In-Memory Computing SUMMIT | NORTH AMERICA 2018

# Ultra-Low Latency with Java and Terabytes of Data

Per Minborg
CTO, Speedment, Inc.

Speedment

Per Minborg
CTO Speedment, Inc.

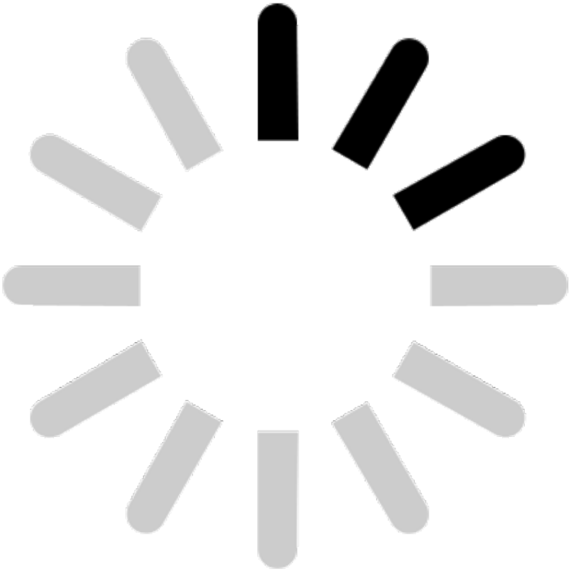
JavaOne™

DEVNEXUS™
the professional developer conference

INDEX
SAN FRANCISCO

In-Memory Computing

UNITED STATES PATENT AND TRADEMARK OFFICE
DEPARTMENT OF COMMERCE



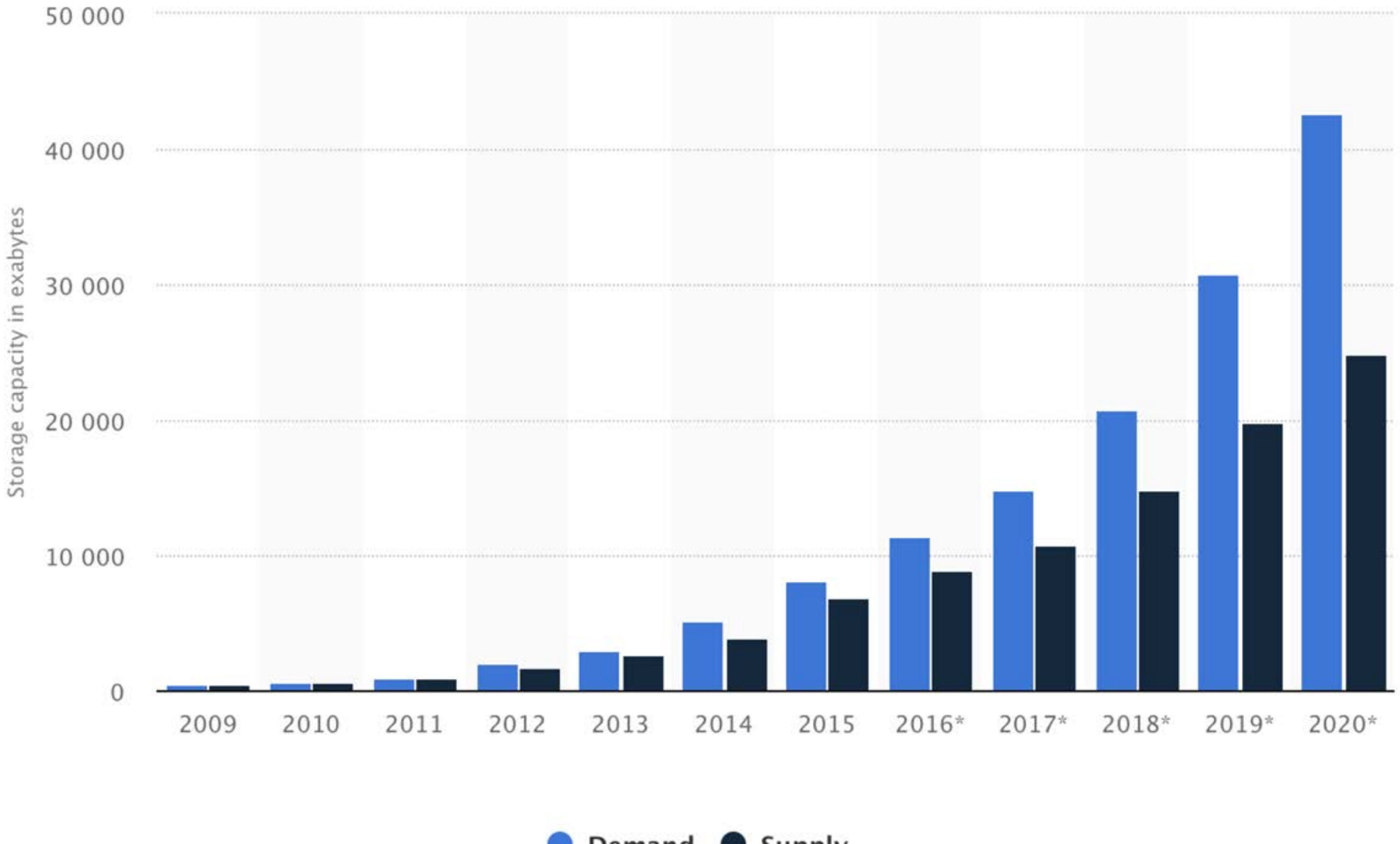
# Ultra-Low latency = 200 ns

# Why Are Applications Slow?

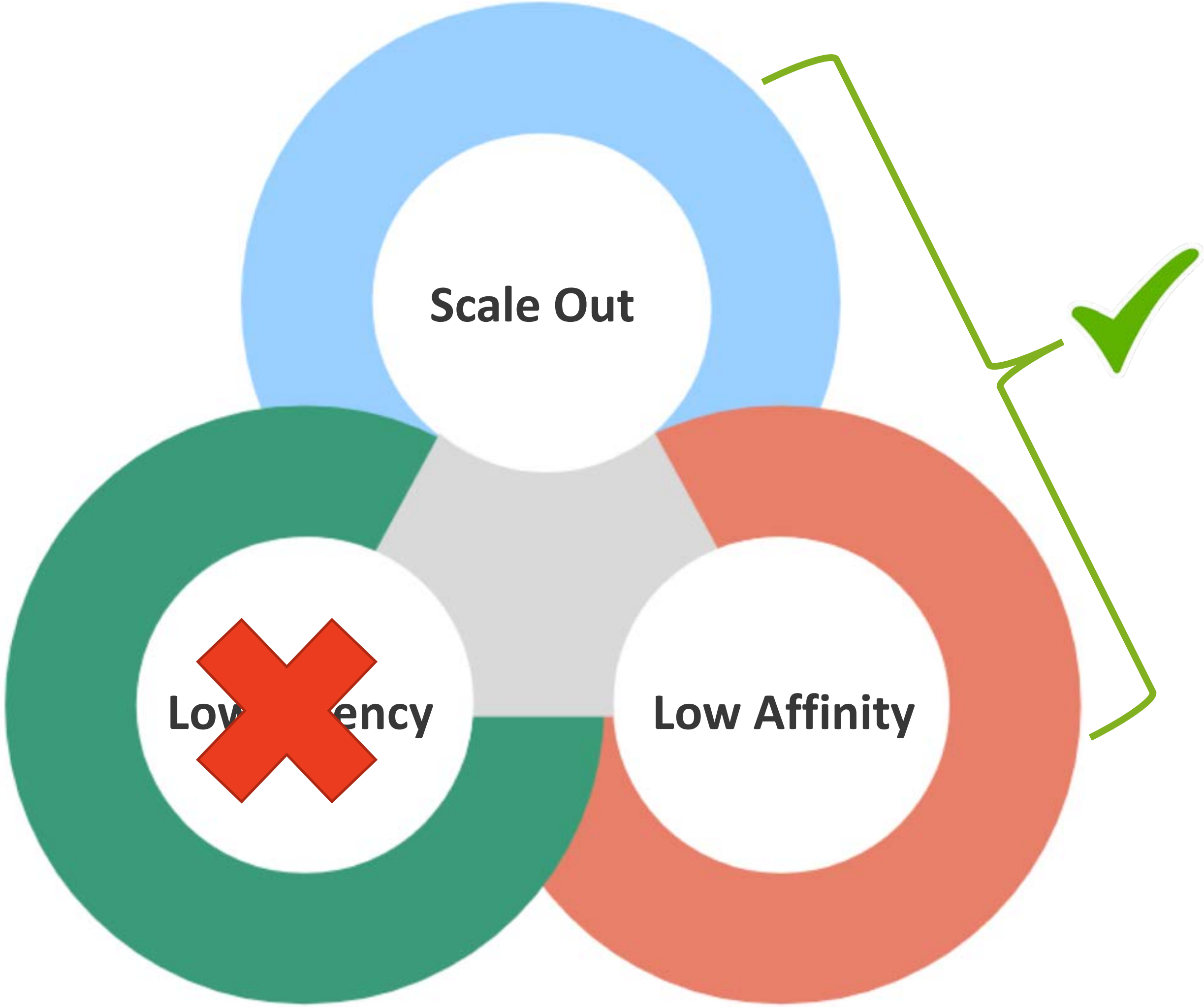
- Slow Databases
- Data on Several Nodes and no Affinity Across Data
- Data is Remote
- Unnecessary Object Creation / Garbage Collect Problem
- Lack of Parallelism

# Why Are Applications Slow? Slow Databases

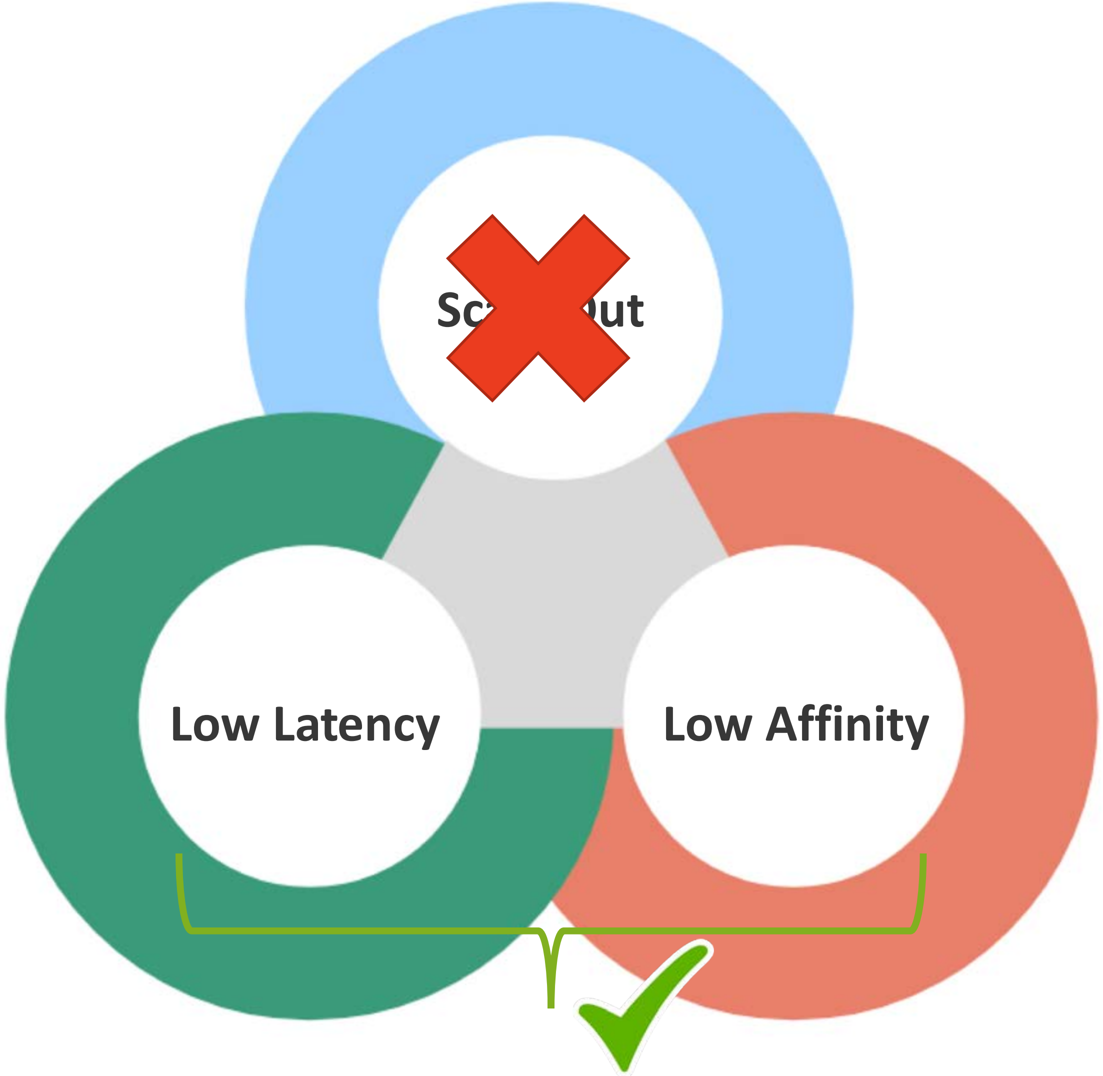
Data grows exponentially, which clogs systems



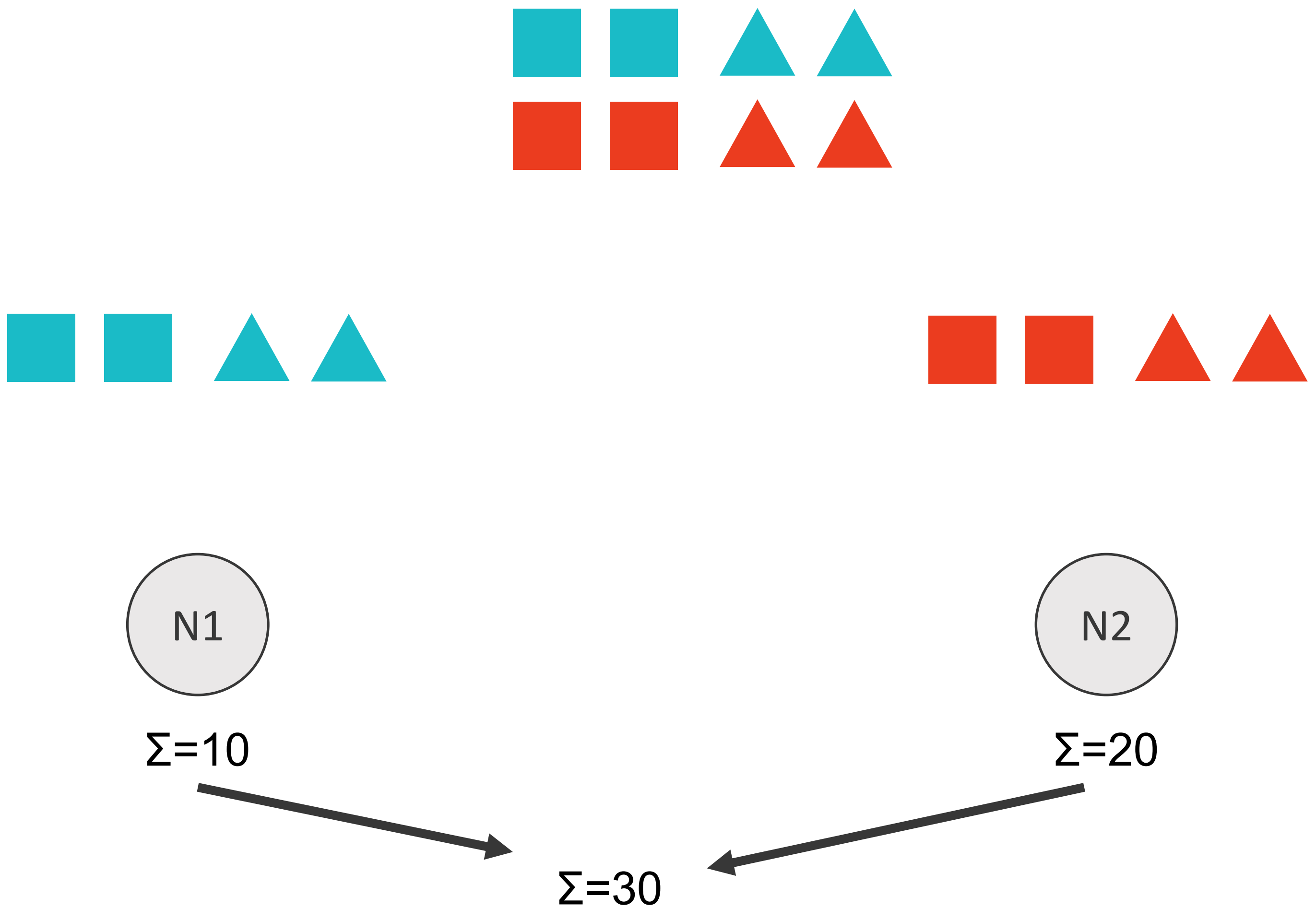
# Why Are Applications Slow? Several Nodes/no Affinity Across Data



# Why Are Applications Slow? Several Nodes/no Affinity Across Data



# Why Are Applications Slow? Several Nodes/no Affinity Across Data



N1

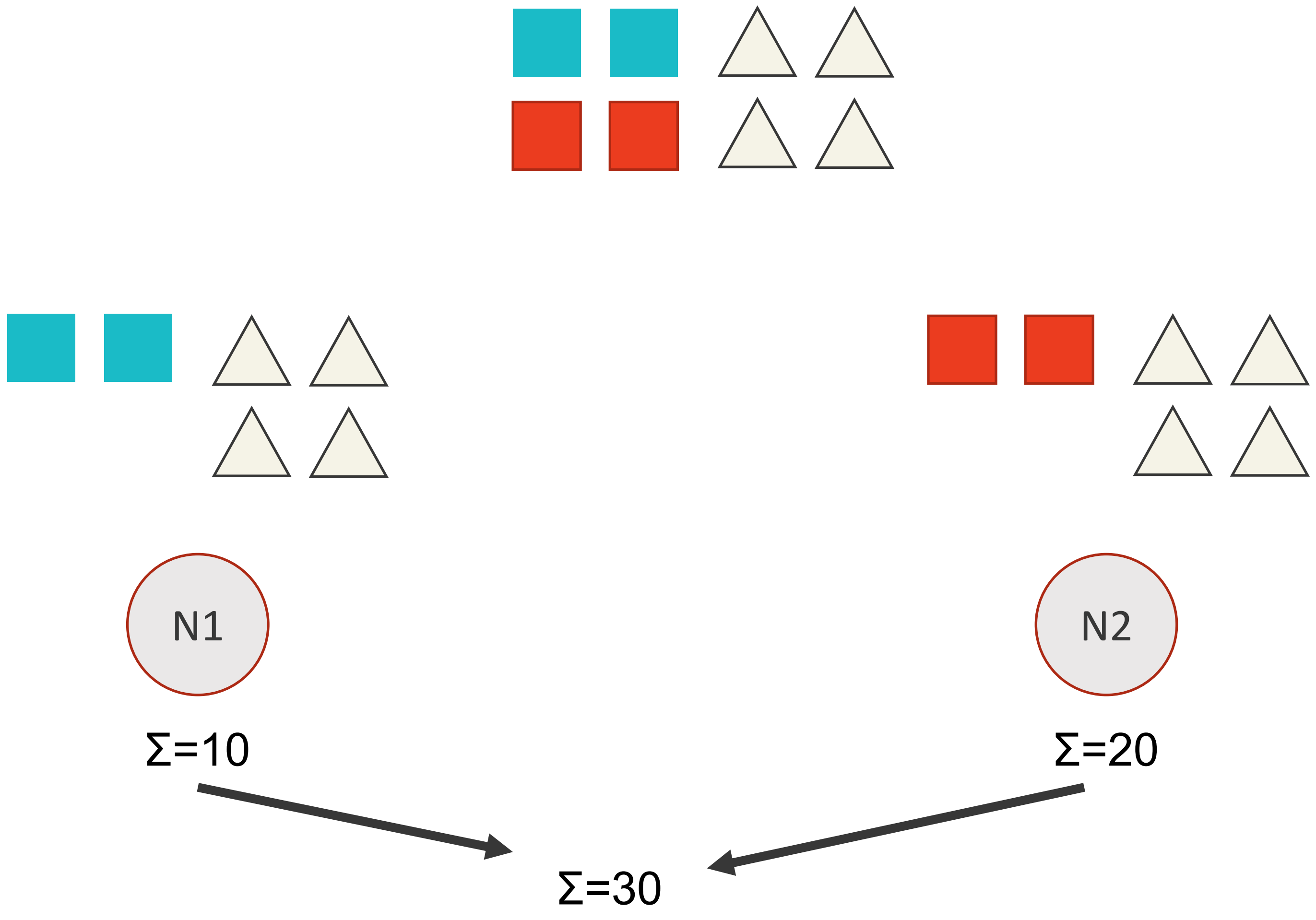
N2

Σ=10

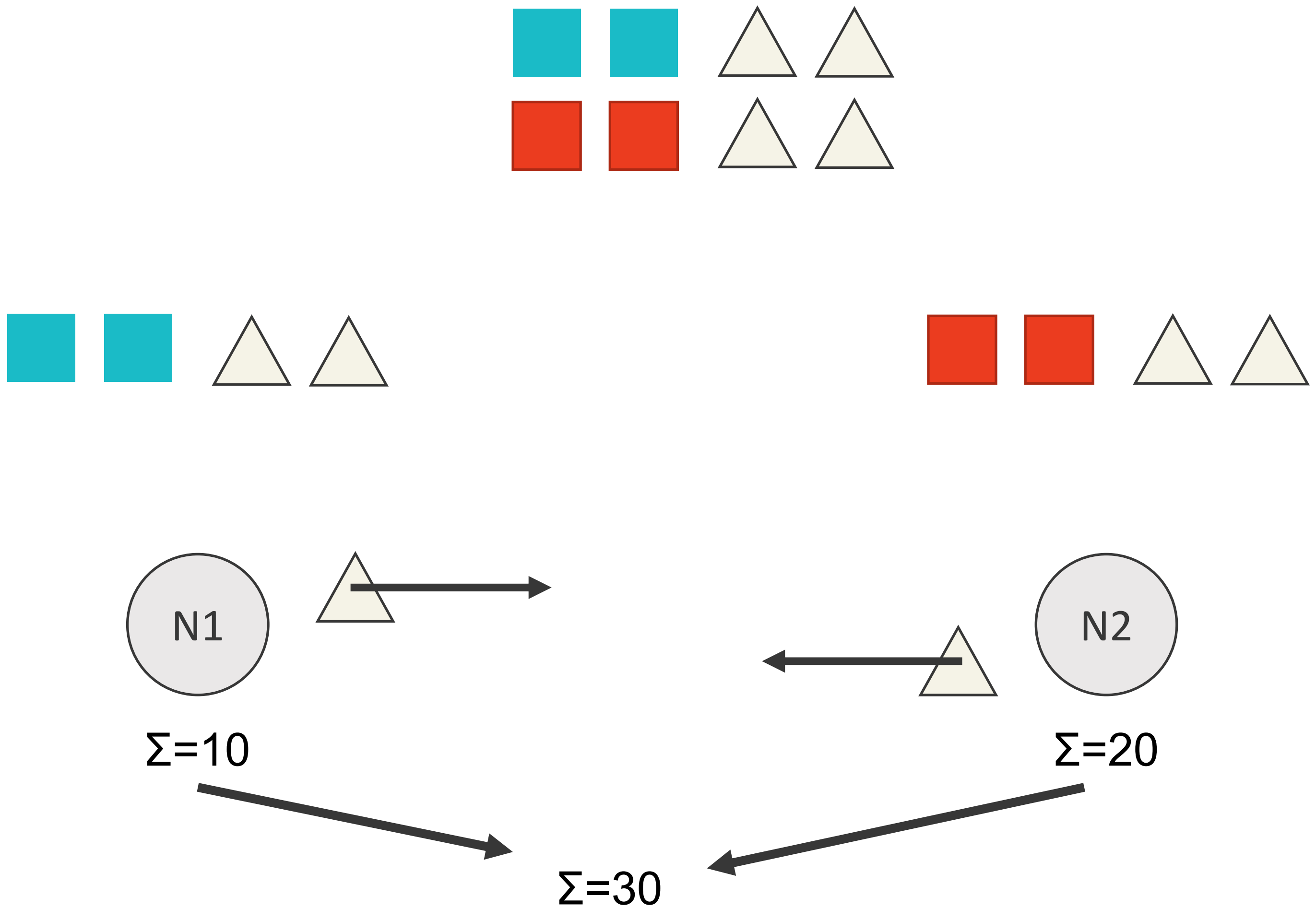
Σ=20

Σ=30

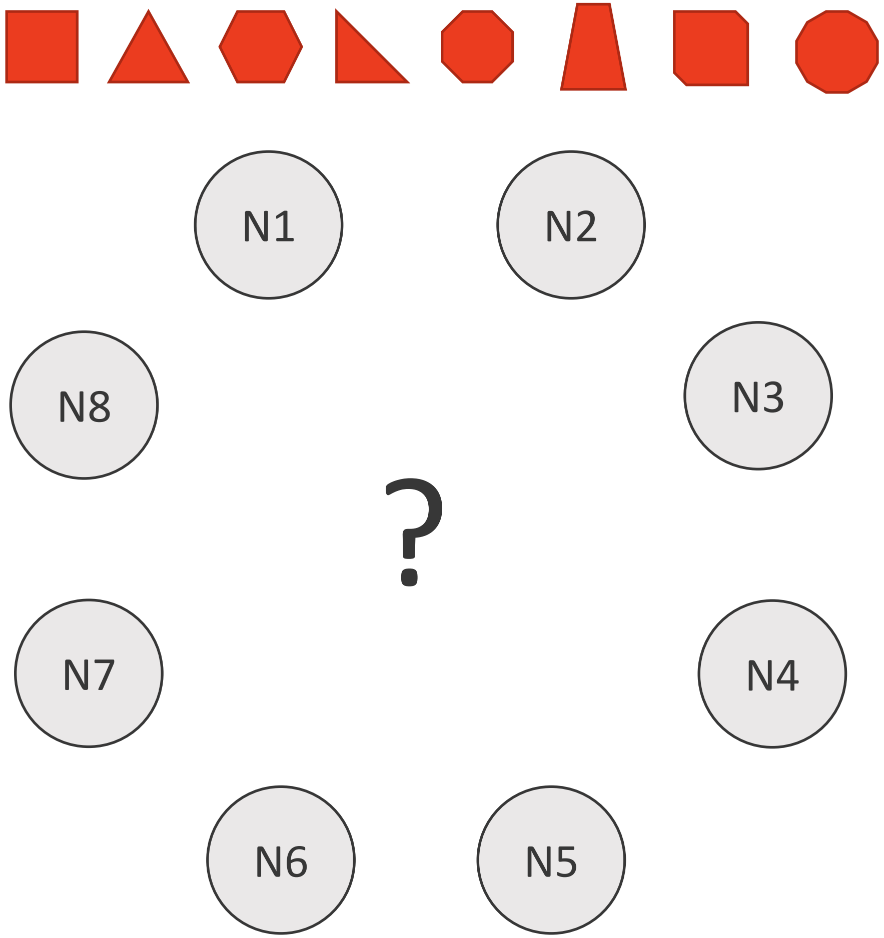
# Why Are Applications Slow?
# Several Nodes/no Affinity Across Data



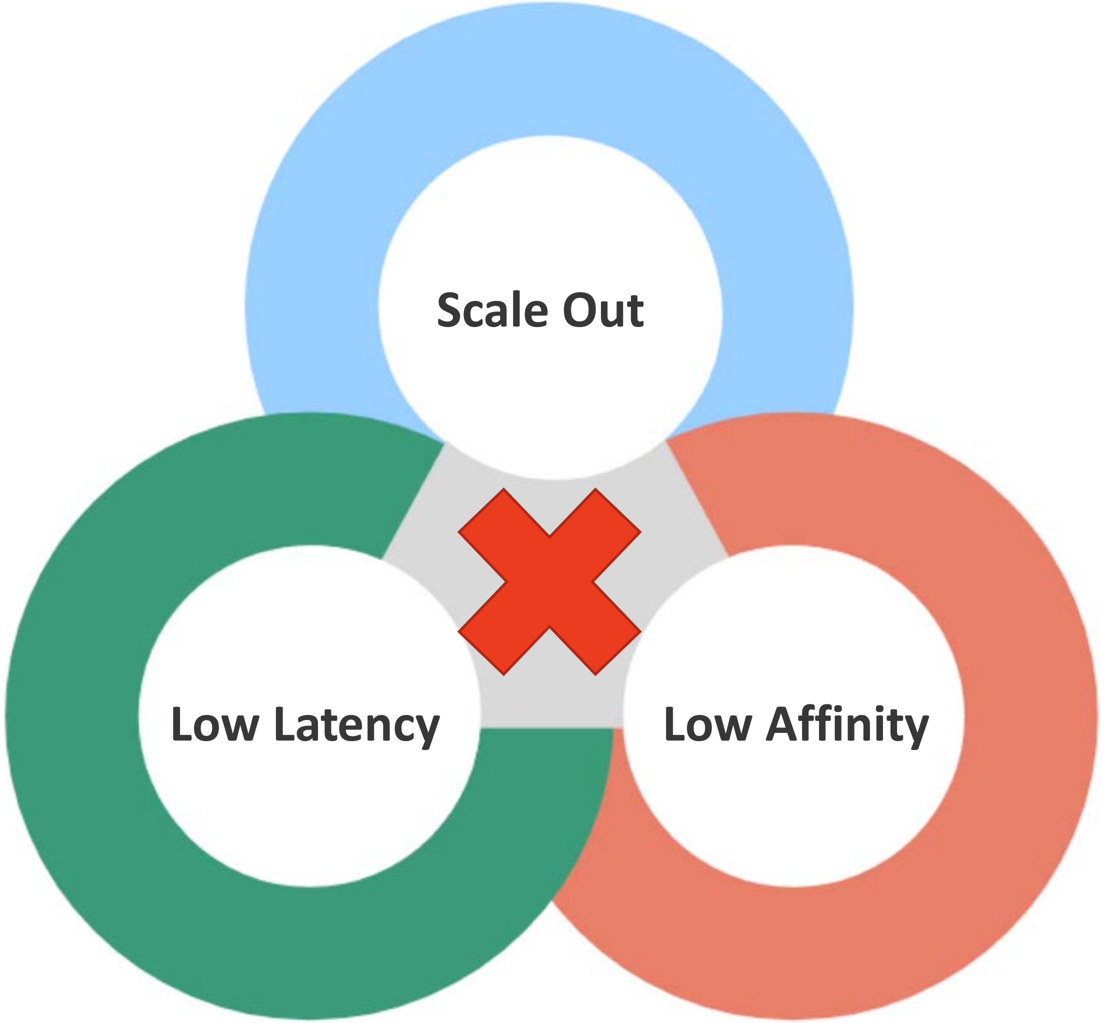
# Why Are Applications Slow? Several Nodes/no Affinity Across Data



# Why Are Applications Slow? Several Nodes/no Affinity Across Data

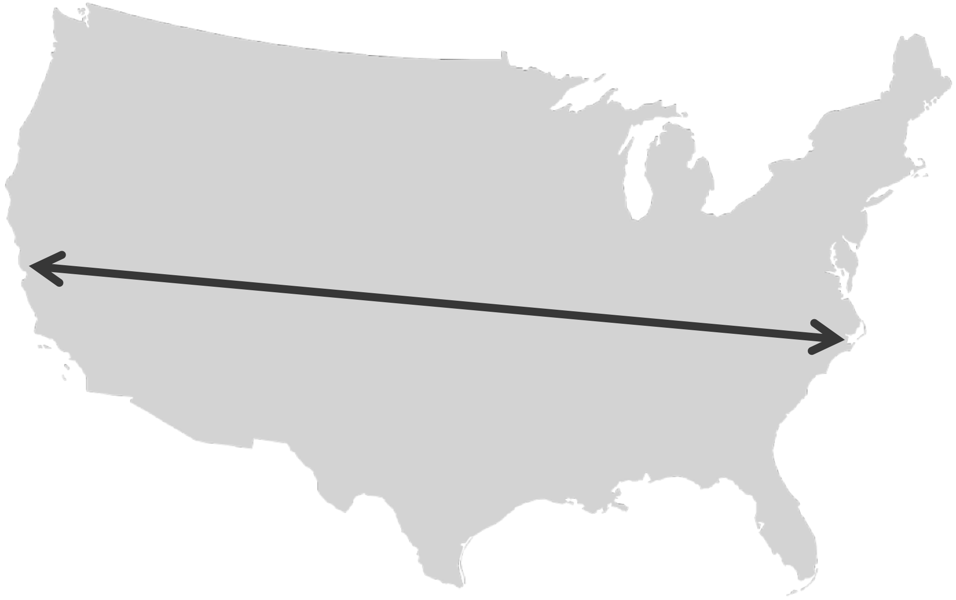


# Why Are Applications Slow? Several Nodes/no Affinity Across Data

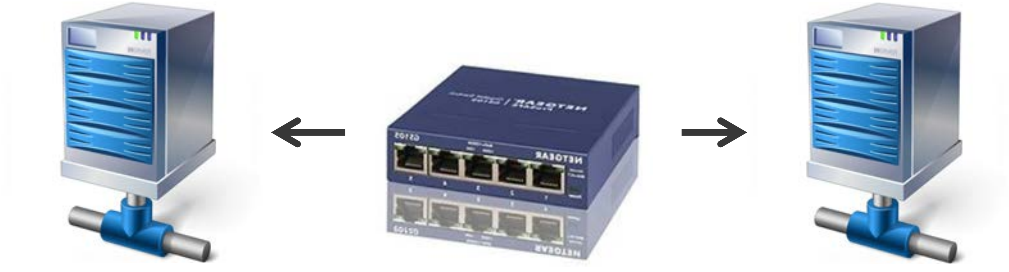


# Why Are Applications Slow?
# Data is Remote: Laws of Nature

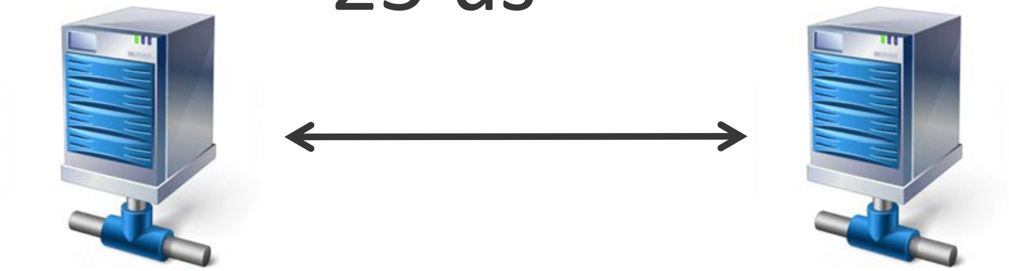
45 ms



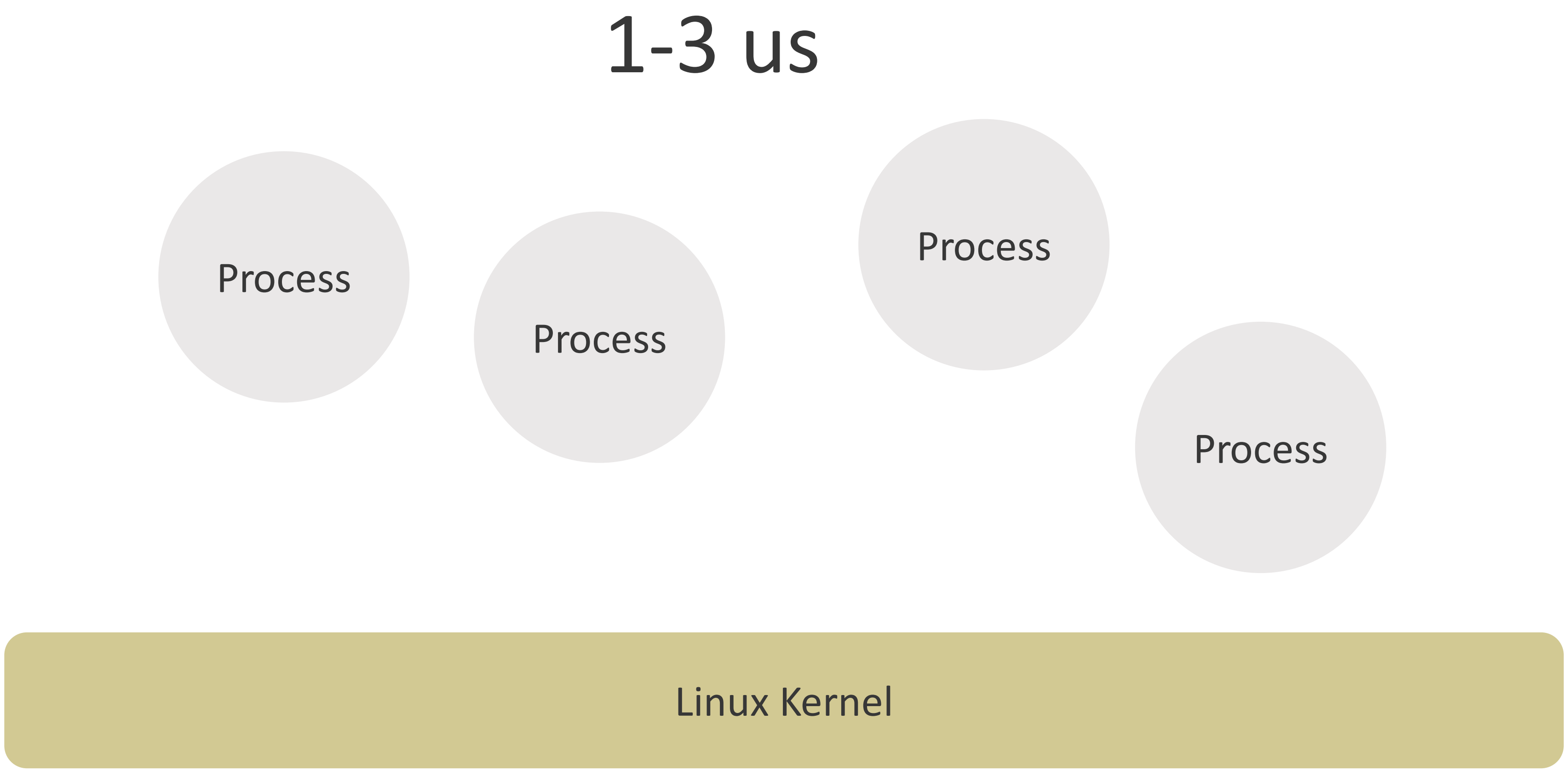
100 us



25 us



# Why Are Applications Slow? Data is Remote: Operating System



1-3 us

Process

Process

Process

Process

Linux Kernel

# Why Are Applications Slow?
# Unnecessary Object Creation

1 s

# Why Are Applications Slow? Unnecessary Object Creation

To write a single Java object to main memory takes 200 ns

```
0000 EA B6 08 E2 02 01 00 00 01 A9 AA FF FF FE 00 01
0010
```

Conclusion: Creating shared objects -> not ultra-low latency

# Why Are Applications Slow?
# Lack of Parallelism

```
$ nproc -all
32

$ top

  PID USER      %CPU  %MEM
 2105 java     100.0   5.4
    1 root       0.5   0.4
```

Speedment

# The Solution: In-JVM-Memory

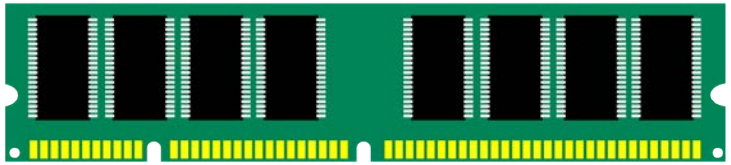
What is That?

# In-Memory vs. In-JVM-Memory

Speedment

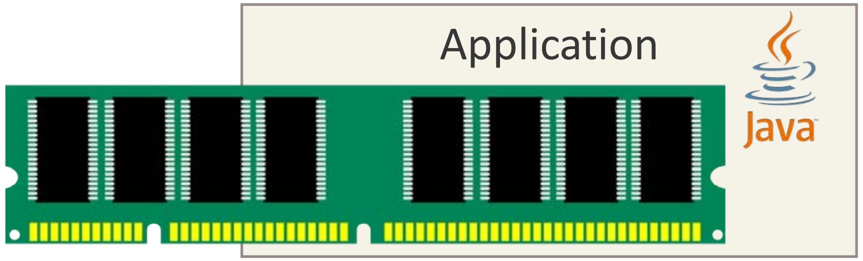
## In-Memory

- Data is in RAM
- The application is remotely connected to a grid, other machine, other process



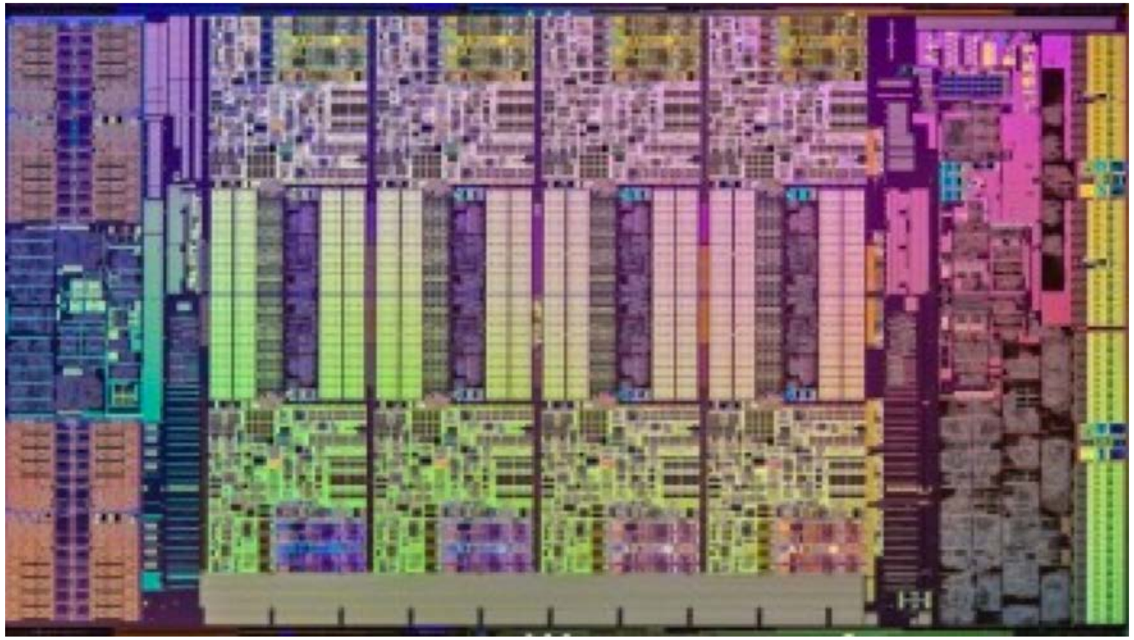
## In-JVM-Memory

- Data is in RAM
- The application and data resides in the same JVM

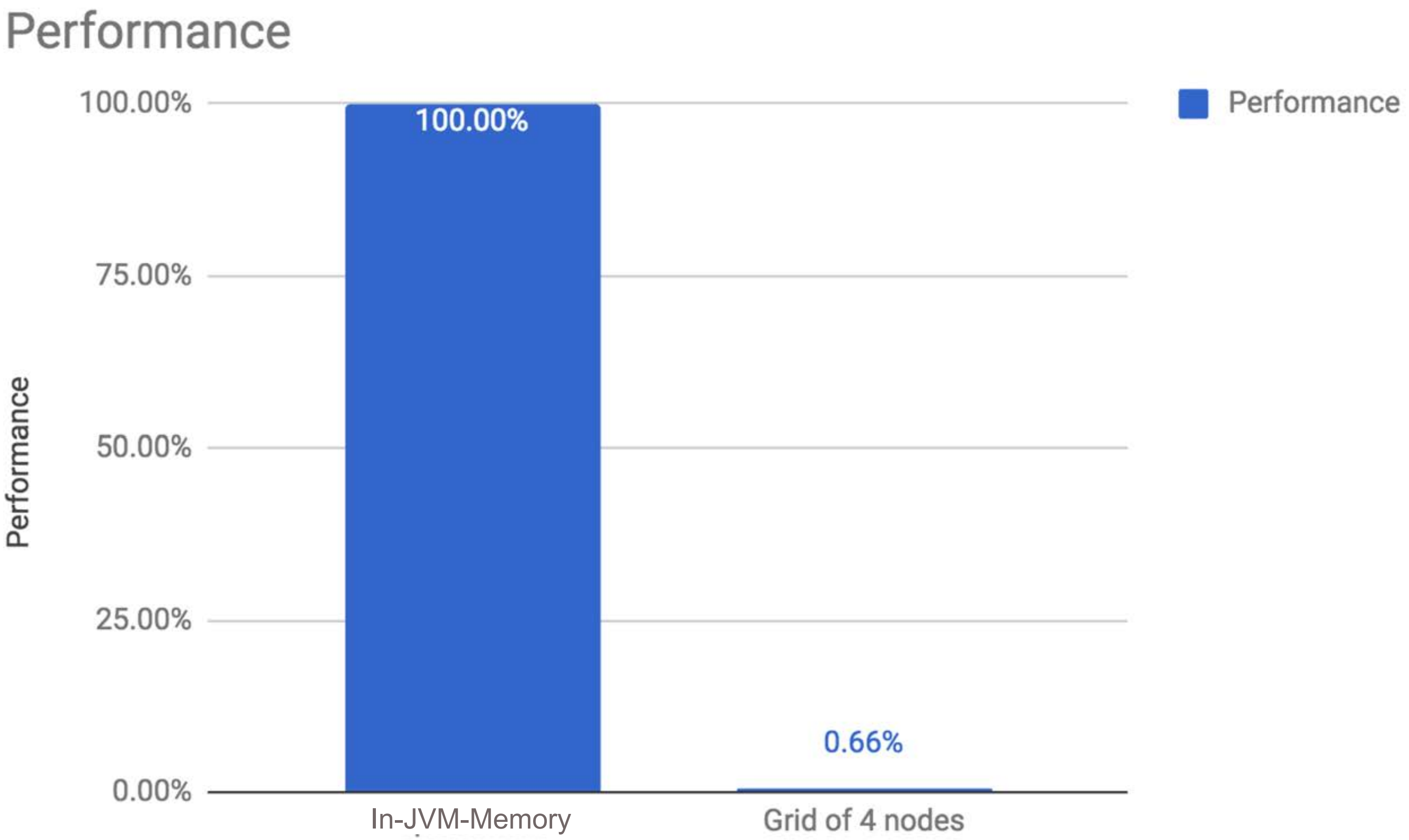


# In-JVM-Memory Makes Ultra Low Latency Possible

- CPU Cache Latencies:
  - L1 ~0.5 ns
  - L2 ~7 ns
  - L3 ~20 ns
- 64-bit Main Memory Read ~100 ns



# In-JVM-Memory vs. In-Memory Performance



Speedment

# In-JVM-Memory Scalability

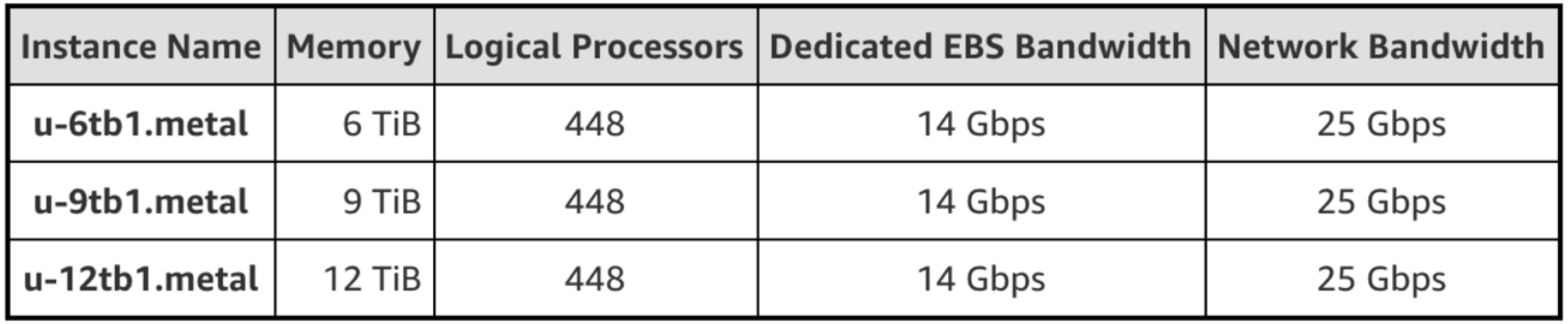
Is That Even Possible?

In-Memory Computing SUMMIT NORTH AMERICA 2018

STARBUCKS
COFFEE

# Scaling up In-JVM-Memory

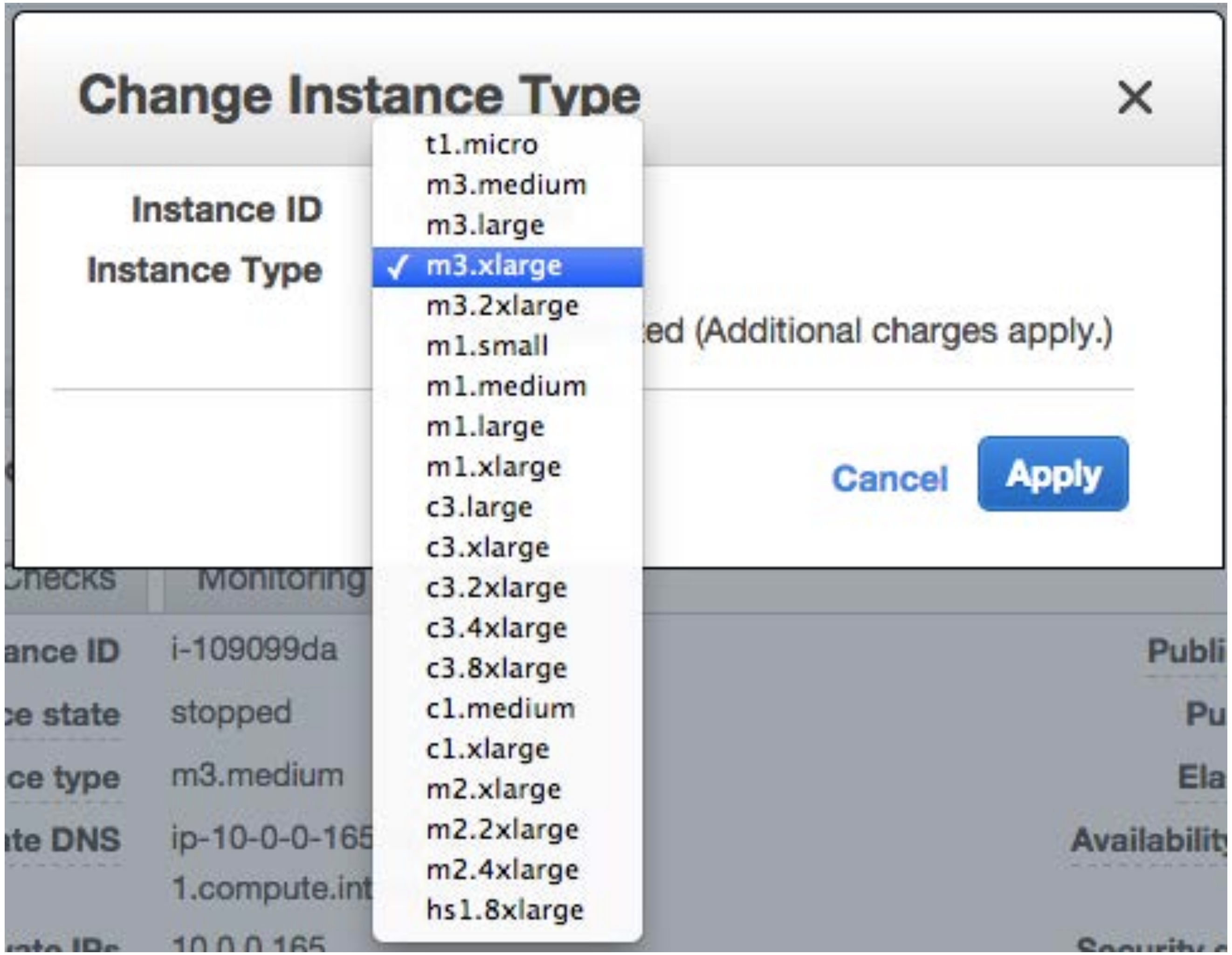
Today: Scale up to 12 TB (Intel® Xeon® Processor E7-8855 v4 * 4)



| Instance Name | Memory | Logical Processors | Dedicated EBS Bandwidth | Network Bandwidth |
|---|---|---|---|---|
| u-6tb1.metal | 6 TiB | 448 | 14 Gbps | 25 Gbps |
| u-9tb1.metal | 9 TiB | 448 | 14 Gbps | 25 Gbps |
| u-12tb1.metal | 12 TiB | 448 | 14 Gbps | 25 Gbps |

Soon: Scale up to 48 TB

# Increase Memory in the Cloud as You Grow



Change Instance Type

Instance ID

Instance Type ✓ m3.xlarge

ed (Additional charges apply.)

ance ID i-109099da

ce state stopped

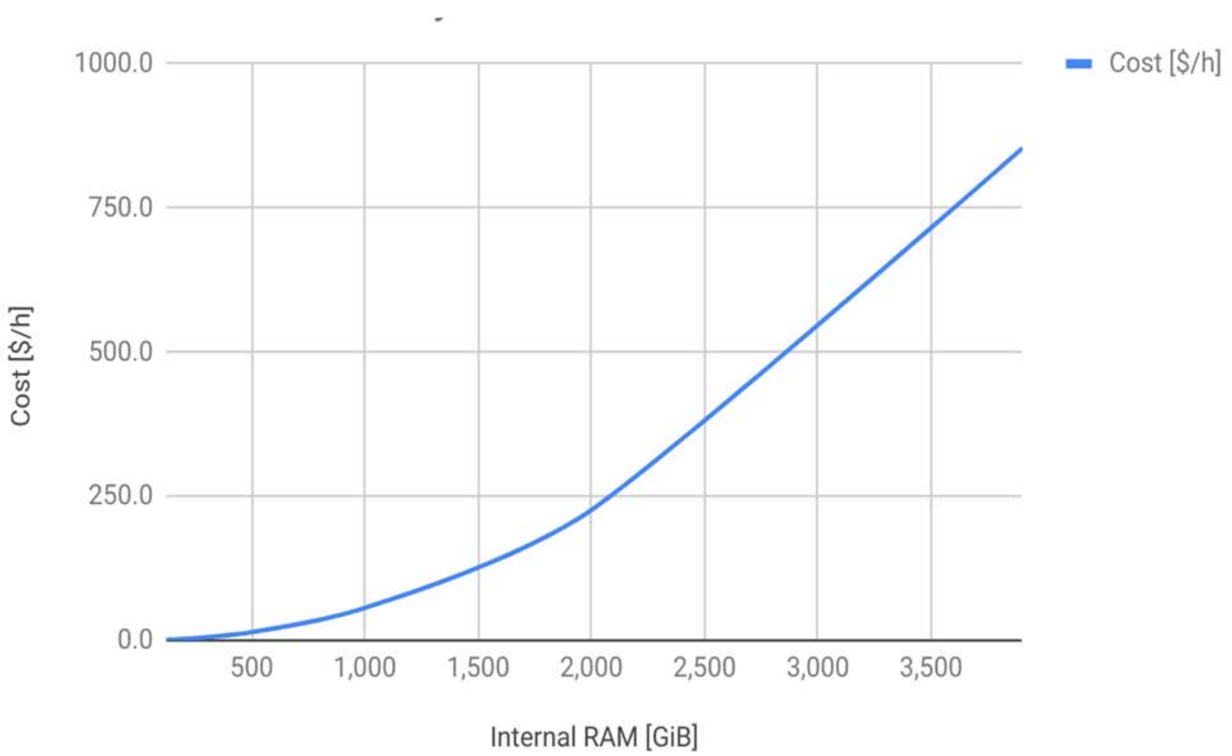
ce type m3.medium

ate DNS ip-10-0-0-165 1.compute.int

ate IPs 10.0.0.165

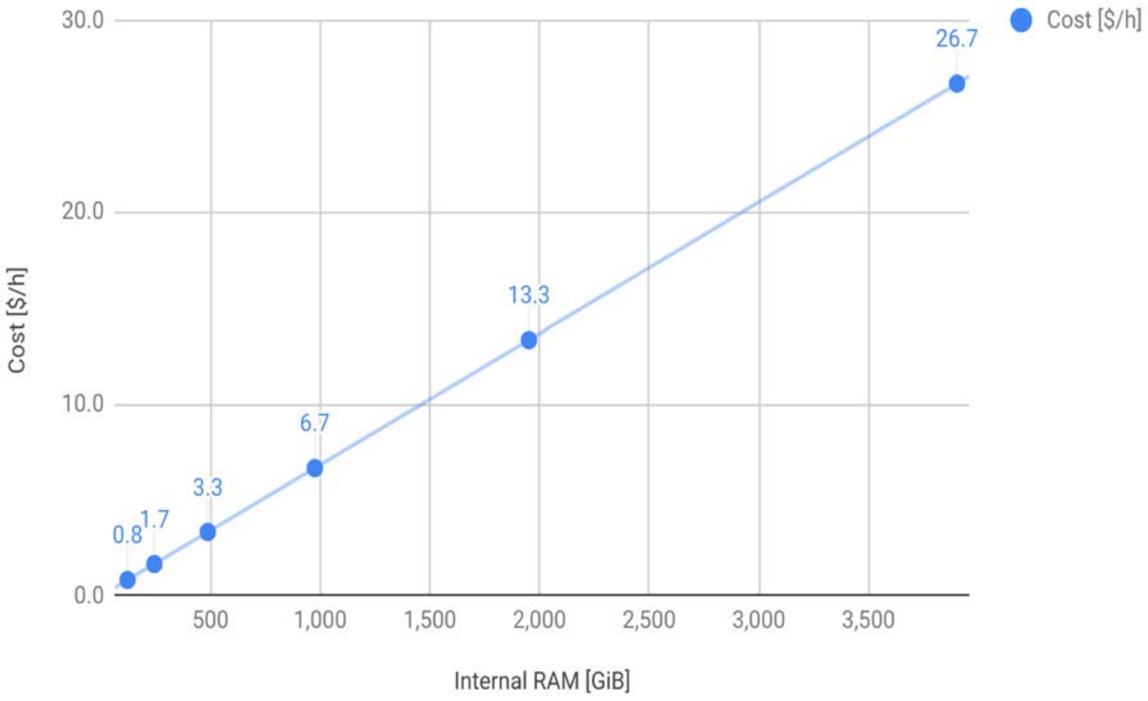
# Is Scale Up Cost Effective?

## General Belief



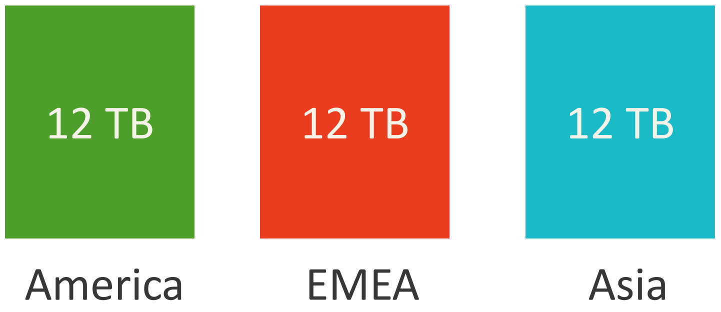
## Fact

AWS "x1e.Nxlarge" Series: Memory Size vs. Cost

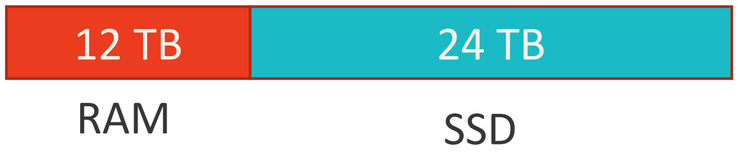


# What if I Have More Than 12 TB?

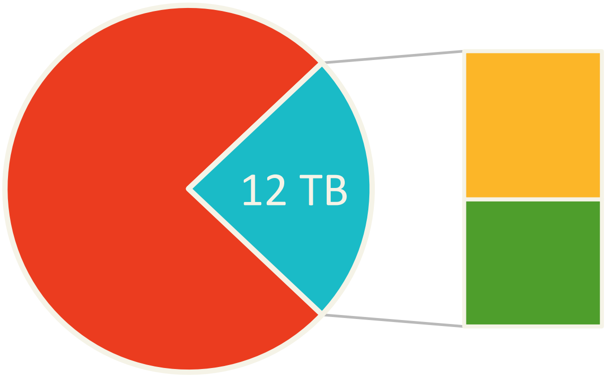
- High Level Sharding
  - Per year, region, segment



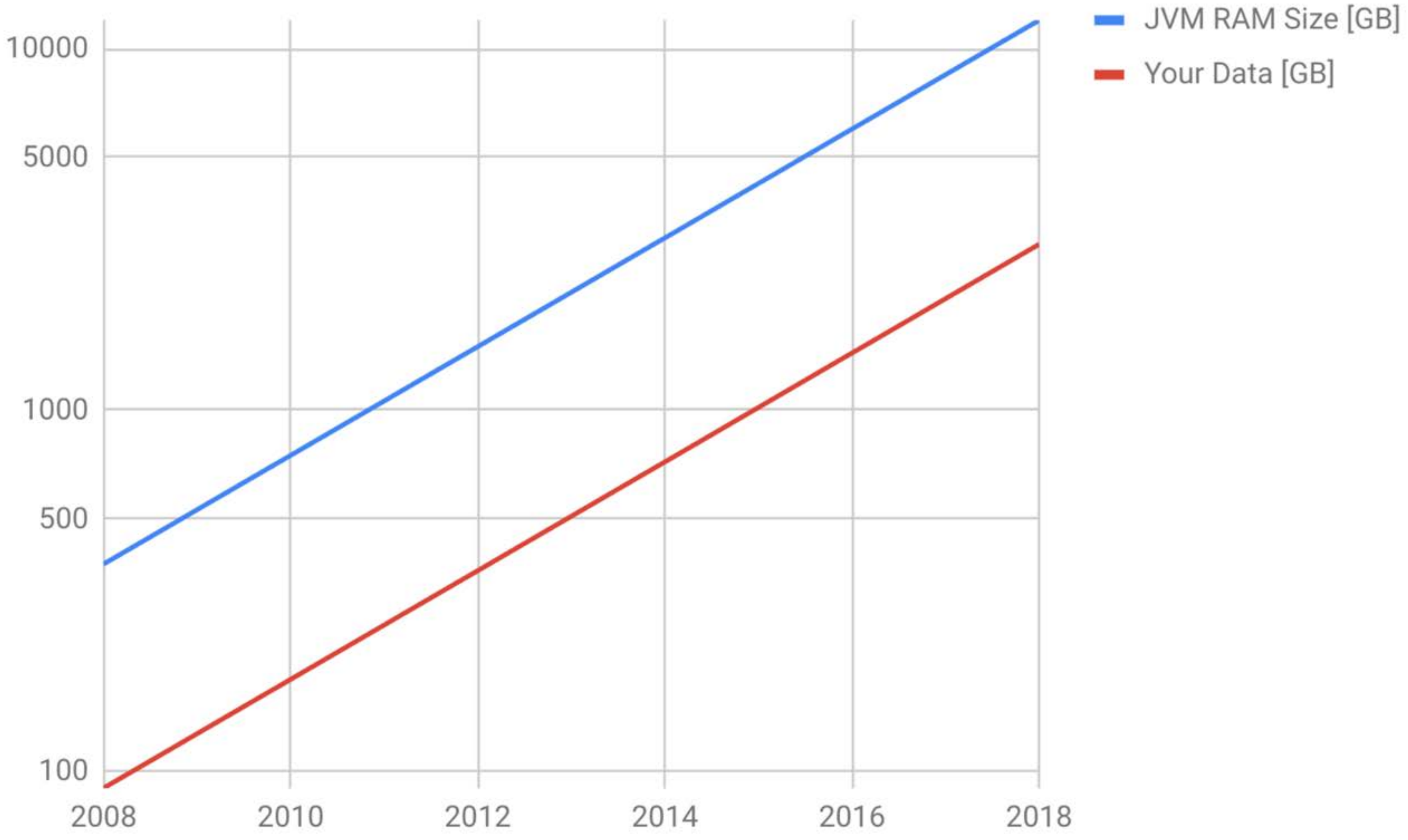
- Memory Mapping (e.g. IMDT)



- Use in-JVM-memory solution as an add on for your current solution for part of your data



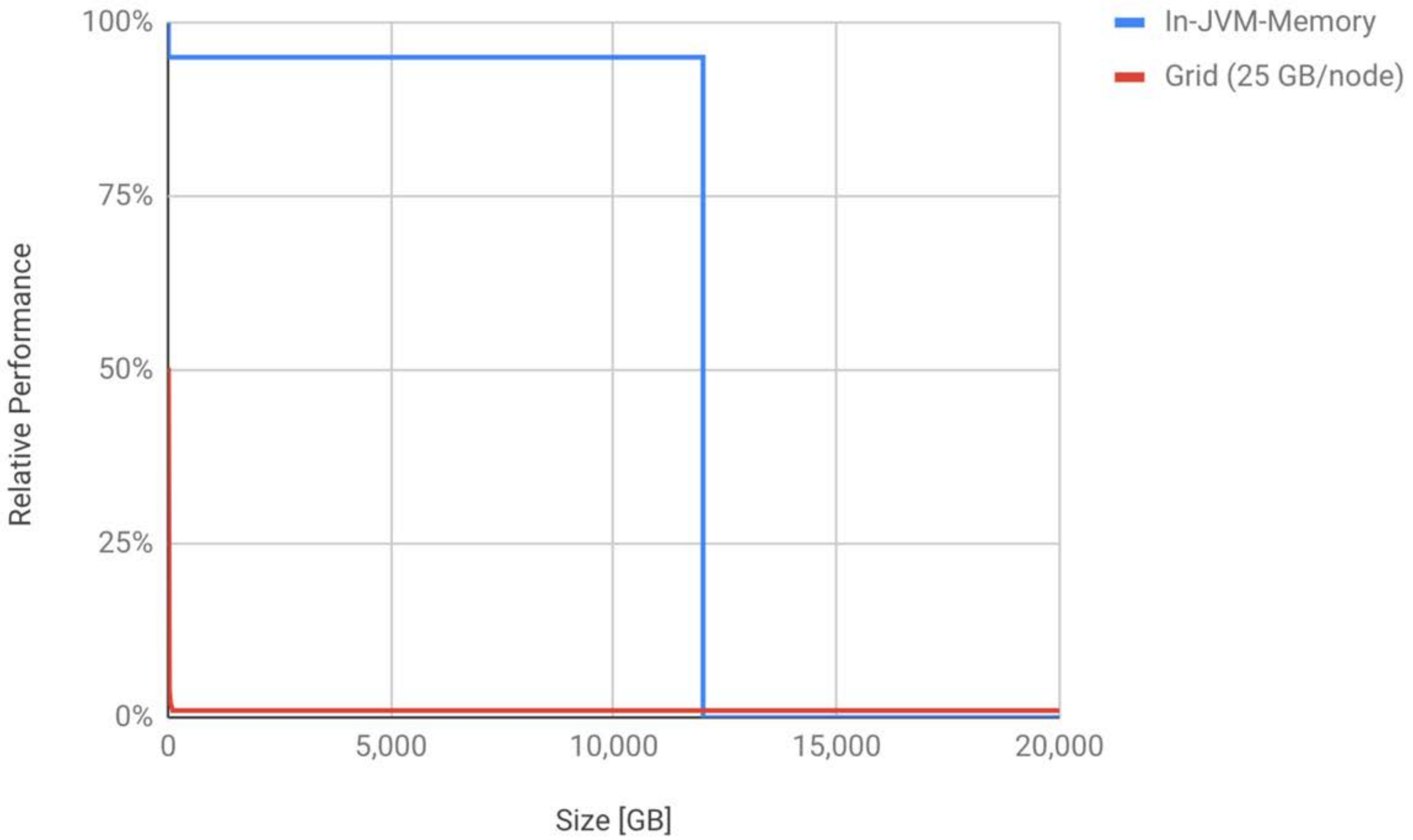
# What if My Data Grows?



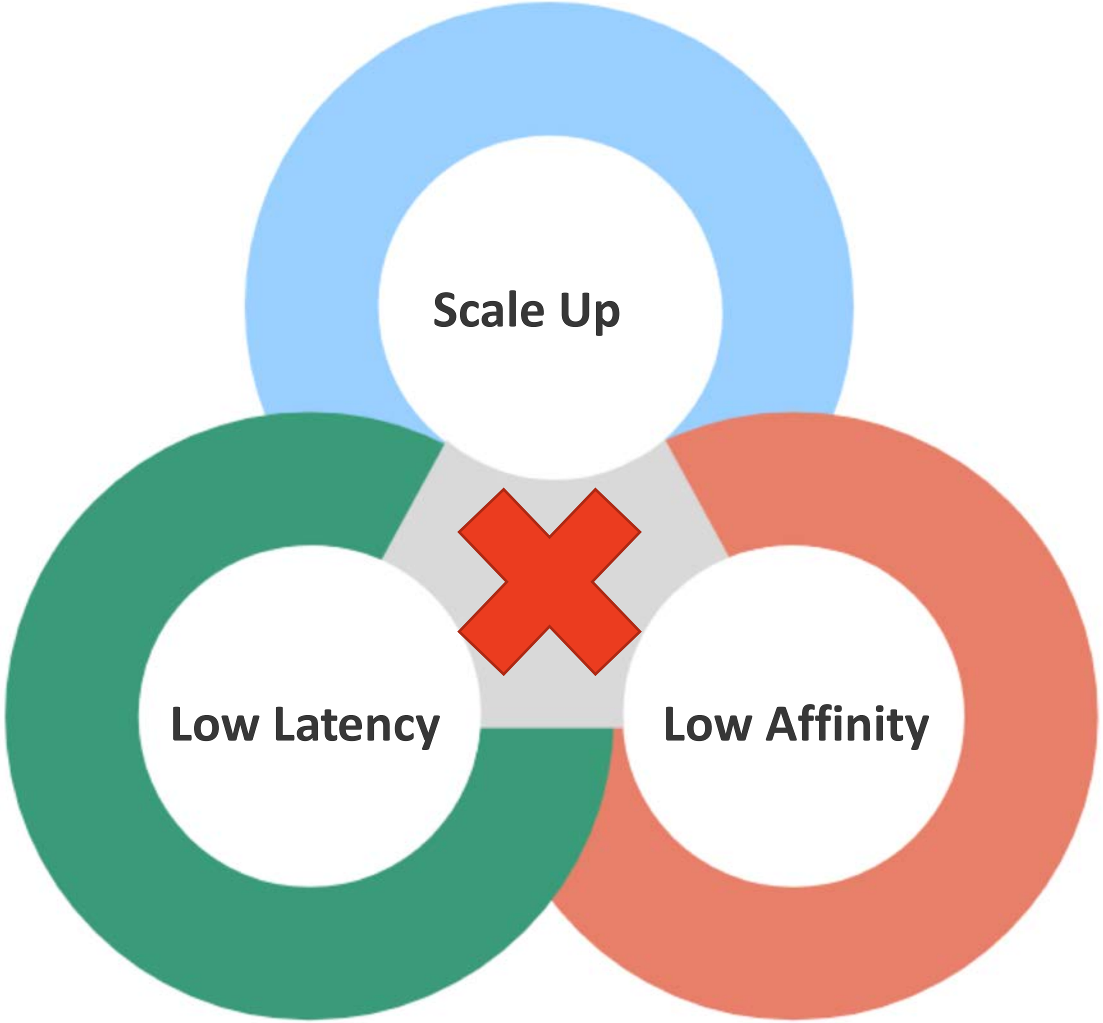
# In-JVM-Memory vs. In-Memory Performance

Speedment

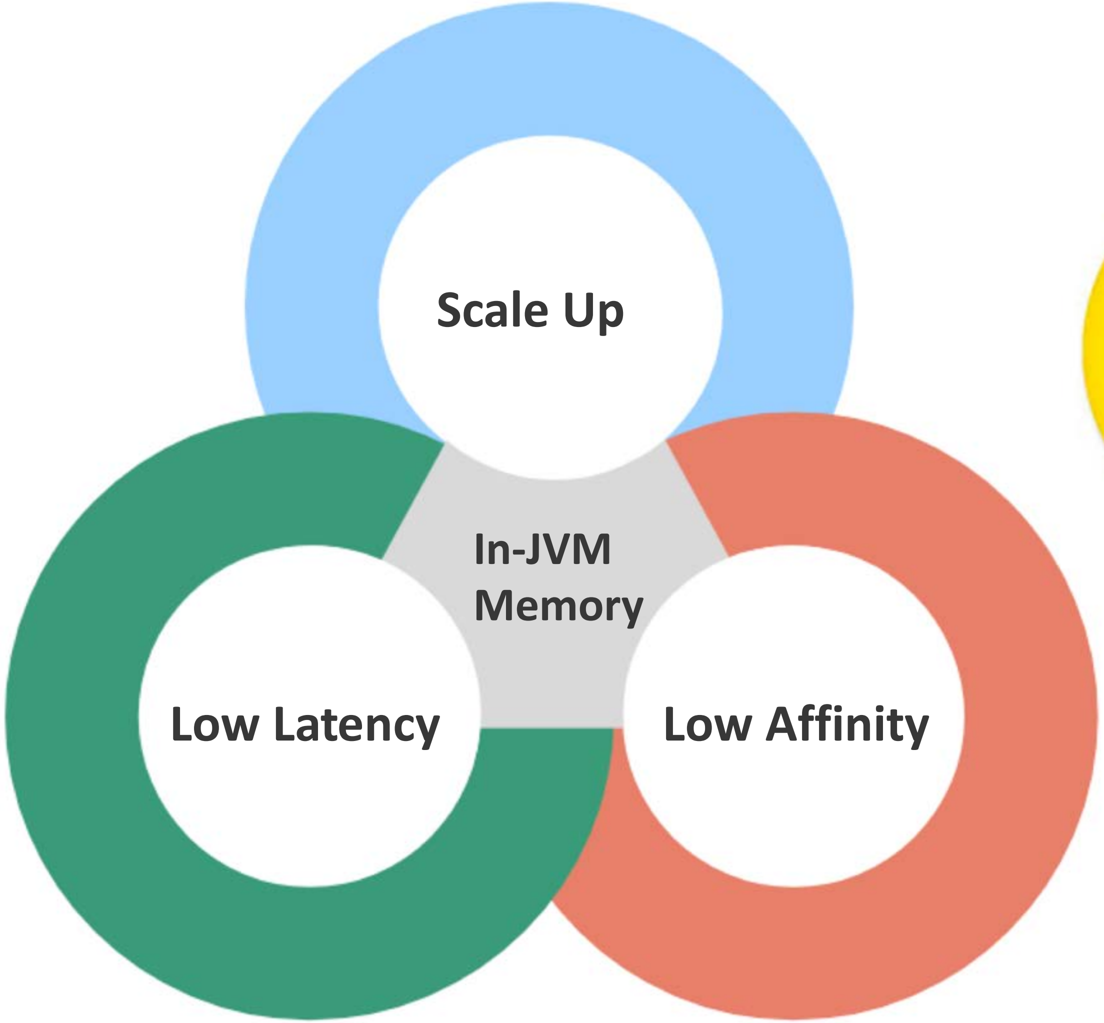
## Data with 75% correlation



# Recap: Impossible to Scale Out AND Get Low Latency When You Have Low Affinity



# Solution: In-JVM Memory



Speedment

# In-JVM-Memory Solution: Speedment

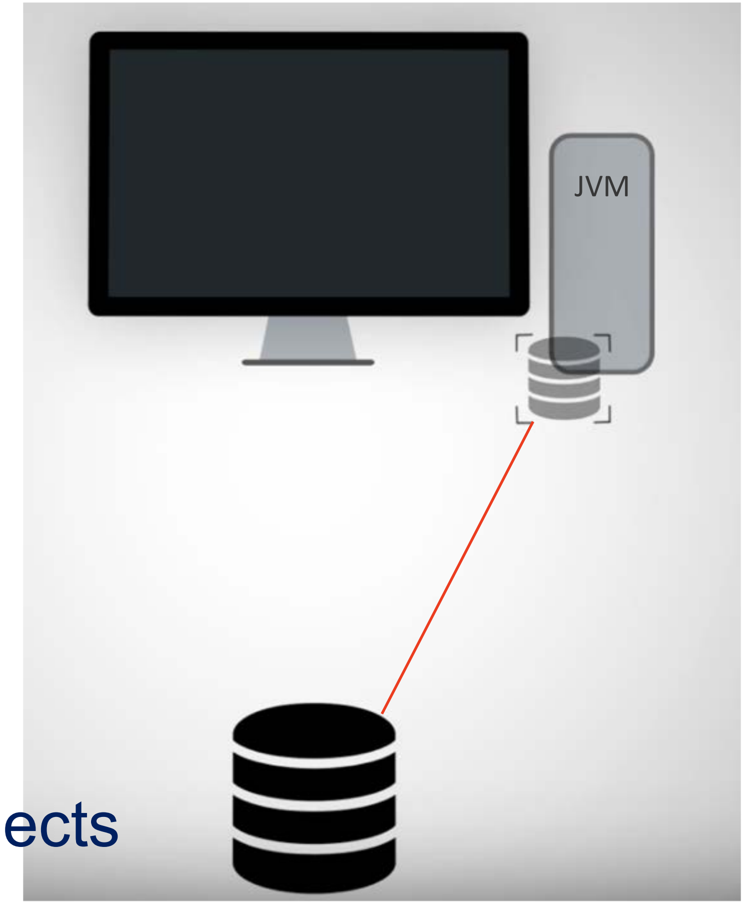
In-Memory Computing SUMMIT NORTH AMERICA 2018

# Speedment: In-JVM-Memory DataStore

- Continuously creates data snapshots from a data source
- Places the copy within the JVM

- Off-Heap Data
- Off-Heap Indexing
- No Impact on Garbage Collect

- Supports off-heap joins and aggregations
- Can operate without creating intermediary objects

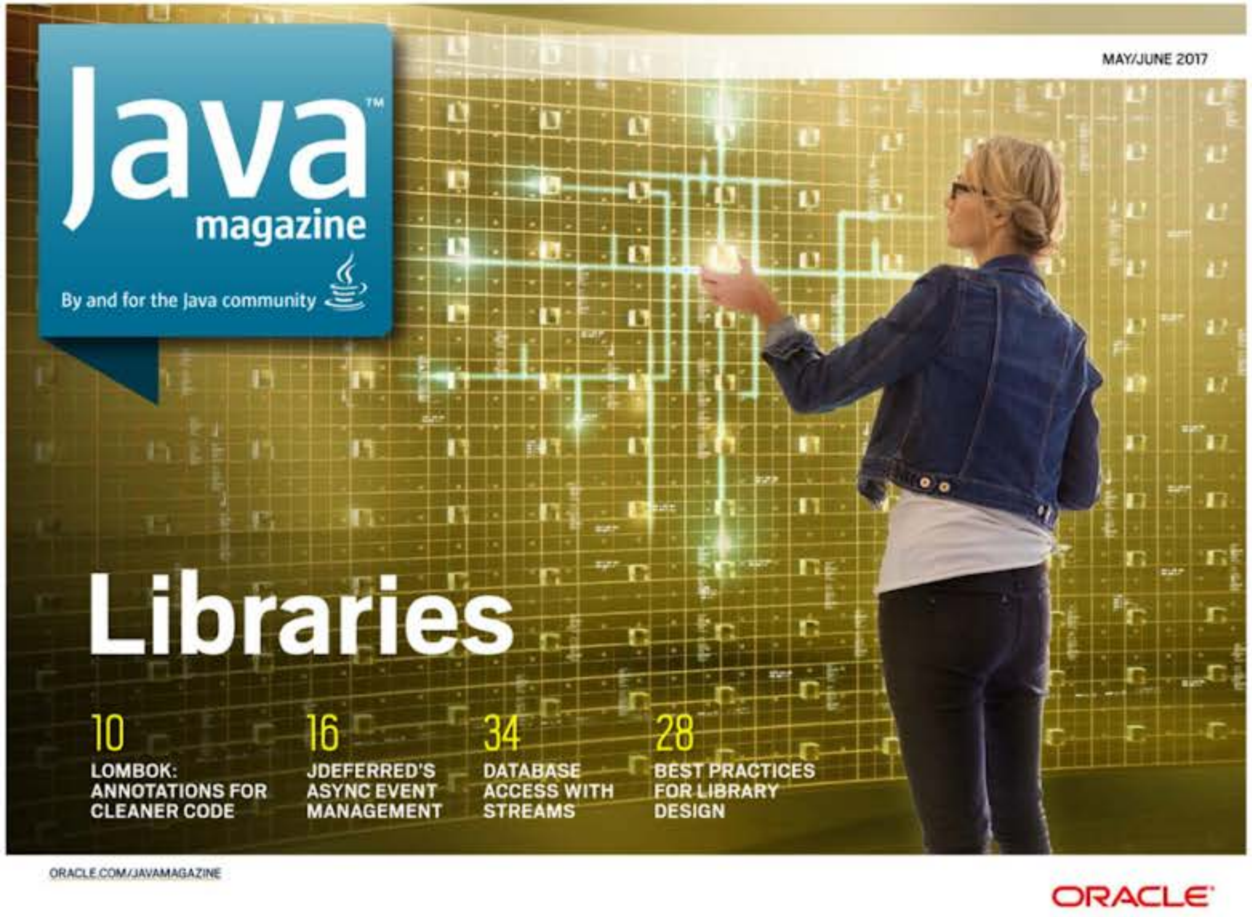


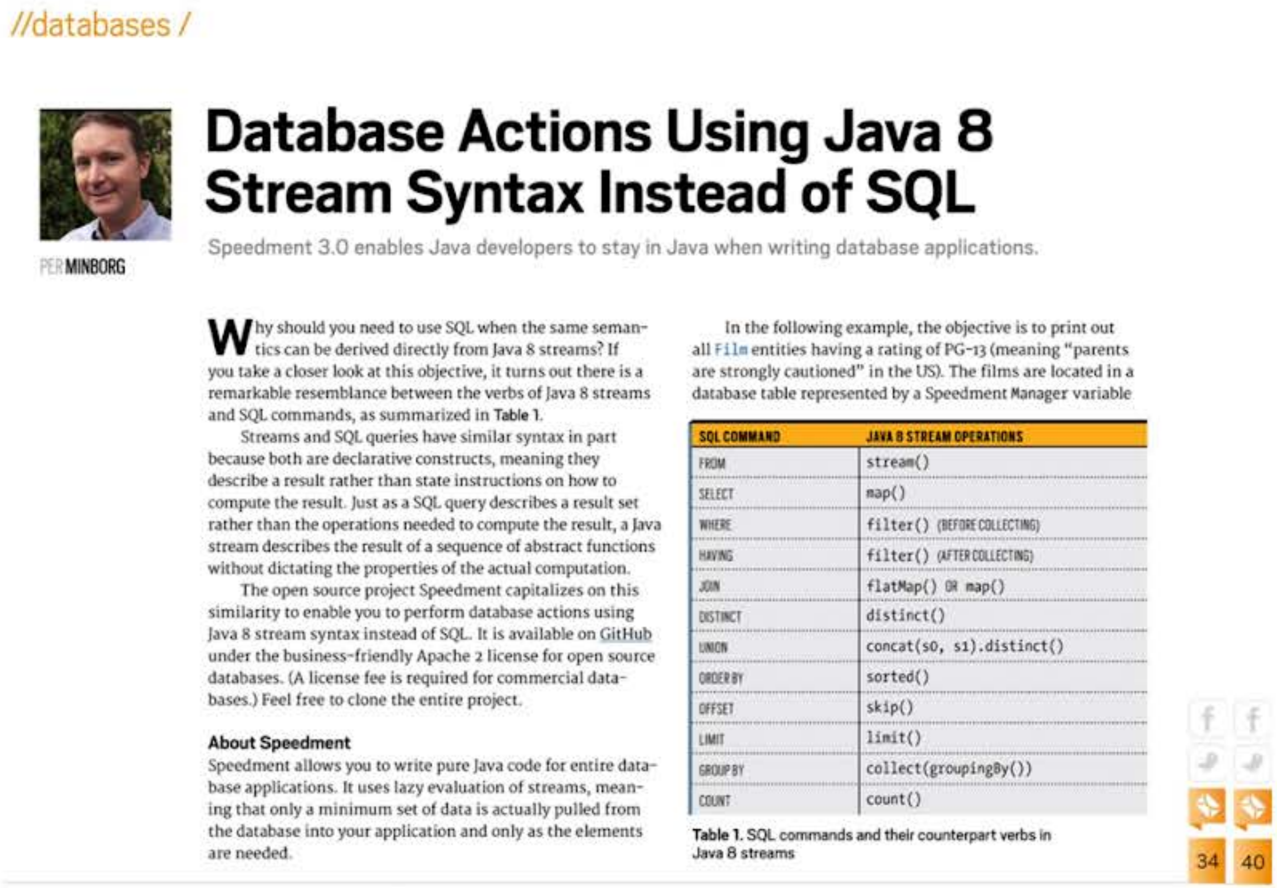
# Speedment API: Java Stream ORM

```
java.util.stream.Stream
```

# Speedment API: Java Stream ORM

Speedment


MAY/JUNE 2017
Java™
magazine
By and for the Java community
Libraries
10 LOMBOK: ANNOTATIONS FOR CLEANER CODE
16 JDEFERRED'S ASYNC EVENT MANAGEMENT
34 DATABASE ACCESS WITH STREAMS
28 BEST PRACTICES FOR LIBRARY DESIGN
ORACLE.COM/JAVAMAGAZINE
ORACLE



//databases /

## Database Actions Using Java 8 Stream Syntax Instead of SQL

PER MINBORG

Speedment 3.0 enables Java developers to stay in Java when writing database applications.

Why should you need to use SQL when the same semantics can be derived directly from Java 8 streams? If you take a closer look at this objective, it turns out there is a remarkable resemblance between the verbs of Java 8 streams and SQL commands, as summarized in Table 1.

Streams and SQL queries have similar syntax in part because both are declarative constructs, meaning they describe a result rather than state instructions on how to compute the result. Just as a SQL query describes a result set rather than the operations needed to compute the result, a Java stream describes the result of a sequence of abstract functions without dictating the properties of the actual computation.

The open source project Speedment capitalizes on this similarity to enable you to perform database actions using Java 8 stream syntax instead of SQL. It is available on GitHub under the business-friendly Apache 2 license for open source databases. (A license fee is required for commercial databases.) Feel free to clone the entire project.

### About Speedment

Speedment allows you to write pure Java code for entire database applications. It uses lazy evaluation of streams, meaning that only a minimum set of data is actually pulled from the database into your application and only as the elements are needed.

In the following example, the objective is to print out all `Film` entities having a rating of PG-13 (meaning "parents are strongly cautioned" in the US). The films are located in a database table represented by a Speedment `Manager` variable

| SQL COMMAND | JAVA 8 STREAM OPERATIONS |
|---|---|
| FROM | `stream()` |
| SELECT | `map()` |
| WHERE | `filter()` (BEFORE COLLECTING) |
| HAVING | `filter()` (AFTER COLLECTING) |
| JOIN | `flatMap()` OR `map()` |
| DISTINCT | `distinct()` |
| UNION | `concat(s0, s1).distinct()` |
| ORDER BY | `sorted()` |
| OFFSET | `skip()` |
| LIMIT | `limit()` |
| GROUP BY | `collect(groupingBy())` |
| COUNT | `count()` |

Table 1. SQL commands and their counterpart verbs in Java 8 streams

# Speedment API: Java Stream ORM

## Declarative Constructs in SQL and Stream

```
SELECT * FROM FILM
  WHERE RATING = ‘PG-13’
```

```
films.stream()
  .filter(Film.RATING.equal(Rating.PG13))
```

# Speedment Can Process Data without Creating Intermediate Objects

```
films.stream()
    .filter(Film.RATING.equal(Rating.PG13))
    .count();
```

# Speedment Can Process Data without Creating Intermediate Objects

```
films.stream()
    .filter(Film.RATING.equal(Rating.PG13))
    .collect(toJsonLengthAndTitle()));
```

# Speedment: Off-Heap Joins/Aggregations

```
var join = joinComponent
   .from(FilmManager.IDENTIFIER)
   .innerJoinOn(Language.LANGUAGE_ID).equal(Film.LANGUAGE_ID)
   .build(Tuples::of);
```

# Speedment: Off-Heap Joins/Aggregations

```
var offHeapAggregator = Aggregator.builder(Result::new)
  .on(Language.LANGUAGE_ID).key(Result::setLanguage)
  .on(Film.RATING).key(Result:setRating)
  .on(Film.LENGTH).average(Result::setAverage)
  .build();
```

# Speedment:Off-Heap Joins/Aggregations

```
var result = join.stream()
  .collect(offHeapAggregator);
```

# Speedment: Parallel Processing

```
join.stream()
   .parallel()
   .collect(offHeapAggregator);
```

# Speedment: Parallel Processing

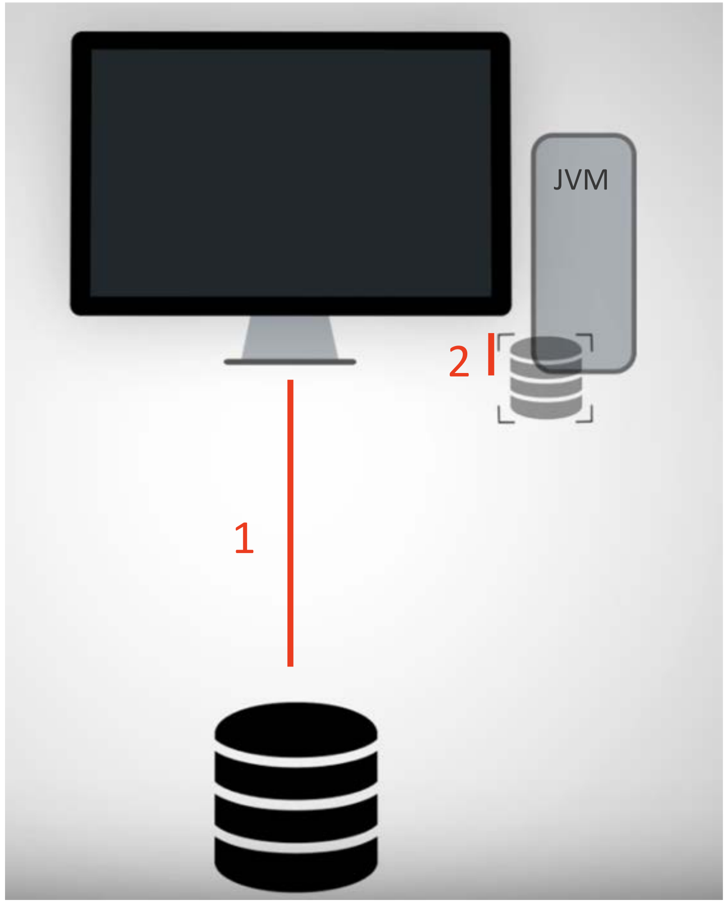
```
$ nproc -all
32

$ top

  PID USER      %CPU  %MEM
 2107 java    3170.0   5.4
    1 root       0.5   0.4
```

# Hands on Demo

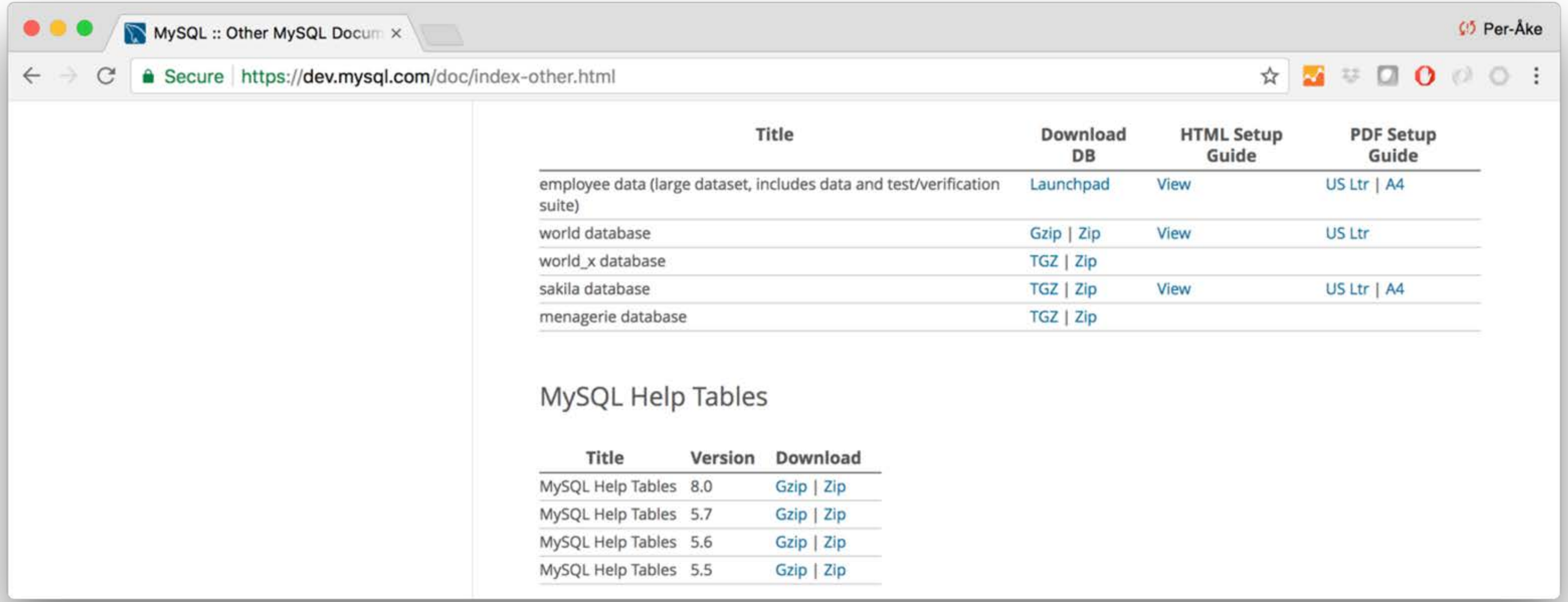
Seeing is Believing

# Demo

```
@Benchmark
public long filterAndCount() {
    return films.stream()
        .filter(RATING_EQUALS_PG_13)
        .count();
}
```



# Demo: Download Sakila Demo Database



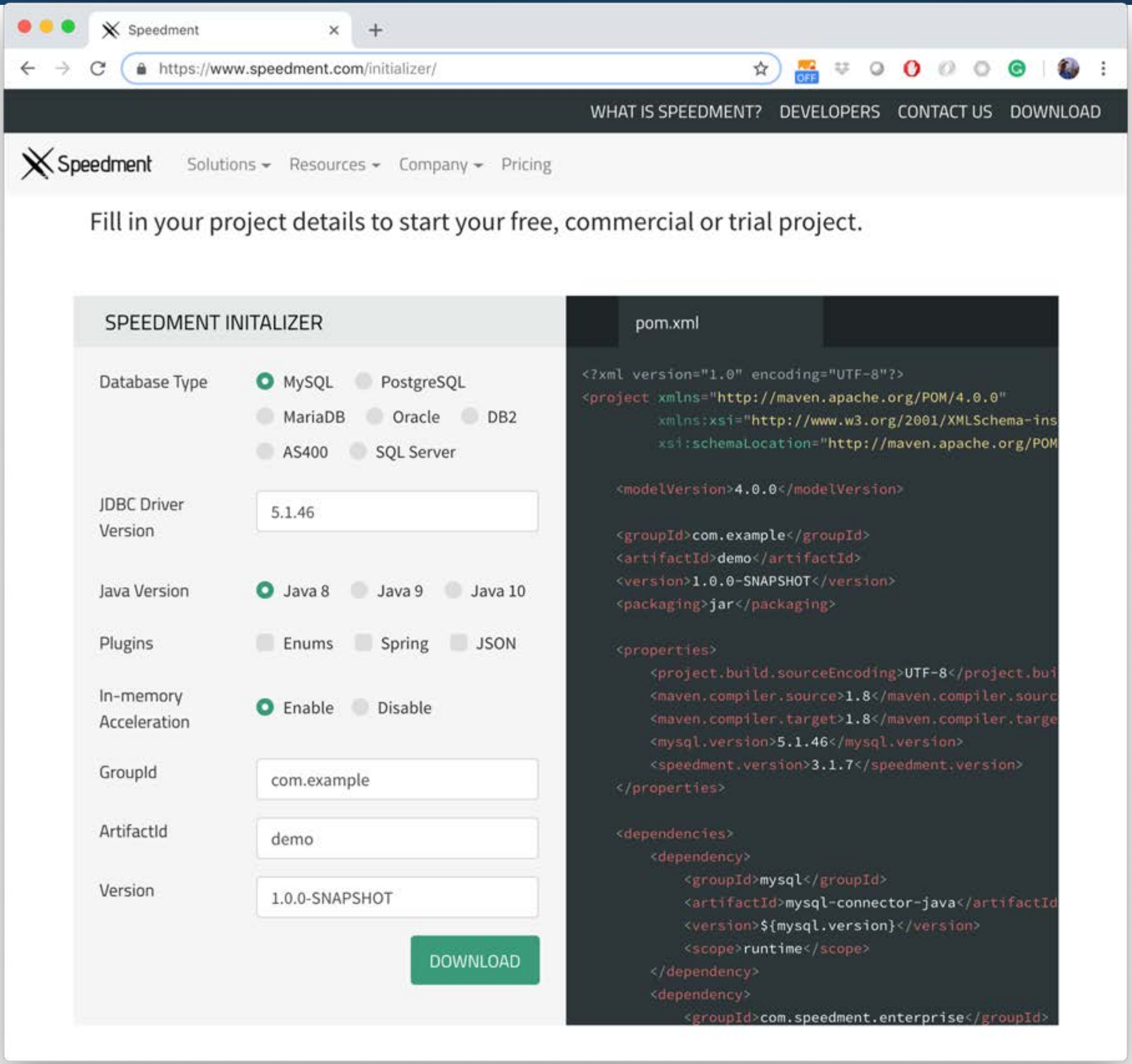
MySQL :: Other MySQL Docum
Per-Åke
Secure
https://dev.mysql.com/doc/index-other.html

| Title | Download DB | HTML Setup Guide | PDF Setup Guide |
|---|---|---|---|
| employee data (large dataset, includes data and test/verification suite) | Launchpad | View | US Ltr \| A4 |
| world database | Gzip \| Zip | View | US Ltr |
| world_x database | TGZ \| Zip | | |
| sakila database | TGZ \| Zip | View | US Ltr \| A4 |
| menagerie database | TGZ \| Zip | | |

## MySQL Help Tables

| Title | Version | Download |
|---|---|---|
| MySQL Help Tables | 8.0 | Gzip \| Zip |
| MySQL Help Tables | 5.7 | Gzip \| Zip |
| MySQL Help Tables | 5.6 | Gzip \| Zip |
| MySQL Help Tables | 5.5 | Gzip \| Zip |

# Demo: Initialize the Project



Fill in your project details to start your free, commercial or trial project.

SPEEDMENT INITALIZER

Database Type ● MySQL ○ PostgreSQL ○ MariaDB ○ Oracle ○ DB2 ○ AS400 ○ SQL Server

JDBC Driver Version: 5.1.46

Java Version ● Java 8 ○ Java 9 ○ Java 10

Plugins □ Enums □ Spring □ JSON

In-memory Acceleration ● Enable ○ Disable

GroupId: com.example

ArtifactId: demo

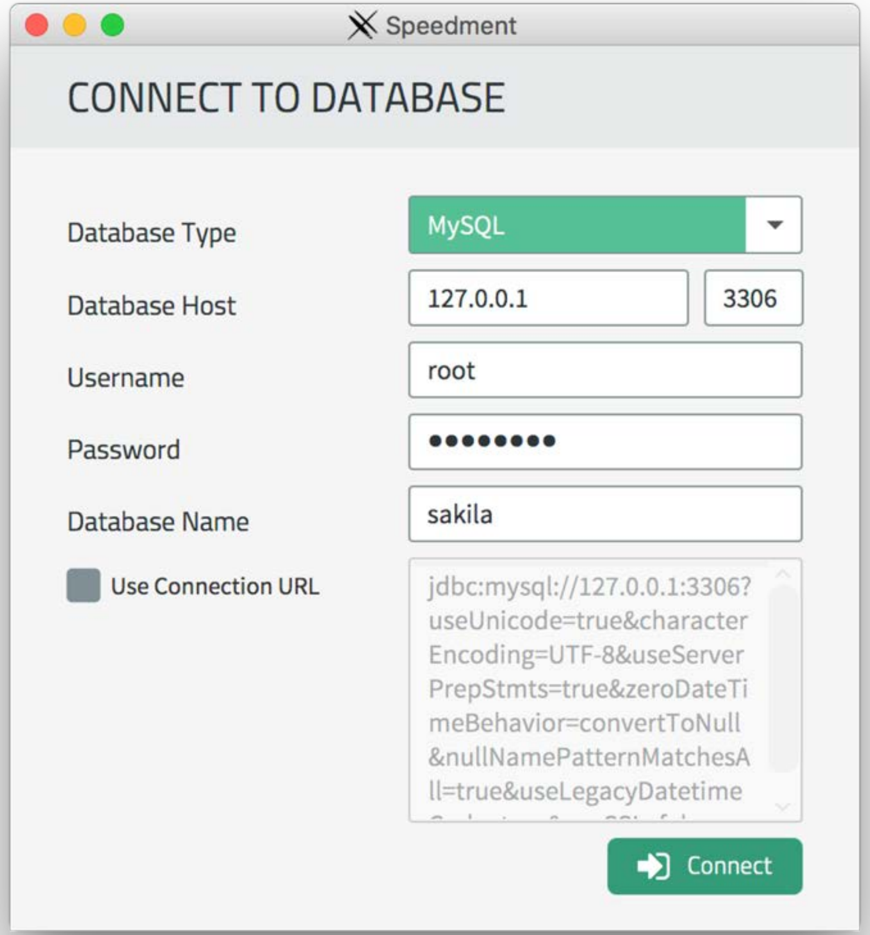
Version: 1.0.0-SNAPSHOT

DOWNLOAD

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
         xsi:schemaLocation="http://maven.apache.org/POM

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.bui
        <maven.compiler.source>1.8</maven.compiler.sourc
        <maven.compiler.target>1.8</maven.compiler.targe
        <mysql.version>5.1.46</mysql.version>
        <speedment.version>3.1.7</speedment.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId
            <version>${mysql.version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>com.speedment.enterprise</groupId>
```

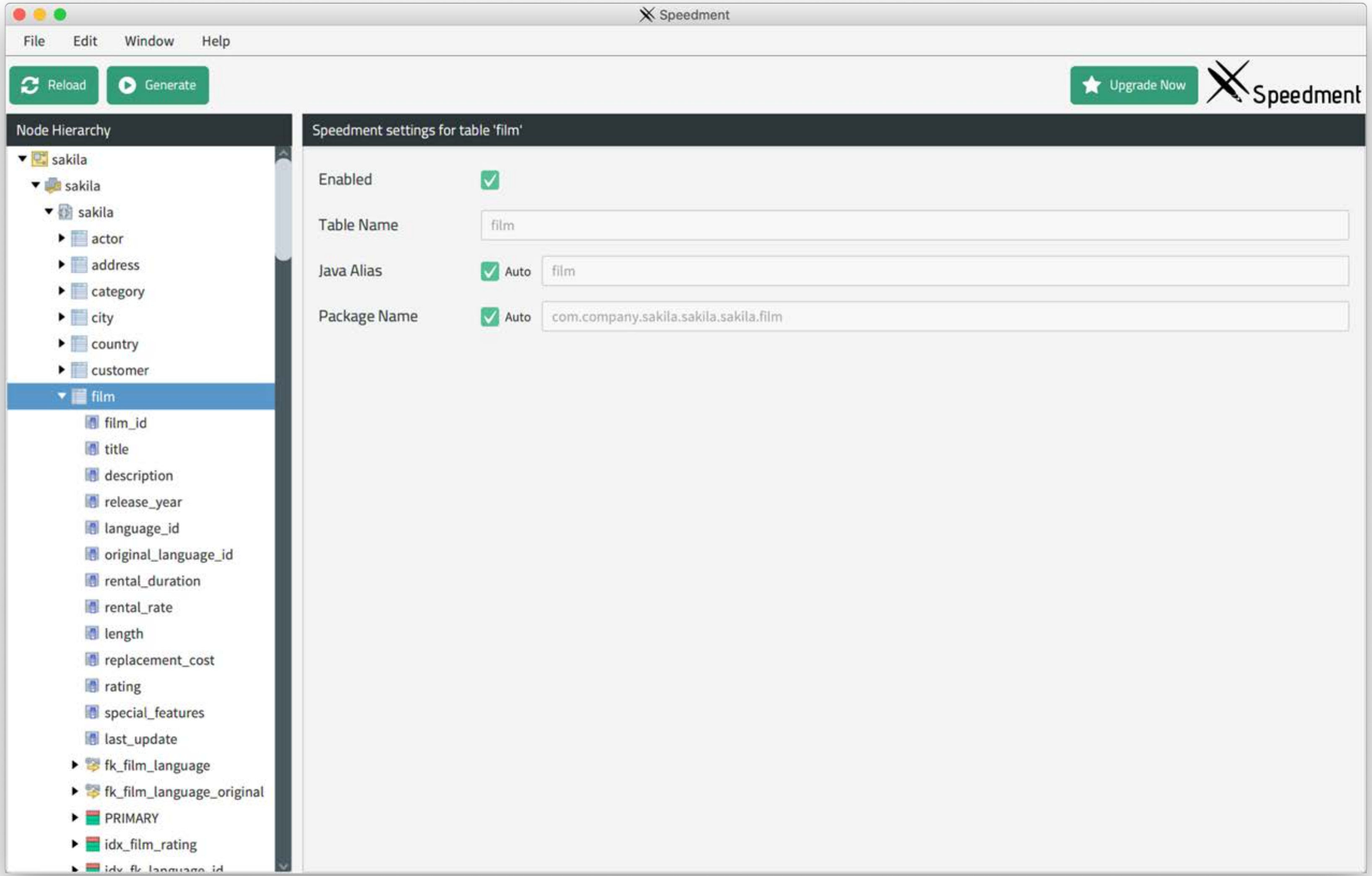
# Demo: Connect to the Sakila Database



CONNECT TO DATABASE

| | |
|---|---|
| Database Type | MySQL |
| Database Host | 127.0.0.1 3306 |
| Username | root |
| Password | ●●●●●●●●● |
| Database Name | sakila |
| ☐ Use Connection URL | jdbc:mysql://127.0.0.1:3306?useUnicode=true&characterEncoding=UTF-8&useServerPrepStmts=true&zeroDateTimeBehavior=convertToNull&nullNamePatternMatchesAll=true&useLegacyDatetime |

# Demo: Generate the Domain Model



Speedment settings for table 'film'

Enabled ☑

Table Name: film

Java Alias: ☑ Auto film

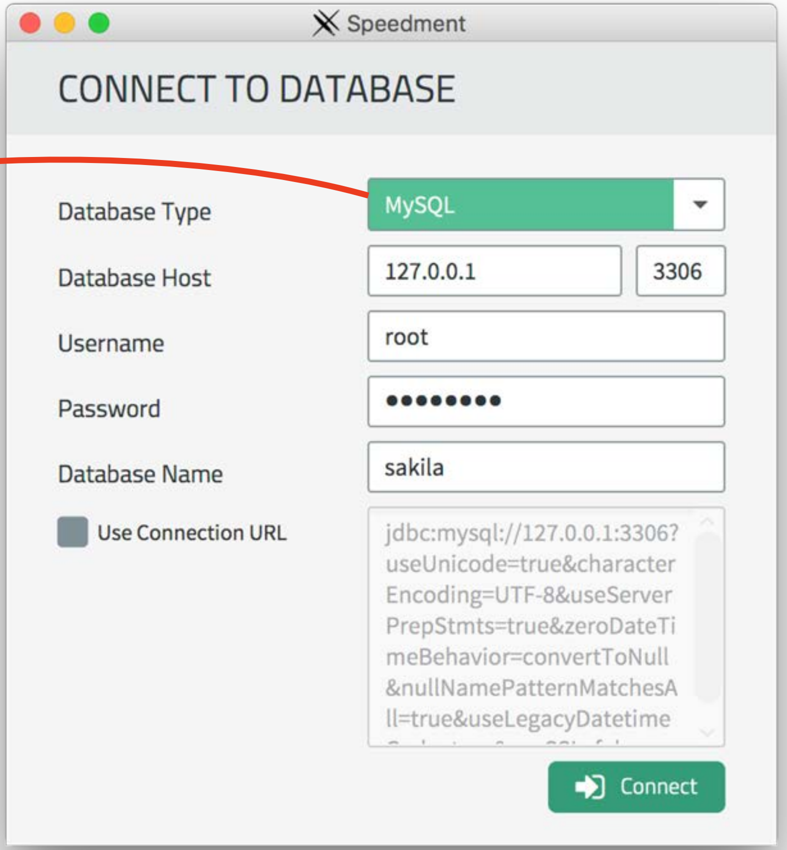
Package Name: ☑ Auto com.company.sakila.sakila.sakila.film

# Use Existing Infrastructure

How does it Fit with What We Have?

# Easy Integration: Any Data Source

ORACLE
DATABASE
IBM DB2
IBM®
AS/400®
PostgreSQL
CSV
BIN
MySQL
MariaDB
Google Sheets
AVRO
Microsoft®
SQL Server



CONNECT TO DATABASE

| | |
|---|---|
| Database Type | MySQL |
| Database Host | 127.0.0.1 3306 |
| Username | root |
| Password | ●●●●●●●●● |
| Database Name | sakila |
| □ Use Connection URL | jdbc:mysql://127.0.0.1:3306?useUnicode=true&characterEncoding=UTF-8&useServerPrepStmts=true&zeroDateTimeBehavior=convertToNull&nullNamePatternMatchesAll=true&useLegacyDatetime |

# Deploy Anywhere

IBM Bluemix™

amazon web services™

Microsoft Azure

On Premise

ORACLE® CLOUD

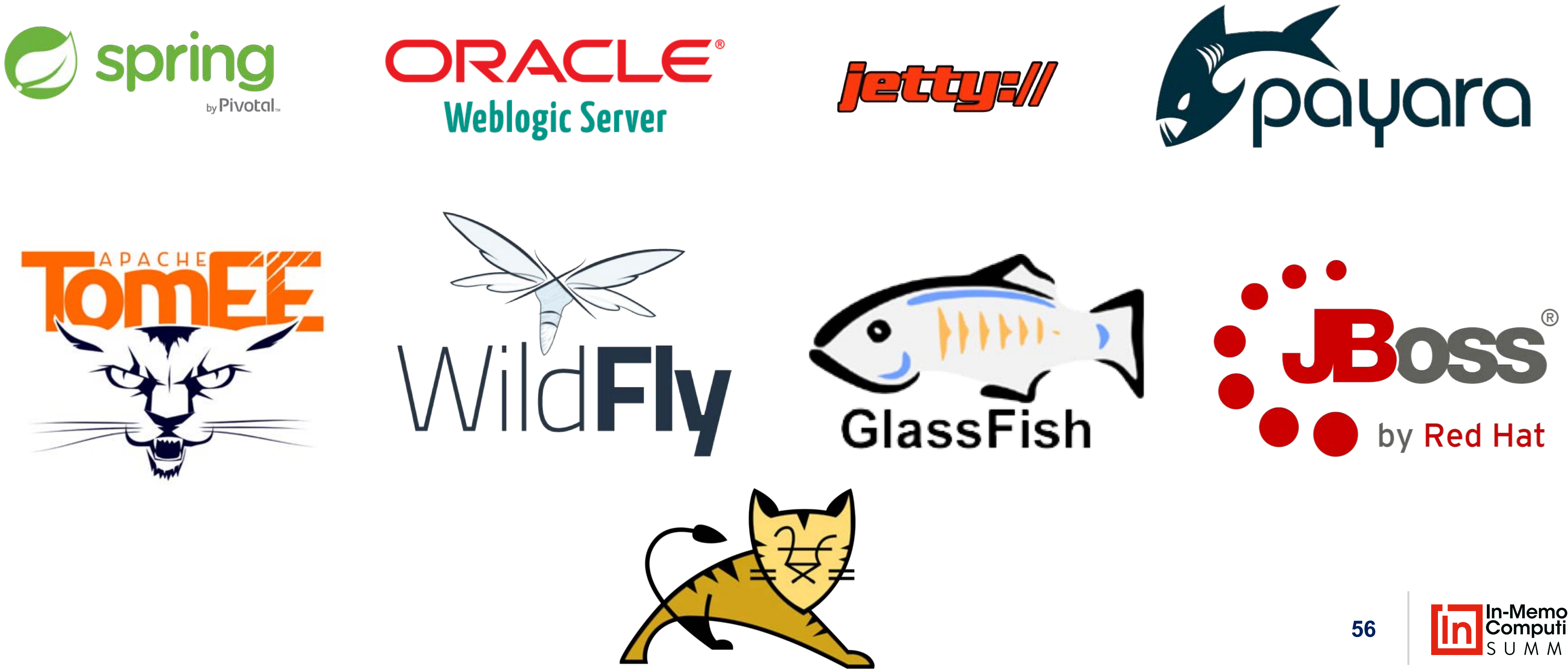
docker

Google Cloud Platform

kubernetes

# IDE Integration

# Web Service Integration



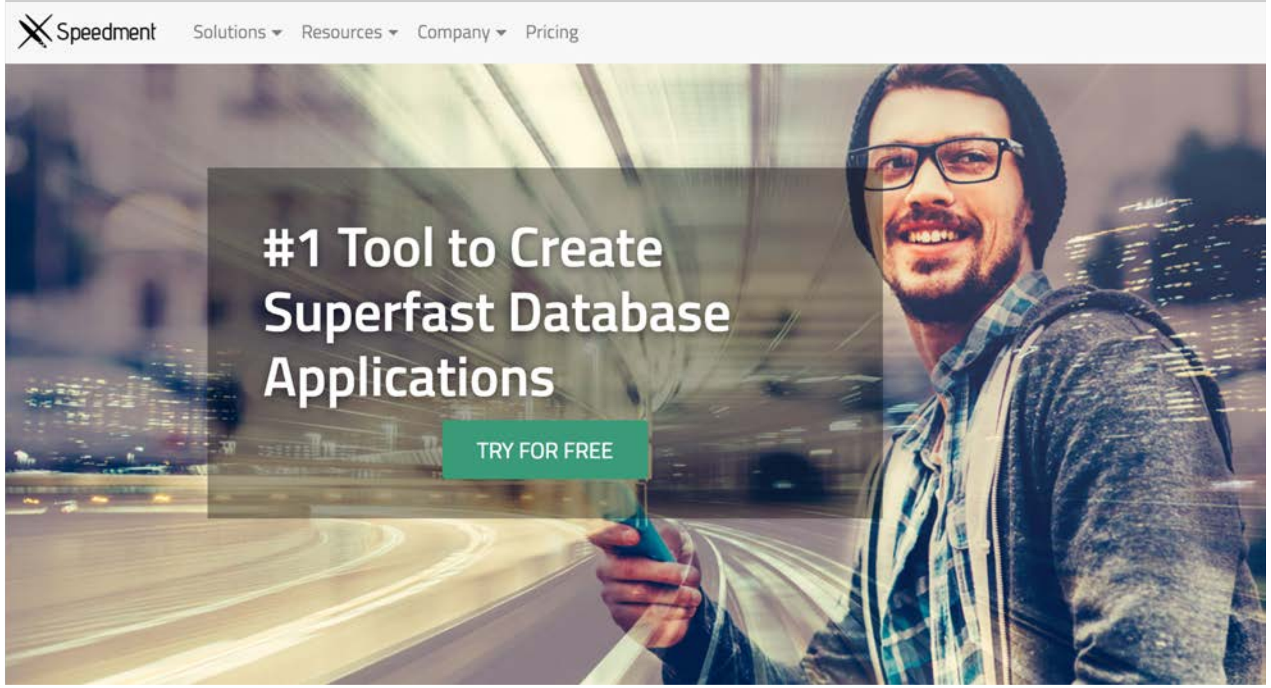
# Thanks

Trial License? Contact:

**Per Minborg**
minborg@speedment.com

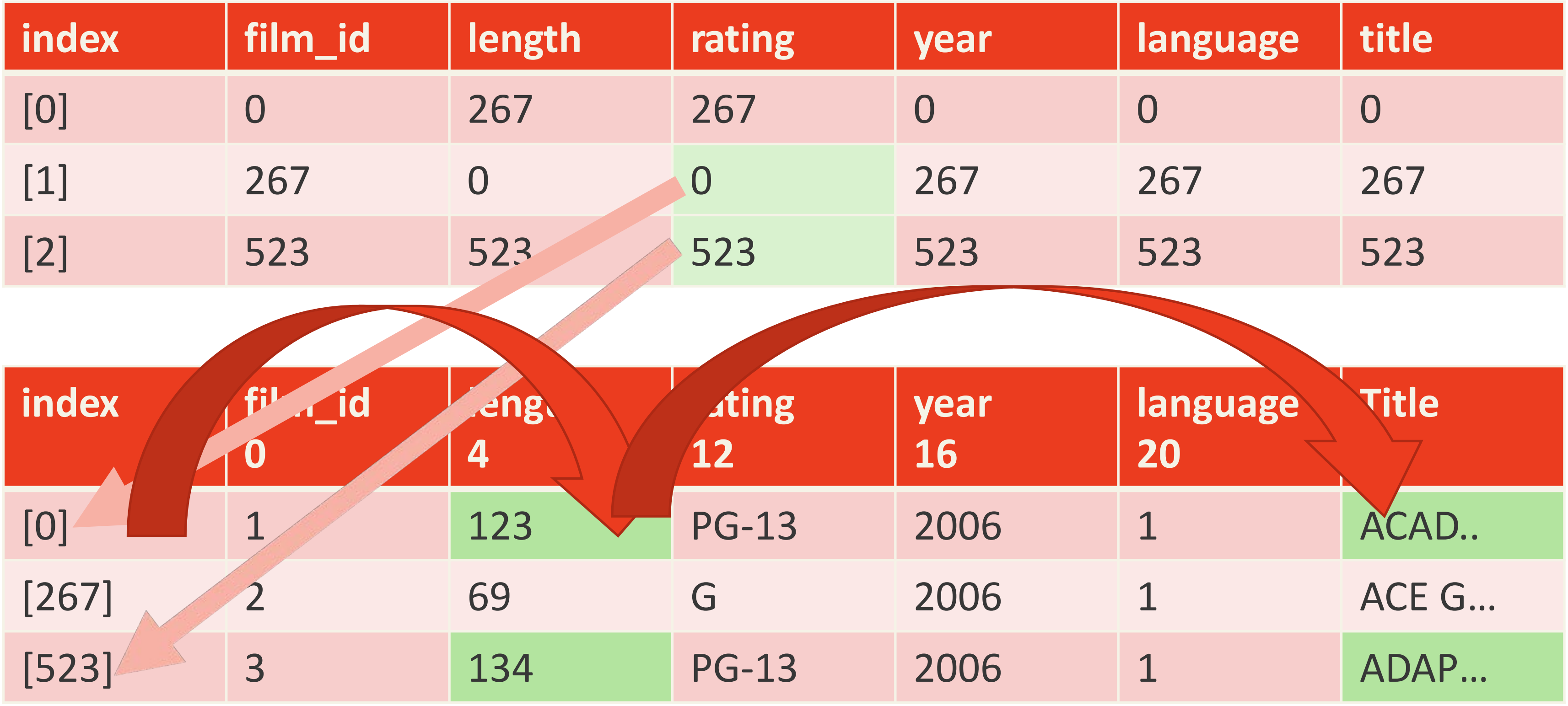
🌐 www.speedment.com/initializer

github.com/speedment/speedment



# Speedment Can Process Data without Creating Intermediate Objects

```
films.stream()
    .filter(Film.RATING.equal(Rating.PG13))
    .collect(toJsonLengthAndTitle()));
```



| index | film_id | length | rating | year | language | title |
|---|---|---|---|---|---|---|
| [0] | 0 | 267 | 267 | 0 | 0 | 0 |
| [1] | 267 | 0 | 0 | 267 | 267 | 267 |
| [2] | 523 | 523 | 523 | 523 | 523 | 523 |

| index | film_id<br>0 | length<br>4 | rating<br>12 | year<br>16 | language<br>20 | Title |
|---|---|---|---|---|---|---|
| [0] | 1 | 123 | PG-13 | 2006 | 1 | ACAD.. |
| [267] | 2 | 69 | G | 2006 | 1 | ACE G… |
| [523] | 3 | 134 | PG-13 | 2006 | 1 | ADAP… |