Implementing Resource Containers in K42

Reza Azimi
Adrian Tam
David Tam
Introduction / Motivation

- **OS Resources:**
  - CPU cycles, mem pages, I/O bandwidth
- **Problem:** usage not accurately accounted
  - incorrect resource allocations
    - e.g.
      - interrupt handling
      - kernel services
      - IPCs
Existing Solutions

- **No sharing**
  - buy more hardware
  - too much isolation

- **Virtual machines**
  - still too much isolation

- **Other techniques**
  - lazy receiver processing (LRP)
  - resource containers

→ Want *isolation* but also want *sharing*
  - accountable
  - fine-grained
Resource Containers

- **Definition:**
  - specifies limits
  - records consumption
  - encapsulates resource usage of a task
  - crosses all boundaries

- **Properties:**
  - all user & kernel level processing charged to container
  - runs with associated limits

- **Our Model:**
  - User-based rather than process-based
  - API:
    - `rcid = create(% of cpu, max # pages);`
    - `bind(rcid);`

```
Resource Container
ID = _____
CPU % = _____
max # pages = _____
etc...
current CPU % = _____
current # pages = _____
etc...
```
Resource Containers

Basic Mechanisms
2. RC – container object
3. IPC – RC transfer points
   implicit vs explicit transfers
4. scheduler – enforce scheduling priority
5. memory manager – enforce memory limits

Why not present in current OSes?
Major overhaul – every kernel object & operation must be:
   1. billed
   2. constrained
K42 Operating System

- Scalability
  - Shared Memory MPs
  - NUMA Architecture
  - 64-bit OS
- Object-Oriented
  - Modularity
  - Customizability
  - Uniformity
  - Scalability
- Micro-kernel based
  - Customizability
  - Performance
- Linux Compatibility
  - API Compatible
  - Binary Compatible
K42 Scheduling

Address Space

User Space

Kernel Space

CPU Domain

Dispatcher Descriptor

Scheduling Class 1

Scheduling Class 2
CPU Domain Scheduling in K42

Process 1

CPU Domain 1
(30%)

Process 2

CPU Domain 2
(20%)

PPC
CPU Domain Scheduling in K42

Process 1

CPU Domain 1 (30%)

Process 2

CPU Domain 2 (20%)

PPC
CPU Domain Scheduling in K42

Process 1

CPU Domain 1 (30%)

Process 2

CPU Domain 2 (20%)
CPU Domain Scheduling in K42

Process 1
CPU Domain 1

Process 2
CPU Domain 2

Process 3
CPU Domain 3

PPC
CPU Domain Scheduling in K42
Memory Management

Resource 1

Cur: 90
Max: 90

Page X
Memory Management

Resource 1
Cur: 25
Max: 90

Resource 2
Cur: 40
Max: 60

Page X
K42 Memory Management

Logical Pages

Region

FCM

PM

FR

Process

HAT

Region

FCM

FR
K42 Memory Management

- **RC**
- **Logical Pages**
- **Process**
- **Region**
- **FCM**
- **PM**
- **FR**
- **HAT**
## Progress And Plan

<table>
<thead>
<tr>
<th>Component</th>
<th>Design &amp; Implement</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure</td>
<td>Yes</td>
<td>Part</td>
</tr>
<tr>
<td>CPU Schedule</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Memory</td>
<td>Account: Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Enforcement: Part</td>
<td></td>
</tr>
</tbody>
</table>

### Milestones

<table>
<thead>
<tr>
<th>Milestones</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Accounting</td>
<td>April 9</td>
</tr>
<tr>
<td>Test Enforcement</td>
<td>April 23</td>
</tr>
</tbody>
</table>