Oxford University TSM 2001 Symposium
St. Catherine's College
Oxford University
England
20th and 21st September 2001

The TSM Client

A Peek Under the Hood

Tivoli

Andrew M. Raibeck
Tivoli Storage Manager Client Development
araibeck@us.ibm.com

IBM Corporation
9000 S. Rita Road
Bldg. 9062, Rm. 2465
Tucson, AZ 85744
USA

The TSM Client: A Peek Under the Hood

- This presentation will provide more detailed information than is generally available in the formal documentation about how certain parts of the TSM client work.
- Notices:
  - This document is based on the Tivoli Storage Manager Backup-Archive Client version 4.2.1.
  - The details described in this document are subject to change without notice at the discretion of IBM or Tivoli.
  - This document does not represent a formal part of the TSM documentation.
  - IBM and Tivoli make no commitment to maintain this document on an on-going basis.

Tivoli

Agenda

- Client and server communication
- Multithreading
- Transaction processing
- Journal-based backup

Client and server communication

TSM client-server verbs

- *Verbs* define the protocol (language) used by the client and server to communicate with each other.
- Some things verbs are used for:
  - Initiate transactions
  - Perform queries
  - Send data to the server
  - Update file space information

TSM client-server verbs

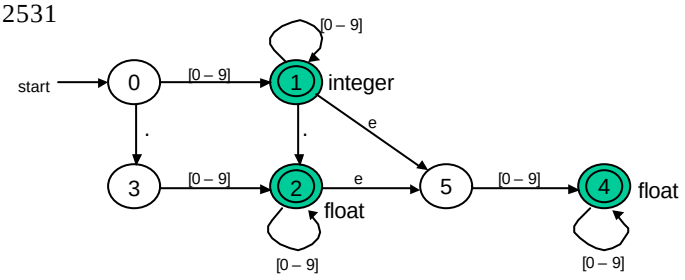
- Each verb represents a *state*.
- *Transitions* from one state to the next occur as the client and server exchange verbs.
- Being a well-defined protocol, when one entity sends a particular verb to the other entity, the first entity has an expectation as to how the second entity might respond.
- Failure to meet this expectation results in a *protocol error.*
- A *state machine* is used to describe the protocol's valid states and transitions.

State machines – a little theory

- What is a state machine?
  - A finite set of *states*
  - A set of *transitions* from one state to the next
  - A special *start* state
  - A set of *accepting* states

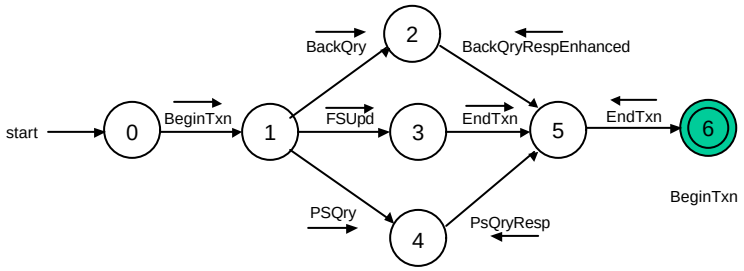
State machines – an example

- A state machine that recognizes positive integer and floating-point constants
- Examples:
  870.5
  .01e6
  2531



Example TSM protocol state machine *

- State machine to:
  - Query policy set information
  - Query backup set information
  - Update file space information



* Not representative of all the TSM verbs

Observing verb exchanges with a trace

SESSVERB.VERBINFO

```
09/14/2001 01:59:37.0220 : session.cpp        (2284): Sent Verb: Length:        4 Code: 00000012 Type: BeginTxn          ->
09/14/2001 01:59:37.0220 : session.cpp        (2263): Send Verb: Length:    14 Code: 000000A0 Type:  PSQry
09/14/2001 01:59:37.0220 : commtcp.cpp        (1861): TcpWrite: 14 bytes written of 14 requested.
09/14/2001 01:59:37.0220 : session.cpp        (2284): Sent Verb: Length:       14 Code: 000000A0 Type: PSQry            ->
09/14/2001 01:59:37.0220 : session.cpp        (2056): Recv Verb: ...
09/14/2001 01:59:37.0220 : commtcp.cpp        (1525): TcpRead: Upper level requested 4 bytes
09/14/2001 01:59:37.0230 : commtcp.cpp        (2060): TcpFlush: 18 bytes written on socket 588.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1642): TcpRead: Issuing recv for 4 bytes.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1743): TcpRead: 4 bytes read.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1564): TcpRead: 4 bytes read of 4 requested.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1525): TcpRead: Upper level requested 325 bytes
09/14/2001 01:59:37.0230 : commtcp.cpp        (1642): TcpRead: Issuing recv for 325 bytes.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1743): TcpRead: 325 bytes read.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1564): TcpRead: 325 bytes read of 325 requested.
09/14/2001 01:59:37.0230 : session.cpp        (2178): Recv Verb:
09/14/2001 01:59:37.0230 : session.cpp        (2179):   Length:   329 Code: 000000A1 Type:    <-          PSQryResp
09/14/2001 01:59:37.0230 : session.cpp        (2056): Recv Verb: ...
09/14/2001 01:59:37.0230 : commtcp.cpp        (1525): TcpRead: Upper level requested 4 bytes
09/14/2001 01:59:37.0230 : commtcp.cpp        (1642): TcpRead: Issuing recv for 4 bytes.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1743): TcpRead: 4 bytes read.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1564): TcpRead: 4 bytes read of 4 requested.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1525): TcpRead: Upper level requested 2 bytes
09/14/2001 01:59:37.0230 : commtcp.cpp        (1642): TcpRead: Issuing recv for 2 bytes.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1743): TcpRead: 2 bytes read.
09/14/2001 01:59:37.0230 : commtcp.cpp        (1564): TcpRead: 2 bytes read of 2 requested.
09/14/2001 01:59:37.0230 : session.cpp        (2178): Recv Verb:
09/14/2001 01:59:37.0230 : session.cpp        (2179):   Length:     6 Code: 00000013 Type:    <-             EndTxn
```

Observing verb exchanges with a trace

SESSVERB.VERBINFO

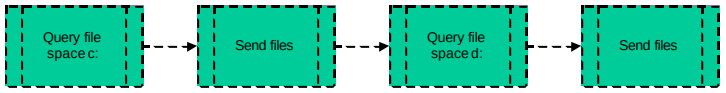
```
09/14/2001 03:46:53.0267 : session.cpp         (2263): Send Verb: Length:    20 Code: 000000C4 Type:  BeginTxnEnhanced
09/14/2001 03:46:53.0267 : session.cpp         (2288): ->
09/14/2001 03:46:53.0287 : cubackup.cpp        ( 874): cuBackIns: fsID: 9, hl: '\TSM\BACLIENT', ll: '\DSM.EXE'
09/14/2001 03:46:53.0287 : cubackup.cpp        ( 875):            objType: FILE, owner: ''
09/14/2001 03:46:53.0287 : cubackup.cpp        ( 876):            mountWait: false, dataSize: 0.2192384, mgmtClass: 1,
copyGroup: 1
09/14/2001 03:46:53.0287 : session.cpp         (2263): Send Verb: Length:   335 Code: 000000C6 Type:  BackInsEnhanced
09/14/2001 03:46:53.0287 : session.cpp         (2288): ->
09/14/2001 03:46:53.0387 : senddata.cpp        (3156): SendFileData: Sending a 32768 byte DataVerb.
09/14/2001 03:46:53.0397 : session.cpp         (2263): Send Verb: Length: 32768 Code: 00000007 Type: Data
09/14/2001 03:46:53.0407 : session.cpp         (2288): ->
09/14/2001 03:46:53.0417 : senddata.cpp        (3156): SendFileData: Sending a 32768 byte DataVerb.
09/14/2001 03:46:53.0417 : session.cpp         (2263): Send Verb: Length: 32768 Code: 00000007 Type: Data
09/14/2001 03:46:54.0598 : session.cpp         (2288): ->
09/14/2001 03:46:54.0598 : senddata.cpp        (3156): SendFileData: Sending a 32768 byte DataVerb.
09/14/2001 03:46:54.0598 : session.cpp         (2263): Send Verb: Length: 32768 Code: 00000007 Type: Data
09/14/2001 03:46:54.0598 : session.cpp         (2288): ->
.
.
.
09/14/2001 03:46:55.0170 : senddata.cpp        (3156): SendFileData: Sending a 29121 byte DataVerb.
09/14/2001 03:46:55.0170 : session.cpp         (2263): Send Verb: Length: 29121 Code: 00000007 Type: Data
09/14/2001 03:46:55.0180 : session.cpp         (2288): ->
09/14/2001 03:46:55.0180 : cutxn.cpp           ( 335): cuEndTxn: Sending vote: 1, reason: 0.
09/14/2001 03:46:55.0180 : session.cpp         (2263): Send Verb: Length:     6 Code: 00000013 Type:  EndTxn
09/14/2001 03:46:55.0180 : session.cpp         (2288): ->
09/14/2001 03:46:55.0180 : session.cpp         (2058): Recv Verb:
09/14/2001 03:46:55.0240 : session.cpp         (2179):   Length:     6 Code: 00000013 Type:    <-             EndTxn
09/14/2001 03:46:55.0240 : cutxn.cpp           ( 682): cuGetEndTxn: Received vote: 1, reason: 0
```


Multithreading

Pre-multithreaded model

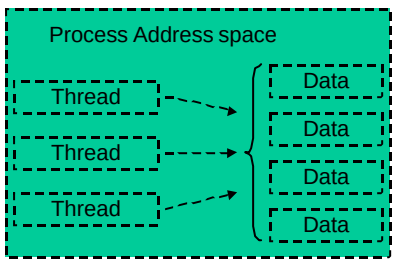
- Prior to TSM 3.7, file specifications were processed one at a time.

dsmc incremental c: d:



The advantages of multithreading

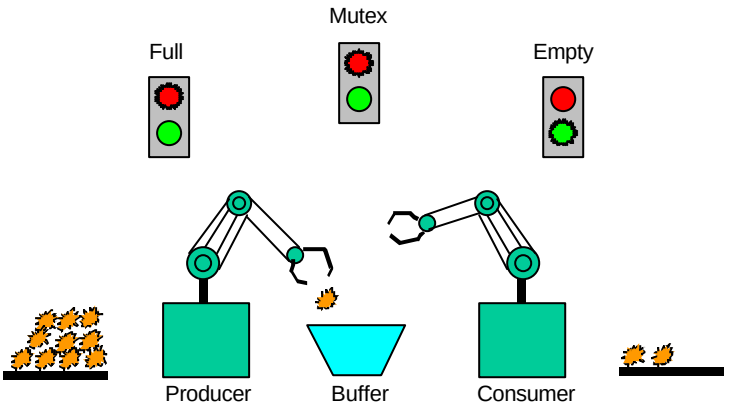
- Multiple threads of execution can run in parallel within a single process's address space.
- The threads within the process can share data and other resources



Multithreaded model: “Producer-Consumer”

- Also known as the “Reader-Writer” model
- Involves two basic types of threads:
  - Producer thread: writes data to a buffer
  - Consumer thread: reads data from a buffer
- Synchronization issues:
  - The producer can not write data to the buffer if the buffer is full.
  - The consumer can not read data from an empty buffer.
  - The consumer needs to know when no more data will be forthcoming.

The Producer-Consumer model



Types of TSM threads

- Main
- Signal Waiting
- Producer
- Consumer
- Performance Monitor

The Main thread

- Handles common housekeeping tasks:
  - Performs general system initialization
  - Parses commands
  - Processes options
  - Performs authentication with the TSM server
  - Policy set retrieval
  - Creates the producer thread
  - Creates the performance monitor thread
  - Queues up file specifications to be processed by the producer thread

The Signal Waiting thread

- Captures signals for the command line client:
  - Thread-level “Trap-style” signals:
    - Invalid memory references
    - Floating point errors
    - Illegal instructions
    - etc.
  - Process-level signals:
    - CTRL-C
    - CTRL-BREAK
- Not applicable for Windows, which uses the Windows *Console Event Handler* instead.

The Producer thread

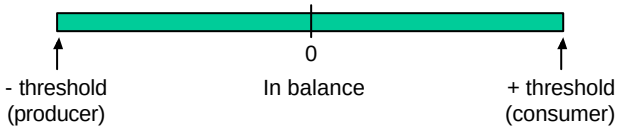
- The “front end” for TSM processing
- Starts a consumer thread
- Retrieves file specifications queued up the the main thread
- Queries the TSM server and examines the file system to determine which files to back up
- Queues up txns to be processed by the consumer thread(s)

The Consumer thread

- The “back end” for TSM processing
- Checks txn queue to see if there is work to do
- Handles File I/O
- Compresses the data (if applicable)
- Encrypts the data (if applicable)
- Sends and commits the data to the TSM server

The Performance Monitor thread

- Attempts to optimize performance by balancing thread usage (within the constraints of the RESOURCEUTILIZATION setting).



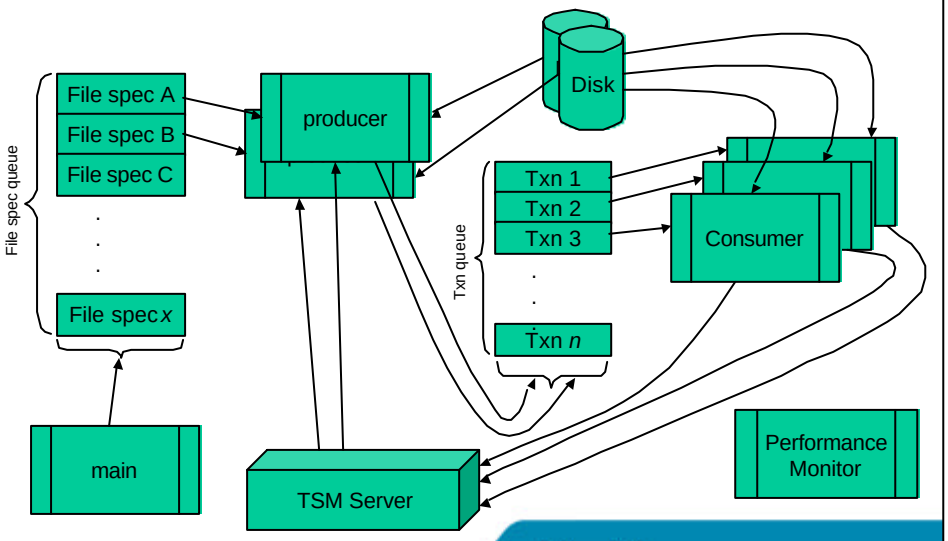
The Performance Monitor thread

- Wakes up periodically (about once every second).
- If the “check” interval has not elapsed, then goes back to sleep.
  - Interval is currently 0.2threshold.
- Attempts to start a new consumer thread if:
  - A new txn needs to be queued, but the txn queue is full.
  - The next txn waiting to be processed is the same as the last time we checked.
- Attempts to start a new producer thread if the next file spec waiting to be processed is the same since the last time we checked.

The Performance Monitor thread

- Quiesces a consumer thread if:
  - more than one consumer thread is running AND
  - the time since anything new has been placed on the txn queue exceeds the threshold.
- Quiesces a producer thread if:
  - more than one producer thread is running AND
  - the file spec queue is empty AND
  - the time since anything new has been placed on the txn queue exceeds the threshold.
- Consumer and producer threads are quiesced when there is no more work to be done.

Multithreaded backup



Multithreaded backup

- The main thread queues up file specifications for processing by the producer thread.
- The producer thread gets a file spec from the queue and determines what files in the spec to back up.
- The producer builds transactions and queues them up for processing by the consumer thread.
- The consumer thread gets a txn from the queue and sends the data to the TSM server.
- The performance monitor thread helps determine whether a new producer or consumer thread may be started.

RESOURCEUTILIZATION values

RESOURCEUTILIZATION *n*	Maximum sessions	Producer sessions	Threshold (seconds)
<default>	2	1	30
1	1	0	45
2	2	1	45
3	3	1	45
4	3	1	30
5	4	2	30
6	4	2	20
7	5	2	20
8	6	3	20
9	7	3	20
10	8	4	10
(undocumented) $11 \le n \le 100$	n	$0.5n$	10

RESOURCEUTILIZATION values

- RESOURCEUTILIZATION > 2 is allowed only when connecting to a version 3.7 (or higher) server.
- Non-root UNIX user is limited to 1 session.
- These are undocumented, internal values that are subject to change without notice.
- RESOURCEUTILIZATION > 10 is unsupported.

RESOURCEUTILIZATION values

- Performance considerations:
  - Multiple front-end (producer) sessions will be started only when RESOURCEUTILIZATION is >= 5 and there are multiple file specifications.
  - On the back-end, multiple consumers reading from a single drive can degrade disk performance.
- Usage considerations:
  - QUERY NODE statistics will be misleading.
  - Review your TSM server's MAXSESSIONS setting.

Observing multithreading with a trace

THREAD,PFM,TID

```
09/14/2001 02:30:51.0080 [1260] : psthread.cpp        ( 788): Thread 1260 is yielding.

                         unknown thread         -1================>
09/14/2001 02:30:51.0080 [1656] : thrdmgr.cpp          (1326): Starting thread Signal waiting thread
09/14/2001 02:30:51.0080 [1656] : pktsd.cpp            ( 248): TSD set specific key, data: 0, a9cef0

                            main thread         0================>
09/14/2001 02:30:51.0080 [1260] : thrdmgr.cpp          ( 673): Thread 0 (main) has created new thread 1 (Signal waiting thread)
09/14/2001 02:30:51.0080 [1260] : pktsd.cpp            ( 248): TSD set specific key, data: 1, a8d5e4
09/14/2001 02:30:51.0190 [1260] : psthread.cpp         ( 788): Thread 1260 is yielding.

                         unknown thread         -1================>
09/14/2001 02:30:51.0200 [500] : thrdmgr.cpp          (1326): Starting thread B/A Txn Producer
09/14/2001 02:30:51.0200 [500] : pktsd.cpp            ( 248): TSD set specific key, data: 0, aaf264

                            main thread         0================>
09/14/2001 02:30:51.0200 [1260] : thrdmgr.cpp          ( 673): Thread 0 (main) has created new thread 2 (B/A Txn Producer)
09/14/2001 02:30:51.0200 [1260] : psthread.cpp         ( 788): Thread 1260 is yielding.

                B/A Txn Producer thread         2================>
09/14/2001 02:30:51.0200 [500] : psthread.cpp         ( 788): Thread 500 is yielding.

                         unknown thread         -1================>
09/14/2001 02:30:51.0200 [1216] : thrdmgr.cpp          (1326): Starting thread B/A Performance
09/14/2001 02:30:51.0200 [1216] : pktsd.cpp            ( 248): TSD set specific key, data: 0, a90294
09/14/2001 02:30:51.0200 [1216] : psthread.cpp         ( 850): Thread 000004C0 delaying for 1 seconds.

                            main thread         0================>
09/14/2001 02:30:51.0200 [1260] : thrdmgr.cpp          ( 673): Thread 0 (main) has created new thread 3 (B/A Performance)
```

Observing multithreading with a trace (cont.)

```
                    unknown thread       -1===============>
09/14/2001 02:30:51.0200 [548] : thrdmgr.cpp         (1326): Starting thread B/A Txn Consumer
09/14/2001 02:30:51.0200 [548] : pktsd.cpp           ( 248): TSD set specific key, data: 0, a900a8

              B/A Txn Producer thread        2===============>
09/14/2001 02:30:51.0200 [500] : thrdmgr.cpp         ( 673): Thread 2 (B/A Txn Producer) has created new thread 4 (B/A Txn
Consumer)

                B/A Performance thread        3===============>
09/14/2001 02:30:52.0201 [1216] : psthread.cpp        ( 857): Thread 000004C0 awakened.
09/14/2001 02:30:52.0201 [1216] : psthread.cpp        ( 850): Thread 000004C0 delaying for 1 seconds.

                B/A Performance thread        3===============>
09/14/2001 02:30:53.0203 [1216] : psthread.cpp        ( 857): Thread 000004C0 awakened.
09/14/2001 02:30:53.0203 [1216] : bacpfm.cpp          ( 345): consumers: 1, producers: 1, on txnQ: 0, on baSpecQ 3.
09/14/2001 02:30:53.0203 [1216] : psthread.cpp        ( 850): Thread 000004C0 delaying for 1 seconds.

                B/A Performance thread        3===============>
09/14/2001 02:30:54.0204 [1216] : psthread.cpp        ( 857): Thread 000004C0 awakened.
09/14/2001 02:30:54.0204 [1216] : bacpfm.cpp          ( 345): consumers: 1, producers: 1, on txnQ: 0, on baSpecQ 3.
09/14/2001 02:30:54.0204 [1216] : bacpfm.cpp          ( 466): Txn queue wait time is: 0, consumer threshold: 10000, producer
threshold: -10000.
09/14/2001 02:30:54.0204 [1216] : bacpfm.cpp          ( 468): Current txn Q entry is 0, previous entry is ffffffd
09/14/2001 02:30:54.0204 [1216] : bacpfm.cpp          ( 517): Txn queue change is 0
09/14/2001 02:30:54.0204 [1216] : bacpfm.cpp          ( 533): Current ba Spec Q entry is a9d6d4, previous entry is ffffffd
09/14/2001 02:30:54.0204 [1216] : psthread.cpp        ( 850): Thread 000004C0 delaying for 1 seconds.
```


Observing multithreading with a trace (cont.)

```
                B/A Performance thread        3===============>
09/14/2001 02:30:56.0207 [1216] : psthread.cpp        ( 857): Thread 000004C0 awakened.
09/14/2001 02:30:56.0207 [1216] : bacpfm.cpp          ( 345): consumers: 1, producers: 1, on txnQ: 0, on baSpecQ 3.
09/14/2001 02:30:56.0207 [1216] : bacpfm.cpp          ( 466): Txn queue wait time is: 0, consumer threshold: 10000, producer
threshold: -10000.
09/14/2001 02:30:56.0207 [1216] : bacpfm.cpp          ( 468): Current txn Q entry is 0, previous entry is ffffffd
09/14/2001 02:30:56.0207 [1216] : bacpfm.cpp          ( 517): Txn queue change is 0
09/14/2001 02:30:56.0207 [1216] : bacpfm.cpp          ( 533): Current ba Spec Q entry is a9d6d4, previous entry is a9d6d4
09/14/2001 02:30:56.0207 [1216] : bacontrl.cpp        (1146): numProducers 1, maxProducerThreads 4, producerInTransit F,
bacInitFlag 1, sharedSessP (00A9CC4C), numServerSessions 1, sessionThreshold 8
09/14/2001 02:30:56.0207 [1216] : psthread.cpp        ( 788): Thread 1216 is yielding.

                    unknown thread        -1===============>
09/14/2001 02:30:56.0207 [300] : thrdmgr.cpp         (1326): Starting thread B/A Txn Producer
09/14/2001 02:30:56.0207 [300] : pktsd.cpp           ( 248): TSD set specific key, data: 0, a9d2fc

                B/A Performance thread        3===============>
09/14/2001 02:30:56.0207 [1216] : thrdmgr.cpp         ( 673): Thread 3 (B/A Performance) has created new thread 5 (B/A Txn
Producer)
09/14/2001 02:30:56.0207 [1216] : psthread.cpp        ( 850): Thread 000004C0 delaying for 1 seconds.
```

Observing multithreading with a trace (cont.)

```
                B/A Performance thread        3================>
09/14/2001 02:31:06.0221 [1216] : psthread.cpp        ( 857): Thread 000004C0 awakened.
09/14/2001 02:31:06.0221 [1216] : bacpfm.cpp          ( 345): consumers: 2, producers: 4, on txnQ: 8, on baSpecQ 0.
09/14/2001 02:31:06.0221 [1216] : bacpfm.cpp          ( 466): Txn queue wait time is: -7072, consumer threshold: 10000,
producer threshold: -10000.
09/14/2001 02:31:06.0221 [1216] : bacpfm.cpp          ( 468): Current txn Q entry is 185a80, previous entry is 185a80
09/14/2001 02:31:06.0221 [1216] : bacontrl.cpp        (1179): numServerSessions 6, sessionThreshold 8, consumerInTransit F,
consumerInError F, numConsumersWaiting 0
09/14/2001 02:31:06.0221 [1216] : psthread.cpp        ( 788): Thread 1216 is yielding.

                      unknown thread        -1================>
09/14/2001 02:31:06.0221 [1156] : thrdmgr.cpp         (1326): Starting thread B/A Txn Consumer
09/14/2001 02:31:06.0221 [1156] : pktsd.cpp           ( 248): TSD set specific key, data: 0, aac8f0

                B/A Performance thread        3================>
09/14/2001 02:31:06.0221 [1216] : thrdmgr.cpp         ( 673): Thread 3 (B/A Performance) has created new thread 9 (B/A Txn
Consumer)
09/14/2001 02:31:06.0221 [1216] : bacpfm.cpp          ( 517): Txn queue change is 7072
09/14/2001 02:31:06.0221 [1216] : bacpfm.cpp          ( 533): Current ba Spec Q entry is 0, previous entry is aaaf7c
09/14/2001 02:31:06.0221 [1216] : psthread.cpp        ( 850): Thread 000004C0 delaying for 1 seconds.
```


Observing multithreading with a trace (cont.)

```
                B/A Performance thread        3================>
09/06/2001 17:47:34.0161 [1516] : psthread.cpp        ( 857): Thread 000005EC awakened.

                B/A Txn Consumer thread       9================>
09/06/2001 17:47:34.0161 [1568] : ntsecurity.cpp      ( 628): getFileSecurityDescriptor(E:\Documents and
Settings\Andy\PrintHood): Entry.
09/06/2001 17:47:34.0161 [1568] : ntsecurity.cpp      ( 903): GetNTFileSecurity(E:\Documents and Settings\Andy\PrintHood):
Entry.

                B/A Performance thread        3================>
09/06/2001 17:47:34.0161 [1516] : bacpfm.cpp          ( 345): consumers: 4, producers: 4, on txnQ: 0, on baSpecQ 4.
09/06/2001 17:47:34.0161 [1516] : bacpfm.cpp          ( 466): Txn queue wait time is: 11036, consumer threshold: 10000,
producer threshold: -10000.
09/06/2001 17:47:34.0161 [1516] : bacpfm.cpp          ( 468): Current txn Q entry is 0, previous entry is ffffffd
09/06/2001 17:47:34.0161 [1516] : dsfifo.cpp          ( 362): fifoQinsert(a9c890): Posting that next object available.
09/06/2001 17:47:34.0161 [1516] : dsfifo.cpp          ( 367): fifoQinsert(a9c890): Queue insert of entry deadbeef, return rc
of 0
09/06/2001 17:47:34.0161 [1516] : bacpfm.cpp          ( 517): Txn queue change is -11036
09/06/2001 17:47:34.0161 [1516] : bacpfm.cpp          ( 533): Current ba Spec Q entry is  deadbeef, previous entry is fffffffd
09/06/2001 17:47:34.0161 [1516] : psthread.cpp        ( 850): Thread 000005EC delaying for 1 seconds.

                B/A Txn Consumer thread       4================>
09/06/2001 17:47:34.0221 [1512] : session.cpp         (2757): Sess (00A9C12C) FREELOCK lock action by thread (5e8):
09/06/2001 17:47:34.0221 [1512] : dsfifo.cpp          ( 587): fifoQgetNext(a9c890): Retrieved entry deadbeef, but next entry
is NULL.
09/06/2001 17:47:34.0221 [1512] : dsfifo.cpp          ( 597): fifoQgetNext(a9c890): Returning entry deadbeef, rc 0.
09/06/2001 17:47:34.0221 [1512] : session.cpp         (2757): Sess (00A9C12C) FREELOCK lock action by thread (5e8):
09/06/2001 17:47:34.0221 [1512] : DccTaskStatus.cpp   ( 850): Entering --> DccTaskStatus::ccDeleteTasklet
09/06/2001 17:47:34.0221 [1512] : DccTaskStatus.cpp   ( 990): Entering --> DccTaskStatus::ccUpdateStats
09/06/2001 17:47:34.0251 [1512] : DccTaskStatus.cpp   ( 874): Exiting  --> DccTaskStatus::ccDeleteTasklet
09/06/2001 17:47:34.0251 [1512] : thrdmgr.cpp         ( 772): Thread B/A Txn Consumer exiting, result =0
```

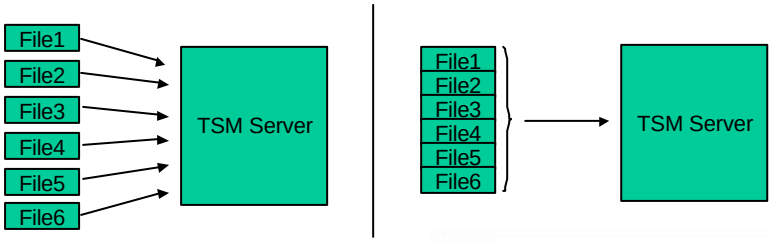
Observing multithreading with a trace (cont.)

```
09/06/2001 17:47:34.0251 [1512] : psthread.cpp        ( 693): Thread 000005E8 exit called.
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 13; ThrdSpecificData[key][i],
[0][i]: 0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 12; ThrdSpecificData[key][i],
[0][i]: 0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 11; ThrdSpecificData[key][i],
[0][i]: 0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 10; ThrdSpecificData[key][i],
[0][i]: 0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 9; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 8; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 7; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 6; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 5; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 4; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 3; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 2; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 1; ThrdSpecificData[key][i], [0][i]:
0, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : pktsd.cpp           ( 410): TSD callCleanUps i, key: 4, 0; ThrdSpecificData[key][i], [0][i]:
a9016c, a9016c.CleanUpFunctions[i] = 0
09/06/2001 17:47:34.0251 [1512] : thrdmgr.cpp         (1124): Thread 4 (B/A Txn Consumer) TSD descP (5e8) clean up.
09/06/2001 17:47:34.0251 [1512] : bacontrl.cpp        (1942): Entering --> DccTxnConsumer::Cleanup
09/06/2001 17:47:34.0251 [1512] : senddata.cpp        (3617): dsDestroyConfirmSettings(): Total cuConfirms issued: 1 .
```


Transaction processing

TSM transactions

- The process involved in updating the TSM database is not entirely negligible.
- Bundling multiple file operations into a single unit of work mitigates a lot of the overhead.



TSM transactions

- Transactions are bounded by the following:
  - Sum of the sizes of all files in the txn (TXNBYTELIMIT)
  - Number of files in the txn (TXNGROUPMAX)
  - Destination (STGPOOL)
- Rule of thumb: the larger the txn, the better the over-all performance.

Building a transaction

- Transactions are built by the Txn Producer:
  - Determine the size of the object
  - If the size of the object plus the size of the current txn exceeds TXNBYTELIMIT, then don't add to the current txn.
    - 300 ≤ TXNBYTELIMIT (KB) ≤ 2097152
  - If the number of items in the current txn is already at TXNGROUPMAX, then don't add to the current txn.
    - 4 ≤ TXNGROUPMAX ≤ 256
  - If the object's destination is different than that for the other items in the current txn, then don't add to the current txn.
    - DEST field in COPYGROUP definition

Building a transaction (cont.)

  - If the current object is a directory and the copy serialization is SHARED STATIC, then change the copy serialization to SHARED DYNAMIC.
  - Add object to the current txn.
  - When the txn has been built
    - queue for processing by a Consumer
    - notify Consumer that a txn is available for processing.

Observing transactions with a trace

TXN.TID

```
                B/A Txn Producer thread         2================>
09/14/2001 03:19:23.0220 [500] : txnprod.cpp          (1835): tlInit: Initializing transaction list, sparseTree: 0

                B/A Txn Producer thread         5================>
09/14/2001 03:19:29.0527 [1644] : txnprod.cpp         (1835): tlInit: Initializing transaction list, sparseTree: 0
09/14/2001 03:19:29.0537 [1644] : filespac.cpp        (2942): fsGetFsRenameState: Client decide PROMPT
09/14/2001 03:19:29.0668 [1644] : txnprod.cpp         (2110): tlBackObj: Add obj type 1 '\\amraibeck\d$' ''
'\tsm421c.odbc.source.zip' at 0, MC: 1, CG: 1

                B/A Txn Consumer thread         4================>
09/14/2001 03:19:29.0688 [1640] : bacontrl.cpp        (1736): Consumer txn entry is: 001A23E8
09/14/2001 03:19:29.0688 [1640] : txncon.cpp          (2629): PrivFlush: count 0, bytes 0
09/14/2001 03:19:29.0688 [1640] : txncon.cpp          (2281): Txn Post Processing 1a23e8, counter is 1, rc is 140.

                B/A Txn Producer thread         5================>
09/14/2001 03:19:29.0048 [1644] : txnprod.cpp         (2110): tlBackObj: Add obj type 1 '\\amraibeck\d$' '\Documents and
Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\YXAXMFQ5' '\DTUE[1].css' at 1, MC: 1, CG: 1

                B/A Txn Consumer thread         4================>
09/14/2001 03:19:29.0048 [1640] : bacontrl.cpp        (1736): Consumer txn entry is: 001B10BC
09/14/2001 03:19:29.0048 [1640] : txncon.cpp          (2629): PrivFlush: count 1, bytes 105387403
09/14/2001 03:19:29.0048 [1640] : txncon.cpp          (2653): PrivFlush: Top of outer loop, num_to_complete: 1
09/14/2001 03:19:29.0048 [1640] : txncon.cpp          (3448): PrivFlush: Examining [0] '\\amraibeck\d$' ''
'\tsm421c.odbc.source.zip' retries: 0 rc: 0 state: 001
09/14/2001 03:19:29.0048 [1640] : senddata.cpp        (1575): sdSendObj: Sending a 105387216 byte File
'D:\tsm421c.odbc.source.zip'
09/14/2001 03:19:29.0048 [1640] : senddata.cpp        (2294): SendFileData(): DataVerbSize is 32764 Byte, 31 KByte.
09/14/2001 03:19:29.0068 [1640] : senddata.cpp        (3135): SendFileData: readLen = 32764
09/14/2001 03:19:29.0078 [1640] : senddata.cpp        (3135): SendFileData: readLen = 32764
```


Journal-based backup

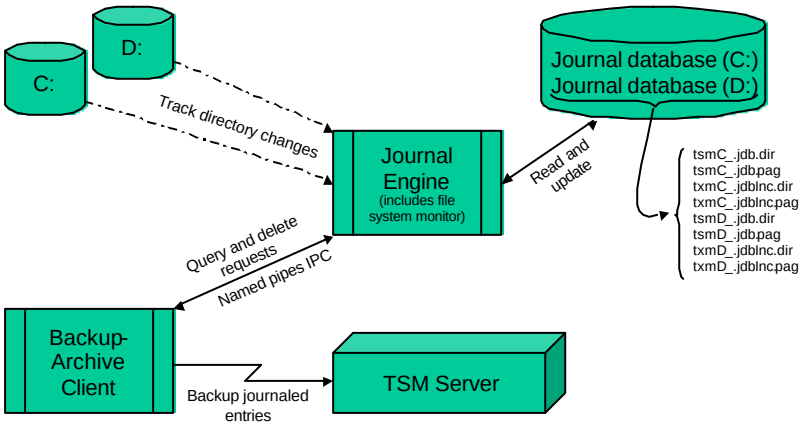
Objectives

- The goal of Journal-Based Backup (JBB) is to improve backup performance:
  - Eliminate the need to query the server for backup information
  - Eliminate the need to traverse the client file systems
  - Eliminate the need for large amounts of memory to store lists of server and local file information
  - To a lesser extent, reduce the network traffic necessary to obtain the information from the server

Components

- File System Monitor
  - Mechanism to detect changes to files and directories
- Change Journal Database
  - Contains entries for changed files and directories
  - Is randomly accessible
- Journal Engine
  - Interfaces with the File System Monitor
  - Controls access to the Change Journal Database
  - Runs as a background process (service on NT/2000)
- Named Pipe interface
  - Used to communicate with the Backup-Archive client

Journal-based backup



Considerations for journal-based backup

- How file and directory events affect the journal database

Event	Database activity
New file or directory	Add entry with timestamp of creation
Deleted file or directory	Add or update entry with timestamp of deletion
File or directory changes	Add or update entry with timestamp of change
File or directory attributes change	Add or update entry with timestamp of change

Considerations for journal-based backup

- Successful backup (regardless of type) of each file is “recorded” in the journal by removing its entry, if it exists.
- Normal client include/exclude criteria are applied to journal-based backup.
- The client will (attempt to) detect and identify situations which would cause the journal to be invalid or the journal-based backup to be incomplete.

Considerations for journal-based backup

- Regular incremental backup will be forced if the journal is deemed invalid.
- The journal is treated as invalid if:
  - files are deleted on the server for a journaled file space
  - a journaled file space is renamed
  - the node name changes
  - a new policy set is activated
  - the journal is corrupt
  - the journal service is stopped for any reason, including system reboot

Considerations for journal db integrity

- Client events that impact journal db integrity:
  - Failure in the journal process
  - Journal not running
  - Include/Exclude changes
  - Corruption of the journal db
  - Corruption of the client file system

Considerations for journal db integrity

- The client file space on the server has integrity with respect to the client when all of the following are true:
  - The file space is defined on the server
  - The last backup start date/time is valid
  - The last backup end date/time is valid
  - The last backup end date/time is greater than the last backup start date/time
  - The file space deletion date/time is less than the last backup start date/time

Considerations for journal db integrity

- Server events that impact journal db integrity:
  - DELETE FILESPACE
  - DELETE VOLUME
  - AUDIT VOLUME with FIX=YES
  - RENAME FILESPACE
  - Restoring the TSM database
  - Database audit operations

Limitations of journal-based backup

- Copy frequency other than 0 can not be enforced.
- Attribute-only updates are not possible; the entire object will need to be backed up.
- Only one journal engine process may be installed on a machine.
- For a particular file space, the journal is valid only for a particular server and node.