Energy-efficient Mapping of Big Data Workflows under Deadline Constraints

Presenter: Tong Shu

Authors: Tong Shu and Prof. Chase Q. Wu
Big Data Center
Department of Computer Science
New Jersey Institute of Technology, Newark, NJ, USA

WORKS on Nov. 14th, 2016 in Salt Lake City, UT, USA
Outline

• Introduction
• Related Work
• Motivation
• Problem Formulation
• Computational Complexity Analysis
• Algorithm Design
  – A Pipeline-structured Workflow on a Homogenous Cluster
  – An Arbitrary Workflow on a Heterogeneous Cluster
• Performance Evaluation
• Summary of Contributions
Introduction

• Problem
  – Large-scale workflows for big data analytics in Hadoop systems consume a significant amount of energy
• Existing main green computing techniques
  – Insufficient to address the energy efficiency issue of big data workflows

Existing techniques

- Task consolidation to reduce static energy consumption (SEC) by turning off idle servers
- Load balancing to reduce dynamic energy consumption (DEC) through dynamic voltage and frequency scaling (DVFS)

Weakness

- Frequently switching on and off a server may reduce its lifespan or cause unnecessary peaks of energy consumption
- DVFS may not be always available on all servers in a cluster

• Our solution
  – Reduce DEC by adaptively determining the degree of parallelism in each MapReduce job to mitigate the workload overhead while meeting a given performance requirement
Related Work in Energy-efficient Scheduling

<table>
<thead>
<tr>
<th>Workflow Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Virtualized environments</strong></td>
</tr>
<tr>
<td>o Migrate active VMs onto energy-efficient PMs [TCC’16]</td>
</tr>
<tr>
<td>o Consolidate jobs with complementary resource demands [SC’10]</td>
</tr>
<tr>
<td><strong>Physical environments</strong> [TPDS’11, TC’12, IS’15, IS’16]</td>
</tr>
<tr>
<td>o Extend virtual deadlines of jobs with less dependencies [TPDS’16]</td>
</tr>
</tbody>
</table>

Existing work only considers workflows comprised of serial or rigid jobs, while our work is focused on workflows comprised of moldable jobs.

<table>
<thead>
<tr>
<th>MapReduce Job Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heterogeneous machines</strong> [TC’12, TPDS’15, ICDCS’15]</td>
</tr>
<tr>
<td><strong>Renewable energy</strong> [EuroSys’12, IPDPS’15]</td>
</tr>
<tr>
<td><strong>Overhead reduction</strong> [ICDCS’13, Middleware’15]</td>
</tr>
</tbody>
</table>

Existing work only considers independent MapReduce jobs, while our work deals with precedence constraints among MapReduce jobs.

<table>
<thead>
<tr>
<th>Moldable/Malleable Job Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DVFS:</strong> P solution ( p = f^\alpha ) and FPTAS ( p = f^\alpha + \delta ) [Euro-Par’12]</td>
</tr>
</tbody>
</table>

Existing work schedules malleable jobs based on theoretical power consumption models, while our work schedules moldable jobs based on measurement-based power consumption models.
Motivation

**Rigid Jobs**
- Running on a fixed number of processors
- Multi-threaded programs

**Moldable Jobs**
- Running on any number of processors decided prior to execution
- MapReduce jobs in Hadoop

**Malleable Jobs**
- Running on a variable number of processors at runtime

---

**Parallel Jobs**

Performance model:
The workload of each component task decreases and the total workload, proportional to DEC, increases as the number of allotted processors increases.

<table>
<thead>
<tr>
<th>deadline</th>
<th>time</th>
<th>workload</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4 ≤ deadline &lt; 6</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3 ≤ deadline &lt; 4</td>
<td>3</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
Performance Model Validation

- Experimental Results in Hadoop/YARN
  - A single real-life MapReduce job runs on a computer server equipped with 2 processors of Intel(R) Xeon(R) CPU E5-2630, each with 6 cores of 2.30GHz and 64GB memory
  - The execution time of the job increases as the number of tasks increases when the server is fully utilized by the job, which means the total workload increases with the number of tasks
Cost Models

• Cluster Model
  – Cluster $M = \{\text{machine } m_i(N_i, p_i, o_i, P_i)\}$
  – Available resource-time (ART) table $\{N^A_i(t) \leq N_i, o^A_i(t) \leq o_i\}$

• Workflow Model
  – DAG with a deadline: $f(G(V, A), d)$, where $t^F_j \leq t^S_{j'}, \forall a_{j, j'} \in A$

• MapReduce Model
  – Each task $s_{j,k}$ in job $v_j$ requires a memory of size $o_j$ and spends a percentage $\mu_{i,j}$ of time executing CPU-bound instructions on a CPU core of machine $m_i$
  – As the number $K_j$ of parallel tasks increases, the workload $w_{j,k}(K_j)$ of each task $s_{j,k}$ decreases and the total workload $w_j(K_j) = K_j \cdot w_{j,k}(K_j)$ of all tasks increases
  – Execution time of each task: $t_{i,j,k} = w_{j,k}(K_j) / (\mu_{i,j} \cdot p_i)$
  – Used resources: $n_i(t) = \sum_{v_j \in V} n_{i,j}(t), o_i(t) = \sum_{v_j \in V} [o_j n_{i,j}(t)]$

• Energy Model
  – Dynamic power consumption proportional to utilization

• Mapping Function
  $\mathcal{M} : \{s_k(v_j) \xrightarrow{[t^S_{j,k}, t^E_{j,k}]} m_i, \forall v_j \in V, \exists m_i \in M, \exists [t^S_{j,k}, t^F_{j,k}] \subset T\}$
Problem Definition: EEWM

• Input
  – Given a cluster \( \{ m_i(N_i, p_i, o_i, P_i) \} \) of machines with an available resource-time table \( \{ N_i^A(t), o_i^A(t) \} \), and a workflow request \( f(G(V,A), d) \), where each job \( v_j \) has a set \( \{ w_j(K_j) \mid K_j = 1, 2, \ldots, K'_j \} \) of workloads for different task partitions, and each task in job \( v_j \) has a memory demand \( o_j \) and a percentage \( \mu_{i,j} \) of execution time for CPU-bound instructions on machine \( m_i \),

• Question
  – Find a mapping function to minimize dynamic energy consumption:

\[
\min_{m} E,
\]

• Constraints
  – Precedence: \( t_j^F \leq t_{j'}^S, \forall a_{j,j'} \in A \),
  – Deadline: \( t^C \leq d \),
  – Resource: \( n_i(t) \leq N_i^A(t), \forall m_i \in M \),
    \( o_i(t) \leq o_i^A(t), \forall m_i \in M \).

Computational Complexity: Strongly NP-complete => No FPTAS
Special Case: A Pipeline-structured Workflow on a Homogenous Cluster (PHO)

- Computational Complexity Analysis
  - The two-choice knapsack problem (TCKP) is NP-hard
    - The knapsack problem is NP-hard
    - Reduce the knapsack problem to TCKP

![Knapsack Problem Diagram]

$max \text{ total value}$
Special Case: A Pipeline-structured Workflow on a Homogenous Cluster (2)

- Computational Complexity Analysis: NP-hardness
  - Reduce TCKP to EEWM-PHO

\[
\begin{align*}
\text{max total value } V & \\
\text{total weight limit } W & \\
\text{min energy } E = \sum_{1 \leq j \leq J} V_j - V & \\
\text{deadline } d = W & \\
\end{align*}
\]

\[
V_j = \frac{(2w_{j,2}v_{j,1} - w_{j,1}v_{j,2})}{(2w_{j,2} - w_{j,1})}
\]

Class \( J \)

Job 1 \( \ldots \) Job \( j \) \( \ldots \) Job \( J \)

\[
\begin{align*}
\text{Item 1} & \\
\text{value } v_{j,1} & \\
\text{weight } w_{j,1} & \\
\text{Item 2} & \\
\text{value } v_{j,2} & \\
\text{weight } w_{j,2} & \\
\end{align*}
\]

\[
\begin{align*}
v_{j,1} & > v_{j,2} \\
w_{j,1} & > w_{j,2} > 0
\end{align*}
\]

Option 1 (1 task):
- energy \( E_j(1) = V_j - v_{j,1} \)
- time \( t_j(1) = w_{j,1} \)

Option 2 (2 tasks):
- energy \( E_j(2) = V_j - v_{j,2} \)
- time \( t_j(2) = w_{j,2} \)
Special Case: A Pipeline-structured Workflow on a Homogenous Cluster (3)
• Fully Polynomial-Time Approximation Scheme (FPTAS)
  – Reducing EEWM-PHO to RSP

- The restricted shortest path (RSP) problem is weakly NP-complete and solvable with FPTAS, so is EEWM-PHO
- EEWM-PHO-FPTAS
  • Time Complexity: $O(JK' \log K' + \frac{1}{\varepsilon})$, where $K' = \max_{1 \leq j \leq J} K'_j$
  • Near optimality and low time complexity (linear w.r.t. $1/\varepsilon$)
Generalized Problem: An Arbitrary Workflow on a Heterogeneous Cluster

• Heuristic -- BAWMEE
  – Iterative critical path selection
    • Iteratively computes a critical path with the earliest last finish time (LFT) from the remaining unmapped workflow branches according to the average execution time of each job running in serial on all the machines.
  – Pipeline mapping (considering machine heterogeneity)
    • Resource/time sufficiency: In energy-efficient pipeline mapping (EEPM), we map all the homogeneous tasks in the same job onto a homogeneous sub-cluster, hence using EEWM-PHO-FPTAS to balance the trade-off between execution time and dynamic energy consumption for each job on a pipeline.
    • Resource/time insufficiency: In minimum delay pipeline mapping (MDPM) with energy awareness, we search for a good task partitioning to minimize the end time of each job by exponentially relaxing the limit on the maximum number of tasks in a job to make a tradeoff between the optimality and the time complexity.
Example for Resource/time sufficiency

<table>
<thead>
<tr>
<th>Time</th>
<th>3</th>
<th>2</th>
<th>5</th>
<th>4</th>
<th>( \Rightarrow )</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>( \Rightarrow )</td>
<td>8</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td># of Tasks</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
<td>( C_2 )</td>
<td></td>
</tr>
</tbody>
</table>

**Pipeline 1:**

1. \((1, C_1):0-3\)
2. \((1, C_1):3-6\)
3. \((1, C_2):6-11\)
4. \((1, C_1):11-14\)
5. \((1, C_2):14-19\)

**Pipeline 2:**

1. \((1, C_1):0-3\)
2. \((1, C_1):3-6\)
3. \((1, C_2):6-11\)
4. \((1, C_1):11-14\)
5. \((1, C_2):14-19\)
Example for Resource/time sufficiency (2)

- **Mapping Results**
  - BAWMEE (DEC = 45)
  - Optimal (DEC = 44)

Pipeline 3:

Pipeline 4:
Simulation Settings

• Cluster
  – Homogeneous sub-clusters
    – Each sub-cluster has the same number of machines

<table>
<thead>
<tr>
<th>Mach. Type</th>
<th>CPU Models</th>
<th># of cores</th>
<th>Freq. (GHz)</th>
<th>DPC per core (W)</th>
<th>Mem. (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6-core Xeon E7450</td>
<td>18</td>
<td>2.40</td>
<td>90</td>
<td>64</td>
</tr>
<tr>
<td>2</td>
<td>Single Core Xeon</td>
<td>6</td>
<td>3.20</td>
<td>92</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>2-Core Xeon 7150N</td>
<td>12</td>
<td>3.50</td>
<td>150</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>Itanium 2 9152M</td>
<td>8</td>
<td>1.66</td>
<td>104</td>
<td>64</td>
</tr>
</tbody>
</table>

• Workflows
  – # of edges = # of jobs x 1.5
  – Arrivals: Poisson distribution with the average arrival interval of 30 min

• Simulation
  – Duration per run: 3 days
  – The number of runs: 20

• MapReduce jobs
  – The maximum number of tasks per job: 12 to 48
  – The workload \( w(k) \) of a job
    • 1 task: 0.6 to 21.6 Tera CPU cycles
    • \( k>1 \) tasks: \( w(k-1)[1+0.2/(k-1)] \) to \( w(k-1)[1+0.6/(k-1)] \)
  – The percentage of execution time for CPU-bound instructions: 0.1 - 1.0
  – Memory demand per task: 0.5GB to 4GB at a step of 0.5GB

• Approximation
  – \( \varepsilon = 0.2 \)
Simulation Results

- Problem Size
  - Deadline factor: 0.1

| Index | \(||V|, |M|, 1/\lambda, T)\) | Index | \(||V|, |M|, 1/\lambda, T)\) |
|-------|-----------------|-------|-----------------|
| 1     | (3-7, 4, 240, 7) | 11    | (53-57, 192, 30, 1) |
| 2     | (8-12, 8, 200, 7) | 12    | (58-62, 256, 25, 1) |
| 3     | (13-17, 12, 160, 7) | 13    | (63-67, 384, 20, 1) |
| 4     | (18-22, 16, 150, 7) | 14    | (68-72, 512, 15, 1) |
| 5     | (23-27, 24, 120, 7) | 15    | (73-77, 768, 12, 1) |
| 6     | (28-32, 32, 105, 3) | 16    | (78-82, 1024, 10, 1/3) |
| 7     | (33-37, 48, 90, 3) | 17    | (83-87, 1536, 8, 1/3) |
| 8     | (38-42, 64, 60, 3) | 18    | (88-92, 2048, 6, 1/3) |
| 9     | (43-47, 96, 45, 3) | 19    | (93-97, 3072, 5, 1/3) |
| 10    | (48-52, 128, 30, 3) | 20    | (98-102, 4096, 4, 1/3) |

[Charts and graphs showing simulation results with graphs for deadline missing rate, DEC reduction, and running time per workflow mapping for different algorithms and workflow structures.]
Simulation Results (2)

• Deadline Constraint
  – Workflow size: 40 - 60 jobs
  – Cluster size: 128 machines
  – Workflow Structure: random
Simulation Results (3)

- Workflow Size
  - Deadline factor: 0.1
  - Workflow size variation: ±2 jobs
  - Cluster size: 128 machines
  - Workflow Structure: random
Simulation Results (4)

- **Cluster Size**
  - Deadline factor: 0.1
  - Workflow size: 40 - 60 jobs
  - Workflow Structure: random
Simulation Results (5)

- **Workflow Structure**
  - Deadline factor: 0.1
  - Workflow size: 40 - 60 jobs
  - Cluster size: 128 machines
Summary of Contributions

• The first to study energy-efficient mapping of big data workflows comprised of moldable jobs in Hadoop systems

• Proved a deadline-constrained pipeline-structured workflow mapping problem for minimum total (energy) cost to be weakly NP-complete and designed an FPTAS for it

• The performance superiority of the proposed heuristic for the general workflow mapping problem was illustrated by simulation results in Hadoop/YARN
  – Dynamic energy saving
  – Deadline missing rate