Summary for Ghosts in the Machine: Interfaces for Better Power Management

Two new interfaces which inhibit device power management.

Monitor

1) An OS power manager allows applications to query the current power mode of I/O devices to evaluate the performance and energy cost of alternative strategies for reading and writing data.

Signal for the information of devices

2) We allow applications to disclose ghost hints that enable better power management in the presence of multiple devices.

1) Which adaptive applications issue to device power managers when they are forced to use a poor I/O path because a device is not in an ideal power mode
2) Which allow devices to implement proactive power management strategies that do not depend upon passive load observation.

From the effort of this cache to increasing the rw speed

Middleware layer supports adaptive disk cache management

Reduces interactive response time for a Web browser by 27% and decreases total energy usage by 9%. For a mail reader, the cache manager decreases response time by 42% and energy use by 5%.

Feature:

How to predict?

1) Applications: Add an Interface of OS power management to querying device performance and energy characteristics. Then expose dynamic information of power mode of each device.

2) Algorithm: Use a hint (Ghost Hint) to predict which power mode the device chooses.

Dynamic power management: observe device access patterns to determine when to change modes. They attempt to use high-performance modes during periods of high activity and power-saving modes during idle periods.

Impact of power management:

What is his Energy formula?

$Q=P\times T=U\times I\times T$

$P\times T$

=Work+Heat

$=P'\times T+I^2RT$

1) Without Power Management, local storage is better than remote.

2) With Power Management,

a) high-performance mode: local is best.

b) power-saving mode:

i) small object(32KB): remote is better

ii) large object(512KB): local is better

The benefit (and limitations) of adaptation

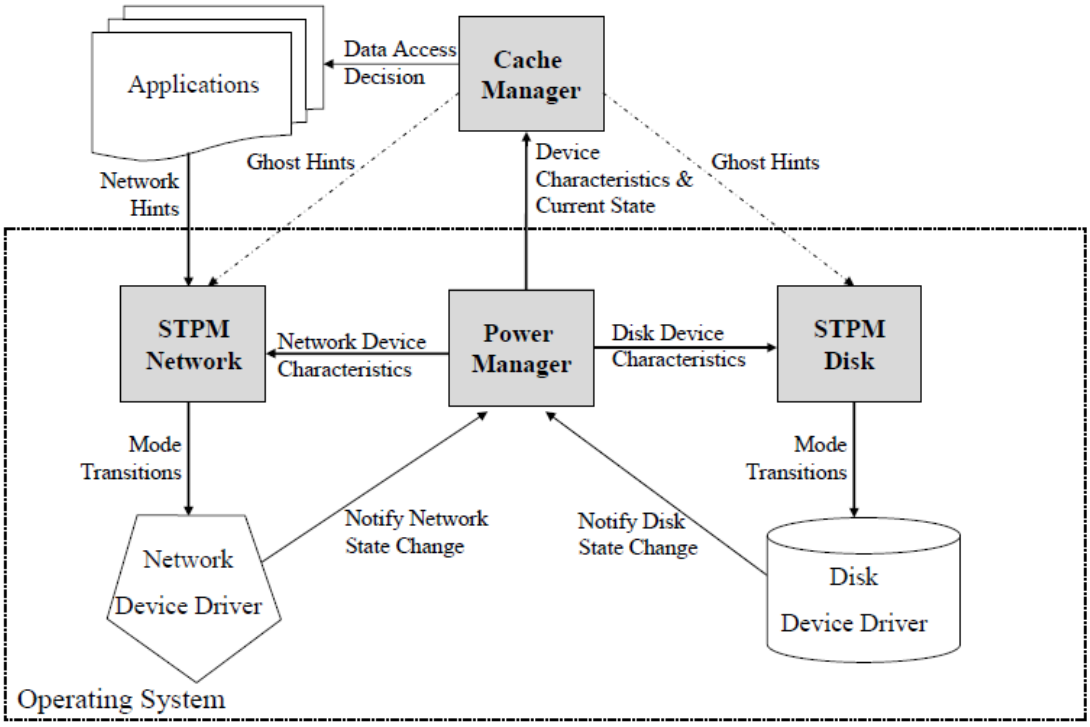

Figure 4: Energy-aware architecture

The adaptive application first calculates the best possible method for accessing data given each device's current power state. It fetches the data using this method. However, the application also computes what method would have been best if all devices had been in the ideal power mode. If this ideal method differs from the method actually used, the application issues a ghost hint to the power management layer that describes the opportunity lost due to a device not being in the ideal power mode.

For device drivers

For application

The OS power manager

```
RegisterDevice      (IN device, IN mode_data, IN transition_costs, OUT handle);
Notify              (IN handle, IN new_mode);
DeregisterDevice    (IN handle);
QueryNumModes       (IN device, OUT num_modes);
QueryModeInfo       (IN device, IN mode, OUT mode_info);
QueryTransitionCost (IN device, IN from_mode, IN to_mode, OUT cost);
QueryCurrentMode    (IN device, OUT mode);
RegisterCallback    (IN device, IN mode);
```

Figure 5: Power manager interface

RegisterDevice() : When a device is loaded, its driver calls it and discloses the device's supported power modes, as well as performance and energy characteristics in each mode.

Notify(): Whenever the device changes to a new power mode, the driver calls it to inform the power manager.

QueryModeInfo(): returns attributes of a specified power mode such as the power drawn by the device and whether the device is capable of reading or writing data in that mode. Further, if the device can read or write data in the queried power mode, QueryModeInfo also returns a model of the performance and energy costs of I/O operations.

QueryTransitionCost(): returns the cost of changing modes- this is expressed in terms of both time and energy.

QueryCurrentMode(): returns the power mode of a specified device.

RegisterCallback(): to block until a device enters a specific mode.

Self-tuning power management

Adjust the relative priority of performance and energy conservation. STPM modules allow applications to supply additional context about device accesses.

$$T_{be} = (E_{up} + E_{down})/(P_{hp} - P_{lp}) \qquad (1)$$

$$C_E = E_{down} - (P_{hp} * T_{down}) + E_{up} + (P_{base} * T_{up}) \qquad (2)$$

$$T_{be} = \frac{(T_{up} * K) + (C_E * (100 - K))}{(P_{hp} - P_{lp}) * (100 - K)} \qquad (3)$$

Middleware for energy-aware caching

```
CacheStatus    (IN name, OUT status, OUT size);
GetData        (IN name, OUT data);
GetMetadata    (IN name, OUT metadata);
PutData        (IN name, IN data);
PutMetadata    (IN name, IN metadata);
Rename         (IN new_name, IN old_name);
Delete         (IN name);
```

Figure 6: Cache manager interface

CacheStatus(): determine whether a file is stored in the cache. This function returns one of three results: the file is not present, the file is present, or the file is present but it would be preferable to fetch the file from the network. It returns the size of the cached file.

$$Cost_{total} = Exp\_time * K + Exp\_energy * (100 - K) \quad (4)$$

PutData(): to store the item in the cache.

PutMetadata(): call stores application-specific metadata for each object.

Ghost hints

Applications expose “accesses that might have been” to device power managers.