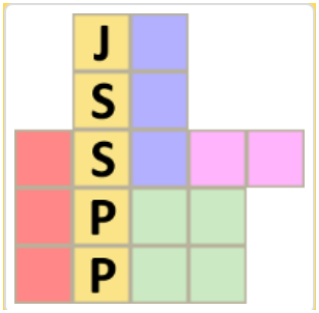


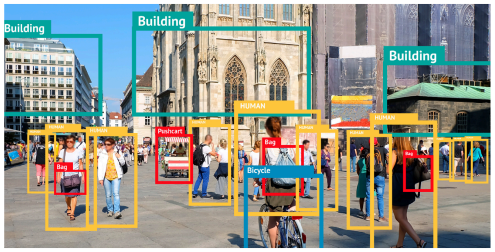
Towards Hybrid Isolation for Shared Multicore Systems

23rd Workshop on Job Scheduling Strategies for Parallel Processing, (JSSPP), 2020

Yoonsung Nam†, Byeonghun Yoo†, Yongjun Choi†,
Yongseok Son§, and Hyeonsang Eom†

Seoul National University† **Chung-Ang University**§

Co-locating Workloads in a Server Machine

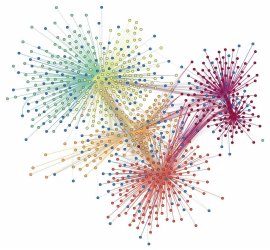


object detection

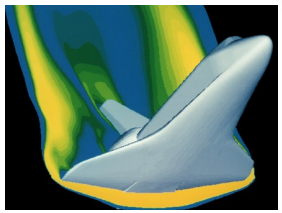
APACHE HTTP SERVER PROJECT
web server

speech recognition

redis
in-memory DB



graph processing



scientific simulation

hadoop
data analytics

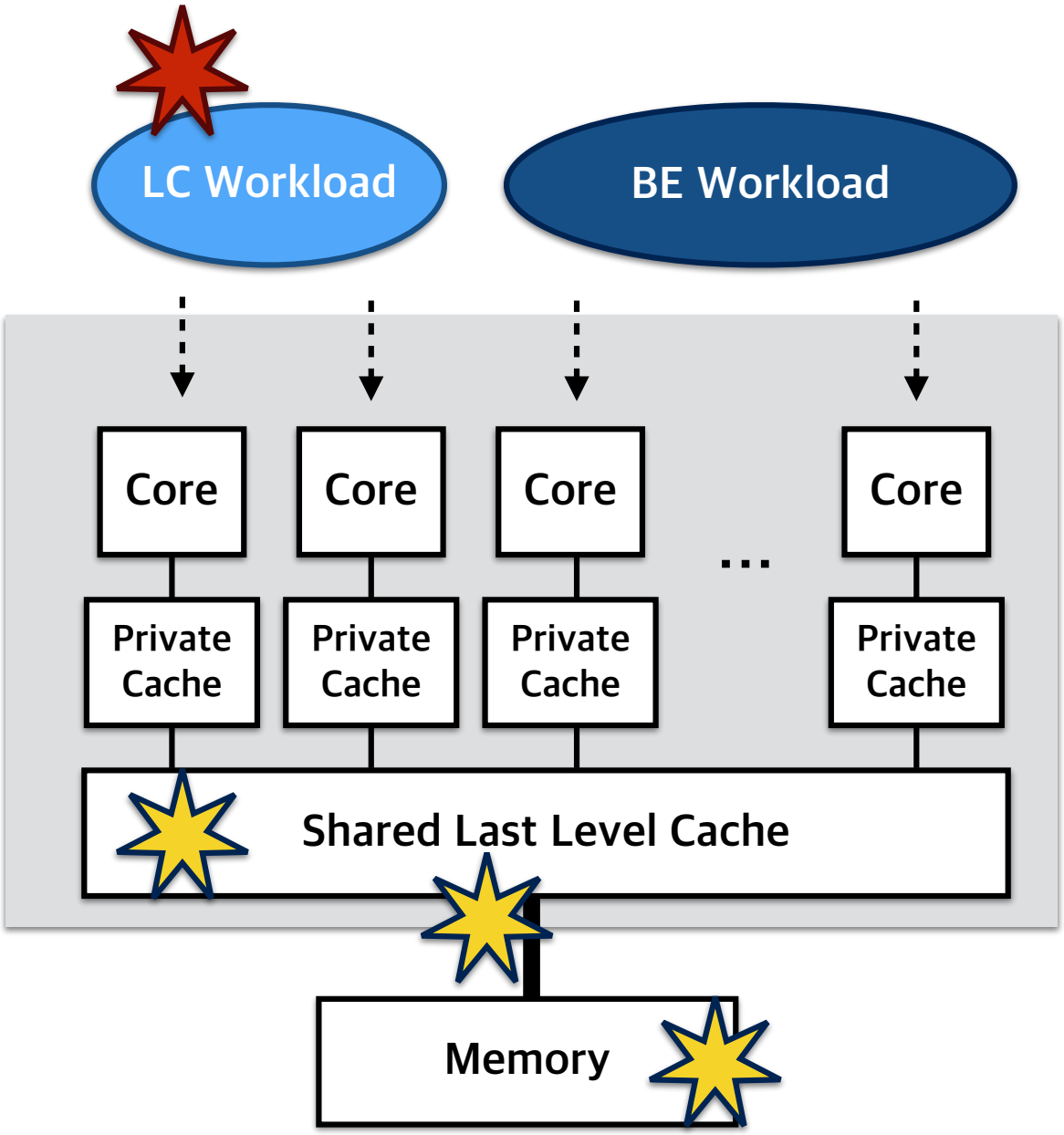
Latency-critical Workloads

Best-Effort Workloads

Multicore Server Machine

- **Colocating multiple workloads in a server**
  - Latency-critical
  - Best-effort
- **Benefits**
  - Higher resource efficiency
  - Running multiple workloads in parallel

Shared Resource Contention in Multicore Systems



- **Problem: Resource Contention**
  - Lower performance
  - Lower resource efficiency
  - Higher service level objective violations

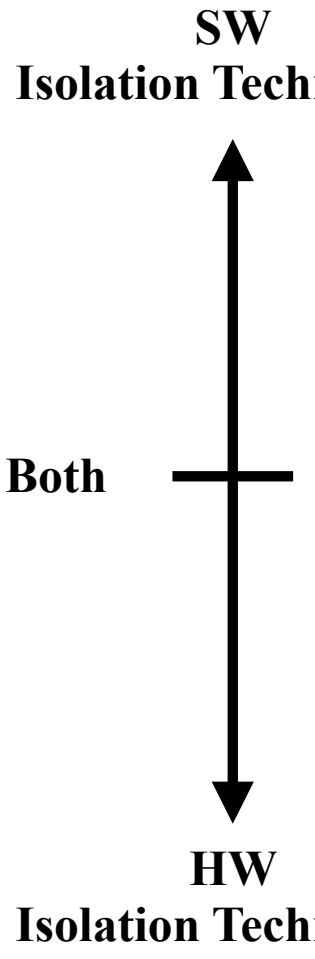
Isolation Techniques

- **Software isolation techniques**
  - Isolation techniques performed by software
  - `cgroup` is one of the representative isolation knob in Linux
  - e.g., ***CPU core allocation***, ***CPU cycle limit***, and ***Thread migration***
- **Hardware isolation techniques**
  - Isolation techniques performed by hardware
  - Hardware vendors provide their own isolation interfaces (e.g., ***Intel CAT*, Intel per-core DVFS,*** and ***Intel RAPL****)
  - e.g, ***Cache partitioning*** and ***Per-core Dynamic Voltage Frequency Scaling (DVFS)***

**CAT: Cache Allocation Technology*
***RAPL: Running Average Power Limit*

Related Work

- **How these isolation techniques have been used to mitigate resource contention?**



CPI2[EuroSys'13]
(CPU cycle limit & Migration)

Memguard[RTAS'13]
(CPU cycle limit)

Quasar[ASPLOS'14]
(CPU allocation & Migration)

Heracles[ISCA'15]
(CPU allocation, Intel CAT, Network HTB*, CPUFreq.)

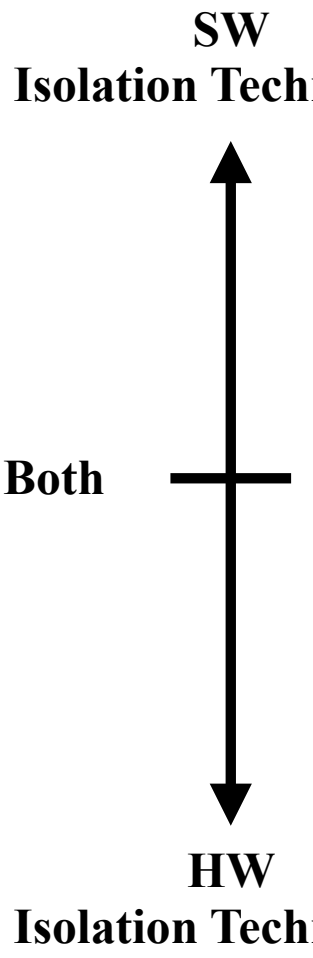
PARTIES[ASPLOS'19]
(CPU allocation, Intel CAT, Network HTB*, Disk qdisc)

Dirigent[ASPLOS'16]
(Intel CAT & per-core DVFS)

*HTB: Hierarchical Token Bucket

Related Work

- **How these isolation techniques have been used to mitigate resource contention?**



CPI2[EuroSys'13]
(CPU cycle limit
& Migration)

Memguard[RTAS'13]
(CPU cycle limit)

Quasar[ASPLOS'14]
(CPU allocation
& Migration)

Heracles[IS
(CPU allocation
Network HTB*

HIS[JSSPP'20]
(CPU allocation, Intel CAT,
CPU Freq, Thread Migration)

TIES[ASPLOS'19]
allocation, Intel CAT,
rk HTB*, Disk qdisc)

Dirigent[ASPLOS'16]
(Intel CAT & per-core DVFS)

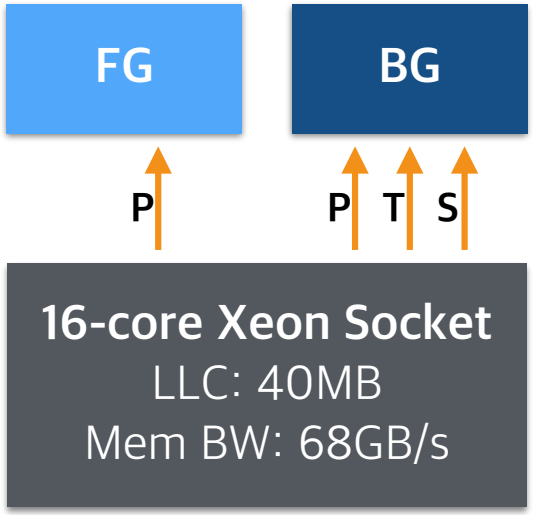
*HTB: Hierarchical Token Bucket

Trade-offs in Existing Isolation Techniques

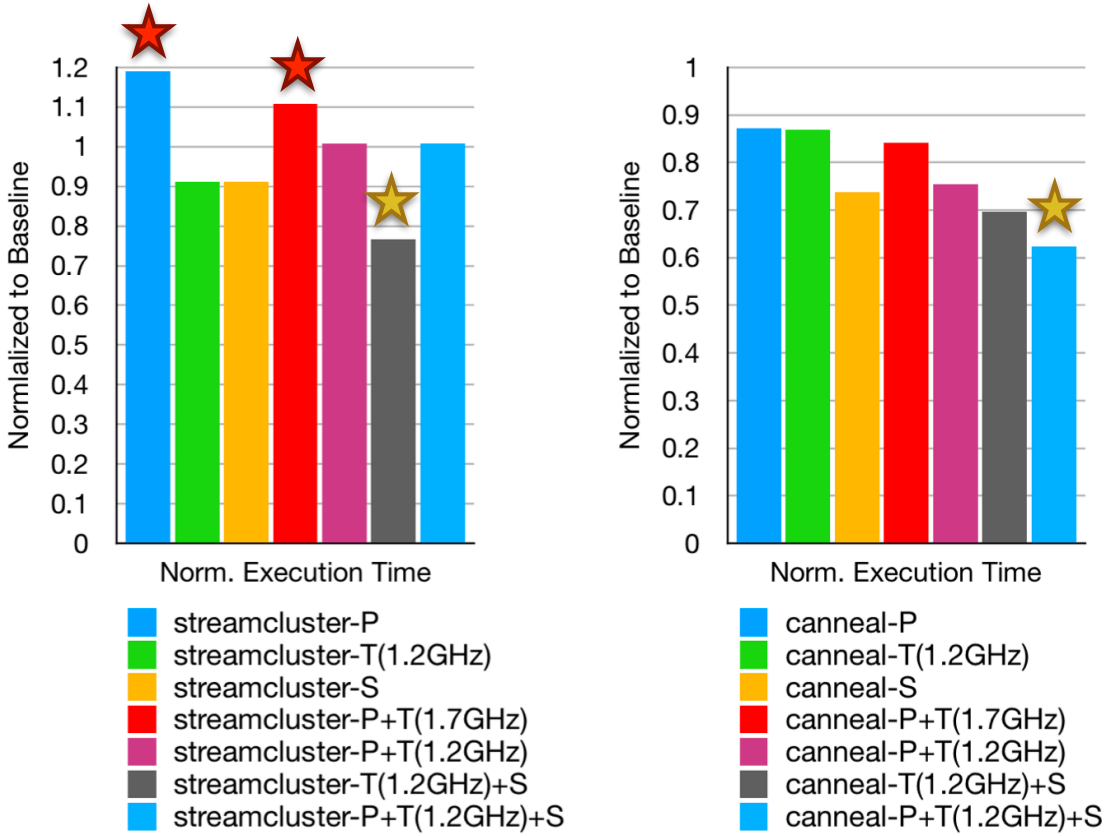
	H/W isolation tech.		S/W isolation tech.		
	Cache Partitioning	**Per-core DVFS**	**CPU Cycle Limit**	**CPU Allocation**	**Thread Migration**
Technology	Intel CAT	Intel Processor (since Haswell)	cgroup:cpu	cgroup:cpuset	cgroup:cpuset, memory
Type	Partitioning	Throttling	Throttling	Scheduling	Scheduling
Latency (ms)*	**3**	**2**	**40~50**	**3**	**90**
Configurations	# of ways (20 per LLC)	# of freq. (10 per core)	quota/period (100)	# of cores (16)	# of sockets (2)
Strictness	High	High	Medium	Medium	Low
Responsiveness	High	High	Medium	High	Low
Flexibility	Low	Low	Medium	High	High

***Latency is time taken for each isolation technique to work on a high memory-intensive workload (*SP* of NPB)**

Ineffective Isolation



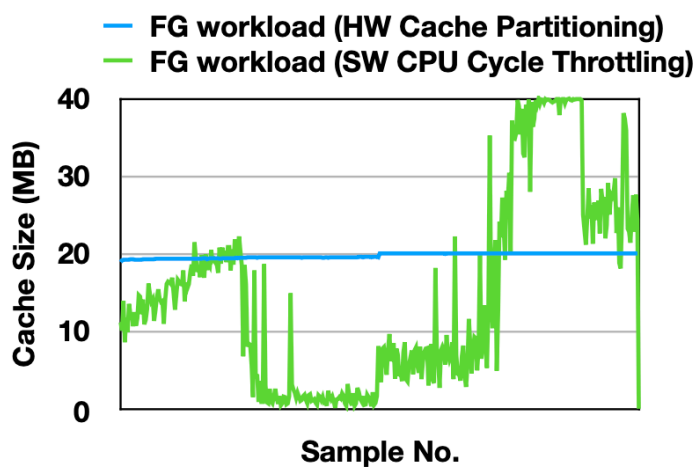
Partitioning(P): Intel CAT
Throttling(T): Per-core DVFS
Scheduling(S): CPU allocation



- **FG**: *streamcluster*(MemBW-int) & *canneal*(LLC-int), **BG**: *SP* (MemBW-int & LLC-int)
- Both workloads show **the highest performance** with ***the combination of different isolation techniques*** (indicated by **yellow stars**)
- Some isolation technique combinations show worse performance than baseline that just allocates 8-cores evenly (indicated by **red stars**)

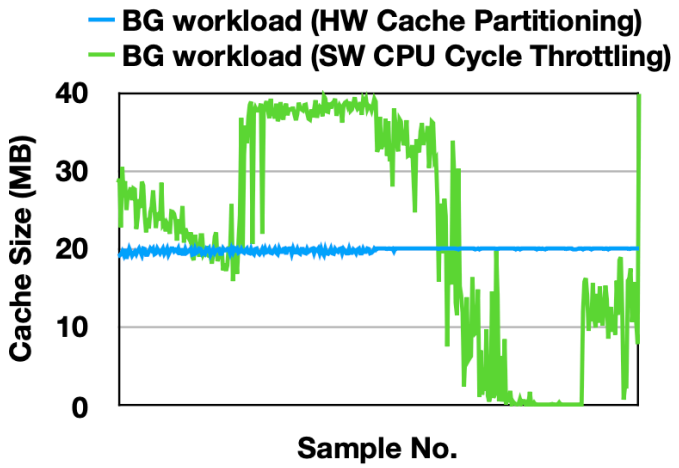
Trade-off: Strictness

FG: streamcluster

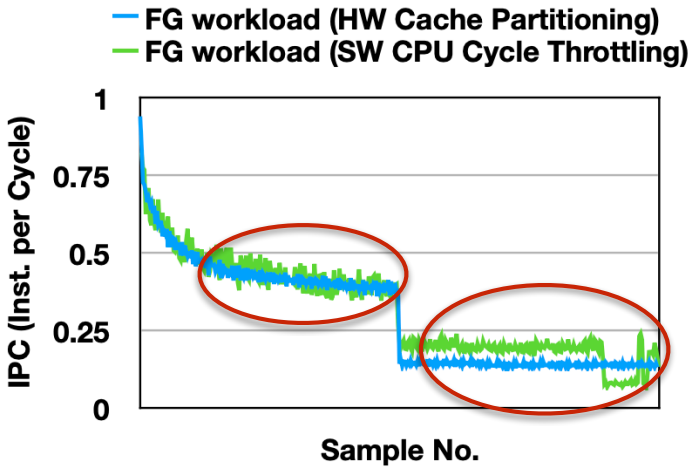


Changes in FG's LLC

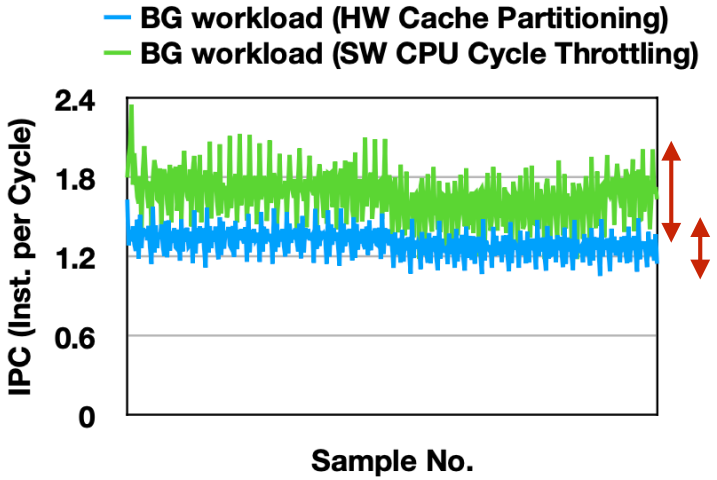
BG: SP



Changes in BG's LLC



Changes in FG's IPC

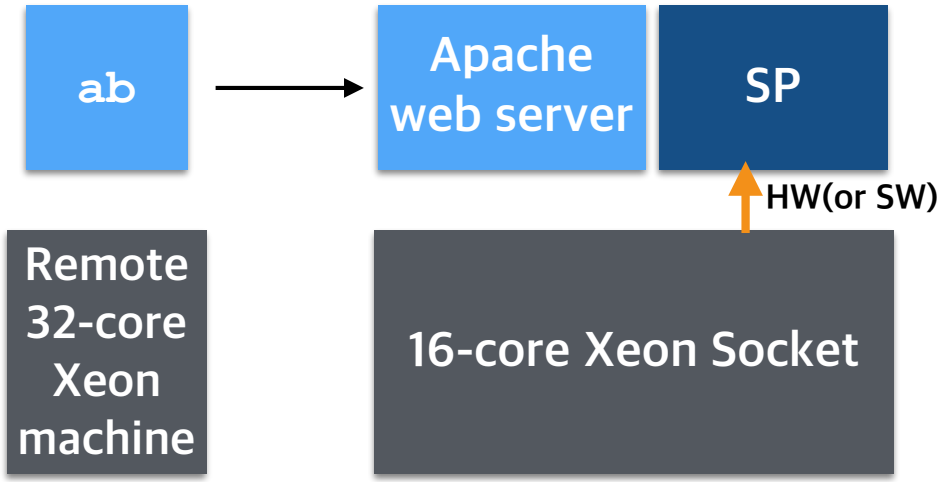


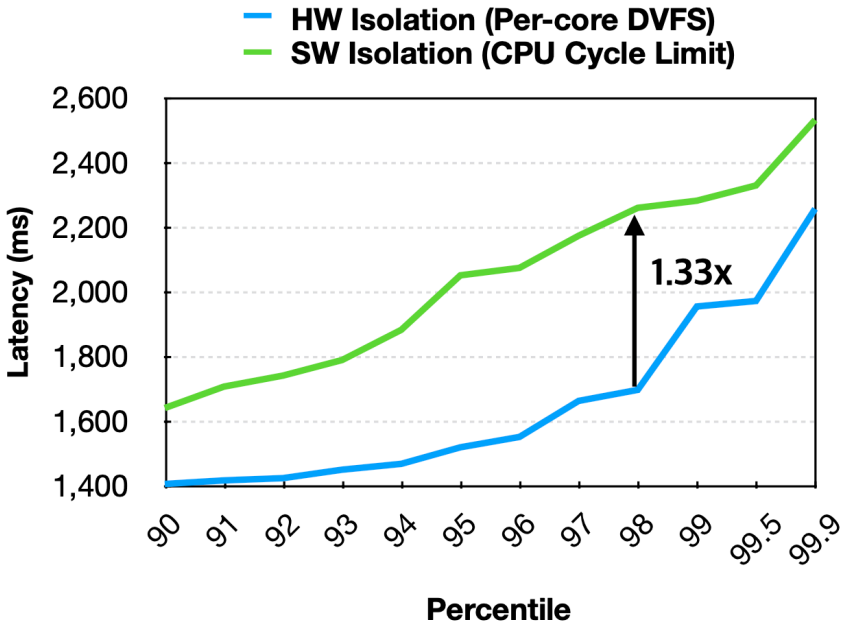
Changes in BG's IPC

*LLC sizes are measured by *Intel Resctrl*

Trade-off: Responsiveness

FG: Apache web server (client: `ab`) **BG: SP**

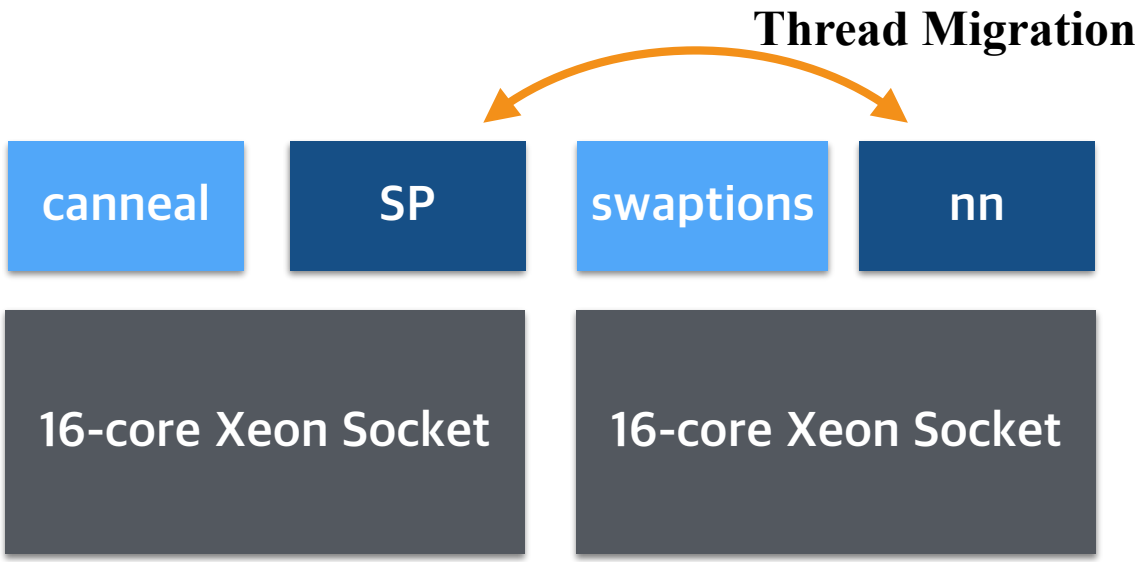


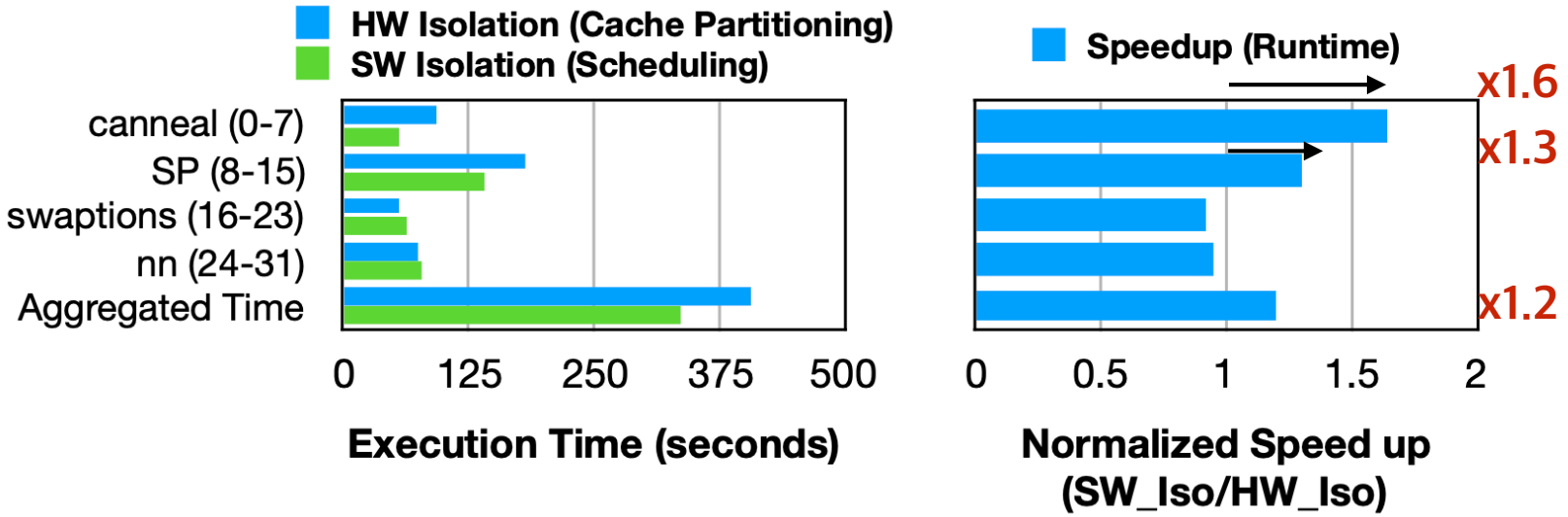


- HW isolation technique (Per-core DVFS) shows ***lower tail latency*** for the latency-critical workload than SW isolation technique (CPU cycle limit)

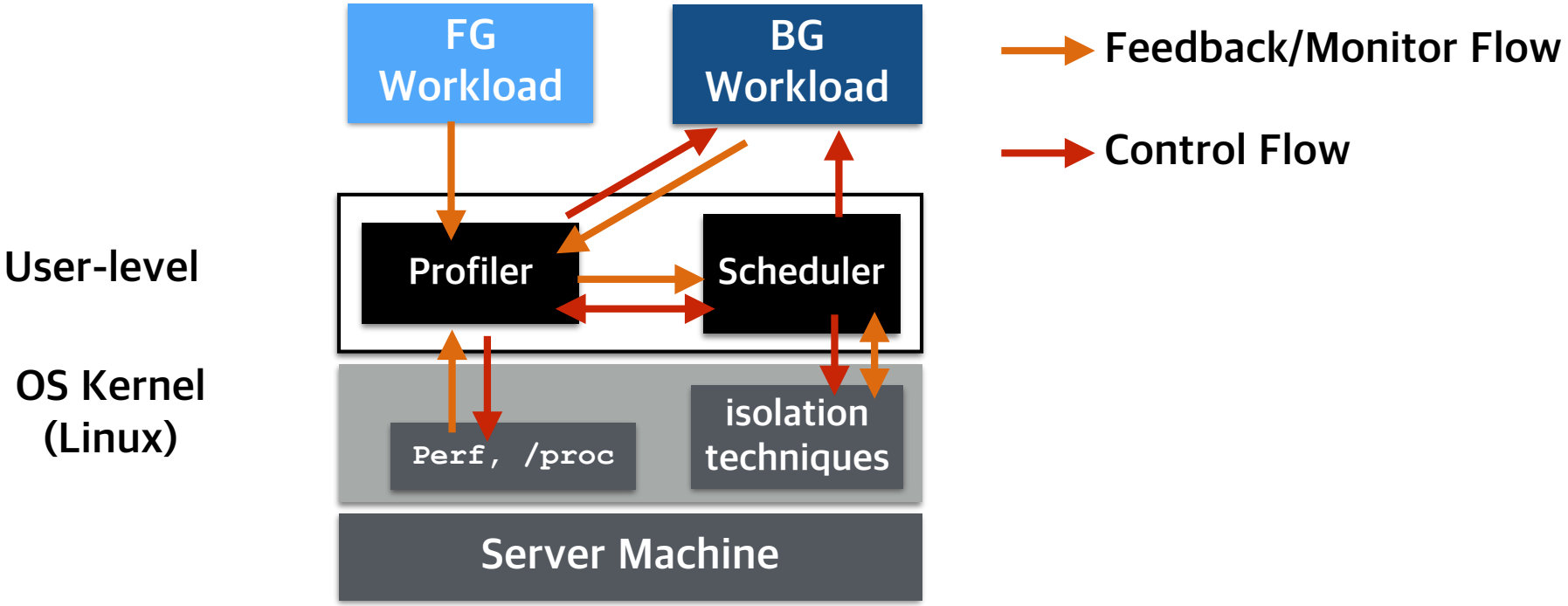
- Because, the hardware isolation techniques enables quick and faster isolation by controlling hardware directly

Trade-off: Flexibility





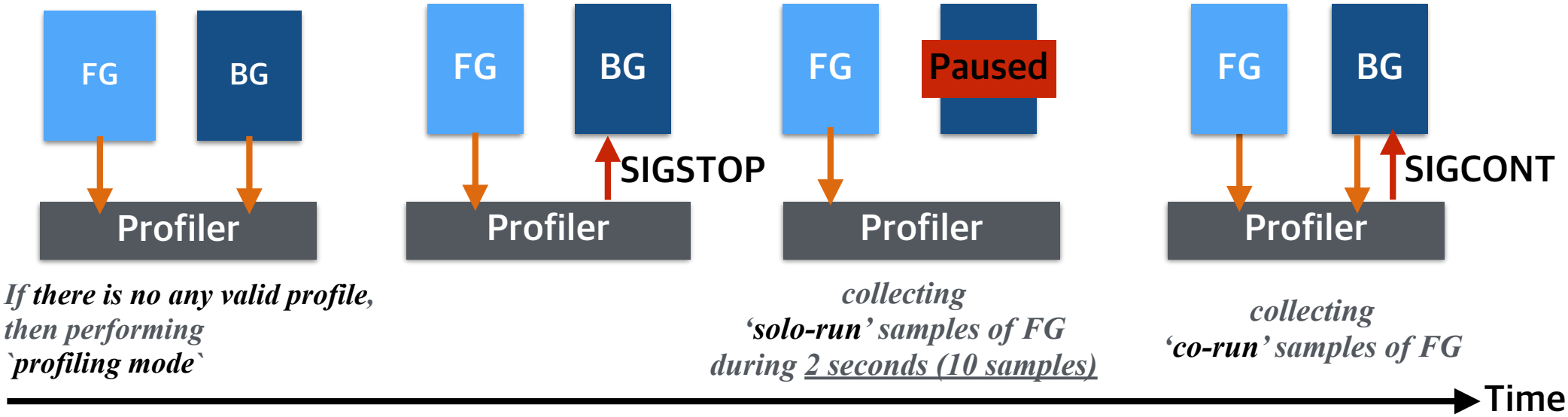
HIS: Hybrid Isolation System



- **Profiler**: monitors ***resource contention*** for the ***foreground*** workload in an online manner

- **Scheduler**: chooses an isolation technique based on ***the most contentious resource, available isolation techniques, and type of techniques (SW or HW)***

HIS Online Profiler



- **Profiler:**
  - Measuring **how much** workloads **suffer from** resource contentions by calculating contention using samples
    $res_{cont} = (res_{co\text{-}run}/res_{solo\text{-}run}) - 1$
  - Running 'profiling mode' if there is no any valid profile (e.g., phase change, no profile data)

Contention Type	Metric	Meaning
LLC	LLC hit ratio	Data reuse
Mem BW	memory BW	BW consumption
CPU Core	free/allocated CPUs, active threads	CPU demand

HIS Scheduler

Contention Type	Isolation techniques
LLC	Intel CAT
Mem BW	CPU Allocation or Per-core DVFS
CPU Core	CPU Allocation or Thread migration

Scheduler

1. Which is **the most contentious** one? — MemBW
2. Which isolation techniques are **available**? — CPU allocation or Per-core DVFS
3. **HW** or **SW** Isolation? — CPU allocation (SW) Per-core DVFS (HW)
4. Isolation technique is selected — Per-core DVFS (HW)

2.1GHz -> 2.0GHz

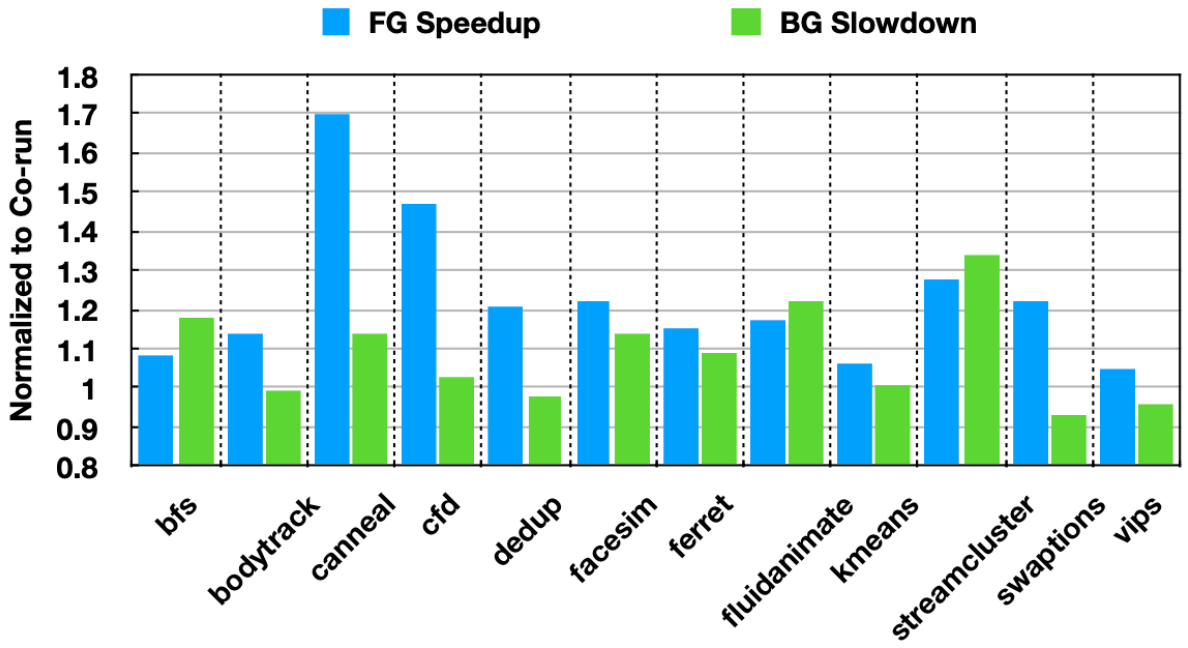
Scheduler

5. Which parameter? (**strengthen** or **weaken**) — Strengthens Memory BW Isolation! Lower CPU freq. (2.1GHz -> 2.0GHz)
6. Get **feedback** from **profiler** — Sleeps 200ms
7a. **Iterate step 1 ~ 6** by the feedback — If still, Mem BW contention is high then go step 2
7b. **If contentions** are **well managed** within the threshold, then **stop** — If the highest contentions are within 5%

Experimental Setup

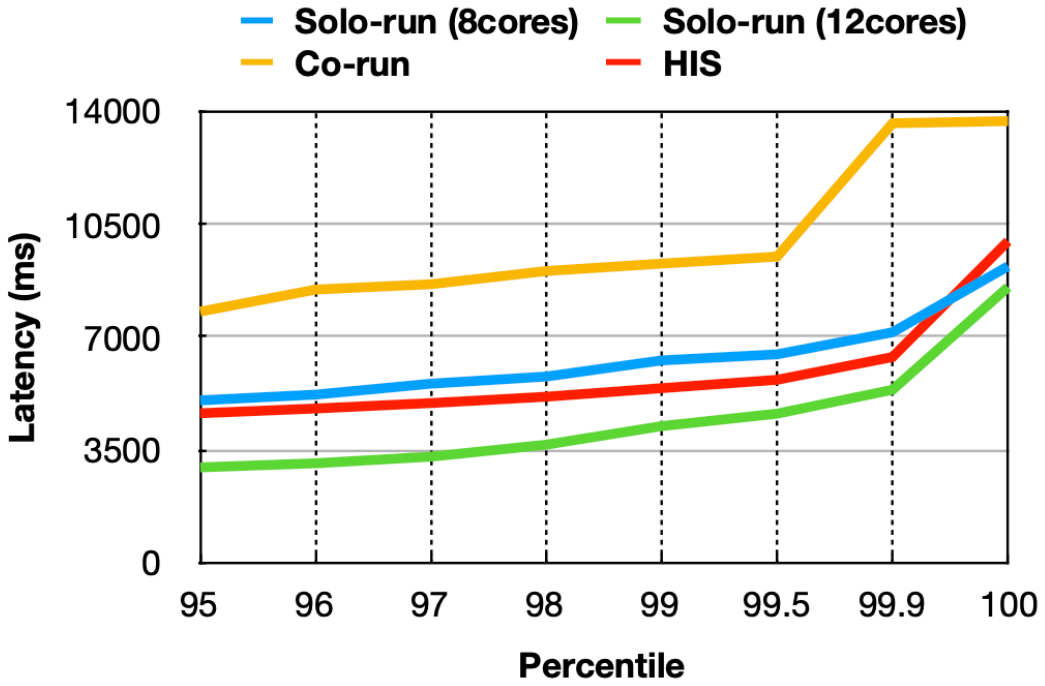
- **Machine**
  - Intel XeonE5-2683 v4 (16-core per socket, 2 sockets)
  - LLC: 40MB, RAM: 32GB, Memory BW: 68GB/s
  - Linux 4.19.0
- **Workloads**
  - Multi-threaded benchmarks (PARSEC, Rodinia, NPB, Apache web server with ab)
  - Co-locates two workloads (FG and BG) in a machine
  - SP is used for BG workload, because SP shows the highest memory bandwidth and LLC usages
- **Comparing with co-running workloads with static CPU isolation**
  - Batch workloads
  - Latency-critical workloads

Evaluation (Batch Workload)



- **FG: benchmarks shown in x-axis & BG: SP of NPB**
- Performance is normalized to the baseline (co-run with static CPU isolation)
- It achieves 1.05~1.7x performance than baseline and BG shows degraded performance up to 1.3x

Evaluation (Latency-critical Workload)



- **FG: Apache web benchmark (with ab) & BG: SP**
  - client (ab) sends requests generated with Pareto distribution
  - Apache web benchmark achieves 2.14x performance than co-run and also shows slightly lower latency than the solo-run (8-core) case
  - The maximum number of allocated CPU cores for web server reached to 12-cores responding to CPU demands

Future work and Conclusion

- **Future Work**
  - Testing with other isolation policies (e.g., selecting isolation techniques)
  - Testing with more diverse benchmarks (e.g., phase-changing workloads)
  - Comparing with recent works (e.g., Heracles-like system)
- **Conclusion**
  - We have analyzed the tradeoffs in HW/SW isolation techniques
  - We have implemented a prototype for hybrid isolation system
  - We have evaluated our prototype using selected benchmarks and achieved 1.7~2.14x performance improvement than static isolation

Thank you!

Questions?

Yoonsung Nam
DCSLab., Seoul National University
yoonsung.nam@snu.ac.kr