Resumen Scheduling Introduction - SistOp

Lautaro Bachmann

Contents

Scheduling: Introduction

Workload Assumptions

workload.

processes running in the system, are called the workload

fully-operational scheduling discipline

We will make the following assumptions about the processes, sometimes called jobs, that are running in the system:

1. Each job runs for the same amount of time.
2. All jobs arrive at the same time.
3. Once started, each job runs to completion.
4. All jobs only use the CPU (i.e., they perform no I/O)
5. The run-time of each job is known.

some assumptions are more unrealistic than others in this chapter.

Scheduling Metrics

What is a metric?

A metric is just something that we use to measure something, and there are a number of different metrics that make sense in scheduling.

turnaround time.

The turnaround time of a job is defined as the time at which the job completes minus the time at which the job arrived in the system.

You should note that turnaround time is a **performance** metric,

More formally

$$T_{turnaround} = T_{completion} - T_{arrival}$$

Definitions

convoy effect

Definition
where a number of relatively-short potential consumers of a resource get queued behind a heavyweight resource consumer.

Example
This scheduling scenario might remind you of a single line at a grocery store and what you feel like when you see the person in front of you with **three carts full of provisions**

non-preemptive schedulers

run each job to completion before considering whether to run a new job.

preemptive,

Virtually all modern schedulers are preemptive and quite willing to stop one process from running in order to run another.

two types of schedulers.

The first type (SJF, STCF)
optimizes turnaround time, but is bad for response time.

The second type (RR)
optimizes response time but is bad for turnaround.

First In, First Out (FIFO)

Behaviour

The name says it all. The first job that appears is executed first.

properties:

it is clearly simple and thus easy to implement.

Shortest Job First (SJF)

Behaviour

it runs the shortest job first, then the next shortest, and so on.

SJF performs much better with regards to average turnaround time than FIFO.

assume that jobs can arrive at any time

suffer the same convoy problem as FIFO.

Shortest Time-to-Completion First (STCF)

Behaviour

Any time a new job enters the system, the STCF scheduler determines which of the remaining jobs (including the new job) has the least time left, and schedules that one.

The result is a much-improved average turnaround time

A New Metric: Response Time

We define response time as the time from when the job arrives in a system to the first time it is scheduled.

More formally:

$$T_{response} = T_{firstrun} - T_{arrival}$$

STCF and related disciplines

are not particularly good for response time. While great for turnaround time, this approach is quite bad for response time and interactivity.

Round Robin

Round-Robin (RR) scheduling

The basic idea is simple: instead of running jobs to completion, RR runs a job for a **time slice** (sometimes called a **scheduling quantum**) and then switches to the next job in the run queue.

the length of a time slice

must be a multiple of the timer-interrupt period;

making the time slice too short

is problematic: suddenly the cost of context switching will dominate overall performance.

turnaround time?

RR is one of the **worst policies** if turnaround time is our metric. any policy (such as RR) that is **fair,** will perform poorly on metrics such as turnaround time.

Incorporating I/O

the currently-running job won't be using the CPU during the I/O; the scheduler should probably schedule another job on the CPU at that time.

how a scheduler might incorporate I/O.

treating each CPU burst as a job, the scheduler makes sure processes that are "interactive" get run frequently. While those interactive jobs are performing I/O, other CPU-intensive jobs run, thus better utilizing the processor.

Summary

We have still not solved the problem of the fundamental inability of the OS to see into the future. Shortly, we will see how to overcome this problem, by building a scheduler that uses the recent past to predict the future. This scheduler is known as the **multi-level feedback queue,** and it is the topic of the next chapter.