FLSCHED: A Lockless and Lightweight Approach to OS Scheduler for Xeon Phi

Heeseung Jo	Chonbuk National University
Woonhak Kang	Georgia Institute of Technology
Changwoo Min	Virginia Tech
Taesoo Kim	Georgia Institute of Technology

Motivation

Growth of Manycore Processors

- Processor manufacturers have increased the number of cores
- Manycore processors are now prevalent
  - in all types of computing devices
  - include mobile devices, servers and h/w accelerators
- Intel Xeon Phi has up to **76 cores, 304 threads**

Motivation

Intel Xeon Processors vs. Xeon Phi Processors

	Xeon Processors	Xeon Phi Processors
Cores	Up to 24 cores	Up to 76 cores
Threads	Up to 48 threads	Up to 304 threads
Vector Registers	16 * 512-bit registers	32 * 512-bit registers

- 3.17x more cores
- 6.33x more threads
- 2x more registers

Motivation

Inefficiency of Existing Schedulers

- When CFS scheduler was introduced, **4-core** servers were dominant in datacenters
- Now, **32-core** servers are standard in data centers
- Moreover, **more than 100 cores** are becoming popular

Motivation

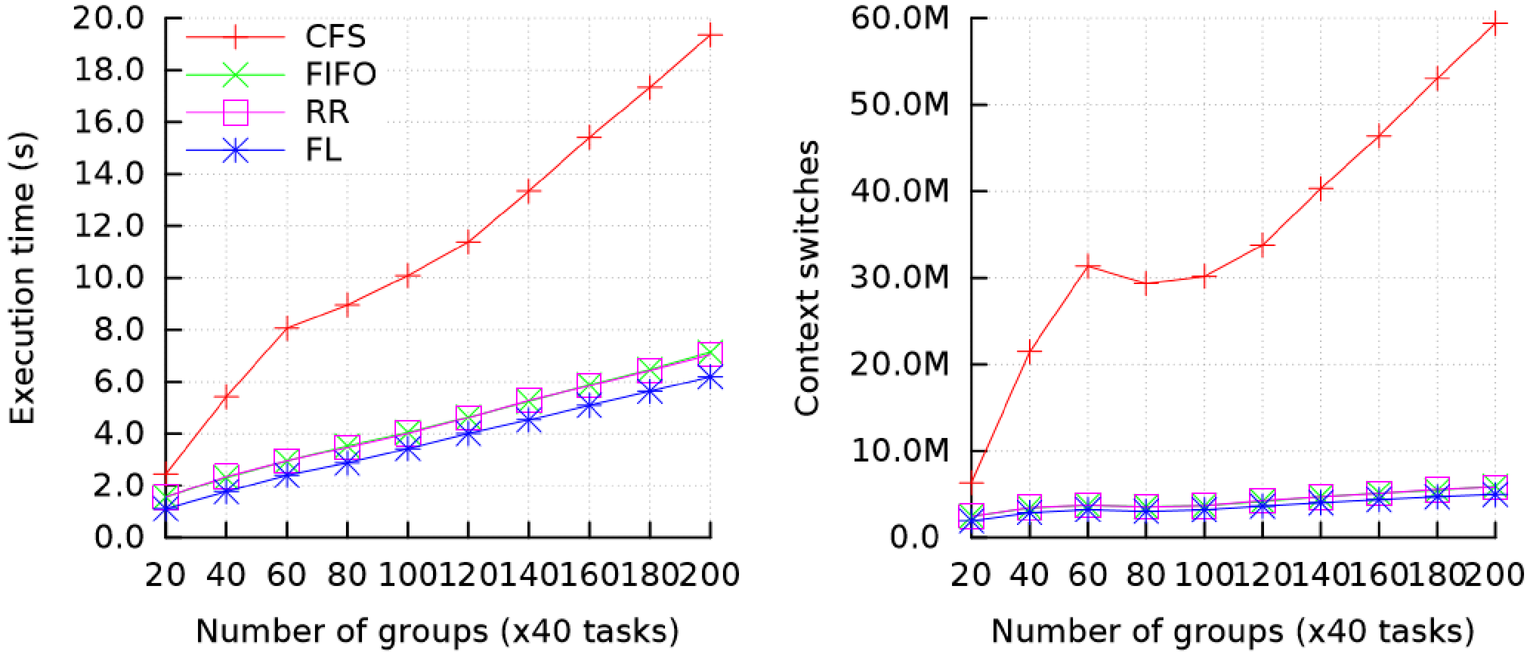
Inefficiency of Existing Schedulers

- The revolution of OS schedulers is **slow** to follow up emerging manycore processors
  - They have various lock primitives
  - Frequent context switches
  - But, these are less important in manycore processors like Xeon Phi
- **Due to these issues, we propose the new OS scheduler, FLSCHED**
  - Lockless design
  - Less context switches

Motivation

Inefficiency of Existing Schedulers

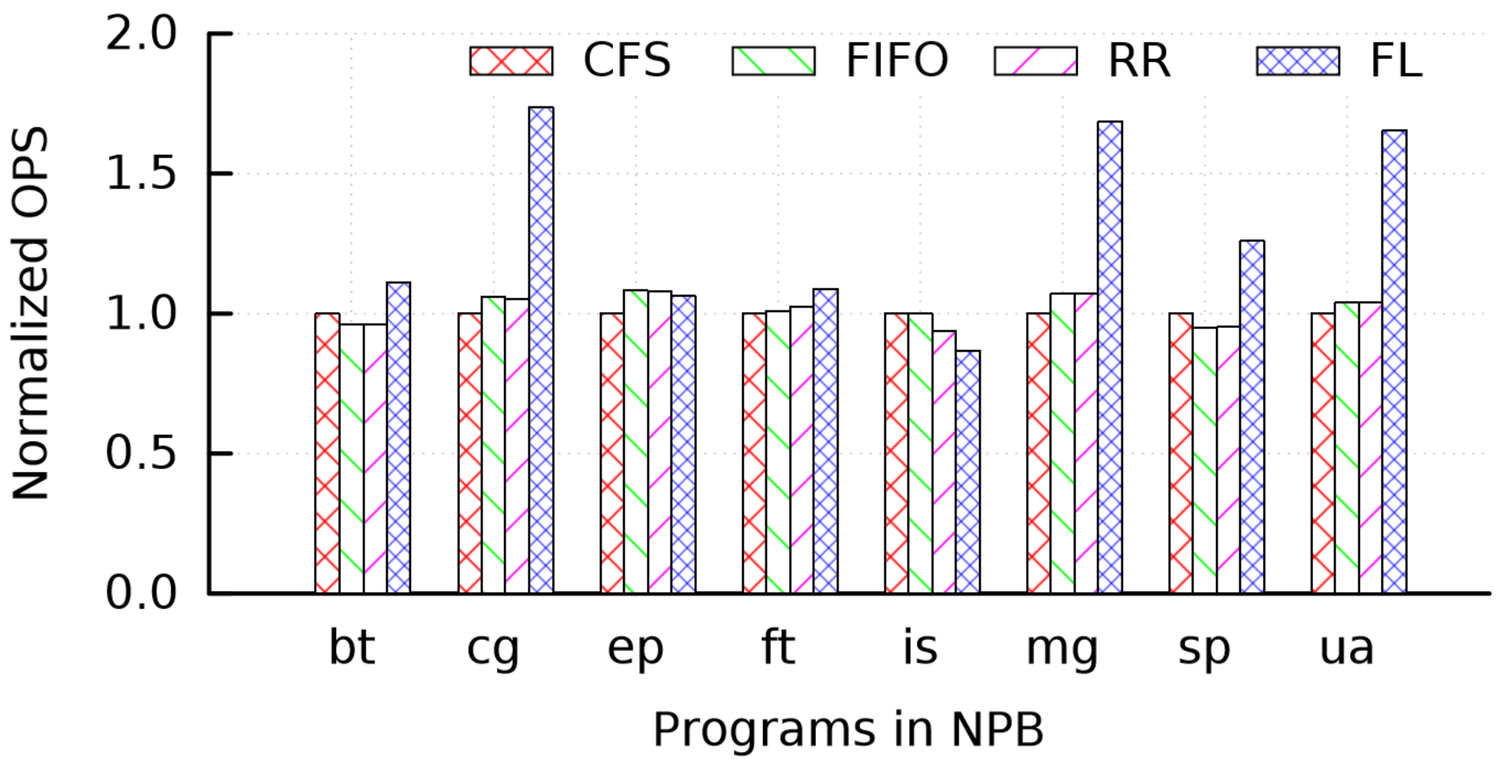
- Hackbench on a Xeon Phi
- Frequent context switches → slower



Motivation

Inefficiency of Existing Schedulers

- Comparison on NAS Parallel Benchmark
- Locks in the schedulers degrade the performance



Design

FLSCHED

- Feather-Like Scheduler
- Designed for manycore processors
  - like Intel Xeon Phi
- Lockless design
- Minimizing the number of context switches

Design

Locklessness

Lock types	CORE	CFS	FIFO/RR	FL
raw_spin_lock	16	1	12	-
raw_spin_lock_irq/irqsave	13	5	2	-
rcu_read_lock	14	5	1	-
spin_lock	-	-	-	-
spin_lock_irq/irqsave	12	-	-	-
read_lock	3	-	-	-
read_lock_irq/irqsave	1	-	-	-
mutex_lock	6	-	-	-
Total	**65**	**11**	**15**	**0**

- Core scheduler code includes highest number of locks
- FLSCHED is **implemented without locks in itself**
  - by restructuring and optimizing the mechanisms

Design

Locklessness: Comparing to RR

- 2 locks are for the runtime statistics
  - It is NOT critical to make scheduling decisions on Xeon Phi
- 5 locks are to balance the load of cores
  - FLSCHED doesn’t use periodic load balance
- 8 locks are used for bandwidth control mechanism
  - It is not important features for Xeon Phi
- Now, We removed 15 locks
  - Since Xeon Phi processors are mostly used for HPC

Design

Less Context Switches

- FLSCHED delays all settings of the reschedule flag to avoid context switches as many as possible
- Computation throughput is MORE important than responsiveness, and fairness
  - Since Xeon Phi processors are mostly used for HPC

Design

Less Context Switches

- Most of preemption is incurred by priority
  - Priority preemption is NOT crucial for Xeon Phi
- FLSCHED does not immediately perform preemption
  - Instead, FLSCHED moves the location of tasks in runqueues and performs normal task switches in later term
  - Since Xeon Phi processors are mostly used for HPC

Design

Faster and efficient scheduling decision

- Scheduling information updates are minimized
    - To make scheduler faster and more efficient
- Remove “update_curr_fair” function
    - It takes very short time
    - But it is called huge number of times with a spinlock
    - It can be non-negligible overhead in manycore processors
- Instead, FLSCHED works based on a given time slice with RR

Design

Faster and efficient scheduling decision

- FLSCHED does not provide 3 scheduling features:
  - Control groups
  - Group scheduling
  - Autogroup scheduling
- These are considered NOT important features for manycore systems like Xeon Phi
  - To get the great performance improvement, sometimes we have to yield small things

Evaluation

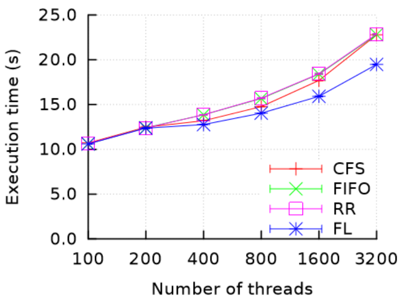
Evaluation Environments

- Intel Xeon E5-2699
    - 18 cores
    - 36 threads
- 64 GB main memory
- Intel Xeon Phi 31S1P
    - 57 cores
    - 228 threads
    - 8 GB internal memory

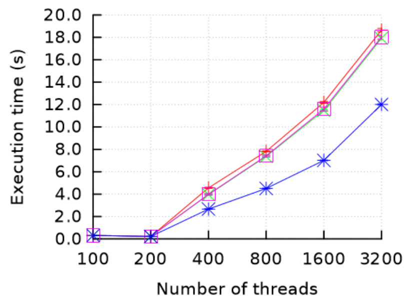
Evaluation

Performance comparison of NAS Parallel Benchmark

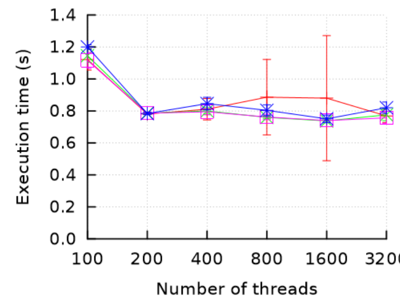
- It shows better performance with FLSCHED



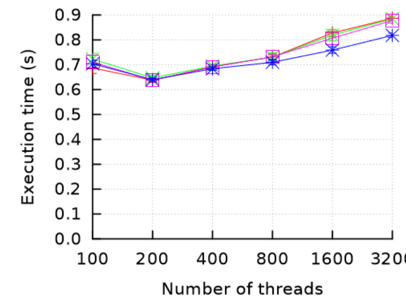
(a) bt



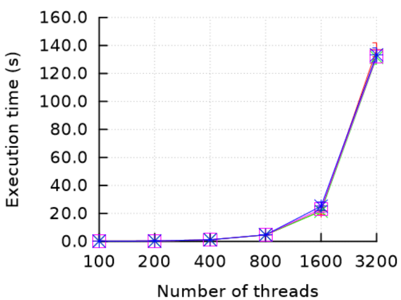
(b) cg



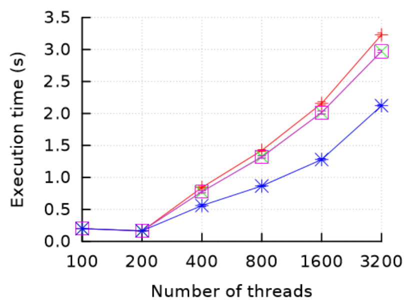
(c) ep



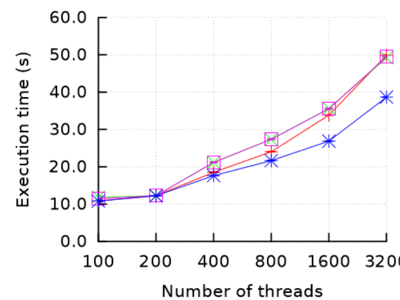
(d) ft



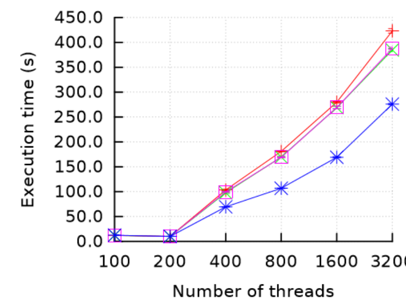
(e) is



(f) mg



(g) sp



(h) ua

Evaluation

Performance comparison of NAS Parallel Benchmark

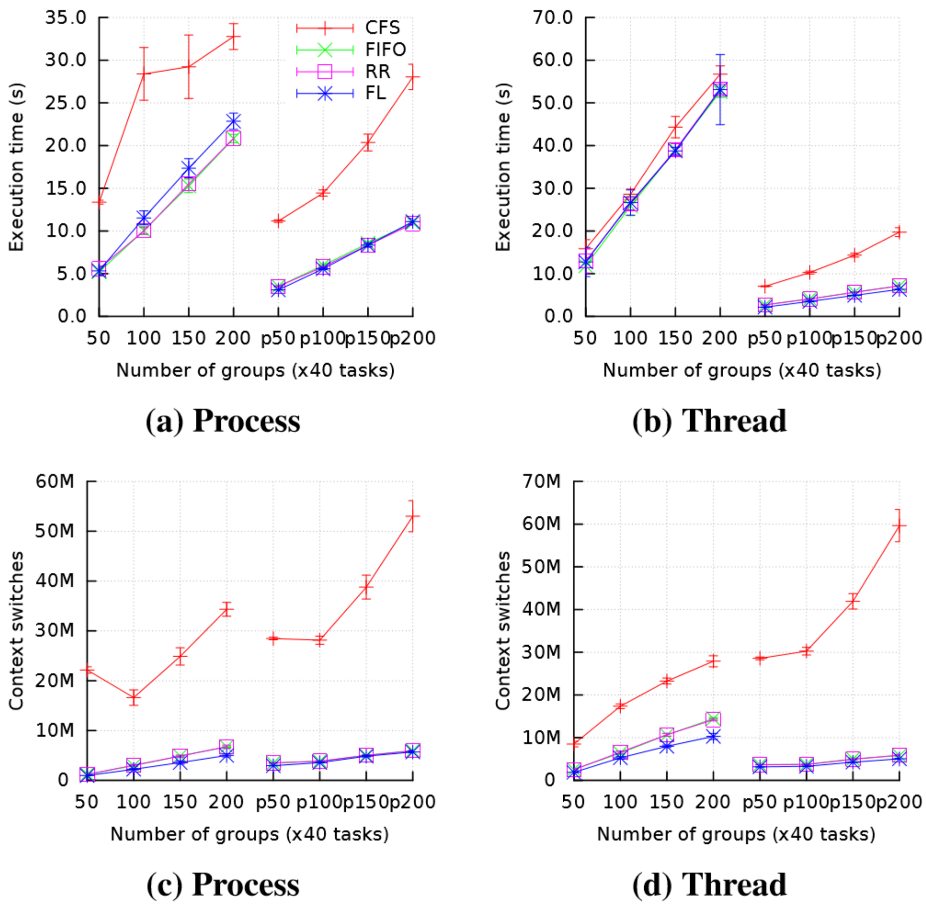
- Execution time of spinlock while executing NPB

NPB program	CFS (%)	FIFO (%)	RR (%)	FLSCHED (%)
bt	7.29	8.53	8.60	3.05
cg	**10.73**	**13.61**	**12.93**	**4.11**
ep	0.97	0.89	0.91	1.10
ft	5.34	5.25	5.57	4.04
is	0.21	0.17	0.18	0.12
mg	**6.84**	**7.30**	**7.15**	**2.85**
sp	**8.23**	**9.95**	**9.98**	**3.58**
ua	**14.63**	**15.79**	**15.46**	**5.96**

Evaluation

Performance comparison of hackbench

- Execution time and number of context switches



(a) Process (b) Thread

(c) Process (d) Thread

One group uses 40 tasks

In X axis,

'p' with the number denotes pipe
The other denotes socket

Evaluation

Performance comparison of hackbench

- Execution count and time of scheduler functions

Scheduler functions	CFS		FLSCHED		Normalized ratio
	Count	Average time (ns)	Count	Average time (ns)	(Average time)
check_preempt	42,184,784	5,058	3,202	917	0.18
dequeue_task	42,476,857	**19,008**	10,646	**3,636**	**0.19**
enqueue_task	42,479,016	**17,314**	10,792	**4,169**	**0.24**
pick_next_task	66,951,729	**5,261**	5,532,392	**1,937**	**0.37**
pre_schedule	-	-	10,646	718	-
put_prev_task	66,503,232	6,185	10,647	1,138	0.18
select_task_rq	42,426,871	**10,837**	8,031	**2,549**	**0.24**
set_cpus_allowed	-	-	1	2,997	-
task_tick	906,640	13,131	112	1,042	0.08
task_waking	42,418,867	2,290	10,792	1	0.00
update_curr	342,354,453	**2**	-	**-**	**0.00**

Total Execution Time:
CFS: 28.037s
FLSCHED: 11.102s

Conclusion

FLSCHED

- Feather-Like Scheduler
  - Designed for manycore processors like Intel Xeon Phi
  - Lockless design
  - Minimizing the number of context switches
- FLSCHED shows better performance than CFS up to
  - 1.73x for HPC applications
  - 3.12x for micro-benchmarks

Thank you

If you have any questions,
Please contact the first author via email:

Prof. Heeseung Jo
heeseung@jbnu.ac.kr