JOIN THE FUN!

• Check the Chat window in Zoom (More options)

• Register to our HPE DEV Slack
  • https://slack.hpedev.io/

• Join the #munch-and-learn channel and start chatting
  • https://hpedev.slack.com/archives/C01GVQUPM3P
  • Upload pictures of your munchies

• Join #ezmeral-data-fabric channel and ask technical question to the community (including Ted & Ellen)
  • https://hpedev.slack.com/archives/CU3JRBTB7
WHAT’S A DATA FABRIC, ANYWAY?

Ted Dunning, Data Fabric CTO, HPE
@ted_dunning
ted.dunning@hpe.com
A DATA FABRIC IS
A DATA FABRIC DOES
RUN THE RIGHT APPLICATIONS
AT THE RIGHT TIME
IN THE RIGHT PLACE
AGAINST THE RIGHT DATA
Your business doesn’t actually happen in a data center
You can’t ignore geography
The value in your business is where your customers are
Let’s talk about some specific examples
How does streaming video actually work?
How does streaming video **actually work**?
Regional content server responds to content requests
Regional content server responds to content requests

How do we know if it is working?
Collect telemetry in a message stream
...and replicate to galactic HQ
The important point is that **one data structure** can span the enterprise.
The important point is that **one data structure** can span the enterprise.

The replication is at a fabric level.
The important point is that **one data structure** can span the enterprise.

The replication is at a fabric level.

...the source doesn’t know.
The important point is that **one data structure** can span the enterprise. The replication is at a fabric level. **...and analytical apps get a global view...** The source doesn’t know.
This idea isn’t limited to just media streaming
This idea isn’t limited to just media streaming.

The telemetry can come from any kind of observation.
The rollout of 5G telecom networks benefits the same way
The rollout of 5G telecom networks benefits the same way

Manufacturing systems can do the same
The rollout of **5G**

*telecom networks* benefits the same way

**Manufacturing systems** can do the same

...as can **medical devices**
Developing autonomous vehicles requires vast amounts of real-world data.
Data that can only be collected by **real cars** on **real roads**
A **data fabric** makes collecting this kind of data from field stations much easier.
But the volumes are prodigious:
- Gigabytes/s per car
- Petabytes of new data per day
- Hundreds of petabytes retained

Having a **single fabric** that can span from edge to core makes these projects **practical**
What happens behind the scenes when you pay for coffee in a café?
The point of sale system talks to a regional data center
The decision to honor the transaction is made and data is forwarded
Ultimately backup centers get the data as well
A **data scientist** builds models that will make future decisions.
In fact, there may be thousands of data scientists all building models
Working on **shared data** allows faster model learning

Multitenancy becomes a key requirement
Transparent data motion

Scale and speed

Sharing safely
RUN THE RIGHT APPLICATIONS AT THE RIGHT TIME IN THE RIGHT PLACE AGAINST THE RIGHT DATA
Let’s dig in and see some details
Over decades of progress, Unix-based systems have set the standard for compatibility and functionality.
Hadoop achieves much higher scalability by trading away essentially all of this compatibility.
MapR repaired this by restoring the compatibility while increasing scalability and performance.
Adding tables and streams enhances the functionality of the base file system.
And now, this technology forms the basis of the HPE Data Fabric
New tools for a new world of data
NEW BOOK COMING

The tradeoffs people assume are unavoidable, in fact, are not.

A unified data infrastructure can free you from these tradeoffs.
A RECENT EXAMPLE

How you can build these systems
Copy

NFS Appliance

File

Copy

HDFS

Copy

S3

Copy

Cold object store

NFS Appliance

File

Copy

HDFS

Copy

S3

Copy

Cold object store
Cold data

HIVE

Spark Apps

Working data

Ingestion

Landing

Extraction

HIVE
Ingestion

Landing

Masking

Extraction

Spark Apps

Working data

Cold data

HIVE

HPE Ezmeral Data Fabric
DEVOPS VIEW

Ingestion → Masking → Extraction

- Reporting
- Data science
Ingestion → Masking → Extraction → Reporting

landing → masked → work

Data science
INFRASTRUCTURE TEAM VIEW

Those guys

Data Fabric
90% of the effort in successful machine learning isn’t the learning ... 

It’s the logistics
old data → training → model → decisions or actions

more data! → ML Code
The fashionable part

ML Code
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Static Analysis of Data Dependencies. In traditional code, compilers and build systems perform static analysis of dependency graphs. Tools for static analysis of data dependencies are far less common, but are essential for error checking, tracking down consumers, and enforcing migration and updates. One such tool is the automated feature management system described in [12], which enables data sources and features to be annotated. Automating checks such as ensuring that all dependencies have the appropriate annotations, and dependency trees can be fully resolved. This kind of tooling can make migration and deletion much safer in practice.

4 Feedback Loops

One of the key features of live ML systems is that they often influence the behavior of the environment if they update over time. This leads to a form of analysis debt, in which it is difficult to predict the behavior of a given model before it is released. These feedback loops can take different forms, but they are all more difficult to detect and address if they occur gradually over time, as may be the case when models are updated infrequently.

Direct Feedback Loops. A model may directly influence the selection of future training data. It is common practice to use standard supervised algorithms, although the theoretically correct solution would be to use bandit algorithms. The problem here is that bandit algorithms (such as contextual bandits [9]) do not necessarily scale well to the size of action spaces typically required for real-world problems. It is possible to mitigate these effects by using some amount of randomization [3], or by isolating certain parts of data from being influenced by a given model.

Hidden Feedback Loops. Direct feedback loops are costly to analyze, but at least the statistical challenge that ML researchers may find natural to investigate [3]. A more difficult case is hidden feedback loops, in which two systems influence each other indirectly through the world. One example of this may be if two systems independently determine facets of a web page, such as one selecting products to show and another selecting related reviews. Improving one system may lead to changes in behavior in the other, as users begin clicking more or less on the other components in reaction to the changes. Note that these hidden loops may exist between completely disjoint systems. Consider the case of two stock-market prediction models from two different investment companies. Improvements (or, more scarily, bugs) in one may influence the bidding and buying behavior of the other.

5 ML-System Anti-Patterns

It may be surprising to the academic community to know that only a tiny fraction of the code in many ML systems is actually devoted to learning or prediction—see Figure 1. In the language of Lin and Ryaboy, much of the remainder may be described as “plumbing” [11].

It is unfortunately common for systems that incorporate machine learning methods to end up with high-debt design patterns. In this section, we examine several system-design anti-patterns [4] that can surface in machine learning systems and which should be avoided or refactored where possible.
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Static Analysis of Data Dependencies. In traditional code, compilers and build systems perform static analysis of dependency graphs. Tools for static analysis of data dependencies are far less common, but are essential for error checking, tracking down consumers, and enforcing migration and updates. One such tool is the automated feature management system described in [12], which enables data sources and features to be annotated. Automating checks ensures that all dependencies have the appropriate annotations, and dependency trees can be fully resolved. This kind of tooling can make migration and deletion much safer in practice.

4 Feedback Loops

One of the key features of live ML systems is that they often influence their own behavior if they update over time. This leads to a form of analysis debt, in which it is difficult to predict the behavior of a given model before it is released. These feedback loops can take different forms, but they are all more difficult to detect and address if they occur gradually over time, as may be the case when models are updated infrequently.

Direct Feedback Loops. A model may directly influence the selection of future training data. It is common practice to use standard supervised algorithms, although the theoretically correct solution would be to use bandit algorithms. The problem here is that bandit algorithms (such as contextual bandits [9]) do not necessarily scale well to the size of action spaces typically required for real-world problems. It is possible to mitigate these effects by using some amount of randomization [3], or by isolating certain parts of data from being influenced by a given model.

Hidden Feedback Loops. Direct feedback loops are costly to analyze, but at least the statistical challenge that ML researchers may find natural to investigate [3]. A more difficult case is hidden feedback loops, in which two systems influence each other indirectly through the world. One example of this may be if two systems independently determine facets of a web page, such as one selecting products to show and another selecting related reviews. Improving one system may lead to changes in behavior in the other, as users begin clicking more or less on the other components in reaction to the changes. Note that these hidden loops may exist between completely disjoint systems. Consider the case of two stock-market prediction models from two different investment companies. Improvements (or, more scarily, bugs) in one may influence the bidding and buying behavior of the other.

5 ML-System Anti-Patterns

It may be surprising to the academic community to know that only a tiny fraction of the code in many ML systems is actually devoted to learning or prediction—see Figure 1. In the language of Lin and Ryaboy, much of the remainder may be described as "plumbing" [11]. It is unfortunately common for systems that incorporate machine learning methods to end up with high-debt design patterns. In this section, we examine several system-design anti-patterns [4] that can surface in machine learning systems and which should be avoided or refactored where possible.
Hidden Technical Debt in Machine Learning Systems

So why focus on the boring bits?
So why focus on the boring bits?

Boring done 10% SLOWER

Your total time budget

Your total smartness budget
So why focus on the boring bits?
So why focus on the boring bits?
Because they will eat the clever bits!
So why focus on the boring bits?

Because they will eat the clever bits!

Also, systems aren’t ready for business until they are properly boring.
Objects span multiple servers

File system

Application

MFS

MFS

MFS
DESIGN ELEMENT - SCALE
SCALING DOWN

Single rack configuration is self-contained insofar as data fabric is concerned.
SCALING DOWN

Multi rack configuration can amortize minimum # nodes
SCALING UP

At very large scale, dense units become preferred

This is a rough rendition of a 24 PB system

Some customers have >500PB systems
DESIGN ELEMENT – GEO-DISTRIBUTION
Volumes look like directories to programs and most users.

But they give us management powers of how data is stored.
Volumes form trees just like directories.
And we can have shadows of volumes that have different properties
Replicated
This lets us optimize how we store different data and nearly erase performance / cost tradeoffs in many cases.
HOW FILES AND TABLES WORK
HOW FILES ARE STORED
HOW FILES ARE STORED

Original   Chunks
HOW FILES ARE STORED

Original

Chunks

8kB blocks
(grouped by 8)
HOW FILES ARE STORED

Original

Chunks

8kB blocks (grouped by 8)

Containers
Original chunks are divided into 8kB blocks (grouped by 8), which are then stored in containers.
FABRIC CONTAINERS

Each container has a replication chain
Updates are transactional
Failures are handled by rearranging replication
**BUT WAIT, THERE’S MORE**

- Directories and files are implemented in terms of B-trees
  - Key is offset, value is data blob
  - Internal transactional semantics guarantees safety and consistency
  - Layout algorithms give very high layout linearization

- Tables are implemented in terms of B-trees
  - Twisted B-tree implementation allows virtues of log-structured merge tree without the compaction delays
  - Tablet splitting without pausing, integration with file system transactions

- Common security and permissions scheme
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Similar to LSM implementations, tables are decomposed by key ranges.
Similar to LSM implementations, tables are decomposed by key ranges.

Distinct from HBase and Level DB, MapR tables used fixed number (greater than 1) of decompositions.

Very unusually, relative to LSM and cousins, data structures at the leaf are mutable.
RE-USE OF PROVEN TECHNOLOGY

Partitions are distributed just like file chunks

Same replication and transaction technology
HOW IT LOOKS
EXAMPLE

```
[tdunning@se-node10 ~]$ ls -F
apache-kylin-0.7.2-incubating-src-source-release.tar.gz
apache-kylin-0.7.2-incubating-src-source-release.tar.gz.d5
bar/
build.xml
car-data.csv
coco.parquet/
counts.parquet/
deep.json
drillbit.log
drillbit.log
edges.ssv
drillbit.log
edges.ssv
edges.ssv
edges.ssv
```

```
[s1]
s2@
schema.json
sf-city-lots-json/
side-log
car-data.csv
```

```
[s3]
t1@
tags.json
time_to_60.view.drill*
```

```
[tdunning@se-node10 ~]$ pwd
/mapr/se1/user/tdunning
```

Clusters

Directories

Files

Streams

Table

Volume mount point
Cluster

Volume mount point

```
[tdunning@se-node10 ~]$ pwd
/mapr/se1/user/tdunning
```

```
THANK YOU

ted.dunning@hpe.com
CONFIDENTIAL DISCLOSURE AGREEMENT

- The information in this presentation is authorized for general distribution but it should not be modified
- Do not remove any classification labels, warnings or disclaimers on any slide or modify this presentation to change the classification level
- Do not remove this slide from the presentation
- HPE does not warrant or represent that it will introduce any product to which the information relates
- The information contained herein is subject to change without notice
- HPE makes no warranties regarding the accuracy of this information
- The only warranties for HPE products and services are set forth in the express warranty statements accompanying such products and services
- Nothing herein should be construed as constituting an additional warranty
- HPE shall not be liable for technical or editorial errors or omissions contained herein
- Strict adherence to the HPE Standards of Business Conduct regarding this classification level is critical
THANK YOU FOR BEING PART OF THE HPE DEV COMMUNITY

• Monthly Newsletter: https://developer.hpe.com/newsletter-signup

• Developer Survey: https://www.developereconomics.net/?member_id=hpe
  • Make your voice heard
  • Closes on February 8th

• Next Munch & Learn session
  • February 24th 5PM CET
  • The benefits of a Container Platform and Machine Learning (ML) Ops
  • Speaker: Tom Phelan, HPE Fellow/CPO Ezmeral Container Platform
  • Industry Host: Nigel Poulton
  • Registration link: https://hpe.zoom.us/meeting/register/tJYkdequqTwsE9LuPAgPDbV-mf1V7jq23Mxj