MGateway Update 2024

Rob Tweed

Director, MGateway Ltd

Twitter: @rtweed

Company Name Change

- M/Gateway Developments Ltd is no more
- Now MGateway Ltd

- WebSite is still the same:
  - https://mgateway.com

Two More Years of Developments

- Lots going on as usual for those who keep track of what we're up to
  – Chris:
    - Interfaces and Language bindings for YottaDB and IRIS (and Cache for legacy users)
  – Me:
    - Front-end and Back-end Application tools and frameworks

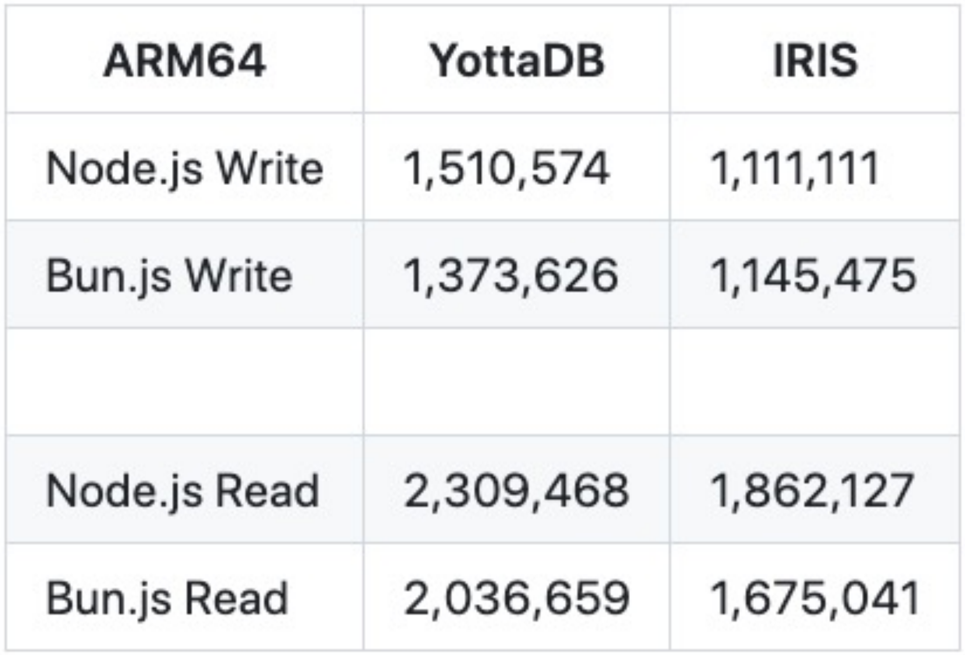
mg-dbx-napi Interface

- JavaScript interface for YottaDB, IRIS & Cache
  - Node.js and Bun.js

  - The fastest JavaScript to Database interface on the planet, by orders of magnitude

Example Performance

- On a standard Apple M1 Mac Mini:
  - Simple key/value create/read loop
  - Results shown in nodes per second:



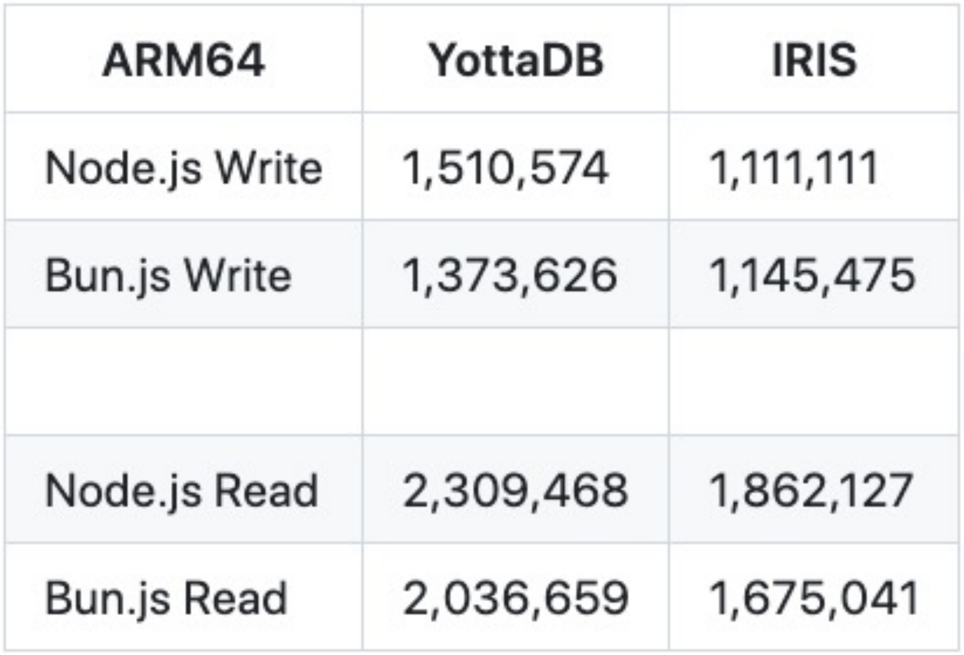
ARM64	YottaDB	IRIS
Node.js Write	1,510,574	1,111,111
Bun.js Write	1,373,626	1,145,475
Node.js Read	2,309,468	1,862,127
Bun.js Read	2,036,659	1,675,041

Example Performance

- On an Apple M3 MacBook Air:
  - Node.js + YottaDB:
    - Sets/writes: 2,000,000 /sec
    - Gets/reads: 3,000,000 /sec

Example Performance

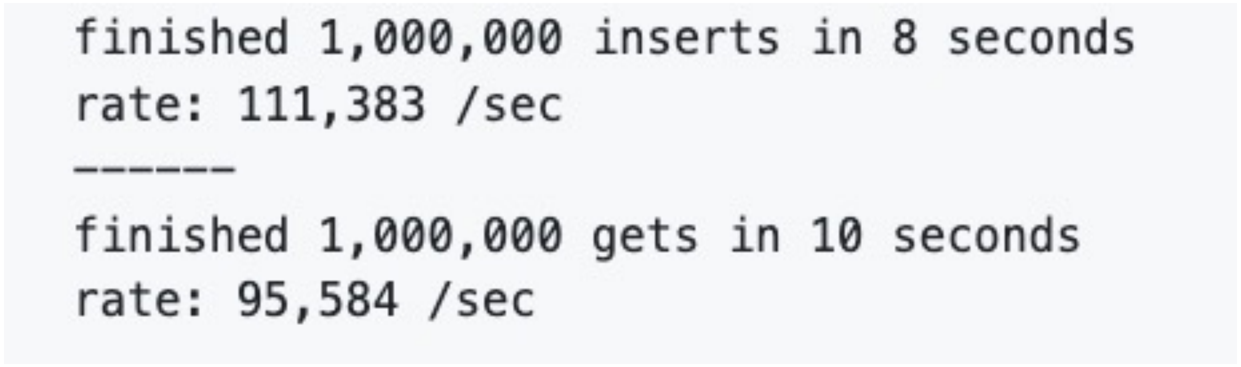
- All comparisons were on M1 Mac Mini, so we'll use these as the reference:



ARM64	YottaDB	IRIS
Node.js Write	1,510,574	1,111,111
Bun.js Write	1,373,626	1,145,475
Node.js Read	2,309,468	1,862,127
Bun.js Read	2,036,659	1,675,041

Comparisons

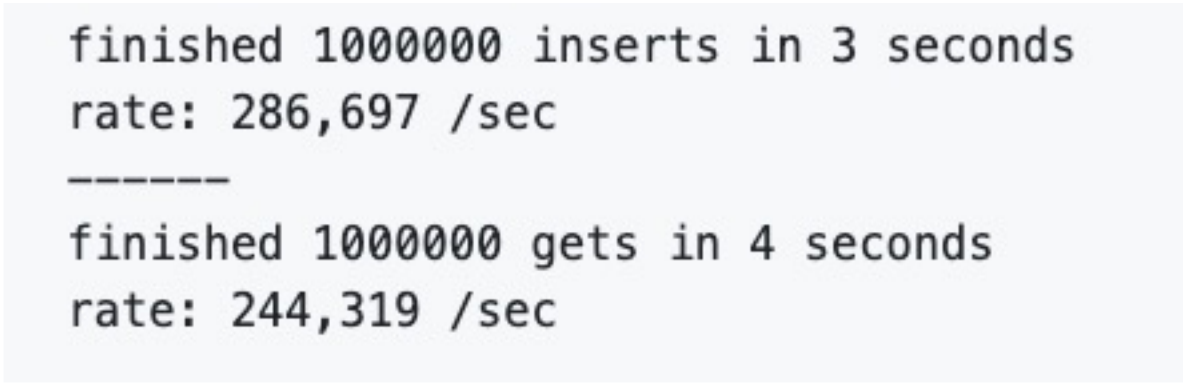
- InterSystems Native Node.js API for IRIS
  - – Network connection only available
  - – Synchronous only!



```
finished 1,000,000 inserts in 8 seconds
rate: 111,383 /sec
------
finished 1,000,000 gets in 10 seconds
rate: 95,584 /sec
```

Comparisons

- NodeM
  - Node.js / YottaDB



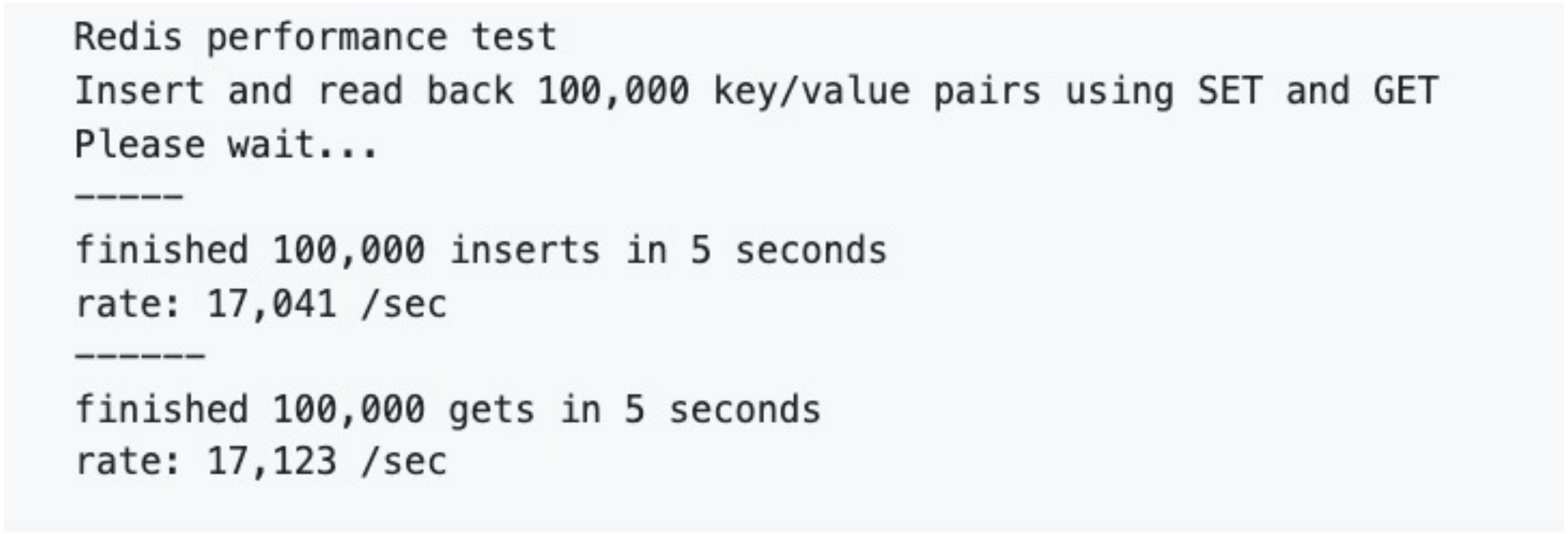
```
finished 1000000 inserts in 3 seconds
rate: 286,697 /sec
------
finished 1000000 gets in 4 seconds
rate: 244,319 /sec
```

Comparisons

- Redis
  - Supposed to be one of the fastest NoSQL databases available
  - Used extensively for high-performance data cacheing
  - How does it compare?

Comparisons

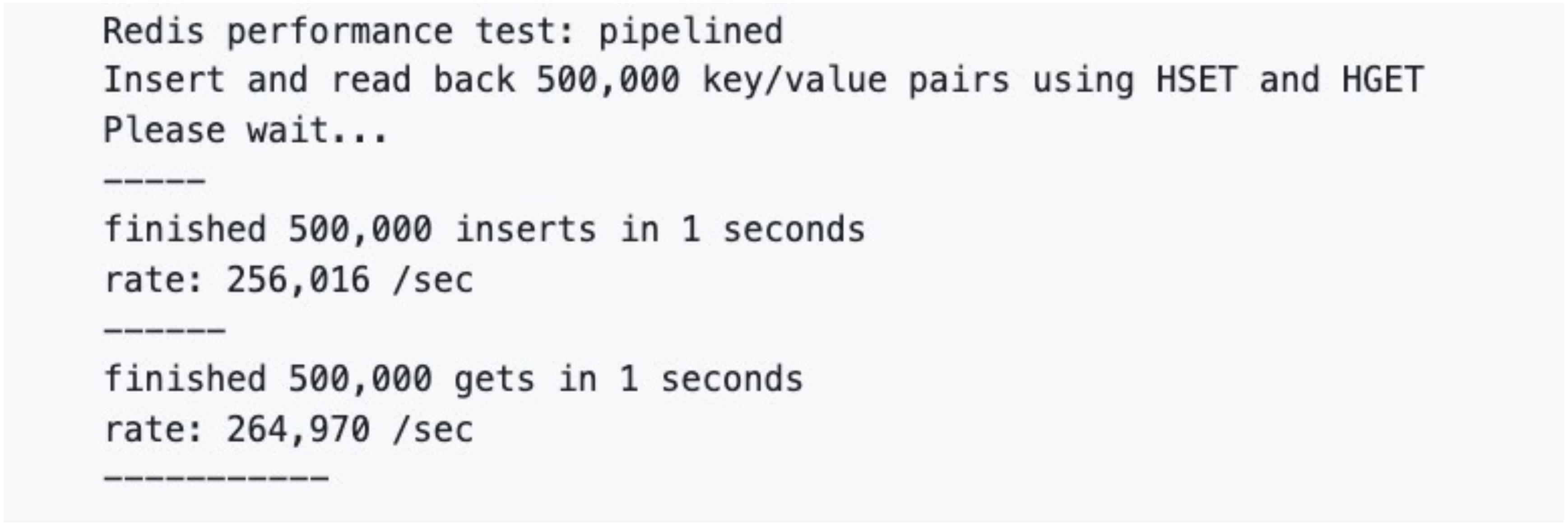
- Redis
  - Node.js / Official Redis Interface Package



```
Redis performance test
Insert and read back 100,000 key/value pairs using SET and GET
Please wait...
-----
finished 100,000 inserts in 5 seconds
rate: 17,041 /sec
------
finished 100,000 gets in 5 seconds
rate: 17,123 /sec
```

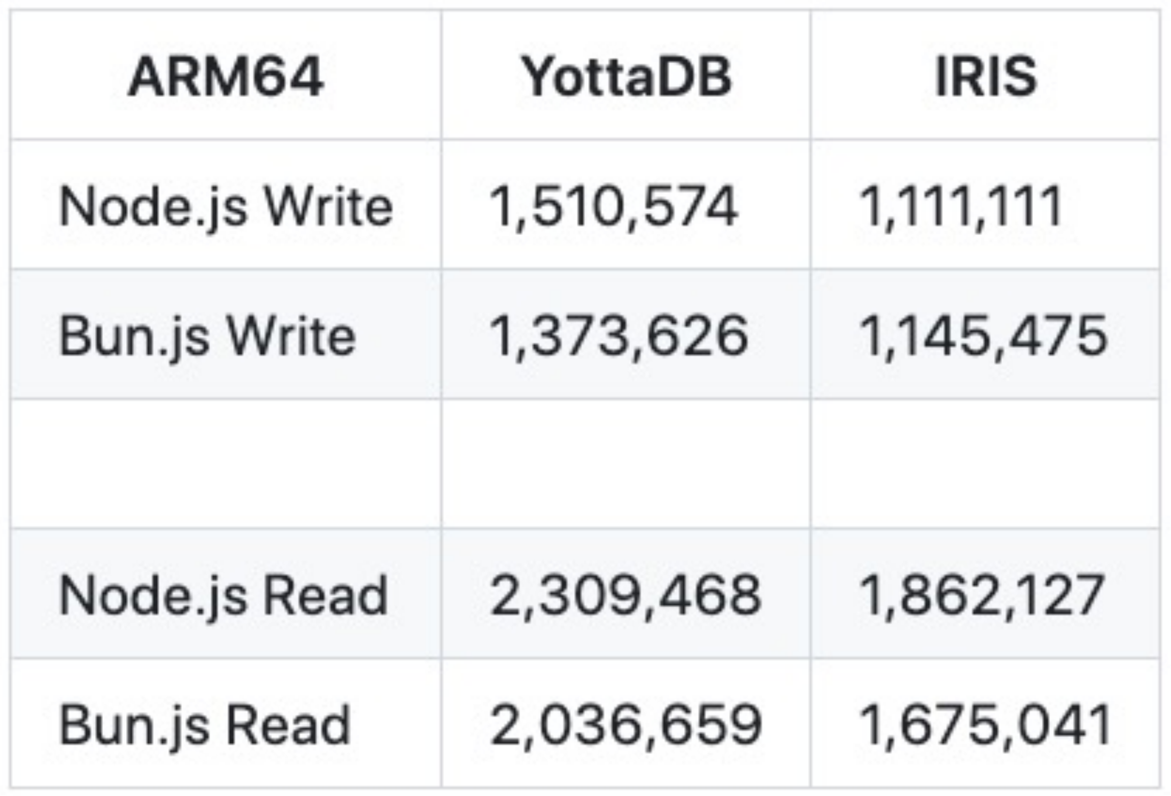
Comparisons

- Redis
  - Pipelined requests



```
Redis performance test: pipelined
Insert and read back 500,000 key/value pairs using HSET and HGET
Please wait...
-----
finished 500,000 inserts in 1 seconds
rate: 256,016 /sec
------
finished 500,000 gets in 1 seconds
rate: 264,970 /sec
-----------
```

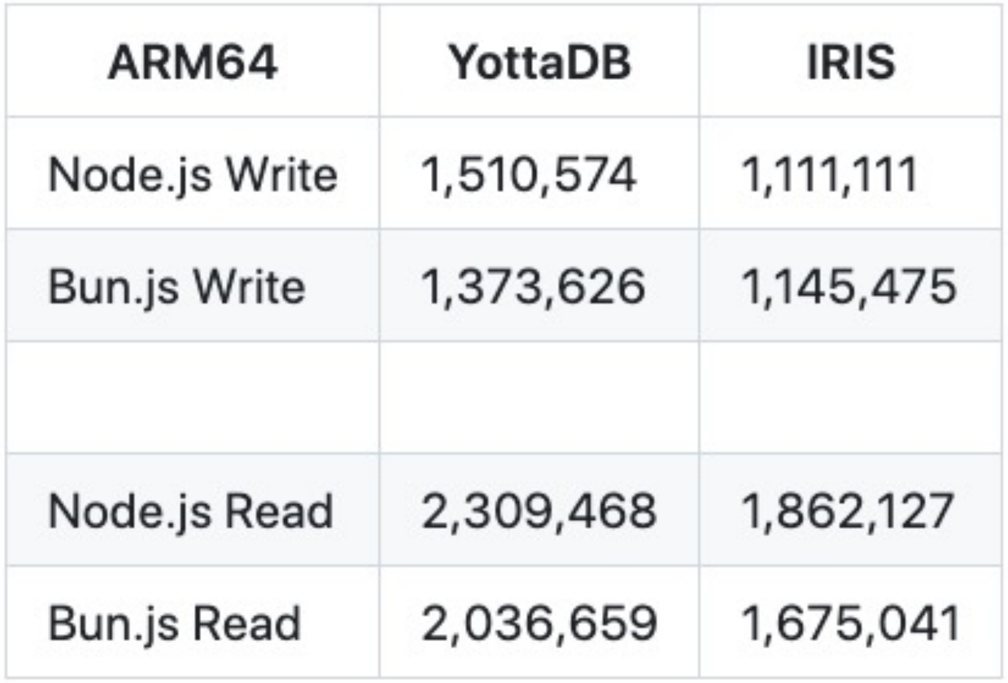
mg-dbx-napi



ARM64	YottaDB	IRIS
Node.js Write	1,510,574	1,111,111
Bun.js Write	1,373,626	1,145,475
Node.js Read	2,309,468	1,862,127
Bun.js Read	2,036,659	1,675,041

Why would you want to use anything else?

mg-dbx-napi



ARM64	YottaDB	IRIS
Node.js Write	1,510,574	1,111,111
Bun.js Write	1,373,626	1,145,475
Node.js Read	2,309,468	1,862,127
Bun.js Read	2,036,659	1,675,041

For Global Storage applications, there's no benefit in using IRIS

You can try it out!

- Our *mg-showcase* Repository
- https://github.com/robtweed/mg-showcase

- Two Dockerfiles you can build and run on your hardware:
  - – One includes YottaDB
  - – One includes the IRIS Community Edition

mg-showcase Containers

- Also include all our other main JavaScript interfaces and frameworks which I'll be covering in this presentation

- No excuse not to try our technologies out
- All you need is Docker
- You'll be up and running in just a few minutes

Our Other Interfaces: Reminder

- Python: mg_python
- Ruby: mg_ruby
- PHP: mg_php
- Go: mg_go

- Work identically on YottaDB, IRIS and Cache

Language Interfaces

- All essentially work the same way
  - Same/equivalent APIs
- Network or in-process API connection
- Access to:
  - M Globals
  - M functions/procedures
  - IRIS/Cache Classes and SQL
- All support YottaDB transactions

YottaDB Integration: reminder

- Process Window: *mg_pwind*

- Access to crypto libraries from YottaDB
- Access from YottaDB to:
  - IRIS/Cache Globals, functions, procedures
  - IRIS/Cache Classes
  - IRIS/Cache Transactions

mg_web

- Becoming a key and very important product:

What is *mg_web*?

- Web Server Add-on for the industrial-strength Web Servers:
  - NGINX
  - Apache
  - IIS

What does *mg_web* do?

- Designed to interface these Web Servers with YottaDB, IRIS or Cache
  - Directly via network or API connection:
    - API (in-process) connection for highest levels of performance
  - Indirectly via a Node.js server process and *mg-dbx-napi*

What is *mg_web* for?

- Designed primarily to support REST APIs
  - Direct connection:
    - Business logic in M / ObjectScript
    - Data storage in Globals or Classes
  - Indirect connection:
    - Business logic in JavaScript
    - *** Data Storage as persistent Objects / JSON ***:
      - Abstracted via our *glsdb* package

mg_web is so much more!

- Not limited to REST interfacing
- Behaviour can be adapted by writing "shims"
- As a result, it can emulate pretty much any other Web Interface for YottaDB, IRIS or Cache

mg_web and WebLink

- WebLink was InterSystems' original web gateway before CSP
  - Many legacy systems out there still use it
- InterSystems no longer support *WebLink* on IRIS
- *mg_web's WebLink* shim makes *mg_web* behave identically to the WebLink interface
- InterSystems have endorsed and supported its use for its major WebLink user, allowing them to migrate to IRIS

mg_web

- At least one group within the VA who use WebLink are assessing *mg_web* to allow migration to IRIS

mg_web

- If you use WebLink (eg with WebLink Developer)
- And if you need to migrate to IRIS
- … then *mg_web* is the solution you need
- You could even use it to migrate to YottaDB instead!

mg_web

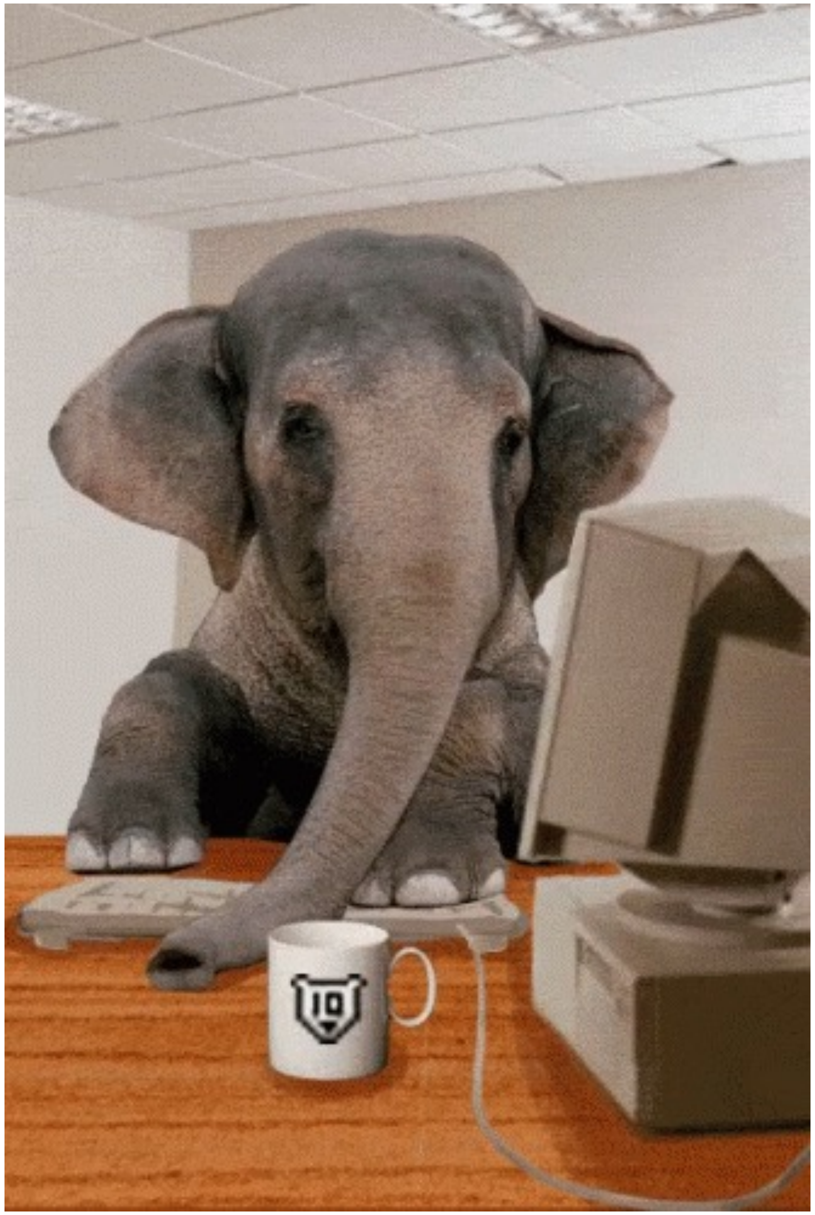
- Completely stateless, just like WebLink
- Modern architecture, making use of the capabilities built-in to NGINX, Apache and IIS.
- Create a pool of Web Server workers
- Excess traffic is queued by Web Server
- Can therefore support any number of concurrent users without any license issues

mg_web

- If you use *mgwsi* with GT.M or YottaDB:
  - *mgwsi* is now deprecated
  - You should migrate to *mg_web* and YottaDB
  - We can assist, if you pay for our support

mg_web for Node.js Users

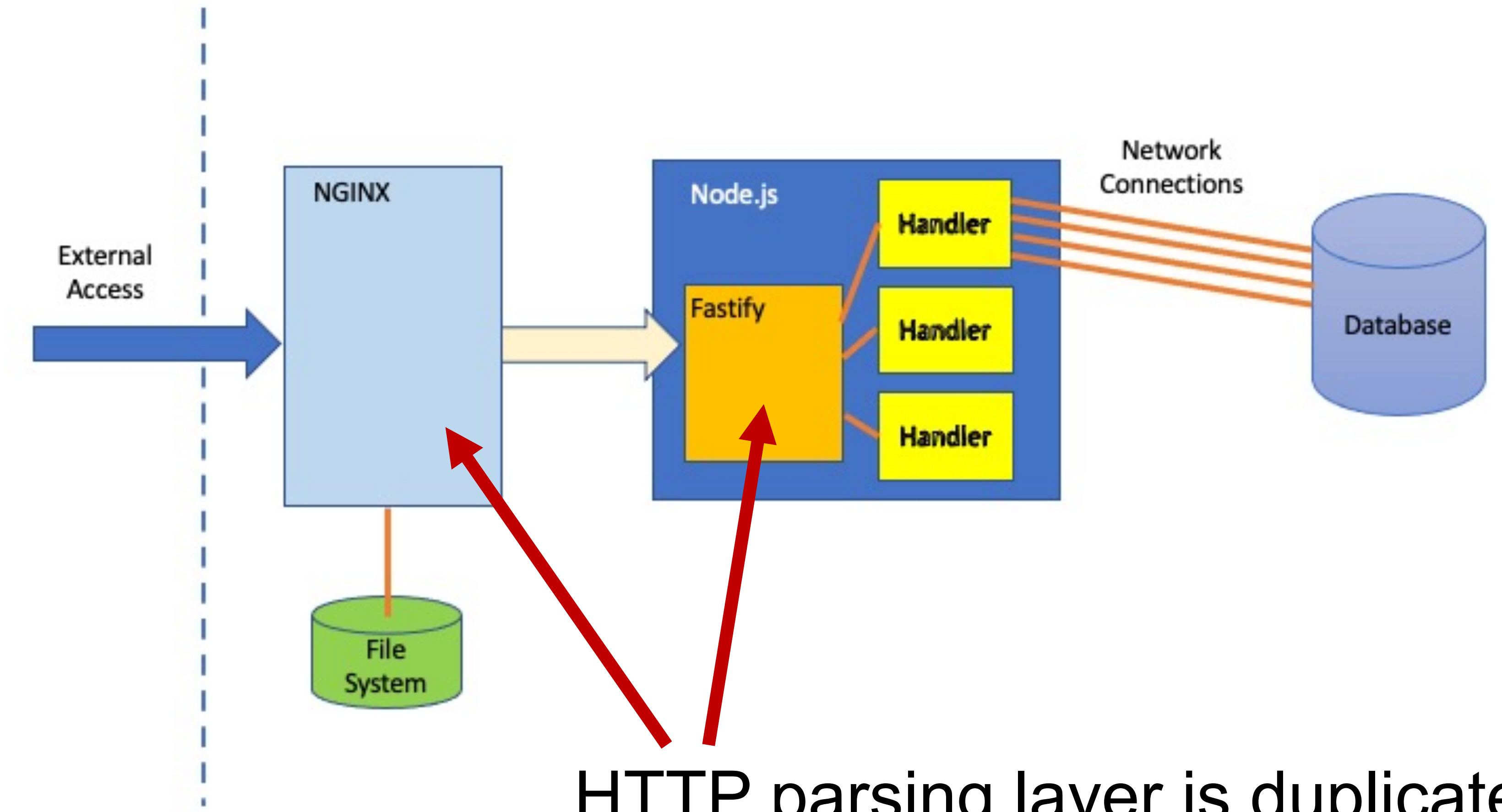
- The most common Node.js Web Framework is Express. Its performance is not great, and it's becoming outdated
- The fastest Node.js Web Framework is Fastify
  - Heavily promoted as the best thing since sliced bread



Security Advisory

- Always put a public-facing Node.js Web Framework behind a reverse proxy such as NGINX
  - Even Fastify officially (and very stongly) recommends this

The Elephant in the Room



HTTP parsing layer is duplicated

Performance Comparisons

- All performed on the same M1 Mac Mini, using a "do nothing" request/response
- Figures shown are the best I could obtain, using various configuration tweaks

Performance Comparisons

- NGINX alone:
  - 200,000 requests/sec using 4 workers

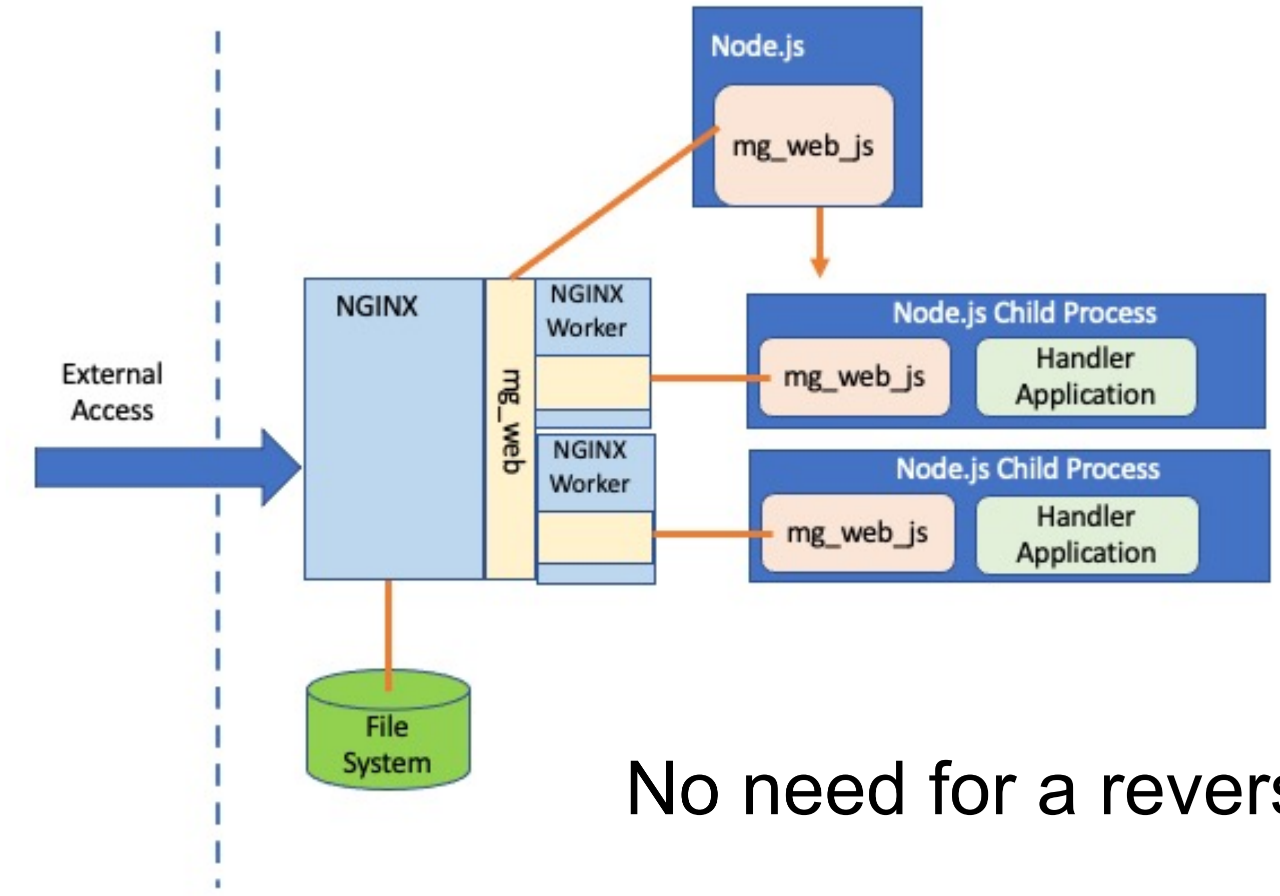
Performance Comparisons

- NGINX alone:
  - – 200,000 requests/sec using 4 workers
- Fastify alone:
  - – 50,000 requests/sec

Performance Comparisons

- NGINX alone:
  - 200,000 requests/sec using 4 workers
- Fastify alone:
  - 50,000 requests/sec
- Fastify proxied with NGINX:
  - 21,000 requests/sec

mg_web.js and NGINX



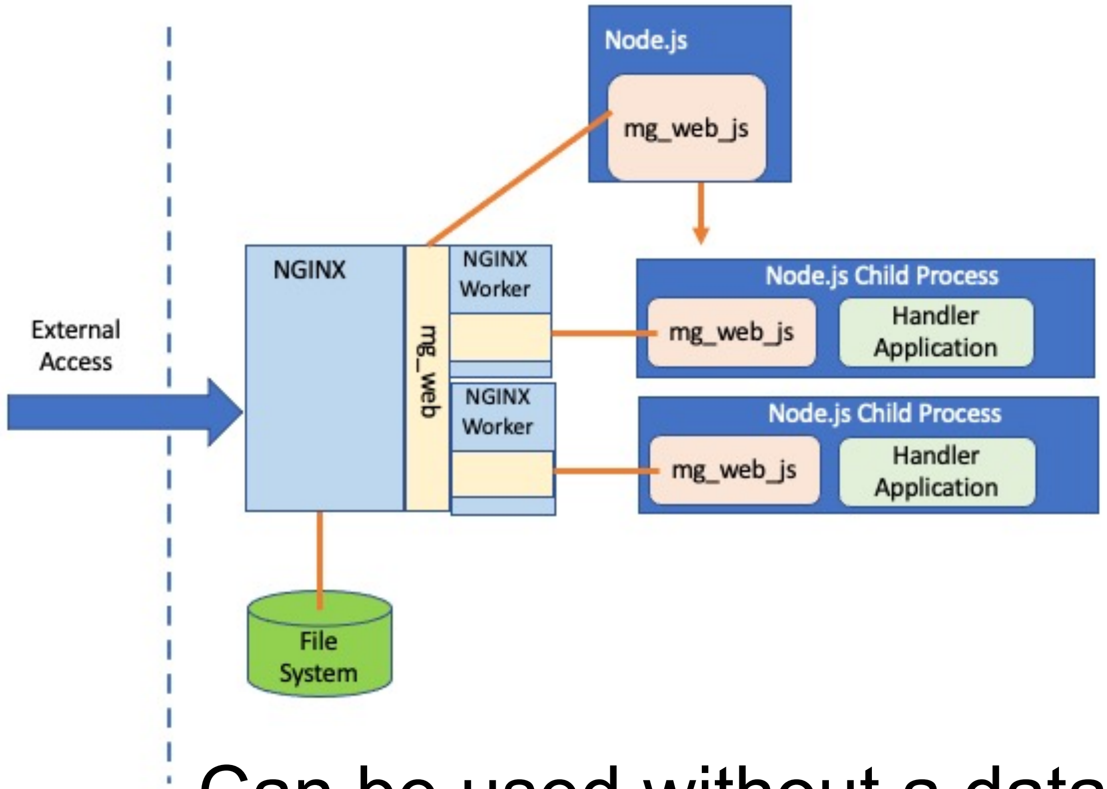
No need for a reverse proxy

Performance of NGINX + mg_web.js

- Same do-nothing request/response:

- 64,000 requests/sec
  - 3 X Fastify + NGINX proxy !

mg_web.js and NGINX



Can be used without a database
as a faster alternative to Fastify + Proxy

mg_web_router

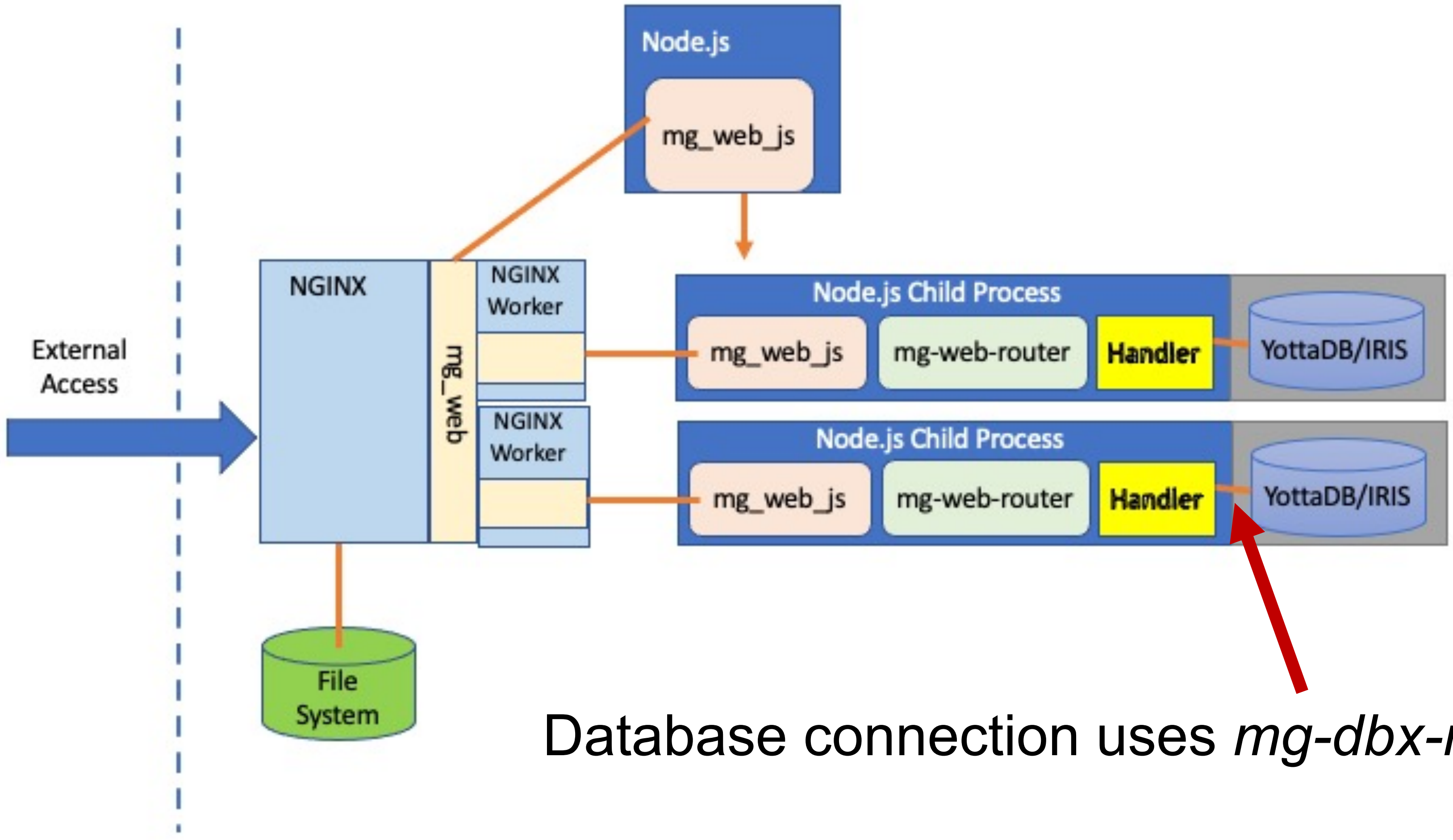
- Express-like router for *mg_web*
- https://github.com/robtweed/mg_web_router

```
router.get('/mgweb/helloworld', (Request, ctx) => {

  return {
    payload: {
      hello: 'world 123',
    }
  };

});
```

mg_web with YottaDB or IRIS



Database connection uses *mg-dbx-napi*

Performance with/without Database

- Without database (do-nothing)
  - 64,000 requests/sec

- With YottaDB (response read from database):
  - 58,000 requests/sec

Typical Fastify + DB Performance

- Fastify (*without NGINX proxy*)
  - Standalone: 50,000 requests/sec
- MongoDB
  - 9,000 requests/sec

mg_web: Fastest Node.js Web Framework

- *mg_web* + NGINX accessing YottaDB is faster than Fastify running standalone without an NGINX Proxy!

REST Server for M Developers

- Pre-built packaged framework
  - *mgweb-server*
  - Available as:
    - Native application
    - Pre-built Docker Container

mgweb-server

- All written in standard M code
- Define REST API Routes
- Connect Routes to Handler Functions:
  - – M globals, functions, procedures
  - – IRIS/Cache Classes & SQL
- *** In-built mapping between M arrays/globals and JSON ***

More information

- Visit our Web Site
  - www.mgateway.com

MetaStatic

- Our WebSite has been re-written using our very latest technology:
  – MetaStatic
  – A syntax and builder script to allow easy maintenance of a modern web site by non-technical people
    - All that's needed is a basic understanding
      – HTML/XML tags and attributes
      – Markdown
      – Editing text files using any favourite editor

The Problem

- Many clubs, societies, small organisations need a web site that is modern and responsive:
  - ie it works on both desktop and mobile devices and alters the layout automatically based on available screen size
- People tasked with maintaining the site typically have little or no technical skills, but know what content they want to use
- Often done in their spare time

What to use?

- Modern Front-end framework such as React, Vue, Angular?
  - Requires serious technical knowledge to build AND maintain
  - So any downstream maintenance will require technical resource

What to use?

- A pre-built template such as W3 Schools or SB Admin
  - Requires a lot of laborious and error-prone work to maintain content
  - Requires detailed CSS knowledge to make changes to UI behaviour if you need to do more than provided by the template
  - Too time-consuming
  - May require on-going technical resource

What to use?

- A subscription-based system + Content Management System, eg:
  - Wordpress
  - Squarespace
  - WIX
    - Locks you in.
    - Expensive
    - Fine if the available templates fit your needs

MetaStatic

- Splits the tasks:
  - One-off technical task to deconstruct a template into re-usable building-blocks represented by Meta Tags (+ attributes to control them)

MetaStatic

- Splits the tasks:
  - Content held in simple text files using Markdown syntax
  - Categorisation simply using naming conventions

  - No CMS to learn or database to learn and maintain

MetaStatic

- Splits the tasks:
  - – Site design and maintenance:
    - Describe using Meta Tag building blocks
    - No programming at all
      - – Tags + attribute values to control behaviour
- Then run MetaStatic build script to create a single HTML file:
  - – Essentially a custom version of a template such as SB Admin

Performance

- All content files are pulled in at build time
- Single HTML file that includes any custom CSS and JavaScript
- No heavyweight JS framework to load

- Sub-second load times
- Try out our web site and see for yourself!

More information

- Visit our Web Site
  - www.mgateway.com