BigData Processing with Apache Spark

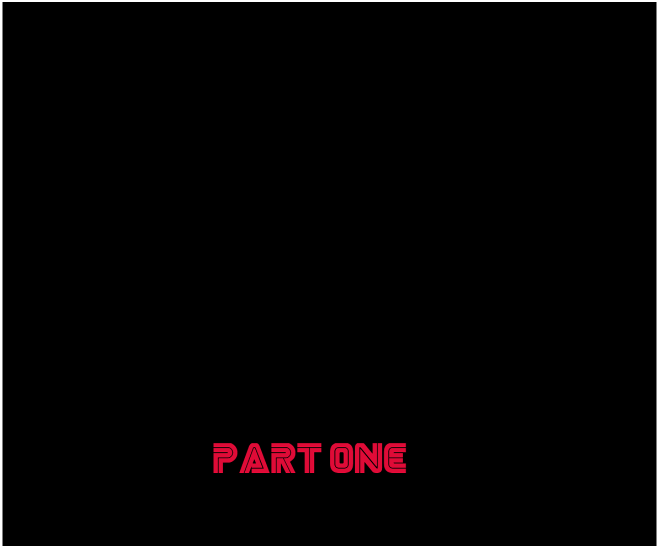
Alfred Thompson

Jul 20, 2016

Approach

- Part 1: Overview
- Part 2: Demo

Overview



What is BigData?

- huge datasets
- overwhelms single node, traditional clusters
- multiple dimensions of analysis
- ever growing
- critical use cases
- go cheap to scale out big

Data as Trash

- it scales...

Figure: litter

Figure: trash heaps

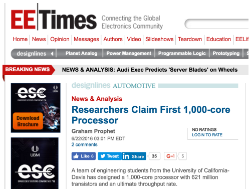
Figure: garbage cities

Figure: oceanic gyres

Data as Exhaust

- logs thrown off from system events, where system is
  - server
  - edge device
  - router/switch
  - mobile phone/watch/tablet
  - media players and servers
  - RFID transactions
- stuff to get
  - search terms
  - indexing logs
  - error logs
- can be correlated
  - *term* was searched from *geo* running *OS*
  - *phone* belonging to *user* entered *store*
  - *pasta and soda* were bought at *timestamp* from *vendor*

Conspiracy of Bigness



EETimes Connecting the Global Electronics Community
Home | News | Opinion | Messages | Authors | Video | Slideshows | Teardown | Education | EELif
designlines | Planet Analog | Power Management | Programmable Logic | Prototyping
esc Download Brochure
UBM esc
Like 6 Tweet Share 35 G+1 5

BREAKING NEWS NEWS & ANALYSIS: Audi Exec Predicts 'Server Blades' on Wheels

designlines AUTOMOTIVE

News & Analysis

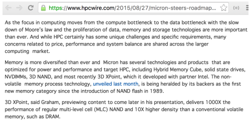
Researchers Claim First 1,000-core Processor

Graham Prophet
6/22/2016 03:01 PM EDT
2 comments

NO RATINGS
LOGIN TO RATE

A team of engineering students from the University of California-Davis has designed a 1,000-core processor with 621 million transistors and an ultimate throughput rate.

Figure: microprocessor



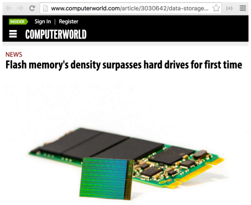
https://www.hpcwire.com/2015/08/27/micron-steers-roadmap...

As the focus in computing moves from the compute bottleneck to the data bottleneck with the slow down of Moore's law and the proliferation of data, memory and storage technologies are more important than ever. And while HPC certainly has some unique challenges and specific requirements, many concerns related to price, performance and system balance are shared across the larger computing market.

Memory is more diversified than ever and Micron has several technologies and products that are optimized for power and performance and target HPC, including Hybrid Memory Cube, solid state drives, NVDIMMs, 3D NAND, and most recently 3D XPoint, which it developed with partner Intel. The non-volatile memory process technology, unveiled last month, is being heralded by its backers as the first new memory category since the introduction of NAND flash in 1989.

3D XPoint, said Graham, previewing content to come later in his presentation, delivers 1000X the performance of regular multi-level cell (MLC) NAND and 10X higher density than a conventional volatile memory, such as DRAM.

Figure: memory

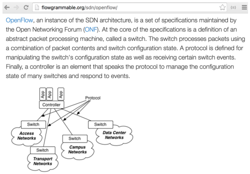


www.computerworld.com/article/3030642/data-storage...

NEWS

Flash memory's density surpasses hard drives for first time

Figure: storage



flowgrammable.org/sdn/openflow/

OpenFlow, an instance of the SDN architecture, is a set of specifications maintained by the Open Networking Forum (ONF). At the core of the specifications is a definition of an abstract packet processing machine, called a switch. The switch processes packets using a combination of packet contents and switch configuration state. A protocol is defined for manipulating the switch's configuration state as well as receiving certain switch events. Finally, a controller is an element that speaks the protocol to manage the configuration state of many switches and respond to events.

Figure: network

What is Grid/Cluster?

- gangs of computers
- HA: high availability
  - focus on nonstop processing
  - use fewer, higher end hardware nodes
  - multipath networking and I/O channels
- HPC: high performance computing
  - focus on max throughput running massive workloads
  - run many, dense capacity nodes
  - fortran is the killer app
  - message passing
- deployment models
  - shared memory
  - shared nothing
  - master-slave
  - multi-master

Grids as Trash

- build grids with off-the-shelf hardware
  - commodity servers
  - low cost
  - bulk availability
  - assume failure, be resilient when disaster comes
  - virtualizable
  - replication
- forerunners: beowulf, KLAT2

Other Conspiracies

- functional programming
- JVM/Groovy/Scala/Clojure
- open source
- mobile platforms
- social networks
- e-commerce
- advertising
- data science

Seminal Efforts

- Google Papers
  - GFS
  - MapReduce
- Yahoo!/Nutch/Hadoop

Hadoop ecosystem

- HDFS
- MapReduce
- YARN
- HBase
- Zookeeper
- plus OLAP, SQLs, graph, event processing, job workflow

Next Stage of Evolution: Spark

- high speed, in memory processing
- simplified programming model
- toolkit for machine learning
- efficiently stream/spill to I/O
- reuse, extend the Hadoop ecosystem

Spark Architecture

- RDD: resilient distributed dataset
- partition
- accumulate transformations over dataset
- process lazily
- program in Scala, Python, Java
- process data as SQL tables, graphs, streams
- deploys standalone, hadoop, mesosphere, AWS

Use Cases

- batch: better mapreduce
- machine learning
- data science
- complex event processing

Break

Demo

Flow

- create a VPC with internet gateway & subnet with external egress
- create a security group allowing SSH ingress
- start an EMR cluster with Spark and Zeppelin installed
- SSH into cluster master to verify install
- SSH tunnel notebook port to localhost
- navigate to notebook port and run Spark operations
- save analysis (as needed)
- terminate cluster

Flight Data

- access zeppelin
- create note header
- load data
- sample data and load to table
- query: 10 airports with most departures
- query: most flight delays over 15 mins
- query: most flight delays over 60 mins
- query: most flight cancellations
- query: most popular flight routes

Access Zeppelin

- navigate to `http://localhost`:8890/

Create Note Header

```
%md
Flight delays and cancellations for domestic flights in the U.S.
===
[Al Thompson](http://github.com/almacro)
---

Reference:
  [New - Apache Spark on Amazon EMR]
  (https://aws.amazon.com/blogs/aws/new-apache-spark-on-amazon-emr)
```

Load Data

```
val parquetFile = sqlContext
  .read
  .parquet("s3://us-east-1.elasticmapreduce.samples/flightdata/input")
parquetFile.count()  /** Number of rows */
```

Sample and Load Data

```
parquetFile.take(3).foreach(println)
parquetFile.registerTempTable("flights")
```

Query: Most Departures

```
%md
The 10 airports with the most departures
===

%sql
SELECT origin, count(*) AS total_departures FROM flights
WHERE year >= 2000
GROUP BY origin
ORDER BY total_departures
DESC LIMIT 10
```

Query: Most 15+ min Delays

```
%md
The most flight delays over 15 minutes
===

%sql
SELECT origin, count(depdelayminutes) AS total_delays
FROM flights
WHERE depdelayminutes > 15 and year >=2000
GROUP BY origin
ORDER BY total_delays
DESC LIMIT 10
```

Query: Most 60+ min Delays

```
%md
The most flight delays over 60 minutes
===

%sql
SELECT origin, count(depdelayminutes) AS total_delays
FROM flights
WHERE depdelayminutes > 60 and year >=2000
GROUP BY origin
ORDER BY total_delays
DESC LIMIT 10
```

Query: Most Cancellations

```
%md
The most flight cancellations
===

%sql
SELECT origin, count(cancelled) AS total_cancellations
FROM flights
WHERE cancelled = 1 and year >=2000
GROUP BY origin
ORDER BY total_cancellations
DESC LIMIT 10
```

Query: Most Cancellations of Total Flights

```
%md
The most flight cancellations out of total flights, as percentage
===

%sql
with total_flights as (
SELECT origin, count(*) AS total_flights
FROM flights
WHERE year >=2000
GROUP BY origin
ORDER BY total_flights
DESC LIMIT 10), total_cancellations as
(
SELECT origin, count(cancelled) AS total_cancellations
FROM flights
WHERE cancelled = 1 and year >=2000
GROUP BY origin
ORDER BY total_cancellations
DESC LIMIT 10)
SELECT flights.origin,
       total_flights,
       total_cancellations,
       format_number(100*total_cancellations/total_flights, 2)
       AS percent
FROM total_flights flights
INNER JOIN total_cancellations cancellations
ON flights.origin = cancellations.origin
ORDER BY percent DESC
```

Query: Most Popular Routes

```
%md
The 10 most popular flight routes
===

%sql
SELECT origin, dest, count(*) AS total_flights FROM flights
WHERE year >= 2000
GROUP BY origin, dest
ORDER BY total_flights
DESC LIMIT 10
```

Resources

- aws blog

https://aws.amazon.com/blogs/aws/new-apache-spark-on-amazon-emr/

- cloud academy

http://cloudacademy.com/blog/big-data-amazon-emr-apache-spark-and-apache-zeppelin-part-one-of-two/

http://cloudacademy.com/blog/big-data-getting-started-with-amazon-emr-apache-spark-and-apache-zeppelin-part-two-of-two/

- spark project

http://spark.apache.org/

- zeppelin project

https://zeppelin.apache.org/

Thanks

Extras

Create Gateway

```
aws ec2 create-internet-gateway
```

Create VPC

```
aws ec2 create-vpc --cidr-block 10.101.0.0/16
```

Configure DNS in VPC

```
aws ec2 modify-vpc-attribute \
    --vpc-id AAA \
    --enable-dns-support "{\"Value\": true}"

aws ec2 modify-vpc-attribute \
    --vpc-id AAA \
    --enable-dns-hostnames "{\"Value\": true}"
```

Configure Networking

```
aws ec2 attach-internet-gateway \
    --internet-gateway-id AAA \
    --vpc-id BBB

aws ec2 create-subnet \
    --vpc-id AAA \
    --cidr-block 10.101.1.0/24

aws ec2 create-security-group \
    --name sparkdemo-sg \
    --description "Spark Demo, allow inbound" \
    --vpc-id AAA

aws ec2 authorize-security-group-ingress \
    --group-id AAA \
    --protocol tcp \
    --port 22 \
    --cidr 0.0.0.0/0
```

Configure Routes

```
aws ec2 create-route-table --vpc-id AAA

aws ec2 create-route \
    --route-table-id AAA \
    --gateway-id BBB \
    --destination-cidr-block 0.0.0.0/0

aws ec2 associate-route-table \
    --subnet-id AAA \
    --route-table-id BBB
```

Create Cluster

```
aws emr create-cluster \
    --name "Demo Spark Cluster" \
    --release-label emr-4.7.2 \
    --applications \
      Name=Spark Name=Zeppelin-Sandbox \
    --ec2-attributes \
      KeyName=AAA,SubnetId=BBB,AdditionalMasterSecurityGroups=CCC \
    --instance-type m4.xlarge \
    --instance-count 3 \
    --use-default-roles
```

Describe Cluster

```
aws emr describe-cluster --cluster-id AAA
```

SSH to Master

```
aws emr ssh --cluster-id AAA --key-pair-file BBB
```

Verify Install

```
spark-shell

pyspark

sparkR
```

SSH Tunnel for Notebook port

```
ssh \
    -o StrictHostKeyChecking=no \
    -o ServerAliveInterval=10 \
    -i AAA \
    -L 8890:127.0.0.1:8890 \
    hadoop@ec2-XXX.compute-1.amazonaws.com \
    -Nv
```

Terminate Cluster

```
aws emr terminate-clusters --cluster-ids AAA
```

All Done