Understanding and Improving Database-backed Applications

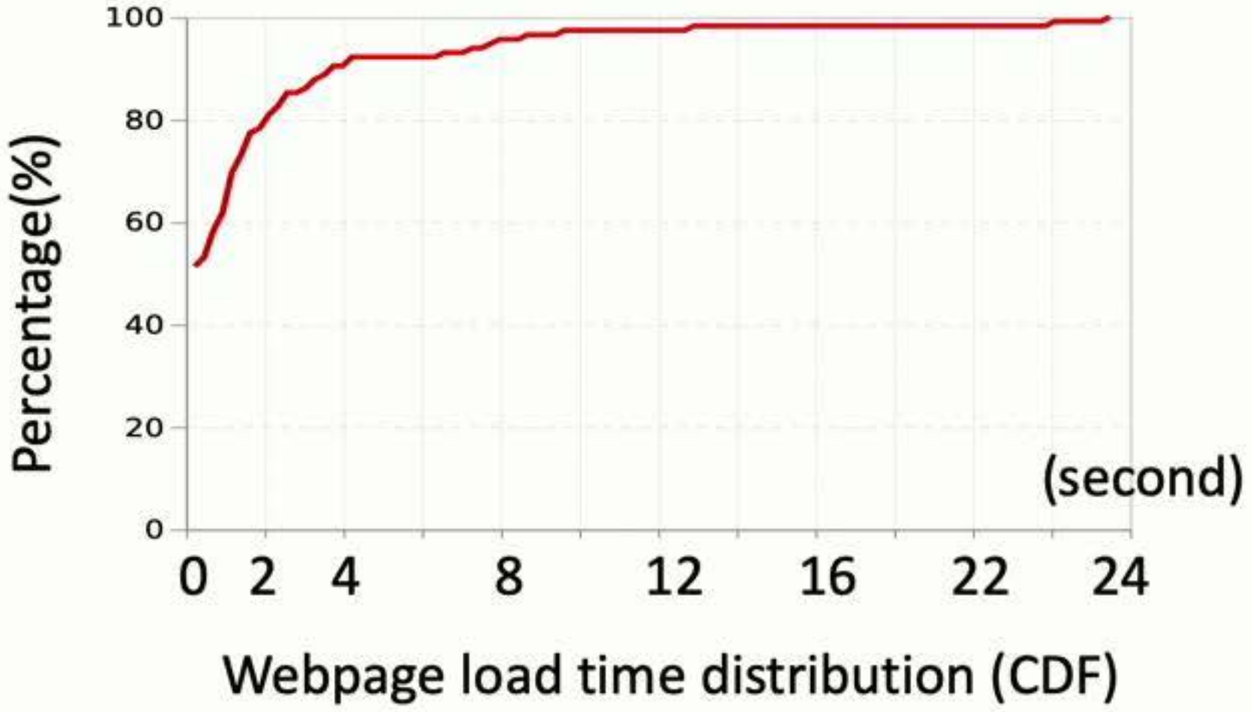
Cong Yan

University of Washington

Real-World Application Performance

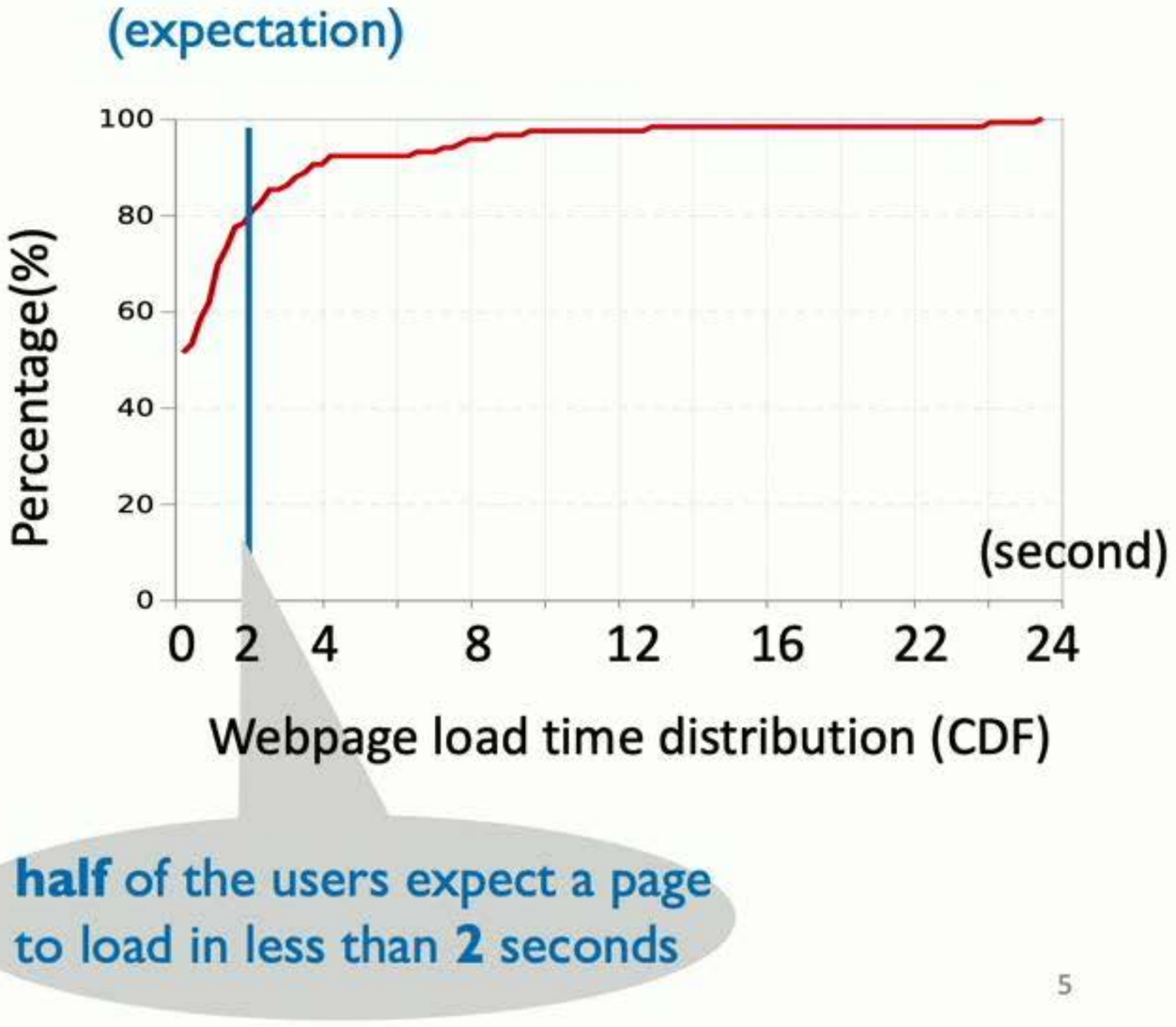
Real-World Application Performance

Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	2.4k
Gitlab (collaboration)	49k
Redmine (collaboration)	3.6k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit (map)	1.1k



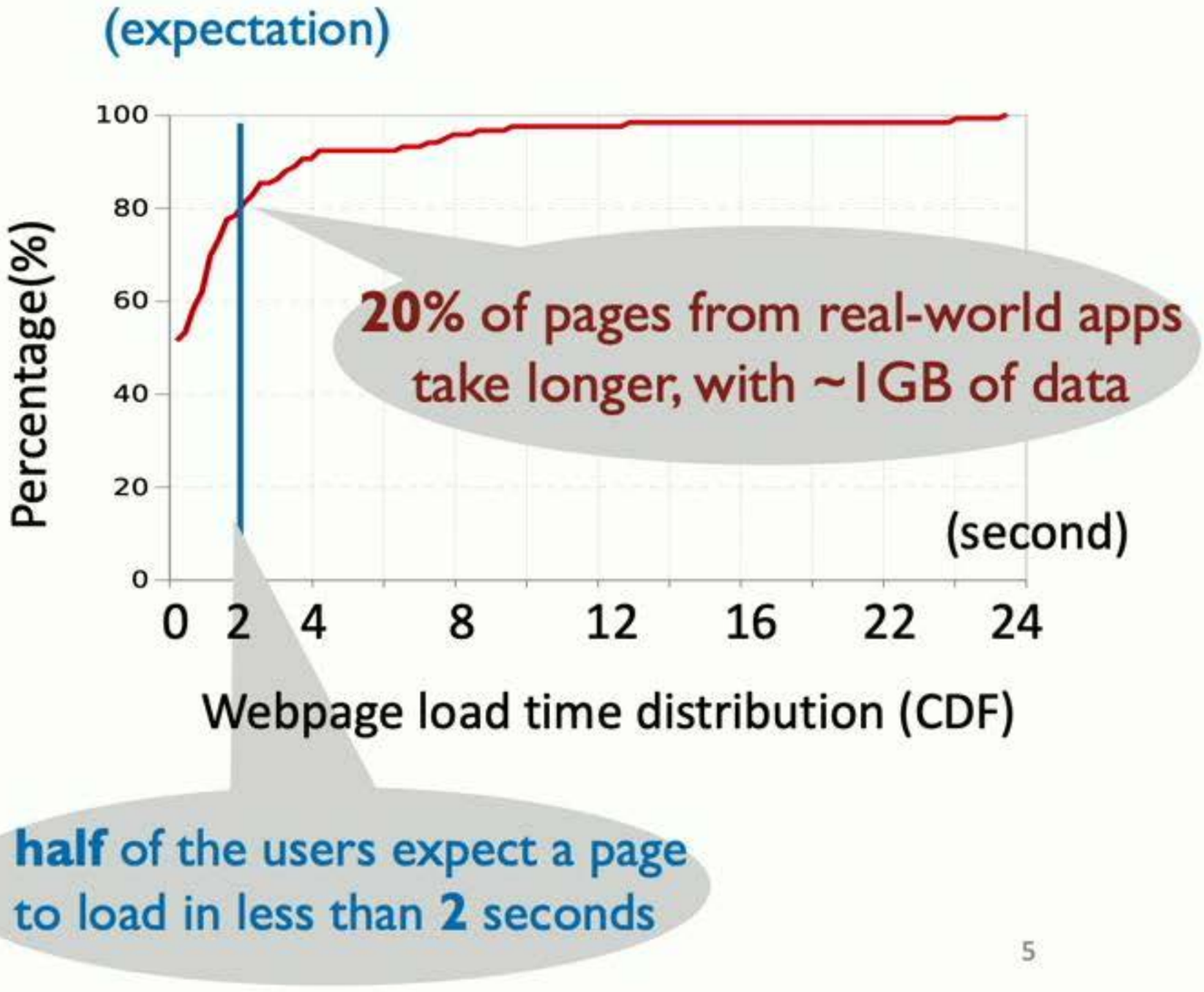
Real-World Application Performance

Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	2.4k
Gitlab (collaboration)	49k
Redmine (collaboration)	3.6k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit (map)	1.1k



Real-World Application Performance

Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	2.4k
Gitlab (collaboration)	49k
Redmine (collaboration)	3.6k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit (map)	1.1k



Decades of Research on Optimizing Each Layer

Web cache

WebAssembly

JavaScript JIT

…

Dead code elimination

Vectorization

Object caching

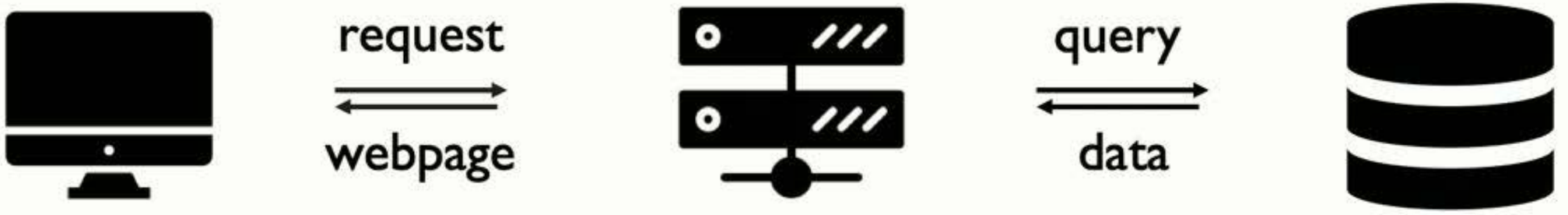
…

Query optimization

Physical design

Concurrency control

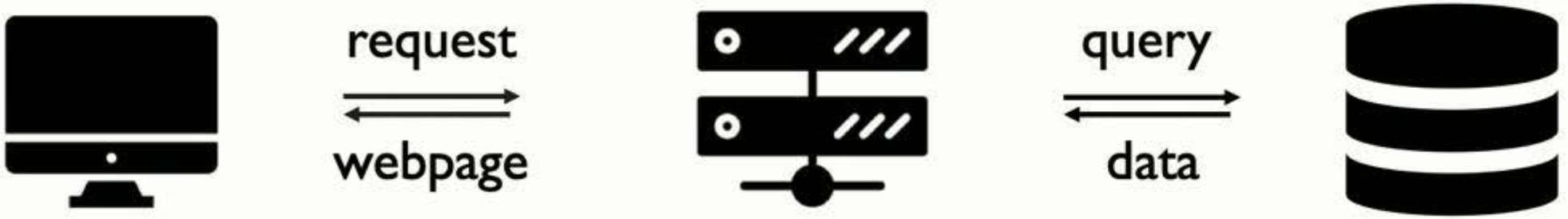
…



Key Insight: Leveraging Application Semantics

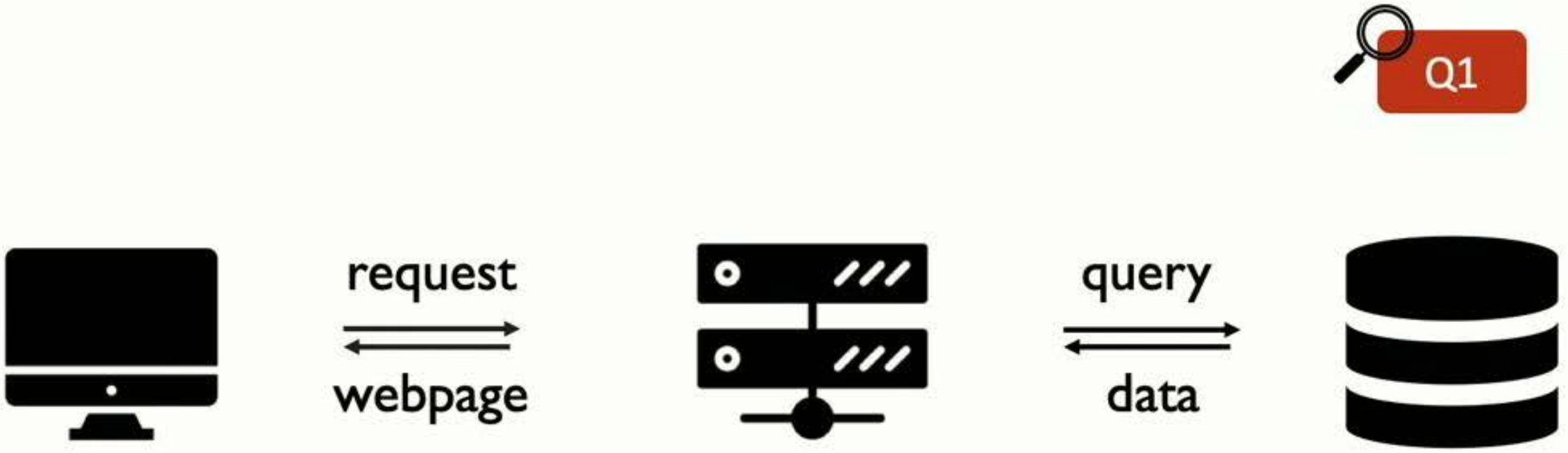
Leveraging Application Semantics To Optimize Each Layer

★ Leverage how query results are used in the application



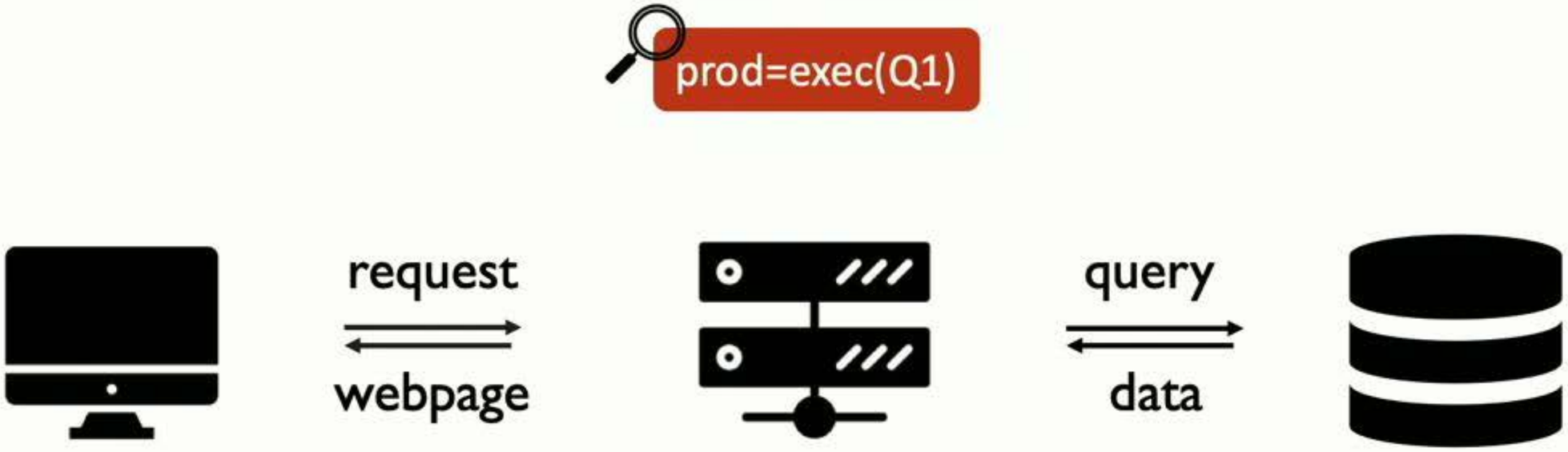
Leveraging Application Semantics To Optimize Each Layer

★ Leverage how query results are used in the application



Leveraging Application Semantics To Optimize Each Layer

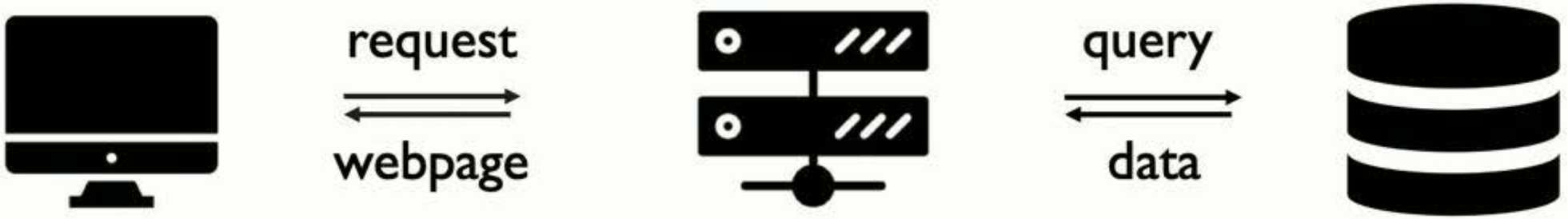
★ Leverage how query results are used in the application



Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

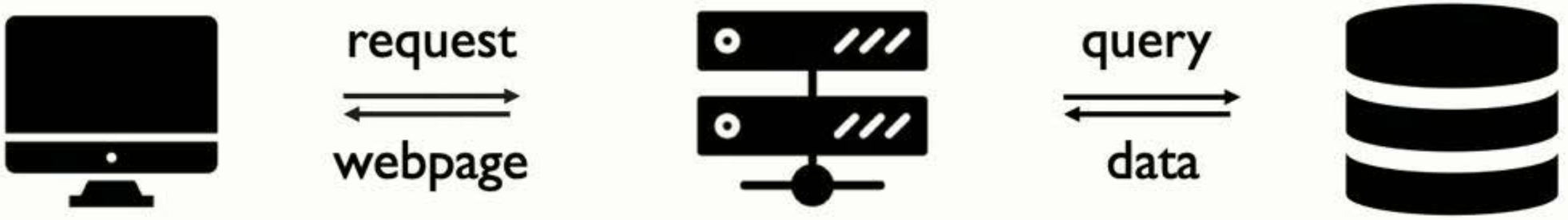
★ Leverage how query results are used in the application



Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

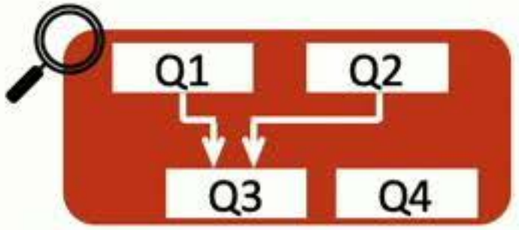
★ Leverage how query results are used in the application

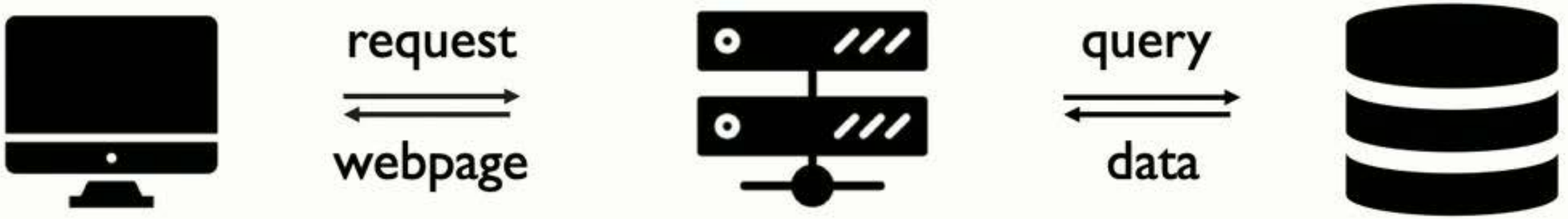


Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



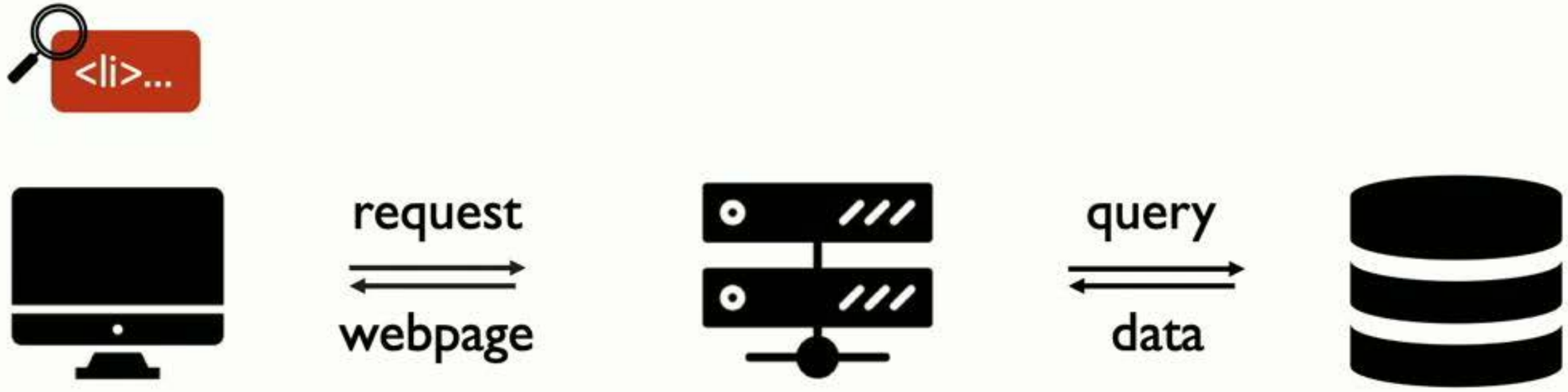


Leveraging Application Semantics To Optimize Each Layer

★ Leverage the data retrieval behind each webpage data

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application

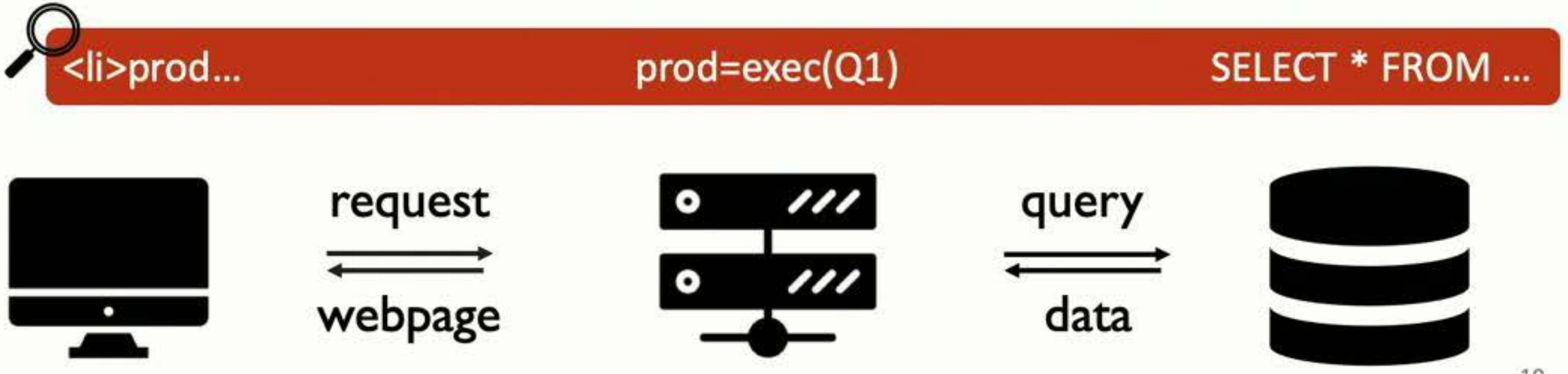


Leveraging Application Semantics To Optimize Each Layer

★ Leverage the data retrieval behind each webpage data

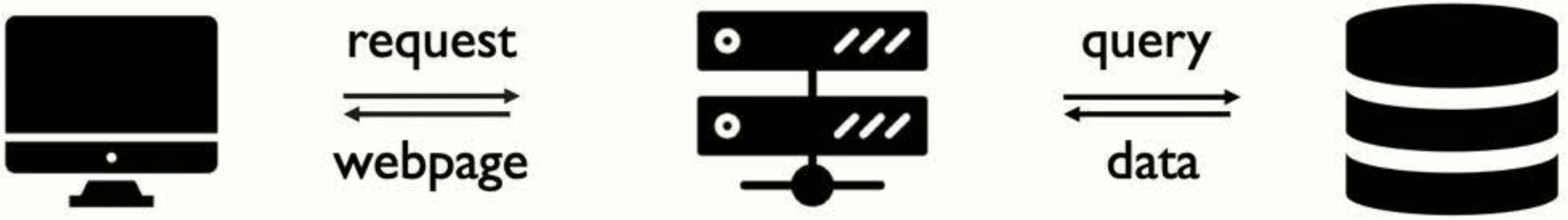
★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



Leveraging Application Semantics To Optimize Each Layer

Panorama: view-driven optimization	Quro: reorder queries	Chestnut: customize data layout
★ Leverage the data retrieval behind each webpage data	★ Leverage how queries are connected in the application	★ Leverage how query results are used in the application



Outline

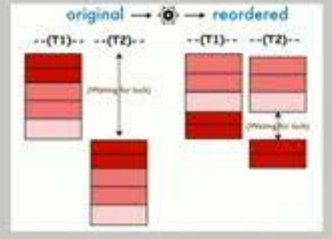
- Leveraging application semantics to optimize each layer

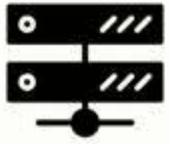
[CIKM'17, FSE'18, ICSE'18, ICSE'19, CIDR'20]

Panorama:
view-driven optimization

[VLDB'16]

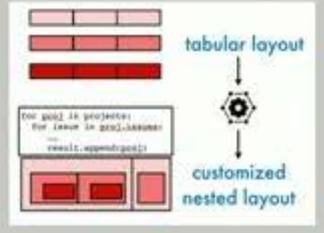
Quro:
reorder queries





[VLDB'19]

Chestnut:
customize data layout



Outline

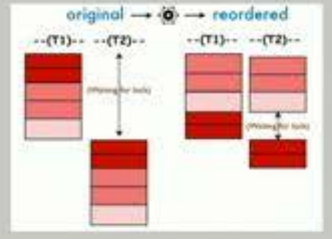
- Leveraging application semantics to optimize each layer

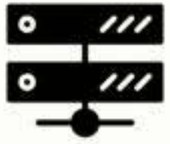
[CIKM'17, FSE'18, ICSE'18, ICSE'19, CIDR'20]

Panorama:
view-driven optimization

[VLDB'16]

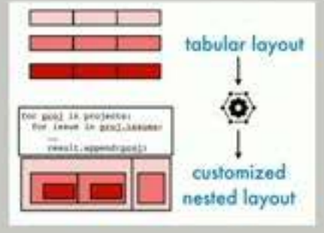
Quro:
reorder queries





[VLDB'19]

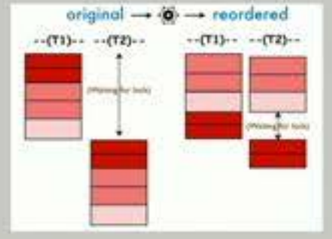
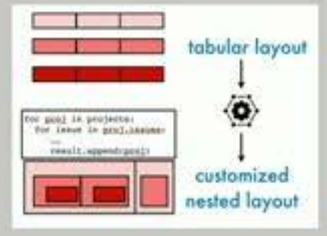
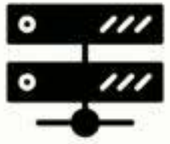
Chestnut:
customize data layout



- Other projects
- Ongoing and future work

Outline

- Leveraging application semantics to optimize each layer

Panorama: view-driven optimization	Quro: reorder queries	Chestnut: customize data layout
		
		

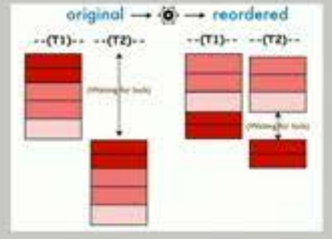
- Other projects
- Ongoing and future work

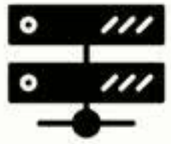
Outline

- Leveraging application semantics to optimize each layer

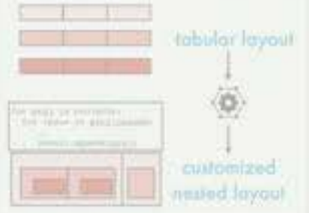
Panorama:
view-driven optimization

Quro:
reorder queries



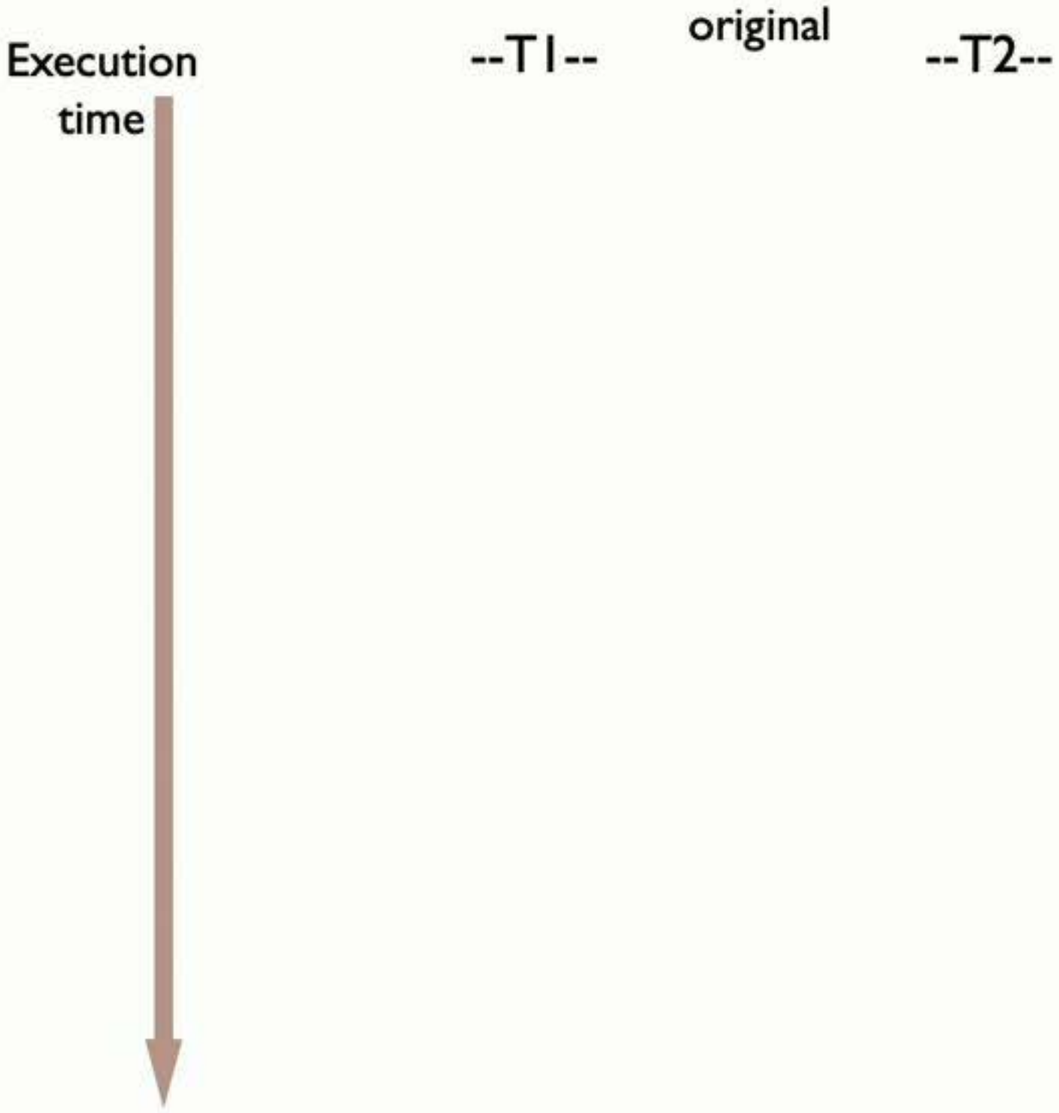


Chestnut:
customize data layout

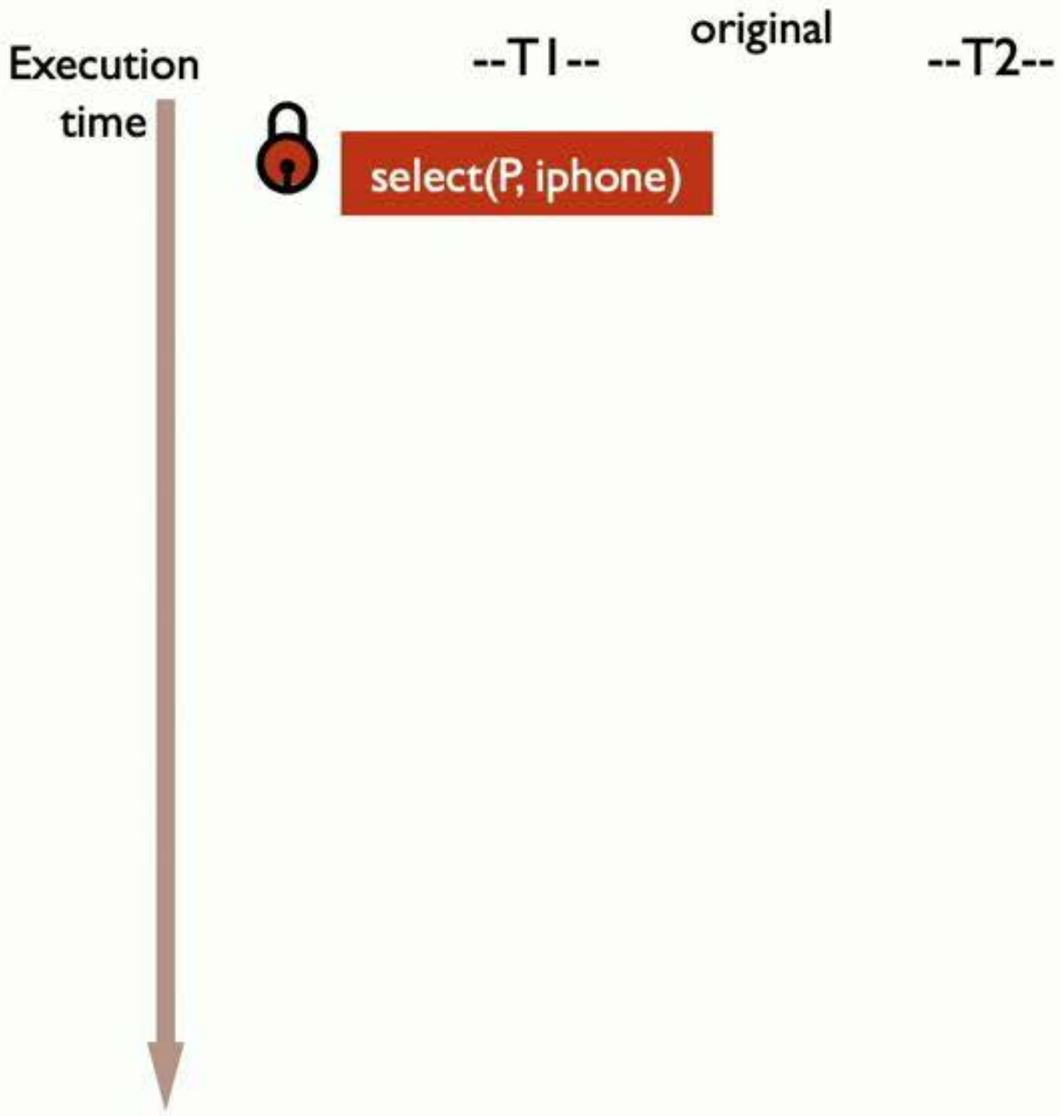


- Other projects
- Ongoing and future work

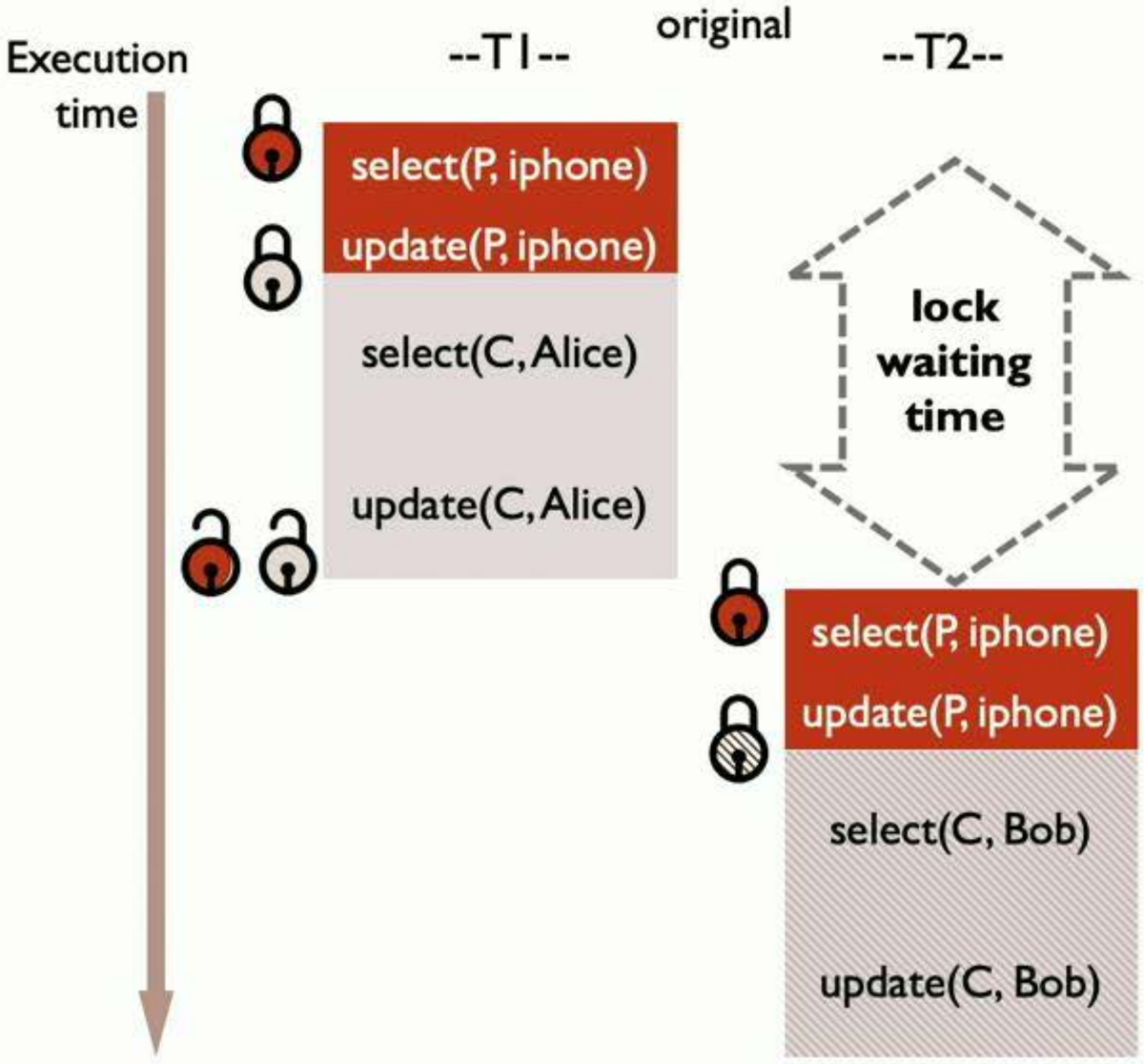
Query Reorder under 2PL



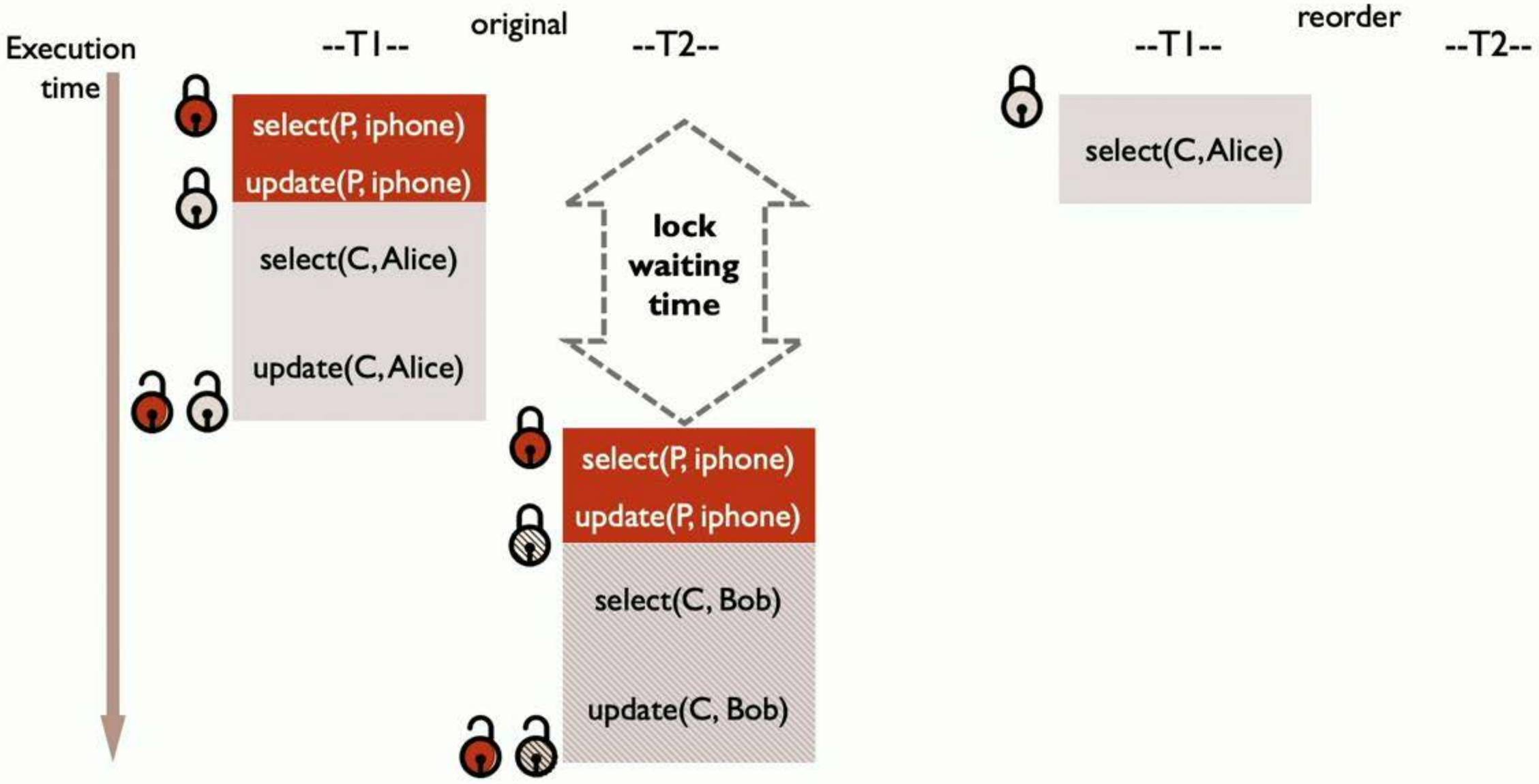
Query Reorder under 2PL



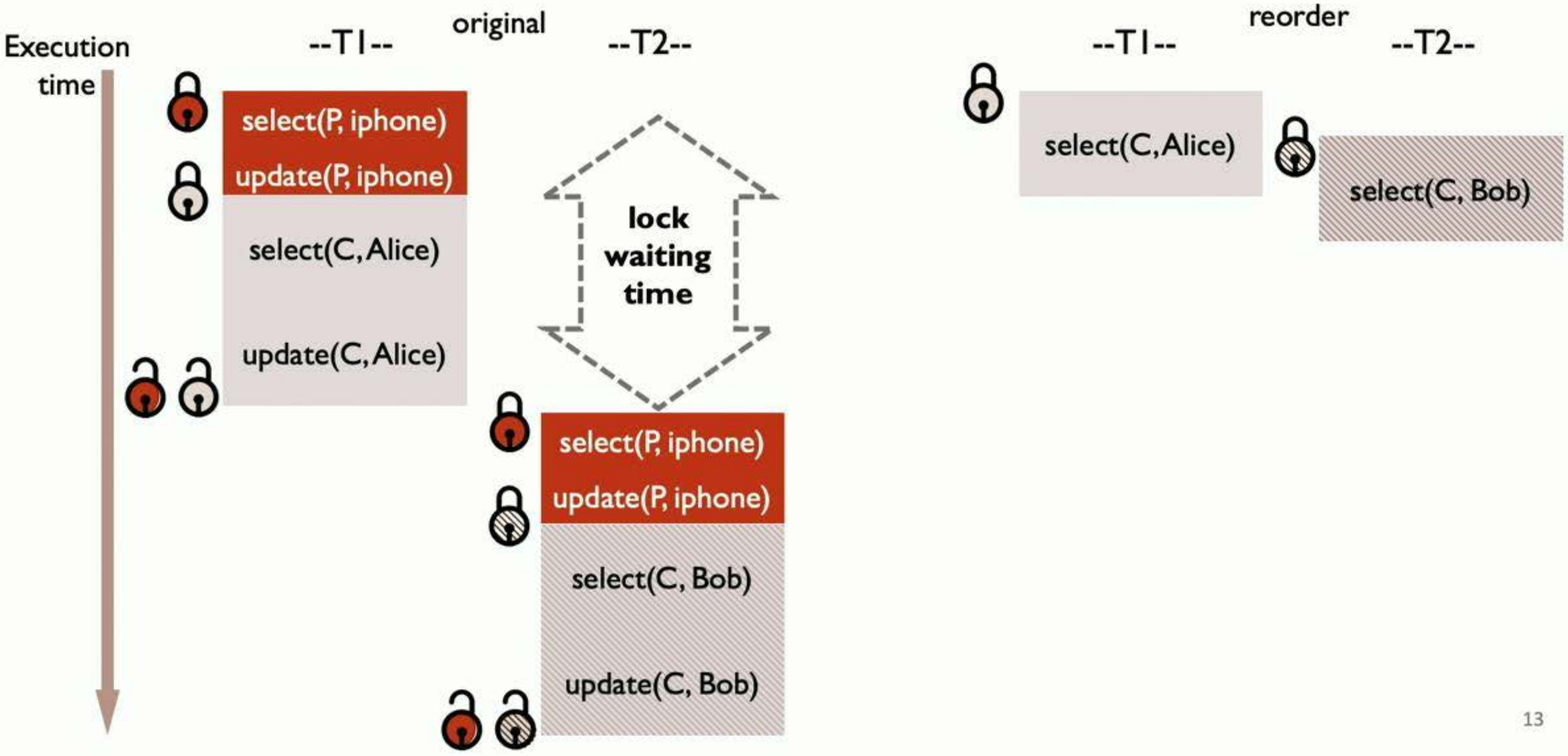
Query Reorder under 2PL



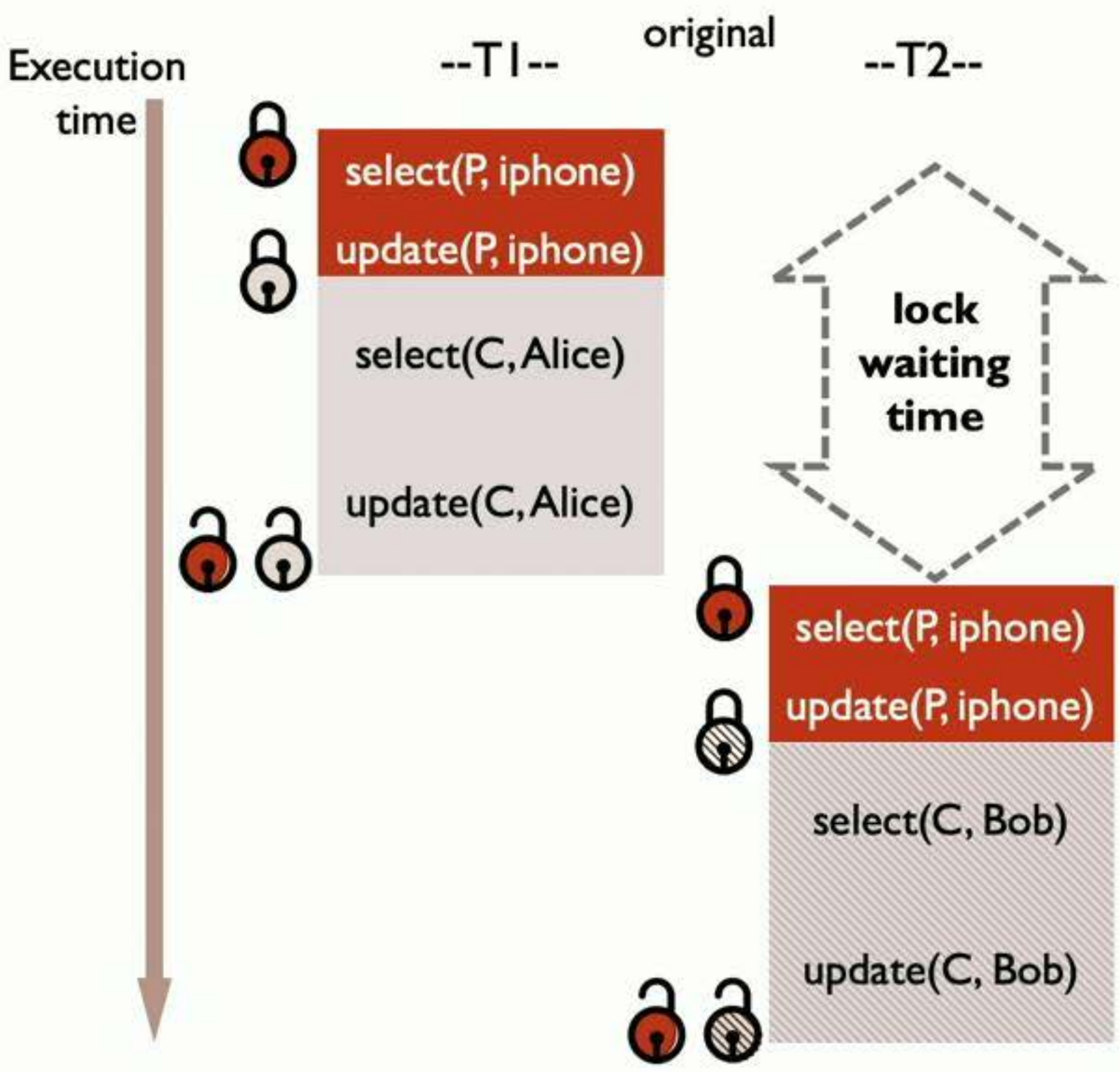
Query Reorder under 2PL

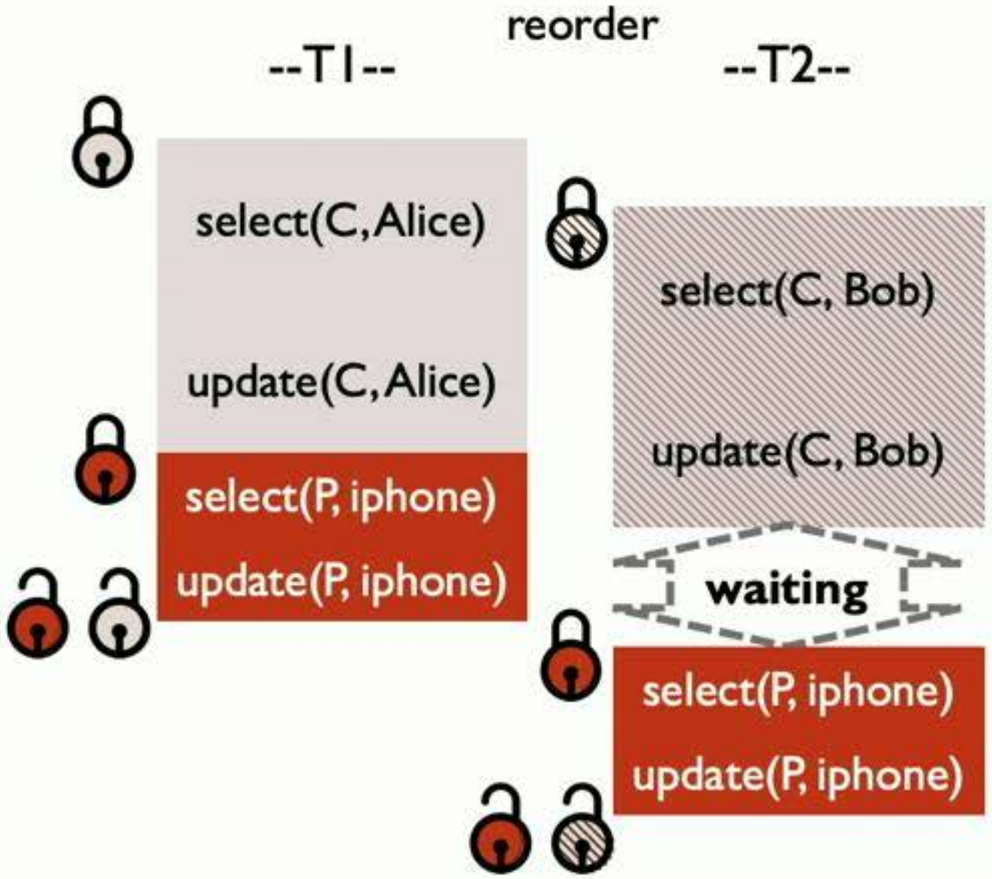


Query Reorder under 2PL

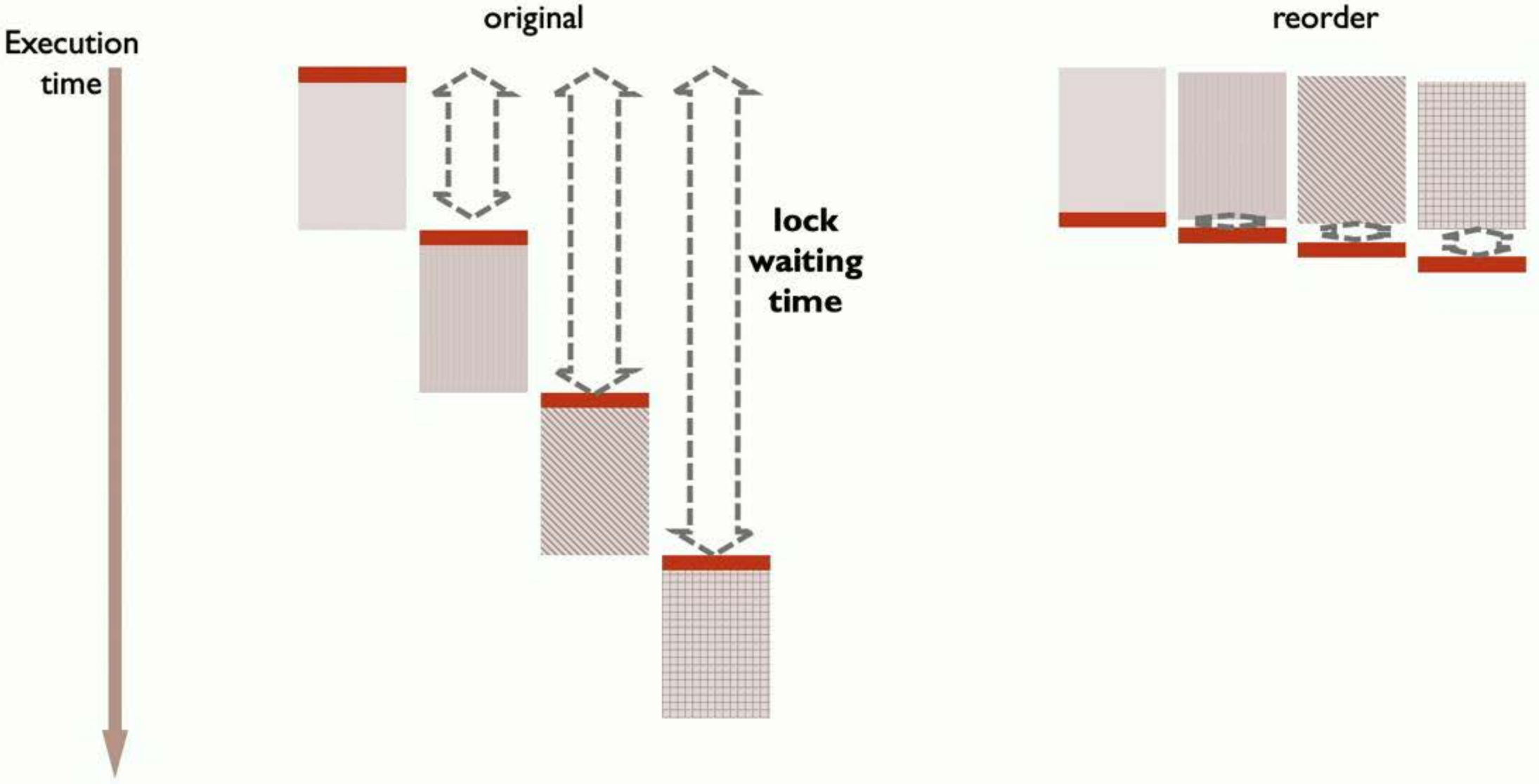


Query Reorder under 2PL





Query Reorder Shortens Lock Wait Time



Quro: Compiler for Query Reorder

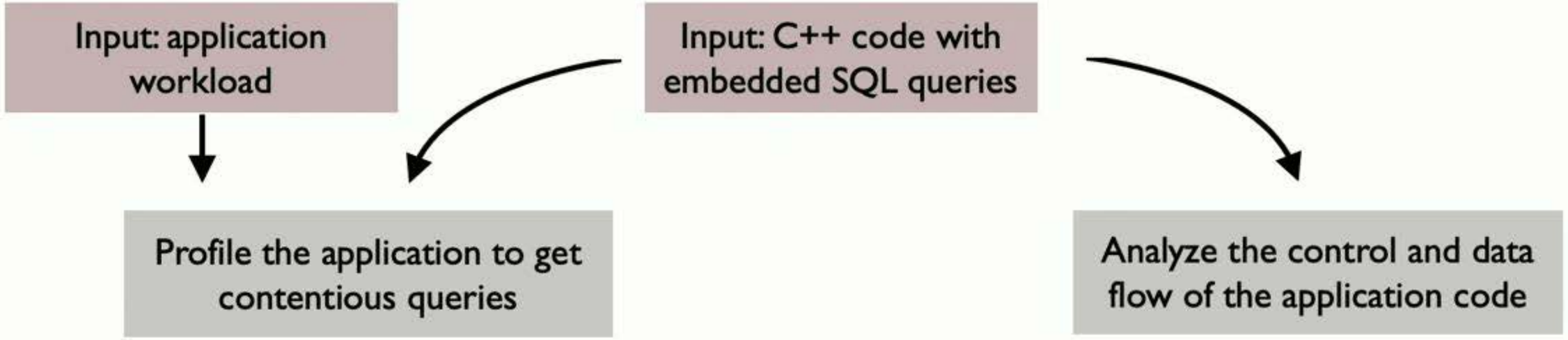
Input: C++ code with embedded SQL queries

Quro: Compiler for Query Reorder

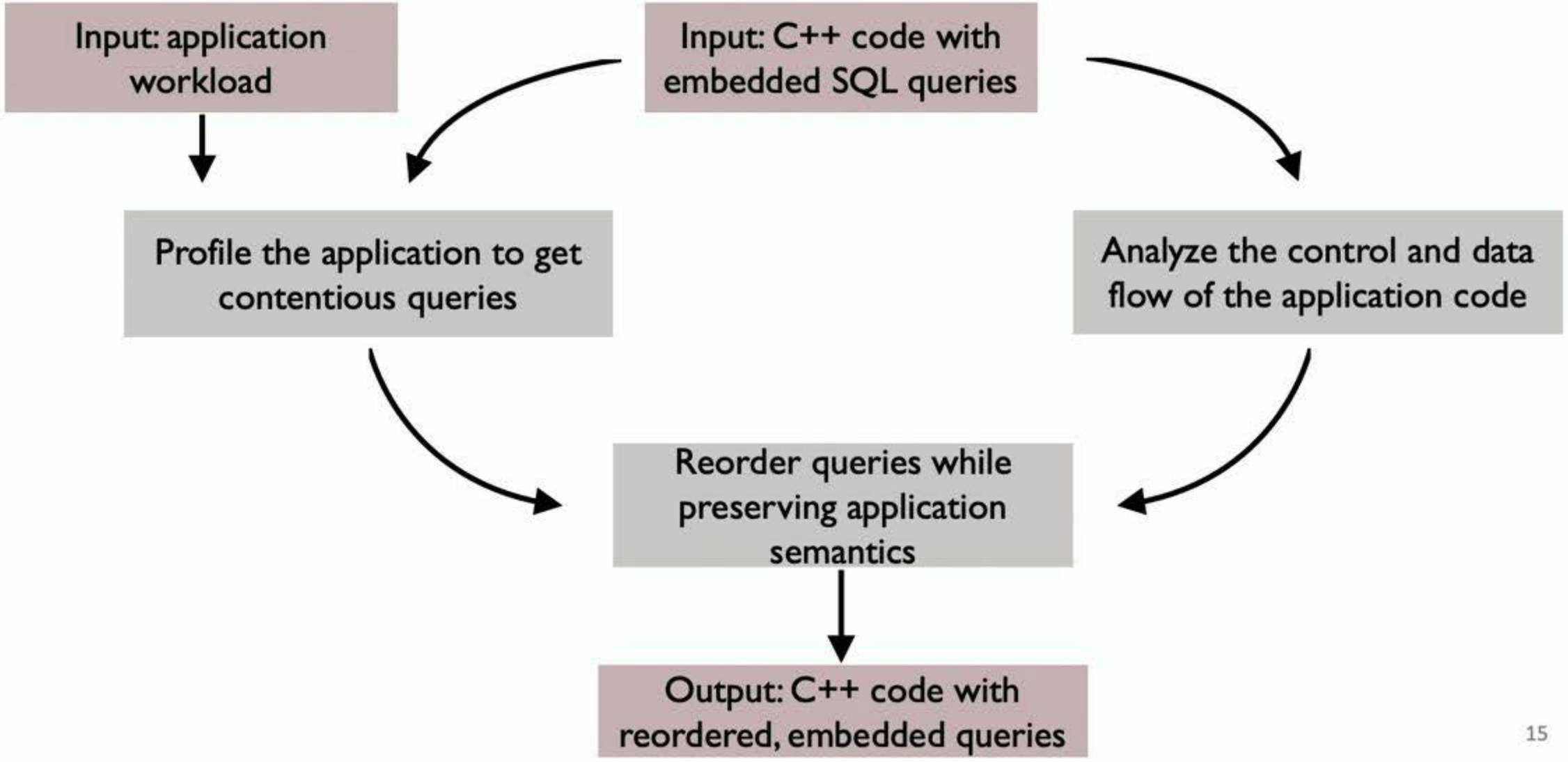
Input: C++ code with embedded SQL queries

Analyze the control and data flow of the application code

Quro: Compiler for Query Reorder



Quro: Compiler for Query Reorder



Quro: Compiler for Query Reorder

Profile the application to get contentious queries

Analyze the control and data flow of the application code

Reorder queries while preserving application semantics

Quantify contention level

- Calculate standard deviation of query running time
- Larger the deviation, more likely to have data conflict

Quro: Compiler for Query Reorder

Profile the application to get contentious queries

Analyze the control and data flow of the application code

Reorder queries while preserving application semantics

Quantify contention level

- Calculate standard deviation of query running time
- Larger the deviation, more likely to have data conflict

Data dependency analysis

- Data dependency among program variables

```
1. v1 = select(table1);
2. v2 = select(table2, v1);
3. update(table1, v1)
```

Quro: Compiler for Query Reorder

Profile the application to get contentious queries

Analyze the control and data flow of the application code

Reorder queries while preserving application semantics

Quantify contention level

- Calculate standard deviation of query running time
- Larger the deviation, more likely to have data conflict

Data dependency analysis

- Data dependency among program variables
- Database constraints

```
1. v1 = select(table1);
2. v2 = select(table2, v1);
3. update(table1, v1)
```

Quro: Compiler for Query Reorder

Profile the application to get contentious queries

Quantify contention level

- Calculate standard deviation of query running time
- Larger the deviation, more likely to have data conflict

Analyze the control and data flow of the application code

Data dependency analysis

- Data dependency among program variables
- Database constraints

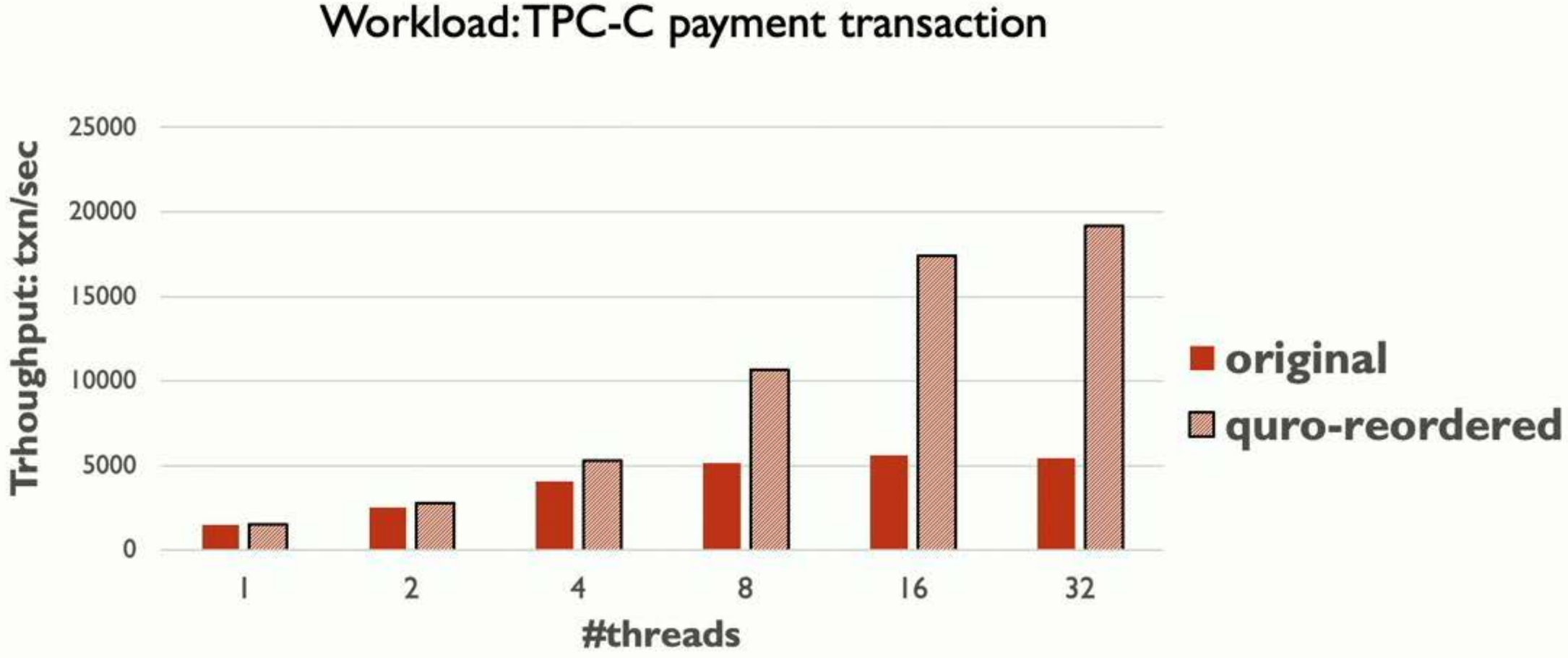
```
1. v1 = select(table1);
2. v2 = select(table2, v1);
3. update(table1, v1)
```

Reorder queries while preserving application semantics

Formalize reorder problem

- Formalize into ILP problem
- Constraint:
  data dependency constraints
- Goal:
  make contentious query appear later in a transaction

Evaluation: Quro Vs. Original



Latency: decrease up to 70%

Quro: Compiler for Query Reorder

Profile the application to get contentious queries

Quantify contention level

- Calculate standard deviation of query running time
- Larger the deviation, more likely to have data conflict

Analyze the control and data flow of the application code

Data dependency analysis

- Data dependency among program variables
- Database constraints

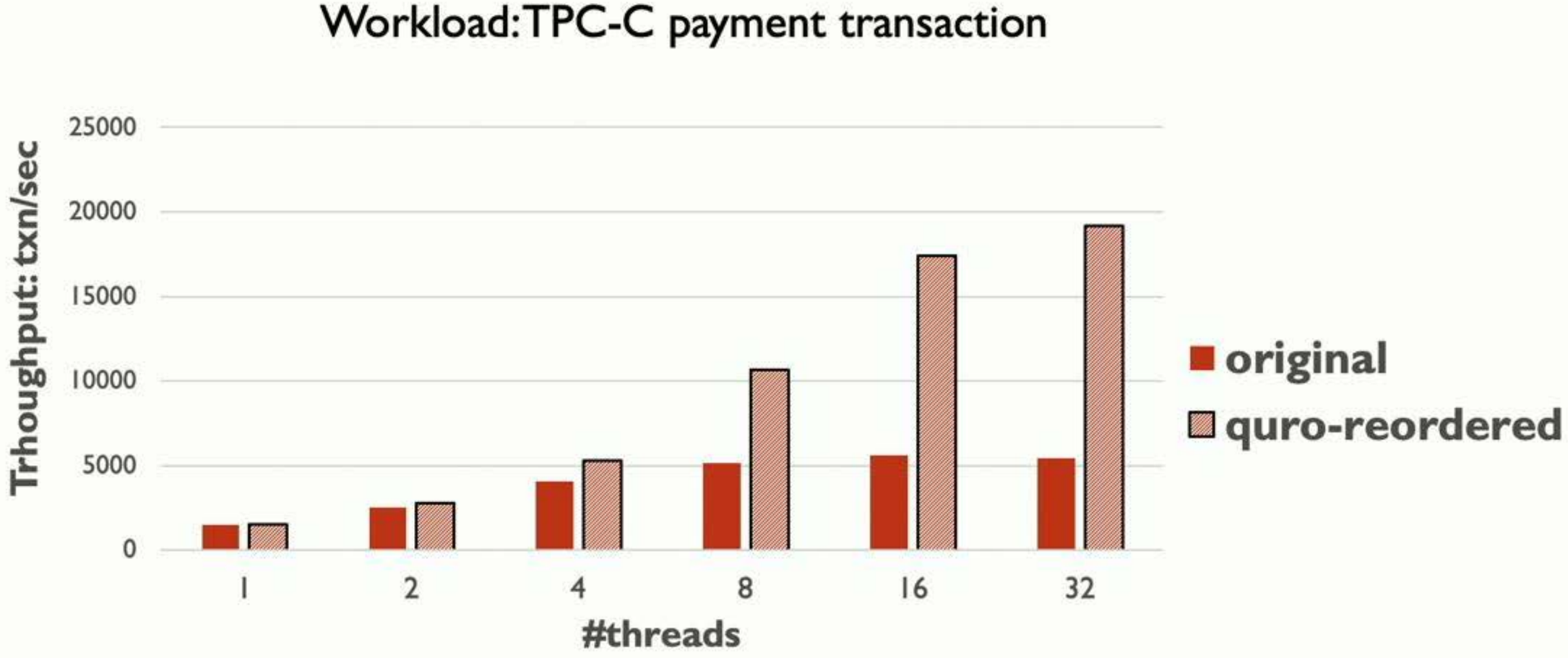
```
1. v1 = select(table1);
2. v2 = select(table2, v1);
3. update(table1, v1)
```

Reorder queries while preserving application semantics

Formalize reorder problem

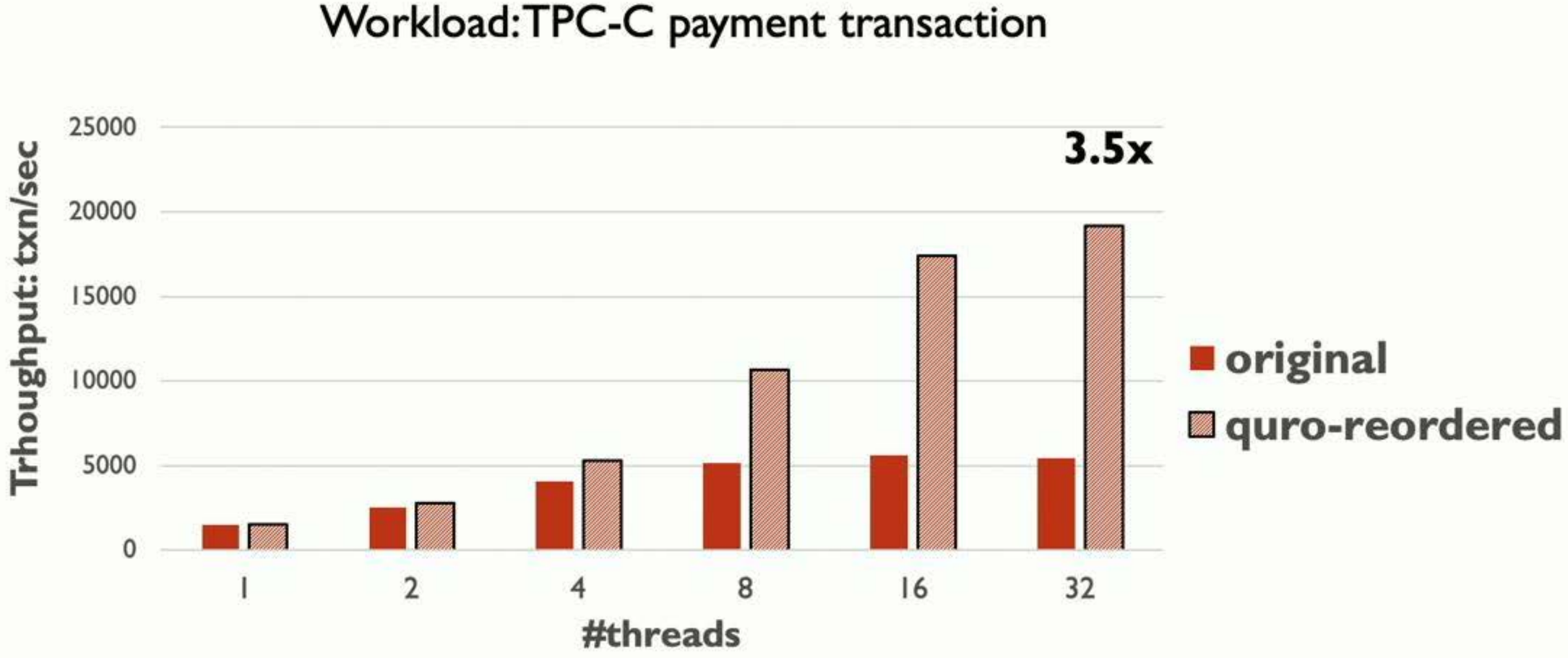
- Formalize into ILP problem
- Constraint:
  data dependency constraints
- Goal:
  make contentious query appear later in a transaction

Evaluation: Quro Vs. Original



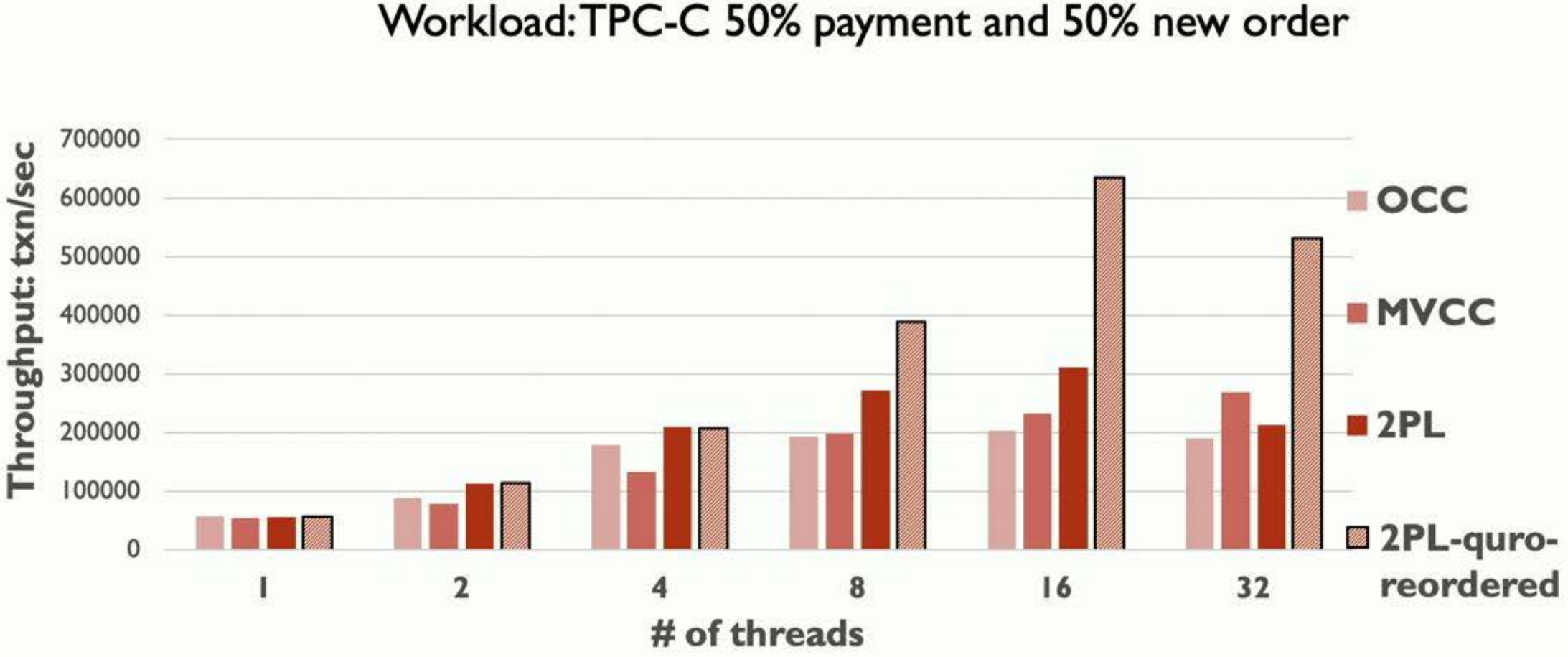
Latency: decrease up to 70%

Evaluation: Quro Vs. Original



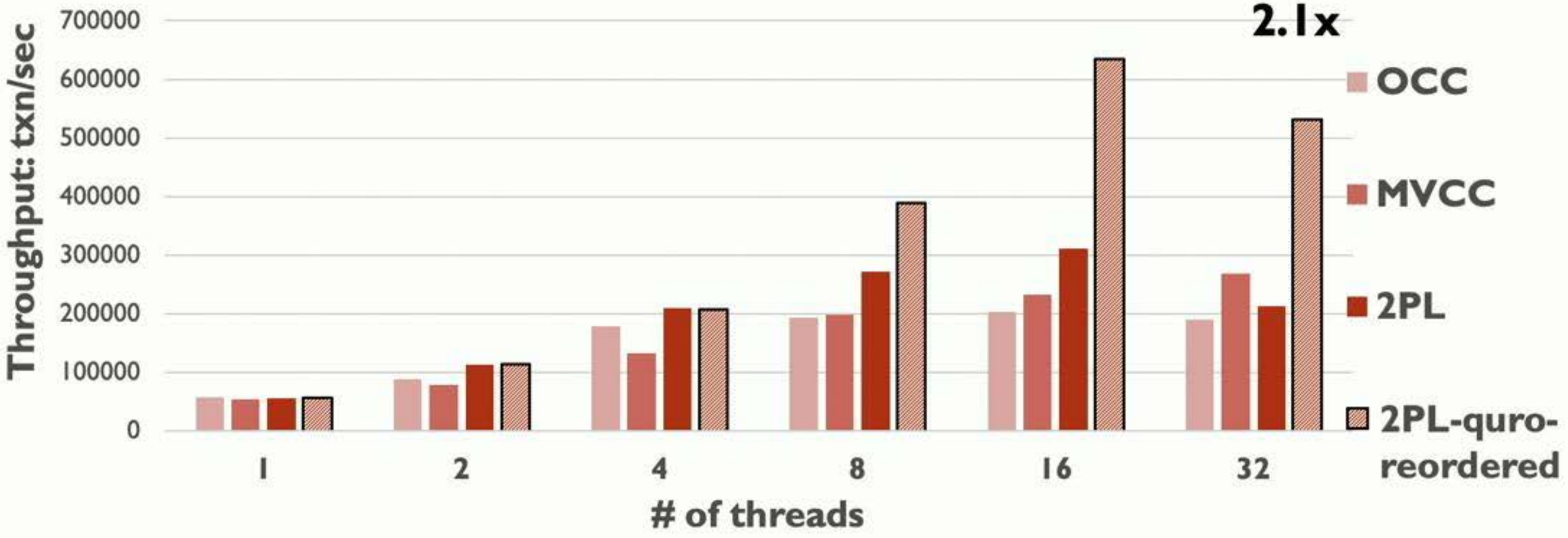
Latency: decrease up to 70%

Evaluation: Quro Vs. Other CC Schemes



Evaluation: Quro Vs. Other CC Schemes

Workload: TPC-C 50% payment and 50% new order



Quro Summary

VLDB'16

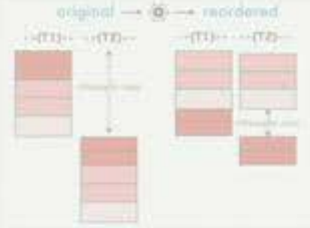
- The order of queries has large impact on transaction performance.
- We build Quro, a compiler that leverages information about query contention, and automatically reorders the queries.
- Quro-generated code improves throughput up to 3.5x, and outperforms other concurrency control schemes under high contention.
- **Reordering is implemented in critical transactions of Taobao in Alibaba**

Outline

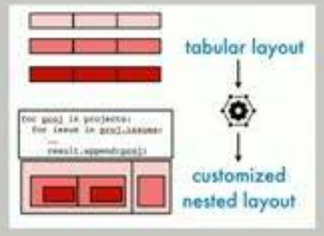
- Leveraging application semantics to optimize each layer

Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout

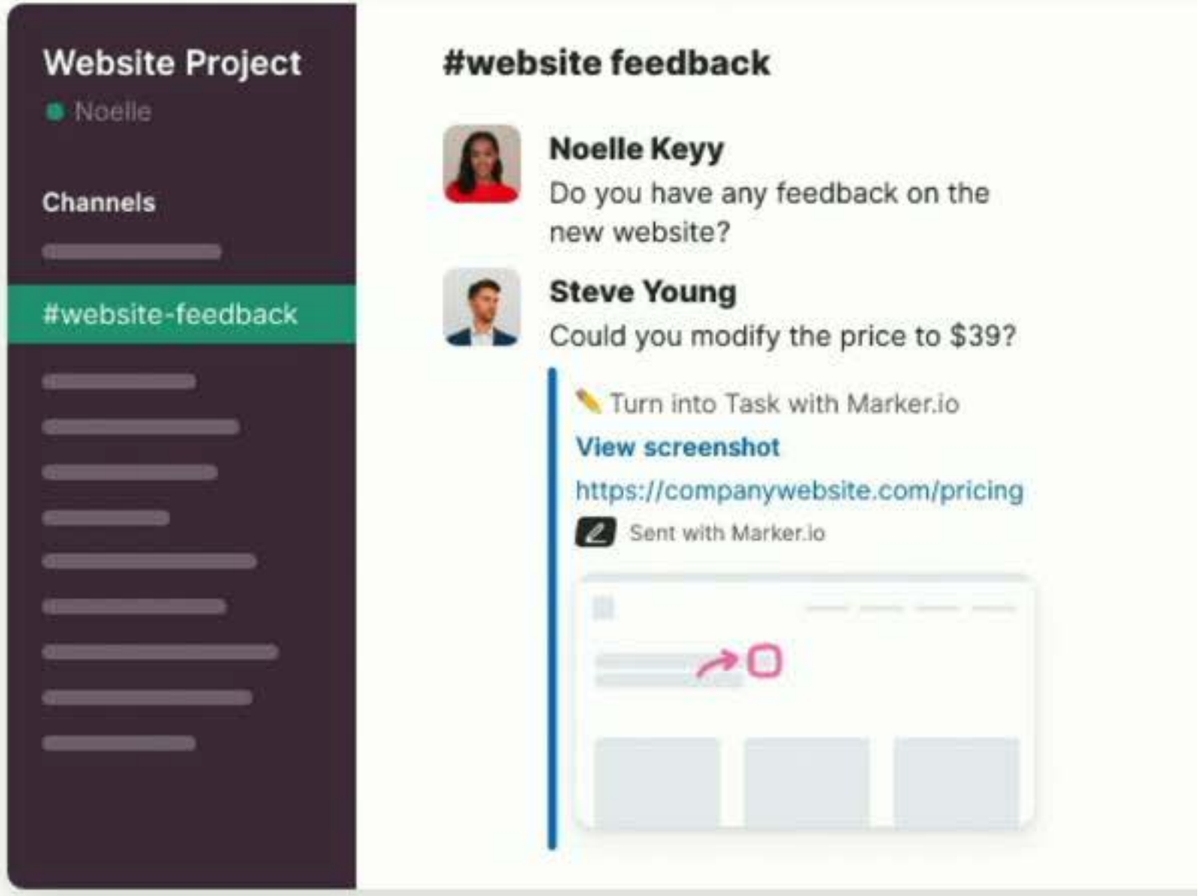


- Other projects
- Ongoing and future work

Data Representation Mismatch

Data Representation Mismatch

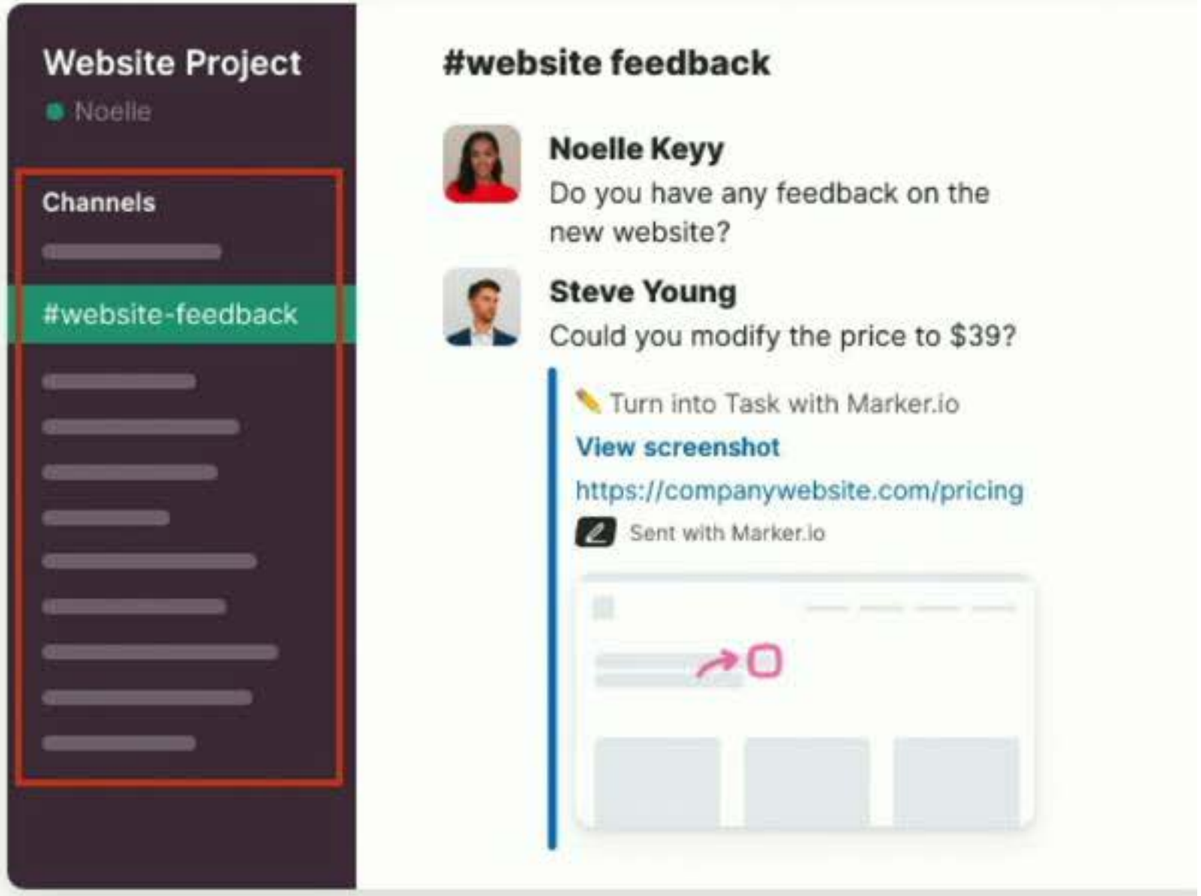
- Nested object vs. tabular data



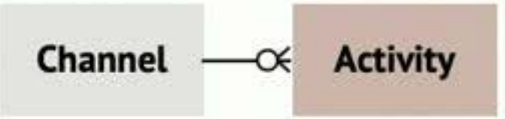
Data Representation Mismatch

Channel

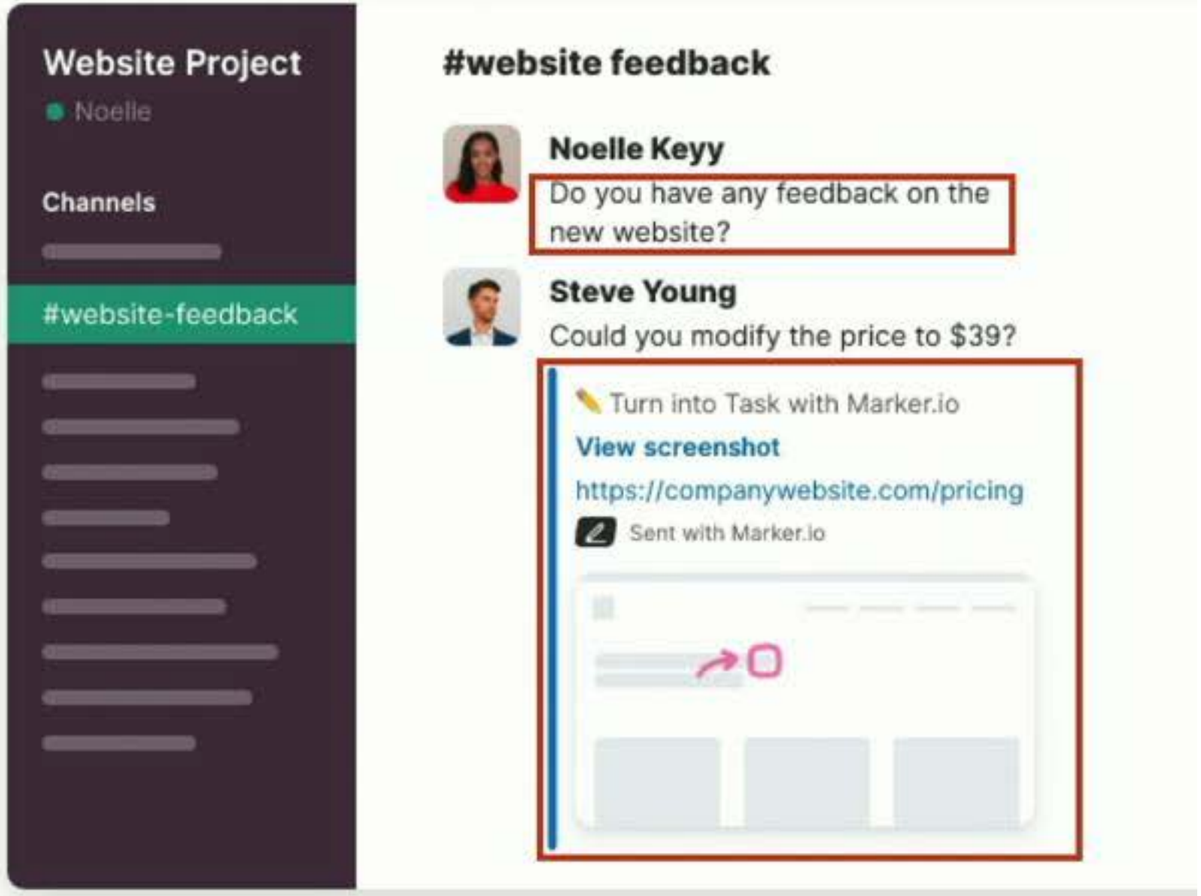
- Nested object vs. tabular data



Data Representation Mismatch

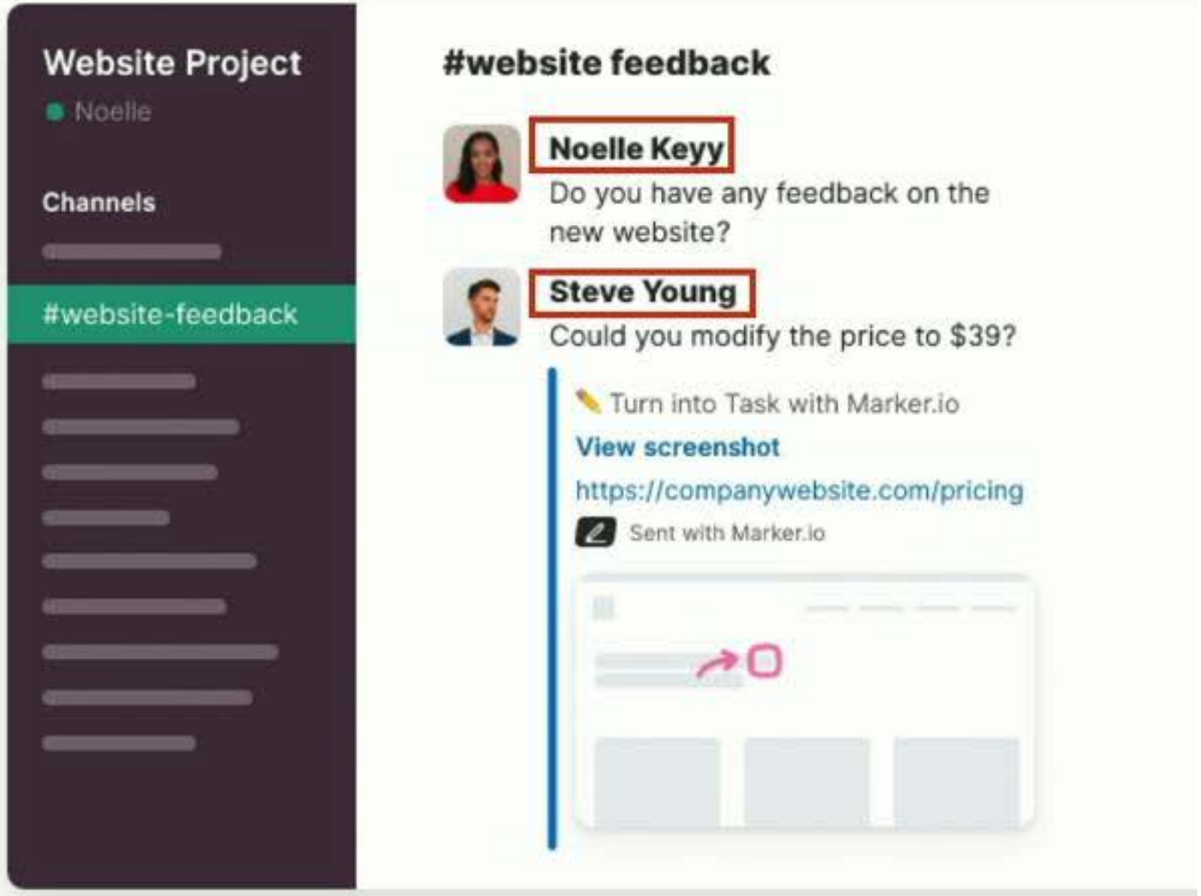


- Nested object vs. tabular data



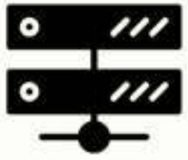
Data Representation Mismatch

- Nested object vs. tabular data

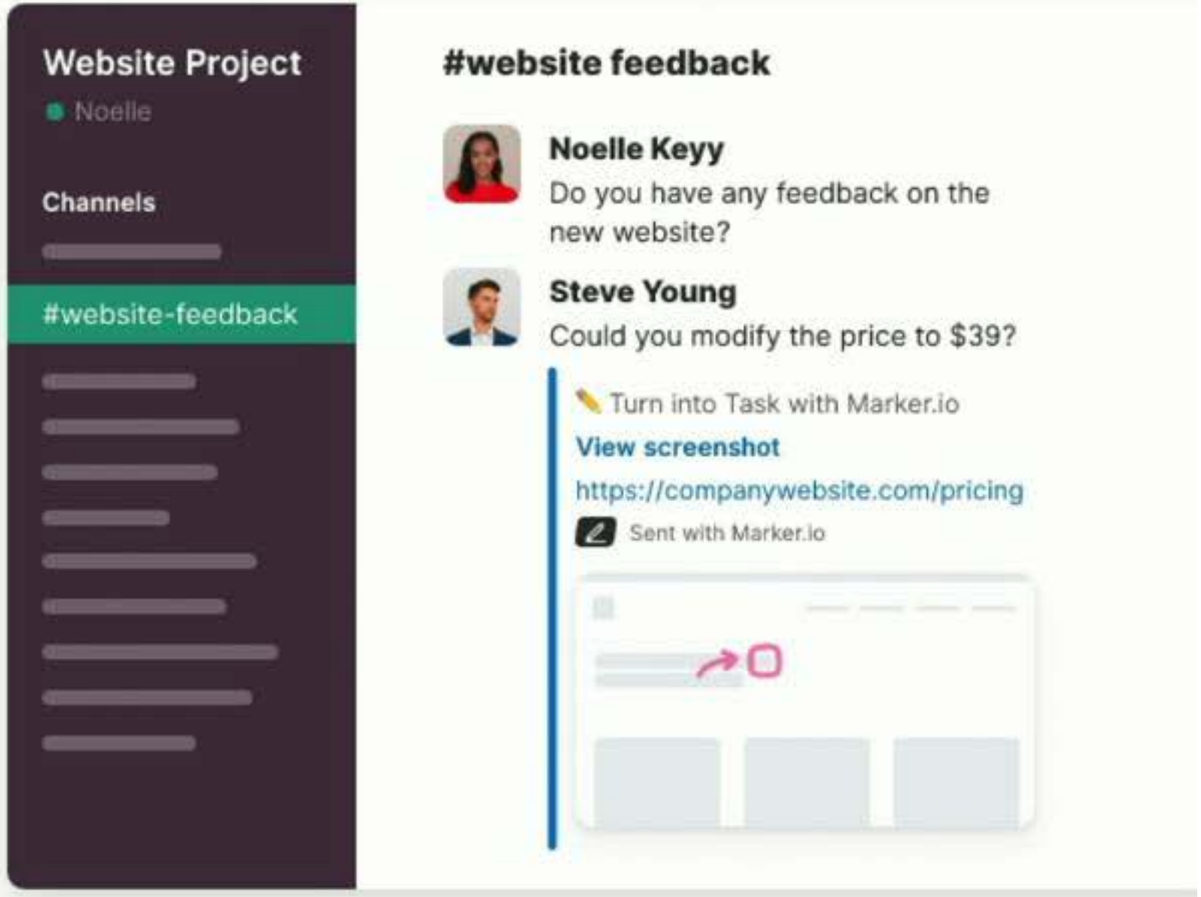


Data Representation Mismatch

- Nested object vs. tabular data

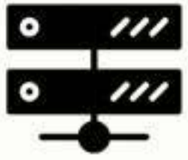


website	dev



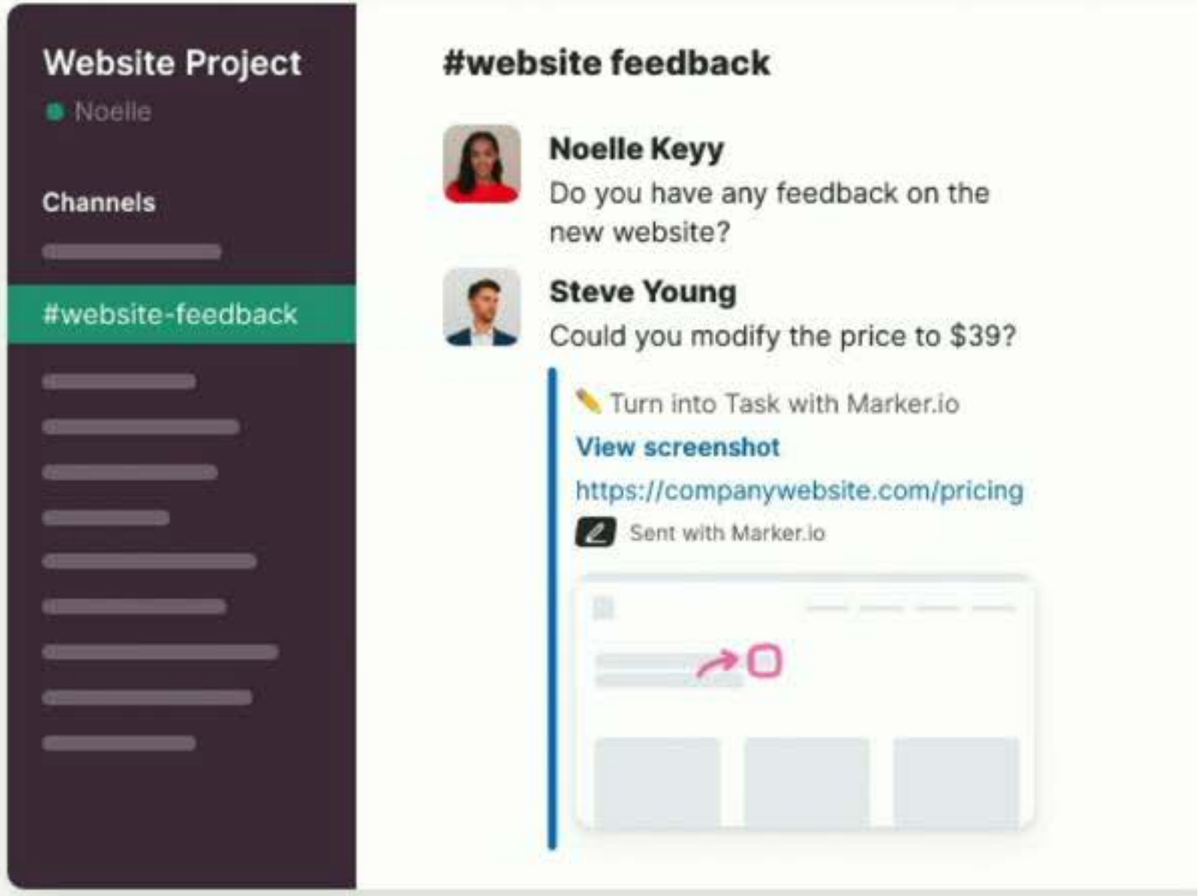
Data Representation Mismatch

- Nested object vs. tabular data



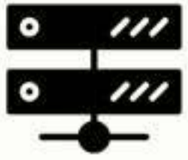
website
msg,"Hey,"
file, a.pdf

dev
file, b.pdf



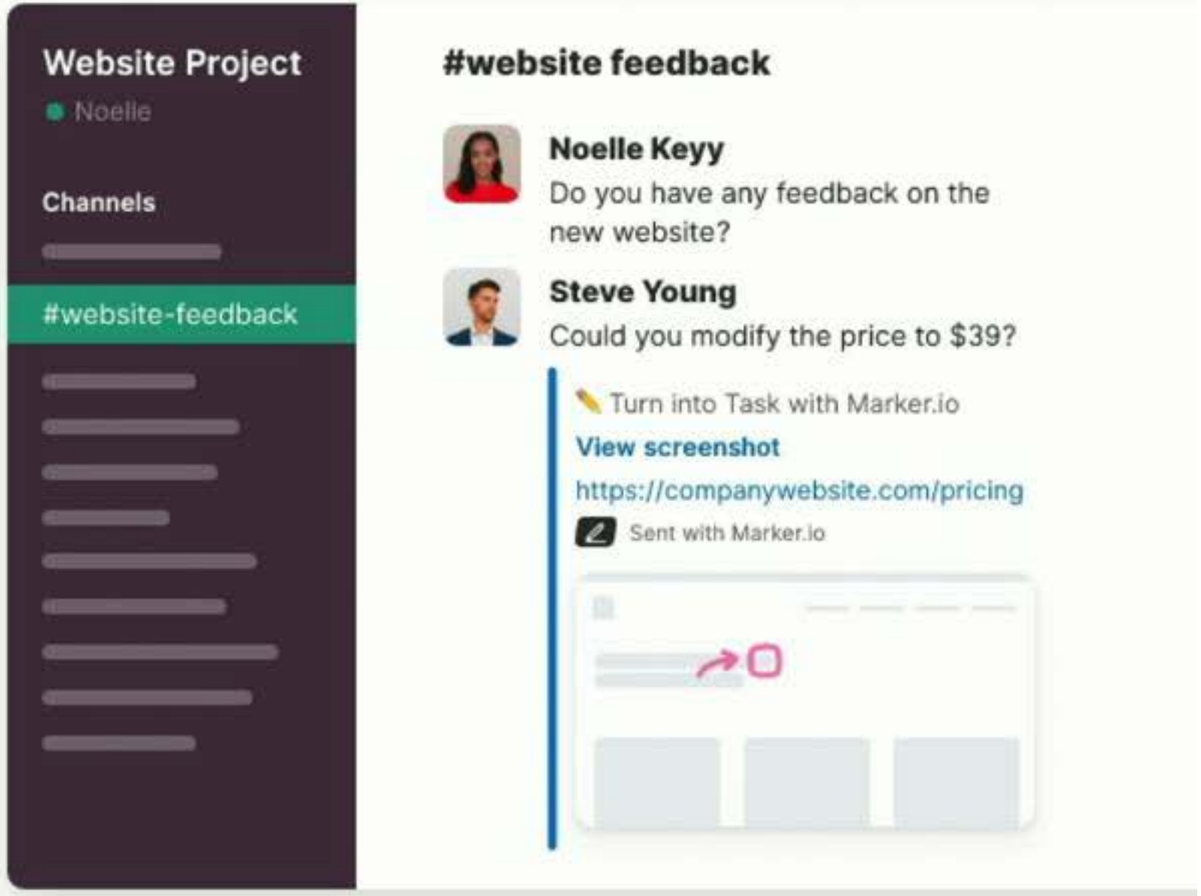
Data Representation Mismatch

- Nested object vs. tabular data



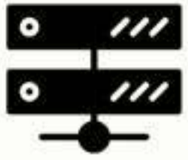
website
msg,"Hey," Keyy
file, a.pdf Young

dev
file, b.pdf Young



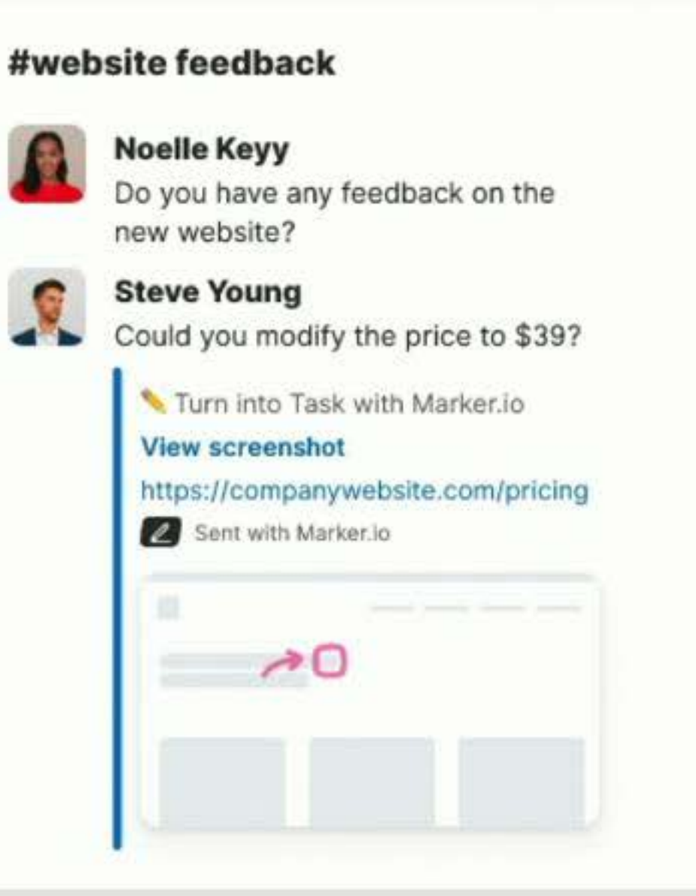
Data Representation Mismatch

- Nested object vs. tabular data



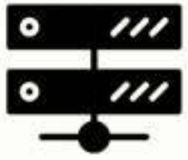
website
msg,"Hey," Keyy
file, a.pdf Young
dev
file, b.pdf Young

website	dev	brunch
msg, "Hey,"	file, a.pdf	file, b.pdf
Keyy	Young	Eve



Data Representation Mismatch

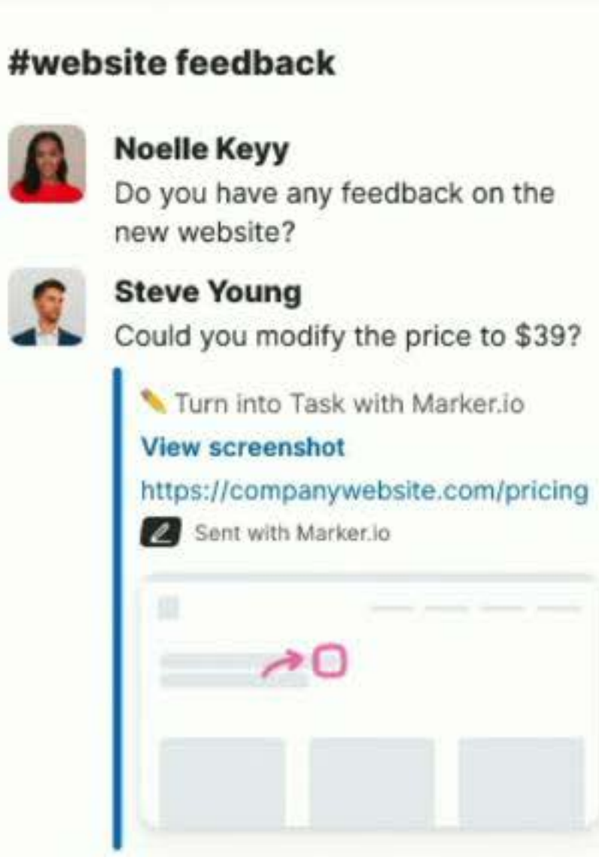
- Nested object vs. tabular data



website
msg,"Hey," Keyy
file, a.pdf Young
dev
file, b.pdf Young

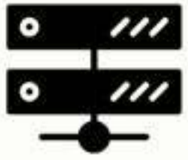
50	website	dev	brunch
10K	msg, "Hey,"	file, a.pdf	file, b.pdf
5K	Keyy	Young	Eve

1.7 sec



Data Representation Mismatch

- Nested object vs. tabular data

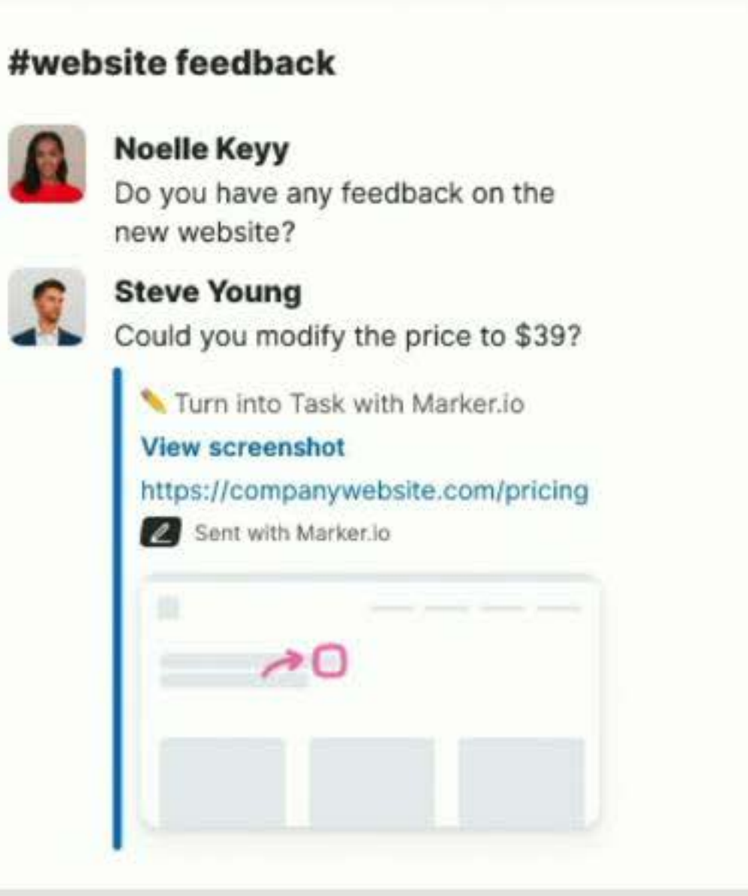


website
msg,"Hey," Keyy
file, a.pdf Young
dev
file, b.pdf Young

55 sec

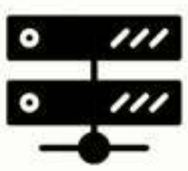
50	website	dev	brunch
10K	msg, "Hey,"	file, a.pdf	file, b.pdf
5K	Keyy	Young	Eve

1.7 sec



Different App Queries Use Different Layout

- Nested object vs. tabular data

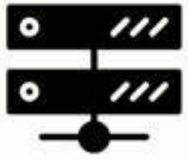


website
msg,"Hey," Keyy
file, a.pdf Young
dev
file, b.pdf Young

website	dev	brunch
msg, "Hey,"	file, a.pdf	file, b.pdf
Keyy	Young	Eve

Different App Queries Use Different Layout

- Nested object vs. tabular data



website
msg,"Hey," Keyy
file, a.pdf Young
dev
file, b.pdf Young

website	dev	brunch
msg, "Hey,"	file, a.pdf	file, b.pdf
Keyy	Young	Eve

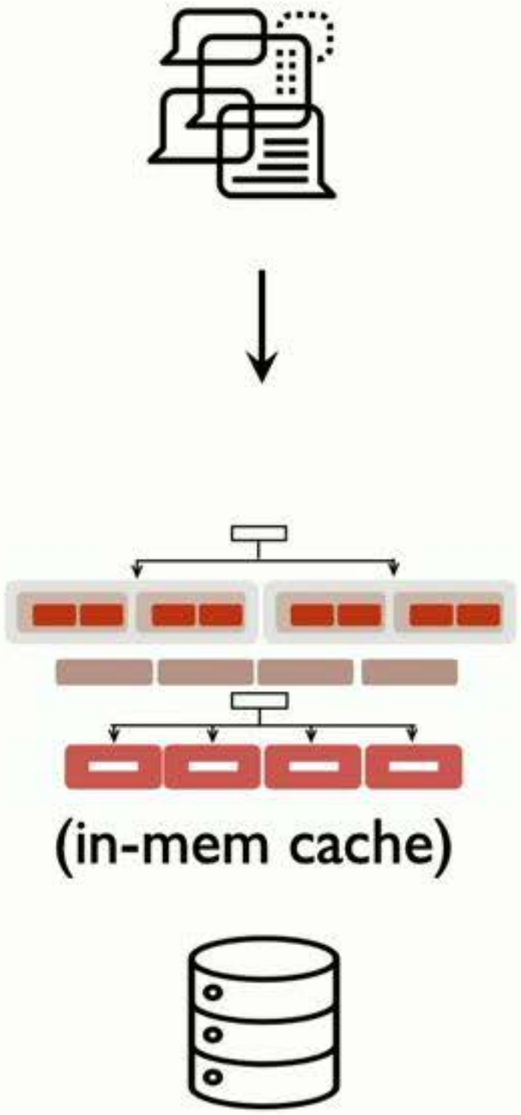
Keyy
msg, "key"
website

Young
file, a.pdf
file, b.pdf
website
dev

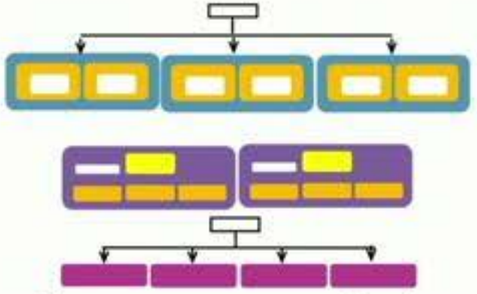
How to Run Queries Faster?

How to Run Queries Faster?

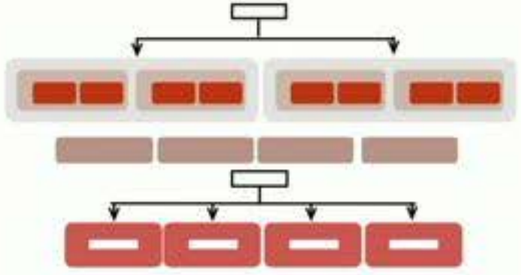
How to Run Queries Faster?



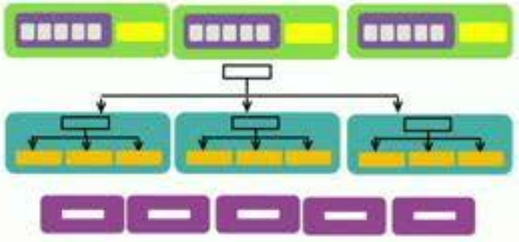
How to Run Queries Faster?



(in-mem cache)

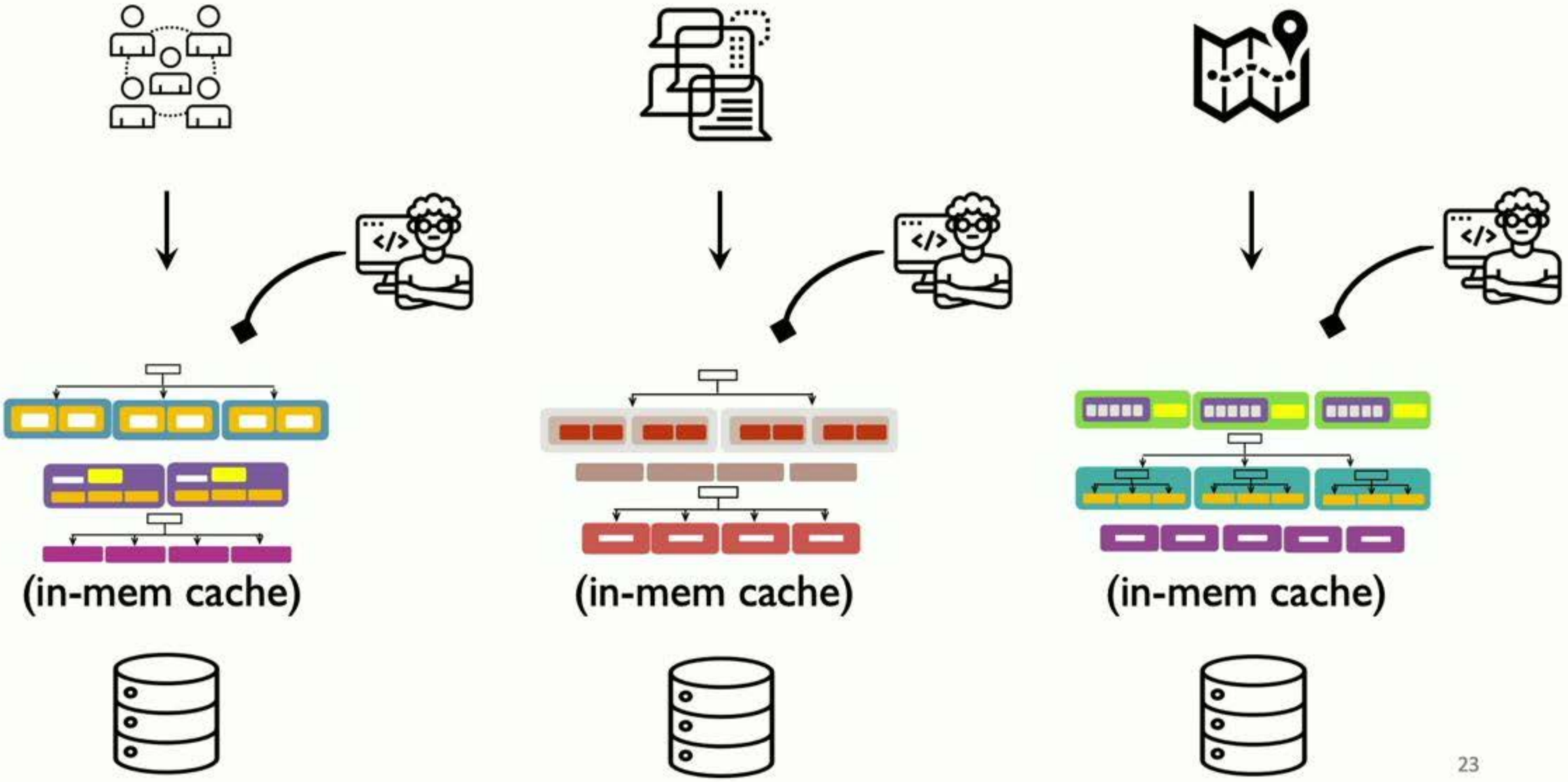


(in-mem cache)

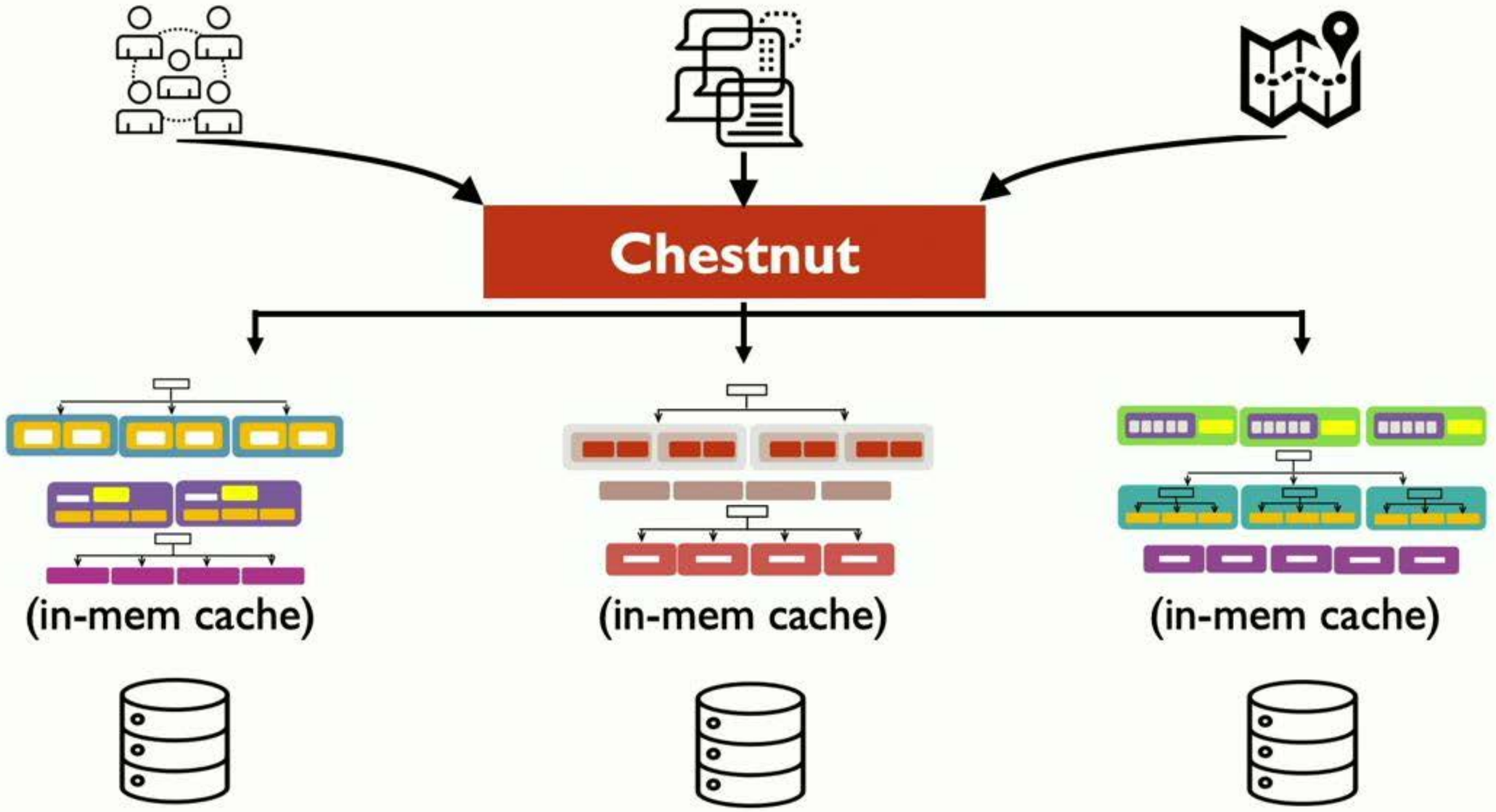


(in-mem cache)

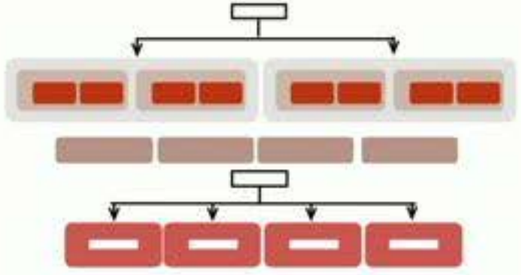
How to Run Queries Faster?



How to Run Queries Faster?

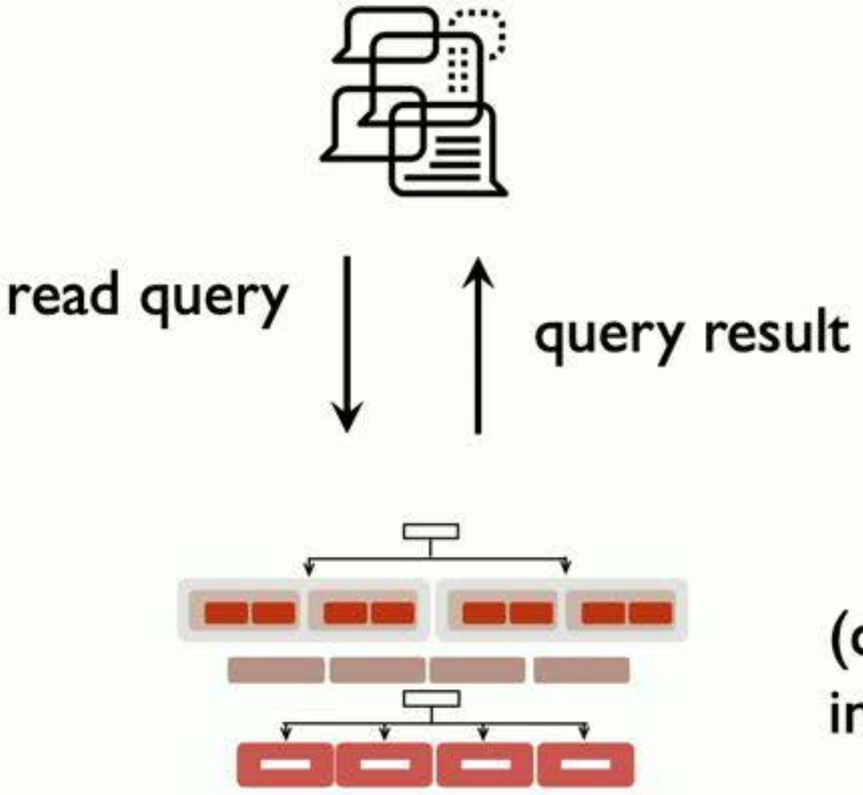


How to Run Queries Faster?



(chestnut-generated in-mem cache)

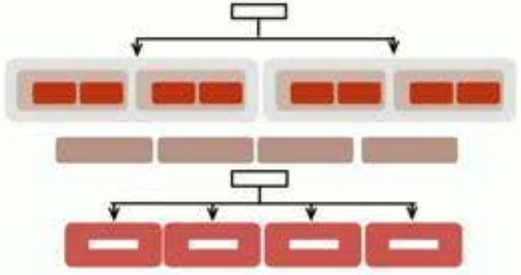
How to Run Queries Faster?



(chestnut-generated in-mem cache)

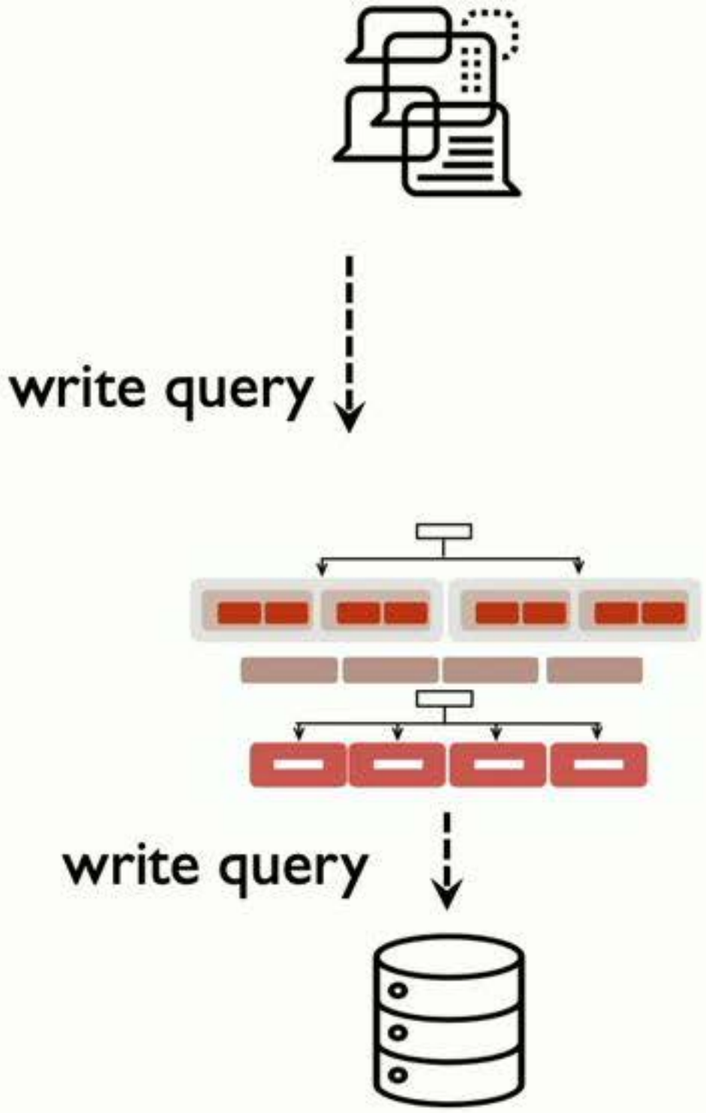
How to Run Queries Faster?

write query



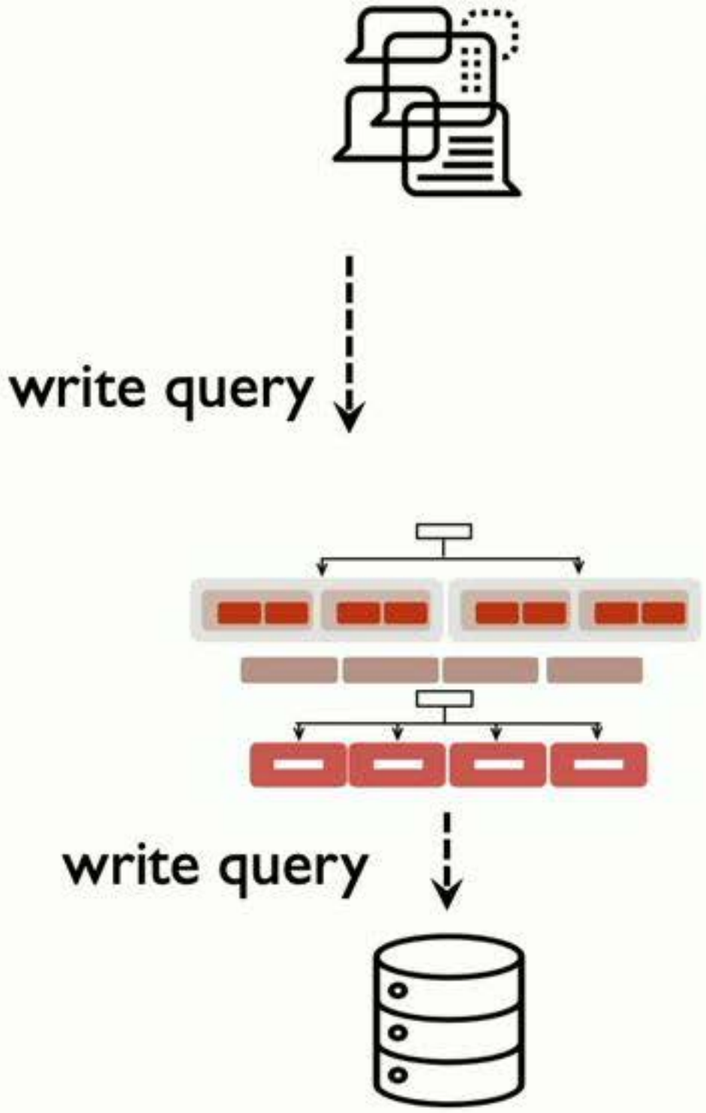
(chestnut-generated in-mem cache)

How to Run Queries Faster?



(chestnut-generated in-mem cache)

How to Run Queries Faster?

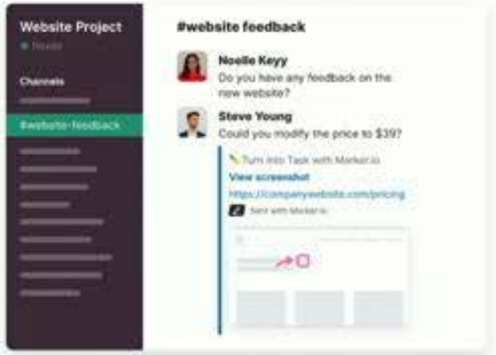


Key challenges:

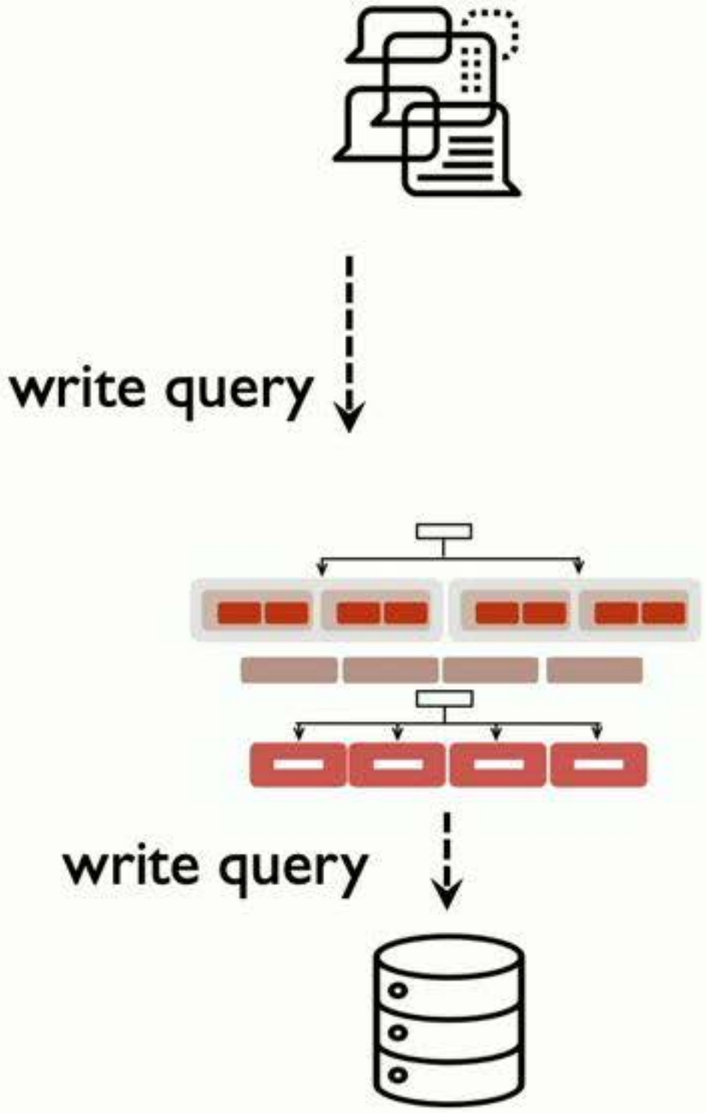
1. How to decide the data layout?
2. How to answer the queries?

(chestnut-generated in-mem cache)

Chestnut Step 1: Layout Enumeration



How to Run Queries Faster?

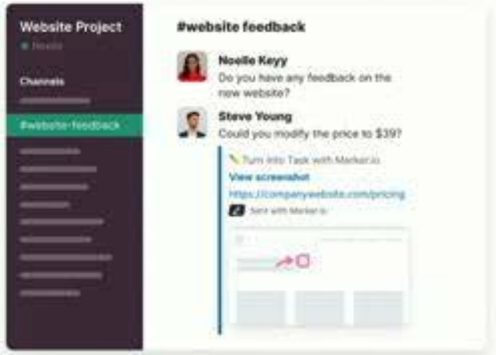


Key challenges:

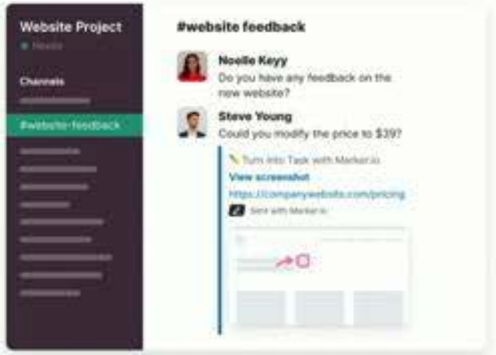
1. How to decide the data layout?
2. How to answer the queries?

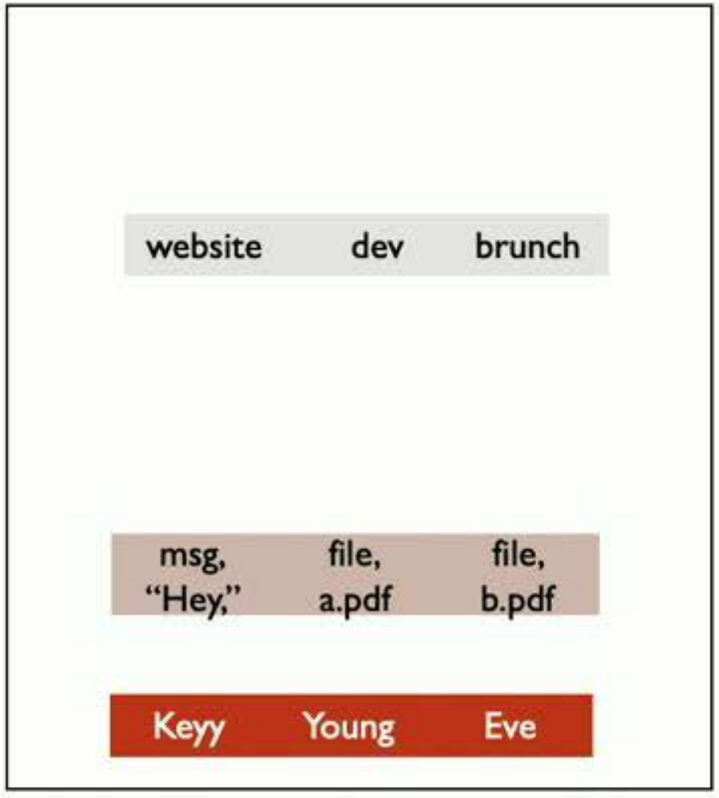
(chestnut-generated in-mem cache)

Chestnut Step 1: Layout Enumeration



Chestnut Step 1: Layout Enumeration

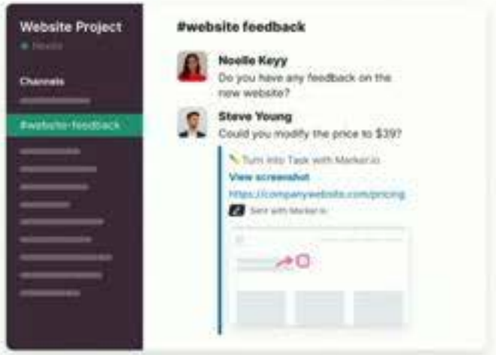


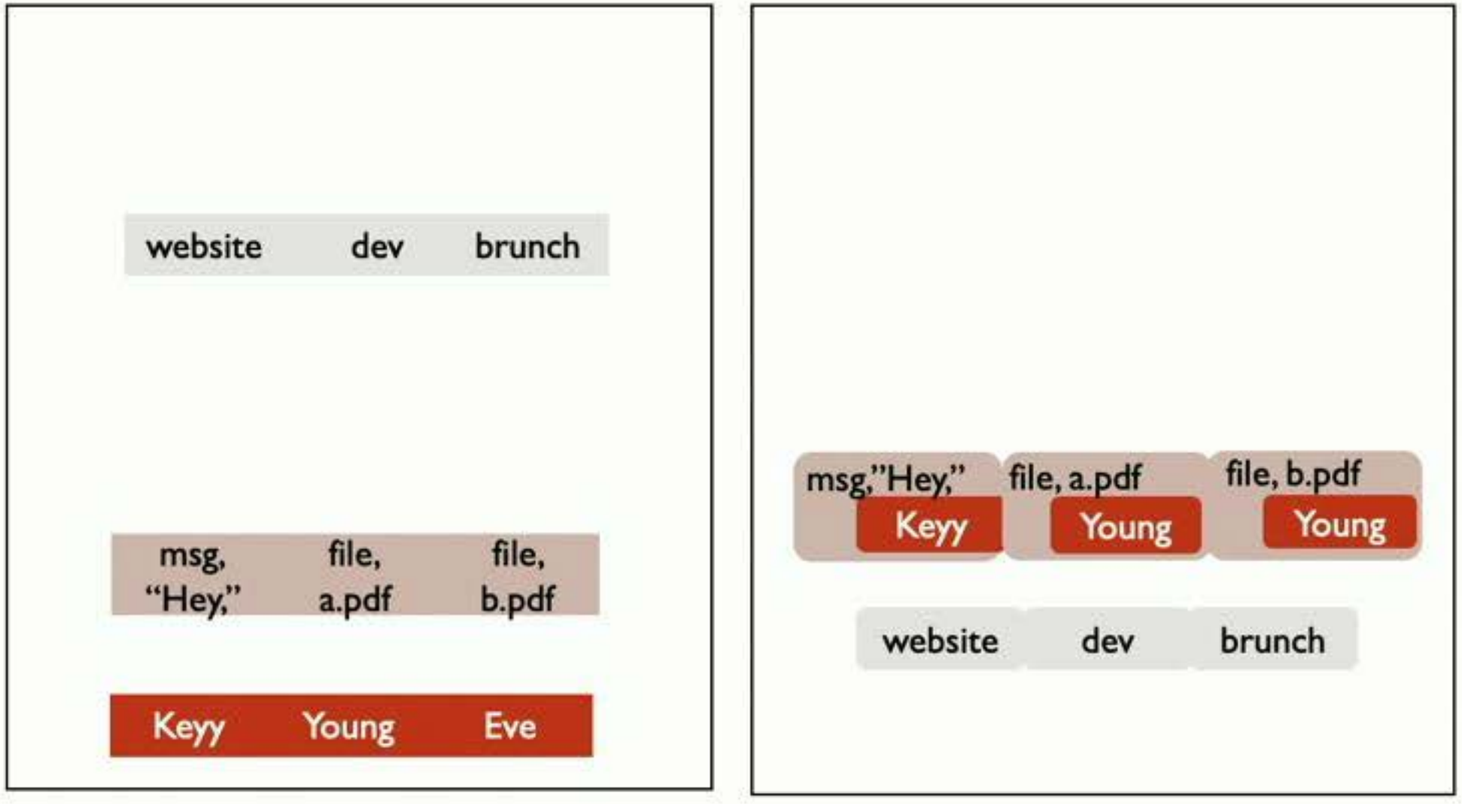


layout 1

...

Chestnut Step 1: Layout Enumeration



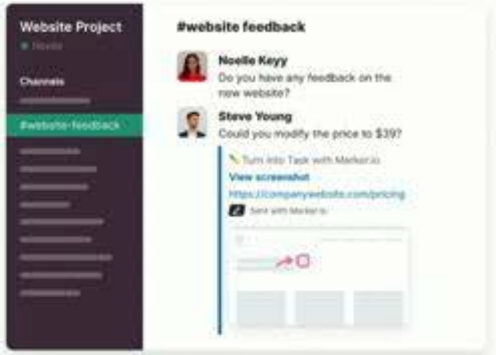


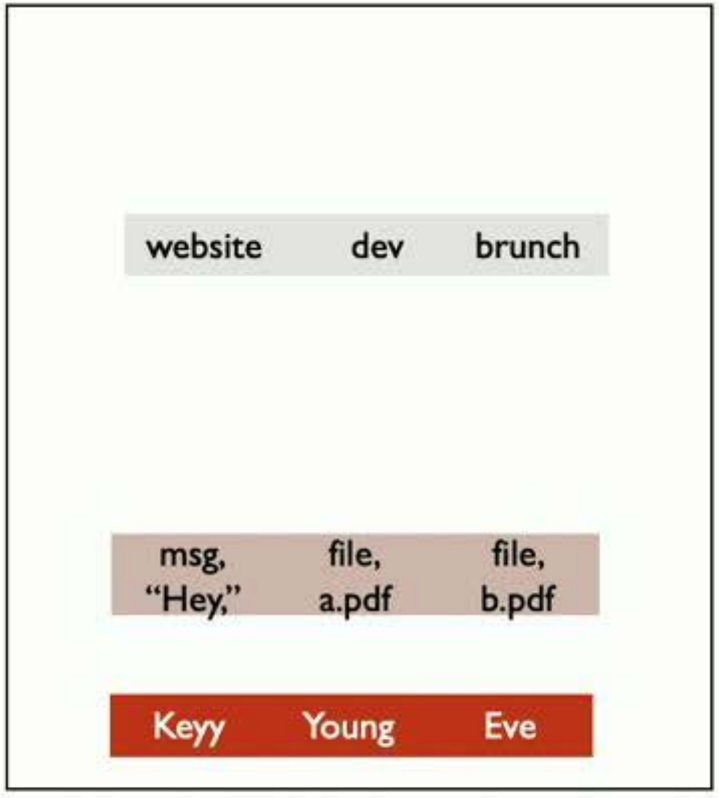
layout 1

layout 2

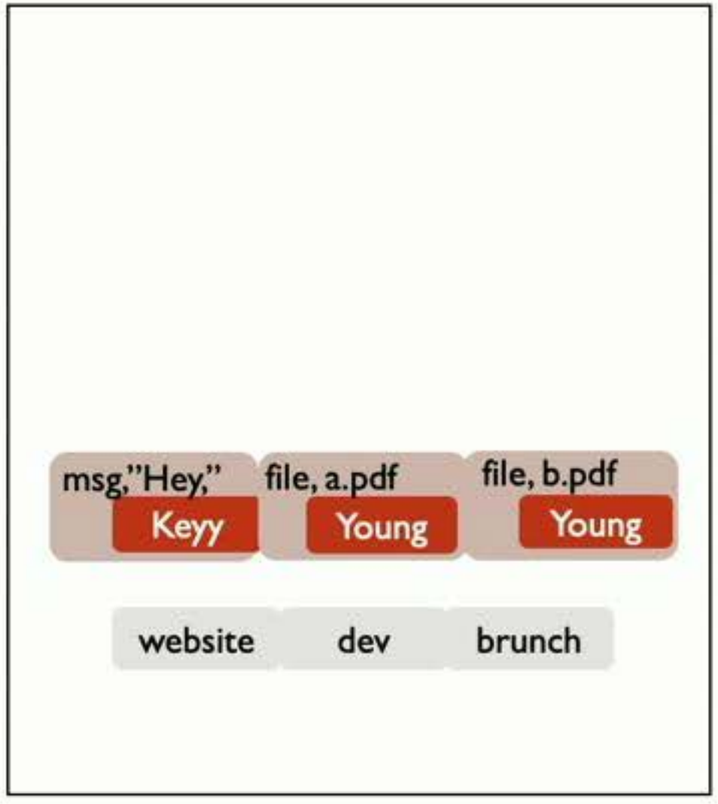
...

Chestnut Step 1: Layout Enumeration

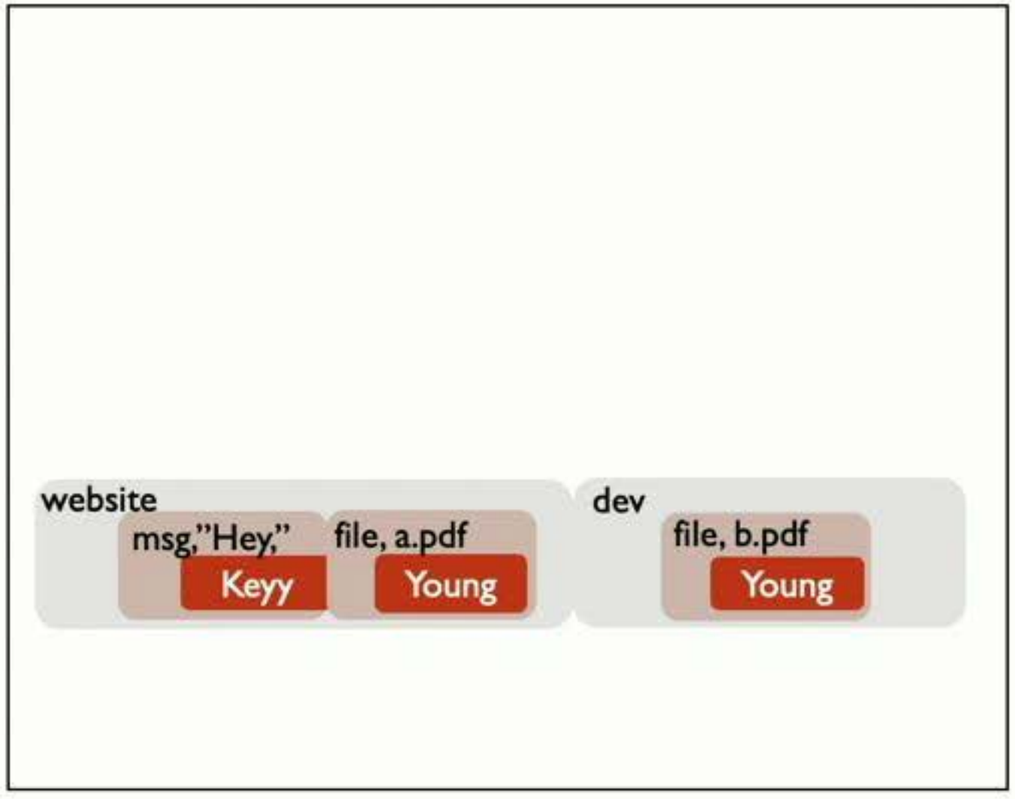




layout 1



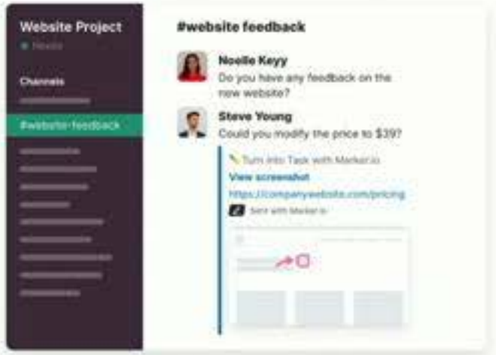
layout 2

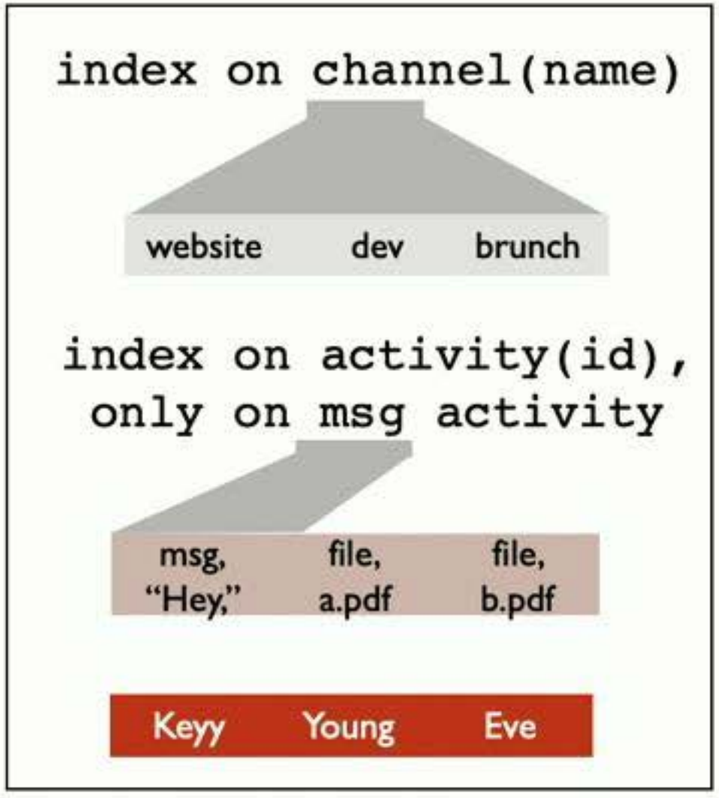


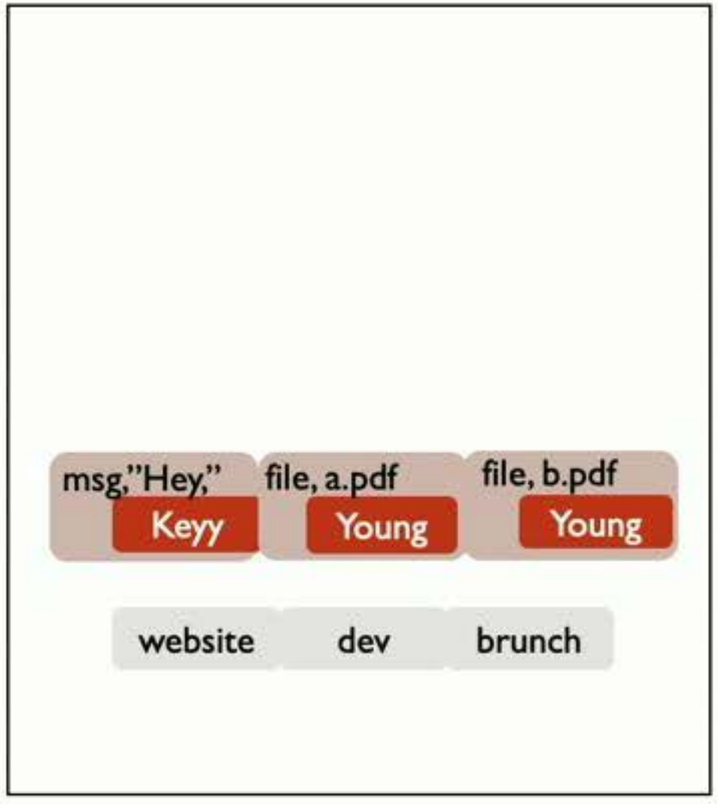
layout 3

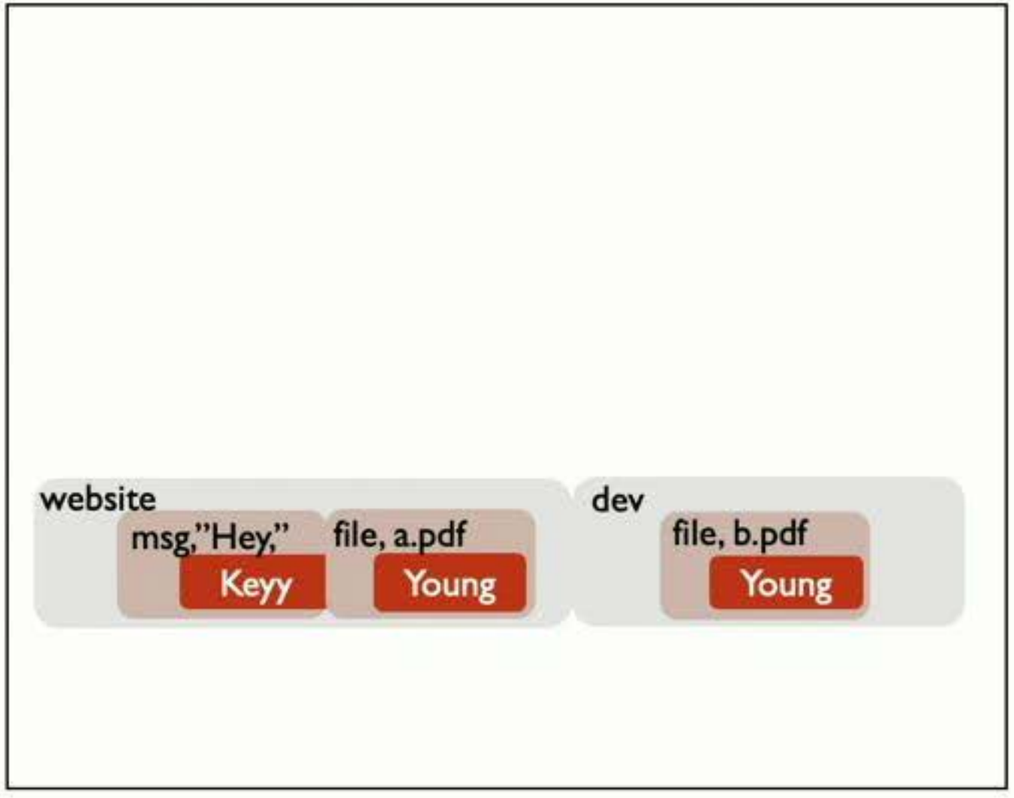
...

Chestnut Step 1: Layout Enumeration









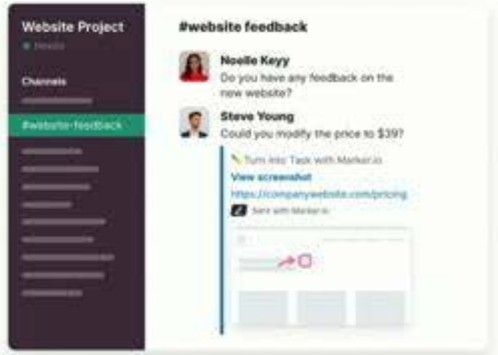
layout 1

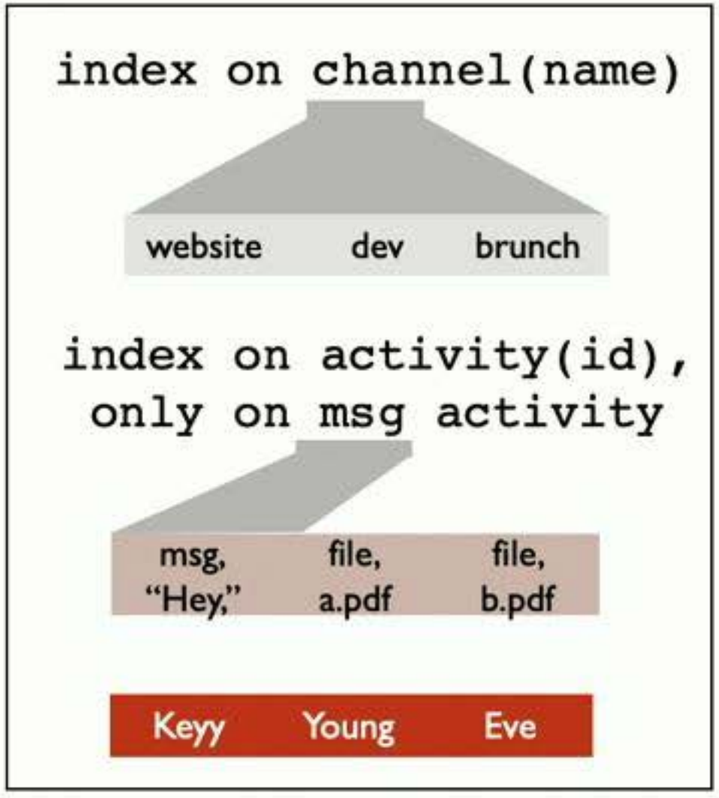
layout 2

layout 3

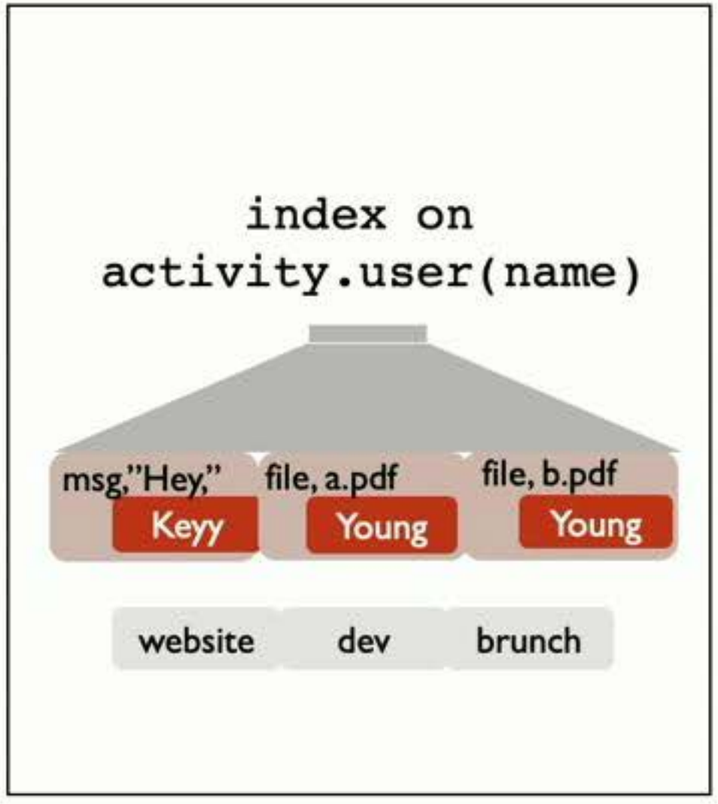
...

Chestnut Step 1: Layout Enumeration

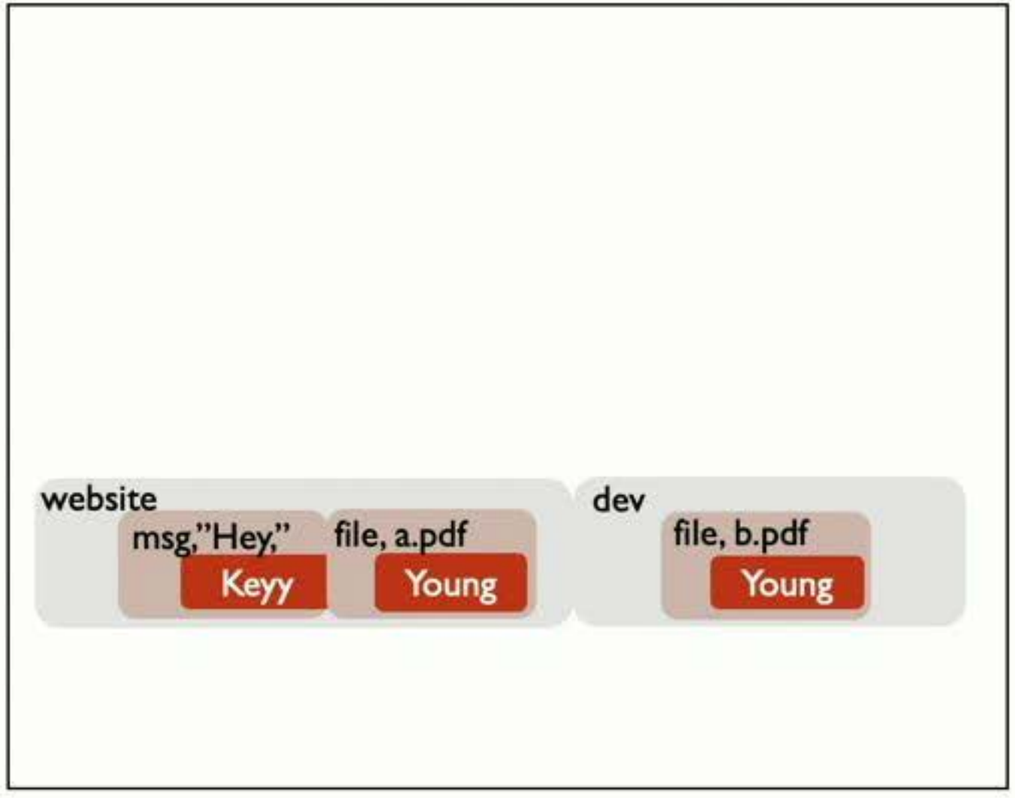




layout 1



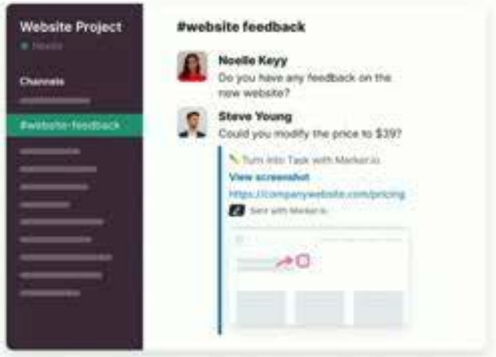
layout 2

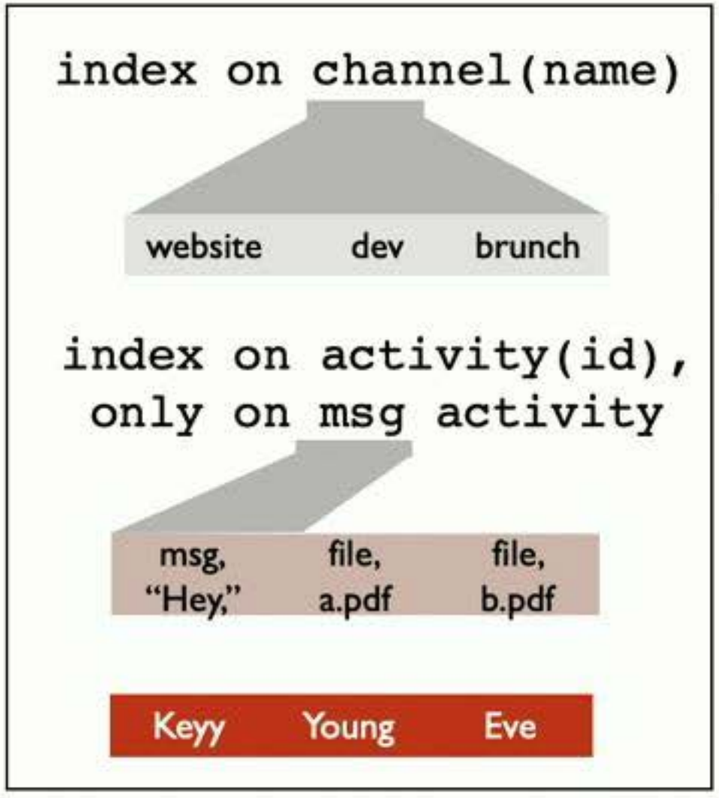


layout 3

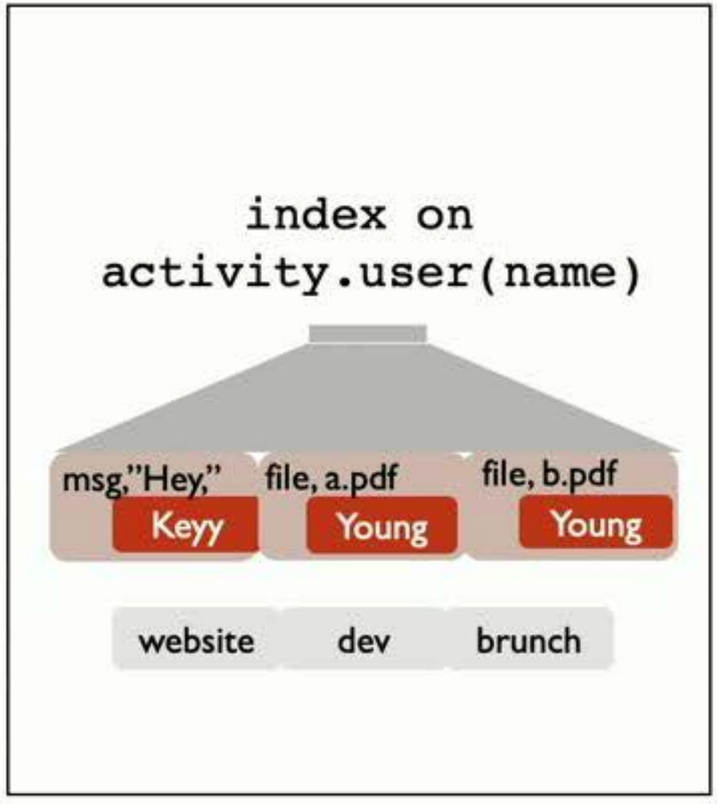
...

Chestnut Step 1: Layout Enumeration

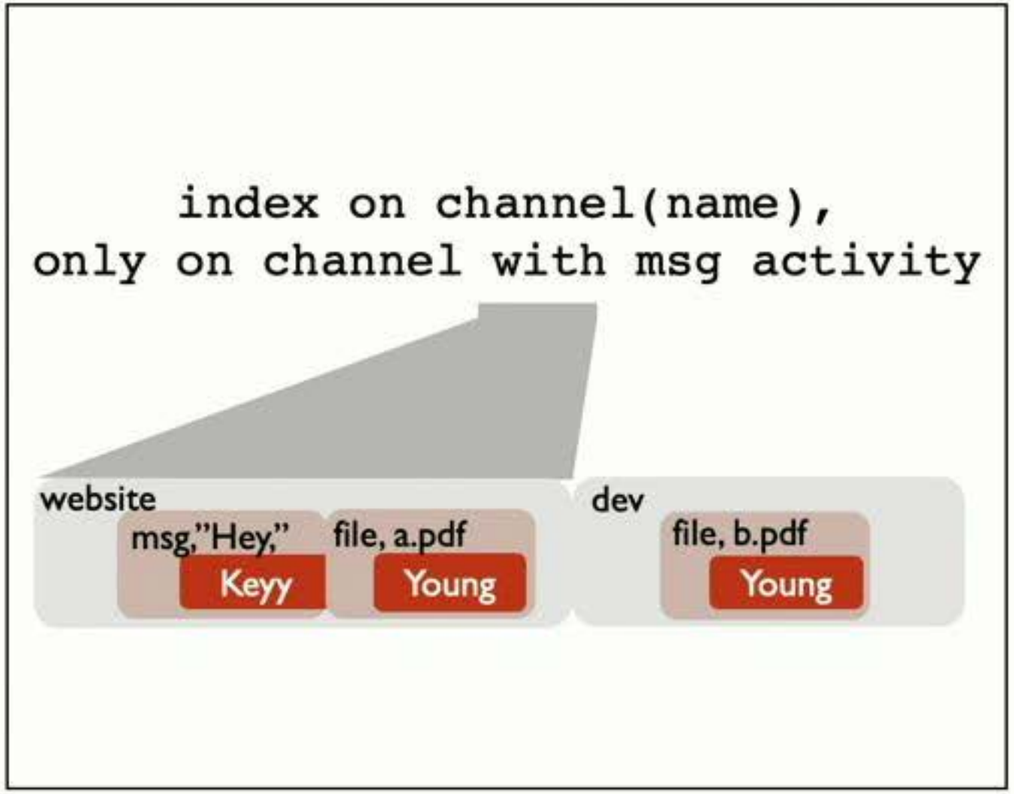




layout 1



layout 2



layout 3

...

Chestnut Step 2: Plan Enumeration

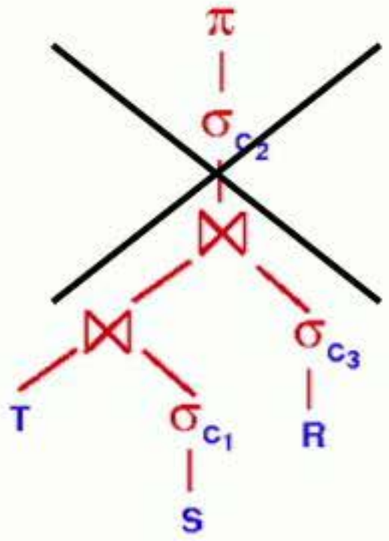
Data layout:

Expected query result:

Chestnut Step 2: Plan Enumeration

Data layout:

Expected query result:



relational query plans
cannot be applied

Chestnut Step 2: Plan Enumeration

Data layout:

Expected query result:


```
for c in array_channel:
    result.append(c)
```

Chestnut Step 2: Plan Enumeration

Data layout:

Expected query result:


```
for c in array_channel:
    result.append(c)
```

symbolic verification

Chestnut Step 2: Plan Enumeration

Data layout:


```
for c in array_channel:
    result.append(c)
```

Expected query result:


```
for c in array_channel:
  for a in array_activity:
    if a.channel_id==c.id:
      c.activities.add(a)
  result.append(c)
```

symbolic verification

Chestnut Step 2: Plan Enumeration

Data layout:


```
for c in array_channel:
    result.append(c)
```

✗

Expected query result:

≠

symbolic verification

```
for c in array_channel:
  for a in array_activity:
    if a.channel_id==c.id:
      c.activities.add(a)
  result.append(c)
```

✗

Chestnut Step 2: Plan Enumeration

Data layout:

Expected query result:


```
for c in array_channel:
    result.append(c)
```

✗

```
for c in array_channel:
  for a in array_activity:
    if a.channel_id==c.id:
      c.activities.add(a)
  result.append(c)
```

✗

```
for c in array_channel:
  if c.name == 'website':
    for a in array_activity:
      if a.channel_id==c.id:
        c.activities.add(a)
    result.append(c)
```

symbolic verification

Chestnut Step 2: Plan Enumeration

Data layout:

Expected query result:

=

symbolic verification

```
for c in array_channel:
    result.append(c)
```

```
for c in array_channel:
  for a in array_activity:
    if a.channel_id==c.id:
      c.activities.add(a)
  result.append(c)
```

```
for c in array_channel:
  if c.name == 'website':
    for a in array_activity:
      if a.channel_id==c.id:
        c.activities.add(a)
    result.append(c)
```

Chestnut Step 3: Update Query

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

read query to identify the objects to update:

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

read query to identify the objects to update: Channel.where(exists(msg, where(content="Hey")))

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

read query to identify the objects to update: Channel.where(exists(msg, where(content="Hey")))

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

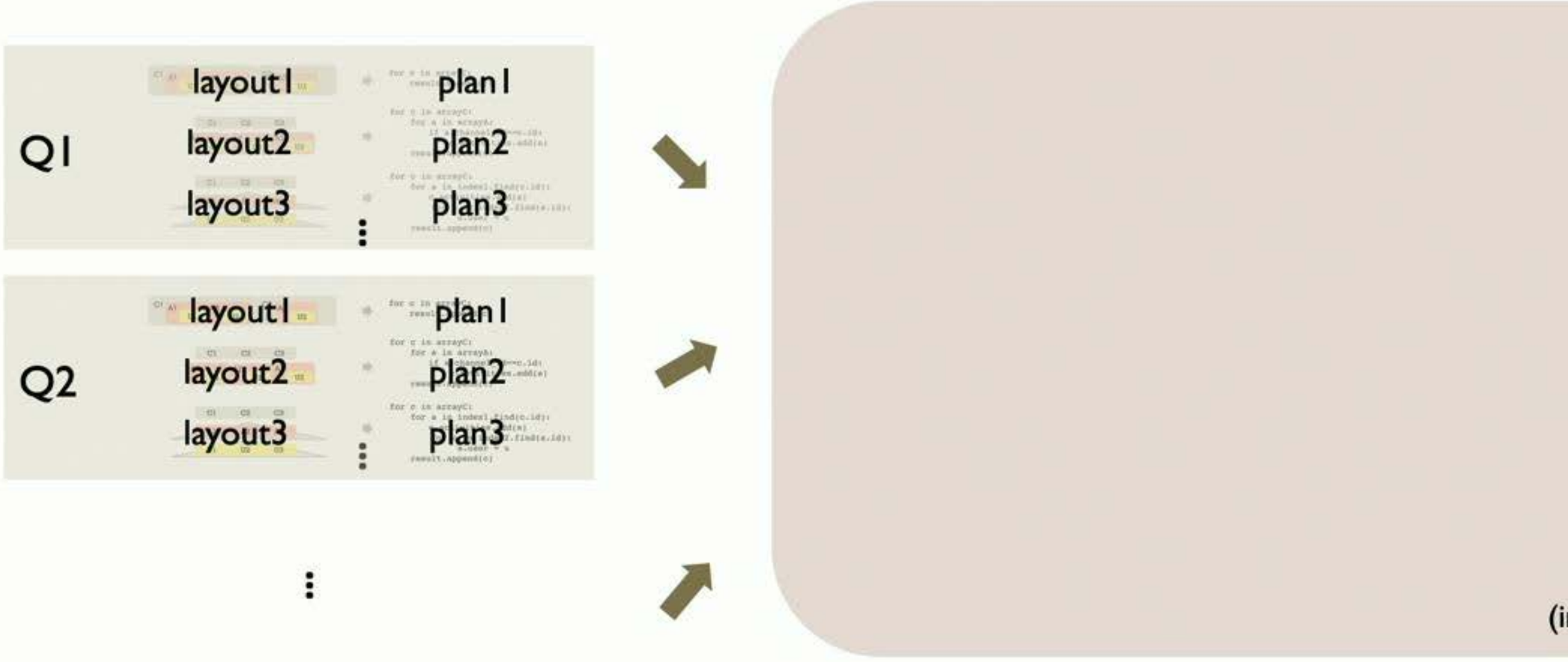
read query to identify the objects to update:

Channel.where(exists(msg, where(content="Hey")))

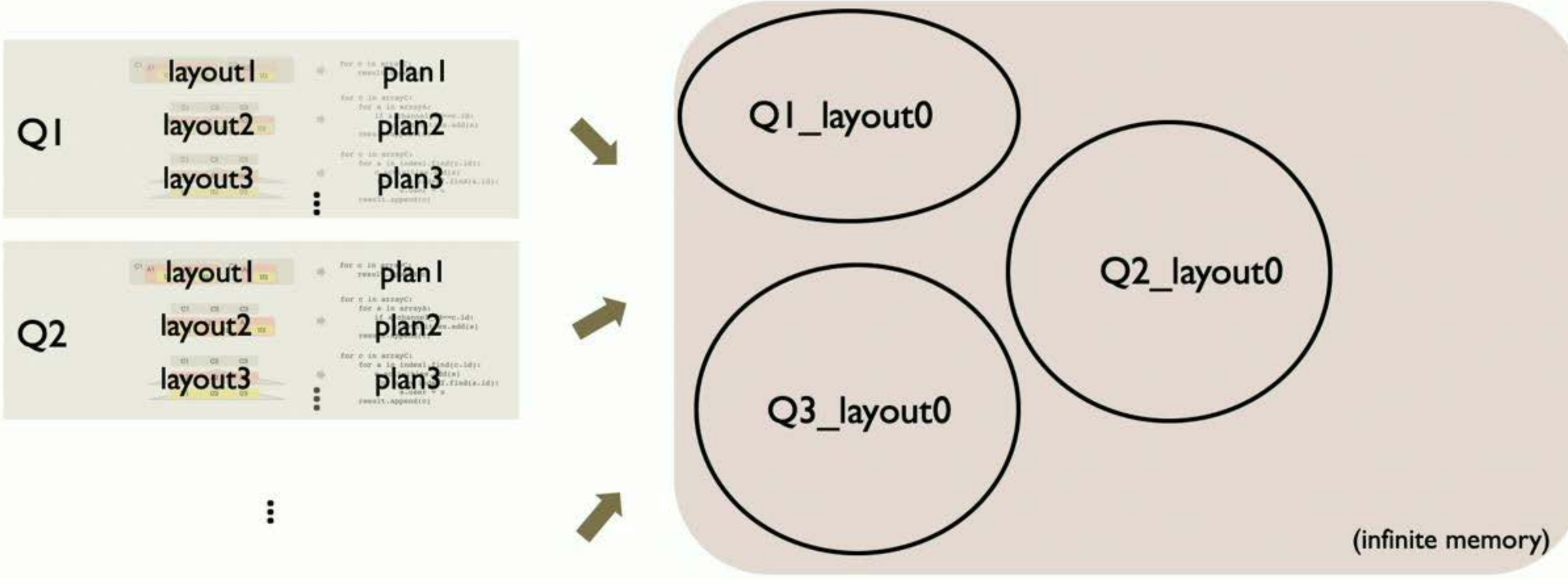
.include(msg, where(content="Hey"))

Chestnut Step 4: Handling Multiple Queries

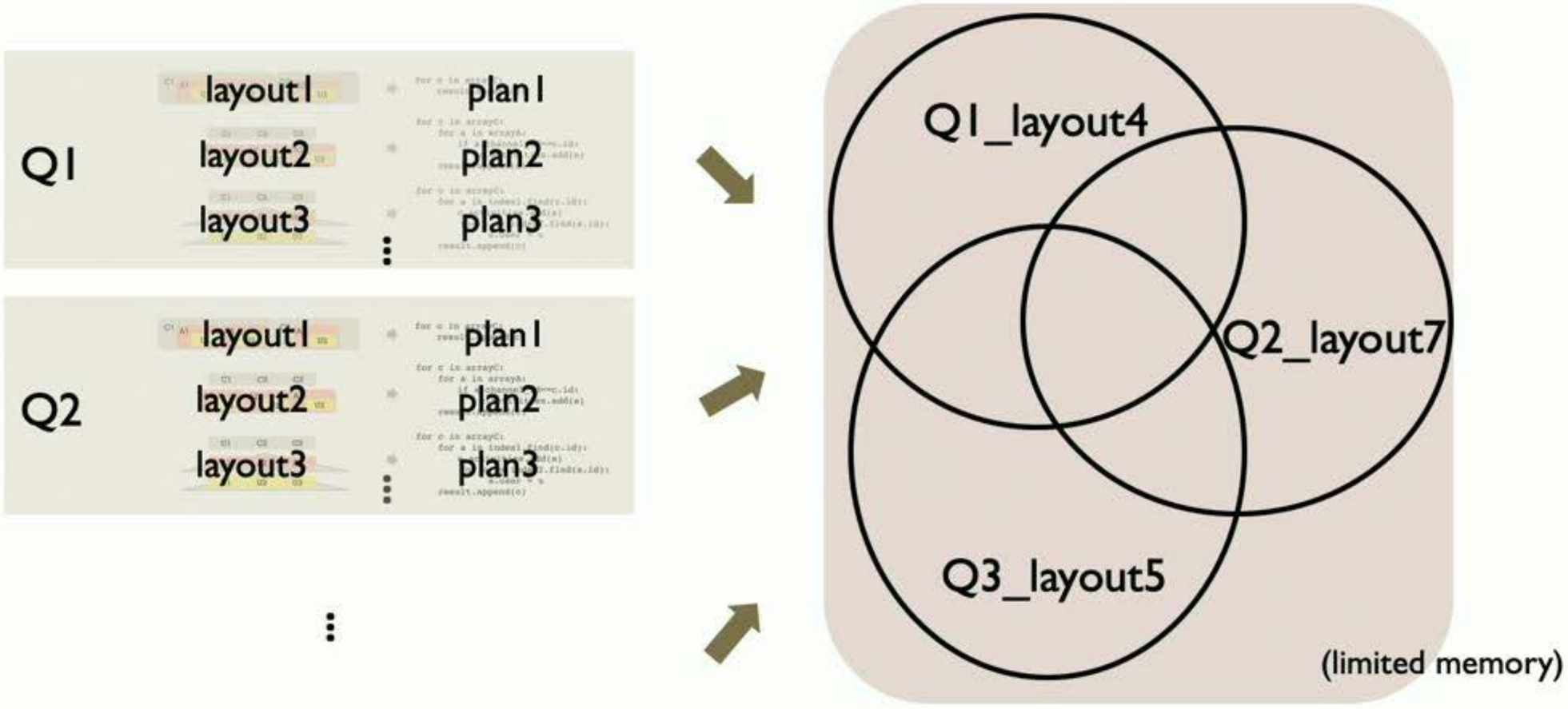
Chestnut Step 4: Handling Multiple Queries



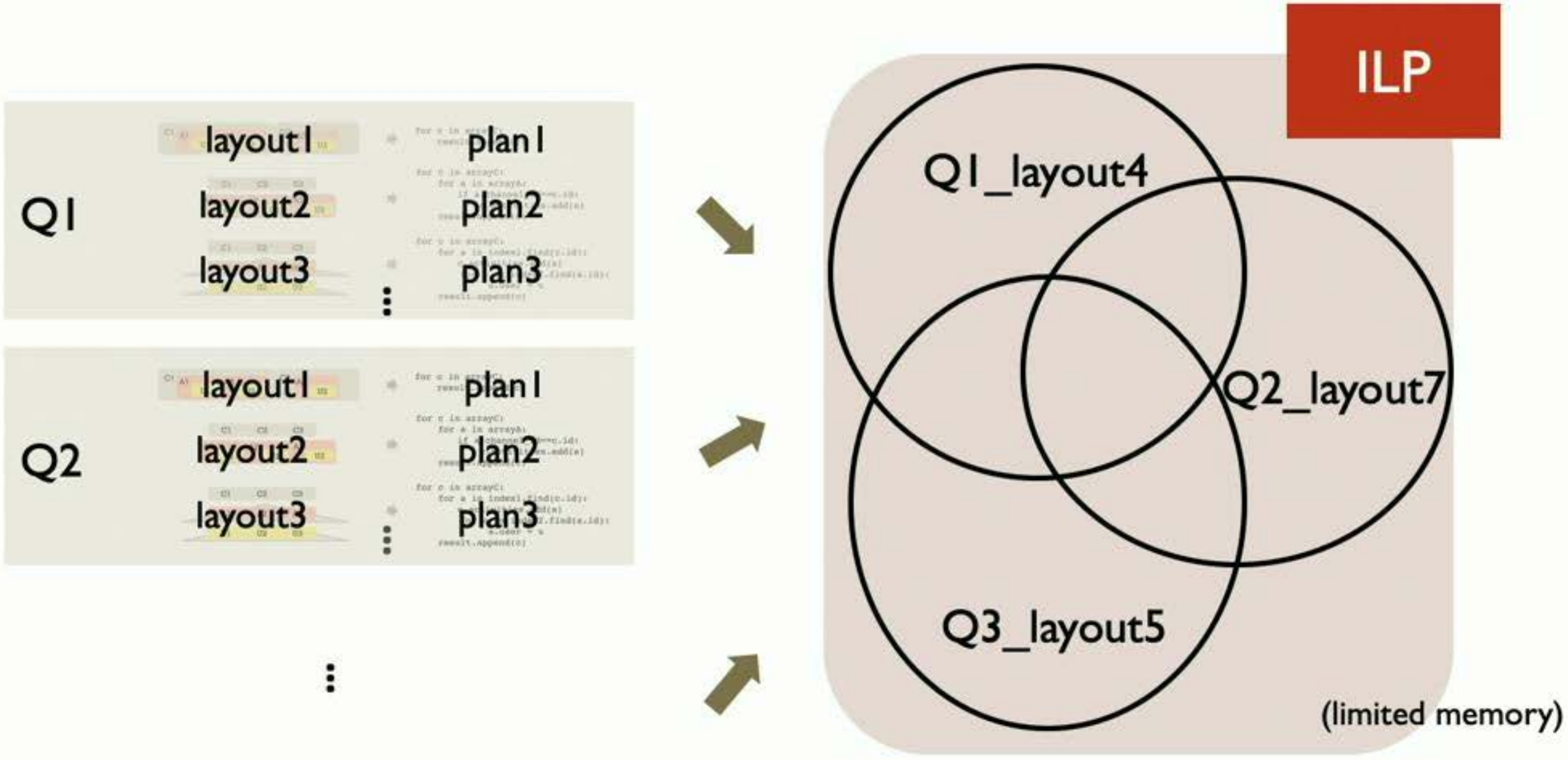
Chestnut Step 4: Handling Multiple Queries



Chestnut Step 4: Handling Multiple Queries



Chestnut Step 4: Handling Multiple Queries



Chestnut Workflow

ILP

Constraint:

- Used data structures within memory bound

Optimization goal:

- Minimize $\sum estimated\ query\ time$

Chestnut Workflow

Output:

- Which data structure to use
- Which query plan to use

Chestnut Workflow

ILP

Constraint:

- Used data structures within memory bound

Optimization goal:

- Minimize $\sum estimated\ query\ time$

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

read query to identify the objects to update:

Channel.where(exists(msg, where(content="Hey")))

.include(msg, where(content="Hey"))

Chestnut Step 3: Update Query

write query: Message.where(content="Hey").**update**(content=>"Hi")

read query to identify the objects to update:

Channel.where(exists(msg, where(content="Hey")))

.include(msg, where(content="Hey"))

Chestnut Workflow

ILP

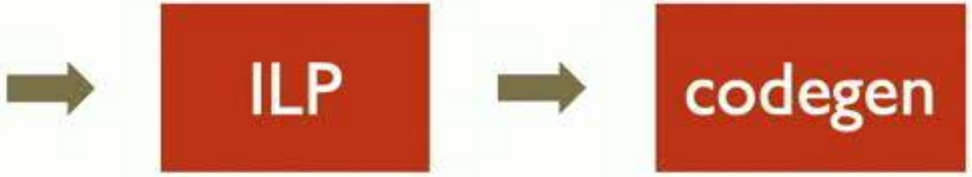
Constraint:

- Used data structures within memory bound

Optimization goal:

- Minimize $\sum estimated\ query\ time$

Chestnut Workflow



Output:

- Which data structure to use
- Which query plan to use

Evaluation

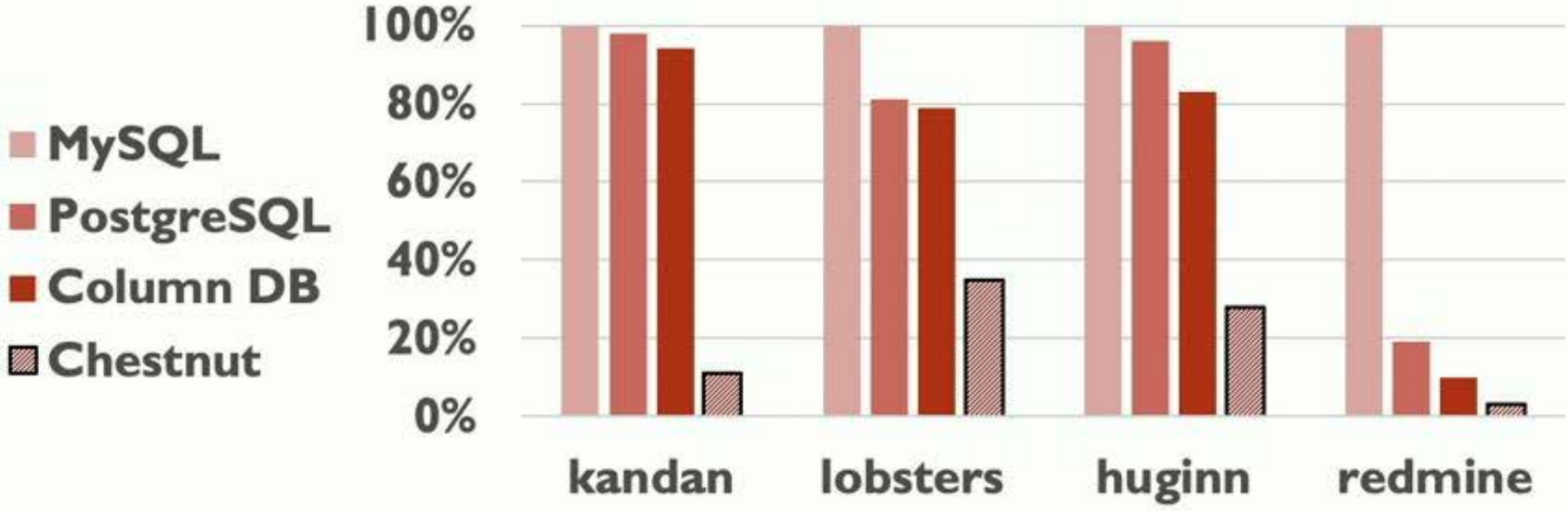
- 4 open-source popular web applications built with Rails
  - kandan: chatting app (2.8k ★)
  - redmine: project management app (3.6k ★)
  - lobsters: forum app (2.4k ★)
  - huginn: web-scraping app (22.6k ★)

Evaluation

- 4 open-source popular web applications built with Rails
  - kandan: chatting app (2.8k ★)
  - redmine: project management app (3.6k ★)
  - lobsters: forum app (2.4k ★)
  - huginn: web-scraping app (22.6k ★)
- Compare against app impl with 3 relational databases:
  - Original setting with MySQL (in-memory)
  - PostgreSQL (in-memory)
  - Commercial in-memory column-store

Evaluation: Web Applications

- 4 open-source popular web applications built with Rails

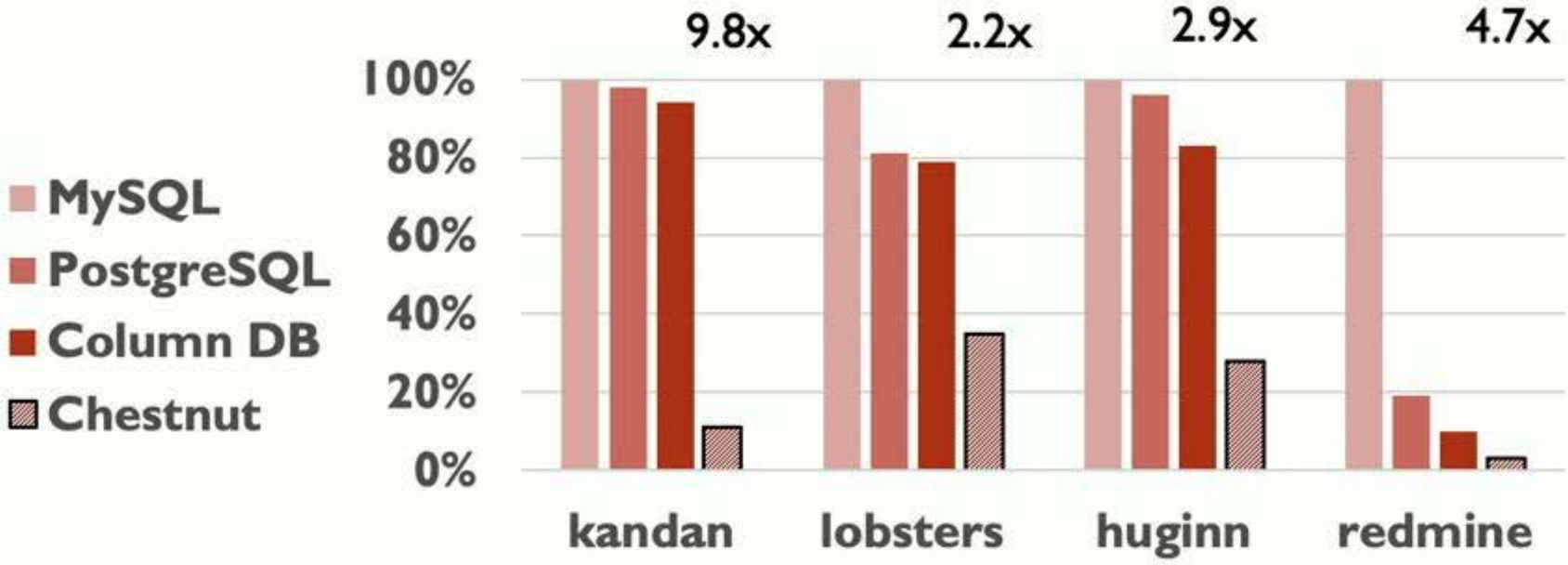


(avg query time, with the same memory)

- top-10 pages
- ~26 queries per app
- include read & write

Evaluation: Web Applications

- 4 open-source popular web applications built with Rails

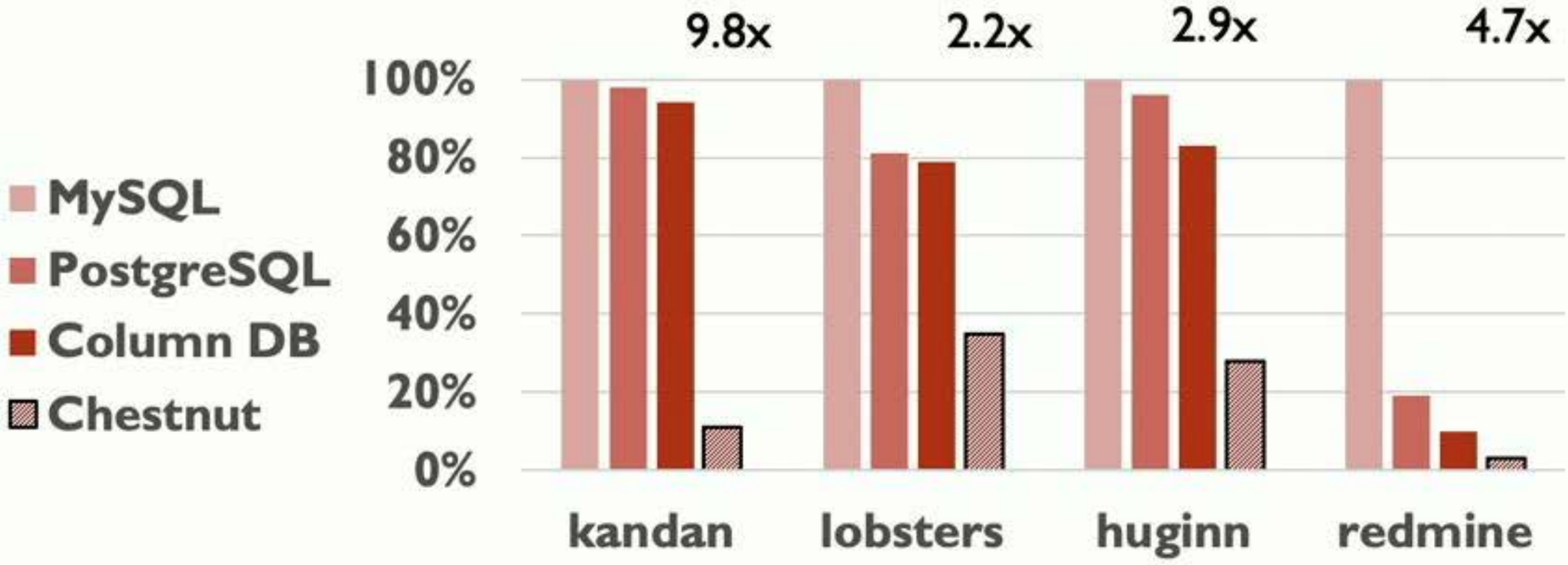


(avg query time, with the same memory)

- top-10 pages
- ~26 queries per app
- include read & write

Evaluation: Web Applications

- 4 open-source popular web applications built with Rails



(avg query time, with the same memory)

Chestnut running time:

- kandan: 1min
- redmine: 10min
- lobsters: 54min
- huginn: 3min

Chestnut Summary

VLDB'19

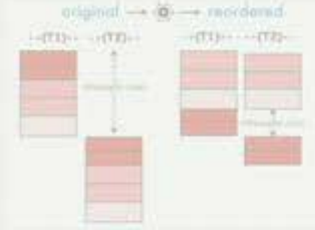
- Object-oriented database applications processes data in nested format, different from the tabular data layout used in DB.
- We propose Chestnut, a data layout designer that customizes data layout and query execution for each application.
- It uses enumeration (layout + query plan) and ILP solver to search for the best data layout given an application.
- Evaluation shows chestnut works for real-world applications, speeding up app queries up to 9.8x.

Outline

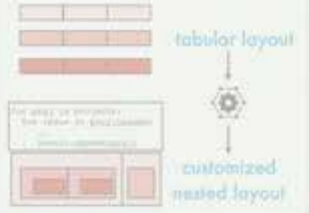
- Leveraging application semantics to optimize each layer

Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout



- Other projects
- Ongoing and future work

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

140 performance issues
from bug tracking system

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

140 performance issues
from bug tracking system

64 performance issues
from profiling

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

140 performance issues from bug tracking system

64 performance issues from profiling

9 anti-patterns

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

140 performance issues from bug tracking system

64 performance issues from profiling

9 anti-patterns

.any? ✗ .exists? ✓

Anti-patterns In Web Apps

- A comprehensive study on 12 popular open-source Rails Applications

[ICSE'18]

140 performance issues from bug tracking system

64 performance issues from profiling

9 anti-patterns

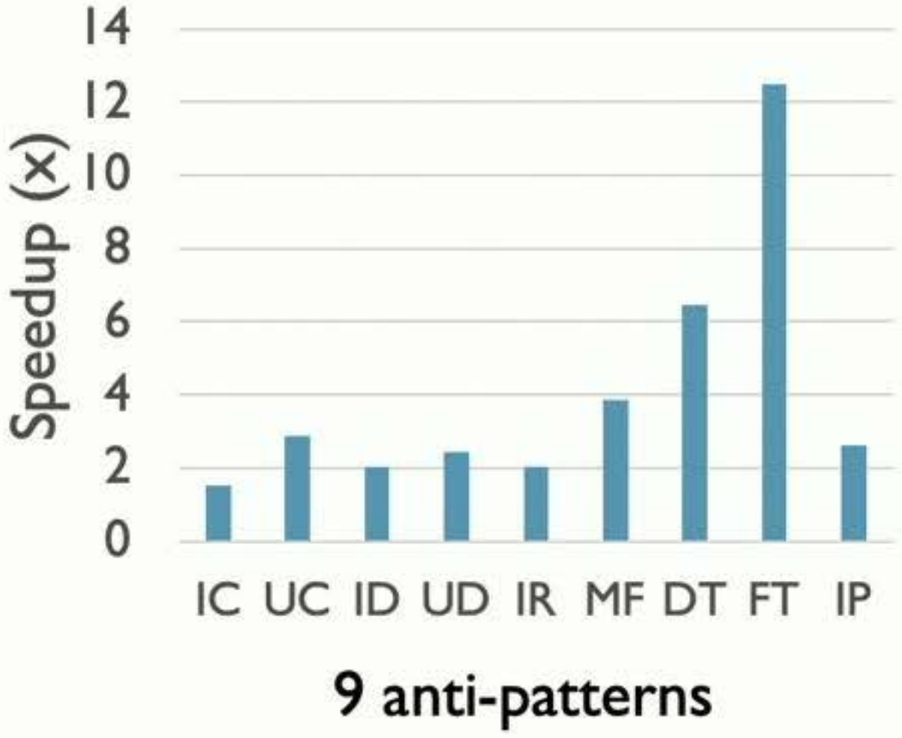
.any? ✗ .exists? ✓

select * from... ✗ select f1 from ... ✓

Inefficiencies Fixed by Manual Patch

- We manually fix the 64 issues found across 40 pages from 12 apps
- Avg speedup of each anti-pattern fix

[ICSE'18]



Automatic Fix of Anti-patterns

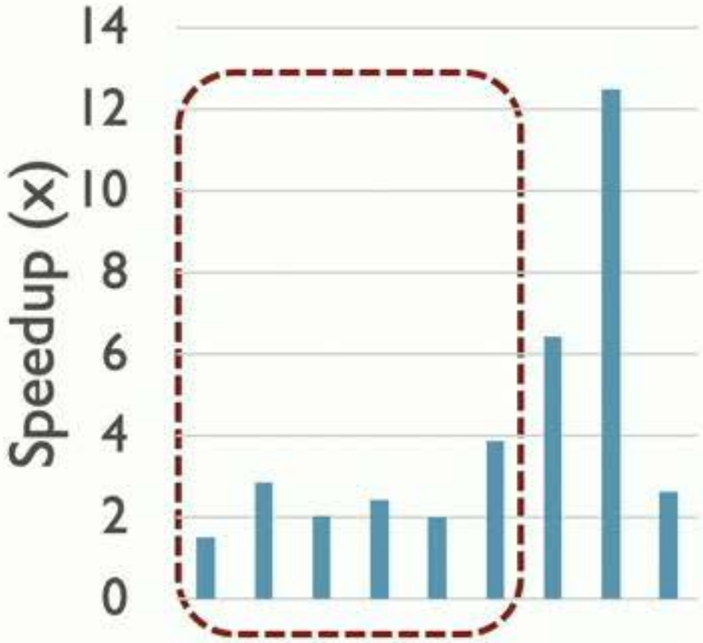
Powerstation

[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns



Automatic Fix of Anti-patterns

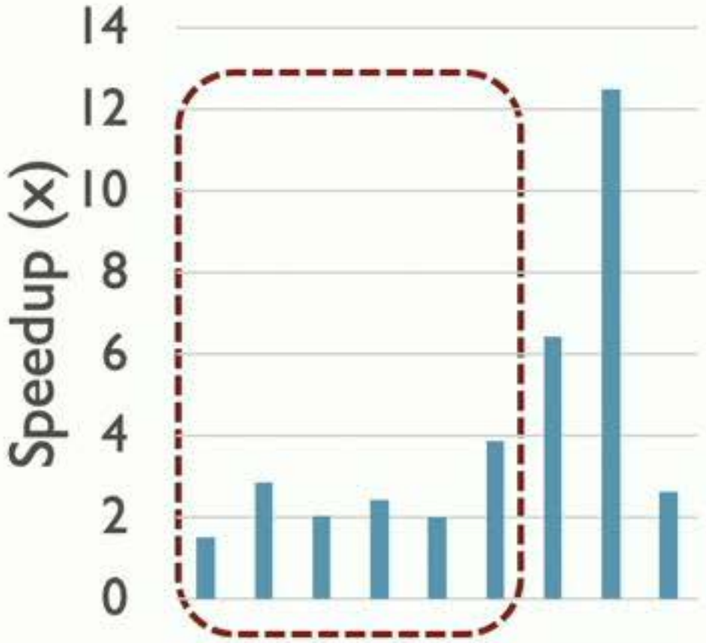
Powerstation

[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns



Semantic-preserving!

Automatic Fix of Anti-patterns

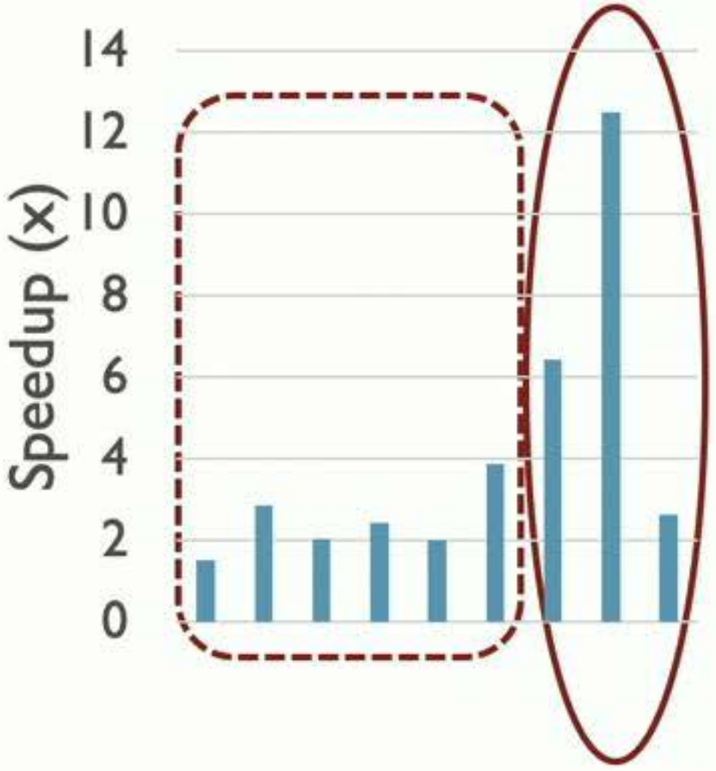
Powerstation

[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns



Semantic-preserving!

Automatic Fix of Anti-patterns

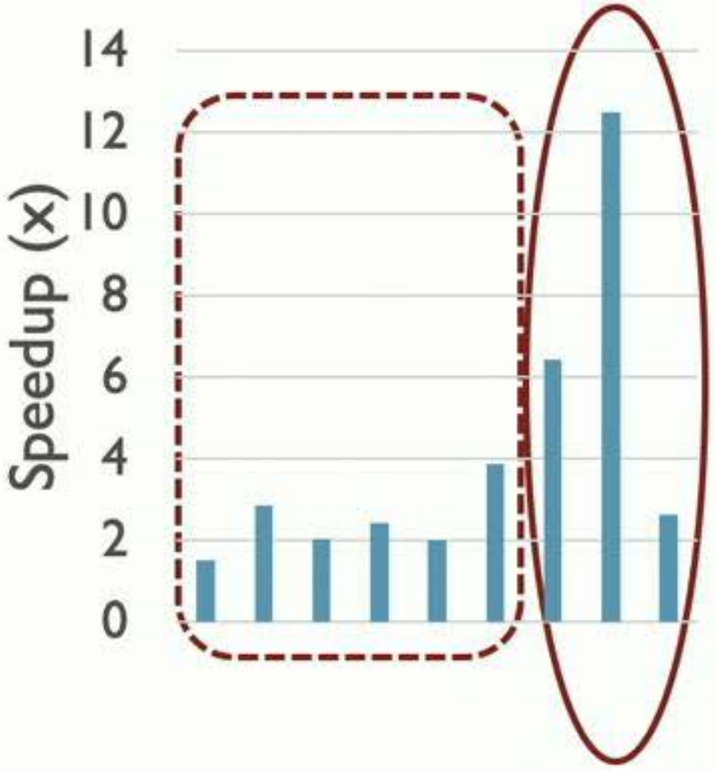
Powerstation

[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns



Non-semantic-preserving!

Webpage design-performance tradeoff

Semantic-preserving!

Automatic Fix of Anti-patterns

Powerstation

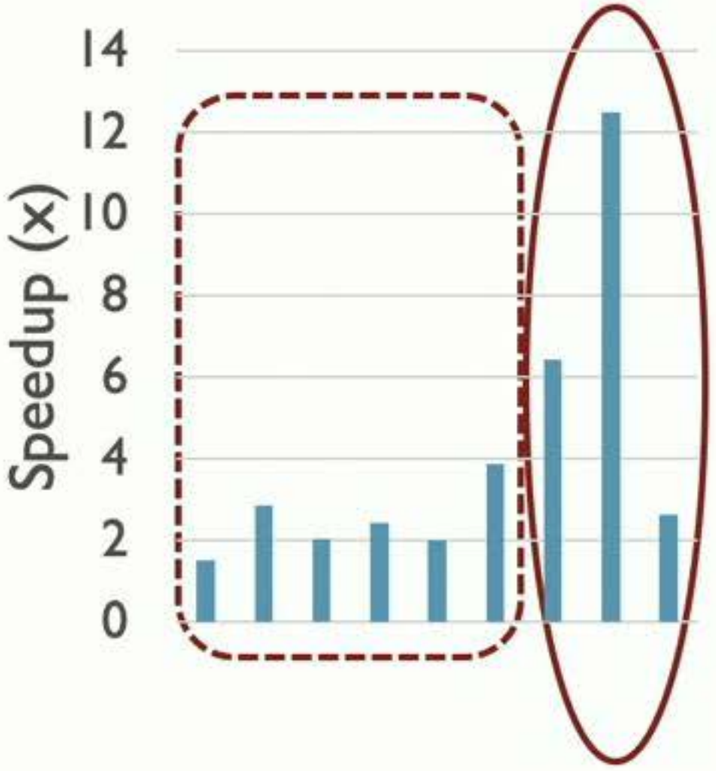
[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns

Semantic-preserving!



Panorama

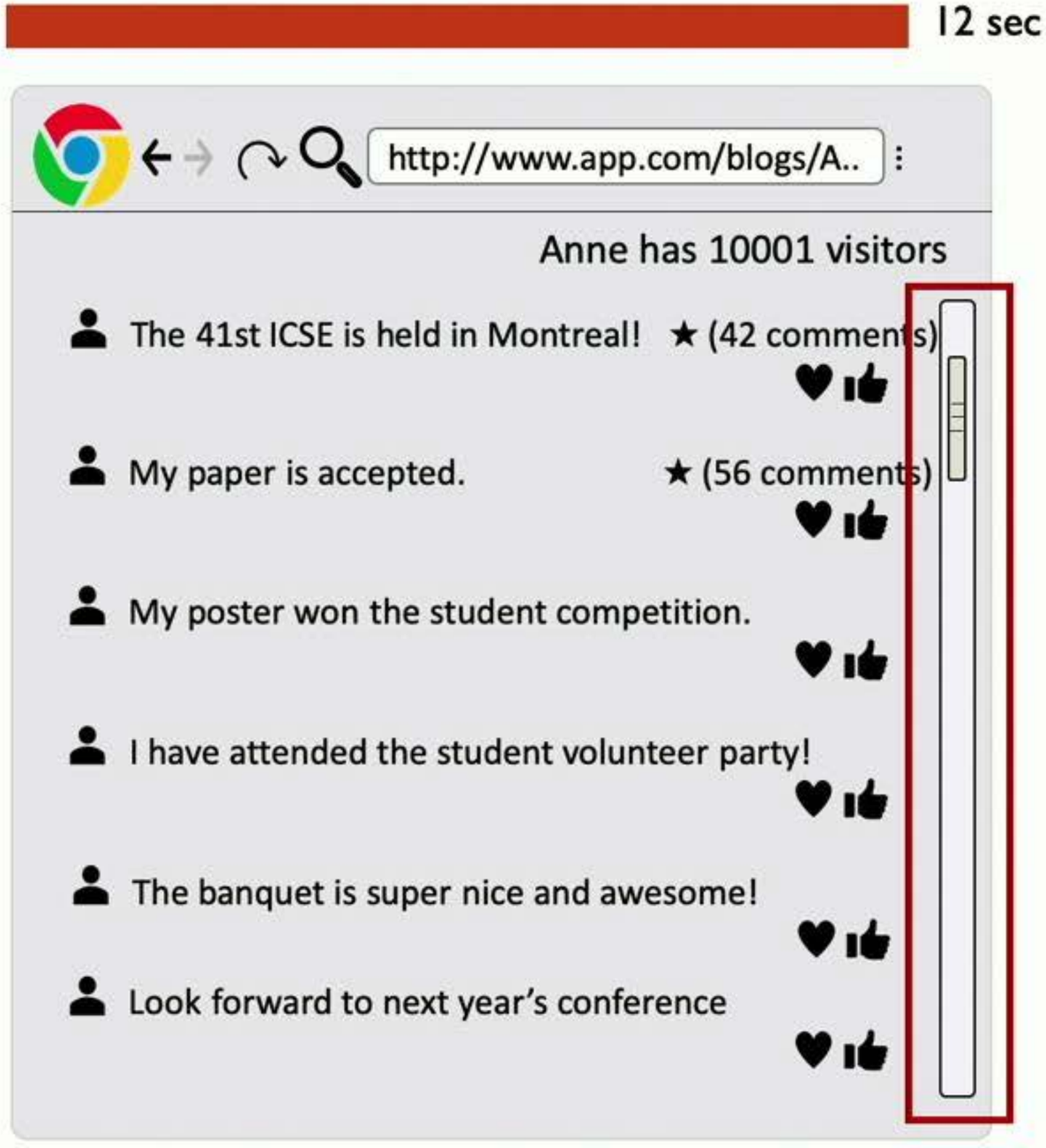
Non-semantic-preserving!

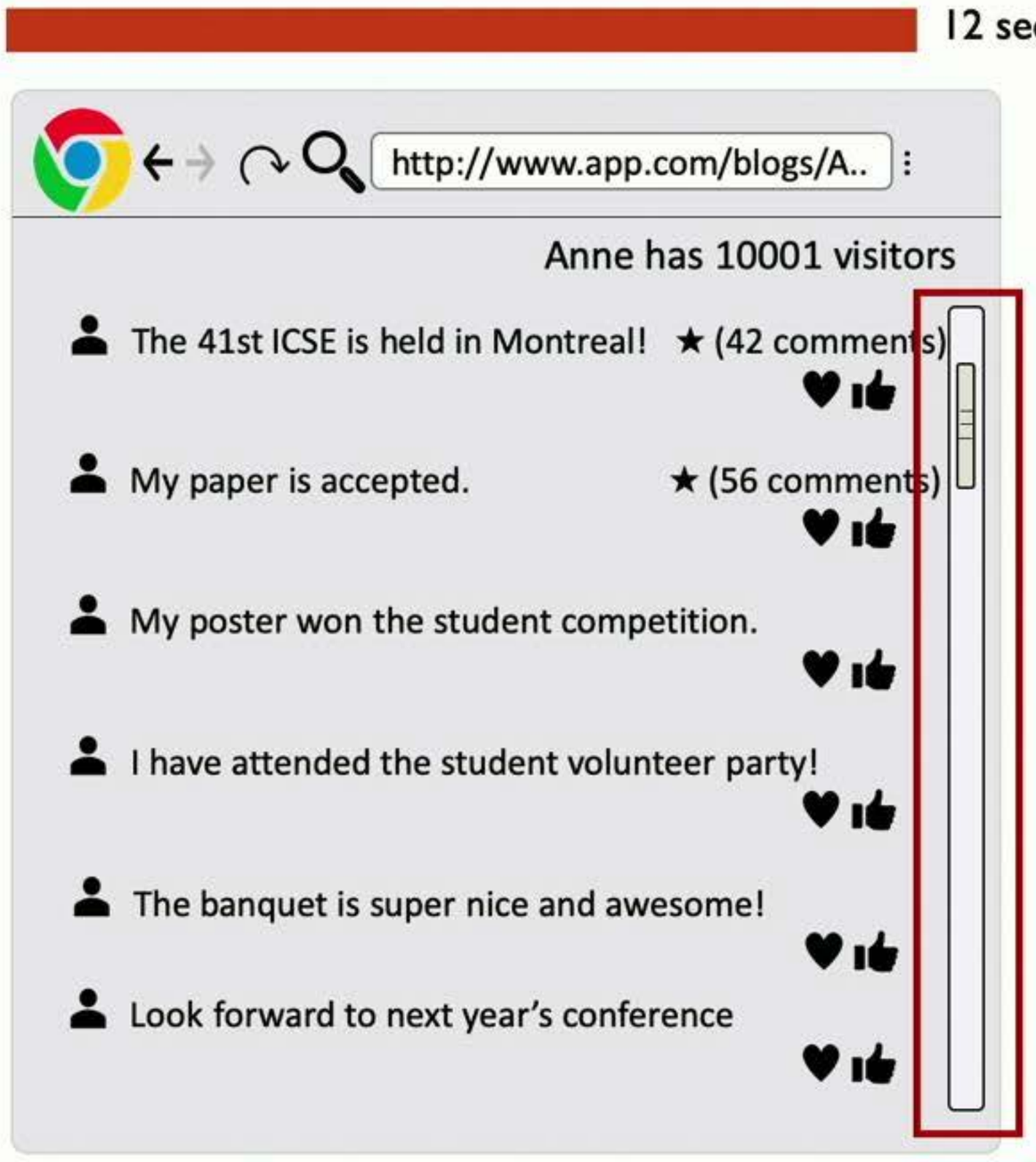
Webpage design-performance tradeoff

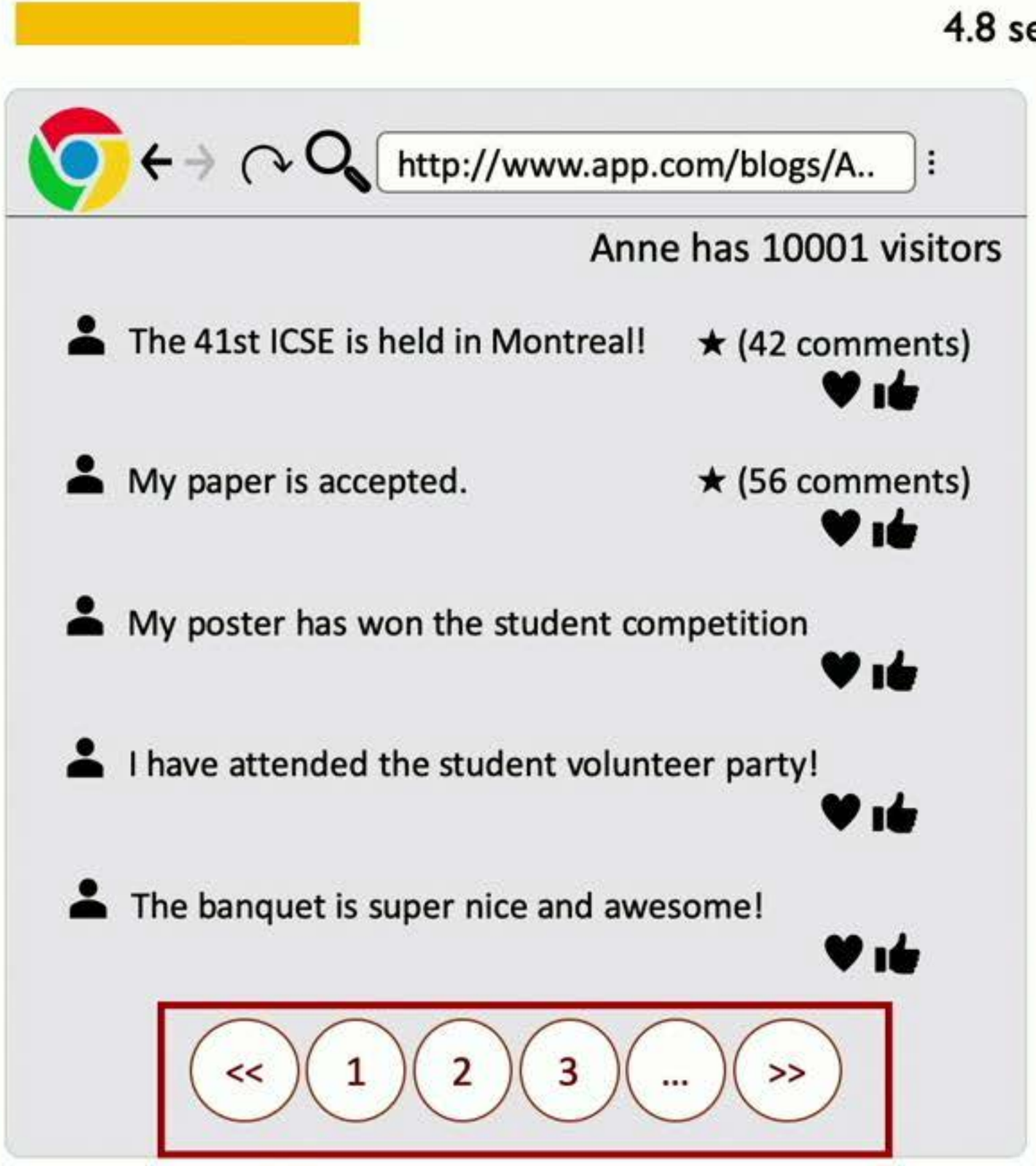

http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference

12 sec


http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference

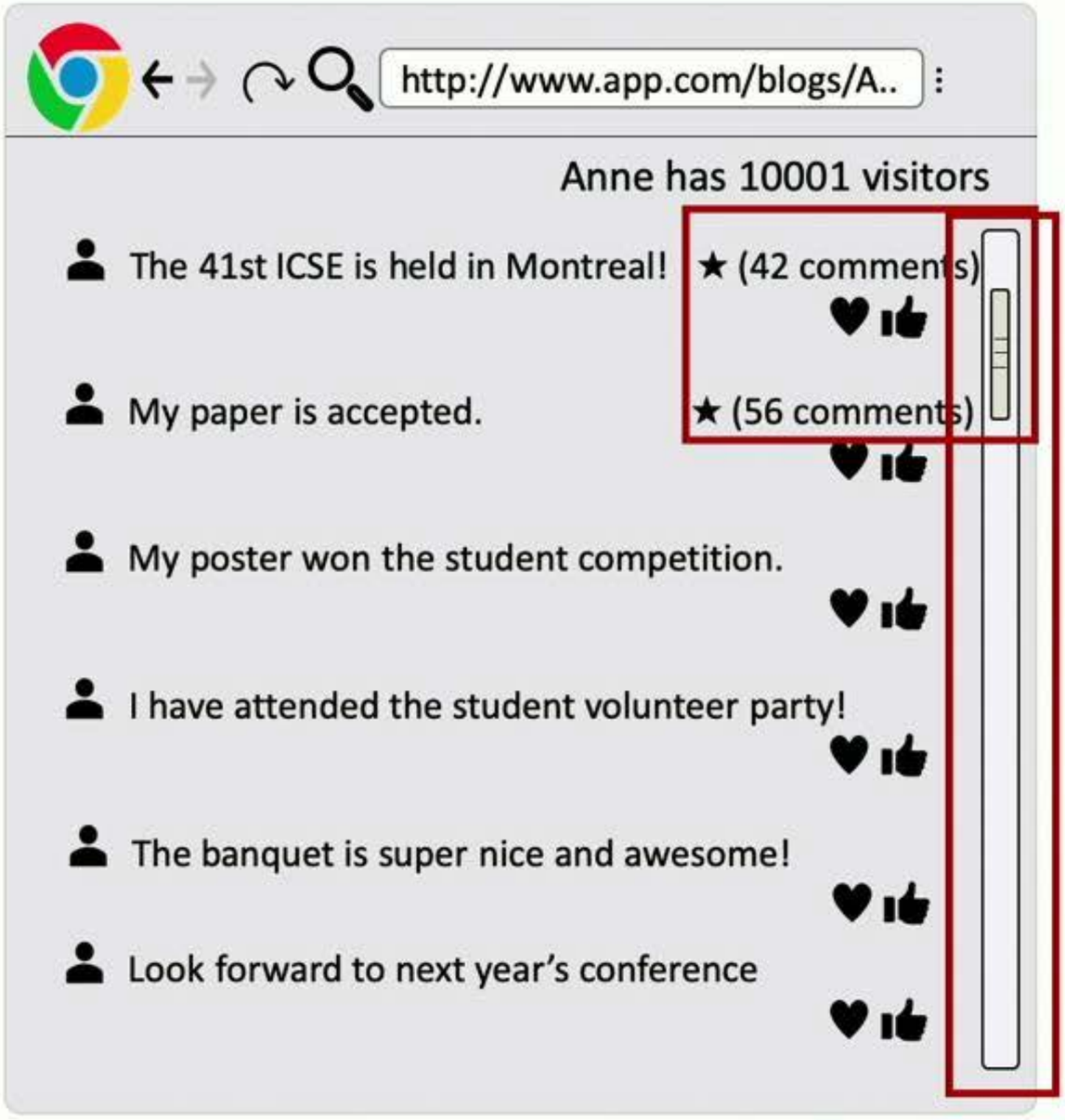

12 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference

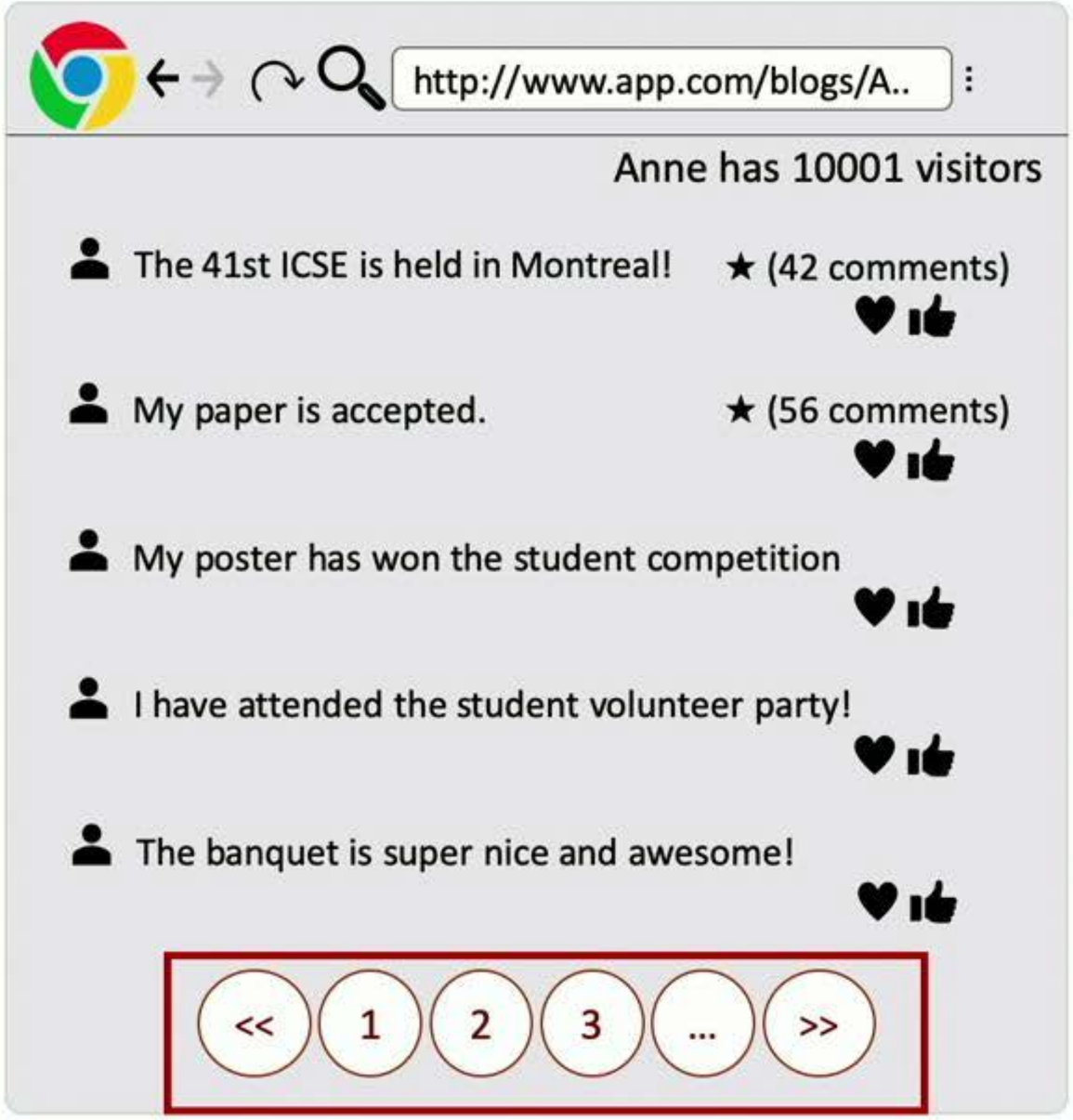

12 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference

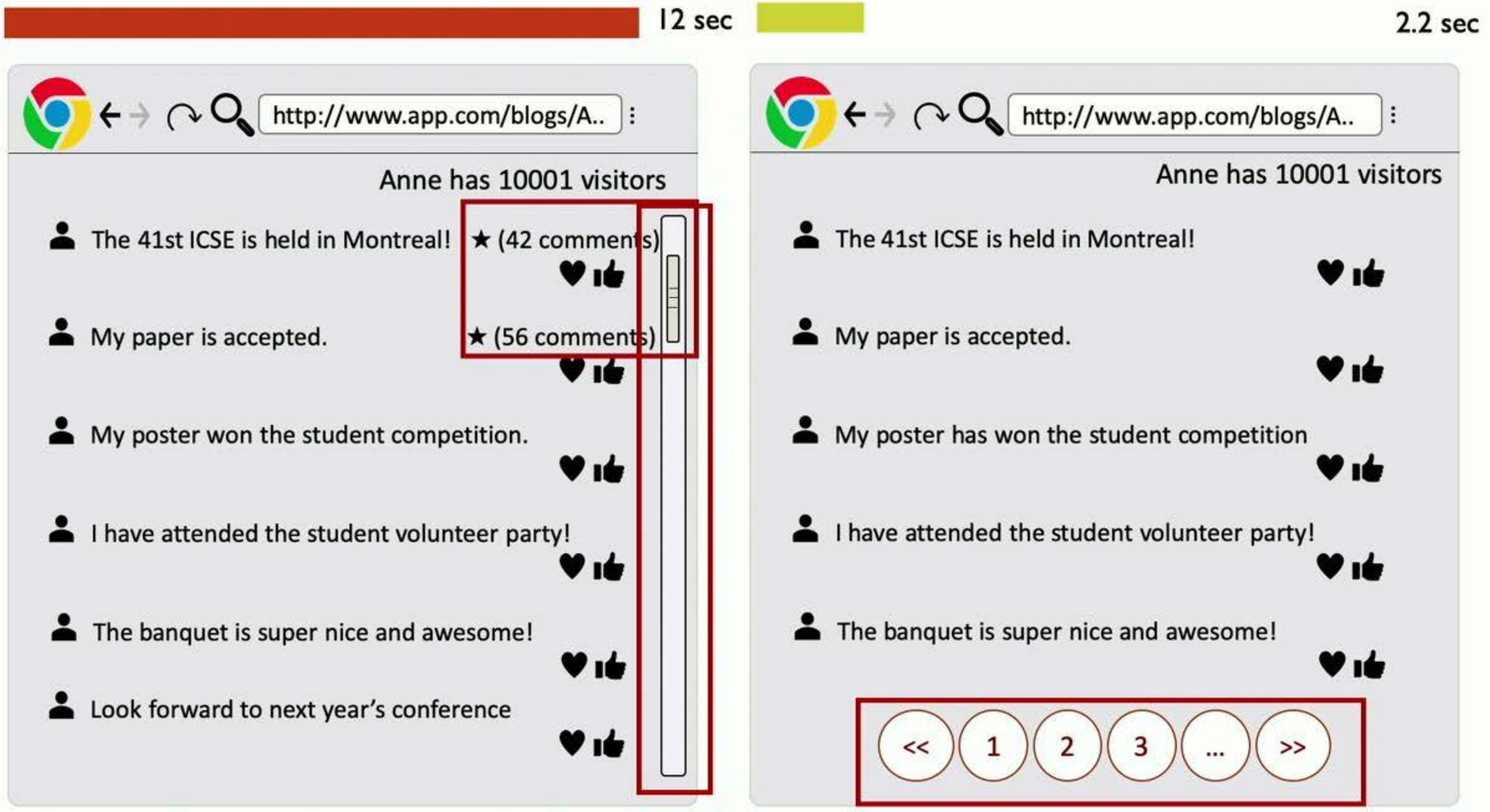

4.8 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster has won the student competition
I have attended the student volunteer party!
The banquet is super nice and awesome!
<< 1 2 3 ... >>

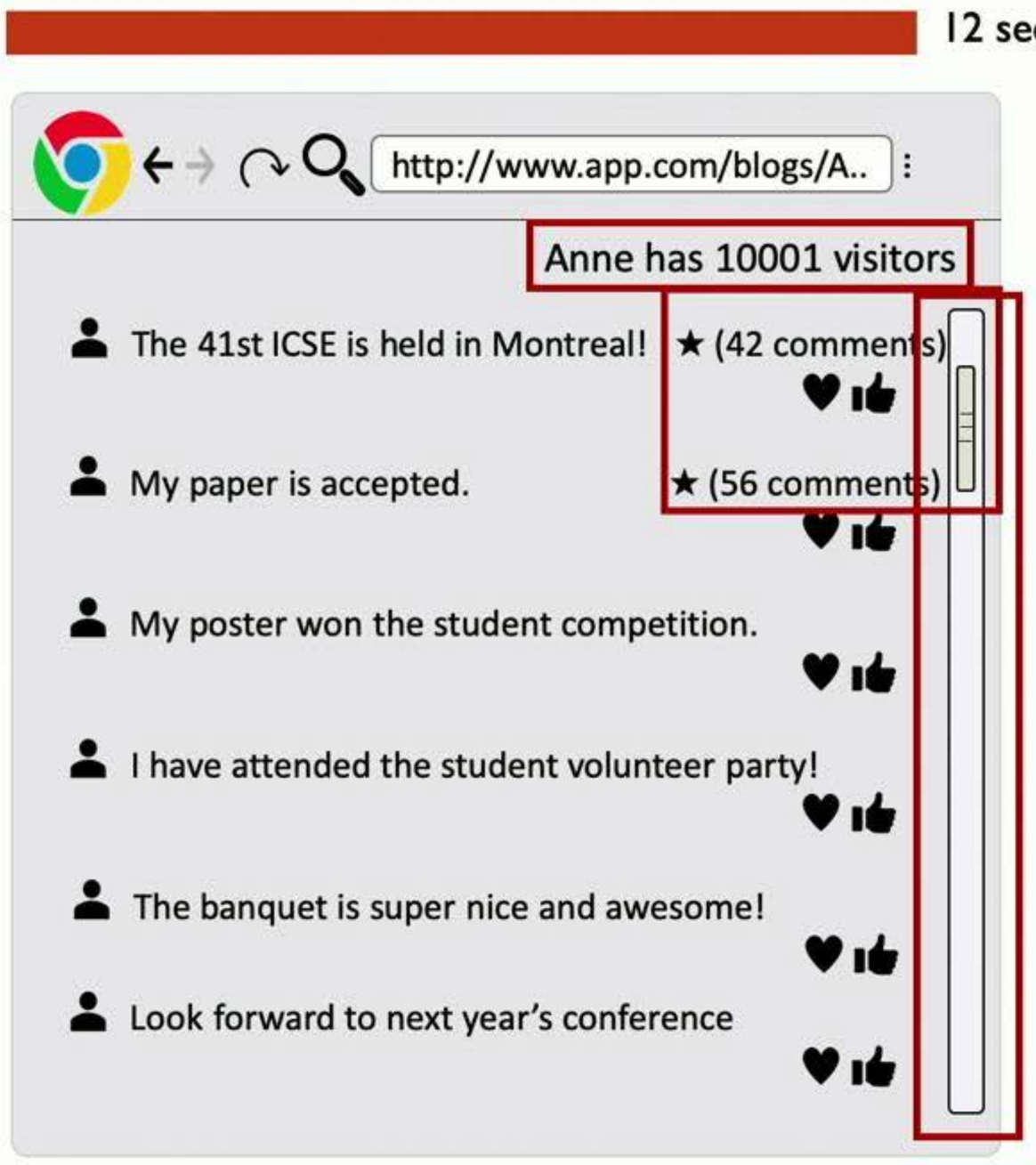
12 sec

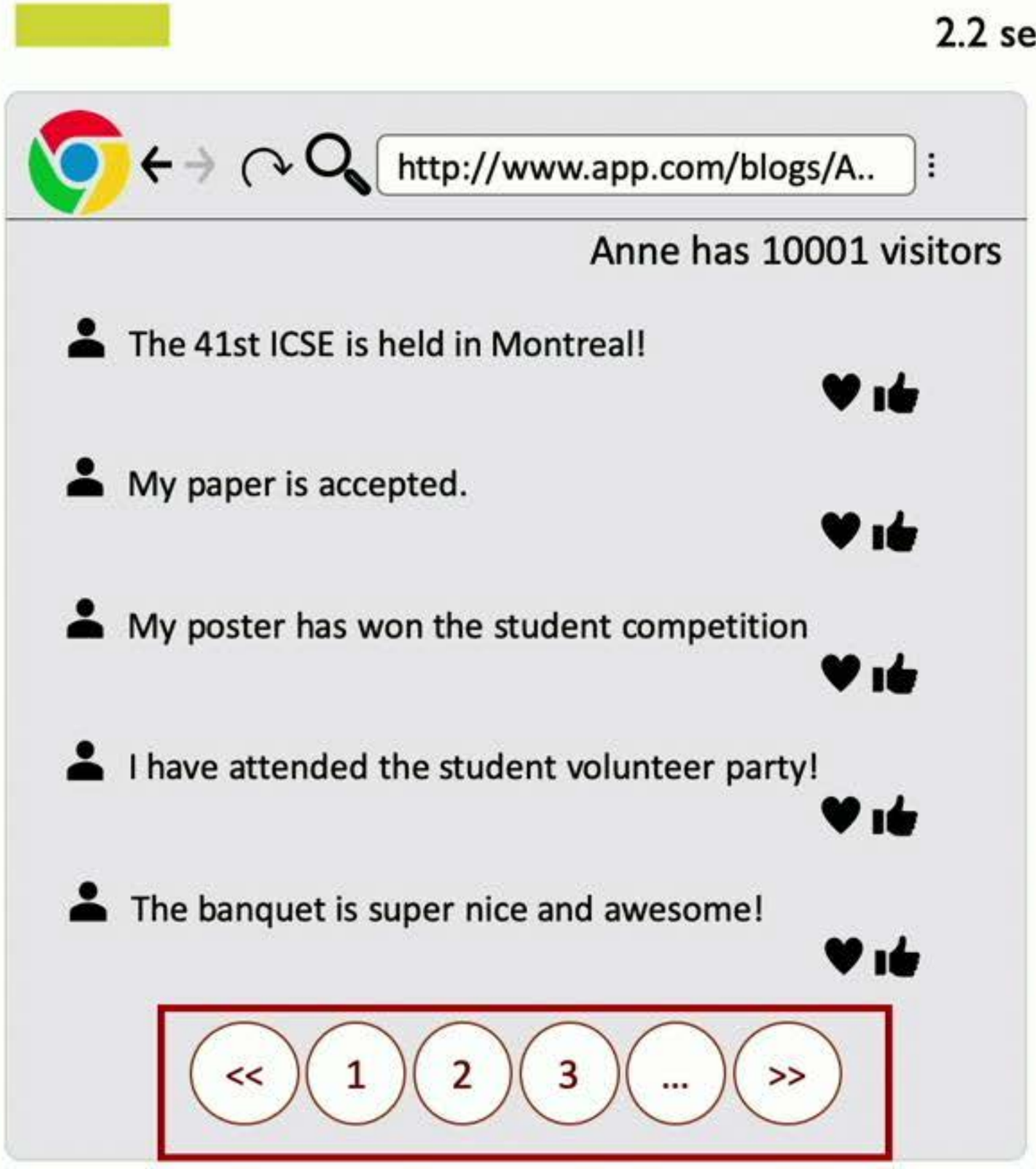
4.8 sec

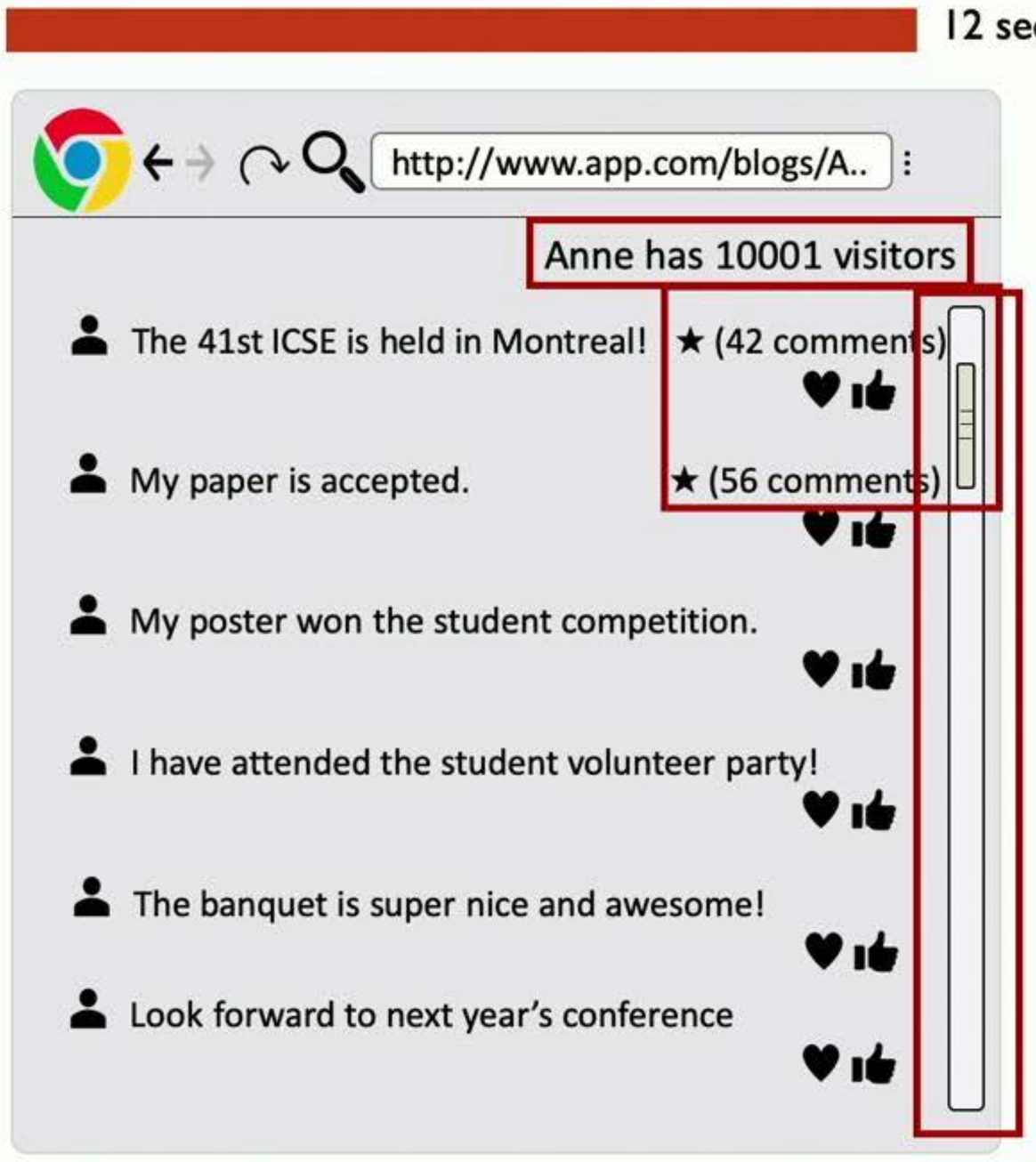

http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference

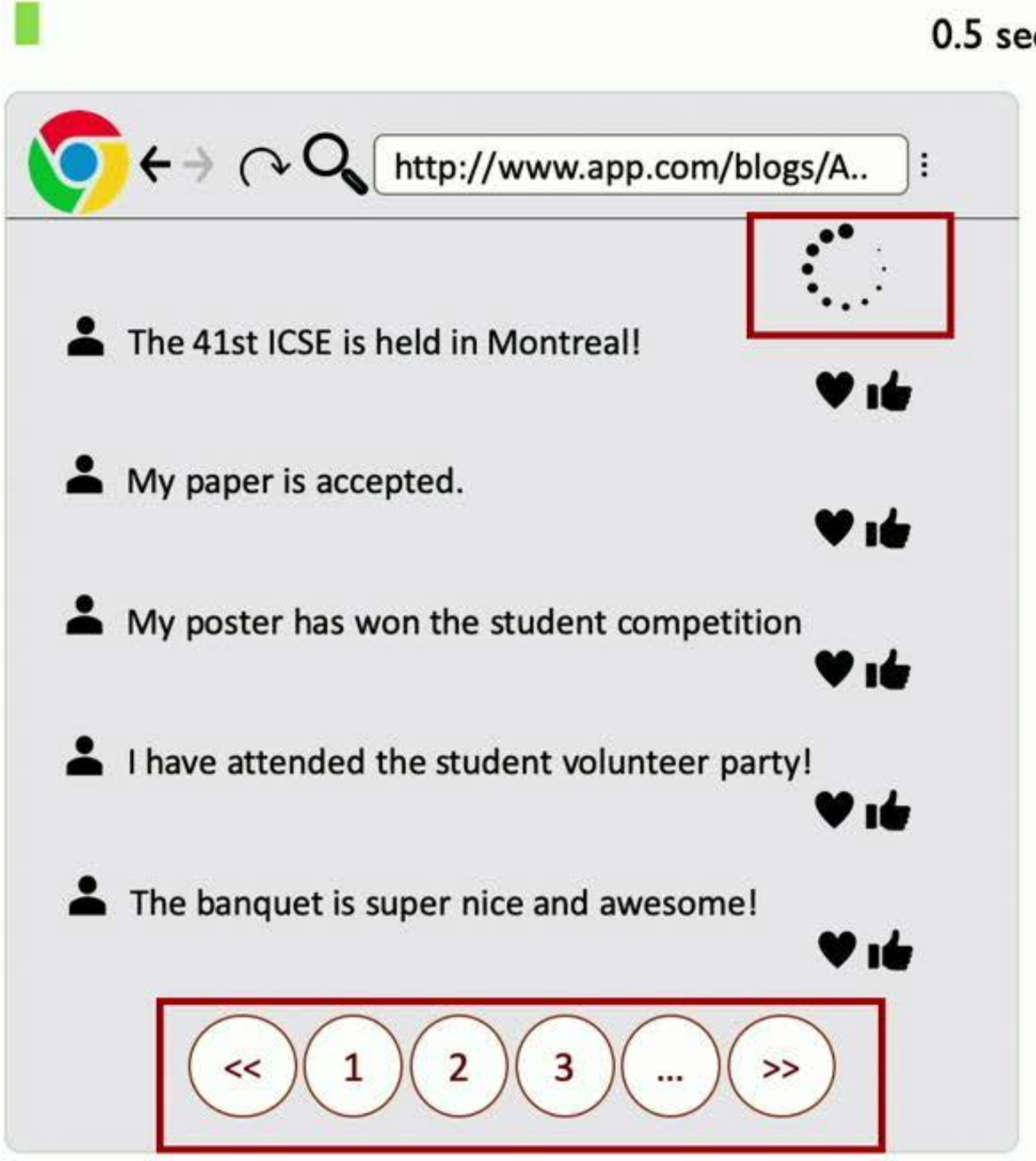

http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal! ★ (42 comments)
My paper is accepted. ★ (56 comments)
My poster has won the student competition
I have attended the student volunteer party!
The banquet is super nice and awesome!
<< 1 2 3 ... >>


12 sec
2.2 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal!
★ (42 comments)
My paper is accepted.
★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal!
My paper is accepted.
My poster has won the student competition
I have attended the student volunteer party!
The banquet is super nice and awesome!
<< 1 2 3 ... >>


12 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal!
★ (42 comments)
My paper is accepted.
★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference


2.2 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal!
My paper is accepted.
My poster has won the student competition
I have attended the student volunteer party!
The banquet is super nice and awesome!
<<
1
2
3
...
>>


12 sec
http://www.app.com/blogs/A..
Anne has 10001 visitors
The 41st ICSE is held in Montreal!
★ (42 comments)
My paper is accepted.
★ (56 comments)
My poster won the student competition.
I have attended the student volunteer party!
The banquet is super nice and awesome!
Look forward to next year's conference


0.5 sec
http://www.app.com/blogs/A..
The 41st ICSE is held in Montreal!
My paper is accepted.
My poster has won the student competition
I have attended the student volunteer party!
The banquet is super nice and awesome!
<<
1
2
3
...
>>

Making Design-Performance Tradeoff is Not Trivial

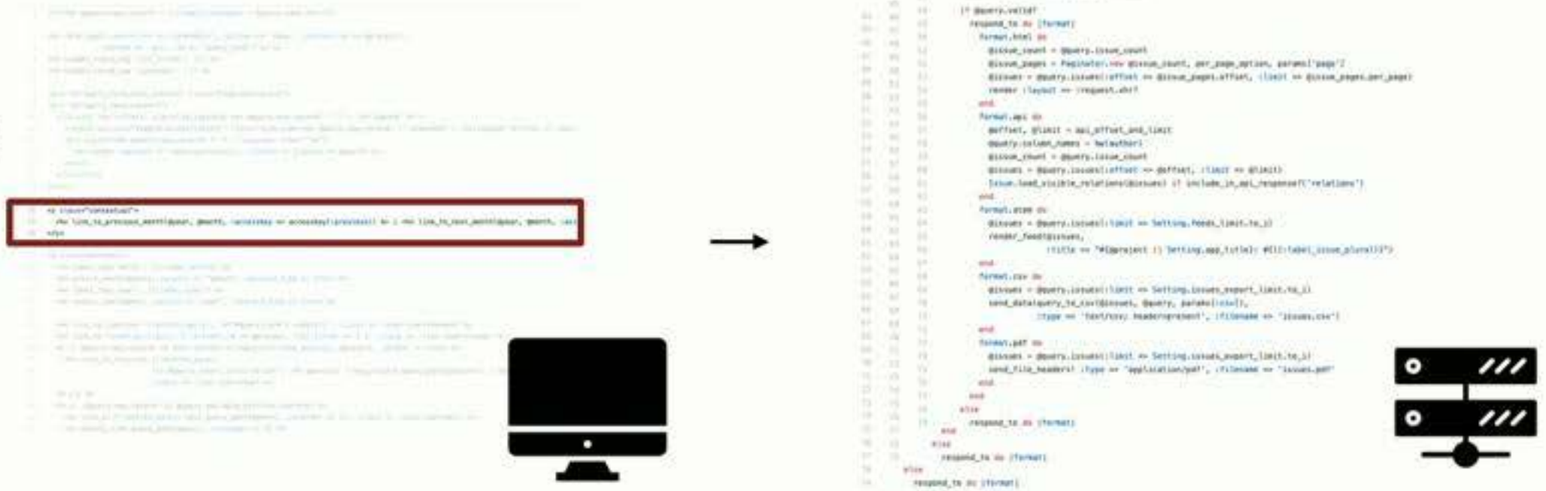
- Changing code across layers

Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers

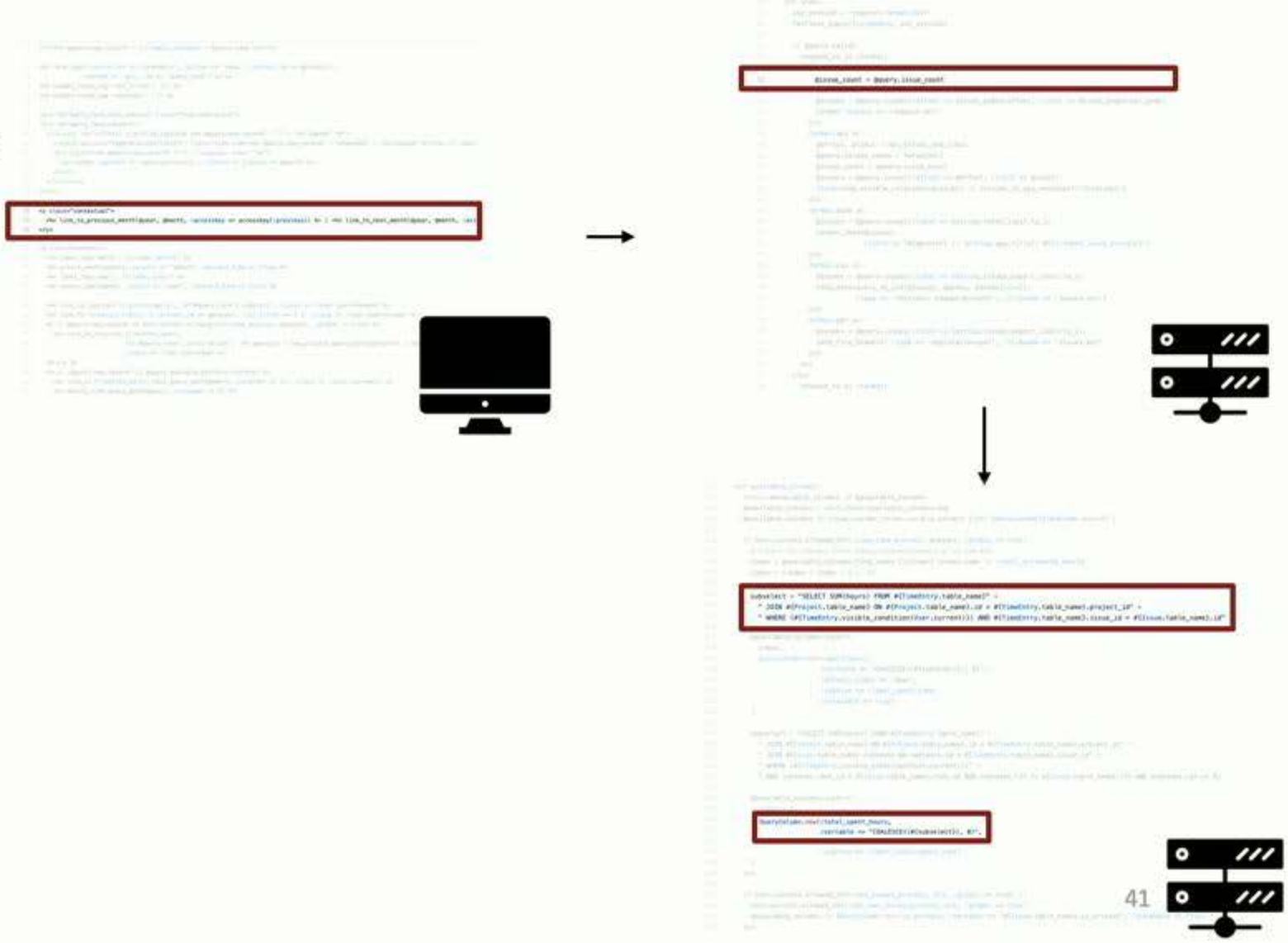
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers



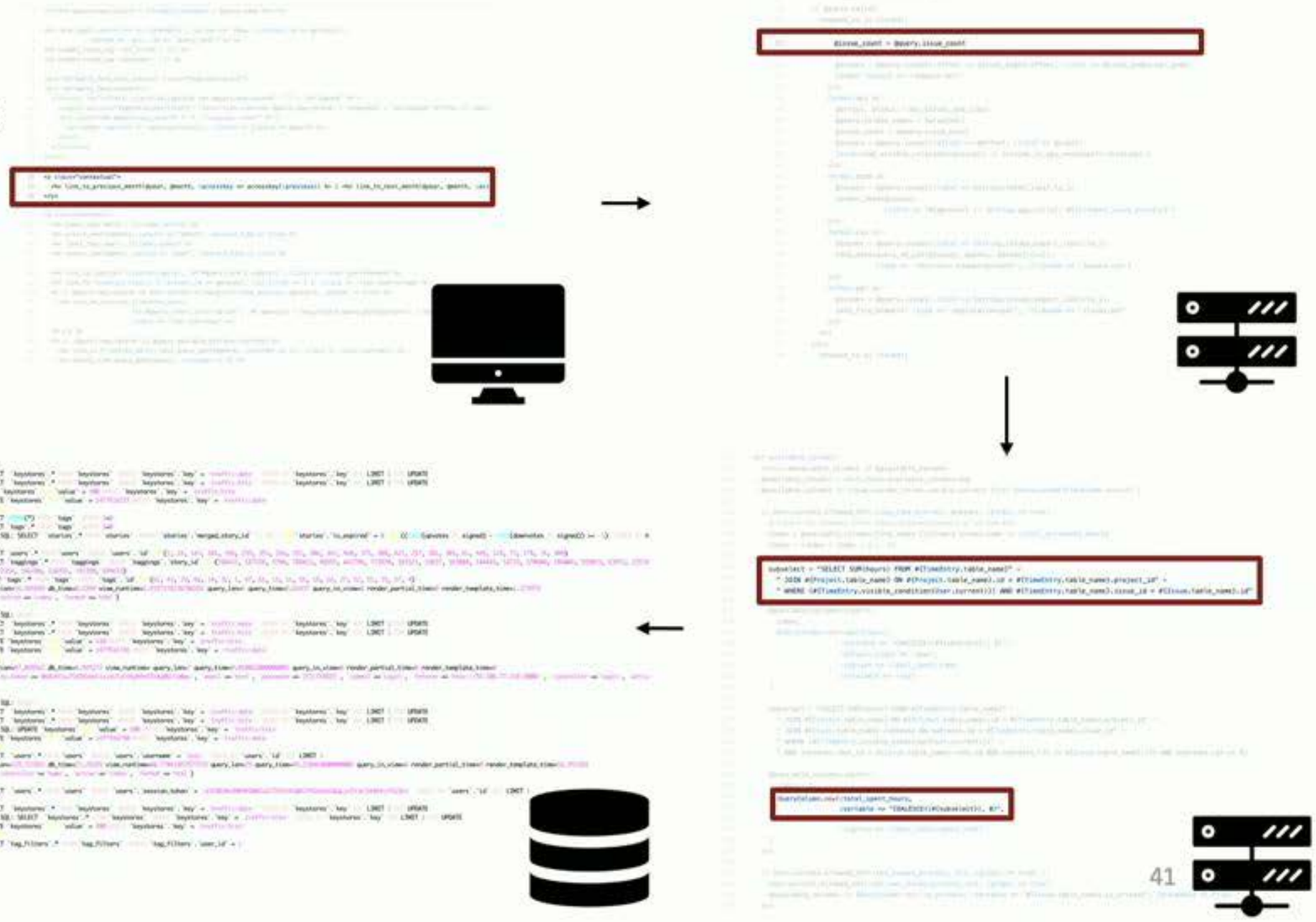
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers



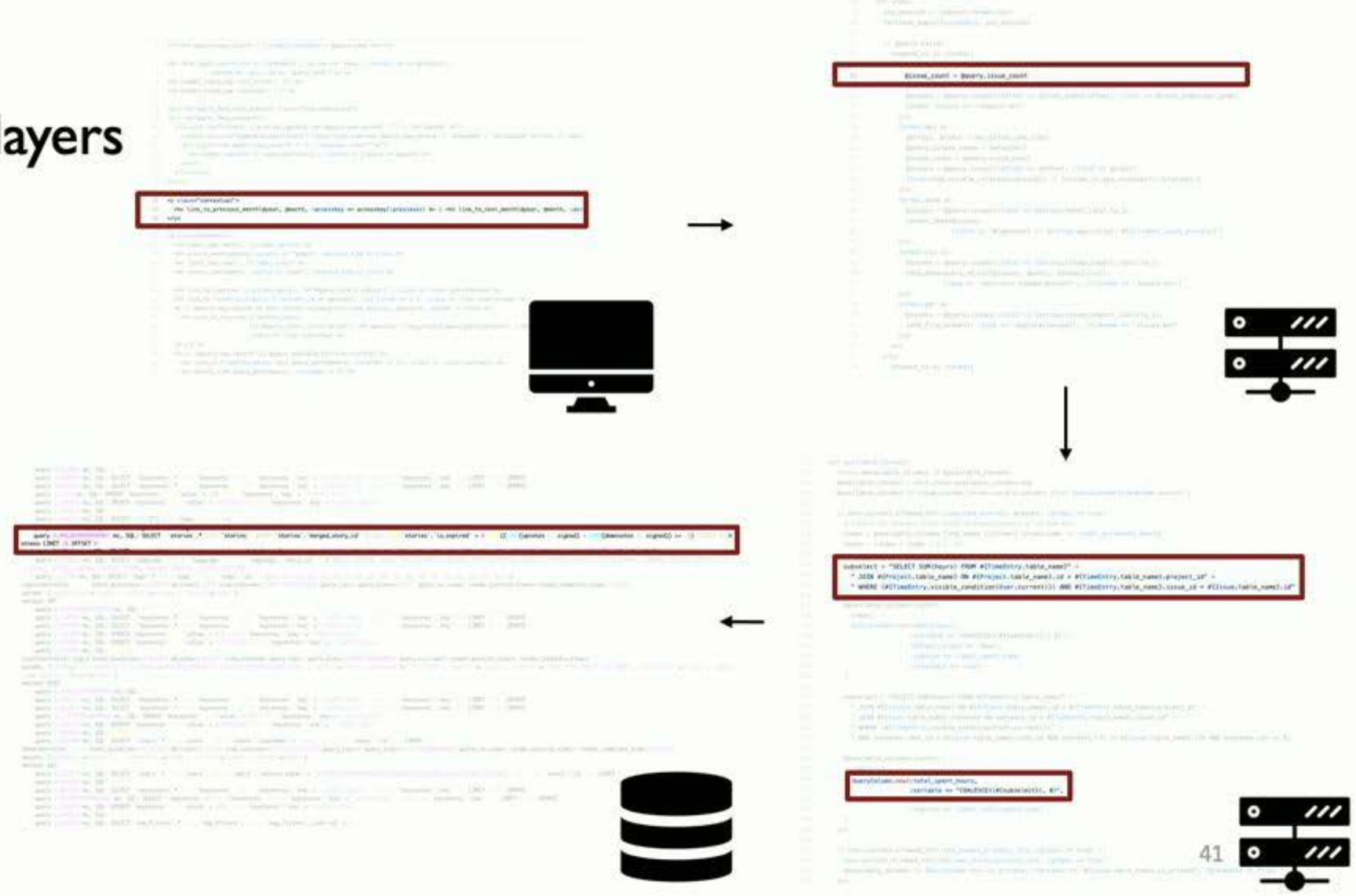
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers



Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers

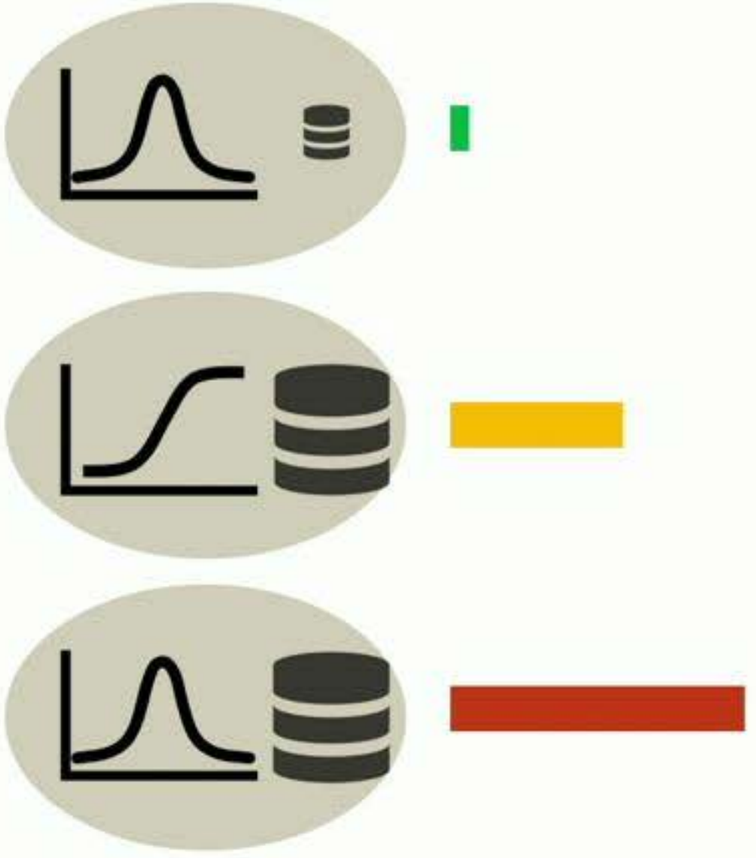


Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers
- Understanding the tradeoff

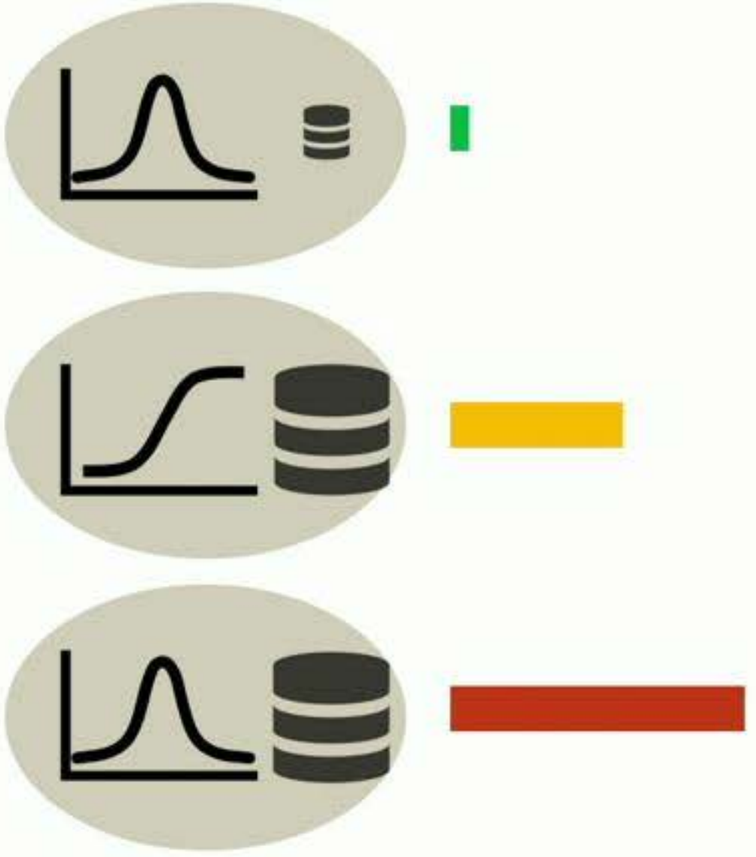
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers
- Understanding the tradeoff

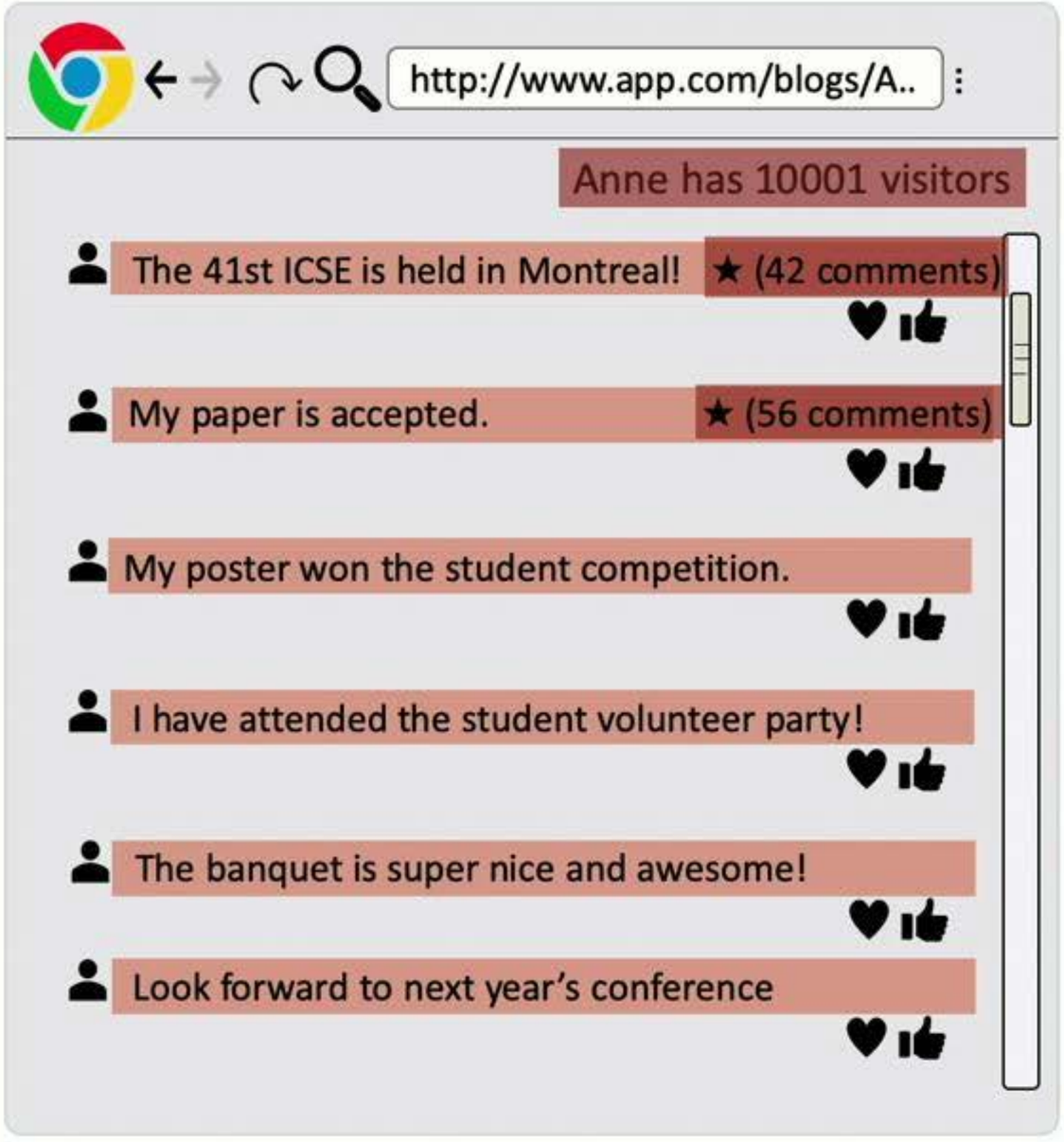


Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers
- Understanding the tradeoff



Panorama User Interface: Heatmap



How does Panorama generate heatmap?

How does Panorama generate heatmap?

index.html.erb

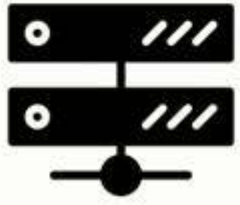
```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

blogs_controller.rb

```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

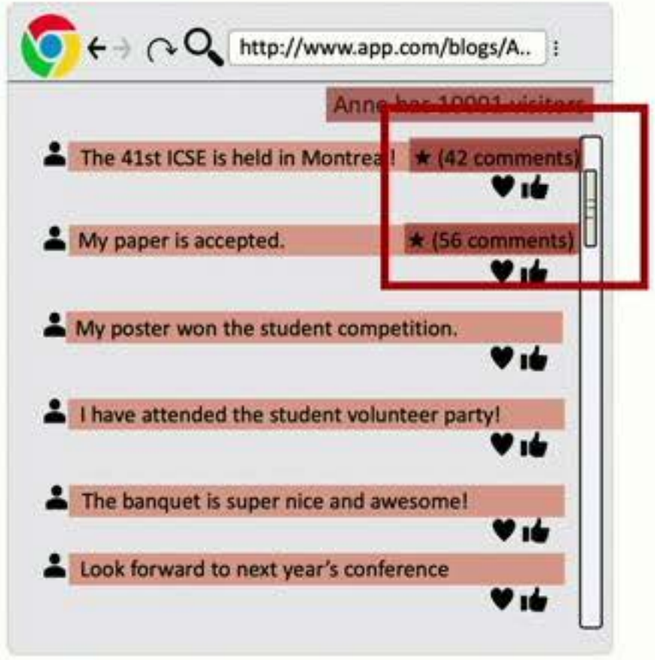
blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```



```
select count(*) from
blogs join … where …
group by …
```

How does Panorama generate heatmap?



index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

blogs_controller.rb

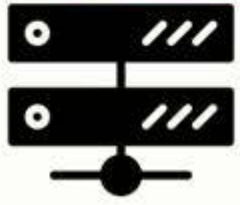
```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

$N_{blogs} * N_{comments}$

```
select count(*) from
blogs join … where …
group by …
```



How does Panorama generate heatmap?

$N_{blogs} * N_{comments}$

index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

blogs_controller.rb

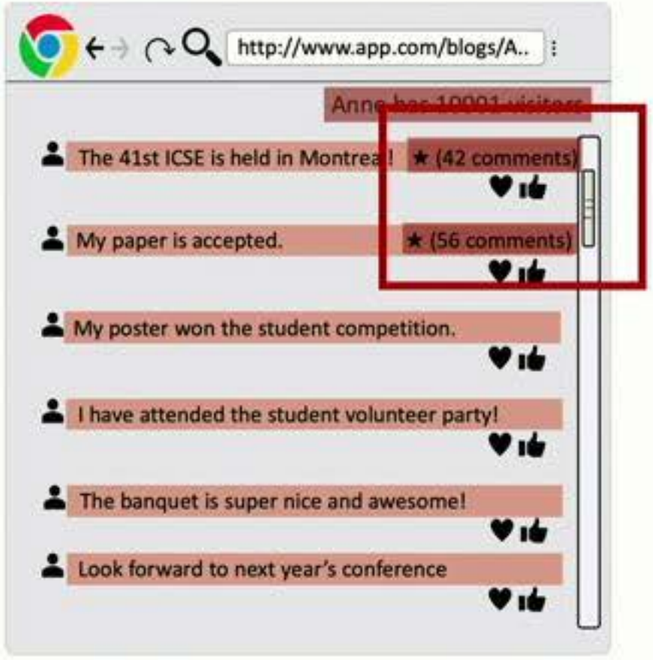
```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

```
select count(*) from
blogs join … where …
group by …
```

How does Panorama generate heatmap?



$N_{blogs} * N_{comments}$

index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

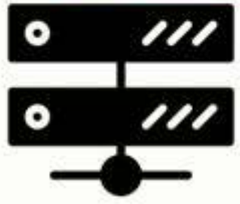
blogs_controller.rb

```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

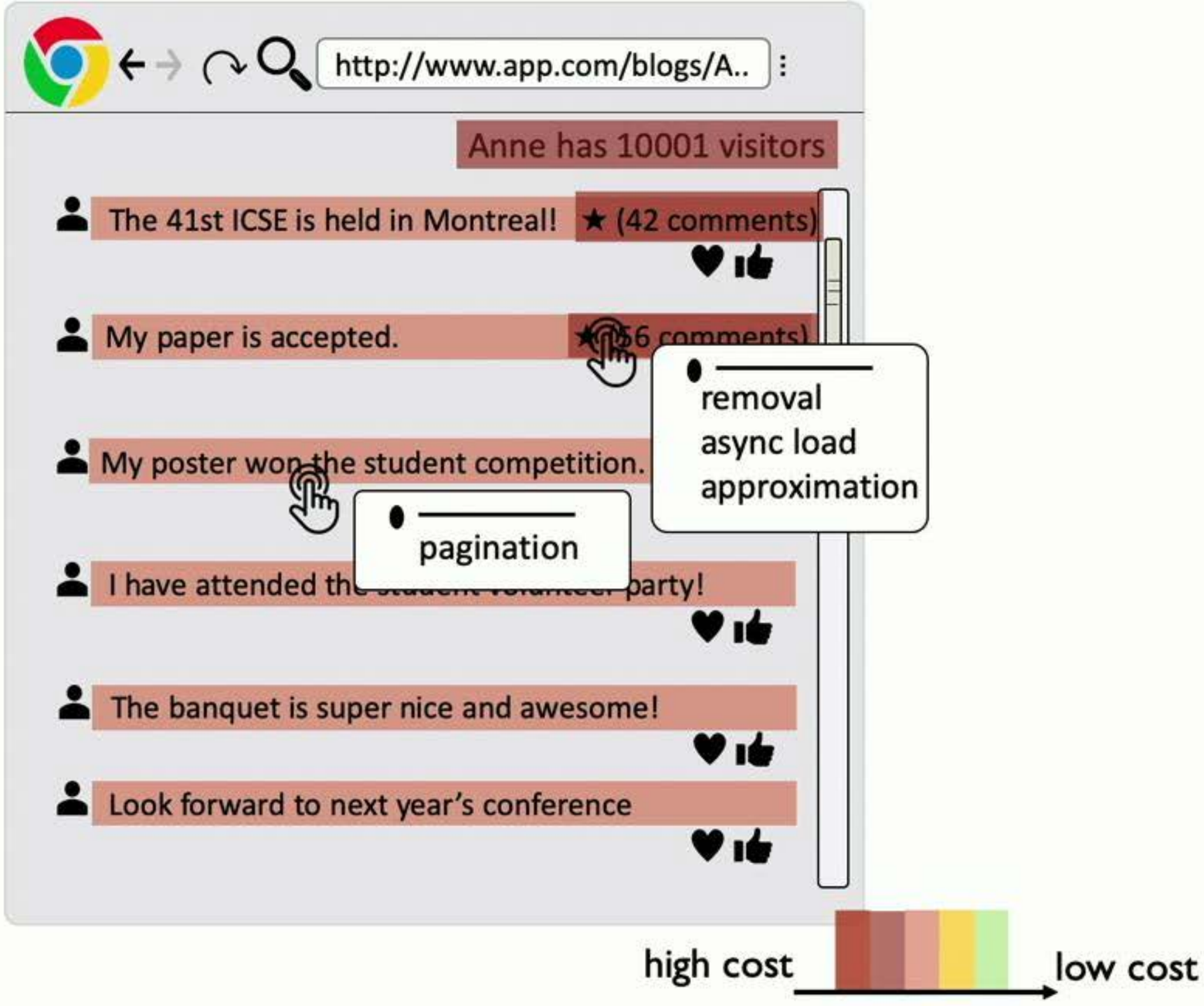
```
select count(*) from
blogs join … where …
group by …
```



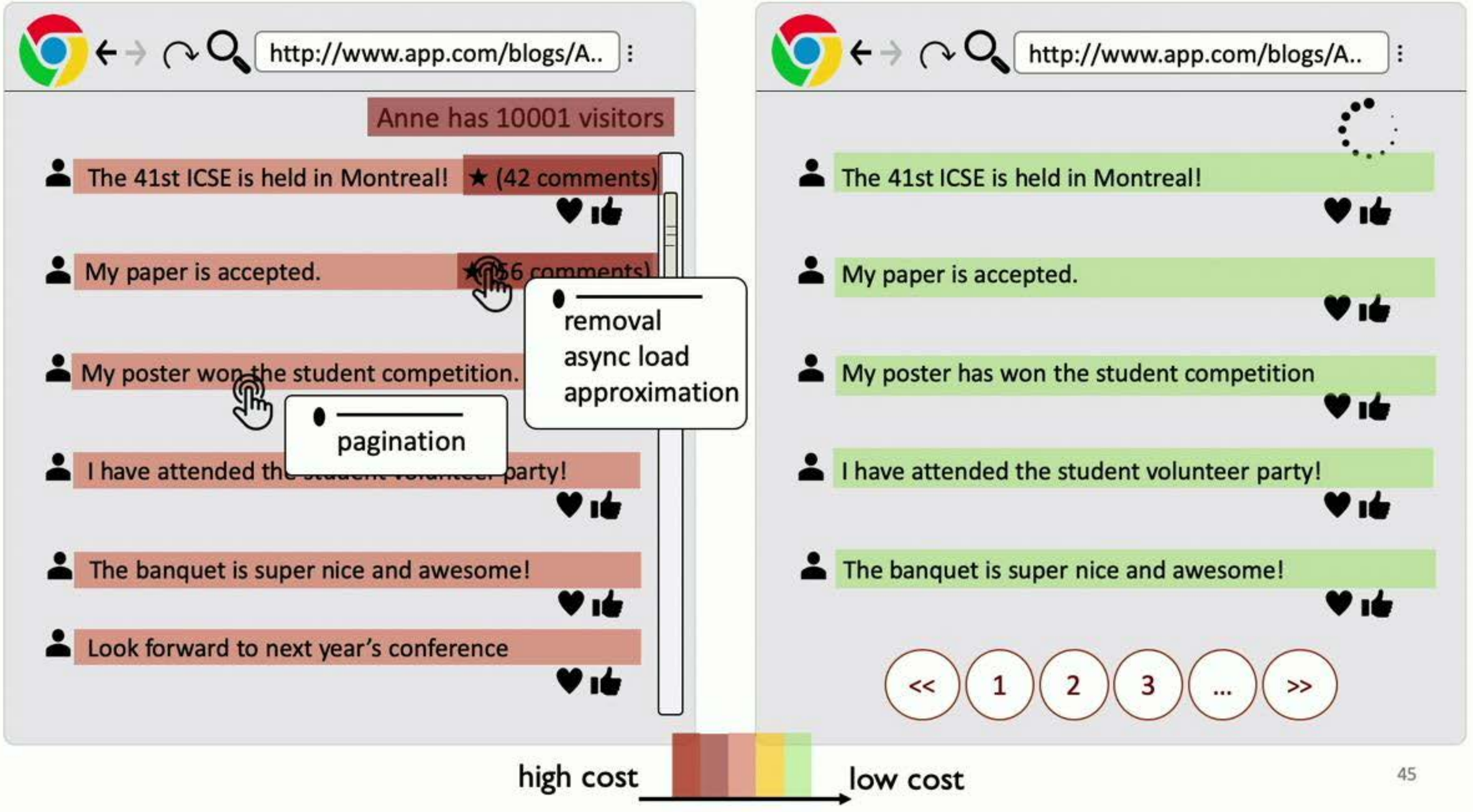
Panorama: Suggests Design Change and Refactors Code Automatically

high cost → low cost

Panorama: Suggests Design Change and Refactors Code Automatically



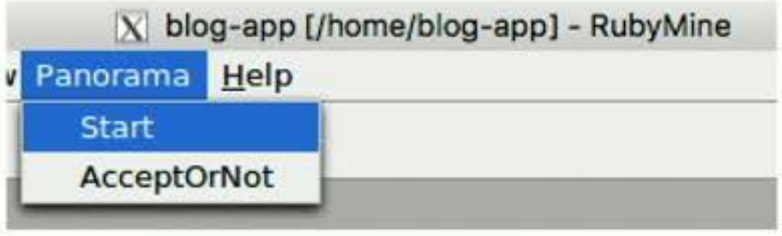
Panorama: Suggests Design Change and Refactors Code Automatically

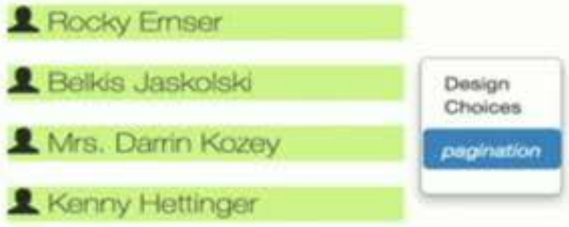


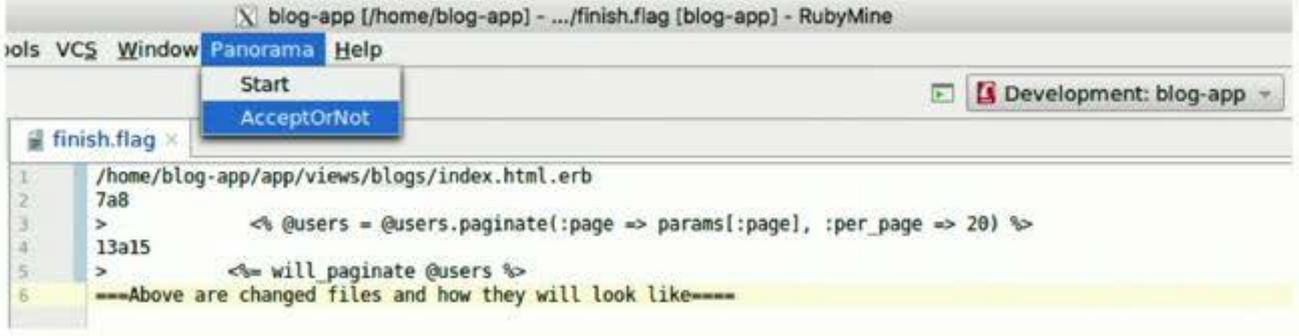
Panorama as An IDE Plugin

RubyMine

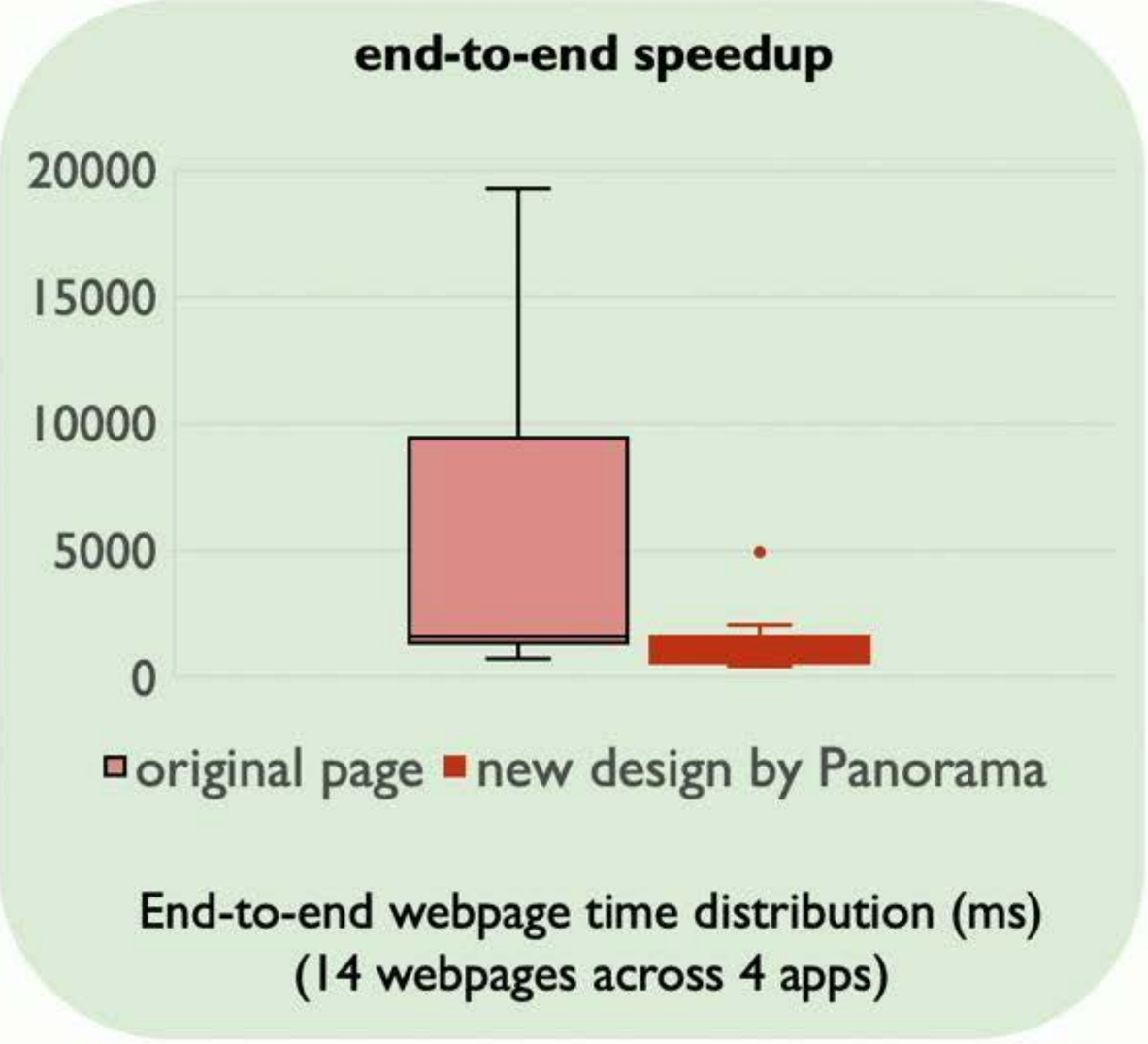
The Most Intelligent Ruby and Rails IDE



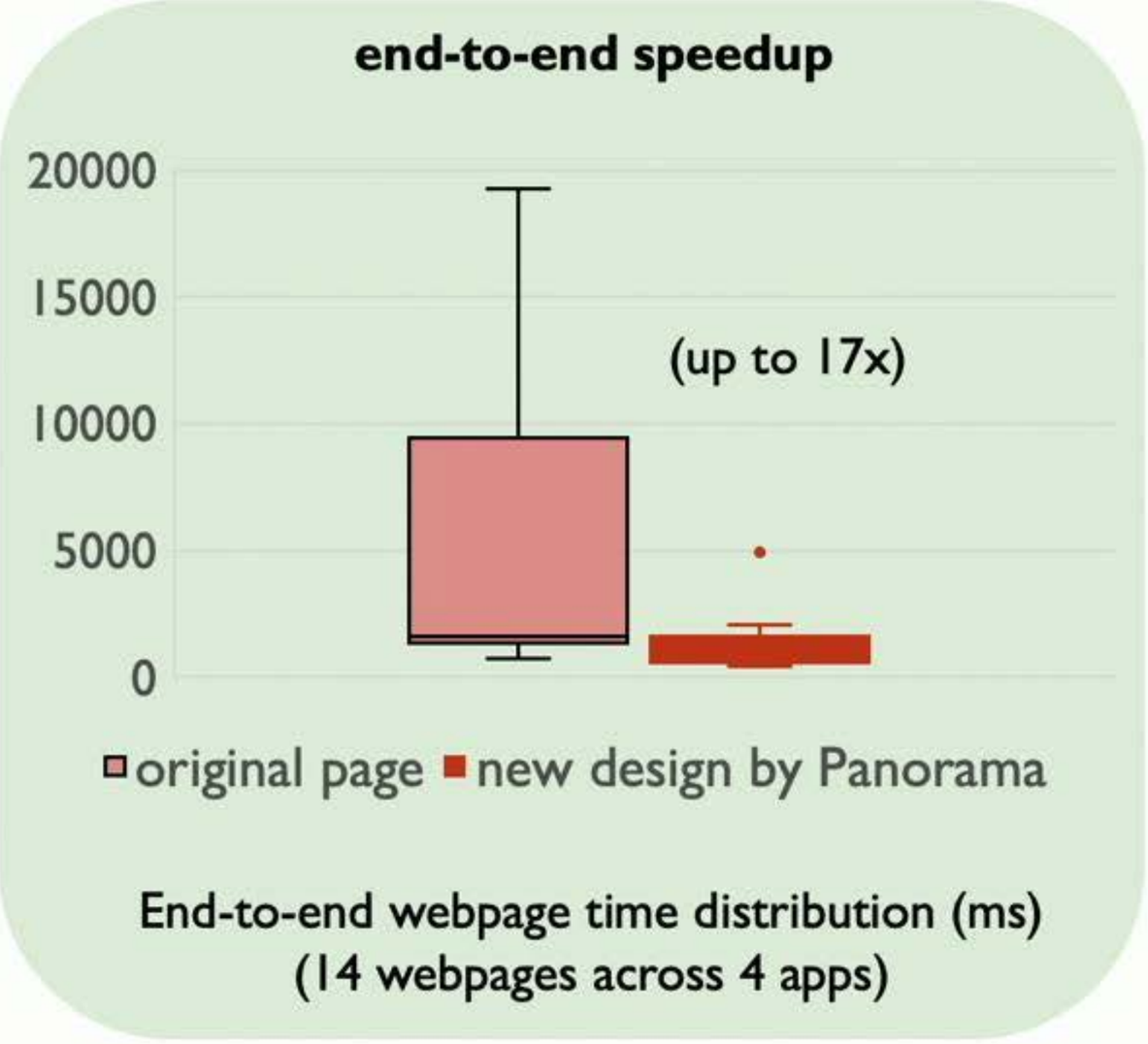




Panorama Evaluation

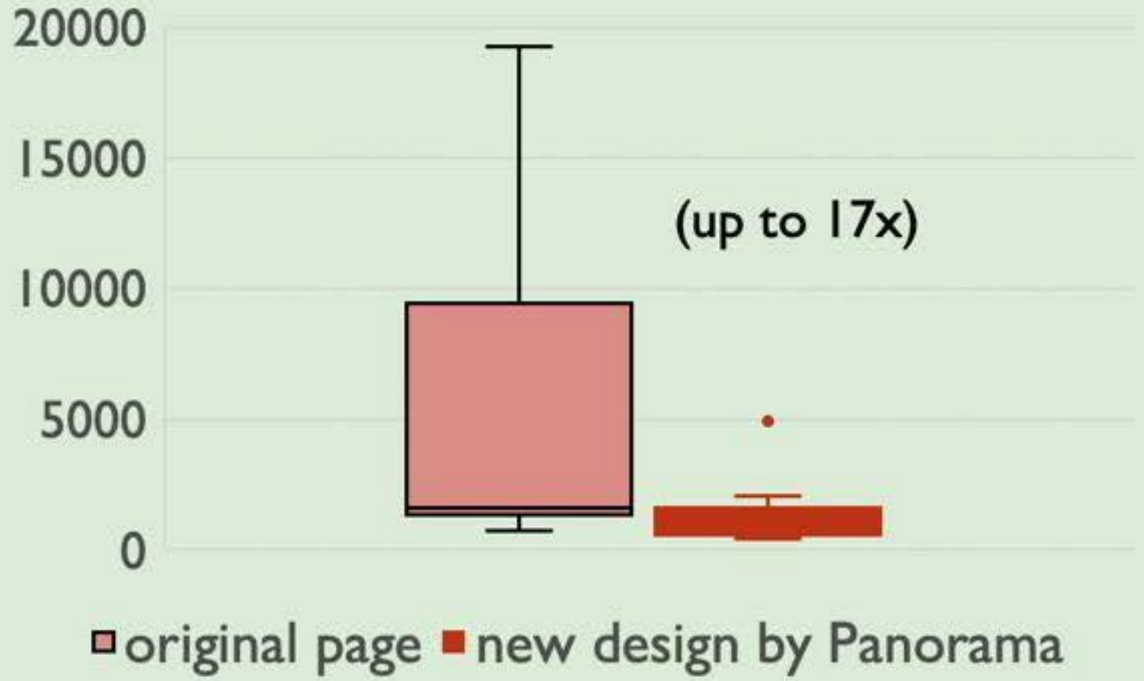


Panorama Evaluation



Panorama Evaluation

end-to-end speedup

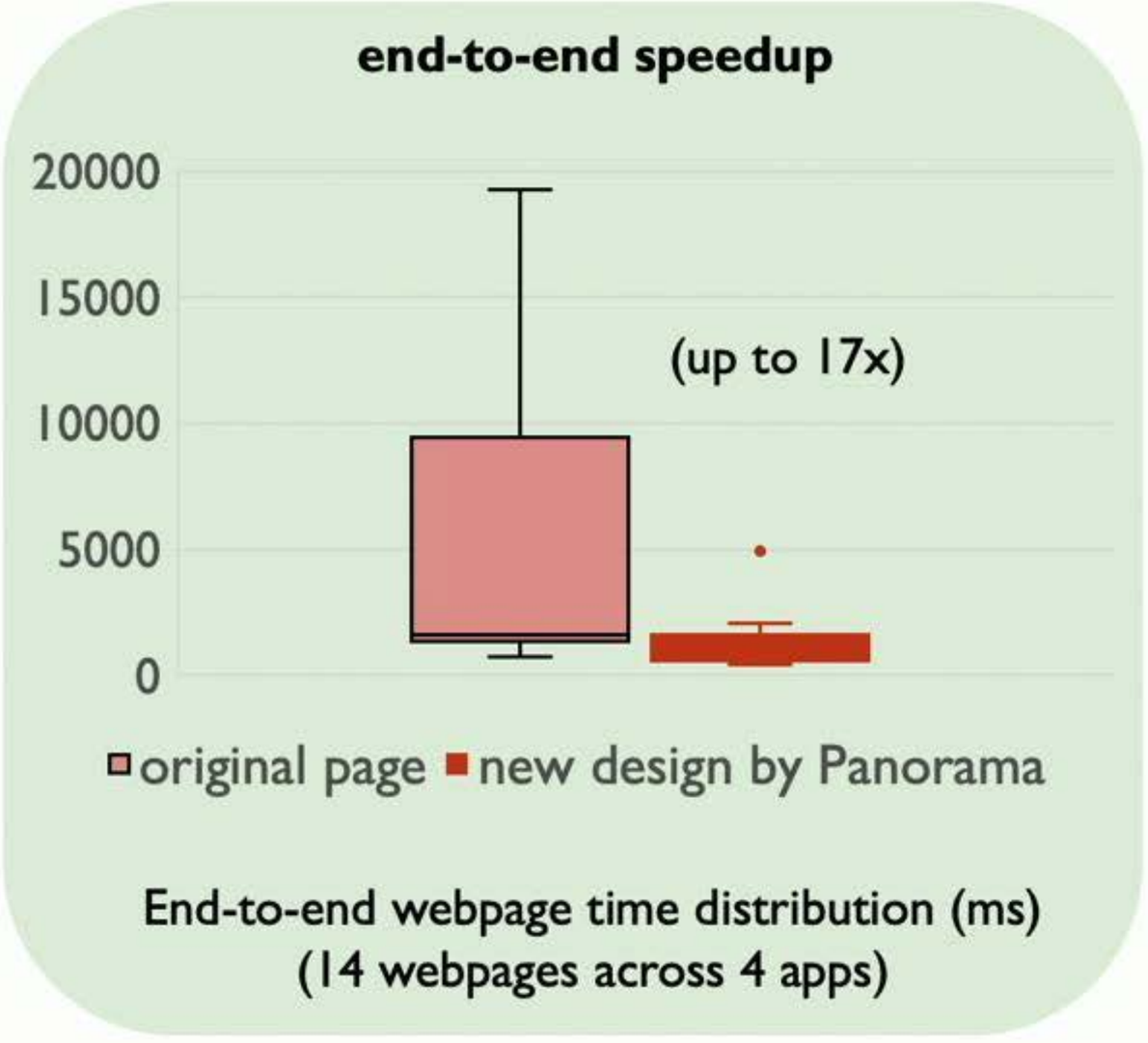


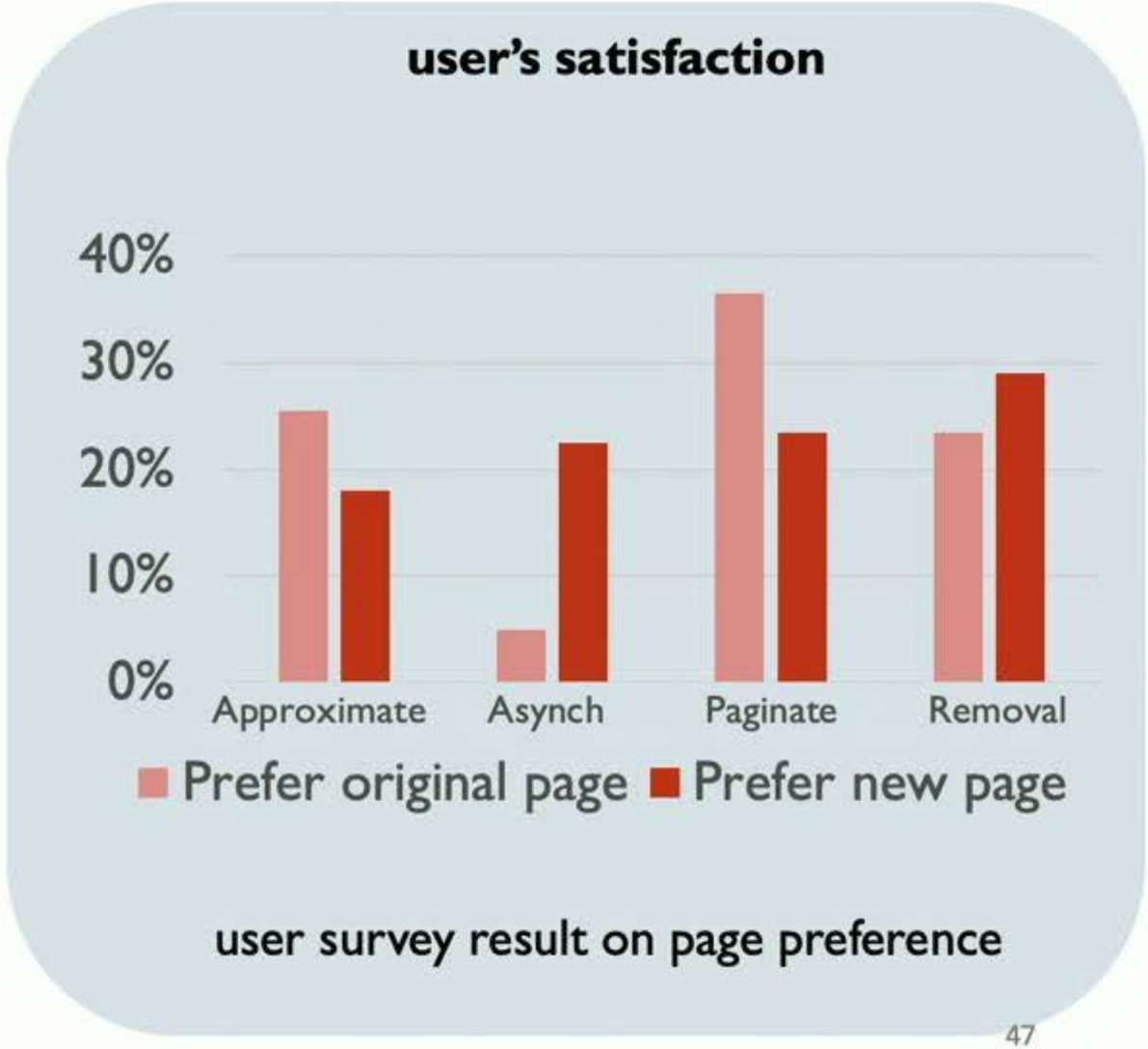
End-to-end webpage time distribution (ms)
(14 webpages across 4 apps)

user's satisfaction

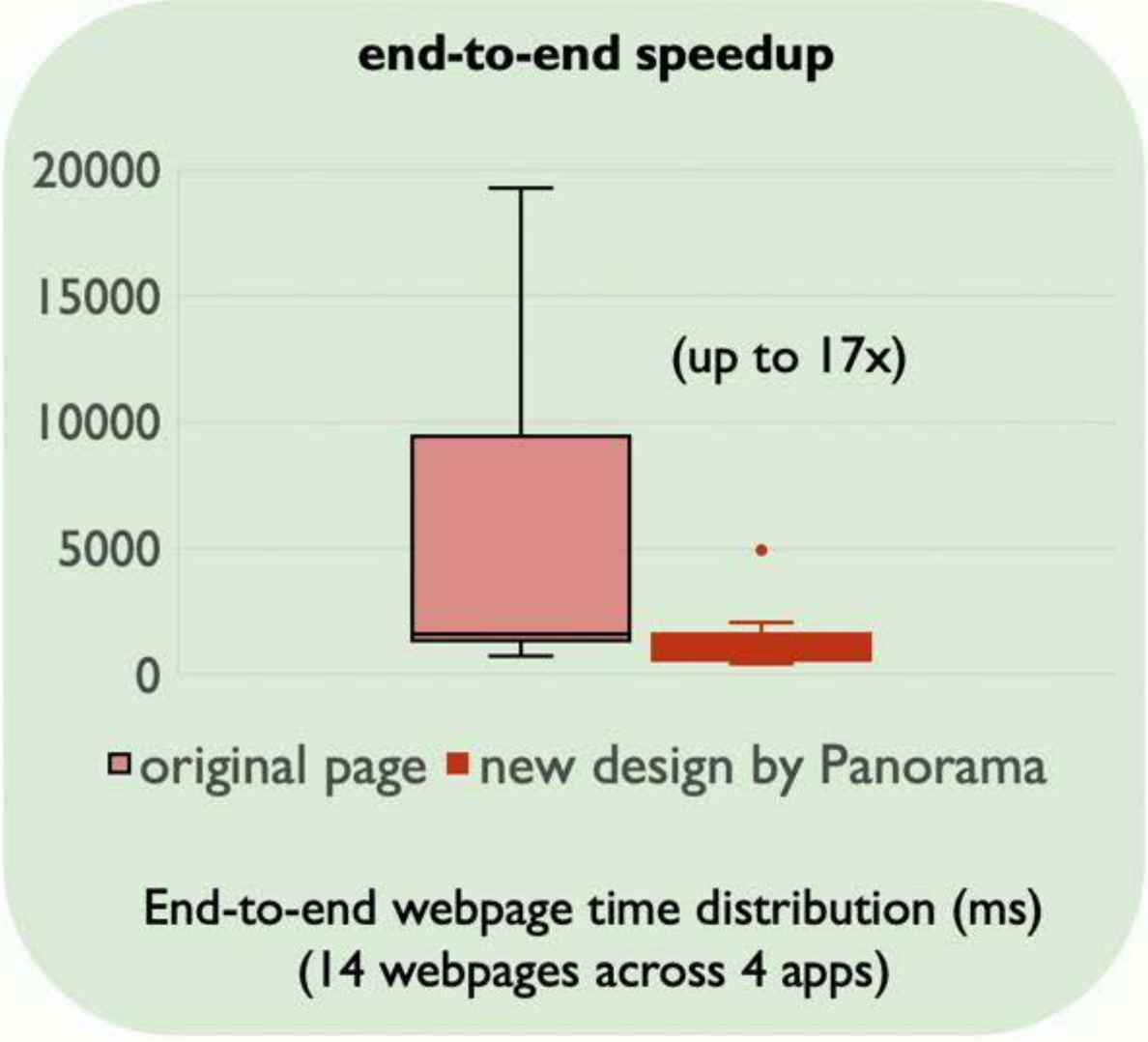
user survey result on page preference

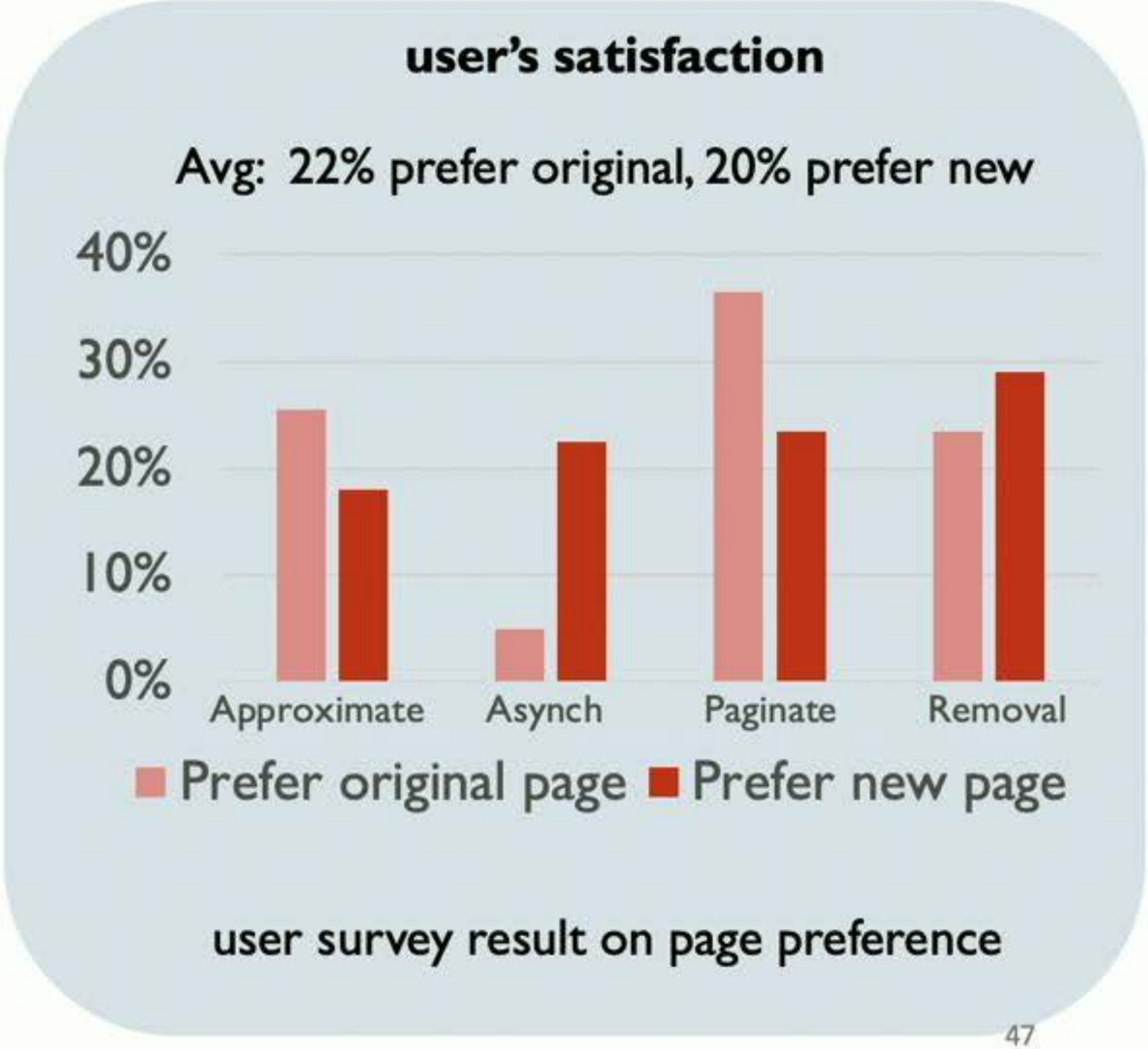
Panorama Evaluation





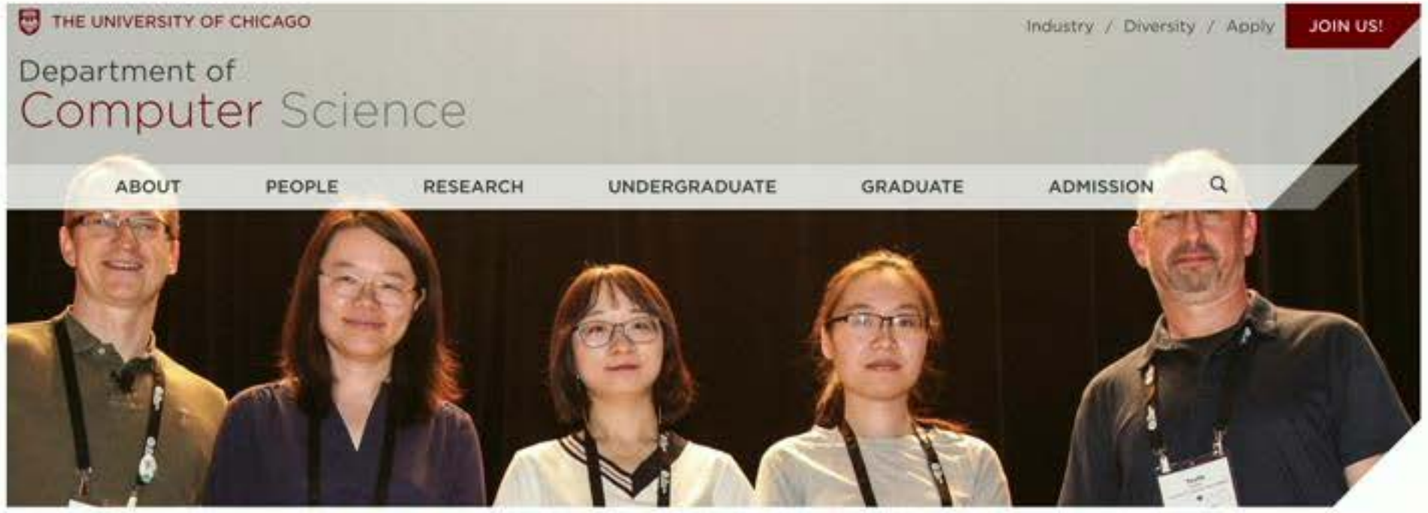
Panorama Evaluation





Press Coverage

- Uchicago news

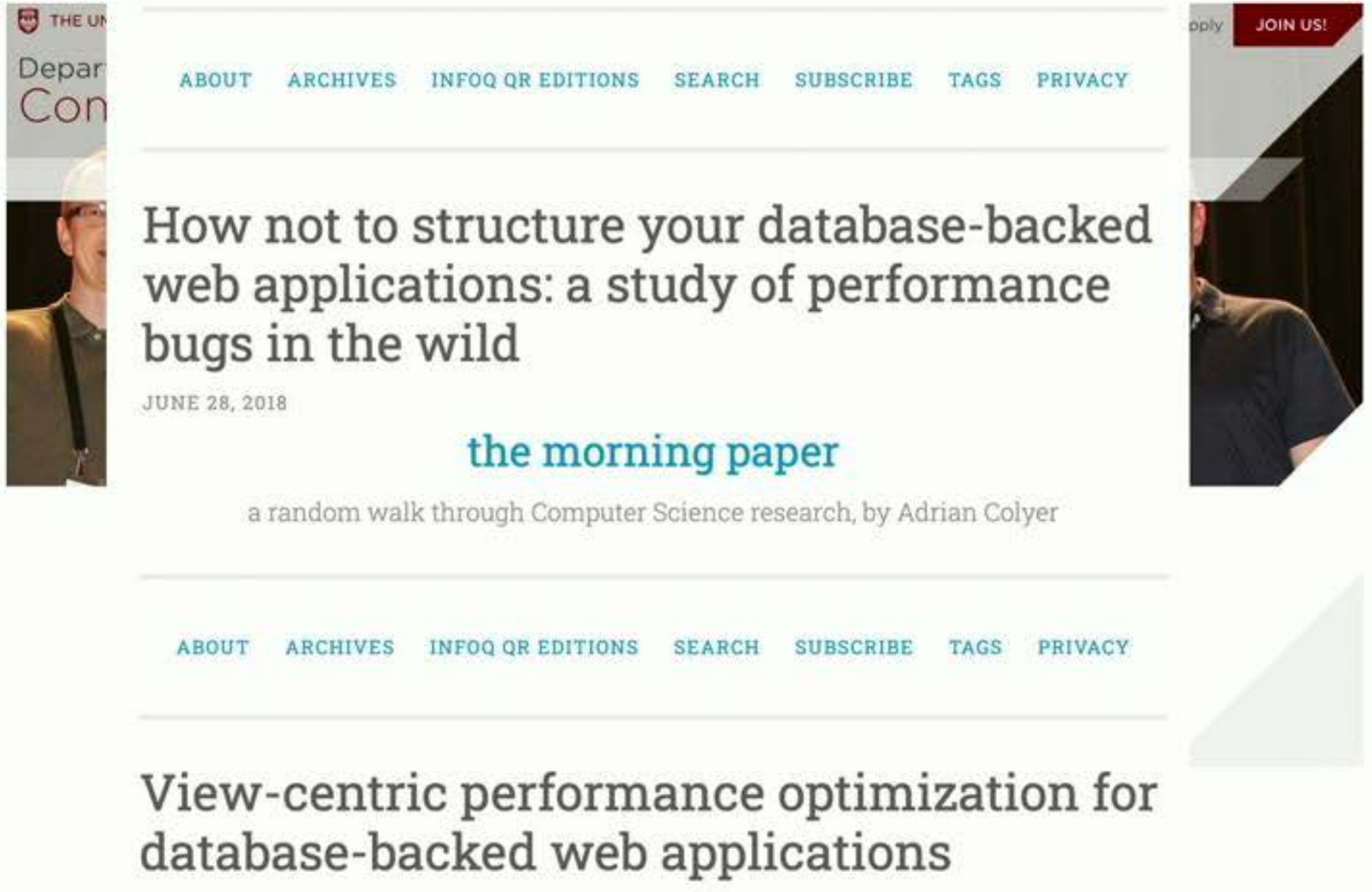


Award-Winning ICSE Paper Offers Web Developers a "Panorama" View of Slowdowns and Fixes

June 24, 2019

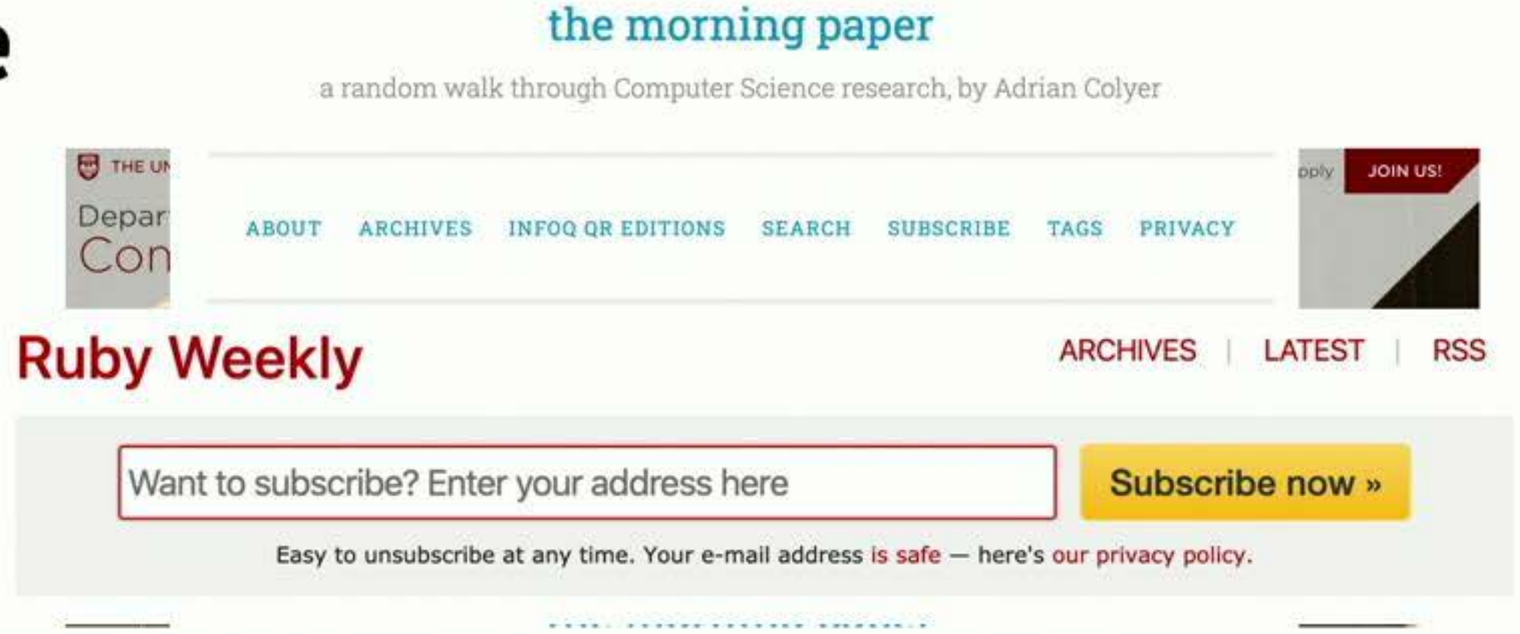
Press Coverage

- Uchicago news
- Morning Paper



Press Coverage

- Uchicago news
- Morning Paper
- Ruby Weekly



Press Coverage

- Uchicago news
- Morning Paper
- Ruby Weekly
- Hackernews

the morning paper

a random walk through Computer Science research, by Adrian Colyer

THE UN
Depar
Con

ABOUT ARCHIVES INFOQ QR EDITIONS SEARCH SUBSCRIBE TAGS PRIVACY

pply JOIN US!

Ruby Weekly

ARCHIVES | LATEST | RSS

Hacker News new | past | comments | ask | show | jobs | submit login

▲ How not to structure database-backed web apps: perfo[illegible]bugs in the wild (acolyer.org)

486 points by godelmachine on June 28, 2018 | hide | past | web | favorite | 308 comments

▲ latch on June 28, 2018 [-]

I've worked on moderately busy backend platforms (~10K-20k rps handled on a ~4 e5-2650 and aiming for 5ms 95p response times).

It greatly depends on what you're doing, but for the majority of systems which are read heavy (and that most certainly includes "dynamic" sites like Amazon or Wikipedia), I hold to two major beliefs:

1 - Have very long TTLs on your internal cache servers with a way to proactively purge (message queues) and refresh in the background. Caching shouldn't be a compromise between freshness and performance. Have both!

2 - Generate message/payloads/views asynchronously in background workers and have your synchronous path as streamlined as possible (select 1 column from 1 table with indexes filters). Avoid serialization. Precaculate and denormalize. Any personalization or truly dynamic content can be done: 1 - By having the client make separate requests for that data 2 - Merging the data into the payload with some composition. 3 - Glueing bytes together (easier/safer with protocol buffers than json)

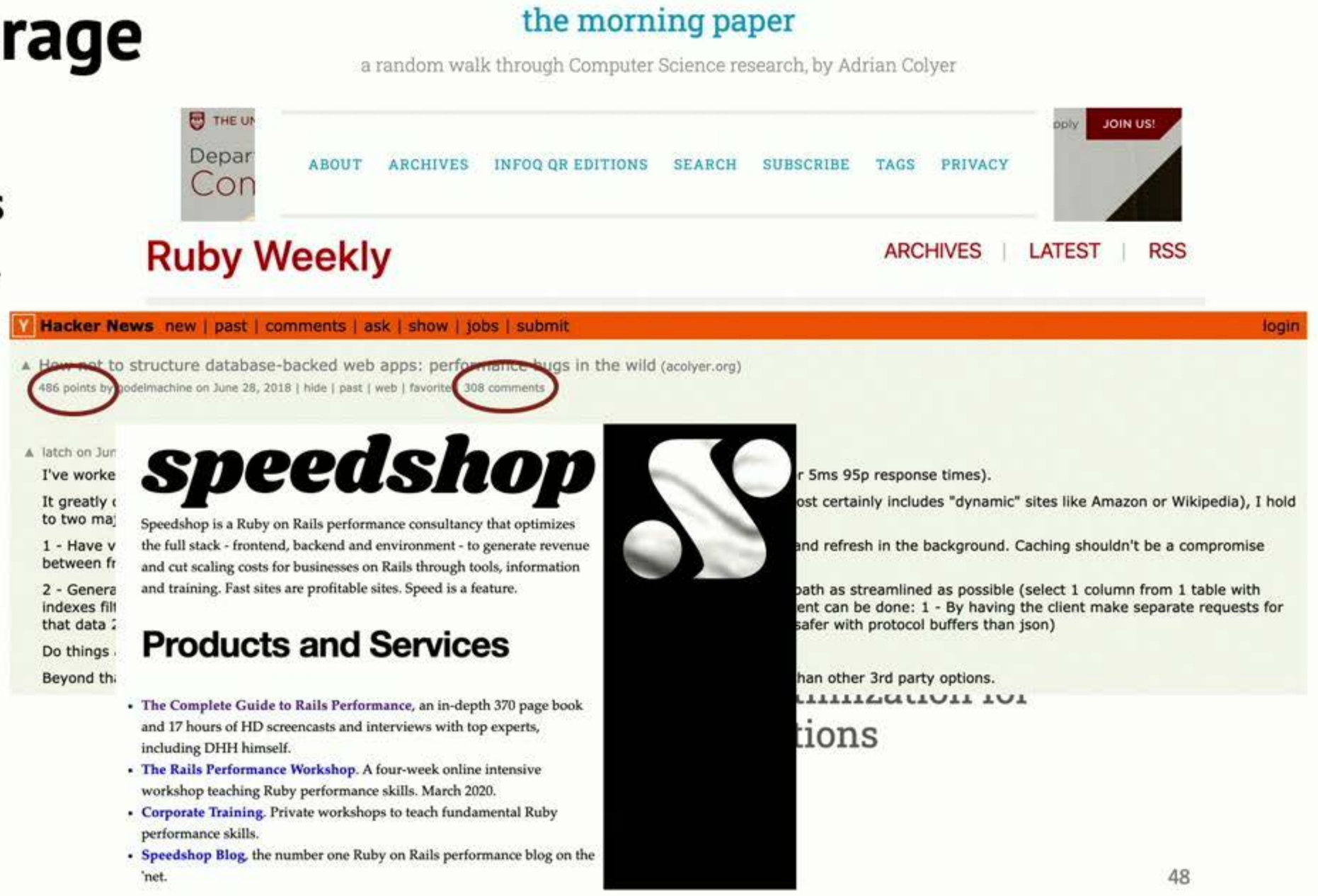
Do things asynchronously. Use message queues / streams.

Beyond that, GC becomes noticeable. For example, Go's net/http used to (might still) allocate much more than other 3rd party options.

[illegible] database-backed web applications

Press Coverage

- Uchicago news
- Morning Paper
- Ruby Weekly
- Hackernews
- Consultancy

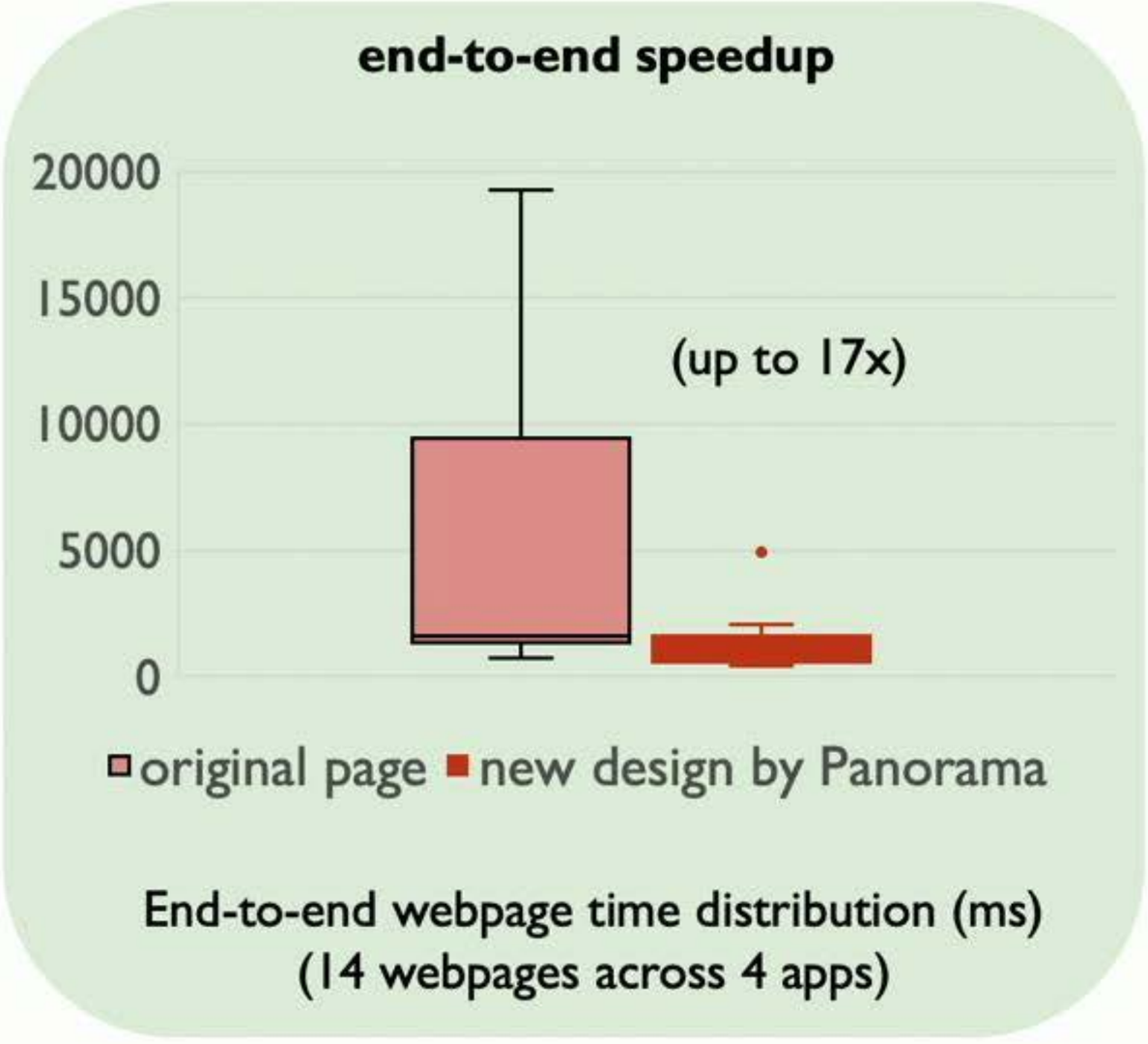


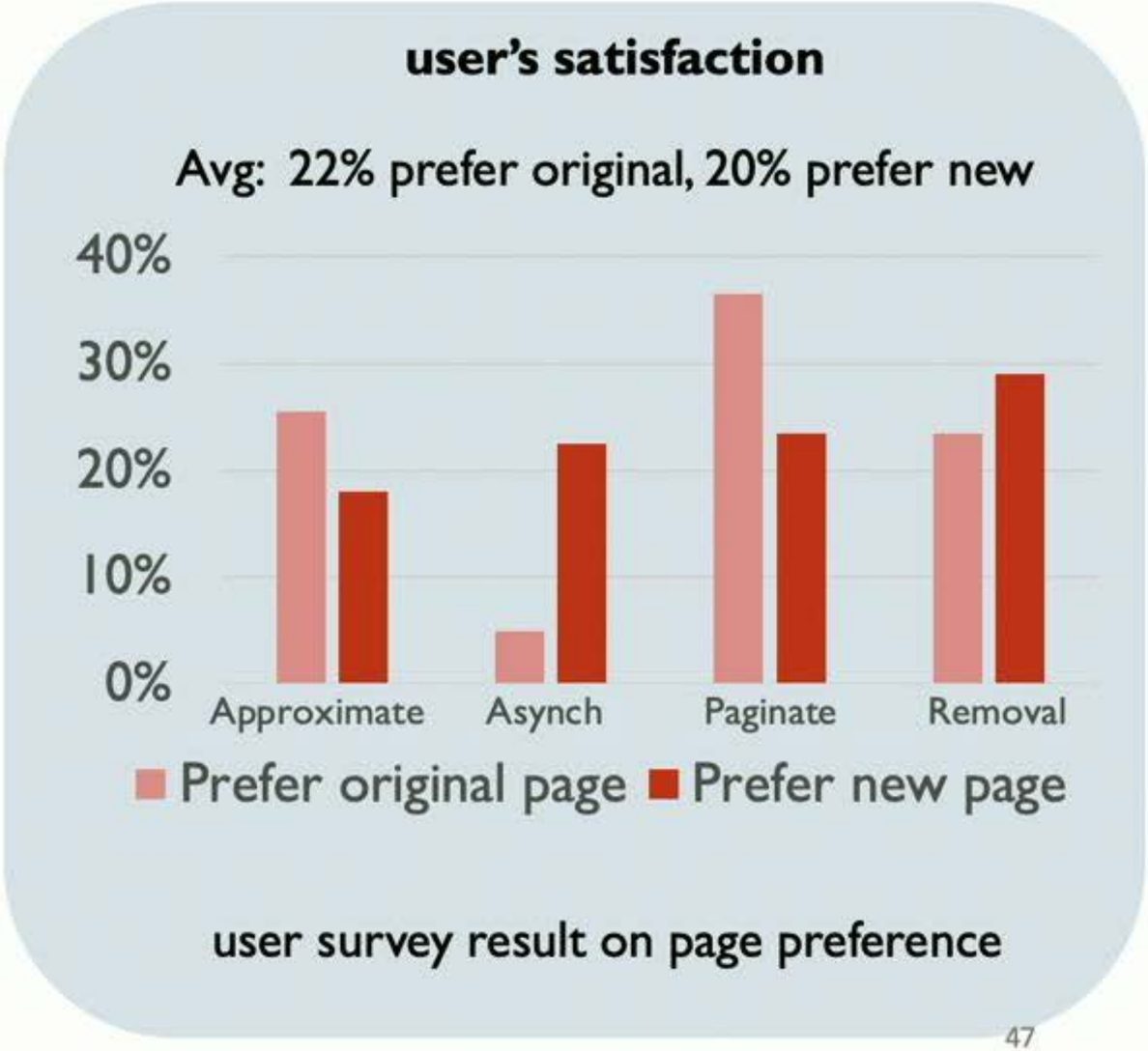
Panorama Summary

*ICSE'19 (**distinguished paper award**)*
CIDR'20

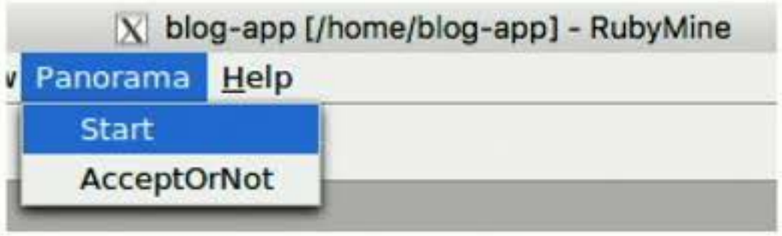
- Changing webpage design to speedup a page is a common practice, yet making design-performance tradeoff is not trivial.
- We build Panorama, a tool that suggests non-semantic-preserving changes.
- It provides an interface for developers to understand the webpage performance and tradeoff by only interact with the webpage.
- Panorama speeds up a page up to 17x, proving that the tradeoff is worthwhile.

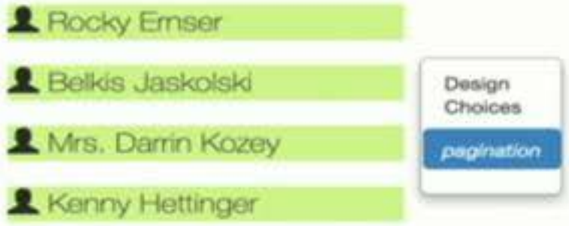
Panorama Evaluation



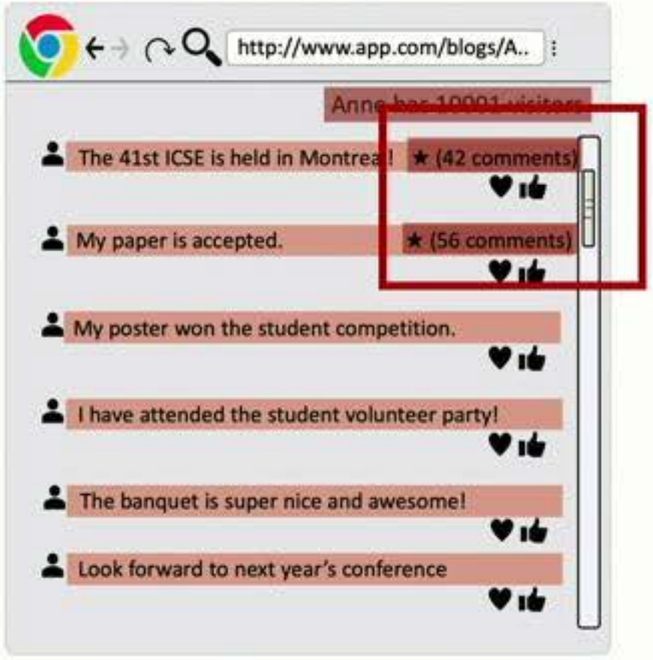


Panorama as An IDE Plugin





How does Panorama generate heatmap?



$N_{blogs} * N_{comments}$

index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

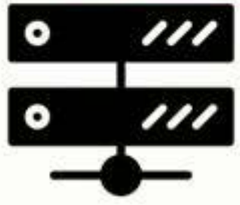
blogs_controller.rb

```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

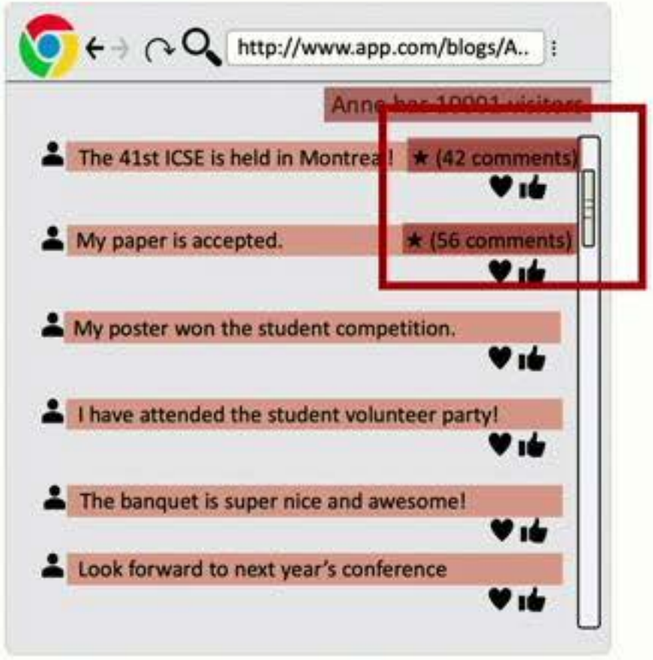
blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

```
select count(*) from
blogs join … where …
group by …
```



How does Panorama generate heatmap?



$N_{blogs} * N_{comments}$

index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

blogs_controller.rb

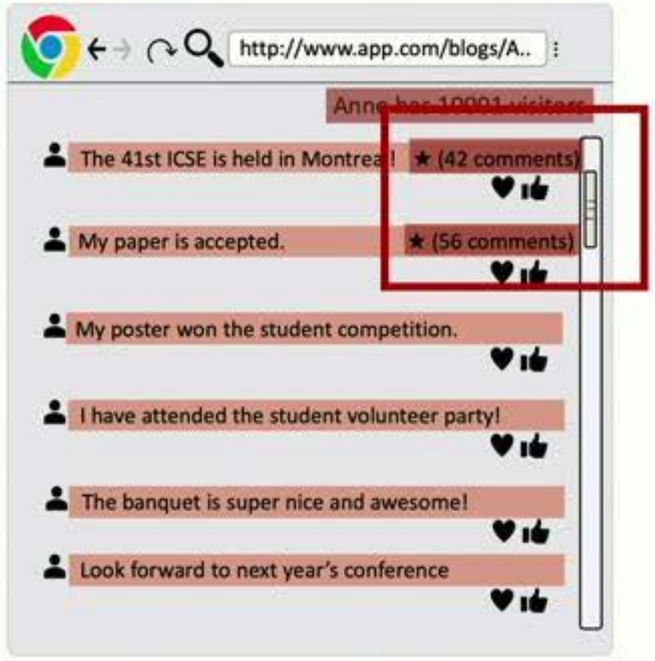
```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

```
select count(*) from
blogs join … where …
group by …
```

How does Panorama generate heatmap?



$N_{blogs} * N_{comments}$

index.html.erb

```
. . .
user.blogs.each |b|
  if @bcount[b]>100:
. . .
```

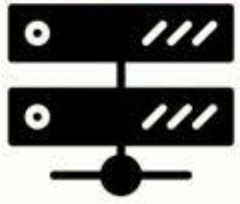
blogs_controller.rb

```
. . .
@bcount=blogs.joins(comments)
    .exclude_self(user).count
. . .
```

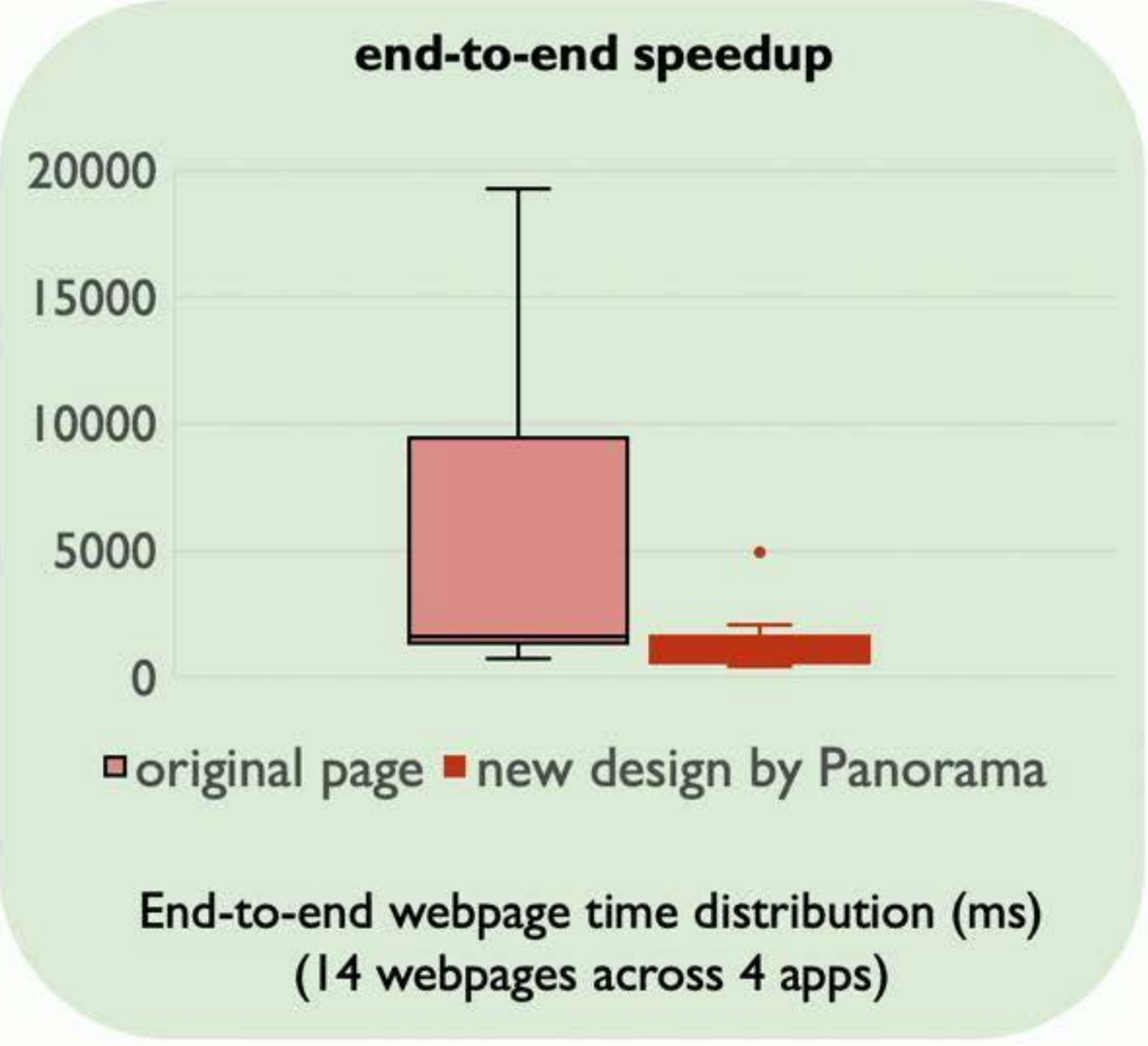
blog.rb

```
class Blog
  def exclude_self(user):
    where('comments.user_id
          !=user.id)
```

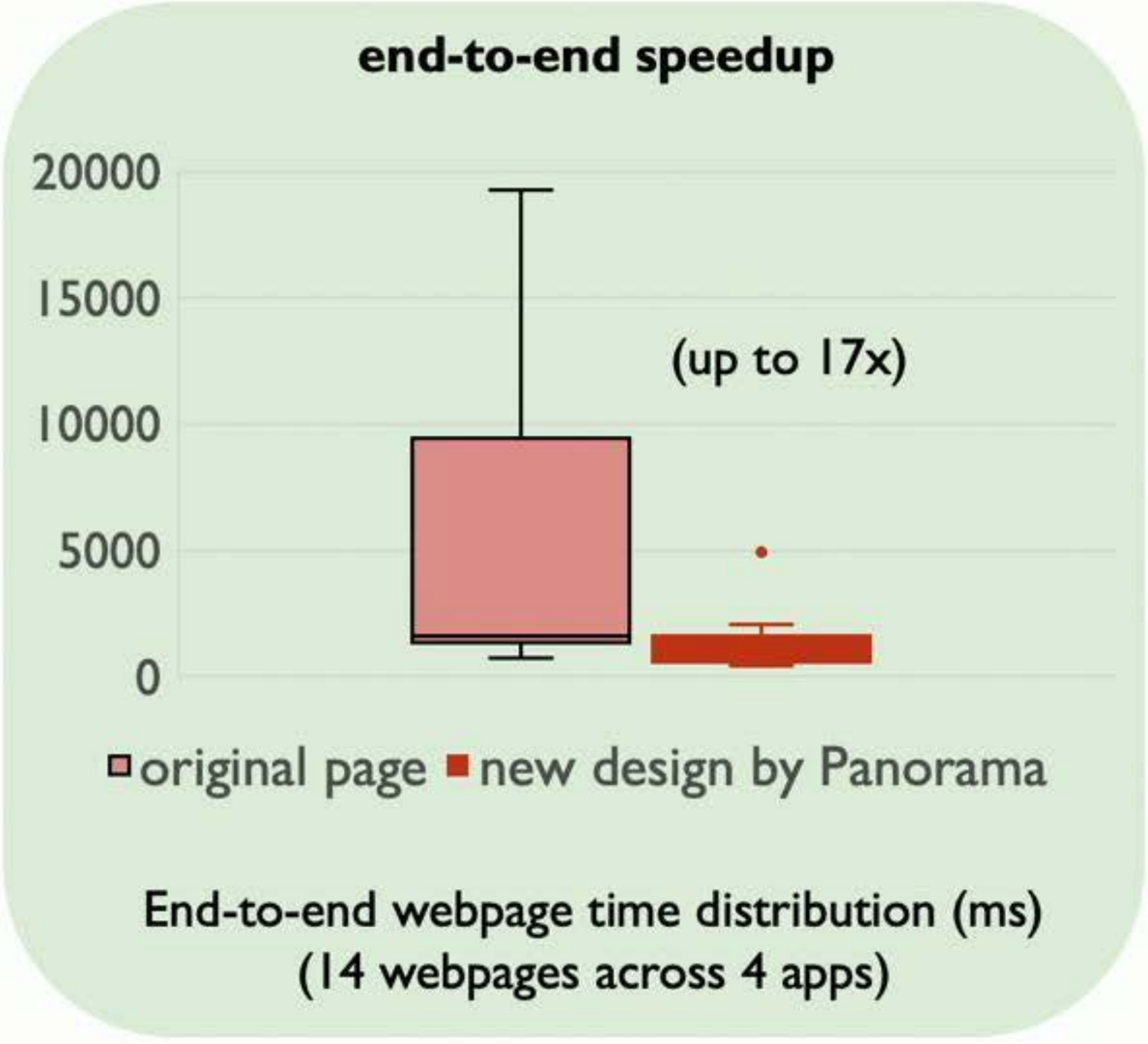
```
select count(*) from
blogs join … where …
group by …
```

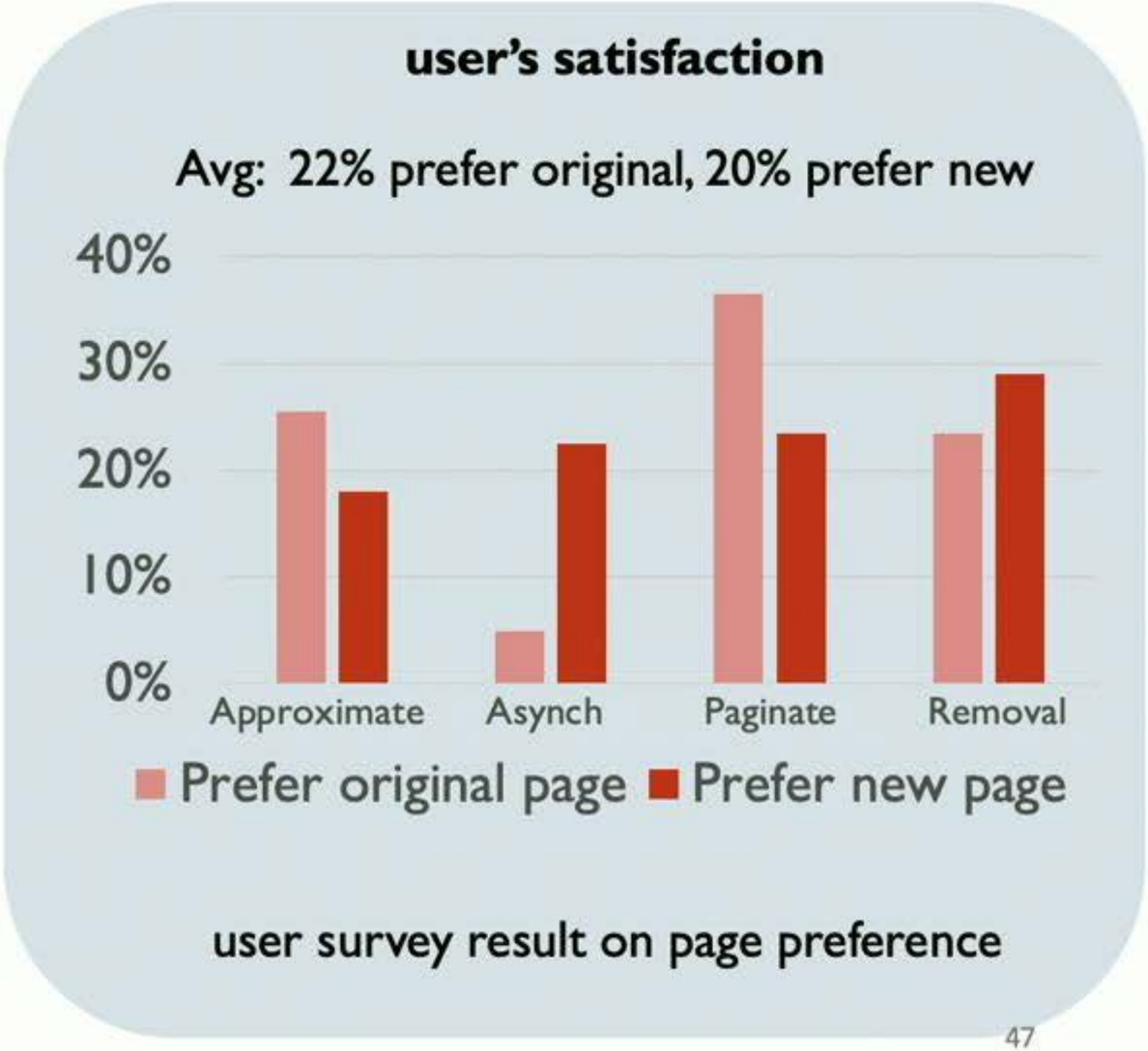


Panorama Evaluation



Panorama Evaluation



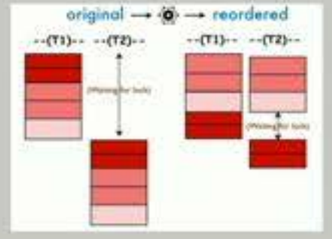


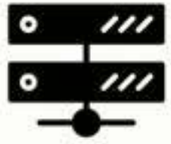
Outline

- Leveraging application semantics to optimize each layer

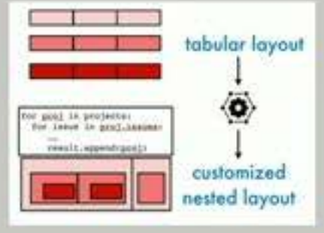
Panorama:
view-driven optimization

Quro:
reorder queries





Chestnut:
customize data layout



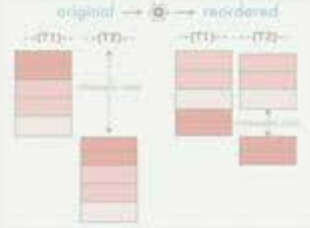
- Other projects
- Ongoing and future work

Outline

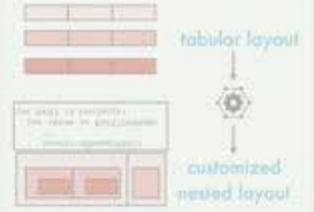
- Leveraging application semantics to optimize each layer

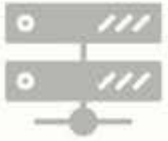
Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout





- Other projects

- Ongoing and future work

Building More Intelligent Data Preparation Systems

- Intern work at DMX (2017, 2019)
- Leveraging **open source code** to help data scientists with their work

- AutoType *[SIGMOD'18]*
  - Automatic validation of semantic data types (e.g., zipcode, credit card) using open source code
  - Find functions for 84 data types across different domains, with 90% precision

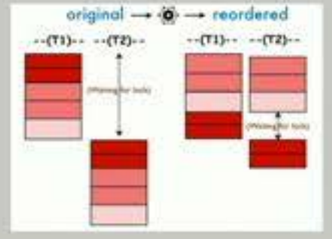
- AutoSuggest *[under submission]*
  - Learn from open source Jupyter notebooks, and recommend data operations based on data, including op parameters (e.g., which column to join/pivot) and op types
  - Achieves much higher precision (>16% higher) than existing commercial vendors

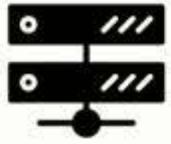
Outline

- Leveraging application semantics to optimize each layer

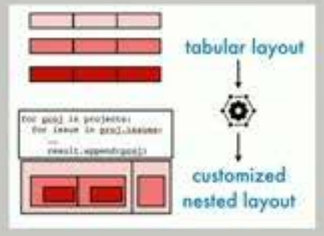
Panorama:
view-driven optimization

Quro:
reorder queries





Chestnut:
customize data layout



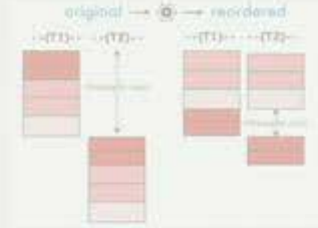
- Other projects
- Ongoing and future work

Outline

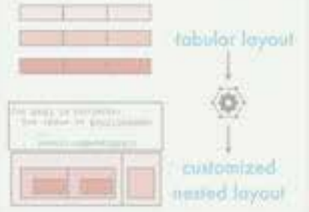
- Leveraging application semantics to optimize each layer

Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout



- Other projects
- Ongoing and future work

Ongoing Work: Maintaining Data Integrity

- Data constraint
  - Constraints among persistent data (e.g., password length, only 1 default option, return and purchase have the same order id, etc.)
  - Defined in HTML/ruby/SQL
  - Very common: 74% data fields involved in constraint, ~1.5 per field

- Constraint issue study *[ICSE'20]*
  - >100 (per app) constraints missing in DB
  - >13 (per app) conflict constraints
  - >12 (per app) constraints defined incorrectly (easily violated)

- We are building a tool to fix data constraint issues

Future Work

- More chances to leverage application semantics
  - E.g., using data constraint for query optimization
  - using client-level statistics (e.g., click rate) to help query optimization
  - …

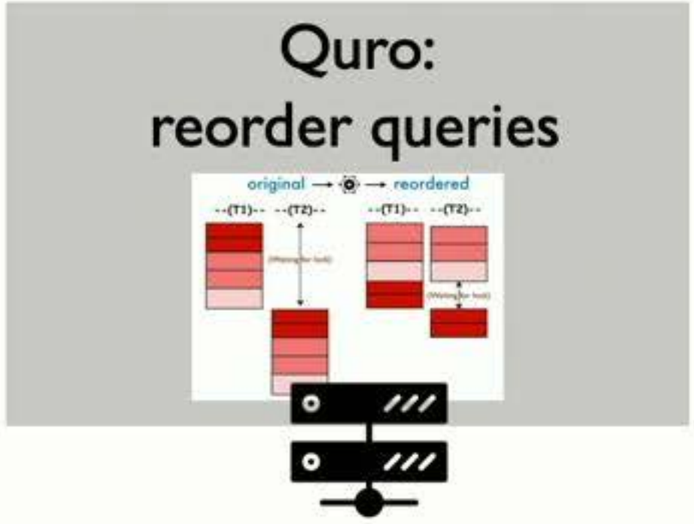
- Building a "smarter" middleware that:
  - Incorporate static/dynamic cross-stack optimizations
  - Provide analysis interface to support adding optimization

Summary

- **Leveraging application semantics** to optimize each layer

Summary

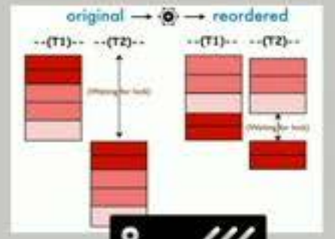
- **Leveraging application semantics** to optimize each layer



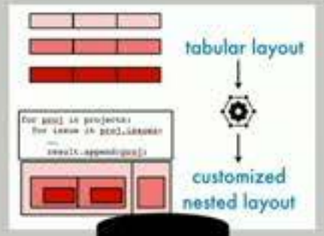
Summary

- **Leveraging application semantics** to optimize each layer

Quro:
reorder queries



Chestnut:
customize data layout

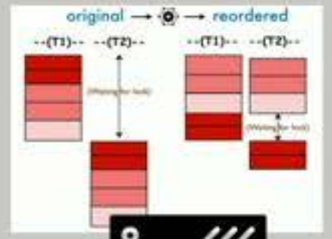


Summary

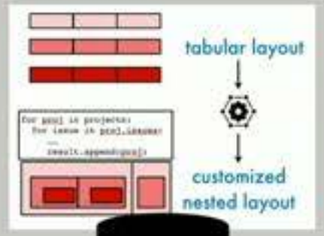
- **Leveraging application semantics** to optimize each layer

Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout

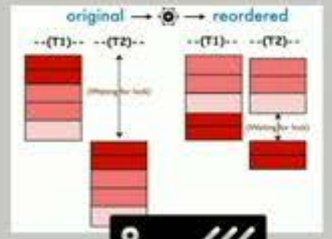


Summary

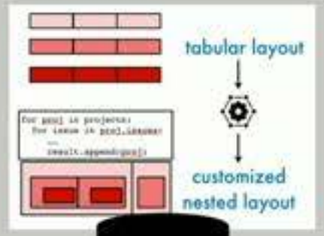
- **Leveraging application semantics** to optimize each layer

Panorama:
view-driven optimization

Quro:
reorder queries



Chestnut:
customize data layout



- Other projects
  - Building more intelligent data preparation systems
- Ongoing and future work
  - Maintaining data integrity (fix data constraint bugs)
  - More chances to leverage application semantics; building a "smarter" middleware