Memory Scheduling

Chris Krawiec

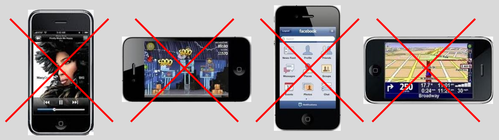
Department of Computer Science

Motivation

- Almost everyone has a smart phone or other mobile device
- We listen to music, surf Facebook, and more *all at once*
- These devices go everywhere with us, and we are constantly using them

Problem

- Our devices have *limited memory* and each application we run requires memory
- There is no good way to predict the memory demands an application will place on a device
- When a device runs out of memory applications may crash or perform poorly

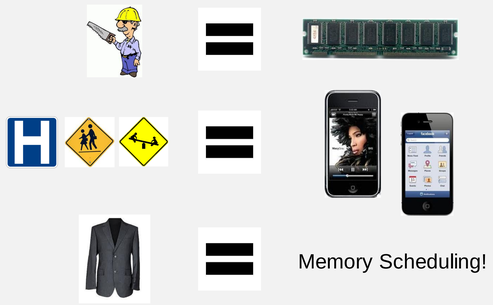


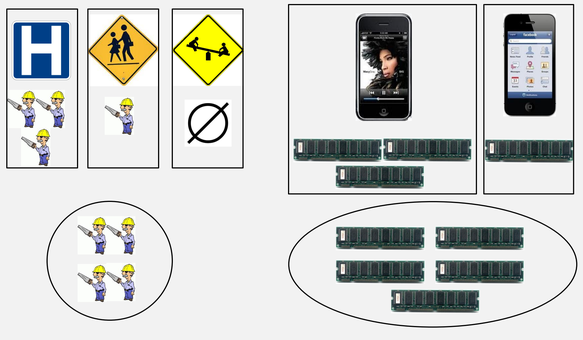
Solution

- *Shuffle* memory between applications
- Define a *policy* to control where and when to shuffle memory
- Memory Scheduling – shuffle memory where needed, when needed

An Analogy

- Imagine Bob's Builders, a construction company
- Bob's Builders has jobs to build a hospital, a school, and a playground
- CEO assigns limited worker pool to complete jobs efficiently, effectively, and on-time



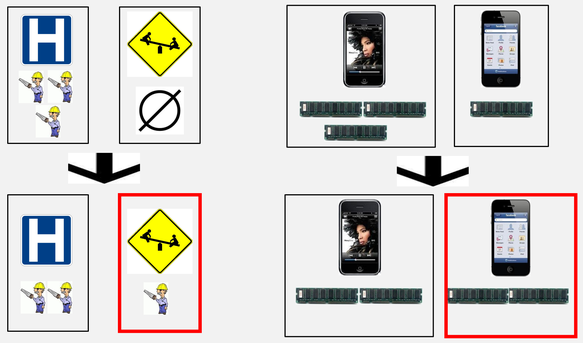


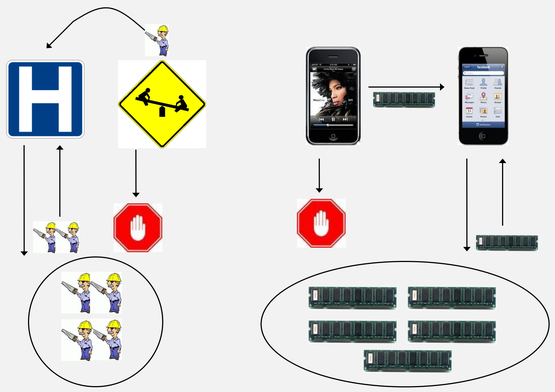
Perspective

- Need perspective to make effective decisions
- Might like to know details like:
  - Number and types of jobs
  - Workers assigned to each job
  - Number of unassigned and idle workers
- Memory Scheduling needs similar perspective

Shuffle

- Some workers actively working, some idle
- Shuffle idle worker from one job to another job to actively work where needed
- Total number of workers at Bob's Builders unchanged, but better utilization





Policy

- Decide where and when to assign workers
- Utilize pool, perspective, and shuffling
- Factor in characteristics of a job such as:
  - Importance of the job
  - Number of workers required for the job
  - Deadline of the job

Technical Details