Dealing With Device Data

Strata London
May, 2017

Mark Madsen
www.ThirdNature.net
@markmadsen
Leapfrog

1. Pick a platform

2. Add 2-4 games, books or maps
The business initiative

• Need new products to compete with other vendors doing similar things
• Internet had become ubiquitous, connectivity is easier so it is possible to see what kids are doing with games
• Can apply more education and psychology-of-learning theory if have the ability to see how well kids are doing
• Engage parents in what the kids are doing
• New products that are better at helping kids learn
• Prompt more use via direct engagement, monitoring of use / lack of use
• New products for broader audience
• More detailed customer knowledge
Connect to your child’s learning!

With the LeapFrog Learning Path, you can follow your child’s developing skills, see areas where he/she excels or needs support and share achievements with friends and family.

Already have a Learning Path account? Just sign in.

SIGN IN

Don’t have an account? Connect your LeapFrog device to get started!

CONNECT
Some of the pertinent specs

Engineering goal: data products that phone home with usage data about game play.

Core architecture constraints:

▪ Concurrency (user scale)
  • 200,000 devices sending log data (at the same time)

▪ Near real-time latency
  • Data must be processed and available to users within minutes of upload

▪ Data volume (data scale)
  • Billions of events

▪ Widely diverging uses for the data
  • External users on web and devices
  • Internal users: monitoring, reporting, analytics

▪ 24 x 7 x 365 availability with seasonal spikes
A note on the tech landscape

Technology choices were much more limited in 2007 for the needs we had:

- “Streaming data” was things like JMS, ActiveMQ, etc.
- Cloud: Amazon EC2 was still in beta
- Hadoop a year from release to Apache
- But we did have S3
What this translates to in engineering:

• Devices, firmware and game software
• Collection infrastructure, 24x7x365, with 10x spikes
• Platform to store and manage the collected play data
• Web site for gameplay and in-product rewards / tracking
• Web site for ecommerce, game-related purchases
• Data collection and management for ecom, online mktg, A/B
• Platform for CRM, usable data for business operations
• Rec engine to support online and email marketing
• Email marketing system, list management, segmentation, list pulls, response modeling
• Data warehouse for business operations, reports, dashboards
• Data for ad-hoc analysis / analytics
• Lots of glue: integration to the core enterprise systems, ERP
What was built: core business and marketing ops

• Analysis
• Management reporting
• Customer segmentation for direct marketing
• Recommendation engine development & support
What was built: management and delivery

• All the processes to support game development
• Device and game data management
• Sales, ecommerce processing
• Digital product distribution
What was built: systems and databases that support marketing, management reporting, email marketing

- Customer demographics
- Online marketing
- Email marketing
- Customer sales & product ownership

© Third Nature Inc.
We’re so focused on the light switch that we’re not talking about the light

This case is not really about the code or architecture, but about the data part that is the core of data products
NOISE AND DATA STREAMS
There is a lot more noise in event data than in TXs

Transactions: legal reasons it has to be perfect, databases ensure it
Events: we can’t promise anything
Problems you see in the data: duplicate / incomplete / corrupted logs, duplicate events, missing events, out of order events, constraint violations, missing keys / data, impossible data.

e.g. One owner with thousands of devices.

Increasing number of logs from device serial # 0.

Caused by a glitch when replacing batteries.

The data is very hard to fix, the source is unfixable.
The traditional data lifecycle has a one-way flow

Transactions
MDM

Data entry → Store → Extract → Cleanse → Load → Use

This runs at human speed

It assumes humans enter data into forms of some sort

Those get written to a transactional DB, and extracted from there for analytic uses

Bad data? Validate data on the input form that caused it. Then fix the data in the database. Problem solved.
The machine data lifecycle is more complex

People write code that runs somewhere else and makes data.
This runs at machine speed, generating data much faster.
Bad data? You can’t validate at the point of creation because there’s no human to re-enter it.
Find the bug and fix the program. Hopefully it’s not in silicon.
Data quality assurance has to be reactive and proactive.
You can never completely fix even known problems

What happens to data when you release patches?
Now you have both good data and bad data coming in. For every firmware, OS, software version. It will be this way forever.
You want flexible persistence for data collection
The missing ingredient from most event data

Specifically, metadata kept separate from the data.
An event contains mainly IDs...that reference other data

Log de-referencing and enrichment is difficult since you can’t enforce integrity like you can in a DB.

What’s the glue that holds it together?
It’s just keys to other data.

e.g. remember that device ID 0 problem?
It’s not just the log data, it’s big data + small data

A product has a complex set of master data and metadata and this needs to be added back to the bare events.
Where does the reference data come from?

The keys come from somewhere. You don’t just make up system-wide unique identifiers in your code.

The lack of local lookup data at event generation leads to development practices that lead to inconsistencies.

Problem we had: product identifiers that didn’t match any known products

Had to fix by analyzing each log of bad-ID events.

Because developers used config files they copied from the PMS and put in the code.
Data management is part of software development

MDM is like version control for data. The way you keep it fresh
Canonical forms belong in the code as well as the repository

Data is a reflection of code. Messy events = messy code.

When there is common data, it should be in a common form.

There is no reason for developers to use custom data structures for common objects.

Downstream schema is the wrong place to enforce structure. It belongs in the code, upstream.
COORDINATION
What you think you have
What you really have

This is called “self-service.” Analysts will love it...
At the other end you hope to reassemble it
MDM again
If you want to link datasets then you must manage the keys
You need canonical forms for common data (in code too)

Event

2010.01.10 01:41:4468 67.189.110.179 Ser40489a07-7f6e4251801a13ae51a6d092 301212631165031 590387153892659 DNIS5555685981-UTF8&1T4ACGW_enUS386US387&fu=0&ifi=1&dtd=204&xpc=1KoLqh374s m100109_44IOJ1-Id=105543CD1A7B8322

Click

2010.01.10 14:26:2468 67.189.110.179 10098213 5046876319474403 MOZILLA/4.0 (COMPATIBLE; TRIDENT/4.0; GTB6; .NET CLR 1.1.4322) https://www.amazon.com/gifts/store/LogonForm?mmc=link-src-email_m100109 http://www.google.com/search?sourceid=navclient&aq=0h&oq=Italian&ie=UTF8&pid=1T4ACGW_13ae51a6d092&q=italian+rose&fu=0&ifi=1&dtd=204&xpc=1KoLqh374s

Cust-user

UID 590387153892659 CID 10098213 Email barry.dylan@odin.com City Paris State Île-de-France Country France
Basic architecture for processing the log data

1. Decompress & process device logs
2. Data collection & web operations
3. Monitoring, reporting, analytics data feeds

Log archive
Where did data flows go wrong?

Decompress & process device logs

Data collection & web operations

Monitoring, reporting, analytics data feeds
Where did data flows go wrong?

1. **Corruption.** Bit of a problem in its ability to shut down the data flow completely. And its ability to crash the collection grid.

2. **Insert failures.** Little problem with making changes at human speed while all else is machine speed, leading to data loss.

Root causes: development coordination with data management, architectural decisions resulting in dependencies, technology choice
Streams vs Transactions

Spikes? Failures? What’s backpressure?

Every architecture has points in it where things can back up and be lost. Many real world scenarios do not allow one to throttle indefinitely.

A traditional RDBMS does not autoscale, so you must size for peak: **DBAs love their Oracle RAC$**
Using relational databases for streaming data collection...

With Oracle: there’s a vision, and there’s reality
Changes to avoid these problems

1. Network error logging via distributed queues, decouple parse and load operations into independent steps

2. Self-describing log schema, field-column maps to maintain backward compatibility and not lose entire records, distributed filesystem in place for DB failures, dynamic table creation considered to accommodate recording new data (abandoned)
Decoupling is the key in the data architecture.

The biggest decision was to separate all data collection from the rest of data management. Trying to do everything in one database is not feasible, nor is using a database.
Why did it go wrong? Disjoint development processes

The log generation and parsing passes QA at every step: device, OS, game, infrastructure.
It fails when users see the data, e.g. the clock bug
Root causes of many data problems are organizational

For example, the cheap clock problem:
Device sends impossible event sequences
Traced to bad timestamps
By analysts downstream
Not the post-processing
Not the parsing
Not the game
Not the OS
It’s the cheap clocks on the boards
-.10/unit tradeoff upstream = +.45/unit downstream
End-to-end process: more than “collect events from IoT”

Total of 8 different development areas

3 major system divisions: the product data & metadata infrastructure to enable repeatable game and data product development, the data product work (collection, processing, delivery), the games and the online game support,
The fix for the root cause is to organize end to end.

Any data product is an end-to-end system. The data and flow, not code, is the core of the architecture. Most firms organize improperly.
CONCLUSION
Data collection today, faster but not better
On optimism and over-engineering

Forecasted sales drove size of deployment.

Architecture used some scale-out components and some scale-up components: this can be a bad model because you size to peak.

Nobody challenged the estimates. This happens in business all the time

Lessons: never trust business estimates, architect (not implementer) must decide
The infrastructure for a data product and the data product are basically the same thing (most of the time)
Data product lesson

Transaction processing is not the same thing as analytics: different architectures, technologies, methods required.

Big data market forgot this and is back on the “one platform” bandwagon.

Data products are more like BI and data warehouses than OLTP: end-to-end data flow systems that reassemble data for use.
Data management is where developers are weakest.
Modern engineering practices are where data management is weakest.
You need to bridge these groups and practices in the organization if you want to do meaningful work with event stream data.
CC Image Attributions

Thanks to the people who supplied the creative commons licensed images used in this presentation:

fast kids truck peru.jpg - http://flickr.com/photos/zerega/1029076197/
CC Image Attributions

Thanks to the people who supplied the creative commons licensed images used in this presentation:

- Building demolition - https://www.flickr.com/photos/gregpc/4429888820
About the Presenter

Mark Madsen is president of Third Nature, a technology research and consulting firm focused on business intelligence, data integration and data management. Mark is an award-winning author, architect and CTO whose work has been featured in numerous industry publications. Over the past ten years Mark received awards for his work from the American Productivity & Quality Center, TDWI, and the Smithsonian Institute. He is an international speaker, a contributor to Forbes Online and on the O’Reilly Strata program committee. For more information or to contact Mark, follow @markmadsen on Twitter or visit http://ThirdNature.net
About Third Nature

Third Nature is a research and consulting firm focused on new and emerging technology and practices in analytics, business intelligence, information strategy and data management. If your question is related to data, analytics, information strategy and technology infrastructure then you’re at the right place.

Our goal is to help organizations solve problems using data. We offer education, consulting and research services to support business and IT organizations as well as technology vendors.

We fill the gap between what the industry analyst firms cover and what IT needs. We specialize in product and technology analysis, so we look at emerging technologies and markets, evaluating technology and how it is applied rather than vendor market positions.