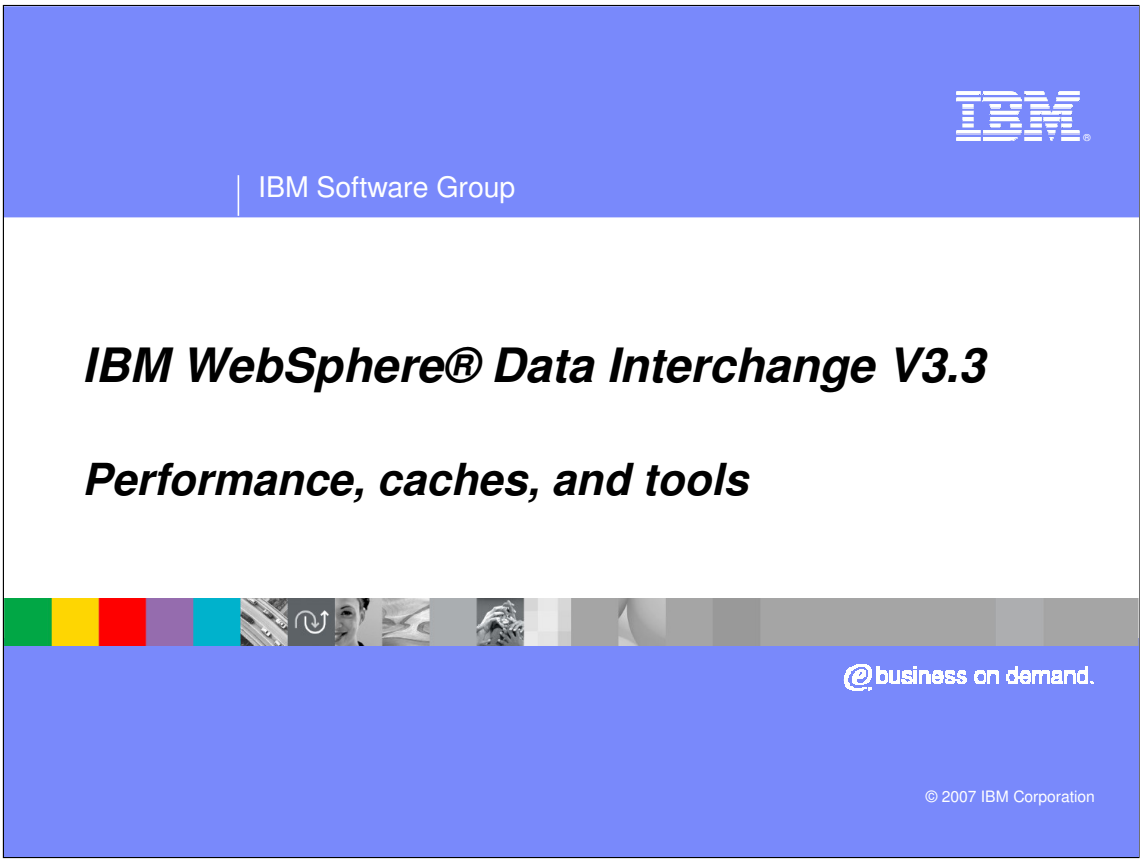


This presentation will identify performance techniques used in WebSphere Data Interchange.

Discussion Objectives

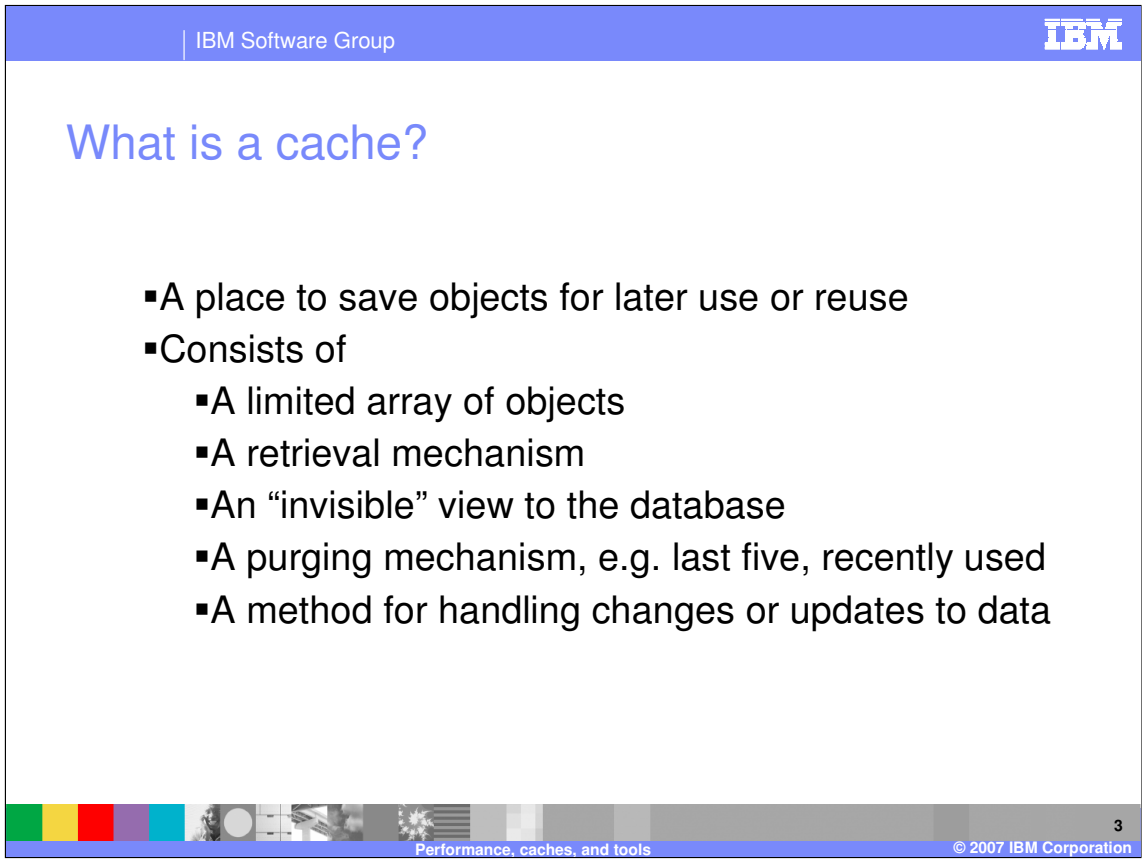
- What is a cache?
- How does WDI use caching?
- What are other performance features of WDI?

WebSphere Data Interchange uses caching and other techniques to improve performance of the translation engines. The objectives of this presentation include answering the questions

What is a cache?

How does WDI use caching? And,

What are other performance features of WDI?



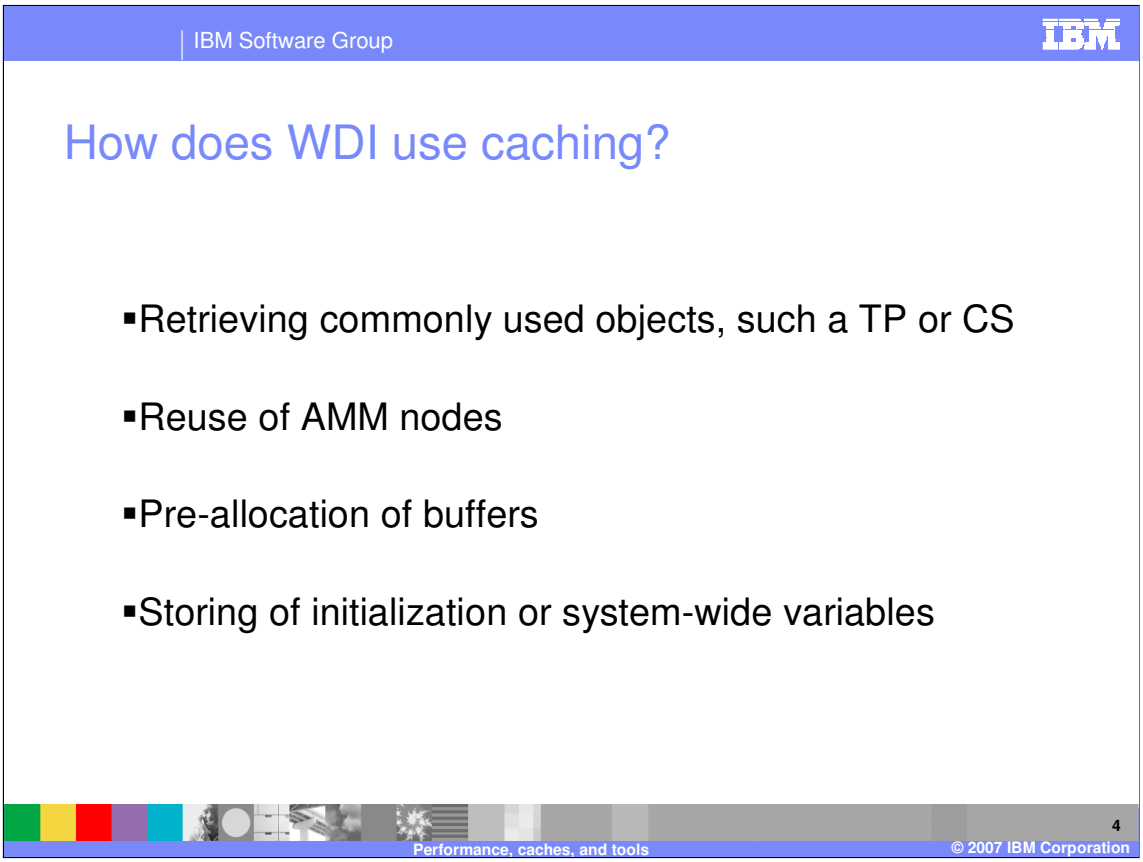
What is a cache?

In early frontier America, hunters and trappers would take several days or weeks to make their rounds of traps. As they caught the animals and prepared the skins, they would store them in caves or underground areas for retrieval during the summer or a time when they were needed for sale.

Computer developers use a storage technique to save objects for later use and most times use the term “cache” for the technique.

Common reasons for using a cache are to save the time required for input / output operations (I/Os). A cache usually has to consider a number of things.

First, the amount of storage that should be devoted to the cache. This usually translates to the number and type of objects to be saved. With WDI this would be Trading Partner profiles, or Map Control Strings. The tradeoff is I/O retrieval time versus main storage utilization. Second, there must be a retrieval mechanism to find the objects in the cache, and thirdly, that mechanism should be invisible to the database. This means that a request for a particular kind of data, e.g. Trading Partner profile, should search the cache, and then the database in one application request within a user exit. Fourth, the limited array should be self-managing and have a criteria for selecting which entry in the array should be removed when a new one is added. It could be a FIFO scheme – first one in is first one out, or a LRU scheme (least recently used). Lastly, cache handling must account for independent updating of objects cached. For instance, if a WDI Client user updates a Trading Partner when it is cached in a Translation execution. The accounting could be as simple as stating a rule that once acquired the object will be used until end of unit of work.



How does WDI use caching?

WDI uses a cache for saving several operational objects. Retrieval mechanisms search the cache first before accessing the database. Ten entries are available for each object type.

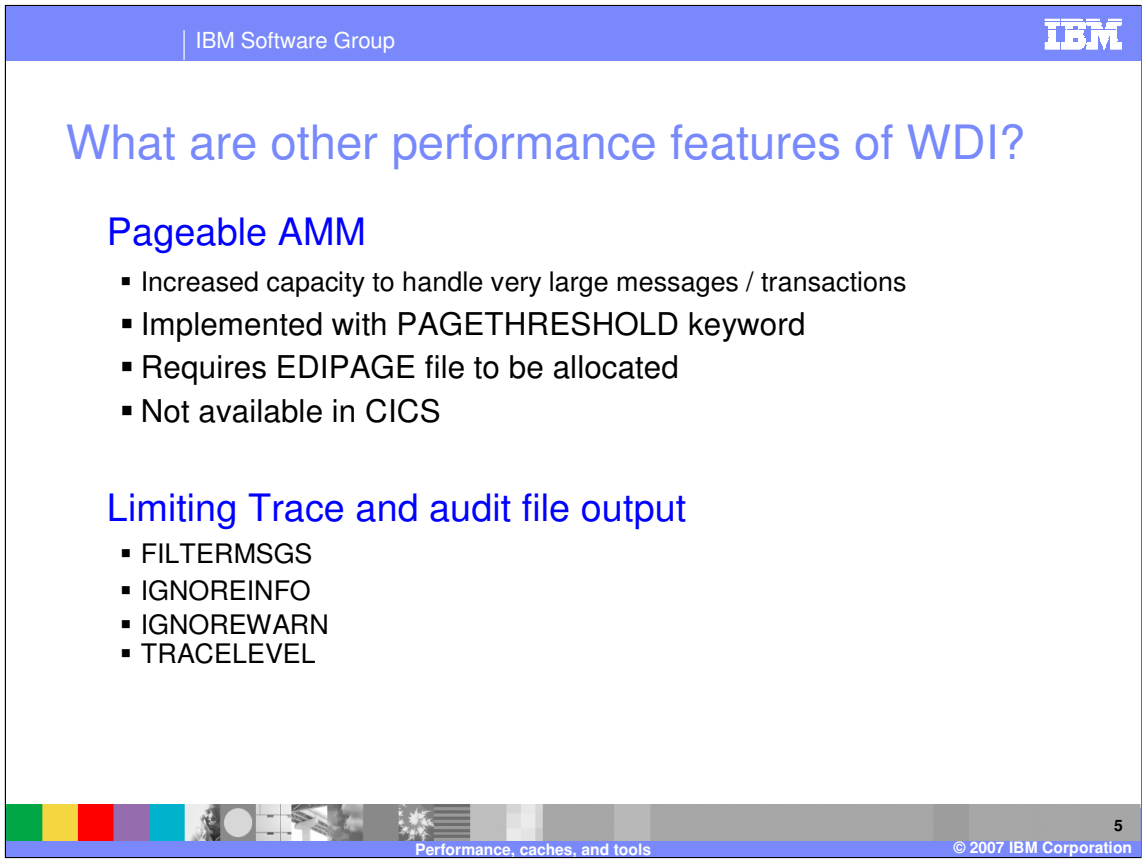
1) Trading Partner profiles
2) Map Control Strings

For S/R translations, a different set of objects are cached.

1) Trading Partner profiles
2) Map Control Strings
3) Map Usages

Another WDI cache is the AMM node pool. This allows subsequent translations to reduce storage acquisitions when constructing the AMM tree.

WDI also pre-allocates buffers for system files and work areas. During WDI startup, storing of initialization or system-wide variables, such as the ALPHANUM Validation table, is also done to keep WDI from repetitively access the database.



The DT Map Pageable AMM feature and the S/R Map Page file are two features that allow WDI to handle very large messages i.e. 500MB. These features write data to DASD when main storage specified limits are reached. Access the data is handled automatically so data is paged in and out as storage is needed.

The "Pageable AMM" (PAMM) is similar to the "paging subsystem" called Pageable Translation in the Send / Receive Translator. This technique "pages out" or writes data that normally resides in memory to a file. Doing so reduces the memory requirement for large EDI translations. The Abstract Message Model or AMM is a WDI internal representation of data during translation. The PAMM will "page out" AMM nodes that are part of repeating data. When there is a repeating loop, element, segment, structure, or record, the AMM subtree containing the repeating data will be paged out of memory when the number of repetitions exceeds a given threshold and the data is no longer immediately needed. The default threshold is 1000. This threshold may be overridden using the PAGETHRESHOLD keyword. PAGETHRESHOLD(0) turns paging off.

The page file is named EDIPAGE. The file must be allocated or the Pageable AMM feature will not be used. If EDIPAGE is not defined, severity-0 message AM0016 will be written to the Print File and processing would continue normally. If an error occurs opening the page file, transformation would continue normally, though severity-0 message AM0015 would be logged. If an error occurs writing to the page file, transformation would continue normally, though severity-4 message AM0015 would be logged.

The EDIPAGE file contains the internal representation of both the input and the output data. Typically, you should allow several times the total of the input and (expected) output file sizes when determining how much space to allocate. This allows for the additional information (name,

What are other performance features of WDI?

EDIWORK for multiple outputs

- Implemented with PAGE(Y) keyword
- User provides workfile named EDIWORK
- Holds output data up to 2 GB
- Not available in CICS

The purpose of this capability is to reduce the amount of memory used during DT translations by "paging out" output data to a work file. The PAGE(Y) keyword on PERFORM TRANSFORM will cause output data to be paged to a work file named EDIWORK once a size threshold is met.
With PAGE(N), output data is collected in a buffer in the Message Broker as translation takes place. When the translation is complete, the output data buffer is passed back to the WDI Utility component where it is written to the user specified output file, as before.

With PAGE(Y) the Message Broker will write the output data to a work file after a size threshold is met. The workfile offset is passed back to the Utility, where the data is read from the work file, and written to the user specified output file, as before.

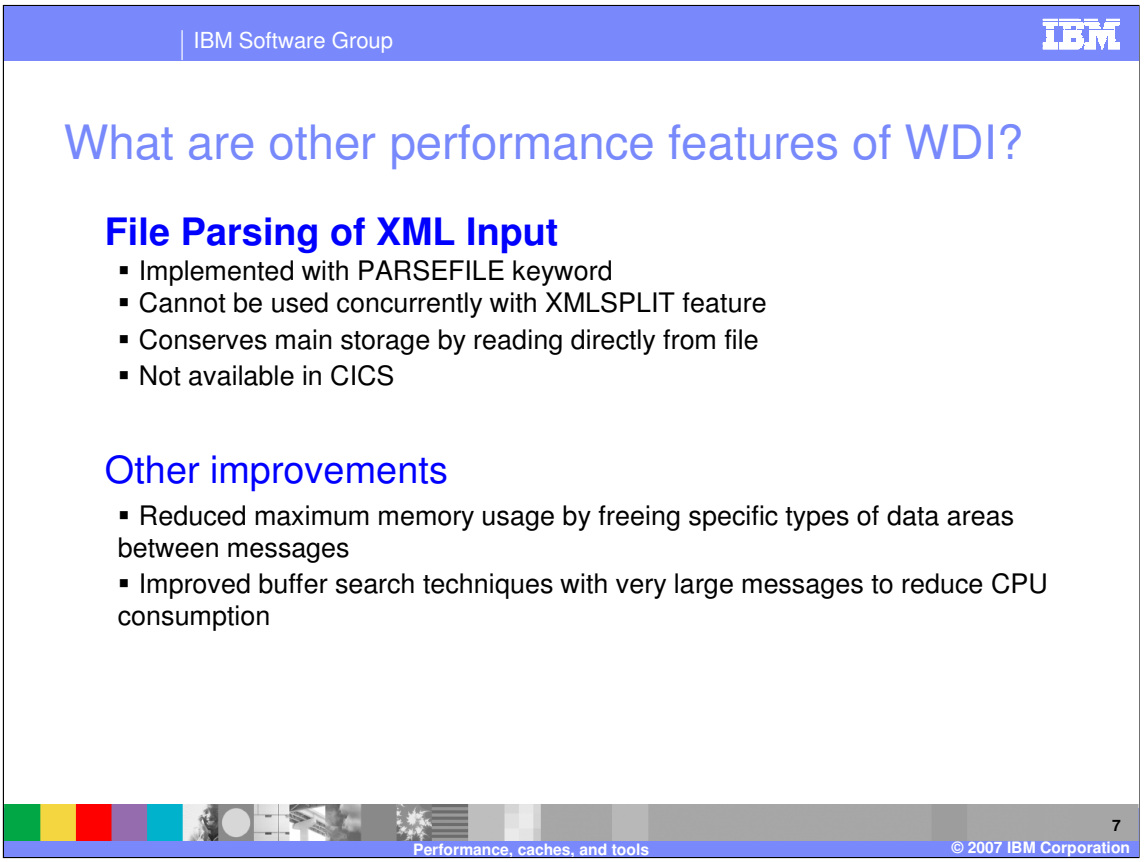
There are two places in the Message Broker that check the output size threshold in order to possibly page the output. If a single output document reaches 10 MB, then the document will be released from memory and written to the work file. If a collection of smaller output documents reach 10 MB, then any remaining output documents are written to the work file.

If PAGE(Y) is specified on a PERFORM TRANSFORM command and file EDIWORK is not defined, not allocated, or if there is a problem opening the file, a severity-4 message, MB0003 or MB0004, will be logged in the print file. In this case, processing would continue as if PAGE(N) was specified. If an error occurs writing to EDIWORK, then a severity-8 message, MB0005, will be logged and the translation will terminate.

The type of output data that may potentially be paged is not restricted. PAGE(Y) applies to ADF, EDI, and XML output data. There is a size restriction on the amount of data that can be paged to EDIWORK. The current restriction is 2 GB on z/OS. If this size is exceeded, a severity-8 message, MB0005, will be logged and the translation will terminate.

On z/OS, because EDIWORK may be reused each time the Utility calls the Message Broker (which occurs multiple times during a PERFORM command), EDIWORK cannot be defined as DISP=MOD or as an append-to-end-of-file file. The base EDIWORK offset is reset to zero each time the Utility calls the Message Broker.

PAGE(Y) is not valid in CICS on PERFORM TRANSFORM commands.



A PERFORM TRANSFORM keyword named, PARSEFILE, will allow XML data to be parsed directly from the input file. Using this keyword can significantly reduce the memory needed for processing a large XML input file. Without this feature, WDI reads the XML document into a buffer, so if a 500 MB XML document is being processed, then WDI will use at least 500 MB to parse the data. With this option, the WDI parsing reads from the file and requires much less memory.

PARSEFILE(Y) means parse-from-file. The default is PARSEFILE(N), which means parse-from-buffer. PARSEFILE is only valid for XML input data, and is not valid in CICS. If PARSEFILE(Y) is specified on a PERFORM TRANSFORM command, SYNTAX must be X, INFILE must exist and contain the name of the XML input file, and INTYPE must not have a value.

There is a restriction when using PARSEFILE(Y). WDI's XML split capability is not supported. This means that PARSEFILE(Y) should only be specified when the input file contains one and only one XML document. More than one document in the file will result in a fatal error in the parser.

Restrictions: The PARSEFILE technique does not apply to ROD/ADF data or EDI data.

Other performance improvements for WDI are freeing of data message buffers so that after very large messages are processed some memory can be recovered, and also buffer search techniques which reduce the CPU usage with large messages.

Summary

- Performance is normally measured by throughput which is a tradeoff among CPU usage, file accesses, and memory usage
- WDI uses many techniques to make it a high performance translator

Performance is normally measured by throughput. Throughput is normally a tradeoff among CPU usage, file accesses, and memory usage. Storing objects in memory reduces the number of relatively slower file accesses but increases memory consumption. Reducing the wait on file access can decrease elapsed time but increase CPU time.

WDI uses many techniques to make it a high performance translator

Trademarks, copyrights, and disclaimers

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

IBM	CICS	IMS	WMQ	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. 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.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.