Contents

Chapter 3

Maintaining State

A concurrent program that uses mutable speculative and shared state is not scalable. Transactional Memory systems buffer speculative and shared state at the expense of increased memory bandwidth which limits scalability. This chapter examines how immutable data offers a means of maintaining both speculative and shared state that permits a concurrent program to scale.

Section 3.1 identifies the choice of where to store speculative state as one of the central design decisions of a Transactional Memory system. The section reviews the mechanisms that Transactional Memory systems employ to support shared and speculative state and it proposes an alternative approach in which both speculative and shared state are stored immutably.

The remainder of the chapter focuses on an implementation of an Immutable Data Structure suitable for use in a concurrent execution environment.

Section 3.2 describes how Immutable Data Structures can be used to store speculative state.

Section 3.3 describes techniques for implementing Immutable Data Structures.

Section 3.4 describes an Immutable Data Structure that we call the Canonical Binary Tree.

Section 3.5 describes how the Canonical Binary Tree can be specialised to implement common ADTs.

Section 3.6 describes how the Canonical Binary Tree can be balanced to minimise access time.

Bibliography