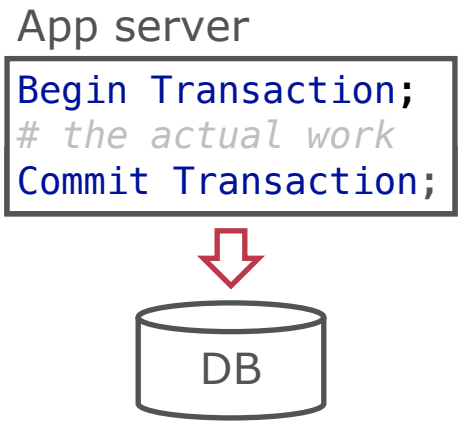
Ad Hoc Transactions in Web Applications:
The Good, the Bad, and the Ugly

Chuzhe Tang, Zhaoguo Wang, Xiaodong Zhang, Qianmian Yu
Binyu Zang, Haibing Guan, Haibo Chen

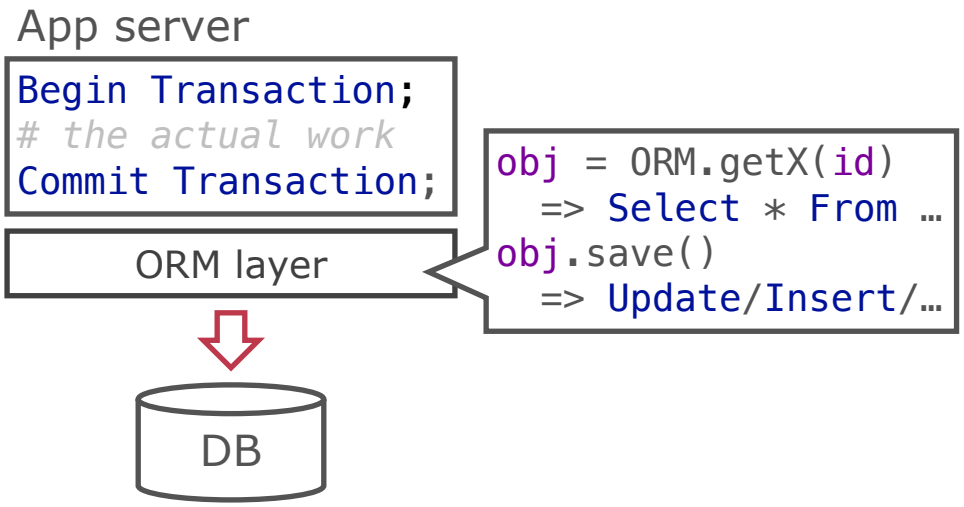
How do applications today use concurrency control?

The intuitive answer, database transactions.



How do applications today use concurrency control?

The intuitive answer, database transactions.



How do applications today use concurrency control?

The intuitive answer, database transactions.

Bailis et al. identified another application-level approach, invariant validation*.

- Developers specify invariants; ORMs validate them.
- E.g., uniqueness of a column's values (w/o using DB unique constraint).

*Bailis et al. Feral Concurrency Control: An Empirical Investigation of Modern Application Integrity. SIGMOD '15.

How do applications today use concurrency control?

In this study, we investigate the third approach, ad hoc transactions.

- They are the "transactions" coordinated by ad hoc constructs (e.g., locks) employed by app developers.

App server (add-cart API)

```
lock(cart_id)
# perform business logic
# use ORM to access DB
unlock(cart_id)
```

Server-side lock table

cart	locked
1	true
2	false

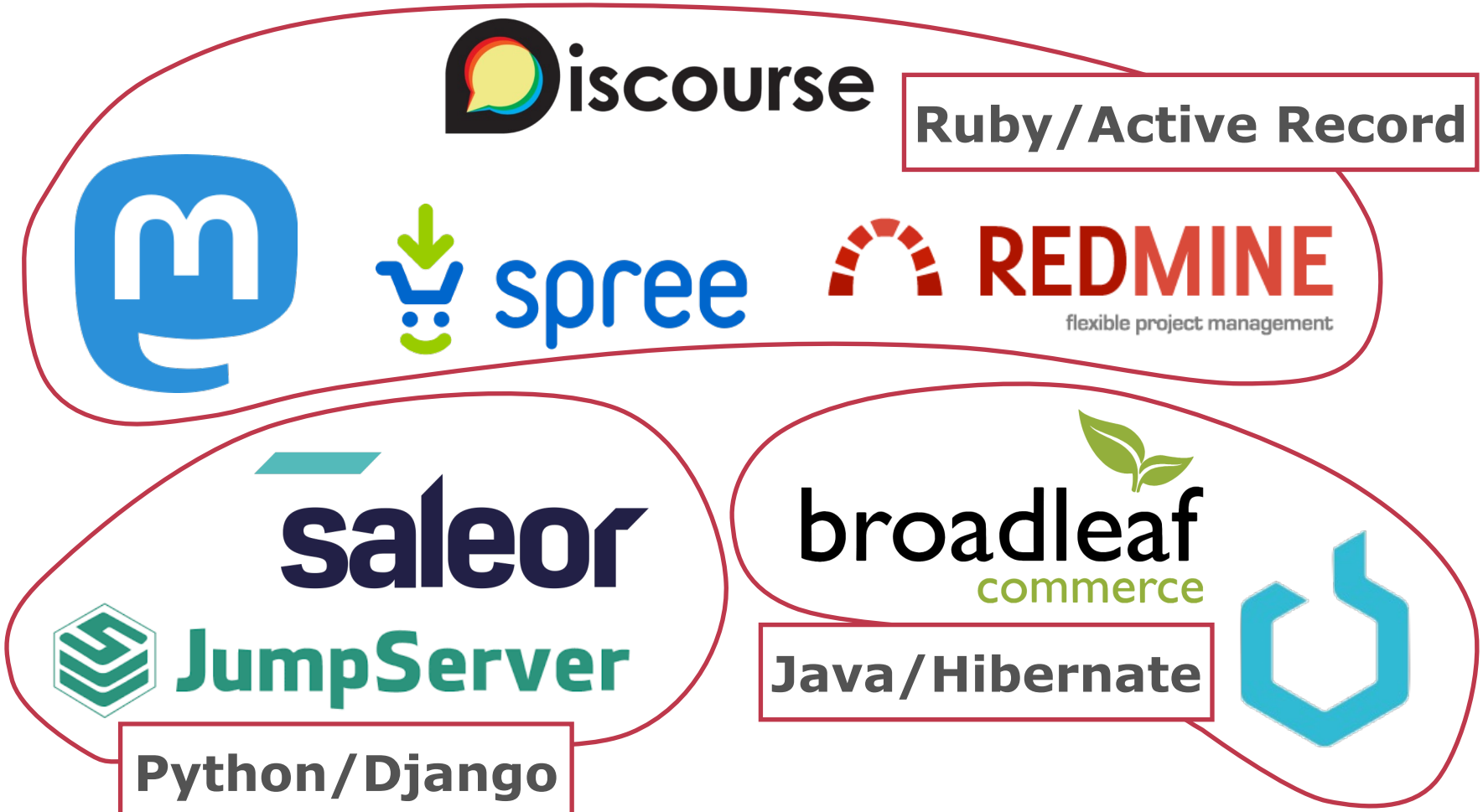
Plain `Select`/`Update`/`Insert`/`Delete` (without DB transactions)

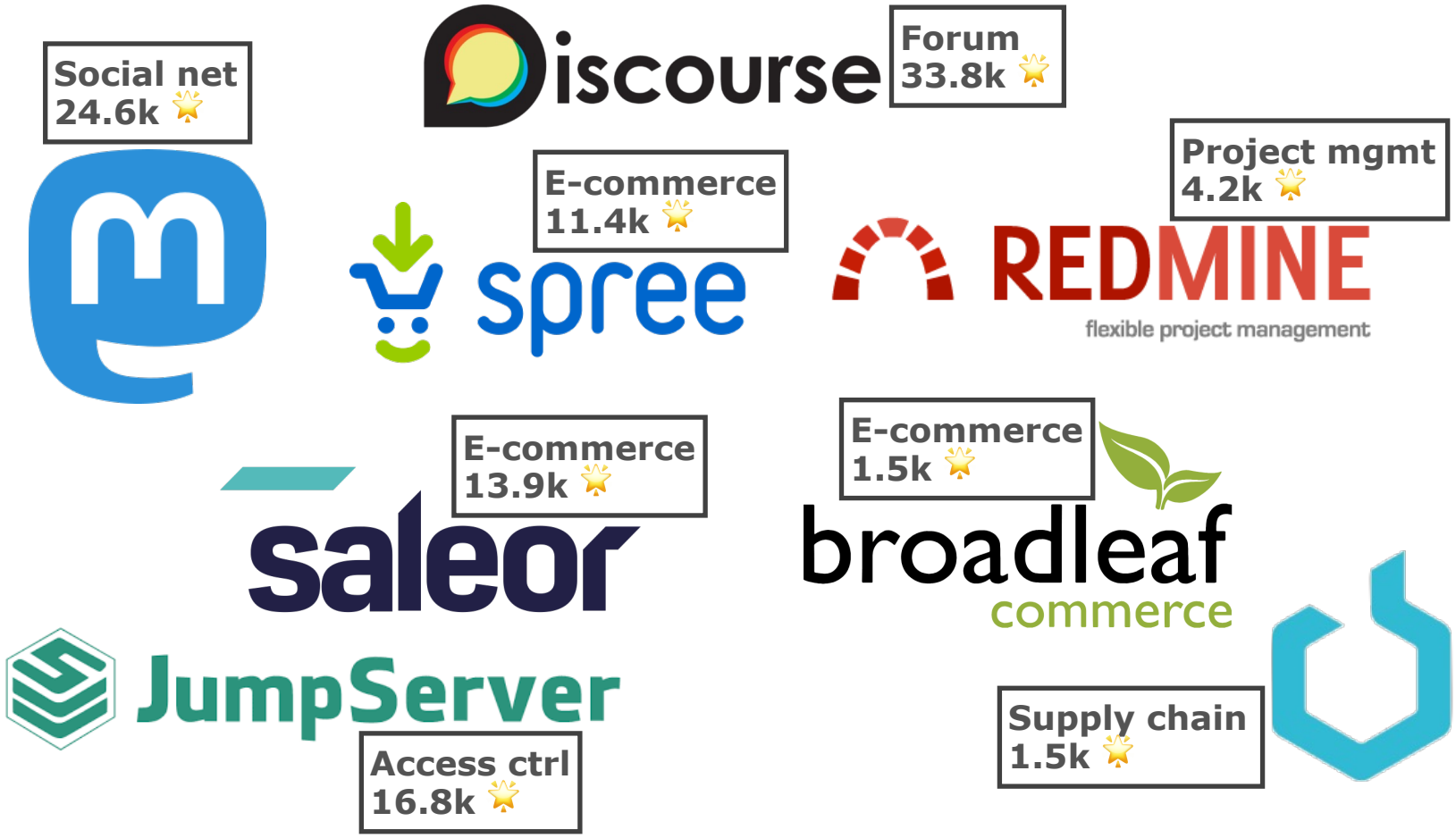
Example is simplified from the Broadleaf e-commerce application.

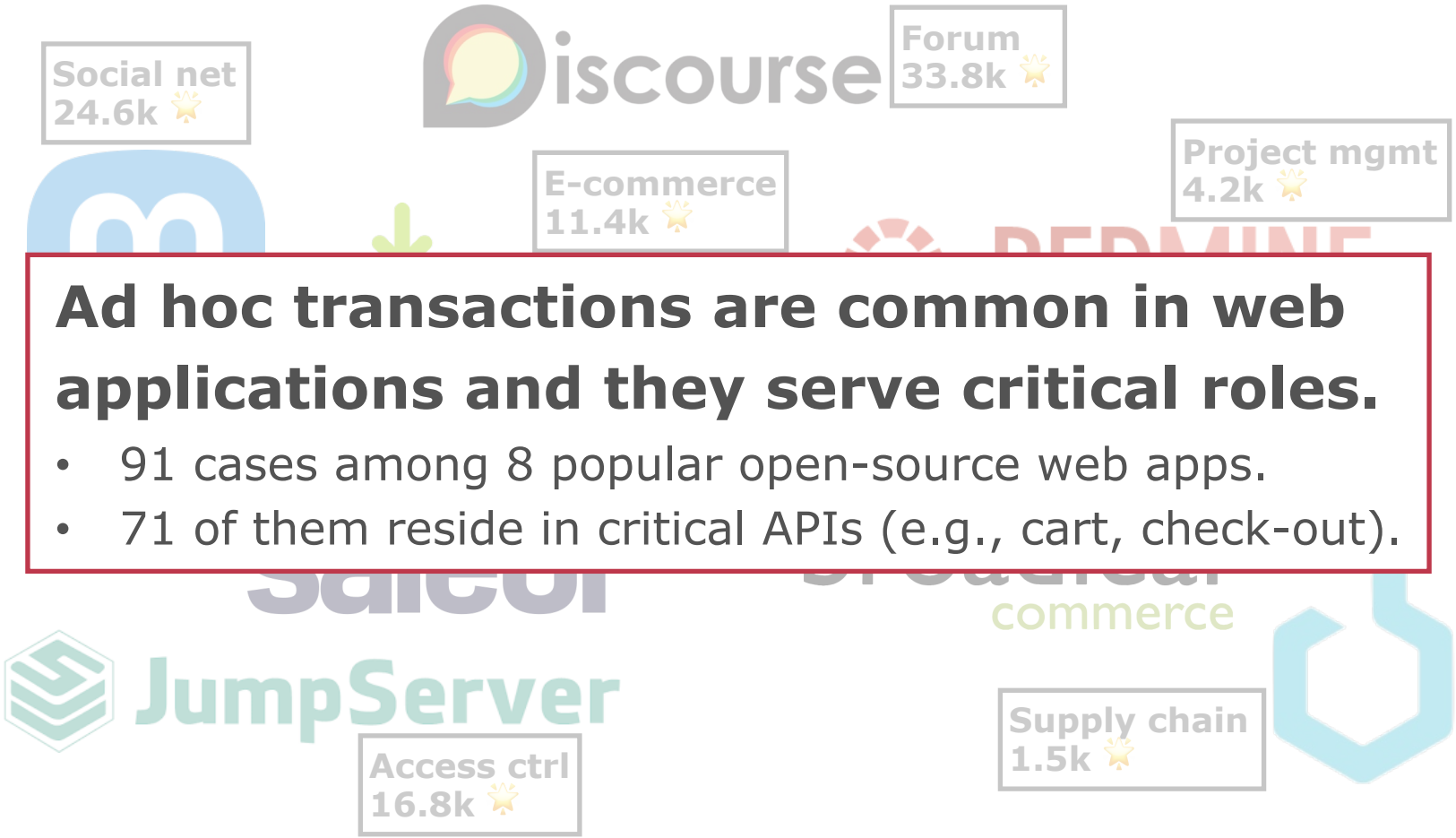
Ad hoc transactions represent the third approach

	Database transactions	**Invariant validation**	**Ad hoc transactions**
WHAT is protected?	Business logic snippets	Invariants on Database rows	Business logic snippets
WHO conducts the protection?	Database CC	ORMs	Developers

What is the state of the practice?


Discourse
Ruby/Active Record
spree
REDMINE
flexible project management
saleor
JumpServer
Python/Django
broadleaf
commerce
Java/Hibernate


Social net
24.6k
Forum
33.8k
Discourse
Project mgmt
4.2k
E-commerce
11.4k
spree
REDMINE
flexible project management
E-commerce
13.9k
saleor
E-commerce
1.5k
broadleaf
commerce
JumpServer
Access ctrl
16.8k
Supply chain
1.5k



Ad hoc transactions are common in web applications and they serve critical roles.

- 91 cases among 8 popular open-source web apps.
- 71 of them reside in critical APIs (e.g., cart, check-out).

In the rest of this talk

Answer the following questions.

- **How do ad hoc transactions look like?**
  They look diverse in many aspects.
- **Are they correct?**
  More than half are incorrect.
- **Do they perform well?**
  They sometimes outperform DB transactions.

Discuss the implications.

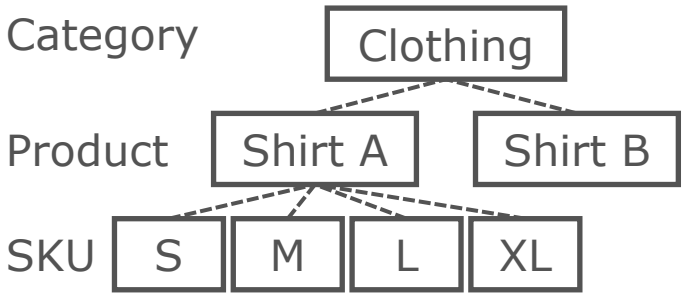
Ad hoc transactions have diverse semantics

They can coordinate DB operations partially.

- 22/91 cases coordinate partially.

```
lock(sku_id)
sku = ORM.getSku(sku_id)
if sku.qty > want:
  sku.qty -= want
  sku.save()
unlock(sku_id)
```

```
Update sku.qty                    # SKUs table
Update sku.prod.update_time       # Products table
for category in sku.prod.cats:
  Update category.update_time # Categories table
```



Example is simplified from the Spree e-commerce application.

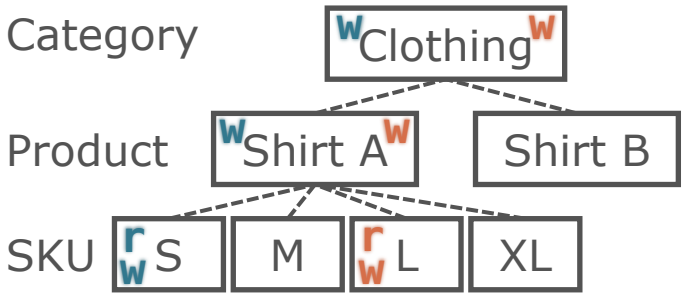
Ad hoc transactions have diverse semantics

They can coordinate DB operations partially.

- 22/91 cases coordinate partially.

```
lock(sku_id)
sku = ORM.getSku(sku_id)
if sku.qty > want:
  sku.qty -= want
  sku.save()
unlock(sku_id)
```

```
Update sku.qty                    # SKUs table
Update sku.prod.update_time       # Products table
for category in sku.prod.cats:
  Update category.update_time # Categories table
```



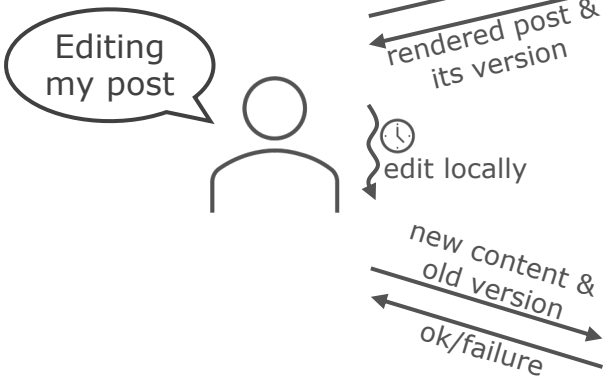
User 1 buys a small shirt A
User 2 buys a large shirt A
Only SKU ops are serialized!

Example is simplified from the Spree e-commerce application.

Ad hoc transactions have diverse semantics

Their coordination can span many requests.

- 10/91 cases coordinate across multiple requests.




```
Begin Transaction
post = ORM.getPost(post_id)
return render(post)
```

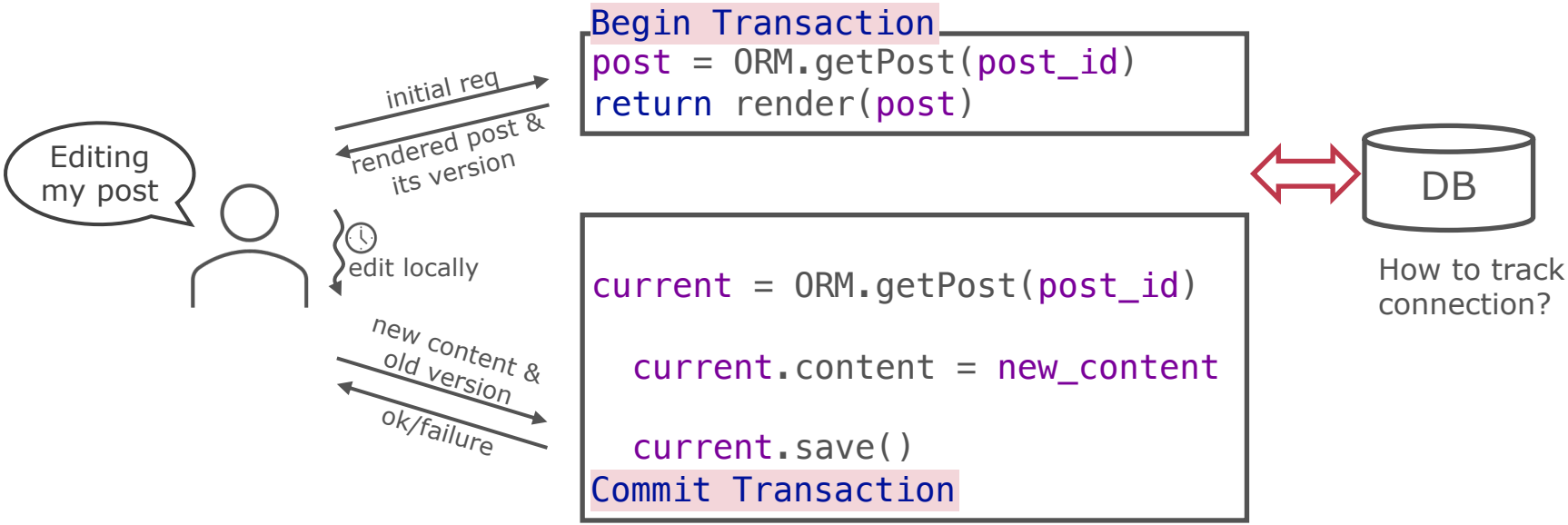
```
lock(post_id)
current = ORM.getPost(post_id)
if current.ver != prev_version:
  current.content = new_content
  current.ver += 1
  current.save()
Commit Transaction
```

Example is simplified from the Discourse forum application.

Ad hoc transactions have diverse semantics

Their coordination can span many requests.

- 10/91 cases coordinate across multiple requests.

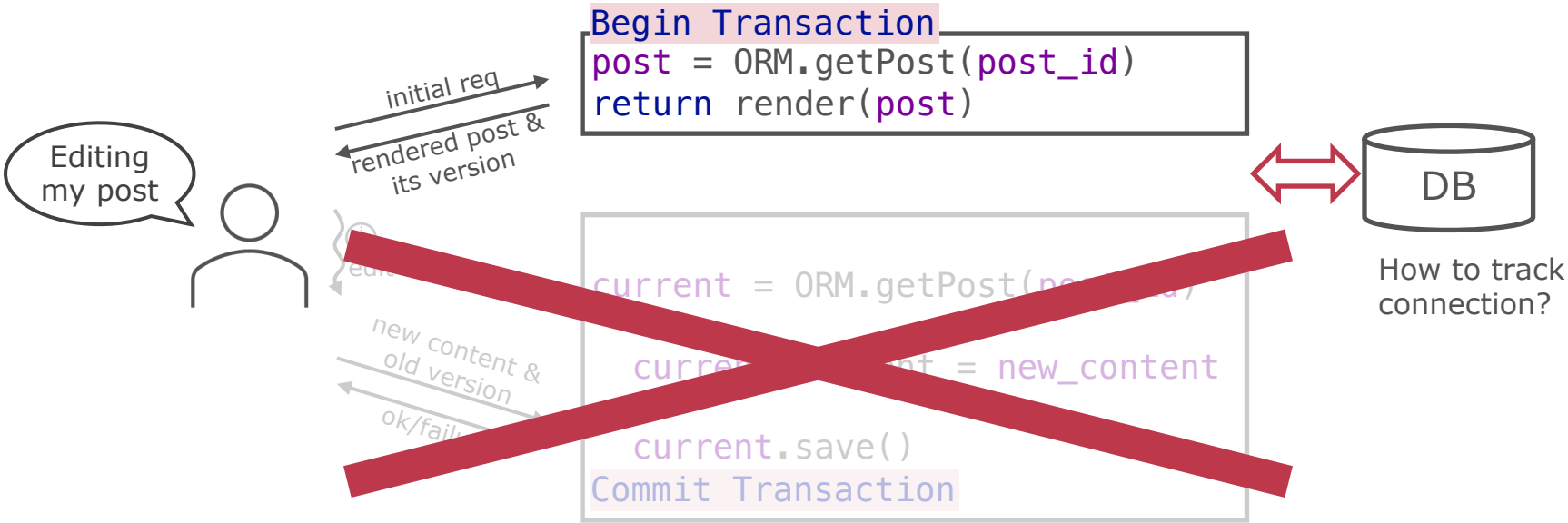


Example is simplified from the Discourse forum application.

Ad hoc transactions have diverse semantics

Their coordination can span many requests.

- 10/91 cases coordinate across multiple requests.

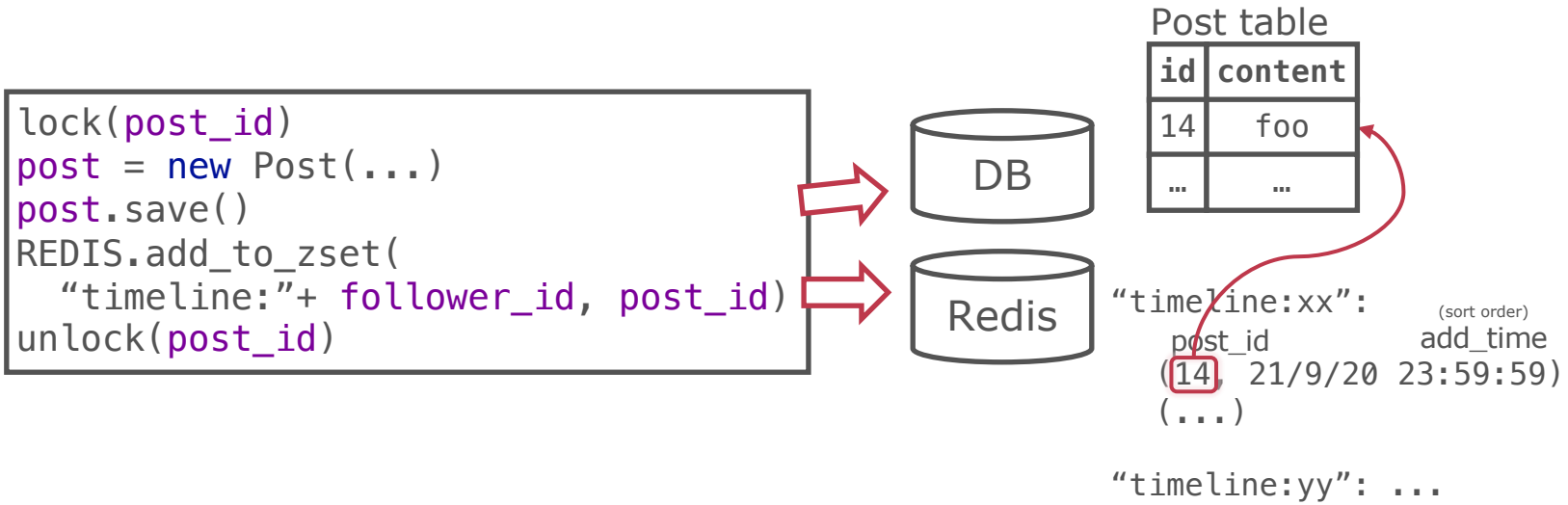


Example is simplified from the Discourse forum application.

Ad hoc transactions have diverse semantics

They can also coordinate non-DB operations.

- 8/91 cases handle non-DB operations.



Example is simplified from the Mastodon social network application.

Ad hoc transactions have diverse implementations

They use either locks or validation procedures for coordination.

For locks, there are 7 different implementations among 8 applications.

- 2 implementations reuse existing locking facilities.

```
store = loadStoreInfo(...)
synchronized(store) # Java keyword
  { /* perform work here */ }
```

Example is simplified from the SCM Suite supply chain management application.

Ad hoc transactions have diverse implementations

They use either locks or validation procedures for coordination.

For locks, there are 7 different implementations among 8 applications.

- 2 implementations reuse existing locking facilities.
- 2 implementations store lock information in Redis.

```
REDIS.set_if_not_exist(lock_key)
# perform work here
REDIS.delete(lock_key)
```

Redis

lock_key	value
“post66”	{owner :xx, expire:yy}
…	…

Example is simplified from the Mastodon social network application.

Ad hoc transactions have diverse implementations

They use either locks or validation procedures for coordination.

For locks, there are 7 different implementations among 8 applications.

- 2 implementations reuse existing locking facilities.
- 2 implementations store lock information in Redis.
- 1 implementation stores lock information in DB tables.
- 2 implementations store lock information in application runtime containers (e.g., Java’s ConcurrentHashMap).

Ad hoc transactions have diverse implementations

They use either locks or validation procedures for coordination.

For locks, there are 7 different implementations among 8 applications.

For validation procedures, there are also 2 categories.

- ORMs can generate some according to annotations. (4 apps)
- Developers also implement the others from scratch. (3 apps)
  - They need to ensure the check-and-update atomicity.

How do ad hoc transactions look like?

They have diverse semantics.

- Partial coordination.
- Multi-request coordination.
- Coordinating non-DB operations.

They have diverse implementations.

- 7 for locks and 2 for validation.

They coordinate at diverse granularity.

- Customized with developers' domain knowledge.
- Either coarser or finer than DB transactions.

They handle failure differently.

- Usually simpler than DB transactions.

Are ad hoc transactions correct?

69 correctness issues are found in 53 cases.*

- 33 cases' issues are confirmed by developers.

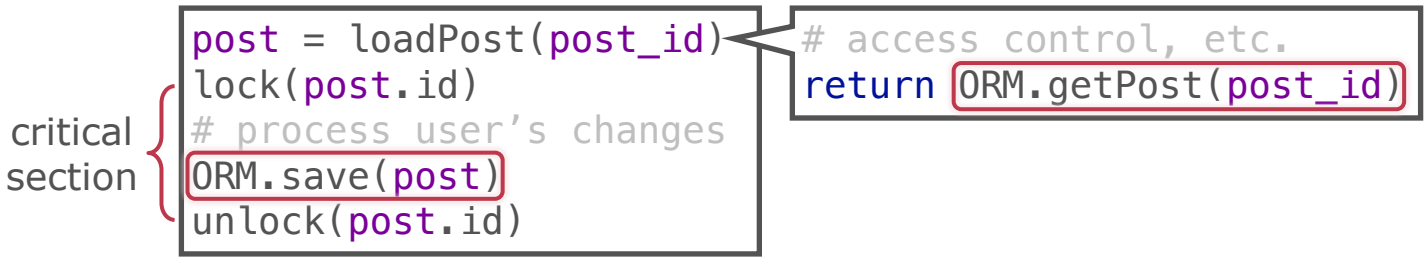
We consider 28 of them severe.

App.	Known severe consequences	Cases
Discourse	Overwritten post contents, page rendering failure, excessive notifications.	6
Mastodon	Showing deleted posts, corrupted account info., incorrect polls.	4
Spree	Overcharging, inconsistent stock level, inconsistent order status, selling discontinued products.	9
Broadleaf	Promotion overuse, inconsistent stock level, inconsistent order status, overselling.	6
Saleor	Overcharging.	3

*Some cases suffer from multiple issues.

Majority of issues stem from wrong locking/validation primitives

36/65 lock-based ad hoc transactions wrongly implement or use locking primitives.



Example is simplified from the Discourse forum application.

Majority of issues stem from wrong locking/validation primitives

36/65 lock-based ad hoc transactions wrongly implement or use locking primitives.

11/26 validation-based ad hoc transactions failed to ensure check-and-update atomicity.

```
ORM.transaction:
  ok = MiniSql.query(
    Update Posts Set version=version+1
    Where id=id And version=version)
  if not ok:
    ORM.abort_transaction()
  # perform updates here
```

Majority of issues stem from wrong locking/validation primitives

36/65 lock-based ad hoc transactions wrongly implement or use locking primitives.

11/26 validation-based ad hoc transactions failed to ensure check-and-update atomicity.

```
ORM.transaction:
  ok = MiniSql.query(
    Update Posts Set version=version+1
    Where id=id
  if not ok:
    ORM.abort_transaction()
  # perform updates here
```

MiniSql is independent of the ORM, thus issuing Update outside of the DB transaction.

Developers sometimes wrongly employ ad hoc transactions

16 issues are caused by incorrect scopes.

- Developers might omit critical operations from coordination in existing ad hoc transactions. (11 cases)
- Developers might forget to employ ad hoc transactions for conflicting procedures. (5 cases)

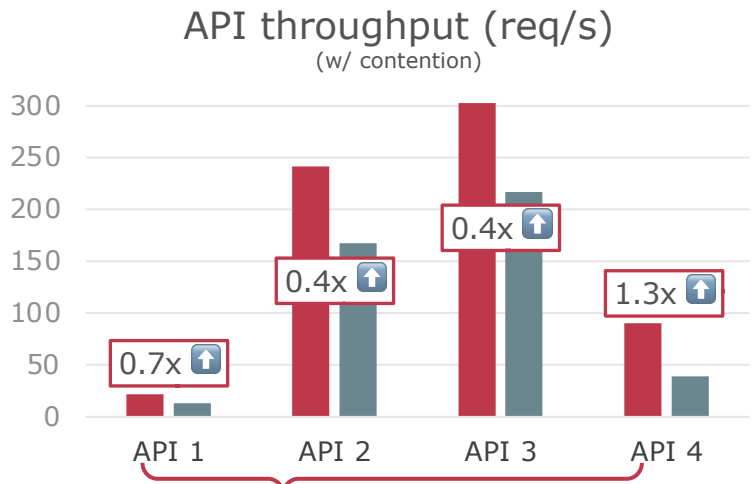
4 issues are caused by incorrect failure handling.

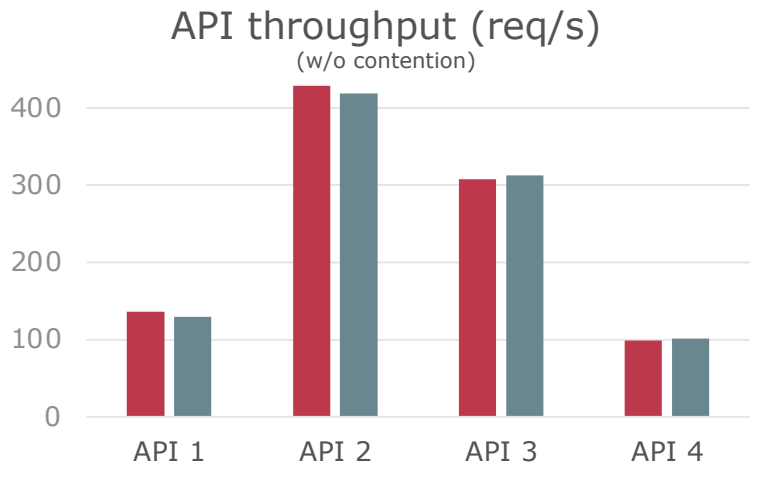
- E.g., crash during ad hoc transactions introduce invalid intermediate states that cause user blocking after reboot.

Do ad hoc transactions perform well?

We deployed the applications and evaluated a subset of APIs with synthetic workloads.

- In comparison with codebase modified to use DB transactions.





Study summary

How do ad hoc transactions look like?

- They look diverse in semantics, implementation, coordination granularities, and failure handling.

Are they correct?

- More than half are incorrect; many issues are severe.

Do they perform well?

- They can outperform DB transactions under contention.

What does it imply?

Why do developers not use DB transactions?

- DB transactions lack important functionalities/properties?
- DB transactions need better integration with applications?
- Or applications are fine with relaxed ACID semantics?

What should we do?

- Further investigate why developers use ad hoc transactions.
- Explore new concurrency abstraction to better suit applications today.
- Build tools to improve existing applications that rely on ad hoc transactions.

Conclusion

Ad hoc transactions are a common approach to concurrency control in web applications.

- They have unique and diverse characteristics in their design and implementation.
- They usually have correctness issues due to their ad hoc nature.
- They have the potential to improve application performance in specific cases.

Their existence presents great opportunities for improving real-world DB applications.