Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System

Takafumi Saito†1,2, Kento Sato†1,3, Hitoshi Sato†1,2 and Satoshi Matsuoka†1,2,4

†1 Tokyo Institute of Technology
†2 JST/CREST
†3 JSPS Research Fellow
†4 National Institute of Informatics
HPC at Extreme scale

• **Exponential growth** in computational power
  – Enables finer grained scientific simulations

• As system size increases, *Power/Energy consumption* and *Fault tolerant* are recognized as the most significant concern towards “Extreme scale”
Power consumption and Failures on HPC systems

- Current supercomputers consume already huge amount of power

- In such a big system, overall failures rate increases accordingly
  - TSUBAME2.0: MTBF = 14 hours

- In future exascale system, it’s projected to consume 20MW

- In future exascale system, MTBF is projected to shrink to a few hours

<table>
<thead>
<tr>
<th></th>
<th>Power Consumption (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSUBAME2.0 (2011)</td>
<td>1.3 MW</td>
</tr>
<tr>
<td>Titan (2012)</td>
<td>8.2 MW</td>
</tr>
<tr>
<td>Tianhe-2 (2013)</td>
<td>17.8 MW</td>
</tr>
</tbody>
</table>

TSUBAME2.0

2 CPUs + 3 GPUs
Power consumption of I/O

- Applications are required to write checkpoints more frequently to survive such failures with lower energy consumption

- During I/O operation, computing nodes perform less computation but consume relatively much power
  - HDD & SSD consume power over half of computation
  - ioDrive consumes almost same amount of power of computation

- Power/Energy-aware I/O is becoming significant towards extreme scale
  ⇒ Focus on minimizing energy consumption
Goal, Proposal and Contribution

• **Goal:** Energy-aware I/O optimization for checkpoint and restart

• **Proposal:** Profile/Model-based optimization using DVFS + dynamic I/O parallelism
  – I/O Profile: To predict power/performance, extract power/performance trend from preliminary exp. under different CPU frequencies + I/O parallelism
  – Optimization: Based on the I/O profile, decide optimal CPU frequency + parallelism to minimize energy by using a checkpoint Markov model

• **Contribution:**
  – Experimental studies showed
    • Improve a whole machine energy consumption by 1.5% in SSD, 4.7% in ioDrive system by only minimizing energy of I/O
    • Especially, more than 2x of improvement of write operation in ioDrive
Outline

• Introduction

• Our target checkpointing scheme

• Proposal
  – Energy-aware optimization based on checkpointing model
  – I/O profile creation

• Experiment

• Conclusion
Scalable Diskless Checkpointing

Generally, checkpoints are written to reliable shared PFS, but ...

- **PFS checkpointing**
  - Cause huge overhead
  - e.g. TSUBAME2.0 (1402 nodes)
  - => 3 hours to write all checkpoints

- **Diskless checkpointing**
  - Create redundant data across node-local storages using an erasure encoding technique such as XOR
  - Can restore lost checkpoints on a failure like RAID-5 technology
  - Scalable, and known as promising approach towards extreme scale
Flash memory: I/O accelerator

• To accelerate I/O and diskless checkpointing, several systems employed SSD for node-local storage
  – TSUBAME2.0@Tokyo Tech: 174TB
  – Gordon@SDSC: 256TB

• Recently, Fusion-io’s ioDrive is gathering attention for big-data processing by the high IOPS and bandwidth

• Those technologies are promising for accelerating diskless checkpointing in future systems

<table>
<thead>
<tr>
<th></th>
<th>SSD</th>
<th>HP SFF 15K 6G SAS HDD</th>
<th>ioDrive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random reads</td>
<td>&gt;20,000 IO/s</td>
<td>340 IO/s</td>
<td>119,790 IO/s</td>
</tr>
<tr>
<td>Random writes</td>
<td>&gt;5,000 IO/s</td>
<td>300 IO/s</td>
<td>89,549 (75/25 r/w mix) IO/s</td>
</tr>
<tr>
<td>Sequential reads</td>
<td>230 MB/s</td>
<td>160 MB/s</td>
<td>750 MB/s</td>
</tr>
<tr>
<td>Sequential writes</td>
<td>180 MB/s</td>
<td>160 MB/s</td>
<td>500 MB/s</td>
</tr>
</tbody>
</table>

Target checkpointing scheme & Approach

- We target diskless checkpointing using a node-local storage such as ioDrive, SSD and HDD
- Aim energy efficient checkpointing by dynamically changing CPU frequency and I/O parallelism

Approach
- Dynamically change I/O configurations
  1. CPU frequency
  2. I/O parallelism
Challenges on this approach

• Determining optimal CPU frequency and I/O parallelism is not easy

1. Different power/performance behavior under different CPU frequency and parallelism
   – ioDrive has different behavior compared to SSD and HDD

2. Resiliency consideration
Impact by CPU frequency

• If decrease CPU frequency, I/O throughput of ioDrive is degraded
• ioDrive relies on CPU cores for
  – *Grooming*: a garbage collector that pre-erases unused blocks in background to accelerate future write operation
  – *Wear leveling*: a balanced write technique to extend the lifetime of a device

![Graph showing I/O throughput (MB/sec) for HDD, SSD, and ioDrive for different CPU frequencies.]

**Machine spec**

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD Opteron Processor 6172 (12 cores) × 4 sockets</td>
</tr>
<tr>
<td>Memory</td>
<td>DDR3-1333 SDRAM DIMM (128GB)</td>
</tr>
<tr>
<td>HDD</td>
<td>Fujitsu MHZ2500B (rpm:4200, seek:12ms)</td>
</tr>
<tr>
<td>SSD</td>
<td>Intel SSD 320 Series 600GB, SSDSA2CW600G3K5 (Sequential read/write: 270/220 [MB/s])</td>
</tr>
<tr>
<td>PCIe-attached flash memory</td>
<td>Fusion-io ioDrive MLC 320GB (Read/Write bandwidth: 735/510 [MB/s])</td>
</tr>
</tbody>
</table>
Impact by I/O parallelism

- In HDD & SSD, I/O throughput decrease because of contention among I/O processes

- In ioDrive,
  - 1-8 procs: I/O throughput increase because a fewer number of I/O processes cannot utilize bandwidth of ioDrive
  - 8-48 procs: I/O throughput decrease because of contention among I/O processes
Challenges on this approach

• Determining optimal CPU frequency and I/O parallelism is not easy

1. Different power/performance behavior under different CPU frequency and parallelism
   – ioDrive has different feature compared to SSD, HDD

2. Resiliency consideration
Resiliency consideration

- If we set to minimal CPU frequency and I/O parallelism, we can reduce power but checkpoint time can increase, which results in:
  - **Increasing re-execution time**: Prolonged checkpoint time has high probability to encounter a failure during the checkpoint
  - **Losing effective runtime**: Prolonged checkpoint time takes up more effective runtime
Challenges on this approach

• Determining optimal CPU frequency and I/O parallelism is not easy

1. Different power/performance behavior under different CPU frequency and parallelism
   - ioDrive has different feature compared to SSD, HDD
     ⇒ I/O profiling technique

2. Resiliency consideration
   ⇒ Energy-aware optimization based on checkpointing model
Outline

• Introduction
• Our target checkpointing scheme

• Proposal
  – Energy-aware optimization based on checkpointing model
  – I/O profile creation

• Experiment
• Conclusion
Checkpointing Markov model

- Application state can be described as Markov model with three states
- If no failure, can transition across compute and checkpoint states in sequence
- If failure happens, transitions to Restart state, rollback to the last compute state after recovery

N. Vaiday’s checkpointing model
Energy-aware Optimization

- Given a system failure rate $\lambda$, the Vaidya’s model gives expected time of each state as follows:

| State       | Expected \{run | I/O\} time | Power |
|-------------|----------------------------|-------|
| Compute     | $T_A = \lambda^{-1} e^{\lambda(T_C + T_R)} \left( e^{\lambda T_A} - 1 \right)$ | $W_A$ |
| Checkpoint  | $T_C = \lambda^{-1} \left( e^{\lambda T_C} - 1 \right)$ | $W_C$ |
| Restart     | $T_R = \lambda^{-1} \left( e^{\lambda T_C} - 1 \right) \left( e^{\lambda T_R} - 1 \right)$ | $W_R$ |

- By computing sum of the products of expected times and powers, we can get expected energy consumption:

$$J = T_A \cdot W_A + T_C \cdot W_C + T_R \cdot W_R$$

- To compute the energy, we need to know time and power consumption of checkpoint/restart, so we create I/O profile.
I/O profile creation

• To create I/O profile, measure power and throughput under different I/O settings

⇒ Given I/O parameters, we can estimate power and throughput
Summary of the energy-aware optimization

Find optimal # of procs and CPU freq by looking up I/O profile to minimize energy

Optimization model

Find
\[
\begin{align*}
(p | p & : \text{# of procs}) \\
(f | f & : \text{CPU freq})
\end{align*}
\]

s.t.
minimize \( J \)
\[
J = T_A \cdot W_A + T_C \cdot W_C + T_R \cdot W_R
\]

set to \( p \) I/O procs

set to \( f \) [GHz]
Design overview
of Energy-aware I/O system

- This work investigate how much our energy-aware I/O optimization can improve energy efficiency
Outline

• Introduction
• Our target checkpointing scheme
• Proposal
  – Energy-aware optimization based on checkpointing model
  – I/O profile creation

• Experiment
• Conclusion
Experimental settings

- **Checkpoint size:** 64GB per node
- **Application’s power consumption** \( (W_A) : 471.1 \text{ W} \)
  - NAS Parallel Benchmark (SP: Class C)
- **Failure rate:** \( \lambda = 1.89 \times 10^{-5} \) (MTBF = 14 hours)

TSUBAME2.0, 14\(^{th}\) in Top500 (June 2012)

- 2.4 PFlops
- 1442 nodes
- 2953 CPU sockets
- 4264 GPUs
- 197 switches
- 58 racks

Failure analysis on TSUBAME2.0

Period: 1.5 years (Nov 1\(^{st}\), 2010 ~ April 6\(^{th}\) 2012)
Observations: 962 node failures in total

- Compute nodes: 791
- Rack: 30
- PSU: 18
- PFS, Core switch: 8
- Edge switch: 6
Experimental settings (cont’d)

• Compare the proposed method (profile lookup) with three other strategies supported by `cpufreq`

<table>
<thead>
<tr>
<th>Compared cpufreq governor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile lookup</td>
</tr>
<tr>
<td>Performance</td>
</tr>
<tr>
<td>Powersave</td>
</tr>
<tr>
<td>Ondemand</td>
</tr>
</tbody>
</table>

• We use energy consumption per unit time for effective application execution (EPE) to compare the efficiency
  – EPE quantify a ratio of how much energy is consumed to compute an effective application time ($T_A$)

\[
EPE = \frac{T_A \cdot W_A + T_C \cdot W_C + T_R \cdot W_R}{T_A}
\]
Energy efficiency comparison

- Our proposed method can save energy by
  - 1.5% in SSD, 4.7% in ioDrive by only optimizing energy of I/O
- The efficiency improvement is limited
  - Application’s power consumption dominate the EPE
  - In a future extreme scale, checkpoint/restart cost may increase, the improvement will become bigger

About 6–17% of power is wasted by checkpoint/restart/re-executions by failures
Energy efficiency of sequential I/O

• Our proposed technique can be applied to general data-intensive applications, which conduct sequential I/O
  – e.g.) MapReduce: word count and inverted indexing (search engine)

\[\text{Energy consumption to write/ read 1MB} (\text{J/MB})\]

- HDD
- SSD
- ioDrive

\[\text{Better efficiency in any cases}\]

Especially, more than 2x of improvement of write operation in ioDrive

Our proposed method has a more beneficial effect on I/O intensive applications
Summary of experiment

Energy-aware optimal CPU frequency & # of procs

<table>
<thead>
<tr>
<th></th>
<th>Read</th>
<th></th>
<th></th>
<th>Write</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reads</td>
<td>SSD</td>
<td>ioDrive</td>
<td>Reads</td>
<td>SSD</td>
<td>ioDrive</td>
</tr>
<tr>
<td>HDD</td>
<td>1.7</td>
<td>1.4</td>
<td>2.1</td>
<td>2.1</td>
<td>1.7</td>
<td>1.4</td>
</tr>
<tr>
<td>SSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU freq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4 ~ 2.1 [GHz]</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td># of procs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- When we write/read checkpoint, the best strategy is ...
  - CPU frequency: 1.4 ~ 2.1 GHz, # of Procs: 1 or 2
Conclusion

- **Power/Energy consumption** and **Fault tolerant** are significant concern towards extreme scale
- Proposed **Profile/Model-based optimization** using DVFS + dynamic I/O parallelism
- Experimental studies showed
  - Improve a whole machine energy-consumption by 1.5 % in SSD, 4.7% in ioDrive system by optimizing only checkpoint/restart operation
  - Especially, **more than 2x** of improvement of write operation in ioDrive
  - More beneficial for I/O intensive applications

- **Future work**
  - Extend to more general I/O-intensive applications
    - e.g.) Support a random, slide access
Q & A

Speaker:
Kento Sato (佐藤 賢斗)
Tokyo Institute of Technology (Tokyo Tech)
Research Fellow of the Japan Society for the Promotion of Science
http://matsu-www.is.titech.ac.jp/~kent/

Authors
Takafumi Saito, Kento Sato, Hitoshi Sato and Satoshi Matsuoka

Acknowledgement
This work was supported by Grant-in-Aid for Scientific Research (S) 23220003, the Japan Science and Technology Agency (JST), and the Core Research of Evolutionary Science and Technology (CREST) research project.