Templates

for scalable data analysis

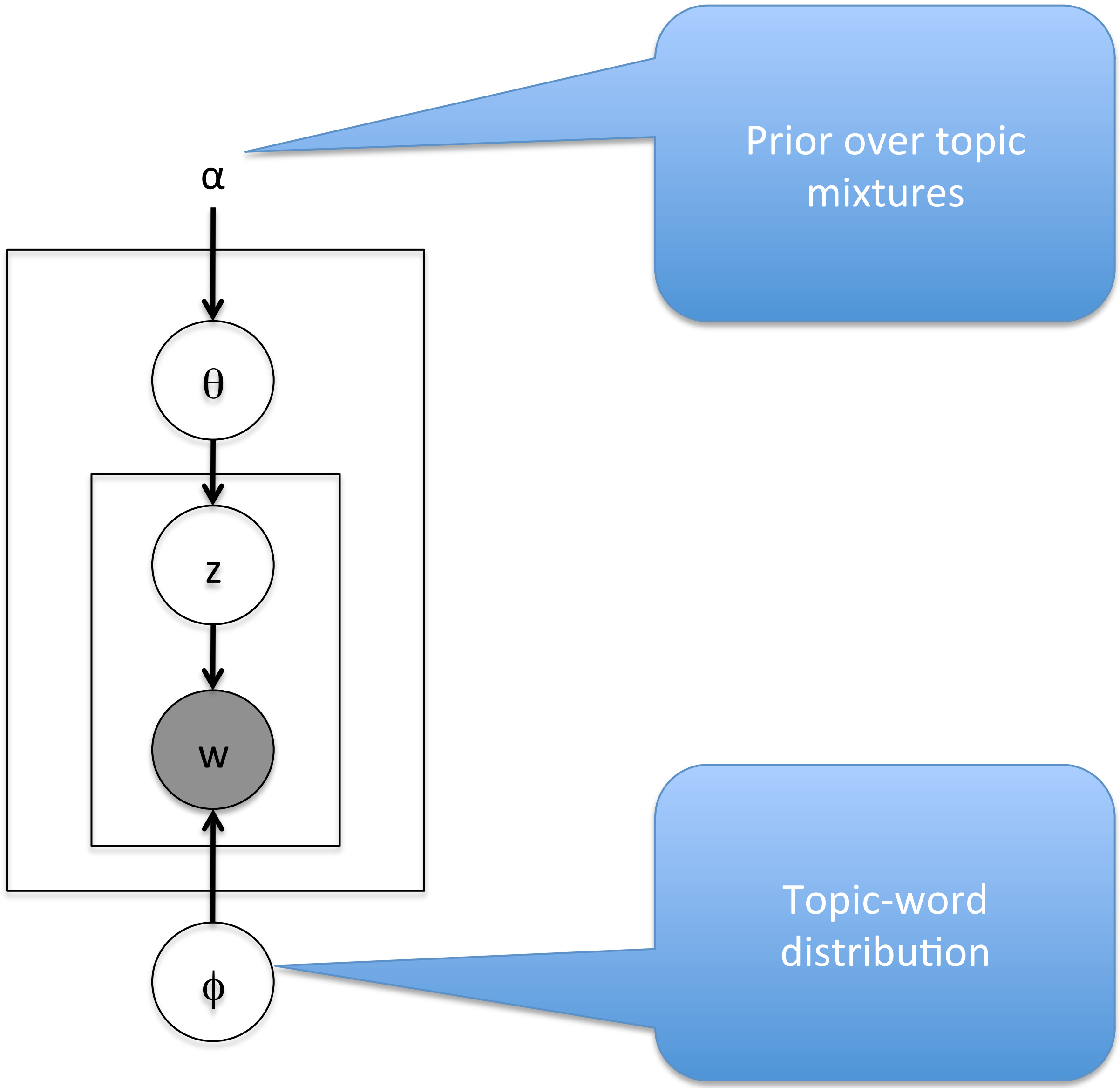
4 Applications:
User Modeling and Graph Factorization

Amr Ahmed, Alexander J Smola, Markus Weimer
Yahoo! Research & UC Berkeley & ANU

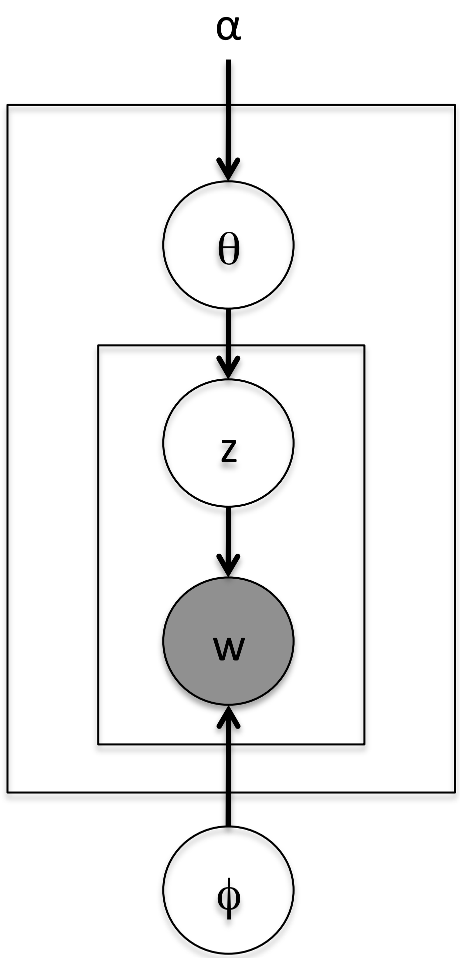
Wrapping up

- Distributed inference in latent variable models
  - Star Synchronization
  - Delta aggregation

Wrapping up …

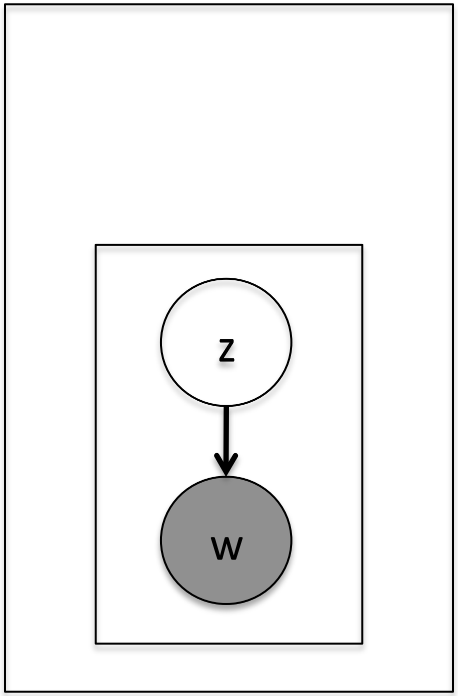


Wrapping up ...



- Global variables
  - Φ: Topic distribution over words
- Local variables
  - θ: topic mixing vector
  - Z: topic indicator

Wrapping up …

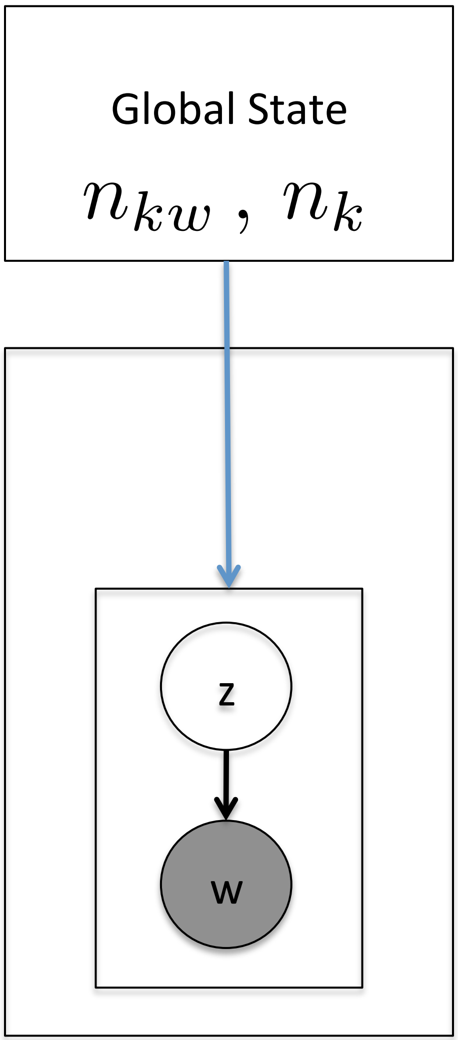


- Collapse global variables
  - Φ
- Collapse local variables
  - θ
- Couples all Zs
- Run collapsed sampler

$$P(z_{di} = k | w_{di} = w, z_{-di}) \propto$$

$$(n_{dk} + \alpha) \frac{n_{kw} + \beta}{n_k + W\beta}$$

Wrapping up ...

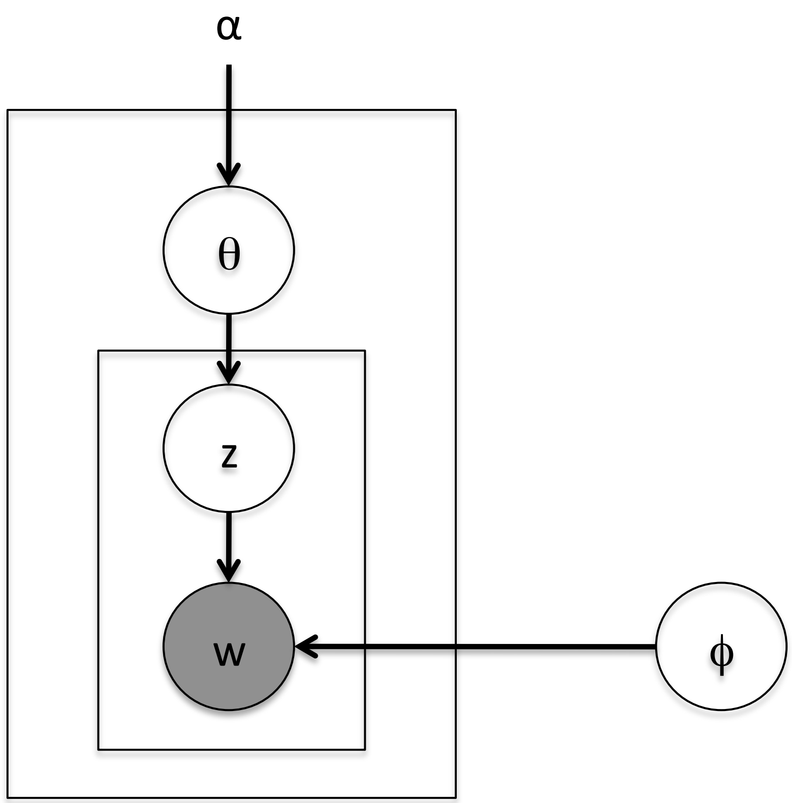


Local counts (local state)

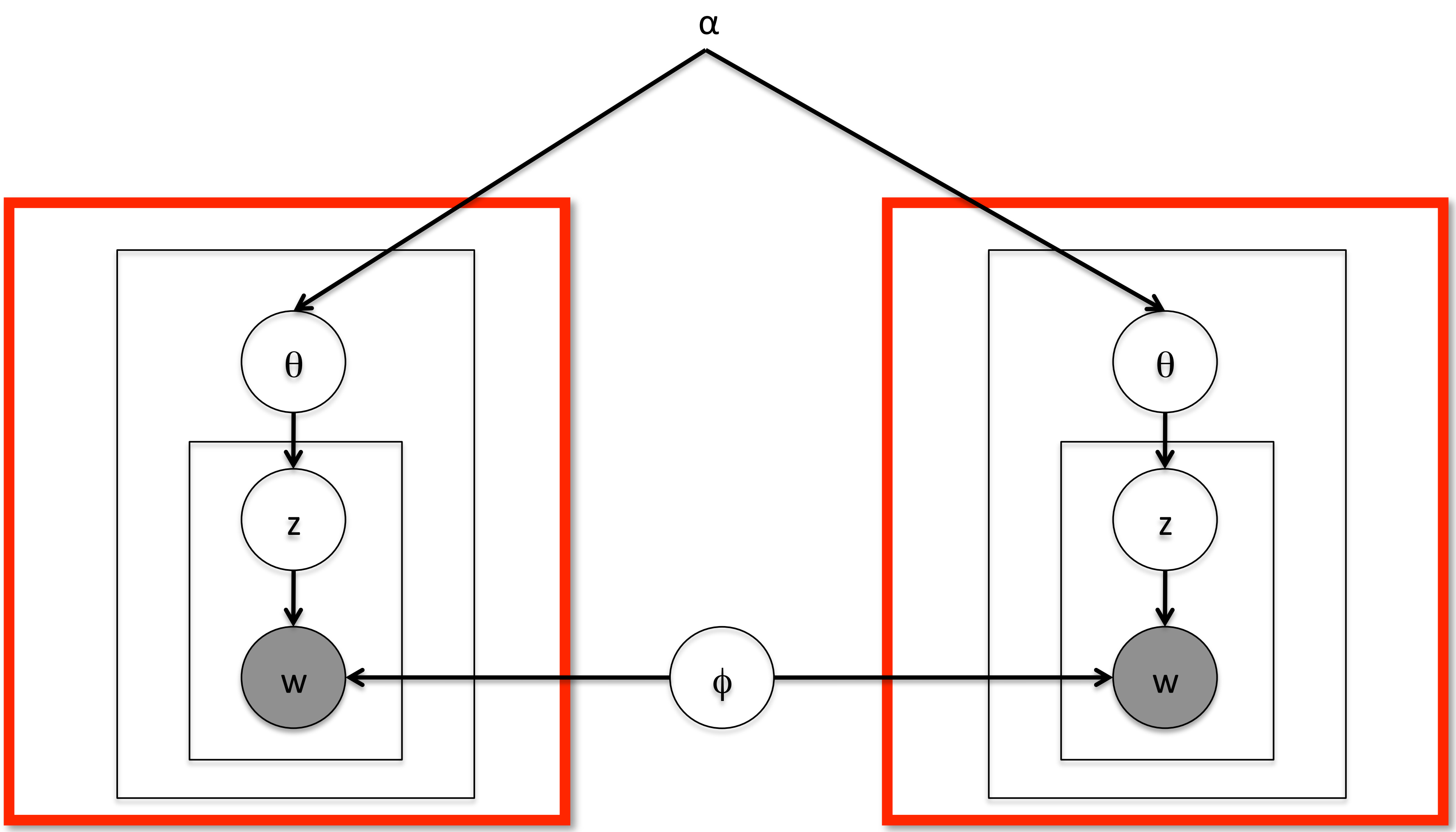
Global counts (global state)

$$P(z_{di} = k | w_{di} = w, z_{-di}) \propto (n_{dk} + \alpha) \frac{n_{kw} + \beta}{n_k + W\beta}$$

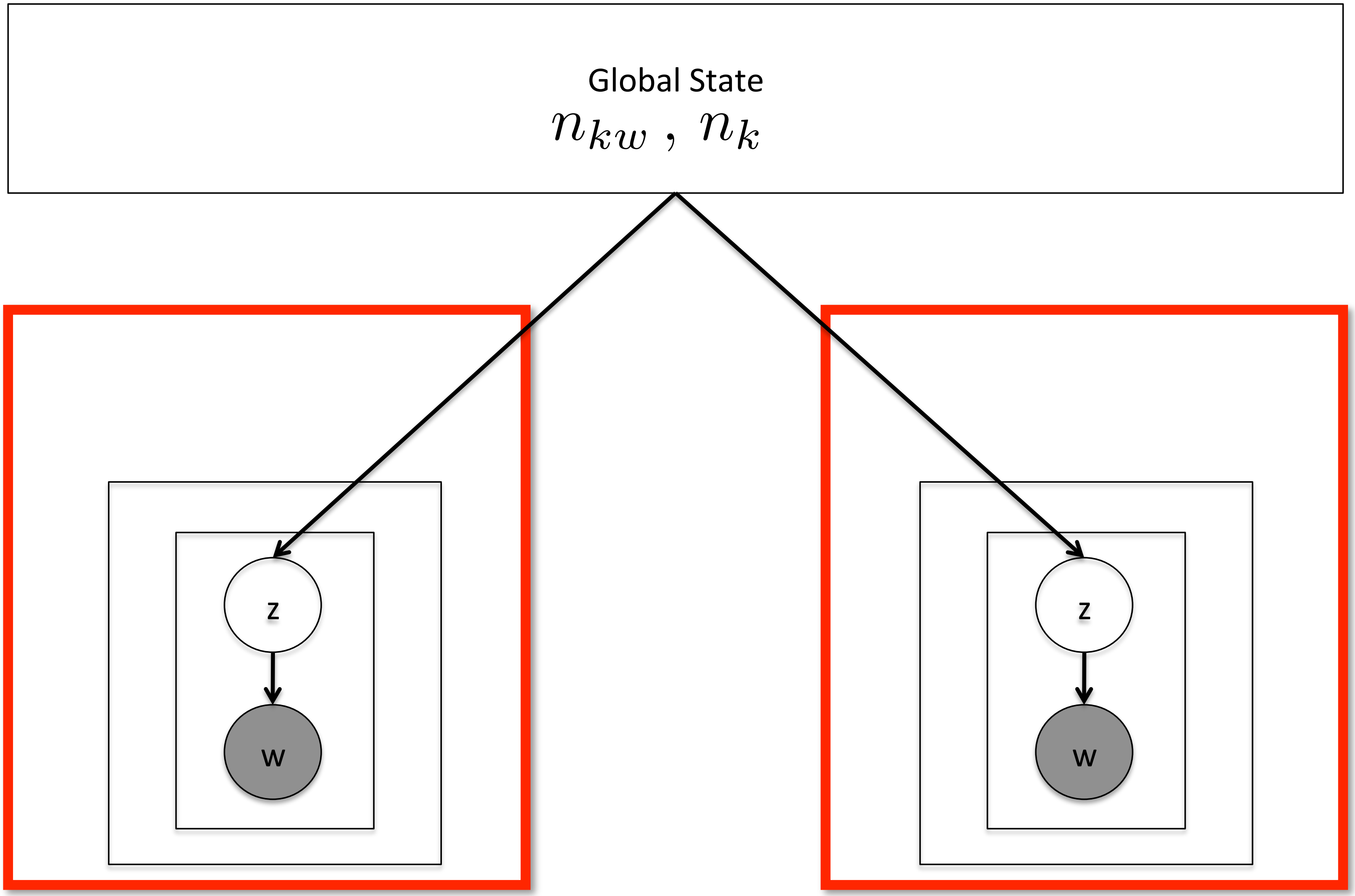
Distributed Inference: LDA



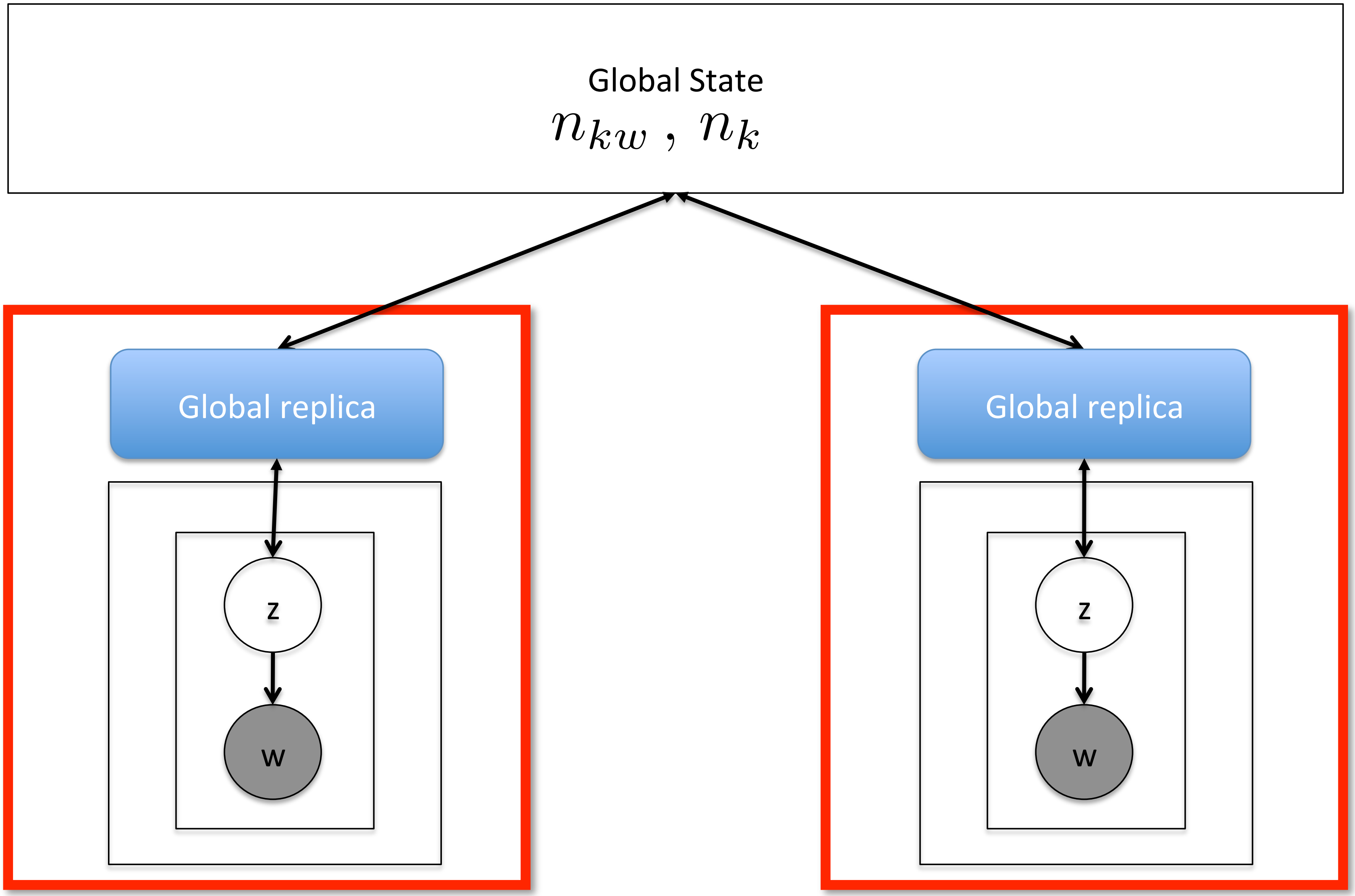
Distributed Inference: LDA



Distributed Inference: LDA

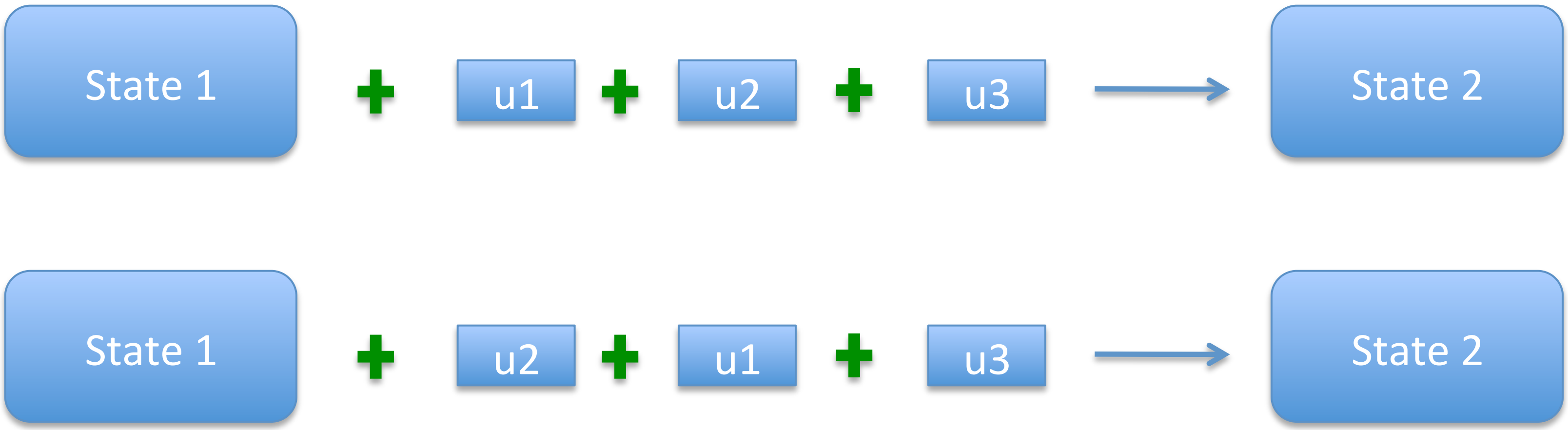


Distributed Inference: LDA



General Architecture

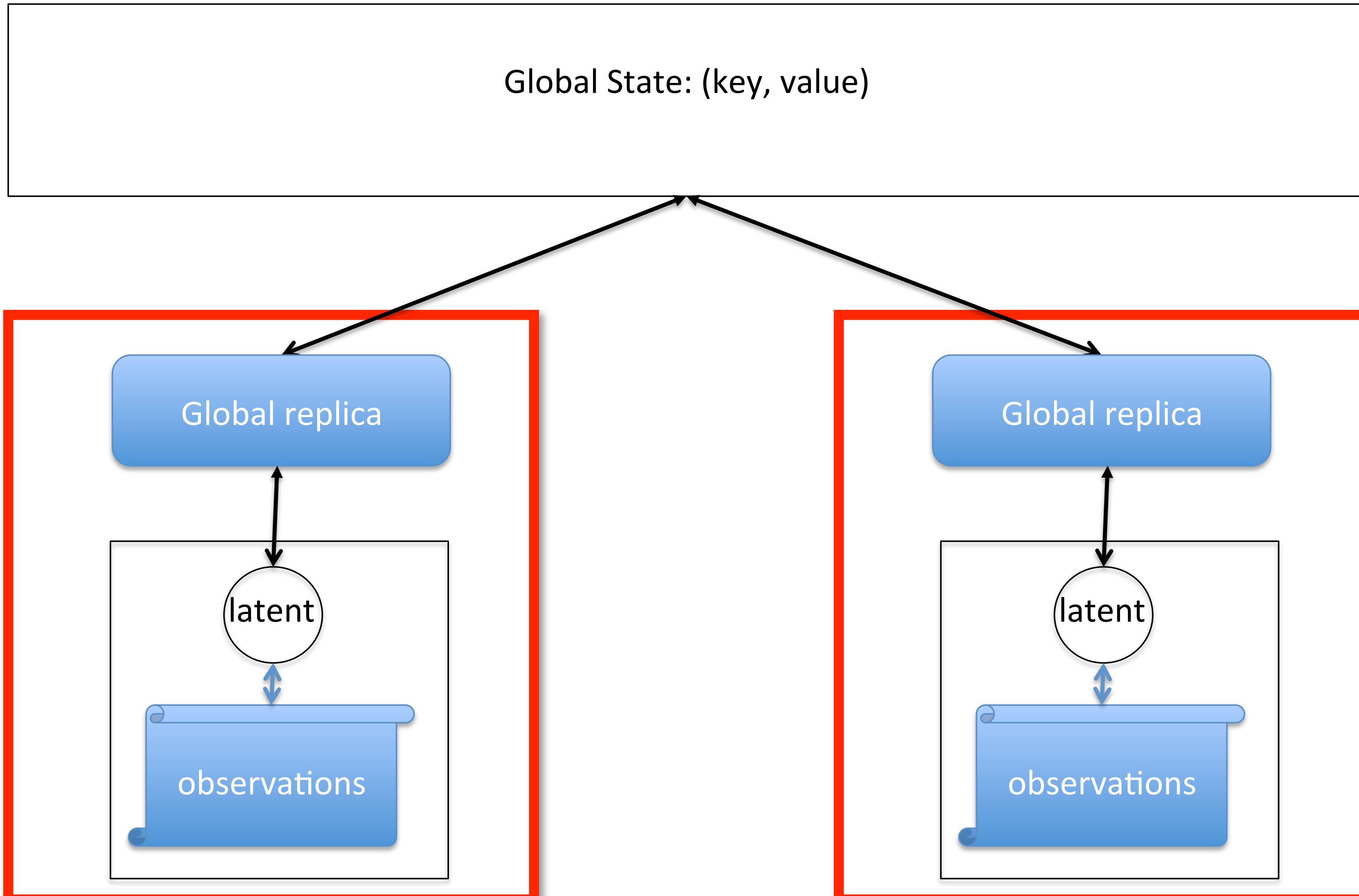
- Star synchronization
  - Works when variables depend on each other via aggregates
    - Counts, sums, etc.
  - When state objects form an Abelian group



Template

- Fit most topic models in collapsed representation
  - Define the state (key, value) pairs
    - Mostly counts, sums, lists, hash tables
  - Define the +,- operations on a state object
  - Write your sampler
    - Input: document, state
    - Output:
      - Update document local variables
      - Update the global state
- Our API will take care of the rest
  - Synchronization, threading, distribution, etc

Distributed Inference: template



State Example: LDA

$$P(z_{di} = k | w_{di} = w, z_{-di}) \propto$$

$$(n_{dk} + \alpha) \frac{n_{kw} + \beta}{n_k + W\beta}$$

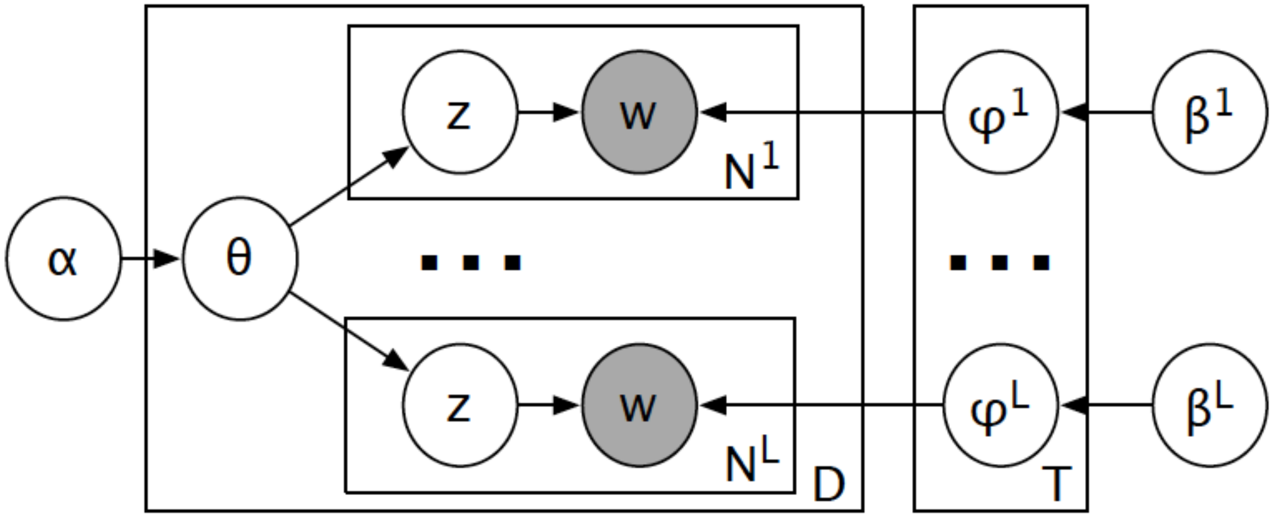
- Alternative 1
  - Key: (topic, word)
  - value: count
  - Operators:
    - +,- are trivially defines
- Alternative 2
  - Key: word
  - value: list of (topic, count)
    - Allows efficient samplers
  - Operators: sparse vector operations
    - Might need to delete and merge

State Example: LDA

- You get the idea?
- Define the state to work with your sampler
- Define +,- for synchronization
- All details are abstracted form the synchronization logic
  - It just uses the +,- operators your just defined
  - Requires an iterator over state objects

Example 2: Multilingual LDA

- Each topic has a distribution over words
- Fits parallel documents
  - Example: Wikipedia



State Example: Multilingual-LDA

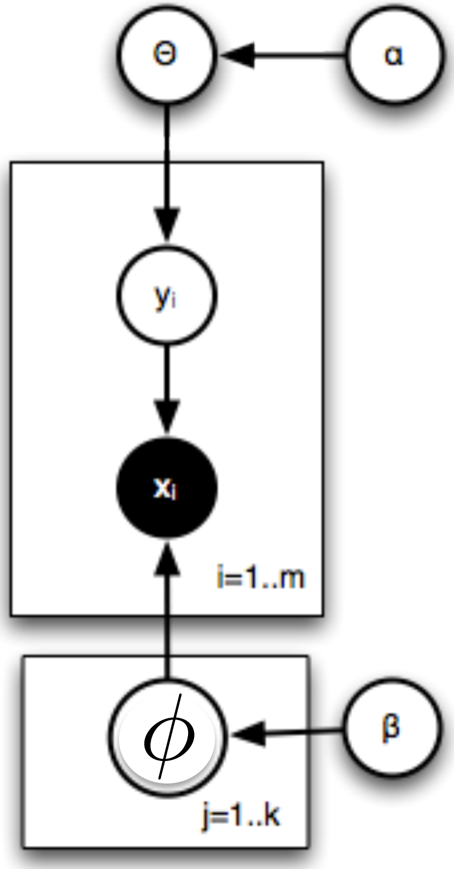
- Alternative 1
  - Key: (topic, language, word)
  - value: count
  - Operators: +,- are trivially defines

- Alternative 2
  - Key: word
  - value: list of (topic, language, count)
    - Allows writing efficient samplers
  - Operators: Sparse vector operations
    - Might need to delete and merge

State Example: Clustering

- Alternative 1
  - Key: Cluster ID
  - value:
    - Document counts
    - Parameter representation
      - Hash table: (word, count)

  - Operations
    - Define +,- over each field
    - You write this code
    - Part of the application logic
    - You have to do it anyhow when:
      - Remove or add a document to a cluster



API Summary

- Template for distributed inference in latent variables models
- Two basic components
  - Document representation
    - You take care of that via Protocol Buffer
  - State representation
    - Key-value pairs
    - Value can be any object
      - Define +,- over that object
    - Provide an iterator over objects for the synchronizer

Code Snippet: object

```
class stats{
public:
    virtual ~stats() { };
    virtual void from_str(const string& serialized_stats) = 0;
    virtual void to_str(string& serialized_stats) = 0;

    virtual void operator+=(stats& inp) = 0;
    virtual void operator-=(stats& inp) = 0;

    virtual int get_id() { return 0; }
    virtual void set_id(int) { }

    virtual void print() { }
};

typedef auto_ptr<stats> stats_ptr;
```

Code Snippet: Container

```
class stats_container{
public:
    virtual ~stats_container() { };

    // copy operator
    virtual void from_stats_container(stats_container&) = 0;

    // lock up operator, get stat object with a given id
    virtual stats_ptr get_stats(int id) = 0;

    // update a state object with a give id
    virtual void update(int id, stats& delta) = 0;

    virtual int size() = 0;

    // iterator
    virtual bool has_next() = 0;
    virtual stats_ptr next() = 0;
    virtual void reset_iter() = 0;

    virtual void print() = 0;
};
```

Code Snippet: LDA Document

```
message LDA_document {
    optional string docID = 1;
    repeated uint32 body = 3 [packed=true]; // w
    repeated uint32 topic_assignment = 4 [packed=true]; //Z
    repeated uint32 topic_counts = 5 [packed=true];    // n_dk
}

message clustering_document {
    optional string docID = 1;
    repeated uint32 words = 2;  // w
    repeated uint32 label = 3;  // cluster assignment
}
```

Code Snippet: Sampler

```
class Model_Trainer {
public:
    virtual ~Model_Trainer() { };
    // read a document from disk
    virtual void* read(google::protobuf::Message&) = 0;

    //That is where you write your logic
    virtual void* sample(void* document) = 0;

    // Call in inference mode
    virtual void* test(void* document) = 0;

    // fold an update into the state
    virtual void* update(void* document) = 0;

    // time for synchronous operations
    virtual void* optimize(void*) = 0;

    // diagnostic
    virtual void* eval(void*,double&) = 0;

    //save
    virtual void write(void*) = 0;

   //need more iterations?
    virtual void iteration_done() = 0;
};
```

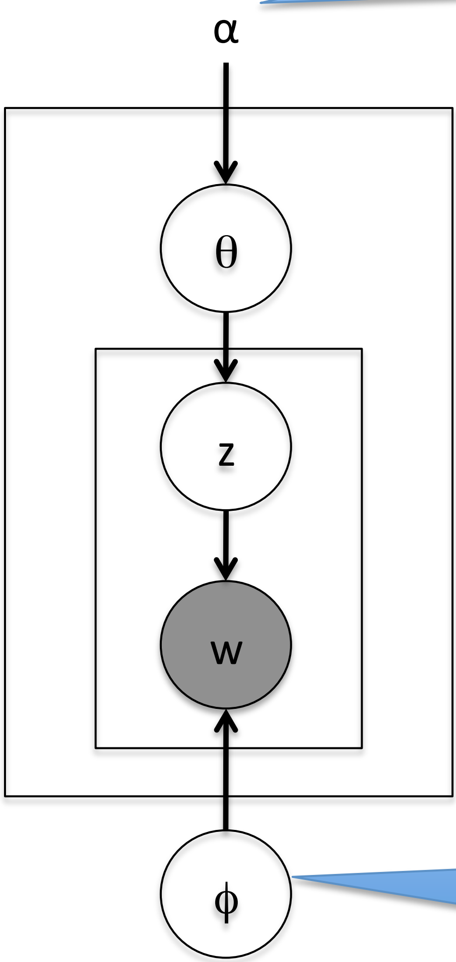
API Summary

- Current Yahoo! LDA release
  - Tightly integrates state, sampler and synchronization
  - Stay tuned for a new release with the new APIs

What Is next?

- Can we fit any model only with those asynchronous primitives?
  - No
- We need synchronous operations
  - Parameter optimization
    - EM style algorithm
  - Non-collapsed global variables

The Need for Synchronous Processing

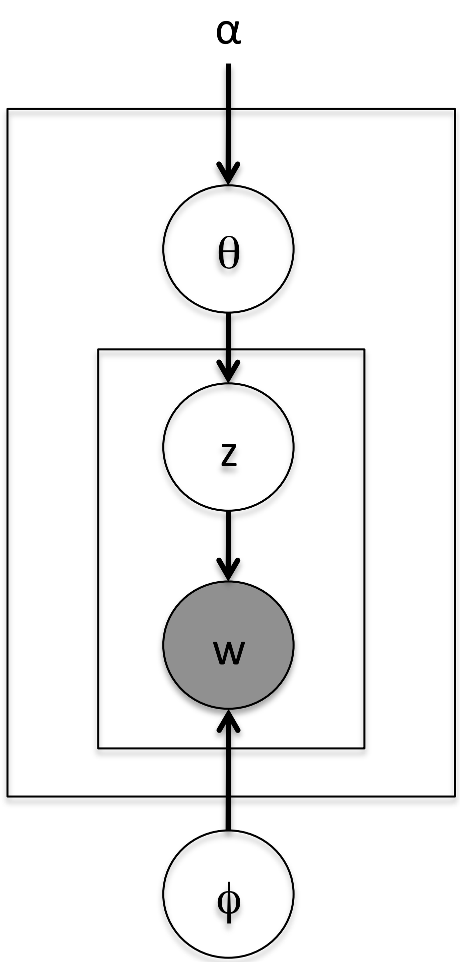


Prior over topic mixtures

What if we need to optimize over α?

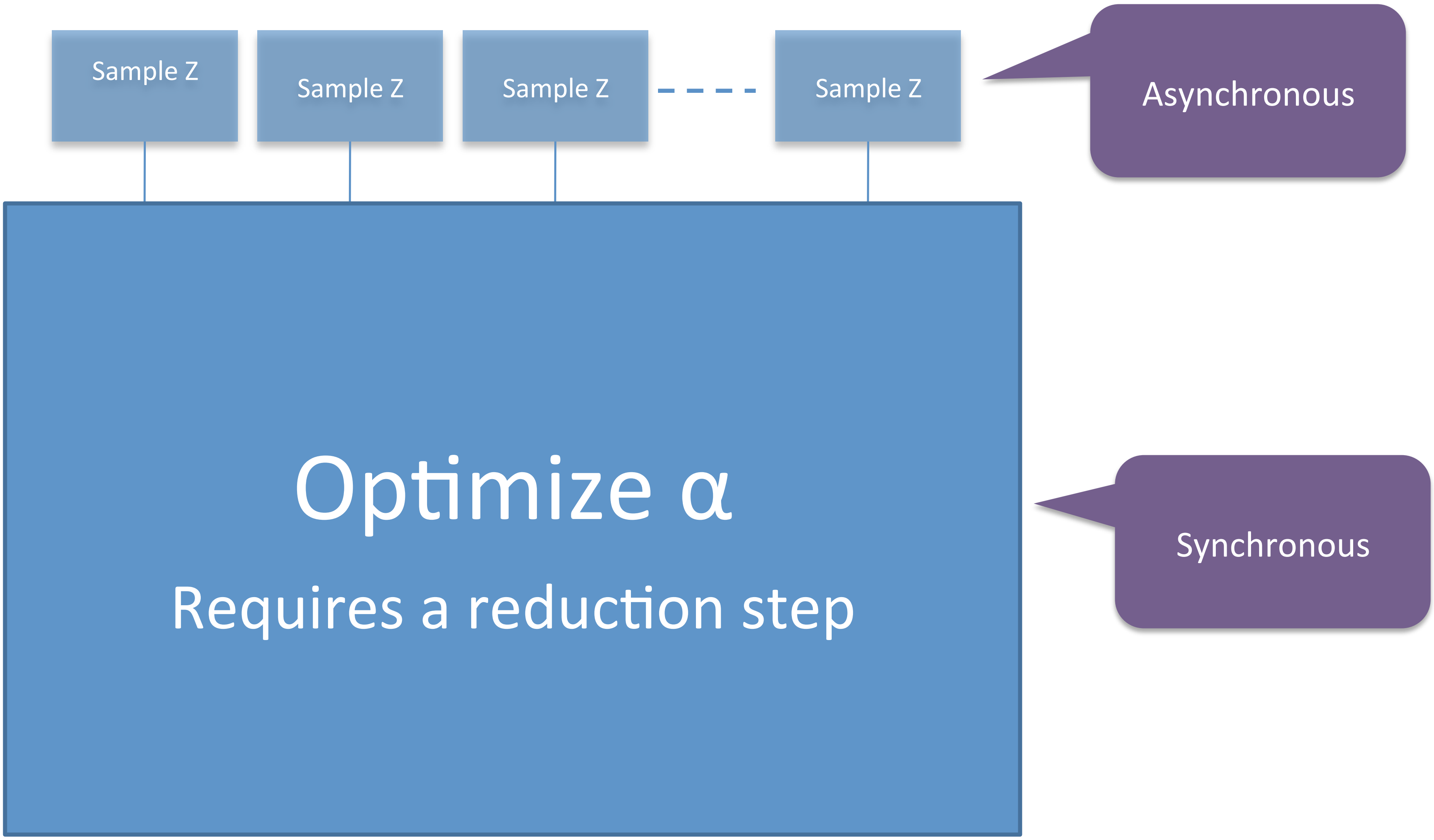
Topic-word distribution

The Need for Synchronous Processing

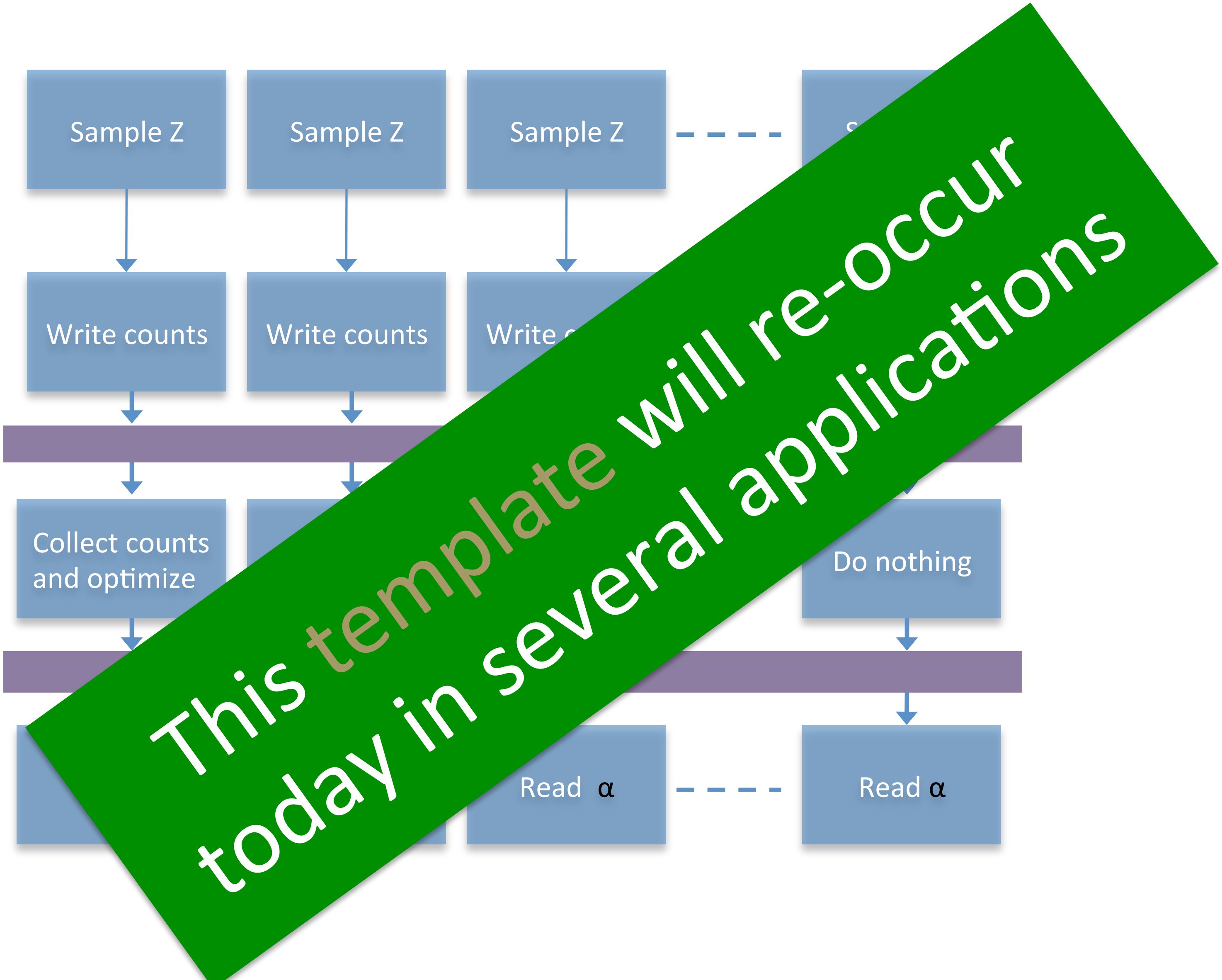


- E-Step
  - Run asynchronous collapsed sampler as before
- M-step
  - Reach a barrier
  - Collect values needed to optimize α
  - One machine optimizes α
  - Broadcast value back

Distributed Sampling Cycle



Distributed Sampling Cycle

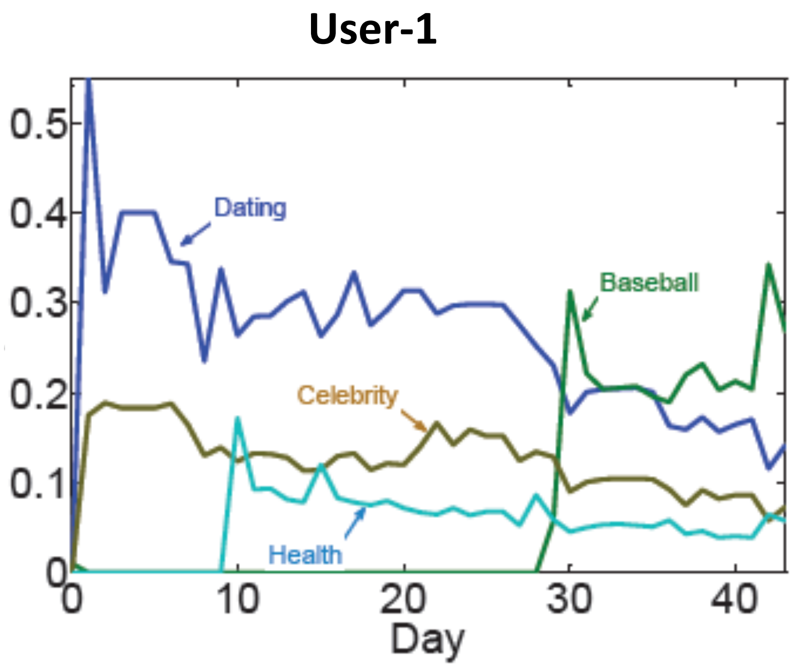


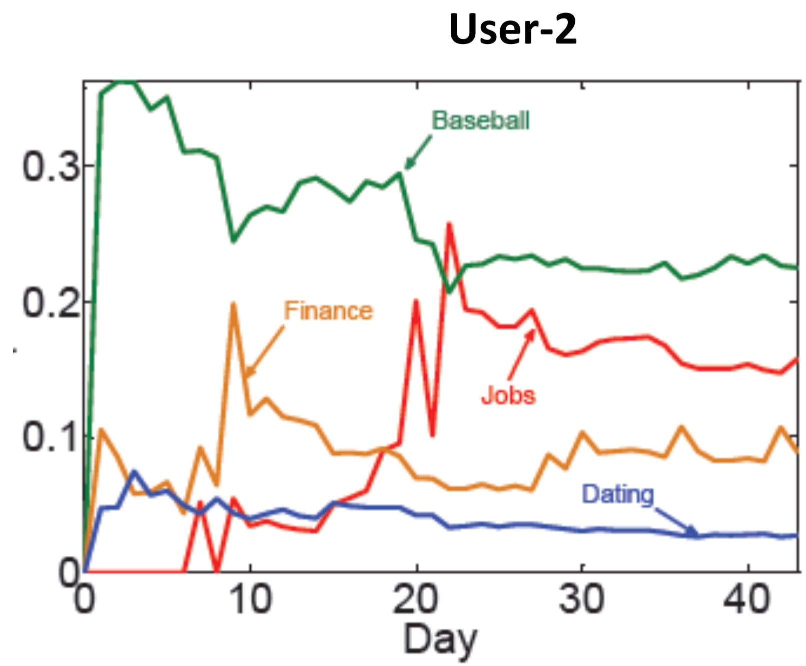
Up next

- Application
  - Temporal Modeling of user interests
  - Multi-domain user personalization

- Asynchronous Distributed Optimization
  - Can we get rid of the synchronous step?
  - Asynchronous consensus
  - Factorizing Y!M graph
    - 200 Million users and 10 Billion edges
    - The largest published work on graph factorization

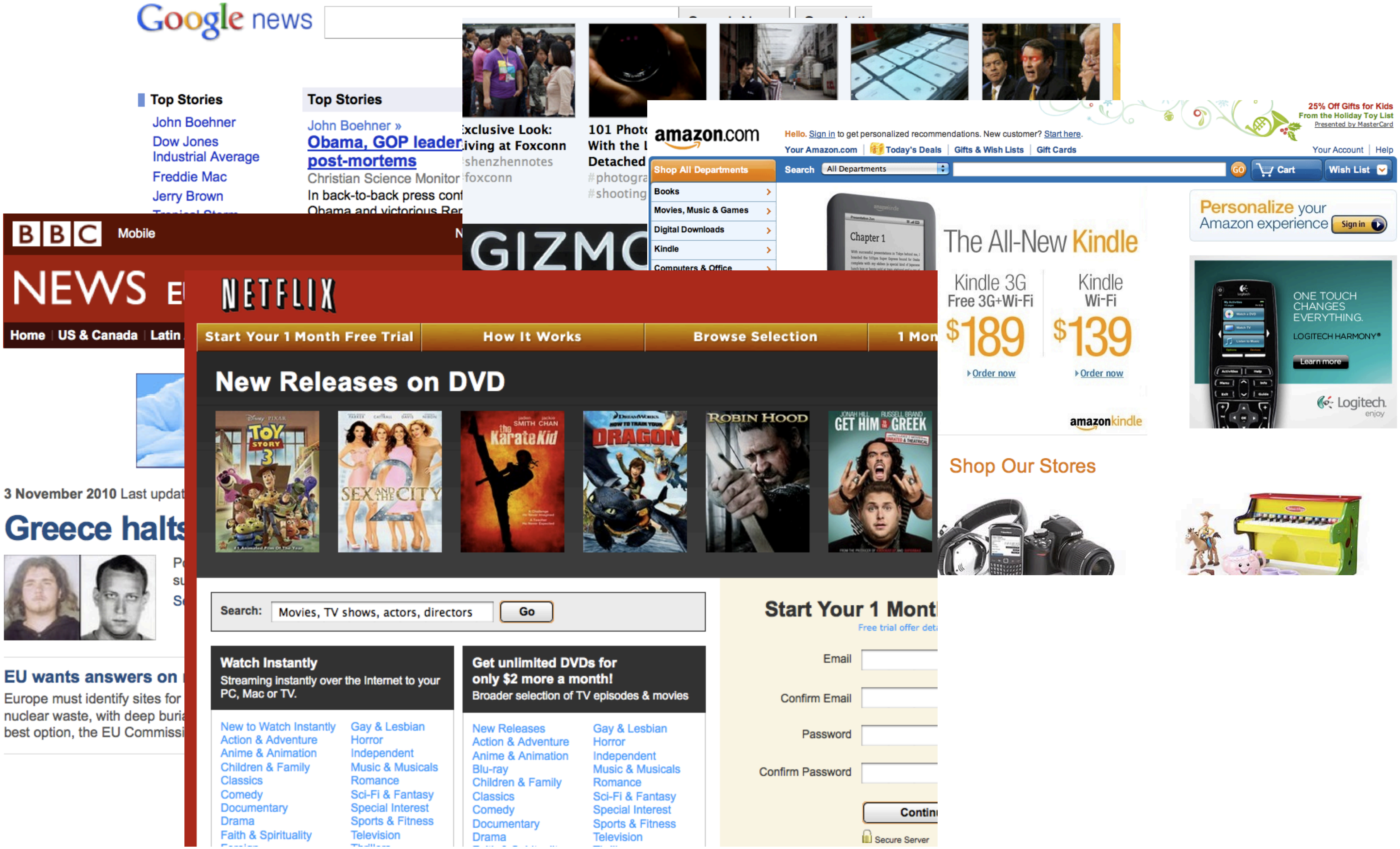
Modeling User Interests



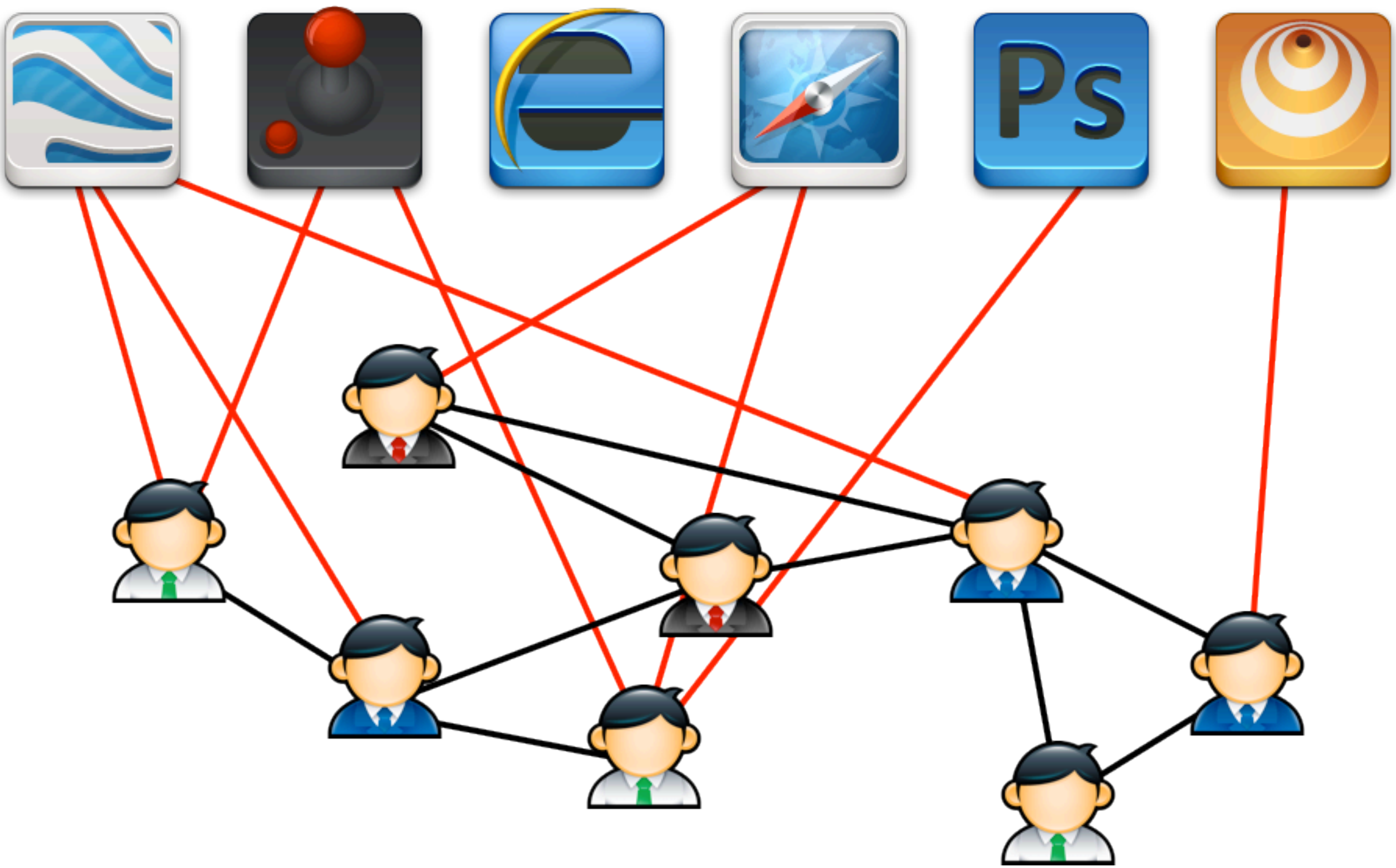


Dating	Baseball	Celebrity	Health	Jobs	Finance
women	League	Snooki	skin	job	financial
men	baseball	Tom	body	career	Thomson
dating	basketball,	Cruise	fingers	business	chart
singles	doublehead	Katie	cells	assistant	real
personals	Bergesen	Holmes	toes	hiring	Stock
seeking	Griffey	Pinkett	wrinkle	part-time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

Multi-domain Personalization



Graph Factorization: Social Network

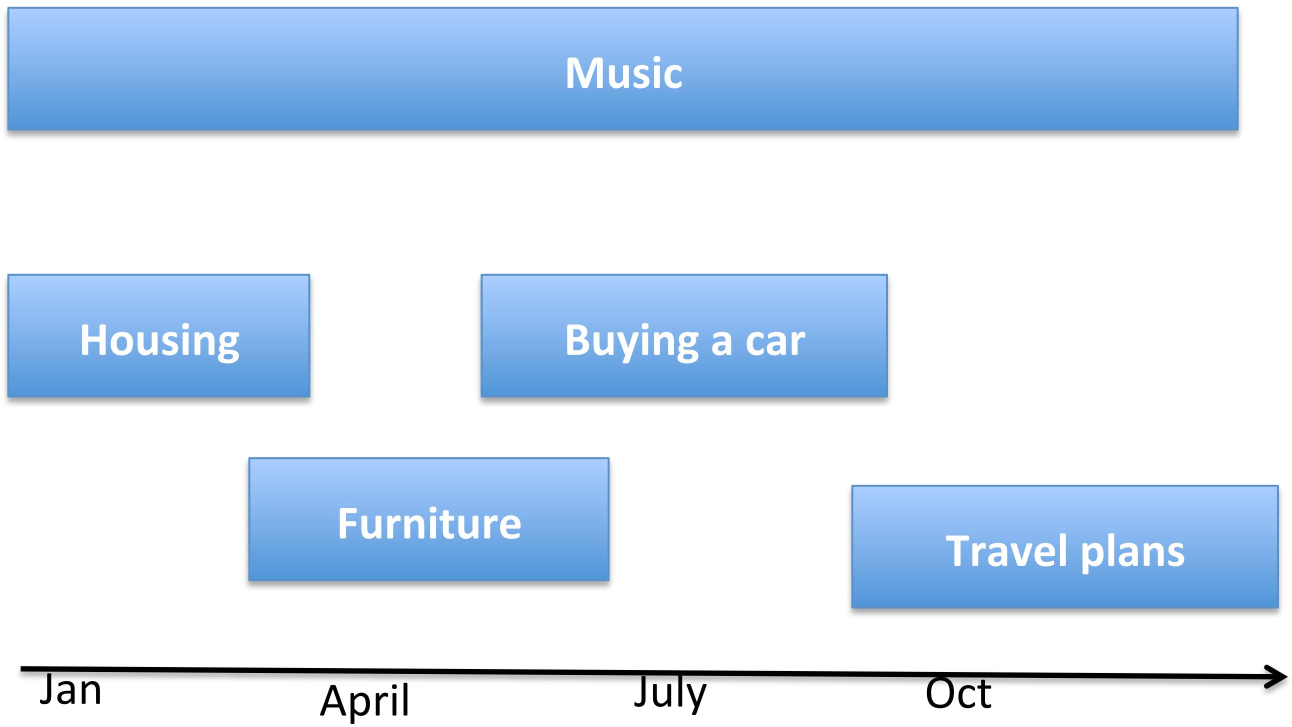


Application

Tracking Users Interest

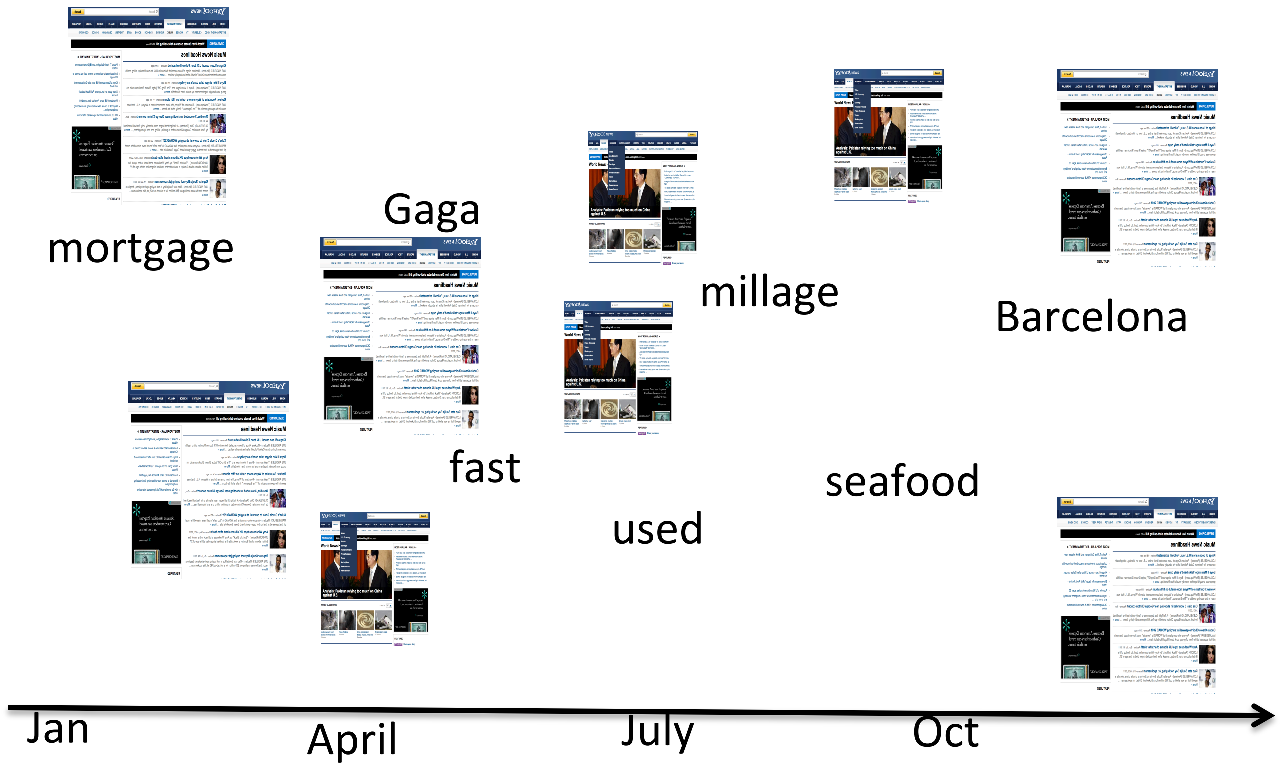
Characterizing User Interests

- Short term vs long-term



Characterizing User Interests

- Short term vs long-term
- Latent



Problem formulation

Input

- Queries issued by the user or tags of watched content
- Snippet of page examined by user
- Time stamp of each action (day resolution)

Output

- Users' daily distribution over interests
- Dynamic interest representation
- Online and scalable inference
- Language independent

Flight
London
Hotel
weather

classes
registration
housing
rent

School
Supplies
Loan
semester

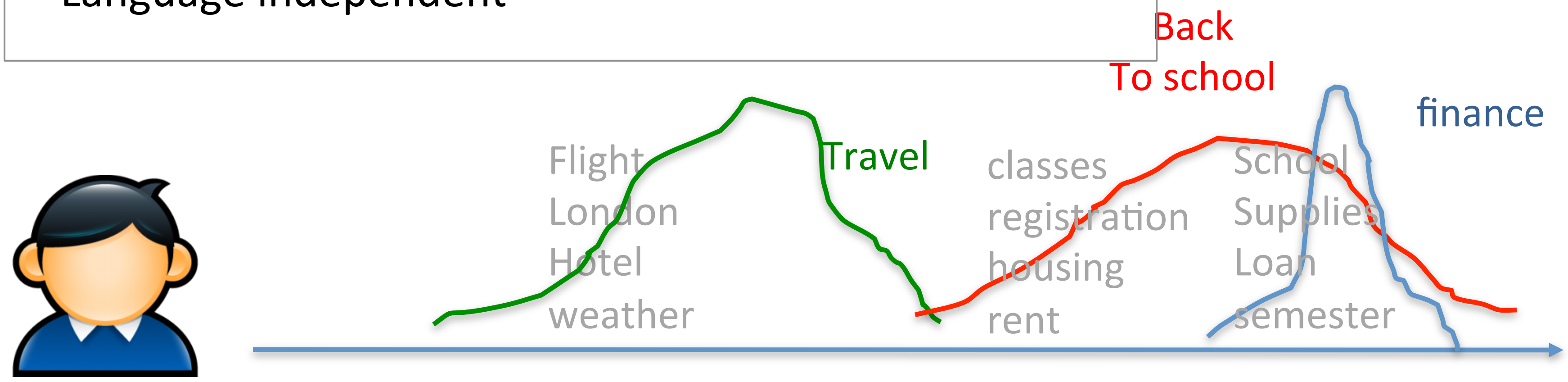
Problem formulation

Input

- Queries issued by the user or tags of watched content
- Snippet of page examined by user
- Time stamp of each action (day resolution)

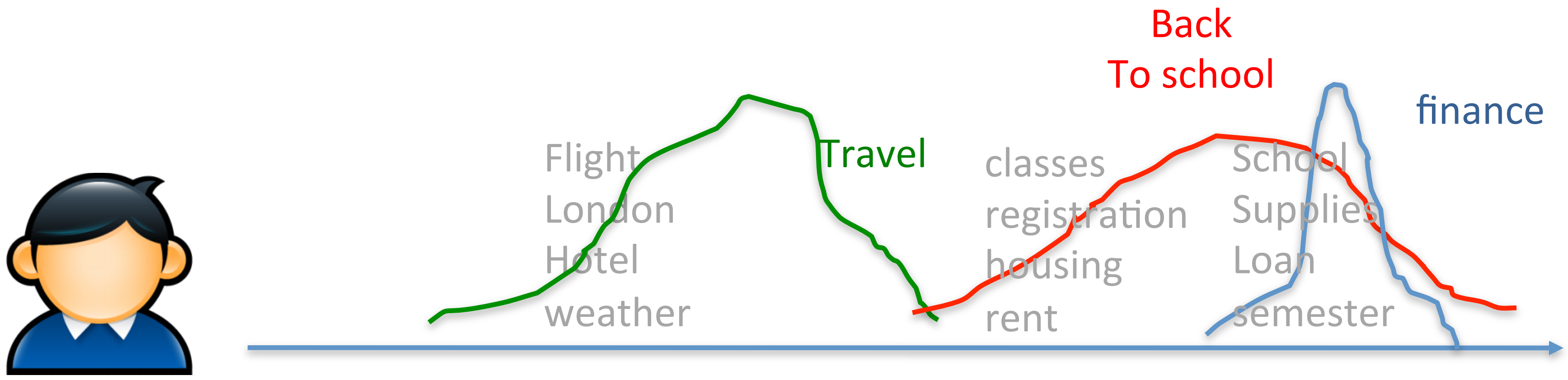
Output

- Users' daily distribution over interests
- Dynamic interest representation
- Online and scalable inference
- Language independent

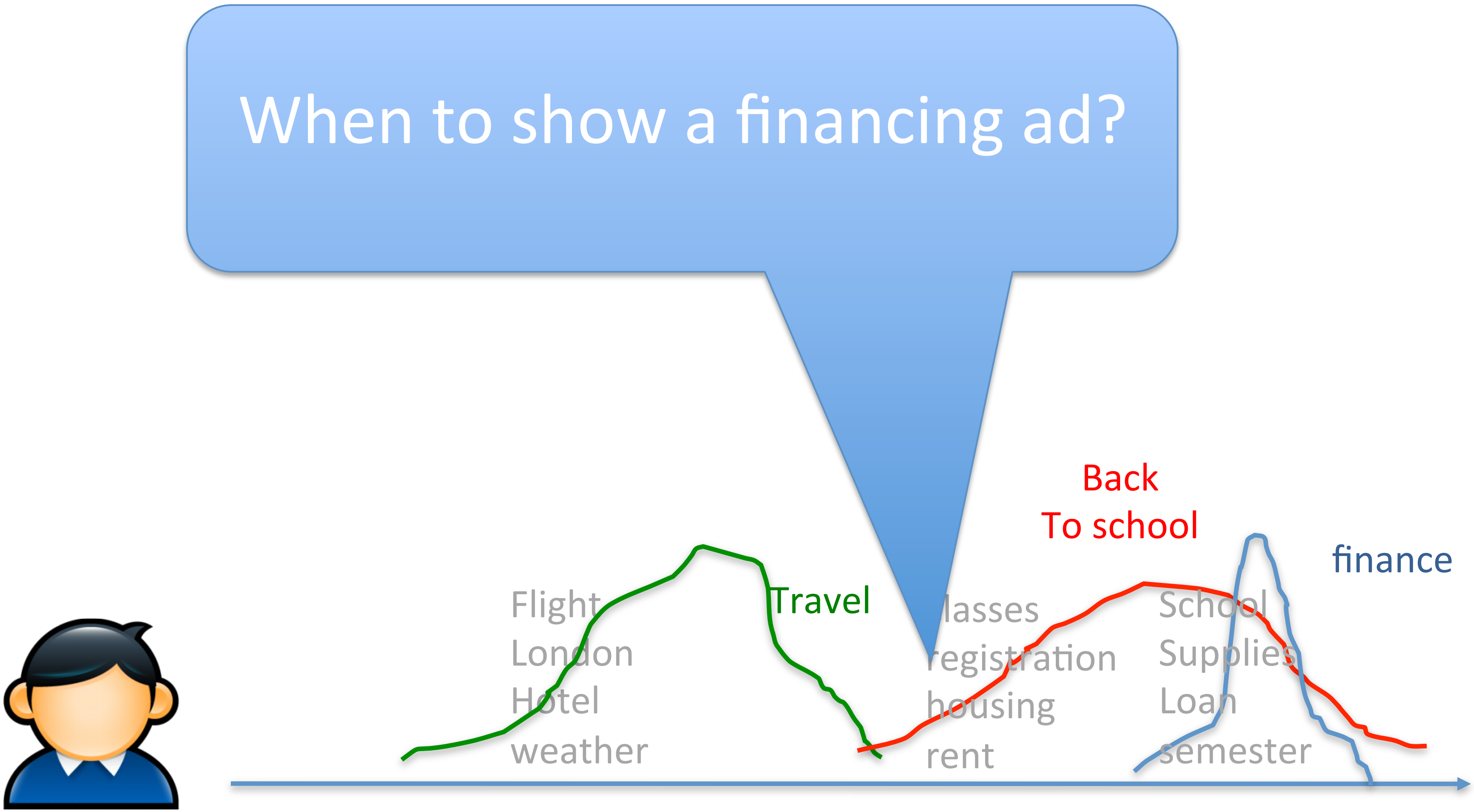


Problem formulation

When to show a financing ad?

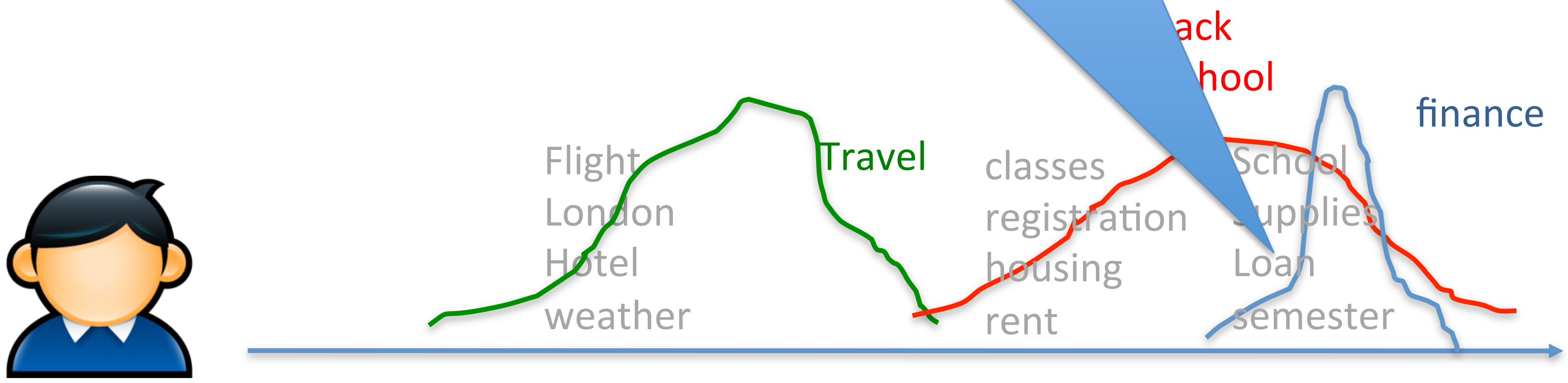


Problem formulation



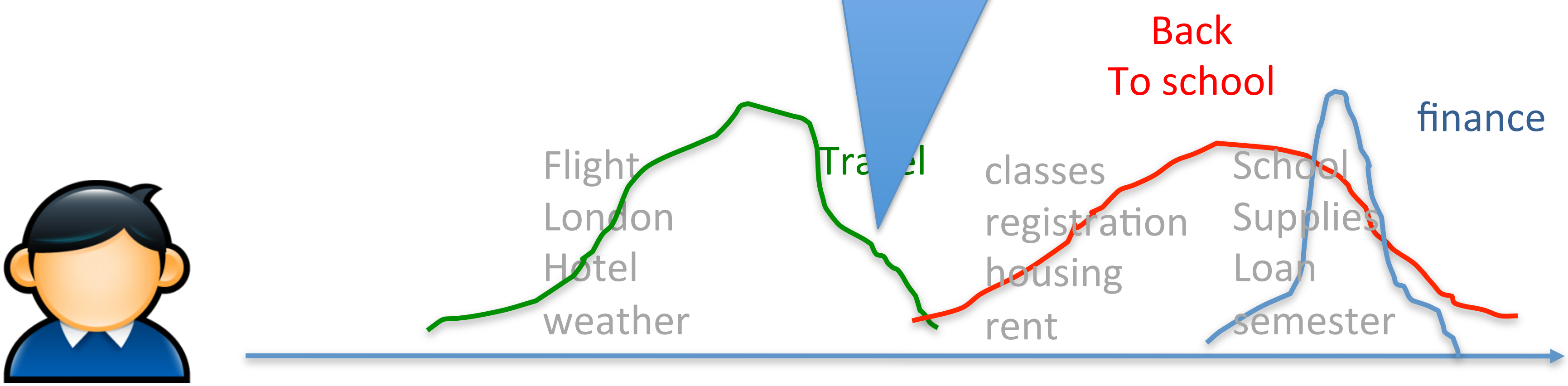
Problem formulation

When to show a financing ad?



Problem formulation

When to show a hotel ad?



Problem formulation

When to show a hotel ad?

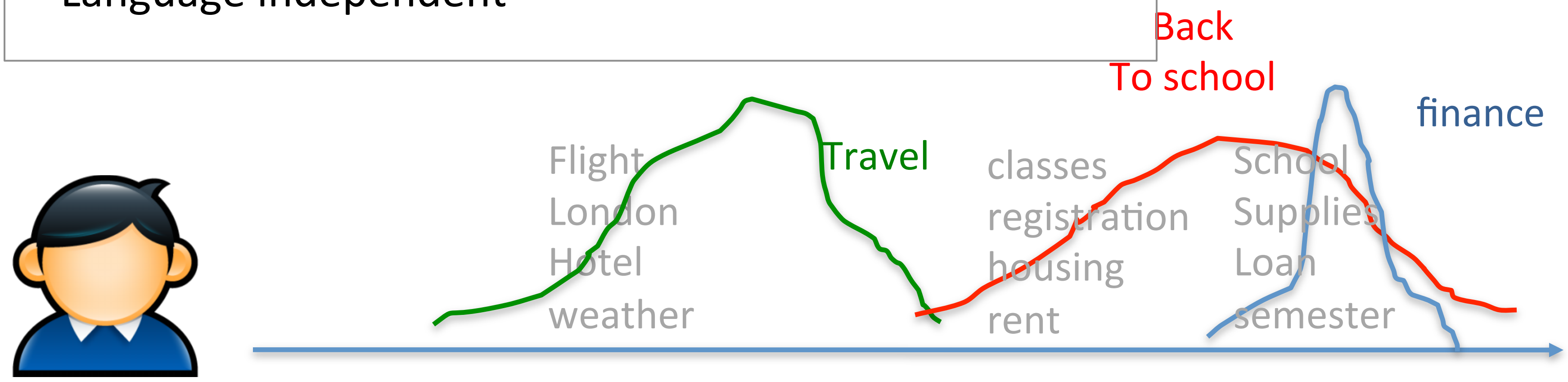
Problem formulation

Input

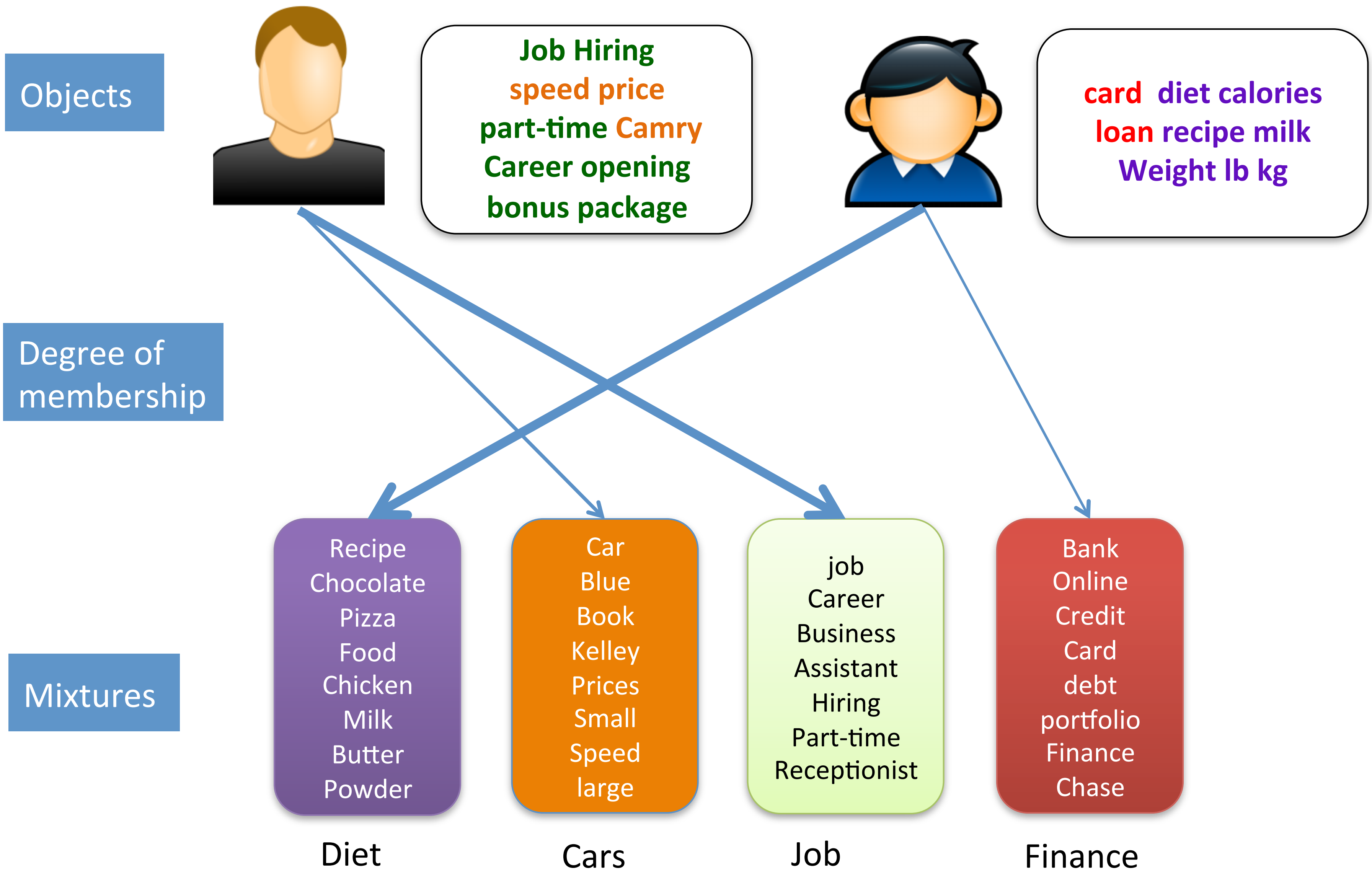
- Queries issued by the user or tags of watched content
- Snippet of page examined by user
- Time stamp of each action (day resolution)

Output

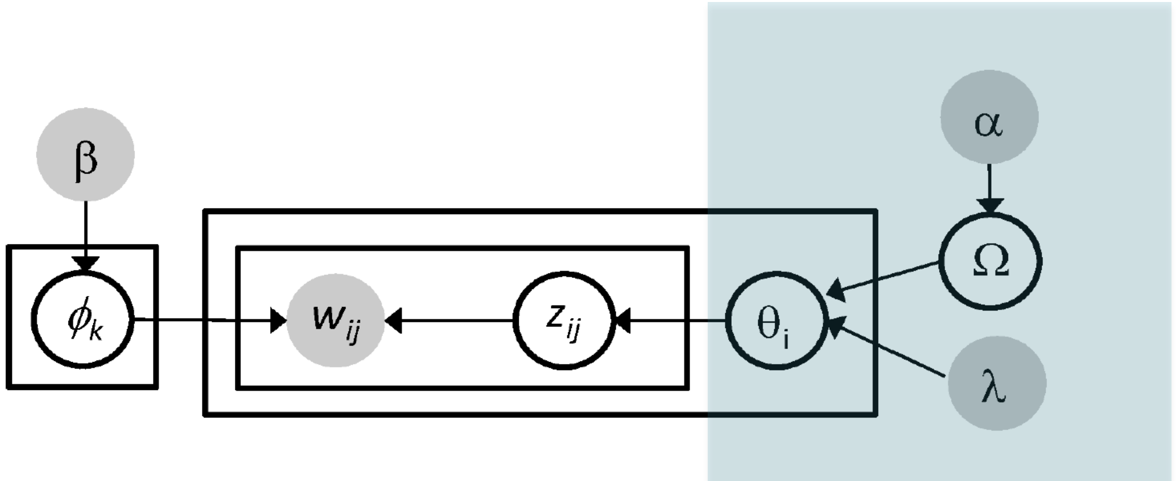
- Users' daily distribution over interests
- Dynamic interest representation
- Online and scalable inference
- Language independent



Mixed-Membership Formulation

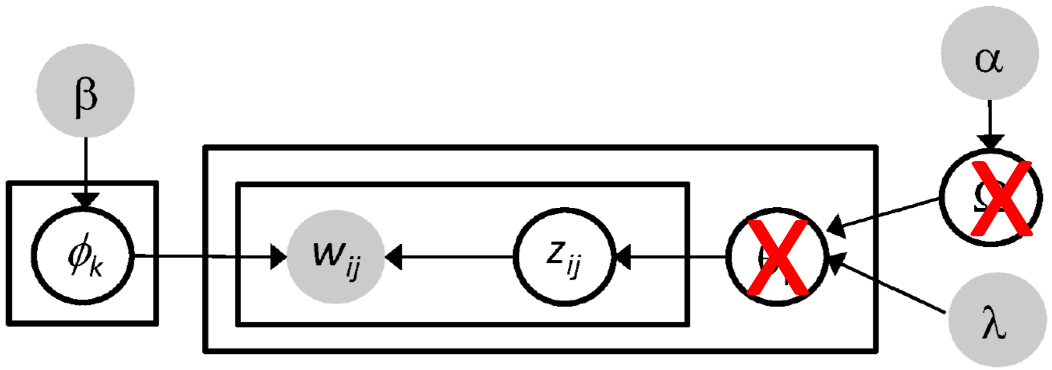


In Graphical Notation

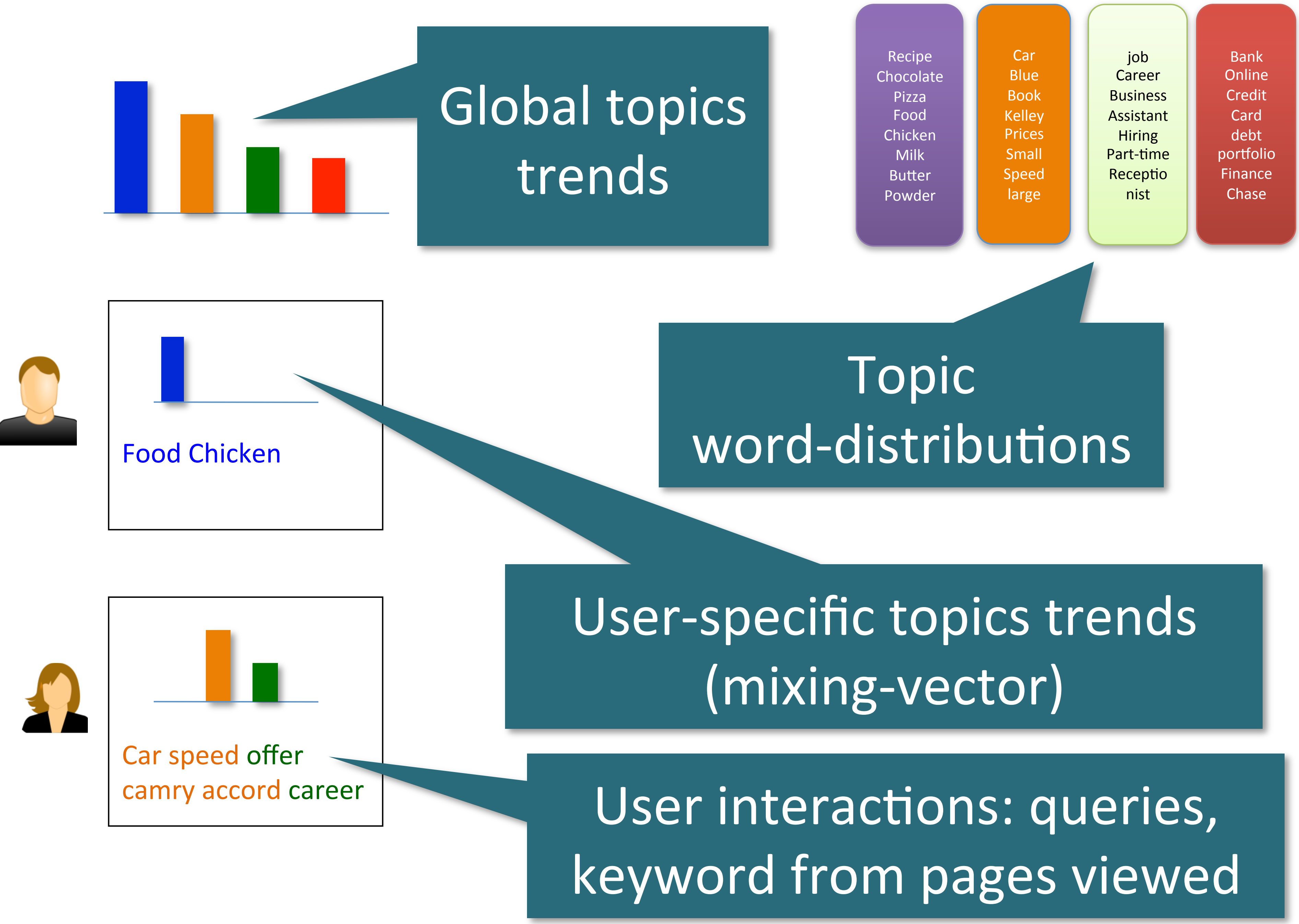


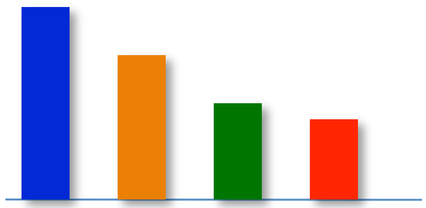
1. Draw once $\Omega|\alpha \sim \mathrm{Dir}(\alpha/K)$.
2. Draw each topic $\phi_k|\beta \sim \mathrm{Dir}(\beta)$.
3. For each user i:
   (a) Draw topic proportions $\theta_i|\lambda, \Omega \sim \mathrm{Dir}(\lambda\Omega)$.
   (b) For each word
      (a) Draw a topic $z_{ij}|\theta_d \sim \mathrm{Mult}(\theta_i)$.
      (b) Draw a word $w_{ij}|z_{ij}, \phi \sim \mathrm{Multi}(\phi_{z_{ij}})$.

In Polya-Urn Representation



- Collapse multinomial variables: θ, Ω
- Fixed-dimensional Hierarchal Polya-Urn representation
  - Chinese restaurant franchise

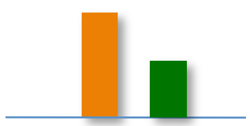

Global topics trends
Recipe Chocolate Pizza Food Chicken Milk Butter Powder
Car Blue Book Kelley Prices Small Speed large
job Career Business Assistant Hiring Part-time Receptio nist
Bank Online Credit Card debt portfolio Finance Chase
Topic word-distributions
Food Chicken
User-specific topics trends (mixing-vector)
Car speed offer camry accord career
User interactions: queries, keyword from pages viewed



Recipe	Car	job	Bank
Chocolate	Blue	Career	Online
Pizza	Book	Business	Credit
Food	Kelley	Assistant	Card
Chicken	Prices	Hiring	debt
Milk	Small	Part-time	portfolio
Butter	Speed	Receptio	Finance
Powder	large	nist	Chase

Food Chicken

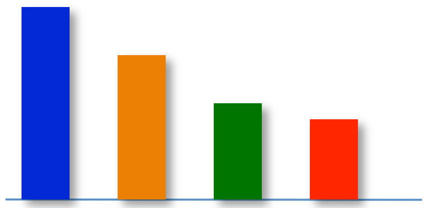
....

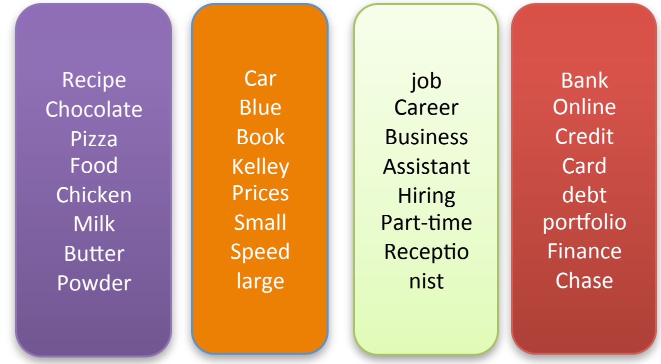


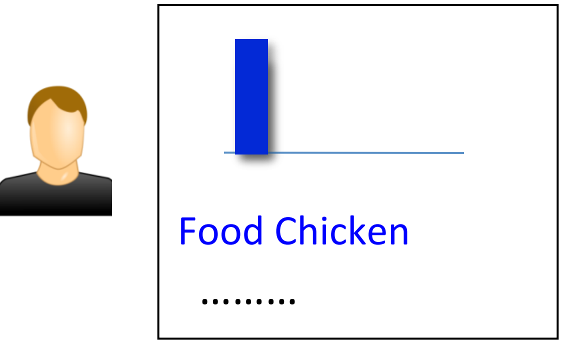
Car speed offer
camry accord career

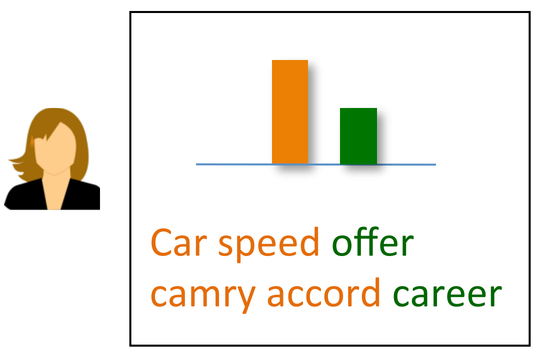
Generative Process

- For each user interaction
  - Choose an intent from local distribution
    - Sample word from the topic's word-distribution
  - Choose a new intent $\propto \lambda$
    - Sample a new intent from the global distribution
      - Sample word from the new topic word-distribution



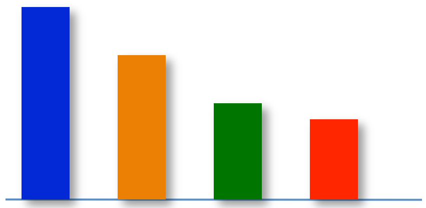


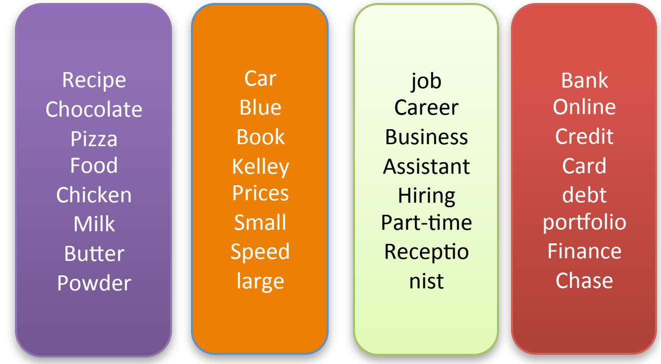


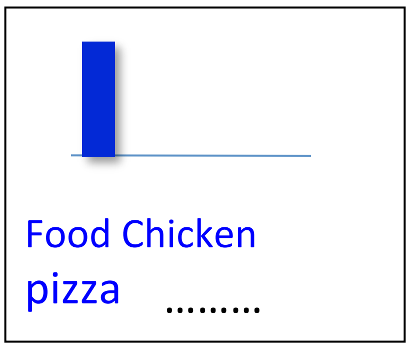


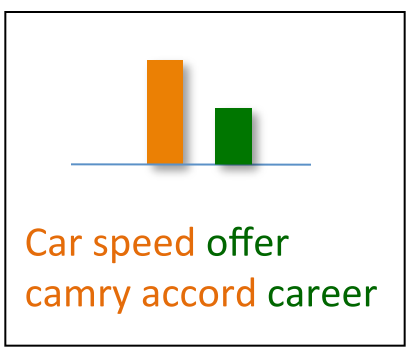
Generative Process

- For each user interaction
  - Choose an intent from local distribution
    - Sample word from the topic's word-distribution
  - Choose a new intent $\propto \lambda$
    - Sample a new intent from the global distribution
      - Sample word from the new topic word-distribution



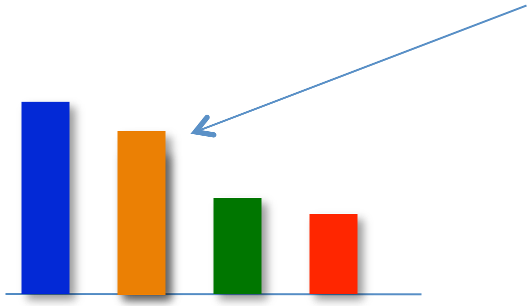


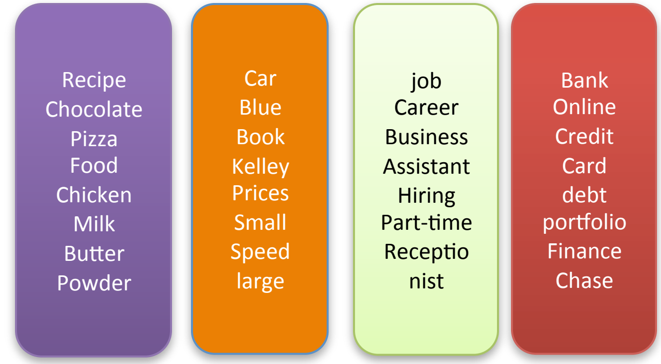


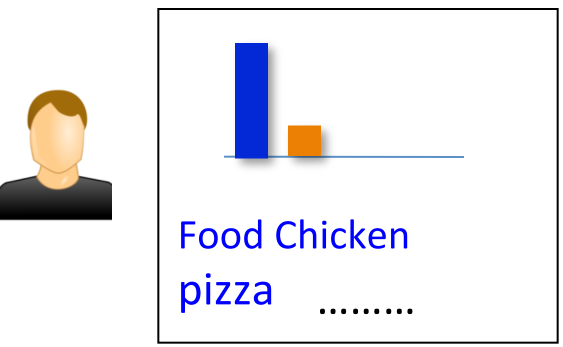


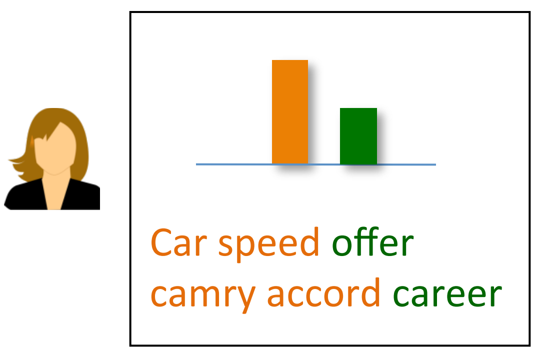
Generative Process

- For each user interaction
  - Choose an intent from local distribution
    - Sample word from the topic's word-distribution
  - Choose a new intent $\propto \lambda$
    - Sample a new intent from the global distribution
      - Sample word from the new topic word-distribution



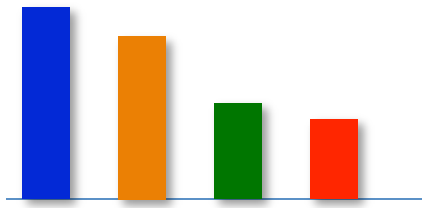






Generative Process

- For each user interaction
    - Choose an intent from local distribution
        - Sample word from the topic's word-distribution
    - Choose a new intent $\propto \lambda$
        - Sample a new intent from the global distribution
            - Sample word from the new topic word-distribution

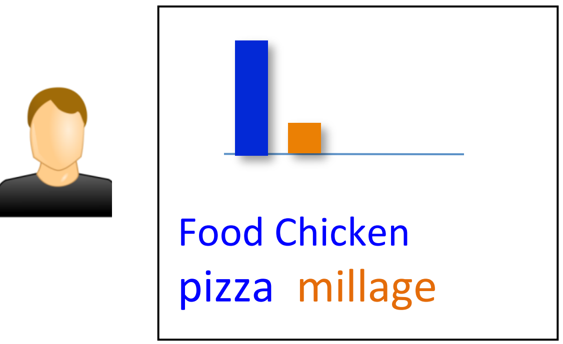


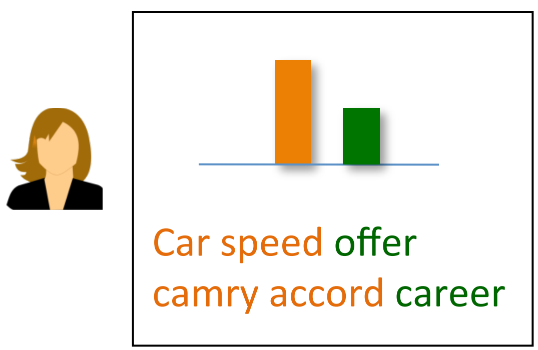
Recipe Chocolate Pizza Food Chicken Milk Butter Powder

Car Blue Book Kelley Prices Small Speed large

job Career Business Assistant Hiring Part-time Receptio nist

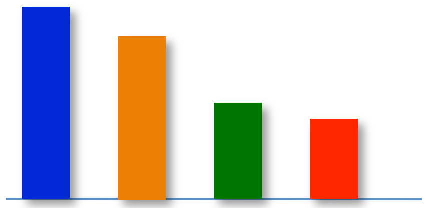
Bank Online Credit Card debt portfolio Finance Chase





Generative Process

- For each user interaction
  - Choose an intent from local distribution
    - Sample word from topic's word-distribution
  - Choose a new intent $\propto \lambda$
    - Sample a new intent from the global distribution
      - Sample from word the new topic word-distribution

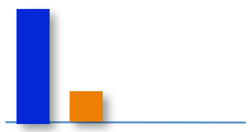


Recipe
Chocolate
Pizza
Food
Chicken
Milk
Butter
Powder

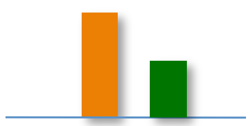
Car
Blue
Book
Kelley
Prices
Small
Speed
large

job
Career
Business
Assistant
Hiring
Part-time
Receptio
nist

Bank
Online
Credit
Card
debt
portfolio
Finance
Chase



Food Chicken
pizza millage

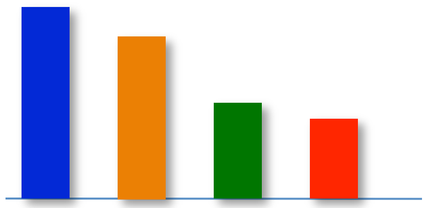


Car speed offer
camry accord career

Problems

- Static Model
- Does not evolve user's interests
- Does not evolve the global trend of interests
- Does not evolve interest's distribution over terms

At time t



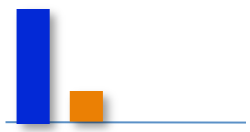
At time t+1

Recipe
Chocolate
Pizza
Food
Chicken
Milk
Butter
Powder

Car
Blue
Book
Kelley
Prices
Small
Speed
large

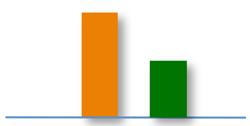
job
Career
Business
Assistant
Hiring
Part-time
Receptio
nist

Bank
Online
Credit
Card
debt
portfolio
Finance
Chase



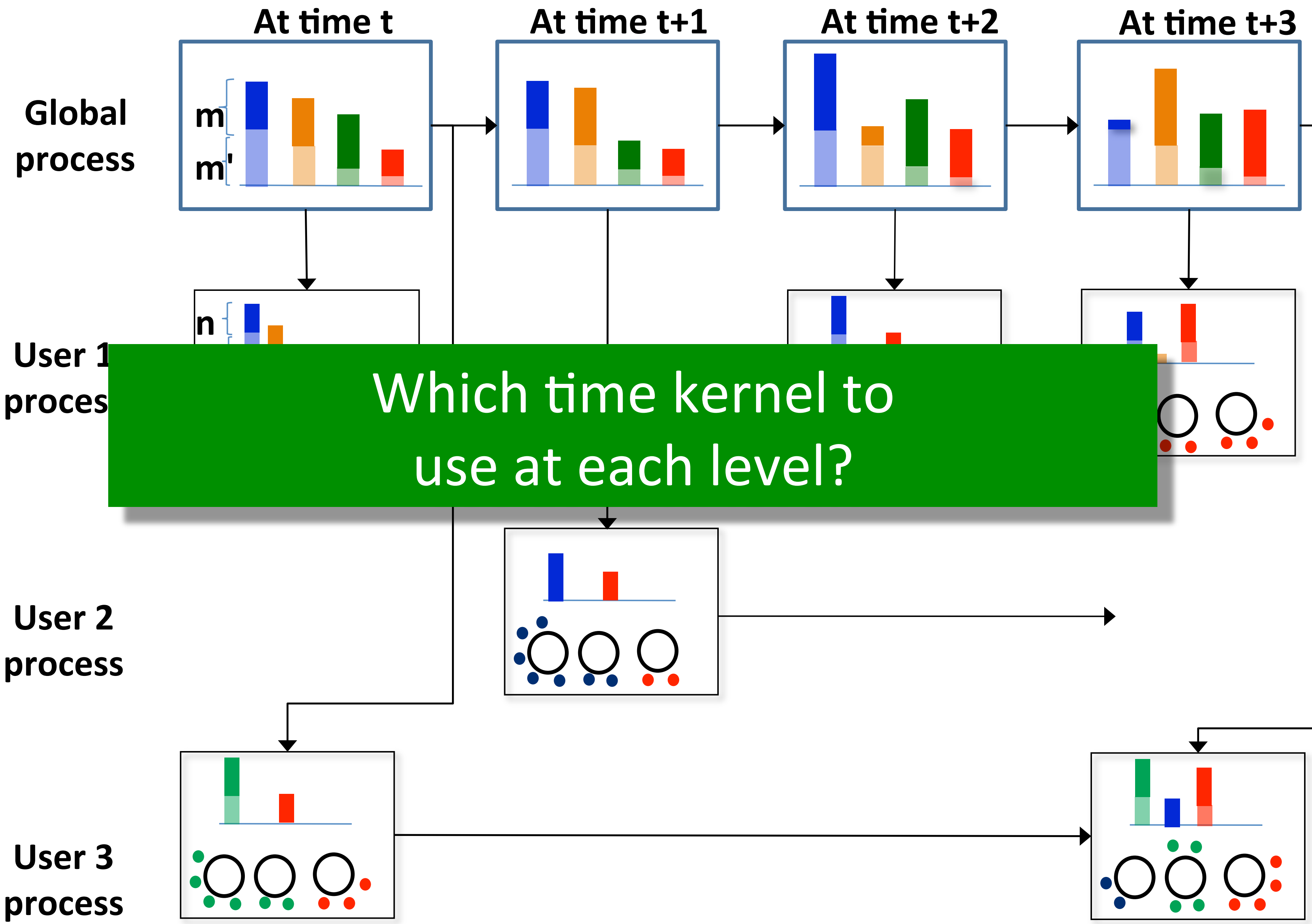
Food Chicken
pizza millage

Build a dynamic model

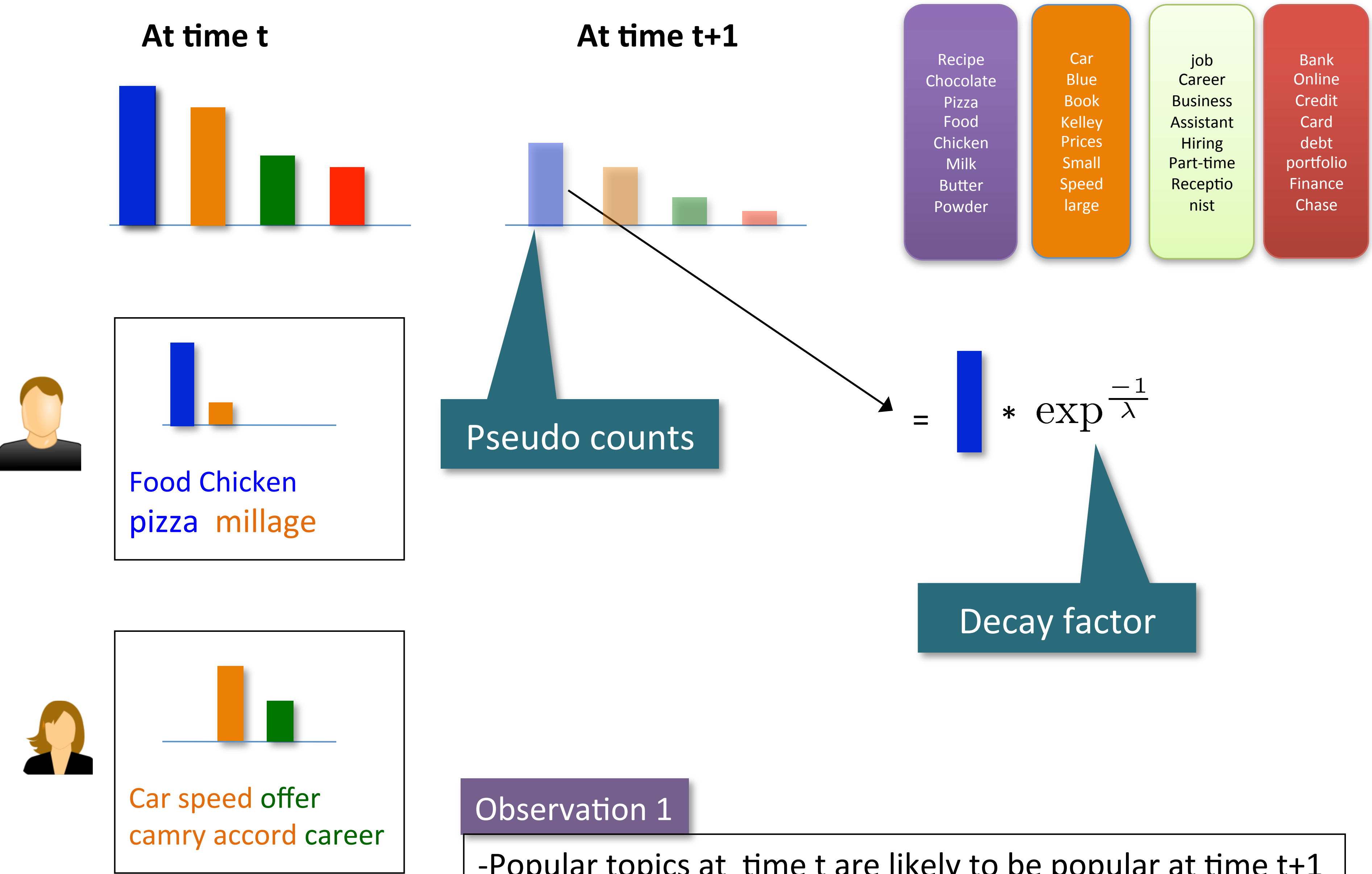


Car speed offer
camry accord career

Connect each level using a RCRP



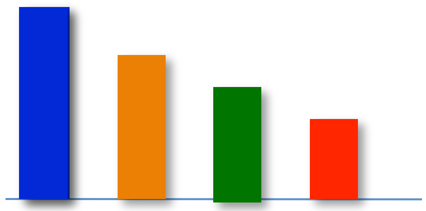
At time t
At time t+1
At time t+2
At time t+3
Global process
m
m'
n
User 1 process
User 2 process
User 3 process
Which time kernel to use at each level?



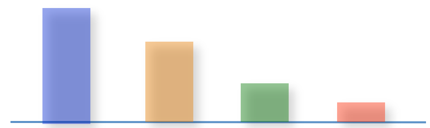
Observation 1

- Popular topics at time t are likely to be popular at time t+1
- $\phi_{k,t+1}$ is likely to smoothly evolve from $\phi_{k,t}$

At time t



At time t+1

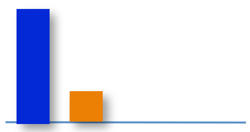


Recipe
Chocolate
Pizza
Food
Chicken
Milk
Butter
Powder

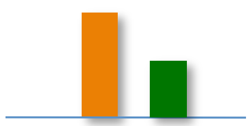
Car
Blue
Book
Kelley
Prices
Small
Speed
large

job
Career
Business
Assistant
Hiring
Part-time
Receptio
nist

Bank
Online
Credit
Card
debt
portfolio
Finance
Chase



Food Chicken
pizza millage



Car speed offer
camry accord career

Car
Altima
Accord
Book
Kelley
Prices
Small
Speed

Intuition

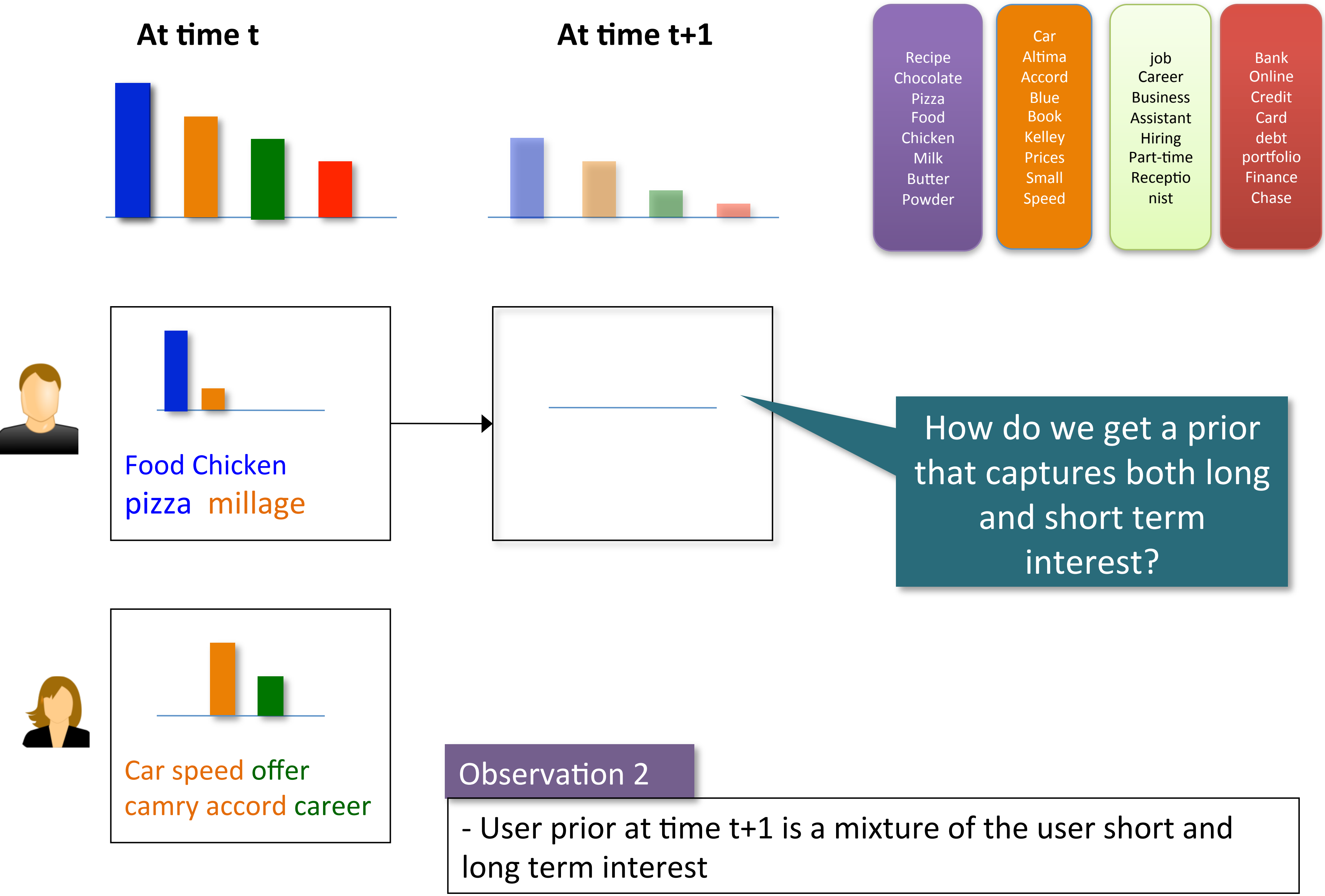
Captures current trend of the car industry
(new release for e.g.)

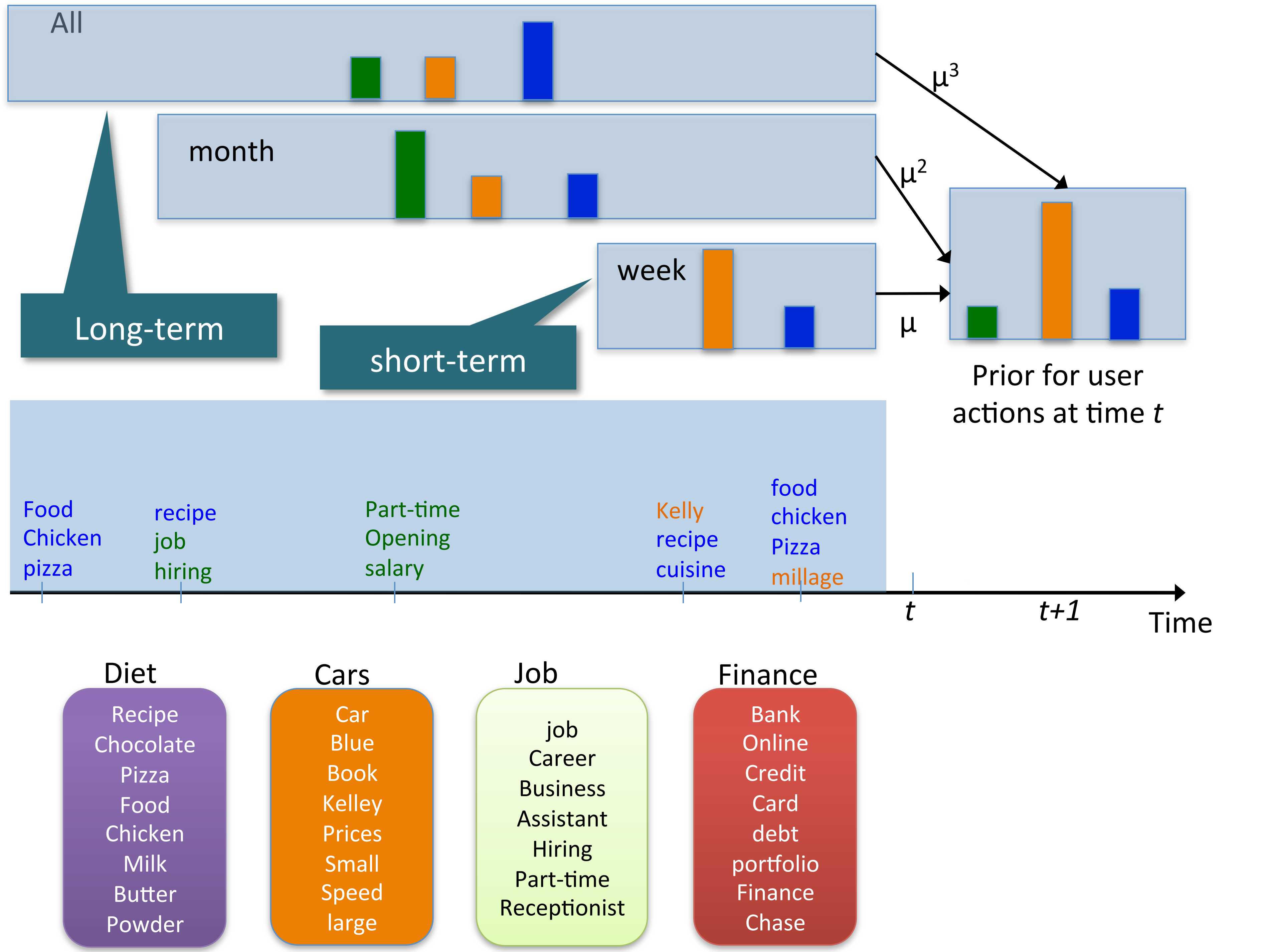
$\phi_{k,t}$ $\qquad$ $\phi_{k,t+1} \sim \text{Dir}(\tilde{\beta}_{k,t+1})$

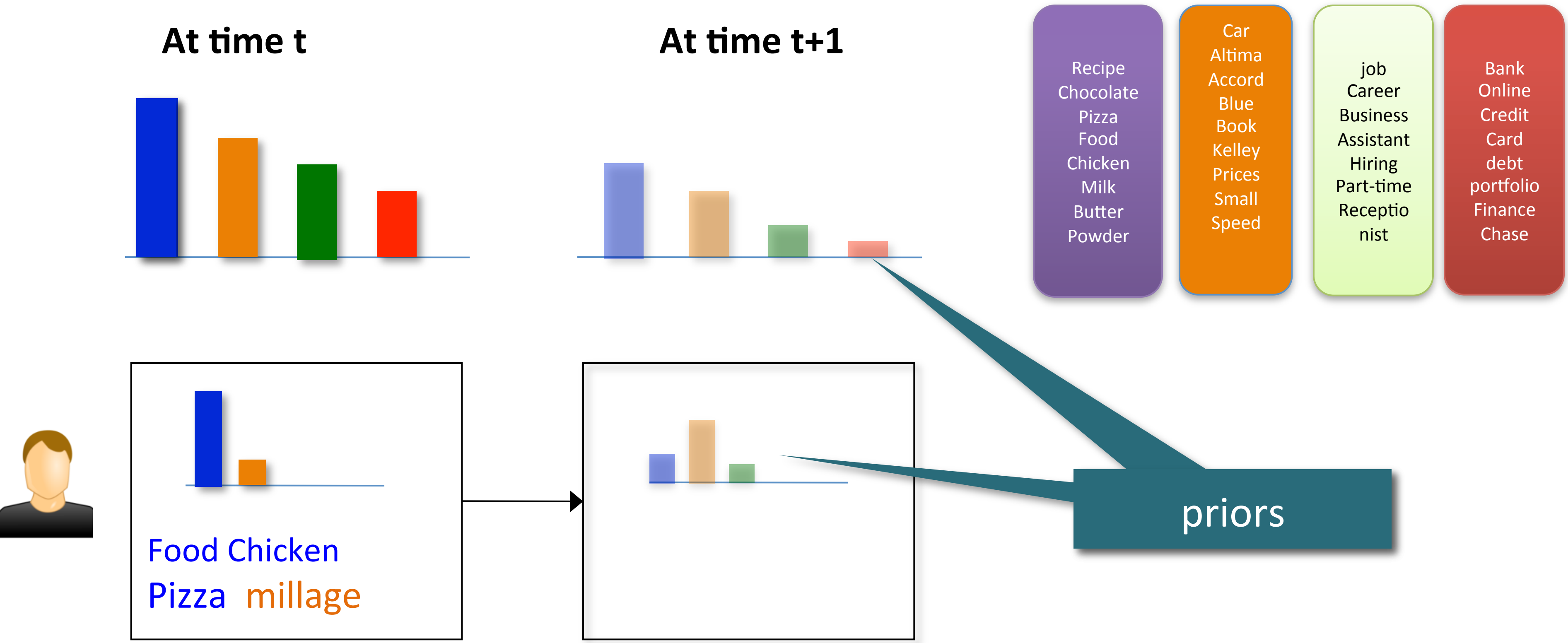
Observation 1

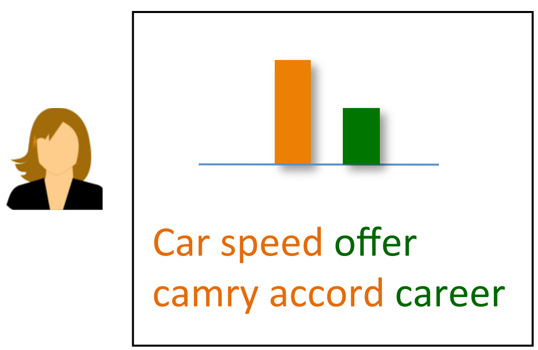
-Popular topics at time t are likely to be popular at time t+1

– $\phi_{k,t+1}$ **is likely to smoothly evolve** from $\phi_{k,t}$


At time t
At time t+1
Recipe Chocolate Pizza Food Chicken Milk Butter Powder
Car Altima Accord Blue Book Kelley Prices Small Speed
job Career Business Assistant Hiring Part-time Receptio nist
Bank Online Credit Card debt portfolio Finance Chase
Food Chicken pizza millage
How do we get a prior that captures both long and short term interest?
Car speed offer camry accord career
Observation 2
- User prior at time t+1 is a mixture of the user short and long term interest

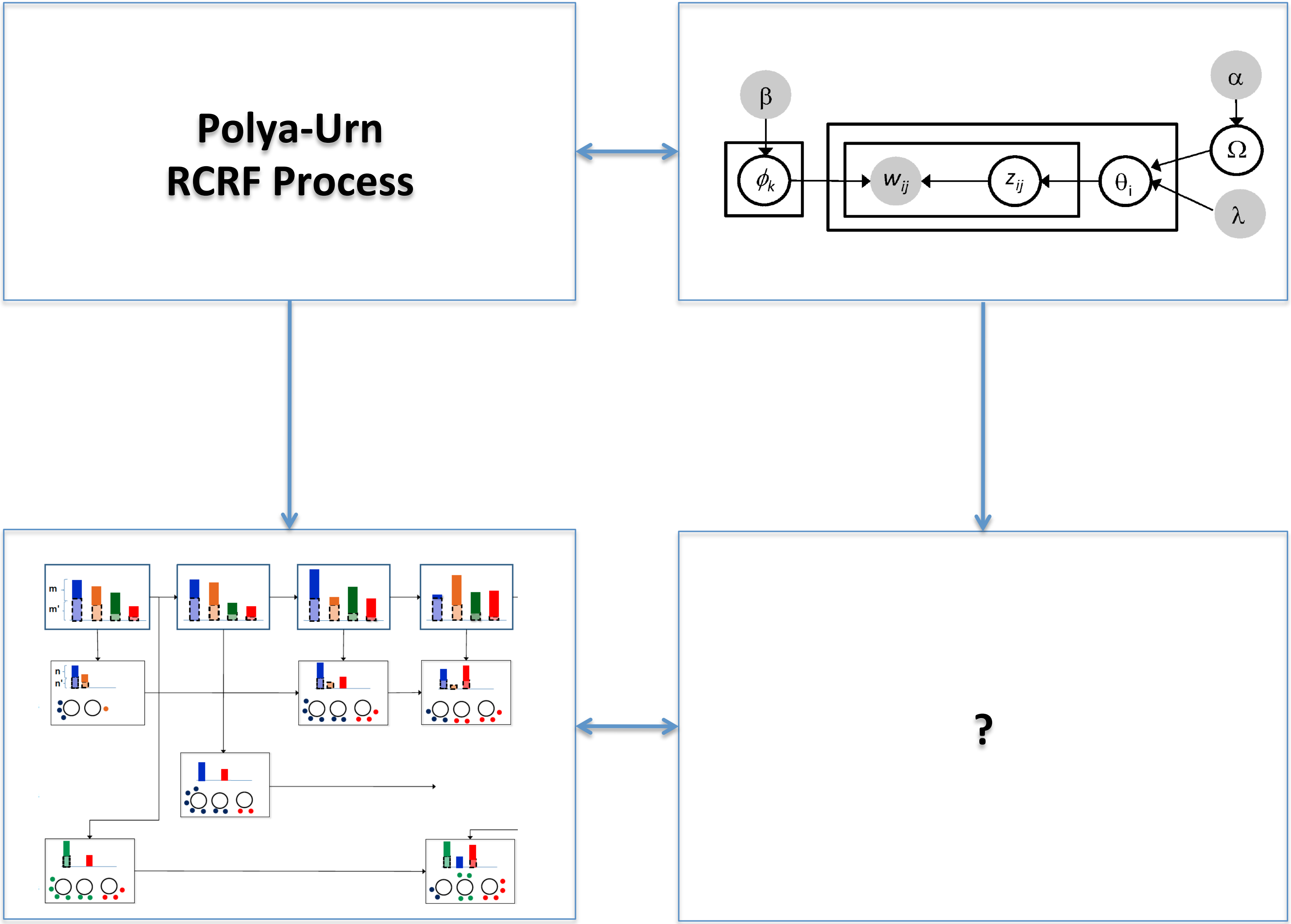

All
month
week
μ^3
μ^2
μ
Long-term
short-term
Prior for user actions at time *t*
Food
Chicken
pizza
recipe
job
hiring
Part-time
Opening
salary
Kelly
recipe
cuisine
food
chicken
Pizza
millage
t
t+1
Time
Diet
Recipe
Chocolate
Pizza
Food
Chicken
Milk
Butter
Powder
Cars
Car
Blue
Book
Kelley
Prices
Small
Speed
large
Job
job
Career
Business
Assistant
Hiring
Part-time
Receptionist
Finance
Bank
Online
Credit
Card
debt
portfolio
Finance
Chase





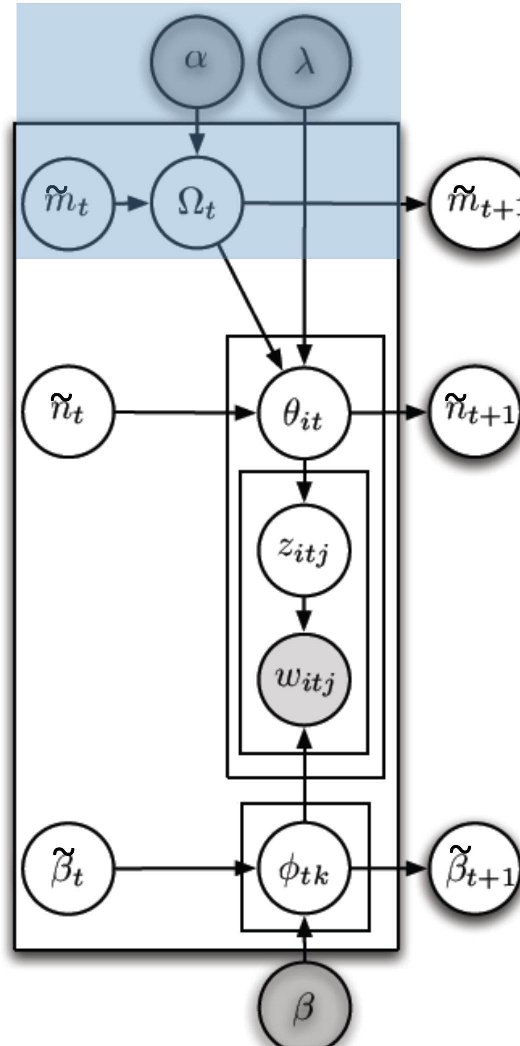
Generative Process

- For each user interaction
  - Choose an intent from local distribution
    - Sample word from the topic's word-distribution
  - Choose a new intent $\propto \lambda$
    - Sample a new intent from the global distribution
      - Sample word from the new topic word-distribution

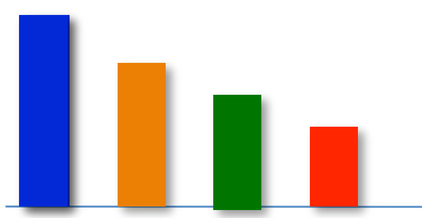

Polya-Urn
RCRF Process
β
α
Ω
λ
?

Simplified Graphical Model

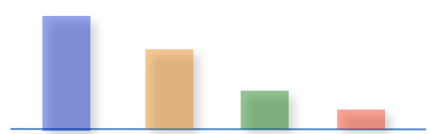
1. Draw once $\Omega^t|\alpha, \tilde{m}^t \sim \mathrm{Dir}\left(\tilde{\mathbf{m}}^t + \alpha/K\right)$.
2. Draw each topic, $\phi_k^t|\beta, \tilde{\beta}_k^t \sim \mathrm{Dir}(\tilde{\beta}_k^t + \beta)$.
3. For each user i:
   (a) Draw topic proportions $\theta_i^t|\lambda, \Omega^t, \tilde{\mathbf{n}}_i^t \sim \mathrm{Dir}(\lambda\Omega^t + \tilde{\mathbf{n}}_i^t)$.
   (b) For each word
      (a) Draw a topic $z_{in}^t|\theta_i^t \sim \mathrm{Mult}(\theta_i^t)$.
      (b) Draw a word $w_{in}^t|z_{ij}^t, \phi^t \sim \mathrm{Multi}(\phi^t_{z_{ij}^t})$.



At time t



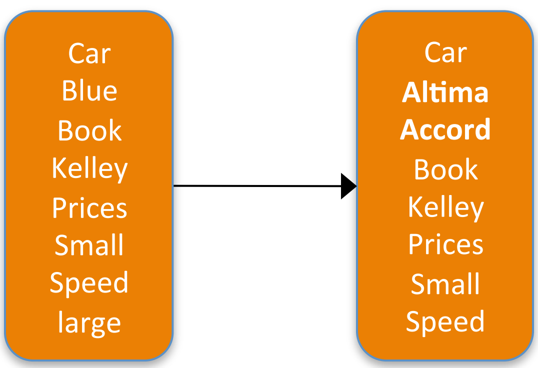
At time t+1

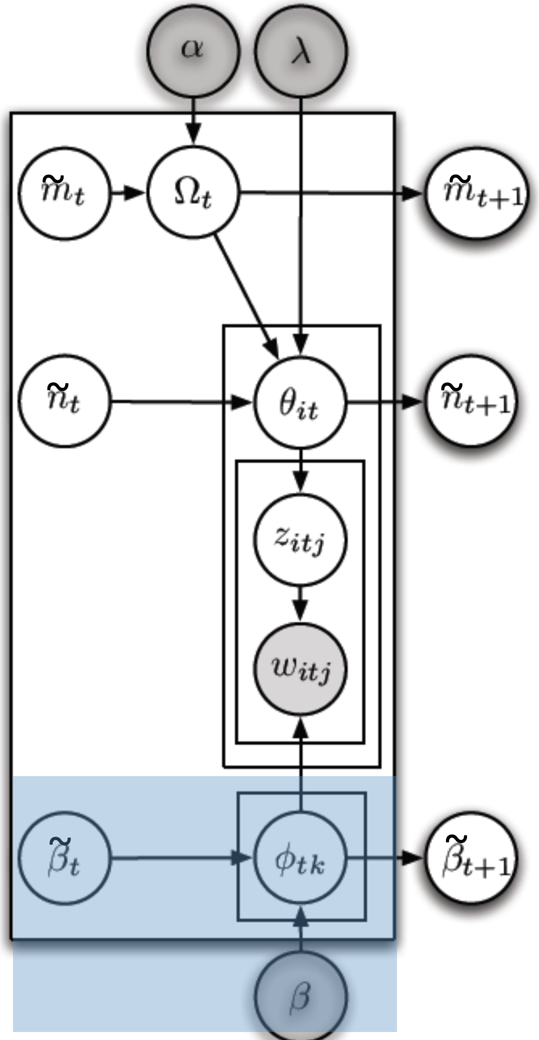


Simplified Graphical Model

1. Draw once $\Omega^t|\alpha, \tilde{m}^t \sim \mathrm{Dir}\left(\tilde{\mathbf{m}}^t + \alpha/K\right)$.
2. Draw each topic, $\phi^t_k|\beta, \tilde{\beta}^t_k \sim \mathrm{Dir}(\tilde{\beta}^t_k + \beta)$.
3. For each user i:
   - (a) Draw topic proportions $\theta^t_i|\lambda, \Omega^t, \tilde{\mathbf{n}}^t_i \sim \mathrm{Dir}(\lambda\Omega^t + \tilde{\mathbf{n}}^t_i)$.
   - (b) For each word
     - (a) Draw a topic $z^t_{in}|\theta^t_i \sim \mathrm{Mult}(\theta^t_i)$.
     - (b) Draw a word $w^t_{in}|z^t_{ij}, \phi^t \sim \mathrm{Multi}(\phi^t_{z^t_{ij}})$.

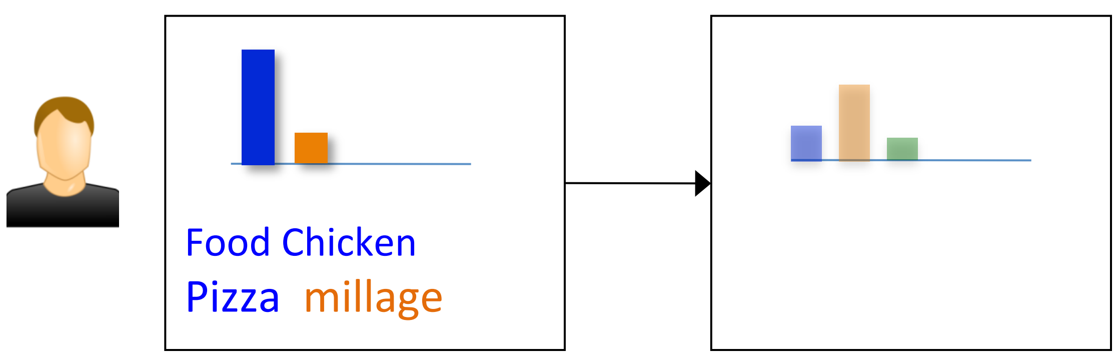
$$\tilde{\beta}^t_{kw} = \sum_{h=1}^{t-1} \exp^{\frac{h-t}{\kappa 0}} n^h_{kw}$$

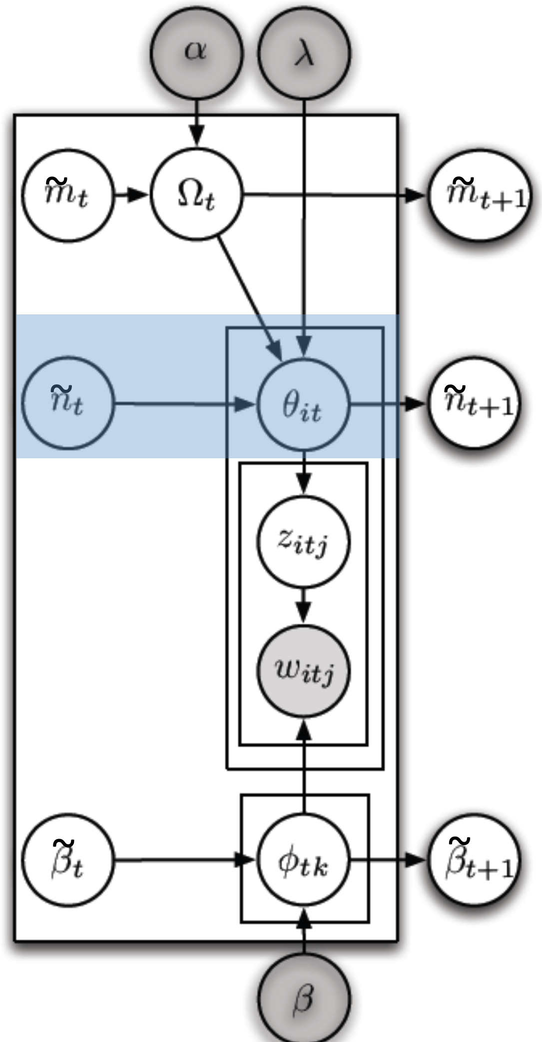




Simplified Graphical Model

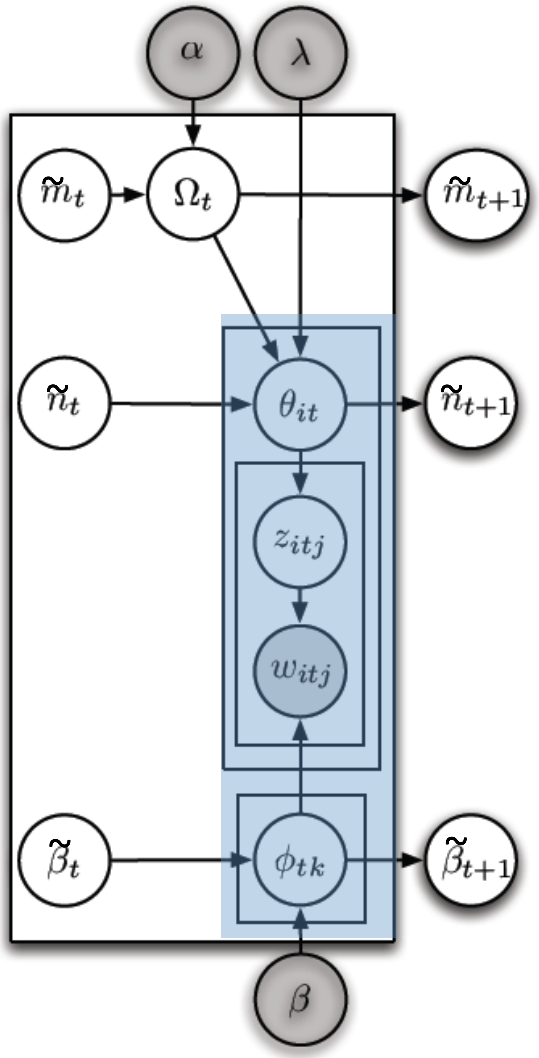
1. Draw once $\Omega^t|\alpha, \tilde{m}^t \sim \text{Dir}\left(\tilde{\mathbf{m}}^t + \alpha/K\right)$.
2. Draw each topic, $\phi^t_k|\beta, \tilde{\beta}^t_k \sim \text{Dir}(\tilde{\beta}^t_k + \beta)$.
3. For each user i:
   (a) Draw topic proportions $\theta^t_i|\lambda, \Omega^t, \tilde{\mathbf{n}}^t_i \sim \text{Dir}(\lambda\Omega^t + \tilde{\mathbf{n}}^t_i)$.
   (b) For each word
      (a) Draw a topic $z^t_{in}|\theta^t_i \sim \text{Mult}(\theta^t_i)$.
      (b) Draw a word $w^t_{in}|z^t_{ij}, \phi^t \sim \text{Multi}(\phi^t_{z^t_{ij}})$.





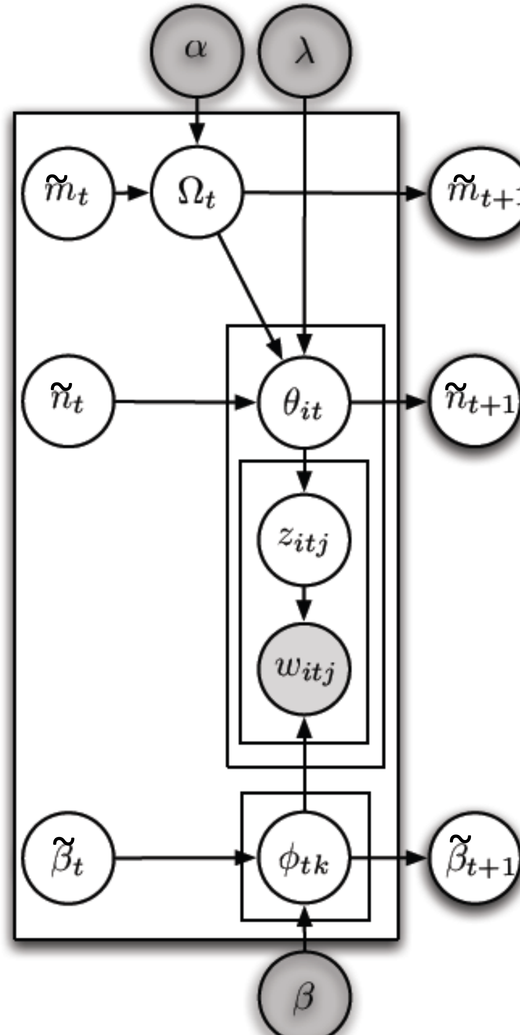
Simplified Graphical Model

1. Draw once $\Omega^t|\alpha, \tilde{m}^t \sim \text{Dir}\left(\tilde{\mathbf{m}}^t + \alpha/K\right)$.
2. Draw each topic, $\phi_k^t|\beta, \tilde{\beta}_k^t \sim \text{Dir}(\tilde{\beta}_k^t + \beta)$.
3. For each user i:
   - (a) Draw topic proportions $\theta_i^t|\lambda, \Omega^t, \tilde{\mathbf{n}}_i^t \sim \text{Dir}(\lambda\Omega^t + \tilde{\mathbf{n}}_i^t)$.
   - (b) For each word
     - (a) Draw a topic $z_{in}^t|\theta_i^t \sim \text{Mult}(\theta_i^t)$.
     - (b) Draw a word $w_{in}^t|z_{ij}^t, \phi^t \sim \text{Multi}(\phi^t_{z_{ij}^t})$.



Simplified Graphical Model

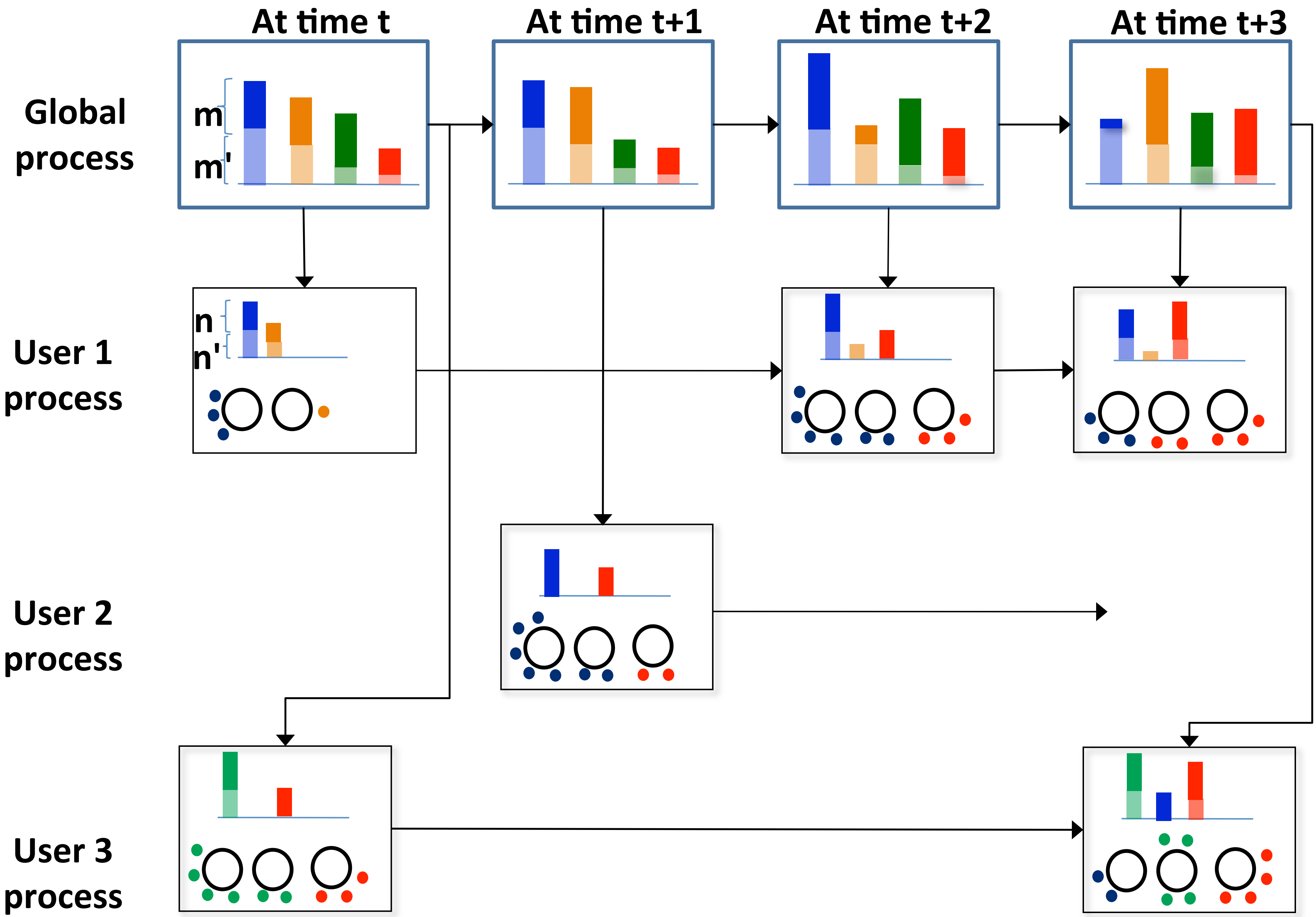
1. Draw once $\Omega^t|\alpha, \tilde{m}^t \sim \text{Dir}\left(\tilde{\mathbf{m}}^t + \alpha/K\right)$.
2. Draw each topic, $\phi_k^t|\beta, \tilde{\beta}_k^t \sim \text{Dir}(\tilde{\beta}_k^t + \beta)$.
3. For each user i:
    (a) Draw topic proportions $\theta_i^t|\lambda, \Omega^t, \tilde{\mathbf{n}}_i^t \sim \text{Dir}(\lambda\Omega^t + \tilde{\mathbf{n}}_i^t)$.
    (b) For each word
        (a) Draw a topic $z_{in}^t|\theta_i^t \sim \text{Mult}(\theta_i^t)$.
        (b) Draw a word $w_{in}^t|z_{ij}^t, \phi^t \sim \text{Multi}(\phi^t_{z_{ij}^t})$.



Topics evolve over time? ✔

User's intent evolve over time? ✔

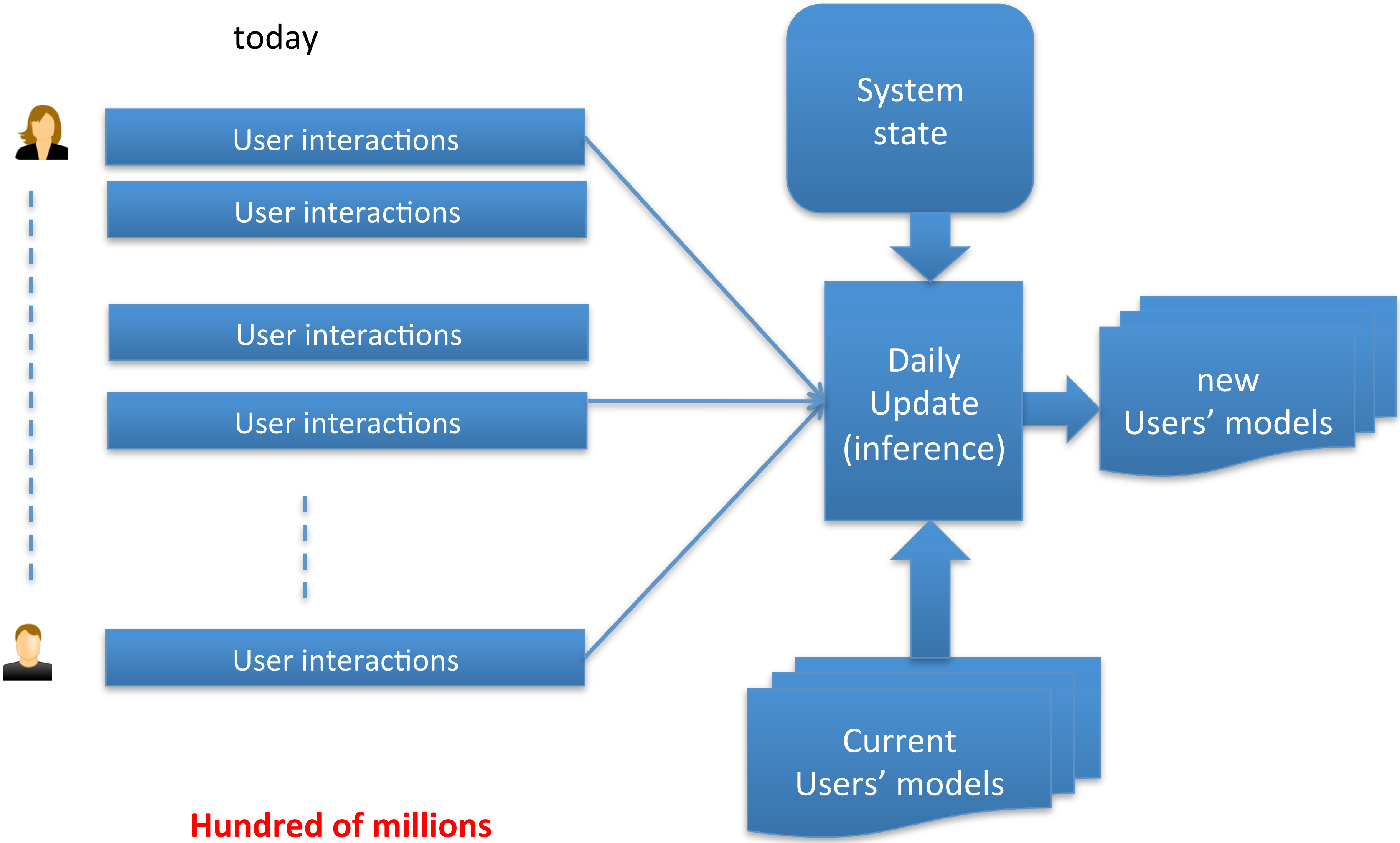
Capture long and term interests of users? ✔


At time t
At time t+1
At time t+2
At time t+3
Global process
m
m'
User 1 process
n
n'
User 2 process
User 3 process

Online Distributed Inference

Work Flow

Work Flow

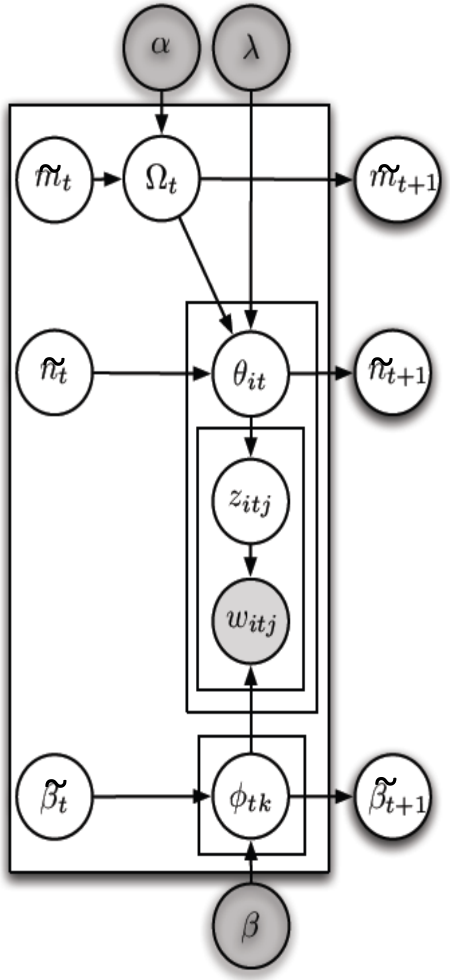

today
User interactions
User interactions
User interactions
User interactions
User interactions
System state
Daily Update (inference)
new Users' models
Current Users' models
Hundred of millions

Online Scalable Inference

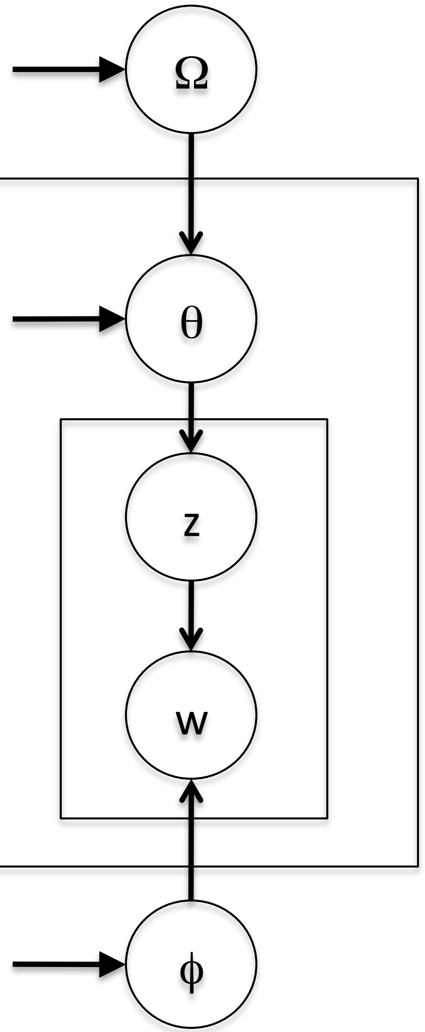
- Online algorithm
  - Greedy 1-particle filtering algorithm
  - Works well in practice
  - Collapse all multinomials except Ω_t
    - This makes distributed inference easier
  - At each time *t:*

$$P(\Omega^t, \mathbf{z}^t | \tilde{\mathbf{n}}^t, \tilde{\beta}^t, \tilde{\mathbf{m}}^t)$$

- Distributed scalable implementation
  - Used first part architecture as a subroutine
  - Added synchronous sampling capabilities

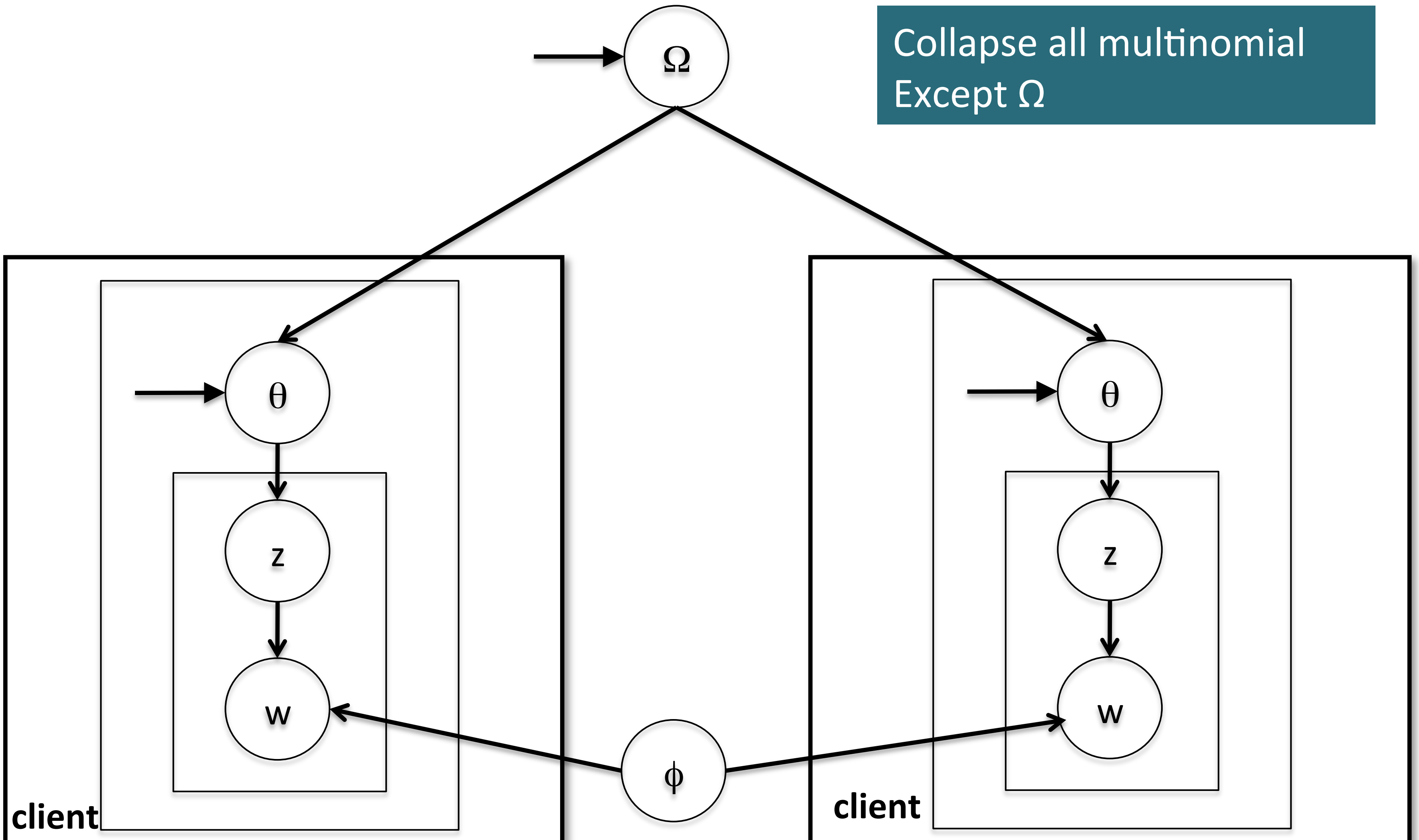


Distributed Inference (at time t)

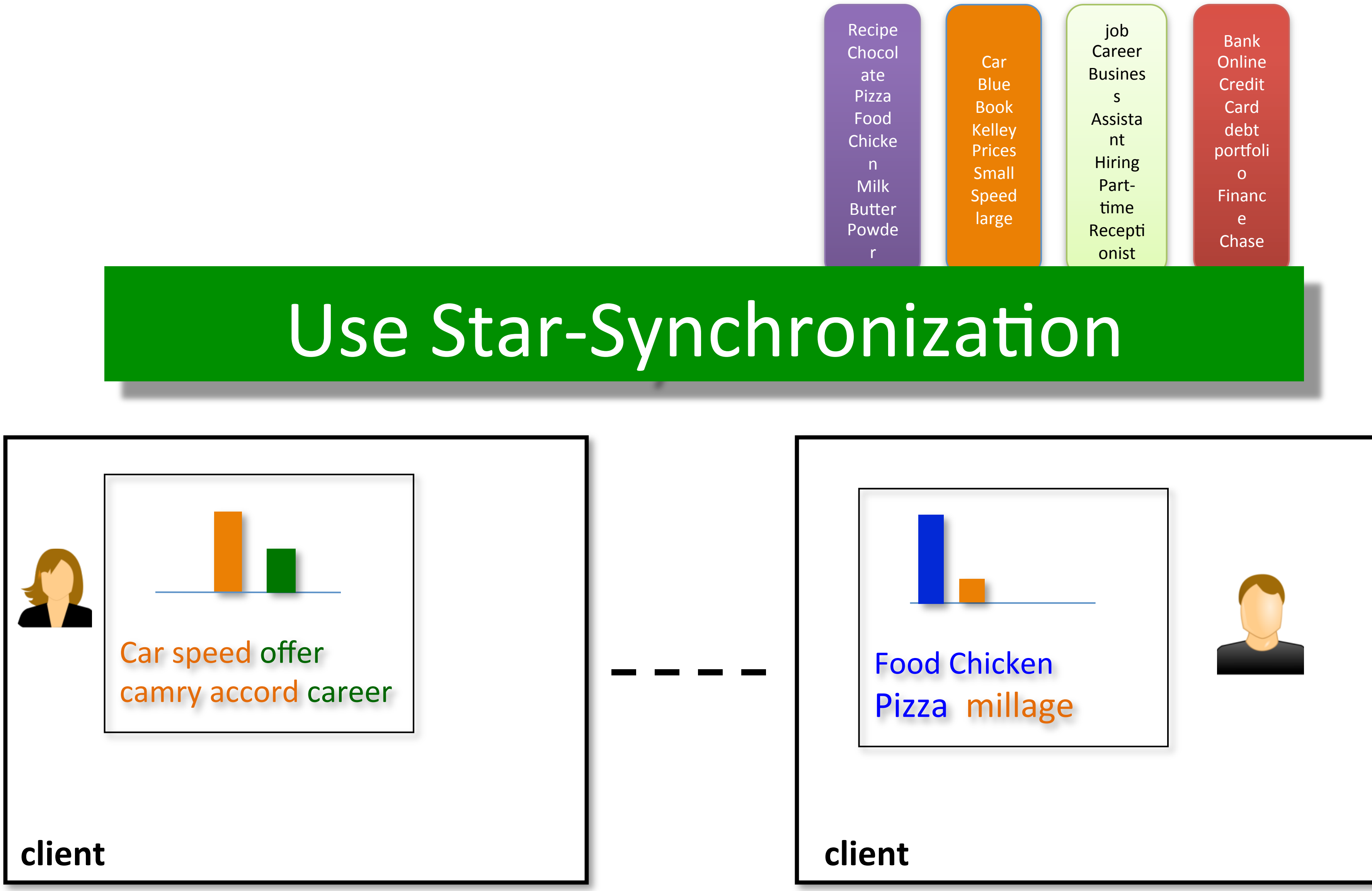


Distributed Inference (at time t)

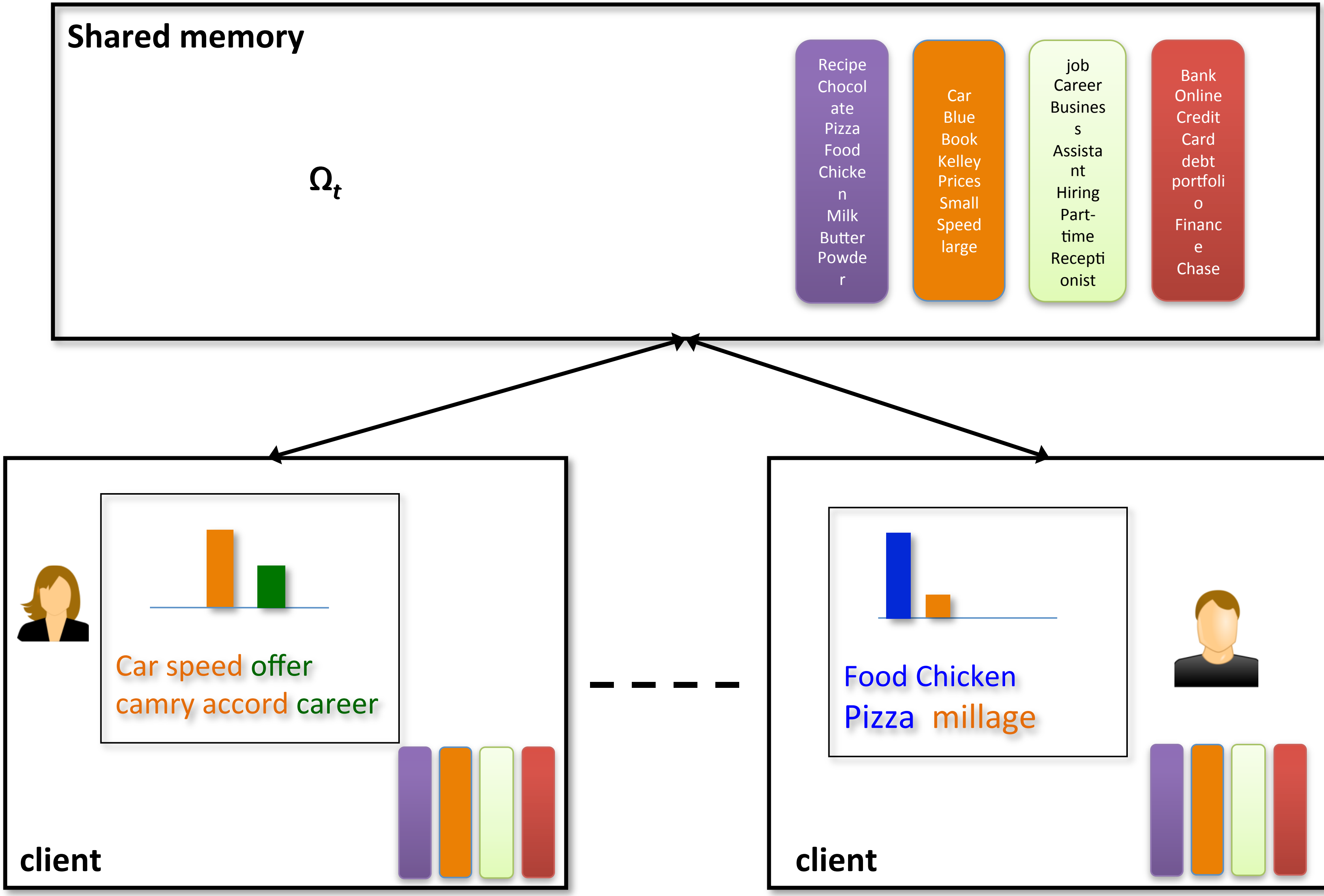
Collapse all multinomial
Except Ω



After collapsing



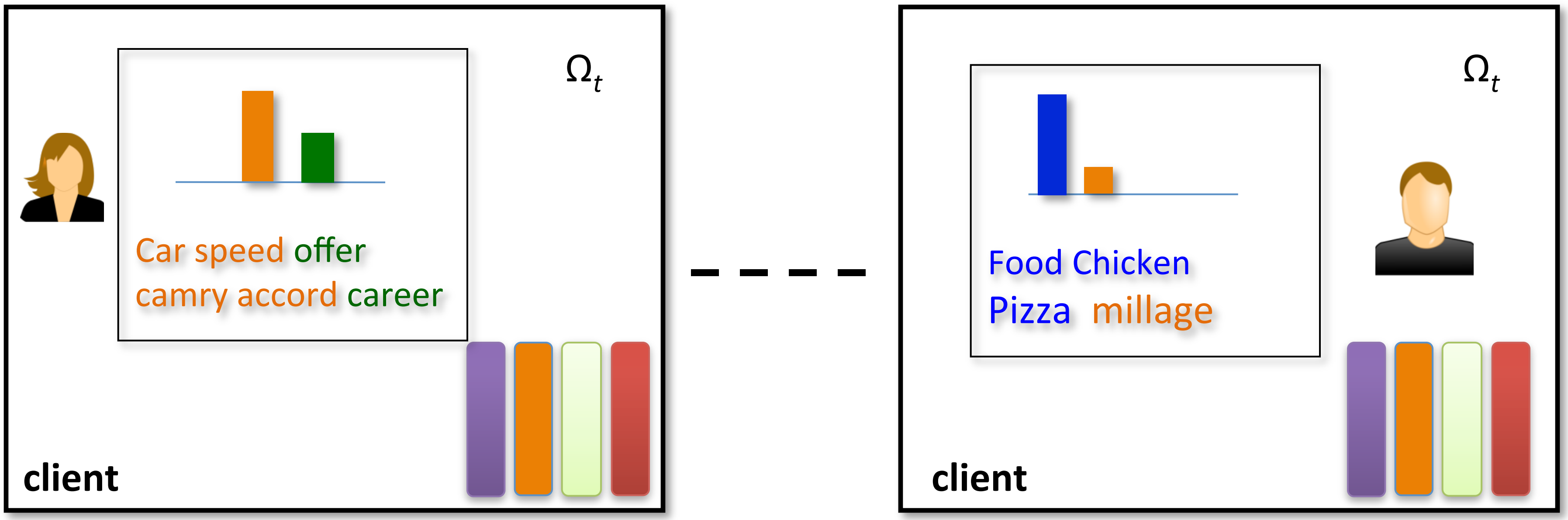
Fully Collapsed



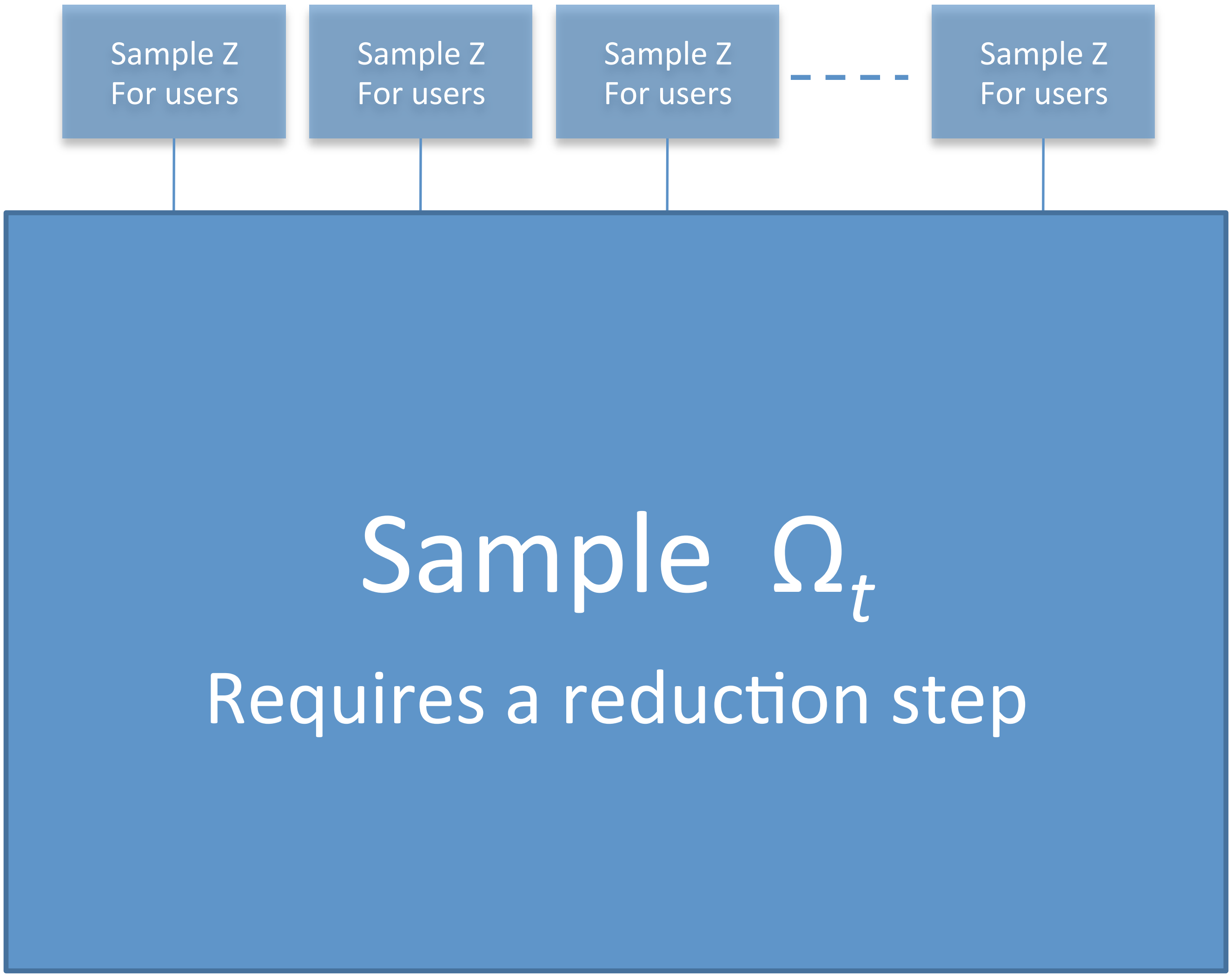
Semi-Collapsed

$$P(z_{ij}^t = k | w_{ij}^t = w, \Omega^t, \tilde{\mathbf{n}}_i^t)$$

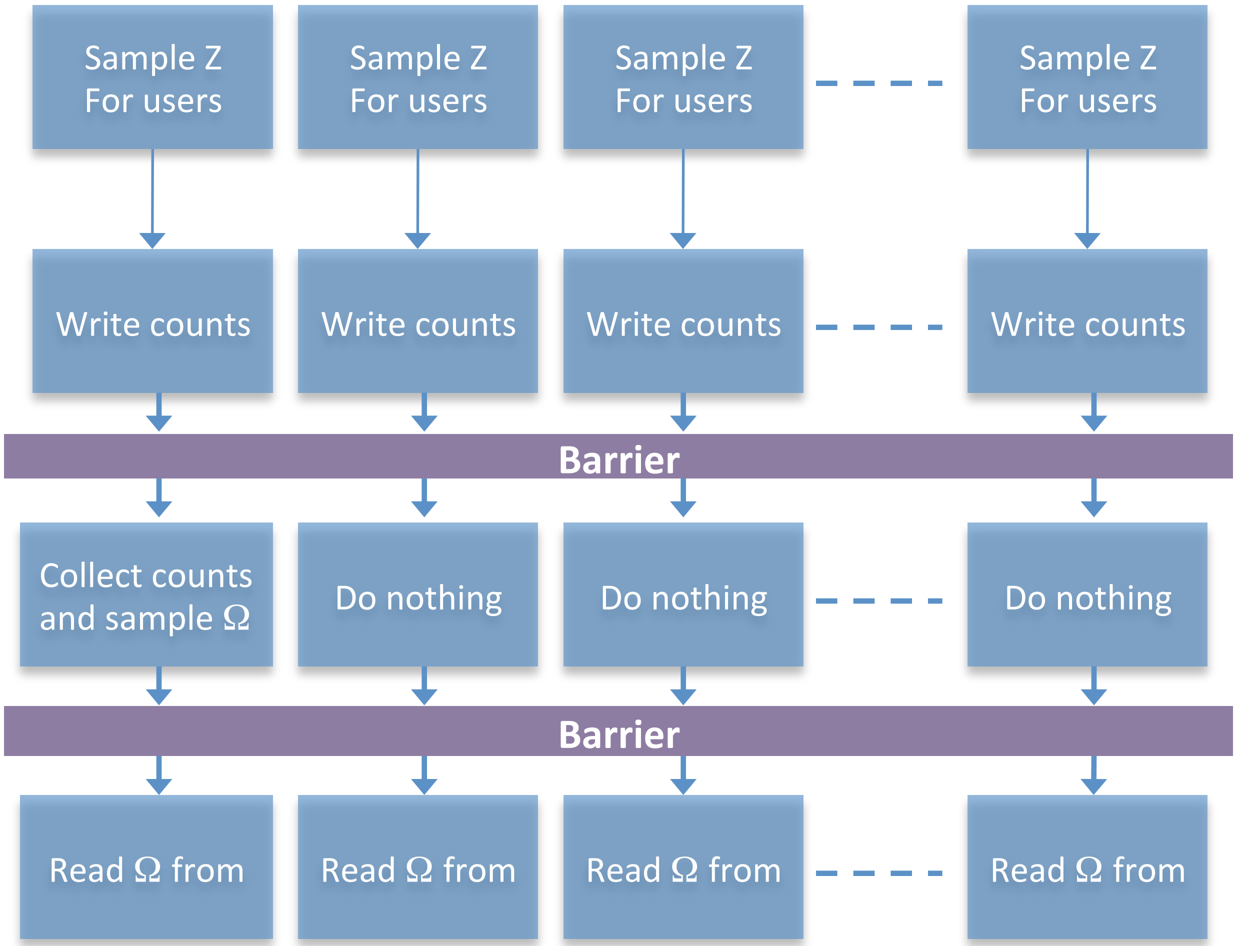
$$\propto \left(n_{ik}^{t,-j} + \tilde{n}_{ik}^t + \lambda \Omega^t \right) \frac{n_{kw}^{t,-j} + \tilde{\beta}_{kw}^t + \beta}{\sum_l n_{kl}^{t,-j} + \tilde{\beta}_{kl}^t + \beta}$$



Distributed Sampling Cycle



Distributed Sampling Cycle



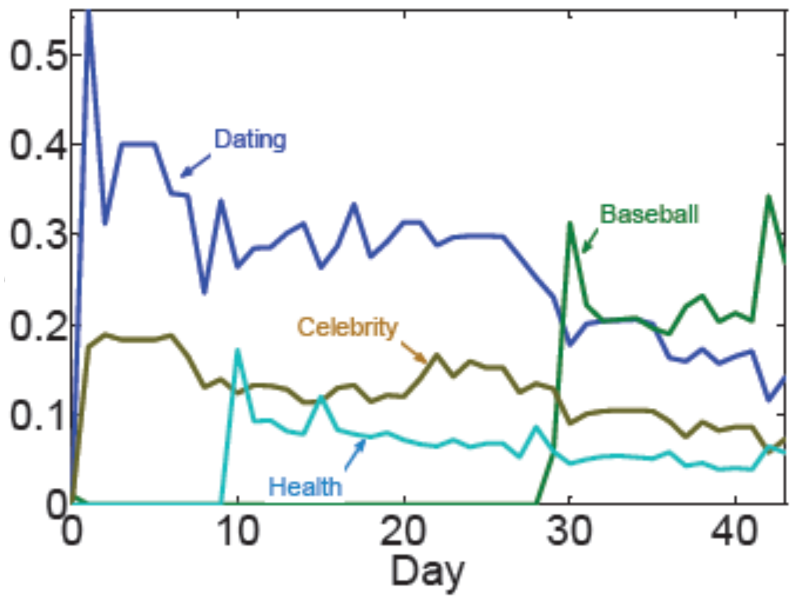
Experimental Results

- Tasks is predicting convergence in display advertising
- Use two datasets
  - 6 weeks of user history
  - Last week responses to Ads are used for testing
- Baseline:
  - User raw data as features
  - Static topic model

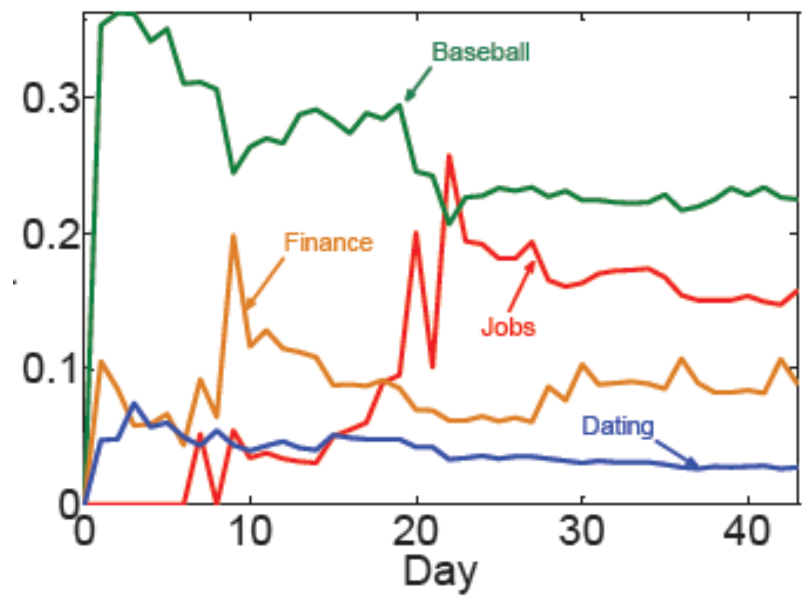
dataset	# days	# users	# campaigns	size
1	56	13.34M	241	242GB
2	44	33.5M	216	435GB

Interpretability

User-1

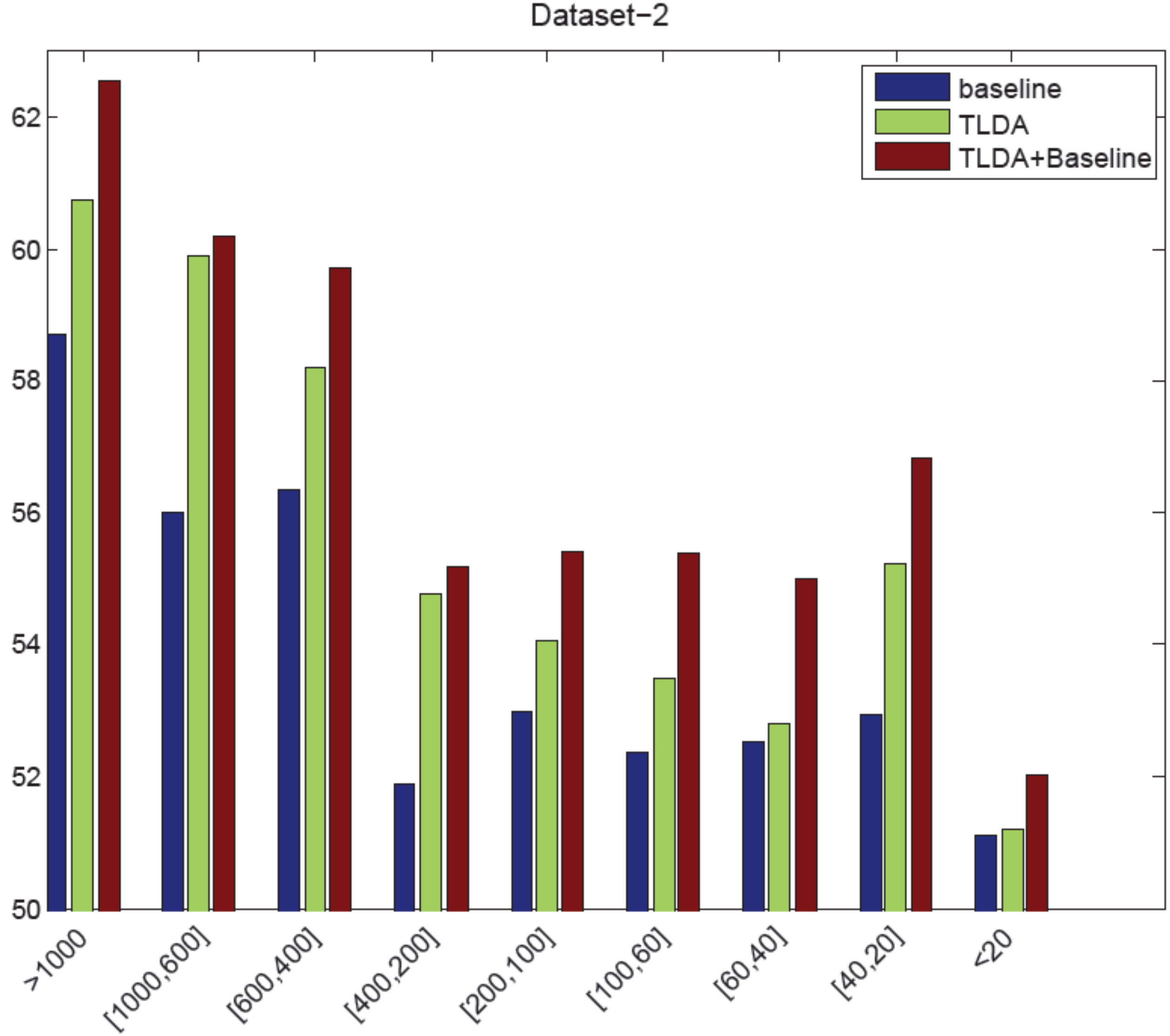


User-2



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			

Performance in Display Advertising



ROC

Number of conversions

Performance in Display Advertising

Weighted ROC measure

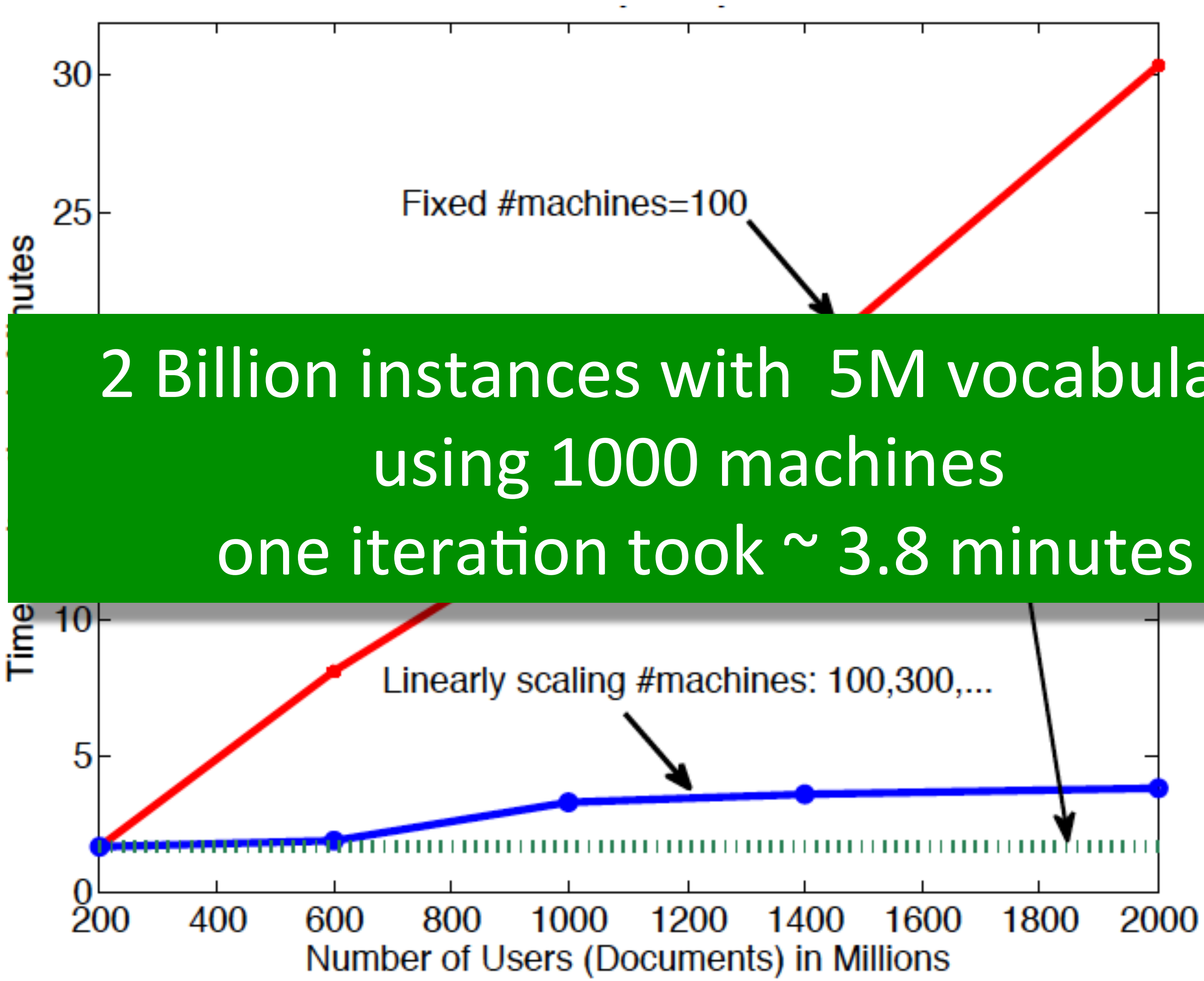
	base	TLDA	TLDA+base	LDA+base
dataset 1	54.40	55.78	**56.94**	55.80
dataset 2	57.03	57.70	**60.38**	58.54

Static
Batch models

Effect of number of topics

	topics	TLDA	TLDA + base
dataset 1	50	55.32	56.01
	100	55.5	56.56
	200	**55.8**	**56.94**
dataset 2	50	59.10	60.40
	100	**59.14**	**60.60**
	200	58.7	60.38

How Does It Scale?

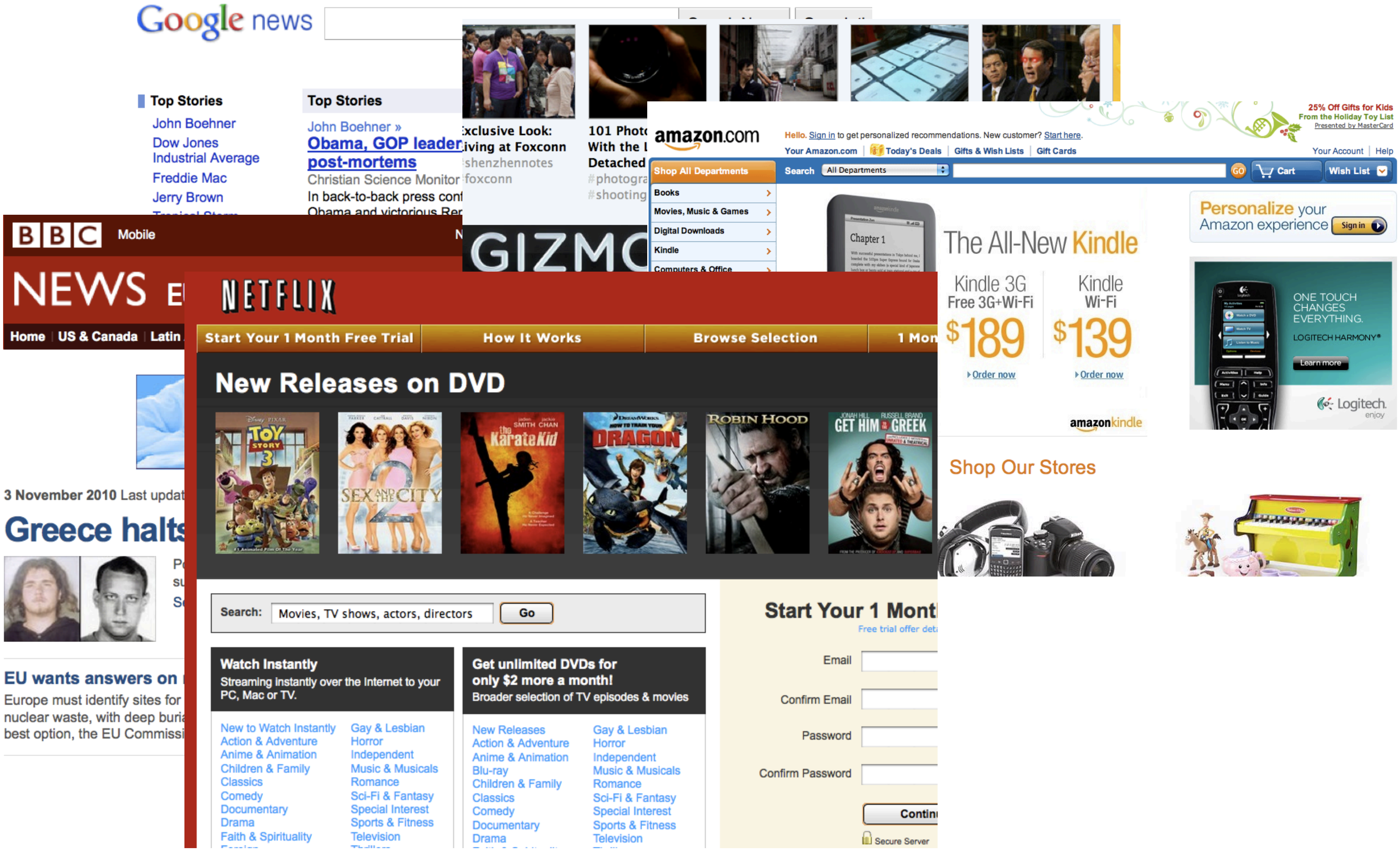


2 Billion instances with 5M vocabulary
using 1000 machines
one iteration took ~ 3.8 minutes

Application

Multi-Domain Personalization

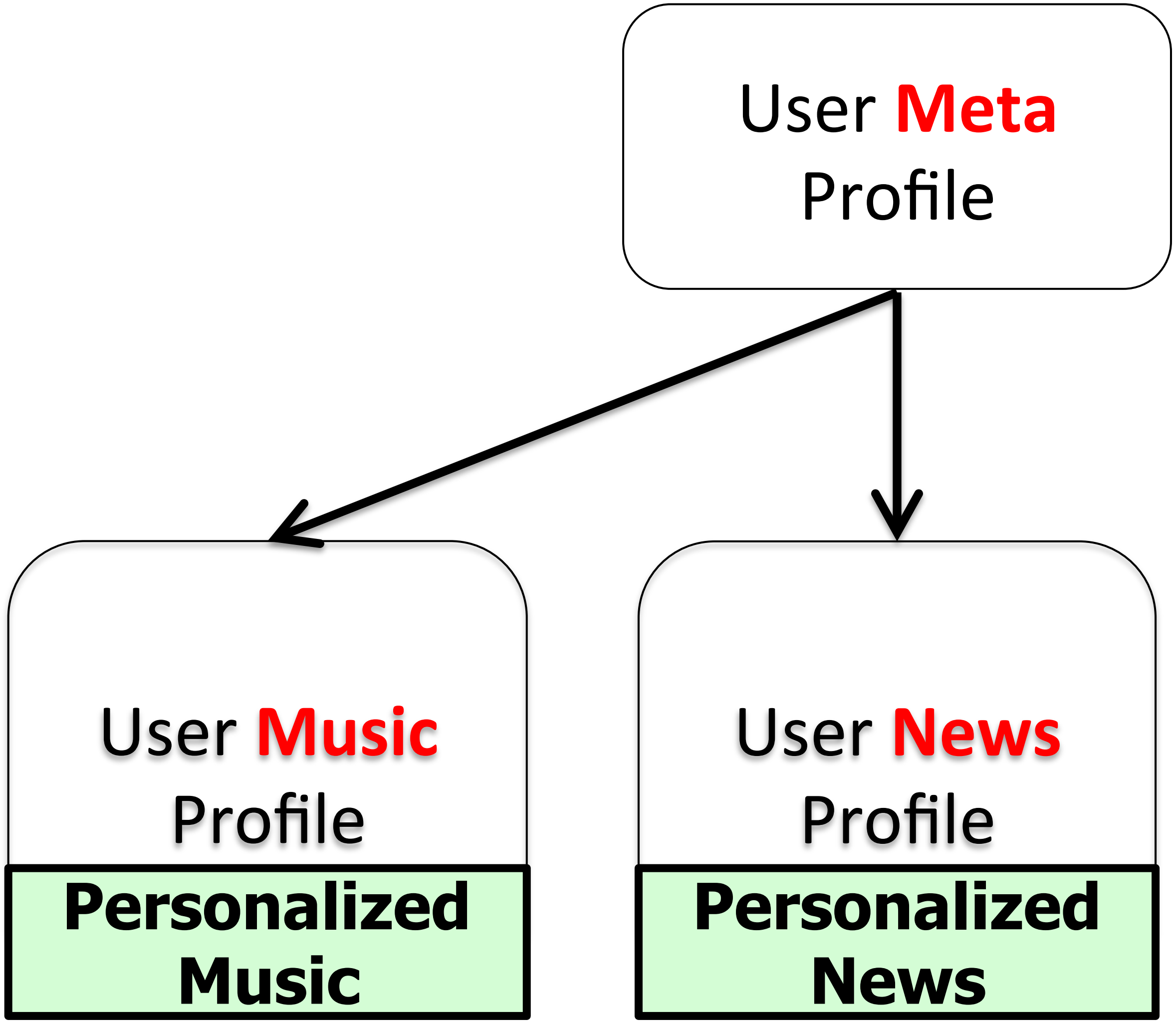
Problem


Google news
Top Stories
John Boehner
Dow Jones Industrial Average
Freddie Mac
Jerry Brown
Top Stories
John Boehner »
Obama, GOP leader
post-mortems
Christian Science Monitor
In back-to-back press conf
BBC
Mobile
NEWS
Home
US & Canada
Latin
3 November 2010 Last updat
Greece halts
EU wants answers on
Europe must identify sites for
nuclear waste, with deep buri
best option, the EU Commissi
Exclusive Look:
Living at Foxconn
shenzhennotes
foxconn
101 Photo
With the
Detached
#photogra
#shooting
GIZMO
amazon.com
Hello. Sign in to get personalized recommendations. New customer? Start here.
Your Amazon.com
Today's Deals
Gifts & Wish Lists
Gift Cards
Your Account
Help
25% Off Gifts for Kids
From the Holiday Toy List
Presented by MasterCard
Shop All Departments
Search
All Departments
GO
Cart
Wish List
Books
Movies, Music & Games
Digital Downloads
Kindle
Computers & Office
The All-New Kindle
Kindle 3G
Free 3G+Wi-Fi
$189
Order now
Kindle
Wi-Fi
$139
Order now
amazonkindle
Personalize your
Amazon experience
Sign in
ONE TOUCH
CHANGES
EVERYTHING.
LOGITECH HARMONY®
Learn more
Logitech
Shop Our Stores
NETFLIX
Start Your 1 Month Free Trial
How It Works
Browse Selection
1 Mon
New Releases on DVD
Search:
Movies, TV shows, actors, directors
Go
Watch Instantly
Streaming instantly over the Internet to your PC, Mac or TV.
New to Watch Instantly
Action & Adventure
Anime & Animation
Children & Family
Classics
Comedy
Documentary
Drama
Faith & Spirituality
Gay & Lesbian
Horror
Independent
Music & Musicals
Romance
Sci-Fi & Fantasy
Special Interest
Sports & Fitness
Television
Get unlimited DVDs for only $2 more a month!
Broader selection of TV episodes & movies
New Releases
Action & Adventure
Anime & Animation
Blu-ray
Children & Family
Classics
Comedy
Documentary
Drama
Gay & Lesbian
Horror
Independent
Music & Musicals
Romance
Sci-Fi & Fantasy
Special Interest
Sports & Fitness
Television
Start Your 1 Mont
Free trial offer det
Email
Confirm Email
Password
Confirm Password
Contin
Secure Server

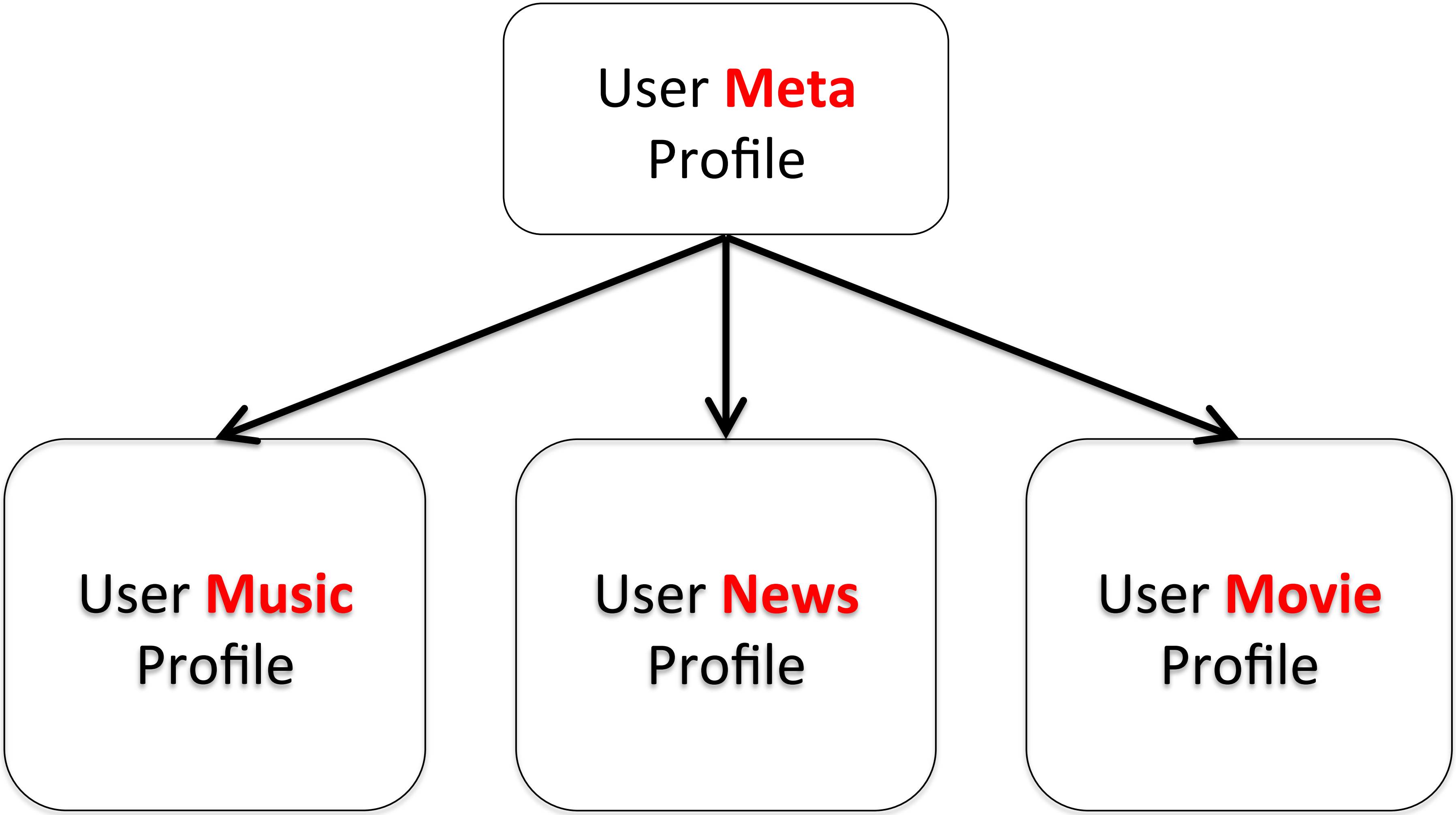
Multi-domain Personalization

- Intuition
  - We observe user interaction with news and movies
  - Can we predict his music taste?
- Interaction definition
  - A bag of words describing objects user interacts with in a given domain

Example


User Meta Profile
User Music Profile
Personalized Music
User News Profile
Personalized News

Example

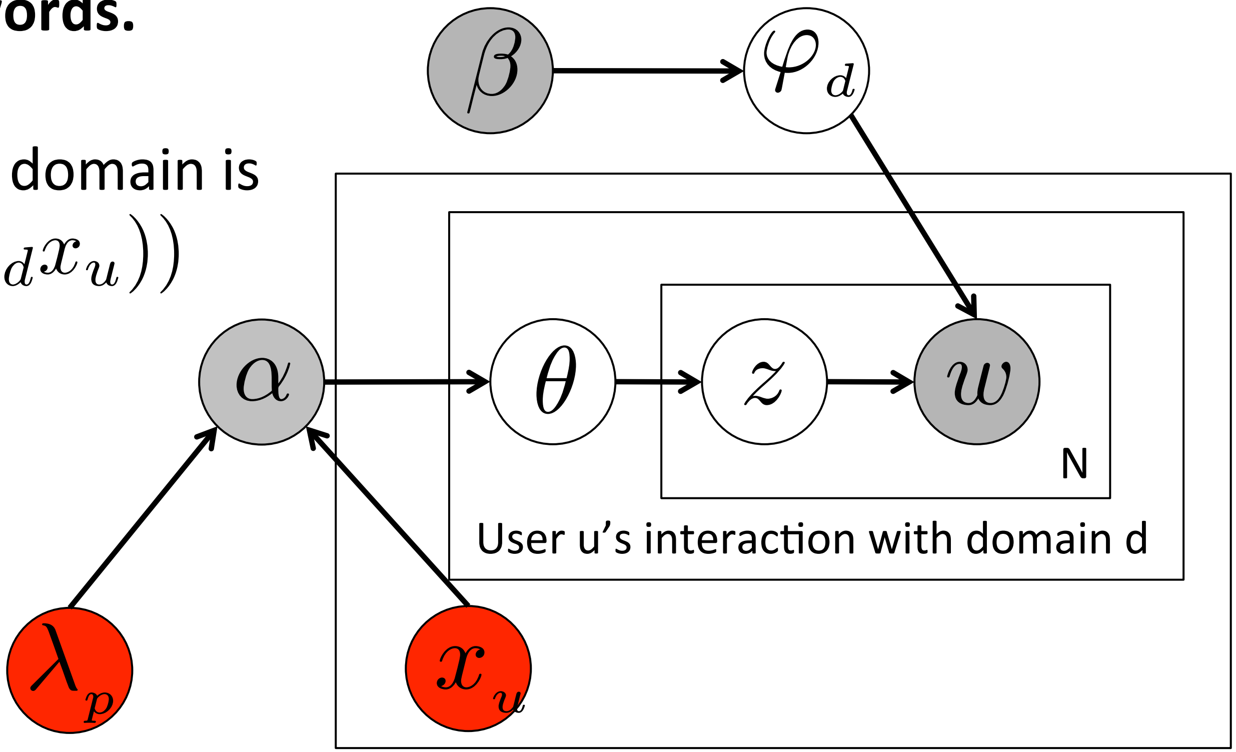

User Meta Profile
User Music Profile
User News Profile
User Movie Profile

The Model

A **user's interaction with a domain** is a **bag of words.**
A **topic** is a **mixture of words.**

User's **prior** interest in a domain is

$$\alpha = \log(1 + \exp(\lambda_d x_u))$$



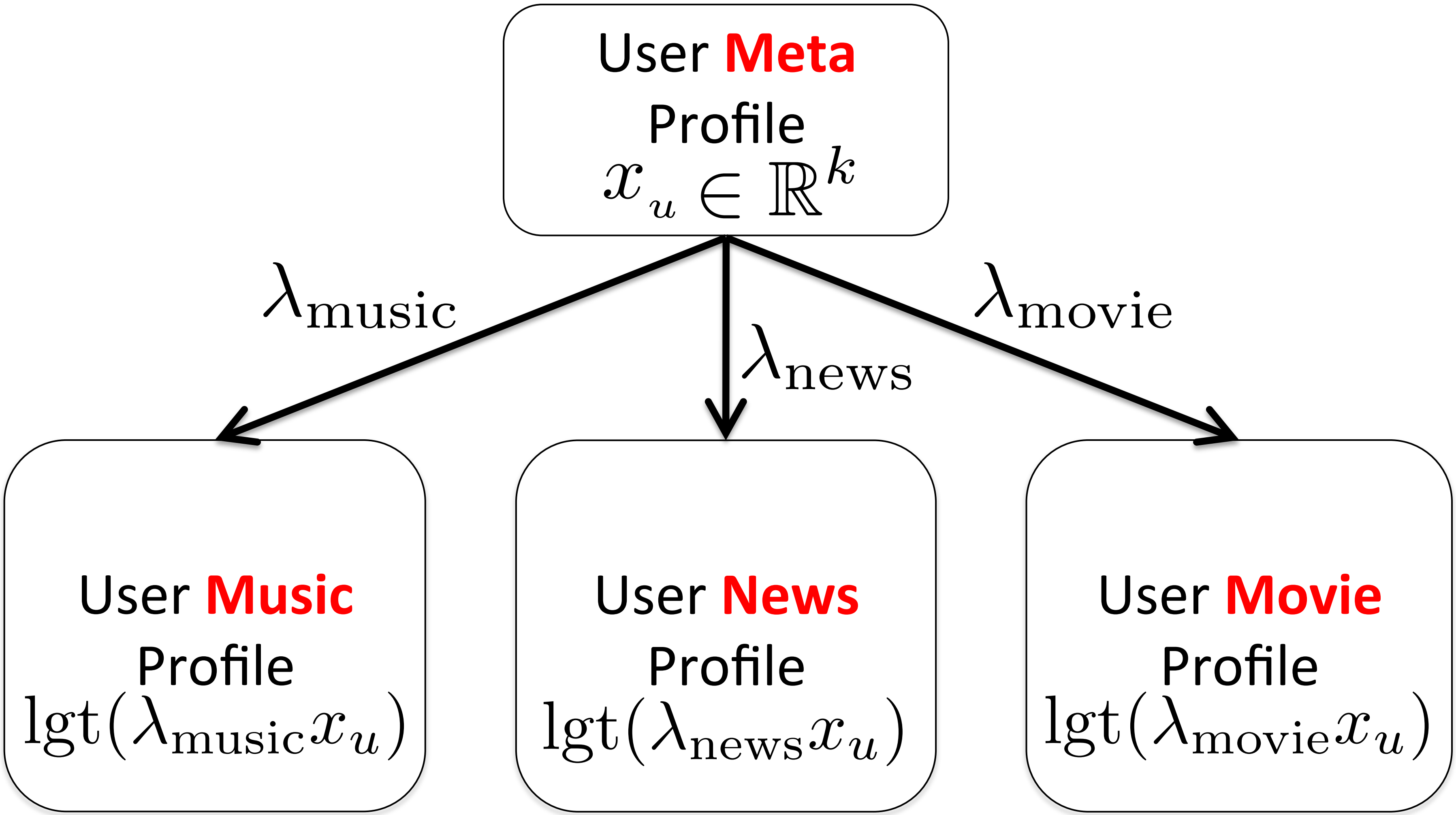
Each user has a meta-profile: $x_u \in \mathbb{R}^k$

Each domain has a latent matrix: $\lambda_d \in \mathbb{R}^{k \times t_d}$

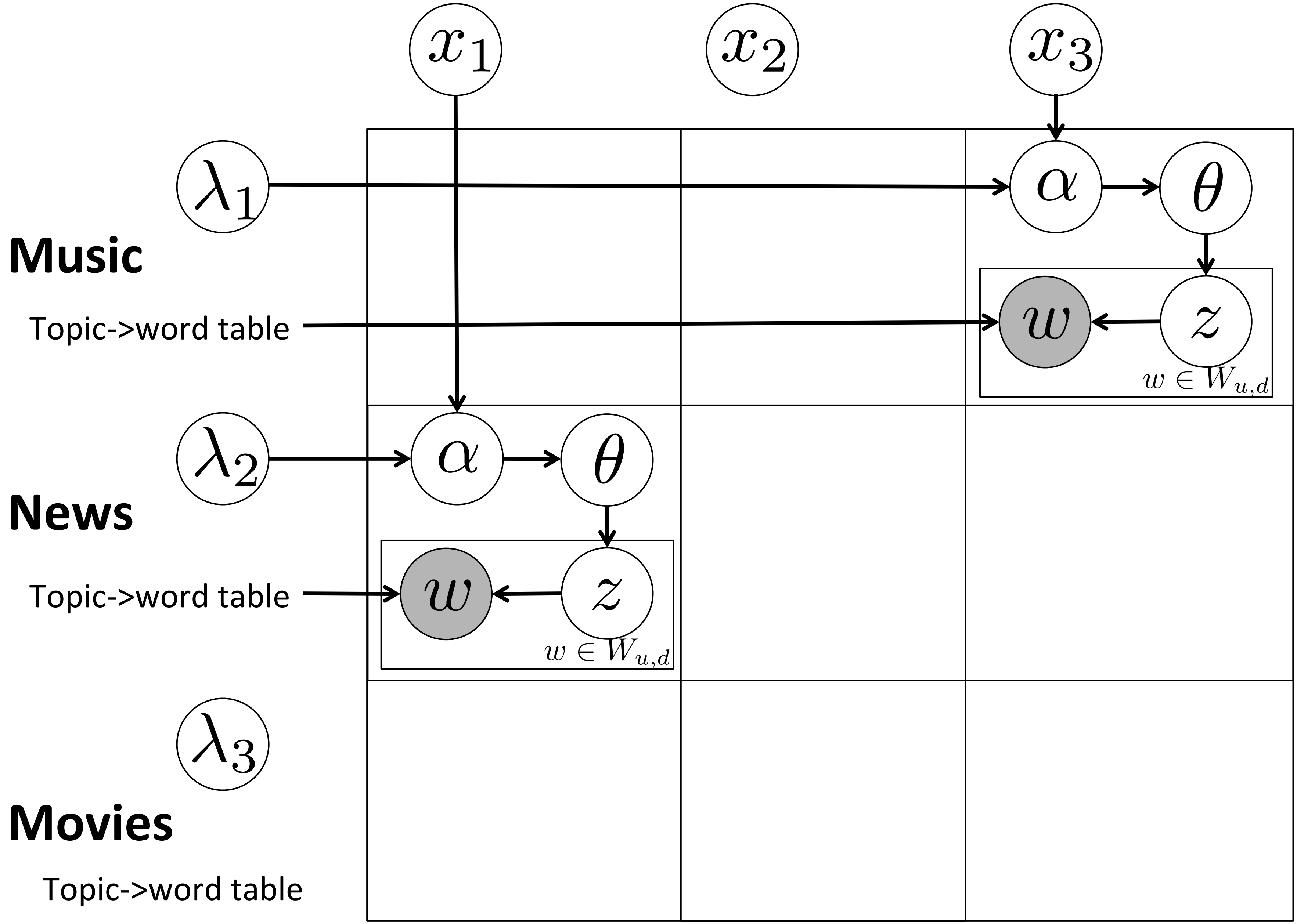
Slide credit Yucheng Low

The Model

$$\mathrm{lgt}(x) = \log(1 + \exp(x))$$

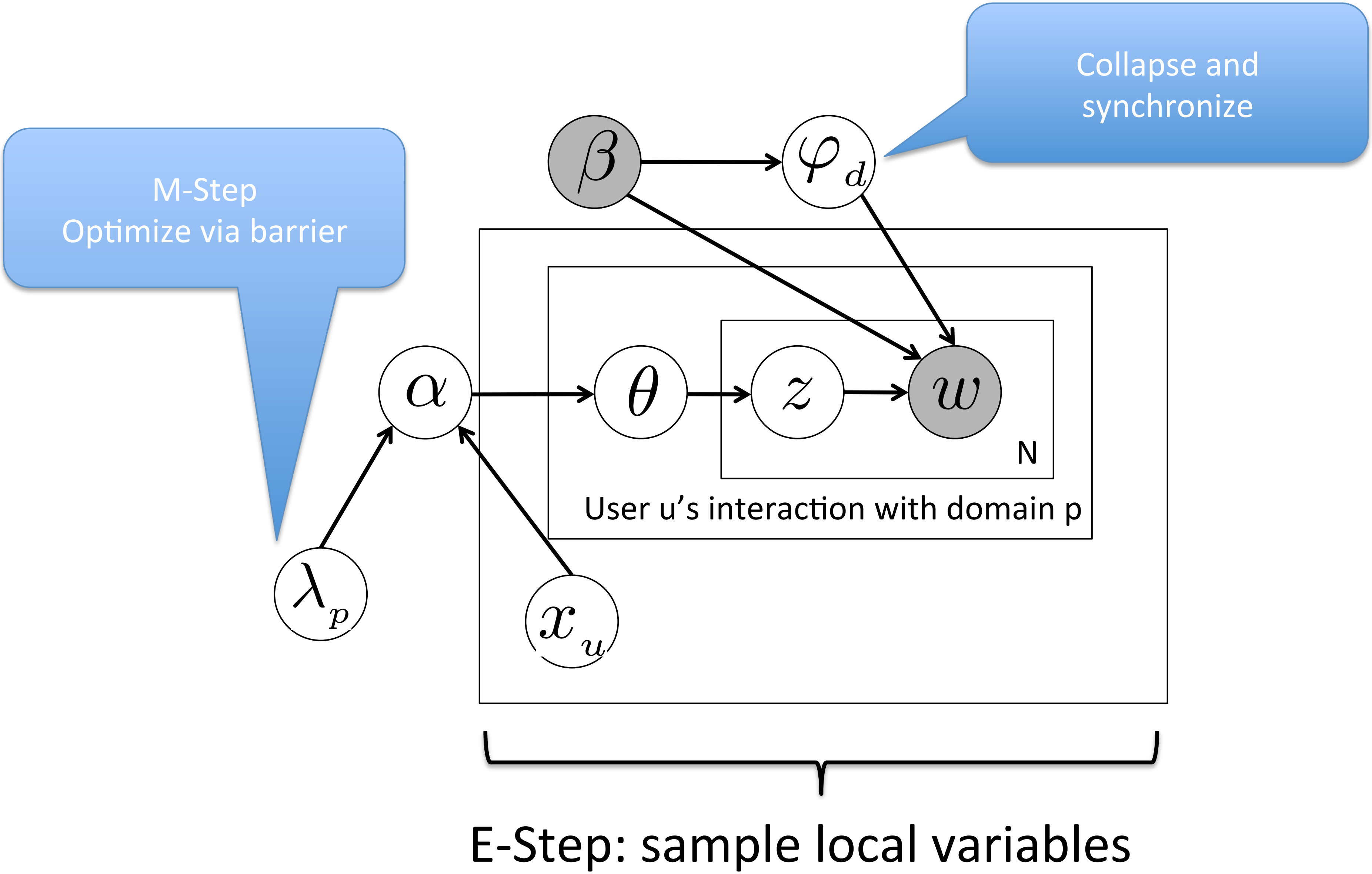


Slide credit Yucheng Low

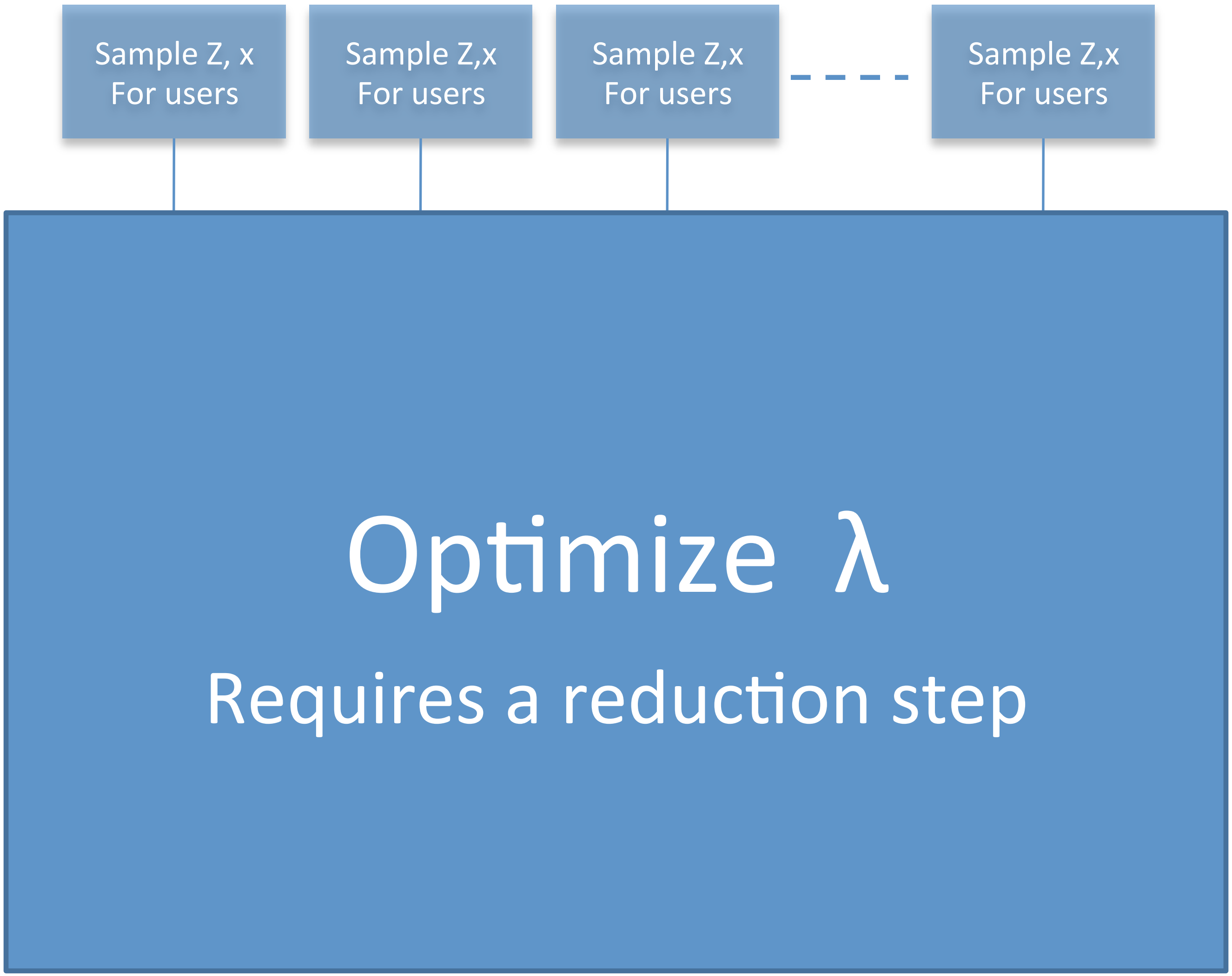


Slide credit Yucheng Low

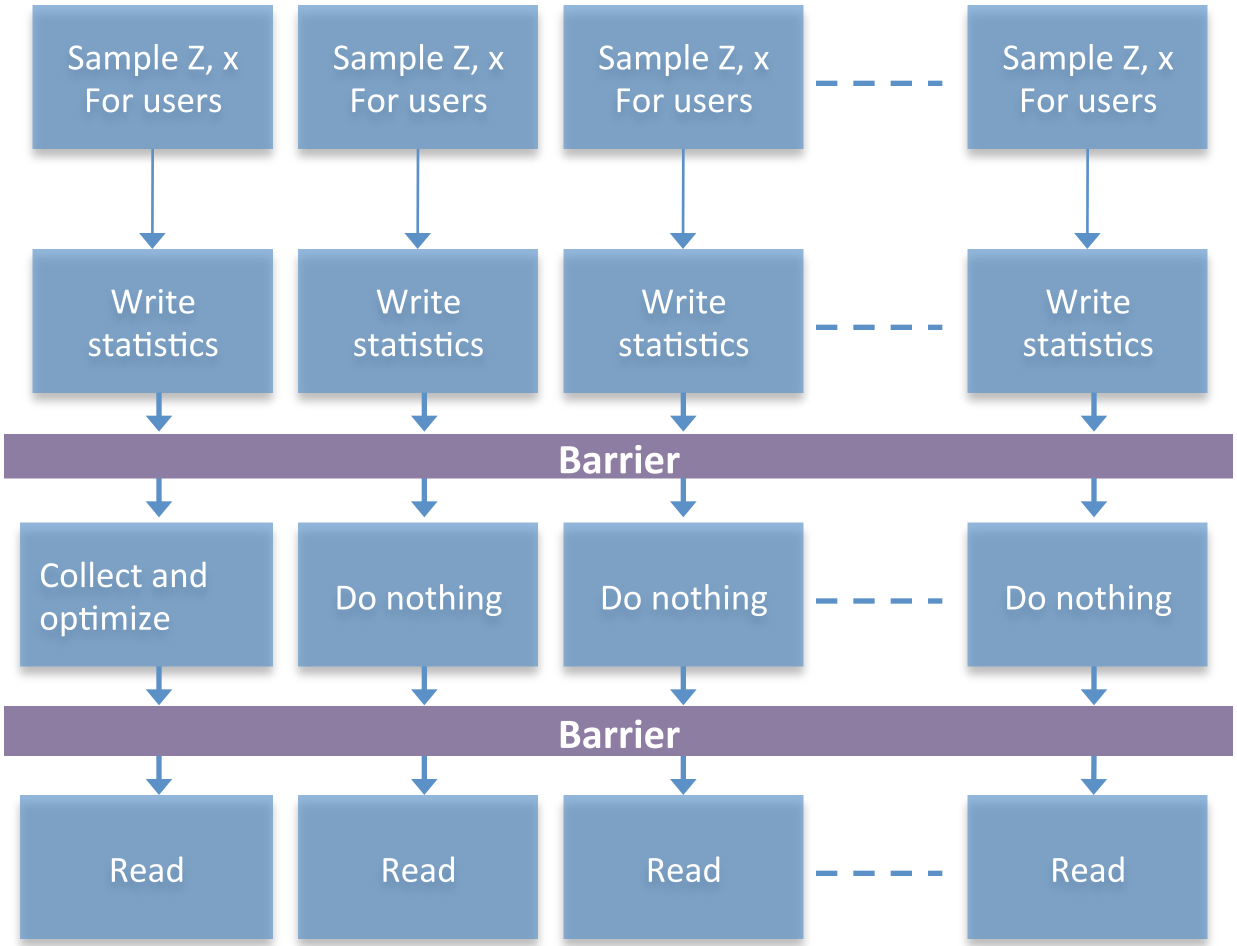
Inference and Learning



Distributed Sampling Cycle



Distributed Sampling Cycle



Results

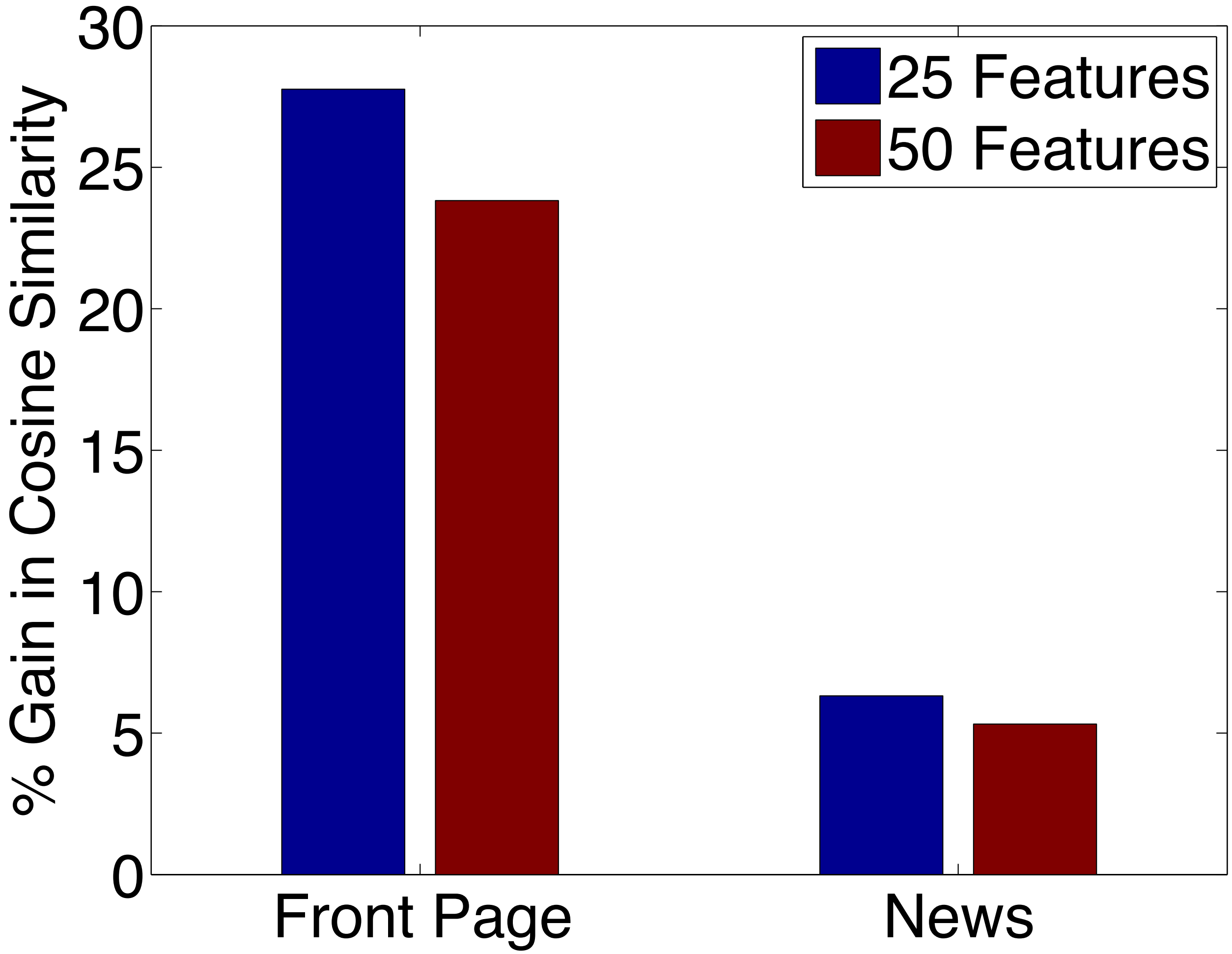
- **2 domain dataset.**

  Frontpage and News clicks of **5.6 million users.**

  Frontpage/News: Article text for each click.

- Measure gain relative to independent models on each domain

Results

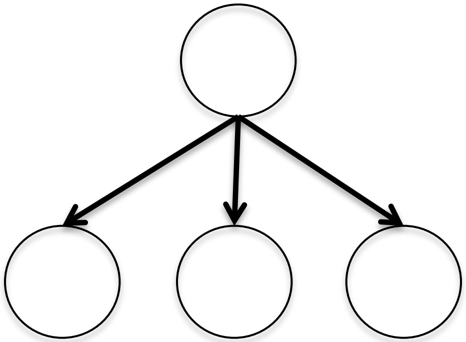


25 Features
50 Features
% Gain in Cosine Similarity
30
25
20
15
10
5
0
Front Page
News

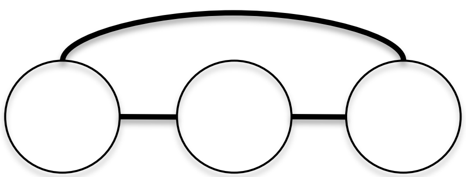
Distributed Inference Revisited

To collapse or not to collapse?

- Not collapsing
  - Keeps conditional independence
    - Good for parallelization
    - Requires synchronous sampling
  - Might mix slowly



- Collapsing
  - Mixes faster
  - Hinder parallelism
  - Use star-synchronization
    - Works well if sibling depends on each others via aggregates
    - Requires asynchronous communication



Inference Primitive

- Collapse a variable
  - Star synchronization for the sufficient statistics
- Sampling a variable
  - Local
    - Sample it locally (possibly using the synchronized statistics)
  - Shared
    - Synchronous sampling using a barrier
- Optimizing a variable
  - Same as in the shared variable case
  - Ex. Conditional topic models

Asynchronous Optimization

Asynchronous Processing

- Needed when
  - Ex: Optimizing a global variable
- Mostly requires a barrier
- Advantages
  - Easy to program
  - Well-understood reusable templates
- Disadvantages
  - The curse of the last reducer
  - You are as fast as the slowest machine!

Asynchronous Processing

- Needed when
  - Ex: Optimize a global varia
- Mostly requires a barr
- Advantages
  - Easy to progr
  - Well-und … sable template
- Disadv
  - The cu … r the last reducer
  - You are as fast as the slowest machine!

Can we do better?

Asynchronous Optimization

Graph Factorization

Graph Factorization Problem

- Factor a graph into low rank components
- Assign a latent vector $Z_i \in \mathcal{R}^k$ with each node
- Optimize:

$$f(Y, Z, \lambda) = \frac{1}{2} \sum_{(i,j) \in E} \left(Y_{ij} - \langle Z_i, Z_j \rangle \right)^2 + \frac{\lambda}{2} \sum_i n_i \|Z_i\|^2$$

Observed value over edges

Predicted value

Regularization

Single-Machine Algorithm

- Just use stochastic gradient decent (SGD)

$$\frac{\partial f}{\partial Z_i} = -\sum_{j \in \mathcal{N}(i)} \big(Y_{ij} - \langle Z_i, Z_j \rangle\big) Z_j + \lambda n_i Z_i$$

- Cycle until convergence
  - Read a node, *i*
  - Update its latent factor

$$Z_i \leftarrow Z_i - \eta\left(\frac{\partial f}{\partial Z_i}\right)$$

Problem Scale

- Yahoo IM and Mail graphs
- Nodes are users
- Edges represent (log) number of messages
- 200 Million vertices
- 10 Billion edges

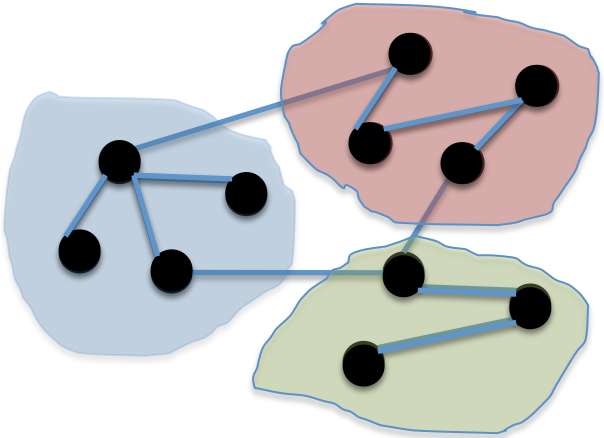
Challenges

- Parameter storage
  - Too much for a single machine
- Approach
  - Distribute the graph over machines
    - How to partition the nodes?
  - Synchronization
    - How to synchronize replicated nodes
  - Communication
    - How to accommodate network topology

Challenges

Can we solve the problem with similar ideas to what we have covered?

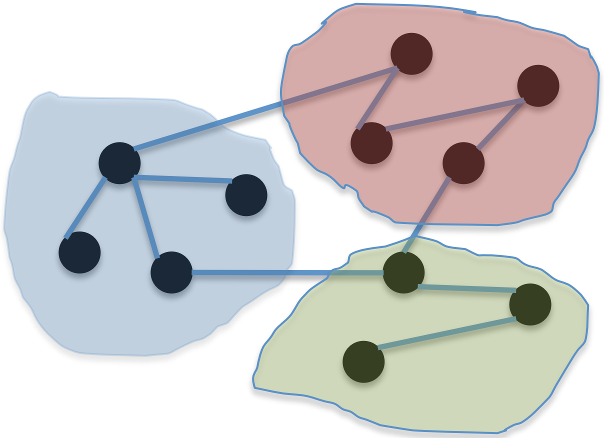
Partition and Replicate

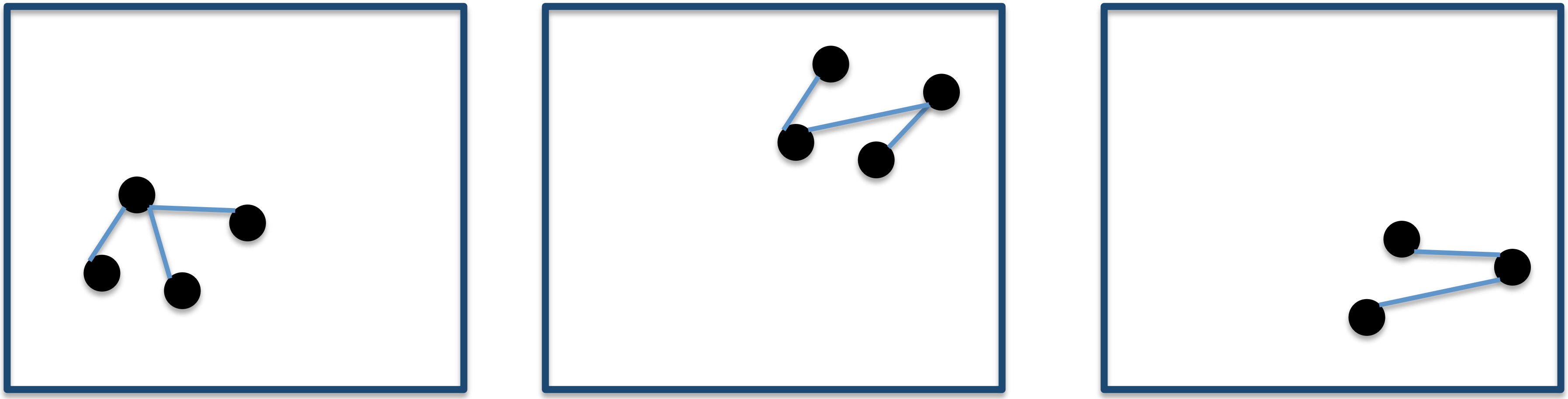


Partition and Replicate

- Cycle until convergence
  – Read a node, *i*
  – Update its latent factor

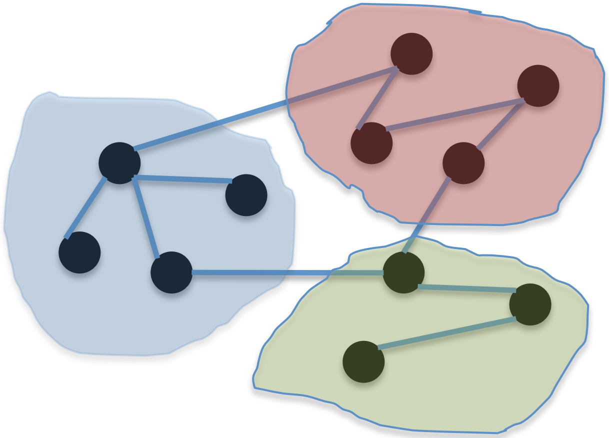
$$Z_i \leftarrow Z_i - \eta\left(\frac{\partial f}{\partial Z_i}\right)$$

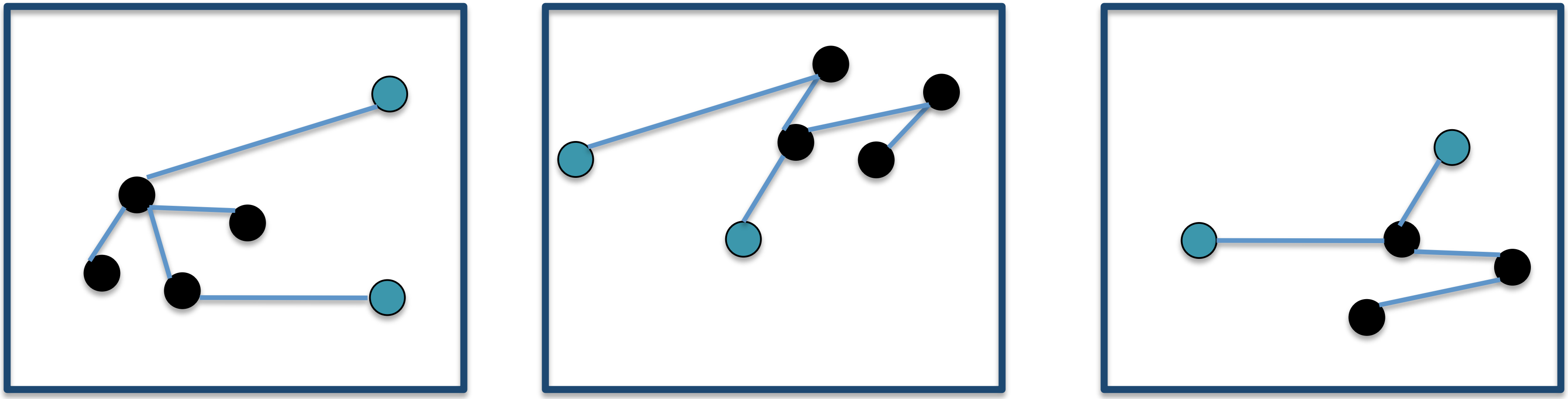




Partition and Replicate

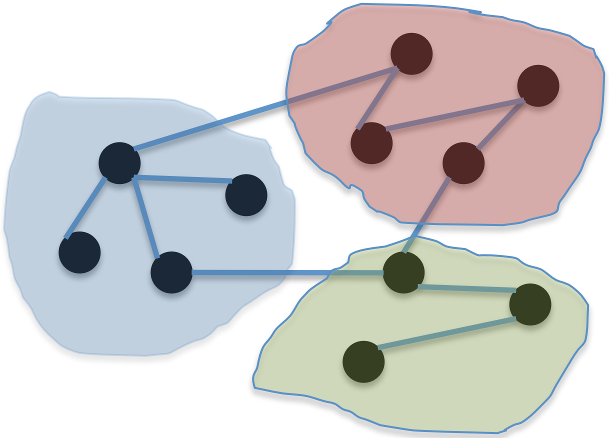
- Problem
  - Some neighbors are missing
- Solution
  - Replicate and synchronize
  - **Borrowed** vs. owned nodes

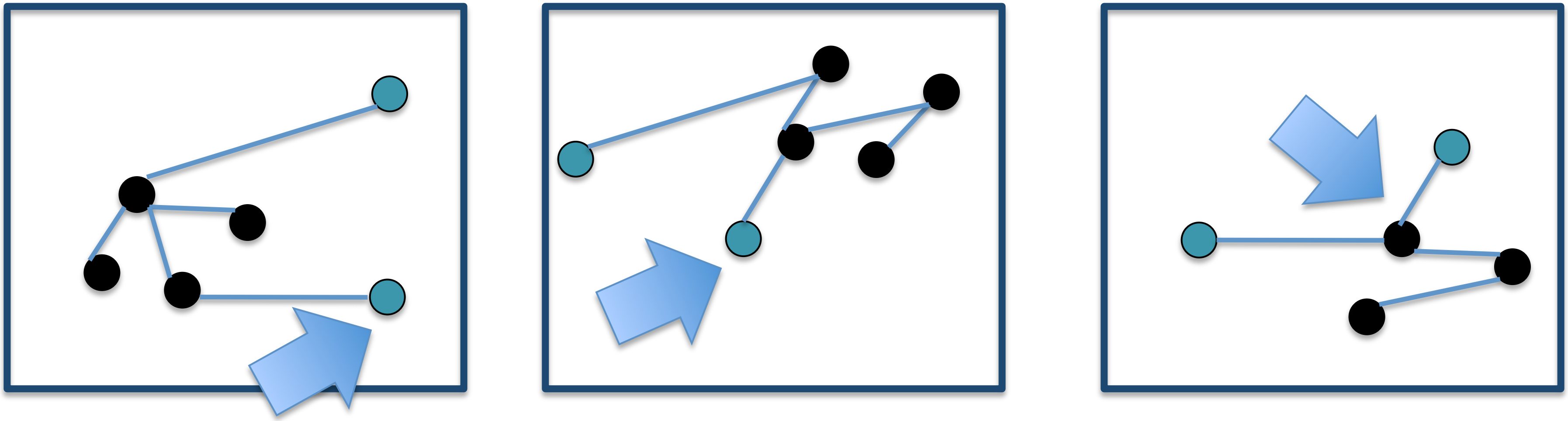




Partition and Replicate

- Formulation
  - Introduce local copies
    - A factor per node X
  - Tie across machines
    - Introduce global factor Z
    - Penalizes deviations





Formulation

- Original problem

$$f(Y,Z,\lambda)=\frac{1}{2}\sum_{(i,j)\in E}\left(Y_{ij}-\langle Z_i,Z_j\rangle\right)^2+\frac{\lambda}{2}\sum_i n_i\|Z_i\|^2$$

- Relaxed problem

$$\sum_{k=1}^{K} f_k(Y,X^{(k)},\lambda)+\frac{1}{2}\sum_{k=1}^{K}\left[\mu\sum_{i\in V_k}\|Z_i-X_i^{(k)}\|^2\right]$$

Global factor

Local factors

Deviation

- Local problem

$$f_k(Y,X^{(k)},\lambda)$$

$$=\frac{1}{2}\left[\sum_{\substack{(i,j)\in E,\\ i,j\in V_k}}\left(Y_{ij}-\langle X_i^{(k)},X_j^{(k)}\rangle\right)^2+\lambda\sum_{i\in V_k}n_i\|X_i^{(k)}\|^2\right]$$

Synchronous Algorithms

- Optimize joint objective over X,Z
- Local parameter updates
  - Run SGD until convergence

$$\text{minimize}_{X^{(k)}} f_k(Y, X^{(k)}, \lambda) + \frac{1}{2}\mu \sum_{i \in V_k} \|Z_i - X_i^{(k)}\|^2$$

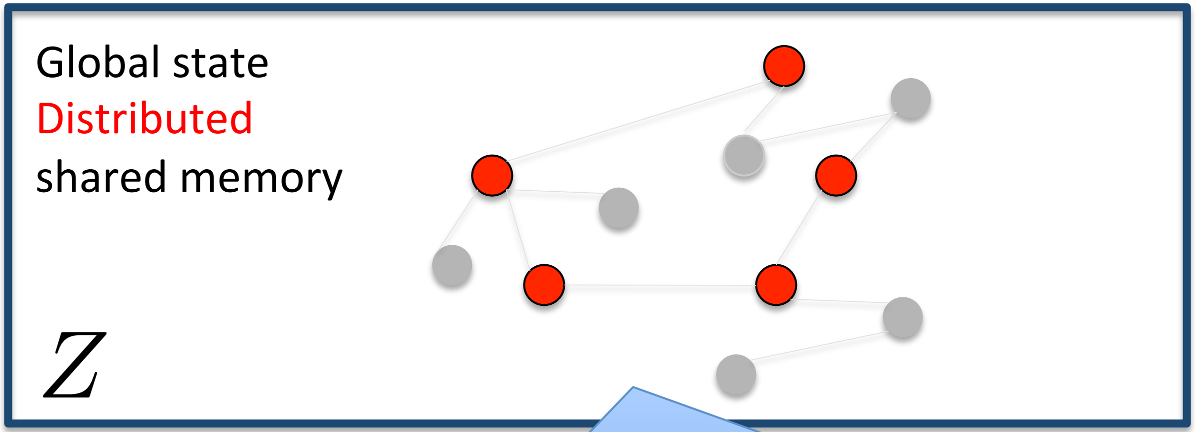
Fit the data

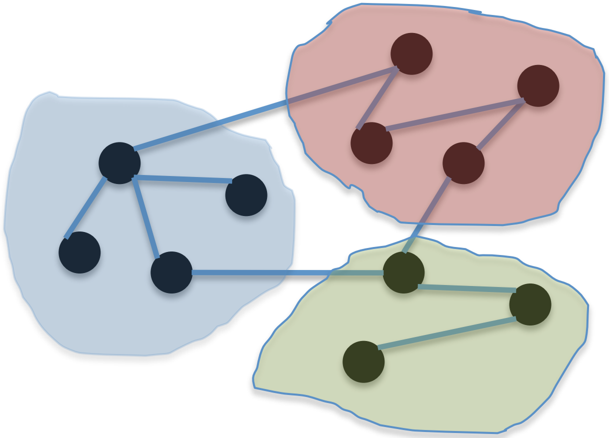
Minimize deviation

- Global parameter updates

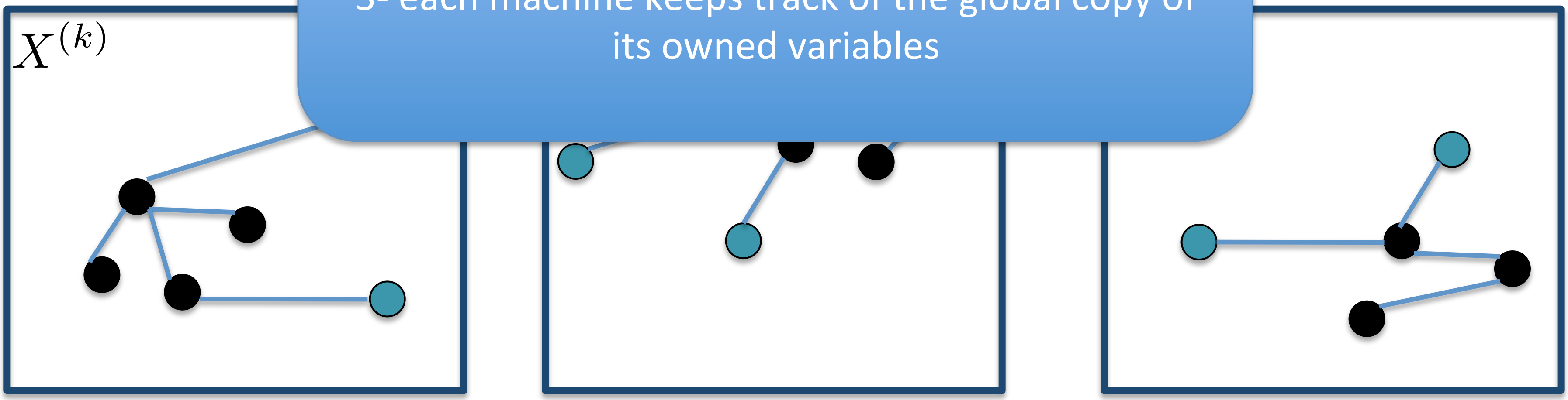
$$\text{minimize}_Z \quad \frac{1}{2} \sum_{k=1}^{K} \left[\mu \sum_{i \in V_k} \|Z_i - X_i^{(k)}\|^2 \right]$$

Synchronous Algorithms

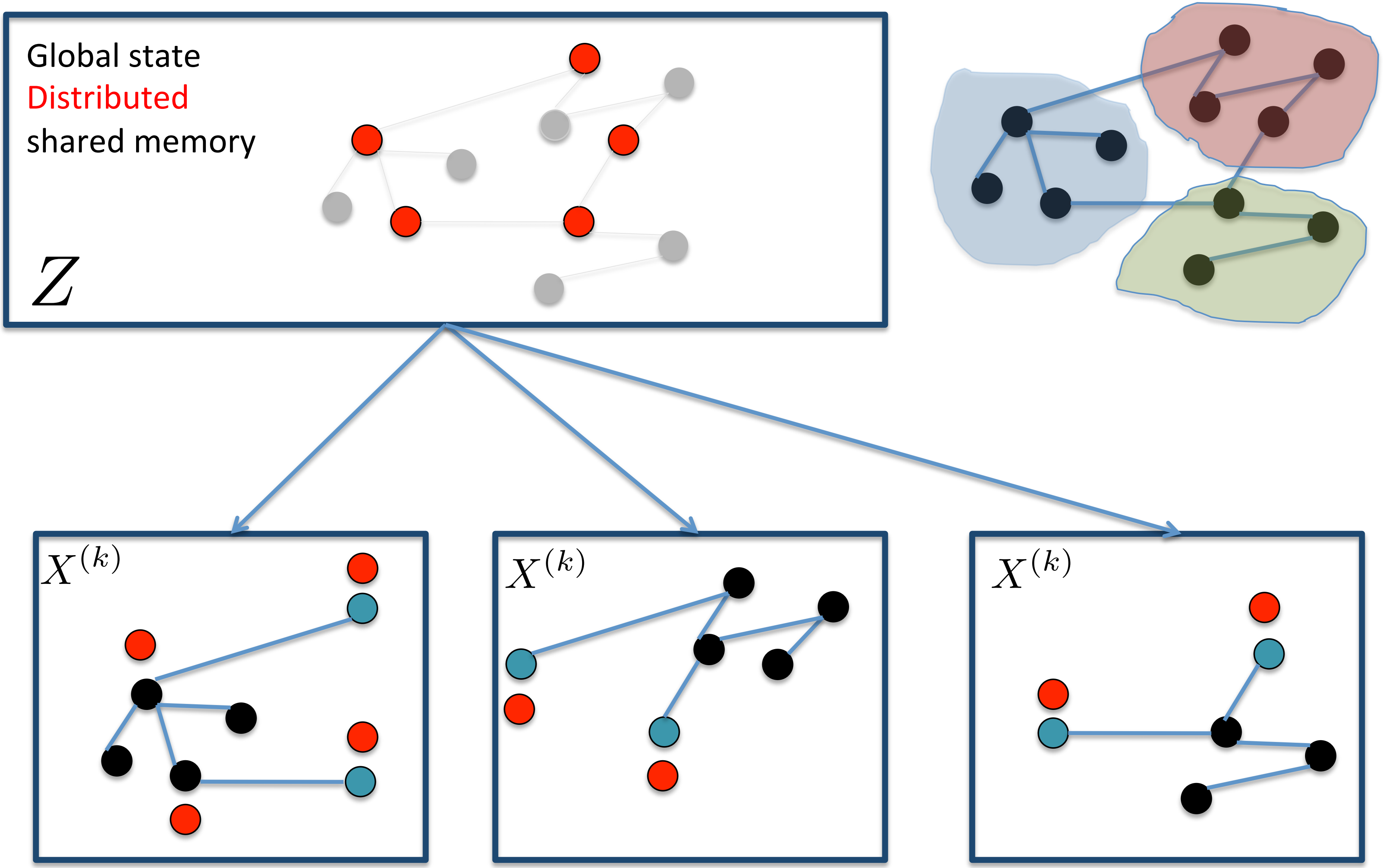




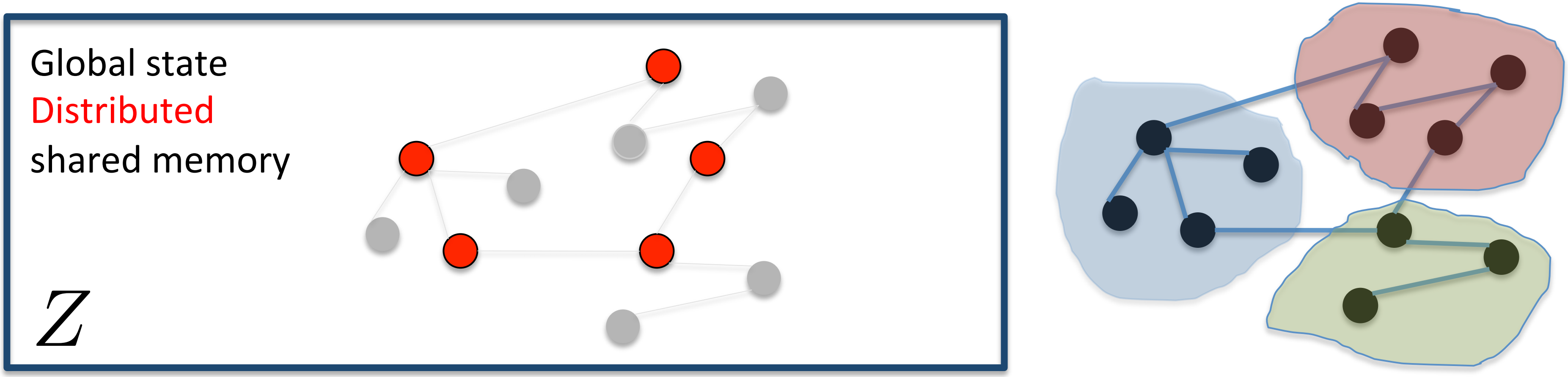
1- We only store replicated nodes
2- The global state is distributed across machines
3- each machine keeps track of the global copy of its owned variables



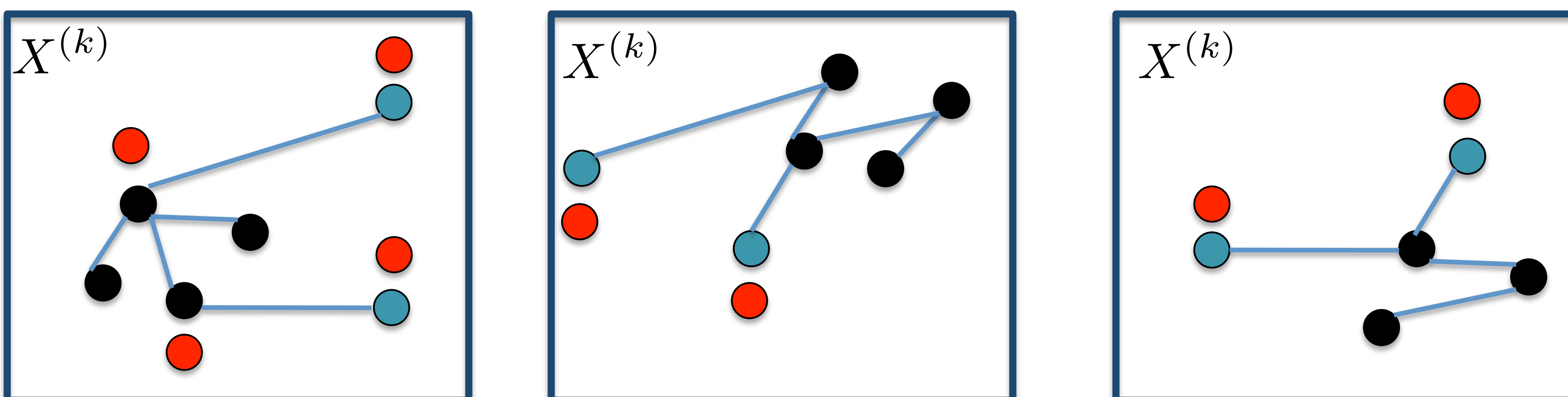
Step 1: Push global variables



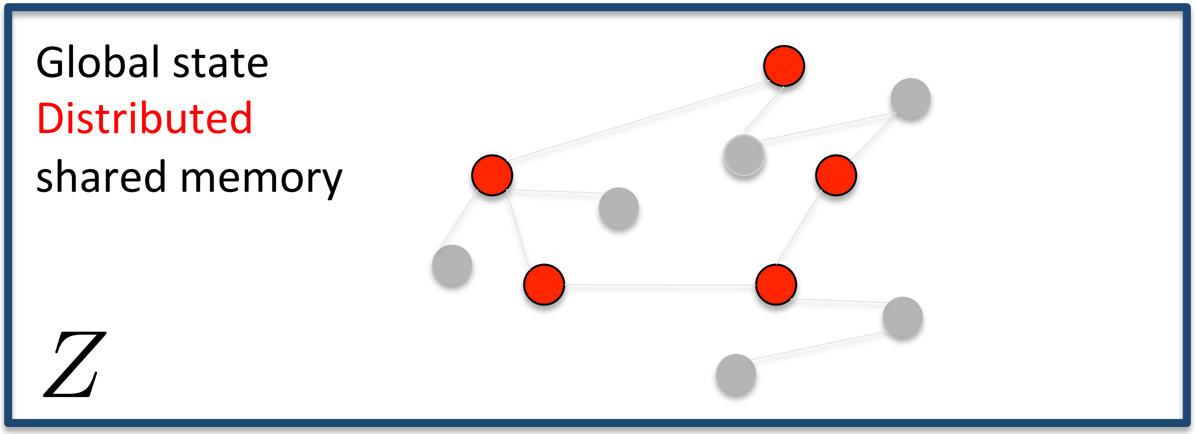
Step 2: Local Optimization

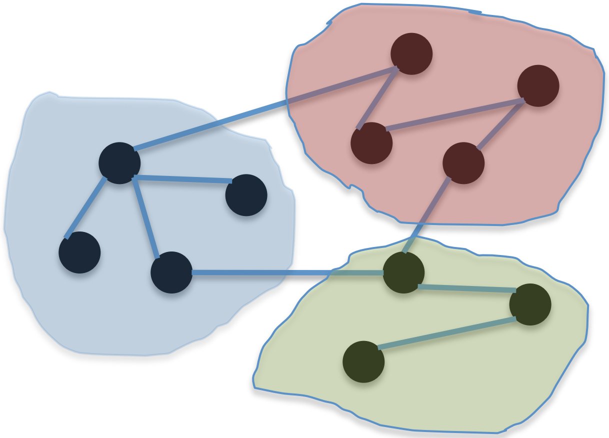


$$\text{minimize}_{X^{(k)}} f_k(Y, X^{(k)}, \lambda) + \frac{1}{2}\mu \sum_{i \in V_k} \|Z_i - X_i^{(k)}\|^2$$

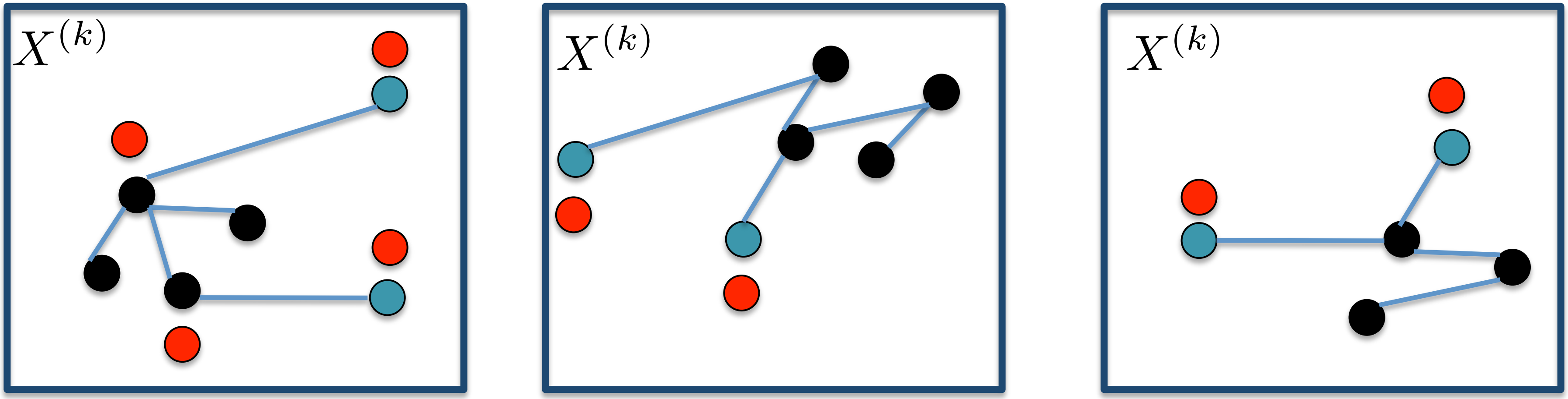


Step 3: Push and average

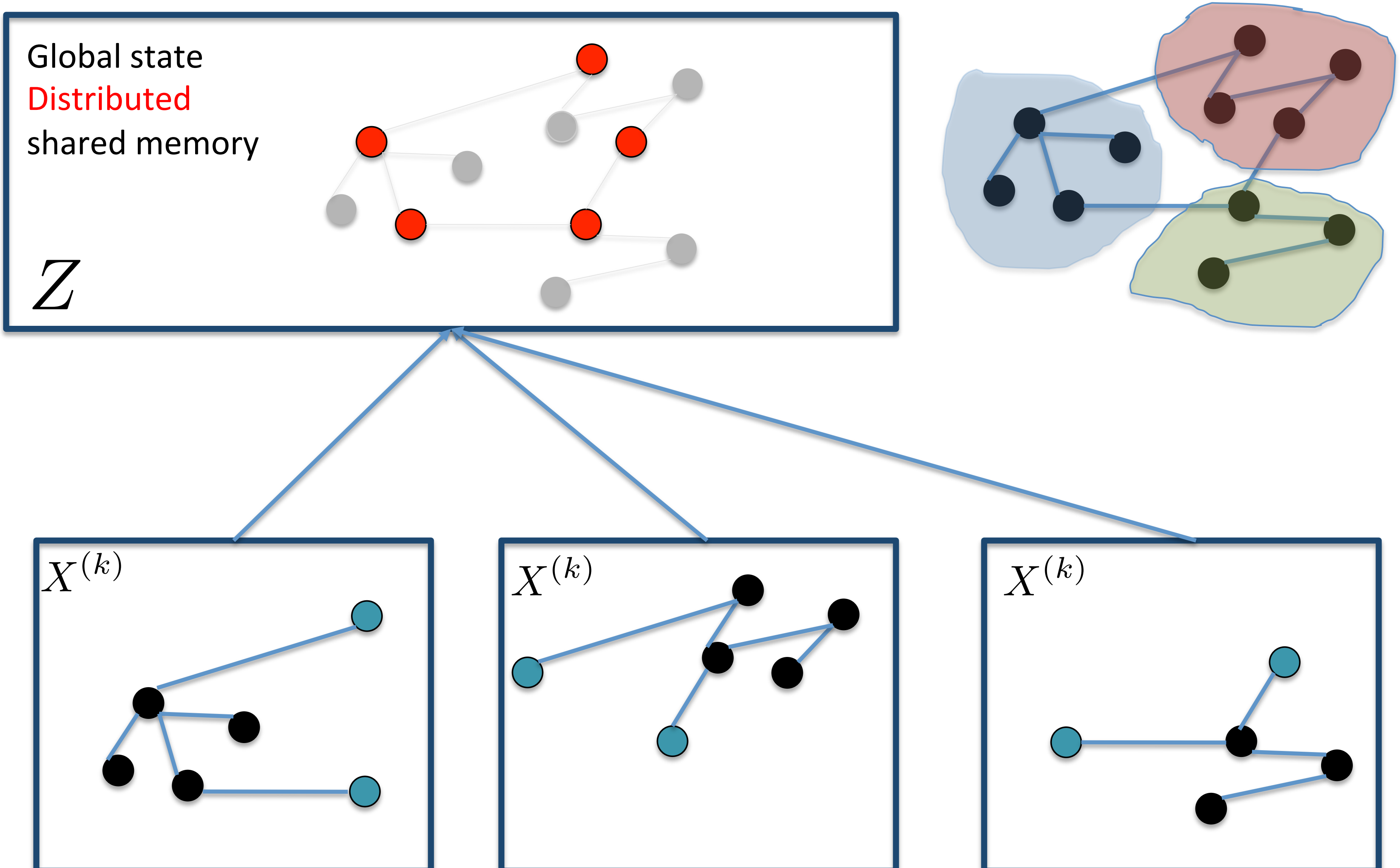




$$\text{minimize}_Z \quad \frac{1}{2}\sum_{k=1}^{K}\left[\mu \sum_{i \in V_k} \|Z_i - X_i^{(k)}\|^2\right]$$



Step 3: Push and average



Summary of Asynchronous Algorithms

- An improvement over standard Map-Reduce
- Curse of the last reducer
- You are as fast as the slowest machine
  - Optimize local variables
  - Barrier
  - Optimize global variables
  - Barrier
- Can we do better?

An Asynchronous Algorithm

- Conceptual idea
  - Optimize X and Z jointly

$$\sum_{k=1}^{K} f_k(Y, X^{(k)}, \lambda) + \frac{1}{2} \sum_{k=1}^{K} \left[\mu \sum_{i \in V_k} \| Z_i - X_i^{(k)} \|^2 \right]$$

- User SGD over (X,Z)
- Pick a local node
- Do a gradient step over corresponding X,Z!

Conceptual Idea

$$\sum_{k=1}^{K} f_k(Y, X^{(k)}, \lambda) + \frac{1}{2}\sum_{k=1}^{K}\left[\mu \sum_{i \in V_k} \|Z_i - X_i^{(k)}\|^2\right]$$

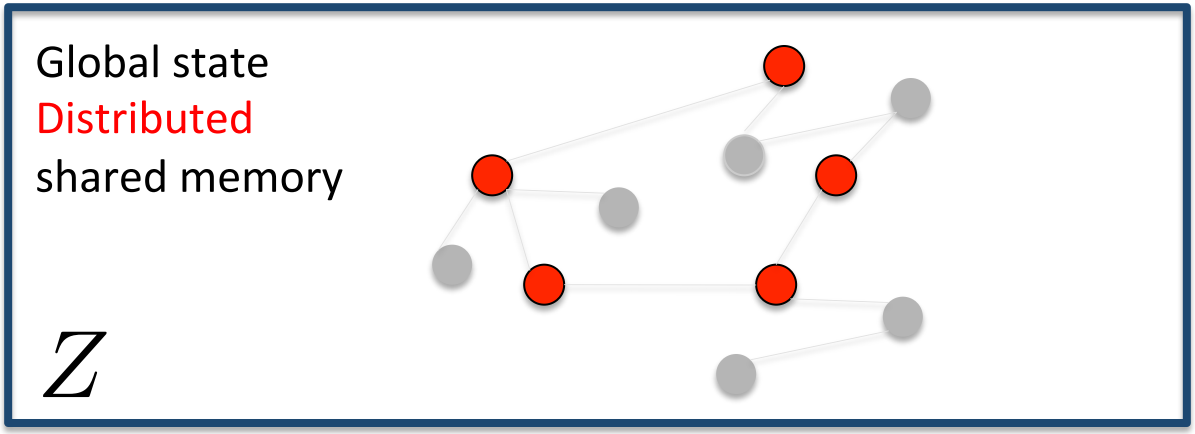
$$\frac{\partial f}{\partial Z_i}\left[X_i^{(k)}\right] = \mu(Z_i - X_i^{(k)}).$$

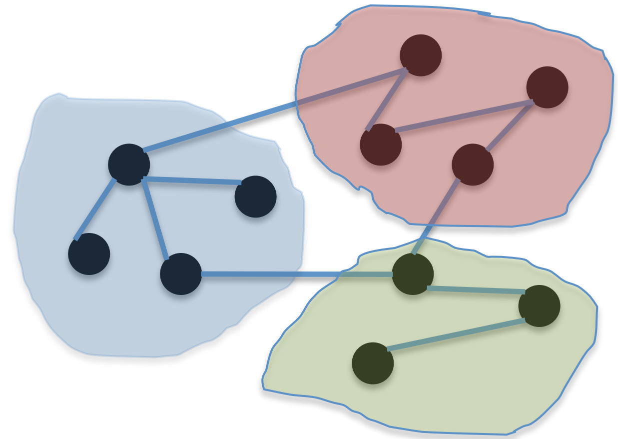
Cache the global variables
Locally (Asynchronous updates)

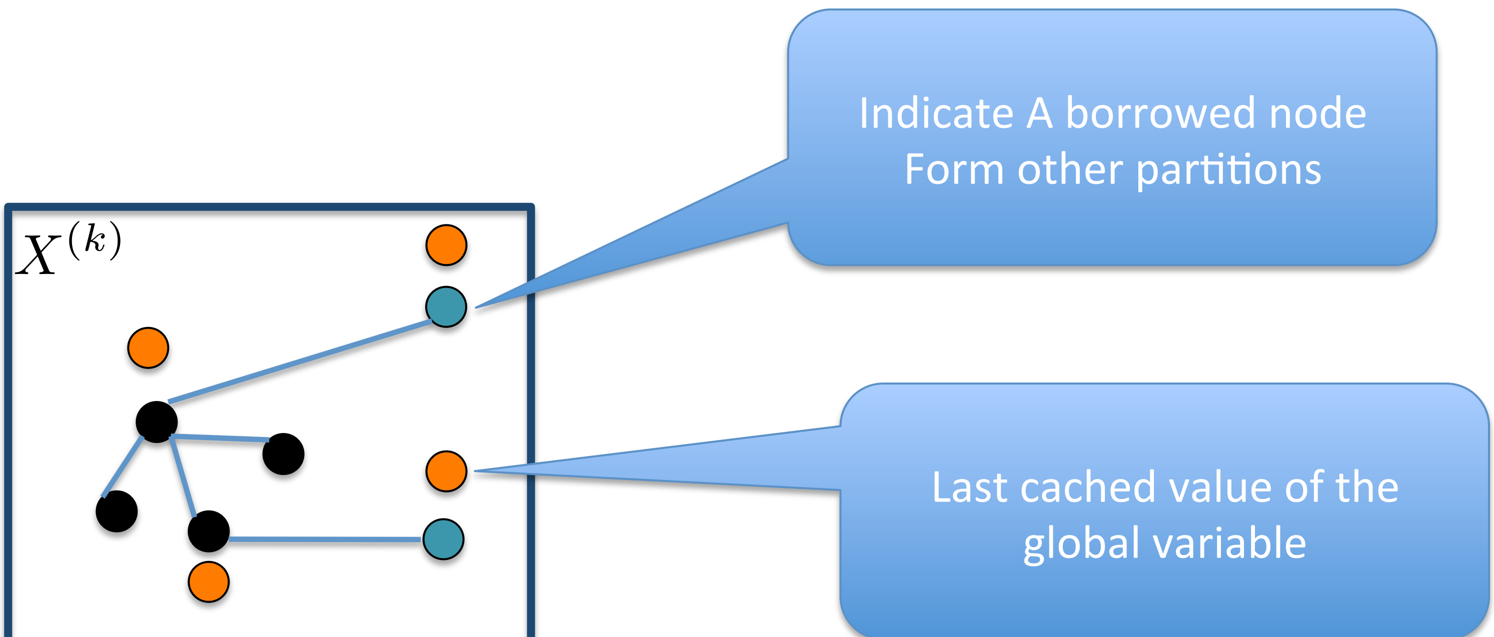
We don't have global copy locally

$$+ \lambda n_i X_i^{(k)} + \mu(X_i^{(k)} - Z_i).$$

Parallel Updates

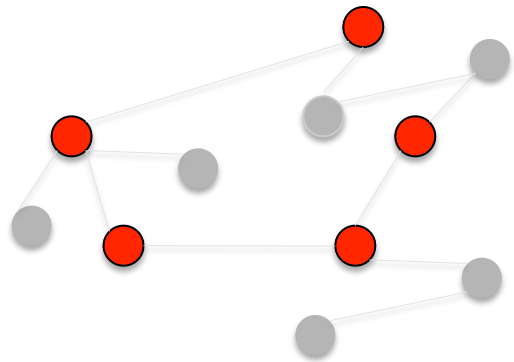






Parallel Asynchronous Updates

Global state
Distributed
shared memory



Z

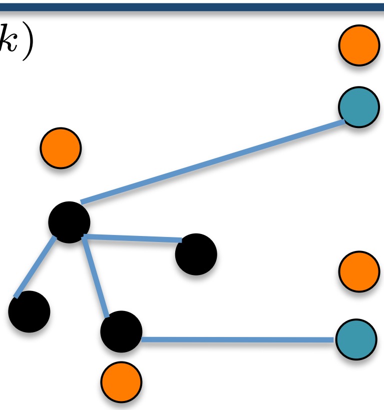
-Receive local copy X_i from k
-Update Z_i
-Send back new Z_i to k

$$\frac{\partial f}{\partial Z_i}\left[X_i^{(k)}\right] = \mu(Z_i - X_i^{(k)}).$$

$$\frac{\partial f}{\partial X_i^{(k)}} = -\sum_{j \in N(i)} \left(Y_{ij} - \langle X_i^{(k)}, X_j^{(k)} \rangle\right) X_j^{(k)} + \lambda n_i X_i^{(k)} + \mu(X_i^{(k)} - Z_i^{(k)}).$$

Synchronization thread Send

$X^{(k)}$



-Cycle through nodes
-Update local copies

Computation thread

-Cycle through nodes
- Send local copy to DSM

-Received Z_i from DSM
- update cached copy

Synchronization thread receive

Convergence

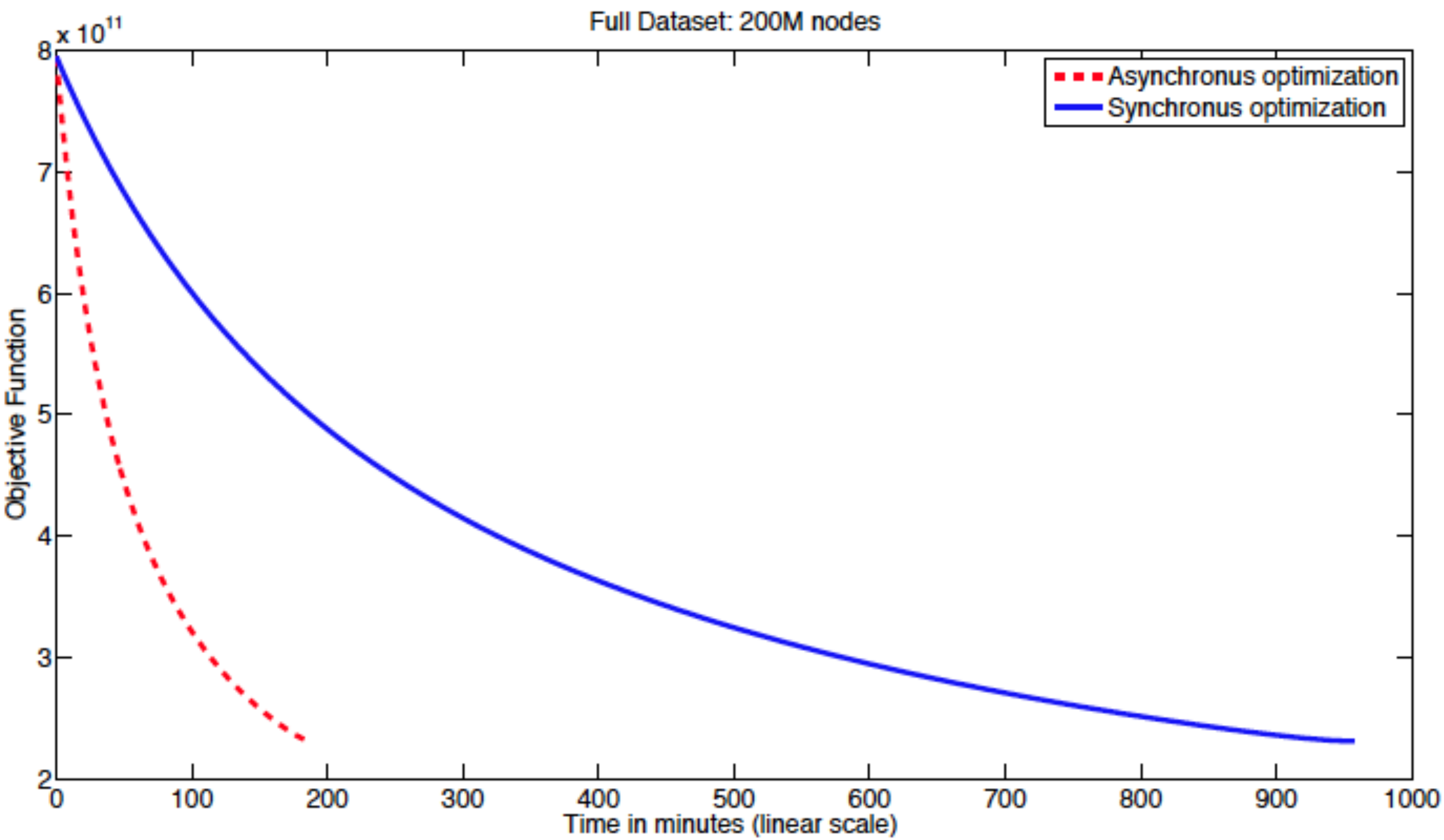
- Can be reduced to lock-free parallel SGD [Hogwild]
- Convergence is affected by
  - Synchronization rate
    - Time needed to refresh the local version of the global variable
    - Number of replicated nodes

Summary of Asynchronous

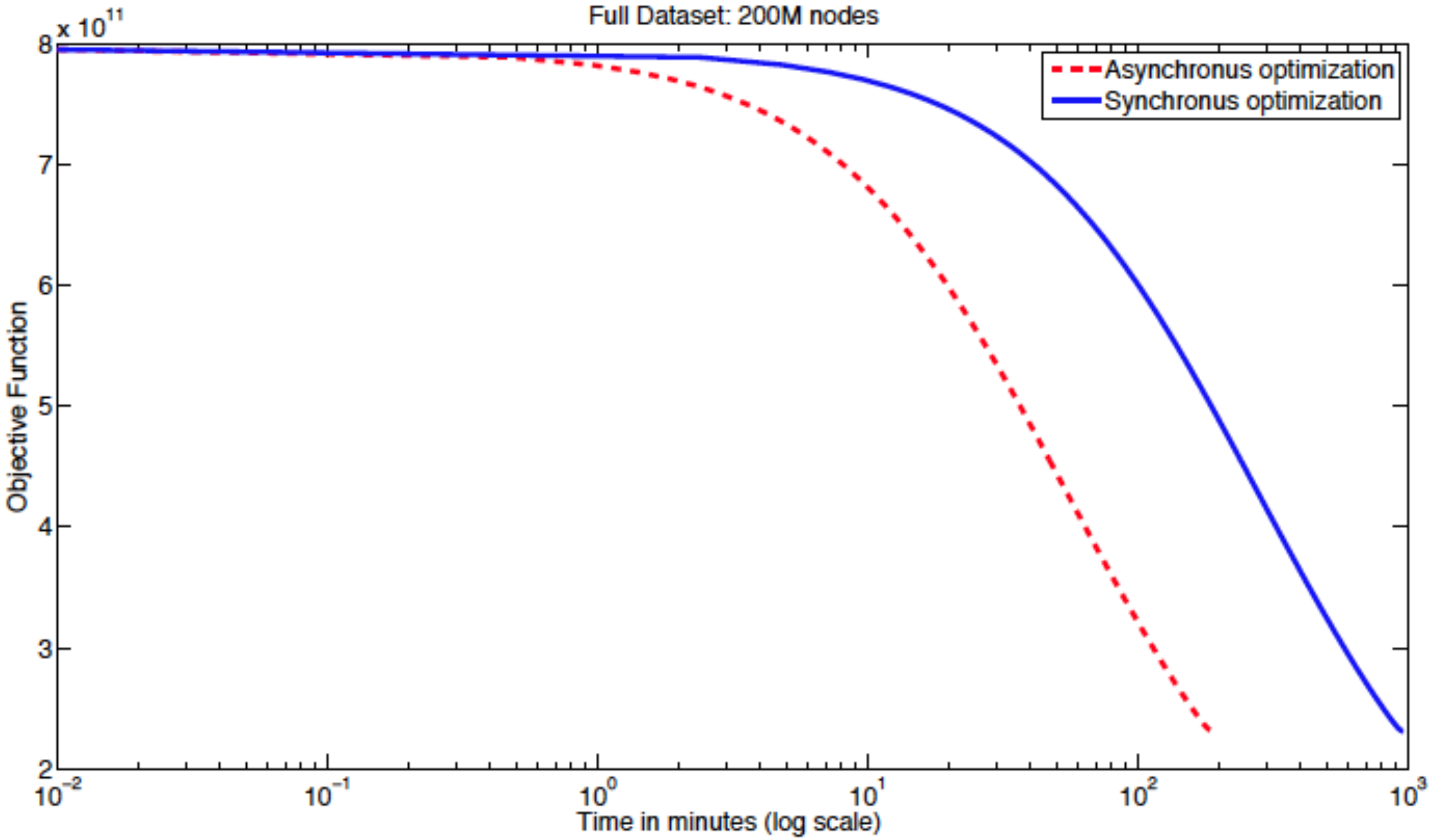
- Continuously update local variables X (via SGD)
- Continuously send local variables to global
- Continuously update global variable Z (via SGD)
- Continuously send & overwrite global variables to local

$$\sum_{k=1}^{K} f_k(Y, X^{(k)}, \lambda) + \frac{1}{2} \sum_{k=1}^{K} \left[\mu \sum_{i \in V_k} \| Z_i - X_i^{(k)} \|^2 \right]$$

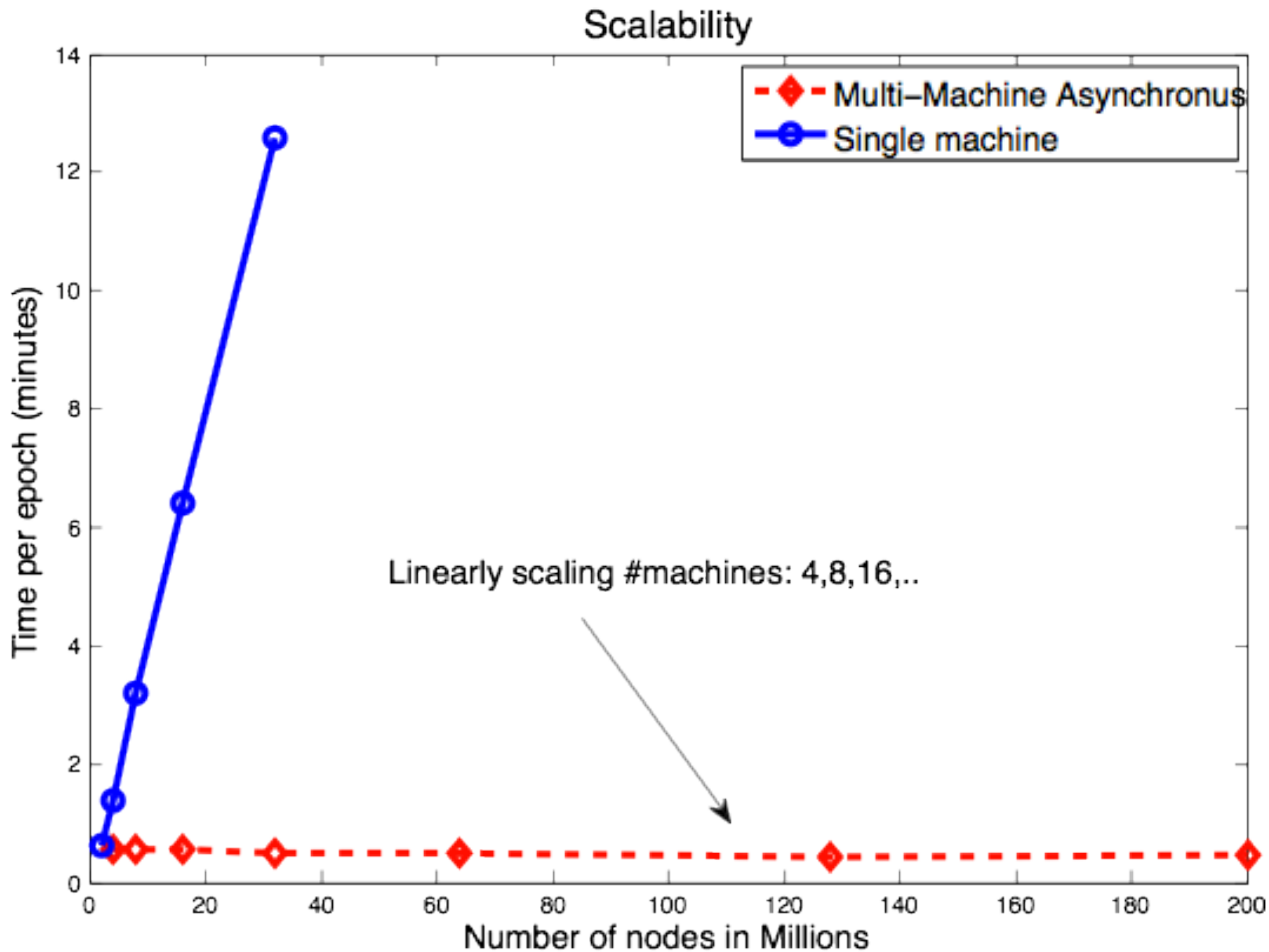
Convergence



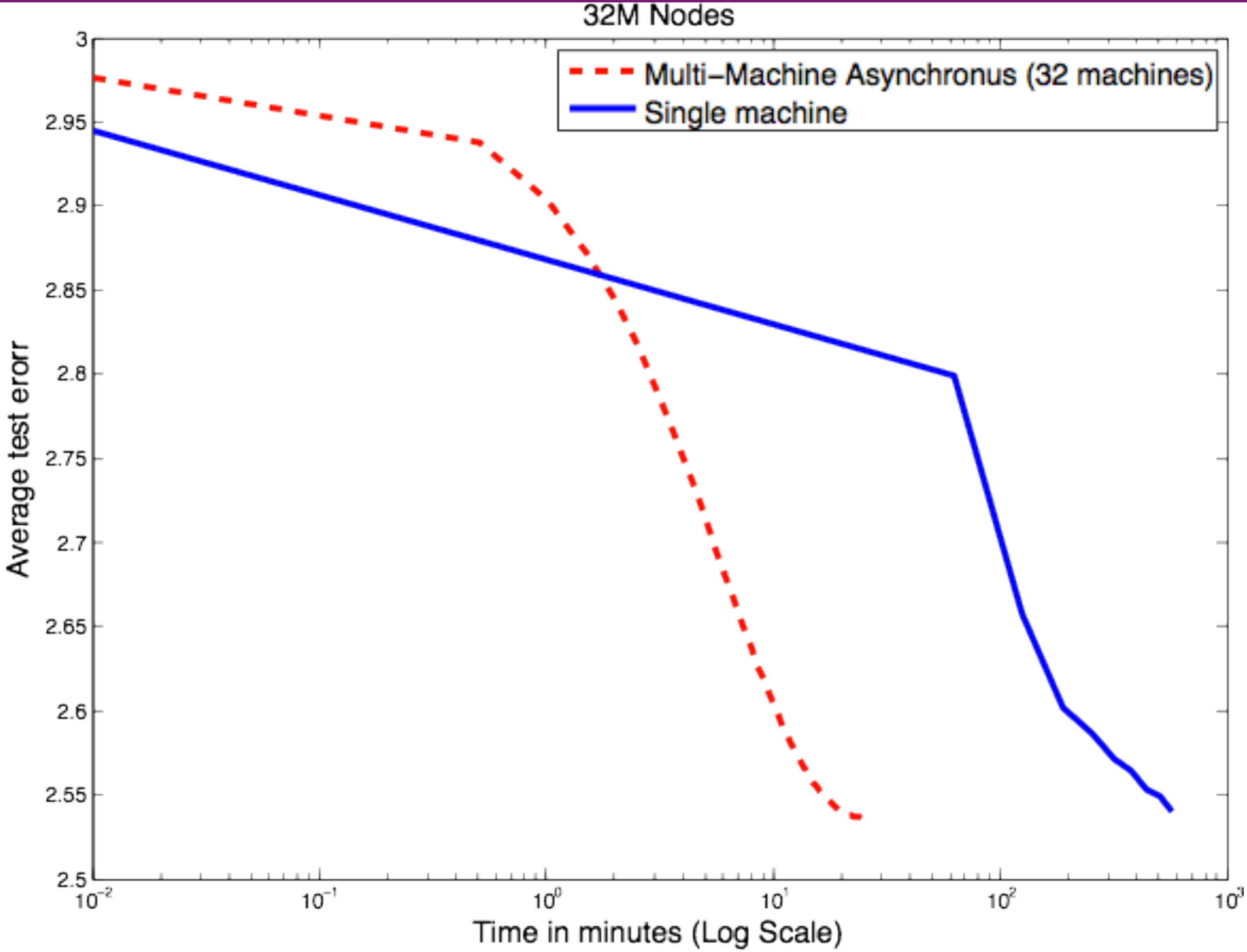
Convergence



Scalability



Solution Quality



Practical Considerations

- How to partition the graph?
  - We want to minimize the number of borrowed nodes
    - Affect convergence
    - Increases the number of deviation penalties
  - Take each machine capacity into consideration
    - Store owned nodes
    - **Borrowed** nodes
    - Cached copies of relevant global variables
- Network Optimization
  - Take network topology into account

Graph Partition

- Find a set of minimally overlapped partitions

“*Decompose the graph to minimize number of vertices + neighbors per partition*”

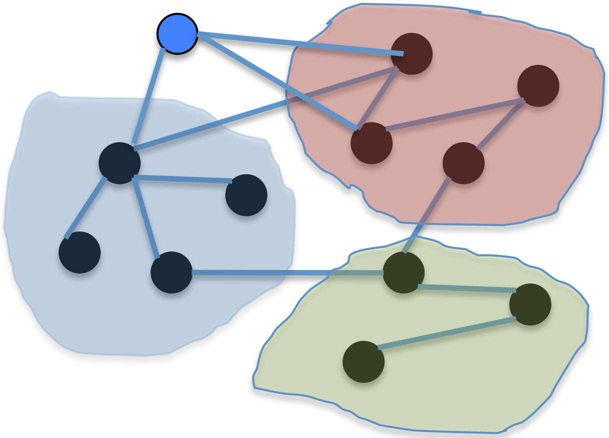
  - NP hard problem by itself [WSDM 2012]
- Under capacity constraints
- We just scratched the surface here
  - Simple greedy algorithm
  - Hierarchal extension
  - LSH and random baselines

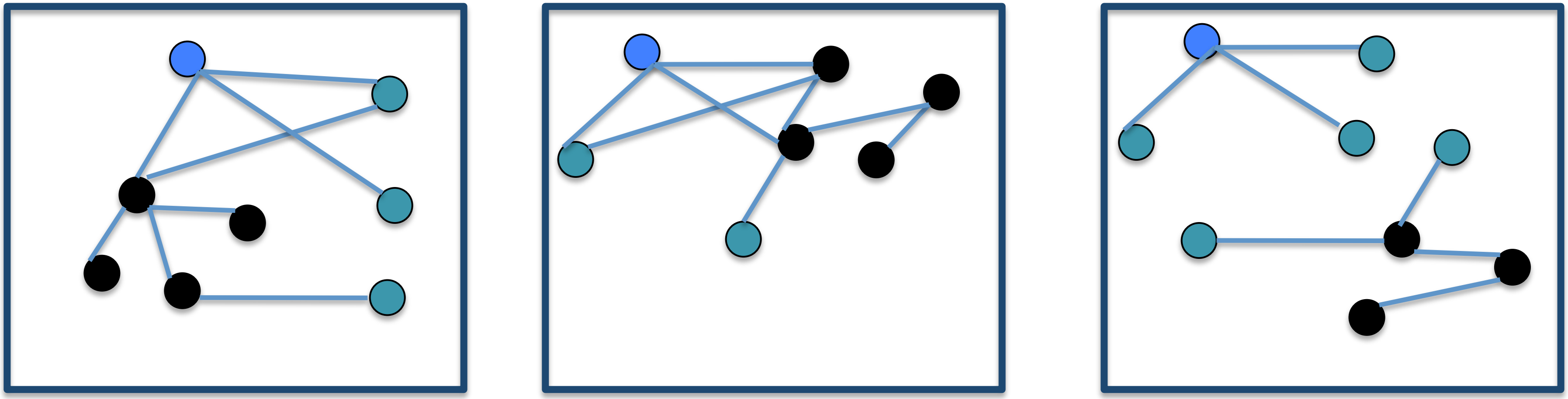
Single Pass Greedy Algorithm

- Intuitively
  - Add each node to where its neighbors are!
- Maintain a set of open partitions
  - Store the borrowed and owned nodes in each partition
- For each vertex *v*
  - For each partition *p*
    - We want to make sure that N(v) are in the same partition
    - Add N(v) / Owned(p) to borrowed of *p*
  - Select p with minimum number of borrowed nodes

Partition and Replicate

- For each vertex *v*
    - For each partition *p*
        - We want to make sure that N(v) are in the same partition
        - Add N(v) / Owned(p) to borrowed of *p*
    - Select p with minimum number of borrowed nodes





Hierarchical Extension

- Two step approach
  - First run greedy with small number of partitions
  - Second, run greedy over the first level partitions
- Time is proportional to number of open partitions
  - Divide and conquer

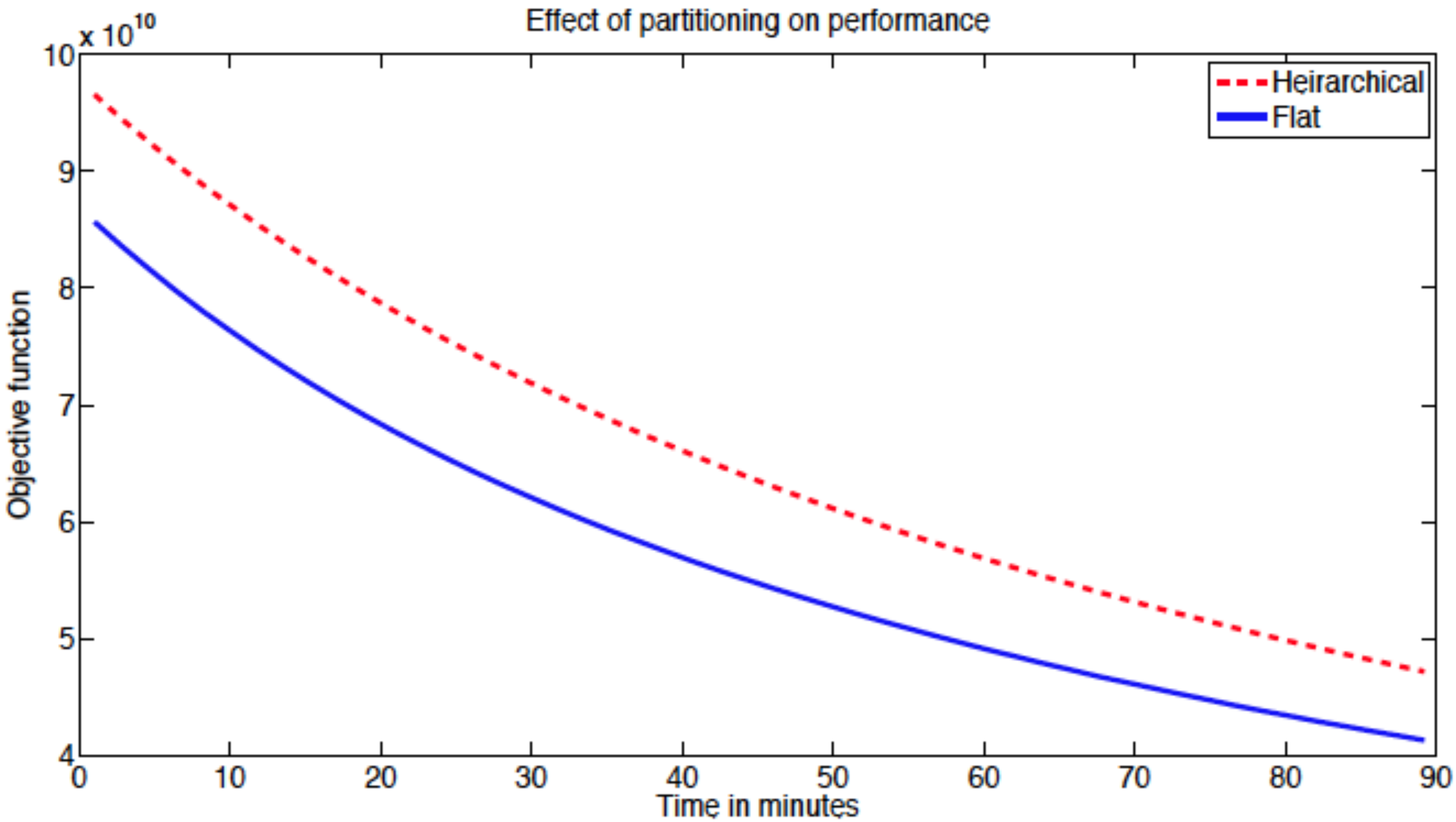
Baselines

- Radom
- LSH-based
  - LSH over adjacency matrix
  - Related to shingle-based graph compression approaches
- Metrics
  - Time to perform partitioning
  - Quality of partitions
    - Number of borrowed nodes
    - Time to perform a full synchronization cycle

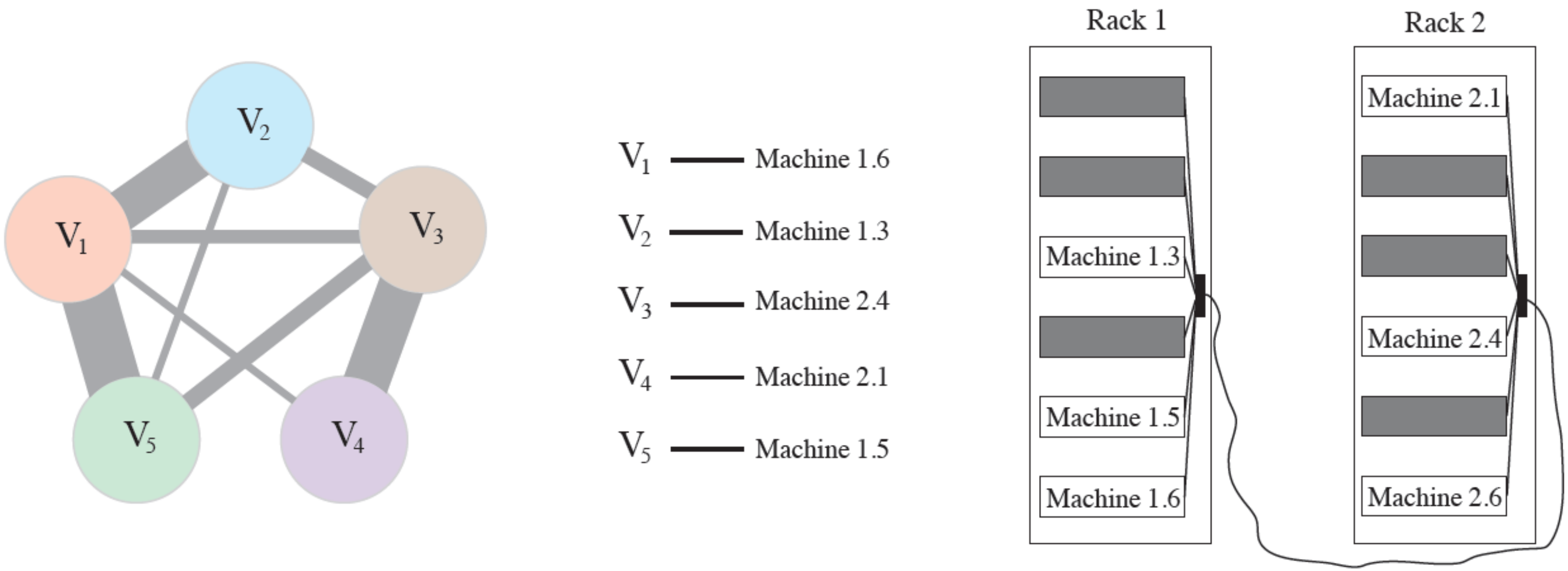
The Effect of Partitioning Quality

Method	Total borrowed nodes (millions)	Partitioning time (minutes)	Sync time (seconds)
Flat	252.31	166	71.5
Hierarchical	392.33	48.67	85.9
Hier-LSH	640.67	17.8	136.1
Hier-Random	720.88	11.6	145.2

The Effect of Partitioning Quality



Network Optimization

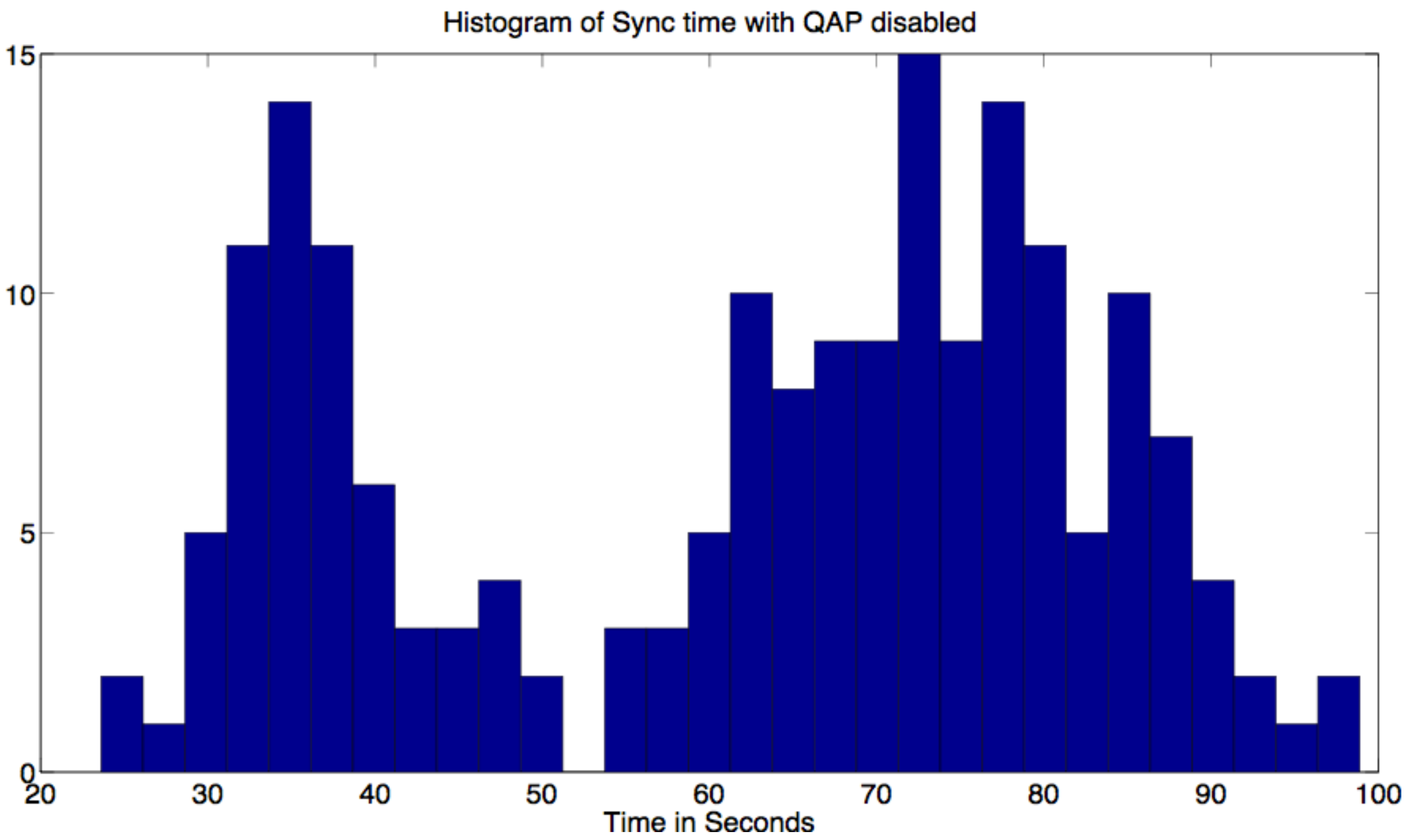


Network Optimization

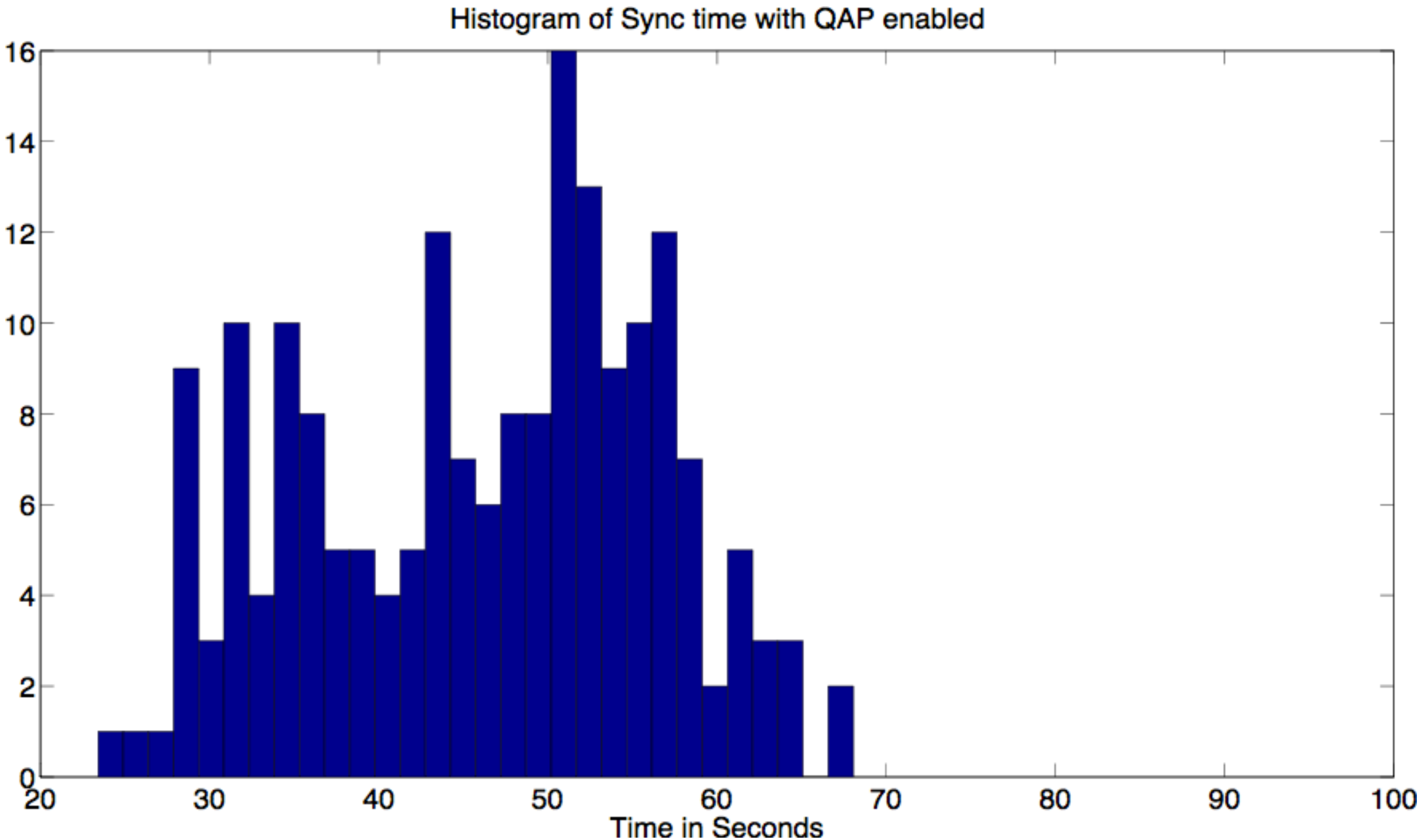
- We only know the layout at run time
  - Inverse network bandwidth D
- Inter-partitions communication
  - Communication requirement C
  - The more overlap, the higher is C
- Solve a quadratic assignment problem

$$T(\pi) = \sum_{kl} C_{kl} D_{\pi(k)\pi(l)} = \sum_{kl} C_{kl} \sum_{uv} \pi_{ku} \pi_{lv} D_{uv} = \operatorname{tr} C \pi D \pi^{\top}$$

Sync time without QAP



Sync time with QAP



Summary

- Model as consensus problem
- Synchronous algorithms
  - Curse of the last reducer
- Asynchronous algorithm
  - Asynchronous parallel updates
  - Network topology optimization
  - Overlapping partitions

Future Directions

Future Directions

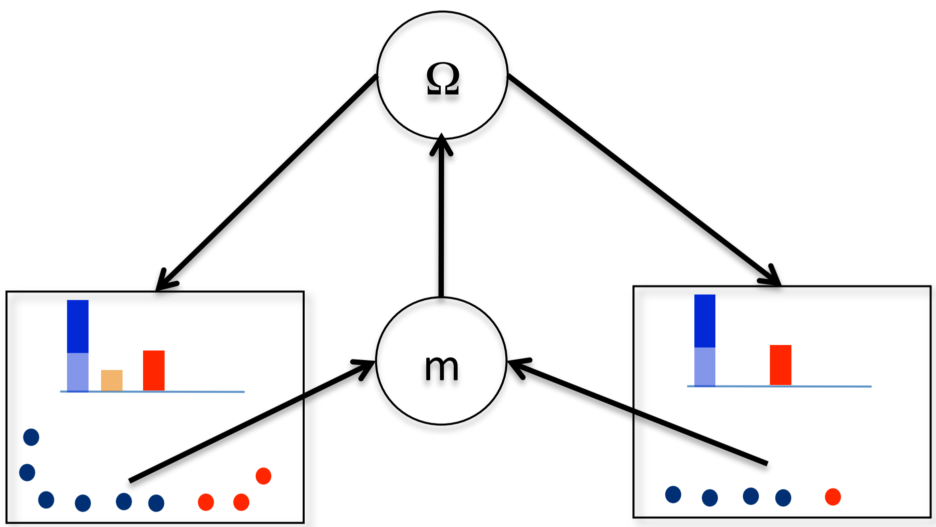
- Theoretical bounds and guarantees
- Non-parametric models
  - Learning structure from data
- Working under communication constraints
- A new release of Yahoo! LDA
- More applications
  - Citation analysis
    - Graph factorization + LDA

Questions?

Sampling Ω

- Introduce auxiliary variable m_{kt}
  - How many times the global distribution was visited
  - $P(m_k^t | n_{1k}^t, \cdots, n_{ik}^t, \cdots)$ ~ AnotniaK

$$P(\Omega^t | \mathbf{m}^t, \tilde{\mathbf{m}}^t) \sim \mathrm{Dir}(\tilde{\mathbf{m}}^t + \mathbf{m}^t + \alpha/K)$$



$$\left(n_{ik}^{t,-j} + \tilde{n}_{ik}^t + \lambda\Omega^t\right)$$

Distributed Sampling Cycle