Summary for Currentcy: Unifying Policies for Resource Management

Four specific energy related goals:

1) currentcy conserving scheduling algorithms that reduce residual battery capacity,

2) proportional energy sharing,

3) response time variation, and

4) energy efficient disk management.

Contributions:

1) Reclaim residual battery capacity.

2) Proportionally shares energy according to user specified preferences.

3) Achieves smooth currentcy consumption, eliminating jitter in many applications.

4) How currentcy provides a common mechanism to expose and effectively manage the subtle interactions of applications using different resources.

5) demonstrate how to shape disk access patterns to amortize the energy costs of spinup/spindown across multiple requests.

6) Show significant savings in residual energy, lower response time variation, and reductions in average power costs for disk accesses.

There are two facets to the allocation strategy.

The first level allocation determines the amount of currentcy to collectively allocate among all system tasks.

There are two facets to the allocation strategy. The first level allocation determines the amount of currentcy to collectively allocate among all system tasks.

Resource Containers to capture the activity of an application or task as it consumes energy throughout the system. Resource containers are the abstraction to which currentcy allocations are granted and the entities to be charged for energy consumption. Resource Containers address variations in program structure that typically complicate accounting.

Currentcy-based Policies

A currentcy conserving policy provides service in response to demand for energy as long as unspent currentcy is available in an epoch. This is important to minimize residual energy when the target battery lifetime has been reached.

The energy consumption can be lower if the requirements of the task are low enough to be fully satisfied by the level of energy available to it.

Reduce the variation in response times.

Encouraging the most efficient use of a device's power saving modes allows performance to be achieved at lower energy costs.

Currentcy Conserving Scheduling

To adjust the overall allocation level when the system detects that the battery is not being drained at the expected rate.

To explicitly redistribute excess currentcy to other tasks with insufficient currentcy for their energy demands.

Proportional Energy Use

We can weigh the basis against which the energy consumed this epoch is compared by a factor defined to be the task's share divided by the amount of currentcy actually spent in the last epoch.

Our EC scheduler also incorporates one final feature called self-pacing that will be described in the next section with the goal of smoothing response times.

There are three aspects of the EC scheduler that can be mixed in various combinations: the consumption-based stride, with or without dynamic shares, and with or without self-pacing.

Low Variance in Response Time

delay a task if its consumption of currentcy is ahead of schedule during an epoch. Progress is defined as the amount of currentcy spent thus far in the current epoch divided by the task's budget for the epoch. If this progress is greater than the ratio of elapsed time in this epoch over epoch length, then the task is delayed and the processor may go idle for a short interval of time.

Energy Efficiency

Currentcy provides a means for passing along the savings to tasks that cooperate through their usage patterns. The disk presents unique challenges and opportunities for currentcy-based policies since it has non-uniform power consumption. The cost of spinning up the disk is much greater than keeping it spinning for a short duration.

Sets the entry price of a disk access that requires a spinup cost much higher than the actual cost. When the access is actually permitted, we then debit the actual cost.

A prefetch buffer allocation policy tied to the average disk access cost.

Trigger prefetching operations that cause spinups using a bidding function based on the fraction of consumed prefetch buffers.

The flush daemon checks for dirty pages that have been idle for FlushStart seconds before starting to write to disk. At this point it writes all dirty pages that have been idle for more than FlushEnd seconds. As with the spinup pricing, the goal is to create bursts of activity.