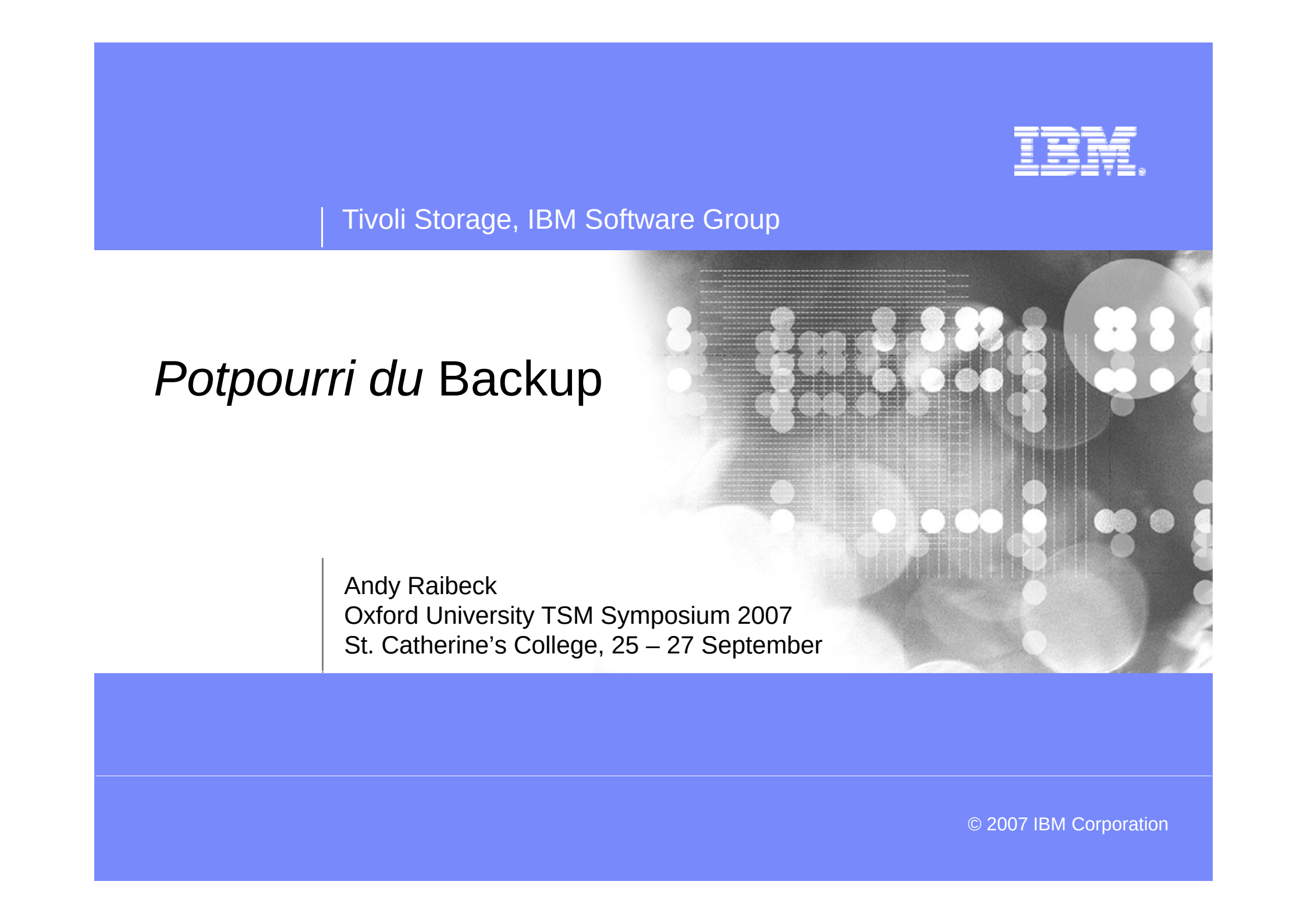


Tivoli Storage, IBM Software Group

Potpourri du Backup

Andy Raibeck
Oxford University TSM Symposium 2007
St. Catherine's College, 25 – 27 September

Trademarks

- The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
  - AIX
  - IBM
  - Tivoli
- Other company, product, and service names may be trademarks or service marks of others.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft 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.

Introduction

- There is a *potpourri* of backup techniques available in TSM
- Probably more to come in the future
- One size does not fit all
- Why can’t TSM decide for me?

What is covered in this presentation

- The various backup techniques offered by TSM Backup-Archive Client:
  - What you get vs. what you give up with different technologies and variations
  - Decision points
- Provide qualitative advice on evaluating different technologies and variations
  - What tools are available today to help with these decisions

What is NOT covered in this presentation

- How to implement:
  - Not a tutorial in option syntax
  - No dsm.opt or dsm.sys examples!
- Other backup or archive considerations
- Recovery or retrieve considerations

Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

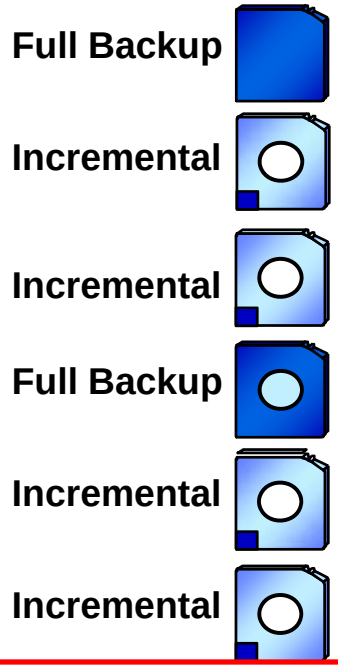
Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

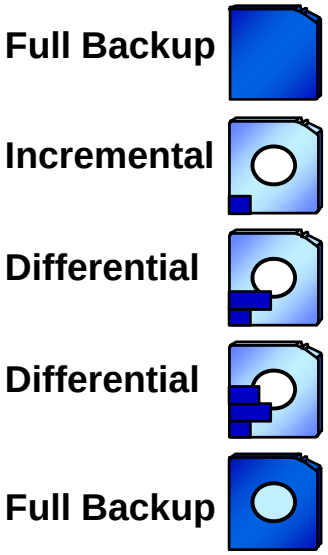
Other Backup Products

- Long restore time
- Poor media utilization
- Lots of copies of unchanged data
- High network utilization
- Higher impact to application server

Full + Incremental



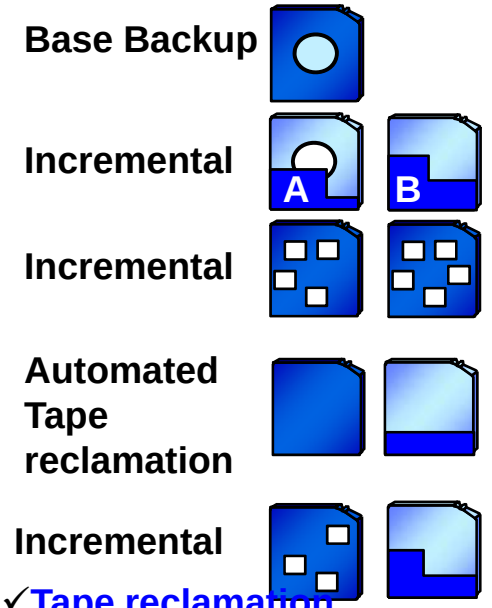
Full + Differential



TSM

- ✚ No unnecessary backups
- ✚ Client data consolidated on a few tapes
- ✚ Lower network utilization
- ✚ Low impact to application

Progressive incremental

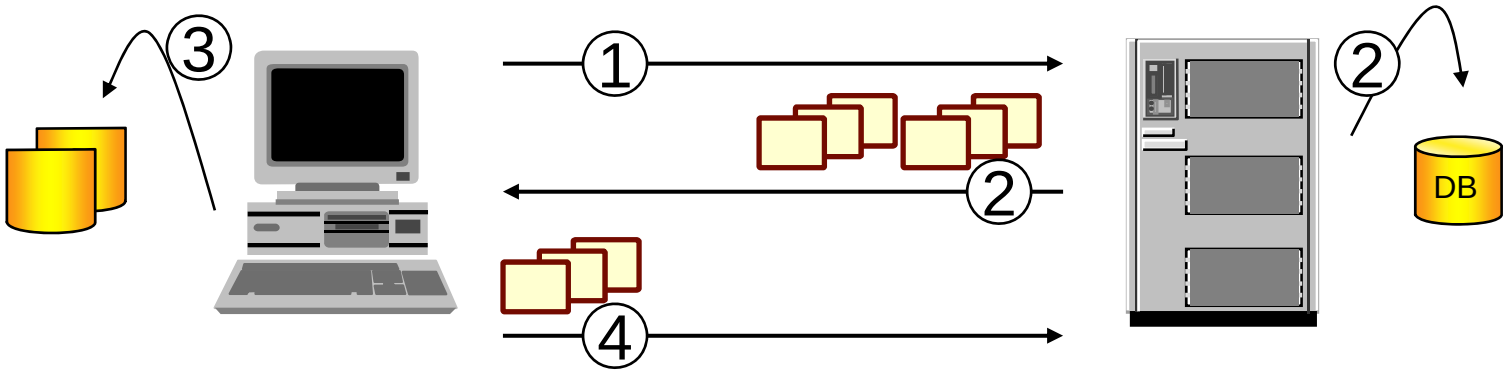


✓Tape reclamation
✓Collocation

Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

Progressive Incremental Backup



1. Client queries server for current view of file system
2. Server returns list of active versions for entire file system
3. Client scans and compares with local file system
4. Client backs up new and changed files

Progressive Incremental

- a.k.a Full Progressive Incremental or Full Incremental or “Incremental Forever”
- What you get
  - Single-instance store
    - Technically progressive incremental backup is a form of SIS (not a de-dup though)
  - Easier restore
    - No concept of applying incremental or differential changes to full

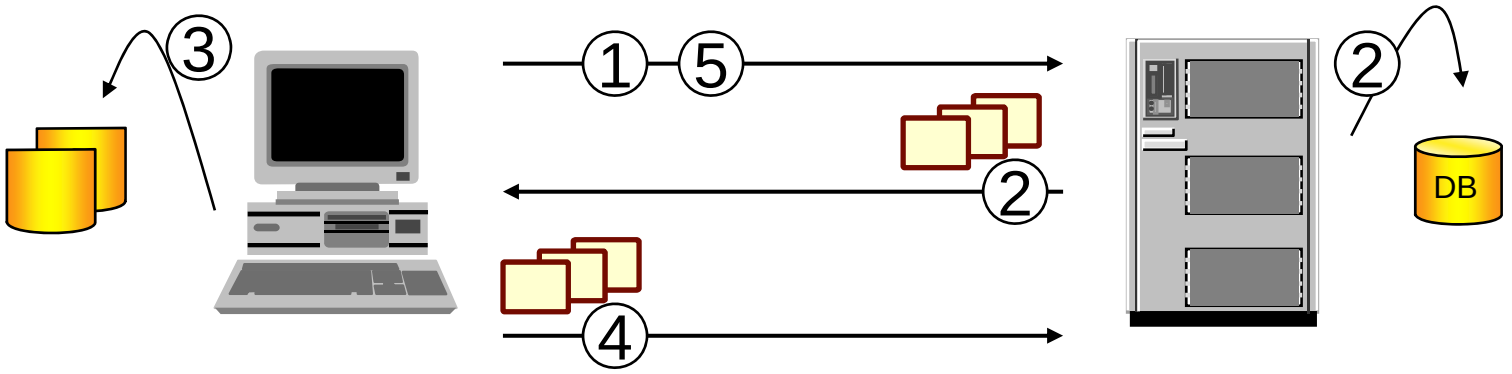
Progressive Incremental

- When to use this
  - Default technology in TSM
  - Hot spots:
    - Memory
      - Number of active backup versions
      - Length of file name
    - Run time
      - Biggest factor is file system scan time

Progressive Incremental Derivatives

- Memory efficient backup
- Memory efficient disk caching
- Virtual mount points
- Incremental by date
- File lists
- Journal-based backups

Memory Efficient

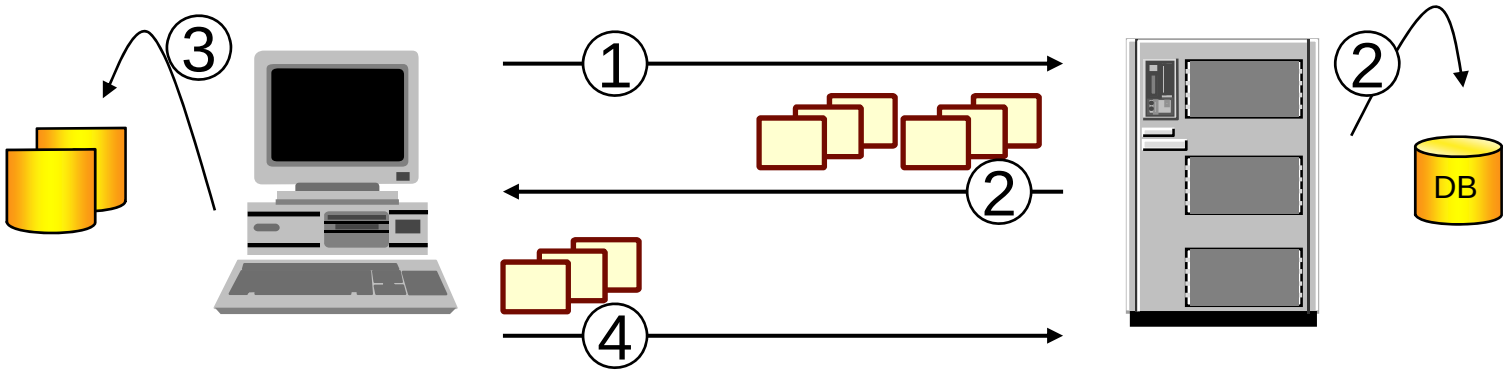


1. Client queries server for current view of first directory
2. Server returns list of files in directory (db query)
3. Client scans and compares with local file system
4. Client backs up new and changed files
5. Client queries server for current view of next directory, etc.

Memory Efficient Backup

- What you get
  - Smaller backup memory footprint
- What you don't get
  - Memory savings in directories with large number of objects
- What you give up
  - More network chit-chat
  - Increased backup run time
- When to use this
  - Incremental backups run out of memory
  - Can get rough estimate on memory usage

Memory Efficient Disk Caching



1. Client queries server for current view of file system
2. Server returns list of files for entire file system; client stores inventory on disk rather than in memory
3. Client scans and compares with local file system
4. Client backs up files

Memory Efficient Disk Caching

- How it works
  - Same as progressive incremental except …
  - B-A client can store TSM Server db queries on disk instead of in memory
- What you get
  - Even smaller memory footprint if regular memoryefficient backup doesn't reduce memory
- What you give up
  - Processing time (takes a bit longer to store and retrieve information from disk)
- When to use this
  - Incremental backup running out of memory and memoryefficient backup not sufficient Need individual file restore
  - Journal based backup technology is not available on your platform (note you might need this just to get the first backup done for journal based backups).

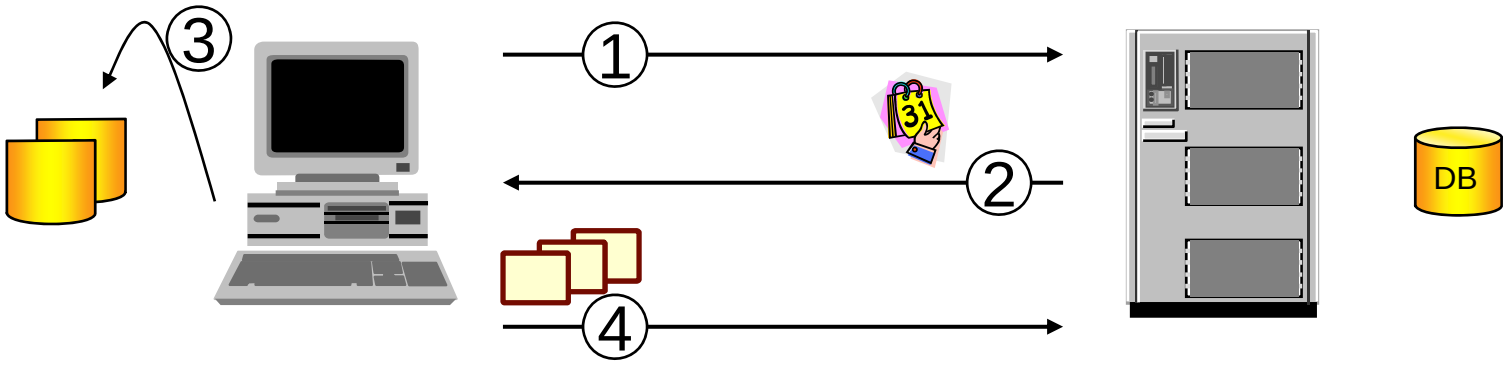
Virtual Mount Point

- Large file system logically partitioned
- Managed as separate file spaces on TSM Server

Virtual Mount Point

- How this works
  - Instead of mapping entire file system to single namespace on TSM Server (file space), user can define
  - What you get
  - Progressive incremental only has to work against subset of file system thus reducing memory requirements
- What you don't get
  - Relief from directories with large number of objects
- What you give up
  - Centralized management of file system on TSM Server
  - Virtual mount points are static and can't be migrated
- When to use this
  - Large, balanced (number of directories and objects) UNIX and Linux file systems can be efficiently divided

Incremental by Date

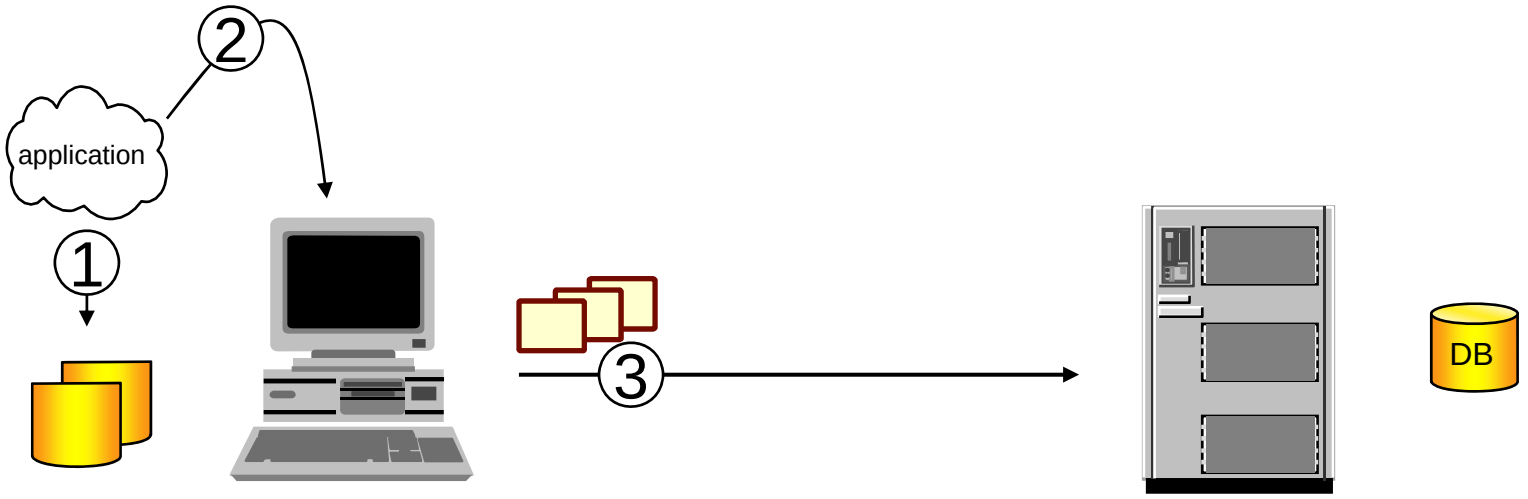


1. Client queries server last backup of entire file system
2. Server returns timestamp of last backup (end) of entire file system
3. Client scans and compares with local file system
4. Client backs up files

Incremental By Date

- What you get
  - Reduced time determining which files have changed
  - Removing processing time on server to query database
  - Removing network traffic to communicate results
- What you give up
  - Flexibility on scope of backup operation; you must be doing backups of the entire file system
  - File backups where changes do not affect date (attribute, mode, ACL)
    - Rename, copy, move, security change
  - Expiration of deleted files
  - Policy rebinding
  - Entire file system still needs to be scanned
  - No free pass for having client and server clocks out of synch
  - May not be able to use when client and server are in different time zones.
- When to use this
  - **Not meeting backup windows and …**
  - File system changes purely additive or changes, e.g., code repository or data that is basically "additive" (think of data like your family photos)
- Can be effective when doing weekly (or periodic) full incremental backups
- Read Chapter 9 in the User's Guide for more critical information on this option

File List Backup

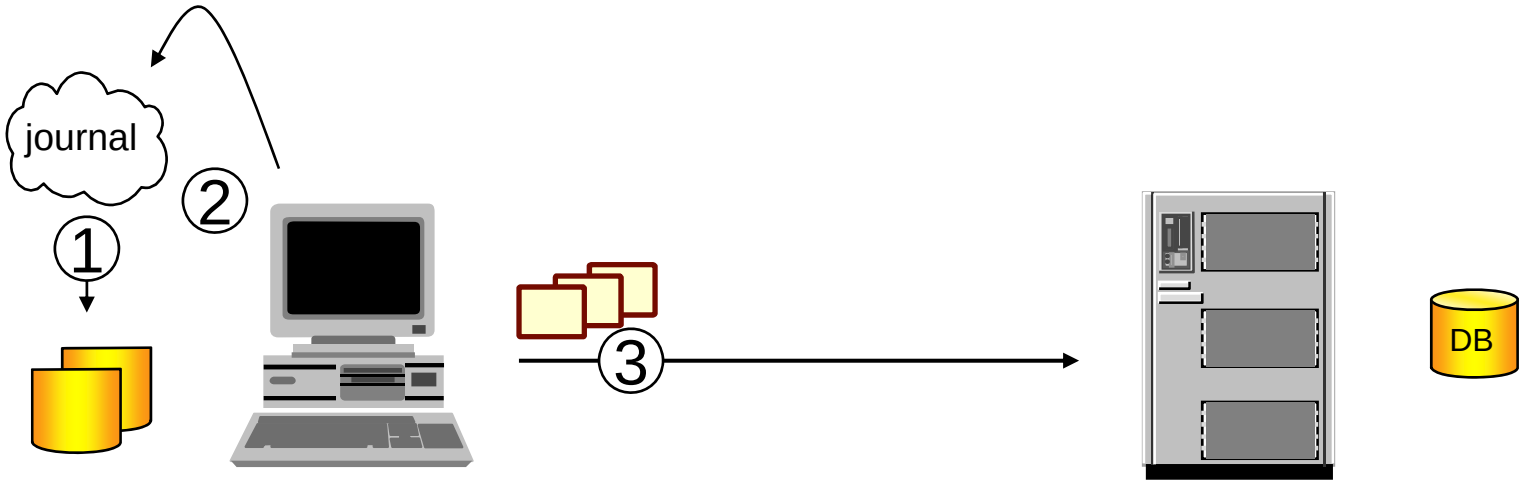


1. Application creates list of files for backup
2. Application passes list of files to client
3. Client backs up files (SELECTIVE backup)

File lists

- What you get
  - Selective backup eliminates the query of the TSM Server db and scan of local file system
- What you give up
  - Time – you have to find some mechanism to feed the file list
  - Implicit file specifications – file list will not accept wild cards or directory recursion
- When to use this
  - **Not meeting backup windows and …**
  - File system changes are predictable, e.g., an application is creating the changes and it would be easy to interface into the application to get that list of changes

Journal Based Backup

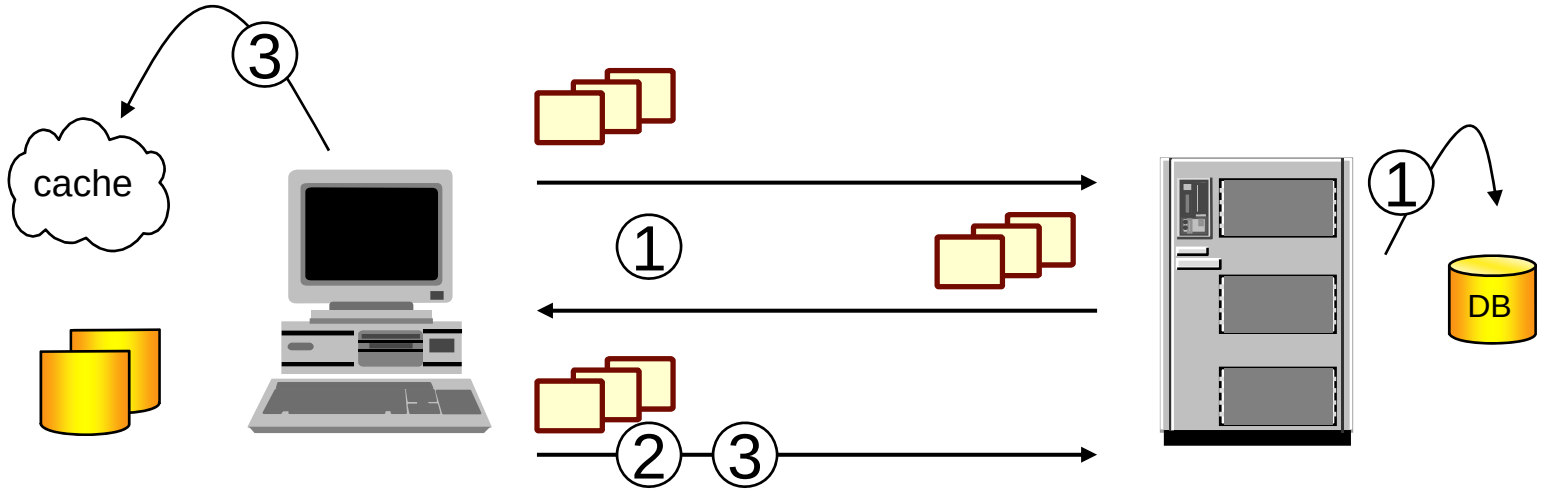


1. Journal monitors file system for changes
2. Client queries journal for file system changes
3. Client backs up files

Journal Based Backup

- What you get
  - Time to determine which files have changed is greatly reduced
- What you don't get
  - Relief from doing a full progressive (full) incremental backup against a file system on some periodic basis:
    - Initial backup to enable the journal
    - If a discrepancy is found between journal and TSM Server database
    - If the velocity of changing files is high enough to cause a notification buffer overrun
    - If the journal service is stopped and restarted
    - In general, it is a good idea to do periodic progressive incremental backups
- When to use this
  - **Not meeting backup window and …**
  - Small number of files (< 1,000,000) and small number of changes between backups (<1,000,000)
  - Large number of objects (<10,000,000) with 10-15% change rate try JBB (should emphasize a low change velocity, too… large numbers of changes over a short timeframe can cause notification buffer overflows)
  - Large number of objects + large number of changes will stress JBB … not a good fit

Adaptive Sub-file Backup



1. Client determines the file is a candidate for backup
2. Client backs up files
3. During data movement, client determines if there is enough information stored in the local cache to determine which sub-file parts have changed

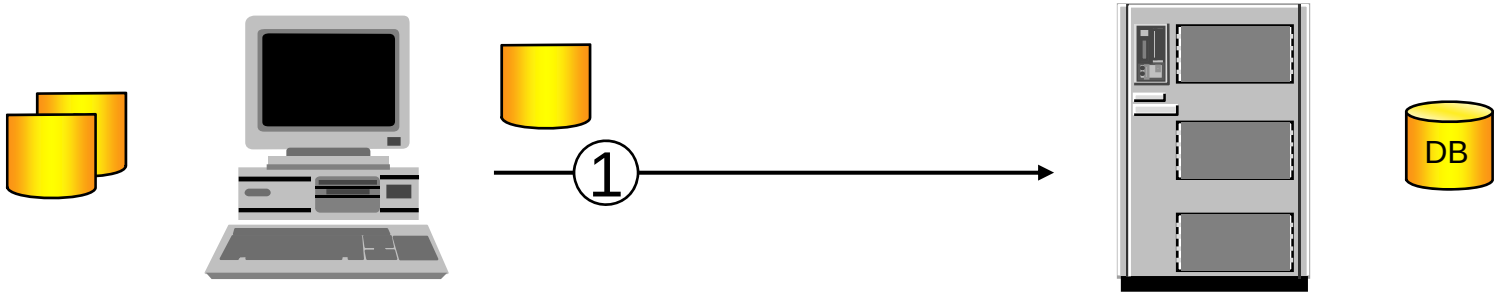
Adaptive Sub-file Backup

- What you get
  - Faster throughput
  - Reduced storage pool consumption
- What you give up
  - Local cache space (in the tens of megabytes)
  - A bit of CPU
  - Restore time (due to restore of base + delta)
  - Possible out-of-space conditions, if disk space is tight during restore, due to how file is reconstructed from base + delta
- When to use this
  - Network constrained
  - Small file profiles (limited to file size < 2 GB)

Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

Image Backup



1. Client sends logical block image of file system to server

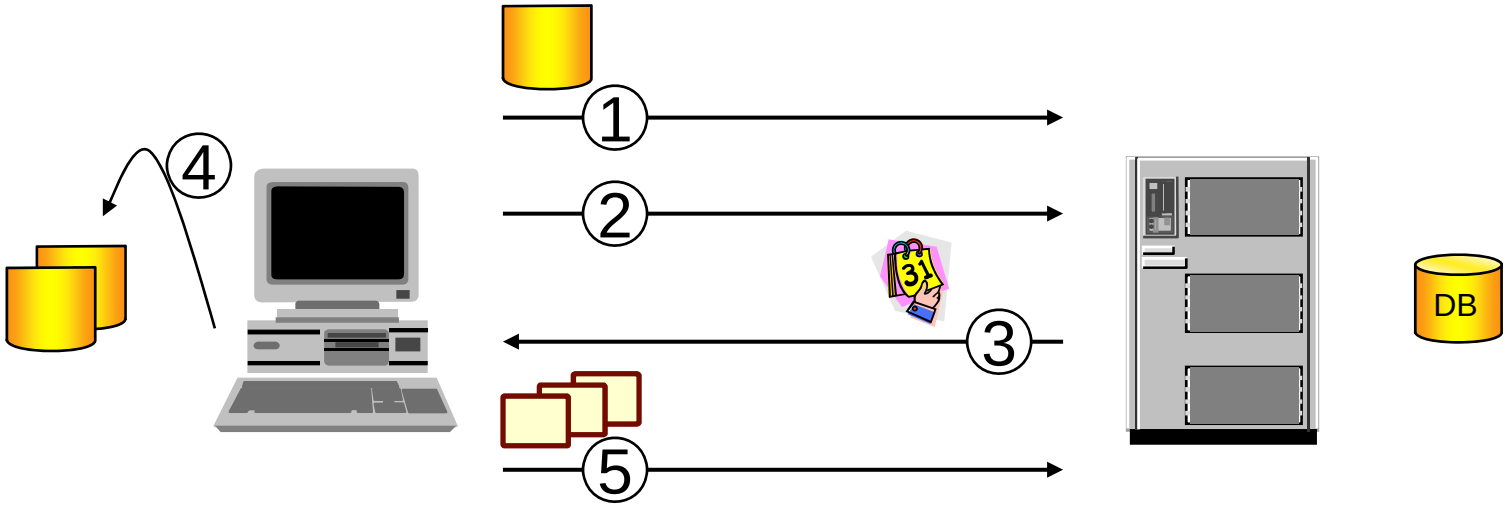
Image Backup

- No mapping between file and block
- Off-line and static backups vs. on-line backups
- What you get
  - Faster backups
  - No scan time to determine what has changed
  - Faster overall data movement
- What you give up
  - Ability to restore individual files directly from TSM Server (we know, we know…)
    - You would need to restore image to alternate location and pull data directly using Explorer or equivalent.

Image Backup

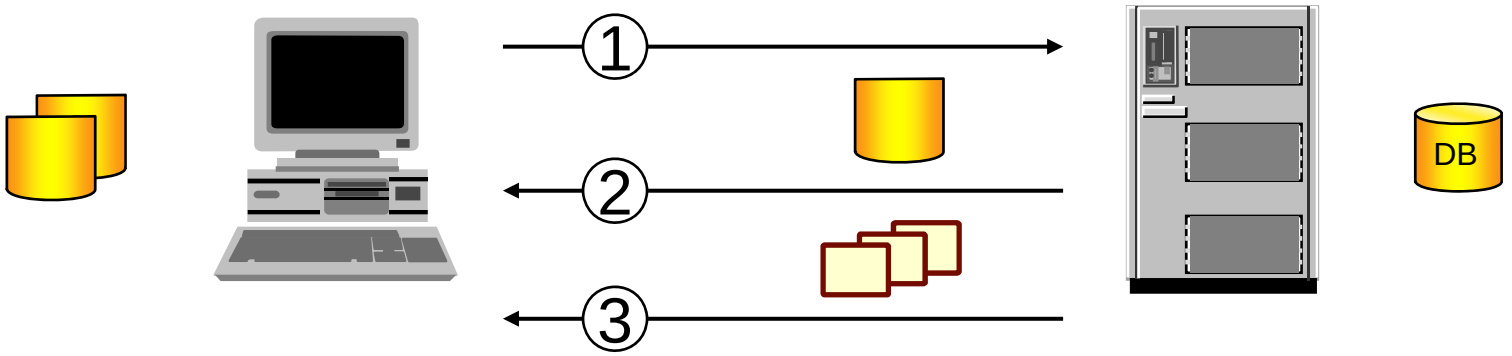
- Image backup
  - Backup at logical volume level
  - Ability to take image + incremental backups to provide single file restore
- Off-line image backup
  - Volume mounted read-only
  - Available for AIX, Windows, Linux x86, Solaris and HP
  - Exploited best by flashcopy operations
- On-line image (dynamic) backup
  - Volume left on-line
  - Fuzzy backup of changing data
- On-line image backup using snapshot
  - Volume left on-line
  - Image backup taken at single point-in-time (“crash consistent”)
  - Available for Windows and Linux x86

Image + Incremental by Date



1. *dsmc backup image* sends logical block image of file system to server
2. Upon subsequent *dsmc backup image –mode=incremental,* client queries server last backup of entire file system
3. Server returns timestamp of last backup of entire file system
4. Client scans and compares with local file system
5. Client creates transactions of files for backup

Image + Incremental by Date Restore Scenario

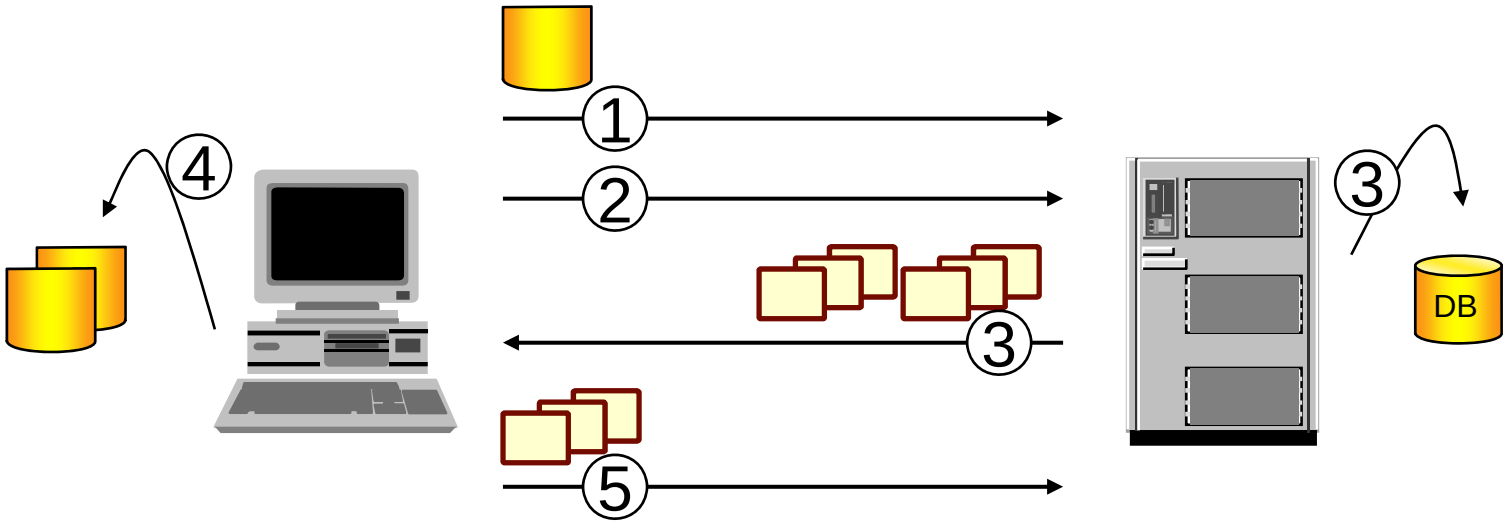


1. Client request incremental image restore
2. Server returns the base image
3. Server returns additional files which need to be applied to the base image to satisfy the recovery point

Image + Incremental By Date

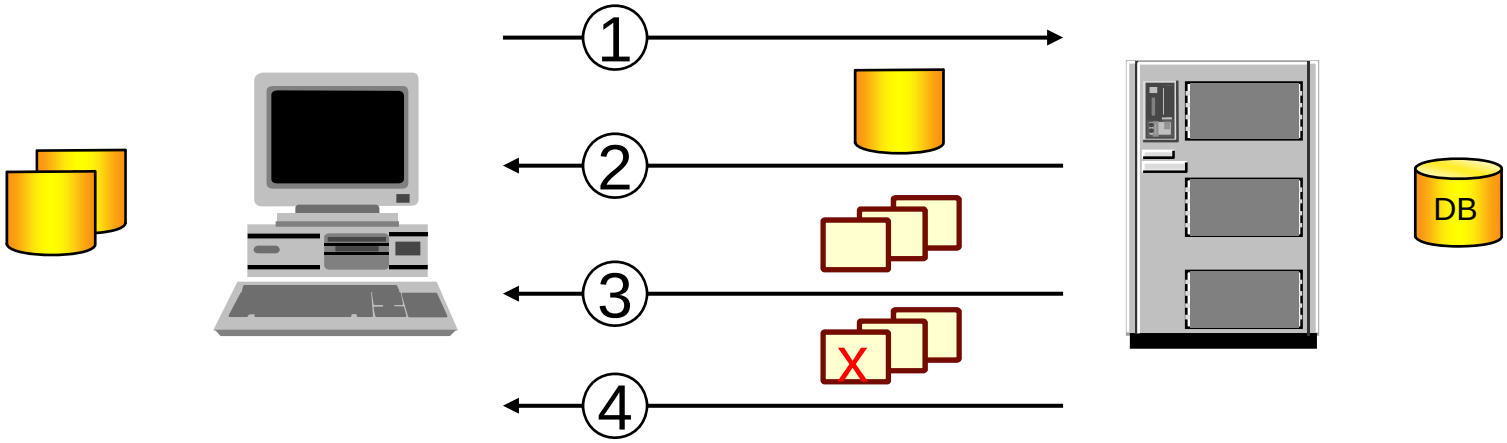
- What you get
  - Benefit of image backup plus some individual file protection of files that are changing
- What you give up
  - See Incremental by Date
  - No reconciliation of deleted files

Image + Progressive Incremental Backup



1. *dsmc backup image* sends logical block image of file system to server
2. Upon subsequent *dsmc incremental,* client queries server for current view of file system
3. Server returns list of files for entire file system
4. Client scans and compares with local file system
5. Client backs up changed files

Image + Progressive Incremental Restore Scenario



1. Client requests incremental image restore
2. Server returns the base image
3. Server returns additional files which need to be applied to the base image to satisfy the recovery point
4. Server optionally returns the list of files which need to be deleted from the base image to satisfy the recovery point

Image + Progressive Incremental

- **What you get**
  - Benefit of image backup restore for improving RTO
  - Benefit of progressive incremental backups for better RPO
- **What you give up**
  - Additional time to take periodic image backups
  - Storage space at TSM Server

Disclaimer

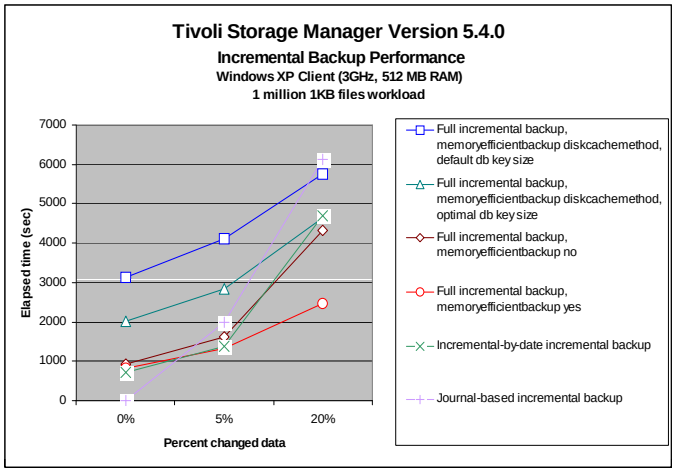
- The performance data contained in the following slides was measured in a controlled environment. Results obtained in other operating environments can vary significantly, depending on factors such as system workload and configuration. Accordingly, this data does not constitute a performance guarantee or warranty.

Improved incremental backup memory usage

Incremental backup, 1 million files workload, Win2003 server, 1 WinXP client, LAN, TCP/IP, Disk storage pool
IBM x360, 4-way 1.6GHz Xeon MP server, 6 GB, IBM PC 1-way 3GHz Pentium 4 client, 512 MB, Compression No, Encryption No, Storage pool CRC No, Bufpoolsize 131072, Txngroupmax 256, TXNBytelimit 2097152, NTFS filespace

Description	Elapsed time (seconds)		
	0% changed data	5% changed data	20% changed data
Full incremental backup, memoryeff diskcachem, default db key size	3138	4102	5742
Full incremental backup, memoryeff diskcachem, optimal db key size	2002	2850	4631
Full incremental backup, memoryefficient no	939	1620	4320
Full incremental backup, memoryefficient yes	813	1329	2479
Incremental-by-date incremental backup	707	1384	4699
Journal-based incremental backup	2	1995	6133

- ✓ The disk cache method writes server inventory data to a client disk file
- ✓ The disk cache method uses less than 20 MB of memory
- The disk cache method is generally slower than other methods

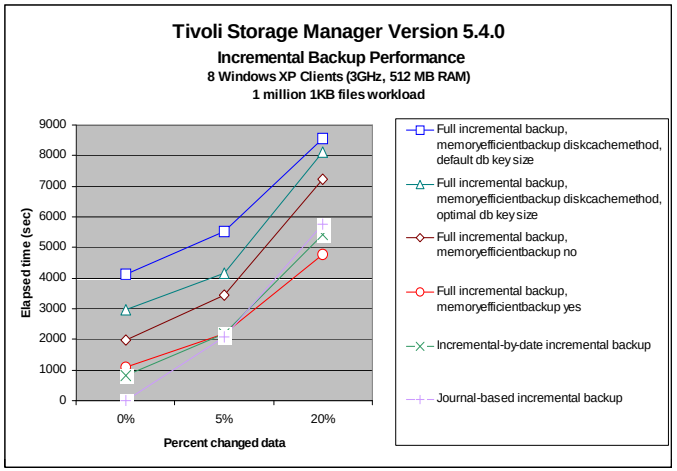


Improved incremental backup memory usage

Incremental backup, 1 million files workload, Win2003 server, 8 WinXP clients, LAN, TCP/IP, Disk storage pool
IBM x360, 4-way 1.6GHz Xeon MP server, 6 GB, IBM PC 1-way 3GHz Pentium 4 clients, 512 MB, Compression No, Encryption No, Storage pool CRC No, Bufpoolsize 131072, Txngroupmax 256, TXNBytelimit 2097152, NTFS filespace

Description	Elapsed time (seconds)		
	0% changed data	5% changed data	20% changed data
Full incremental backup, memoryeff diskcachem, default db key size	4142	5520	8551
Full incremental backup, memoryeff diskcachem, optimal db key size	2969	4168	8120
Full incremental backup, memoryefficient no	1975	3436	7241
Full incremental backup, memoryefficient yes	1086	2177	4763
Incremental-by-date incremental backup	812	2189	5420
Journal-based incremental backup	4	2074	5746

- For all tests measured with 8 WinXP clients, the disk cache method was slower than the 'memoryefficientbackup yes' method
- For 5% changed data, this elapsed time difference was measured at +1991 seconds, or 91% longer

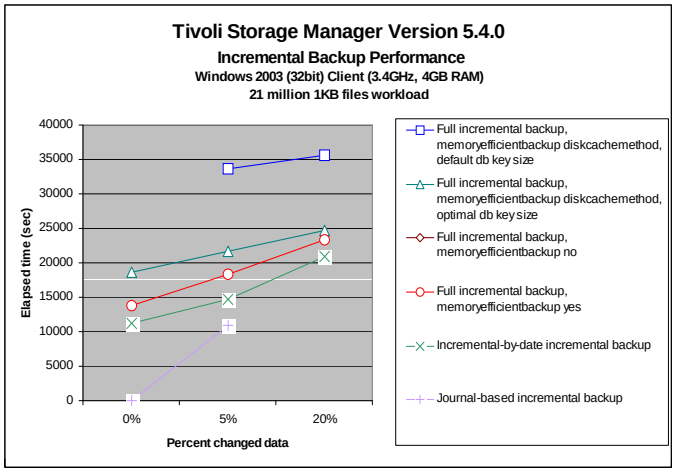


Improved incremental backup memory usage

Incremental backup, 21 million files workload, Win2003 server, Win2003 client, LAN, TCP/IP, Disk stgpool
IBM x360, 4-way 1.6GHz Xeon MP server, 6 GB, IBM x346, 2-way 3.4GHz client, 4GB, Compression No, Encryption No, Storage pool CRC No, Bufpoolsize 131072, Txngroupmax 256, TXNBytelimit 2097152, NTFS filespace

Description	Elapsed time (seconds)		
	0% changed data	5% changed data	20% changed data
Full incremental backup, memoryeff diskcachem, default db key size	n/a	33699	35536
Full incremental backup, memoryeff diskcachem, optimal db key size	18654	21678	24680
Full incremental backup, memoryefficient no	Failed	Failed	Failed
Full incremental backup, memoryefficient yes	13760	18394	23310
Incremental-by-date incremental backup	11262	14679	20902
Journal-based incremental backup	2	10916	Failed

- The 'memoryefficient no' method fails for this workload because it needs to use more virtual memory than is available to a 32 bit client
- The journal-based backup method fails when there are lots of changed files
- ✓ The disk cache method with optimal key size is only 18% slower than the 'memoryefficient yes' method at 5%

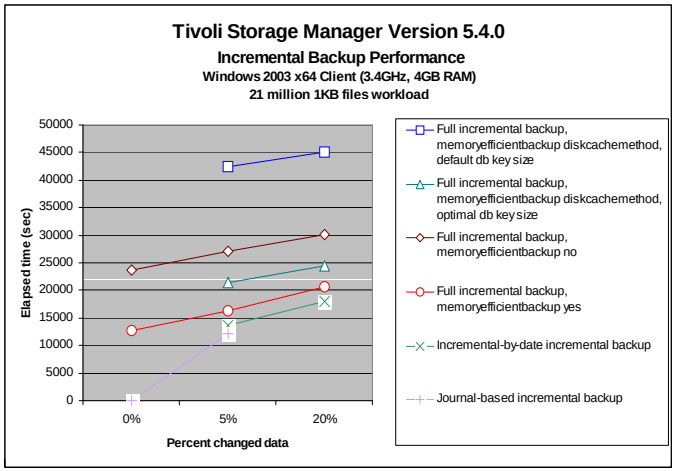


Improved incremental backup memory usage

Incremental backup, 21 million files workload, Win2003 server, Win2003 x64 client, LAN, TCP/IP, Disk stgpool
Intel 4-way 3GHz (Paxville dual core, hyperthread) server, IBM x346, 2-way 3.4GHz client, 4GB, Compression No, Encryption No, Storage pool CRC No, Bufpoolsize 131072, Txngroupmax 256, TXNBytelimit 2097152, NTFS filespace

Description	Elapsed time (seconds)		
	0% changed data	5% changed data	20% changed data
Full incremental backup, memoryeff diskcachem, default db key size	n/a	42496	45117
Full incremental backup, memoryeff diskcachem, optimal db key size	n/a	21327	24350
Full incremental backup, memoryefficient no	23652	27176	30088
Full incremental backup, memoryefficient yes	12724	16244	20714
Incremental-by-date incremental backup	n/a	13681	17974
Journal-based incremental backup	2	12075	Failed

- ✓ The disk cache method was faster than 'memoryefficientbackup no' method on Windows x64
- ✓ 64bit OS allows larger process virtual memory, but large real memory demand cause paging
- ✓ The disk cache method disk I/O is more efficient than paging disk I/O



Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

What Problem Are You Trying to Solve?

- RTO and RPO
- Progressive Incremental backup is still most comprehensive backup
  - Best method to detect any type of file change
  - Individual file restore
- $64,000 question is “Why can’t you use progressive incremental backup?”
  - Memory
  - Backup window

Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

tools

- tsmstats.ini
  - Introduced in TSM 5.4.0 Backup-Archive Client
- direx.exe
  - Coming soon from TSM
- others

tsmstats.ini

```
[fileSystemStatistics.\\patmos\m$]
maxPath=68
totalLocalFiles=1381
totalLocalDirs=552
maxDirCount=30
maxDirName=M:\test\Hiragana
```

Windows – In Your Toolkit

Progressive incremental backup	
Journal based backup	
Virtual mount points	
Adaptive sub-file backup	
Image backup (on-line and “used block only”)	

Windows – Attacking Memory Problems

- You need to solve the memory problem first
  - JBB needs progressive incremental before the journal is enabled
  - JBB will fall-back to a progressive incremental at some point
  - Incremental by date needs to use a periodic full progressive incremental at some point
- Is memoryefficient yes going to work
  - Run direx.exe to find out biggest directory bottleneck
    - Most objects in single directory * 300 < 2 GB ?
      - Then use memoryefficient yes
    - Most objects in single directory * 300 > 2 GB ?
      - Memoryefficient yes not going to solve the problem
        - > Use memoryefficient diskcache or
        - > Break-up file system or
        - > Use image incremental
- If memoryefficient backup is going to help, you can now consider JBB to further reduce backup window

Windows – Attacking the Backup Window

- Consider JBB
  - JBB will have to fall back to progressive incremental model at some point
    - Notification buffer overrun (this is a “safety valve”)
  - How often is this going to happen in your environment
  - How often can you tolerate this

JBB – Tracking Down the Buffer Overrun Safety Valve

- Run TSM journal daemon stand-alone (i.e., not communicating with the B-A client)
  - Look for notification buffer overflow errors in error log
- New TSM 5.4.1.2 trace flag helps you see activity in journal daemon
  - Can use to tune journal daemon exclude list

Creating Stand-Alone JBB Daemon

- Modify journal .ini file to journal one or more file systems
  - Can use configuration wizard (just don't start the service).
- Modify the .ini file
  - Add trace semantics (see next page)
  - Modify communications (named pipe) so that the B-A client can't talk to it
- Start the journal service
  - Alternatively run in foreground: *tsmjbbd.exe i*

tsmjbbd.ini

```
…
[JournalSettings]
Traceflags=jbbfilemoncsv
Tracefile=M:\jbbtrace
JournalPipe=\\.\pipe\bogusPipe
Errorlog=jbberror.log
JournalDir=C:\Program Files\Tivoli\TSM\baclient
…
```

Buffer Overrun Safety Valve (jbberror.log)

```
…
06/04/2007 15:48:02 fifoQInsert(00A4ED04): Queue threshold reached, thread 960 returned
from wait.
06/04/2007 15:48:02 fifoQInsert(00A4ED04): Queue threshold reached, thread 960 will block
until queue is under threshold.
06/04/2007 15:48:02 fifoQInsert(00A4ED04): Queue threshold reached, thread 960 returned
from wait.
06/04/2007 15:48:05 psFsMonitorThread(tid 960): Notification buffer overrun for monitored
FS 'C:\', journal will be reset.
06/04/2007 15:48:05 jnlDbCntrl(): Resetting journal for fs 'C:'.
…
```

Now What?

- How often are you seeing buffer overruns?
  - More often then backup window?
    - you will not be able to use JBB effectively
  - Sometimes?
    - You have to decide if it is acceptable
  - Never?
    - WOOHOO!

    - JBB is a good fit
- Don’t forget you can tune JBB’s exclude list and eliminate unnecessary journal entries

Image Backup

- Consider using image
  - Can’t resolve the memory problems
  - Too many changes (> 30%) for JBB or progressive incremental backup
  - Most of the file system is small files (avg. size < 1 MB)
  - Need faster recovery time then file-level restore can provide
- Try to keep the file change rate < 30% when using image and incremental combinations

Windows

- Other thoughts
  - Compression … if you have relatively new CPU and no hardware compression… why not?
  - Take care of your exclude lists
    - Aggressive use of EXCLUDE.DIR where possible
  - Open File Support
    - Are there files that are open that you need protected, e.g., ntuser.dat on workstations?
    - … and can you not get the files off-line in some other manner, e.g., presched command?

Agenda

- Backup techniques
  - What was available before ADSM/TSM came on the scene
  - TSM Progressive incremental backup model and its derivatives
  - Image backup
- How to determine which backup technique best fits
  - What problem are you trying to solve
  - Windows
  - Other operating systems

AIX – In Your Toolkit

Progressive incremental backup	
Journal based backup	
Virtual mount points	
Adaptive sub-file backup	
Image backup	

Linux – In Your Toolkit

Progressive incremental backup	
Journal based backup	
Virtual mount points	
Adaptive sub-file backup	
Image backup (on-line)	

UNIX – In Your Toolkit

Progressive incremental backup	
Journal based backup	
Virtual mount points	
Adaptive sub-file backup	
Image backup (off-line)	

Macintosh and NetWare – In Your Toolkit

Progressive incremental backup	
Journal based backup	
Virtual mount points	
Adaptive sub-file backup	
Image backup	

Other Platforms – Attacking Memory Problems

- You need to solve the memory problem first
  - JBB (AIX) will have to be enabled; and it will fall back to a progressive incremental model at some point
  - Incremental by date needs to use a periodic full progressive incremental at some point
- Is memoryefficient yes going to work
  - Run direx.exe to find out biggest directory bottleneck
    - Most objects in single directory * 300 < 2 GB ?
      - Then use memoryefficient yes
    - Most objects in single directory * 300 > 2 GB ?
      - Memoryefficient yes not going to solve the problem
        - > Use memoryefficient diskcache or
        - > Break-up files system or
        - > Use image incremental
- If memoryefficient backup is going to help, now consider JBB (AIX) to further reduce backup window

AIX – Attacking the Backup Window

- Consider JBB
  - JBB will have to fall back to progressive incremental model at some point
    - Notification buffer overrun (this is a “safety valve”)
  - How often this will happen in your environment
  - How often can you tolerate this

JBB – Tracking Down the Buffer Overrun Safety Valve

- Run TSM journal daemon stand-alone (i.e., not communicating with the B-A client)
  - Look for notification buffer overflow errors in error log
- New TSM 5.4.1.2 trace flag helps you see activity in journal daemon
  - Can use to tune journal daemon exclude list

AIX - Creating Stand-Alone JBB Daemon

- Modify journal .ini file to journal one or more file systems
- Modify the .ini file
  - Add trace semantics (see next page)
  - Rename daemon so that the B-A client can't talk to it
- Start the journal daemon

tsmjbbd.ini

```
…
[JournalSettings]
Traceflags=jbbfilemoncsv
Tracefile=M:\jbbtrace
Errorlog=jbberror.log
…
```

Now What?

- How often are you seeing buffer overruns?
  - More often then backup window? You will not be able to use JBB effectively
  - Sometimes? You have to decide if it is acceptable
  - Never? WOOHOO! JBB is a good fit

- Don’t forget you can tune JBB’s exclude list and eliminate unnecessary journal entries

Image Backup – non-Windows

- Consider using image
  - Can't resolve the memory problems
  - Too many changes (> 30%) for JBB or progressive incremental backup
  - File system is at least 60% full
    - UNIX and Linux paying for bandwidth by backing-up unused blocks
  - Where no on-line image support is available, you can unmount the file system
  - Most of the file system is small files (ave. < 1 mb)
  - Need faster recovery time then file-level restore can provide
- Try to keep the file change rate < 30% when using image and incremental combinations

Other Platforms

- Other thoughts
  - Compression … if you have relatively new CPU and no hardware compression… why not?
  - Take care of your exclude lists
    - Aggressive use of EXCLUDE.DIR where possible
  - Snapshot-based file backup or using *snapshotroot* option with 3rd party snapshot provider
    - Are there files that are open that you need protected?
    - … and can you not get the files off-line in some other manner, e.g., presched command

Conclusion

- Myriad of backup techniques in TSM and more to come
- One size does not fit all
- Start with progressive incremental backup at it is still most comprehensive backup
- Move to other progressive incremental or image techniques as the need arises