Java In Oracle RDBMS

Christine Swanson

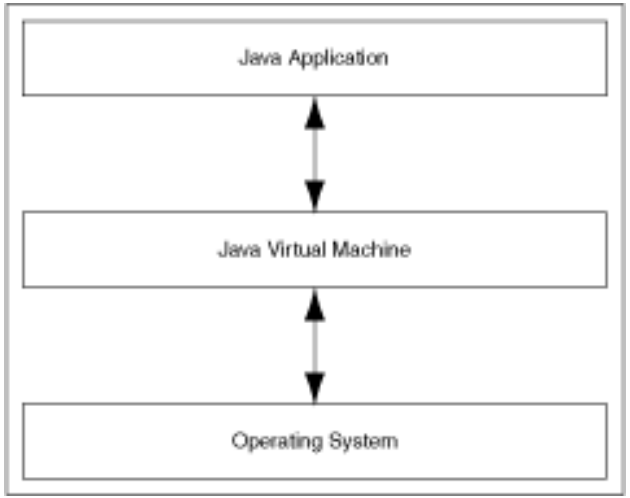
Paychex

Who Am I

- Senior Software Developer at Paychex
- Been there 9+ years
- Oracle OCA (PL/SQL) and OCE (SQL)
- Using Java since 2003

What is Java?

- World’s most popular programming language
- High level language with syntax similar to C
- Object oriented
- Runs in a JVM – Java Virtual Machine
  - “write once run anywhere”



Why are we talking about Java

- A JVM has been inside Oracle RDBMS since version 8i (~1999)
- Oracle bought WebLogic Server in 2008 via acquisition of BEA
  - Major Java EE application server
- Oracle bought Java in 2010 via acquisition of Sun Microsystems
- Chances are, if you are running something Oracle, there's Java somewhere in your stack

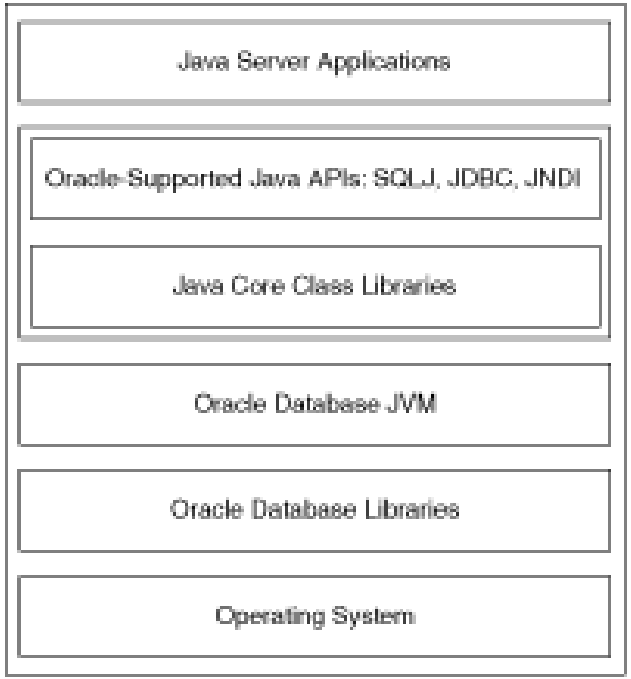
https://docs.oracle.com/cd/E18283_01/java.112/e10588/toc.htm

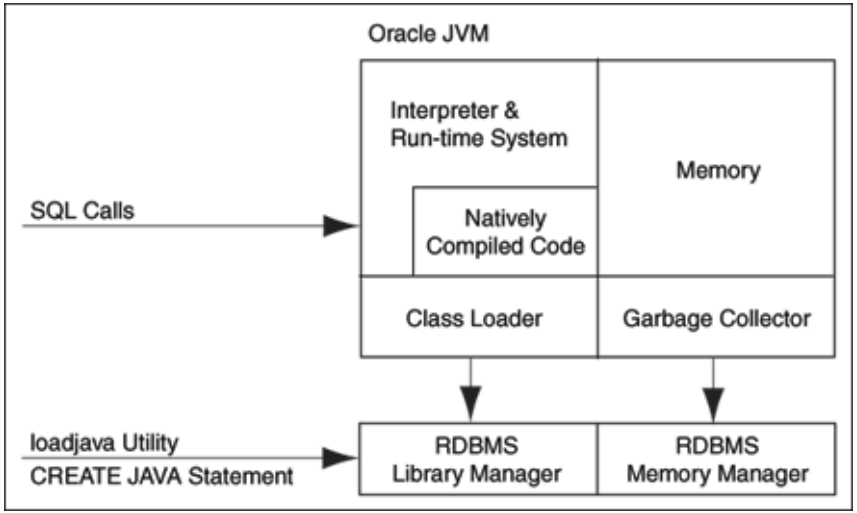
How does Java run inside the DB?

- The DB includes a complete Java2-compliant environment for running Java applications – the OJVM
- Runs in the same process space and address space as the database kernel by sharing its memory heaps and directly accessing its relational data
- Tightly integrated with the shared memory architecture of the database
  - Java pool (shared definitions) live in SGA
  - Session based things depend on server mode
    - Dedicated -> PGA
    - Shared -> Java pool

https://docs.oracle.com/cd/E18283_01/java.112/e10588/chone.htm#insertedID6

Oracle JVM architecture





https://docs.oracle.com/cd/E18283_01/java.112/e10588/chone.htm#BABFJCGE

What Java version do I have?

- Unfortunately you don’t get to pick the version of your OJVM
  - If you are on 12c, you have 2 choices

- SELECT ***dbms_java.get_ojvm_property***(PROPSTRING=>'java.version') FROM dual

- Compatibility:

RDBMS	JDK
8.1.7.x	1.2
9.0.1.x	1.2
9.2.x	1.3
10.1.x	1.4
10.2.x	1.4
11.1.x	1.5
11.2.0.1 to 11.2.0.3	1.5
11.2.0.4	1.6
12.1.x	1.6 or 1.7

Code Examples

How do I load a java program into the DB?

- Create “hello world” style class in your favorite IDE (Greeter.java):

```
public class Greeter {
   public static String sayHi(String name) {
      return "Hello " + name + "!";
   }
}
```

- Compile with javac to generate Greeter.class file:

```
C:\Users\cswanso9\Documents\java_example>%JAVA_HOME%\bin\javac Greeter.java
```

  - Note: Make sure you are compiling for the right JVM version

- Load .class file into database via “loadjava”:

```
C:\Users\cswanso9\Documents\java_example>loadjava -u ops$cswanso9@benx01 Greeter.class

Password:
**********

C:\Users\cswanso9\Documents\java_example>
```

On the DB…

- Create a stored function to call the “sayHi” method on the “Greeter” class

```
create or replace function say_hello(inName varchar2) return varchar2
as language java
name 'Greeter.sayHi(java.lang.String) return java.lang.String';
```

- Call stored function like you would any other PL/SQL:

```
SQL> select say_hello('Chris') from dual;

SAY_HELLO('CHRIS')
--------------------------------------------------------------------------------
Hello Chris!

SQL>
```

What can I do with Java?

- Call Java code from your PL/SQL
- Use internal JDBC driver
  - Supports standard JDBC things like queries, DML, transactions, PL/SQL interactions
- SQLJ lets you embed SQL in Java
- Make service calls out of the database
  - This doesn't necessarily require Java depending on the type of call

https://docs.oracle.com/cd/E18283_01/java.112/e10588/chone.htm#BABHHGGD
https://docs.oracle.com/cd/E18283_01/java.112/e10588/chthree.htm#CACCIEAH

JDBC

```
// Assume you already have a JDBC Connection object conn
// Define Java variables
String name;
int id=37115;
float salary=20000;

// Set up JDBC prepared statement.
PreparedStatement pstmt = conn.prepareStatement
("SELECT ename FROM emp WHERE empno=? AND sal>?");
pstmt.setInt(1, id);
pstmt.setFloat(2, salary);

// Execute query; retrieve name and assign it to Java variable.
ResultSet rs = pstmt.executeQuery();
while (rs.next())
{
  name=rs.getString(1);
  System.out.println("Name is: " + name);
}

// Close result set and statement objects.
rs.close()
pstmt.close();
```

https://docs.oracle.com/cd/E18283_01/java.112/e10588/chthree.htm#CACHFBBF

SQLJ

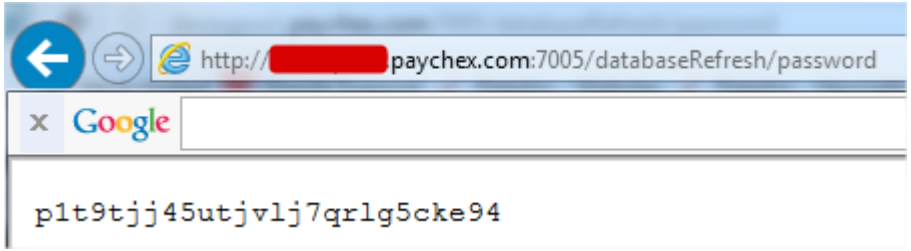
```
import java.sql.*;
import sqlj.runtime.ref.DefaultContext;
import oracle.sqlj.runtime.Oracle;

#sql ITERATOR MyIter (String ename, int empno, float sal);

public class MyExample
{
  public static void main (String args[]) throws SQLException
  {
    Oracle.connect("jdbc:oracle:thin:@oow11:5521:sol2", "scott", "tiger");
    #sql { INSERT INTO emp (ename, empno, sal) VALUES ('SALMAN', 32, 20000) };
    MyIter iter;
    #sql iter={ SELECT ename, empno, sal FROM emp };
    while (iter.next())
    {
      System.out.println(iter.ename()+" "+iter.empno()+" "+iter.sal());
    }
  }
}
```

Calling out of the database

Bridge SQL to J2EE

Utl_http

- Allows for making HTTP calls out of the database via PL/SQL
- Can call internet if ACLs/proxies/firewalls allow
- Useful for calling REST services
  - Example service generates random string



  - Note: this is a contrived example.

http://docs.oracle.com/cd/E11882_01/appdev.112/e40758/u_http.htm#ARPLS070

Set up

```
begin
   DBMS_NETWORK_ACL_ADMIN.CREATE_ACL(acl => '/sys/acls/devtools.xml',  description => 'tools ACL',
                                     principal => 'OPS$CSWANSO9', is_grant => true, privilege   => 'connect');
   DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE(acl => '/sys/acls/devtools.xml', principal => 'OPS$CSWANSO9',
                                        is_grant  => true, privilege => 'resolve');
   DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL(acl => '/sys/acls/devtools.xml', host => '[redacted].paychex.com');
end;
/
```

```
create or replace function call_rest_service return varchar2
is
   request     UTL_HTTP.REQ;
   response    UTL_HTTP.RESP;
   line        varchar2(4000);
begin
   request := UTL_HTTP.BEGIN_REQUEST('http://[redacted].paychex.com:7005/databaseRefresh/password');
   response := UTL_HTTP.GET_RESPONSE(request);
   UTL_HTTP.READ_LINE(response, line, true); -- normally you'd loop here but we only get one line from this service
   UTL_HTTP.END_RESPONSE(response);
   return line;
end;
```

Call stored function

- Weblogic server returns randomized string to DB session via JAX-RS service

UTL_DBWS

- Used for calling JAX-WS web services that communicate via XML
  - e.g. SOAP

- Doesn’t come out of the box but here are the installation instructions: https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=552196822917850&parent=SrDetailText&sourceId=3-8906396591&id=841183.1&_afrWindowMode=0&_adf.ctrl-state=hw7j2yf2i_230

Example service

- This Oracle sample is deployed to a Weblogic server on my desktop
  - It accepts a parameter “message” and basically echoes it

```java
package examples.webservices.hello_world;

// Import the @WebService annotation
import javax.jws.WebService;

@WebService(name = "HelloWorldPortType", serviceName = "HelloWorldService")
/**
 * This JWS file forms the basis of simple Java-class implemented WebLogic Web
 * Service with a single operation: sayHelloWorld
 */
public class HelloWorldImpl {
    // By default, all public methods are exposed as Web Services operation
    public String sayHelloWorld(String message) {
        try {
            System.out.println("sayHelloWorld:" + message);
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        return "Message: " + message;
    }
}
```

https://docs.oracle.com/middleware/1213/wls/WSGET/jax-ws-examples.htm#WSGET112

Create stored function to call service

```
create or replace function call_soap_service(message in varchar2) return varchar2
IS
   service_name   varchar2(100) := 'HelloWorldService';
   response       sys.xmltype;
   end_point      varchar2(100) := 'http://[redacted].paychex.com:7001/soapHelloWorld/HelloWorldService';
   request        xmltype := xmltype('<ns1:sayHelloWorld xmlns:ns1="http://hello_world.webservices.examples/"><arg0>'
                          || message || '</arg0></ns1:sayHelloWorld>');
   service        sys.utl_dbws.service;
   service_call   sys.utl_dbws.call;
BEGIN
      service := sys.utl_dbws.create_service(sys.utl_dbws.to_qname(null, service_name));
      service_call := sys.utl_dbws.create_call(service);
      sys.utl_dbws.set_target_endpoint_address(service_call, end_point);

      sys.utl_dbws.set_property(service_call, 'SOAPACTION_USE', 'true');
      sys.utl_dbws.set_property(service_call, 'SOAPACTION_URI', end_point);

      response := sys.utl_dbws.invoke(service_call, request);
      dbms_output.put_line(response.getStringVal());

      sys.utl_dbws.release_call(service_call);
      sys.utl_dbws.release_service(service);

      return response.extract('//return/text()').getStringVal();
END call_soap_service;
```

Call stored function from sqlplus

- We are sending our parameter to the service and it is building a message with it

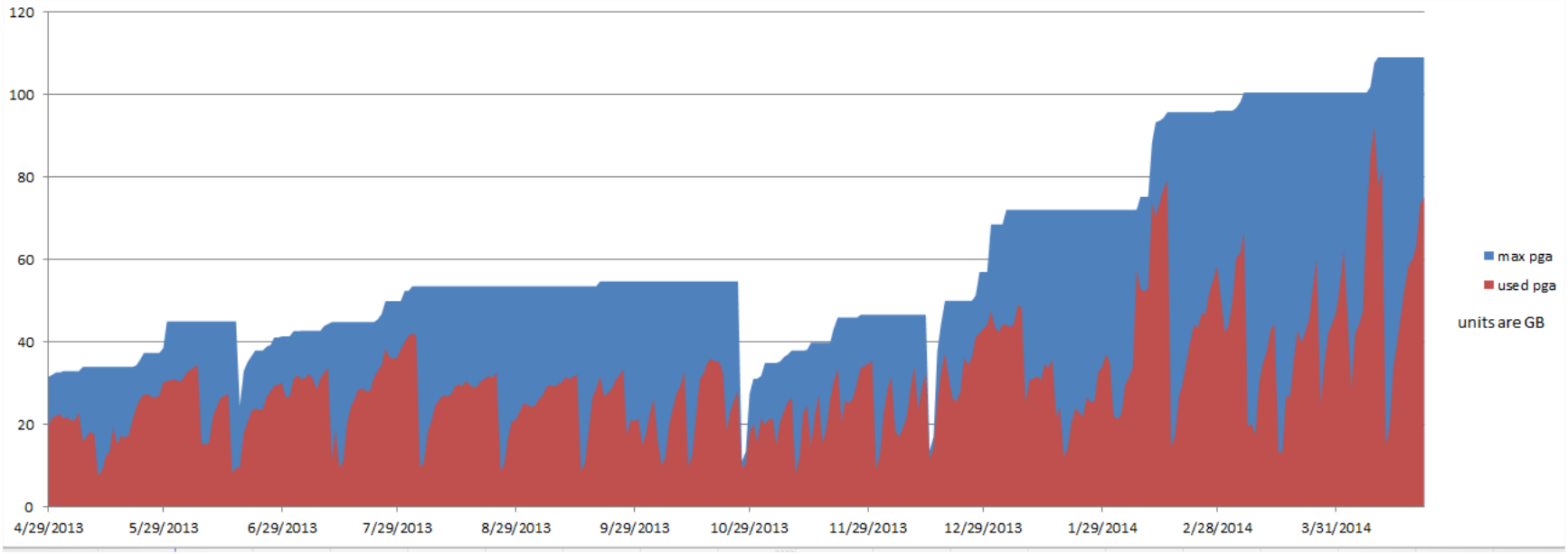
```
SQL> select call_soap_service('hi!!!') from dual;

CALL_SOAP_SERVICE('HI!!!')
--------------------------------------------------
Message: hi!!!

SQL> select call_soap_service('bye!!!') from dual;

CALL_SOAP_SERVICE('BYE!!!')
--------------------------------------------------
Message: bye!!!

SQL>
```

Some Pitfalls…

UTL_DBWS Bug (and solution)

- One day we noticed that our PGA usage was going crazy
- Bad enough to cause swapping storm on DB instance and node failure
- Failures becoming more frequent
- SR 3-8906396591

Memory usage (GB)



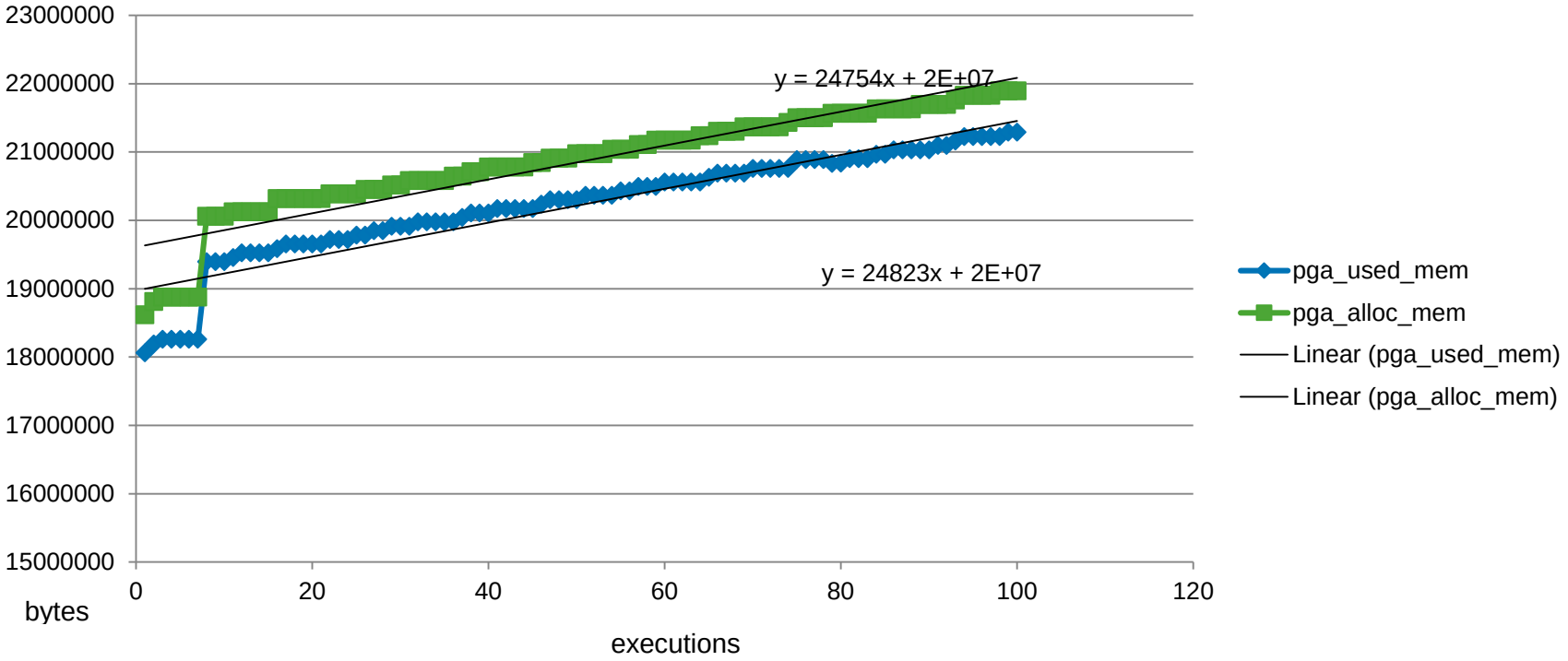
We isolated it…

- To apps that were making service calls through the database
- The PGA growth was mostly showing in the java heap

```
select st.value
  from v$session x, v$sesstat st, v$statname n
 where 1 = 1
   and x.sid = :sid
   and x.sid = st.sid
   and n.statistic# = st.statistic#
   and n.name = 'java session heap used size';
```

- Apps would run queries, and those queries would call out to services to get data from other departments within the company

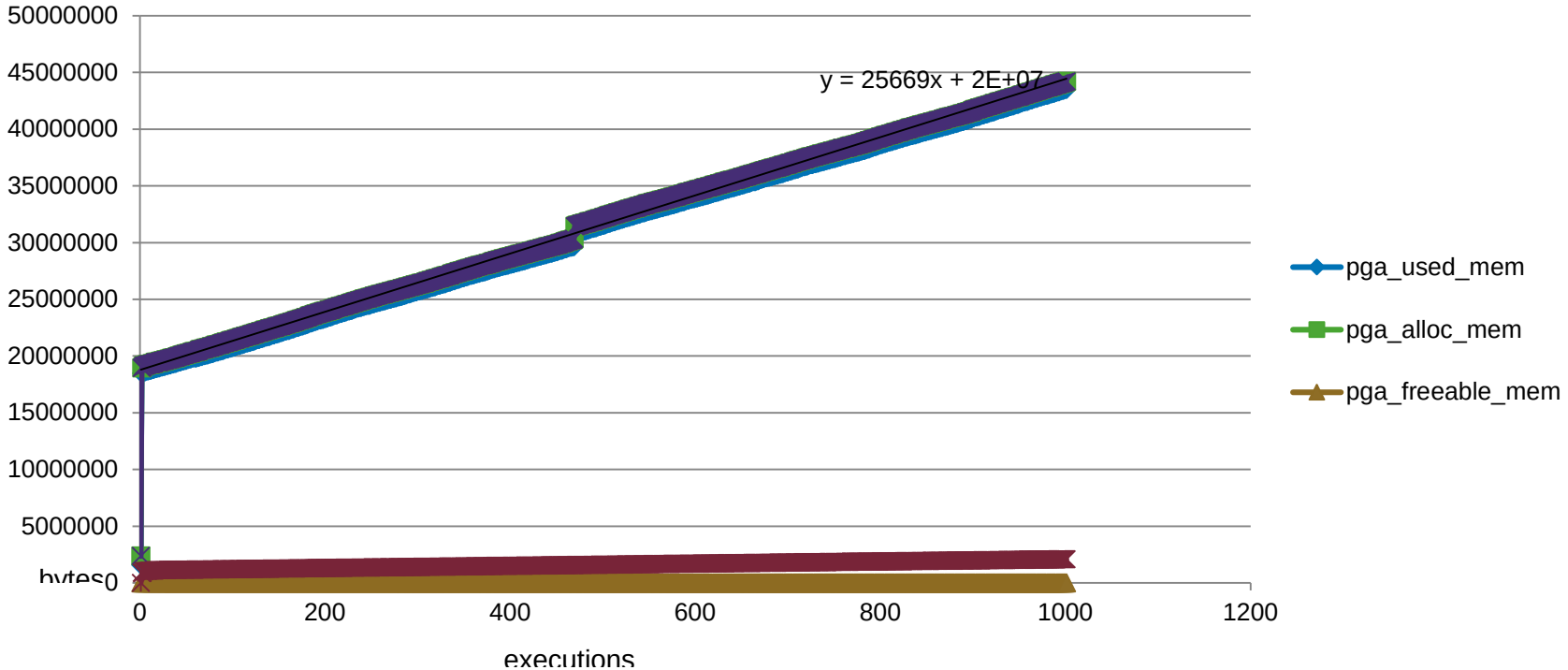
Reproducible on test



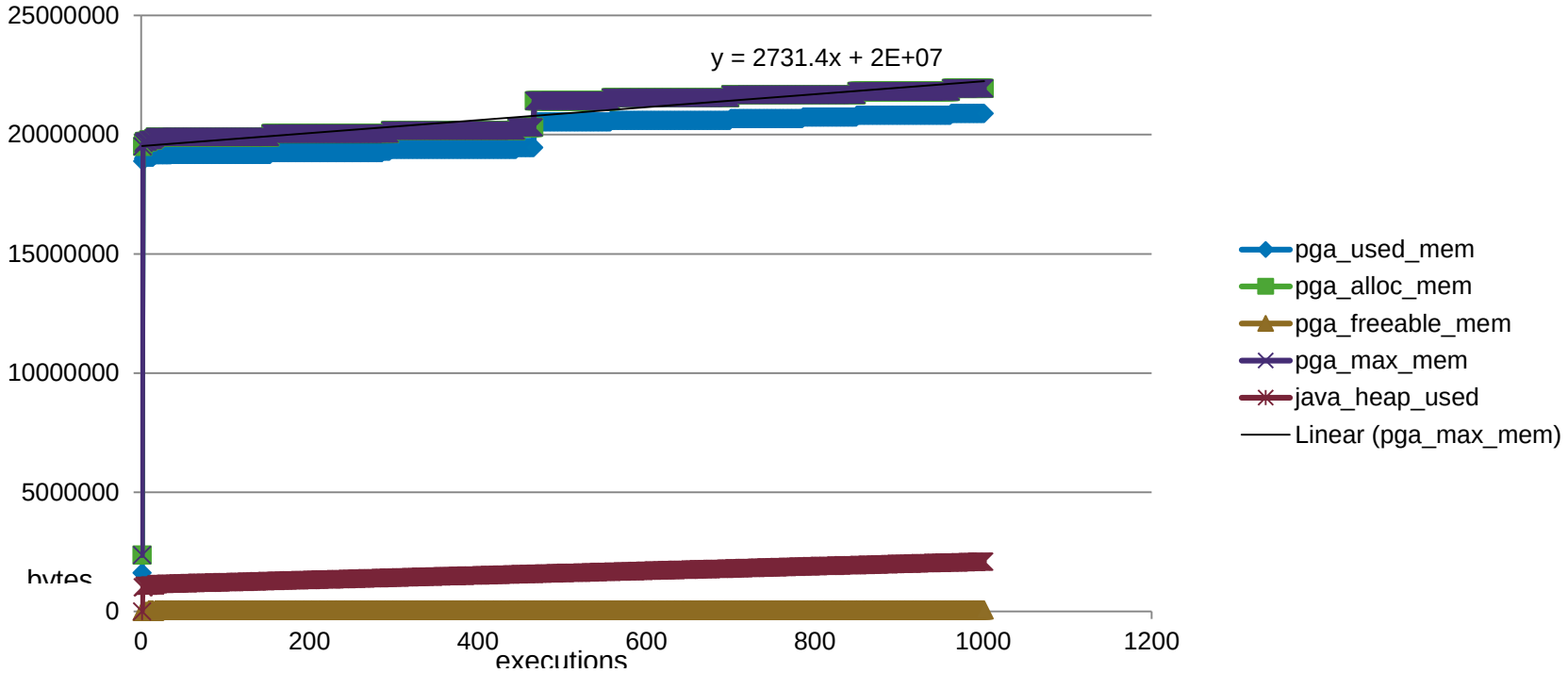
Turns out…

- There is a memory leak in utl_dbws
- If you want to use it, make sure you get the patched version
- Made us question our architecture decisions

Before patch



After patch


y = 2731.4x + 2E+07
25000000
20000000
15000000
10000000
5000000
bytes
0
0
200
400
600
800
1000
1200
executions
pga_used_mem
pga_alloc_mem
pga_freeable_mem
pga_max_mem
java_heap_used
Linear (pga_max_mem)

Just in time compilation - JIT (since 11.1)

- Intepreted
  - For each bytecode instruction, look up the hardware instruction for the current platform, and run it
  - Flexible, but not efficient

- Jit (default)
  - At run time, take all of the bytecode at once and directly into machine code
  - More overhead on load but faster repeated executions (of native machine code)
  - Allows for adaptive optimization
    - Keeps track of heavily used methods and further compiles/optimizes

- Param = java_jit_enabled

https://docs.oracle.com/cd/E18283_01/java.112/e10588/chone.htm#BABCIHGA

Why JIT…

- After recent round of OJVM patching with Oracle Support's help, all of a sudden Java calls in the DB were taking 4x as long
- Application performance was noticeably affected
- Further investigation revealed that JIT had been disabled during troubleshooting
- DBA re-enabled JIT and performance improved by 4x (back to normal)
- For heavily used Java methods, you probably want JIT

Architecture Decisions

Oracle’s selling points for Java in the DB

- Create component-based, network-centric applications that can be easily updated as business needs change
- Move applications and data stores off the desktop and onto intelligent networks and network-centric servers
- Access applications and data stores from any client device
- Performant and secure
- Bridge SQL to J2EE

https://docs.oracle.com/cd/E18283_01/java.112/e10588/chone.htm#BABFICAE

Why you might not want to

- Now your database can talk to everything! But should it?
- You database maybe scalable but it might be the most expensive thing in your stack to scale
  - Do you really want more workload running in there?
- One more thing to maintain/patch/worry about
  - More risk?
- The OJVM is usually a bit behind current Java version
  - No choice on versions below 12c
  - On 12c, can use 1.6 and/or 1.7
  - Java 8 SE is current

Where is Oracle going with Java?

- Some have speculated that Oracle's commitment to Java has been lagging lately.

- Here's what Oracle said recently:
  - *"Oracle is committed to Java and has a very well defined proposal for the next version of the Java EE specification—Java EE 8—that will support developers are as they seek to build new applications that are designed using micro-services on large-scale distributed computing and container-based environments on the Cloud. Oracle is working closely with key partners in the Java community to finalize the proposal and will share the full details with the broader Java community at JavaOne in September."*

- Note the mention of micro-services.
  - Current architecture trends are tending towards breaking up applications into small, modular pieces using varieties of technologies
  - Highly questionable that one would want to also use database as an application server
    - Monolithic design
  - Then again, maybe you have a micro service you want to call?

“Free advice”

- You can do some cool things with Java in the DB

- I see use cases mostly for small or internal apps or where there isn’t other infrastructure
  - Internal tools for things like continuous delivery, monitoring, maintenance etc.
  - There is no external application layer, you just have everything in the database
  - Small shop with limited servers/resources

- Have seen it used to patch together far flung legacy enterprise architecture, but I don’t recommend that as long term solution
  - We’ve seen situations where apps are calling into the database, then the database turns around and calls out to more services, which come right back to the DB for more data.
  - Time for some refactoring

- Would not recommend for large scale, mission critical applications especially if you have the resources to run applications somewhere else
  - You can do a lot in pure PL/SQL if you really want business logic in the DB

The End

Questions?