IBM eServer J iSeries J

Session: 401918

# DB2 UDB: Making IBM's ODBC Fly!

Brent Nelson
bmnelson@us.ibm.com





## Agenda

- Introduction
- Performance Considerations
  - Application Design
  - Network
  - Database Design
- Summary
- Appendices

## Introduction





## Performance Considerations

- **Application design**
- Network
- Database design

## Application Design Overview

- Choice of programming interface
- Tips on calling ODBC APIs
- Block insert
- Isolation level and concurrency





## Choosing a Programming Interface

- If you use a higher level language, e.g. ADO, you should understand how an ADO call gets converted to multiple ODBC calls.
- Write ODBC 3.x APIs: Don't use 2.x APIs for new applications
  - Coding directly to the ODBC APIs allows application to take advantage of server-specific features, e.g. block insert. Also you have better control of your application

## ODBC Tips

- Leave application connected
- Allocate statement once – statement pooling
- Prepare statement once/execute statement repeatedly
- Use bound variables on fetch APIs
  - Use the right one (SQLFetch, SQLFetchScroll, SQLExtendedFetch) with right block size
  - SQLBindCol vs. SQLGetData
  - Rebinding with new offset
- Do not call API to get the proper returned buffer length if length can be known in advance



IBM eServer iSeries

## ODBC Tips

- Use parameter markers
- Perform domain/validity checking at the client: avoid scalar functions, etc... but make sure rules match those at the server

© 2003 IBM Corporation

## Block Insert Example 1

Example1:  Insert with Constants

```
/*****************************************/
/* repeat for each row to be inserted */
/*****************************************/
strcpy(stmt,"insert into IBMLIB.TAB1 values('ax',123,'c')");

rc = SQLExecDirect(hStmt,stmt,SQL_NTS);
:
```



IBM eServer iSeries
IBM

## Block Insert Example 1

Example 1 comments:

- Gets the job done, but:
  - Requires full parse for each statement
  - Requires full open/close of target table on server
  - Requires send/receive for each row

© 2003 IBM Corporation

## Block Insert Example 2

Example 2: Insert with Parameter Markers

```
strcpy(stmt,"insert into IBMLIB.TAB1 values (?,?,?)");
rc = SQLPrepare(hStmt,stmt,SQL_NTS);
/* Set bind variables for each parameter marker */
rc = SQLBindParameter(hStmt, .... )
:
for (i=0;i<ROW_COUNT;i++) {
    /* Set variables for corresponding parameter */
    /* markers, and then execute the statement.  */
      strcpy(col1,value);
      rc = SQLExecute(hStmt);
 }
```



IBM eServer iSeries

## Block Insert Example 2

Example 2 comments:

- Better than constants because:
  - Doesn't require parsing
  - Doesn't require full open/close of target table
  - Still requires flows for each statement

© 2003 IBM Corporation

## Block Insert Example 3

```
Example 3:  Blocked Insert

/* Note, the "? rows" clause is needed when accessing pre-V5R1 systems */
strcpy(stmt,"insert into IBMLIB.TAB1 values (?,?,?)");

rc = SQLPrepare(hStmt,stmt,SQL_NTS);

rc = SQLSetStmtAttr(hStmt,SQL_ATTR_PARAMSET_SIZE,
       (PTR)ROW_COUNT,SQL_IS_INTEGER);

/* Set bind variables for each parameter marker */
rc = SQLBindParameter(hStmt, .... )
:
for (i=0;i<ROW_COUNT;i++) {
        /* Set variables in the array for corresponding parameter */
        /* markers, and then execute the statement.                */
        strcpy(col1[i],value);
        :
 }
rc = SQLExecute(hStmt);
```





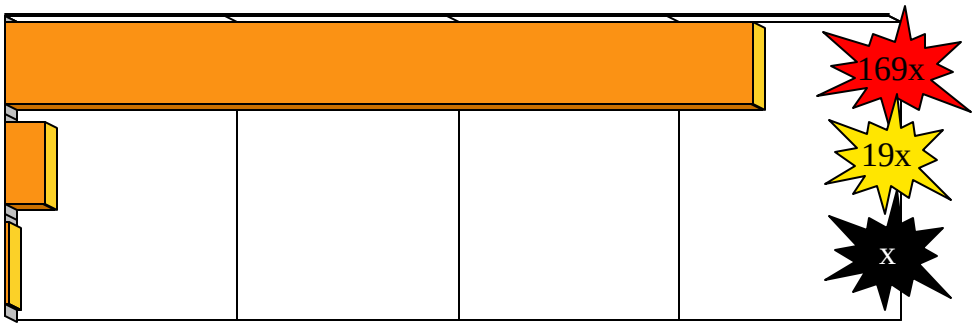
## Block Insert Example 3

Example 3 comments:

- Best alternative because:
  - Parsed only once
  - Avoids full open/close of target table
  - 1 send/receive for N rows
  - Path optimized from client->database
- Drawbacks
  - May not be practical for all applications
  - AS/400 only feature if "? rows" clause is used

## Block Insert Performance

For 500 36-byte rows with three columns



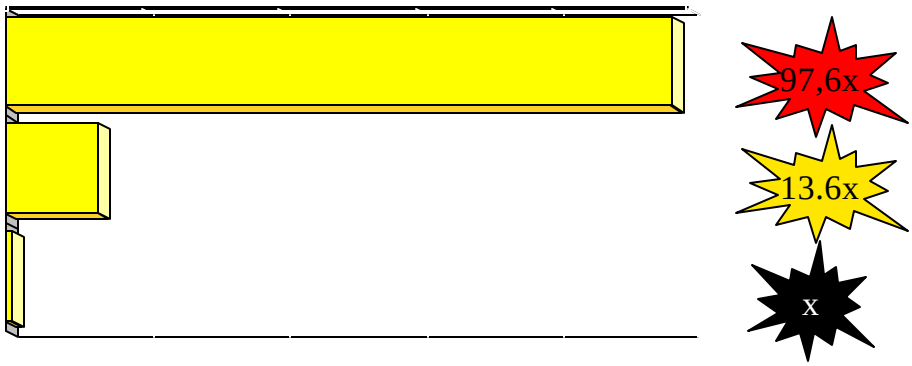

Server CPU time (169x, 19x, x)





## Block Insert Performance

For 500 36-byte rows with three columns

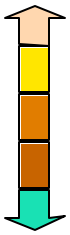



Response Time (97.6x, 13.6x, x)

## Isolation Level

- By default, ODBC runs with autocommit ON
  - Equivalent to *NONE on iSeries
- Must use SQLSetConnectAttr to turn autocommit OFF to use other isolation levels
- Use lowest level of transaction isolation for least overhead

| | Isolation Level | Overhead |
|---|---|---|
| (*NONE) | Commit immediate | less |
| (*CHG) | Read uncommitted | |
| (*CS) | Read committed | |
| (*ALL) | Repeatable read | |
| (*RR) | Serializable | more |







## Concurrency

- Concurrency is ability of two transactions to use the same data at the same time
- Increasing isolation level leads to reduction in concurrency
- Use appropriate ODBC cursor concurrency setting (SQL_ATTR_CONCURRENCY)
- Avoid “Open all cursors as updatable” DSN setting

## Application Design Summary

- Write directly to the APIs for greatest control
- Use parameter markers:
  - SQLPrepare once, SQLExecute many times
- Use blocking
- Set appropriate isolation level





## Application Design Summary

- General ODBC tips & techniques
  - http://www.ibm.com/as400/developer/client/index.html
- ODBC DSN keywords
  - http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.h

## Performance Considerations

- Application design
- **Network**
- Database design





## Network

- About 1/3 of ODBC performance problems are actually network performance problems
- ODBC sends much larger blocks of data then most applications

## CWBCOPWR

- Found in \Program Files\IBM\Client Access directory
- See CWBCOPWR.HTM for help
- Options to concentrate on:
  - Communication buffer size (Option /SC)
  - TCP/IP buffer size (Options /WSS and /WSR)
  - TCP/IP nagling (Option /NGL)





## Reduce Trips to Server

- Ways to reduce trips
  - Stored procedures
  - Triggers
  - Connection pooling
  - Block inserts
  - Block fetches
  - Block with fetch of 1 row
  - Lazy close
  - Pre-fetch

## Reduce Data Sent to Server

- Ways to reduce data
  - Data compression
  - LOB threshold
  - Catalog library list
  - Avoid “select *” SQL statements





## Data Compression

- “Enable Data Compression” DSN setting ON by default
- Can be activated at the connection level or statement level
- Highly recommended for variable length field with repetitive characters, e.g. VARCHAR, VARGRAPHIC
- New compression algorithm in V5R1
  - compression of data from and to server
  - also supports double byte characters

## Network Summary

- Reduce Trips to Server
- Reduce Data Sent to Server
- Use Data Compression



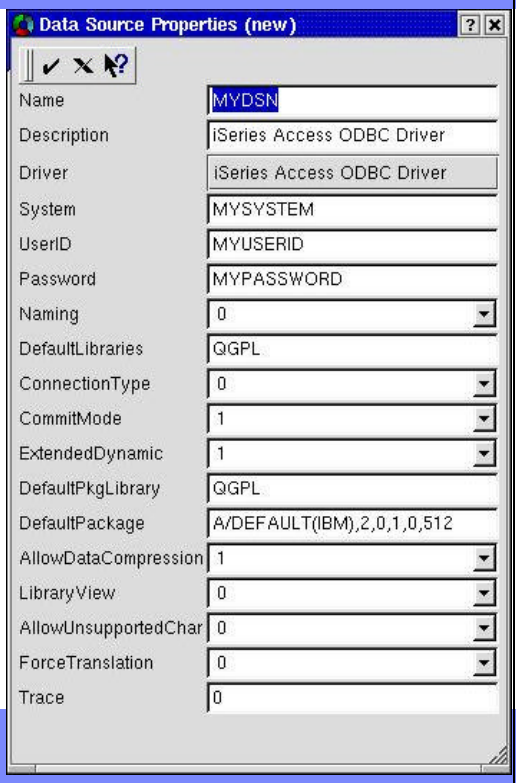


## Windows ODBC DSN GUI

- Performance tab
- Advanced performance options

## Linux ODBC DSN GUI

- ExtendedDynamic
- AllowDataCompression
- Other options are added to the .odbc.ini file or programmatically specified via SQLDriverConnect API








## Performance Considerations

- Application design
- Network
- **Database design**

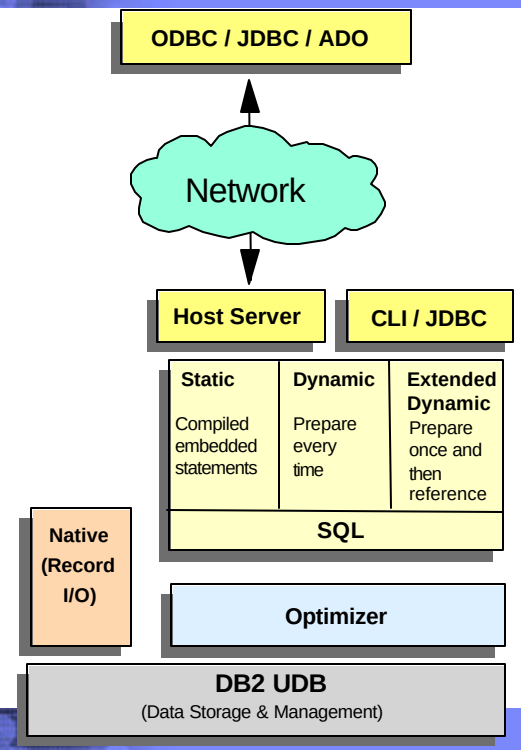
## Database Design Overview

- Indexes
- Extended dynamic support
- Stored procedures
- Trigger programs





## SQL Interfaces



## SQL Statement Tuning

- Avoid SELECT *
- SQL clauses:
  - FOR FETCH ONLY / FOR UPDATE
  - FETCH FIRST N ROWS ONLY





## Indexes

- Two methods for accessing data--keyed and sequential
  - Lack of understanding can seriously impact performance
- Aimed specifically at optimizing queries (SELECTs)
- Most important aspect--proper indices for queries

## Indexes

- Index is required for following cases:
  - ORDER BY
  - GROUP BY
  - JOIN of two tables
- Optimizer will create index if an appropriate one doesn't exist



IBM eServer iSeries

## Indexes

- Create index over tables when queries return less than 20% of table
  - Create index over columns used in WHERE clause
  - Create index over columns used to join tables
  - Create index on grouping columns

© 2003 IBM Corporation

## Access Plans and ODPs

- Minimize access plan builds
  - Access plans are the “way query is processed to retrieve result data”
  - Reduces CPU use on server
  -
- Reuse (Open Data Path) ODPs
  - An important performance tuning tip
  - Prepare once/run many, statement pooling, connection pooling, use parameter markers, etc
  - Cache package locally (statement names are reused, hence ODP is reused)





## Extended Dynamic Access

- Caches SQL statements on server or local
- Allows reuse of statements
  - Prepare once, execute multiple times (cross sessions)
- Statements can be shared among many users (cross sessions)

## Extended Dynamic Access

- Extended dynamic option can have the biggest impact IF parameter markers are used. Also can reuse the ODP by using the local cache.
- Package can become unusable if package attributes do not match application
  - Different CCSID, Date & time format attributes, decimal delimiter, default collection, etc
  - With ODBC packages, a default collection for unqualified names can be specified - if package already exists and the client application has a different default collection, then package cannot be used
  - If package unusable, new requests are executed as "pure" Dynamic SQL





## Stored Procedures

- Powerful tool--accepts input parameters, returns output parameters
- SQL procedure
- Stored procedure as external procedure
  - Does not need to contain SQL
  - Can be any C, RPG, CL, COBOL, Java program
- Can return multiple result sets
- Can hide details of application from user

## Stored Procedures

- Stored Procedures can be utilized to provide...
  - Static SQL performance behavior to dynamic ODBC & JDBC client requests
  - Access to tuning knobs/precompiler options such as ALWCPYDTA, etc





## Stored Procedures Example 1

Example:
Logic to execute 2 statements w/o stored procedure:

```
/* Application runs INSERT statement - if the     */
/* statement is successful, it runs an UPDATE   */
/* statement.                                                */
strcpy(stmt,"INSERT INTO IBMLIB.TAB1 VALUES(?,?,?)");
rc=SQLPrepare(hstmt1,stmt,SQL_NTS);
strcpy(stmt2,"UPDATE IBMLIB.TAB2 SET COL1 = ...");
rc=SQLPrepare(hstmt2,stmt2,SQL_NTS);
rc=SQLBindParameter(hstmt1,... );
:
rc = SQLExecute(hstmt1);
if (rc==SQL_SUCCESS)
 {
       rc=SQLExecute(hstmt2);
       if (rc==SQL_SUCCESS)
       :
```

## Stored Procedures Example 2

Example:
Logic to execute the same statements using stored procedure:

```
/* Application invokes stored procedure passing all      */
/* the parameters necessary for both the INSERT and */
/* UPDATE statement                                      */
strcpy(stmt,"CALL IBMLIB.PROC1 (?,?,?)");
rc = SQLPrepare(hstmt1,stmt,SQL_NTS);
rc = SQLBindParameter(hstmt1,... );
:
rc = SQLExecute(hstmt1);
if (rc==SQL_SUCCESS) {
   :
```





## Stored Procedures – Executing CL Commands

- Execute system CL commands via the QCMDEXC stored procedure
- CALL QSYS/QCMDEXC ('CHGQRYA QRYTIMLMT (0)', 0000000021.00000)
  - 21 is the character length of the CL command string
- Call a stored procedure that executes system commands or utilities like STRDBMON

## Trigger Programs

- Based on target file/table
- Invoked for each INSERT/UPDATE/DELETE against the table as defined by the triggers
- Advantage over stored procedures
  - Better performance (marginal)
- Caveat:  All operations, regardless of origin, fire the trigger program





## Exit Programs For Job Initialization

- When a JDBC or ODBC job (QZDASOINIT) is initiated in subsystem (QSERVER), the registered exit program is called automatically
- Add program to exit point for ODBCINIT via WRKREGINF CL command at QIBM_QZDA_INIT
- Example:  Create an exit program with CL that conditionally activates debug mode (STRDBG UPDPROD(*YES) ) so that you can see the optimizer debug messages for an ODBC or JDBC request

## Database Design Summary

- Understand your application and the queries it runs
- Identify most frequently used selection/join columns
- Create indices where needed
- Use extended dynamic access
- Use stored procedures where possible





## Additional Client-side Information

- ODBC
  - http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/rz
  -
- OLE-DB
  - See OLE-DB Tech Ref installed with iSeries Access
  -
- JDBC
  - http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/rz

## Additional Information

- DB2 UDB for iSeries home page
  - http://ibm.com/servers/eserver/iseries/db2/
- Newsgroups
  - comp.sys.ibm.as400.misc
  - comp.databases.ibm-db2
- Education Resources - Classroom & Online
  - http://ibm.com/servers/eserver/iseries/db2/db2educ_m.htm
- DB2 UDB for iSeries Publications
  - Online Manuals: http://ibm.com/servers/eserver/iseries/db2/books.htm
  - Indexing Strategies for DB2 UDB for iSeries: http://www.iseries.ibm.com/developer/bi/documents/strategy/strategy.pdf
  - DB2 UDB for AS/400 Redbooks (http://ibm.com/redbooks)
    - DB2 UDB for AS/400 Object Relational Support (SG24-5409)
    - DB2/400 Advanced Database Functions (SG24-4249-02)
- New book - SQL/400 Developer's Guide by Paul Conte & Mike Cravitz
  - 29th Street Press, ISBN 1-882419-70-7
  - http://as400network.com/str/books/Uniquebook2.cfm?NextBook=183





## Database Tools

- Graphical
  - Centerfield Technology DB Essentials
    - http://www.centerfieldtechnology.com
  - Client Access Operations Navigator
    - http://www-1.ibm.com/servers/eserver/iseries/access/
  - Visual Explain
    - http://publib.boulder.ibm.com/iseries/v5r2/ic2924/info/rzajq/rzajqmstvisexpl
- Text-based
  - DBMON (Database monitor)
  - Debug joblogs
  - SST (iSeries Communication Trace)
  - ODBC trace (SQL.LOG)

# DB2 UDB Tech Tip: Improving Query Performance

- DB2 UDB for iSeries has a phenomenal query optimizer built into it.
  - without the DB2 Symmetric Multiprocessing (SMP) feature your SQL database tasks and index bui running single-threaded?
  - the DEFAULT system tuning setup could be significantly hindering ODBC performance?
  - the database utility called DB2 OLAP can provide sub-second response times to complex queries?
- For more information about query optimization, check out these resources:
  - S6140 - DB2 UDB for iSeries SQL and Query Performance Tuning and Monitoring Workshop:
    - http://www-3.ibm.com/servlet/com.ibm.ls.lsow.servlets.CourseDescriptionServlet?coursecod
  - White paper: "Indexing Strategies for DB2 UDB for iSeries"
    - http://www-919.ibm.com/servers/eserver/iseries/developer/bi/documents/strategy/strategy.p
  - DB2 Symmetric Multiprocessing and DB2 OLAP Utilities:
    - http://isource.ibm.com/cgi-bin/goto?on=c4268db2/products





# Performance Service Tips

- Before calling SupportLine with a query performance problem...
  - Run query in DEBUG mode and check JOBLOG
    - Index recommendations
    - Understand query implementation
  - Check resources and Work Management
    - QQRYDEGREE or CHGQRYA
    - Memory and MAX ACTIVE settings
    - What else is running?
    - Does QQQOPTIONS data area exist?
  - Check file stats
    - Size of objects, number of rows
    - Number of indexes
  - Understand your data
  - Save JOBLOGs and system settings

## Summary

- Understand tradeoffs for performance
- Application Design:
  - Use parameter markers
  - Isolation level
- Network:
  - Use data compression
  - Use blocking
- Database Design:
  - Indexes
  - Use extended dynamic
  - Use stored procedures





## Appendices

## Appendix A1

- Extended dynamic access
- Use the extended dynamic feature of iSeries Access ODBC driver (also available to iSeries JDBC Toolbox driver)
  - Feature externalized as data source option
  - Used in conjunction with "Cache Package Locally" option, the majority of reuse processing will occur on client - especially with SQLExecDirect
  - Allows for private or shared packages





## Appendix A2

- Extended dynamic
  - Use PRTSQLINF command on the iSeries to dump the package containing your statements
  - PRTSQLINF produces spoolfile showing syntax and optimization information
  - Use PRTSQLINF to ensure all SELECT statements have parameter markers

# Appendix A3

Extended Dynamic
Sample PRTSQLINF output:

```
5722SS1 V5R2M0 030905     Print SQL information         SQL package QGPL/ODBCXXXFBA
Object name.............QGPL/ODBCXXXFBA
Object type.............*SQLPKG
  CRTSQL***
     PGM(QGPL/ODBCXXXFBA)
     SRCFILE(        /        )
     SRCMBR(         )
     COMMIT(*NONE)
     OPTION(*SQL *PERIOD)
     TGTRLS(*PRV)
     ALWCPYDTA(*OPTIMIZE)
     CLOSQLCSR(*ENDPGM)
     STATEMENT TEXT CCSID(37)
STATEMENT NAME:  QZ84DC1FE6AC488000
select * from qiws.qcustcdt where lstnam=?
  SQL4021  Access plan last saved on 09/16/02 at 12:47:36.
  SQL4020  Estimated query run time is 1 seconds.
  SQL4027  Access plan was saved with DB2 UDB Symmetric Multiprocessing installed on the system.
  SQL4010  Table scan access for table 1.
  :
```





# Appendix A4

- Extended dynamic access
- The following SQL statements are put into extended dynamic packages:
  - Statements that contain parameter markers
  - INSERT with subselect
  - Positioned UPDATE or DELETE
  - SELECT FOR UPDATE
  - DECLARE PROCEDURE
  - Can force all SELECT statements into package by adding Debug=4 to your SQLDriverConnect connection string (if Debug keyword already exists, add 4 to the value instead of overriding it with 4)

## Appendix A5

- Extended dynamic access
- First time an SQL statement is prepared, the package is created (if it doesn't exist yet)
- Can specify a name and location for package on the data source or let the system do that work
  - Default ODBC SQL package name is created by taken the first 7 characters of the application name and appending 3 letters that are encoding of the package configuration attributes
    - Default package name for Lotus Approach would be: APPROACFBA
    - New setup GUI allows setting of package name for a specific application
  - Default library determined by data source configuration





## Appendix B

- Data compression of ODBC data
  - Can be activated at the connection level or statement level
  - Connection level settings
    - COMPRESSION=1 in SQLDriverConnect connection string OR...
    - SQLSetConnectAttr(hdbc, 2106, 1) OR...
    - “Enable Data Compression” option on ODBC DSN setup GUI
  - Statement level settings
    - SQLSetStmtAttr(hstmt, 2106, 1)
  - Delivered via V4R2 PTF SF49349 and V4R3 PTF SF51501

## Appendix C

- How do you determine which ODBC server job (QZDASOINIT) is the one that you want to analyze?
  - WRKOBJLCK OBJ(userid) OBJTYPE(*usrprf)
  - Returns the QZDASOINIT job servicing your ODBC request
- In V5R1, job information can be obtained by calling SQLGetConnectAttr API with option 2110.





## Sample Code Disclaimer

# Trademarks and Disclaimers


References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

| | |
|---|---|
| AS/400 | IBM |
| AS/400e | IBM (logo) |
| eServer | iSeries |
|  | OS/400 |

Lotus and SmartSuite are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.
Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.
C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.