OmniSci and RAPIDS: An End-to-End GPU Data Science Workflow

ODSC Webinar - May 30, 2019

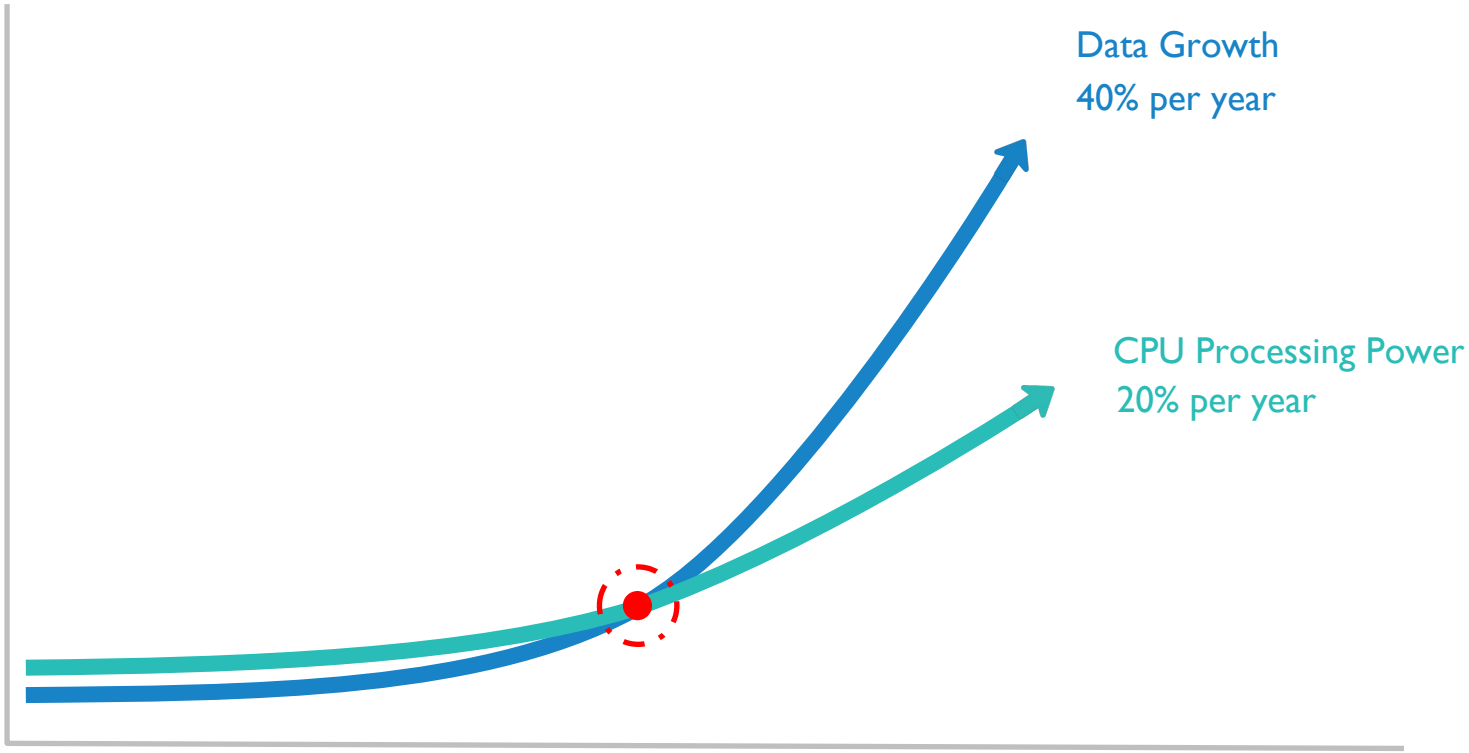
omnisci

Randy Zwitch
Senior Director of Developer Advocacy

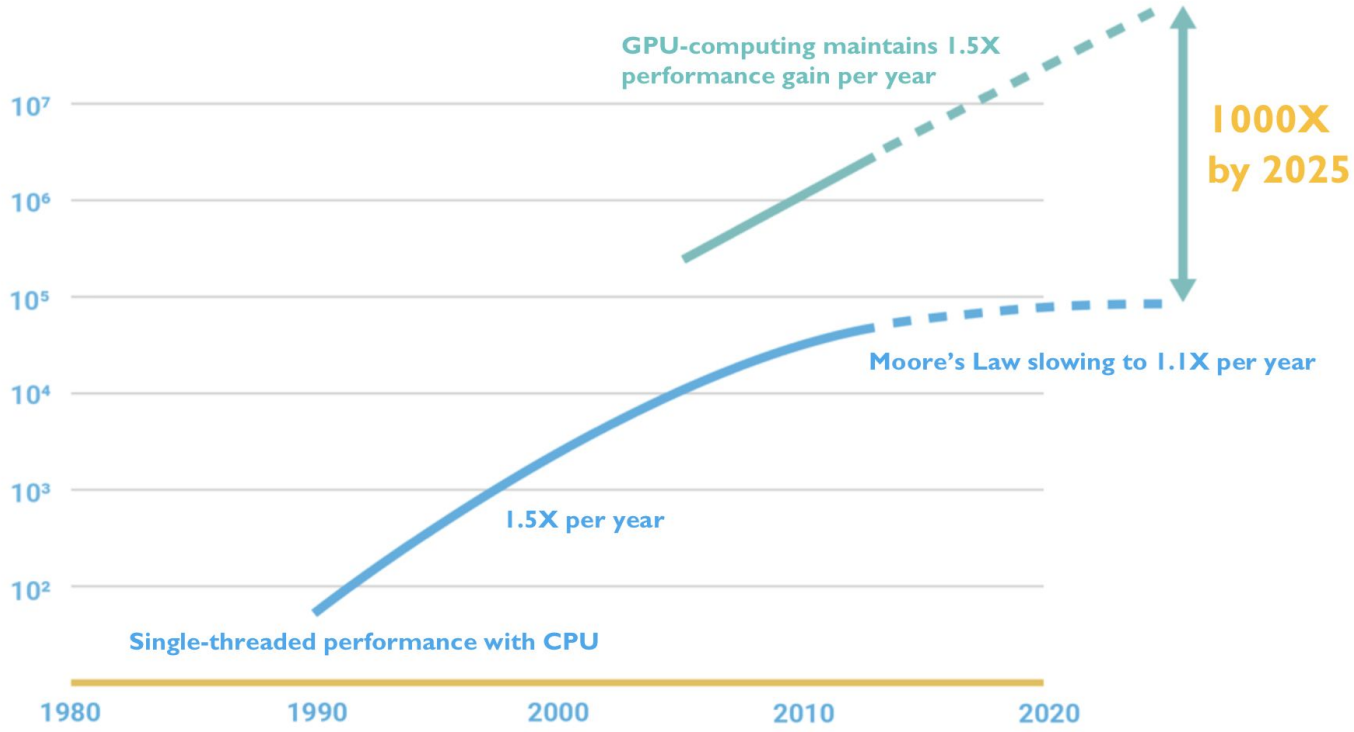
@randyzwitch
randy.zwitch@omnisci.com
/in/randyzwitch/
/randyzwitch


Volume
Agility
Spatio-
Temporal

Data Grows Faster Than CPU Processing

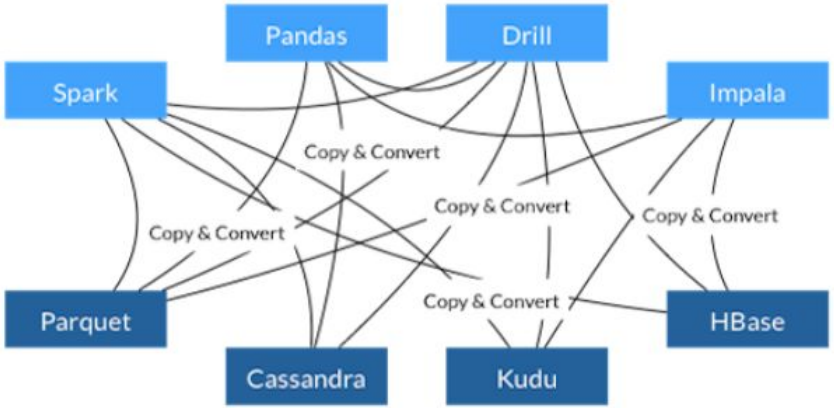


GPU Processing Keeps Moore's Law Alive

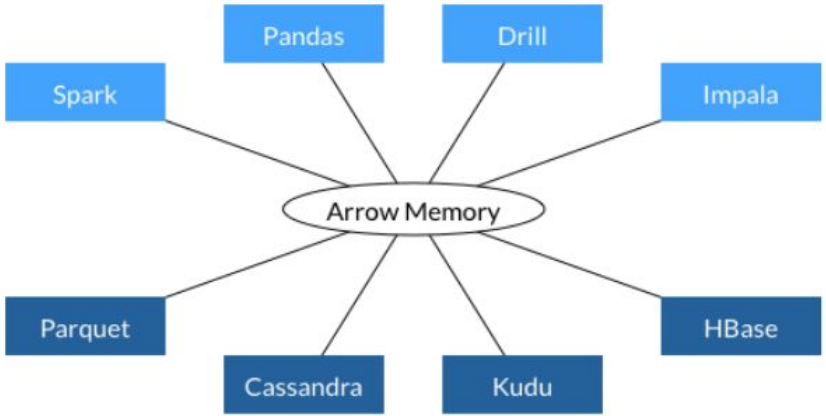


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

Apache Arrow: Shared Memory Layout



- Each system has its own internal memory format
- 70-80% computation wasted on serialization and deserialization
- Similar functionality implemented in multiple projects



- All systems utilize the same memory format
- No overhead for cross-system communication
- Projects can share functionality (eg, Parquet-to-Arrow reader)

Source: https://arrow.apache.org/

RAPIDS and the GPU DataFrame

Born from the GPU Open Analytics Initiative – fusing Machine Learning and GPU analytics

CONTRIBUTORS

ADOPTERS

OPEN SOURCE

By adopting a common memory layout in Apache Arrow, these tools can work seamlessly with one another with zero-copy memory transfer!

Demo: Real-Time Bikeshare Data

GitHub: https://github.com/omnisci/gbfs_kafka

Goal: Monitor Bike Availability in Real-Time

Architecture Considerations:

- Hundreds of API feeds conforming to GBFS specification: https://github.com/NABSA/gbfs

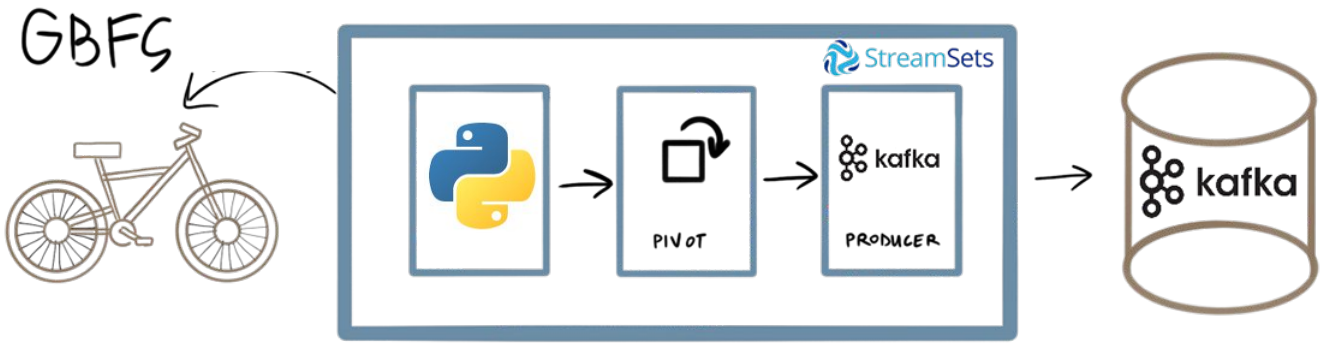
- Each feed provides relatively small amount of info as JSON; need to pre-process before loading to OmniSci

- Feeds have different TTL values; want to be respectful when pinging API endpoints

Data Pre-Processing: Python and StreamSets

Using Azure HDInsight, we set up a managed Apache Kafka cluster

- Kafka serves several purposes: aggregating feeds into a single stream, buffers for a more consistent throughput
- StreamSets is a data pipeline tool



Artwork: https://www.jowanza.com/blog/2018/9/8/real-time-station-tracking-ford-gobike-and-mapd

API Access Code Using Python

72 lines (54 sloc) | 2.57 KB — Raw | Blame | History

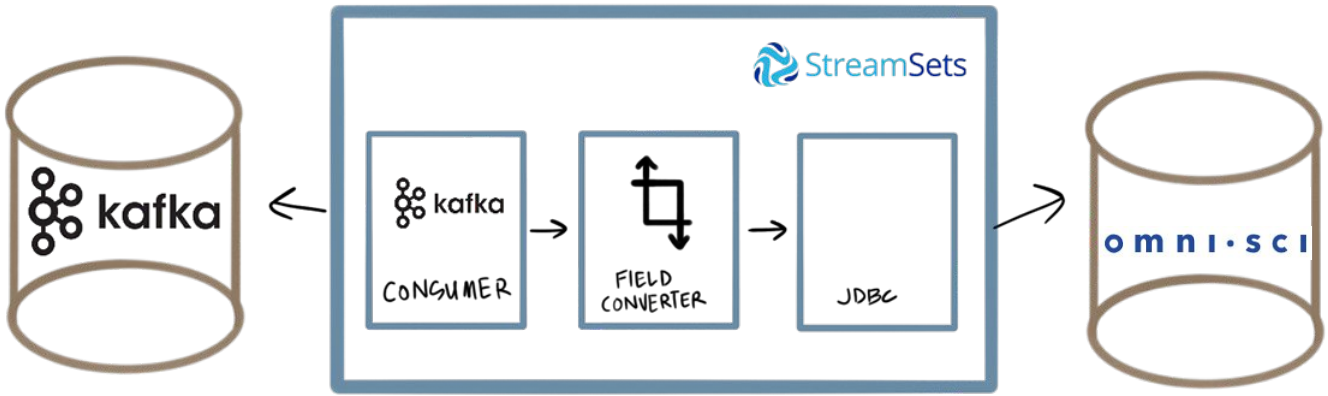
```python
import pandas as pd
import requests
from multiprocessing import Pool
import datetime
import simplejson as json

from credentials import hosts

#round to minute, set the same for all records
#this is so that each load represents same time period
start_time = int(datetime.datetime.now().replace(second=0, microsecond=0).timestamp())

# read in endpoints
endpoints_df = pd.read_csv("data/gbfs_endpoints.csv")

# simple function to be p.map[ped]
def get_url(url):

    r = None
    try:
        r = (requests.get(url, timeout=2).json(), url)
    except:
        pass

    return r

def feedlist(df, feedtype):
    return df[df["feedtype"] == feedtype].feedurl

# no reason for 12 specifically, just "some" parallelism
p = Pool(12)

#### free_bike_status
bike_status = p.map(get_url, feedlist(endpoints_df, "free_bike_status"))
```


https://github.com/omnisci/gbfs_kafka/blob/master/src/free_bike_status.py

Data Ingestion: StreamSets to OmniSci

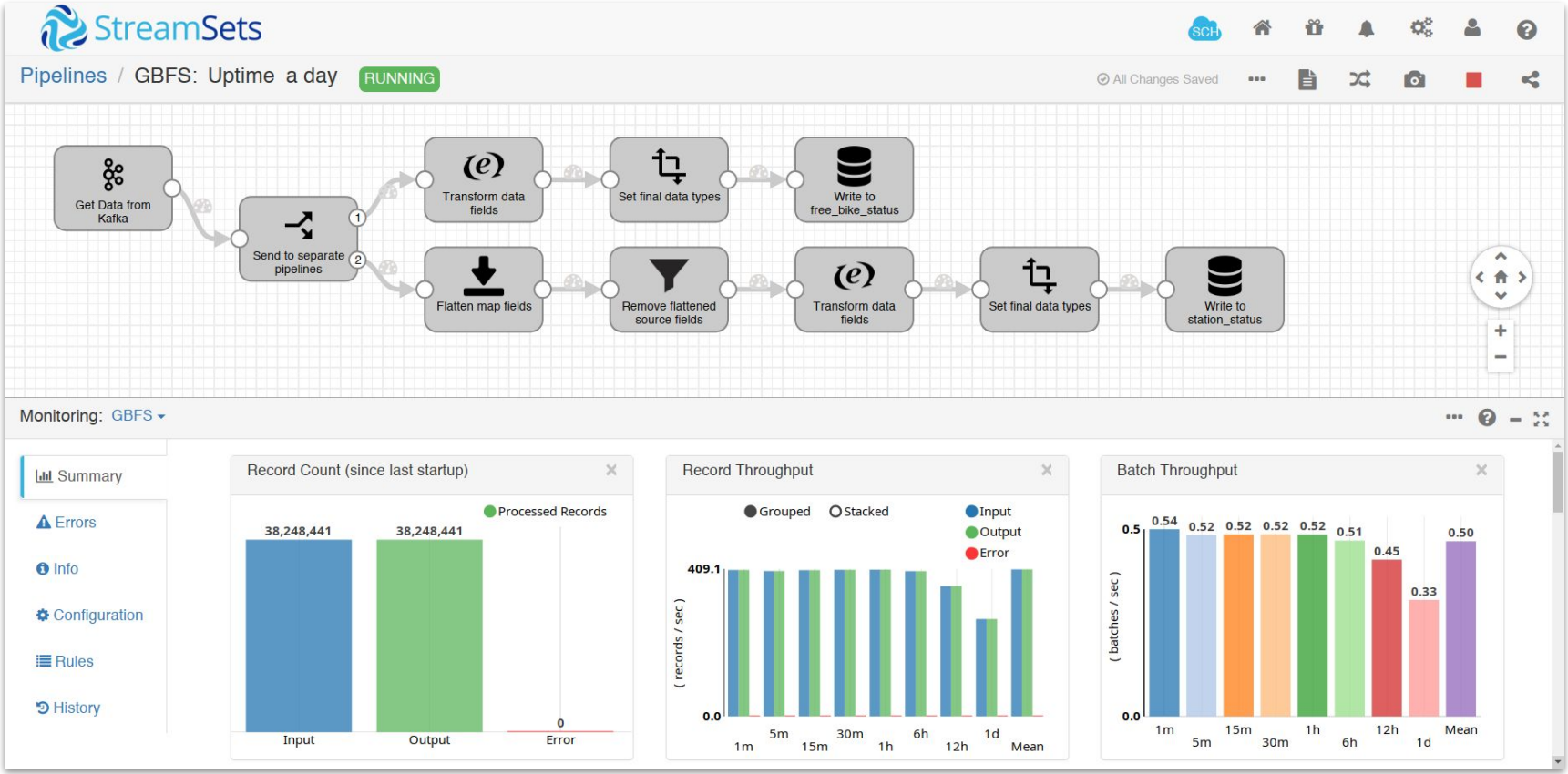
With feeds aggregated to single Kafka Producer (topic), ingest to OmniSci via JDBC

- OmniSci supports data streaming directly from Kafka, but using StreamSets allows for additional transformation
- Using JDBC along with StreamSets also allows StreamSets to manage retries and Kafka offsets



Artwork: https://www.jowanza.com/blog/2018/9/8/real-time-station-tracking-ford-gobike-and-mapd

Data Ingestion: StreamSets to OmniSci



Demo: Python and OmniSci for Data Science

GitHub: https://github.com/omnisci/odscwebinar

Exploring the Data Using Python

ODSC Webinar: OmniSci and RAPIDS

An End-to-End GPU Data Science Workflow

May 30, 2019

In [ ]:

```
#import pymapd to connect to OmniSci
#importing pandas is for convenience with the pd.read_sql method
#ibis is alternate method for querying using pandas-like API
import pymapd
import pandas as pd
import ibis
from credentials import credentials
```

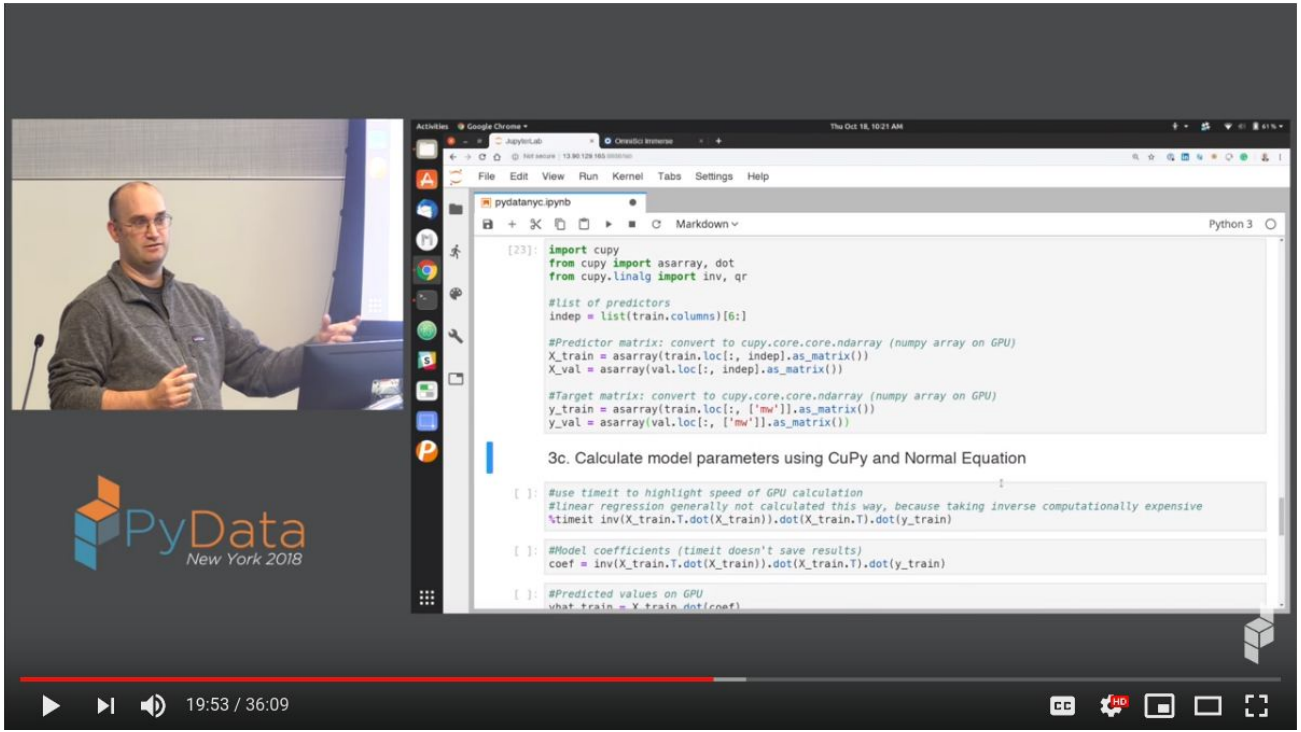
In [ ]:

```
# Connect to OmniSciDB, get list of tables in database
conn = pymapd.connect(host="localhost",
                      dbname=credentials["dbname"],
                      user=credentials["user"],
                      password=credentials["password"])
conn.get_tables()
```

In [ ]:

```
query = """SELECT
date_trunc(minute, accessed_on) accessed_on,
COUNT(*) AS records
FROM free_bike_status
WHERE accessed_on BETWEEN TIMESTAMP(0) '2019-05-30 00:33:21' AND TIMESTAMP(0) '2019-05-30 00:43:21'
GROUP BY 1
ORDER BY 1 DESC
LIMIT 60"""

df = pd.read_sql(query, conn)
```

Machine Learning With OmniSci and cudf



https://www.youtube.com/embed/gQszQcFHcZc

Questions

Randy Zwitch
Senior Director of Developer Advocacy

@randyzwitch
randy.zwitch@omnisci.com
/in/randyzwitch/
/randyzwitch