

TECHWAVE²⁰⁰⁸

SYBASE USER TRAINING & SOLUTIONS CONFERENCE

Backup Server Internals & Troubleshooting

Bret Halford

Sybase, Inc.

Principal Product Support Engineer (ASE)



Backup Server Internals & Troubleshooting

The core of this presentation was written a long time ago by Robert Pickering, to whom I am greatly indebted. He did such a good job I have only had to make minor updates regarding new features and removing obsolete references to such things as the VAX/VMS platform.

The format has been adjusted to fit the screen.

No animals were hurt during the production of this presentation.

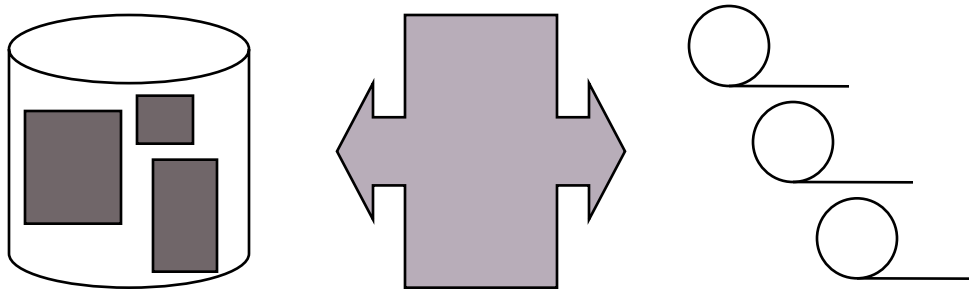
Concessions are available in the Lobby.

Some History ...

- The need to keep copies of important data has been around since ancient times ... even before computers !
- Sybase SQL Server allows backup and restore of databases via the T-SQL *dump* and *load* commands.
 - Transaction logs can also be backed up and restored provided they are on a separate segment to the data
 - ASE dumps allows backup of database/log in parallel with normal database operation ... no need to quiesce.
- Prior to System 10, SQL Server did all the work during backup and restore
 - Lots of I/O overhead for SQL Server
- System 10 introduced Backup Server to handle all this I/O

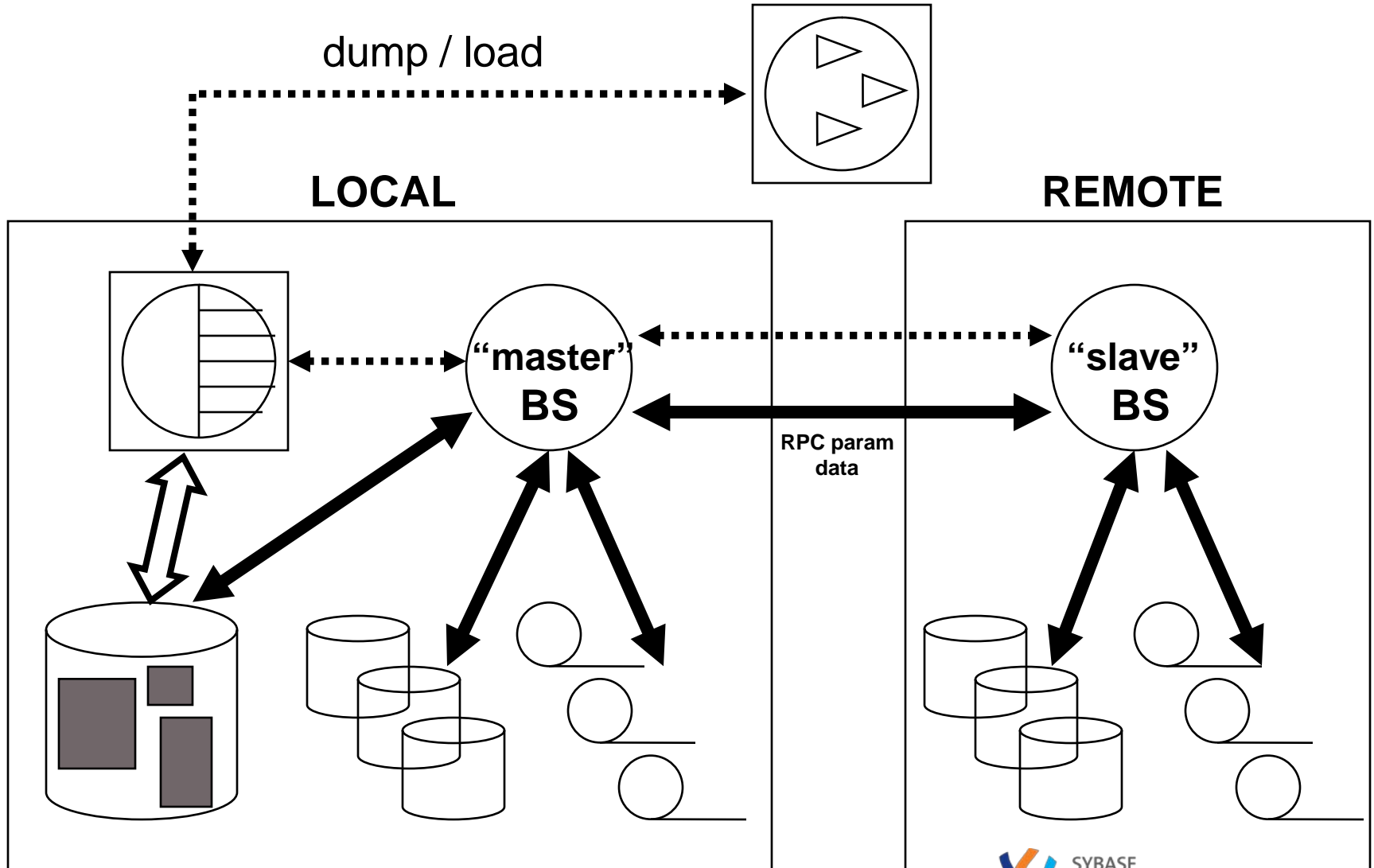
I/O Processing With Backup Server

The next slides deal with the major function of Backup Server ...

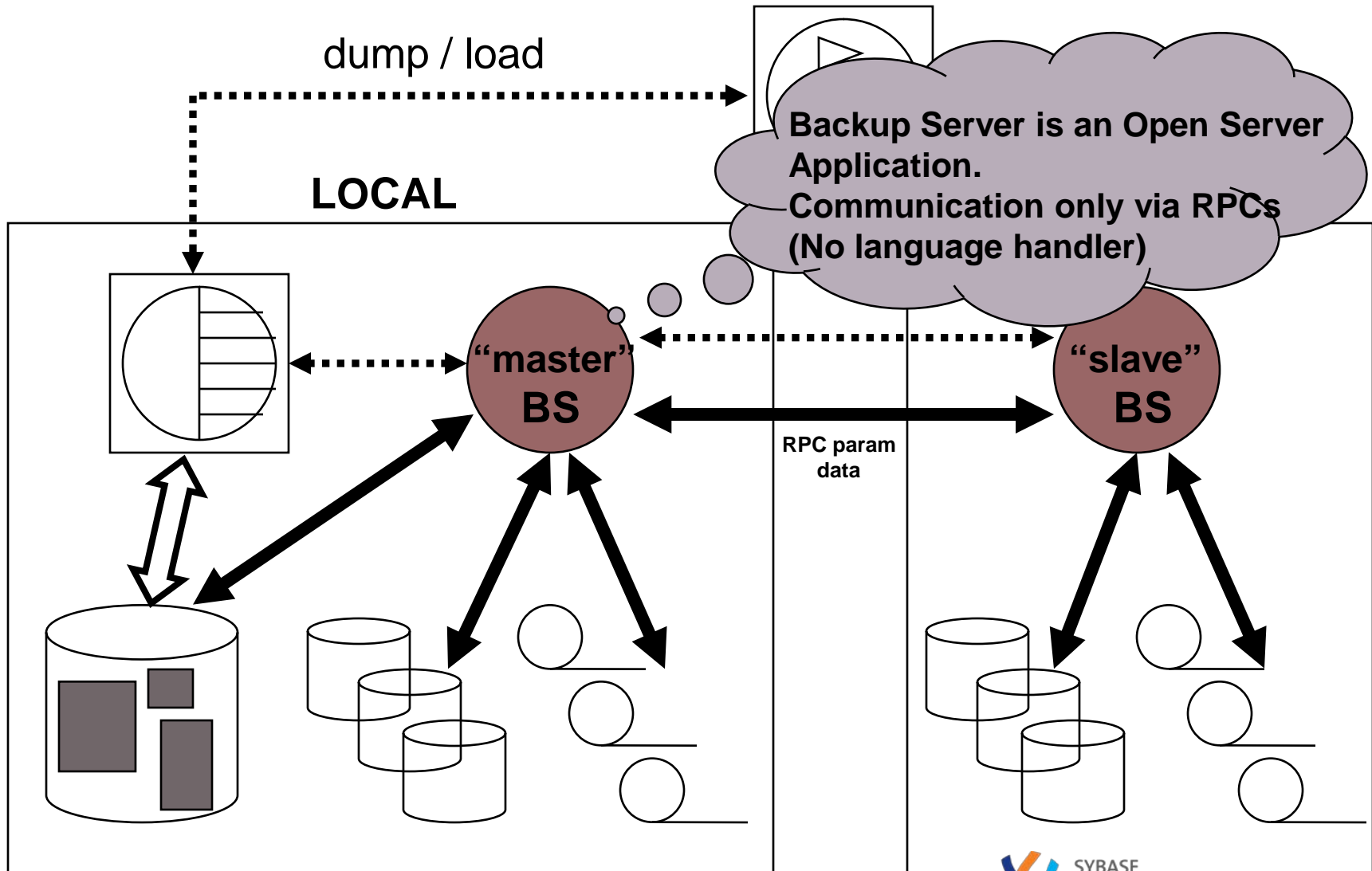


***Moving large volumes of data between
database and archive devices***

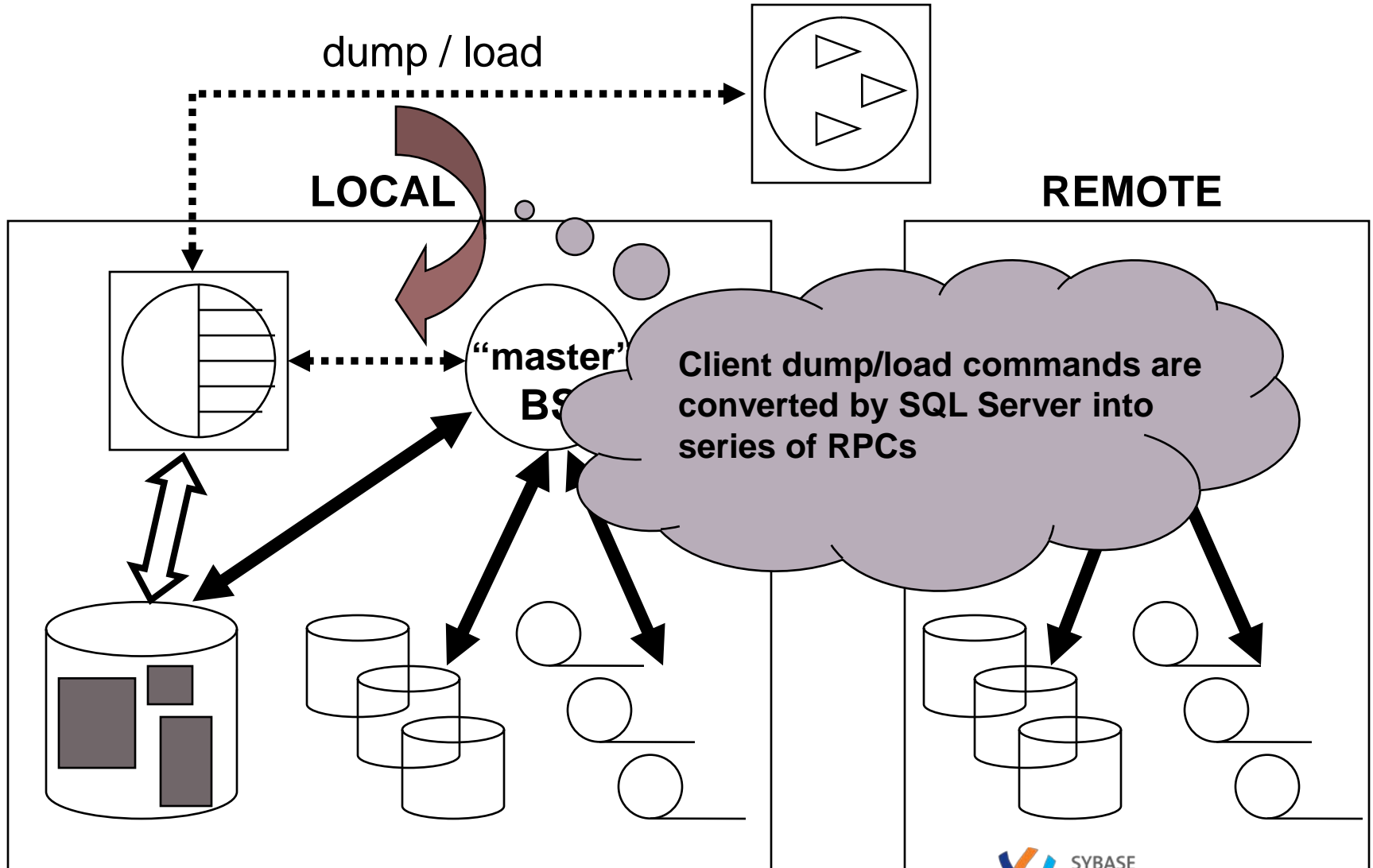
The "Big Picture"



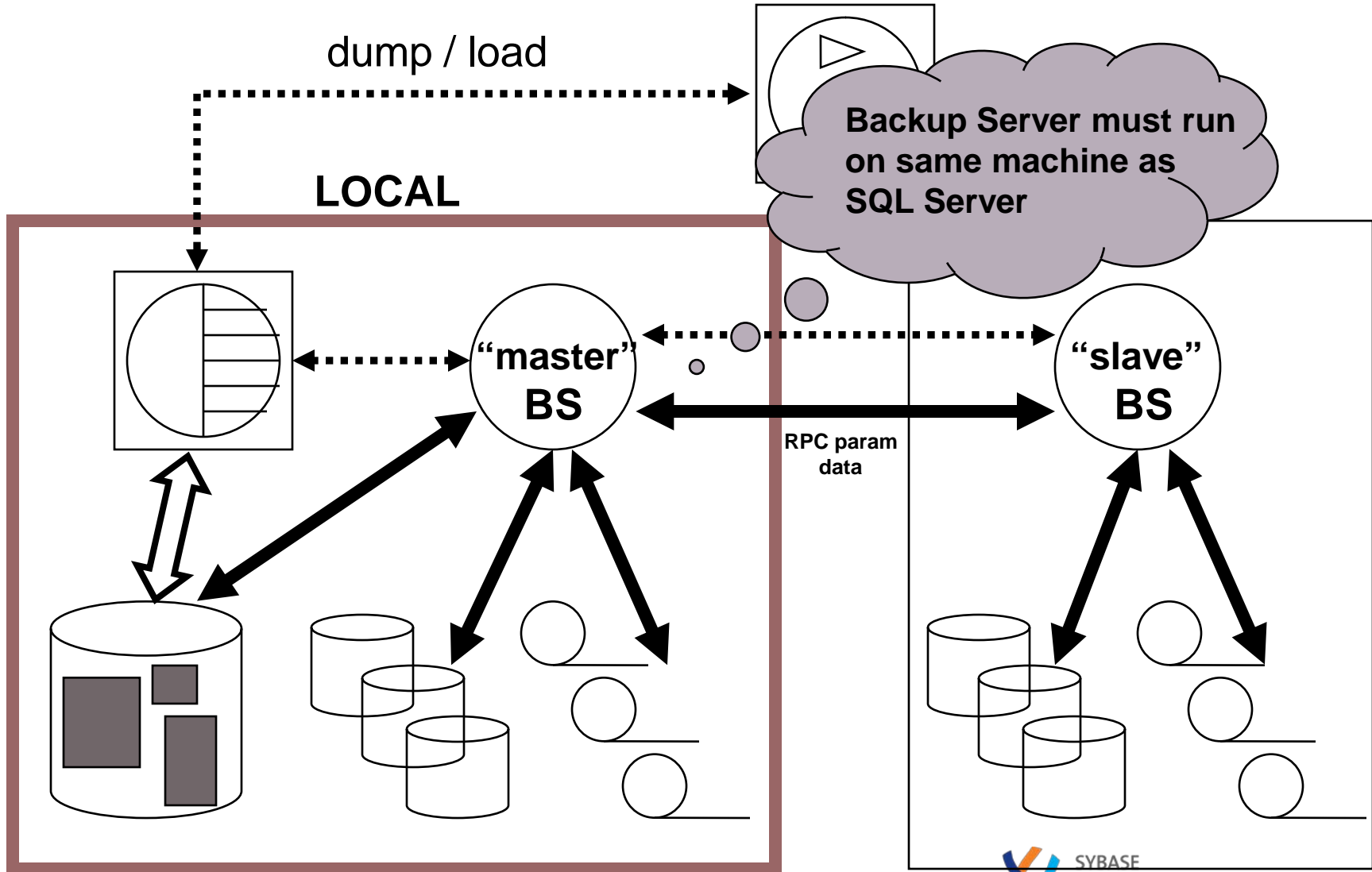
The "Big Picture"



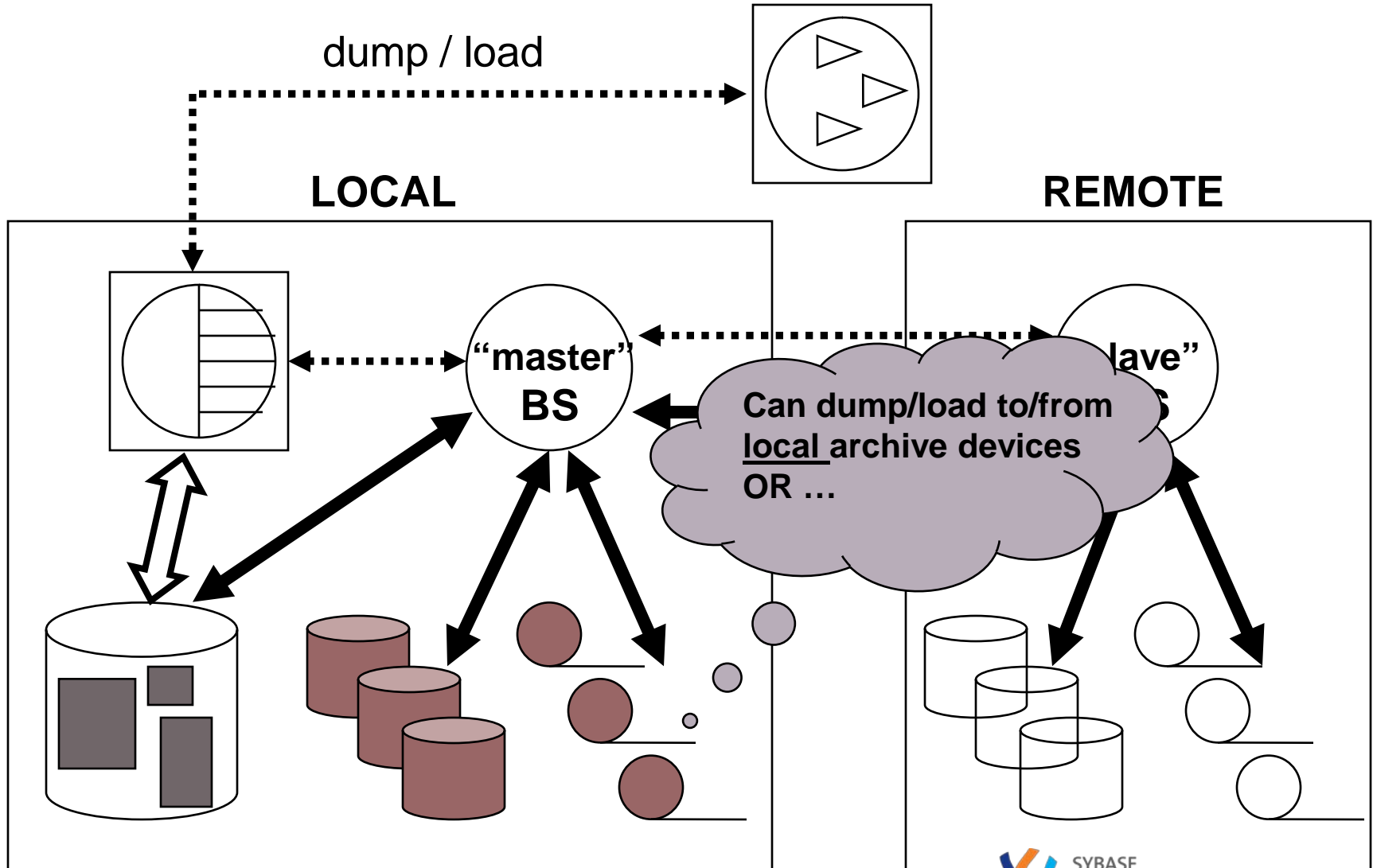
The "Big Picture"



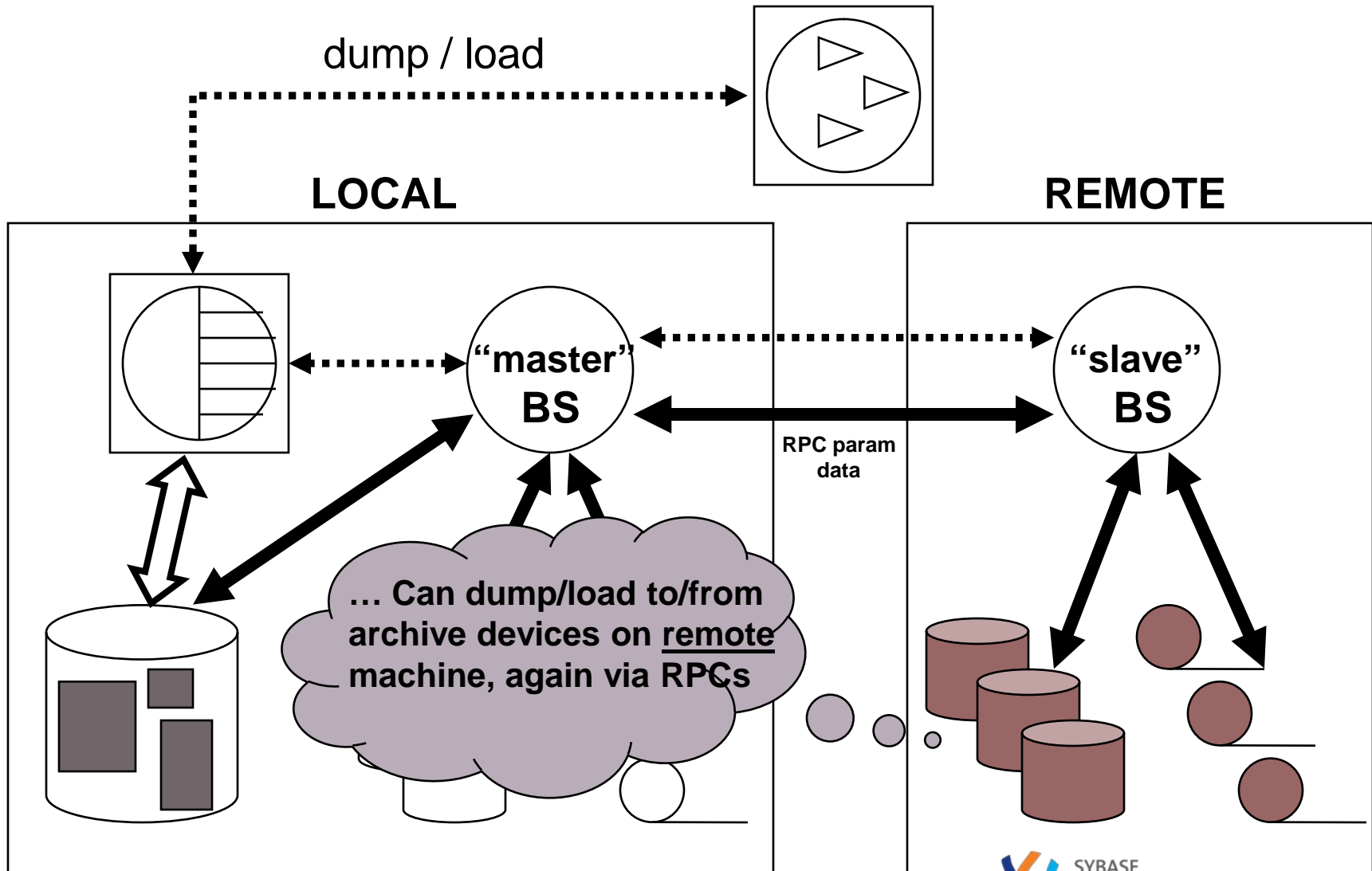
The "Big Picture"



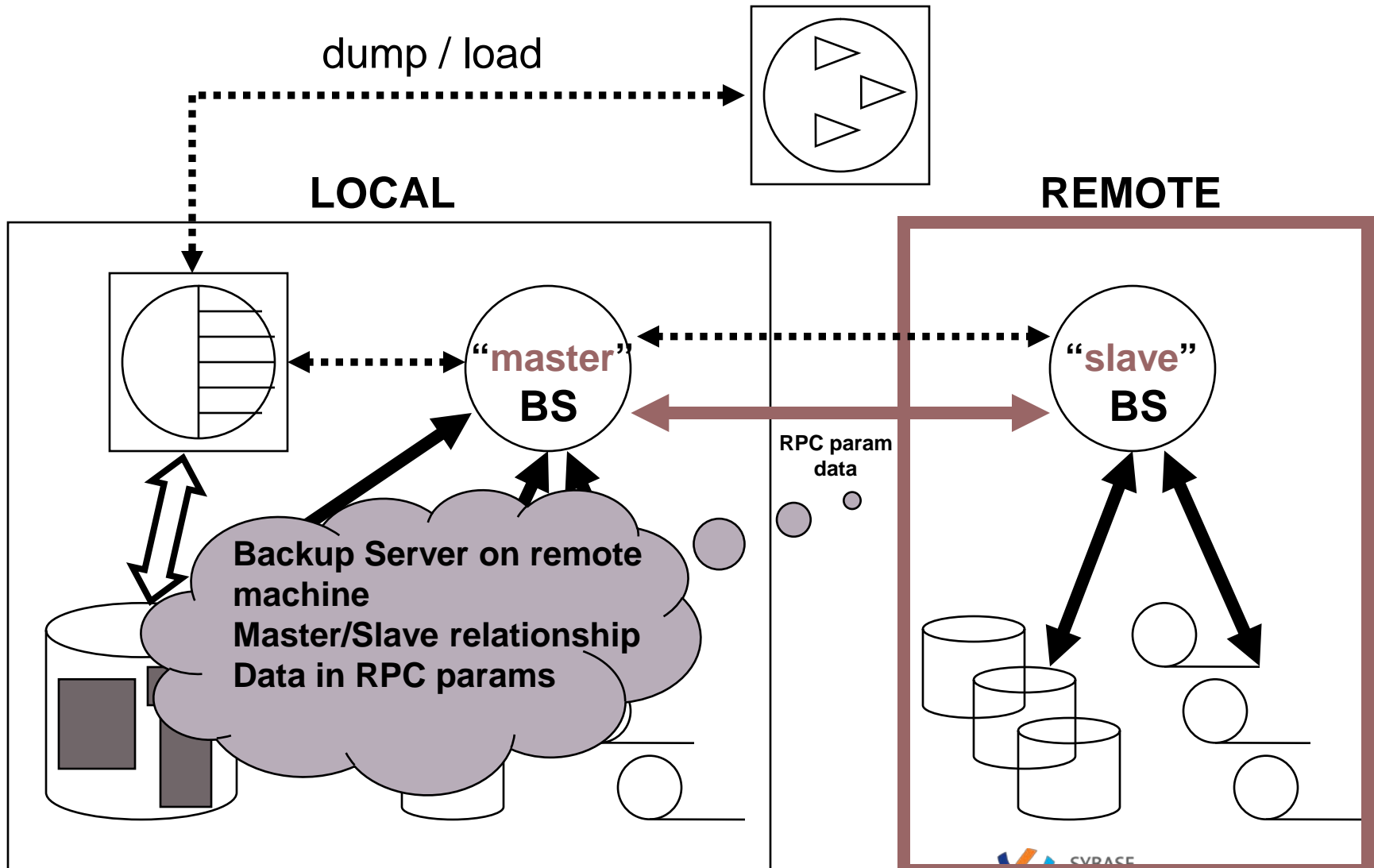
The "Big Picture"



The "Big Picture"

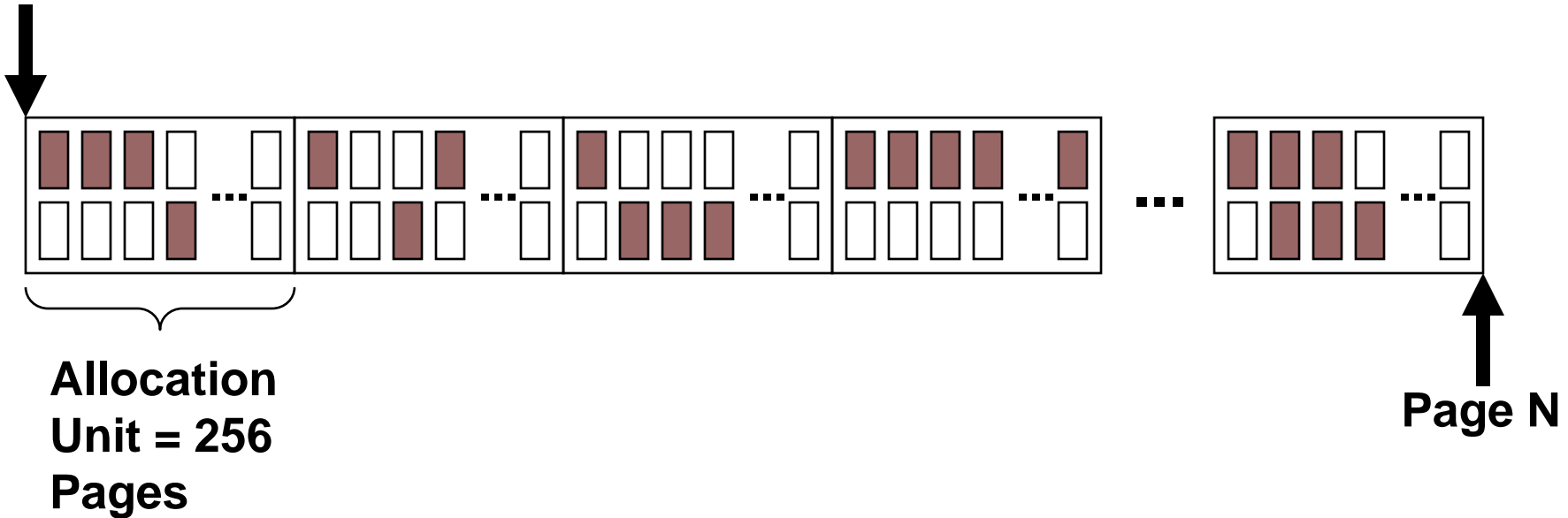


The "Big Picture"



Sybase Database Structure - Review

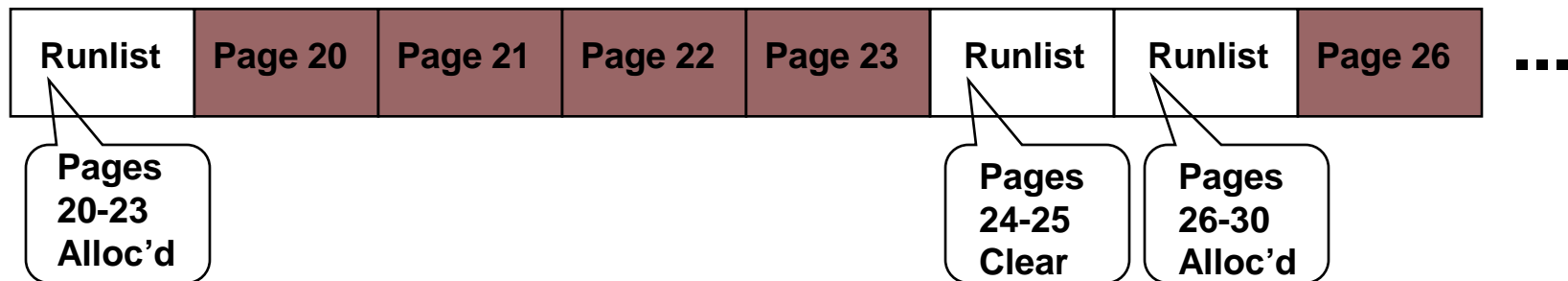
Page 0



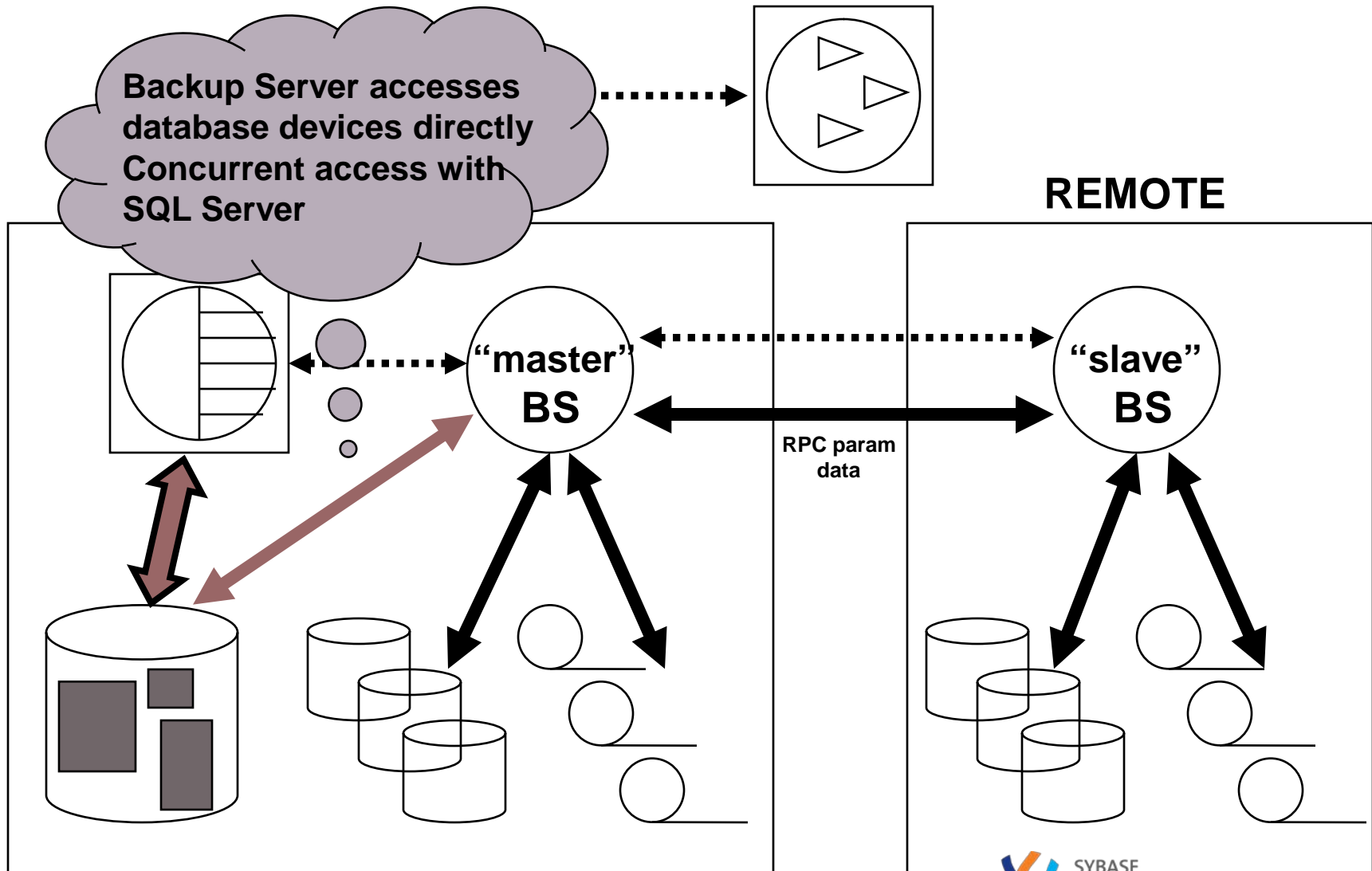
- Page is smallest unit of storage
- 2KB in size
- Database = contiguous sequence of logical pages
- **Allocated** and unallocated pages within database

Backup Server and Database Structure

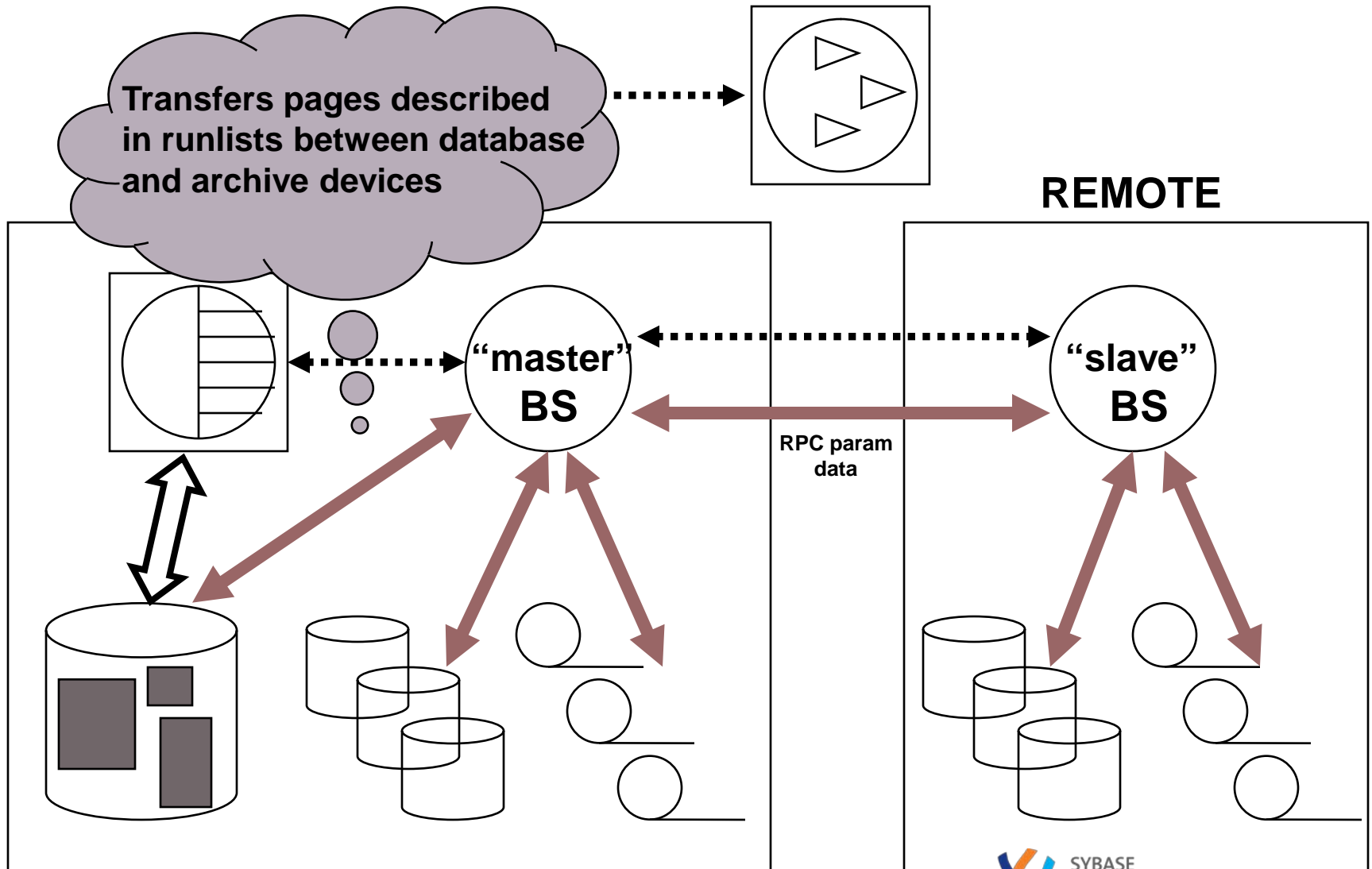
- Generally only allocated pages in the database are dumped to archive devices however, there was a performance enhancement in ##### that does large i/o that can include unallocated pages
- Lists of unallocated pages are recorded to be zeroed-out upon load
- “runlists” describe sequences of pages
- What is written to the dump is a runlist, followed by its data pages (if any), then another runlist followed by its data pages, etc.



The "Big Picture"



The "Big Picture"



Significant Options on dump/load

Example :

```
dump database mydb
```

```
to stripe_dev1 [ at remote_bs ]
```

```
[ stripe on stripe_dev2 [ at remote_bs ] ]
```

```
[ stripe on stripe_dev3 [ at remote_bs ] ]
```

- Applicable to both dump and load commands
- Applicable whether dump/load database OR transaction

Significant Options on dump/load

Example :

```
dump database mydb
```

```
to stripe_dev1 [ at remote_bs ]
```

```
[ stripe on stripe_dev2 [ at remote_bs ] ]
```

```
[ stripe on stripe_dev3 [ at remote_bs ] ]
```

- Archive devices `stripe_dev1`, `stripe_dev2`, `stripe_dev3`
- Physical name (eg. `/dev/rmt/0n`) OR
- Logical name in `sysdevices` via `sp_addumpdevice` (local only)

Significant Options on dump/load

Example :

```
dump database mydb
```

```
to stripe_dev1 [ at remote_bs ]
```

```
[ stripe on stripe_dev2 [ at remote_bs ] ]
```

```
[ stripe on stripe_dev3 [ at remote_bs ] ]
```

- For remote dump/load, specify remote Backup Server name
- Name must be in interfaces file
- Be consistent with names, avoid aliases etc ... more later

Significant Options on dump/load

Example :

```
dump database mydb
```

```
to stripe_dev1 [ at remote_bs ]
```

```
[ stripe on stripe_dev2 [ at remote_bs ] ]
```

```
[ stripe on stripe_dev3 [ at remote_bs ] ]
```

- **Striping to multiple archive devices**
- **Just one reason ... PERFORMANCE !**
- **Database is “split” and written in parallel to N archive devices**

Significant Options on dump/load

Backupserver API

You can specify a Backupserver API Module as a stripe device.
The most common API module is the Sybase-provided “compress”

dump database mydb

```
to "api_name::<data>"  
[ stripe on "api_name:: <data>" ]  
[ stripe on "api_name::<data>" ]
```

Example:

dump database mydb

```
to "compress::4::/work/mydb_stripe1.cmp"  
to "compress::4::/work/mydb_stripe1.cmp"
```

Significant Options on dump/load

With Password

*dump database mydb to /work/mydump
with password = "foobar"*

The password option prevents the casual loading of a dump file without the password, but it isn't strong protection.

It does not, for instance, encrypt the dump, so the contents of the dump can still be explored using hex editor.

Encrypted data in the database remains encrypted in the dump.

Significant Options on dump/load

With Compression

*dump database mydb to /work/mydump
with compression = 4*

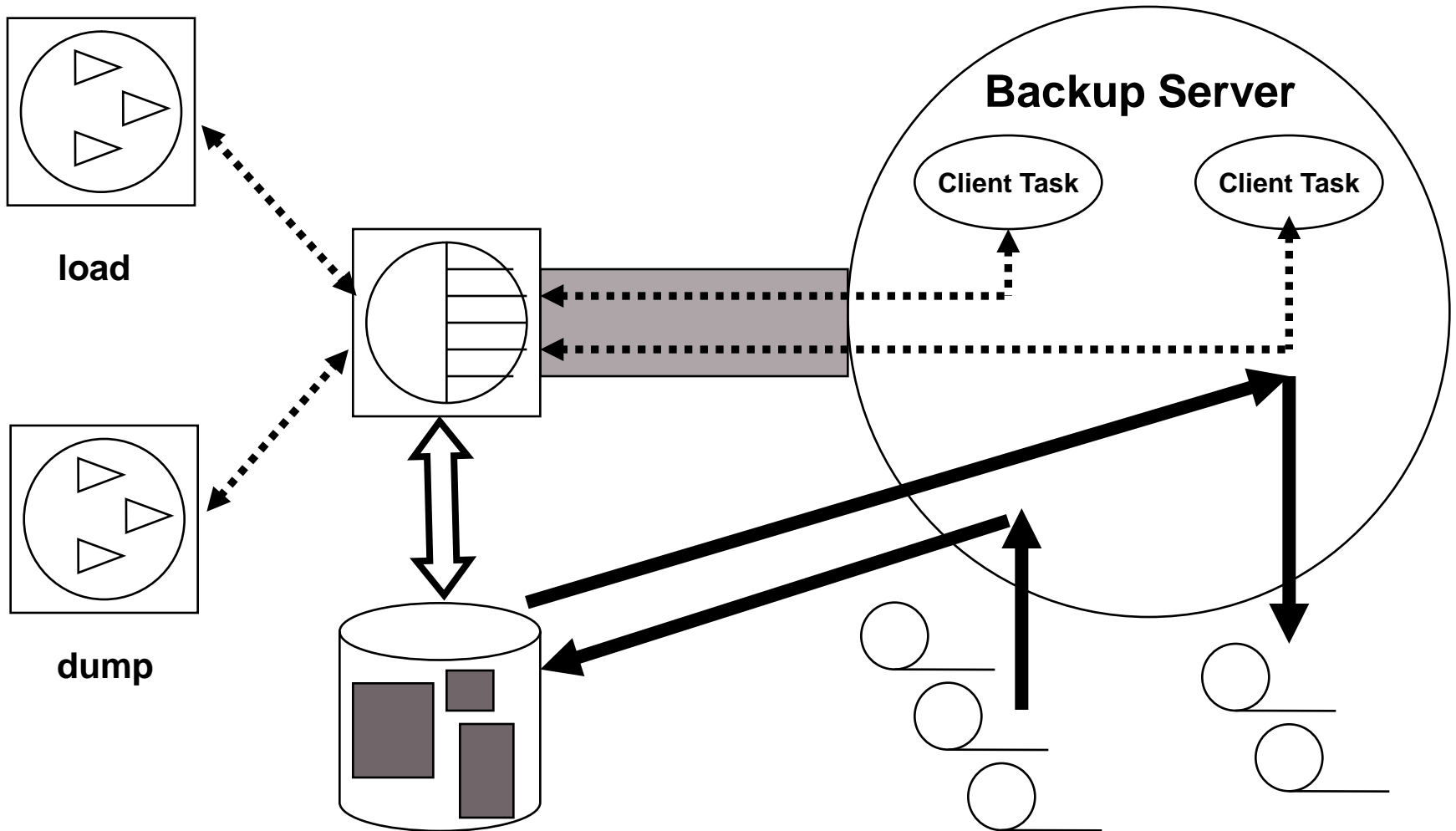
Also known as “native” compression, the functionality is similar to the “compress::” backupserver API.

The main advantage is that native compression can be used with a remote backupserver, but API modules cannot.

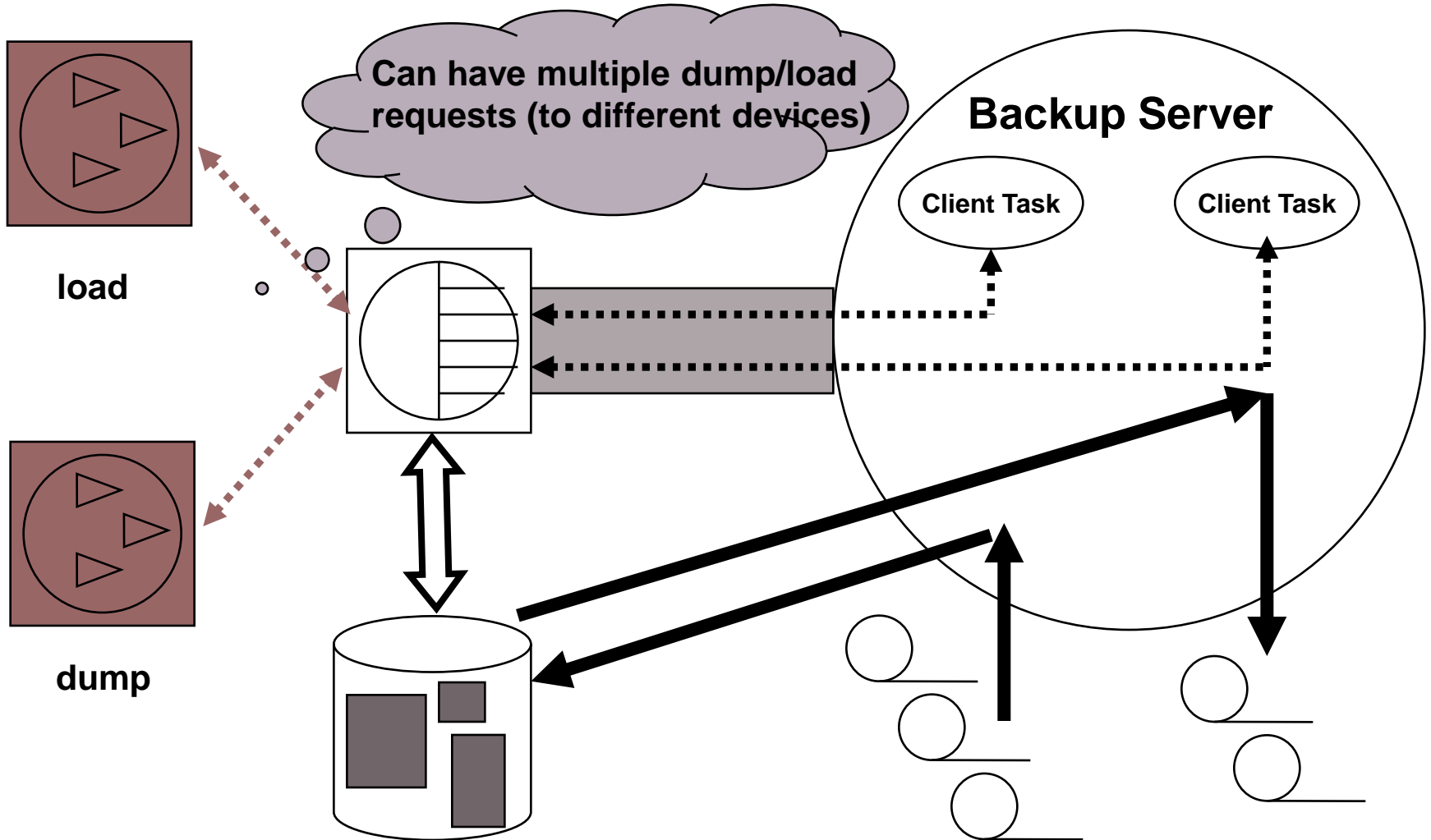
Placeholder for other newer features

- Placeholder slide for other newer features.

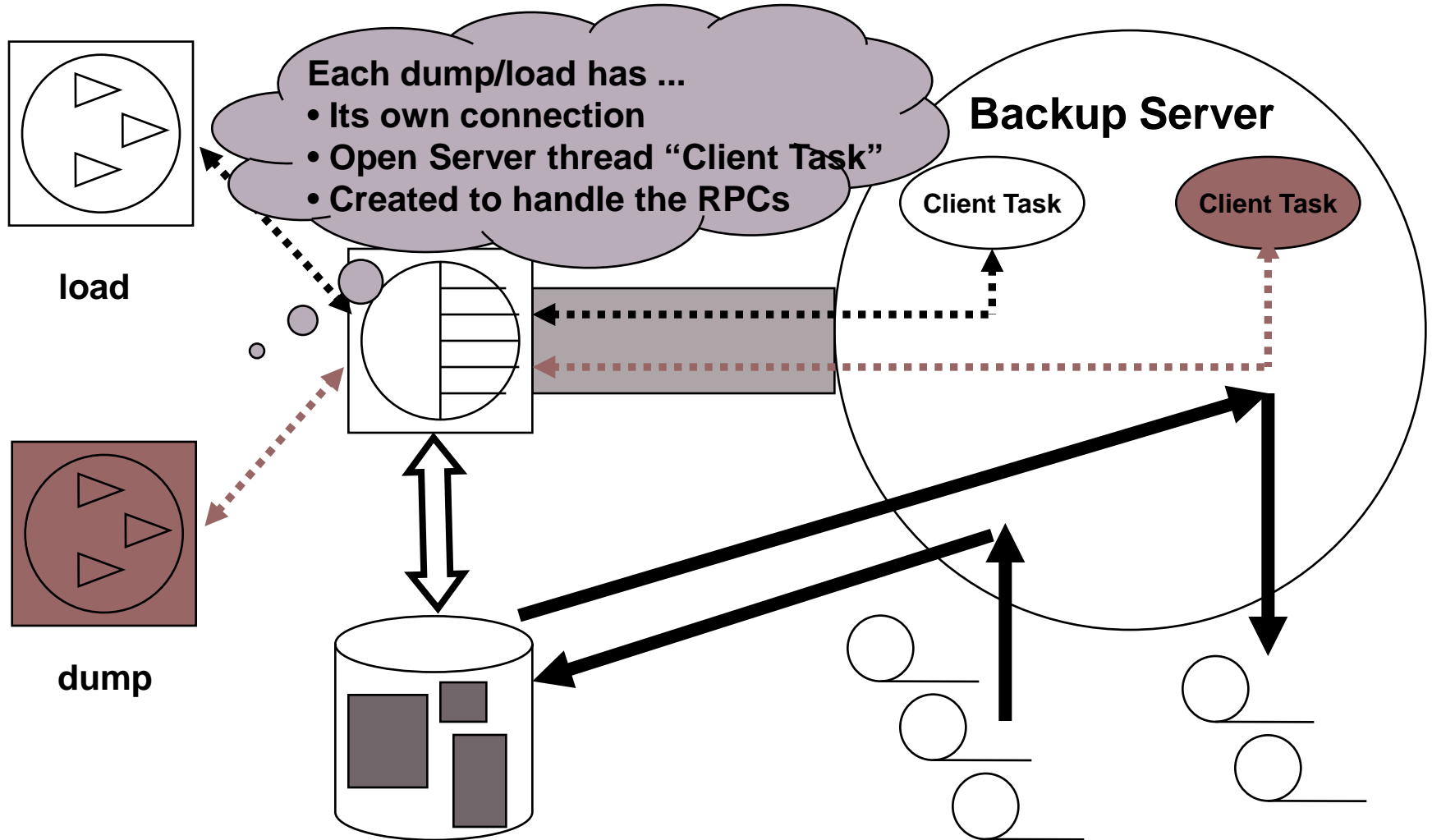
Connections, Sessions and Tasks - 1



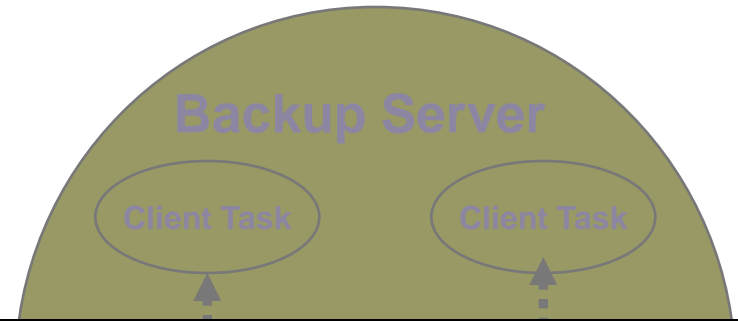
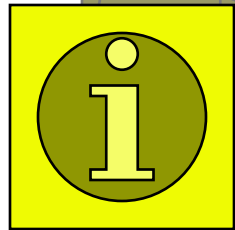
Connections, Sessions and Tasks - 1



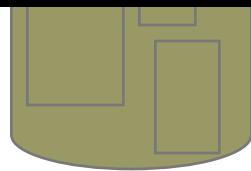
Connections, Sessions and Tasks - 1



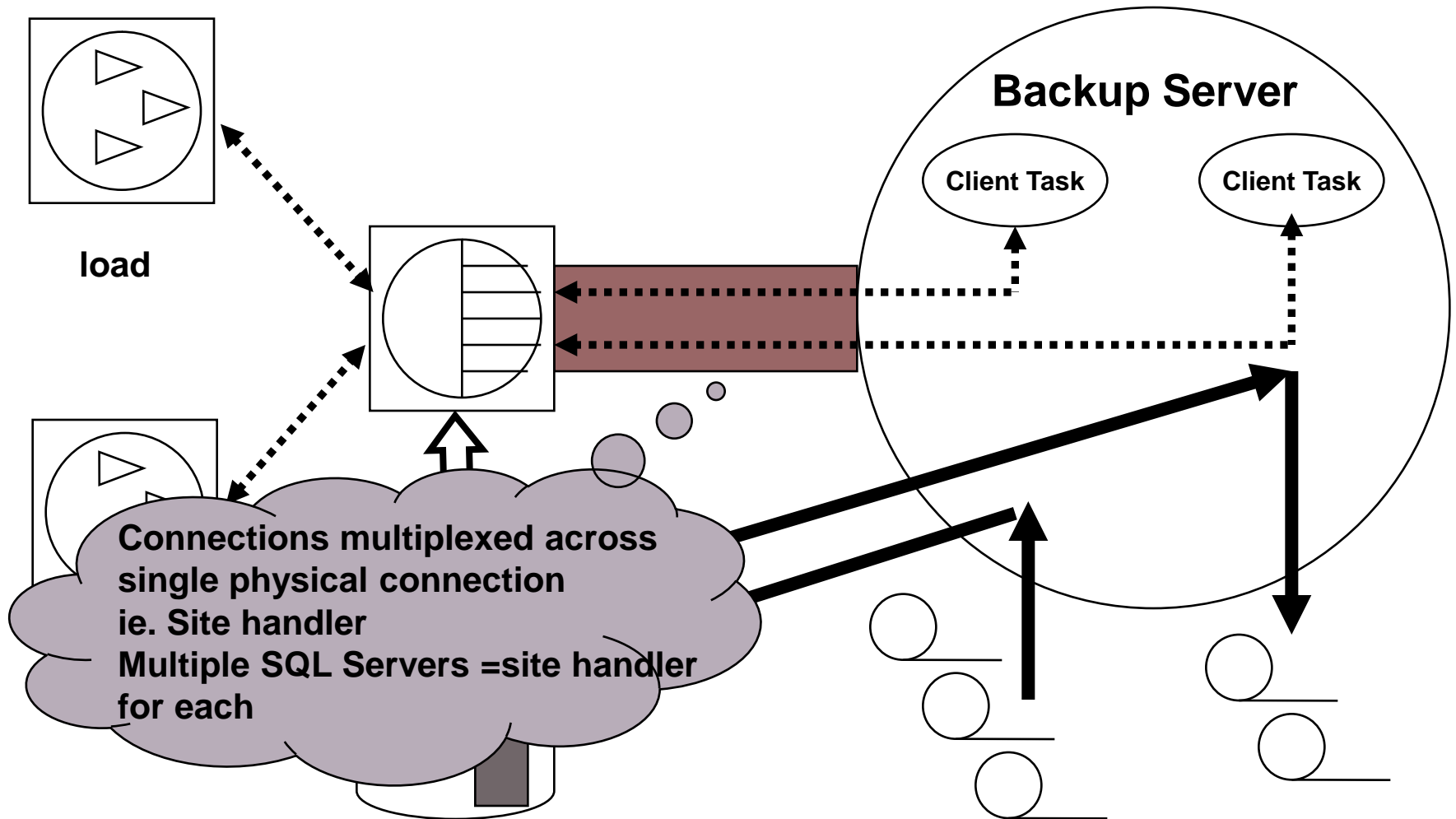
Connections, Sessions and Tasks - 1



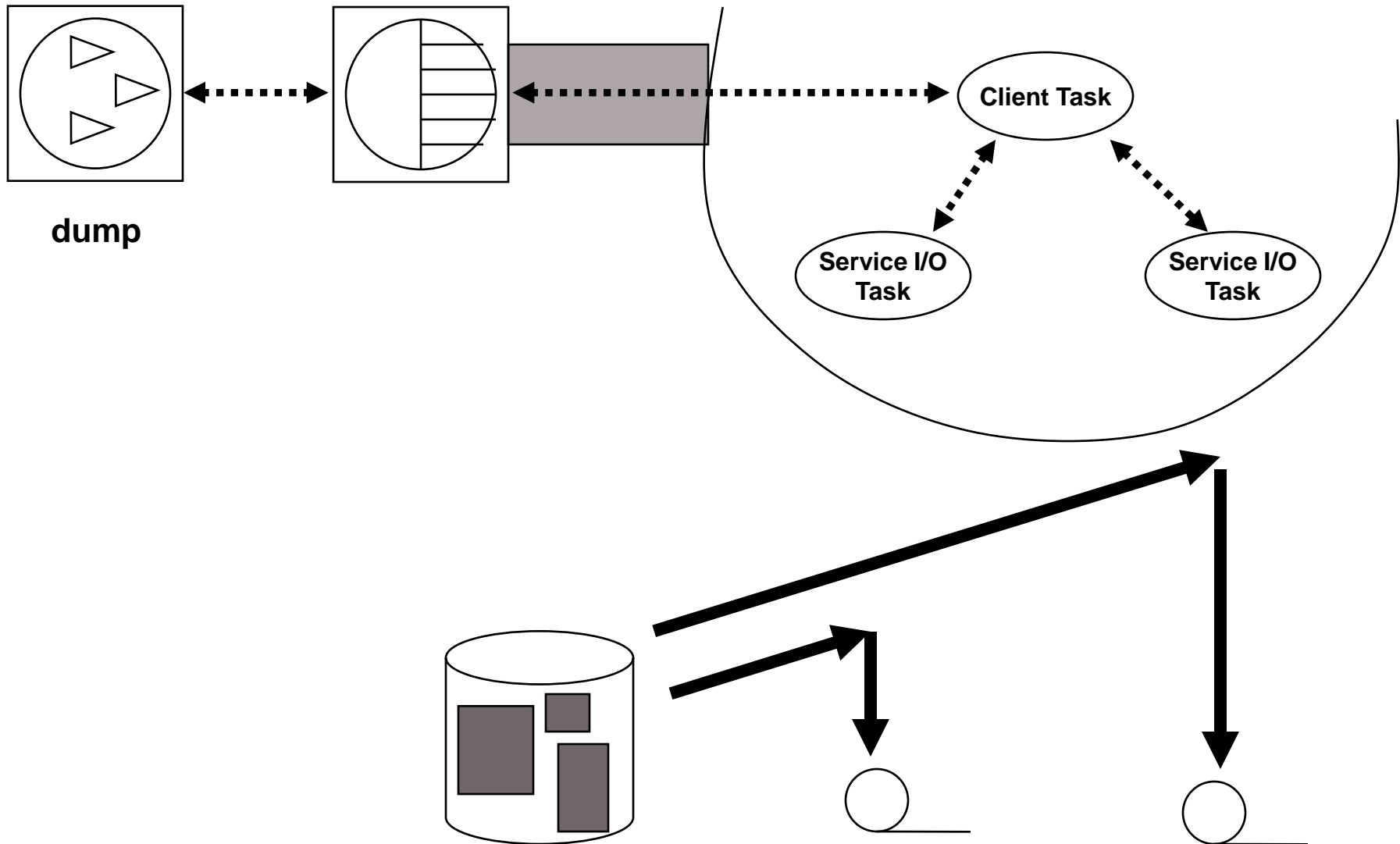
- **Connections referred to as “Sessions”**
- **Each has unique “Session id”**
- **SQL Server actually opens 2 connections**
 - **Primary used most of time**
 - **Secondary in case first gets blocked (eg. Backup Server scanning database mapping)**
- **Might be additional connections (eg. for volume changed replies)**



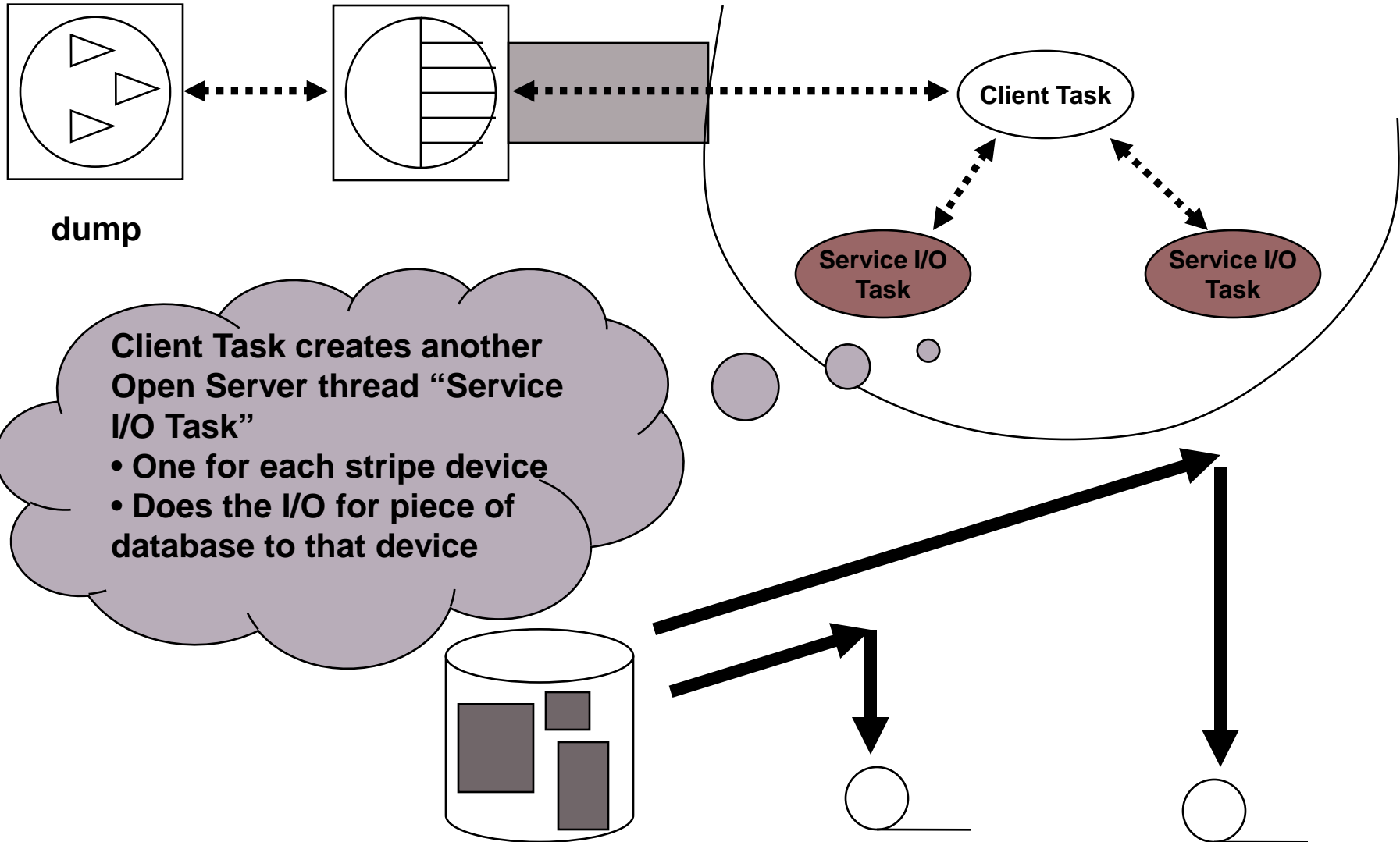
Connections, Sessions and Tasks - 1



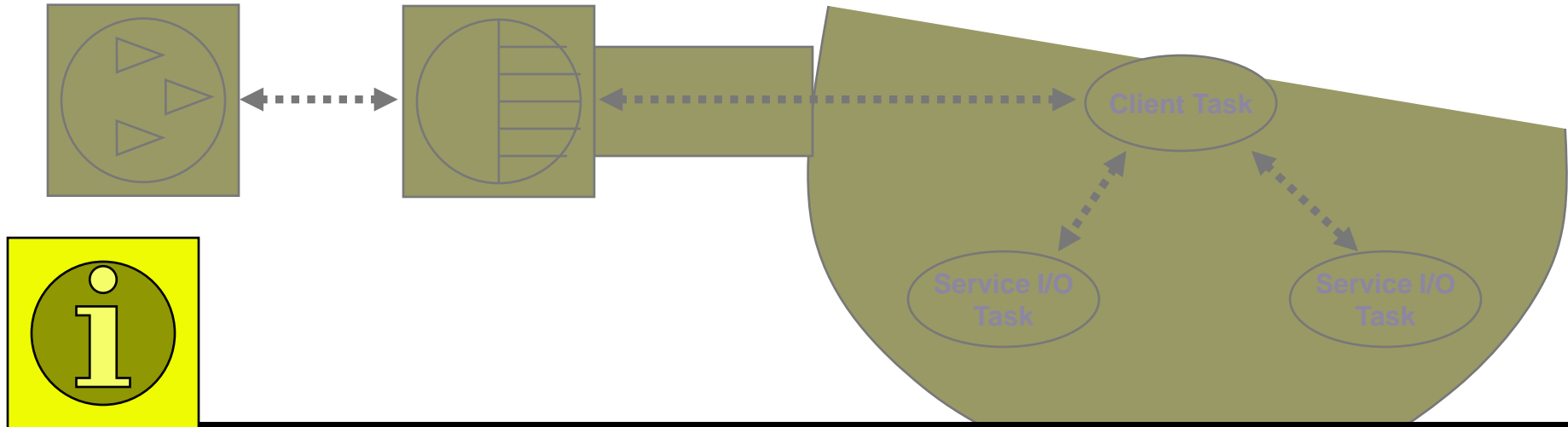
Connections, Sessions and Tasks - 2



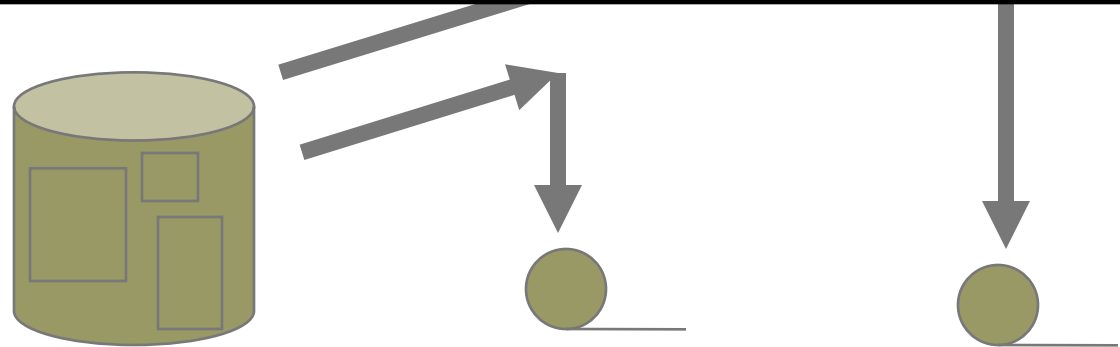
Connections, Sessions and Tasks - 2



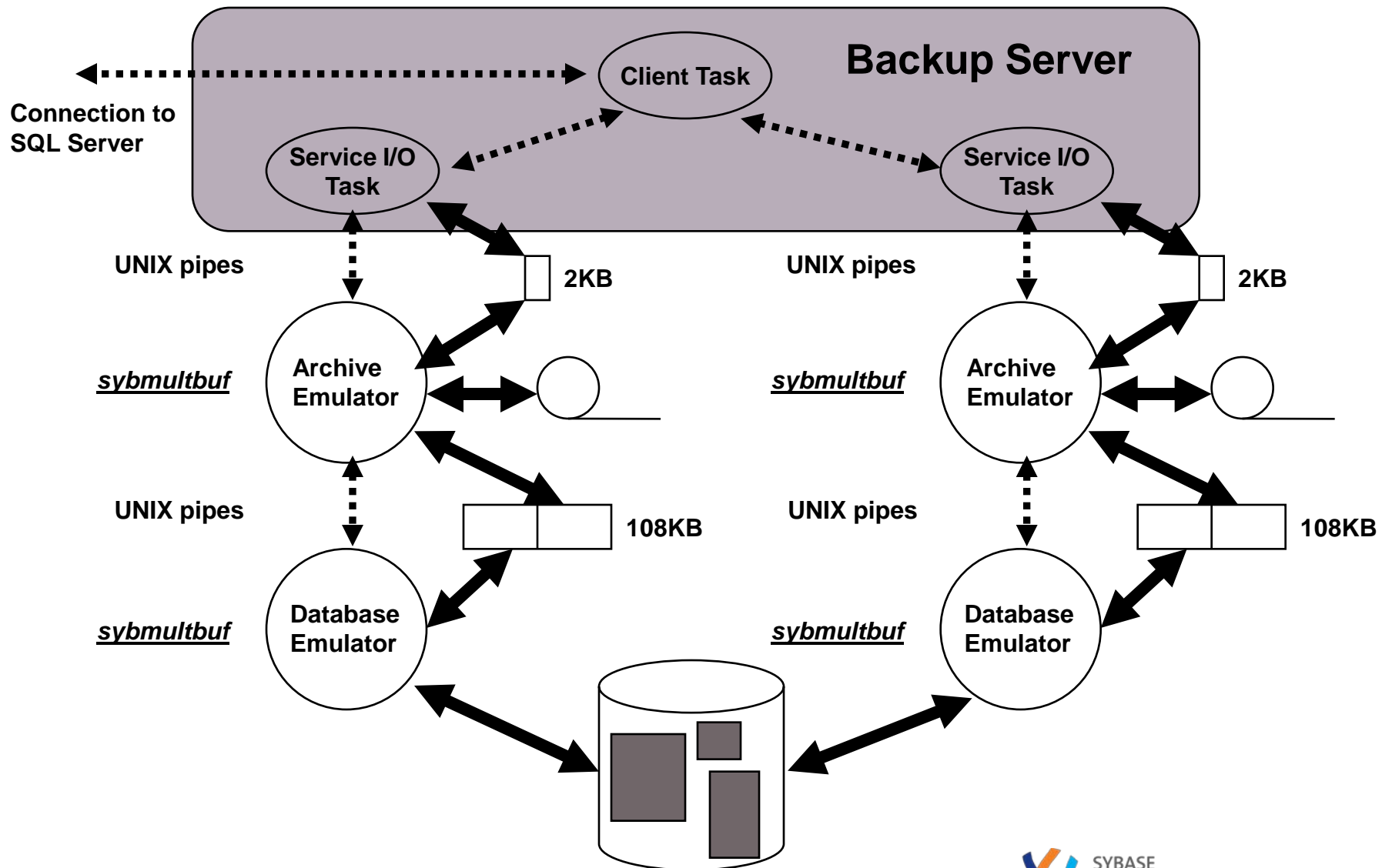
Connections, Sessions and Tasks - 2



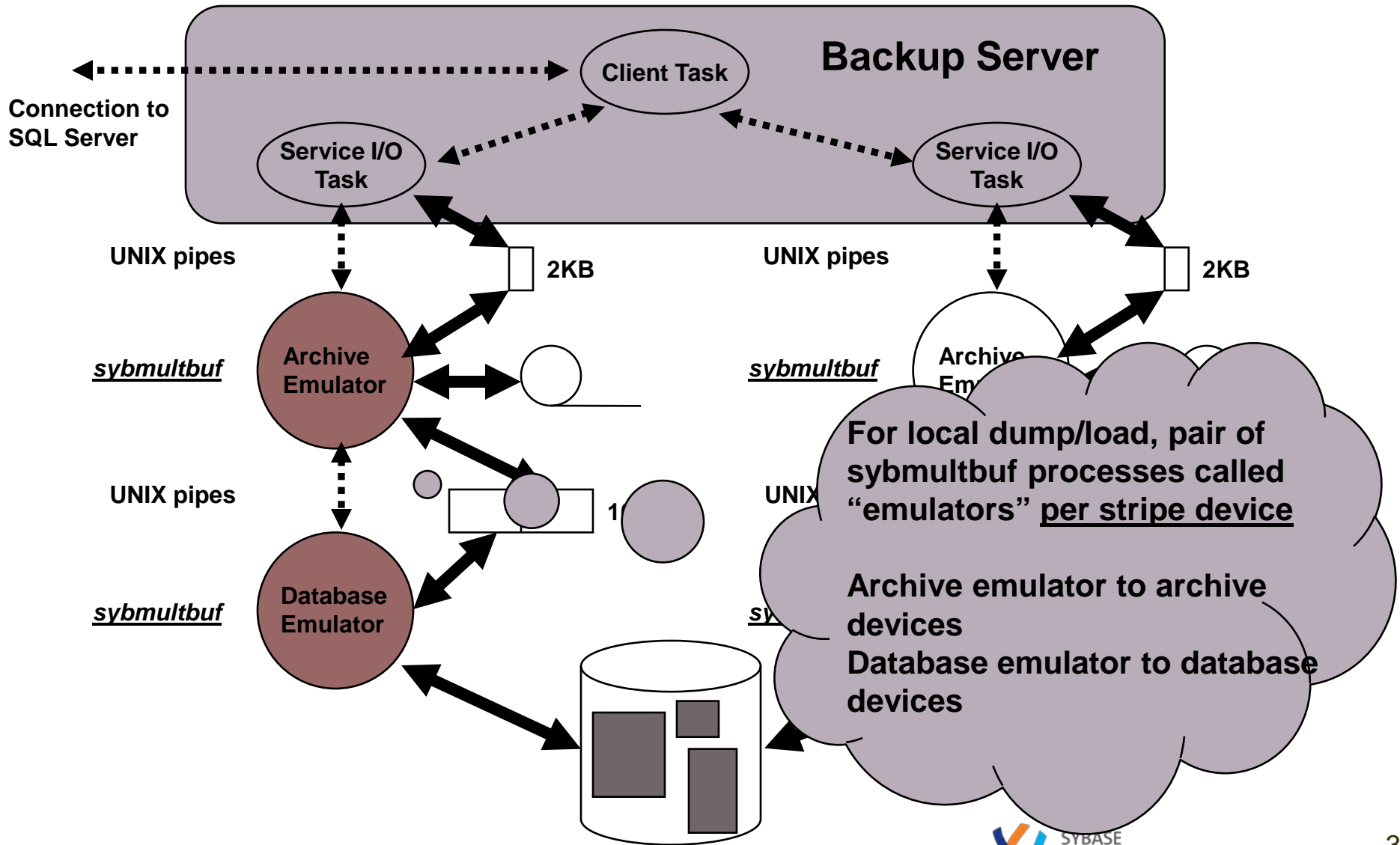
- Client Task sends global directives to all its Service I/O Tasks
- Waits for acknowledgement from each
- At end of dump/load session, all these threads terminate



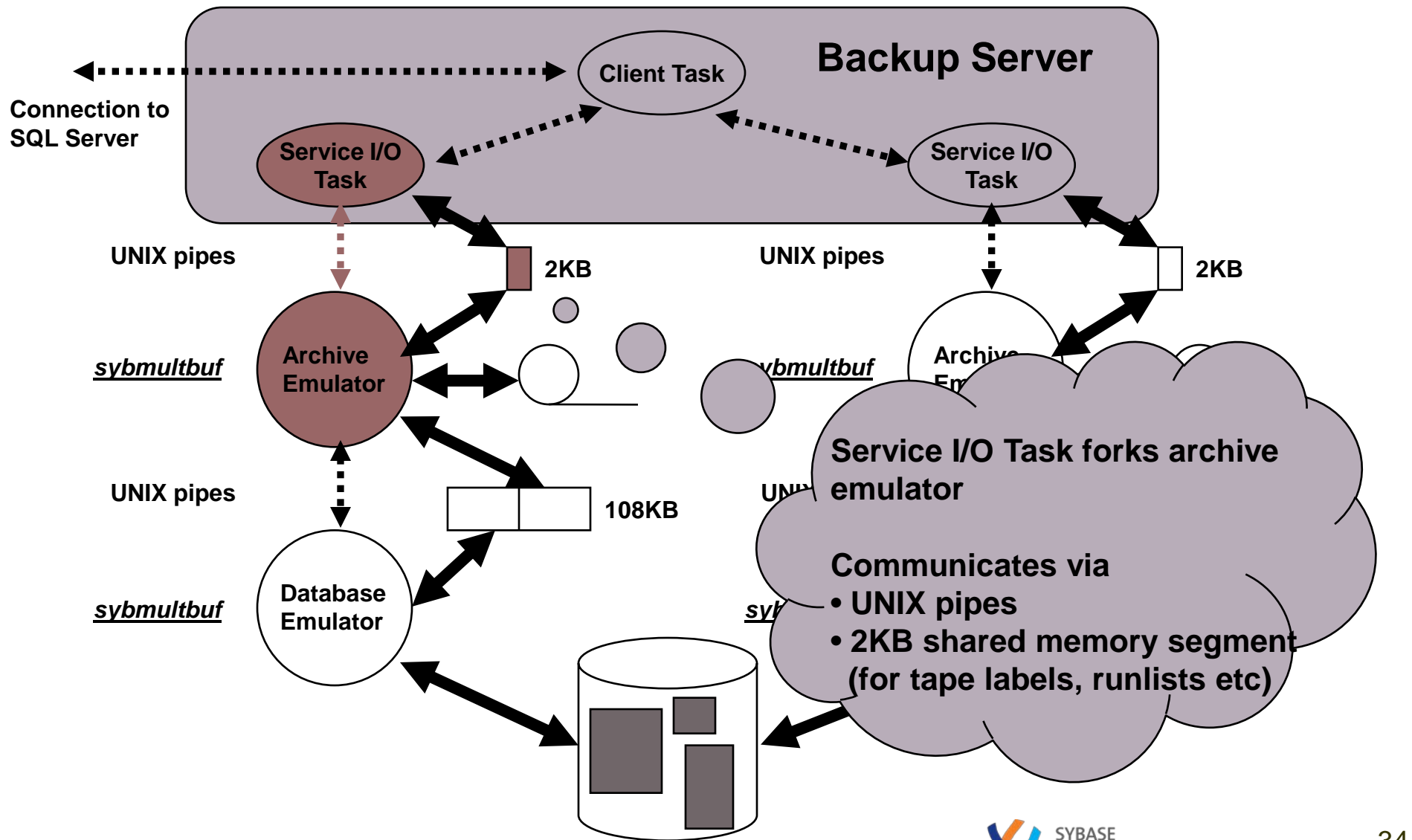
UNIX Emulated Multi-buffering



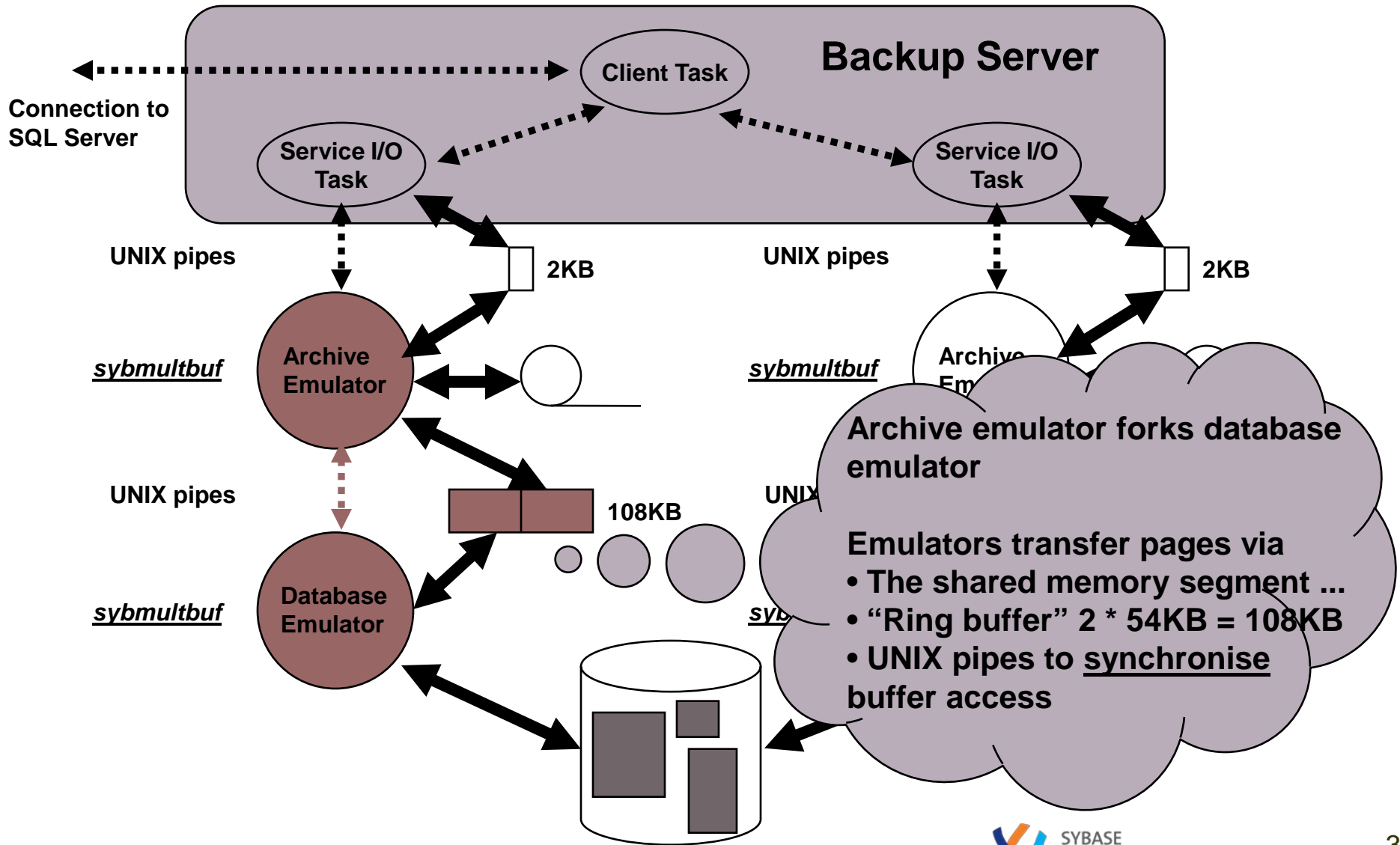
UNIX Emulated Multi-buffering



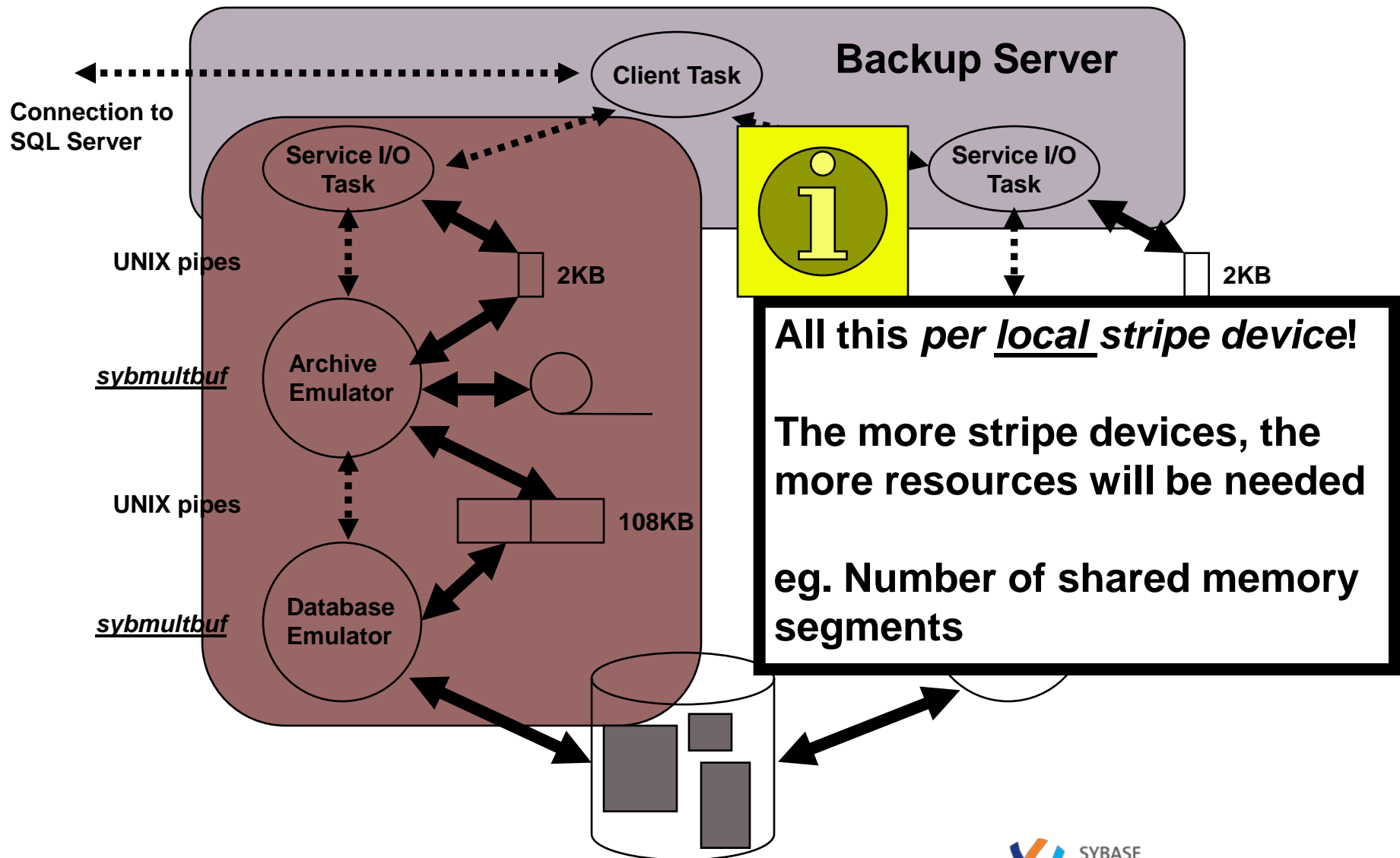
UNIX Emulated Multi-buffering



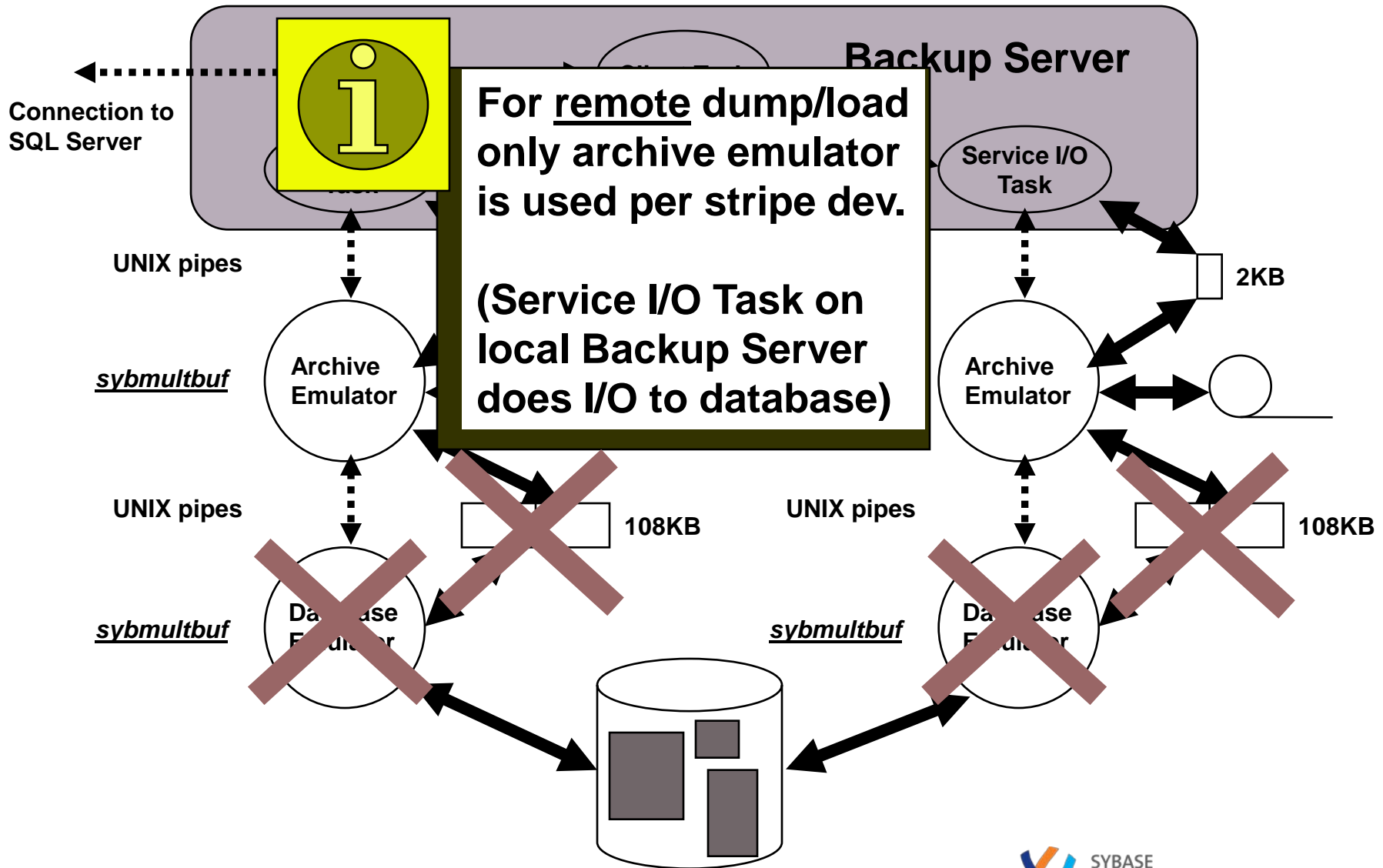
UNIX Emulated Multi-buffering



UNIX Emulated Multi-buffering



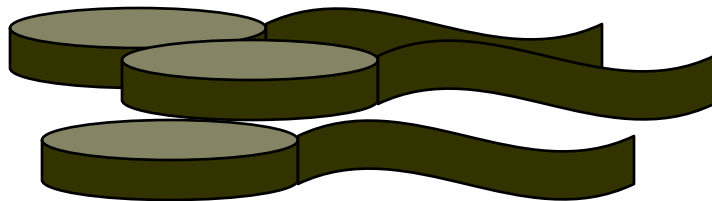
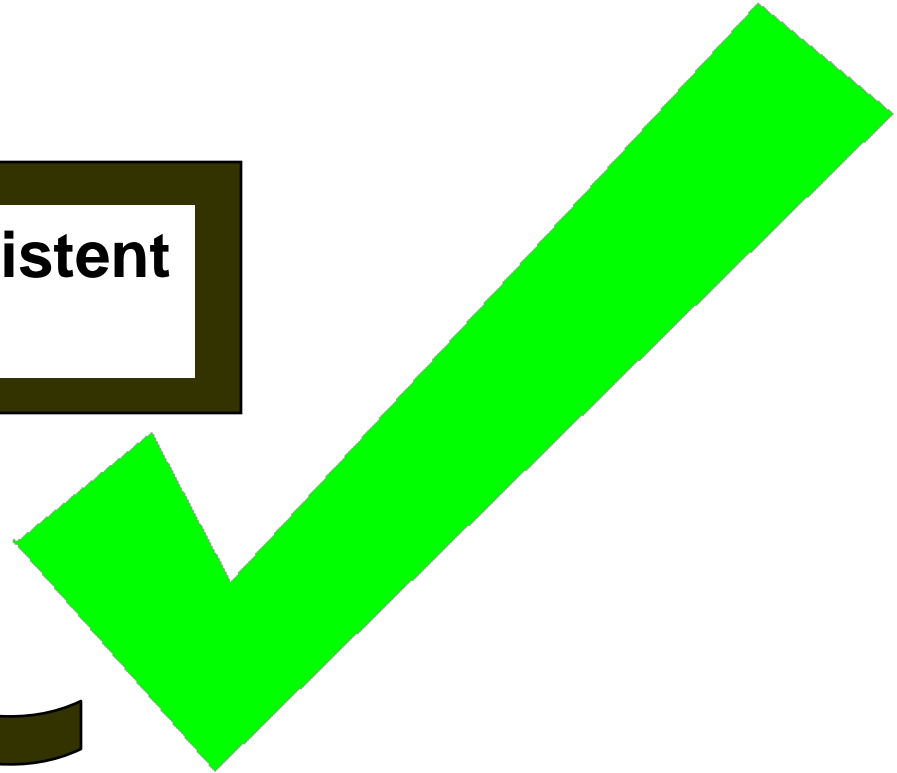
UNIX Emulated Multi-buffering



Ensuring A Recoverable Dump



**Transactionally-consistent
view of database ?**



Ensuring A Recoverable Dump

- Previous slides showed how Backup Server does I/O processing
- **QUESTION:** How to ensure that the pages written to a dump reflect a transactionally-consistent view of the database?
- Especially since dump is concurrent with regular database access
- **ANSWER:** Extra logic and synchronisation is required...

Dump Phases

dump database

1) DBPAGES

Dumps all pages in database
(except log if separate segment)

2) FLUSHPAGES

Dumps any pages flushed to disk
during DBPAGES that avoided
logging (eg. fast bcp, select into,
text pages, page splits, index create)

3) SCANLOGPAGES

Dumps chain of log pages from
oldest active xact (before DBPAGES
phase) to last page of log

Primary phase

Secondary phase

*(ensures that the dump includes
all pages both logged and non-logged
that were written or re-written during
the primary phase)*

dump transaction

1) SCANLOGPAGES

Dumps chain of log pages from
oldest active xact to last log page

Dump Phases - Rules

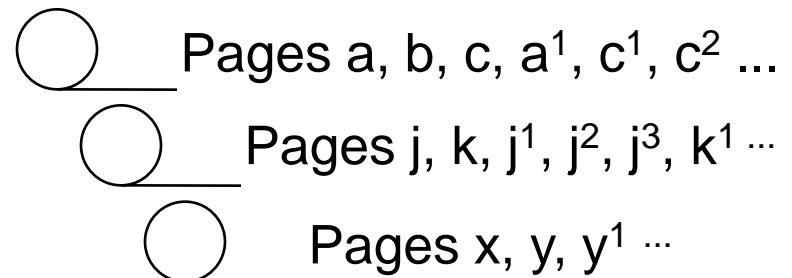
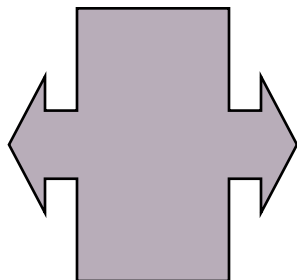
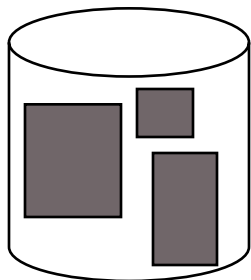
- In general, pages in a phase can be dumped in any order
- But all pages in phase N must be dumped before phase N+1
- Similar for load, all pages for phase N must be loaded before phase N+1
- Phase N must end before phase N+1 begins

Dump Phases - Processing

- In primary phase, SQL Server does not send runlists
- Instead, Backup Server scans disk pieces and transfers pages itself (faster than overhead of sending runlists)
- In secondary phase, SQL Server always performs runlist generation
- Runs supplement pages dumped in primary phase
- During dump, runs for a future phase may arrive before that phase has begun
- Backup Server saves these until specified phase begins

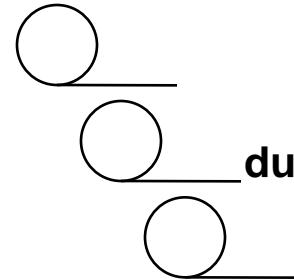
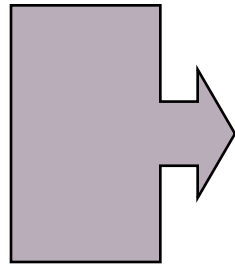
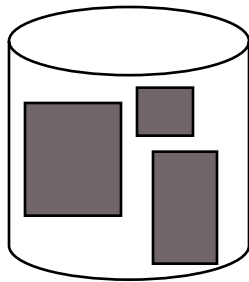
Stripe Affinity

- Dump phases implement a recoverable dump
- At load, every page is restored with its most recent image
- **QUESTION:** How to achieve this and use striped archive devices ?
- **ANSWER:** Stripe affinity - every image of a page in database/log appears on exactly one stripe throughout the dump

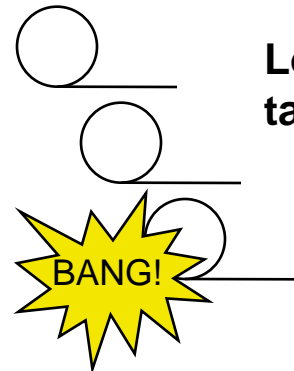
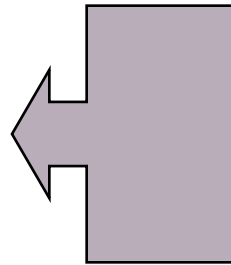
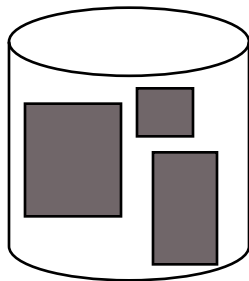


Page p^{N+1} is newer image of page p^N

Stripe Affinity and Volume Sets



dump database to 3-way stripe



Load database but one of our tape drives has blown up !

Can still load a 3-way striped dump on just 2 tape drives

The load is ...

- **Stripe order independent**
- **Volume set dependent**

Because all images of a page are on exactly one stripe

Because modified images of a page follow each other physically and chronologically

Stripe Affinity / Volume Set Example

Stripe 1: taps1a, taps1b
Stripe 2: taps 2a, taps2b
Stripe 3: taps 3a, taps3b

} *3 stripes, 2 tapes per stripe*

Drive 1: taps1a, taps1b
Drive 2: taps2a, taps2b, taps3a, taps3b

} *Allowed*

Drive 1: taps3a, taps3b, taps2a, taps2b
Drive 2: taps1a, taps1b

} *Allowed*

Drive 1: taps1b, taps1a *Wrong order !*
Drive 2: taps2a, taps2b, taps3a, taps3b

} *NOT allowed*

Also, you cannot dump to fewer stripes than you attempt to load from

Eg. Dump to single stripe using 6 tapes then try and load from 6 drives!

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days }  
  ]
```

} *dump ONLY*

- Applicable whether dump/load database OR transaction
- Applicable to both dump and load commands ... EXCEPT where indicated

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days } } dump  
  ] ONLY
```

- Name of the dump file
- If not specified, Backup Server uses a default
- Need to know correct dump file name when loading from a multi-file volume
- *load ... with listonly=full* is useful to find out dump file names !

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days } } dump  
  ] ONLY
```

- Determines whether tapes rewind after the dump/load completes
- But for dump, tape is always rewound first ... then positioned
- For multi-file dumps, tape is rewound between each dump, then positioned at end
- Necessary for capacity calculations

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days } } dump  
  ] ONLY
```

- Specifies capacity of the tape media
- If dump fills the tape, a new tape volume will need to be written
- Necessary on UNIX to underestimate to ensure don't hit EOT
- Don't forget ... for devices with data compression, the compression depends upon the data !

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days } } dump  
  ] ONLY
```

- Determines if dump files should be appended or should overwrite
- Need to use *with init* if archive media holds non-Sybase data
- (Prevents overwriting other sources of valuable data)
- Also need *with init* to overwrite a first dump file which has not yet expired ...

More Options on dump/load

Example :

```
dump database mydb to ...  
  [ with  
    { file = file_name,  
      [ nounload | unload ],  
      capacity = number_kilobytes,  
      [ noinit | init ],  
      retaindays = number_days } } dump  
  ] ONLY
```

- Specifies number of days that Backup Server prevents you overwriting a dump
- Applicable only to first dump file on a volume / volume set
- Default is *sp_configure "tape retention in days"* parameter
- Otherwise overwrite using *dump ... with init*

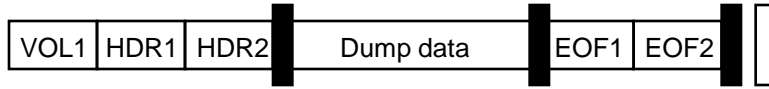
Dump Format On Archive Media



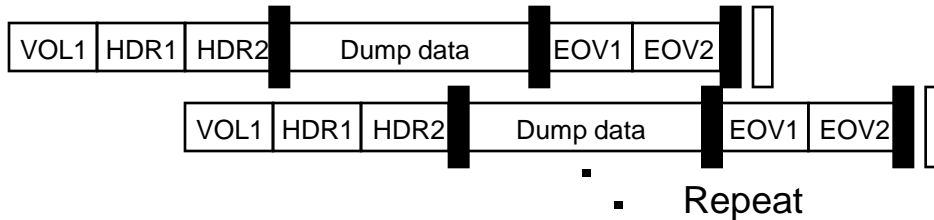
What does the dump look like on tape or disk ?

Media Layout - Tape

Single file,
single volume

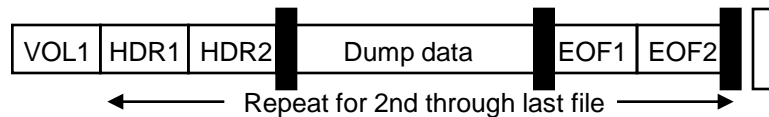


Single file,
multi-volume

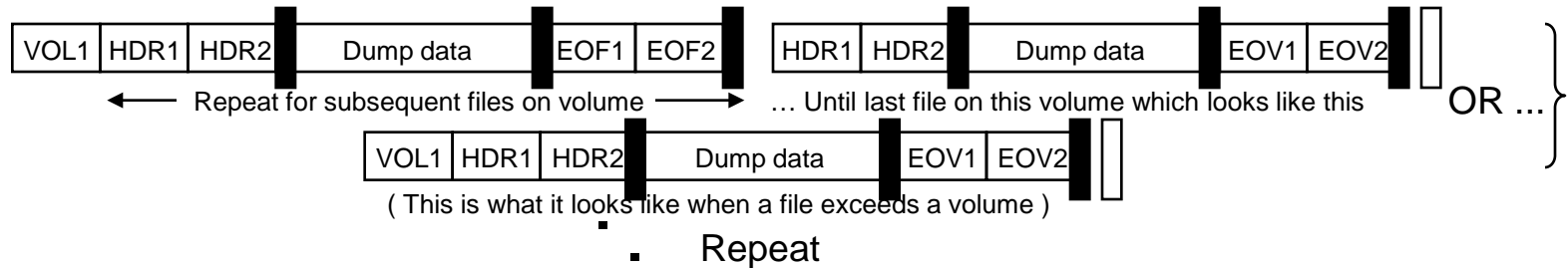


Q. One or two tapemarks after EOF2/EOV2 ?
A. System 10 = 2
System 11 = 1 or 2

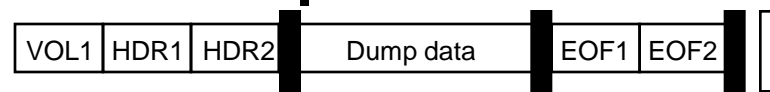
Multi- file,
single volume



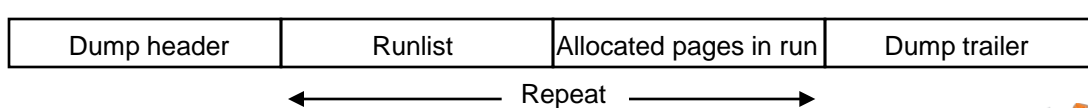
Multi- file,
multi- volume



Dump data



“Dump data”
expands to ...

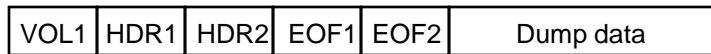


About Tapemarks

- **System 10 always wrote 2 tapemarks after the EOF2/EOV2**
- **Some tape devices don't allow overwriting a tapemark**
- **Thus not possible to append files for multi-file dump**
- **System 11 has Extended Dump Format**
- **For devices that support overwriting a tapemark, it still writes 2 tapemarks**
- **For those that do not support overwriting, it writes just a single tapemark, thus we can now append files**

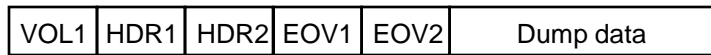
Media Layout - Disk

Single file,
single volume



Used for removable (eg. floppies) and non-removable disk media(eg. filesystem files or raw partitions)

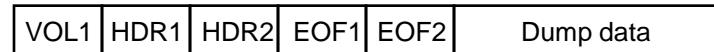
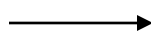
Single file,
multi-volume



Used for removable disk media only (eg. floppies)

▪ Repeat

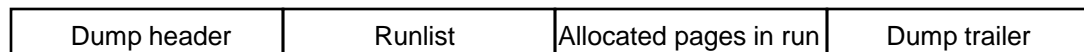
Editor's note: need to check last one ends with EOF not EOVS



Important differences compared to tape media ...

- **No tape marks !**
- **EOF/EOVS labels come before the dump data**

“Dump data”
expands to ...



← Repeat →

ANSI Label Format - VOL1, HDR1 etc

- The ANSI labels on a dump can be examined
- Use **load ... with listonly = full** command
- This loads no data but allows examination of volume set
- **VERY USEFUL !**

VOL1

<u>Field</u>	<u>#chars</u>	<u>listonly=full</u>	<u>Description</u>
vlblid	4	Label name	VOL1
vid	6	Volume id	volume id (name)
vstdvers	1	Labelling version	Version of ANSI standard = 3

ANSI Label Format - VOL1, HDR1 etc

HDR1, EOF1, EOVI

<u>Field</u>	<u>#chars</u>	<u>listonly=full</u>	<u>Description</u>
I1lblid	4	Label id	HDR1 or EOF1 or EOVI
I1fid	17	File name	File id (name of this dump)
I1fsid	4	Stripe count	Count of stripes in set
I1fscnt	2	Device typecount	Count stripe on this device type
I1fsect	4	Archive volume number	Volume number within volume set
I1fseq	4	Stripe position	Which stripe this belongs to
I1creyear, creday	2, 3	Create date & time	Year 00-99, day in year 001-366
I1cretime	5		Create wallclock time 5 hex digits
I1expyear, expday	2, 3	Expiration date & time	As above but no time just 00:00:00
I1blkcnt	6	File block count	0 for HDR1, else # 2KB blocks
I1syscode	8	Sybase id string	'Sybase ' identifies our dump tape

ANSI Label Format - VOL1, HDR1 etc

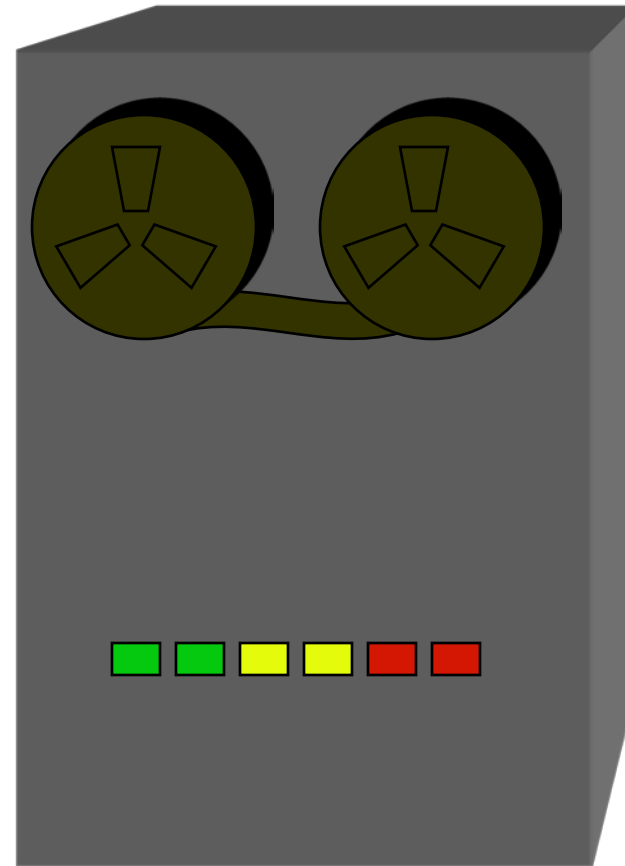
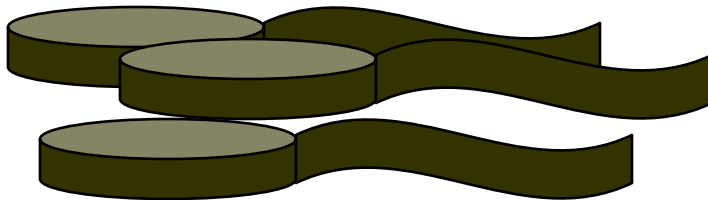
HDR2, EOF2, EOVS

<u>Field</u>	<u>#chars</u>	<u>listonly=full</u>	<u>Description</u>
I2lblid	4	Label id	HDR2 or EOF2 or EOVS
I2recfmt	1	Record format	Record format = F for fixed length
I2blklen	5	Max. bytes/block	Max number of chars per block
I2reclen	5	Record length	Record length = 2048
I2archver	2	Backup format version	Used for backward compatibility
I2dbname	30	Database name	Database name in dump

Devices And Volumes



Device capabilities
Volume positioning



The Problem ...

- **Many Operating Systems**
- **Subtle differences in OS system calls and behaviour**
- **Different kinds of archive media (eg. QIC, DAT, DLT)**
- **Very many makes and models of hardware devices !**
- **Hardware device make X slight differences to make Y ?**
- **System 10 Backup Server was extremely device (make, model, type) dependent**
- **Many tape drives not supported or only as single-file**

The Solution ...

System 11 Backup Server improved the situation

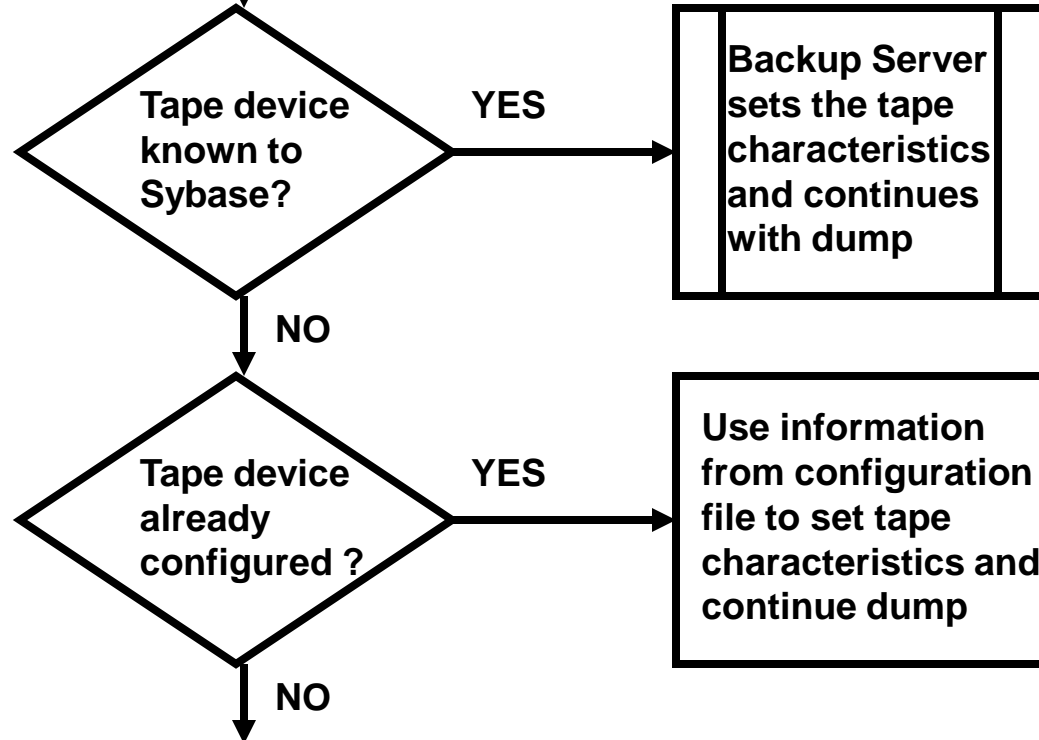
- **More flexible, less device-dependent**
- **Allow multiple dump files to be written to as many tape devices as possible**

This is achieved with

- **“Extended Dump Format” (tapemarks covered earlier)**
- **“Automatic Tape Configuration”**

Automatic Tape Configuration

dump database/transaction



Only if dump with init, run Auto Tape Configuration. Various tests write data to tape to see what device can do. Details stored in configuration file. Then continues dump using that information.

Tape Configuration File

- Default is `$SYBASE/backup_tape.cfg`
- Can be specified on backupserver command line -c option
- Example:

```
Revision 1  
mymachine /dev/rmt/0hn 2 1 65536 44 0 1  
mymachine /dev/rmt/0n 2 1 65536 44 0 1
```

- | | |
|---------------------------|---|
| • hostname | Machine name |
| • tape_device_name | Name of the device |
| • filemarks | Number of tapemarks written before file is closed |
| • append_strategy | Integer representing strategy to use to append new dump |
| • blocksize | Maximum blocksize in bytes |
| • ostype | Code used by OS to determine device type |
| • fm_type | Type of tapemarks device is configured with |
| • cls_rdetpmk | Indicates position of tape after read of tapemark |

Append Strategies

0) Device supports only single-file

- Only one dump can be written per tape

1) Back Skip File

- Tape device supports overwriting of tapemarks
- Most efficient way of appending
- Tape positioned to append between last 2 tapemarks

2) Seek To End

- “Seek to end of written data” system call
- Volume read to see if last in volume set
- Device rewind and then “seek to end of written data” to append

3) Skip N Tapemarks

- “Forward skip file” system call
- Forward skips exact number of tapemarks currently on tape

4) Skip N+1 Tapemarks

- As above but skips one more than number of tapemarks
- Number of tapemarks discovered finding end of volume set

Volume Handling

Positioning on the volume(s) is as follows :

- For dump

- If “with init” specified position at beginning of first volume
- Otherwise position at the end of last volume
- BUT, if single-file volume set, we cannot append
- So check expiration date in HDR1 of first volume
- If “expired”, position at beginning of first volume
- Otherwise volume set has not expired so prompt to see if should overwrite volume set or mount another

Only 2 possible positions

- 1) *Beginning of first volume*
- 2) *End of last volume*

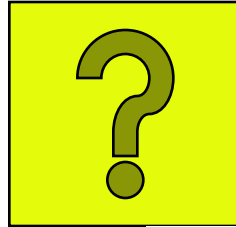
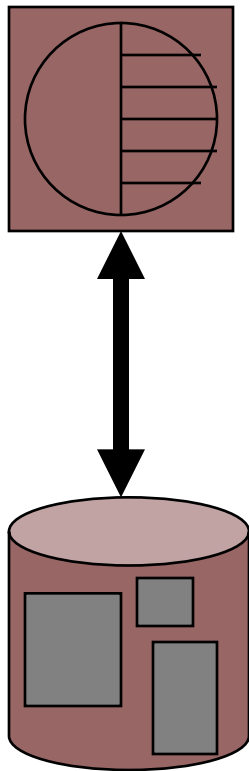
- For load

- If a specific dump file was not specified, position at beginning of first volume
- If a specific dump file was specified, and searching locates the dump file in the volume set, position at start of requested file
- May abort the load if further label/stripe validation fails
- If specified dump file not found either prompt for volume change or abort load (eg. if non-removable media)

Only 2 possible positions

- 1) *Beginning of first volume*
- 2) *Start of requested dump file*

The ASE Side Of The Story



Look into ...

- **dump database**
- **load database**
- **dump transaction**
- **load transaction**

Segment remapping

ASE - dump database algorithm

Initialisation

Various structures, database name from command, check OPER role, permissions etc

Synchronisation

Address lock to prevent other dump db's/tran's at same time on this database. Freeze db mapping for duration of dump (ie. prevent alter db etc)

Open Connections

Primary and secondary connection (latter if we would hang on primary, eg. when BS is scanning database)

Determine Phase Sequence

Is log mixed with data ? YES, then dump entire db mapping, including inactive portion of the log. NO, then dump data and just active portion of log.

Initial RPCs to Backup Server

Begin the dump, identify disk pieces and map of pages to dump, names of archive devices to use etc

CHECKPOINT

Several reasons why. Determine start of log for SCANLOGPAGES phase. Should reduce load db recovery time since may get more recent “oldest active xact” in the log. If log on separate segment then use this to our benefit. The log in the dump is effectively truncated saving dump time and space on tape ! Log in db is still intact !

Send dump header to BS

Includes checkpoint marker from above.

ASE - dump database algorithm

Dump Flush List (DFL) Synchronisation

Pages that get flushed to disk during DBPAGES phase but are not logged need to be remembered and included in the dump. Eg. fast bcp, select into, text pages, page slits, index create. Effectively we set a bit that is tested in the parts of SQL Server that do these flushes, telling them to keep track of such pages in a special list.

DBPAGES Phase

Dump every page in the mapping except dedicated log segment (if exists). Tell BS to do the scan since much faster than overhead of sending runlists. BUT, it may take BS a long time to do the scan tying up the primary connection with that RPC. The above flush activity might be so much that we fill our list. So we send runlists of pages for next phase (FLUSHEDPAGES) earlier than normal to free list space. Hence the second connection.

FLUSHEDPAGES Phase

As above, we may have sent runlists of flushed pages to BS already. Any operations wishing to flush pages are blocked at this time until this phase is over. We send runlists for any/all pages still mentioned in our flush list.

Dump Instant Marker

Now all non-log database pages are dumped, we determine ending boundary of log coverage for period spent dumping them.

End DFL Synchronisation

Terminus of log is known. Operations wanting to flush pages that were blocked above are free to continue (and they do so without needing to record pages on the special flush list)

ASE - dump database algorithm

SCANLOGPAGES Phase

All log pages in the scope of the dump have been created by their xacts. Now is time to flush the log. Any stale log records that we copy will be for xacts that will not commit until after the dump instant. If log is mixed with data, then much of log has already been copied in DBPAGES phase but we must still continue to cover for time spent dumping in the DBPAGES phase. Could duplicate some/much/most/all of what was dumped previously. If log is on its own segment, we send runlists of log pages for the active portion of the log only. This effectively truncates the log in the dump. Log is still intact in the db itself ! NOTE: It would be nice to get BS to scan the log segment similar to DBPAGES phase. But it would pick up allocation pages that could contain post-dump instant allocation information. So we have to use runlists.

Phases complete, cleanup

Tell BS that dump is complete. Release resources and close connections. Also, now we have completed our dump database, we can clear those conditions that prevented a dump transaction. They are

- A new database or one just upgraded cannot have dump tran until after dump db
- Any minimally logged activity (eg. select into) that occurred since the last dump database
- If a dump tran .. with truncate only has occurred since last dump database

ASE - load database algorithm

Initialisation

Various structures, exception handlers etc. Determine if regular load or load with headeronly/listonly. Load of master database requires single-user mode.

Tell Backup Server to begin load

Sends RPCs. But if not a regular load of all data pages then ...

Special cases

... If headeronly/listonly, we are done - info has been "printed". Make sure in single-user mode if load of master.

Prepare to read in pages

Will the dumped database fit in this database? Mark sysdatabases row as in load. Take db offline. If dump contains only active portion of log (ie. truncated in dump), then clear all allocation pages for log before we load. Recovery will deal with any reallocations. Invalidate any in-memory references to objects in the old database. Throw out any pages in cache for the old database.

Read in pages of the dump

Backup Server uses the runlist information to load allocated pages or clear unallocated ones.

Read dump trailer

Locates the Dump Instant Marker

End load of pages

Inform Backup Server via RPCs then release resources used for RPCs.

ASE - load database algorithm

Merge database diskmaps as necessary

If diskmap of dumped database differs from diskmap of database we are loading into, attempt to merge the maps to ensure log/data separation

Mirroring tasks

If using Sybase mirroring, Backup Server only wrote to one side so copy to the other side now.

New database bigger than dumped database ?

Initialise any pages leftover at the end

Patch the log

If load contains only active portion of log, zero the previous page pointer on the log's first page. Must patch the log's last page to show Dump Instant Marker is last record in the log.

Prepare for recovery

Especially set no checkpoint on recovery (which is about to happen). This would put any subsequent dumps out of sequence.

Run recovery

Exactly the same process as when SQL Server is booted. Commit or rollback transactions up to Dump Instant.

Retry merge of database diskmaps

True of master database since could not update sysusages earlier (had to wait for recovery to run)

ASE - load database algorithm

Post-recovery actions

If master db, bring it online. If this SQL Server is running different charset/sortorder to that from which database was dumped, check and flag all indexes built on char/varchar columns. If this is NOT master db, then mark all stored procedures for reresolution.

Final tasks

DBO of current database set to current user. Flush all buffers from cache. The load only used default data cache. When db is online'd, other caches may come into effect. Prevents reading same buffers into other caches. Mark sysdatabases row as not in load.

Cleanup

Release resources, close the database just loaded. If this is reload of master database, force a reboot of SQL Server.

Segment Mapping Issues (sysusages)

- It is preferable to load into a target database (**target db**) that has same fragments and same segment mapping as the original database that was dumped (**source db**)
- Preferable *but not essential*
- SQL Server will attempt to merge diskmaps to accommodate
- Diskmap = row from sysusages for this database
- SQL Server will remap sysusages for the target database using the following rules :

1) Absolutely CANNOT change logical to virtual page mapping

What is in the dump are logical page numbers - a “picture” of the source db - they cannot be re-ordered. During load they must fit within the storage defined by the target db. Segment boundaries are secondary !

2) Target db provides physical storage and logical/virtual page mapping

Physical storage = sysusages.vstart and size, mapping is between sysusages.lstart and sysusages.vstart

3) Source db provides logical storage boundaries

Quite simply, sysusages.segmap

4) SQL Server will create as many sysusages rows as necessary

May need to split physical fragments into extra segments thus needing extra rows. May also combine fragments.

Remapping Sysusages - Example

source db has following sysusages rows ...

segmap	lstart	size	vstart	vdevno	comment
3	0	2048	16777216	1	(3 = system & default = data)
3	2048	2048	33554432	2	
4	4096	2048	50331648	3	(4 = log segment)
3	6144	2048	16779264	4	

target db has these sysusages rows ...

segmap	lstart	size	vstart	vdevno	comment
7	0	4096	16777216	1	(7 = system & default & log
7	4096	4096	33554432	2	= data & log)

FINAL PRODUCT has following sysusages rows ...

segmap	lstart	size	vstart	vdevno	comment
3	0	4096	16777216	1	
4	4096	2048	33554432	2	} data & log on same device
3	6144	2048	33556480	2	

Remapping Sysusages - Example

source db has following sysusages rows ...

map	Istart	size	vstart
3	0	2048	16777216
3	2048	2048	33554432
4	4096	2048	50331648
3	6144	2048	67108864

... (m & default = data)

... (nt)

Two adjacent fragments with same segment map on source db can be joined into single fragment with contiguous storage on target db
 $2048 + 2048 = 4096$

target db has these sysusages rows

segmap	Istart	size	vstart
7	0	4096	16777216
7	4096	4096	33554432

... (m & default & log = data & log)

FINAL PRODUCT has following sysusages rows ...

map	Istart	size	vstart
3	0	4096	16777216
4	4096	2048	33554432
3	6144	2048	33556480

vdevno 1
 vdevno 2 } data & log on same device
 vdevno 2

Remapping Sysusages - Example

source db has following sysusages rows ...

segmap	Istart	size	vstart
3	0	2048	16777216
3	2048	2048	33554432
4	4096	2048	50331648
3	6144	2048	16779264

vdevno 1 (3 = system & default = data)
 vdevno 2
 vdevno 3 (4 = log segment)
 vdevno 4

target db has these sysusages rows ...

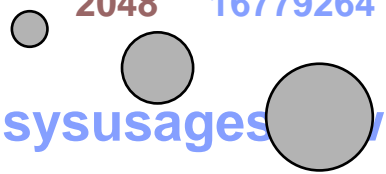
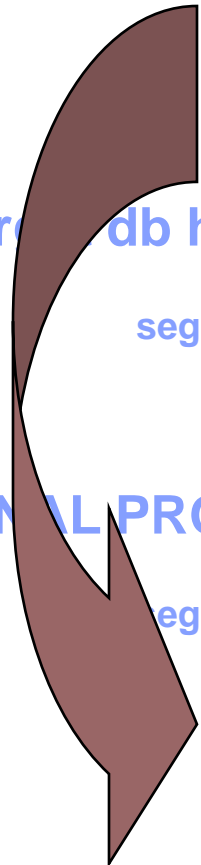
segmap	Istart	size	vstart
7	0	4096	16777216
7	4096	4096	33554432

Log segment must start at logical page 4096 and last data segment must start at logical page 6144 - has to be that way ! The only place left in target db for these is on vdevno 2.
 $2048 + 2048 = 4096$
 Segmaps can be adjusted but physically on same vdevno.

FINAL PRODUCT has following sysusage rows ...

segmap	Istart	size	vstart
3	0	4096	16777216
4	4096	2048	33554432
3	6144	2048	33556480

vdevno 2
 vdevno 2 } data & log on same device



Remapping Sysusages - Example



Note how target db started off with 2 sysusages rows each with segmap 7 but ended up with 3 rows none of which had segmap 7.

Because the source db provides the segment mapping

target db has these sysusages rows ...

segmap	Istart	size	vstart	vdevno 1	vdevno 2
7	0	4096	16777216	(7 = system & default & log	
7	4096	4096	33554432		= data & log)

FINAL PRODUCT has following sysusages rows ...

segmap	Istart	size	vstart	vdevno 1	vdevno 2
3	0	4096	16777216		
4	4096	2048	33554432		
3	6144	2048	33556480		

} data & log on same device

Remapping Sysusages - Example

source db has following sysusages rows ...



segmap	Istart	size	vstart	vdevno 1	(2 = system & default = data)
3	0	2048	16777216		

The solution to this problem ...

create database target_db on

```

dev1 = 8,          /* 8MB for data on one device */
dev2 = 4,          /* 4MB for log on a different device */
dev1 = 4           /* another 4MB on 1st device again */
for load
    
```

target db has

segmap

& log
& log)

FINAL PRODUCT has following sysusages rows ...

segmap	Istart	size	vstart	vdevno 1	vdevno 2	vdevno 2	
3	0	4096	16777216				
4	4096	2048	33554432				} data & log on same device
3	6144	2048	33556480				

Remapping Sysusages - Example

source db has following sysusages rows ...

segmap	lstart	size	vstart	vdevno	
3	0	2048	16777216	1	(3 = system & default = data)
3	2048	2048	33554432	2	
4	4096	2048	50331648	3	(4 = log segment)
3	6144	2048	16779264	4	

target db now has these three sysusages rows ...

segmap	lstart	size	vstart	vdevno	
7	0	4096	16777216	1	(7 = system & default & log
7	4096	2048	33554432	2	= data & log)
7	6144	2048	16781312	1	

FINAL PRODUCT has following sysusages rows ...

segmap	lstart	size	vstart	vdevno	
3	0	4096	16777216	1	} 
4	4096	2048	33554432	2	
3	6144	2048	16781312	1	

Troubleshooting



Troubleshooting - Connections

- Is the local/remote Backup Server running ?
 - showserver, ps etc
- Using the correct port number ?
 - Check interfaces, compare with netstat -a, telnet hostname port#
- Do basic RPCs work ?
 - eg. exec SYB_BACKUP...sp_who
- If basic RPCs do not work ...
 - Did the RPC return error, nothing, just the prompt ?
 - Could be site handler issue - sp_dropserver/sp_addserver, set timeouts false ?
 - Try RPCs to Backup Server from another SQL Server ie. is it SQL Server's fault ?
 - sp_helpserver to see logical & physical network names in syssservers
 - It is important to be consistent about network names! Don't try aliases !
 - SQL Server remote connections set-up ? sp_configure
 - Enough connections ?
 - Backupserver -C option specifies number of server connections
 - ie. incoming connections (from SQL Server or a master Backup Server)
 - Backupserver -N specifies total number of network connections (DBPROCESSes)
 - ie. outgoing connections to a slave Backup Server (remote dump/load)

Placeholder for Traceflags

- Placeholder for BackupServer and OpenServer traceflags

Troubleshooting - Device Access

- **Problems accessing database devices ...**
 - Are you sure you are going to right Backup Server on right machine ?
 - `sp_helpserver` for physical network name
 - Permissions on devices (who is Backup Server running as ?)
 - HP-UX specific issues - has to use async I/O - so any OS parameters, patches, shared memory and other resource issues ?
- **About HP-UX ...**
 - Ordinarily, Backup Server uses synchronous read/write system calls.
 - HP-UX is an exception because could use the block or char-special devices.
 - This might result in Backup Server doing buffered I/O, and SQL Server doing raw I/O, with obvious data synchronisation issues.
 - Async I/O calls on block-special devices force raw I/O for data integrity.
- **Problems accessing archive devices ...**
 - Are they still in use ? eg. Any `sybmultbuf` zombie processes ? Use `ps` and `kill`.
 - Remember that archive emulator forked first, then database emulator. ie. PPIDs !
 - Auto-tape configuration
 - `backupserver -D 2` forces tape configuration (even though tape might be in `backup_tape.cfg` file already)
 - `backupserver -D 4` traces the tape configuration process
 - Edit the `backup_tape.cfg` file by hand ? Make sure you test load as well as dump !

Troubleshooting - Sybmultbuf

- **Sybmultbuf errors go to UNIX stderr - redirect stderr to a file on backupserver command line**
- **Resource issues ? eg. Shared memory, kernel parameters, physical memory, swap space etc.**
- **Remember that on local Backup Server there is a pair of sybmultbufs per stripe device - so the wider the stripe, the more resources are consumed !**

Troubleshooting - Volume Handling

- Transferring dumps between tape media and disk media and vice versa
 - From earlier slides you will see that format is different
 - ie. On disk media, EOF/EOV labels come before the dump data
 - Sometimes it is possible to move dumps between media but it is
 - a) Tricky - chop up the dump into ANSI header labels, dump body, ANSI trailer labels then glue together in order
 - b) Not guaranteed for all devices, all platforms
 - c) NOT SUPPORTED !
- Specifying capacity
 - UNIX does not detect EOT very well (or UNIX's aren't consistent)
 - Therefore try to avoid hitting physical EOT on UNIX (VMS handles it OK)
 - Specify a capacity on dump/load command
 - 70% rule of thumb - specify a capacity which is 70% of what tape media can record
 - Compressing tape drives - difficult to specify capacity because compression depends on data.
 - Always underestimate to be on safe-side,

End Of Presentation ...

Or was that End Of Tape ?

