Reactors: A Case for Predictable, Virtualized Actor Database Systems

Vivek Shah, Marcos Antonio Vaz Salles
University of Copenhagen (DMS GROUP @ DIKU)

Motivation

Is the programming interface of stored procedures good enough for modern OLTP databases?

- Performance Challenges
  - Local and intra-transaction parallelism
  - Low-level deployment control

- Software Engineering Challenges
  - Modularity and isolation
  - Abstractions to reason about performance

How can we integrate actor programming models in modern relational databases to improve programmability of stored procedures?

Relational Actor (Reactor) Programming Model

- A reactor is an application-defined actor encapsulating the state as relations
- Declarative queries are used to access the encapsulated reactor state
- Communication across reactors only through asynchronous function calls
- Computations across reactors provide ACID guarantees

Reactors: A Case for Predictable, Virtualized Actor Database Systems

Vivek Shah, Marcos Antonio Vaz Salles
University of Copenhagen (DMS GROUP @ DIKU)

New OLTP Application Trends

Increasing application logic complexity
- Latency is critical
- Scalability matters

Relational Actor (Reactor) Programming Model

- A reactor is an application-defined actor encapsulating the state as relations
- Declarative queries are used to access the encapsulated reactor state
- Communication across reactors only through asynchronous function calls
- Computations across reactors provide ACID guarantees

Reactors: A Case for Predictable, Virtualized Actor Database Systems

Vivek Shah, Marcos Antonio Vaz Salles
University of Copenhagen (DMS GROUP @ DIKU)

Modular, Concurrent and Asynchronous

Relational Actor (Reactor) Programming Model

- A reactor is an application-defined actor encapsulating the state as relations
- Declarative queries are used to access the encapsulated reactor state
- Communication across reactors only through asynchronous function calls
- Computations across reactors provide ACID guarantees

Reactors: A Case for Predictable, Virtualized Actor Database Systems

Vivek Shah, Marcos Antonio Vaz Salles
University of Copenhagen (DMS GROUP @ DIKU)

A Simplified Digital Currency Exchange Application using Reactors

Design and Implementation of ReactDB

Run-time Deployments

Multi-core machine

- Container
  - Transaction Coordinator
  - Transport Driver
  - Transaction Executor
  - Request Queue

- Shared-nothing

- Shared-everything-with-affinity

- Shared-everything-without-affinity

- Single transaction execution, sequential execution to leverage shared memory

- Multi-Container
  - OCC + 2PC protocol
  - Multiple transaction contexts, asynchronous execution across containers

Implementation Overview

Thread Management in Transaction Executors:
- Thread pool uses cooperative multitasking
- Configurable multi-programming level

Storage Layer:
- Primary Masstree indices

Concurrency Control:
- Single Container
  - OCC protocol (Silo, Tu et al. [SOSP 2013])
  - Single transaction context, sequential execution to leverage shared memory
- Multi-Container
  - OCC + 2PC protocol
  - Multiple transaction contexts, asynchronous execution across containers

Multi-Unit TPC-C with 30,000 users

- 0.5x TPC-C new-order + 1.5x change order + 2.5x payment + 3.5x settlement

- 100% TPC-C new-order + artificial delay of 300-400 ms per transaction

- Varying complexity of programs

- Asynchronicity gains diminish with increasing complexity of application logic

- Asynchronicity gains manifest with increasing load of procedures

- Asynchronicity gains diminish with increasing load of procedures

Experimental Evaluation

Machine

- 2x AMD Opteron 6274 with 8 cores @ 2.1 GHz
- 4x 32 GB DDR3 RAM
- 64 KB L1, 2 MB L2, 6 MB L3 cache

Memory access affinity matters for sequential execution of classic OLTP workloads

- Standard TPC-C benchmark min
- Each warehouse is a reactor
- Final scale factor of with varying load on database as a whole (workers > 3 stimulus overlaid on the database)