

Understanding and Improving Database-backed Applications

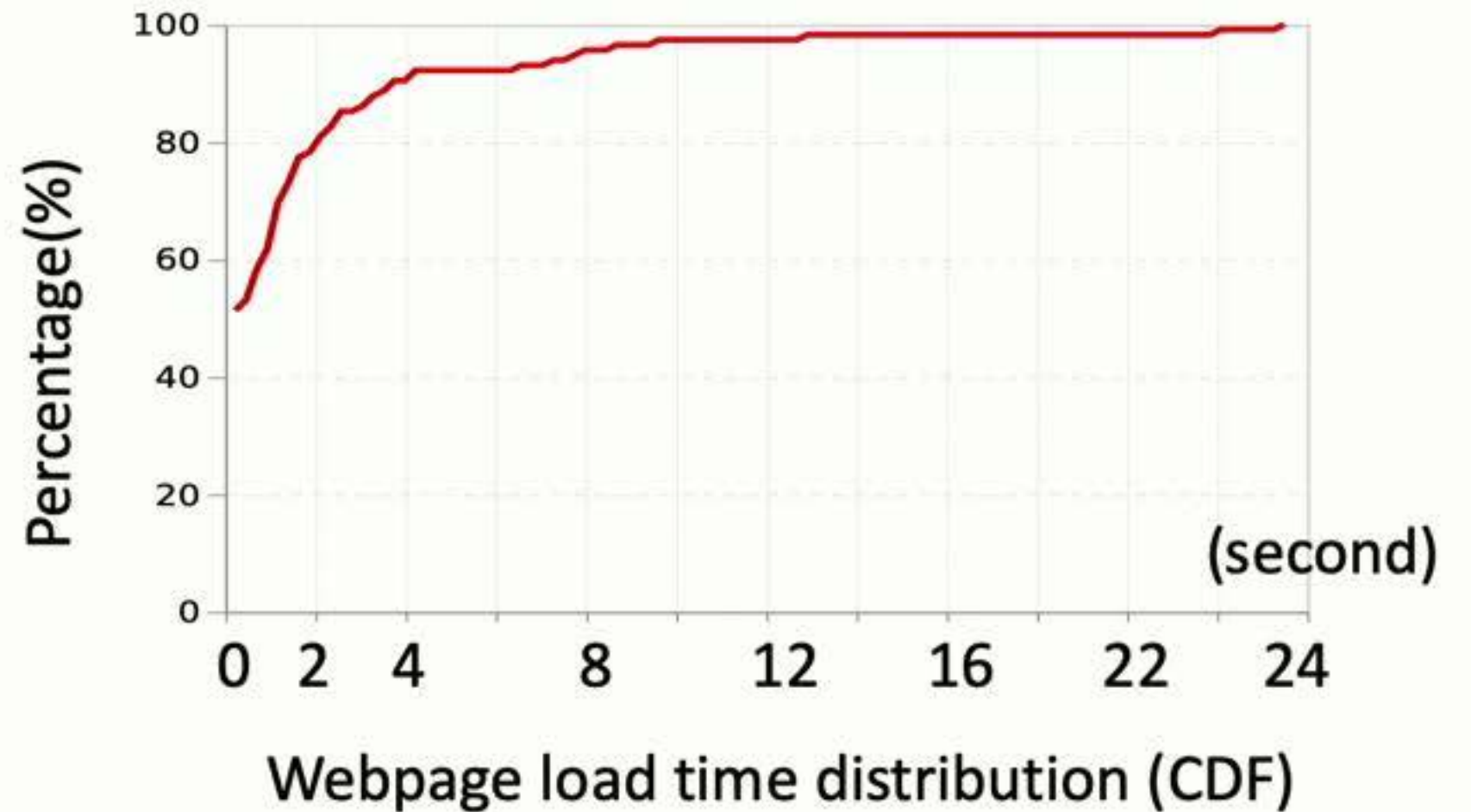
Cong Yan

University of Washington

Real-World Application Performance

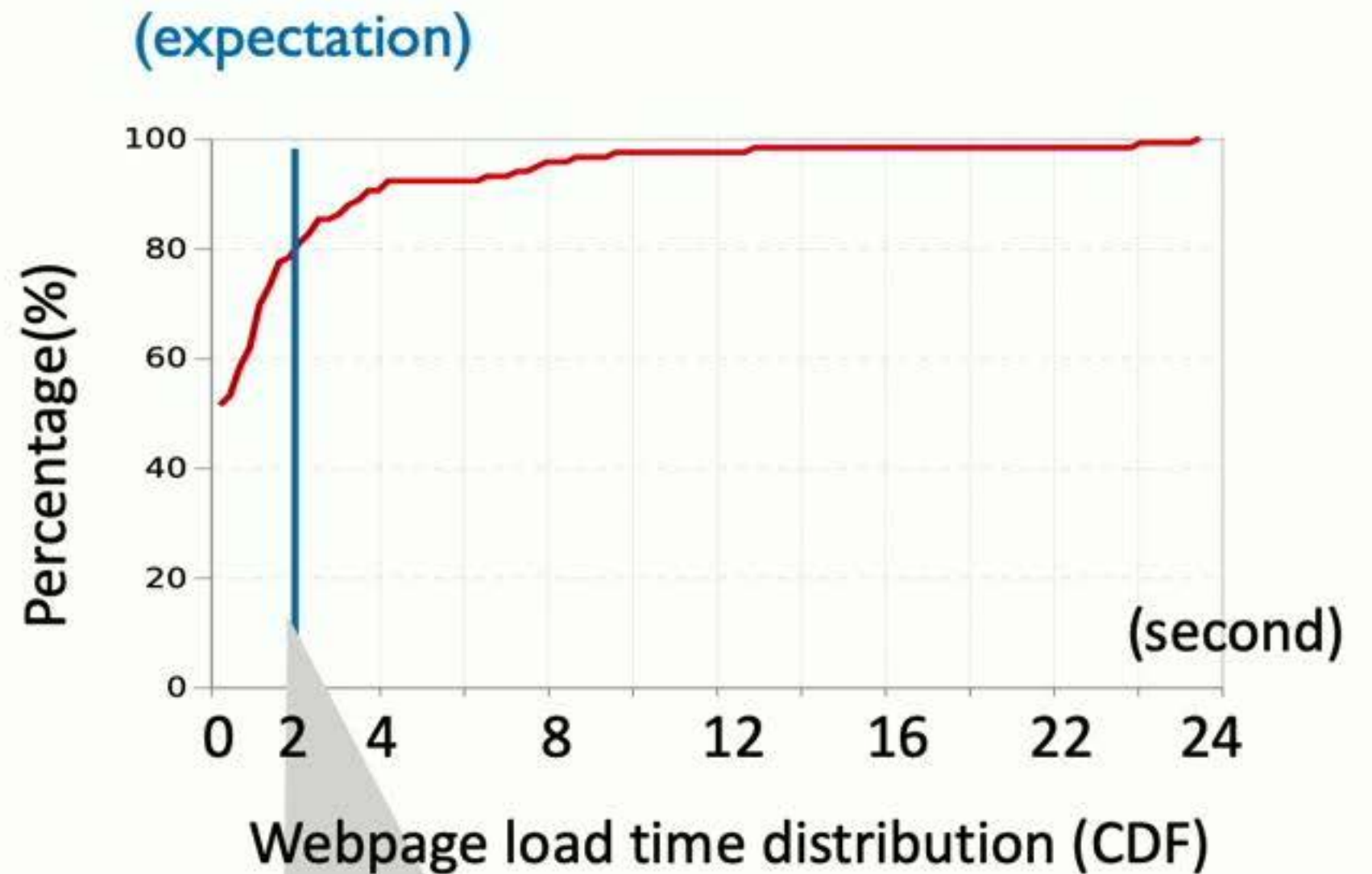
Real-World Application Performance

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

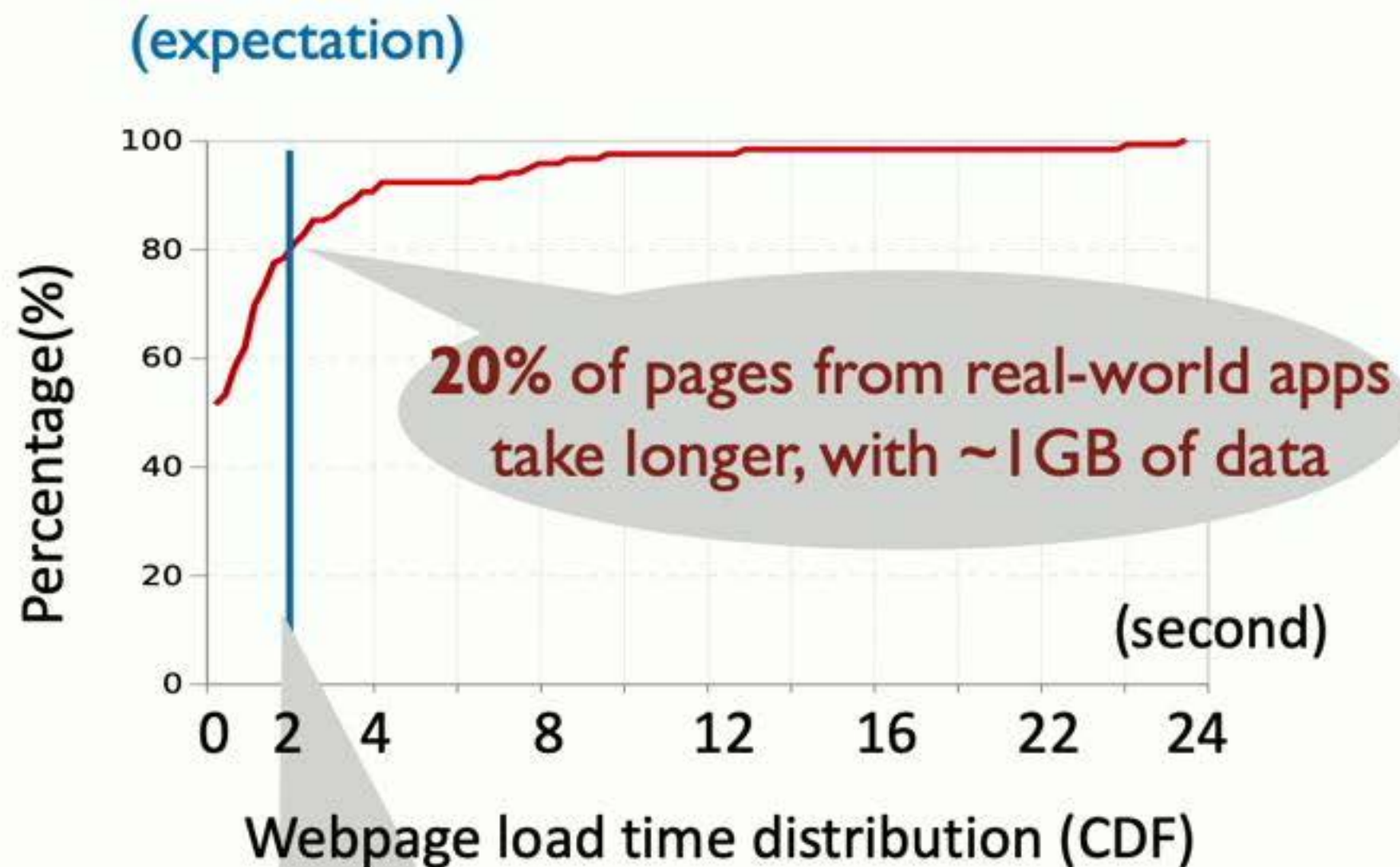
Application	# github stars
Discourse (forum)	22k
Lobsters (forum)	2.4k
Gitlab (collaboration)	49k
Redmine (collaboration)	3.6k
Spree (E-commerce)	17k
ROR Ecommerce	1.7k
Fulcrum (task mgmt)	697
Tracks (task mgmt)	3.5k
Diaspora (social network)	18k
Onebody (social network)	1.2k
Openstreetmap (map)	8k
Fallingfruit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



half of the users expect a page to load in less than 2 seconds

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

...



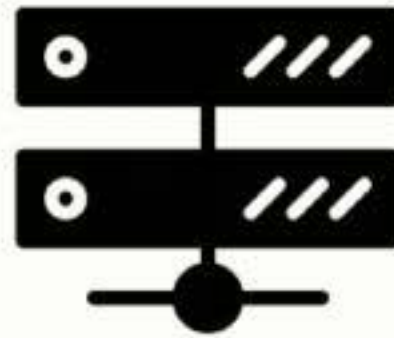
request
→
←
webpage

Dead code elimination

Vectorization

Object caching

...



query
→
←
data

Query optimization

Physical design

Concurrency control

...



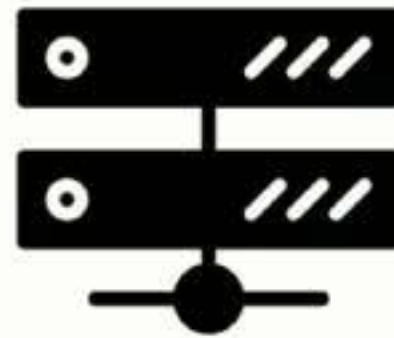
Key Insight: Leveraging Application Semantics

Leveraging Application Semantics To Optimize Each Layer

★ Leverage how query results are used in the application



request
→
←
webpage



query
→
←
data

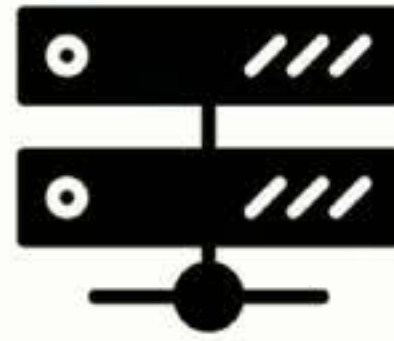


Leveraging Application Semantics To Optimize Each Layer

★ Leverage how query results are used in the application



request
→
←
webpage



query
→
←
data



Leveraging Application Semantics To Optimize Each Layer

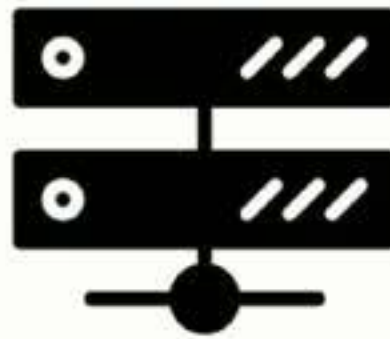
★ Leverage how query results are used in the application



request
→
←
webpage



prod=exec(Q1)



query
→
←
data



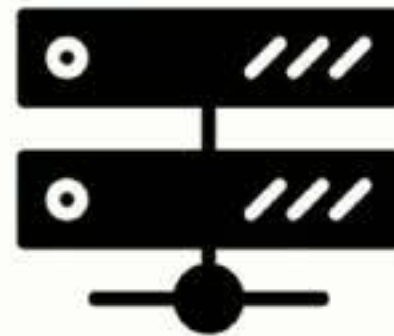
Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



request
→
←
webpage



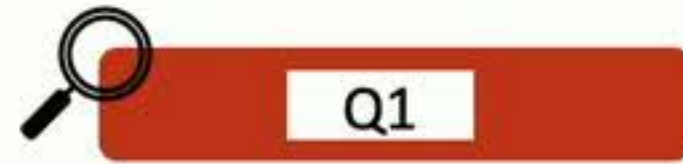
query
→
←
data



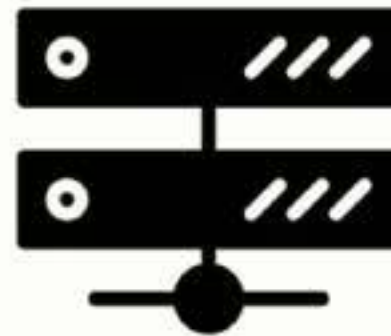
Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



request
⇌
webpage



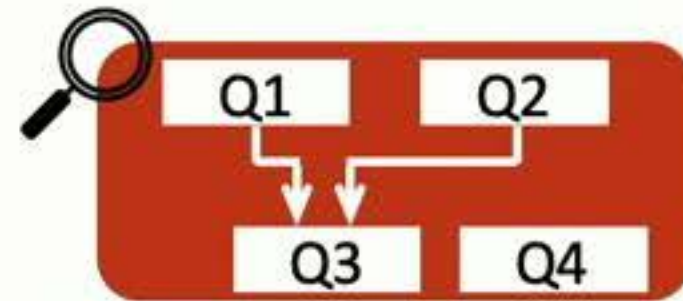
query
⇌
data



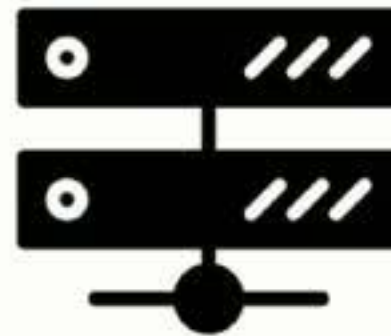
Leveraging Application Semantics To Optimize Each Layer

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



request
→
←
webpage



query
→
←
data



Leveraging Application Semantics To Optimize Each Layer

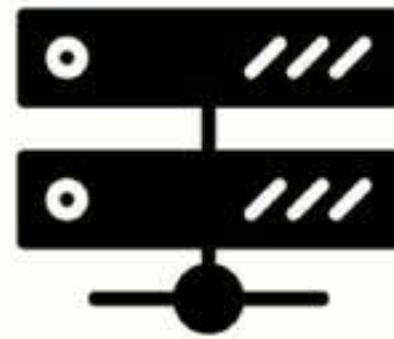
★ Leverage the data retrieval behind each webpage data

★ Leverage how queries are connected in the application

★ Leverage how query results are used in the application



request
⇌
webpage

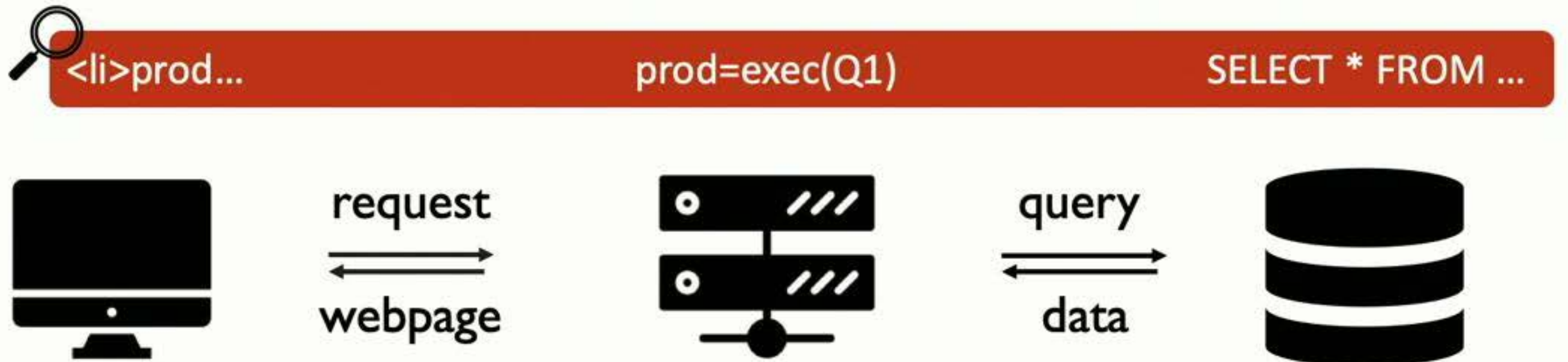


query
⇌
data



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

★ Leverage the data retrieval behind each webpage data

Quero:
reorder queries

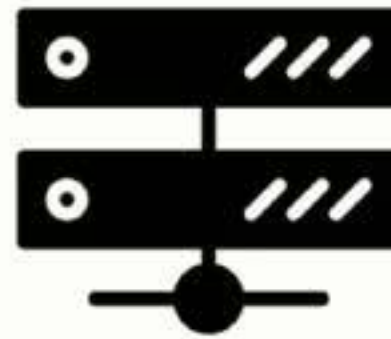
★ Leverage how queries are connected in the application

Chestnut:
customize data layout

★ Leverage how query results are used in the application



request
→
←
webpage



query
→
←
data



Outline

- Leveraging application semantics to optimize each layer

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

[VLDB'16]

[VLDB'19]

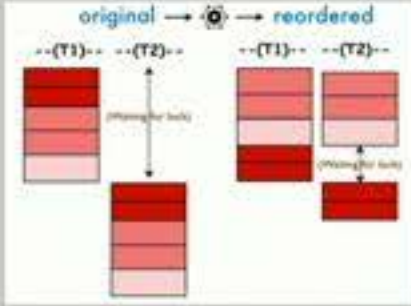
Panorama:
view-driven optimization



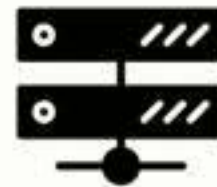
The diagram shows a 'slow webpage' on the left and a 'fast webpage' on the right. A central gear icon represents the optimization process. The fast webpage is a more compact version of the slow one, with elements rearranged for better performance.



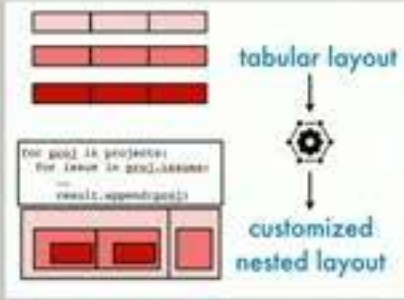
Quro:
reorder queries



The diagram illustrates the reordering of queries. It shows an 'original' query plan with two parallel paths labeled (T1) and (T2). A gear icon indicates the reordering process. The 'reordered' plan shows the paths interleaved, which can improve performance by better utilizing system resources.



Chestnut:
customize data layout



The diagram shows a 'tabular layout' on the left and a 'customized nested layout' on the right. A gear icon represents the customization process. The customized layout is a more efficient representation of the data, tailored to the specific application's needs.



Outline

- Leveraging application semantics to optimize each layer

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

Panorama:
view-driven optimization

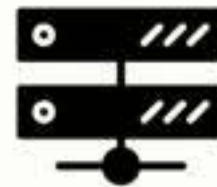
The diagram shows a 'slow webpage' on the left and a 'fast webpage' on the right. A gear icon with a checkmark is positioned between them, indicating an optimization process. The fast webpage is a more compact and readable version of the slow one.



[VLDB'16]

Quoro:
reorder queries

The diagram illustrates the reordering of a query plan. It shows an 'original' query plan with two parallel paths, (T1) and (T2), leading to a join. The 'reordered' plan shows the paths swapped, with (T2) leading to the join first, followed by (T1). This is done to optimize the execution order based on data distribution.



[VLDB'19]

Chestnut:
customize data layout

The diagram shows a 'tabular layout' of a query result being transformed into a 'customized nested layout'. The query is:

```
for (gob) in projects;
for (team in gob.teams)
  result.append(gob);
```

 The tabular layout is a flat list of rows. The customized nested layout is a tree structure where each team's projects are grouped together, reflecting the nested structure of the query.



- Other projects
- Ongoing and future work

Outline

- Leveraging application semantics to optimize each layer



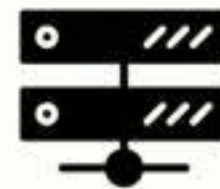
Panorama:
view-driven optimization

The diagram shows a browser window on the left with a list of items. On the right, a search bar is shown with a magnifying glass icon. Below the search bar, a gear icon represents settings. Two arrows point from the search bar area to the browser window: one labeled 'slow webpage' pointing to a slower-loading state, and another labeled 'fast webpage' pointing to a faster-loading state, illustrating how search and settings can optimize the view.



Quoro:
reorder queries

The diagram illustrates query reordering. It shows two sets of red blocks representing data blocks. The left set is labeled 'original' and has two columns of blocks, with the first column containing blocks labeled '(T1)' and '(T2)'. The right set is labeled 'reordered' and has two columns, with the first column containing blocks labeled '(T2)' and '(T1)'. Arrows indicate the movement of blocks between the two sets, showing that the order of blocks is changed to optimize query execution.



Chestnut:
customize data layout

The diagram shows a SQL query:

```
for gobj in projects;
for item in gobj.items;
-- result.append(gobj);
```

 Above the query, there are three rows of red blocks representing a 'tabular layout'. Below the query, there are three rows of red blocks representing a 'customized nested layout'. A gear icon is positioned between the two layouts, with arrows pointing from the tabular layout to the gear and from the gear to the customized nested layout, indicating a transformation or optimization process.



- Other projects
- Ongoing and future work

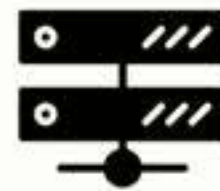
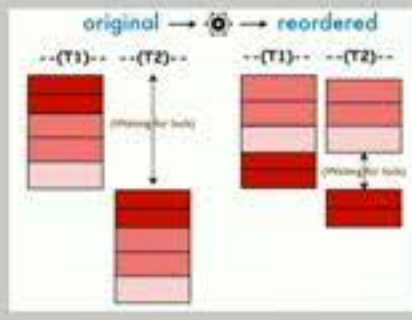
Outline

- 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

Execution
time

--T1-- original --T2--



Query Reorder under 2PL

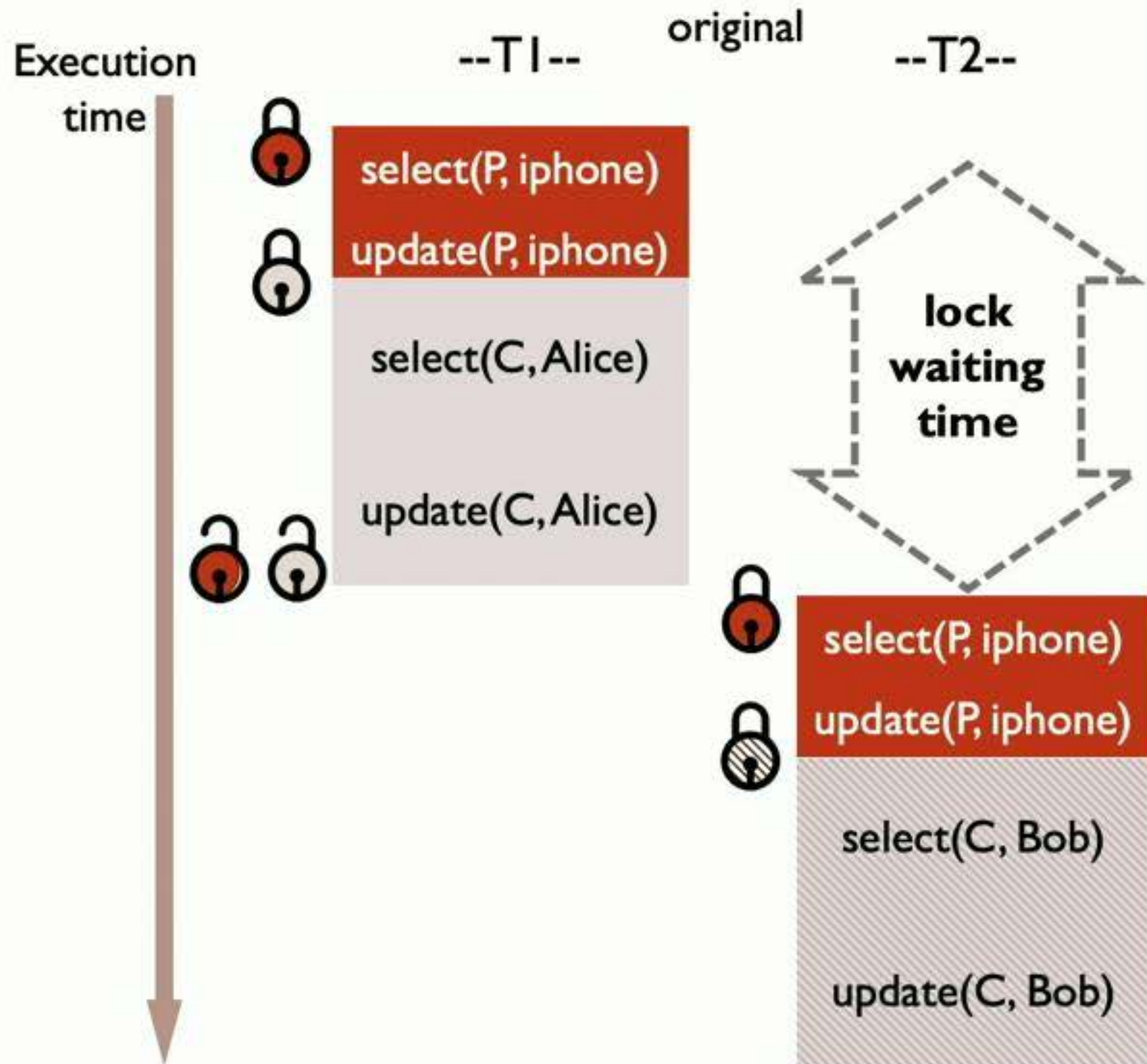
Execution
time



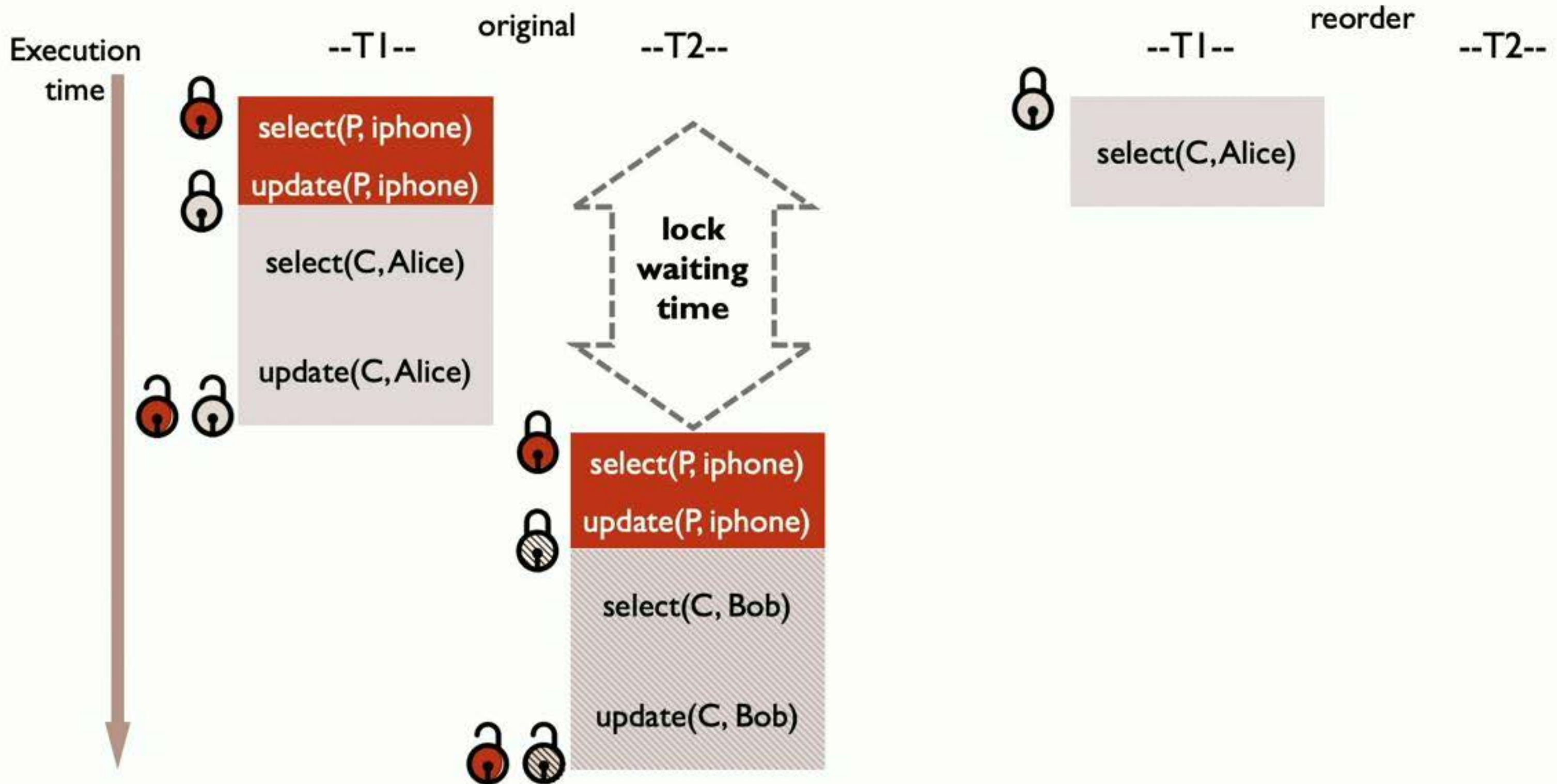
select(P, iphone)

--T1-- original --T2--

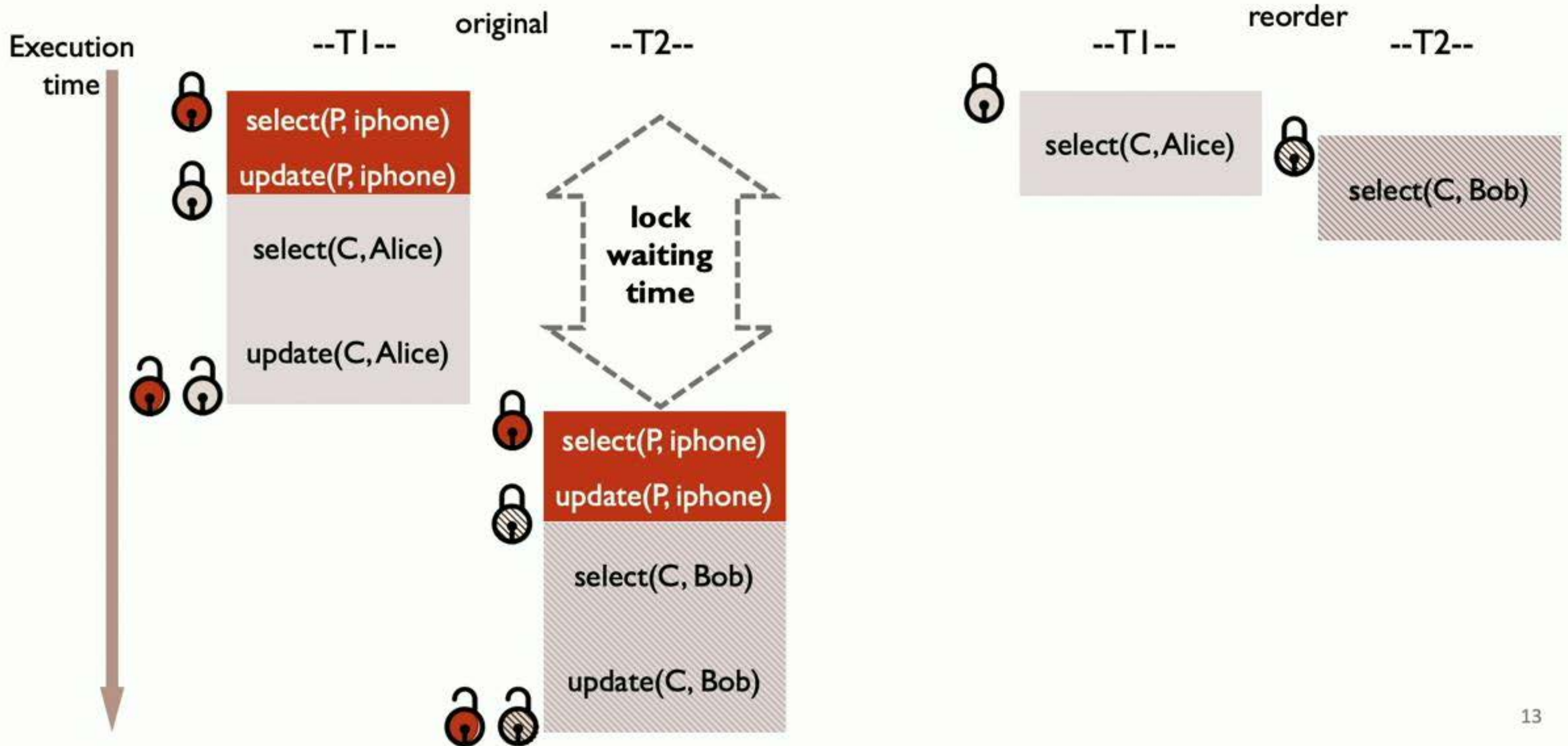
Query Reorder under 2PL



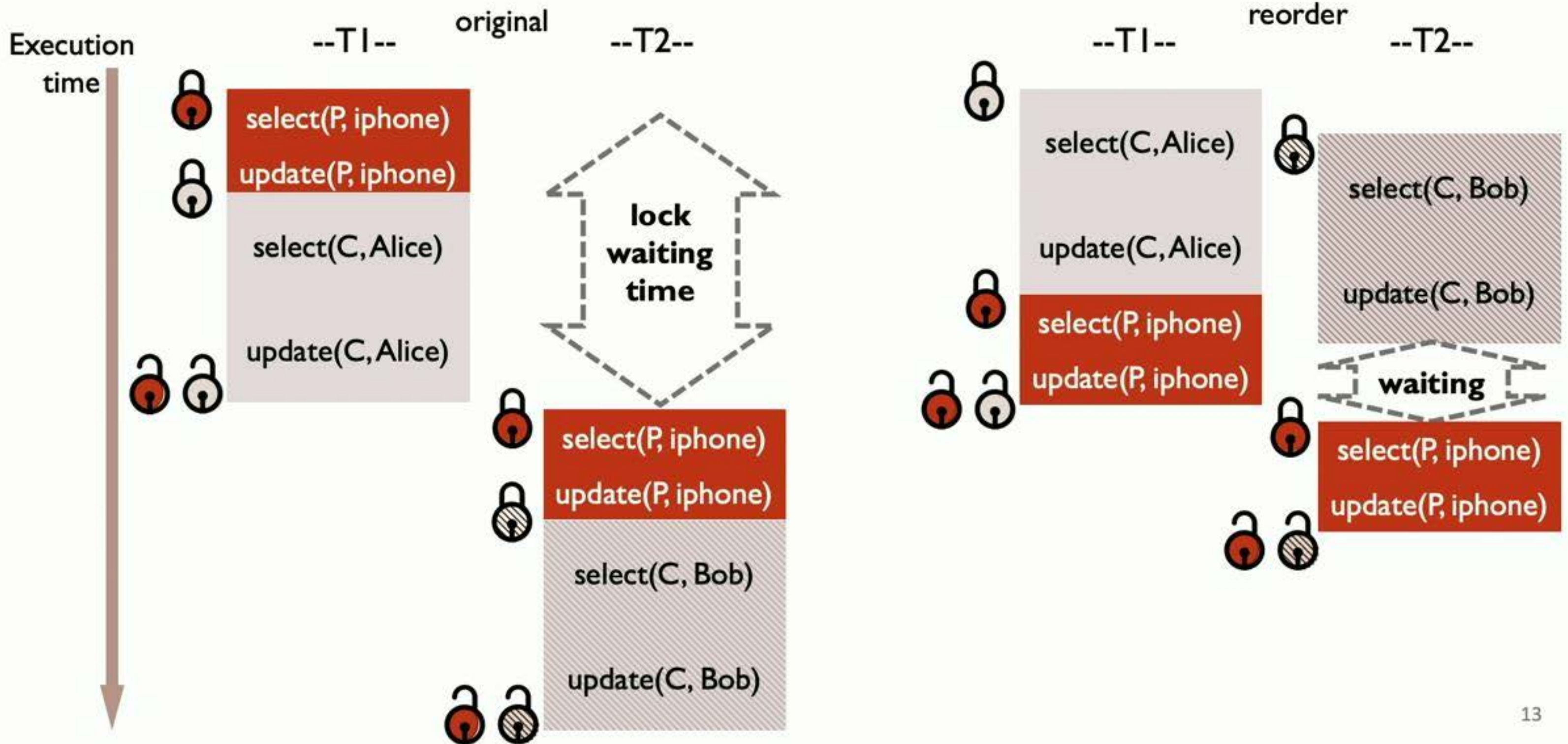
Query Reorder under 2PL



Query Reorder under 2PL



Query Reorder under 2PL

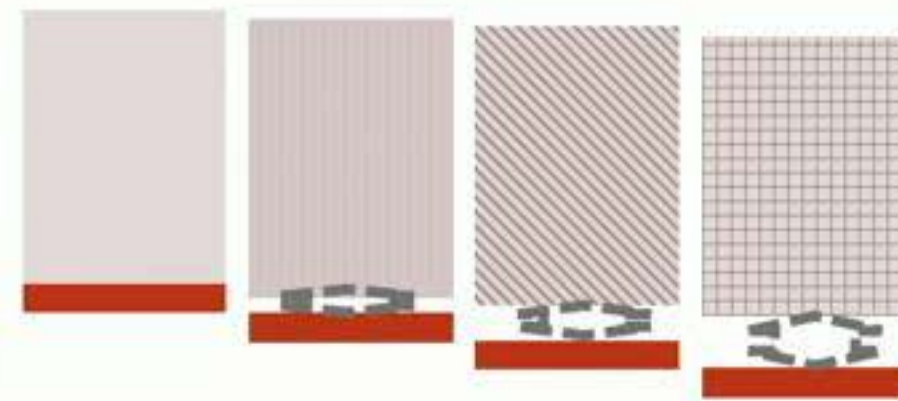
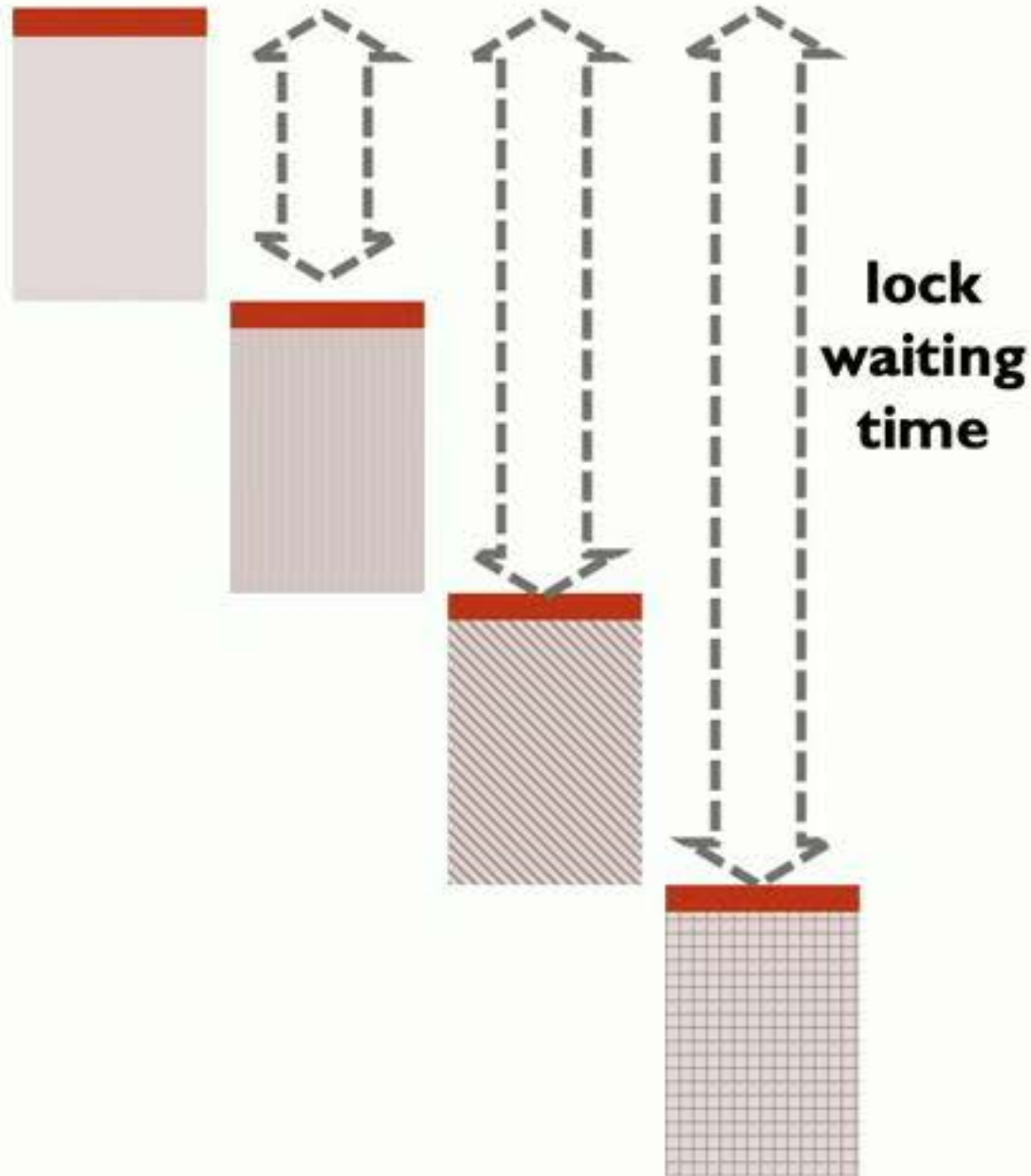


Query Reorder Shortens Lock Wait Time

Execution
time

original

reorder



Quro: Compiler for Query Reorder

Input: C++ code with
embedded SQL queries

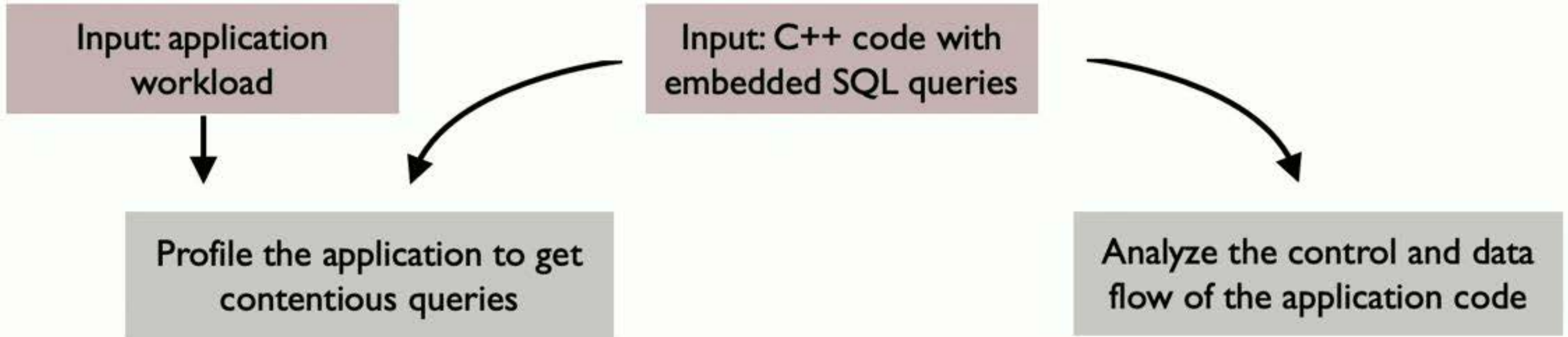
Quro: Compiler for Query Reorder

Input: C++ code with
embedded SQL queries

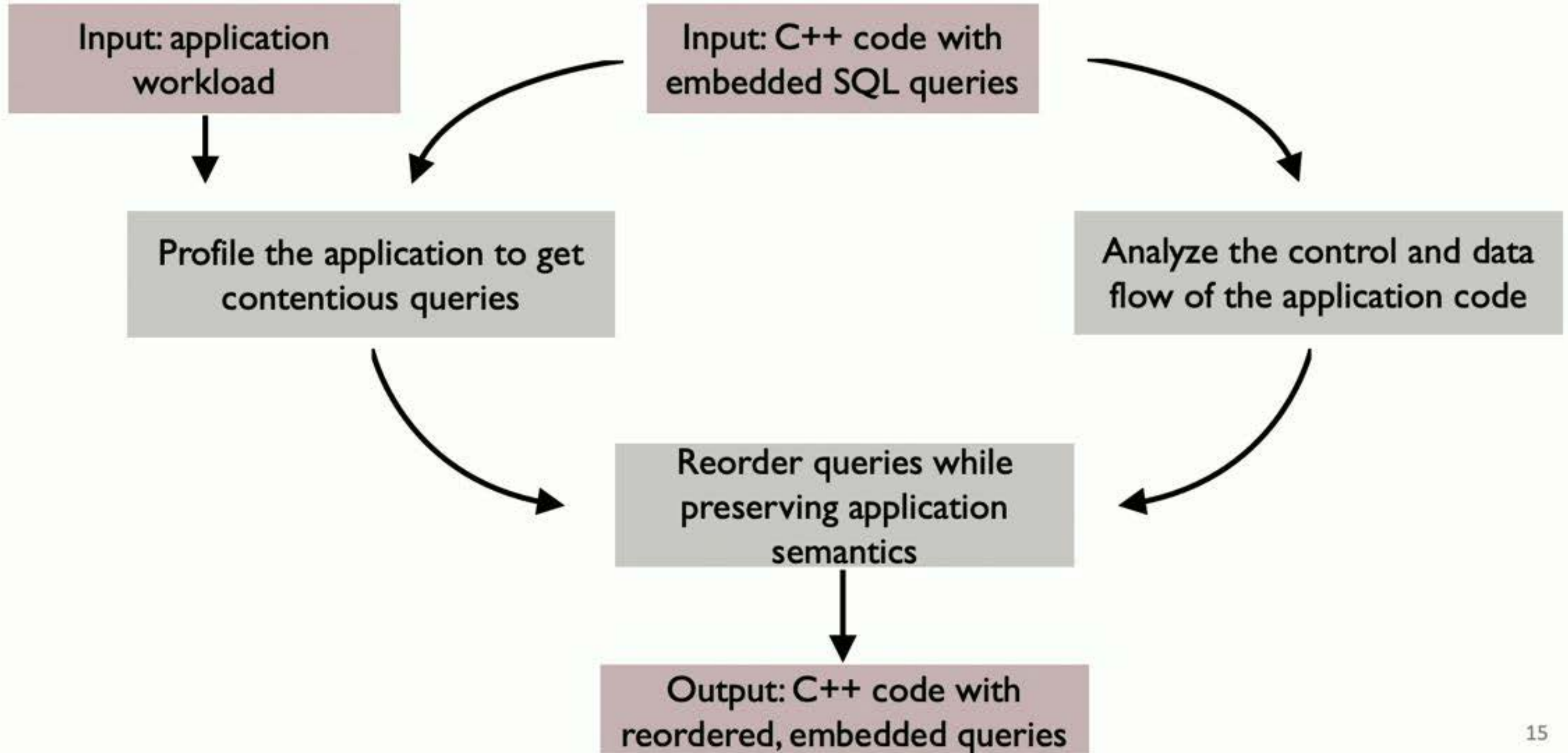


Analyze the control and data
flow of the application code

Quero: Compiler for Query Reorder



Quro: Compiler for Query Reorder



Quero: 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

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

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

Reorder queries while preserving application semantics

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

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
```

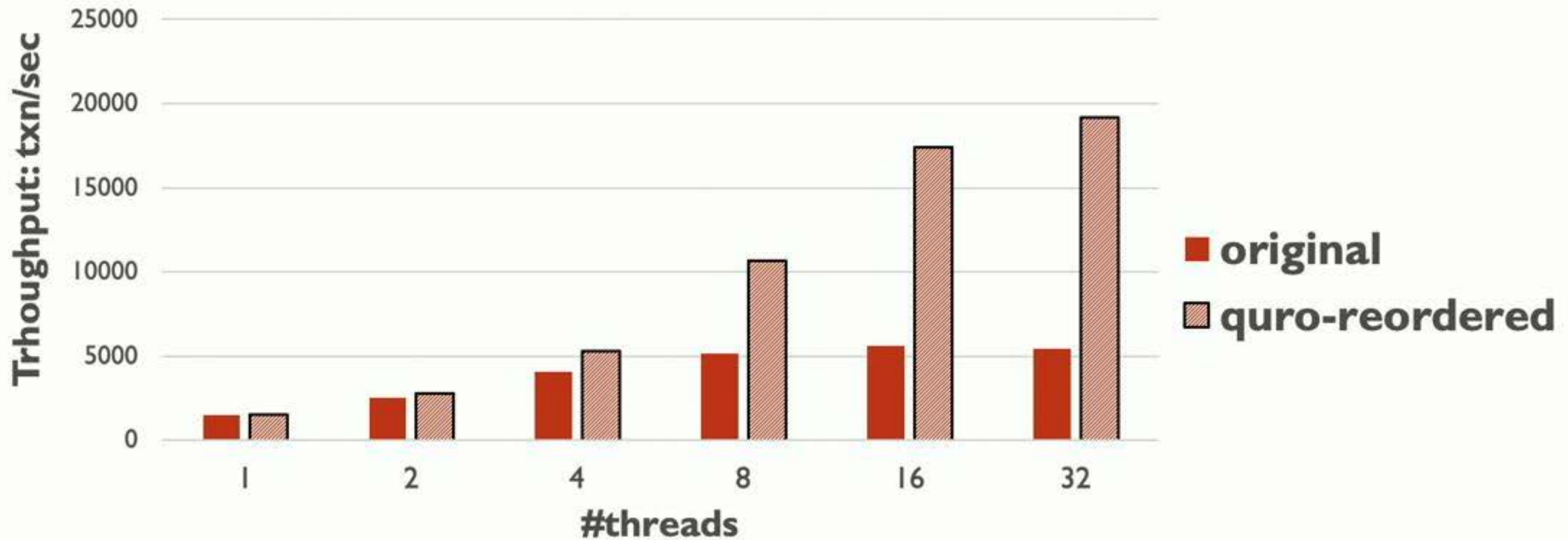
Reorder queries while preserving application semantics

Formalize reorder problem

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

Evaluation: Quro Vs. Original

Workload: TPC-C payment transaction



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
```

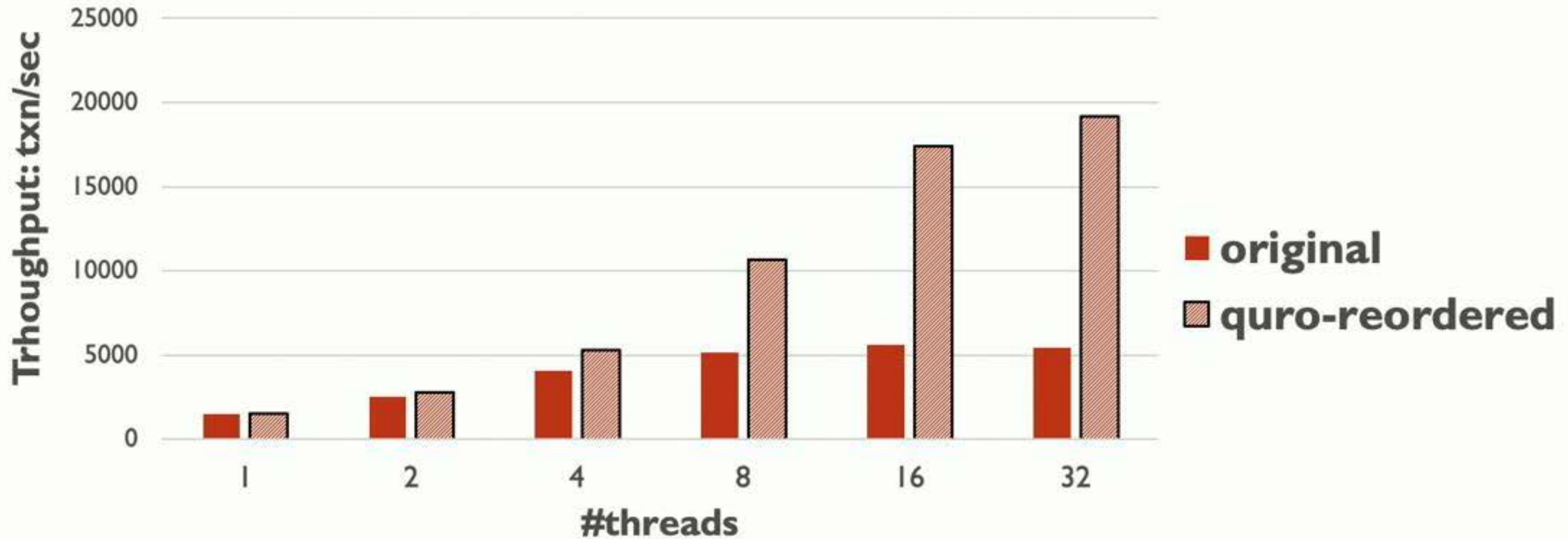
Reorder queries while preserving application semantics

Formalize reorder problem

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

Evaluation: Quro Vs. Original

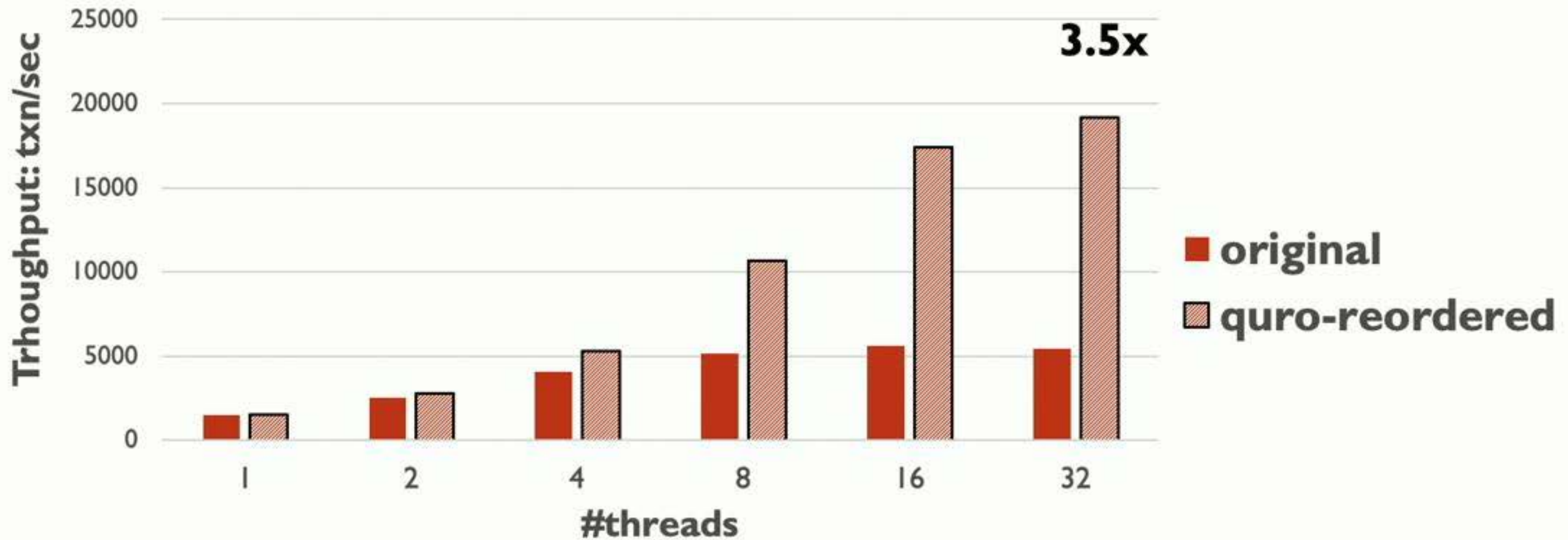
Workload: TPC-C payment transaction



Latency: decrease up to 70%

Evaluation: Quro Vs. Original

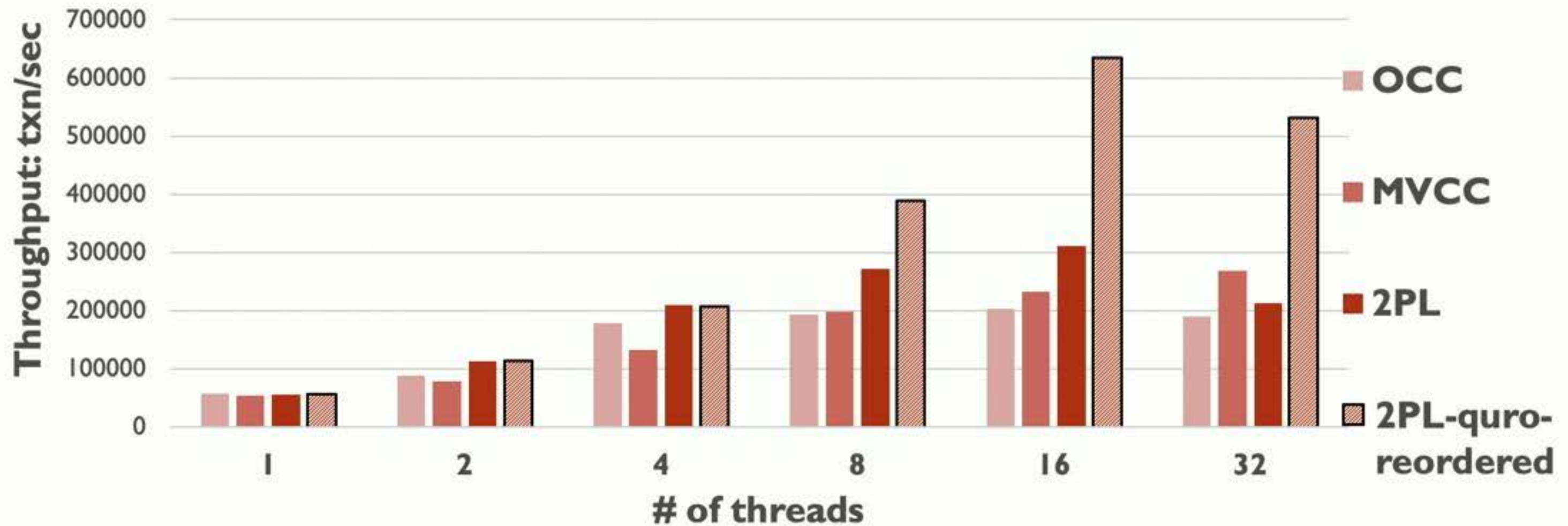
Workload: TPC-C payment transaction



Latency: decrease up to 70%

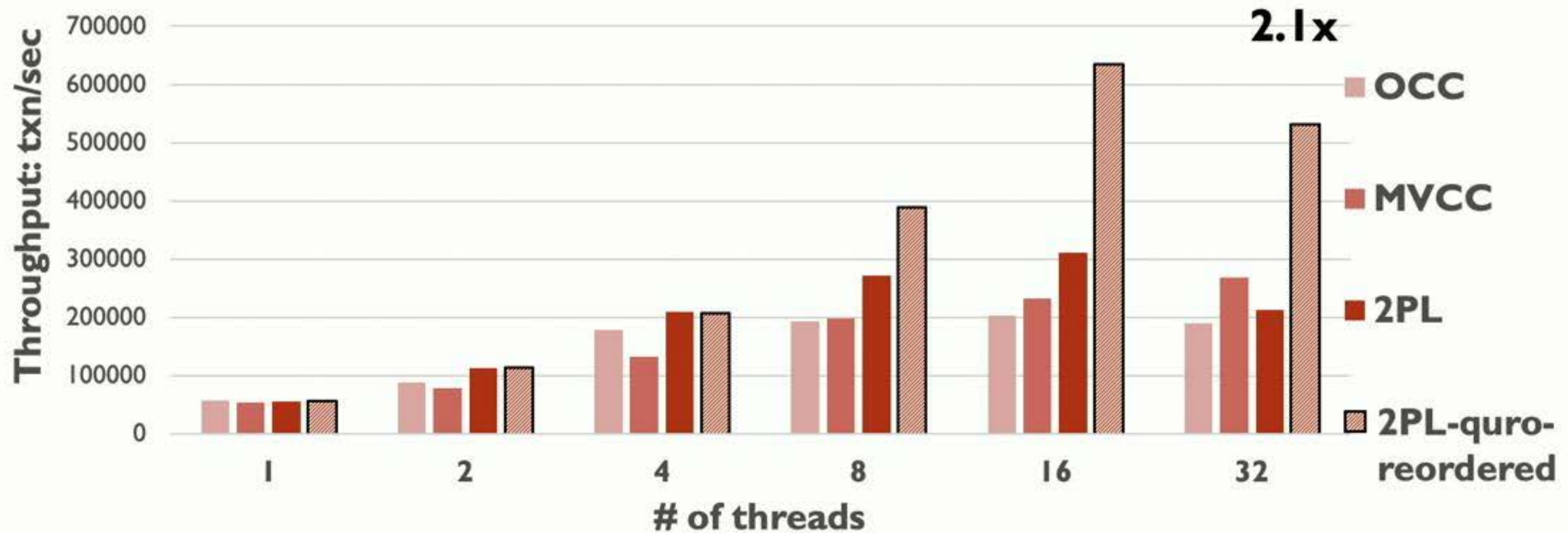
Evaluation: Quro Vs. Other CC Schemes

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



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**



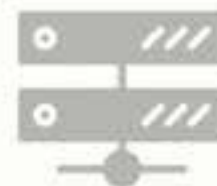
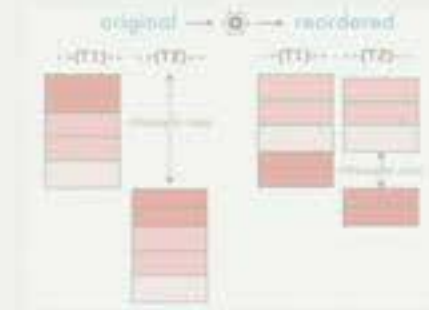
Outline

- Leveraging application semantics to optimize each layer

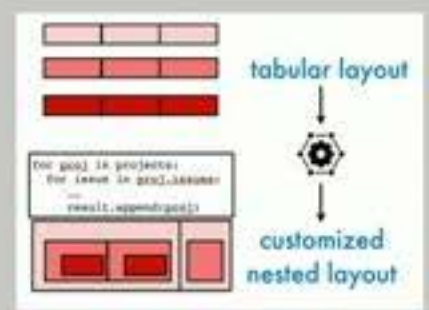
Panorama:
view-driven optimization



Quoro:
reorder queries



Chestnut:
customize data layout

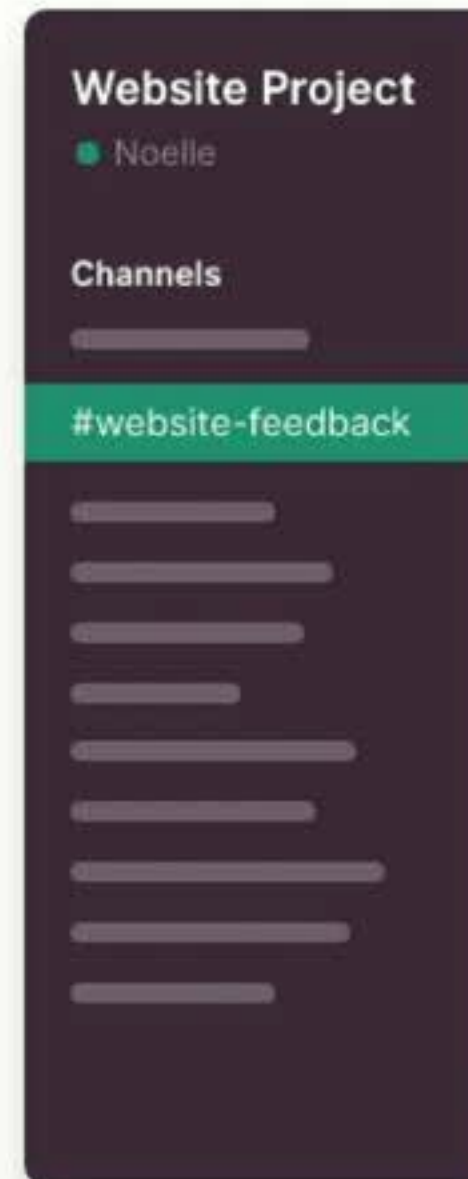


- Other projects
- Ongoing and future work

Data Representation Mismatch

Data Representation Mismatch

➤ Nested object vs. tabular data



#website feedback



Noelle Keyy

Do you have any feedback on the new website?




Steve Young

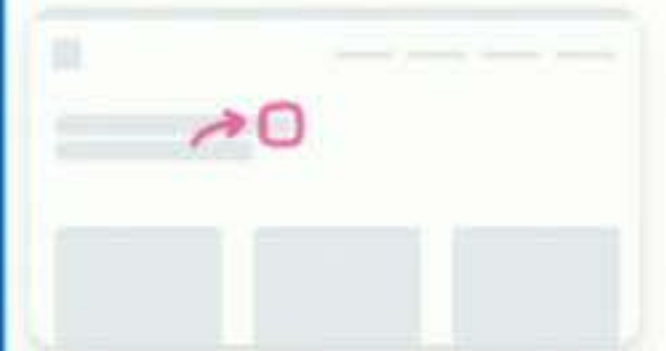
Could you modify the price to \$39?

 Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

 Sent with Marker.io



Data Representation Mismatch

Channel

➤ Nested object vs. tabular data

The image illustrates a data representation mismatch between a mobile app's nested object structure and a chat channel's tabular data structure.

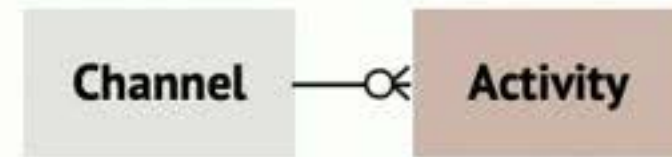
Mobile App View (Left): A dark-themed interface titled "Website Project" by "Noelle". It features a "Channels" section with a list of channels. The channel "#website-feedback" is highlighted in green. Below this, there are several horizontal bars representing other channels or content items.

Chat Channel View (Right): A light-themed chat interface titled "#website feedback". It shows two messages:

- Noelle Keyy:** "Do you have any feedback on the new website?"
- Steve Young:** "Could you modify the price to \$39?"

Below the messages, there is a blue vertical bar with a yellow notepad icon and the text "Turn into Task with Marker.io". Below this is a link "View screenshot" with the URL "https://companywebsite.com/pricing" and a "Sent with Marker.io" icon. At the bottom, there is a screenshot of a website with a red arrow pointing to a price tag.

Data Representation Mismatch



➤ Nested object vs. tabular data

The screenshot shows a mobile application interface. On the left is a dark sidebar with the title 'Website Project' and a user profile 'Noelle'. Below this is a 'Channels' section with a list of channels, where '#website-feedback' is highlighted in green. On the right is the main content area for the '#website-feedback' channel. It shows a list of messages: one from 'Noelle Keyy' asking for feedback, and one from 'Steve Young' asking for a price modification. The message from Steve Young includes a screenshot of a website with a red arrow pointing to a price field, and it is annotated with 'Turn into Task with Marker.io', a 'View screenshot' link, the URL 'https://companywebsite.com/pricing', and a 'Sent with Marker.io' icon.

Data Representation Mismatch



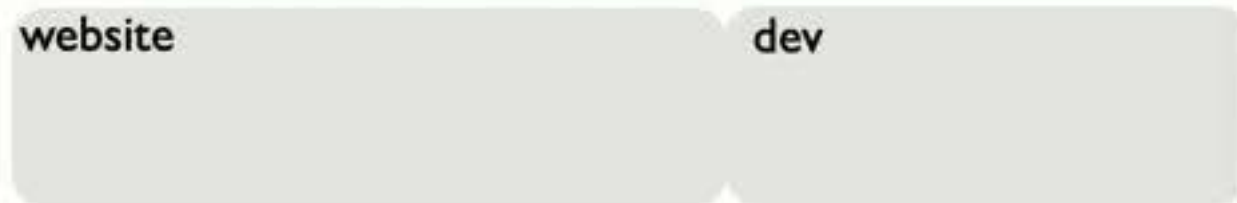
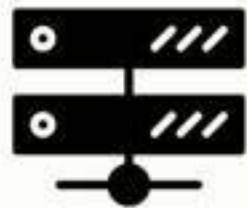
➤ Nested object vs. tabular data

The screenshot shows a mobile application interface. On the left, a dark purple sidebar contains the text "Website Project" with a green dot next to "Noelle", followed by "Channels" and a list of channels including "#website-feedback". On the right, a white card displays "#website feedback" with two entries: "Noelle Keyy" asking for feedback on a new website, and "Steve Young" asking for a price modification to \$39. Below the second entry is a blue vertical bar with a yellow notepad icon, the text "Turn into Task with Marker.io", a "View screenshot" link, a URL "https://companywebsite.com/pricing", and a "Sent with Marker.io" icon. At the bottom of the card is a screenshot of a website with a red arrow pointing to a price field.

Data Representation Mismatch



➤ Nested object vs. tabular data



A screenshot of a mobile app interface. At the top, it says 'Website Project' with a green dot next to the name 'Noelle'. Below that is a 'Channels' section with a list of channels. The selected channel is '#website-feedback', which is highlighted in green. Below the channel list are several horizontal bars representing content.

#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

Could you modify the price to \$39?

👉 Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

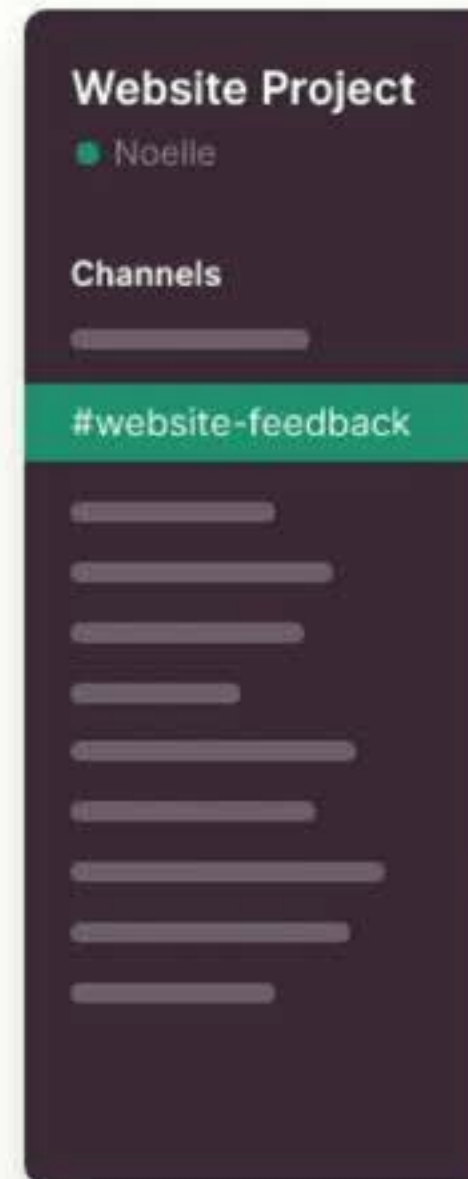
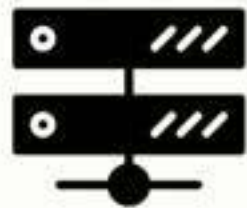
📎 Sent with Marker.io



Data Representation Mismatch



➤ Nested object vs. tabular data



#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

Could you modify the price to \$39?

👉 Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

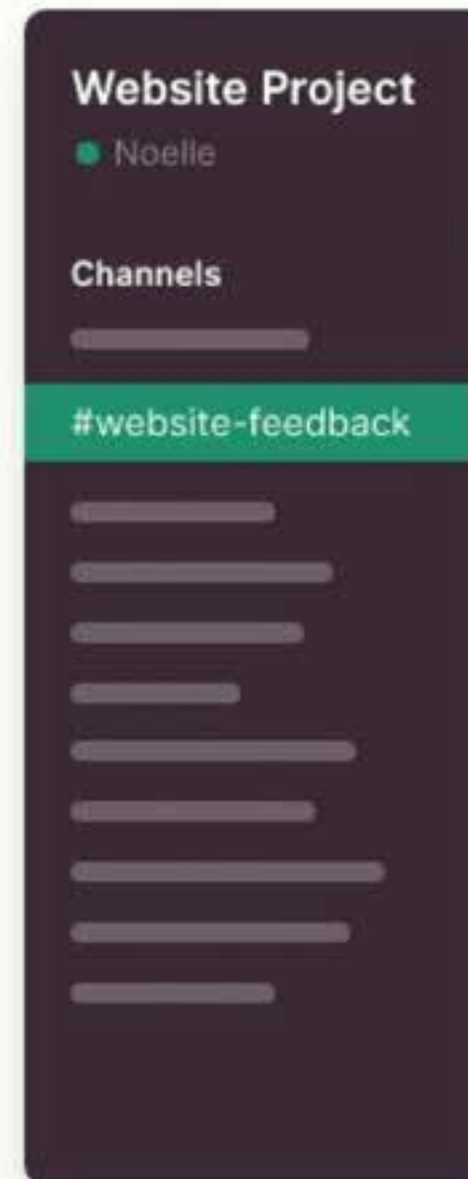
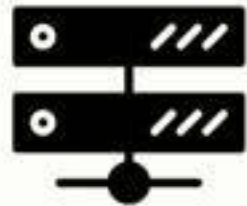
📎 Sent with Marker.io



Data Representation Mismatch



➤ Nested object vs. tabular data



#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

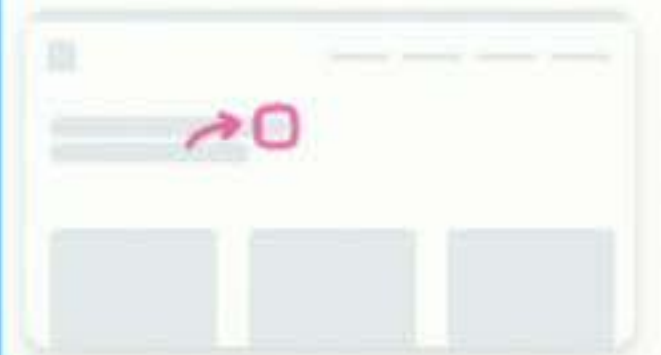
Could you modify the price to \$39?

👉 Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

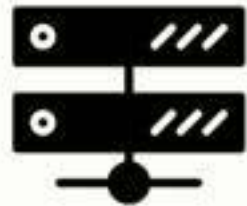
📎 Sent with Marker.io



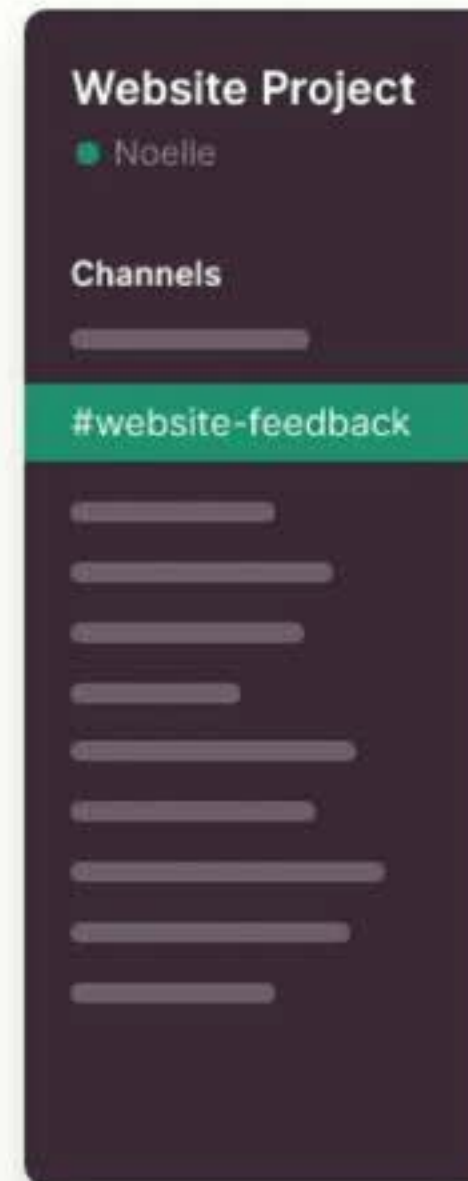
Data Representation Mismatch



➤ Nested object vs. tabular data



website	dev	branch
msg, "Hey,"	file, a.pdf	file, b.pdf
Key	Young	Eve



#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

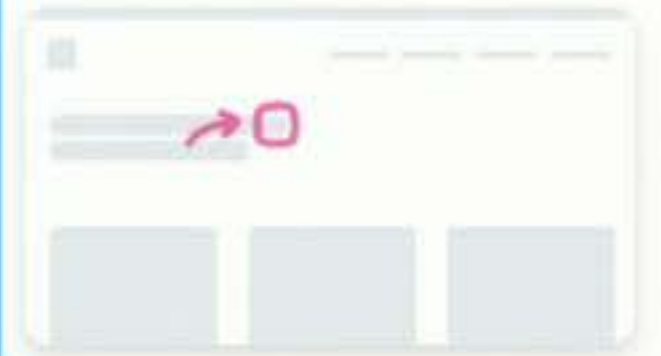
Could you modify the price to \$39?

👉 Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

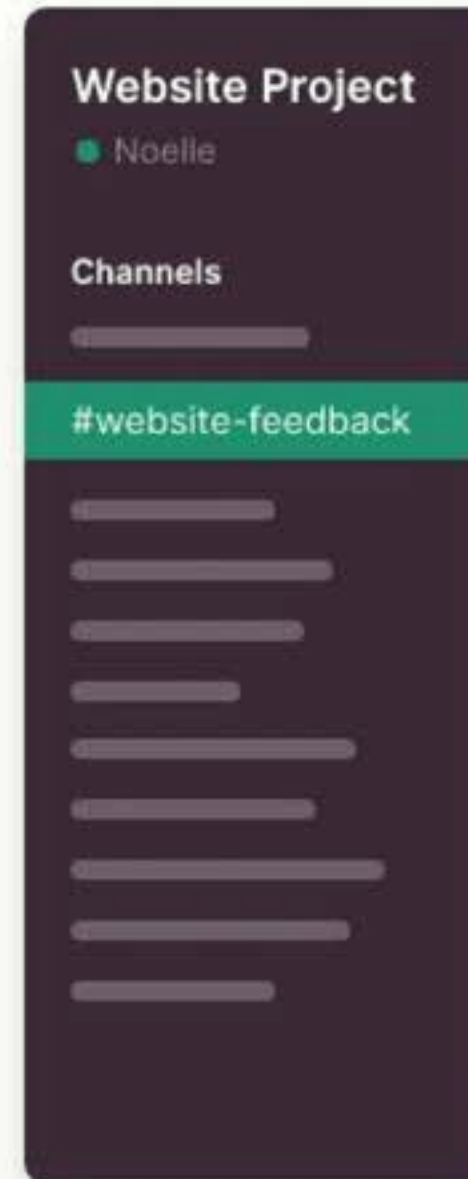
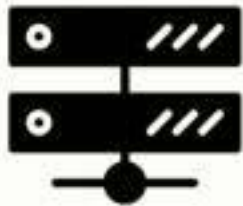
📎 Sent with Marker.io



Data Representation Mismatch



➤ Nested object vs. tabular data



#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

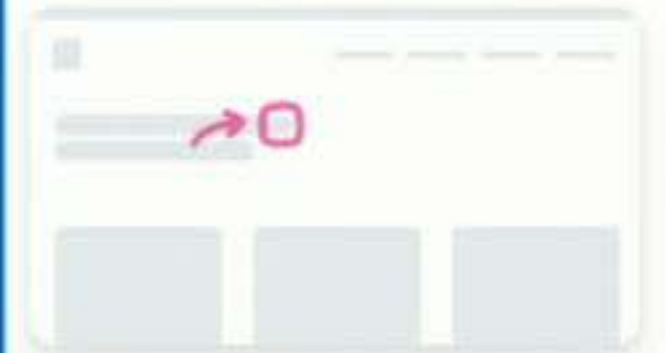
Could you modify the price to \$39?

Turn into Task with Marker.io

[View screenshot](#)

<https://companywebsite.com/pricing>

Sent with Marker.io



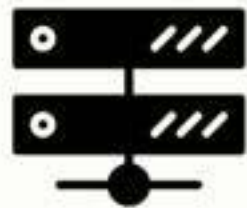
50	website	dev	branch
10K	msg, "Hey,"	file, a.pdf	file, b.pdf
5K	Keyy	Young	Eve

1.7 sec

Data Representation Mismatch



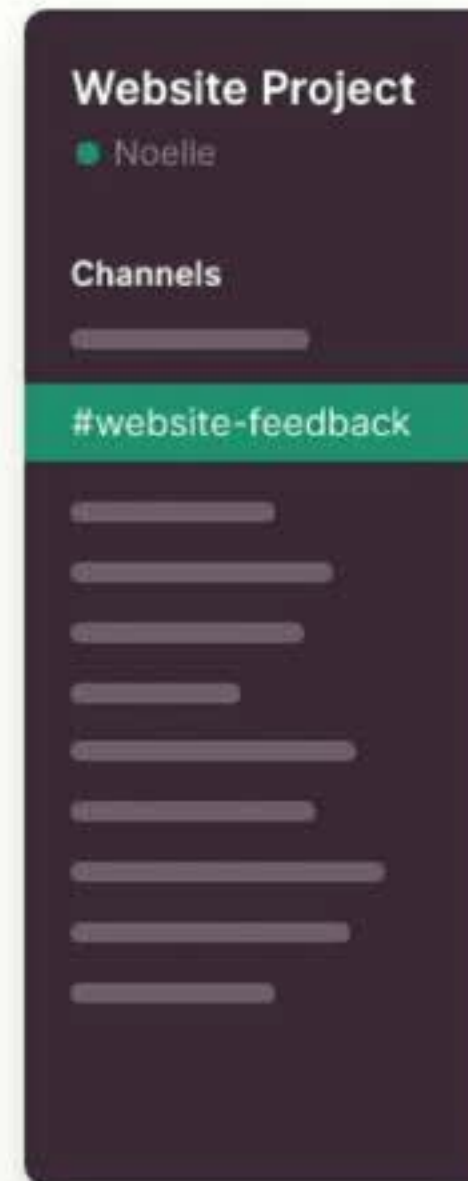
➤ Nested object vs. tabular data



↑ 55 sec



50	website	dev	brunch	
10K	msg, "Hey,"	file, a.pdf	file, b.pdf	1.7 sec
5K	Keyy	Young	Eve	



#website feedback



Noelle Keyy

Do you have any feedback on the new website?



Steve Young

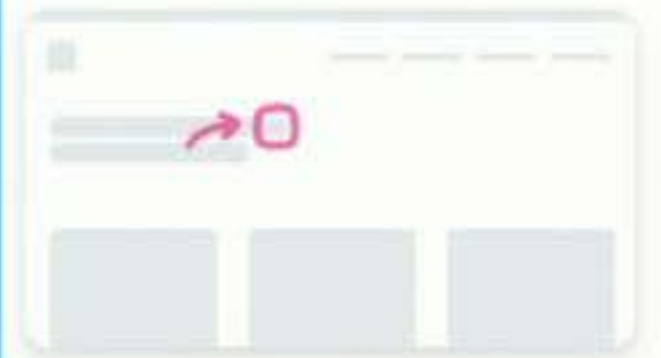
Could you modify the price to \$39?

Turn into Task with Marker.io

[View screenshot](#)

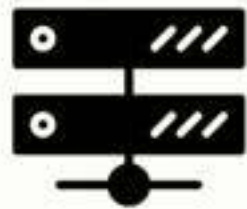
<https://companywebsite.com/pricing>

Sent with Marker.io



Different App Queries Use Different Layout

➤ Nested object vs. tabular data

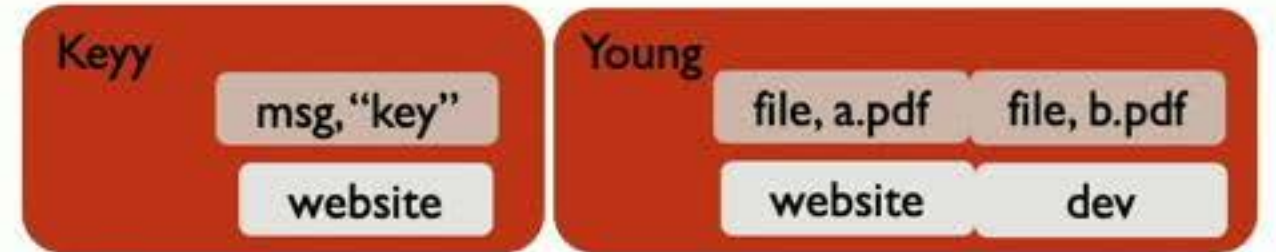
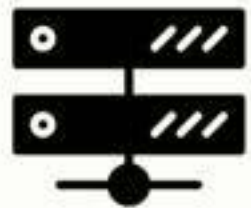


website	dev	branch
msg, "Hey,"	file, a.pdf	file, b.pdf
Key	Young	Eve



Different App Queries Use Different Layout

➤ Nested object vs. tabular data



website	dev	branch
msg, "Hey,"	file, a.pdf	file, b.pdf
Key	Young	Eve



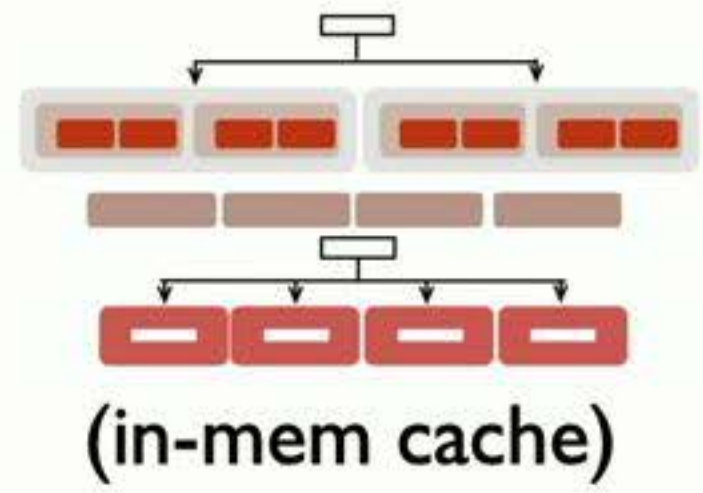
How to Run Queries Faster?



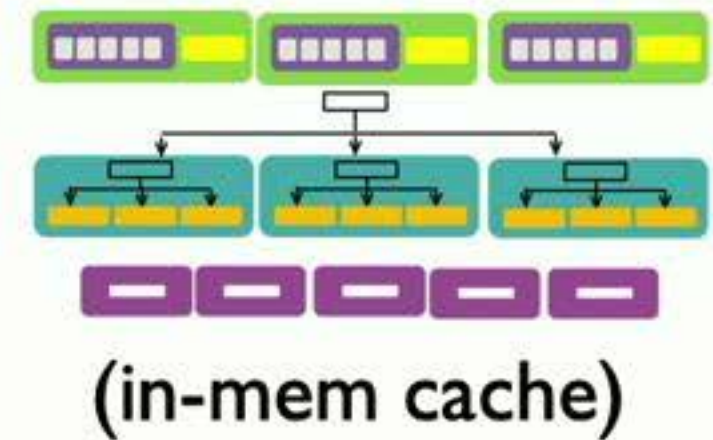
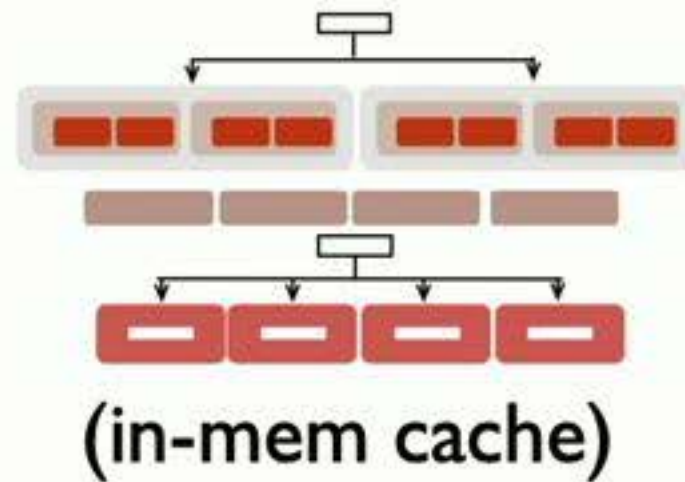
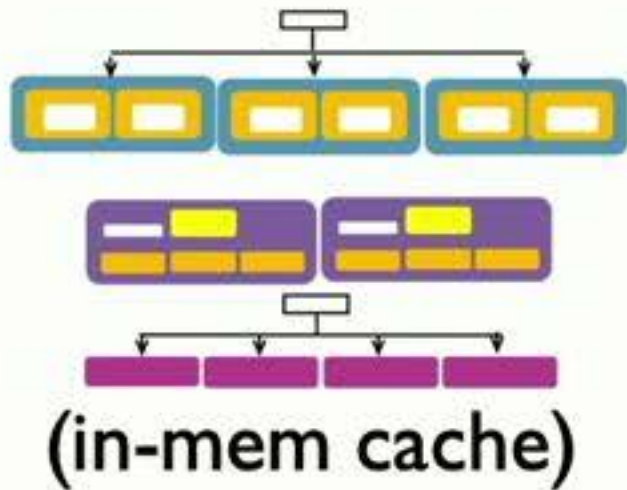
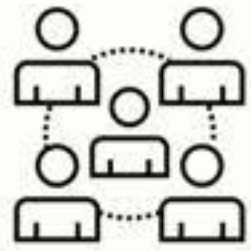
How to Run Queries Faster?



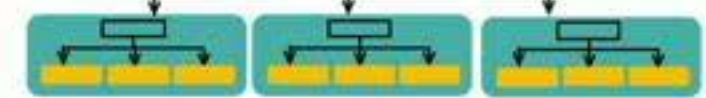
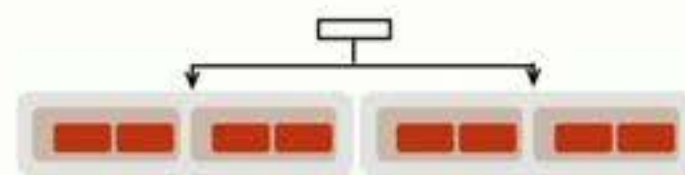
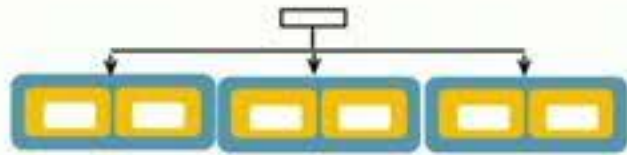
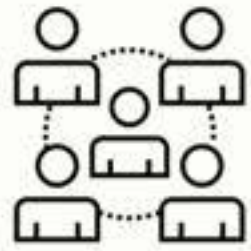
How to Run Queries Faster?



How to Run Queries Faster?



How to Run Queries Faster?



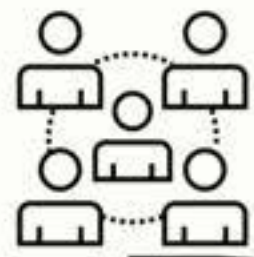
(in-mem cache)

(in-mem cache)

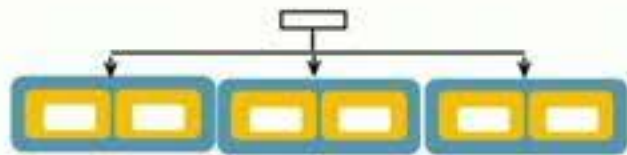
(in-mem cache)



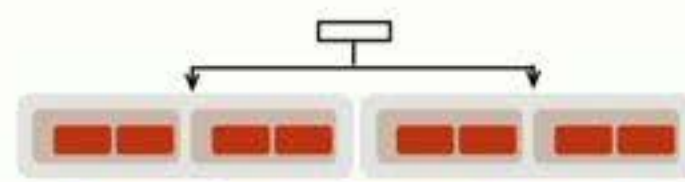
How to Run Queries Faster?



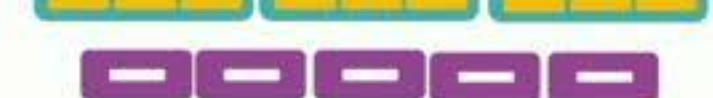
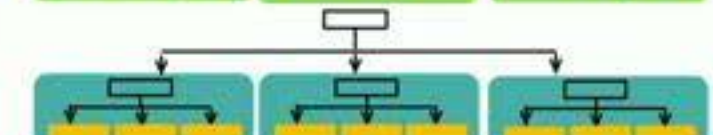
Chestnut



(in-mem cache)



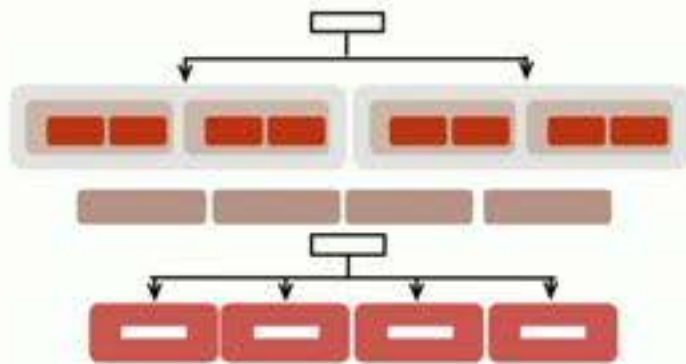
(in-mem cache)



(in-mem cache)



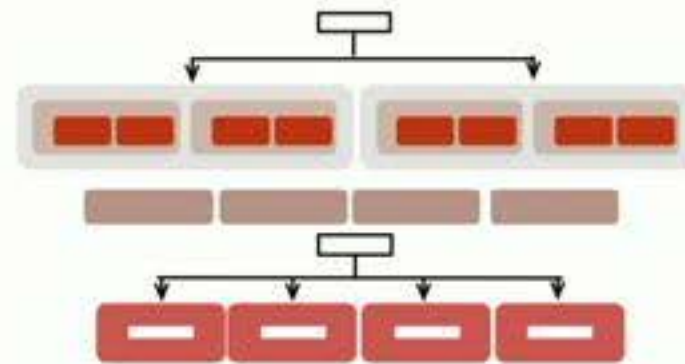
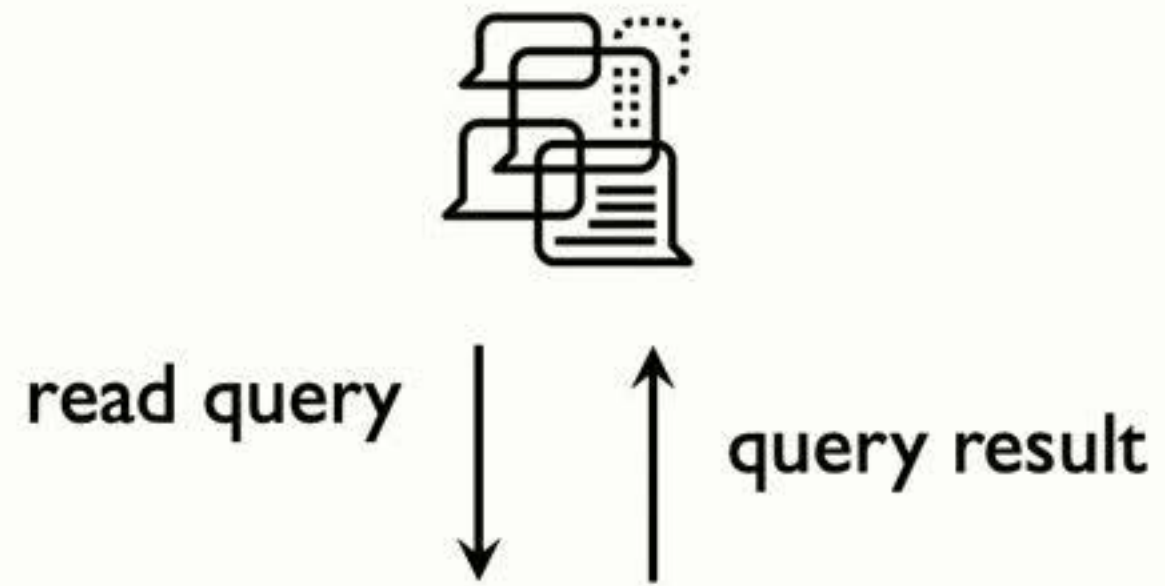
How to Run Queries Faster?



(chestnut-generated
in-mem cache)



How to Run Queries Faster?



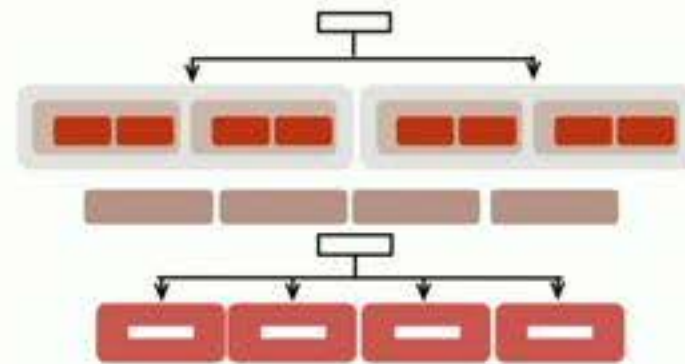
(chestnut-generated
in-mem cache)



How to Run Queries Faster?



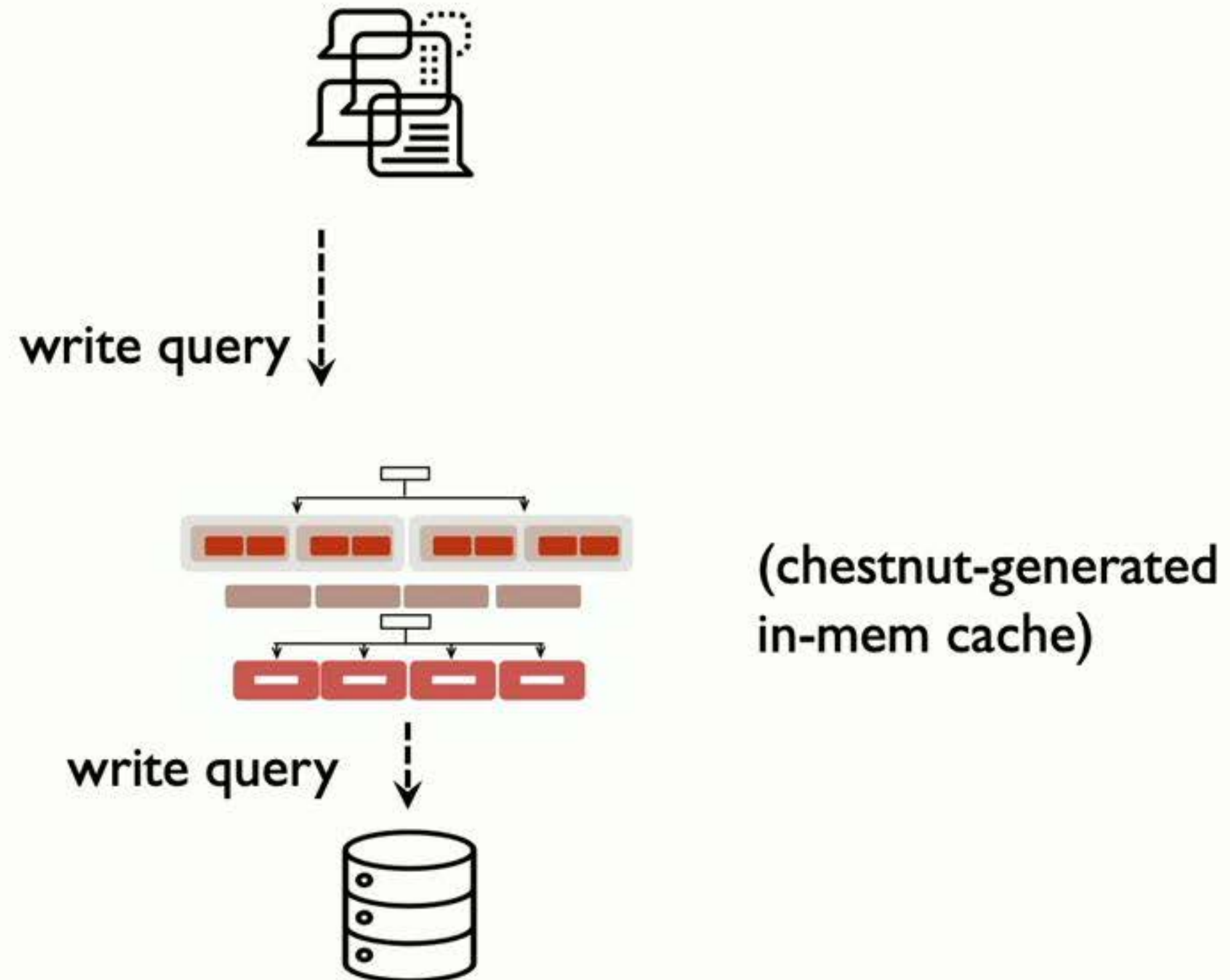
write query



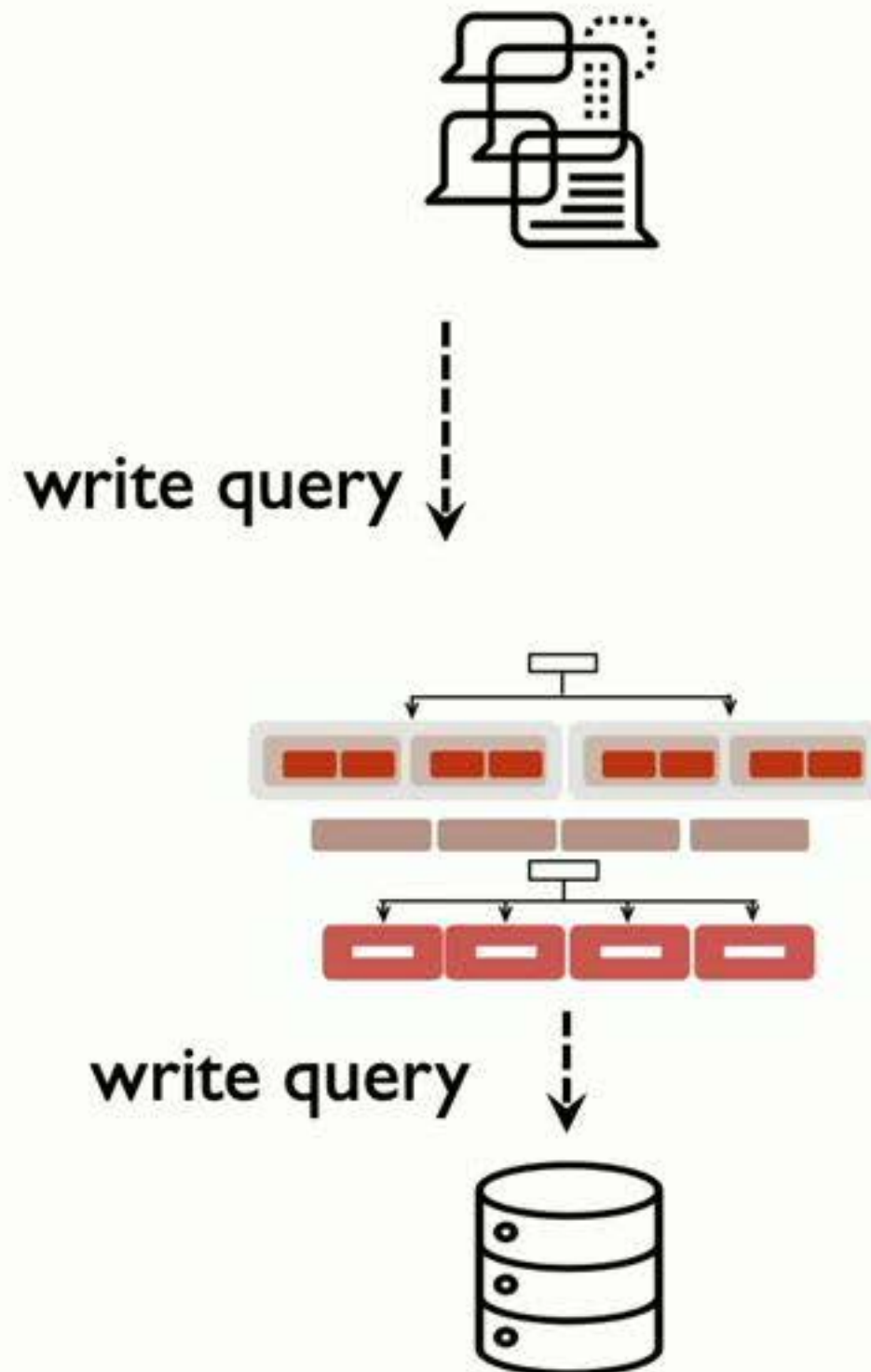
(chestnut-generated
in-mem cache)



How to Run Queries Faster?



How to Run Queries Faster?

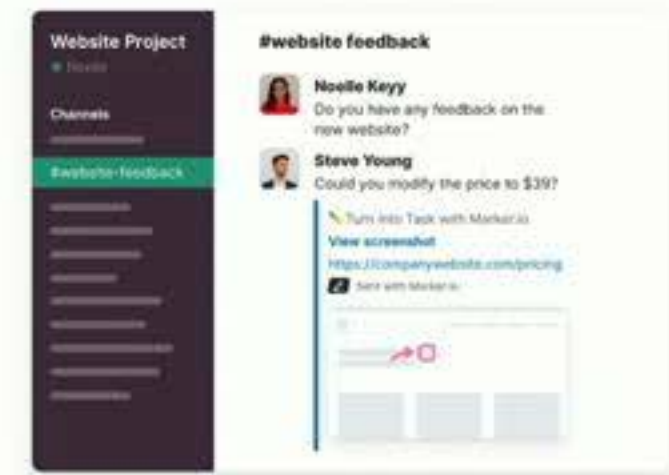


Key challenges:

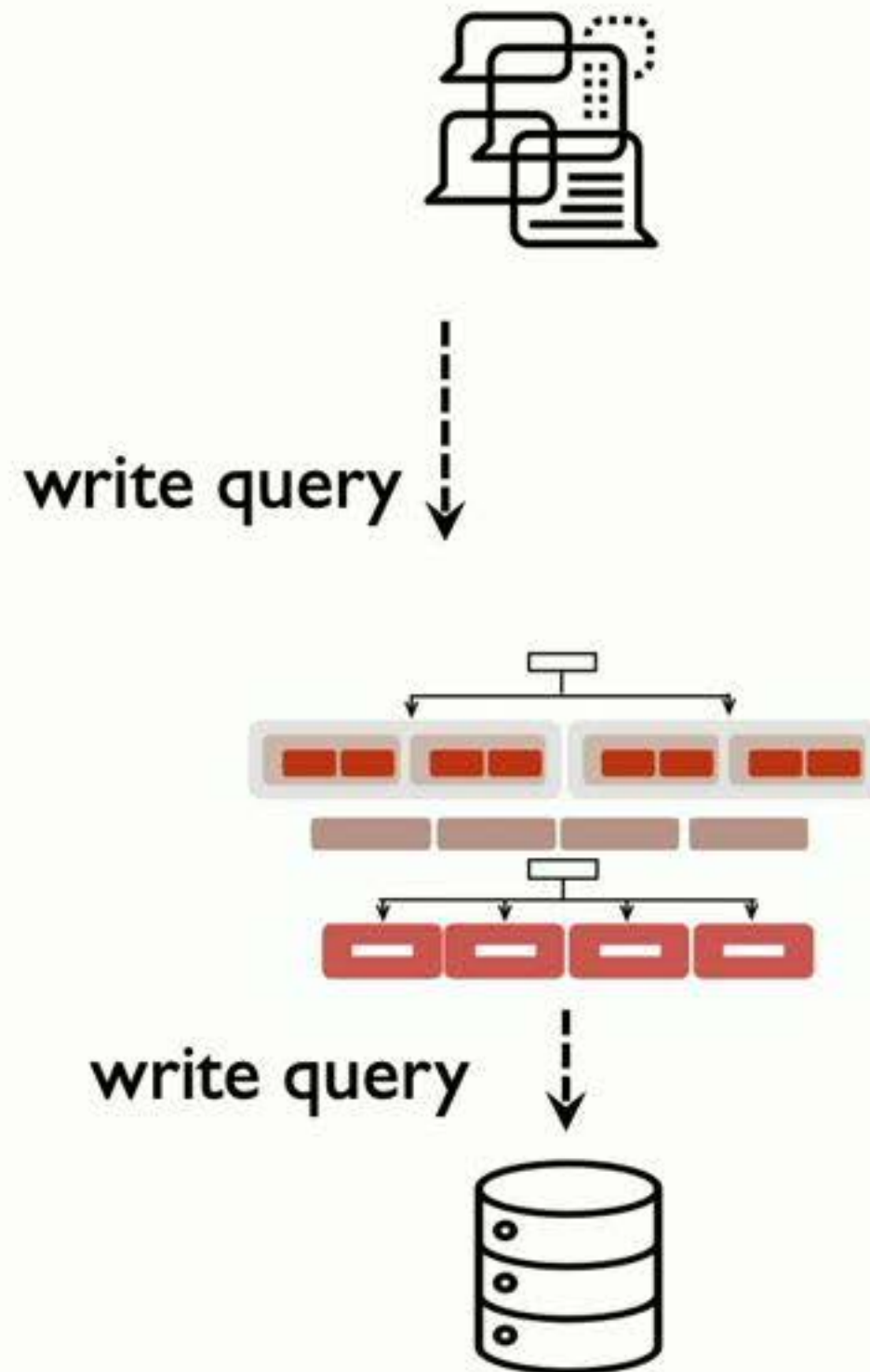
1. How to decide the data layout?
2. How to answer the queries?

(chestnut-generated
in-mem cache)

Chestnut Step 1: Layout Enumeration



How to Run Queries Faster?

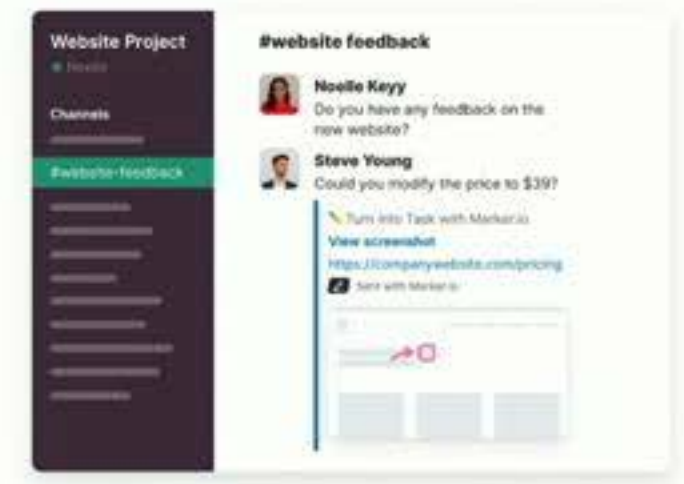


Key challenges:

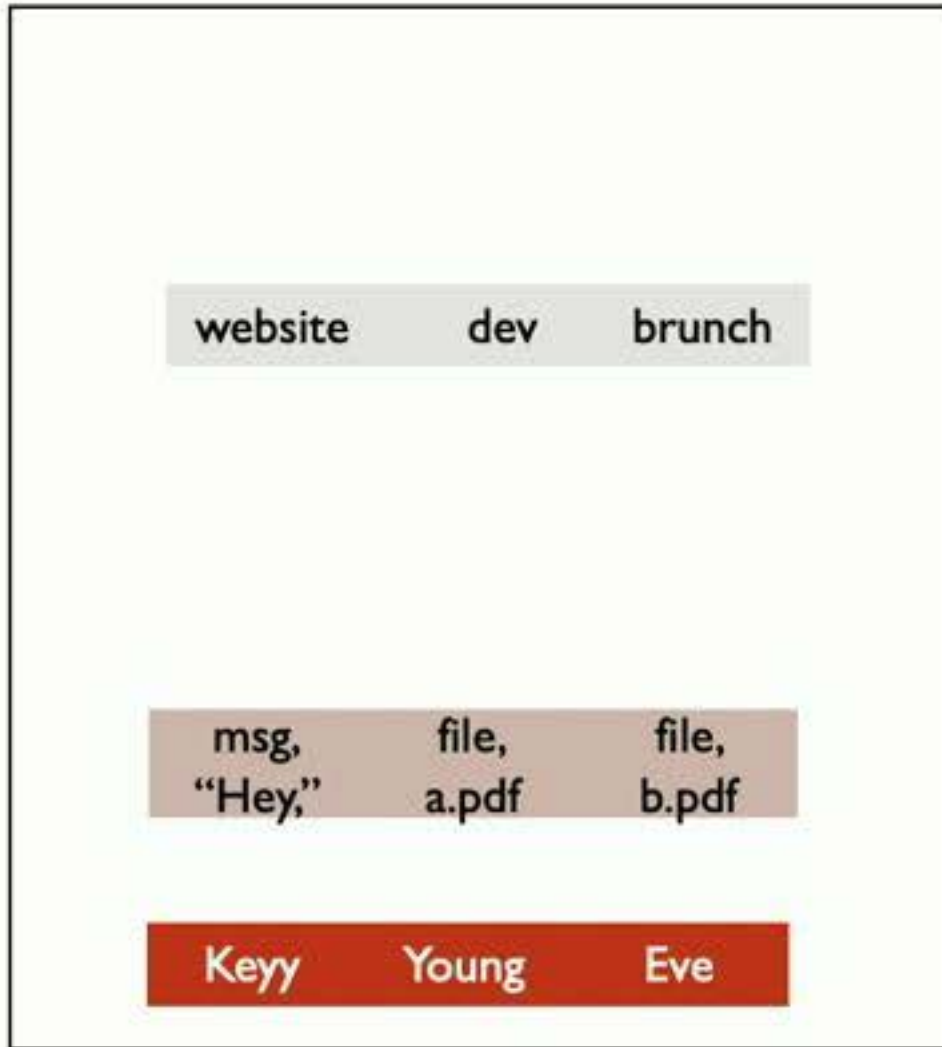
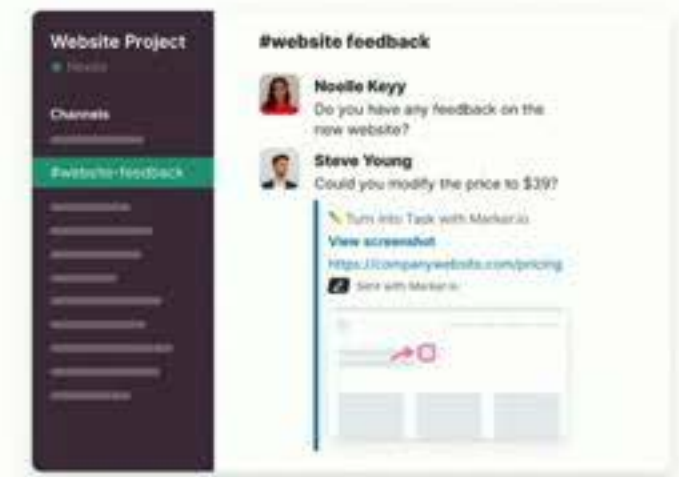
1. How to decide the data layout?
2. How to answer the queries?

(chestnut-generated
in-mem cache)

Chestnut Step 1: Layout Enumeration



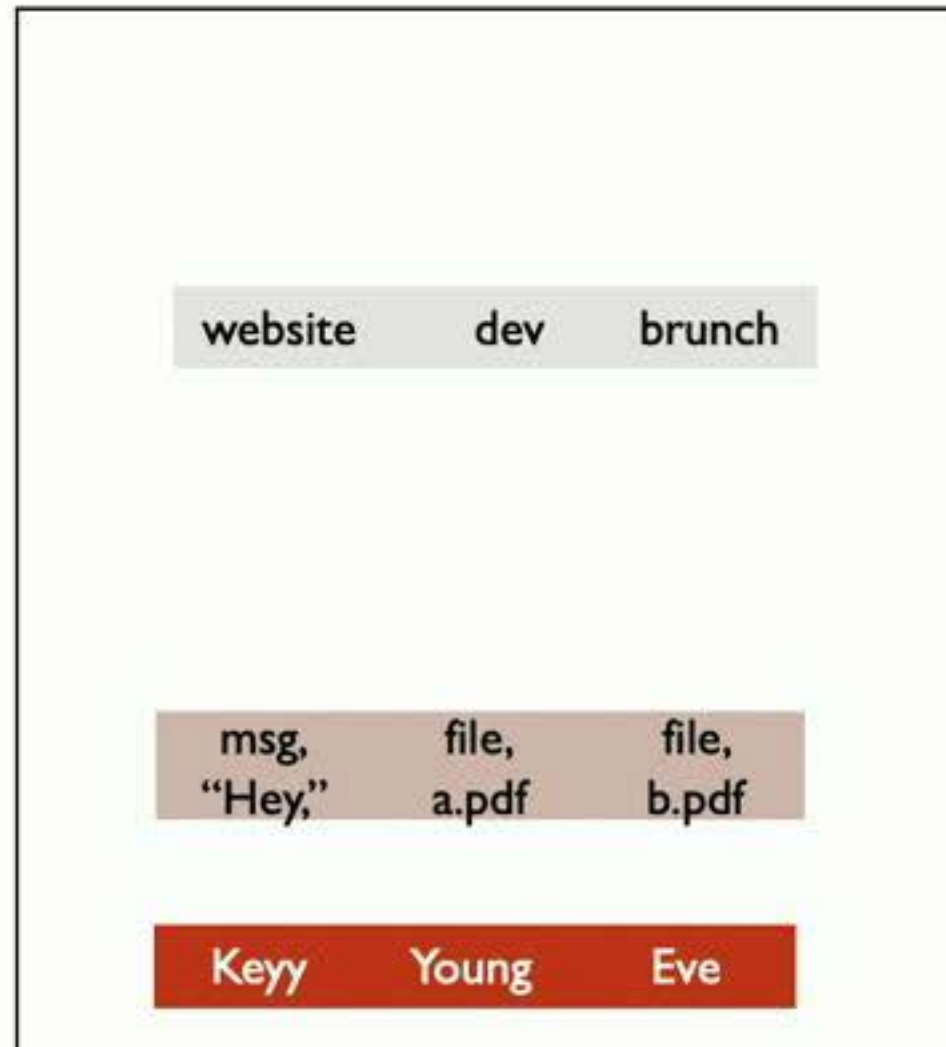
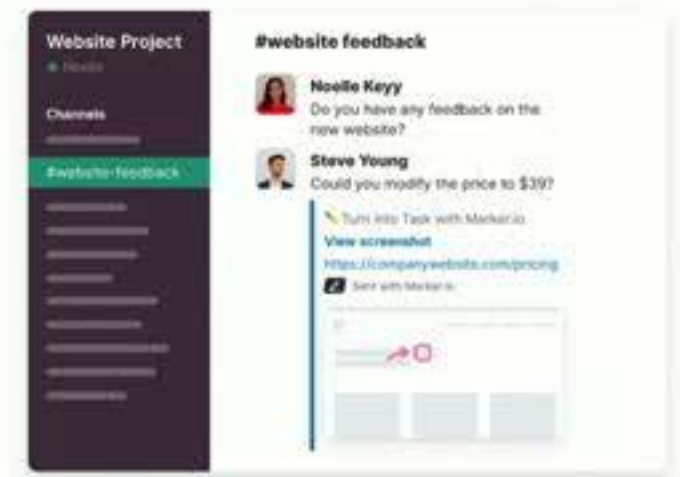
Chestnut Step 1: Layout Enumeration



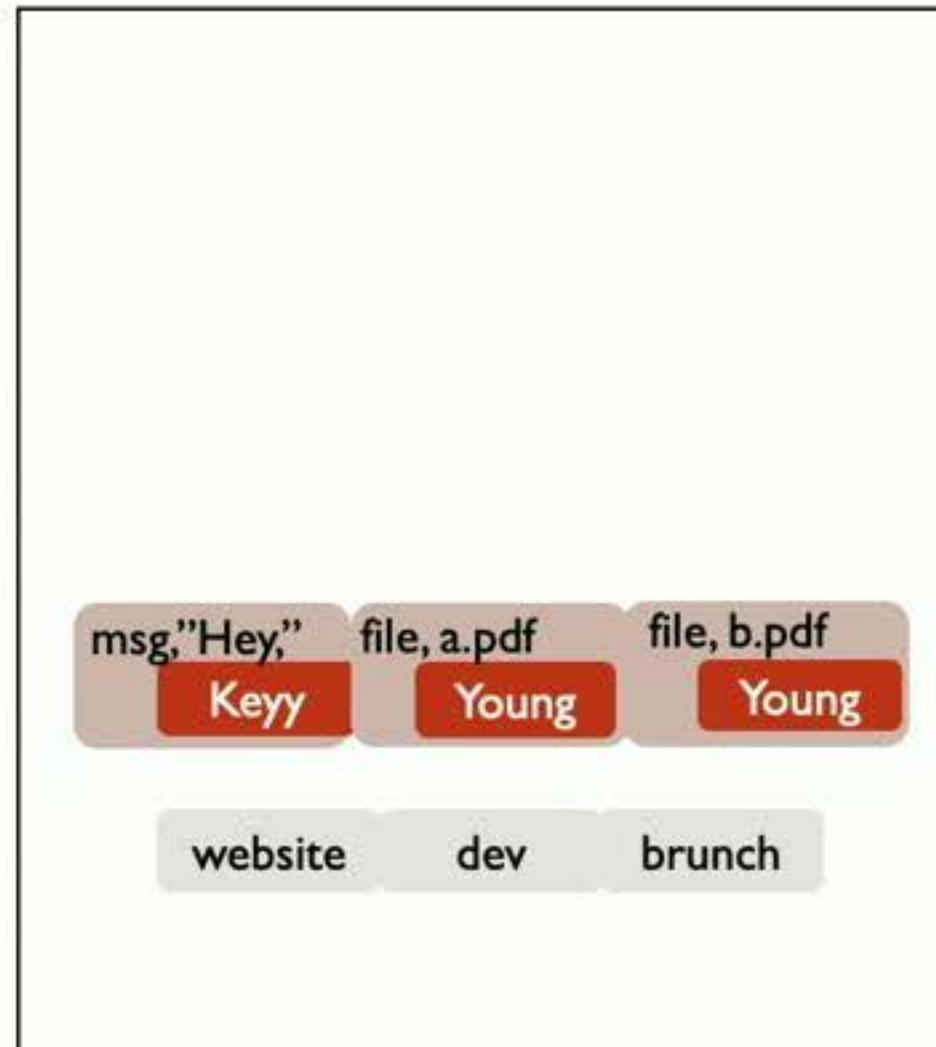
layout 1



Chestnut Step 1: Layout Enumeration



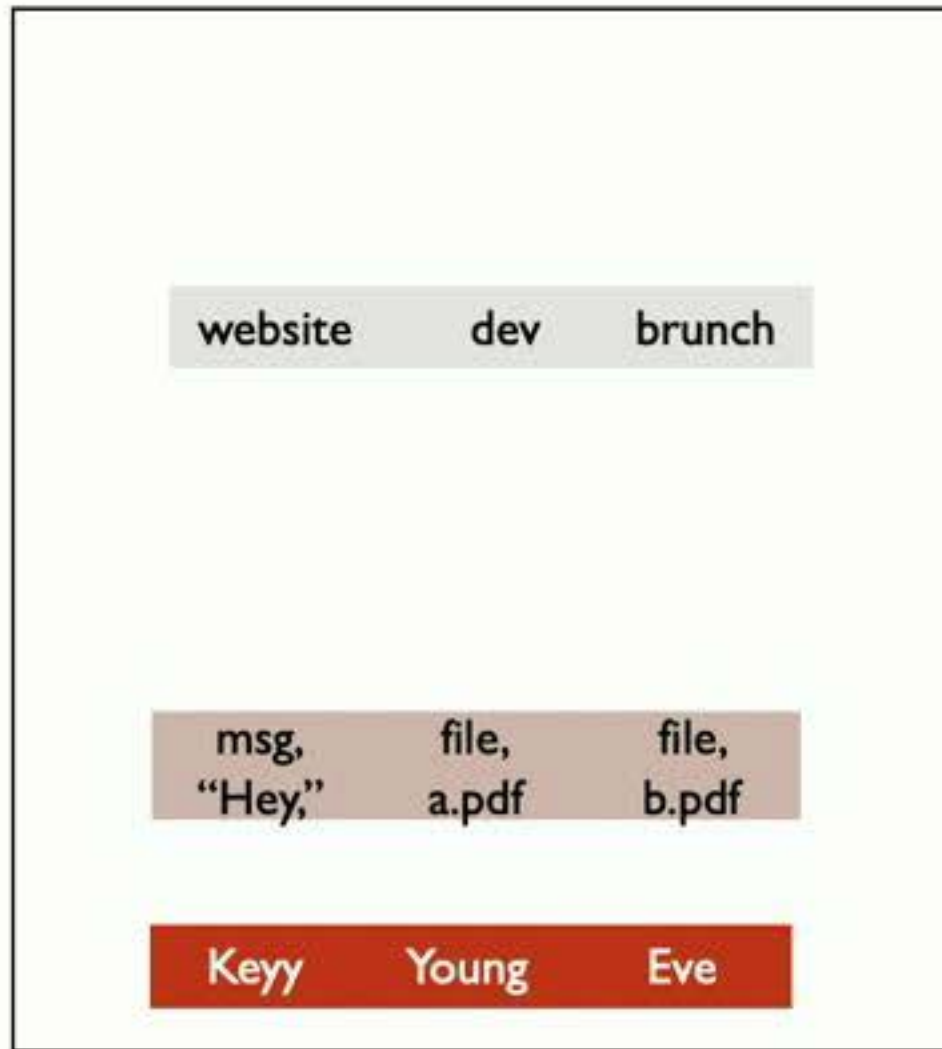
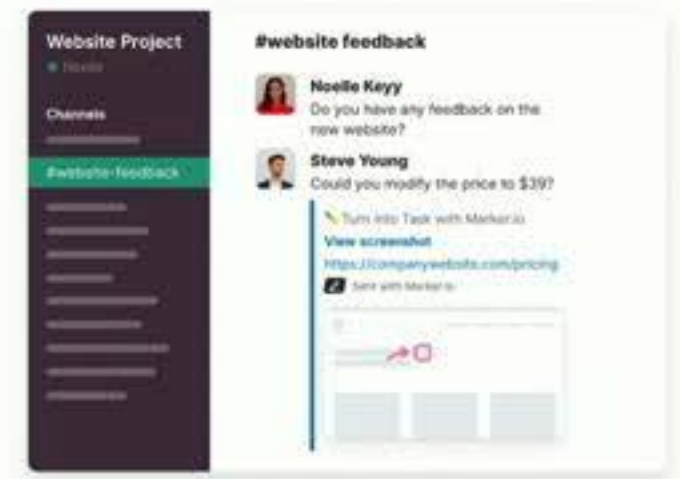
layout 1



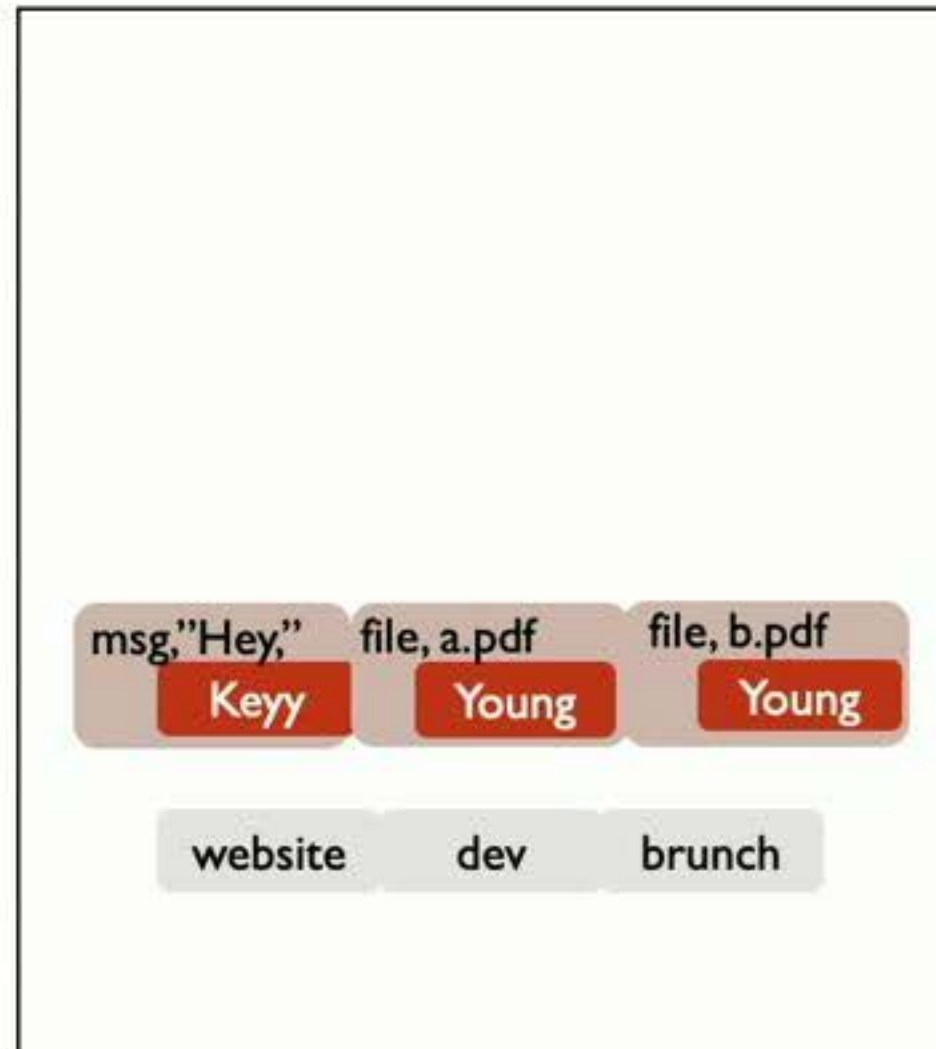
layout 2



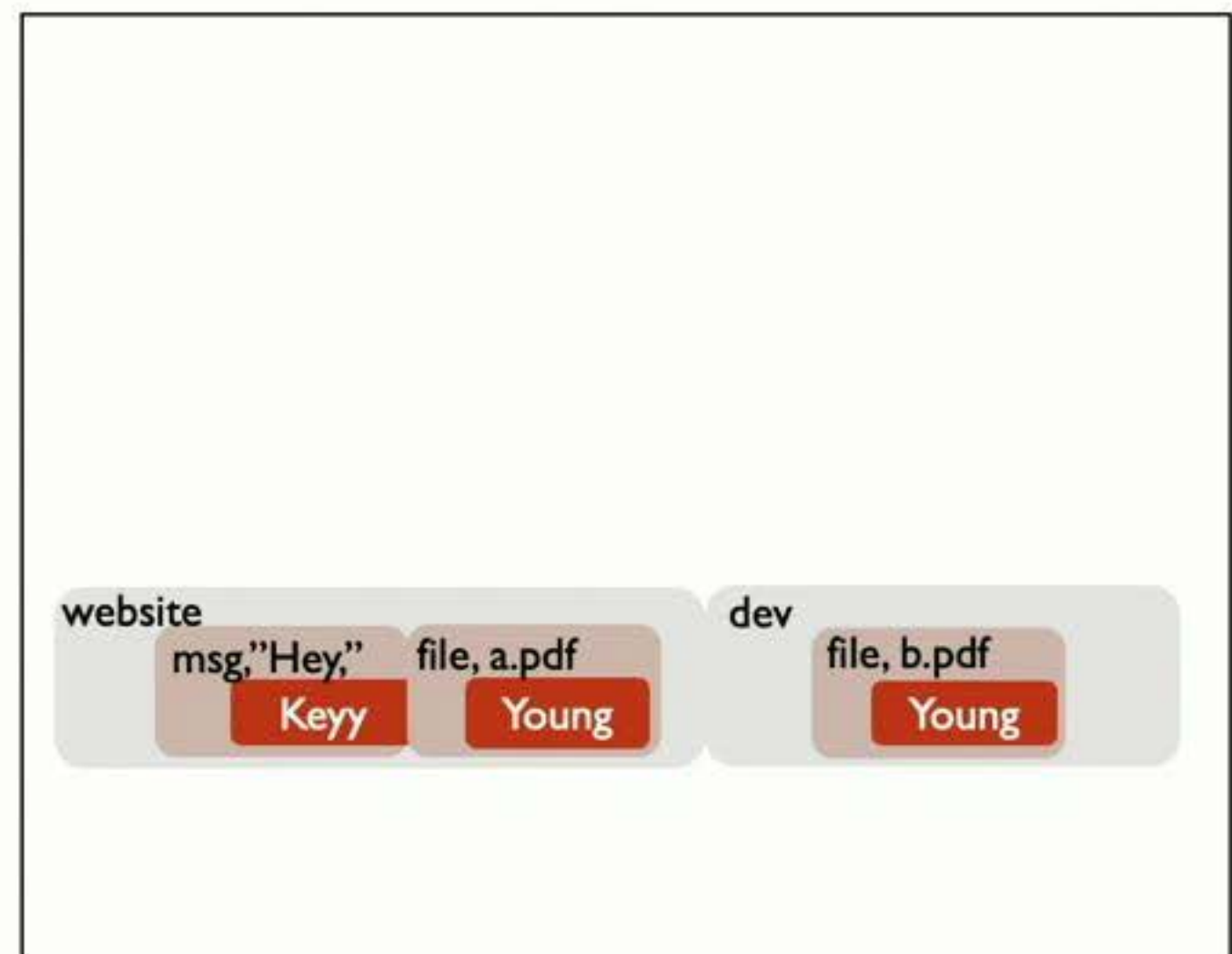
Chestnut Step 1: Layout Enumeration



layout 1



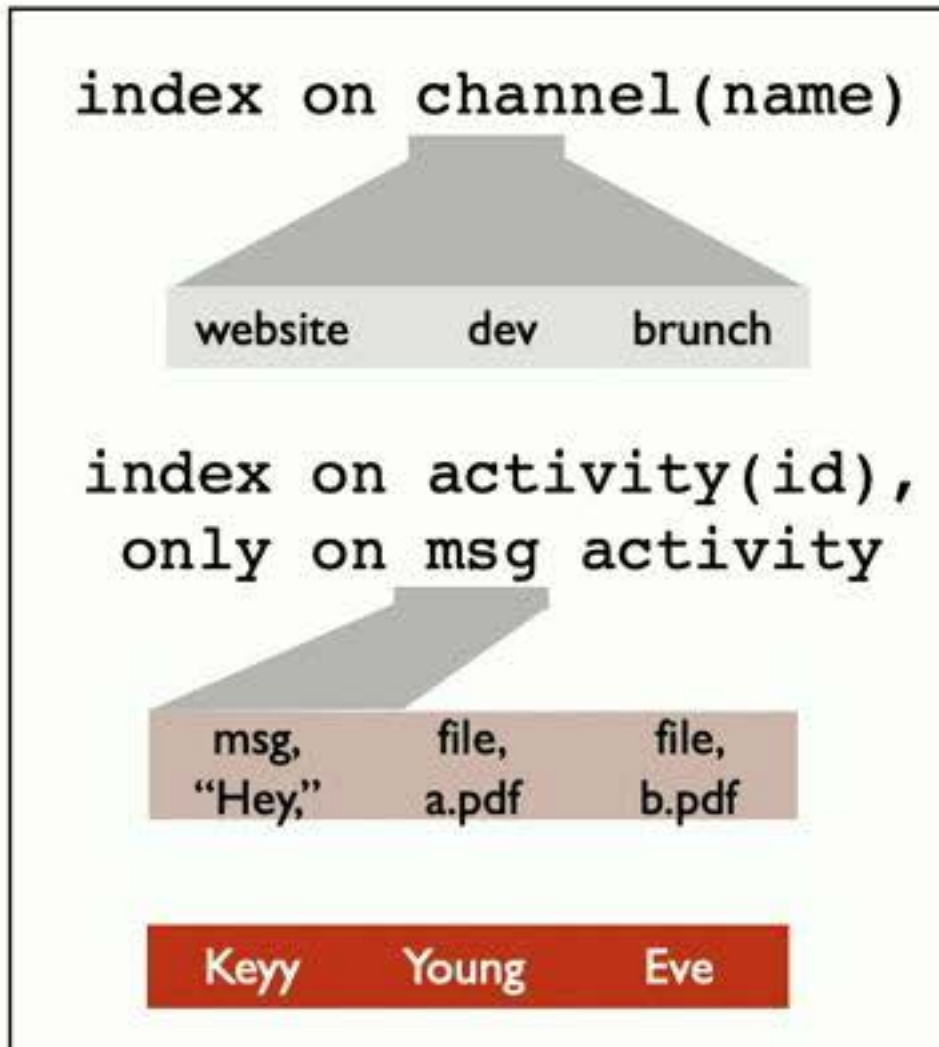
layout 2



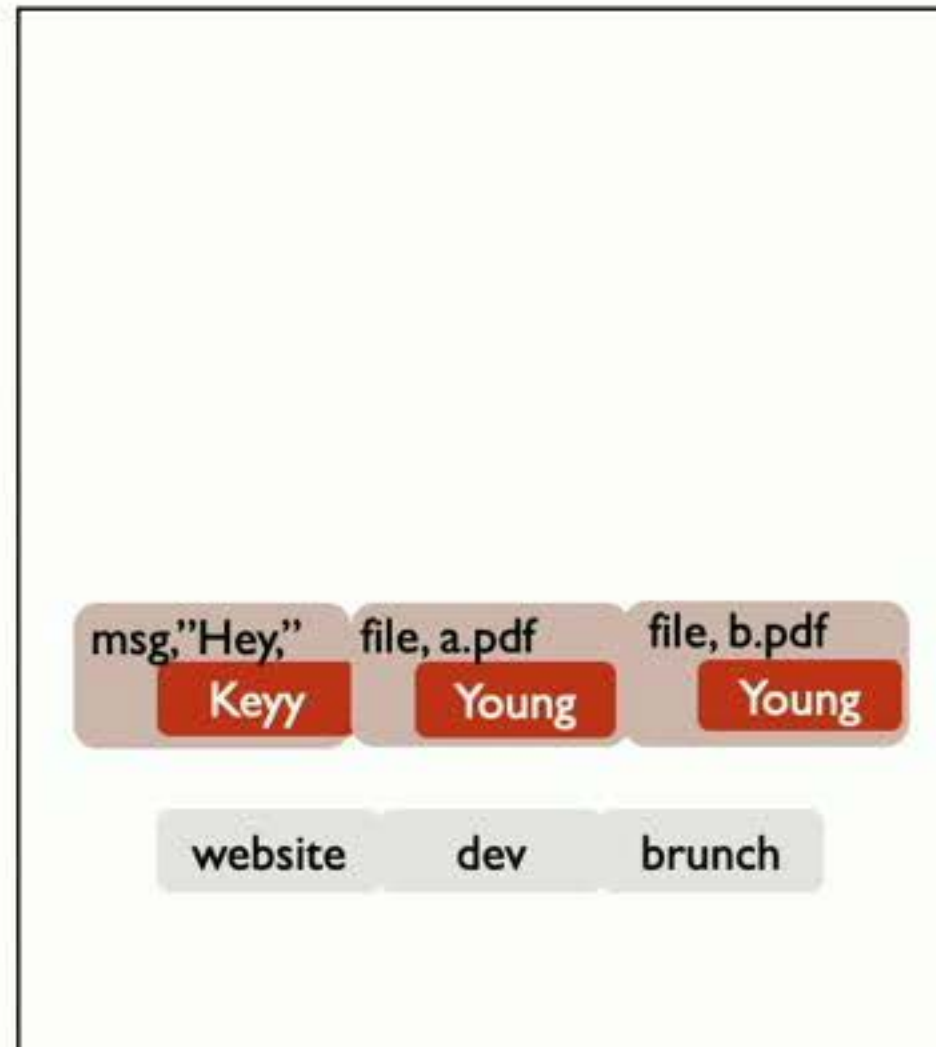
layout 3

...

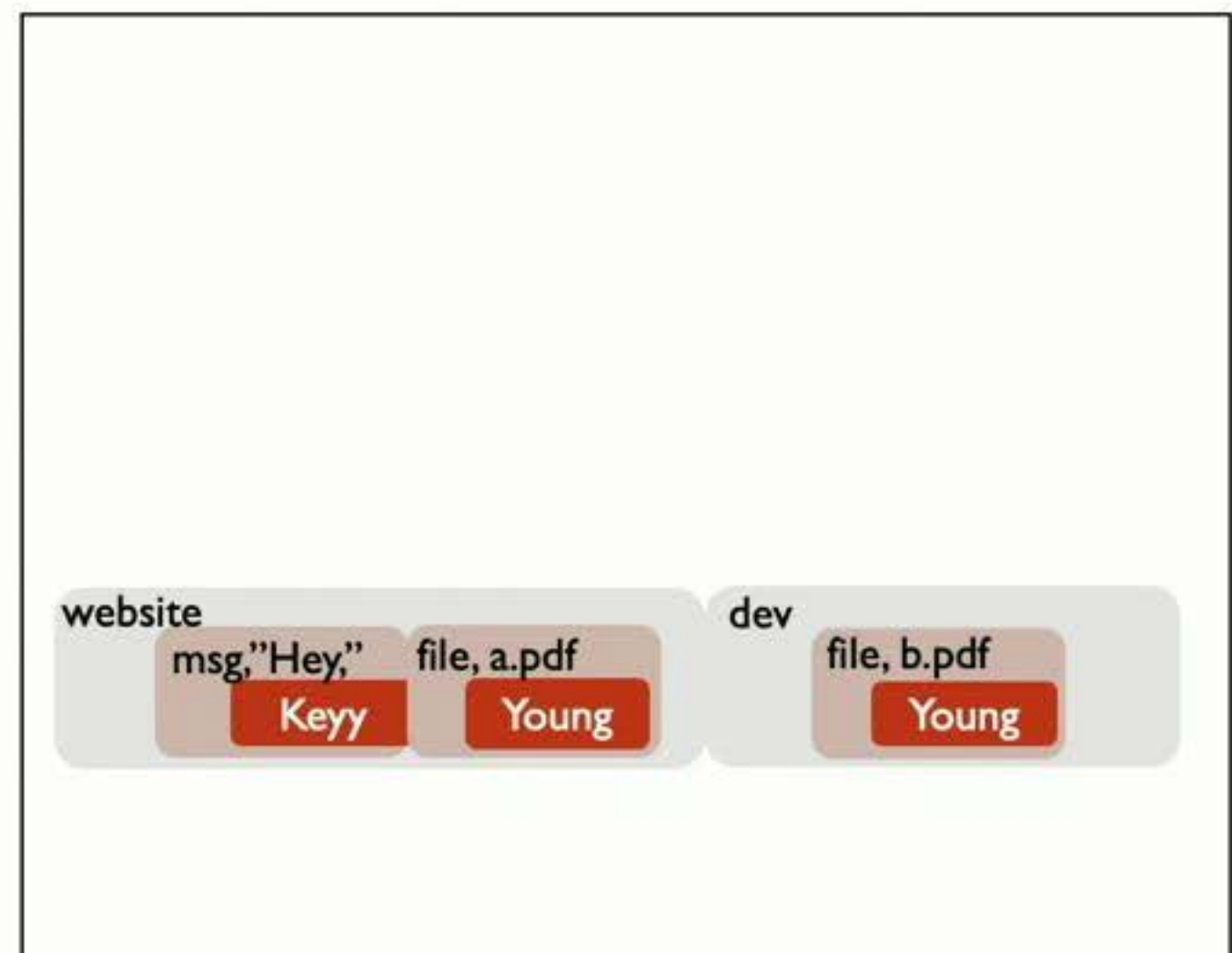
Chestnut Step 1: Layout Enumeration



layout 1



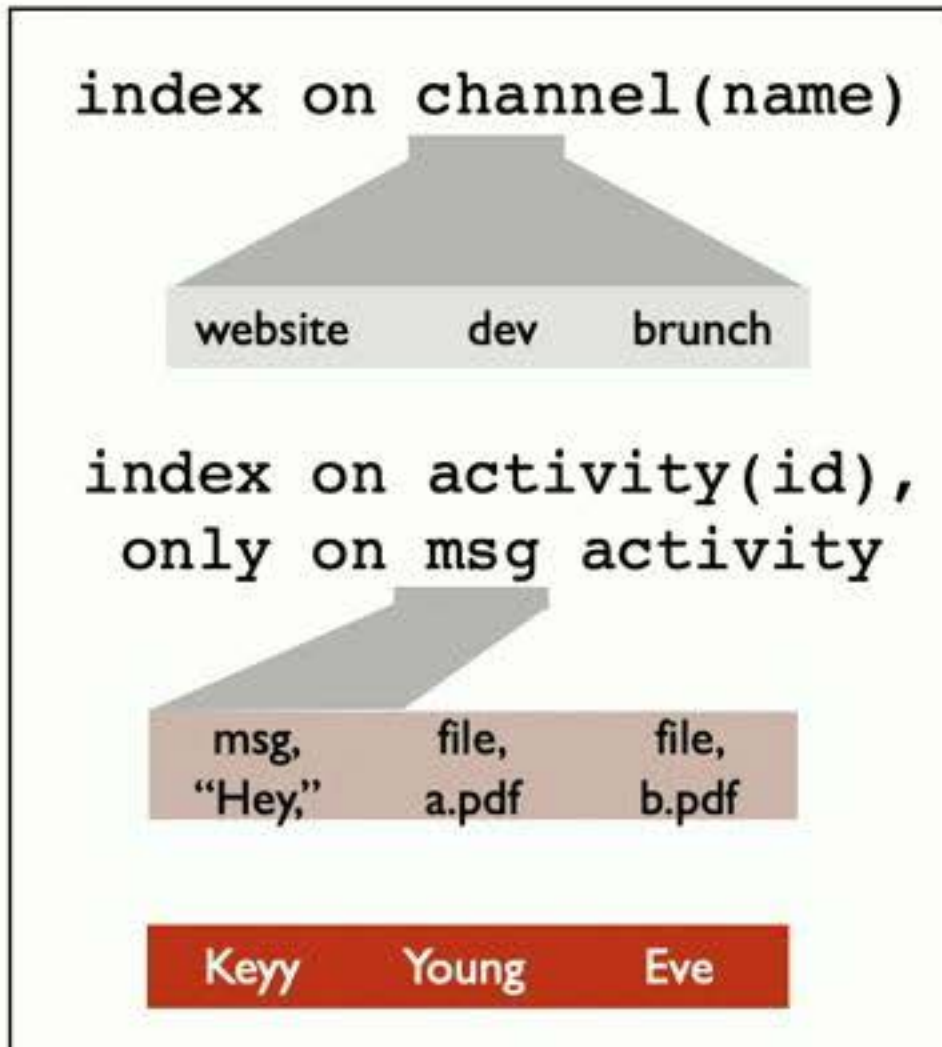
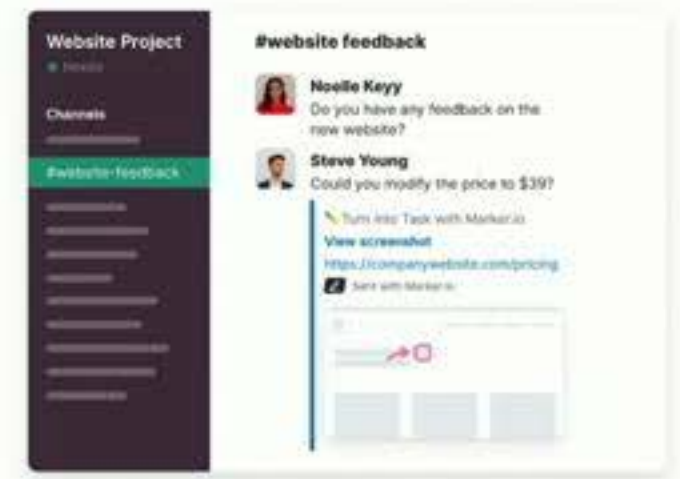
layout 2



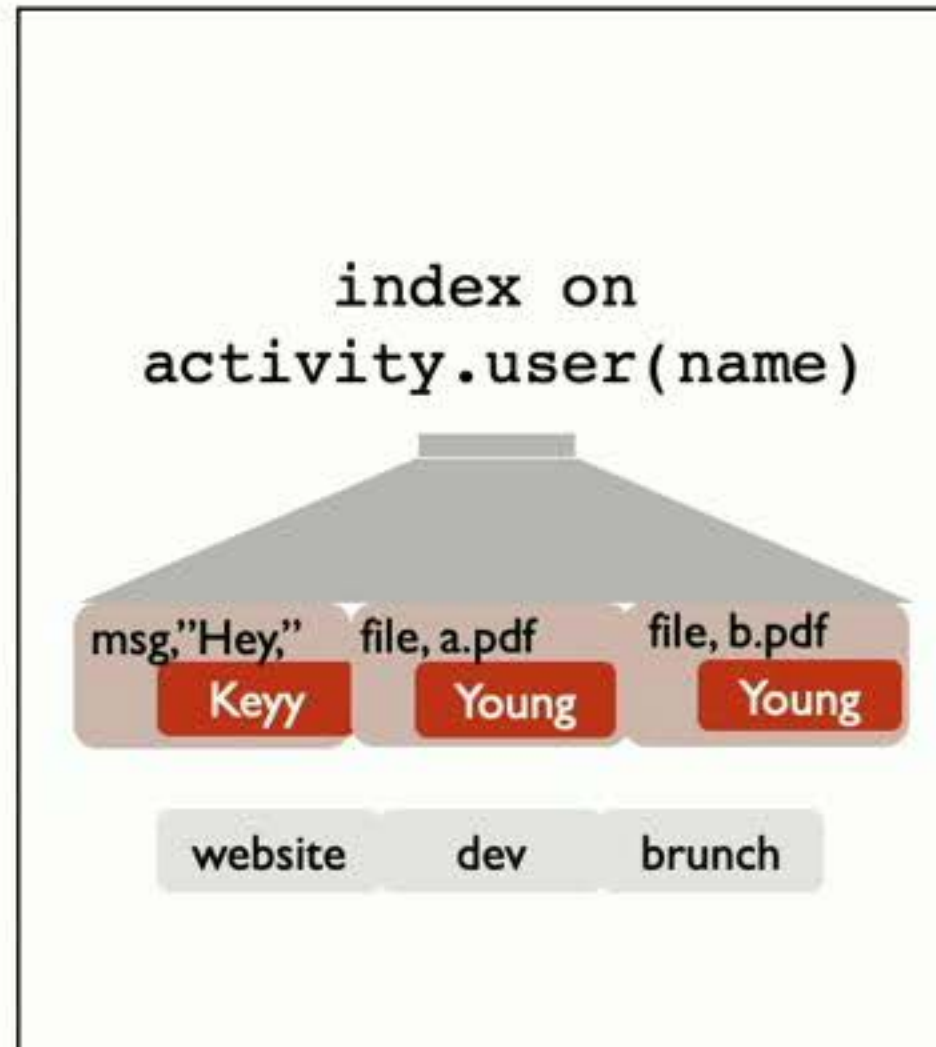
layout 3

...

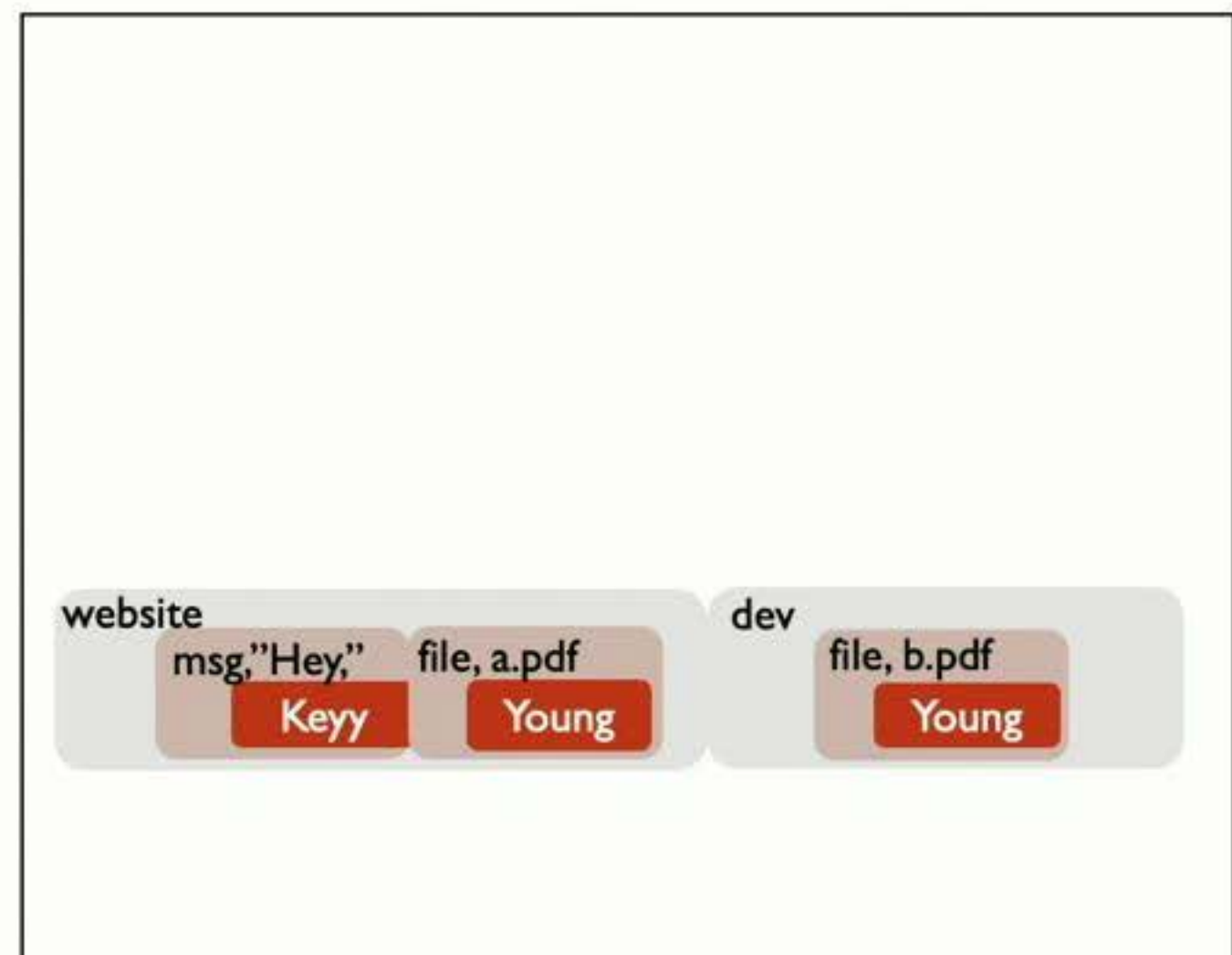
Chestnut Step 1: Layout Enumeration



layout 1



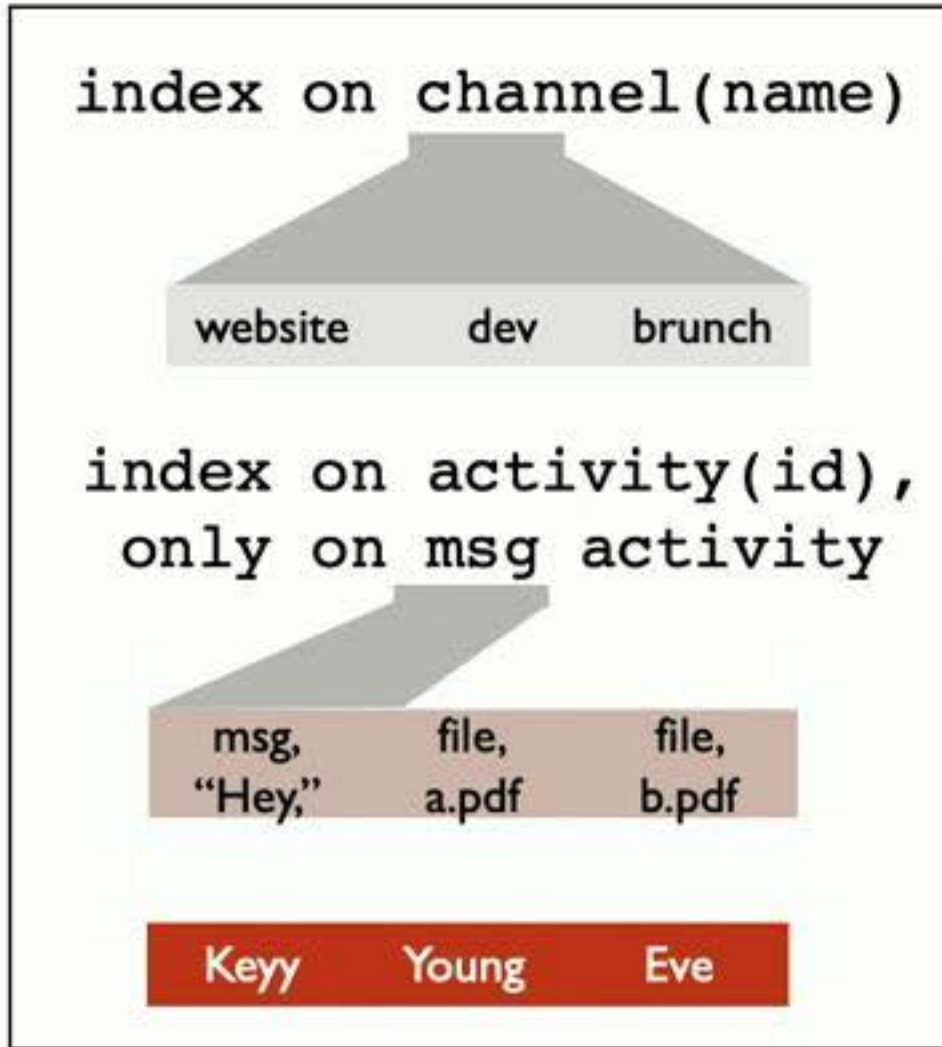
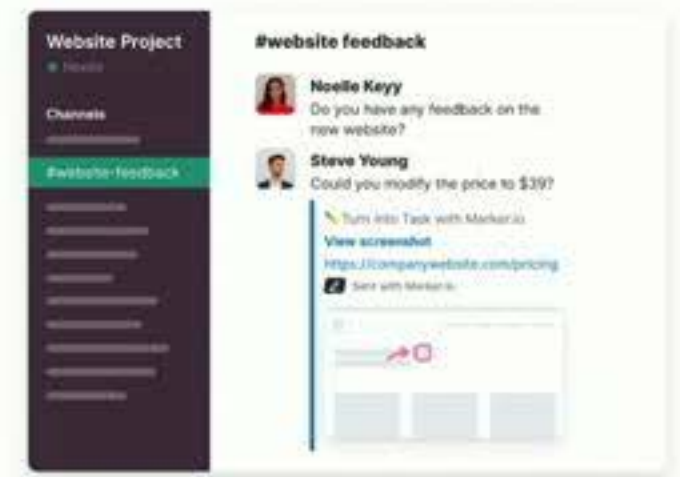
layout 2



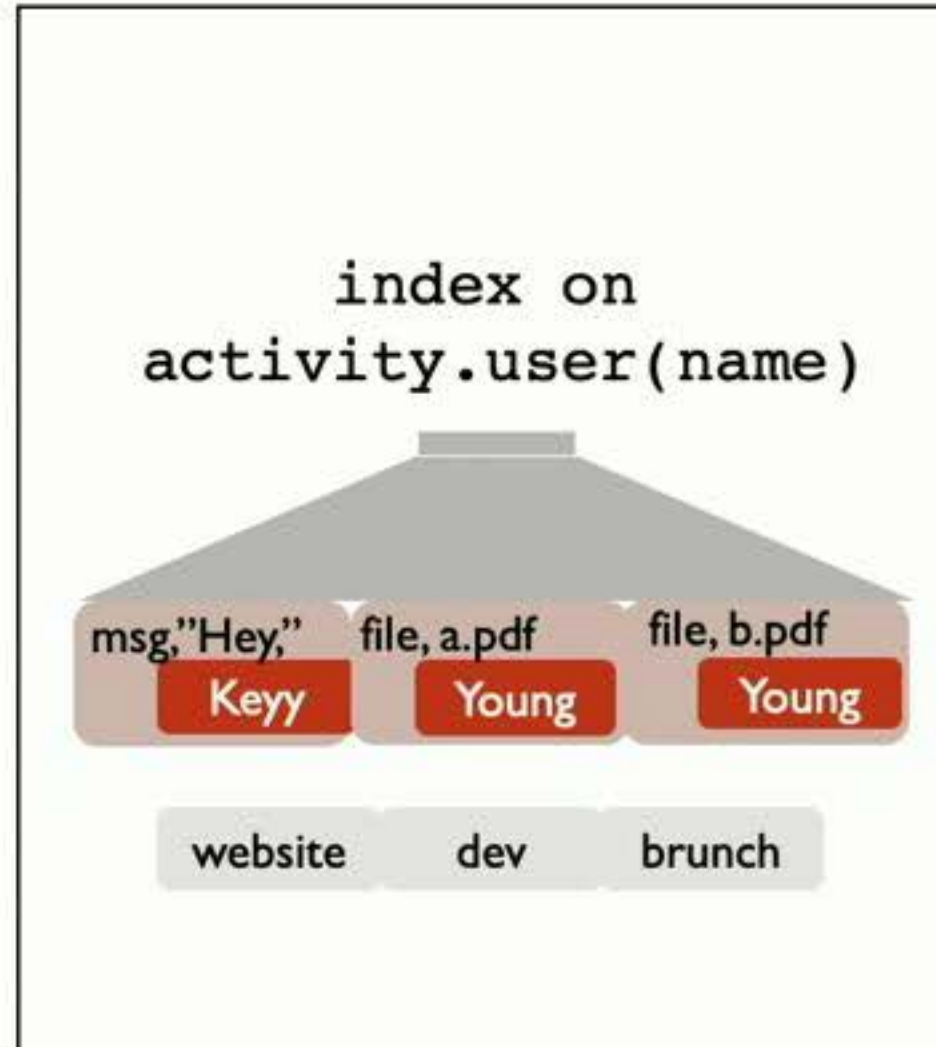
layout 3

...

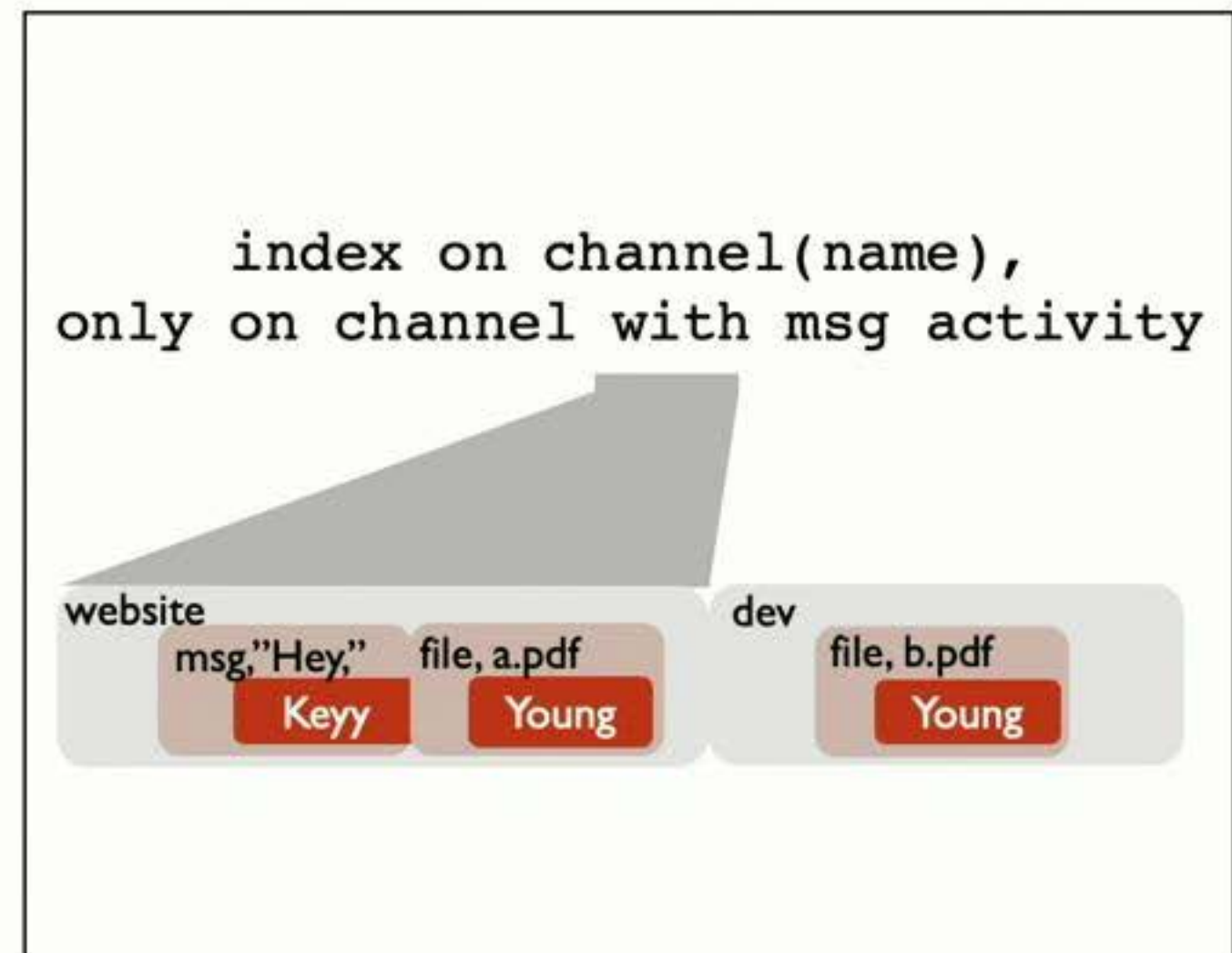
Chestnut Step 1: Layout Enumeration



layout 1



layout 2



layout 3

...

Chestnut Step 2: Plan Enumeration

Data layout:



Expected query result:

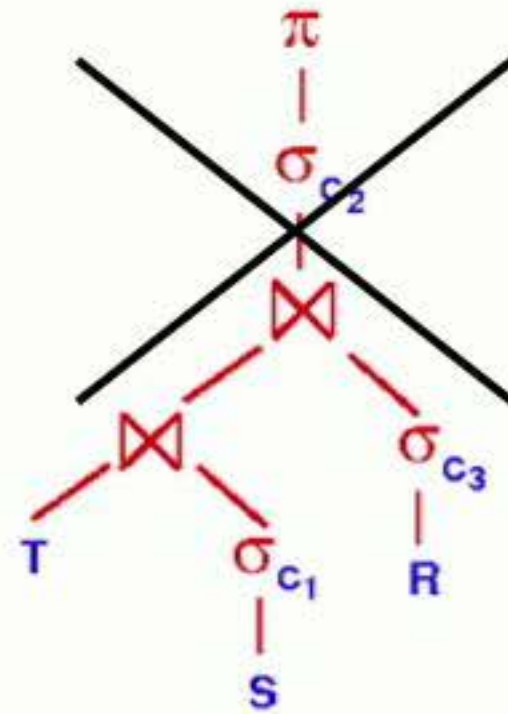


Chestnut Step 2: Plan Enumeration

Data layout:



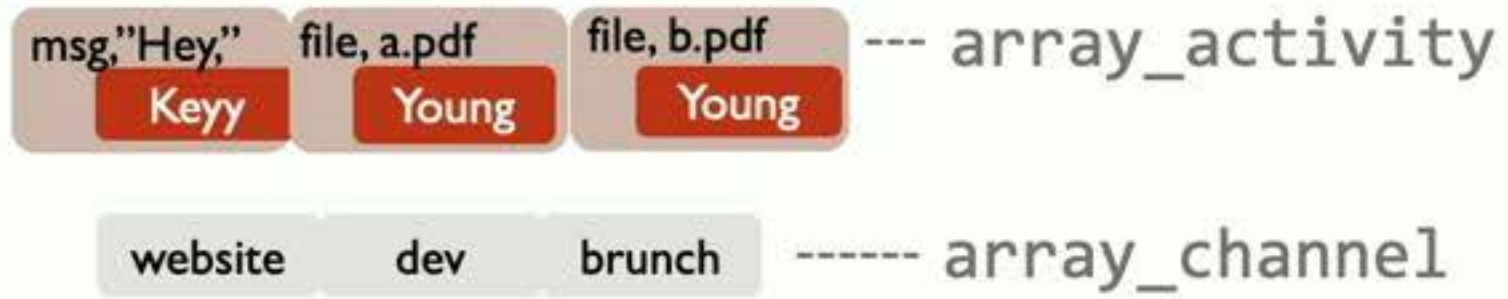
Expected query result:



relational query plans
cannot be applied

Chestnut Step 2: Plan Enumeration

Data layout:



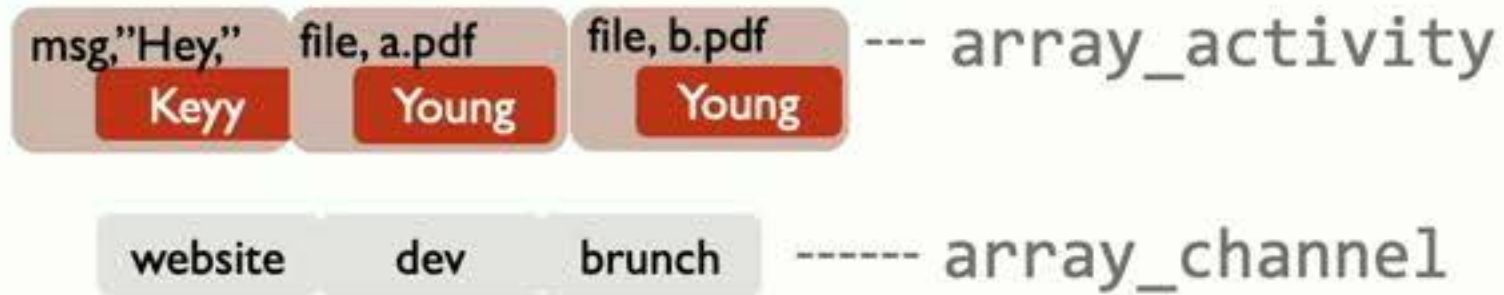
Expected query result:



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


Chestnut Step 2: Plan Enumeration

Data layout:



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



Expected query result:

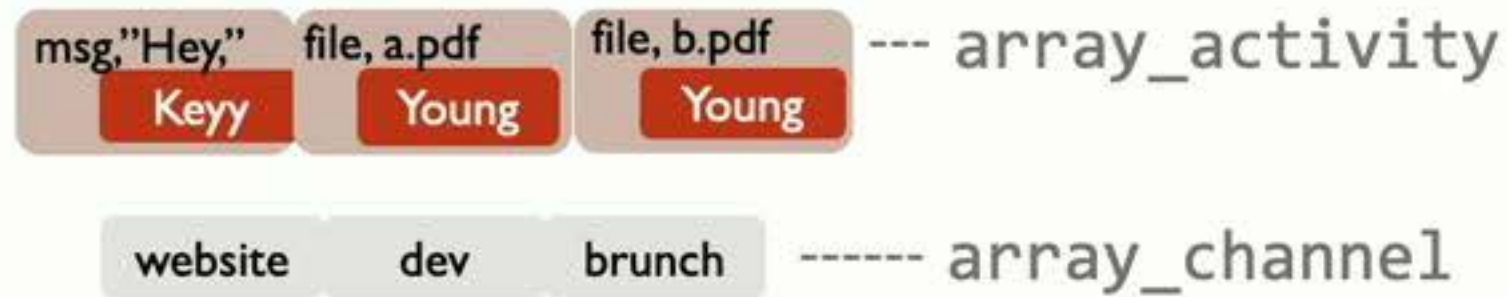


```
for c in array_channel:  
    for a in array_activity:  
        if a.channel_id==c.id:  
            c.activities.add(a)  
    result.append(c)
```

symbolic
verification

Chestnut Step 2: Plan Enumeration

Data layout:



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



Expected query result:



≠

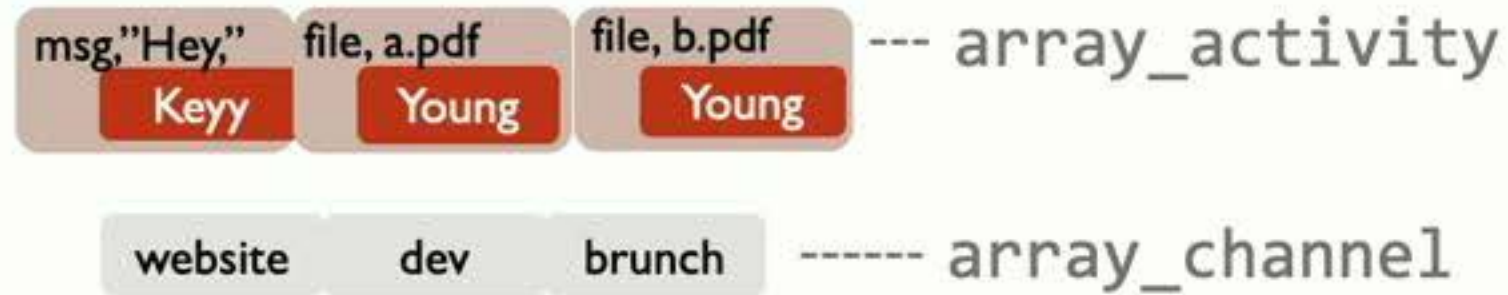
symbolic
verification

```
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:



```
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)
```



Expected query result:

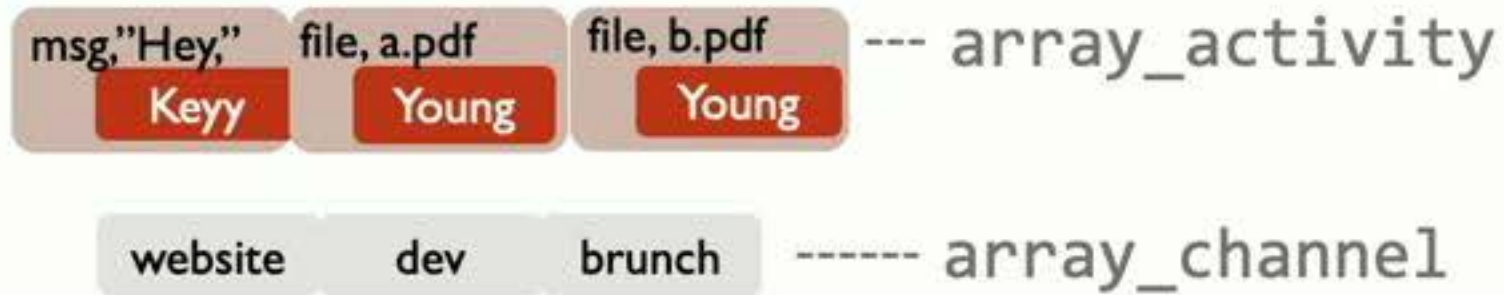


```
for c in array_channel:  
    if c.name == 'website':  
        for a in array_activity:  
            if a.channel_id==c.id:  
                c.activities.add(a)  
    result.append(c)
```

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

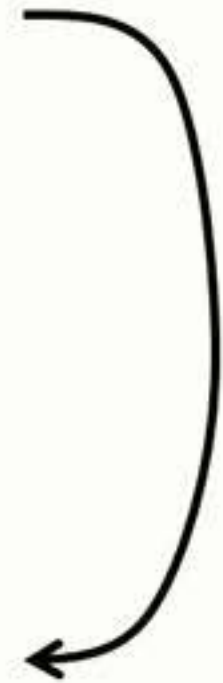
write query:

```
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

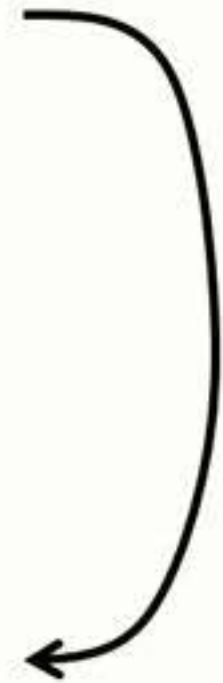
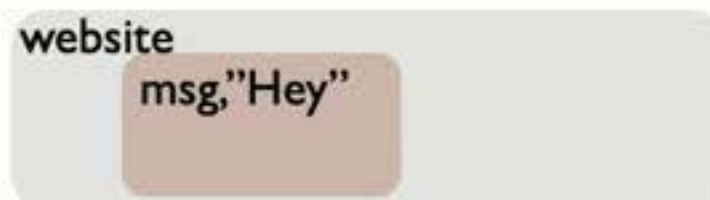
write query:

```
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

Q1

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

Q2

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

⋮



Chestnut Step 4: Handling Multiple Queries

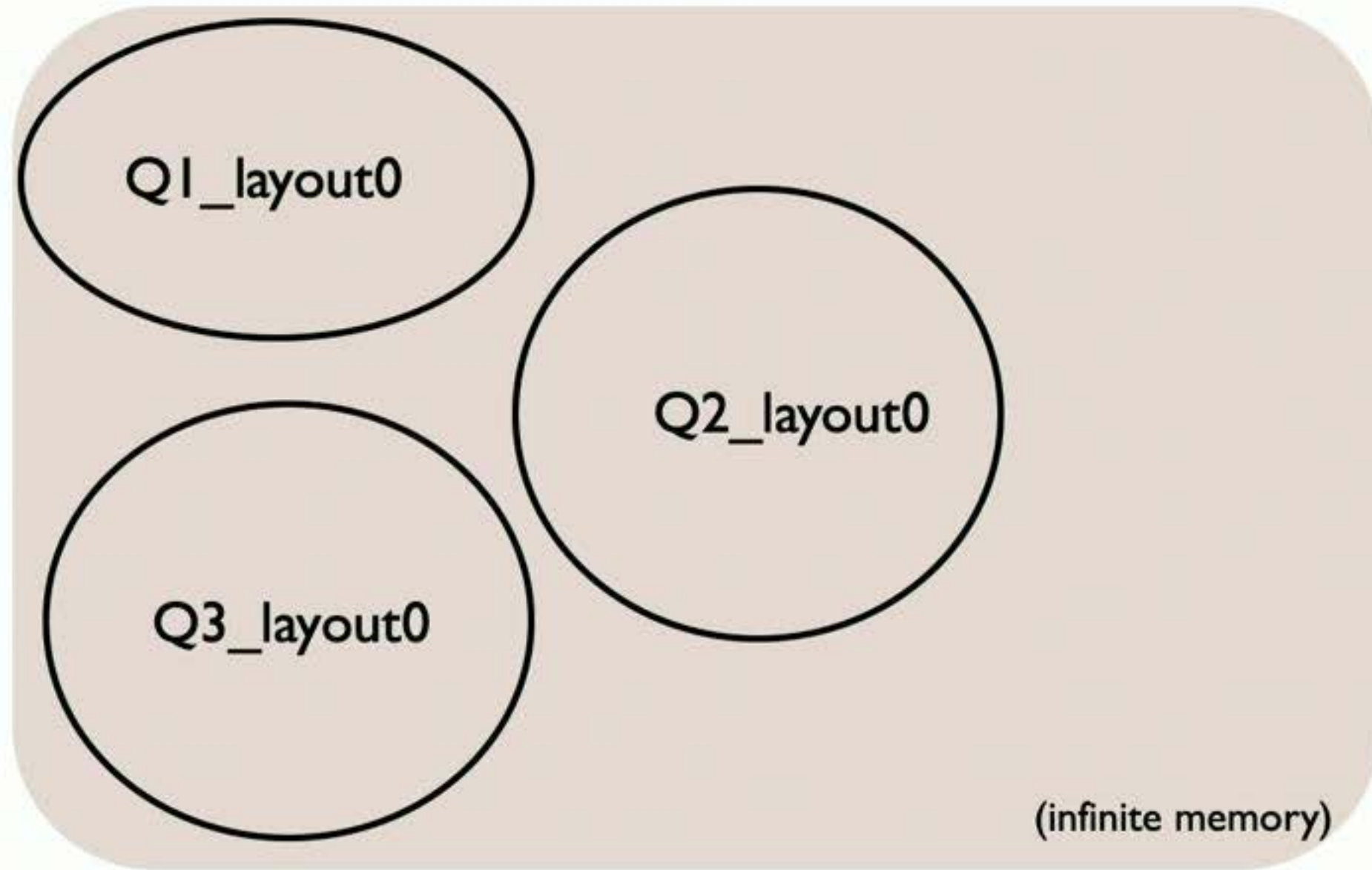
Q1

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

Q2

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

⋮



Chestnut Step 4: Handling Multiple Queries

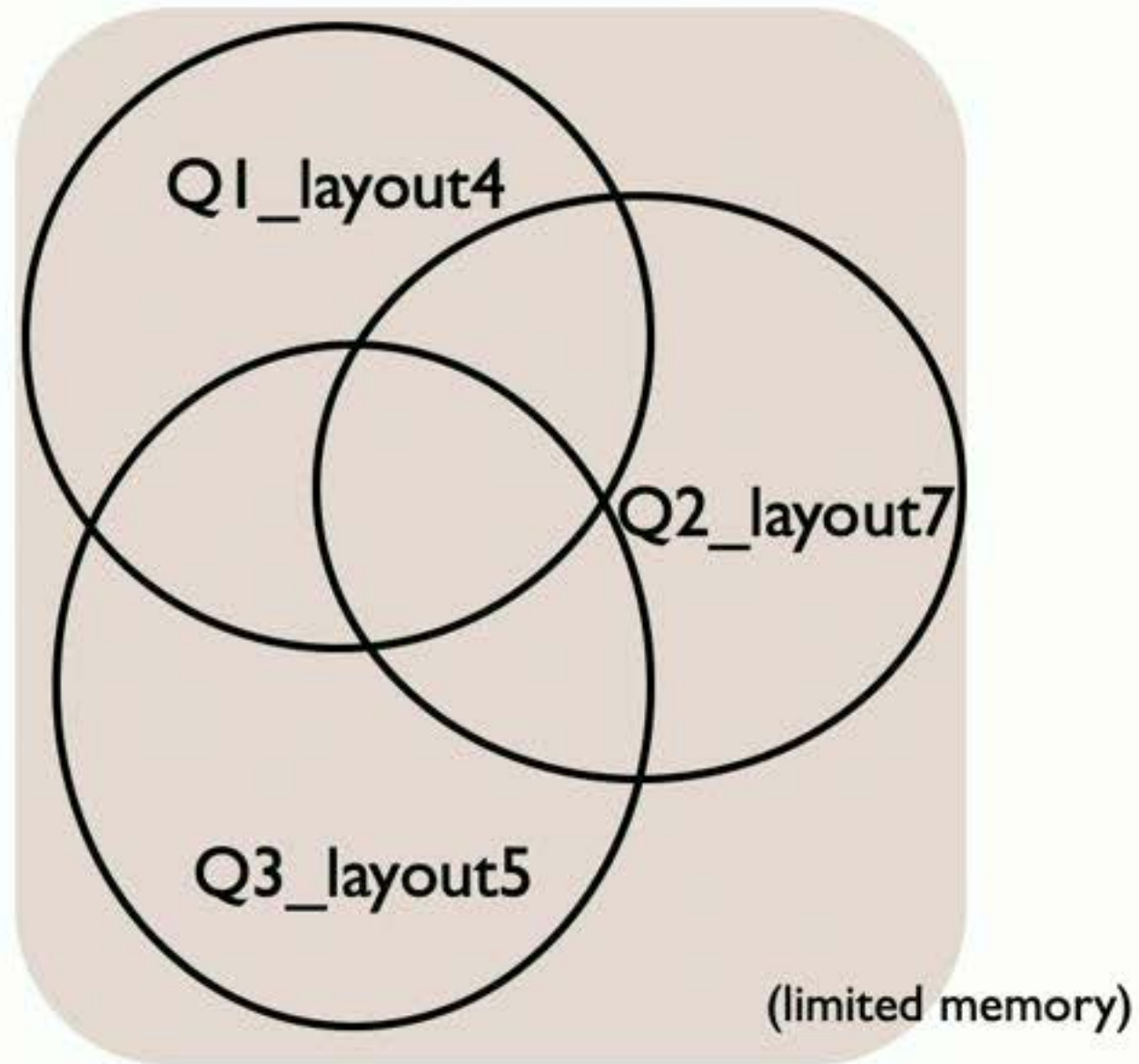
Q1

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

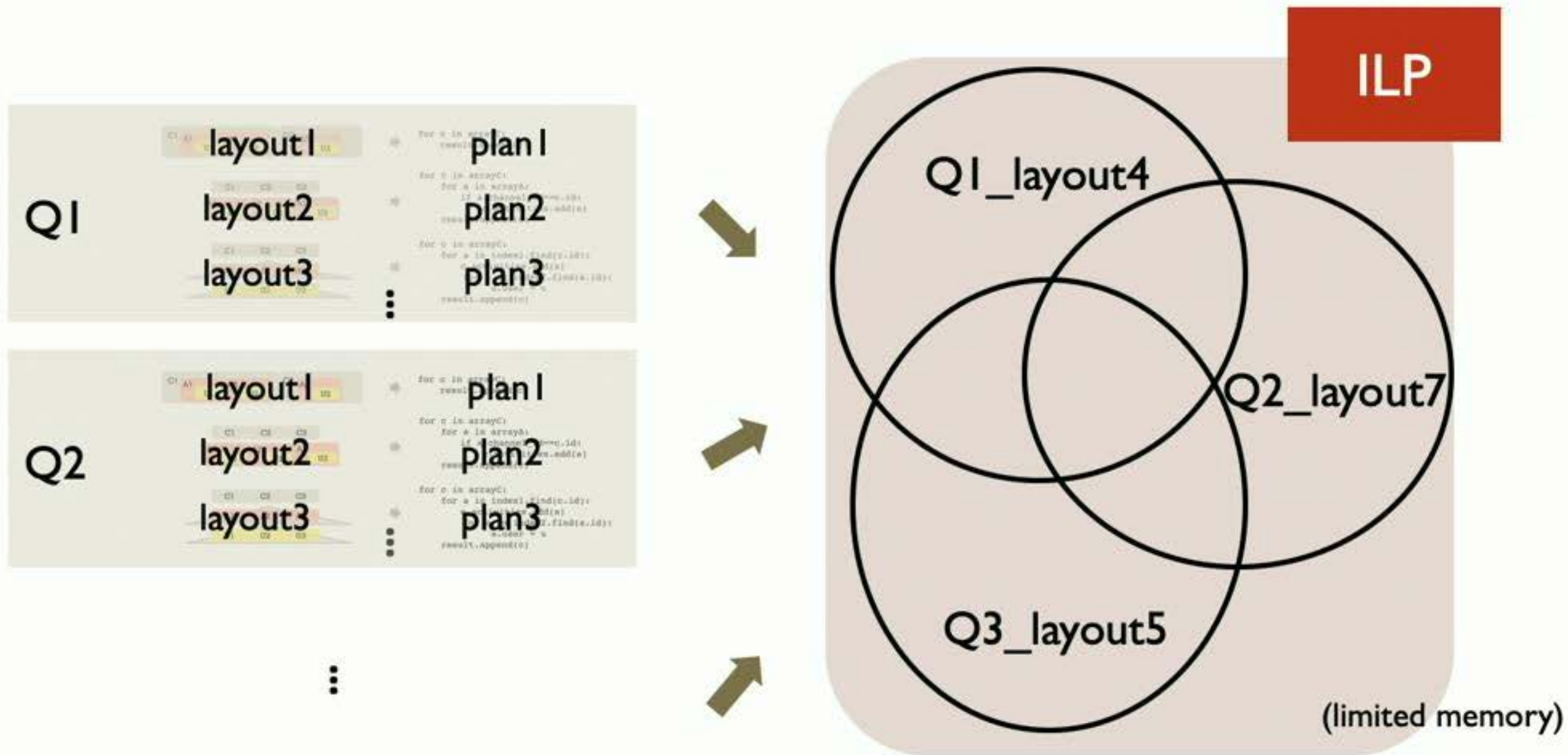
Q2

layout1	plan1
layout2	plan2
layout3	plan3
⋮	⋮

⋮



Chestnut Step 4: Handling Multiple Queries



Chestnut Workflow



Constraint:

- Used data structures within memory bound

Optimization goal:

- Minimize \sum *estimated query time*

Chestnut Workflow



ILP

Output:

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

Chestnut Workflow



Constraint:

- Used data structures within memory bound

Optimization goal:

- Minimize \sum *estimated query time*

Chestnut Step 3: Update Query

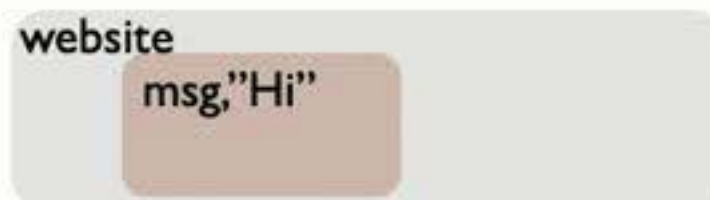
write query:

```
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

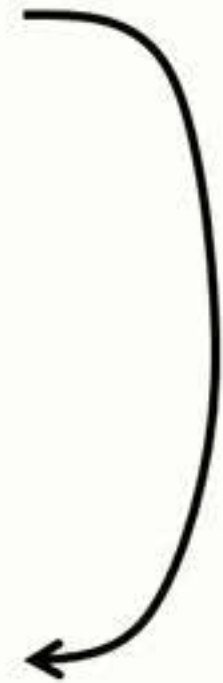
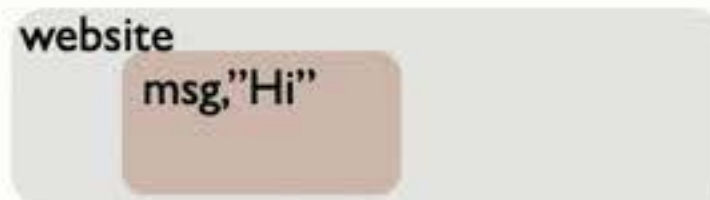
write query:

```
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



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

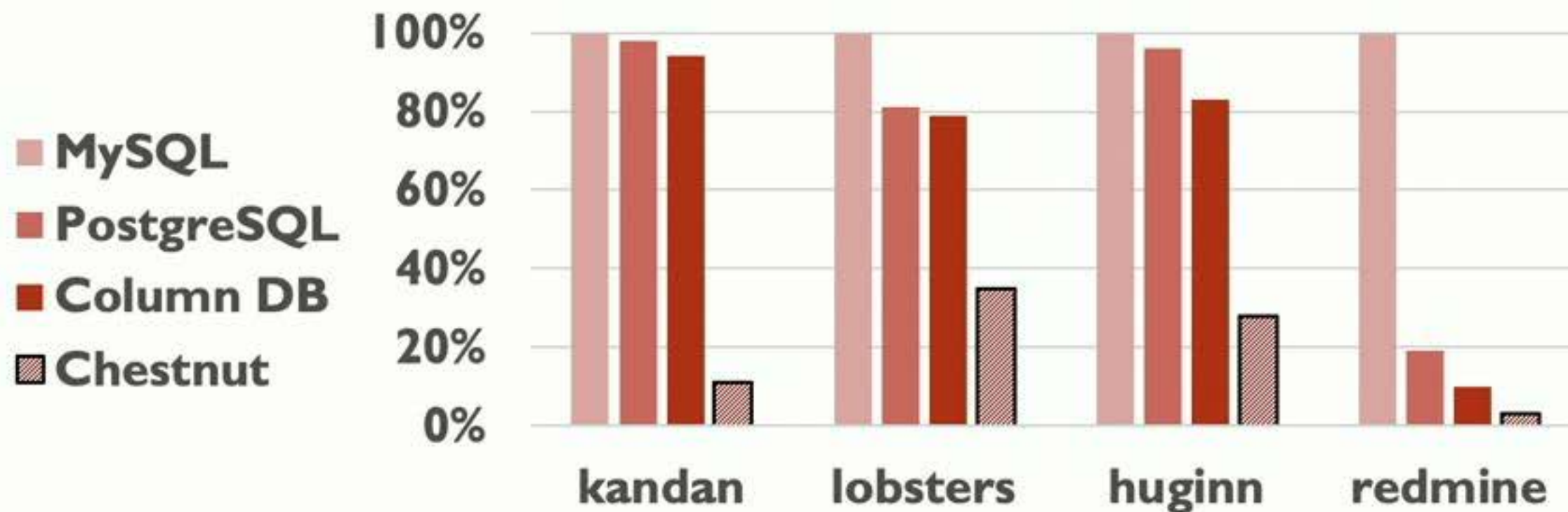
- **4 open-source popular web applications built with Rails**
 - kandan: chatting app (2.8k ★)
 - redmine: project management app (3.6k ★)
 - lobsters: forum app (2.4k ★)
 - huginn: web-scraping app (22.6k ★)

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

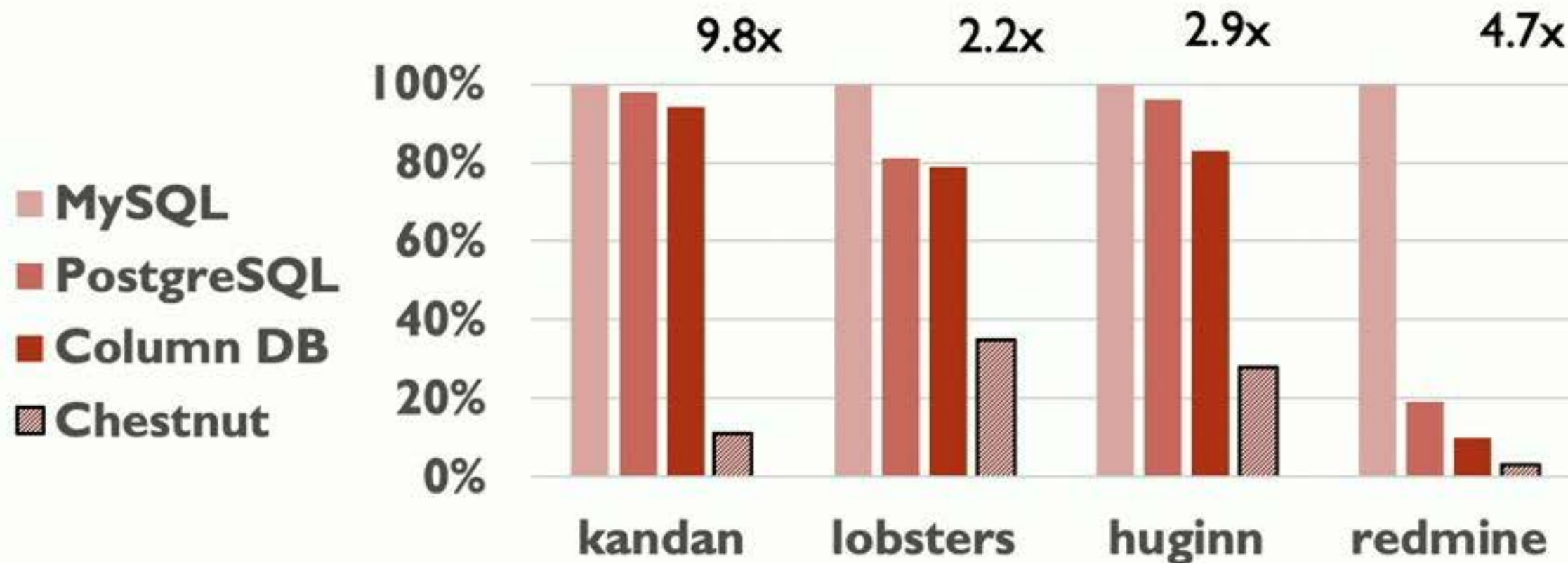


(avg query time, with the same memory)

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

Evaluation: Web Applications

- 4 open-source popular web applications built with Rails

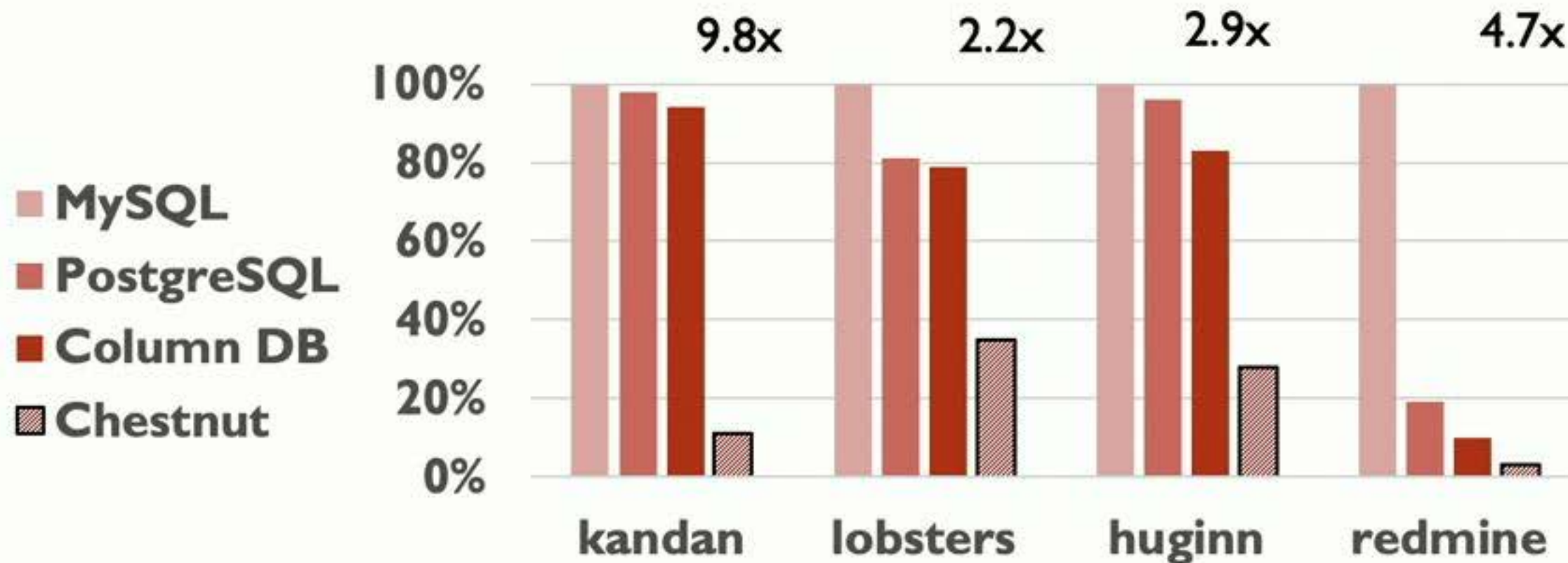


(avg query time, with the same memory)

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

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

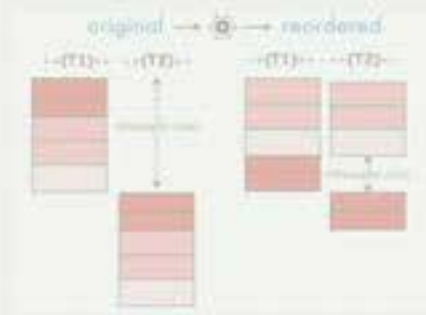
Panorama:
view-driven optimization



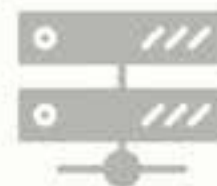
The diagram shows a 'slow webpage' on the left with a search icon and a 'fast webpage' on the right with a gear icon. A central gear icon indicates the optimization process. The slow webpage has a complex layout with many elements, while the fast webpage is simplified and more focused.



Quro:
reorder queries



The diagram shows 'original' and 'reordered' query execution plans. The original plan has a sequence of operations (T1) and (T2) that are inefficient. The reordered plan rearranges these operations to optimize performance. The operations are represented as red blocks in a tree structure.



Chestnut:
customize data layout



The diagram shows a 'tabular layout' on the left and a 'customized nested layout' on the right. A central gear icon indicates the customization process. The tabular layout is a simple grid, while the customized nested layout is more complex and tailored to the data.



- 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? 

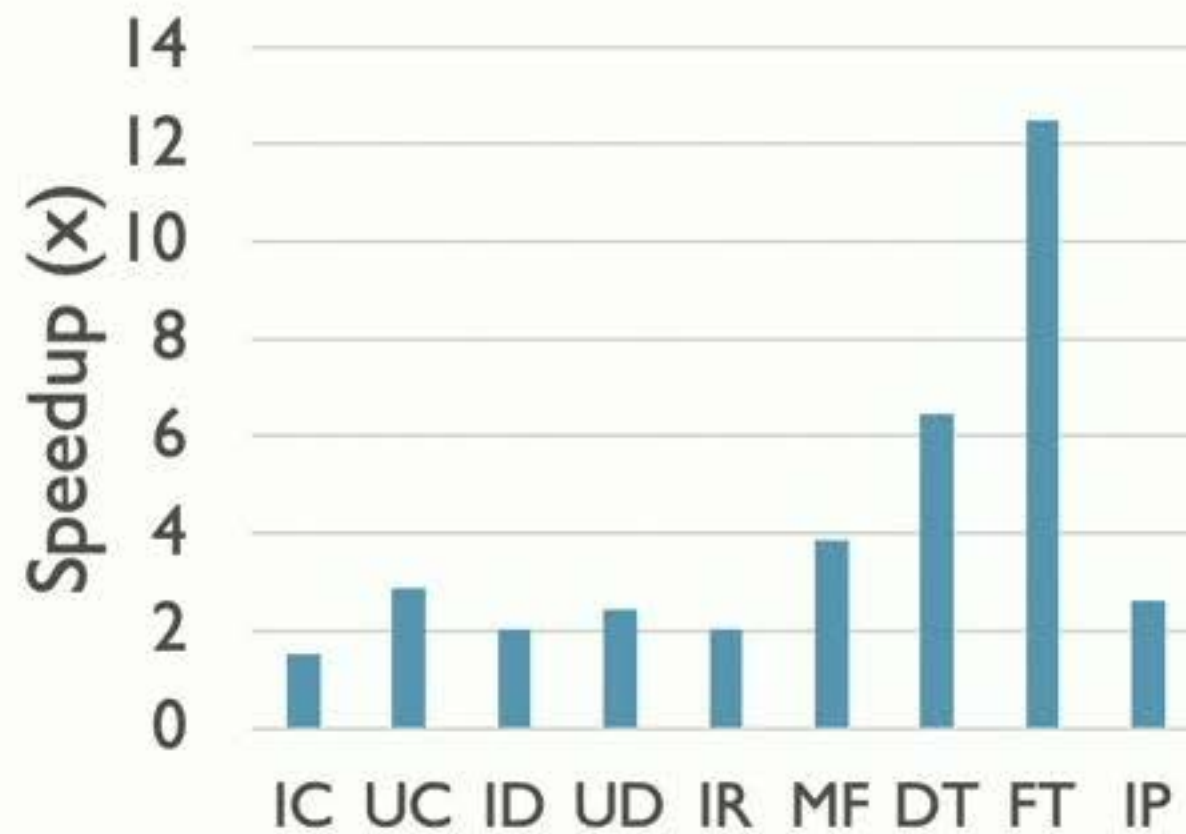
select * from... 

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]



9 anti-patterns

Automatic Fix of Anti-patterns

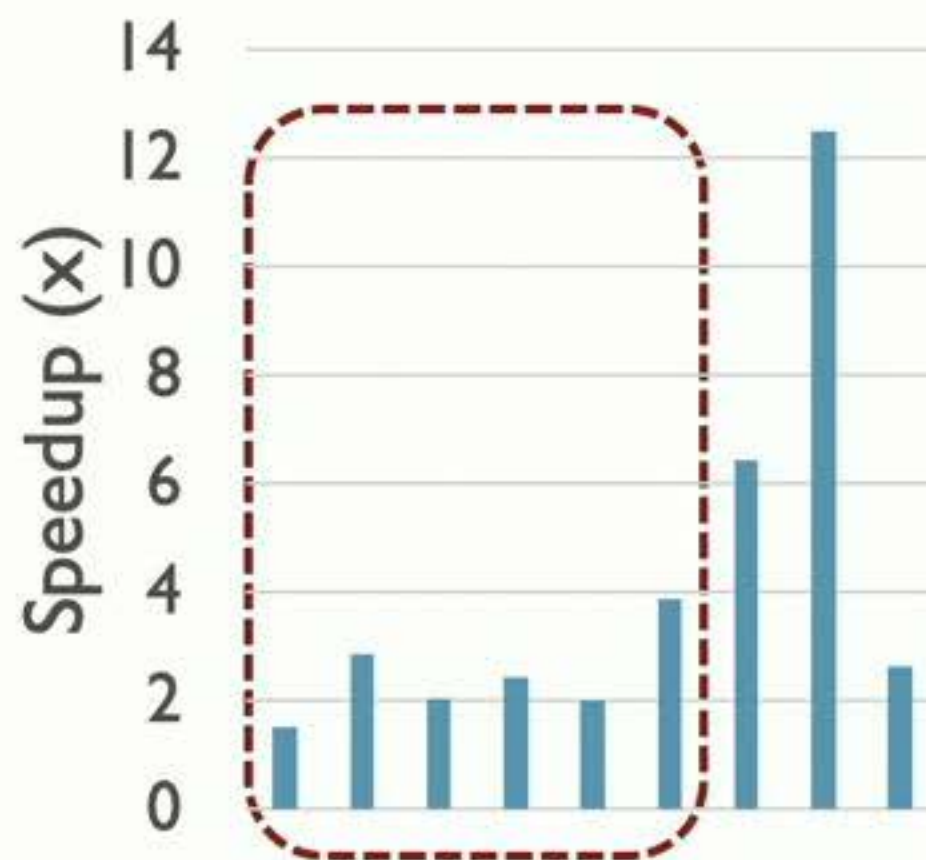
Powerstation

[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns



Automatic Fix of Anti-patterns

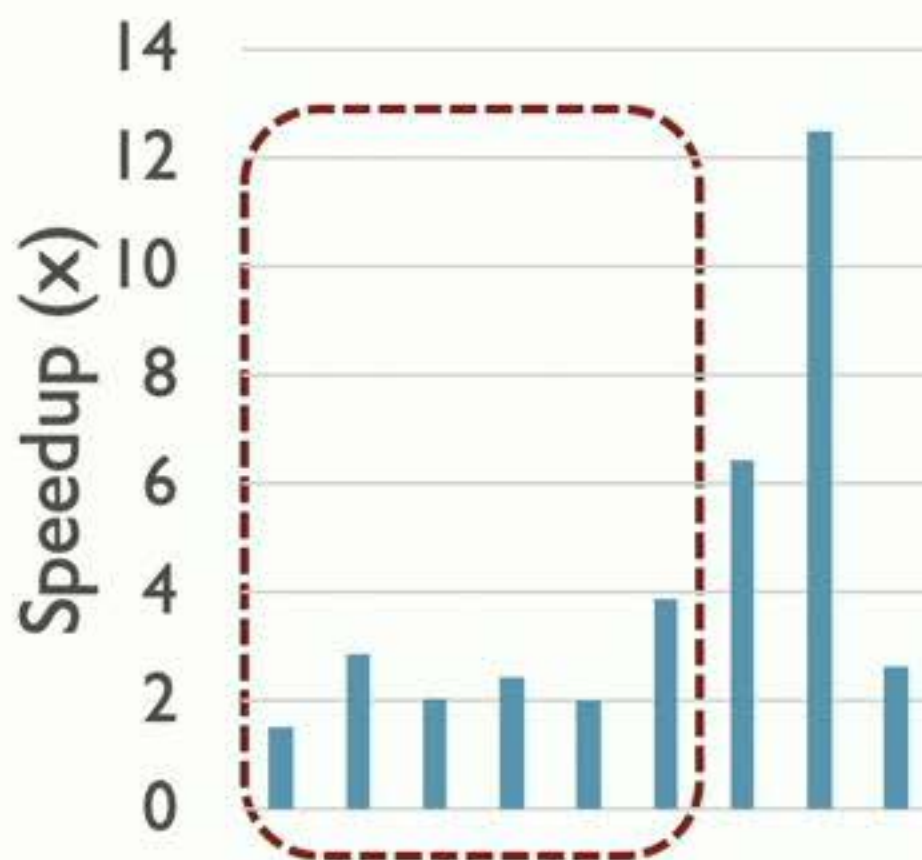
Powerstation

[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

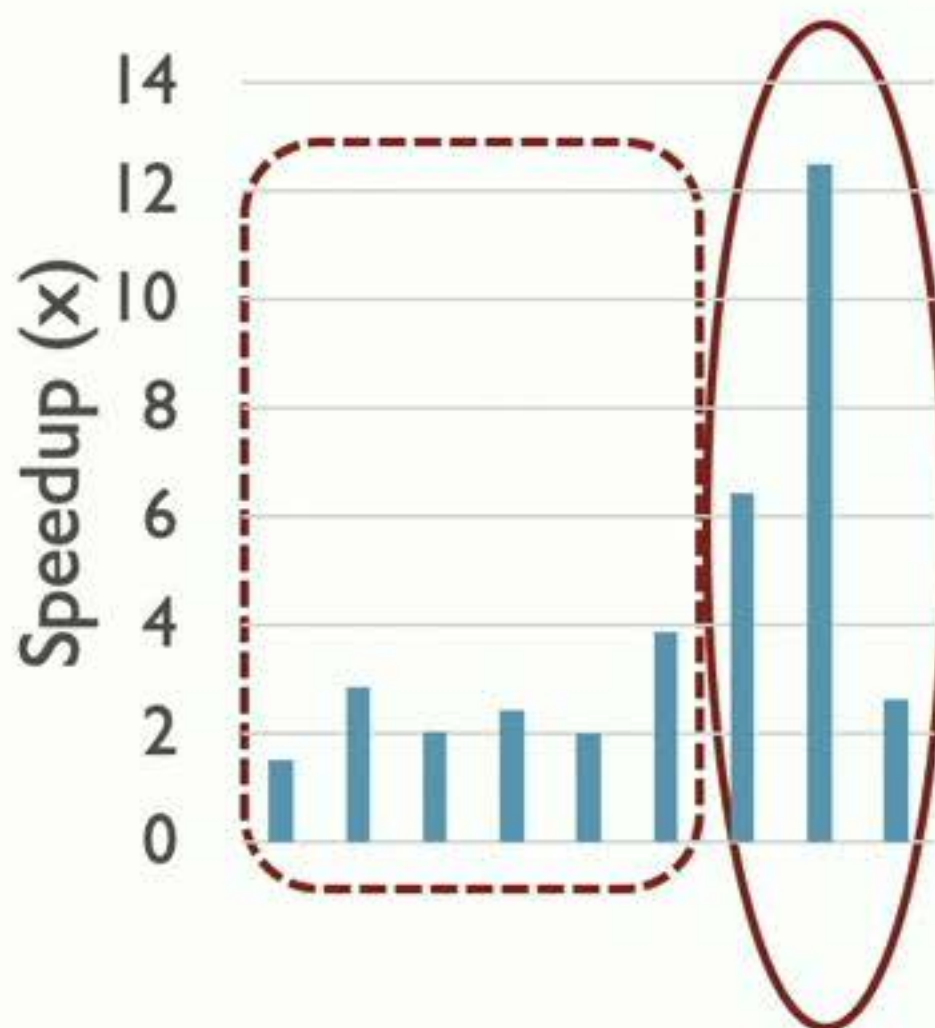
[CIKM'17, FSE'18]

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

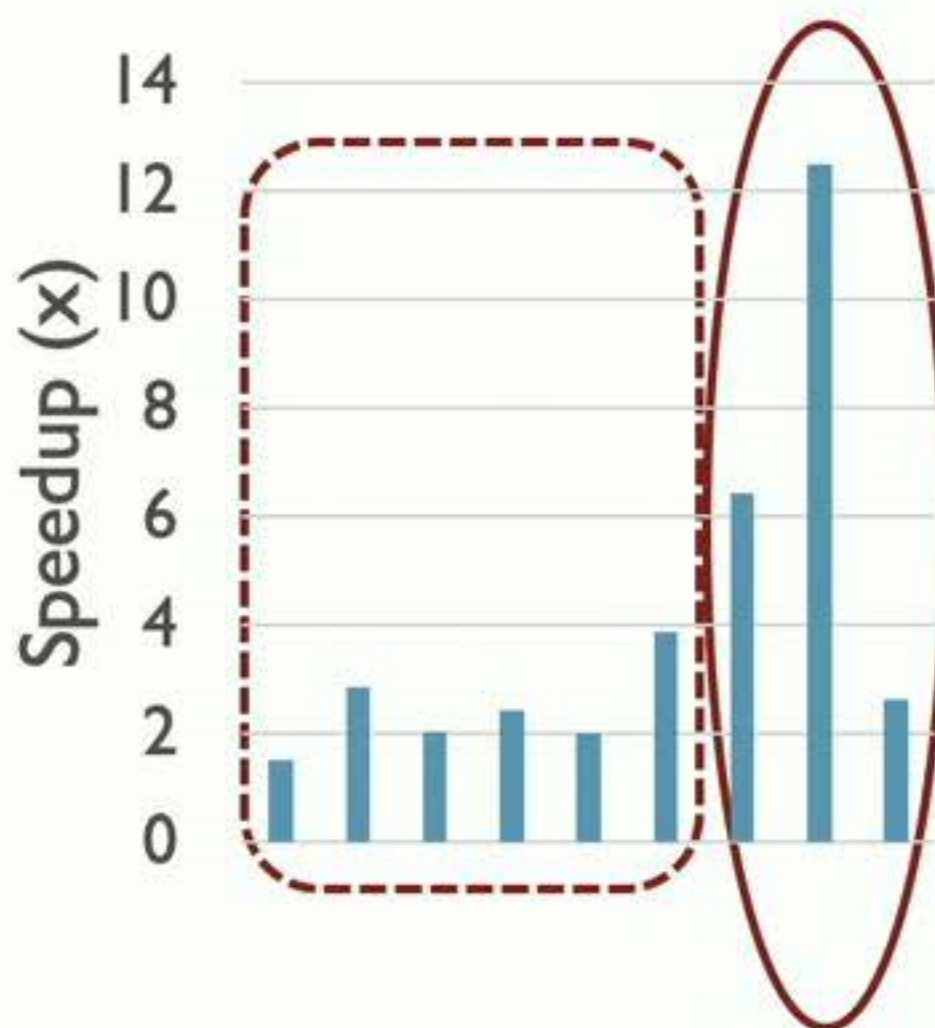
[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns

Semantic-preserving!



Non-semantic-preserving!

**Webpage
design-performance
tradeoff**

Automatic Fix of Anti-patterns

Powerstation

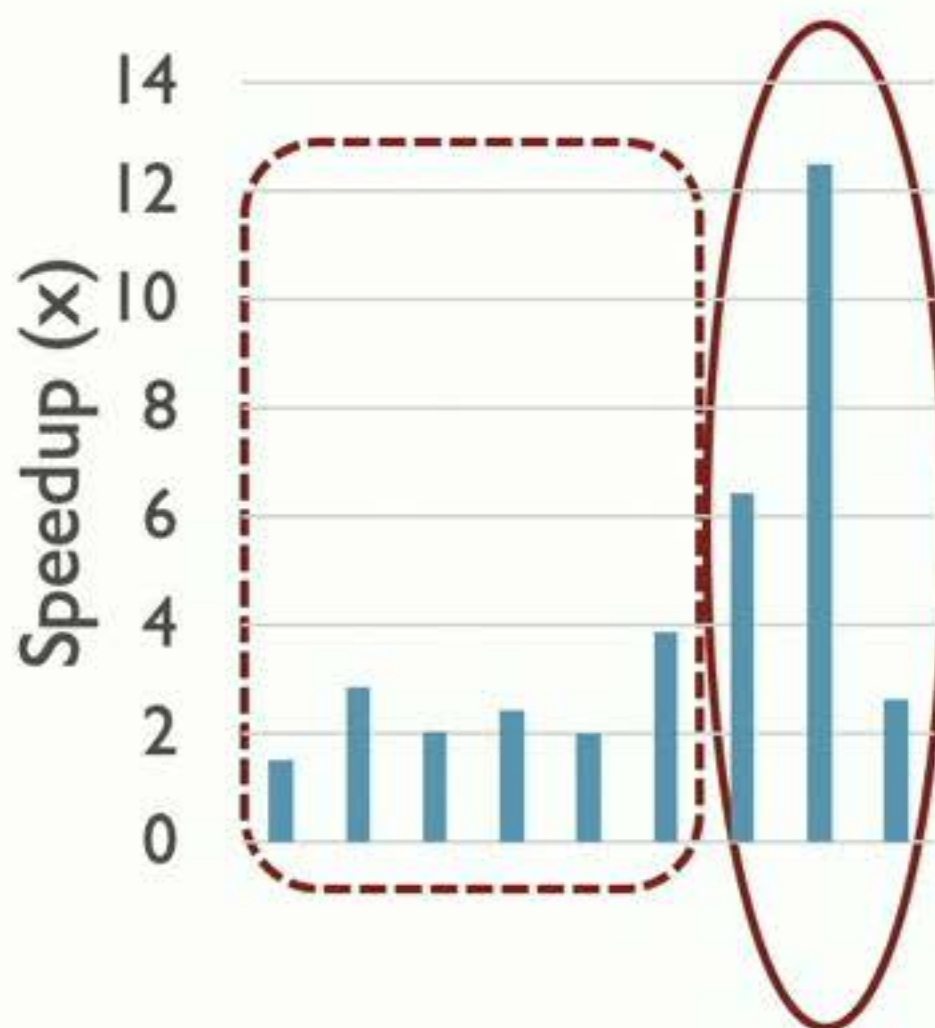
[CIKM'17, FSE'18]

It fixes 6 anti-patterns

Static analysis on the source code

Pattern-match to find anti-patterns

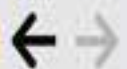
Semantic-preserving!



Panorama


Non-semantic-preserving!

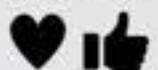
**Webpage
design-performance
tradeoff**




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!



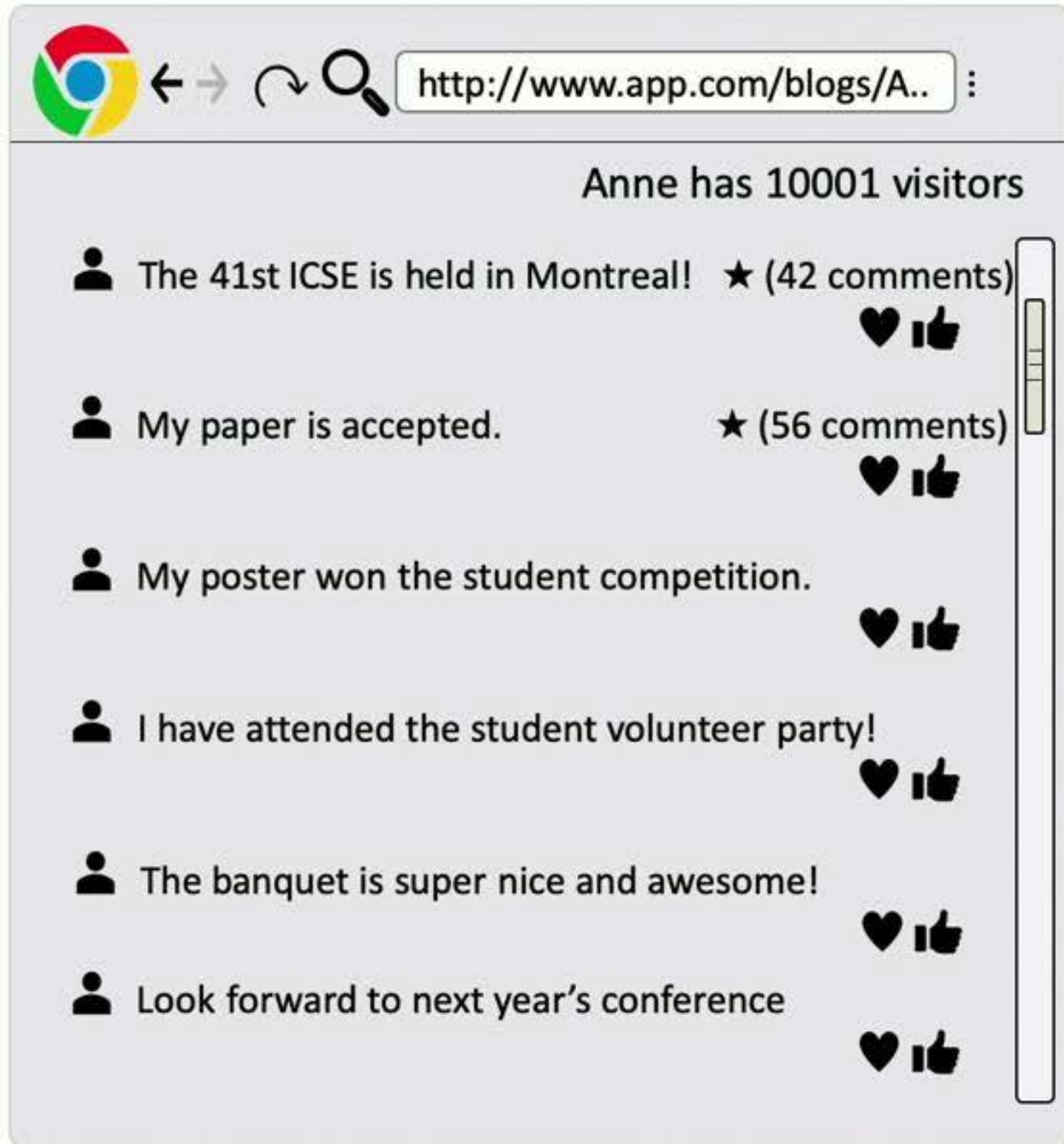
 The banquet is super nice and awesome!



 Look forward to next year's conference



12 sec



The screenshot shows a web browser window with the following elements:

- Address Bar:** Contains the URL `http://www.app.com/blogs/A..` and navigation icons (back, forward, refresh, search).
- Page Header:** Displays the text "Anne has 10001 visitors".
- Post List:** A vertical list of seven posts, each starting with a person icon. The first two posts include comment counts. Each post has a heart icon and a thumbs-up icon to its right.
- Scrollbar:** A vertical scrollbar is located on the right side of the post list.

Post 1: The 41st ICSE is held in Montreal! ★ (42 comments)

Post 2: My paper is accepted. ★ (56 comments)

Post 3: My poster won the student competition.





Post 4: I have attended the student volunteer party!

Post 5: The banquet is super nice and awesome!














Post 6: Look forward to next year's conference




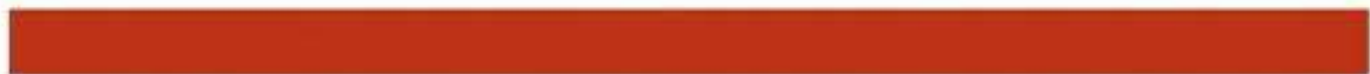
12 sec

     :

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



4.8 sec

Chrome browser interface showing a list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

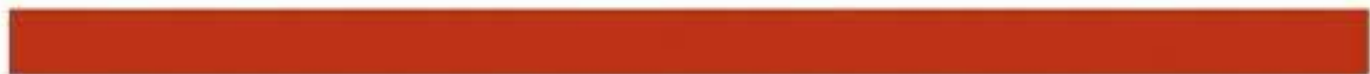
- The 41st ICSE is held in Montreal! ★ (42 comments)
- My paper is accepted. ★ (56 comments)
- My poster won the student competition.
- I have attended the student volunteer party!
- The banquet is super nice and awesome!
- Look forward to next year's conference

Each post has a heart icon and a thumbs-up icon. A vertical scrollbar is visible on the right side of the list, highlighted with a red box.

Chrome browser interface showing the same list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

- 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!

Each post has a heart icon and a thumbs-up icon. At the bottom of the page, a pagination control is visible, highlighted with a red box, containing buttons for '<<', '1', '2', '3', '...', and '>>'.



12 sec



4.8 sec

Chrome browser interface showing a list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

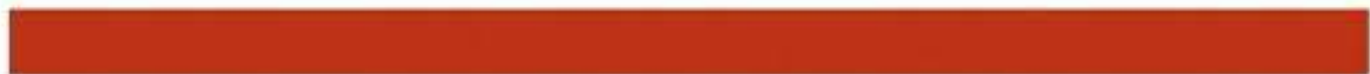
- The 41st ICSE is held in Montreal! ★ (42 comments)
- My paper is accepted. ★ (56 comments)
- My poster won the student competition.
- I have attended the student volunteer party!
- The banquet is super nice and awesome!
- Look forward to next year's conference

Each post has a heart icon and a thumbs-up icon. A red box highlights the first two posts and a vertical scrollbar on the right.

Chrome browser interface showing the same list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

- 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!

Each post has a heart icon and a thumbs-up icon. A red box highlights a pagination bar at the bottom with buttons for '<<', '1', '2', '3', '...', and '>>'.



12 sec



2.2 sec

:

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

:

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



2.2 sec

Chrome browser interface showing a list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

- 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

Each post has a heart icon and a thumbs-up icon. A vertical scrollbar is visible on the right side of the list. A red box highlights the first two posts and the scrollbar.

Chrome browser interface showing the same list of blog posts. The address bar contains 'http://www.app.com/blogs/A..'. The page title is 'Anne has 10001 visitors'. The list includes:

- 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!

Each post has a heart icon and a thumbs-up icon. At the bottom, a pagination control is visible with buttons for '<<', '1', '2', '3', '...', and '>>'. A red box highlights the pagination control.



12 sec



0.5 sec

Chrome browser interface showing a list of blog posts. The address bar contains `http://www.app.com/blogs/A..`. A red box highlights the text "Anne has 10001 visitors" above the first post. Another red box highlights the comment count "(42 comments)" and the like/dislike icons for the first post. A vertical scrollbar is visible on the right side of the list.

- 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

Chrome browser interface showing the same list of blog posts. A red box highlights a loading spinner icon in the top right corner. At the bottom, a red box highlights a pagination control with buttons for "<<", "1", "2", "3", "...", and ">>".

- 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!

Navigation: << 1 2 3 ... >>

Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers

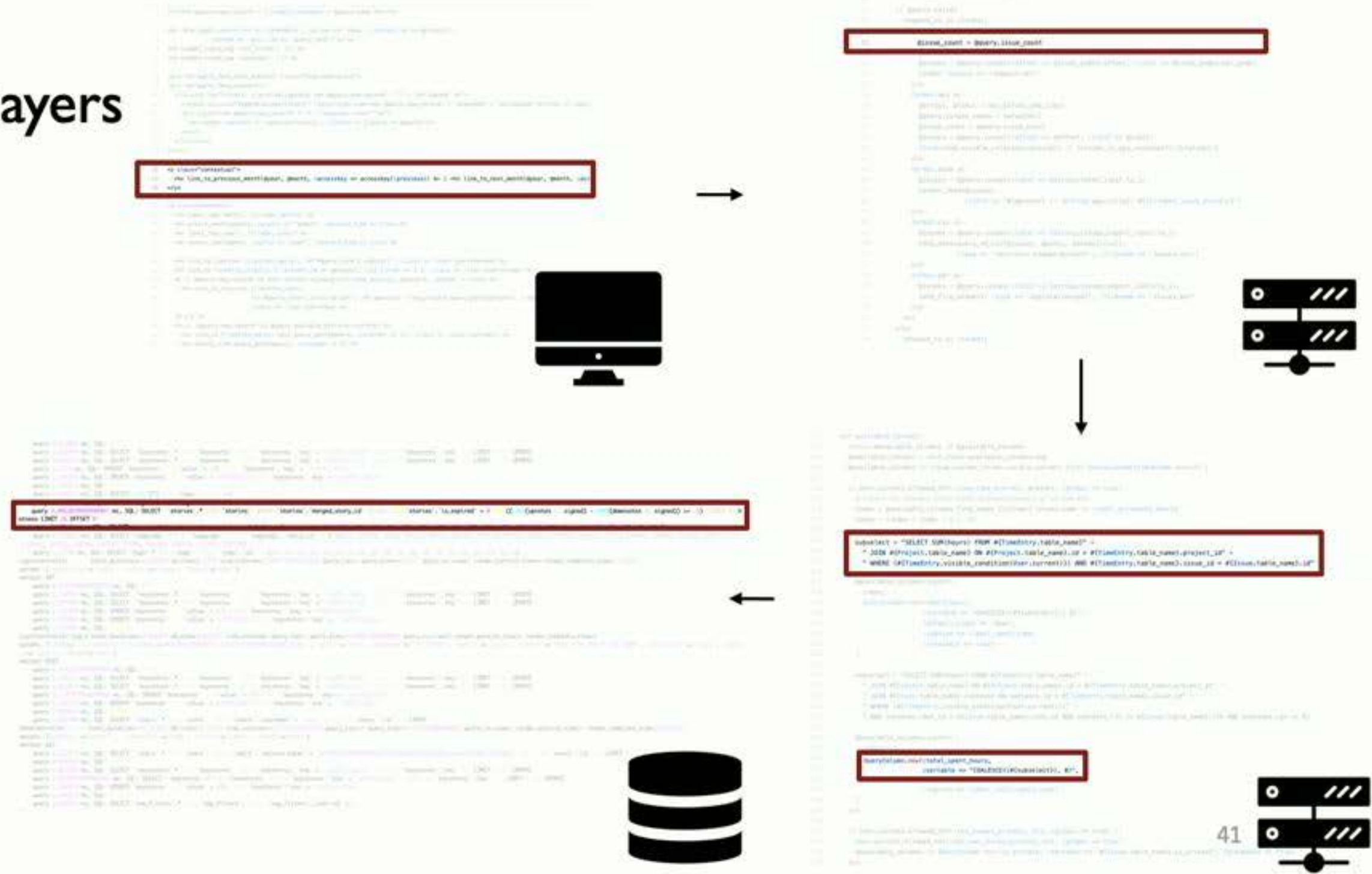
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers



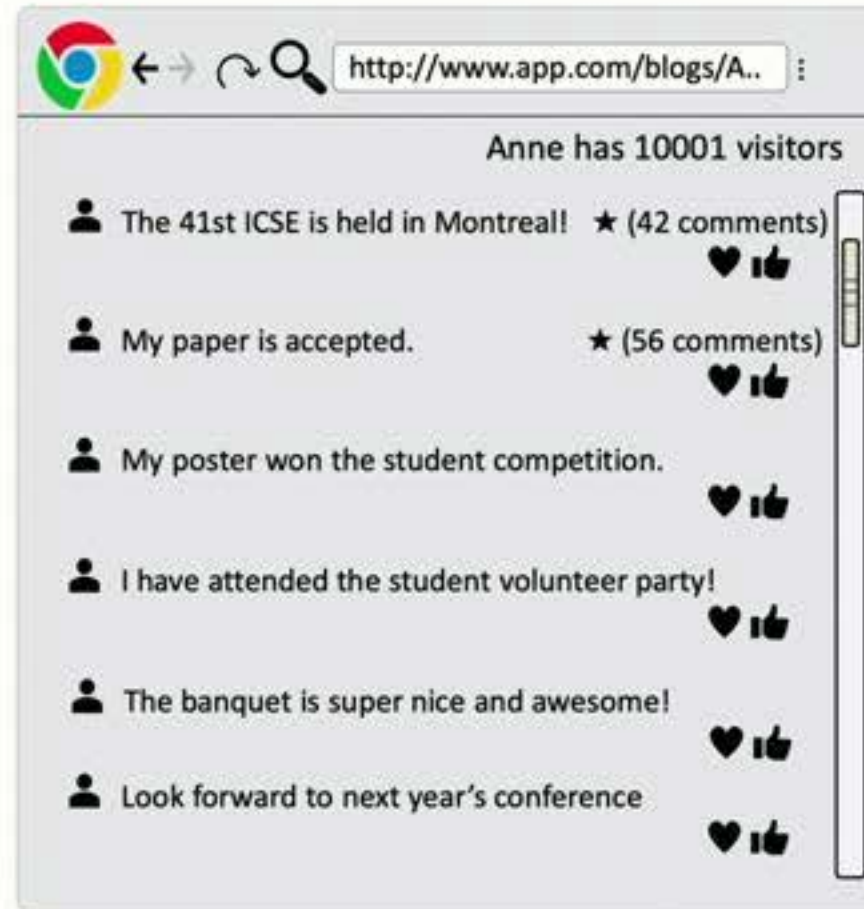
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers



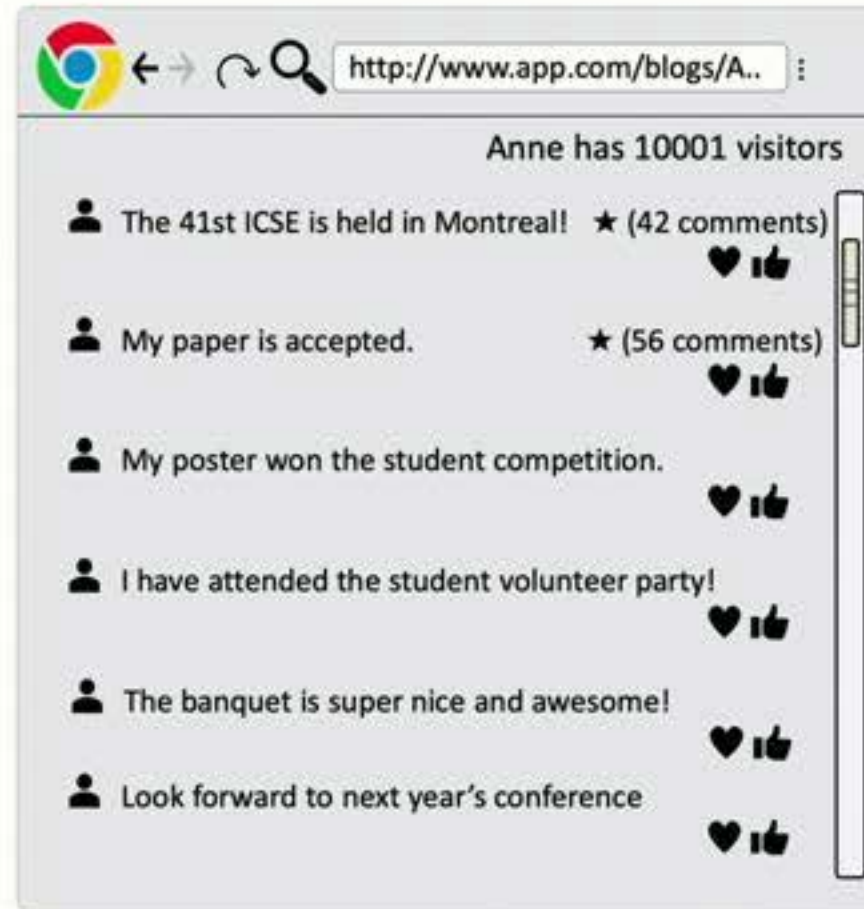
Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers
- Understanding the tradeoff

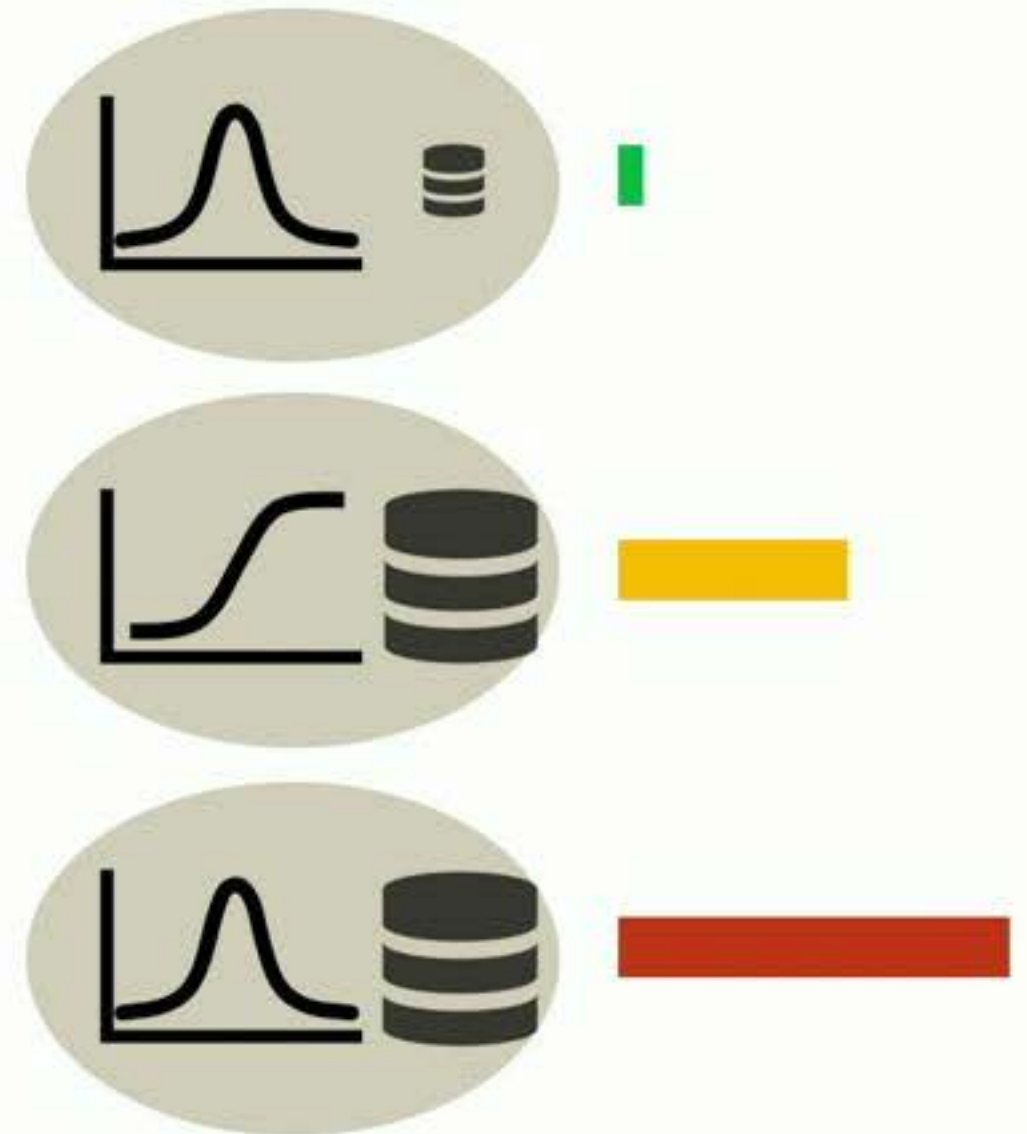


Making Design-Performance Tradeoff is Not Trivial

- Changing code across layers
- Understanding the tradeoff

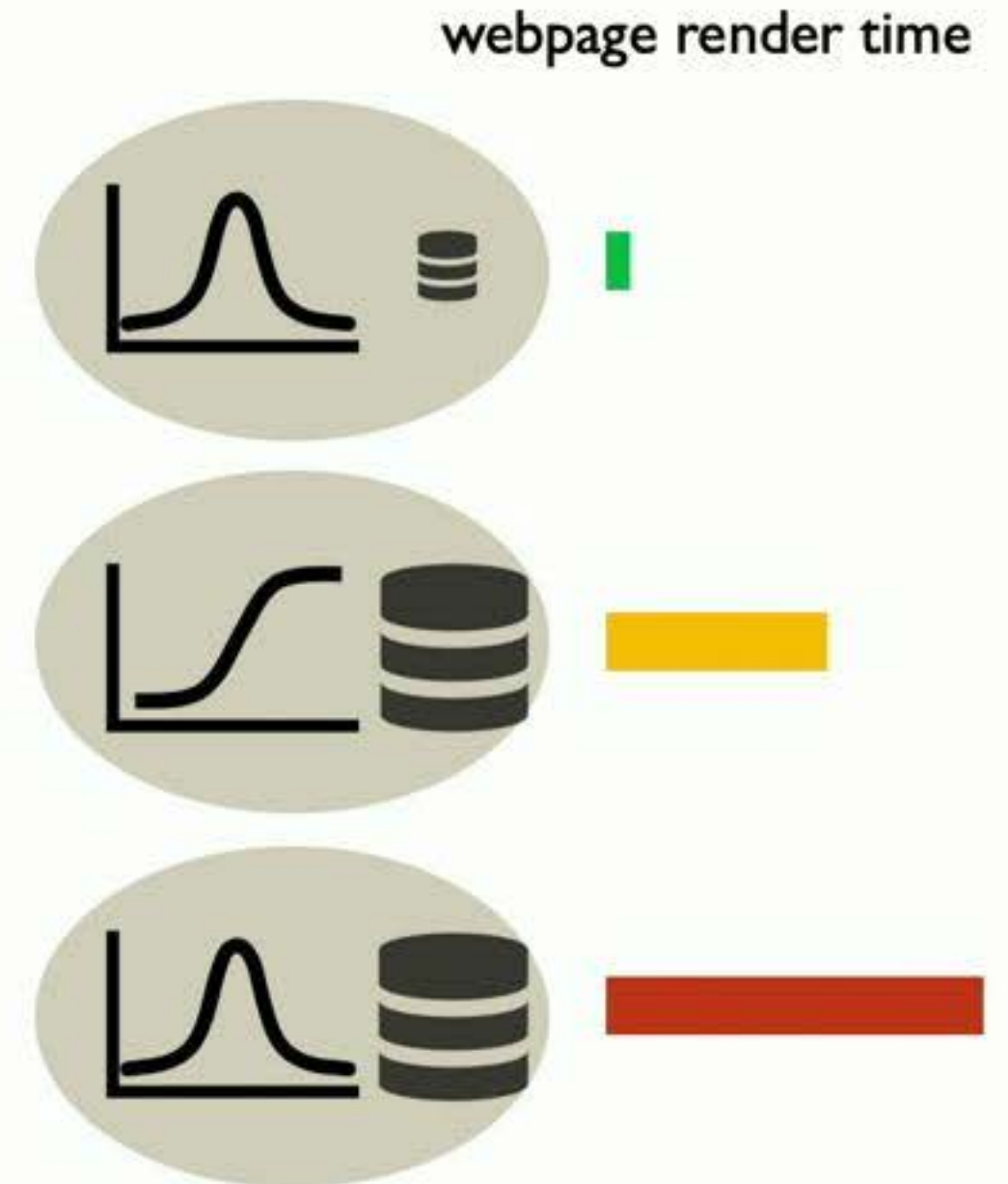
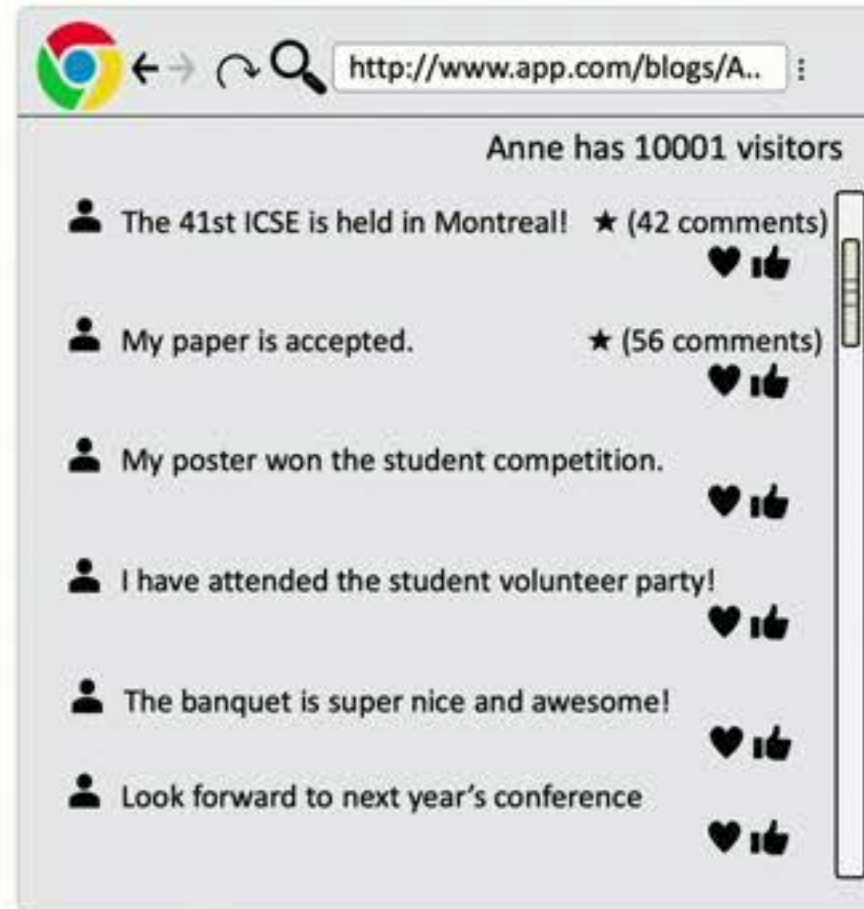


webpage render time

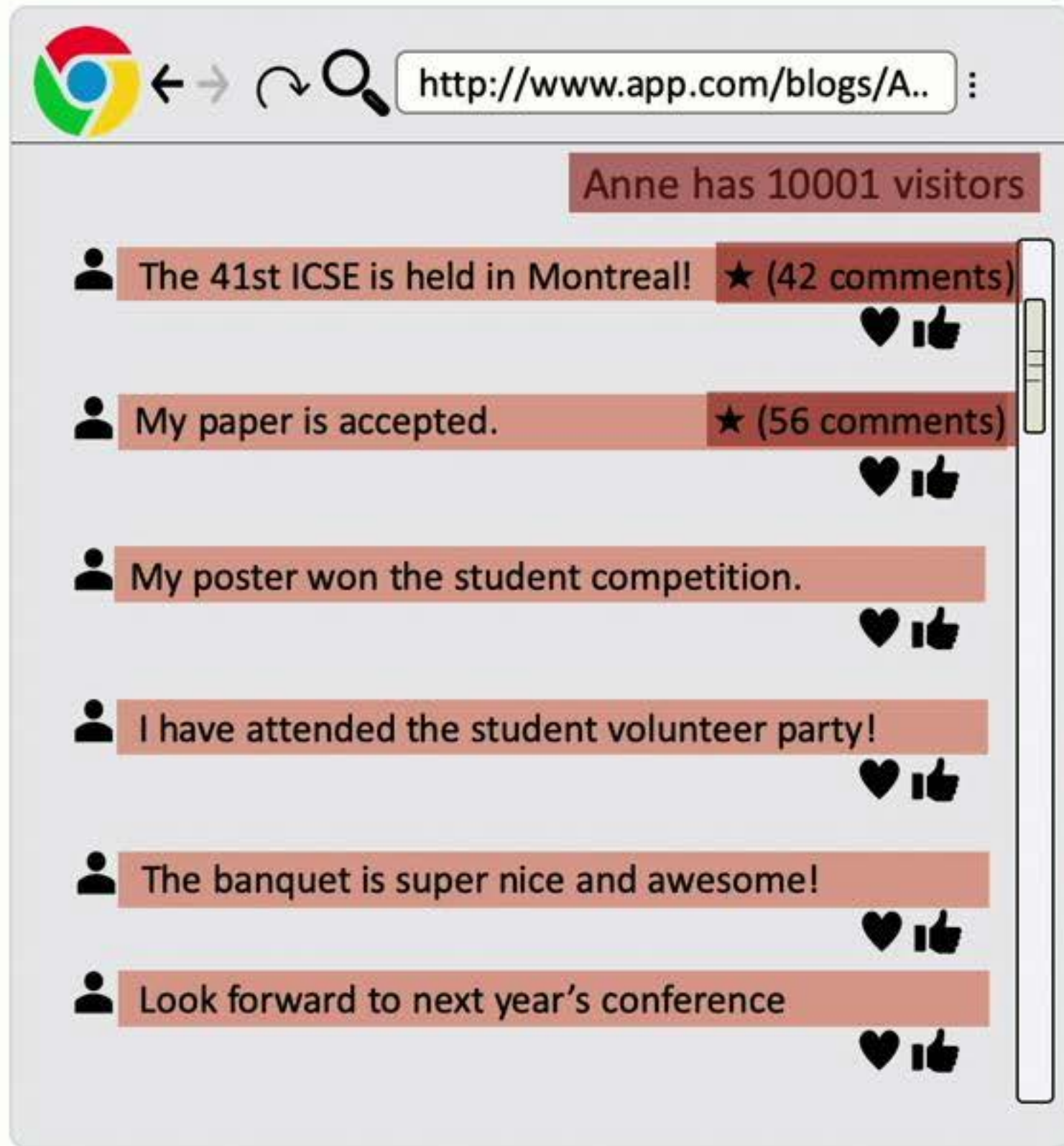


Making Design-Performance Tradeoff is Not Trivial

- 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:
  ...
```

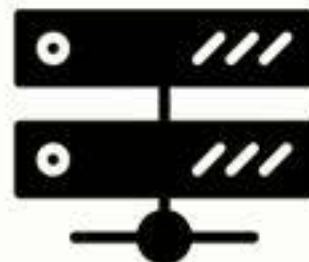


blogs_controller.rb

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

blog.rb

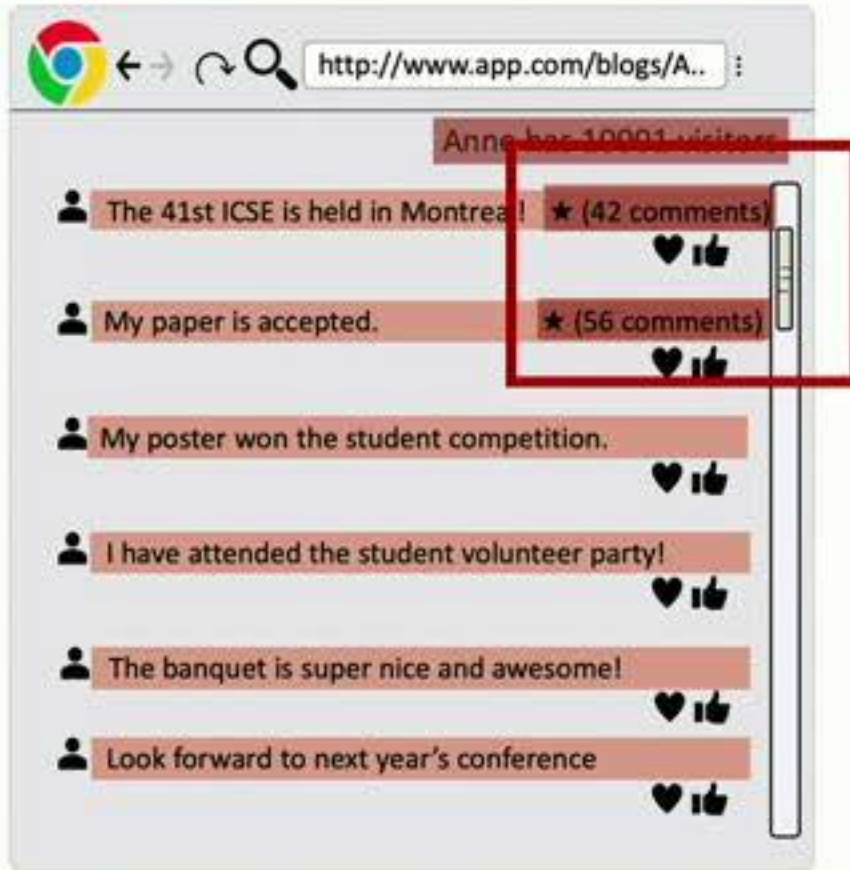
```
class Blog
  def exclude_self(user):
    where('comments.user_id
      !=user.id)
```



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



How does Panorama generate heatmap?

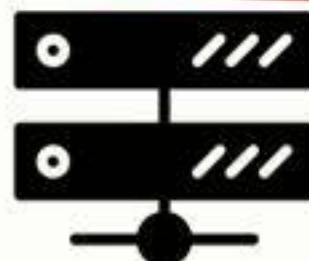


```
index.html.erb  
...  
user.blogs.each |b|  
  if @bcount[b] > 100:  
    ...  
  end  
end
```



```
blogs_controller.rb  
...  
@bcount=blogs.joins(:comments)  
  .exclude_self(:user).count  
...  
end
```

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

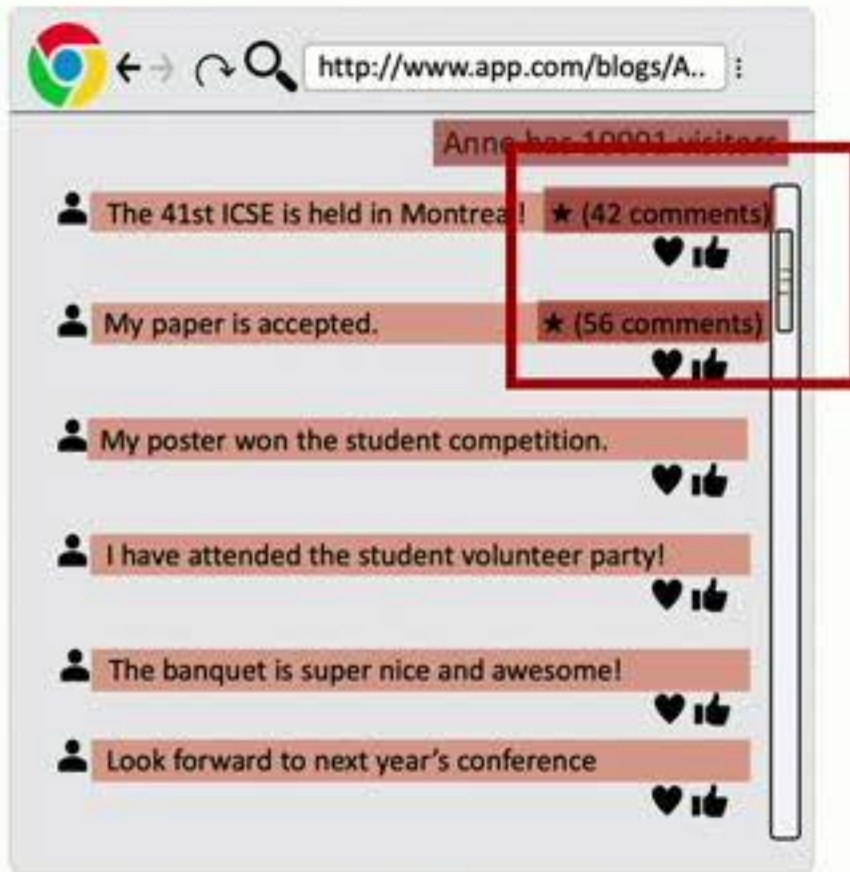


$N_{\text{blogs}} * N_{\text{comments}}$

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



How does Panorama generate heatmap?



$N_{\text{blogs}} * N_{\text{comments}}$

index.html.erb

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

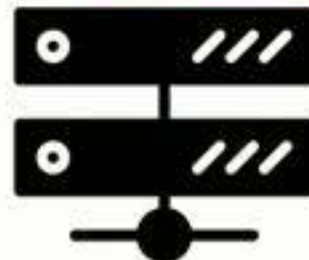


blogs_controller.rb

```
...  
@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?



$$N_{\text{blogs}} * N_{\text{comments}}$$

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

```

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

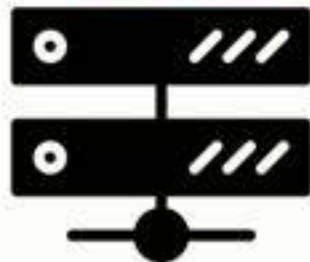
```

```
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

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

high cost → low cost

Panorama: Suggests Design Change and Refactors Code Automatically

The image shows a browser window with a list of blog posts. The browser's address bar contains the URL `http://www.app.com/blogs/A..`. The page content includes a header "Anne has 10001 visitors" and a list of posts:

- "The 41st ICSE is held in Montreal! ★ (42 comments)" with a heart and thumbs-up icon.
- "My paper is accepted. ★ (56 comments)" with a hand cursor over the star icon.
- "My poster won the student competition." with a hand cursor over the text.
- "I have attended the student volunteer party!" with a heart and thumbs-up icon.
- "The banquet is super nice and awesome!" with a heart and thumbs-up icon.
- "Look forward to next year's conference" with a heart and thumbs-up icon.

Annotations include:

- A box labeled "removal" pointing to the star icon of the second post.
- A box labeled "async load approximation" pointing to the star icon of the second post.
- A box labeled "pagination" pointing to the text of the third post.

At the bottom, a horizontal arrow points from "high cost" (dark red) to "low cost" (green), with a color gradient bar above it.

Panorama: Suggests Design Change and Refactors Code Automatically

The image illustrates the Panorama tool's ability to suggest design changes and refactor code automatically. It shows two side-by-side browser windows displaying a blog page with a list of posts. The left window shows the original page with high-cost elements highlighted in red, and the right window shows the refactored page with low-cost elements highlighted in green.

Original Page (Left):

- Header: Anne has 10001 visitors
- Post 1: The 41st ICSE is held in Montreal! (42 comments)
- Post 2: My paper is accepted. (56 comments)
- Post 3: My poster won the student competition.
- Post 4: I have attended the student volunteer party!
- Post 5: The banquet is super nice and awesome!
- Post 6: Look forward to next year's conference

Refactored Page (Right):

- Header: [Loading spinner]
- Post 1: The 41st ICSE is held in Montreal!
- Post 2: My paper is accepted.
- Post 3: My poster has won the student competition
- Post 4: I have attended the student volunteer party!
- Post 5: The banquet is super nice and awesome!
- Footer: << 1 2 3 ... >>

Annotations:

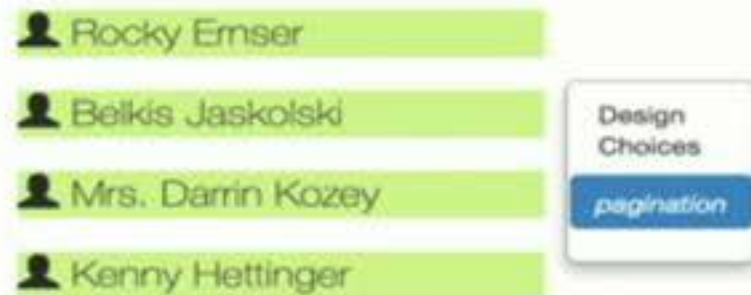
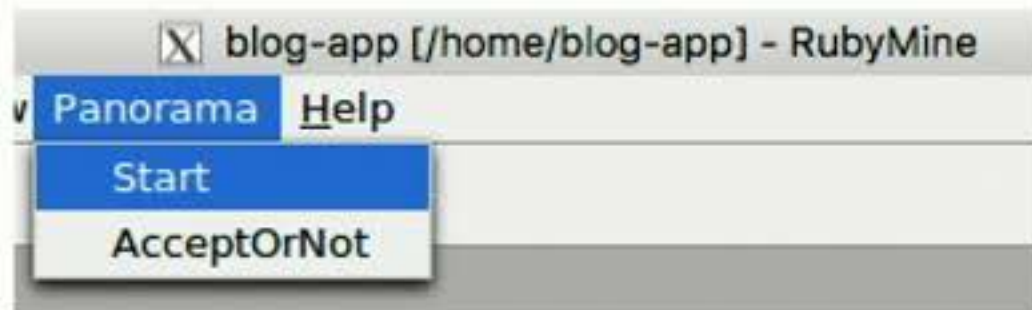
- removal**: A callout box pointing to the comment counts in the original page.
- asnc load approximation**: A callout box pointing to the pagination controls in the original page.
- pagination**: A callout box pointing to the pagination controls in the original page.

Cost Scale: A color-coded bar at the bottom indicates the cost reduction from high cost (dark red) to low cost (green).

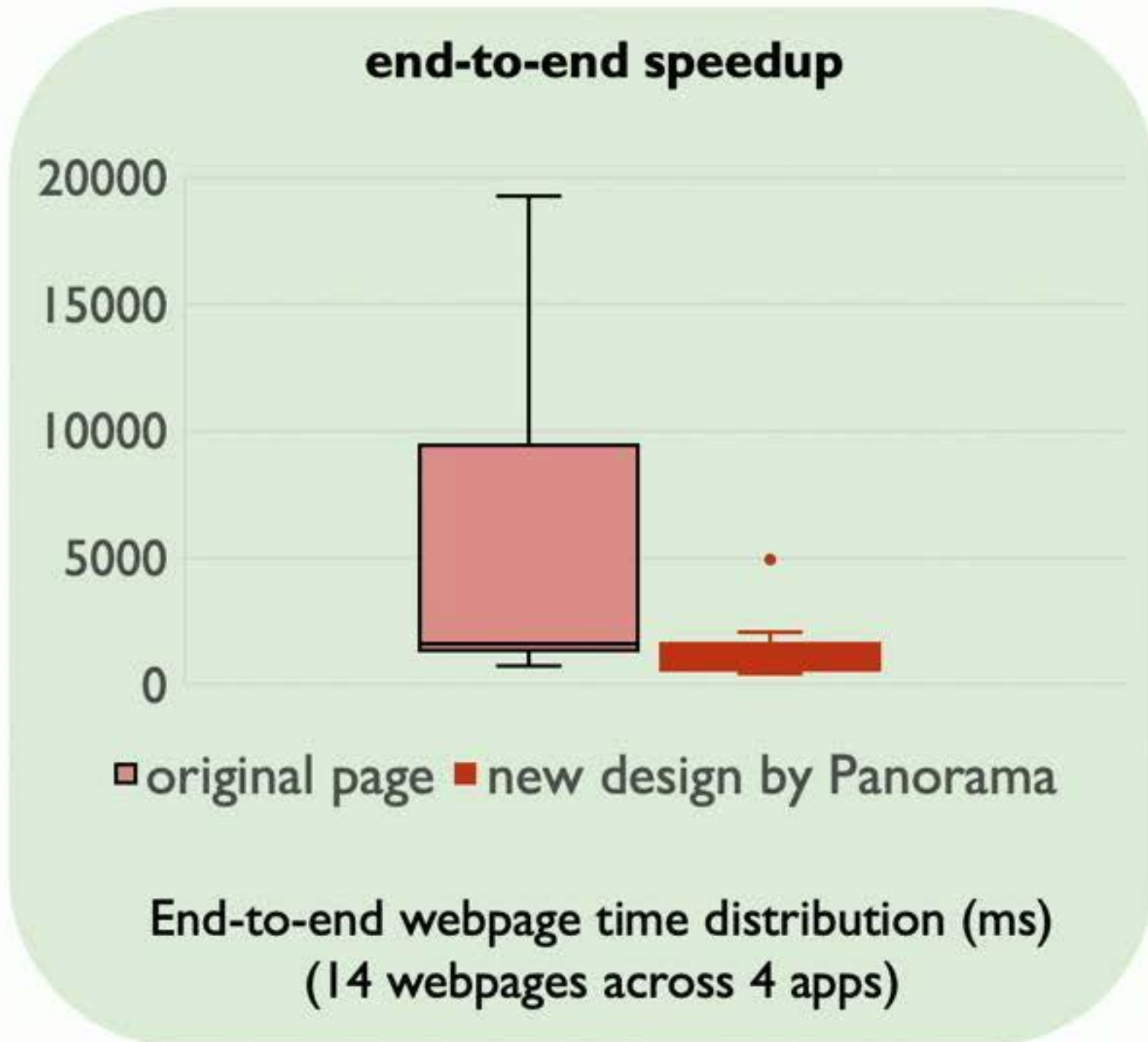
Panorama as An IDE Plugin



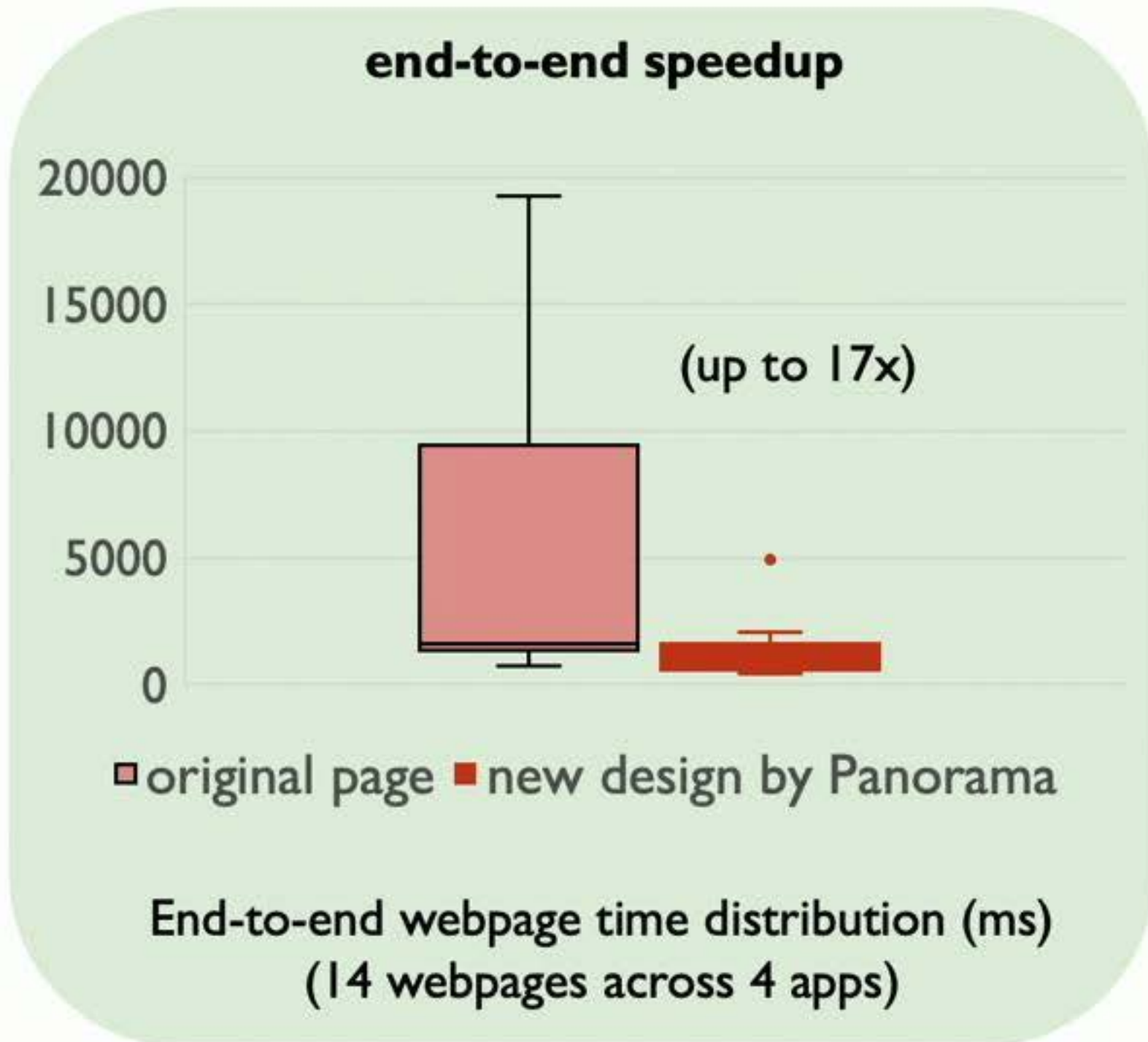
The Most Intelligent Ruby and Rails IDE



Panorama Evaluation

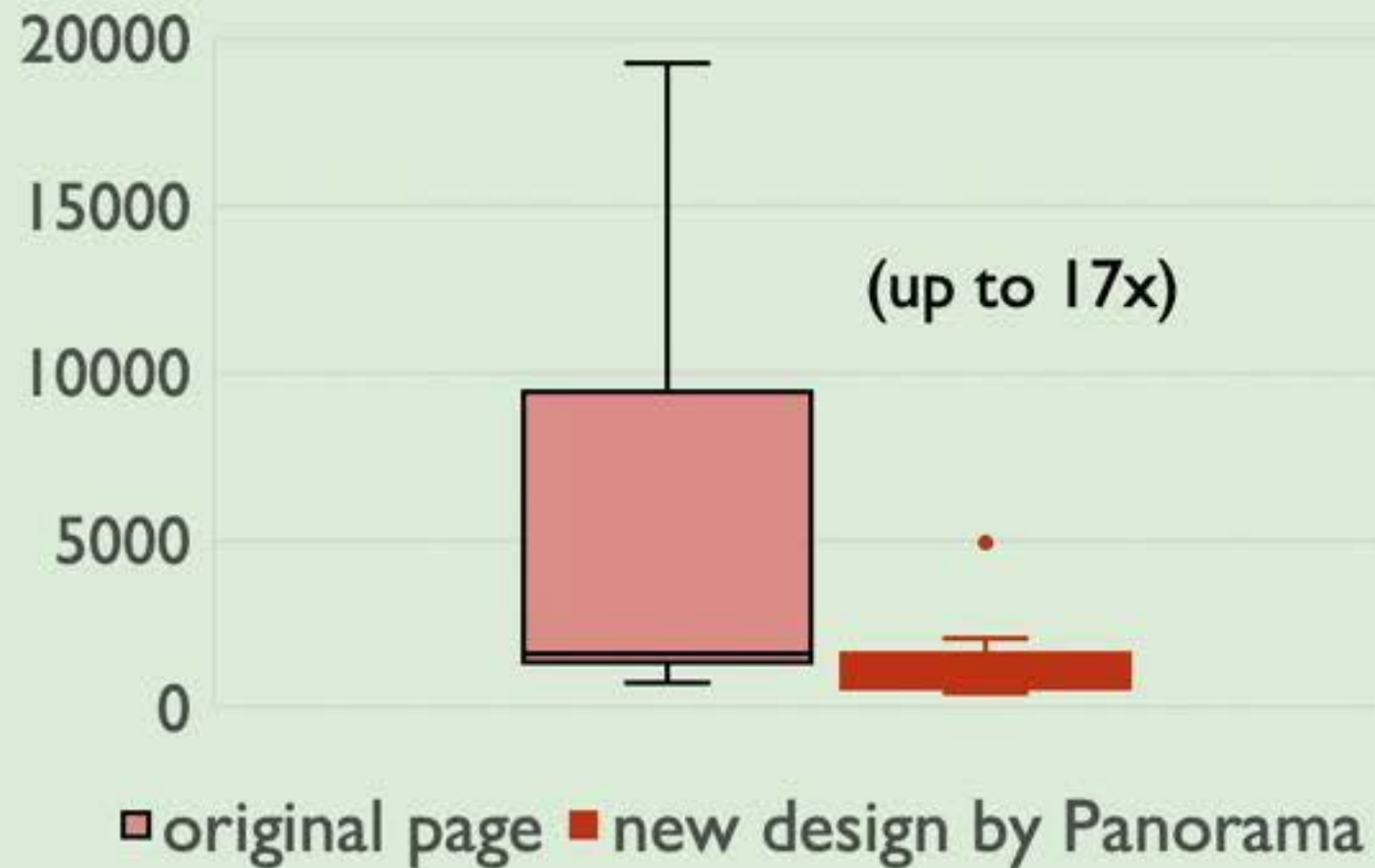


Panorama Evaluation



Panorama Evaluation

end-to-end speedup



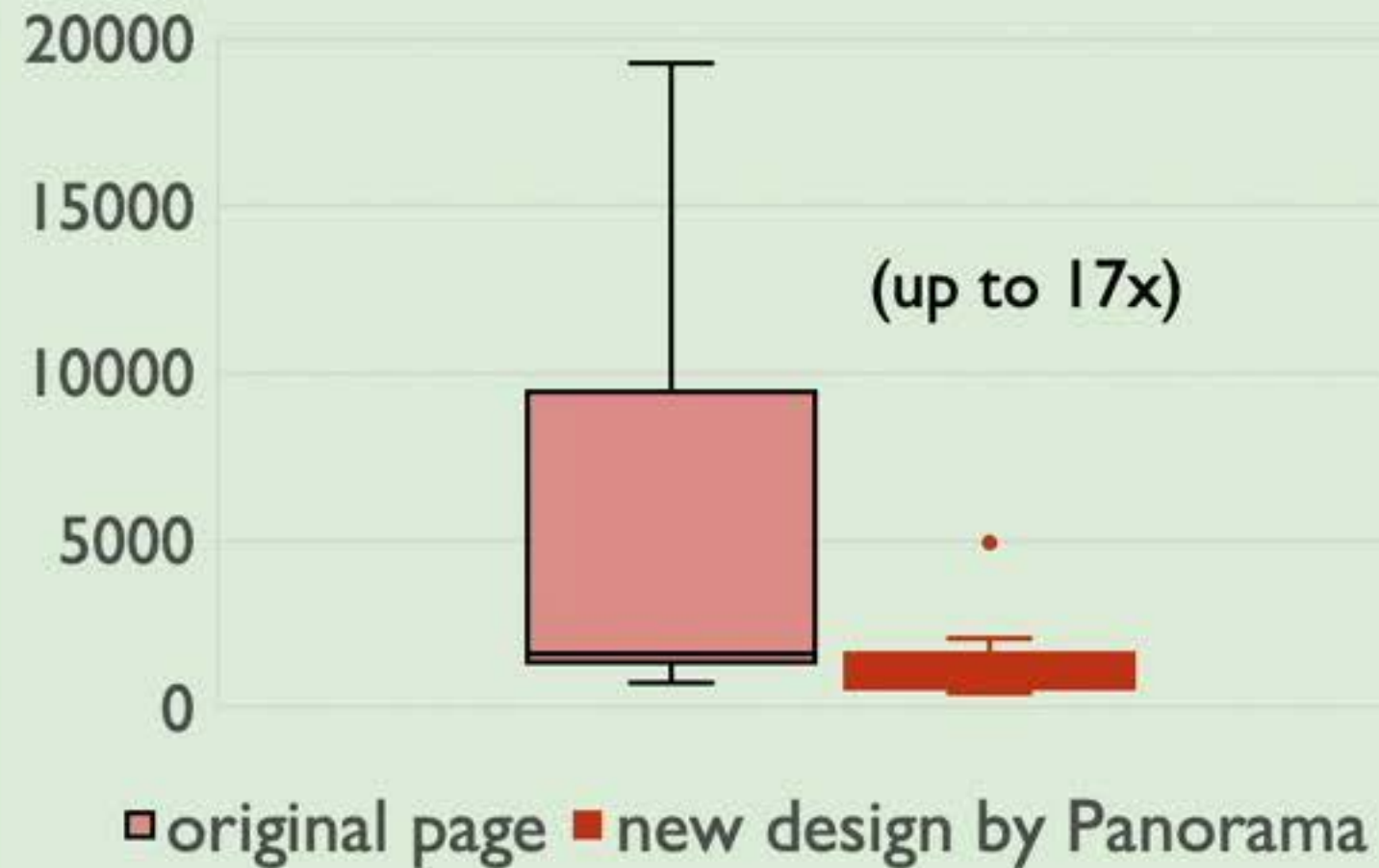
End-to-end webpage time distribution (ms)
(14 webpages across 4 apps)

user's satisfaction

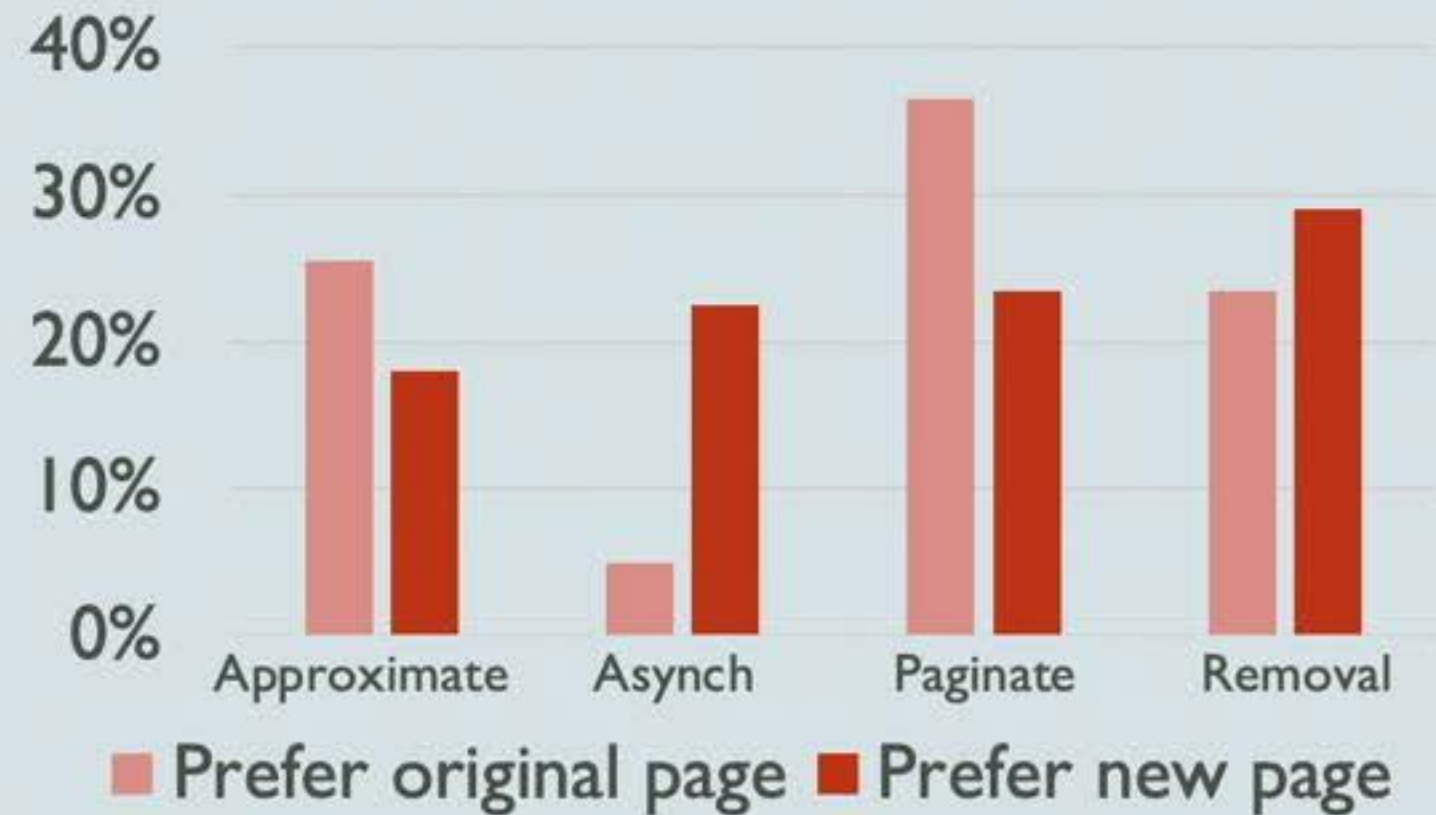
user survey result on page preference

Panorama Evaluation

end-to-end speedup



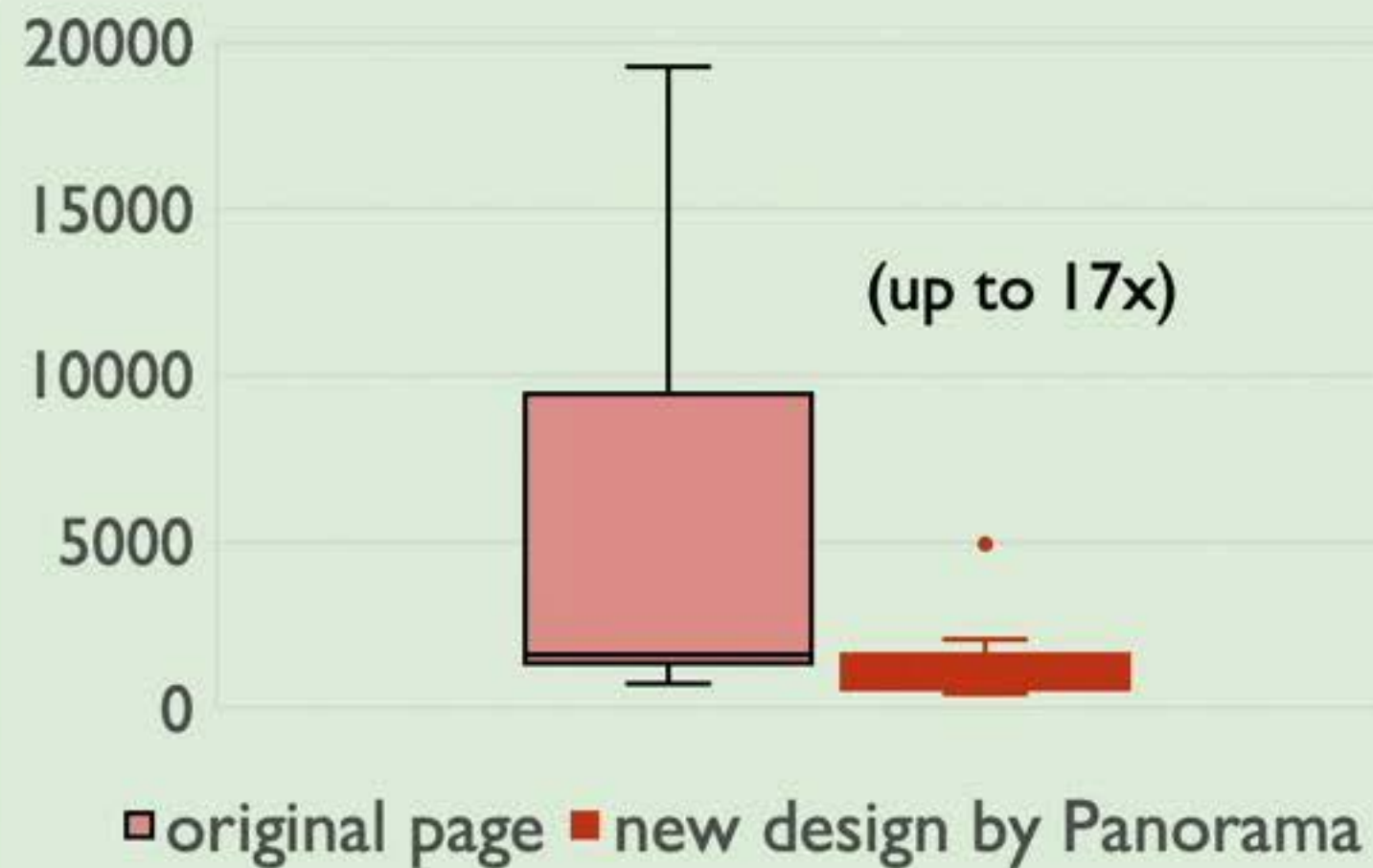
user's satisfaction



user survey result on page preference

Panorama Evaluation

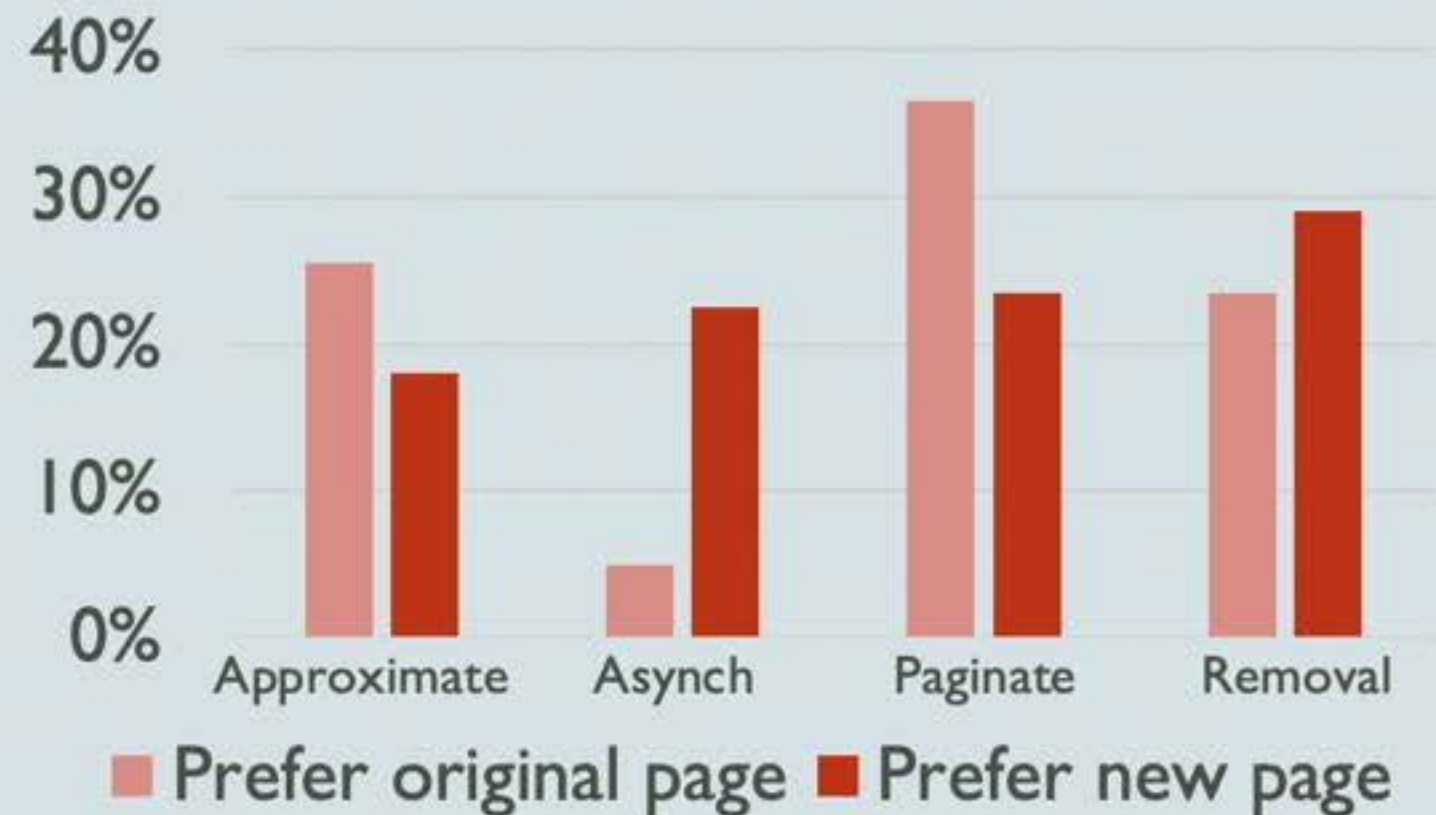
end-to-end speedup



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

user's satisfaction

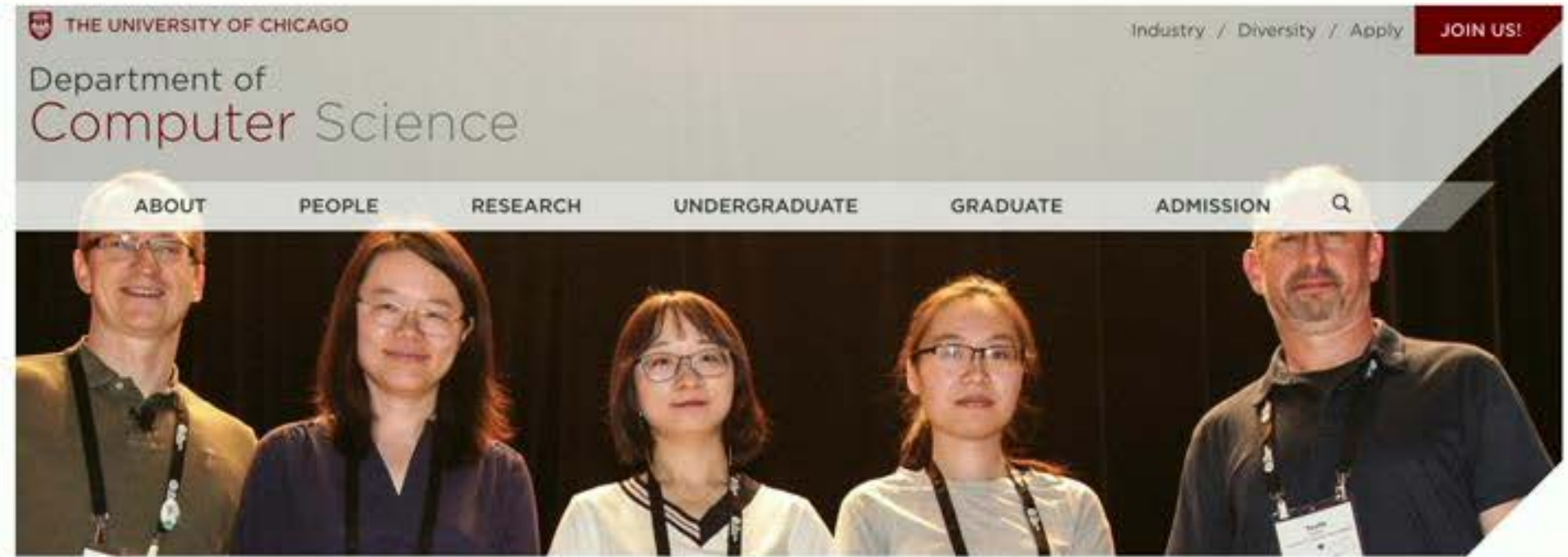
Avg: 22% prefer original, 20% prefer new



user survey result on page preference

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



the morning paper
a random walk through Computer Science research, by Adrian Colyer

[ABOUT](#) [ARCHIVES](#) [INFOQ QR EDITIONS](#) [SEARCH](#) [SUBSCRIBE](#) [TAGS](#) [PRIVACY](#)

How not to structure your database-backed web applications: a study of performance bugs in the wild

JUNE 28, 2018

the morning paper
a random walk through Computer Science research, by Adrian Colyer

[ABOUT](#) [ARCHIVES](#) [INFOQ QR EDITIONS](#) [SEARCH](#) [SUBSCRIBE](#) [TAGS](#) [PRIVACY](#)

View-centric performance optimization for database-backed web applications



Press Coverage

- Uchicago news
- Morning Paper
- Ruby Weekly

The screenshot shows the website 'the morning paper' with the subtitle 'a random walk through Computer Science research, by Adrian Colyer'. The navigation menu includes links for ABOUT, ARCHIVES, INFO QR EDITIONS, SEARCH, SUBSCRIBE, TAGS, and PRIVACY. A 'JOIN US!' button is visible in the top right corner. Below the navigation menu, the text 'Ruby Weekly' is displayed in red. A subscription form is present with the text 'Want to subscribe? Enter your address here' and a yellow 'Subscribe now »' button. Below the form, it states 'Easy to unsubscribe at any time. Your e-mail address is safe — here's our privacy policy.' The main content area features a link to an article: 'How Not to Structure Your Database-Backed Webapps' — A breakdown of a paper that studies 12 of the most popular Rails apps looking for and fixing ORM issues. The author is listed as ADRIAN COLYER. Below the article link, the text 'View-centric performance optimization for database-backed web applications' is visible.

Press Coverage

- Uchicago news
- Morning Paper
- Ruby Weekly
- Hackernews

The screenshot shows the homepage of 'the morning paper', a website dedicated to Computer Science research by Adrian Colyer. The site features a navigation menu with links for ABOUT, ARCHIVES, INFO Q R EDITIONS, SEARCH, SUBSCRIBE, TAGS, and PRIVACY. A 'JOIN US!' button is also visible. Below the navigation, the title 'Ruby Weekly' is displayed in red. The main content area shows a Hacker News article titled 'How not to structure database-backed web apps: performance bugs in the wild (acolyer.org)'. The article has 486 points and 308 comments. The article text discusses performance optimization for database-backed web applications, mentioning moderate busy backend platforms and specific optimization techniques like long TTLs, asynchronous processing, and avoiding serialization.

the morning paper
a random walk through Computer Science research, by Adrian Colyer

ABOUT ARCHIVES INFO Q R EDITIONS SEARCH SUBSCRIBE TAGS PRIVACY

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: performance bugs in the wild (acolyer.org)
486 points by modelmachine 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. Precalculate 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.

view source performance optimization for database-backed web applications

Press Coverage

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

The screenshot shows the homepage of 'the morning paper', a website described as 'a random walk through Computer Science research, by Adrian Colyer'. The site features a navigation menu with links for ABOUT, ARCHIVES, INFO Q R EDITIONS, SEARCH, SUBSCRIBE, TAGS, and PRIVACY. A 'JOIN US!' button is visible in the top right corner. Below the navigation, the title 'Ruby Weekly' is displayed in red. The main content area features a post from Hacker News titled 'How not to structure database-backed web apps: performance hogs in the wild (acolyer.org)'. The post has 486 points and 308 comments. The post content includes a section for 'speedshop', which is a Ruby on Rails performance consultancy. The 'Products and Services' section lists several offerings: 'The Complete Guide to Rails Performance' (a 370-page book with 17 hours of HD screencasts), 'The Rails Performance Workshop' (a four-week online intensive), 'Corporate Training' (private workshops), and 'Speedshop Blog' (the number one Ruby on Rails performance blog on the 'net'). A large black redaction box covers the right side of the page, obscuring the Speedshop logo and part of the text.

the morning paper
a random walk through Computer Science research, by Adrian Colyer

ABOUT ARCHIVES INFO Q R EDITIONS SEARCH SUBSCRIBE TAGS PRIVACY

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: performance hogs in the wild (acolyer.org)
486 points by modelmachine on June 28, 2018 | hide | past | web | favorite | 308 comments

catch on Jun
I've worke
It greatly c
to two maj
1 - Have v
between fr
2 - Genera
indexes fill
that data
Do things
Beyond th

speedshop

Speedshop is a Ruby on Rails performance consultancy that optimizes the full stack - frontend, backend and environment - to generate revenue and cut scaling costs for businesses on Rails through tools, information and training. Fast sites are profitable sites. Speed is a feature.

Products and Services

- [The Complete Guide to Rails Performance](#), an in-depth 370 page book and 17 hours of HD screencasts and interviews with top experts, including DHH himself.
- [The Rails Performance Workshop](#). A four-week online intensive workshop teaching Ruby performance skills. March 2020.
- [Corporate Training](#). Private workshops to teach fundamental Ruby performance skills.
- [Speedshop Blog](#), the number one Ruby on Rails performance blog on the 'net.

for 5ms 95p response times).
most certainly includes "dynamic" sites like Amazon or Wikipedia), I hold
and refresh in the background. Caching shouldn't be a compromise
path as streamlined as possible (select 1 column from 1 table with
ment can be done: 1 - By having the client make separate requests for
safer with protocol buffers than json)
than other 3rd party options.
ations

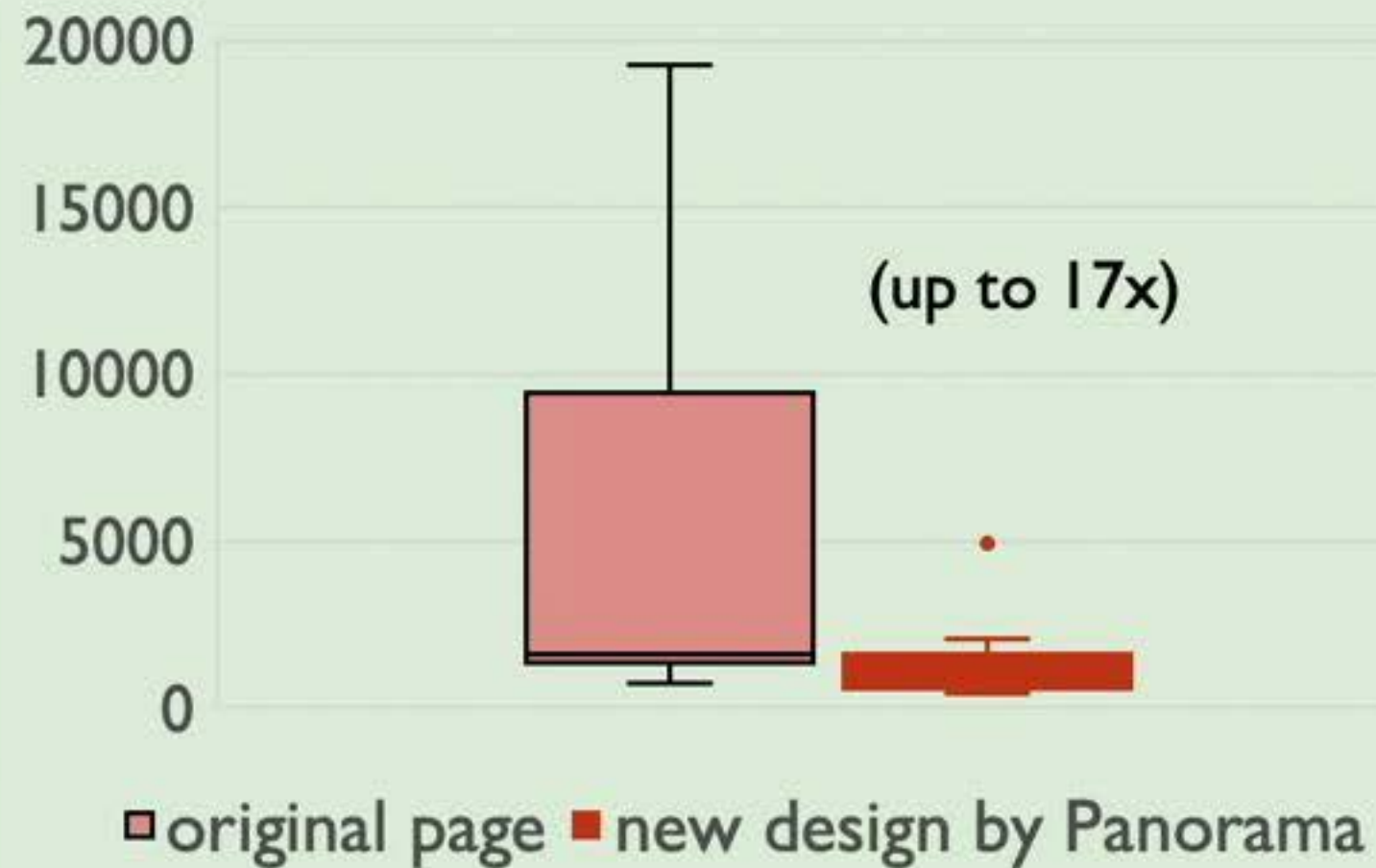
Panorama Summary

*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

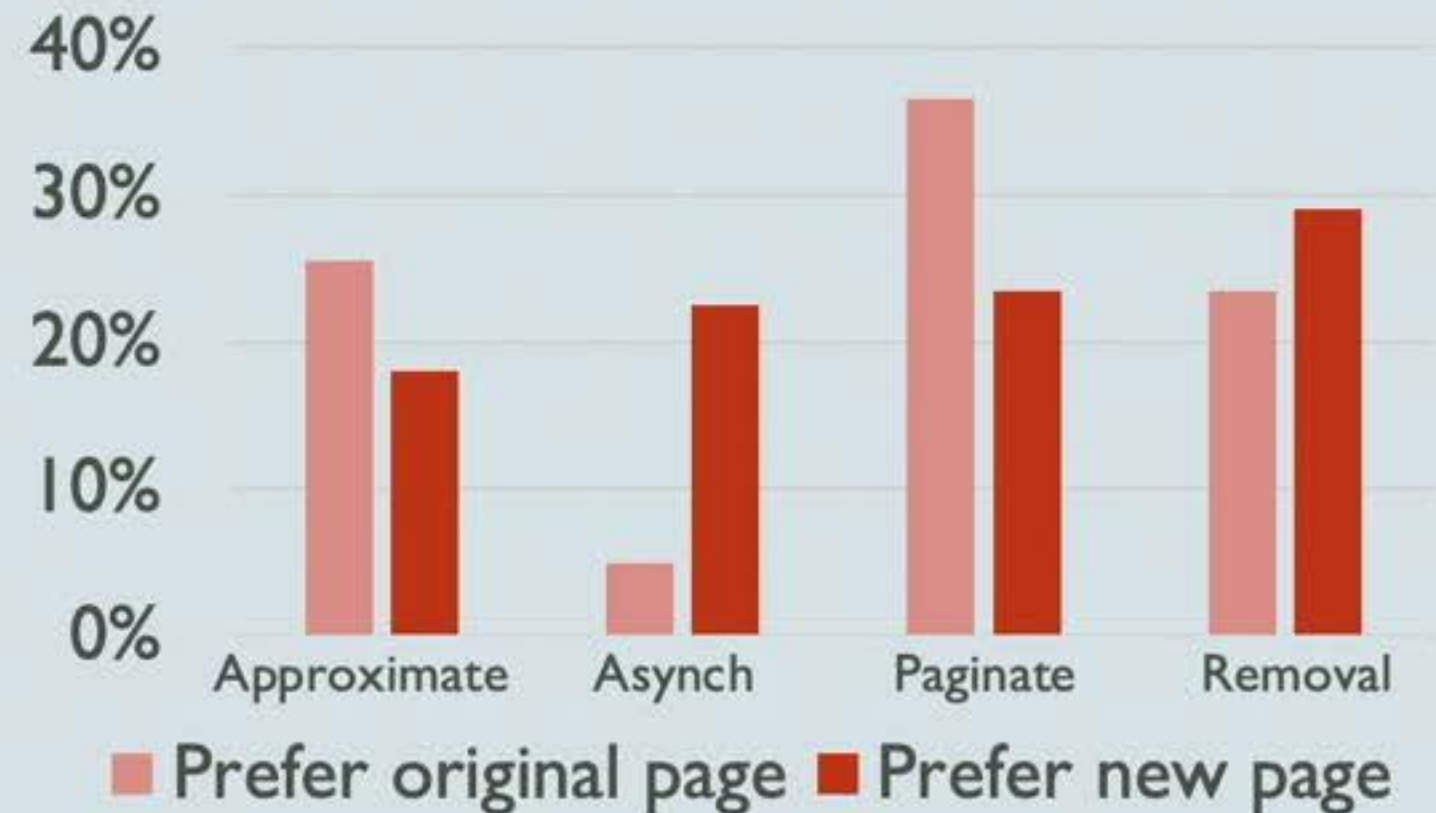
end-to-end speedup



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

user's satisfaction

Avg: 22% prefer original, 20% prefer new

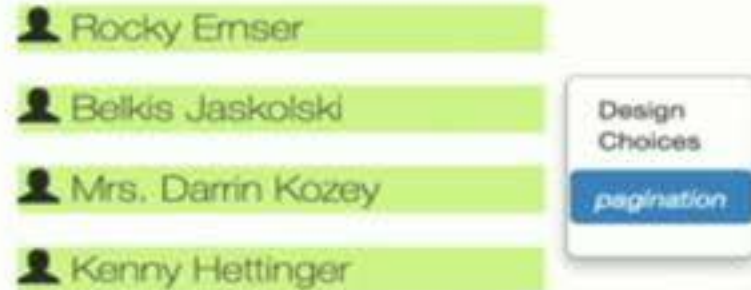
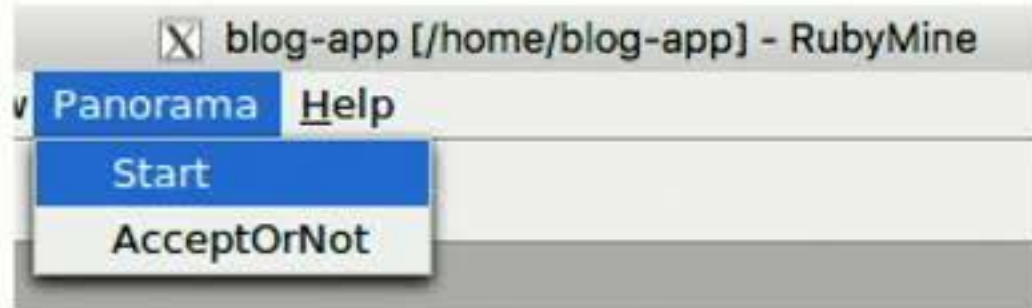


user survey result on page preference

Panorama as An IDE Plugin

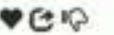


The Most Intelligent Ruby and Rails IDE

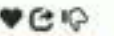


There are 114 blogs There are 6 comments

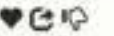
I Non voluptate consequatur. Ut qui vel. Praesentium qui autem. Distinctio dolor omnis. Quod aut quos. Quas sequi nostrum. Praesentium explicabo facilis. Qui molestiae accusantium. Quia qui ut. Aperiam consectetur repellat. Minima id ut. Quia ducimus harum. Vult quidem fugiat. Et et nisi. Illo sapiente atque. Consectetur cum sit. Voluptas quo ratione. Ut consequuntur itaque. Accusantium et laborum. Ut et soluta. Rerum voluptates ipsam. Officia eum veniam. Unde suscipit quae. Quae nihil culpa. Provident autem autem. Dolores et perspiciatis. Voluptas vel cupiditate. Ab velit excepturi. Ut aspernatur architecto. Optio eos maxime.



I Ad totam eligendi. Reiciendis qua et. Ad et placeat. Explicabo voluptatem ad. Iure aut cum. Et molestiae quia. Eaque sit aut. Sit doloremque est. Sit qui earum. Vero sit nam. Et sunt provident. Nulla saepe commodi. Soluta dolores et. Dolores eos enim. Quis consequatur id. Minima facere quibusdam. Quia maiores et. Necessitatibus eum ullam. Officis ut exercitationem. Ea ipsum reprehenderit. Repellat rerum aperiam. Sunt tenetur est. Quae rerum aut. Rerum dolenti alias. Iste provident id. Nemo necessitatibus debitis. Minus possimus est. Optio fugit doloribus. Maxime ut ex. Magni ipsum inventore.

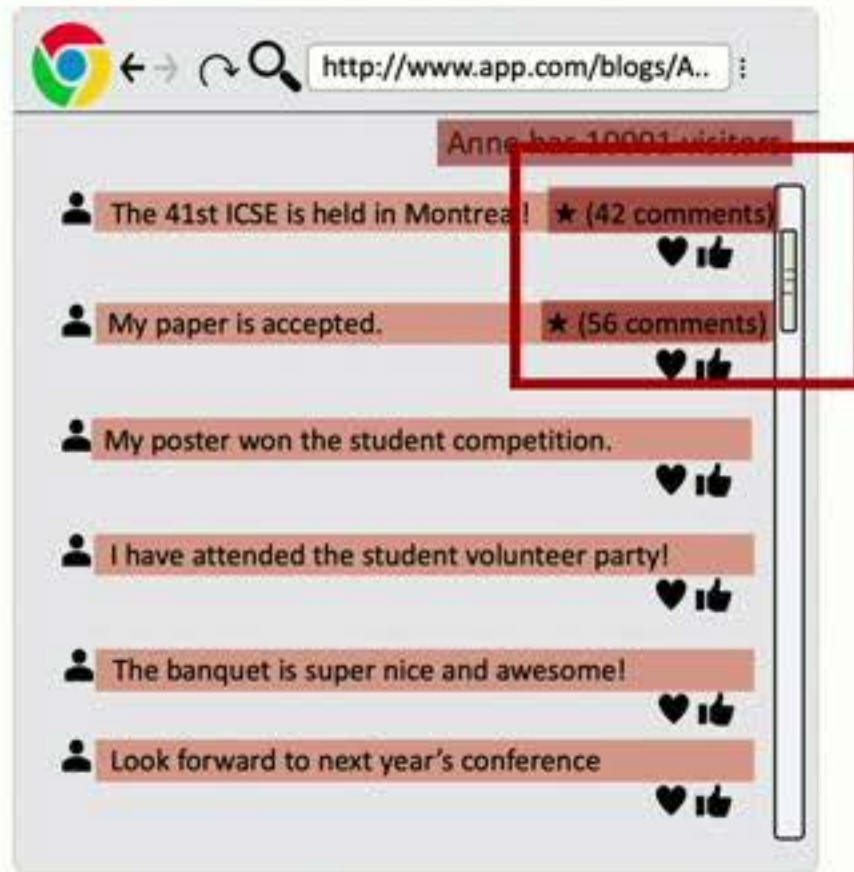


I Nam non eos. Quis repudiandae est. Id ab rerum. Nihil sit culpa. Et iusto et. Incidunt et aut. Perspiciatis odit laborum. Quaerat quasi et. Qui odio ea. Dicta quos labore. Et et architecto. Velit fugit quia. Magnam et enim. Dicta voluptatem hic. Est cum doloribus. Dolor eum dolorem. Aut magnam neque. Consequatur nam eligendi. Suscipit sit laboriosam. Cupiditate blanditis totam. Explicabo dolore itaque. Temporibus hic officia. Quaerat facere est. Dolorem quisquam consequatur. Quam odio quos. Reiciendis labore est. Non ea magni. Quae doloribus laborum. Possimus pariatur laborum. Alias earum corrupti.



<< Previous 1 2 3 4 5 6 7 8 9 10 11 12 ... 8100 8101 Next >>

How does Panorama generate heatmap?

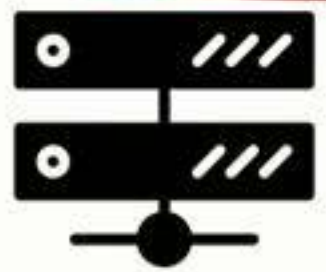


$$N_{\text{blogs}} * N_{\text{comments}}$$

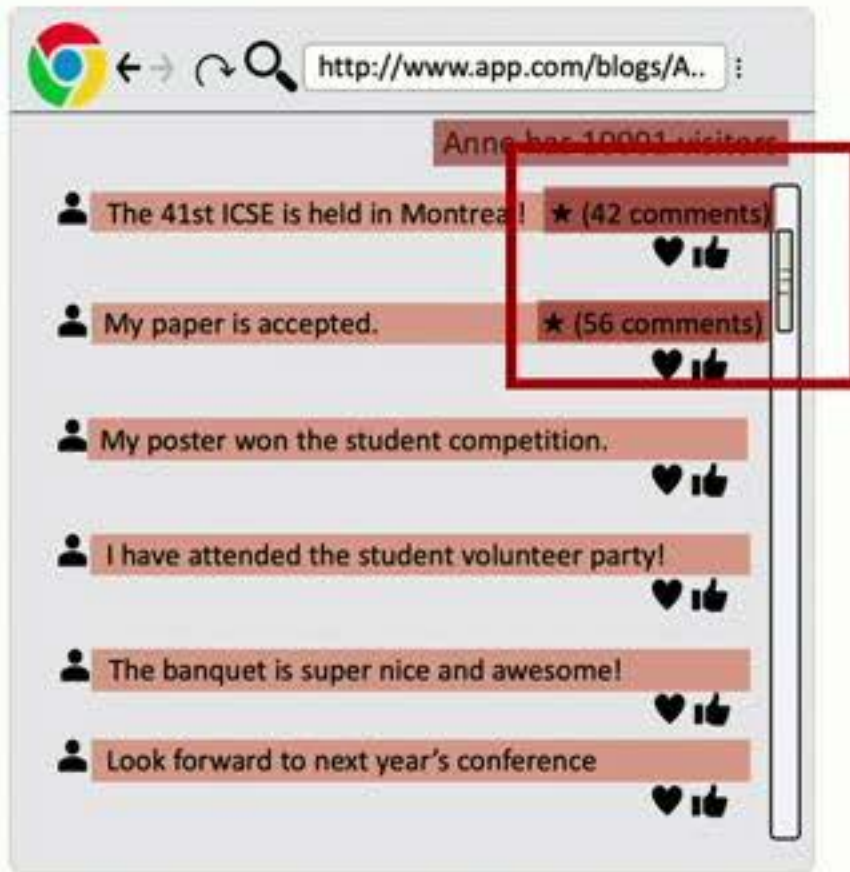
```
blogs_controller.rb  
...  
@bcount=blogs.joins(comments)  
                .exclude_self(user).count  
...  
                .where('comments.user_id != user.id')
```

```
index.html.erb  
...  
user.blogs.each |b|  
  if @bcount[b] > 100:  
...  
  end
```

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



How does Panorama generate heatmap?



$N_{\text{blogs}} * N_{\text{comments}}$

index.html.erb

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

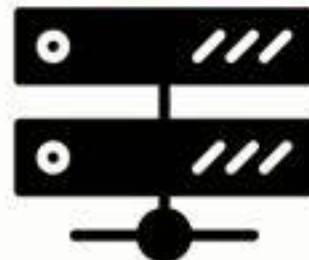


blogs_controller.rb

```
...  
@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?



$$N_{\text{blogs}} * N_{\text{comments}}$$

```
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)

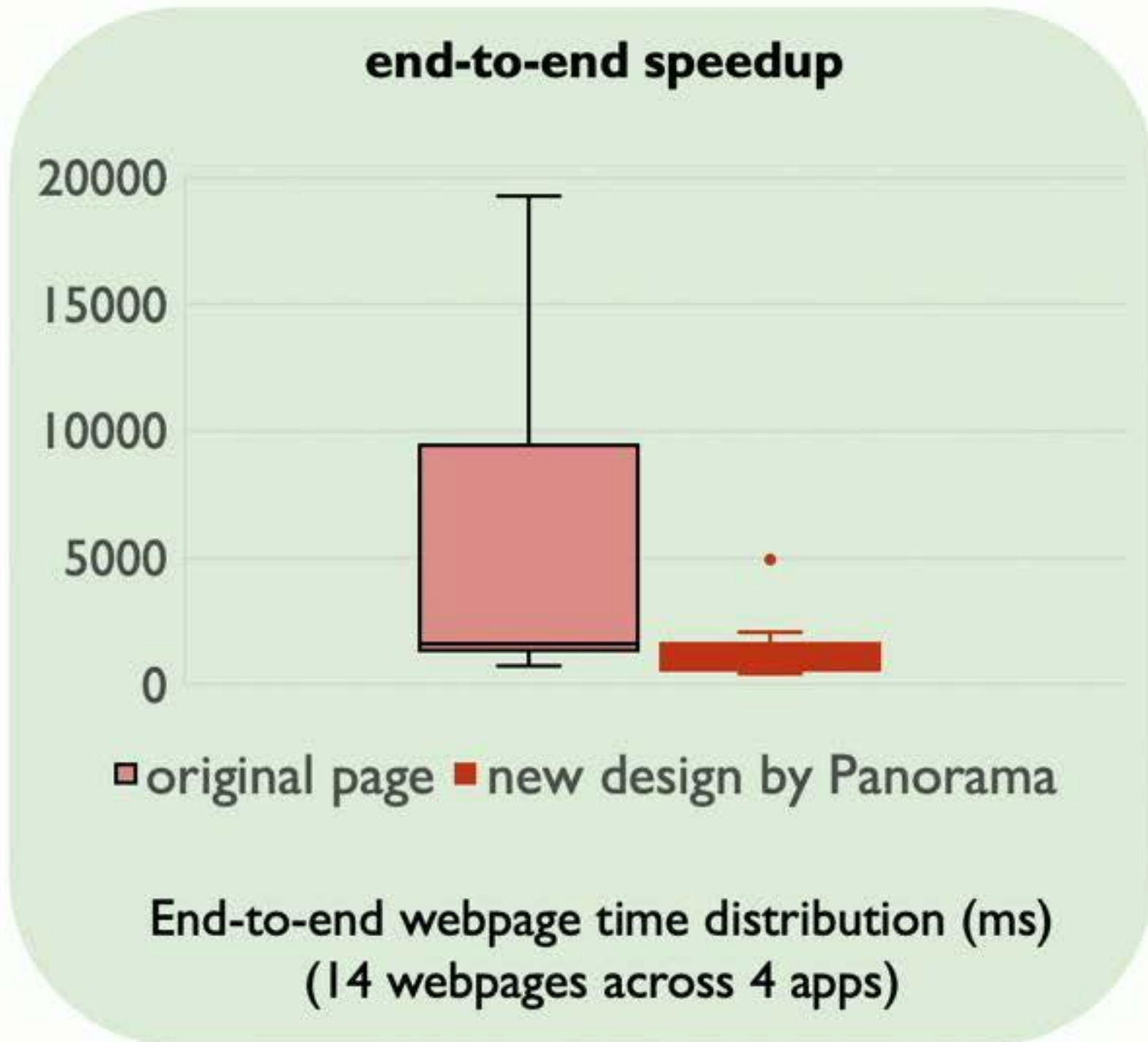
```

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

```

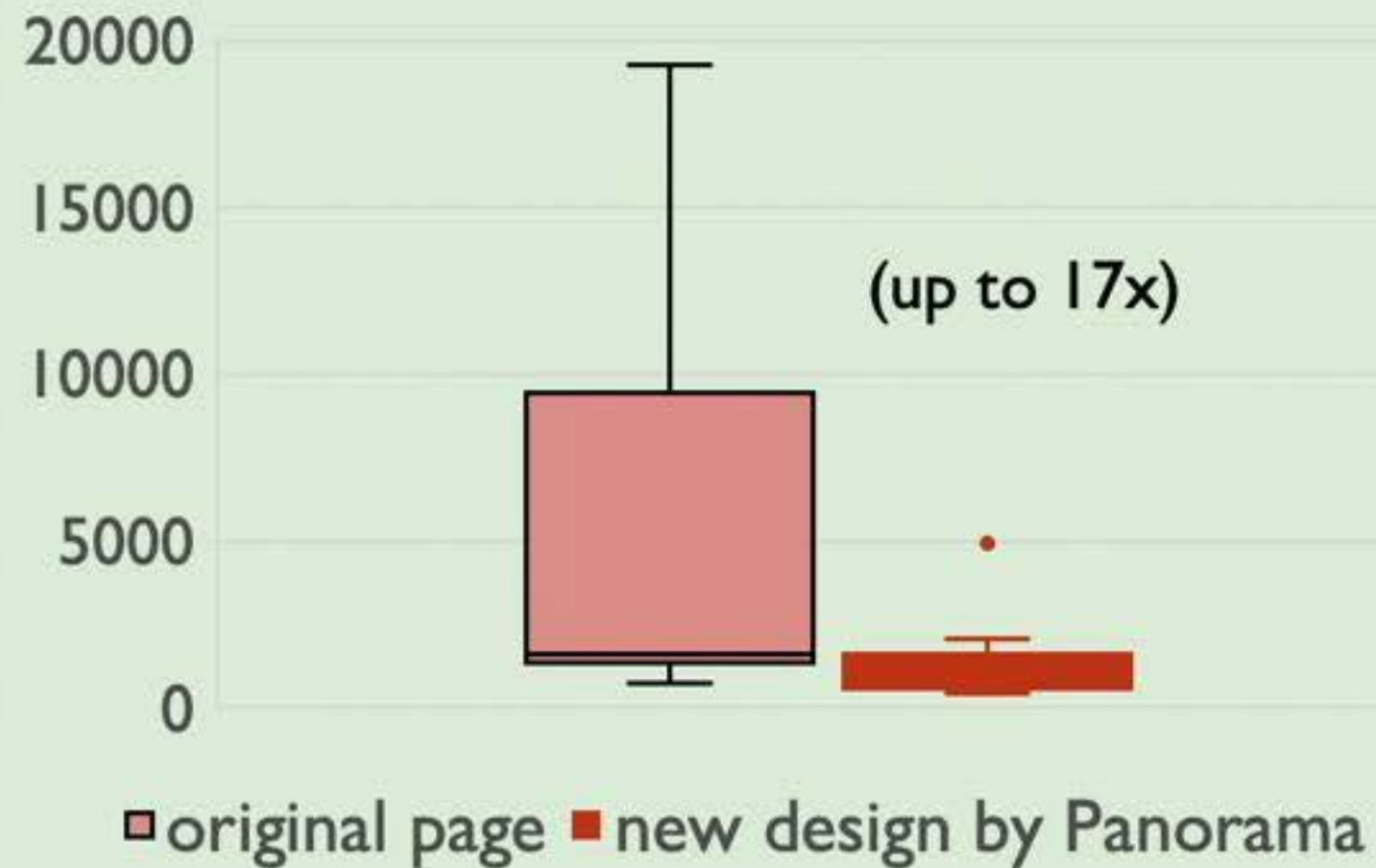


Panorama Evaluation



Panorama Evaluation

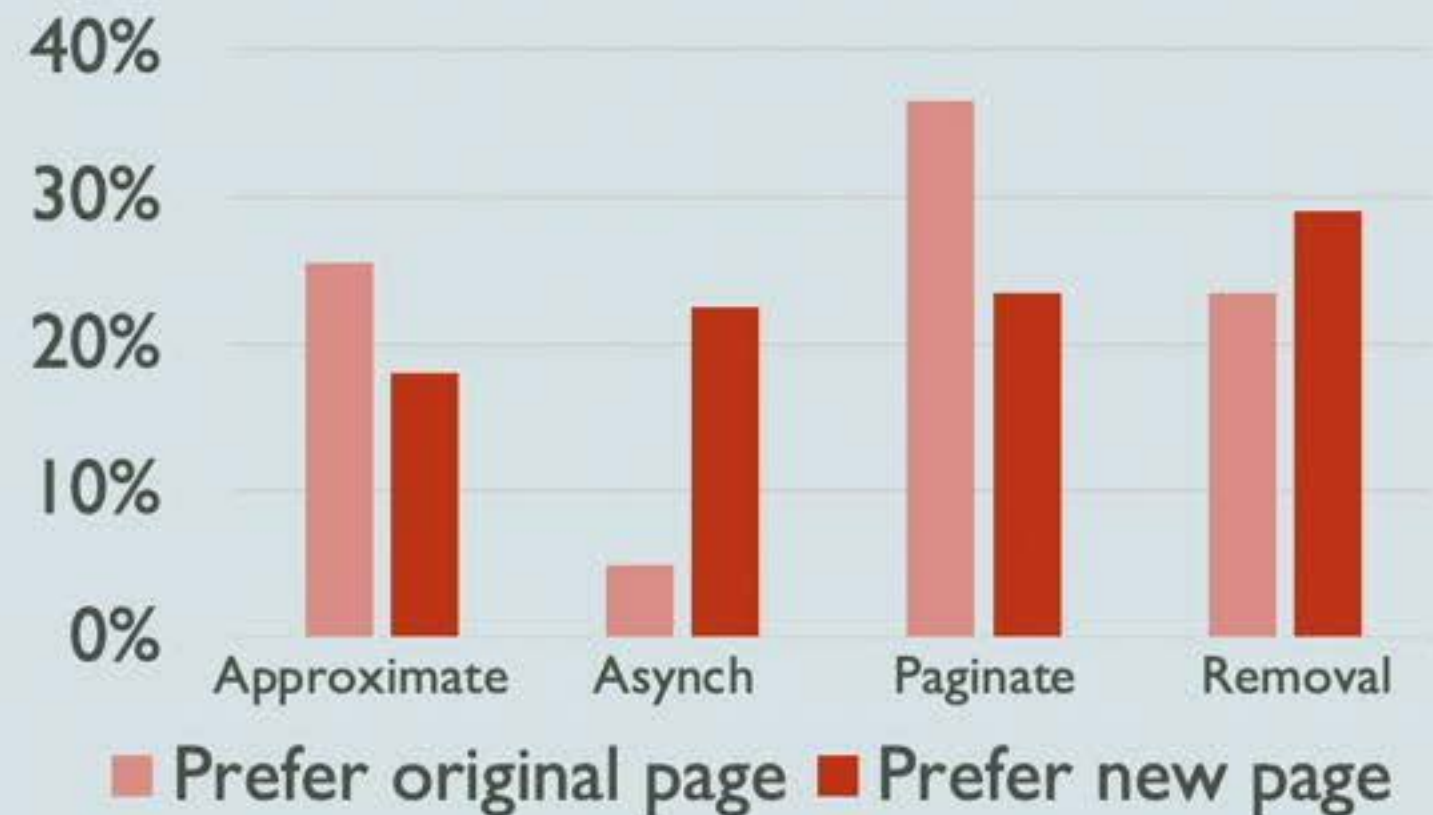
end-to-end speedup



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

user's satisfaction

Avg: 22% prefer original, 20% prefer new



user survey result on page preference

Outline

- Leveraging application semantics to optimize each layer

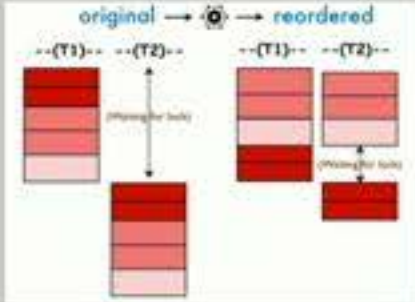
Panorama:
view-driven optimization



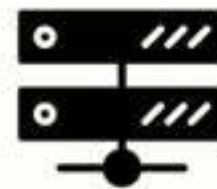
The diagram shows a browser window with a list of items. A search icon is present. Two arrows point from the browser to a gear icon labeled 'fast webpage', and another arrow points from the gear icon back to the browser, labeled 'slow webpage'.



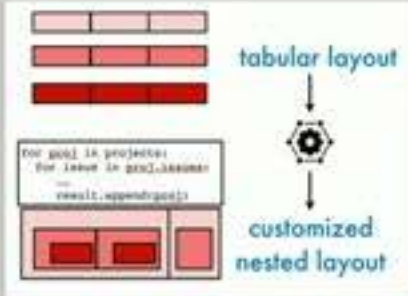
Quro:
reorder queries



The diagram shows two query execution plans. The left one is labeled 'original' and the right one 'reordered'. Both show a tree structure with nodes labeled '(T1)' and '(T2)'. Arrows indicate the flow of data between nodes. The 'reordered' plan shows a different sequence of operations compared to the 'original' plan.



Chestnut:
customize data layout



The diagram shows a SQL query:

```
for gobj in projects;
for item in gobj.items;
-- result.append(gobj);
```

 Above the query are three rows of colored blocks representing data layout. An arrow labeled 'tabular layout' points to a gear icon, and another arrow labeled 'customized nested layout' points to a different arrangement of colored blocks.



- Other projects
- Ongoing and future work

Outline

- Leveraging application semantics to optimize each layer

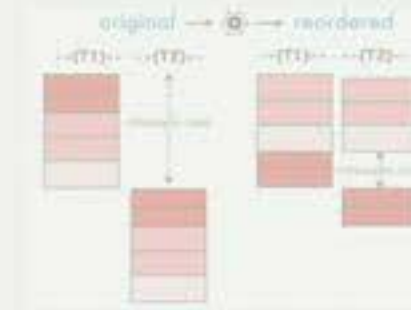
Panorama:
view-driven optimization



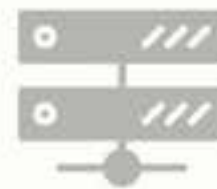
The diagram illustrates the process of view-driven optimization. It shows a 'slow webpage' on the left, which is a long, narrow list of items. An arrow points to a 'fast webpage' on the right, which is a more compact, scrollable view of the same data. A gear icon and a magnifying glass icon are also present, suggesting optimization and search capabilities.



Quoro:
reorder queries



The diagram shows two query execution plans. The 'original' plan on the left has a sequence of operations: a join (T1) followed by a join (T2). The 'reordered' plan on the right has a different sequence: a join (T2) followed by a join (T1). This illustrates how reordering queries can optimize execution.



Chestnut:
customize data layout



The diagram shows a 'tabular layout' on the left, which is a simple grid of data. An arrow points to a 'customized nested layout' on the right, which is a more complex, nested structure. A gear icon is placed between the two layouts, indicating a transformation or customization process.



- 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

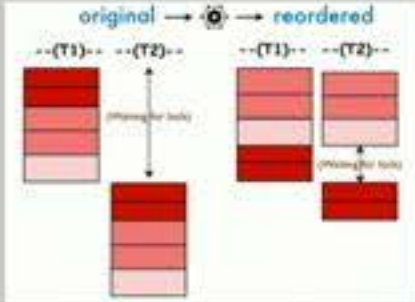
Panorama:
view-driven optimization



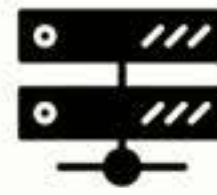
The diagram shows a browser window with a list of items. A search icon is present. Two arrows point from the browser to a gear icon, labeled 'slow webpage' and 'fast webpage', indicating the optimization process.



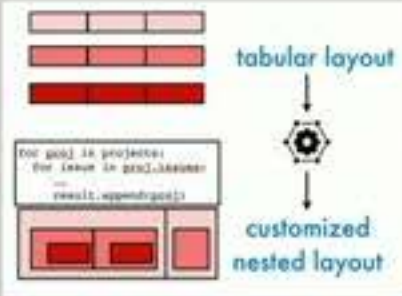
Quro:
reorder queries



The diagram shows a query execution plan. It starts with 'original' and 'reordered' labels. Below, there are two sets of red blocks representing data partitions. The first set is labeled '(T1)' and '(T2)'. The second set is labeled '(T1)' and '(T2)'. Arrows indicate the flow of data between these sets, showing how the order of operations is changed for optimization.



Chestnut:
customize data layout



The diagram shows a query execution plan. It starts with a 'tabular layout' and a 'customized nested layout'. Below, there is a code snippet:

```
for proj in projects:  
  for issue in proj.issues:  
    result.append(issue)
```

 The code is shown in a box, and arrows indicate the flow of data between the tabular and nested layouts.



- Other projects
- Ongoing and future work

Outline

- Leveraging application semantics to optimize each layer

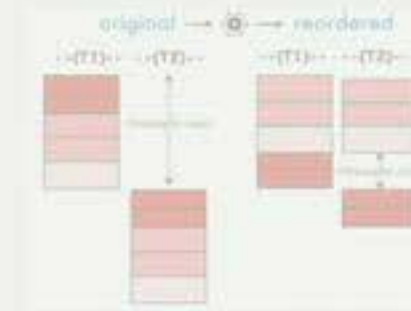
Panorama:
view-driven optimization



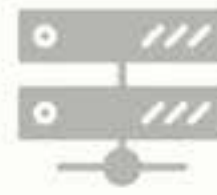
The diagram shows a vertical list of webpage elements on the left, with arrows pointing to a central 'slow webpage' and a 'fast webpage' on the right. The 'fast webpage' is a simplified version of the 'slow webpage' with only the most important elements visible.



Quoro:
reorder queries



The diagram shows two query execution plans. The 'original' plan has a root node connected to two leaf nodes (T1 and T2). The 'reordered' plan has a root node connected to a single leaf node (T1), which is then connected to another leaf node (T2).



Chestnut:
customize data layout



The diagram shows a 'tabular layout' of data with three rows and three columns. Below it, a 'customized nested layout' shows the same data organized into a more compact, nested structure.



- 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

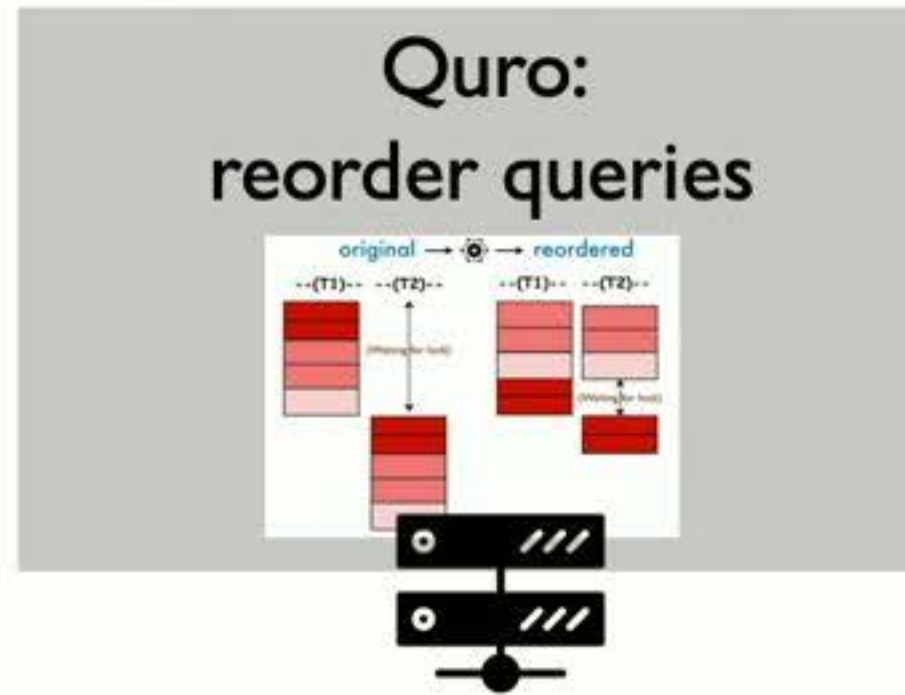
- More chances to leverage application semantics
 - E.g., using data constraint for query optimization
 - using client-level statistics (e.g., click rate) to help query optimization
 - ...
- Building a “smarter” middleware that:
 - Incorporate static/dynamic cross-stack optimizations
 - Provide analysis interface to support adding optimization

Summary

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

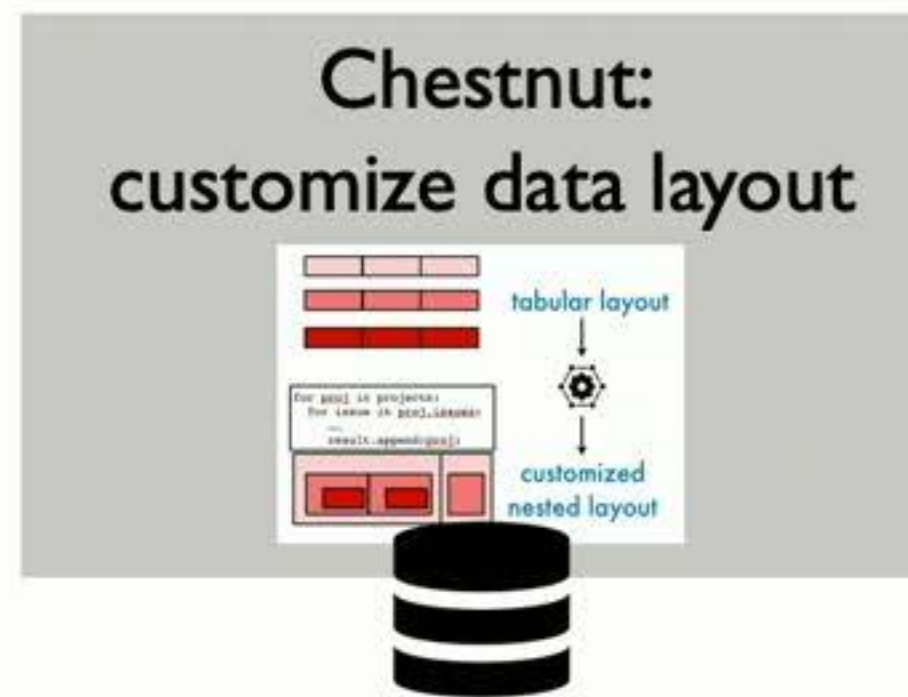
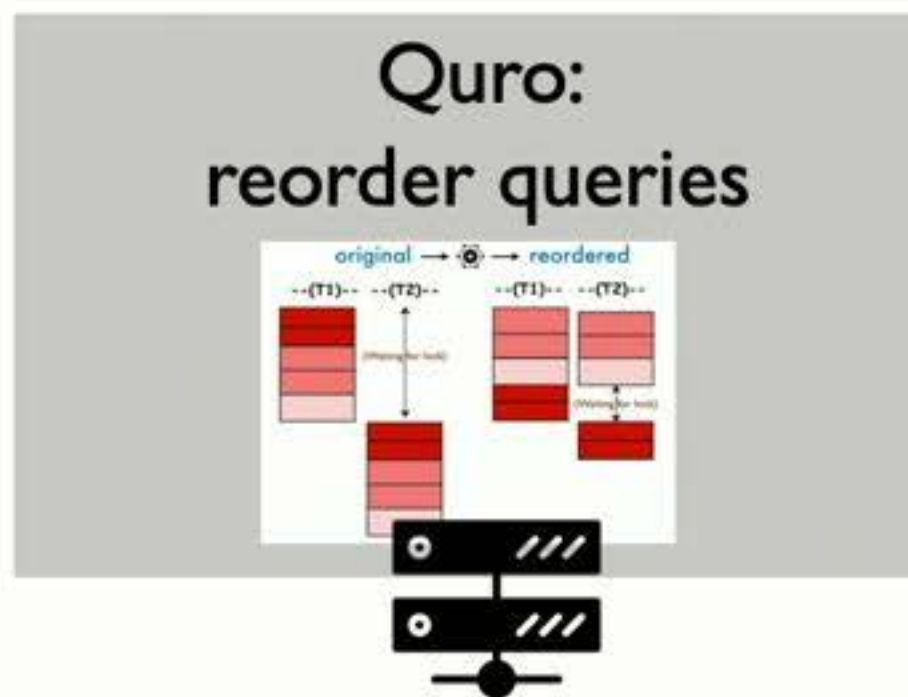
Summary

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



Summary

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



Summary

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

Panorama:
view-driven optimization

The diagram shows a computer monitor displaying two versions of a webpage. The top version is labeled 'slow webpage' and the bottom version is labeled 'fast webpage'. A gear icon is positioned between the two versions, indicating an optimization process. The 'fast webpage' version shows a more compact layout with elements repositioned to reduce scrolling.

Quro:
reorder queries

The diagram illustrates the reordering of database queries. It shows an 'original' query plan with two parallel paths, each containing a join operation (T1) and a join operation (T2). A gear icon indicates the reordering process. The 'reordered' query plan shows the join operations rearranged to optimize the execution flow. Below the diagrams is a server rack icon.


Chestnut:
customize data layout

The diagram shows the customization of data layout. It starts with a 'tabular layout' represented by a grid of red blocks. A gear icon indicates the transformation process. The resulting 'customized nested layout' shows the data reorganized into a more efficient nested structure. Below the diagrams is a database cylinder icon.

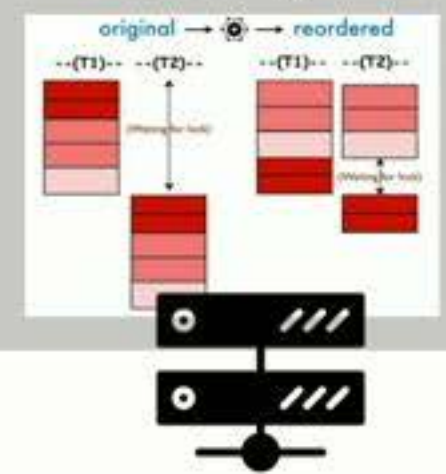
Summary

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

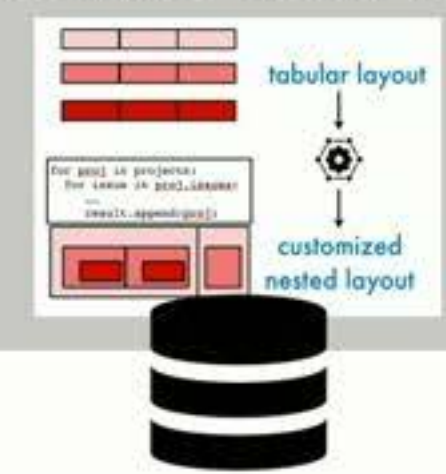
Panorama:
view-driven optimization



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