local tickets for every thread/process

* global tickets for users

...une-grained assignment of proportions...

33.33%

50.00%

25.00%

33.33%

33.33%

Time Sharing vs. Proportional Share Scheduling

CPU usage w/ proportional share schedulers

CPU usage w/ default (time sharing) schedulers
more computation intensive
+ next pick predictable

Stride Scheduling •

+ easier to implement

number of tickets held
chance of being picked is probabilistic and proportionate to

"instantaneously fair"
 random pick a ticket and hereby the thread

Lottery Scheduling •

Lottery vs. Stride Scheduling:

Proportional Share Scheduling
- low-priority threads may starve
- (in same priority/class) every thread is treated the same
- TS Time Sharing - lowest priorities
IA Interactive - e.g., child processes of window manager
SYS System - e.g., pager daemon
RT Real-Time - highest priorities
+ multiple scheduling classes
+ fast, efficient, "simple data structures"
- round robin with priority queues

Current Solaris Scheduling Schemes
Kernel Internals: Dispatch Queue

- Dispatcher does load balancing over CPUs
- One for each CPU, one for the kernel (undispatched threads)
- Holds linked lists of runnable threads for all priorities
Kernel Internals: Threads

- Class specific functions for fork, preempt, sleep...
- Threads hold class specific data
- Dispatcher picks next thread from dispatch queue
- Scheduler decides priority/time slice for thread
Lottery Threads (LT): Our Approach

- Minimize/avoid changes to other kernel parts (e.g., dispatcher)
- Work within current framework
- LT scheduler is a module
• fixed priority 0 (lowest), time slice 40ms

• LT scheduler holds lottery to put next thread on do

• dispatcher only finds one LT thread/CPU to run

• scheduler only puts one LT thread/CPU on dispatch queues

• like IA, LT is only a special case of TS
set front dq

++/--

SLEEP

fork

select parent

select child

insert LT

LT unclass

SLEEP

stop

free DQ
{ 

close(fd);
read(fd, &buf, 1);
int fd = open("/etc/motd", O_RDONLY);
char buf;
}
while(t ++ )
}
void main()

:: LOOP-SCALAR - "I/O Bound"

{ 

{}

}

void main()

:: LOOP-WHILE - "Computation Bound"

:: TEST PROGRAMS
I CPU, 2 users, 3 threads, 1:2, while

TIME -->

% CPU

653-39809
658-101640
659-101640

Michael Kropfberger@emx.net "SMP-Lottery Scheduler for Solaris"
8 CPU, 2 users, 24 Threads, 8:16, while