

 SILICON VALLEY

 In-Memory  
Computing / SUMMIT  
2017

# HOW TO USE JAVA STREAMS TO ACCESS EXISTING DATA WITH ULTRA-LOW LATENCY

PER MINBORG, CTO, SPEEDMENT, INC.

# WHO AM I?

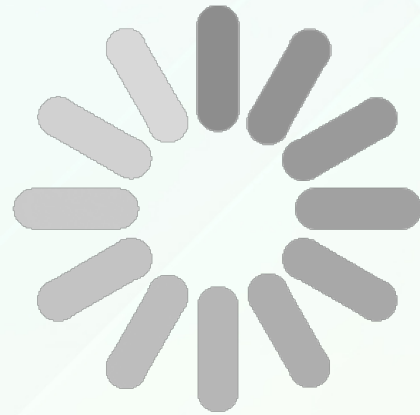
- Serial Entrepreneur
- +15 US Patents
- Java Expert
- Palo Alto
- Minborg's Java Pot



# NEED FOR SPEED™



# SPEED INVERTED



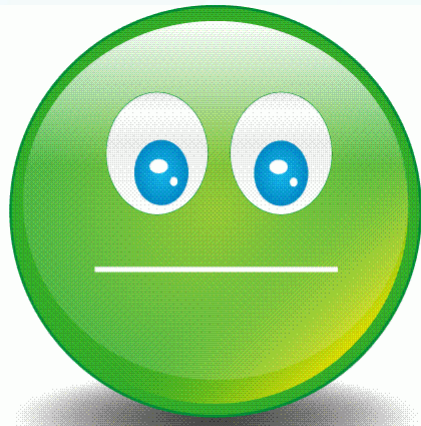
# WHY ARE DELAYS A PROBLEM?

- Bad User Experience
  - 100 ms : direct response
  - 1 second: experienced a delay
  - 3 seconds: becomes frustrated, 57% leave the site
  - 10 seconds: 100% tired

# WHY ARE DELAYS A PROBLEM?



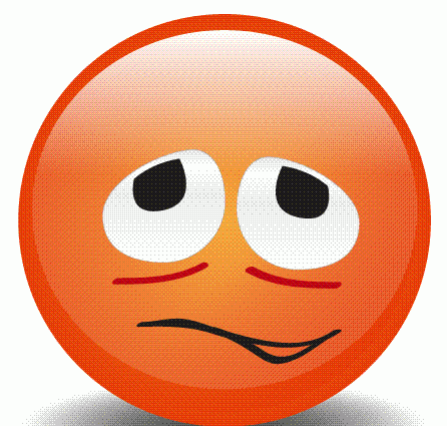
100 ms



1 s



3 s



10 s

# WHY ARE DELAYS A PROBLEM?

- Less Page Views      Google lost 20% traffic with half a second delay
- Less Revenue      Amazon lost 1% of sales for every 100 ms delay
- Higher Overhead      Unnecessary hardware and license cost
- Destroys the Brand      44% worry when paying transactions take too long

Sun



# OTHER AREAS WHERE SPEED MATTERS

- Fintech and High Frequency Trading
- AI
- IoT
- Defense, Intelligence and Situation Awareness
- Logistics
- Science Applications (e.g. Space, DNA)
- Microservice Architecture
- General Computing

# REQUIREMENTS

- Low-latency
- Deterministic behavior
- Low memory footprint
- Low CPU utilization
- Low memory pressure
- Parallelism
- Scale out capability
- ...



# LATENCY REQUIREMENT BREAK-DOWN

- It all adds up...
- $L_{\text{tot}} = \sum L_n$  with maybe millions of steps in less than perhaps one second
- We need operations that can complete well into the nanoseconds (~200 ns)

# WHAT ABOUT CLUSTERS OF NODES?

- SF - NY speed of light latency is  $> 15 \text{ ms} * 2 * (3/2) > 45 \text{ ms}$  for fiber
- TCP roundtrip latency with two Linux hosts connected directly with 10Gb/s Ethernet
  - Some tweaks 40 us
  - Busy polling and CPU affinity 30 us
  - Expert mode ~25 us
- Routers and switches introduce significant additional delays
- AWS, Google Cloud, Bluemix etc. introduces significant additional network delays even on co-located servers

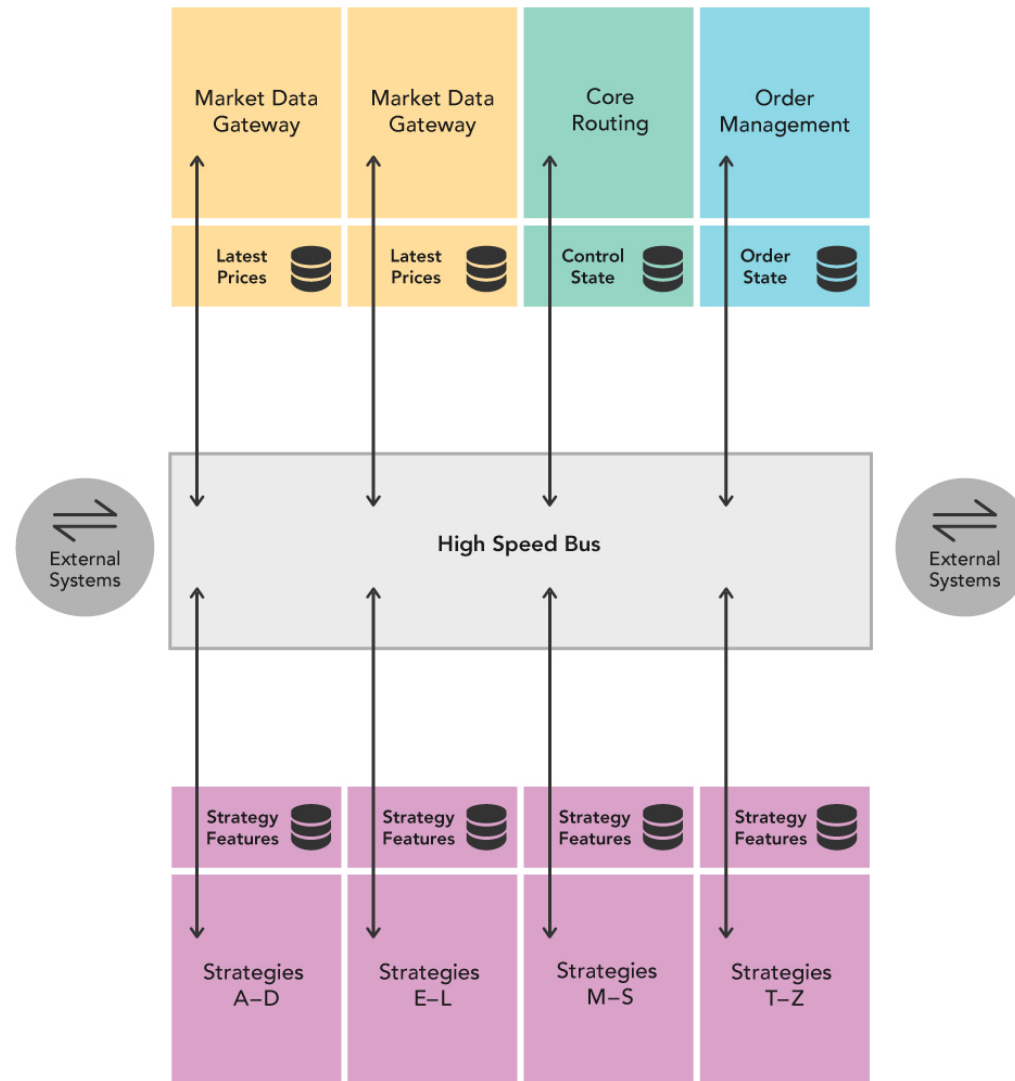
# HOW ABOUT DIFFERENT PROCESSES ON THE SAME MACHINE?

- Inter-Process Communication is in the milliseconds
- Context Switch -> L1, L2, L3 + TLB affected

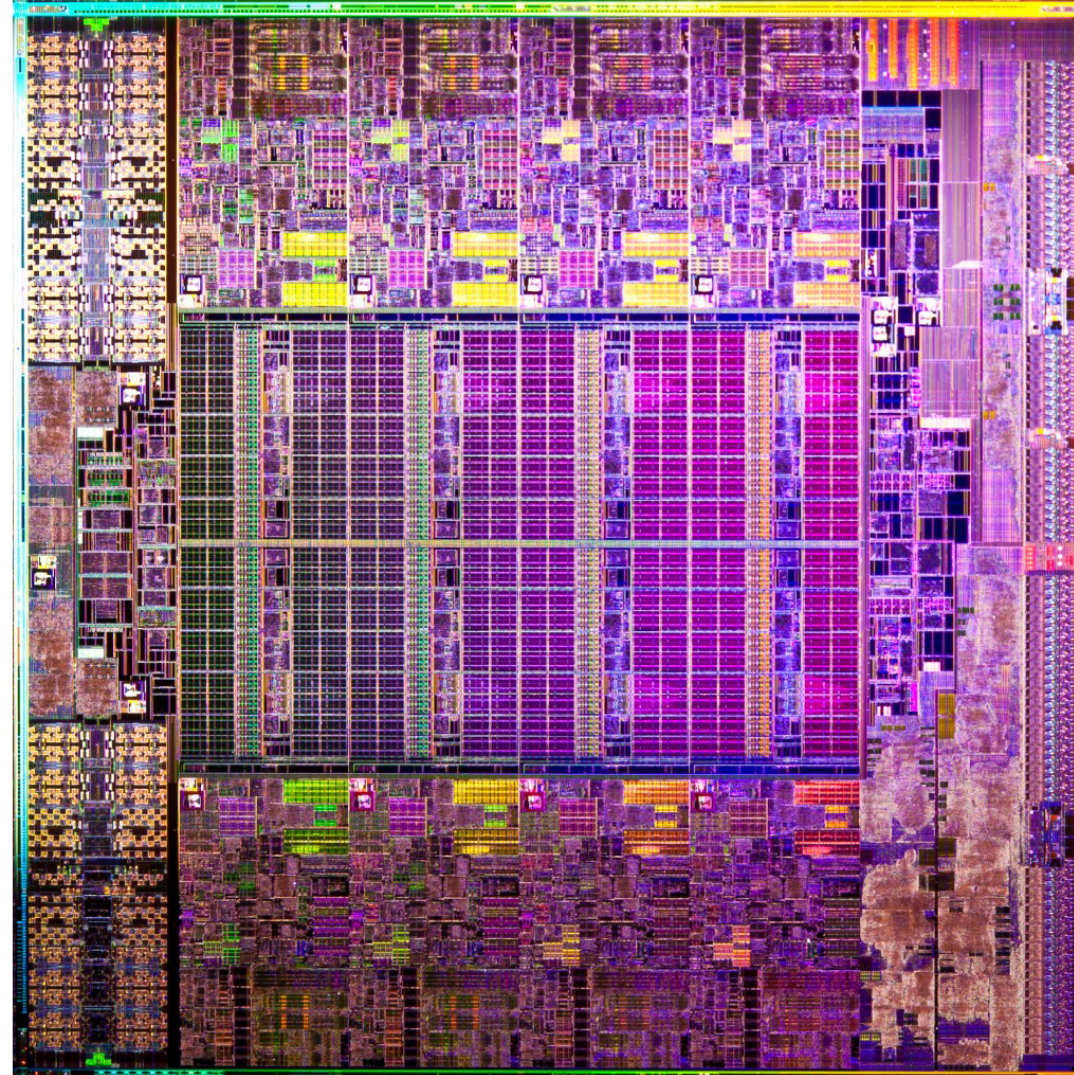
# WITHIN THE JVM ITSELF

- Main Memory Read ~100 ns
- Volatile read
- L3 ~20ns
- L2 ~7ns
- L1 ~0.5ns
- CPU Registers

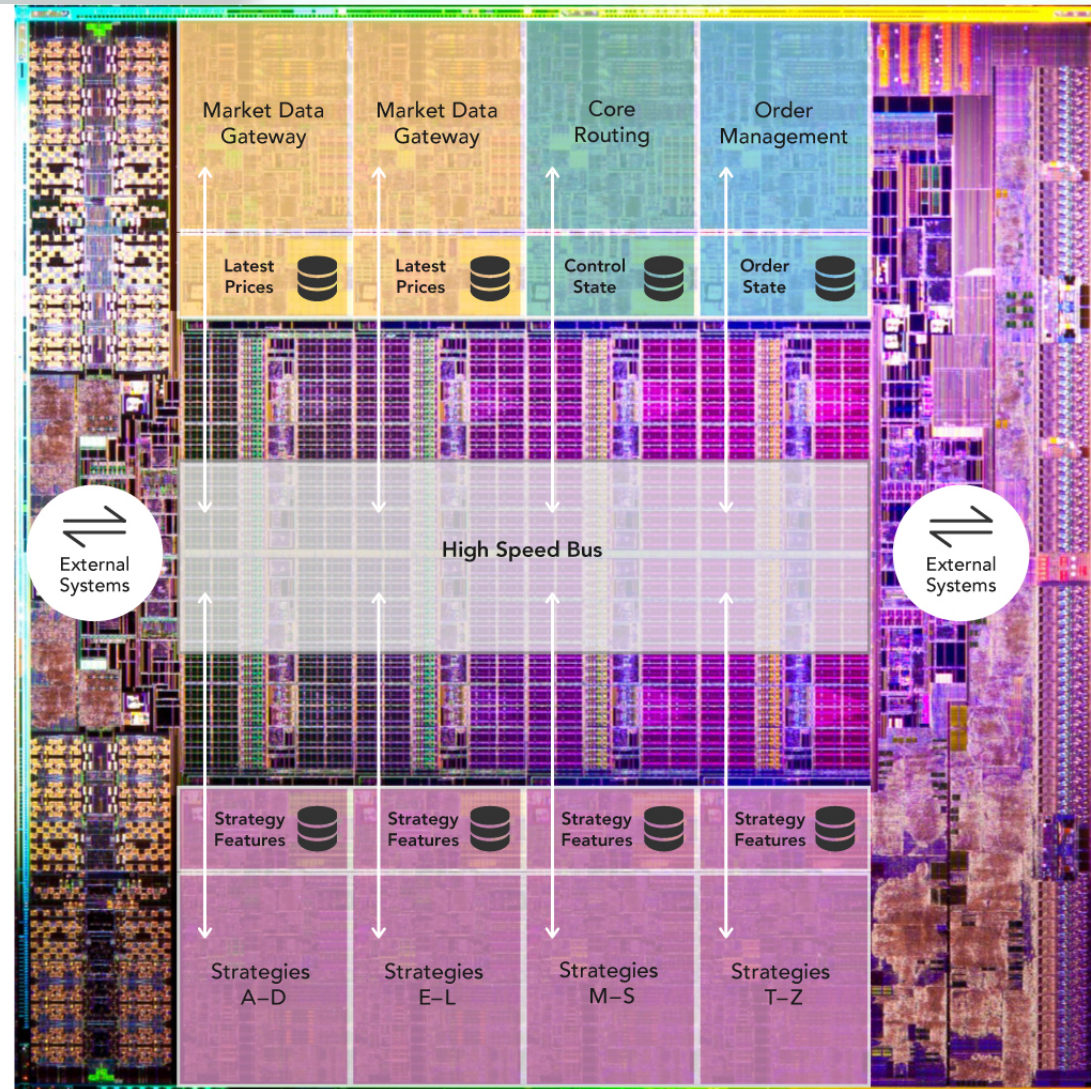
# MICROSERVICE ARCITECTURE APPLICATION



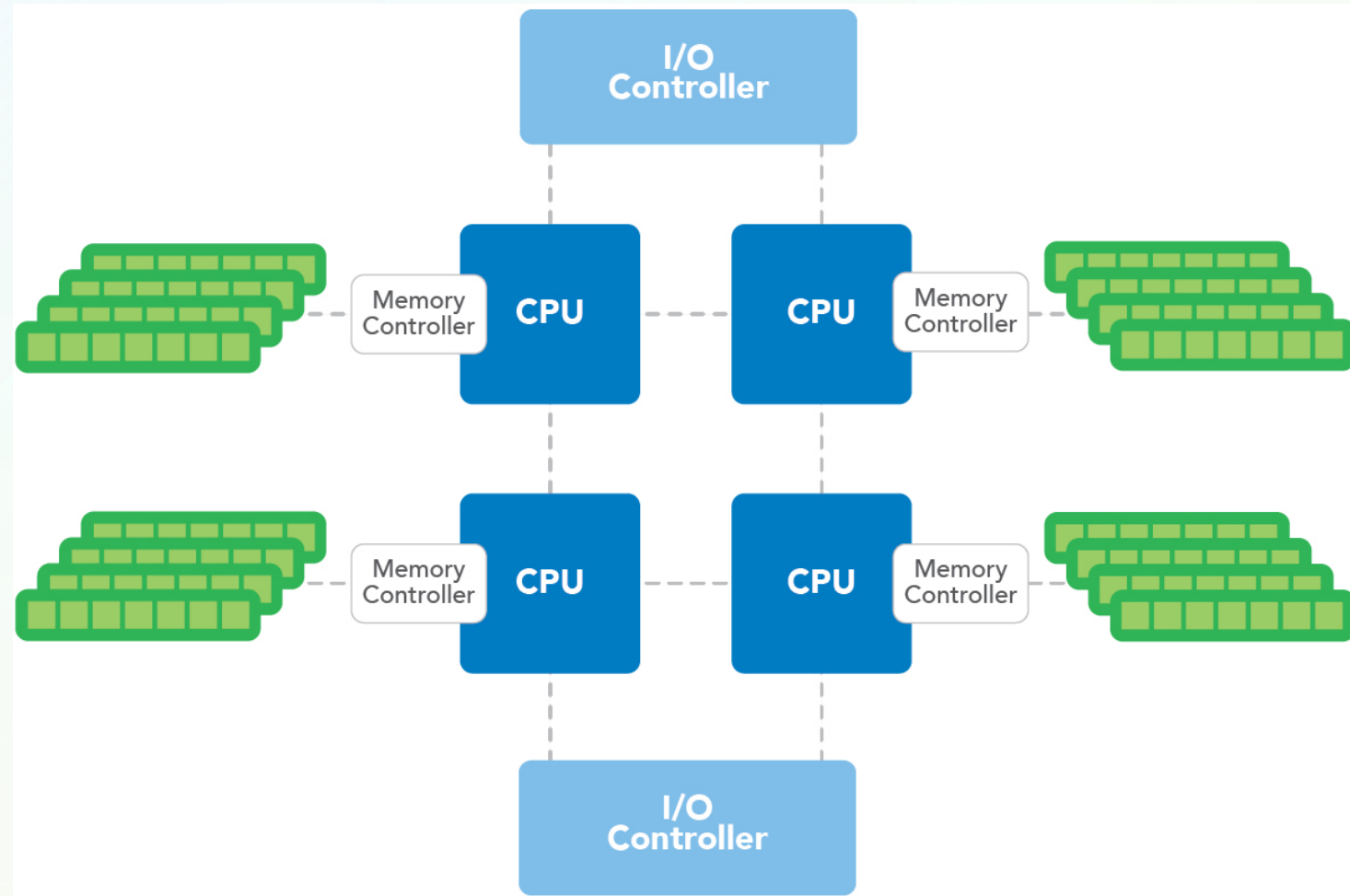
# MULTI-CORE INTEL CPU



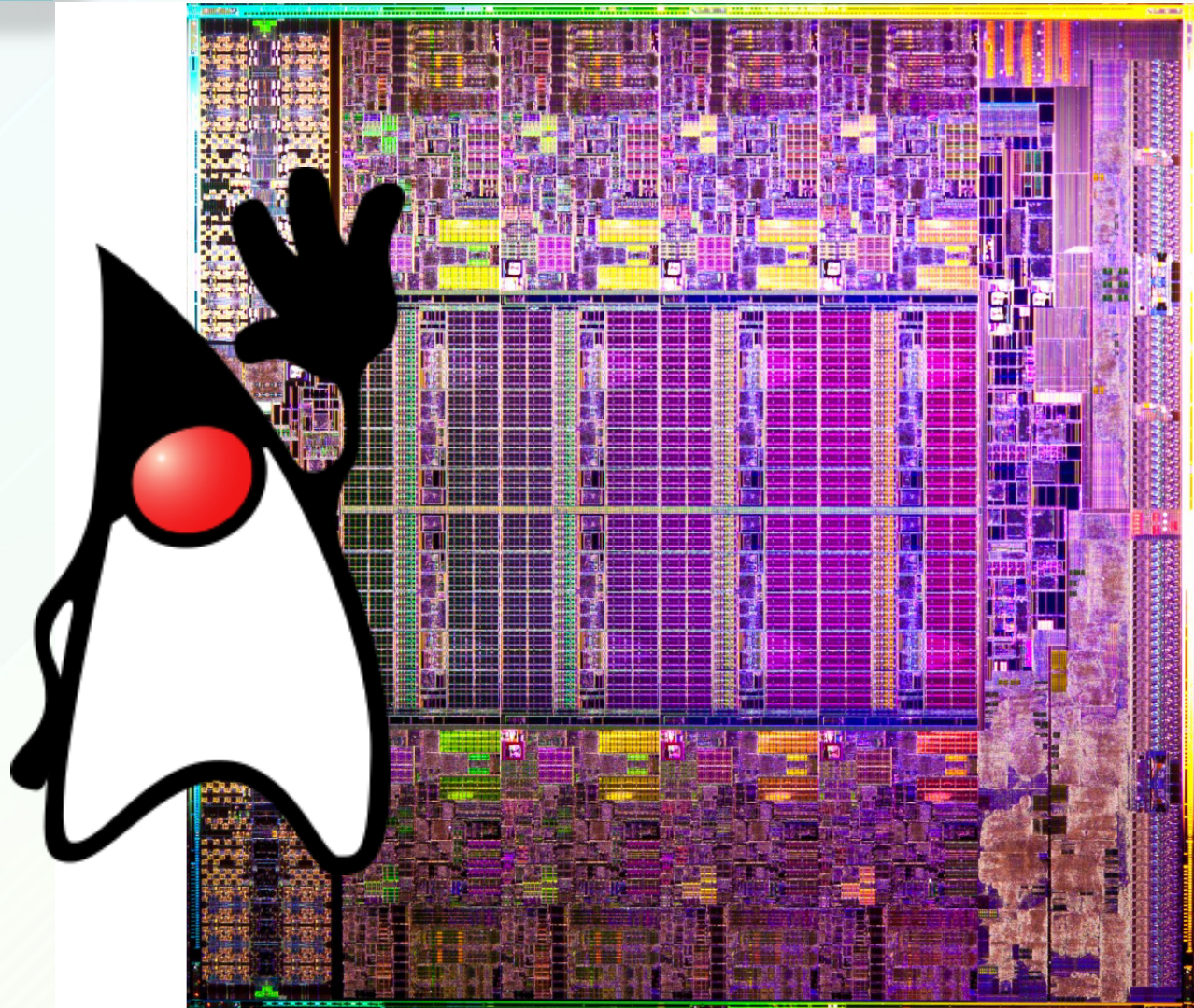
# UNDERSTANDING HARDWARE



# UNDERSTANDING HARDWARE



# CONCLUSION: IN-JVM-MEMORY



# API – STANDARD JAVA STREAM



# COMPARISON BETWEEN SQL AND STREAM OPERATIONS

SQL	Java Stream Operations(s)
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()

Java™  
magazine

By and for the Java community

MAY/JUNE 2017

# Libraries

10 LOMBOK:  
ANNOTATIONS FOR  
CLEANER CODE

16 JDEFERRED'S  
ASYNC EVENT  
MANAGEMENT

34 DATABASE  
ACCESS WITH  
STREAMS

28 BEST PRACTICES  
FOR LIBRARY  
DESIGN

MAY/JUNE 2017

```
//databases /
```



PER MINBORG

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

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

**W**hy 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



# DECLARATIVE CONSTRUCTS IN BOTH SQL AND STREAM

```
SELECT * FROM FILM  
WHERE RATING = 'PG-13'
```

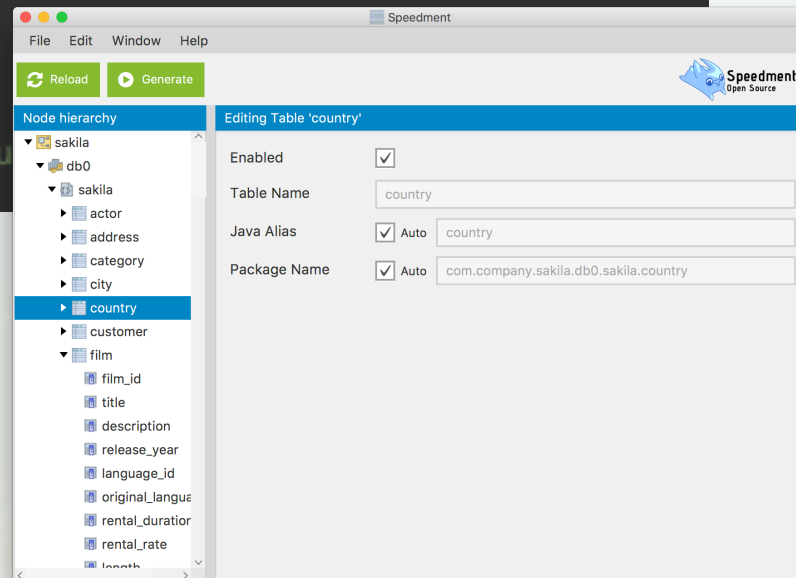
```
films.stream()  
  .filter(Film.RATING.equal(Rating.PG13))
```

# SPEEDMENT

## 1. Java stream ORM-tool

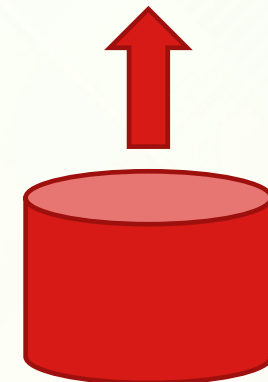
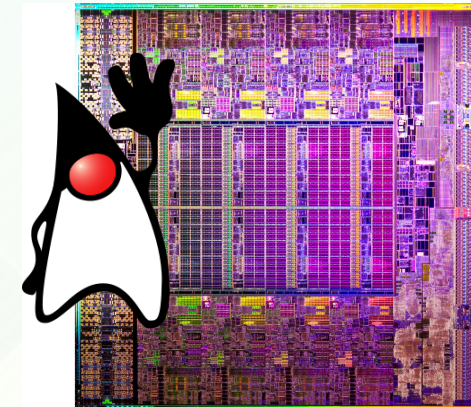
```
long count = films.stream()
    .filter(Film.RATING.equal("PG-13"))
    .filter(Film.LENGTH.greaterThan(75))
    .map(Film::getTitle)
    .sorted()
    .count();

System.out.format("Four
```

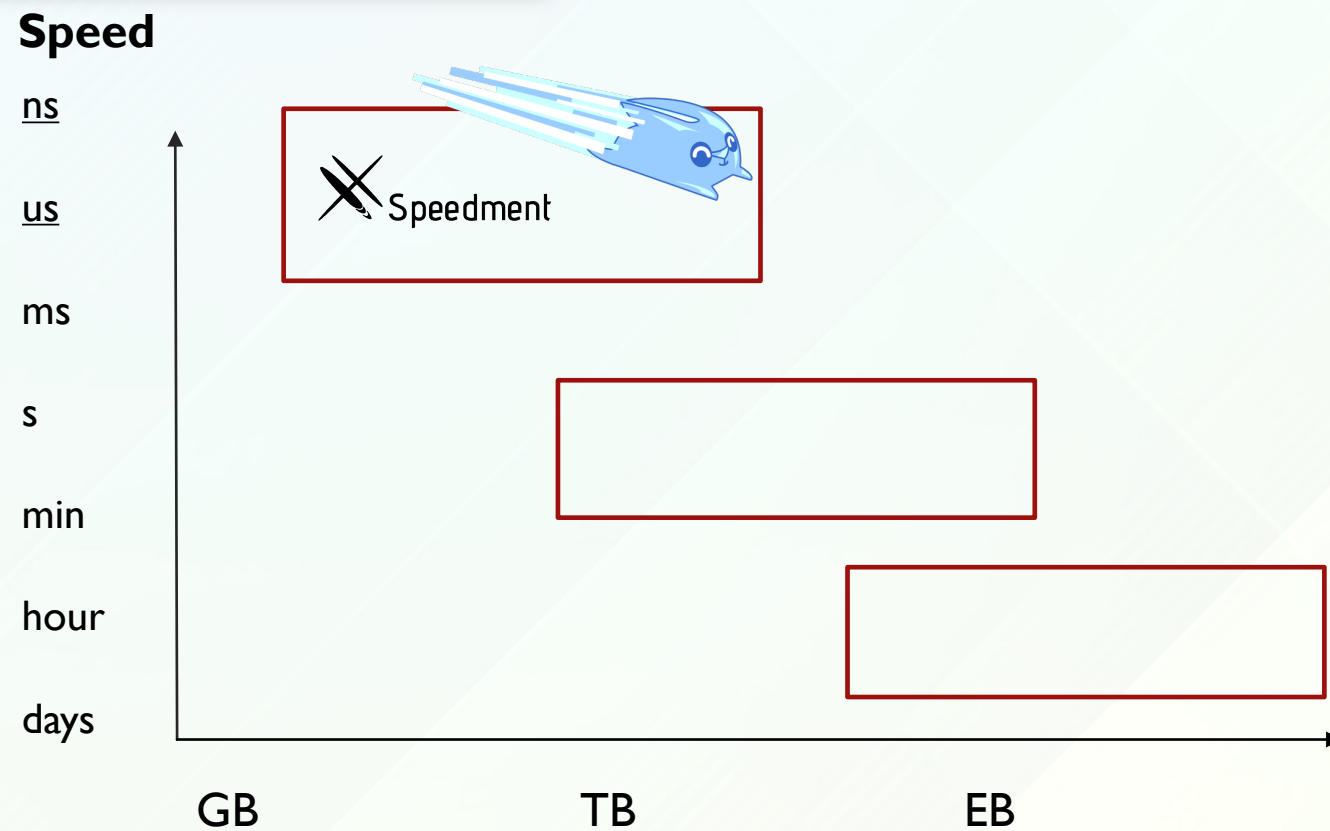


## 2. In-JVM Memory

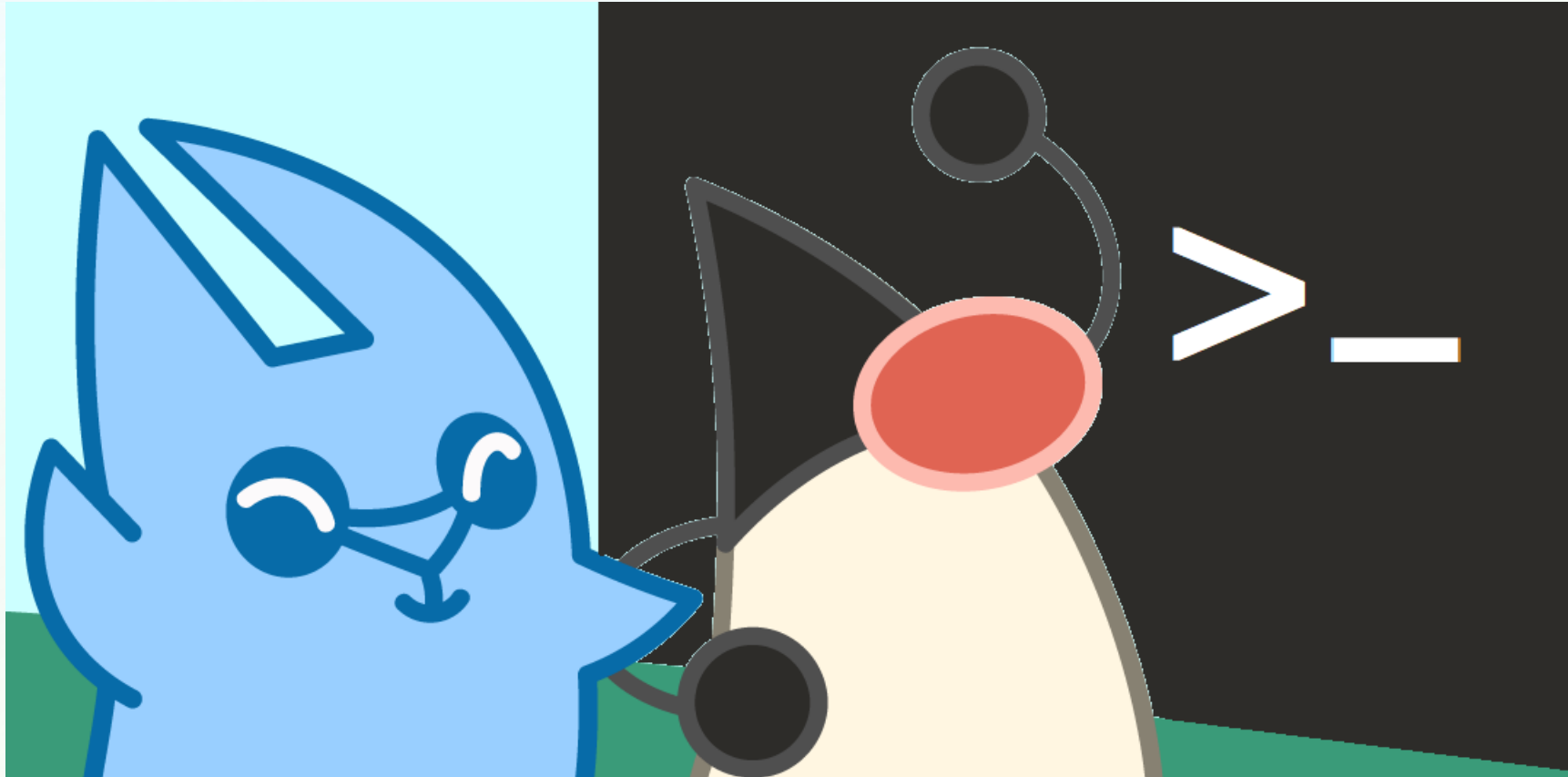
&



# MARKET POSITION



# JAVA 9 DEMO



# THE SOLUTION

- In-JVM-Memory Access with a Java Stream API
- Streams introspect their own pipeline
- Off-Heap storage
- MVCC immutable snapshots
- Light weighted Off-Heap indexing
- $O(1)$  and  $O(\log(N))$  operations
- Collectors that do not create intermediate objects
- Aggregators that do not create intermediate objects
- Snapshot compression/folding
- Stack allocation of objects instead of heap allocation

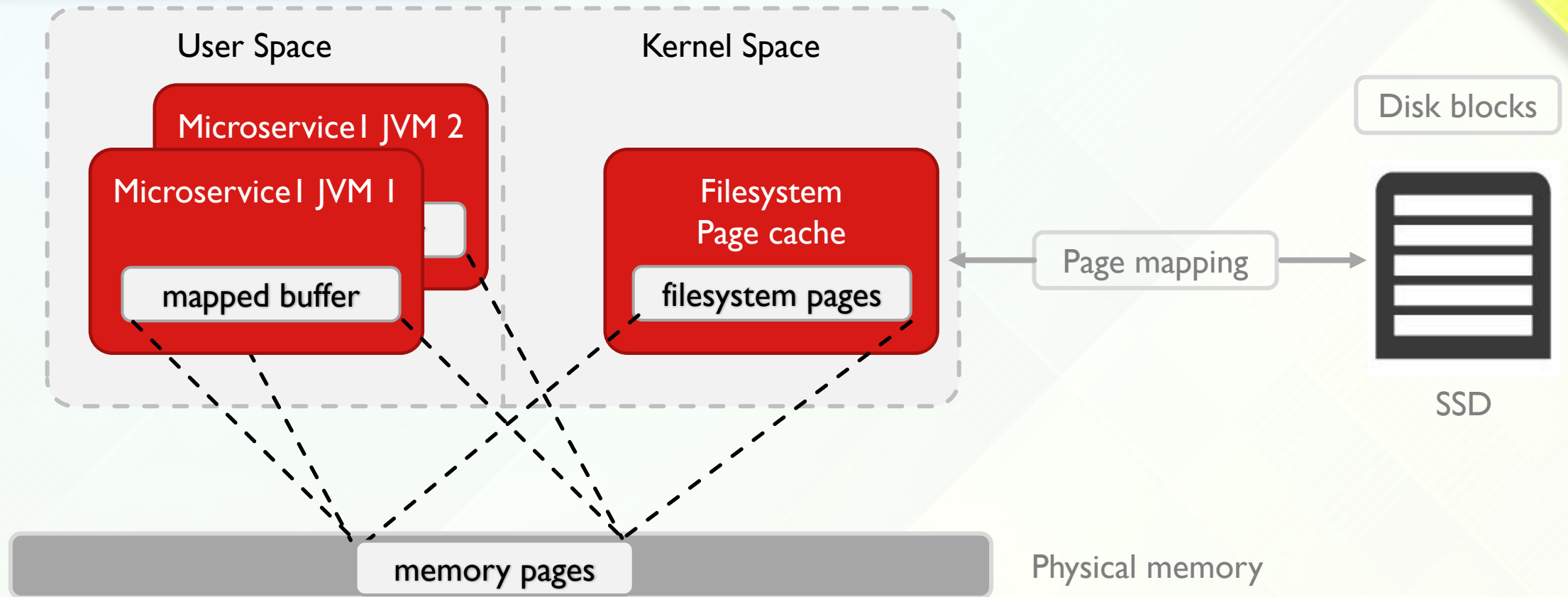
# COLLECTOR WITHOUT INTERMEDIATE OBJECTS

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

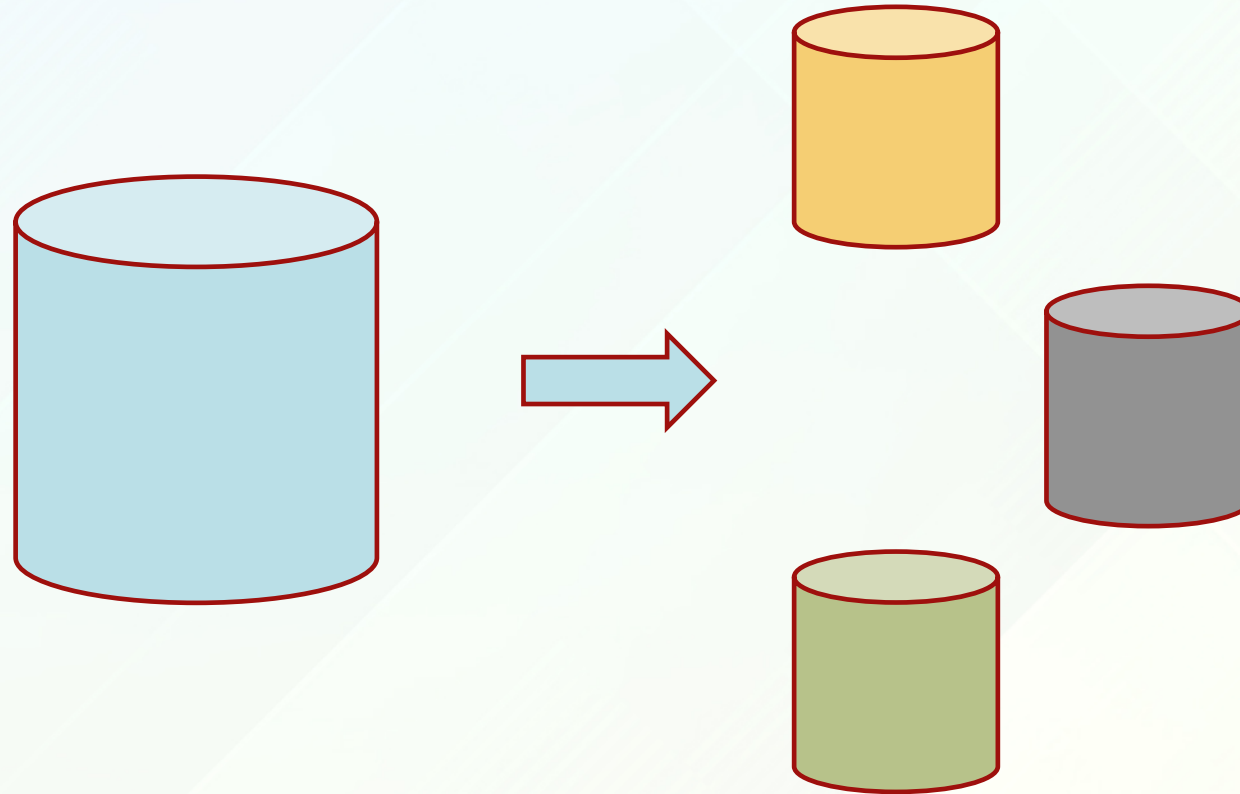
index	film_id	length	rating	year	language	title
[0]	0	267	267	0	0	0
[1]	267	0	0	267	267	267
[2]	523	523	523	523	523	523

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


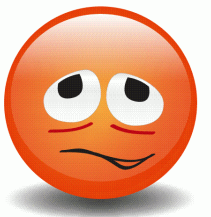
# SCALING OUT – MULTIPLE NODES



# SCALING OUT - SHARDING

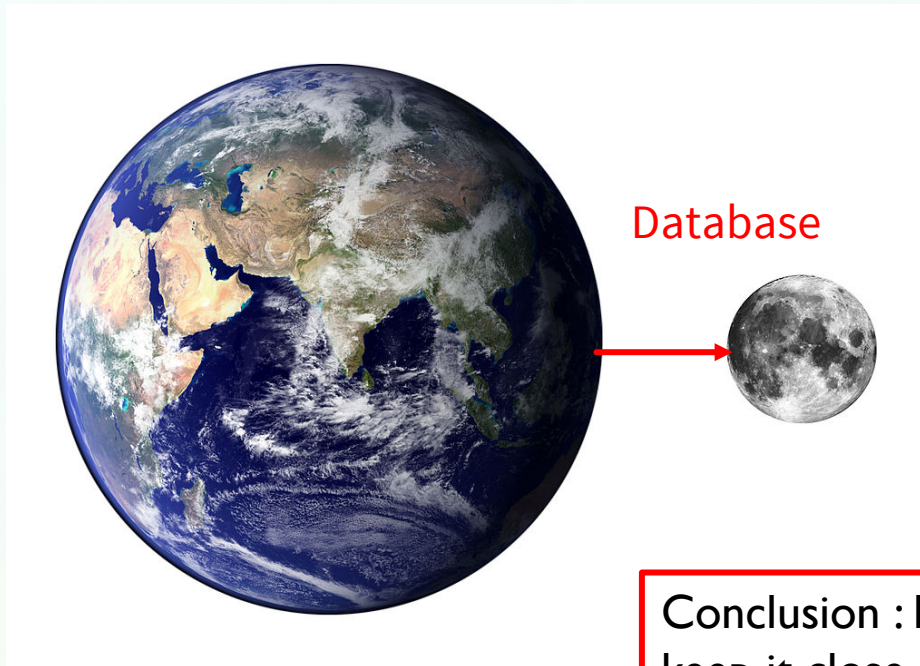


# WHY IS SPEED IMPORTANT?

	Off-Heap in-JVM-memory		Objects in-memory	
Average latency [ms]	105		1,100	
99.5% percentile [ms]	160		~7,000	
Nodes	2		8	
Major GCs	0		27	
Total RAM [GB]	128		2,048	
Total CPUs	4		64	
Average CPU utilization	40%		2,100%	
Initial ingestion time [min]	2		28	
Operating cost	\$		\$\$\$	
User experience	+++		+	

# THE DIFFERENCE

During the time a database makes a one second query, how far will the light move?



Conclusion : Do not place your data on the moon, keep it close by using in-JVM-memory technology!

# INTEGRATES WITH ANY DATA SOURCE



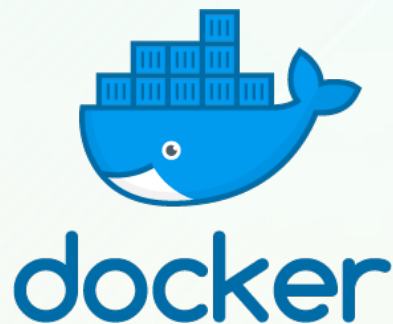
# DEPLOY ANYWHERE



On Premise



**ORACLE**<sup>®</sup> Cloud



# IDE INTEGRATION



# INTEGRATION



# APPLICATION API



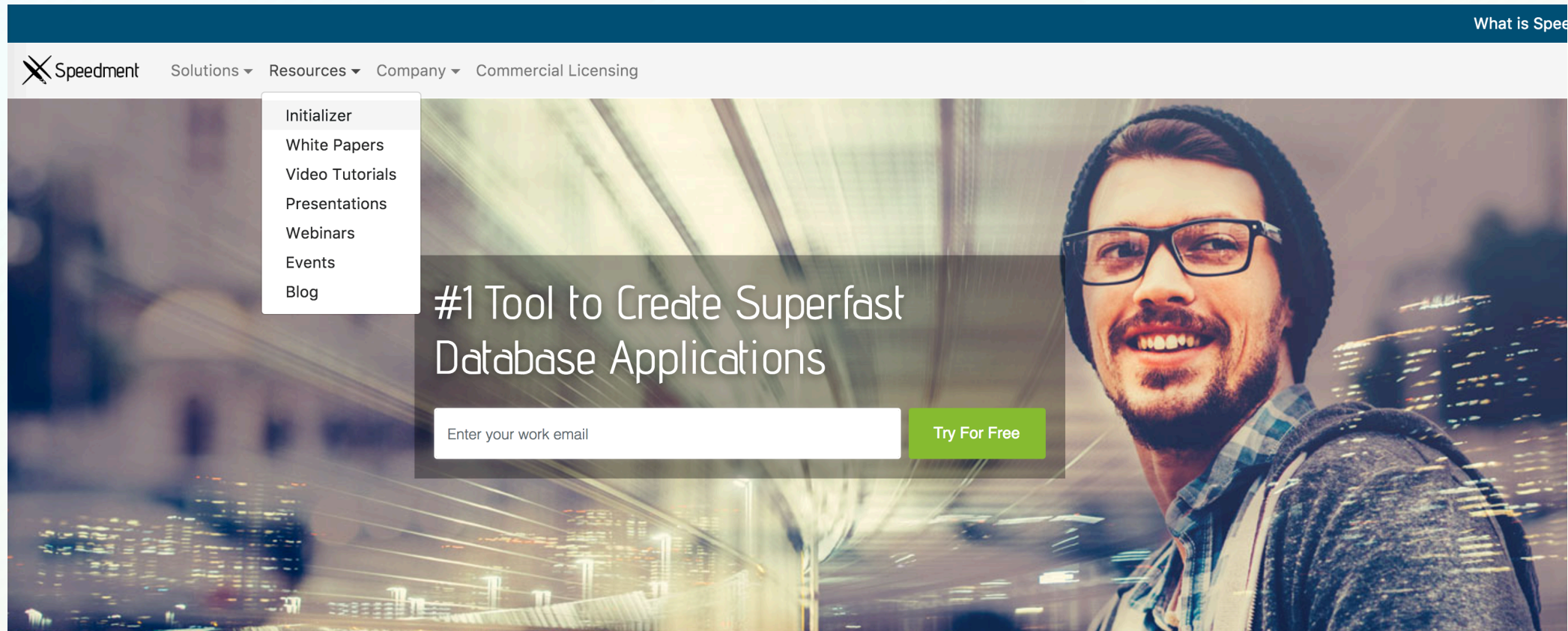
vaadin }>



# STEPWISE INTRODUCTION



# TRY IT!



The screenshot shows the Speedment website homepage. The header is dark blue with the Speedment logo (a stylized 'X' icon) and navigation links: Solutions, Resources, Company, and Commercial Licensing. A dropdown menu is open under 'Resources', listing:\_INITIALIZER, White Papers, Video Tutorials, Presentations, Webinars, Events, and Blog. The main hero section features a background image of a smiling man with glasses and a beanie, with a blurred city skyline at night. Overlaid on this image is the text '#1 Tool to Create Superfast Database Applications'. Below this text is a white input field labeled 'Enter your work email' and a green button labeled 'Try For Free'.

What is Speedment

Speedment Solutions Resources Company Commercial Licensing

- Initializer
- White Papers
- Video Tutorials
- Presentations
- Webinars
- Events
- Blog

#1 Tool to Create Superfast Database Applications

Enter your work email Try For Free

# TRY IT!



Solutions ▾ Resources ▾ Company ▾ Commercial Licensing

## Initializer

[Home](#) > Initializer

The Initializer makes it easy to setup a new Speedment project with Maven. Fill in your project details and see the configuration change in real time. If you choose any Enterprise features you will be asked to fill in a License Key. Just send us a request and you will [get a 30 days free trial license!](#)

Database Type	<input checked="" type="radio"/> MySQL <input type="radio"/> PostgreSQL <input type="radio"/> MariaDB <input type="radio"/> Oracle <input type="radio"/> DB2 <input type="radio"/> AS400 <input type="radio"/> SQL Server
JDBC Driver Version	<input type="text" value="5.1.42"/>
In-memory Acceleration	<input type="radio"/> Enable <input checked="" type="radio"/> Disable
GroupId	<input type="text" value="com.example"/>
ArtifactId	<input type="text" value="demo"/>
Version	<input type="text" value="1.0.0-SNAPSHOT"/>

\* Enterprise Features require a valid License Key.

[Generate Project](#)

Main.java

pom.xml

```
public static void main(String... param) {  
    YourApplication app = new YourApplicationBuilder()  
        .withUsername("your-dbms-username")  
        .withPassword("your-dbms-password")  
        .build();  
  
    // You are ready to go!  
}
```

# THANK YOU!

- [minborg@speedment.com](mailto:minborg@speedment.com)
- Mention IMCS to get 30 min free consultation (Nov)
- [Calendly.com/speedment](https://calendly.com/speedment)



# INTEGRATION

