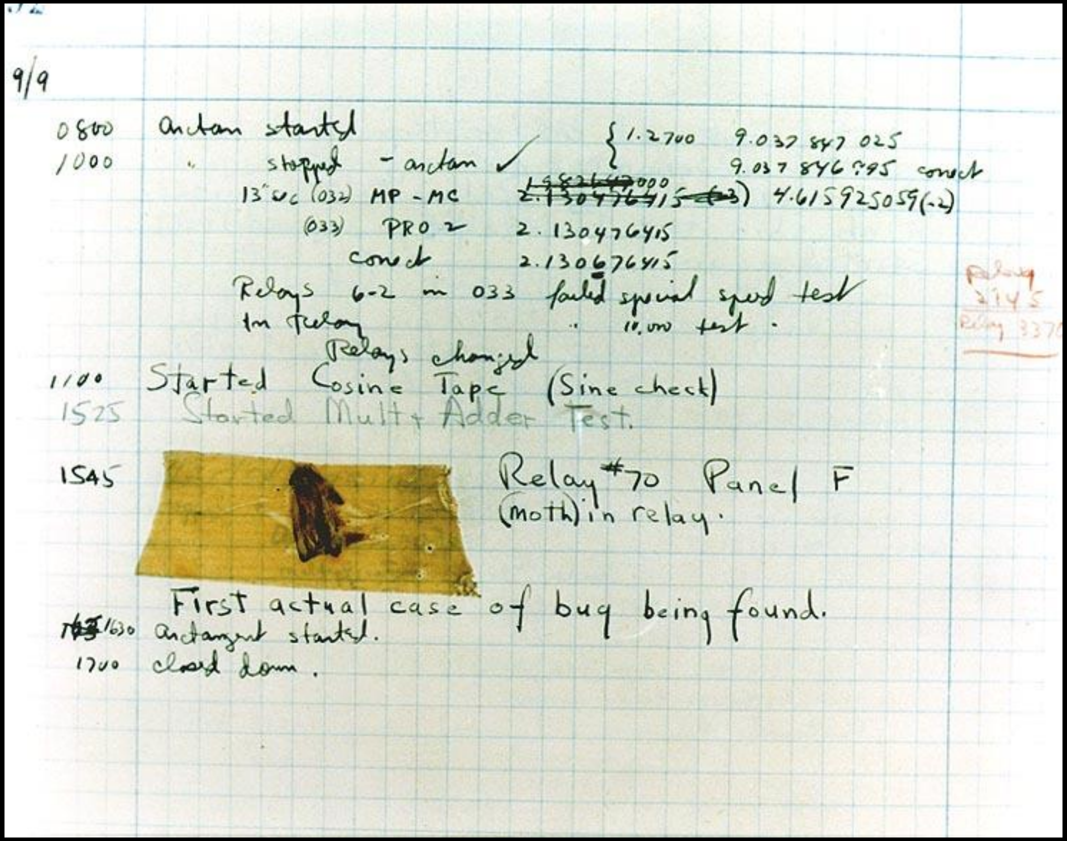
追求JDBC on Oracle最佳性能？如何才好？

Maclean Liu

古希腊的Delphi(世界中心)，屹立着Parnassus Mount(诗檀山)，山上有一座阿波罗神庙，庙中住着女祭司(Oracle)

兴一利
不如
除一害



Started Cosine Tape (Sine check)
Started Mult+ Adder Test.
Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.

Jdbc性能案例1：问题

```
for (i = 0; i < 1000000; i++) {
  conn = ds.getConnection(userid, password);

  pstmt =
    conn.prepareStatement("SELECT name FROM employees WHERE id = " +
i);
  rset = pstmt.executeQuery();

  ......

  conn.close();
}
```

循环内获得connection 并解析执行，connection、非缓存的解析在Oracle中都很昂贵

Jdbc性能案例1：结果

- 响应时间: 150ms
- 吞吐量: 300tps

- CPU方面Sys和User都很高

Jdbc1性能案例1：对策

```
conn = ds.getConnection(userid, password);

for (i = 0; i < 1000000; i++) {
  pstmt =
    conn.prepareStatement("SELECT name FROM employees WHERE id = " +
i);
  rset = pstmt.executeQuery();

  ......
}

conn.close();
```

只建立必要的connection到数据库

Jdbc1性能案例1：改良后

- 响应时间: 20ms
- 吞吐量: 3,000tps

- latch: shared pool

此时的瓶颈凸显在解析(parse)并发争用上

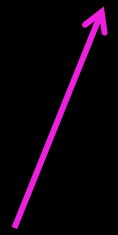
Jdbc性能案例1：原因

```
conn = ds.getConnection(userid, password);

 for (i = 0; i < 1000000; i++) { {
   pstmt =
conn.prepareStatement("SELECT name FROM employees WHERE id = " + i);
   rset = pstmt.executeQuery();

   ......
}

conn.close();
```

每次执行都使用拼凑的SQL语句，未启用绑定变量，默认情况下每次均硬解析hard parse

Jdbc性能案例1：进一步对策

```
conn = ds.getConnection(userid, password);

 for (i = 0; i < 1000000; i++) { {
   pstmt =
     conn.prepareStatement("SELECT name FROM employees WHERE id = ?");
   pstmt.setInt(1, i);
   rset = pstmt.executeQuery();

   ......
}

conn.close();
```

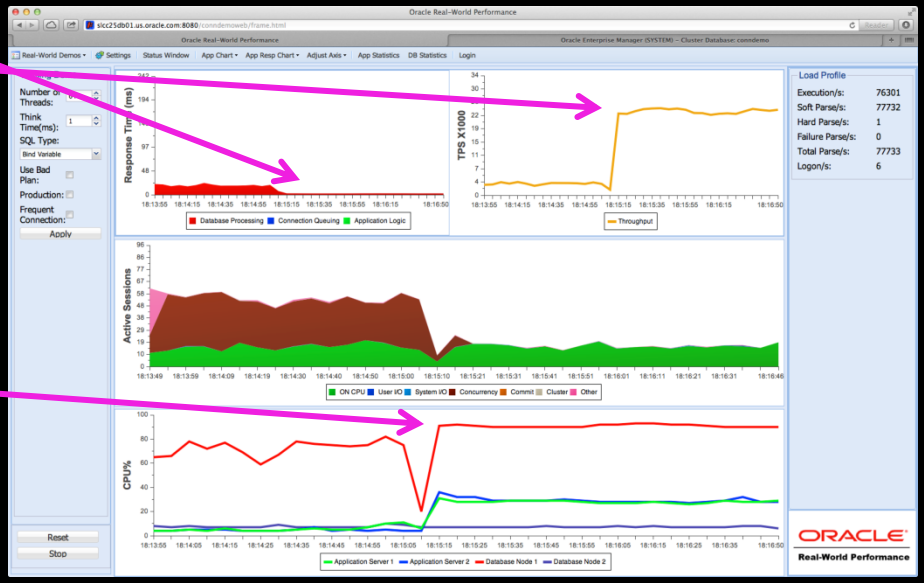


使用绑定变量，避免每次执行都硬解析

Jdbc1性能案例1：再改良后

- 响应时间: 1ms
- 吞吐量: 25,000tps

- 并发争用减少
- CPU被充分使用



Jdbc1性能案例1：正确使用PrepareStatement

- PrepareStatement
  - SQL语句解析
- Bind
  - 变量绑定
- Execute
  - SQL语句执行

×

```
For  {
     PrepareStatement
     Bind
     Execute
}
```

○

```
PrepareStatement
For {
     Bind
     Execute
}
```

Jdbc1性能案例1：进一步对策

```
conn = ds.getConnection(userid, password);
pstmt =
  conn.prepareStatement("SELECT name FROM employees WHERE id = ?");

 for (i = 0; i < 1000000; i++) { {
   pstmt.setInt(1, i);
   rset = pstmt.executeQuery();

   ......
}

conn.close();
```

预解析，只解析一次

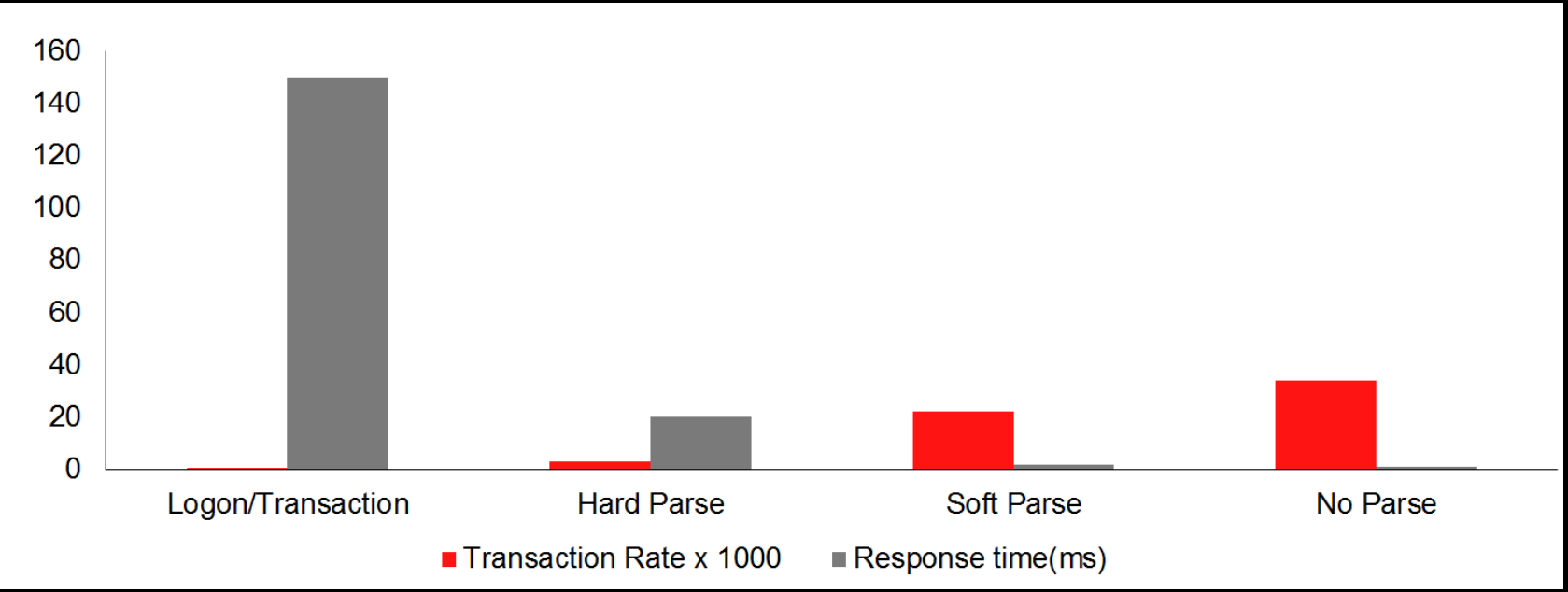
For prepareStatement , http://www.oracle.com/technetwork/testcontent/jdbc-ch5-131209.pdf

Jdbc1性能案例1：再再改良后

- 响应时间: 1ms
- 吞吐量: 34,000tps

- 吞吐量进一步提高

Jdbc1性能案例1：再改良后



Jdbc1性能案例2：永远不够的open_cursor

- ORA-1000报错” maximum open cursors exceeded”达到session 最大游标数
- 故障发生业务停滞
- 大量SQL*Net break/reset to client等待事件
- OPEN_CURSOR参数已经很大了，都到10000了，开发人员要求DBA进一步加大该参数到30000

Jdbc1性能案例2：原因

```
try {
  ......
  rset = pstmt.executeQuery();
  ......
  rset.close();
  pstmt.close();
} catch (SQLException e) {
  e.printStackTrace();
}
```

代码里是写了关闭游标，但存在还没执行就跑到异常处理里去的可能。

Jdbc1性能案例2：对策

```
try {
  ......
  rset = pstmt.executeQuery();
  ......
  rset.close();
} catch (SQLException e) {
  e.printStackTrace();
} finally {
  if (pstmt != null) try {pstmt.close();} catch (SQLException e) {...}
}
```

Finally中调用关闭游标，
TRY块结束后总会执行

Jdbc1性能案例3：永远不够的processes

- 吞吐量接近于零
- 负载也接近于零
- CPU使用率也接近于零

- 告警日志中出现ORA-00020错误？
- 吞吐量暴跌，应用会话无法连接数据库？开发人员要求DBA进一步加到processes参数？

Jdbc1性能案例3：原因

```
try {
  ......
  rset = pstmt.executeQuery();
  ......
  rset.close();
  pstmt.close();
  conn.close();
} catch (SQLException e) {
  e.printStackTrace();
}
```

代码里是写了关闭connection，但存在还没执行就跑到异常处理里去的可能。

Jdbc1性能案例3：对策

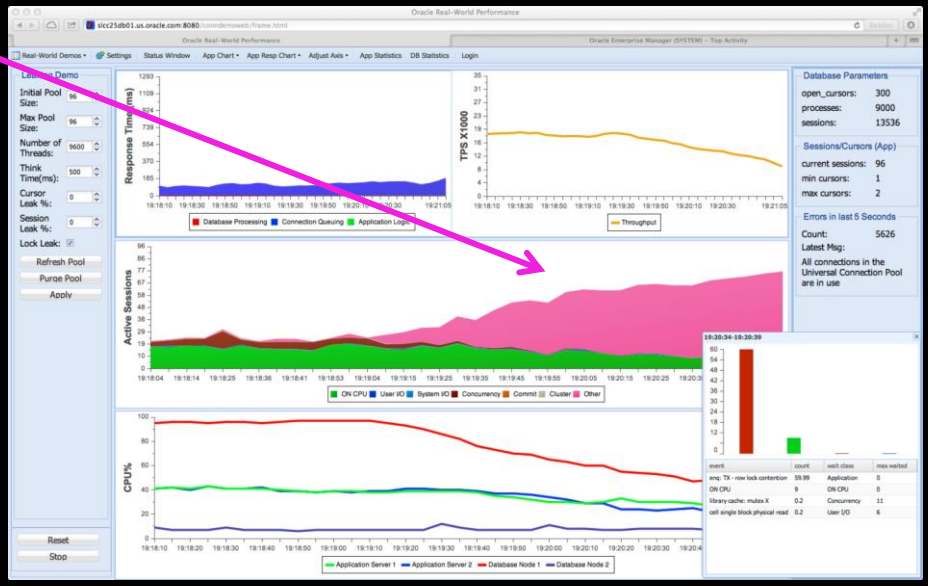
```
try {
  ......
  rset = pstmt.executeQuery();
  ......
  rset.close();
} catch (SQLException e) {
  e.printStackTrace();
} finally {
  if (pstmt != null) try {pstmt.close();} catch (SQLException e) {...}
  if (conn != null) try {conn.close();} catch (SQLException e) {...}
}
```

Finally中调用关闭connection，TRY块结束后总会执行

Jdbc1性能案例4

- 莫名出现大量锁等待

- 没有任何告警，但所有应用全部超时？
- 大量session处于锁等待挂起状态
- 开发人员要求DBA去kill锁数据的session?



Jdbc1性能案例4：原因

```
try {
  ......
  rset = pstmt.executeUpdate();
  ......
  conn.commit();
  pstmt.close();
  conn.close();
} catch (SQLException e) {
  e.printStackTrace();
}
```

代码中写了commit，但没考虑异常处理

可能的session leak

Jdbc1性能案例4：对策

```
try {
  ......
  rset = pstmt.executUpdate();
  ......
  conn.commit();
} catch (SQLException e) {
  if (conn != null ) try {conn.rollback();} catch (SQLException e) {...}
  e.printStackTrace();
} finally {
  if (pstmt != null) try {pstmt.close();} catch (SQLException e) {...}
  if (conn != null) try {conn.close();} catch (SQLException e) {...}
}
```

Other tips:Set AutoCommit Off

关闭自动commit，在应用真正需要的时候commit，
即维护了必要的事务又减少了log file sync等待

```
Connection.setAutoCommit(false);
```

Other tips:批量Insert

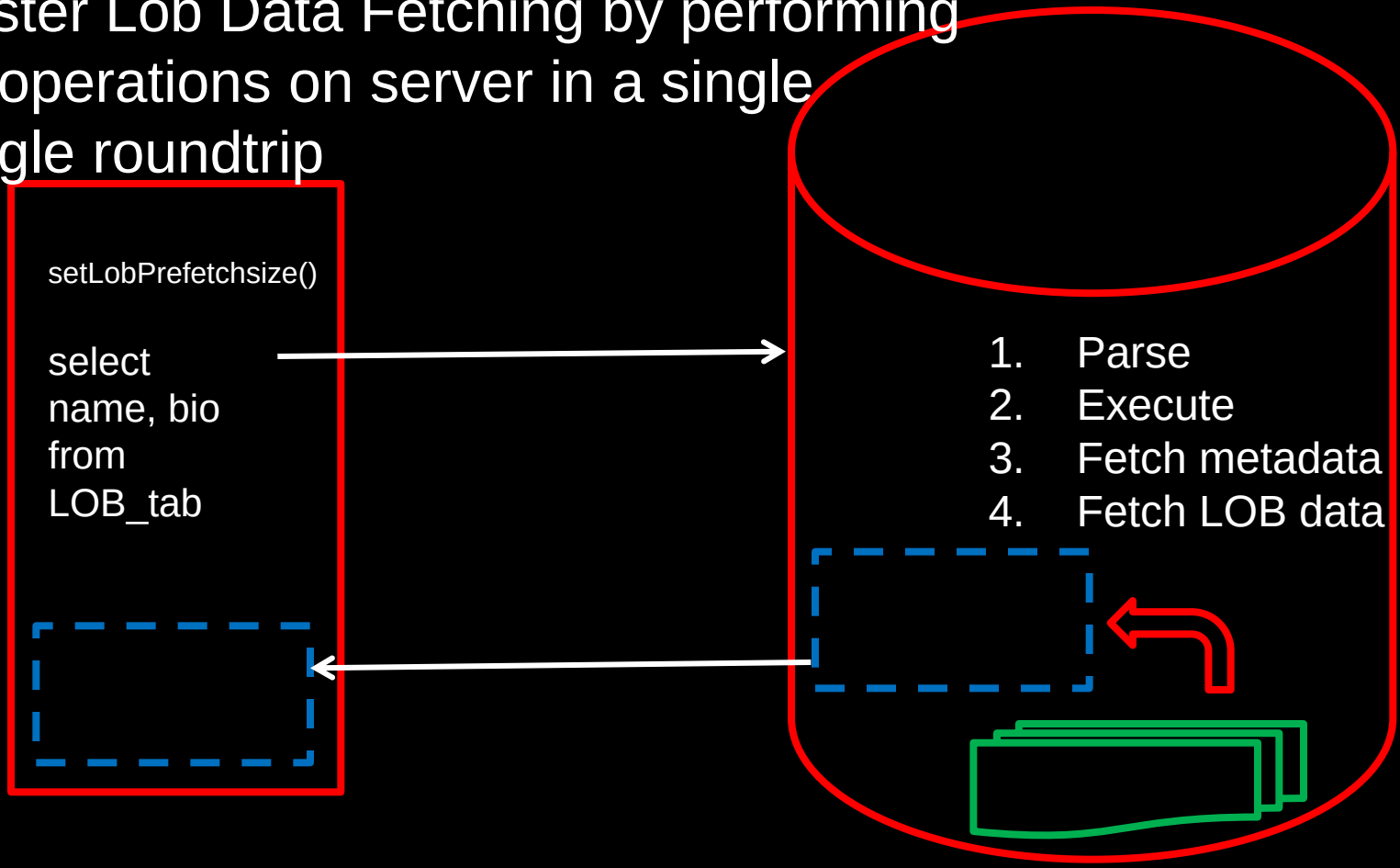
```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection conn = DriverManager.getConnection ("jdbc:oracle:oci8:@","scott","tiger");
PreparedStatement ps = conn.prepareStatement ("insert into dept values (?, ?,?)");
//Change batch size for this statement to 3
((OraclePreparedStatement)ps).setExecuteBatch (3);
ps.setInt (1, 23);
ps.setString (2, "Sales");
ps.setString (3, "USA");

ps.executeUpdate (); //JDBC queues this for later execution
ps.setInt (1, 24);
ps.setString (2, "Blue Sky");
ps.setString (3, "Montana");
ps.executeUpdate (); //JDBC queues this for later execution
ps.setInt (1, 25);
ps.setString (2, "Applications");
ps.setString (3, "India");
ps.executeUpdate (); //The queue size equals the batch value of 3
//JDBC sends the requests to the
// database
ps.setInt (1, 26);
ps.setString (2, "HR");
ps.setString (3, "Mongolia");
ps.executeUpdate (); //JDBC queues this for later execution
((OraclePreparedStatement)ps).sendBatch ();
//JDBC sends the queued request
……..
```

Other tips:Prefetch Rows

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection conn = DriverManager.getConnection
("jdbc:oracle:oci8:@","scott","tiger");
//Set the default row prefetch setting for this connection
((OracleConnection)conn).setDefaultRowPrefetch (7);
/* The following statement gets the default row prefetch value for
the connection, that is, 7.
*/
Statement stmt = conn.createStatement ();
/* Subsequent statements look the same, regardless of the row
prefetch value. Only execution time changes.
*/
ResultSet rset = stmt.executeQuery ("select ename from emp");
System.out.println ( rset.next () );
while( rset.next () )
System.out.println ( rset.getString (1) );
//Override the default row prefetch setting for this statement
( (OracleStatement)stmt ).setRowPrefetch (2);
rset = stmt.executeQuery ("select ename from emp");
System.out.println ( rset.next () );
while( rset.next () )
System.out.println ( rset.getString (1) );
```

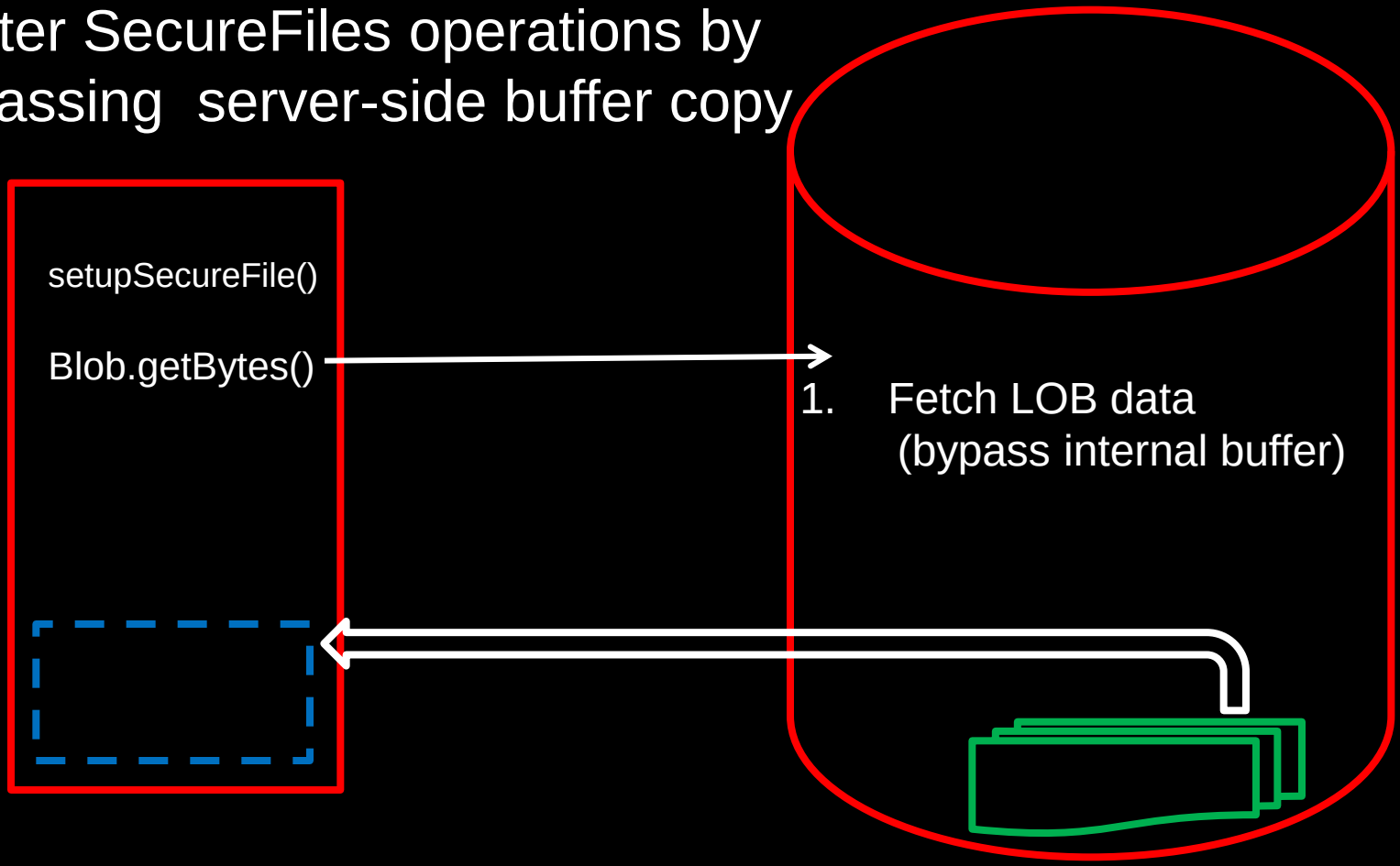
Oracle Database 11g R2
JDBC BasicFiles LOB Pre-Fetch

Faster Lob Data Fetching by performing all operations on server in a single single roundtrip



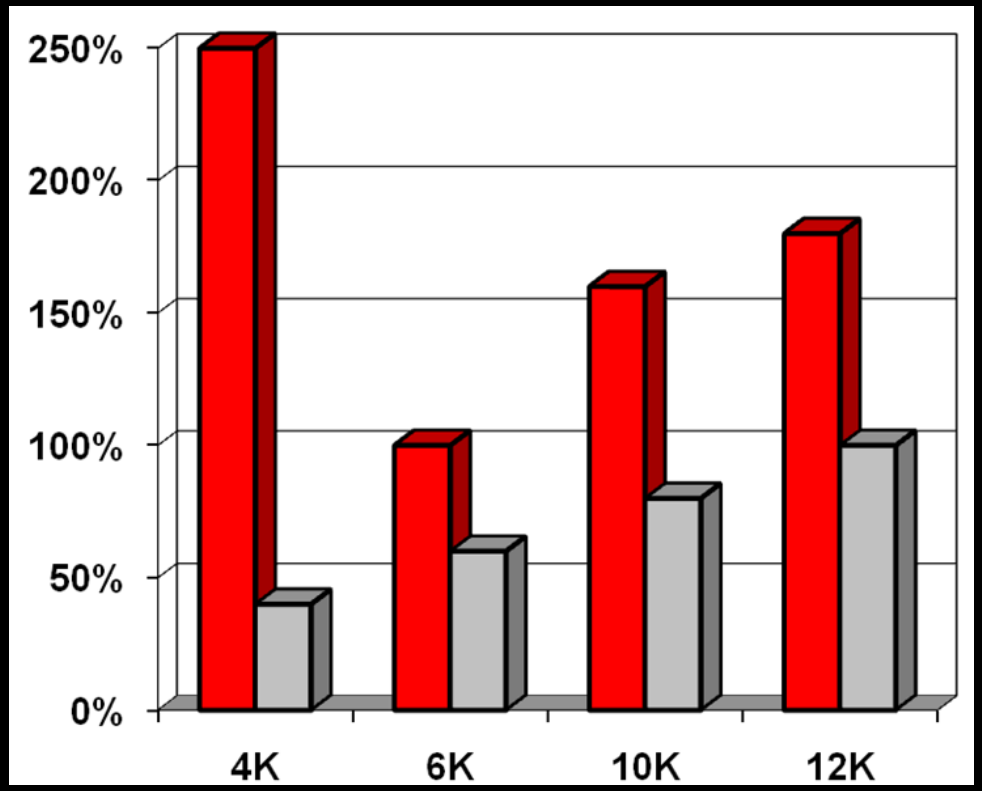
Oracle Database 11g R2
Zero-Copy SecureFiles LOB

Faster SecureFiles operations by bypassing server-side buffer copy



JDBC Lob Pre-Fetch Performance

Performance benchmark fetches CLOBs of sizes 1K to 50K with **PREFETCH enabled** and **PREFETCH disabled**, against BASICFILE LOBs and SECUREFILE LOBs.



ParnassusData™

www.parnassusdata.com

400-690-3643

Thank You