Multithread Programlama

Actor Modeli

Akka

Selim Öber

İstanbul Coders

istanbul
CODERS

Mesleğinde daha iyiyi arayan bir grup insan

Keşfet

İstanbul Coders

- **Site:** http://istanbulcoders.org/
- **Twitter:** @istanbulcoders
- **Mail grup:** https://groups.google.com/forum/#!forum/istanbul-coders
- **Iletisim:** http://istanbulcoders.org/contact/

İstanbul Coders - Takvim

Etkinlik Takvimi

Istanbul Coders

Today | **Saturday, October 26, 2013** | Print | Week | Month | **Agenda**

Showing events after 26/10. *Look for earlier events*

Thursday, December 12, 2013

19:00 Spyne Internals

Thursday, January 2

19:00 Marmara Drone - Açık Kaynak İnsansız Hava Aracı Kontrol Sistemi

Thursday, January 9

19:00 Yazılım Geliştirme Sürecinde Teşhis ve Tedavi. Scrum, Scrum BuTT, Scrum And

Thursday, January 16

19:00 Multithread programlama ve Aktor modeli

Thursday, January 23

19:00 Asp.Net MVC

Thursday, January 30

19:00 Cassandra ve Storm

Showing events until 31/1. *Look for more*

http://istanbulcoders.org/calendar/

Gundem

- Multithreaded Programlama **10-15**
- Aktor modeli **1-2**
- Akka **5-10**

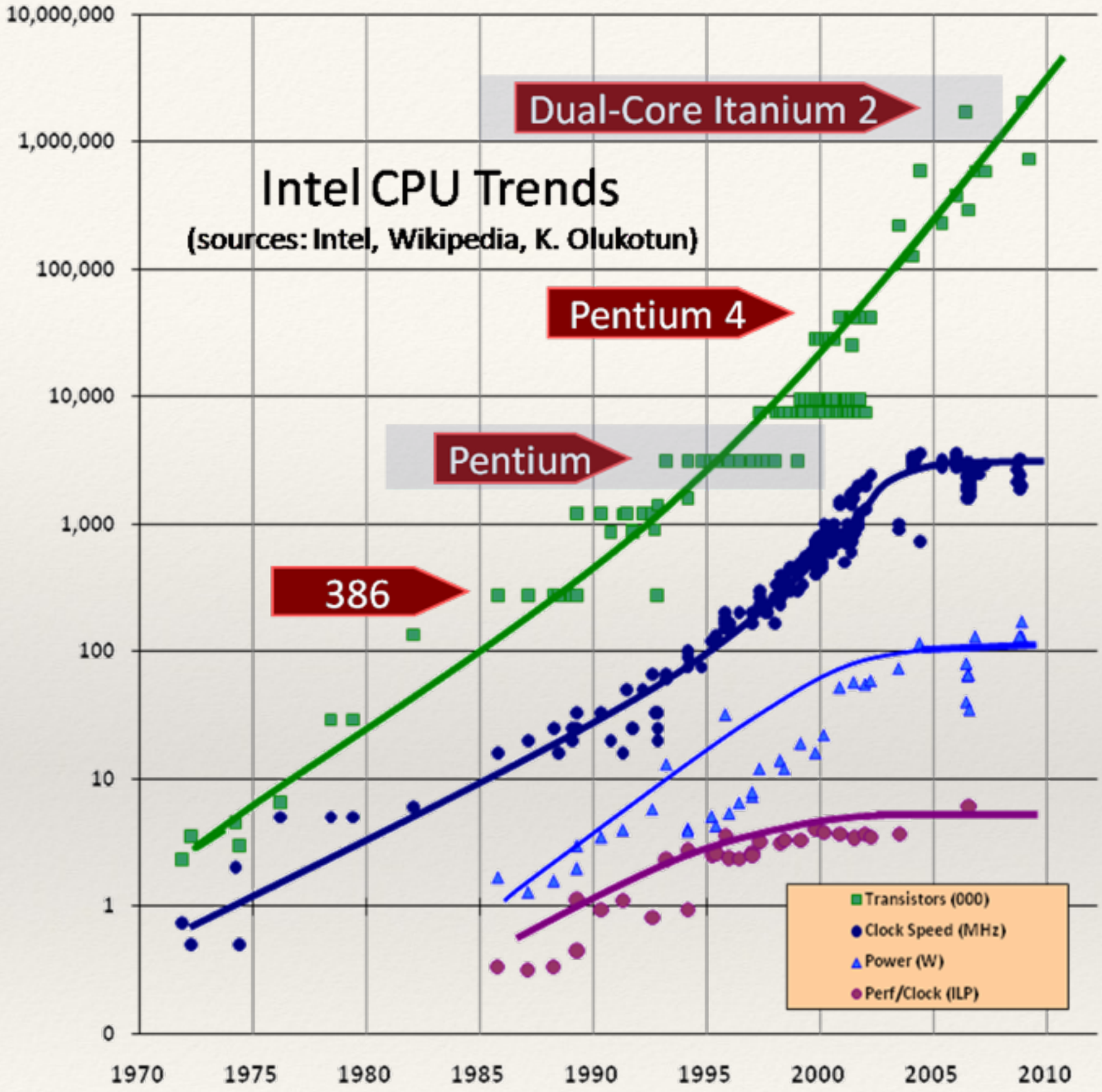
Multithread Programlama

Mecbur muyum?

"Free lunch is over."

–Herb Sutter, 2005

http://www.gotw.ca/publications/concurrency-ddj.htm



http://www.gotw.ca/publications/concurrency-ddj.htm

Concurrency

koşut zamanlı

eşzamanlı

müşterek

vs

Parallism

paralel

Concurrency ⊃ Parallism

Problem?

- Zor
  - Teknolojik zorluklar
  - Bilissel (cognitive) zorluklar

Problem?

- Teknolojik zorluklar
  - Thread kullanimini optimize etmek icin asynchronous API'ler gerekiyor.
  - Hata yonetimi icin geleneksel yontemler olan Exception firlatma ve hata kodu dondurme asynchronous sistemlerde calismiyor.
- Data'ya erisimi kontrol etmek surekli dikkat isteyen bir is. Compiler sizi korumuyor.
- Kilitleri (deadlocks) ve yaris durumlarini (race conditions) yonetmek icin sinyalizasyon mekanizmalarini kullanmak gerekiyor.
- Teknik borca yatkin programlar olusturuluyor.
- Test, Heisenbug mi dediniz?

Problem?

- Zekasal problemler
  - Unutuyoruz.
  - Sirali calisan (sequential) karsitlarina gore gorsellestirmesi zor.
- Dar bogazlari (bottlenecks) minimize eden veri akislari saglamak guc.
- Hata durumunda ne olacagini tasarlamak devreleri yakabiliyor.

Bilmemiz Gerekenler

happens-before

visibility

Wait-freedom

Threads

Lock-freedom

immutability

atomicity

reentrant locks

Deadlocks

thread safety

volatile

monitor

Race Conditions

Starvation

Executor

thread confinement

synchronized

Livelocks

Amdahl's law

Obstruction-freedom

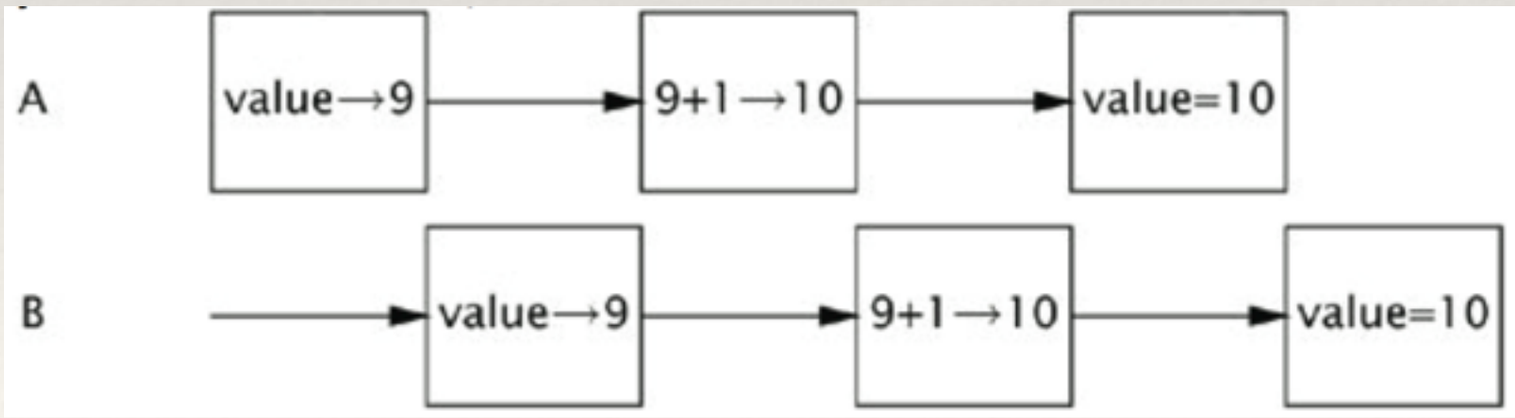
Optimistic concurrency control

Yaygin Modeller

- Synchronization / Sharing state
- Software Transactional Memory (STM)
- Functional Programming (Stateless | Immutable)
- Actor

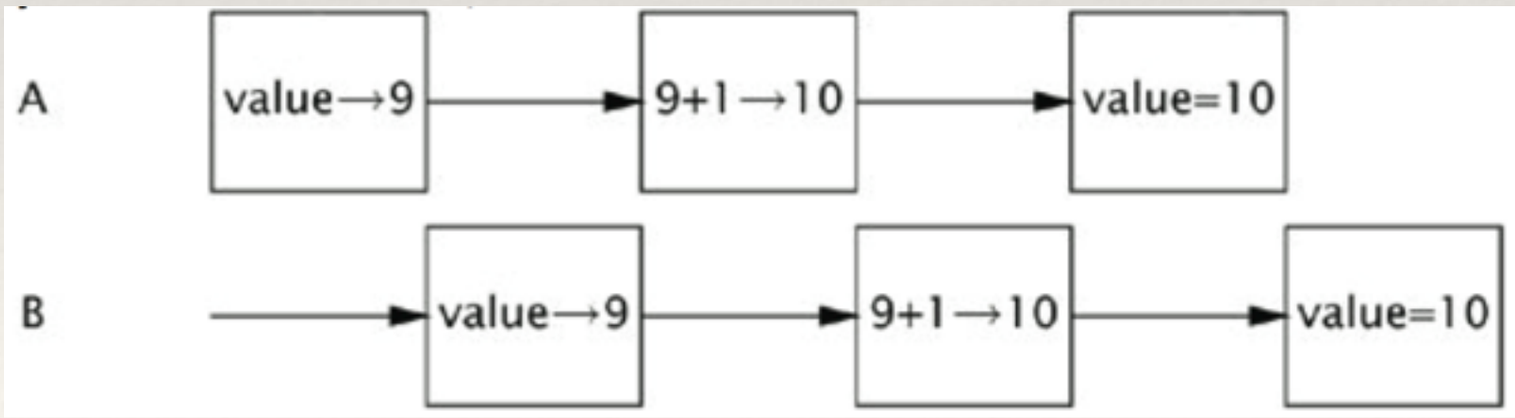
Synchronization / Sharing state

```
@NotThreadSafe
public class UnsafeSequence {
 private int value;

 /** Returns a unique value. */
 public int getNext() {
    return value++;
 }
}
```



Synchronization / Sharing state

```
@ThreadSafe
public class UnsafeSequence {
 private int value;

 /** Returns a unique value. */
 public synchronized int getNext() {
    return value++;
 }
}
```



Synchronization / Sharing state

```
@NotThreadSafe
public class LazyInitRace {
    private ExpensiveObject instance = null;

    public ExpensiveObject getInstance() {
        if (instance == null)
          instance = new ExpensiveObject();
        return instance;
    }
}
```

Synchronization / Sharing state

```
public class NoVisibility {
    private static boolean ready;
    private static int number;
    private static class ReaderThread extends Thread {
        public void run() {
            while (!ready)
                Thread.yield();
            System.out.println(number);
        }
    }

    public static void main(String[] args) {
        new ReaderThread().start();
        number = 42;
        ready = true;
    }
}
```

!

```
volatile boolean asleep;
...
    while (!asleep)
        countSomeSheep();
```

Software Transactional Memory (STM)

?

Functional Programming

- Programlamayi matematiksel fonksiyonlarin calistirilmasi olarak ele alir.
- Yan etkisi (side effects) yoktur.
- State'i (turkce?) olmayan veya degistirilemeyen (immutable) nesneleri kullanir.

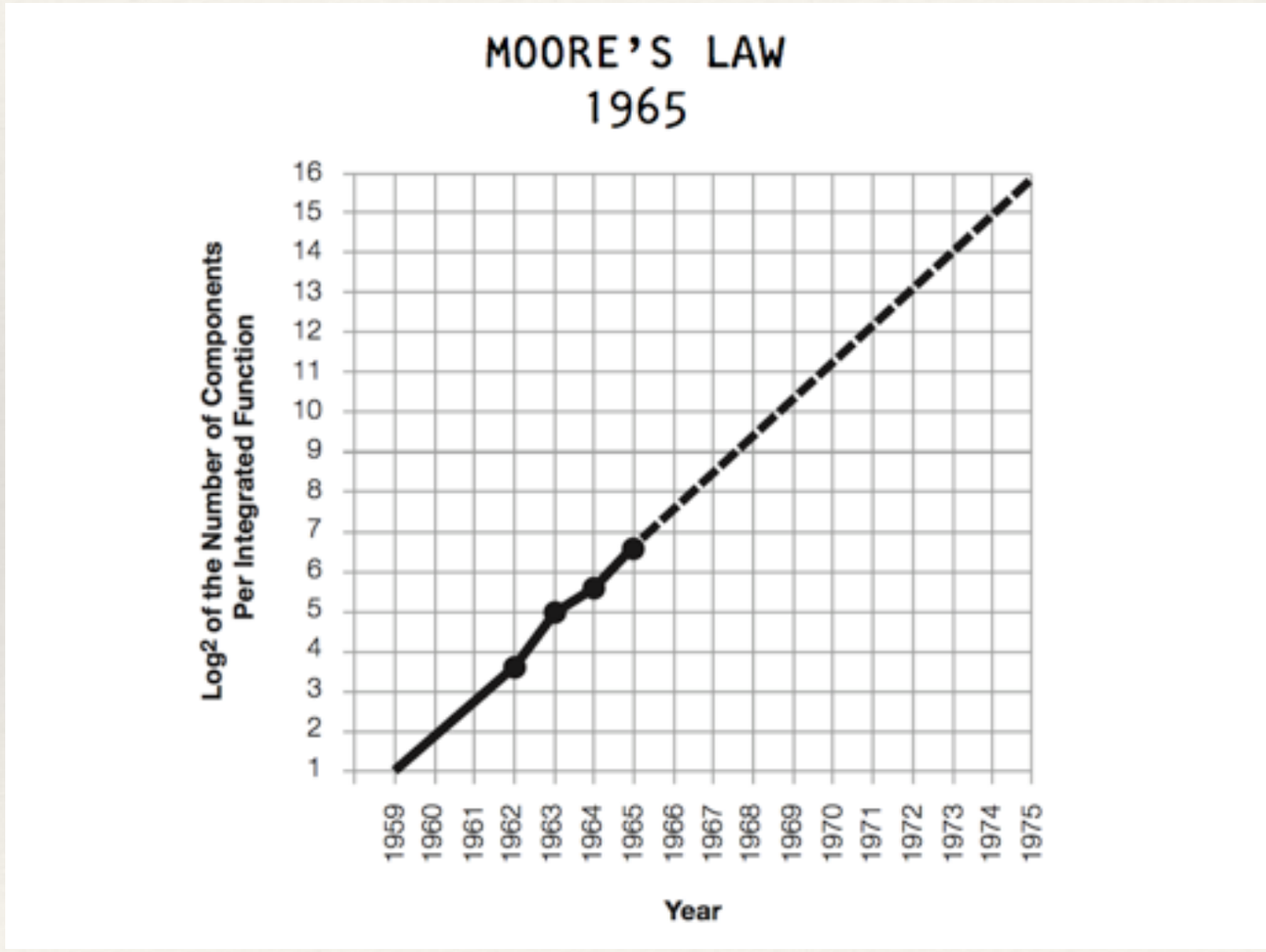
Aktor Modeli

Actor Modeli

- Hepimiz Aktoruz!
- Bir paradigma (!!)
- Carl Hewitt; Peter Bishop; Richard Steiger (1973). A Universal Modular Actor Formalism for Artificial Intelligence. IJCAI
- Erlang
- Parantez (

Bret Victor
"The Future of Programming"

Bret Victor
"The Future of Programming"



Bret Victor
"The Future of Programming"

PATTERN MATCHING

Ralph Grisworld
SNOBOL
1962

Ken Thompson
REGULAR EXPRESSIONS
1967

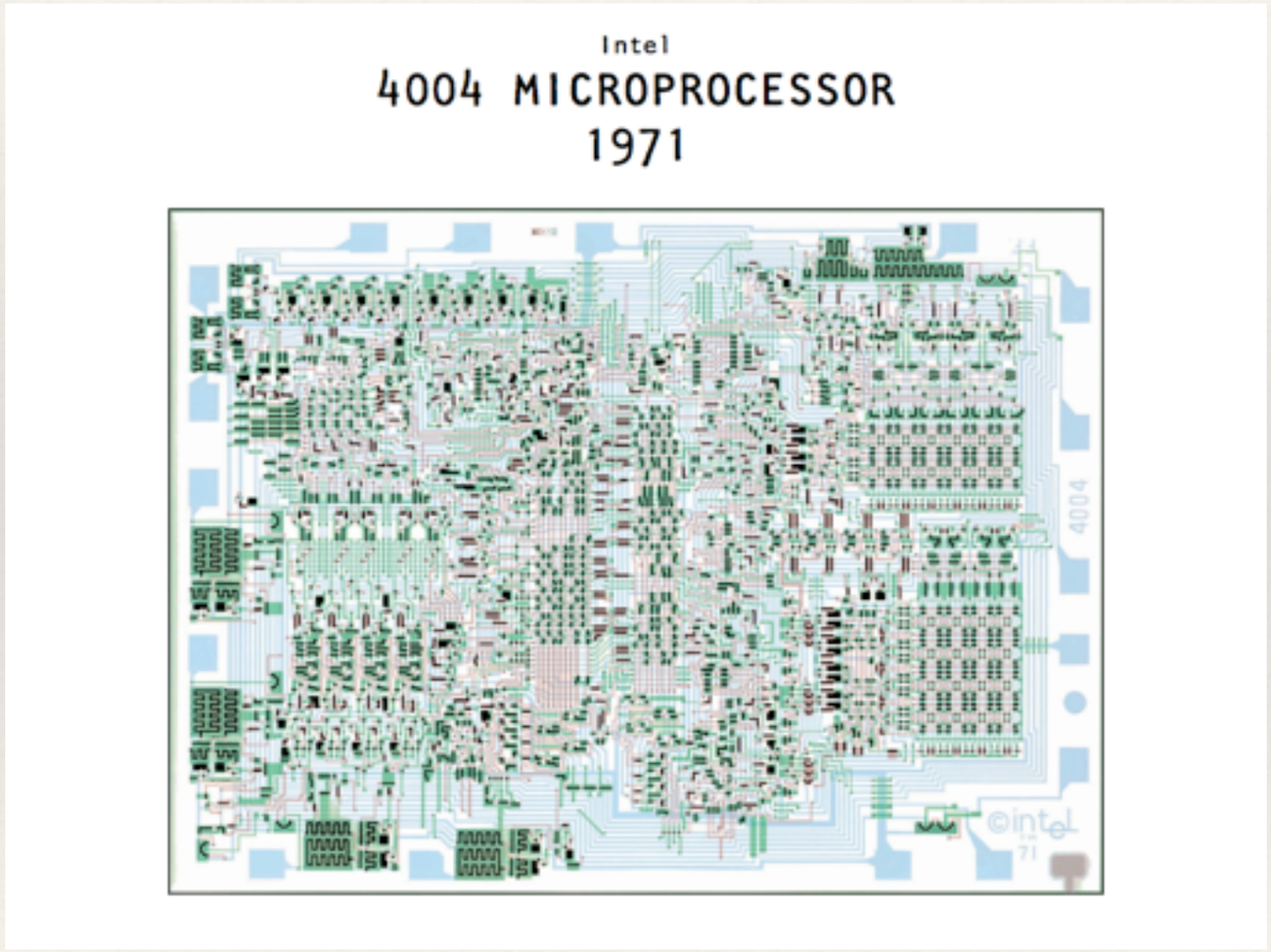
Bret Victor
"The Future of Programming"

Doug Engelbart

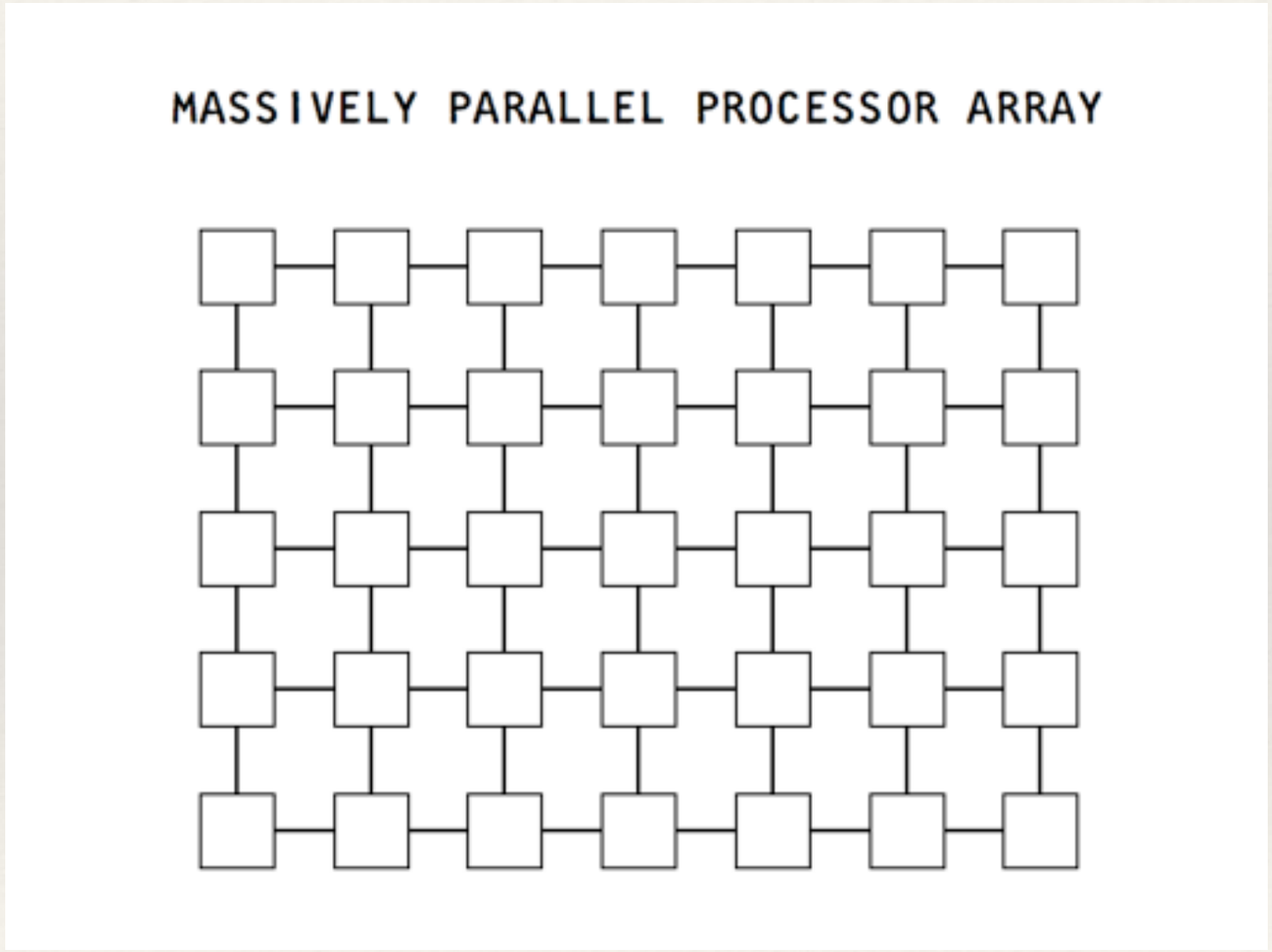
NLS

1968

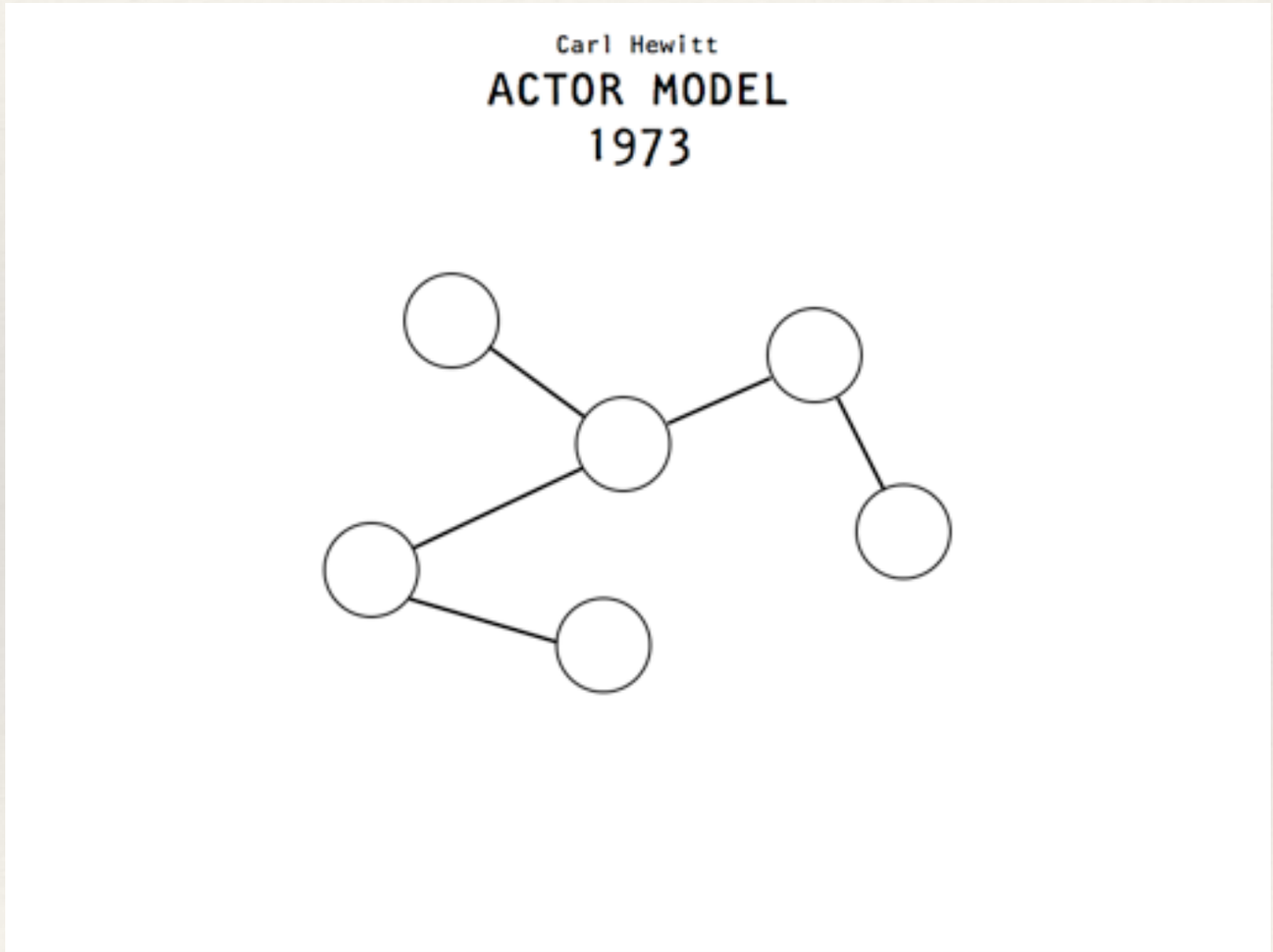
Bret Victor
"The Future of Programming"



Bret Victor
"The Future of Programming"



Bret Victor
"The Future of Programming"



)

Actor Modeli

- Bir aktor:
  - Diger aktorlerden mesaj kabul eder.
  - Diger aktorlere mesaj gonderir.
  - Baska aktorler yaratir.
  - Bir sonraki gelecek mesaj icin davranisini belirler.
- Yukaridaki davranislar icin onceden belirlenmis bir sira yoktur. Herhangi bir sirada olabilirler.

Actor Modeli

- Sevmeyeni de var:
  - http://noelwelsh.com/programming/2013/03/04/why-i-dont-like-akka-actors/
- Ozet olarak:
  - Kaba Soyutlama (Coarse Abstractions)
  - Birlesmiyorlar (do not compose)
  - Type bilgisi yetersiz.

Actor Modeli

Name	Status	Latest release	License	Languages
Actor	Active	2013-05-30	MIT	Java
Actor Framework	Active	2013-11-13	Apache 2.0	.NET
Akka	Active	2013-07-09	Apache 2.0	Java and Scala
Ateji PX	Active	?	?	Java
F# MailboxProcessor	Active	same as F# (built-in core library)	Apache License	F#
Korus	Active	2010-02-01	GNU GPL 3	Java
Kilim [38]	Active	2011-10-13[39]	MIT	Java
ActorFoundry (based on Kilim)	Active?	2008-12-28	?	Java
ActorKit	Active	2011-09-14[40]	BSD	Objective-C
Cloud Haskell	Active	2012-11-07[41]	BSD	Haskell
NAct	Active	2012-02-28	LGPL 3.0	.NET
Retlang	Active?	2011-05-18[42]	New BSD	.NET
JActor	Active	2013-01-22	LGPL	Java
Jetlang	Active	2012-02-14 [43]	New BSD	Java
Haskell-Actor	Active?	2008	New BSD	Haskell
GPars	Active	2012-12-19	Apache 2.0	Groovy
PARLEY	Active?	2007-22-07	GNU GPL 2.1	Python
Pulsar	Active	2013-06-30	New BSD	Python
Pulsar	Active	2013-07-19 [44]	LGPL/Eclipse	Clojure
Pykka	Active	2012-12-12 [45]	Apache 2.0	Python
Termite Scheme	Active?	2009	LGPL	Scheme (Gambit implementation)
Theron	Active	2012-08-20[46]	MIT[47]	C++
Quasar	Active	2013-07-19 [48]	LGPL/Eclipse	Java
Libactor	Active?	2009	GPL 2.0	C
Actor-CPP	Active	2012-03-10[49]	GPL 2.0	C++
S4	Active	2011-11-28[50]	Apache 2.0	Java
libcppa	Active	2012-08-22[51]	LGPL 3.0	C++11
Celluloid	Active	2012-07-17[52]	MIT	Ruby
LabVIEW Actor Framework	Active	2012-03-01[53]	?	LabVIEW
QP frameworks for real-time embedded systems	Active	2012-09-07[54]	GPL 2.0 and commercial (dual licensing)	C and C++
libprocess	Active	2012-07-13	Apache 2.0	C++
SObjectizer	Active	2014-01-04	New BSD	C++

Typesafe

Akka

Akka is a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications on the JVM.

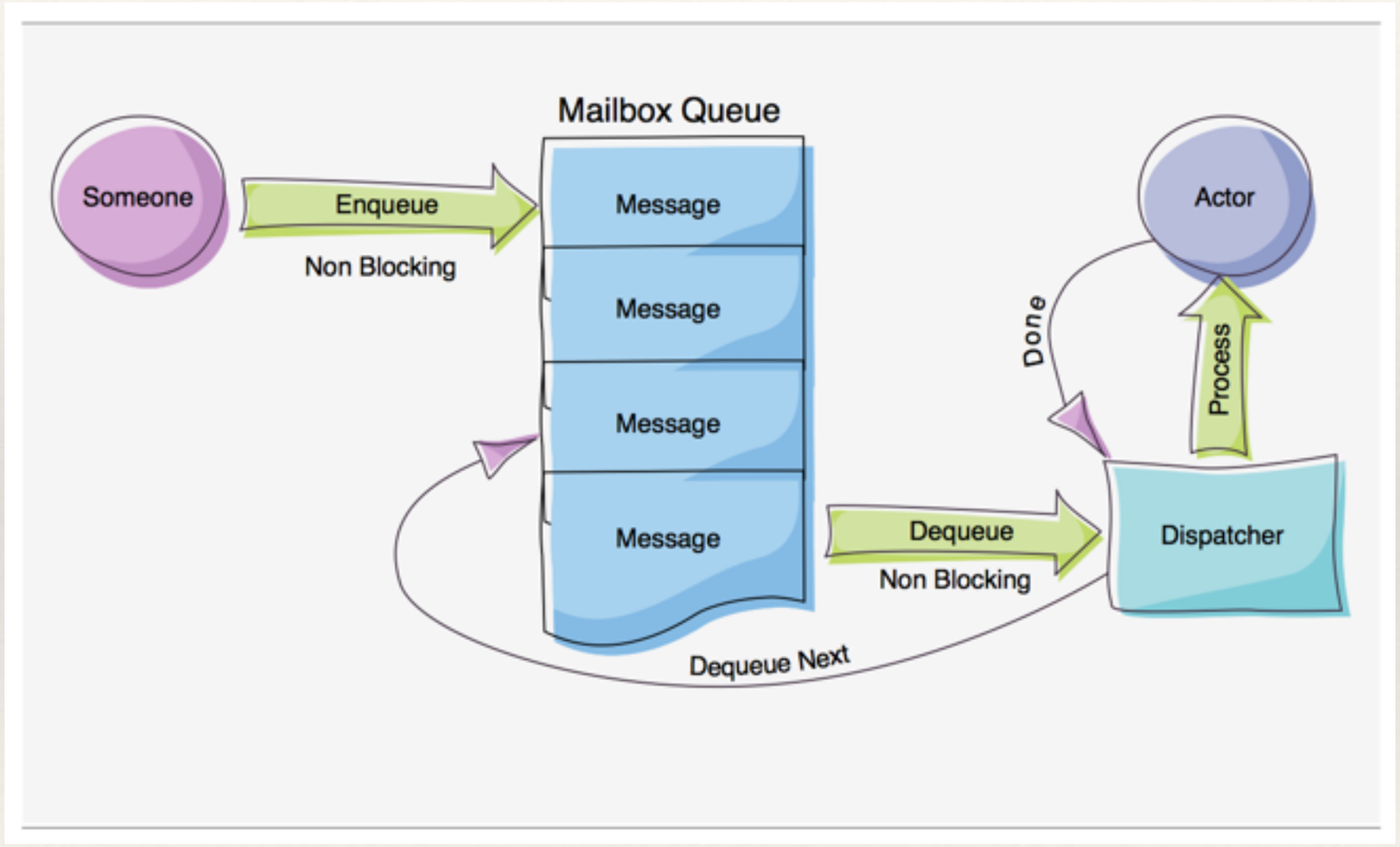
Akka

- Java / Scala
- Dagitik.
- Kendi kendini iyilestiren.
- Ayristirilmis durum (isolated state).

Akka

- Actor Bilesenleri:
  - Actor
  - ActorRef
  - Mailbox
  - Dispatcher

Akka


Mailbox Queue
Someone
Enqueue
Non Blocking
Message
Message
Message
Message
Actor
Done
Process
Dequeue
Non Blocking
Dispatcher
Dequeue Next

Akka

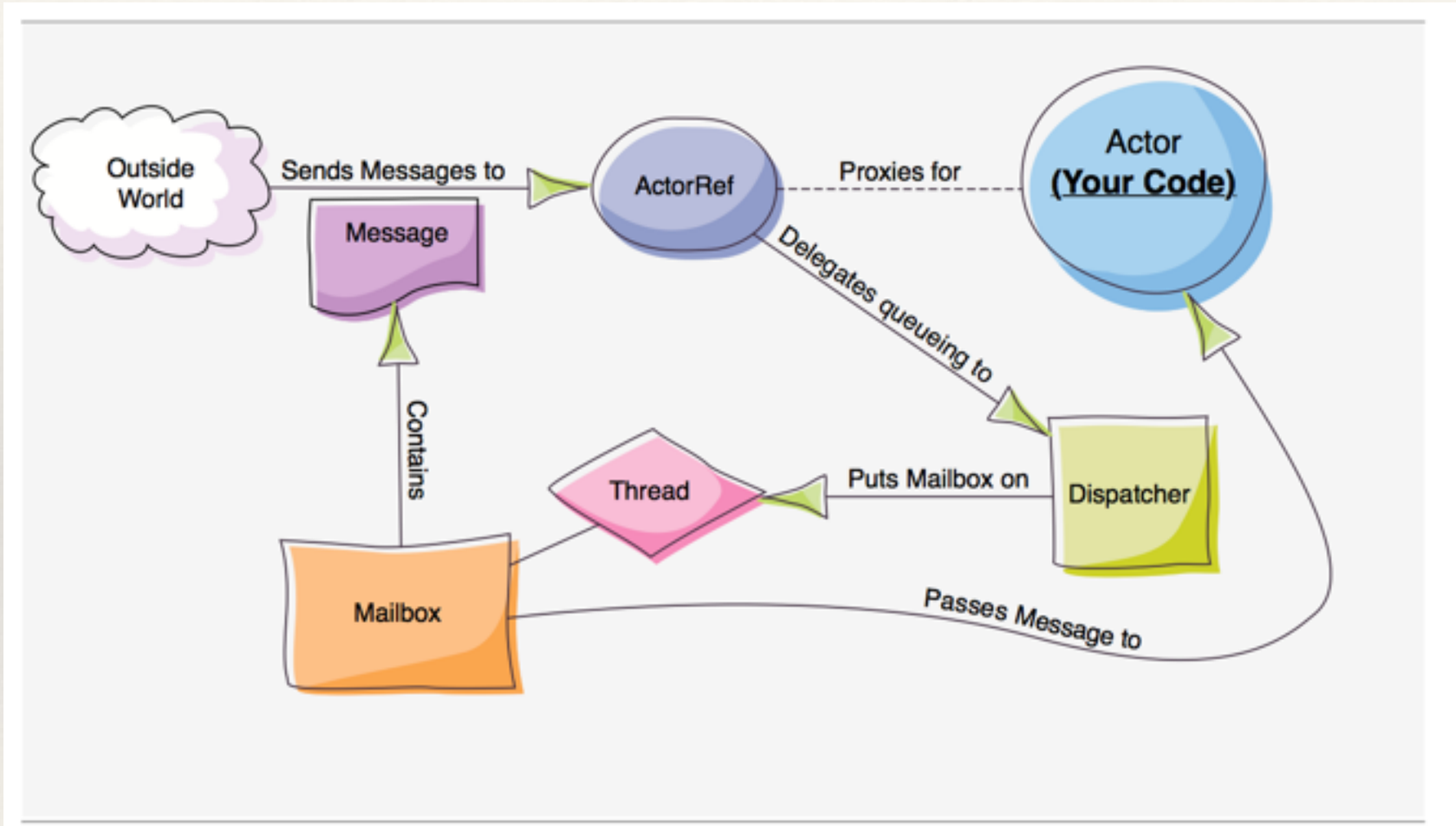
Java

```
public class Greeting implements Serializable {
  public final String who;
  public Greeting(String who) { this.who = who; }
}

public class GreetingActor extends UntypedActor {
  LoggingAdapter log = Logging.getLogger(getContext().system(), this);

  public void onReceive(Object message) throws Exception {
    if (message instanceof Greeting)
      log.info("Hello " + ((Greeting) message).who);
  }
}

ActorSystem system = ActorSystem.create("MySystem");
ActorRef greeter = system.actorOf(Props.create(GreetingActor.class), "greeter");
greeter.tell(new Greeting("Charlie Parker"), ActorRef.noSender());
```

Scala

```
case class Greeting(who: String)

class GreetingActor extends Actor with ActorLogging {
  def receive = {
    case Greeting(who) ⇒ log.info("Hello " + who)
  }
}

val system = ActorSystem("MySystem")
val greeter = system.actorOf(Props[GreetingActor], name = "greeter")
greeter ! Greeting("Charlie Parker")
```

Akka


Outside World
Sends Messages to
Message
ActorRef
Proxies for
Actor
(Your Code)
Delegates queueing to
Contains
Thread
Puts Mailbox on
Dispatcher
Mailbox
Passes Message to

Akka

- Aktorler canlidir.
- Hafiftir. (~400b)
- Receive metodu boyunca calistiklari thread'i bloklarlar.
- Isimleri vardir ve bu isimler kullanilarak cagirilirlar.

Akka

- Bir actor'u oldurmenin uc yolu:
  - ActorRefFactory.stop() (asynch) (ActorContext veya ActorSystem)
  - PoisonPill
  - gracefulStop()

Akka

- Mesaj Gondermek:

```
import akka.actor.UntypedActor;
import akka.event.Logging;
import akka.event.LoggingAdapter;
public class MyUntypedActor extends UntypedActor {

  LoggingAdapter log = Logging.getLogger(getContext().system(), this);
  public void onReceive(Object message) throws Exception {
    if (message instanceof String) {
      log.info("Received String message: {}", message);
      getSender().tell(message, getSelf());
    } else
      unhandled(message);
  }
}
```

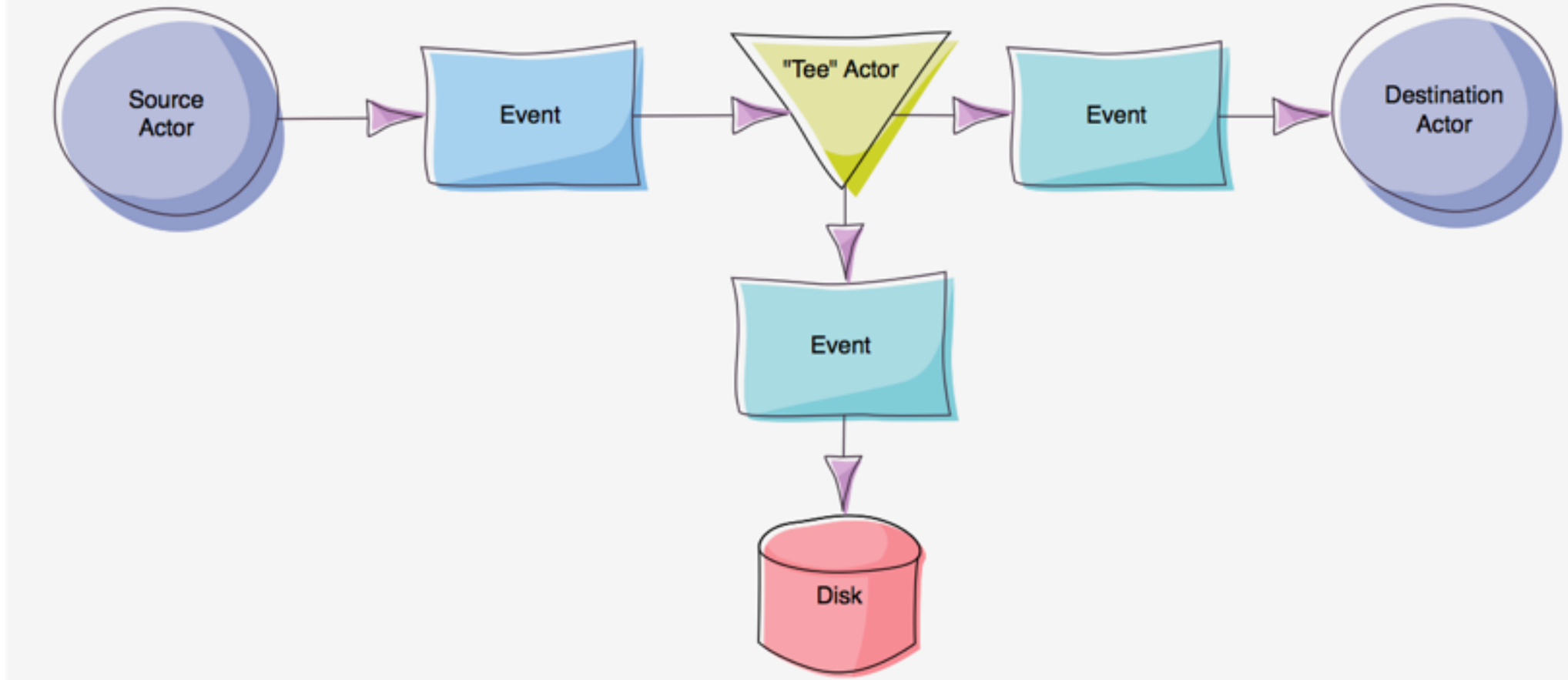
Akka

- HOCON (Human-Optimized Config Object Notation)
  - Log seviyesi ve mekanizmalari
  - Remote calismayi saglamak
  - Router'lar
  - Dispatcher'larin ayarlanmasi

Akka

```
akka {
  # Loggers to register at boot time (akka.event.Logging$DefaultLogger logs
  # to STDOUT)
  loggers = ["akka.event.slf4j.Slf4jLogger"]

  # Log level used by the configured loggers (see "loggers") as soon
  # as they have been started; before that, see "stdout-loglevel"
  # Options: OFF, ERROR, WARNING, INFO, DEBUG
  loglevel = "DEBUG"

  # Log level for the very basic logger activated during AkkaApplication startup
  # Options: OFF, ERROR, WARNING, INFO, DEBUG
  stdout-loglevel = "DEBUG"

  actor {
    default-dispatcher {
      # Throughput for default Dispatcher, set to 1 for as fair as possible
      throughput = 10
     }
  }

  remote {
    server {
        # The port clients should connect to. Default is 2552 (AKKA)
        port = 2562
    }
  }
}
```

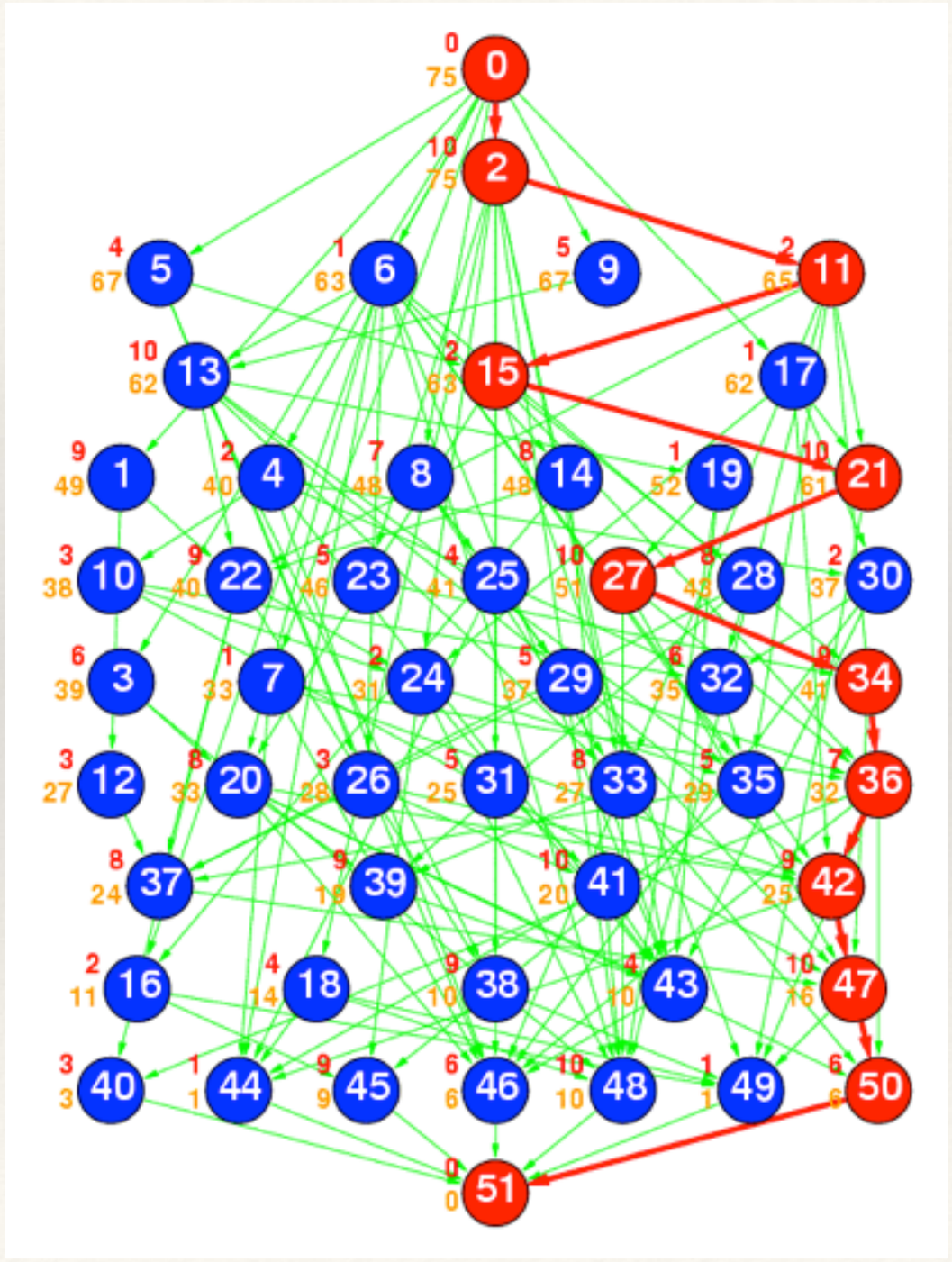
Akka


Source Actor
Event
"Tee" Actor
Event
Destination Actor
Event
Disk

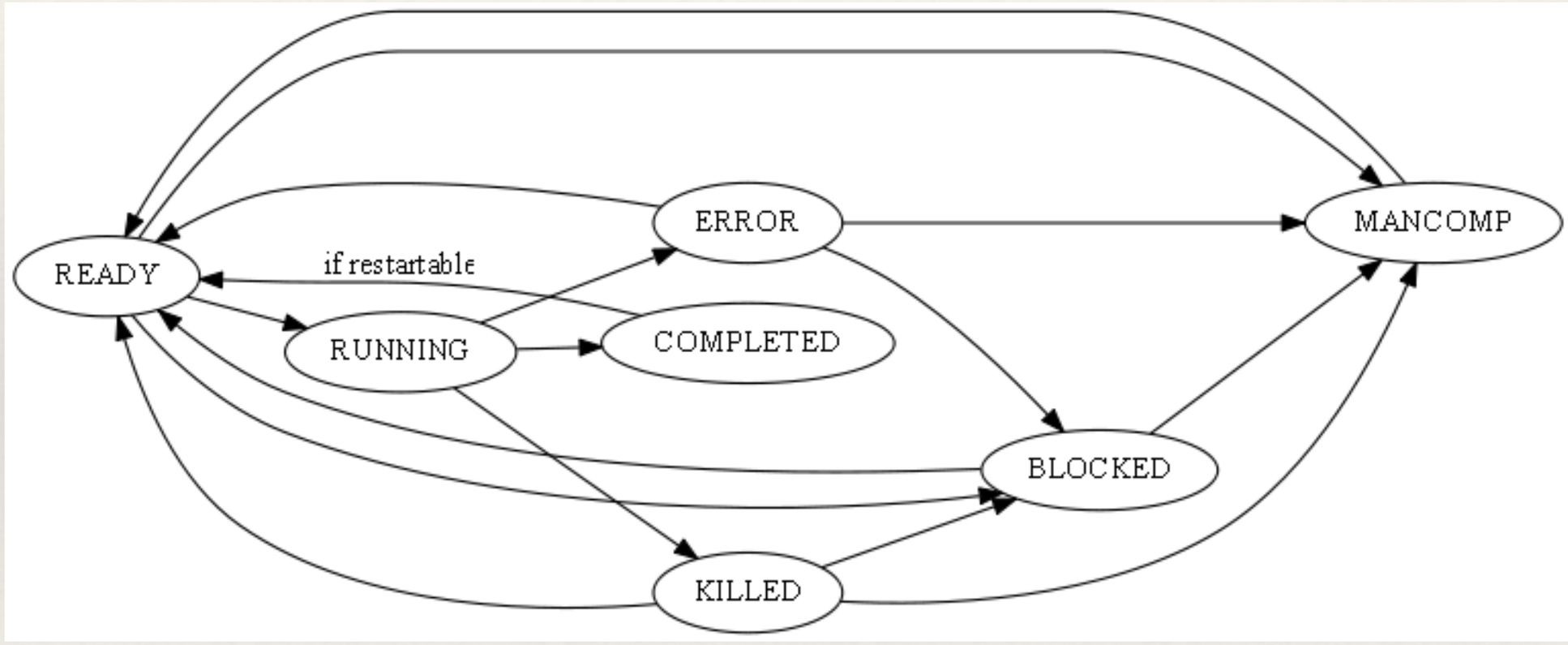
Turkcell - Scheduler

- Data warehouse scheduler
- Plans
  - 10 - 3000 Jobs / Plan
  - Bagimli job’lar

Turkcell - Scheduler



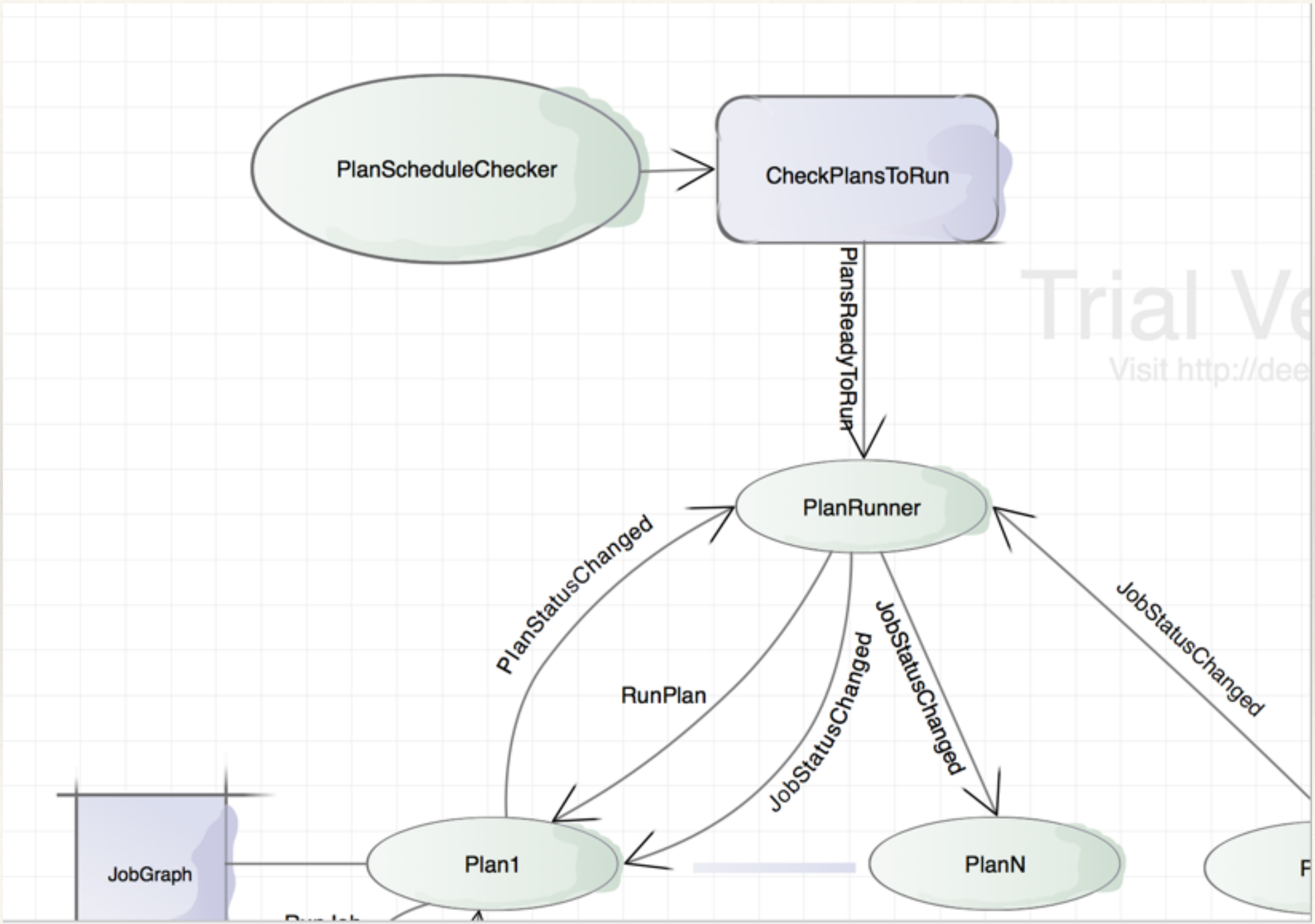
Turkcell - Scheduler



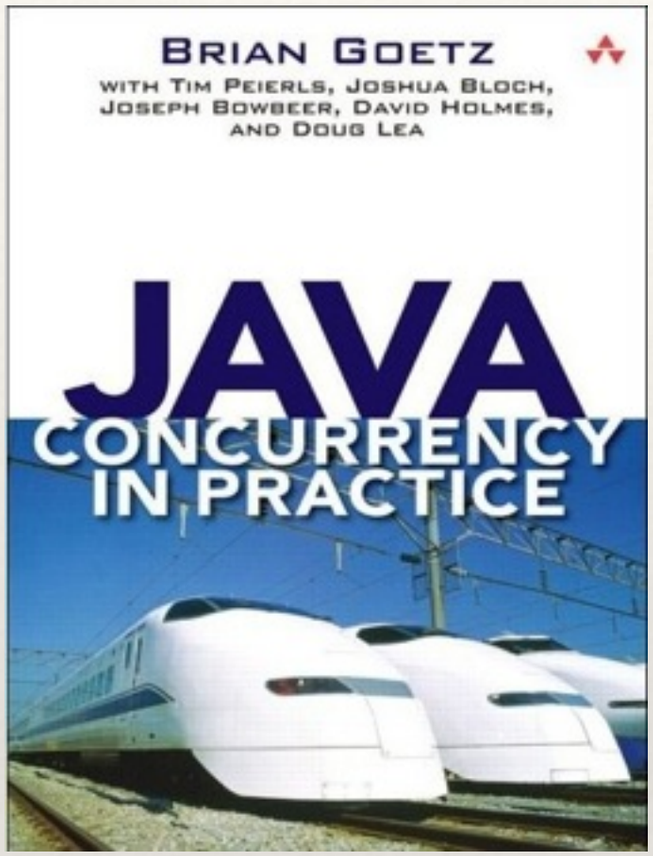
Turkcell - Scheduler

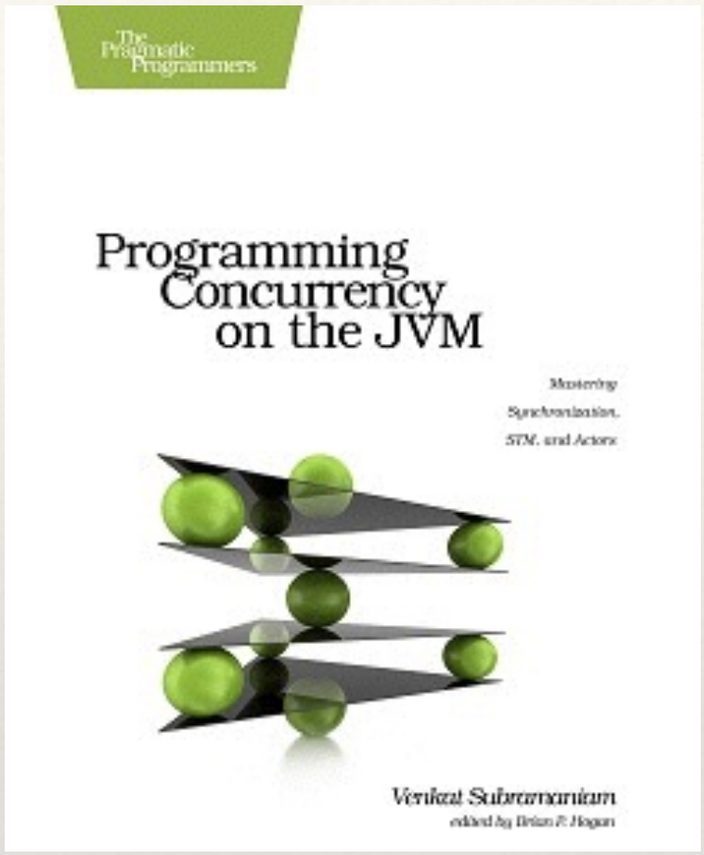
- Runtime'da executor sayisinin degistirilmesi
- Calisan job'lara mudale edebilme
- Module / Parent Module executor limitleri
- …

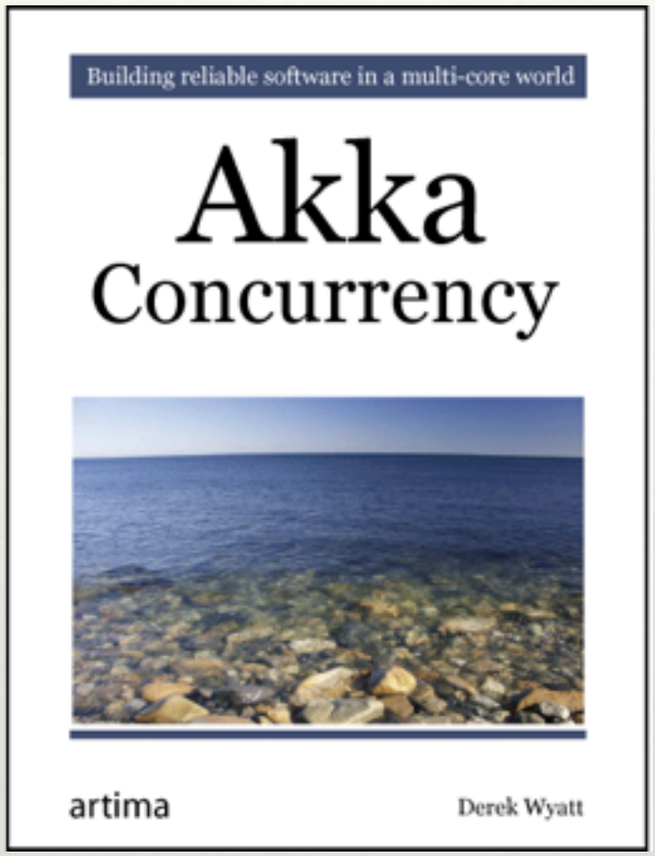
Turkcell - Scheduler



Referanslar







- Akka Documentation: http://akka.io/docs/
- Bret Victor, Future of Programming : http://vimeo.com/71278954
- Bret Victor, Inventing on Principle : http://vimeo.com/36579366
- Herb Sutter, Free lunch is over: http://www.gotw.ca/publications/concurrency-ddj.htm

Teşekkürler

Selim Öber

http://selimober.com

@selimober