Multithreading Programming

-- by Wally

Background - Parallel Computing

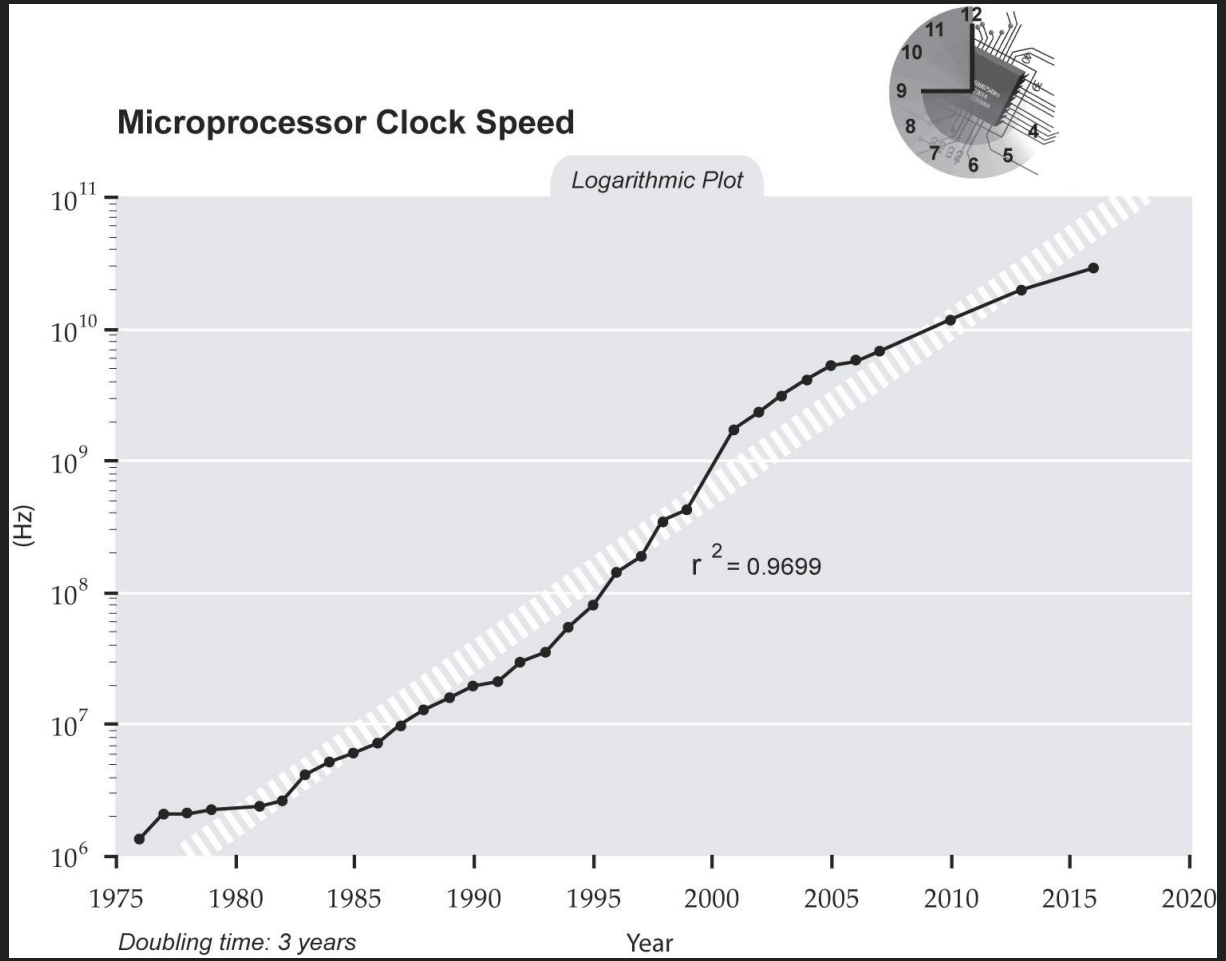
- What's Parallel Computing?
- Why Parallel Computing?
- Why Multithreading?

Parallel Computing

- "many calculations or the execution of processes are carried out simultaneously"

Concurrent Computing?

- "several computations are executed during overlapping time periods"


Microprocessor Clock Speed
Logarithmic Plot
10^{11}
10^{10}
10^{9}
10^{8}
10^{7}
10^{6}
(Hz)
$r^2 = 0.9699$
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
Doubling time: 3 years
Year

Parallel Computing

- "many calculations or the execution of processes are carried out simultaneously"

Concurrent Computing?

- "several computations are executed during overlapping time periods"

Parallel Computing

- Multi-Processing
- Multi-Threading
- GPU Computing (Massive Threading)
- Distributed Computing (Through Networks)

POSIX Thread API

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start_routine) (void *), void *arg);
```

- A Hello World Example

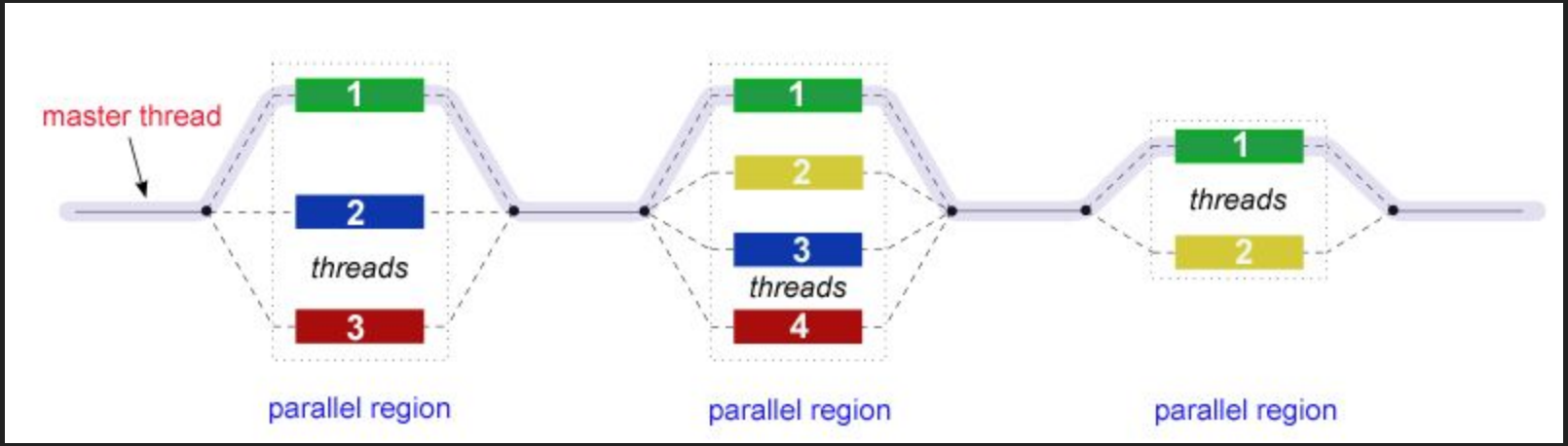
POSIX Thread API

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start_routine) (void *), void *arg);
```

```
int pthread_join(pthread_t thread, void **retval);
```

- A Revised Hello World Example

Fork-Join Model



Spawn Threads

- 32 Threads
- 10000 Threads

Data Race!

Mutual Exclusion! (Mutex)

POSIX Mutex API

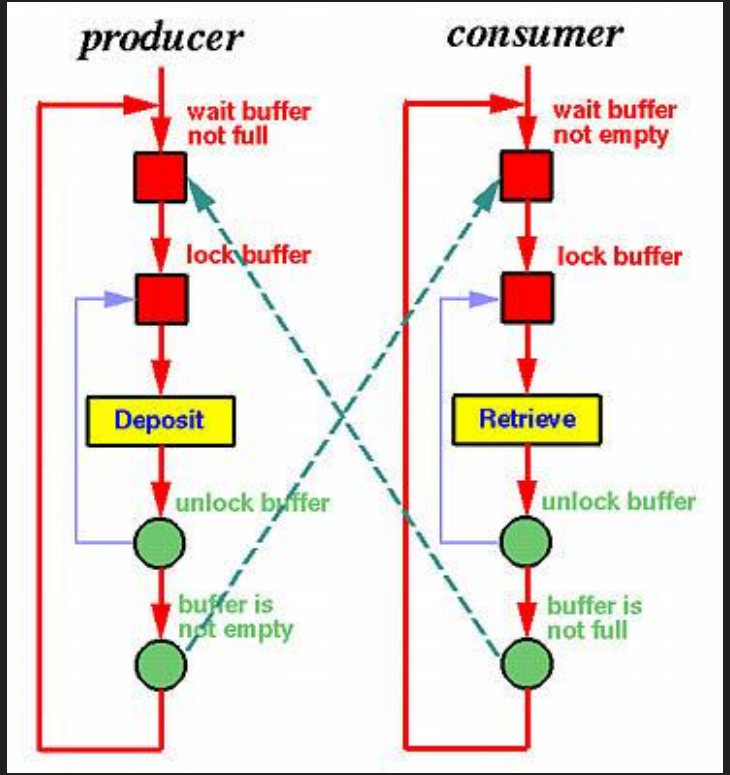
```
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
            const pthread_mutexattr_t *restrict attr);

int pthread_mutex_lock(pthread_mutex_t *mutex);

int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

- A Revised Addition Example

Producer-Consumer Problem



Producer-Consumer Problem

- A buffer (array)
- One or more producer threads put data into the buffer
- One or more consumer threads extract data out of the buffer
- Only one producer or consumer can modify the buffer at some time
- Producer should wait if the buffer is full
- Consumer should wait if the buffer is full

How to pause the thread?

Semaphore!

Semaphore

- An integer value
- `wait`: decrement the value by one, wait if value is negative
- `post`: increment the value by one, if there are waiting threads, wake one

POSIX Semaphore API

```
int sem_init(sem_t *sem, int pshared, unsigned int value);

int sem_post(sem_t *sem);

int sem_wait(sem_t *sem);
```

Producer Consumer Code Example

OpenMP
Simple Multi Threading

OpenMP Pragma

```
#include <omp.h>

#pragma omp parallel num_threads(4)

int omp_get_thread_num();
```

- OpenMP Hello World

Even Simpler: OpenMP `for`

```
#pragma omp parallel for
```

- Another OpenMP Hello World

OpenMP Mutual Exclusion: Critical

```
#pragma omp critical
```

- OpenMP Calculation

pthreads vs. OpenMP

- Why pthreads:
  - Maximum thread control
  - Flexible
  - Can be used with other POSIX API (pthread_mutex, condition variable, semaphore etc.)
- Why OpenMP:
  - Easy to use
  - Cross-platform
  - (Almost) no need to change the code

Thank you!