Why Study Multimedia?

- Improvements:
  - Telecommunications
  - Environments
  - Communication
  - Fun
- Outgrowth from industry
  - telecommunications
  - consumer electronics
  - television

Continuous Media

- Subset of multimedia
- Includes timing relationship between server and client
- Stream:
  - video: mpeg, H.261, avi, QuickTime, MediaPlayer
  - audio: MP3, µ-law

Multimedia Resource Requirements

- Step up in media requires more bytes
- But not as much as some applications
  - Graphics or transaction processing

Influences on Quality

An End-To-End Problem
Application Performance in the QLinux Multimedia Operating System
Sundaram, A. Chandra, P. Goyal, P. Shenoy, J. Sahni and H. Vin
UMass Amherst, U of Texas Austin
In Proceedings of ACM Multimedia Conference November 2000

Introduction

- General purpose operating systems handling diverse set of tasks
  - Conventional best-effort with low response time
  - Throughput intensive applications
  - Soft real-time applications
- Many studies show can do one at a time, but when do two or more grossly inadequate
  - MPEG-2 when compiling has a lot of jitter

Reason? Lack of service differentiation
- Provide ‘best-effort’ to all
- Special-purpose operating systems are similarly inadequate for other mixes
- Need OS that:
  - Multiplexes resources in a predictable manner
  - Service differentiation to meet individual application requirements

Solution: QLinux

- Enhance standard Linux
- Hierarchical schedulers
  + classes of applications or individual applications
  - CPU, Network, Disk

Outline

- QLinux philosophy
- CPU Scheduler
  - Evaluation
- List of other topics in paper
  - Packet Scheduler
  - Disk Scheduler
  - Lazy Receiver Processing
- Conclusion

QLinux Design Principles

- Support for Multiple Service Classes
  - Interactive, Throughput-Intensive, Soft Real-time
- Predictable Resource Allocation
  - Priority not enough (starvation of others)
  - Ex: mpeg_decoder at highest can starve kernel
  - Not static partitioning since unused can be used by others
QLinux Design Principles

- Service Differentiation
  - Within a class, applications treated differently
  - Uses hierarchical schedulers
- Support for Legacy Applications
  - Support binaries of all existing applications (no special system calls required)
  - No worse performance (but may be better)

QLinux Components

Hierarchical Start-time Fair Queuing (H-SFQ) CPU Scheduler

- Uses a tree
- Each thread belongs to 1 leaf
- Each leaf is an application class
- Weights are of parent class

![Hierarchical Start-time Fair Queuing](image)

CPU Scheduler System Calls

- Nodes can be created on the fly
- Processes can move from node to node
- Defaults to top-level fair scheduler if not specified
- Utilities to do external from application
  - Allow support of legacy apps without modifying source

### Experimental Setup

- Cluster of PCs
  - P2-350 MHz
  - 64 MB RAM
  - RedHat 6.1
  - QLinux based on Linux 2.2.0
- Network
  - 100 Mb/s 3-Com Ethernet
  - 3Com Superstack II switch (100 Mb/s)
- “Assume” machines and net lightly loaded

### Experimental Workloads

- Inf: executes infinite loop
  - Compute-intensive, Best effort
- Mpeg_play: Berkeley MPEG-1 decoder
  - Compute-intensive, Soft real-time
- Apache Web Server and Client
  - I/O intensive, Best effort
- Streaming media server
  - I/O intensive, Soft real-time
- Dhrystone: measure CPU performance
  - Compute-intensive, Best effort
CPU Scheduler Evaluation-1

- Two classes, run $Inf$ for each
- Assign weights to each (ex: 1:1, 1:2, 1:4)
- Count the number of loops

CPU Scheduler Evaluation-1 Results

"count" is proportional to CPU bandwidth allocated

CPU Scheduler Evaluation-2

- Two classes, equal weights (1:1)
- Run two $Inf$
- Suspend one at $t=250$ seconds
- Restart at $t=330$ seconds
- Note count

CPU Scheduler Evaluation-2 Results

(Counts twice as fast when other suspended)

CPU Scheduler Evaluation-3

- Two classes: soft real-time & best effort (1:1)
- Run:
  - $MPEG\_PLAY$ in real-time (1.49 Mbps)
  - $Dhrystone$ in best effort
- Increase Dhrystone’s from 1 to 2 to 3 …
  - Note MPEG bandwidth
- Re-run experiment with Vanilla Linux

CPU Scheduler Evaluation-3 Results
CPU Scheduler Evaluation-4

- Explore another best-effort case
- Run two Web servers (representing, say 2 different domains)
- Have clients generate many requests
- See if CPU bandwidth allocation is proportional

CPU Scheduler Evaluation-4 Results

- Application Isolation for Web Servers
- CPU Scheduler Overhead Evaluation
  - Scheduler takes some overhead since recursively called
  - Run Inf at increasing depth in scheduler hierarchy tree
  - Record count for 300 seconds

CPU Scheduler Overhead Evaluation Results

QLinux Components

Disk
- Not evaluated

Packets
- Sending and “Lazy” Processing for Receiving

Conclusion

- Some improvement and some ideas
- Still Much work to be done
  - scheduling
  - memory management
  - network
  - disk
- M.S. Thesis
  - One piece in OS support puzzle