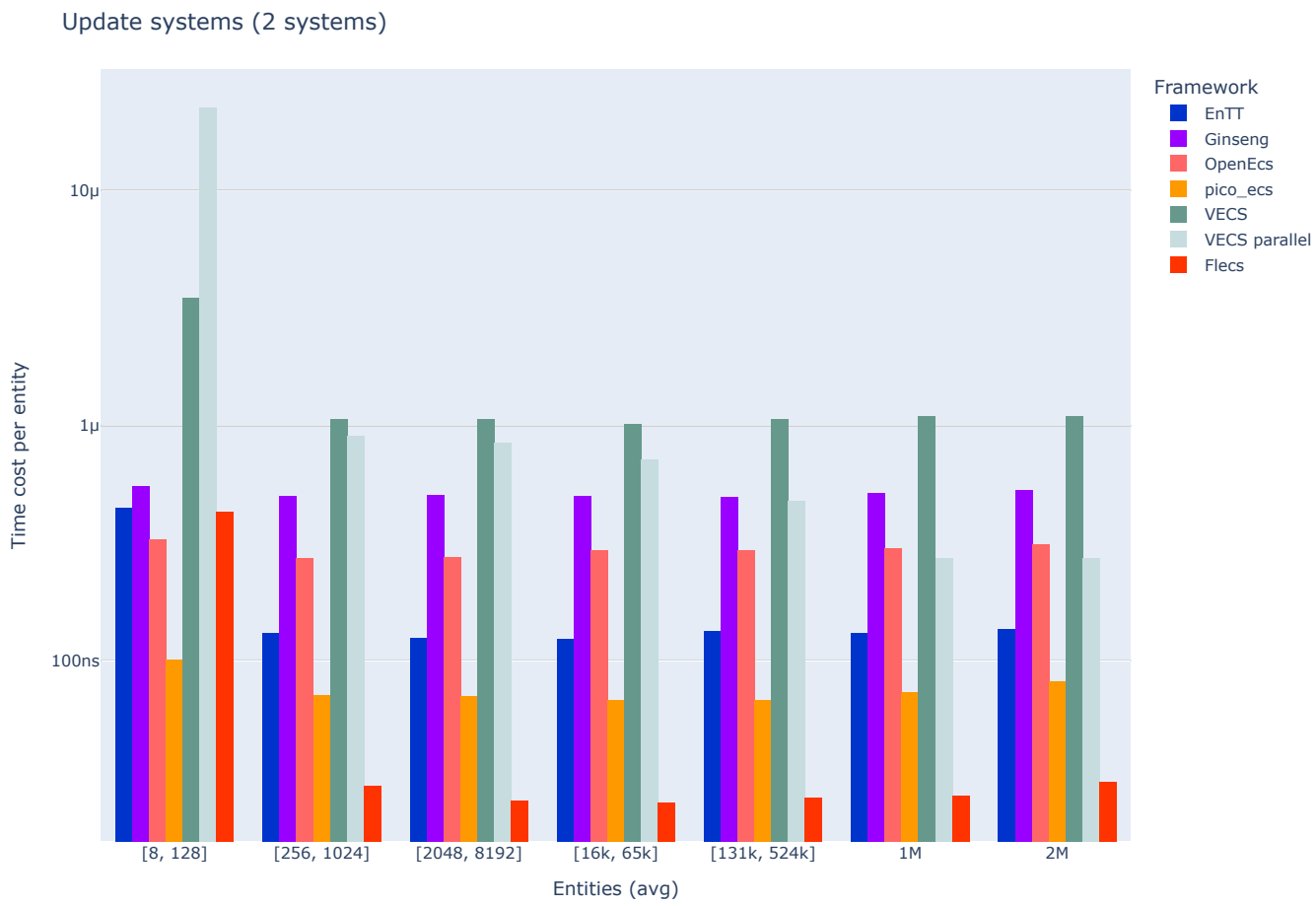
Results

TL;DR Results



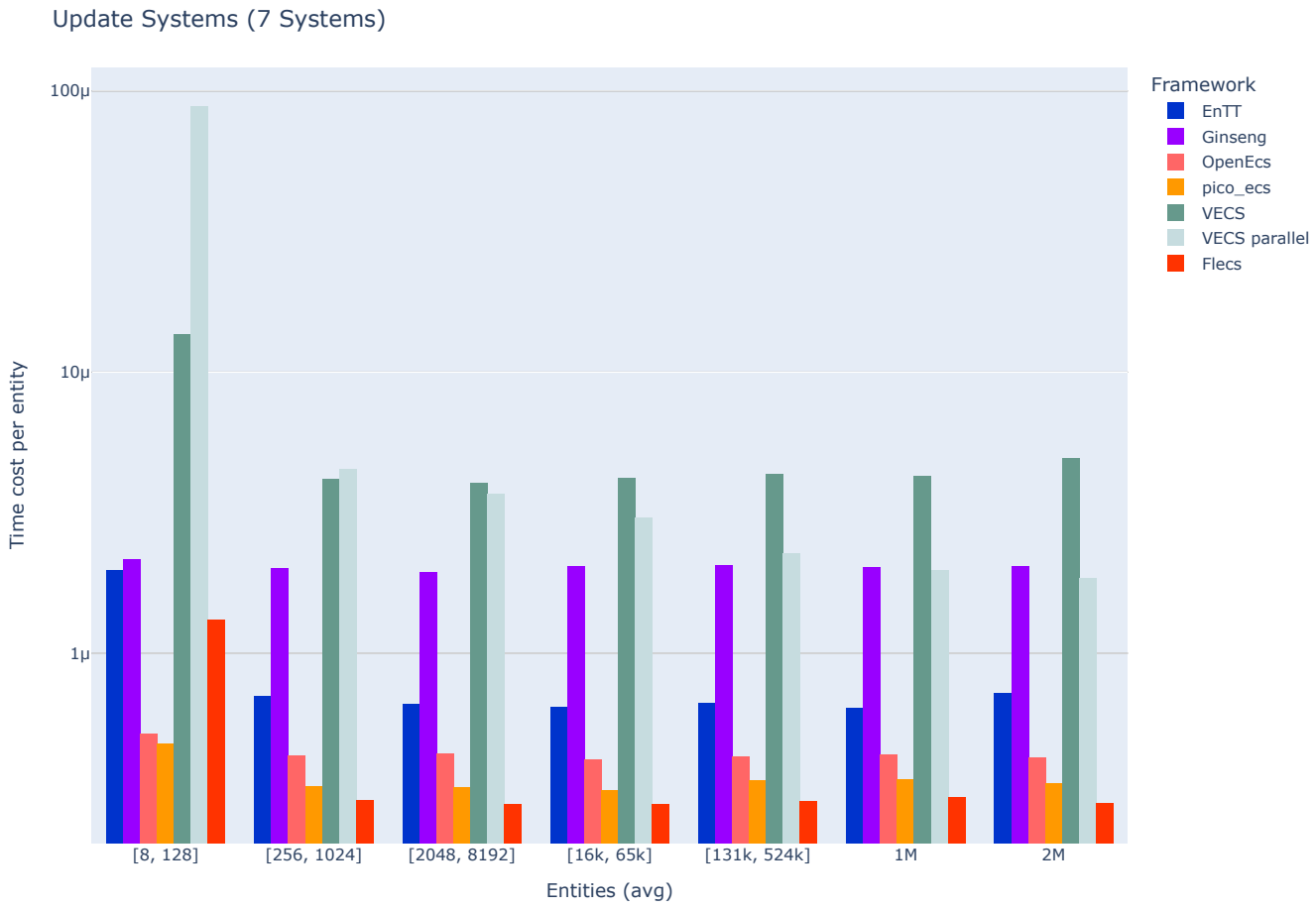
Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update 256 entities with 7 systems	188us	530us	109us	84us	1056us	1256us	78us
Update ~1K entities with 7 systems	697us	2016us	455us	355us	4359us	5726us	302us
Update ~4K entities with 7 systems	2667us	8032us	1769us	1408us	16322us	13629us	1192us
Update ~16K entities with 7 systems	10483us	33185us	6837us	5244us	70758us	56663us	4884us
	EnTT	**Ginseng**	**OpenEcs**	**pico_ecs**	**VECS**	**VECS parallel**	**Flecs**

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update ~65K entities with 7 systems	42ms	134ms	27ms	21ms	285ms	184ms	18ms
Update 262K entities with 7 systems	179ms	538ms	111ms	88ms	1211ms	594ms	81ms
Update ~1M entities with 7 systems	670ms	2130ms	457ms	372ms	4470ms	2075ms	322ms
Update ~2M entities with 7 systems	1516ms	4281ms	897ms	726ms	10341ms	3886ms	615ms

Benchmarks

Update systems (for-each entities in 7 systems)



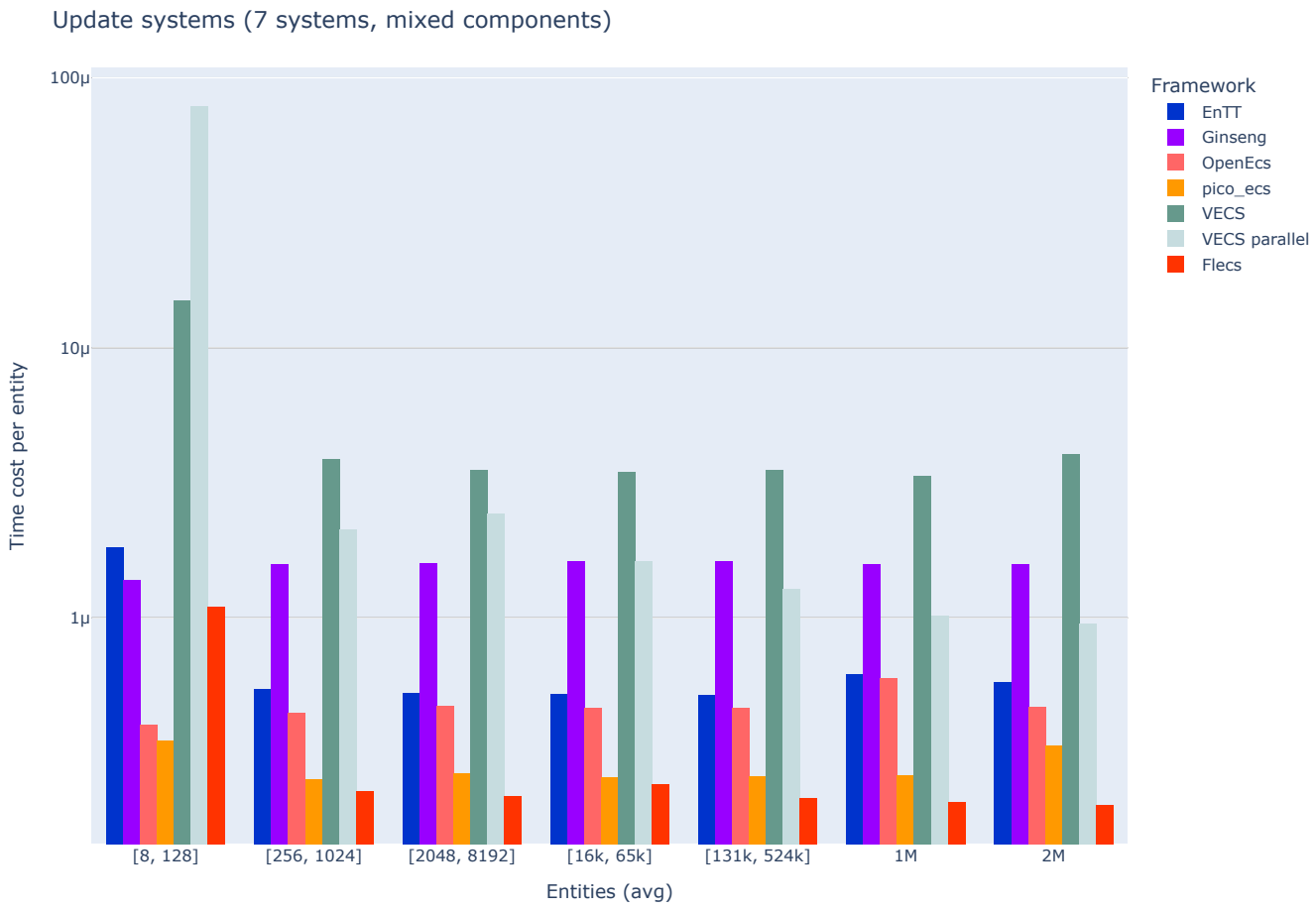
Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update 256 entities with 7 systems	188us	530us	109us	84us	1056us	1256us	78us

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update ~1K entities with 7 systems	697us	2016us	455us	355us	4359us	5726us	302us
Update ~4K entities with 7 systems	2667us	8032us	1769us	1408us	16322us	13629us	1192us
Update ~16K entities with 7 systems	10483us	33185us	6837us	5244us	70758us	56663us	4884us

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update ~65K entities with 7 systems	42ms	134ms	27ms	21ms	285ms	184ms	18ms
Update 262K entities with 7 systems	179ms	538ms	111ms	88ms	1211ms	594ms	81ms
Update ~1M entities with 7 systems	670ms	2130ms	457ms	372ms	4470ms	2075ms	322ms
Update ~2M entities with 7 systems	1516ms	4281ms	897ms	726ms	10341ms	3886ms	615ms

Update systems (for-each entities (with mixed components) in 7 systems)



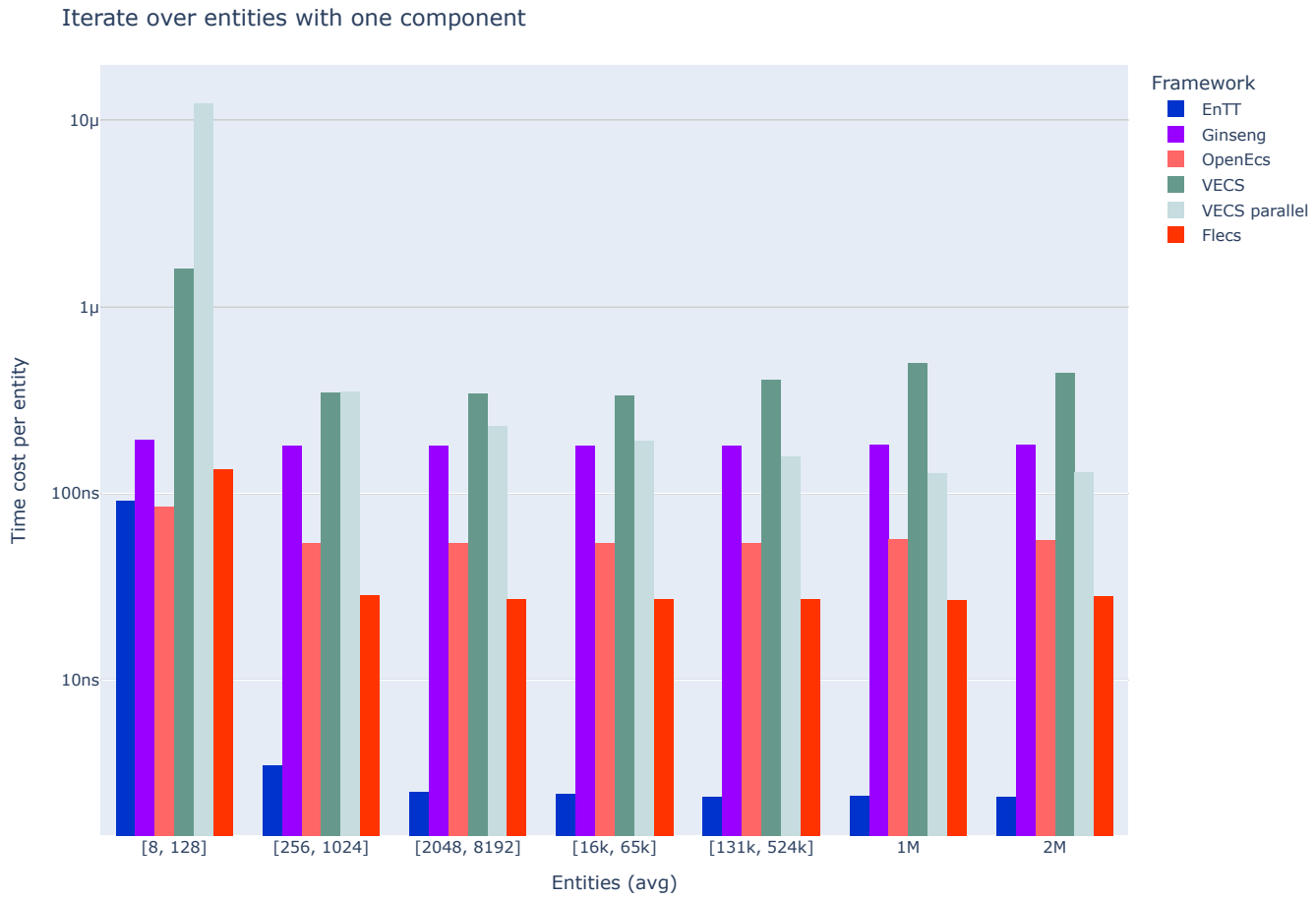
Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update 256 entities with 7 Systems	138us	407us	110us	64us	1041us	758us	59us
Update ~1K entities with 7 Systems	535us	1600us	465us	257us	3830us	1553us	226us
Update ~4K entities with 7 Systems	2170us	6534us	1922us	1052us	14412us	9566us	909us
Update ~16K entities with 7 Systems	8552us	26199us	7593us	4306us	58048us	29161us	4837us

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update ~65K entities with 7 Systems	34ms	108ms	30ms	16ms	215ms	97ms	13ms
Update 262K entities with 7 Systems	134ms	414ms	120ms	66ms	1099ms	362ms	57ms

	EnTT	Ginseng	OpenEcs	pico_ecs	VECS	VECS parallel	Flecs
Update ~1M entities with 7 Systems	647ms	1655ms	626ms	272ms	3498ms	1063ms	216ms
Update ~2M entities with 7 Systems	1214ms	3302ms	978ms	705ms	8484ms	1990ms	425ms

Iterate over entities with one component



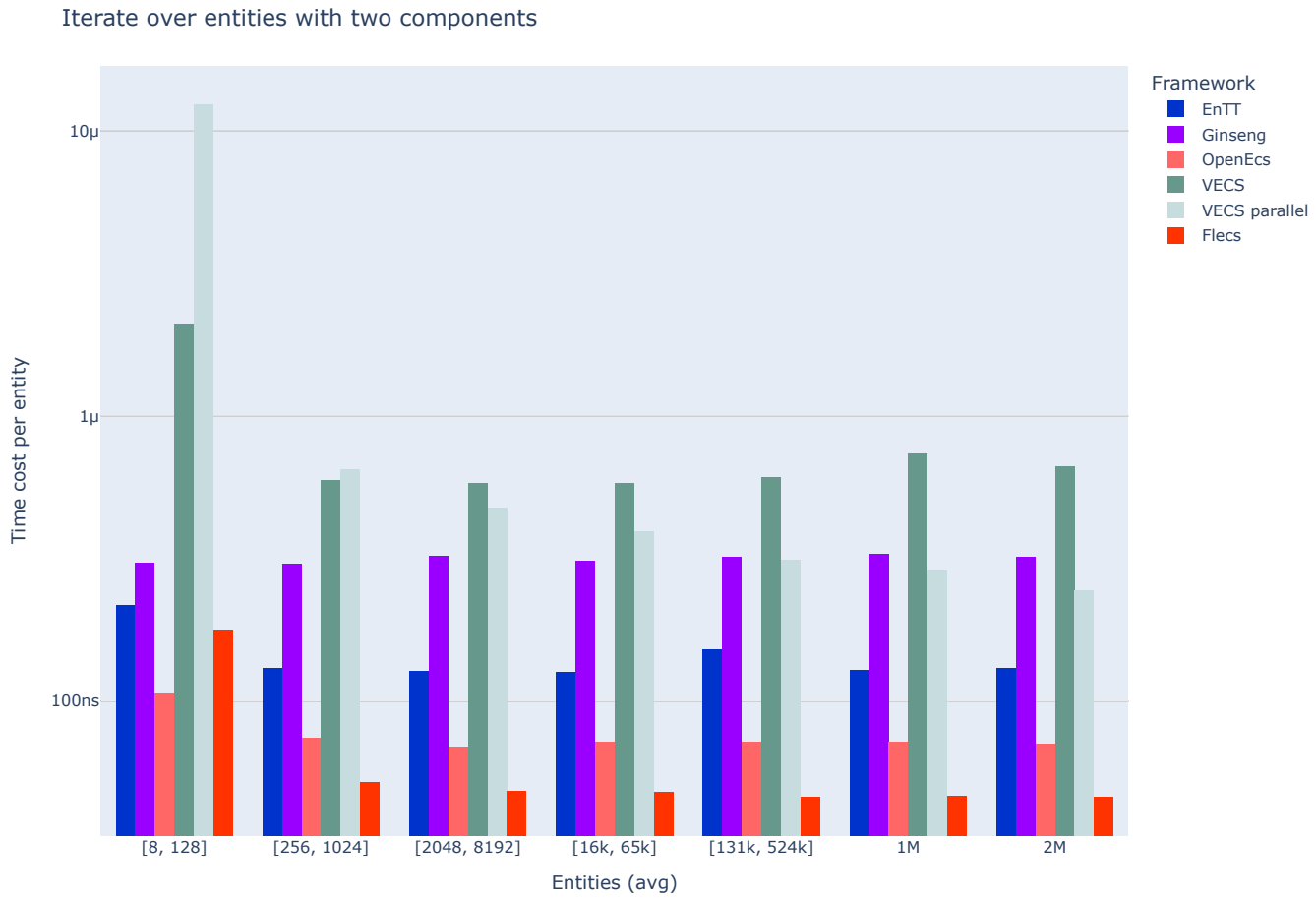
Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over 256 entities with one component	1us	46us	13us	91us	96us	7us
Iterate over ~1K entities with one component	2us	184us	54us	346us	472us	28us
Iterate over ~4K entities with one component	10us	738us	222us	1388us	892us	109us

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over ~16K entities with one component	40us	2951us	876us	5338us	3153us	437us

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over ~65K entities with one component	0ms	11ms	3ms	22ms	12ms	1ms
Iterate over 262K entities with one component	0ms	47ms	14ms	97ms	44ms	7ms
Iterate over ~1M entities with one component	2ms	190ms	59ms	521ms	134ms	28ms
Iterate over ~2M entities with one component	4ms	381ms	118ms	929ms	273ms	59ms

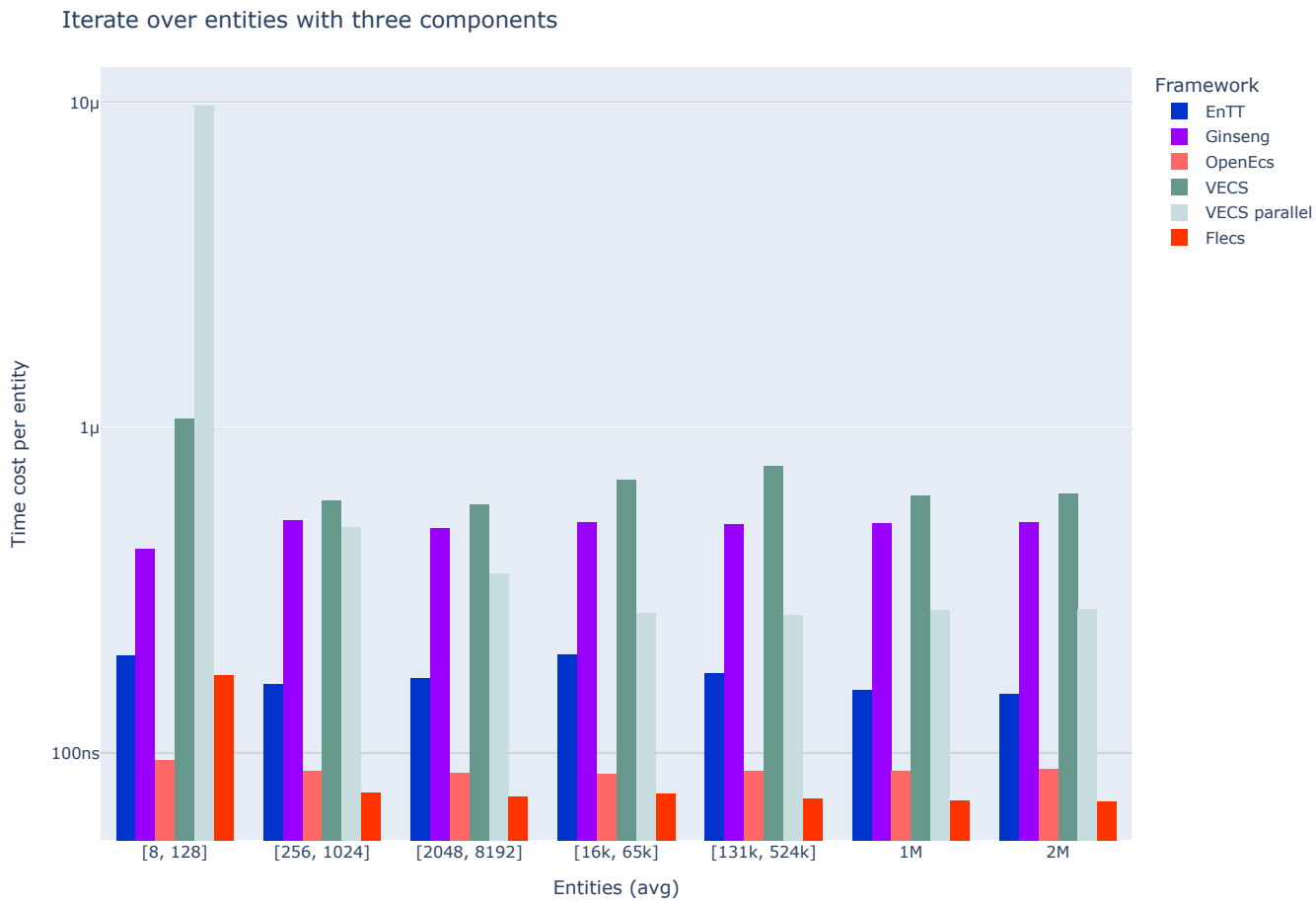
Iterate over entities with two components



Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over 256 entities with two components	32us	77us	18us	154us	177us	12us
Iterate over ~1K entities with two components	136us	311us	73us	599us	783us	58us
Iterate over ~4K entities with two components	525us	1299us	282us	2359us	2000us	194us
Iterate over ~16K entities with two components	2110us	5139us	1147us	9598us	6467us	778us
	EnTT	**Ginseng**	**OpenEcs**	**VECS**	**VECS parallel**	**Flecs**
Iterate over ~65K entities with two components	8ms	20ms	4ms	37ms	26ms	3ms
Iterate over 262K entities with two components	48ms	82ms	19ms	160ms	84ms	12ms
Iterate over ~1M entities with two components	134ms	344ms	75ms	776ms	300ms	48ms
Iterate over ~2M entities with two components	274ms	673ms	148ms	1394ms	513ms	96ms

Iterate over entities with three components



Graph shows cost per entity, tables shows total cost. lower is faster.

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over 256 entities with three components	41us	132us	22us	158us	134us	19us
Iterate over ~1K entities with three components	167us	566us	92us	615us	480us	75us
Iterate over ~4K entities with three components	667us	2015us	351us	2404us	1545us	301us
Iterate over ~16K entities with three components	3584us	8781us	1407us	9449us	4501us	1187us

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over ~65K entities with three components	12ms	32ms	5ms	45ms	17ms	5ms
Iterate over 262K entities with three components	46ms	133ms	23ms	202ms	68ms	19ms

	EnTT	Ginseng	OpenEcs	VECS	VECS parallel	Flecs
Iterate over ~1M entities with three components	163ms	532ms	92ms	648ms	289ms	75ms
Iterate over ~2M entities with three components	319ms	1071ms	187ms	1319ms	580ms	148ms

Candidates

EntityX by @alecthomas

> Entity Component Systems (ECS) are a form of decomposition that completely decouples entity logic and data from the entity "objects" themselves. The Evolve your Hierarchy article provides a solid overview of EC systems and why you should use them.

Version: 1.1.2 (Apr 2023)

EnTT by @skypjack

> EnTT is a header-only, tiny and easy to use library for game programming and much more written in modern C++.

Version: v3.13.2

Ginseng by @apples

> Ginseng is an entity-component-system (ECS) library designed for use in games.

The main advantage over similar libraries is that the component types do not need to be listed or registered. Component types are detected dynamically.

Any function-like object can be used as a system. The function's parameters are used to determine the required components.

Version: 1.1 (Dec 2021)

mustache by @kirillochnev

> A fast, modern C++ Entity Component System

Version: 0.2 (Feb 2024)

OpenEcs by @Gronis

> Open Ecs is an Entity Component System that uses metaprogramming, cache coherency, and other useful tricks to maximize performance and configurability. It is written in c++11 without further dependencies.

Version: 0.1.101 (Apr 2017)

Flecs by @SanderMertens

> Flecs is a fast and lightweight Entity Component System that lets you build games and simulations with millions of entities.

Version: v4.0.1

pico_ecs by @empyreanx

> A collection of cross-platform single header libraries written in C. Pure and simple ECS.

Version: 2.3 (Sep 2023)

gaia-ecs by @richardbiely

> Gaia-ECS is a fast and easy-to-use ECS framework.

Version: v0.8.6

VECS by @hlavacshelmut

> The Vienna Entity Component System (VECS) is a C++20 based ECS for game engines.

Version: 0.1

VECS parallel by @hlavacshelmut @hoelzlisabella

> The Vienna Entity Component System (VECS) is a C++20 based ECS for game engines.

Version: 0.1

Environment

- **OS:** Windows
- **CPU:** 2.11GHz @ 8Cores
- **RAM:** 15.78GB