Scheduling

We now look at scheduling: how to choose which process to run next

Scheduling

We now look at scheduling: how to choose which process to run next

This is an *extremely* difficult problem and still has not been solved to everybody's satisfaction

Scheduling

A list of scheduling algorithms, from Wikipedia:

Scheduling

A list of scheduling algorithms, from Wikipedia:

Borrowed-Virtual-Time Scheduling (BVT), Completely Fair Scheduler (CFS), Critical Path Method of Scheduling, Deadline-monotonic scheduling (DMS), Deficit round robin (DRR), Dominant Sequence Clustering (DSC), Earliest deadline first scheduling (EDF), Elastic Round Robin, Fair-share scheduling, First In, First Out (FIFO), also known as First Come First Served (FCFS), Gang scheduling, Genetic Anticipatory, Highest response ratio next (HRRN), Interval scheduling, Last In, First Out (LIFO), Job Shop Scheduling (see Job shops), Least-connection scheduling, Least slack time scheduling (LST), List scheduling, Lottery Scheduling, Multilevel queue, Multilevel Feedback Queue, Never queue scheduling, $O(1)$ scheduler, Proportional Share Scheduling, Rate-monotonic scheduling (RMS), Round-robin scheduling (RR), Shortest expected delay scheduling, Shortest job next (SJN), Shortest remaining time (SRT), Staircase Deadline scheduler (SD), “Take” Scheduling, Two-level scheduling, Weighted fair queuing (WFQ), Weighted least-connection scheduling, Weighted round robin (WRR), Group Ratio Round-Robin: $O(1)$

Scheduling

And they are just the ones people can be bothered to write about on Wikipedia

Scheduling

Think of the problems

Scheduling

Think of the problems

- Try to give each process its fair share of CPU time

Scheduling

Think of the problems

- Try to give each process its fair share of CPU time
- and no starvation of any process

Scheduling

Think of the problems

- Try to give each process its fair share of CPU time
- and no starvation of any process
- Try to make interactive processes respond in human timescales

Scheduling

Think of the problems

- Try to give each process its fair share of CPU time
- and no starvation of any process
- Try to make interactive processes respond in human timescales
- Try to give as much computation time as possible to compute-heavy processes

Scheduling

Think of the problems

- Try to give each process its fair share of CPU time
- and no starvation of any process
- Try to make interactive processes respond in human timescales
- Try to give as much computation time as possible to compute-heavy processes
- Ensuring critical real-time processes are dealt with before it is too late

Scheduling

- Try to service peripherals in a timely way

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast
- Try to distribute work amongst multiple devices; e.g, CPUs and networks

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast
- Try to distribute work amongst multiple devices; e.g, CPUs and networks
- Try to make best use of the hardware and use it efficiently

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast
- Try to distribute work amongst multiple devices; e.g, CPUs and networks
- Try to make best use of the hardware and use it efficiently
- Try to make behaviour predictable: we don't want wildly erratic behaviour

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast
- Try to distribute work amongst multiple devices; e.g, CPUs and networks
- Try to make best use of the hardware and use it efficiently
- Try to make behaviour predictable: we don't want wildly erratic behaviour
- Try to degrade gracefully under heavy load

Scheduling

- Try to service peripherals in a timely way
- Understanding the various requirements of hardware: mice and printers are slow; networks and disks are medium; memory is fast
- Try to distribute work amongst multiple devices; e.g, CPUs and networks
- Try to make best use of the hardware and use it efficiently
- Try to make behaviour predictable: we don't want wildly erratic behaviour
- Try to degrade gracefully under heavy load
- And so on

Scheduling

And do it all quickly!

Scheduling

Here we shall concentrate on CPU scheduling, but remember the CPU is just one resource of many

Scheduling

Here we shall concentrate on CPU scheduling, but remember the CPU is just one resource of many

A related problem is *I/O scheduling*, managing requests and responses to other devices, such as disks, to make best use of them

Scheduling

Here we shall concentrate on CPU scheduling, but remember the CPU is just one resource of many

A related problem is *I/O scheduling*, managing requests and responses to other devices, such as disks, to make best use of them

I/O scheduling is important, but we shall not talk about it here

Scheduling

Here we shall concentrate on CPU scheduling, but remember the CPU is just one resource of many

A related problem is *I/O scheduling*, managing requests and responses to other devices, such as disks, to make best use of them

I/O scheduling is important, but we shall not talk about it here

But we will note in passing that the various schedulers for the various resources may not agree on what should be done next!

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

- CPU cycles used

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

- CPU cycles used
- Memory used

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

- CPU cycles used
- Memory used
- Disk used

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

- CPU cycles used
- Memory used
- Disk used
- Network used

Scheduling

All this needs to be quantified somehow so we can use the numbers to make choices

Example measurements include:

- CPU cycles used
- Memory used
- Disk used
- Network used
- Etc.

Scheduling

And we can quantify results

Scheduling

And we can quantify results

- Throughput; more or fewer jobs finished in a given time

Scheduling

And we can quantify results

- Throughput; more or fewer jobs finished in a given time
- Turnaround; response time: interactive response is snappy or sluggish

Scheduling

And we can quantify results

- Throughput; more or fewer jobs finished in a given time
- Turnaround; response time: interactive response is snappy or sluggish
- Real-time; we *must* deal with this data now else the car will crash (deadlines)

Scheduling

And we can quantify results

- Throughput; more or fewer jobs finished in a given time
- Turnaround; response time: interactive response is snappy or sluggish
- Real-time; we *must* deal with this data now else the car will crash (deadlines)
- Money; we've been given money to get this data ready in the next hour

Scheduling

And we can quantify results

- Throughput; more or fewer jobs finished in a given time
- Turnaround; response time: interactive response is snappy or sluggish
- Real-time; we *must* deal with this data now else the car will crash (deadlines)
- Money; we've been given money to get this data ready in the next hour
- Etc.

Scheduling

All this information was originally collected to figure out how much money to charge the user

Scheduling

All this information was originally collected to figure out how much money to charge the user

These days most people are not so worried about charging as we all have our own computers. We are more concerned about making the best use of our computer

Scheduling

All this information was originally collected to figure out how much money to charge the user

These days most people are not so worried about charging as we all have our own computers. We are more concerned about making the best use of our computer

Though it's still important: cloud services (e.g., Amazon, Google, Microsoft) sell time on their machines

Scheduling

All this information was originally collected to figure out how much money to charge the user

These days most people are not so worried about charging as we all have our own computers. We are more concerned about making the best use of our computer

Though it's still important: cloud services (e.g., Amazon, Google, Microsoft) sell time on their machines

They charge based on disk storage, data input and output and compute (CPU) used

Scheduling

All this information was originally collected to figure out how much money to charge the user

These days most people are not so worried about charging as we all have our own computers. We are more concerned about making the best use of our computer

Though it's still important: cloud services (e.g., Amazon, Google, Microsoft) sell time on their machines

They charge based on disk storage, data input and output and compute (CPU) used

There's nothing new in Computer Science: just recurring fashions!