```scala
trait LatencyBound {
  // execute readOp with strongest consistency possible
  // within the latency bound
  def rush[T](bound: Duration,
              readOp: ConsistencyLevel => T): Rushed[T]
}

/* Generic reservaton pool, conceptually one per
 * ADT instance. `max` recomputed as needed
 * (e.g. for percent error) */
abstract class ReservationPool(max: () => Int) {
  def take(n: Int): Boolean // try to take tokens
  def sync(): Unit      // sync to regain used tokens
  def delta(): Int      // # possible ops outstanding
}

/* Counter with ErrorBound (simplified) */
class Counter(key: UUID) with ErrorBound {
  def error: Float // error bound
  def computeMax(): Int = (cass.read(key) * error).toInt

  val incrPool = ReservationPool(computeMax)
  val decrPool = ReservationPool(computeMax)

  def value(): Interval[Int] = {
    val v = cass.read(key)
    Interval(v - decrPool.delta,
             v + incrPool.delta)
  }

  def incr(n: Int): Unit = {
    waitFor(incrPool.take(n)) {
      cass.incr(key, n)
    }
  }
}
```