MACHINE LEARNING
DEPARTMENT

Scalable Machine Learning

Models, Architectures and Algorithms

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

Google

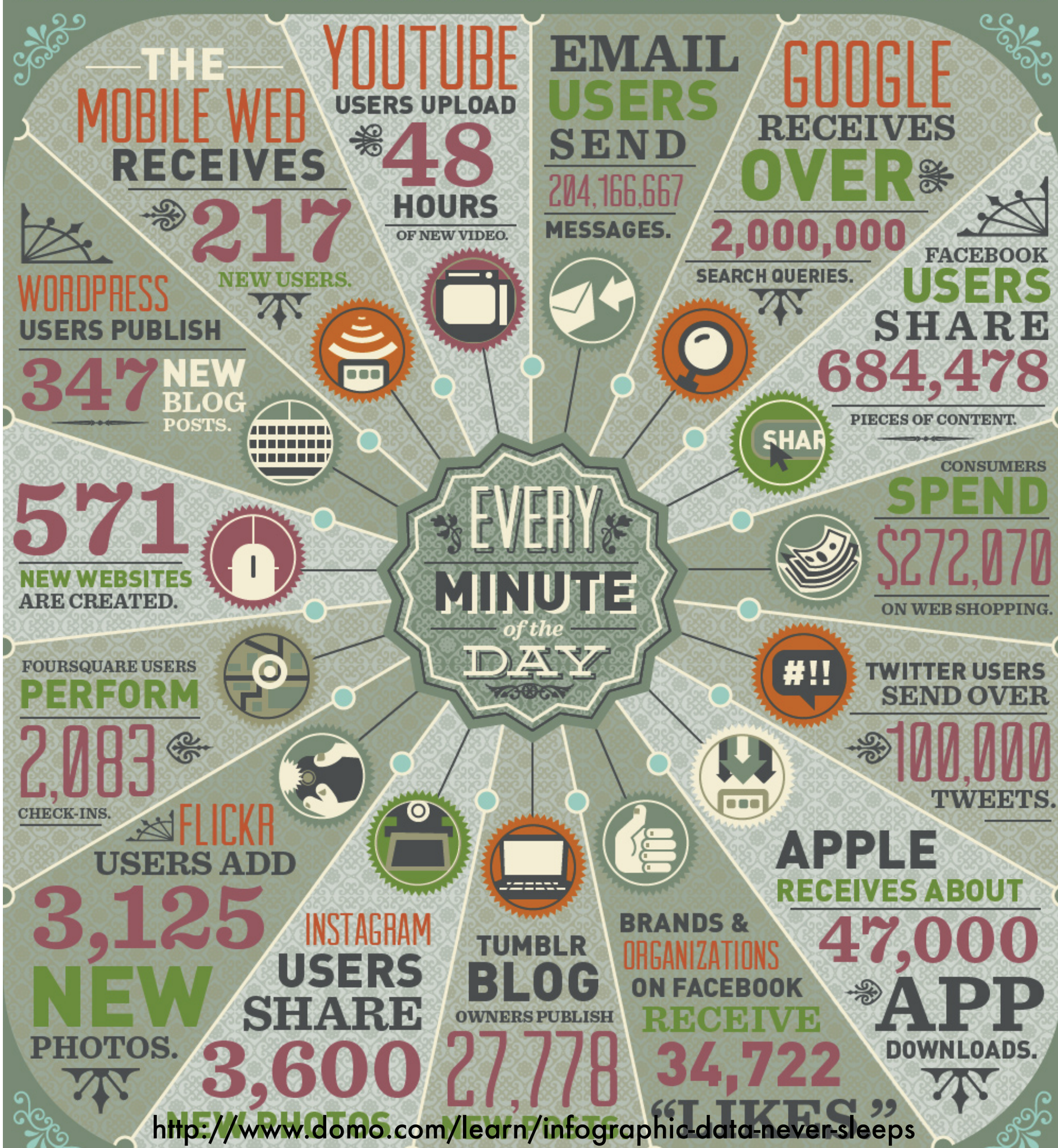
Carnegie Mellon University

Outline

- **Data**
  Actions, Interactions, User generated content
- **Architectures**
  MapReduce, Graphs, Streams, Parameterserver
- **Models and Algorithms**
  - Logistic regression (advertising, search)
    Distributed proximal gradient
  - Topic models (personalization, profiling)
    Stochastic variational
  - Randomized algorithms for large models
    Hash kernel, Fastfood, Alias sampling, Sketches


Data

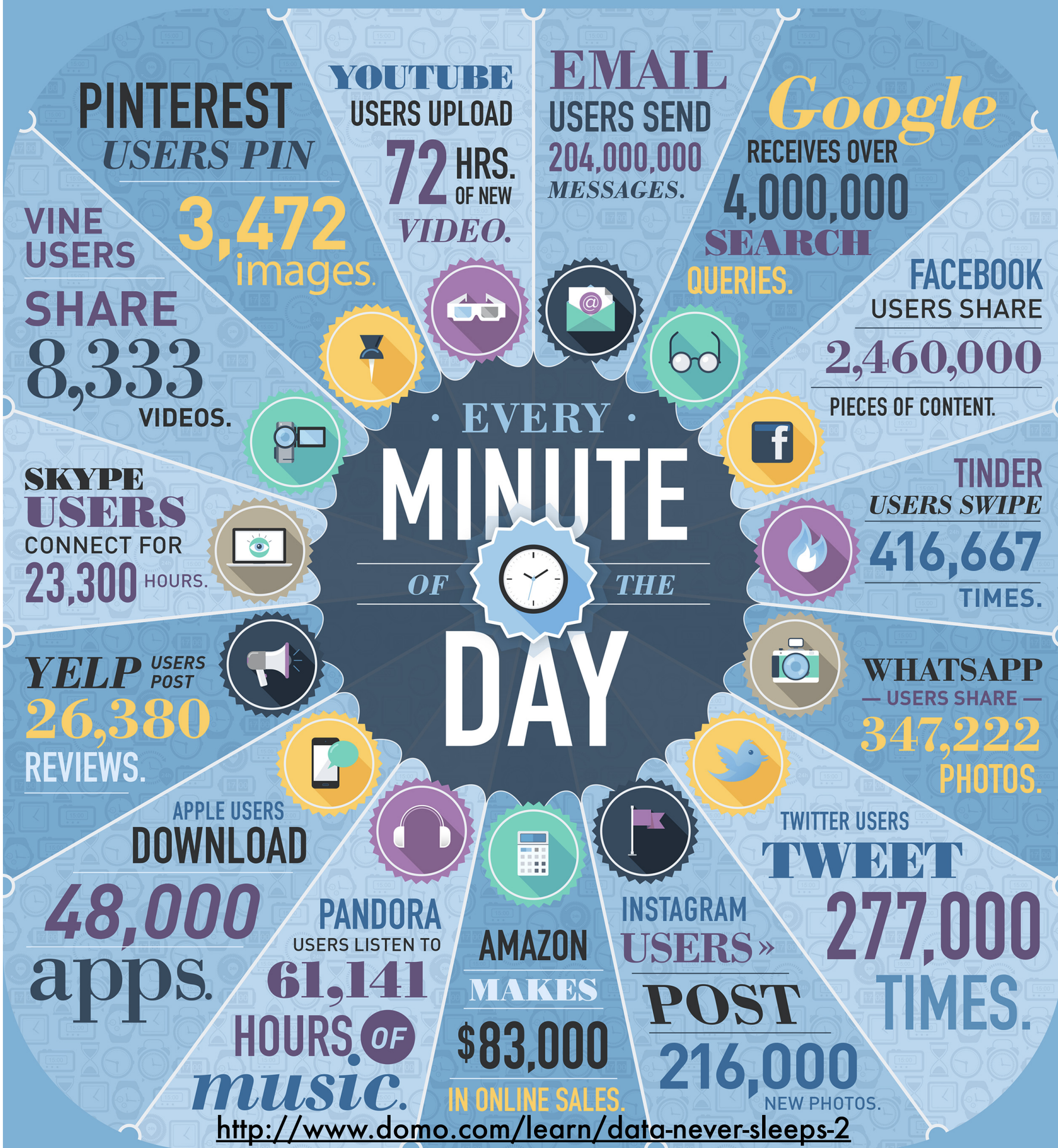
Data per minute 2012



Google

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

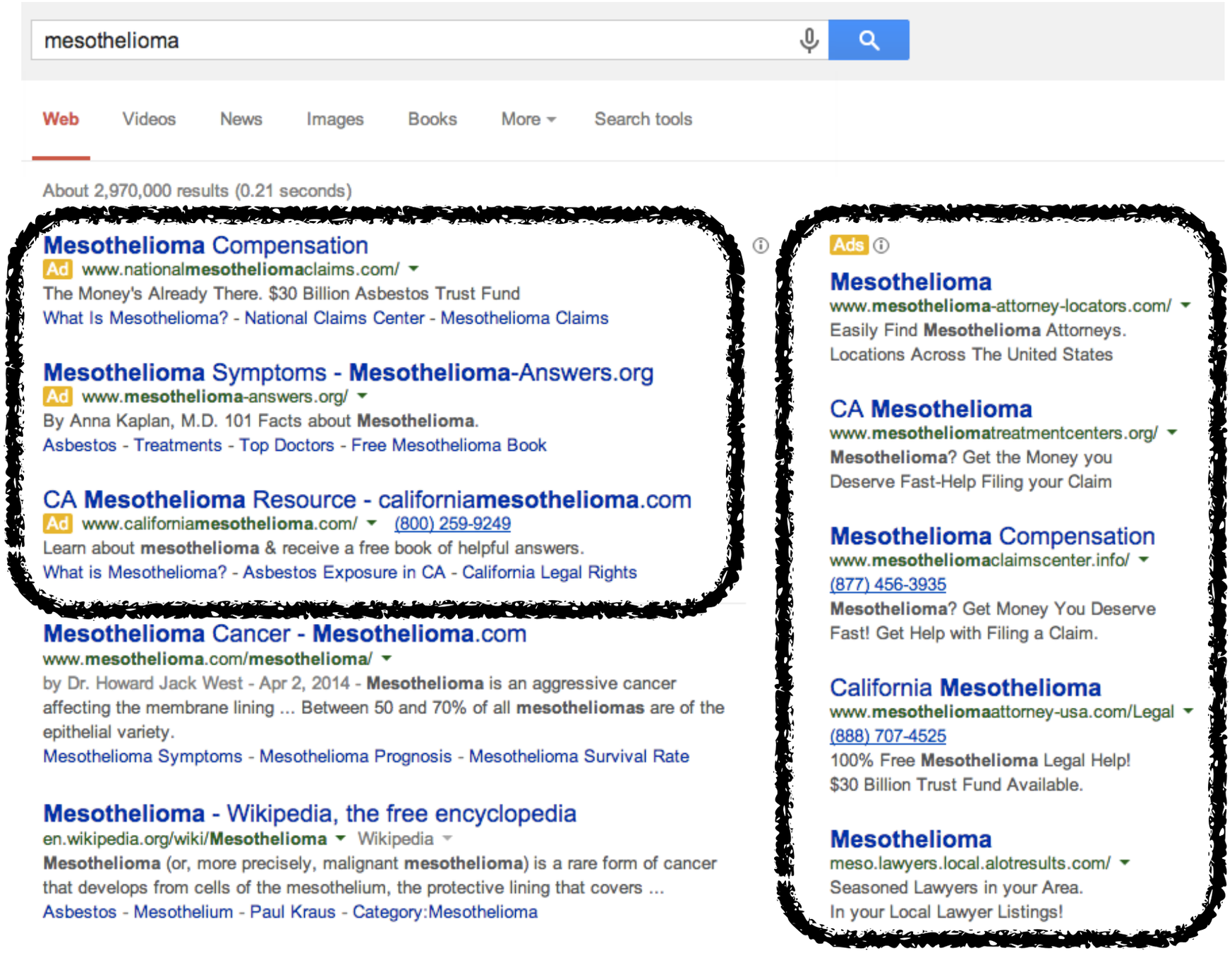
Data per minute 2014



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

Google

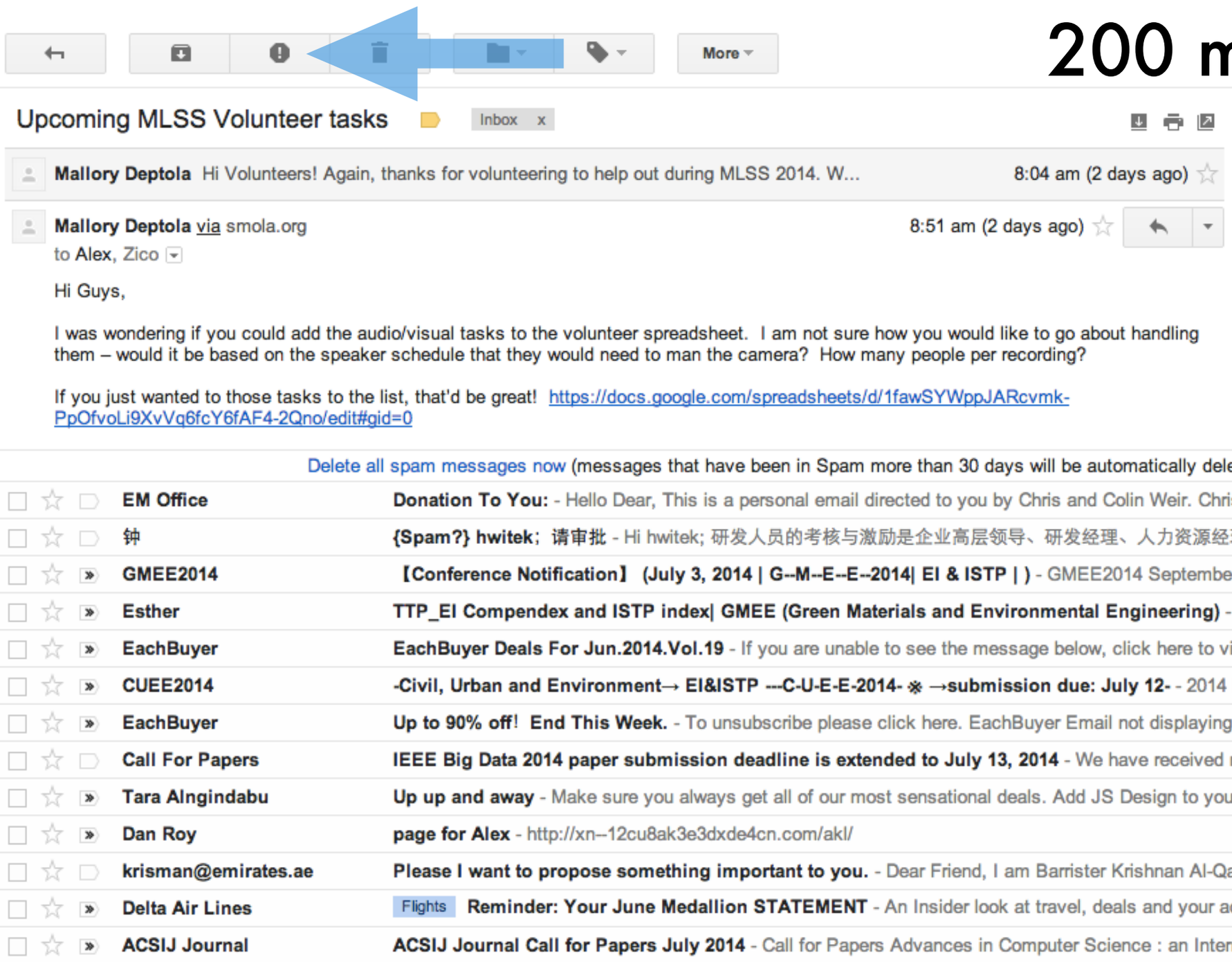
Carnegie Mellon University

Spam filtering

200 million/minute

imbalanced

dataset

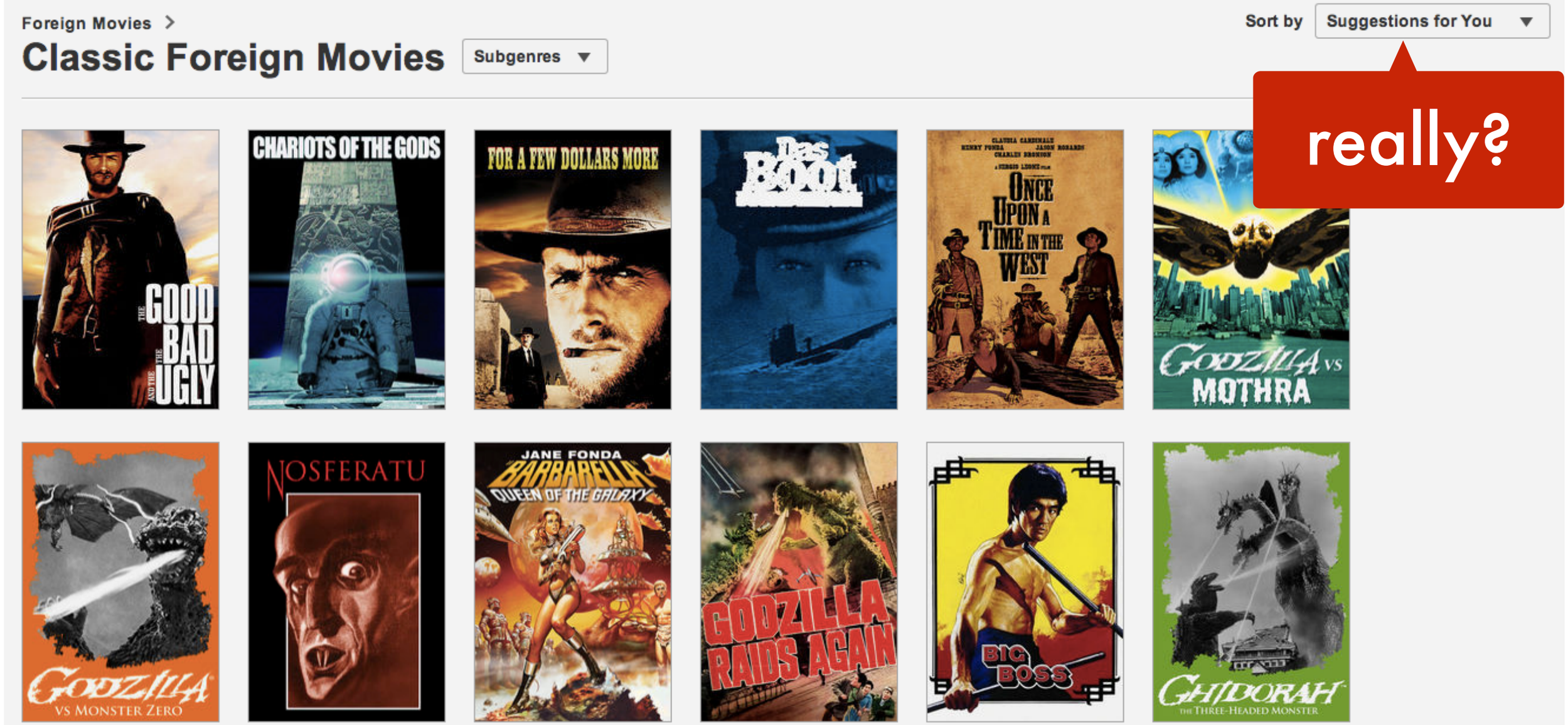


Google

Carnegie Mellon University

Recommendation & Ranking

maximize interaction probability for whole page



Google

Carnegie Mellon University

Time series & trends

Topics

Subscribe

machine learning
Search term

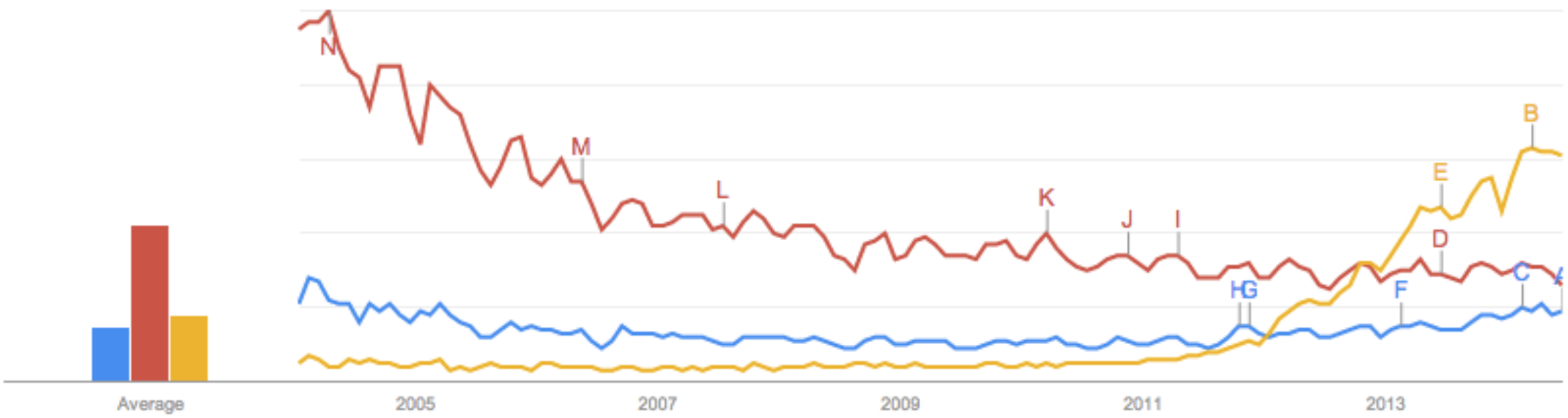
data mining
Search term

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

User generated content

- Webpages (content, graph)
- Clicks (ad, page, social)
- Users (OpenID, FB Connect)
- e-mails (Hotmail, Y!Mail, Gmail)
- Photos, Movies (Flickr, YouTube, Vimeo ...)
- Cookies / tracking info (see Ghostery)
- Installed apps (Android market etc.)
- Location (Latitude, Loopt, Foursquared, Google Now)
- User generated content (Wikipedia & co)
- Ads (display, text, DoubleClick, Yahoo)
- Comments (Disqus, Facebook)
- Reviews (Yelp, Y!Local)
- Third party features (e.g. Experian)
- Social connections (LinkedIn, Facebook)
- Purchase decisions (Netflix, Amazon)
- Instant Messages (YIM, Skype, Gtalk)
- Search terms (Google, Bing)
- Timestamp (everything)
- News articles (BBC, NYTimes, Y!News)
- Blog posts (Tumblr, Wordpress)
- Microblogs (Twitter, Jaiku, Meme)
- Link sharing (Facebook, Delicious, Buzz)
- Network traffic

flickr™

YouTube

yelp®

DISQUS

Google

Carnegie Mellon University

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

Architecture

Real Hardware

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

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

Jeff Dean's Stanford slides

Numbers everyone should know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Google

Jeff Dean's Stanford slides

Carnegie Mellon University

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

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

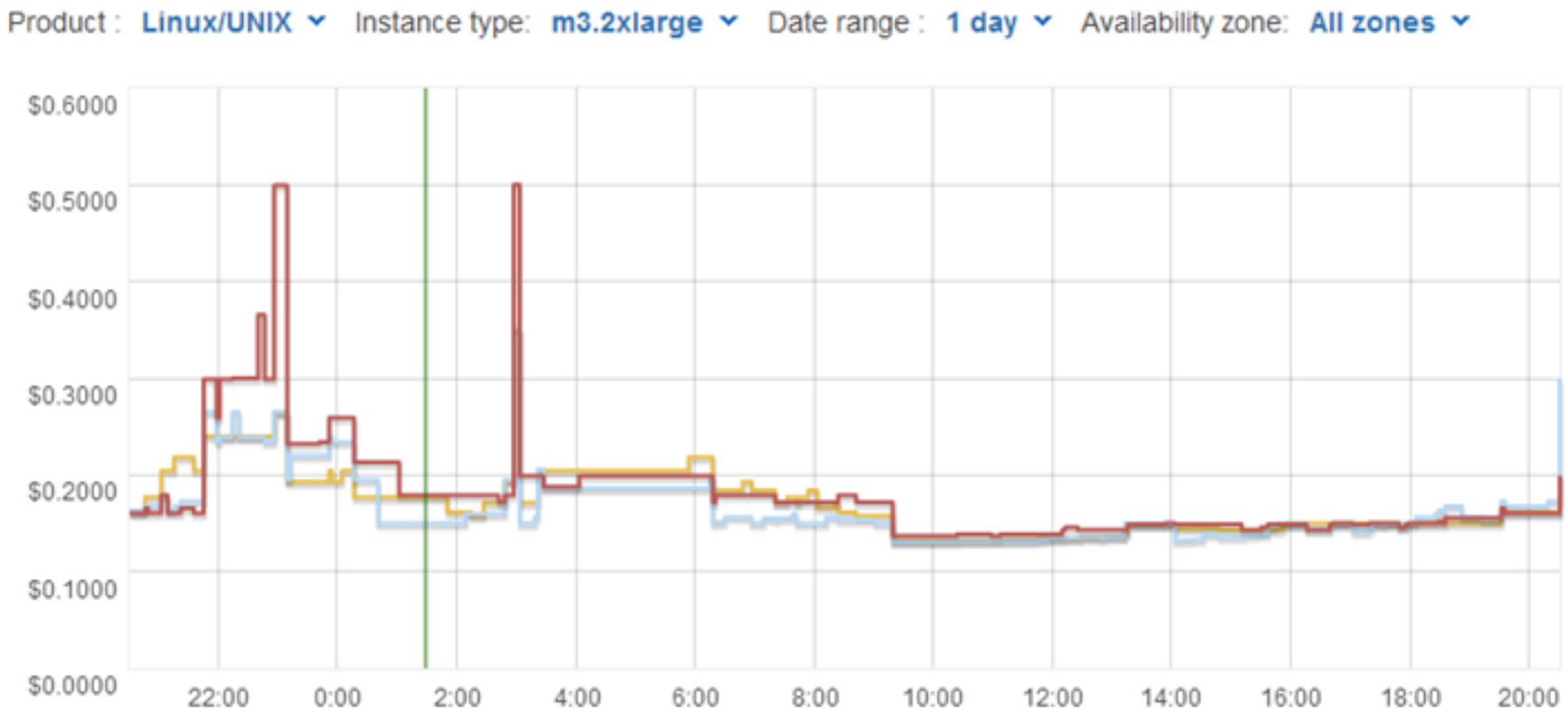
Amazon EBS Magnetic volumes
- $0.05 per GB-month of provisioned storage
- $0.05 per 1 million I/O requests

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

Spot instances much cheaper

Real Hardware

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



Work & storage

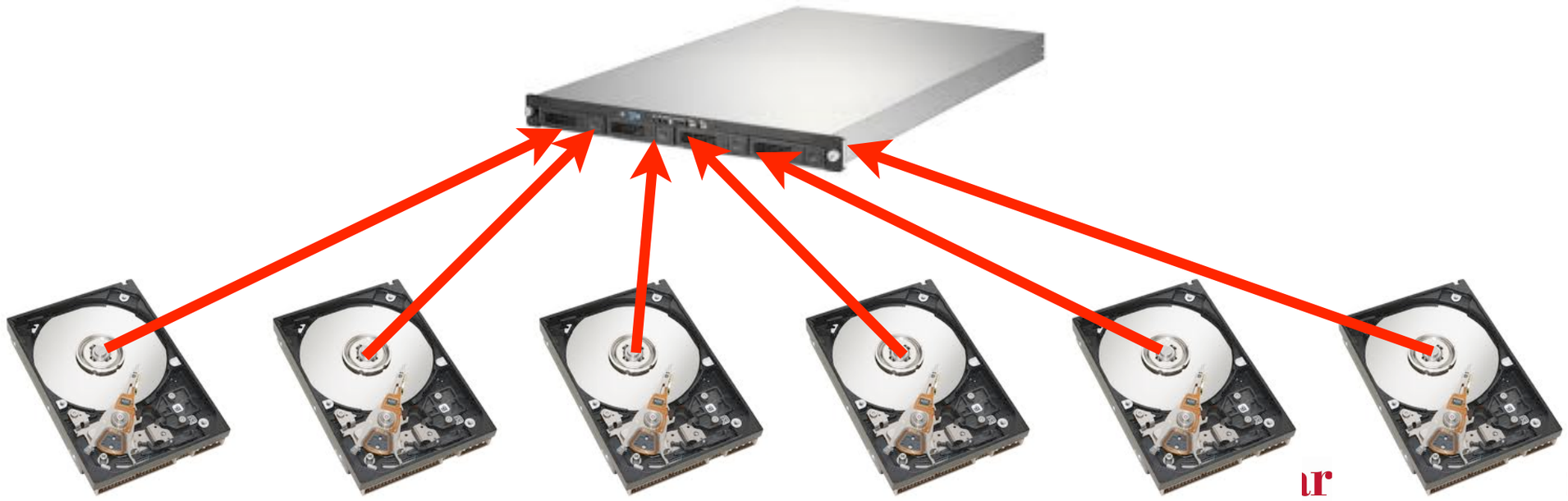
File systems

Google

Carnegie Mellon University

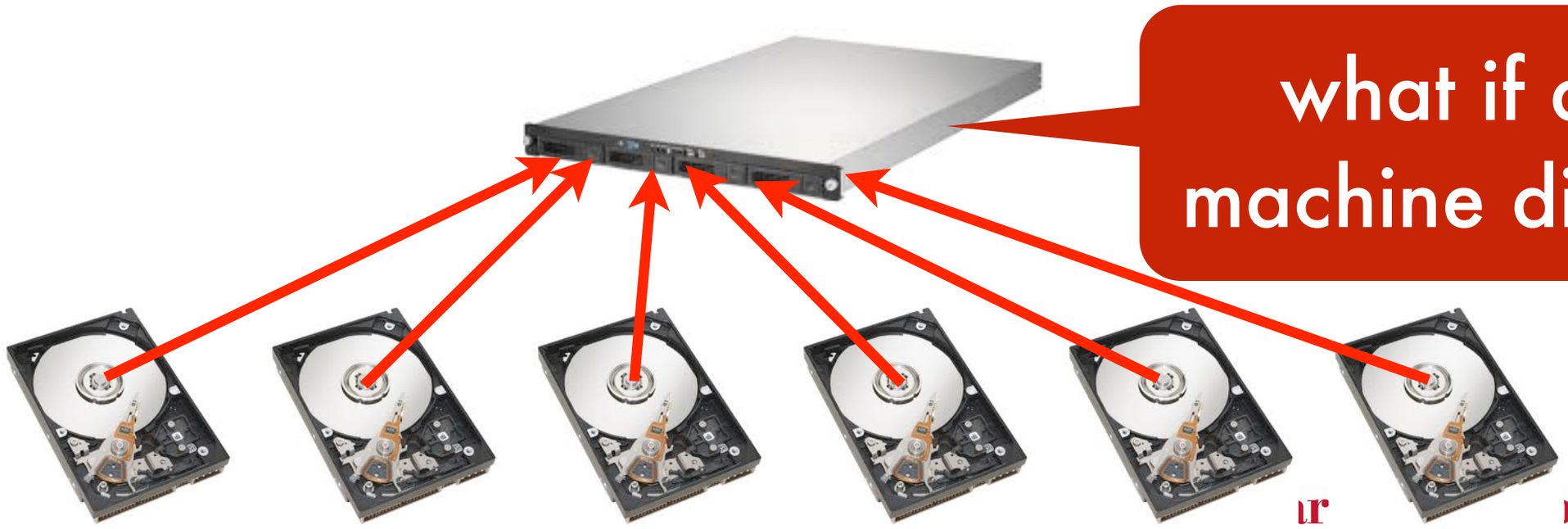
RAID

- 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

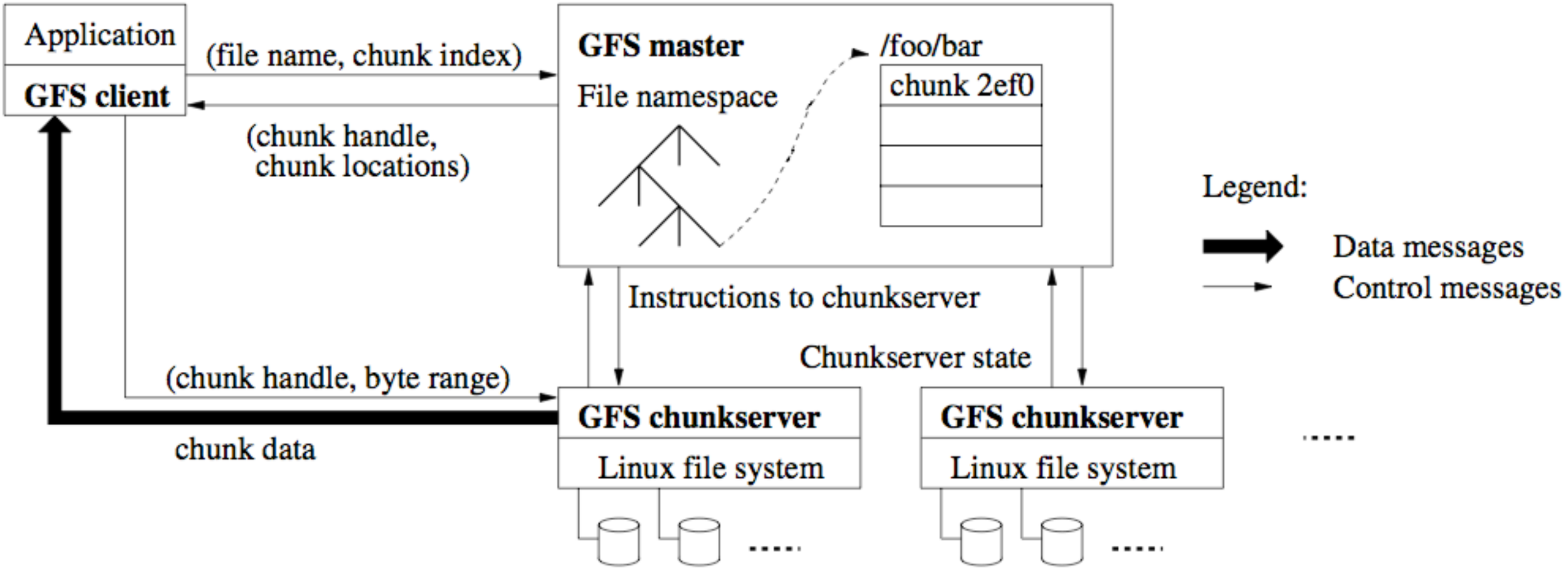
- Redundant array of inexpensive disks (optional fault tolerance)
  - Aggregate storage of many disks
  - Aggregate bandwidth of many disks
- RAID 0 - stripe data over disks (good bandwidth, faulty)
- RAID 1 - mirror disks (mediocre bandwidth, fault tolerance)
- RAID 5 - stripe data with 1 disk for parity (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

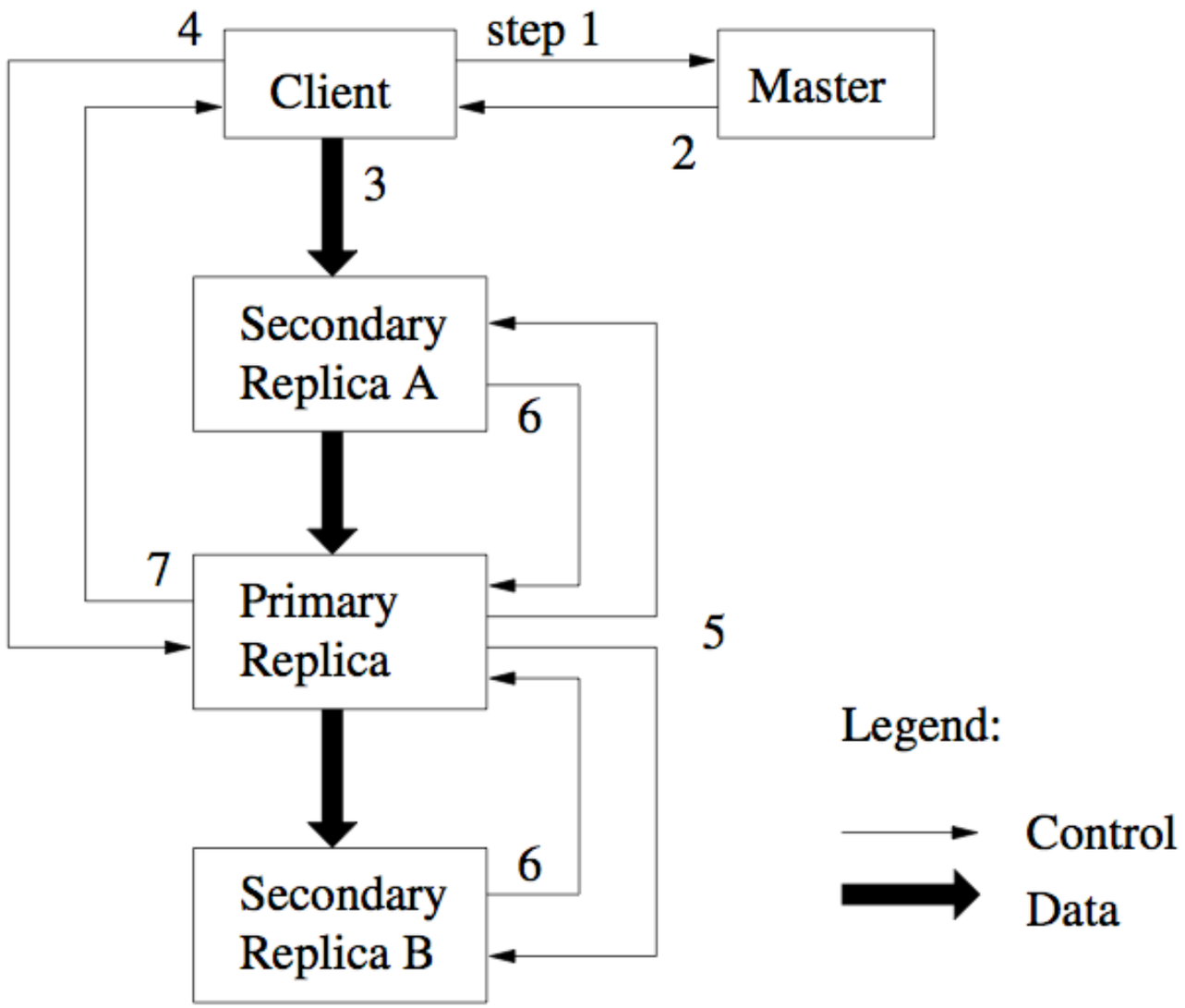


Ghemawat, Gobioff, Leung, 2003

- 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

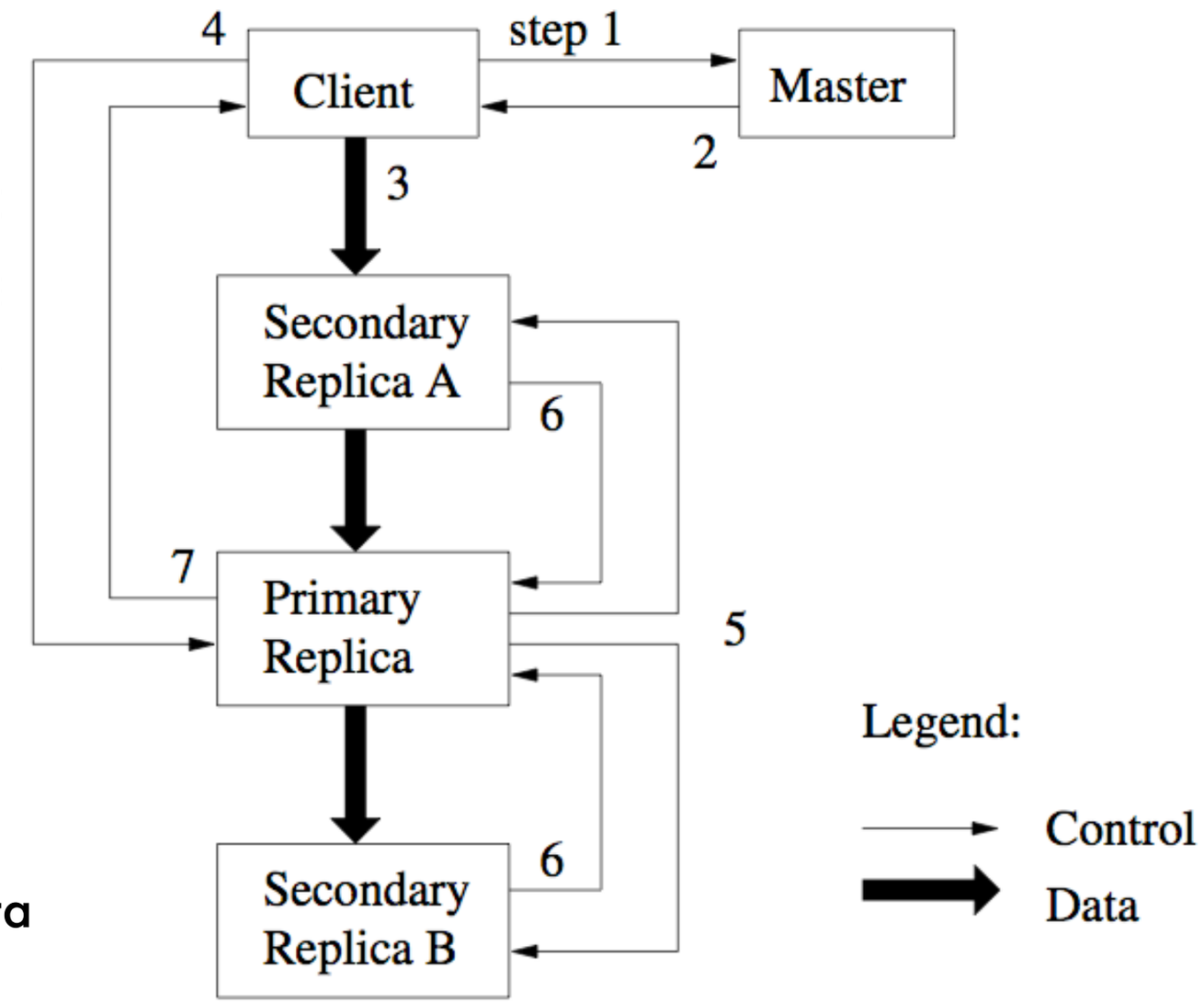
Google File System / HDFS

- 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

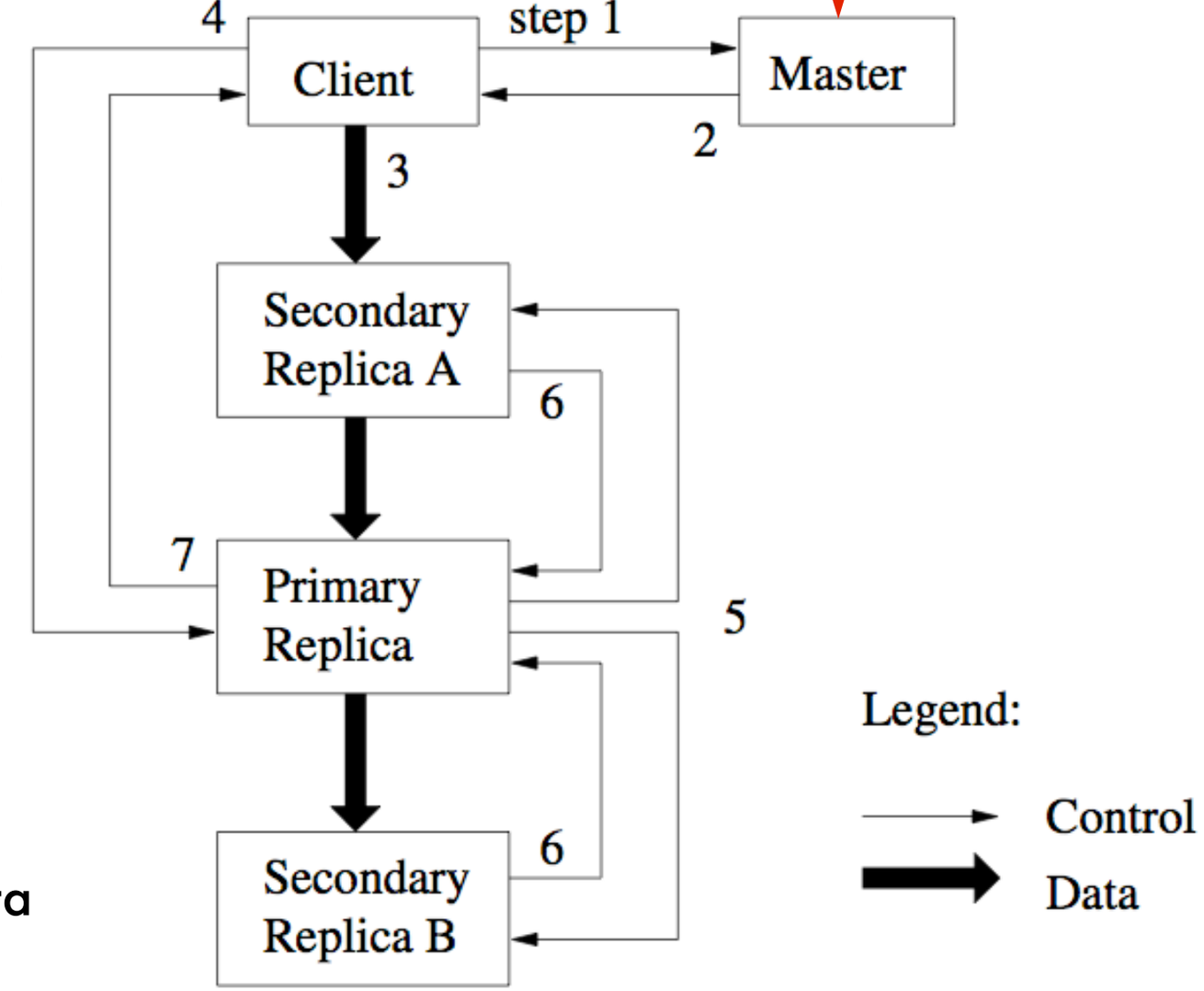


- Master ensures nodes are live
- Chunks are checksummed
- Can control replication factor for hotspots / load balancing
- Deserialize master state by loading data structure as flat file from disk (fast)

Google 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

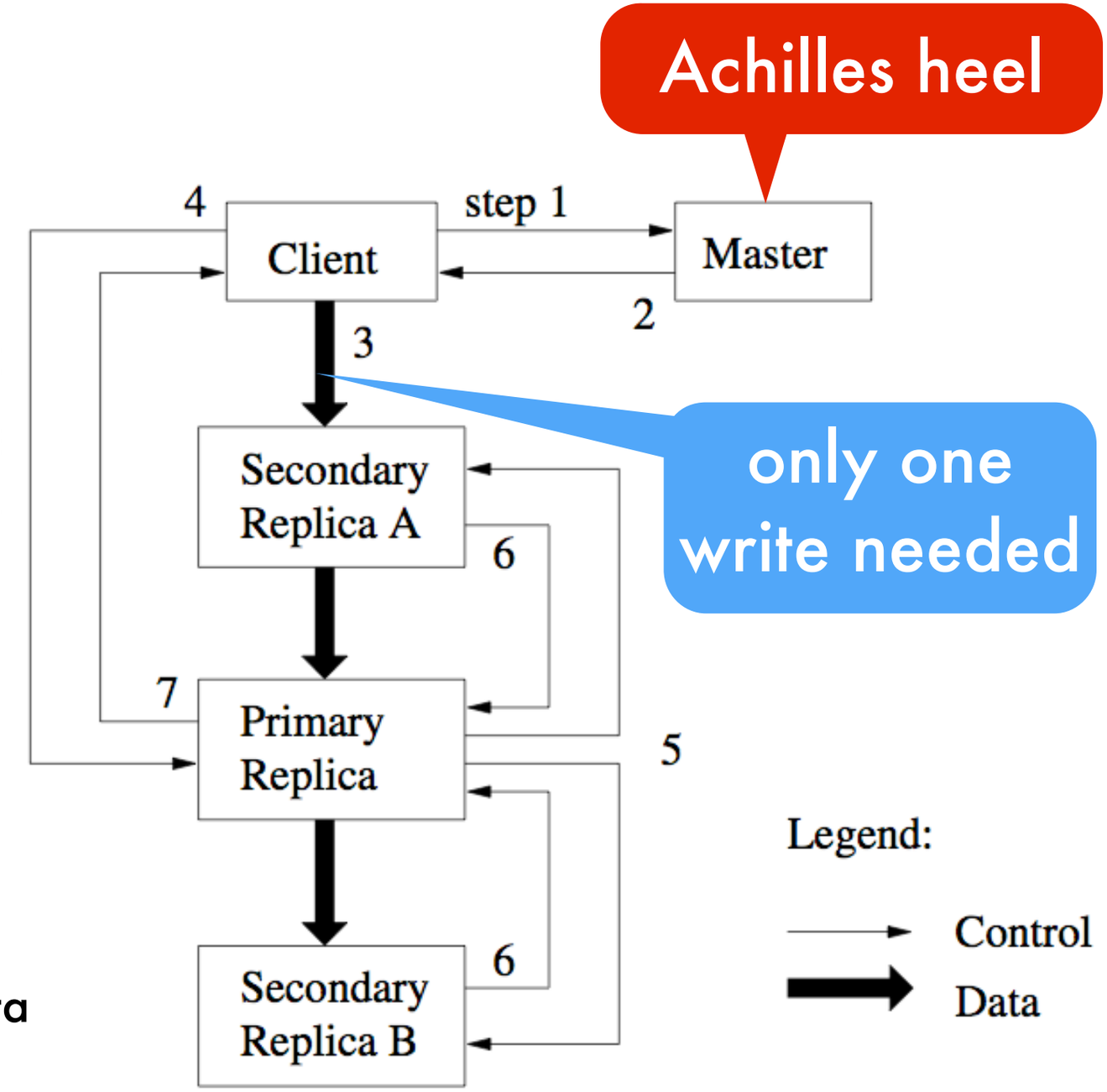
Achilles heel



- 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

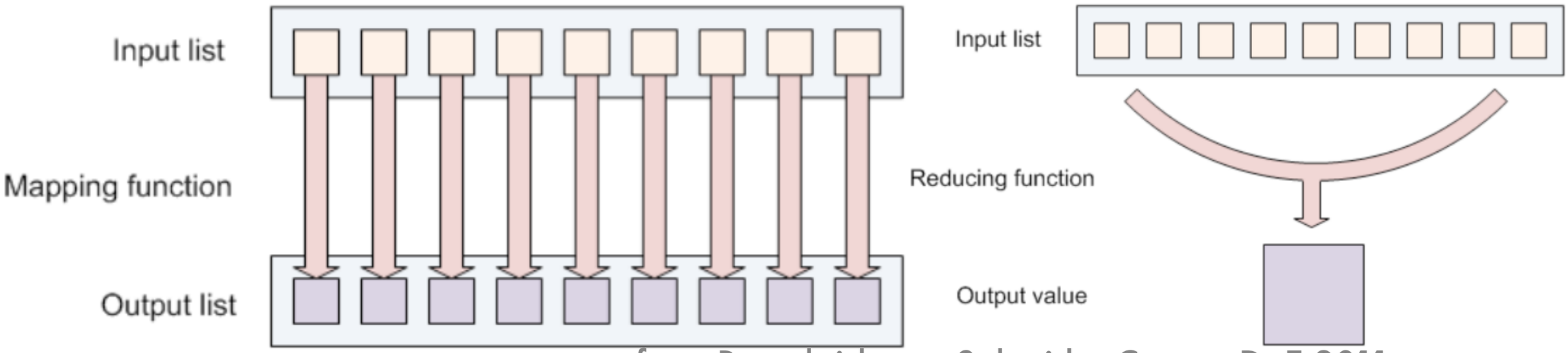


- Master ensures nodes are live
- Chunks are checksummed
- Can control replication factor for hotspots / load balancing
- Deserialize master state by loading data structure as flat file from disk (fast)

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

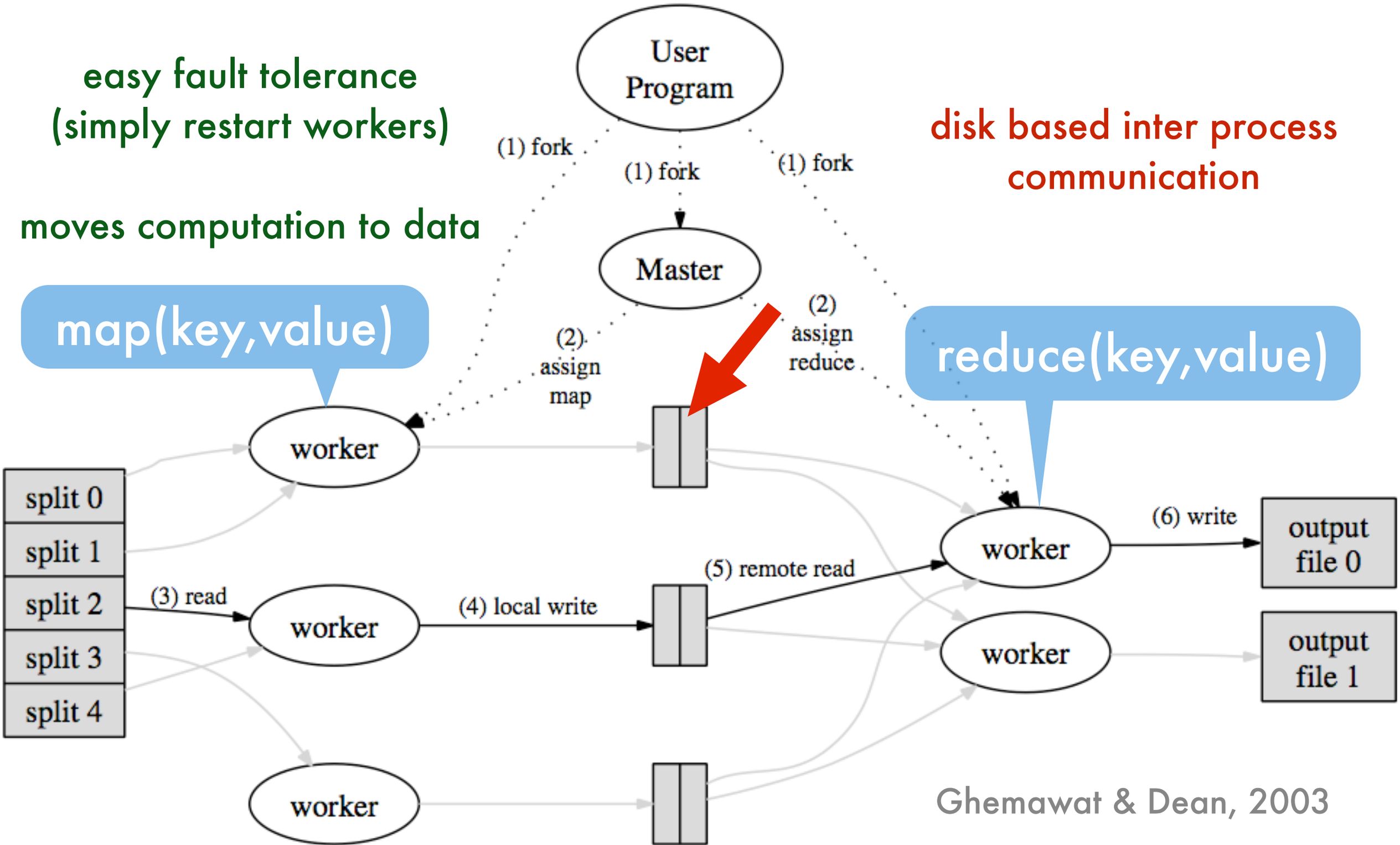
- 1000s of (faulty) machines
- Lots of jobs are embarrassingly parallel (except for a sorting/transpose phase)
- Functional programming origins
  - Map(key,value) processes (key,value) pairs and outputs new (key,value) pair
  - Reduce(key,value) reduces all instances with same key to aggregate
- Example - (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**

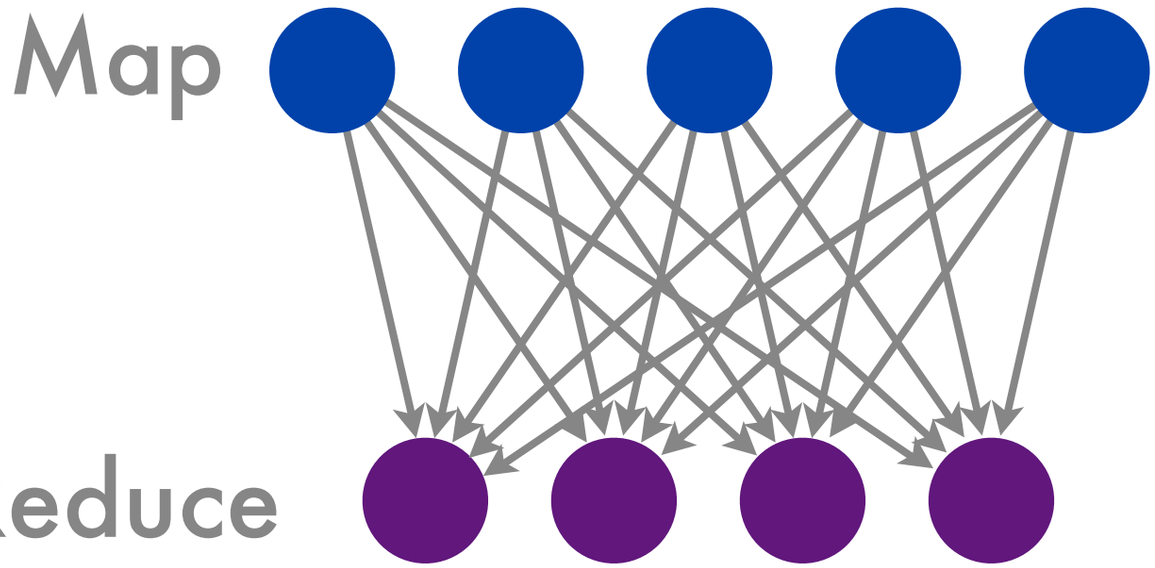
$$g := \sum_{i=1}^{m} \partial_w l(x_i, y_i, w)$$

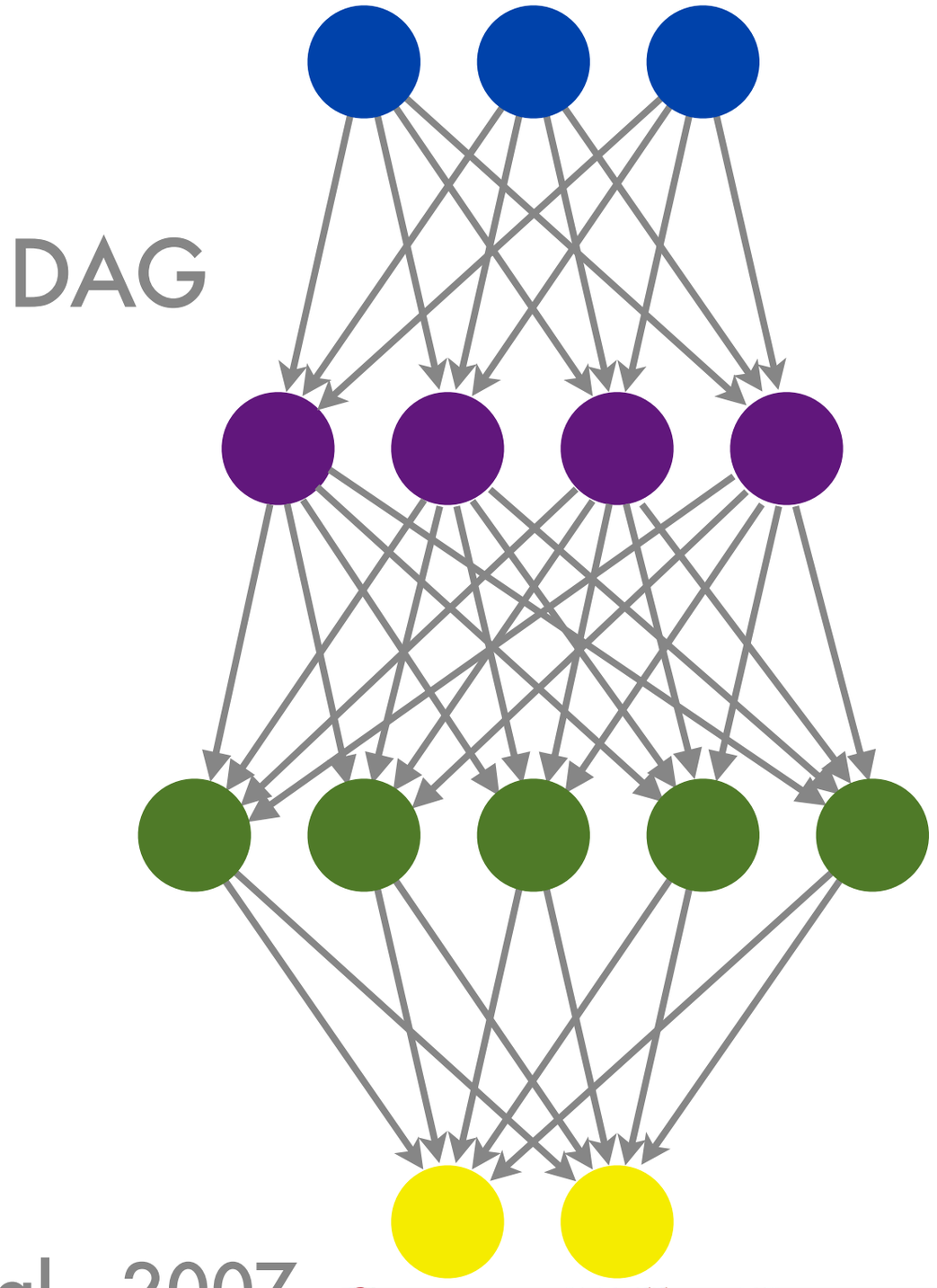
    - **On each data point via Map(i,data)**
    - **Sum gradient via Reduce(coordinate)**
  - **perform update step (much better with line search)**

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

  - **repeat**

Dryad

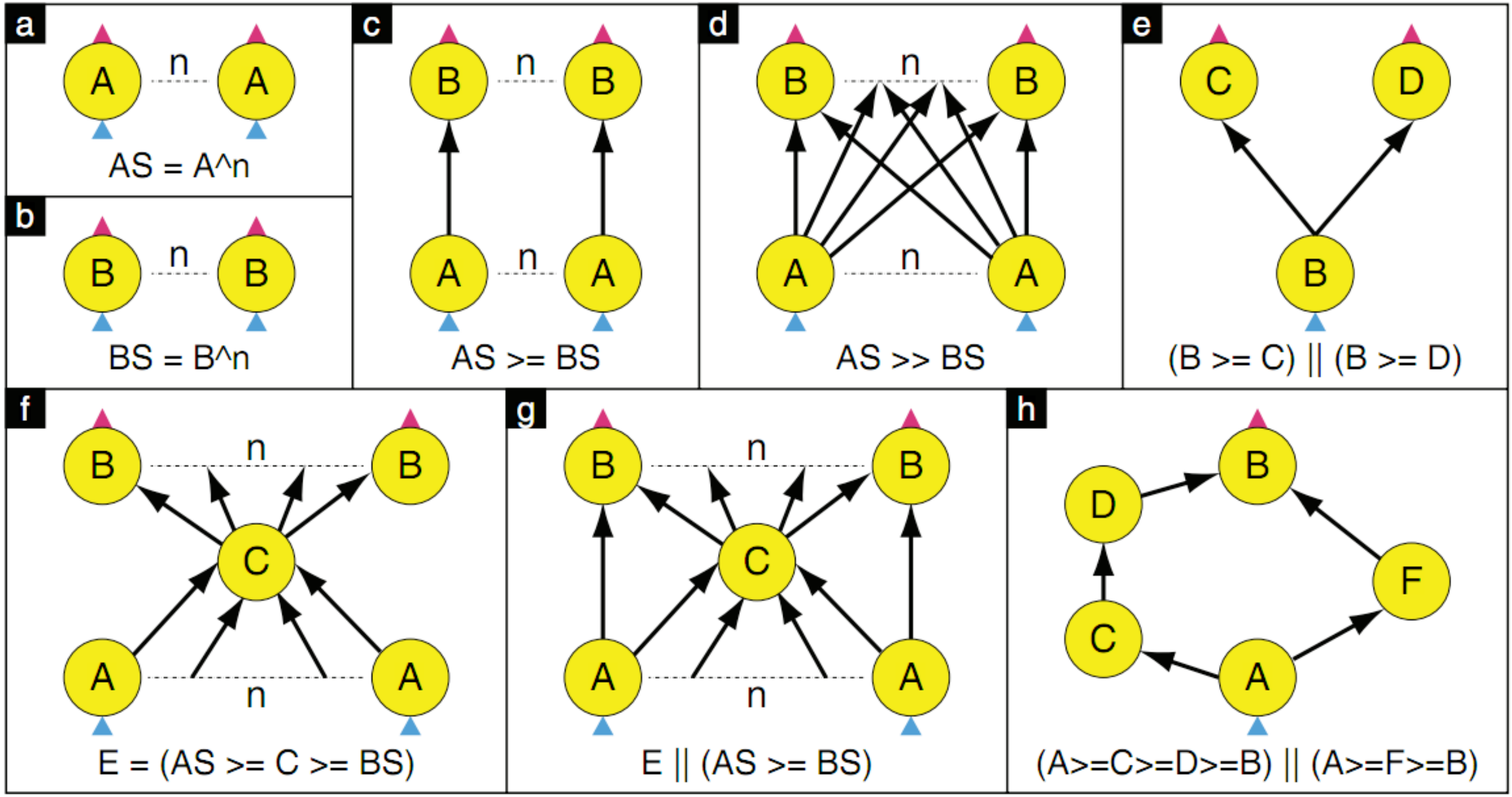




- Directed acyclic graph
- System optimizes parallelism
- Different types of IPC (memory FIFO/network/file)
- Tight integration with .NET (allows easy prototyping)

Isard et al., 2007

DRYAD

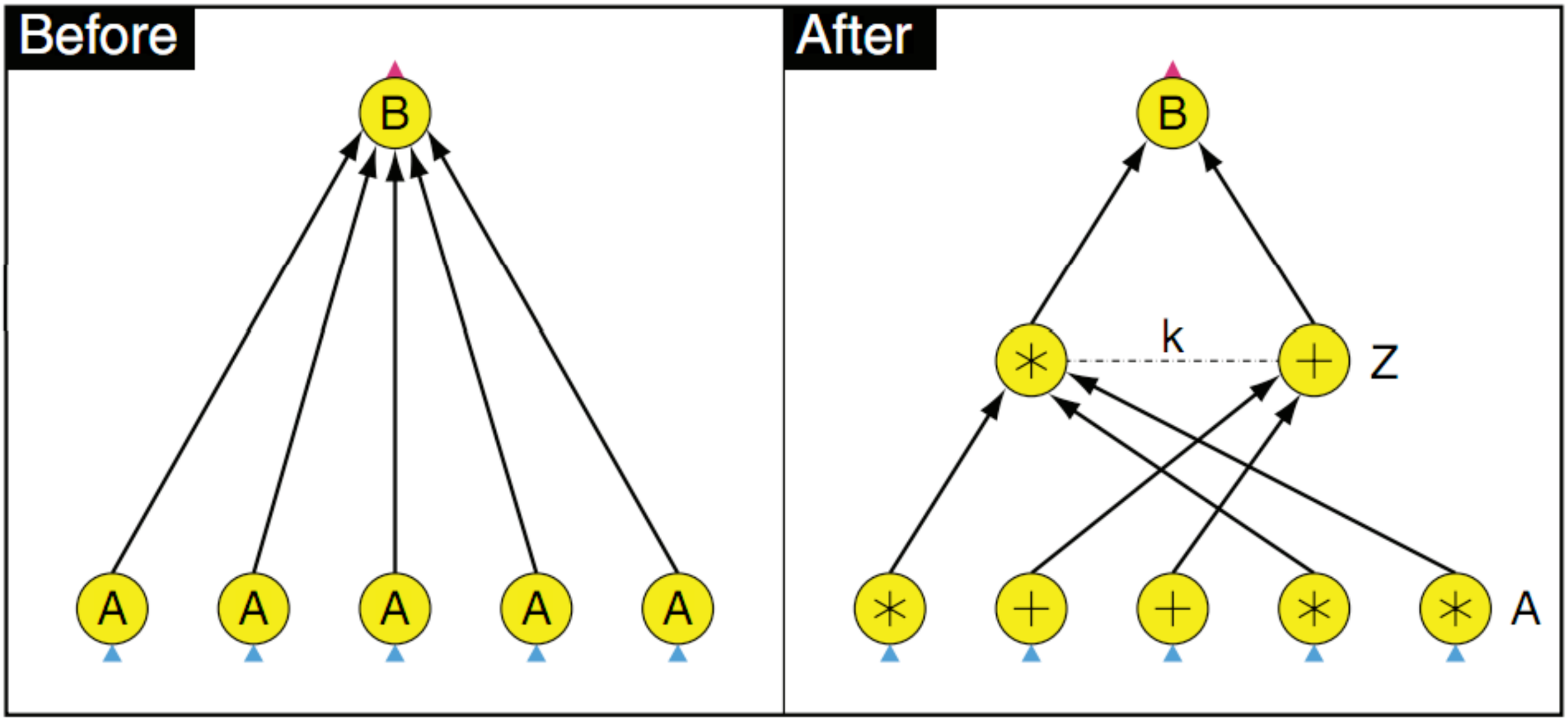


graph description language

Google

Carnegie Mellon University

DRYAD



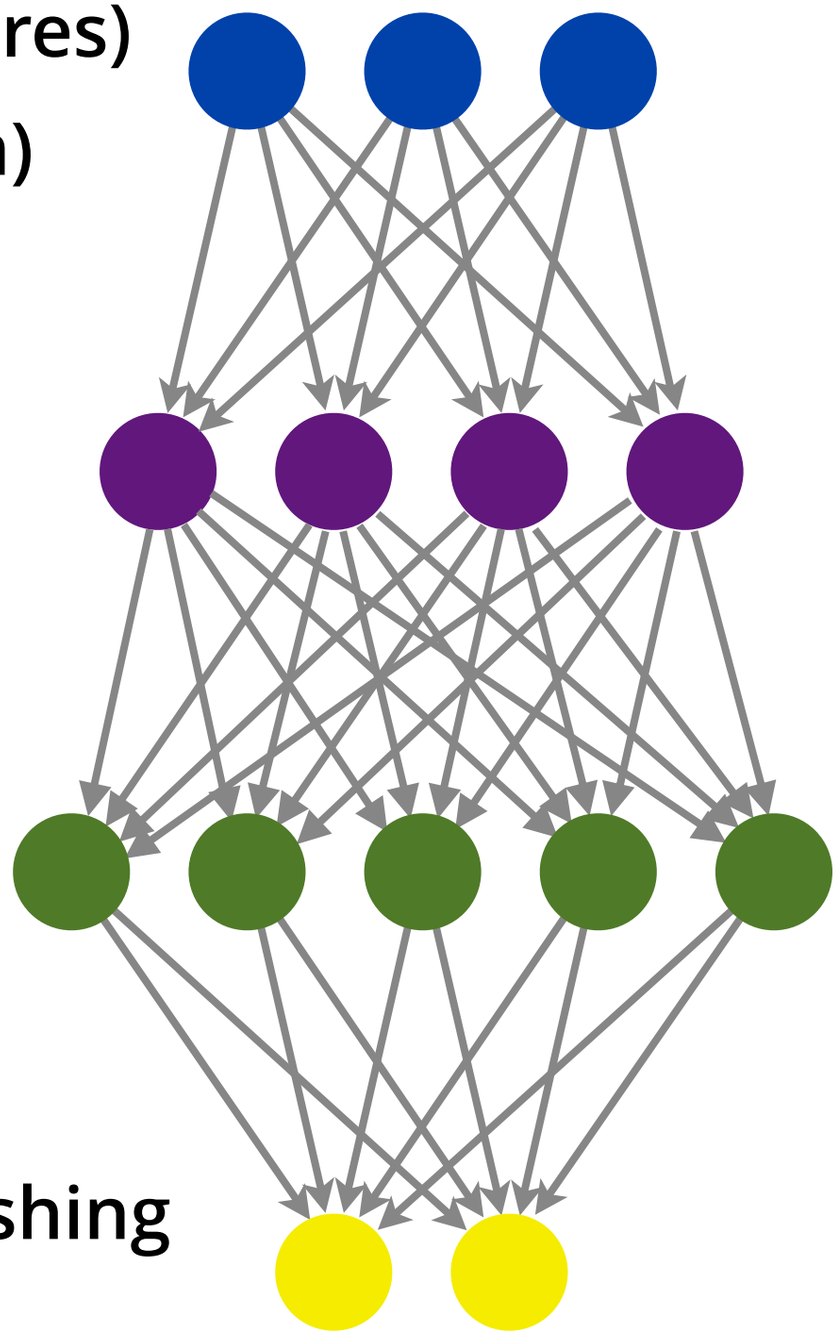
automatic graph refinement

Google

Carnegie Mellon University

S4

- Directed acyclic graph (Dryad-like features)
- Real-time processing of data (as stream)
- Scalability (decentralized & symmetric)
- Fault tolerance
- Consistency for keys



- Processing elements
  - Ingest (key, value) pair
  - Capabilities tied to ID
  - Clonable (for scaling)
- Simple implementation - consistent hashing

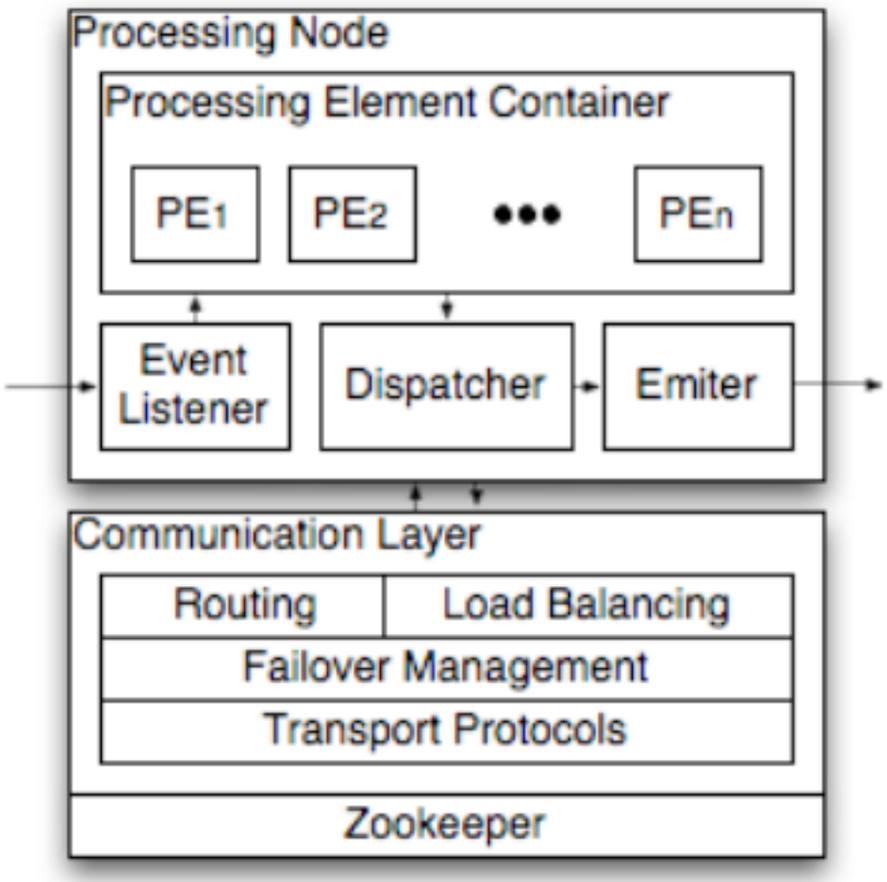
http://s4.io Neumeyer et al, 2010

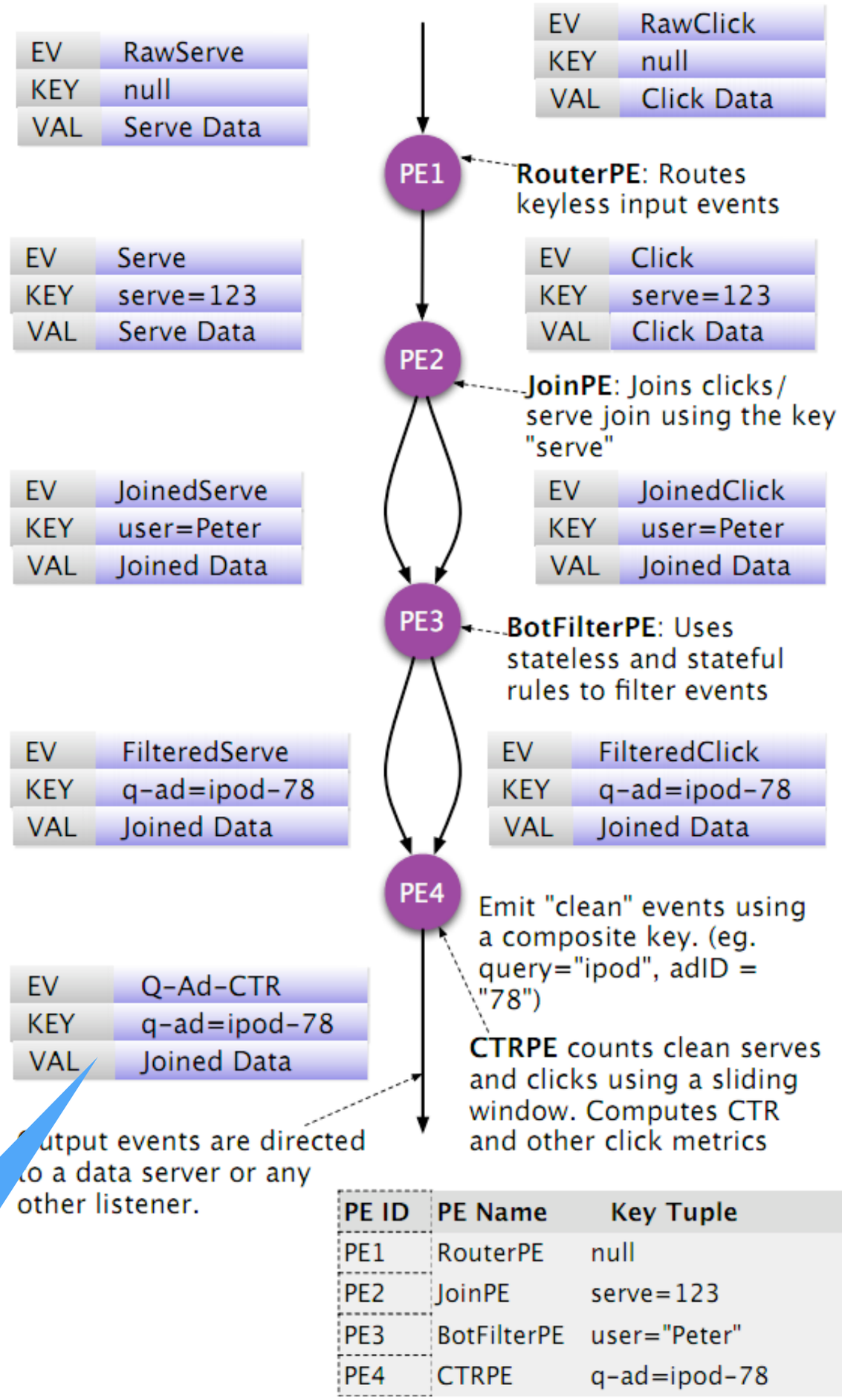
Google

Carnegie Mellon University

S4

processing element





PE ID	PE Name	Key Tuple
PE1	RouterPE	null
PE2	JoinPE	serve=123
PE3	BotFilterPE	user="Peter"
PE4	CTRPE	q-ad=ipod-78

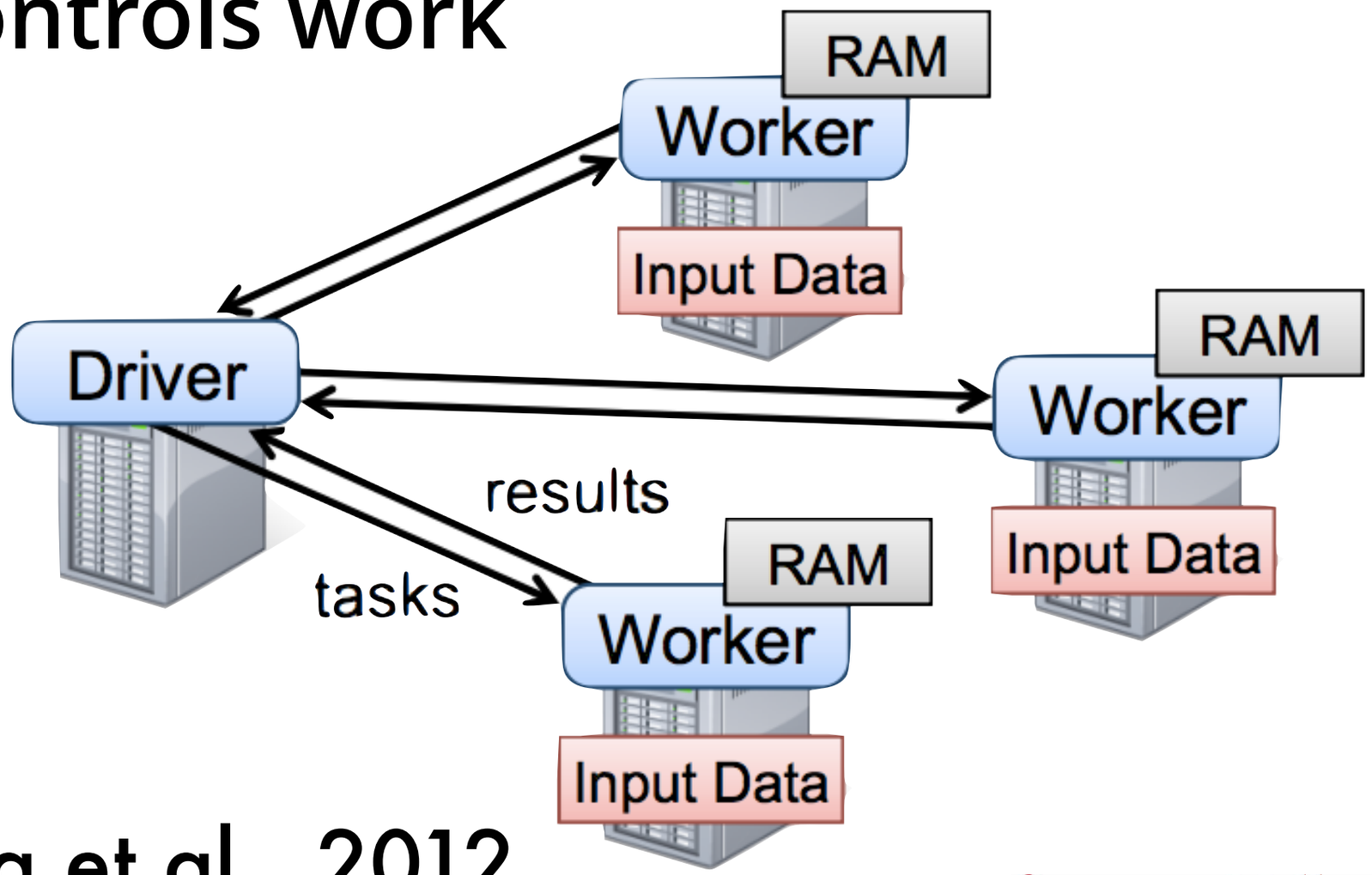
click through rate estimation

Google

Spark

Resilient Distributed Datasets

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



Zaharia et al., 2012

Beyond MapReduce

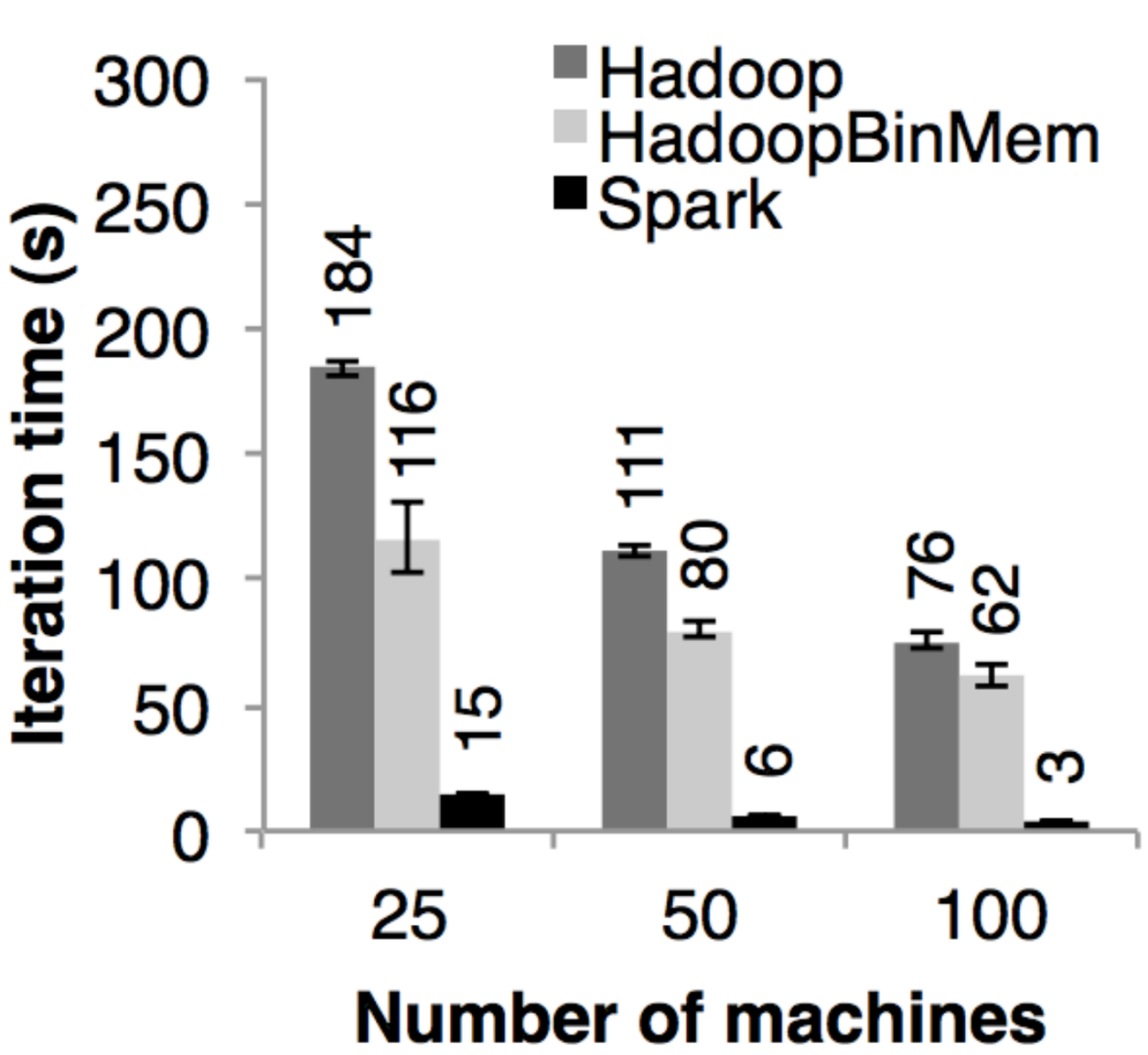
Transformations	*map*(*f* : T ⇒ U)	:	RDD[T] ⇒ RDD[U]
	filter(*f* : T ⇒ Bool)	:	RDD[T] ⇒ RDD[T]
	flatMap(*f* : T ⇒ Seq[U])	:	RDD[T] ⇒ RDD[U]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	groupByKey()	:	RDD[(K, V)] ⇒ RDD[(K, Seq[V])]
	reduceByKey(*f* : (V, V) ⇒ V)	:	RDD[(K, V)]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

rich language & preprocessor

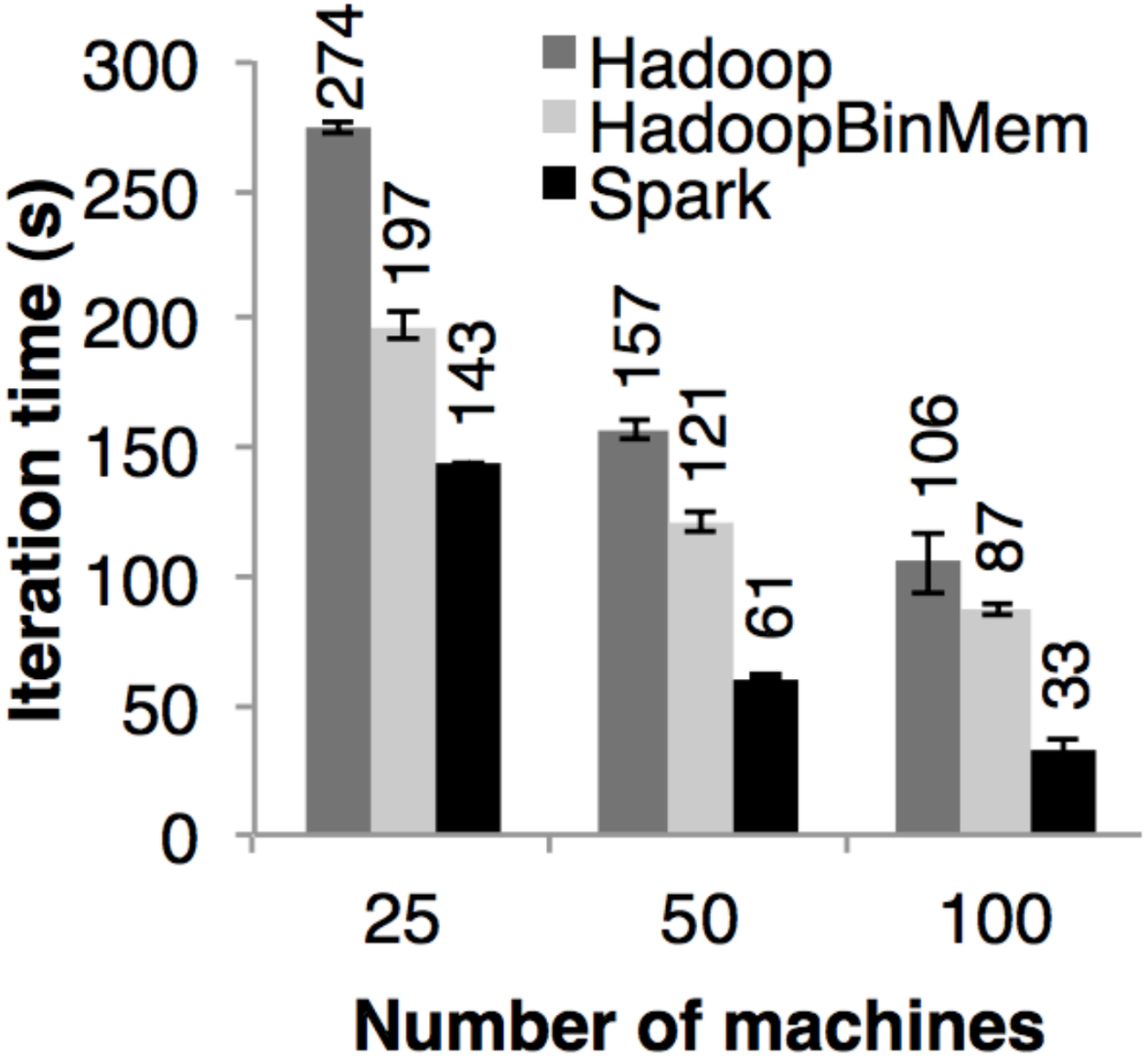
Google

Carnegie Mellon University

Improvement over MapReduce



(a) Logistic Regression

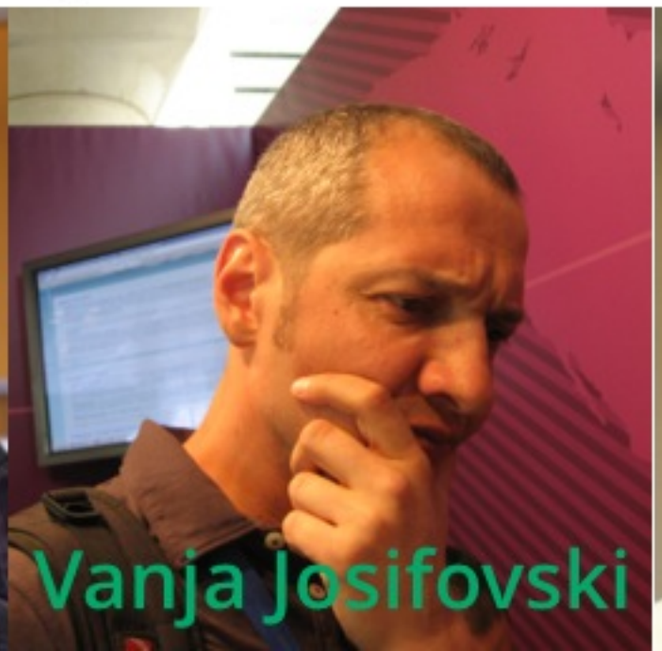


(b) K-Means

Google

Carnegie Mellon University

parameterserver.org



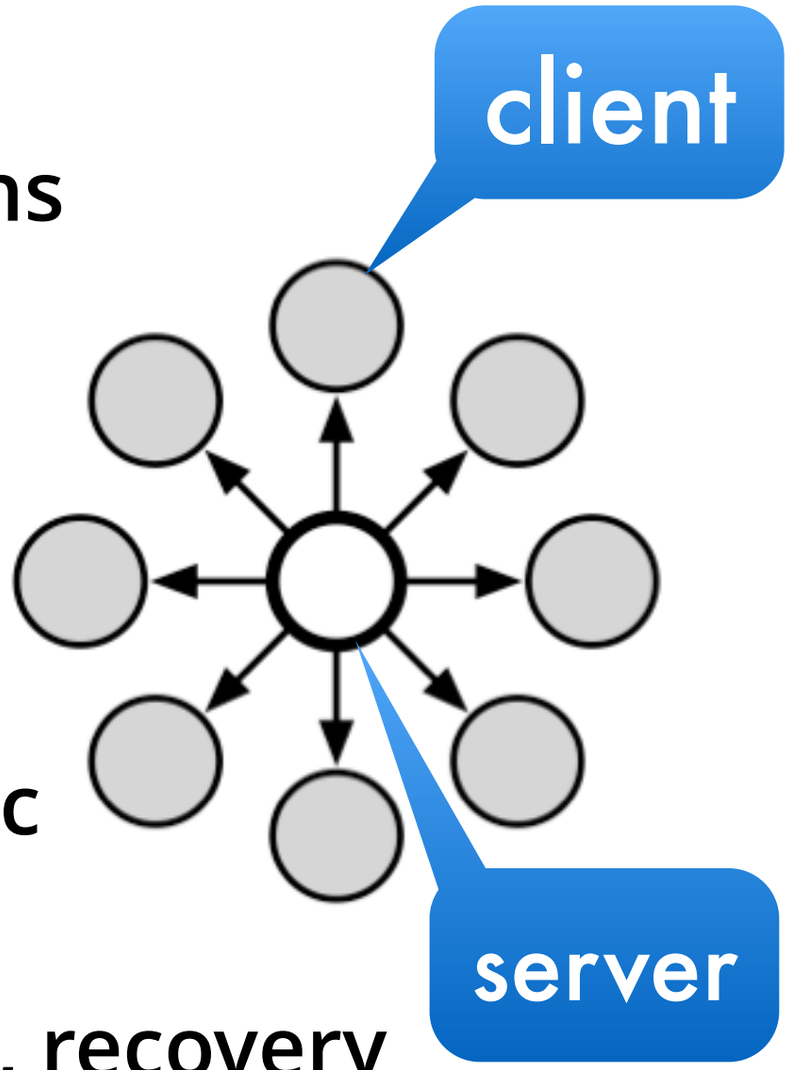
Background

The Challenge

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

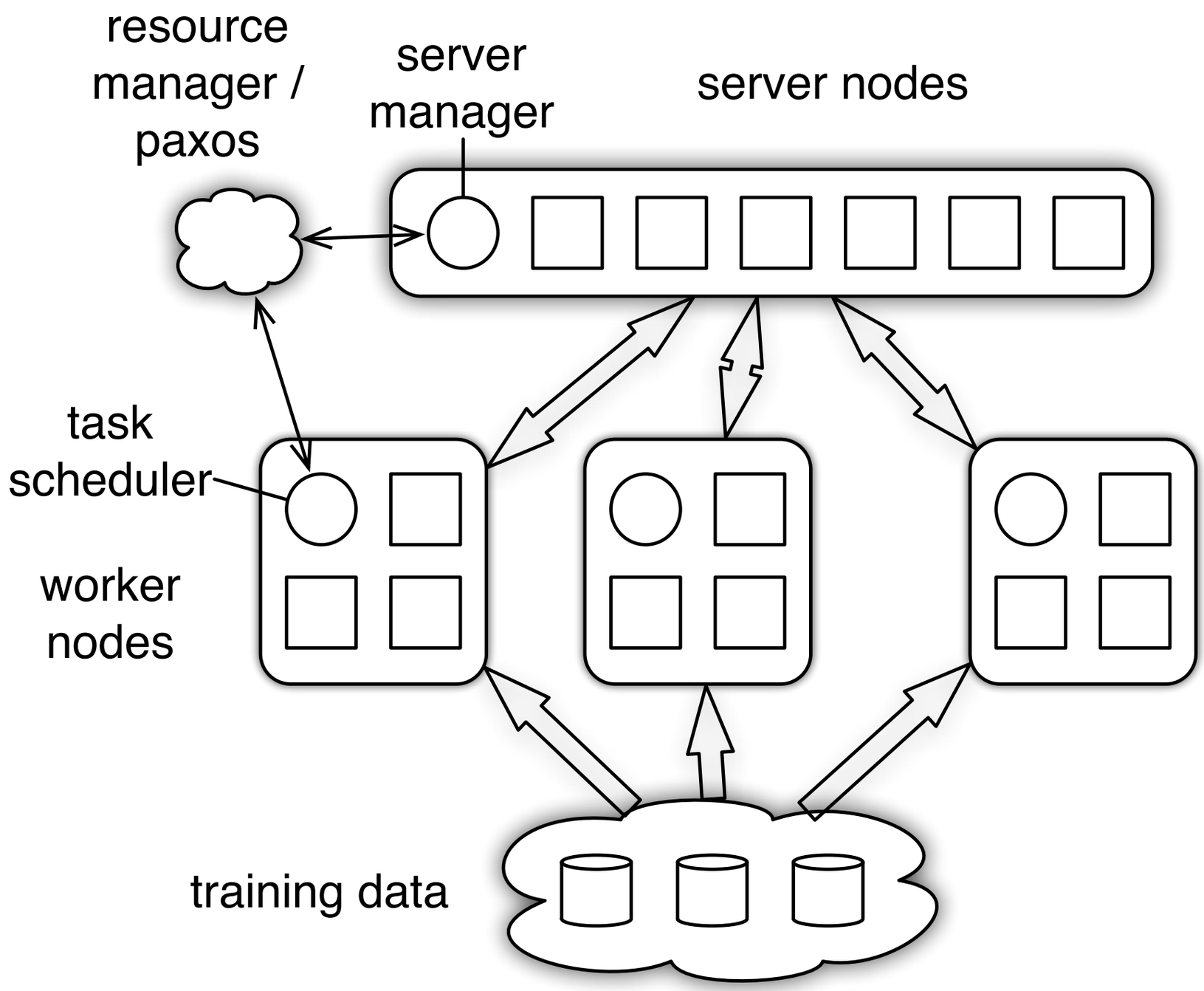
General parallel algorithm template

- 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

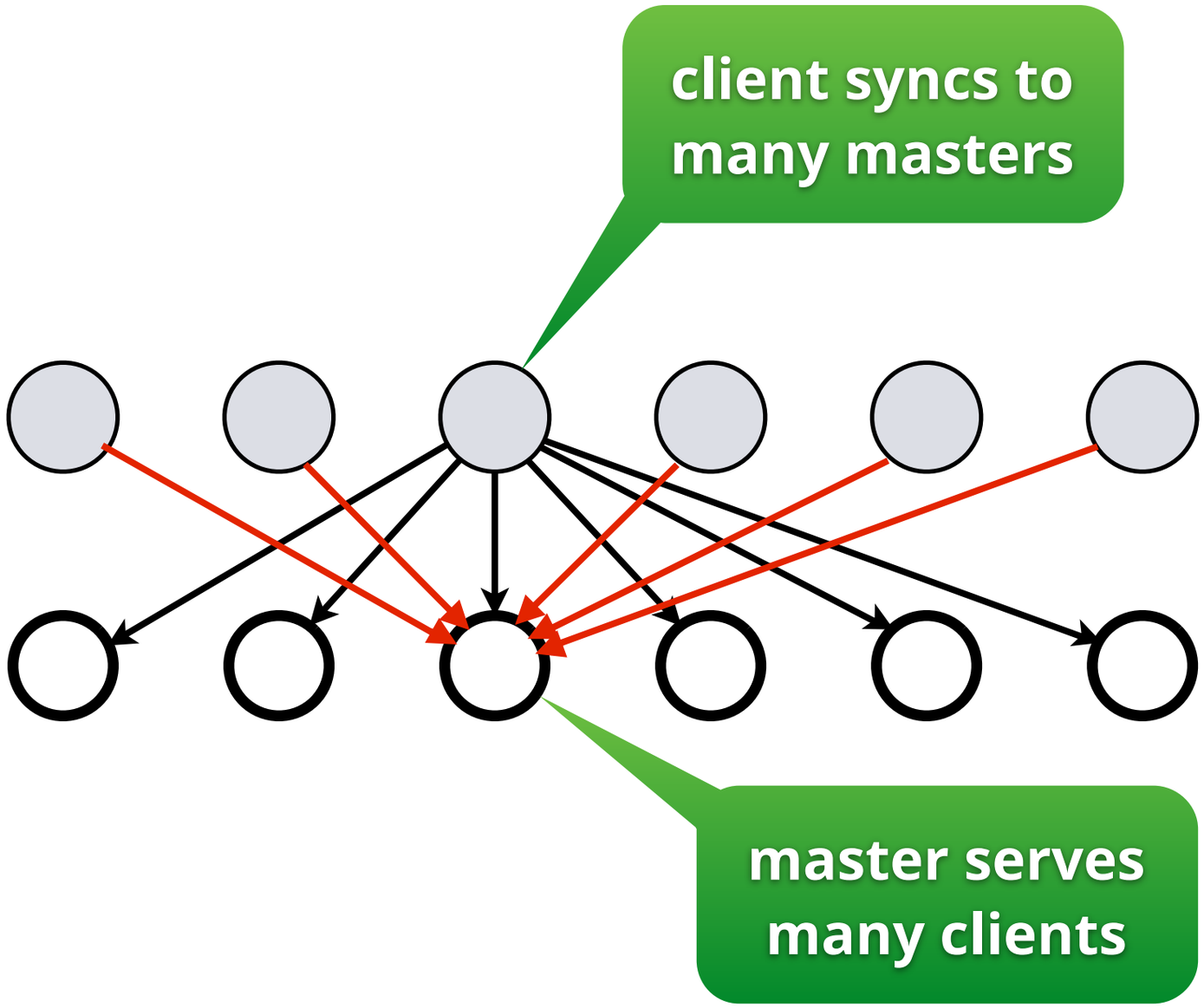


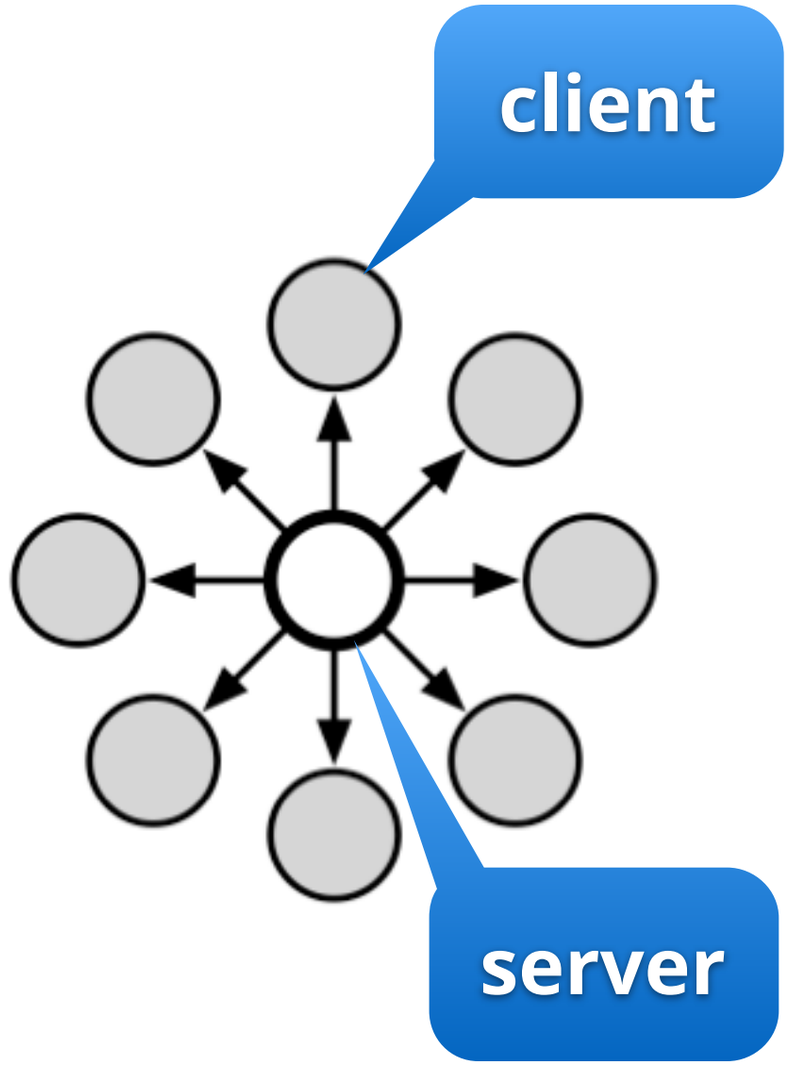
Smola & Narayanamurthy, 2010, VLDB
Gonzalez et al., 2012, WSDM
Shervashidze et al., 2013, WWW

Architecture



Communication pattern

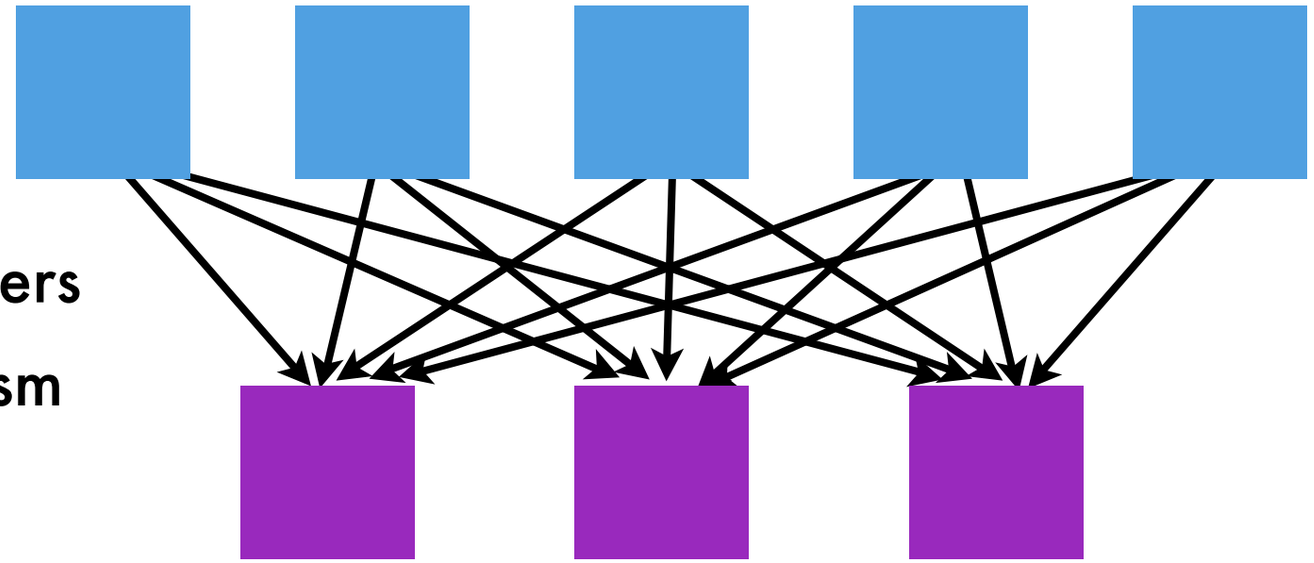




Key layout & recovery

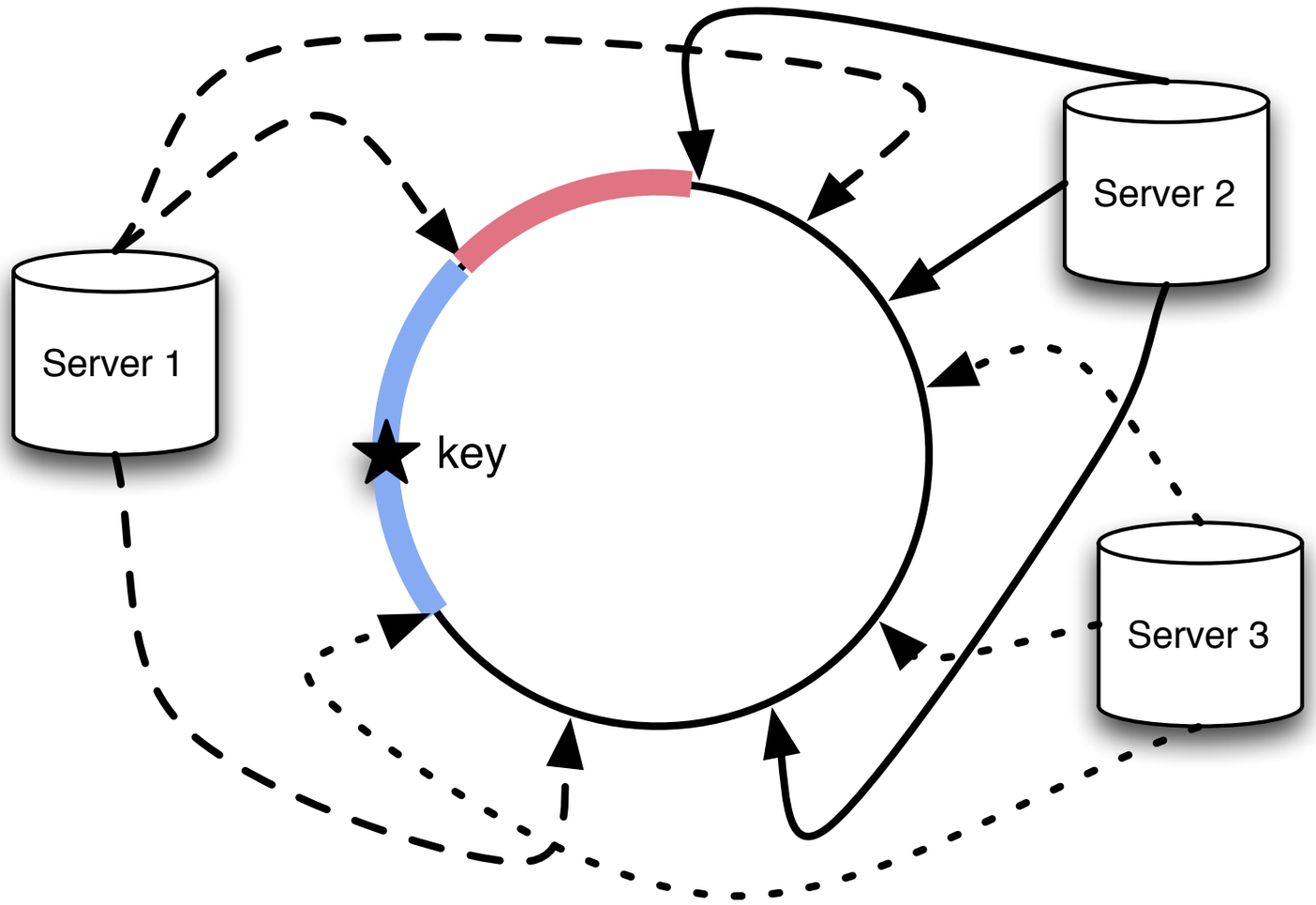
Consistent Hashing

- Caching problem
  - Store many (key,value) pairs
  - Linear scalability in clients and servers
  - Automatic key distribution mechanism
- 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
  - very expensive to iterate over all keys for a given server



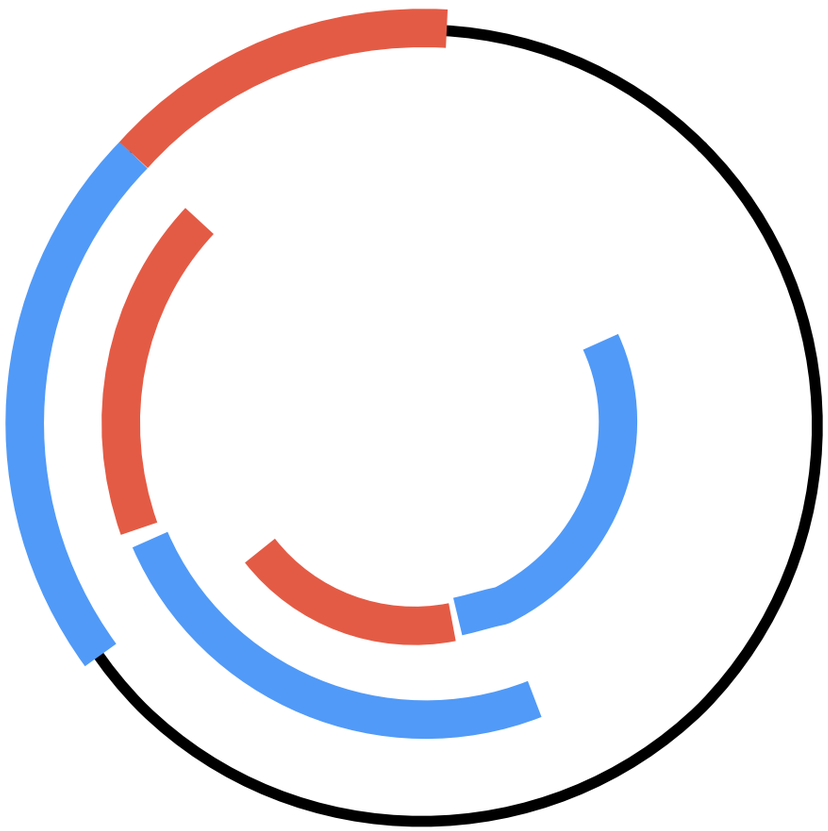
$$m(\text{key}, \mathcal{M}) = \underset{m' \in \mathcal{M}}{\operatorname{argmin}}\, h(\text{key}, m')$$

Keys arranged in a DHT



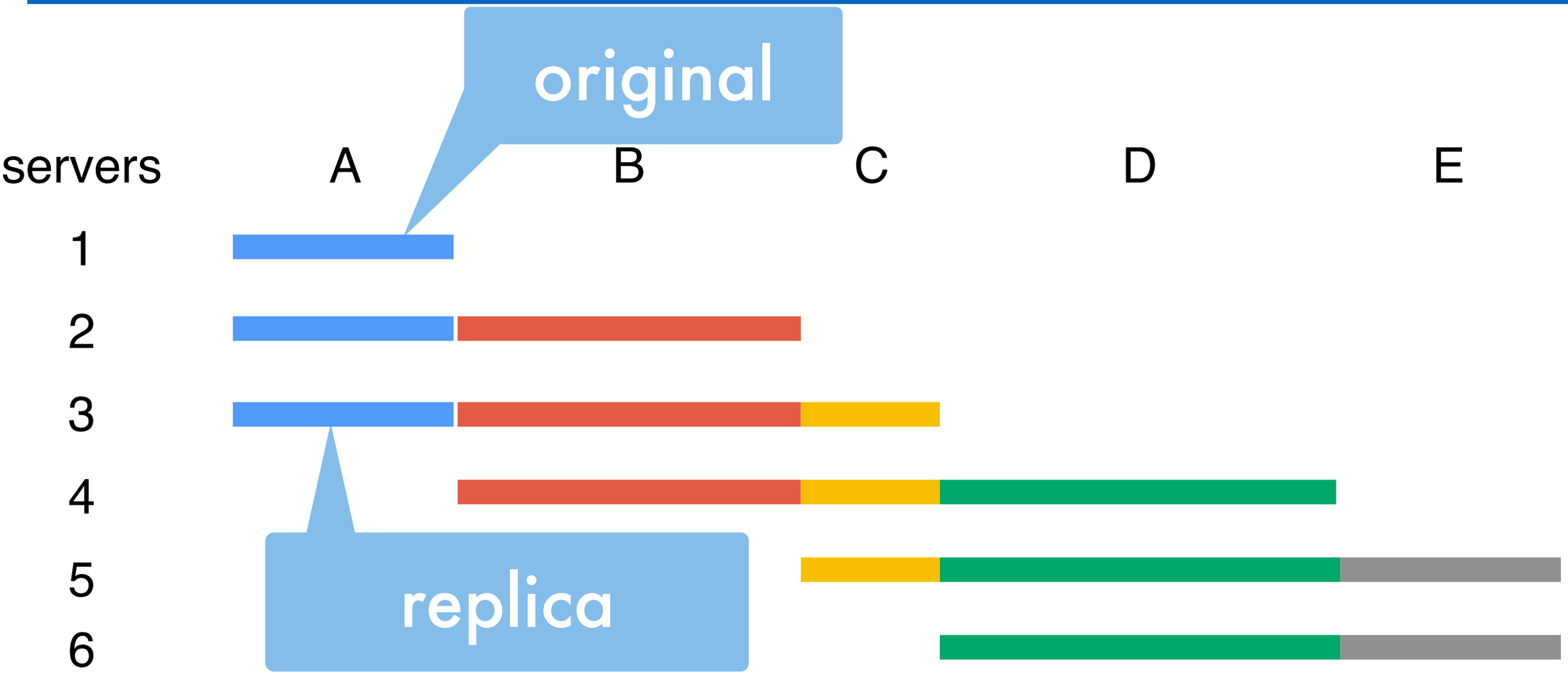
- Virtual servers
  - loadbalancing
  - multithreading
- DHT
  - contiguous key range for clients
  - easy bulk sync
  - easy insertion of servers

Key Replication

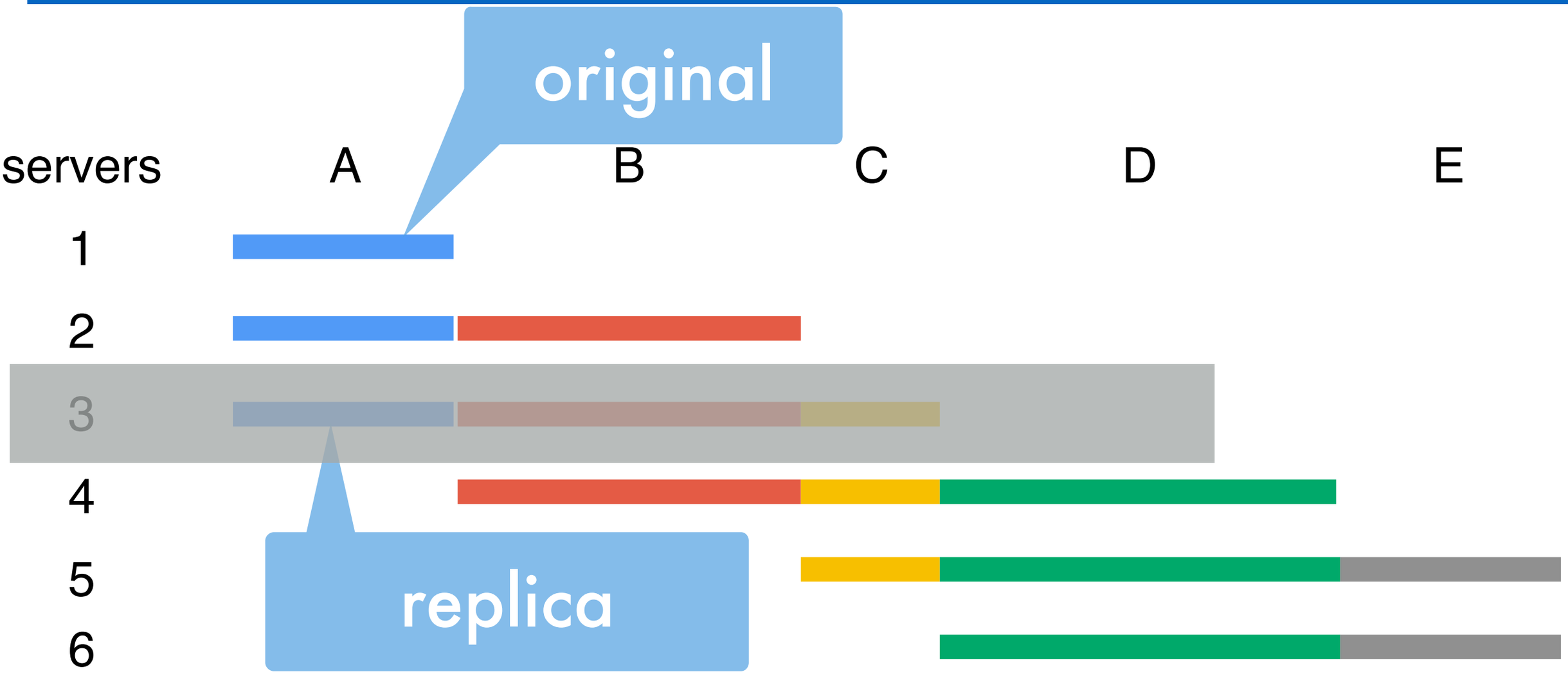


- Each segment is owned by one virtual server
- Subsequent machines hold replicas
- Easy fallback
- Easy insertion / repair
- Dynamic load balancing (important e.g. on EC2)

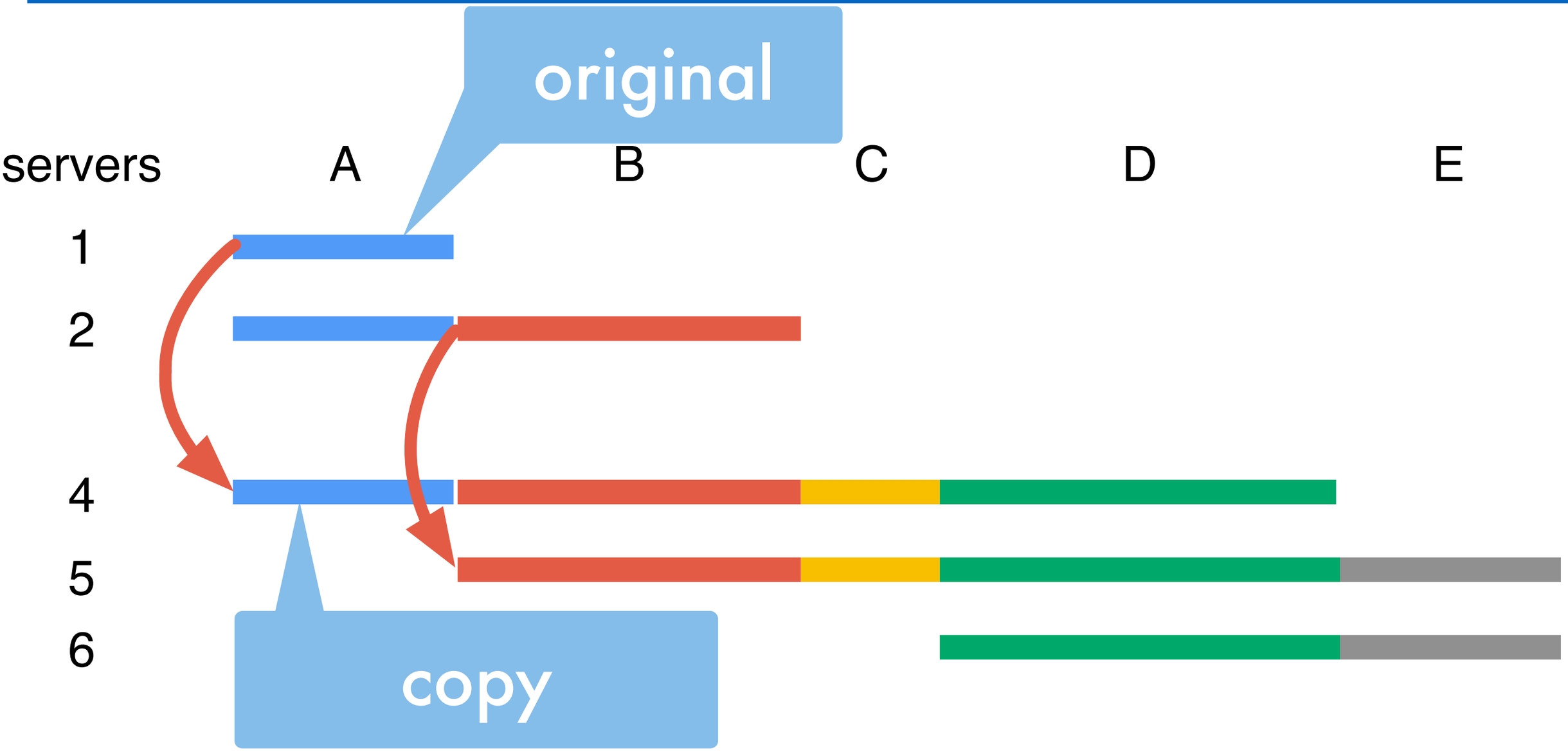
Key layout



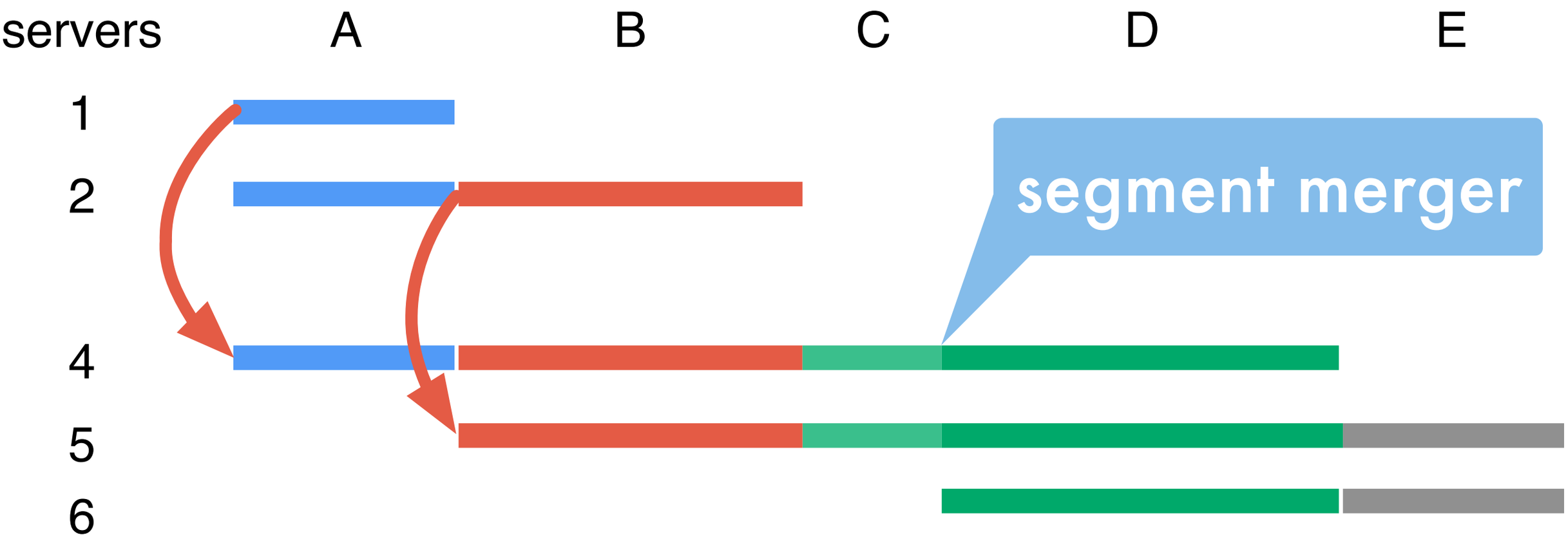
Key layout



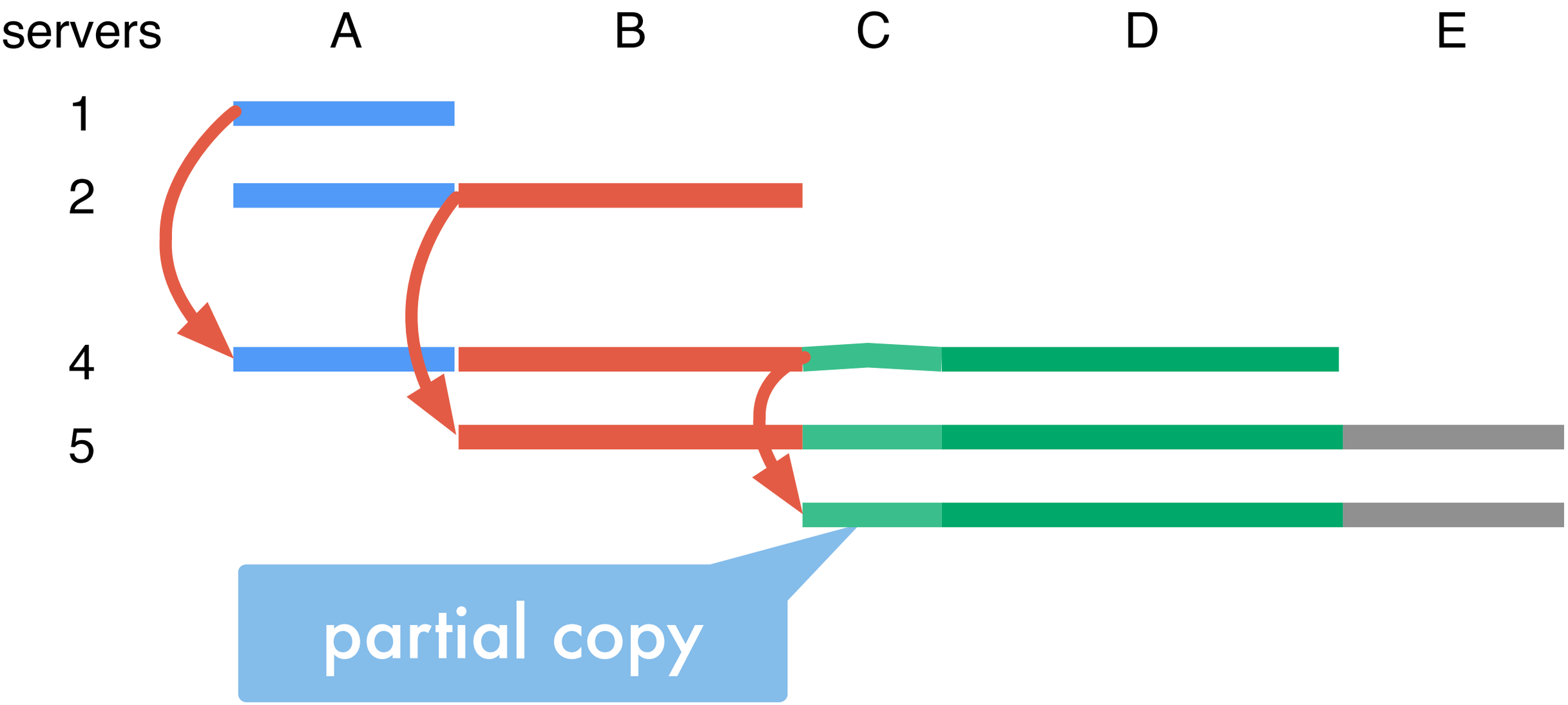
Key layout



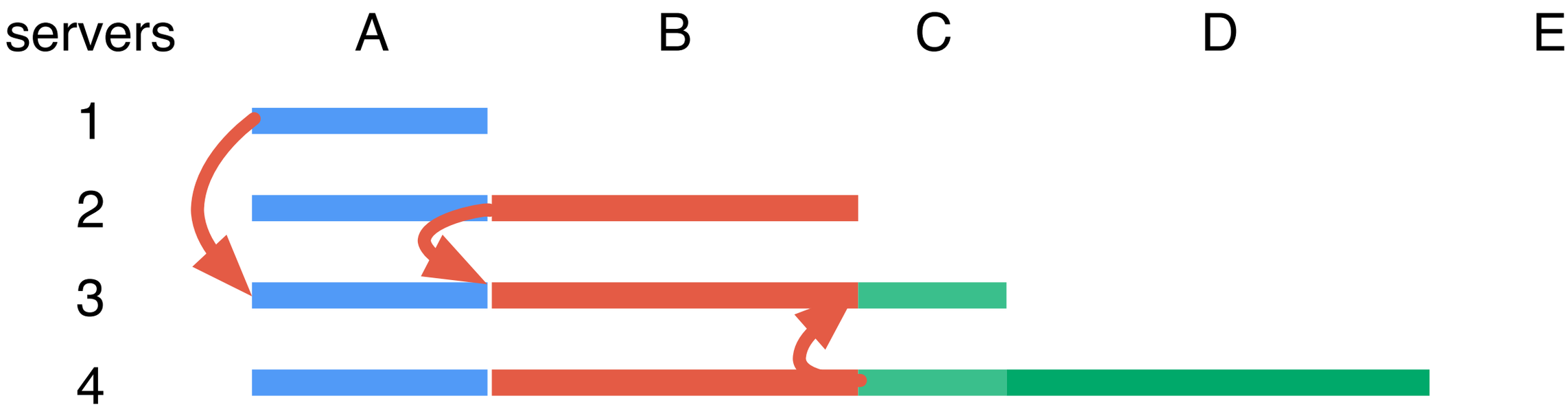
Key layout



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

Communication

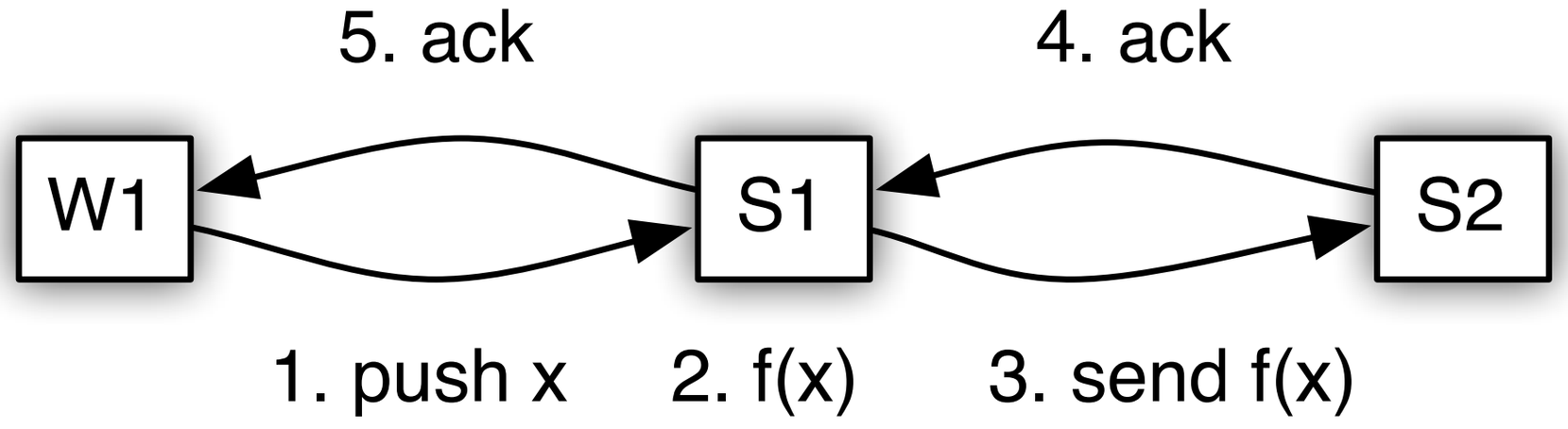
Communication tools

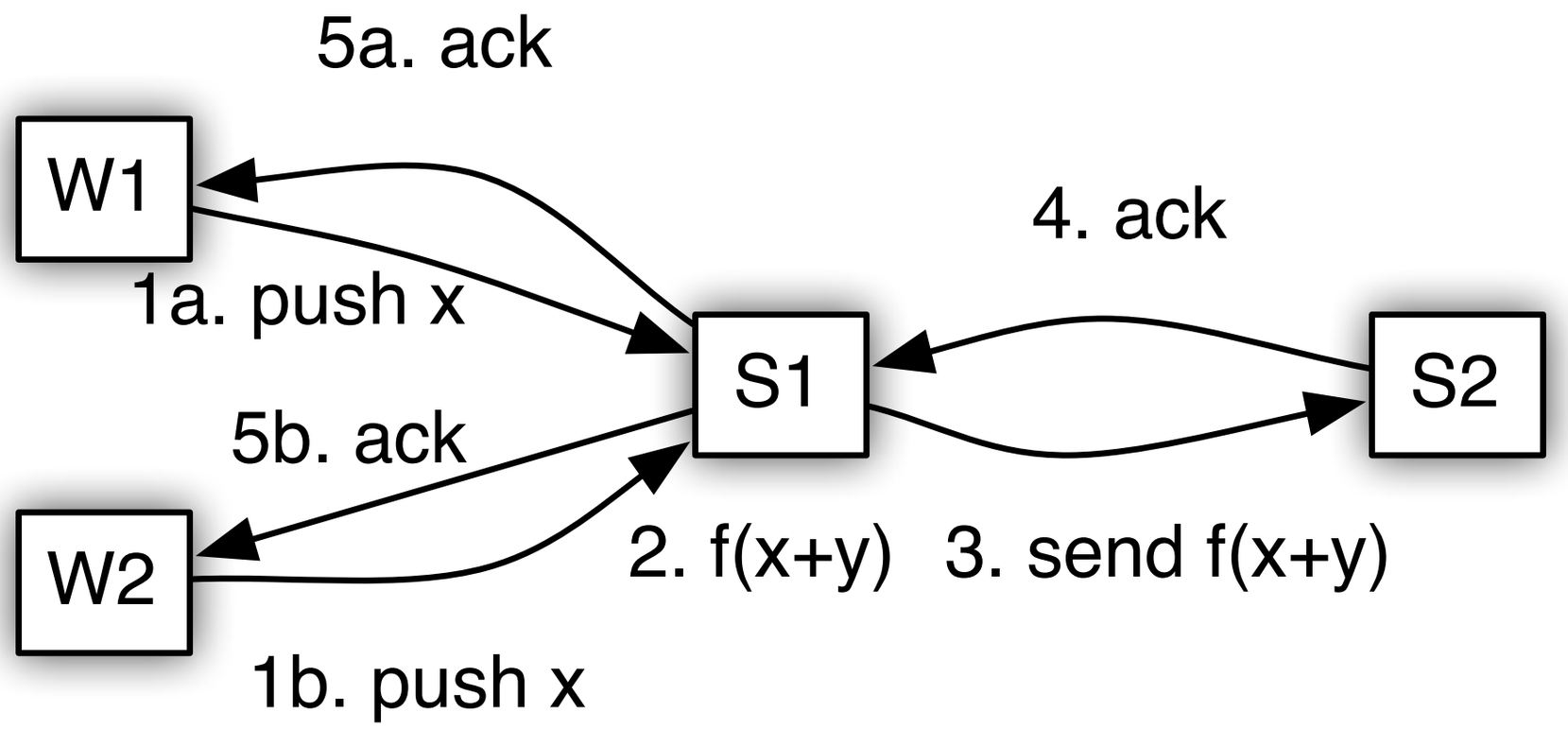
- ZeroMQ for messages
  - memcached way too slow (and no locks)
  - ICE is slow (YahooLDA uses it … mea culpa)
  - MPI is stone age awkward
- Google Protobuf for serialization
  good message compression
- Common pitfalls
  - Too small messages
  - Large metadata overhead

aggregation

compression
+ sparsity

Message Aggregation on Server

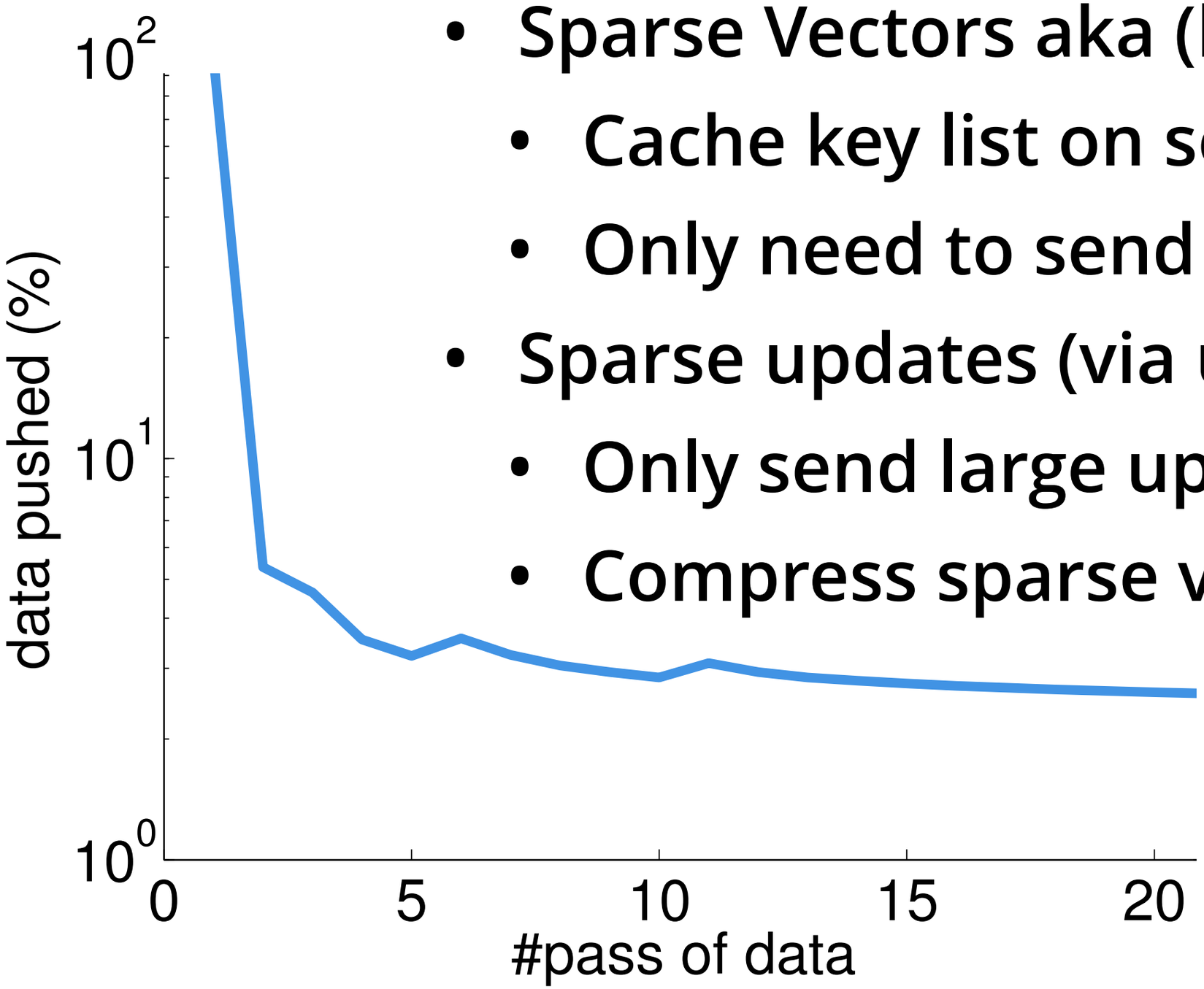




Message Compression

- 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

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

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)

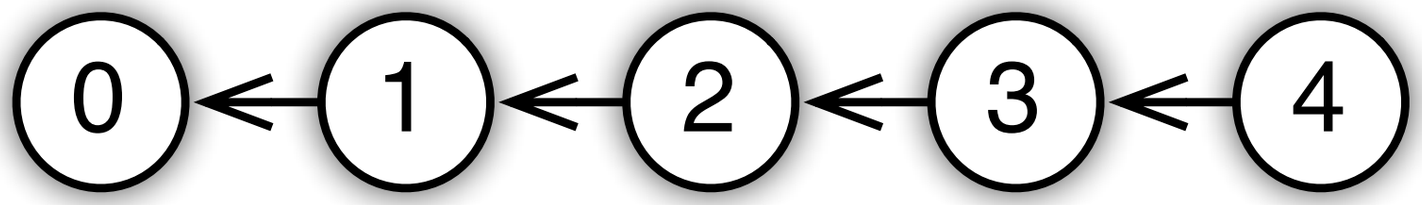
Mailbox design

- Postmaster
  handles server liveness (i.e. master controller)
- Postoffice
  messages arrive here
- Van
  holds the message
- Box
  holds the actual datatype

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

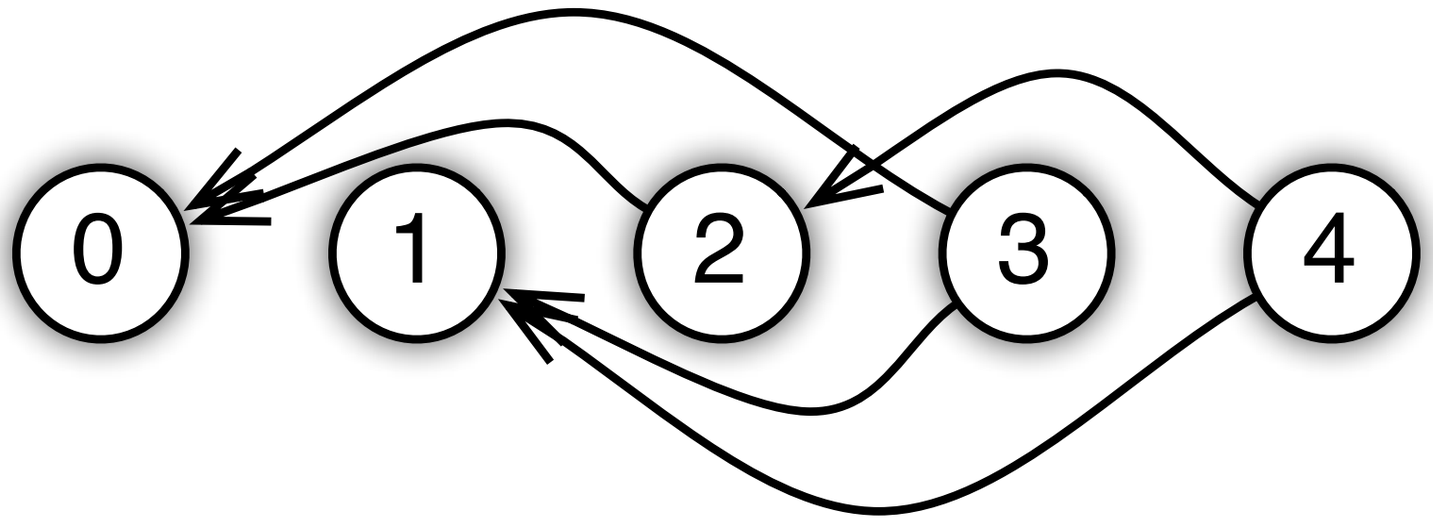
Consistency models

(a) Sequential



(b) Eventual

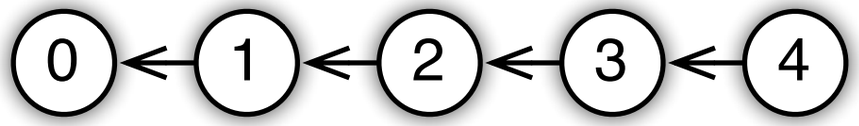
(c) Bounded delay



via task processing engine on client/controller

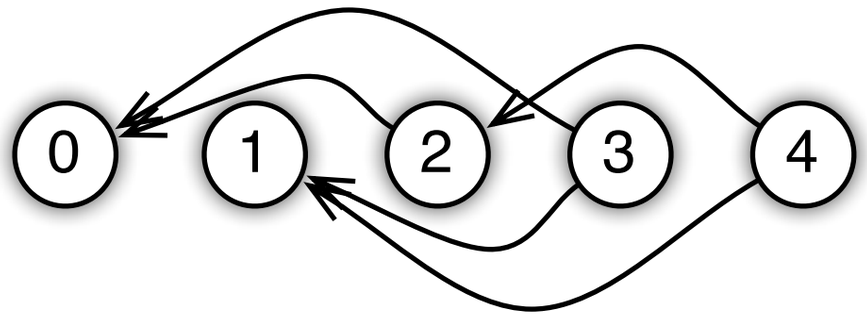
Consistency models

(a) Sequential



(b) Eventual

(c) Bounded delay

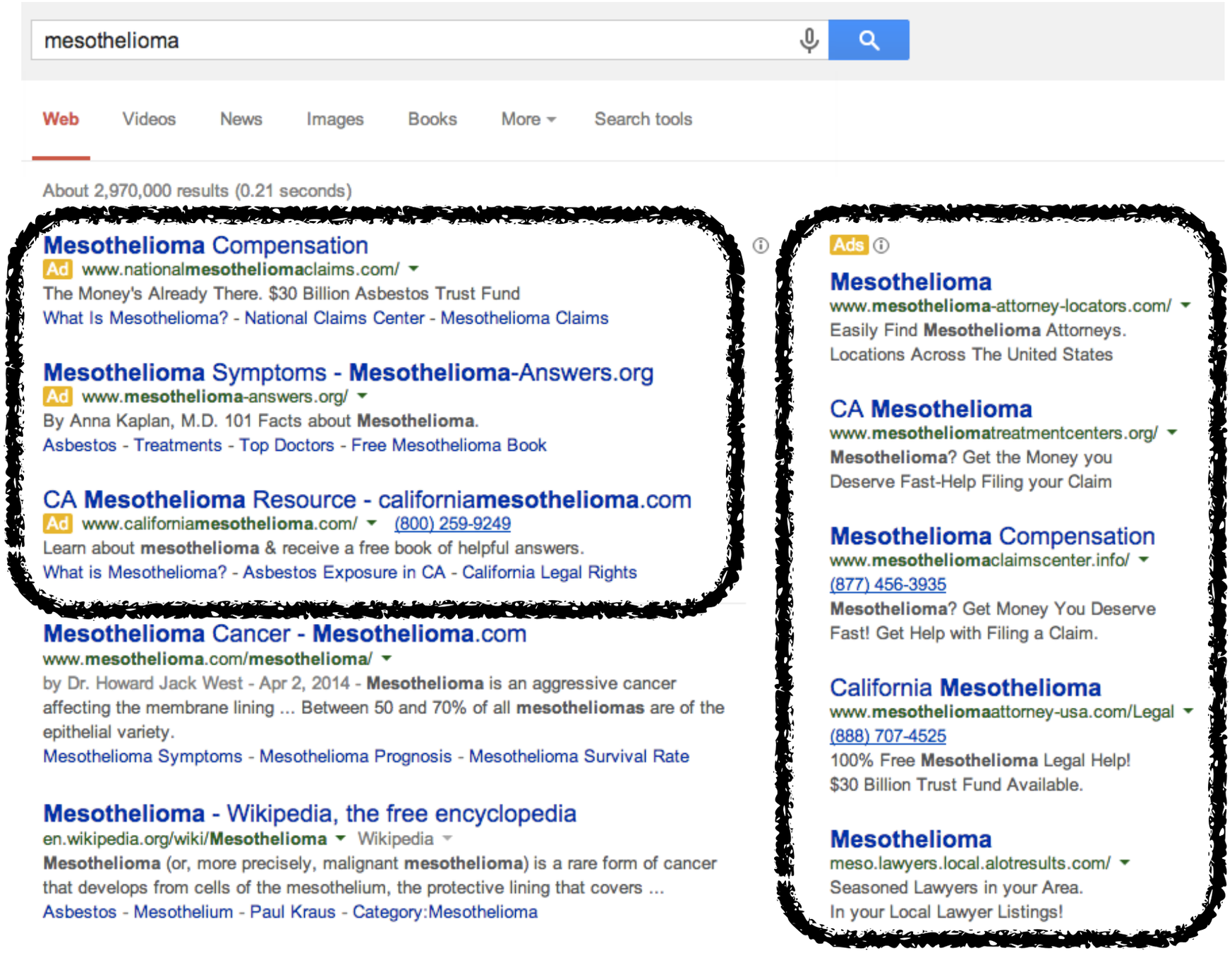


- Change dependency on the fly
- Task granularity programmatically defined (small or large tasks)
- Subtree controlled by worker

Models

Logistic Regression

Recall - Computational Advertising



sponsored search picks position of ad using

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

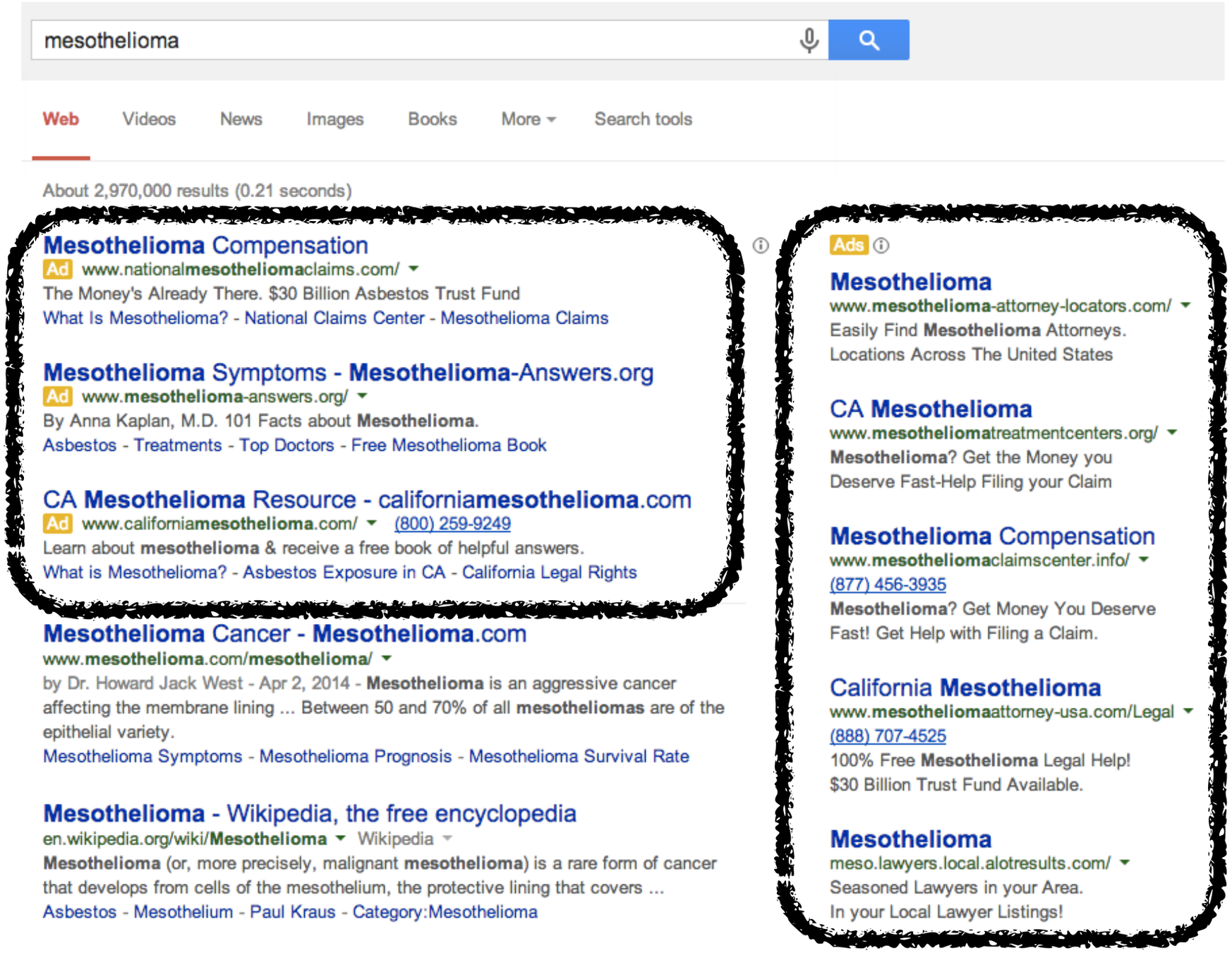
estimate it

4 million/minute

Google

Carnegie Mellon University

Recall - Computational Advertising



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

Estimating Probabilities

- **Logistic model (exponential family)**

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

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

- **Normalizing terms**

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

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

(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

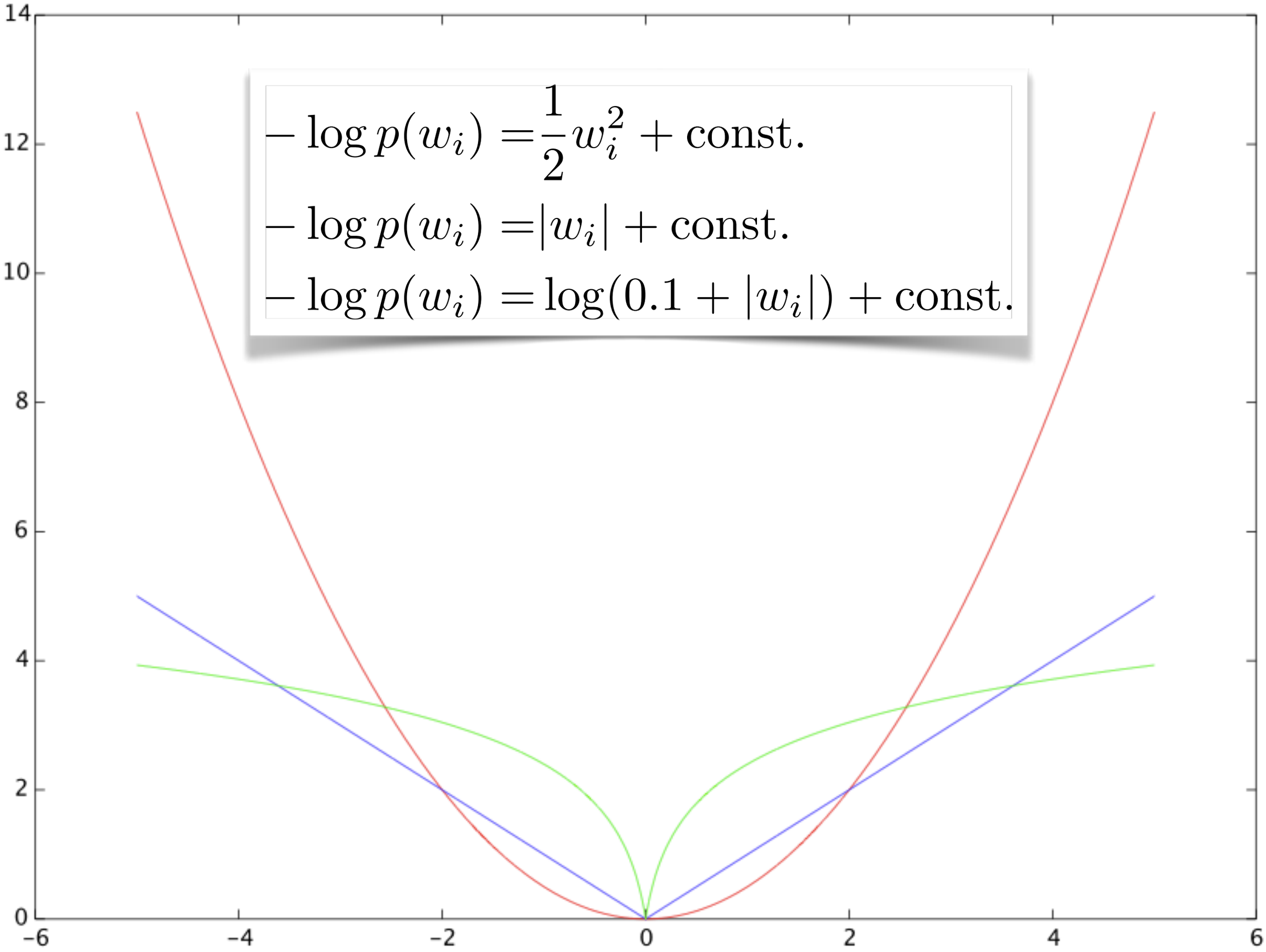
$$\underset{w}{\text{minimize}} - \log p(f|X, Y)$$

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

$$-\log p(w_i) = \frac{1}{2} w_i^2 + \text{const.}$$

$$-\log p(w_i) = |w_i| + \text{const.}$$

$$-\log p(w_i) = \log(0.1 + |w_i|) + \text{const.}$$



Google

Carnegie Mellon University

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)

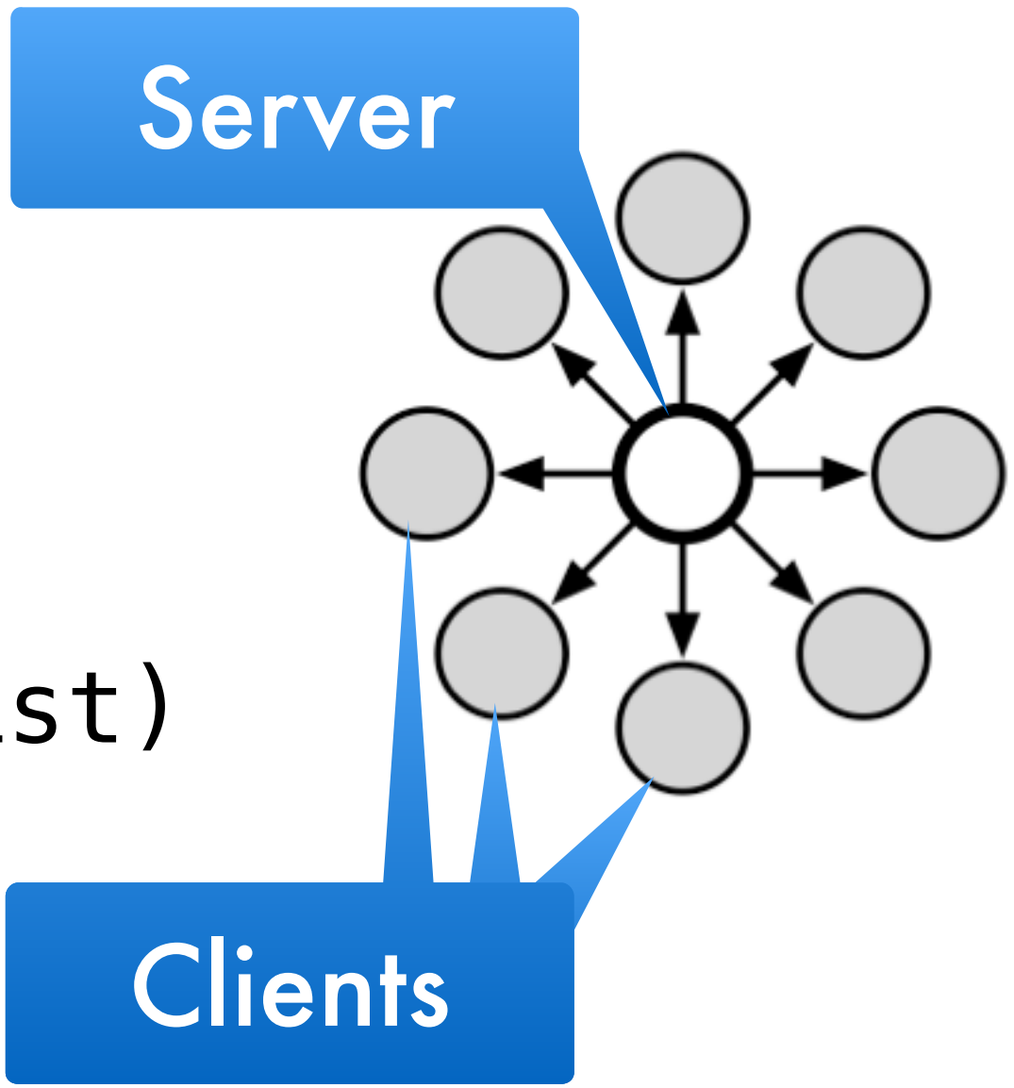
- Updates for l_1 are

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

more detail in Mu's tutorial

Generic Parallel Template

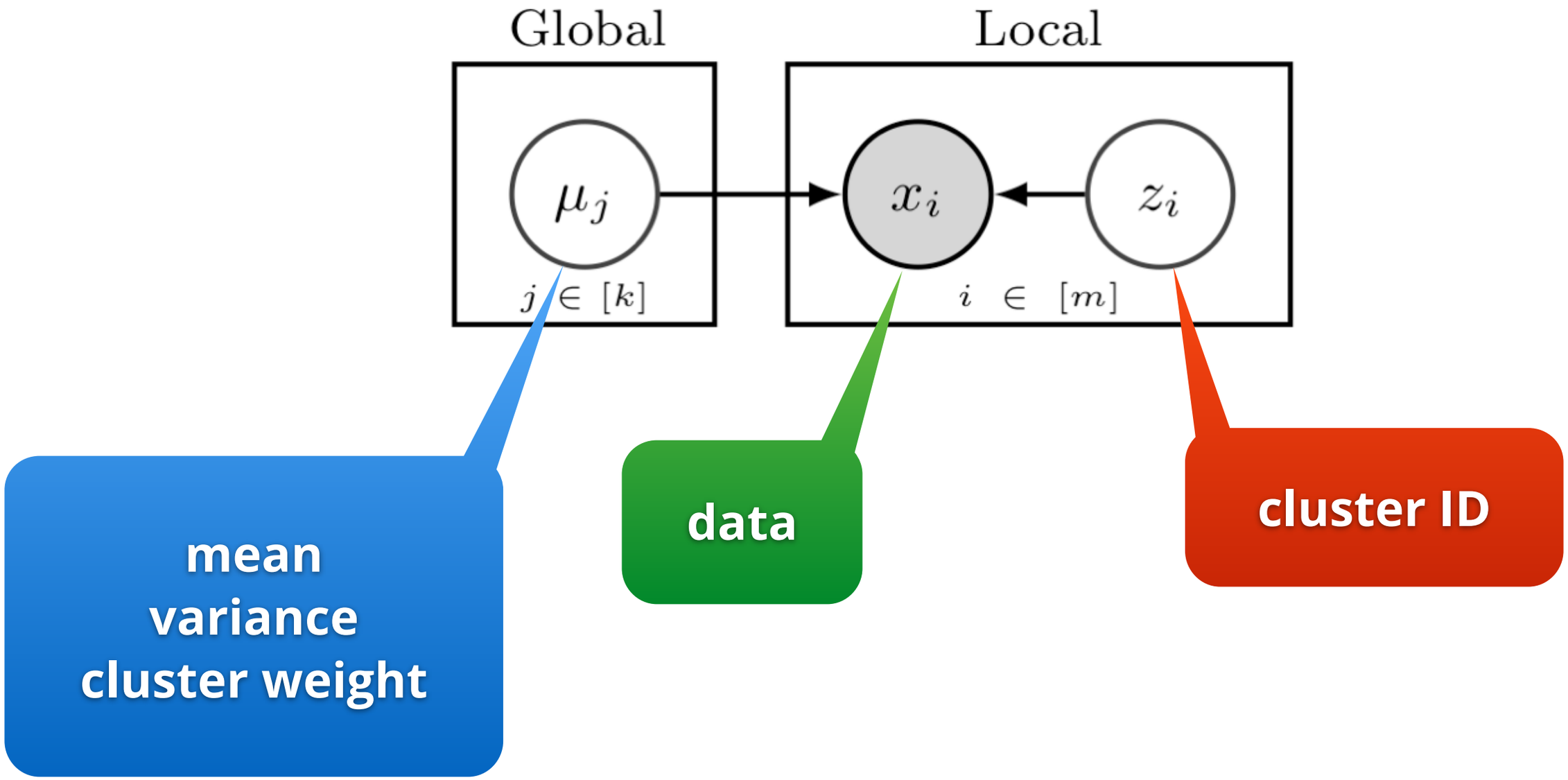
- Compute gradient on (subset of data) on each client
- Send gradient from client to server asynchronously
  `push(key_list,value_list)`
- Proximal gradient update on server
- Server returns parameters
  `pull(key_list,value_list)`



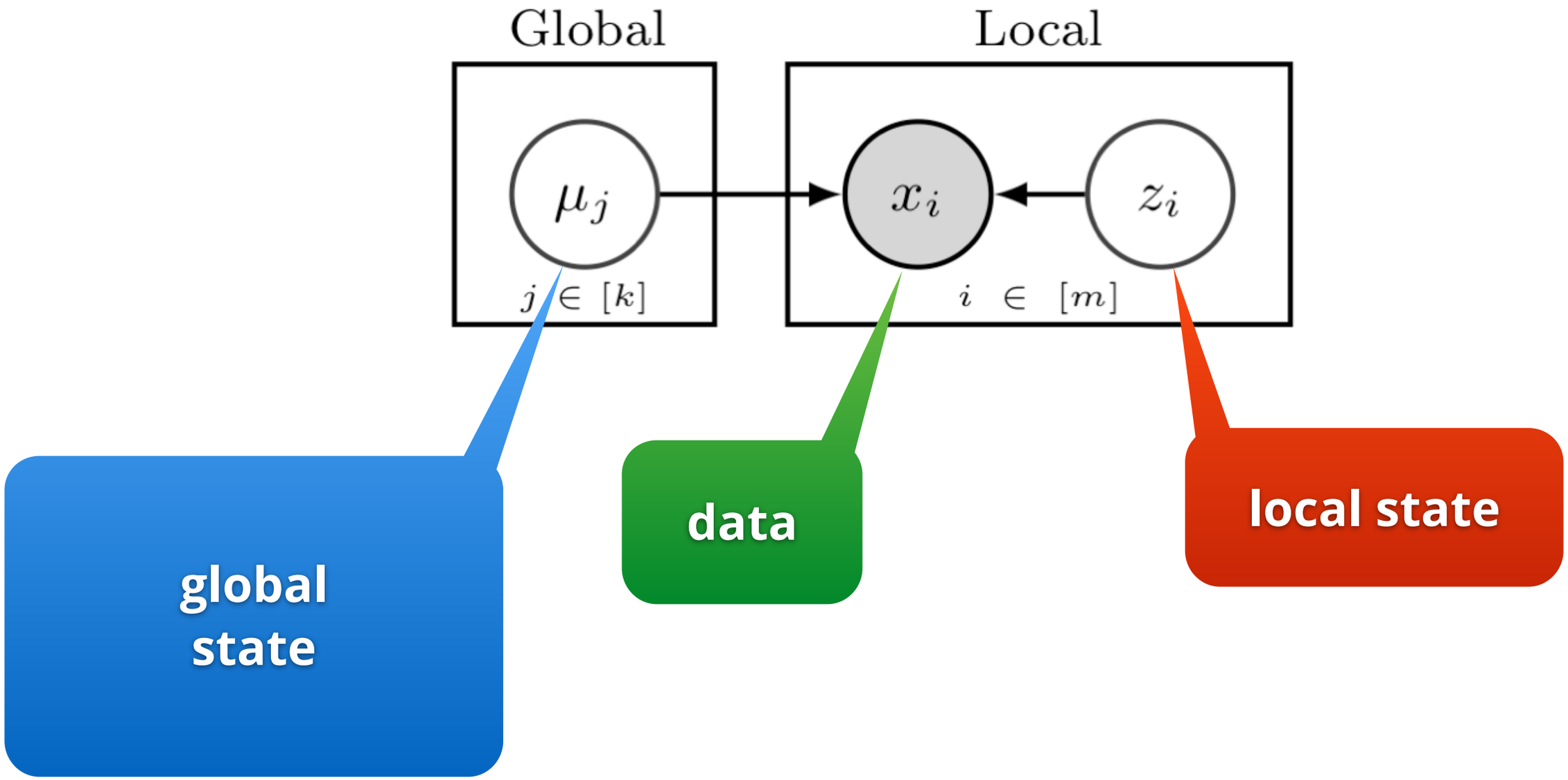
Scaling Latent Variable Models

e Mellon University

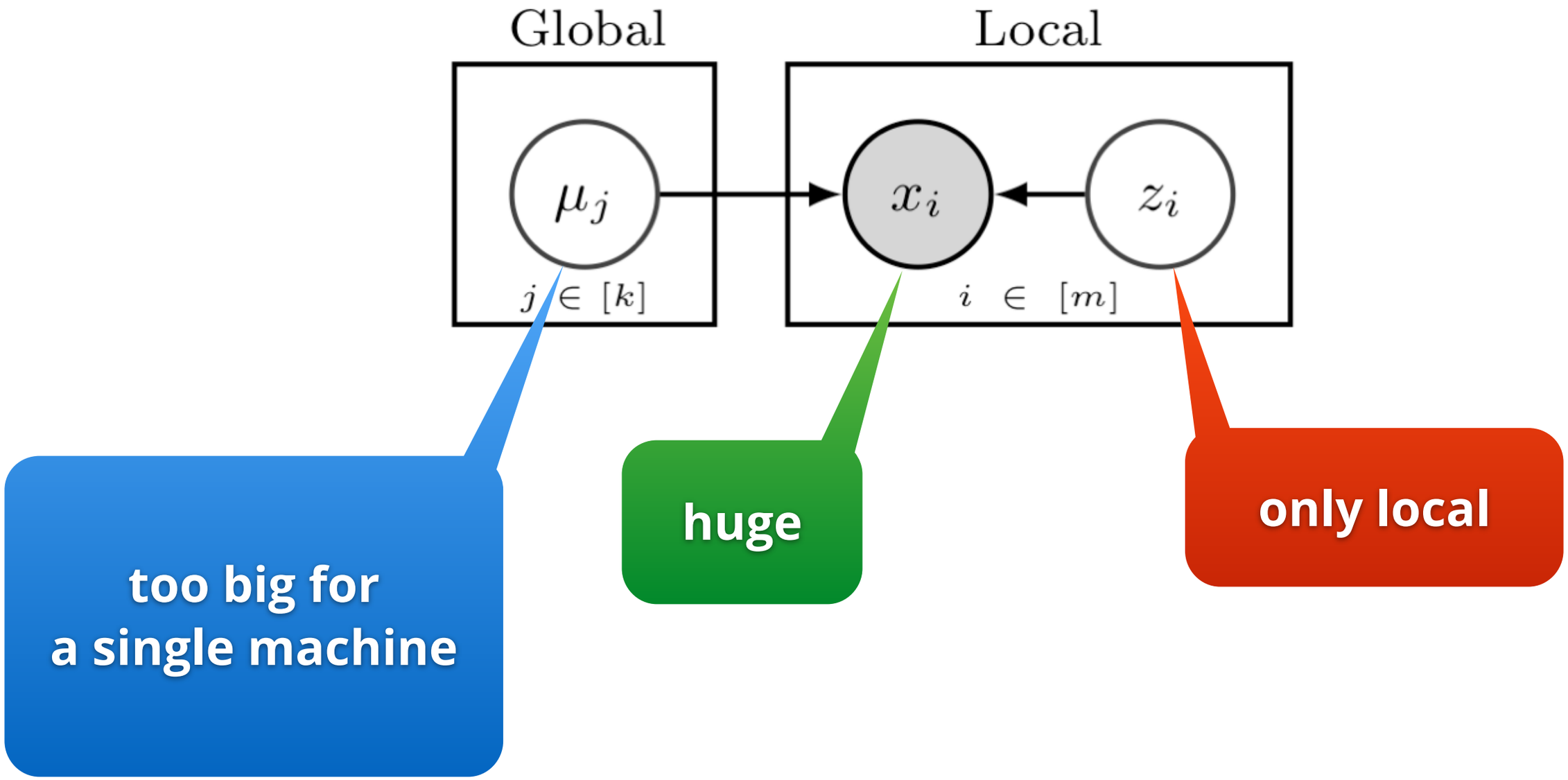
Clustering


Global
Local
μ_j
x_i
z_i
$j \in [k]$
$i \in [m]$
mean
variance
cluster weight
data
cluster ID

Clustering


Global
Local
μ_j
x_i
z_i
$j \in [k]$
$i \in [m]$
global state
data
local state

Clustering


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

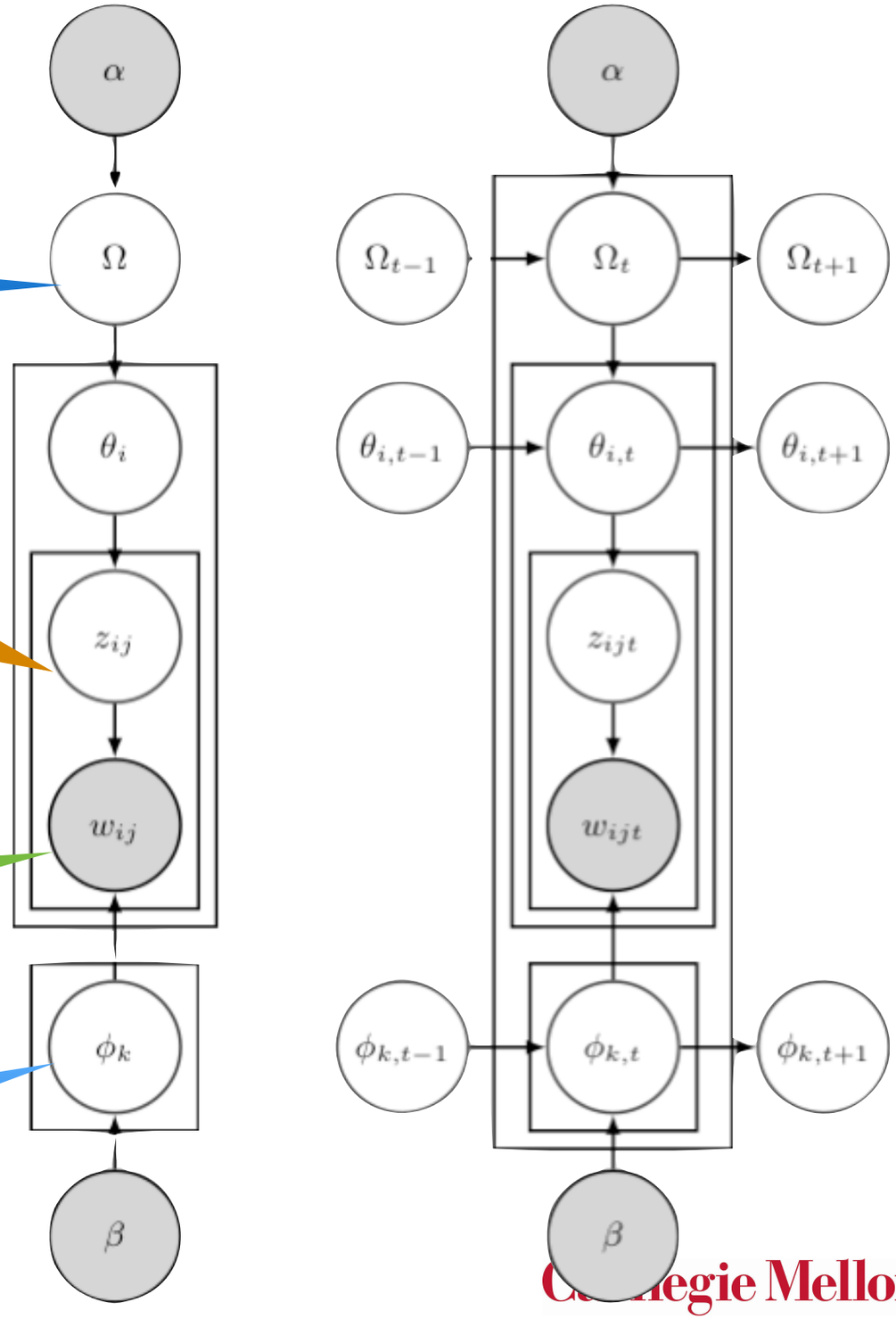
Fancy models

global state

local state

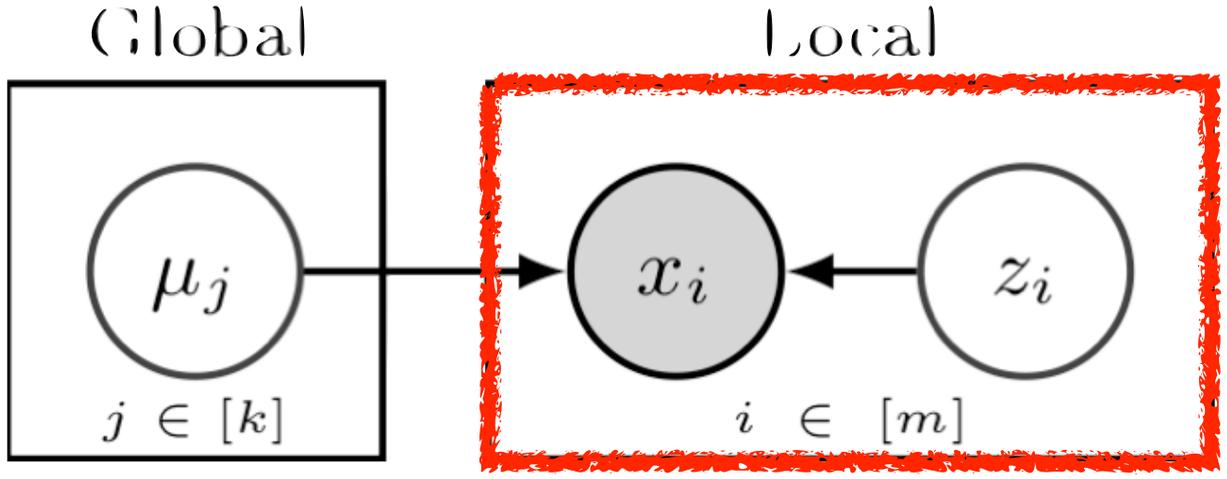
data

global state



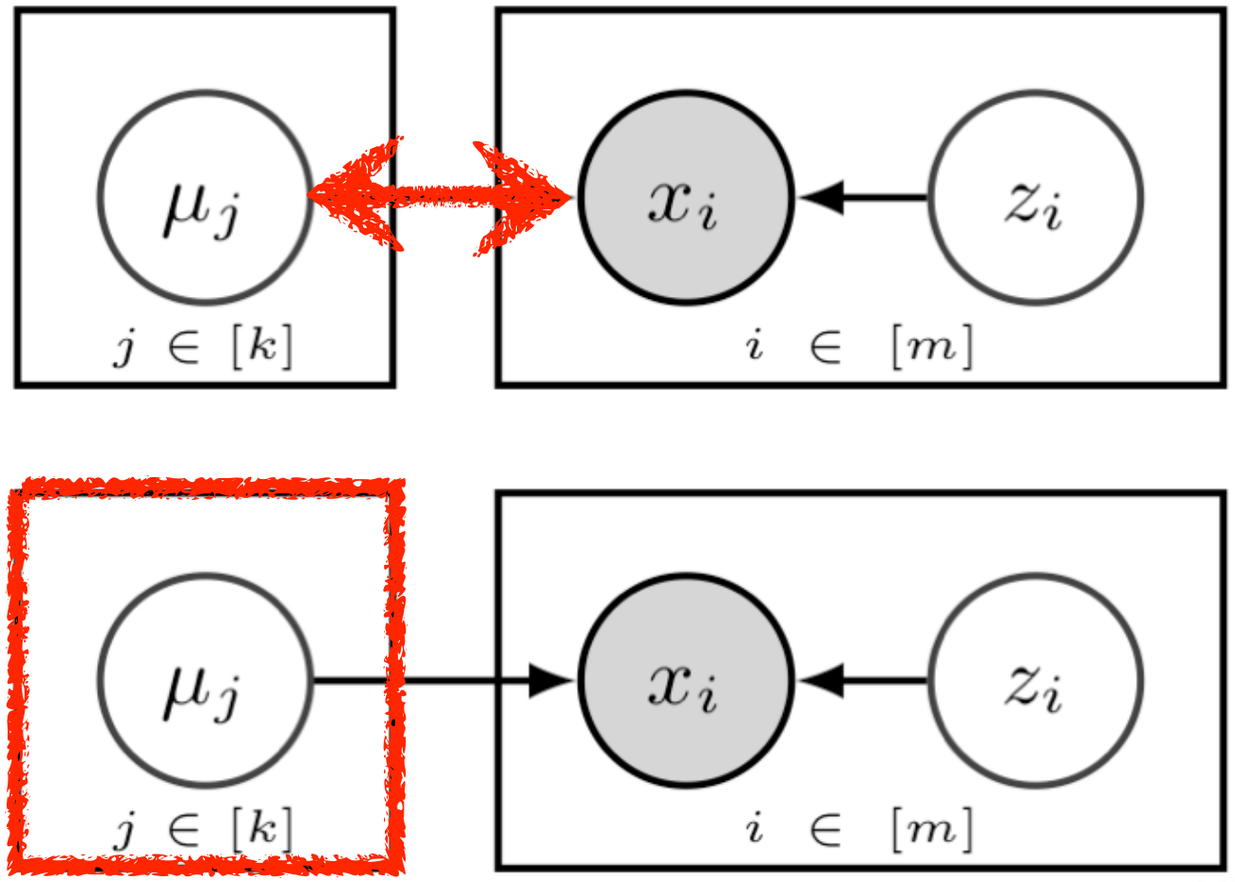
ParameterServer to the rescue

local state
is too large



does not fit
into memory

global state
is too large

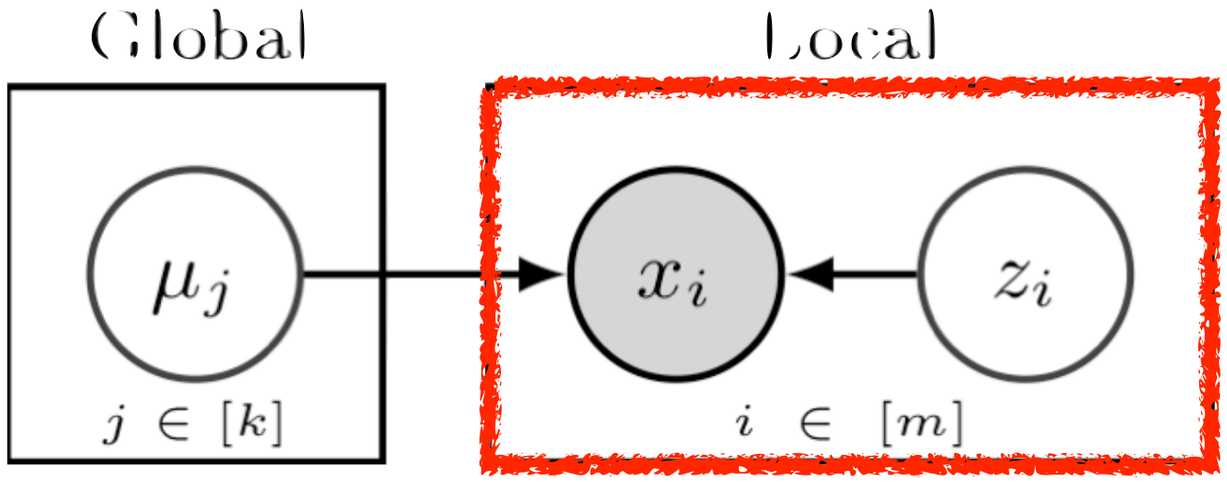


network load
& barriers

does not fit
into memory

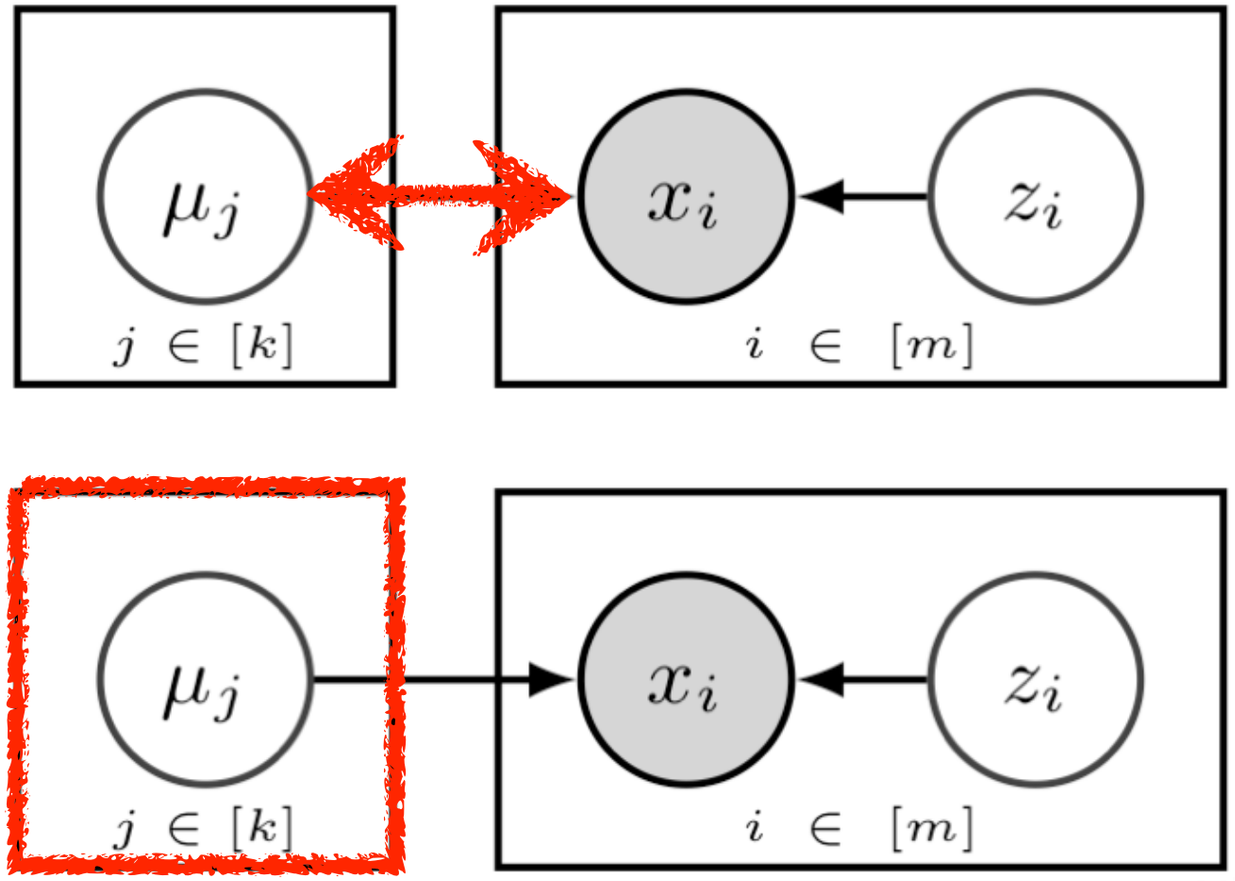
ParameterServer to the rescue

local state
is too large



stream local
data from disk

global state
is too large

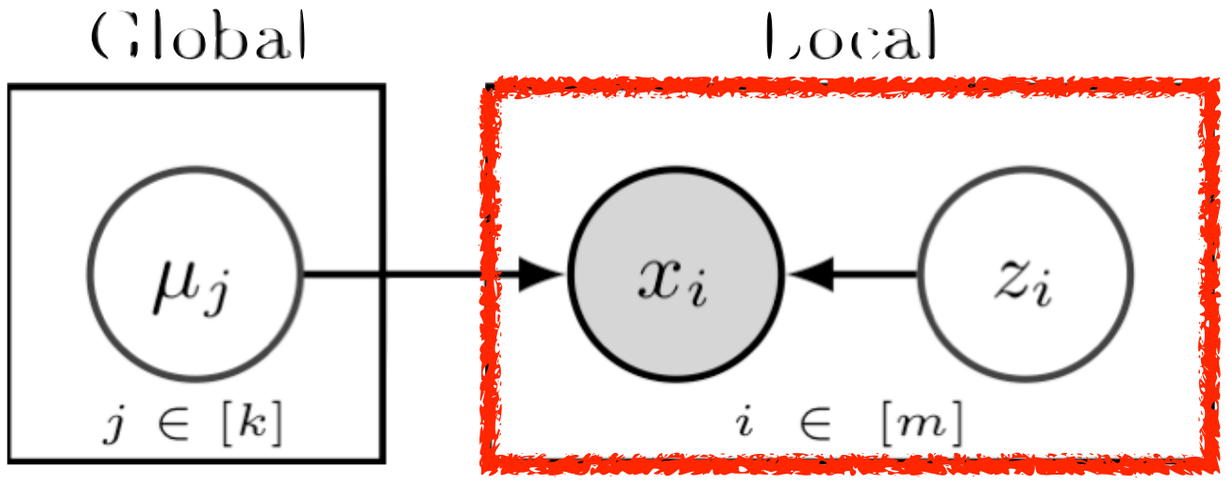


network load
& barriers

does not fit
into memory

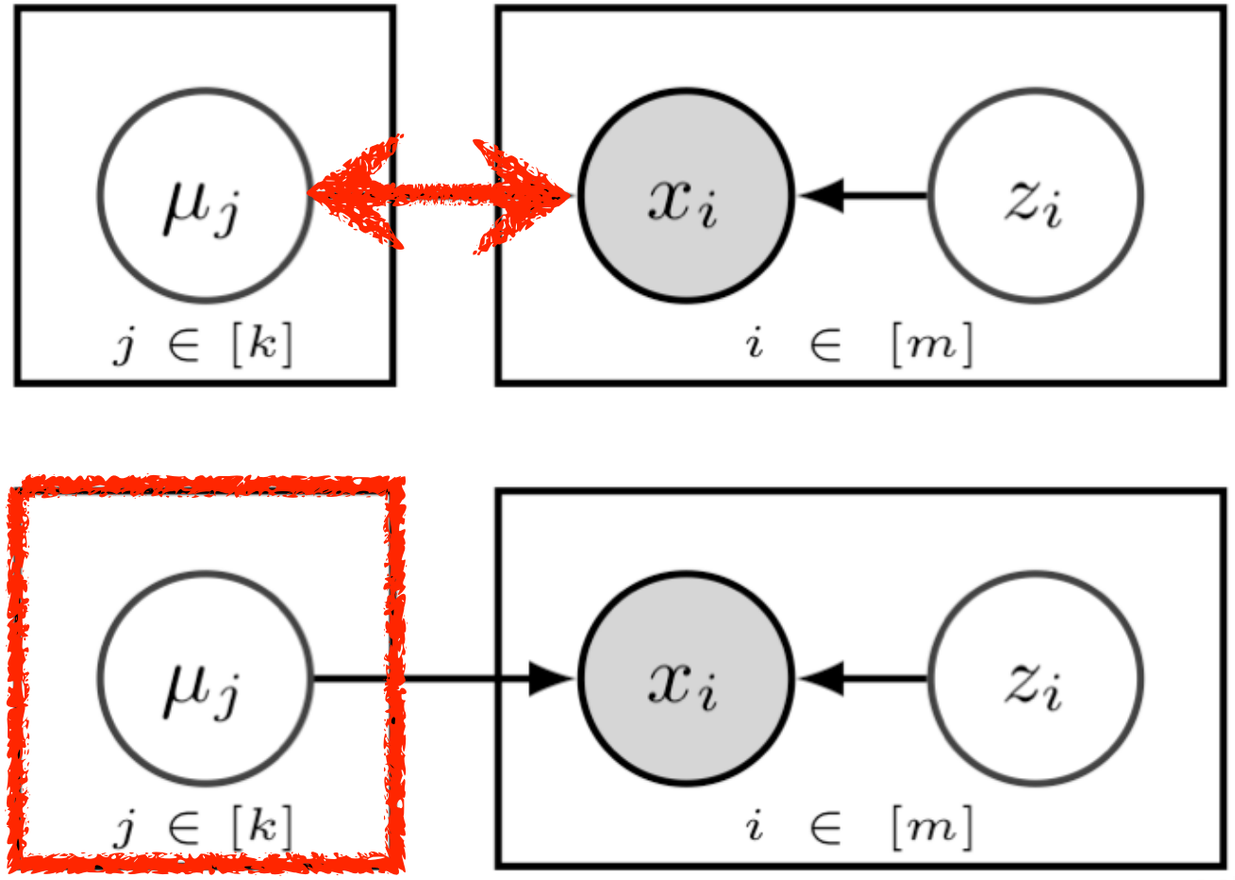
ParameterServer to the rescue

local state
is too large



stream local
data from disk

global state
is too large

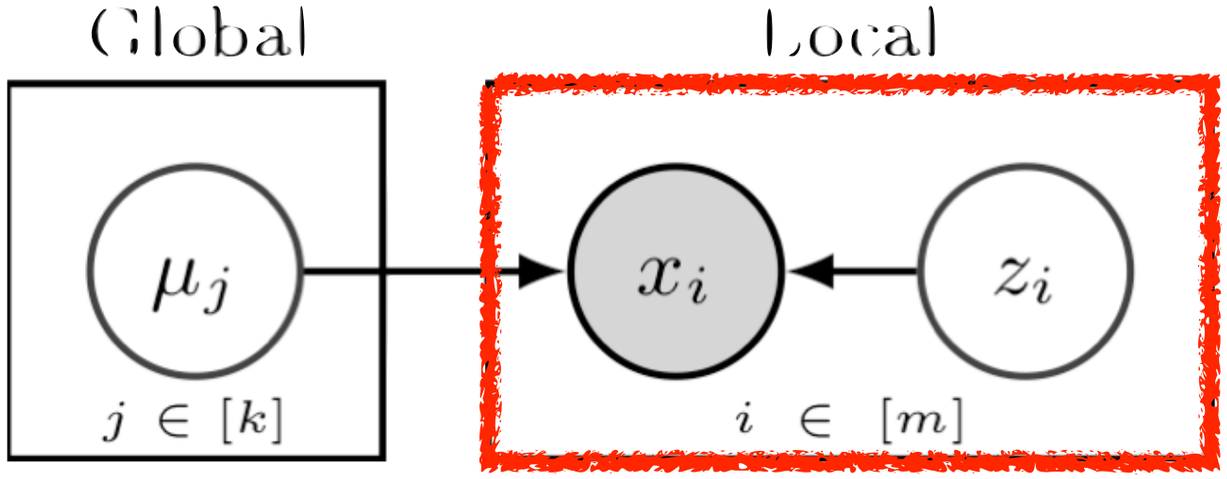


asynchronous
synchronization

does not fit
into memory

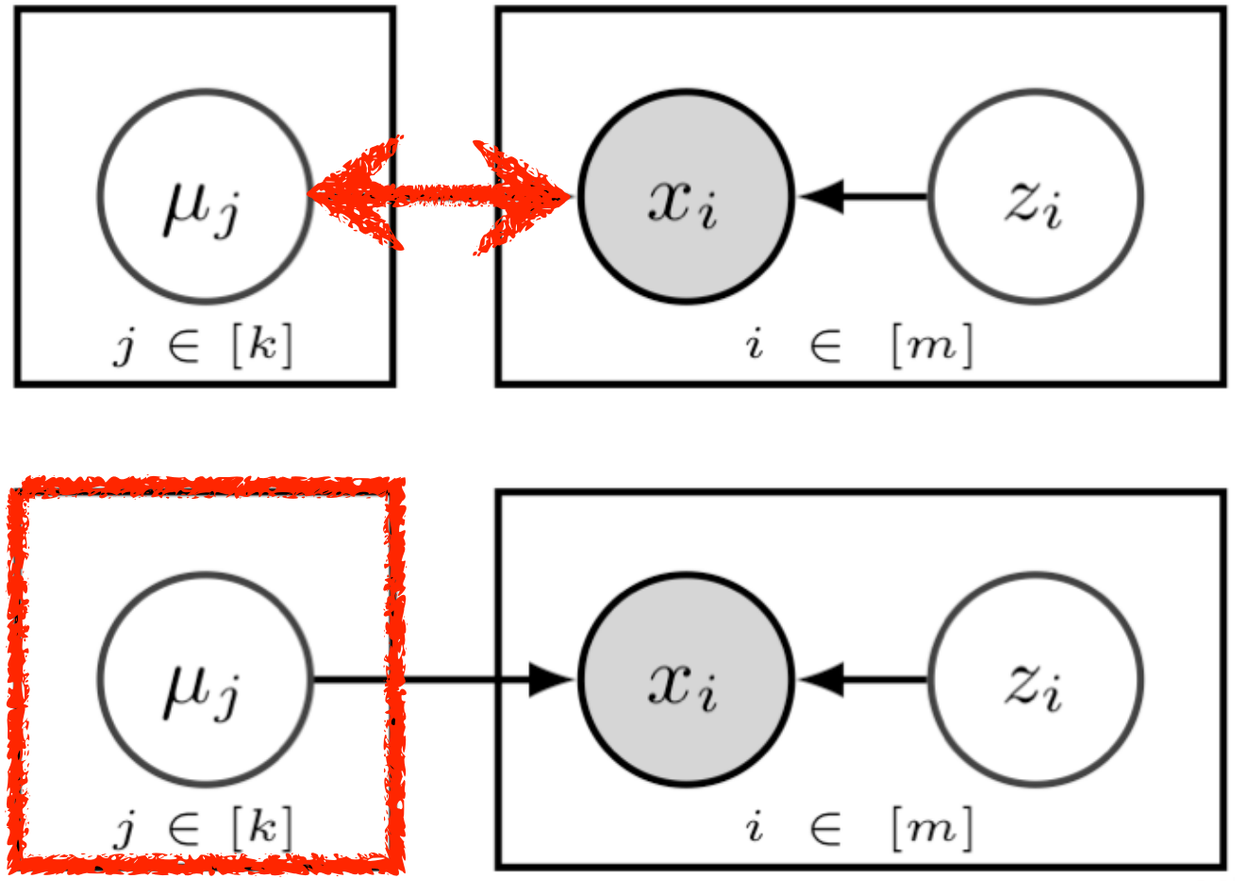
ParameterServer to the rescue

local state
is too large



stream local
data from disk

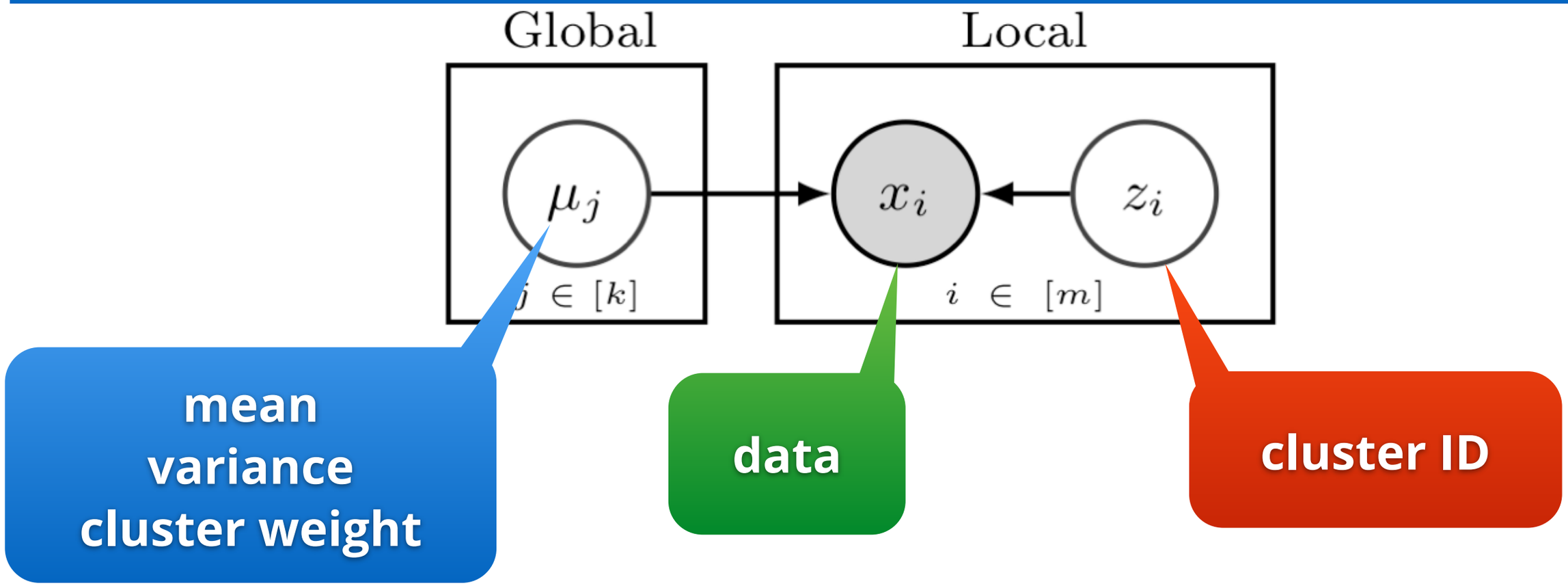
global state
is too large



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**

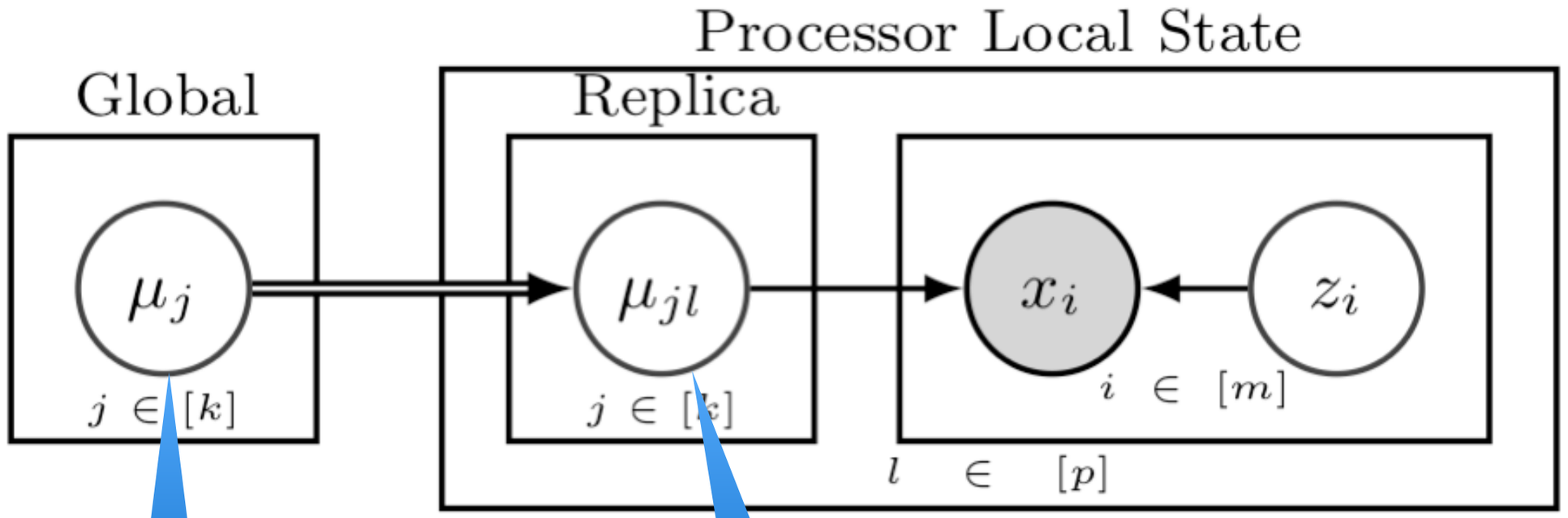
$$p(z_i = z|Z^{-i}) = \frac{n_z^{-i} + \alpha_z}{n - 1 + \sum_{z'} \alpha_{z'}}$$

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

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

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

Local and Global Variables

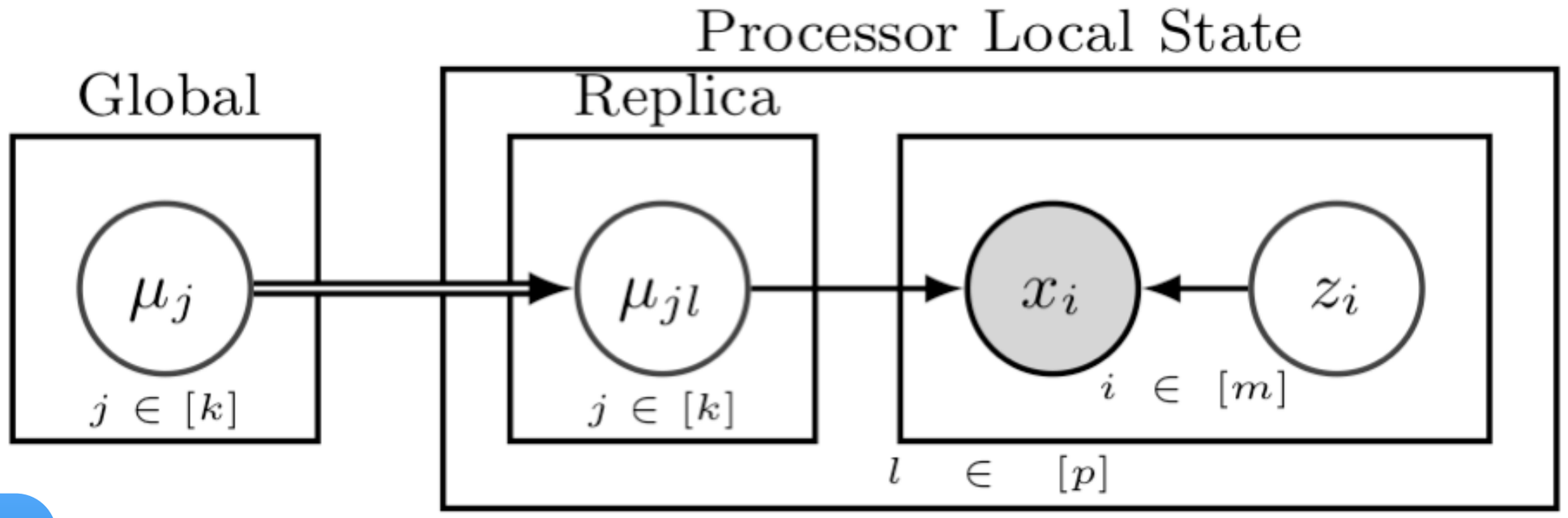


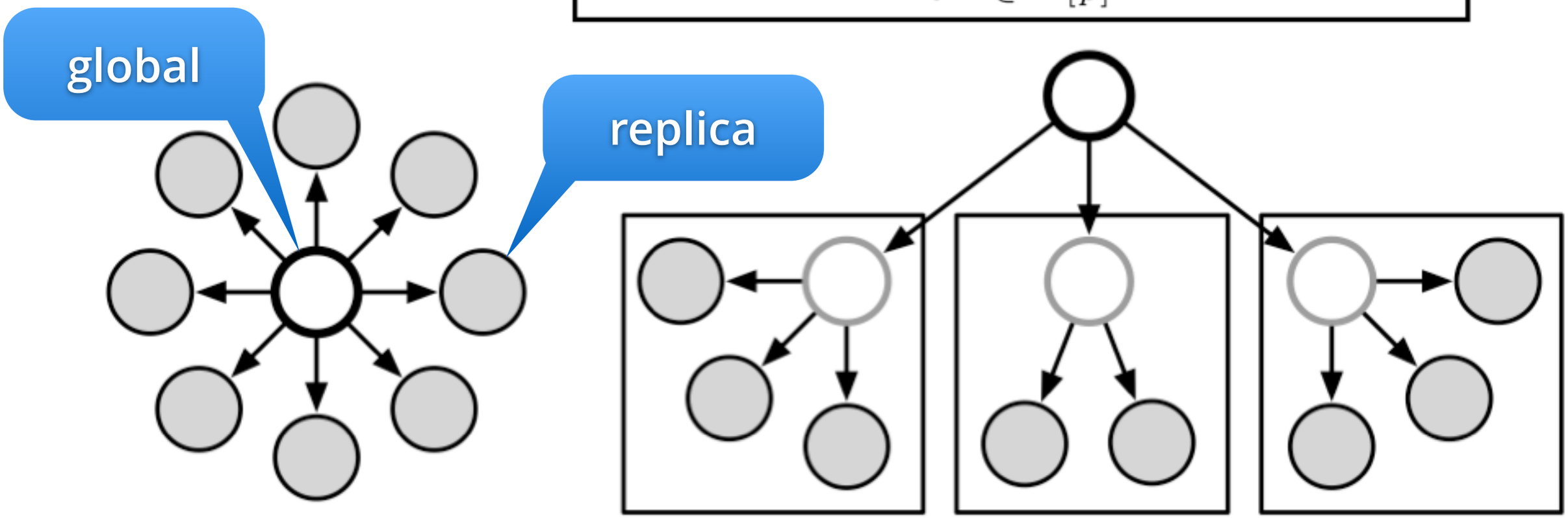
background sync

1 copy per machine

- 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

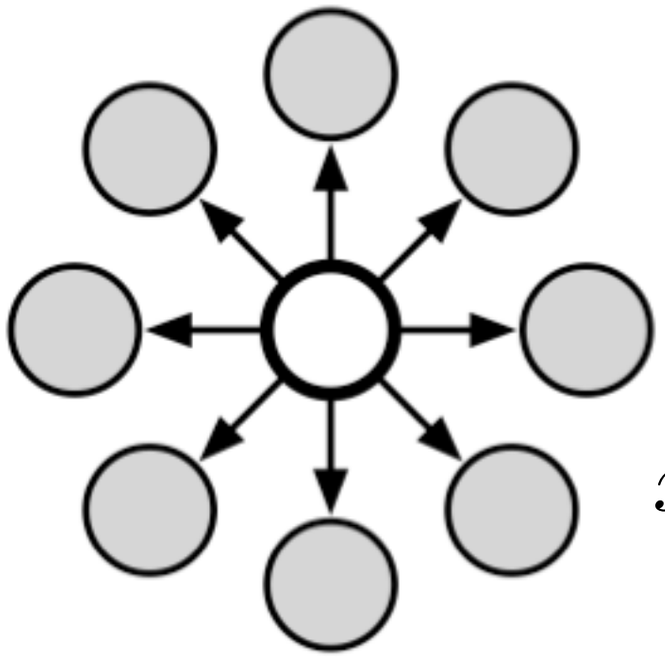
- Start with common state
- Child stores old and new state
- Parent keeps global state
- Transmit differences asynchronously
  - Inverse element for difference
  - Abelian group for commutativity
    (sum, log-sum, cyclic group, exponential families)

local to global

$$\delta \leftarrow x \ominus x^{\text{old}}$$

$$x^{\text{old}} \leftarrow x$$

$$x^{\text{global}} \leftarrow x^{\text{global}} \oplus \delta$$



global to local

$$x \leftarrow x \oplus (x^{\text{global}} \ominus x^{\text{old}})$$

$$x^{\text{old}} \leftarrow x^{\text{global}}$$

Google

Carnegie Mellon University

Models

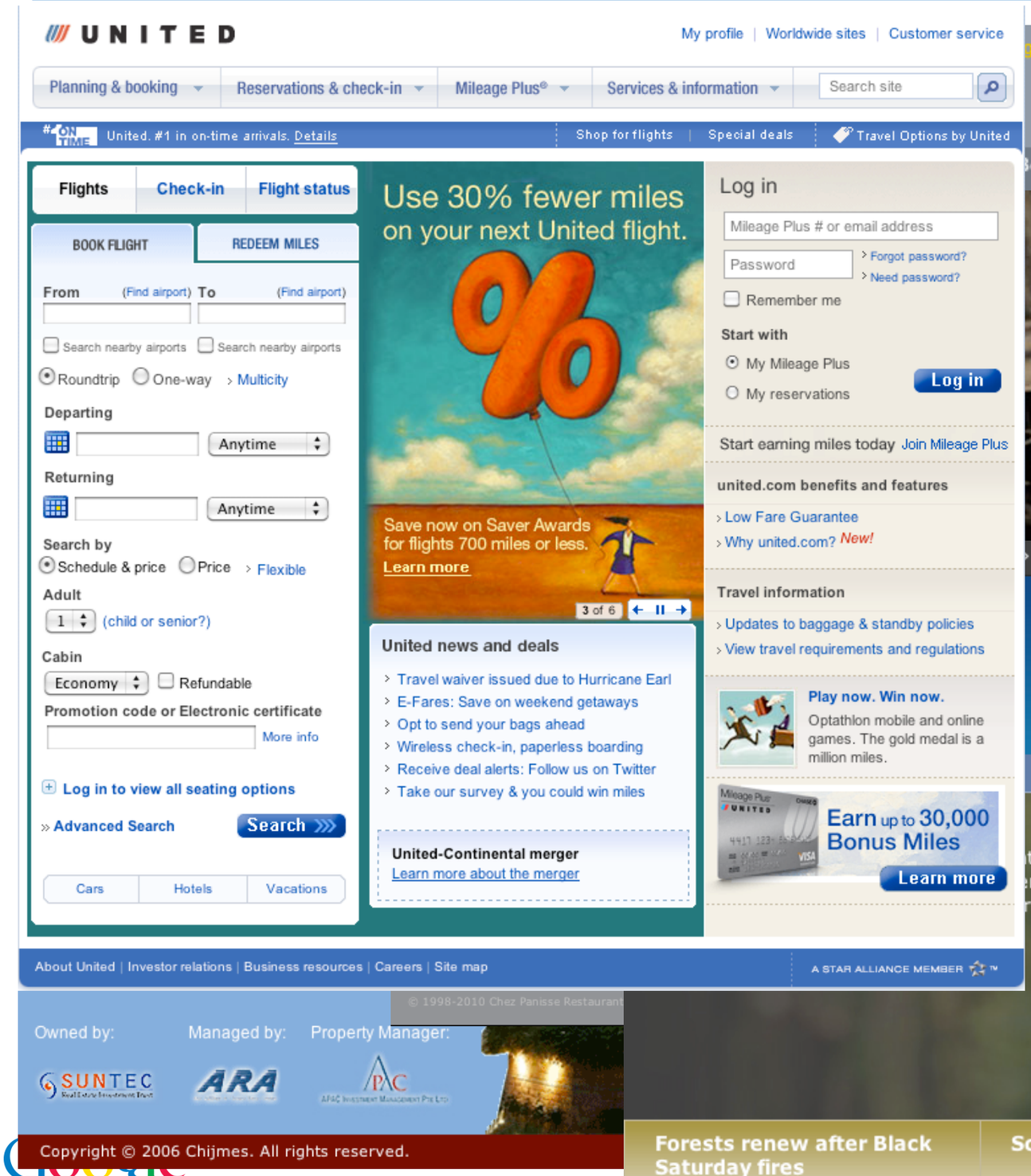
Google

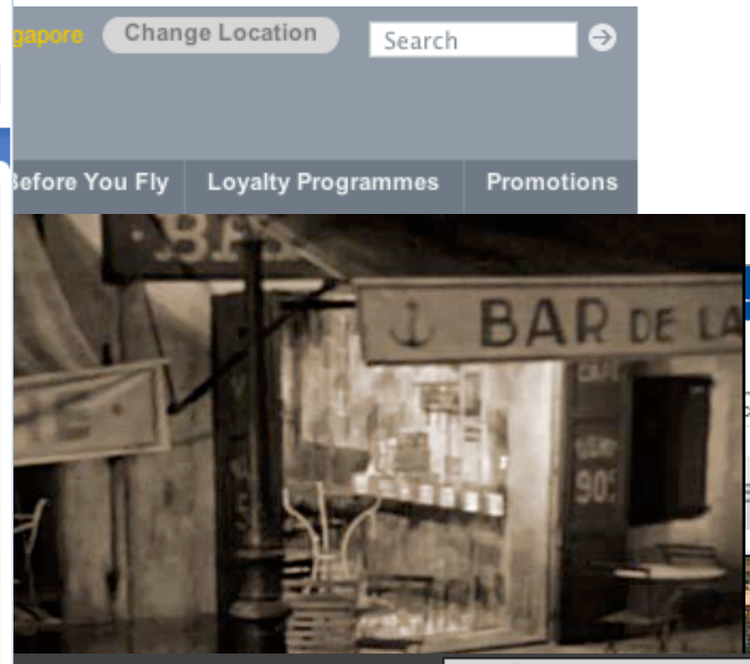
Carnegie Mellon University

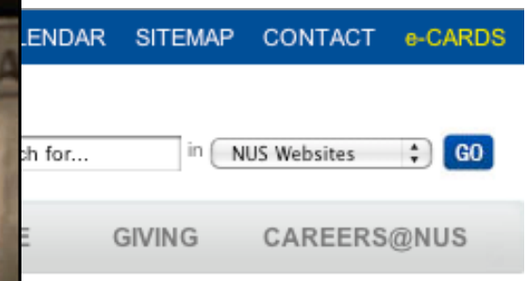
Grouping objects

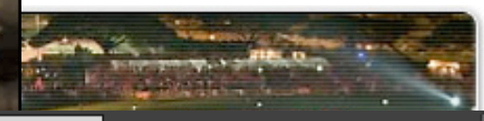
Grouping objects

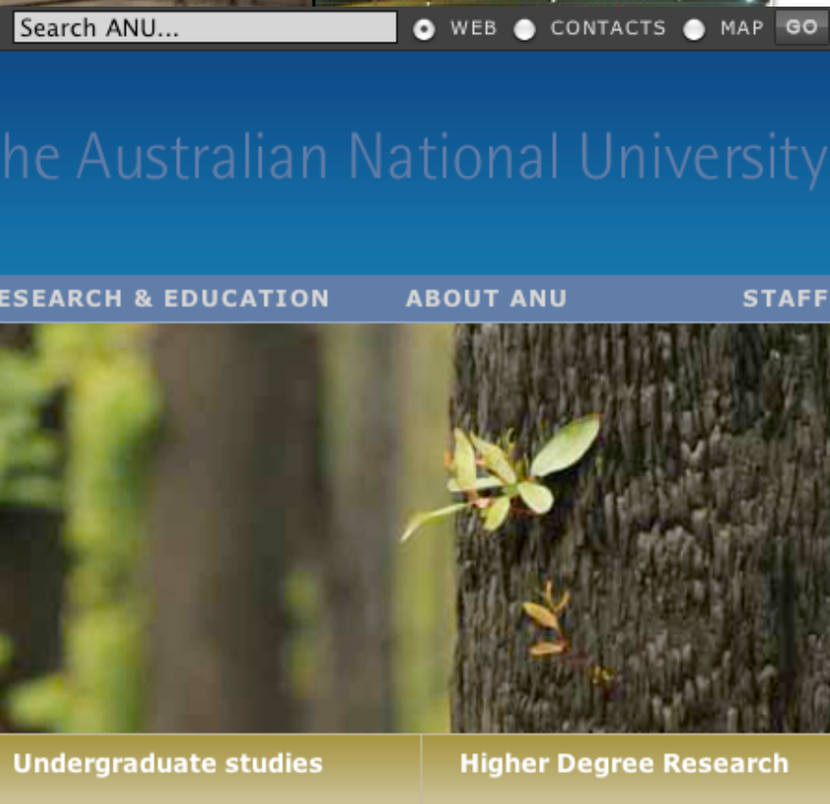
Grouping objects



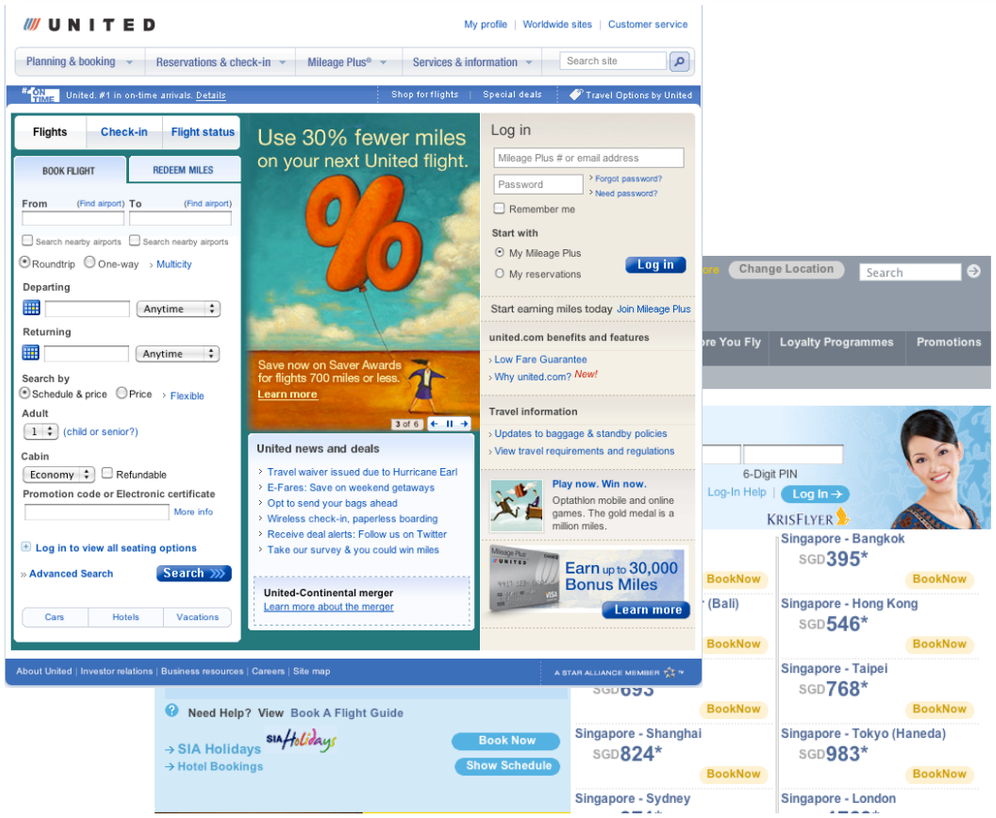


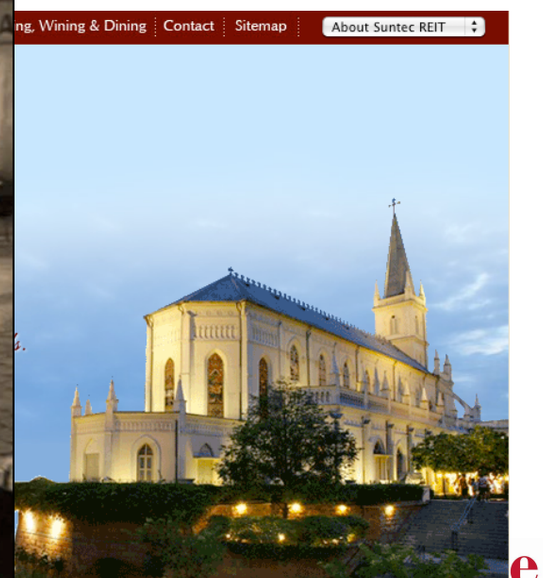






Grouping objects





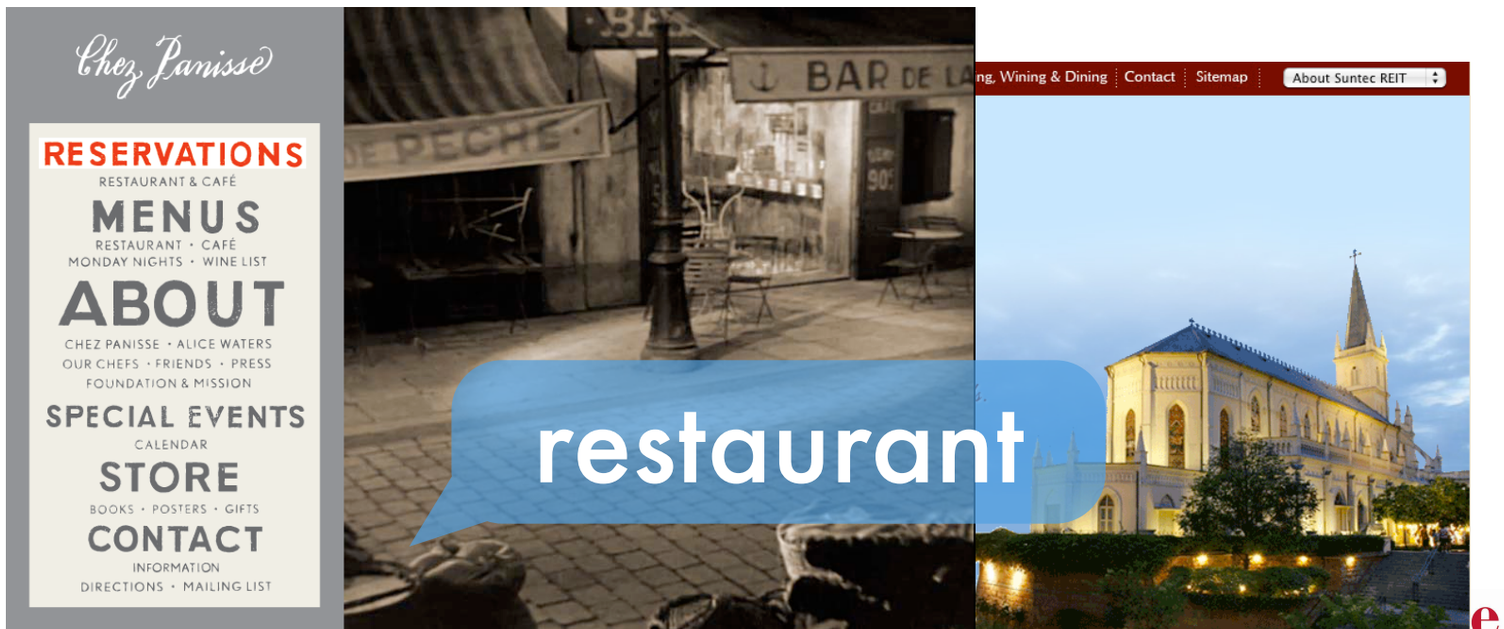
Google

e Mellon University

Grouping objects

airline

university

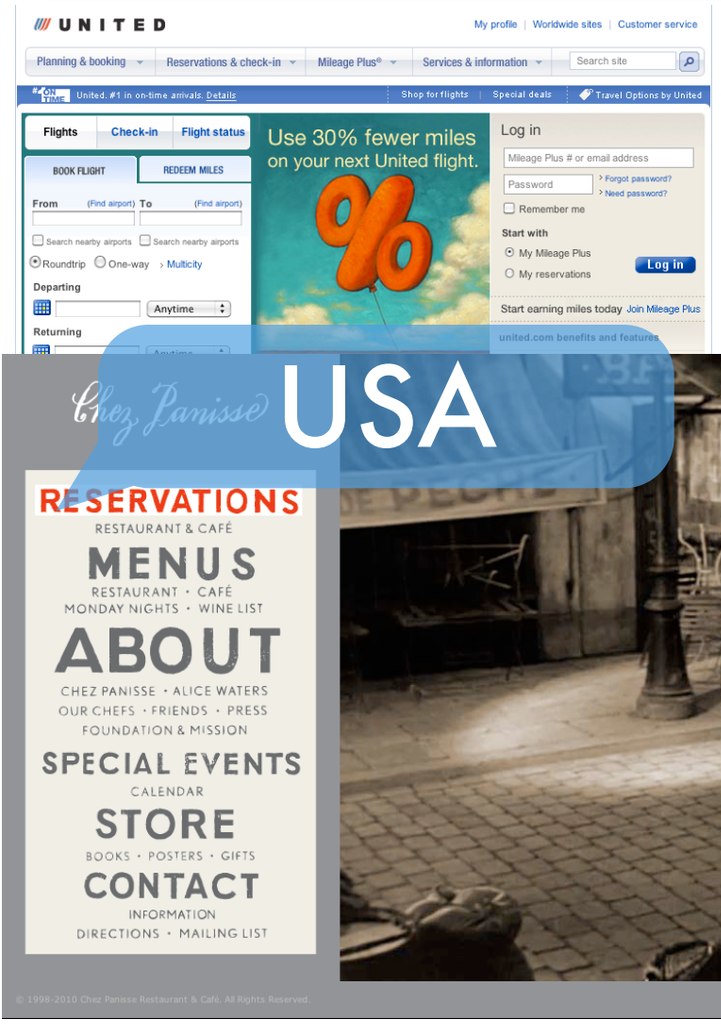


restaurant

Google

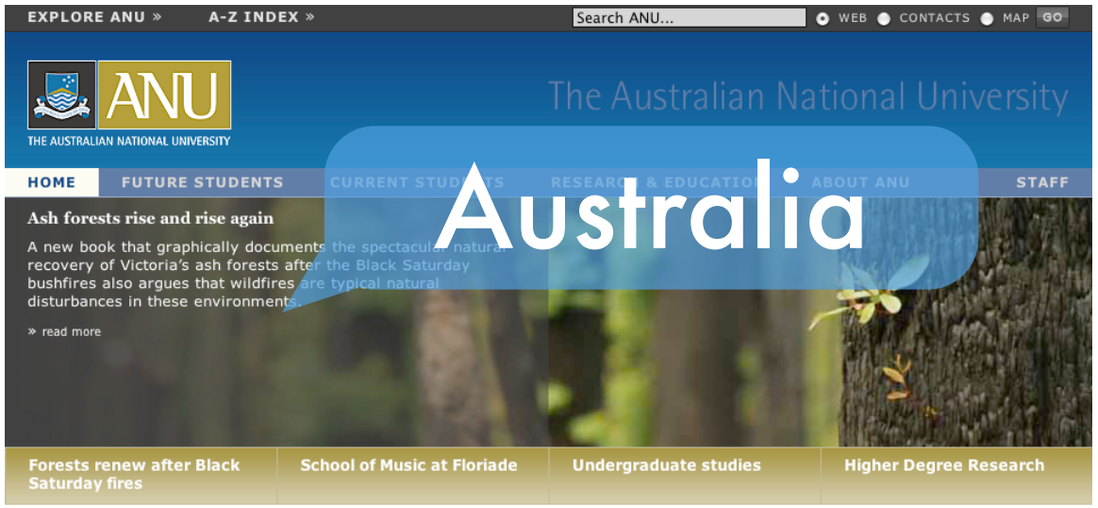
e Mellon University

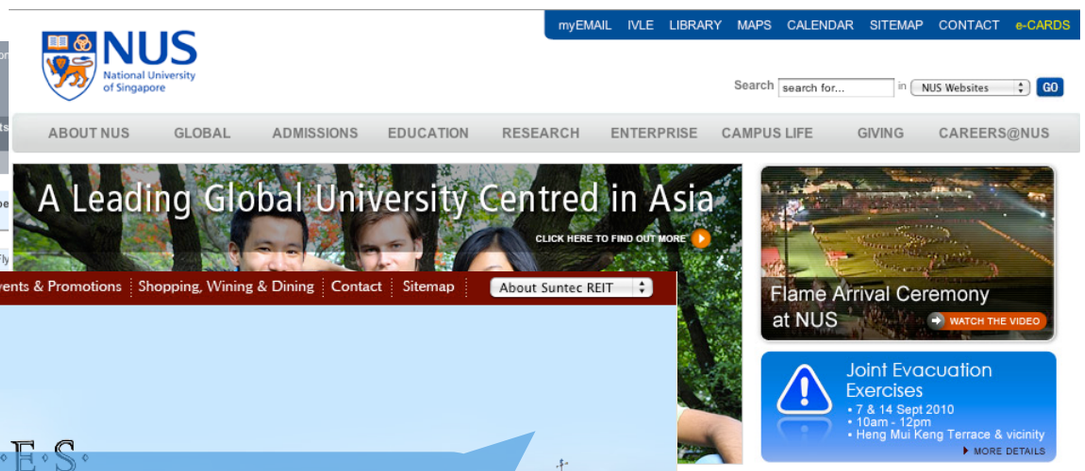
Grouping objects



USA

Australia





Singapore

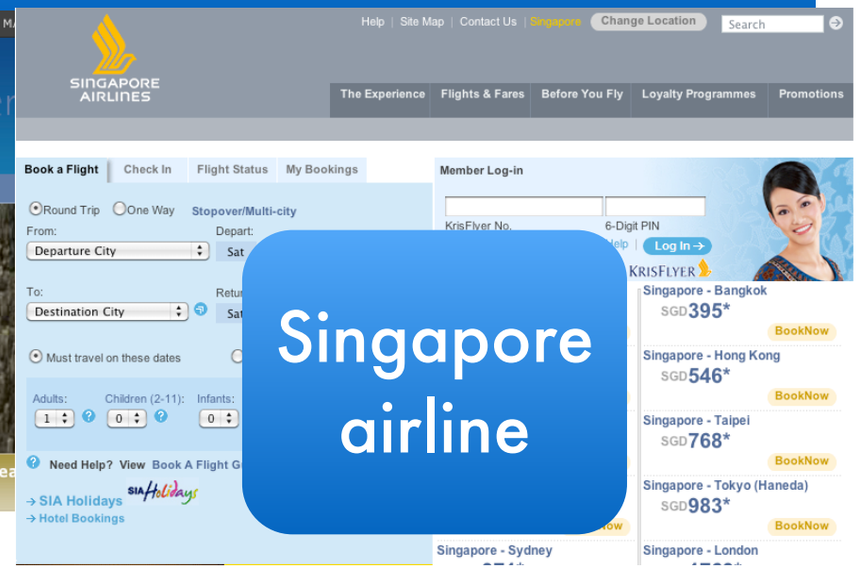
Google

ellon University

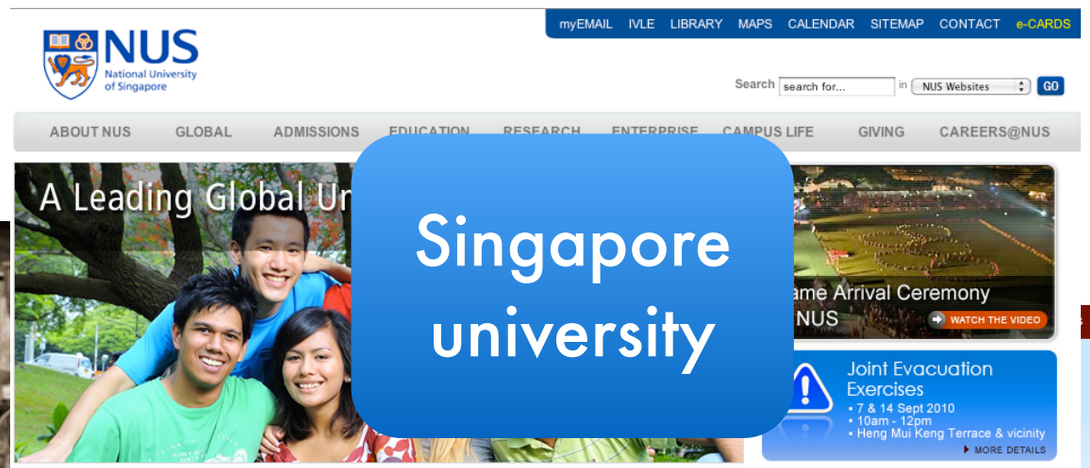
Topic Models

USA airline

Australia university



Singapore airline



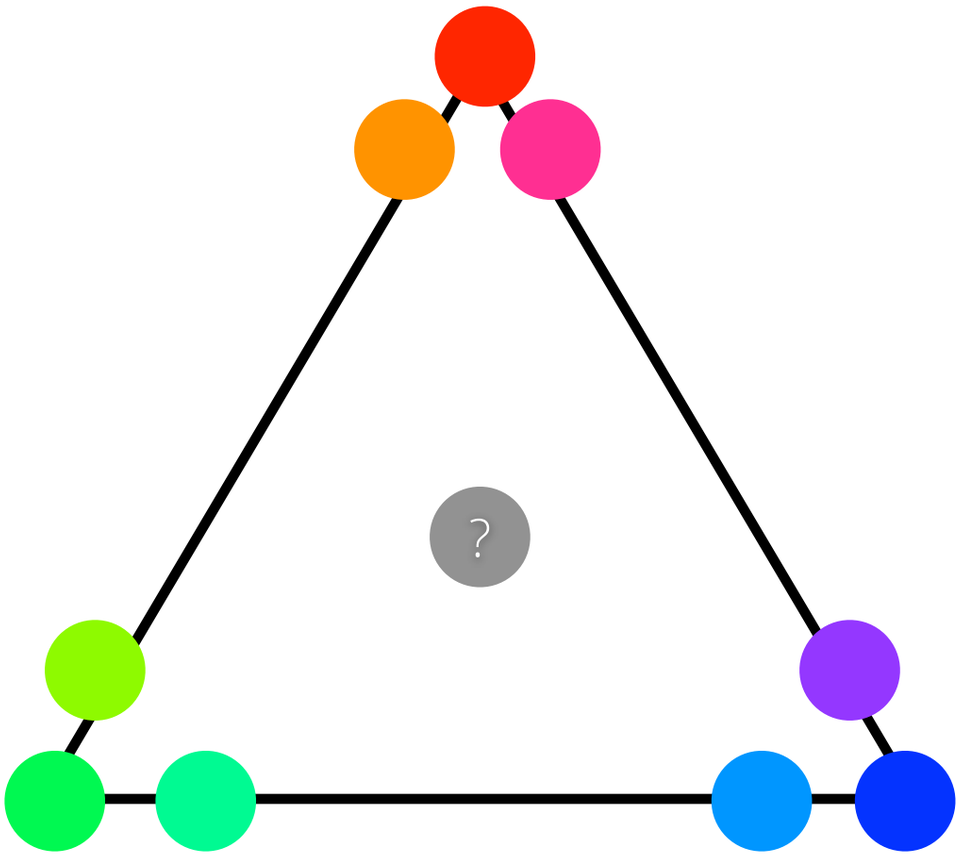
Singapore university

USA food

Singapore food

Clustering & Topic Models

Clustering



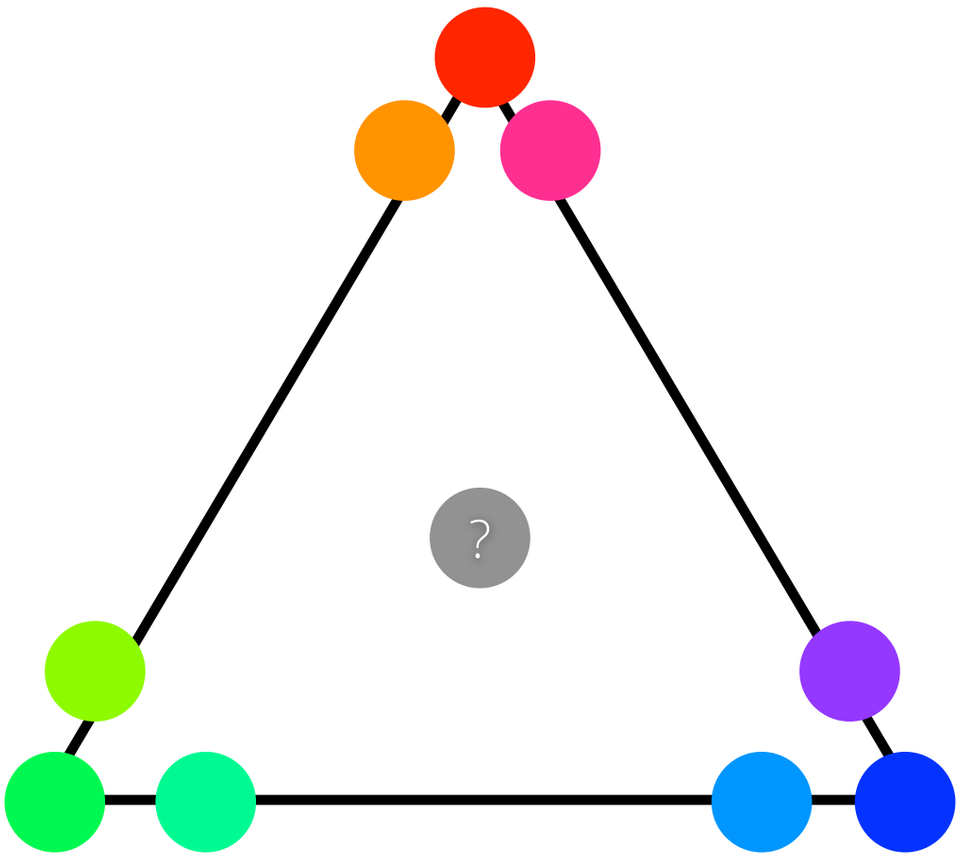
group objects
by prototypes

Google

Carnegie Mellon University

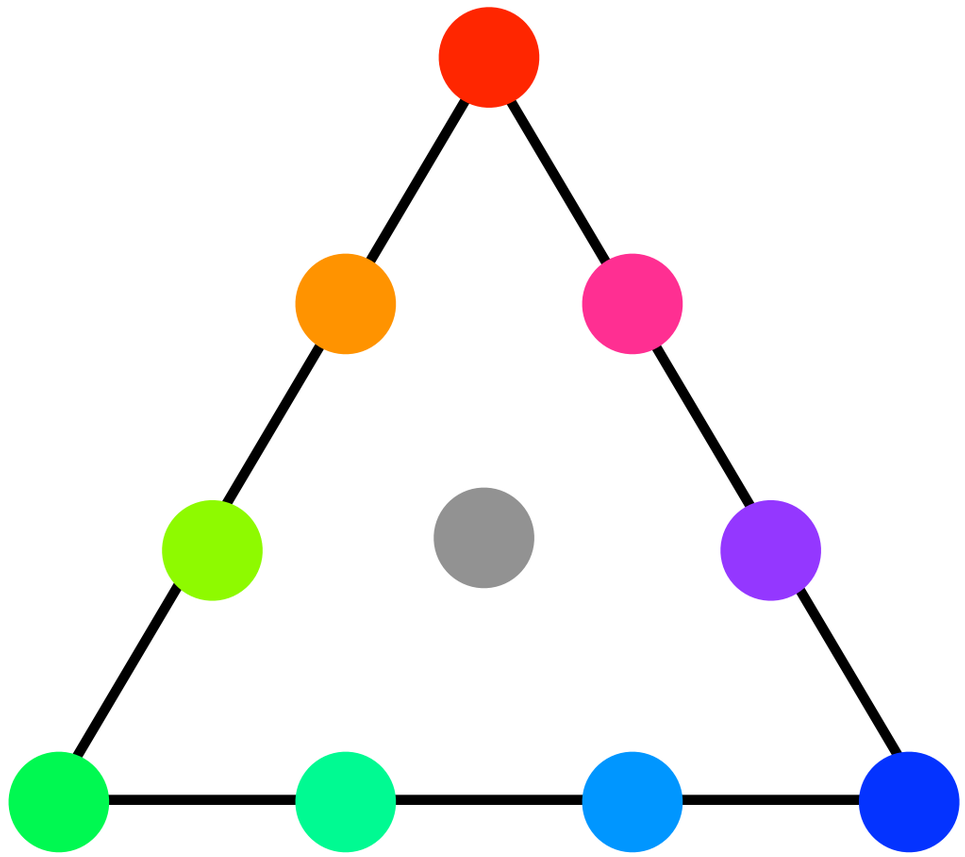
Clustering & Topic Models

Clustering



group objects
by prototypes

Topics



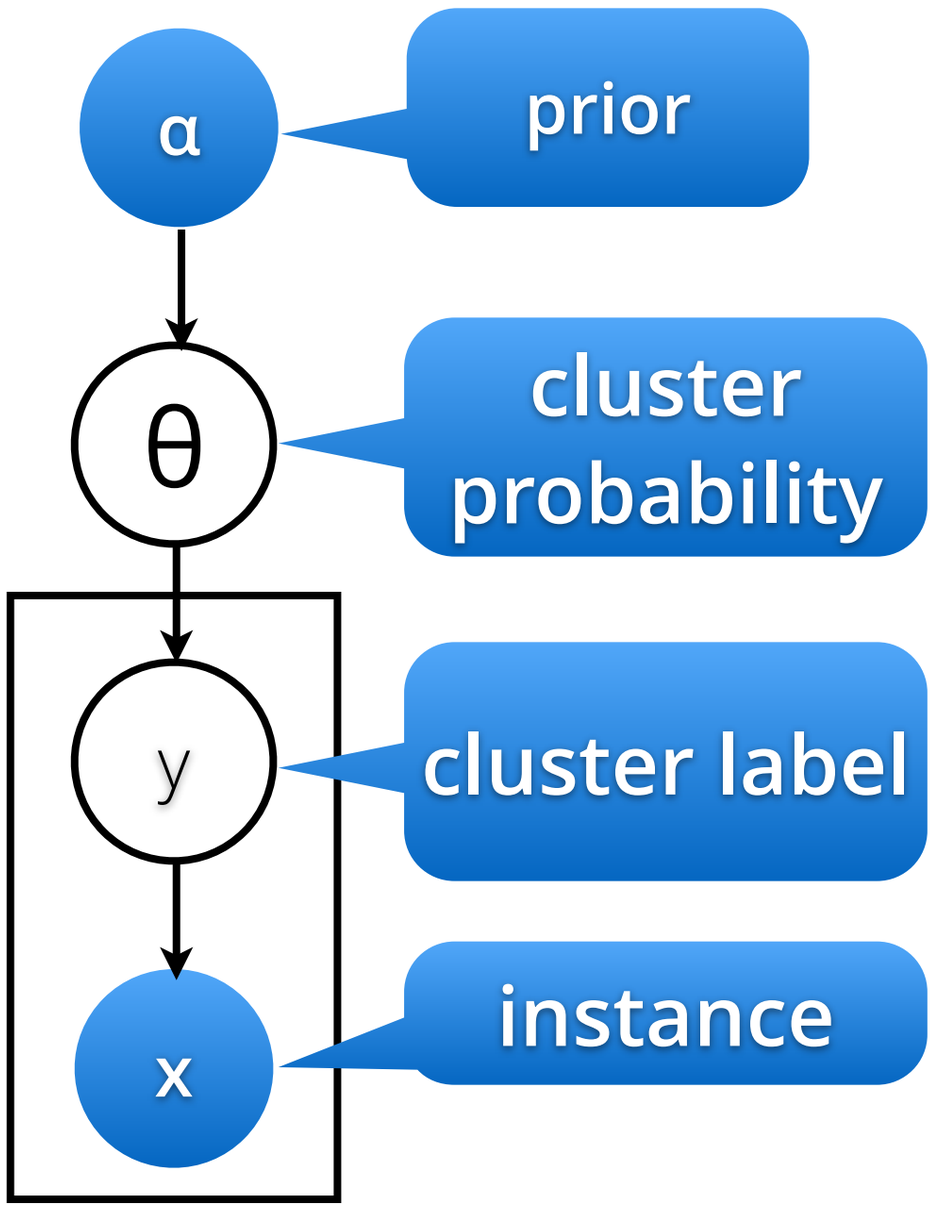
decompose objects
into prototypes

Google

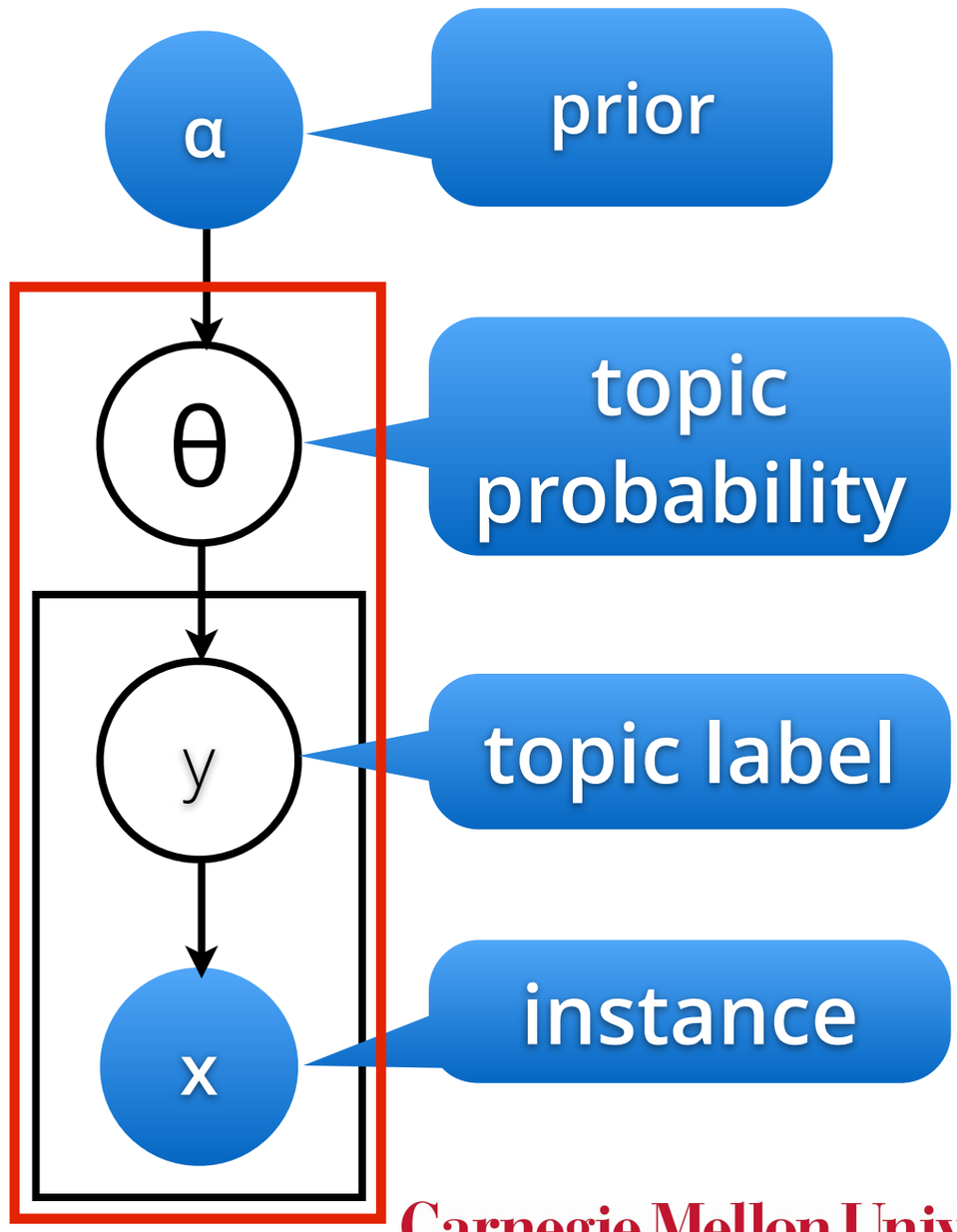
Carnegie Mellon University

Clustering & Topic Models

clustering



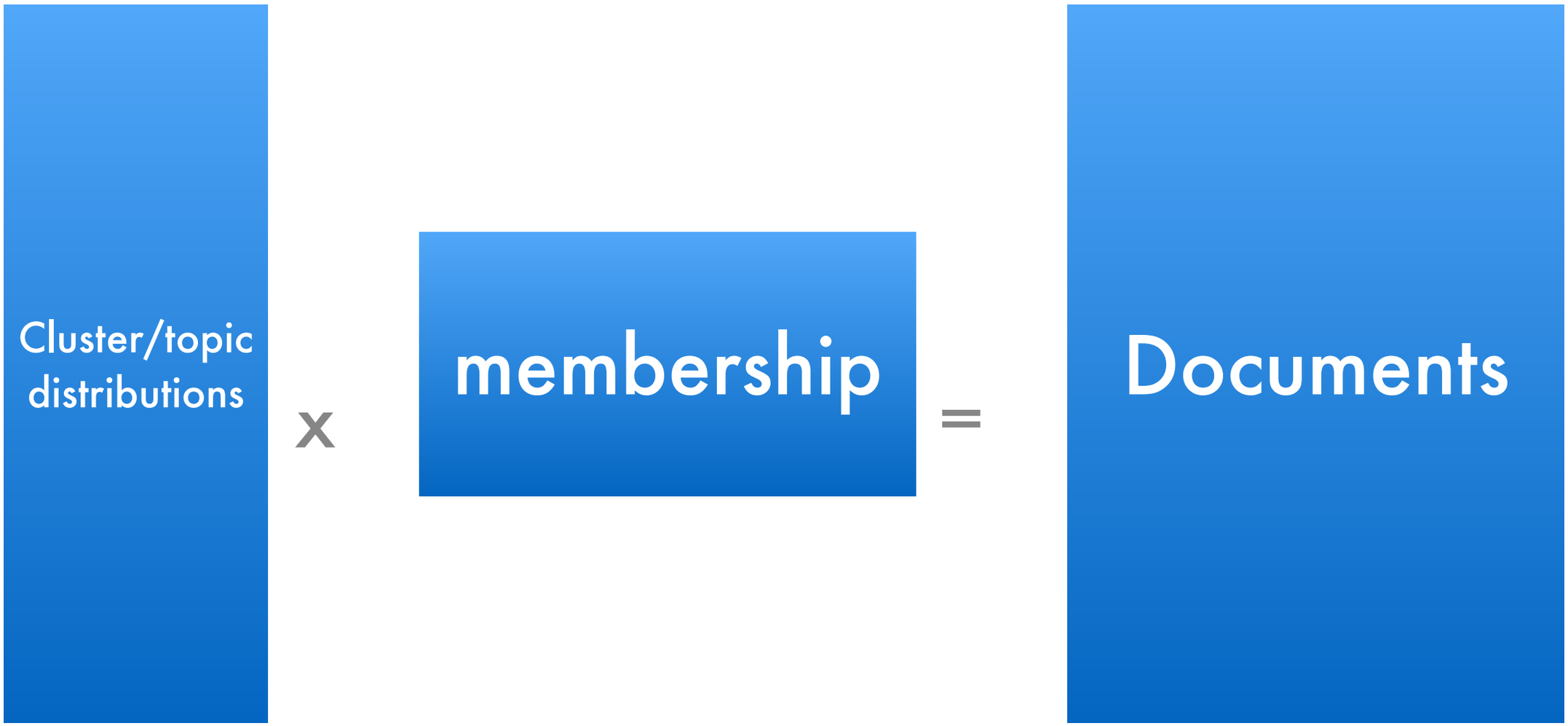
Latent Dirichlet Allocation



Google

Carnegie Mellon University

Clustering & Topic Models



clustering: (0, 1) matrix

topic model: stochastic matrix

LSI: arbitrary matrices

Google

Carnegie Mellon University

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

Google

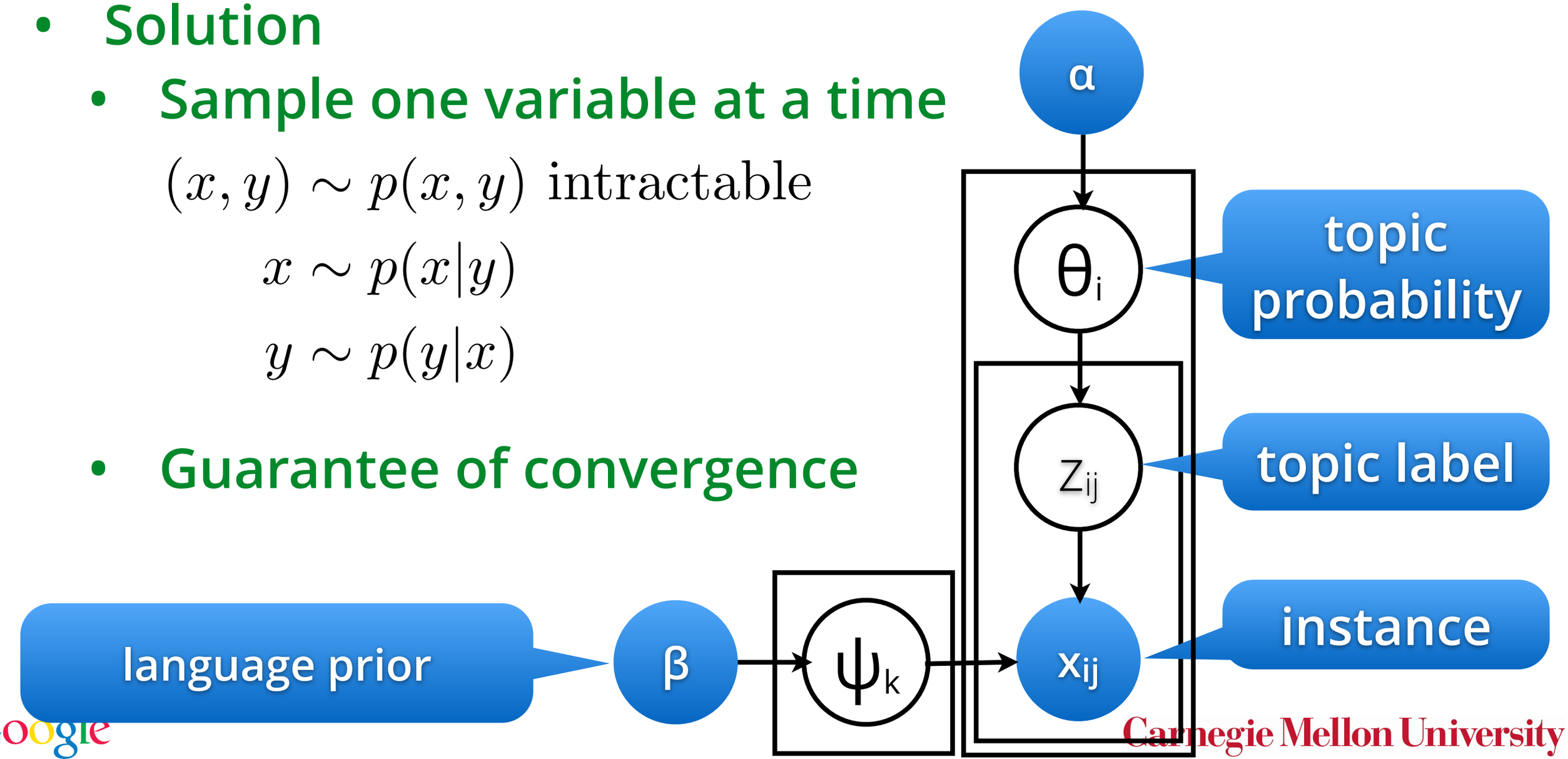
Carnegie Mellon University

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

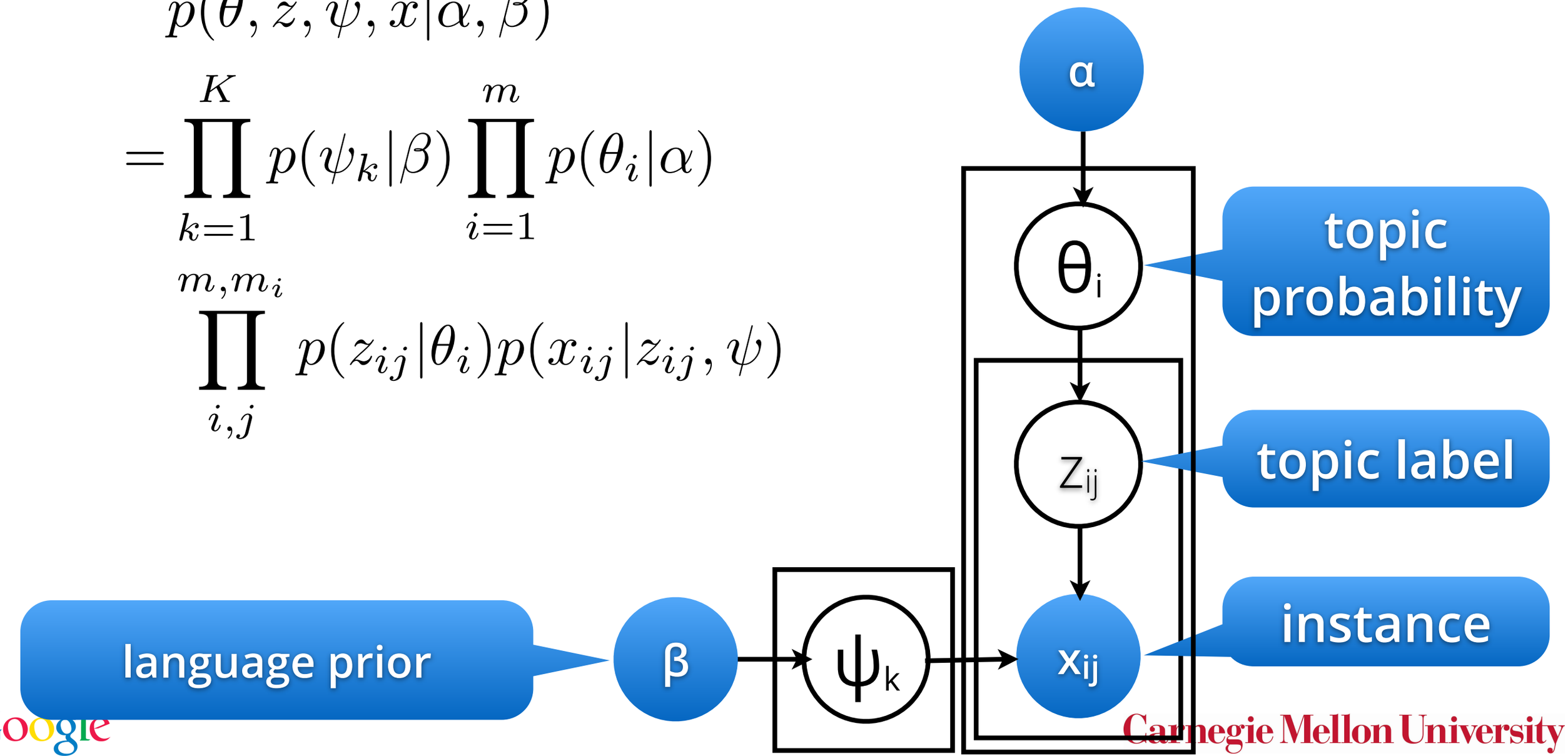


Joint Probability Distribution

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

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

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



Joint Probability Distribution

sample Ψ independently

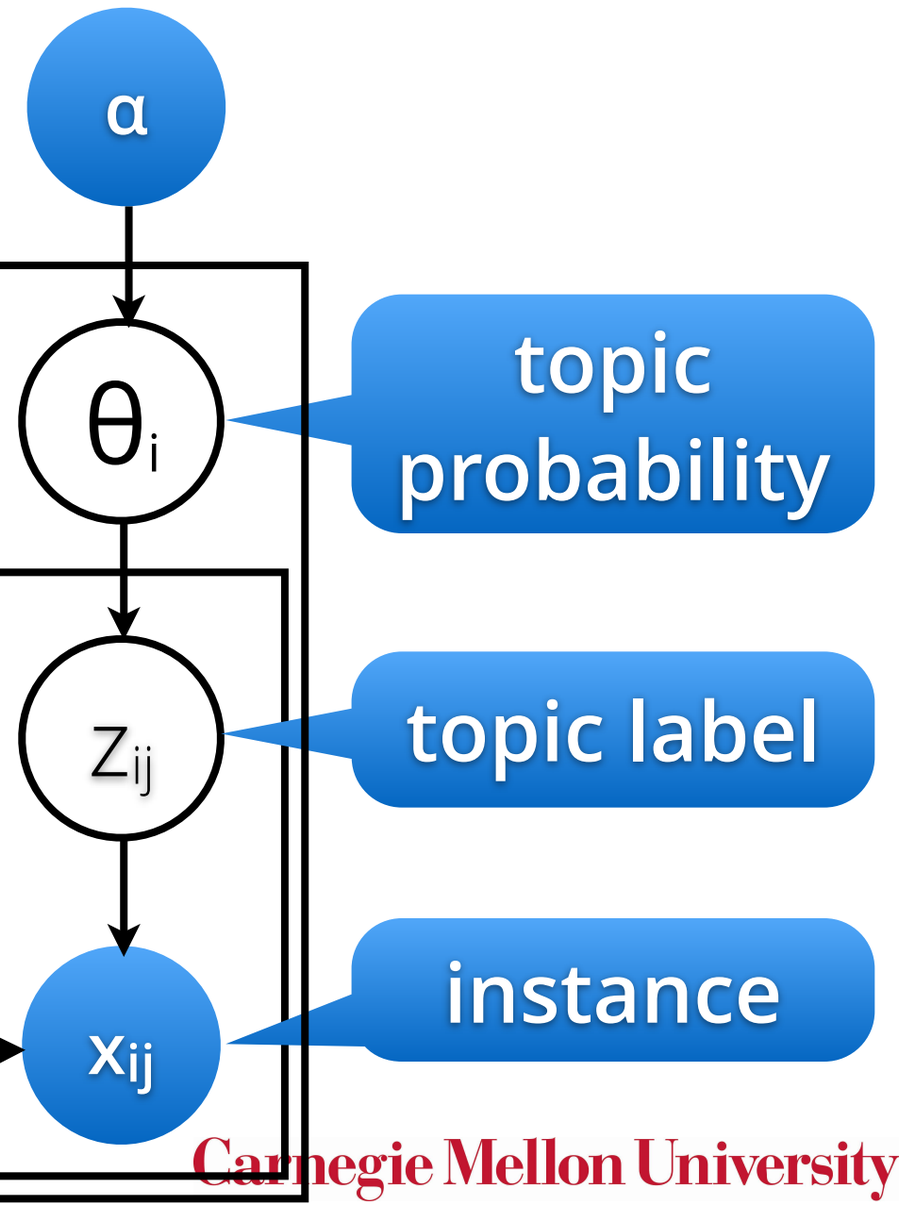
sample θ independently

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

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

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

sample z independently



Joint Probability Distribution

sample Ψ independently

sample θ independently

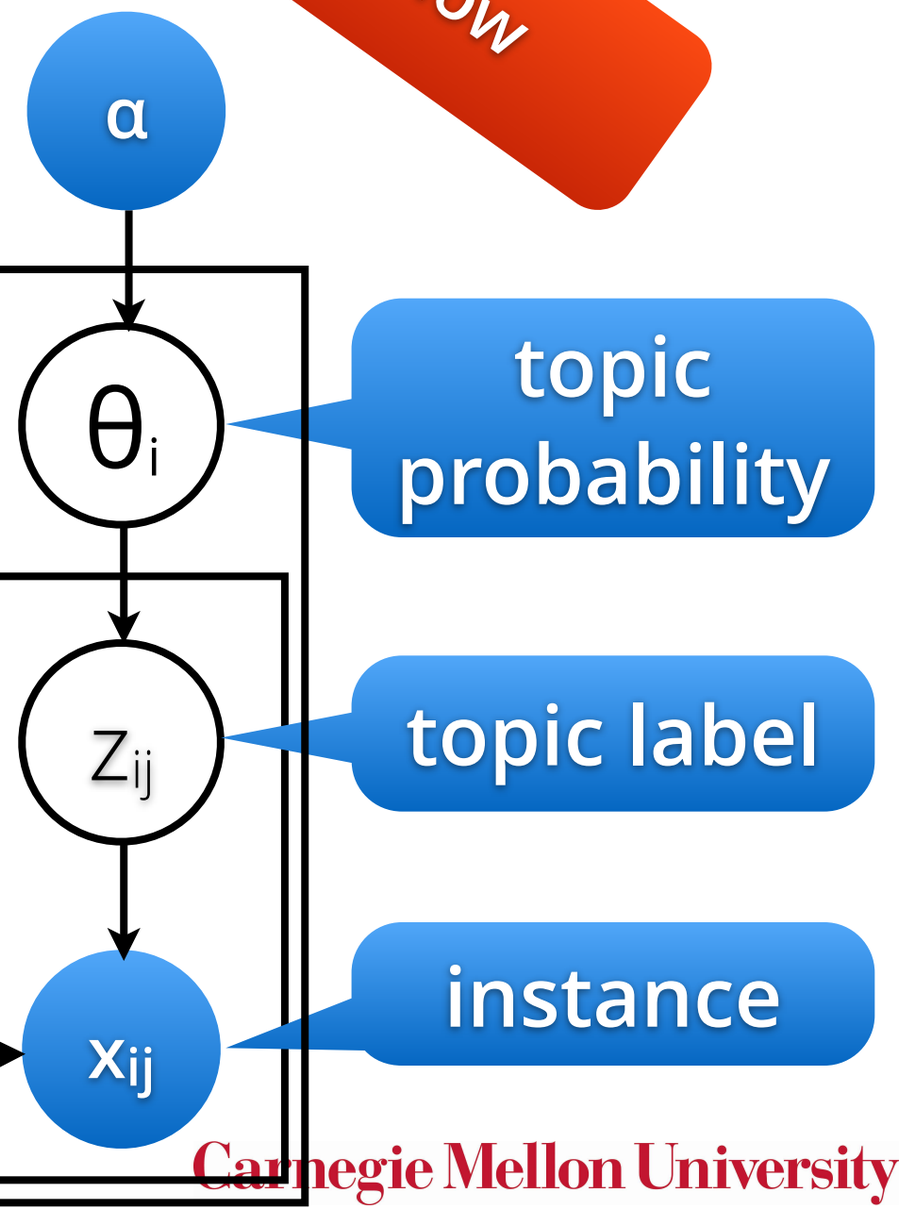
$$p(\theta, z, \psi, x|\alpha, \beta)$$

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

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

sample z independently

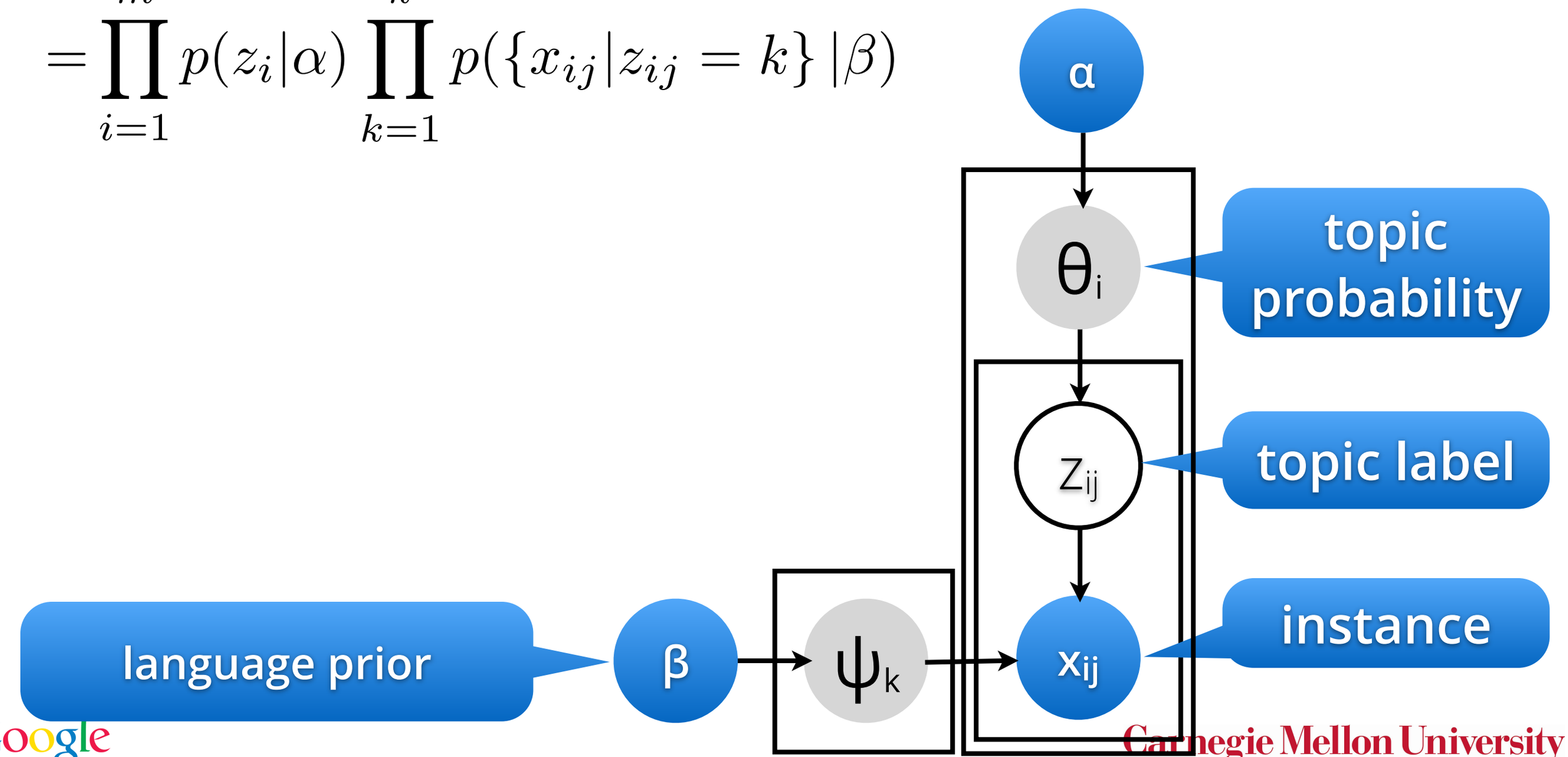
slow



Collapsed Sampler

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

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

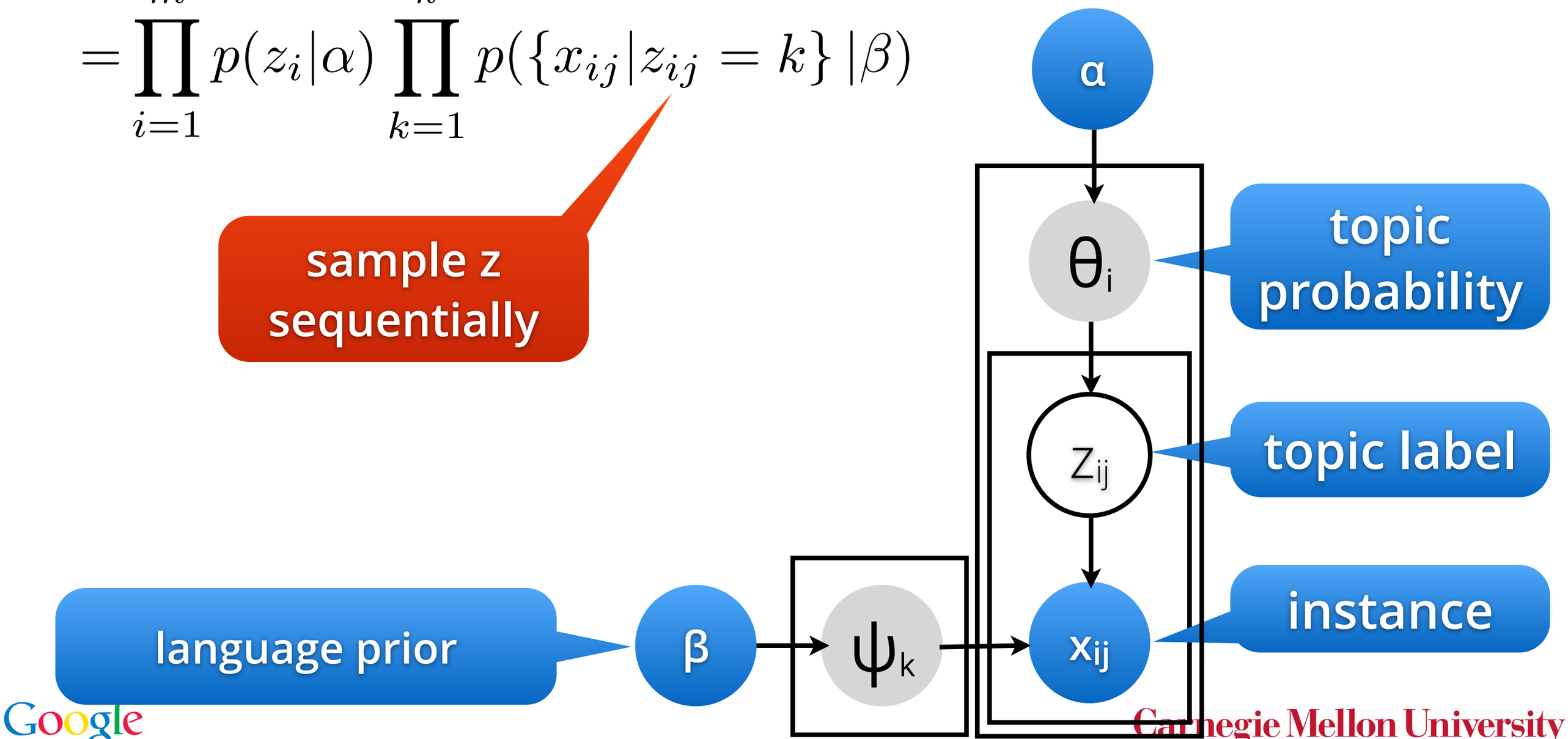


Collapsed Sampler

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

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

sample z sequentially



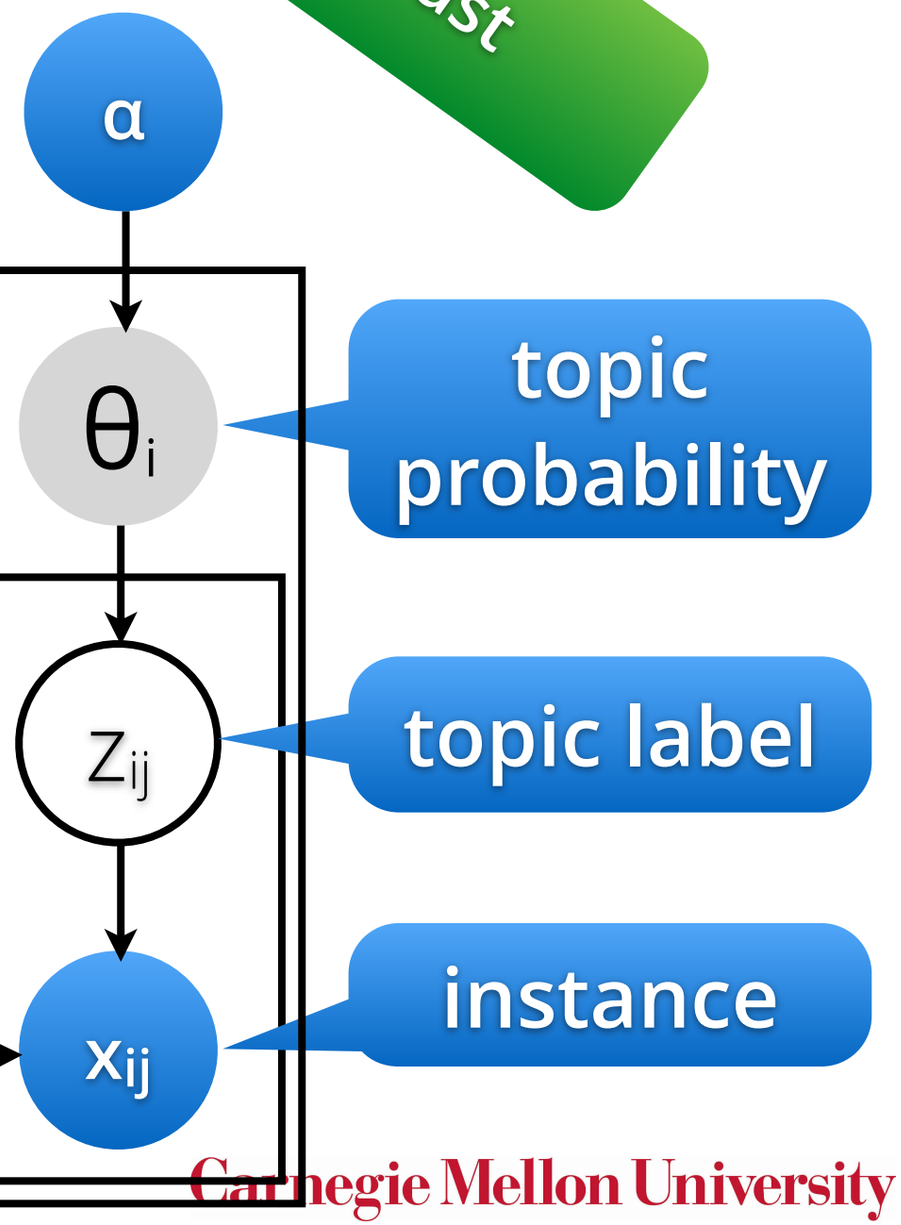
Collapsed Sampler

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

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

sample z sequentially

fast



Collapsed Sampler

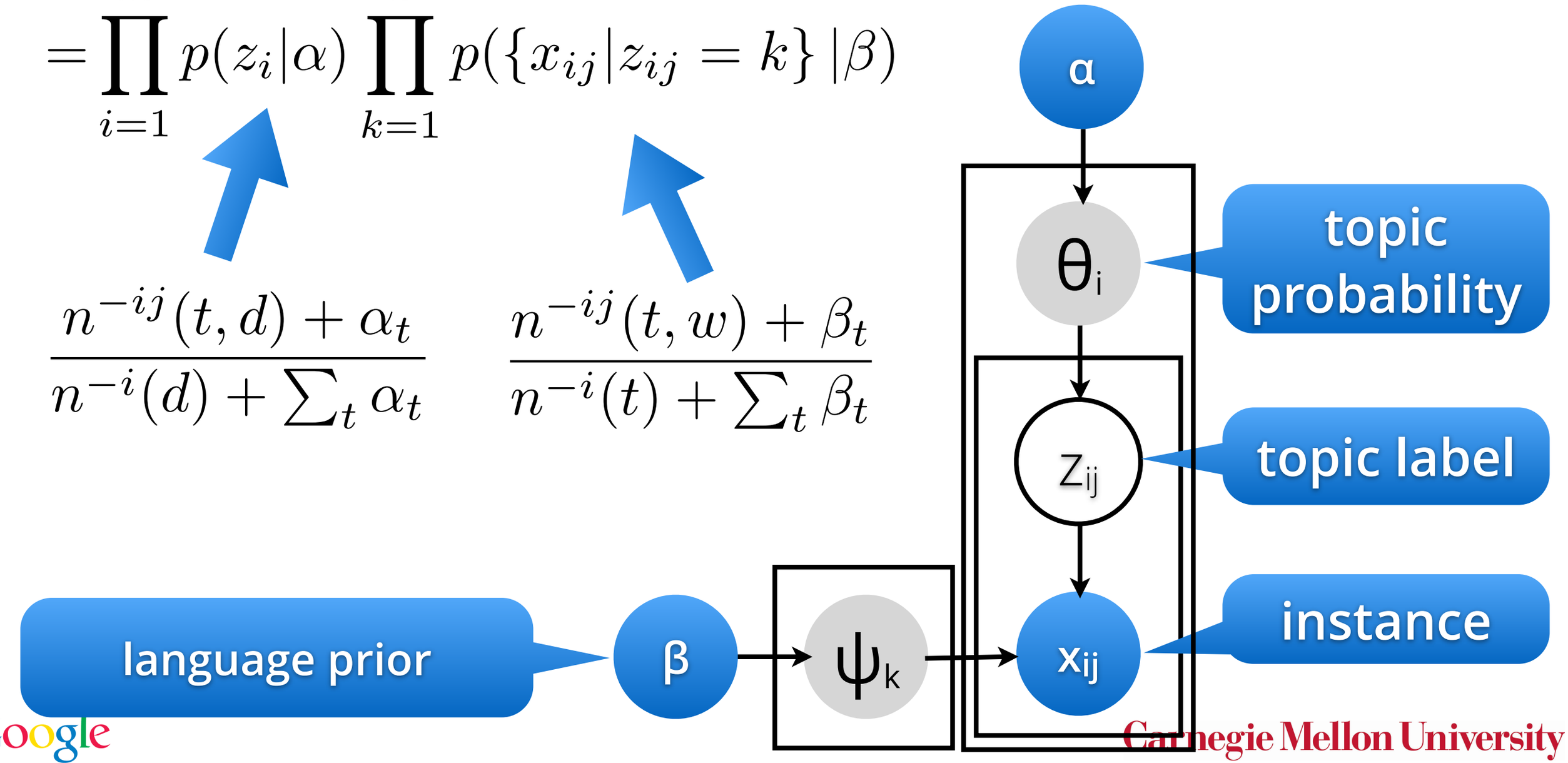
Griffiths & Steyvers, 2005

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

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

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

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



Collapsed Sampler

Griffiths & Steyvers, 2005

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

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

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

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

fast

α

θ_i

topic probability

z_{ij}

topic label

x_{ij}

instance

ψ_k

β

language prior

Google

Carnegie Mellon University

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

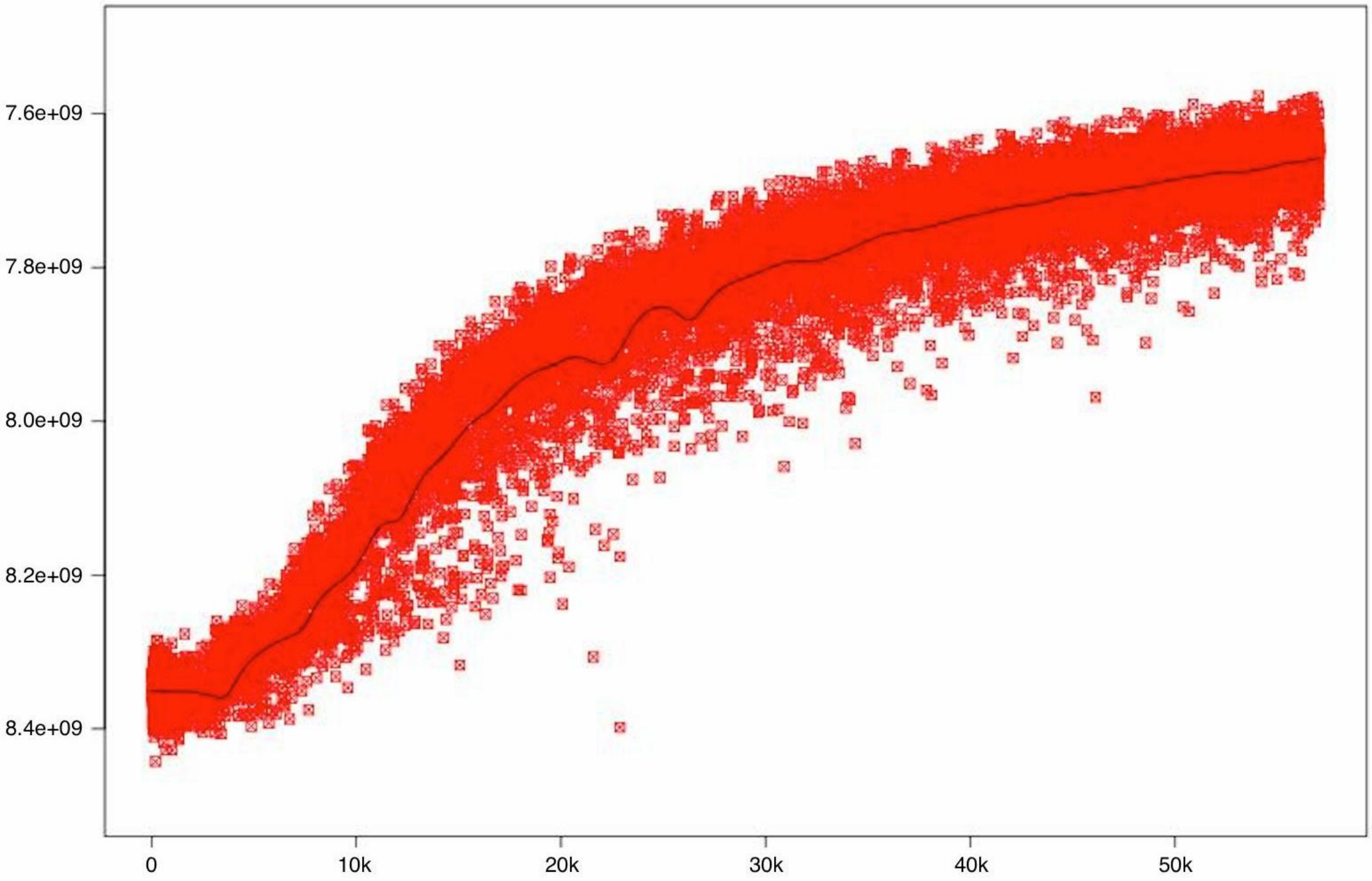
Gibbs Sampler

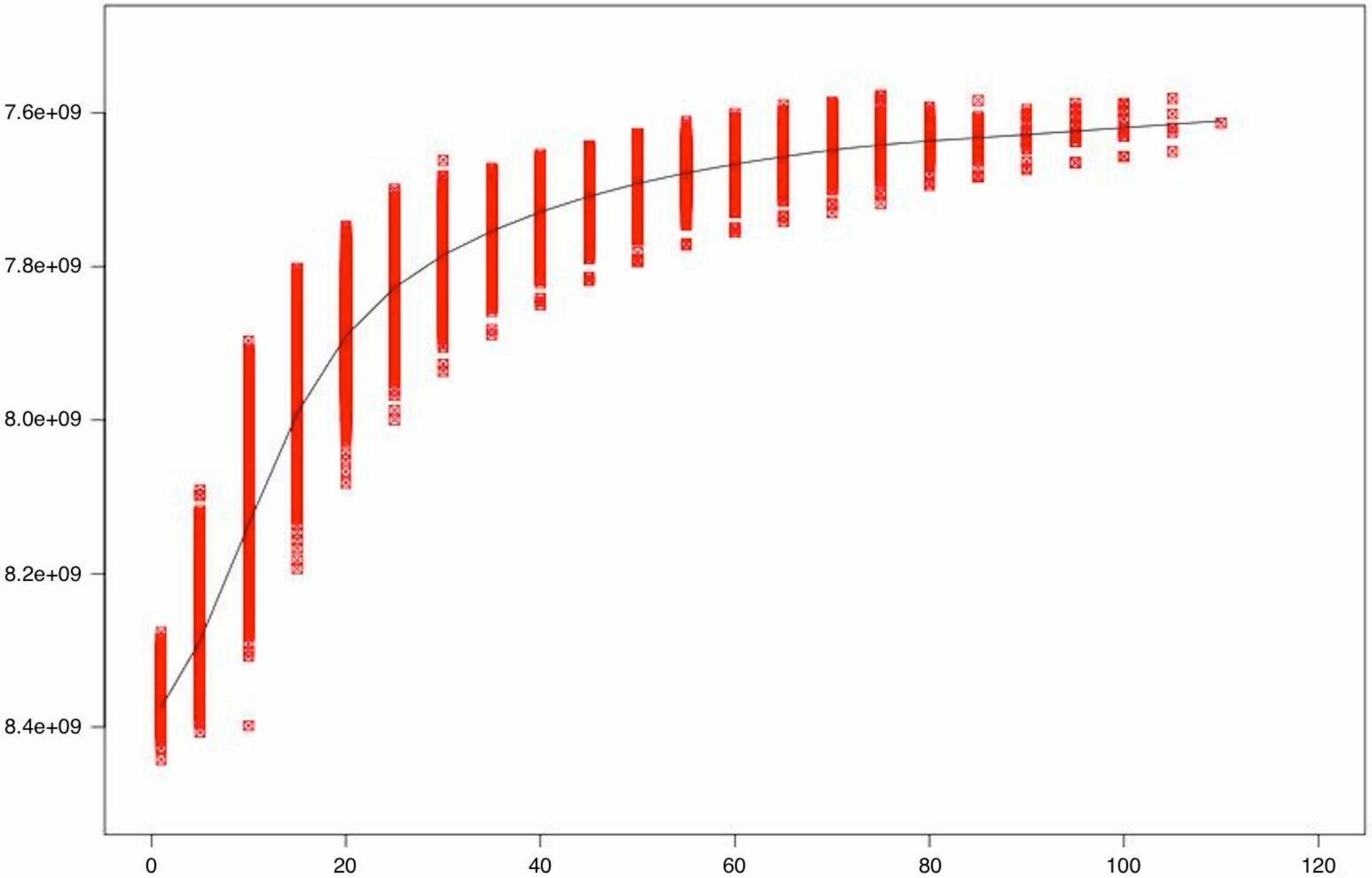
- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Lock local (word,topic) table
      - Update local (document, topic) table
      - Update local (word,topic) table
      - Unlock local (word,topic) table
  - Synchronize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

this kills multithreading

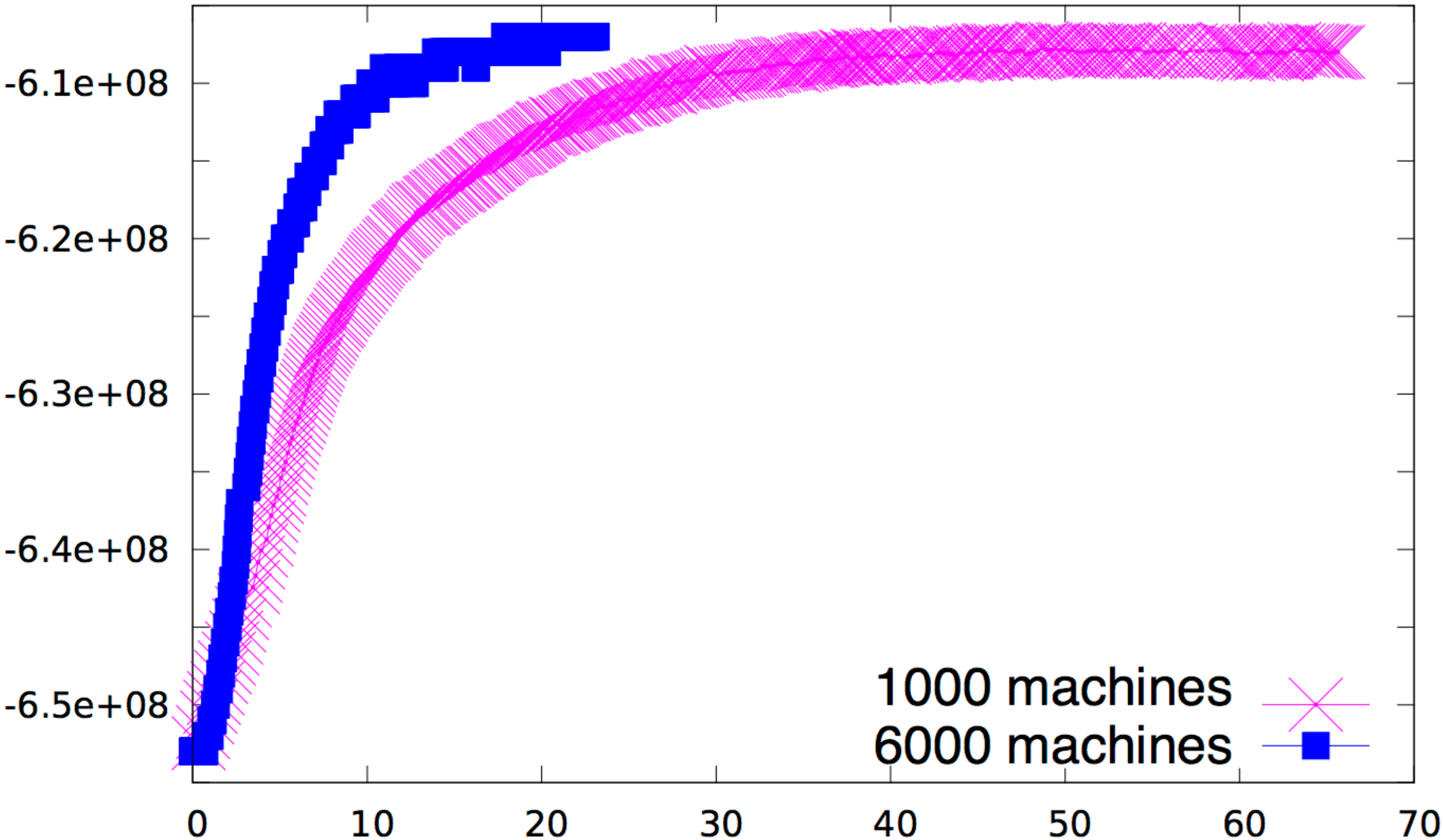
Gibbs Sampler

- 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


7.6e+09
7.8e+09
8.0e+09
8.2e+09
8.4e+09
0
10k
20k
30k
40k
50k
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

Techie

Mike

25-34 single male living with friends

Mike works as a graphic designer in a small agency and one day wants to run his own agency.

He's got an iPhone and a Vodafone 360 H1 by Samsung phone, one personal and one for work. He follows friends and key people in the design industry via Twitter, blogs, and RSS feeds. He uses his iPhone for work emails and his H1 for Facebook.

He uses Twitter to post updates about what he's up to with his project work as well as using it as a tool to find out what people are up to and to invite them to events. He uses Facebook to share personal photos and video and keeps a Tumblr blog to post interesting things he discovers and share them with his friends and followers.

Socialite

Zoë

18-33 single female living with friends

Zoë is studying a Masters in International Development unsure of what the future lies ahead of her.

She is constantly using the Facebook app on her Vodafone 360 M1 by Samsung phone as well as on her PC to upload and tag photos and videos from places she's been to with her friends, as well as to find out and comment on who's been where at which club nights and parties.

She regularly texts and messages her friends to find out if they've heard about a new pop-up shop she heard about via a flyer, or one-off warehouse party started by friends of friends.

Cost-conscious

Geoff

35-49 married male with young kids

Geoff works as an senior architect in a large practice, and has a wife and a young girl and 6-month baby boy. He thinks the time is right to start looking for a bigger home for his family.

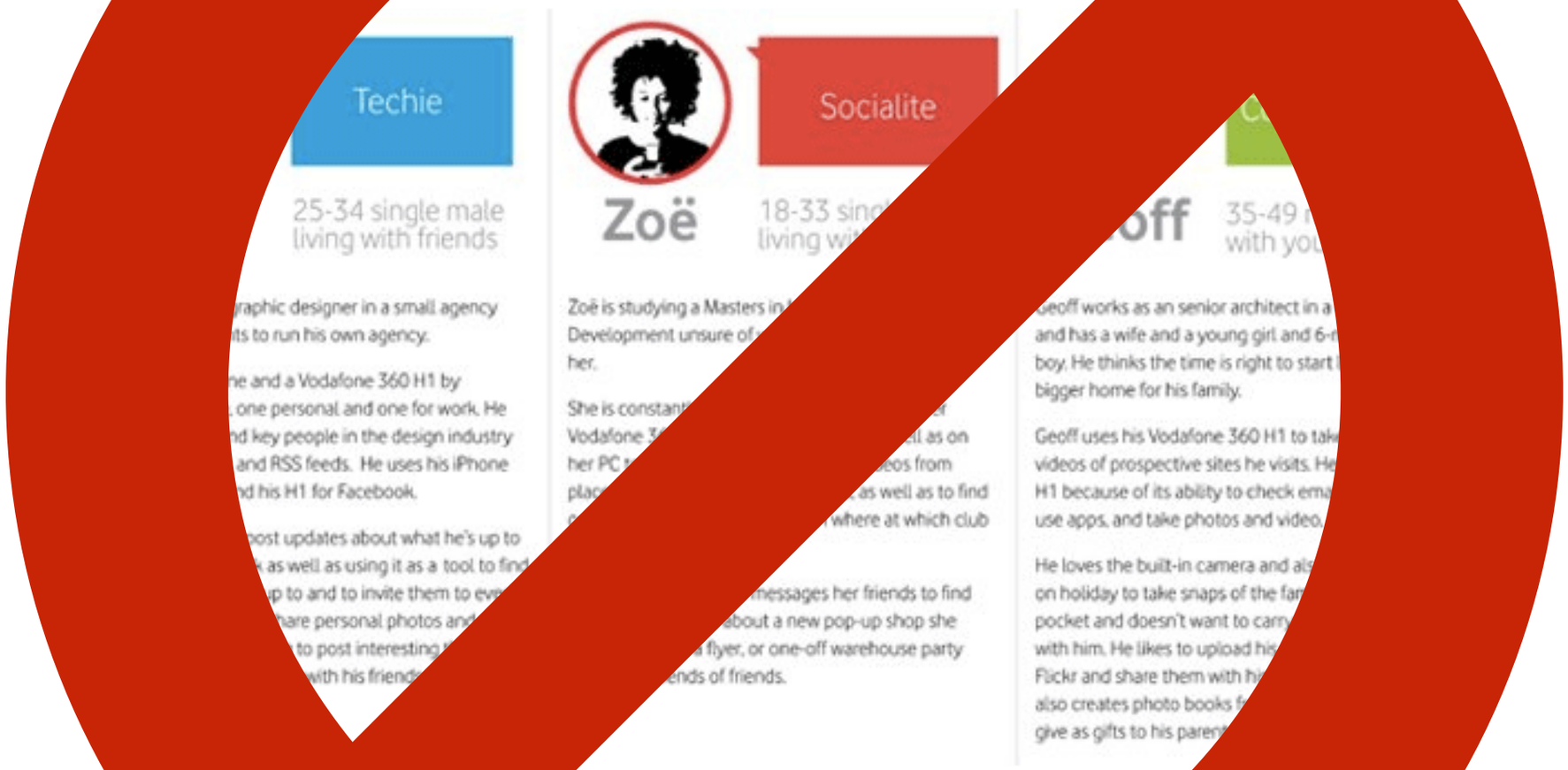
Geoff uses his Vodafone 360 H1 to take photos and videos of prospective sites he visits. He purchased the H1 because of its ability to check email, surf the web, use apps, and take photos and video.

He loves the built-in camera and also uses this phone on holiday to take snaps of the family as it fits in his pocket and doesn't want to carry a large SLR around with him. He likes to upload his photos and video to Flickr and share them with his family and friends. He also creates photo books from his holidays snaps to give as gifts to his parents.

User Profiling

Google

Carnegie Mellon University



User Profiling

Google

Carnegie Mellon University

Buying a Camera

Google

Carnegie Mellon University

Buying a Camera

Sponsor Results

YAHOO!

Web Images Video Local Shopping News More

panasonic lx5

Search In: the Web pages in English, French, German, Italian and Spanish

Also try: panasonic lz5, more...

Sponsored Results

Panasonic LX5 Cheap
Best Value for **Panasonic LX5**. Find NexTag Sellers
www.**NexTag.com**

Panasonic Lumix DMC-**LX5** Review (white
$434.00 as of Oct 17, 2010 Despite its shortcomings the **Panasonic** Lumix DMC- **LX5** delivers an exceller fastest in its class ...
reviews.cnet.com/digital-cameras/**panasonic**-lumix-
this site

Panasonic LX5 | Get The Lowest Price On
Panasonic LX5 with 14.1MP captures enough detail. **Panasonic LX5** Camera
www.**panasoniclx5.com** - Cached - More from this s

Panasonic Lumix DMC-**LX5** White Digital (
shopping.yahoo.com
The Panasonic Lumix DMC-LX5 is a compact digital photo enthusiasts the ideal way for capturing profess photos and High De...

Price: **$434 to $513.99**

Reviews | Price & Details | Specs

amazon.com Prime

Hello, Alexander Smola. We have recommendations for you. (Not Alexander?)

Alexander's Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards

Shop All Departments | Search Electronics | GO

Camera & Photo | All Electronics | Brands | Bestsellers | Digital SLRs & Lenses | Point-And-Shoots | Camcorders | Pro

Instant Order Update for Alexander Smola. You purchased this item on October 6, 2010. View t

Color: **Black**

Prime

Member: Alexander Smola

Alexander Smola: This item is eligible for Amazon Prime. Click here to turn on 1-Click and make Prime even better for you. (With 1-Click enabled, you can always use the regular shopping cart as well.)

Panasonic Lumix DMC-LX5 10.1 MP Digital Camera with 3.8x Optical Image Stabilized Zoom and 3.0-Inch LCD (Black)

by Panasonic

(40 customer reviews)

List Price: $499.00

Price: $444.95 & eligible for free shipping with **Amazon Prime**

You Save: $54.05 (11%)

new

Color: Black

Sign

Expres

Alexa

What

time

Google

Carnegie Mellon University

Buying a Camera

Sponsor Results

YAHOO!

Web Images Video Local Shopping News More

panasonic lx5

Search In: the Web pages in English, French, German, Italian and Spanish

Also try: panasonic lz5, more...

Panasonic LX5 Cheap — Sponsored Results
Best Value for **Panasonic LX5**. Find NexTag Sellers
www.NexTag.com

Panasonic Lumix DMC-LX5 Review (white
$434.00 as of Oct 17, 2010 Despite its shortcomings the **Panasonic** Lumix DMC- **LX5** delivers an exceller fastest in its class ...
reviews.cnet.com/digital-cameras/panasonic-lumix-
this site

Panasonic LX5 | Get The Lowest Price On
Panasonic LX5 with 14.1MP captures enough detail. **Panasonic LX5** Camera
www.panasoniclx5.com - Cached - More from this s

Panasonic Lumix DMC-LX5 White Digital (
shopping.yahoo.com
The Panasonic Lumix DMC-LX5 is a compact digital photo enthusiasts the ideal way for capturing profess photos and High De...

Price: $434 to $513.99

Specs

amazon.com Prime

Hello, Alexander Smola. We have recommendations for you. (Not Alexander?)
Alexander's Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards

Shop All Departments | Search Electronics | GO

Camera & Photo | All Electronics | Brands | Bestsellers | Digital SLRs & Lenses | Point-And-Shoots | Camcorders | Pro

Instant Order Update for Alexander Smola. You purchased this item on October 6, 2010. View t
Color: Black

Prime
Member: Alexander Smola

Alexander Smola: This item is eligible for Amazon Prime. Click here to turn on 1-Click and make Prime even better for you. (With 1-Click enabled, you can always use the regular shopping cart as well.)

Panasonic Lumix DMC-LX5 10.1 MP Digital Camera with 3.8x Optical Image Stabilized Zoom and 3.0-Inch LCD (Black)
by Panasonic
(40 customer reviews)

List Price: $499.00
Price: $444.95 & eligible for free shipping with **Amazon Prime**
You Save: $54.05 (11%)

new
Color: Black

Sign i
Expres
Alexa
What

show ads now

time

Google

Carnegie Mellon University

Buying a Camera

Sponsor Results

YAHOO!

Web Images Video Local Shopping News More

panasonic lx5

Search In: the Web pages in English, French, German, Italian and Spanish

Also try: panasonic lz5, more...

Panasonic LX5 Cheap — Sponsored Results
Best Value for **Panasonic LX5**. Find NexTag Sellers
www.NexTag.com

Panasonic Lumix DMC-LX5 Review (white
$434.00 as of Oct 17, 2010 Despite its shortcomings the **Panasonic** Lumix DMC- **LX5** delivers an exceller fastest in its class ...
reviews.cnet.com/digital-cameras/panasonic-lumix-
this site

Panasonic LX5 | Get The Lowest Price On
Panasonic LX5 with 14.1MP captures enough detail. **Panasonic LX5** Camera
www.panasoniclx5.com - Cached - More from this s

Panasonic Lumix DMC-LX5 White Digital (
shopping.yahoo.com
The Panasonic Lumix DMC-LX5 is a compact digital photo enthusiasts the ideal way for capturing profess photos and High De...

Price: $434 to $513.99

Specs

show ads now

amazon.com Prime

Hello, Alexander Smola. We have recommendations for you. (Not Alexander?)
Alexander's Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards

Shop All Departments | Search Electronics | GO

Camera & Photo | All Electronics | Brands | Bestsellers | Digital SLRs & Lenses | Point-And-Shoots | Camcorders | Pro

Instant Order Update for Alexander Smola. You purchased this item on October 6, 2010. View t
Color: Black

Prime
Member: Alexander Smola

Alexander Smola: This item is eligible for Amazon Prime. Click here to turn on 1-Click and make Prime even better for you. (With 1-Click enabled, you can always use the regular shopping cart as well.)

Panasonic Lumix DMC-LX5 10.1 MP Digital Camera with 3.8x Optical Image Stabilized Zoom and 3.0-Inch LCD (Black)
by Panasonic
(40 customer reviews)

List Price: $499.00
Price: $444.95 & eligible for free shipping with Amazon Prime
You Save: $54.05 (11%)

Sign i
Expres
Alexa
What

too late

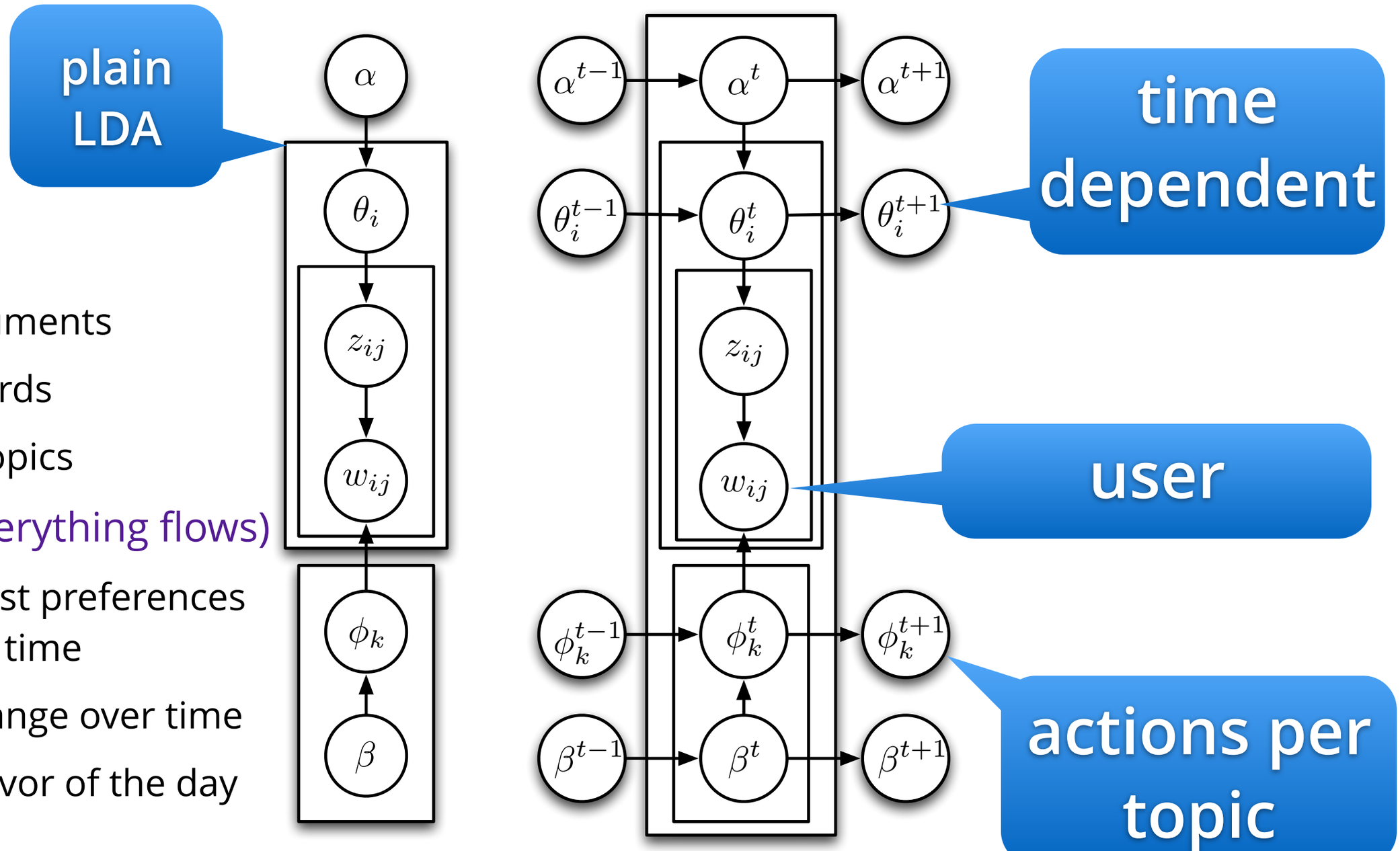
time

Google

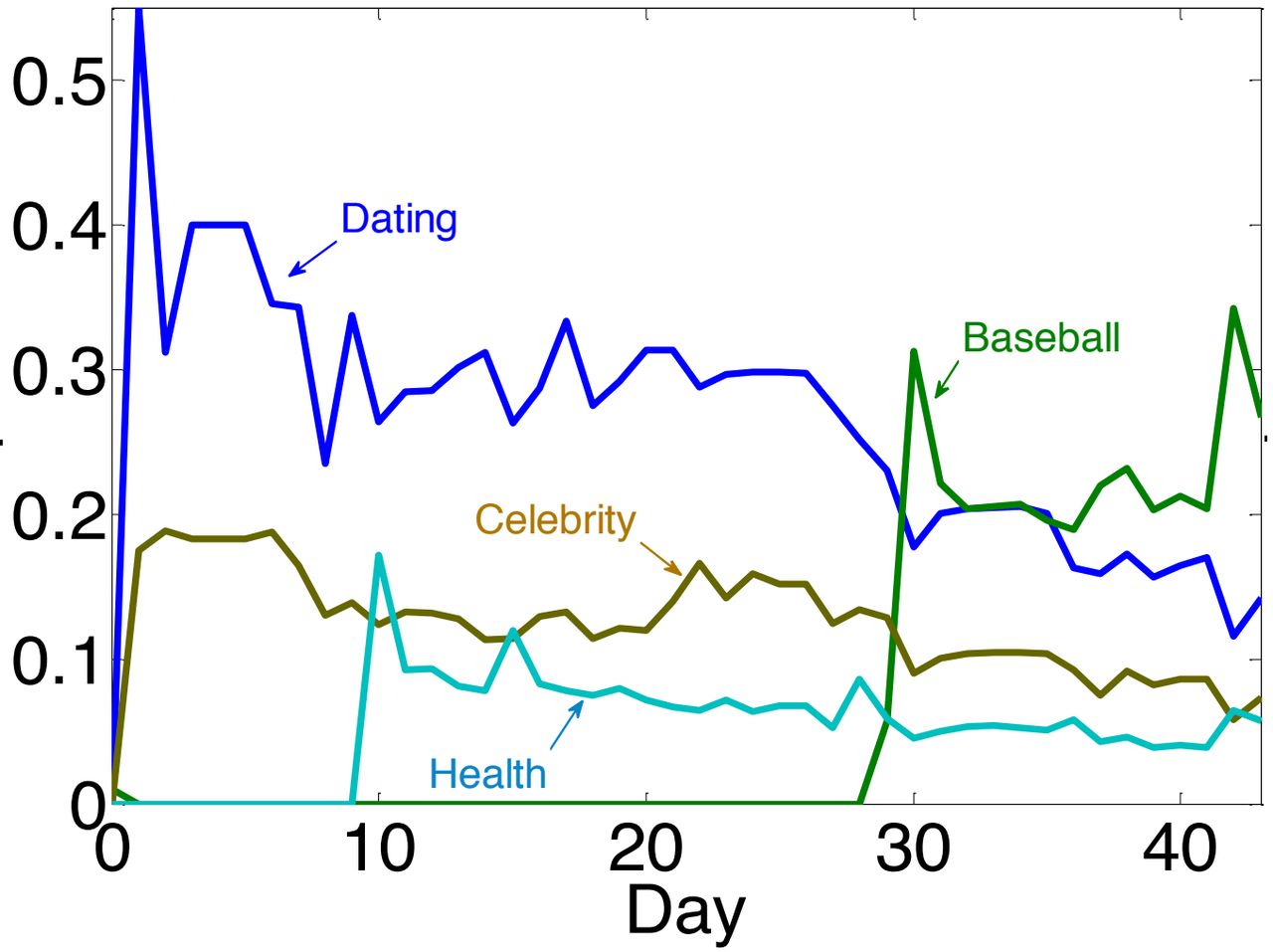
Carnegie Mellon University

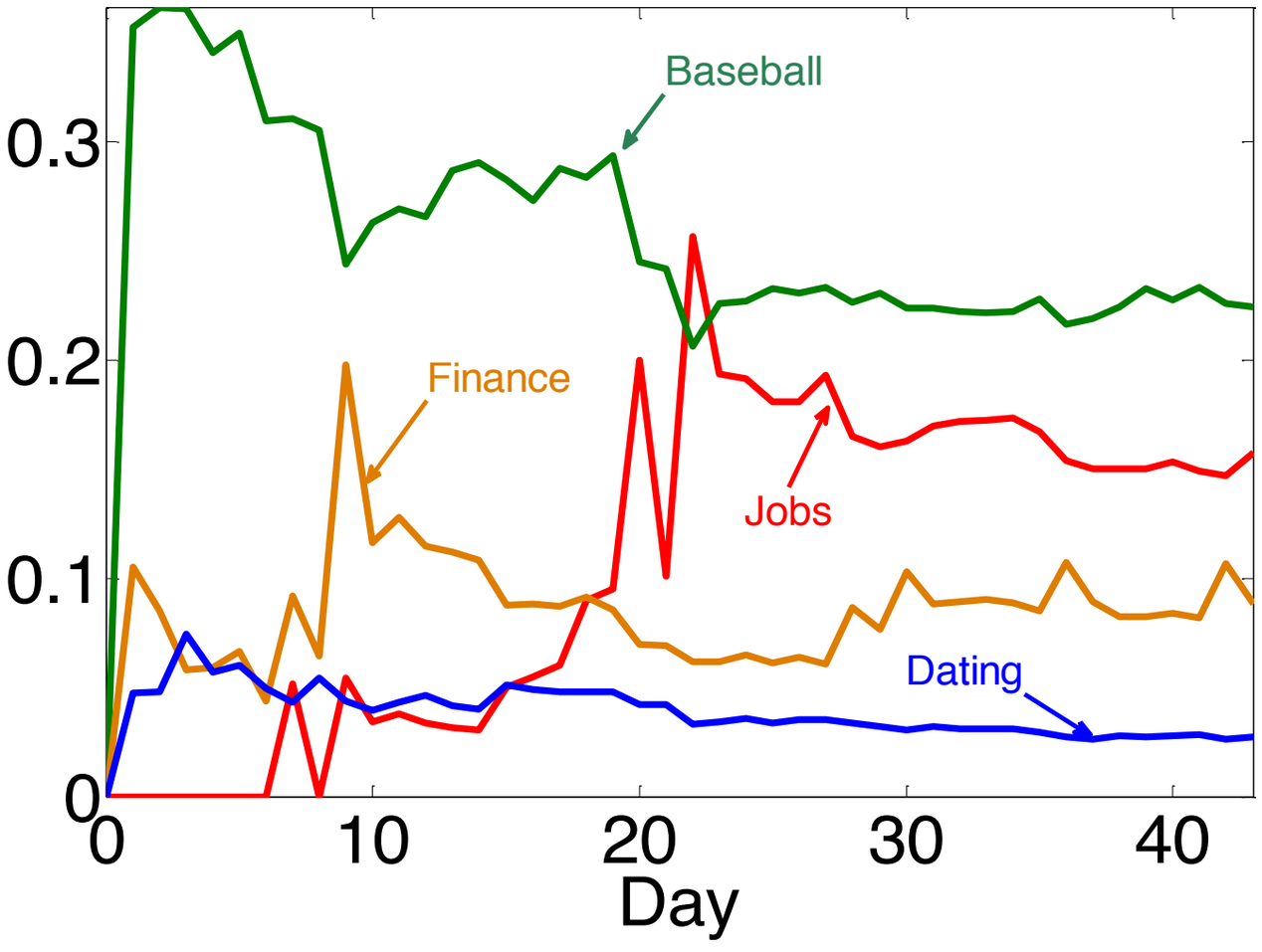
Statistical Model

- 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



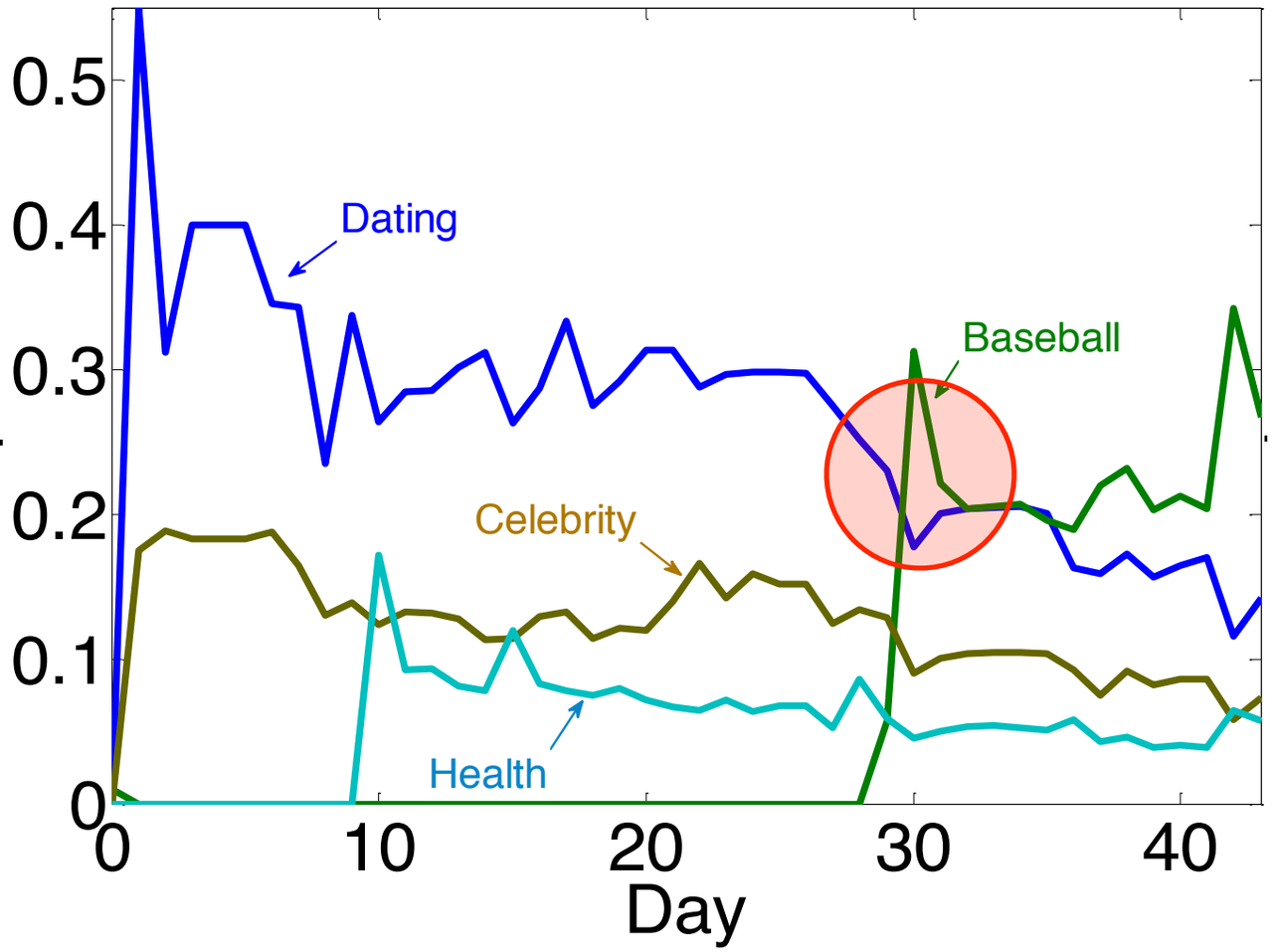
Some Users

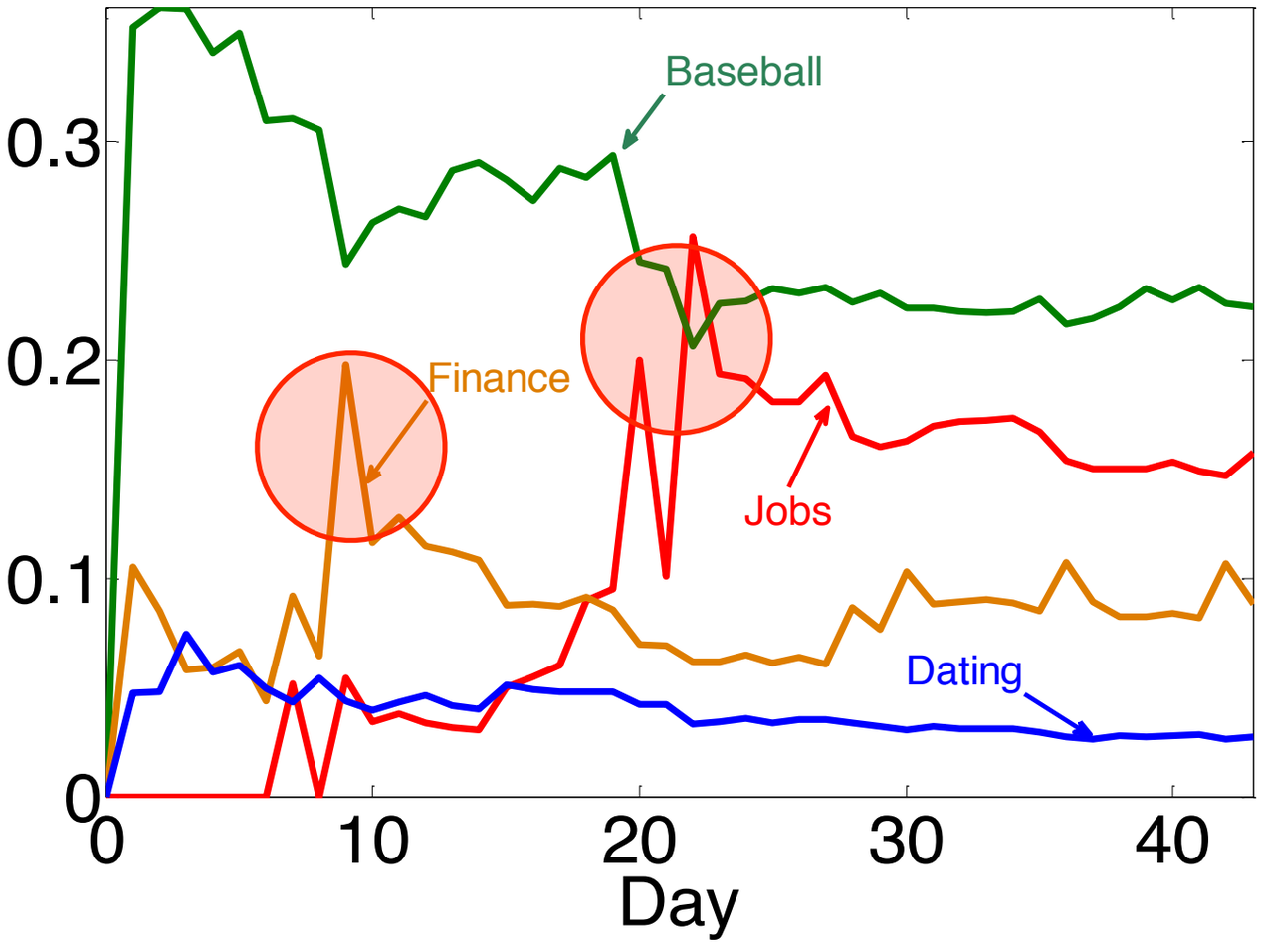




Dating	Baseball	Celebrity	Health	Jobs	Finance
women	League	Snooki	skin	job	financial
men	baseball	Tom	body	career	Thomson
dating	basketball,	Cruise	fingers	business	chart
singles	doublehead	Katie	cells	assistant	real
personals	Bergesen	Holmes	toes	hiring	Stock
seeking	Griffey	Pinkett	wrinkle	part-time	Trading
match	bullpen	Kudrow	layers	receptionist	currency
	Greinke	Hollywood			

Some Users



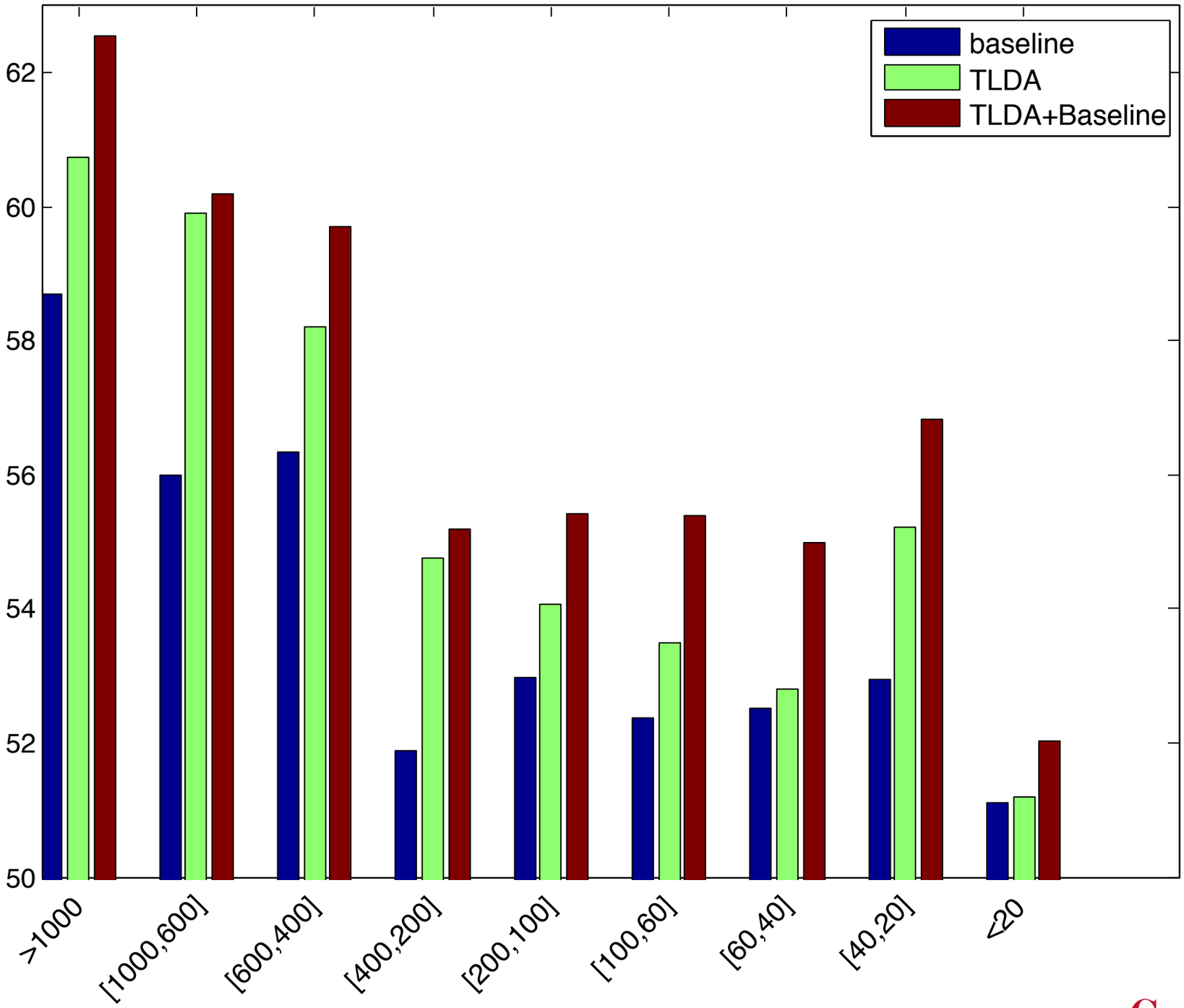


Dating	Baseball	Celebrity	Health	Jobs	Finance
women	League	Snooki	skin	job	financial
men	baseball	Tom	body	career	Thomson
dating	basketball,	Cruise	fingers	business	chart
singles	doublehead	Katie	cells	assistant	real
personals	Bergesen	Holmes	toes	hiring	Stock
seeking	Griffey	Pinkett	wrinkle	part-time	Trading
match	bullpen	Kudrow	layers	receptionist	currency
	Greinke	Hollywood			

Google

University

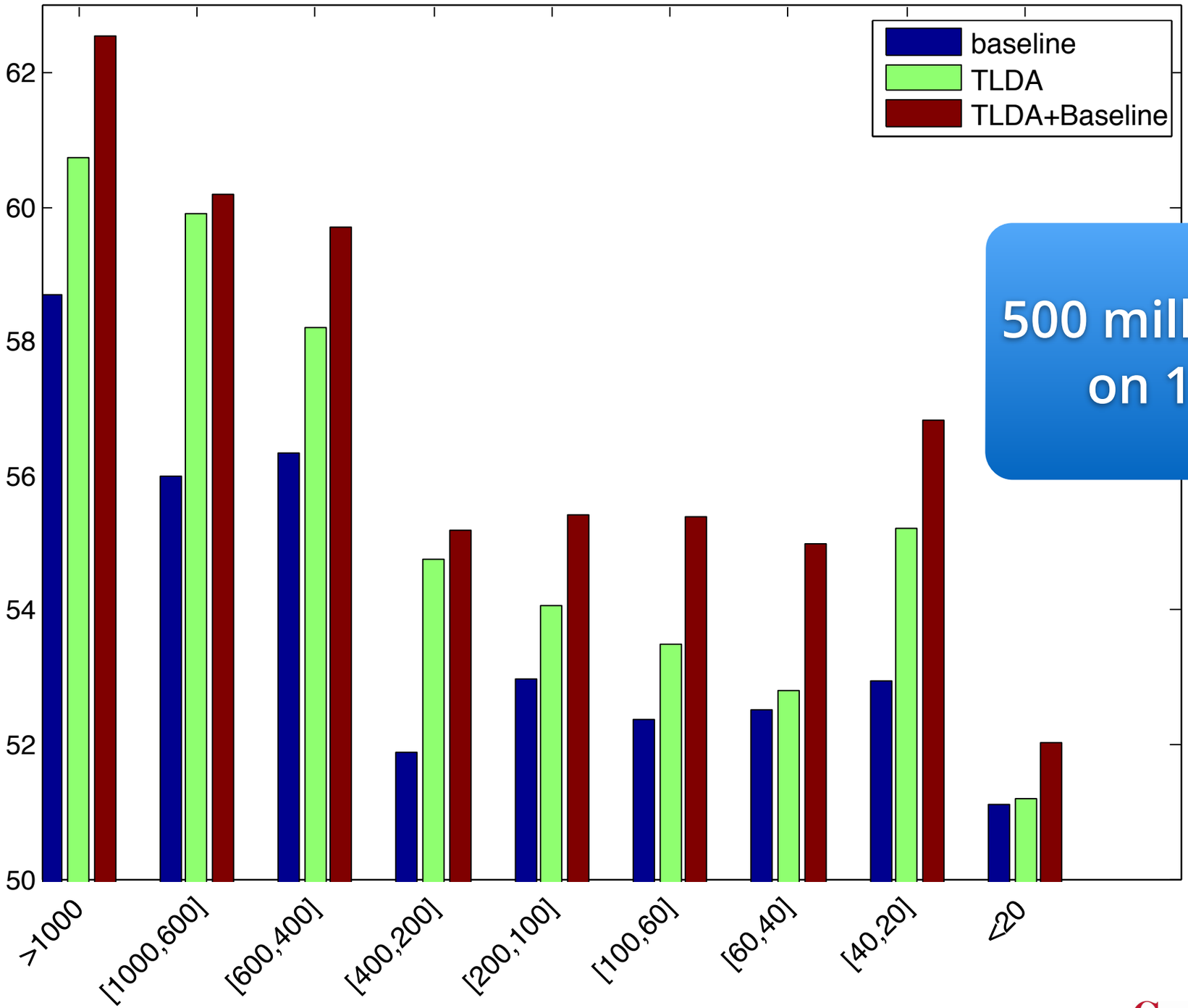
Improvement ($$$)



Google

Carnegie Mellon University

Improvement ($$$)



500 million users per day on 1000 machines

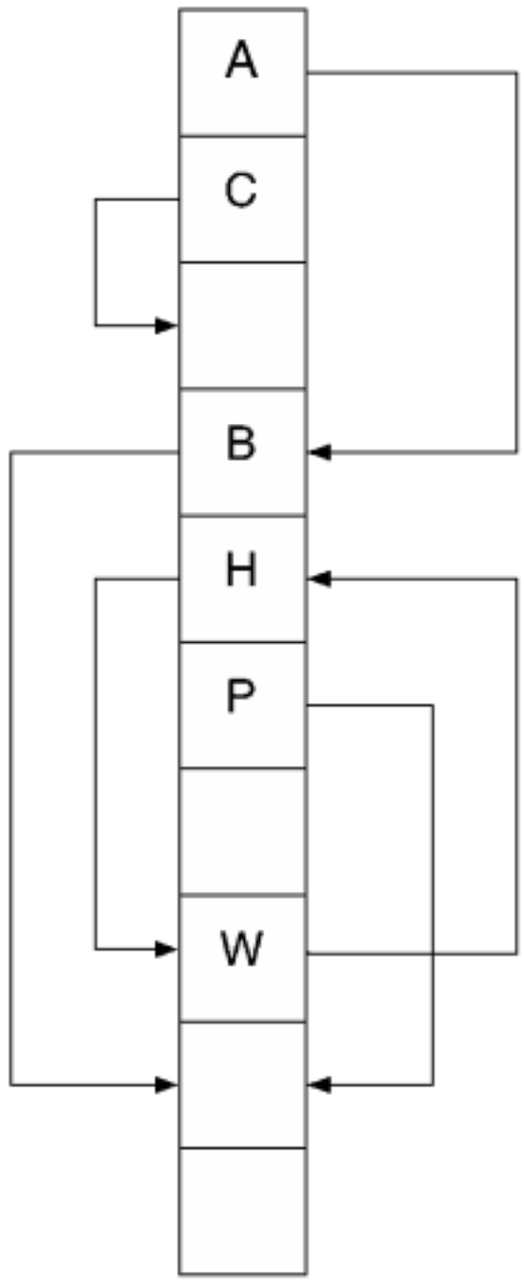
Google

Carnegie Mellon University

Cuckoo Hashing

Fast and exact L1 linear algebra

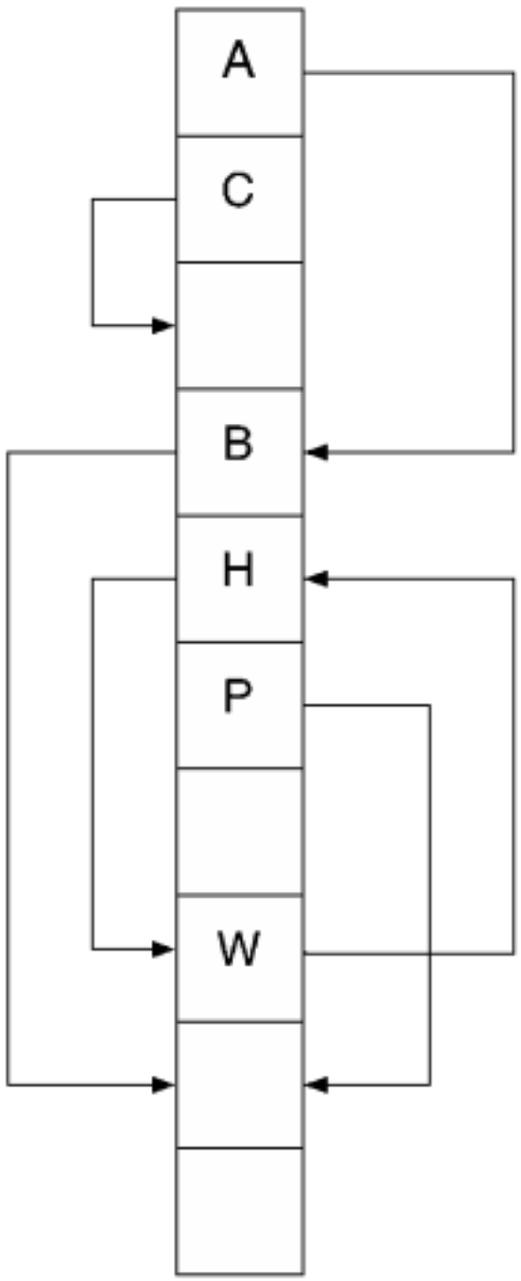
Cuckoo Hash



- Use power of two choices trick
  - Given key x insert into array at h1(x) or h2(x)
  - If both slots are full, evict existing key
  - This key now starts finding a new location
- O(1) lookup for arbitrary keys
- Short eviction chain

Pagh & Rodler 01

Cuckoo Hash



Andersen et al. 2012

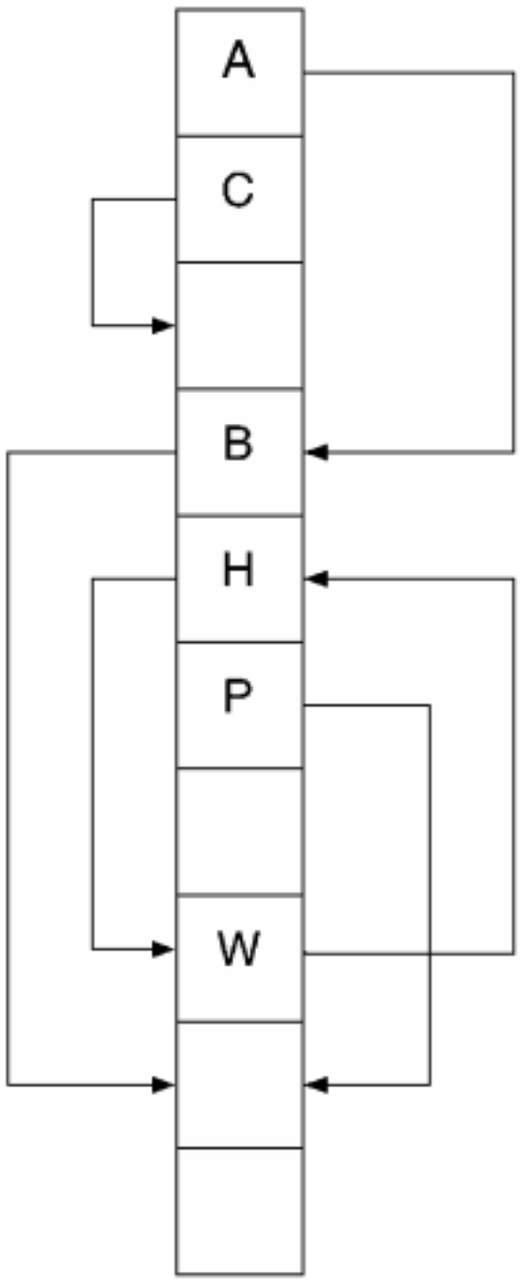
- O(1) lookup for arbitrary keys
- Short eviction chain
- Need to keep key to compute when switching.
- Alternative

$$h_2(x) = h_1(x) \,\mathrm{XOR}\, h(s(x))$$

$$\text{hence } h_1(x) = h_2(x) \,\mathrm{XOR}\, h(s(x))$$

- Poor fill rate
- Use multiple slots per location

Cuckoo hashes



- In practice 80% fill rate is quite reasonable
- Dot products cost O(n) time (this is all we need)
- Merging lists costs O(n) time (for gradient updates)
- Prefetching from RAM to deal with cache misses
- No need to pre-synchronize lists (good for distributed algorithms)

Performance

- Accuracy and timing for l2

	LibLinear	Cuckoo	Hash Kernel $N = 27$	Hash Kernel $N = 20$
Pre-processing Time (sec)	514	0	0	0
Transpose Time (sec)	32	122	70	50
Training Time (sec)	2077	1520	2373	370
Total Time (sec)	2623	1643	2444	421
Memory Used (GB)	29	21	30	23
Accuray	93.22%	93.23%	93.25%	90.96%

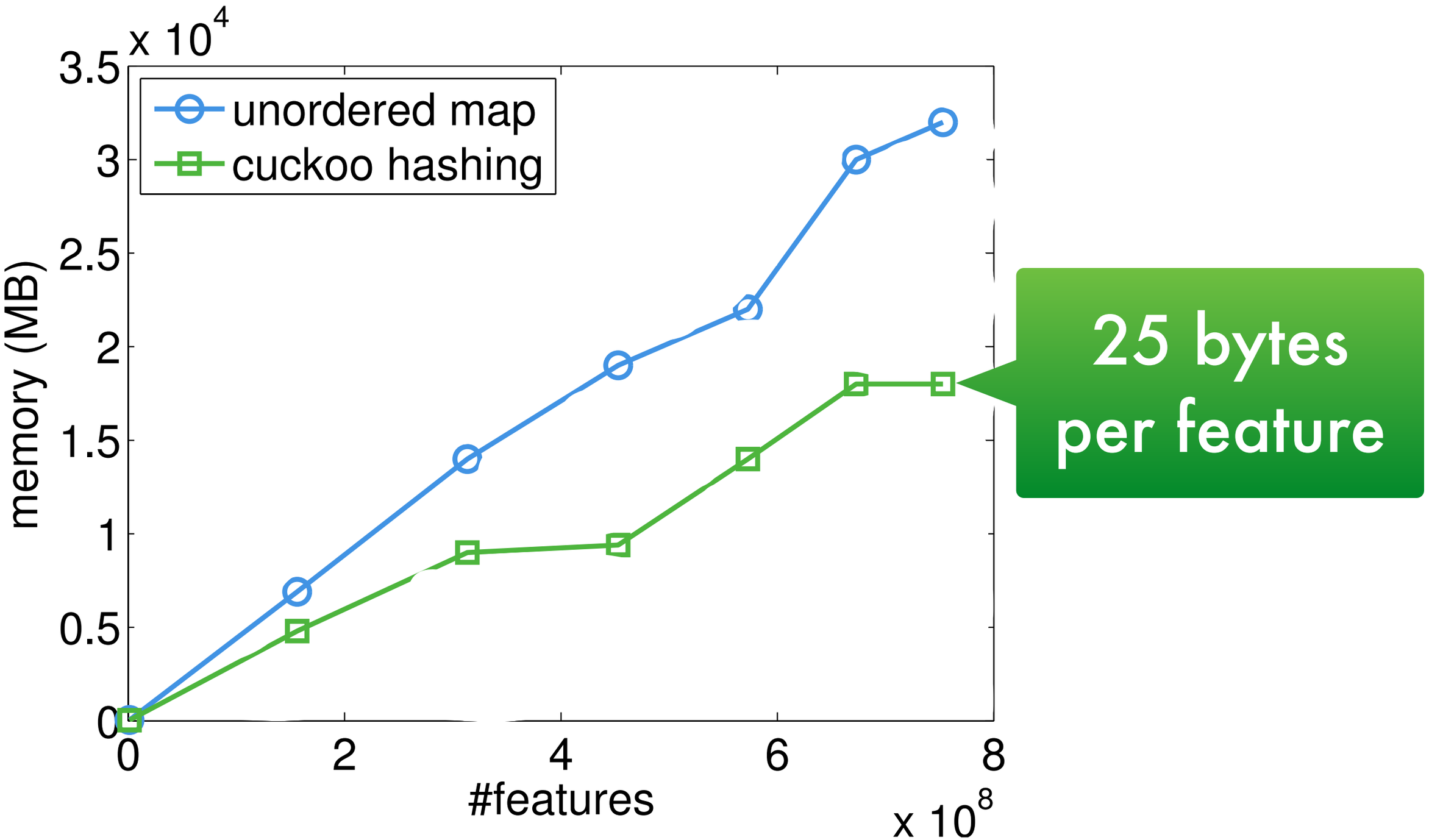
- Reconstruction accuracy for l1 (Jaccard score for different SGD runs)

LibLinear	Cuckoo	hash kernel	
		$N = 20$	$N = 27$
0.8777	0.8773	0.0052	0.2638

Zhou, Li, Andersen, Smola, 2014 (submitted)

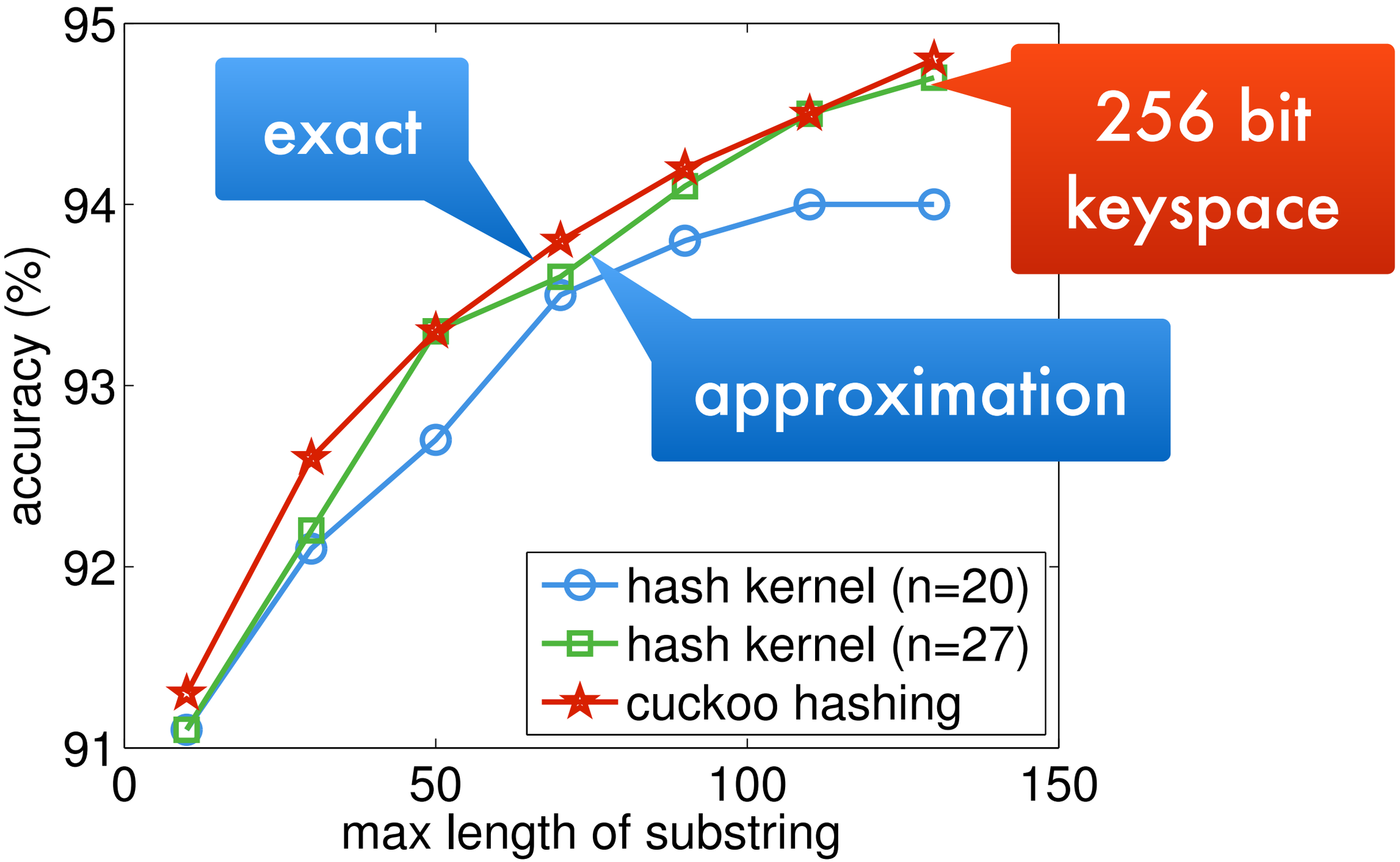
Memory efficiency

(cuckoo hash vs. unordered STL map)



Generating all substrings

(Pascal gene classification data)



Hash Kernels

Hash Kernels

instance

Hey,

please mention subtly during your talk that people should use Yahoo* search more often.
Thanks,

Someone important

dictionary

task/user

1
2
1
1
1

sparse

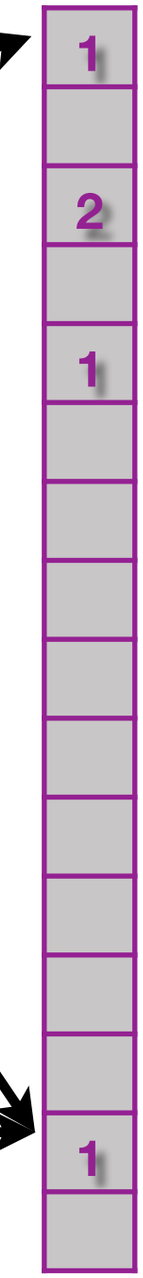
Hash Kernels

instance dictionary

Hey,

please mention subtly during your talk that people should use Yahoo* search more often.
Thanks,

Someone important



task/user

sparse

*in the old days

Hash Kernels

instance

Hey,

please mention subtly during your talk that people should use Yahoo* search more often.
Thanks,

Someone important

task/user

dictionary

hash function

$h()$

sparse

1
2
1
1
1

sparse

1
3
2
1
1

*in the old days

Hash Kernels

instance

Hey,

please mention subtly during your talk that people should use Yahoo* search more often.
Thanks,

Someone important

mention

hash function

$h()$

1
3
2
1

sparse

mention_barney

task/user

*in the old days

Hash Kernels

instance

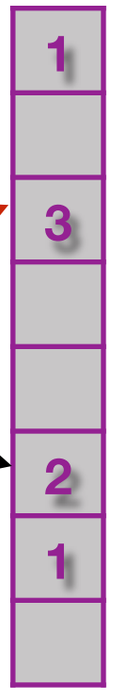
Hey,

please mention subtly during your talk that people should use Yahoo* search more often.
Thanks,

Someone important

hash function

$h(\text{mention})\sigma(\text{mention})$



sparse

task/user

$h(\text{mention}, \text{barney})\sigma(\text{mention}, \text{barney})$

Similar to count sketch (Charikar, Chen, Farrach-Colton, 2002)

*in the old days

Hash Kernels

- Linear model

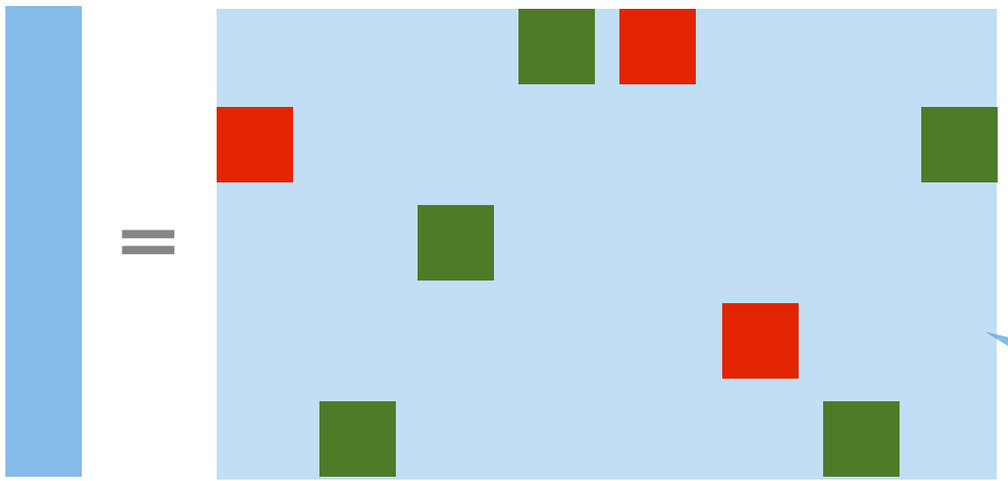
$$f(x) = \langle w, x \rangle = \sum_i w_i x_i$$

- Hashed model

$$f(x) = \sum_i w_{h(i)} x_i \sigma(i)$$

- Applications
  - Multitask learning
  - Fast string kernels
  - Collaborative filtering (compress matrix)

Hash Kernels - the matrix view



$z = Px$

maximally sparse matrix

$$\langle w, x \rangle = \sum_i w_i x_i \qquad \langle \bar{w}, \bar{x} \rangle = \sum_j \left[\sum_{i:h(i)=j} w_i \sigma(i) \right] \left[\sum_{i:h(i)=j} x_i \sigma(i) \right]$$

$$\mathbf{E}_\sigma[\sigma(i)\sigma(i')] = \delta_{ii'}$$

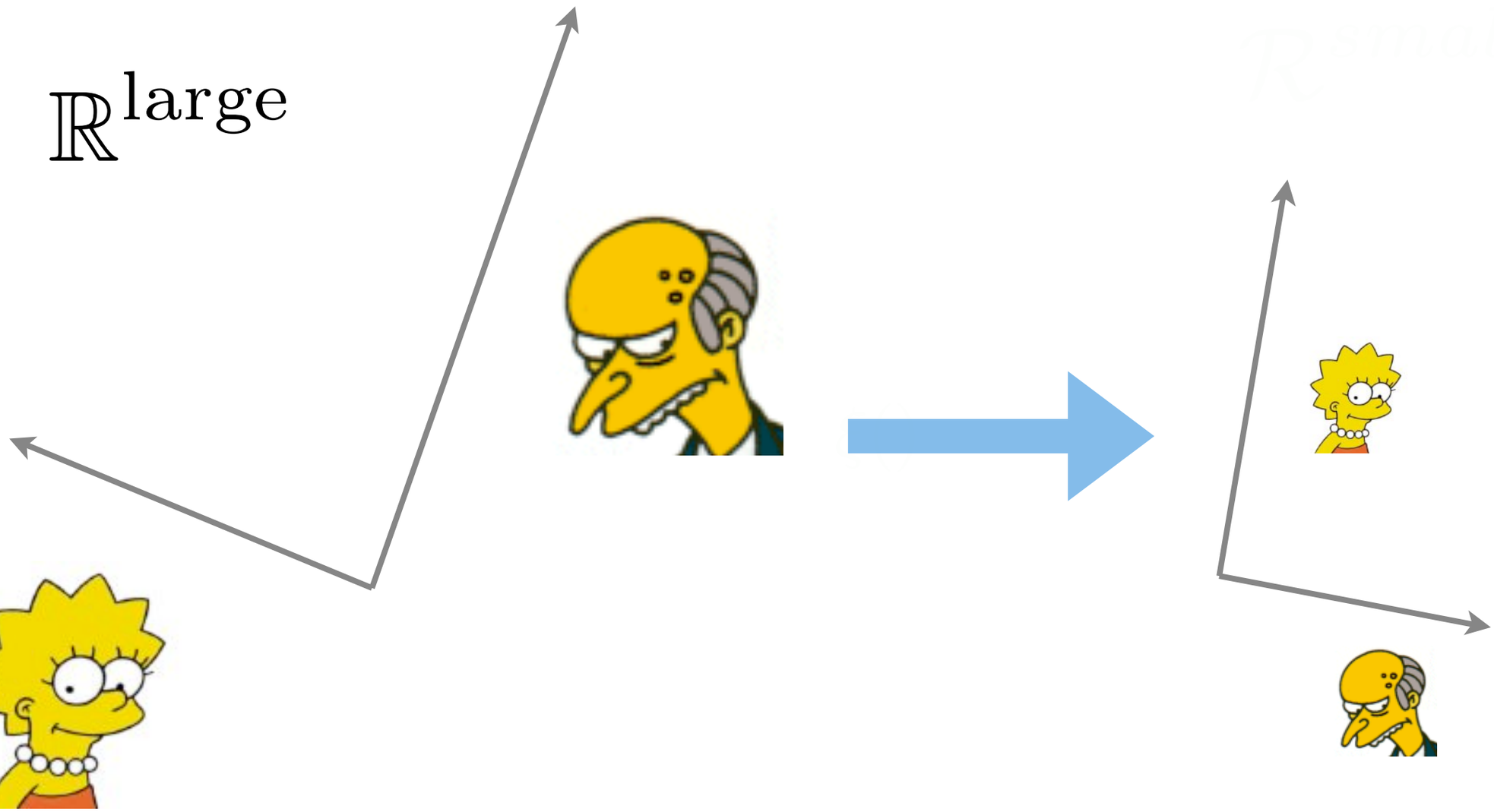
Advantages of hashing

- No dictionary needed
  No tokenization table
- Content drift is no problem

- All memory used for classification
  No projection matrix (vs. LSH)
- Finite memory footprint
- Sparsity preserving! (vs LSH)

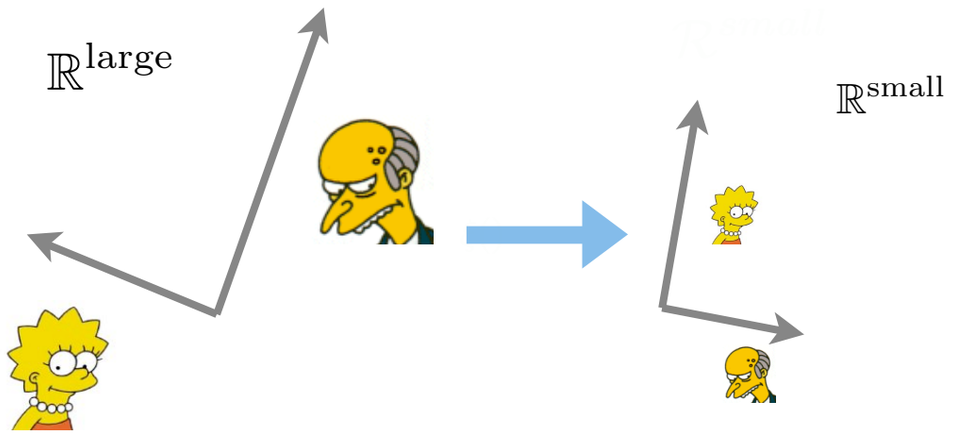
Intuition

$\mathbb{R}^{\text{large}}$



$\mathbb{R}^{\text{small}}$

Guarantees



- For a random hash function the inner product vanishes

$$\Pr\{|\langle w_v, h_u(x)\rangle| > \epsilon\} \leq 2e^{-C\epsilon^2 m}$$

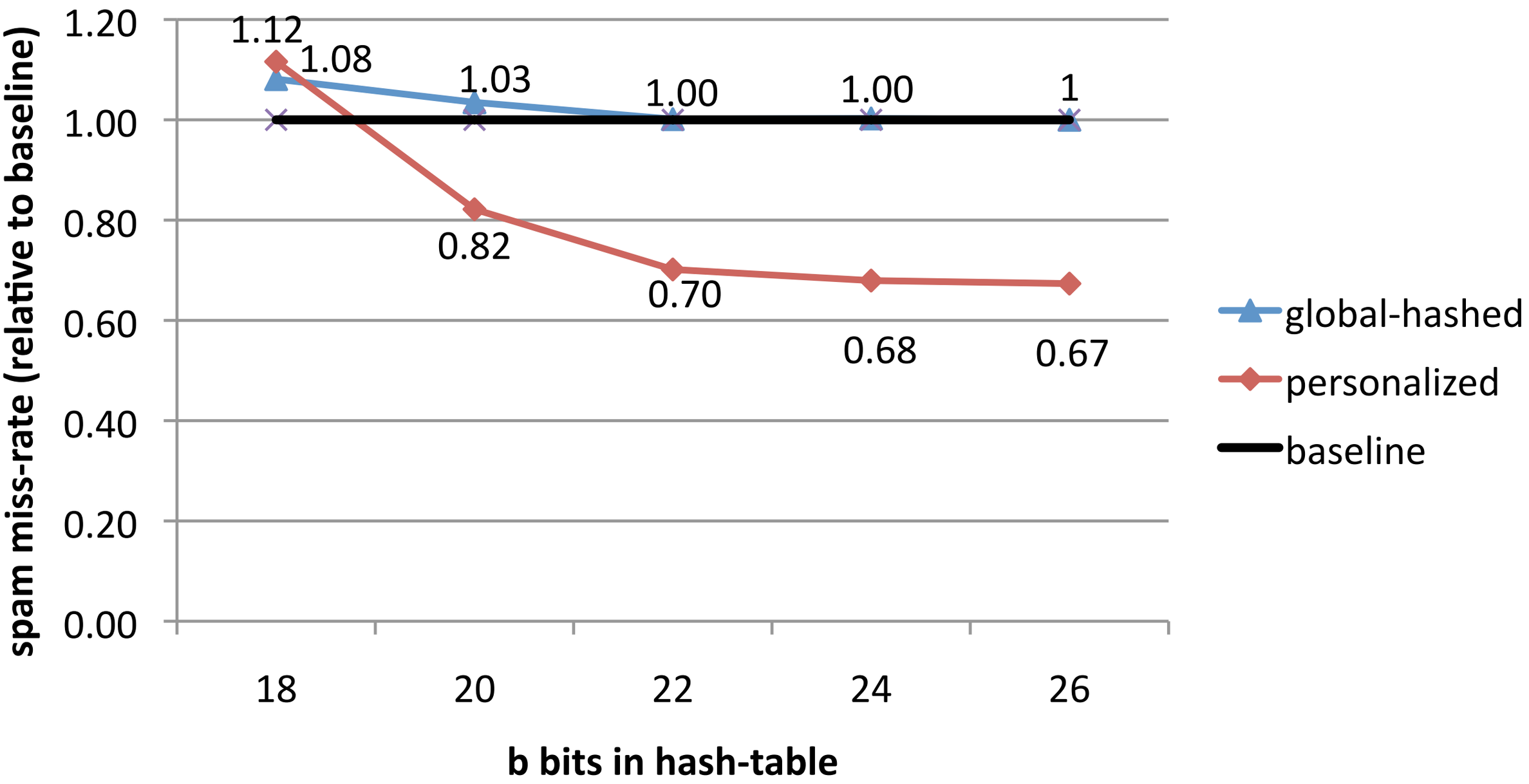
- Direct sums in Hilbert space

Direct sum in Hilbert Space → Sum in Hash Space

- Hashed inner product is unbiased, variance is O(1/n)
- Restricted isometry property (Kumar, Sarlos, Dasgupta 2010)

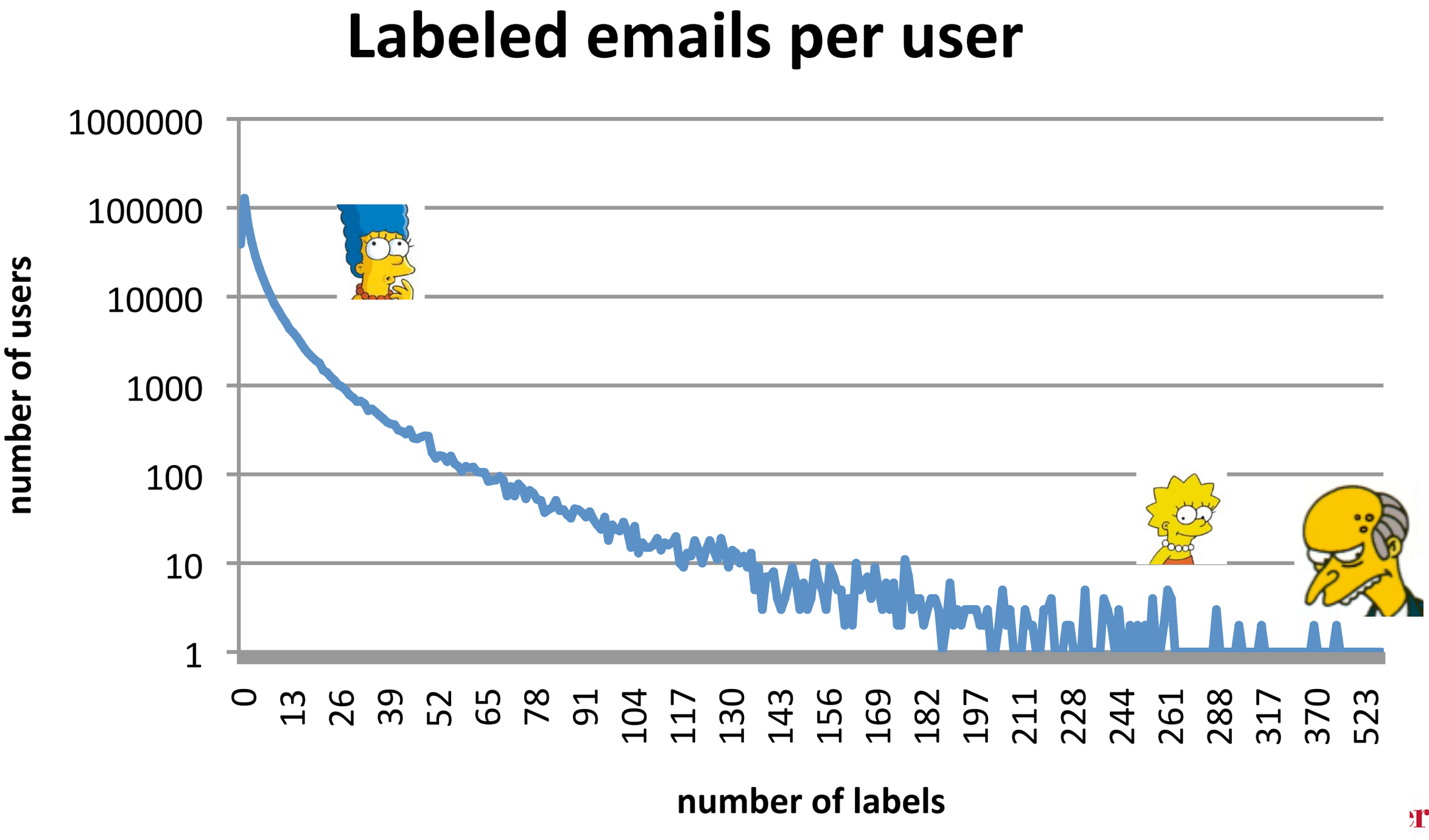
Weinberger, Dasgupta, Attenberg, Langford and Smola
Feature Hashing for Large Scale Multitask Learning, ICML'09

Spam classification results

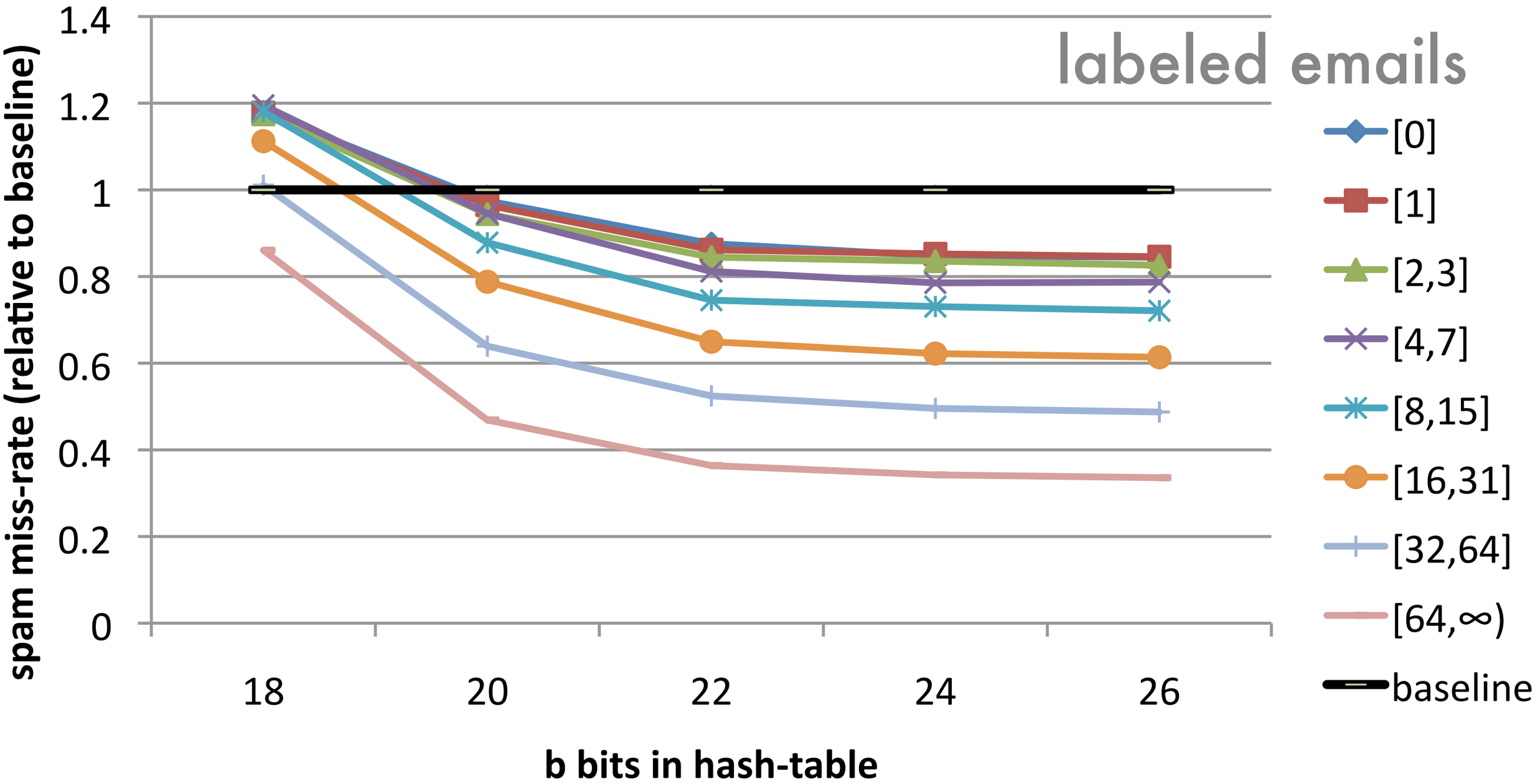


20 Million documents, 400k users

Lazy users …


Labeled emails per user
number of users
1000000
100000
10000
1000
100
10
1
0
13
26
39
52
65
78
91
104
117
130
143
156
169
182
197
211
228
244
261
288
317
370
523
number of labels

Results by user group



Results by user group