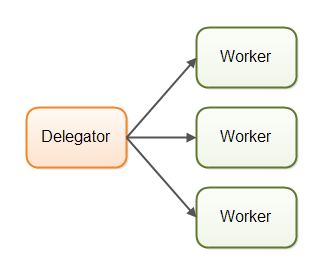
**Concurrency Models :**

The concurrency models described in this text are similar to different architectures used in distributed systems

In a concurrent system different threads communicate with each other. In a distributed system different processes communicate with each other (possibly on different computers).

**1.Parallel Workers**



Many of the concurrency utilities in **the [java.util.concurrent Java package](http://tutorials.jenkov.com/java-util-concurrent/index.html)** are designed for use with this model. You can also see traces of this model in the design of the **Java Enterprise Edition application servers.**

**Parallel Workers Advantages**

easy to understand. To increase the parallelization of the application you just add more workers.

**Parallel Workers Disadvantages**

### Shared State Can Get Complex

[race conditions](http://tutorials.jenkov.com/java-concurrency/race-conditions-and-critical-sections.html), [deadlock](http://tutorials.jenkov.com/java-concurrency/deadlock.html)

**Modern**[**non-blocking concurrency algorithms**](http://tutorials.jenkov.com/java-concurrency/non-blocking-algorithms.html) may decrease contention and increase performance, but non-blocking algorithms are hard to implement.

**Persistent data structures** are another alternative. A persistent data structure always preserves the previous version of itself when modified.

The Scala programming contains several persistent data structures.

### Stateless Workers : performance hit in reading latest state.

### Job Ordering is Nondeterministic

## 2. Assembly Line ??????

# 3. Same-threading

## A same-threaded system.