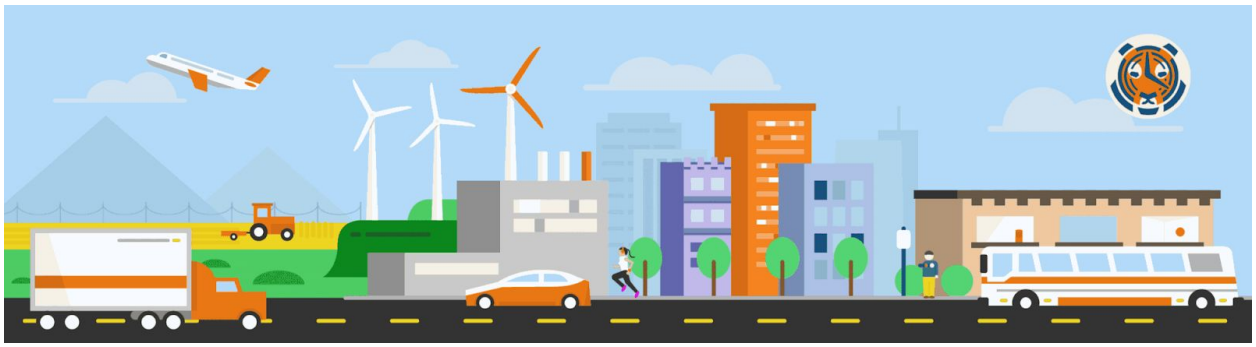
Timescale raises $16M from Benchmark, NEA, Two Sigma Ventures for the database to help us understand our machines and our world

Humanity is now living with machines and swimming in machine data. Timescale just raised $16M to help developers, businesses, and society make sense of it all.

—Ajay Kulkarni, Co-Founder/CEO of Timescale



We are excited to announce that Timescale has raised $12.4M in a Series A financing led by Benchmark, with participation from NEA, Two Sigma Ventures, and prominent angels, including the CEOs/founders of Hortonworks, Cloudflare, Data Domain, Nicira, Barefoot Networks, Grand Central Tech, Right Media, Moat, Segment, and Cockroach Labs.

Combined with our earlier (never before announced) $3.7M Seed financing led by NEA, Timescale now has raised over $16M to address what we believe is one of the largest challenges (and opportunities) in databases for years to come: helping developers, businesses, and society make sense of the data that our machines are generating in copious amounts.

With this financing, Benchmark's Peter Fenton will join NEA's Forest Baskett and co-founders Ajay Kulkarni and Mike Freedman on the Timescale Board of Directors.

"Timescale has struck a nerve with the open source community by seeing the need for a new kind of time series database at just the right moment in time. At Benchmark we've seen time and again that this kind of community and adoption around a project in the early days can lead to transformative businesses, and I look forward to working side-by-side with Mike and Ajay on this journey."
— Peter Fenton, General Partner, Benchmark

A new database for the machine era

Today we live with machines. Computers are nearly ubiquitous, embedded in our vehicles, our manufacturing lines, our farms, our power plants, our homes, even in our bodies. As a result, businesses are swimming in machine data. And if we want these machines to truly make our lives more productive, more convenient, and more enjoyable, then we need a way to properly make sense of all that machine data.

We realized that this problem needed a new kind of database, and so we built TimescaleDB: the first open-source time-series database designed for scalability and complex analysis.

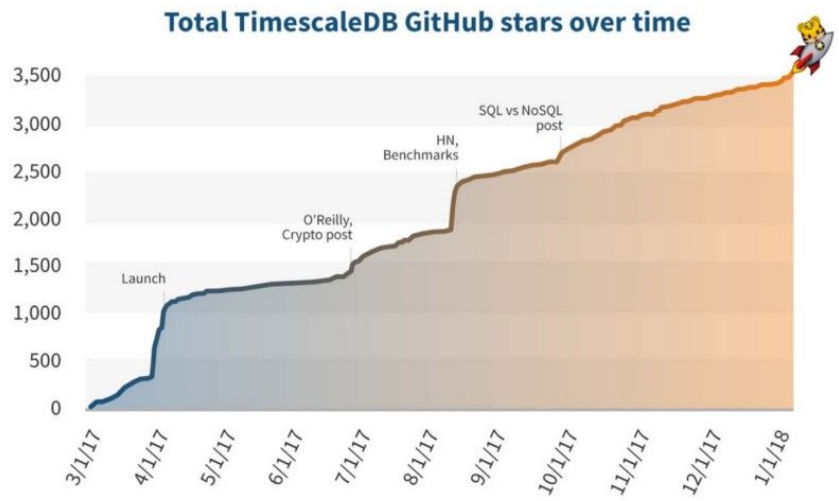
TimescaleDB solves this problem with a unique approach. First, it employs automatic space-time partitioning to seamlessly scale for time-series workloads. But at the same time, TimescaleDB is packaged as a Postgres extension, enabling it to inherit the reliability, tooling, and vast ecosystem of Postgres. (It does this while sustaining 20x higher inserts than Postgres, with more benchmarking comparisons here.)

Thanks to this native Postgres integration, at its launch last April, TimescaleDB already enjoyed the largest ecosystem of any time-series database. And a few months later, businesses were already successfully deploying TimescaleDB into production.

"Operationalizing time-series data is a ripe opportunity, and with their approach Timescale is poised to lead this market."
— Rob Bearden, CEO, Hortonworks

Rapid adoption: 100,000+ downloads, 9 months

Today, after just 9 months, developers have downloaded TimescaleDB over 100,000 times, finding it superior for their workloads to Oracle, SQL Server, Cassandra, Redis, other time-series databases, and other established database systems. Businesses all over the world trust TimescaleDB for powering mission-critical applications, including industrial data analysis, complex monitoring systems, operational data warehousing, financial risk management, and geospatial asset tracking. TimescaleDB is deployed in industries as varied as manufacturing, utilities, oil & gas, mining, smart spaces, ad tech, finance, telecom, and more..



From 0 to 3,500: TimescaleDB growth reflected in its GitHub stars, a mechanism that allows GitHub users to bookmark open source repositories that interest them

TimescaleDB users include:

- STE Energy, to back operational dashboards used to monitor 47 power plants in Italy, Albania, Colombia, Perú, Honduras, Costa Rica, and Chile, replacing Redis
- InTraffic, which monitors all of the roughly 5,000 remote controllable railroad switches of the Dutch railways, for use with Tableau as a Business Intelligence solution
- One of the largest American telecom companies, for storing server health data
- A top financial services firm, to replace Oracle for powering interactive analytics of market data
- A public lighting and semiconductor company, for deployment in a small-footprint IoT edge device collecting and analyzing sensor readings

"TimescaleDB adds the critical yet missing time-series puzzle piece to Postgres, simplifying geo-financial analytics and data visualization on temporal+spatial datasets."
— Erik Anderson, Lead Software Engineer, Bloomberg

Download TimescaleDB today

TimescaleDB is open-source under the Apache 2 license, and can be installed today via various methods or directly from source. And if you'd like to reach out, please feel free to contact us via email (hello@timescale.com), or join us on Slack.

By one measure, today is just a single milestone on a decades long journey for Mike and myself. When we were roommates at MIT over 20 years ago, Mike and I studied, built projects, and even ran the Boston Marathon together. Today we find ourselves again working together, surrounded by an incredible team of highly-technical, hard-working, customer-centric, yet also humble, helpful, curious, and open-minded individuals. We feel fortunate to have the opportunity to tackle such a large, fascinating problem with such a talented team.

If this is the kind of mission that excites you, and this is the type of team that you'd like to be a part of, we're hiring across technical and non-technical roles, and we encourage you to apply.

We're proud of everything our team has achieved in less than a year, but know that we're still just getting started. This new funding will help us continue to support our fast-growing community, grow production deployments, develop additional tools and capabilities, and hire the best possible team to make all this happen.

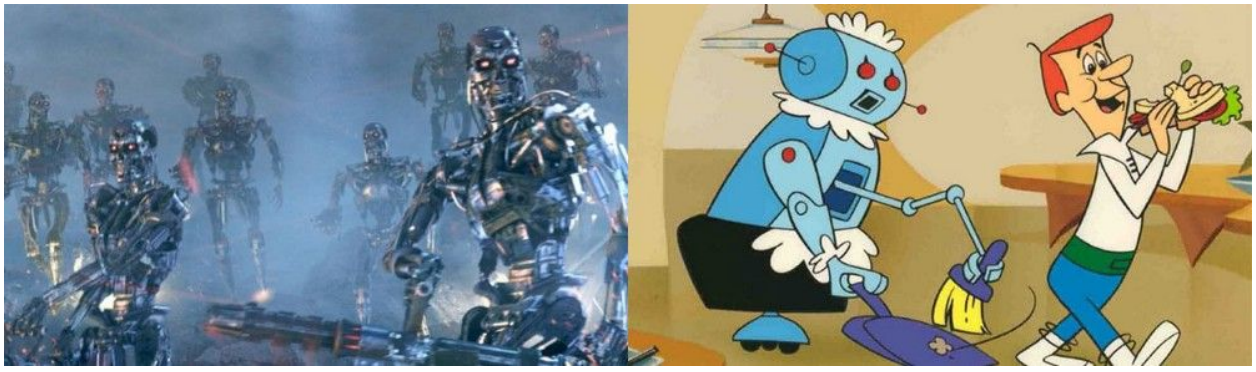
While it's clear from our adoption and community support that Timescale is filling a much-needed gap in the market, some of you may be wondering why, and why now?

imescaleDB is an extension to PostgreSQL and can be combined with many other PostgreSQL extensions to provide more flexibility and power. PostGIS gives you new data types, functions, index types, etc. to deal with geospatial data. TimescaleDB + PostGIS means you can explore geospatial and time-series data with a single query.

We'd like to point to 3 reasons:

1. The Rise of Machines
2. The Evolution of Data Resolution
3. The Resurgence of Postgres

Reason 1: The Rise of Machines



Two diametrically opposed visions of the future. In order to navigate this decades long journey, preferably towards a benevolent future, we need to understand the data our machines are generating.

Historically, the computing industry has evolved in waves: mainframes, desktops, laptops, smartphones. With each wave, computers became smaller, yet more powerful, and more prolific. Microsoft famously strove for "A Computer on Every Desk"; smartphones are ensuring a computer in every pocket. And today we have entered a new wave, with computers so small and pervasive that they are in "every thing."

Across that same history of computing, humans have generated most of our data: financial transactions, Internet web pages, email, social media, photographs, video. And while that data has grown exponentially, leading to the "big data" industry, it is limited by two critical bottlenecks: our own ability to create data, and human population growth.

Machines don't have those same limitations. Their ability to generate data is limited by their hardware, which continues to get better, cheaper, and faster. And our ability to replicate them is only bounded by our fabs and assembly lines. Indeed, as of last year, connected machines now outnumber human beings, and will grow in number much faster than we can.
The machines are here, and they need a new kind of database.

All of these machines are constantly sensing, thinking, acting. And in a world with abundant Internet connectivity (including cellular, WiFi, Bluetooth, and wired connections), they are also constantly communicating to us and to each other.

We are living with machines and swimming in machine data.

Before we started developing TimescaleDB, we recognized a few key characteristics of this kind of data. There was a lot of it. It needed to be analyzed quickly, often in complex ways, but also reliably. (For example, if you're monitoring your power plant, you need to know quickly if there's a problem, without much room for error.) And it was largely "time-series" in nature: timestamp, measurement, metadata; collected regularly or irregularly; and collected often, sometimes multiple times a second.

Yet at that point, developers only had two choices for storing this data: relational databases, which were reliable, powerful, and easy-to-use, but hard to scale; or NoSQL databases, which scaled, but not for complex workloads, and were less reliable and harder to use.

We thought this was a terrible choice, and after careful study, a false one. We realized that, thanks to the nature of time-series data, there was a way to combine the best of both worlds and build a new database.

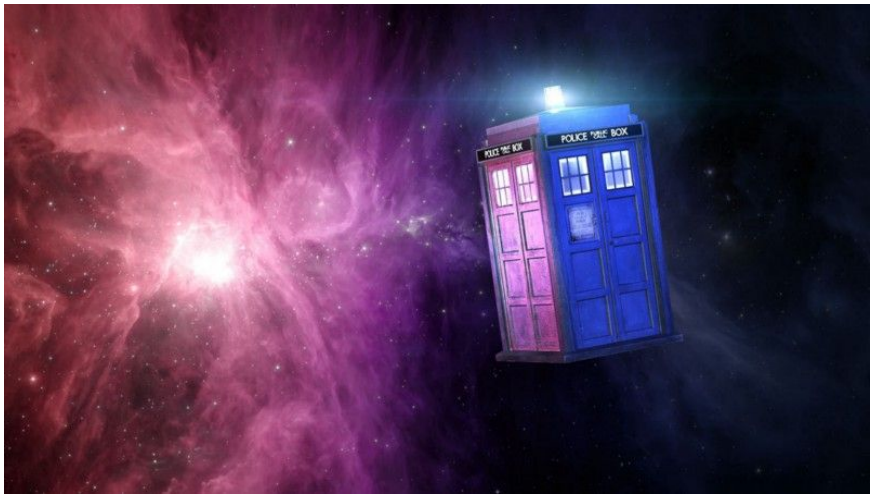
The machines are here, and they need a new kind of database.

And so we built and launched TimescaleDB, the first open-source time-series database to combine the power, reliability, and ease-of-use of a relational database with the scalability typically seen in NoSQL databases. And while the response from the developer community since then suggests that we are on the right track, we're just getting started.

Of course, the rise of machines is not without its own set of challenges: How will they affect human employment? How will they affect human society? How do we ensure benevolent machines that improve the quality of life for every human being, as opposed to things that cater to the few, let alone the dystopian killers that Hollywood likes to portray?

These questions are not easy to answer. But the first step to understand our machines is to understand the data they are generating.

Reason 1: The Rise of Machines



The future holds more and more time-series data, thanks to the perpetual march towards cheaper storage and more powerful computing

While TimescaleDB has its roots in understanding machines, its usage is not limited to machine data. For example, our users include financial services companies storing market time-series data for interactive analytics, advertising technology companies storing eventing data

to understand human behavior, and logistics companies storing package location and meta data to manage distribution networks in real-time.

And so, a second reason why TimescaleDB is solving a real problem today: the evolution of data resolution.

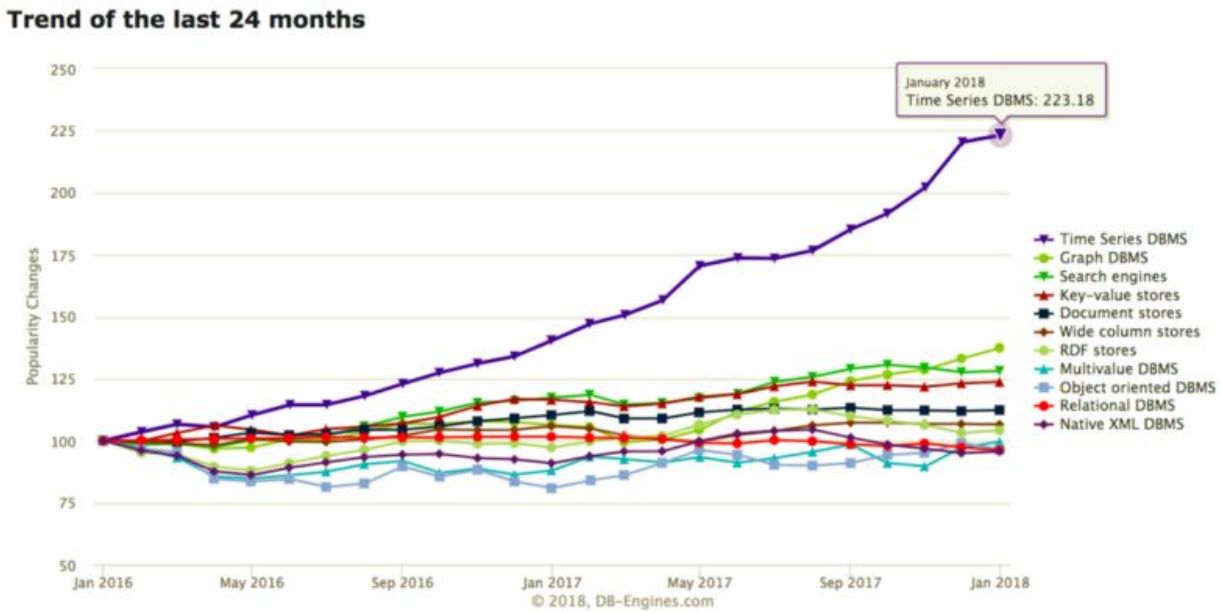
Let's travel through time for a moment. Early on in the computing era, storage was expensive, so we stored the least amount of information necessary (e.g., the current balance of each bank account). Then, as storage became cheaper and cheaper (outpacing even our revered Moore's law), we began to store data at higher resolutions (e.g., every bank transaction) and even higher resolutions (e.g., every customer banking interaction).

The higher resolution data you capture, the more time-series data you are storing. Bank transactions are time-stamped events; customer interaction data represents a larger number of time-stamped events.

Storing data at higher resolutions is valuable. It allows us to analyze trends over time, studying the past to help predict the future. Bank accounts reflect the current state of the system; bank transactions indicate how that state is changing; tracking every interaction can help predict how the customer relationship is changing. Each level introduces additional, vital information.

Thanks to the perpetual march towards cheaper storage and more powerful computing, more and more of our data is evolving into time-series data.

You can see this trend in the emergence of time-series as the fastest growing category of databases today:



Time-series is the fastest growing database category today (Source: DB Engines, January 2018)

As an industry, we started storing these large datasets in data warehouses, and later, massive data lakes. But storing data is not enough; we also need to quickly access it, analyze it, act on it.

Businesses need an operational time-series data storage layer. Businesses need a time-series databases.

Reason 3: The Resurgence of Postgres

There is also a third, more technical reason why TimescaleDB is the right solution for this problem.

When we first recognized the explosion of time-series and machine data, and that it required a new kind of database, the obvious path would have been to build an entirely new database from scratch.

But we took a different, heretical route: we decided to build TimescaleDB on top of Postgres, a 20-year old, seemingly boring database.

We did this for several reasons:

1. We realized that not rebuilding all the layers of a database from scratch would allow us to solve this problem sooner (as opposed to the 5–10 years it typically takes new databases to become "enterprise-ready")
2. We liked the reliability, ease-of-use, and ecosystem of Postgres
3. We saw how the Postgres extension framework would allow the development of a powerful new database that solved Postgres's time-series weaknesses while retaining its strengths
4. And because we understood that sometimes boring is awesome, especially when it comes to your database. We wanted to build something that would just work, not wake you up at 3am.

This decision turned out to be the right decision, at exactly the right time.

2017: The year of Postgres

Rank			DBMS	Database Model	Score		
Jan 2018	Dec 2017	Jan 2017			Jan 2018	Dec 2017	Jan 2017
1.	1.	1.	Oracle	Relational DBMS	1341.94	+0.40	-74.78
2.	2.	2.	MySQL	Relational DBMS	1299.71	-18.36	-66.58
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1148.07	-24.42	-72.89
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	386.18	+0.75	+55.81
5.	5.	↓ 4.	MongoDB	Document store	330.95	+0.18	-0.96

In 2017, Postgres was the fastest growing database in the world. Source: DB Engines (January 2018)

We chose to build on top of Postgres just as Postgres was becoming the fastest growing database in the world.

In an age where tech companies continuously roll out shiny new products, how did this ancient database with roots in the 1980s (and a name only a Postgres developer could love) beat every other database management system? Especially when all other major relational databases (Oracle, MySQL, SQL Server) lost market share?

We can point to a few reasons behind the resurgence of Postgres (also captured in our Tweetstorm from earlier this month):

The exodus from Oracle to open-source
One of the software industry's worst kept secrets is that large companies are migrating away from proprietary databases like Oracle and SQL Server to open-source options. This exodus reportedly includes large Oracle customers like Salesforce and Amazon.

Postgres is the leading open source relational database system that is not owned by Oracle (which owns MySQL). Postgres also sports a large community, offers a broad set of features, and, thanks to 20+ years of development, is quite reliable. So many businesses leaving proprietary databases like Oracle have headed towards Postgres, and in turn to us. Users already have moved their workloads away from Oracle to TimescaleDB.

The SQL comeback
For decades, SQL was the de facto interface for businesses to access their data. Yet in the past several years, the continued growth of the Internet and "big data" caused many software developers to cast aside SQL and relational databases as relics that couldn't scale with these growing data volumes, leading to the rise of NoSQL.

Yet today, we find that the pendulum is swinging back to SQL. Many engineers have discovered that NoSQL query languages lack the richness of SQL, and that the SQL ecosystem of tools, clients, and visualization options far outsizes that of any NoSQL database. Database developers have also found new ways to scale SQL in ways similar to a NoSQL database.

Not only does the SQL comeback drive developers to Postgres, but also to TimescaleDB, which is the only time-series database that natively supports SQL.

The growth of the Postgres community

There are a few ways we can measure the growth of the Postgres community. One is by the development of new features in the core product. In the past couple years, Postgres core developers have introduced a variety of useful new features: support for JSON/JSONB, full text search, better support for partitioning, improved parallelization, and more.

Another is by the development of new features on top of the product. One of the wonderful aspects of Postgres is its extension framework, which enables third-party developers to easily build and integrate entirely new functionality on top of the database. Popular extensions include PostGIS for geospatial data, hstore for storing sets of key-value pairs in a single value, and of course TimescaleDB for time-series data.

Other indications of Postgres community growth are the enthusiastic support by the major cloud providers (Amazon AWS RDS and Aurora, Microsoft Azure Database for Postgres, Google Cloud SQL for Postgres), and the increasing attendance at Postgres meetups and conferences.

All of this means that, as part of the Postgres community, TimescaleDB benefits from an increasing number of features, capabilities, and deployment options, which we build on top of and offer to our users.

Come join us

Thanks to the rise of machines, the evolution of data resolution, and the resurgence of Postgres, the world today needs a new kind of database optimized for time-series, scalability, and complexity, while also reliable and easy-to-use. Postgres provides an ideal platform: reliable and mature, yet still burgeoning.

TimescaleDB is that database, and we look forward to helping developers, businesses, and society make sense of their machine and time-series data.

If you'd like to give it a whirl, you can install TimescaleDB via various methods or directly from source (Github). Also feel free to reach out via email (hello@timescale.com), or find us on Slack. Also, we are currently hiring across technical and non-technical positions.

Whichever path you choose, please come join us, and together we will understand our machines and build a better world.

Tempus neminem manet!