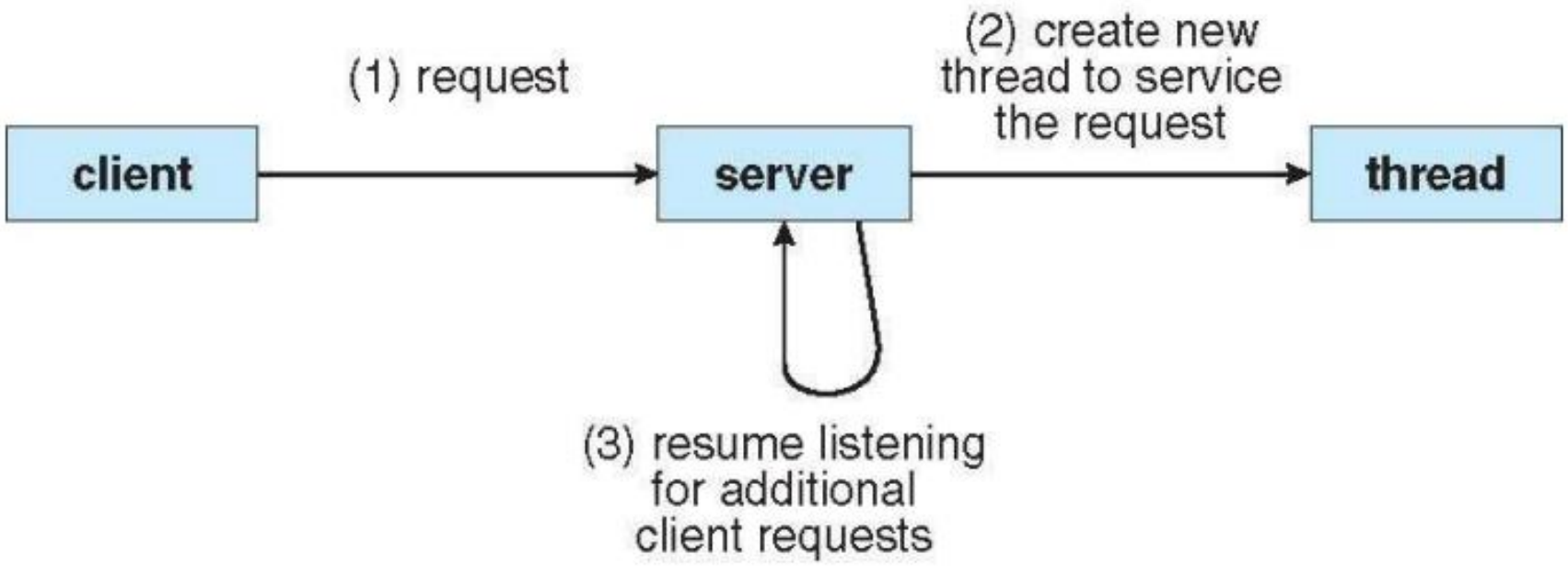
İş Parçacıkları (Threads)

Dr. Günay TEMÜR

Bölüm 4: İş Parçacıkları

- Genel Bakış
- Çoklu İş Parçacığı Modelleri
- İş Parçacığı Kütüphaneleri
- İş Parçacıkları ile İlgili Meseleler
- İşletim Sistemi Örnekleri
- Windows XP İş Parçacıkları
- Linux İş Parçacıkları

Çok İş Parçacıklı Sunucu Mimarisi

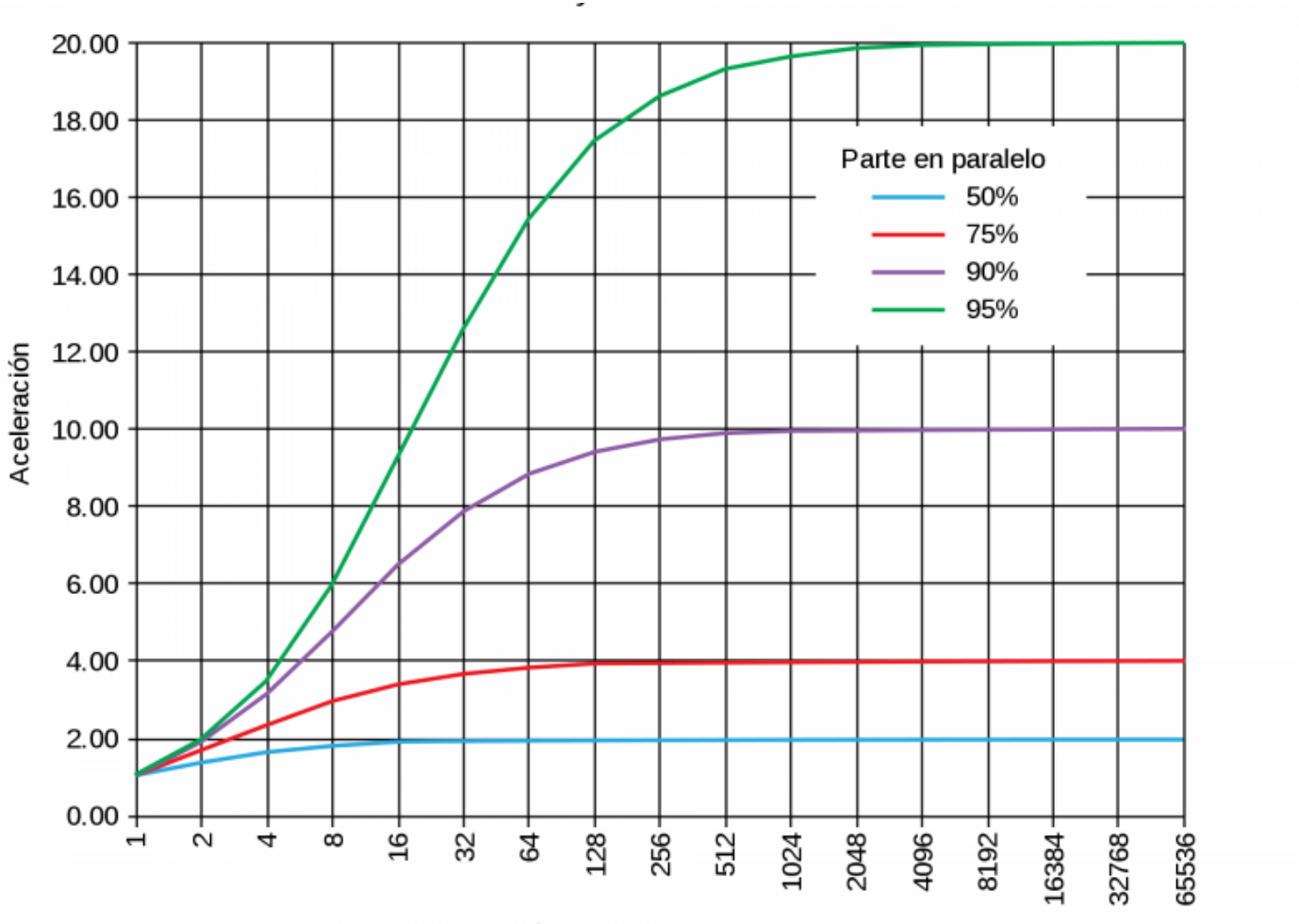


Faydalar

- Cevap Verebilirlik (Responsiveness)
- Kaynak Paylaşımı (Resource Sharing)
- Ekonomi (Economy)
  - Solaris: thread creation (1/30) ve context switch (1/5)
- Ölçeklenebilirlik (Scalability)

Amdahl Law Ölçeklenebilirlik

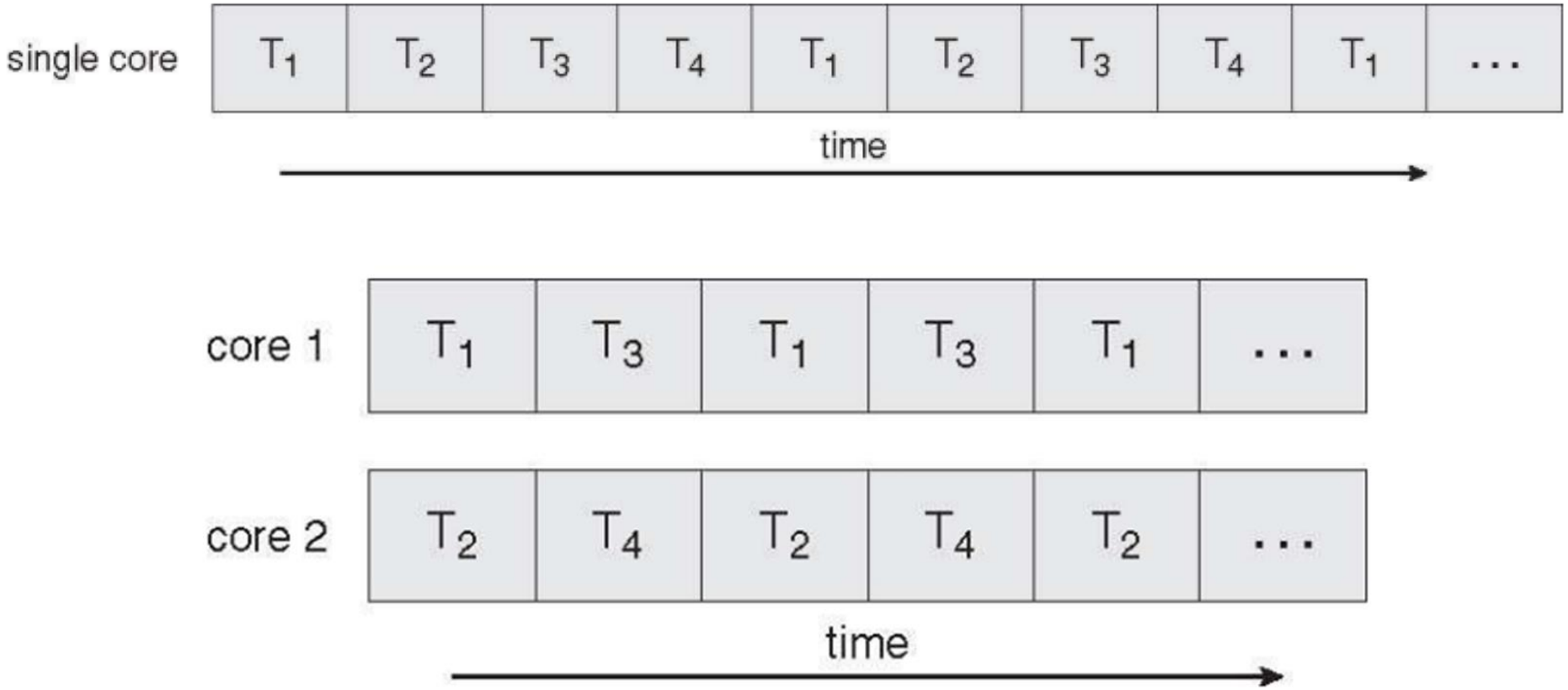
Sistemin bir parçasının hızlandırılması sonucunda, sistemin bir bütün olarak ele alındığında toplam hızlanmasının ne olacağını hesaplamak için kullanılır. **Sisteminizde iyileştirme yapmanın buna değip değmeyeceğini tanımlayın**



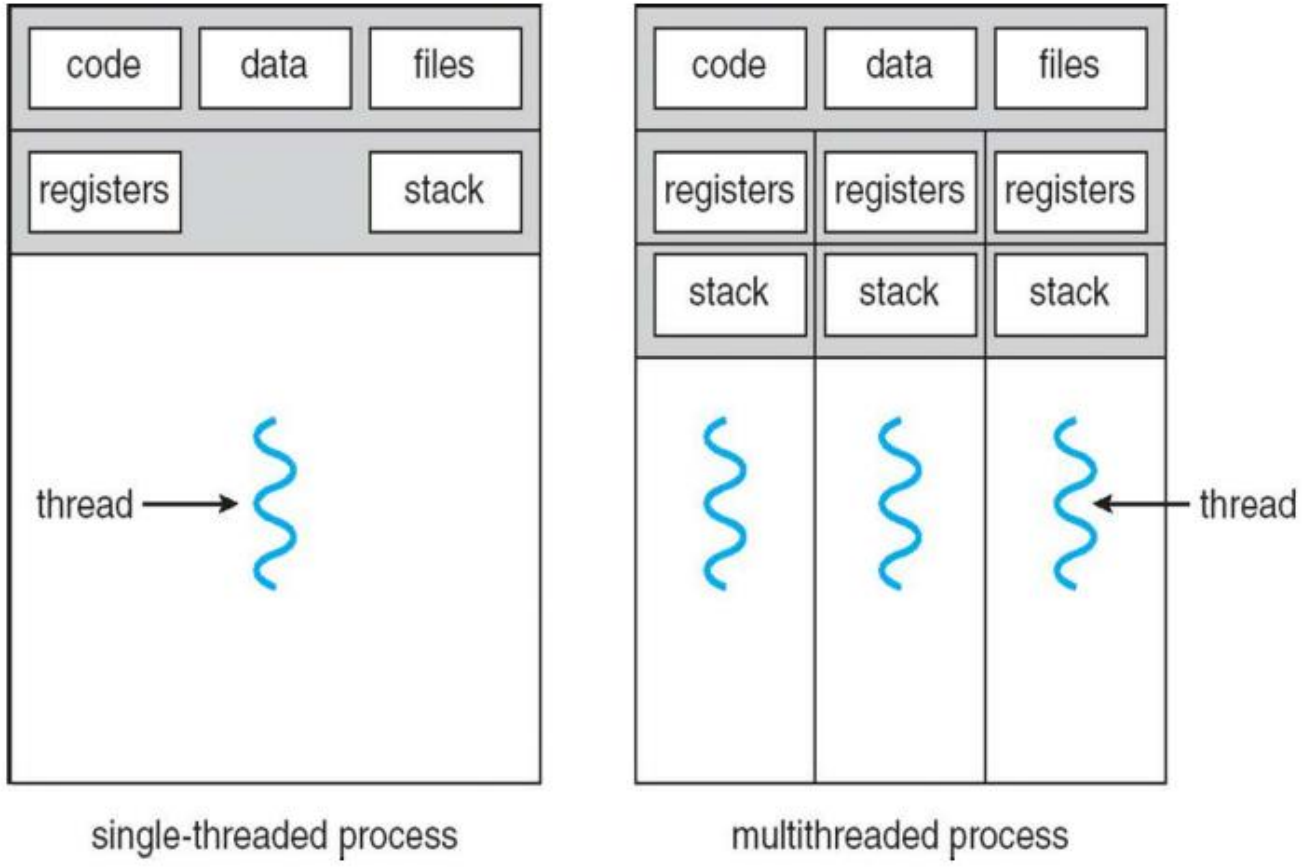
Çok Çekirdekli Programlama

- Çok-çekirdekli sistemler programcıları çok iş parçacıklı uygulamalar yazmaya zorluyor. Bu konudaki zorluklar:
  - Aktiviteleri bölmek (dividing activities)
  - Denge (balance)
  - Bilgileri Ayırmak (data splitting)
  - Veri bağımlılığı (data dependency)
  - Test ve Hata Ayıklama (testing and debugging)

Tek Çekirdekli Sistemde Eş Zamanlı Çalıştırma



Tek ve Çok İş Parçacıklı İşlemler



Kullanıcı İş Parçacıkları

- İş parçacığı yönetimi kullanıcı seviyesinde tanımlı iş parçacığı kütüphaneleri ile sağlanır
- Üç ana iş parçacığı kütüphanesi:
  - POSIX Pthreads
  - Win32 iş parçacıkları
  - Java iş parçacıkları

Çekirdek İş Parçacıkları

- Çekirdek tarafından desteklenirler
- Örnekler
  - Windows XP/2000
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

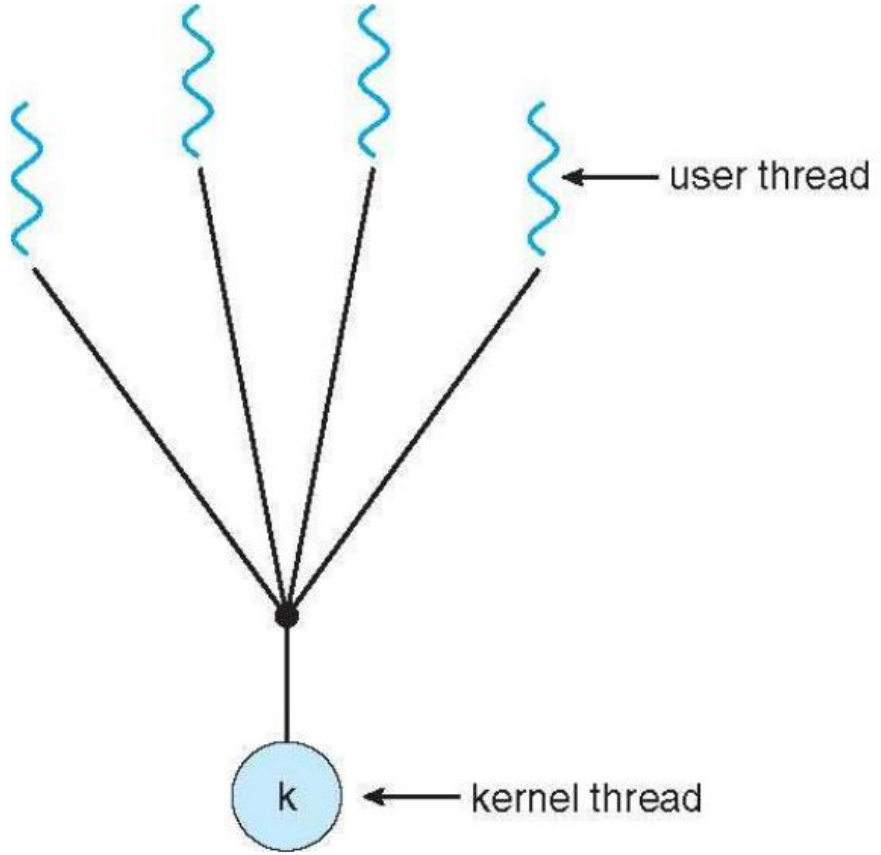
Çok İş Parçacığı Modelleri

- Çoktan-Teke (Many-to-One)
- Teke-Tek (One-to-One)
- Çoktan-Çoka (Many-to-Many)

Çoktan Teke (Many to One)

- Pek çok kullanıcı seviyesindeki iş parçacığı tek bir çekirdek iş parçacığı ile eşleşir
- Örnekler:
  - Solaris Green Threads
  - GNU Portable Threads

Çoktan Teke



Zamanlayıcılar

- Kısa vadeli zamanlayıcı çok sık çalıştırılır (milisaniye) hızlı çalışmalı
- Uzun vadeli zamanlayıcı çok sık çalıştırılmaz (saniye, dakika) yavaş olabilir
- Uzun vadeli zamanlayıcı çoklu programlamanın (multiprogramming) seviyesini kontrol eder
- İşlemler aşağıdaki iki kategoriye ayrılabilir:
  - I/O ağırlıklı işlemler (I/O-bound process) – CPU üzerinde kısa süreli işlerler yaparlar ve zamanlarının çoğu I/O işleri yaparak geçer
  - CPU ağırlıklı işlemler (CPU-bound process) – zamanlarının çoğunu CPU üzerinde uzun işlemler yaparak geçirirler, I/O işleri çok daha azdır

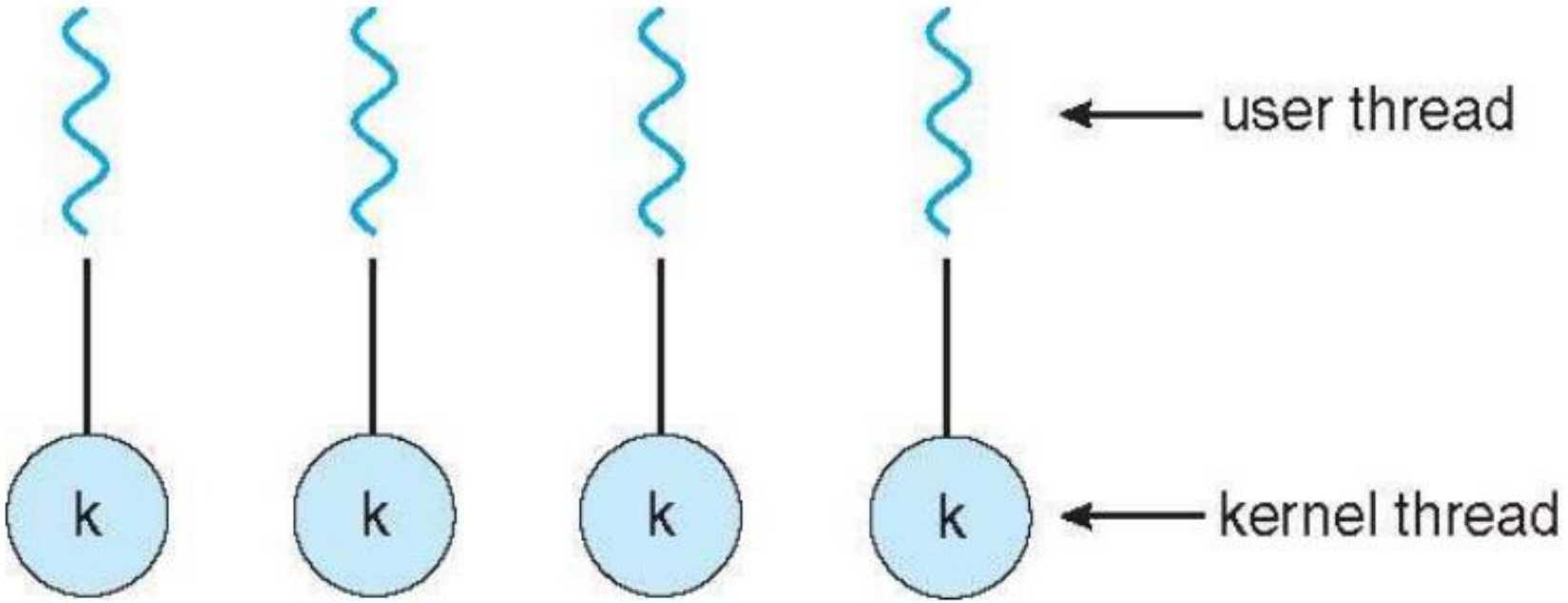
Ortam Değişikliği

- Context Switch
- CPU başka bir işleme geçerken, sistem eski işlemin durumunu kaydetmeli ve yeni işlemin kaydedilmiş durumunu yüklemelidir
- Bir işlemin ortamı (context) PCB'de tutulur
- Ortam değişikliği için harcanan zaman ek yüktür. Değişim sırasında işlemci kullanıcının işine direk yarayan bir iş yapmamaktadır
- Harcanan zaman donanım desteğine bağlıdır

Teke Tek (One to One)

- Her bir kullanıcı seviyesi iş parçacığı tek bir çekirdek iş parçacığı ile eşleşir
- Örnekler
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 ve sonrası

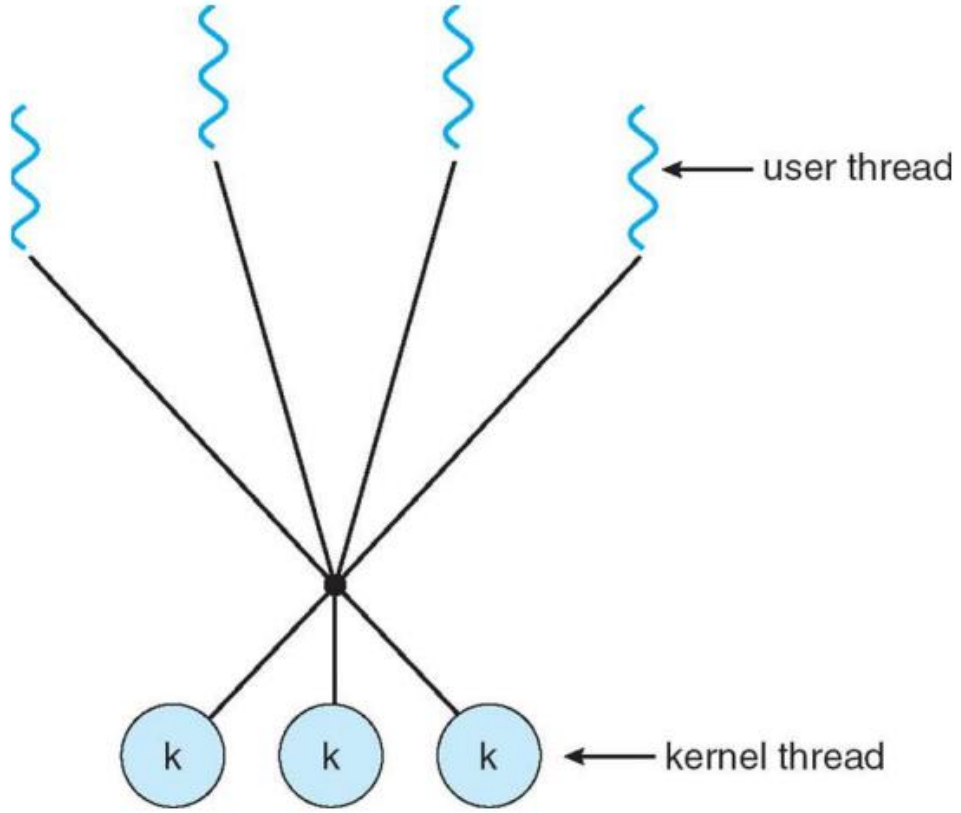
Teke Tek Model



Çoktan Çoka (Many to Many)

- Pek çok kullanıcı seviyesi iş parçacığı pek çok çekirdek iş parçacığı ile eşleşir
- İşletim sisteminin yeterince çekirdek iş parçacığı oluşturmasını sağlar
- Solaris'in version 9'a kadar olan versiyonları
- ThreadFiber paketi ile Windows NT/2000

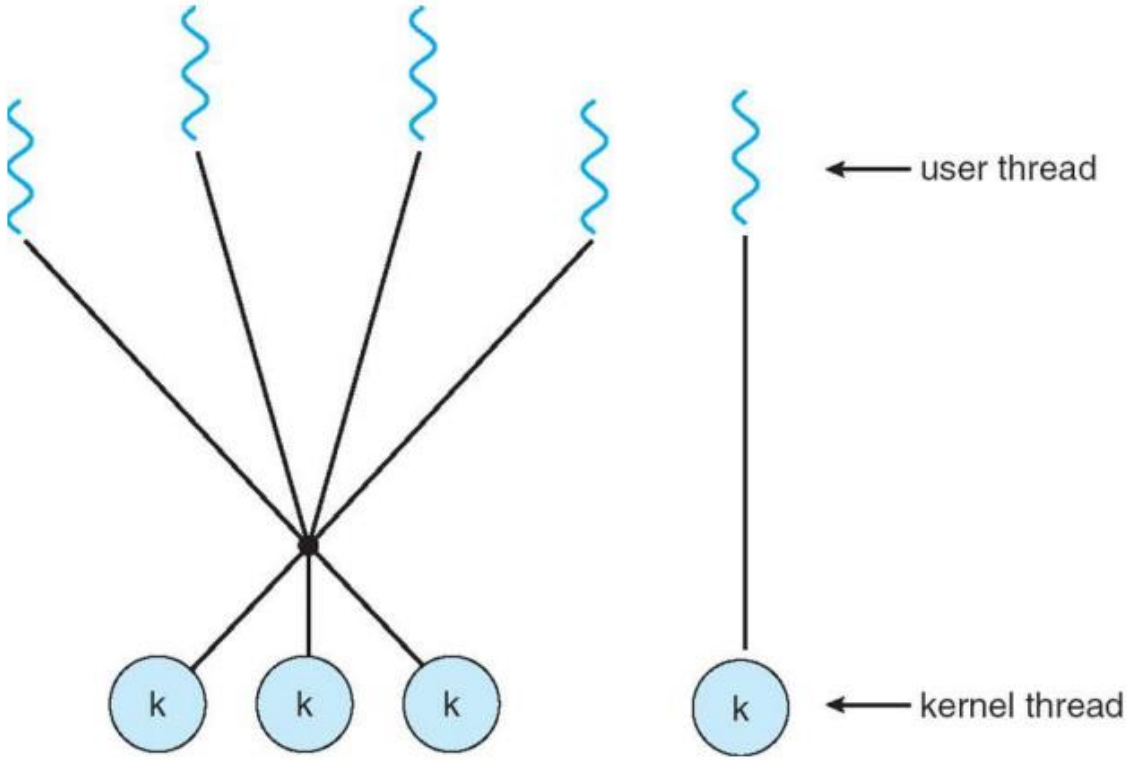
Çoktan Çoka



İki seviye Model

- M:M'e benzer. Farklı olarak kullanıcı iş parçacığının çekirdek iş parçasına bağlanmasına izin verir (bound to kernel thread)
- Örnekler
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 ve öncesi

İki Seviye Model



İş Parçacığı Kütüphaneleri

- İş parçacığı kütüphanesi programcıya iş parçacıklarının oluşturulmasını ve yönetilmesini sağlayan bir API sunar
- Gerçekleştirim için iki temel yol
  - Kütüphane tamamen kullanıcı alanında
  - İşletim sistemi tarafından desteklenen çekirdek seviyesinde kütüphane

Pthreads

- Kullanıcı seviyesinde veya çekirdek seviyesinde sunulabilir
- İş parçacığı oluşturmak ve iş parçacıklarının senkronizasyonunu sağlamak için bir POSIX standardı (IEEE 1003.1c)
- API iş parçacığı kütüphanesinin davranışını tanımlıyor. Gerçekleştirim kütüphanenin gerçekleştirimine bağlı
- UNIX işletim sistemlerinde genel olarak kullanılıyor (Solaris, Linux, Mac OS X)

Pthreads Example

```
#include"pthread.h"
#include<stdio.h>
#include<iostream>
#include<string>
#include"pch.h"
using namespace std;
int sum;
void *runner(void *parm);
```

```
int main(int argc, char *argv[])
{
pthread_t tid;
pthread_attr_t attr;
if (argc != 2)
{
fprintf(stderr,"usage: a.out<integer value>\n");
return -1;
}
if (atoi(argv[1]) < 0)
{
fprintf(stderr,"%d must be >= 0\n, atoi(argv[1])");
return -1;
}
```

```
pthread_attr_init(&attr);
pthread_create(&tid, &attr, runner, argv[1]);
pthread_join(tid, NULL);
printf("sum = %d\n", sum);
}
void *runner(void *parm)
{
int i, upper = atoi(parm);
sum = 0;
for (i = 1; i <= upper; i++)
sum += i;
}
```

BİTTİ