Carnegie Mellon University

Andy's Guide on

How to Get Tenure in Databases

@andy_pavlo

Research Papers
Classes Taught
Grants Funded


Research Papers
Classes Taught
Grants Funded

of Crazy Emails!

- → Physics: $E \neq mc^2$
- → Math: Fermat's Thm
- → ComSci: P=NP

of C

→ Phys

→ Math

→ Com

new database with constant time operations

From: [redacted]

To: pavlo@cs.cmu.edu

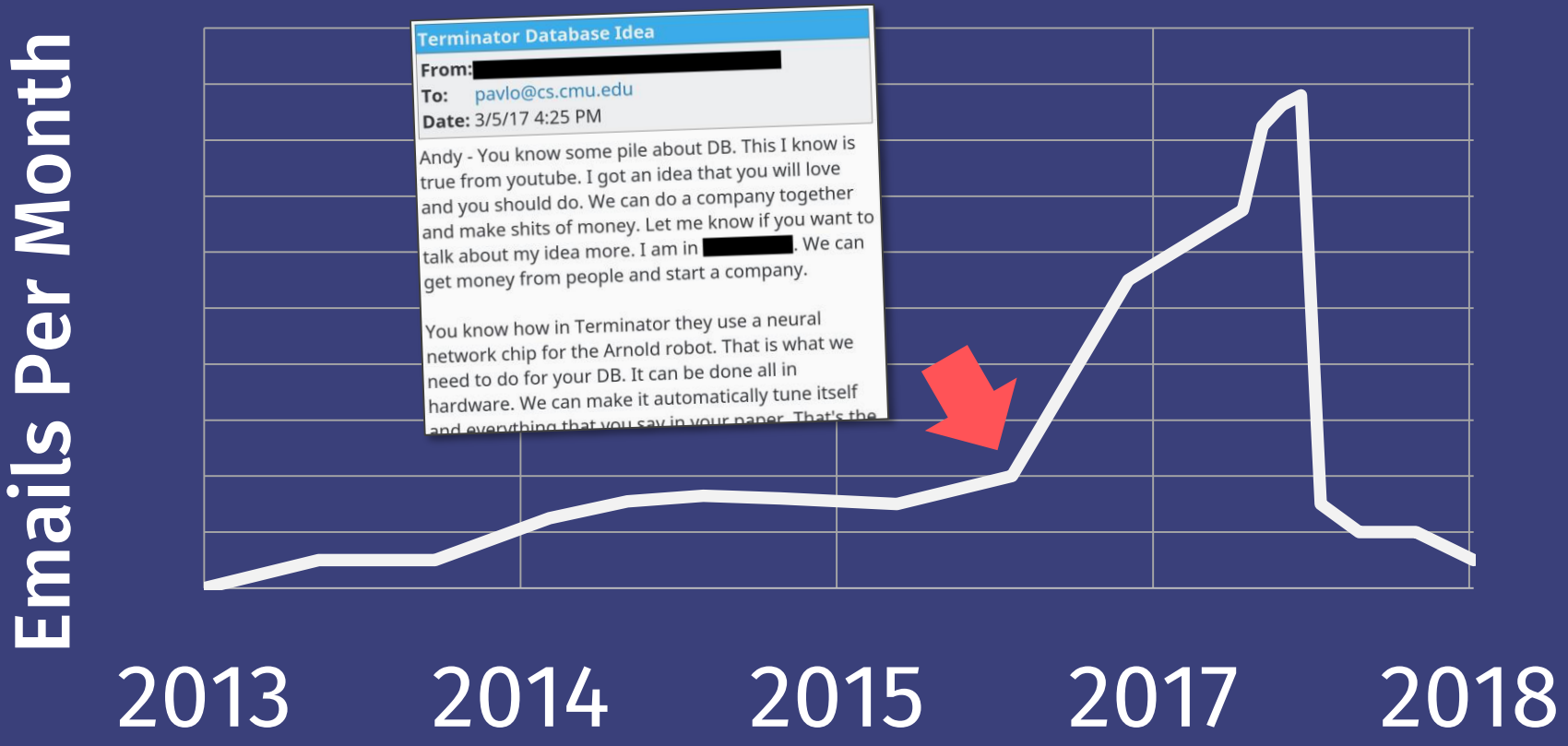
Date: 8/3/15 8:55 AM

Hi Andy,

I contact you to let you know, that I have developed an initial prototype of a database, that it could be of your interest.

The prototype is a Key/Value in-memory database, with the ability to perform data access in constant time. The data is grouped into DATASET of ordered data. Each INSERT automatically orders the new element and in Constant time. You you can also

Crazy Emails Received



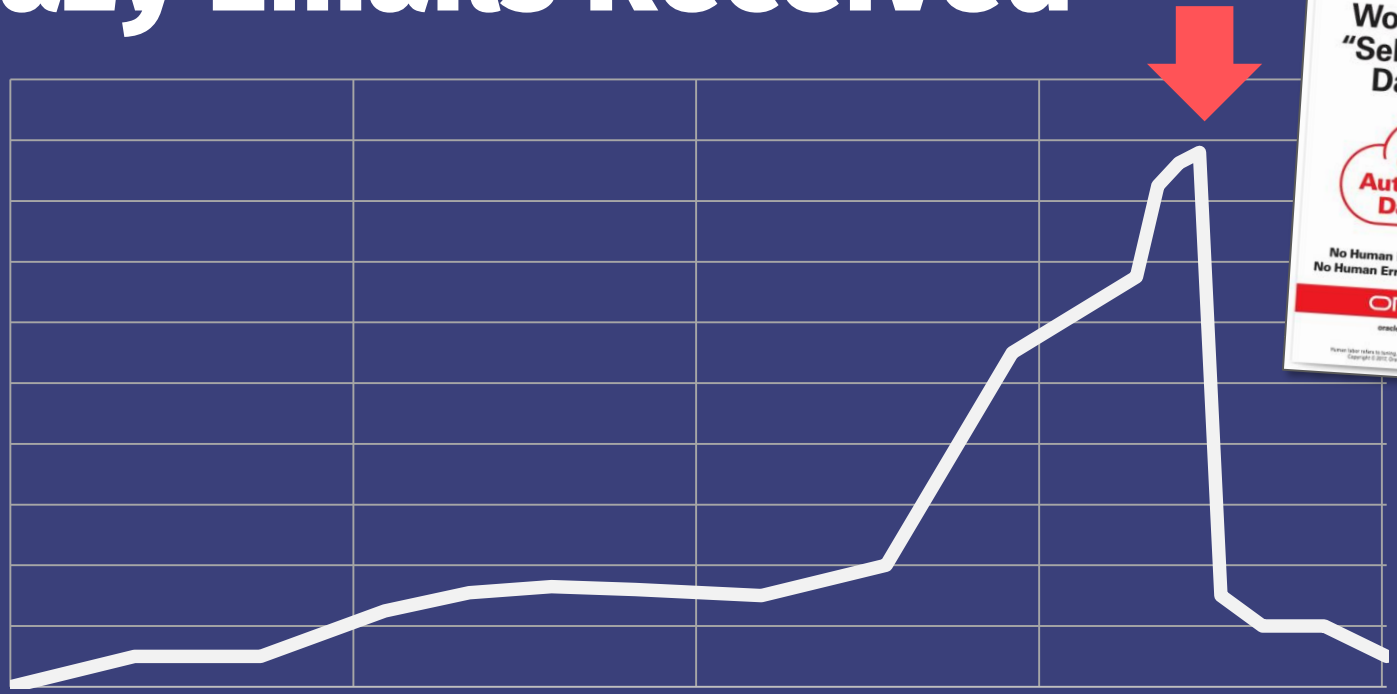
Terminator Database Idea

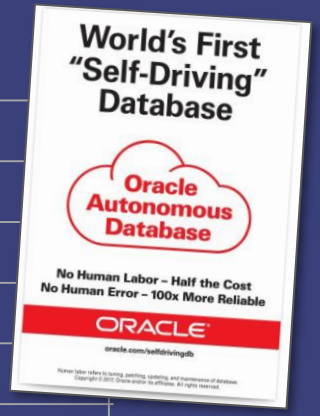
From: [redacted]
To: pavlo@cs.cmu.edu
Date: 3/5/17 4:25 PM

Andy - You know some pile about DB. This I know is true from youtube. I got an idea that you will love and you should do. We can do a company together and make shits of money. Let me know if you want to talk about my idea more. I am in [redacted]. We can get money from people and start a company.

You know how in Terminator they use a neural network chip for the Arnold robot. That is what we need to do for your DB. It can be done all in hardware. We can make it automatically tune itself and everything that you say in your paper. That's the

Crazy Emails Received





1970s: Self-Adaptive

1990s: Self-Tuning

2010s: Self-Driving

1970s: Self-A

1990s: Self-T

2010s: Self-

Self-Driving Database Management Systems

Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon
Todd C. Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic
Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu*, Ran Xian, Tieying Zhang
Carnegie Mellon University, *National University of Singapore

ABSTRACT

In the last two decades, both researchers and vendors have built advisory tools to assist database administrators (DBAs) in various aspects of system tuning and physical design. Most of this previous work, however, is incomplete because they still require humans to make the final decisions about any changes to the database and are reactionary measures that fix problems after they occur.

What is needed for a truly "self-driving" database management system (DBMS) is a new architecture that is designed for autonomous operation. This is different than earlier attempts because all aspects of the system are controlled by an integrated planning component that not only optimizes the system for the current workload, but also predicts future workload trends so that the system can prepare itself accordingly. With this, the DBMS can support all of the previous tuning techniques without requiring a human to determine the right way and proper time to deploy them. It also enables new optimizations that are important for modern high-performance DBMSs, but which are not possible today because the complexity of managing these systems has surpassed the abilities of human experts.

This paper presents the architecture of Peloton, the first self-driving DBMS. Peloton's autonomic capabilities are now possible due to algorithmic advancements in deep learning, as well as improvements in hardware and adaptive database architectures.

1. INTRODUCTION

The idea of using a DBMS to remove the burden of data management was one of the original selling points of the relational model and declarative query languages from the 1970s [31]. With this approach, a developer only writes a query that specifies what data they want to access. The DBMS then finds the most efficient way to store and retrieve data, and to safely interleave operations.

Over four decades later, DBMSs are now the critical part of every data-intensive application in all facets of society, business, and science. These systems are also more complicated now with a long and growing list of functionalities. But using existing automated tuning tools is an onerous task, as they require laborious preparation of workload samples, spare hardware to test proposed updates, and above all else intuition into the DBMS's internals. If the DBMS could do these things automatically, then it would remove many of the complications and costs involved with deploying a database [40].

8th Biennial Conference on Innovative Data Systems Research (CIDR '17) January 8-11, 2017 , Chaminade, California, USA.

Much of the previous work on self-tuning systems is focused on standalone tools that target only a single aspect of the database. For example, some tools are able to choose the best logical or physical design of a database [16], such as indexes [30, 17, 58], partitioning schemes [6, 44], data organization [7], or materialized views [5]. Other tools are able to select the tuning parameters for an application [56, 22]. Most of these tools operate in the same way: the DBA provides it with a sample database and workload trace that guides a search process to find an optimal or near-optimal configuration. All of the major DBMS vendors' tools, including Oracle [23, 38], Microsoft [16, 42], and IBM [55, 57], operate in this manner. There is a recent push for integrated components that support adaptive architectures [36], but these again only focus on solving one problem. Likewise, cloud-based systems employ dynamic resource allocation at the service-level [20], but do not tune individual databases.

All of these are insufficient for a completely autonomous system because they are (1) external to the DBMS, (2) reactionary, or (3) not able to take a holistic view that considers more than one problem at a time. That is, they observe the DBMS's behavior from outside of the system and advise the DBA on how to make corrections to fix only one aspect of the problem after it occurs. The tuning tools assume that the human operating them is knowledgeable enough to update the DBMS during a time window when it will have the least impact on applications. The database landscape, however, has changed significantly in the last decade and one cannot assume that a DBMS is deployed by an expert that understands the intricacies of database optimization. But even if these tools were automated such that they could deploy the optimizations on their own, existing DBMS architectures are not designed to support major changes without stressing the system further nor are they able to adapt in anticipation of future bottlenecks.

In this paper, we make the case that self-driving database systems are now achievable. We begin by discussing the key challenges with such a system. We then present the architecture of **Peloton** [1], the first DBMS that is designed for autonomous operation. We conclude with some initial results on using Peloton's integrated deep learning framework for workload forecasting and action deployment.

2. PROBLEM OVERVIEW

The first challenge in a self-driving DBMS is to understand an application's workload. The most basic level is to characterize queries as being for either an OLTP or OLAP application [26]. If the DBMS identifies which of these two workload classes the application belongs to, then it can make decisions about how to optimize the database. For example, if it is OLTP, then the DBMS should store tuples in a row-oriented layout that is optimized for writes. If it is OLAP, then the DBMS should use a column-oriented

Self-Driving DBMS

- What to change?
- When to change it?
- Was it helpful?

Today @ 11:30am

Room 203

@andy_pavlo