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

Takafumi Saito[†1,2], <u>Kento Sato</u>[†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*

Fault Tolerance for HPC at eXtreme Scale 2013

With HPDC 2013
New York City
June 17-21, 2013

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

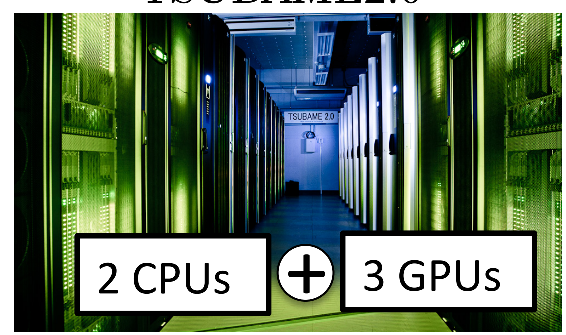
power consumption

TSUBAME2.0 (2011)	1.3 MW
Titan (2012)	8.2 MW
Tianhe-2 (2013)	17.8 MW

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

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

TSUBAME2.0



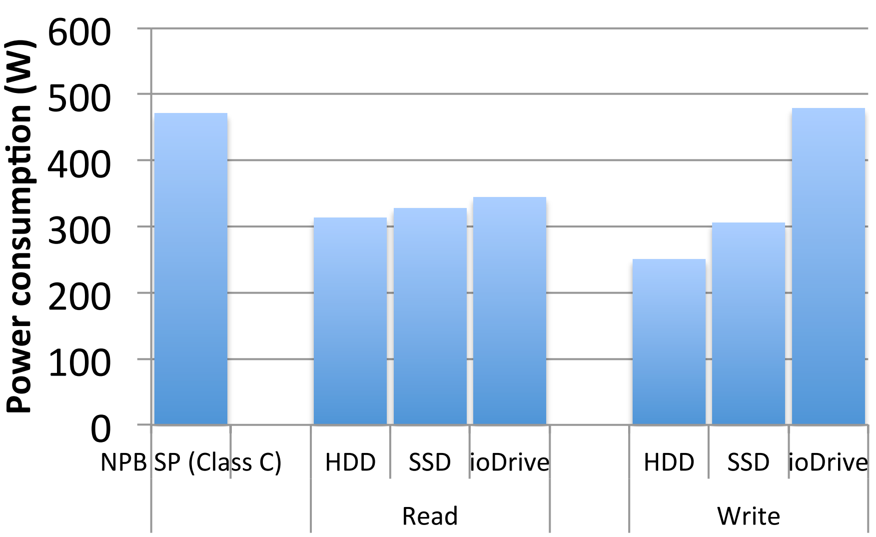
- In future exascale system, MTBF is projected to shrink to a few hours

Power consumption of I/O

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

HDD	Fujitsu MHZ2500B (rpm:4200, seek:12ms)
SSD	Intel SSD 320 Series 600GB, SSDSA2CW600G3K5 (Sequential read/write: 270/220 [MB/s])
PCIe-attached flash memory	Fusion-io ioDrive MLC 320GB (Read/Write bandwidth: 735/510 [MB/s])

- 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

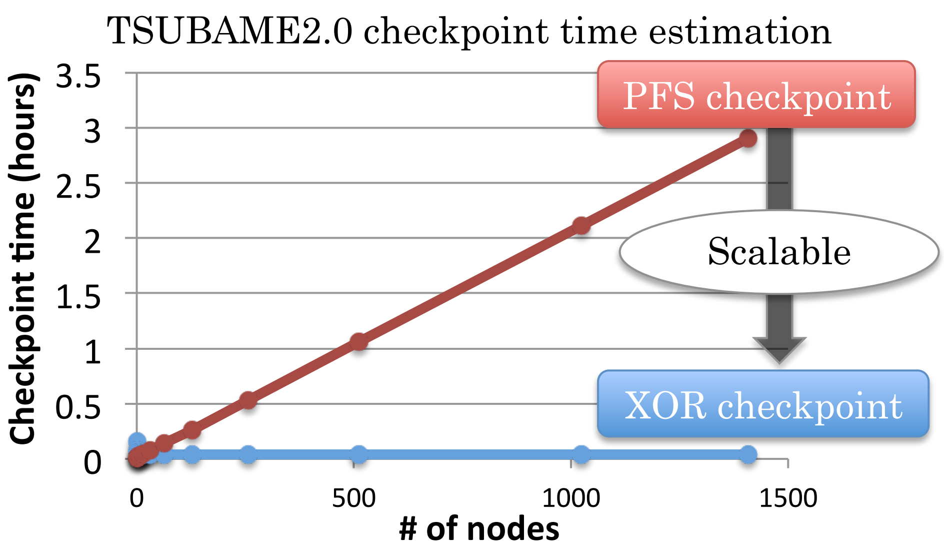
- Introduction
- Our target checkpointing scheme
- Proposal
  - Energy-aware optimization based on checkpointing model
  - I/O profile creation
- Experiment
- Conclusion

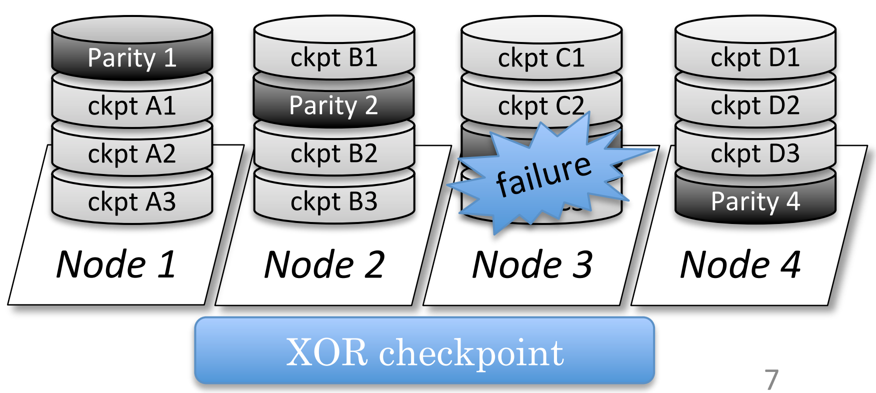
Scalable Diskless Checkpointing

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

- PFS checkpointing
  - Cause huge overhead
  - e.g.) TSUBAME2.0 (1402nodes)

  => 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

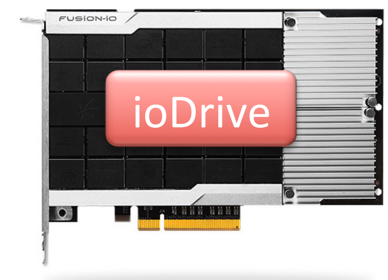




Flash memory: I/O accelerator

- 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



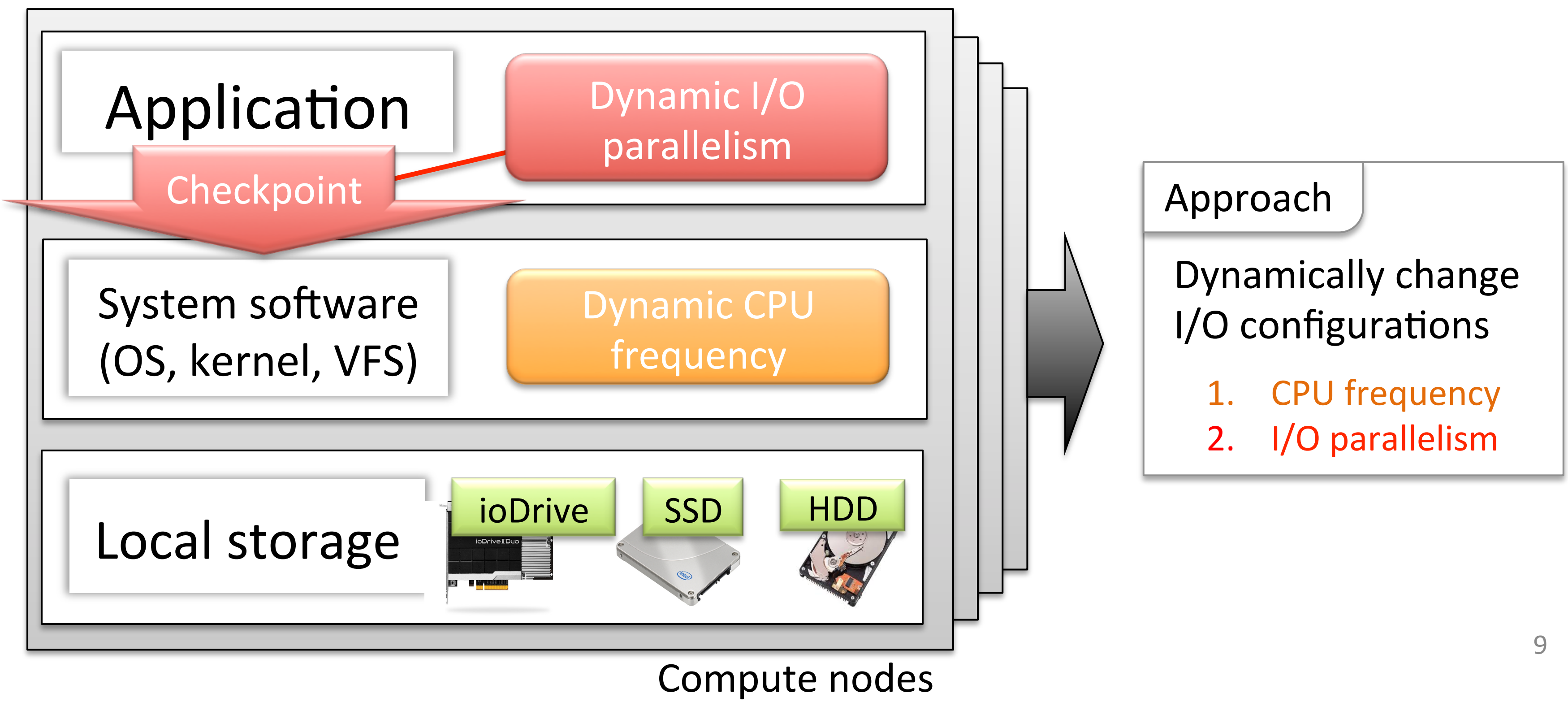
- Those technologies are promising for accelerating diskless checkpointing in future systems

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

Source: HP, "A comparison of SSD, ioDrives, and SAS rotational drives using TPC-H Benchmark", Technical white paper, April 2011

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



Challenges on this approach

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

Challenges

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

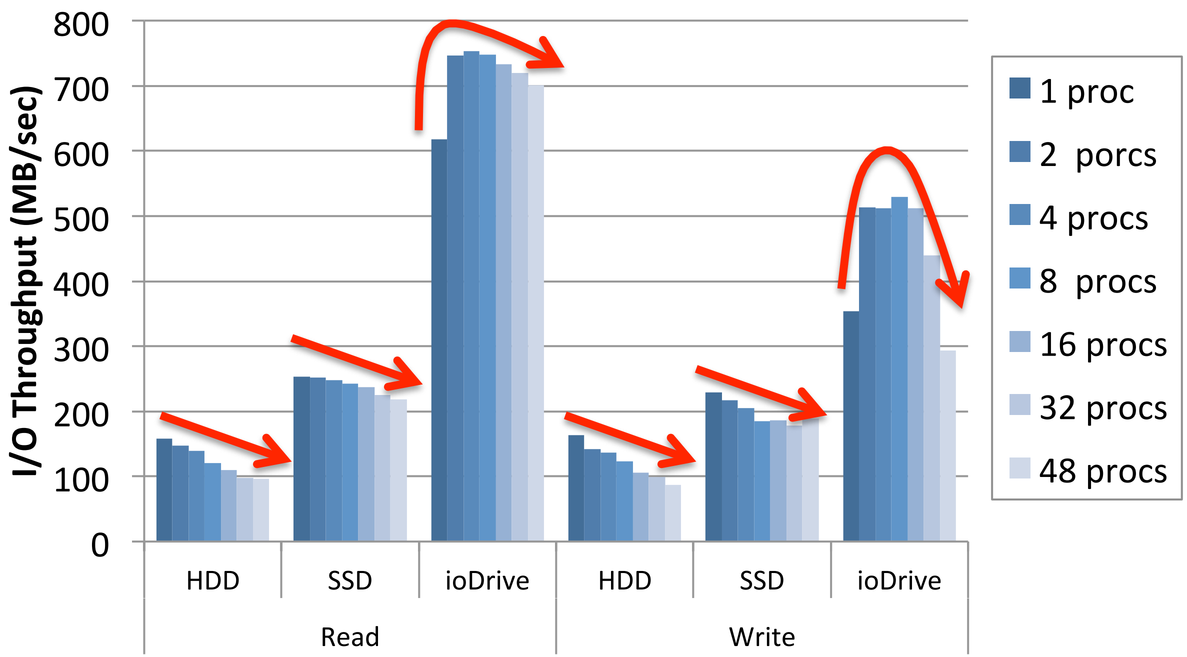
- If decrease CPU frequency, I/O throughput of ioDrive is degraded
- ioDrive relies on CPU cores for
  - *<u>Grooming</u>*: a garbage collector that pre-erases unused blocks in background to accelerate future write operation
  - *<u>Wear leveling</u>*: a balanced write technique to extend the lifetime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

Machine spec

CPU	AMD Opteron Processor 6172 (12 cores) × 4 sockets
Memory	DDR3-1333 SDRAM DIMM (128GB)
HDD	Fujitsu MHZ2500B (rpm:4200, seek:12ms)
SSD	Intel SSD 320 Series 600GB, SSDSA2CW600G3K5 (Sequential read/write: 270/220 [MB/s])
PCIe-attached flash memory	Fusion-io ioDrive MLC 320GB (Read/Write bandwidth: 735/510 [MB/s])

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

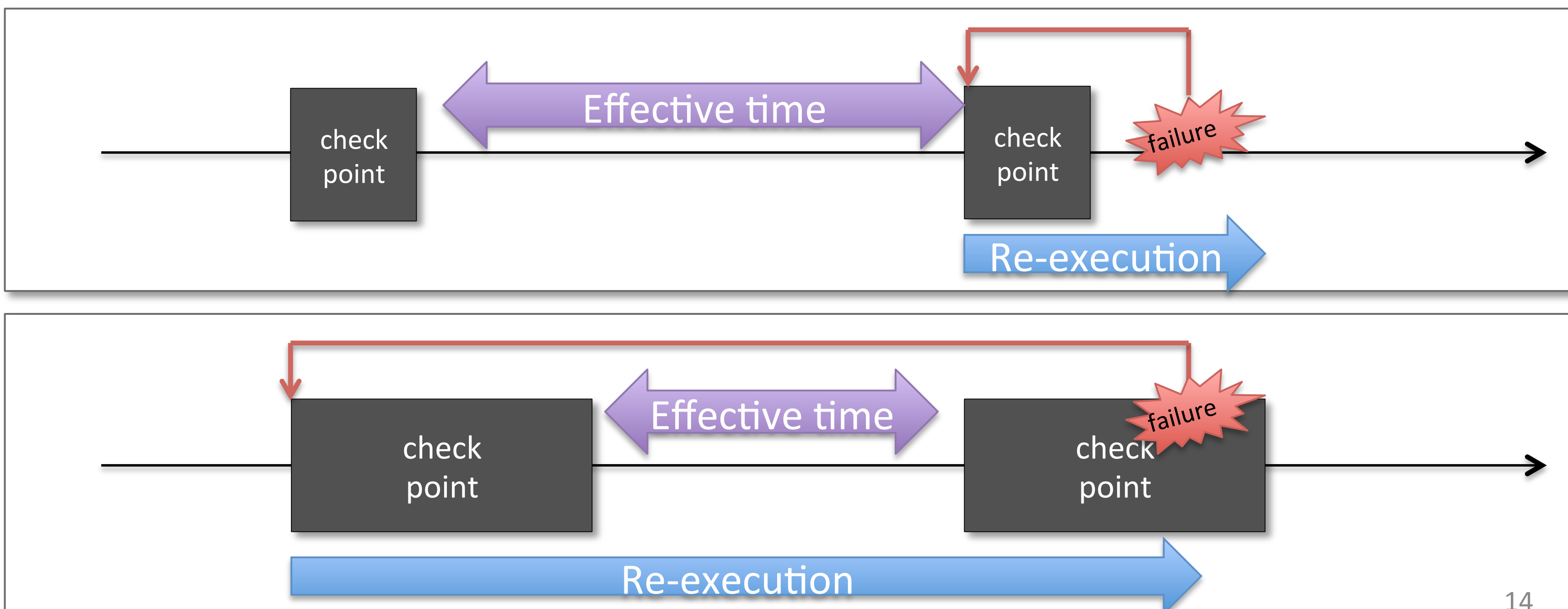
Challenges

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

Challenges

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

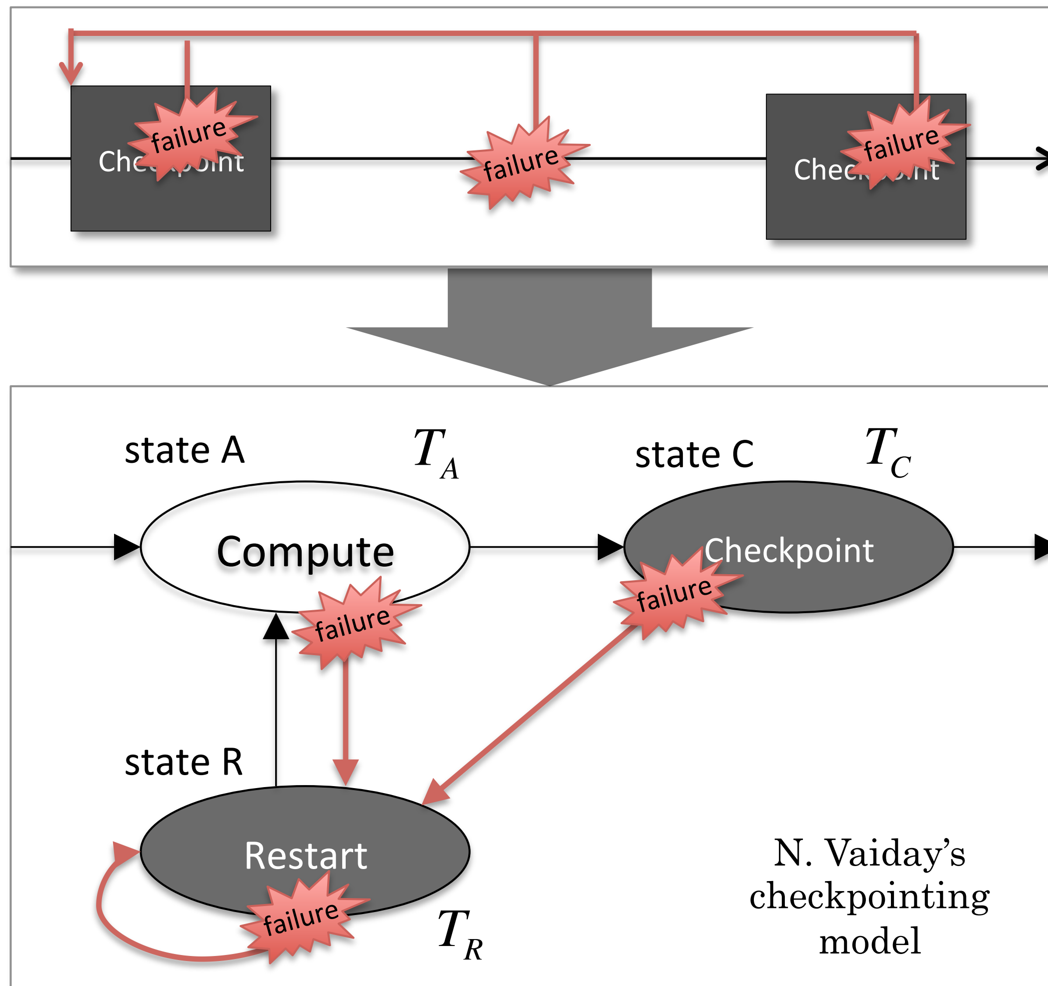
   ⇒ Energy-aware optimization based on checkpointing model

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



Energy-aware Optimization

- Given a system failure rate λ, the Vaidya's model gives expected time of each state as follows:

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

- By computing sum of the products of expected times and powers, we can get expected energy consumption
- To compute the energy, we need to know time and power consumption of checkpoint/restart, so we create I/O profile

$$J = T_{\bar{A}} \cdot W_A + T_{\bar{C}} \cdot W_C + T_{\bar{R}} \cdot W_R$$

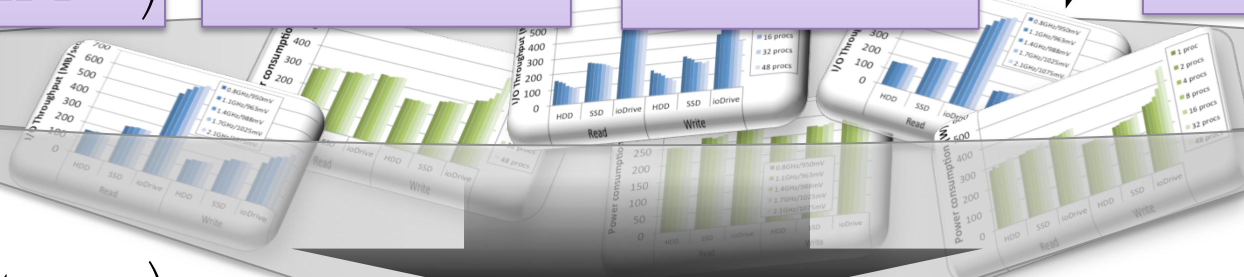
I/O profile creation

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

  ⇒ Given I/O parameters, we can estimate power and throughput

I/O type · I/O device · # of procs · CPU frequency → Measurement

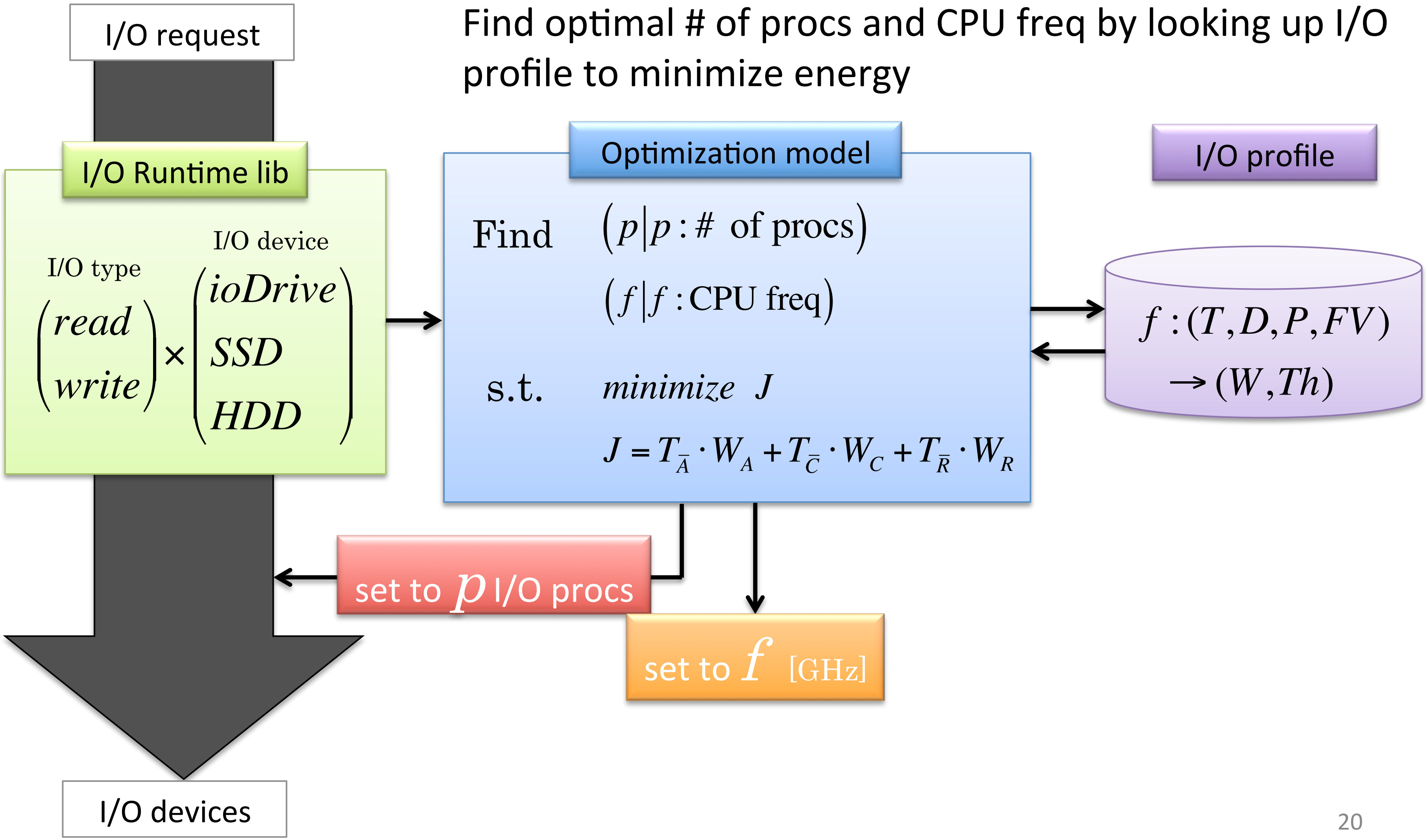
$$\begin{pmatrix} read \\ write \end{pmatrix} \times \begin{pmatrix} ioDrive \\ SSD \\ HDD \end{pmatrix} \times \left(p \mid p : \#\ \text{of procs} \right) \times \left(f \mid f : \text{CPU freq} \right) \Rightarrow \begin{pmatrix} Power \\ Througput \end{pmatrix}$$



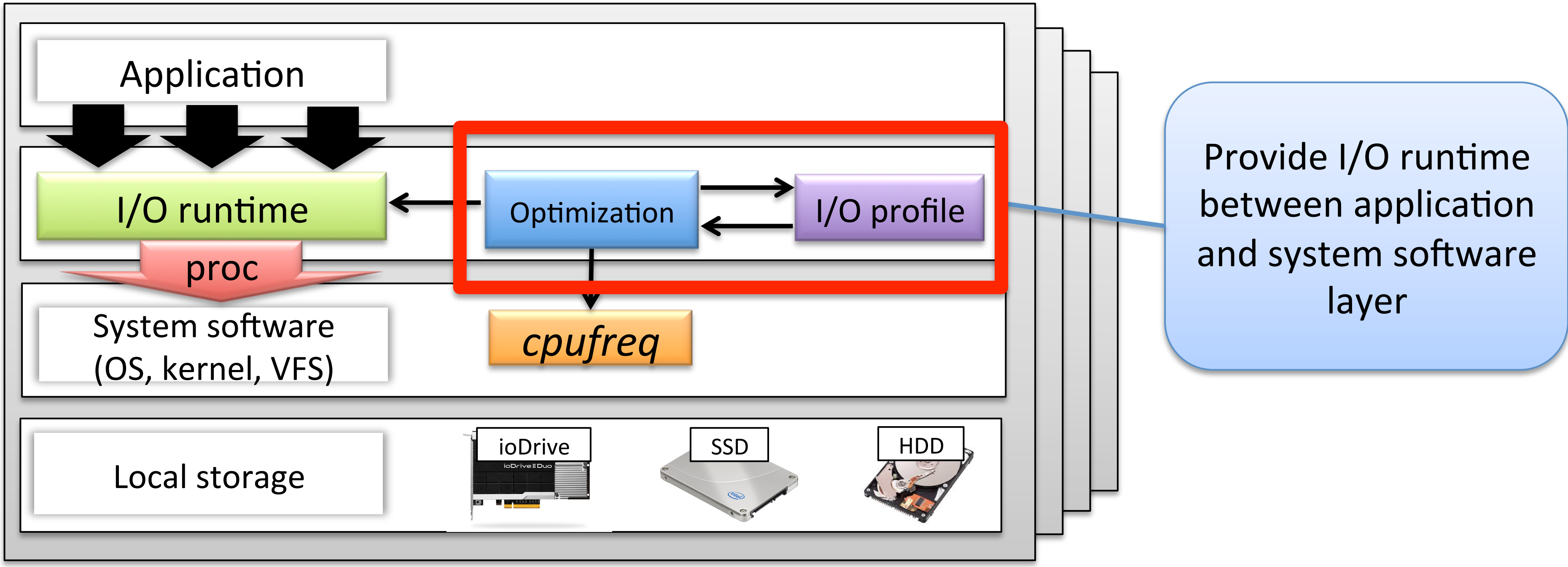
$$\begin{pmatrix} T : \text{I/O type} \\ D : \text{I/O device} \\ P : \#\ \text{of procs} \\ F : \text{CPU freq} \end{pmatrix} \Rightarrow \text{I/O profile} \quad f : (T, D, P, FV) \rightarrow (W, Th) \Rightarrow \begin{pmatrix} W : \text{Power (w)} \\ Th : \text{Throughput} \end{pmatrix}$$

Summary of the energy-aware optimization

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



Design overview of Energy-aware I/O system



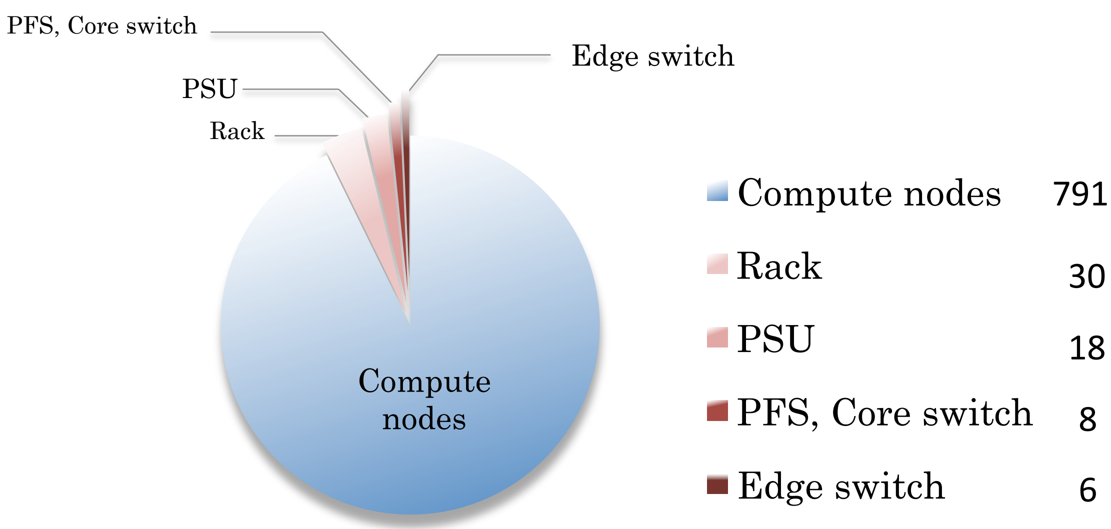
- This work investigate how much our energy-aware I/O optimization can improve energy efficiency

Outline

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

Experimental settings

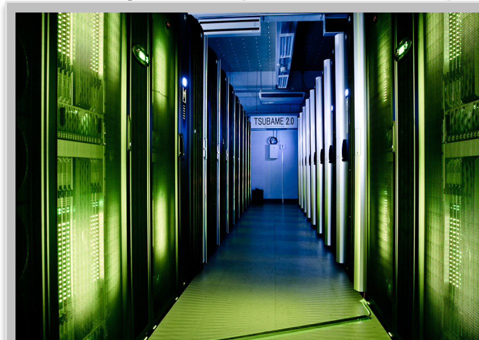
- Checkpoint size: 64GB per node
- Application's power consumption (W_A) : 471.1 W
  - NAS Parallel Benchmark (SP: Class C)
- Failure rate: $\lambda = 1.89 \times 10^{-5}$ (MTBF = 14 hours)



Failure analysis on TSUBAME2.0

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

TSUBAME2.0, 14th in Top500 (June 2012)



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

Experimental settings (cont'd)

- Compare the proposed method (profile lookup) with three other strategies supported by *cpufreq*

Compared cpufreq governor

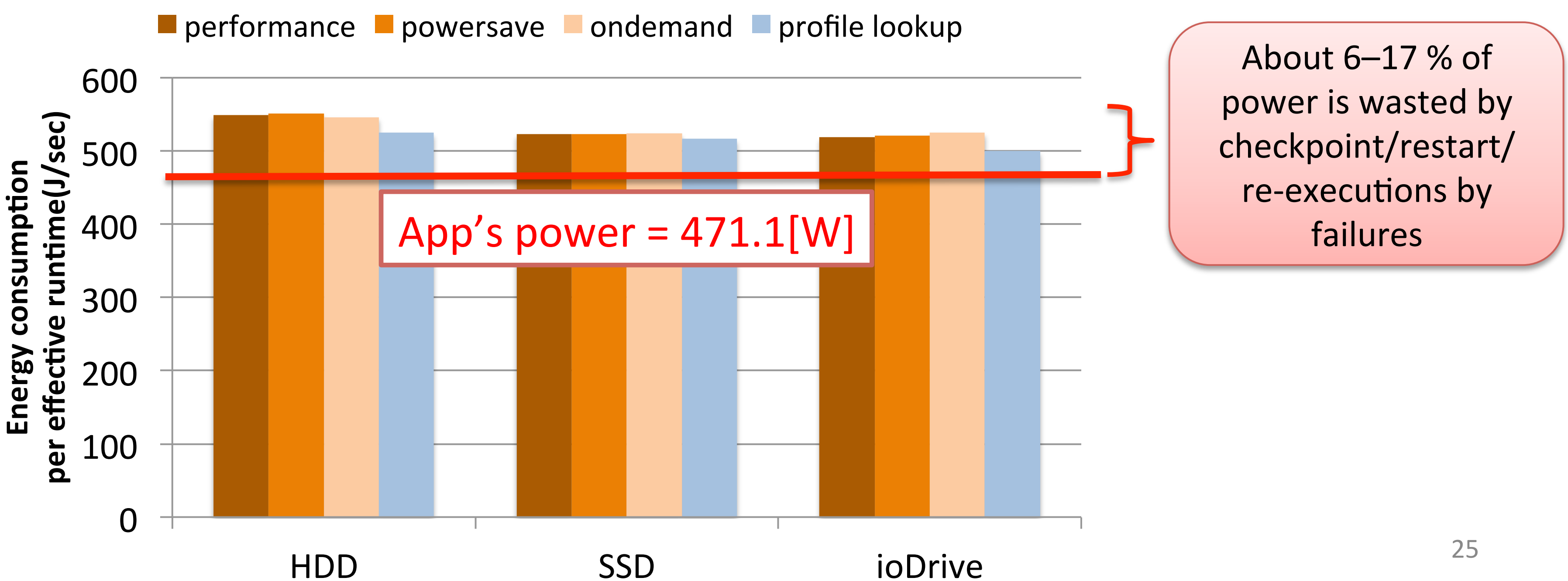
Profile lookup	Proposed energy-aware I/O optimization method
Performance	Set CPU frequency to maximum supported frequency regardless of CPU usage
Powersave	Set CPU frequency to the lowest supported frequency regardless of CPU usage
Ondemand	Adjust CPU frequency according to CPU usage

- 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)

$$EPE = \frac{T_{\bar{A}} \cdot W_A + T_{\bar{C}} \cdot W_C + T_{\bar{R}} \cdot W_R}{T_A}$$

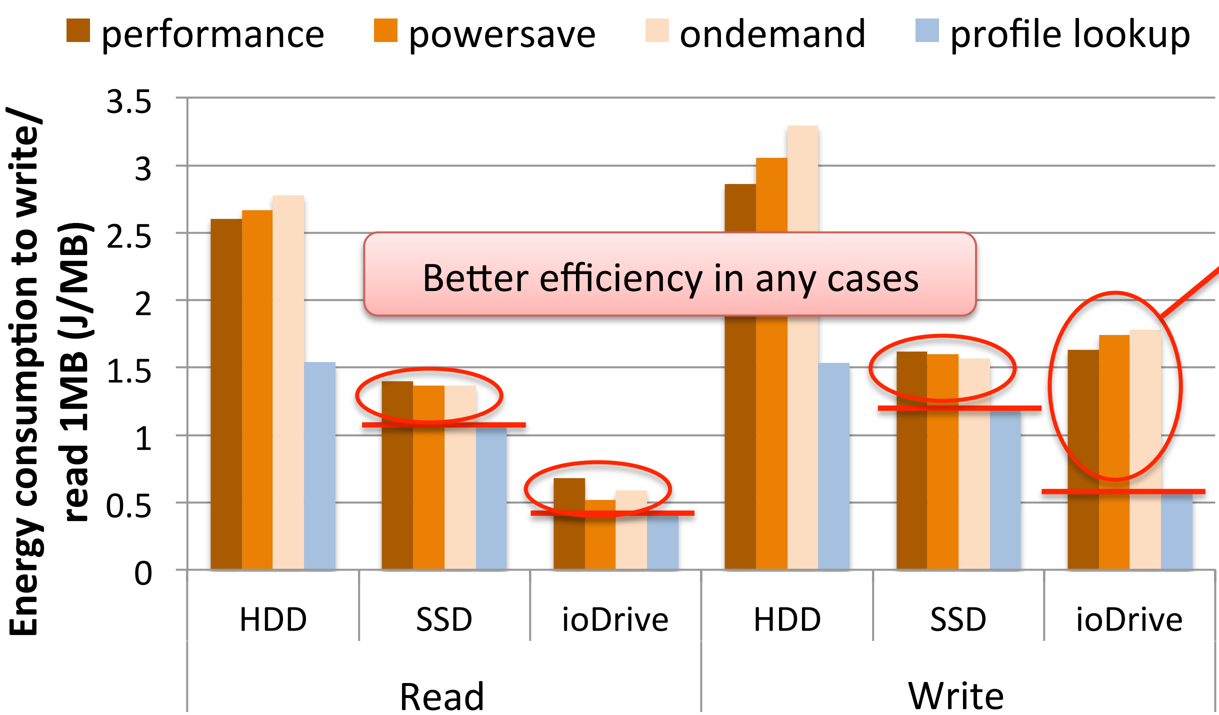
Energy efficiency comparison

- Our propsed 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



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)



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

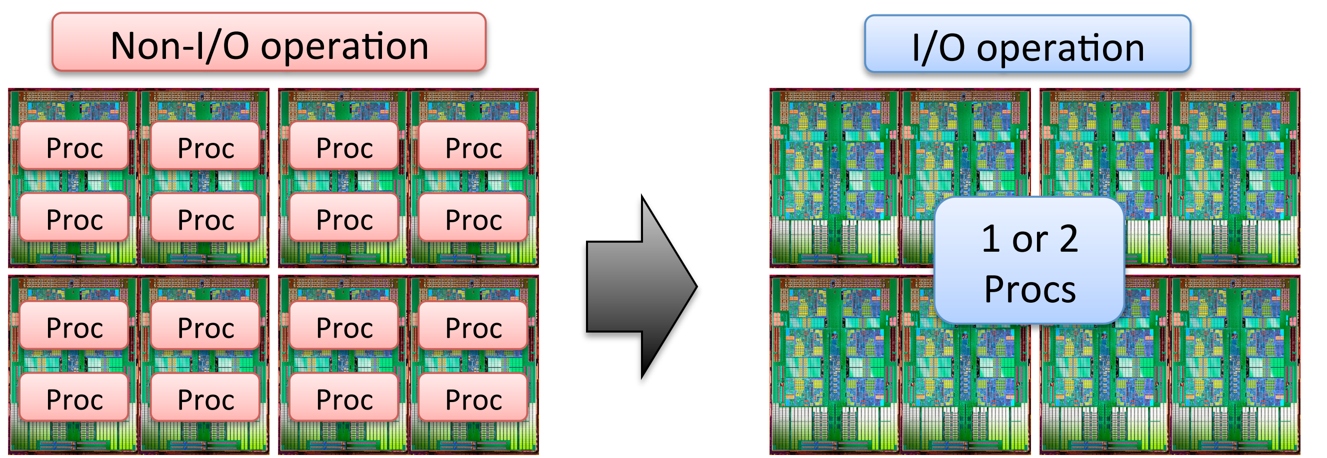
Energy-aware optimal CPU frequency & # of procs

	Read			Write		
	HDD	SSD	ioDrive	HDD	SSD	ioDrive
CPU freq	1.7	1.4	2.1	2.1	1.7	1.4
# of procs	1	1	2	1	1	2

CPU freq range

0.8GHz
1.1GHz
1.4GHz
1.7GHz
2.1GHz

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



CPU frequency: 2.1 [GHz]

CPU frequency: 1.4 ~ 2.1 [GHz]

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 machie 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.