



# ABCD: AI, BigData, Cloud 深入 集成 ( D )



嘉宾：田丰

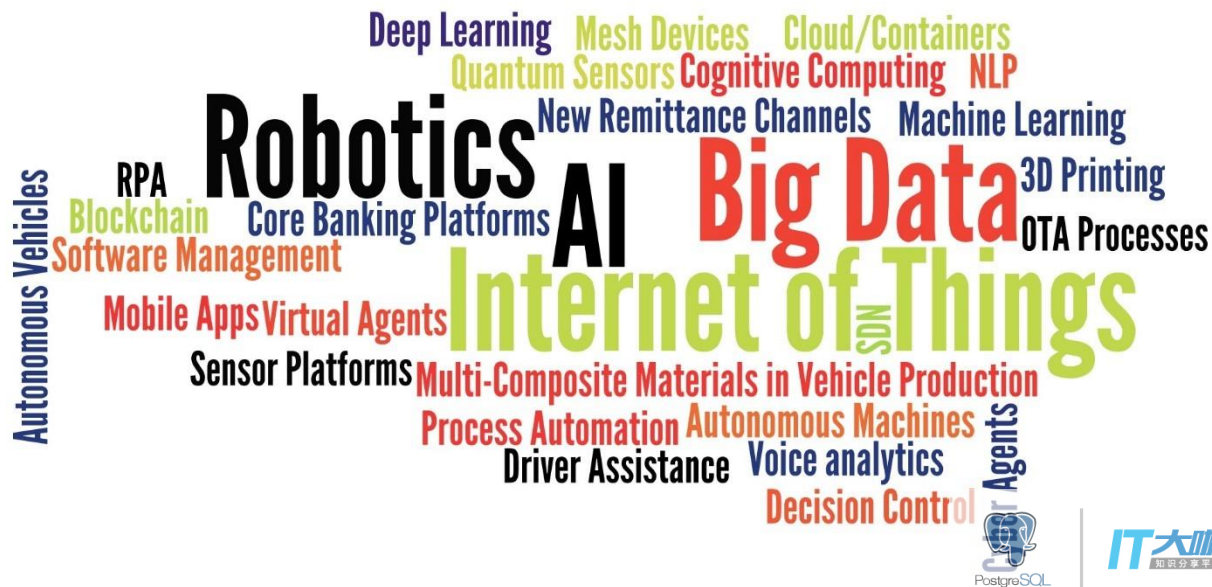
公司：vitedata.cn





# ABC

- AI + BigData + Cloud
- **DATA!!!**





# DATA!

- ETL
- 数据清理
- 数据集成
- SQL
- AI, Deep Learning
- Machine Learning





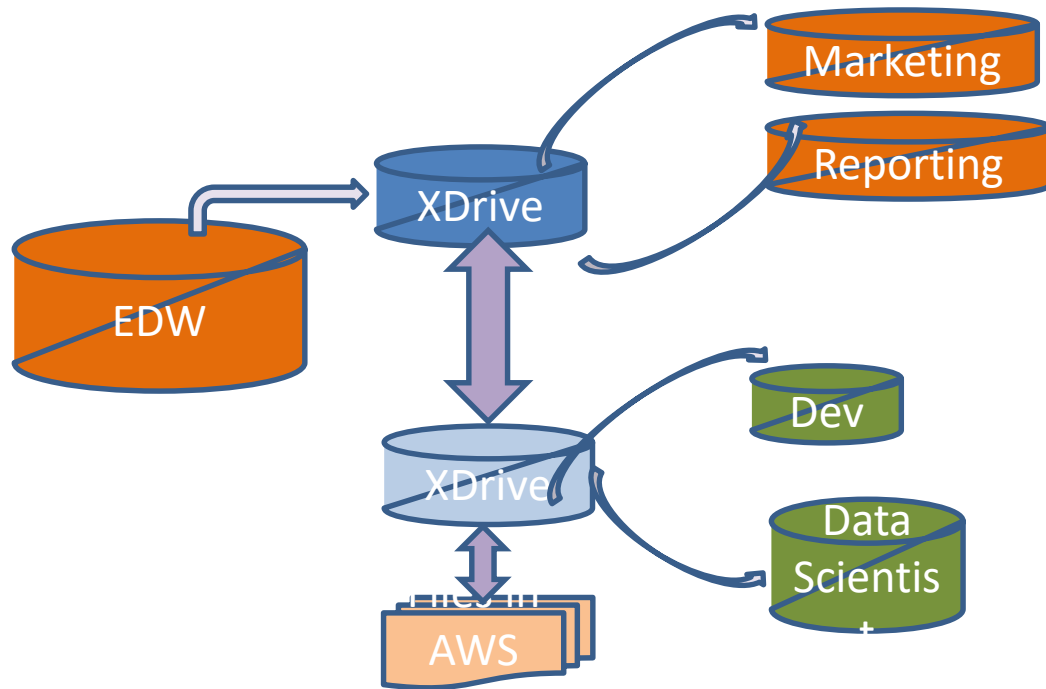
# Deepgreen

- MPP
  - Greenplum Compatible\*
    - Except quicklz, replaced with lz4.
  - 高性能，扩展
  - 新功能
    - Decimal, re\_xxx, sample, approximate\_count\_distinct,
- Cloud
  - AWS
  - 近期推出其它云服务





# XDrive: DataLake 解决方案





# XDrive

- High Perf Protocol, 并行
- 数据共享
- 条件下推
- 内建
  - nfs/fs, hdfs, S3
  - Csv, spq

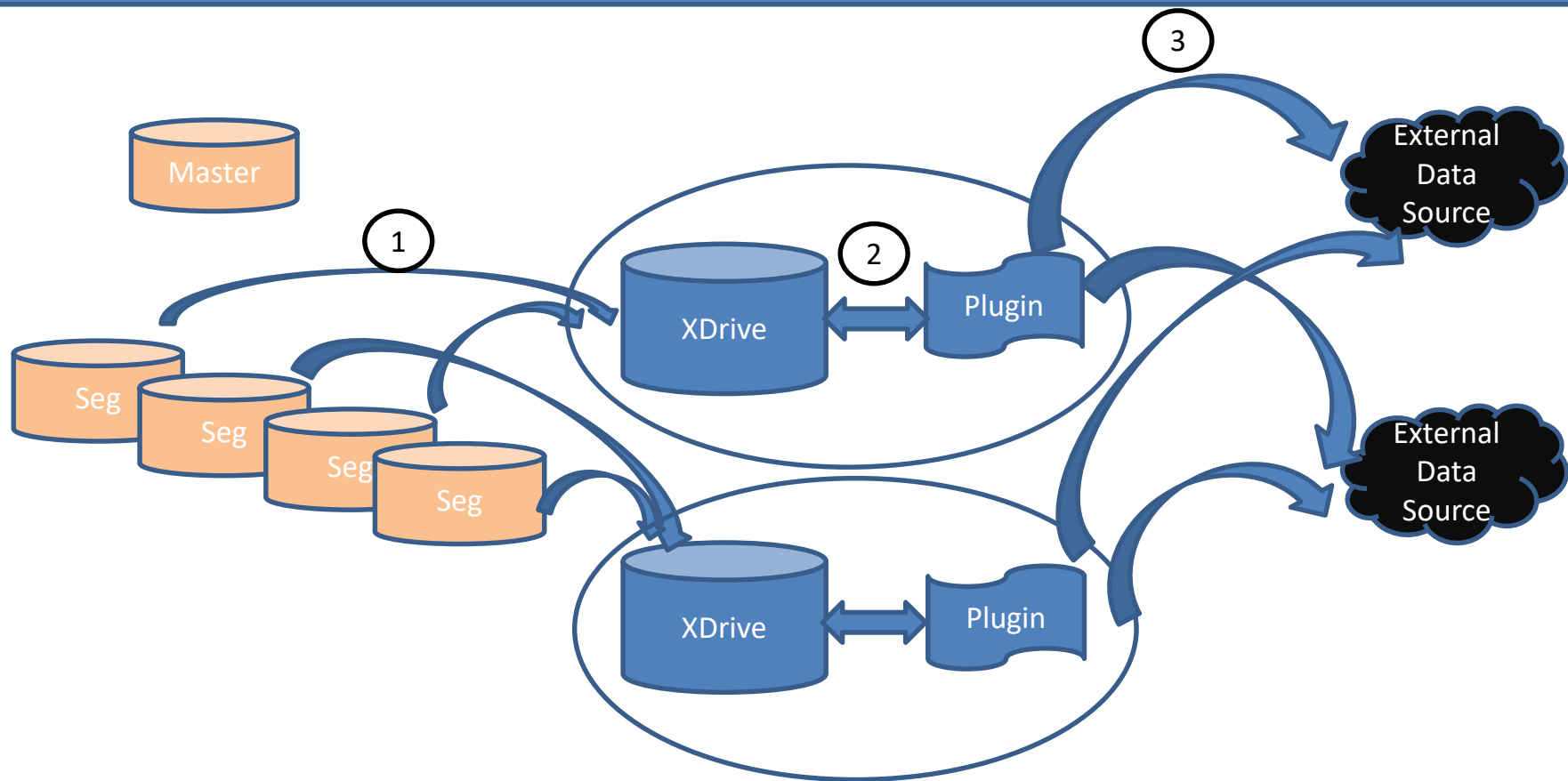




# XDrive

- 计算和存储分离
- 配置简单
- 弹性伸缩









# Xdrive Plugin

- 已有
  - HDFS + Parquet , Orc
  - Elastic search





# 关系型数据库

- 关系型的数据
- 关系型的查询





# 关系型的查询 ( SQL )

- Relational Algebra/Calculus and First Order Logic
  - Select, filter, project, join, aggregate, union/all, order by ...
  - OLAP Window functions
  - With Recursive





# Turing Complete!

- Turing Tarpit





# 新的（非关系型）查询

- Time series
- 图
- AI, 学习？
- ...





# PHI

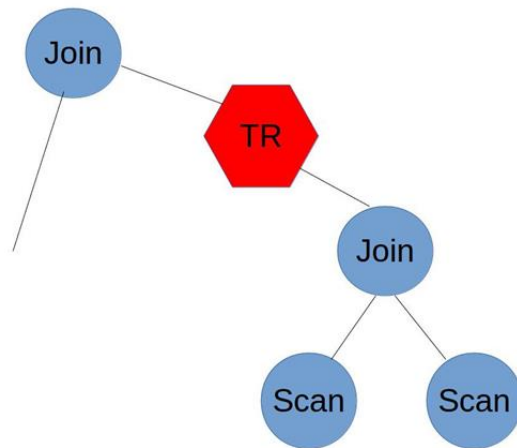
- Pretty humble intelligence, pretty high integration





# Transducer

- SQL (UDF 语法 )
- Scripting DB Engine
  - Go
  - Python
- 一定需要深度集成
  - 优化器
  - 保证SQL语义
  - 执行/并行





# 例子：Basic

```
Ret function(Args) {  
    // TransducerBody  
}
```

```
Select  
... -- return columns  
TransducerBody  
... -- arg columns  
FROM  
....
```





## Select id, t from t where id % 3 = 1

Explain and explain analyze all works.

### QUERY PLAN

Gather Motion 2:1 (slice1; segments:

2) (cost=0.00..2.05 rows=2 width=4)

-> **Transducer** (cost=0.00..2.05 rows=1 width=4)

-> Seq Scan on eachseg (cost=0.00..2.05 rows=1 width=4)

Optimizer status: legacy query optimizer

(4 rows)

```

Select transducer_col_int4(1) as id, ①
       transducer_col_text(2) as t,
       transducer($$PHIExec go ②
// The following is a valid go program ③
// BEGIN INPUT
// id int32
// t text
// END INPUT
// BEGIN OUTPUT
// id int32
// t text
// END OUTPUT
package main
func main() {
  for rec := NextInput(); rec != nil; rec = NextInput() { ④
    id, _ := rec.Get_id() ⑤
    t, _ := rec.Get_t()
    if id % 3 == 1 { ⑥
      var outrec OutRecord
      outrec.Set_id(id)
      outrec.Set_t(t)
      WriteOutput(&outrec) ⑦
    }
    WriteOutput(nil) ⑧
  }
} $$), ⑨
t.id, t.t ⑩
From t

```

Program Listing 1: Simple Transducer





# 几点注释

- Python ( interpreted ) , Go ( compiled )
- Code generation
  - 少量数据Go会慢一点，大量的数据快
- UDF语法，除此之外和UDF没关系
- 不是row by row





# 例子：ETL/DBLink

- 任何外部数据
  - Go/python driver
  - 并行？

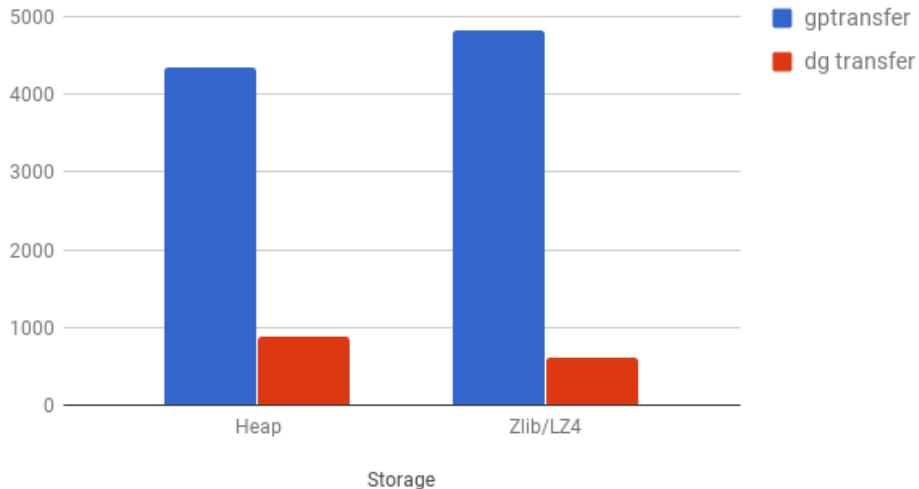
```
... boilerplate code ...
Transducer ($$ PhiExed go
// boilerplate code
open connection to another database
rs, err := conn.Execute("select * from lineitem")
for rs.Next() {
    WriteOutput(...)
}
WriteOupt(nil)
$$)
... boiler plate code ...
```



# 例子：dg transfer

- Transducer+Xdrive
- 6+ billion rows in 10 min
- 10 million rows per second

gptransfer and dg transfer 1TB 4 machines with 2x10GigE





# 例子：run

- 找出有连续 10 天上涨的股票。
- OLAP Window function
- Cursor ?
- 和去年的数





```

WITH run AS ( select                                1
Transducer_col_text(1) as symbol,
Transducer_col_int4(2) as begin,
Transducer_col_float8(3) as beginprice,
Transducer_col_int4(3) as end,
Transducer_col_float8(4) as endprice,
Transducer($$PHIExec go
// ... input and output types ...
var outrec *OutRecord                               2
for r:=NextInput(); r!=nil; r=NextInput() {
symbol, _ := r.Get_symbol()
day, _ := r.Get_day()
price, _ := r.Get_price()
if day == 0 {                                       3
// new symbol, output prev run and start
// a new run
if outrec != nil {
WriteOutput(outrec)
}
outrec = new(OutRecord)                             4
outrec.Set_symbol(symbol)
outrec.Set_begin(day)
outrec.Set_beginprice(price)
outrec.Set_end(day)
outrec.Set_endprice(price)
}

```

Program Listing 2: Compute Stock Runs

```

} else {
// is still a run?                                  5
Isuprun := ...
Isdownrun := ...
If isuprun || isdownrun {
Outrec.Set_end(day)                                6
Outrec.Set_endprice(price)
} else {
WriteOutput(rec)
... // start a new run
}
}
if outrec != nil {
WriteOutput(outrec)
}
WriteOutput(nil)
...
$$), t.symbol, t.day, t.price FROM
( select row_number() over
(partition by symbol order by day),                7
Symbol, day, price
FROM stock
) t)
SELECT stock from run where end - day > 10 8

```

Program Listing 2: Compute Stock





## QUERY PLAN

---

Gather Motion 2:1 (slice1; segments: 2) (cost=48.58..74.58 rows=800 width=44)

-> **Transducer (cost=48.58..74.58 rows=400 width=44)**

-> Subquery Scan t (cost=48.58..74.58 rows=400 width=44)

-> **Window (cost=48.58..54.58 rows=400 width=15)**

**Partition By: stock.symbol**

**Order By: stock.day**

-> Sort (cost=48.58..50.58 rows=400 width=15)

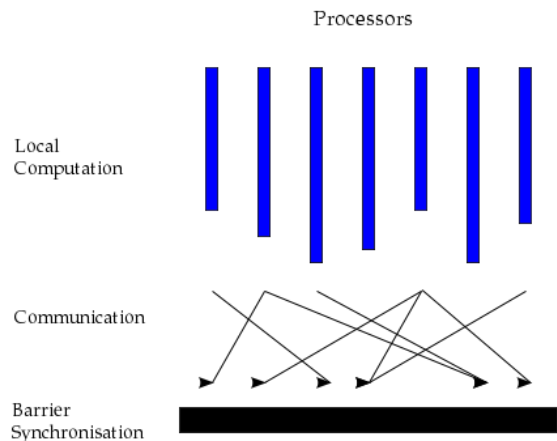
Sort Key: stock.symbol, stock.day

-> Seq Scan on stock (cost=0.00..10.00 rows=400 width=15)





# BSP (Bulk Synchronous Parallel)



- Google Pregel
  - Graph
    - BFS , SSSP
    - Triangle #
    - PageRank
- Recursive Query Processing
  - BOM





# BSP

- BspInit
- BspSend
- BspNext
- BspSync





# 例子：图 BFS

```

... boilerplate type code ...
... suppose graph edge is (i, j)
func main() {
  BspInit(2)

```

```

// SuperStep 1, redistribute
For r:=NextInput(); r!=nil; r=NextInput() {
  BspSend(i%2, r)
  BspSend(j%2, r)
}
BspSync(false)

```

```

// SuperStep 2, build graph. Graph is
// a map from node id to out edges.
// Each node has a depth, initialized to -1
// except start node of the search (depth 0)
BspSend(start%2, start)
BspSync(false)

```

Program Listing 3. BFS Using BSP

```

For Sstep := 2; ; sstep++ {
  For rr:=BspNext(); rr!=nil; rr=BspNext() {
    // BSF, if depth is -1, mark it
    // BspSend to next SuperStep
  }
  // If no node is marked in this step, vote
  // true, otherwise vote false
  Done := BspSync(ifNoMarkThisStep)
  If Done {
    Break
  }
}

```

```

For node in graph {
  WriteOutput(id, depth)
}
WriteOutput(nil)

```

```

... boilerplate code
FROM Graph;

```

Program Listing 3. BFS Using BSP





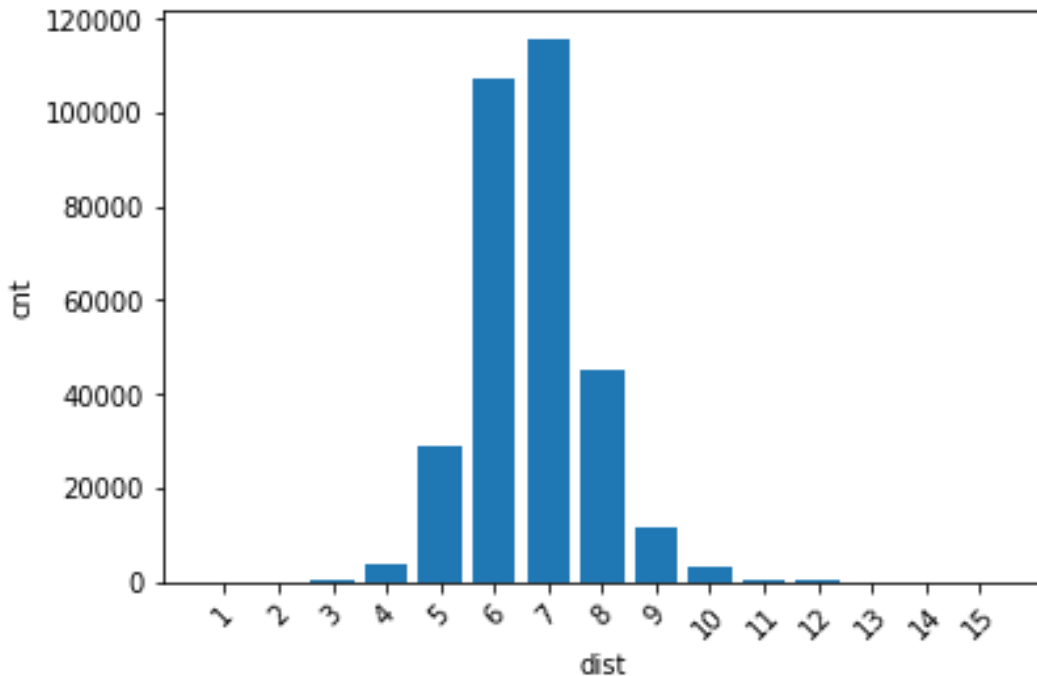
# DBLP

- 300K nodes (作者)
- 1M edges (合作)
- 有 Gap (从不和别人合作的人)





# DBLP





# 例子：SSSP (Bellman-Ford from Wikipedia)

```
function BellmanFord(list vertices, list edges, vertex source)
::distance[],predecessor[]

// This implementation takes in a graph, represented as
// lists of vertices and edges, and fills two arrays
// (distance and predecessor) with shortest-path
// (less cost/distance/metric) information

// Step 1: initialize graph
for each vertex v in vertices:
    distance[v] := inf      // At the beginning , all vertices
                             have a weight of infinity
    predecessor[v] := null // And a null predecessor

    distance[source] := 0   // Except for the Source, where
                             the Weight is zero
```

```
// Step 2: relax edges repeatedly
for i from 1 to size(vertices)-1:
    for each edge (u, v) with weight w in edges:
        if distance[u] + w < distance[v]:
            distance[v] := distance[u] + w
            predecessor[v] := u
```

```
// Step 3: check for negative-weight cycles
for each edge (u, v) with weight w in edges:
    if distance[u] + w < distance[v]:
        error "Graph contains a negative-weight cycle"
return distance[], predecessor[]
```





# SSSP

```
for {
  sstep++
  relaxed := false
  for brec := BspNext(); brec != nil; brec = BspNext() {
    // For each node, we recieved a update
    a, _ := brec.Get_a()
    b, _ := brec.Get_b()
    w, _ := brec.Get_w()
    n := graph[b]
    if n.w >= w {
      // relax edges.
      n.w = w
      n.pre = a
      n.sstep = sstep
      relaxed = true
    }
  }
}
```

```
// If we relaxed an edge in this iteration, need to continue.
if relaxed {
  for a, n := range graph {
    if n.sstep == sstep {
      // Relaxed in this iteration.
      for _, e := range n.edge {
        var rr BspRecord
        rr.Set_a(a)
        rr.Set_b(e.next)
        rr.Set_w(n.w + e.w)
        BspSend(e.next%2, &rr)
      }
    }
  }

  sync := BspSync(!relaxed)
  if sync < 0 {
    break
  }
}
```





# Bellman Ford

- 把 BFS 的 loop 换掉
- 100 行
- 去掉 boilerplate , 20 行
- Negative weighted cycle
  - 练习题
- Simplicity
  - debug a performance problem?





# Triangle Count, Page Rank ...

- $A^*$

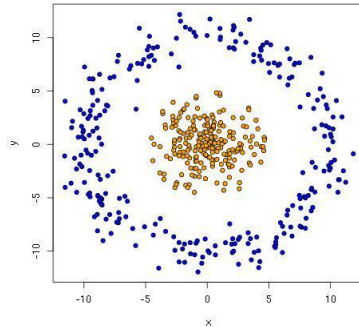
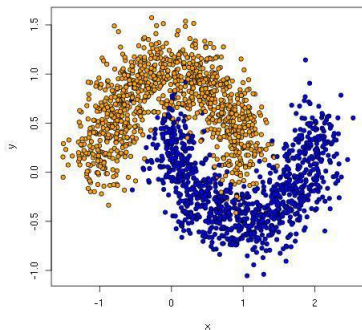
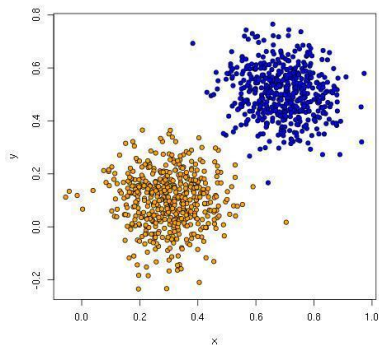






# 例子：Learning/Tagging

- Tensorflow
- <http://bcomposes.com/2015/11/26/simple-end-to-end-tensorflow-examples/>





# 实现

- CSV -> 数据库
  - categorical
- 并行





# TensorFlow

```
WITH EVAL AS (  
  Select transducer_col_int4(1) as prediction, 1  
         Transducer_col_int4(2) as tag,  
         Transducer_col_float4(3) as x,  
         Transducer_col_float4(4) as y,  
         Transducer($PHIExec python 2  
         ... boilerplate code ...
```

```
Def nextbatch(batch): 3  
  While true:  
    If cnt == BATCH_SIZE:  
      Break  
    Rec = NextInput()  
    If not rec:  
      Break  
    Cnt += 1  
    batch.append(rec)
```

```
... same tensorflow code ...  
... use nextbatch() to get input data ...  
... use WriteOutput() ...  
... to feed data backed to database ...
```

```
$$), data.*
```

```
FROM (  
  Select tag,  
         Case when cat = 'linear' then 1.0 4  
              When cat = 'moon' then 2.0  
              When cat = 'saturn' then 3.0  
         End,  
         X, y from points  
) data  
)  
Select * from EVAL where prediction <> tag 5
```

Program Listing 4. Use tensorflow to do prediction





# BigData As A Service

- Data-as-a-service
- Algorithm-as-a-service





# Sample

- 最常用工具，但不好做
- `SELECT * FROM T LIMIT SAMPLE x PERCENT;`
- `SELECT * FROM T LIMIT SAMPLE x ROWS;`



Thanks!

