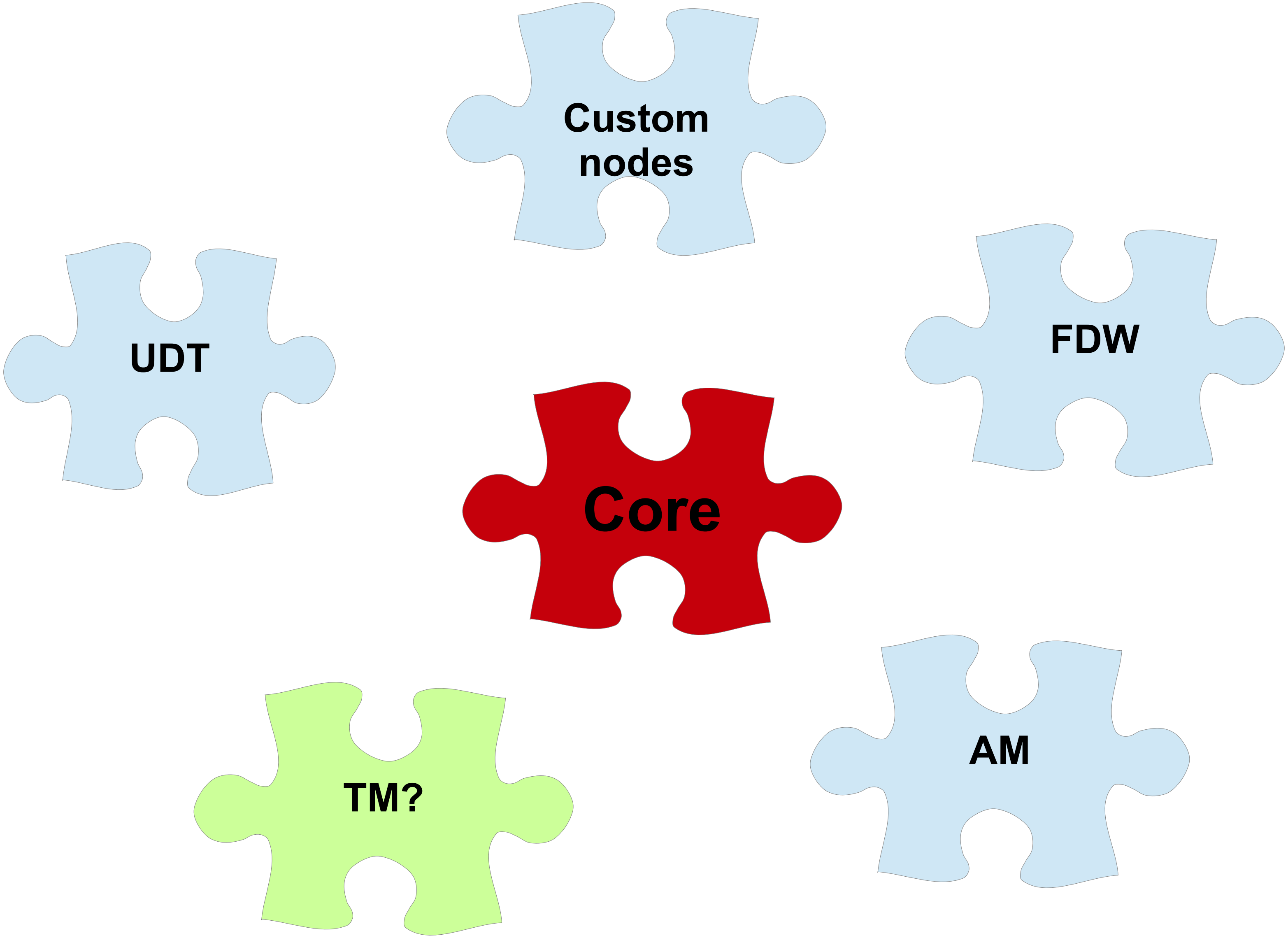
Distributed Transaction Manager

Stas Kelvich
Konstantin Knizhnik
Konstantin Pan

Мы пойдём другим путём...

Pluggable transaction API



Custom nodes

UDT

FDW

Core

TM?

AM

eXtensible Transaction API

- XidStatus (*GetTransactionStatus)(TransactionId xid, XLogRecPtr *lsn);
- void (*SetTransactionStatus)(TransactionId xid, int nsubxids, TransactionId *subxids, XidStatus status, XLogRecPtr lsn);
- Snapshot (*GetSnapshot)(Snapshot snapshot);
- TransactionId (*GetNewTransactionId)(bool isSubXact);
- TransactionId (*GetOldestXmin)(Relation rel, bool ignoreVacuum);
- bool (*IsInProgress)(TransactionId xid);
- TransactionId (*GetGlobalTransactionId)(void);
- bool (*IsInSnapshot)(TransactionId xid, Snapshot snapshot);

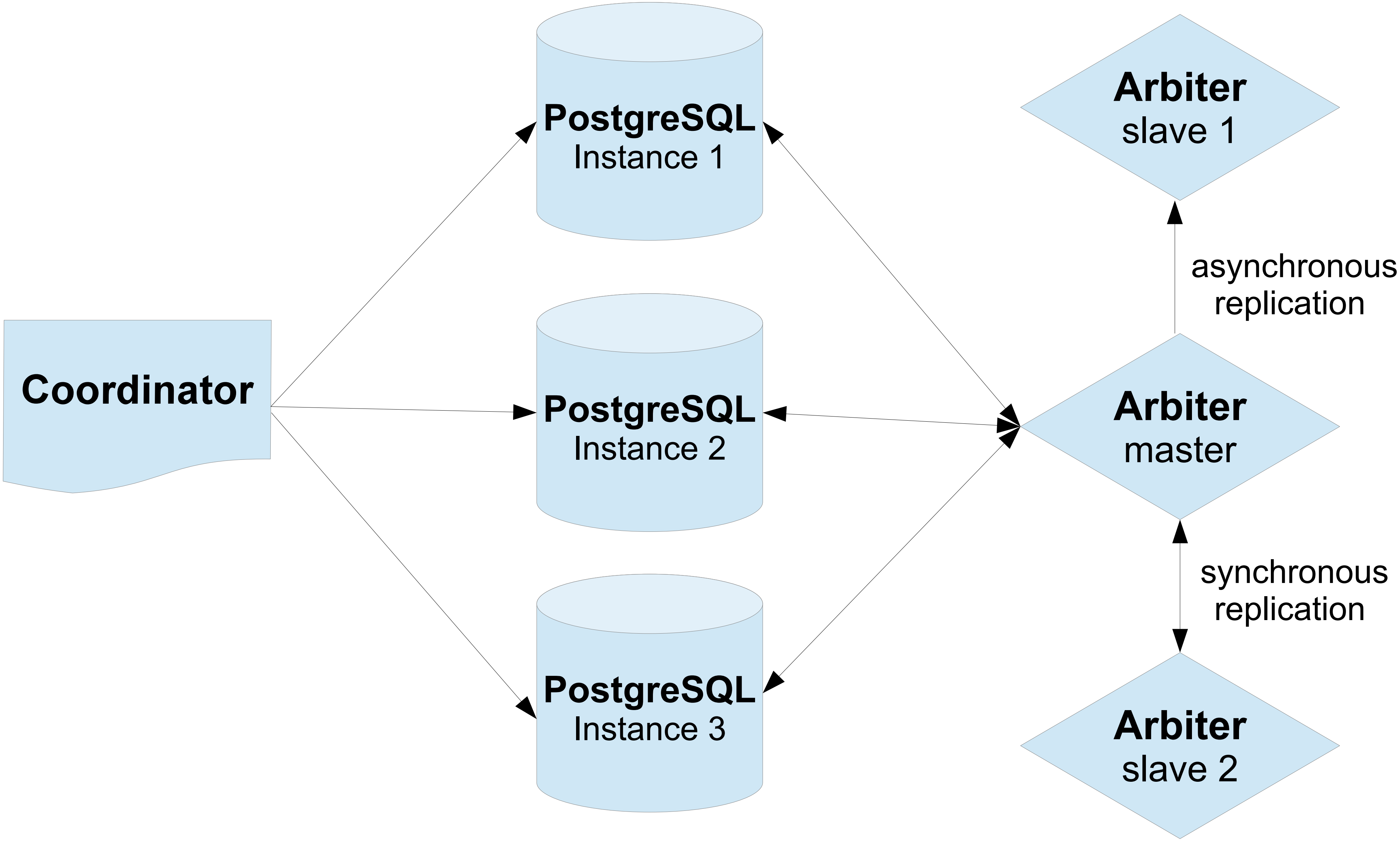
New commit callback events

- XACT_EVENT_START,
- XACT_EVENT_COMMIT,
- XACT_EVENT_PARALLEL_COMMIT,
- XACT_EVENT_ABORT,
- XACT_EVENT_PARALLEL_ABORT,
- XACT_EVENT_PREPARE,
- XACT_EVENT_PRE_COMMIT,
- XACT_EVENT_PARALLEL_PRE_COMMIT,
- XACT_EVENT_PRE_PREPARE,
- XACT_EVENT_COMMIT_PREPARED,
- XACT_EVENT_ABORT_PREPARED

Different DTM implementations

	Local transactions	2PC	Arbiter	Examples
Snapshot sharing			✓	XL, DTM
Timestamp	✓	✓		Spanner, Cockroach, tsDTM
Incremental	✓		✓	SAP HANA

DTM architecture



DTM from client's point of view

Primary server	Secondary server
create extension pg_dtm;	create extension pg_dtm;

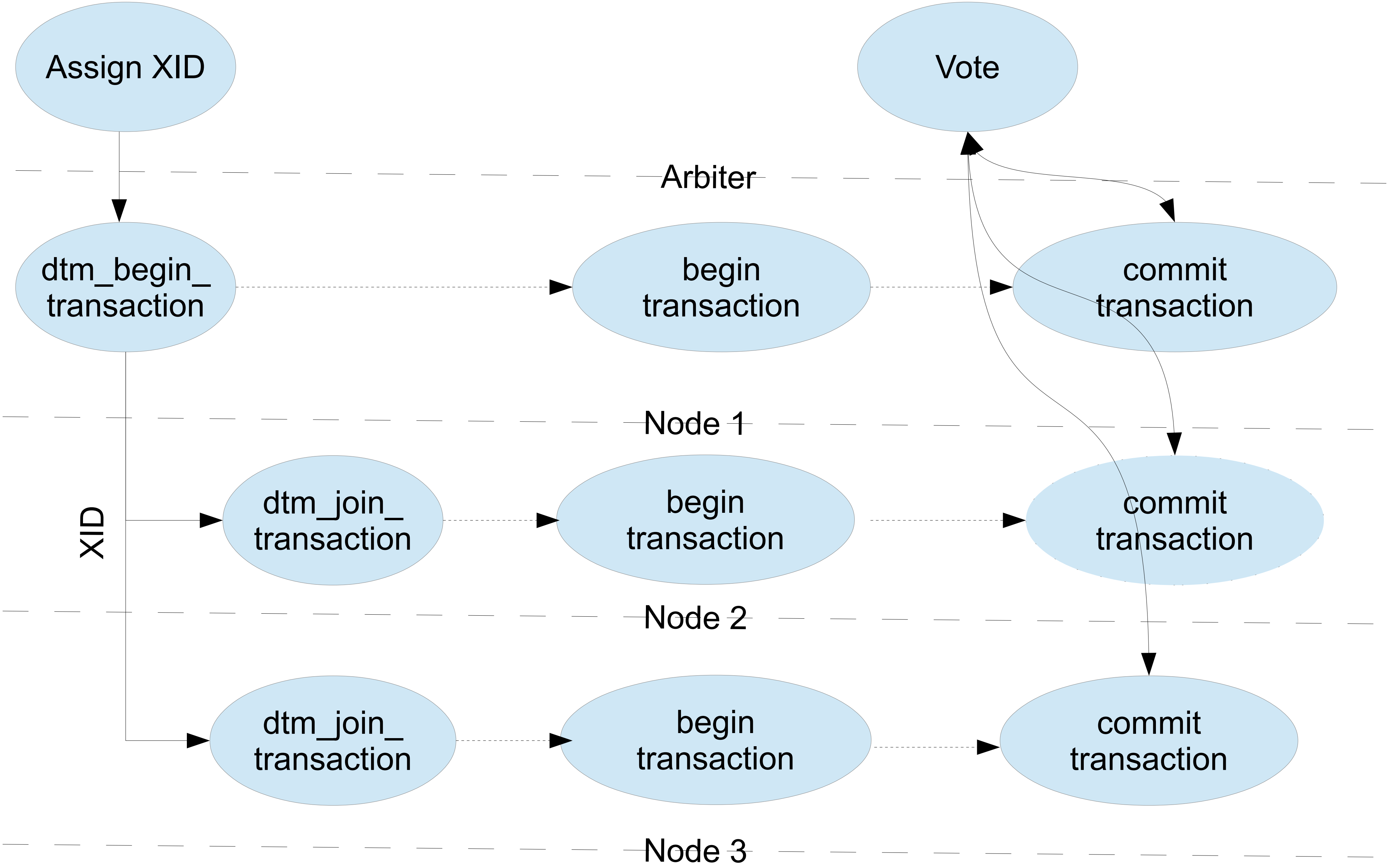
select dtm_begin_transaction();
begin transaction;
update...;
commit;
	select dtm_join_transaction(xid);
	begin transaction;
	ppdate...;
	commit;

DTM from client's point of view

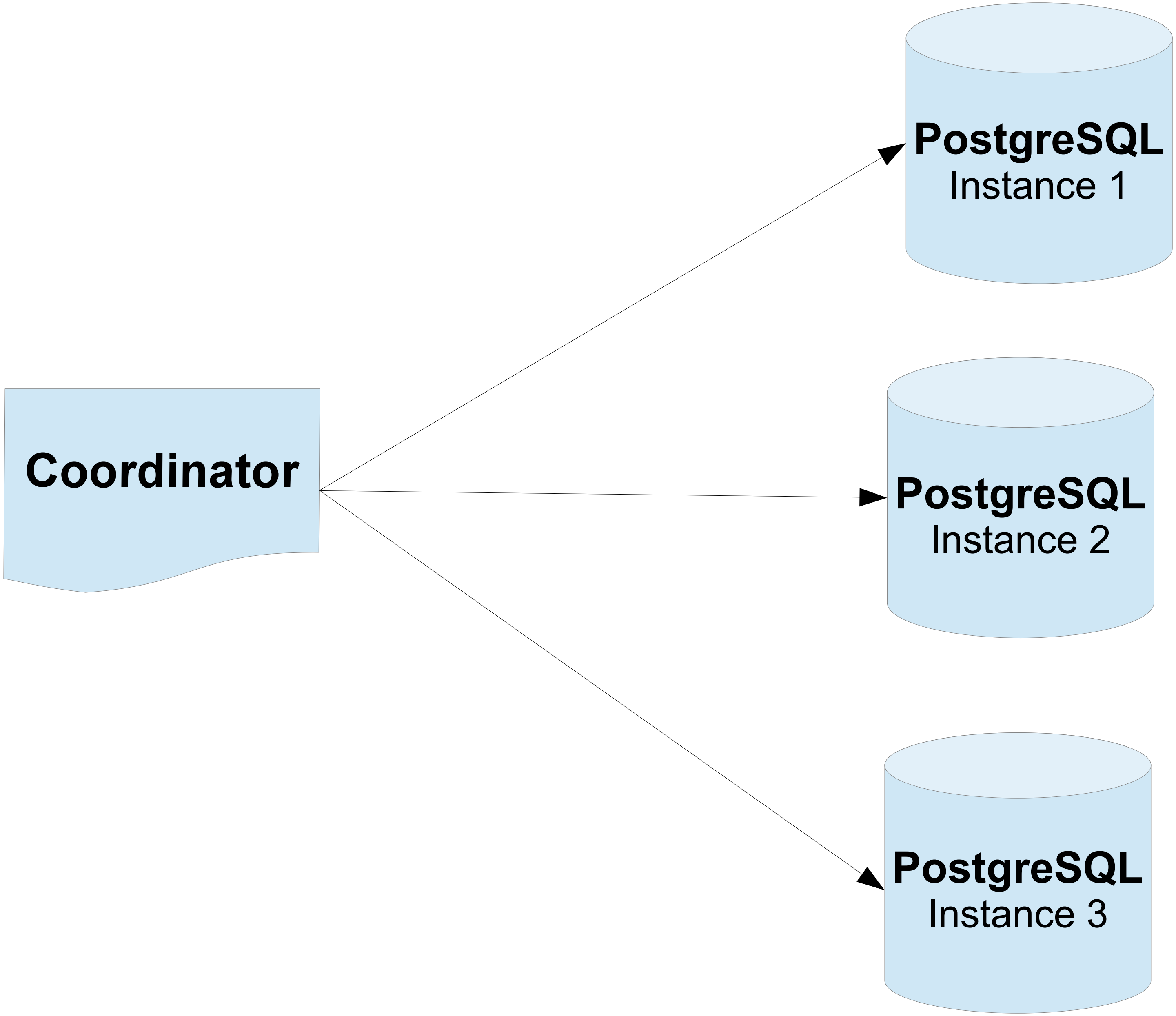
Primary server	Secondary server
create extension pg_dtm;	create extension pg_dtm;

select dtm_begin_transaction();
begin transaction;
update...;
commit;
	select dtm_join_transaction(xid);
	begin transaction;
	ppdate...;
	commit;

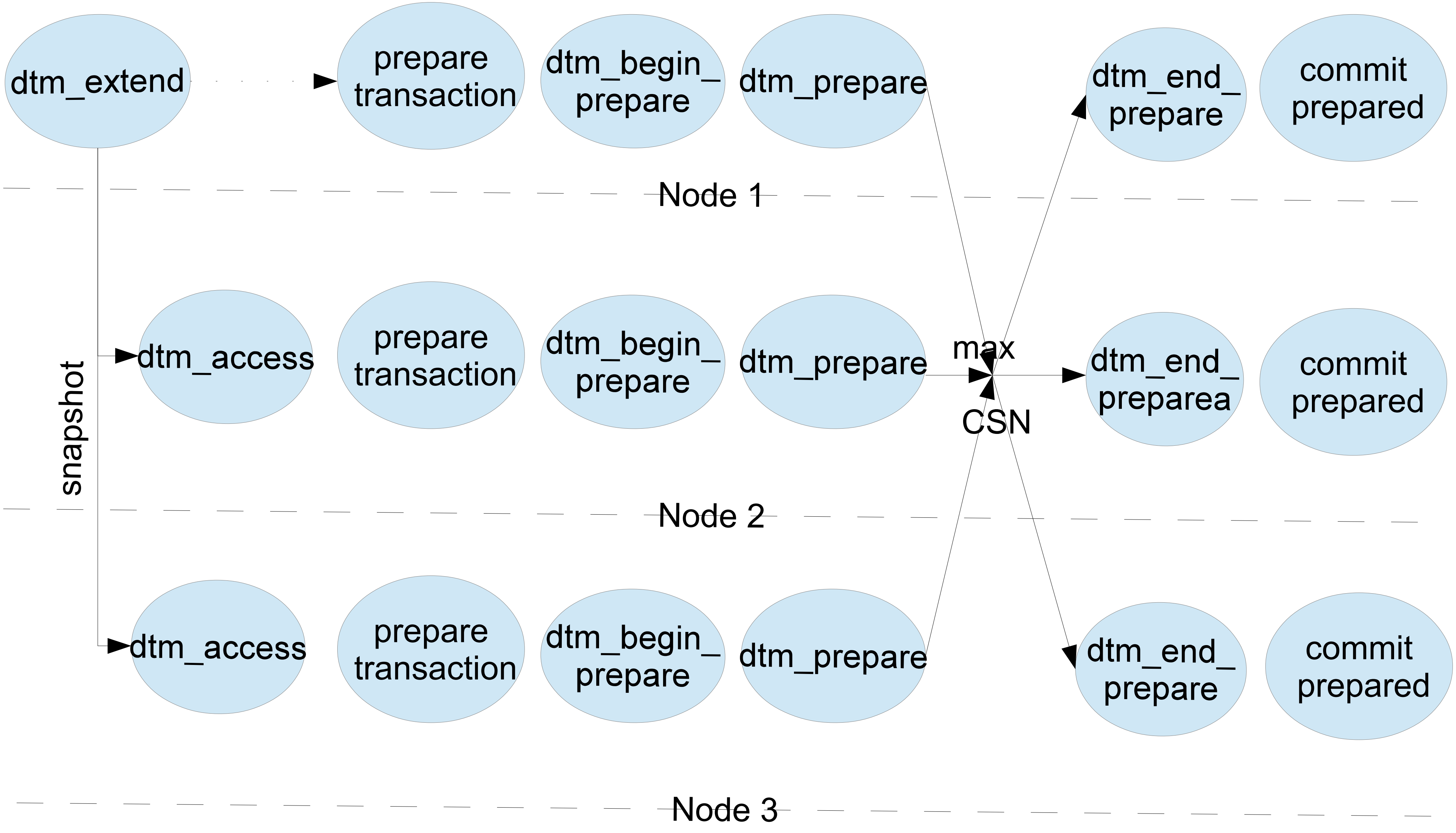
DTM transaction control flow



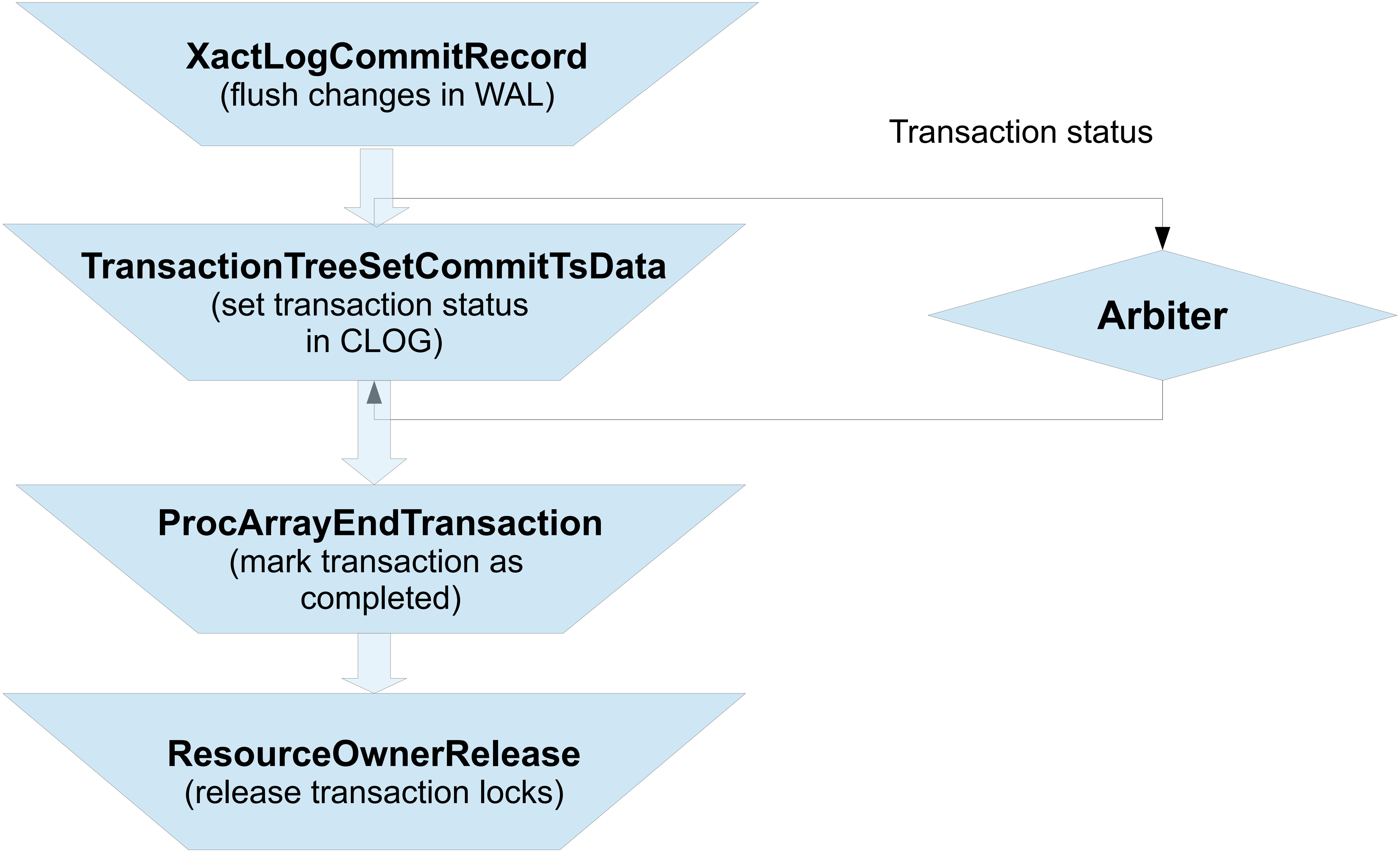
tsDTM architecture



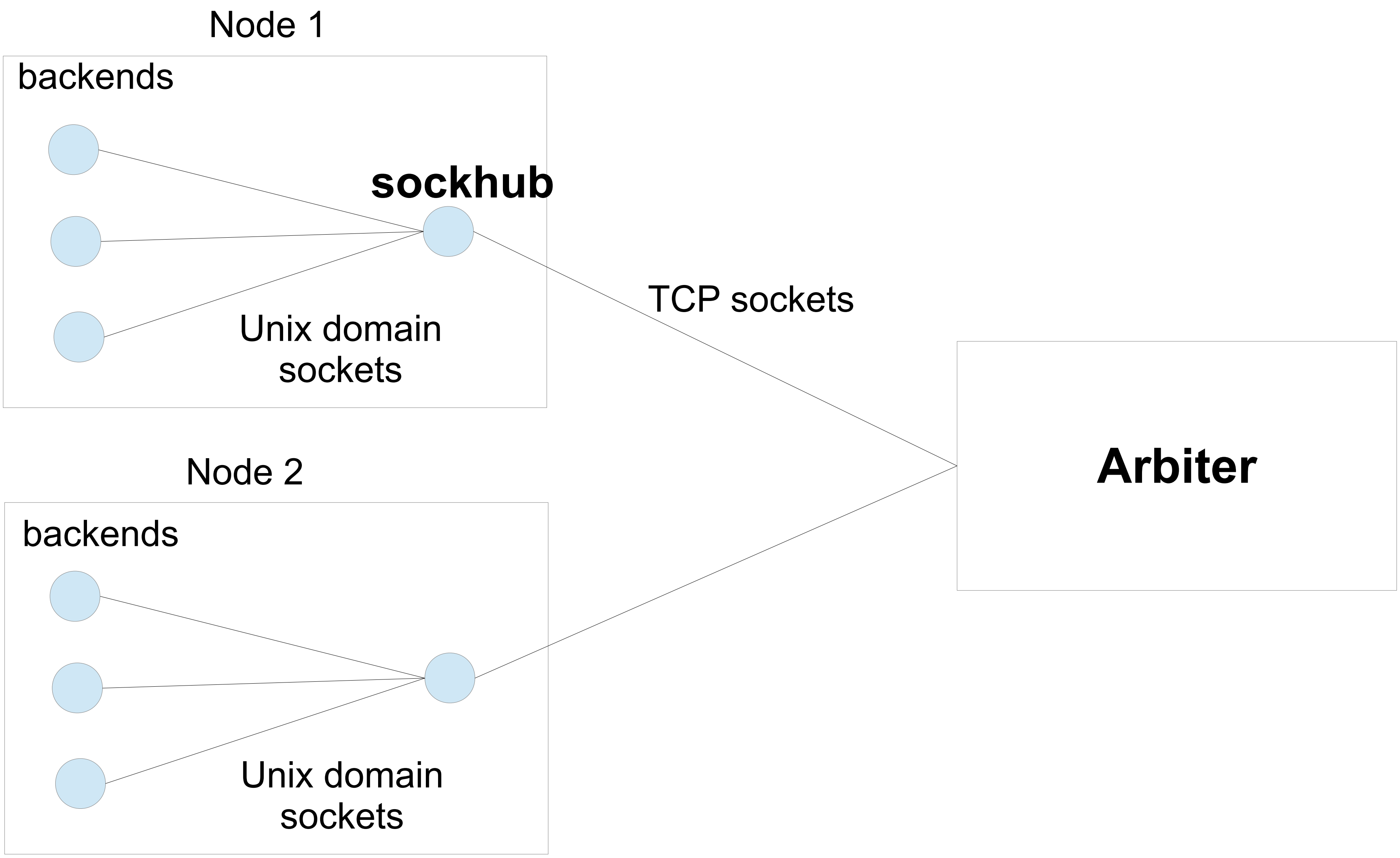
tsDTM transaction control flow



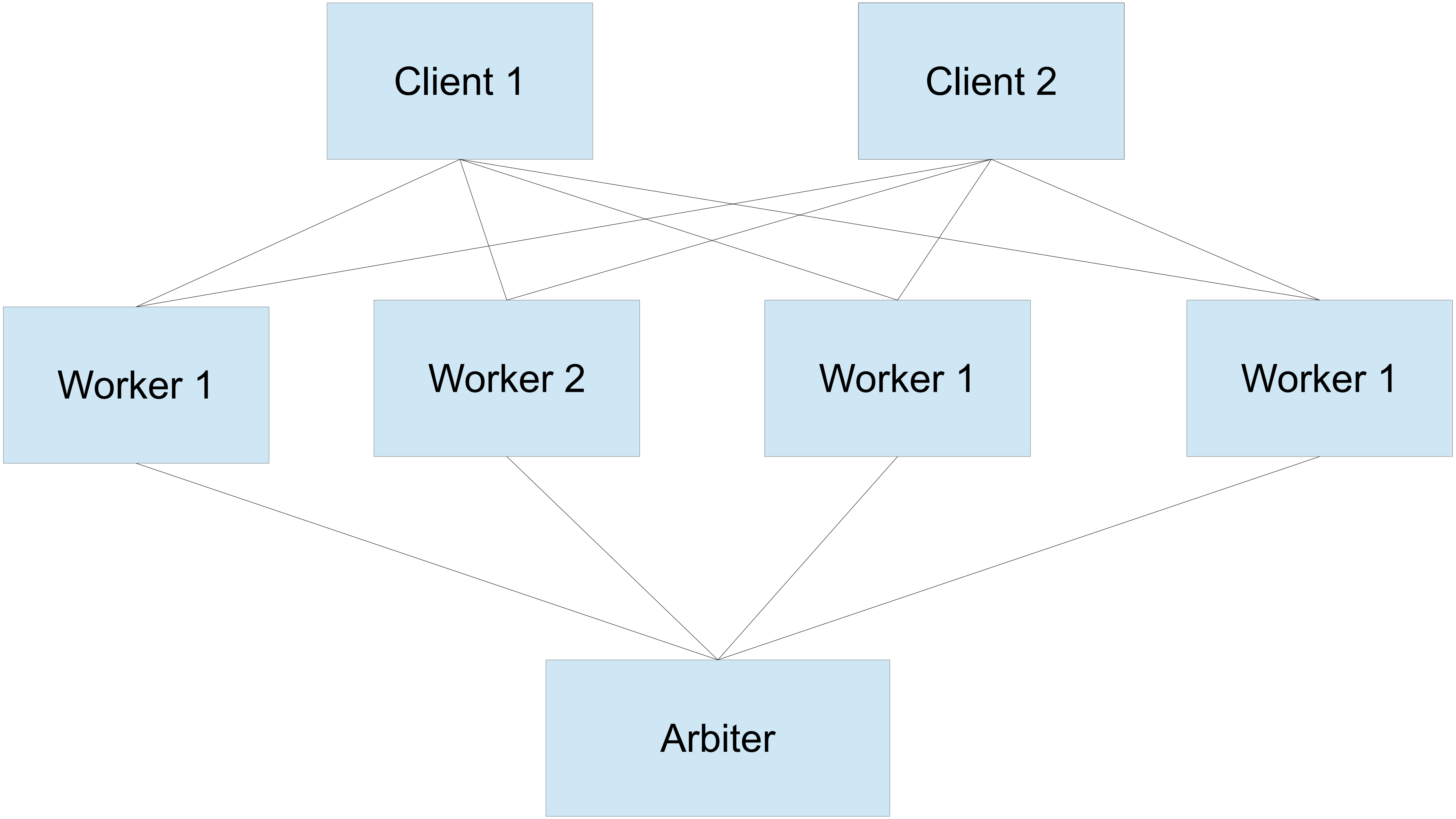
Lightweight two-phase commit



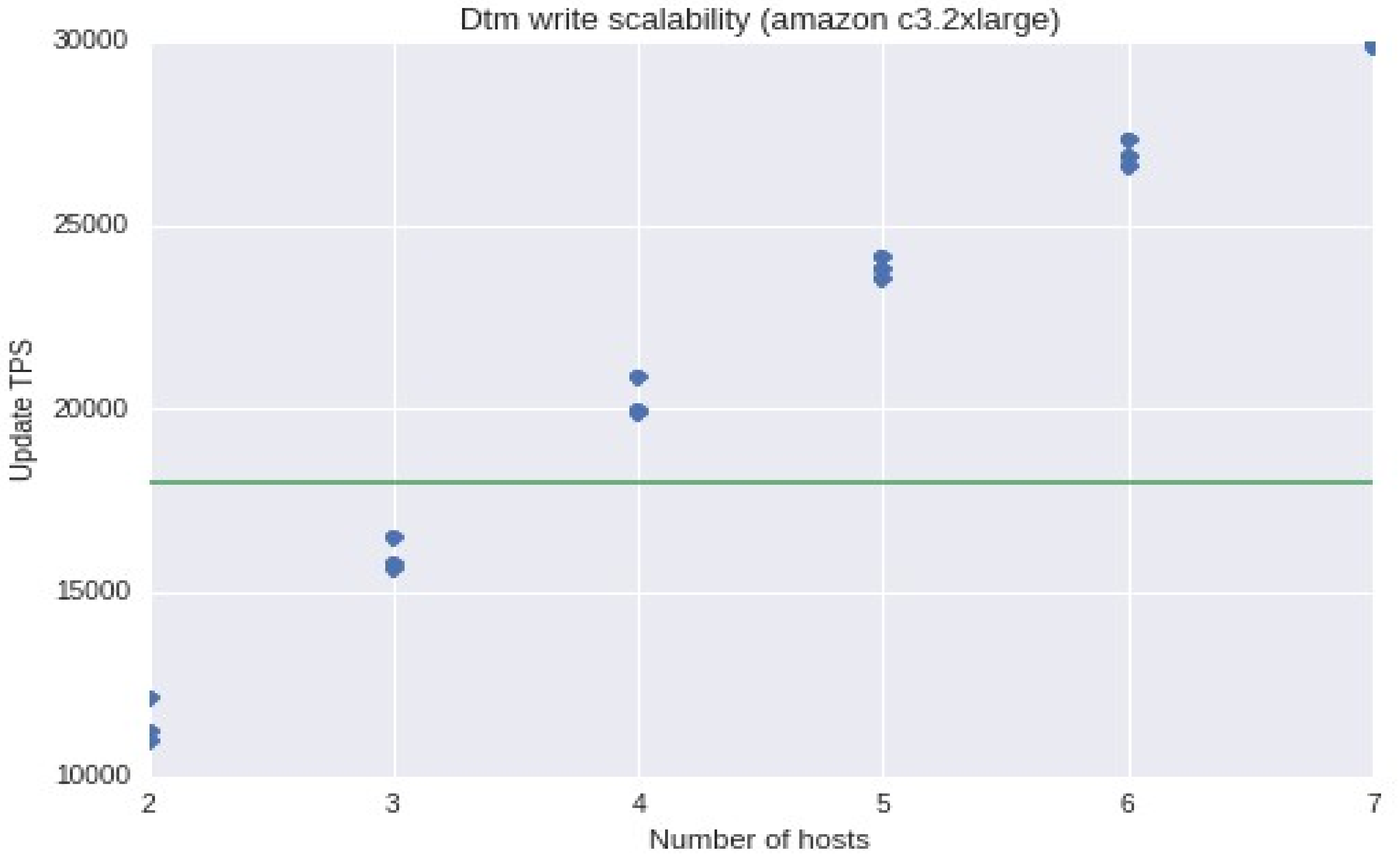
Multiplexing



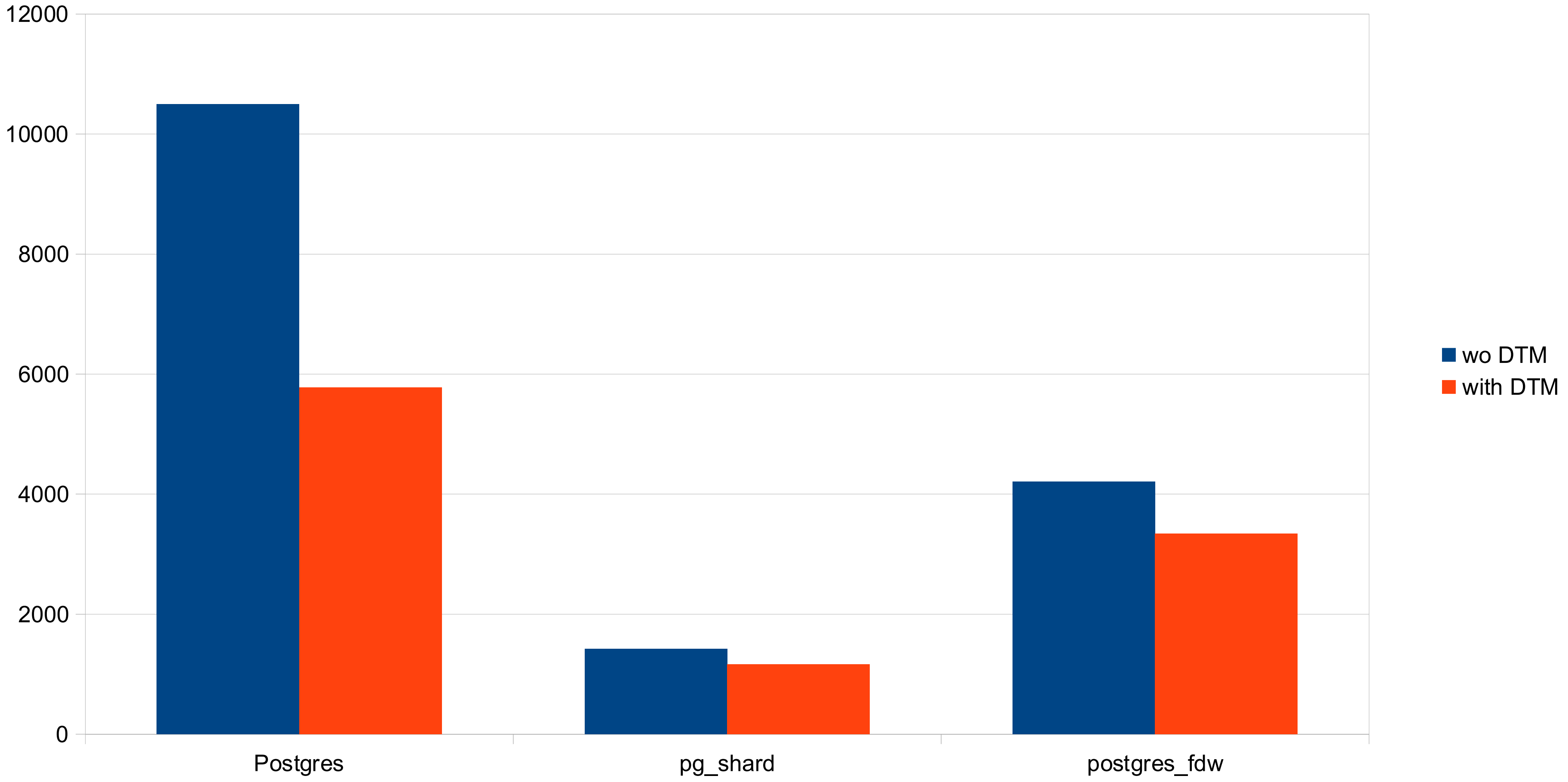
Test configuration



DTM scalability

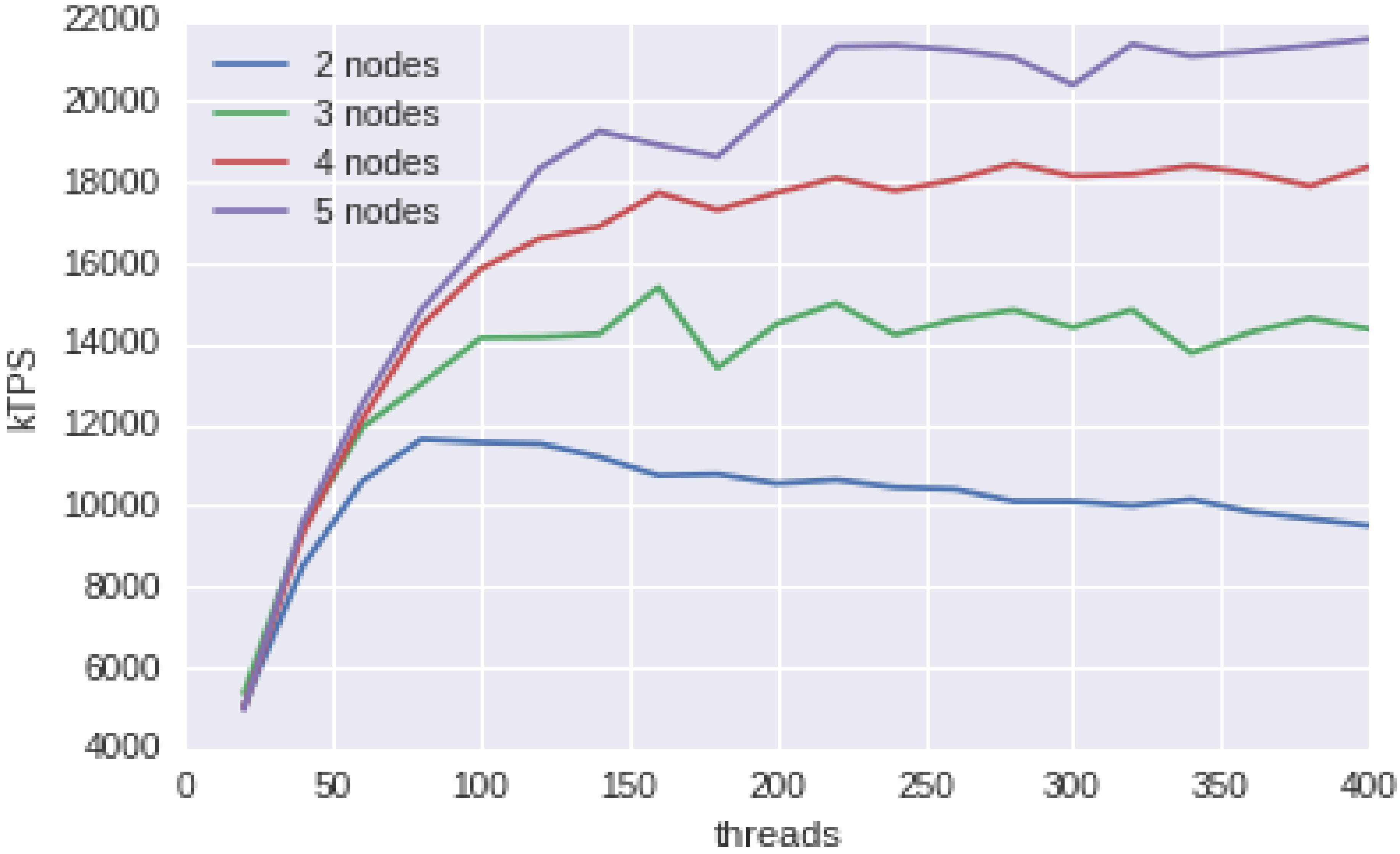


DTM overhead



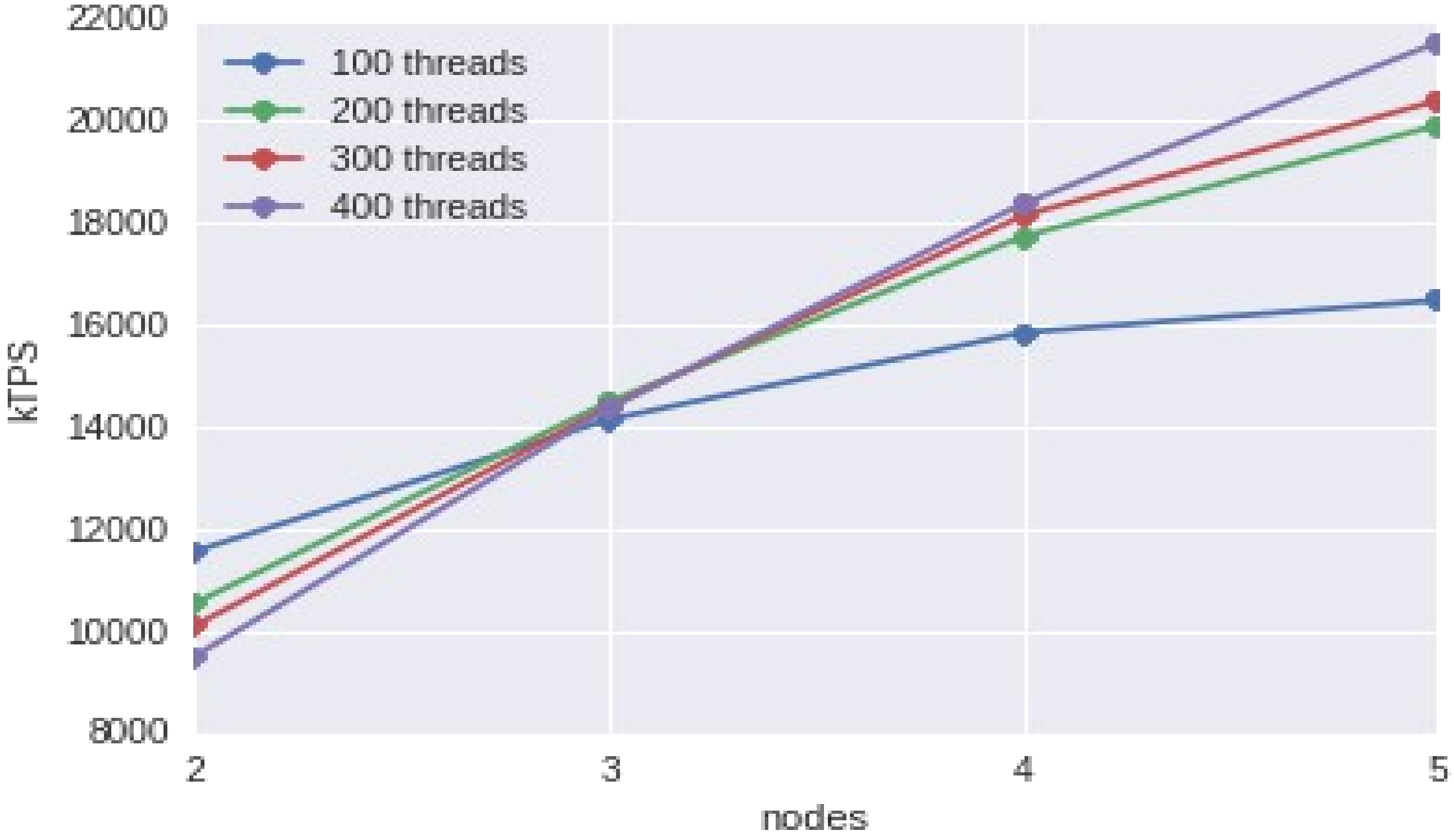
tsDTM scalability

(SAI cluster)

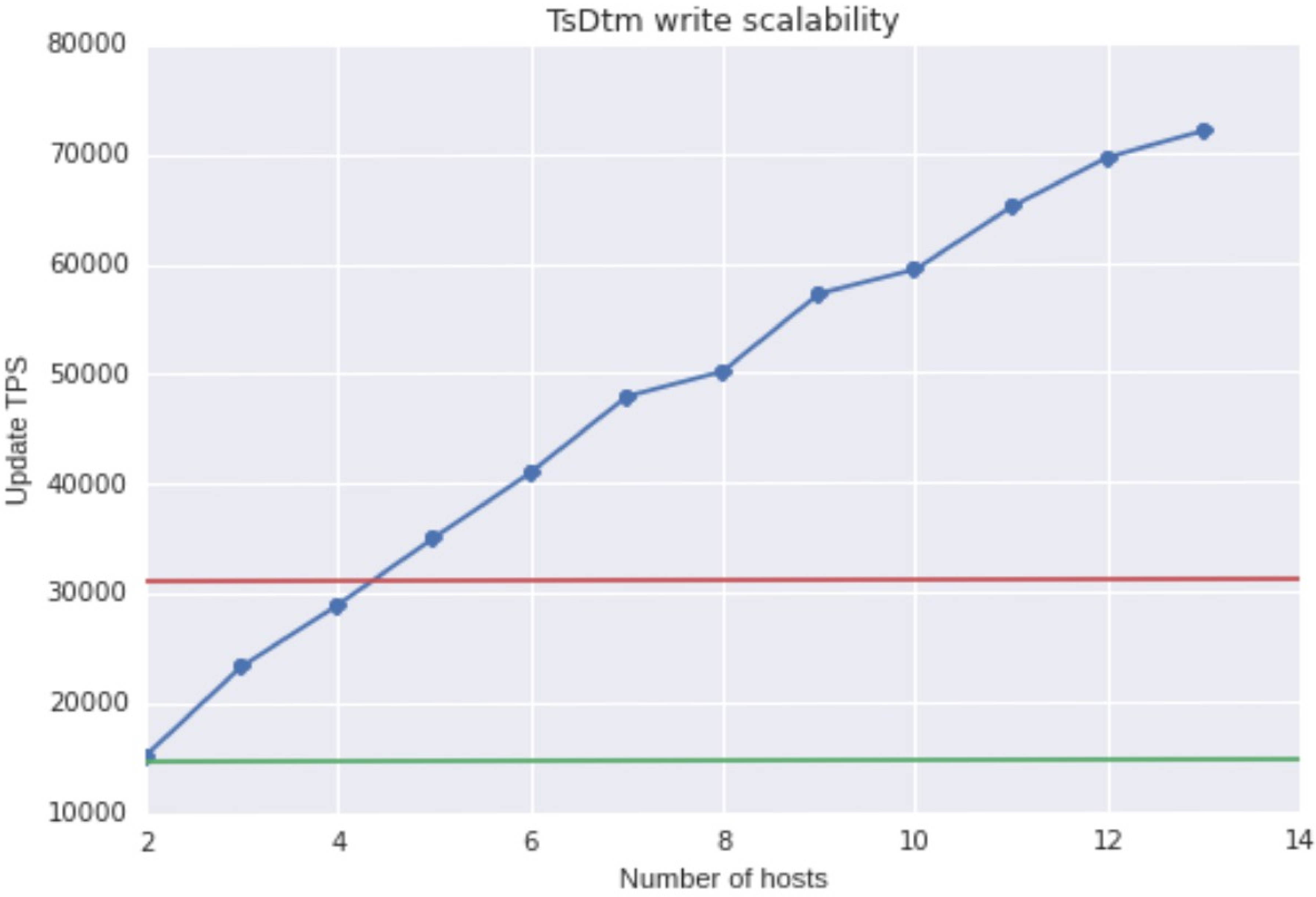


tsDTM scalability

(SAI cluster)



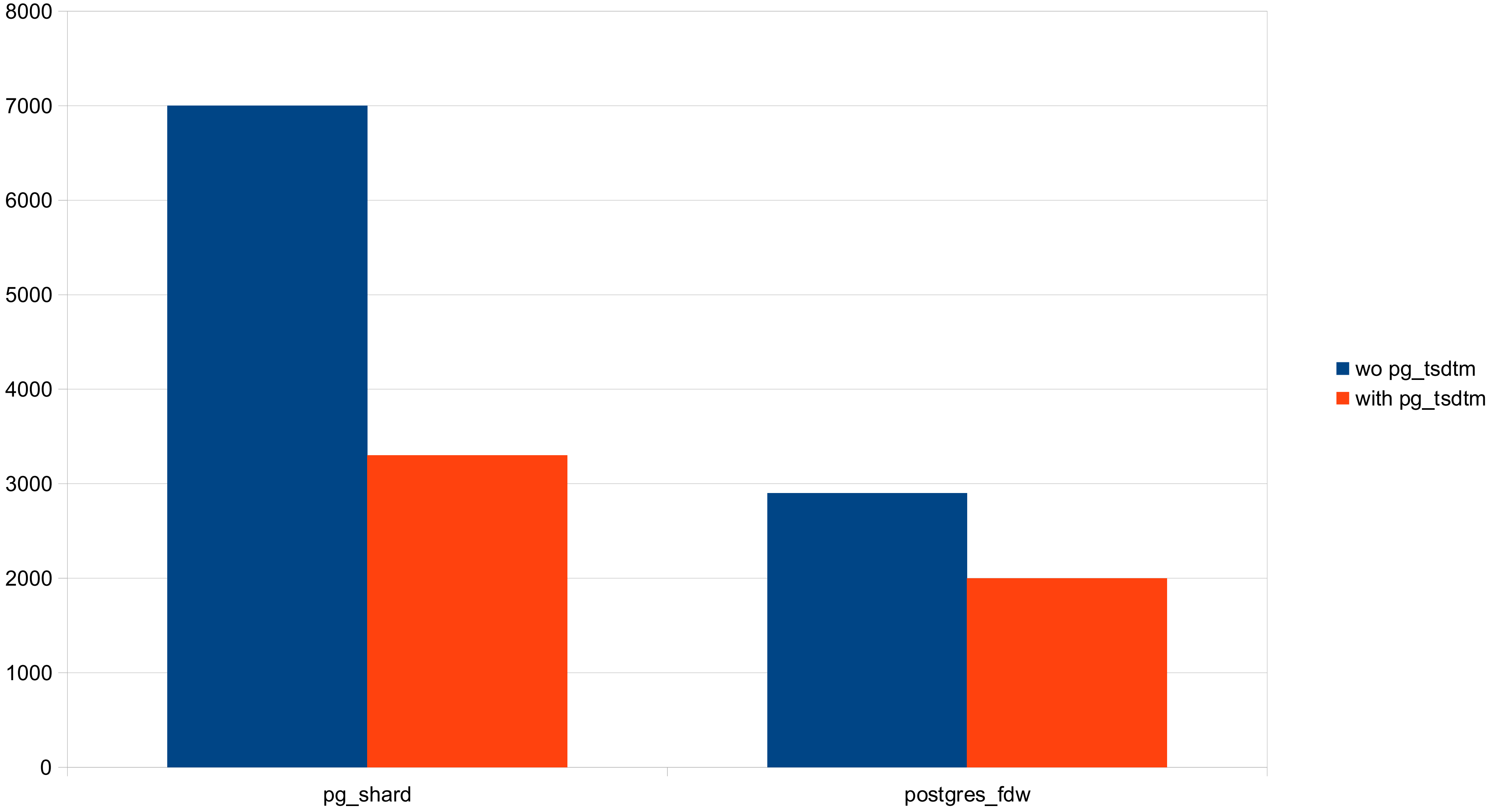
tsDTM write scalability



1PC

2PC

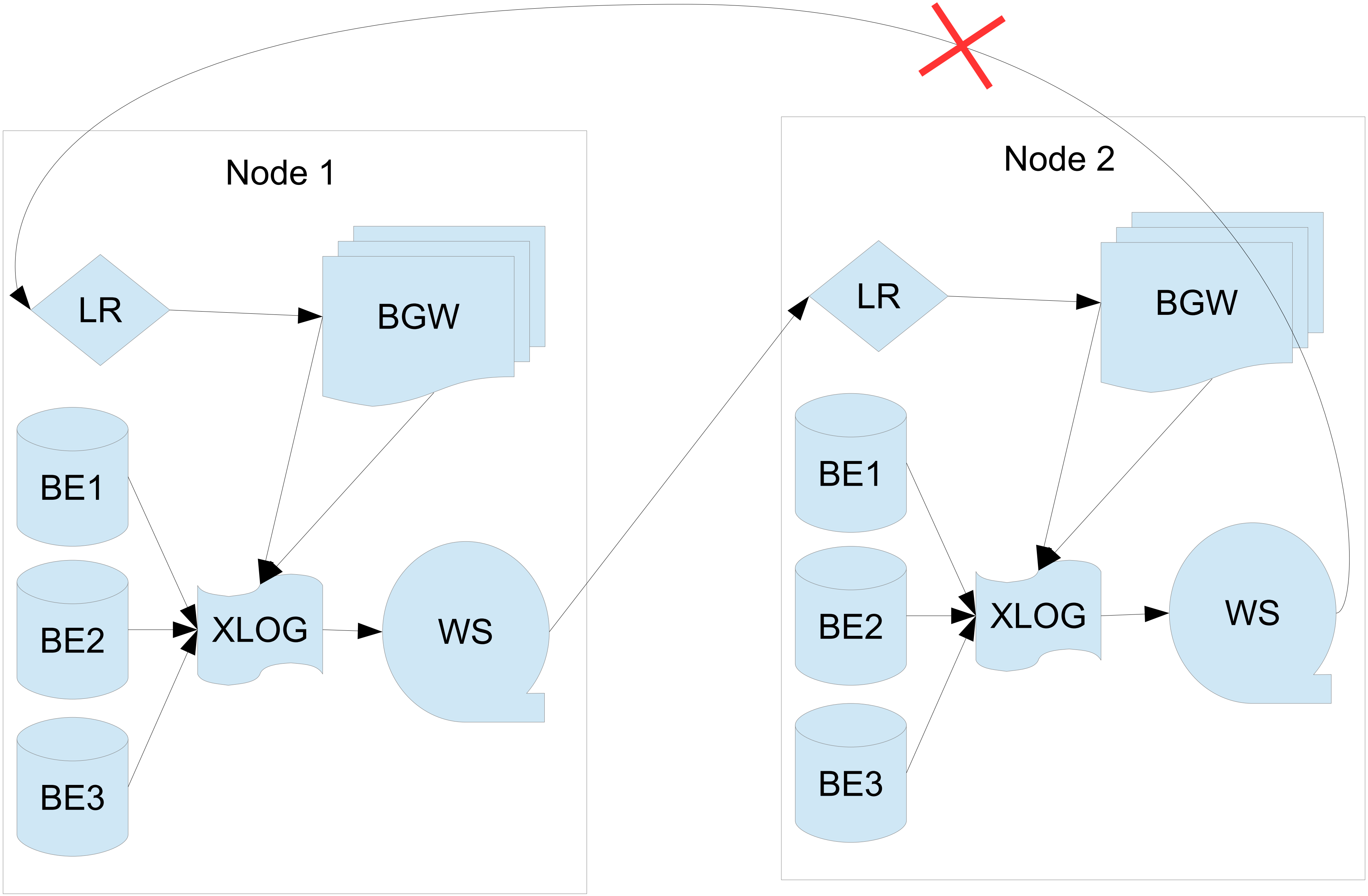
tsDTM overhead



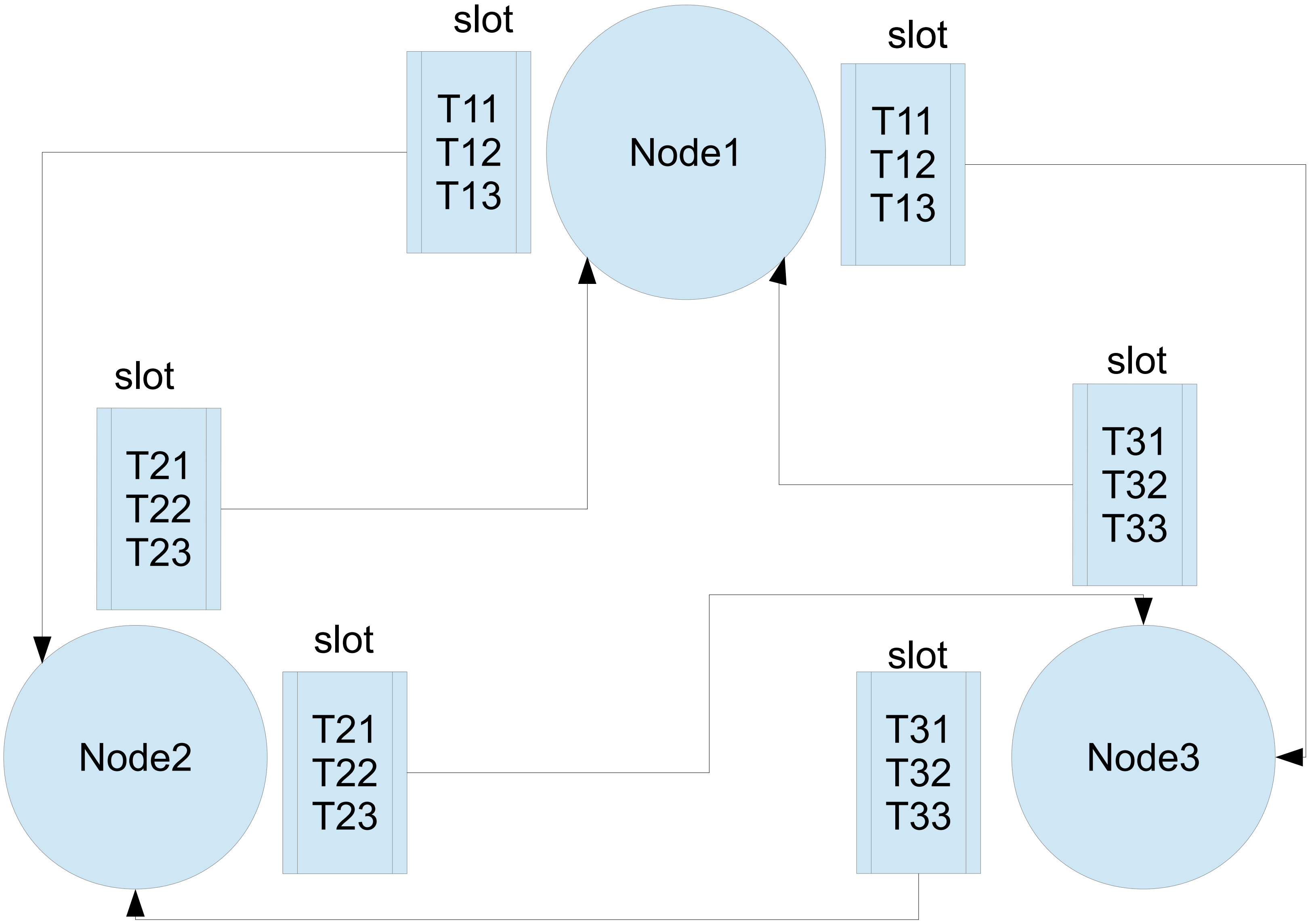
Multimaster formula

MM = BDR + DTM

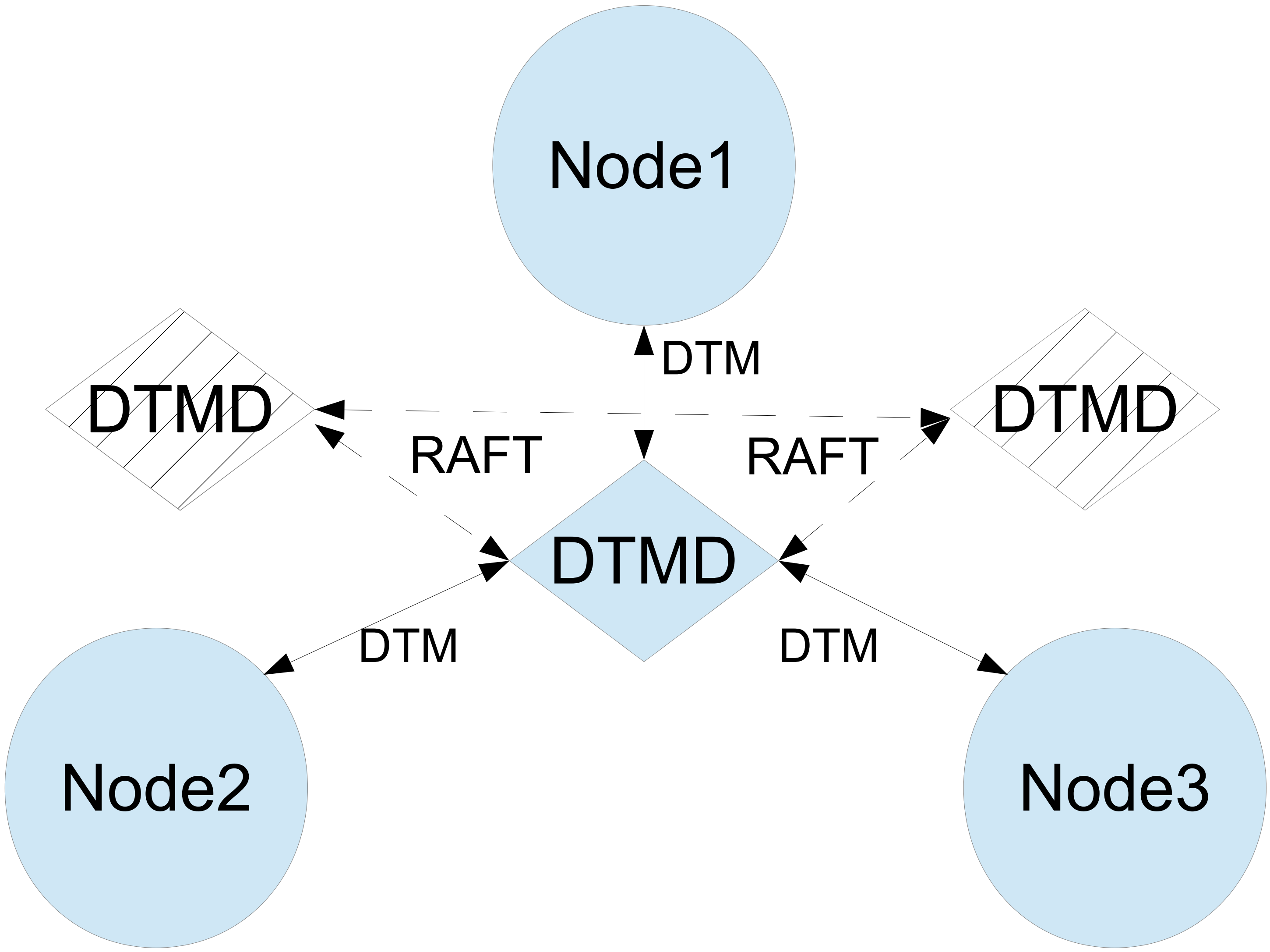
Multimaster based on logical replication



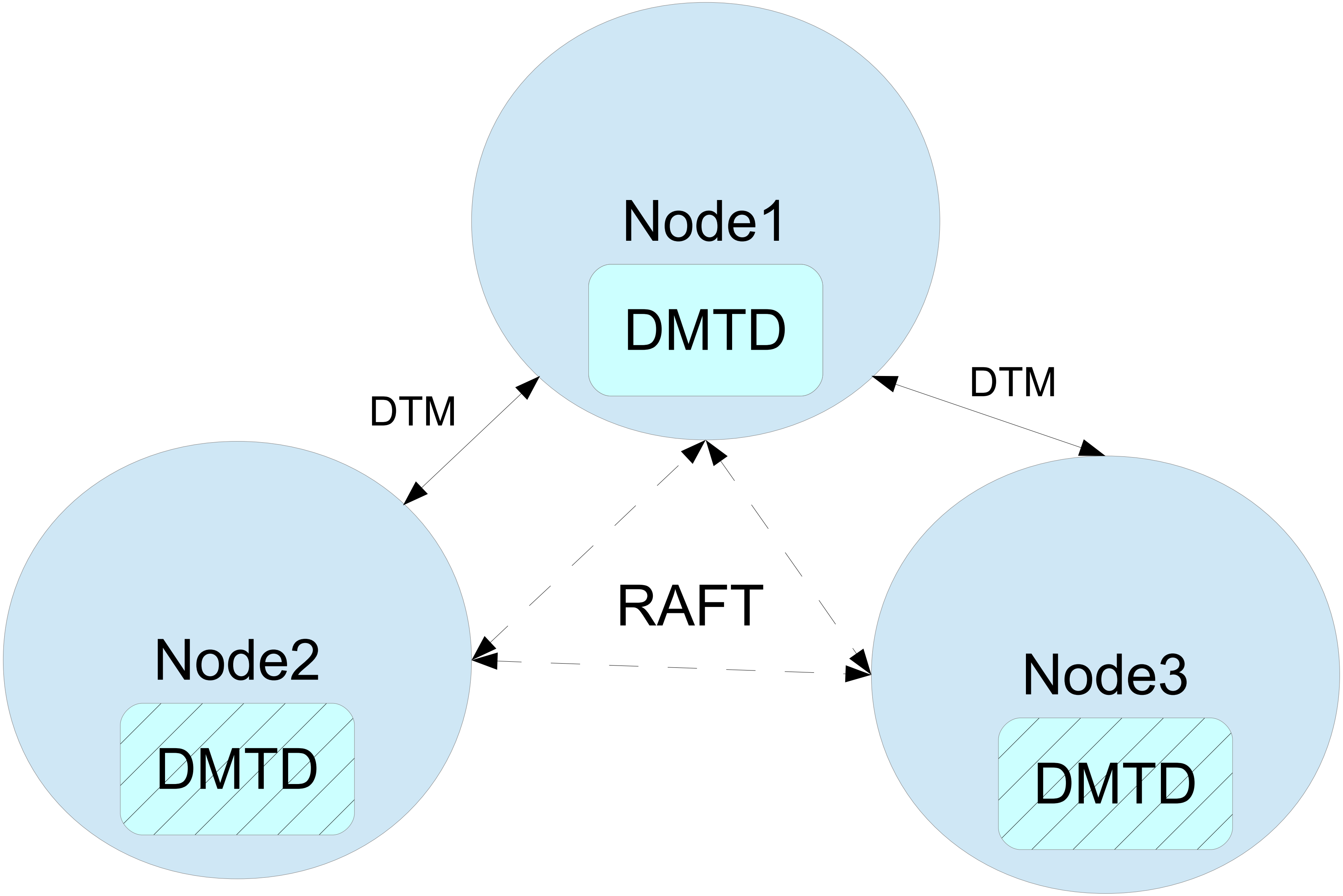
Logical repliction slots



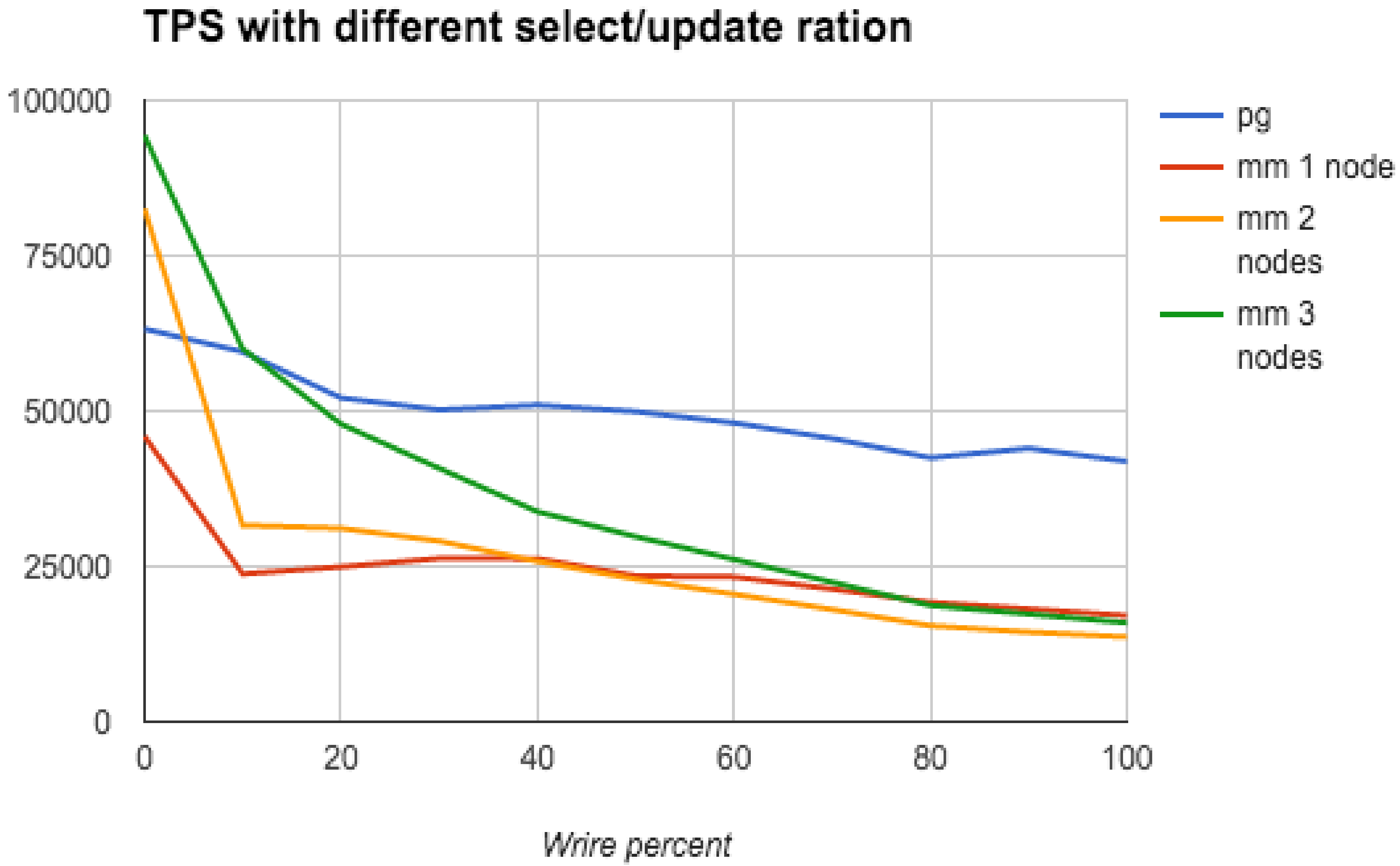
Current multimaster topology



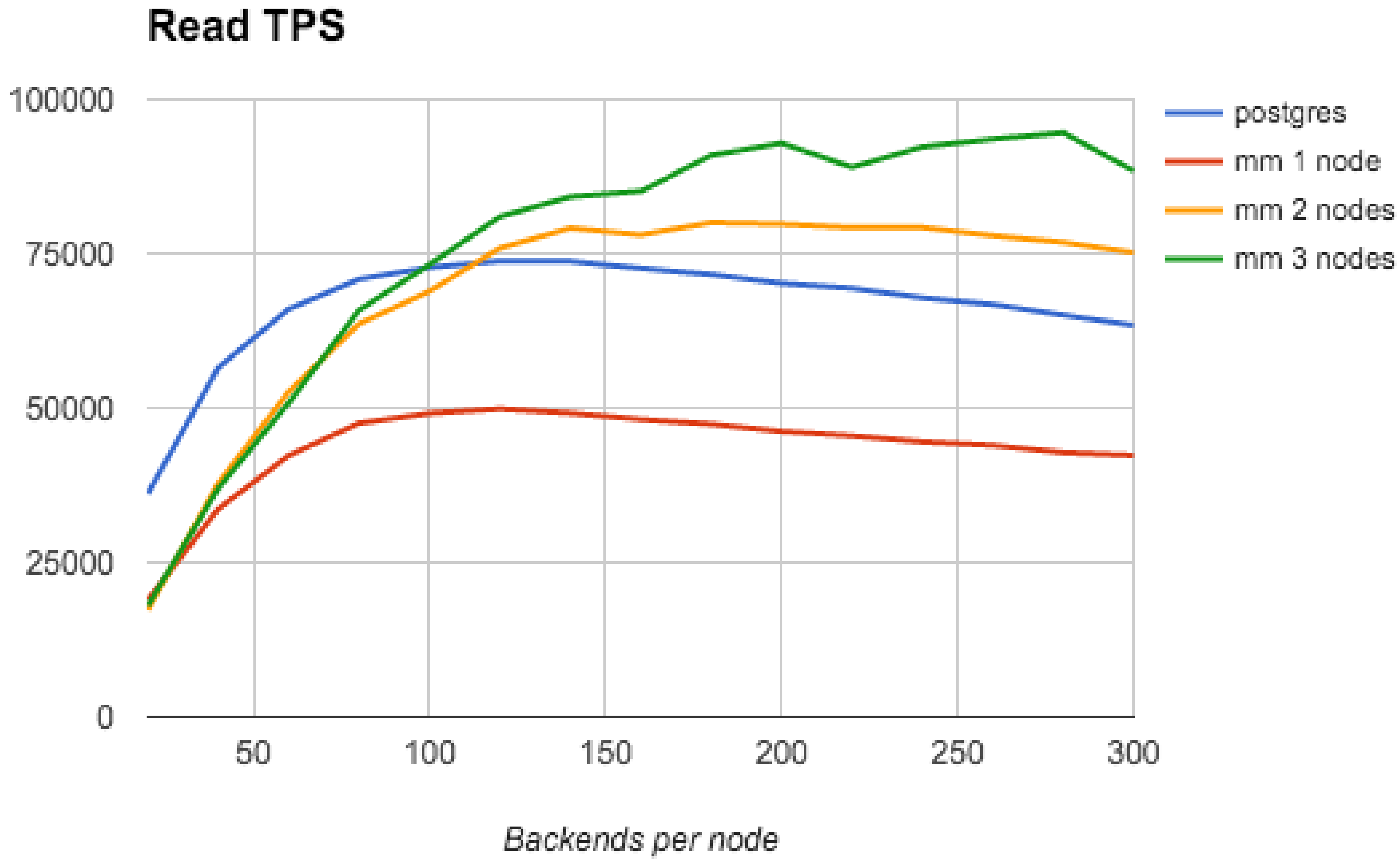
HA topology



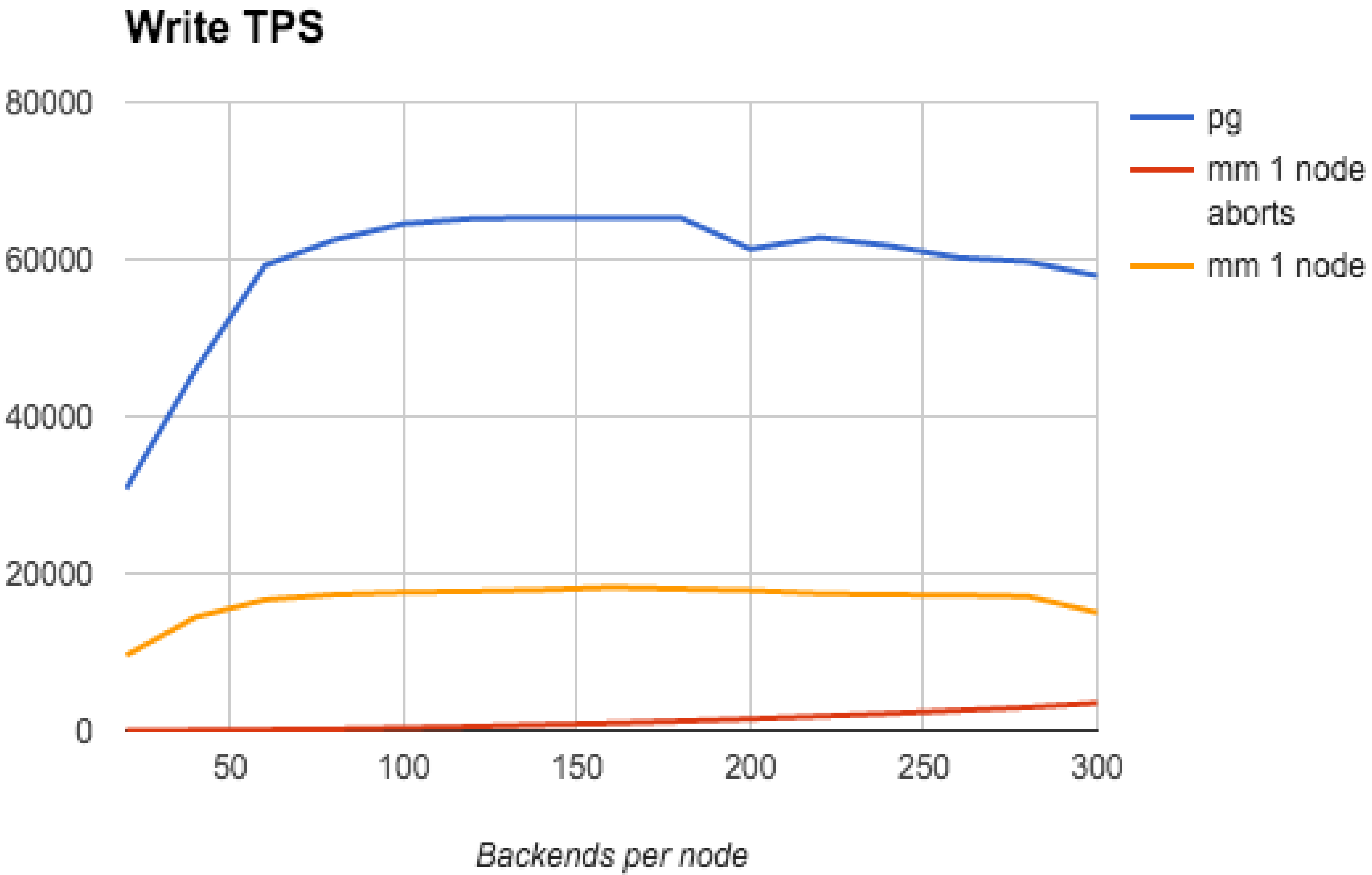
Multimaster performance



Multimaster read scaling

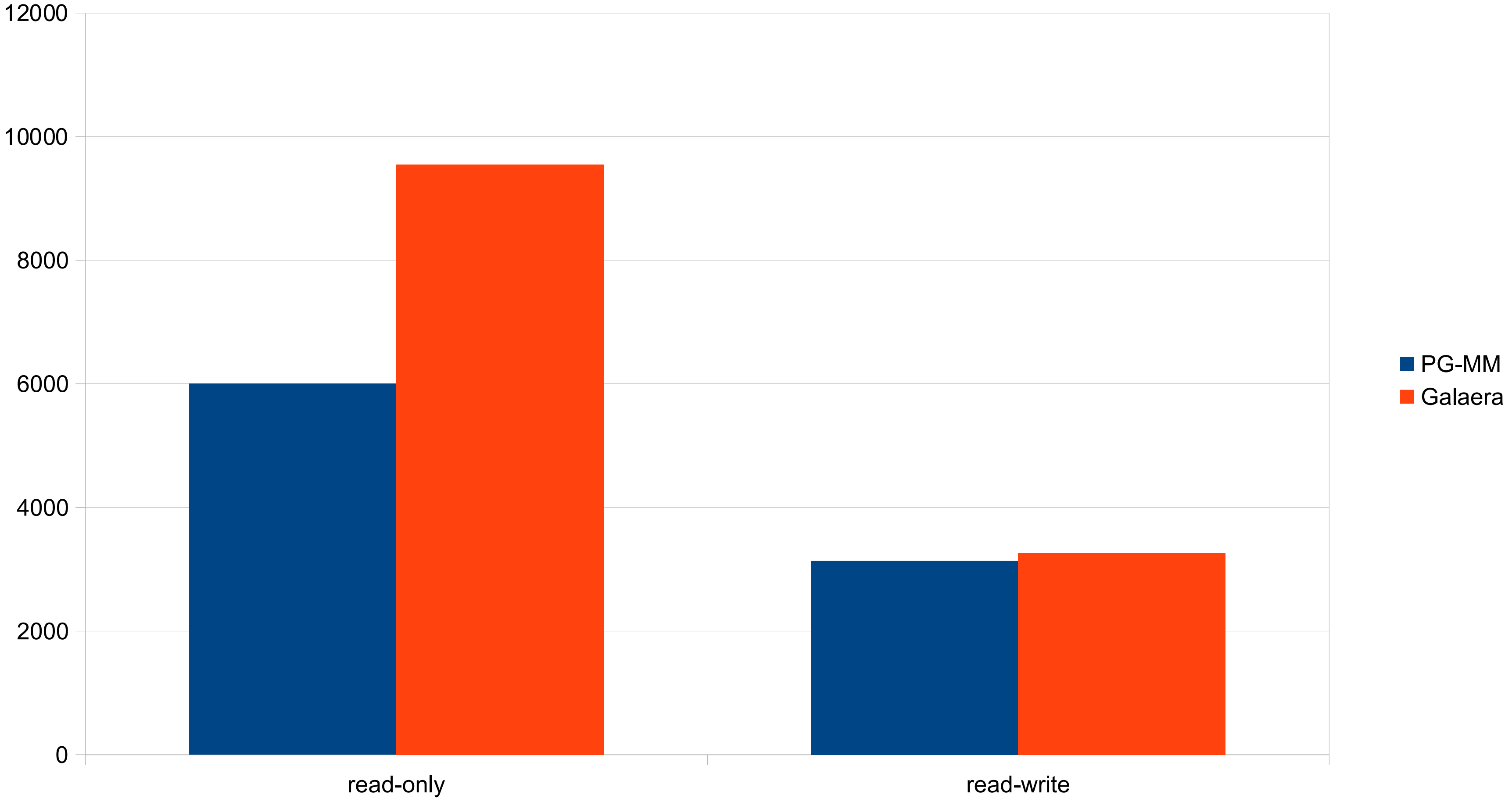


Multimaster writer “scaling”



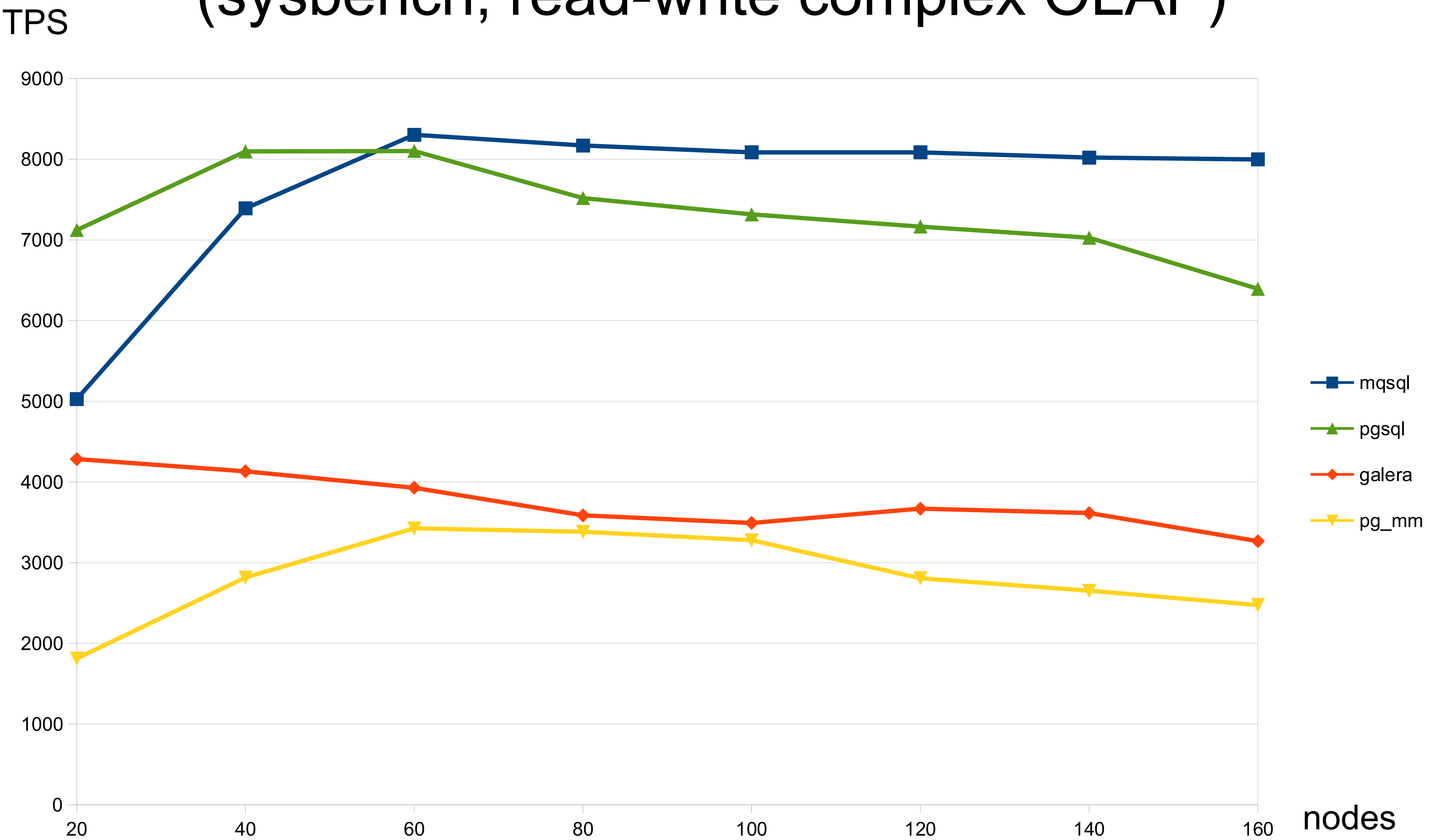
Galera vs. Postgres Multimaster

(sysbench OLAP-complex, 10M records, 3 nodes)

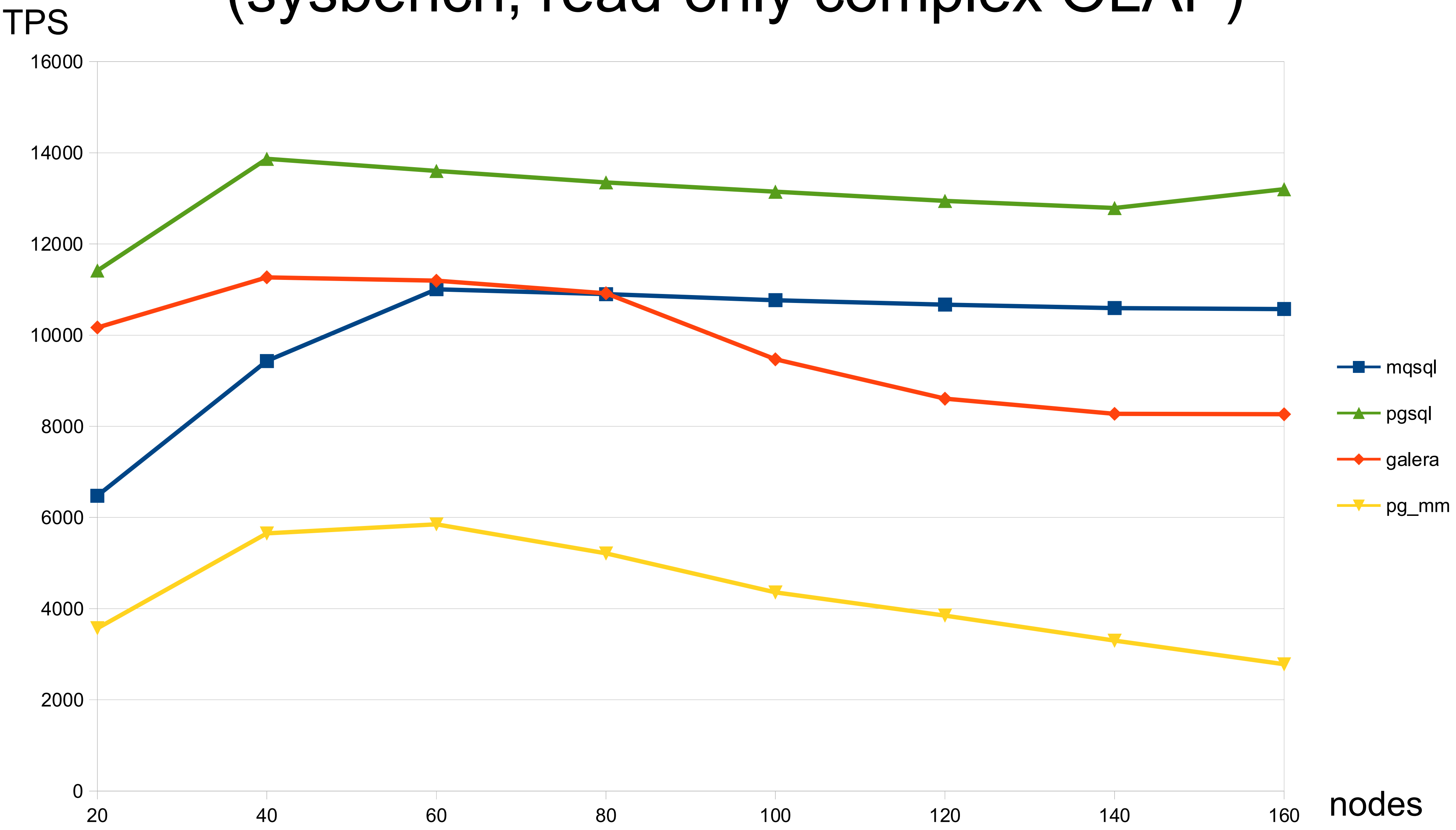


Galera vs. PostgreSQL multimaster

(sysbench, read-write complex OLAP)



Galera vs PostgreSQL multimaster
(sysbench, read-only complex OLAP)



Current multimaster limitations

- Table should have primary key.
- DDL is not handled by logical replication and requires separate replication channel which currently is not using 2PC.
- Subtransactions are not supported (limitation of DTM).
- Explicit locks are not distributed.
- Number of concurrent transactions is limited by number of BG workers

Questions?

https://github.com/postgrespro/postgres_cluster.git