Resource Management

COS 316 Lecture 13

Amit Levy
Administrativia
• Assignment 4: Object Relational Mapper
  • Released tomorrow (Nov. 5th)
  • Due two weeks from tomorrow (Nov. 19th)

• General rule: something due every Tuesday, alternating assignment and problem set
Term Paper: System Analysis

• Due on dean’s date

• Groups of 1 or 2 (groups highly encouraged)

• Topic proposals due after Thanksgiving break

• What is the paper about?
  
  1. Pick an open source system, or component of an open source system
  2. Describe it in detail using analytical tools from the course

• Expect ~10 pages, but no strict limit or minimum
Term Paper: Example Systems to Analyze

- Nix package manager
- Node Package Manager (NPM), Ruby gems, Rust crates, etc
- Kubernetes
- MapReduce, TensorFlow, Spark
- The Linux device driver API
- ActiveRecord from Ruby on Rails
- WordPress plugin system
Resource Management
• So far, we’ve talked about naming and caching as ways of making an application simpler to organize, more efficient, etc.

• Rest of the semester we’ll focus on design principles for mediating and managing multiple applications:
  • Resource management
  • Virtualization
  • Access Control
What is resource management?

Recall one of our systems criteria from first lecture: 
*Mediates access to shared resources*

Touched on this a little:

- Allocation in naming schemes are policies for sharing a namespace, names indicate unique partitions of a resource
- Eviction algorithms are policies for sharing a cache
What is resource management?

• Policies and layers of abstraction that expose a resource to multiple applications

• Policies and abstractions have important effect on:
  • Performance
  • Flexibility
  • Security
Key Characteristics

• Fixed or arbitrary number of applications
  • How many applications can the policy support? At what cost?

• Mandatory or cooperative sharing
  • Are applications trusted to yield the resource?

• Virtual or explicit sharing
  • Are applications aware they are using a shared resource?
Which shared resources are there?

Which resources are shared by controllers or request handlers in a web application?

- CPU
- Memory
- Files, database, disk
- Local network controller
- Network and Internet links
Which shared resources are there?

Which resources are shared by controllers or request handlers in a web application?

- CPU
- Memory
- Files, database, disk
- Local network controller
- Network and Internet links
Example: sharing the CPU

Figure 1: How do we run more applications than cores?
Three Policies for Sharing the CPU

1. Cooperative scheduling
2. Static scheduling
3. Round-robin timeslice scheduling
Figure 2: Cooperative Scheduling
Cooperative Scheduling

• Fixed number of applications
  • Apps have to *know* who they are yielding to

• Cooperative sharing
  • An application *could* never yield

• Explicit sharing
  • Applications must be written with sharing in mind
Figure 3: Static Scheduling
Static Scheduling

• Fixed number of applications
  • Scheduler subdivides time into fixed number of slots

• Mandatory sharing
  • Applications preempted when their timeslice is up

• Virtual sharing
  • Applications can be written as though they have exclusive access
Figure 4: Round-Robin Timesliced Scheduling
Round-Robin Timesliced Scheduling

- Arbitrary number of applications
  - Just add more applications to the queue

- Mandatory & Cooperative sharing
  - Applications preempted when their timeslice is up
  - Applications may also yield via I/O or sleep operations

- Virtual sharing
  - Applications can be written as though they have exclusive access
Performance Comparison

- **Cooperative sharing**
  - CPU never has to do extra work for spurious context switches
  - Applications switch when most convenient for performance

- **Mandatory sharing**
  - Each application guaranteed some base level of performance
  - Wasteful if application doesn’t use entire timeslice, or if context switches are frequent
Flexibility Comparison

• Fixed number of applications clearly less flexible than arbitrary number

• Virtual sharing means scheduling policy can be replaced under the hood

• Cooperative sharing allows applications to use slightly more or slightly less time on CPU
Applications can learn secrets from other applications

```go
func application1(secret string) {
    if secret == "password" {
        spin_for_10_seconds()
    }
    yield()
}
```
Most systems run applications that share lots of resources

System defines abstractions and policies for sharing resources

Key characteristics:
- Fixed vs arbitrary number of applications
- Mandatory vs cooperative sharing
- Virtual vs explicit sharing

Resource management policies affect performance, flexibility, security
1. Wednesday: Network Layers (Prof. Rexford)
3. Wednesday: Congestion Control (Prof. Freedman)