Tungsten Cluster Master Class

Advanced: Maximize Performance With Parallel Apply

Chris Parker

VP of Customer Success, EMEA

Topics

In this short course, we will discuss:

- What parallel apply means and how it works
- Key benefits and use cases
- Important limitations and best practices
- Configuration and tuning options
- How to monitor and verify performance

The Challenge: Single Threaded Apply

- By default, replication applies transactions serially
- This can become a bottleneck for large or highly concurrent workloads
- Particularly limiting for data warehousing, batch loads, and multi-table update

What is Parallel Apply

- Splits transaction streams into multiple apply channels
- Transactions are split based on Schema
- Each channel can process transactions concurrently where safe to do so
- Goal: increase throughput while maintaining data consistency

How Parallel Apply Works

- Tungsten Replicator divides events into parallel apply channels
- Each channel applies events independently, partitioned by schema
- Transactions that touch multiple schemas are serialized in a single channel to maintain ordering
- Overall goal: increase throughput while ensuring consistent results

When Parallel Apply Works Best

- ROW Based Logging at Source
- Multiple Schemas used in Source database
- Transactions do NOT span across multiple schemas
- Sufficient OS resources:
  - Multi-Core
  - Free memory in OS Page Cache

When NOT to Use Parallel Apply

- Single Schema, or one “hot” schema and many “idle” schemas
- Small or OLTP-style transactions that intermix tables
- Systems sensitive to commit ordering
- Difficult to debug replication lag per channel

Configuring Parallel Apply

- Enable by adding the following two parameters to your tungsten.ini:
  - `svc-parallelization-type=disk`
  - `channels=nn`
- nn = Number of channels to configure
  - 1 – Default – Disables parallel apply
  - Start small and test
- **IMPORTANT:** Always take replicator offline cleanly after enabling

Deployment and Tuning

- Start with 2–4 channels and benchmark
- Gradually increase channels, monitor latency
- Avoid over-parallelization on small datasets
  - Avoid channels greater than total number of schemas
- Match to target database capacity and I/O throughput
- Check `serializationCount`

What is serializationCount?

- Number of transactions that cannot be applied in parallel
- Check using `trepctl status -name stores`
- >2% = may be losing benefit of parallel

```
shell> trepctl status -name stores
Processing status command (stores)...
...
NAME VALUE
---- -----
criticalPartition : -1
discardCount : 0
estimatedOfflineInterval: 0.0
eventCount : 1512
headSeqno : 78022
maxOfflineInterval : 5
maxSize : 10
name : parallel-queue
queues : 5
serializationCount : 26
serialized : false
```

(1512 events processed / 26 serialized) = 1.7%

Monitoring

- `trepctl status -name shards`
- `trepctl status -name tasks`
- `trepctl perf`
- Monitor:
  - **Latency** per channel
  - Compare before/after enabling parallel apply

Summary

What we learnt in this course:

- What *parallel apply* means and how it works
- Key benefits and use cases
- Important limitations and best practices
- Configuration and tuning options
- How to monitor and verify performance

Thank you for listening

continuent.com

Chris Parker

VP of Customer Success, EMEA