Consensus Number

魏恒峰

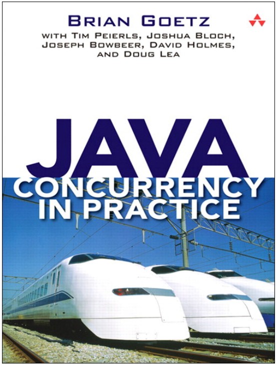
hfwei@nju.edu.cn

2017 年 12 月 14 日

Feel Free to
Ask Questions

Are you Familar with **Concurrent Programming**?

The Key: **Synchronization**!


BRIAN GOETZ
WITH TIM PEIERLS, JOSHUA BLOCH, JOSEPH BOWBEER, DAVID HOLMES, AND DOUG LEA
JAVA
CONCURRENCY IN PRACTICE

Using the **Synchronization Primitives**
Provided by Your Favorite Languages.

synchronized
Semaphore

BlockingQueue
ConcurrentMap

Phaser
Barrier

synchronized
Semaphore
BlockingQueue
ConcurrentMap
Phaser
Barrier

```
class AtomicInteger:
  get()
  set(int newValue)

  getAndIncrement()
  getAndDecrement()
  getAndSet(int newValue)

  compareAndSet(int expectedValue, int newValue)
compareAndSet(int expectedValue, int newValue)
compareAndSwap(int expectedValue, int newValue)
```

CAS — CMPXCHG

`impl.`

`usage`

Consensus

"It looks like we have a consensus."

Propose

Decide

Propose

Decide

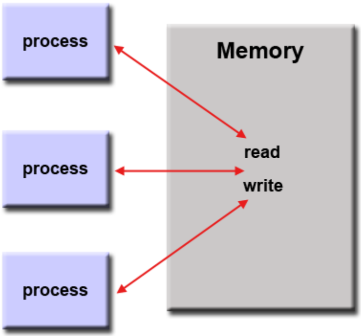
Definition (The Consensus Problem)

Agreement All (non-faulty) processes must **agreen on the same value**.

Validity The common decision value must be the value **proposed** by some process.

Termination Each (non-faulty) process must **eventually** decide on a value.

More clarification on “termination”
binary consensus problem



(redraw)

consensus object (fig here)

```
    public interface Consensus<T> {
      T decide(T value);
    }
```

consensus protocol

```
public abstract class ConsensusProtocol<T>
                    implements Consensus<T> {
  protected T[] proposed = (T[]) new Object[N];

  void propose(T value) {
    proposed[ThreadID.get()] = value;
  }

  public abstract T decide(T value);
}
```

implement X using Y

```java
public class CASConsensus<T>
     extends ConsensusProtocol<T> {
  private final int FIRST = -1;
  private AtomicInteger r = new AtomicInteger(FIRST);

  @Override
  public T decide(T value) {
    propose(value);

    int i = ThreadID.get();
    if (r.compareAndSet(FIRST, i))  // I won
      return proposed[i];
    else                            // I lose
      return proposed[r.get()];
  }
}
```

Theorem (Computational Power of CAS)

A register providing compareAndSet() and get() methods can solve the consensus problem for any number of threads.

Theorem (Computational Power of CAS)

A register providing compareAndSet() and get() methods can solve the consensus problem for any number of threads.

In terms of “Consensus Number”:

Theorem (Consensus Number of CAS)

A register providing compareAndSet() and get() methods has infinite **consensus number***.*

Definition (Consensus Number)

The **consensus number** for X is the largest n for which X solves n-thread consensus.

If no largest n exists, the consensus number is said to be infinite.

Lemma (Y Implements X)

Theorem (Consensus Number as . . .)

in the following, main results (table)

beautiful ideas and proofs

Thank You!