Isolation with Flexibility: 
A Resource Management Framework for Central Servers

David G. Sullivan and Margo I. Seltzer
VINO Research Group
Harvard University, Cambridge, MA
{sullivan,margo}@eecs.harvard.edu
Resource Management on Central Servers

• Users are increasingly competing for the resources of central servers.
  – virtually hosted Web sites
  – centralized databases
  – thin-client computing

• Resource management goals:
  – provide resource principals with resource shares that reflect their relative importance
  – meet applications’ differing resource needs
Lottery Scheduling Framework [Waldspurger & Weihl]

- Tickets encapsulate resource rights.
  - Proportional-share approach

- Currencies issue tickets.
  - Use to group and isolate resource principals
Secure Isolation vs. Flexible Allocation

- The resource shares protected by isolation may not correspond to the actual needs of applications.
Secure Isolation vs. Flexible Allocation

- The resource shares protected by isolation may not correspond to the actual needs of applications.

- **Ideal:** give resource principals the flexibility to safely adjust their own allocations while preserving secure isolation.
Our Extended Framework

• Increased flexibility in adjusting resource rights
• Multiple resources
• Access controls
• Hard and soft resource shares
Talk Outline

• Problem description
• **Extended lottery-scheduling framework**
  – securely managing multiple resources
  – isolation with increased flexibility
• Prototype implementation
• Performance results
• Conclusions
Securely Managing Multiple Resources

• Resource-specific tickets
  – CPU tickets, disk tickets, etc.

• Access controls
  – encapsulated in a broker associated with each currency
  – A currency’s mode, like a UNIX file mode, specifies who may perform various operations on it.

• Soft and hard resource shares
  – soft: A receive twice the share of B.
  – hard: C should receive 20% of the resource.
Flexible Allocation vs. Secure Isolation

• Currencies impose both upper *and* lower limits on resource allocations.

• Other resource-management frameworks impose similar limits through currency-like abstractions.
  – Rialto’s *activities* [Jones et al., 1997]
  – Eclipse’s *reservation domains* [Bruno et al., 1998]
  – *Software Performance Units* [Vergheese et al., 1998]
  – *Resource containers* [Banga et al., 1999]
Problem: Currencies Impose Upper Limits

- Essential to providing isolation
- May be unnecessarily restrictive
Solution: Ticket Exchanges

- Allow applications to safely modify their resource rights
  - Other principals’ resource rights are not affected.

- Take advantage of applications’ differing resource needs

```
I/O hog

  cpu tickets: 150 50 200
  disk tickets: 200 50 150

CPU hog
```
Carrying Out an Exchange

- Problem:
  - Exchanged tickets should have a fixed base value.
  - The value of subcurrency tickets can change.
Carrying Out an Exchange

• Need to use base-currency tickets.
  But how can we remove the tickets that are traded away?

```plaintext
    base
   / | \
  /  |  \
50 200 200 200
  |   |   \\
 margo  dave  \\
 100 100 100
  |   |
 task1 task2
```

memory tickets
disk tickets
Carrying Out an Exchange

- Solution: use *negative* tickets that reduce a principal’s base value.
Problem: Currencies Impose Lower Limits

- Difficult to support the semantics of nice

- hog can still end up with all of my resource rights.
Solution: Transfer Resource Rights

- Employ the same ticket-exchange system call.

- See Petrou et al., 1999 for an alternate solution.
Talk Outline

- Problem description
- Extended lottery-scheduling framework
  - securely managing multiple resources
  - isolation with increased flexibility
- Prototype implementation
- Performance results
- Conclusions
Extended Framework in VINO

- Full support for tickets and currencies
- System-call interface and utilities for creating currencies, funding them, unfunding them, etc.
- One currency per user
  - maintain a mapping from user id to currency id
  - re-fund process when it changes its real uid
Managing Multiple Resources

- **CPU Time**
  - original randomized lottery algorithm
  - compensation tickets and ticket transfers

- **Disk Bandwidth**
  - YFQ algorithm (Bruno et al., 1999)
  - similar to weighted fair queuing

- **Memory (limited solution)**
  - only give memory tickets to privileged processes that explicitly request them
  - pageout daemon skips pages owned by processes with less than their guaranteed shares
Ticket Exchanges: CPU and Disk

- Each program starts with 1000 tickets per resource.
- Also run one extra cpuhog and four extra iohogs.
Resource Rights Are Preserved

![Graph showing the percentage of CPU and disk bandwidth usage with bars representing the number of tickets exchanged. Left bars indicate the exchanging copy, right bars indicate non-exchanging.](image)
Ticket Exchanges: Memory and Disk

- **small**: 4-MB database (70,000 entries)
  
- **big**: 64-MB database \(2^{20}\) entries

- Limit memory (11.1 MB for users) and run four iohogs
At start: mem. tickets worth 1375 and disk tickets worth 1667. *Small* proposes exchange after 10,000 queries.
Trading 400 Memory Tickets from \textit{Big} to \textit{Small}

At start: mem. tickets worth 1375 and disk tickets worth 1667. \textit{Small} proposes exchange after 10,000 queries.
Conclusions

- We *can* provide isolation with greater flexibility.
- The best resource allocations for an application depend on the activity of the applications with which it is competing.
- Applications can achieve performance improvements by taking advantage of their differing resource needs.