Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Object-oriented Enterprise Application Programming Model for In-Memory Databases

Minimal Projection Enforcement

31. Mai 2012 | Jonas Witt

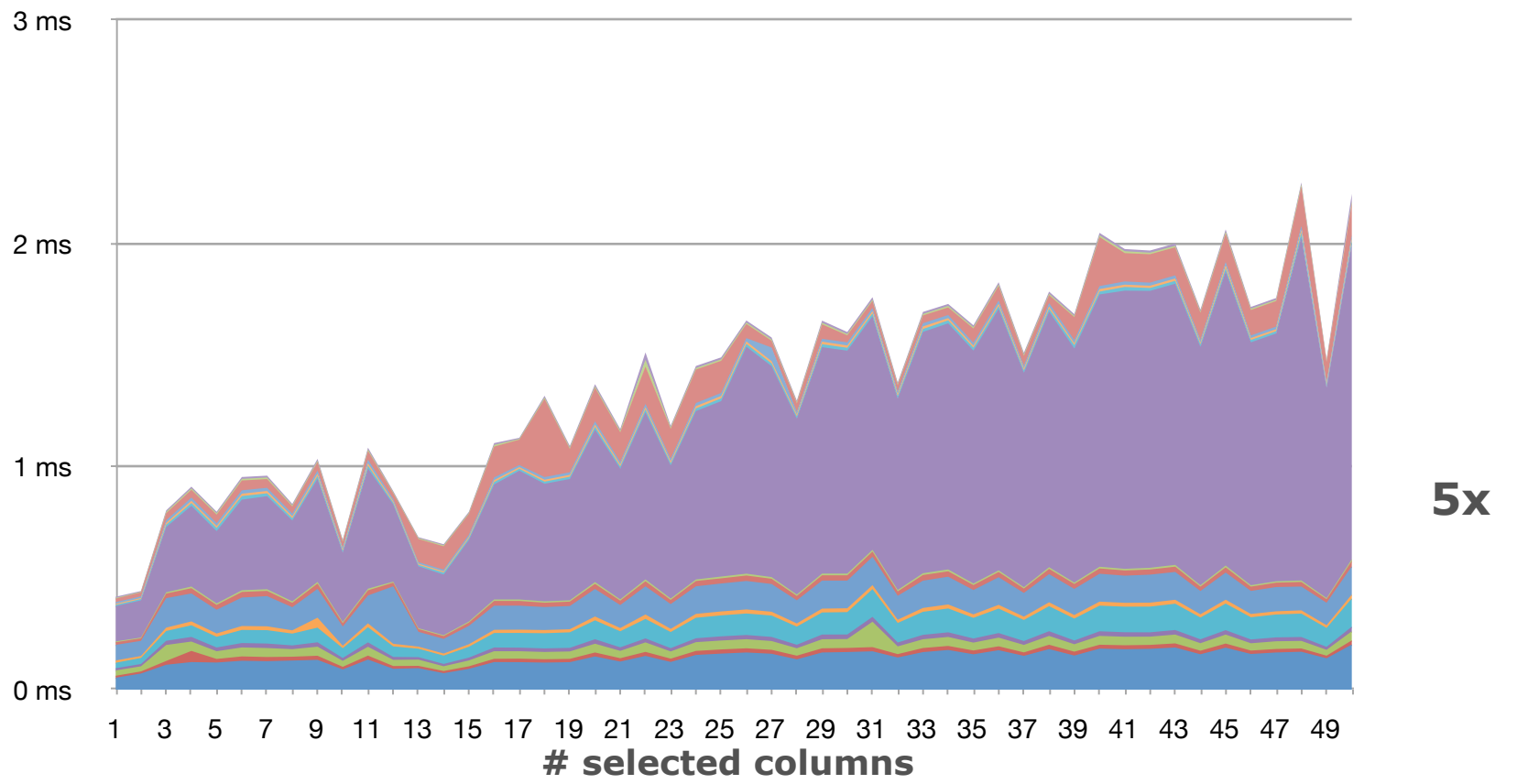
Minimal Projections

- AKA “How to avoid SELECT *”
- As a developer, it’s convenient to write SELECT * and don’t think about which attributes are going to be used
  - See analysis of SAP ERP code
- ORMs use SELECT * / full projections by default
- Problem:
  - It’s slower, since unused attributes are transmitted to the client
- Solutions:
  - Educate developers about the issue (?)
  - Make ORMs smarter

How serious is this?

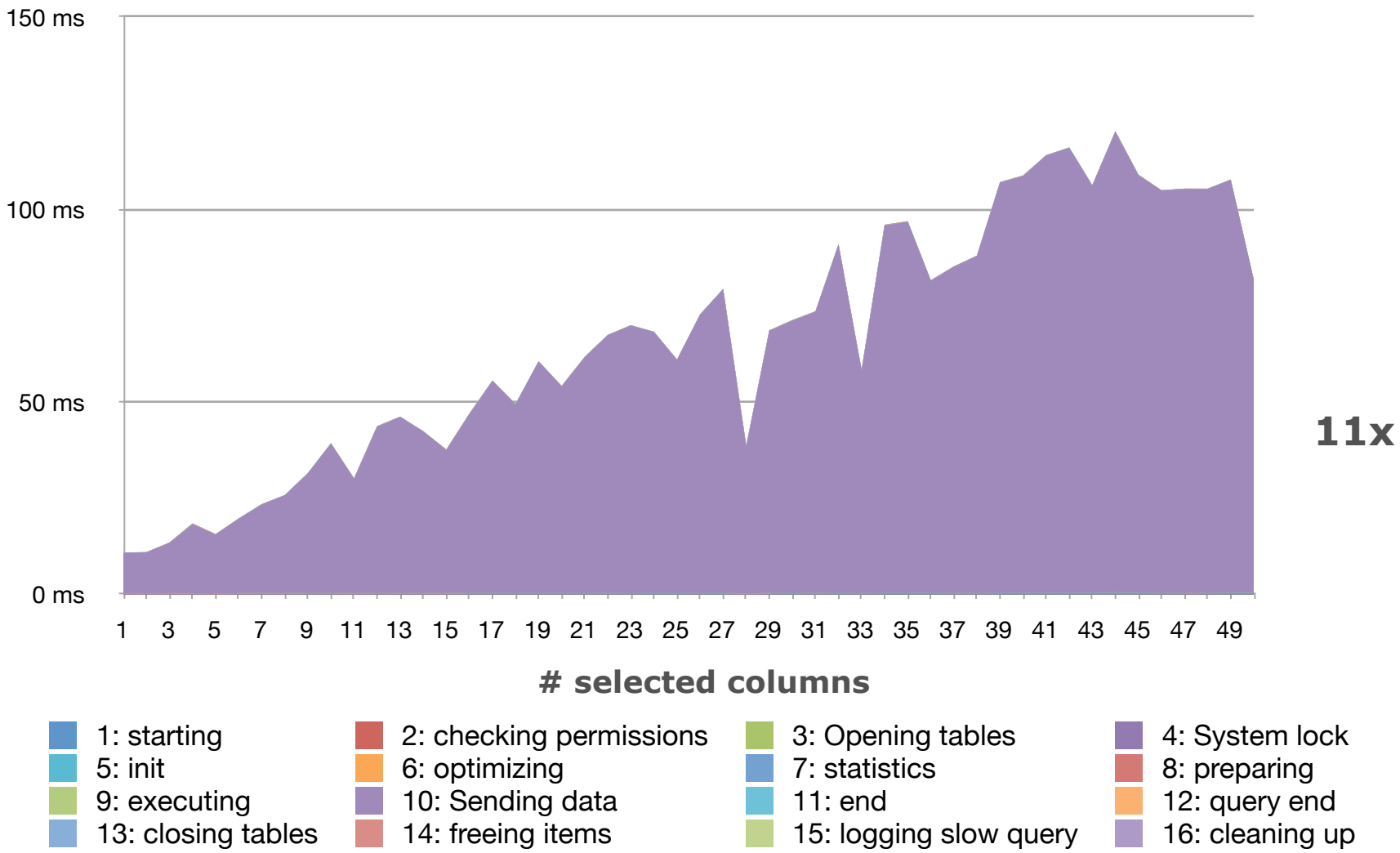
- Tested on MySQL 5
  - TODO: test on column-based DB
- 50 columns (mixed varchar(255) & integer)
  - Average row length = 850 Bytes
- Vary number of SELECTed columns
  - Fastest case: SELECT column1 FROM table
  - Worst case: SELECT * FROM table
- Use MySQL profiler to analyze where time is spent

SELECT 50 rows

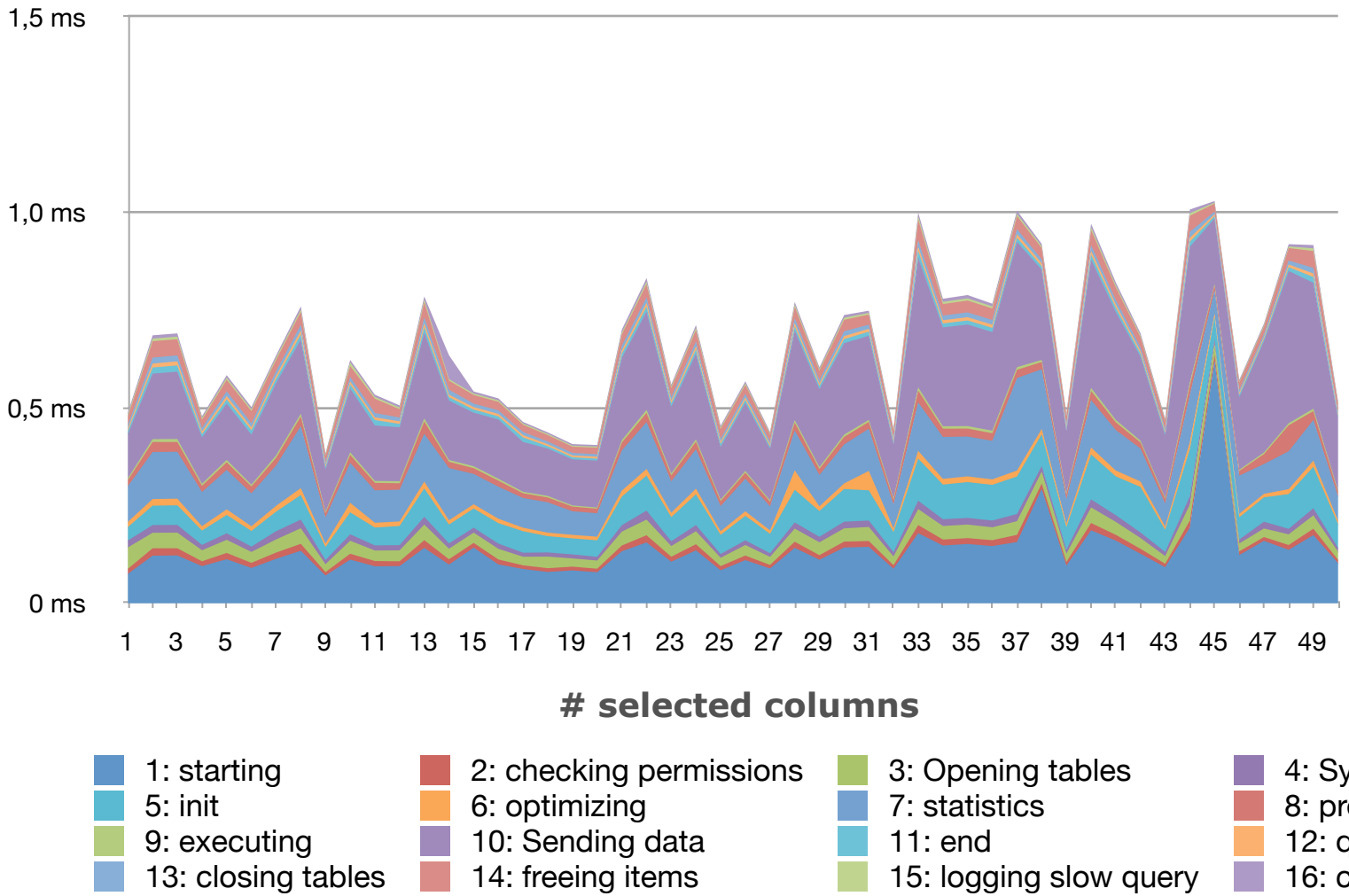


5x

SELECT 5000 rows



SELECT 5 rows



1.2x

Optimized ORM SELECT

- 1) Stats Collection Phase
  - Use query backtrace as unique identifier
  - SELECT *
  - record which attributes have been accessed
- 2) Optimized Execution Phase
  - Lookup stats using query identifier
  - SELECT only attributes that are likely to be used & PKs
  - When unfetched attributes are accessed, load them using the record's PK & update stats

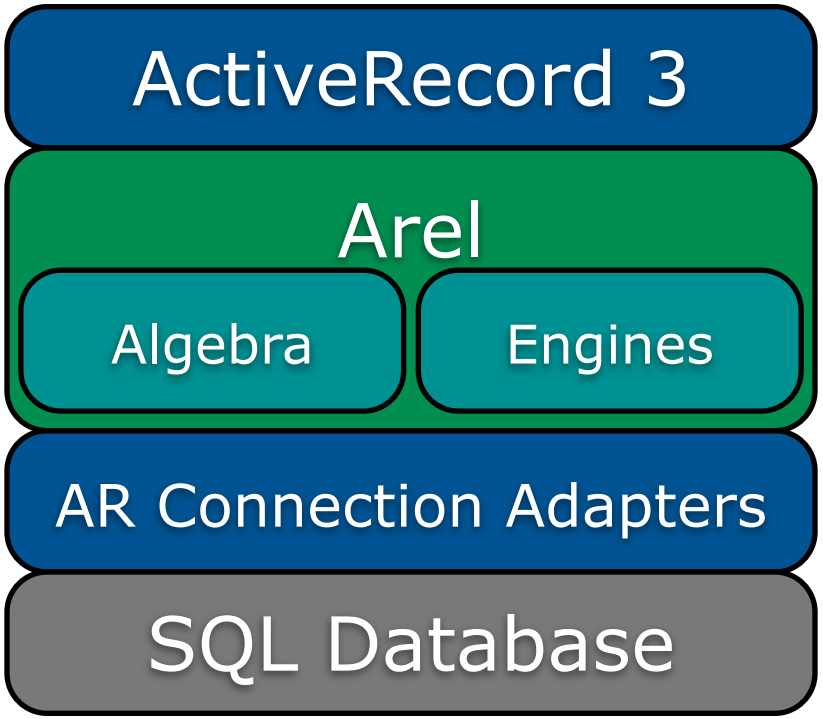
ORM Test Case

- ActiveRecord: default "ORM" used in Ruby on Rails
  - Active record pattern + inheritance + associations
- Arel: An object-oriented interpretation of relational algebra in Ruby

```
unis = University.where(:city
=> "Potsdam").order("size
DESC").limit(10)

unis = unis.select(:name)
```

Ruby on Rails 3:



http://www.slideshare.net/brynary/arel-ruby-relational-algebra