SLAC - PUB 5290
June 1990
(M)

Applications Servers Provide Easy Data Access*

Frank Rothacker

Stanford Linear Accelerator Center
Stanford University, Stanford, CA 94309-4349

Introduction

An application server can do some things that are truly remarkable. Used in the right situation, an application server can drastically improve response time, increase reliability, enhance security, add flexibility, or reduce software costs. It can do things that would be otherwise impossible, like real-time exchange of data between incompatible database management systems. In one case it reduced the response time for a simple Oracle query from over six seconds to a half-second, while also allowing access to the Oracle data from an application written in a non-relational system.

An application server combines the technologies of a database management system, an application specific background process, and simple operating system or network commands designed for communicating between users.

Oracle two task process

-Lets begin with a review of the Oracle processes. On multi-user computing platforms, Oracle is implemented using a two-task architecture as illustrated in figure 1.

One process is called the foreground process. It is created when the user logs onto the operating system from a terminal. Various applications, including SQL*PLUS, SQL*FORMS, and high level language interface programs, run in the foreground process. There is one foreground process for each logged on user.

The other process is called the background process. The oracle kernel runs in this process. In the Oracle implementations for most operating systems,

* Work supported by the Department of Energy Contract, DE-AC03-76SF00515
Submitted for Publication

several background processes are needed to perform kernel functions. The background processes are either created automatically by the operating system, or by other Oracle background processes using AUTOLOG or similar operating system commands. Because no user terminal is connected to the background processes, they are sometimes call disconnected processes, or virtual service machines.

On single computer implementations (non-distributed), the Oracle foreground and background communicate through operating system inter-task protocols, such as IUCV or VMCF. On distributed implementations, the foreground and background processes communicate over SQL*NET.

Oracle overhead

The Oracle architecture is very powerful and general. But Oracle also has a lot of overhead, reducing performance and making applications more difficult to set up. Consider the problems of setting up an on-line phone directory. Suppose a company has one or more large mainframe computers that are used primarily for scientific applications. Most of the users do not require access to Oracle except for the phone book. Some users would like to look up phone numbers while they are in a CAD/CAM application.

Using SQL*PLUS, it would be easy to search an Oracle table for a name, and then display the phone number. Also, appropriate use of SQL scripts and EXEC's could relieve the user of the need to understand Oracle commands. But consider the following problems:

1. The user would have to be using a computer that has either Oracle or SQL*NET running. SQL*PLUS would have to be installed on the user's computer.

2. The proper environment would have to be set up to run an SQL*plus application. That includes accessing disks or files containing the Oracle software and the application software. Also the user would have to have enough virtual memory to run SQL*PLUS. The user might not be able to run the phone directory application at all if he/she is in another application that conflicts with SQL*PLUS.

3. The user would have to wait for a connection to the Oracle database. On an IBM 3090, this may take as long as six seconds, even though the actual query might take only a fraction of a second.

4. Upon completion of the phone directory application, the software would have to restore the user's environment.

Because of the above problems, the phone directory might be sluggish and inconvenient. The key to solving these problems is to move the application to a background process called an application server.

Application servers

An application server is a disconnected process that is automatically logged on by the operating system. The application server connects to a database and remains connected while it processes the requests of a variety of users. An Oracle application server may run SQL*PLUS or a procedural language program through Oracle's host language interface. To the Oracle kernel, the application server appears to be an ordinary user.

Users communicate with the application server through simple operating system or networking commands. Most multi-user operating systems have a message command for communicating with the operator or other logged on users. On most networkedcomputers, these commands have been expanded so that a user on one computer can send a message to a user on another computer.

Applications servers differ from database servers in that application servers process only certain transactions for one application or a closely related set of applications. An application server trades off generality for efficiency and application specific flexibility. Although an application server is being used, authorized persons can still directly access database tables. On a given computer system, several application servers may be run simultaneously -- each to perform a certain class of tasks.

Advantages of using an application server

Application servers give instantaneous responses to simple queries. Because the application server is continuously connected to the database, there is no need to wait for a connection. Since a number of users are sharing the same application server, there are fewer connections to the kernel, improving the performance of Oracle.

Users of an application server, can access Oracle data while working in almost any environment. Most interactive computer applications provide the user with the ability to execute basic operating system commands, like responding to messages from the operator or other users. For example, Oracle has the HOST command. Most other applications have similar commands. In this way, users do not have to leave their application to interact with the database.

The application server does not have to be running on the same computer as the user. Even SQL*NET does not have to be running. All that is needed is a way to send a message to the other computer. The computers can even communicate over existing e-mail networks, such as Internet, Bitnet, Decnet, or Arpanet. All the user has to do is send e-mail in a pre-defined format to the application server, and wait for a reply. This system allows data access over vast distances, without the need to licence any Oracle product on the user's local computer. Application servers have had a long history of providing remote access to databases. Much of this work was done at Stanford on a database called Spires[1,2,3].

Application servers can provide an unparalleled degree of security and control. The unauthorized user lacks access to the data or to the programs running on the application server. Users need not be granted access to Oracle tables or views. Since the security system in running on the application server, the user cannot modify it or bypass it. Even sophistication users having thorough knowledge of Oracle and the operating systemcannot bypass the security system.

Consider the possibilities. Suppose a company wanted to restrict access to its customer list to on-line viewing only. Management does not want employees to print a list that they could take home -- or sell to competitors. Further, suppose management wants to prevent an employee from systematically viewing each customer record to manually prepare such a list. An application server could easily enforce such security requirements. It could count the number of queries by each employee, restrict each employee to so many per day, and print reports of unusual activity.

An example

Stanford Linear Accelerator Center used an application server to implement a table look up command on an IBM 3090 running the VM/XA operating system. The look up command can be issued directly from the CMS command prompt, and has the syntax:

```
LOOKUP viewname viewkey attribute attribute . . .
```

This command allows any authorized user to look up data in selected Oracle tables and views. Each “view” in this application is defined to have one look up key. By specifying a **viewname** and the value of a **viewkey,** a unique row of an Oracle table is selected. The requested attributes of that row are displayed on the user’s screen. For example, to look up the title and status of a general ledger account, the user would issue the cornrnand:

```
LOOKUP  ACCOUNT  12345  TITLE  STATUS
```

To find the name and department for a given employe number, the user would use the command:

```
LOOKUP  EMPNO 1234  NAME  DEPT
```

The look up command is implement with a REXX EXEC that accepts the user’s parameters and passes them to the application server. An early (and simple) version of this exec is shown below:

```
/* Lookup driver */
Arg arg                         /* Fetch user parms */
'CP SMSG LOOKS.ERV'  arg        /* Call the server */
```

The above EXEC simply passes the user’s parameters to the virtual machine LOOKSERV running the application server.

An early version of the application server was implemented with the following EXEC:

```
/* Oracle Application Server (LOOKSERV EXEC)*/
fid = 'TEMP FILE A'
Arg arg          /* Parms from prior call to myself */

If arg='' Then Do           /* First time only */
     'WAKEUP (AUTH'         /* Authorize SMSG */
     Push 'HOST EXEC LOOKSERV WAIT'    /*Call myself*/
     Push 'HOST SPOOL'
     'EXEC SQLPLUS /'            /* from sqlplus */
     Exit
     End

If arg~='WAIT' Then Do      /* Send result to user */
     user = arg
     /* Read result from the Oracle spool file */
     'EXECIO 1 DISKR' fid QFILE(fid,'RECNO')-2 '(LIFO'
                    /* QFILE returns the record count */
     Pull result
     'CP MESSAGE' user result         /* send to user */
     End
```

```
 'WAKEUP (SMSG'               /*Wait for message from user/*
Pull . user vname vkey attr
Push 'HOST EXEC LOOKSERV' user
Push 'SELECT' attr 'FROM' vname 'WHERE KEY='vkey';'
```

The above EXEC is somwhat complicated because SQL*PLUS lacks a REXX interface. The only way an EXEC can regain control after calling SQL*PLUS is by the EXEC stacking a call to itself. After SQL*PLUS is invoked, it will read the stack and call the EXEC. Oracle Corporation is now developing a REXX interface, and life may get easier.

The LOOKSERV EXEC is actually three EXEC's rolled into one. The first section is invoked after the service machine is autologed by the system. In this case, no pararneters are supplied and the EXEC does initialization. It stacks a call to itself, then invokes SQL*PLUS. SQL*PLUS in turn, reads the stacked command and calls the EXEC.

When called the second time, the EXEC waits for a message from the user. After receiving a message, it stacks an Oracle SELECT command, and also stacks another call to itself.

On subsequent invocations, the EXEC reads the SQL*PLUS spool file and sends the results back to the user. It then waits for another user message.

Since its original implementation, the above code has been greatly enhanced to improve security, reliability, error detection, and recovery. The early version is presented here because it contains fewer extraneous details. Tailoring an application server for a particular purpose is left to the reader's imagination.

The LOOKUP command is now being used access Oracle data from an aplication written in another database system. This technique has eliminated the need to maintain duplicate copies of the same data in two systems.

LOOKUP has also become a way of checking on the health of the Oracle system. A program has been written which calls LOOKUP with a pre-determined query every ten minutes. It checks results and logs the response time. If the wrong result is returned, or if the response time is unacceptable, it sends a message to the Oracle DBA.

Creating your own application server

The method of programming an application server depends mainly on its intended use. But the following elements are in common:

1. A background process is needed for the server. Most multi-user computing platforms have such a provision. On a VAX it is called a disconnected process, and on a VM system it is called a VM service machine. The background process is brought up automatically with the operating system.

2. A method is needed for communication between user and server. The simplest solution is to use the operating system facility for sending interactive messages between users. Structured e-mail files also work well, and allow access through existing international networking facilities.

3. There must be a way to capture messages and synchronize processes. On the VM system the WAKEUP command works well for this purpose.

4. There should be software running on the user end which formats messages to the server and passes results back to the user in the manner most helpfull.

Limitations

Application servers are not for everything. They work best when many users are executing short infrequent transactions. An application server eliminates the overhead of each user connecting to the database, but it introduces its own overhead, mainly as additional inter-process communication. Heavy data entry applications should not use an application server. But the use of application servers does not preclude concurrent use of the database in the normal manner.

Refferences

1. Crane, George, “Networking Applications in Spires”, Spires Fall Workshop, October 7, 1985.

2. Boeheim, Chuck, "Spires Network Servers", Spires Spring Workshop, March 12, 1986.

3. Crane, George, "Overview of Remote Spires", Spires Fall Workshop, September, 29, 1988.

Oracle processes

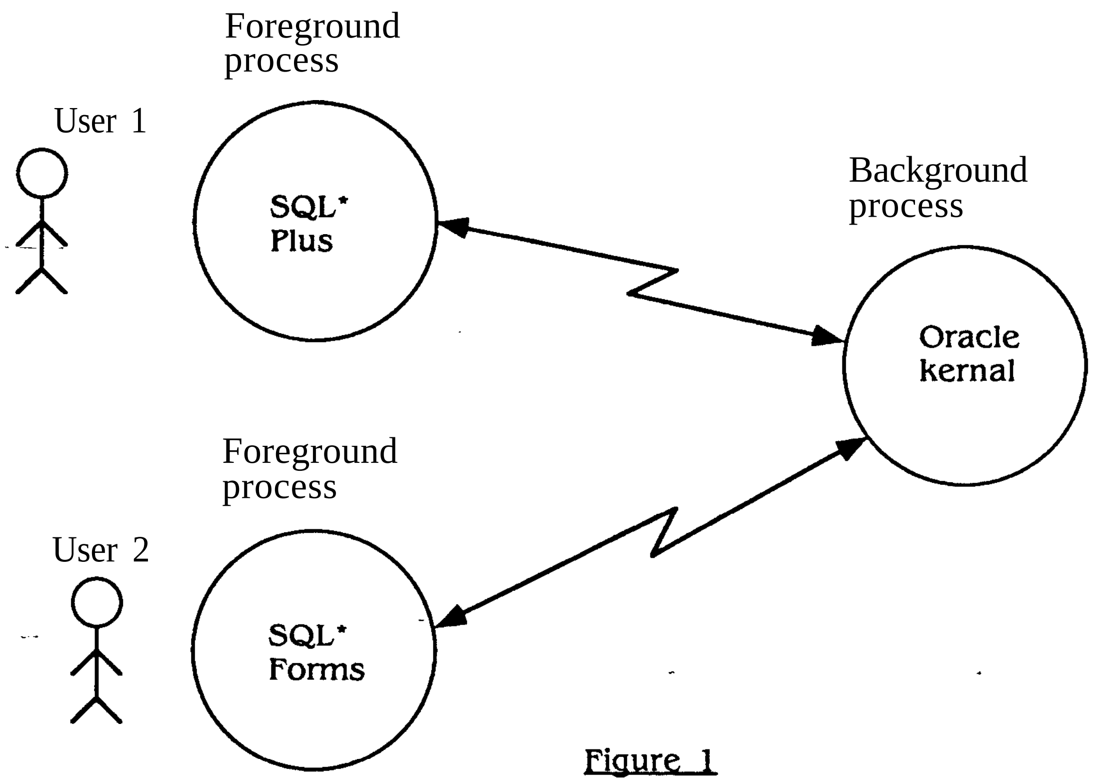
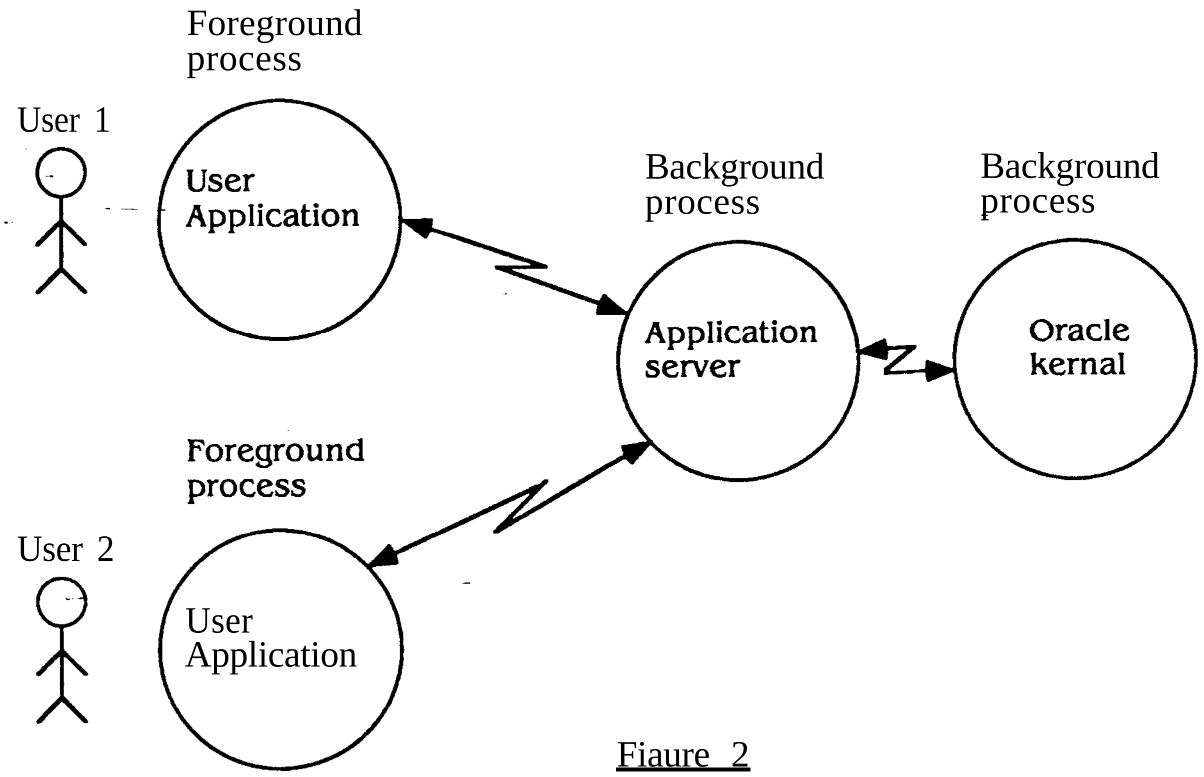


Figure 1

Application server



Fiaure 2