Humboldt University

Computer Science Department
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Operating Systems Principles

Event-
und
Threadbasierte
Programmierung

Aufgabe

- 2 Wochen Zeit zum Lösen der Aufgaben
  - Aufgabenstellung auf der SAR Website
  - Abgabe über GOYA
  - Abgabefrist: 01.12.2014 09:00
- In dieser Woche
  - Erläutern der Aufgabenstellung
  - Nächste Woche: Fragen zur Aufgabe, Cachegrind
- Nächste Veranstaltung
  - 21.11.2014

Aufgabe

Aufgabe

Es soll thread- und eventbasierte Programmierung anhand einer Client-Server-Anwendung verglichen werden. In welchen Situationen hat die eventbasierte und in welcher die threadbasierte Variante performanter. Welche Vorteile haben die einzelne Verfahren.

Details

Implementieren Sie einen Server, der zum Einen Dateien ausliefern und zum Anderen eine Berechnung durchführen kann. Der Client soll sich mittels TCP zum Server verbinden und dann wahlweise die Berechnung anstoßen oder eine Datei anfordern. Der Name der Datei ist fix, d.h. es wird nur eine Datei angefordert. Die Verbindung soll auch nach der Anfrage bestehen bleiben und weitere Anfragen sollen möglich sein.

Aufgabe

Server

Der Server soll mehrere Client gleichzeitig bedienen können. Eine Verbindung vom Client zum Server soll so lange bestehen bis der Client

sie beendet. Der Server muss entsprechend mehrere Verbindungen halten.

Eventbasiert

Mit Hilfe von "select" soll sowohl auf neue Verbindungsanfragen als auch auf Kommandos von bereits verbundenen Clienten gewarten werden. Anfragen werden sofort beantwortet, d.h. es wird entsprechend eine Datei ausgeliefert oder eine Berechnung durchgeführt. Verbindungsanfragen werden akzeptiert und der die neue Verbindung mit "select" überwacht.

Threadbasiert

Bei einer Verbindungsanfrage akzeptiert der Server diese und erzeugt einen neuen Thread, der von da an mit diesem neuen Client kommuniziert, seine Kommandos entgegennimmt und bearbeitet. sobald der Client die Verbindung beendet, wird auch der Thread beendet.

Aufgabe

Client

Als Client soll httperf verwendet werden. Das Kommando (Filetransfer oder Berechnung) und das Argument (Grösse der Primzahl bzw. Anzahl der übertragenen Bytes) wird in der URL kodiert.

Der Client kann ein Exit-Kommando schicken und so den Server beenden.

Ziel der Aufgabe

Vergleichen Sie die Performance vom Thread- und Event-basiertem Server unter verschiedenen Bedingungen. Welcher ist bei I/O-Anfragen, welcher bei CPU-lastigen Anfragen besser. Warum? Belegen Sie Ihre Annahmen mit entsprechenden Untersuchungen. Werten Sie diese geeignet aus.

Details

Vorgabe

Eigene Vorlagen**:**

- **Eventqueue aus der letzten Aufgabe**

Folgende Vorgaben gibt es:

- Makefile
- server_(event|simple|thread).c
- server_socket.(c|h)
- prim_calc.(c|h)
- protocol.h & server_config.h
- session_*.txt & start_http_test.sh

Makefile

- Targets
  - all: baut alles
  - test_simple: testet den einfachen Server
  - test_thread: testet den Thread-basierten Server
  - test_event: testet den Event-basierten Server
  - pack: erstellt ein tar.gz welches über goya abgegeben wird

Für den Test wird *httperf* verwendet (siehe ./start_http_test.sh). Der test besteht aus 3 Messung:

- CPU-Last
- IO-Last
- Mixed

Die Dateien session_*.txt sind Bestandteil des Tests.

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Details

server_(event|simple|thread).c

Implementierung der jeweiligen Server. Die Vorlage enthält eine fertige Implementierung von server_simple. Nutzen Sie diese für die Implemetierung des Event-basierten bzw. Thread-Basierten Servers.

server_socket.(c|h)

Diese Dateien enthalten Funktionen und Strukturen für den Server (TCP-Socket).

prim_calc.(c|h)

Funktionen zur Berechnung von Primzahlen.

Details

protocol.h

Funktionen zur Decodierung von Kommandos. Bitte nutzen Sie die Funktion: get_cmd_url

Die Kommandos:

- Exit: Server beendet sich
- Filetransfer: *Args* Bytes einer Datei werden übertragen.
- Calculation: Grösste Primzahl kleiner *Args* wird berechnet.

Hinweise zur Aufgabe

Schauen Sie die Implementierung von server_simple an. Nutzen Sie den Filetransfer, den Aufruf der Berechnung und das Decodierung des Kommandos aus dieser Vorlage.

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Contest

Implementieren Sie einen Server, der in verschiedenen Situationen möglichst performant ist. Nutzen Sie dazu die Erfahrungen mit den Event- und thread-basierten Server.

Systems Architecture Group
http://sar.informatik.hu-berlin.de

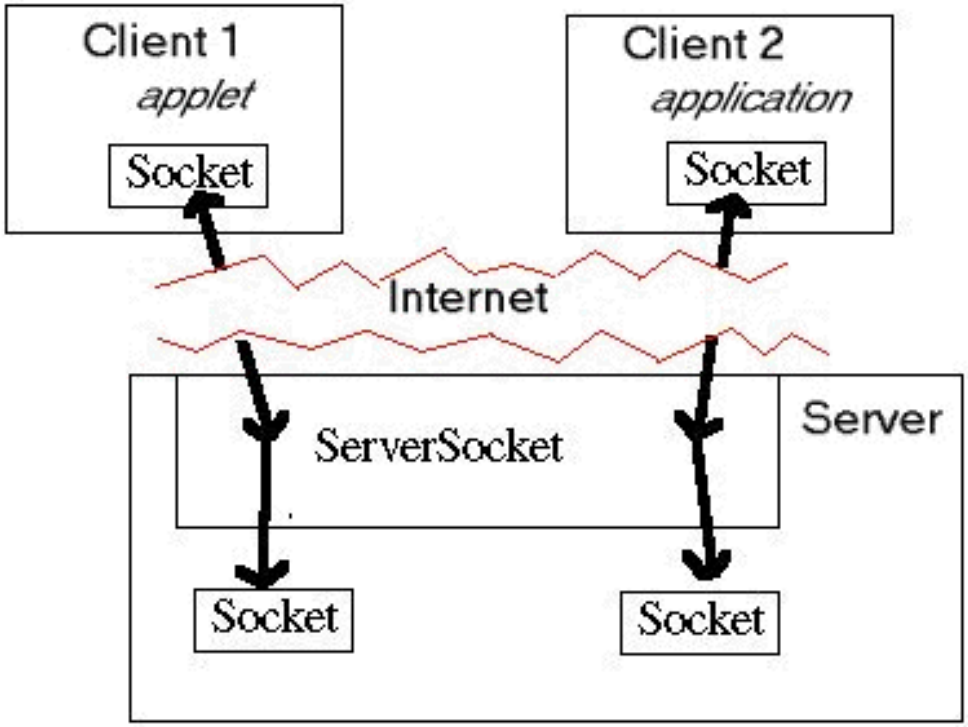
Hinweise

Sockets

Socket

is one endpoint of a two-way communication link between two programs running on the network

is bound to a port number so that the TCP layer can identify the application that data is destined to be sent

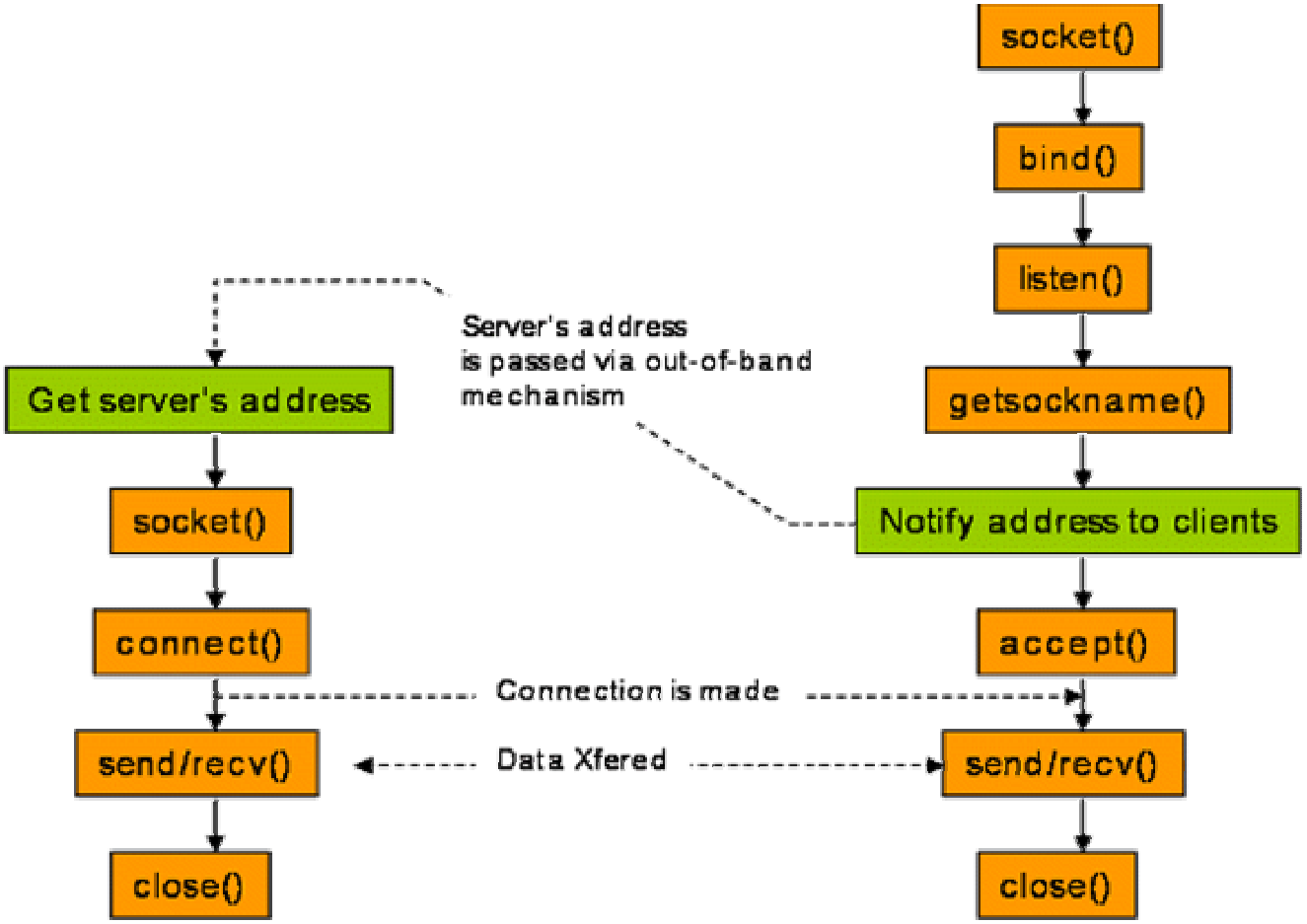


Systems Architecture Group
http://sar.informatik.hu-berlin.de

Sockets

Socket-API

socket, bind, connect,…..



Systems Architecture Group
http://sar.informatik.hu-berlin.de

Sockets

```
struct sockaddr {
   unsigned short    sa_family;    // address family, AF_xxx
   char              sa_data[14];  // 14 bytes of protocol address
};
```

IPv4 sockets (AF_INET)

```
struct sockaddr_in {
    short          sin_family;   // e.g. AF_INET, AF_INET6
    unsigned short   sin_port;     // e.g. htons(3490)
    struct in_addr   sin_addr;     // see struct in_addr, below
    char           sin_zero[8];  // zero this if you want to
};
```

Sockets

#include <sys/socket.h>
int socket(int domain, int type, int protocol);

create an unbound socket in a communications domain
Return value: file descriptor

domain
Specifies the communications domain in which a socket is to be created
(e.g. AF_INET)

type
Specifies the type of socket to be created.
SOCK_STREAM -> TCP
SOCK_DGRAM -> UDP

protocol
Specifies a particular protocol to be used with the socket.

Socket

#include <sys/socket.h>

int bind(int socket, const struct sockaddr *address, socklen_t address_len);

assign a local socket address address to a socket identified by descriptor socket

socket

Specifies the file descriptor of the socket to be bound.

address

Points to a sockaddr structure containing the address to be bound to the socket. The length and format of the address depend on the address family of the socket.

address_len

Length of the address

Socket

```
#include <sys/socket.h>
int listen(int socket, int backlog);
```

listen

listen for socket connections

backlog

Limit the queue of incoming connections

Socket

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *addr, socklen_t
    addrlen);
```

connects the socket referred to by the file descriptor *sockfd*

to the address specified by *addr*

addrlen argument specifies the size of *addr*.

Socket

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(int sockfd, struct sockaddr *addr, socklen_t
    *addrlen);
```

extracts the first connection request on the queue of pending connections for the listening socket

recvfrom

NAME

recvfrom - receive a message from a socket

SYNOPSIS

#include <sys/socket.h>

ssize_t recvfrom(int *socket*, void **buffer*, size_t *length*, int *flags*, struct sockaddr **address*, socklen_t **address_len*);

DESCRIPTION

The *recvfrom()* function receives a message from a connection-mode or connectionless-mode socket. It is normally used with connectionless-mode sockets because it permits the application to retrieve the source address of received data.

Threads

POSIX-Threads (Pthreads)

Standard IEEE P1003.1c

In vielen Systemen realisiert

Konzepte in anderen Thread-Realisierungen meist ähnlich

POSIX-konforme Realisierungen unter Linux

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Thread-Erzeugung

Programmiermodell

- Bei Start eines Prozesses existiert genau ein Thread
- Dieser erzeugt ggf. weitere Threads und wartet auf deren Ende
- Terminierung des Prozesses bei Terminierung des Master-Threads

Funktion zur Thread-Erzeugung

```
int pthread_create(
        pthread_t *new_thrd_ID,
        const pthread_attr_t *attr,
        void *(*start_func)(void *),
        void *arg)
```


Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Thread-Attribute

Stackgröße

Scheduling-Schema und Priorität

detachstate: Verhalten bei Beendigung desThreads

Bei **detached thread** erfolgt die Freigabe der Ressourcen sofort bei Beendigung, ansonsten erst nach Abschlußsynchronisation

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Thread-Verwaltung

pthread_self

Liefert eigene Thread-ID

pthread_exit

Eigene Terminierung

pthread_cancel

Terminiert einen Thread

Terminierung nur an bestimmten Punkten

Vor Terminierung **cleanup handler** aufrufen

Die Pthread-Schnittstelle

Thread-Verwaltung...

pthread_join

Wartet auf Terminierung des spezifizierten Threads (Abschlußsynchronisation)

pthread_sigmask

Setzt Signalmaske (jeder Thread hat eigene Maske)

pthread_kill

Sendet Signal an anderen Thread innerhalb des Prozesses

Von außen nur Signal an irgendeinen Thread möglich

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Mutex-Operation (wechselseitiger Ausschluß)

pthread_mutex_init

Initialisiert Sperrvariable (mutex)

pthread_mutex_destroy

pthread_mutex_lock

Blockiert Thread, bis Sperre frei ist und belegt dann die Sperre

pthread_mutex_trylock

Belegt Sperre, falls möglich / kein Blockieren

pthread_mutex_unlock

Anmerkungen

Bei Terminierung eines Threads werden Sperren nicht automatisch freigegeben

Die Pthread-Schnittstelle

- Bedingungsvariablen
- Zur Signalisierung von Bedingungen zwischen Threads
- Erlauben Realisierung von Monitoren (strukturierte Form des wechselseitigen Ausschluß nach Hoare)
  - Extern sichtbare Funktionen eines Moduls stehen unter wechselseitigem Ausschluß
  - Aufrufer braucht sich damit nicht um Synchronisation kümmern