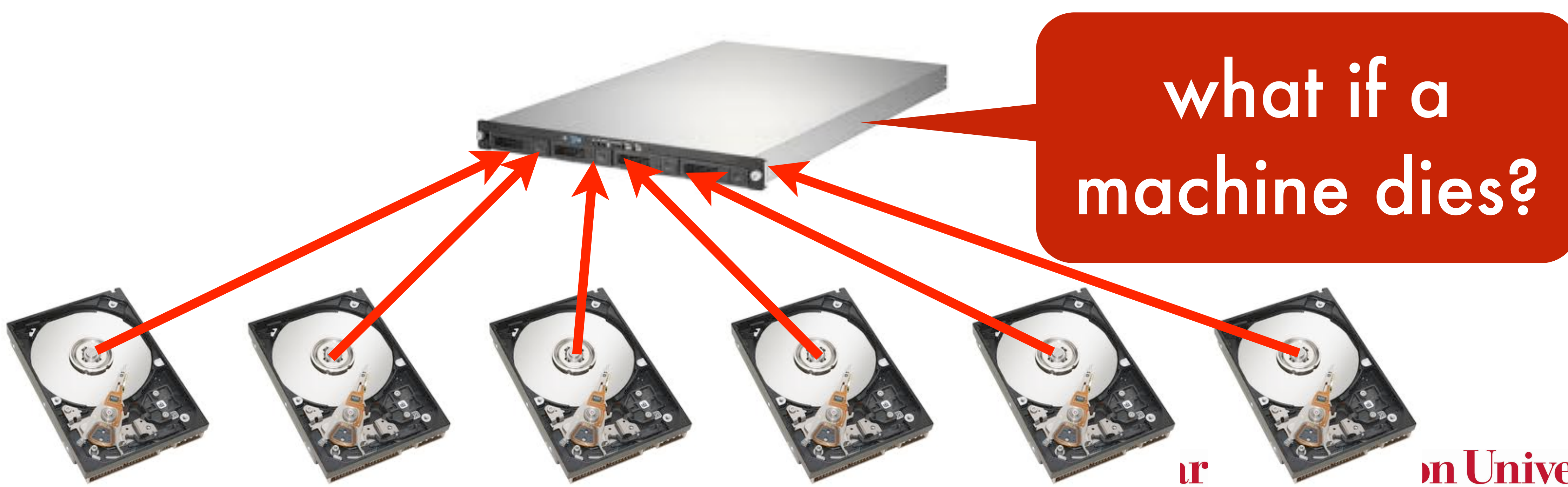
RAID

- Redundant array of inexpensive disks (optional fault tolerance)
  - Aggregate storage of many disks
  - Aggregate bandwidth of many disks
- RAID 0 - stripe data over disks (good bandwidth, faulty)
- RAID 1 - mirror disks (mediocre bandwidth, fault tolerance)
- RAID 5 - stripe data with 1 disk for parity (good bandwidth, fault tolerance)
- Even better - use error correcting code for fault tolerance, e.g. (4,2) code, i.e. two disks out of 6 may fail



RAID

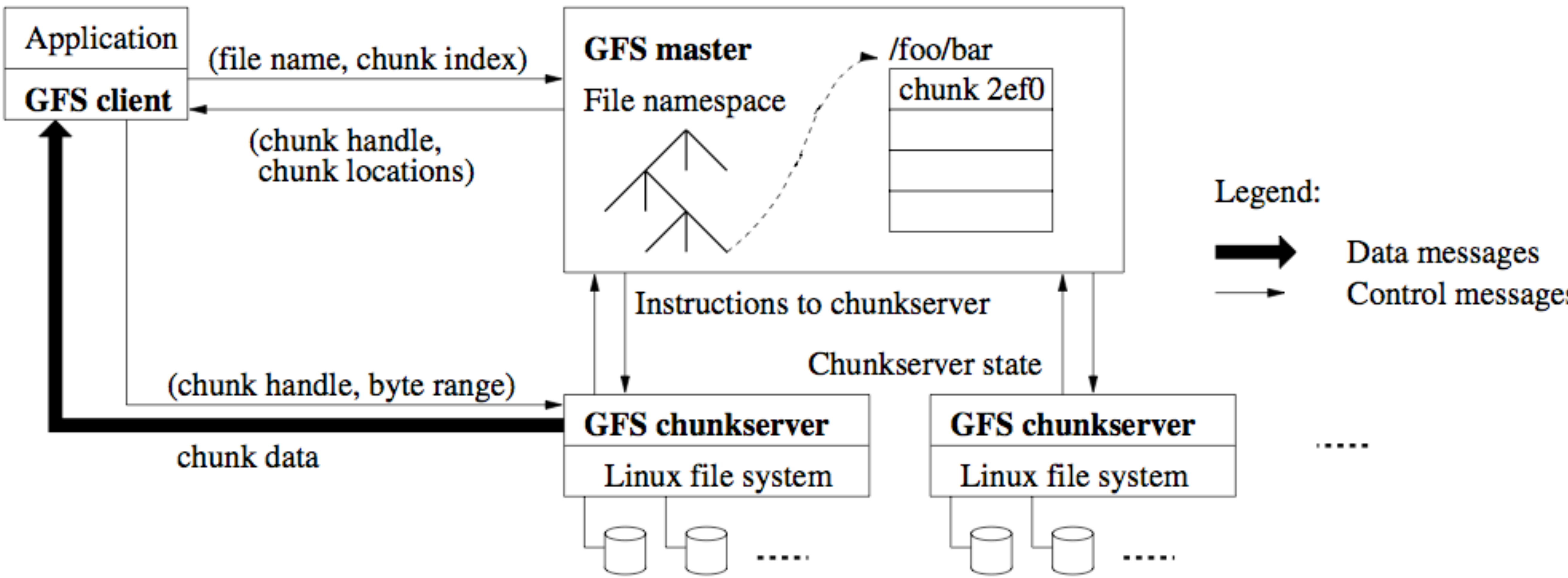
- Redundant array of inexpensive disks (optional fault tolerance)
  - Aggregate storage of many disks
  - Aggregate bandwidth of many disks
- RAID 0 - stripe data over disks (good bandwidth, faulty)
- RAID 1 - mirror disks (mediocre bandwidth, fault tolerance)
- RAID 5 - stripe data with 1 disk for parity (good bandwidth, fault tolerance)
- Even better - use error correcting code for fault tolerance, e.g. (4,2) code, i.e. two disks out of 6 may fail



Distributed replicated file systems

- Internet workload
  - Bulk sequential writes
  - Bulk sequential reads
  - No random writes (possibly random reads)
  - High bandwidth requirements per file
  - High availability / replication
- Non starters
  - Lustre (high bandwidth, but no replication outside racks)
  - Gluster (POSIX, more classical mirroring, see Lustre)
  - NFS/AFS/whatever - doesn't actually parallelize

Google File System / HadoopFS

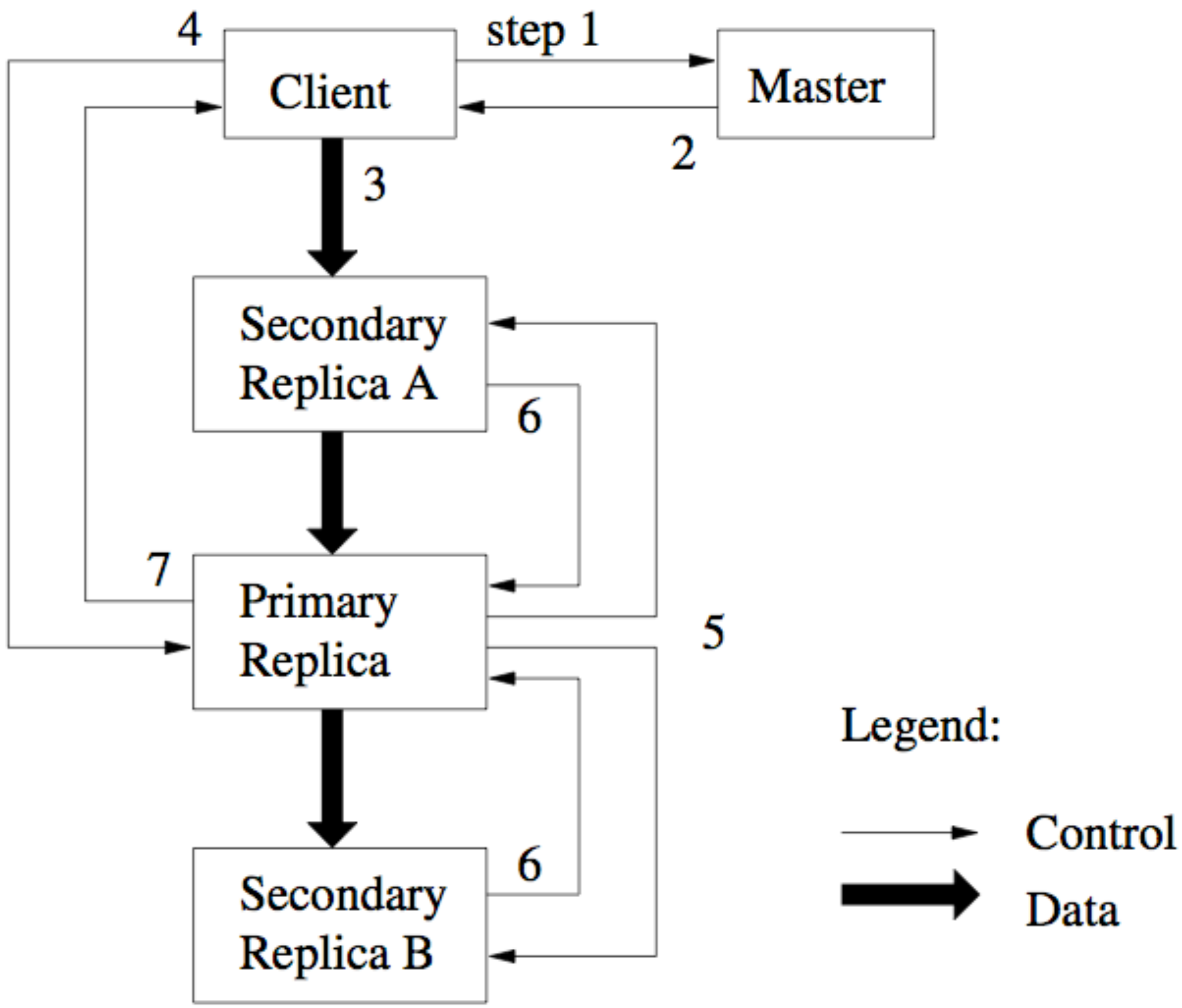


Ghemawat, Gobioff, Leung, 2003

- Chunk servers hold blocks of the file (64MB per chunk)
- Replicate chunks (chunk servers do this autonomously). More bandwidth and fault tolerance
- Master distributes, checks faults, rebalances (Achilles heel)
- Client can do bulk read / write / random reads

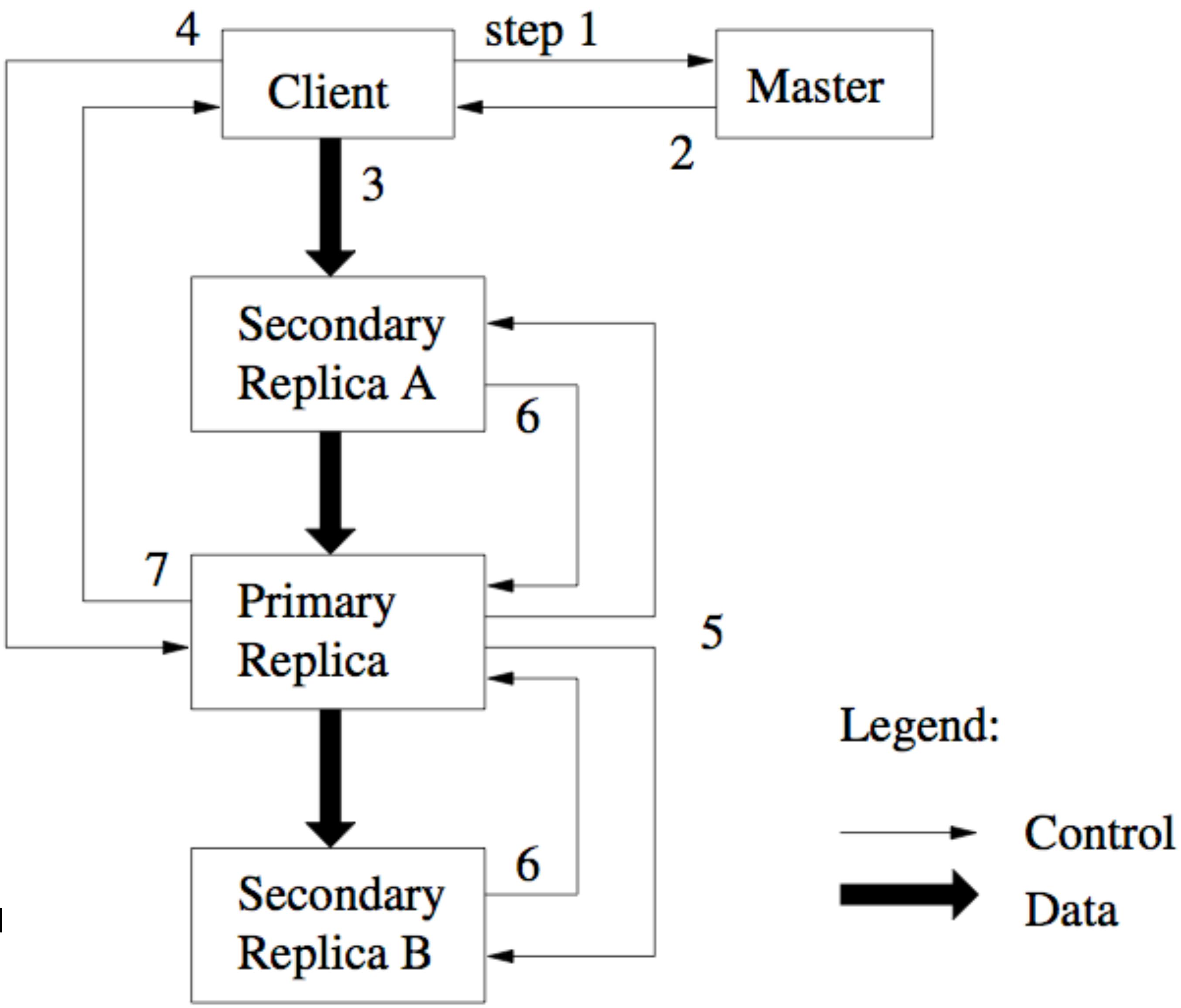
Google File System / HDFS

- Client requests chunk from master
- Master responds with replica location
- Client writes to replica A
- Client notifies primary replica
- Primary replica requests data from rep
- Replica A sends data to Primary replica
- Primary replica confirms write to clien



Google File System / HDFS

- Client requests chunk from master
- Master responds with replica location
- Client writes to replica A
- Client notifies primary replica
- Primary replica requests data from rep
- Replica A sends data to Primary replica
- Primary replica confirms write to clien



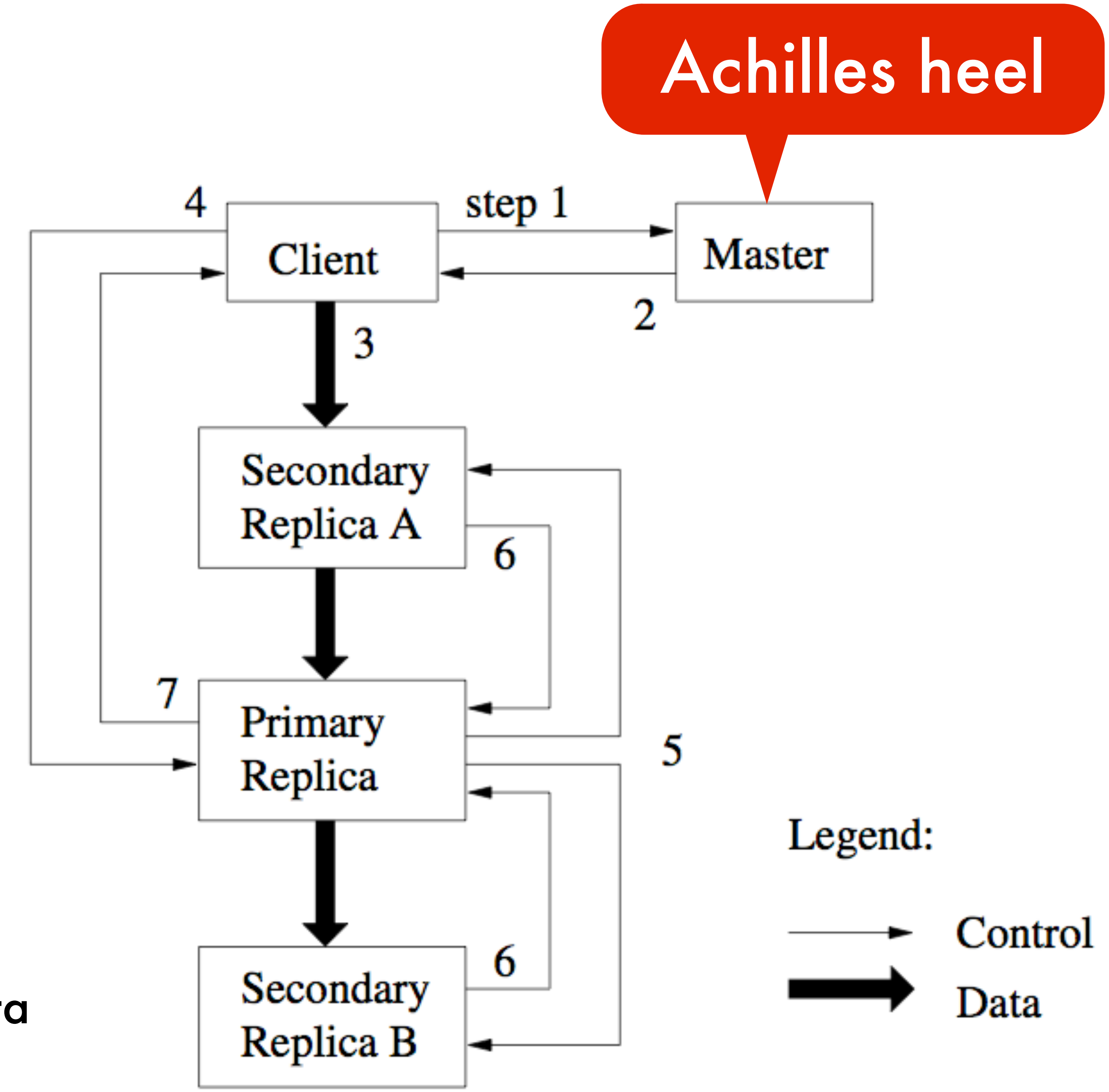
- Master ensures nodes are live
- Chunks are checksummed
- Can control replication factor for hotspots / load balancing
- Deserialize master state by loading data structure as flat file from disk (fast)

Google

Carnegie Mellon University

Google File System / HDFS

- Client requests chunk from master
- Master responds with replica location
- Client writes to replica A
- Client notifies primary replica
- Primary replica requests data from rep
- Replica A sends data to Primary replica
- Primary replica confirms write to clien

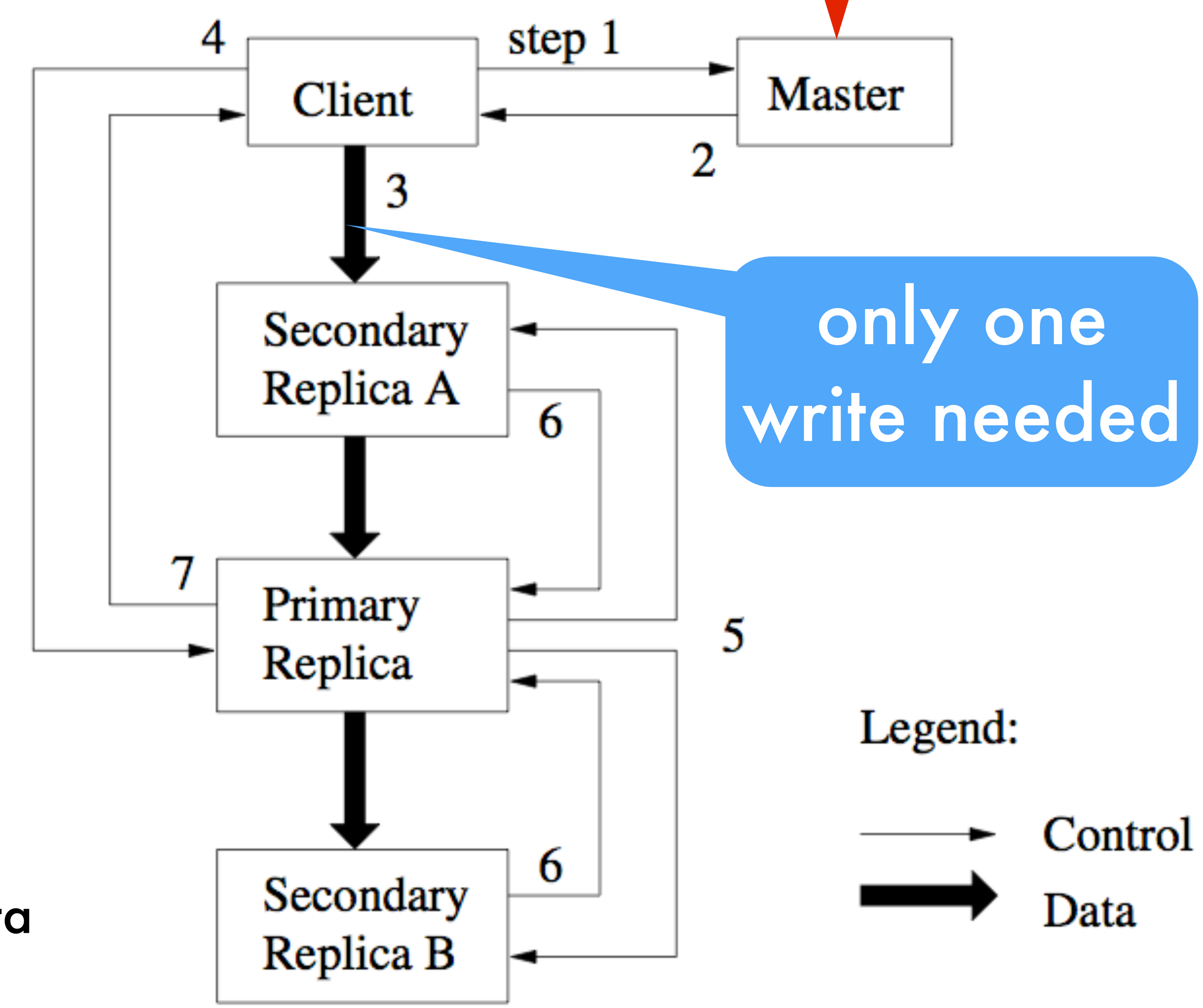


- Master ensures nodes are live
- Chunks are checksummed
- Can control replication factor for hotspots / load balancing
- Deserialize master state by loading data structure as flat file from disk (fast)

Google File System / HDFS

- Client requests chunk from master
- Master responds with replica location
- Client writes to replica A
- Client notifies primary replica
- Primary replica requests data from rep
- Replica A sends data to Primary replica
- Primary replica confirms write to clien

Achilles heel



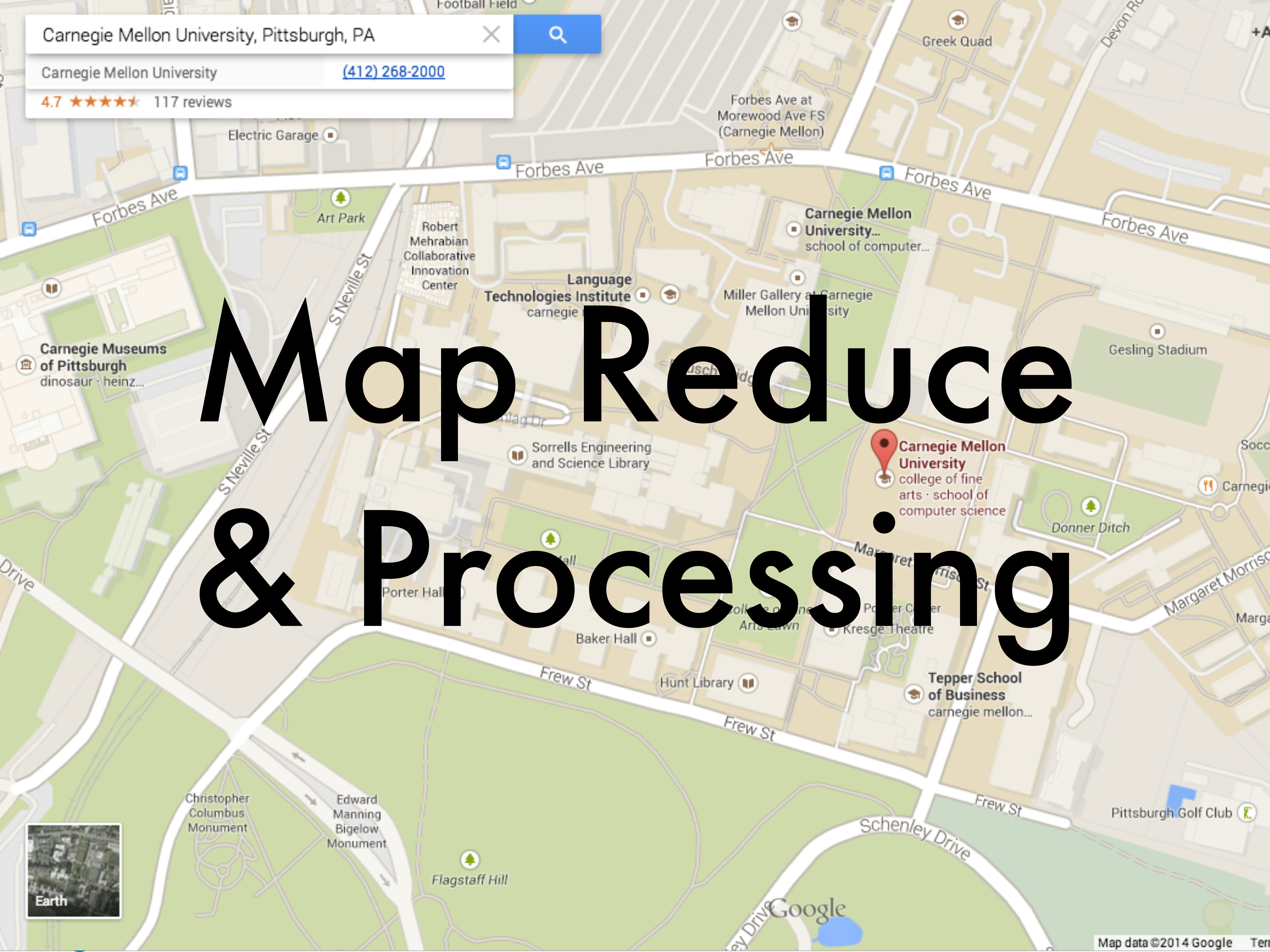
only one write needed

- Master ensures nodes are live
- Chunks are checksummed
- Can control replication factor for hotspots / load balancing
- Deserialize master state by loading data structure as flat file from disk (fast)

Google

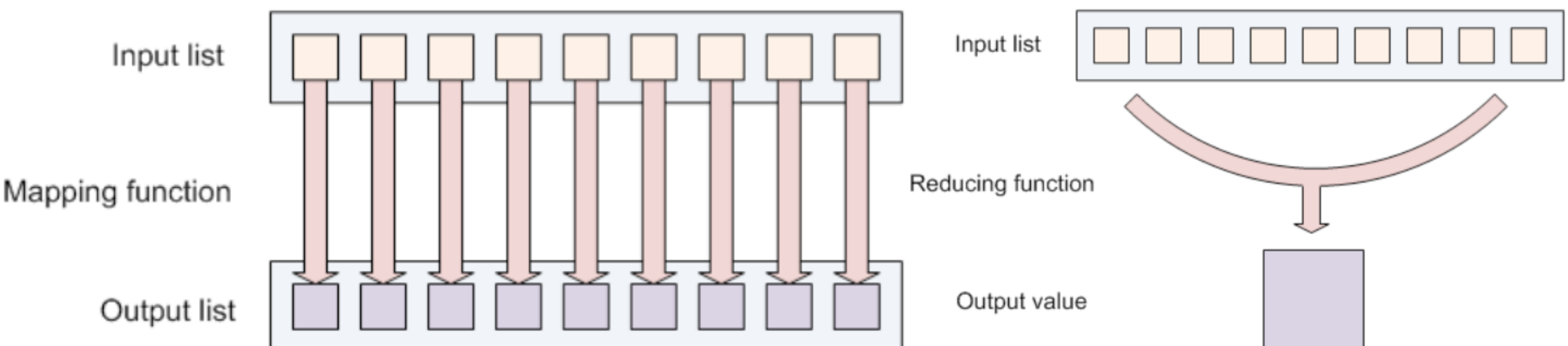
Carnegie Mellon University

Map Reduce & Processing



Map Reduce

- 1000s of (faulty) machines
- Lots of jobs are embarrassingly parallel (except for a sorting/transpose phase)
- Functional programming origins
  - Map(key,value) processes (key,value) pairs and outputs new (key,value) pair
  - Reduce(key,value) reduces all instances with same key to aggregate
- Example - (naive) wordcount
  - Map(docID, document) for each document emits many (wordID, count) pairs
  - Reduce(wordID, count) sums over all wordID, emits (wordID, aggregate)



from Ramakrishnan, Sakrejda, Canon, DoE 2011

Map Reduce

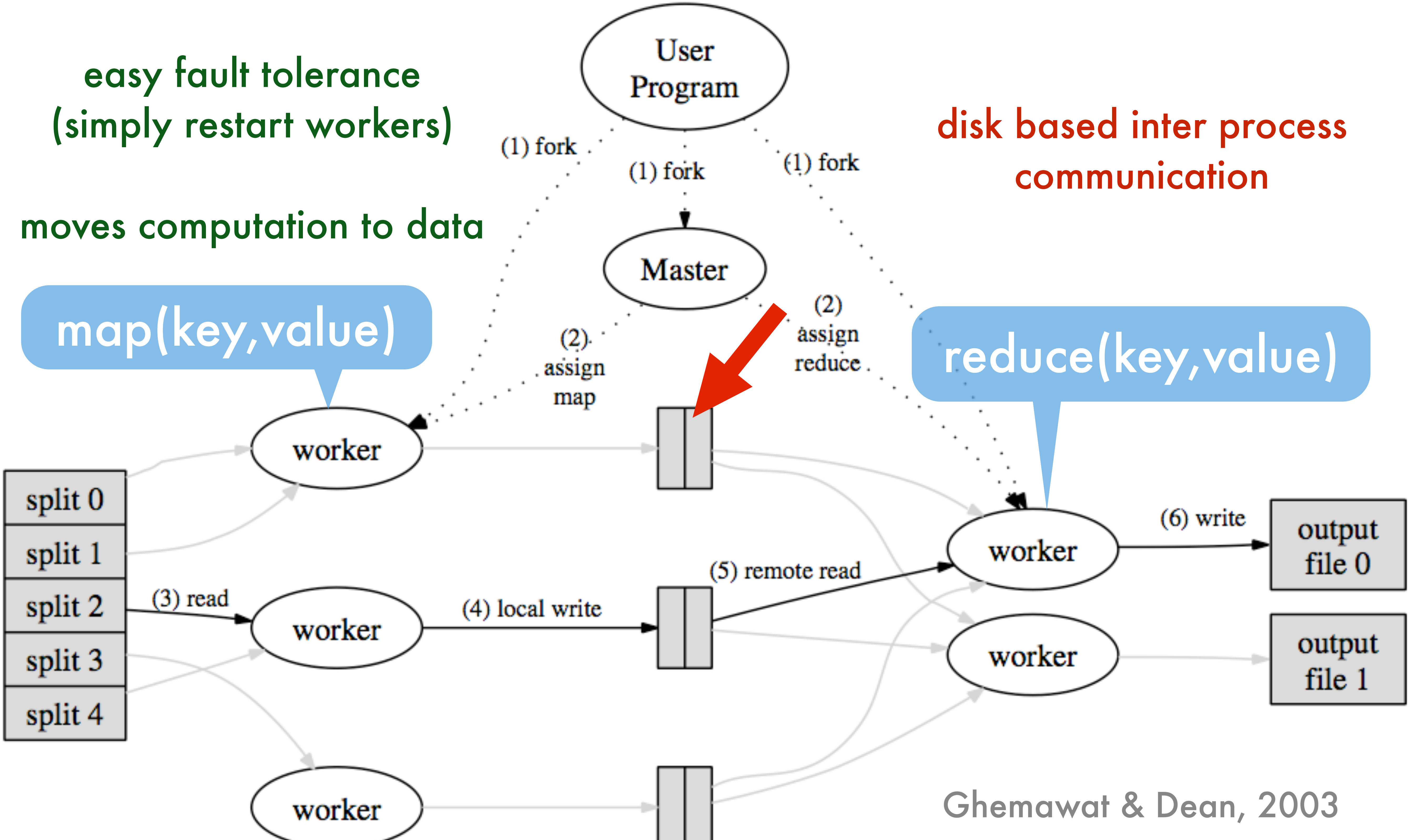
- 1000s of (faulty) machines
- Lots of jobs are embarrassingly parallel (except for a sorting/transpose phase)
- Functional programming origins
  - Map(key,value) processes (key,value) pairs and outputs new (key,value) pair
  - Reduce(key,value) reduces all instances with same key to aggregate
- Example - (naive) wordcount
  - Map(docID, document) for each document emits many (wordID, count) pairs
  - Reduce(wordID, count) sums over all wordID, emits (wordID, aggregate)

Map Reduce

easy fault tolerance
(simply restart workers)

moves computation to data

disk based inter process
communication



Ghemawat & Dean, 2003

Map Combine Reduce

- Combine aggregates keys within machine before sending to reducer
- Map must be stateless in blocks
- Reduce must be commutative in data
- Fault tolerance
  - Start jobs where the data is (move code not data)
  - Restart machines if maps fail (have replicas)
  - Restart reducers based on intermediate data
- Good fit for many algorithms
- Good if only a small number of MapReduce iterations needed
- Need to request machines at each iteration (time consuming)
- State lost in between maps
- Communication only via file I/O

Example - Gradient Descent

- Objective

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} l(x_i, y_i, w) + \frac{\lambda}{2} \|w\|^2$$

- Algorithm
  - compute gradient

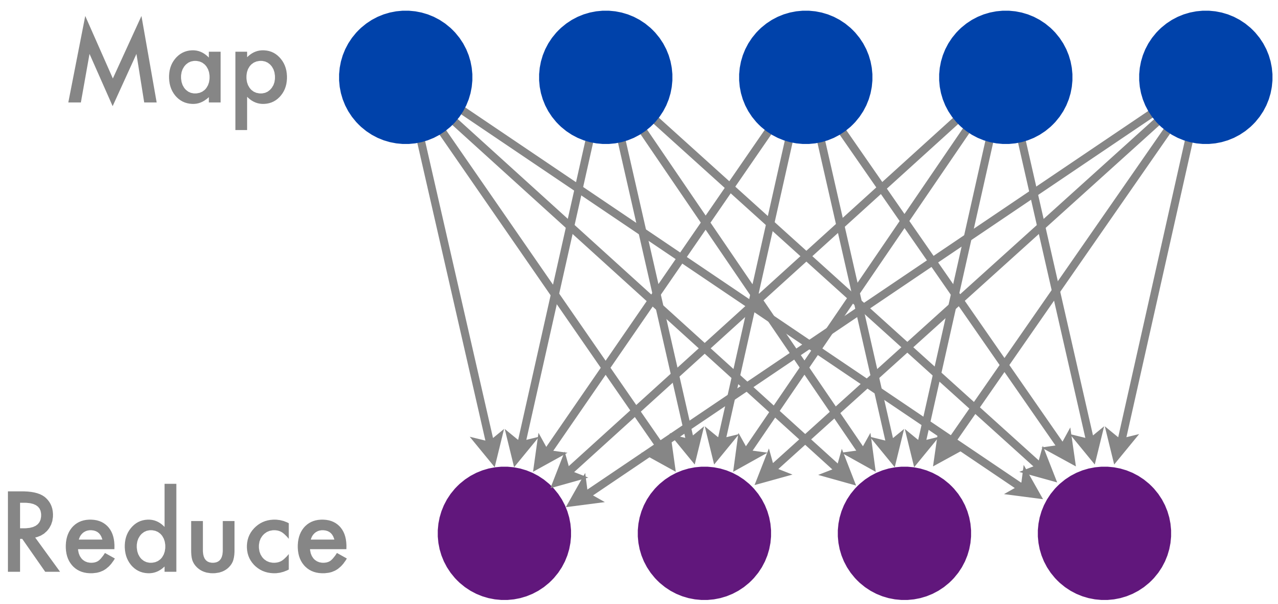
$$g := \sum_{i=1}^{m} \partial_w l(x_i, y_i, w)$$

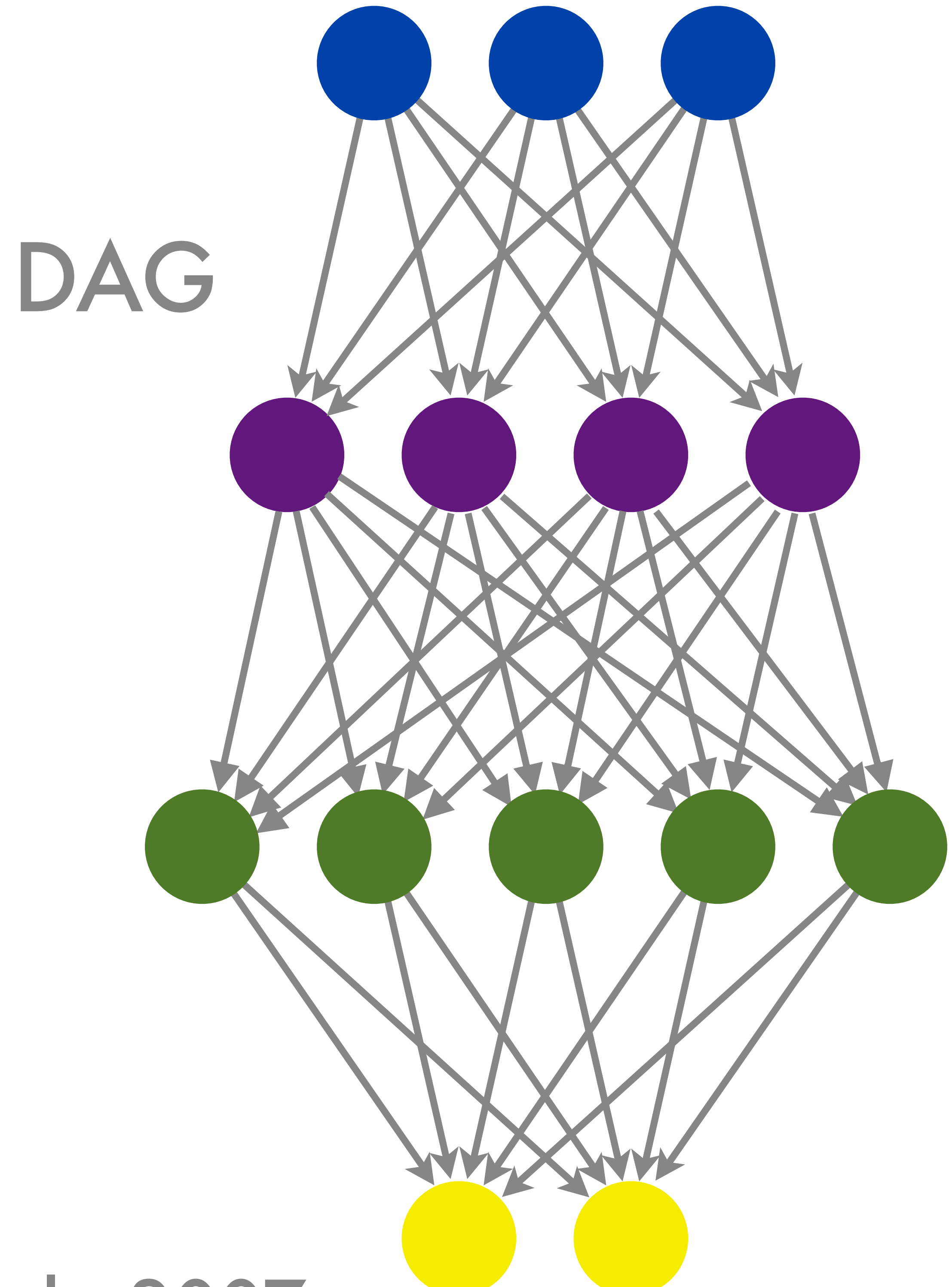
    - On each data point via Map(i,data)
    - Sum gradient via Reduce(coordinate)
  - perform update step
    (much better with line search)

$$w \leftarrow w - \eta(g + \lambda w)$$

  - repeat

Dryad



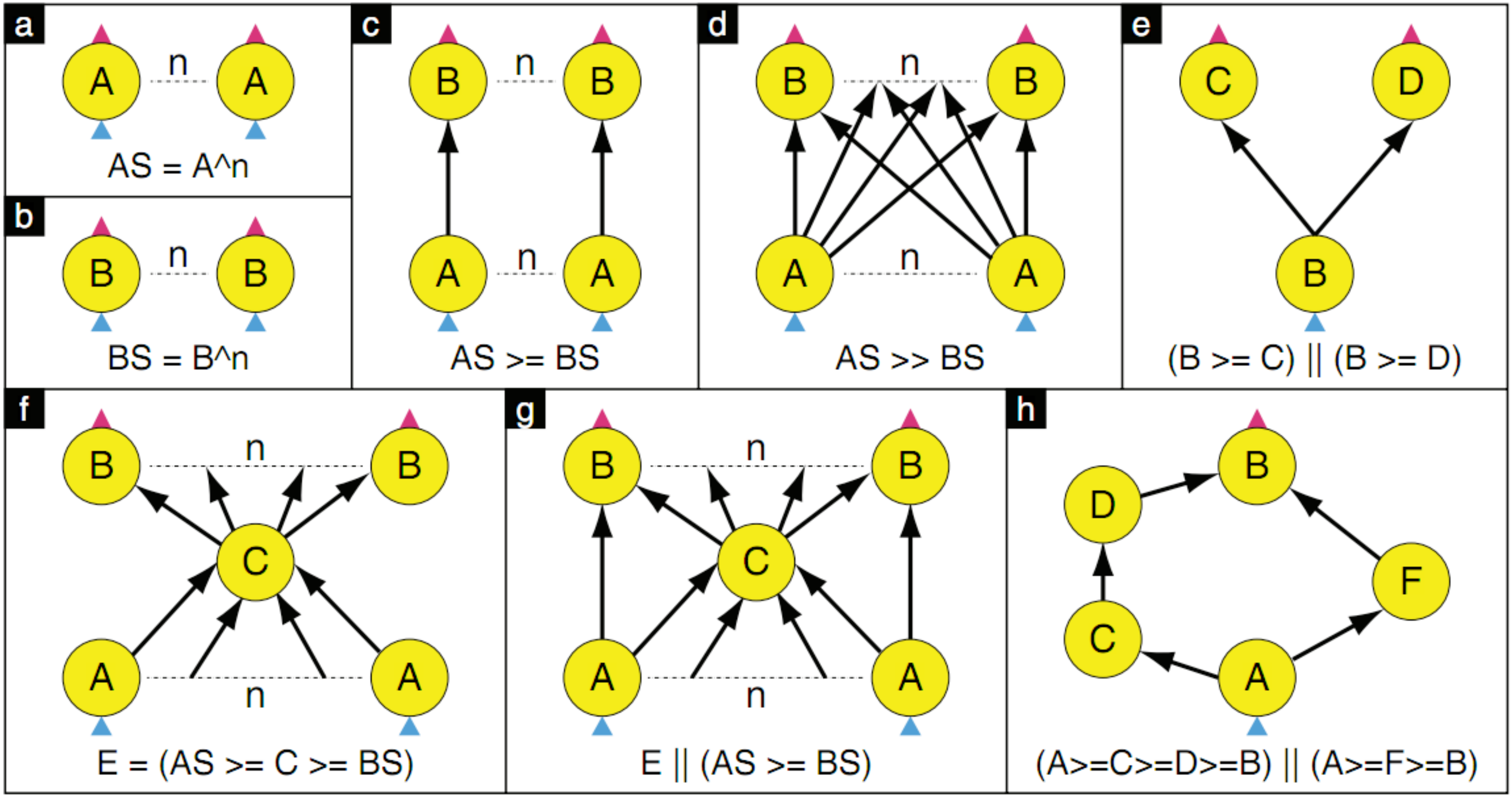


- Directed acyclic graph
- System optimizes parallelism
- Different types of IPC (memory FIFO/network/file)
- Tight integration with .NET (allows easy prototyping)

Isard et al., 2007

Google Carnegie Mellon University

DRYAD

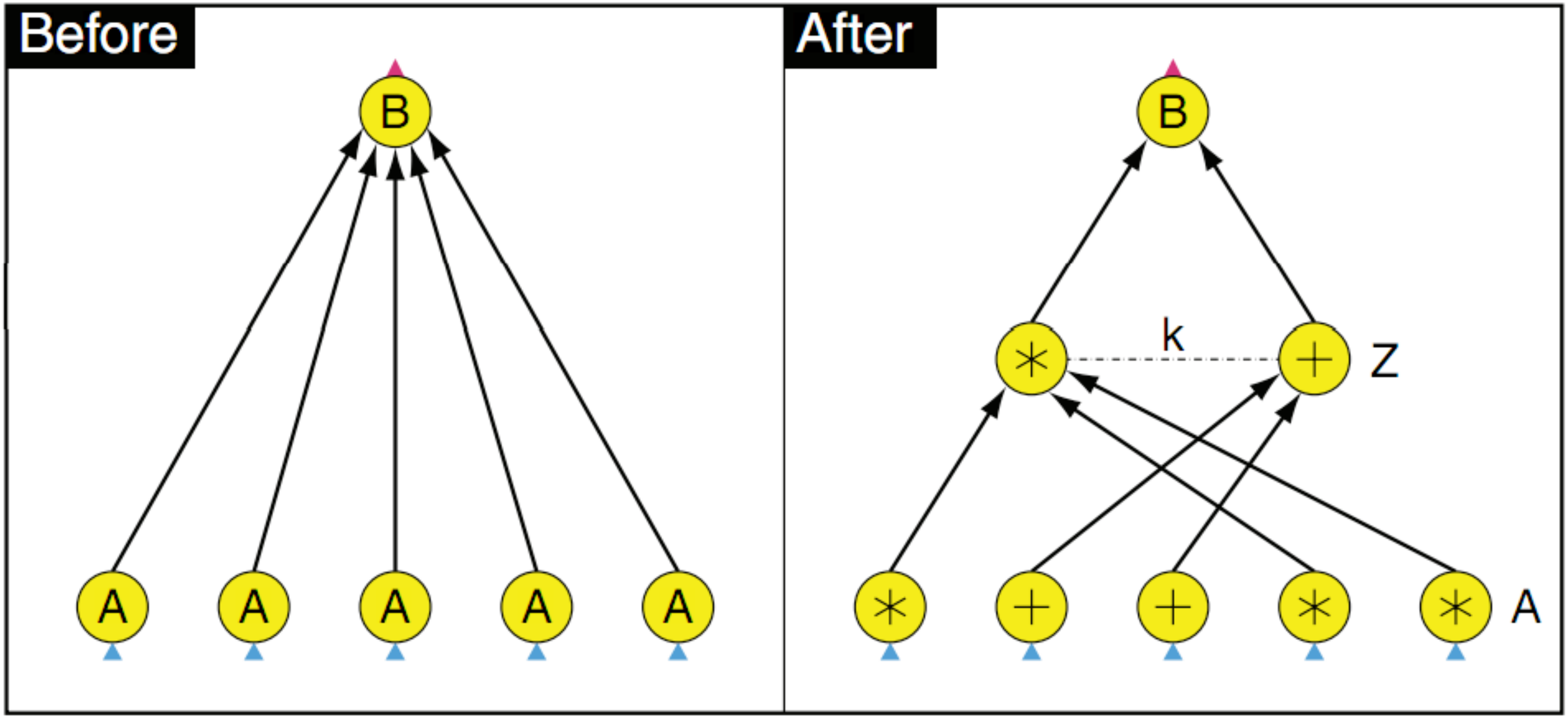


Google

graph description language

Carnegie Mellon University

DRYAD

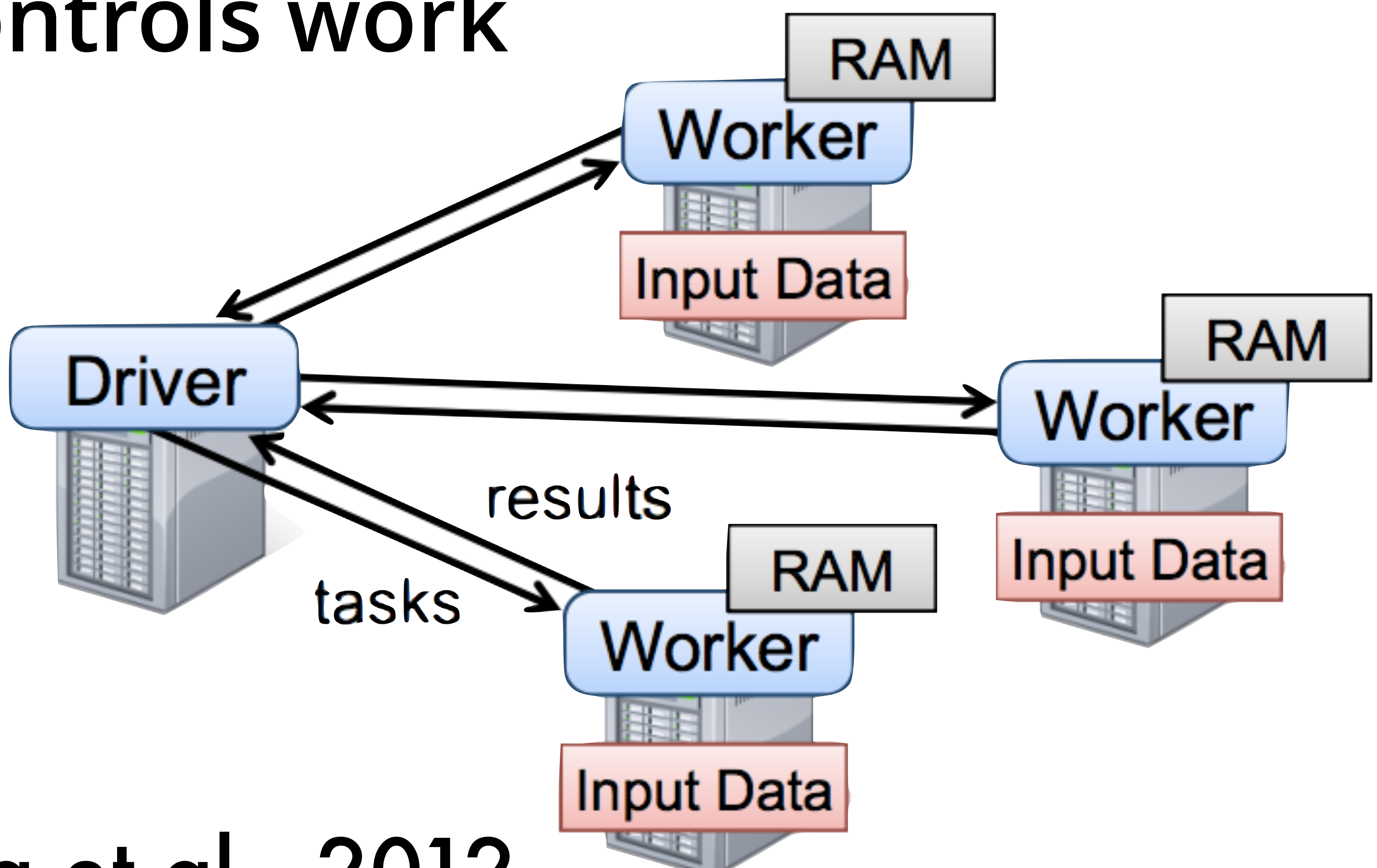


automatic graph refinement


Spark
SPARK
MICHIGAN
028M179
MANUFACTURER

Resilient Distributed Datasets

- Data is transformed by processing
- Store intermediate data using lineage
- Driver controls work



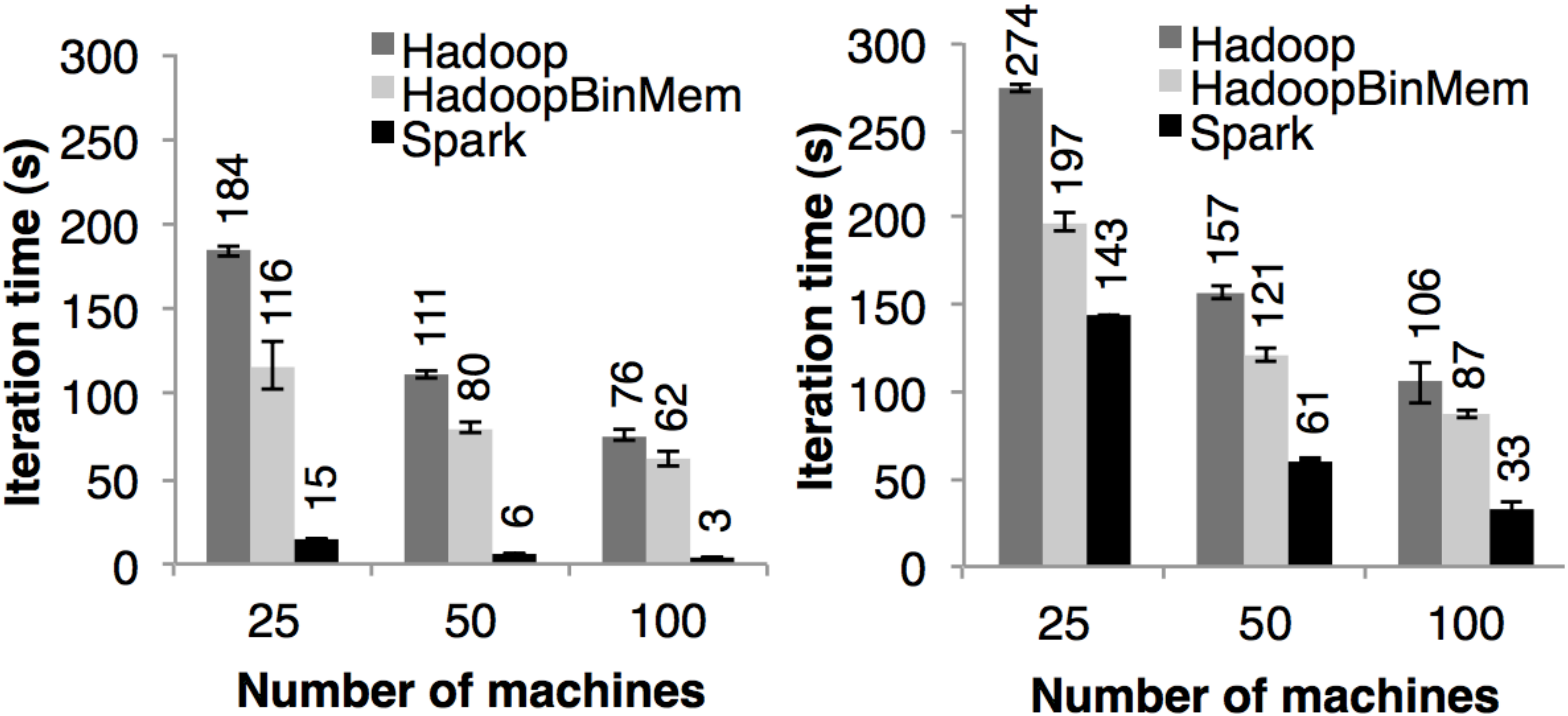
Zaharia et al., 2012

Beyond MapReduce

Transformations	$map(f : \mathrm{T} \Rightarrow \mathrm{U})$	:	RDD[T] $\Rightarrow$ RDD[U]
	$filter(f : \mathrm{T} \Rightarrow \mathrm{Bool})$	:	RDD[T] $\Rightarrow$ RDD[T]
	$flatMap(f : \mathrm{T} \Rightarrow \mathrm{Seq[U]})$	:	RDD[T] $\Rightarrow$ RDD[U]
	$sample(fraction : \mathrm{Float})$	:	RDD[T] $\Rightarrow$ RDD[T] (Deterministic sampling)
	$groupByKey()$	:	RDD[(K, V)] $\Rightarrow$ RDD[(K, Seq[V])]
	$reduceByKey(f : (\mathrm{V}, \mathrm{V}) \Rightarrow \mathrm{V})$	:	RDD[(K, V)] $\Rightarrow$ RDD[(K, V)]
	$union()$	:	(RDD[T], RDD[T]) $\Rightarrow$ RDD[T]
	$join()$	:	(RDD[(K, V)], RDD[(K, W)]) $\Rightarrow$ RDD[(K, (V, W))]
	$cogroup()$	:	(RDD[(K, V)], RDD[(K, W)]) $\Rightarrow$ RDD[(K, (Seq[V], Seq[W]))]
	$crossProduct()$	:	(RDD[T], RDD[U]) $\Rightarrow$ RDD[(T, U)]
	$mapValues(f : \mathrm{V} \Rightarrow \mathrm{W})$	:	RDD[(K, V)] $\Rightarrow$ RDD[(K, W)] (Preserves partitioning)
	$sort(c : \mathrm{Comparator[K]})$	:	RDD[(K, V)] $\Rightarrow$ RDD[(K, V)]
	$partitionBy(p : \mathrm{Partitioner[K]})$	:	RDD[(K, V)] $\Rightarrow$ RDD[(K, V)]
Actions	$count()$	:	RDD[T] $\Rightarrow$ Long
	$collect()$	:	RDD[T] $\Rightarrow$ Seq[T]
	$reduce(f : (\mathrm{T}, \mathrm{T}) \Rightarrow \mathrm{T})$	:	RDD[T] $\Rightarrow$ T
	$lookup(k : \mathrm{K})$	:	RDD[(K, V)] $\Rightarrow$ Seq[V] (On hash/range partitioned RDDs)
	$save(path : \mathrm{String})$	:	Outputs RDD to a storage system, *e.g.*, HDFS

rich language & preprocessor

Improvement over MapReduce

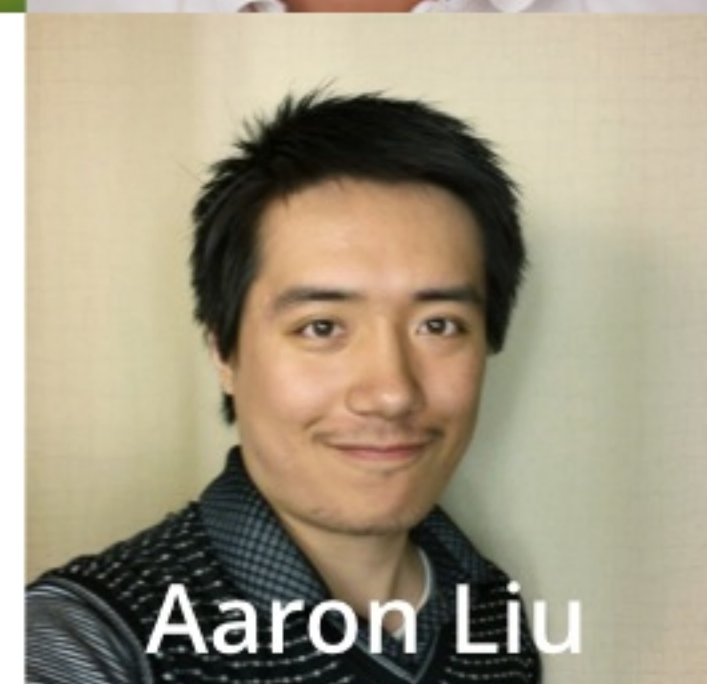


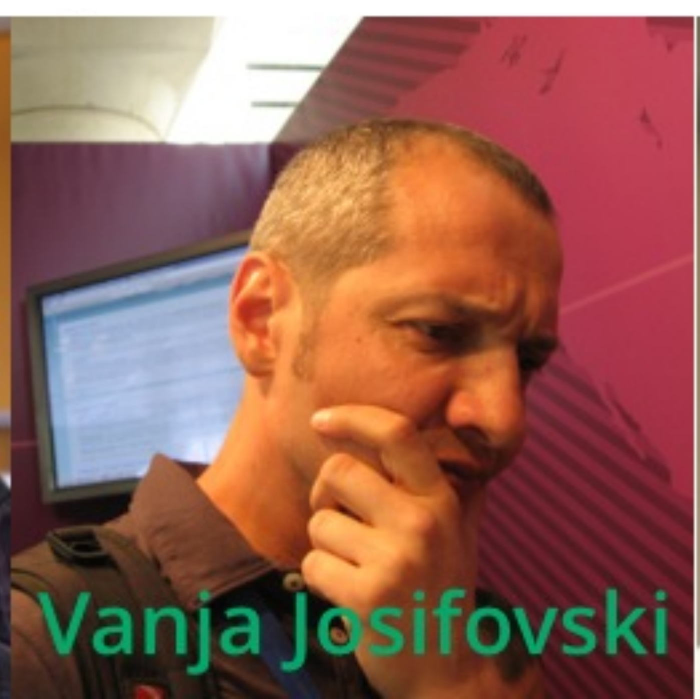
(a) Logistic Regression (b) K-Means

Google Carnegie Mellon University

parameterserver.org

blog.smola.org @smolix






Background

The Challenge

- Scale
  - 100s Terabytes of data
  - 1000s of computers
  - 100 Billions of parameters
- Reality
  - Faulty machines
  - Shared cluster
- Performance
  - Front end serving machines
  - Real time response

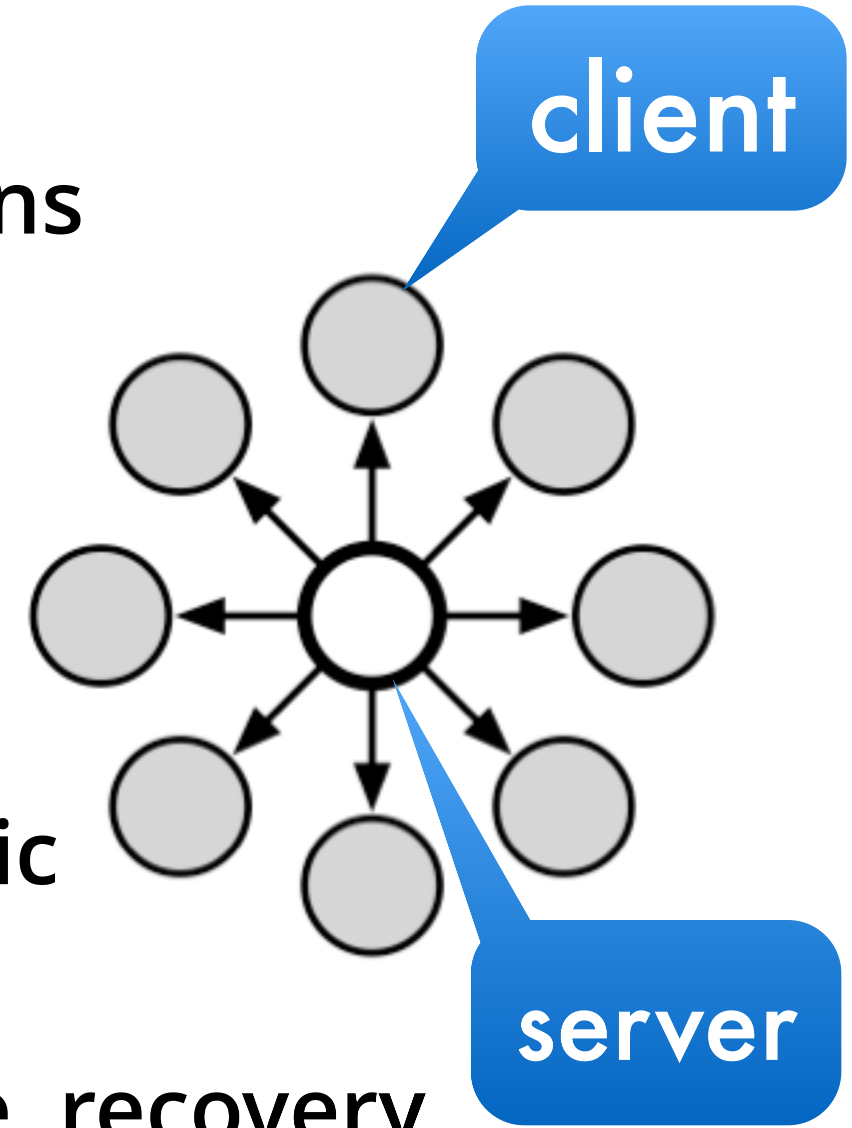
Machine Learning Problems

- Many models have O(1) blocks of O(n) terms (LDA, logistic regression, recommender systems)
- More terms than what fits into RAM (personalized CTR, large inventory, action space)
- Local model typically fits into RAM
- Data needs many disks for distribution
- Decouple data processing from aggregation

- Optimize for the 80% of all ML problems

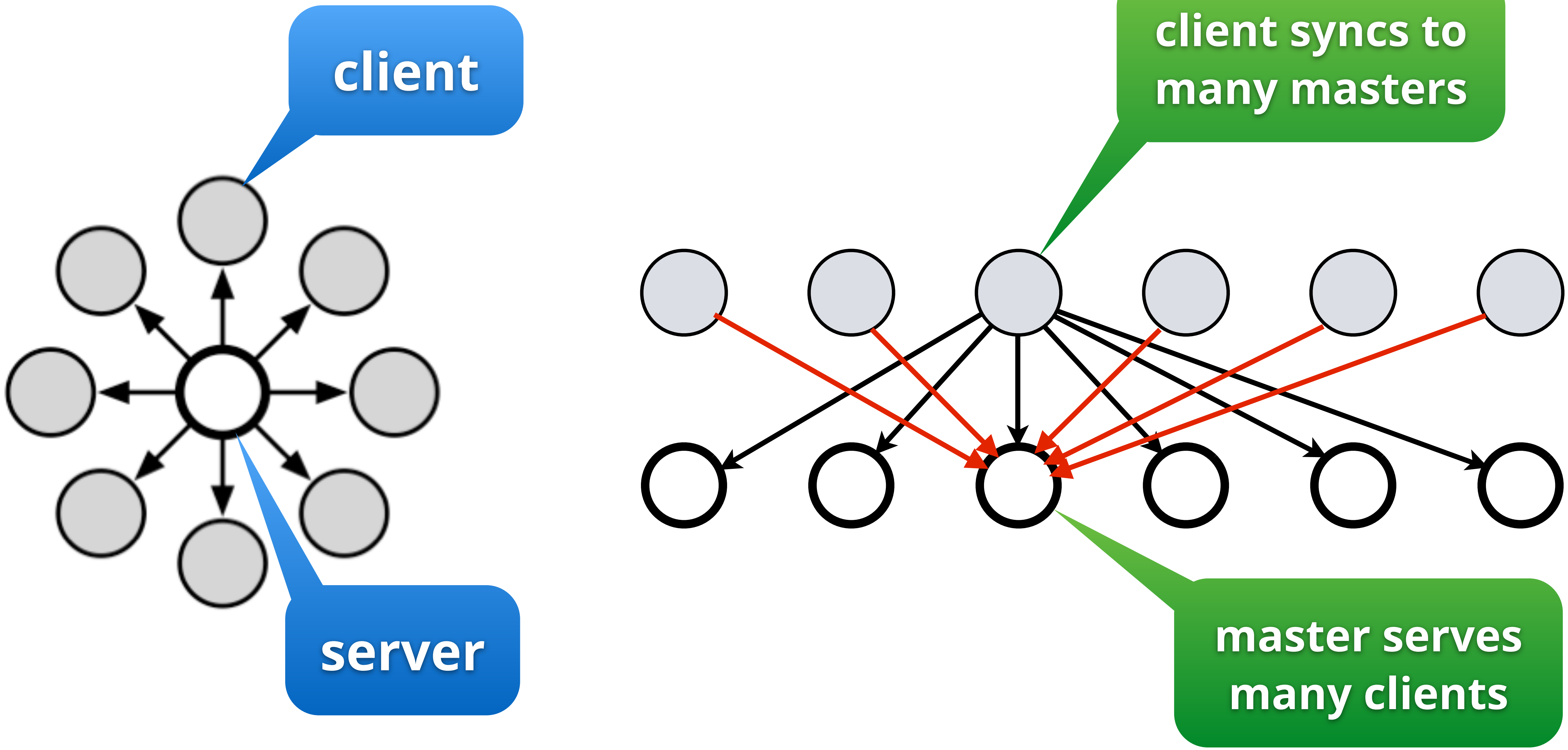
General parallel algorithm template

- Clients have local view of parameters
- P2P is infeasible since $O(n^2)$ connections
- Synchronize with parameter server
  - Reconciliation protocol
    average parameters, lock variables
  - Synchronization schedule
    asynchronous, synchronous, episodic
  - Load distribution algorithm
    uniform distribution, fault tolerance, recovery



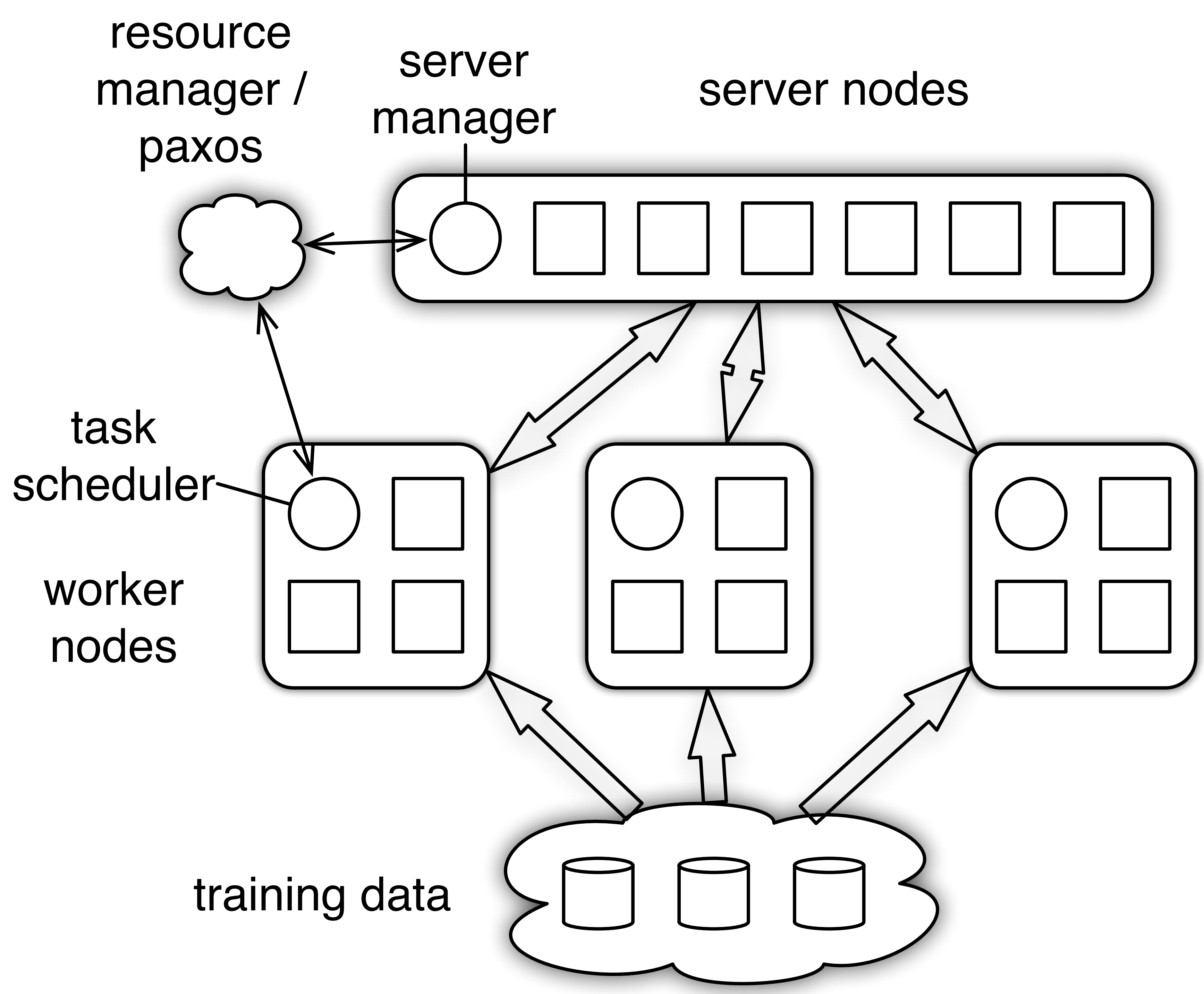
Smola & Narayanamurthy, 2010, VLDB
Gonzalez et al., 2012, WSDM
Shervashidze et al., 2013, WWW

Communication pattern



put(keys,values,clock), get(keys,values,clock)

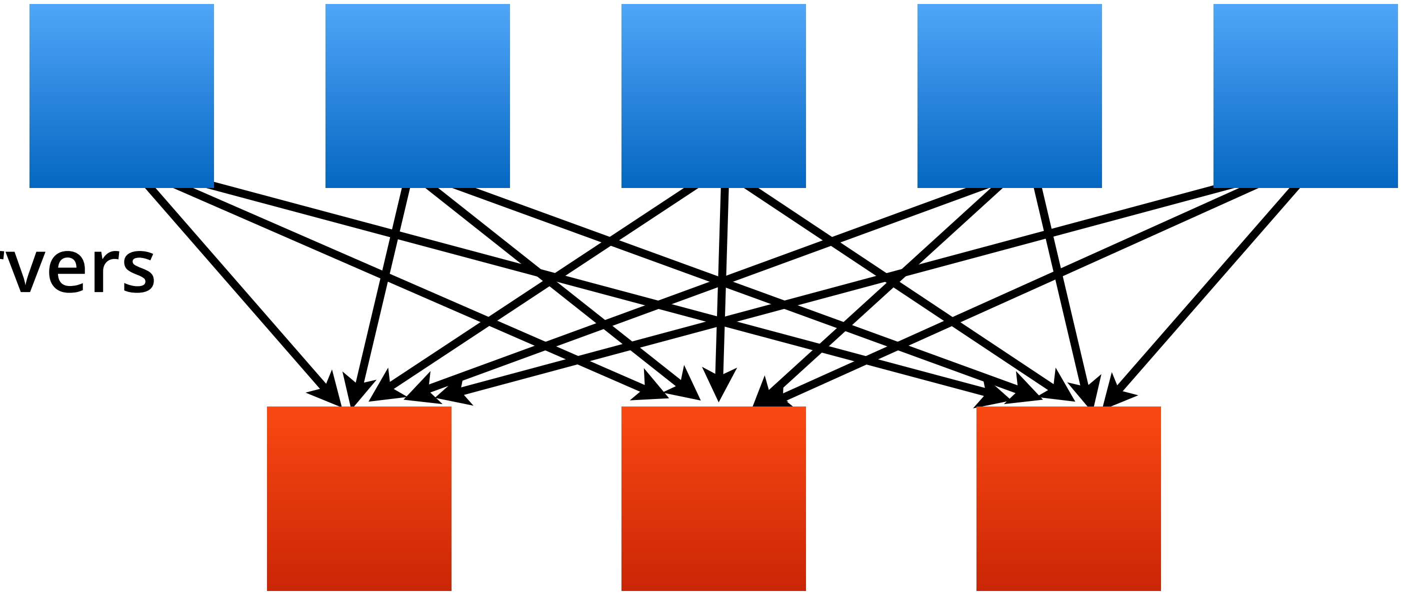
Architecture




Key layout & recovery

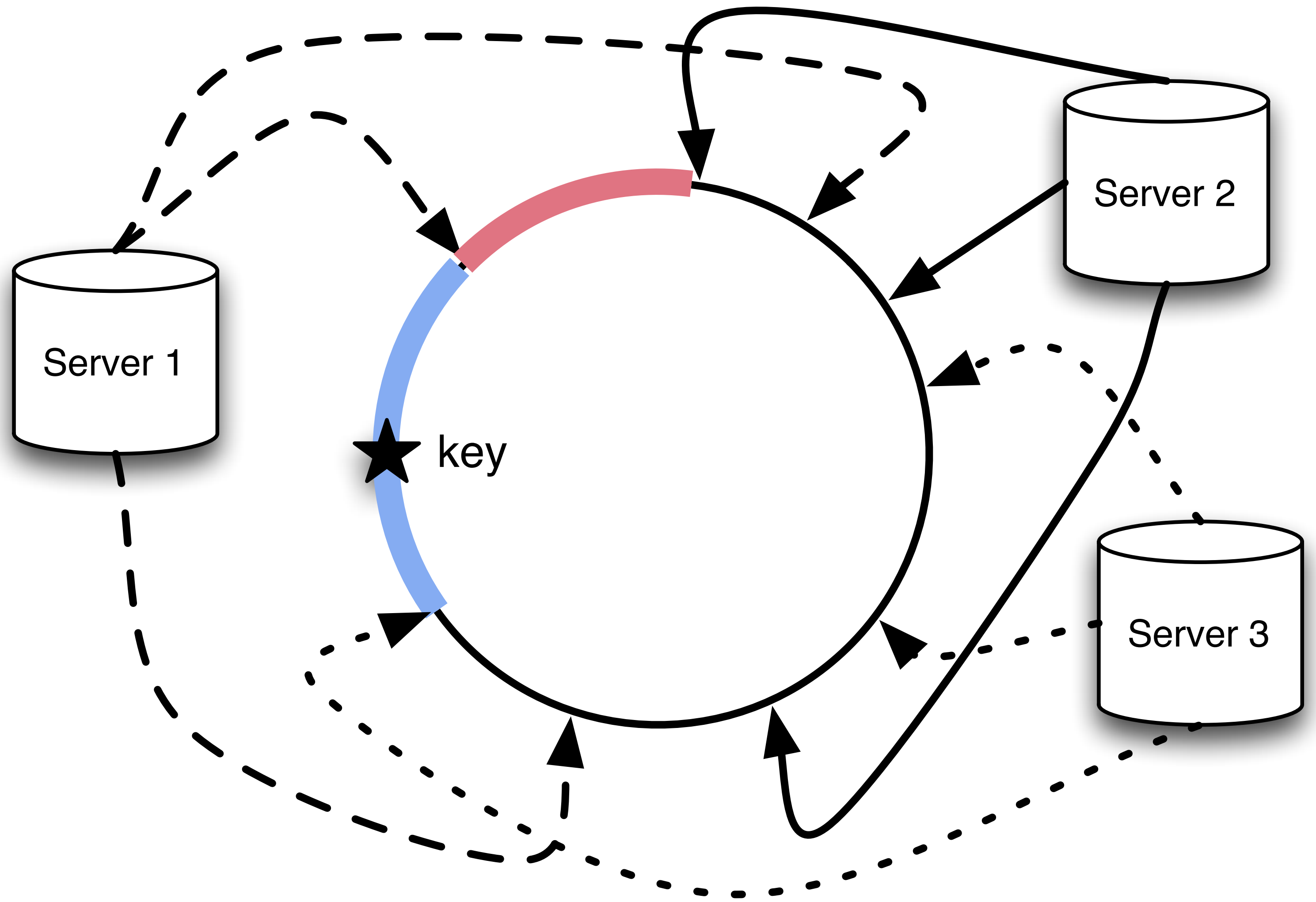
Consistent Hashing

- Caching
  - Store many (key,value) pairs
  - Linear scaling in clients & servers
  - Automatic key distribution
- memcached
  - (key,value) servers
  - client access library distributes access patterns
  - randomized O(n) bandwidth
  - aggregate O(n) bandwidth
  - load balancing via hashing
  - no versioned writes / vector clocks
  - very expensive to iterate over all keys for a given server



$$m(\text{key}, \mathcal{M}) = \operatorname*{argmin}_{m' \in \mathcal{M}} h(\text{key}, m')$$

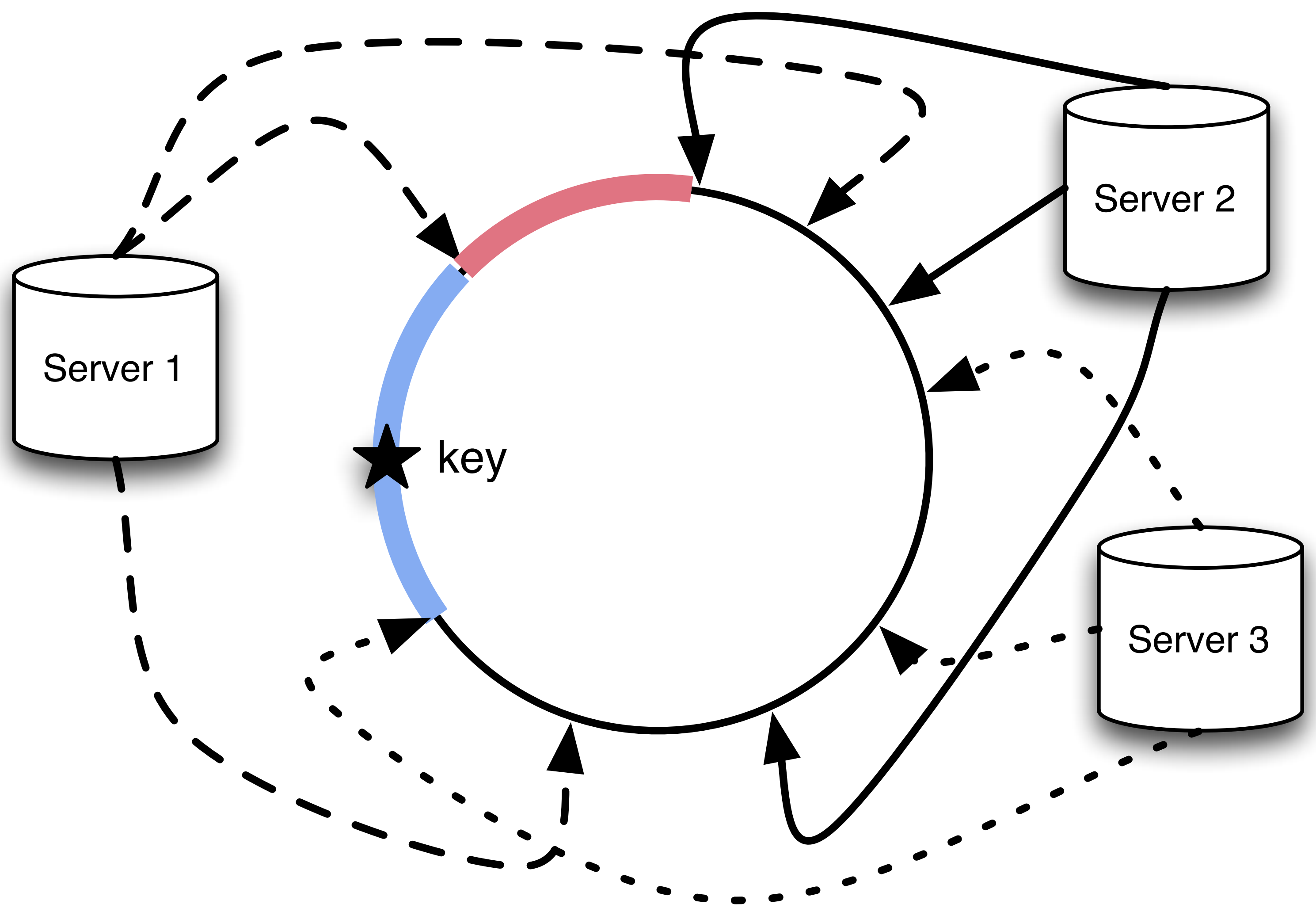
Keys arranged in a DHT



- Virtual servers
  - loadbalancing
  - multithreading
- DHT
  - contiguous key range for clients
  - easy bulk sync
  - easy insertion of servers
- Replication
  - Machines hold replicas
  - Easy fallback
  - Easy insertion / repair

Carnegie Mellon University

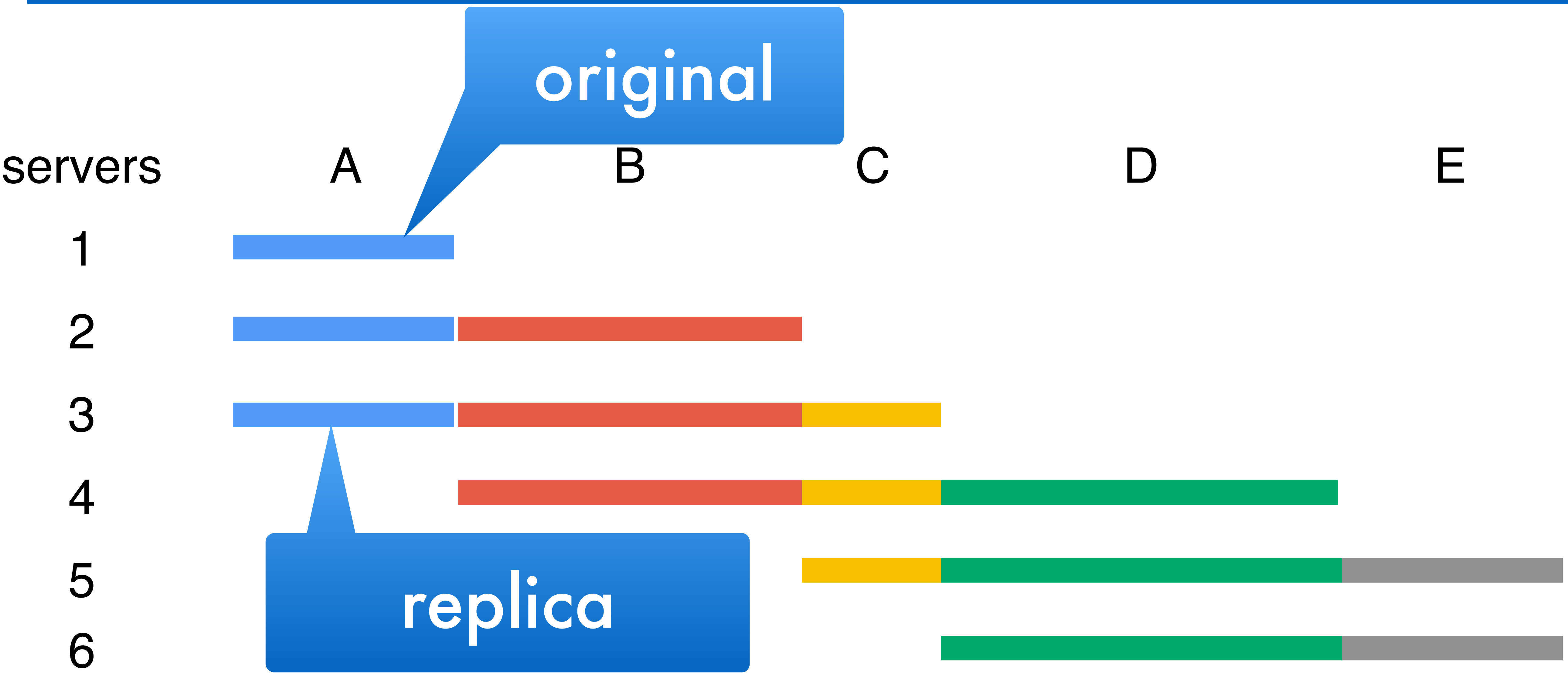
Keys arranged in a DHT



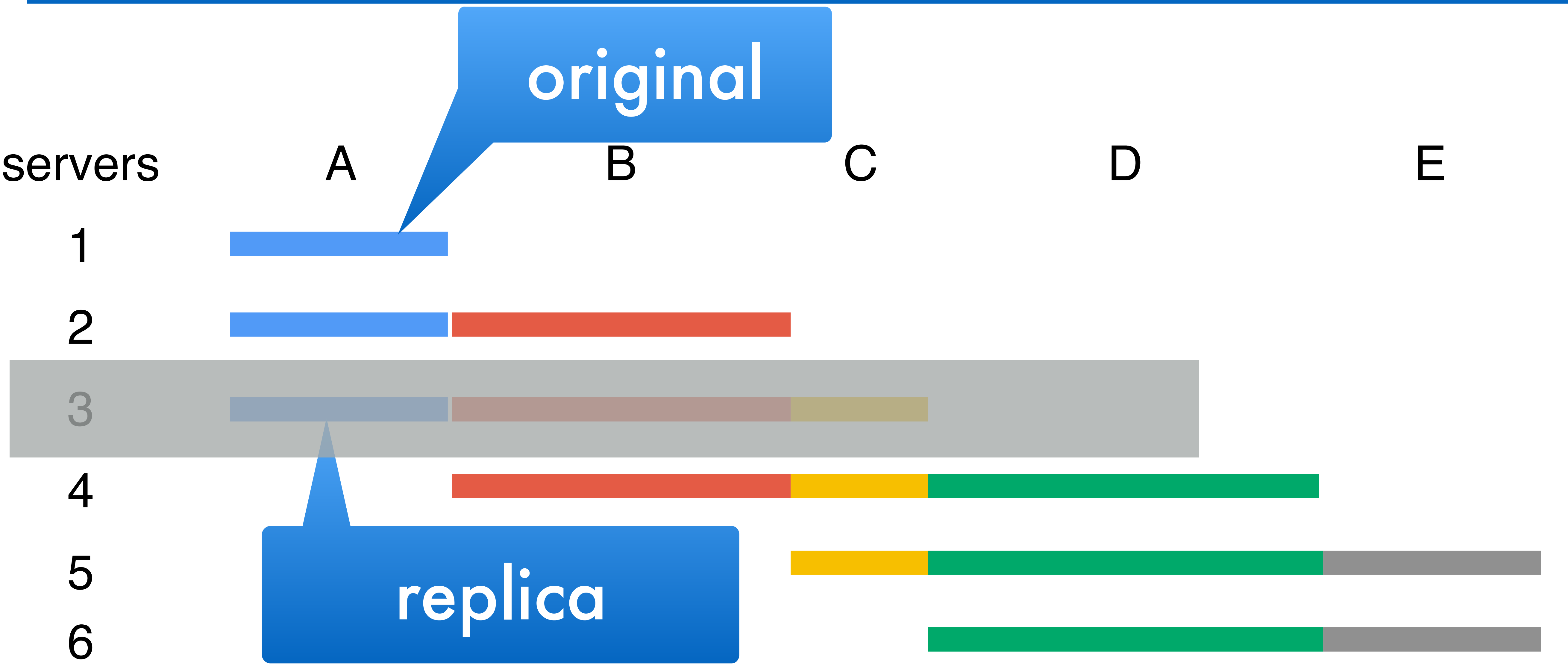
Yes, we screwed up before!
And everyone copied us!

- Virtual servers
  - loadbalancing
  - multithreading
- DHT
  - contiguous key range for clients
  - easy bulk sync
  - easy insertion of servers
- Replication
  - Machines hold replicas
  - Easy fallback
  - Easy insertion / repair

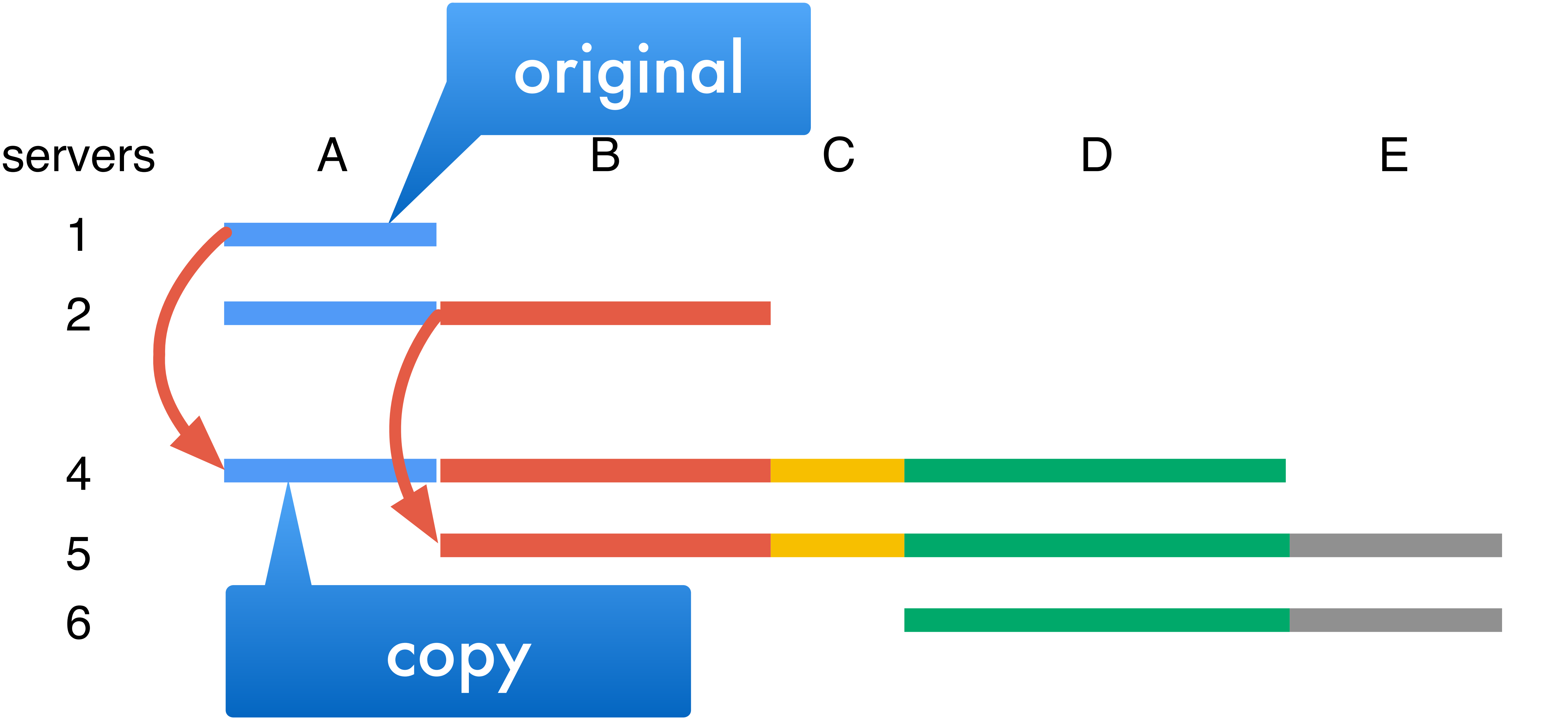
Key layout



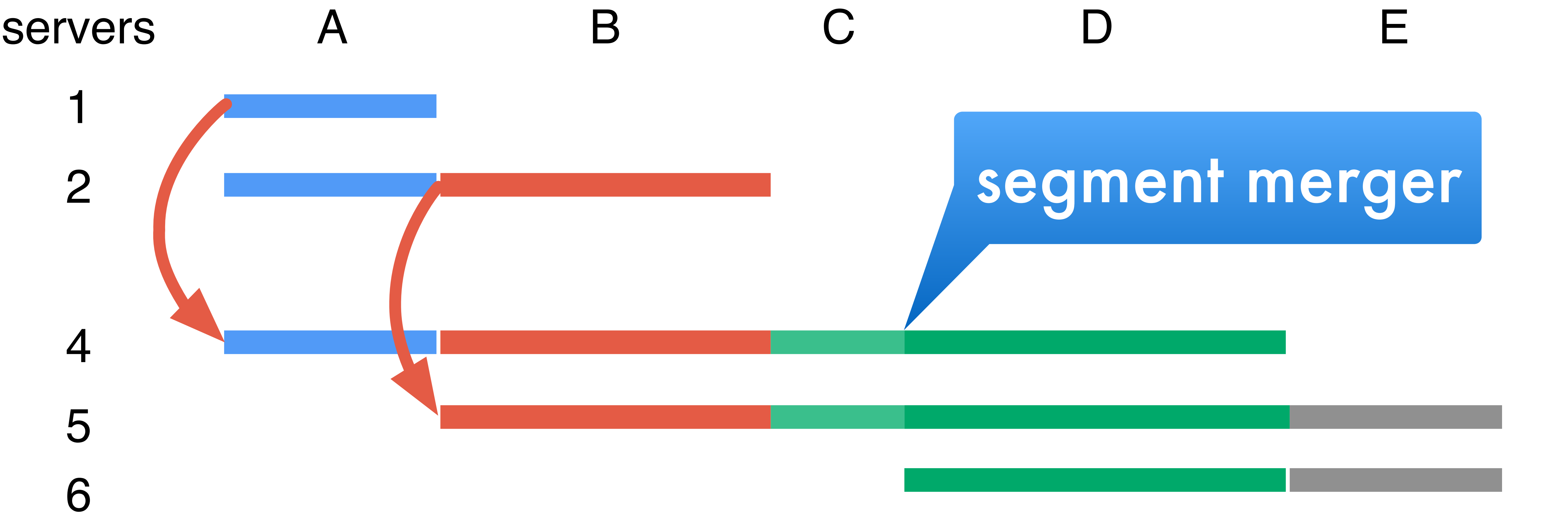
Key layout



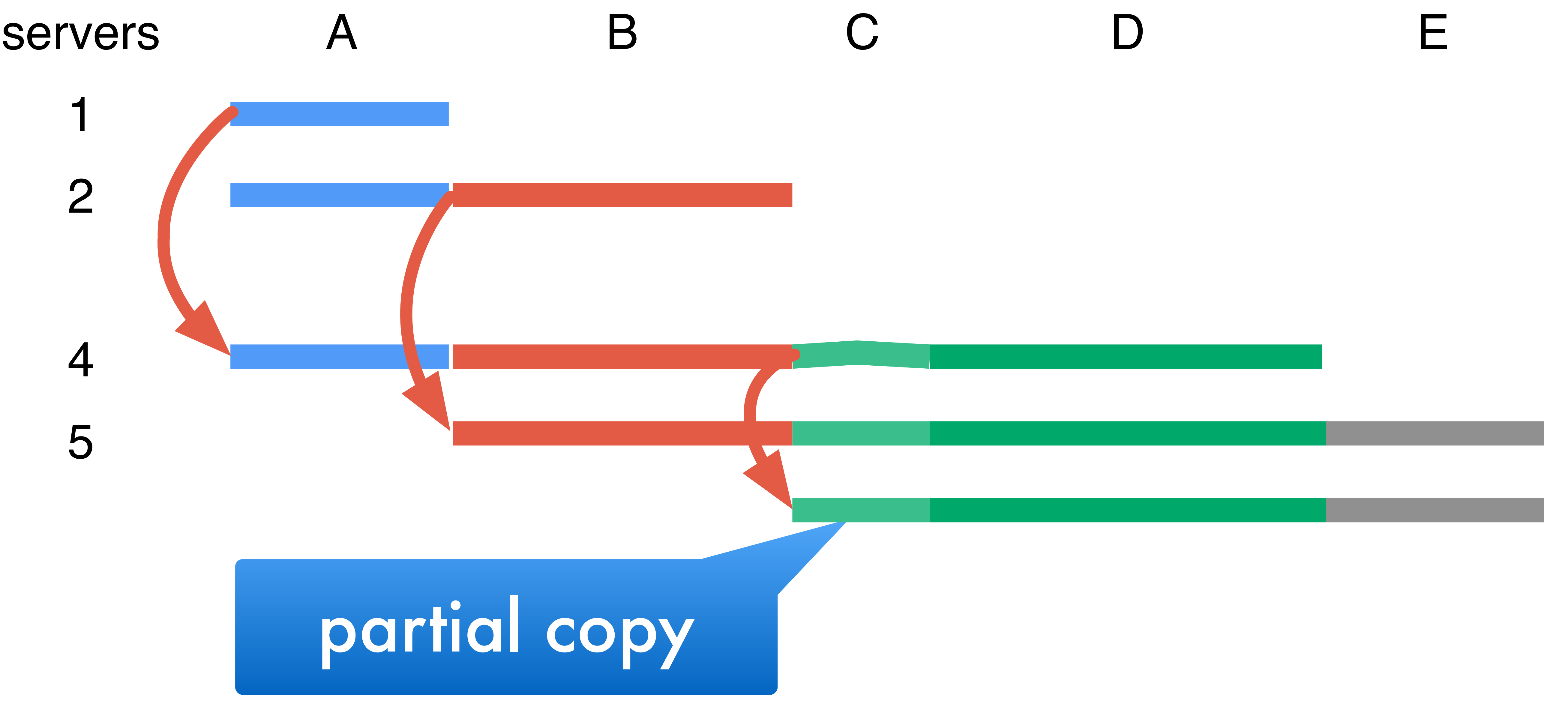
Key layout



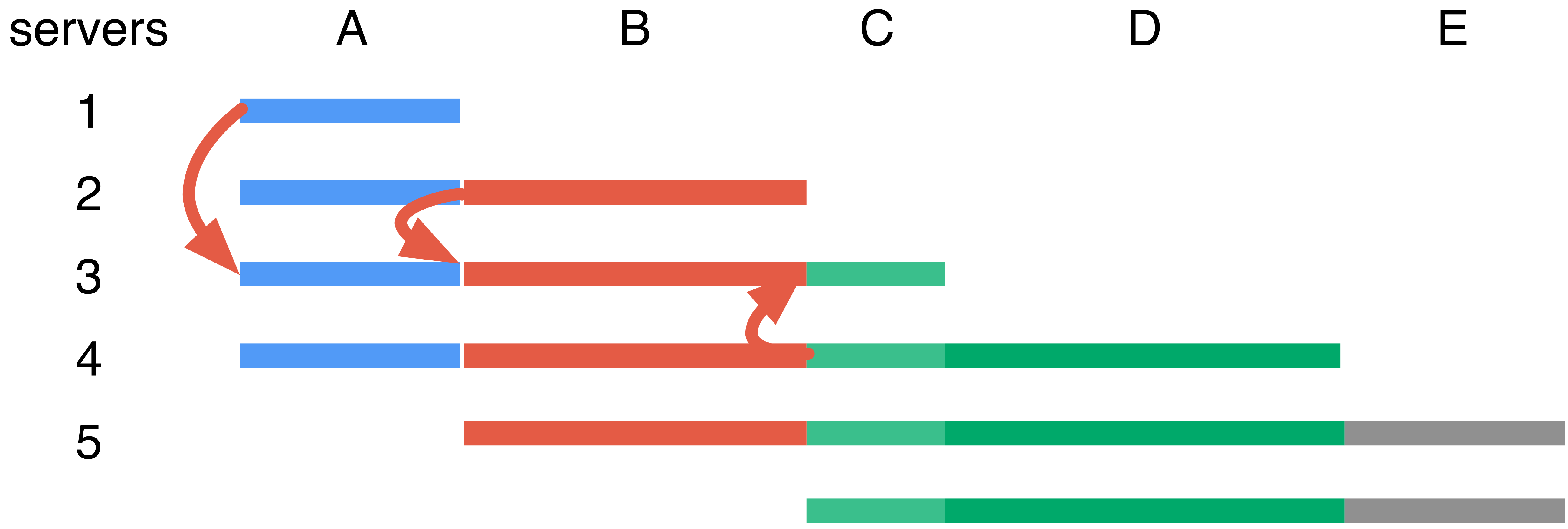
Key layout



Key layout



Recovery / server insertion



- Precopy server content to new candidate (3)
- After precopy ended, send log
- For k virtual servers this causes $O(k^{-2})$ delay
- Consistency using vector clocks

Communication

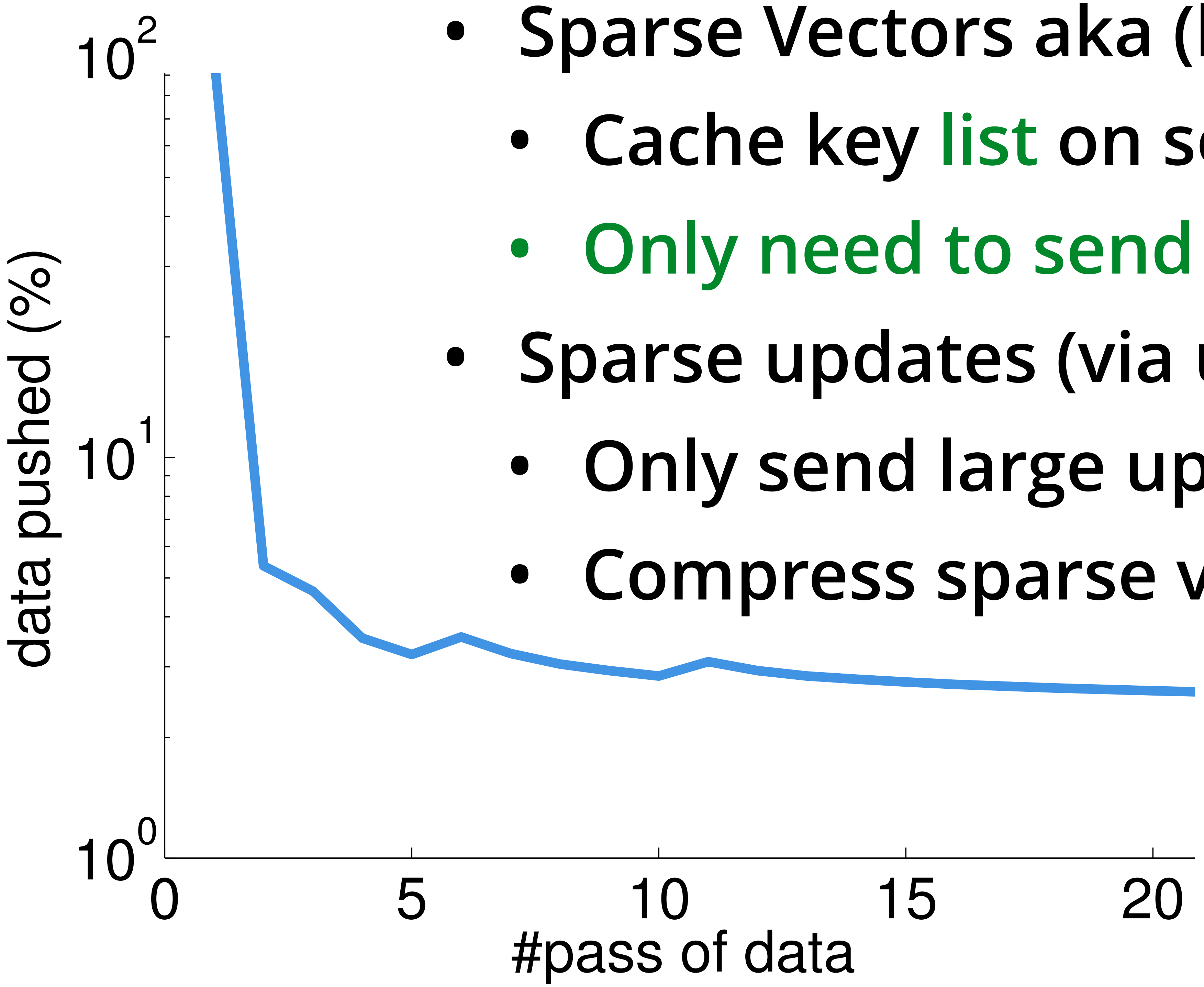
Message Compression

- Convergence speed depends on communication efficiency
    - Sending (key,value) pairs is inefficient
      Send only values (cache key list) instead
    - Sending small gradients is inefficient
      Send only sufficiently large ones instead
    - Updating near-optimal values is inefficient
      Send only large violators of KKT conditions
- Filter data before sending

Filters

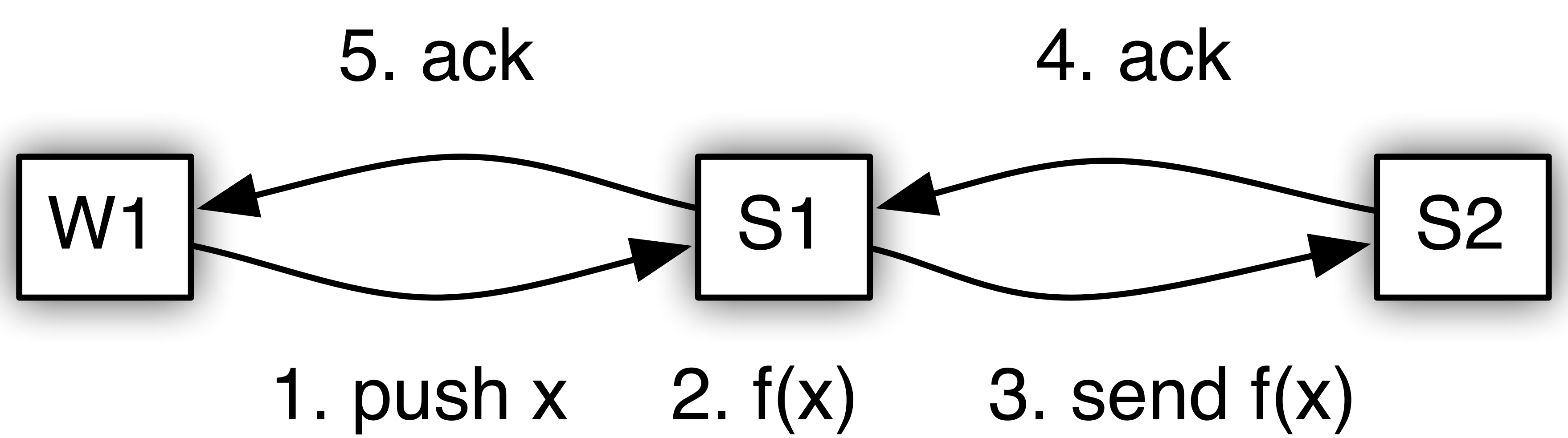
- Scheduling
  have controller decide when to send
  (this requires very smart controller)
- Filtering
  have algorithm decide when to shut up
  - Gradient (only send large gradients)
  - KKT (only send variables violating KKT)
  - Randomized (sparse random vectors)
  - Quantization (reduce accuracy)

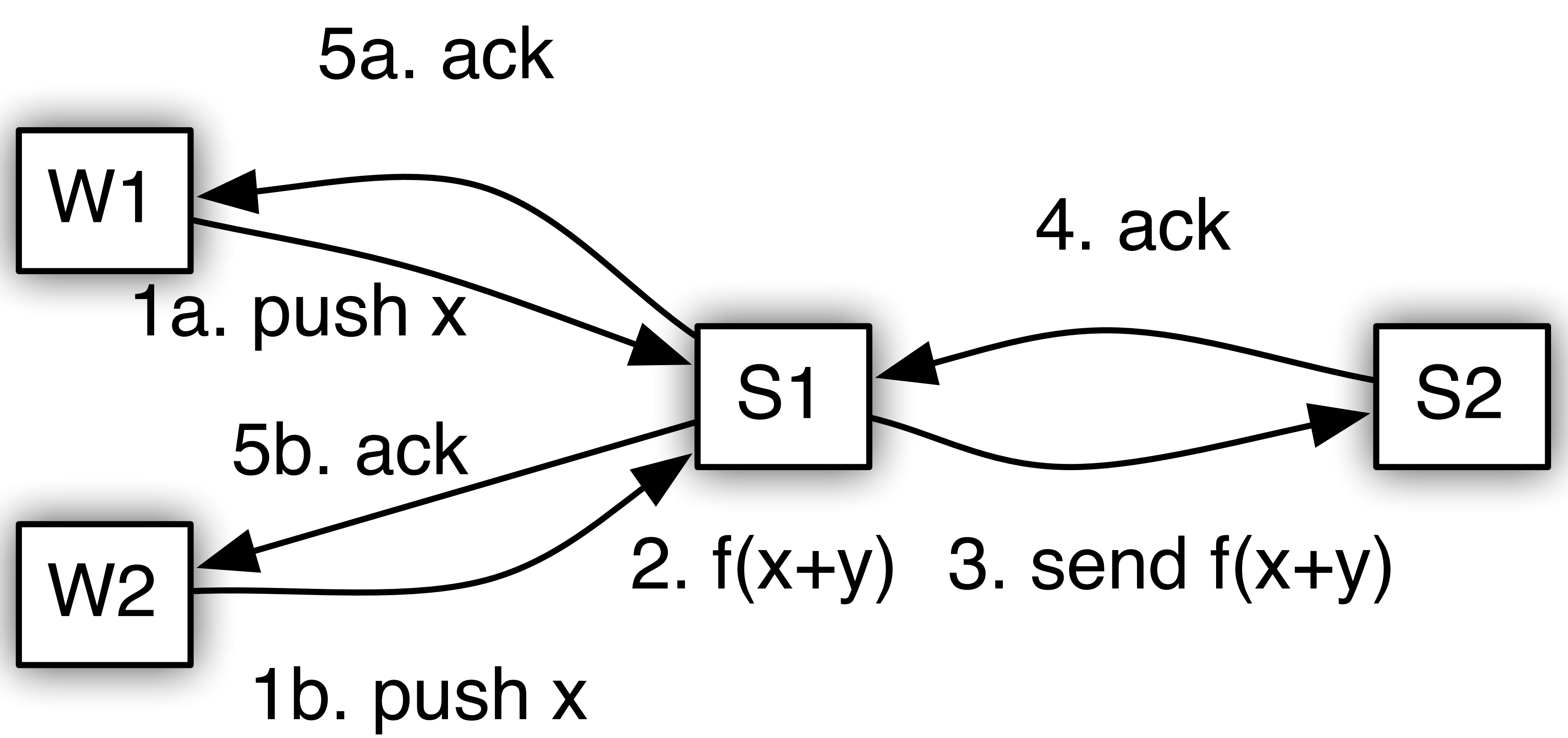
Message Compression



- Sparse Vectors aka (key,value) pairs
  - Cache key list on server
  - Only need to send **values**
- Sparse updates (via user defined filter)
  - Only send large updates
  - Compress sparse value list

Message Aggregation on Server





Messaging

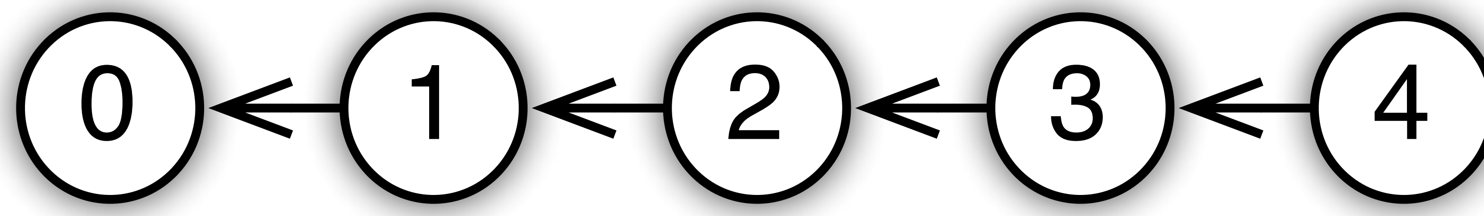
- Datatypes are `eigen3` native
  - Dense vectors
  - Sparse vectors
- `Push(Header flag)`
- `Pull(Header flag)`

  `Flag` may specify
  - Value or delta update
  - key range
  - recipient (all server, all clients, particular node)

Shared pointer. No copy on queue (by default)!

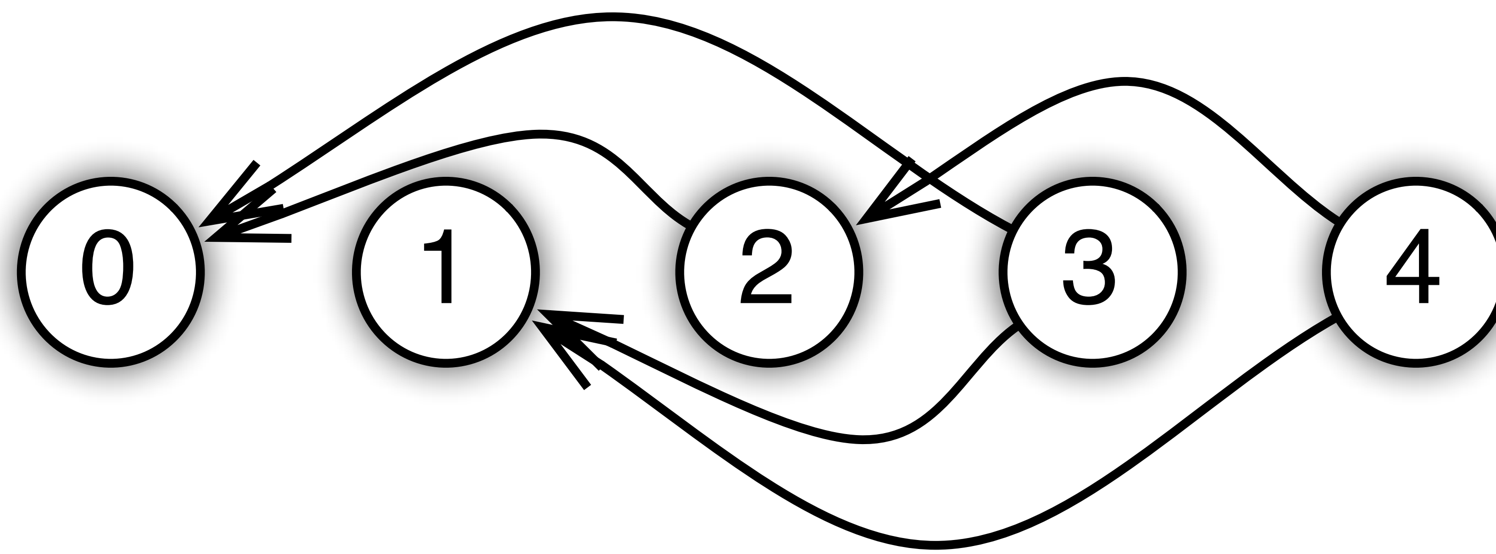
Consistency models

(a) Sequential



(b) Eventual

(c) Bounded delay



via task processing engine on client/controller

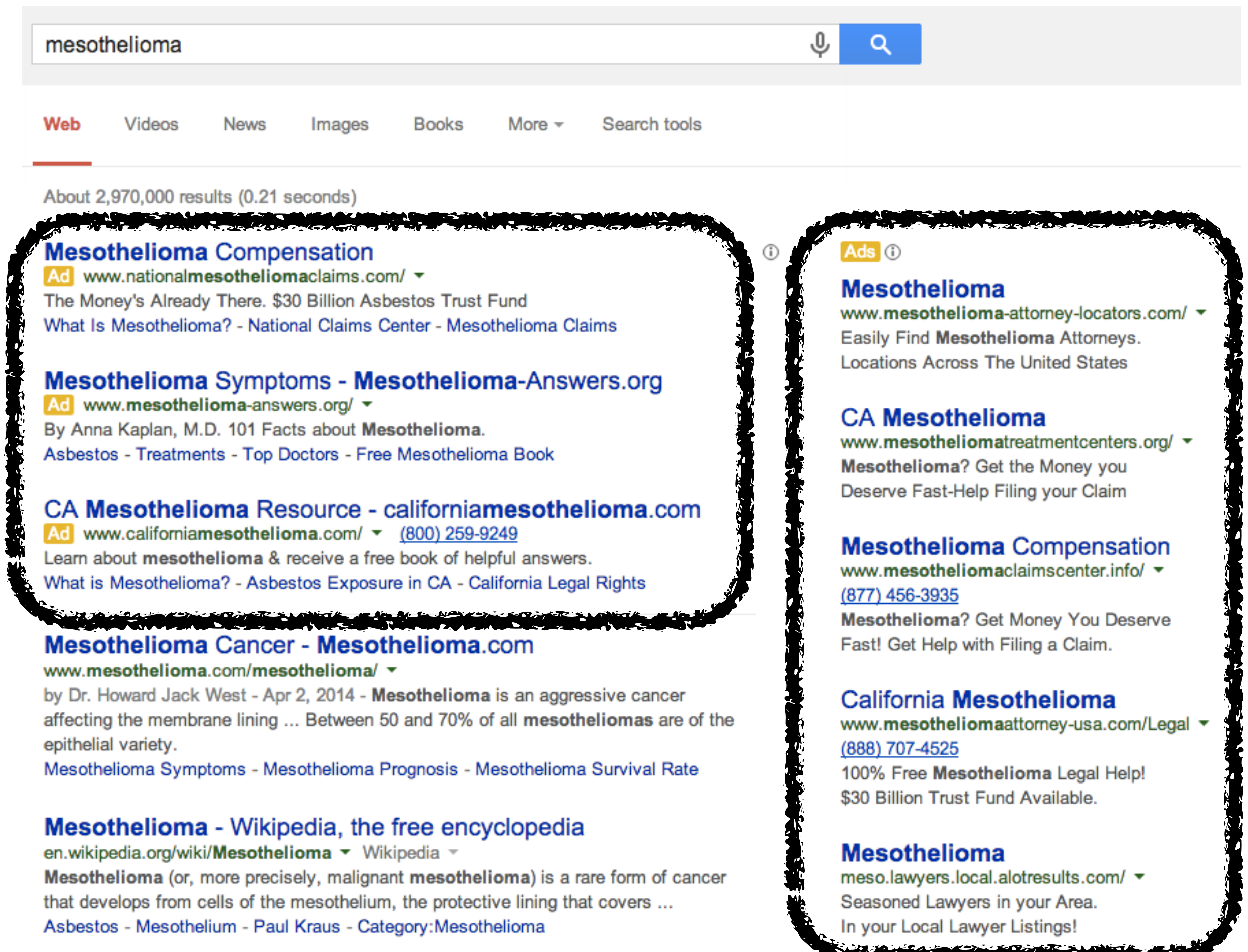
Vector Clocks for Ranges

- Keep track of when we received an update from a client / server.
- For c clients this means O(c) metadata
  This is impossible to store per key (Dynamo)
- Very cheap and feasible for ranges
- When inconsistent ranges, split segments [A,D] splits into [A,B], [B,C] and [C,D] when receiving message for [B,C]
- This is infrequent + defragmentation


Models


MSC
MSC
Logistic Regression

Recall - Computational Advertising



sponsored search picks position of ad using

$$p(\text{click}|\text{ad}) \cdot \text{bid}(\text{ad})$$

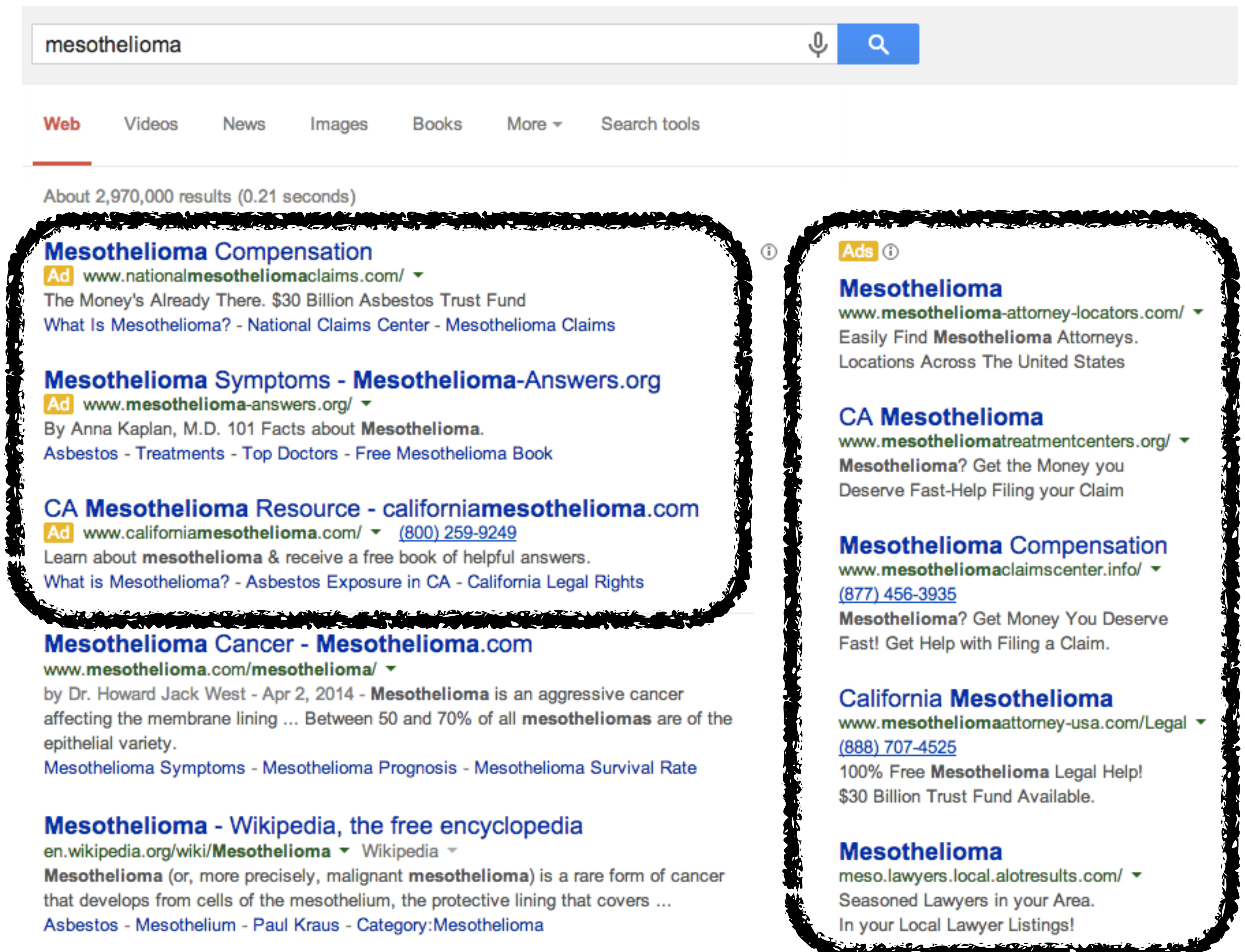
estimate it

4 million/minute

Google

Carnegie Mellon University

Recall - Computational Advertising



Google

sponsored search picks position of ad using

$$p(\text{click}|\text{ad}) \cdot \text{bid}(\text{ad})$$

estimate it

4 million/minute

Carnegie Mellon University

Estimating Probabilities

- **Logistic model (exponential family)**

$$p(y|t) \propto \exp\left(\frac{1}{2}yt\right) \text{ where } y \in \{\pm 1\}$$

y will tend to agree with the sign of t (find t)

- **Normalizing terms**

$$p(y|t) = \frac{\exp\left(\frac{1}{2}yt\right)}{\exp\left(\frac{1}{2}t\right) + \exp\left(-\frac{1}{2}t\right)} = \frac{\exp\left(\frac{1}{2}yt\right)}{\exp\left(\frac{1}{2}yt\right) + \exp\left(-\frac{1}{2}yt\right)}$$

$$= \boxed{\frac{1}{1 + \exp(-yt)}}$$

(Penalized) Maximum Likelihood

- Goal
  Find t that correlates with y
- Strategy
  Use covariates x and function f(x)

$$p(y|x) = \frac{1}{1 + \exp(-yf(x))}$$

- Penalty against overfitting / Bayes rule

$$p(f|X, Y) \propto p(f) \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i f(x_i))}$$

Penalized Maximum Likelihood

- **Picking a function class**

$$f(x) = \langle w, x \rangle$$

- **Picking a prior**

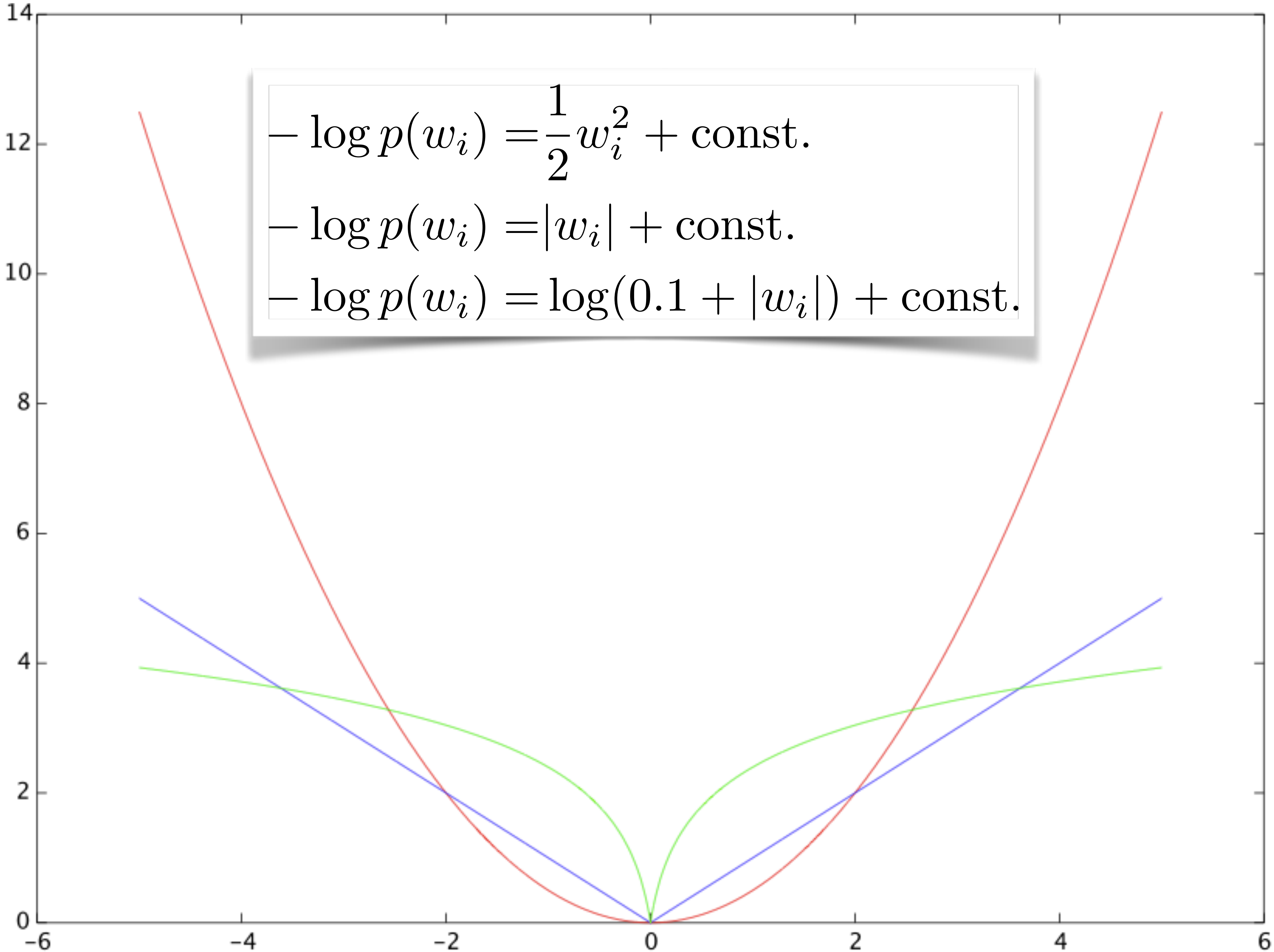
$$\log p(f) = \lambda \|w\|_1 + \text{const.}$$

we want sparse models for advertising

- **Picking an inference strategy**

$$\underset{w}{\text{minimize}} - \log p(f|X, Y)$$

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle w, x_i \rangle)) + \lambda \|w\|_1$$


$-\log p(w_i) = \frac{1}{2} w_i^2 + \text{const.}$
$-\log p(w_i) = |w_i| + \text{const.}$
$-\log p(w_i) = \log(0.1 + |w_i|) + \text{const.}$
14
12
10
8
6
4
2
0
−6
−4
−2
0
2
4
6

Proximal Algorithm

- Problem - l_1 norm is non-smooth
- Proximal operator

$$\underset{w}{\operatorname{argmin}} \|w\|_1 + \frac{\gamma}{2} \|w - (w_t - \eta g_t)\|$$

(more generally use penalty on w)

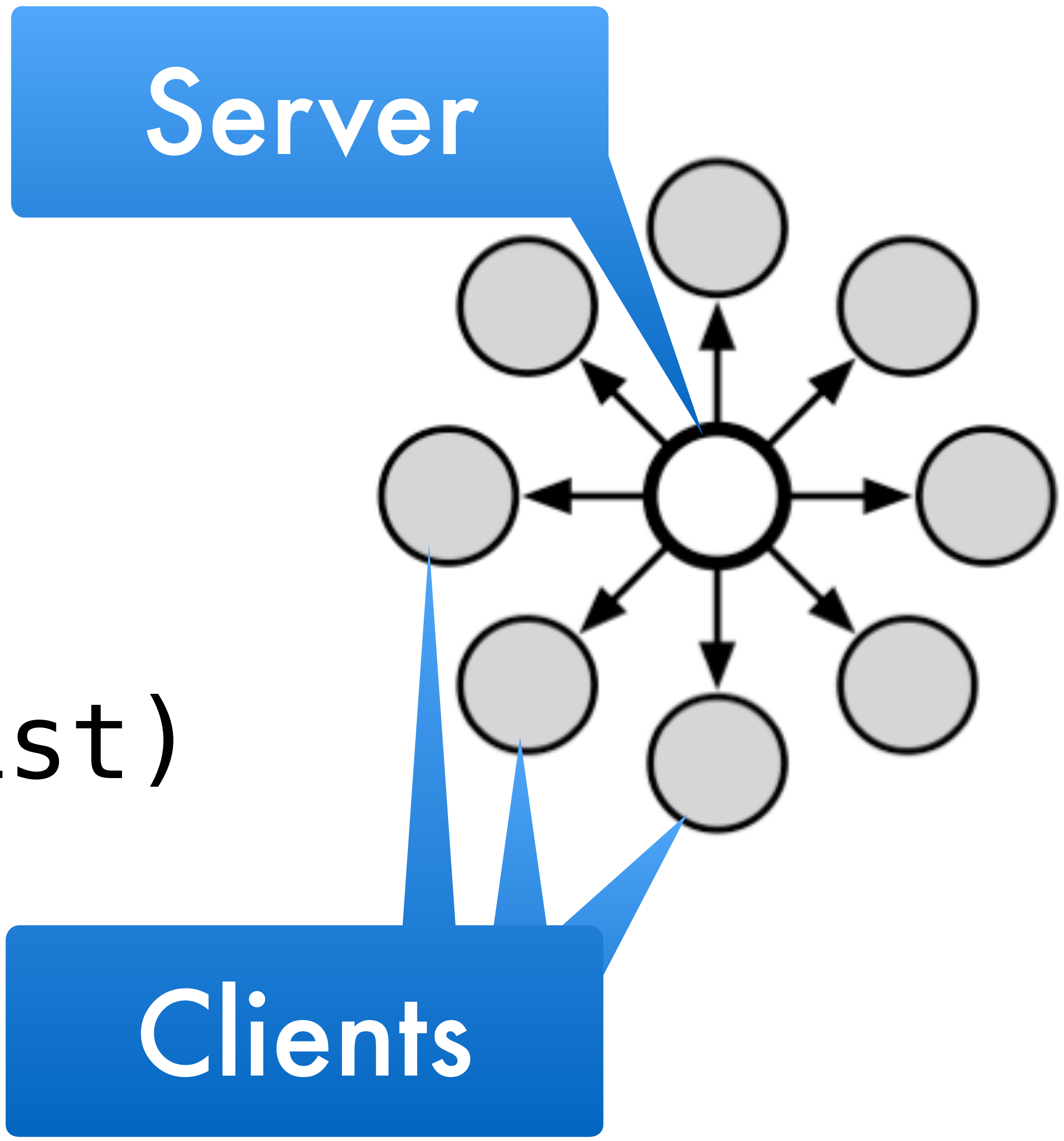
- Updates for l_1 are

$$w_i \leftarrow \operatorname{sgn}(w_i) \max(0, |w_i| - \epsilon)$$

(update and project back to polytope)

Generic Parallel Template

- Compute gradient on (subset of data) on each client
- Send gradient from client to server asynchronously
  `push(key_list,value_list)`
- Proximal gradient update on server
- Server returns parameters
  `pull(key_list,value_list)`



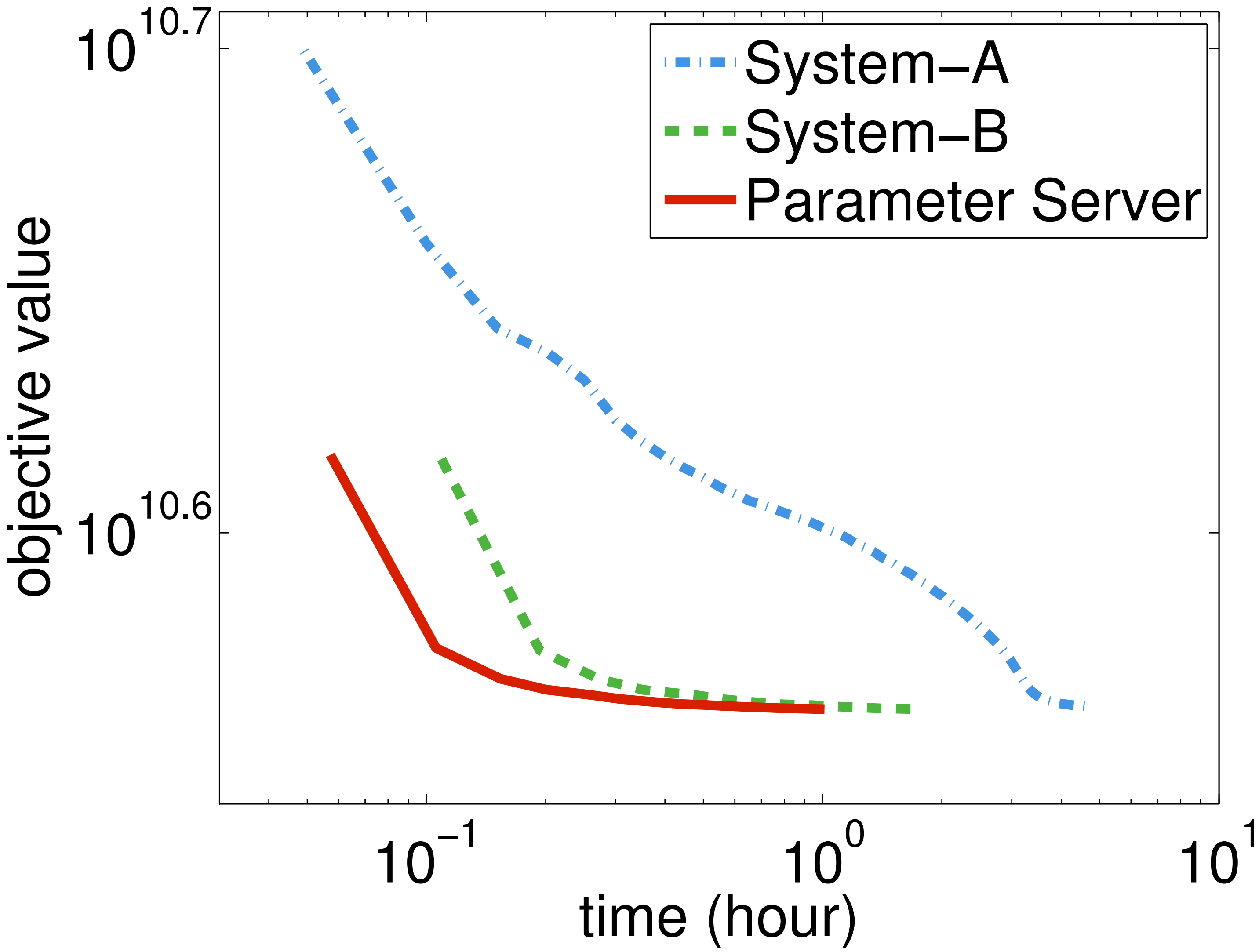
Guinea pig - logistic regression

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} \log(1 + \exp(-y_i \langle x_i, w \rangle)) + \lambda \|w\|_1$$

- **Implementation on Parameter Server**

	Method	Consistency	LOC
System-A	L-BFGS	Sequential	10,000
System-B	Block PG	Sequential	30,000
Parameter Server	Block PG	Bounded Delay KKT Filter	300

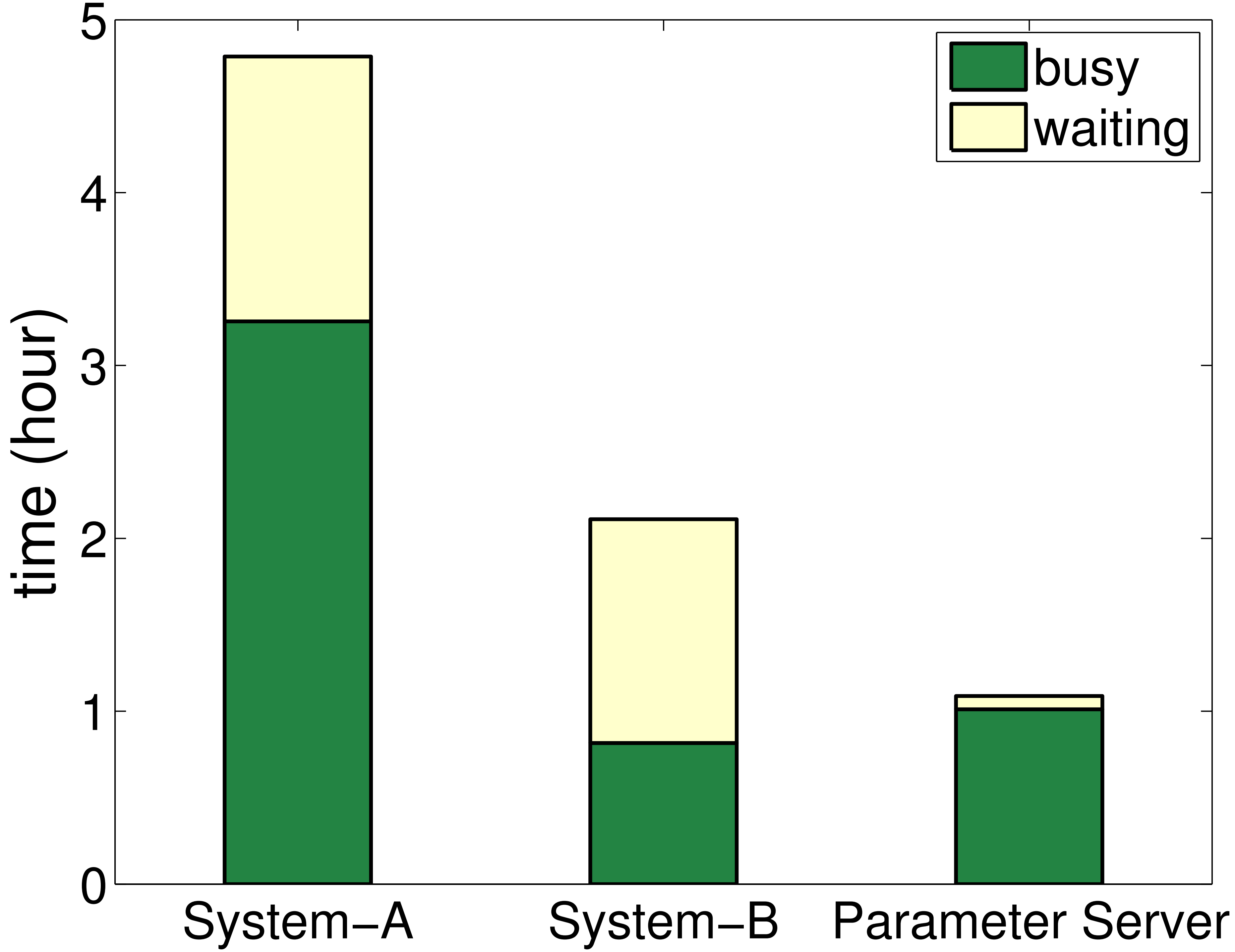
Convergence speed



500TB CTR data
100B variables
1000 machines

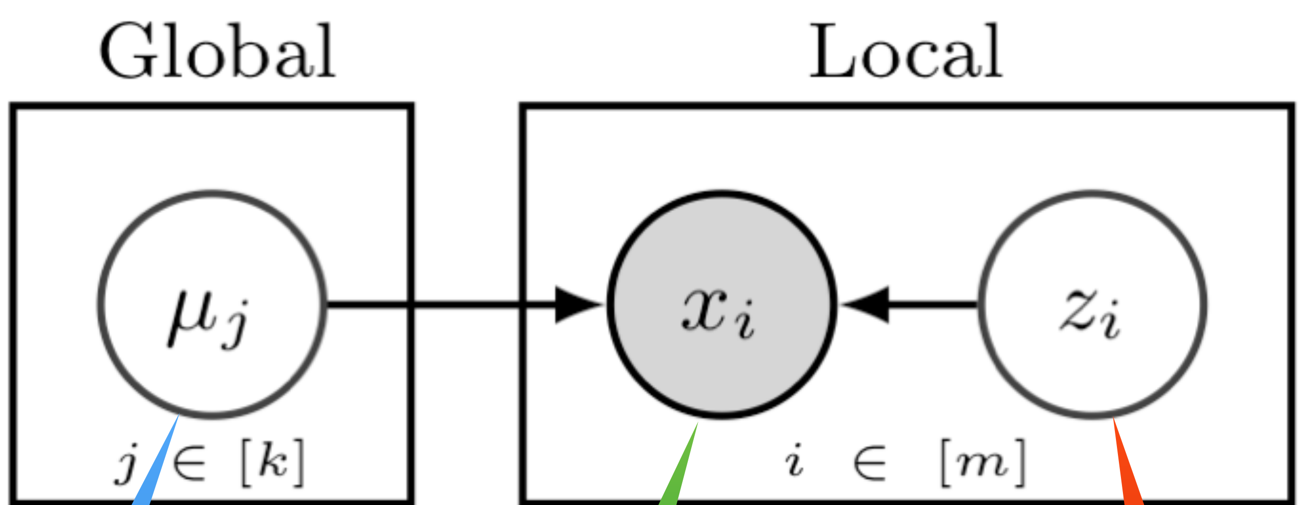
- System A and B are production systems at a very large internet company …

Scheduling Efficiency



Scaling Latent Variable Models

Clustering

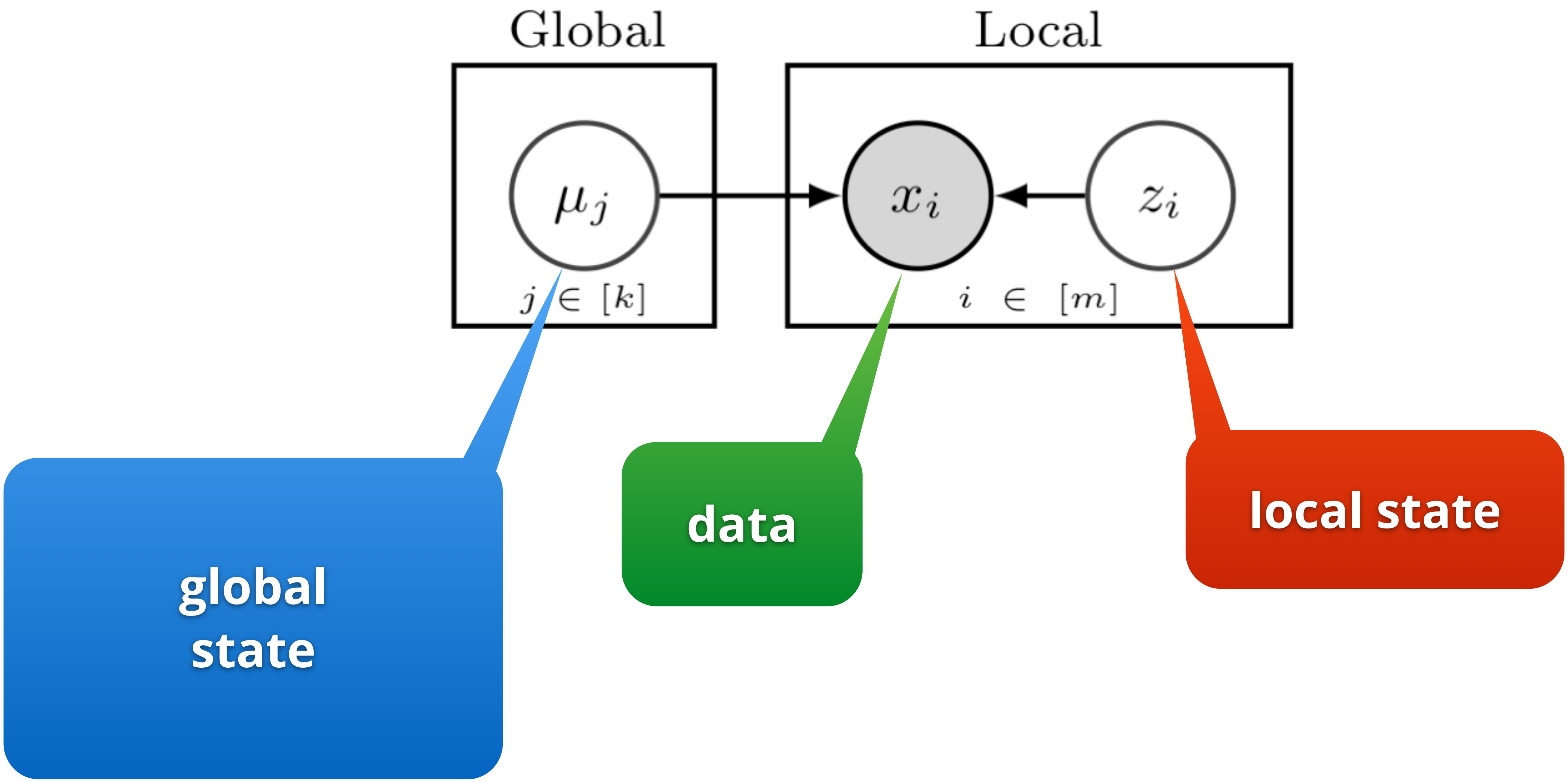


mean
variance
cluster weight

data

cluster ID

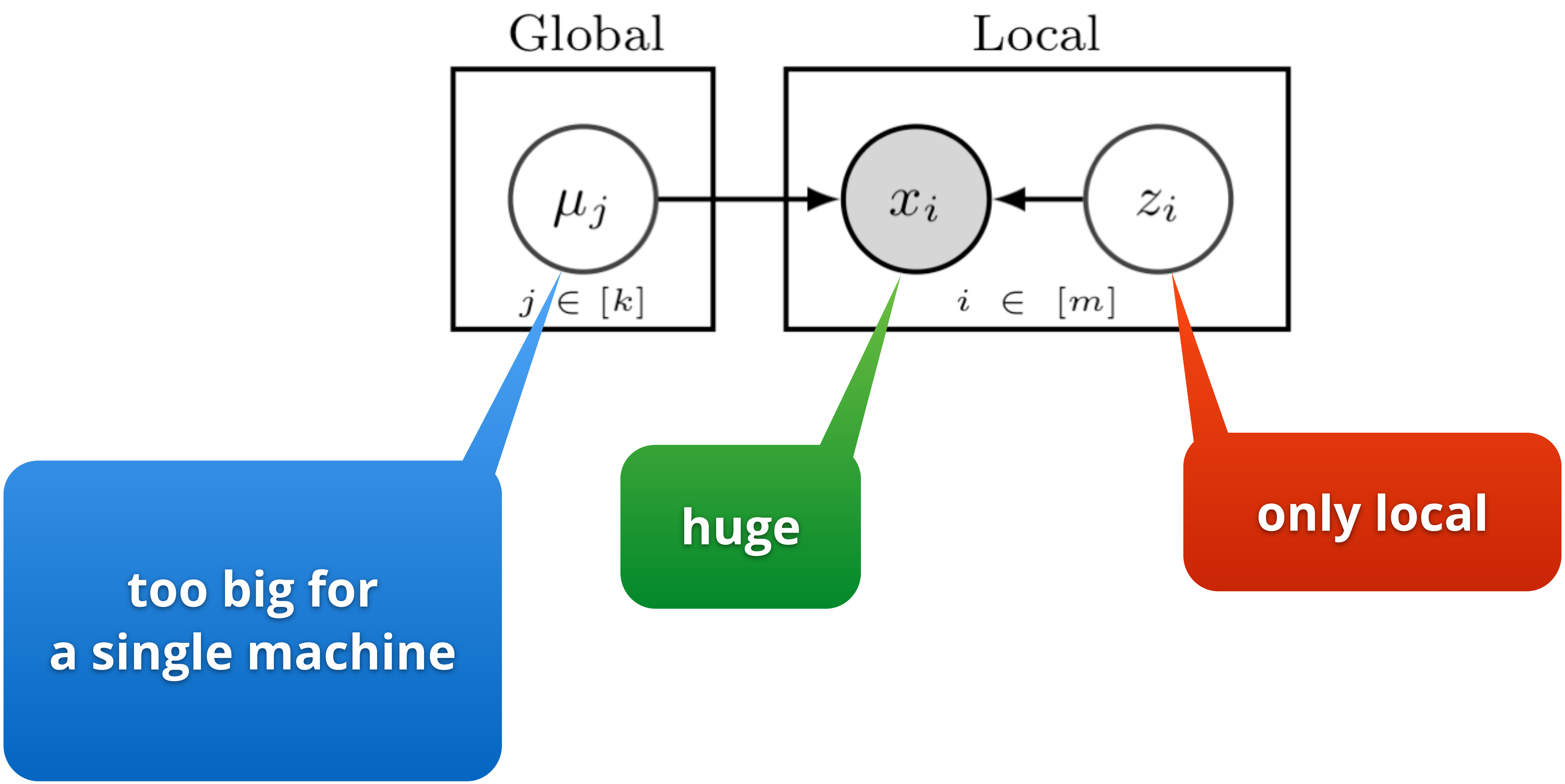
Clustering



Google

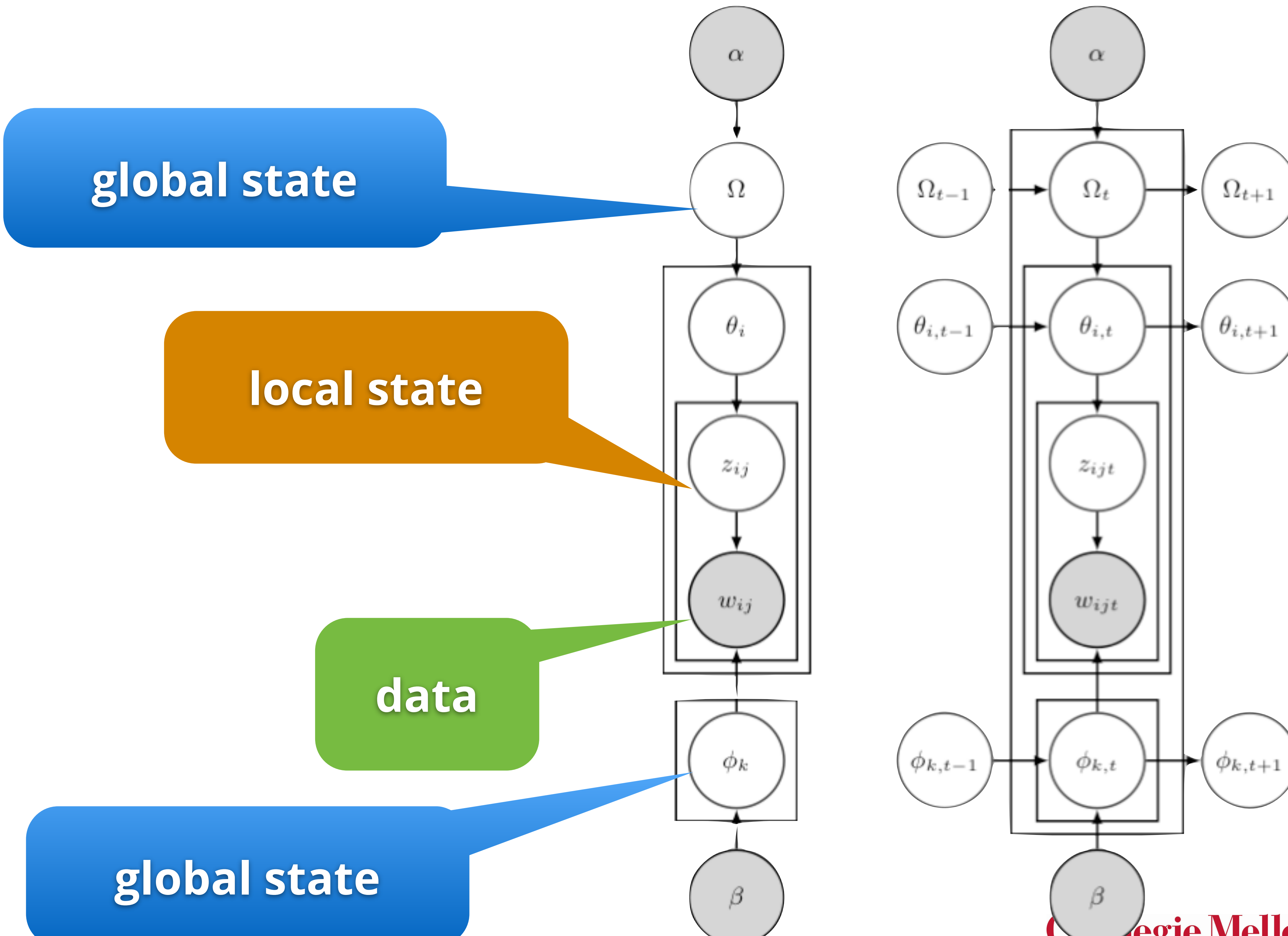
Carnegie Mellon University

Clustering



Global
Local
μ_j
x_i
z_i
$j \in [k]$
$i \in [m]$
too big for
a single machine
huge
only local

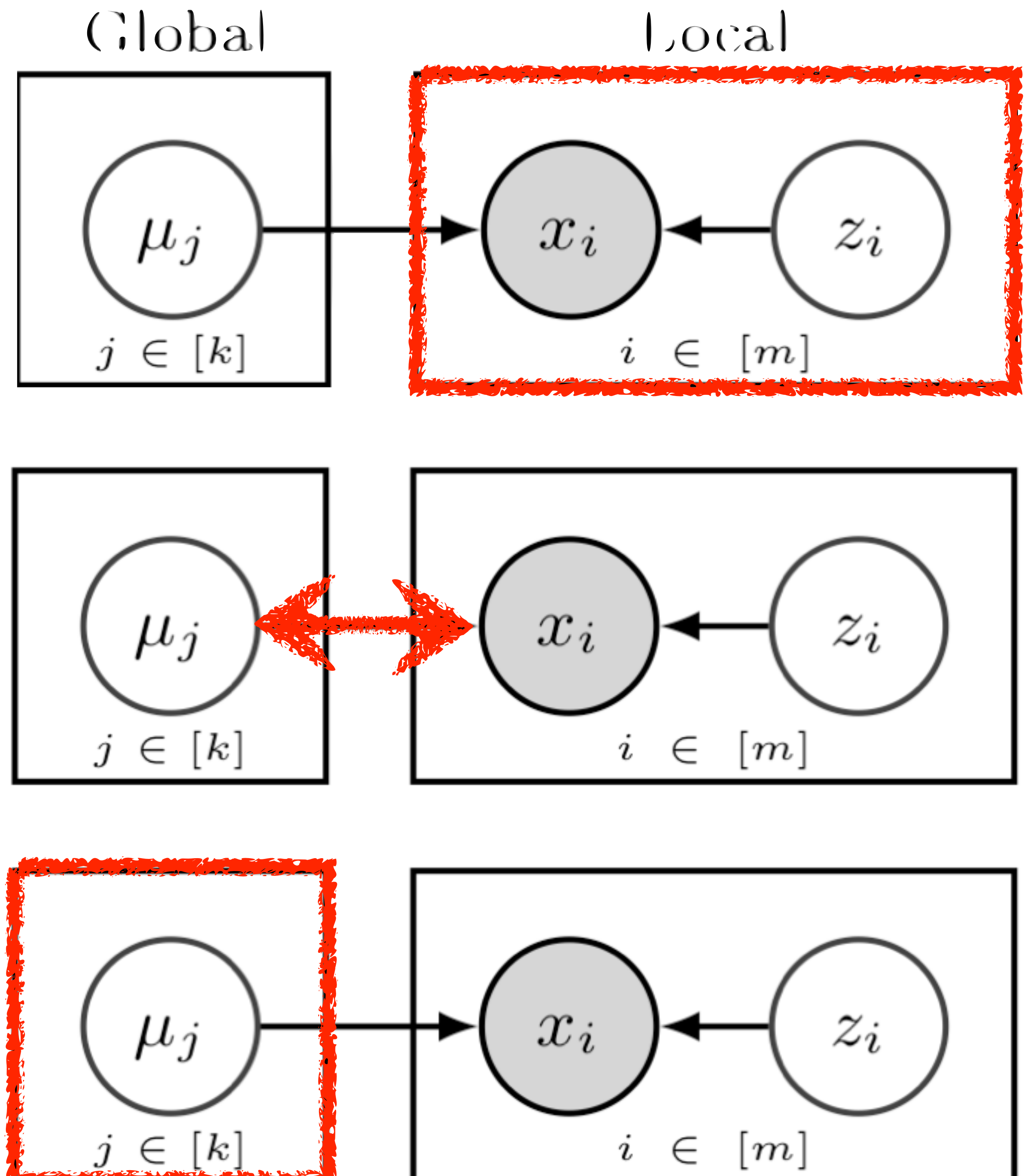
Fancy models



Google Carnegie Mellon University

ParameterServer to the rescue

local state
is too large



does not fit
into memory

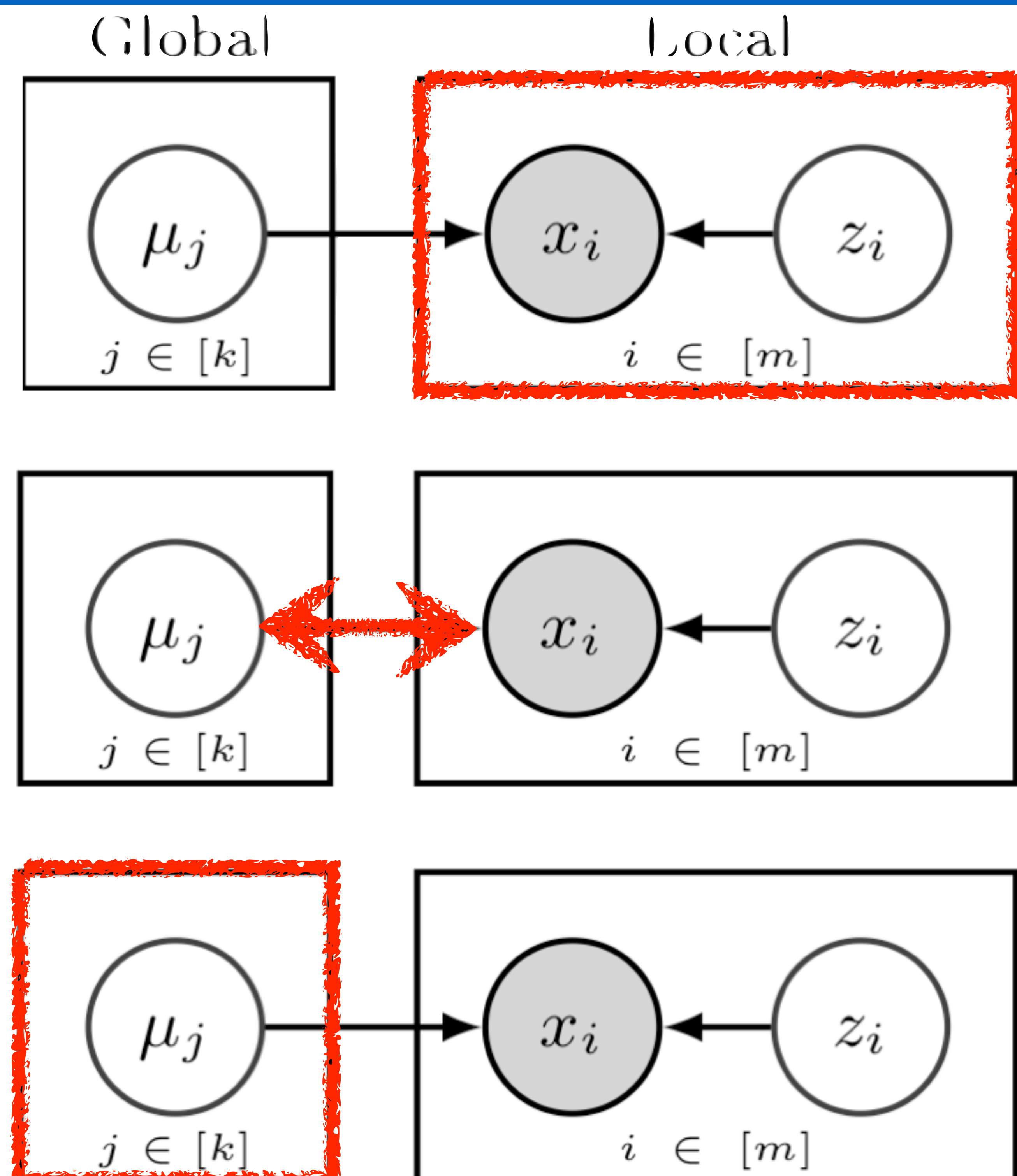
network load
& barriers

global state
is too large

does not fit
into memory

ParameterServer to the rescue

local state
is too large



stream local
data from disk

network load
& barriers

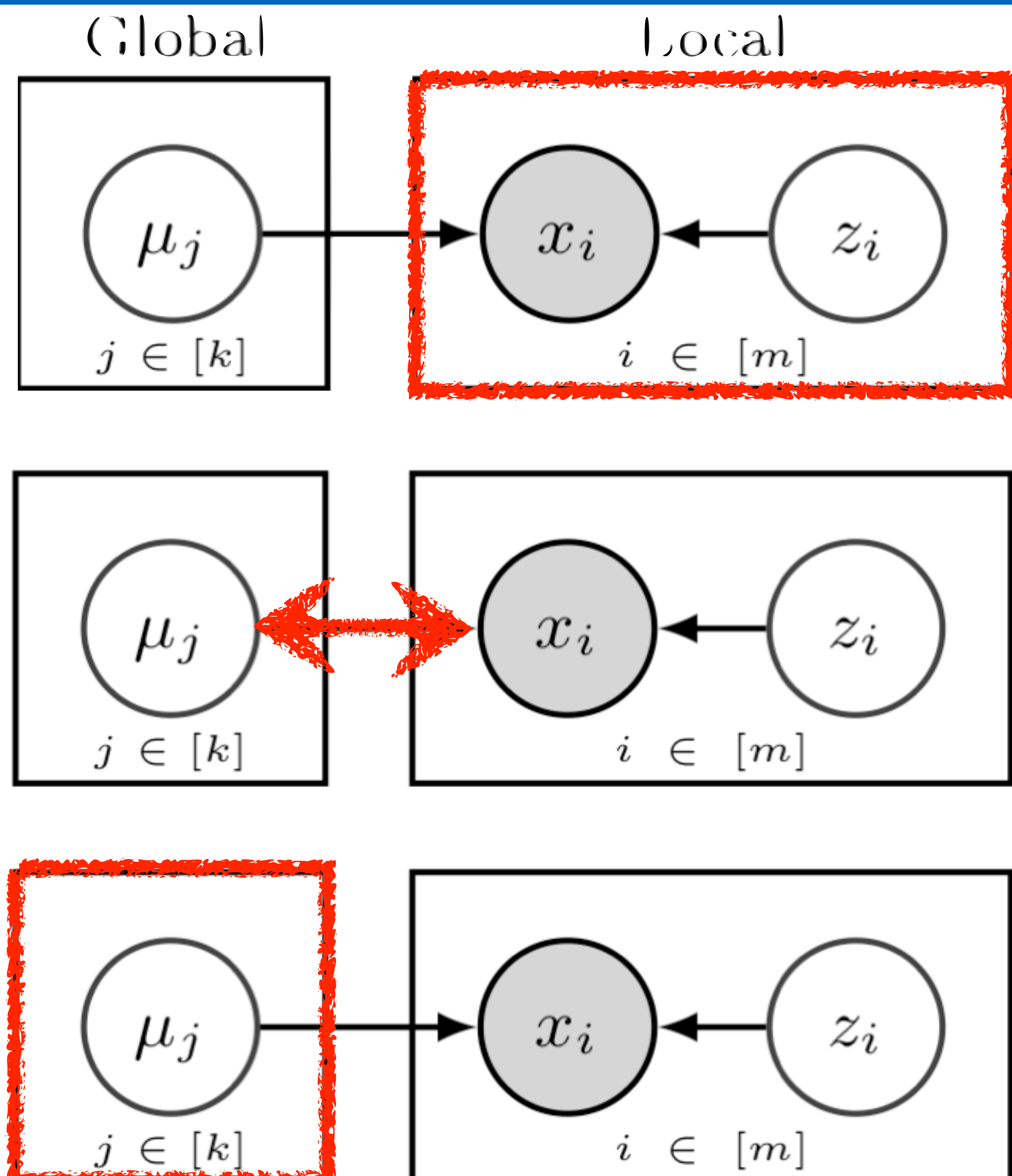
global state
is too large

does not fit
into memory

ParameterServer to the rescue

local state is too large

global state is too large



stream local data from disk

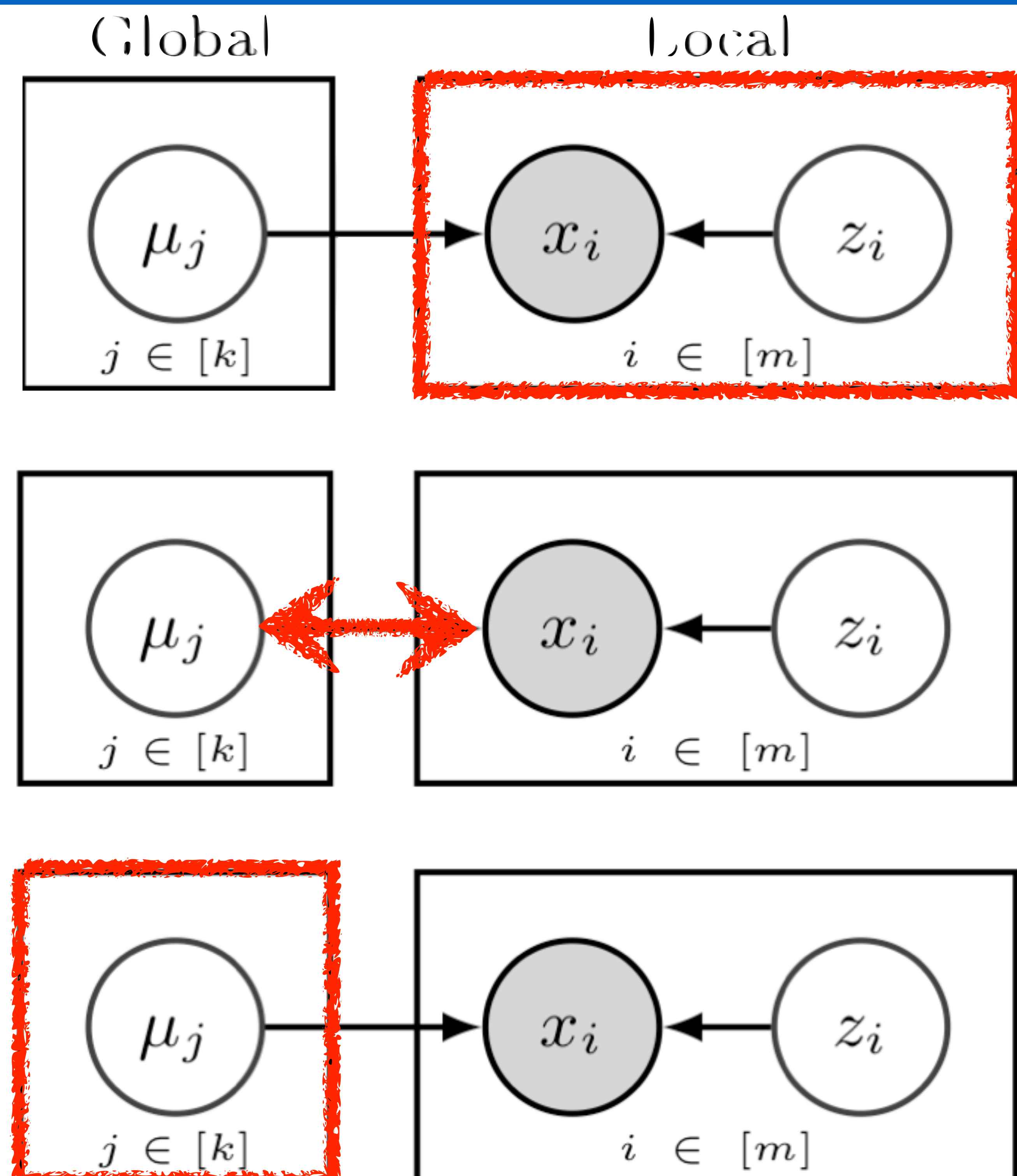
asynchronous synchronization

does not fit into memory

ParameterServer to the rescue

local state
is too large

global state
is too large

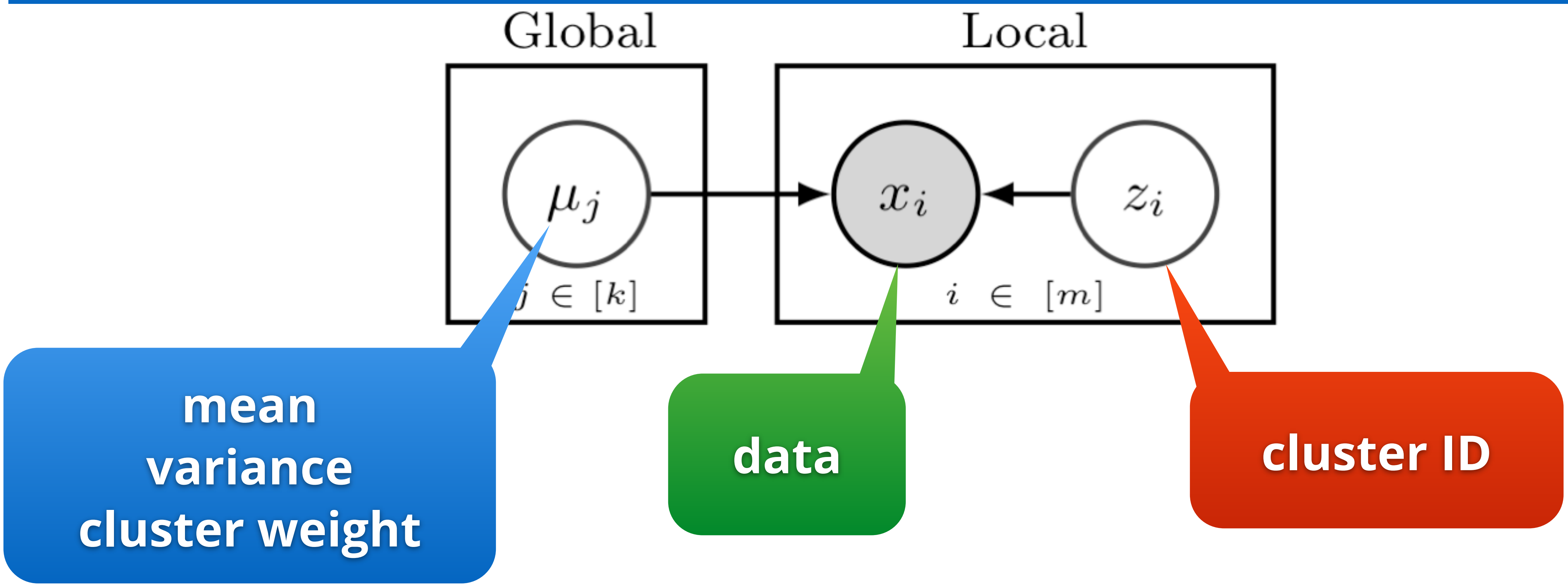


stream local data from disk

asynchronous synchronization

partial view shared between threads

Synchronization Strategy



- Locally Gibbs Sample cluster ID

$$p(z_i | x_i, \text{rest}) \propto p(z_i | Z^{-i}) p(x_i | X^{-i}, Z^{-i}, z_i)$$

- Communicate changes in statistics of data to server (mean, variance, cluster size)

Mixture of Gaussians

- **Multinomial with Dirichlet for cluster ID**

$$p(Z|\theta) = \prod_{i=1}^{m} \theta_{z_i} \text{ and } p(\theta|\alpha) = \mathrm{Dir}(\alpha)$$

- **Integrating out multinomial yields collapsed**

$$p(z_i = z|Z^{-i}) = \frac{n_z^{-i} + \alpha_z}{n - 1 + \sum_{z'} \alpha_{z'}}$$

- **Gaussian with Gauss-Wishart for data**

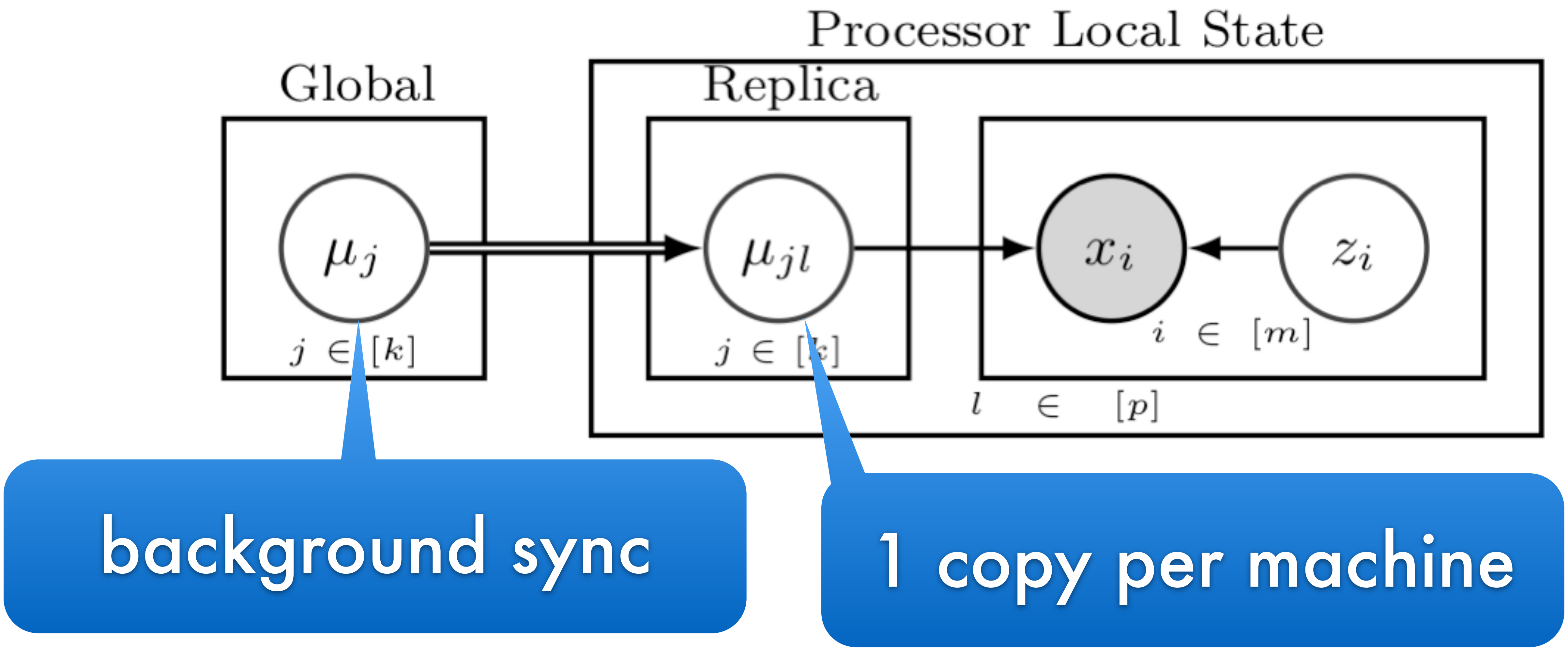
$$x_i|z_i \sim \mathcal{N}\left(\mu_{z_i}, \Sigma_{z_i}\right) \text{ and } \left(\mu_{z_i}, \Sigma_{z_i}\right) \sim \mathrm{GaussWishart}(m_0, \mu_0, Q_0)$$

- **Only need to sync** $(n_z, l_z, Q_z) := \sum_{z_i = z} (1, x_i, x_i x_i^\top)$

Google

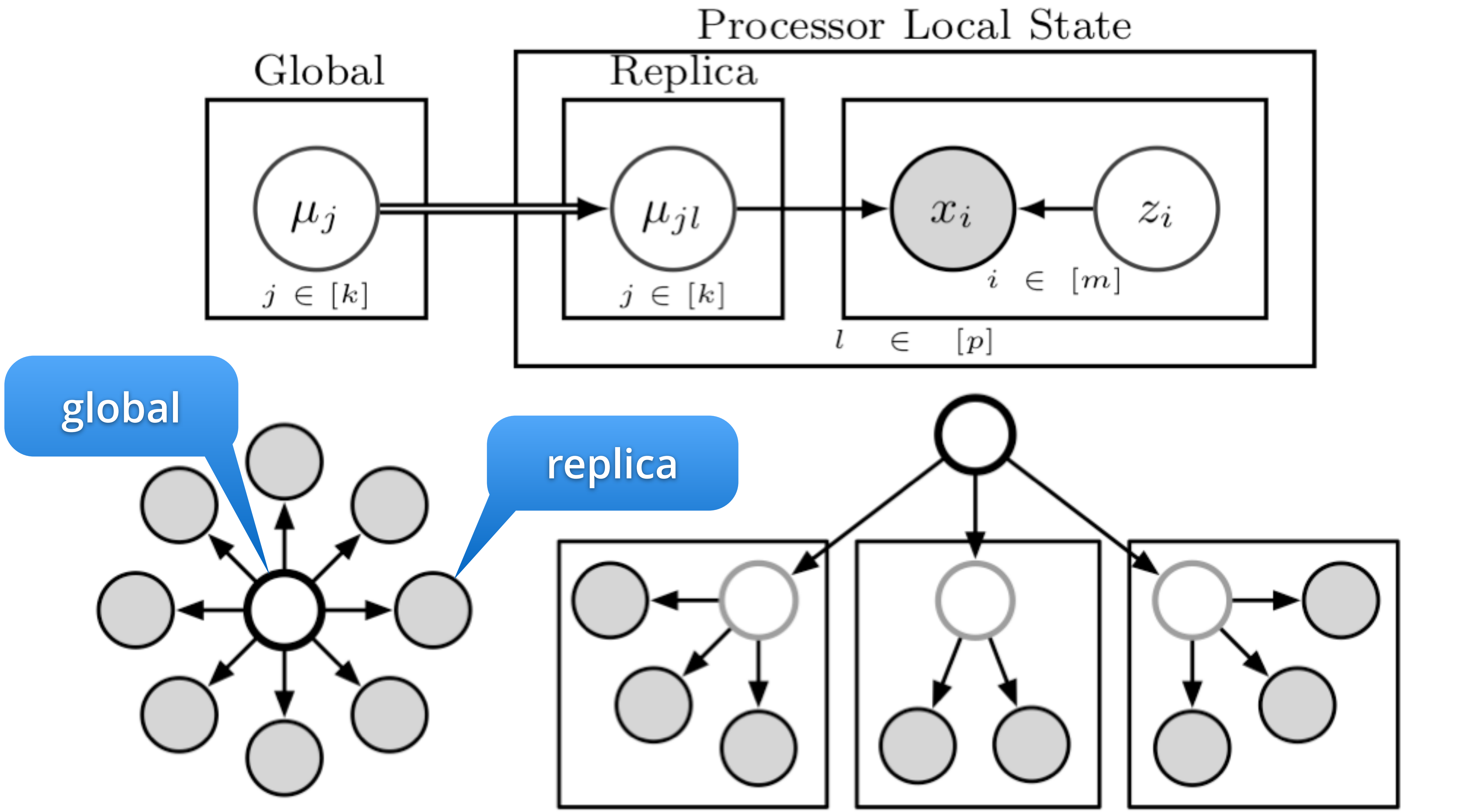
Carnegie Mellon University

Local and Global Variables



- No locks between machines to access z
- Synchronization mechanism for global μ needed
- In LDA this is the local copy of the (topic,word) counts

Local and Global Variables

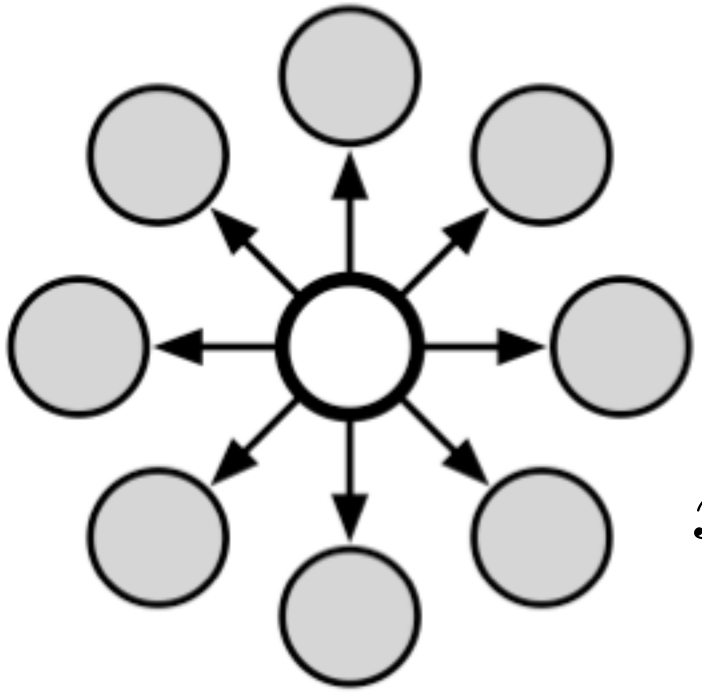


Message Passing

- Start with common state
- Child stores old and new state
- Parent keeps global state
- Transmit differences asynchronously
  - Inverse element for difference
  - Abelian group for commutativity
    (sum, log-sum, cyclic group, exponential families)

local to global

$$\delta \leftarrow x - x^{\text{old}}$$
$$x^{\text{old}} \leftarrow x$$
$$x^{\text{global}} \leftarrow x^{\text{global}} + \delta$$



global to local

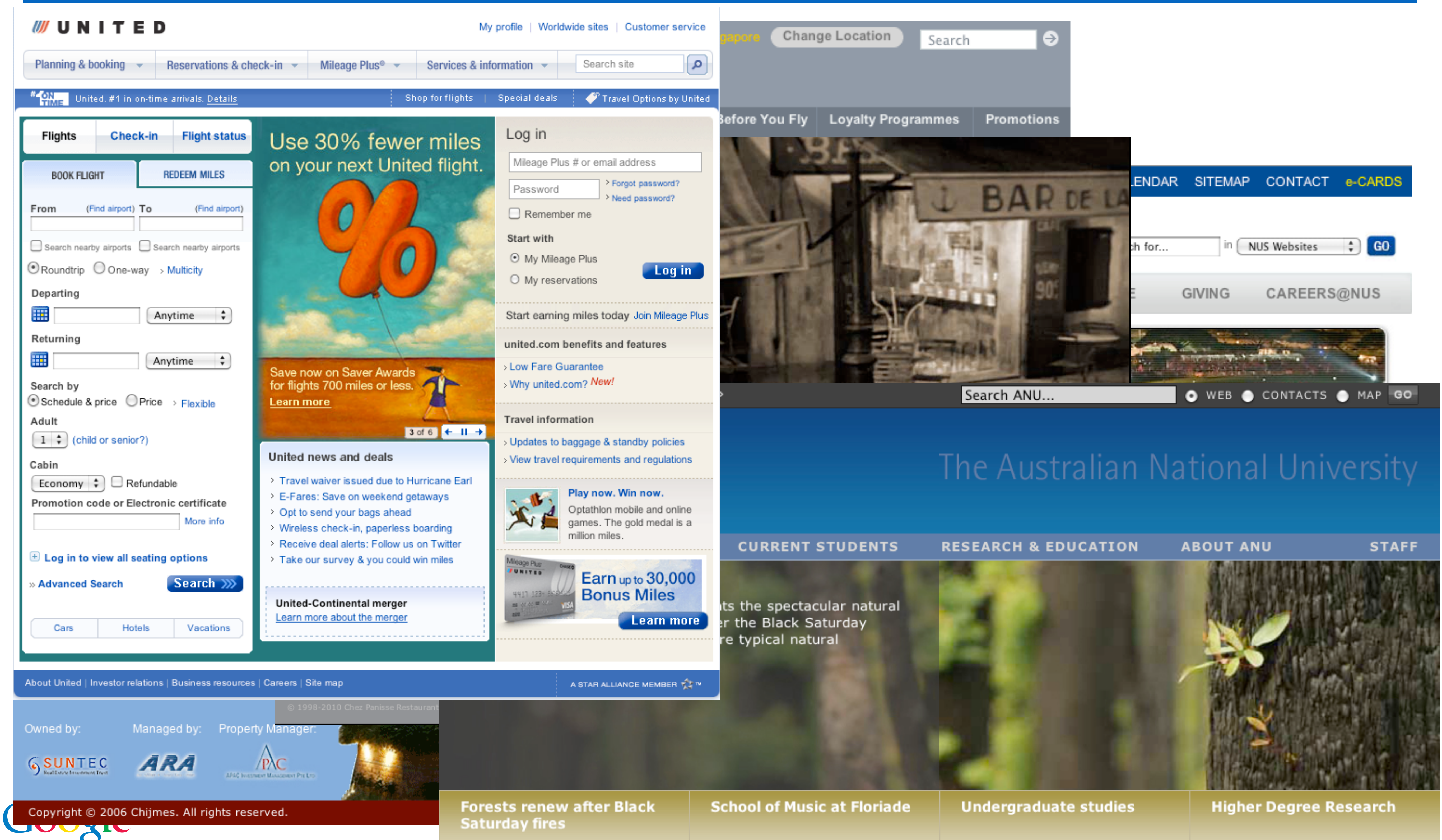
$$x \leftarrow x + (x^{\text{global}} - x^{\text{old}})$$
$$x^{\text{old}} \leftarrow x^{\text{global}}$$

Models

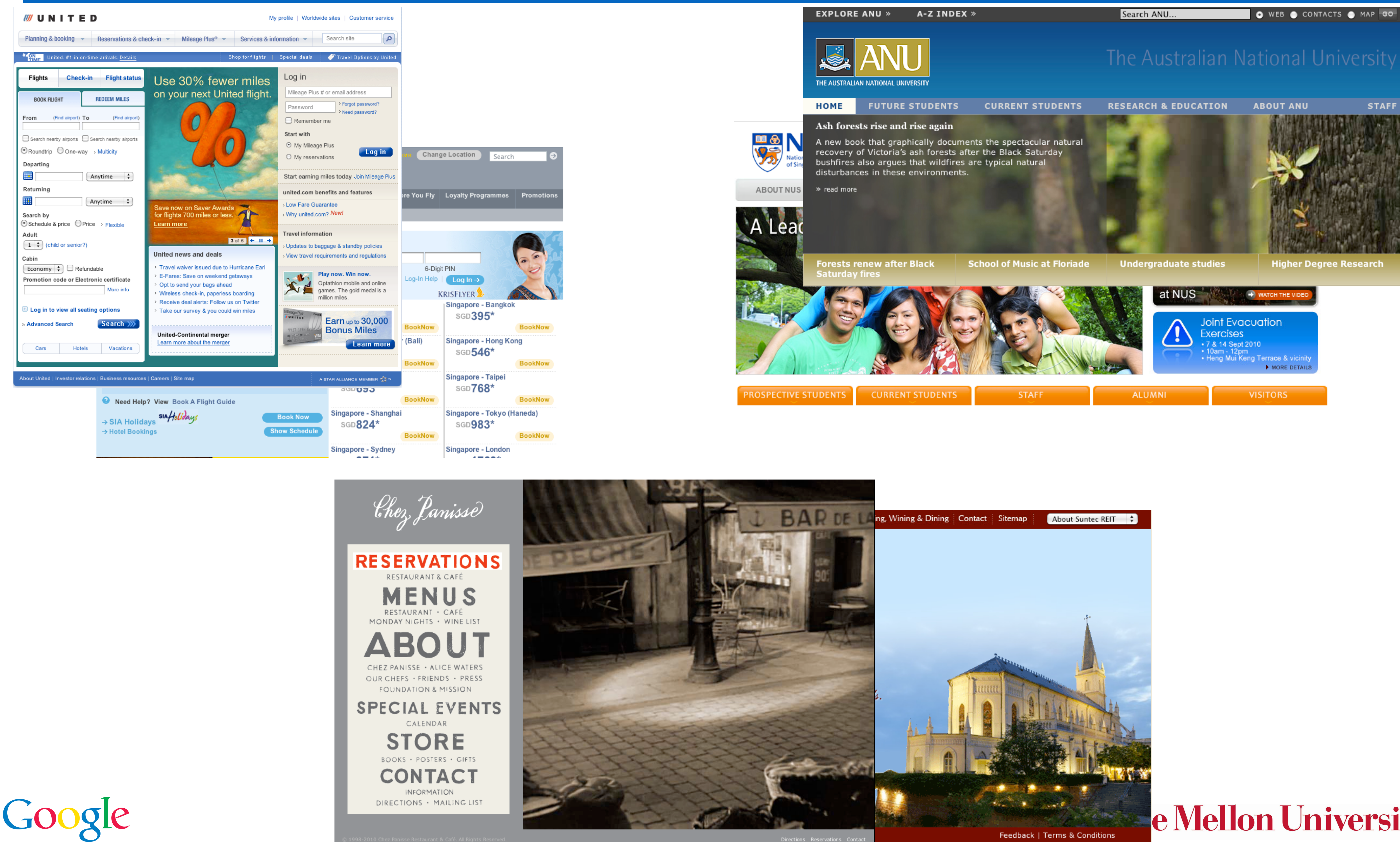
Grouping objects

Grouping objects

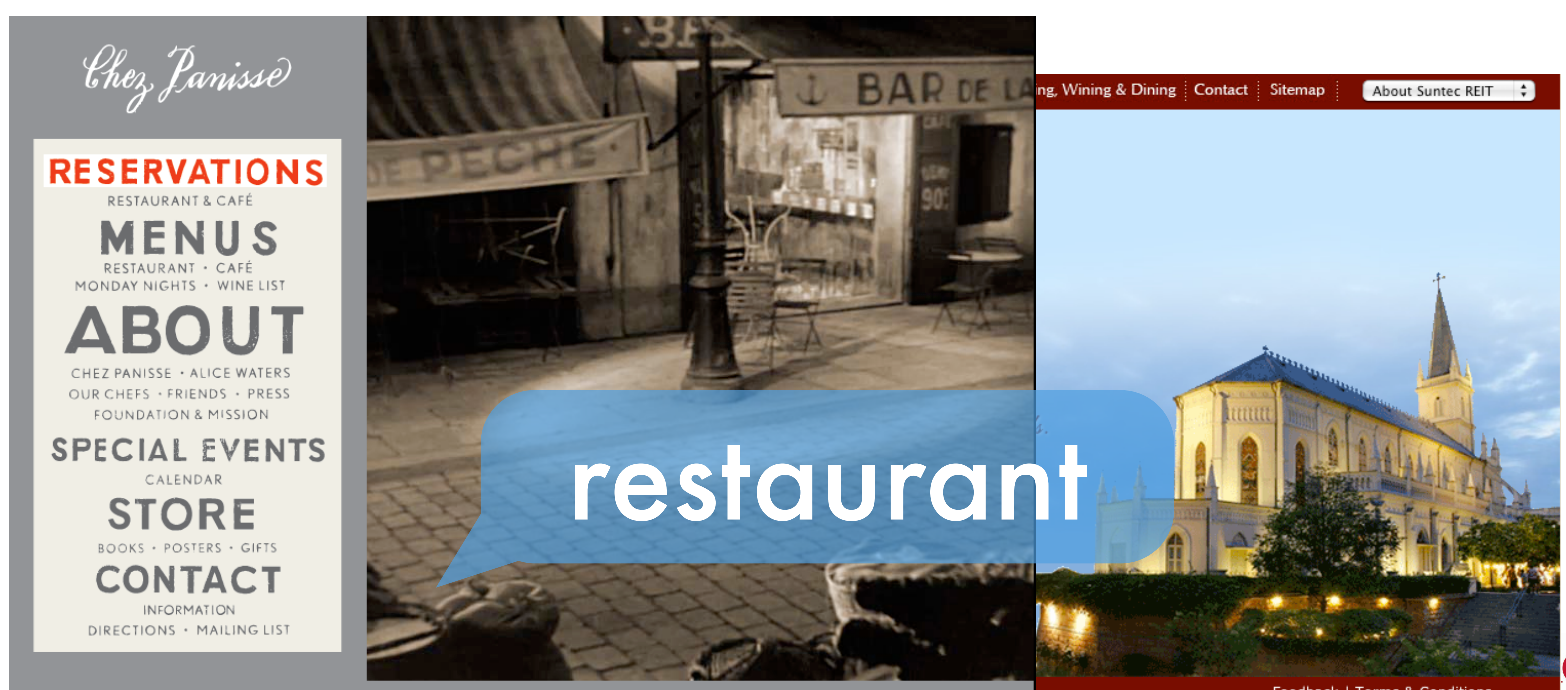
Grouping objects



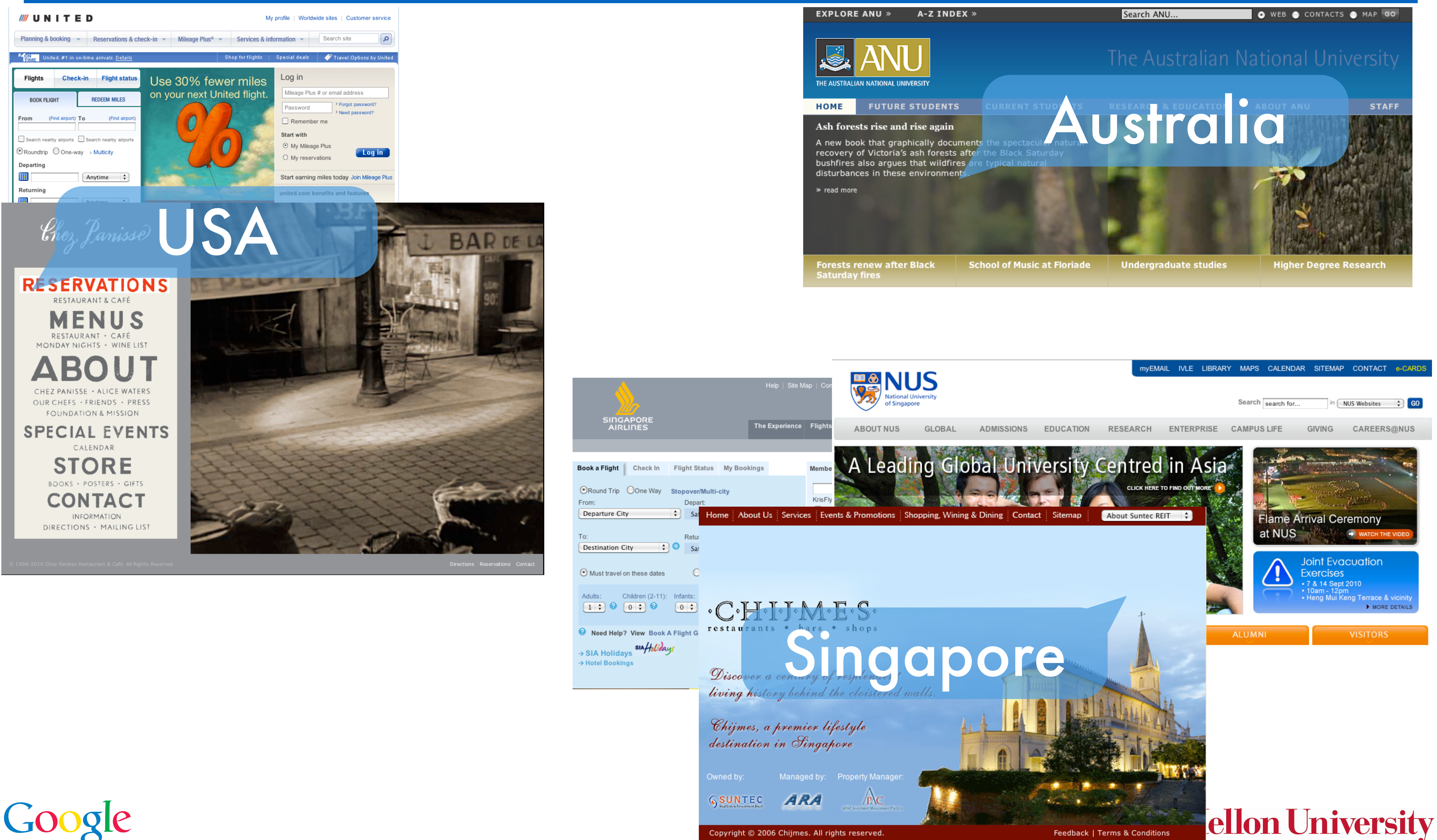
Grouping objects



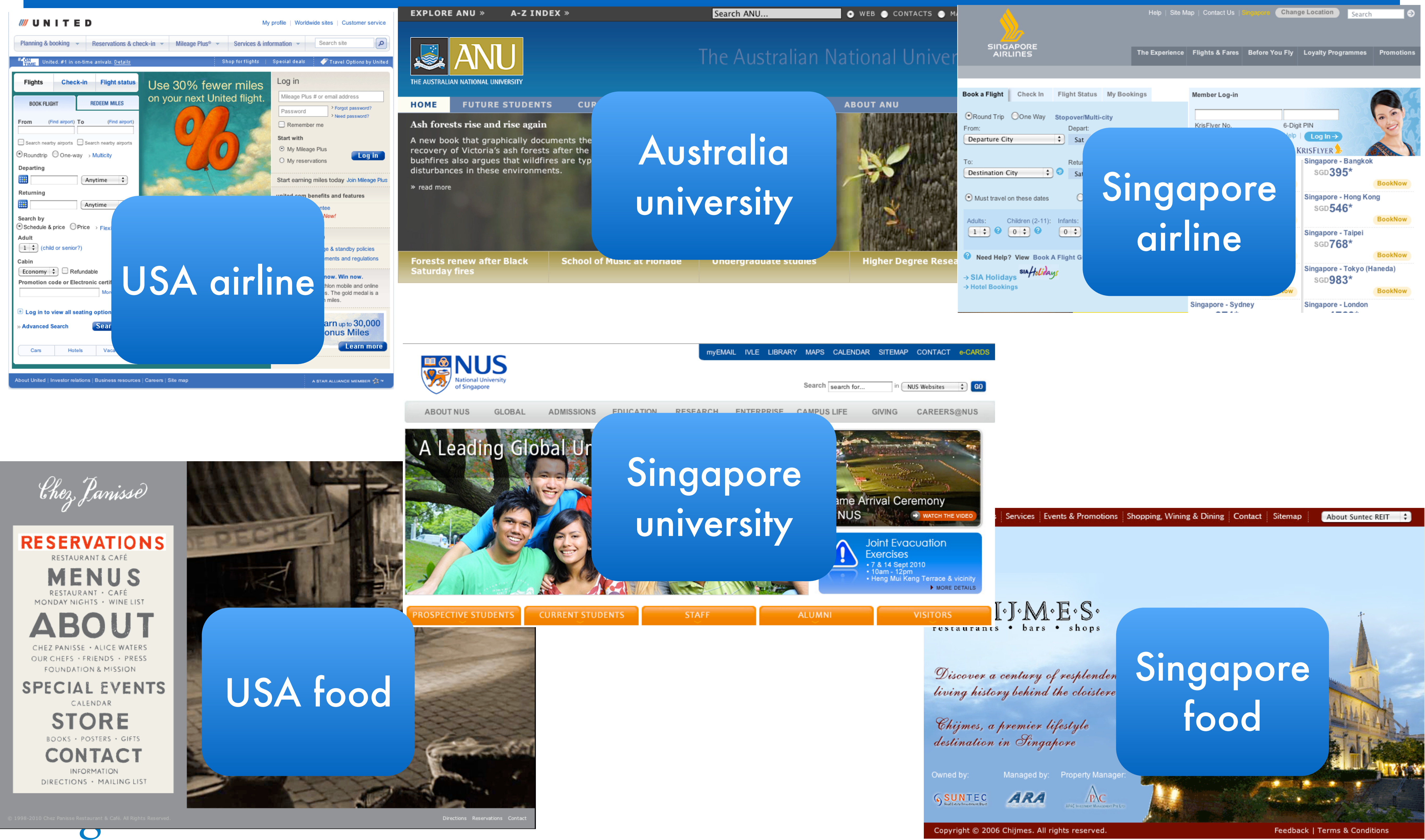
Grouping objects



Grouping objects

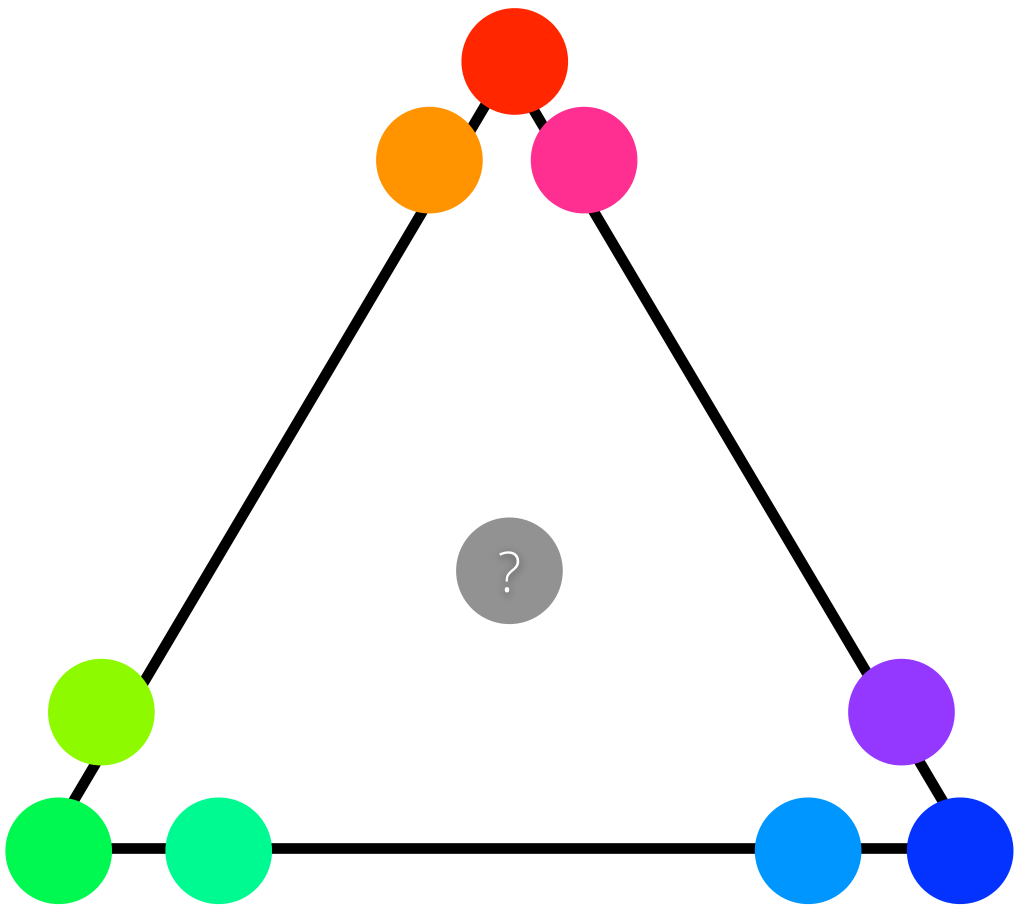


Topic Models



Clustering & Topic Models

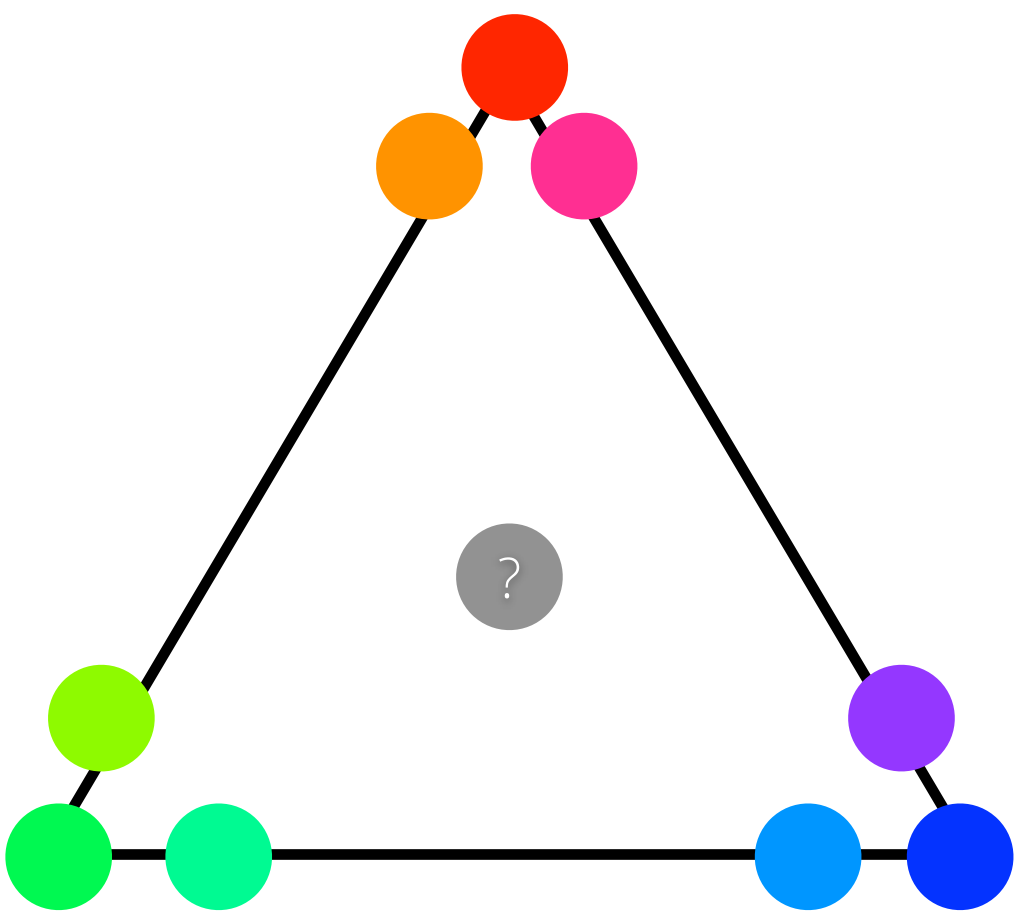
Clustering



group objects
by prototypes

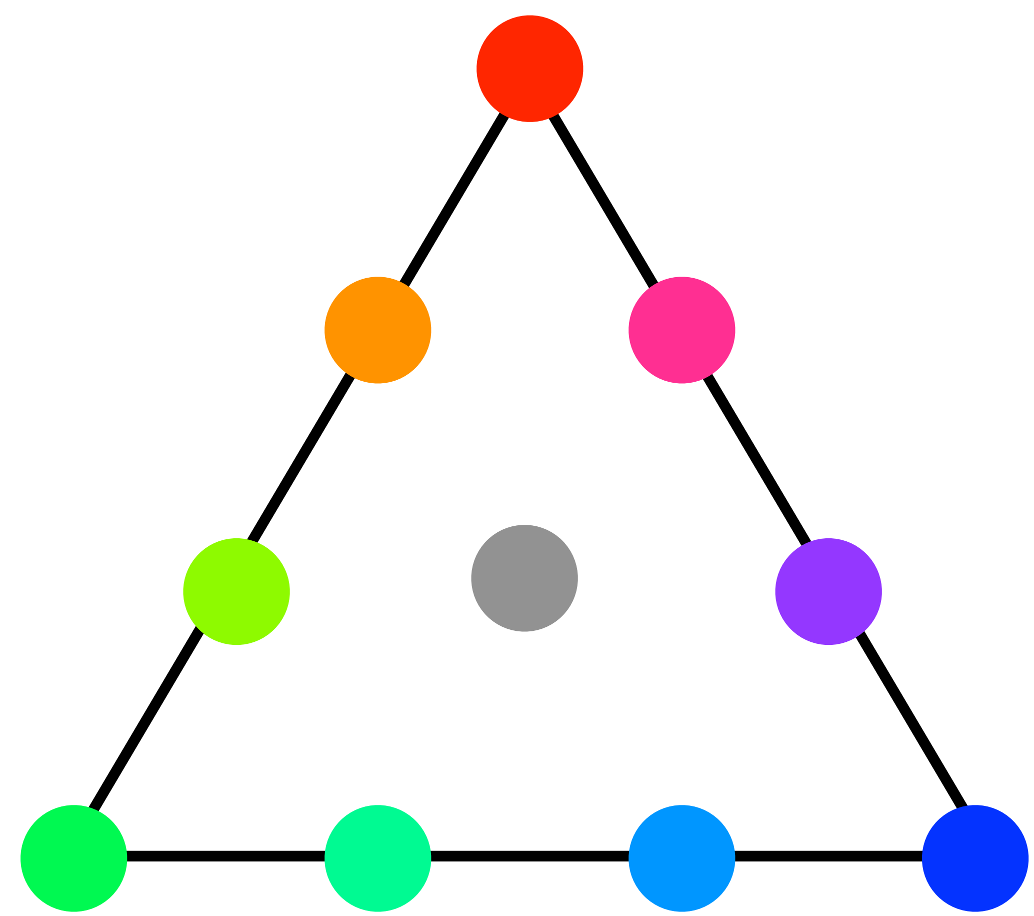
Clustering & Topic Models

Clustering



group objects
by prototypes

Topics



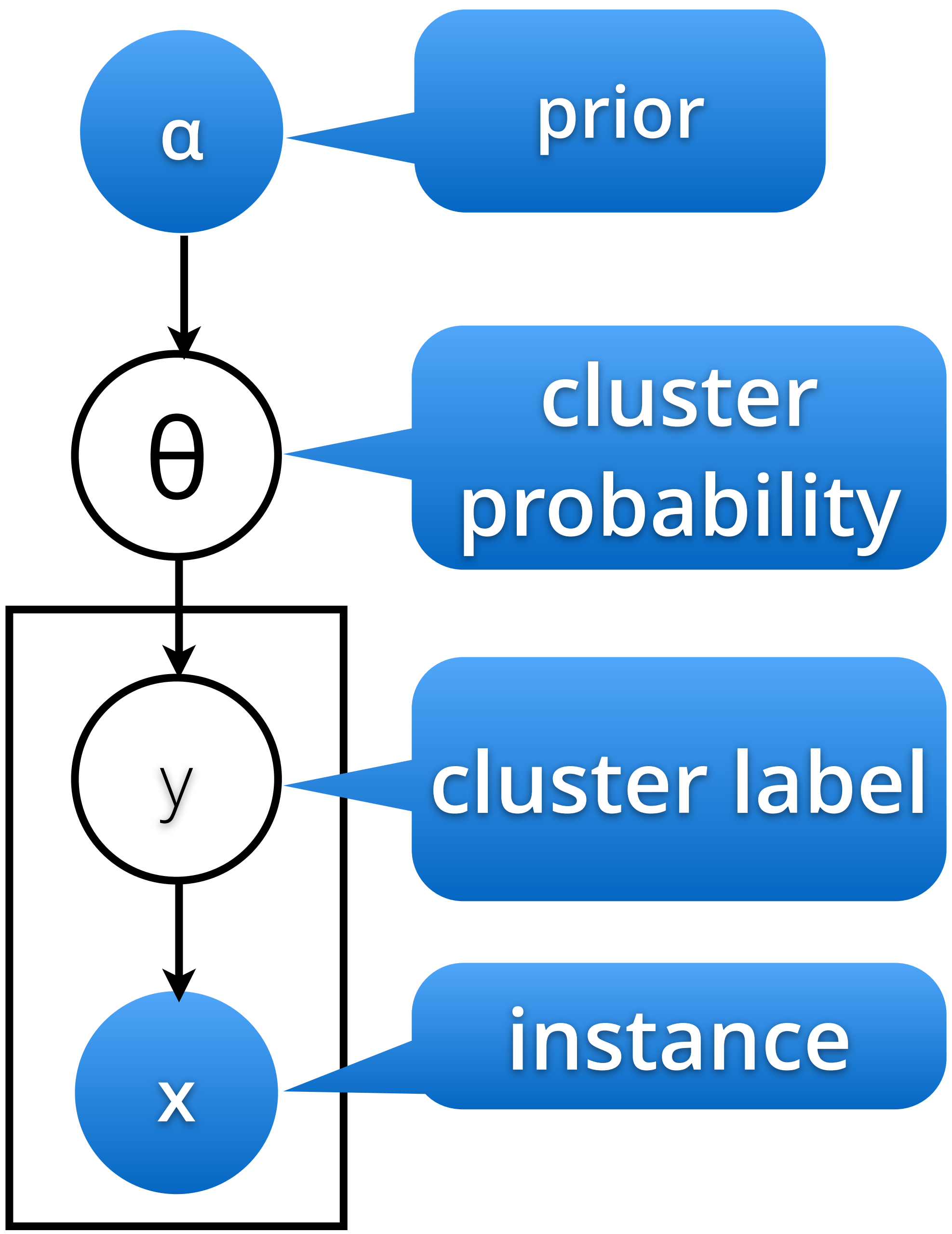
decompose objects
into prototypes

Google

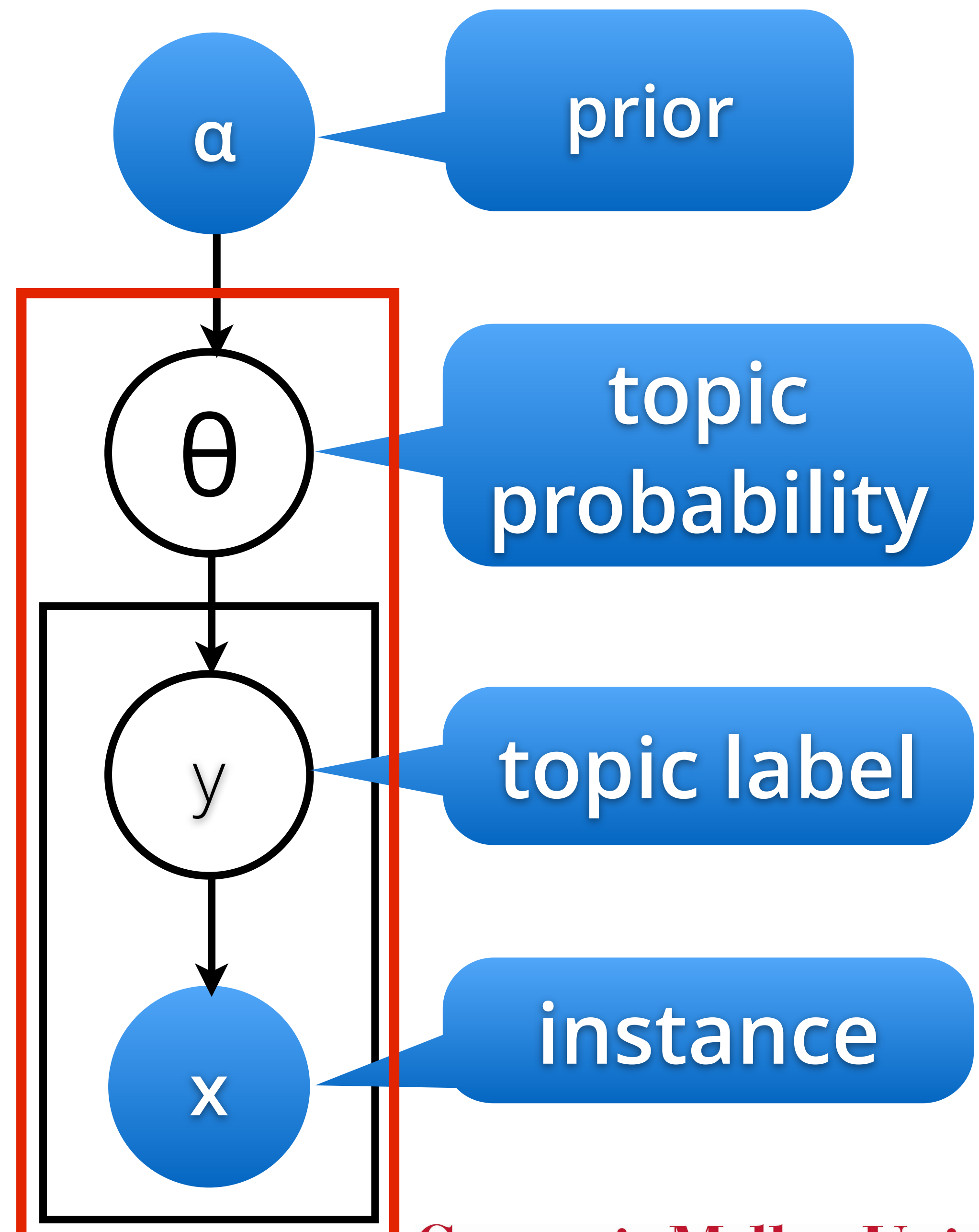
Carnegie Mellon University

Clustering & Topic Models

clustering



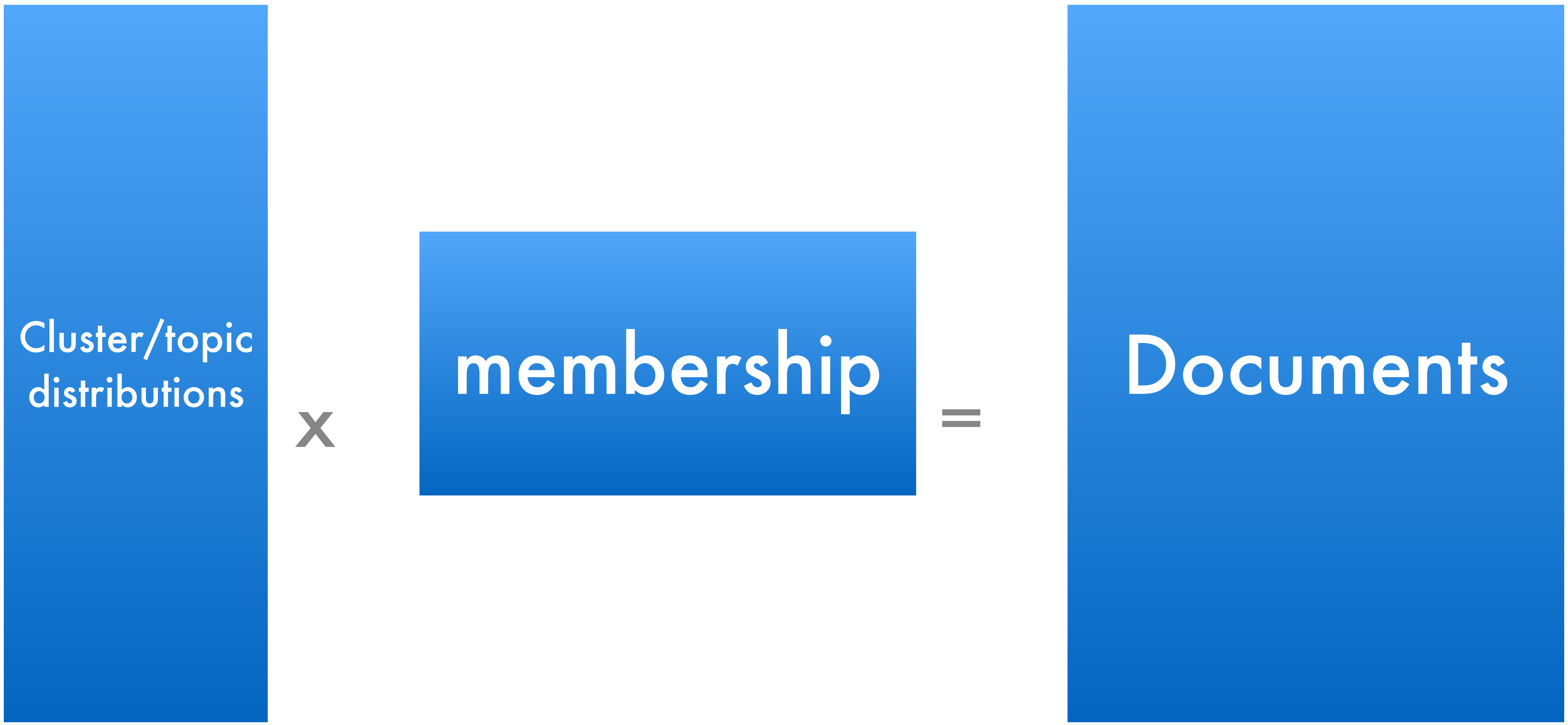
Latent Dirichlet Allocation



Google

Carnegie Mellon University

Clustering & Topic Models



clustering: (0, 1) matrix

topic model: stochastic matrix

LSI: arbitrary matrices

Topics in text

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. “Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services,” Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center’s share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

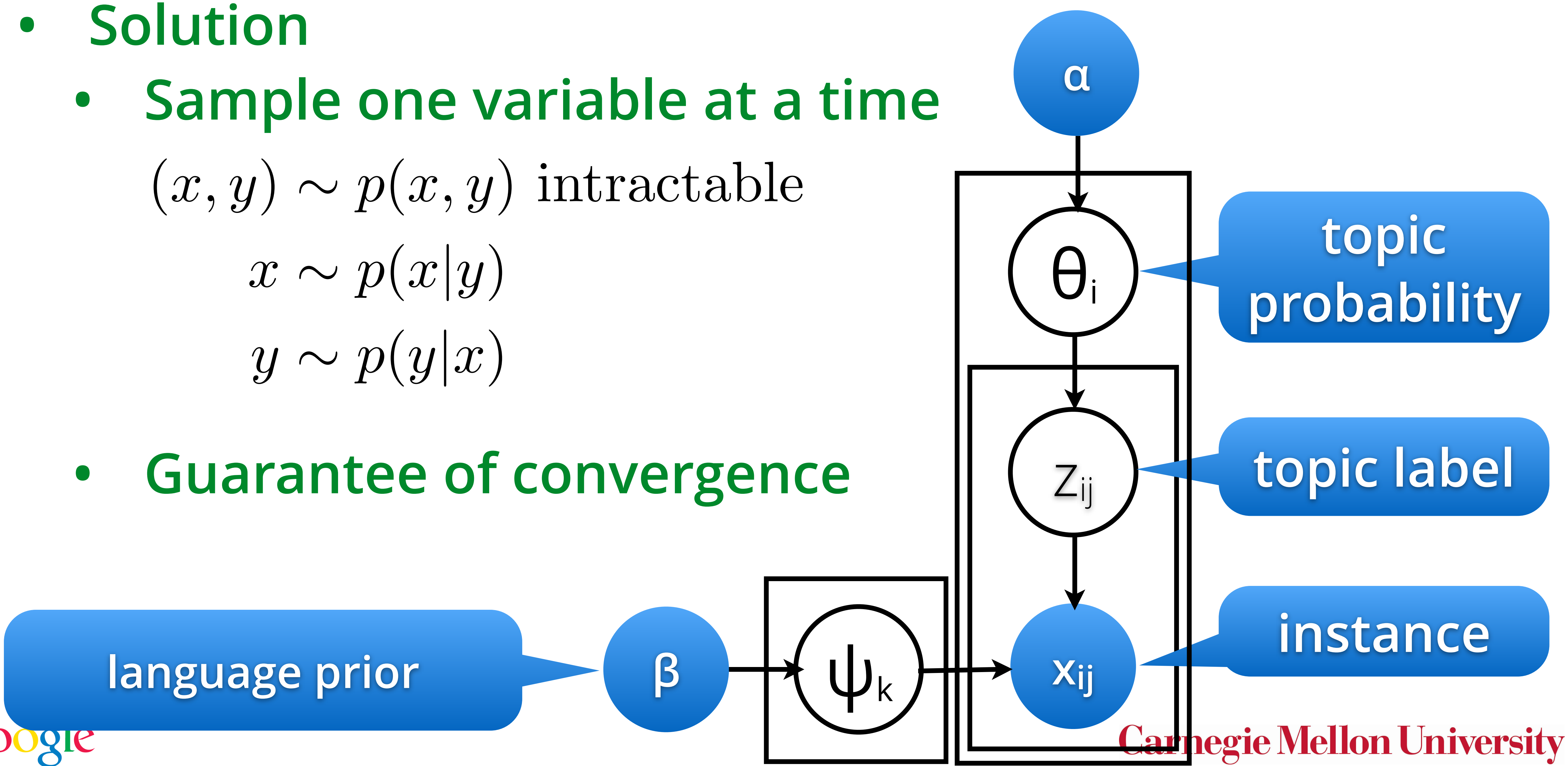
Latent Dirichlet Allocation; Blei, Ng, Jordan, JMLR 2003

Gibbs Sampling

- Goal - sample topics and language model
- Problem - joint distribution intractable
- Solution
  - Sample one variable at a time

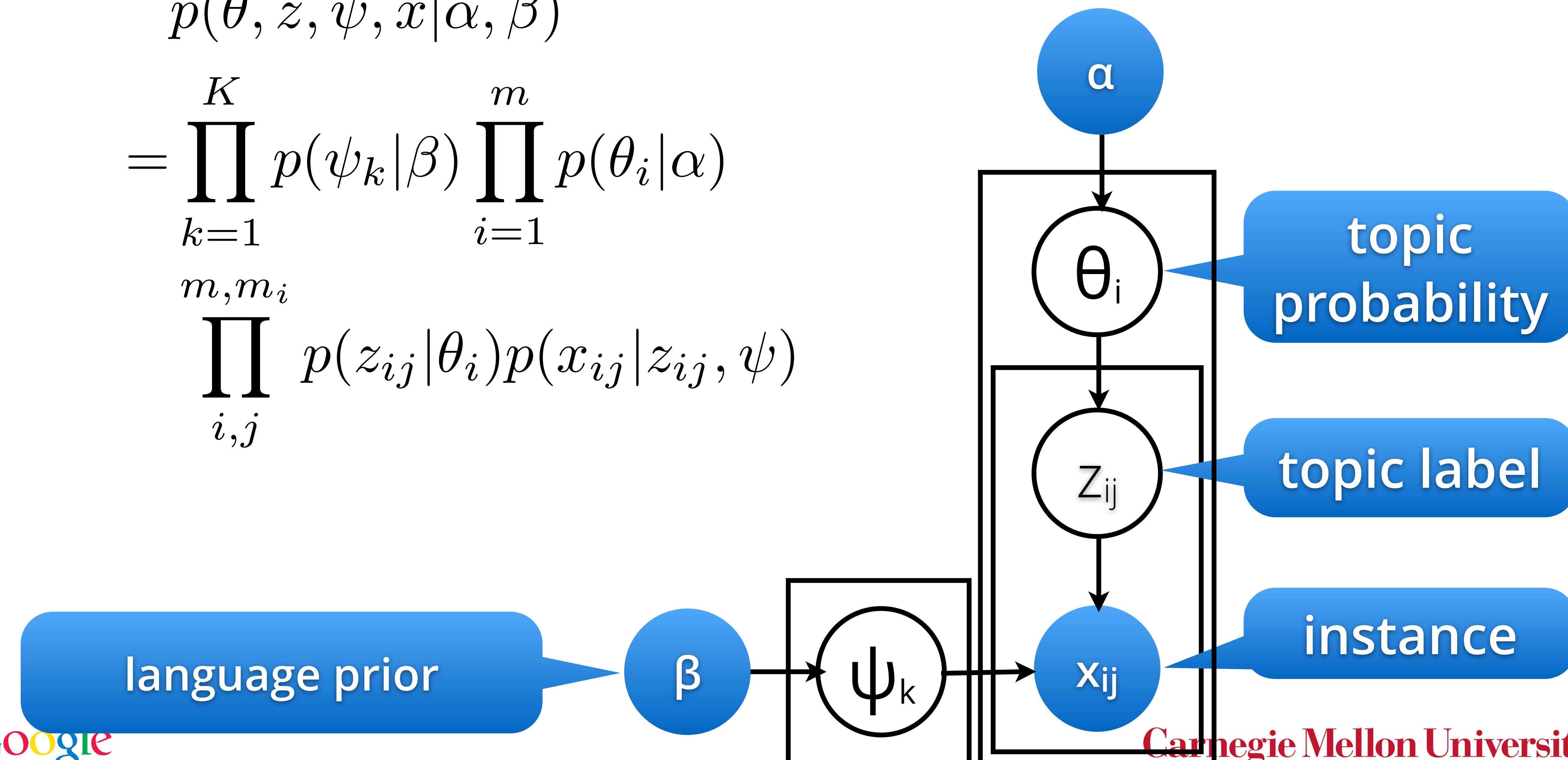
$$(x, y) \sim p(x, y) \text{ intractable}$$
$$x \sim p(x|y)$$
$$y \sim p(y|x)$$

  - Guarantee of convergence



Joint Probability Distribution

$$p(\theta, z, \psi, x | \alpha, \beta) = \prod_{k=1}^{K} p(\psi_k | \beta) \prod_{i=1}^{m} p(\theta_i | \alpha) \prod_{i,j}^{m,m_i} p(z_{ij} | \theta_i) p(x_{ij} | z_{ij}, \psi)$$



Joint Probability Distribution

sample Ψ independently

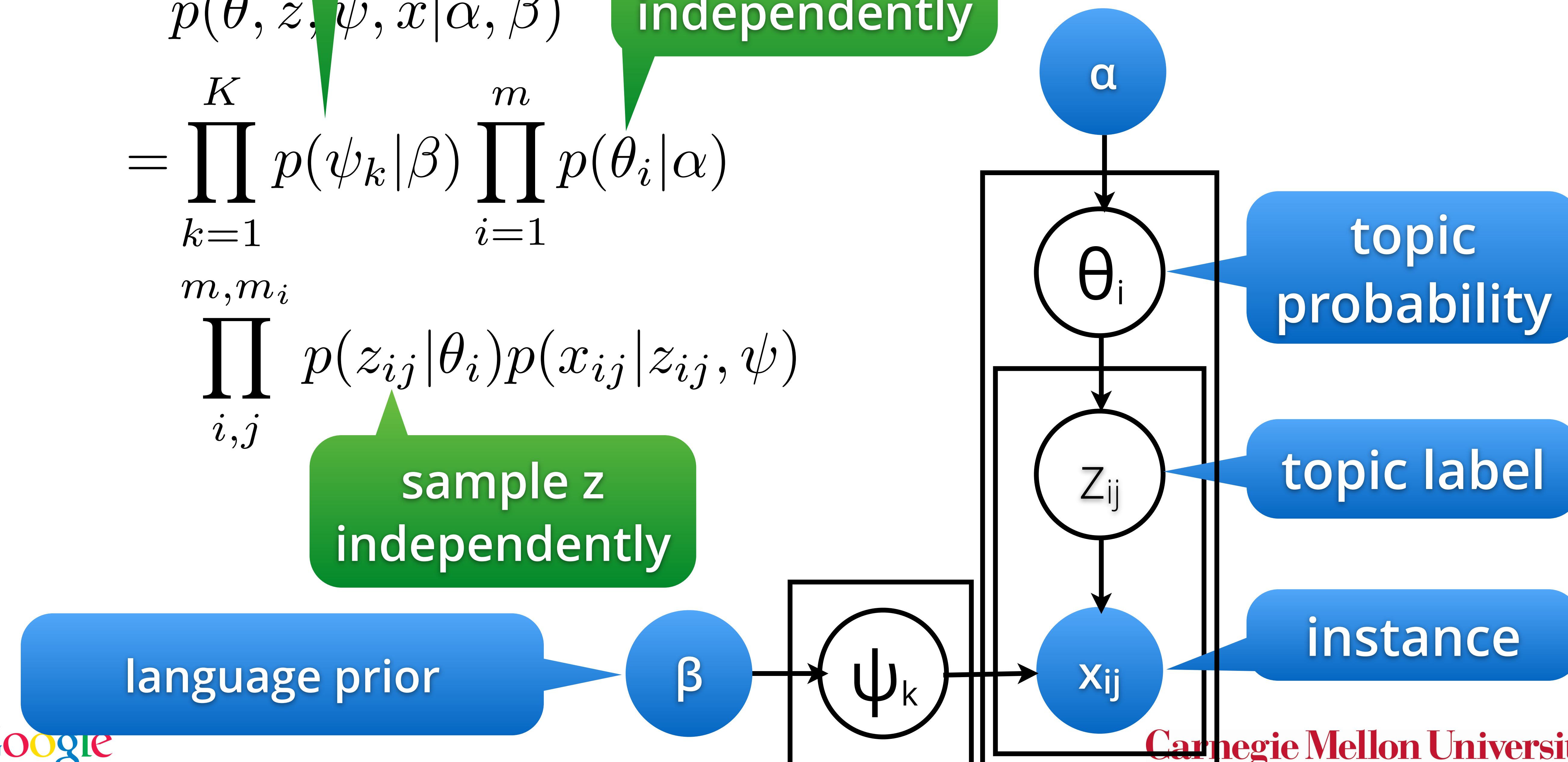
sample θ independently

$$p(\theta, z, \psi, x|\alpha, \beta)$$

$$= \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{i=1}^{m} p(\theta_i|\alpha)$$

$$\prod_{i,j}^{m,m_i} p(z_{ij}|\theta_i) p(x_{ij}|z_{ij}, \psi)$$

sample z independently



Joint Probability Distribution

sample Ψ independently

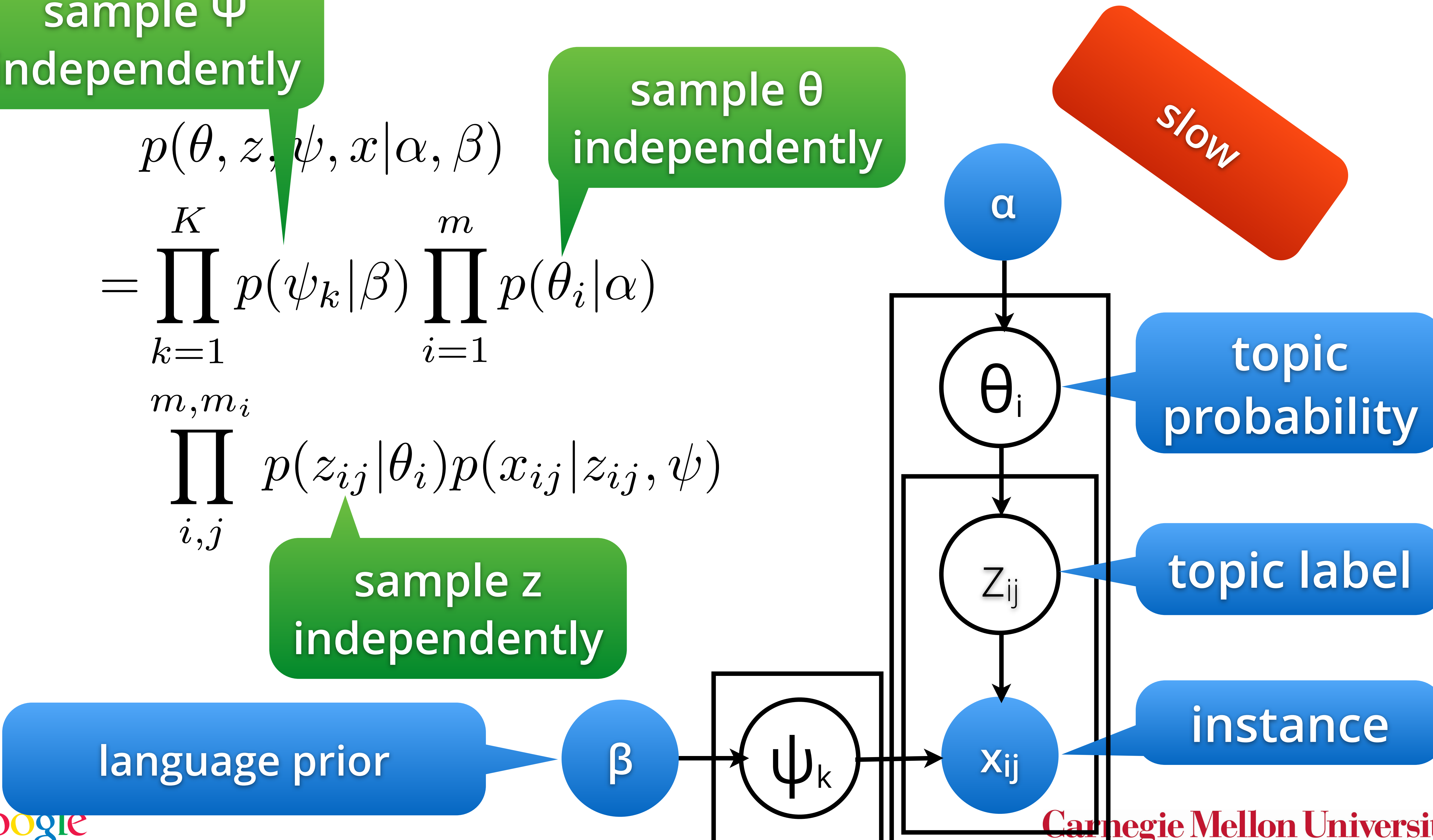
sample θ independently

$$p(\theta, z, \psi, x|\alpha, \beta)$$

$$= \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{i=1}^{m} p(\theta_i|\alpha)$$

$$\prod_{i,j}^{m,m_i} p(z_{ij}|\theta_i)p(x_{ij}|z_{ij}, \psi)$$

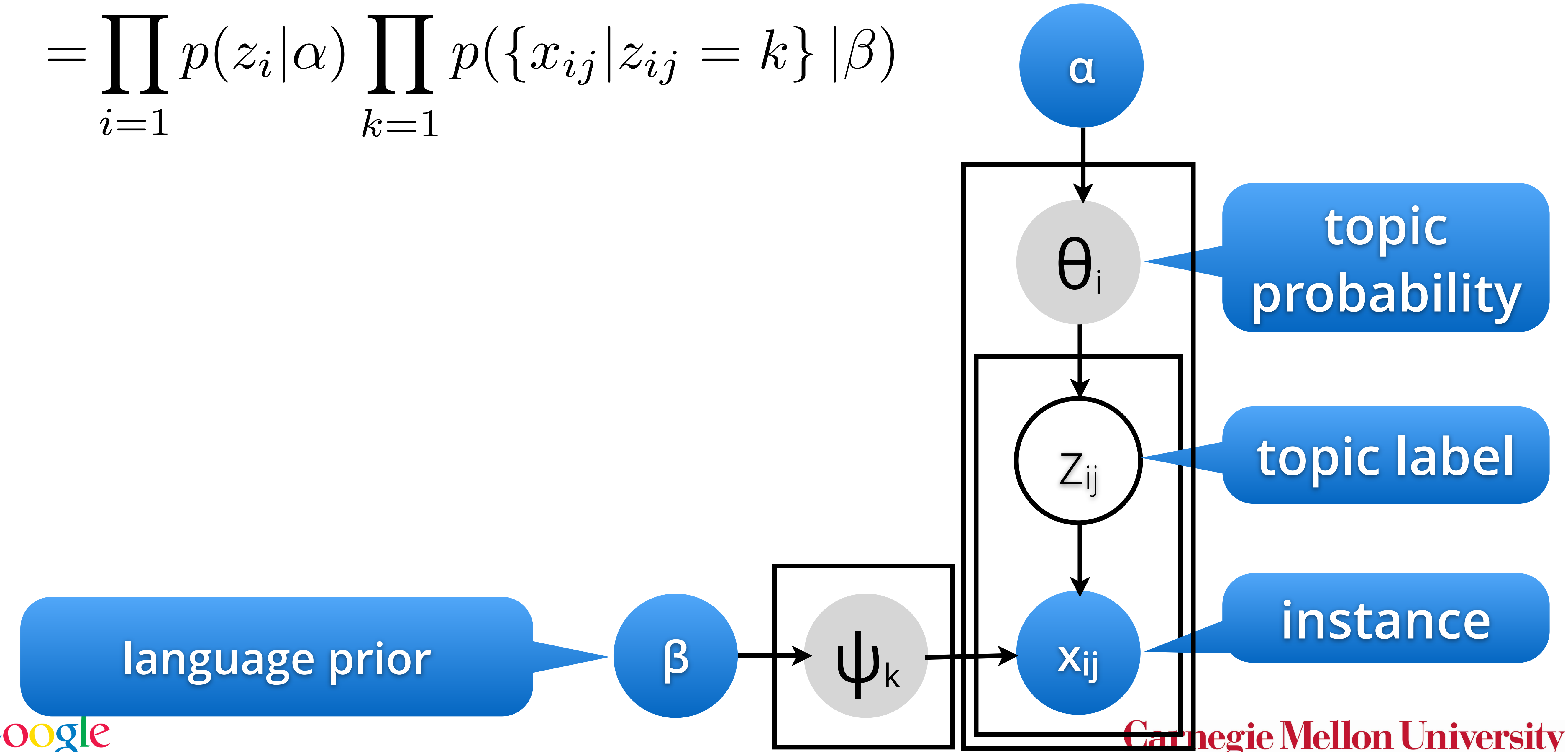
sample z independently



Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\} |\beta)$$

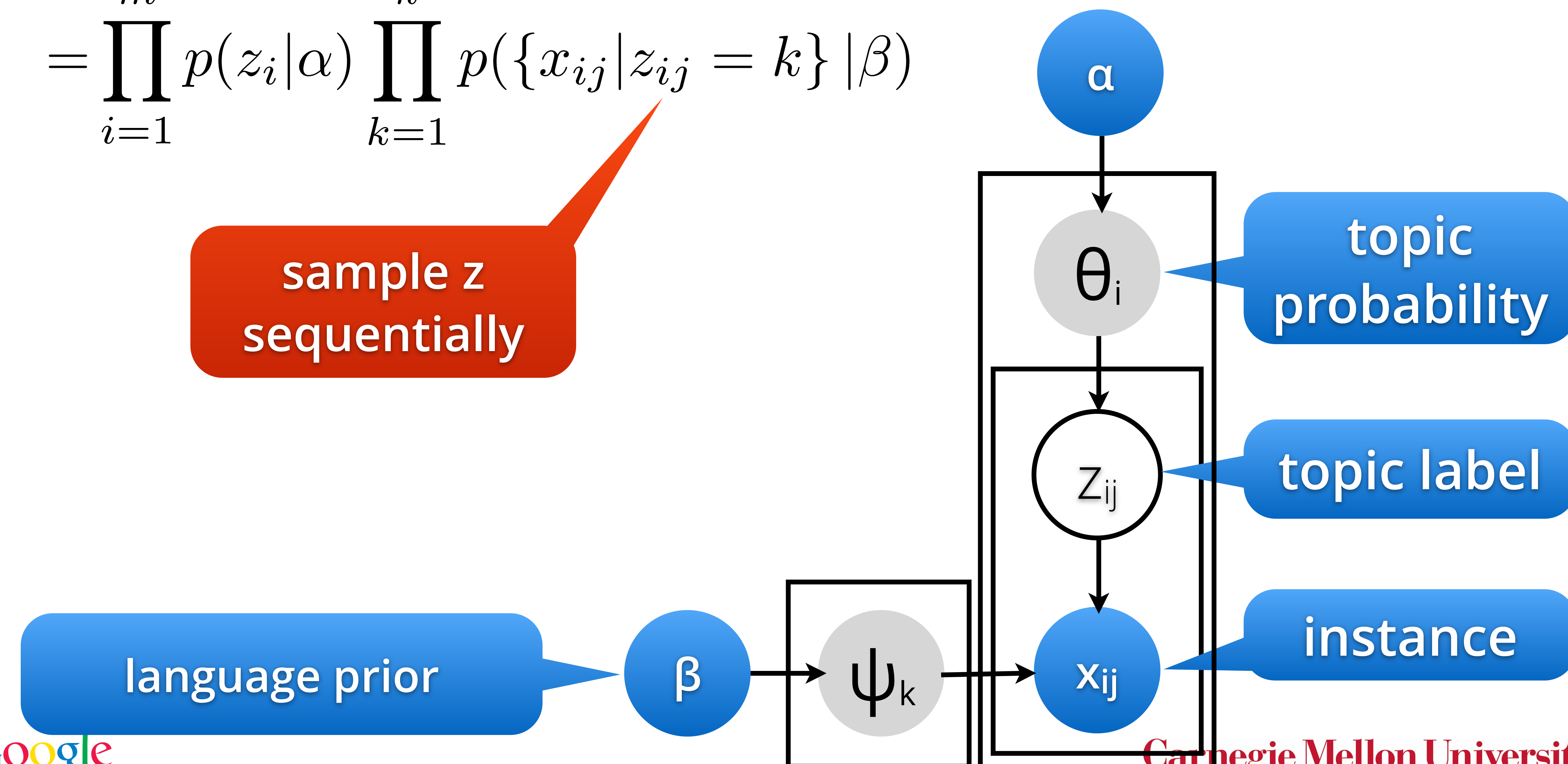


Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\} |\beta)$$

sample z sequentially



Google

Carnegie Mellon University

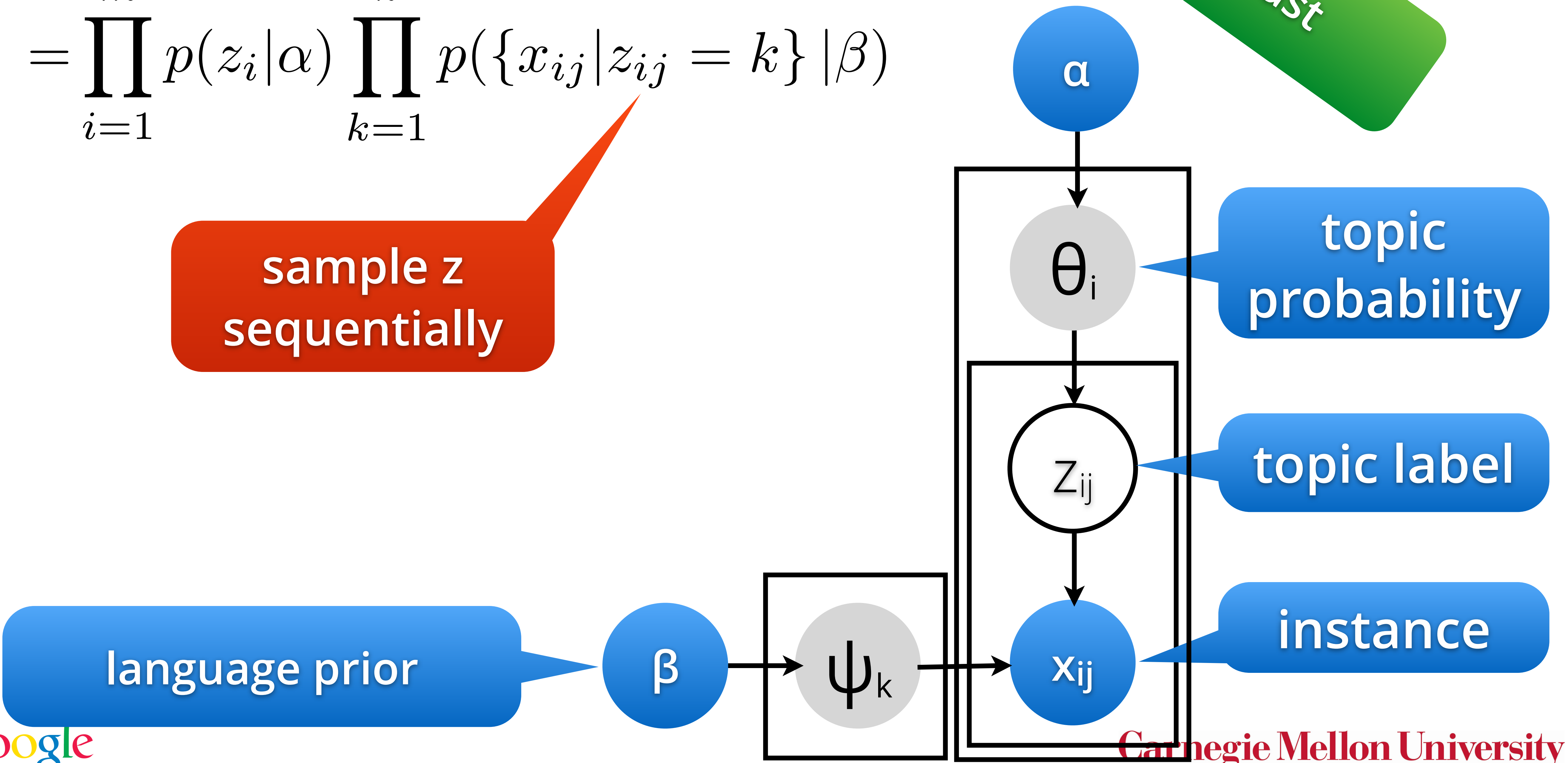
Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}\,|\beta)$$

sample z sequentially

fast



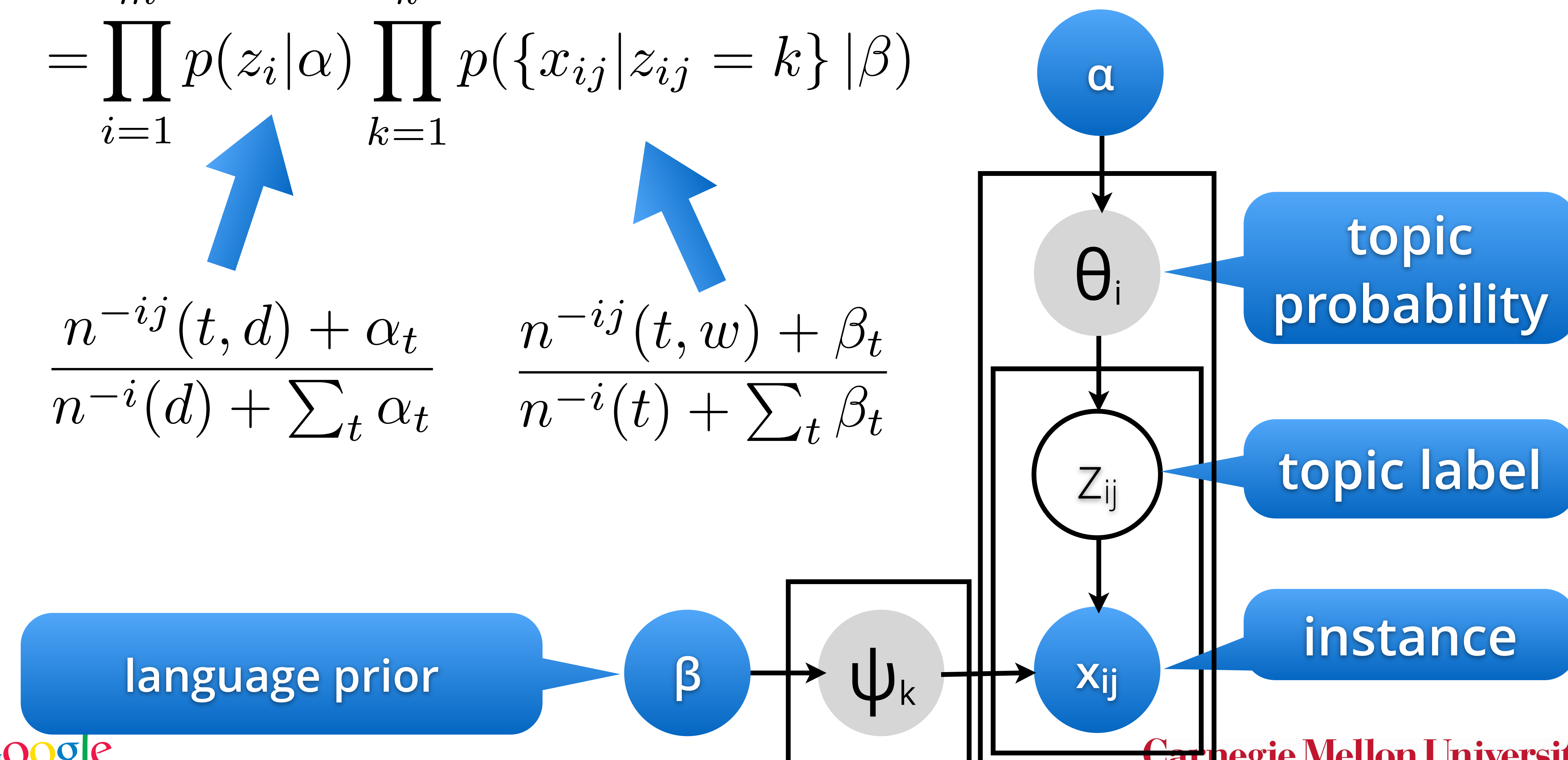
Collapsed Sampler

Griffiths & Steyvers, 2005

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}\,|\beta)$$

$$\frac{n^{-ij}(t,d) + \alpha_t}{n^{-i}(d) + \sum_t \alpha_t} \qquad \frac{n^{-ij}(t,w) + \beta_t}{n^{-i}(t) + \sum_t \beta_t}$$



Collapsed Sampler

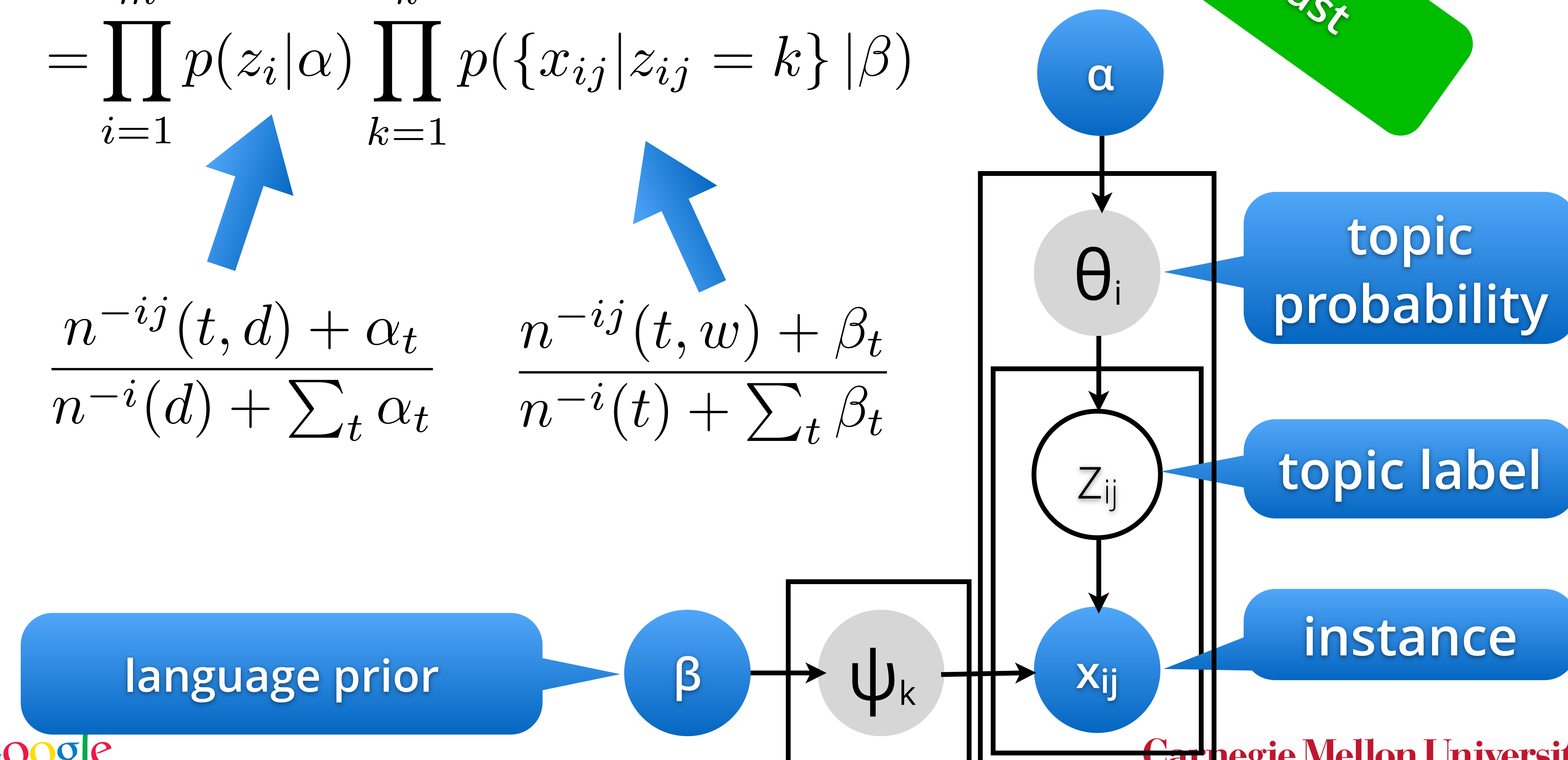
Griffiths & Steyvers, 2005

fast

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}|\beta)$$

$$\frac{n^{-ij}(t,d) + \alpha_t}{n^{-i}(d) + \sum_t \alpha_t} \qquad \frac{n^{-ij}(t,w) + \beta_t}{n^{-i}(t) + \sum_t \beta_t}$$



Gibbs Sampler

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Lock (word,topic) table
      - Update local (document, topic) table
      - Update (word,topic) table
      - Unlock (word,topic) table

this kills parallelism

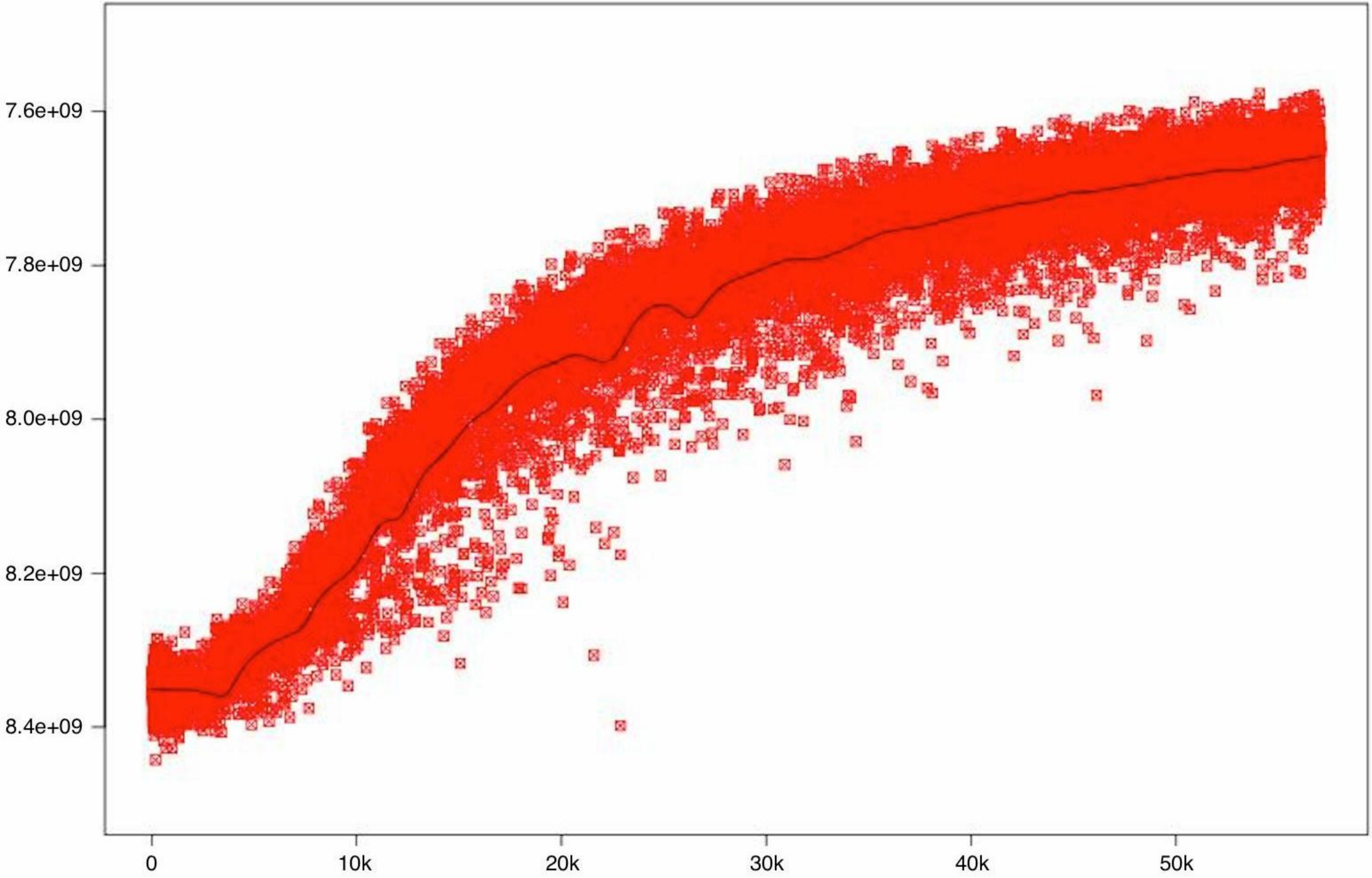
Gibbs Sampler

- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Lock local (word,topic) table
      - Update local (document, topic) table
      - Update local (word,topic) table
      - Unlock local (word,topic) table
    - Synchronize local and global tables

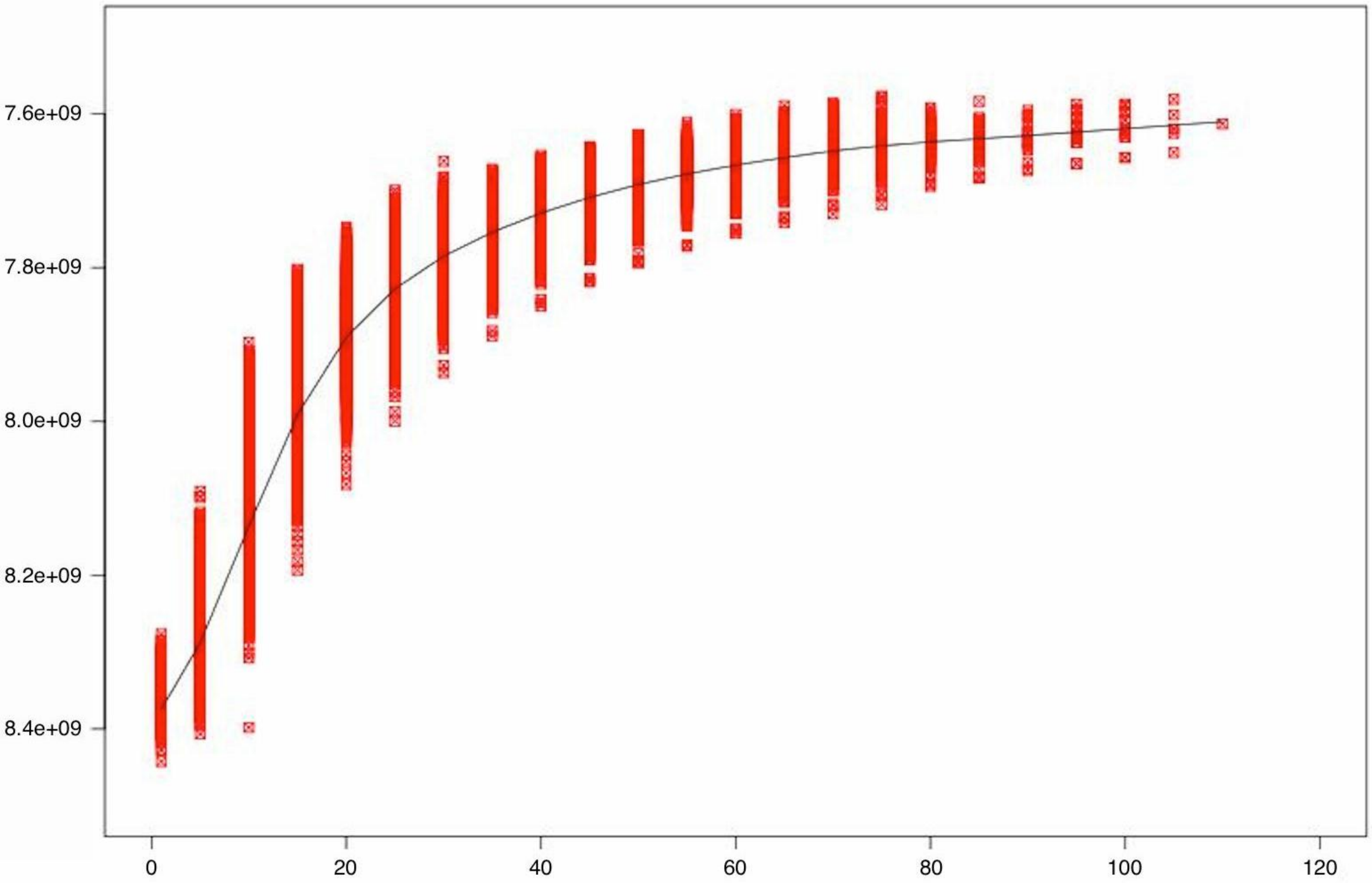
this kills multithreading

Gibbs Sampler

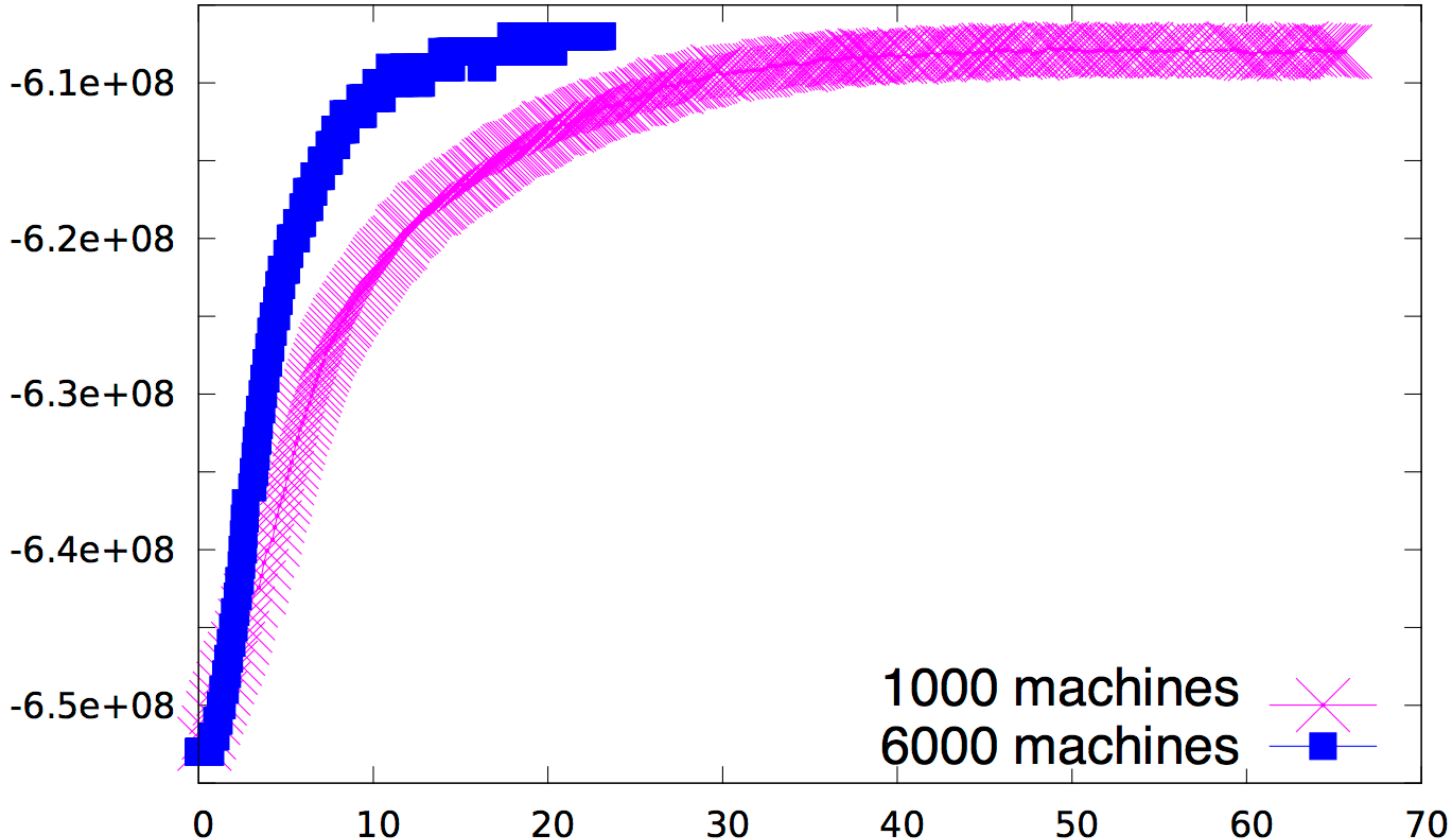
- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Update local (document, topic) table
      - Generate local update message
    - Update local table
      - Lock local (word,topic) table
      - Update local (word,topic) table
      - Unlock local (word,topic) table
    - Synchronize local and global tables



Log-Likelihood distribution as a function of runtime (s) for workers

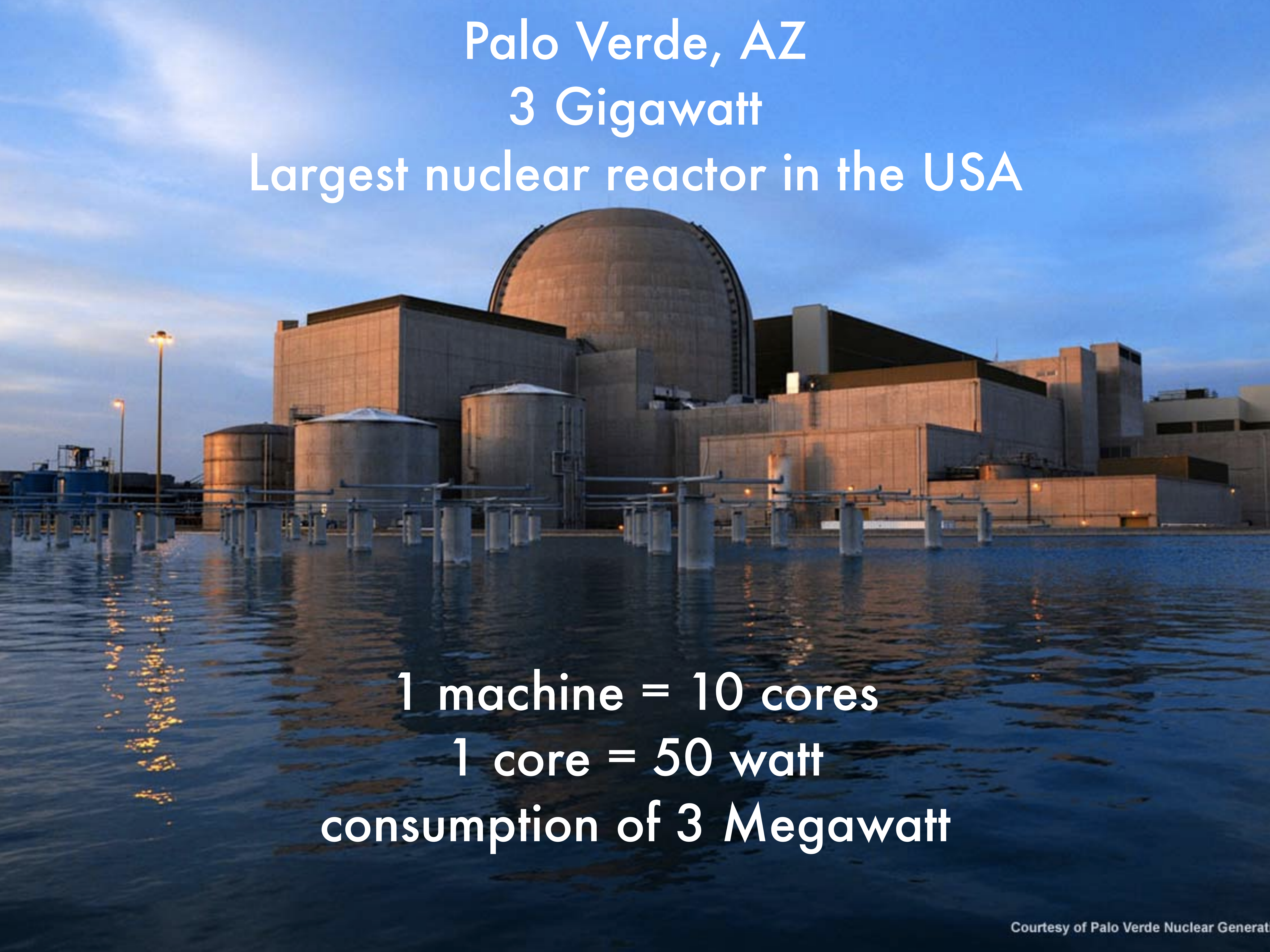


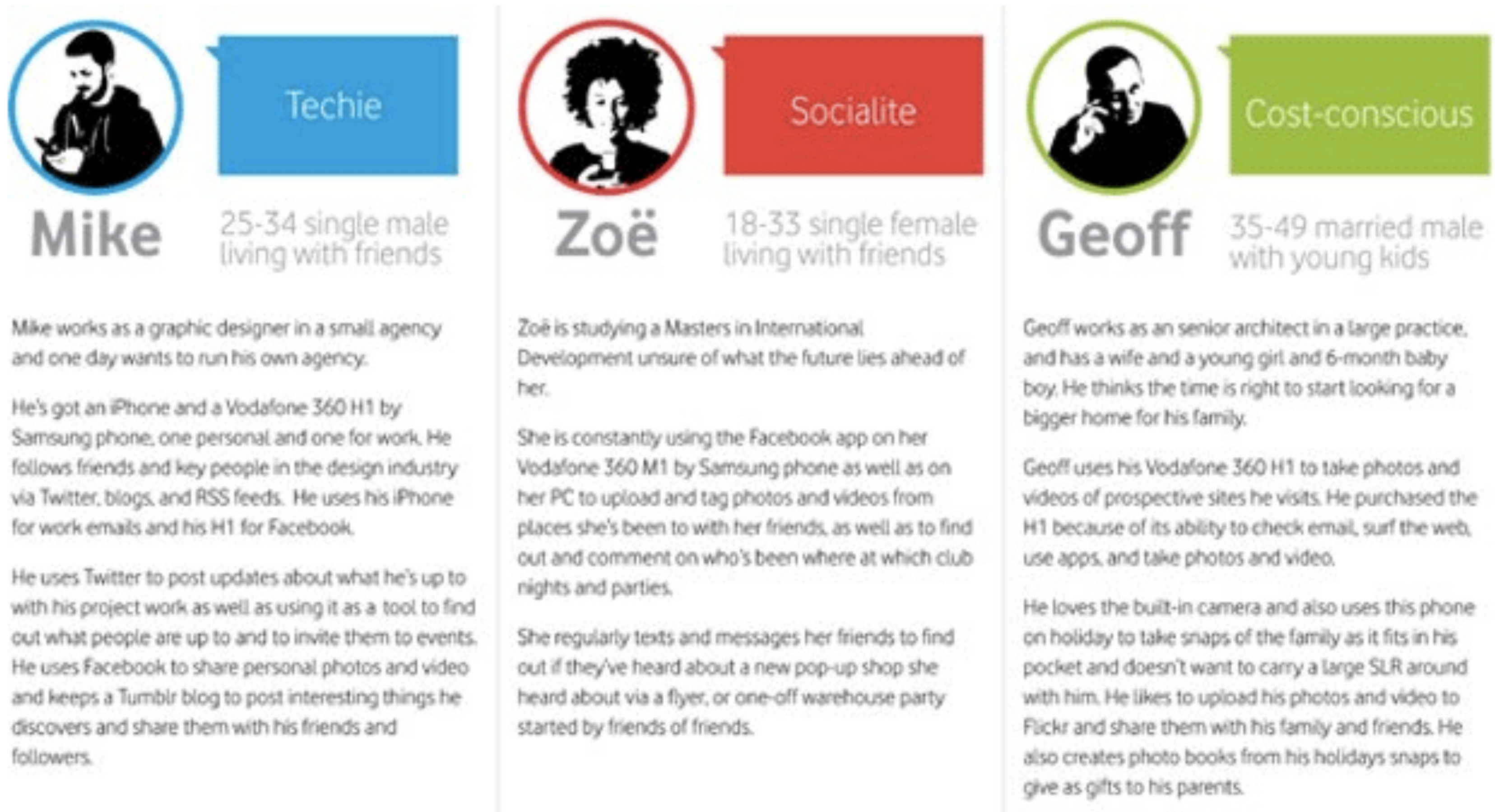
Log-Likelihood distribution as a function of iteration count for workers



-6.1e+08
-6.2e+08
-6.3e+08
-6.4e+08
-6.5e+08
0
10
20
30
40
50
60
70
1000 machines
6000 machines


Palo Verde, AZ
3 Gigawatt
Largest nuclear reactor in the USA
Courtesy of Palo Verde Nuclear Generat


Palo Verde, AZ
3 Gigawatt
Largest nuclear reactor in the USA
1 machine = 10 cores
1 core = 50 watt
consumption of 3 Megawatt
Courtesy of Palo Verde Nuclear



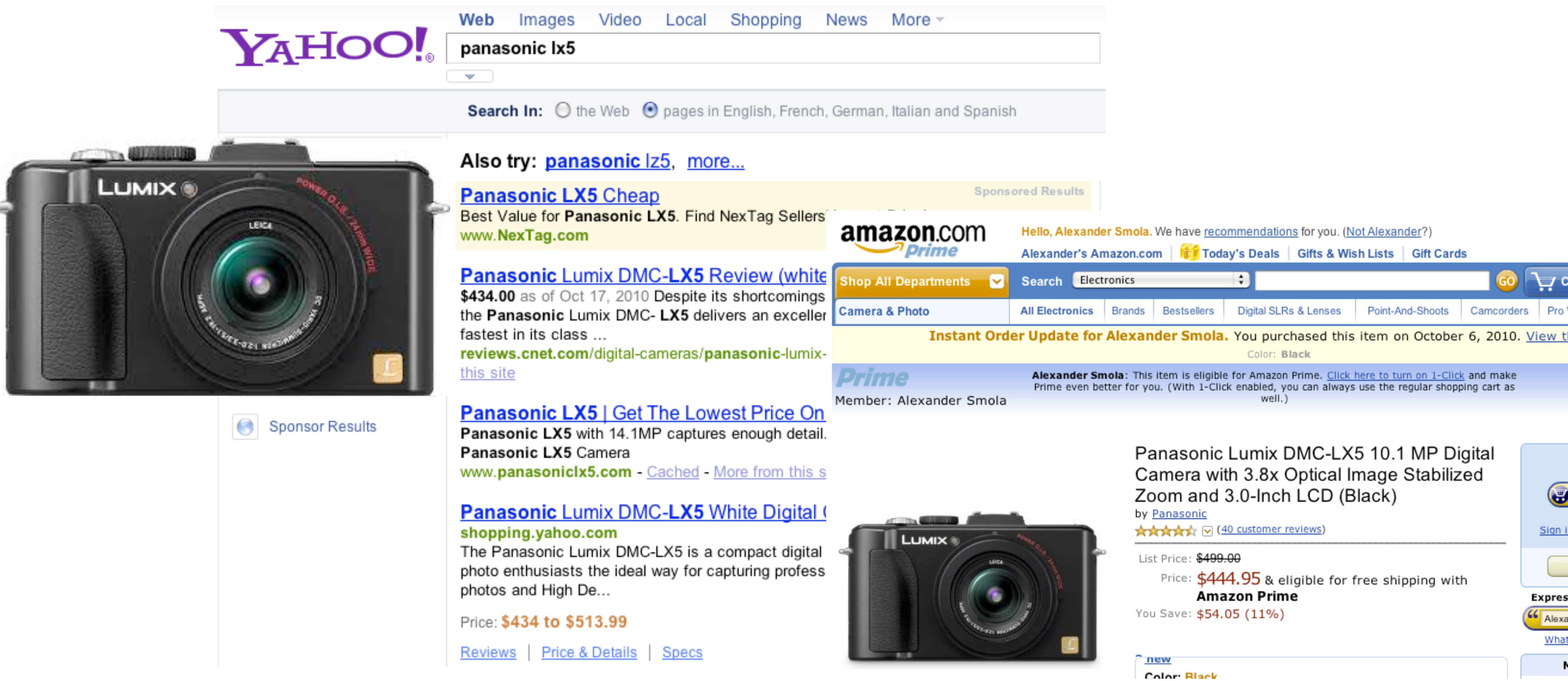
User Profiling

User Profiling

Buying a Camera

time

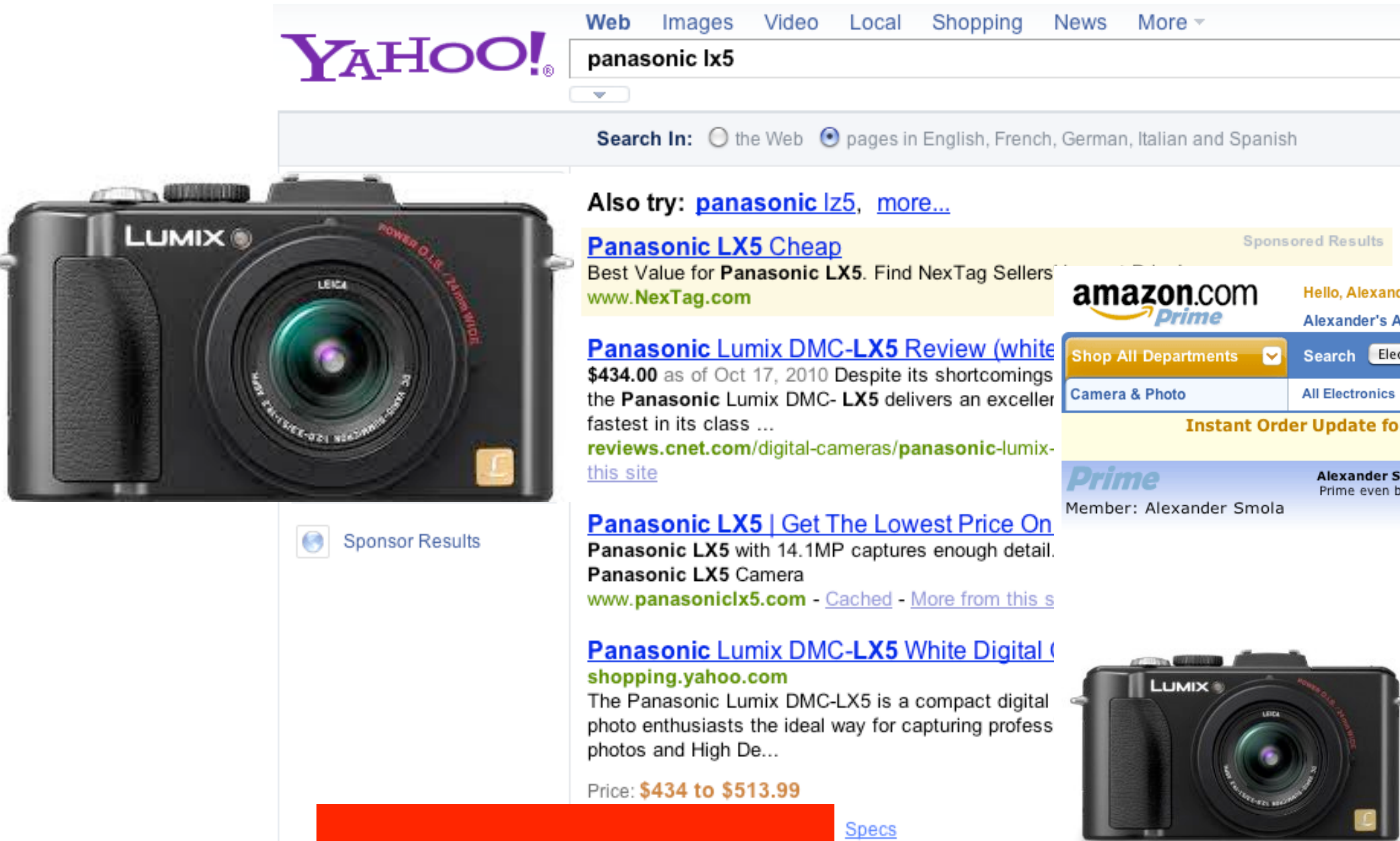
Buying a Camera

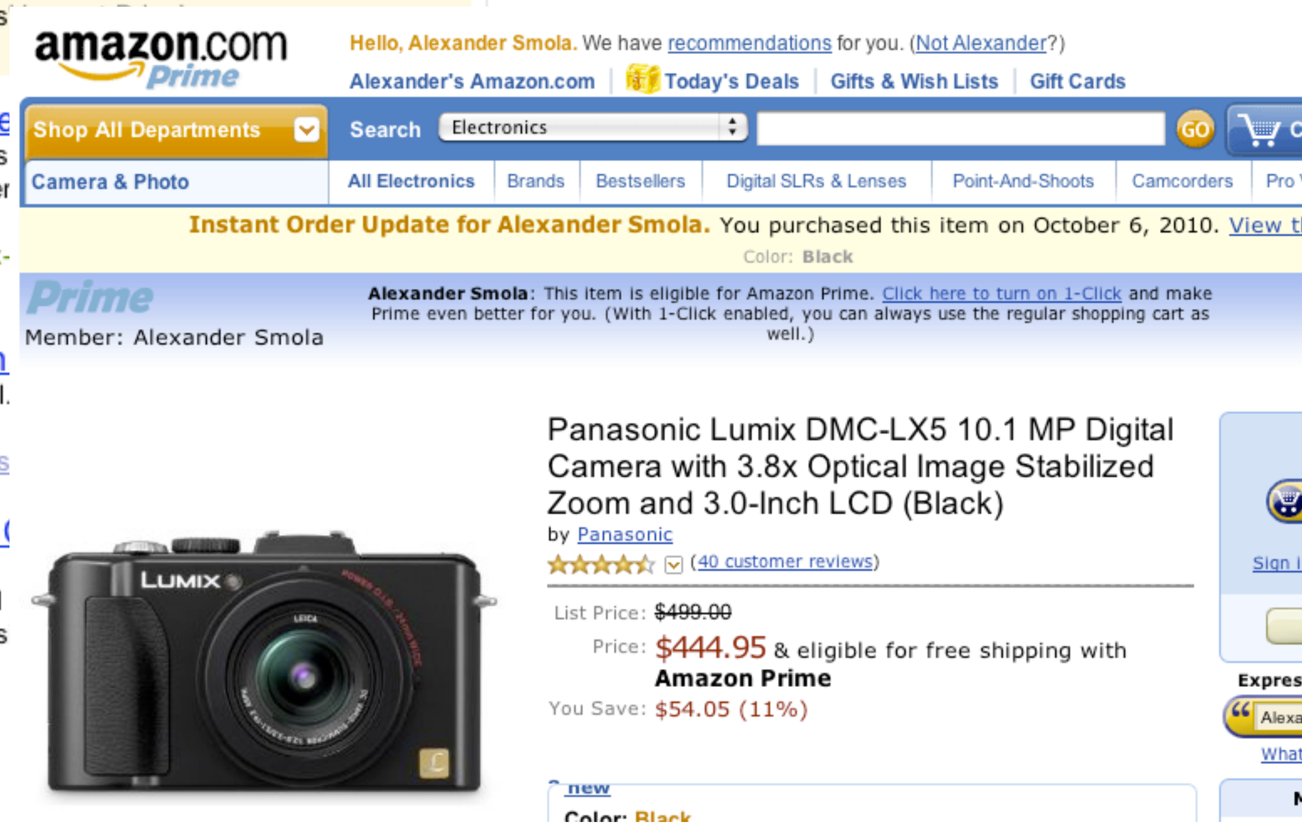


Google

Carnegie Mellon University

Buying a Camera





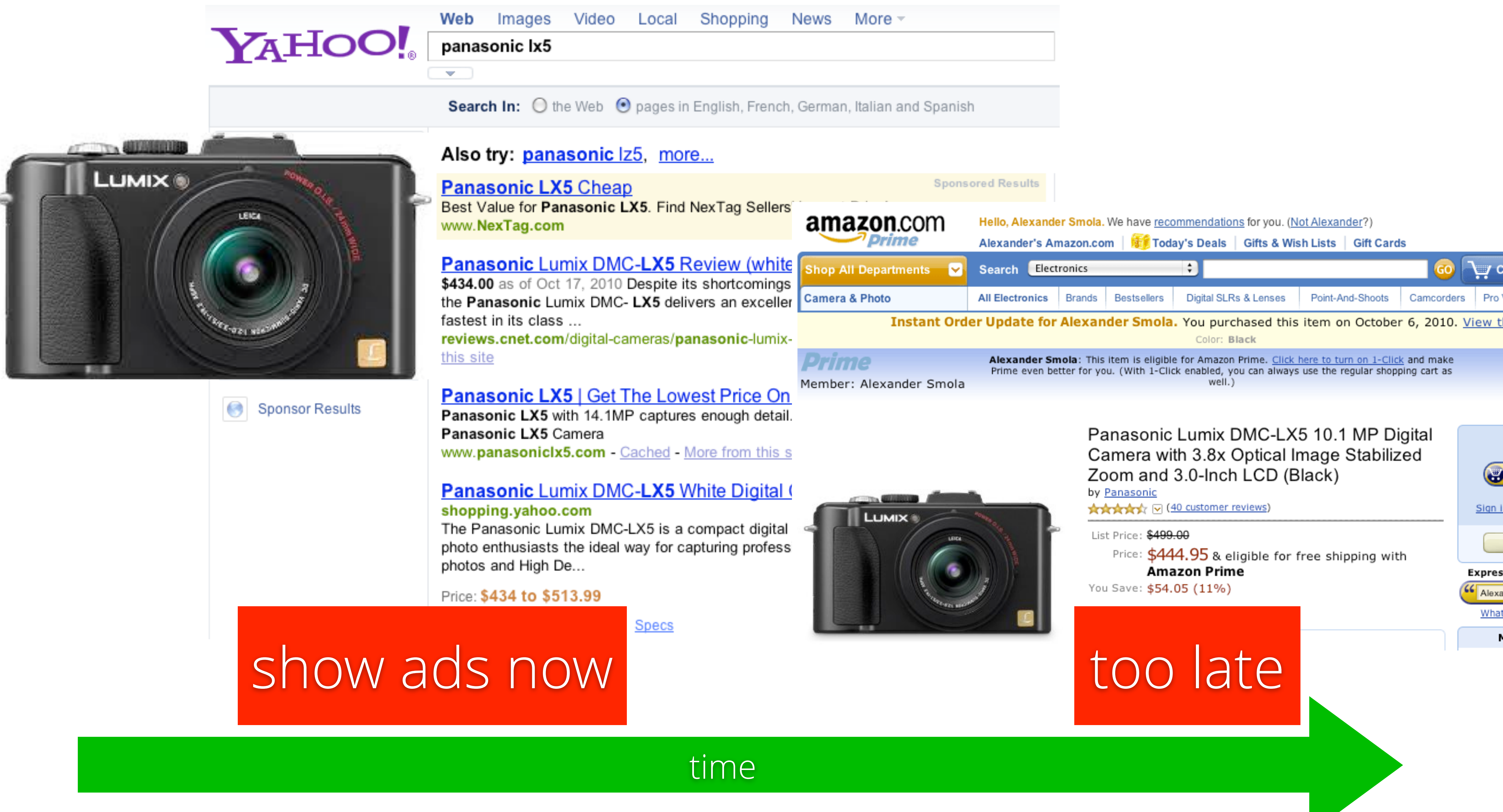
show ads now

time

Google

Carnegie Mellon University

Buying a Camera

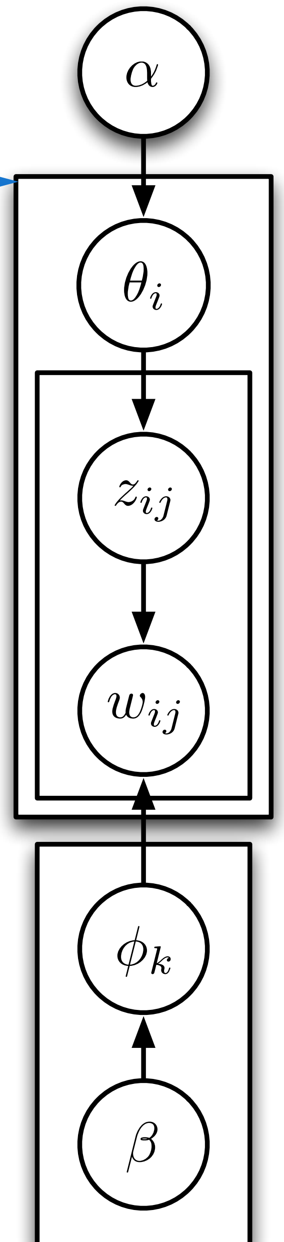


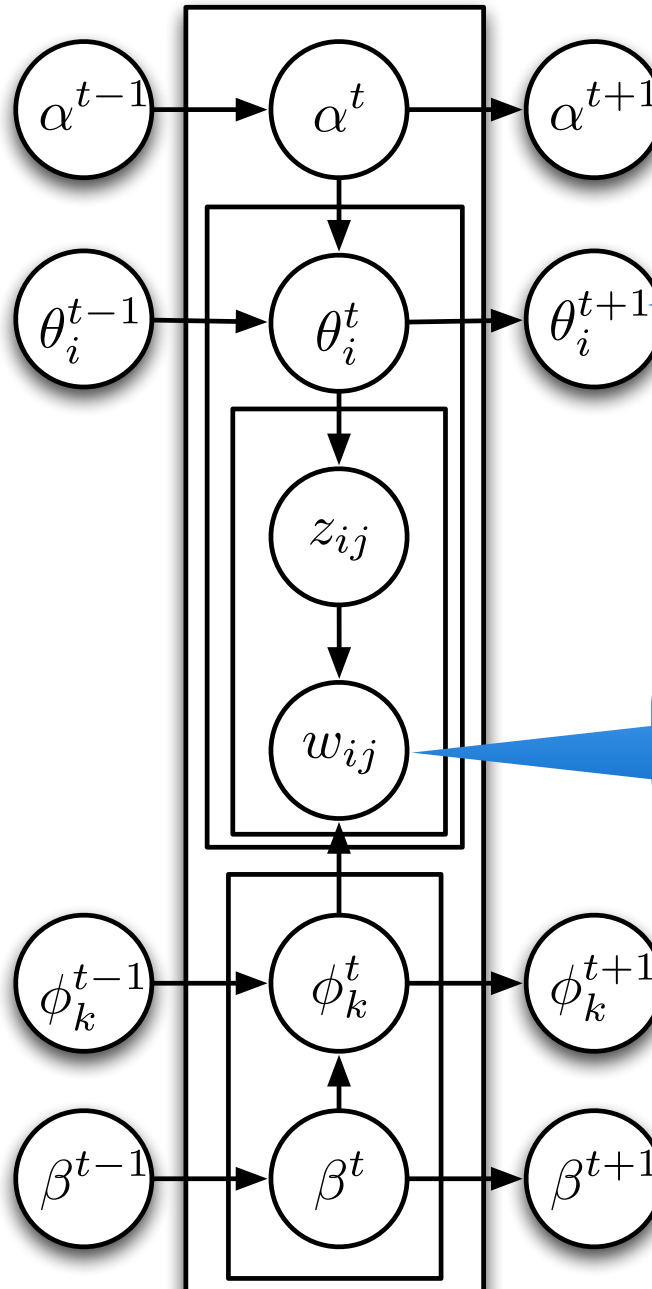
Google

Carnegie Mellon University

Statistical Model

plain LDA





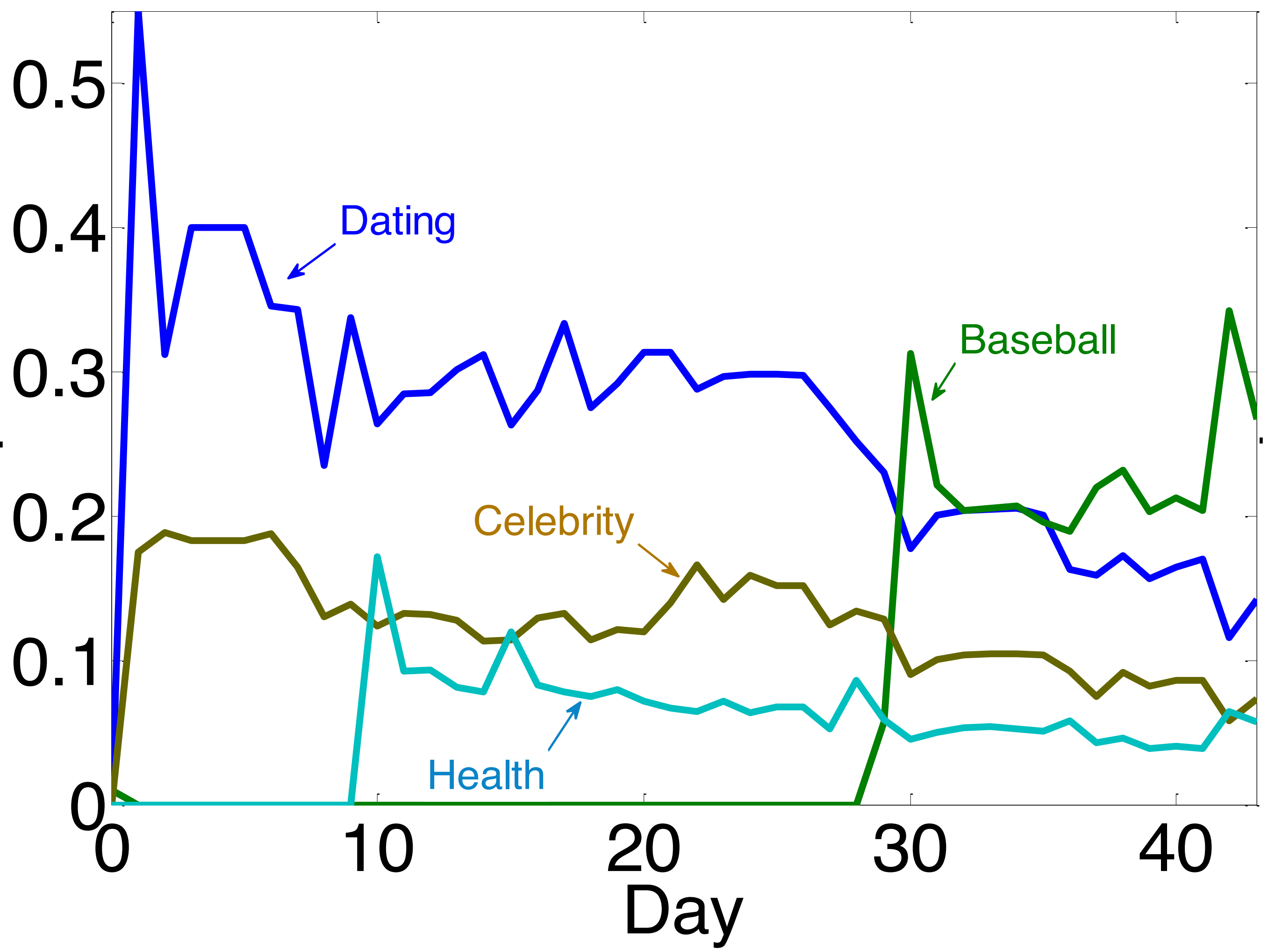
time dependent

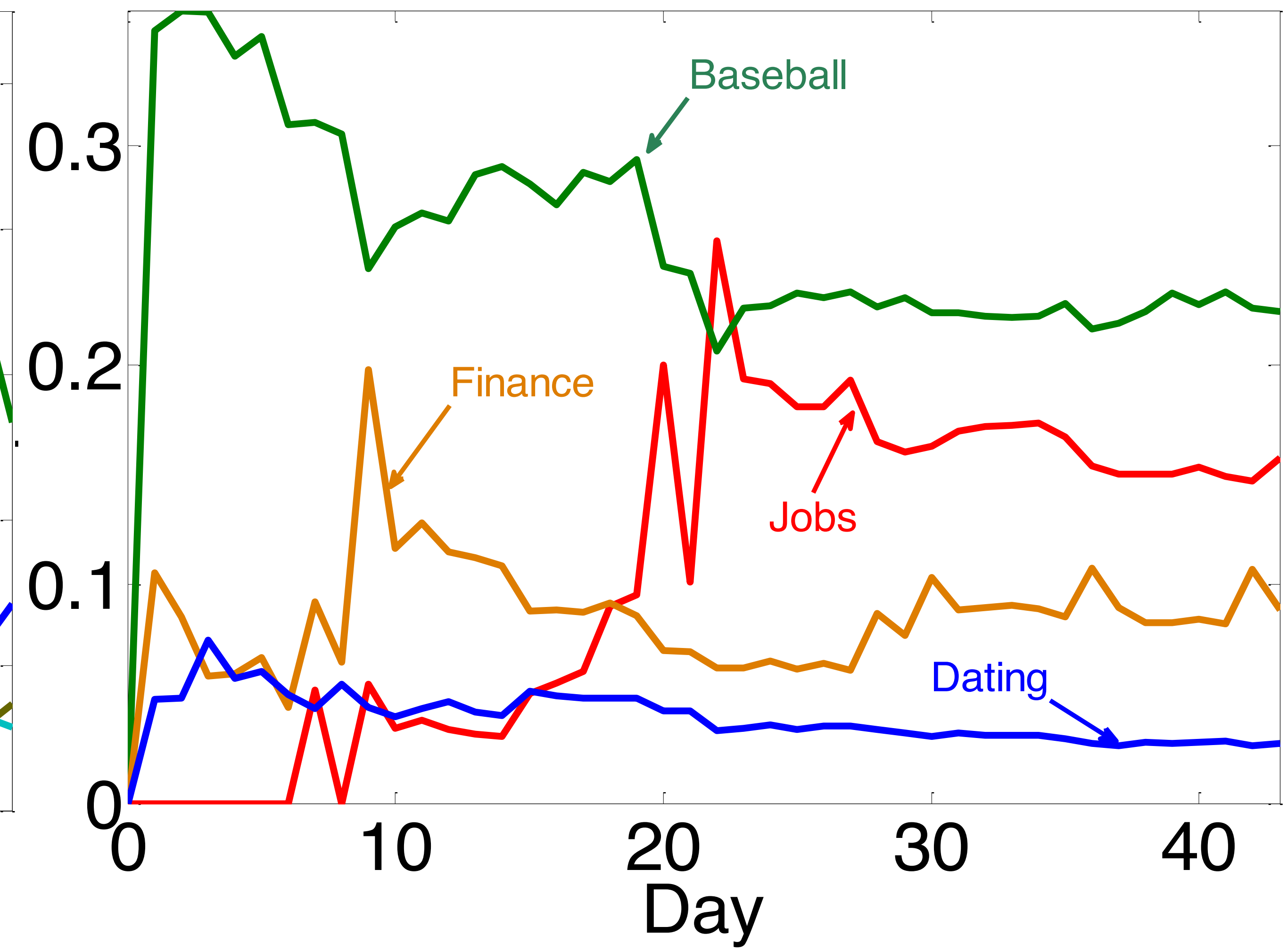
user

actions per topic

- Topic model
  - Users - Documents
  - Actions - Words
  - Interests - Topics
- Παντα ρει (everything flows)
  - Users' interest preferences change over time
  - Interests change over time
  - Changing flavor of the day

Some Users





Dating

women
men
dating
singles
personals
seeking
match

Baseball

League
baseball
basketball,
doublehead
Bergesen
Griffey
bullpen
Greinke

Celebrity

Snooki
Tom
Cruise
Katie
Holmes
Pinkett
Kudrow
Hollywood

Health

skin
body
fingers
cells
toes
wrinkle
layers

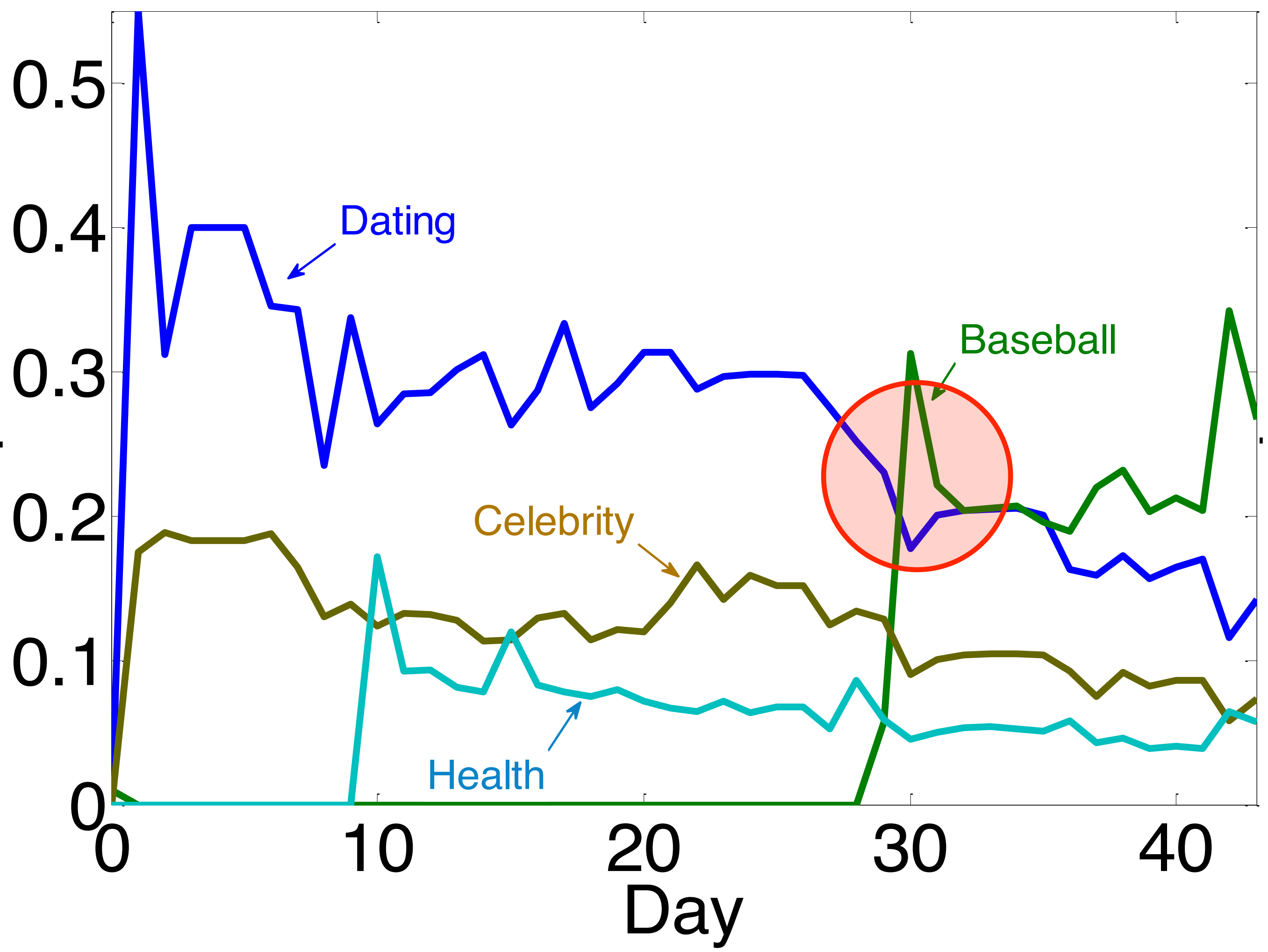
Jobs

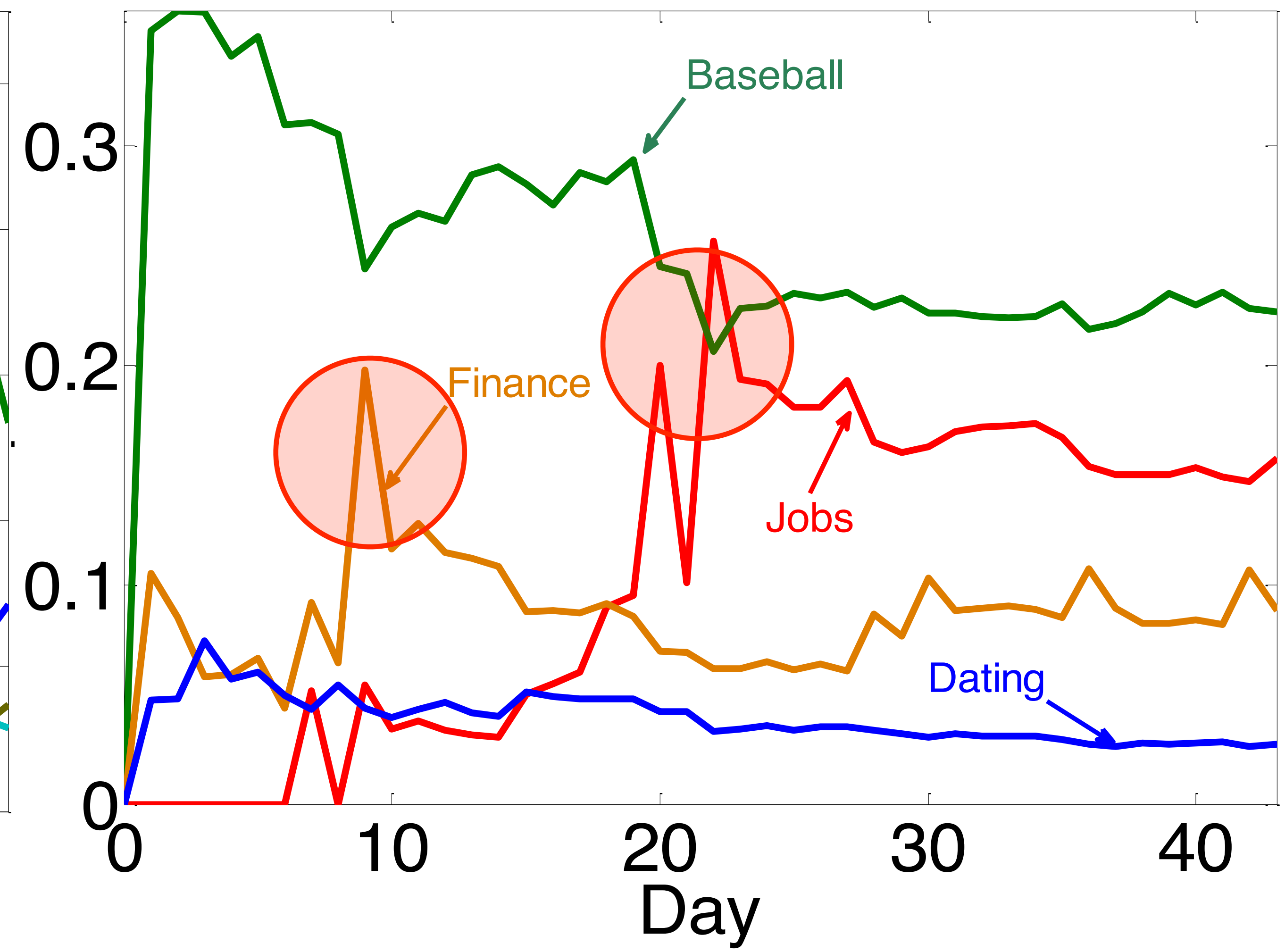
job
career
business
assistant
hiring
part-time
receptionist

Finance

financial
Thomson
chart
real
Stock
Trading
currency

Some Users





Dating

women
men
dating
singles
personals
seeking
match

Baseball

League
baseball
basketball,
doublehead
Bergesen
Griffey
bullpen
Greinke

Celebrity

Snooki
Tom
Cruise
Katie
Holmes
Pinkett
Kudrow
Hollywood

Health

skin
body
fingers
cells
toes
wrinkle
layers

Jobs

job
career
business
assistant
hiring
part-time
receptionist

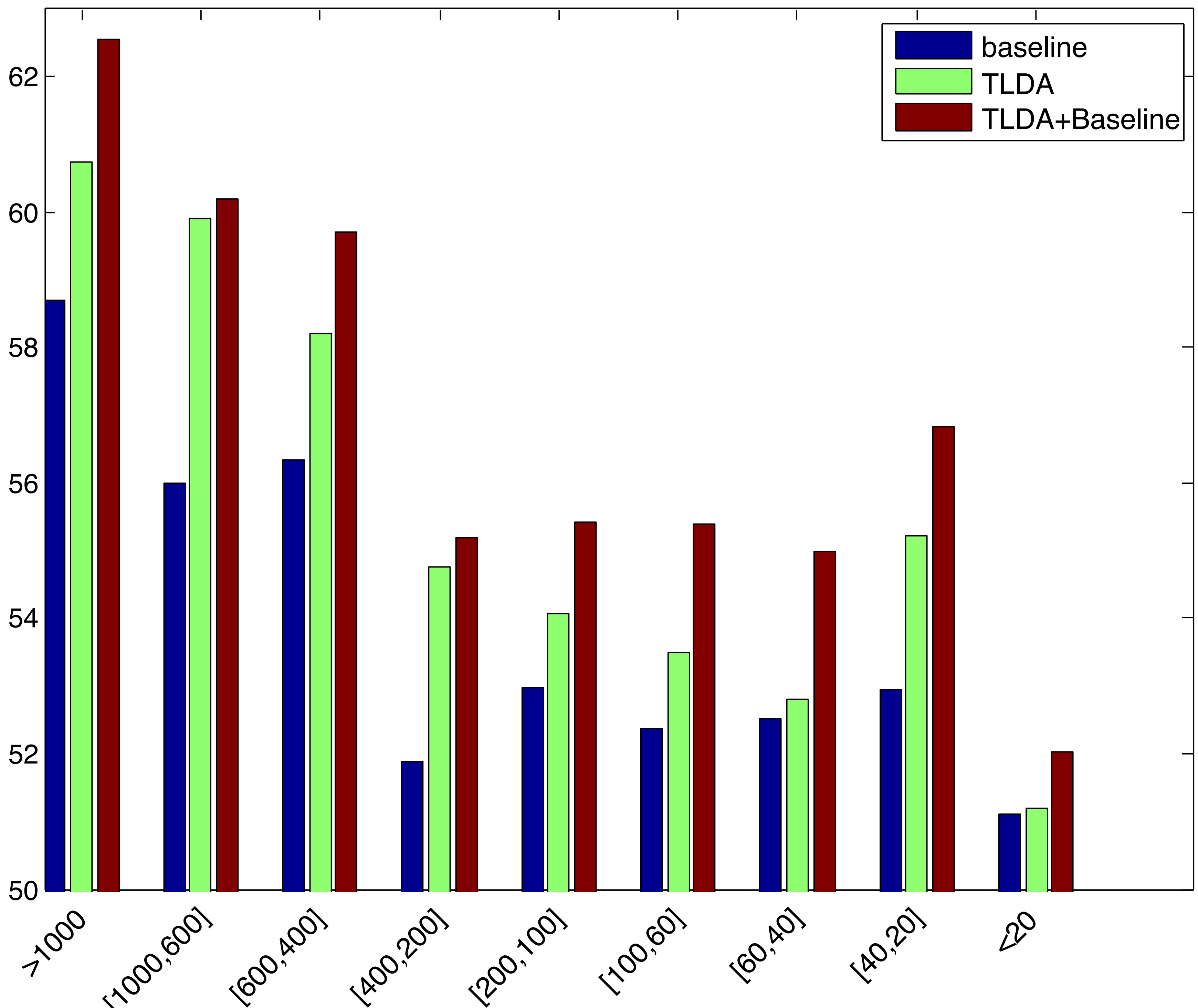
Finance

financial
Thomson
chart
real
Stock
Trading
currency

Google

University

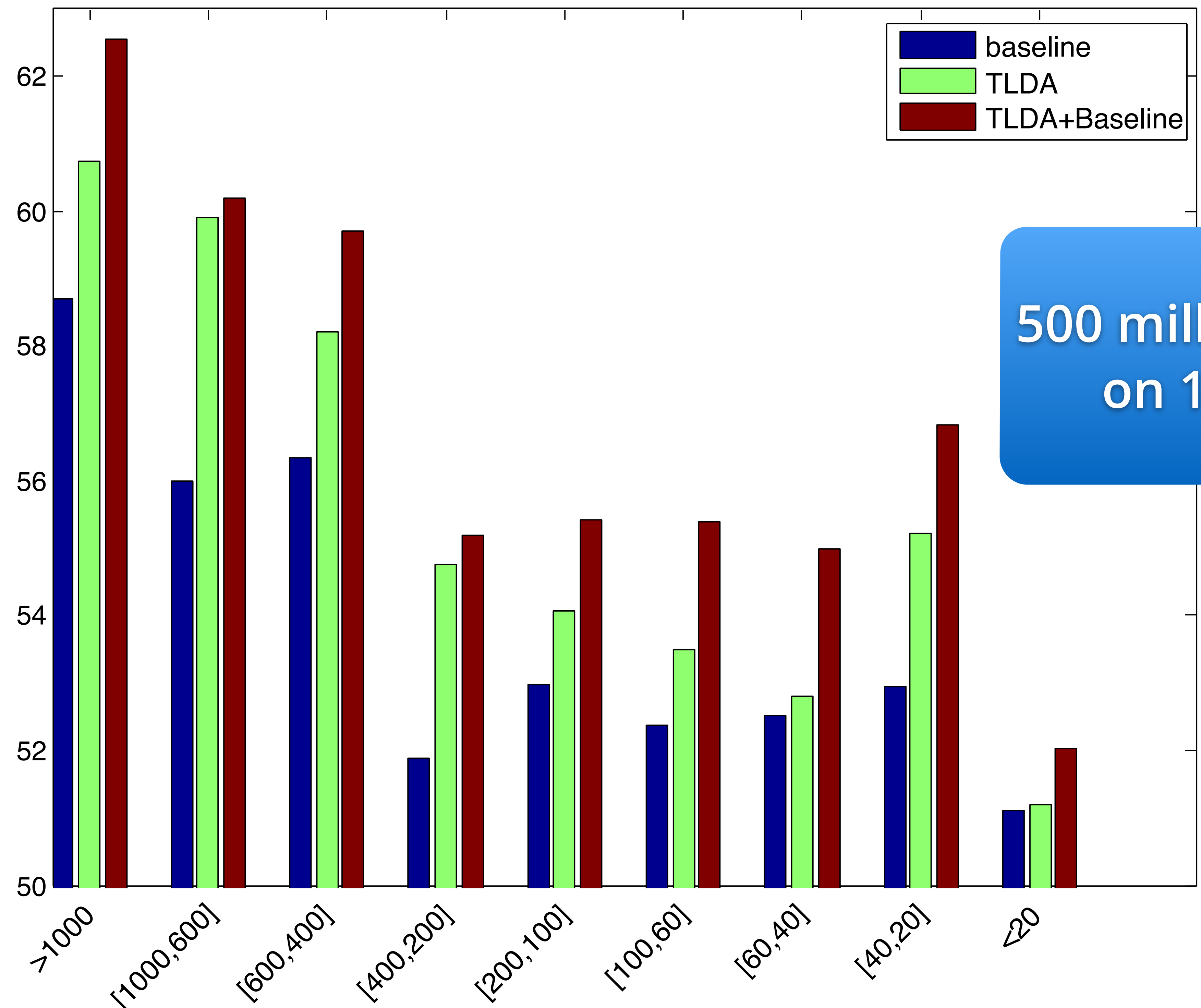
Improvement ($$$)



Google

Carnegie Mellon University

Improvement ($$$)

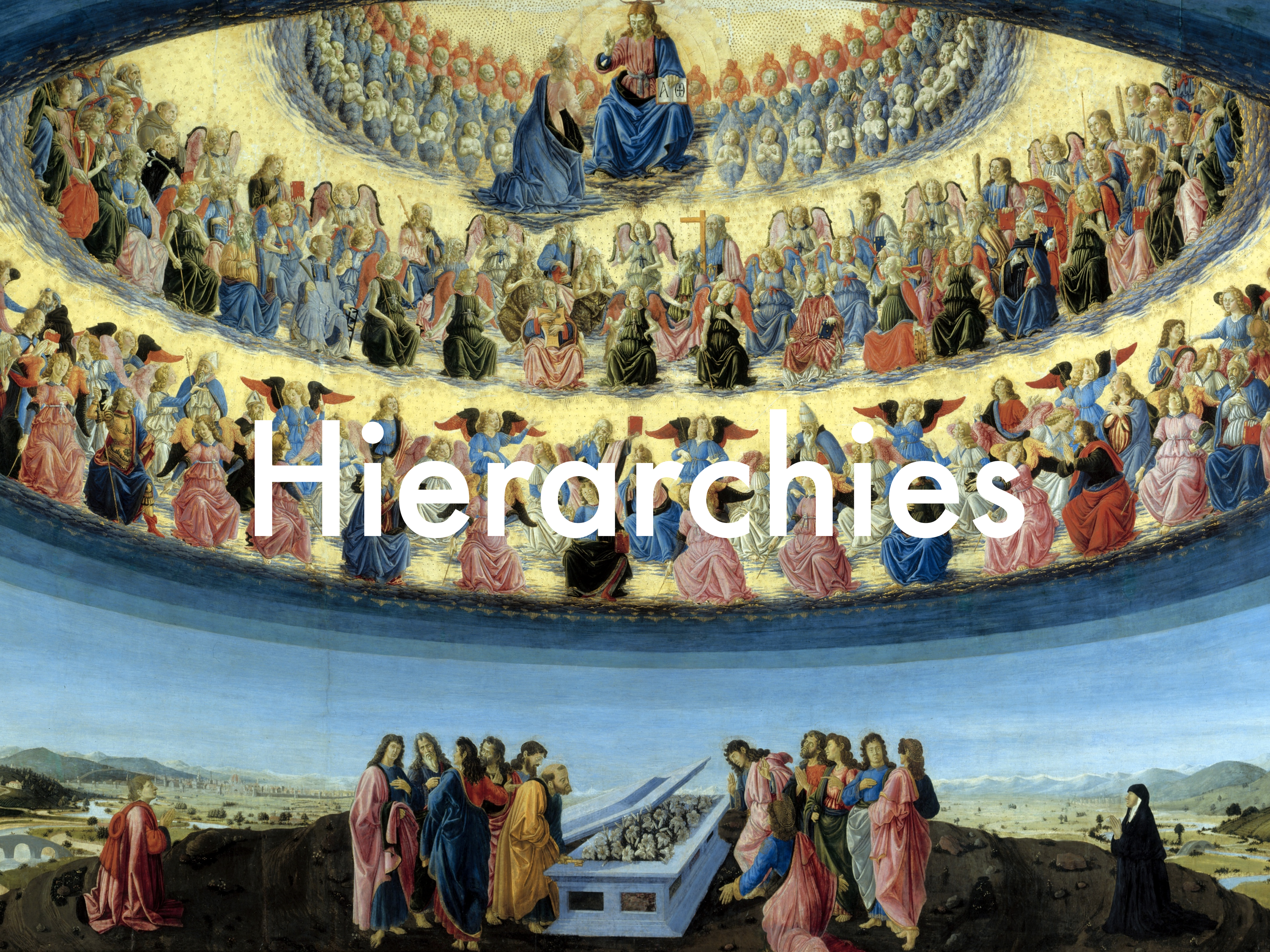


500 million users per day on 1000 machines

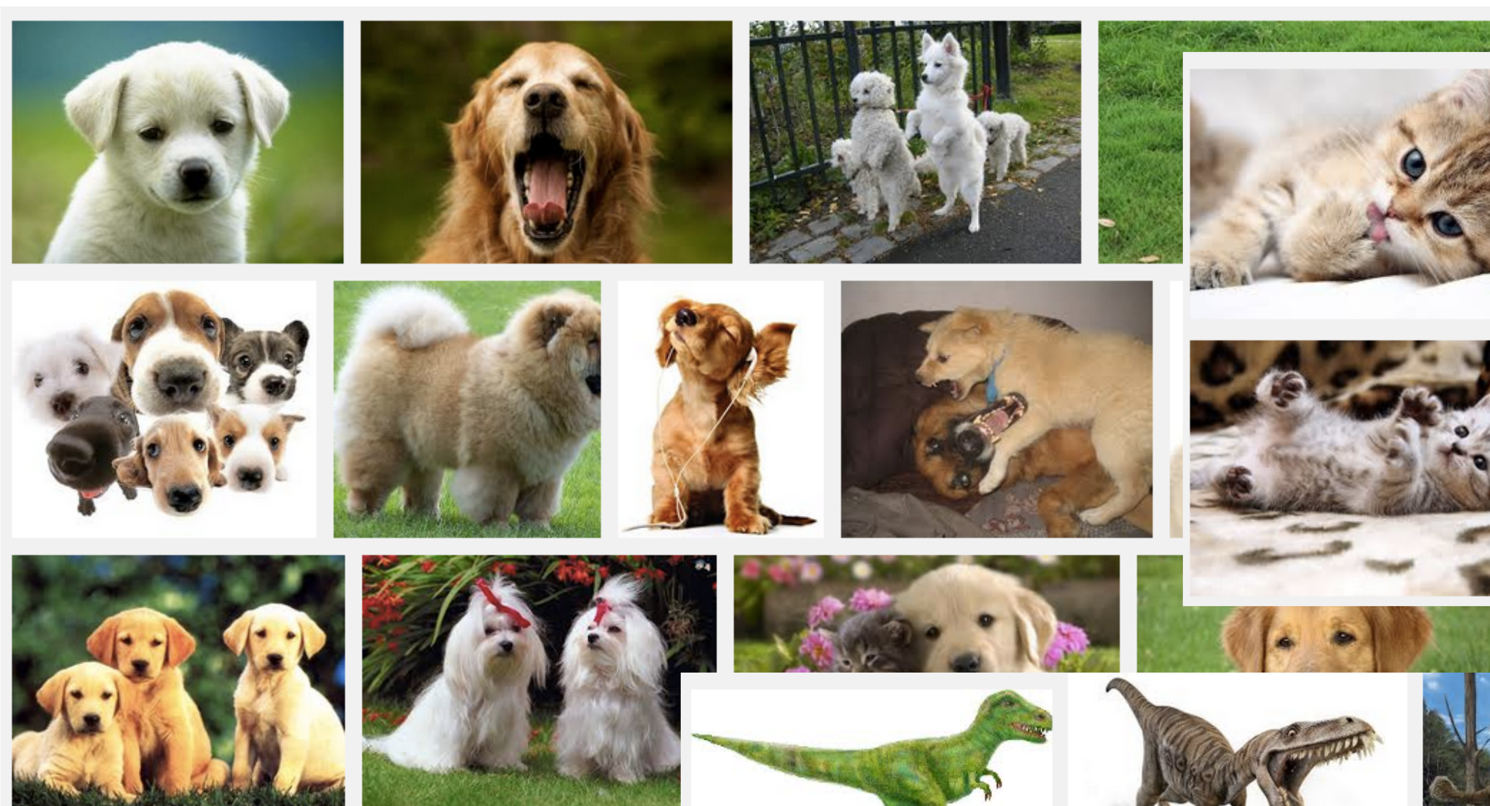
Google

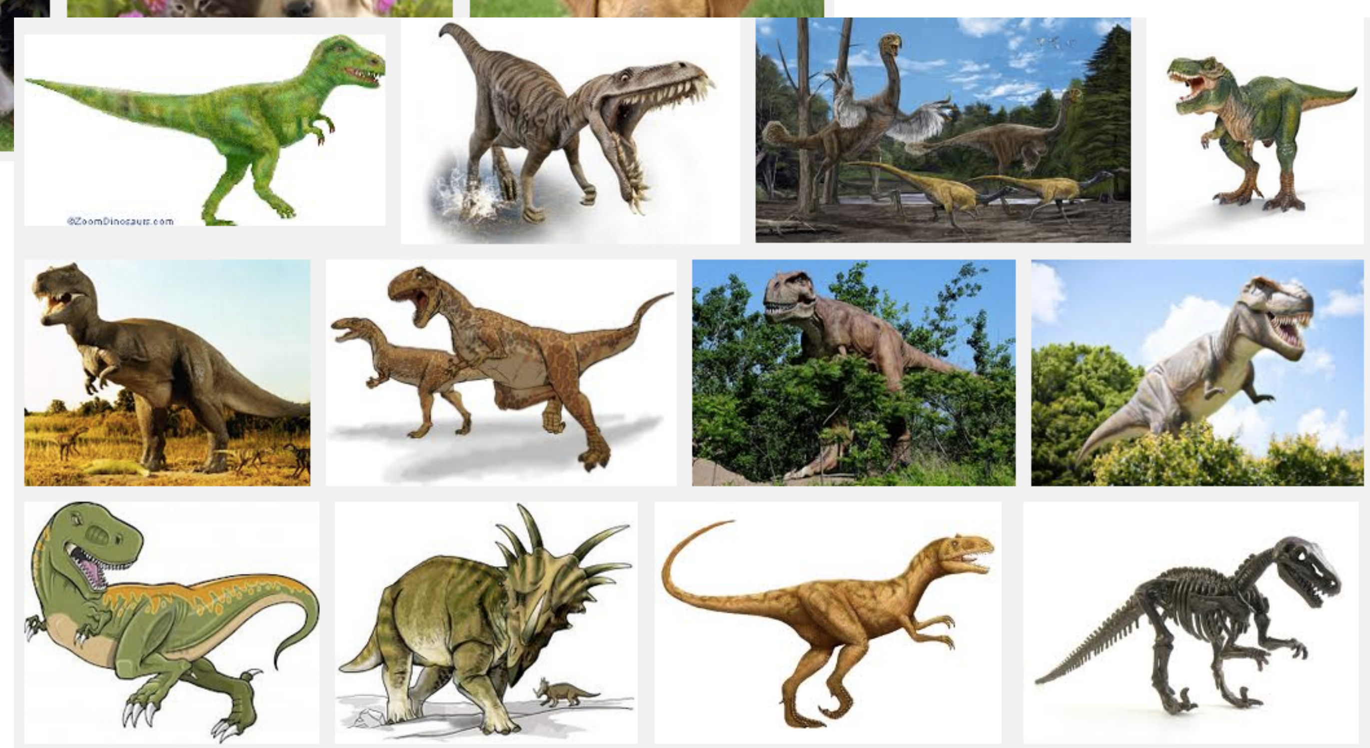
Carnegie Mellon University

Hierarchies

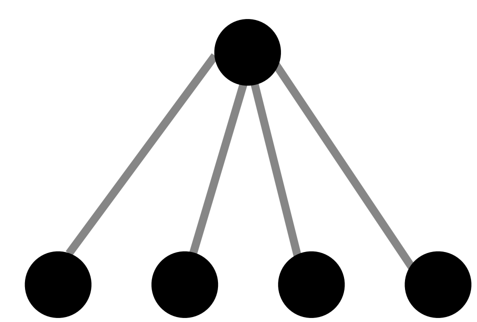


Modeling stuff - Clusters



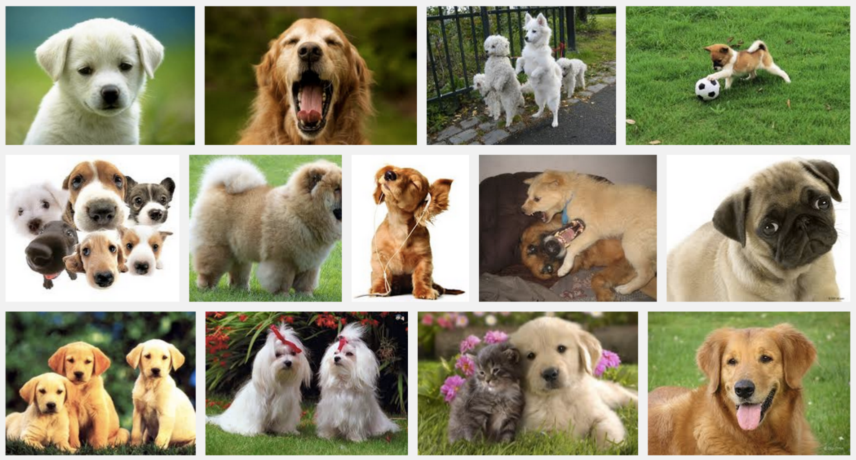


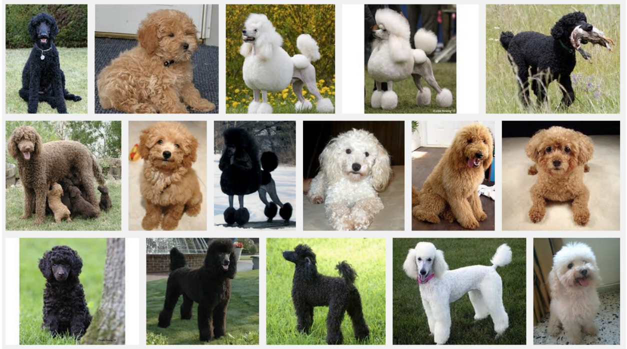
Dirichlet process



Modeling stuff
Hierarchy of Clusters

Adams, Ghrahramani, Jordan, 2008





e.g. hierarchical stick breaking

Carnegie Mellon University

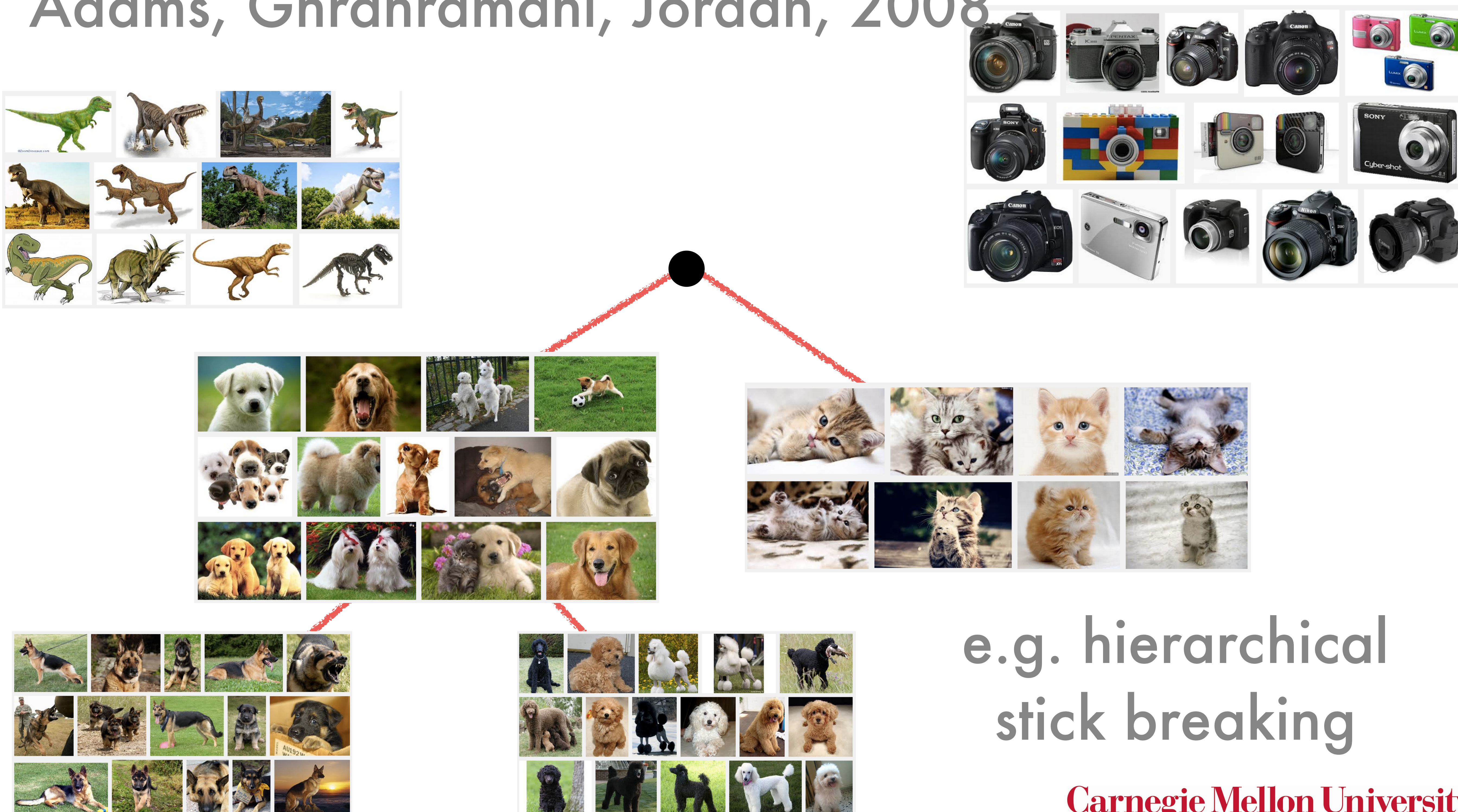
Modeling stuff
Hierarchy of Clusters

Adams, Ghrahramani, Jordan, 2008

e.g. hierarchical stick breaking

Modeling stuff
Hierarchy of Clusters

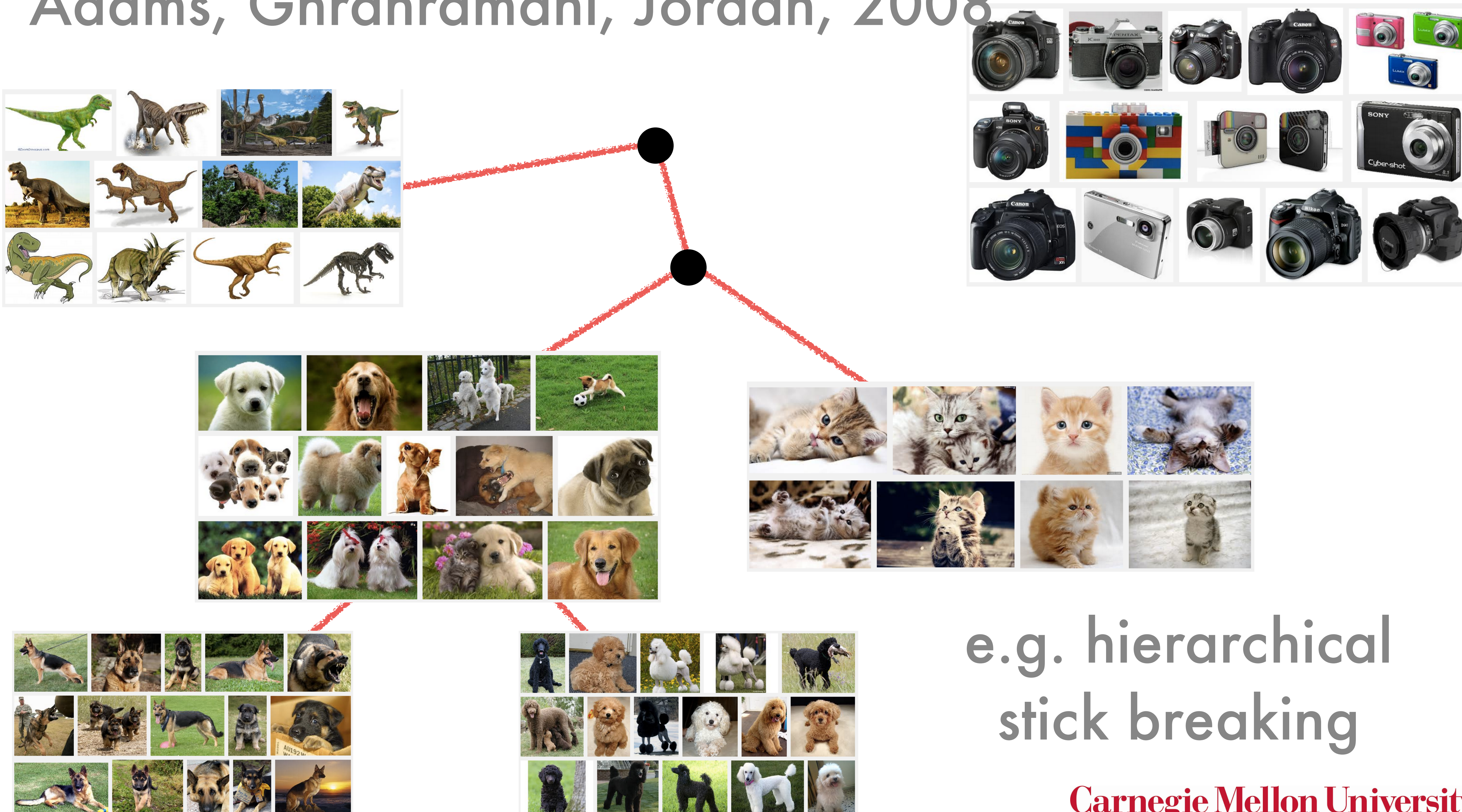
Adams, Ghrahramani, Jordan, 2008



e.g. hierarchical stick breaking

Modeling stuff
Hierarchy of Clusters

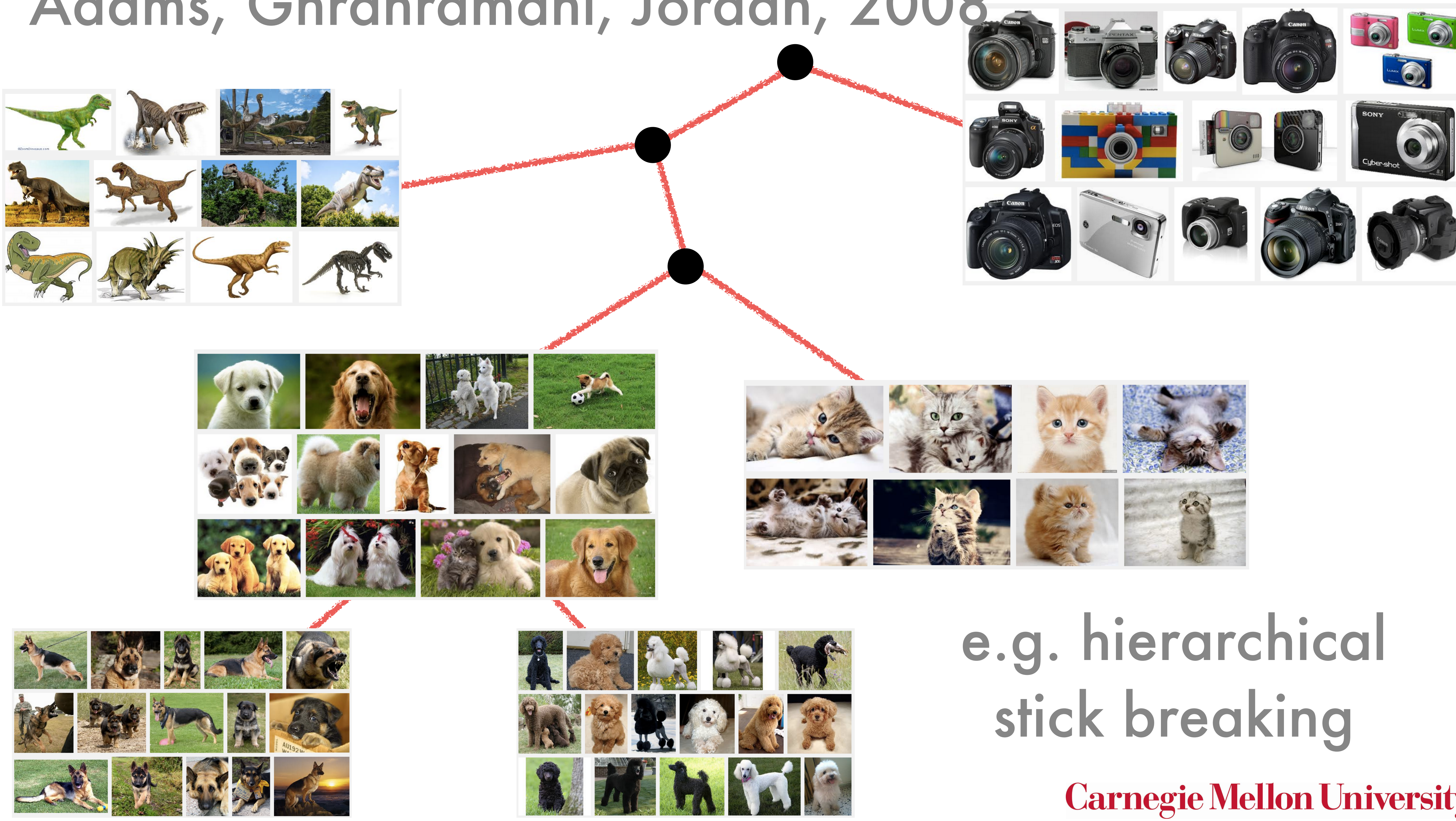
Adams, Ghrahramani, Jordan, 2008



e.g. hierarchical stick breaking

Modeling stuff
Hierarchy of Clusters

Adams, Ghrahramani, Jordan, 2008



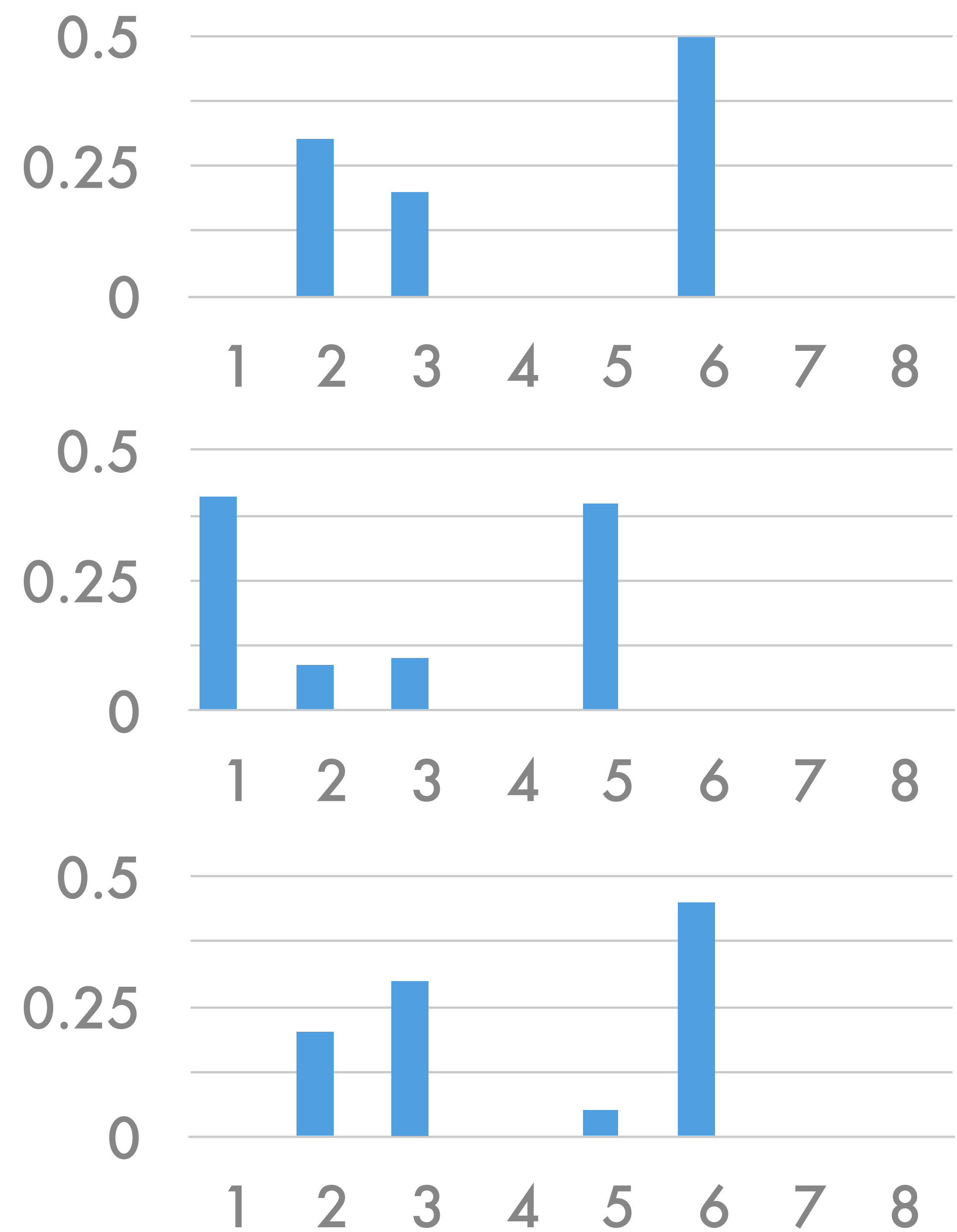
e.g. hierarchical stick breaking

Recall - Factorial representations

Blei, Ng, Jordan 2003

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

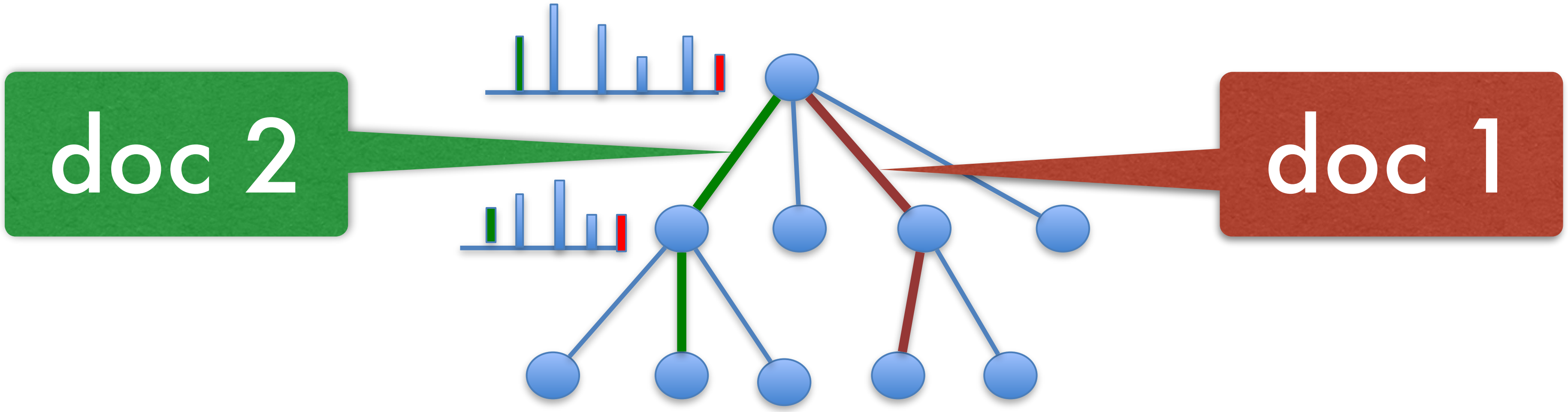


Hierarchical factorial representations

- Hierarchical Dirichlet Process (Teh et al. 2006)
  - Given hierarchy of objects
  - DP on children inherits from parent

$$G_i \sim DP(G_0, \gamma') \text{ and } G_0 \sim DP(H, \gamma)$$

- Nested Chinese Restaurant Process (Blei et al. 2010)



- Pachinko allocation (McCallum et al., 2010) (use directed acyclic graph, often predefined)

Hierarchical factorial representations

- Hierarchical Dirichlet Process (Teh et al. 2006)
  - Given hierarchy of objects
  - DP on children inherits from parent

$$G_i \sim DP(G_0, \gamma') \text{ and } G_0 \sim DP(H, \gamma)$$

- Nested Chinese Restaurant Process (Blei et al. 2010)

(1) For each node t in the infinite tree draw a topic β_t independently.
(2) For each document d draw

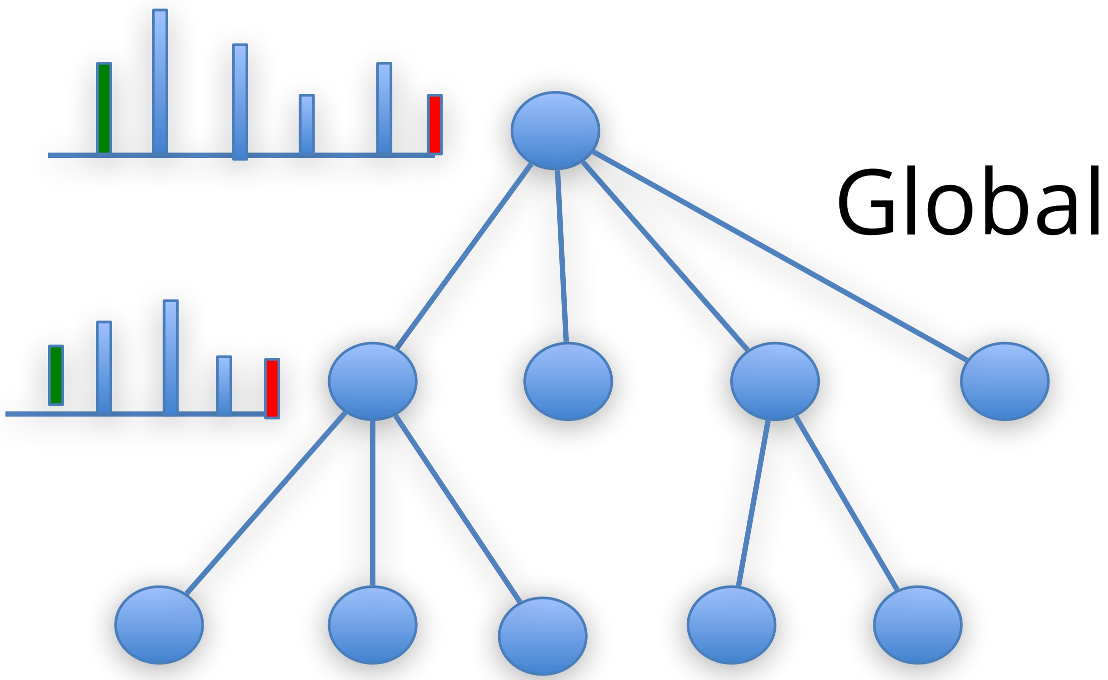
(a) Draw a path c_d over the tree using the nCRP.
(b) Draw a distribution θ_d over levels in the tree using $\text{GEM}(\alpha_1, \alpha_2)$.
(c) For all words in d draw a level t from θ_d and a corresponding word from β_t.

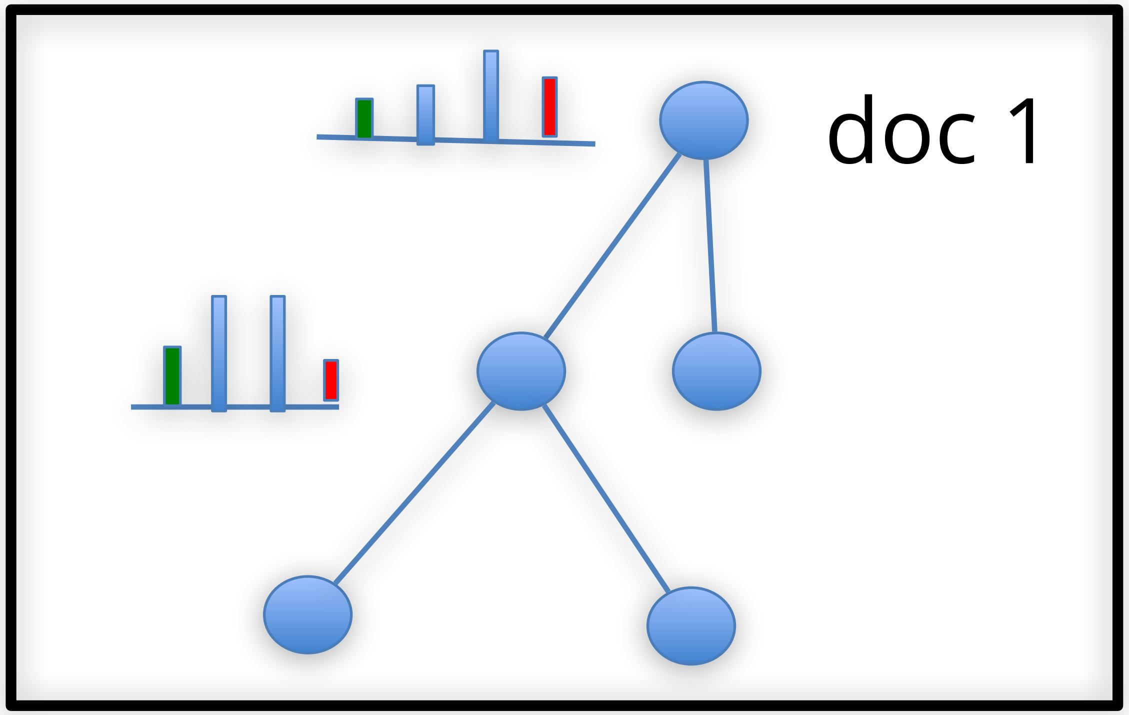
- Pachinko allocation (McCallum et al., 2010)
(use directed acyclic graph, often predefined)

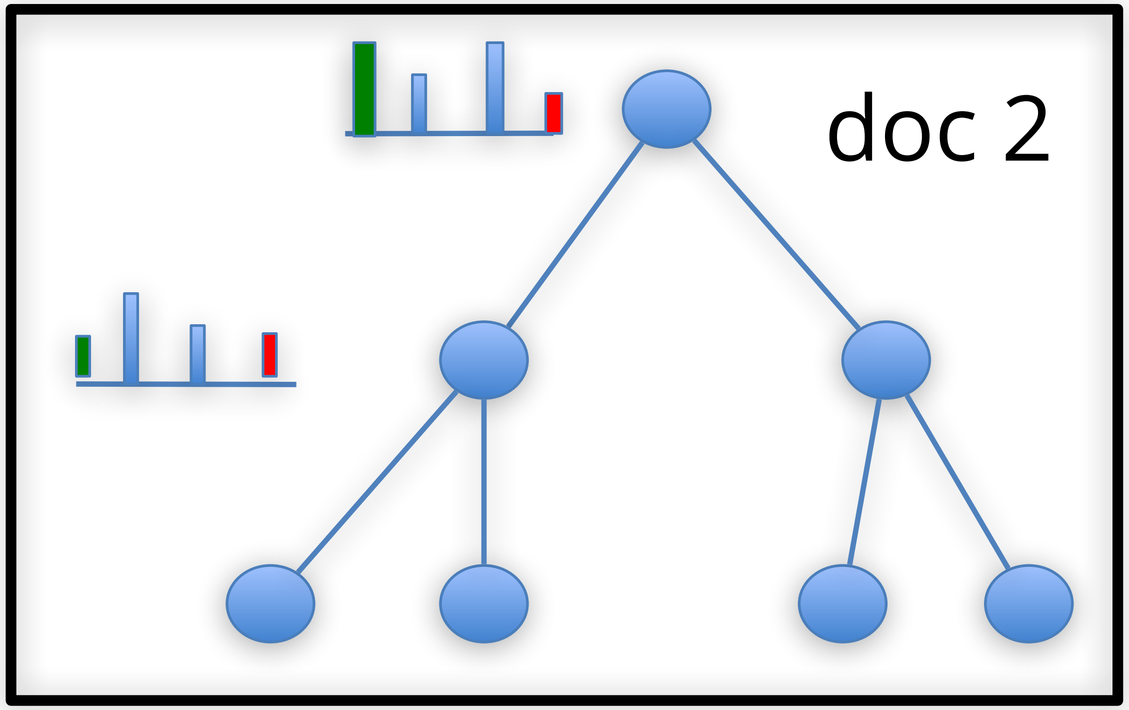
Variable resolution models

- Users have different levels of detail for preferences
  (photography, Shostakovich, Bauhaus, -)
  (Panasonic m43, classical music, -, NFL)
- Documents have different topics & levels of detail
  (49ers, sports, and the Bay Area)
  (Dirichlet process, machine learning, Twitter)
- Want tree distribution per object. Sharing of strength between different trees
- Nested Hierarchical Dirichlet Process (Paisley, Wang, Blei, Jordan, 2012)
- Nested Chinese Restaurant Franchise (Ahmed, Hong, Smola, 2013)

Generative Process

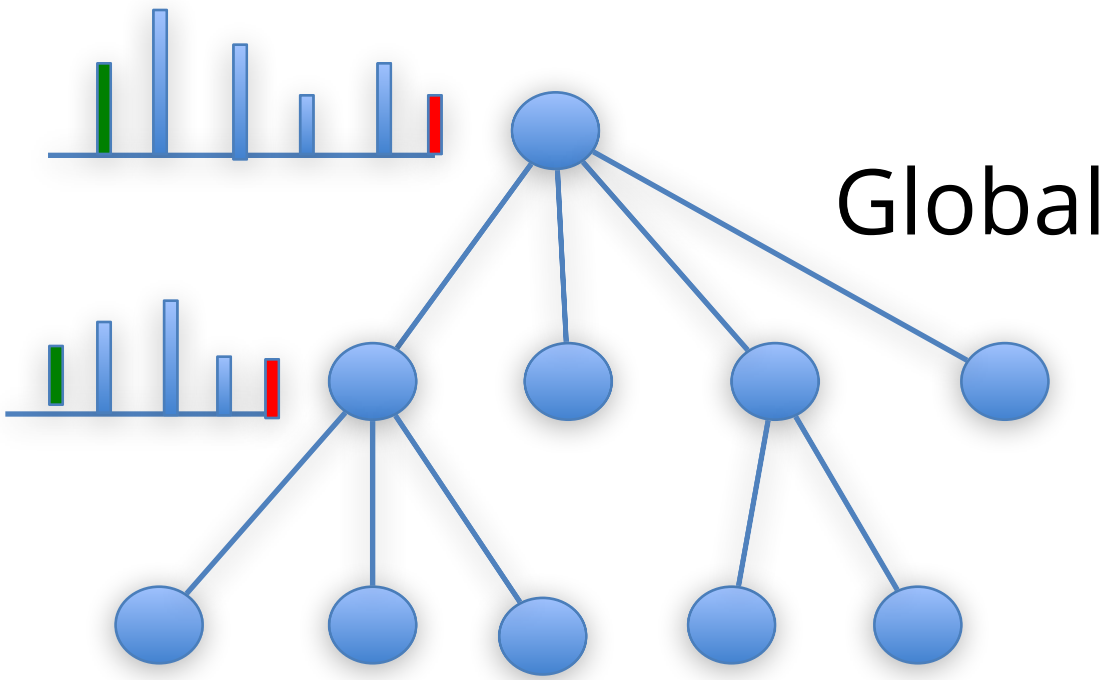


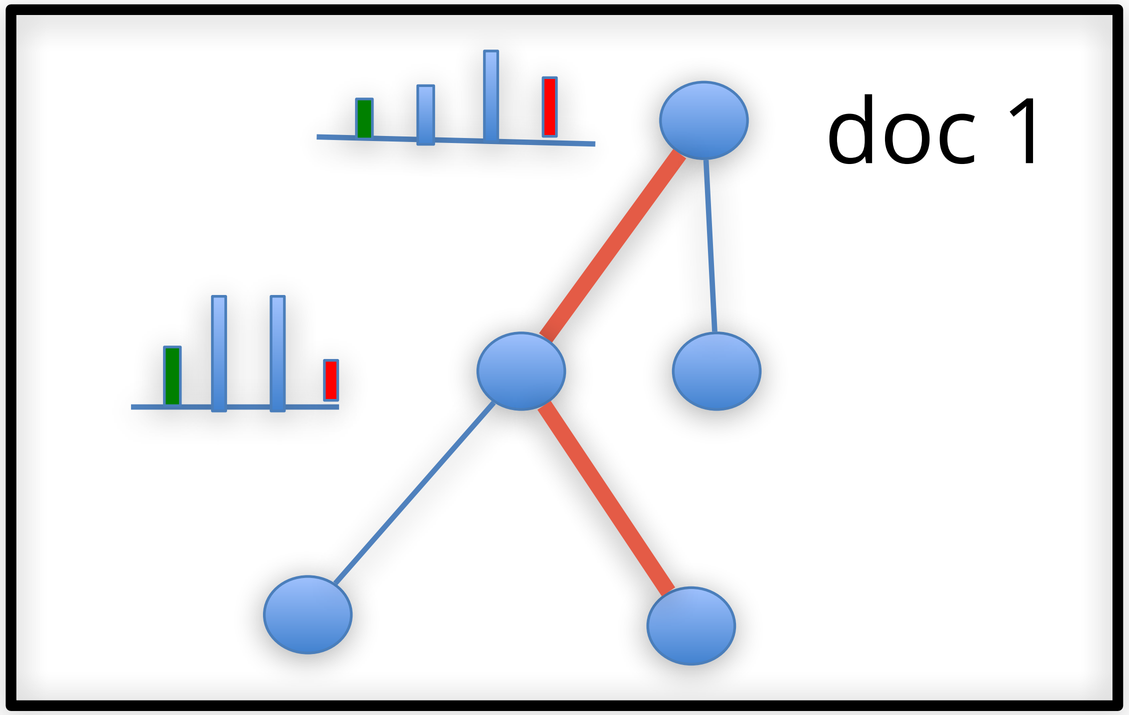


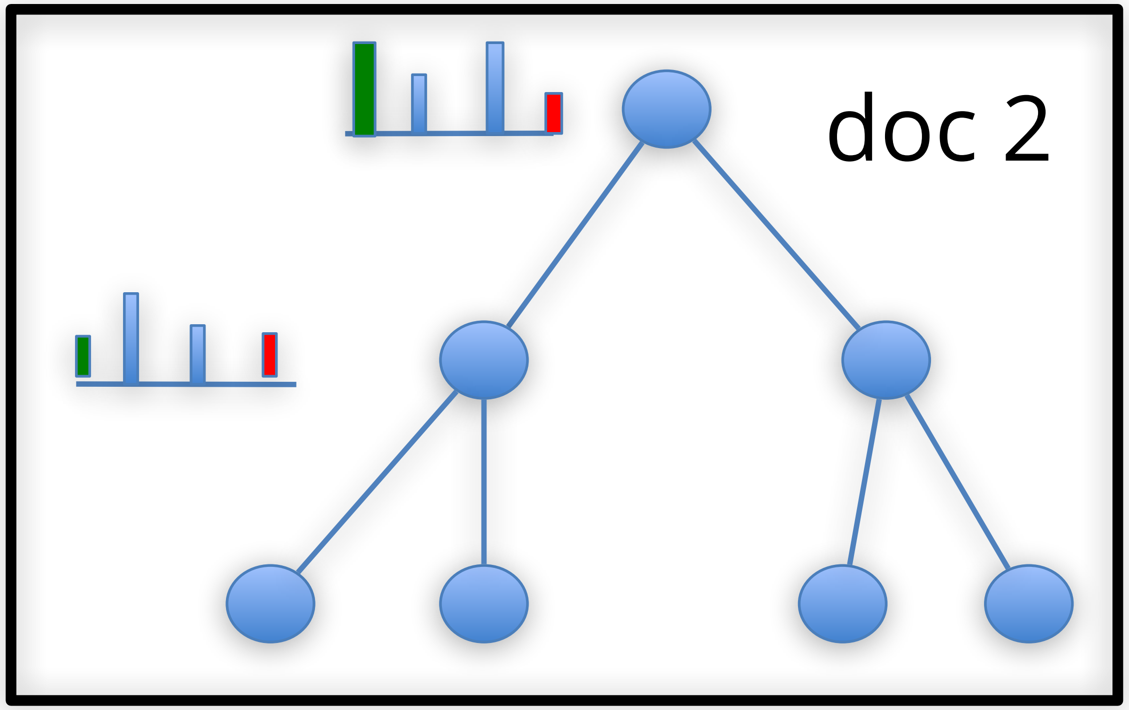


- for each document
  - for each word
    - select path in doc
    - if new in document
      then select from global
    - if new in global
      then add new path

Generative Process

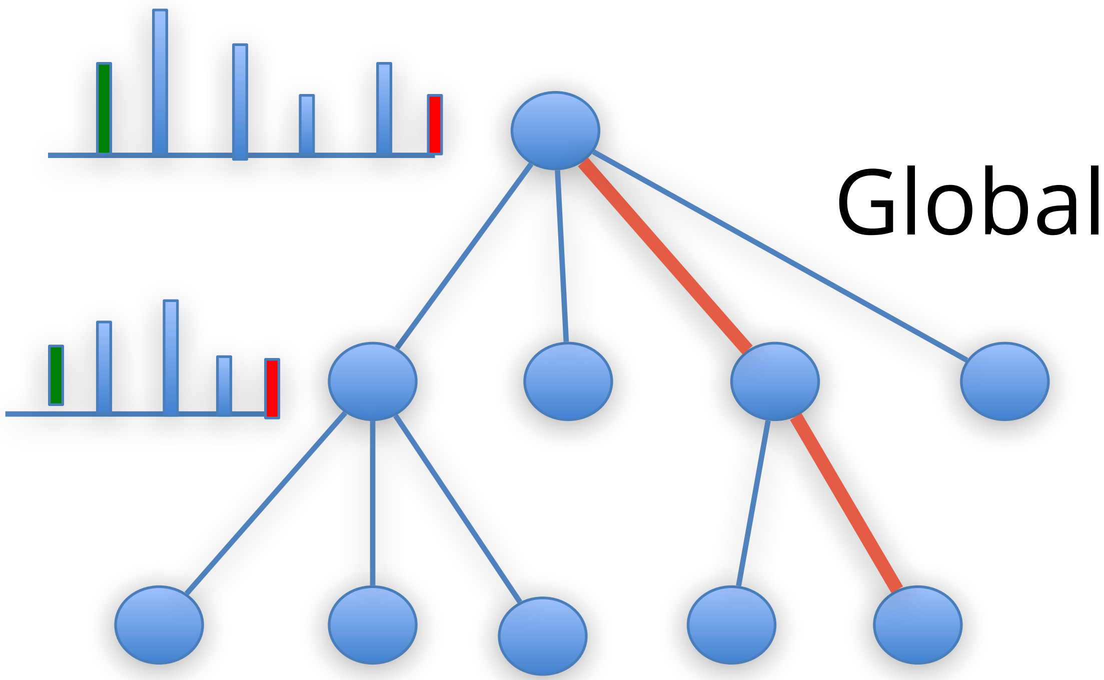


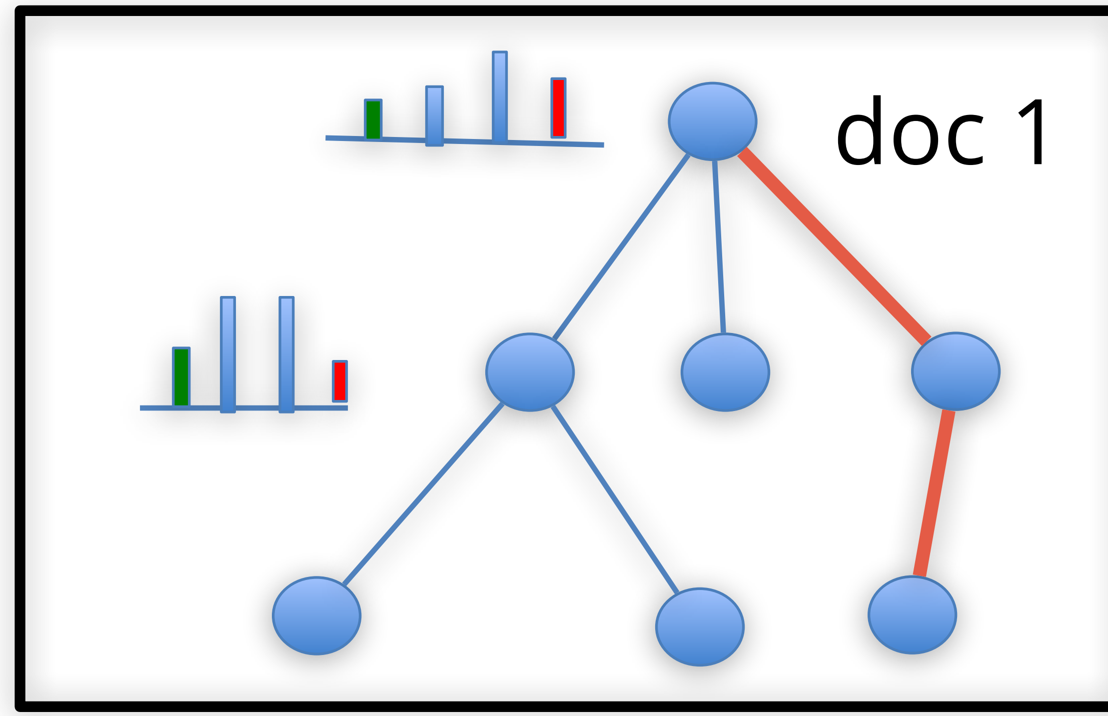


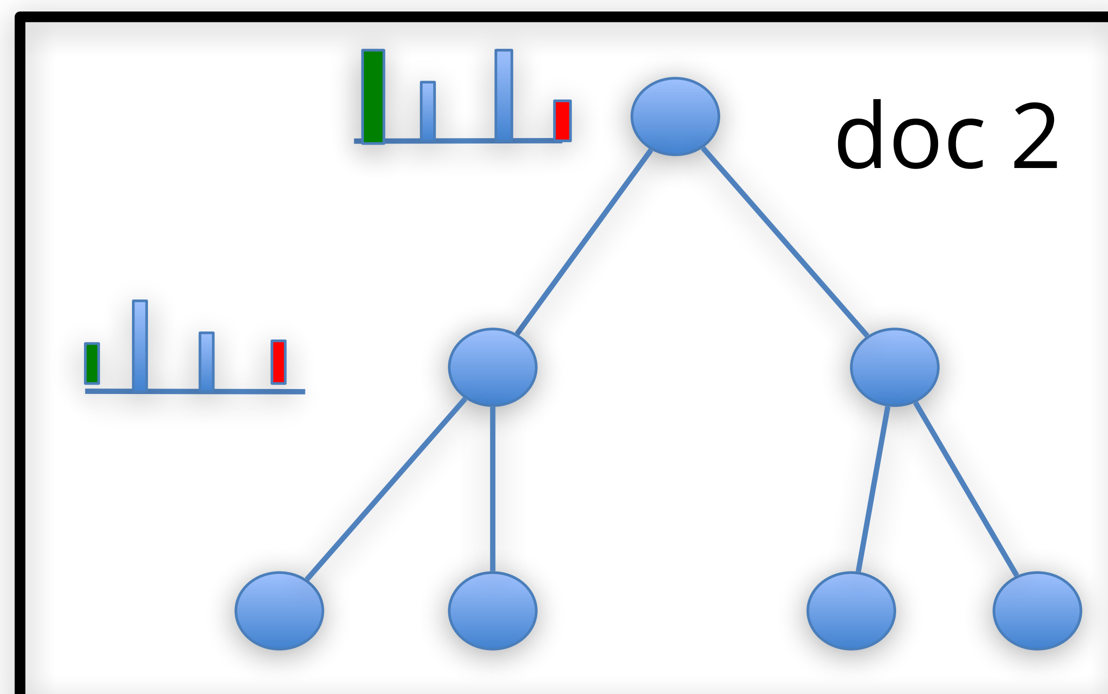


- for each document
  - for each word
    - select path in doc
    - if new in document
      then select from global
    - if new in global
      then add new path

Generative Process

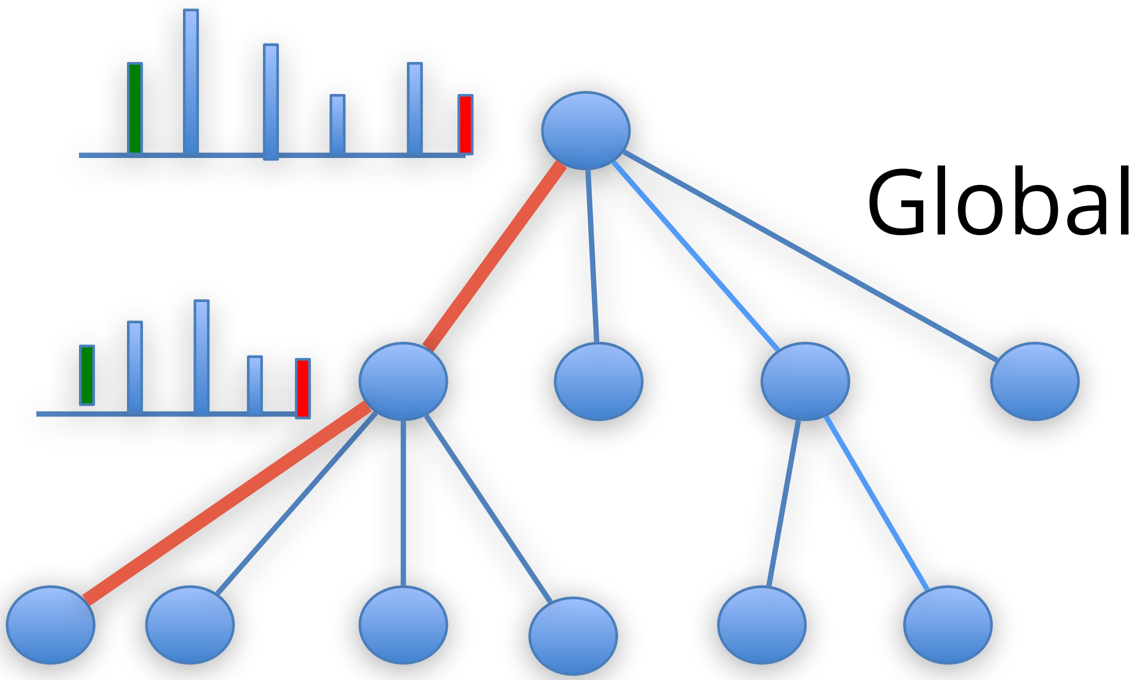


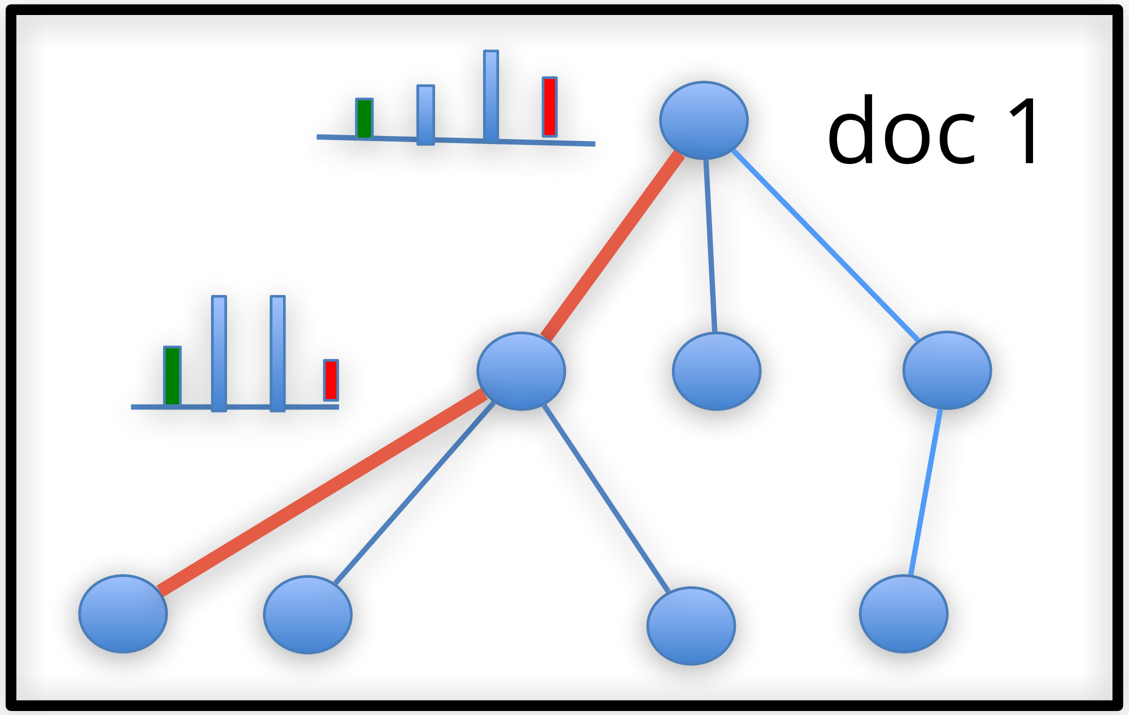


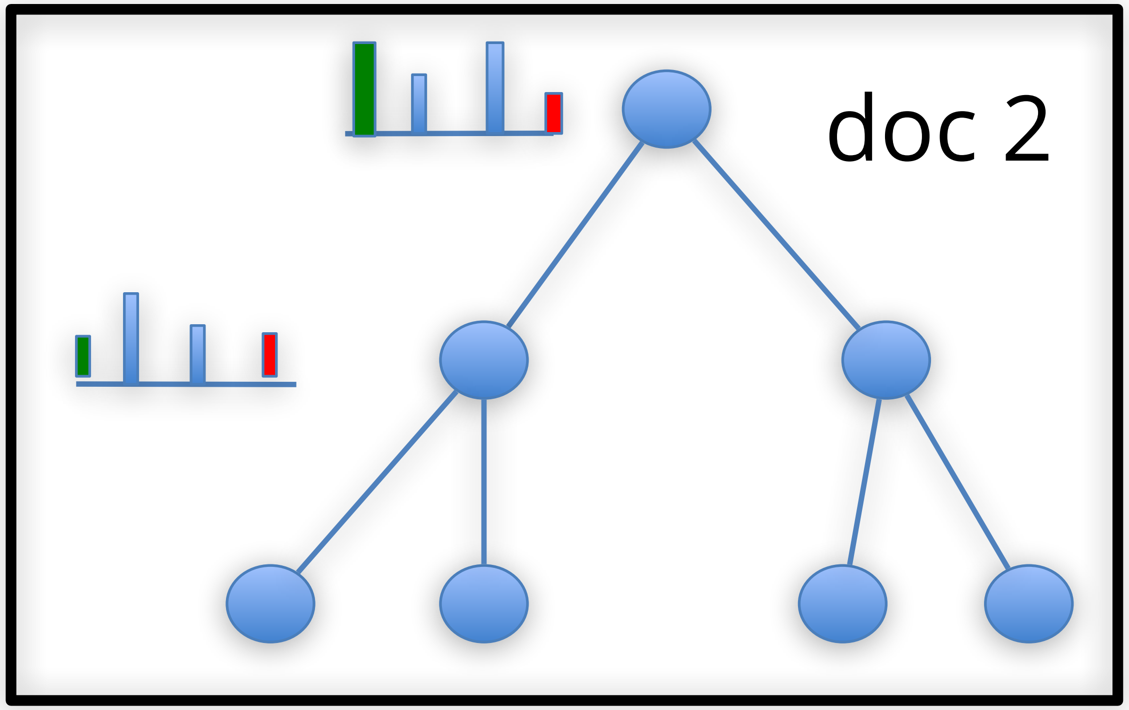


- for each document
  - for each word
    - select path in doc
    - if new in document
      then select from global
    - if new in global
      then add new path

Generative Process







- for each document
    - for each word
        - select path in doc
        - if new in document
          then select from global
        - if new in global
          then add new path

Document Modeling

(Ahmed, Hong, Smola, 2012)

Carnegie Mellon University

Generative Process

For each word i in document d:

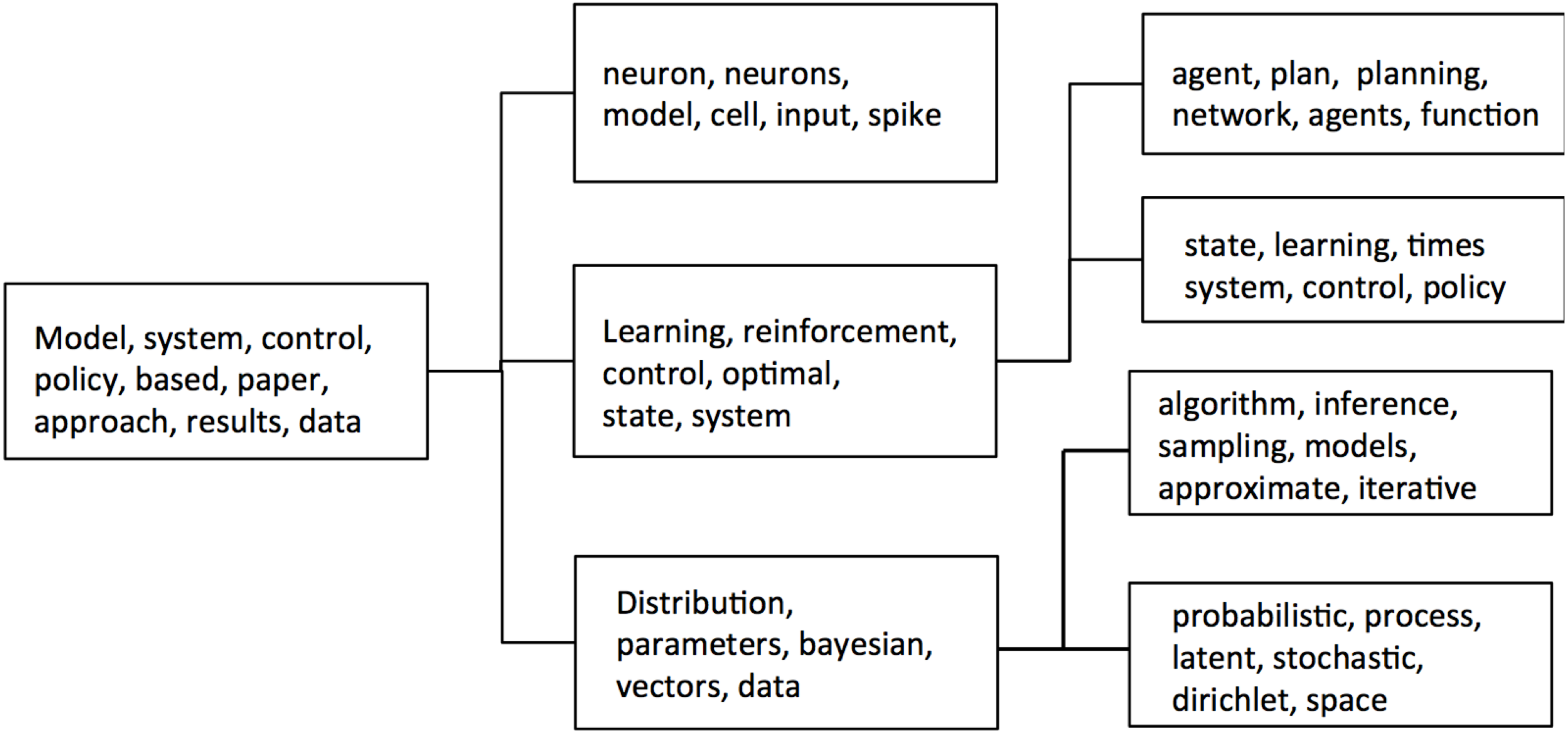
(a) Sample a node $v_{di} \sim \text{nCRF}(\gamma, \alpha, d)$.

(b) If node $v_{d,i}$ is a *globally* new node then

i. $\phi_{v_{d,i}} \sim \text{Dir}\left(\omega \phi_{\pi(v_{d,i})}\right)$

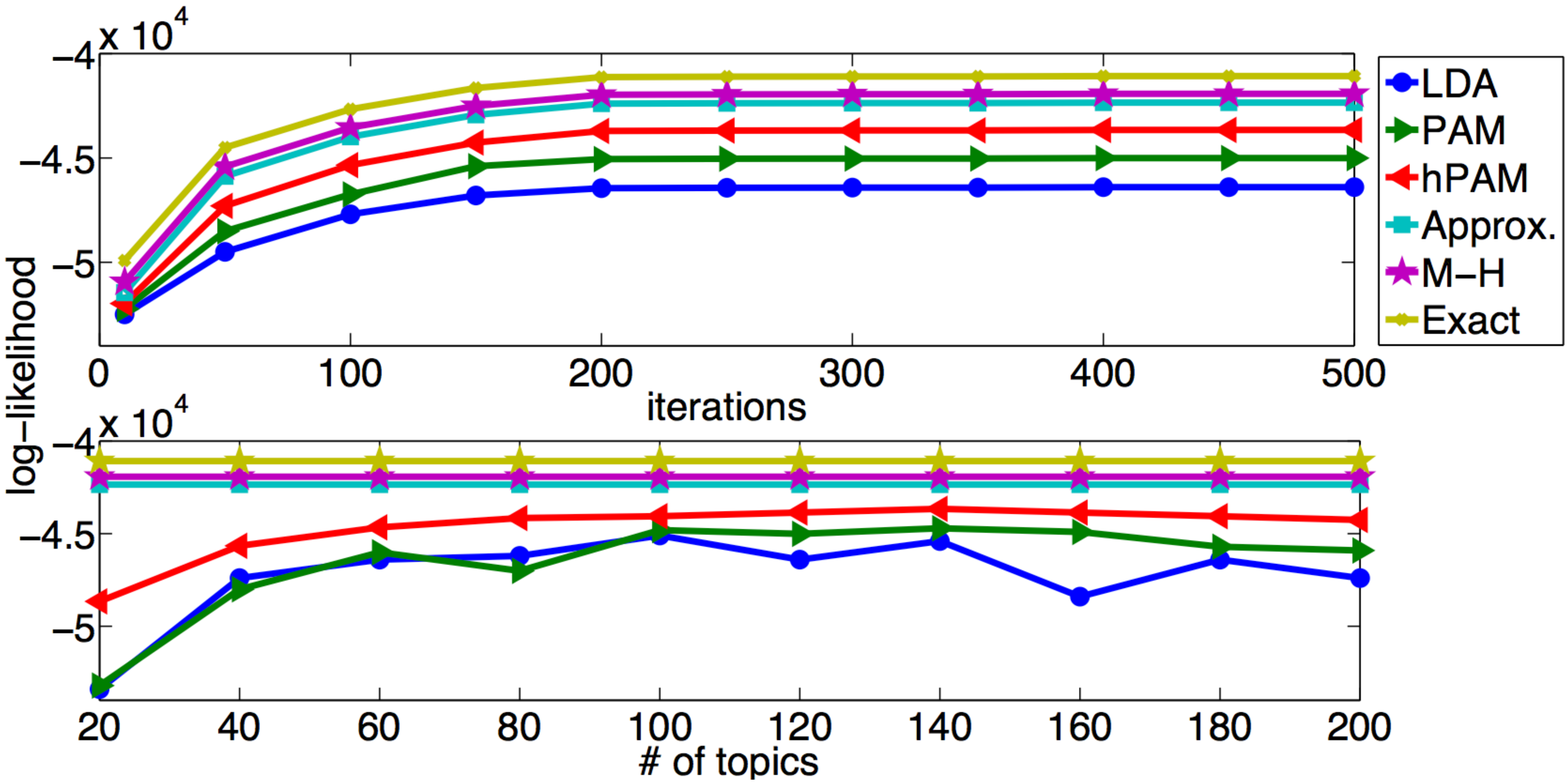
(c) Sample word $w_{d,i} \sim \text{Multi}(\phi_{v_{d,i}})$.

NIPS Corpus

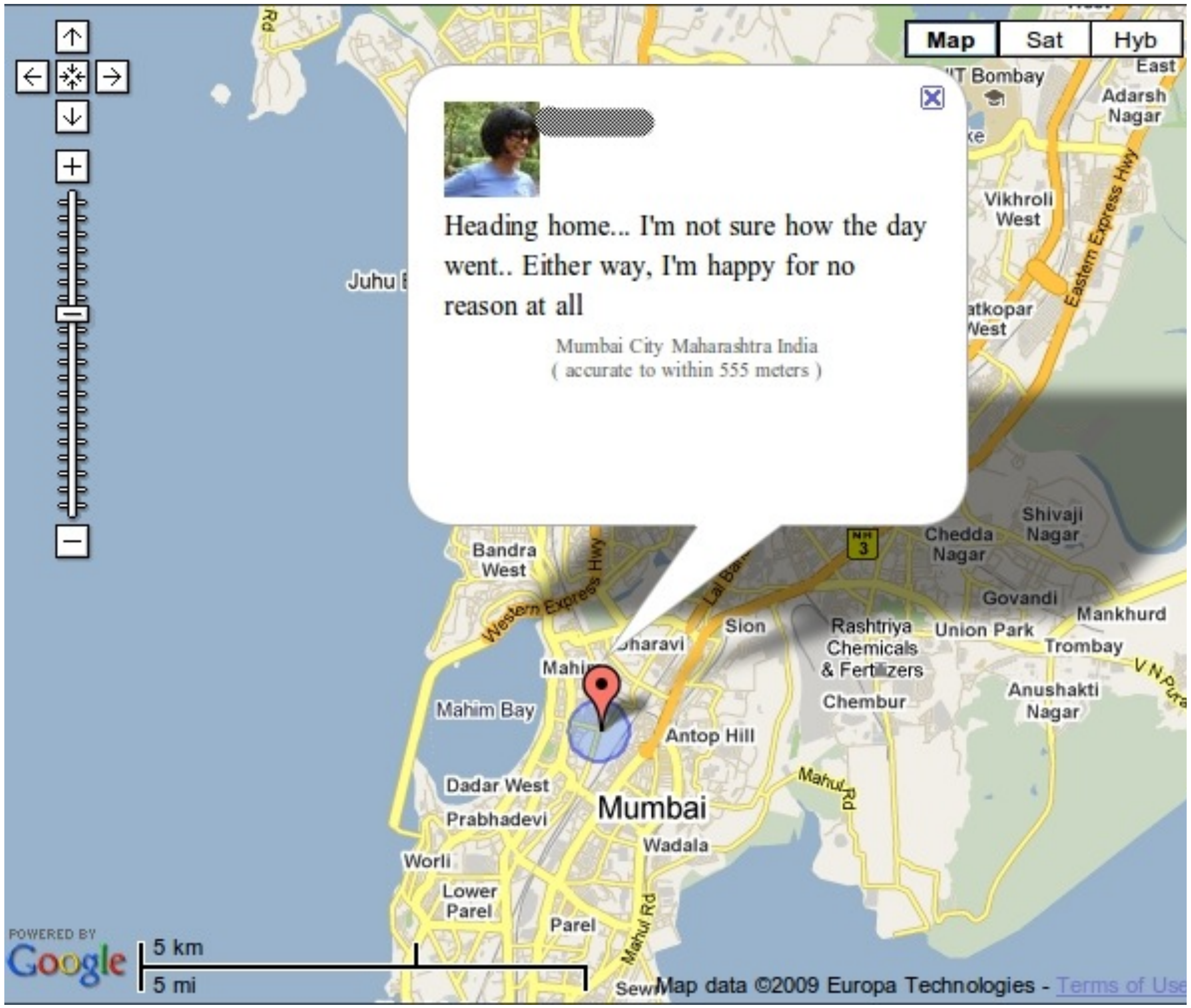


Model, system, control, policy, based, paper, approach, results, data
neuron, neurons, model, cell, input, spike
Learning, reinforcement, control, optimal, state, system
Distribution, parameters, bayesian, vectors, data
agent, plan, planning, network, agents, function
state, learning, times system, control, policy
algorithm, inference, sampling, models, approximate, iterative
probabilistic, process, latent, stochastic, dirichlet, space

NIPS Corpus



- Issue is sampling from hierarchy
  Exact is very slow, Metropolis-Hastings somewhat faster, Approximation a la Wallach et al.



Location Modeling

(Ahmed, Hong, Tsioutsiouliklis, Smola 2013)

Data

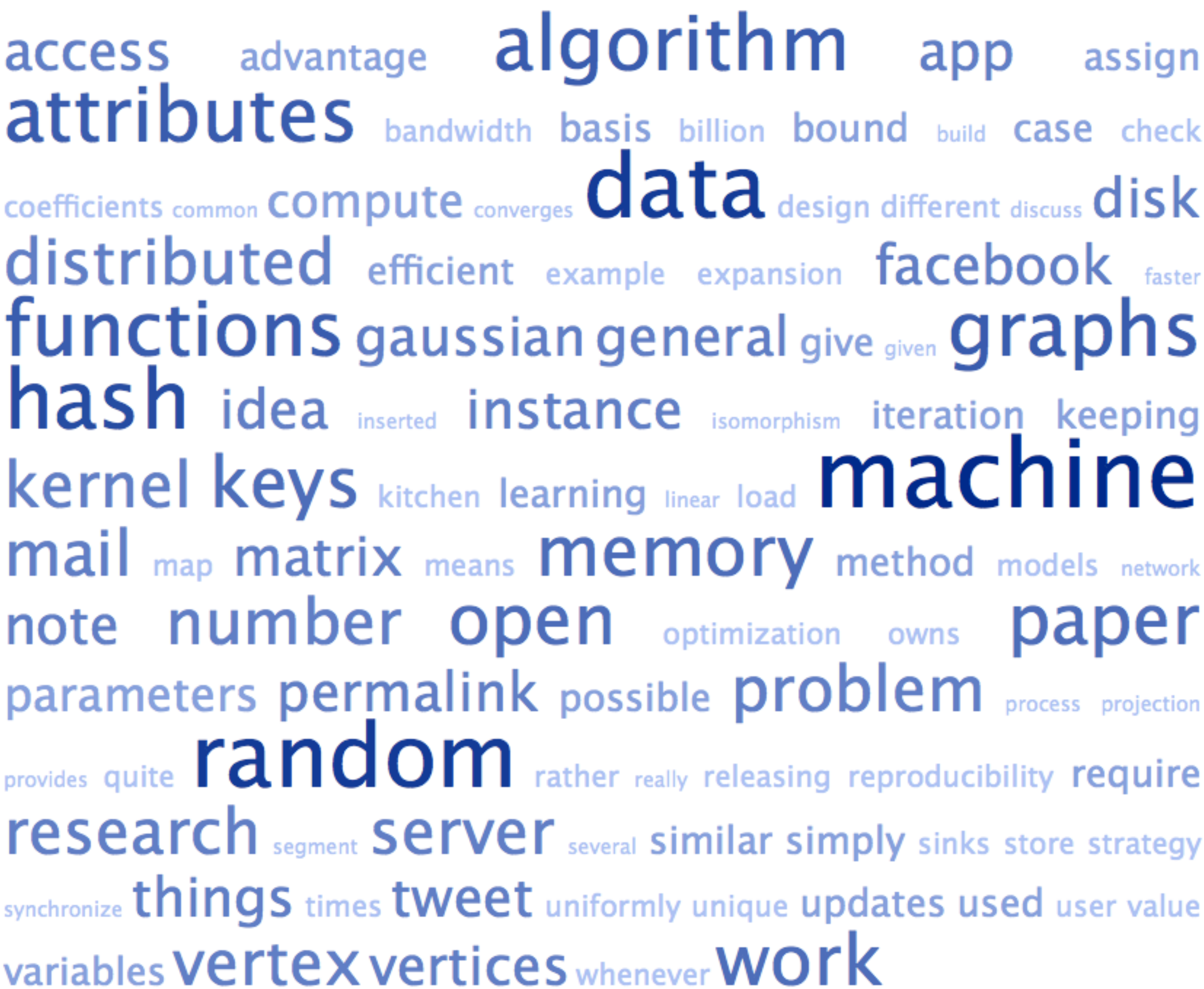
- Tweets
  - `140 character string`
  - `User ID`
  - `Time`
  - `Location (on small subset of data)`
- Location estimation
  Geographical targeting, content filtering
- User profiling
  Locations, interests

Modeling assumptions

User locations at variable resolution

Carnegie Mellon University

Modeling assumptions



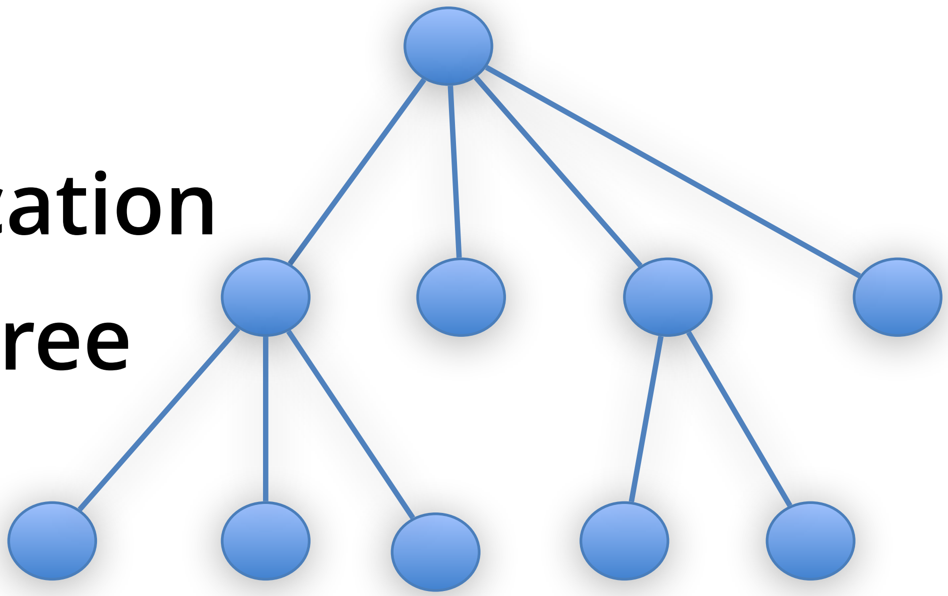
tagcloud on
blog.smola.org

Modeling assumptions

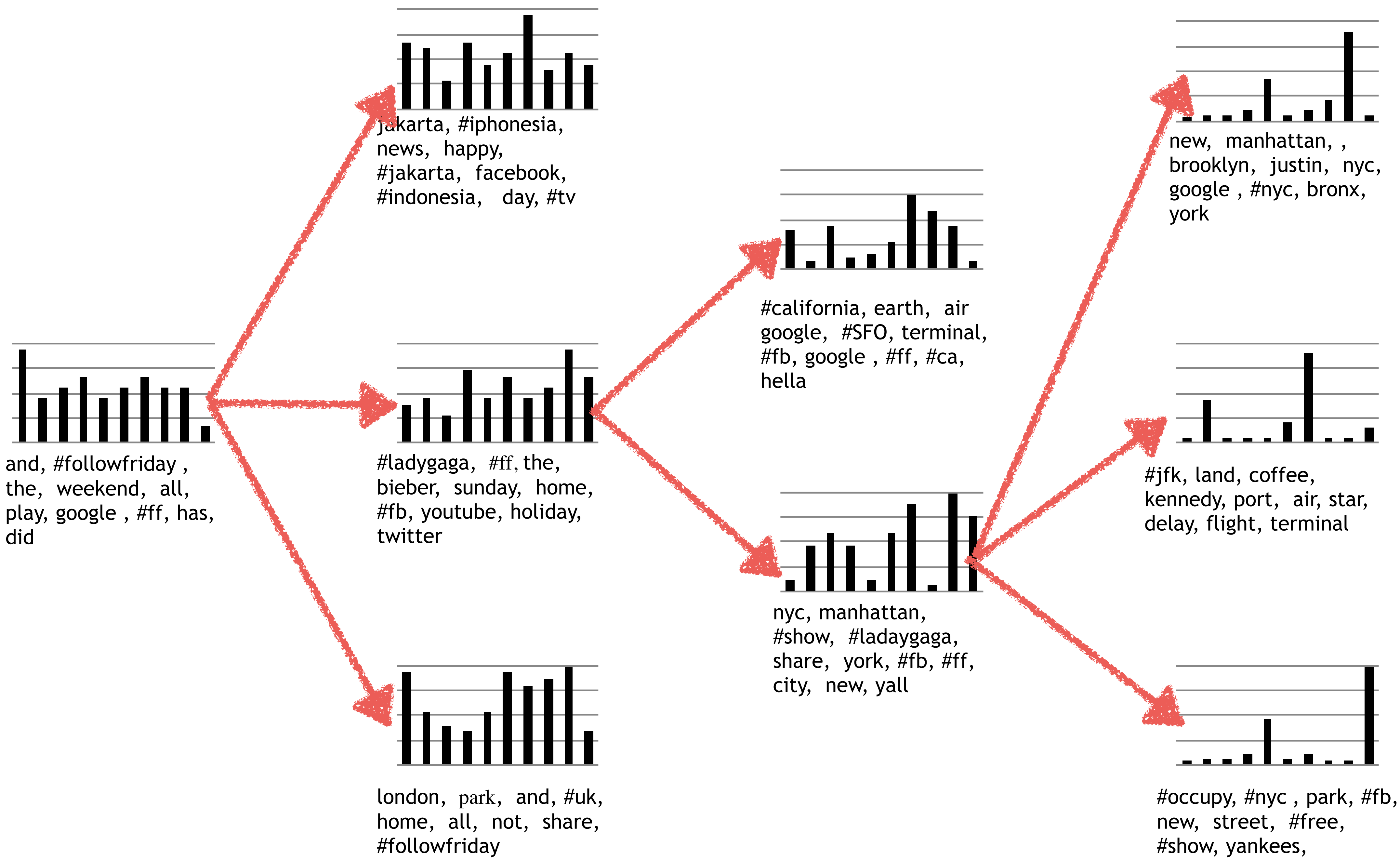
- User location has variable resolution (places, neighborhoods, cities)
- User content has variable detail (entities, stories, topics)
- Geographical affinity of location and topic (*I landed in SFO* probably means airport)
- Hierarchical model for text and location

Hierarchical modeling

- Arrange regions in a tree
  - Each node is a region
  - Each node models both text and location
- Cascade these distributions over the tree
  - Tree Gaussian MRF for locations
  - HDP for text
- Each user as a distribution over this region-tree



Topic hierarchy



Carnegie Mellon University

Hierarchical modeling

- Arrange regions in a tree
  - Each node is a region
  - Each node models both text and location
- Cascade these distributions over the tree
  - Tree Gaussian MRF for locations

$$\mu_r \sim \mathcal{N}\left(\mu_{\pi(r)}, \Sigma_{\pi(r)}\right)$$

$$\Sigma_r = \frac{1}{\mathrm{level}(r)} \Sigma_0.$$

  - Topic preference

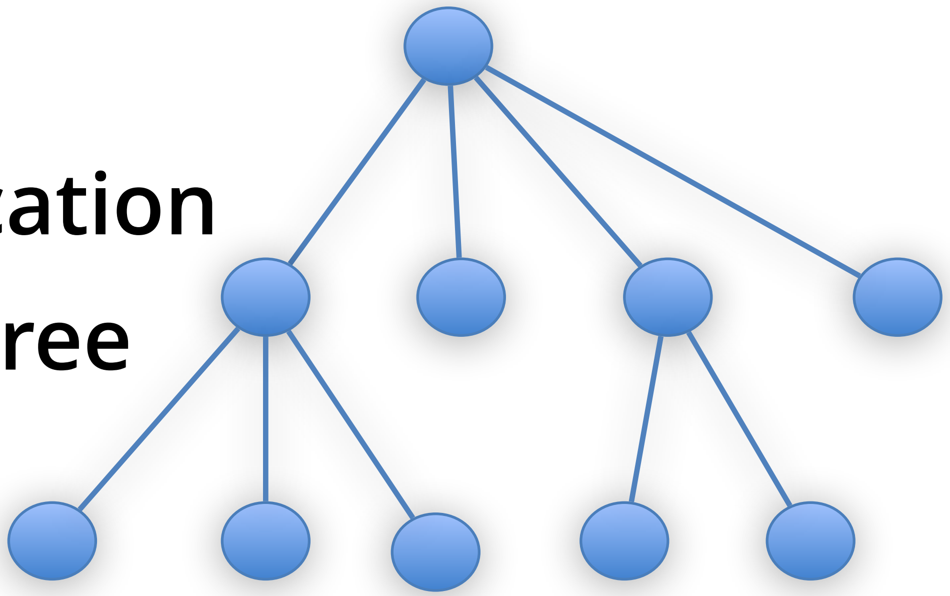
$$\theta_0 \sim \mathrm{Dir}(\beta)$$

$$\theta_r \sim \mathrm{Dir}\left(\lambda \theta_{\pi(r)}\right)$$

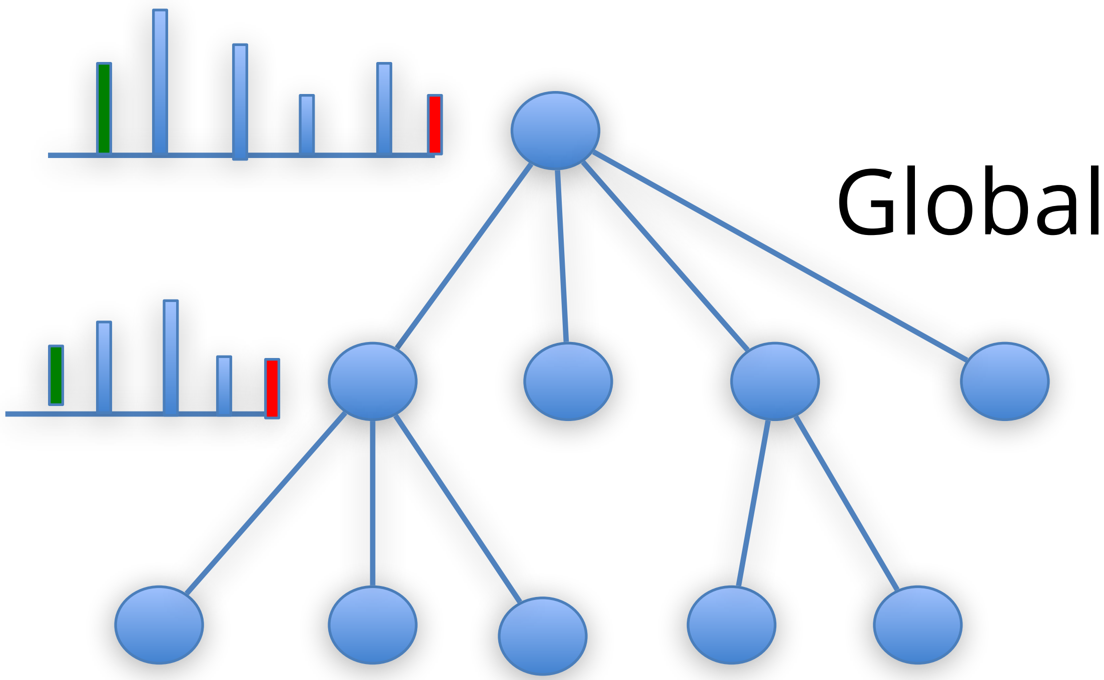
  - Language model

$$\phi_0 \sim \mathrm{Dir}(\eta).$$

$$\phi_r \sim \mathrm{Dir}\left(\omega \phi_{\pi(r)}\right)$$



Generative Process



For each tweet d written by each user u:

(a) Sample a node $r_d \sim \text{nCRF}(\gamma, \alpha, u)$.

(b) If node r_d is a *globally* new node then

  i. $\mu_{r_d} \sim \mathcal{N}\left(\mu_{\pi(r_d)}, \Sigma_{\pi(r_d)}\right)$

  ii. $\phi_{r_d} \sim \text{Dir}\left(\omega \phi_{\pi(r_d)}\right)$

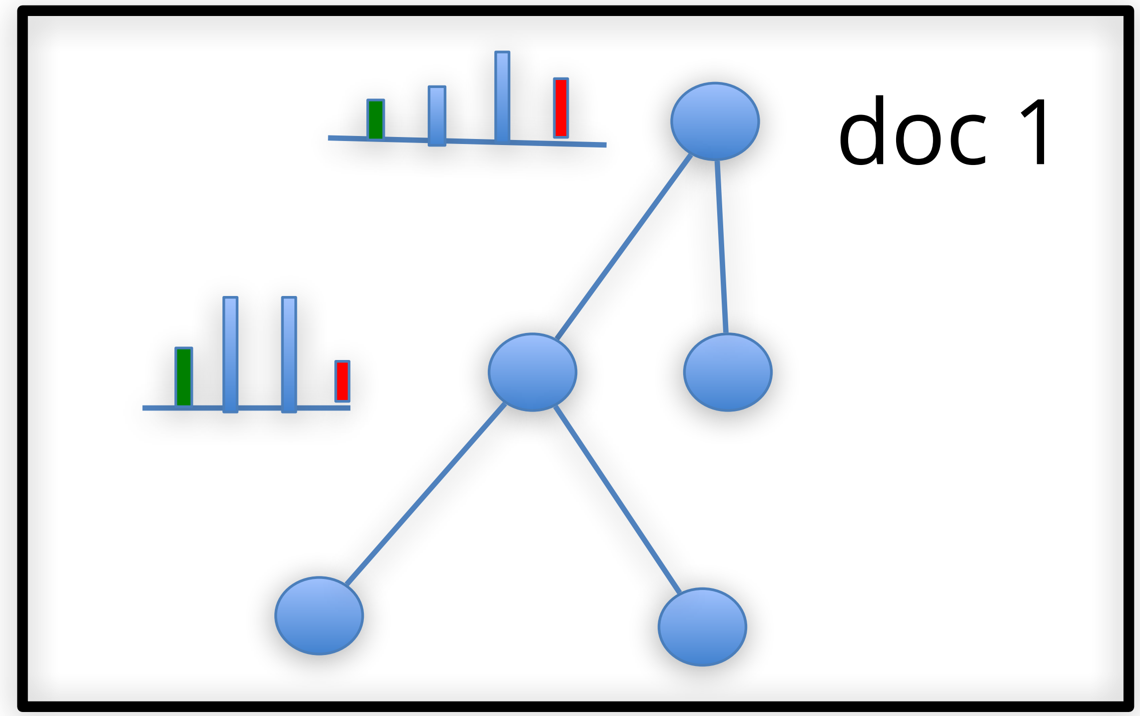
  iii. $\theta_{r_d} \sim \text{Dir}\left(\lambda \theta_{\pi(r_d)}\right)$

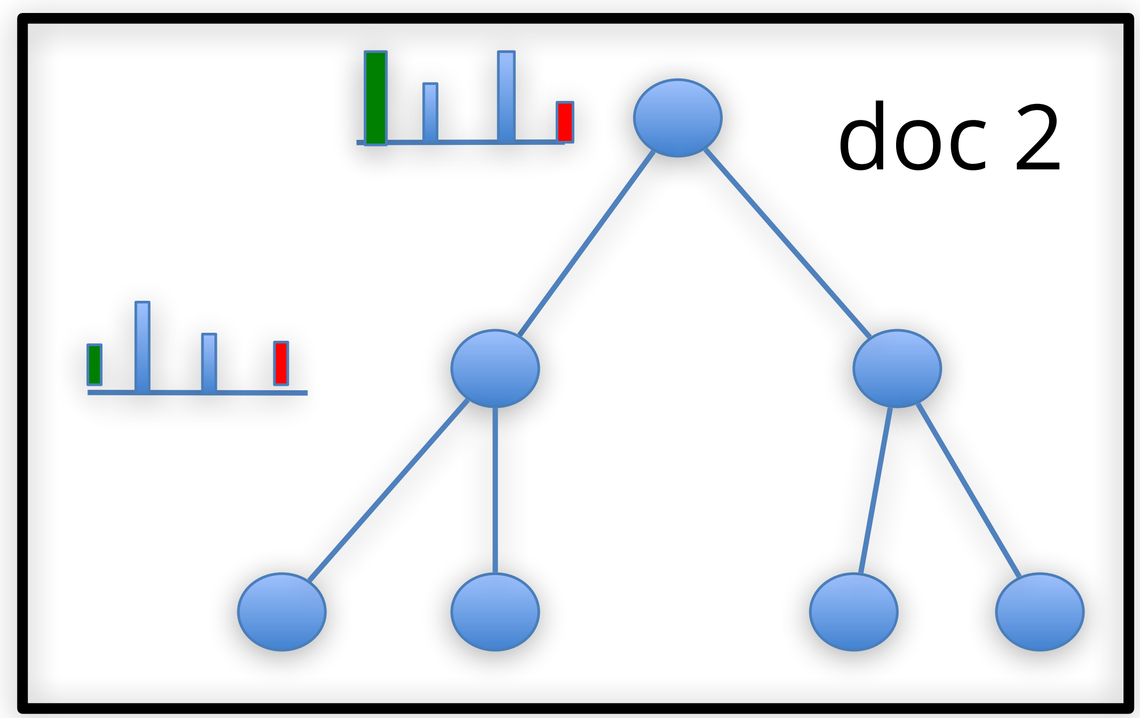
(c) Sample a location $l_d \sim \mathcal{N}(\mu_{r_d}, \Sigma_{r_d})$.

(d) For each word $w_{(d,i)}$:

  i. Sample a topic index $z_{(d,i)} \sim \text{Multi}(\theta_{r_d})$.

  ii. Sample word $w_{(d,i)} \sim \text{Multi}(\overline{\Pi}_{z_{(d,i)}})$.



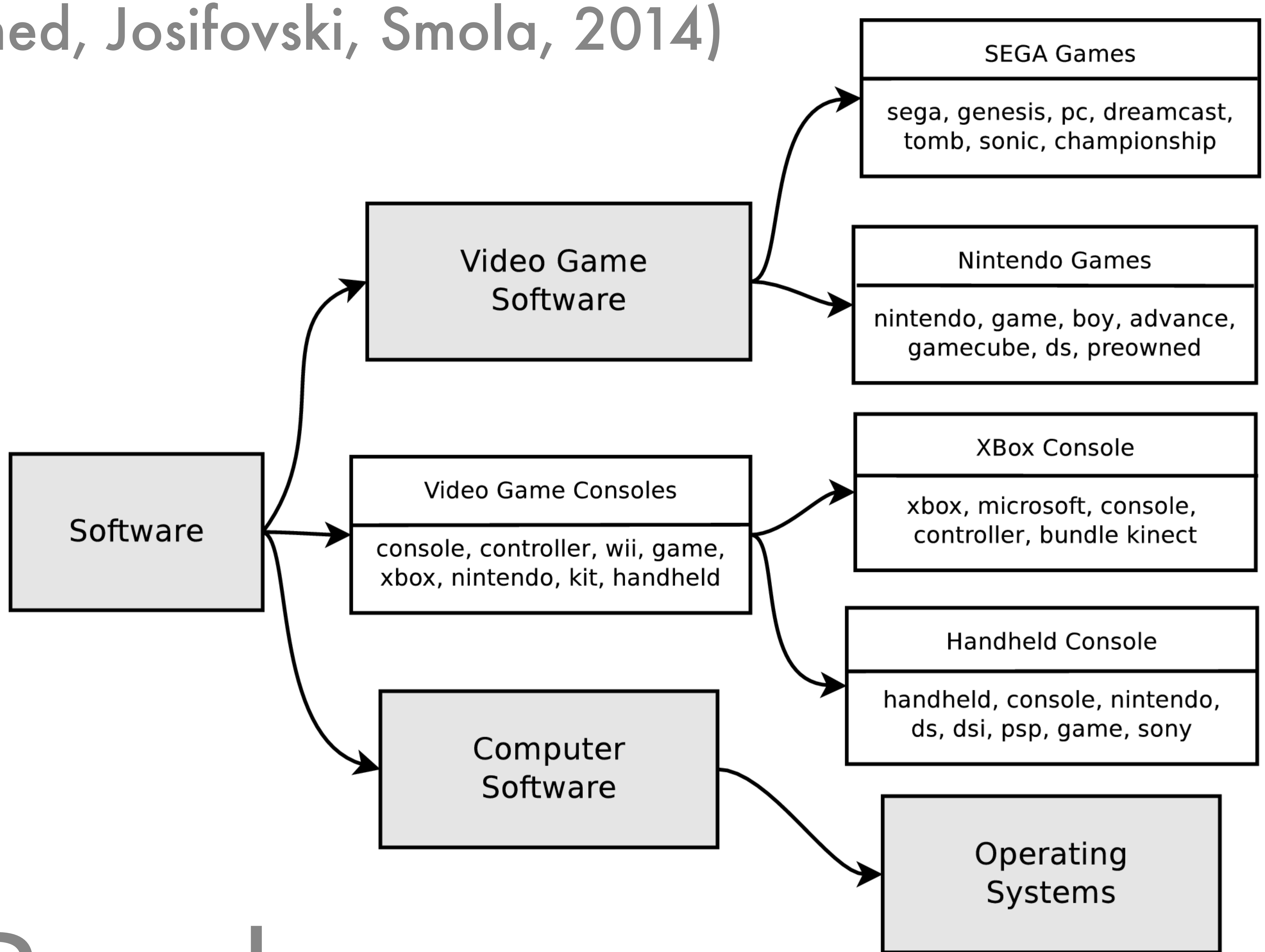


Results

Results on DS1	Avg. Error	Regions
(Yin et al., 2011)	150.06	400
(Hong et al., 2012)	118.96	1000
Approx.	91.47	2254
MH	90.83	2196
Exact	83.72	2051

Results on DS2	Avg. Error	Regions
(Eisenstein et al., 2010)	494	
(Wing & Baldridge, 2011)	479	
(Eisenstein et al., 2011)	501	
(Hong et al., 2012)	373	100
Approx.	298	836
MH	299	814
Exact	275	823

(Zhang, Ahmed, Josifovski, Smola, 2014)



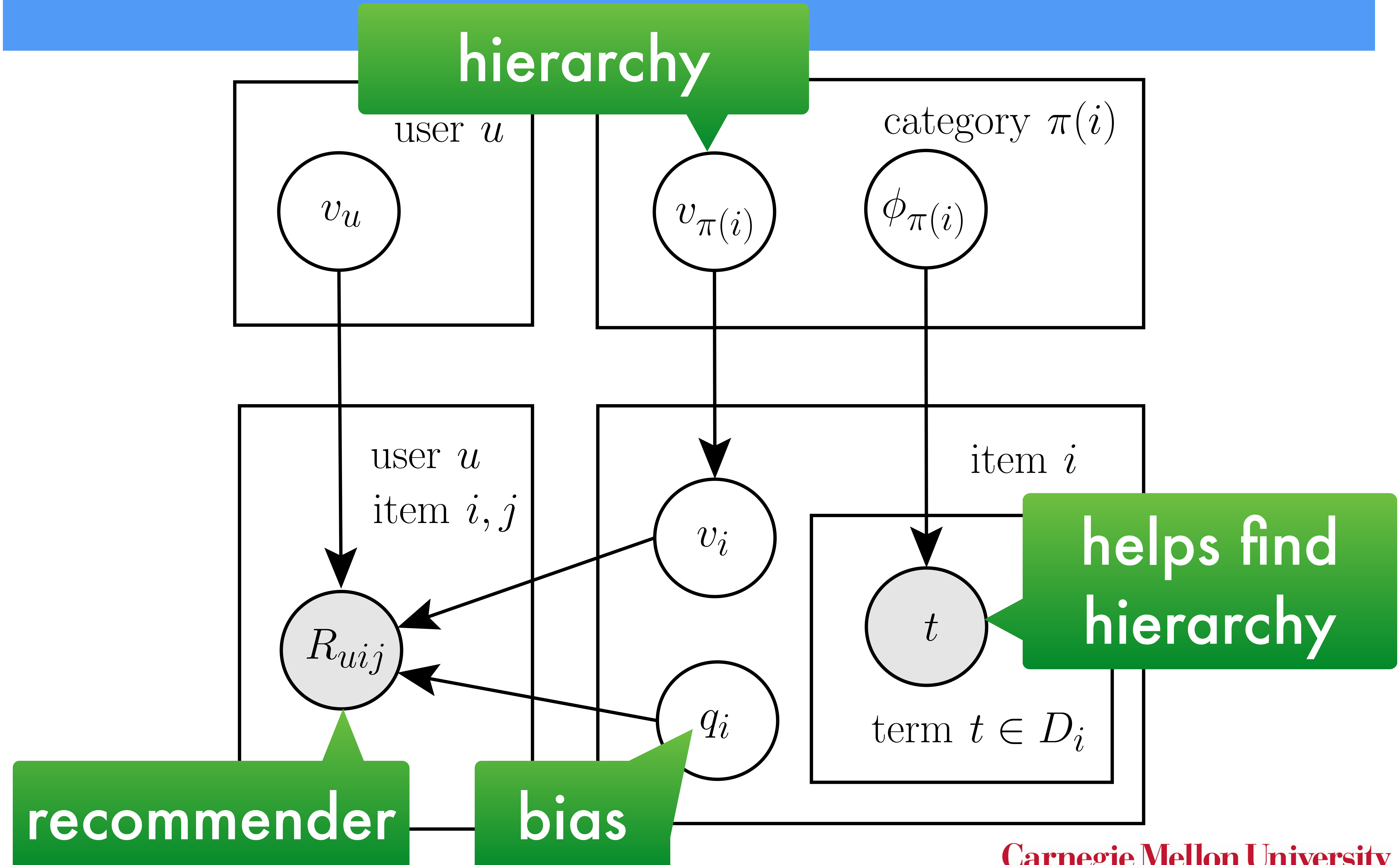
Purchase Recommendation

The Challenge

- Lots of (Co)purchase information per user
- Item metadata (brand, price, text)
- Recommend items

- Human generated taxonomy (modest cover)
  - Expensive to add items
  - Not always accurate for purchase (hardware->(ps3, xbox, wii), software->(ps3 games, xbox games, wii games)
- Want to recycle it if we have it

The model



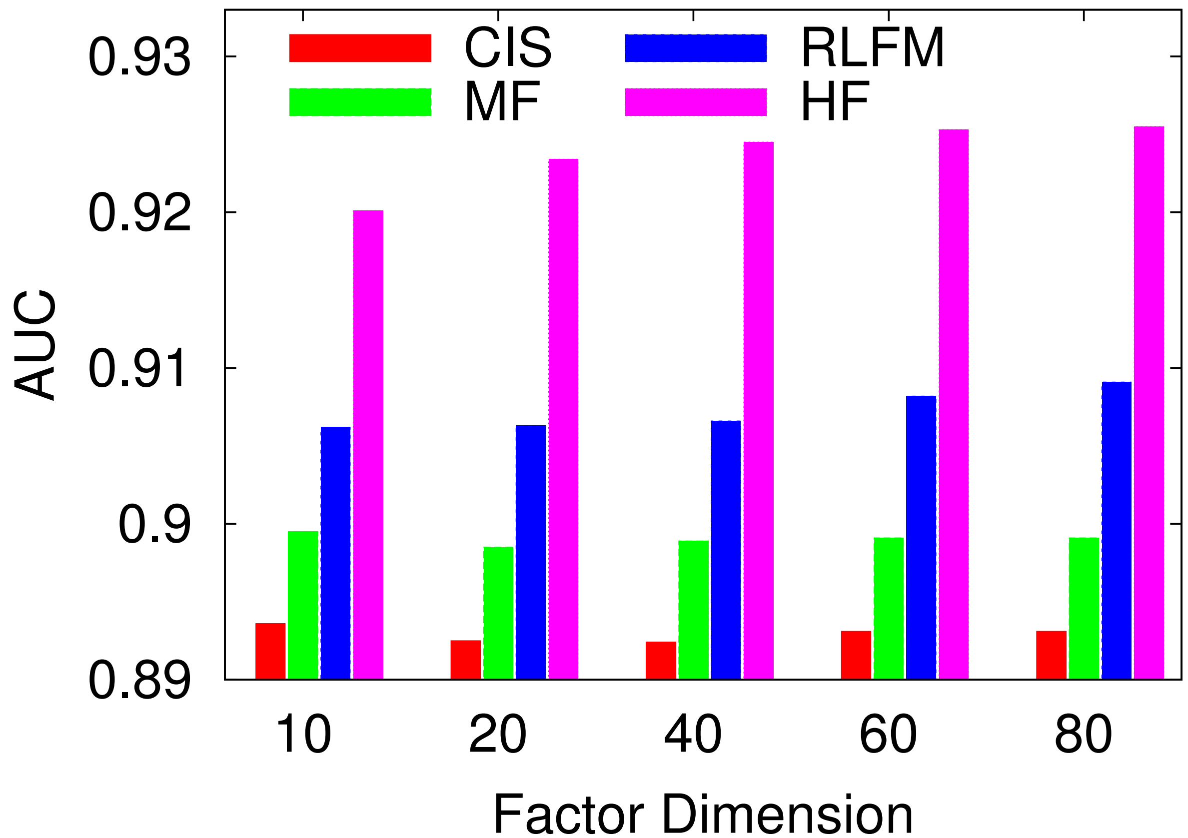
Carnegie Mellon University

Inference

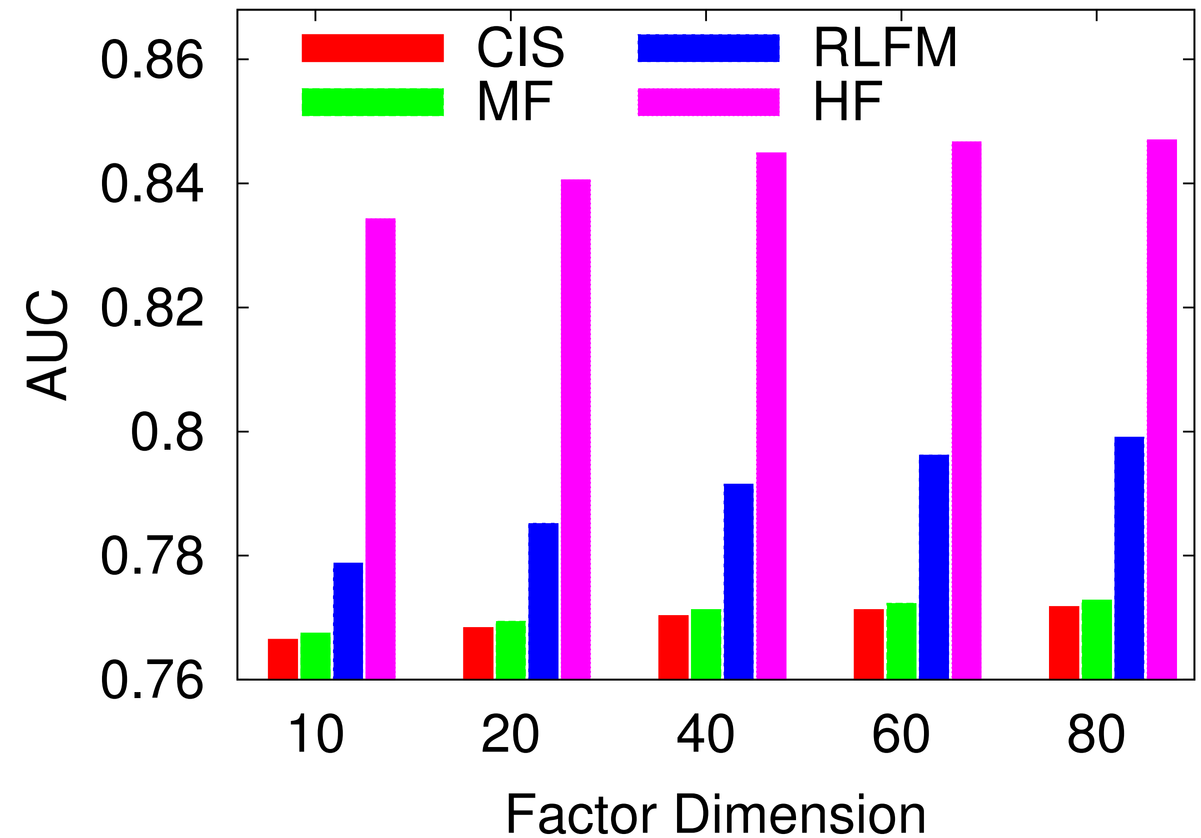
- Collapsed Gibbs sampling for hierarchy
- Stochastic gradient ascent to maximize recommendation likelihood
  - BPR style recommendation
  - Hierarchical parameter inheritance
- Incorporate existing human-generated ontology as prior on hierarchy

Results

dense data

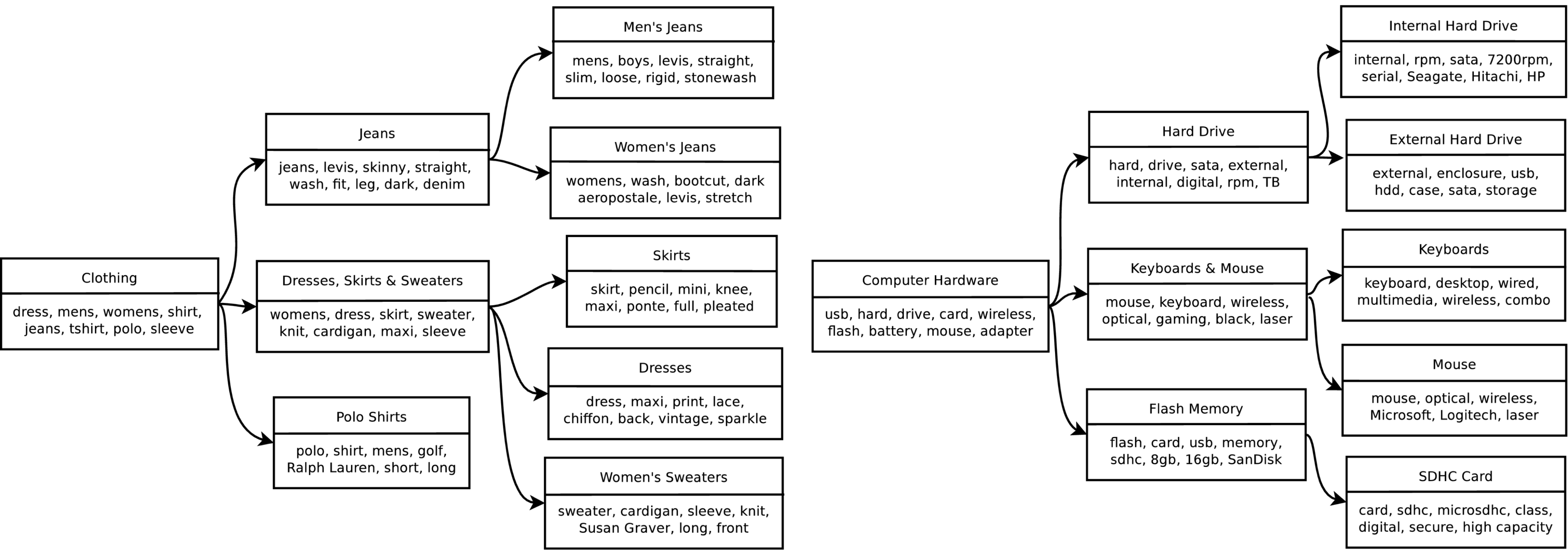


sparse data (singletons)



Item Frequency	1 - 10	11 - 30	31 - 100	101 - 300	301 - 1000	> 1000	Overall
MF	0.453	0.878	0.961	0.987	0.996	0.9996	0.899
CIS	0.444	0.860	0.948	0.982	0.995	0.9996	0.893
RLFM	0.529	0.863	0.957	0.987	0.996	0.9995	0.908
HF	**0.617***	**0.891***	**0.965***	**0.989**	**0.997**	**0.9996**	**0.925***

Inferred taxonomy



Google

Carnegie Mellon University

News

EXPRESS
THE WORLD'S GREATEST NEWSPAPER
Migrant controls are just big con
SEE PAGE FIVE
WIN A NIKON DIGITAL CAMERA
WORTH £500 EACH SET
Mud - tastic
Festival round up Page 3
REGIONAL NEWSPAPER OF THE YEAR
POST
WWW.BIRMINGHAMPOST.NET
Five die as aircraft collide
theguardian
Backlash over Blair's school revolution
City academy plans condemned by ex-education secretary Morris
Second night of rioting
Column five The shape of things to come
FT Business
DAILY Mirror
REBECCA'S TV BOMBSHELLS
Beckham has got
What he wrote
After sex he fed me strawberries
Posh pictures are
STILL TIME TO JOIN IN EURO HOLS FROM £9.50
30p
Baby-faced father of little Maisie
DAD AT
METRO
'Online hoodies'
THE TIM
NEWSPAPER OF THE YEAR
How to survive a

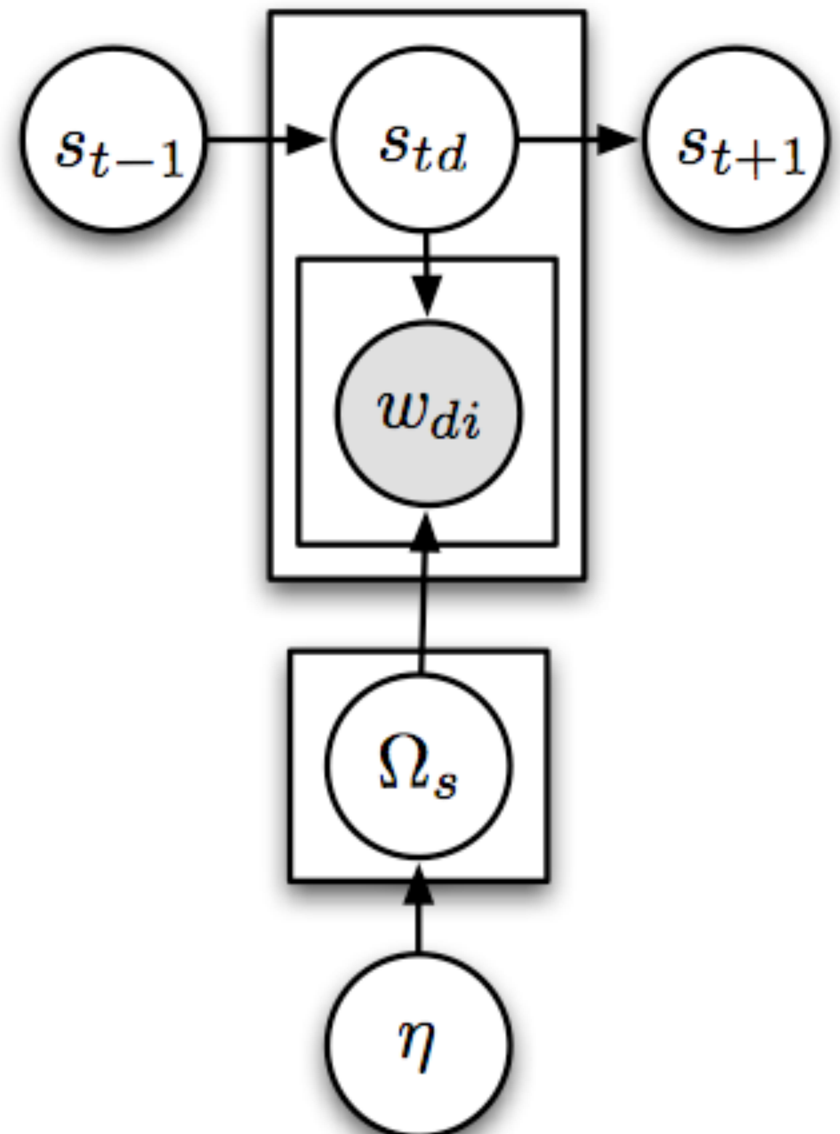
News Stream

- Over 1 high quality news article per second (many orders of magnitude more for UGC)
- Multiple sources (Reuters, AP, CNN, ...)
- Same story from multiple sources
- Stories are related

- Goals
  – Aggregate articles into a storyline
  – Analyze the storyline (topics, entities)

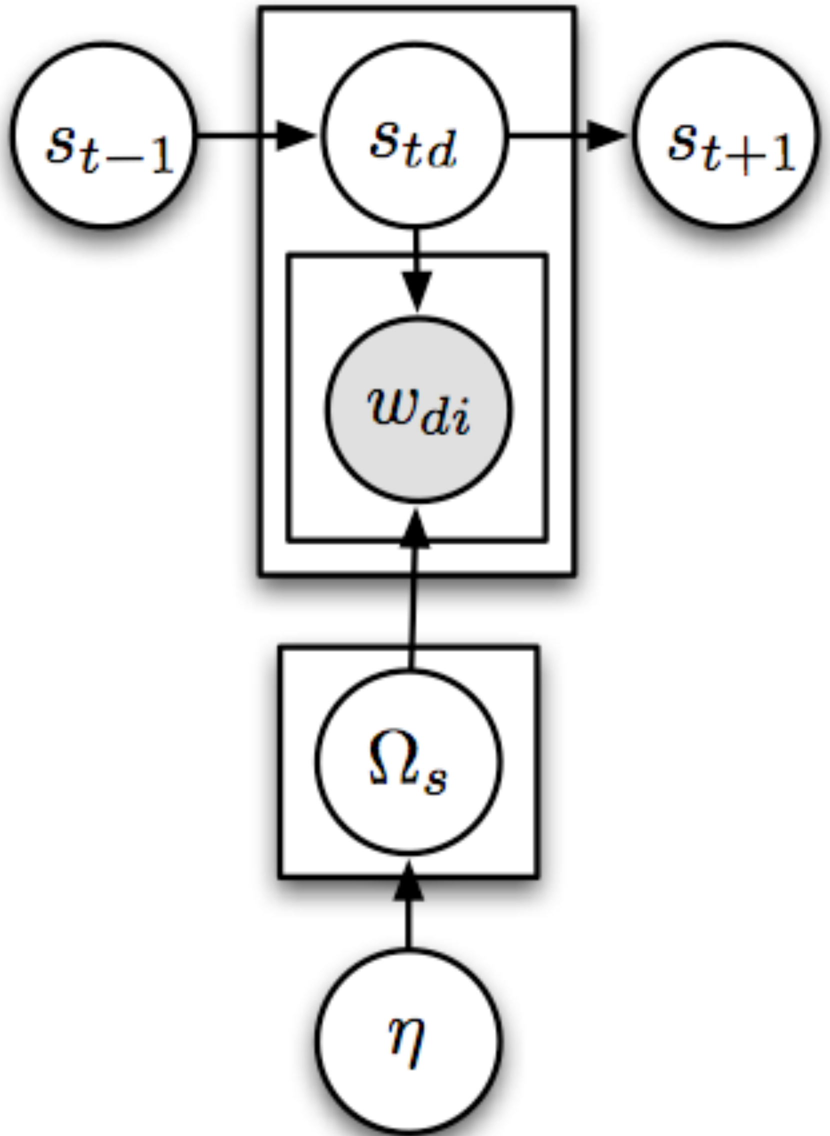
Clustering / RCRP

- Assume active story distribution at time t
- Draw story indicator
- Draw words from story distribution
- Down-weight story counts for next day
  Ahmed & Xing, 2008



Clustering / RCRP

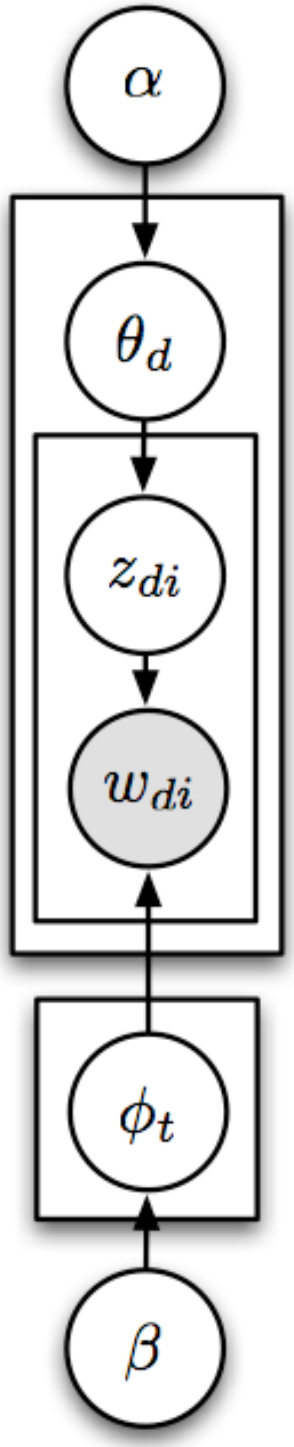
- Assume active story distribution at time t
- Draw story indicator
- Draw words from story distribution
- Down-weight story counts for next day
  Ahmed & Xing, 2008



- Pro
  - Nonparametric model of story generation
  - No fixed number of stories
  - Efficient inference via collapsed sampler
- Con
  - We learn nothing!
  - No content analysis

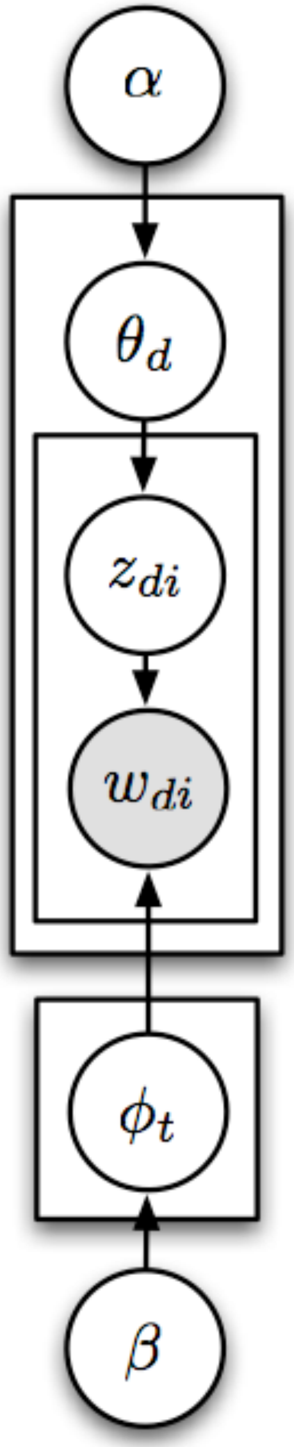
Latent Dirichlet Allocation

- Generate topic distribution per article
- Draw topics per word from topic distribution
- Draw words from topic specific word distribution
  Blei, Ng, Jordan, 2003



Latent Dirichlet Allocation

- Generate topic distribution per article
- Draw topics per word from topic distribution
- Draw words from topic specific word distribution
  Blei, Ng, Jordan, 2003
- Pro
  - Topical analysis of stories
  - Topical analysis of words (meaning, saliency)
  - More documents improve estimates
- Con
  - No clustering

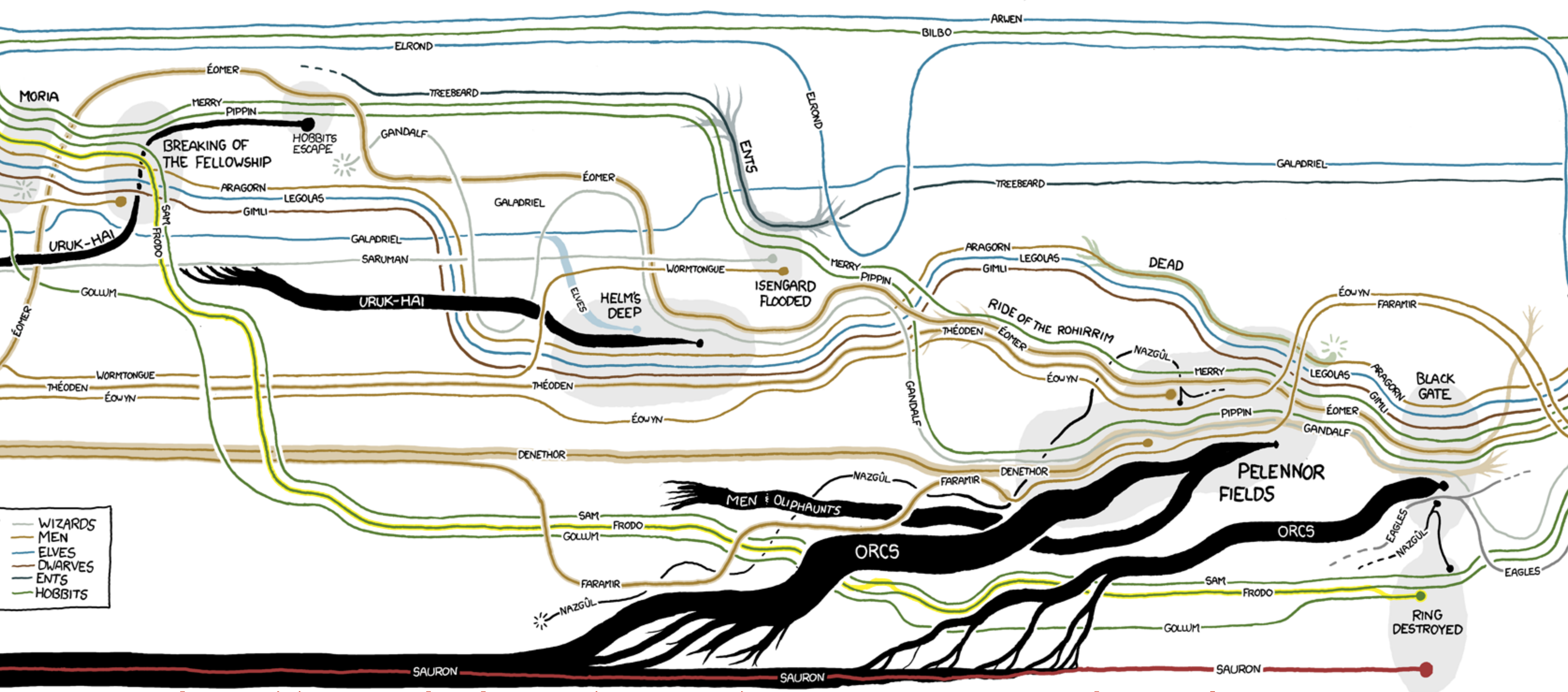


More Issues

- Named entities are special, topics less
  (e.g. Tiger Woods and his mistresses)
- Some stories are strange
  (topical mixture is not enough - dirty models)
- Articles deviate from general story
  (Hierarchical DP)

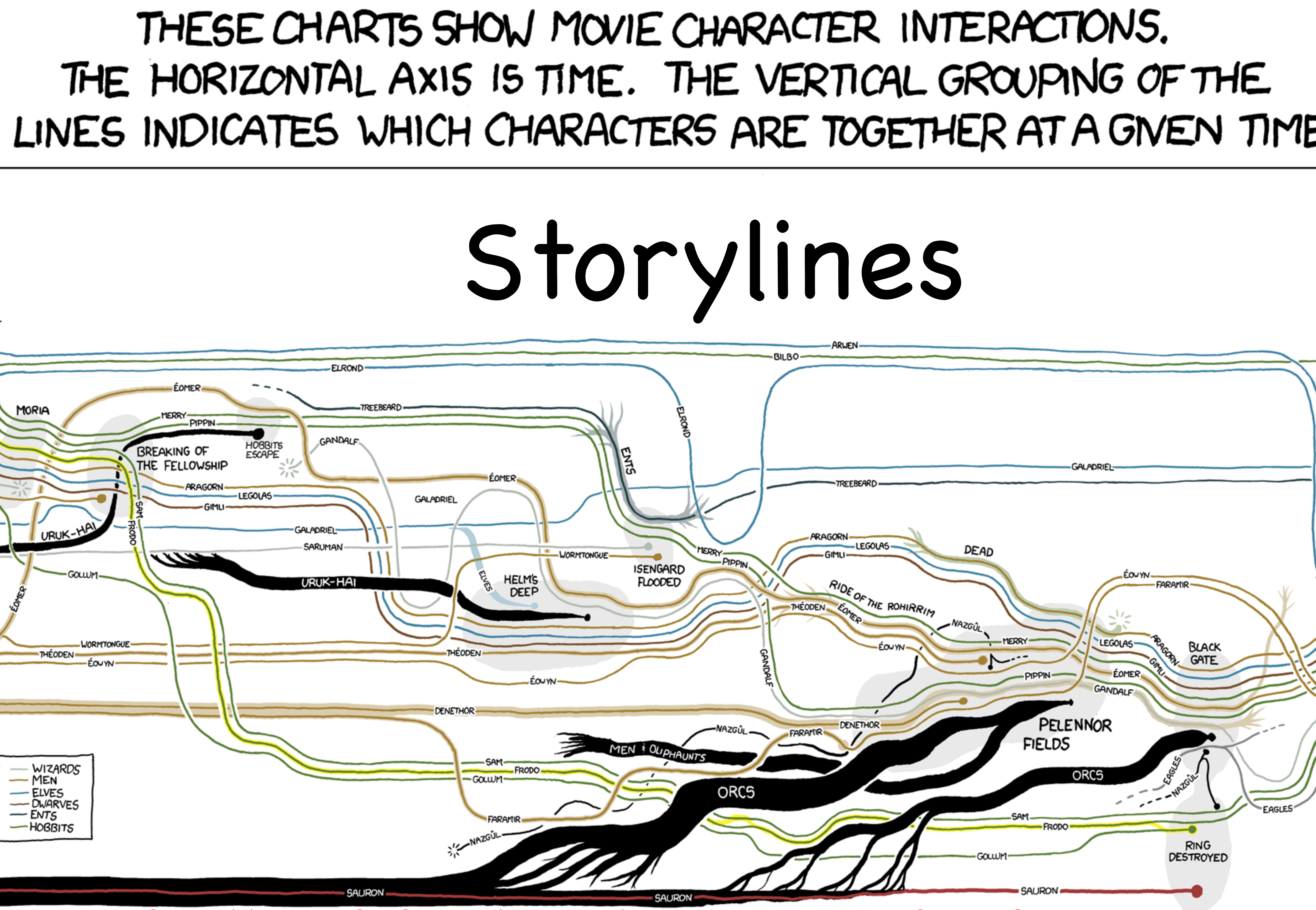
THESE CHARTS SHOW MOVIE CHARACTER INTERACTIONS. THE HORIZONTAL AXIS IS TIME. THE VERTICAL GROUPING OF THE LINES INDICATES WHICH CHARACTERS ARE TOGETHER AT A GIVEN TIME

LORD OF THE RINGS



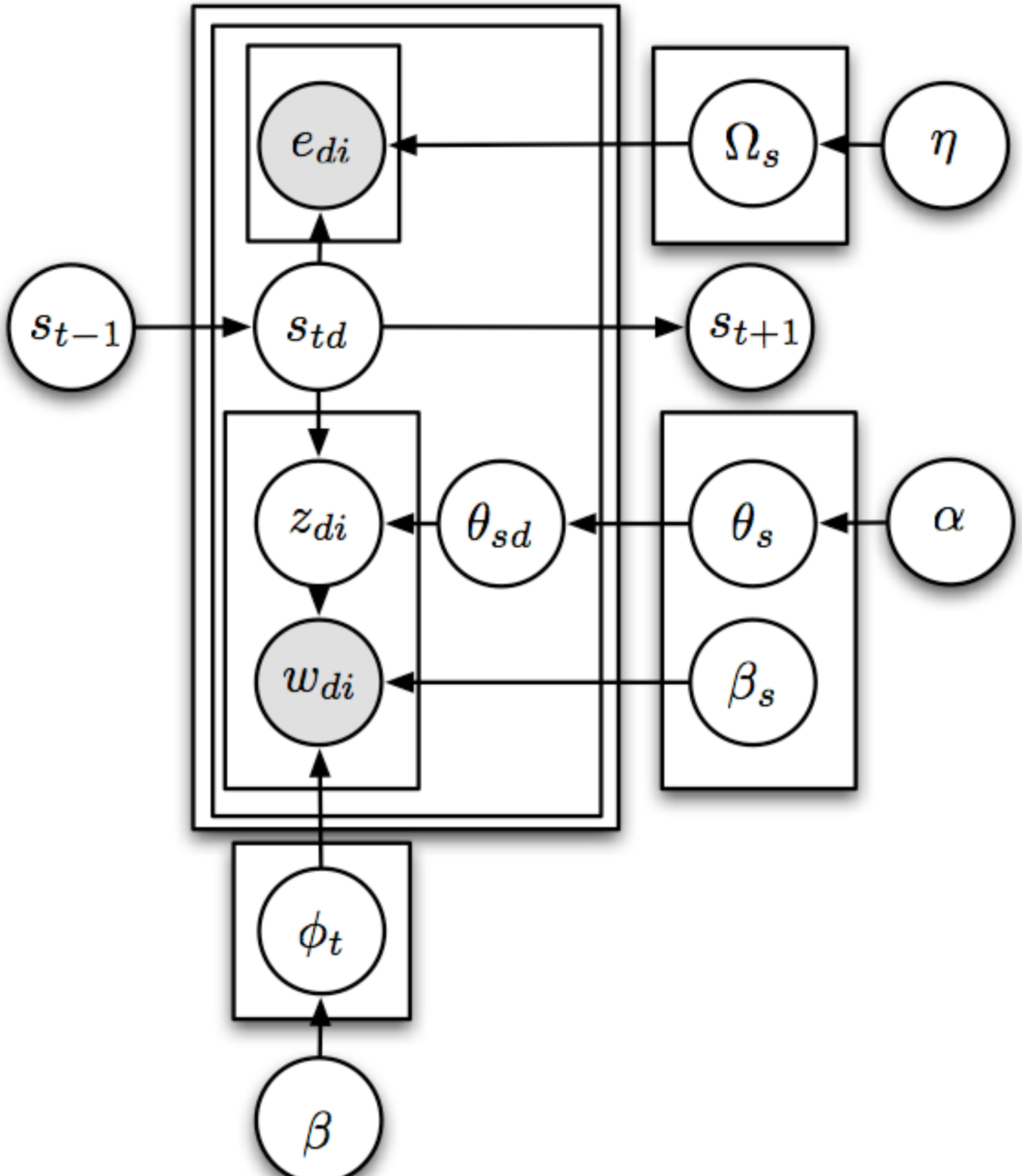
http://imgs.xkcd.com/comics/movie_narrative_charts_large.png

Storylines



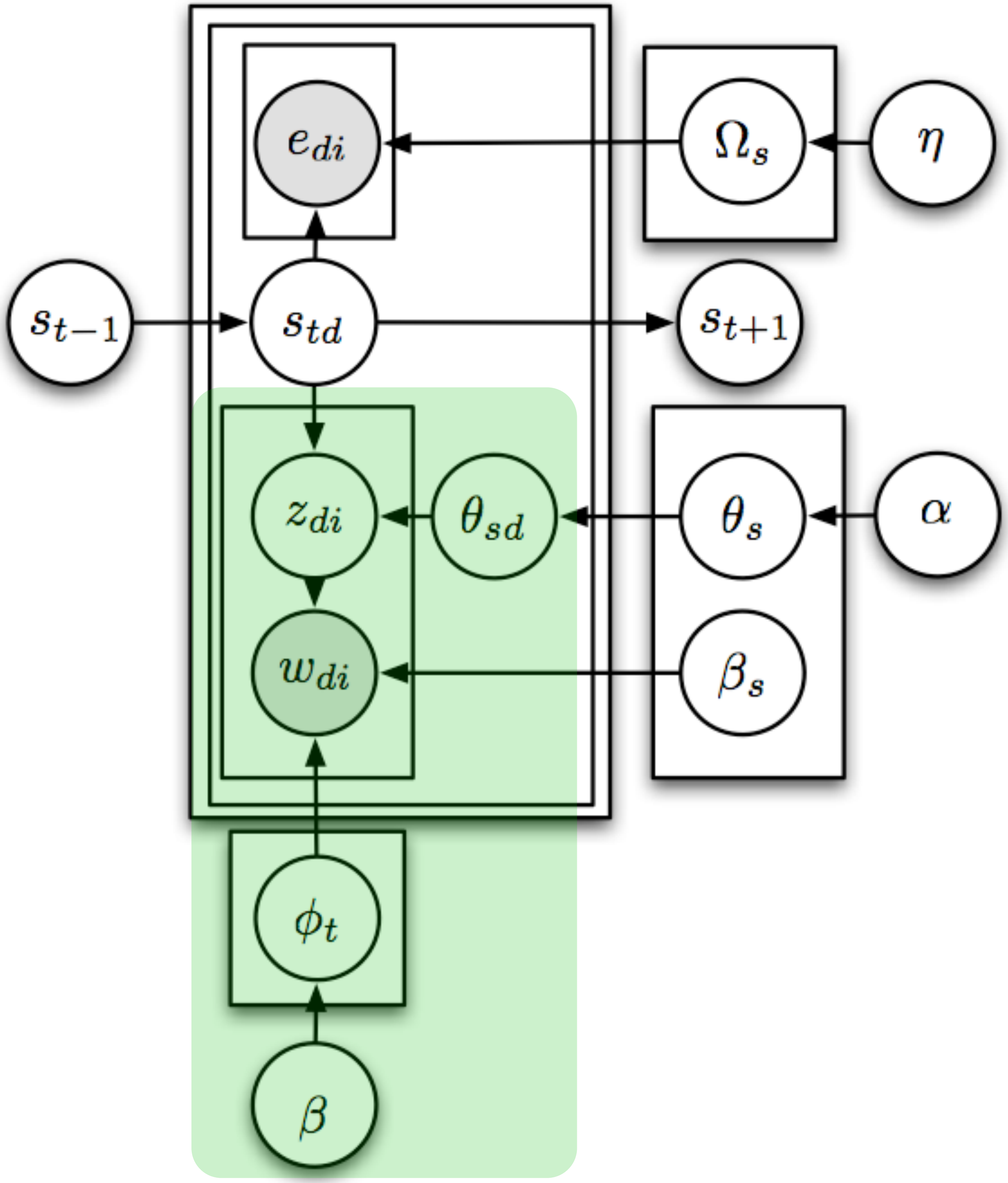
http://imgs.xkcd.com/comics/movie_narrative_charts_large.png

Storylines Model



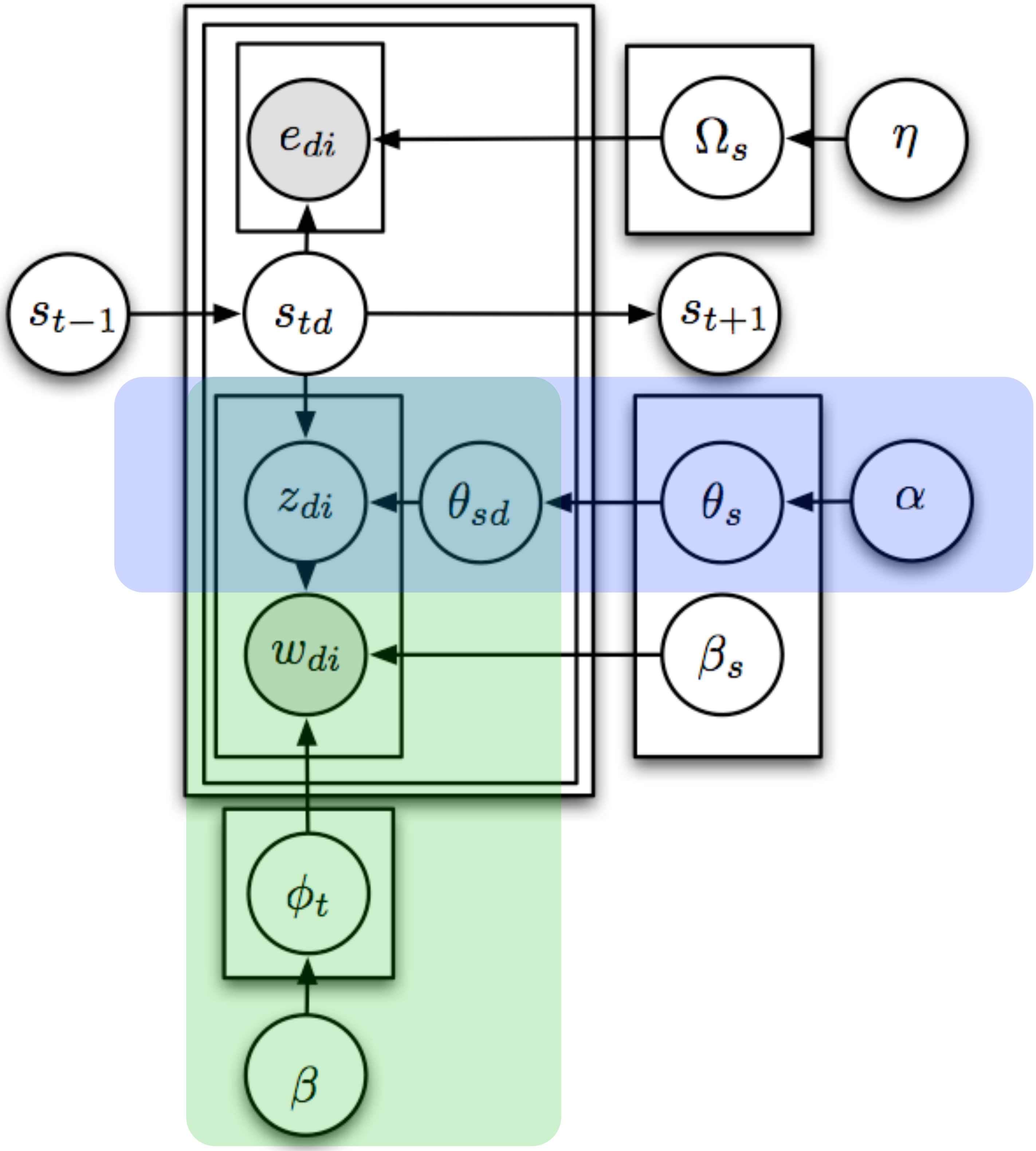
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model



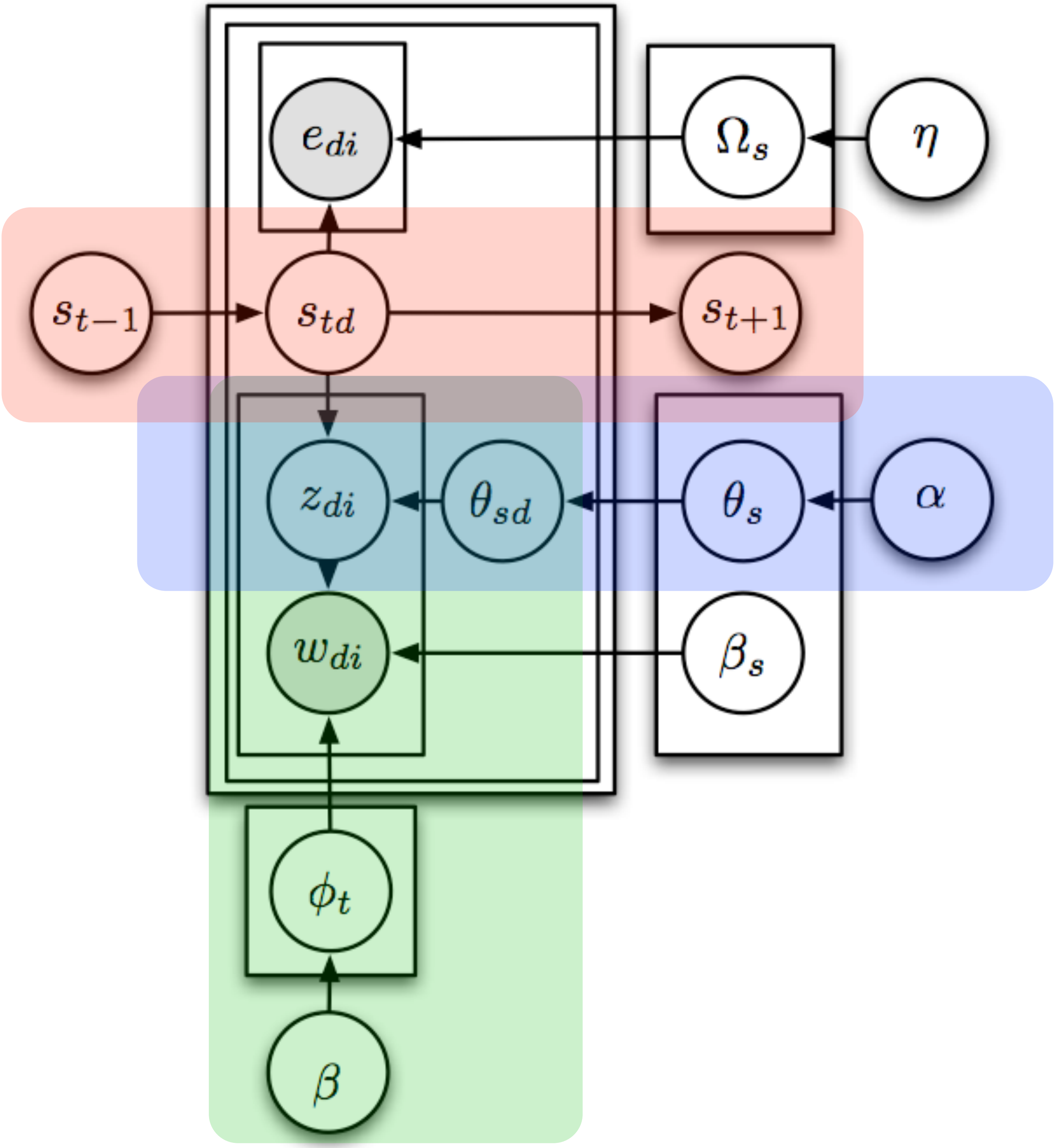
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model



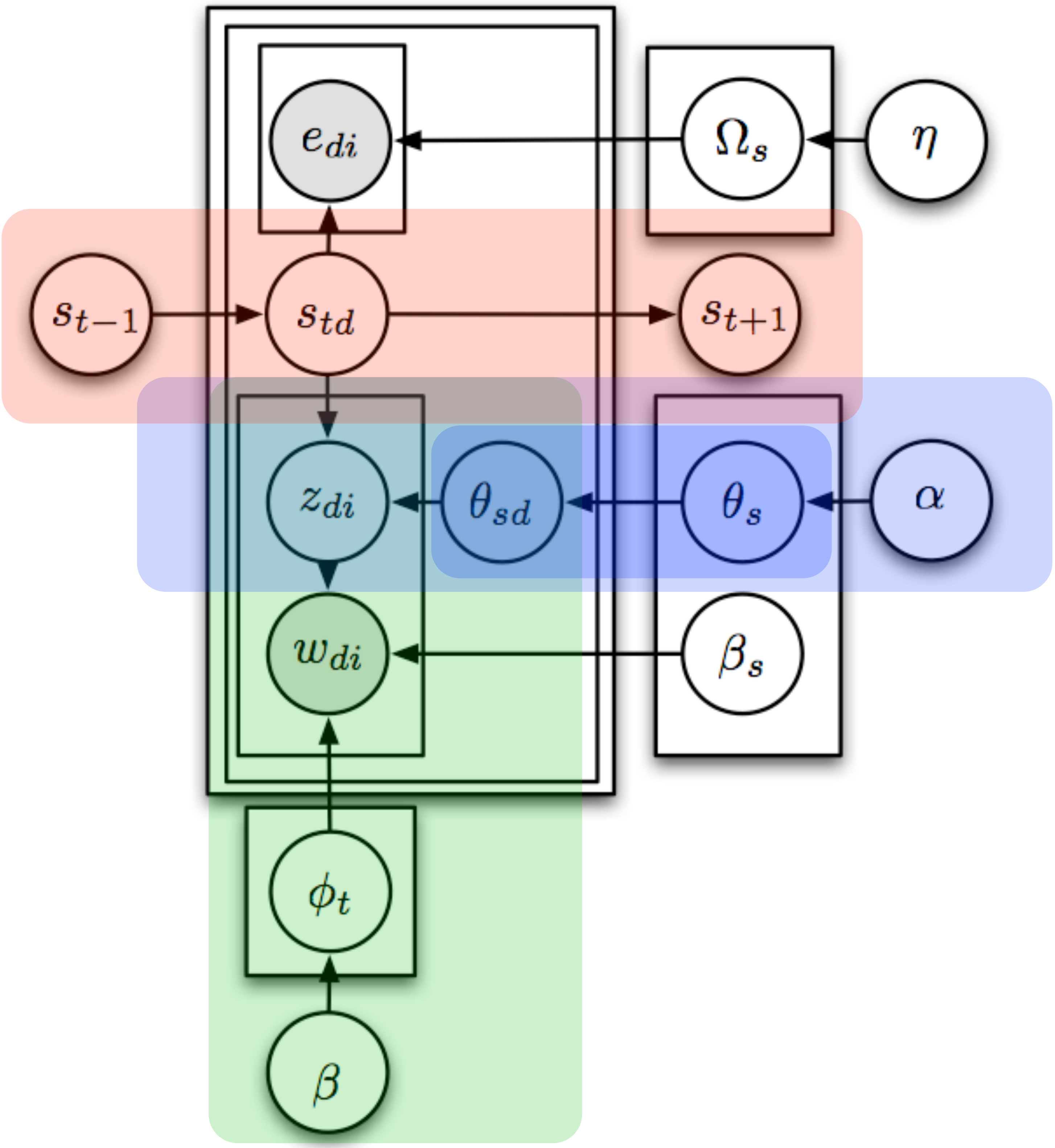
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model



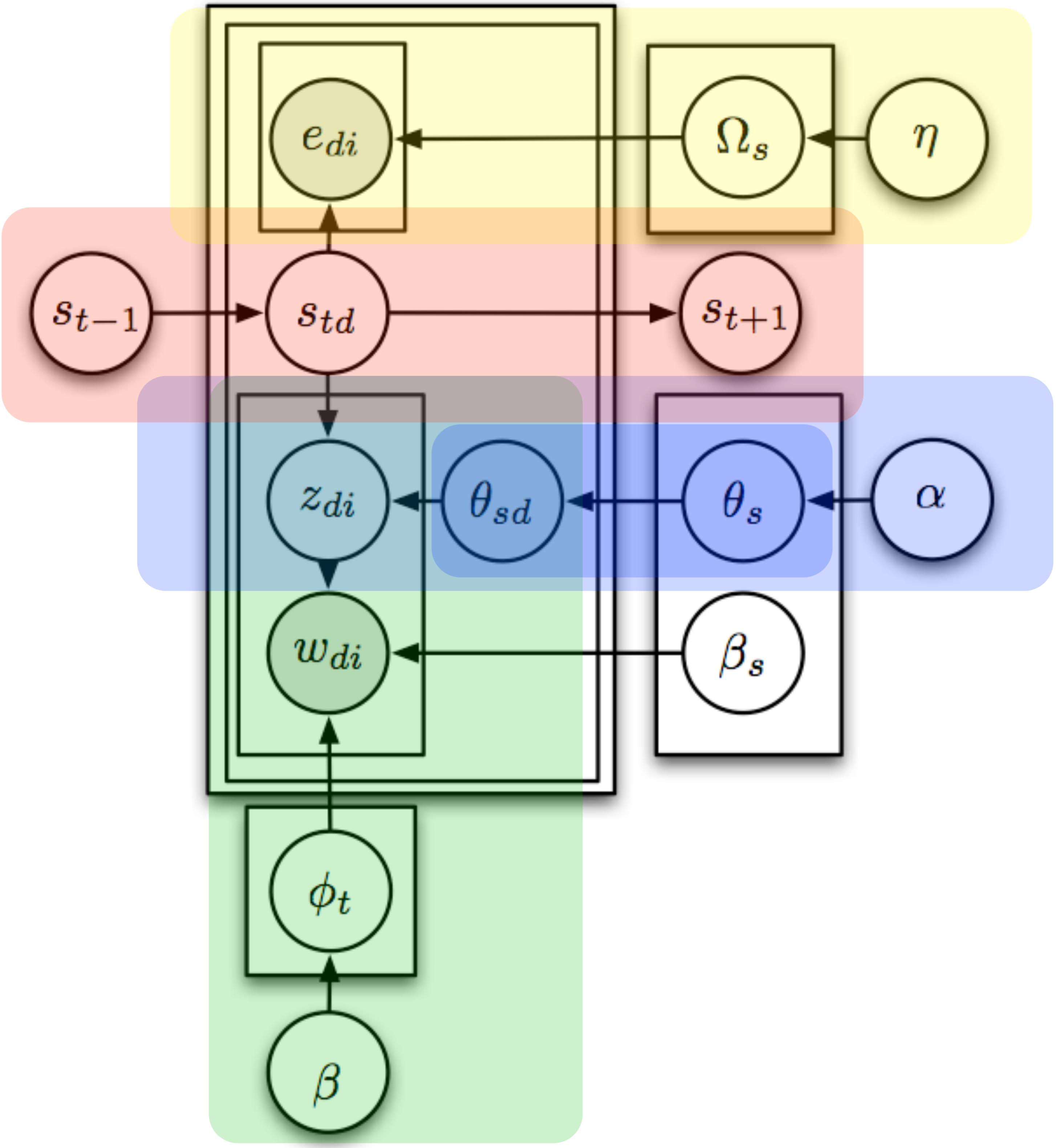
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model



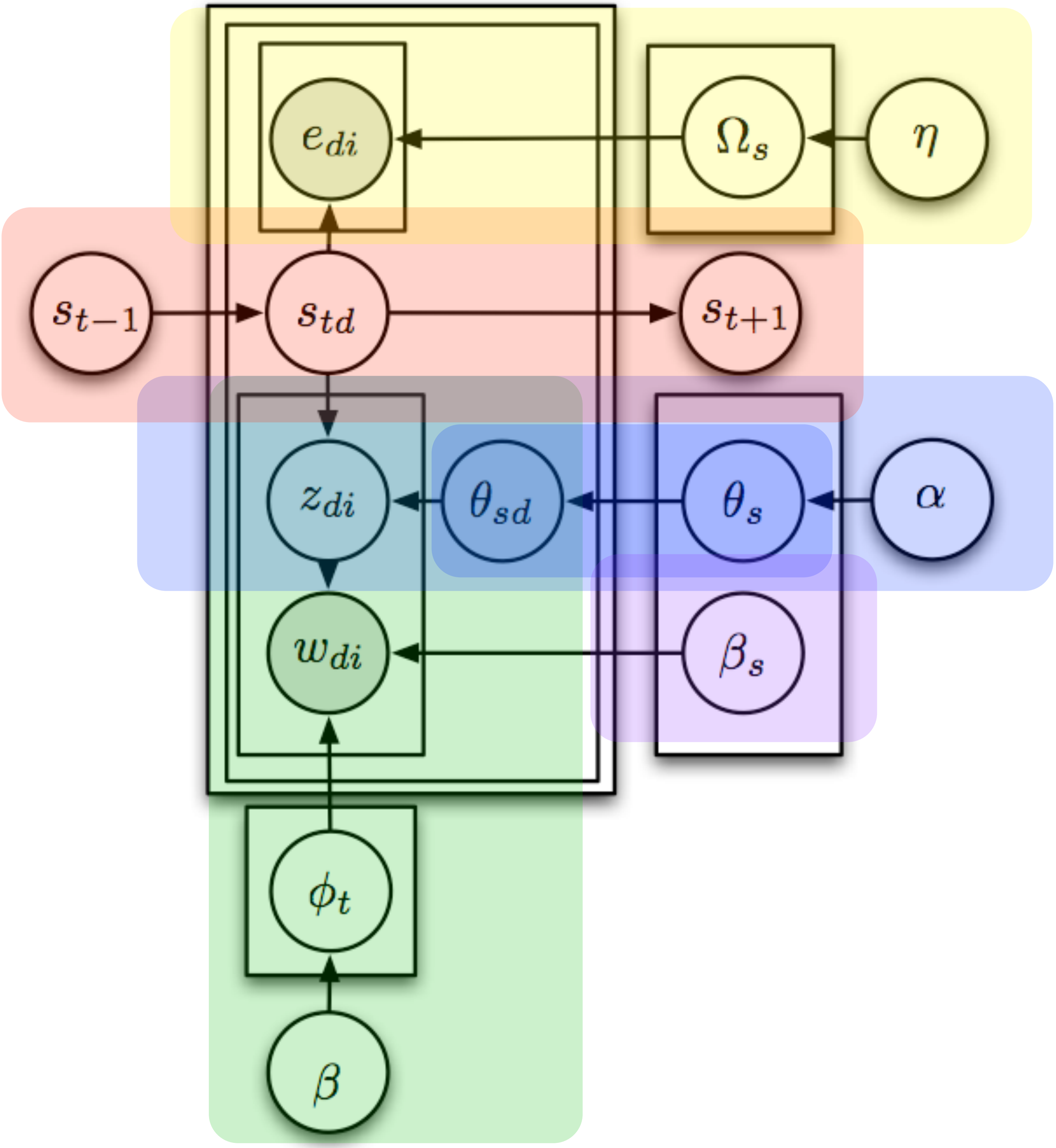
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model



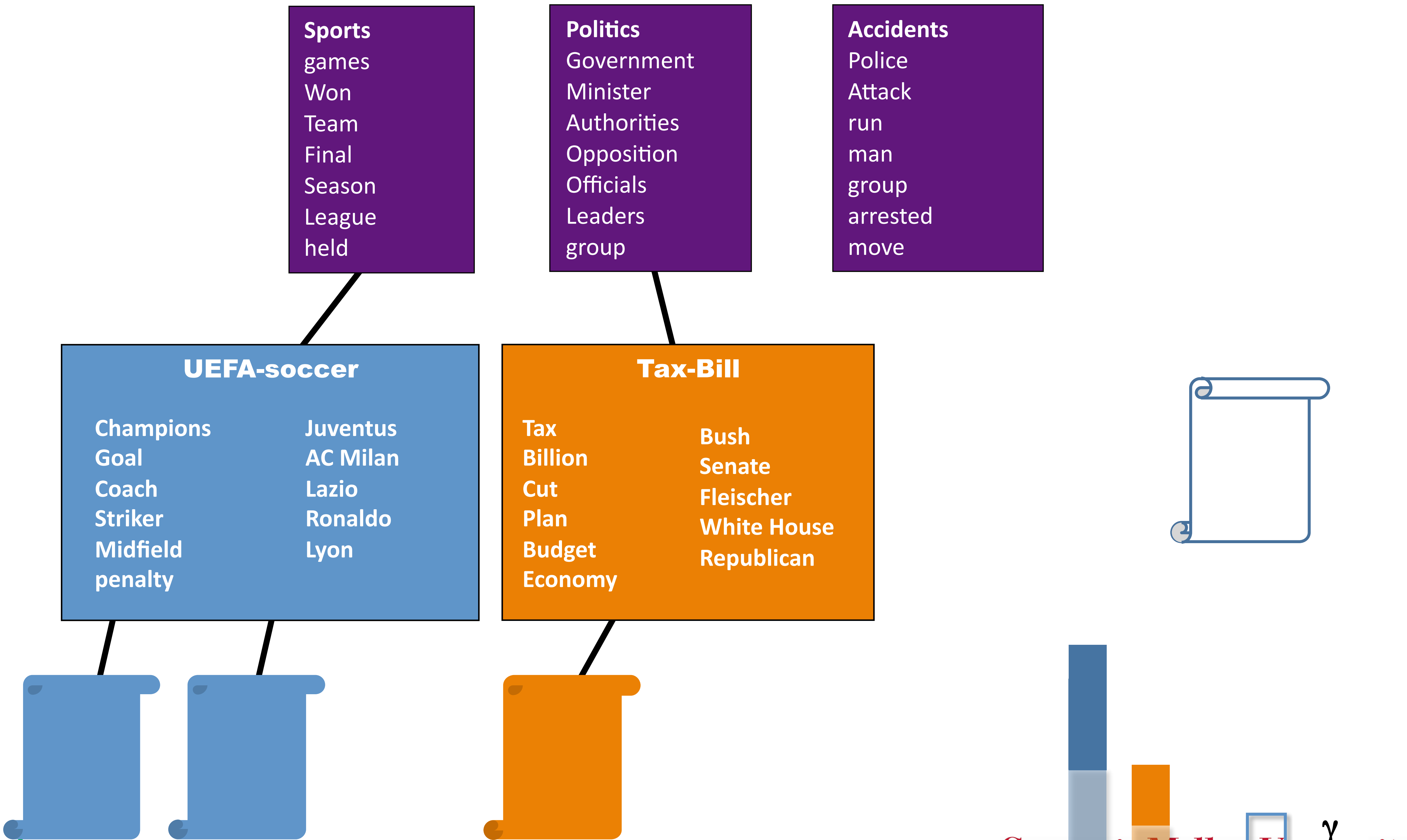
- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

Storylines Model

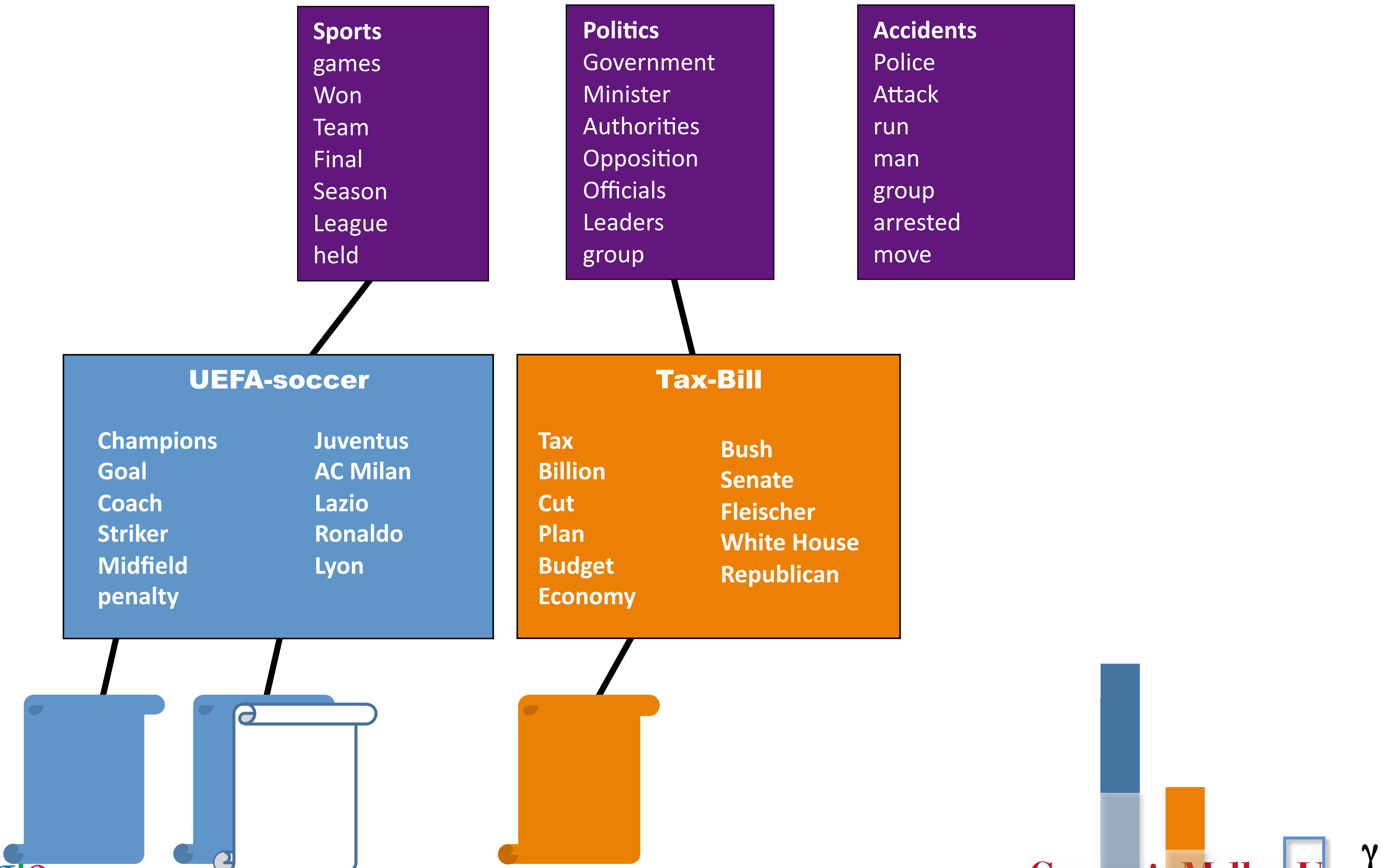


- Topic model
- Topics per cluster
- RCRP for cluster
- Hierarchical DP for article
- Separate model for named entities
- Story specific correction

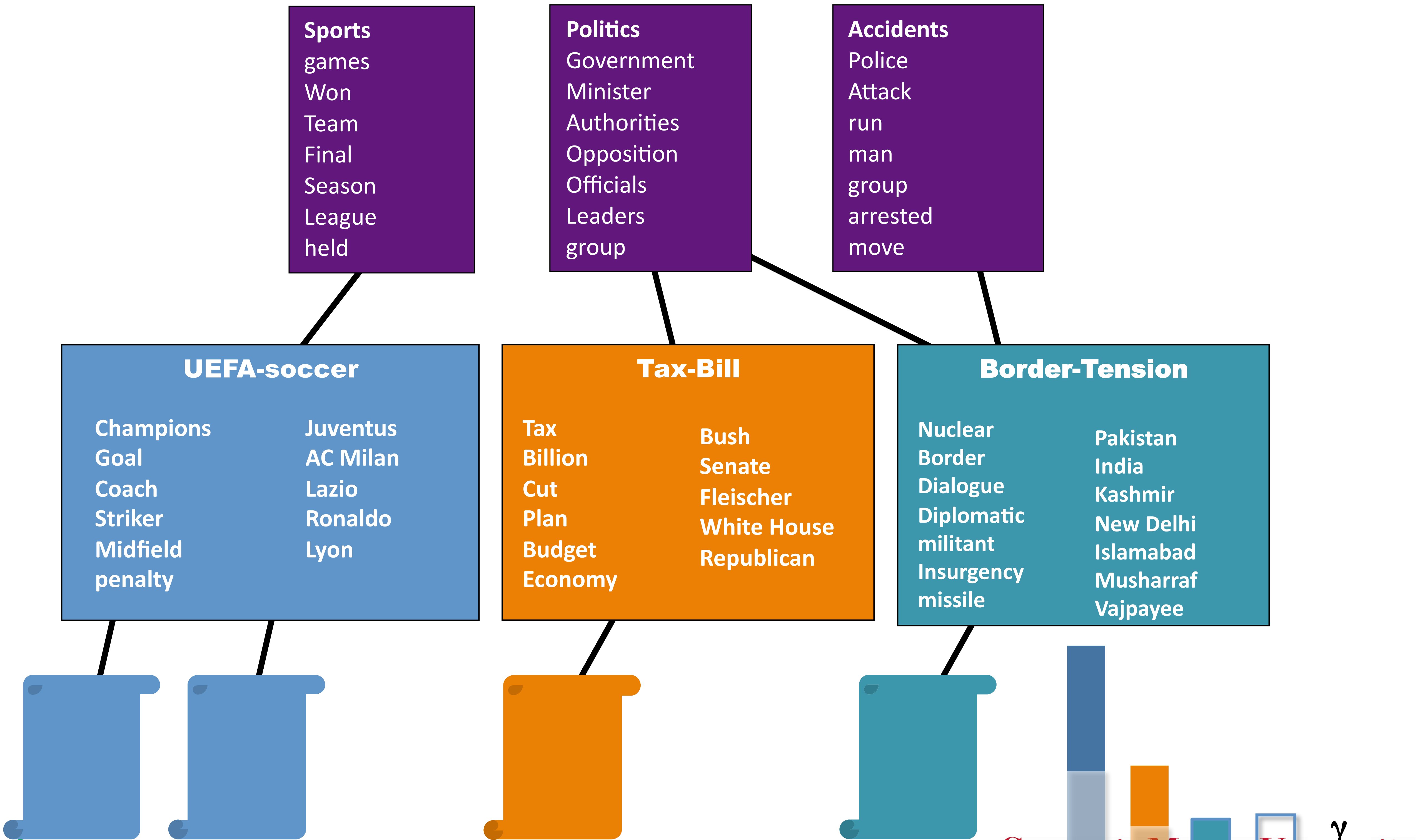
Dynamic Cluster-Topic Hybrid



Dynamic Cluster-Topic Hybrid



Dynamic Cluster-Topic Hybrid



Inference

- We receive articles as a stream
  Want topics & stories now
- Variational inference infeasible
  (RCRP, sparse to dense, vocabulary size)
- We have a 'tracking problem'
  - Sequential Monte Carlo
  - Use sampled variables of surviving particle
  - Use ideas from Cannini et al. 2009

Particle Filter

- Proposal distribution - draw stories s, topics z

$$p(s_{t+1}, z_{t+1} | x_{1 \ldots t+1}, s_{1 \ldots t}, z_{1 \ldots t})$$

using Gibbs Sampling for each particle

- Reweight particle via

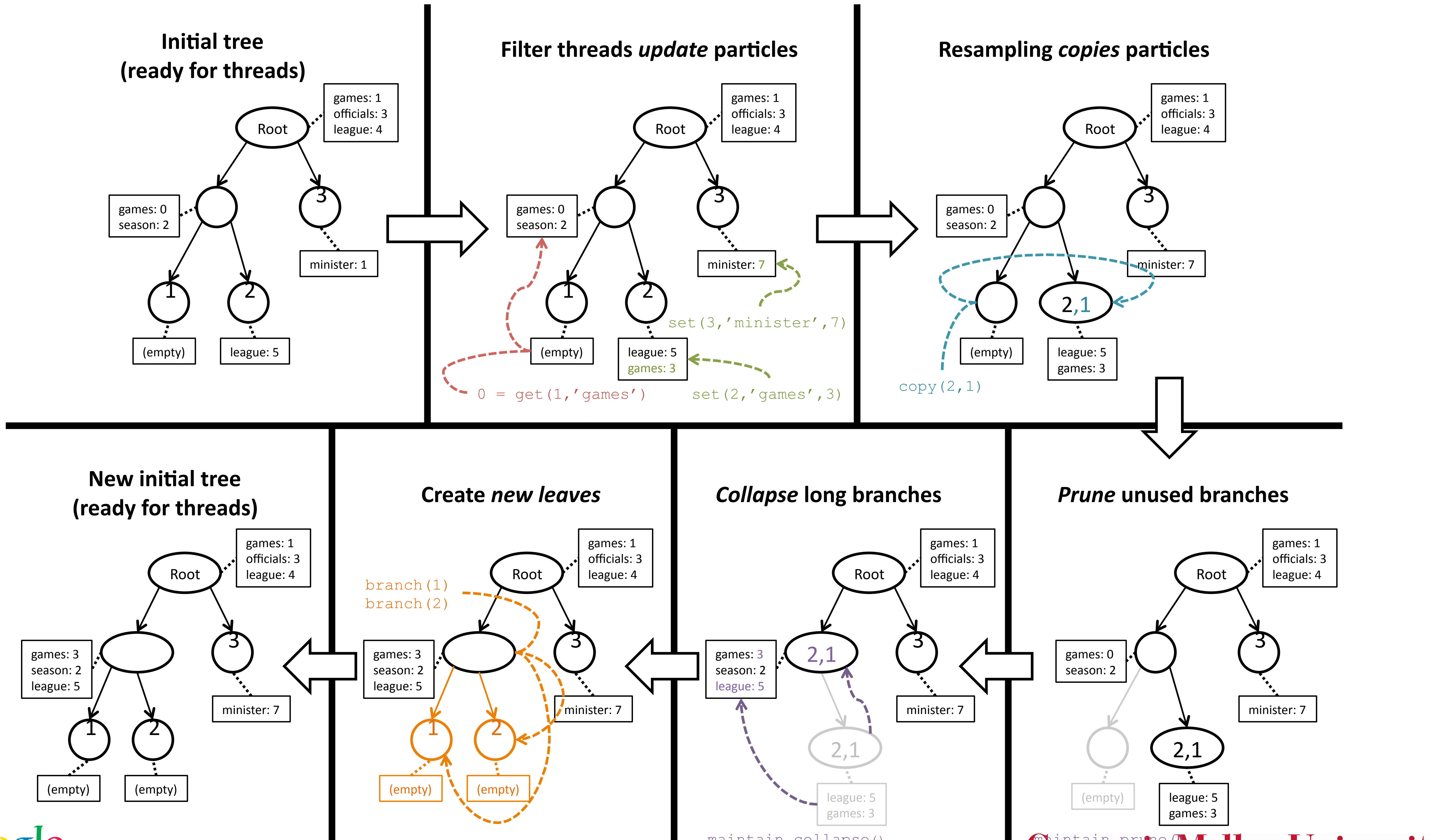
$$p(x_{t+1} | x_{1 \ldots t}, s_{1 \ldots t}, z_{1 \ldots t})$$

new data

past state

- Resample particles if l_2 norm too large (resample some assignments for diversity, too)
- Compare to multiplicative updates algorithm In our case predictive likelihood yields weights

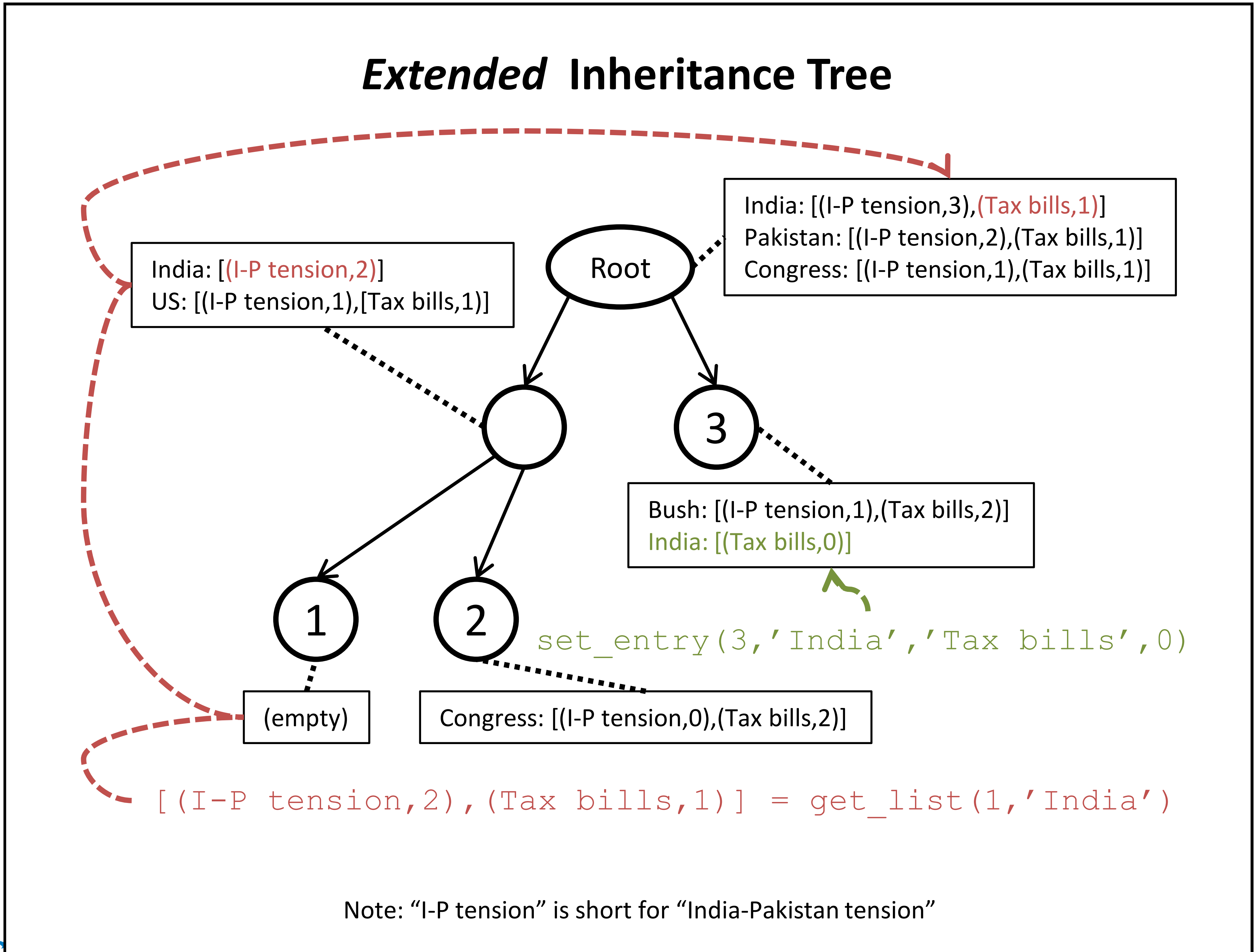
Inheritance Tree



Google

Carnegie Mellon University

Extended Inheritance Tree

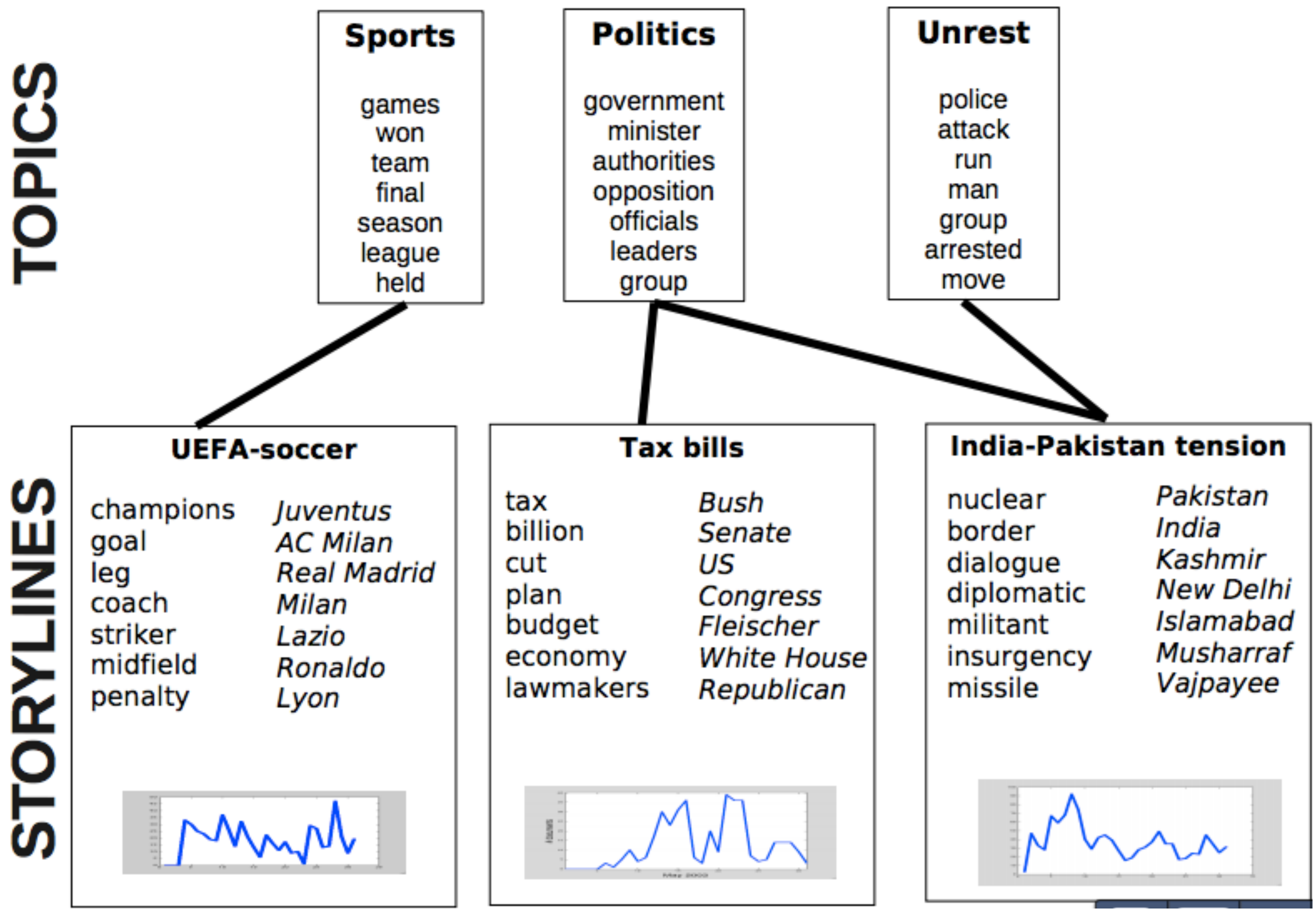


write only in the leaves (per thread)

Google

Carnegie Mellon University

Stories & Topics



Google

Carnegie Mellon University