VirginiaTech
Invent the Future

CS 5204 Operating Systems

Resource Containers: A new facility for resource management in server systems

G. Banga, P. Druschel, Rice Univ.
J. C. Mogul, Compaq

OSDI 1999

Outline

- Background
- Previous Approaches & Problems
- Motivation
- The Novel Idea: Resource Containers
- Performance & Evaluation
- Conclusion
- Current Status of R.C.

HTTP Servers …

Background

- Many users' perceived computing performance is based on the capacity of the remote servers. The underlying OS & hardware are hard to provide specific concerns on web browsing.

- Servers can accomplish different tasks each consuming different types of resources. However, depending on the resource allocation mechanisms of the systems it is hard to achieve QoS, fairness, etc. for clients.

Terms …

- Resource Principal:
  - entities for which separate resource allocation and accounting are done. So resource principals are the units at whose granularity resource scheduling is done.

- Protection Domain:
  - entities that need to be isolated from each other.

Traditional Process Abstraction (a dual function)

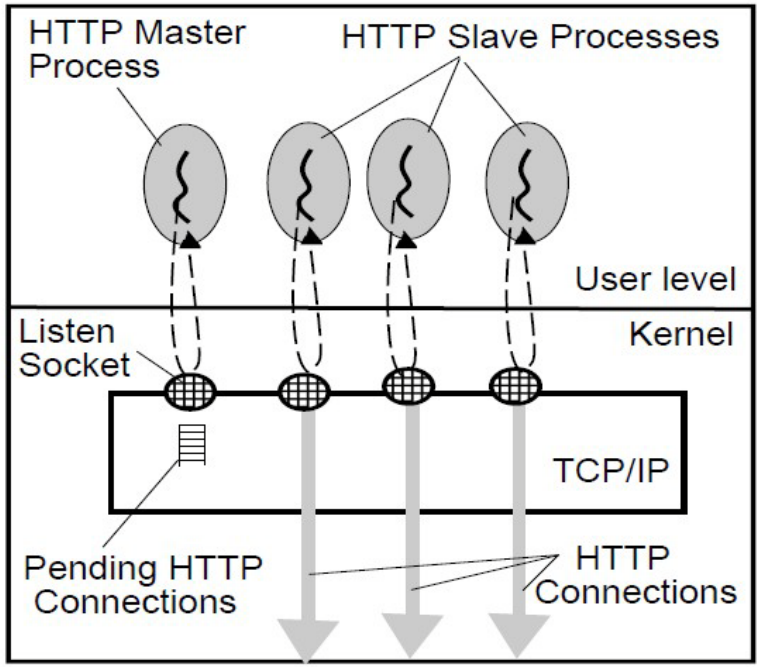
Background

- **Protection domain** and **Resource principal** coincide in the **process abstraction**.
- Do **NOT** allow process to **directly control** resource consumption of its kernel part.
- **Process** is what constitutes an **independent activity.**

Previous Approaches

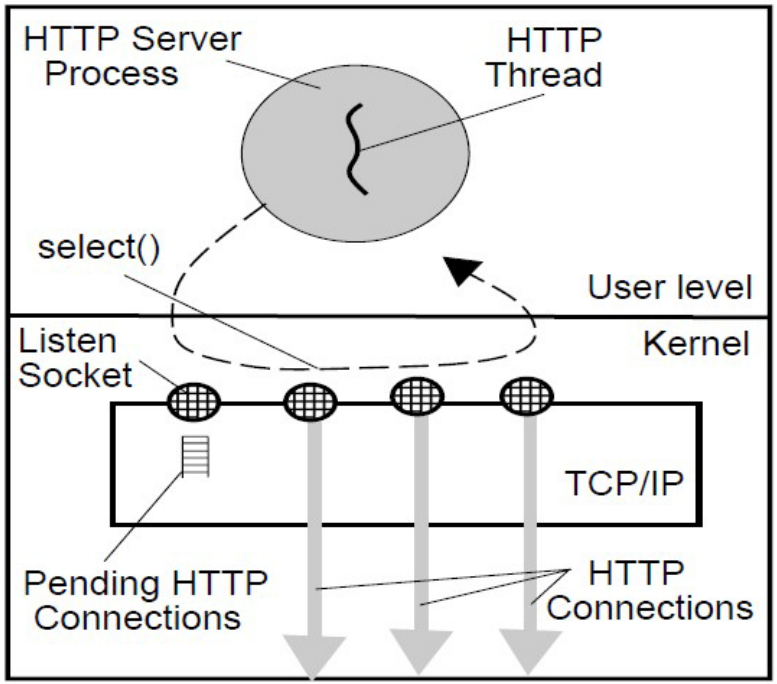
Process-per Connection …

- A master process listens on port for new connection requests
- For each new connection a new process is forked
- Drawbacks:
  - Forking overhead
  - Suffers from context switch
  - IPC



Single-Process Event-Driven Server …

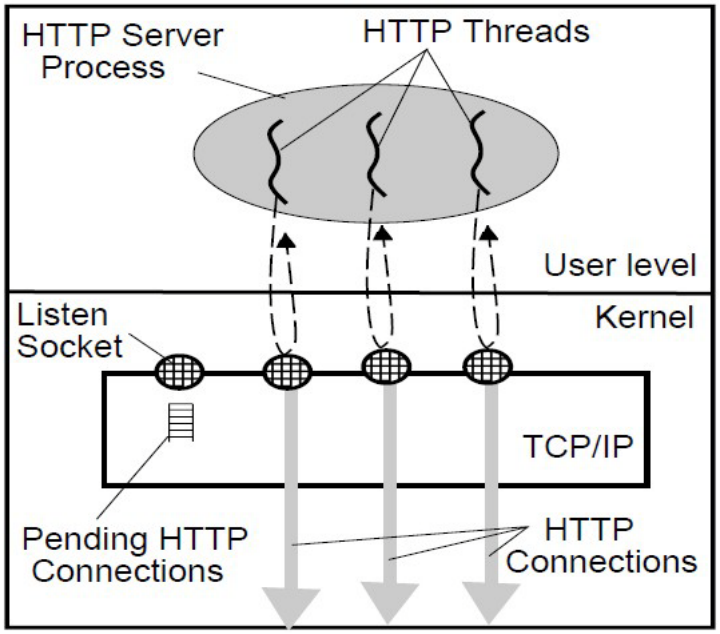
- Single process runs handlers in main loop for each ready connection
- Avoid IPC & context switches
- Drawbacks:
  - Not really concurrent
  - But can fork multi-processes if on multi-processor system



Previous Approaches

Multi-threaded Server …

- Each connection gets its own thread
- Threads are scheduled by thread scheduler
- Idle threads listen for next connection
- Avoids context switches and scales better

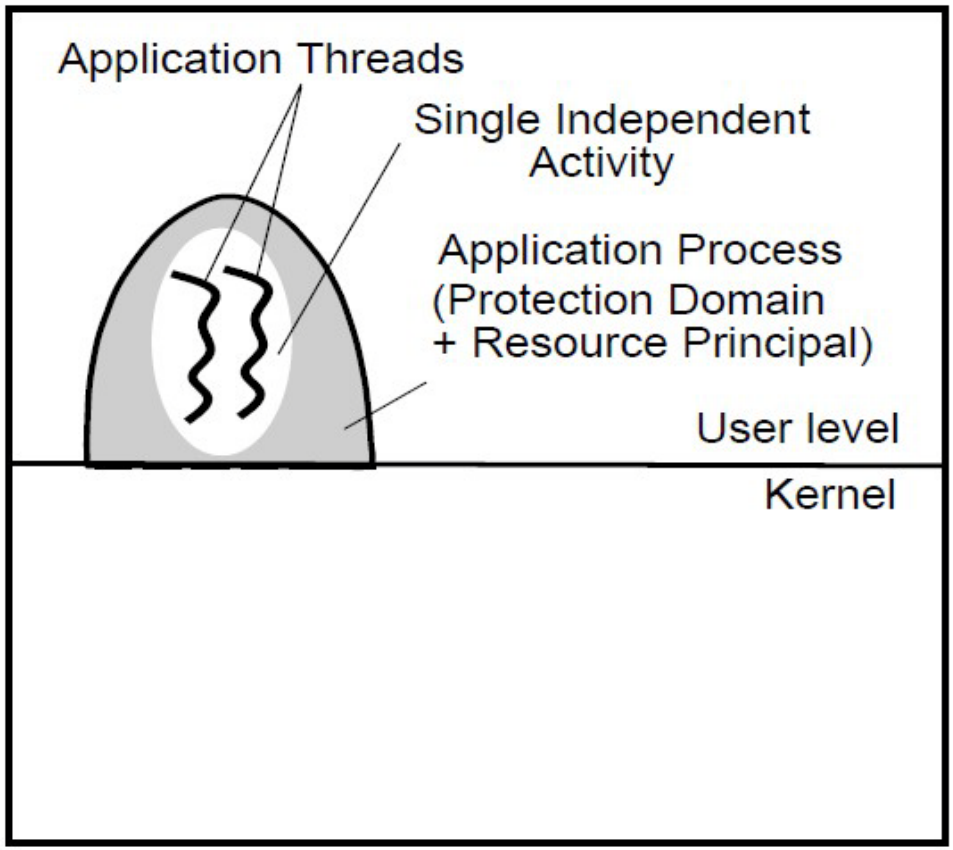


Other resources …

- Dynamic Resources:
  - Such as pages created in response to user input
  - This usually results in another process being created to handle the dynamic request
- Kernel Resources:
  - Kernel does network processing
  - Buffers, sockets, etc.
  - Separate from server app and charged to either one or any unlucky process!

General Assumptions of Servers

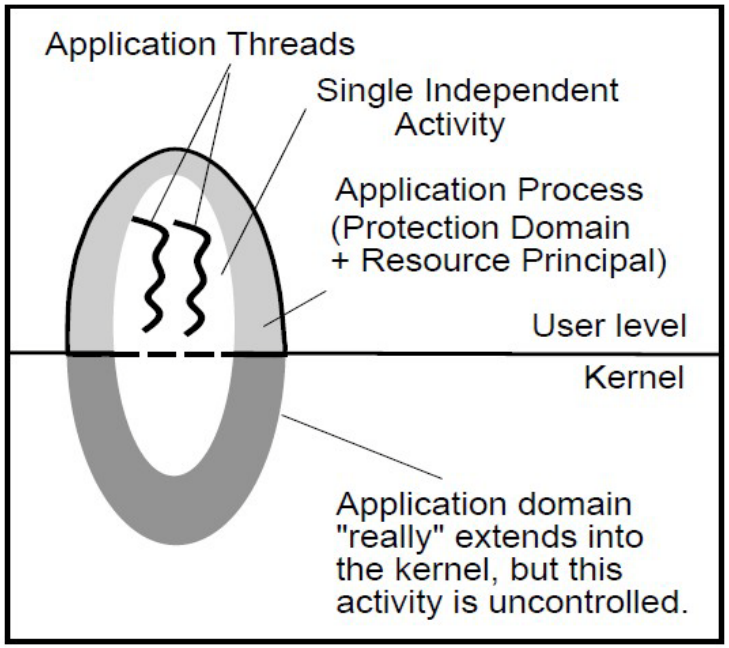
Problems



A network-intensive app.

Problems

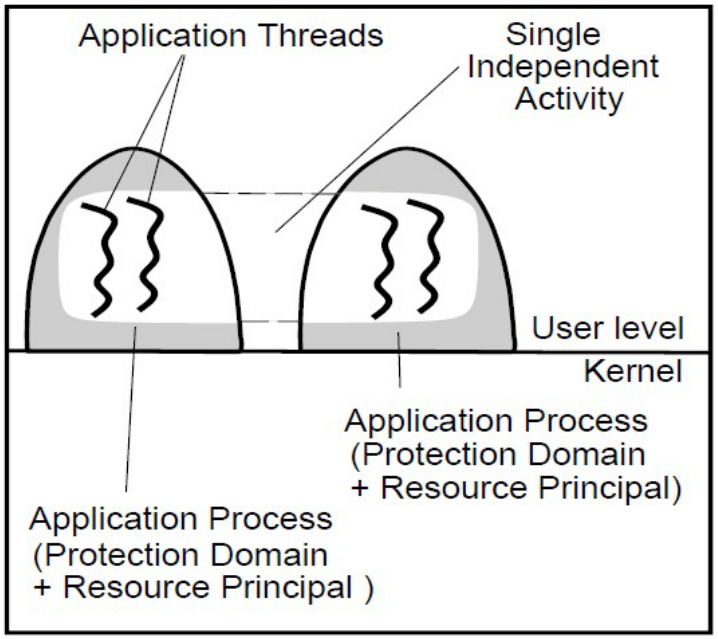
- The resources consumed by the kernel are unaccounted. i.e. The process is the right unit for protection, but it does not encompass all the resource consumption being done for the application



A multi-process app.

Problems

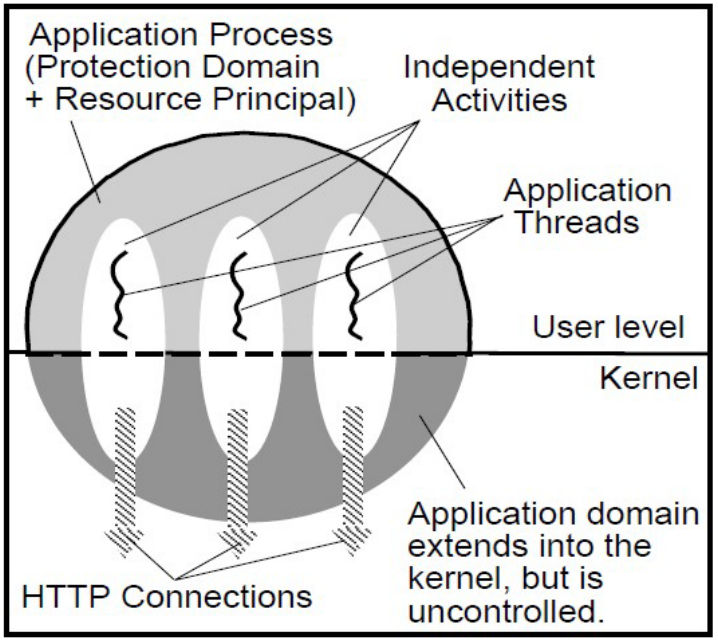
- An application composed of multiple user space processes, which are cooperating to perform a single activity. So the unit of resource management is set of all the processes rather than individual process



A single-process multi-threaded server

Problems

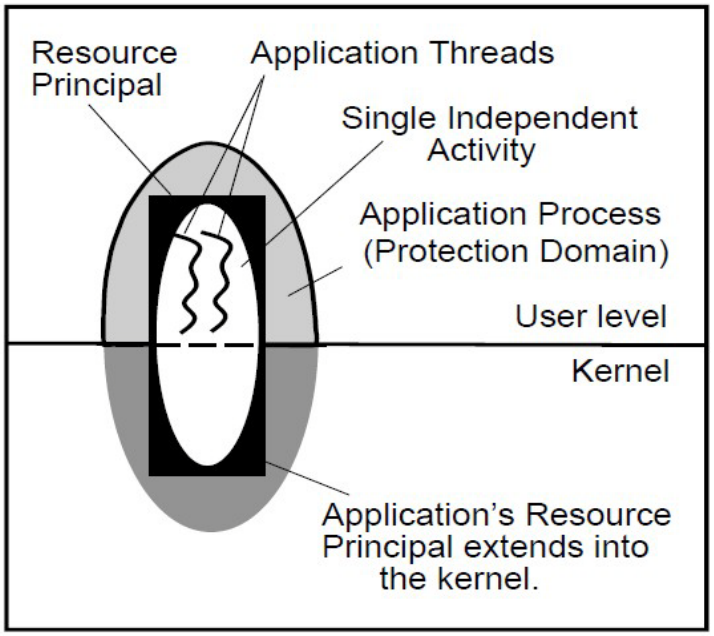
- The process is using multiple independent threads, one for each connection. The correct unit of resource management is smaller than a process, It is the set of all resources used to accomplish a single independent activity



Integrating network processing with resource management

Problems

- **L**azy **R**eceiver **P**rocessing
- Maintains equivalence between resource principal & process
- However, still associates protection domain and resource (i.e. an equivalence in between)



A good story to tell …

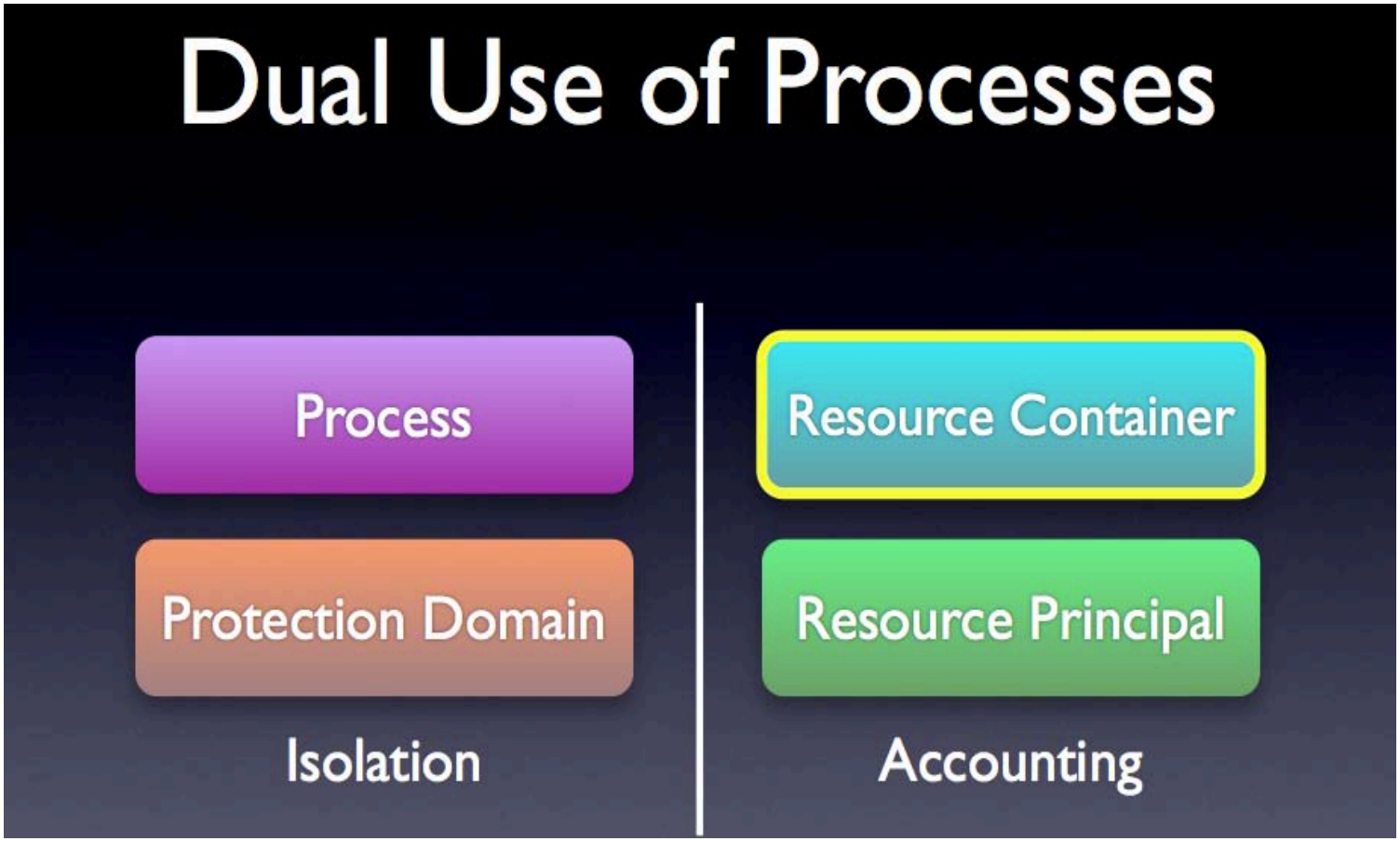
Motivation

- Dual function – protection domain and resource principal coincidence is not a good idea:
  - System does not allow app to directly control resource consumption, e.g. via priority
  - App has no control over resource management that is performed by kernel on behalf of app
- Web servers should be able to provide some kind of guarantee to clients, accurately accounting for the resources consumed

Resource Containers!!!

The Novel Idea

- "An abstract OS entity that logically contains all system resources being used by an app to achieve a particular independent activity"
  - Resources:
    - CPU, mem, socket, buffer
  - Attributes:
    - Scheduling Para, resource limits, network QoS values

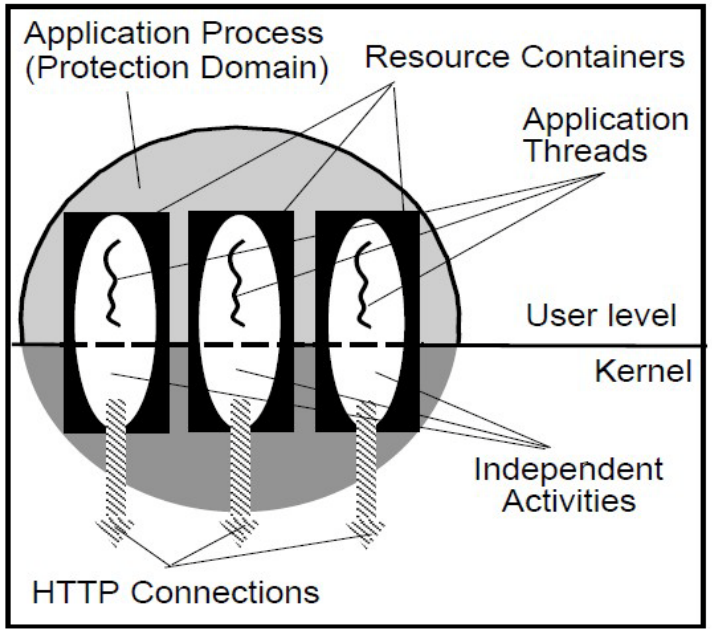


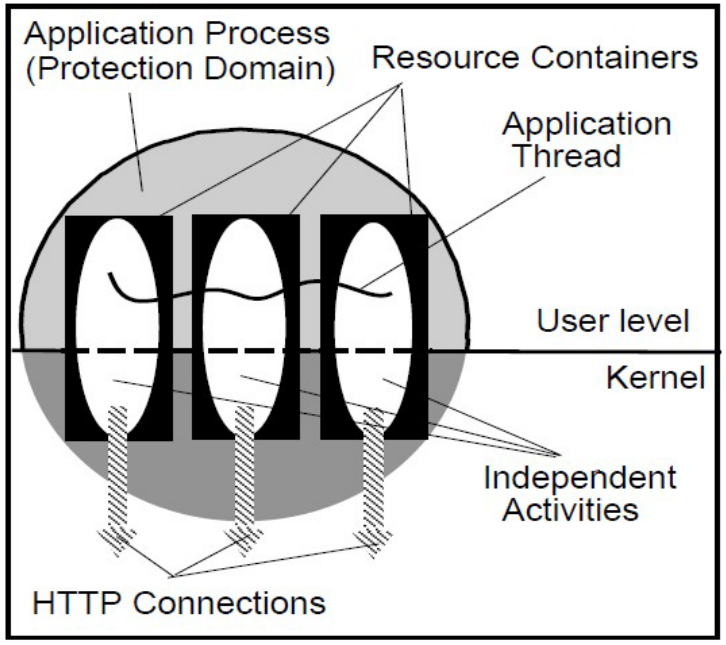
Resources

- Processes / Threads
  - Resource binding is the relation between resource / processing domains and the associated resource principals, effectively decoupling the two
  - Charge resources within kernel like LRP
  - Dynamic resource binding: based on the activity or purpose they are serving

Scheduling

- Threads may serve one container or many, existing within the same protection domain
- To avoid rescheduling threads after every resource container binding, a list of containers is associated with a thread and the thread is scheduled based on combined attributes
- Threads initially inherit their parents' container



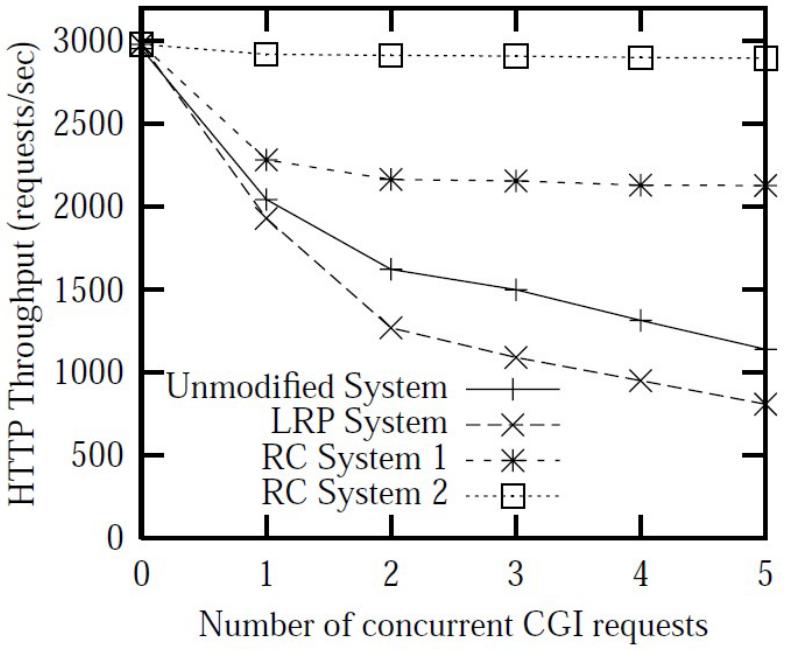


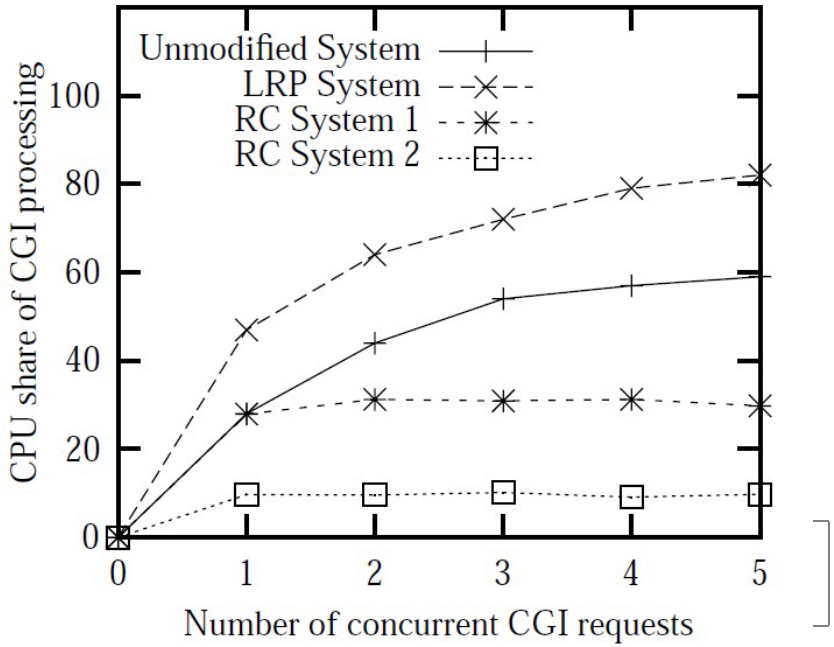
Implementation

- Modifications to Digital UNIX 4.0D
  - Changes to the CPU scheduler to treat resource containers as the resource principals.
  - A resource container can obtain a fixed share over a time scale of several seconds, or it can choose to time-share the resources assigned to its parent container with its sibling containers. (scheduling algorithm used ?)
  - TCP/IP subsystem modified to implement LRP
- Server software: single-process, event-driven
- Clients used the S-Client software

Isolation of separate activities

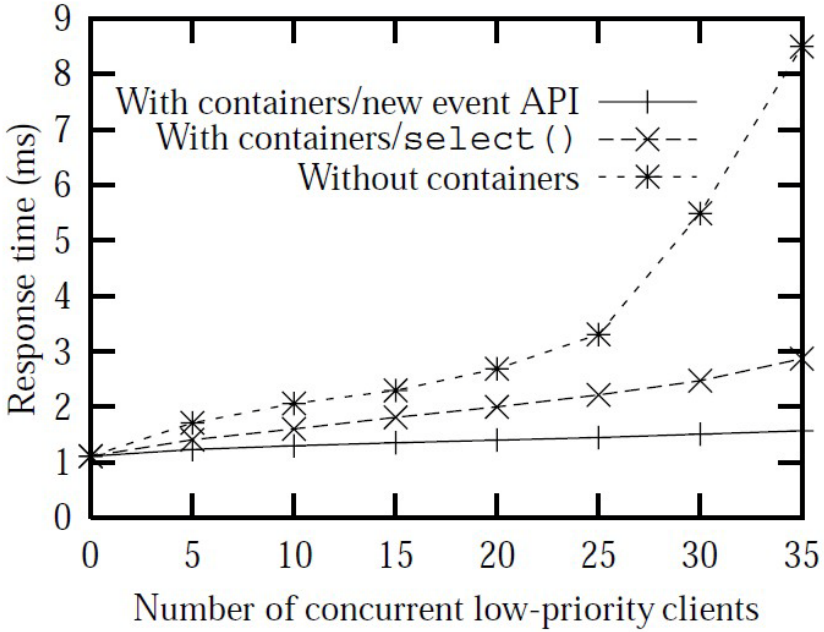
- Can static requests maintain throughput under pressure of many dynamic requests?
- Constrain the resource usage of dynamic requests





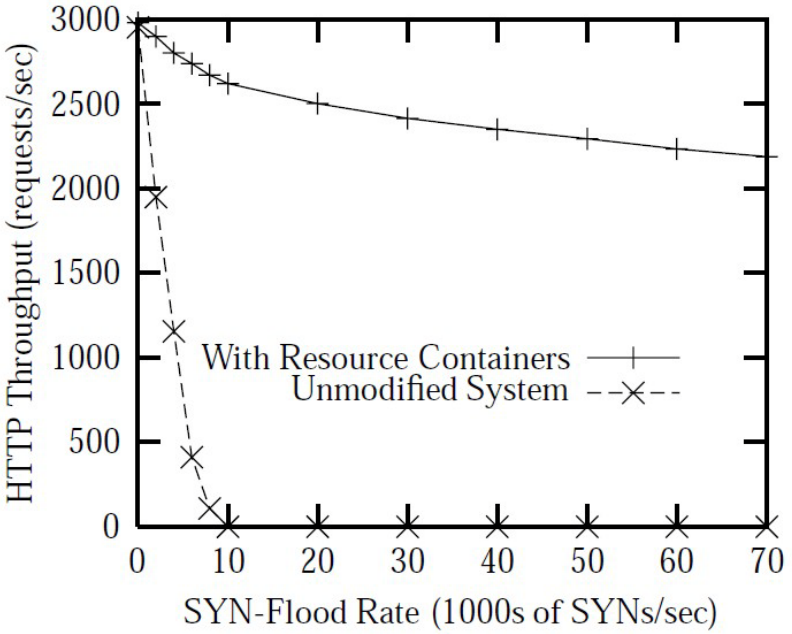
Priority based scheduling

- Assign resource container ‘T-high’ to high priority connections
- How does ‘T-high’ response time change with increase in low priority connections



Protecting against SYN flooding

- They had a set of known mis-behaving clients SYN flooding attach to the server
- Measured the concurrent throughput to other clients



Conclusion

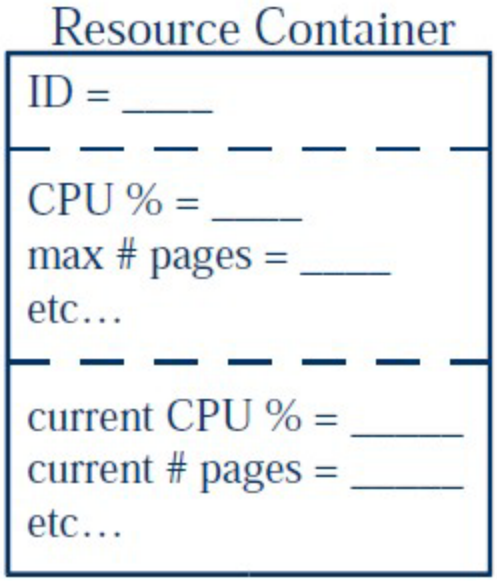
- Resource containers decouple resource principals from protection domains and allow explicit and fine-grained control over resource consumption at both user-level and kernel-level in the system

- Combined with accurate resource accounting can help web servers provide differentiated QoS

Current Status

Since R.C. was the work done 12 yrs ago, there are so many subsequent implementation work

- Resource Containers in K42 (2003)
  - User-based rather than process-based
  - API
    - rcid = create(# of CPU, max # pages)
    - bind(rcid)



Current Status

Since R.C. was the work done 12 yrs ago, there are so many subsequent implementation work …

- FreeBSD Foundation announces Resource Containers Project (2010)
  - Goal:
    - create a single, unified framework for controlling resource utilization API
    - to use that framework to implement per-jail resource limits

Thank you!

Any Questions?