# Every Cache A Painting, Revisited

Jody Donetti
*R&D + coding*

FusionCache
*Free + OSS*

# Jody Donetti

## R&D + coding

Doing stuff (mainly) on the web for about 30 years.

Dealt with most types of caches: memory, distributed, HTTP, CDN, offline.

🏆 Google Open Source Award

🏆 Microsoft MVP Award

# FusionCache

## Free + OSS

Easy to use, fast and robust hybrid cache with advanced resiliency features.

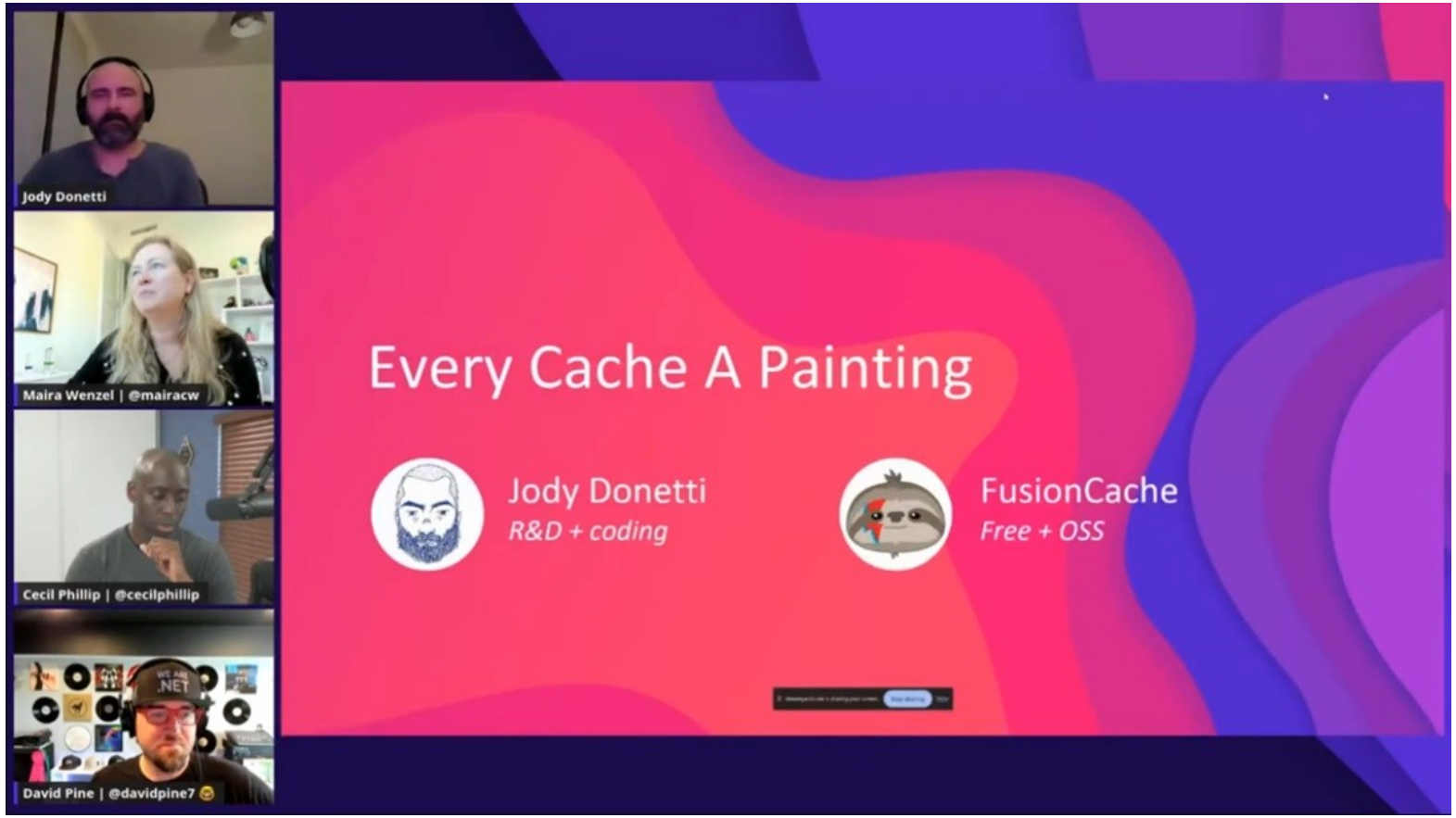
https://github.com/ZiggyCreatures/FusionCache

⬇️ 15M downloads

⭐ 2.8K stars

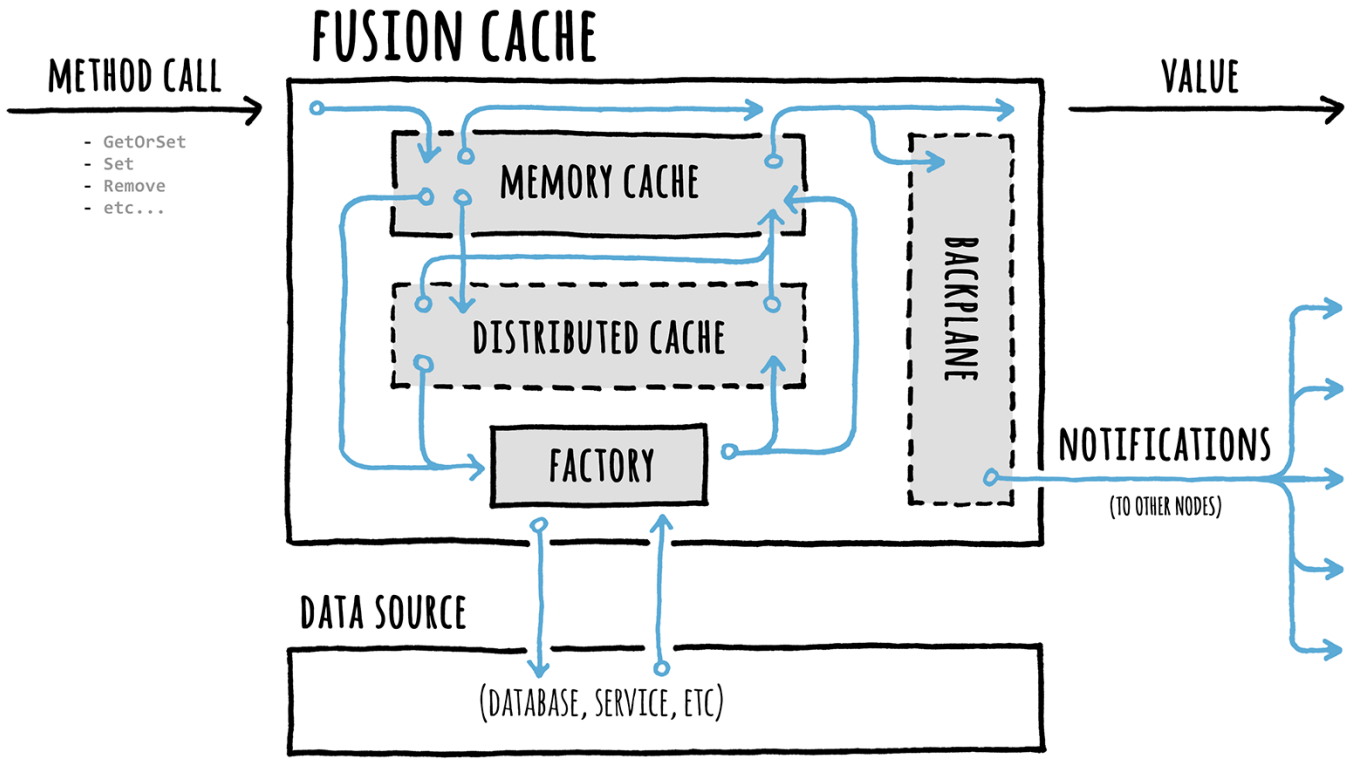
📜 MIT license

# Feb 2024: so, where were we?



# FusionCache

Easy to use, fast and robust hybrid cache with advanced resiliency features.



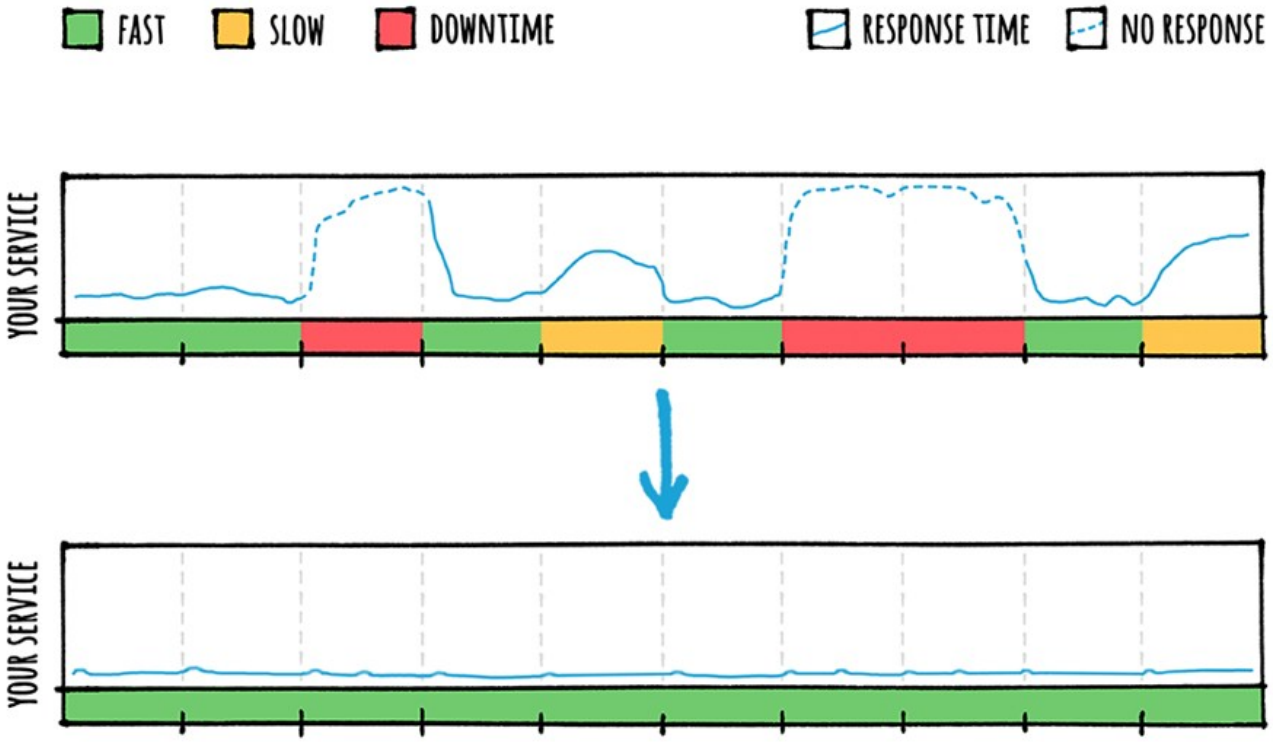
# FusionCache: back then

Based on `IMemoryCache` + `IDistributedCache`.

Features:

- Cache Stampede protection
- Fail-Safe
- Soft/Hard Timeouts
- Eager Refresh
- Conditional Refresh
- Adaptive Caching
- optional L2 + Backplane
- Auto-Recovery
- Named Caches
- dependency injection + builder
- sync/async
- options (global + granular with defaults)
- cancellation (via `CancellationToken`)
- Observability (`ILogger` + `OpenTelemetry`)
- Simulator
- rich docs (IntelliSense + online)
- .NET Standard 2.0 (old + new)
- MIT license

# 🦥 So, ideally




It helps us when 💩 hits the fan

# What happened in a year?

😬 A lot!

# 🥳 FusionCache v1.0

As promised, a couple of weeks after the episode (Feb 29th 2024) FusionCache finally went v1.0!

## v1.0.0

### FusionCache is now v1.0 🥳

Yes, it finally happened.

Let's see what this release includes.

#### 🚀 Performance, performance everywhere

FusionCache always tried to be as optimized as possible, but sometimes useful new features took some precedence over micro-optimizing this or that.

Now that all the major features (and then some) are there, it was time to do a deep dive and optimize a cpu cycle here, remove an allocation there and tweak some hot path to achieve the best use of resources.

So here's a non-comprehensive list of nice performance improvements in this release:

- zero allocations/minimal cpu usage in Get happy path
- reduced allocations/cpu usage in Set happy path
- less allocations/cpu usage when not using distributed components

# 📦 FusionCache v1.1, v1.2 ...

Last year I released multiple versions with new features, performance improvements and more.

A brief recap:

- 🔑 Keyed Services
- ♊ Auto-Clone
- 💣 soft Fail-Safe (no exceptions)
- 🧙 more flexible Adaptive Caching
- 🦅 more robust Eager Refresh
- ↩️ better Auto-Recovery
- 🚀 perf improvements overall
- ⚡ improved serialization
- 🔭 better OpenTelemetry
- 🔓 extensible Memory Locker
- 🐞 some bug fixes
- 📦 multi-targeting
- ✅ more tests (~1400)
- 📕 better docs

Oh, and about the docs...

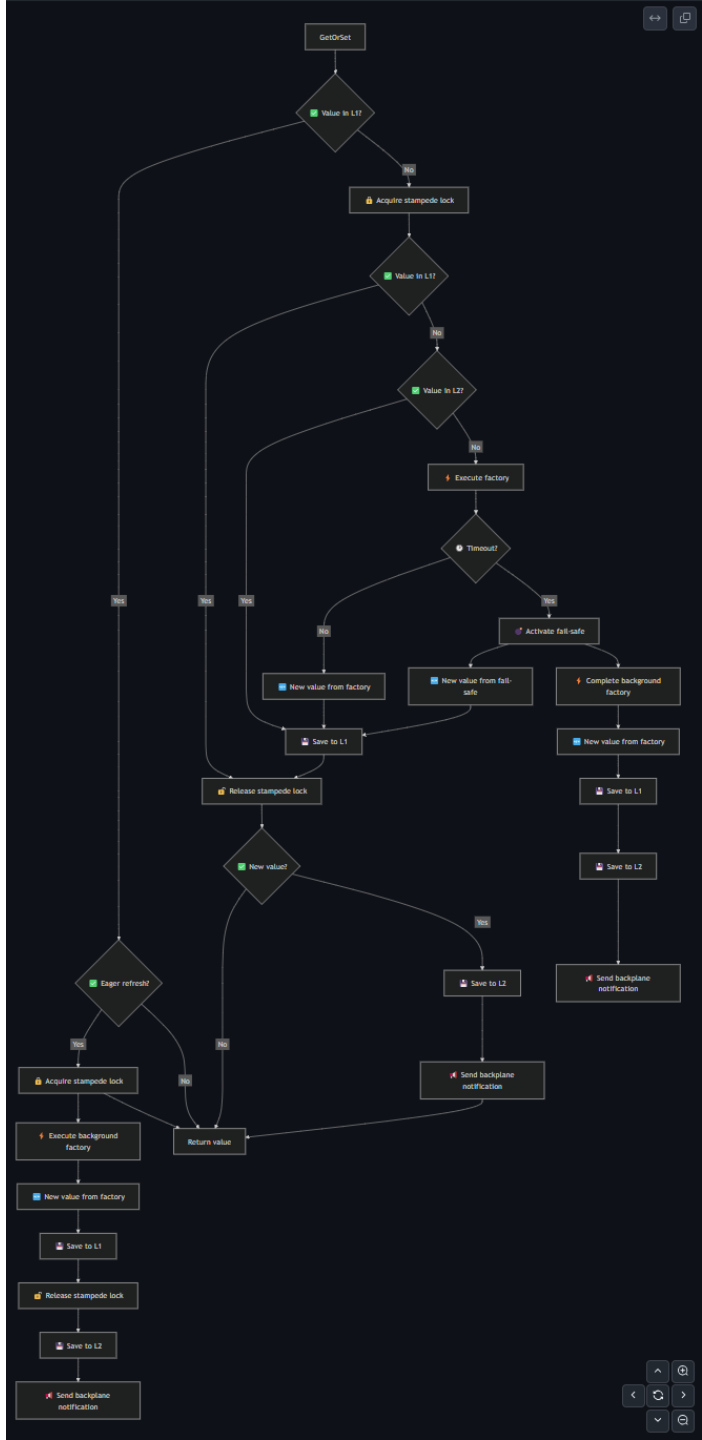
# Docs

Docs have been expanded and made overall better.

Meaning:

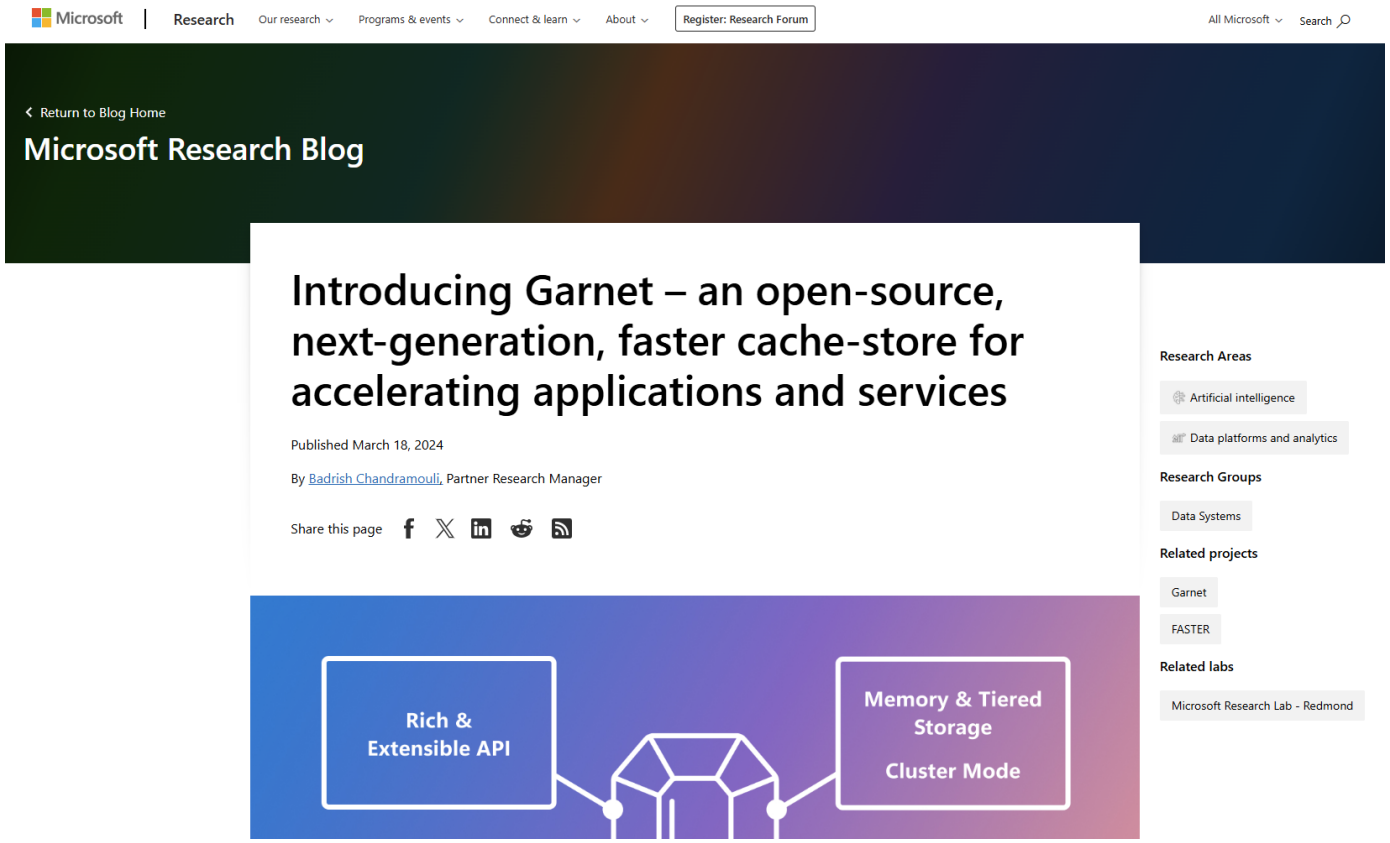
- more concepts covered
- more illustrations
- more practical examples
- feature design overviews

Also, I added diagrams!



# Microsoft Garnet

In March 2024 Microsoft launched Garnet: a super fast, Redis-compatible cache store.



# Microsoft Garnet: Redis compatible?

The existing `IDistributedCache` implementation for Redis required LUA, which was a no-go with Garnet.

So I opened an issues on the repo.

## Plans for an `IDistributedCache` implementation? #85

Closed

**jodydonetti** opened on Mar 22, 2024

Issue body actions

Hi all, are you planning on creating an `IDistributedCache` implementation anytime soon?

Since Garnet has been announced as being protocol-compatible with Redis I thought it would've worked right away by using the standard Redis implementation (see here), but after actually playing with it I discovered that Garnet does not currently support LUA scripting:

> 3. Garnet does not support Lua scripting. We have an experimental version, but it was noted to be too slow for realistic use so we have not added it to the project.

LUA scripting is what has been used in the aforementioned impl (see here) and therefore without it it's not possible to use Garnet this way.

Having a working impl of `IDistributedCache` for Garnet would drive the adoption a lot, since that is the standard way of using a distributed cache in .NET.

Also, that would unlock other great possibilities like being able to use Garnet in projects that use `IDistributedCache` as a "*lingua franca*", like FusionCache (mine) which is compatible with any impl of such interface (here for more).

All in all I think that, if the effort for such impl is not huge, that is something that you should consider to skyrocket the usage.

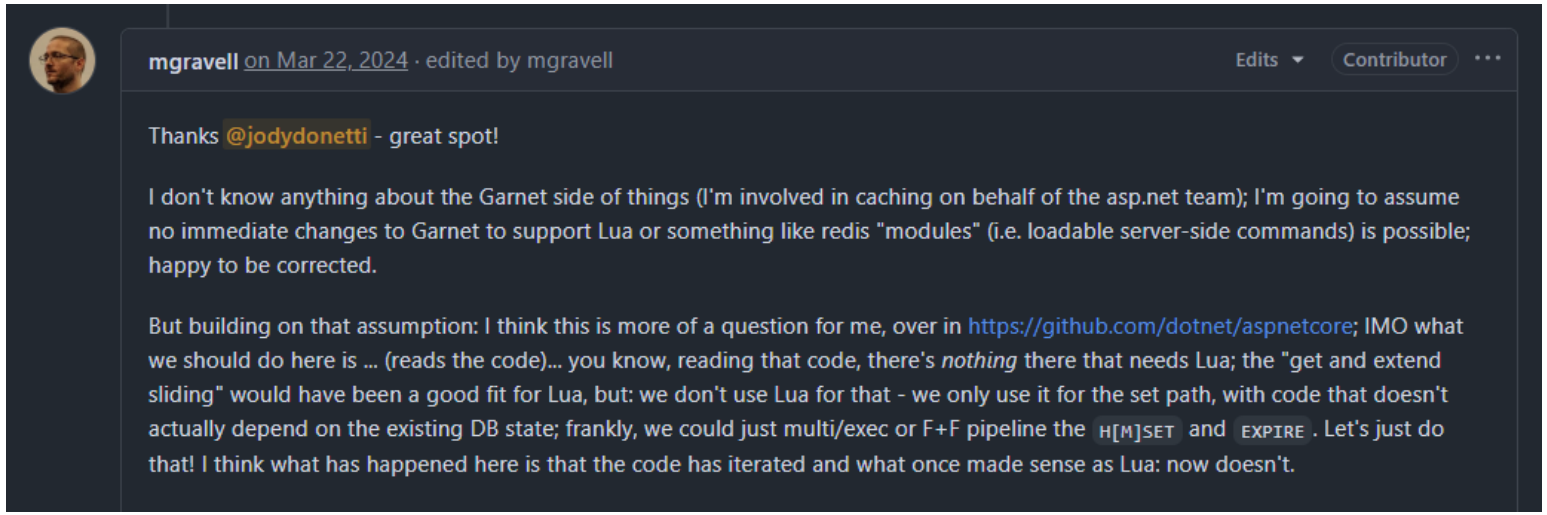
Thoughts?

ps: maybe **@mgravell** can give us some hints about such approach? Are there any hidden pitfalls that should be looked at?

👍 19 🚀 2

# Microsoft Garnet: it’s a win

Long story short, with .NET 9 not a problem anymore.



**mgravell** on Mar 22, 2024 · edited by mgravell  Edits  Contributor

Thanks @jodydonetti - great spot!

I don't know anything about the Garnet side of things (I'm involved in caching on behalf of the asp.net team); I'm going to assume no immediate changes to Garnet to support Lua or something like redis "modules" (i.e. loadable server-side commands) is possible; happy to be corrected.

But building on that assumption: I think this is more of a question for me, over in https://github.com/dotnet/aspnetcore; IMO what we should do here is ... (reads the code)... you know, reading that code, there's *nothing* there that needs Lua; the "get and extend sliding" would have been a good fit for Lua, but: we don't use Lua for that - we only use it for the set path, with code that doesn't actually depend on the existing DB state; frankly, we could just multi/exec or F+F pipeline the `H[M]SET` and `EXPIRE`. Let's just do that! I think what has happened here is that the code has iterated and what once made sense as Lua: now doesn't.

**NOTE:** thanks Marc!

# 👷 Data API Builder

Microsoft created **DAB** (Data API Builder).

It’s a tool to easily create modern web APIs:

- REST + GraphQL
- OpenAPI
- native CLI
- any database, any platform (cloud + on premise)
- Docker friendly
- ... caching!

Did you say caching? Go on 😌

# 👷 DAB + OSS

Data API Builder ❤️ OSS:

- FusionCache: caching engine
- HotChocolate: GraphQL engine

On **May 2024** Data API Builder became Generally Available.

https://devblogs.microsoft.com/azure-sql/data-api-builder-ga/

> What does it mean to be open source? That we work and think in the open. The code is right there; we collaborate with projects like HotChocolate and FusionCache. We have a monthly community standup hosted on YouTube, our issues and discussions are managed in GitHub, and – most of all – we accept your pull . https://aka.ms/dab

Thanks for the mention!

🚀 Ok, next big things after v1?

# Tagging

Since the early days of FusionCache people started asking me for a way to «group together cache entries» in a way or another.

The need is to invalidate multiple cache entries all at once because they have a «relation» or a «dependency» on the same underlying data.

So, during 2024, I finally decided time had come to finally deal with a monumental beast: Tagging.

See [issues/319]



[FEATURE] Tagging #319

Closed

jodydonetti opened on Oct 20, 2024 · edited by jodydonetti

Edits

## The Need

Time and time again, there have been requests from the community to support some form of *"grouping together cache entries"*, primarily for multi-invalidation purposes.

Here are some examples:

- Cache Regions? (2021)
- .Clear mechanism (2022)
- How can I manually expire the cache in the following scenarios? (2023)
- Are there any developments regarding integration with HybridCache and tags support? (2024)
- Group cache invalidation (cache regions) (2024)

On top of this, the upcoming HybridCache from Microsoft that will tentatively be released at around the .net 9 timeframe and for which FusionCache want to be an available implementation of (and actually the first one!), will seemingly support tagging.

So, it seems the time has come to finally deal with this monumental beast.

*Shadow of the Colossus*
*Copyright © Sony, All Rights Reserved*

# 🏷️ Tagging: it’s complicated

We need to basically “tag entries” and then “remove by tag”.

Now, cache invalidation is already incredibly complex in and on itself, even just ”only” on L1.

Then throw into the mix:

- L1 + L2 + Backplane: oh my
- L2 + Backplane means distributed (see Fallacies Of Distributed Computing)
- resiliency (Fail-Safe, Soft Timeouts, Auto-Recovery)
- multiple Named Caches (so, maybe shared L1/L2 caches)
- inherent limitations of underlying abstractions (eg: `IDistributedCache`)
- potentially devastating perf hit

# 🏷️ Tagging: Server VS Client

There are basically 2 potential approaches: **Server-Assisted** and **Client-Assisted**.

Server-Assisted, where “remove by tag” is done on the L2:

- PRO: it’s easy to implement, in theory (eg: `DELETE * FROM table WHERE "my-tag" IN tags`)
- CON: potentially massive perf hit
- CON: basically no L2 support it... so it doesn’t exist 😅

Client-Assisted, where the magic happens on the client:

- CON: super hard to implement well
- PRO: L2 does not need to support it
- PRO: if done well, perf may be... interesting (see later)

# 🏷️ Tagging: Client-Assisted

FusionCache stores tags with each cache entry.

Each “remove by tag” operation creates an internal, special cache entry with the timestamp.

At every “get” operation there’s a check for the related tags + automatic (maybe) cleanup.

Basically, it’s like a “high-pass filter” in electronics engineering.

Wait, is this the dreaded `SELECT N+1` problem?
Nope, because inherent caching nature + behaviour + probability theory (see “Birthday Paradox”).

# Tagging: inspiration

What do these things have in common?

- the Seoul jazz scene
- Chuck Mangione
- a sip of whisky
- an old, late-night doom scrolling session on Wikipedia
- ancient Egypt and Suffolk, England

Crinkle Crankle walls!

Their peculiar **DESIGN** and the way the bricks are **DISTRIBUTED** allow a Crinkle Crankle wall with a single line of bricks to be as **ROBUST**, if not **MORE**, than **NORMAL** straight walls composed of more lines of bricks, lowering the **COST**: even though they are typically 22% **LONGER** (if straighten) they use **LESS RESOURCES**.



## Of Ancient Egypt and Suffolk, England

Look at this beauty:


*A crinkle crankle wall in Bramfield, Suffolk*
*Nat Bocking / Crinkle-Crankle Wall in Bramfield / CC BY-SA 2.0*

# 🏷️ Tagging: what about?

Ok, what about:

- app restarts? automatic L1/L2 coordination
- data for "tag1" needed by 2 cache entries at the same time? Cache Stampede protection
- multiple nodes? Backplane + specific optimizations
- dynamic tags based on cached value? Adaptive Caching
- zombie entries? Inherent caching nature
- potential transient errors? Fail-Safe and Auto-Recovery
- slow distributed operations? advanced Timeouts and Background Distributed Operations

All thanks to the solid foundations built into FusionCache for years.

# Tagging: show me the code

Easy peasy:

```
// NORMAL
cache.Set<int>("foo", 1, tags: ["tag-1", "tag-2"]);
cache.GetOrSet<int>("bar", _ => 2, tags: ["tag-1", "tag-3"]);

// ADAPTIVE
cache.GetOrSet<int>(
    "baz",
    (ctx, _) =>
    {
        // SET TAGS HERE DYNAMICALLY
        ctx.Tags = ["tag-1", "tag-3"];
        return 3;
    }
);

// LATER
cache.RemoveByTag("tag-1");
```

# Clear

Based on Tagging + 2 special tags, to support both “remove all” and “expire all” (see Fail-Safe).

It all just works.

```
cache.Set("foo", 1);
cache.Set("bar", 2, tags: ["tag-1", "tag-2"]);
cache.Set("baz", 3);

// CLEAR
cache.Clear();

// HERE maybeFoo.HasValue IS false
var maybeFoo = cache.TryGet<int>("foo");
```

# 📅 Wait, back to early 2024

# Then, this happened

A surprise at https://github.com/dotnet/aspnetcore/issues/53255

## Epic: IDistributedCache updates in .NET 9 #53255

New issue

Open | Feature | Listed in #53178

**adityamandaleeka** opened on Jan 9, 2024 · edited by mgravell

Edits

Update:

`HybridCache` has relocated to dotnet/extensions:dev; it *does not* ship in .NET 9 RC1, as a few missing and necessary features are still in development; however, we expect to ship either alongside or very-shortly-after .NET 9! ("extensions" has a different release train, that allows additional changes beyond the limit usually reserved for in-box packages; `HybridCache` has always been described as out-of-box - i.e. a NuGet package - so: there is no reason for us to limit ourselves by the runtime restrictions)

---

Status: feedback eagerly sought

- API proposal part 1, core API: Hybrid Cache API proposal #54647 and Add HybridCache public API #55084
- Initial cut of proposal part 1: Implement minimal implementation of HybridCache #55147 and fixes: HybridCache - minor post-PR fixes #55251
- API proposal part 2, tags and invalidation: HybridCache - tags and invalidation #55308
- `By` nomenclature: HybridCache - rename RemoveKeyAsync and RemoveTagAsync to "By" #55332

Assignees

mgravell

Labels

Epic | area-networking

Type

Feature

Projects

No projects

Milestone

9.0.0

No due date

# HybridCache

A new *cache thing* from Microsoft? Let's see:

- combine `IMemoryCache` + `IDistributedCache`
- a unified serialization interface, to work with `byte[]`
- entry options + global defaults, `DefaultEntryOptions`

FusionCache design validated!

# 🥊 Let’s fight?

So, Microsoft is “killing OSS”?

What now for FusionCache?

Declare defeat?

A full-on fight?

🤝 Naaah, let’s collaborate!

# HybridCache: mega-comments

Sharing my experiences & my ideas with the team.

(about that: sorry again Marc 😅 )

**jodydonetti** on Feb 14, 2024 — Contributor

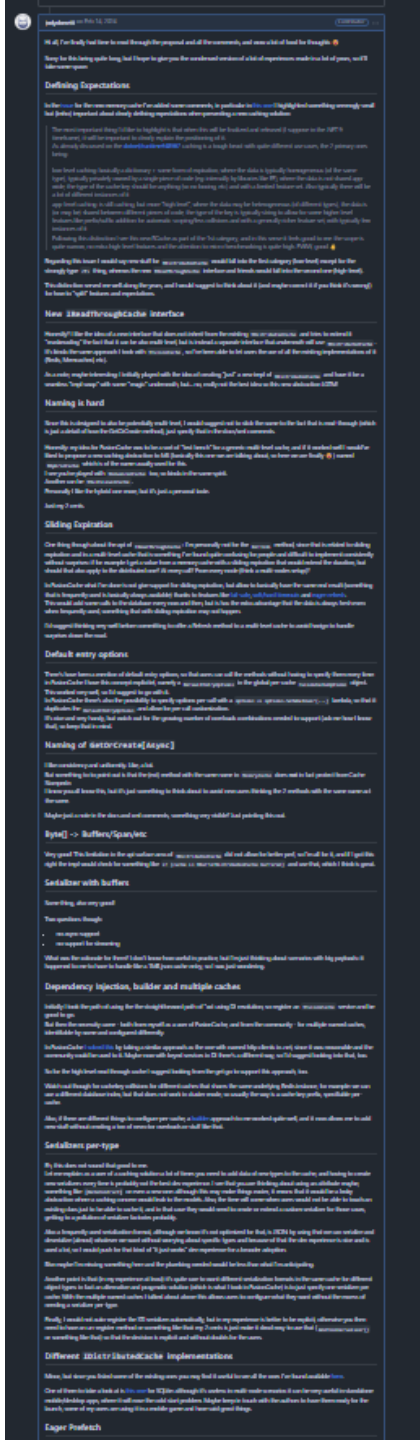
Hi all, I've finally had time to read through the proposal and all the comments, and wow a lot of food for thoughts 😅

Sorry for this being quite long, but I hope to give you the condensed version of a lot of experiences made in a lot of years, so it'll take some space.

## Defining Expectations

In the issue for the new memory cache I've added some comments, in particular in this one I highlighted something seemngly small but (imho) important about clearly deifning expectations when presenting a new caching solution:

> The most important thing I'd like to highlight is that when this will be finalized and released (I suppose in the .NET 9 timeframe), it will be important to clearly explain the positioning of it.
> As already discussed on the dotnet/runtime#48567 caching is a tough beast with quite different use cases, the 2 primary ones being:



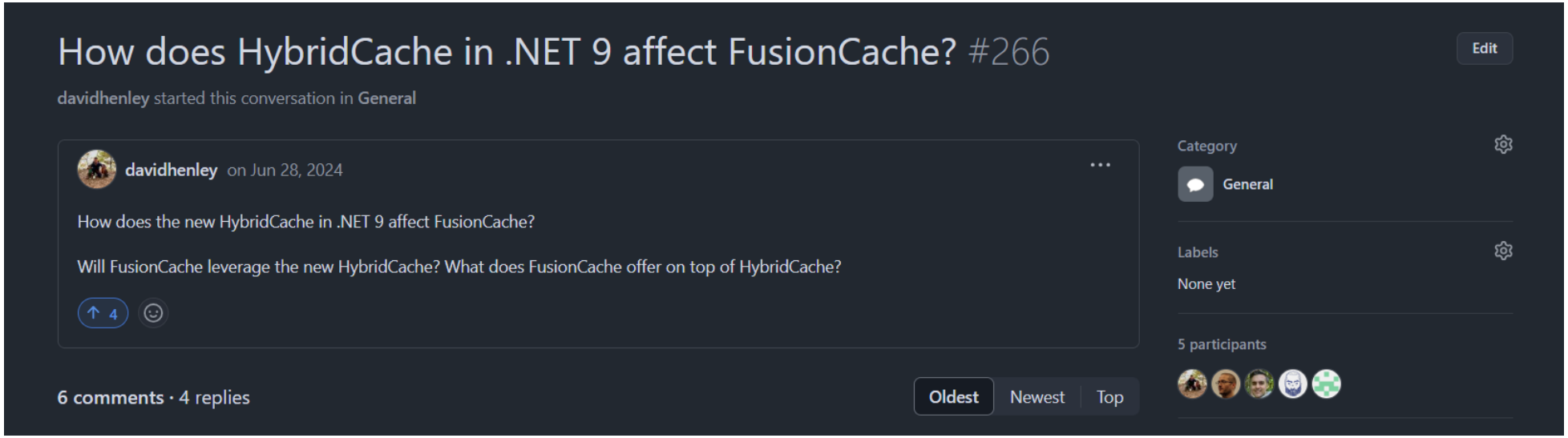
# HybridCache: it’s an abstraction

HybridCache it’s not *just* a Microsoft-provided **implementation**: it’s first and foremost an **abstraction**.

Think of it like `IDistributedCache` but for multi-level, hybrid caches.

And FusionCache *is* a multi-level, hybrid cache... see where I’m going?

And yes, of course people asked me about it.



# HybridCache: the plan

FusionCache remains its own, independent thing.

But there's value in being able to work with a shared, Microsoft-provided abstraction.

So FusionCache can ALSO be used as an implementation of the new `HybridCache` abstraction.

Oh, and while keeping the added extra features of FusionCache 😬

Cool? Yeah, even if I say so myself.

# HybridCache: Setup

How?

Super easy, barely an inconvenience (cit):

```
services.AddFusionCache()
  .AsHybridCache(); // MAGIC
```

# HybridCache: Usage

Taken as-is, directly from the Microsoft website:

```
public class SomeService(HybridCache cache)
{
    private HybridCache _cache = cache;

    public async Task<string> GetSomeInfoAsync(string name, int id, CancellationToken token = default)
    {
        return await _cache.GetOrCreateAsync(
            $"{name}-{id}", // Unique key to the cache entry
            async cancel => await GetDataFromTheSourceAsync(name, id, cancel),
            cancellationToken: token
        );
    }

    public async Task<string> GetDataFromTheSourceAsync(string name, int id, CancellationToken token)
    {
        string someInfo = $"someinfo-{name}-{id}";
        return someInfo;
    }
}
```

# HybridCache: want more?

Using FusionCache as an implementation of HybridCache gives you:

- most FusionCache extra features: Fail-Safe, Soft Timeout, Auto-Recovery, etc
- can FusionCache + HybridCache at the same time: both protected from Cache Stampede
- sync + async, together: still shared and protected
- more control over L1/L2 setup

Also, multiple instances (see Named Caches) including Keyed Services support!

```
services.AddFusionCache()
    .AsKeyedHybridCache("Foo");

// LATER

public class SomeService([FromKeyedServices("Foo")] HybridCache cache) {
    ...
}
```

# 🥳 FusionCache v2

# FusionCache V2

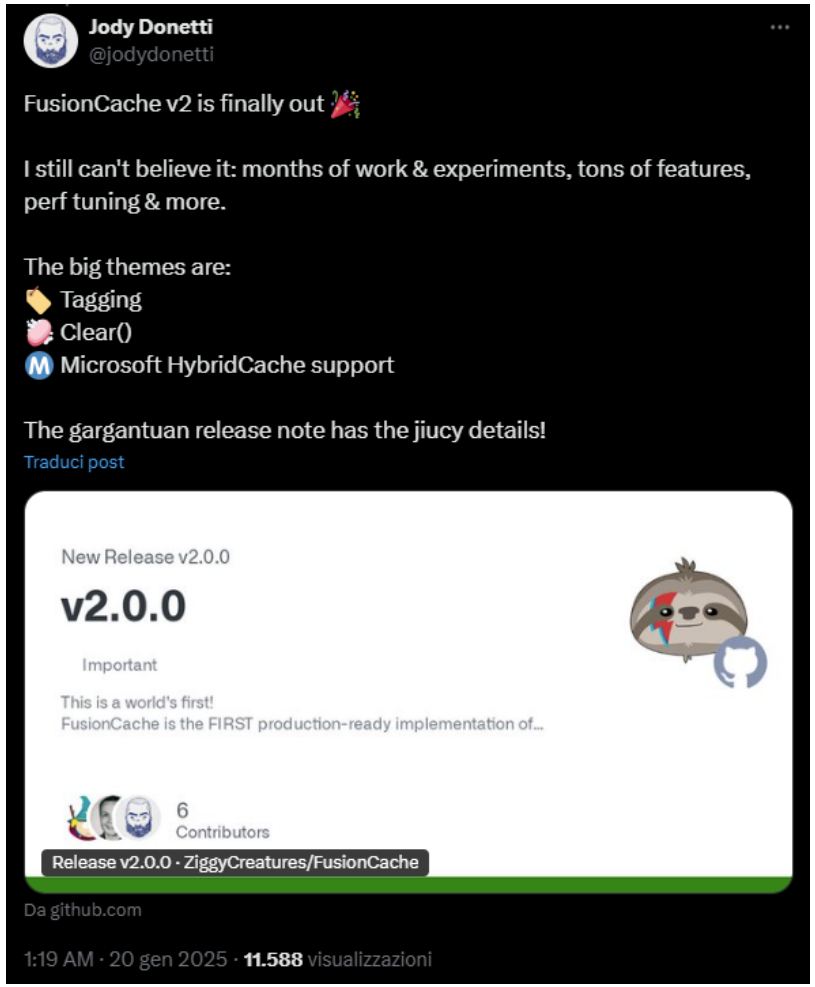
After months of work, in Jan 2025 I finally released FusionCache v2.

Main themes:

- Tagging
- Clear
- Microsoft HybridCache

Also: perf improvements, more flexibility, better observability and more.

Some of these were inspired by HybridCache & Marc (thanks).



# 😳 A strange turn of events

# 😳 A strange turn of events

The timeline:

- Nov 2024: HybridCache **abstraction** released (with .NET 9)
- Jan 2025: FusionCache v2 released
- Mar 2025: Microsoft's HybridCache **implementation** released

So...

> **Important**
>
> This is a world's first!
> FusionCache is the FIRST production-ready implementation of Microsoft HybridCache: not just the first 3rd party implementation, which it is, but the very first implementation AT ALL, including Microsoft's own implementation which is not out yet.
>
> Read below for more.

# ❤️ Microsoft + OSS

A beautiful example of what it *may* look like when Microsoft and the OSS community have a constructive dialog.

### 🙏 Microsoft (and Marc) and OSS

To me, this can be a good example of what it may look like when Microsoft and the OSS community have a constructive dialog.

First and foremost many thanks to the HybridCache lead @mgravell for the openness, the back and forth and the time spent reading my mega-comments.

I think this can be a really good starting point, and future endeavours by Microsoft to come up with new core components already existing in the OSS space should go even beyond, and be even more collaborative: frequent meetings between the maintainers of the main OSS packages in that space, to be all aligned and have a shared vision while respecting each other's work.

With that, Microsoft can provide - if it makes sense in each case - a basic default implementation but even more importantly a shared abstraction, which btw **must** be designed to allow augmentation by the OSS alternatives: in doing so Microsoft must inevitably accept strong inputs from the OSS community to do this well, and yes I know it takes time and resources, but imho it's the only way to make it work.

Then it should also give visibility to the OSS alternatives (in the main docs, samples, videos, etc), and encourage all .NET users to discover and try the alternatives: in doing so they will not lose anything, and instead in turn the .NET ecosystem as a whole will thrive.

In the past this has not always been the case, but the future *may* be different.

Just my 2 cents.

And thanks for the FusionCache mention in the official docs! Needed (imho), but not to take for granted.

# ⏭️ After v2

# Integrate all the things!

With FusionCache v2 out, let's integrate stuff:

- EF Core Second Level Cache Interceptor (by Vahid Nasiri)
- ASP.NET Output Cache, the FusionCache way

Output Cache example:

```
// STANDARD FUSION CACHE SETUP
services.AddFusionCache();

// MAGIC
services.AddFusionOutputCache();

// STANDARD OUTPUT CACHE SETUP
services.AddOutputCache(options =>
{
  options.AddPolicy("Expire2", builder =>
    builder.Expire(TimeSpan.FromSeconds(2))
  );
  options.AddPolicy("Expire5", builder =>
    builder.Expire(TimeSpan.FromSeconds(5))
  );
});
```

And more to come...

# 🎉 Bonus content

# Data API Builder: let’s get distributed!

I’m collaborating with the team to get L2+Backplane to DAB!

[Enhancement]: Add Level 2 Caching #2543

New issue

Open | Feature | Parent: [Enhancement]: "Complete" Cache Features

JerryNixon opened on Jan 29 · edited by JerryNixon

Edits

## Configuration Change

### Original

Runtime

```
{
  "runtime": {
    "cache": {
      "enabled": <true> | <false> (default),
      "ttl-seconds": <integer; default: 5>
    }
  }
}
```

Assignees

aaronburtle

jodydonetti

Labels

enhancement

Type

Feature

Projects

No projects

Milestone

https://github.com/Azure/data-api-builder/issues/2543

# 🈑 Other projects/languages?

Just like other projects inspired me over the years, it would've been nice for FusionCache to be an inspiration for other projects, maybe even in other languages.

And so: **BentoCache** (TypeScript)!

By Julien Ripouteau, see [bentocache.dev](bentocache.dev)

## Prior art and inspirations

Bentocache was inspired by several other caching libraries and systems. Especially FusionCache, which is probably the most advanced caching library I've ever seen, no matter the language. Huge kudos to the author for his amazing work.

## Acknowledgements

The concept of client-side tagging in Bentocache was **heavily inspired** by the huge work done by Jody Donetti on FusionCache.

Reading his detailed explanations and discussions on GitHub provided invaluable insights into the challenges and solutions in implementing an efficient tagging system.

A **huge thanks** for sharing his expertise and paving the way for innovative caching strategies

# 💰 Oh, one last thing

# 💰 No. Nope. Nein. Nada.

## 💰 Support

Nothing to do here.

After years of using a lot of open source stuff for free, this is just me trying to give something back to the community.

Will FusionCache one day switch to a commercial model? Nope, not gonna happen.

Mind you: nothing against other projects making the switch, if done in a proper way, but no thanks not interested. And FWIW I don't even accept donations, which are btw a great thing: that should tell you how much I'm into this for the money.

Again, this is me trying to give something back to the community.

If you really want to talk about money, please consider making 💗 **a donation to a good cause** of your choosing, and let me know about that.

# Thanks

# Contacts

jodydonetti

jodydonetti

jody-donetti

jodydonetti.bsky.social