ΕΠΛ 372: Παράλληλη Επεξεργασία

Εισαγωγή Στον Προγραμματισμό Παράλληλων Συστημάτων
Παράλληλος Προγραμματισμός με Fork

Εργαστήριο 1

Πέτρος Παναγή

E. Dijkstra - 1972

In 1972 **E. Dijkstra** during a Turing Award Lecture has stated that:

“as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."

Προγραμματισμός Υπολογιστών

"First Software Crisis" in the 60's and 70's

Computer programming was done using **Machine Code** and **Assembly**.

Fortran and **C** made their appearance to resolve this crisis.

"Second Software Crisis" in the 80's and 90's

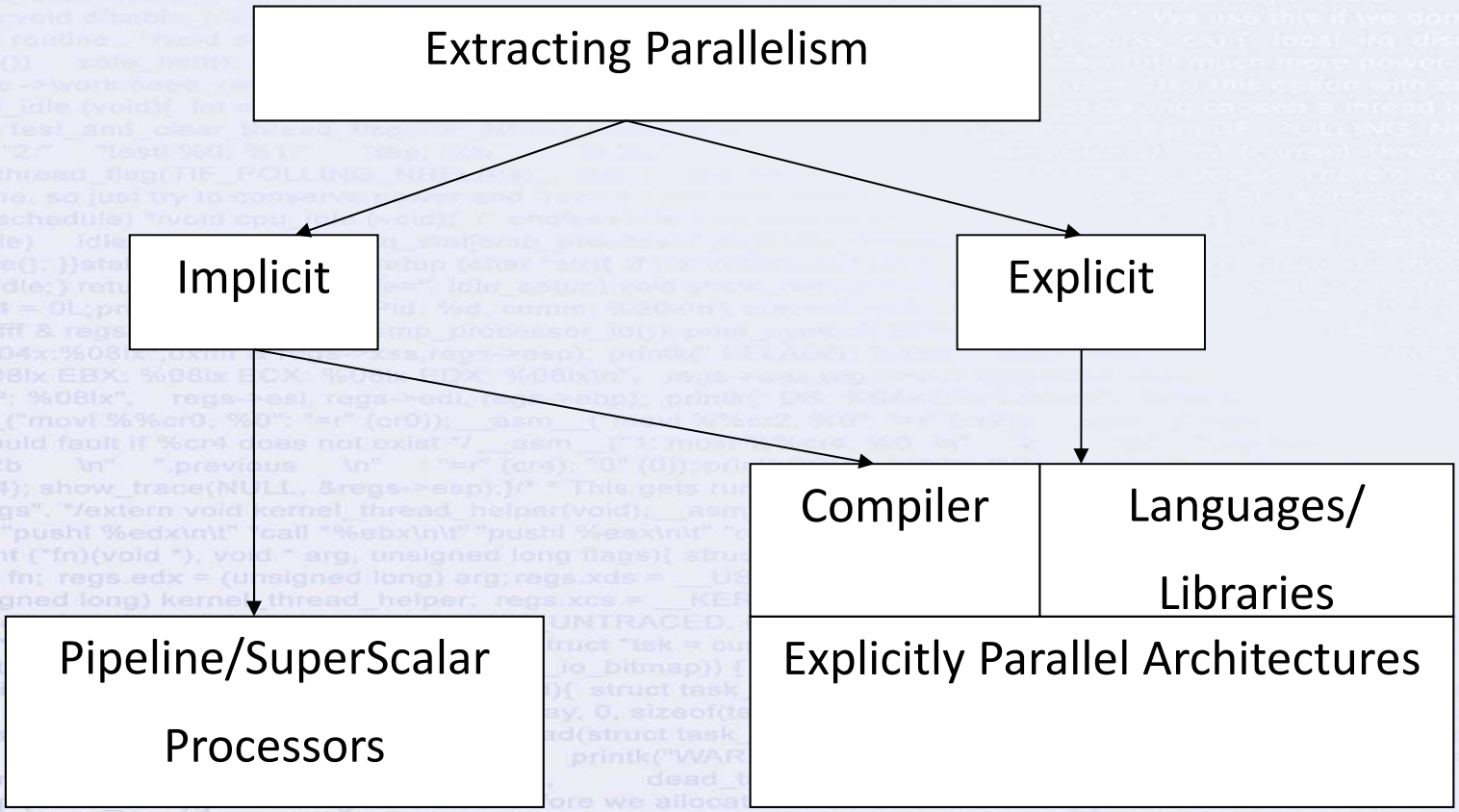
The inability to build and maintain complex and robust applications like Operading Systems, DB etc..

Object Oriented Programming came as the solution (C++, Java, C#)

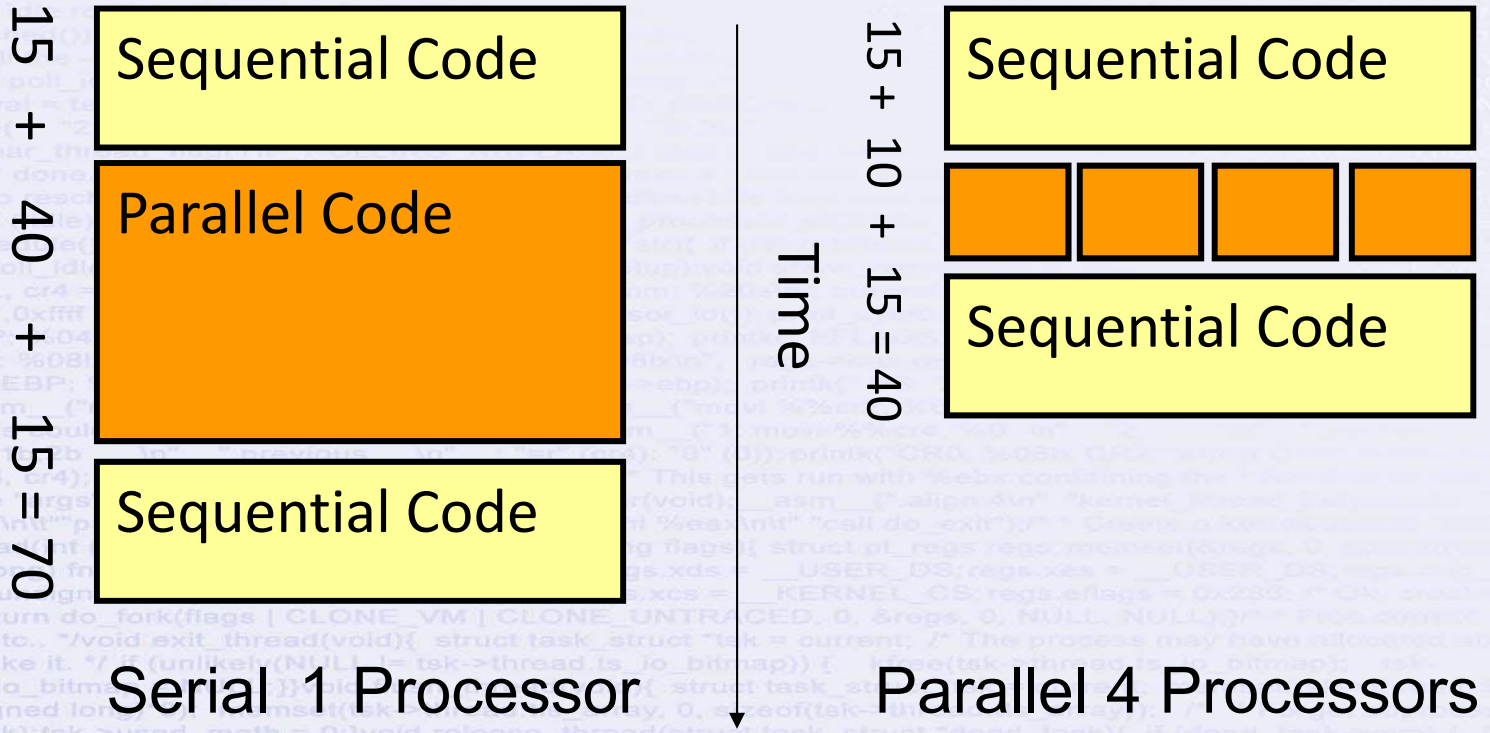
"Third Software Crisis" in the 00's until today

Parallel Computes-Multi Processor – Multi Core Machine

Προγραμματισμός Παράλληλων Συστημάτων

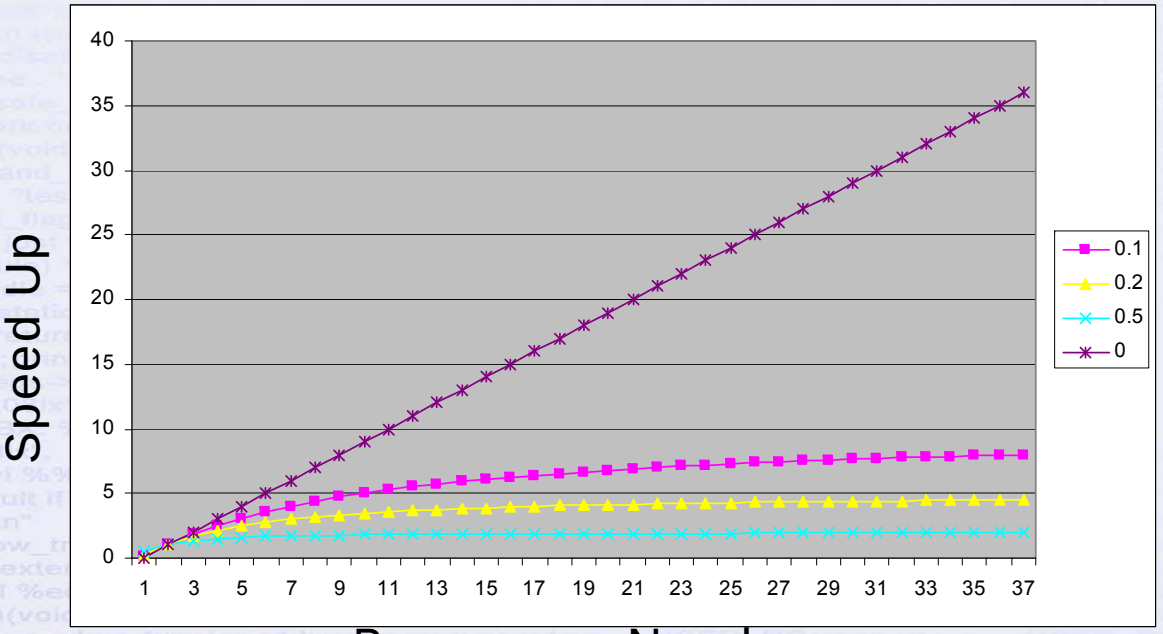


Amdahl's Law



Amdahl's Law Example. Speedup = 70/40 = **1.75**

Amdahl's Law:

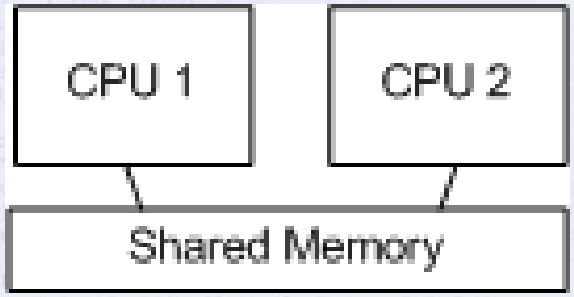


F the fraction of a calculation that is sequential

$$\frac{1}{F + (1 - F)/N}$$

Try it on Excel…

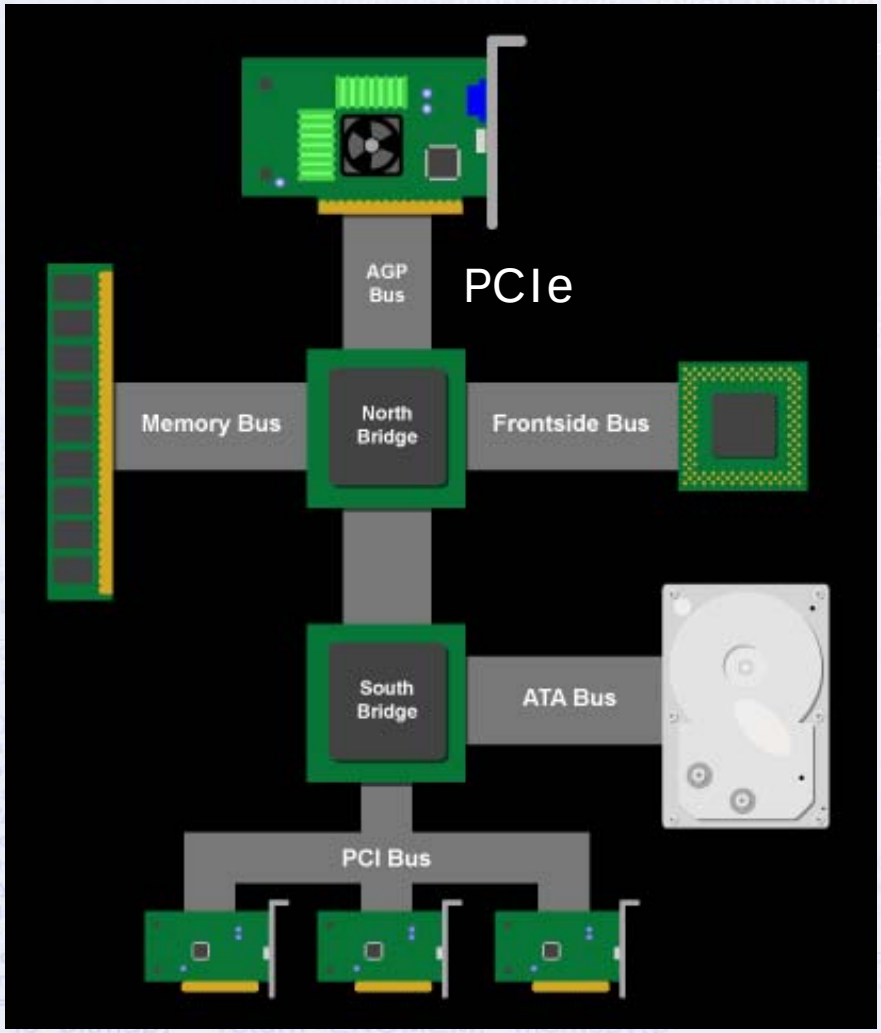
Shared Memory Model



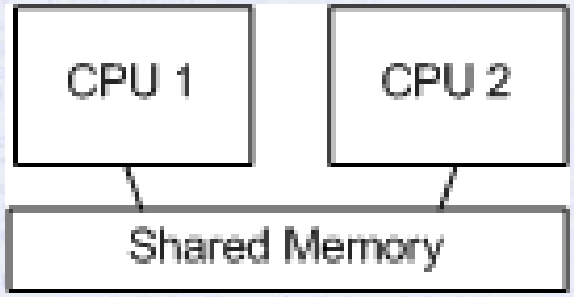
Πολλοί Επεξεργαστές μοιράζονται την ίδια μνήμη.

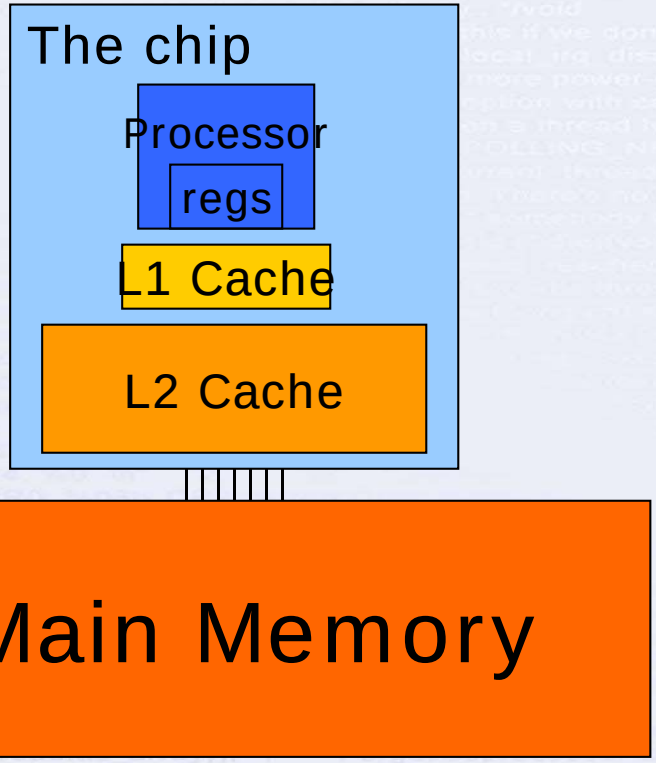
Παραδείγματα:

- Threads (POSIX Threads, pThreads)
- OpenMP



Shared Memory Model





Πολλοί Επεξεργαστές μοιράζονται την ίδια μνήμη.

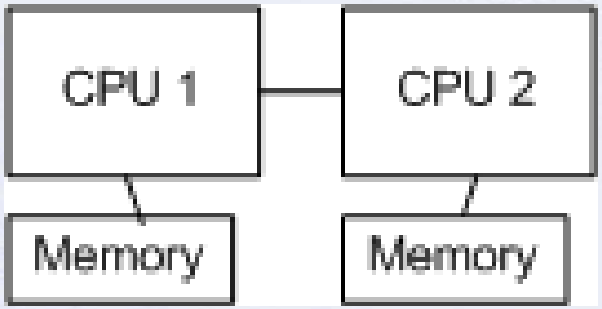
Παραδείγματα:

Threads (POSIX Threads, pThreads)

OpenMP – pragma based

Fork() Vs. Threads ???

Distributed Memory System



Κάθε επεξεργαστής έχει την δική του μνήμη. Επικοινωνούν μεταξύ του με μηνύματα. (Message Passing Model)

Παραδείγματα:

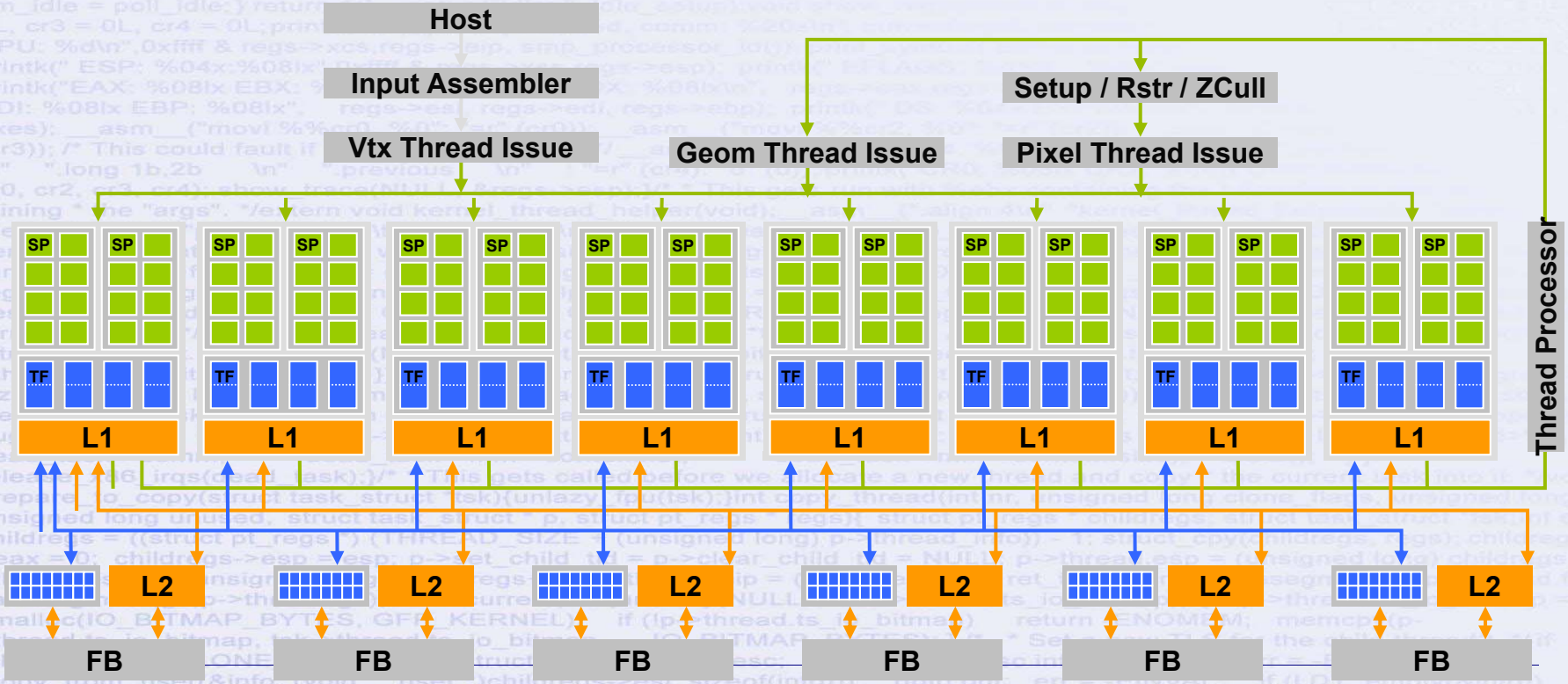
Message Passing Interface (MPI)/ Remote Procedure Call (RPC)/Remote Method Invocation (RMI)

Επιταχυντές – Accelerators- Heterogeneous Programming

FPGAs

GPU: GPGPU – CUDA – OpenCL

Heterogeneous Processors - Cell BE



Επιταχυντές - Accelerators

TOP 10 Systems - 11/2010

1	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C
2	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz
3	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU
4	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows
5	Hopper - Cray XE6 12-core 2.1 GHz
6	Tera-100 - Bull bullx super-node S6010/S6030
7	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	1684.20	IBM Thomas J. Watson Research Center	NNSA/SC Blue Gene/Q Prototype	38.80
2+	1448.03	National Astronomical Observatory of Japan	GRAPE-DR accelerator Cluster, Infiniband	24.59
2	958.35	GSIC Center, Tokyo Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows	1243.80
3	933.06	NCSA	Hybrid Cluster Core i3 2.93Ghz Dual Core, NVIDIA C2050, Infiniband	36.00
4	828.67	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	57.96
5	773.38	Universitaet Wuppertal	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Universitaet Regensburg	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Forschungszentrum Juelich (FZJ)	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
8	740.78	Universitaet Frankfurt	Supermicro Cluster, QC Opteron 2.1 GHz, ATI Radeon GPU, Infiniband	385.00
9	677.12	Georgia Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5660 2.8Ghz, nVidia Fermi, Infiniband QDR	94.40
10	636.36	National Institute for Environmental Studies	GOSAT Research Computation Facility, nvidia	117.15

fork

fork() – function call creates a new process.

SYNOPSIS

#include <unistd.h>
pid_t fork(void);

DESCRIPTION

The ***fork*()** function creates a new process. The new process (child process) is an exact copy of the calling process (parent process) which has a unique ID.

Upon successful completion, ***fork*() returns 0 to the child process and returns the process ID of the child process to the parent process**. Both processes continue to execute from the *fork*() function. Otherwise, If -1 is returned to the parent process, no child process is created.

ps -e -a pid,ppid,stat,cmd (try also ps –A –f)

```
  PID    PPID    PGID    WINPID   TTY  UID    STIME COMMAND
 3080       1    3080      3080  con 1011 10:14:00 /usr/bin/bash
 3456    3080    3456      3904  con 1011 11:20:51 /usr/bin/ps
```

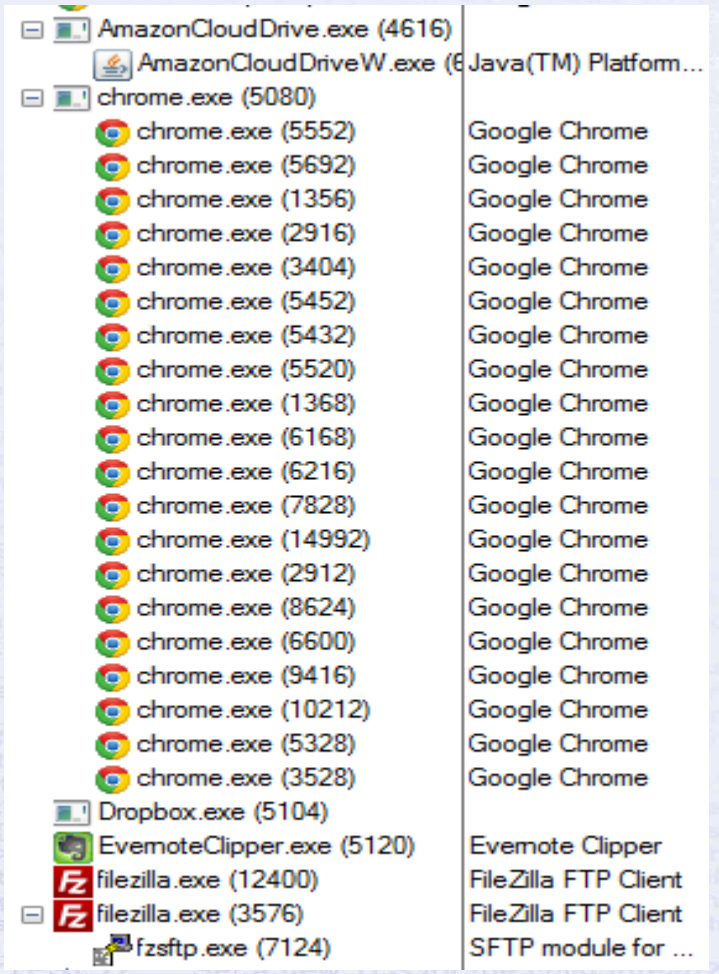
http://man7.org/linux/man-pages/man2/fork.2.html

Processes tree

ps -A --forest

```
2123 ?        00:00:00 hald
2124 ?        00:00:00  \_ hald-runner
2165 ?        00:00:00      \_ hald-addon-inpu
2166 ?        00:00:00      \_ hald-addon-acpi
2189 ?        00:00:02 automount
2212 ?        00:00:00 sshd
8296 ?        00:00:00  \_ sshd
8301 ?        00:00:00      \_ sshd
8302 pts/0    00:00:00          \_ tcsh
8379 pts/0    00:00:00              \_ ps
2220 ?        00:00:00 xinetd
2242 ?        00:00:00 ntpd
2267 ?        00:00:00 abrt-dump-oops
2287 ?        00:00:01 crond
2298 ?        00:00:00 atd
2313 ?        00:00:00 gdm-binary
2350 ?        00:00:00  \_ gdm-simple-slav
2352 tty1     00:00:04      \_ Xorg
2438 ?        00:00:00      \_ gnome-session
2462 ?        00:00:00      |   \_ at-spi-registry
2474 ?        00:00:00      |   \_ metacity
2476 ?        00:00:01      |   \_ gnome-power-man
2477 ?        00:00:00      |   \_ polkit-gnome-au
2478 ?        00:00:03      |   \_ gdm-simple-gree
2479 ?        00:00:00      |   \_ plymouth-log-vi
2490 ?        00:00:00      \_ gdm-session-wor
2319 tty2     00:00:00 mingetty
2322 tty3     00:00:00 mingetty
```

Windows Process Monitor



fork (): Copy and Run Example 1

```c
#include <stdio.h>
/* has the definition of pid_t */
#include <sys/types.h>
/* The <unistd.h> header defines miscellaneous symbolic constants and types, and declares miscellaneous functions. Like fork()*/
#include <unistd.h>
#include <time.h>
int randomNumber;
int main ()
{
    int i;
    pid_t parent_id=getpid (), child_pid;
    /* initialize random seed: */
    srand(time(NULL) );
    randomNumber = rand()%1000;
    printf ("The main program process ID is %d\n", (int)parent_id);
    /*The fork() function creates a new process. The new process (child process) is an exact copy of the calling process (parent process).*/
        for (i = 0; i< 10 && getpid () == parent_id;i++)
        child_pid = fork ();
        if (child_pid != 0) {
            printf ("This is the parent process, with id %d ", (int) getpid ());
            printf ("and my child's process ID is %d\n", (int) child_pid);
        }
        else{
            printf ("This is the child process, with id %d and RANDOM %d\n", (int) getpid (), randomNumber++);

            }
    return 0;
}
```

fork (): Example 2

- Download and Try Example 2