HDCAS: Hard Drive Condition Alert System

CS6365 Project proposal
Woradorn Kamolpornwijit
Spring 2020

Motivation

In today's world, much of our information is stored in computer storage instead of physical copies. For example, photos of significant life moments that used to get developed onto paper and filed into physical photo albums are now filed away as digital copies stored in a computer. Many other important items, such as copies of paperworks, are also treated the same way. This presents a great potential for damage and losses in the case of storage hardware failure. While most power users are aware of the risks and do have backups of their most important data, this leaves many of less technical-minded users at the mercy of their luck and expensive data recovery services when the time comes. Additionally, even those who backup their most important data are unlikely to have a full disk backup that they can just immediately swap into their computer in the case of drive failures. This presents a risk of downtime for all except the exceptionally well-prepared among us.

This project aims to prevent the disruption to life caused by disk failures by giving users advanced warning of impending disk failure. The additional warning period allows the user to transfer all the important data of their dying disk before the failure occurs. Additionally, the user is afforded time that they could use to prepare replacement disks and schedule their own downtime for maintenance, instead of having their life randomly disrupted by unexpected failure.

The operation of our project is based on select infrastructures and datasets available to the public. First, modern hard drives come with a monitoring system called SMART. This allows important drive health metrics to be read from the drives. However, most of the parameters available are either indications of an already failing drives (e.g. reallocated sector counts) or parameters with opaque link to the future failure of the drive (e.g. temperature). This leaves most monitoring software applying basic heuristics on such parameters. Our project will be different due to our second piece of knowledge: Backblaze's hard drive dataset. Using this dataset, our project will hopefully be able to make more accurate and useful predictions on drive failures, and give our users time to prepare for such occasions.

Related works:

Many software solutions, such as Speedfan (Comparetti, 2020) and CrystalDiskInfo (Crystal Dew World, 2020) are available for reading of SMART parameters and warning based on simple heuristics. Specifically, Speedfan does come with an online analysis using hddstatus.com (Hddstatus.com, 2020). However, based on analysis of their website, hddstatus.com seems to have been abandoned some time after 2014. Their data analysis also seems to have been based on simple statistics on collected data.

The Backblaze dataset for hard drive failure (Backblaze.com, 2020) have also been analyzed in many papers ((Shen et al., 2018), (Aussel et al., 2017), (Huang, 2017)). Many of such papers applied various statistical and machine learning methods to the dataset in the hope of predicting future hard drive failures. While they got impressive prediction performance, none of them had yet been deployed in the real world.

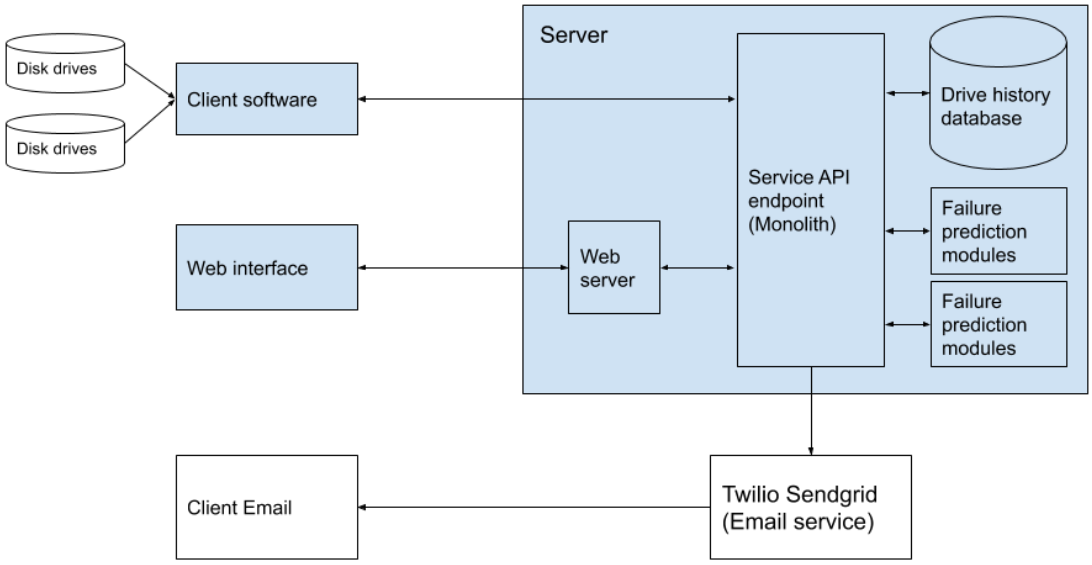
Features of Proposed work:

The goal of our project is to build a system which notifies the user of impending drive failure in advance. From the user's perspective, the system will consist of a client program to be run on the target machine with the drives installed, and a web interface to manage notifications and keep track of hard drives in their fleet.

The client program will run in the background and periodically send the SMART information from the hard drives in the system to the server. If the server predicts impending drive failure, the client software will notify the user. By evaluating the predictions on the server side, more advanced prediction algorithms can be deployed without concerns about leaking the potentially proprietary prediction algorithm out to the world.

The web interface allows the user to keep track of the health of all the hard drives in their fleet, potentially from multiple machines. This is also where email notification can be configured. The system will send out email notifications when a drive is detected to be failing. The system will also notify the user if a drive had not checked in with the server for a long period of time, urging the user to mark the drive as either failed or retired without failure.

System architecture:



The system consists of three main components: the client software, the server, and the web interface. The client software is a small Python program. It dumps SMART data from installed hard drives using `smartctl` (Smartmontools.org, 2020) and `pySmart` library (Herndon, 2020). The collected information for each drive is then sent to the server. The client then receives the reply back from the server containing drive failure predictions. If required, the client will then display a warning of impending drive failure to the user.

The server consists of four main parts: the web server, the drive history database, the monolithic backend service, and multiple failure prediction modules. The server as a whole will be programmed in Python unless otherwise required. The web server serves the web interface to the client and handles other associated tasks. This part of the server will be built using existing web server software to reduce development risks and complexity. The drive database history is a Postgresql database containing all data received from the client program. This database can then be used to train the failure prediction modules. The monolithic backend is a monolith which glues all the other components together. The monolith is also responsible for dealing with Twilio Sendgrid for notification emails. Finally, the failure prediction modules are self-contained Python modules which implements the failure prediction logic. Each module is responsible for their own training using data from the drive history database. Each module can be built with varying degrees of complexity, from simple heuristics to complex machine learning models. Since the complexity of each module is encapsulated, this simplifies the design of the monolithic backend server. All of these server components will be hosted on Amazon AWS for simplicity and reproducibility.

While a monolith is not the best design pattern in terms of scalability, I believe that the benefit in development simplicity and speed outweighs this concern. Past experience with other systems in actual production have shown that a monolithic design is good for quickly getting a product off the ground and finding a market fit. The monolith can then be broken up into microservices later in the lifecycle of the project, when product market fit have been confirmed and revenue streams have been established.

The web interface allows the user to quickly check the last updated status of all their hard drives. The interface also allows the user to configure the notification and mark retired drives as either failed or retired without issue. This provides feedback to the drive history database and, in turn, the failure predictor modules. The web interface will likely be constructed using React.js or other industry-standard Javascript frontend frameworks.

In terms of data source, the system will be initially populated with the drive health data from Backblaze (Backblaze.com, 2020). This dataset covers many models of large-capacity desktop-grade (non enterprise) hard drives. However, the dataset is restricted to drives deployed in a datacenter context. This may adversely impact the accuracy of the predictions. However, the system will be able to learn from hard drives it sees out in the world after its deployed, alleviating this accuracy concern.

Evaluation/Testing method:

The project will be evaluated in two major areas: performance and accuracy. For performance evaluation, the system response time and resource requirement will be characterized as a function of incoming request rate. For accuracy evaluation, a set of select drives will be picked from the dataset and evaluated by the service. The result will then be compared to the actual outcome recorded in the dataset. Other evaluations may also be added as appropriate.

Foreseen Risk/challenges:

- Accurate hard drive failure prediction is difficult, as shown by the numbers of research papers on the topic. This is also not helped by the very well-controlled but not real-world condition the Backblaze dataset was collected from.
- The scope of the project is quite large for a single-person project. This will be mitigated by a breadth-first development schedule, where a basic version of each and every part of the project are put into place before detailed development on one. This reduces the risk of the system missing a significant part at the conclusion of the project.

Deliverable:

- Project source code and deployment instruction
- Final report on project design and implementation
- Project presentation

Future works:

- One-click installer for client software
    - The current implementation will assume all dependencies (Python, smartctl) are already available on the client machine for simplicity
- Failure prediction module improvements
    - Automated module training
    - More advanced prediction
- Support for wider variety of client OS and environments
    - The basic system will be developed for Windows clients, due to development environment availability.
- Support for solid-state drives (SSD)
    - This is due to both lack of in-service data and proprietary, non-standard nature of SMART data reported by such drives

Work distribution:

This section is not applicable as this is a single-person project.

Workplan

Sprint	Date	content
1	Feb 17 - 28	● Get “hollow” (no business logic) client application/server/database/web interface running and talking to each other ● SMART data reading and parsing from client
2	Mar 2 - 13	● Figure out DB schema and implementation details ● Get Backblaze data into DB ● Implement server framework (with basic predictor module)
3	Mar 16 - 26	● Implement advanced predictor module ● Implement module training ● Implement email notification with Twilio Sendgrid
4	Mar 30 - Apr 9	● Performance benchmark ● Report and presentation writing ● Schedule slack for potential overruns

References:

- Comparetti, A. (2020). SpeedFan - *Access temperature sensor in your computer*. [online] Almico.com. Available at: http://www.almico.com/speedfan.php [Accessed 14 Feb. 2020].

- Crystal Dew World. (2020). *CrystalDiskInfo*. [online] Available at: https://crystalmark.info/en/software/crystaldiskinfo/ [Accessed 14 Feb. 2020].

- Hddstatus.com. (2020). *HDDStatus - Tool to check hard disk status, health and reliability.* [online] Available at: http://www.hddstatus.com/ [Accessed 14 Feb. 2020].

- Backblaze.com. (2020). *Backblaze Hard Drive Stats.* [online] Available at: https://www.backblaze.com/b2/hard-drive-test-data.html [Accessed 14 Feb. 2020].

- Shen, J., Wan, J., Lim, S. and Yu, L. (2018). *Random-forest-based failure prediction for hard disk drives. International Journal of Distributed Sensor Networks*, [online] 14(11), p.155014771880648. Available at: https://journals.sagepub.com/doi/full/10.1177/1550147718806480 [Accessed 14 Feb. 2020].

- Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E. and Chabridon, S. (2017). *Predictive Models of Hard Drive Failures Based on Operational Data. 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [online] Available at: https://hal.archives-ouvertes.fr/hal-01703140/document [Accessed 14 Feb. 2020].

- Huang, X. (2017). *Hard Drive Failure Prediction for Large Scale Storage System. Master of Science in Statistics.* University of California Los Angeles. Available at: https://escholarship.org/uc/item/11x380ng [Accessed 14 Feb. 2020].

- Smartmontools.org. (2020). *smartmontools*. [online] Available at: https://www.smartmontools.org/ [Accessed 14 Feb. 2020].

- Herndon, M. (2020). *pySMART.smartx*. [online] PyPI. Available at: https://pypi.org/project/pySMART.smartx/ [Accessed 14 Feb. 2020].