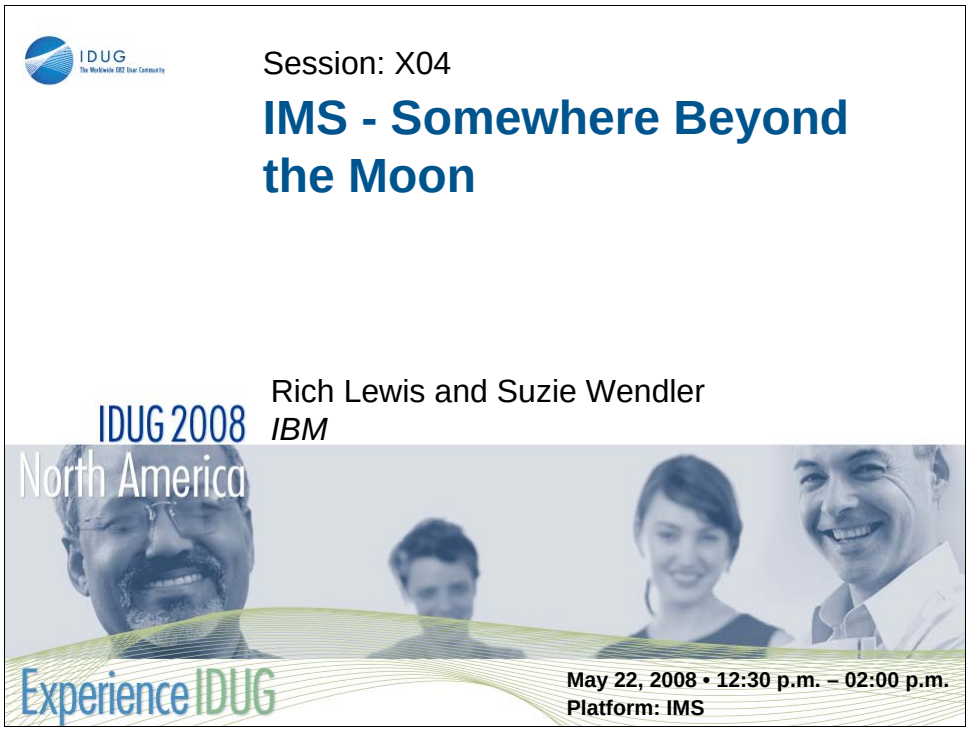


IMS was introduced in 1968. It was originally designed as part of the Apollo Moon project. It has evolved to meet the changing demands of a diverse set of users for 40 years. Nevertheless, application programs written in the early days of IMS are still being used. Many of these programs are doing things that those who wrote them did not imagine. This session will discuss how IMS has evolved and continues to evolve. Illustrations will be from the personal experiences of the presenters.

IMS - Somewhere Beyond the Moon

- Learn how IMS has evolved since its introduction 40 years ago
- Learn how the evolution of IMS has matched the changing environments of its users
- Understand how IMS is enabled in today's SOA world
- Gain insight into the future of IMS
- Celebrate IMS's 40th anniversary

Topics

- Origins of IMS
- Release History
- Characteristics of IMS
  - How they influenced its growth
  - Some illustrations
- Future of IMS

Protecting and enhancing your investment in IMS

Origins of IMS

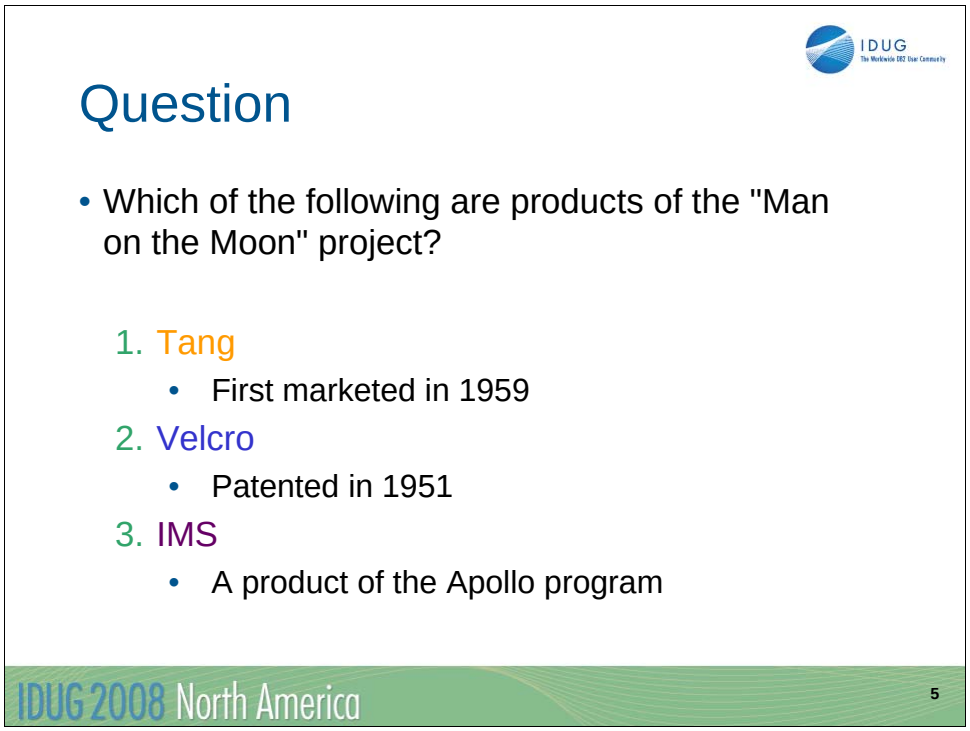
40th Birthday!

- IMS/360
  - Announced in 1968; delivered in 1969
  - Result of joint project by IBM, Rockwell, and Caterpillar
    - Rockwell was contractor for NASA Apollo project
      - Needed to track parts for the Apollo capsule
      - Built on 1965 IBM/Rockwell project using IBM 7010
    - Caterpillar
      - Needed to track parts for tractors and other products
  - Written for the IBM S/360
    - Rockwell implemented on S/360 Model 65
      - This was a very large machine for its day

IMS/360 was announced in 1968 and delivered in 1969.

It was the result of a joint project of IBM, Rockwell, and Caterpillar. Rockwell was a contractor for the NASA Apollo project. Rockwell and IBM has started a project on the IBM 7010 computer. This was rewritten for the S/360 system. Caterpillar joined the project. Both Caterpillar and Rockwell needed an online system for tracking parts.

The first IMS system at Rockwell was implemented on a S/360 Model 65. This was a large machine for its day.



You might have heard that Tang and Velcro are two products of the "space age." Some have claimed that they were produced for NASA projects. Actually, both were developed before NASA. Nevertheless, they are associated with NASA because they were used by the Apollo project.

IMS was developed as a direct result of the Apollo program.

Original Environment

- S/360 and OS/360
- Machines speeds: 200K to 800K instructions/second
- Machine memory: 256K to 512K
  - All memory was real
- 2311 Disk
  - 200 cylinders, 5 tracks/cylinder
  - 7,250,000 bytes per volume
    - 4GB would have required about 600 volumes!

It interesting to see what kind of systems were used by the first IMS installations. IMS/360 was implemented on S/360 hardware using the OS/360 operating system. Typical machine speeds were in the range of 200K to 800K instructions per second. Machine memory was real. This was before the implementation of virtual memory.

The IBM 2311 disk was the typical DASD used by early S/360 systems. These disks had 200 cylinders with 5 tracks per cylinder. A volume could hold approximately 7,250,000 bytes. Compare that with a modern IMS database data set. Many users have 4GB data sets. Remember OSAM data sets may be up to 8GB.

Think about your PC or laptop. What is its disk capacity? Few systems are sold today with less than 80GB disks!

IMS Runs the World

- Manufacturing
  - Aerospace, equipment, automobile, …
- Banking
- Telecommunications
- Transportation
- Insurance
- Finance/Credit Card Processing
- Package delivery
- Government
- Retail
- …










The IMS slogan is, "IMS Runs the World." It's true. IMS systems are used in many industries and organizations as a vital part of their operations. We will see how this came about.

History of Releases

- IMS/360
  - 1.0 Announced 1968; GA 1969
    - Multi-region architecture
      - Applications ran in separate regions
    - Scheduling by intent on entire DB
  - 2.0 GA 1971
    - Scheduling by intent on segment type
  - 2.3.0
  - 2.4.1 GA 1972
    - MFS, 3270s

The next few slides has a partial history of IMS releases and their major content.

You may notice that several IMS/360 releases are missing from this slide. I (Rich Lewis) began working with IMS 2.3.0. I don't know the exact content of releases between 2.0 and 2.3.0.

History of Releases

- IMS/VS - Virtual Storage
  - 1.0.1 1972
    - VSAM
    - Secondary indexing
    - Program Isolation
  - 1.1.1
    - Parallel DLI
    - GSAM and application checkpoint/restart
    - VSAM buffer pools

Parallel DLI is the parallel processing of multiple DLI calls. Before IMS/360 1.1.1 processing of DLI calls was serialized.

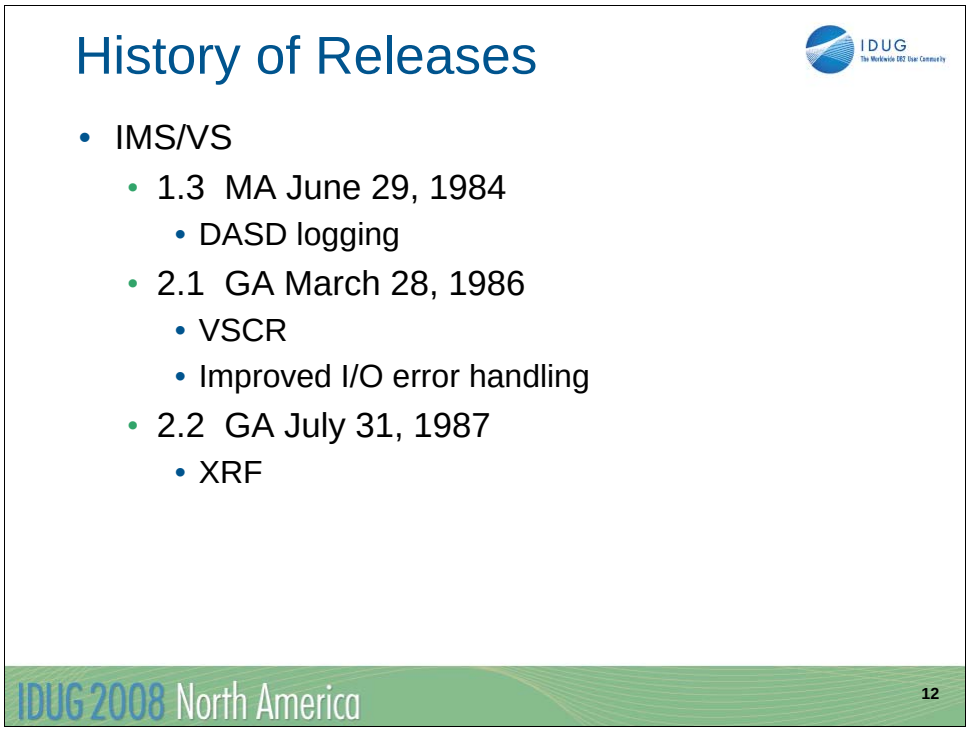
History of Releases

- IMS/VS
  - 1.1.3 GA 1975
    - MSC
    - SMP
  - 1.1.4 GA April 29, 1977
    - Fast Path
    - DBRC
  - 1.1.5 GA Nov. 8, 1978
    - Fast Path mixed mode
    - AOI and CMD call

Fast Path mixed mode is the processing of Fast Path databases from MPP or BMP regions which also have access to full function databases or the processing of full function databases from IFP regions which also have access to Fast Path databases.

History of Releases

- IMS/VS
  - 1.1.6 GA July 25, 1980
    - 31 dependent regions
    - ISC
    - MSC VTAM
  - 1.2 GA May 15, 1981
    - Data sharing (BLDS GA July 12, 1982)
    - IRLM
    - PROCOPT=GON/GOT



Before IMS/VS 1.3 IMS online systems logged to tape.

Improved I/O handling includes keeping copies of blocks or CIs in memory when they cannot be written to DASD due to a write error. These blocks and CIs remain available to the IMS system. The writes are retried and if successful, the blocks and CIs are no longer kept in memory. This handling avoids database recoveries in many cases.

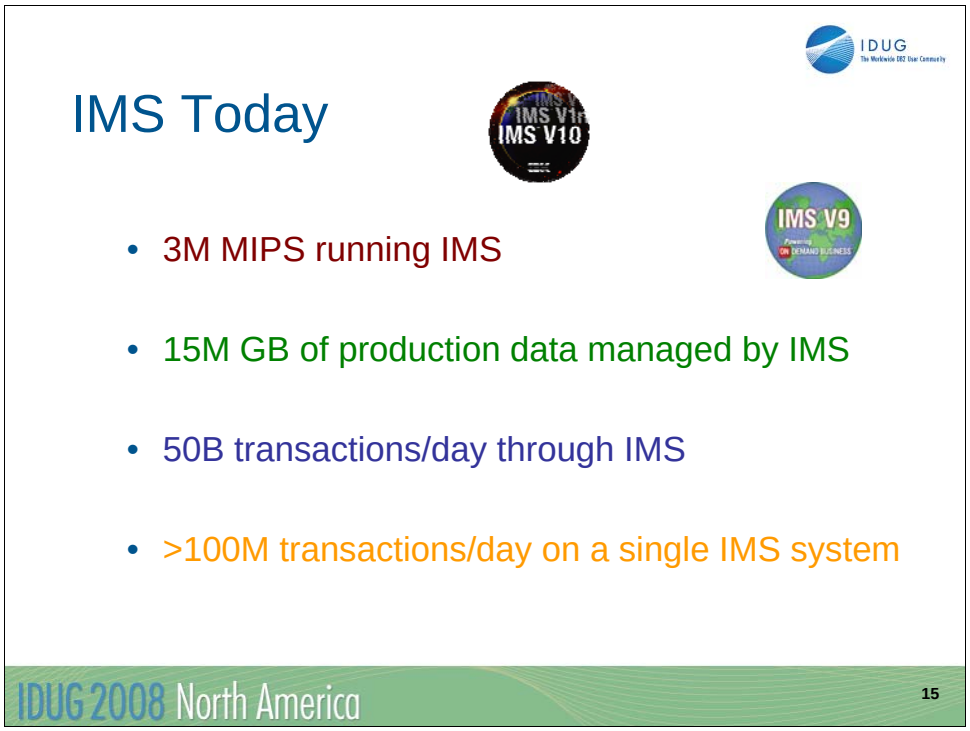
XRF (extended recovery facility) is a "hot standby" system. The XRF alternate monitors the active system and takes over its work if the active fails.

History of Releases

- IMS/ESA
  - 3.1 GA Dec. 15, 1989
    - DBCTL
  - 4.1 GA May 28, 1993
    - APPC
    - ETO
  - 5.1 GA 1995
    - Parallel Sysplex data sharing
    - OTMA
  - 6.1 GA Oct. 30, 1998
    - Shared queues
    - ODBA

History of Releases

- IMS
  - 7.1 GA Oct. 27, 2000
    - HALDB
  - 8.1 GA Oct. 25, 2002
    - Common Service Layer (CSL)
    - Global online change
  - 9.1 GA Oct. 29, 2004
    - HALDB OLR
    - Integrated IMS Connect
  - 10 GA Oct. 26, 2007
    - DRD



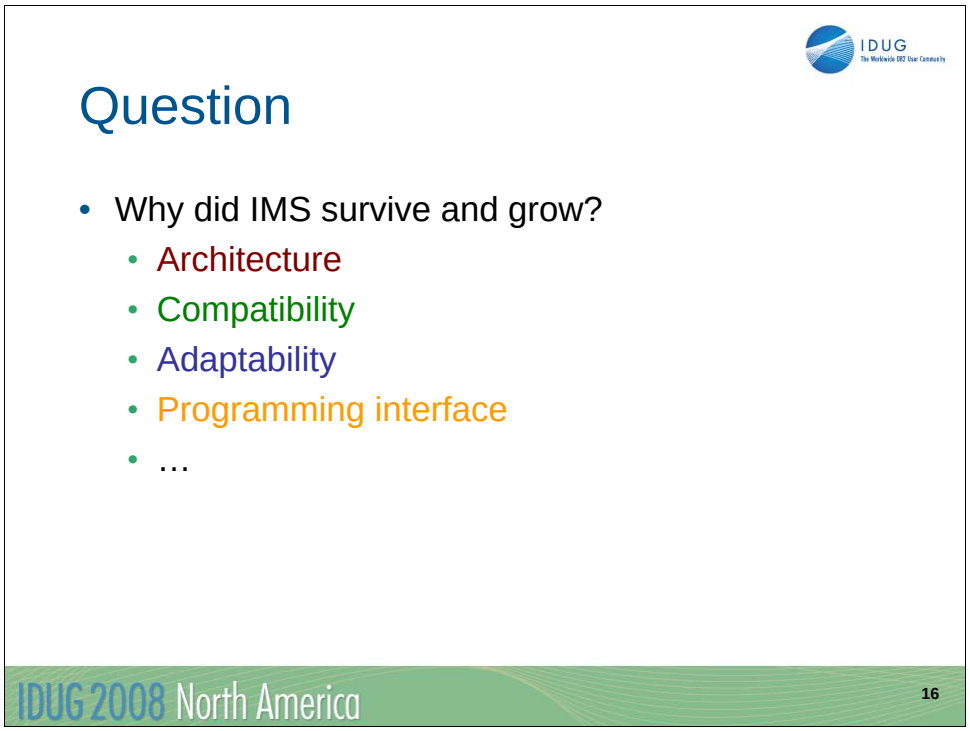
Remember the typical environment used when IMS/360 was introduced? IMS and the world it runs in has changed.

Today, there are approximately 3,000,000,000,000 instructions per second executing IMS.

IMS production databases occupy about 15GM of disk space.

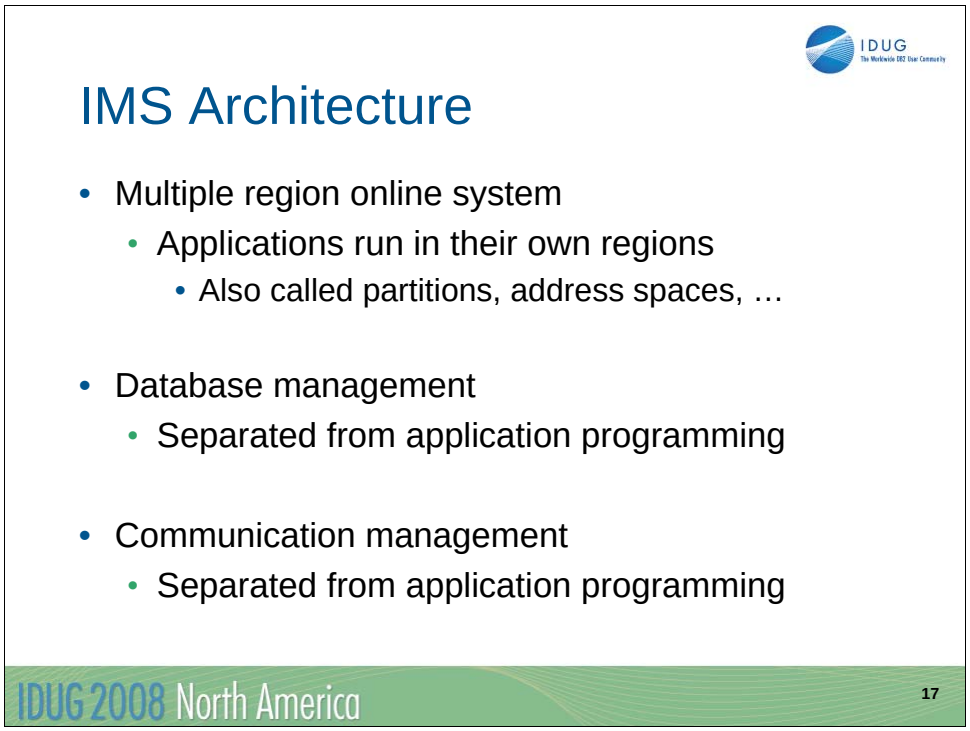
Approximately 50,000,000,0000 IMS transactions are processed every day.

A single IMS system has processed over 100,000,000 transactions in a day.



Why has IMS been able to survive and grow when the world in which it exists has changed so dramatically. That's what we will discuss next.

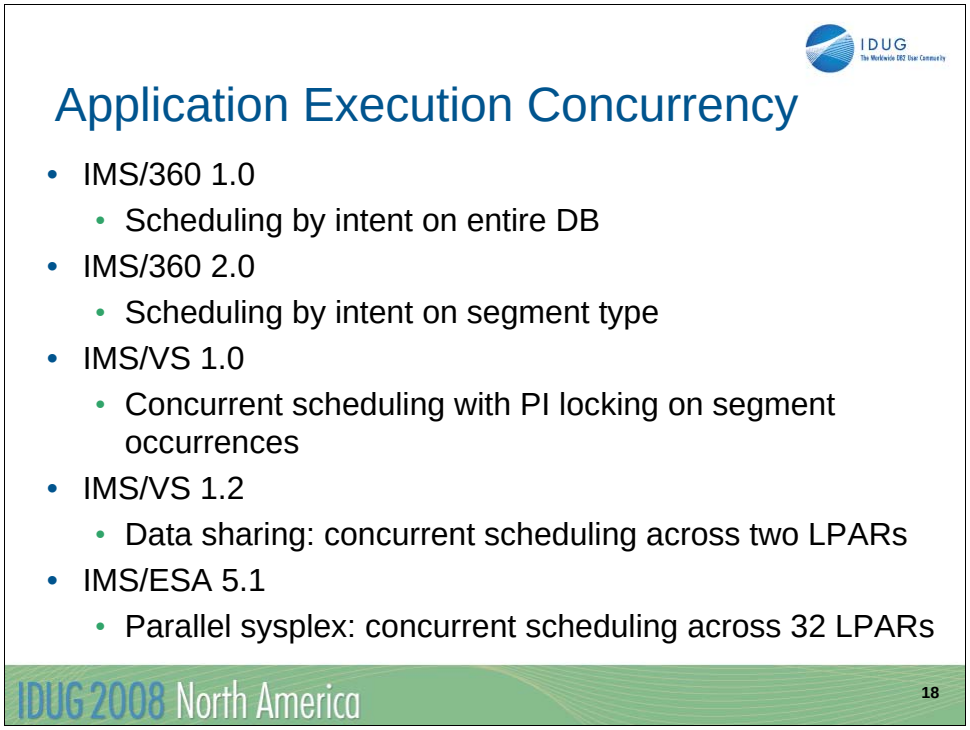
The short answer is that IMS has grown because of its basic architecture, its commitment to compatibility, its adaptability, and the programming interface that was introduced in the beginning. You may have other reasons that you would suggest.



IMS has had a multiple region online system from the beginning. IMS online application programs run in their own partition, region, or address space. The exact implementation has changed as the operating systems have changed, but the fundamental idea has remained the same. This was a very advanced idea in 1968.

One of the most fundamental goals of the original IMS system was to separate the definition of the database (data sets and files) from the application program. This was not typical in the 1960's. In those days, a file was typically created by one program and read by one other program. The definition of the file was in the programs. IMS changed that. The developers of IMS realized that moving the definitions out of the program would have substantial benefits when many programs would access the same database.

A similar idea was applied to the communications. The sending and receiving of input and output messages was removed from programs. IMS's queues were critical to its initial design.



One way of looking at expanding capacity is to consider how many application programs can access a database concurrently.

In IMS/360 1.0 if an application program had update capability for a database, no other program with access to the database would be scheduled concurrently. This was very restrictive. On the other hand, in those days most updates were done by batch programs, not online transactions.

IMS/360 2.0 introduced scheduling by intent on segment types, not databases. Programs which updated different segments in the database could process concurrently.

IMS/VS 1.0 introduced Program Isolation locking and concurrent scheduling of programs with update intent for the same segment types. PI locking prevented them from updating the same segment occurrence, but allowed them to run concurrently.

IMS/VS 1.2 introduced data sharing. This allowed programs in IMS systems on different machines to update the same database at the same time. This was limited to two machines.

IMS/ESA 5.1 introduced Parallel Sysplex N-way data sharing. IMS systems on up to 32 LPARs could concurrently update the same databases.

IMS System Growth

Number of concurrent application programs

IMS release		# Dep. Regions	Multiple System Support
IMS/360 V1	1969	15	
IMS/VS 1.1.3	1975		MSC
IMS/VS 1.1.6	1980	31	ISC (including CICS)
IMS/VS 1.2	1981		2-way Data Sharing
IMS/VS 1.3	1984	255	
IMS/ESA 5.1	1995		N-way Data Sharing 255 IMS systems on 32 LPARs
IMS/ESA 6.1	1998	999	

This is another way of looking at expanding capacity. This shows how the number of concurrently executing application programs has grown. IMS/360 supported up to 15 concurrently executing application programs. That is, it supported up to 15 dependent regions.

IMS/VS 1.1.3 introduced MSC. This allowed users to connect different IMS systems and ship transactions between them. Each of these systems could have up to 15 dependent regions.

IMS/VS 1.1.6 introduced ISC. This was another way to ship transactions between IMS systems. It also allowed IMS to send transactions to CICS and CICS to send transactions to IMS.

IMS/VS 1.1.6 also expanded the number of dependent regions in an IMS system to 31.

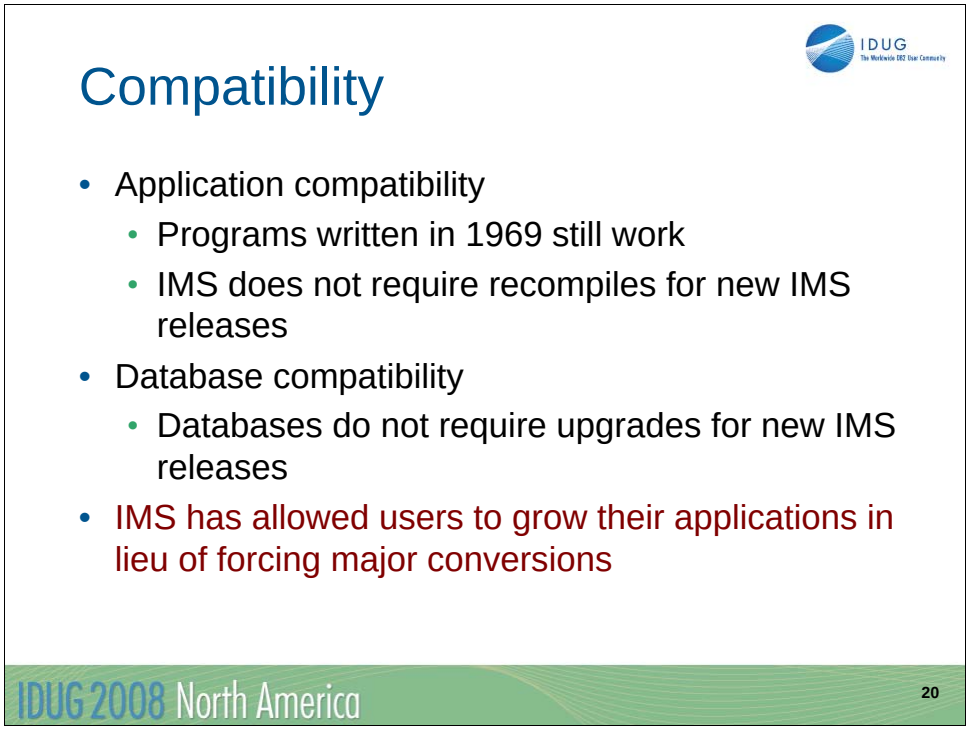
IMS/VS 1.2 introduced 2-way data sharing. This doubled the number of dependent regions that could be accessing the same databases.

IMS/VS 1.3 expanded the number of dependent regions to 255.

IMS/ESA 5.1 introduced N-way data sharing. This greatly increased the potential number of applications running concurrently.

IMS/ESA 6.1 expanded the number of dependent regions to 999.

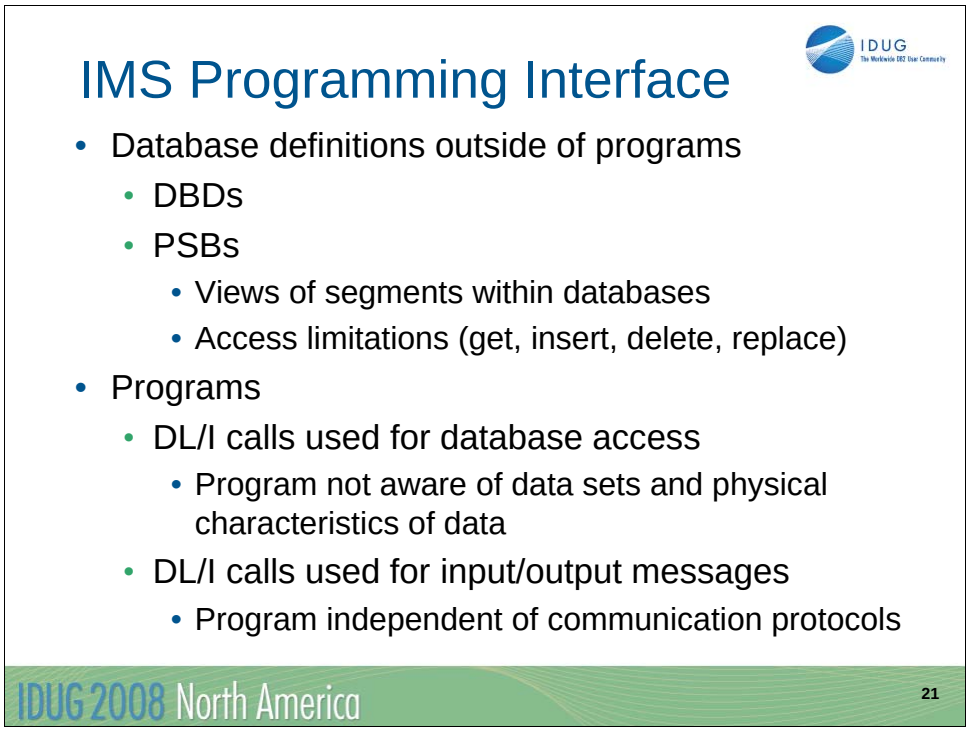
Theoretically, you could have 255 systems, each with 999 dependent regions. That's 254,745 dependent regions as opposed to 15 in 1969.



A major reason for the success of IMS has been its commitment to compatibility. Applications written to the original interface still work. Interestingly, IMS does not even require its users to recompile or rebind (link-edit) programs when migrating to new releases.

Similarly, IMS is committed to database compatibility. When you migrate to a new release of IMS, the databases do not have to be converted. They are usable in their old form.

This commitment to compatibility makes IMS users more productive. They can migrate to new IMS releases which take advantage of new hardware and software capabilities without spending time on application or database upgrades.

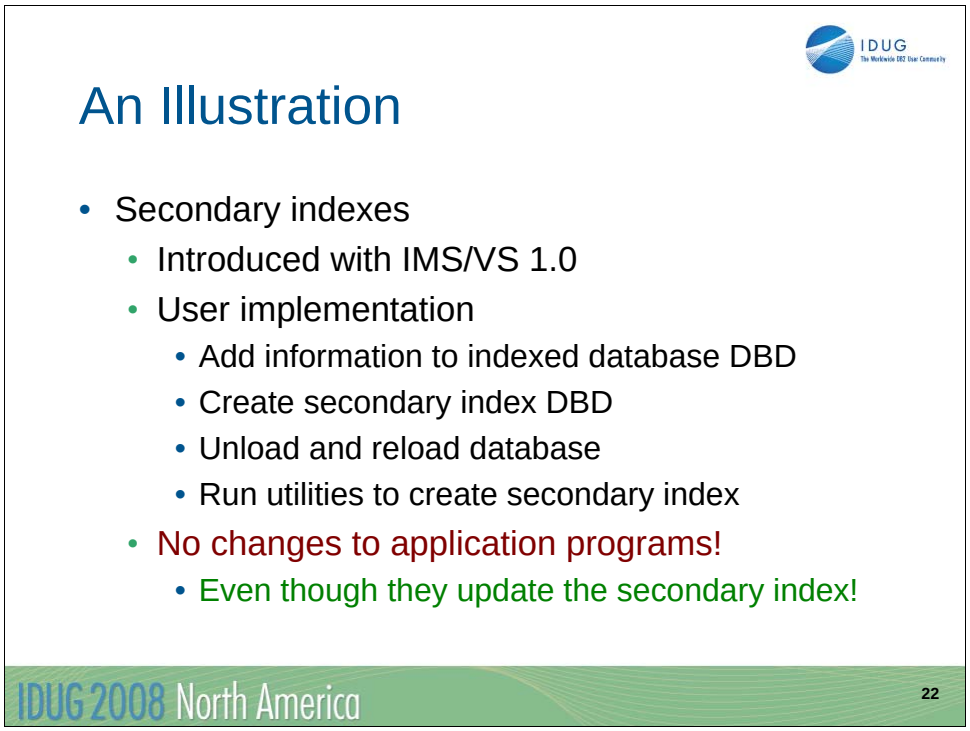


The IMS programming interface is simple.

Database definitions are not in programs. Instead, they are in DBDs. Application views of data and access limitations are in PSBs. This eliminates many potential requirements for updates to application programs.

To access IMS databases, programs issue DL/I calls. The program does not have to be aware of the physical characteristics of a database. It may be one to 10 data set groups. It may be broken into hundreds of Fast Path areas or HALDB partitions. The application does not have to be aware of this. If the data set groups, areas, or partitions are changed, the program is not affected.

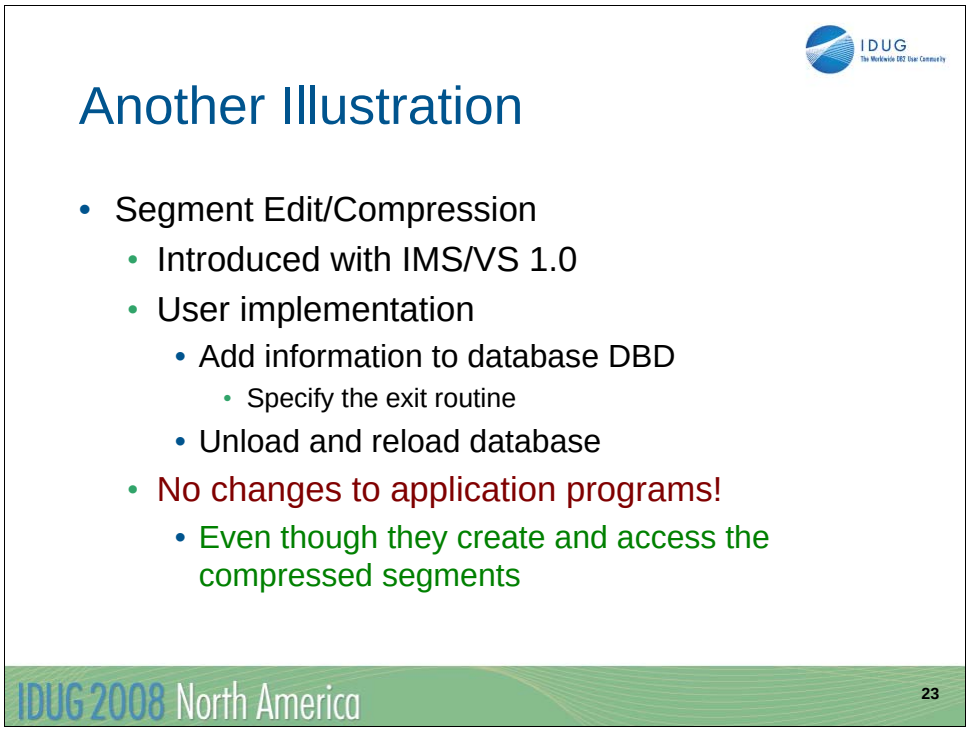
Similarly, input and output messages are retrieved and created with DL/I calls. Applications do not deal with communication protocols. Over the years these have changed from BTAM, to VTAM, to TCP/IP. Data streams have changed from 3270 to XML. These changes have been made without application program changes.



The next several slides have illustrations of the advances made in IMS with application program changes.

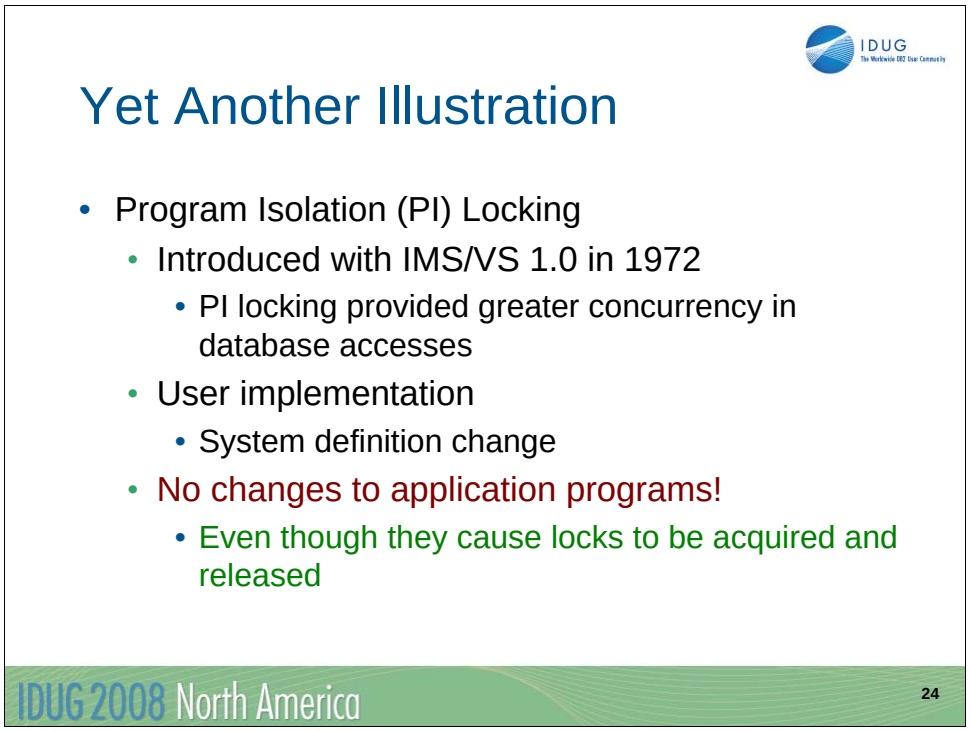
I (Rich Lewis) remember this well, even though it happened over 30 years ago. Secondary indexes were introduced in IMS/VS 1.0. This was an important advance. We could have more than one index into a database. Of course, we wrote new programs to use the new indexes. Interestingly, we did not have to write new programs to maintain the new secondary indexes. Updates to the databases which were being done by programs written before the introduction of secondary indexes did that. All we had to do was to implement the secondary indexes. This was done by unloading the database, defining the new secondary indexes, reloading the database, and running the utilities to create the secondary indexes from the work files that the reload produced.

We got a significant new application capability without having to write programs to maintain the new indexes.



IMS/VS 1.0 also introduced the segment edit/compression exit routine. This allowed us to have larger databases with using more disk space. Remember that disk space was very expensive 30 years ago.

Once again, we did not have to change application programs. All we had to do was unload the database, redefine the database with the new exit routine, and reload the database. As you are probably aware, IMS supplies a good compression exit routine with the product. If you want to get even better compression you can buy a separate product.

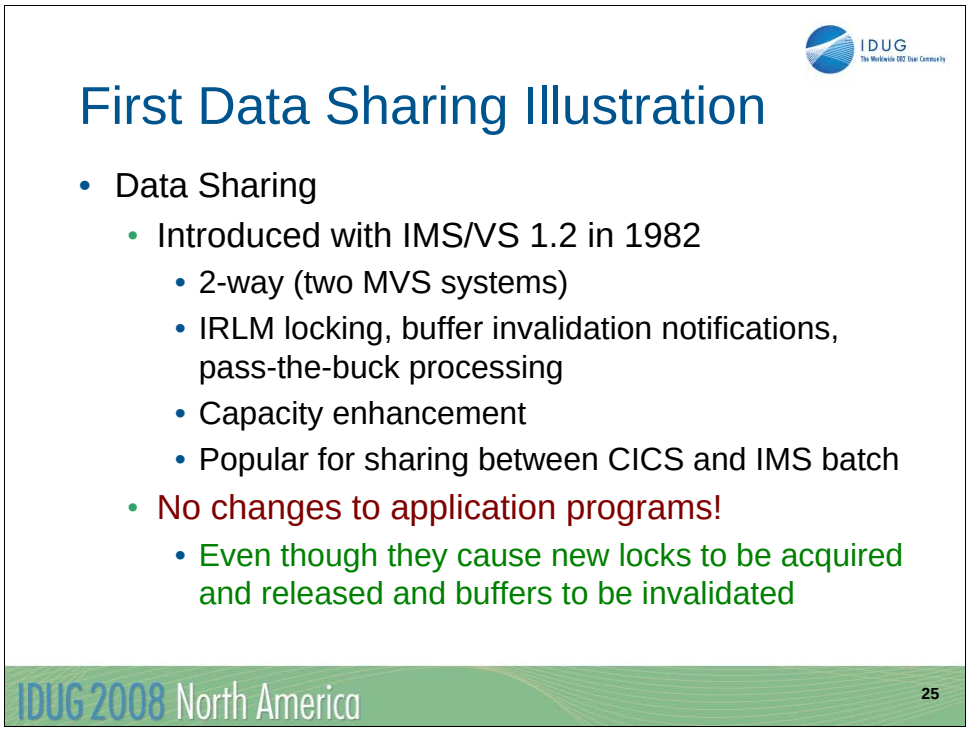


IMS/VS 1.0 was an important IMS release. This is another illustration from that release. It introduced Program Isolation locking. As mentioned earlier this provided greater concurrency in database accesses.

This was even easier to implement. We did not have to do anything to application programs or databases. At that time, PI was a option in the system definition.

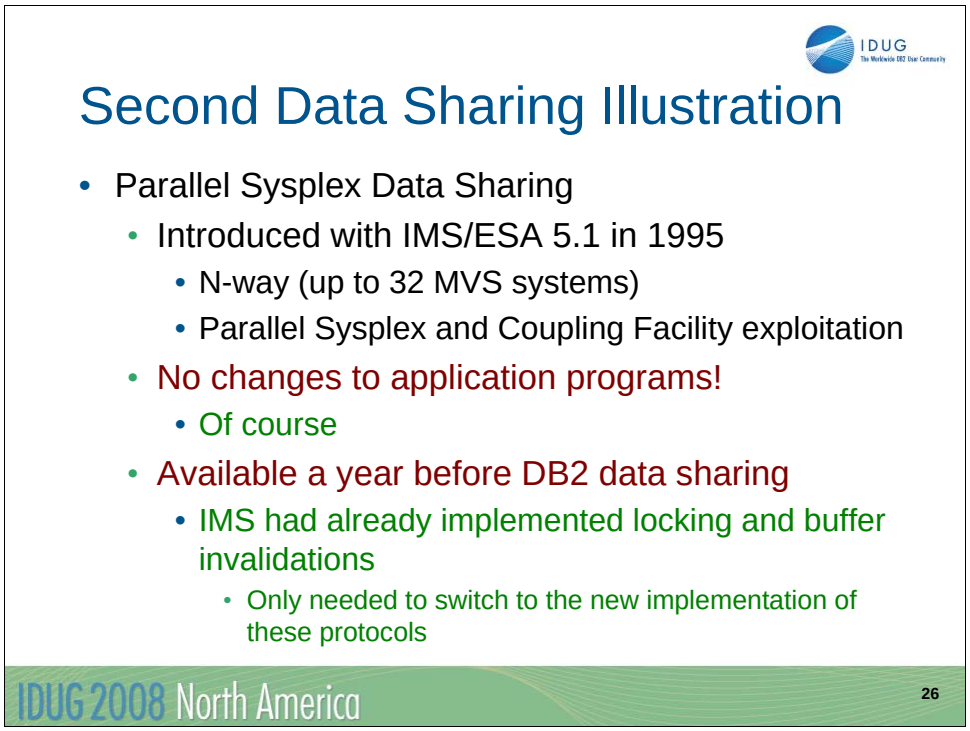
The implementation of PI changed a lot of IMS processing. It now got locks, detected deadlocks, and invoked backout. Nevertheless, we didn't have to make any application changes to implement it.

You may be thinking that we could get a lot of lock conflicts if we weren't careful. That's true, but it was still better than segment intent scheduling. Before PI if a program had update intent for a segment, no other program with access to that segment could schedule.



Data sharing was introduced by IMS/VS 1.2. This was a significant capacity enhancement to IMS. We could apply more that one system to the workload for a set of databases. This implementation of data sharing used two instances of the IRLM for locking and buffer invalidation notifications. The two IRLMs communicated with each other via VTAM using "pass the buck" processing. Data sharing was also important to CICS users with IMS databases. It provided better performance for batch jobs running concurrently with CICS than CICS's "shared DB".

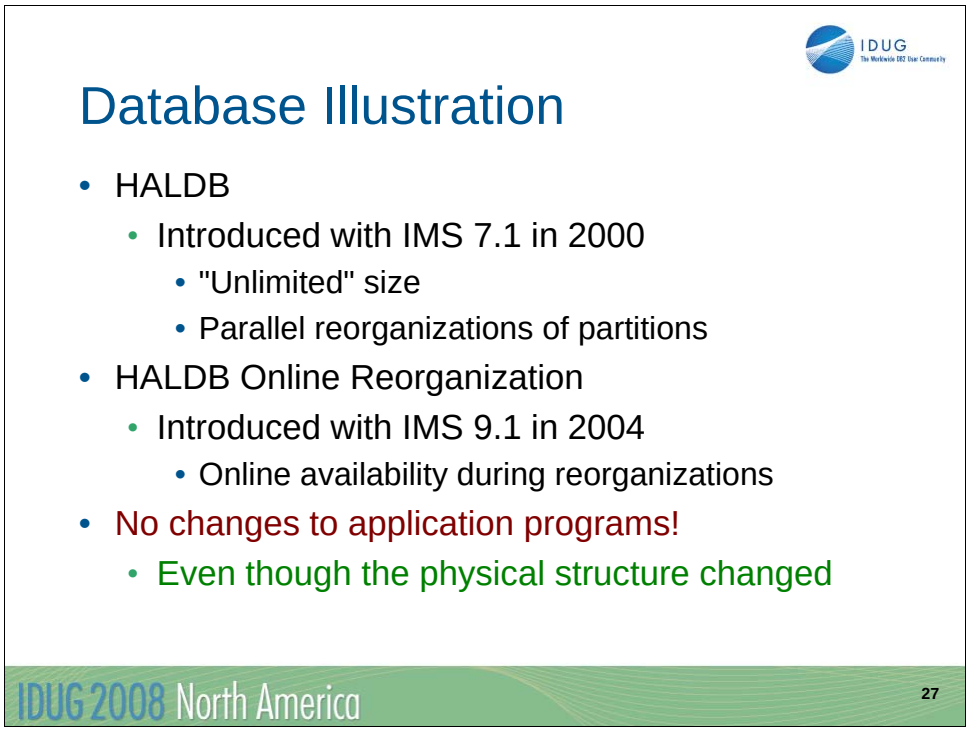
Once again, we did not have to make application changes to implement data sharing. The system took care of the protocols to ensure integrity. This included new kinds of locks and the invalidations of buffers holding blocks that were updated in another IMS system.



Data sharing was enhanced by IMS/ESA 5.1. It implemented Parallel Sysplex facilities including Coupling Facility structures. This expanded data sharing from 2-way to N-way. N-way means up to 255 IMS systems on up to 32 LPARs.

Once again there were no application program changes.

IMS implemented Parallel Sysplex data sharing a year before DB2 implemented it. That was to be expected. IMS had included the acquiring and releasing of locks, the invalidation of buffers, and other data sharing protocols in its code in 1982. Parallel Sysplex just modified this a bit. IMS only had to change what code it called at these points. DB2 had a bigger job. They had to implement data sharing for the first time.



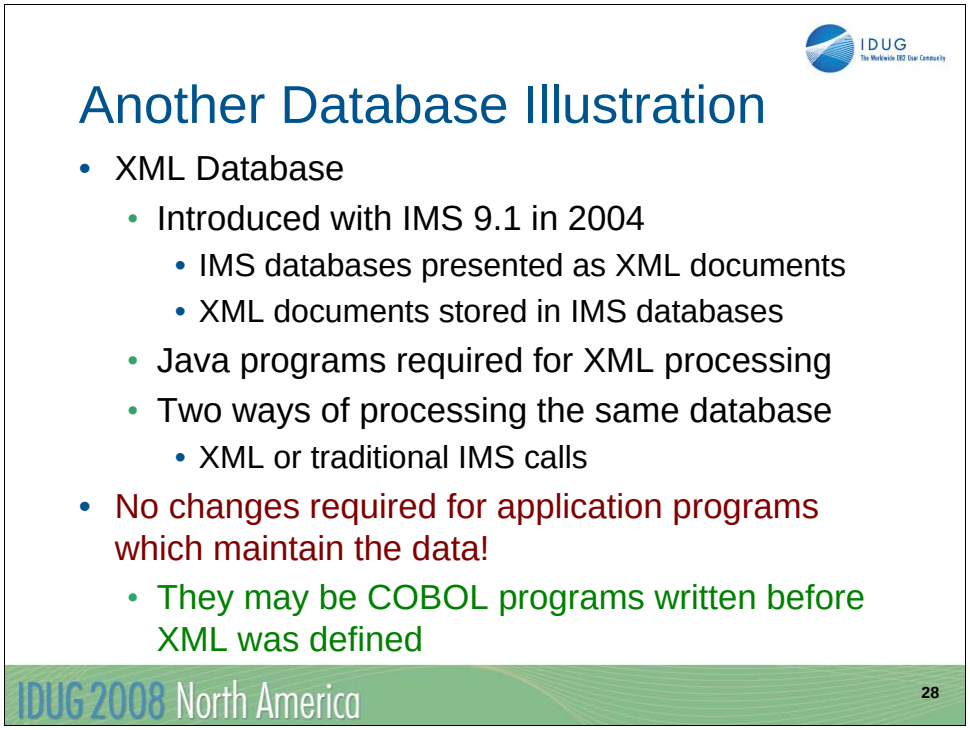
Let's move to something a bit more recent.

HALDB was introduced in IMS 7.1. HALDB allows IMS full function databases to be almost unlimited in size. A database can have up to 1001 partitions. Each partition could have 10 data set groups. HALDB also allows the partitions to be processed in parallel. This is especially important for reducing the elapsed time of database reorganizations.

IMS 9.1 extended HALDB to include HALDB Online Reorganization.

I guess you get the idea by now. No application changes were required for HALDB. This is true even though the underlying physical structure has changed.

There may be a few exceptions for application programs. If you have a secondary index with duplicate data, you use the system related /SX field to provide VSAM key uniqueness, and you process the secondary index as a database, not just use it as an index, you have to make a small modification to your program. That typically is a very, very small number of programs.

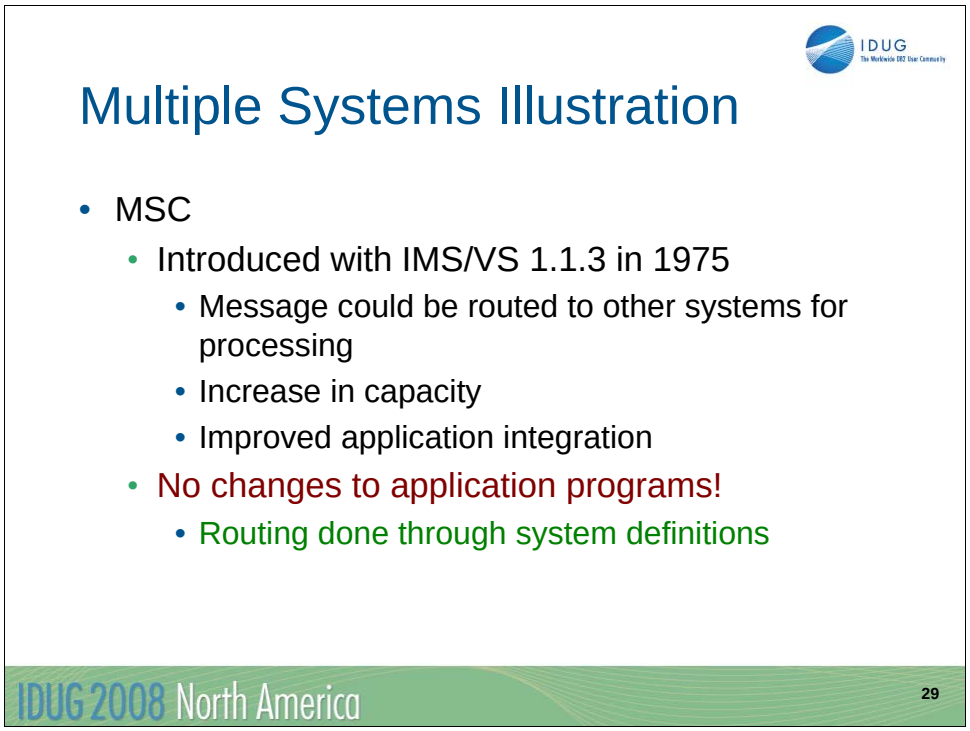


Here is something even more recent.

XML Database was introduced in IMS 9.1. It allows users to present data from IMS databases as XML documents. It also allows users to store XML documents in IMS databases. These databases could be databases that were defined before XML was conceived. They might be databases that were developed in the early days of IMS.

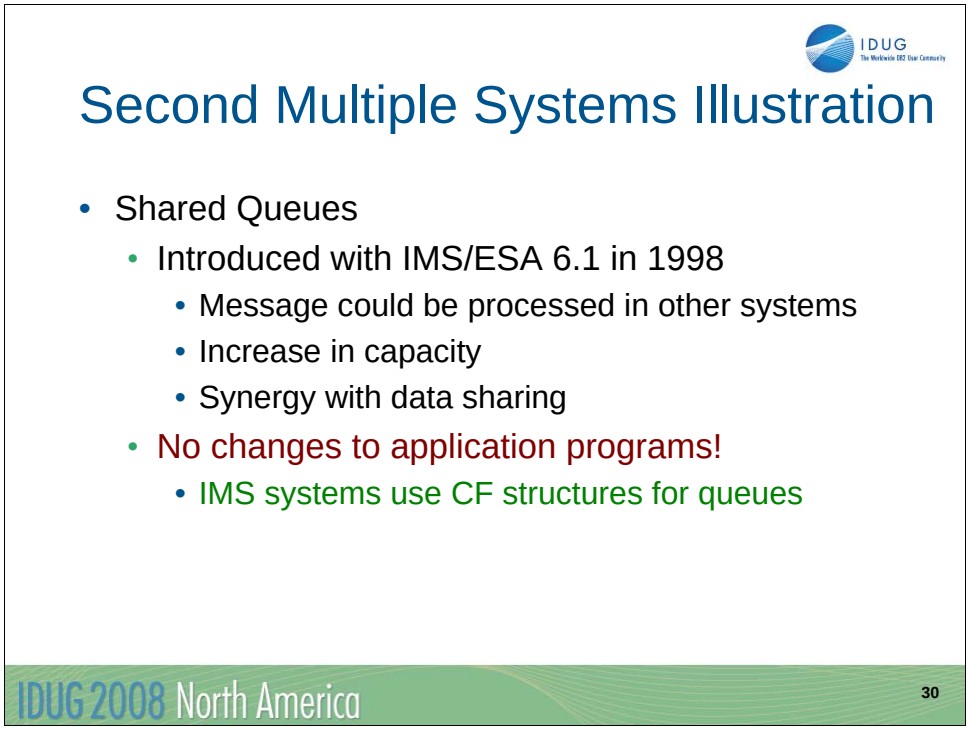
XML Database is provided for Java programs. This means you need new programs to process XML. That's reasonable since XML is a fairly recent development. Another way to look at this is that databases can be viewed either in the traditional IMS ways or as XML documents. That's a real compatibility advantage.

This is not all that XML database can do. You can define new databases for the storage and retrieval of XML documents. That would depend on your application requirements.



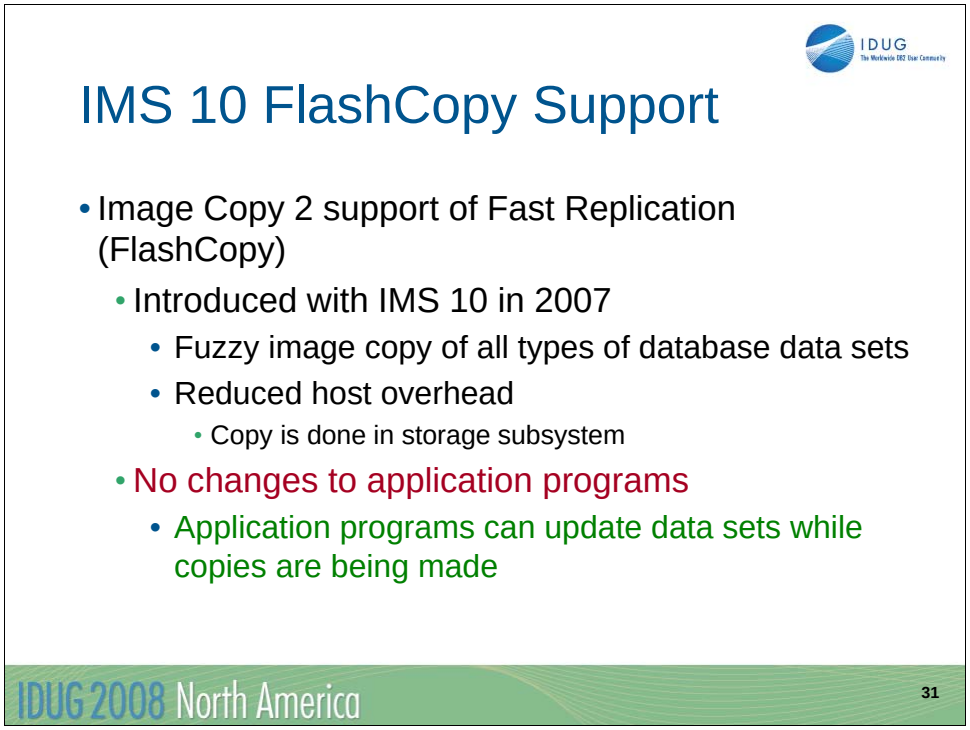
There is another significant advance that was made without application changes. MSC was introduced in IMS/VS 1.1.3. MSC allowed us to define transactions as remote. IMS would ship these transactions to another IMS for processing. It would return the reply to the originating IMS system which would deliver it to the user terminal. Simply stated, it added a capability to route transactions among IMS systems.

Once again no application changes were required. Those transactions written before the idea of MSC handled the messages arriving via MSC and created response messages that were delivered via MSC.



Shared queues is another illustration of how IMS has expanded the processing of messages beyond one IMS system. Shared queues was introduced in IMS/ESA 6.1. With shared queues multiple IMS systems share a set of message queues. Any IMS system may schedule a program to process a message, even though it was placed on the queue by another IMS system. This was another way to increase the capacity and availability of IMS systems. Shared queues has synergy with data sharing. An IMSplex using data sharing and shared queues allows multiple machines to handle large and variable work loads. IMS systems may be added or deleted from the IMSplex dynamically to fit the workload.

Shared queues moves the queues from IMS systems to Coupling Facility structures. It is a very different way of handling messages. Nevertheless, application programs are not changed when shared queues is implemented. Once again we benefited from the architecture of IMS.



This is a much more recent enhancement to IMS. IMS 10 introduced support for Fast Replication in Image Copy 2 and the Database Recovery utility. Fast replication invokes data set FlashCopy on the storage subsystem. This makes a copy of the database data set while it is open and being updated by application programs.

No application changes are required.

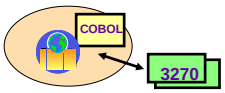
Similar changes in system and utility functions have been made throughout the life of IMS.

IMS Database and System Evolution

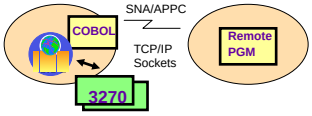
- Application compatibility
- Database compatibility
- System capacity growth
- High availability enhancements
- Leveraging of mainframe system capabilities

IMS Connectivity Evolution

- Remote access (1970s)
  Host focus

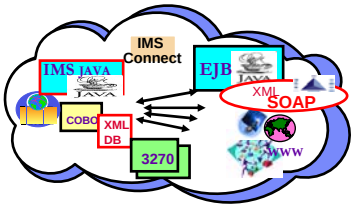


→ Client/Server (mid 1980s – mid 1990s)
Program-Program distributed access



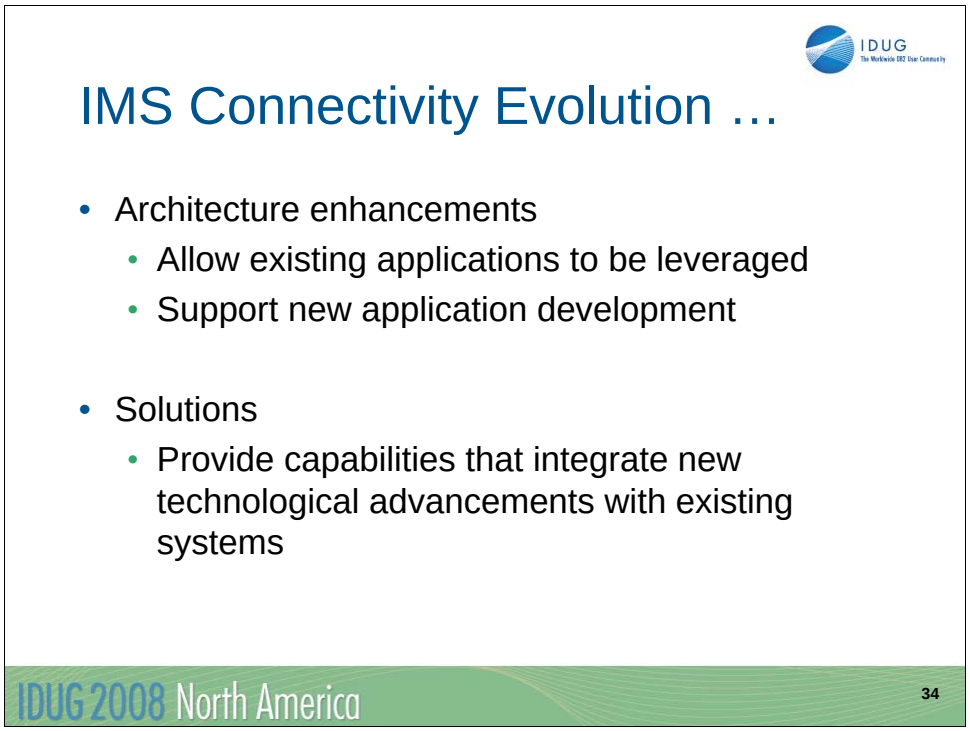
→ Web support (mid 1990s – 2000s)
The Internet

→ e-business..On Demand..
SOA (2000s)
Integration and Standardization

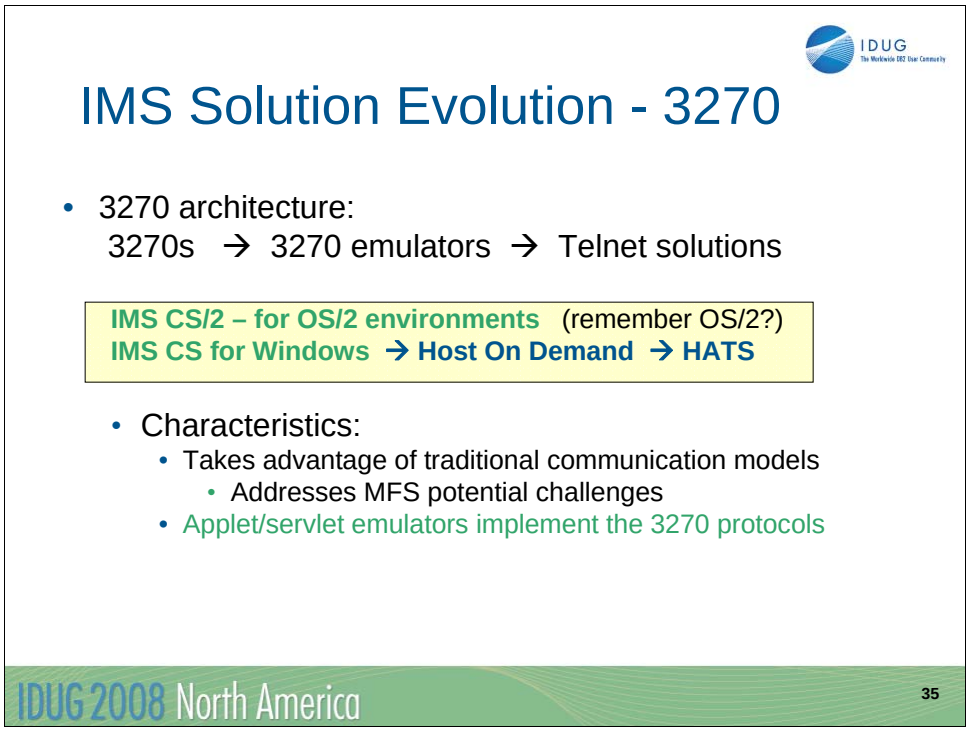


IDUG 2008 North America 33

Throughout the past several decades, IMS architected interfaces and solutions have evolved to keep pace with technology. The bullets on this visual describe the advancements that have characterized the IMS environment from simple host-based access to today's more complex world of the commercial internet. The underlying theme, however, is that even as IMS has opened up to each of these new interfaces, the IMS runtime environment continues to leverage existing applications with minimal or no change. In circumstances where modifications are needed or new application development is required, application programmers can continue to code to familiar interfaces.

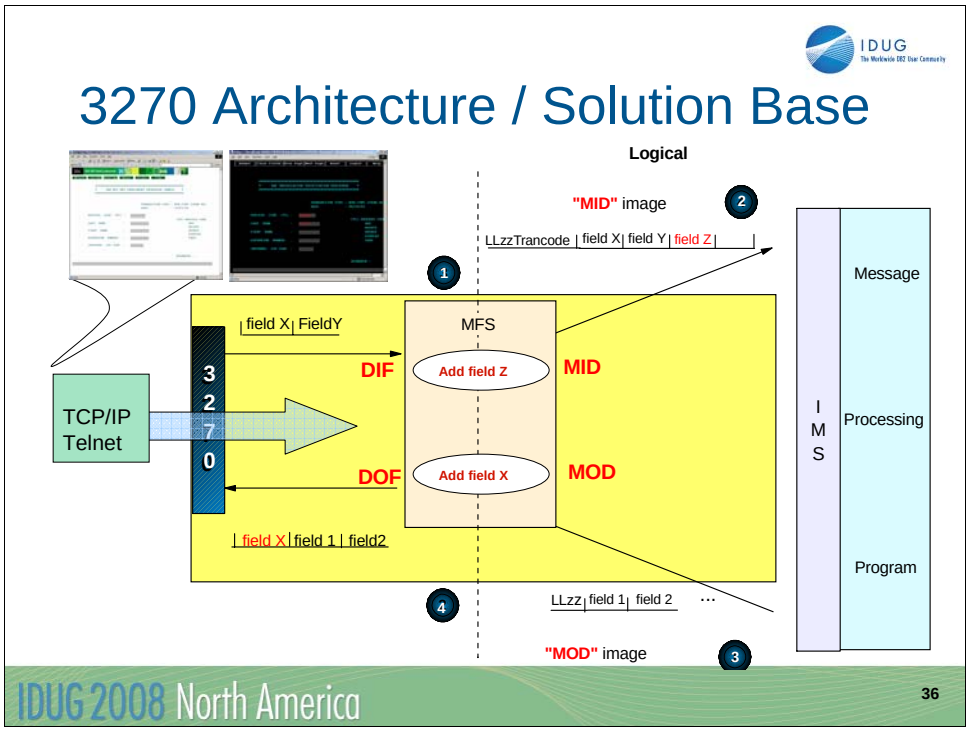


The evolution of the IMS environment is primarily characterized by enhancements in architected interfaces as well as by IMS-provided solutions that leverage the new architecture enhancements and facilitate migration to and exploitation of the new technologies.



3270 support was introduced into IMS in 1972. In the early 1980s, with the evolution of the PC, IBM introduced a computer program, the 3270 emulator, which duplicated the functionality of the 3270 device. As access through TCP/IP networks into a mainframe host became desirable, many emulators began to communicate using the TN3270 variant of the standard TCP/IP TELNET protocol. In 1993, IMS introduced the IMS Client Server family of products. The IMS Client Server/2 product was a way for OS/2 applications to invoke IMS transactions on an enterprise server. The complexity of 3270 EHLLAPI data streams were hidden from the workstation application developers through a simple-to-use RPC-like call application programming interface (API). IMS Client Server for Windows was introduced as another capability that extended the IMS CS/2 functionality so that application programs running in Windows operating systems could leverage the RPC call interface to access not only IMS but other mainframe systems such as CICS, TSO, etc. Additionally, applications that were developed in the workstation could also take advantage of the workstation capabilities such as GUI interfaces and drag-and-drop functionality.

As the client/server architectures progressed into the internet environment, the IMS CS for Windows functionality also evolved into another capability called Host On Demand which was a java-based internet application for host access. Host On Demand (HOD) initially provided a java applet with 3270 emulation interfaces that could be downloaded on demand, or as needed, into a browser. The applet communicated with a TN3270 server with the 3270 data stream encapsulated in TCP/IP protocol headers. Later enhancements to HOD provided more comprehensive capabilities. HOD also became a foundation for other products such as Host Integration Solution (HIS) and, more recently, Host Access Transformation Services (HATS). HATS addresses the workflow and navigation of host applications, without any access or modification to application source code, and also creates Service Oriented Architecture (SOA) assets from logic contained in the terminal applications. These products are of value to IMS environments because they take advantage of traditional LU2 communication models that address potential issues with MFS-based applications.



The illustration on this visual shows the role that MFS plays for an IMS application. MFS allows the application to be independent of both the communication protocol that is used to access IMS as well as any device-specific interaction.

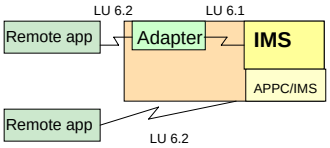
In this environment, IMS continues to communicate using LU2 protocols through VTAM to the Telnet server which is the partner endpoint of the LU2 session. Because Telnet uses encapsulation, the 3270 data stream is preserved until it can be interpreted by the TN3270 emulator on the workstation/browser. This technology supports a straightforward and easy to implement solution for IMS applications that take advantage of MFS capabilities such as adding literals or data into the data stream either inbound to IMS or outbound to the remote partner.

The example on this visual is intended to show how MFS is used to add field Z into the inbound data stream for the IMS application and field X on the outbound message from the IMS application to the 3270 device. When using 3270 emulation solutions, MFS continues to be invoked.

IMS Solution Evolution - APPC

- Client Server program-to-program architecture:

LU6.1/6.2 Adapter PO (IMS V2.2)
→ APPC/IMS - IMS V4



IMS WWW Templates
Component Broker (CORBA) – IMS connector

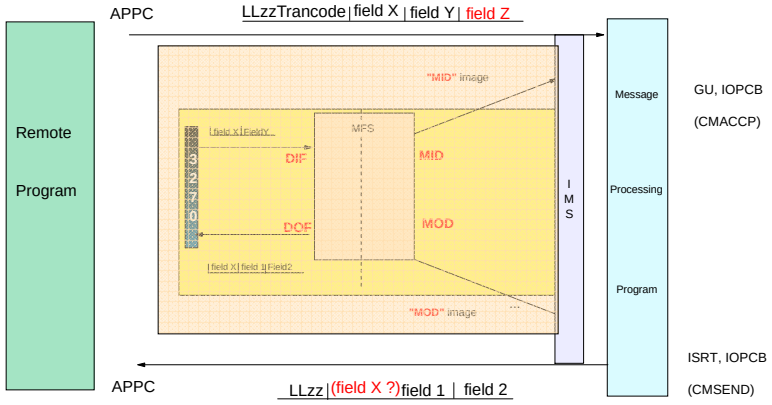
- Characteristics:
  - Takes advantage of SNA/VTAM standards
  - Implicit and Explicit support in IMS

Sometime in the decade of the 1980s, as programmable workstations and distributed servers began to gain popularity in commercial computing, the SNA specification for LU 6.2 was defined and accepted as a standard for program-to-program solutions. IMS introduced a program offering in the IMS V2.2 timeframe called the LU 6.1 Adapter for LU 6.2 Applications. This capability ran as an MVS address space that communicated using LU 6.2 protocols to a remote application and then prepared a message for IMS using an LU 6.2 link. In 1993, IMS V4 added native support for APPC into the product. With APPC/IMS, an IMS application could choose to use the implicit interface (DL/I calls) or it could take advantage of the explicit interface and control the communication sequence with CPIC (APPC's common programming interface for communication) verbs.

The APPC interface opened IMS up to a variety of new environments and solutions. The IMS WWW Templates was the name given to a packaging of samples and tools that were delivered through the OS/390 BonusPakII in the mid-1990s. They contained templates for providing Web access to IMS programs that accepted APPC connections. The templates were invoked from the Web server on any APPC platform (such as S/390, Windows, AIX, Sun Solaris, or OS/400) and provided the conversion between URL and HTTP information and APPC protocols. These gateway programs provided a combination of formatting routines and macros for customers to use as models in providing their own access.

Another evolving specification included the concept of distributed objects. IBM's Component Broker implemented the CORBA (common object request broker architecture) standard. The concepts of OO such as inheritance, encapsulation and polymorphism, provided a framework into which the developer could plug in his or her own application components. The Component Broker functioned as an object server and provided an application environment that let remote client applications work with back-end systems like IMS through object-oriented middleware. The interface to IMS, the IMS Connector, leveraged APPC.

APPC Architecture / Solution Base



Allows existing IMS programs to be invoked by the remote APPC program
- Possible additional logic on the remote program to deal with some applications

Supports new IMS programs that can be coded to existing DLI interface or explicit APPC API

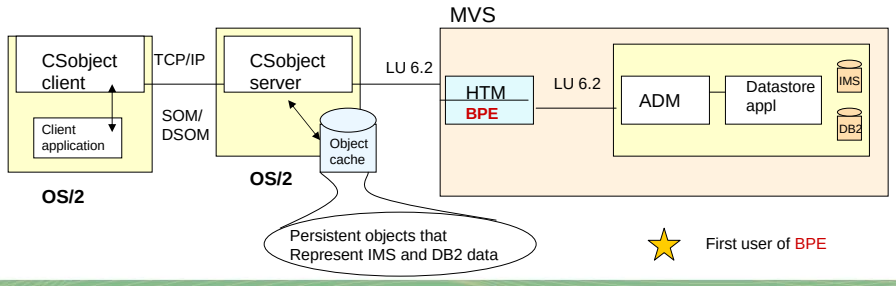
Since APPC bypasses MFS, remote programs must send the data stream in a form that the IMS application program expects to see when it performs the GU,IOPCB to retrieve the message. Likewise, the remote program will receive whatever the IMS application program places into the IOAREA with an ISRT, IOPCB. For many IMS applications, this bypassing of MFS is not an issue. On the other hand there might be some situations where the remote program would need extra coding to account for what MFS or the remote 3270 device would have supplied, e.g., literals or data.

The example in this visual shows the use of APPC to access the same transaction that was originally created for a 3270 interaction. Since MFS is not invoked, if the IMS application needs the field Z which would have been added by MFS, the remote program needs to account for this and create the message (at the top of the visual) to also include field Z. On the outbound side, note that if the IMS application program is not modified to add field X, then the remote program will need to deal with just the data in field 1 and field 2. Alternatively, the IMS DFSLUEE0 exit can be used to add the MOD name that would have been used (specified by the application program in the ISRT,IOPCB call) to the output message. The remote application could take the MOD name out of the message and use it to determine what the field X value should be. In essence, the remote program would take over the role of MFS for the message stream.

And…

- Early to mid 1990s: SOM/DSOM
  - Beginning of objects standards
    - Object-oriented interface for managing state of objects persistently

IMS CSobject – IMS Client Server Object Manager (IMS V5)



In the early to mid 1990s, the System Object Model and Distributed System Object Model specifications, based on CORBA standards, evolved as a means of defining client-based object-oriented access to data. As a reminder, CORBA was defined as an open standard for distributed object computing by the Object Management Group (OMG). The specifications described the concept of an *object bus* that could enable a client program to invoke methods on remote objects on a server, independent of the language the objects were written in or their location. Object request brokers (ORBs) mediated the interaction on both the client and server sides.

IMS CSobject was provided in the IMS V5 timeframe to allow distributed client object access to both IMS and DB2 data. The mainframe components included the HTM (Host Transport Manager) which was an MVS application that provided a mechanism to schedule application data managers (ADMs) and route to them the requests it received from the CSobject Servers. ADMs were IMS message processing programs that scheduled datastore application programs and managed IMS CSobject communications. Datastore applications programs were IMS application programs that invoked either IMS databases or DB2 tables.

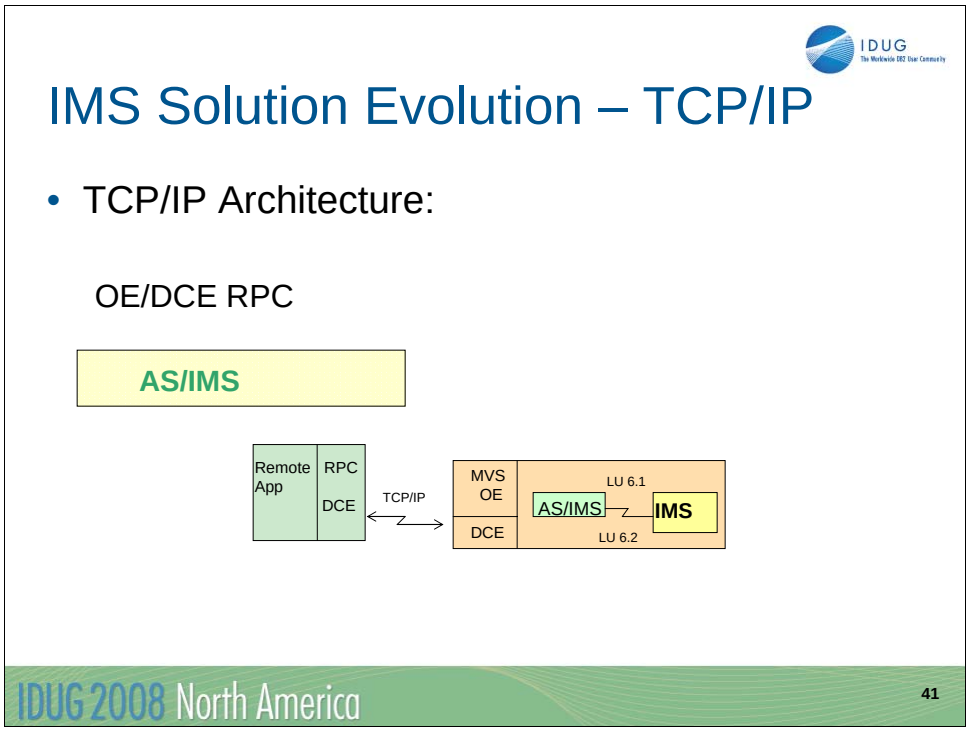
IMS CSobject was also the first user of the Base Primitive Environment (BPE)

And …

data objects - object strategy → ODBA (IMS V6)
IMS DB resource adapter

Meanwhile…

The concepts behind object-oriented access to data influenced IMS's object strategy towards access to IMS data. In the IMS V6 timeframe, the Open Data Base Access (ODBA) capability of IMS was introduced as an architecture layer to facilitate access to IMS data from environments outside IMS. The IMS DB resource adapter which leverages ODBA will be discussed in later visuals.



TCP/IP networks and architectures began to play a very important role in the enterprise world. Since IMS, as a VTAM application, did not have direct TCP/IP access capabilities, several products came about to provide a link from the TCP/IP environment to IMS.

Meanwhile, the Open Software Foundation (OSF) evolved as a nonprofit partnership of many companies with a unified purpose of making the computer marketplace an Open Systems environment. A result of this collaboration was the OSF Distributed Computing Environment (OSF DCE). To accomplish its task, OSF issued an invitation to the members of the computer industry worldwide to submit existing technology components for inclusion in OSF DCE. The answer to this call resulted in DCE capabilities that provide a high-level, coherent environment for developing and running applications on a distributed system along with tools for developing distributed applications and services for running applications.

More specifically, DCE defined the remote procedure call (RPC) interface to allow applications to access procedures on a network as if they were local subroutines. In this environment, the Application Server/IMS (AS/IMS) product was introduced in cooperation with MVS OpenEdition (OE) to support the specified OSF/DCE RPC model. AS/IMS ran as an MVS address space communicating using OE/DCE services to remote TCP/IP DCE environments. The AS/IMS interface to IMS was configured using either LU 6.1 (pre-IMS V4) or LU 6.2 protocols (IMS V4). This capability allowed remote applications to issue standard RPC calls (send/receive) to access IMS transactions.

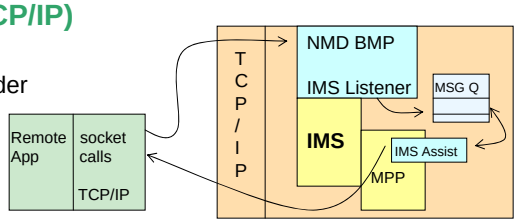
IMS Solution Evolution – TCP/IP …

- TCP/IP Architecture …

Sockets

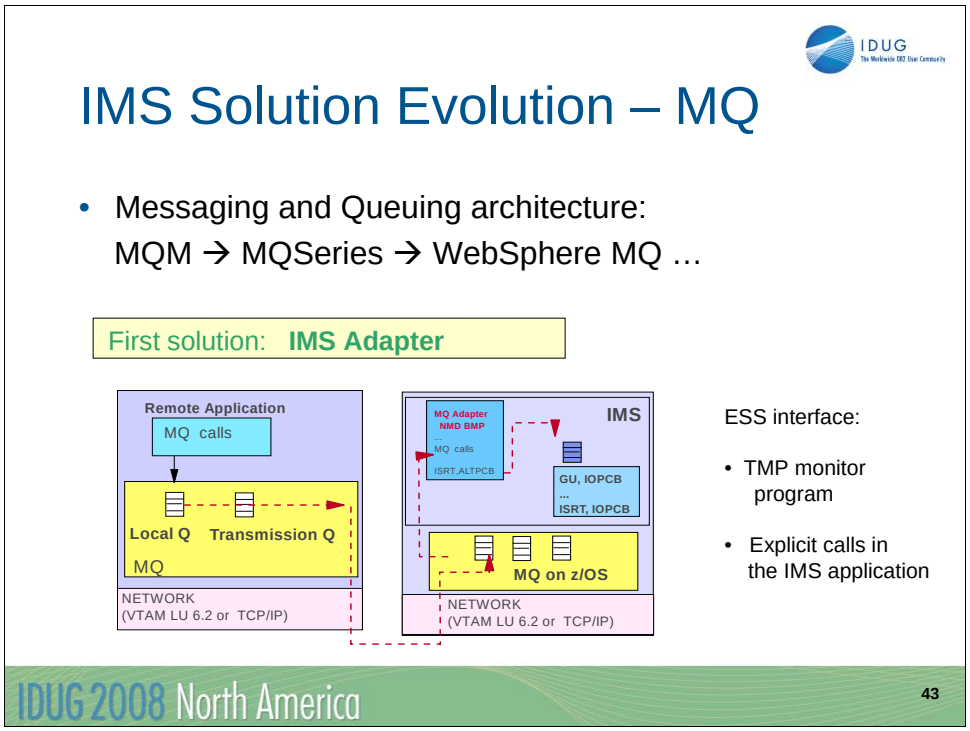
IMS Sockets (MVS TCP/IP)

- Implicit and Explicit
  - TIM (*LISTNR*) header
- C sockets
- Extended sockets
  - "Sokets" without "C"



At a simpler and lower architecture layer, MVS TCP/IP provided a function to provide a sockets interface to IMS. IMS Sockets was introduced in the mid-1990s as a two-way client/server communication capability between an IMS MPP and TCP/IP-connected clients using the sockets API. IMS Sockets is still available today and exists as two components. The first is the IMS Listener which is a non-message driven BMP program that waits for connection requests from TCP/IP-connected hosts. When a request arrives, the IMS Listener schedules the appropriate transaction and passes a TCP/IP socket, representing the connection, to that server. The message that is sent to the IMS application program is given a special segment containing the TIM (Transaction Initiation Message) with an identifier of *LISTNR* followed by the socket information. The IMS Listener holds the connection until the MPP starts and accepts the connection. Then the Listener is no longer involved. The second component is the IMS Assist module which is an optional subroutine included in the IMS application program and supports two APIs for the interaction. The implicit interface allows IMS application programs to continue to use DL/I calls for their message interactions by relying on the IMS Assist module to issue all socket calls on its behalf. The explicit interface opens up IMS application programs to actually issues TCP/IP socket calls.

In addition to IMS Sockets, MVS TCP/IP also introduced an extended socket interface. This capability was intended for application programs written in languages other than C. Socket calls are native to the C language but an interface needed to be provided for other languages. The extended sockets support provides a call interface for languages such as Cobol, Pl/I and Assembler.



IBM's messaging middleware solution which was originally known as MQSeries was launched in 1992. In its first interface to IMS, MQ provided a solution known as the IMS Adapter. This solution which is still applicable to today's environment uses the ESAF (external subsystem attach facility) and allows an IMS application to issue explicit MQI calls to MQ while participating in IMS syncpoint processing.

The IMS Adapter also provides a trigger monitor program which runs as a non-message driven BMP (CSQQTRMN) and is triggered by an MQ event, such as the receipt of an inbound message. Much like the solution described for IMS Sockets, the TMP places a special message on the IMS message queue for the target IMS transaction along with the information needed to access the message on the MQ queue.

OTMA

- Do you know what OTMA is?

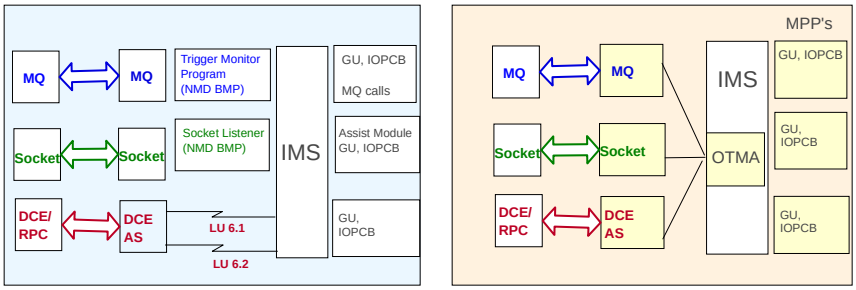
Wikipedia says that it was the affectionate group name used by the daughters of Tsar Nicholas II and Empress Alexandra:
Olga, **T**atiana, **M**aria and **A**nastasia

But … IMS modernized its meaning

Do you know the meaning of the acronym OTMA?

IMS Architecture Evolution– OTMA

- OTMA – Open Transaction Manager Access:
  - Standardization of access to IMS



- Optional use of an OTMA callable interface (OTMA CI)

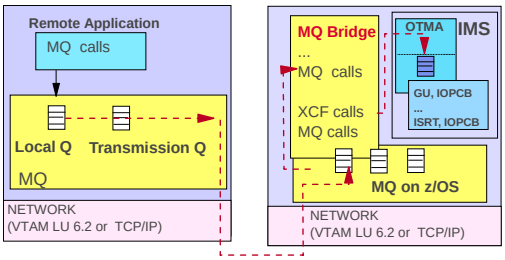
OTMA (Open Transaction Manager Access) was introduced in IMS V5 as an interface for host-based communications servers accessing IMS TM applications through the z/OS *Cross Systems Coupling Facility* (XCF). Through OTMA, IMS provided a standard interface for interaction with IMS that was geared towards scalability, performance and security. Since its introduction, many IBM and non-IBM solutions have taken advantage of this interface.

An OTMA client is defined as any MVS address space that uses the XCF interface to access IMS through OTMA. Oftentimes OTMA clients, e.g., the MQ Bridge, IMS Connect, etc., are themselves servers to remote client programs across a network boundary. These types of OTMA clients are assembler programs that are coded with an understanding of the technical protocols behind MVS XCF and OTMA. On the other hand, in the IMS V6 timeframe, IMS introduced the OTMA Callable Interface. OTMA CI is a simpler API interface for OTMA interactions which abstracts the implementation details of XCF and OTMA and is callable from MVS programs running in either authorized or unauthorized states.

OTMA Clients

- MQM → MQSeries → WebSphere MQ …

Second solution: **IMS Bridge**

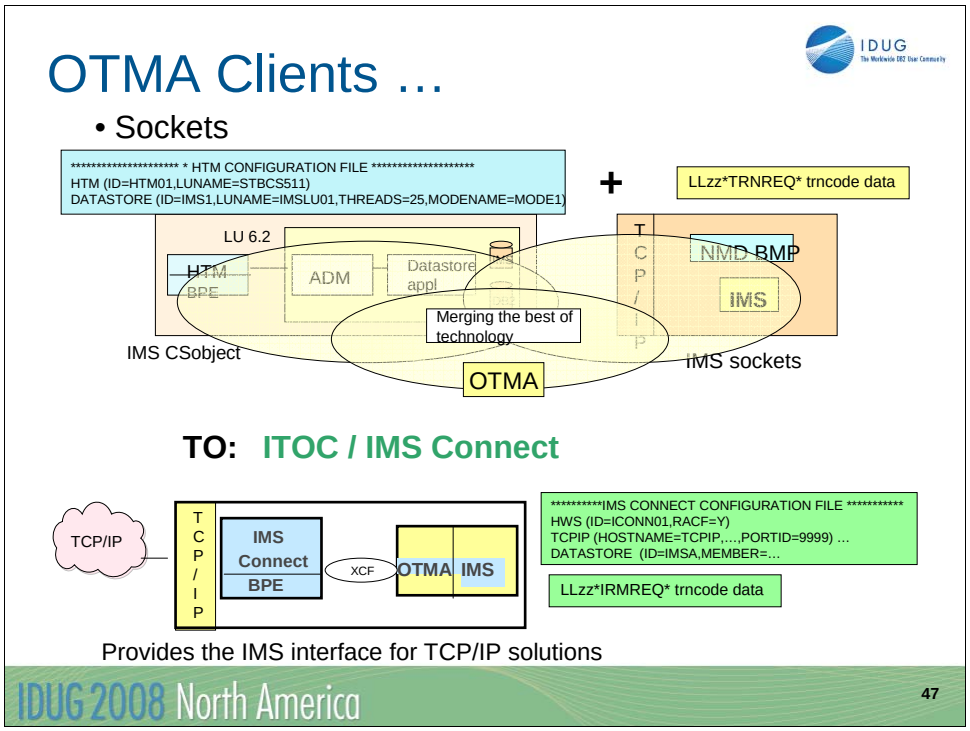


OTMA interface:

- Queue to queue capability
- Addresses issues such as security

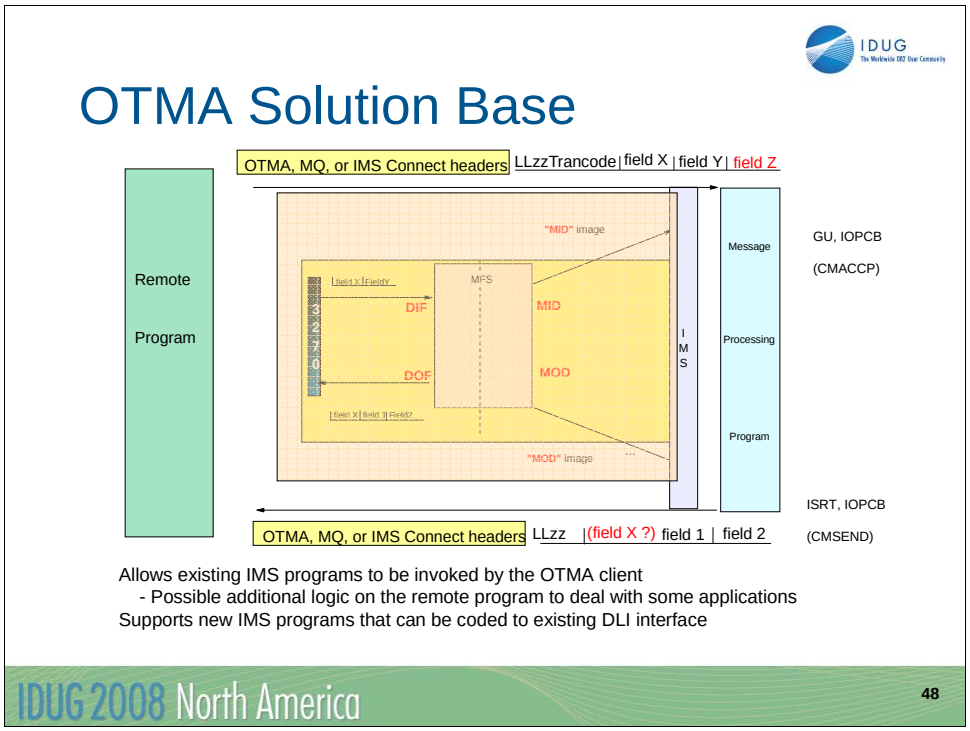
IDUG 2008 North America 46

One of the first products that took advantage of the OTMA capability was MQ. MQ's OTMA interface which is called the IMS Bridge supports a queue to queue interaction with IMS. MQ automatically sends messages to the IMS message queue and, likewise, provides a mechanism to receive messages from the IMS message queue. This interface, over and above MQ's original support with the trigger monitor program which ran as a BMP, provides the capability to invoke standard IMS functions such as security (userid access to a trancode) and allows the IMS application programs to continue to use DL/I calls to deal with the messages. With this scenarios, IMS programs do not need to issue MQ explicit calls.



The TCP/IP socket server support for IMS is another capability that leverages OTMA. As the visual shows, the structure of the MVS component of the IMS CSobject solution along with the TCP/IP IMS Sockets capability were merged on top of an OTMA interface to produce the ITOC (IMS TCP/IP OTMA Connection). ITOC was created as downloadable code that eventually evolved into IMS Connect. IMS Connect was initially provided as a fee-based tool until IMS V9 when it was integrated into IMS and shipped as part of the base product.

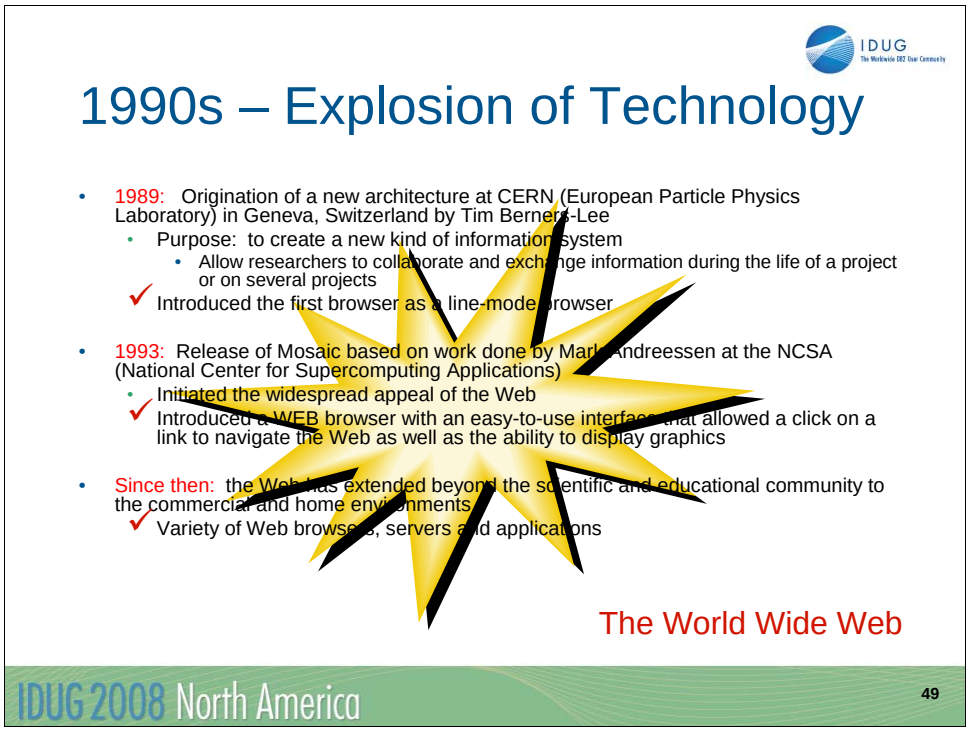
As IMS's TCP/IP socket server, IMS Connect interacts between the remote TCP/IP application and the IMS application. It shields the remote application from having to deal with IMS specific issues and acts as the interpreter to facilitate the communication on both ends. Through IMS Connect, remote programs can access existing IMS transactions with minimal to no change on the IMS side. Because IMS Connect defines and documents its application level protocol, remote applications coded to the sockets API can determine how to define messages to send to IMS and what to expect for output replies. On the IMS side, the IMS application programs continue to issue DLI GU and ISRT calls.



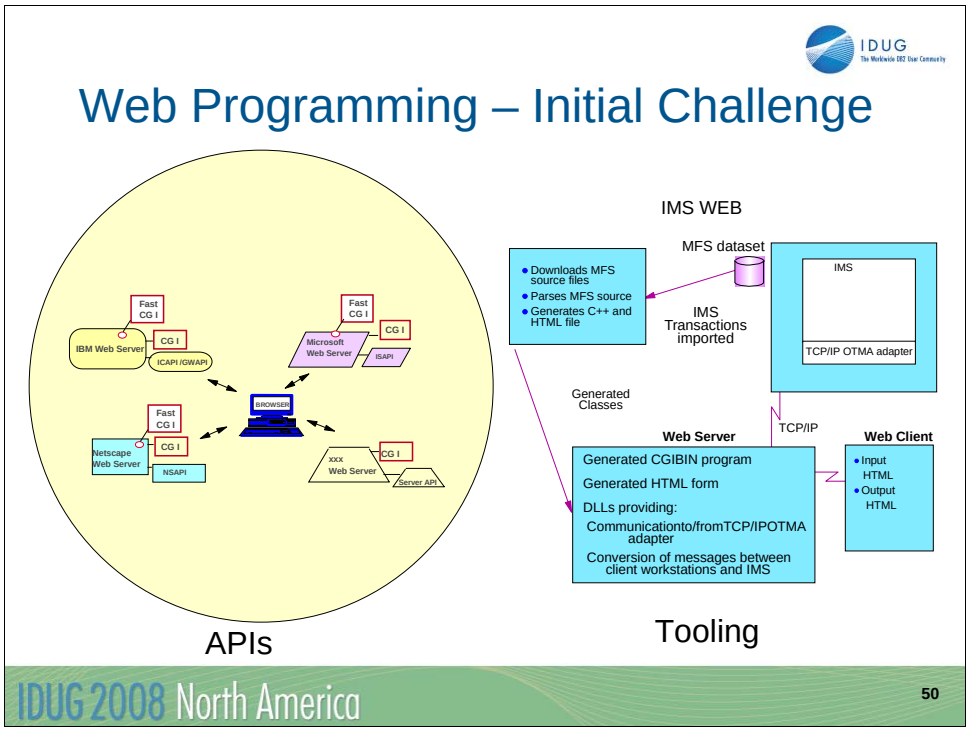
A primary consideration for all application interactions that are built on top of OTMA-based solutions is that MFS is not invoked.

If the IMS application expects data to be in the input message, data that would have been added by MFS in 3270 based scenarios, then the remote application is expected to include that data as part of the input message. Alternatively, an exit routine, e.g., the OTMA Input/Output Exit Routine (DFSYIOE0), can be created to provide this additional information.

Likewise, any data that would have been added by MFS on the outbound side will no longer be added to the message. The remote program has the option to request that the mod name that IMS would have used be passed back, in which case the remote program can determine what to do with the output message. As with the input side, exit routines such as DFSYIOE0 can also be used to add data to the message.

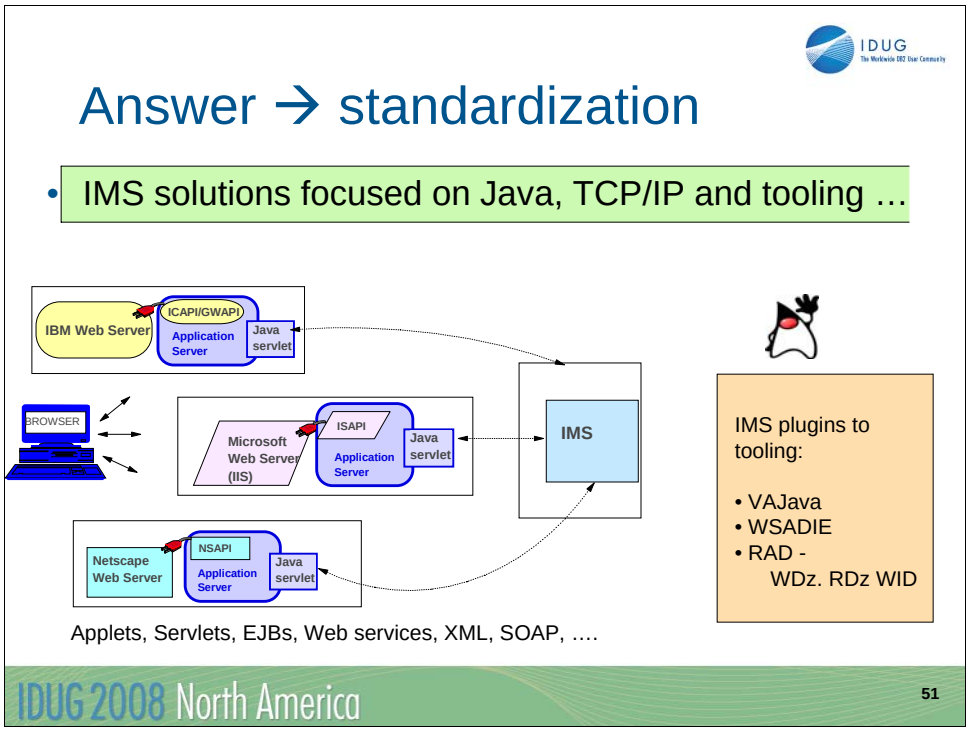


In the meantime, the World Wide Web exploded into the technology arena. From its inception in 1989 as a mechanism for research collaboration to today, the web has changed the way people communicate and market their businesses.



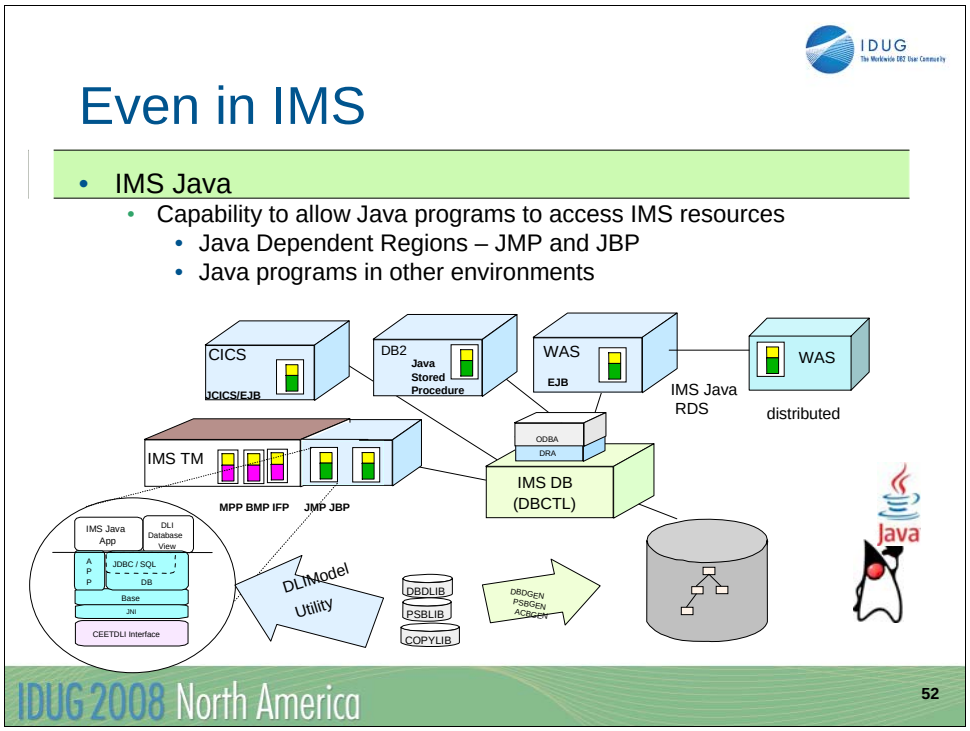
As the use of the Web exploded into the commercial domain, a variety of vendor solutions also emerged in the attempt to provide ways to integrate their environments and platforms into this new arena. The initial challenges evolved quickly especially in the areas of programming styles and application programming interfaces (APIs). For example, IBM's web servers provided the Internet Connection API (ICAPI) and the Go Web API (GWAPI) interfaces, Microsoft supported its Internet Server API (ISAPI), Netscape introduced the Netscape Server API (NSAPI), etc., Each server evolved with its own API making it confusing for application programmers who needed to program across multiple platforms. As a result the Common Gateway Interface (CGI) API quickly emerged as a standard for interfacing external applications with information servers. Additionally, Fast CGI also emerged as a version of CGI with lower overhead and additional mechanisms to address handling more web page requests at once. Many of the servers immediately took advantage of the standard CGI and Fast CGI specifications to address the growing problems of portability and interoperability.

Within the context of this fast-paced evolution, IMS also provided a set of solutions. IMS Web was introduced as a way to jumpstart IMS users into this new world of web-oriented access. IMS Web was PC-based tool that parsed MFS source and generated several components including a C++ CGI program for implementation in a web server, HTML files representing the IMS messages for browser access, and a set of Dynamic Link Library (DLL) files that supported the conversion of messages between the client workstation as well as the communication calls to access IMS using the ITOC. As a reminder, the ITOC was the predecessor to IMS Connect and functioned as a TCP/IP sockets connection adapter for IMS.



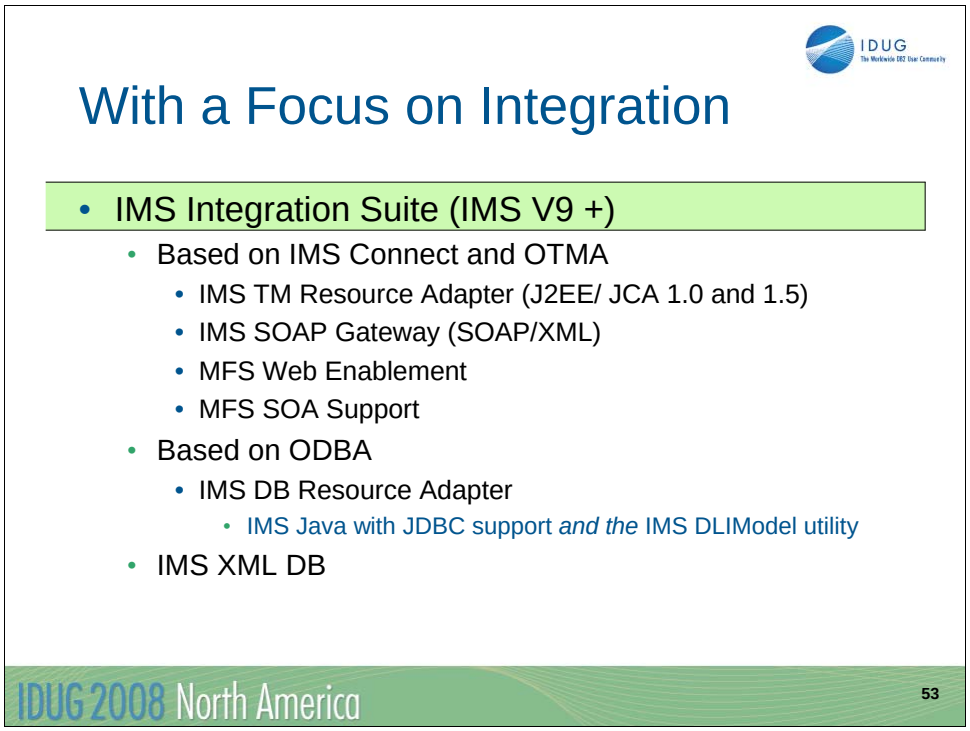
In 1995, Sun Microsystems released a new programming language called Java with the intent of supporting a "write once run anywhere" goal based on a virtual machine implementation and an object-oriented C, C++ style of notation. Following the success and acceptance of Java as a de facto programming standard, *application server* code was added to many web servers to support the java standard.

IMS provided support for this standardization in IBM tooling. The IMS code, originally called the IMS Connector for Java (IC4J), was provided as a plug-in to IBM's toolkit VisualAge for Java (VAJava). VAJava evolved to become the WebSphere Application Developer Integration Edition (WSAD-IE), which them merged with the Rational Application Developer (RAD). RAD can be implemented as a stand-alone tool or as an integrated part of other IBM tooling such as the WebSphere Developer for zSeries (WDz), Rational Developer for System z (RDz), and WebSphere Integration Developer (WID). The results of the development tool result in artifacts, e.g., an Enterprise Archive (EAR) file, that can be deployed to severs such as the WebSphere Application Server (WAS). Additionally, the IMS support includes runtime code, the IMS Resource Adapter (RAR) file, that is also deployed in the application server. IC4J is the basis for today's support which is called the IMS TM Resource Adapter.



Beginning with IMS V7, Java support also became an important function within IMS. To support this new programming language, IMS also introduced java virtual machines in two new dependent region types, the Java Message Processing (JMP) region and Java Batch Processing (JBP) region .

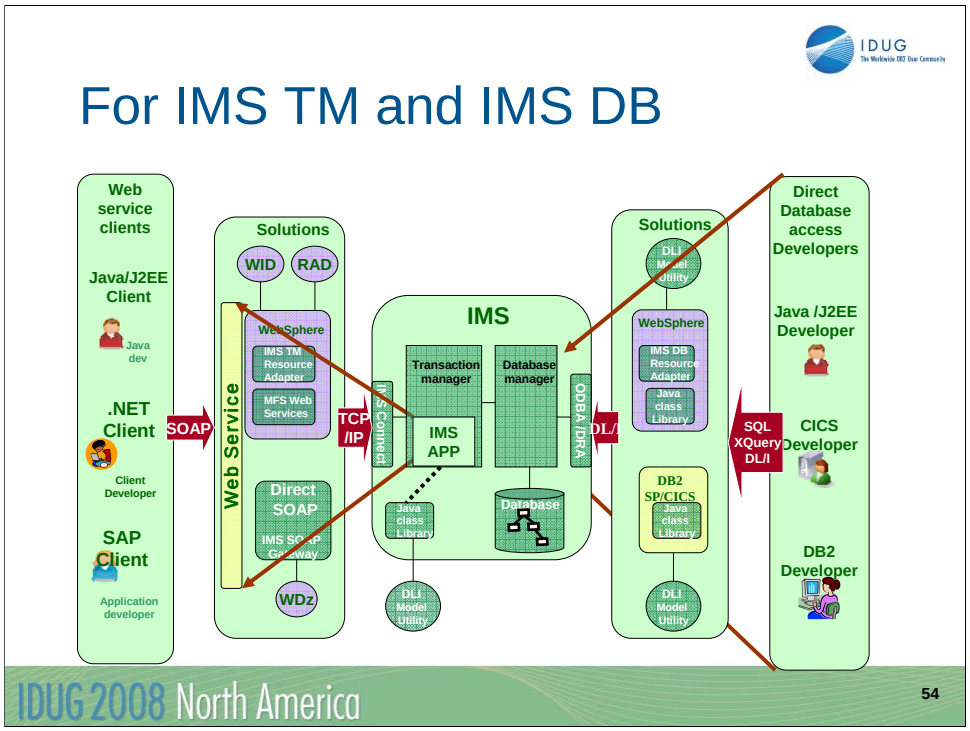
With this capability, Java programmers can easily access IMS resources through a set of IMS Java classes. Additionally, by running the IMS DLIModel utility to create an SQLview of the IMS hierarchical database, IMS databases can be accessed using JDBC calls. The IMS JDBC support has also been extended to other MVS java environments including JCICS, WAS on zSystems, and DB2 java stored procedures. In IMS V9, IMS Java Remote Data Services (RDS) enhanced the IMS environment to allow JDBC calls from distributed WAS environments to access IMS databases using either local or global J2EE transaction semantics.



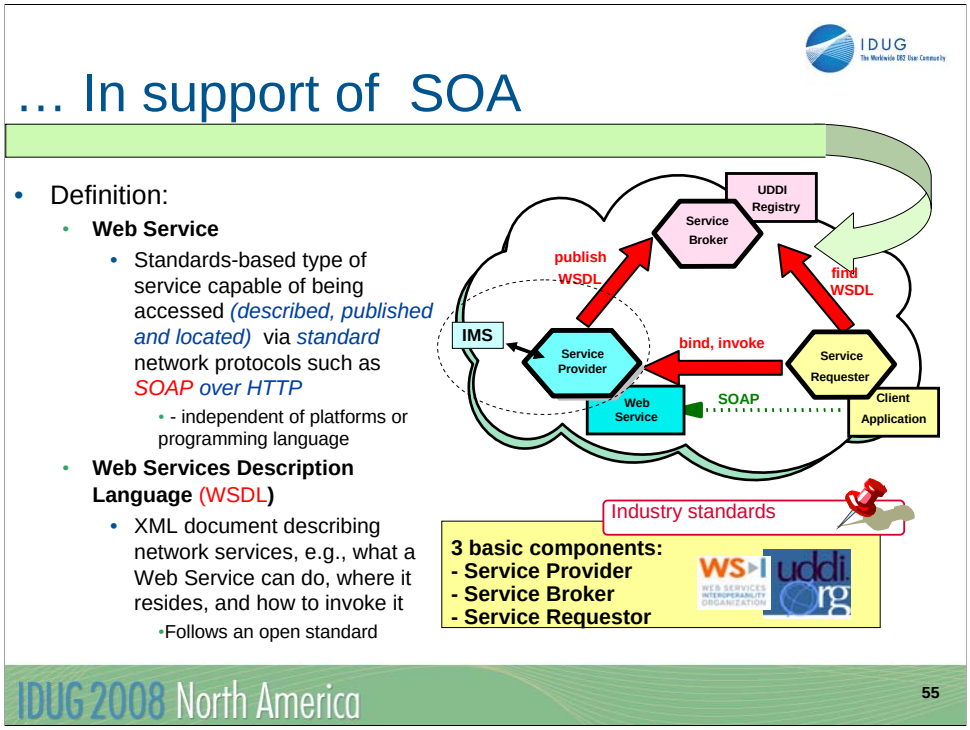
With the introduction of IMS V9, integration became the IMS focus for connectivity.

For transaction access, IMS enhanced the existing functionality of IMS Connect, OTMA and the IMS TM Resource Adapter as well as added new solutions with the IMS SOAP Gateway, MFS Web Enablement and MFS SOA Support. The IMS SOAP Gateway provides a stand-alone, lightweight (no requirement for a J2EE server) and robust solution for applications that need to access IMS transactions with the use of standard SOAP protocols and XML messaging. MFS Web Enablement allows the continued look and feel of an MFS-based 3270 web GUI for end-user access even as the underlying communication mechanism is converted to TCP/IP socket access, and MFS SOA Support takes advantage of MFS source to create web services.

The standardization for IMS database access is based on the IMS Open Database Access (ODBA) capability that was first introduced in IMS V6. The IMS DB Resource Adapter which is ultimately based on ODBA is a combination of the IMS Java JDBC support discussed on the previous visual along with the IMS DLIModel utility. The IMS DB Resource Adapter can be used to interface with the IMS XML DB capability that was discussed earlier in the presentation.



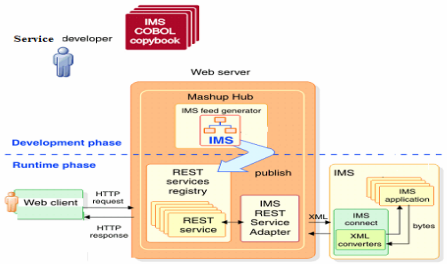
Putting it all together, this visual shows the broad range of solutions and access mechanisms that are provided by the base IMS products.



All of that brings us to today and SOA (service oriented architecture). A little bit of a background might be helpful in understanding this environment. As the web evolved, its dynamic nature along with continued technological advancements, resulted in increased complexities and challenges for true interoperability. Enterprises that were betting their businesses on these capabilities needed a guarantee of compatibility across platforms, applications and vendors. This set of challenges led to a collaboration in the technical and business community which ultimately defined an architecture specification geared towards standardization. SOA is all about standardization. The solutions that were shown in the previous visual allow IMS resources to be “exposed” as web services in support of this model.

And emerging technologies

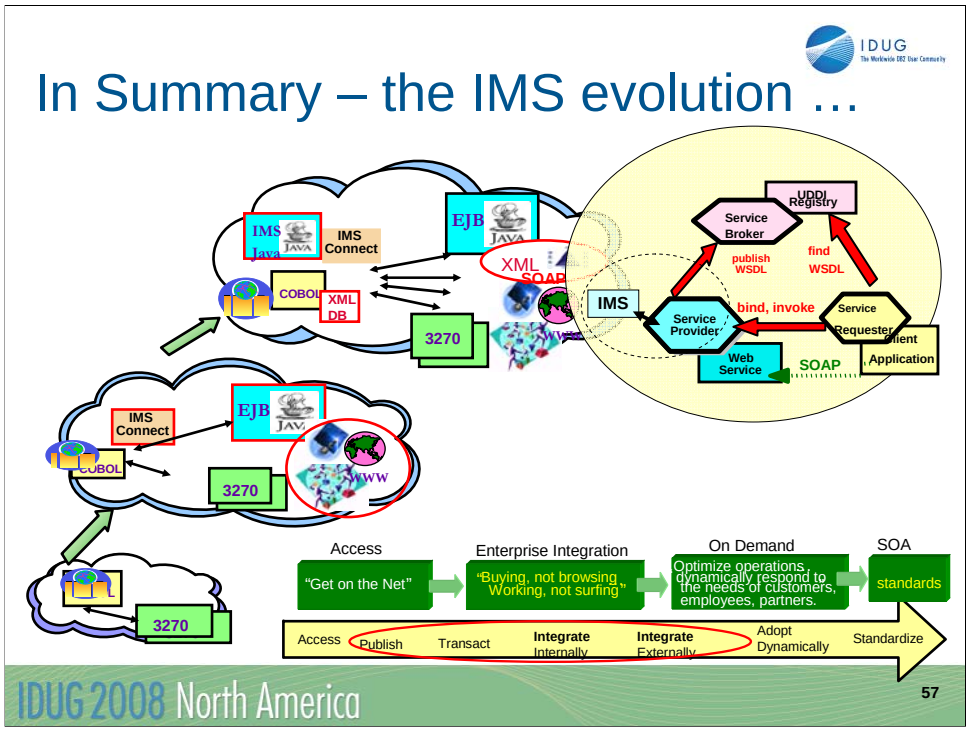
- Web 2.0
  - A second-generation of web communities and hosted services
    - Transitioning from websites containing isolated information to interlinked environments
    - The idea is that the web itself is a computing platform
- IMS Info 2.0
  - Tech preview on Alphaworks
    - Plugin to the IBM Mashup Starter Kit
  - Demo video on YouTube:
    - http://www.youtube.com/watch ?v=BWJGSC-RyXQ



… AND IMS continues to move forward with emerging technologies. Web 2.0 is a trend in web technology and design. The term is used to reference a "second generation of web-based communities and hosted services" as well as the changes in the ways programmers and end-users use webs rather than an update to any technical specifications. A key concept in Web 2.0 is that of a mashup which, in technology terms, is a web application that combines data from more than one source into a single integrated view. The primary idea behind Web 2.0 is ease of use and reuse.

IBM's Web 2.0 offering, Info 2.0, consists of a Mashup Hub as well as Lotus Mashups. The goals of a Mashup Hub include the transformation of (enterprise) data sources into Representation State Transfer (RESTful) services, i.e., simplified web services and feeds, along with the support of operations (aggregate, sort, filter) for creating composite services. The Lotus Mashups capability provides a visual editor for constructing Web mashups out of Web 2.0 enabled widgets, feeds, and services. Together, these tools assist in the transformation of enterprise assets into Web 2.0 services, and the customization of Web mashups from these services.

IMS Info 2.0 is a plugin to the IBM Mashup Hub Enterprise Edition that transforms IMS resources, e.g., transactions and data, into Web 2.0 RESTful services. IMS Info 2.0 does not require the service developer to have any knowledge on how to create or modify IMS source code, or how to send and process SOAP messages for invoking a traditional web service application. With IMS Info 2.0, IMS customers can remix and mashup their data rapidly to extend their business logic without the need to write a single line of code. Once created, an IMS RESTful service can easily be invoked through basic HTTP methods. For additional information, view the demo video on YouTube.



In summary, IMS continues to be a premier server in today's world. With new architected interfaces and solutions, the existing IMS environment answers the challenges that are presented by today's business needs. Existing IMS resources can continue to be accessed, as they have in the past, by simple straight-forward requests such as those that are requested using traditional 3270 interfaces and emulators, or they can participate in distributed requests from a variety of servers using TCP/IP and web-based solutions. As technology has advanced, the On Demand and SOA evolution with IMS continues to support the scalability, security, integrity and performance requirements that IMS has always provided.

The main point is that applications that were developed years ago with older communication architectures and technology can continue to be accessed in today's environment. IMS has always shielded its applications from having to understand changes in communication mechanisms. On the other hand, IMS also provides the ability to modernize its applications to directly participate in enhanced functions.

One more thing, there are many, many other solutions that are provided by vendors other than IBM. This visual is not intended to be an all inclusive, rather it is intended to give a feeling of how pervasive the value of IMS is felt in industry. As technology advances, IMS continue to emerge.

IMS Futures

- Leverage existing applications
  - Programs and databases
- Support new application development
- Integrate new technological advancements with existing systems
- Simplify operations and maintenance
- Increase high availability
- Expand capacity
- Leverage mainframe system capabilities

Now that we have seen the past and present of IMS, we might ask what's in the future. We think the answer is obvious. IMS will continue doing what it has done in the past to keep a vital part of our industry for almost 40 years.

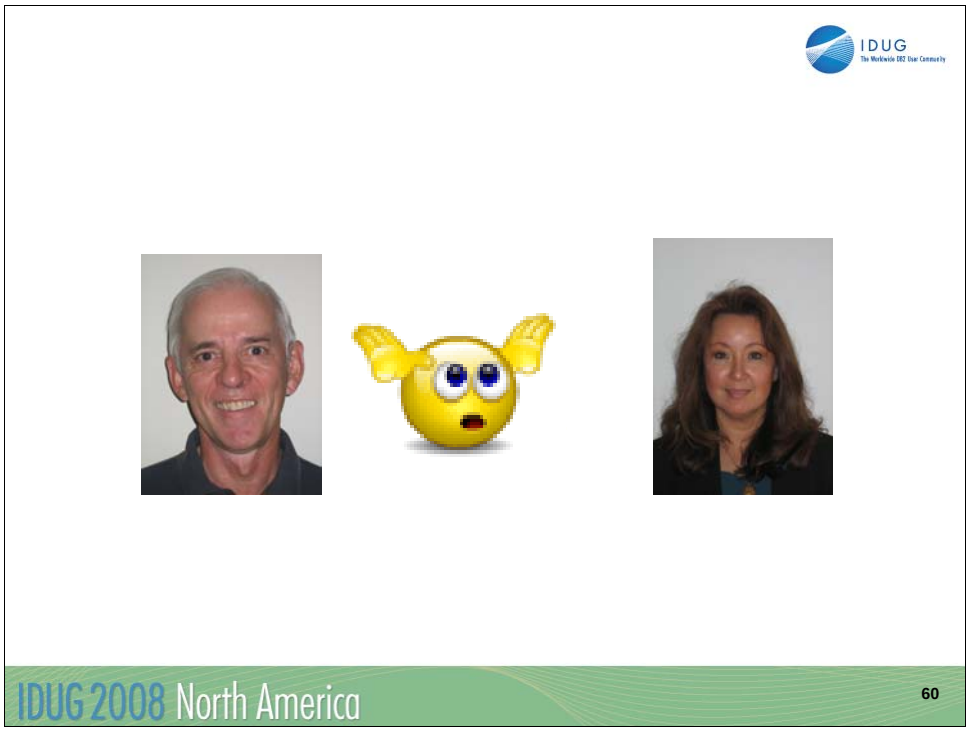
IMS will continue to leverage your investment in programs and databases. In will also support the development of new applications. An example from the present illustrates this. The introduction of XML database leverages existing IMS databases and programs which access and maintain them by adding the capability for Java programs to view that data as XML documents.

IMS will continue to integrate new technological advancements with existing systems. SOA and the support of Web 2.0 is one set of examples of this trend.

IMS will continue to simplify its operation and simplify its maintenance. The addition of Dynamic Resource Definition (DRD) and Abend Search and Notification in IMS 10 illustrate these trends. Obviously, more enhancements will be made in these areas.

Every release of IMS expands its capacity. Often restrictions to growth are relieved before users encounter them. For example, IMS 10 increases the number of Fast Path buffers and output threads that may be defined for a system.

IMS is solidly based on the IBM mainframe, the platform which delivers the highest qualities of service for transactions and data. As the platform raises the standards for capacity, availability, and integration IMS will leverage these capabilities.



Thank you for attending and participating in this session.

X04

IMS - Somewhere Beyond the Moon

Rich Lewis and Suzie Wendler

IBM

rzlewis@us.ibm.com

wendler@us.ibm.com