Energy-Aware Scheduling in Disk Storage Systems

Jerry Chou, Jinoh Kim, Doron Rotem

Lawrence Berkeley National Lab
Data Management Group

6/22/2011 @ ICDCS
Energy

- Battery lifetime for small and remote devices
- How about desktop, server, datacenter?
Why care energy for Datacenter?

- [source IBM 2006]
- [source LBNL 2007]
Why Disk Storage Systems?

These are typical relative values. Your usage should vary based on:

- Size and scale of IT environment
- Application and workload types
- Storage capacity vs. I/O centric
- Compute vs. data centric
- Types and ages of equipment
- Management approaches

Source: StorageIO, Greg Sculz  www.storageio.com
Data Intensive Computing

- Driven by computation model & workload
  - Cloud computing / HPC computer.
  - Datacenter (less complexity, cheap equipments).
- Rapidly increasing data
  - Facebook: 130TB/day users log, 300TB/data user data.
  - Google: > 25PB/day processed data.
- Cheaper storage and computation cost
Disk Energy Consumption

- Disk can turn off motor in sleep mode
  - Sleep power << Idle power
- Delay penalty for disk spin-up/down
  - Spin-up/down is over 10 seconds
- Idle threshold
  - Idle time before switching to sleep mode

Idle threshold (54s) →

Active (13.3W)

Spin-up (24W, 15s) →

Spin-down (0.34W, 10s)

Idle (9.3W) →

Sleep (0.8W)
Disks Storage Systems Model

- Data spread across disks.
- Data replicated for availability and performance.
- Each request for a single data block (512B):
  - Data I/O time << idle threshold
  - Disks cannot sleep b.c. no prolonged idle time.
Energy-Aware Scheduling

- Exploit data replication on-the-fly.
- No data placement interference.
- No additional deployment overhead.
Energy-Aware Scheduling Algorithms

• Online
  – Heuristic cost function.
  – Consider both energy and performance.

• Batch
  – Equivalent to set cover problem.
  – Apply to common batch system/workload model.

• Offline
  – Equivalent to independent set problem.
  – Analyze for optimal solutions.
Optimal Batch Scheduling Example

- All requests access disks at the same time.
- Energy consumption is proportional to the number of scheduled disks.
- Minimize energy = minimize scheduled disks.
- Optimal schedule is the minimum covering set.

Both disks consume \((E_{up} + E_{down} + T_1 \times P_1)\)
Optimal Offline Scheduling Example

Idle threshold = 5, disk unit power = 1
Offline Scheduling Observations

- The “potential” energy saving from any pair of requests is determined by their arrival time $t$.

- Energy saving from request $r_i$, $r_j$ on disk $d_k$:

$$X(i, j, k) = \begin{cases} 
0, & \text{if } t_j - t_i \geq T_I + T_{up} + T_{down} \\
E_{up} + E_{down} + (T_I - (t_j - t_i)) \times P_I, & \text{otherwise}
\end{cases}$$

- Intuition:
  – More saving from longer idle time overlap.
  – No saving after a disk has been spun down.
Offline Scheduling Observations

• The “real” energy saving of a schedule is the sum of saving from its requests.

• Example:
  – The energy saving of above schedule is
    \[ X(1,2,1) + X(2,3,1) + X(5,6,4) = 4 + 3 + 4 = 11 \]
Offline Scheduling Algorithm

• Put it all together:
  Step 1: Compute all potential energy saving from requests

- R1:b1
- R2:b2
- R3:b3
- R4:b4
- R5:b5
- R6:b6

- X(1,2,1) = 4
- X(1,3,1) = 2
- X(2,3,1) = 3
- X(2,3,2) = 3
- X(5,6,4) = 4
Offline Scheduling Algorithm

Put it all together:

**Step 1:** Compute all potential energy savings from requests

**Step 2:** Add schedule constraints

The successor of R1 is either R2 or R3

The location of R3 is either d1 or d2

The location of R3 is either d1 or d2
Offline Scheduling Algorithm

- Put it all together:
  Step 1: Compute all potential energy saving from requests
  Step 2: Add schedule constraints between $X(i,j,k)$ and $X(i’,j’,k’)$
  Step 3: Find \textit{maximum weighted independent set}

This is optimal!!
Online Scheduling Algorithm

• Schedule one request at a time.
• Use per node cost function:

\[ C(d_k) = E(d_k) \times \frac{\alpha}{\beta} + P(d_k) \times (1 - \alpha) \]

  – C(d_k): Performance cost can be computed by load.
  – E(d_k): Energy cost can be computed by disk idle time.
  – \( \alpha \beta \): Cost parameter.

• The cost function can also be applied to our set cover batch scheduling algorithm.
  – Set the weight of disks to their cost.
Experimental Evaluations

• Workload trace:
  – Data block (512B) I/O traces
  – Cello: collected by IBM

• Simulator:
  – Omnet++ for system simulation
  – DiskSim for disk simulation

• Data placement:
  – 180 disks
  – Original data is skewed distributed (zipf)
  – Replicated data is uniform distributed.
Energy Consumption

- Random save no energy under uniform data.
- Static cannot take advantage of data replication.
- Energy-aware scheduling saves 50% energy.
Number of Disk Spin-up/down
Energy-aware schedules have shorter response time. It is because the number of disk spin-up/down operations is also minimized.
• Energy-aware scheduling is resilient to data placement even under uniform data locality.
Energy-Aware Scheduling Summary

• Propose energy-aware scheduling technique without interference to data placement.
• Propose scheduling algorithms for online, batch and offline models
• Prove the optimal schedule can be reduced to independent set problem and is NP-complete.
• Show significant performance and energy improvement using realistic traces.
• Future work on better online scheduling algorithm.
Thank You!
Diverted Accesses

- Replicas are segregated from original data.
- Only access replicated data for performance or fault tolerance.
- Interfere with data placement manager.
PDC (Popular Data Concentration)

- Dynamically migrate data among disks.
- Less popular disks have longer idle time.
- High cost and difficult implementation.
Batch Scheduling Example

- Example setting:
  - No disk spin-up/down and sleep power.
  - 1 unit of power per second for idle disk.
  - Idle threshold is 5 seconds.
  - Disks are sleeping initially.

Arbitrary schedule: energy = 3 x 5 = 15
Batch Scheduling Example

• Example setting:
  – No disk spin-up/down and sleep power.
  – 1 unit of power per second for idle disk.
  – Idle threshold is 5 seconds.
  – Disks are sleeping initially.

Optimal batch schedule: energy = 2 x 5 = 10
Batch Scheduling Example

- Example setting:
  - No disk spin-up/down and sleep power.
  - 1 unit of power per second for idle disk.
  - Idle threshold is 5 seconds.
  - Disks are sleeping initially.

Optimal batch scheduling algorithm is the Minimum Set Cover algorithm!
Offline Scheduling Example

time

R1:b1  R2:b2  R3:b3  R4:b4  R5:b5  R6:b6

d1  d2  d3  d4
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1

time

1 2 4 6 13 14

R1:b1 R2:b2 R3:b3 R4:b4 R5:b5 R6:b6

d1 d2 d3 d4

energy

d1 idle sleep

d2 sleep

d3 sleep

d4 sleep
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1

<table>
<thead>
<tr>
<th>time</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>idle</td>
<td>idle</td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1

time

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1:b1</td>
<td>R2:b2</td>
<td>R3:b3</td>
<td>R4:b4</td>
<td>R5:b5</td>
<td>R6:b6</td>
</tr>
</tbody>
</table>

d1

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1 idle idle idle sleep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td>sleep</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>sleep</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>sleep</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1

<table>
<thead>
<tr>
<th>time</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>idle</td>
<td>idle</td>
<td>sleep</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>idle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>idle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>sleep</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>sleep</td>
</tr>
</tbody>
</table>
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1

<table>
<thead>
<tr>
<th>time</th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>idle</td>
<td>idle</td>
<td>idle</td>
<td>sleep</td>
<td>idle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d₂</td>
<td></td>
<td></td>
<td></td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d₃</td>
<td>sleep</td>
<td>idle</td>
<td></td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d₄</td>
<td></td>
<td></td>
<td></td>
<td>sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Offline Scheduling Example

Idle threshold = 5, disk unit power = 1
**Offline Scheduling Example**

- **Idle threshold = 5, disk unit power = 1**
- **Optimal batch schedule: energy = 23**

### Diagram Description
- **Time:** 1, 2, 4, 6, 13, 14
- **Disks:**
  - **d1**: idle, sleep
  - **d2**: idle
  - **d3**: sleep, idle
  - **d4**: sleep

**Legend**:
- **R1:b1**
- **R2:b2**
- **R3:b3**
- **R4:b4**
- **R5:b5**
- **R6:b6**
Optimal Offline Scheduling Algo.

• Problem definition:
  – Given a list of requests and data placement
  – The arrival time and requested data is known
  – Minimize energy consumption

• Optimal algorithm:
  – The problem is NP-complete .
  – Using maximum weighted independent set (MWIS) algo.
  – By proving our scheduling problem can be reduced to MWIS and vice versa.
Optimal Scheduling Problem

- Max energy consumption of a request is:
  - $E_{up} + E_{down} + \text{Idle power}(P_i) \times \text{Idle threshold}(T_i)$
Offline Scheduling Problem

- The “potential” energy saving from any pair of requests is determined by their arrival time $t$.

- Example of scheduling R1, R2 on disk d1:
  - Maximum energy consumption = $(E_{up} + E_{down} + T_{I} \cdot P_{I}) \cdot 2$
  - Actual consumption = $E_{up} + E_{down} + T_{I} \cdot P_{I} + (1-0) \cdot P_{I}$
  - Actual saving = $E_{up} + E_{down} + (T_{I} - (t_{2} - t_{1})) \cdot P_{I}$
Optimal Scheduling Algorithm

- Key observations:
  
  (2). The potential energy saving from any pair of requests is determined by their arrival time t.

Example of scheduling R3, R4 on disk d2:

- Maximum energy = \((E_{up} + E_{down} + T_i * P_i) * 2\)
- Energy consumption = \(E_{up} + E_{down} + T_i * P_i + (5-3) * P_i\)
- Energy saving = \(E_{up} + E_{down} + (T_i - (t_i - t_3)) * P_i\)
Optimal Offline Scheduling Algo.

• Put it all together:
  Step 1: Compute all potential energy saving from requests

\[ X(1,2,1) = 4 \]
Optimal Offline Scheduling Algorithm.

- Put it all together:
  Step 1: Compute all potential energy saving from requests

\[
X(1,2,1) = 4 \quad X(1,3,1) = 2
\]
Optimal Scheduling Algorithm

• Put it all together:
  Step1: Compute all potential energy saving from requests

- $X(1,2,1) = 4$
- $X(1,3,1) = 2$
- $X(2,3,1) = 3$
Optimal Scheduling Algorithm

• Put it all together:
  Step 1: Compute all potential energy saving from requests

\[ X(1,2,1) = 4 \]
\[ X(1,3,1) = 2 \]
\[ X(2,3,1) = 3 \]
\[ X(2,3,2) = 3 \]
### Offline Scheduling Algorithm

1. **Step 1:** Compute all potential energy saving from requests.
2. **Step 2:** Add schedule constraints between $X(i,j,k)$ and $X(i',j',k')$.
   
   (a) Single request location: $k = k'$ if $(i,j) \cap (i'j') \neq \emptyset$

---

**Diagram:**

- **time:** 1, 2, 4, 6, 13, 14
- **Requests:** R1:b1, R2:b2, R3:b3, R4:b4, R5:b5, R6:b6
- **Locations:** d1, d2, d3, d4

*The location of R2 is either d1 or d2*

*The location of R3 is either d1 or d2*
Offline Scheduling Algorithm

• Put it all together:

   Step 1: Compute all potential energy saving from requests
   Step 2: Add schedule constraints between \( X(i,j,k) \) and \( X(i',j',k') \)

   (a). Single request location: \( k = k' \) if \( (i,j) \cap (i'j') \neq \emptyset \)

   (b). Single request appearance: \( i \neq i', j \neq j' \)

The successor of \( R1 \) is either \( R2 \) or \( R3 \)

The predecessor of \( R3 \) is either \( R1 \) or \( R2 \)

The predecessor of \( R3 \) is either \( R1 \) or \( R2 \)
Optimal Offline Scheduling Algo.

- Put it all together:

Optimal offline scheduling algorithm is the Maximum Weighted Independent Set algorithm!

Steps: Find maximum weighted independent set

This is optimal !!
Experimental Evaluations

- **Algorithms:**
  - Energy-aware Heuristic (online cost func. \( \alpha = 0.2, \beta = 100 \))
  - Energy-aware WSC (batch 0.1s scheduling interval)
  - Energy-aware MWIS (offline optimal)
  - Static (always schedule to original data)
  - Random (uniform distributed to all data locations)
Data Placement

- Both heavily depend on data locality.
- Maximum energy saving is less than 20%.
Response Time

- Energy-aware schedules have shorter response time.
- It is because the number of disk spin-up/down operations is also minimized.
90 Percentile Response Time

- Always-on
- Random
- Static
- Energy-aware Heuristic
- Energy-aware WSC (batch 0.1s)
- Energy-aware MWIS (offline)

Data replication factor

90 percentile request response time (ms)
Cost Function