Conviva Unified Framework (CUF) for Real Time, Near Real Time and Offline Analysis of Video Streaming With Spark and Databricks

Jibin Zhan

CONVIVA®

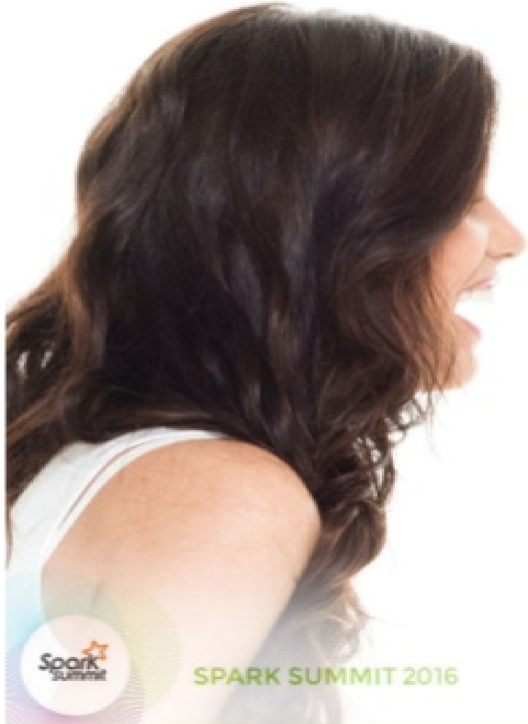
SPARK SUMMIT 2016

DATA SCIENCE AND ENGINEERING AT SCALE

JUNE 6-8, 2016 SAN FRANCISCO

CONVIVA®

is a video experience management platform that maximizes viewer engagement



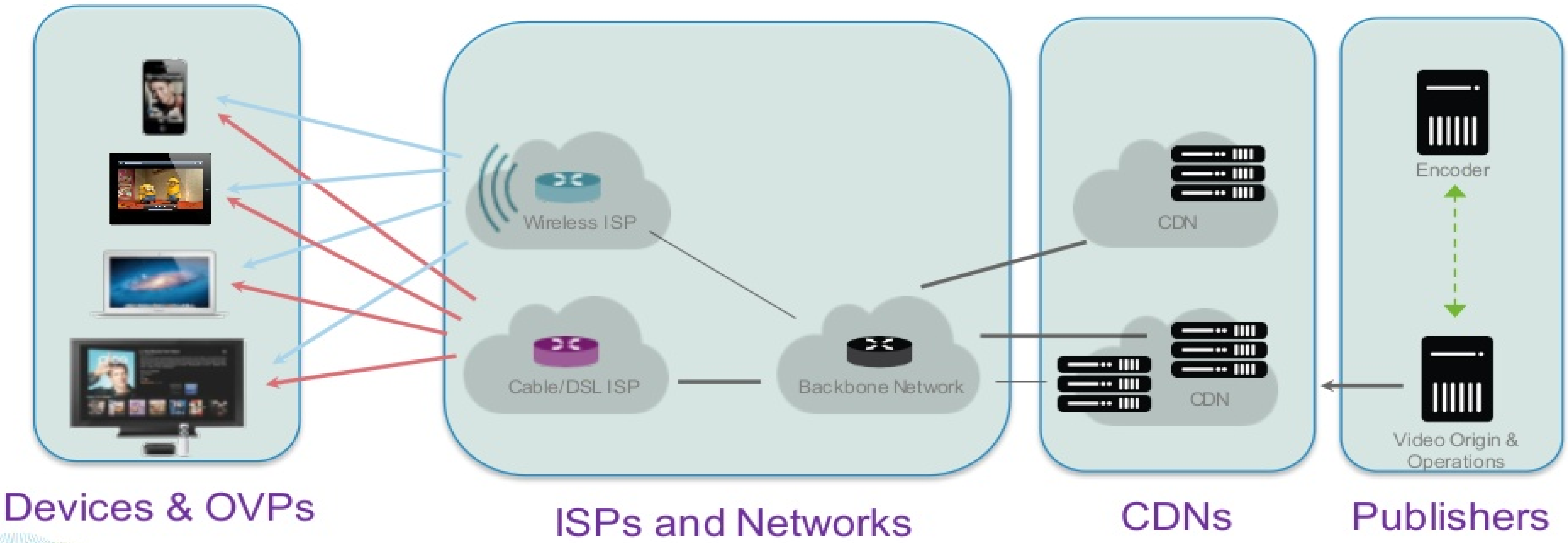
Unleashing the Power of OTT

- Video streaming over the internet (OTT) is growing rapidly
- Major industry shifts in the last couple of years
  - HBO Now
  - ESPN/SlingTV
  - Verizon Go90
  - Facebook, Twitter
  - Amazon Prime Video

Online Video – A Hugely Important Application
“Big Bang” Moment is Unfolding – Right Now

Internet Video Streaming is Hard

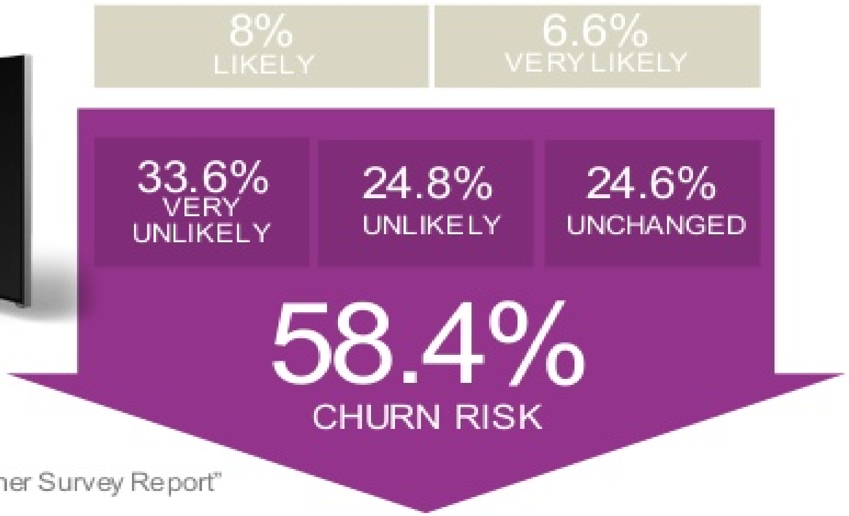
Many parties, many paths but no E2E owner



QoE is Critical to Engagement

For both – Video and Advertisement business

HOW LIKELY ARE YOU TO WATCH
FROM THAT SAME PROVIDER AGAIN?



Source: Conviva, "2015 Consumer Survey Report"

Viewers are expecting TV like quality and better

Experience Matters!!

Must solve for EXPERIENCE and ENGAGEMENT

Success is more than just great content…
Experience impacts engagement

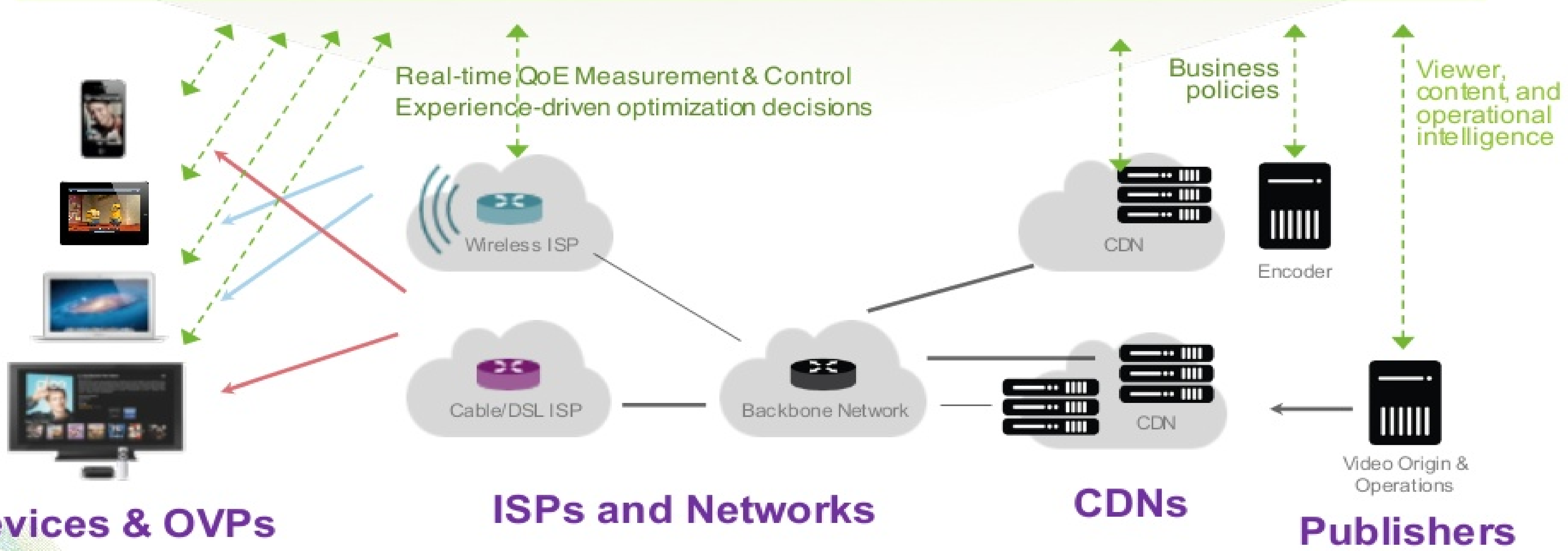
Competition for eyeballs increasing…
Internet of Content > Traditional TV viewing

TV revenues are up for grabs…
Internet offers SVOD, AVOD, PPV & "Unbundled choices"

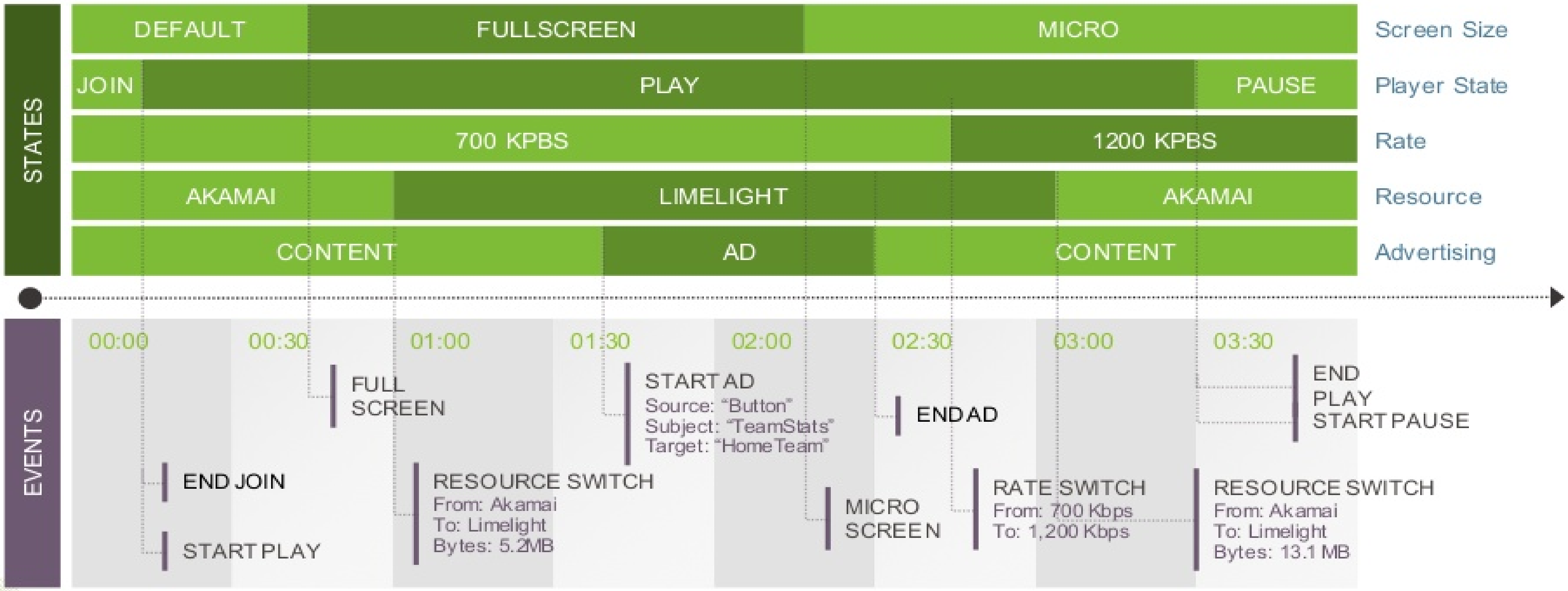
OR ELSE All bets are off!

Publishers and Service Providers cannot lose touch with viewers' experience

CONVIVA EXPERIENCE MANAGEMENT PLATFORM



Granular Dataset Indexed by Rich Metadata



Scale of Deployment

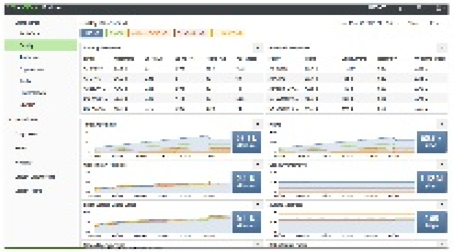
AVOD

SVOD

Infra

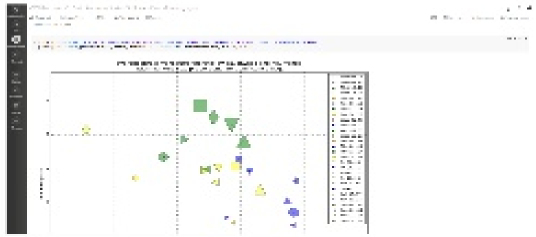
Scale of Deployment

Use Cases requiring 3 Stacks



- Real time metrics
- Real time alerts
- Real time optimization

- Near real time metrics
- Historical trending
- Benchmarking

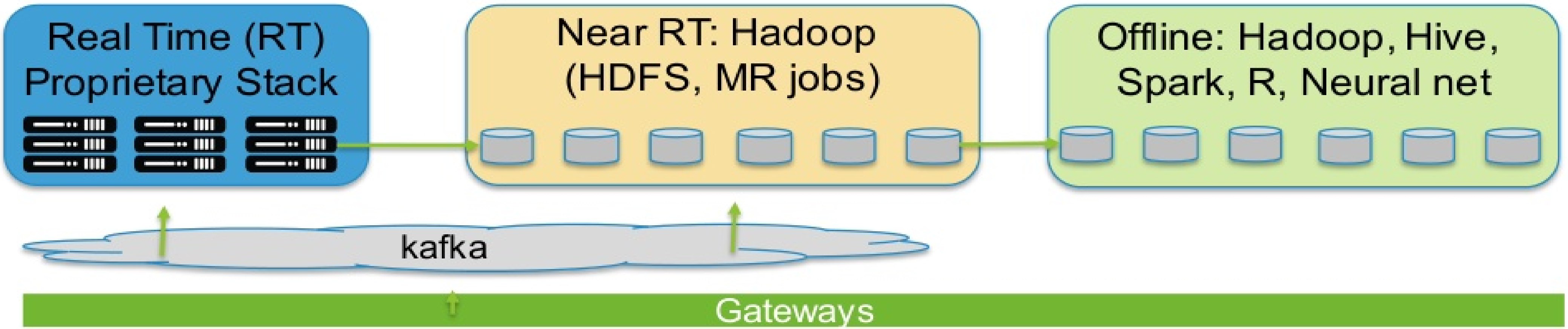


- In depth & ad hoc analysis
- Data exploration
- ML model training

Live (RT) (speed) ⟷ Historical (near RT) (batch) ⟷ Offline (batch)

Gateways

Old Architecture

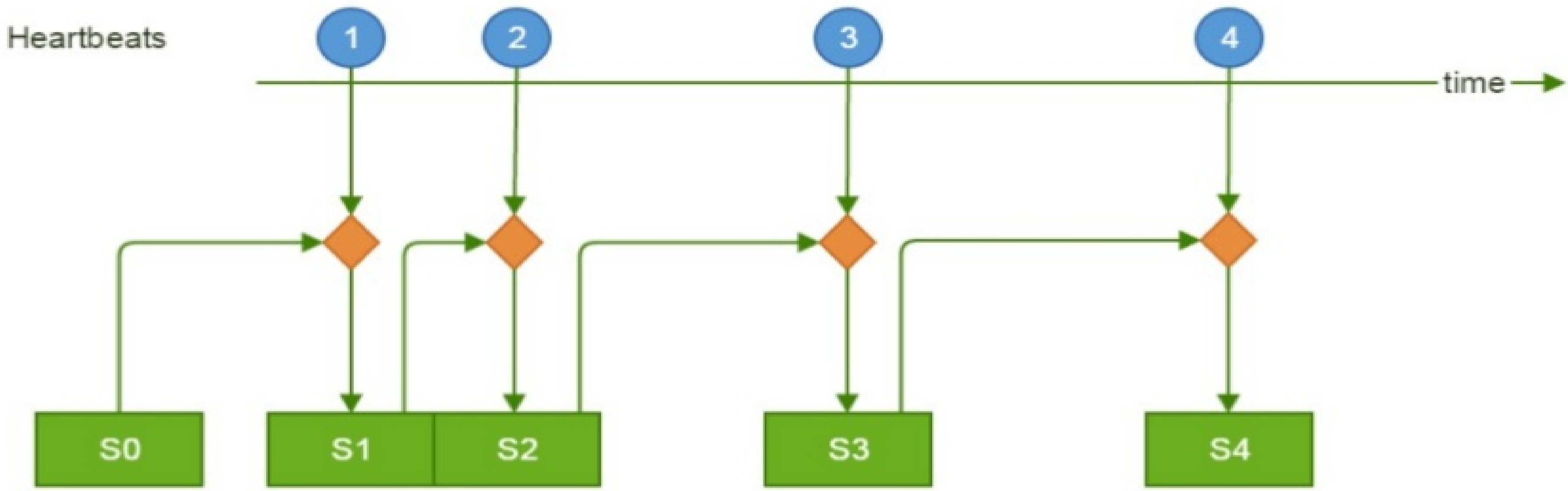


- RT and near RT stacks get input from Kafka independently
- RT and near RT run independently (except some RT results saved to HDFS for some near RT calculation)
- Offline gets data from near RT Hadoop, with additional calculation specifics to offline analysis.
- Hive/Spark/R/NeuralNet etc. are used for various offline tasks

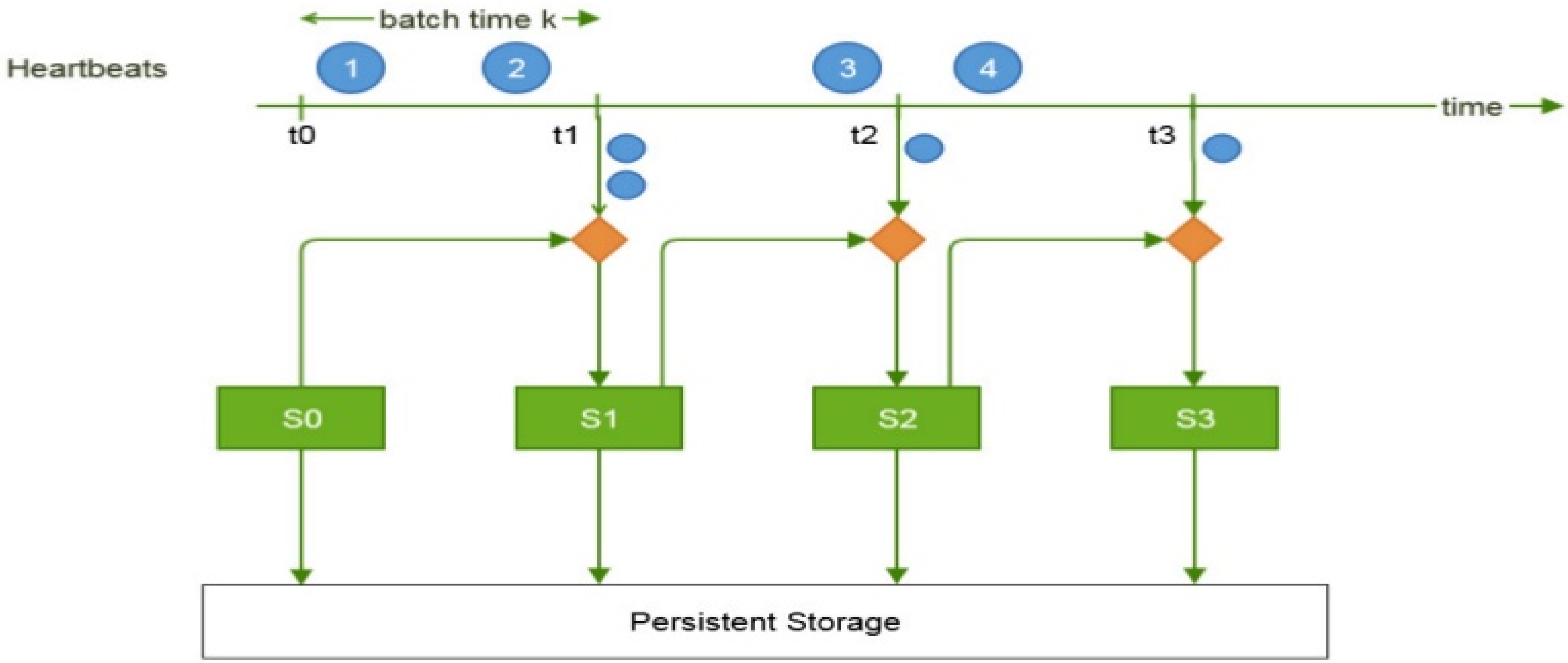
Major Issues with old stack

- **Code discrepancy among all 3 separate stacks**
  - RT: Pure updating model vs near RT: batch model
  - Offline: separate Hive layer; can have different calculation logic scattered in hive queries. (some standard UDFs/UDAFs help to certain extend)
- **A very complex and vulnerable RT stack**
  - Tricky thread locking
  - Mutable objects
  - Fixed data flow, specific delicate data partition, load balance.
- **Metric discrepancies cross all 3 stacks**
- **Different stacks also incur a lot of overhead of development, deployment**

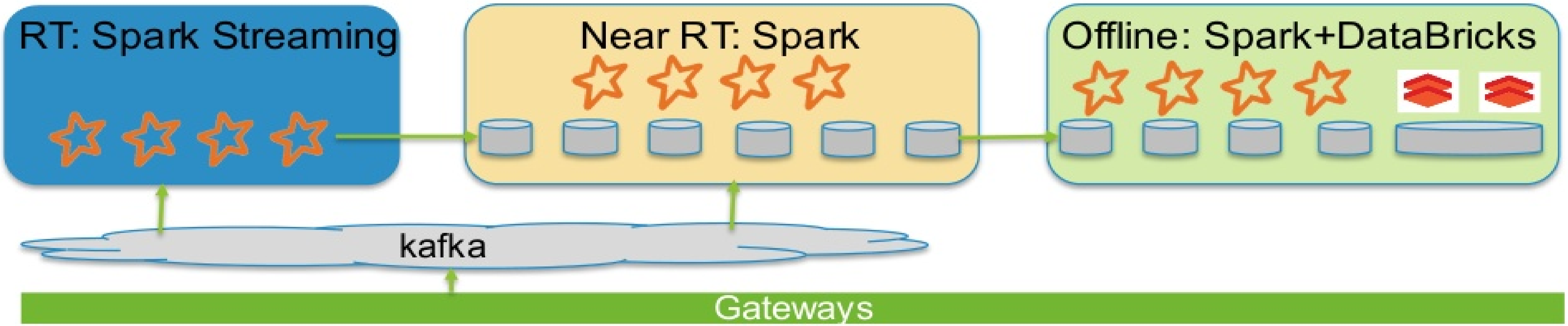
Proprietary Real Time Stack



Hadoop Batch Based Near RT Stack

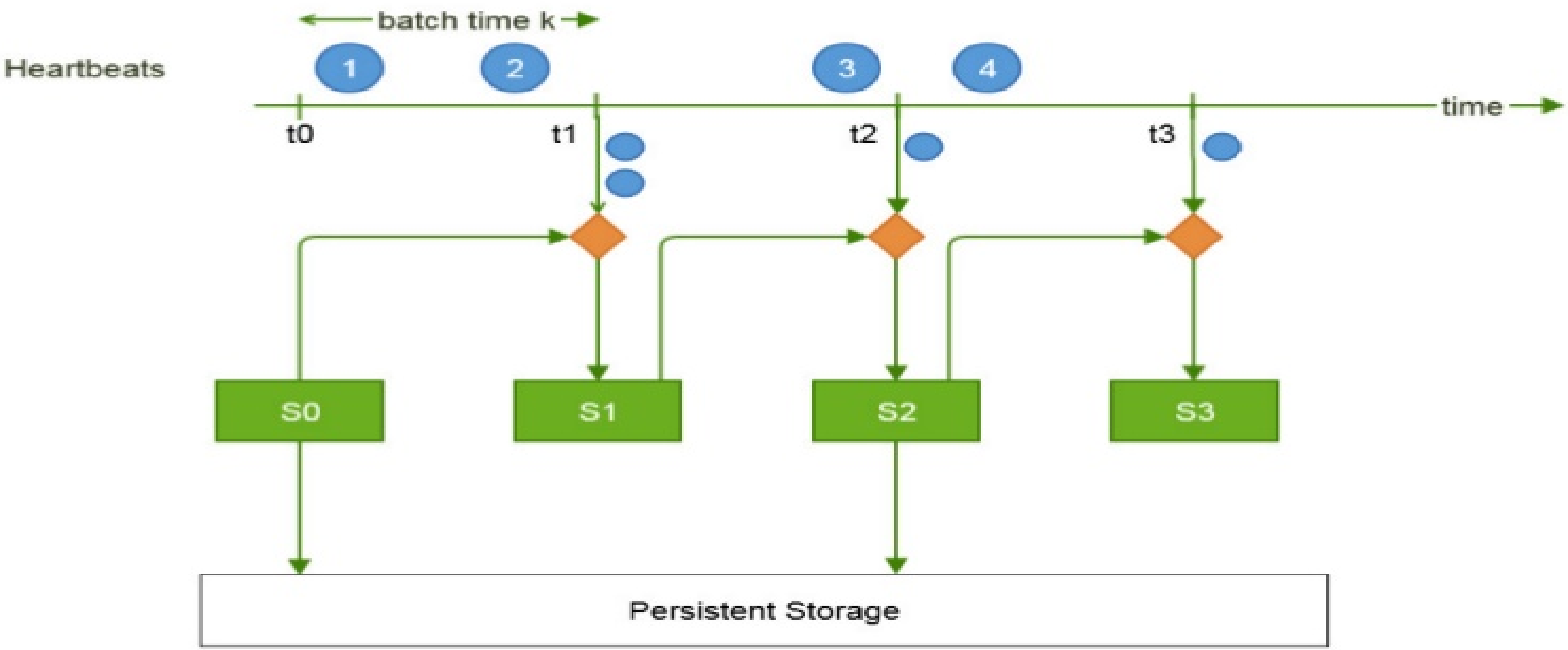


New Architecture



- All Converging to Spark based Technology.
- Max. sharing of code cross all 3 stacks
- Offline: with better cluster management (Databricks), running over many on-demand clusters instead of one pre-built

Unified Stack



Unified Stack High Level Code

```
val rawlogs = (DStream from rawlogs)

val ssMapped = rawlogs.transform {
   (map function)
}

val ssReduced = ssMapped.updateStateByKey{
   (reduce/update function)
}

(every n batches)
saveToStorage(ssReduced)
```

updateStateByKey, mapWithState

- Acts as the 'reduce' phase of the computation
- Helps maintain the evolving state shown earlier
- The performance of `updateStateByKey` is proportional to the size of the state instead of the size of the batch data
- In Spark 1.6, will be replaced in our workflow by `mapWithState`, which only updates as needed

Deployment

- RT portion In production environment for ~5 months
- Backward compatible migration first, major improvements later
- Performance tuning is important
- For RT: `Checkpointing`, reliability vs performance

Importance of the Offline Stack

- For data centric companies, most important innovations are happening with exploring and learning through the big data
- Speed and efficiency of offline data analysis and learning is the key to the success
- Data and Insights accessible to many internal organizations besides data scientists (customer support, SE, PM,…) is extremely important for the overall success

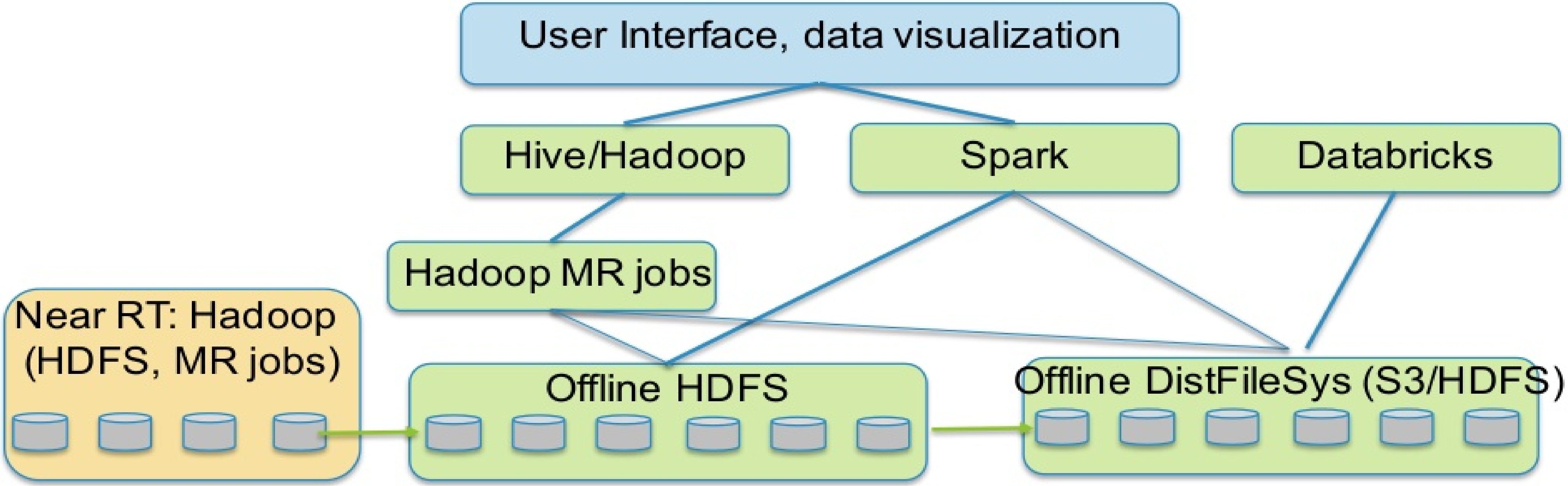
What's Important to Data Scientists

- Efficient access to all the data
- Can work independently with all the resources needed.
- Can work with the teams (internally and with other teams)
- Interactivity for data exploration
- Easy to use, powerful data visualization
- Machine learning tools

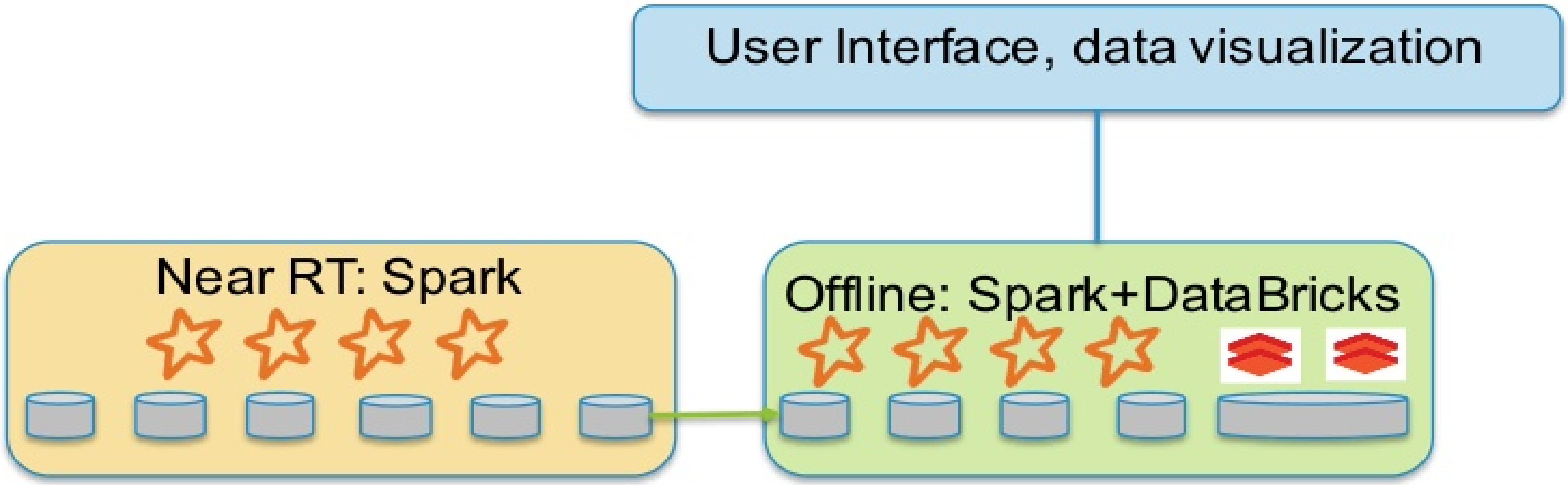
What's Important to Data Scientists

- Re-use of existing production logic/code when applicable
- Easy transfer of work into production
- Integrated environments with engineering discipline
  - Code management and version control
  - Design and code reviews

Old Architecture



New Architecture



Benefits of Databricks

- Cluster management:
  - Instead of one shared cluster, everyone can launch/manage his/her own clusters
- Interactive environments
  - Notebook environment is very convenient and powerful
- Easy to share/working together
  - Sharing notebooks are easy (with permission control)
- Data visualization
  - Good visualization tools: matplotlib, ggplot inside R
- Reasonably good machine learning tools
  - MLLib, R, other packages (H2O)

Benefits of Databricks

- Same code can potentially be moved to other stacks and production
- New features built faster here:
  - Harder to change production environment
  - New features developed, tested and deployed faster
- Huge efficiency gain for the data science team
- Production issue debugging also using Databricks with big efficiency gain

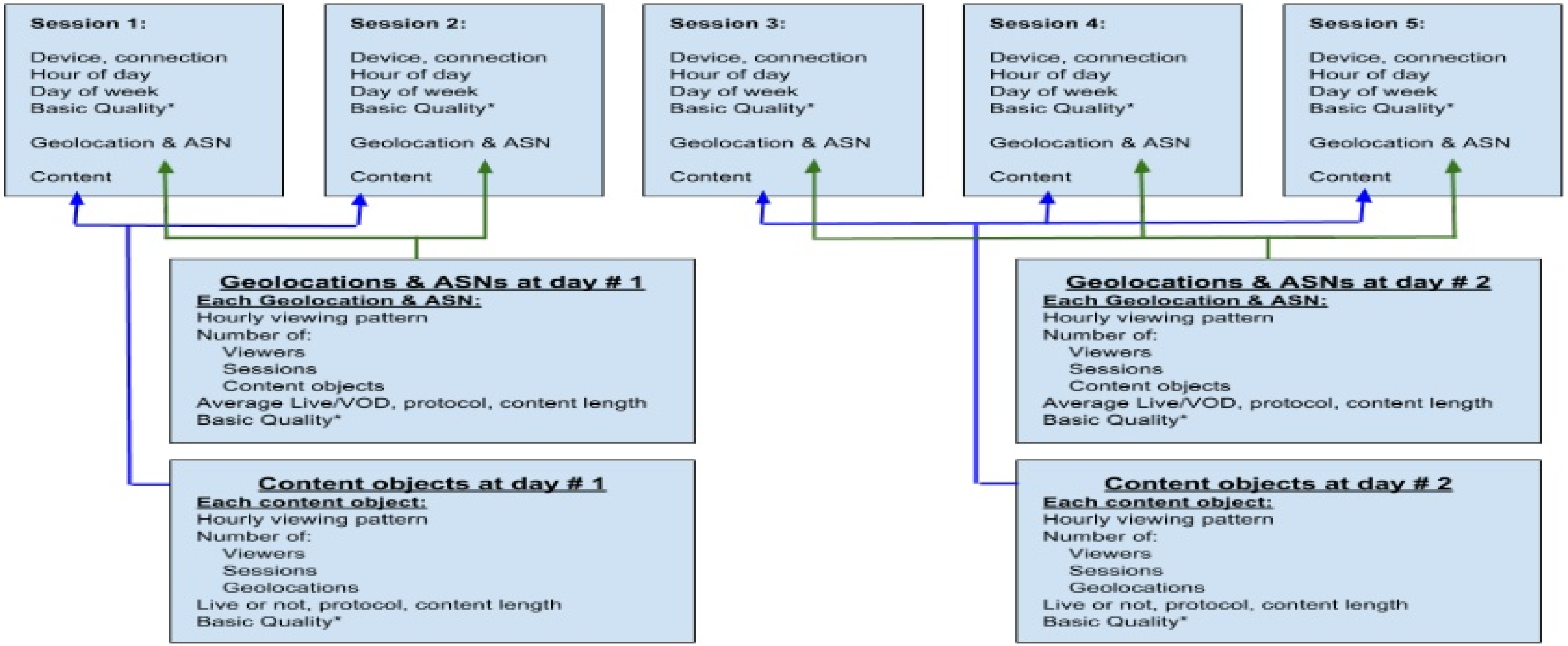
ML Example: Long Term Retention

- Long Term Retention Analysis
  - Months/years of data: many billions of views, many millions of subscribers per publisher/service provider.
  - Determining appropriate time interval for subscriber history and for subscriber abandonment.
  - Finding best features for predictive model.
  - Handling categorical features with too many possible values.

Characterization of Categorical Features

- One-hot encoding:
  - Some categorical features (e.g. Device) with dense limited values
- Some features have too many sparse categorical values
  - City & ASN
  - Video Content
- Aggregated features of many months of subscriber behavior:
  - All content that the subscriber watched
  - all Geolocations from which the viewer watched

Geolocation & Day



(*) ***Basic Quality***: Failed/not, Start-time, Buffering, Bit-rate

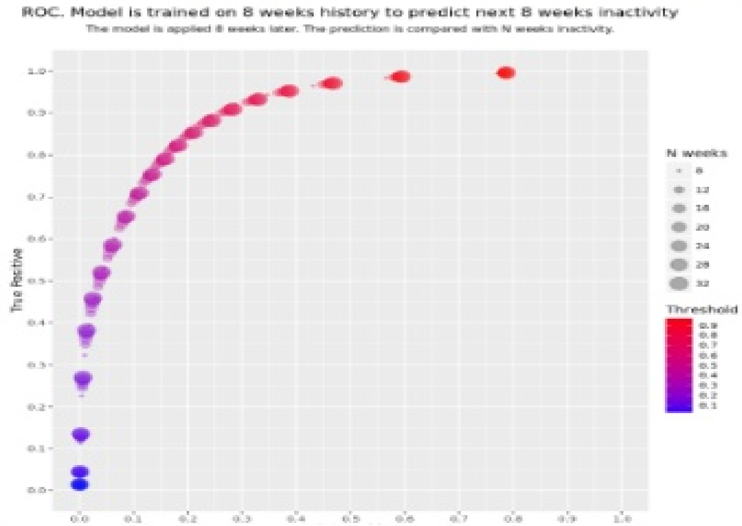
Work Flow Inside Databricks

- Create dataframes with features for each geo x day, content x day
- For each subscriber history, for each video session, replace geo and content with features of geo x day, content x day for that day
- Aggregate each subscriber history to obtain final features
- All done inside Databricks environment. Highly iterative process: especially related to feature design and extractions (many iteractions)
- Use Spark MLLib, various models, such as Gradient Boosted Tree Regression
- User visualization inside Databricks

Sample Results

Common ROC (churn) representation

```r
> ggplot(dfrROC,aes(x=dfrROC$FalsePositiveChurn, y=dfrROC$Selectivity)) +
    geom_point(alpha=0.3, aes(size=dfrROC$targ, color=dfrROC$tP)) +
    scale_size_continuous(name="N weeks", breaks=c(8,12,16,20,24,28,32), labels=c("8","12","16","20","24","28","32")) +
    scale_color_gradient(name="Threshold", low="blue", high="red", breaks=c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9), labels=c("0.1","0.2","0.3","0.4","0.5","0.6","0.7","0.8","0.9")) +
    ggtitle(expression(atop("ROC. Model is trained on 8 weeks history to predict next 8 weeks inactivity",
                            atop("The model is applied 8 weeks later. The prediction is compared with N weeks inactivity."),""))) +
    labs(x="False Positive",y="True Positive") +
    coord_cartesian(xlim=c(0,1), ylim=c(0,1)) +
    scale_x_continuous(breaks=c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1)) + scale_y_continuous(breaks=c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1))
```



Much More Work Ahead

- Improve the real time performance, trading off latency vs metrics accuracy/failure handling.
- <100ms real time processing and response still need more work.
- Making sure modular design for all the logics so that it can be shared cross all 3 stacks whereever possible
- More exciting and challenge works …

We Are Hiring

http://www.conviva.com/our-team/careers/

THANK YOU.

jibin@conviva.com

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO