Using PostgreSQL with Java

Álvaro Hernández Tortosa< aht@8kdata.com >

PgConf.Ru 2017

Who I am

- What we do @8Kdata:
  - ✓Creators of ToroDB.com, NoSQL & SQL database
  - ✓Database R&D, product development
  - ✓Training and consulting in PostgreSQL
  - ✓PostgreSQL Support

Twitter: @ahachete

Linkedin:
http://es.linkedin.com/in/alvarohernandeztortosa/

CEO @ 8Kdata, Inc.

Founder of the Spanish PUG (postgrespana.es) President,

PostgreSQL España Association (~750members as of today)

Agenda

1. Introduction to Java and PostgreSQL
2. Ways of connecting to PostgreSQL from Java (not only JDBC!)
3. Introduction to JDBC. JDBC types. PostgreSQL JDBC
4. Code demo: JDBC with PostgreSQL. From Java 1.7 to Java 8, best practices and code samples
5. Code demo: MyBatis, jOOQ
6. Java inside PostgreSQL
7. JDBC performance
8. HikariCP + FlexyPool
9. The future of Java and PostgreSQL

PostgreSQL and Java

PostgreSQL and Java. Do they fit well?

- There seems to be small interest within the community:

  ✓ pgsql-jdbc is not a high traffic ml (2.15 msg/day)

  ✓ pl-java started strong, faded away, came back as 1.5 and now works for 9.4 & 9.5

  ✓ JavaScript, Python, Ruby, Go seem to rule the programming ecosystem around postgresql

- There are no native APIs for Java
- Java = ORM = Hibernate = suckz → Java suckz

PostgreSQL and Java. They do fit well

- Java IS the enterprise language
- Arguably, there is more Java code accessing PostgreSQL than from any other programming language
- Both Java and PostgreSQL are mature, reliable and trusted
- Several commercial, big data and postgres derivatives use and/or are interfaced via Java
- There is a mature and reliable option to connect to PostgreSQL (the JDBC PostgreSQL driver)

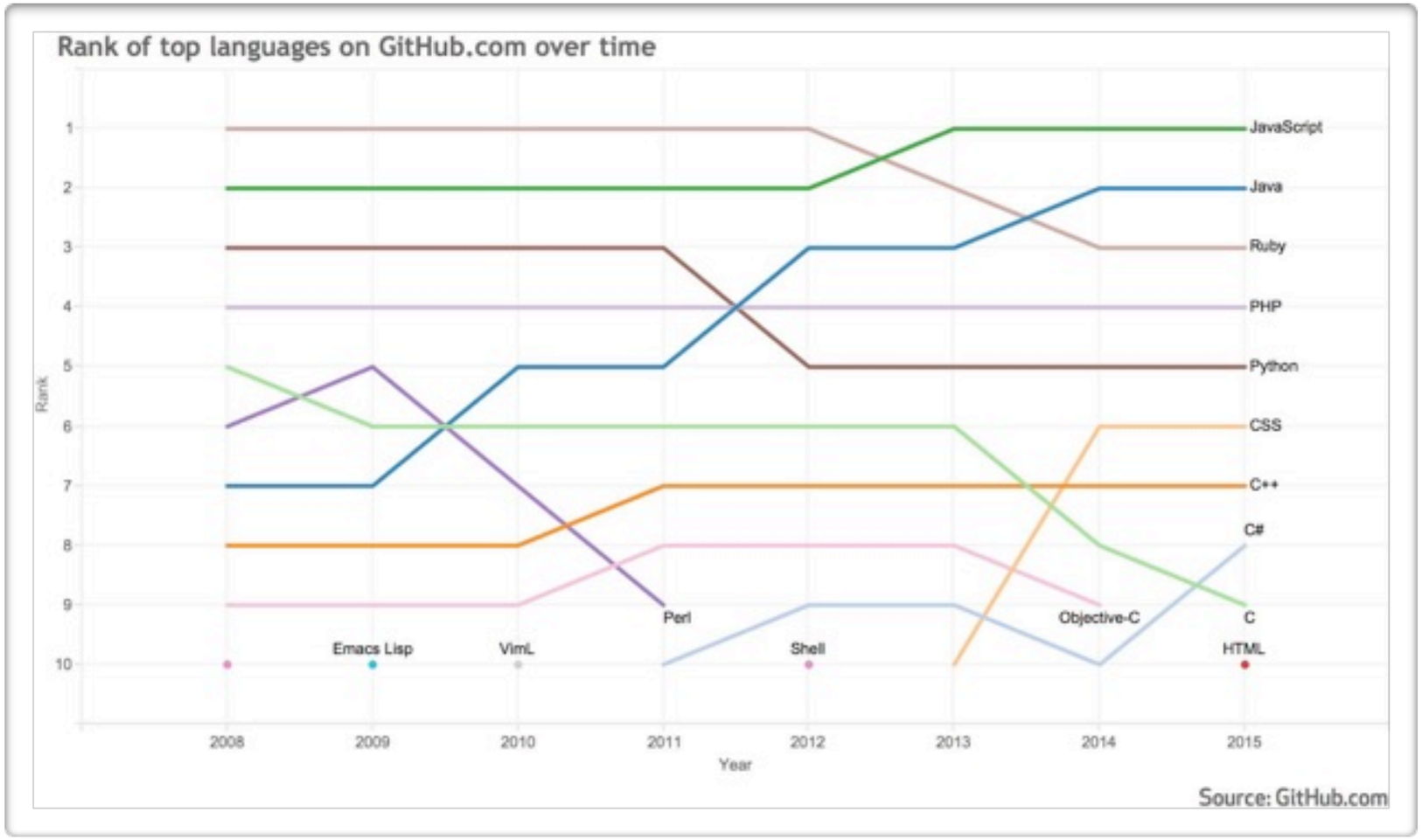
PostgreSQL and Java. Java popularity

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	2		C	7.742%	-6.86%
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%
6	7	^	Visual Basic .NET	3.174%	+0.61%
7	6	v	PHP	3.009%	+0.24%
8	8		JavaScript	2.667%	+0.33%
9	11	^	Delphi/Object Pascal	2.544%	+0.54%
10	14	^^	Swift	2.268%	+0.68%

Programming Language	2017	2012	2007
Java	1	1	1
C	2	2	2
C++	3	3	3
C#	4	4	7
Python	5	7	6
PHP	6	5	4
JavaScript	7	9	8

http://www.tiobe.com/tiobe-index/

PostgreSQL and Java. Open Source Java



http://www.tiobe.com/tiobe-index/

PostgreSQL from Java: Methods to connect

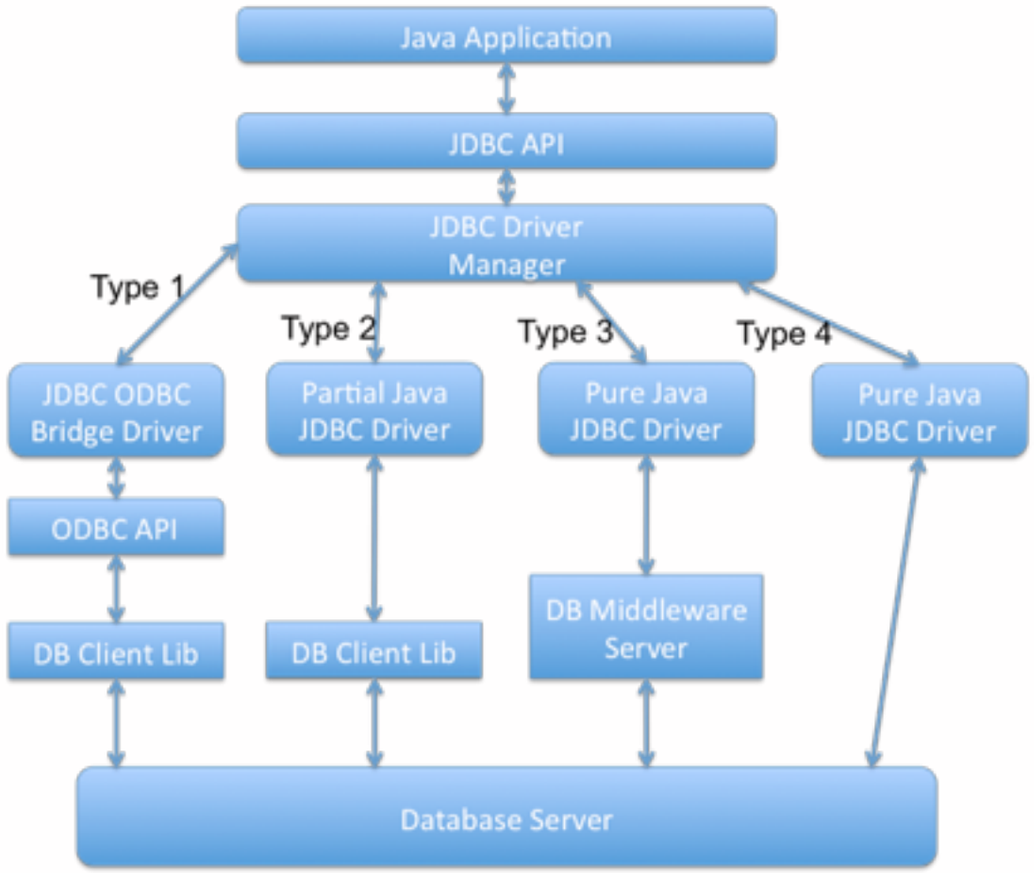
- JDBC is the de facto standard --and also a standard :)
- But there are also other ways
  - ✓ Pgjdbc-ng driver (alternative/extension of JDBC driver)
  - ✓ Phoebe (WIP)
  - ✓ Exec psql from Java through ProcessBuilder (!!)
  - ✓ Wrap PQ's C library in Java :)
  - ✓ Roll your own Java implementation of FE/BE.

PostgreSQL from Java: JDBC

- Java DataBase Connectivity. Available since Java 1.1
- Classes contained in packages java.sql and javax.sql
- Current JDBC version is 4.2 (Java 8)
- It's standard, and database-independent
- Provides a call-level API for SQL-based database access
- Allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities
- It consists of a client layer and a database-dependent layer, used to implement the database driver

PostgreSQL from Java: JDBC types

- There are four JDBC driver types:
  - ✓ **JDBC Type 1:** It is just a JDBC-ODBC bridge.
    Requires a working ODBC connection and driver to the database
  - ✓ **JDBC Type 2:** Native database interface (for Java dbs)
    or JNI-wrapped client interfaces.
    Eliminates ODBC's overhead.
  - ✓ **JDBC Type 3:** Native Java driver that talks to middleware
    (which exposes an API and interfaces to database)
  - ✓ **JDBC Type 4:** 100% Pure Java-based driver,
    Implements database communication protocol.



PostgreSQL from Java: JDBC drivers

- For PostgreSQL, there are several JDBC drivers:
  - ✓ The official, most widely used, JDBC driver:
  - ✓ jdbc.postgresql.org
  - ✓ Pgjdbc-ng (presented later)
  - ✓ EnterpriseDB's JDBC driver for Postgres Plus
  - ✓ Progress' DataDirect "Type 5" Java JDBC Driver
  - ✓ (https://www.progress.com/jdbc/postgresql)

If in doubt, just use jdbc.postgresql.org

PostgreSQL from Java: JDBC driver

- It is a Type 4 driver, natively written in Java, implementing the FE/BE protocol.
- Once compiled, it is system independent.
- Download from jdbc.postgresql.org or use it directly from Maven & friends (g: org.postgresql, a: postgresql)
- Supports trust, ident, password, md5 and crypt authentication methods
- Use UTF-8 encoding for the database
- Supports protocol versions 2 and 3, and SSL

PostgreSQL from Java: JDBC driver

- Mainly uses text mode of the protocol. Has non-standard options for COPY mode and other extensions
- Connection URL

```
jdbc:postgresql://host:port/database?options
```

- Latest version: 42.0.0 (2017-02-20)
  - ✓ Version bumped 9.4.1212 to 42.0.0 to avoid version clash with PostgreSQL version
  - ✓ Supports PostgreSQL versions below 8.2 was dropped
  - ✓ Replication protocol API was added!!!
- JDBC 4.0 (Java 6), JDBC 4.1 (Java 7), JDBC 4.2 (Java 8)

PostgreSQL from Java: JDBC driver

PostgreSQL from Java: JDBC driver options

- The connection parameters can be set via the connection URL or via the setProperty() method of Properties class.
- Relevant params:

```
ssl = true | false

loglevel =  OFF | DEBUG | TRACE

logUnclosedConnections = true | false

loginTimeout = int (seconds)

socketTimeout = int (seconds)
```

Full list of parameters: https://jdbc.postgresql.org/documentation/head/connect.html

PostgreSQL from Java: JDBC driver and SSL

- The JDBC driver, by default, validates SSL certificate CA's signature.
- Connection is refused if validation fails (psql does not behave this way)
- If your certificate server is self-signed or signed by a CA whose certificate is not in the Java keystore, you may:
  - Add the CA certificate to the Java keystore (preferred method):

    https://jdbc.postgresql.org/documentation/head/ssl-client.html
  - Or set sslfactory connection parameter to *org.postgresql.ssl.NonValidatingFactory* will turn off all SSL validation (not recommended, but simpler)

PostgreSQL from Java: JDBC driver statements

- In JDBC there are Statement and PreparedStatement objects to represent the queries.
- Except for complex dynamic queries, use PreparedStatement (more secure, no SQL injection)
- PreparedStatement objects do not result in server-side prepared statements until the query is executed a minimum number of times (5 by default)
- Adjust connection parameter PrepareThreshold=int to control when to switch to server side prepared statements

PostgreSQL from Java: JDBC driver & concurrency

- JDBC driver is thread-safe. However, a given Connection can only be used by a single thread at a time (others threads block)
- if it is needed to use it concurrently, create a new Connection per thread.
- Obviously, you may want to use connection pooling. The JDBC driver offers connection pooling, but is recommended to use an external pooler: https://jdbc.postgresql.org/documentation/head/ds-ds.html

PostgreSQL from Java: JDBC driver & app servers

- If not included by default, copy JDBC jar file to app server library include dir
- Create a JDBC connection pool within the app server. Use org.postgresql.ds.PGSimpleDataSource for the Datasource classname property (of type javax.sql.DataSource)
- At least, set the JDBC connection pool properties:

```
databaseName
portNumber
serverName
user
password
```

Optionally (recommended) export the connection pool as a JNDI resource

PostgreSQL from Java: Logical decoding and JSON

- Significant novelty appeared in pgjdbc 42
- In PostgreSQL, logical decoding is implemented by decoding the contents of the WAL.

PostgreSQL from Java: pgjdbc-ng

- T“A new JDBC driver for PostgreSQL aimed at supporting the advanced features of JDBC and Postgres”
- Project started in 2013 by Kevin Wooten to overcome some limitations of the “standard” JDBC driver
- Last version 0.7.1 (February 2017):

```
Netty 4.1.8 support

SSL support

Support for JSON/JSONB data types

OSGi support

Better Windows support

Travis CI support
```

BSD licensed http://impossibl.github.io/pgjdbc-ng/

PostgreSQL from Java: pgjdbc-ng

- Completely written from scratch. Does not support legacy versions of Java (requires Java 8+) nor PostgreSQL (requires 9.1+).
- Built via Maven
- Uses netty 4.1.8 as a network framework. Greater performance, less threading overhead, enabler for async ops
- Speaks (mostly) the binary version of the FEBE protocol
- Supports advanced and composite, custom, array and JSON/JSONB type
- DataSource / ConnectionPoolDataSource / XADataSource support

PostgreSQL from Java: pgjdbc-ng

- Download latest release from: http://impossibl.github.io/pgjdbc-ng/get.html or compile from git repo (mvn clean package)
- Use the JDBC URL:

```
jdbc:pgsql://<server>[:<port>]/<database>
```

- If configuring a DataSource / ConnectionPoolDataSource / XADataSource, classes are:

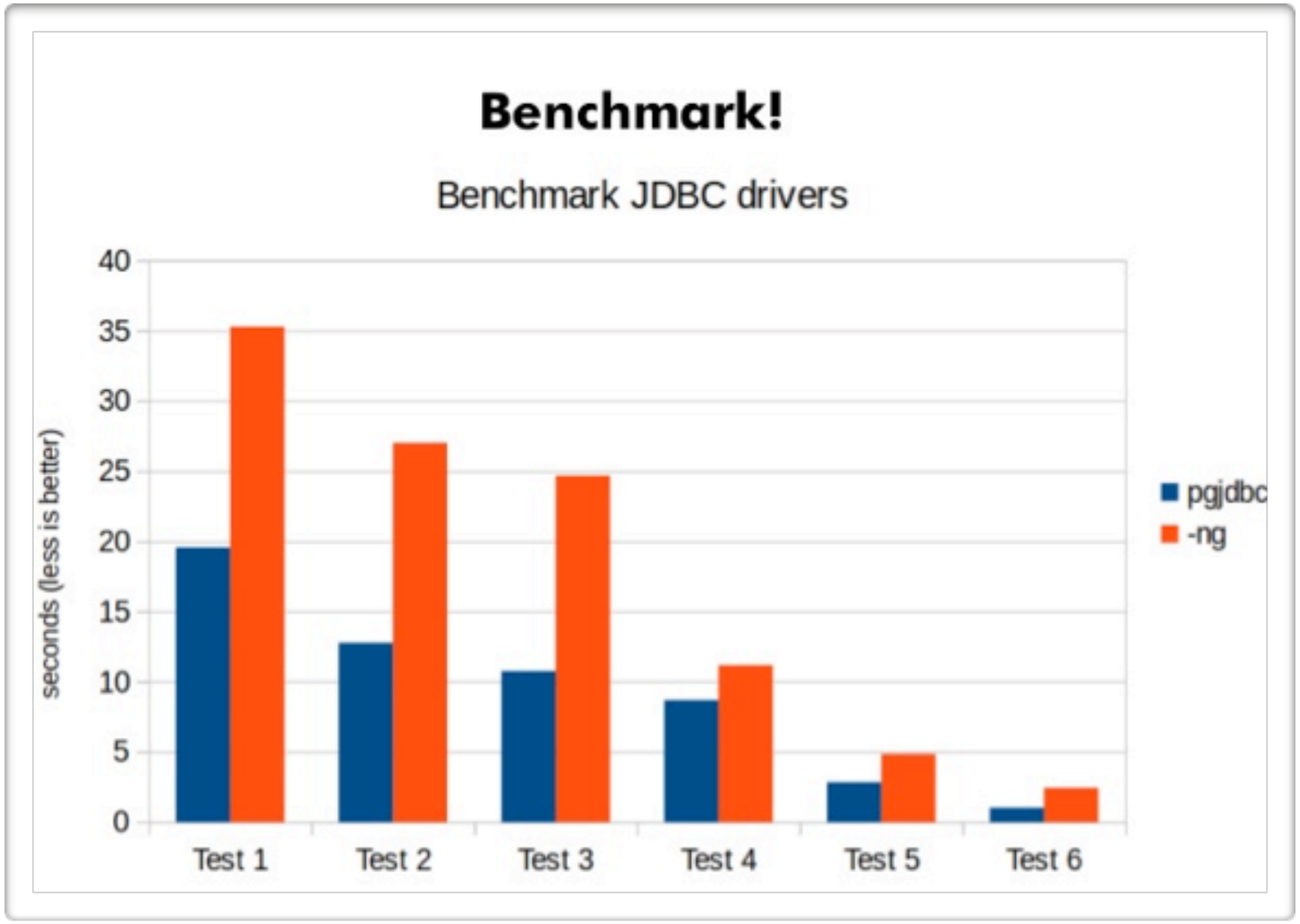
```
com.impossibl.postgres.jdbc.PGDataSource

com.impossibl.postgres.jdbc.PGConnectionPoolDataSource

com.impossibl.postgres.jdbc.xa.PGXADataSource
```

- Report success cases / bugs!

PostgreSQL from Java: pgjdbc vs pgjdbc-ng



PostgreSQL from Java: ProcessBuilder

- Hack only suitable for very specific use cases:
  - ✓psql or client program available in the same machine as the JVM
  - ✓Very simple operations (usually not involving returning a lot of information or complex types from the database), as they require manual text parsing.
- Using ProcessBuilder, an external process is executed. This process (typically psql) use the database. Commands and results are accessed via pipes connected to the process.
- it's very simple to use, no Java dependencies.

Works great!

PostgreSQL from Java: ProcessBuilder

PostgreSQL from Java: Phoebe (WIP)

- New PostgreSQL driver
- Async & Reactive by design. RxJava based
- Target clusters, not only individual servers
- Netty-based, async off-heap I/O

PostgreSQL from Java: Phoebe (WIP)

- Expected features:

  - ✓ Binary mode
  - ✓ Unix Domain Sockets
  - ✓ Logical decoding
  - ✓ Query pipelining
  - ✓ Fully asynchronous operation
  - ✓ Execute query on rw or ro nodes
  - ✓ Fluent-style API
  - ✓ Compatible with Java >= 6 Phoebe (WIP)

PostgreSQL from Java: Phoebe (WIP)

- Current API design:

```
RxPostgresClient client = RxPostgresClient
.create()
.tcpIp("::1", 5432)
.tcpIp("localhost", 5433)
.allHosts()
.init();
client.onConnectedObservable().subscribe(
c -> System.out.println(c) );
```

PostgreSQL from Java: wrap libPQ in Java

- libpq is the C application programmer's interface to PostgreSQL, allowing programs to pass queries to the PostgreSQL backend server and to receive the results https://www.postgresql.org/docs/current/static/libpq.html

- Has interfaces for C++, Perl, Python, Tcl and ECPG

- Why not wrap it in Java?
  - ✓ The only reason would be to have support for features exported by libpq but not available in JDBC driver

PostgreSQL from Java: wrap libPQ in Java

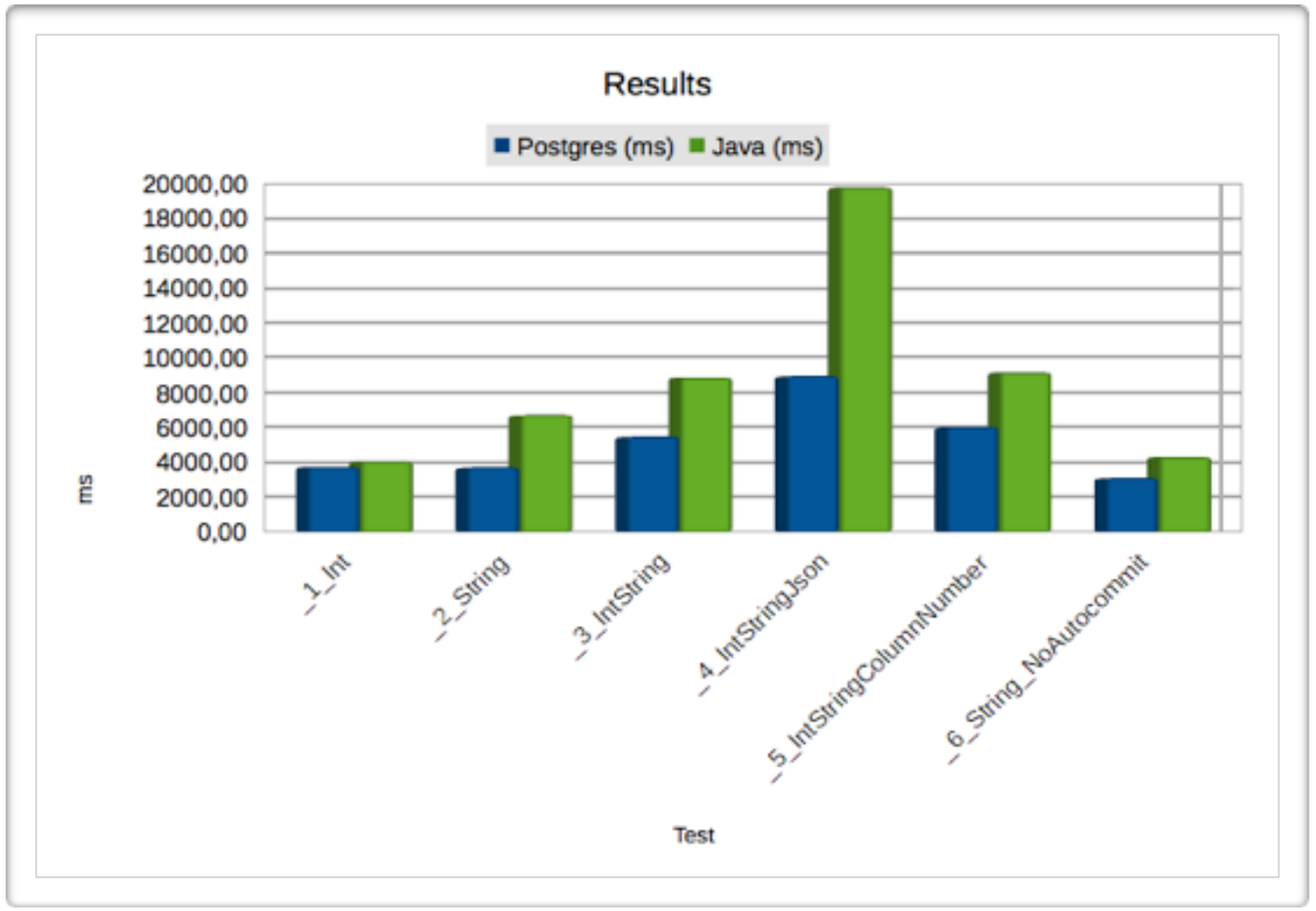
- A simple way to wrap it is to use swig: http://www.swig.org
- Tell swig to wrap libpq-fe.h
- Generate C files, compile, generate wrapper lib and add to the java.library.path
- Detailed instructions and example code: http://benfante.blogspot.com.es/2013/02/using-postgresql-in-java-without-jdbc.html

PostgreSQL from Java: own FE/BE implementation

- JDBC is verbose, complex and sometimes too low-level (ex. SQLException). Its design is quite old
- There are libraries of higher level, but ultimately depend on JDBC
- Why not a modern, non-JDBC Java API?
- Steps:

1. Study the FE/BE protocol: https://www.postgresql.org/docs/current/static/protocol.html
2. Write your own implementation
3. Publish it (preferably as open source)

PostgreSQL from Java: JDBC performance



- Total Java execution time =
  - ✓ PostgreSQL query execution
  - ✓ + Network costs (eth + tcp overhead, bw)
  - ✓ + Java driver overhead
  - ✓ + Java app processing Let's drill down the execution time

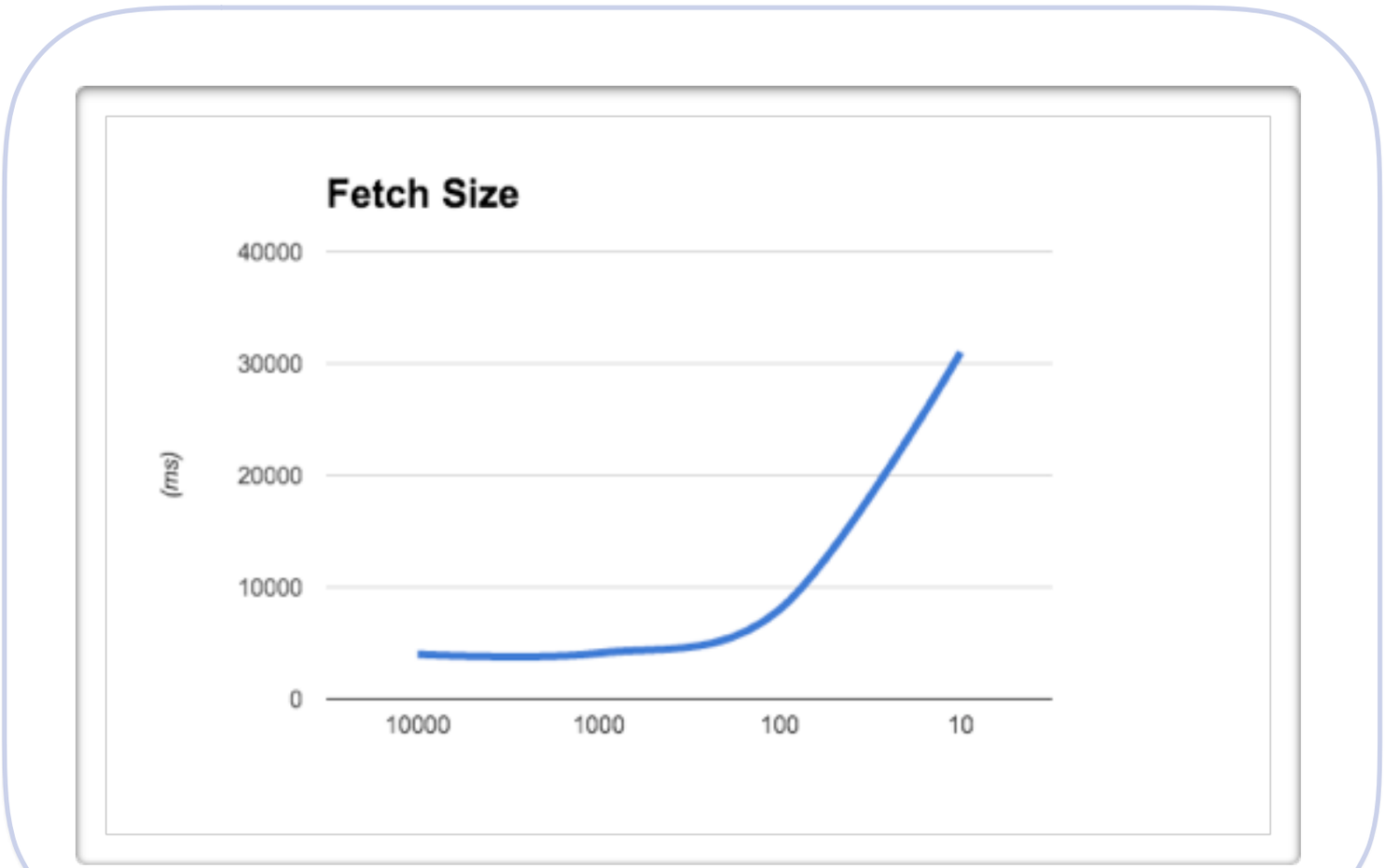
Java overhead is up 120%!!! (run on localhost to minimize network)

- ➡ Data facts: 10M records
- ➡ 1.6GB table

PostgreSQL from Java: JDBC performance

- Some things that really did improve performance
- Set **Fetch Size**: Fetch a large amount of data with different fetch sizes

```
PGProperty.DEFAULT_ROW_FETCH_SIZE.set(properties, FETCH_SIZE);
```



PostgreSQL from Java: JDBC performance

- What are the options for inserting lots of data:

  ✓ For each row Insert values (row1), (row2), … (rowN) hand rolled code

  ✓ For each row insertBatch:

    ➡ For each row Insert into perf (a,b,c) values (?,?,?)

    ➡ After N rows executeBatch

    ➡ More data inserted per statement, less statements

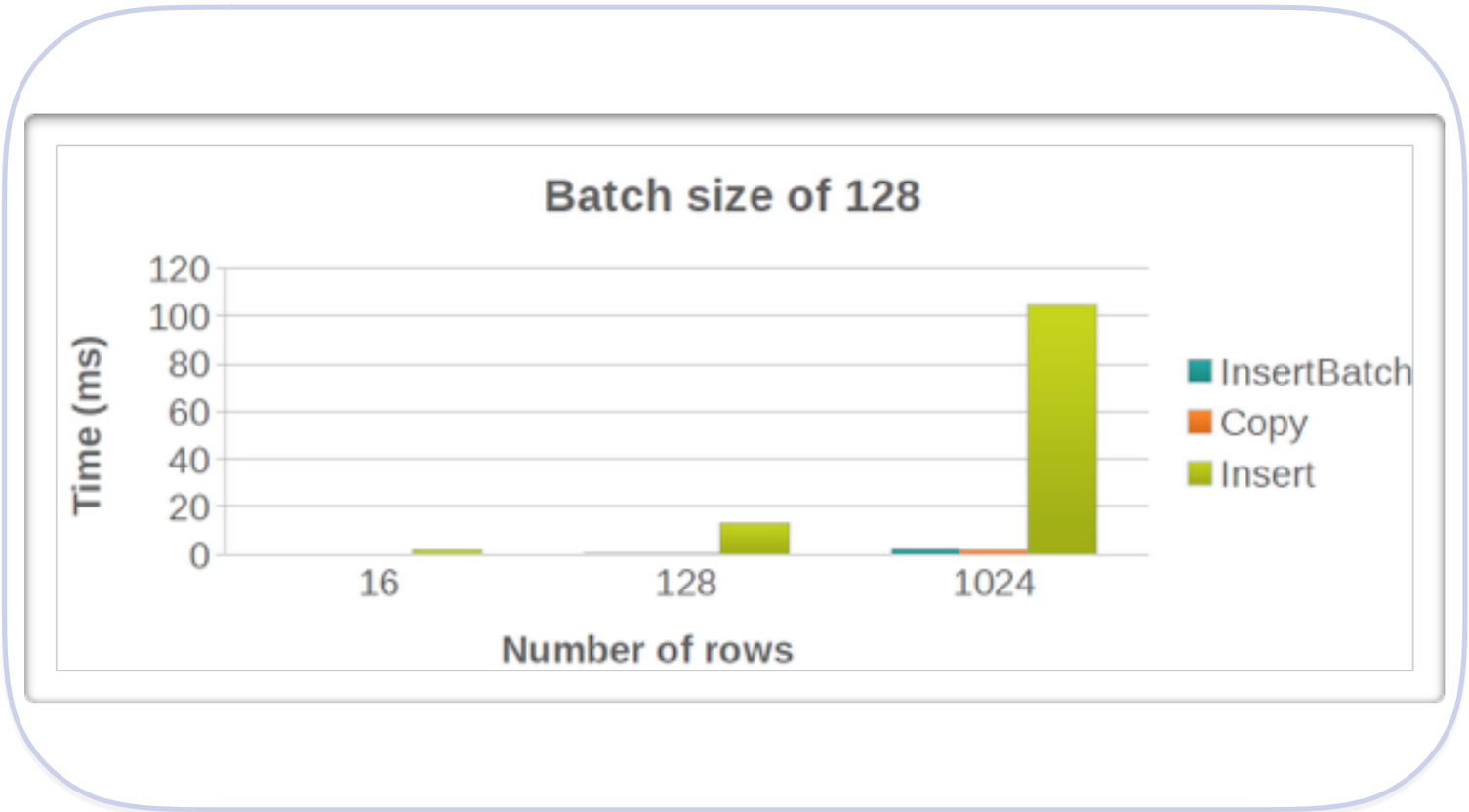
  ✓ Copy:

    ➡ Loop over the rows creating

    ➡ the input string in memory

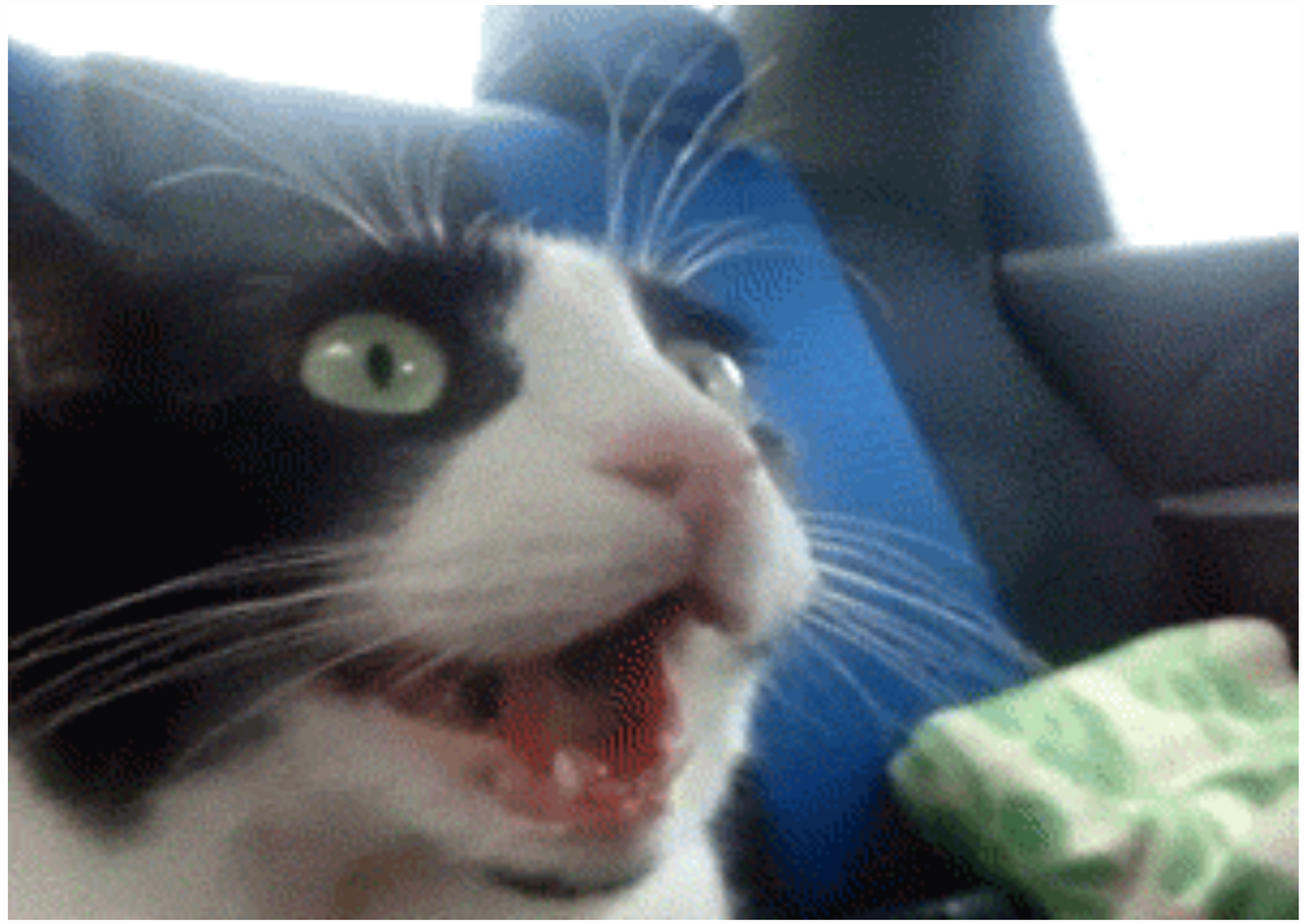
    ➡ Require using specific driver classes to execute

    ➡ Can be used for reading also



PostgreSQL from Java: ProcessBuilder

Let see a DEMO !!



PostgreSQL from Java: HikariCP + FlexyPool

- HikariCP:

✓ Optimized down to the bytecode to minimize pooling impact on JIT.

✓ Use elision logic and do not fear connection spikes.

✓ Has basic metrics (use dropwizard metrics).

✓ Nice configuration but less compared to other pool... and this does not mean it is bad.

✓ Remove the caching of statement since it is an anti-pattern. Delegates on driver that knows how to cache that stuff!

✓ Same as above for logging statements / slow queries

✓ ...and do not forget to keep you clock synchronized or you'll be taunted on twitter!!!

PostgreSQL from Java: HikariCP + FlexyPool

- FlexyPool:
  - ✓ From the guy who wrote “High-Performance Java Persistence”.
  - ✓ Powerful metrics (use dropwizard metrics too).
  - ✓ Dynamic configuration strategies that allow resizing the pool beyond the configured maximum.
  - ✓ It can be attached to many other connection pool!

PostgreSQL from Java: HikariCP + FlexyPool

Let see a DEMO !!

ORMs: JPA, Hibernate, is this what you need?

- Most certainly, not: “To ORM or not to ORM” (http://www.pgcon.org/2010/schedule/events/235.en.html)
- ORM-ing is inherently hard (“Vietnam of CS”)
- They help with CRUD-like operations, but they don't support, either natively or without breaking the abstraction:
  - ✓Custom SQL
  - ✓Database functions, triggers
  - ✓Query projections and views
- They may introduce significant inefficiencies and performance problems (eager/lazy joins)

ORMs: Return to SQL

- With the advent of NoSQL, “industry” seemed to ditch SQL.Now, it has come back (never gone, really) stronger than ever

- SQL is a very powerful, high-level, declarative language. Disqus achieved x20 improvement using recursive queries

- PostgreSQL is a very powerful database, don't reduce it to a CRUD-only data store. Unleash the power of window functions, CTEs, custom data types, aggregates, etc

- Don't create the database from Java objects, write your schema directly using database tools

Return to SQL: DRY

- DRY: don't do SQL by hand, mapping by hand. Too much boilerplate, error-prone, not focused on business logic
- Non-ORM, SQL-oriented, helper tools:
  - ✓Spring's JDBC Template
  - ✓Apache DbUtils
  - ✓MyBatis
  - ✓jOOQ
  - ✓…
- Write DAOs or similar abstractions to wrap your SQL and your database access methods and drivers

MyBatis: a great mapper for PostgreSQL

- MyBatis (formerly iBatis) is a mapper tool: you write the SQL code, MyBatis maps queries to POJOs
- You basically need a POJO (which basically becomes a DTO) per query, plus a method in an interface
- Alternatively, queries may be mapped to Maps
- Queries are written in SQL in XML files, with support for:
  - ✓Query parameters (Java POJOs)
  - ✓Dynamic SQL (if-then, loops, etc)
  - ✓Arbitrary classification of queries into separate files

MyBatis: a great mapper for PostgreSQL

Let see a DEMO !!

jOOQ: Get back in control of your SQL

- jOOQ lets you write SQL in a programmatic way
- Unlike other SQL "APIs", it does not restrict you to a very basic subset of SQL (least common denominator of all databases, intersection the "easy" part of SQL)
- It even simulates some features not present in some databases by crafting more complex SQL behind the scenes
- Uses a fluent API (nice, short to write) which supports almost all the standards SQL features

jOOQ: Get back in control of your SQL

- There's basic syntax validation support in the API, to check for invalid queries
- It shines when you use jOOQ's code generator: with it, all the queries are strongly-typed and completely syntax validated. The code generator supports every database object you could use, including data types, procedures...
- It gives you database independence, hiding in the API implementation the details of SQL dialects
- Even implements as part of the API some “clever SQL tricks” like the seek (aka pagination done right)
- You can use it with Java 8’s lambdas

jOOQ: Get back in control of your SQL

Let see a DEMO !!

Java inside PostgreSQL: PL/Java

- PL/Java
  - ✓Latest version 1.5
  - ✓Coming back! First release since 2011
  - ✓Do not support postgres below 8.2
  - ✓Modernized, more active community
  - ✓Works with 9.4 & 9.5, Java 6-8 :)

PostgreSQL from Java: PL/Java

Let see a DEMO !!

Java inside PostgreSQL: Introducing pgj

- Run Java code **inside** PostgreSQL
- PL/Java is cool but not enough: it has to be like a server, run independently of the user
- Background workers provide the base
- Wrap SPI calls with JNI and profit!
- Join the pgj project and run Java code inside PostgreSQL with minimal overhead

Let's Talk!

Using PostgreSQL with Java

www.8kdata.com

info@8kdata.com