Transaction Support for Log-Based Middleware Server Recovery

Presented by XiaoFei Zhao

Outline

- Introduction.
- *Middleware server process* (MSP) architecture.
- State recovery based on logging.
- State recovery based on results logging.
- Caveats about database management systems (DBMSs).
- Related work.
- Conclusion.

Introduction

- Middleware servers.
- High availability and exactly-once semantics.
- Recovery after failure.
- Middleware server process & concurrency.
- Transactional methods.
- Results logging.
- Low overhead and little or no change to back-end infrastructure.

Middleware server process (MSP) architecture

Sessions & clients

One request per session

Session variables vs. shared variables

Transactional methods & transactional requests

Committed transactions vs. aborted transactions

Local transactions vs. distributed transactions

Strict two phase locking

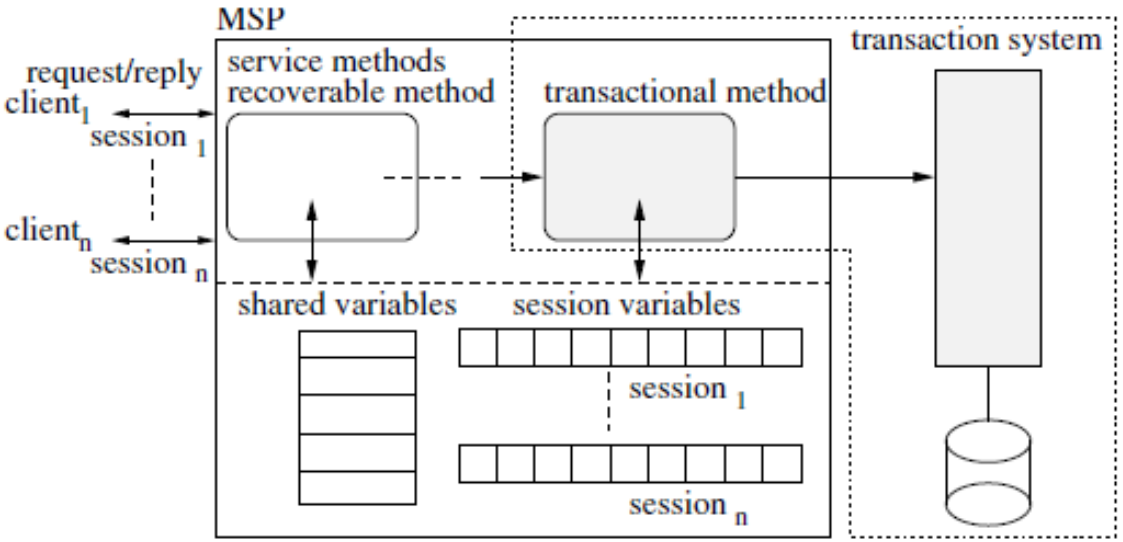

Fig. 1. Transaction-extended MSP architecture.

State recovery based on logging

Log all non-deterministic events
Log shared variables, do not log session variables
Replay log to recover
Session variables: re-execute; shared variables: read log
Take checkpoints

TABLE I

AN EXECUTION SCENARIO AND ITS LOG RECORDS.

execution scenario	*log records*
execute recoverable method 1	request for method 1 with parameters
read session variable *sev1*	
read shared variable *shv1*	value of *shv1* being read
write *sev1*	
write *shv1*	new value of *shv1* being written
execute recoverable method 2	request for method 2 with parameters
read session variable *sev2*	
write *sev2*	
write shared variable *shv1*	new value of *shv1* being written

State recovery based on results logging

Log the results of execution

Read most recent values of both shared vars and session vars to recover

Save the whole in-memory state

TABLE II

A TRANSACTIONAL METHOD EXECUTION AND RESULTS LOGGING.

execution scenario	*logging*
execute transactional method 1	**request log record** for method 1
read session variable *sev1*	
read shared variable *shv1*	
open a DBMS connection	
read/write database	
write *sev1*	
write *shv1*	
return *retv*	
	committing log record:
	retv, new values of *sev1* and *shv1*
	flush log buffer
	send commit decision to TranMan
	wait for transaction outcome
	result-status log record:
	committed or aborted
automatically close connection	

Caveats about database management systems (DBMSs).

- Cannot re-execute transactional methods.
- Flush log buffer before committing.
- Let the transaction manager remember committed transactions' status until being explicitly told to discard such status.
- Use distributed log flush with dependency vector.

Related work

- Fault tolerance via replication.
- E-transactions.
- Phoenix project: message logging.
- Transactional Web methods.

Conclusion

- Transaction support for log-based recovery of middleware servers.
- Results logging can recover both in-memory business state and persistent business state while incurring modest overhead and requiring almost no change to existing transaction systems.
- Results logging and existing transaction system recovery facilities take care of system failures.


Thank you !
XiaoFei Zhao

Discussion

- If the middleware crashes but all transactions were recorded in a persistent storage medium which survived the crash, then can we recover the system to its state just before the last recorded transaction?
- What are the strengths and weaknesses of the replication-based and log-based recoveries respectively?