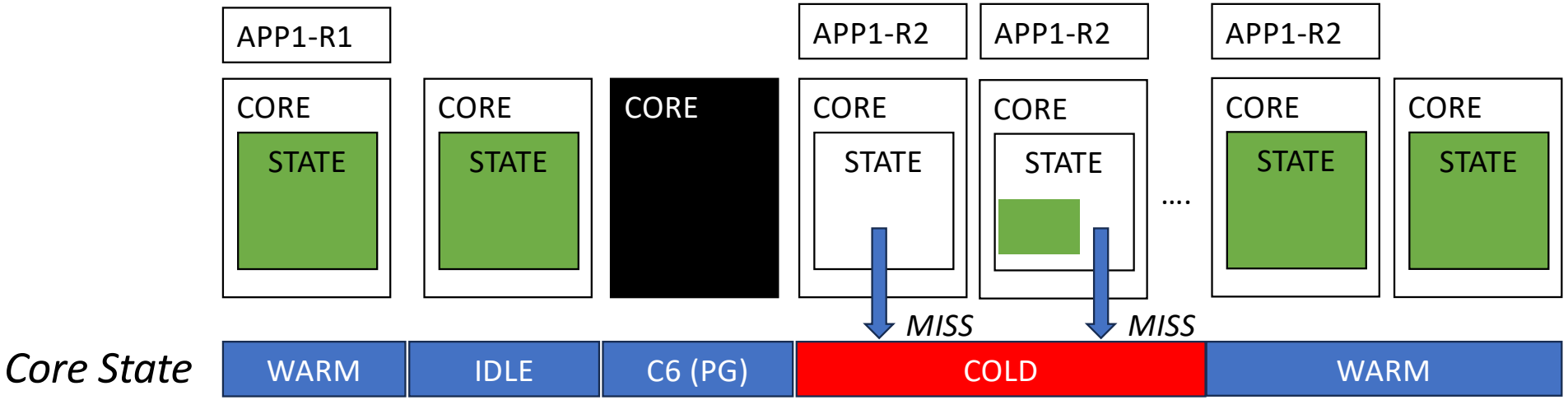
AgileWatts: C6Awarm

Motivation



- Power-gating causes state loss
- Once reactivated the core faces longer execution time due to cache misses and branch mispredicts
- The magnitude of this effect depends on the application (state reusability among requests)
- Idle governors do not consider this overhead before transitioning to deep sleep states

Why can Cold-Start Latency affect Microservices?

1. Microservices operate at low utilizations (5%-20% [Luo_2021])

2. Microservices have strict QoS constraints (< 1ms)
   - Sensitive to killer microsecond overheads
   - Previous works have estimated cold start to be in the order of µs [Arora_2015]

3. A query executes on multiple services running on different cores
   - Idle governors have local visibility (core)
   - A query might experience cold-start latency multiple times during its lifetime

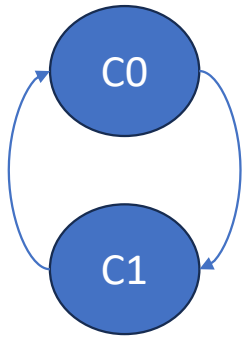
Goal

- Quantify the impact of cold-start latency on the performance of microservices

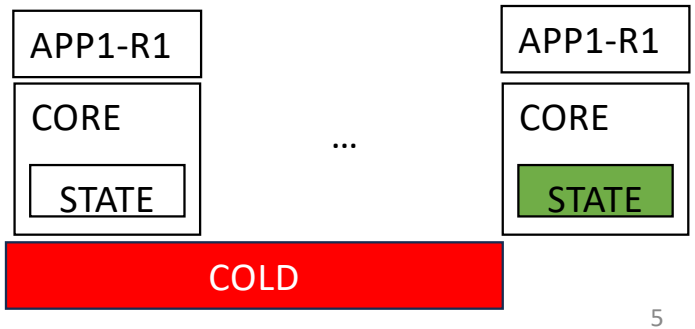
- Redesign the AW architecture to have the following characteristics:
  - Fast transition and cold-start latency – similar to shallow sleep states i.e., C1
  - Significant power savings – similar to deep sleep states i.e., C6

Sources of Performance Degr. for Deep Sleep States

- Transition Latency (C6A): Time required to transition from an active state to an idle state and vice versa
  - Examples of steps included: turn on/off VR/PLL, flush L1/L2 cache
  - Depends both on architecture choices and workload characteristics
  - Hardware Round Trip Time



- Cold-Start Latency (C6Awarm): Warm-up time of core data arrays (i.e., caches, branch predictor)
  - We focus on cold-start latency caused by power gating
  - Depends on workload and specifically reusability of state among queries



Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	**Caches**		
	Context		

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	Caches		
	Context		
Non-Architectural			

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	Caches		
	Context		
Non-Architectural			
PDN, Clock			

Interplay of Core C-states Overhead with Core Components

		Transition Latency	**Cold-Start Latency**
Architectural	**Caches**	DL1, L2, DTLB, STLB	IL1, Uop$, DL1, L2, ITLB, DTLB, STLB
	Context	Control Registers, Patch RAM, microcode	
Non-Architectural			Branch Prediction, BTB, RAS, IJB, Prefetcher
PDN, Clock		PLL, FIVR, CLOCK	

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	**Caches**	DL1, L2, DTLB, STLB	IL1, Uop$, DL1, L2, ITLB, DTLB, STLB
	Context	Control Registers, Patch RAM, microcode	
Non-Architectural			Branch Prediction, BTB, RAS, IJB, Prefetcher
PDN, Clock		PLL, FIVR, CLOCK	

- All transition latency resources are architectural

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	**Caches**	DL1, L2, DTLB, STLB	IL1, Uop$, DL1, L2, ITLB, DTLB, STLB
	Context	Control Registers, Patch RAM, microcode	
Non-Architectural			Branch Prediction, BTB, RAS, IJB, Prefetcher
PDN, Clock		PLL, FIVR, CLOCK	

- All transition latency resources are architectural
- All cold-start latency resources are non-architectural

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	**Caches**	DL1, L2, DTLB, STLB	IL1, Uop$, DL1, L2, ITLB, DTLB, STLB
	Context	Control Registers, Patch RAM, microcode	
Non-Architectural			Branch Prediction, BTB, RAS, IJB, Prefetcher
PDN, Clock		PLL, FIVR, CLOCK	

- All transition latency resources are architectural
- All cold-start latency resources are non-architectural
- There is a subset of resources that affect both categories

Interplay of Core C-states Overhead with Core Components

		Transition Latency	Cold-Start Latency
Architectural	**Caches**	DL1, L2, DTLB, STLB	IL1, Uop$, DL1, L2, ITLB, DTLB, STLB
	Context	Control Registers, Patch RAM, microcode	
Non-Architectural			Branch Prediction, BTB, RAS, IJB, Prefetcher
PDN, Clock		PLL, FIVR, CLOCK	

- All transition latency resources are architectural
- All cold-start latency resources are non-architectural
- There is a subset of resources that affect both categories
- The taxonomy reveals the resources:
  - The state of which must be preserved when power-gating (architectural)
  - The state of which must be preserved only if it affects significantly the performance (non-architectural)

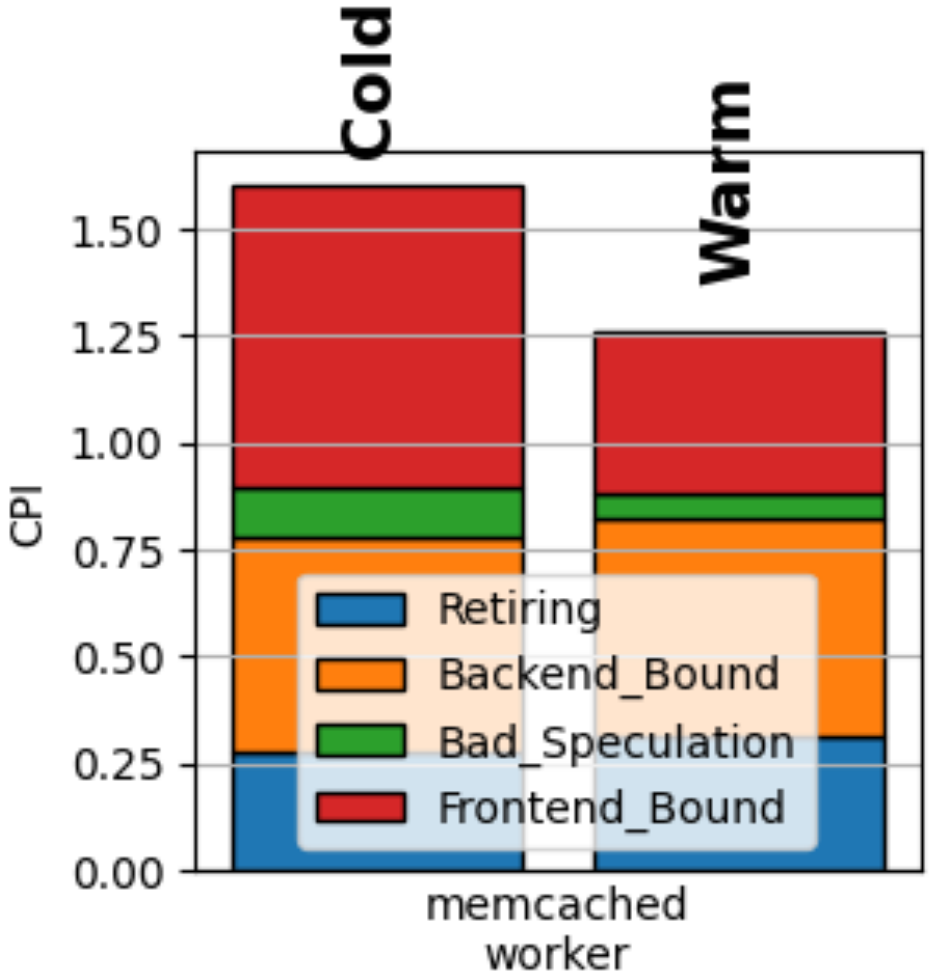
Experimental Methodology

- Workloads: We quantify cold-start latency using Memcached and MicroSuite

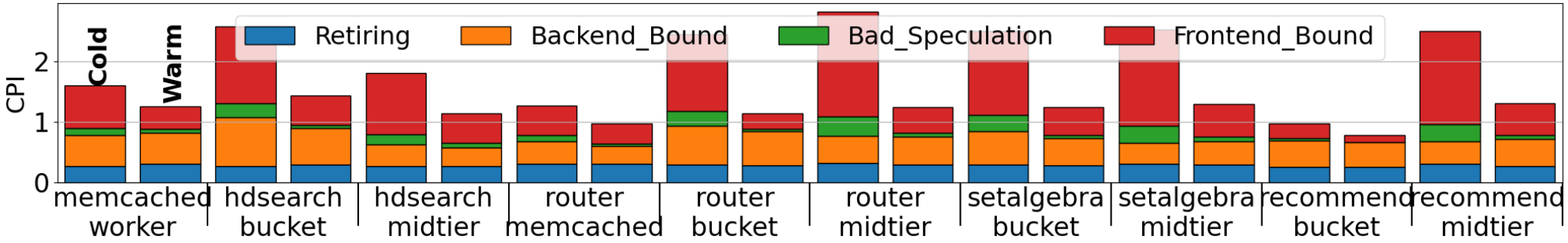
- Scenarios: We use two scenarios to quantify the worst-case cold-start latency
  - Warm: All queries execute on warm resources
  - Cold: All queries execute on cold resources(C6)

- Tools: We use performance counters and Top-Down Analysis

Cold-Start Latency Impact (1)



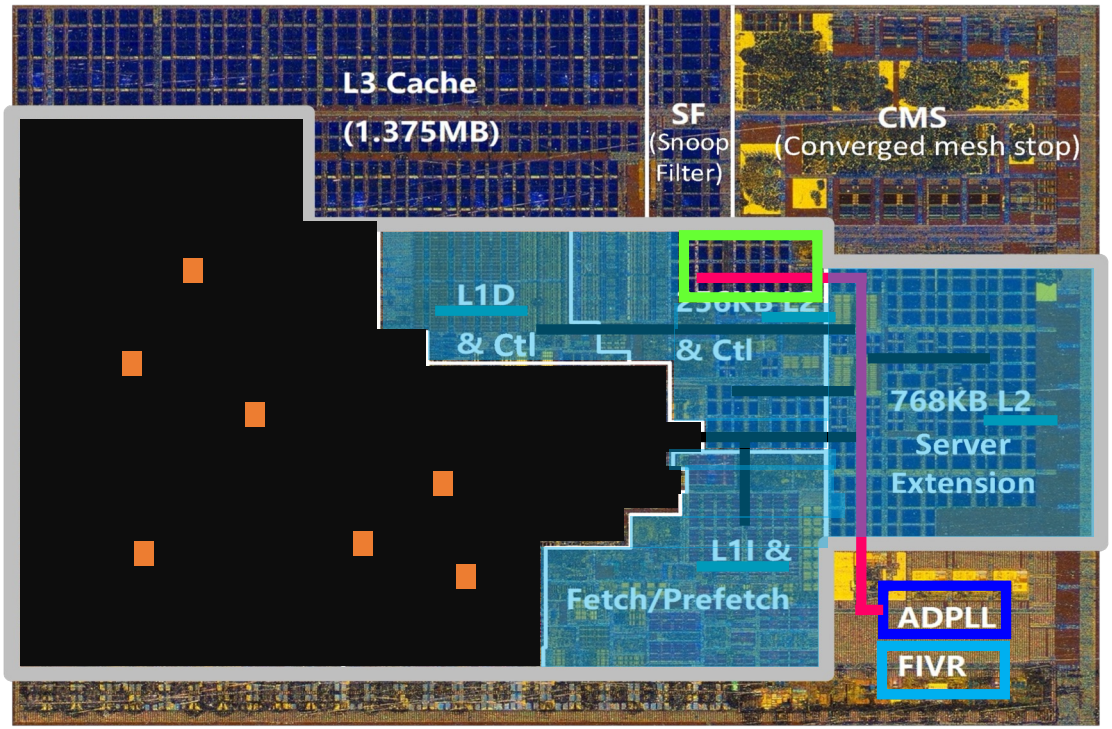
Cold-Start Latency Impact (2)



Top-Down Analysis of an application running on cold (power-gated) and warm resources (power-ungated)

- Cold-Start latency affects the performance of a microservice (25%-126%)
- Frontend Bound and Bad Speculation are the categories with the highest impact
- To eliminate this overhead, frontend resources must be ON

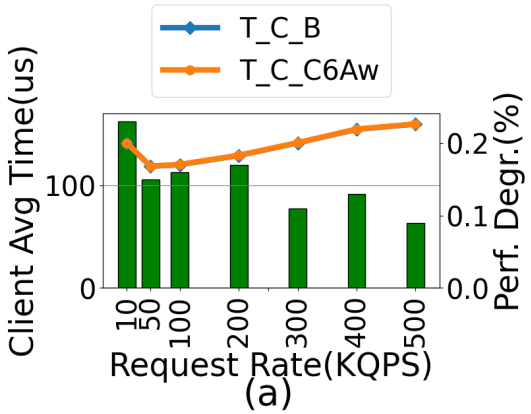
C6Awarm Architecture



C-State	Transition Time	Residency Time	Power per core
C0(P1)	N/A	N/A	~4W
C0(Pn)	N/A	N/A	~1W
C1(P1)	2µs	2µs	1.44W
C6Awarm(P1)	2µs	2µs	~0.35W
C1E(Pn)	10µs	20µs	0.88W
C6AwarmE(Pn)	10µs	20µs	~0.27W
C6	133µs	600µs	~0.1W

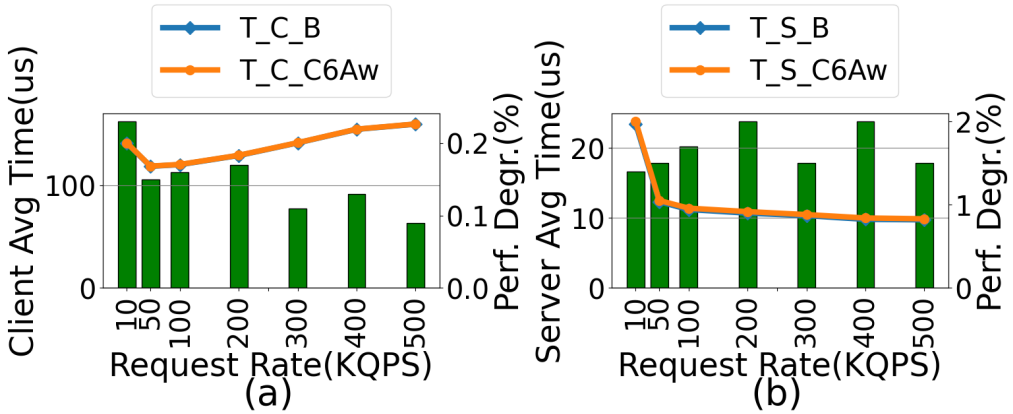
C-State	Clocks	ADPLL	L1/L2 Cache	Voltage	Context
C6Awarm	Most Stopped	On	Coherent	PG/Ret/Nom	Maintained

Power Savings and Overheads at Varying Loads



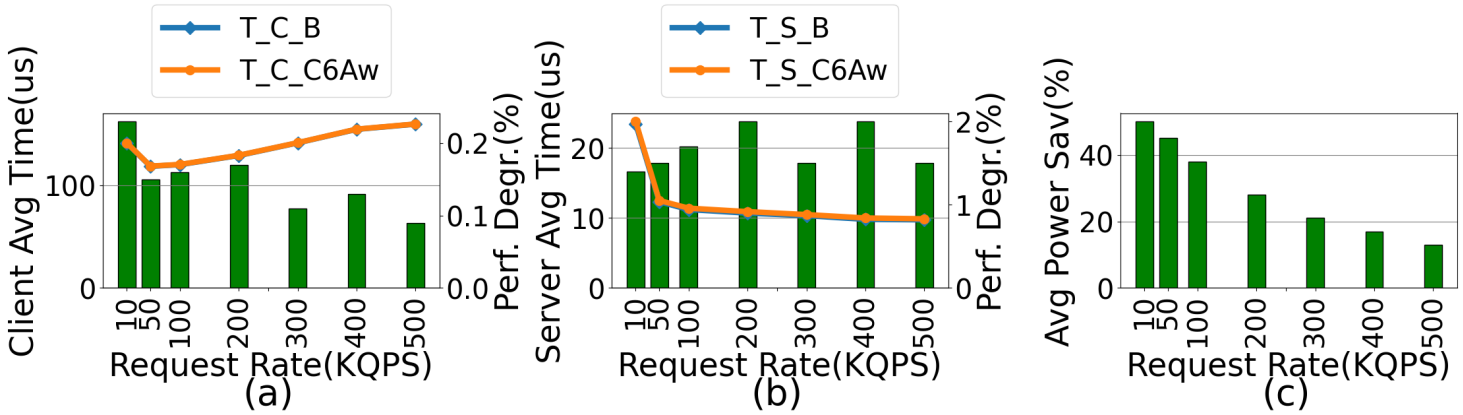
- C6Awarm/C6AwarmE at varying loads
  - Client-side performance degradation < 0.25%

Power Savings and Overheads at Varying Loads



- C6Awarm/C6AwarmE at varying loads
  - Client-side performance degradation < 0.25%
  - Server-side performance degradation < 2%
  - End-to-End dominated by network latency

Power Savings and Overheads at Varying Loads



- C6Awarm/C6AwarmE at varying loads
  - Client-side performance degradation < 0.25%.
  - Server-side performance degradation < 2%.
  - End-to-End dominated by network latency.
  - Power savings reach up to 50% for low qps and ~12% for high

Summary

- Cold-start latency significantly affects the performance of microservices (25%-126%)
- C6Awarm: first core C-state architecture that targets both the transition latency and cold effect latency
- C6Awarm employs the same techniques as AgileWatts to mitigate the transition latency of C-states. Additionally, C6Awarm keeps the frontend of the core ON and in retention mode to eliminate the cold effect latency and maintain the power savings of deep idle states
- Our evaluation shows that C6Awarm can reduce the energy consumption significantly with minimal performance overhead compared to baseline(only C1 C-state enabled)