# DOT: 一个开发处理大数据软件的分析模型

## 张晓东

## 美国俄亥俄州立大学

Team Members

**Yin Huai**[1], Rubao Lee[1],
Simon Zhang[2], Cathy H. Xia[1]

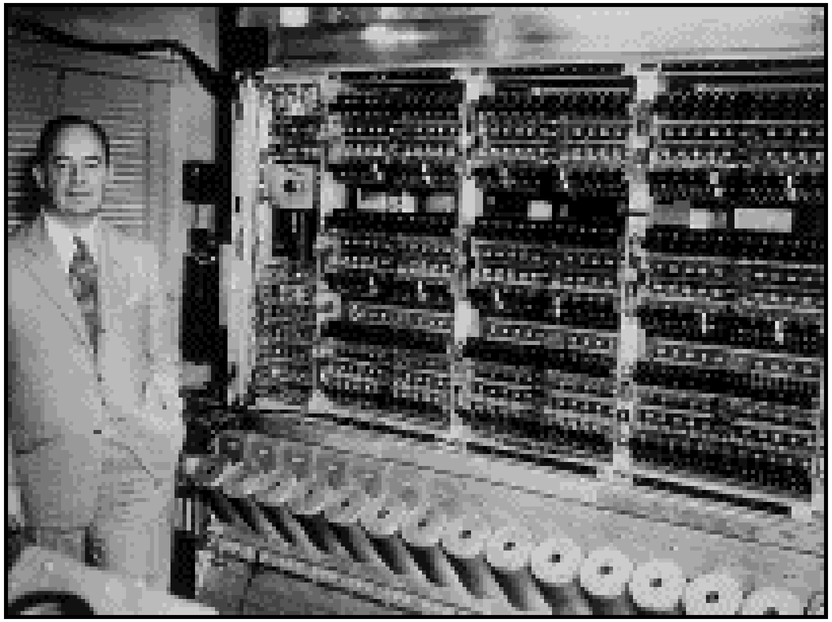
[1]Dept. of Computer Sci. & Eng., The Ohio State University

[2]Department of Computer Science, Cornell University

# The Evolution of Computing

- **Centralized computing era**
  - The **Von Neumann model** (1945): A baseline model to guide centralized computer architecture design
- **Parallel computing era**
  - The Bulk Synchronous Parallel (**BSP**) model (Leslie Valiant, 1990): A "scale-up" model to guide parallel architecture improvement and software optimization software for HPC
- **"The data center as a computer" era**
  - Only Software frameworks for big data analytics available
    - MapReduce, Hadoop, Dryad and several others
  - **No models yet**

# Von Neumann Model: Computer Architecture Design



Von Neumann model

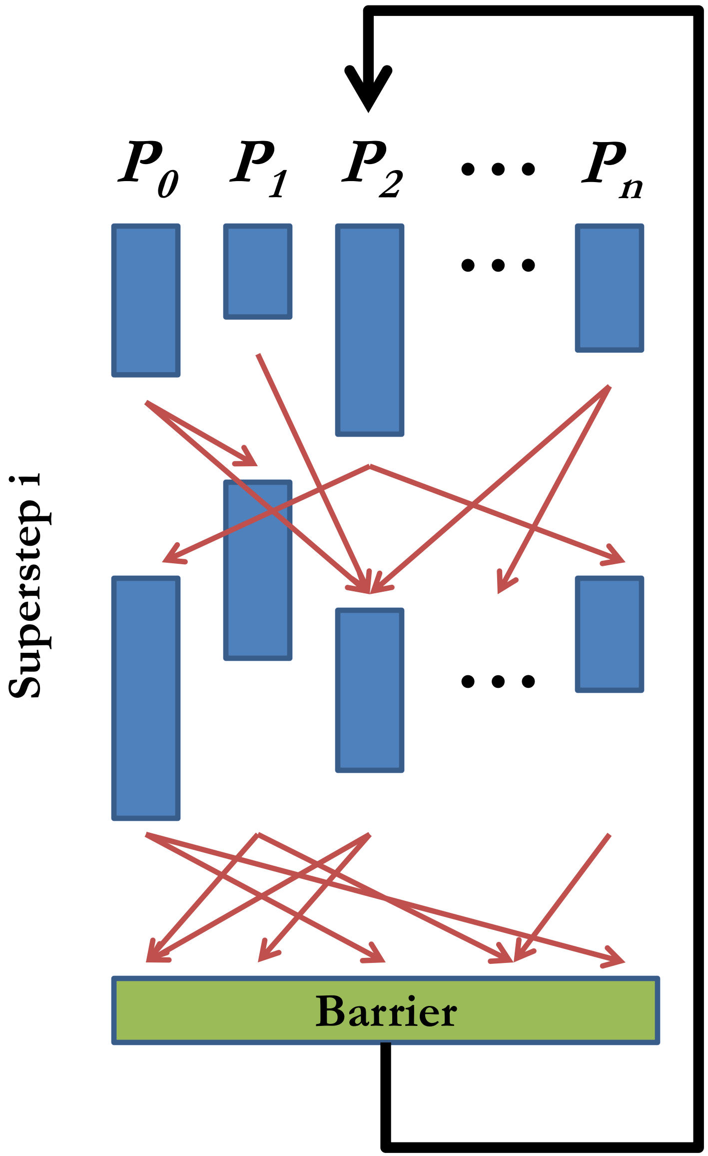
- a memory containing both data and instructions
- a computing unit for both arithmetic and logical operations
- a control unit to interpret instructions and make actions

Before Von Neumann's computer project, several operational computers were built:

- 1936, Zuse's Z1 (1st binary ditital computer) in Germany
- 1937, Atanasoff and Berry's ABC (1st electronic digital computer) in Iowa State University
- 1943, ENIAC based on ABC in UPenn

The most important milestone Von Neumann left is his paper: "First Draft of a Report to the EDVAC", 1945. (a consulting report to US Army) before his IAS Computer Project

# BSP is a Scale-Up Model for HPC

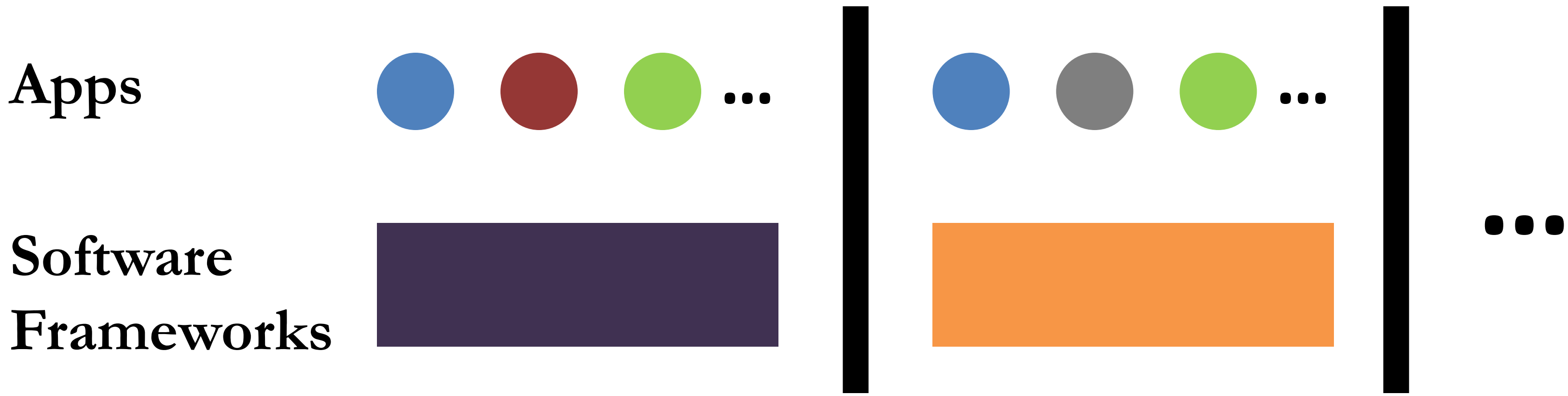



- ❑ **A parallel architecture model**
  - **Key parameters:** p, node speed, message speed, synch latency
  - **Low ratio of computing/message** is key
- ❑ **A programming model**
  - **Reducing message/synch latency** is key:
    - Overlapping computing and communication
    - Exploiting locality
    - Load balance to minimize synch latency
- ❑ **A cost model**
  - Combining both hardware/software parameters, we can predict **execution time**
- ❑ **BSP does not support**
  - **Data-intensive applications, big data**
  - **hardware independent performance**
  - **sustained scalability and high throughput**

# Scale-out is the Foundation for Big Data Analytics

- **Scale-out = sustained throughput growth as # nodes grows**
  - MR processed **1 PB** data in 6h 2m on **4000 nodes** in 11/2008
  - MR processed **10 PB** in 6 h 27 m on **8000 nodes** 9/2011 (VLDB'11)
  - The **data is not movable** after it is placed in a system
- **Existing big data software is in a scale-out mode by**
  - Focusing on scalability and fault-tolerance in large systems
  - Provide a easy-to-programming environment
- **Effectively responds urgent big data challenges**
  - Parallel databases with limited scalability cannot handle big data
  - Big data demands a large scope of data analytics
  - Traditional database business models are not affordable
  - Scale-out model is required for big data analytics

# Existing Software Practice for Big Data Analytics




- **For a given framework, is it scalable and fault-tolerant?**
  - What is the basis and principle of scalability and fault-tolerance ?

- **A unified model for big data analytics is needed**
  - Define a common distributed environment
  - Abstract the processing paradigm (basic behaviors of computing and communication) in a scalable and fault-tolerant manner (sufficient condition)

# Outline

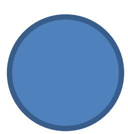
- Introduction
- The DOT Model
- General Optimization Rules
- Effectiveness of the DOT Model: Case Studies
  - Compare MapReduce and Dryad
  - Query optimization through the DOT model
- Conclusion

# An Overview of DOT Model

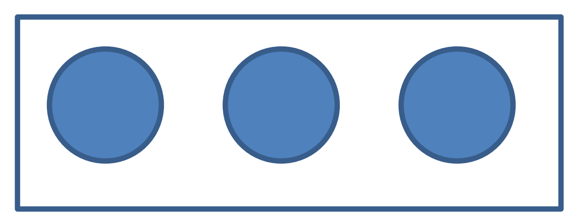
- **DOT is represented by**
  - **D:** distributed data sets
  - **O:** concurrent data processing operations
  - **T:** data transformations

- **The DOT model consists three components to describe a big data analytics job**

1: **An elementary DOT block**
(a root building block)



2: **A composite DOT block**
(an extended building block)

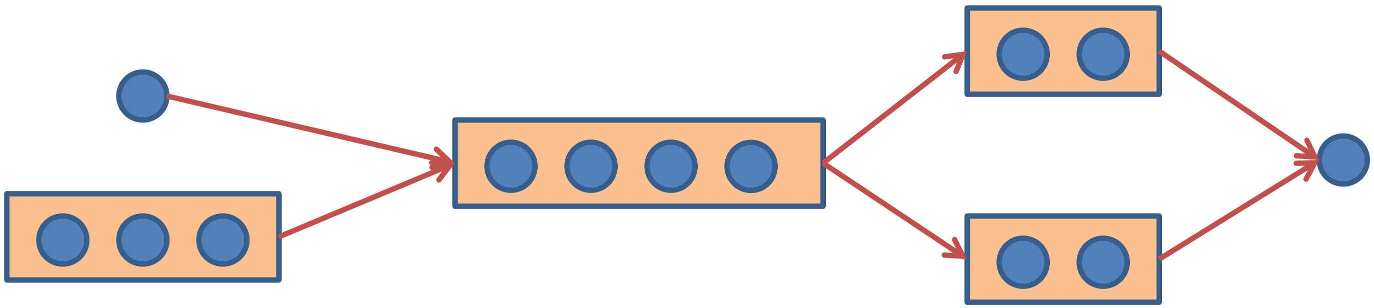


# An Overview of DOT Model

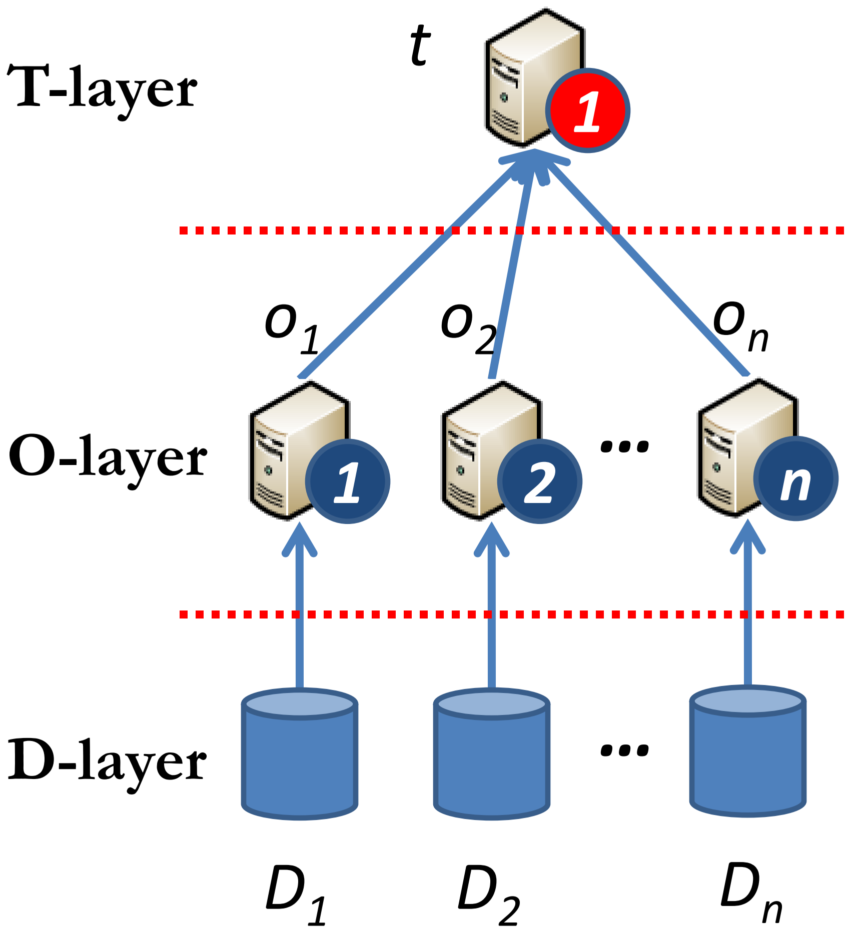
- **DOT is represented by**
  - **D:** distributed data sets
  - **O:** concurrent data processing operations
  - **T:** data transformations

- **The DOT model consists three components to describe a big data analytics job**

3: **A method to describe execution/dataflow DOT blocks.**



# An Elementary DOT block




**Phase 2:**

**1** worker collects all intermediate results and transforms them to final results (based on operator **t**)

**Phase 1:**

**n** workers perform concurrent operations (operator $o_1$ to $o_n$).

**No dependency among workers**

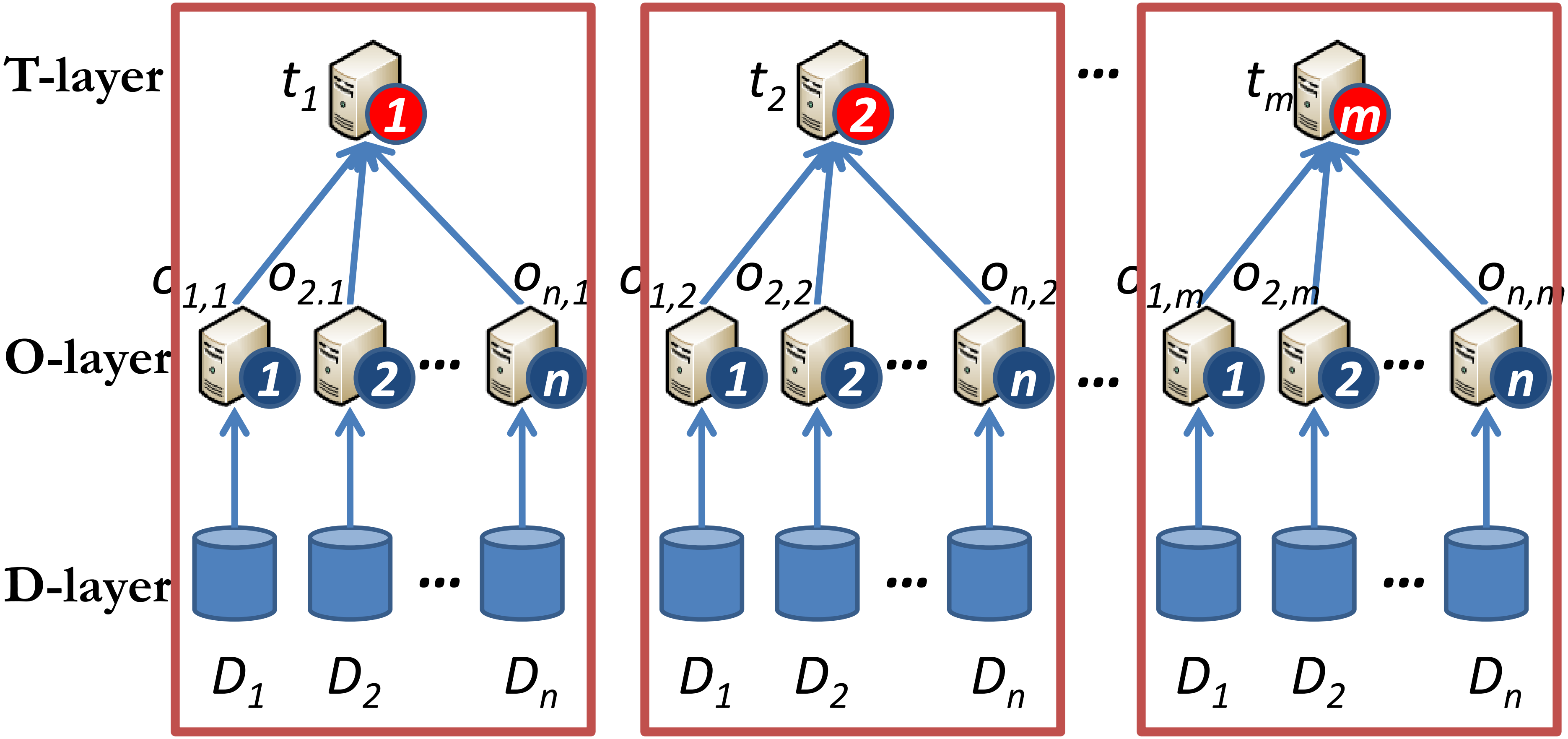
Data is divided into **n** sub-datasets ($D_1$ to $D_n$) and distributed among workers. We call a sub-dataset a **chunk**

# A scale-out action: a Composite DOT block

- **An elementary DOT is a starting point of scale-out**
  - The *n* data sets can be processed by **multi-groups** of workers
    - to maximize parallelism. e.g. calculate the sum of odd and even numbers from a data set: two concurrent groups of *n* workers
  - The size of intermediate results can be distributed
    - The intermediate data is too large to fit into a single worker
- **A composite DOT block is created**
  - Multiple independent elementary DOT blocks are organized to form a composite DOT block

# A Composite DOT block

m independent elementary DOT blocks are organized as a composite DOT block



# DOT Expression: a Representation of Dataflow

## ❑ Basic Functions and Operators

- **Operands:** input and output data vectors
- The data vector result of a DOT block can be the input vector of another DOT block
- $o_i(D_i)$: in an elementary DOT block, apply operator $o_i$ on chunk $D_i$ and return the transformed chunk
- $\bigsqcup_{i=1}^{n}(o_i(D_i))$ : in an elementary DOT block, collect $n$ chunks to form a data collection of $(o_1(D_1), \cdots, o_n(D_n))$
- $\biguplus_{j=1}^{m}(\vec{D}O_jT_j)$ : form a composite DOT block from multiple elementary DOT block
- $\bigoplus$ : form a data vector from operands
- $\biguplus$ : form a data vector and remove duplicated chunks

# The Matrix Representation

**D-layer** **O-layer** **T-layer**

$$\vec{DOT} = \begin{bmatrix} D_1 & D_2 & \cdots & D_n \end{bmatrix} \begin{bmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,m} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ o_{n,1} & o_{n,2} & \cdots & o_{n,m} \end{bmatrix} \begin{bmatrix} t_1 & 0 & \cdots & 0 \\ 0 & t_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_m \end{bmatrix}$$

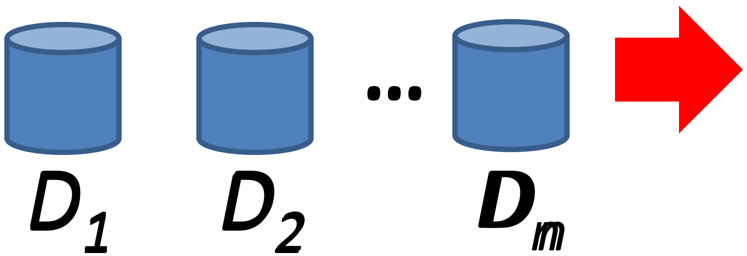
$$= \begin{bmatrix} \bigsqcup_{i=1}^{n} (o_{i,1}(D_i)) & \bigsqcup_{i=1}^{n} (o_{i,2}(D_i)) & \cdots & \bigsqcup_{i=1}^{n} (o_{i,m}(D_i)) \end{bmatrix} \begin{bmatrix} t_1 & 0 & \cdots & 0 \\ 0 & t_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_m \end{bmatrix}$$

## ❑ Three benefits of the matrix representation

- Concurrent data processing and data movement are explicitly represented by the "**matrix multiplication**"
- The fact that **no dependency** among workers in O-layer/T-layer is explicitly represented (column/row workers)
- The **optimization opportunities** are explicitly represented (talk about it latter)

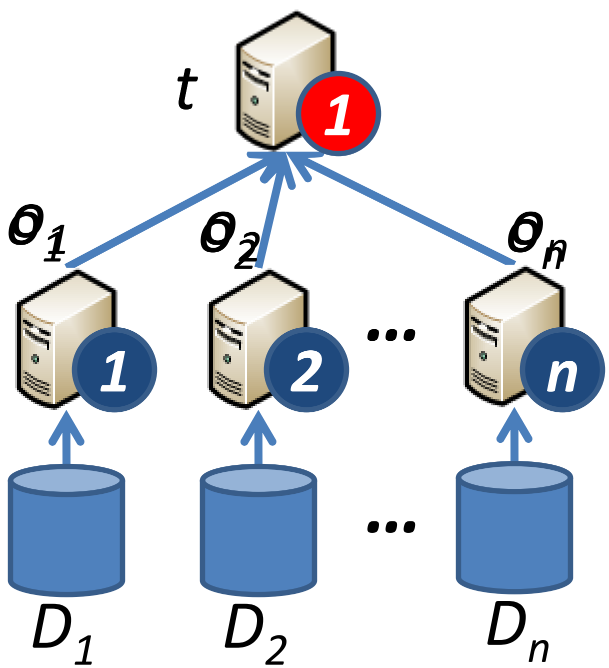
# Matrix Representation of an Elementary DOT Block

- A data set is represented by a data vector




$$\vec{D} = [\, D_1 \quad D_2 \quad \cdots \quad D_n \,]$$

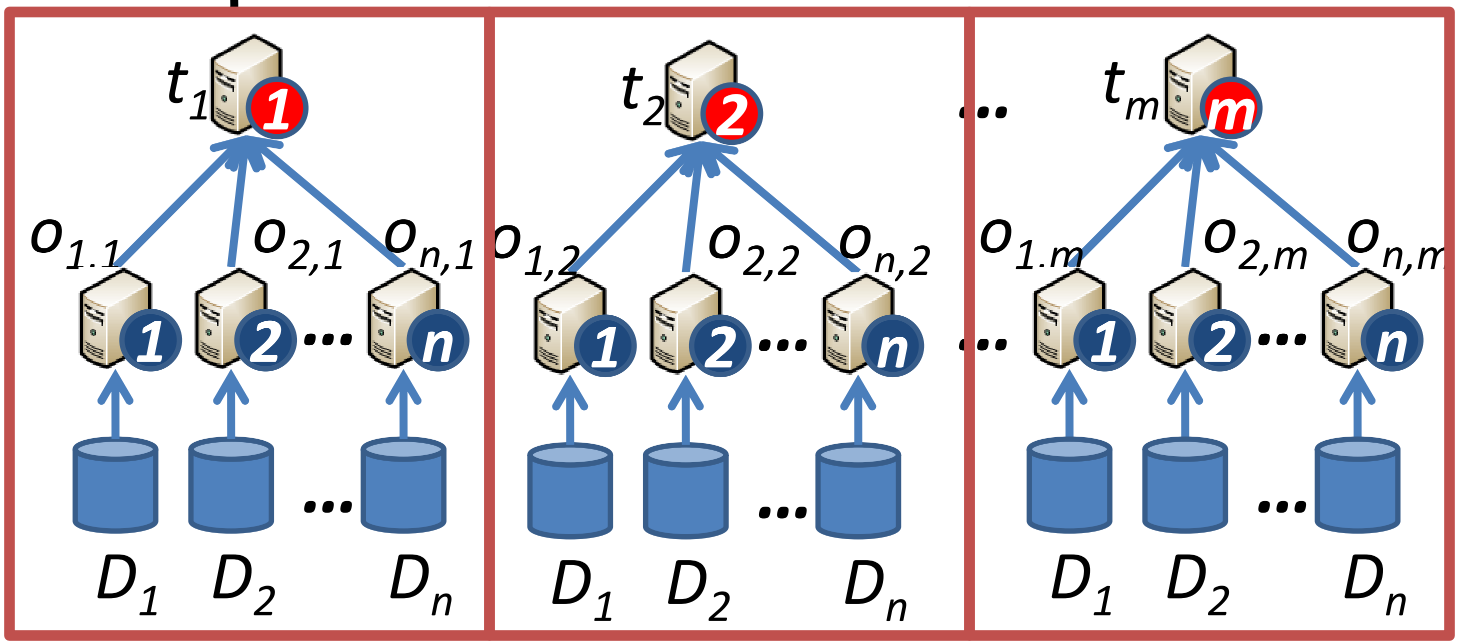
- The elementary DOT block




$$\vec{D}OT = [\; \;]$$

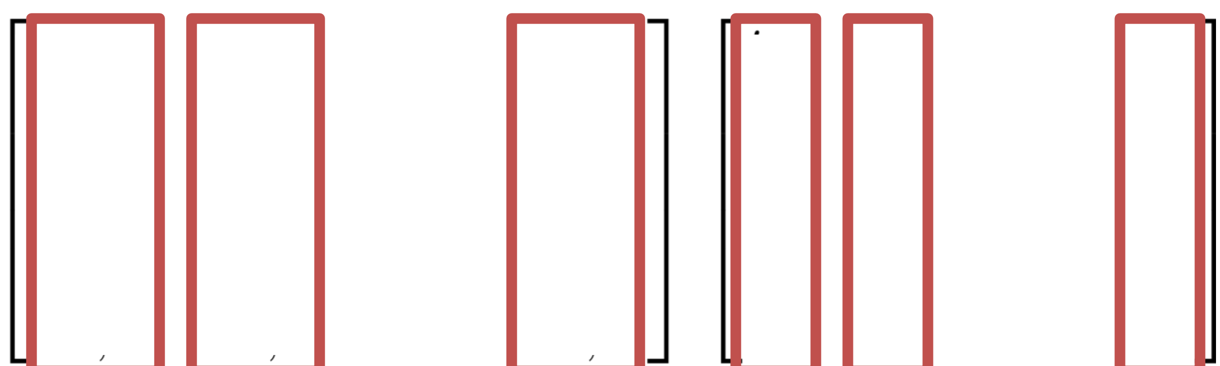
# Matrix Representation of a Composite DOT Block

- The composite DOT block




$\vec{D}O_{composite}T_{composite}$

$= [ \quad ]$



# A Big Data Analytics Job

## ❑ Dataflow

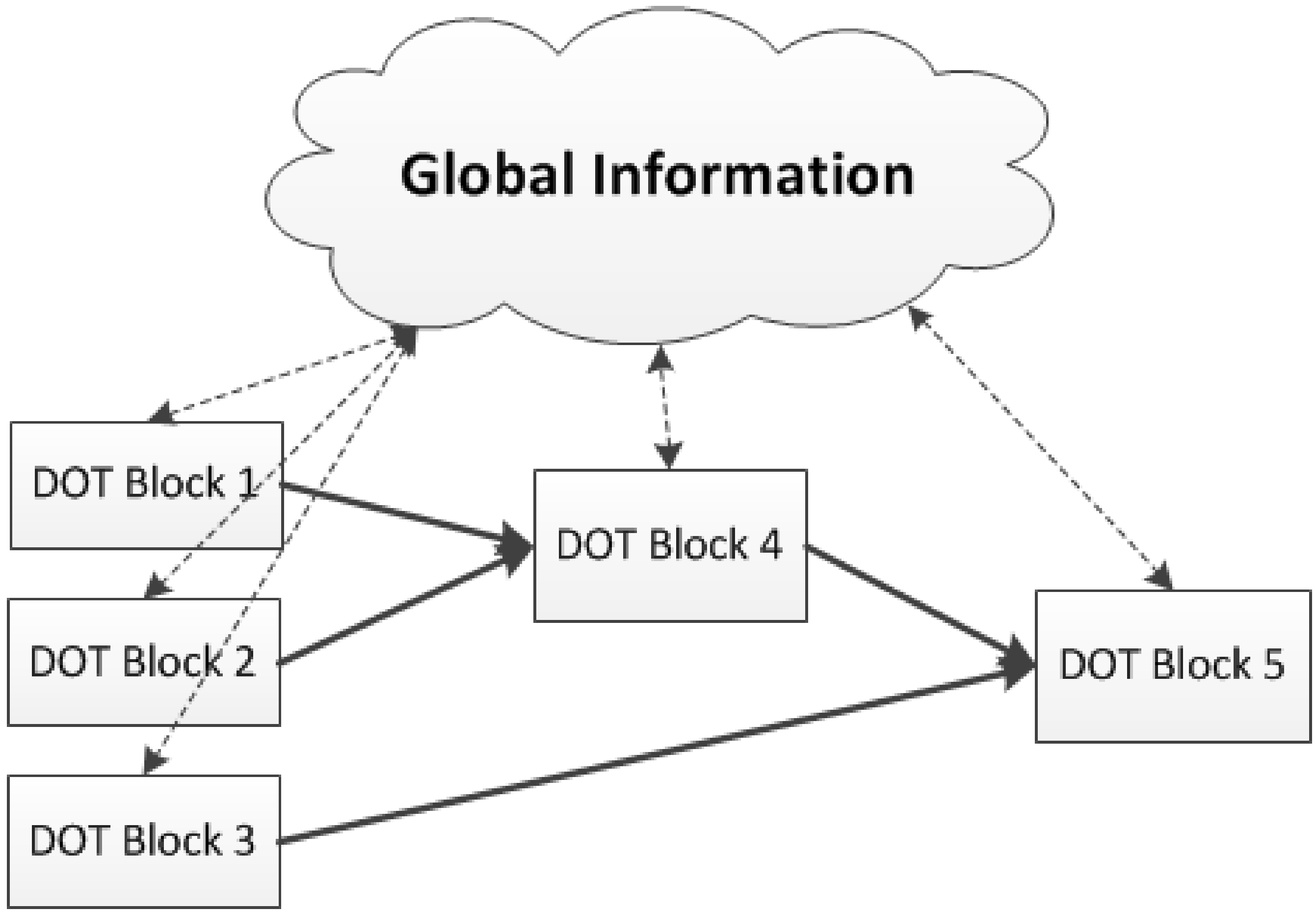
- Described by the combination of a specific or non-specific numbers of elementary /composite DOT blocks

## ❑ Two supporting components of the dataflow

- Global Information
  - Lightweight global information, e.g. job configurations
- Halting conditions
  - Used to determine under what conditions a job will stop
  - Have to be specified jobs described by a non-specific numbers of DOT blocks

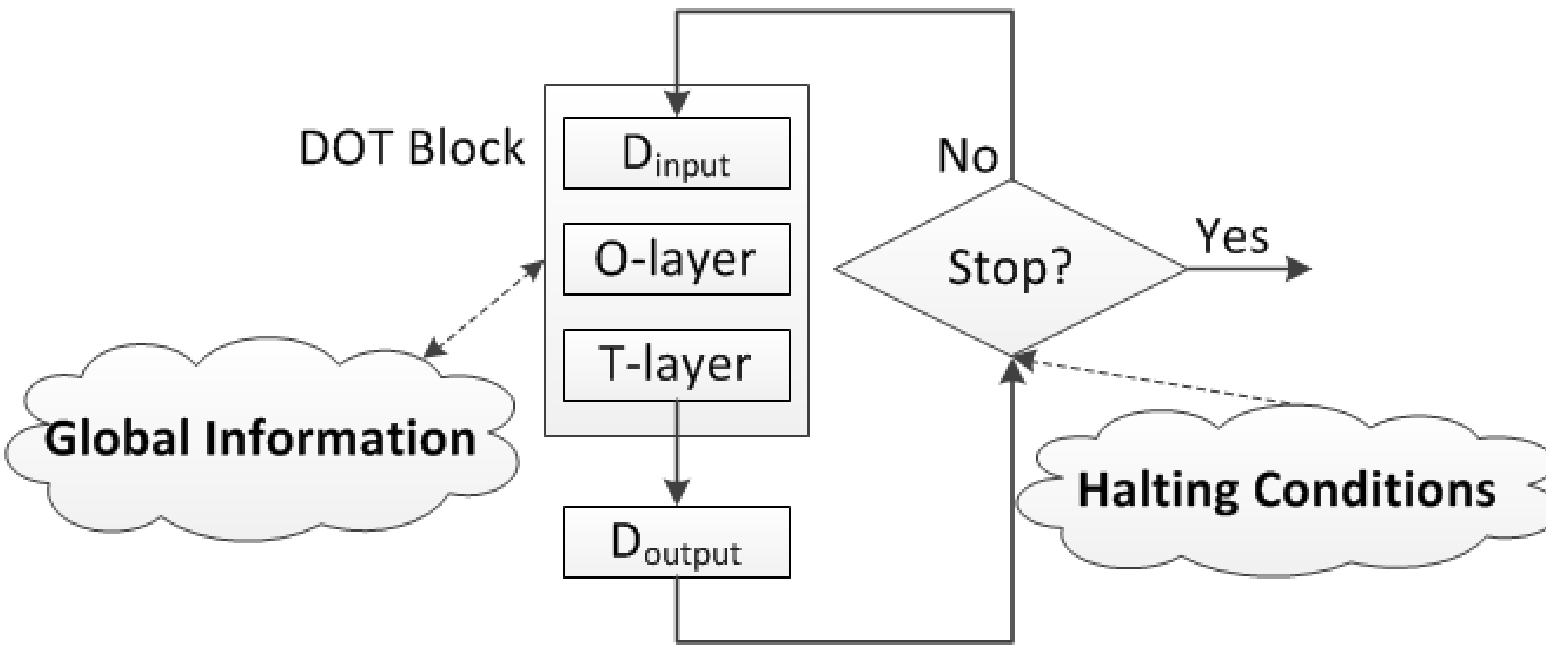
# A DOT Expression for Multiple Composite Blocks

$$((\vec{D}_1 O_1 T_1 \uplus \vec{D}_2 O_2 T_2) O_4 T_4 \oplus \vec{D}_3 O_3 T_4) O_5 T_5$$



# A DOT Expression for an Iterative Process

$$\vec{D}(t) = \vec{D}(t-1)O(t)T(t)$$



# Scalability and Fault-tolerance

- **Two different definitions**
  - Scalability and fault-tolerance of **a job**
  - Scalability and fault-tolerance of **a processing paradigm**
- **The processing paradigm of the DOT model is scalable and fault-tolerant** (proofs in SOCC'11)
- **A sufficient condition:**
  - a framework can be represented by the DOT model, i.e. any job of this framework can be represented by the DOT model => the processing paradigm of this framework is scalable and fault-tolerant
  - Proofs in SOCC'11

# Outline

- ❑ Introduction
- ❑ The DOT Model
- ❑ General Optimization Rules
- ❑ **Effectiveness of the DOT Model: Case Studies**
  - Comparison between MapReduce and Dryad
  - Query optimization through the DOT model
- ❑ **Conclusion**

# Optimization Rules

- **Substituting Expensive Remote Data Transfers with Low-Cost Local Computing**
  - Unnecessary data transfers are reduced
- **Exploiting Sharing Opportunities**
  - Sharing common chunks
    - Unnecessary data scan is eliminated
  - Sharing common operations
    - Unnecessary operations on the data is eliminated
- **Exploiting the Potential of Parallelism**
  - Unnecessary data materialization is eliminated
- **All of these optimization opportunities are explicitly represented by the DOT matrix representation**

# Exploiting Sharing Opportunities

## ❑ Sharing common data chunks

- Unnecessary data scan is eliminated

$$\left(\begin{bmatrix} A & B \end{bmatrix} \oplus \begin{bmatrix} A & C \end{bmatrix}\right) \begin{bmatrix} \alpha & o_{1,2} & 0 & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$

$$\begin{bmatrix} A & B & C \end{bmatrix} \begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$

# Exploiting Sharing Opportunities

## ❑ Sharing common operations

- Unnecessary operations on the data is eliminated

$$
\begin{bmatrix} A & B & C \end{bmatrix}
\begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix}
\begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}
$$

$$
\begin{bmatrix} A & B & C \end{bmatrix}
\begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ o_{4,3} & 0 & o_{4,3} & o_{4,4} \end{bmatrix}
\begin{bmatrix} \beta & 0 & 0 \\ 0 & t_2 & 0 \\ 0 & 0 & t_4 \end{bmatrix}
$$

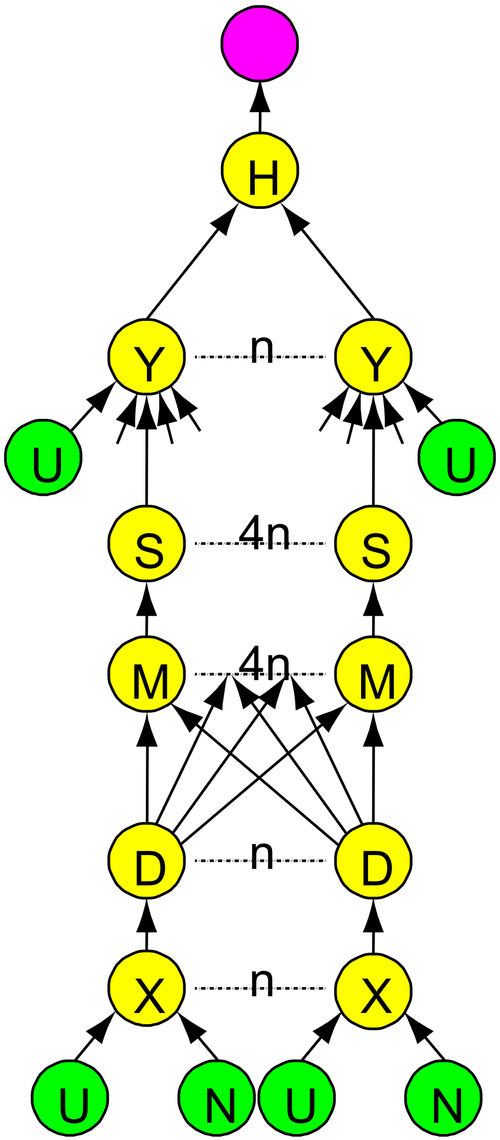
# Outline

- Introduction
- The DOT Model
- General Optimization Rules
- **Effectiveness of the DOT Model: Case Studies**
  - Comparison between MapReduce and Dryad
  - Query optimization through the DOT model
- Conclusion

# Representing A Dryad Job by the DOT model

- **A Dryad job is represented by a directed acyclic graph (DAG)**
- **Represent a Dryad job in the DOT model**
  - A method based on graph search
  - A Dryad job is represented by a DOT expression

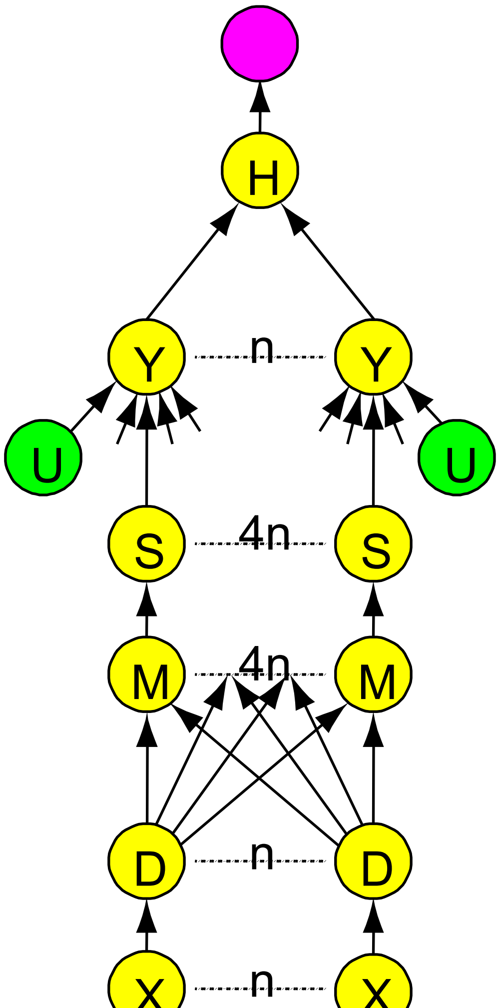
# Representing A Dryad Job by the DOT model




$$
\begin{aligned}
Job = (&\vec{D}_{U,N} O_{1,2n\times n} T_{1,n\times n} \\
&O_{2,n\times 4n} T_{2,4n\times 4n} \oplus \vec{D}_U) \\
&O_{3,5n\times n} T_{3,n\times n} O_{4,n\times 1} T_{4,1\times 1}
\end{aligned}
$$

[Michael Isard et al. EuroSys 2007 ]

# Representing A Dryad Job by the DOT model




n=2

$$\left([U_1 \quad U_2 \quad N_1 \quad N_2] \begin{bmatrix}1 & 0\\ 0 & 1\\ 1 & 0\\ 0 & 1\end{bmatrix} \begin{bmatrix}X & 0\\ 0 & X\end{bmatrix} \begin{bmatrix}D_1 & \cdots & D_8\\ D_1 & \cdots & D_8\end{bmatrix} \begin{bmatrix}S(M) & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & S(M)\end{bmatrix} \oplus [U_1 \quad U_2]\right) \begin{bmatrix}1 & 0\\ \vdots & \vdots\\ 1 & 0\\ 0 & 1\\ \vdots & \vdots\\ 0 & 1\end{bmatrix} \begin{bmatrix}Y & 0\\ 0 & Y\end{bmatrix} \begin{bmatrix}1\\ 1\end{bmatrix} [H]$$

**A DOT expression can be executed by Dryad (details in SOCC'11)**

# Representing A MapReduce Job by the DOT model

## ❑ A MapReduce job

- Map function: $o_{map}$
- Reduce function: $t_{reduce}$
- Partitioning function: $p$
  - $p_i$: get the intermediate results that will be sent to reducer i

$$\vec{D}OT = \begin{bmatrix} D_1 & \cdots & D_n \end{bmatrix} \begin{bmatrix} p_1(o_{map}) & \cdots & p_m(o_{map}) \\ \vdots & \ddots & \vdots \\ p_1(o_{map}) & \cdots & p_m(o_{map}) \end{bmatrix} \begin{bmatrix} t_{reduce} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & t_{reduce} \end{bmatrix}$$

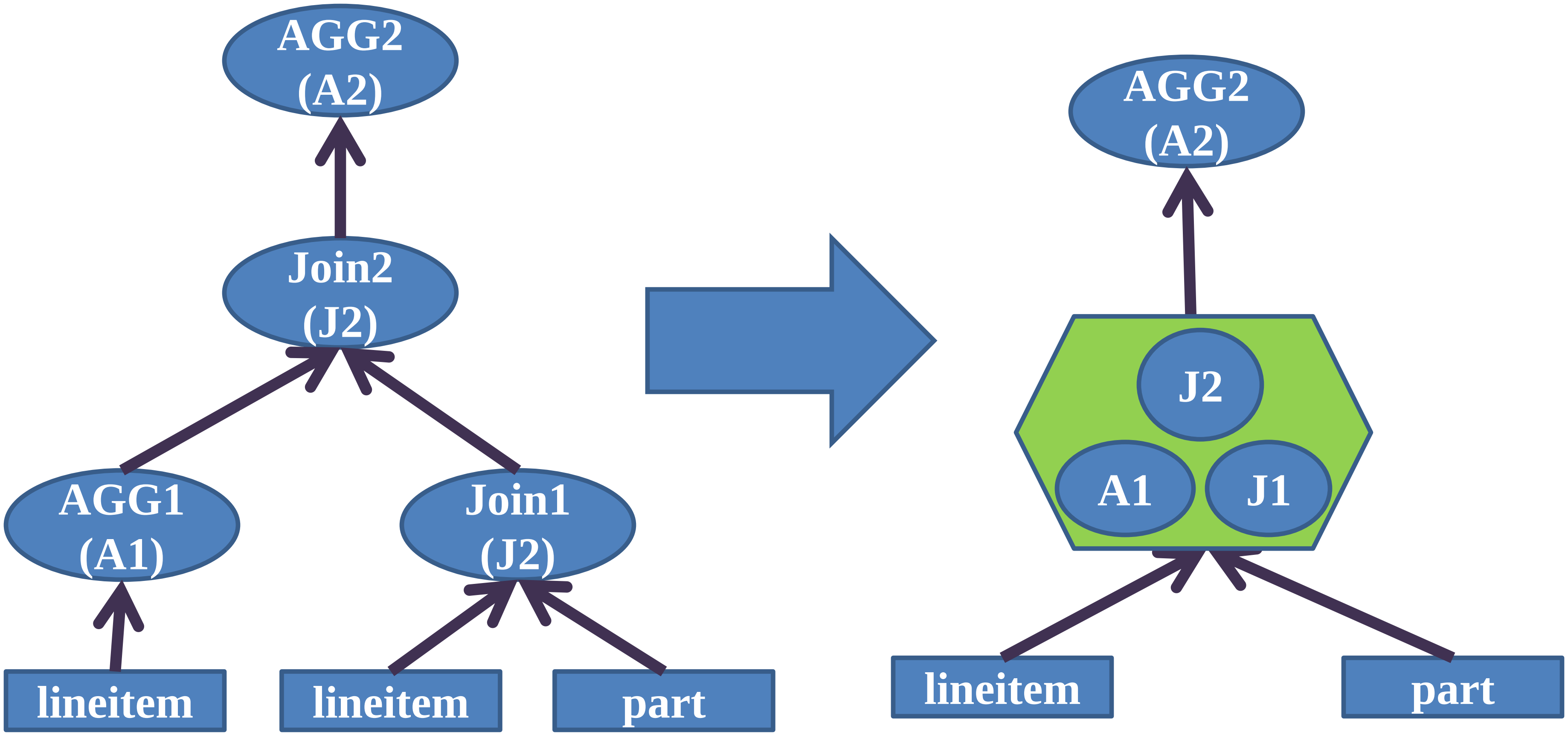
**A DOT expression can be executed by MapReduce (details in SOCC'11)**

# A Comparison between MapReduce and Dryad

- ❑ Every MapReduce/Dryad job can be represented by the DOT model
  - ➢ The processing paradigms of MapReduce and Dryad are scalable and fault-tolerant
- ❑ A DOT expression can be executed by MapReduce/Dryad
  - ➢ Comparing basic behaviors of computing and communication (processing paradigm), Dryad and MapReduce do not have fundamental difference.

# Query Optimization in Hive: TPC-H Q17

- Optimizing the performance of TPC-H Q17 on the MapReduce framework




4 MapReduce jobs

2 MapReduce jobs

# DOT Representation of TPC-H Q17 in Hive

$$Q17 = ((\vec{D}_L O_{A1} T_{A1} \oplus (\vec{D}_L \oplus \vec{D}_P) O_{J1} T_{J1}) O_{J2} T_{J2}) O_{A2} T_{A2}$$

$$= (([D_{L,1} \quad D_{L,2}] \begin{bmatrix} p_{A1,1}(o_{A1,1,1}) & p_{A1,2}(o_{A1,1,2}) \\ p_{A1,1}(o_{A1,2,1}) & p_{A1,2}(o_{A1,2,2}) \end{bmatrix} \begin{bmatrix} t_{A1,1} & 0 \\ 0 & t_{A1,2} \end{bmatrix} \oplus$$

$$([D_{L,1} \quad D_{L,2}] \oplus [D_{P,1} \quad D_{P,2}]) \begin{bmatrix} p_{J1,1}(o_{J1,1,1}) & p_{J1,2}(o_{J1,1,2}) \\ p_{J1,1}(o_{J1,2,1}) & p_{J1,2}(o_{J1,2,2}) \\ p_{J1,1}(o_{J1,3,1}) & p_{J1,2}(o_{J1,3,2}) \\ p_{J1,1}(o_{J1,4,1}) & p_{J1,2}(o_{J1,4,2}) \end{bmatrix} \begin{bmatrix} t_{J1,1} & 0 \\ 0 & t_{J1,2} \end{bmatrix})$$

$$\begin{bmatrix} p_{J2,1}(o_{J2,1,1}) & 0 \\ 0 & p_{J2,2}(o_{J2,2,2}) \\ p_{J2,1}(o_{J2,3,1}) & 0 \\ 0 & p_{J2,2}(o_{J2,4,2}) \end{bmatrix} \begin{bmatrix} t_{J2,1} & 0 \\ 0 & t_{J2,2} \end{bmatrix})$$

$$\begin{bmatrix} p_{A2,1}(o_{A2,1,1}) & p_{A2,2}(o_{A2,1,2}) \\ p_{A2,1}(o_{A2,2,1}) & p_{A2,2}(o_{A2,2,2}) \end{bmatrix} \begin{bmatrix} t_{A2,1} & 0 \\ 0 & t_{A2,2} \end{bmatrix}$$

$p_{A1,1}=p_{J1,1}=p_{J2,1}$

$p_{A1,2}=p_{J1,2}=p_{J2,2}$

# Query Optimization by DOT for TPC-H Q17 in Hive

$$Q17 = (\vec{D}_{P,L} O_{A1,\,J1} T_{A1,\,J1}) O'_{J2} T_{J2}) O_{A2} T_{A2}$$

$$= ((\begin{bmatrix} D_{L,\,1} & D_{L,\,2} & D_{P,\,1} & D_{P,\,2} \end{bmatrix} \begin{bmatrix} p_1(o_{A1,1,1},\, o_{J1,1,1}) & p_2(o_{A1,1,2},\, o_{J1,1,2}) \\ p_1(o_{A1,2,1},\, o_{J1,2,1}) & p_2(o_{A1,2,2},\, o_{J1,2,2}) \\ p_1(o_{J1,3,1}) & p_2(o_{J1,3,2}) \\ p_1(o_{J1,4,1}) & p_2(o_{J1,4,2}) \end{bmatrix}$$

$$\begin{bmatrix} (t_{A1,1}, t_{J1,1}) & 0 \\ 0 & (t_{A1,2}, t_{J1,2}) \end{bmatrix}) \begin{bmatrix} (o_{J2,1,1}, o_{J2,3,1}) & 0 \\ 0 & (o_{J2,2,2}, o_{J2,4,2}) \end{bmatrix} \begin{bmatrix} t_{J2,1} & 0 \\ 0 & t_{J2,2} \end{bmatrix})$$

$$\begin{bmatrix} p_{A2,1}(o_{A2,1,1}) & p_{A2,2}(o_{A2,1,2}) \\ p_{A2,1}(o_{A2,2,1}) & p_{A2,2}(o_{A2,2,2}) \end{bmatrix} \begin{bmatrix} t_{A2,1} & 0 \\ 0 & t_{A2,2} \end{bmatrix}$$

**It is a diagonal Matrix.**
**Merge this DOT block into the T matrix of the previous one**

# Final Form of Query Optimization by DOT for TPC-H Q17 in Hive

$$
\begin{aligned}
Q17 =& (\vec{D}_{P,L} O_{A1,\,J1,\,J2} T_{A1,\,J1,\,J2}) O_{A2} T_{A2} \\
=& ( \begin{bmatrix} D_{L,\,1} & D_{L,\,2} & D_{P,\,1} & D_{P,\,2} \end{bmatrix} \begin{bmatrix} p_1(o_{A1,1,1},\, o_{J1,1,1}) & p_2(o_{A1,1,2},\, o_{J1,1,2}) \\ p_1(o_{A1,2,1},\, o_{J1,2,1}) & p_2(o_{A1,2,2},\, o_{J1,2,2}) \\ p_1(o_{J1,3,1}) & p_2(o_{J1,3,2}) \\ p_1(o_{J1,4,1}) & p_2(o_{J1,4,2}) \end{bmatrix} \\
& \begin{bmatrix} t_{J2,1}((o_{J2,1,1}, o_{J2,3,1})(t_{A1,1}, t_{J1,1})) & \\ 0 & t_{J2,2}((o_{J2,2,2}, o_{J2,4,2})(t_{A1,2}, t_{J2,2})) \end{bmatrix} ) \\
& \begin{bmatrix} p_{A2,1}(o_{A2,1,1}) & p_{A2,2}(o_{A2,1,2}) \\ p_{A2,1}(o_{A2,2,1}) & p_{A2,2}(o_{A2,2,2}) \end{bmatrix} \begin{bmatrix} t_{A2,1} & 0 \\ 0 & t_{A2,2} \end{bmatrix}
\end{aligned}
$$

With this optimization, we got more than **2x speedup** in our large-scale experiments.

For details, please refer **YSmart** patched in Hive [ICDCS 2011]

# Conclusion

- **DOT is an unified model for big data analytics in distributed systems**
- **Its matrix format and related analysis provide**
  - A sufficient condition of scalability and fault-tolerance of a processing paradigm
  - A set of optimization rules for applications on various software frameworks with analytical basis
  - A mathematical tool to fairly compare different software frameworks
- **To guide a simulation-based software design for big data analytics**
- **A bridging model for execution migration among different software frameworks**
- **References: RCFile (ICDE'11), YSmart (ICDCS'11), and DOT (SOCC'11)**

# What we will do next based on DOT?

- **A more rigorous math structure to gain more insights**
  - Other properties of scalability and fault tolerance
  - Finding necessary conditions
  - Correlating linear algebra theorems to various matrix representations of big data analytics jobs
  - A relaxed DOT model
- **Beyond machine-independent natural parallelism**
  - Building hidden (implicit) communication mechanisms in DOT
  - A DHT-based worker-mapping structure: group communication-related workers in a single node or a cluster of neighbor nodes
  - Physical node information will be build in the model
- **A DOT-based cost model**
  - To guide resource allocations and deployment of large distributed systems under different performance and reliability objectives

# 叔本华：抽象的价值在于它的普遍义

- 一切理性知识都是从直观知识中抽象而来的， 这是一切知识根源。
- 直观是一切真理的源泉，是一切科学的基础。

摘自叔本华（Arthur Schopenhauer, 1788-1860）《主观与客观的世界》（The World as Will and Representation)

**Thank You!**

# Backup

# An Algebra for Representing the Dataflow of a job

## Operand

- Data vectors (a DOT block is also a data vector)

## Operations on data vectors

$$\vec{D}_1 = \begin{bmatrix} D_{1,1} & D_{1,2} \end{bmatrix} \qquad \vec{D}_2 = \begin{bmatrix} D_{2,1} & D_{2,2} \end{bmatrix}$$

$\bigoplus$ $\quad \vec{D}_1 \oplus \vec{D}_2 = \begin{bmatrix} \vec{D}_1 & \vec{D}_2 \end{bmatrix} = \begin{bmatrix} D_{1,1} & D_{1,2} & D_{2,1} & D_{2,2} \end{bmatrix}$

$\biguplus$ $\quad$ if $D_{1,1} = D_{2,1} \quad \vec{D}_1 \uplus \vec{D}_2 = \begin{bmatrix} D_{1,1} & D_{1,2} & D_{2,2} \end{bmatrix}$

## Operations on DOT blocks

- Two direct-dependent DOT blocks: a DOT block is the data vector (input) of another DOT block

$$(\vec{D}_1 O_1 T_1) O_2 T_2$$

- Two independent DOT blocks

$$\vec{D}_1 O_1 T_1 \oplus \vec{D}_2 O_2 T_2 \qquad \vec{D}_1 O_1 T_1 \uplus \vec{D}_2 O_2 T_2$$

# Optimization Rules

- **Substituting Expensive Remote Data Transfers with Low-Cost Local Computing**
  - Unnecessary data transfers are reduced
- **Exploiting Sharing Opportunities**
  - Sharing common chunks
    - Unnecessary data scan is eliminated
  - Sharing common operations
    - Unnecessary operations on the data is eliminated
- **Exploiting the Potential of Parallelism**
  - Unnecessary data materialization is eliminated
- **All of these optimization opportunities are explicitly represented by the matrix representation**
  - See our paper for details

# Substituting Expensive Remote Data Transfers with Low-Cost Local Computing

## ❑ Substituting Expensive Remote Data Transfers with Low-Cost Local Computing

- In a DOT block, transfer computation in the matrix O (or T) to matrix T (or O) to reduce the amount of intermediate results

$$\begin{bmatrix} D_{1,1} & D_{1,2} \end{bmatrix} \begin{bmatrix} o_{1,1,1} & o_{1,1,2} \\ o_{1,2,1} & o_{1,2,2} \end{bmatrix} \begin{bmatrix} t_{1,1} & 0 \\ 0 & t_{1,2} \end{bmatrix}.$$

⬇ e.g. $t_{1,1}$ is summation operation $\sum$

$$\begin{bmatrix} D_{1,1} & D_{1,2} \end{bmatrix} \begin{bmatrix} \sum(o_{1,1,1}) & o_{1,1,2} \\ \sum(o_{1,2,1}) & o_{1,2,2} \end{bmatrix} \begin{bmatrix} \sum & 0 \\ 0 & t_{1,2} \end{bmatrix}.$$

# Exploiting Sharing Opportunities

## ❑ Sharing common chunks

- Unnecessary data scan is eliminated

$$\left(\begin{bmatrix} A & B \end{bmatrix} \oplus \begin{bmatrix} A & C \end{bmatrix}\right) \begin{bmatrix} \alpha & o_{1,2} & 0 & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$

$$\begin{bmatrix} A & B & C \end{bmatrix} \begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$

# Exploiting Sharing Opportunities

## ❑ Sharing common operations

- Unnecessary operations on the data is eliminated

$$
\begin{bmatrix} A & B & C \end{bmatrix}
\begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix}
\begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}
$$

$$
\begin{bmatrix} A & B & C \end{bmatrix}
\begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ o_{4,3} & 0 & o_{4,3} & o_{4,4} \end{bmatrix}
\begin{bmatrix} \beta & 0 & 0 \\ 0 & t_2 & 0 \\ 0 & 0 & t_4 \end{bmatrix}
$$

# Exploiting the Potential of Parallelism

## ❑ Merge two chained DOT blocks

- The condition: if either DOT block's matrix O is a diagonal matrix
- Unnecessary data materialization is eliminated

$$\begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} o_{1,1,1} & o_{1,1,2} \\ o_{1,2,1} & o_{1,2,2} \end{bmatrix} \begin{bmatrix} t_{1,1} & 0 \\ 0 & t_{1,2} \end{bmatrix} \begin{bmatrix} o_{2,1,1} & 0 \\ 0 & o_{2,2,2} \end{bmatrix} \begin{bmatrix} t_{2,1} & 0 \\ 0 & t_{2,2} \end{bmatrix}$$

$$\begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} o_{1,1,1} & o_{1,1,2} \\ o_{1,2,1} & o_{1,2,2} \end{bmatrix} \begin{bmatrix} t_{2,1}(o_{2,1,1}(t_{1,1})) & 0 \\ 0 & t_{2,2}(o_{2,2,2}(t_{1,2})) \end{bmatrix} \begin{bmatrix} t_{2,1} & 0 \\ 0 & t_{2,2} \end{bmatrix}$$

# Exploiting Sharing Opportunities

## ❑ Sharing common chunks

- Unnecessary data scan is eliminated

$$([A \quad B] \oplus [A \quad C]) \begin{bmatrix} \alpha & o_{1,2} & 0 & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$

$$[A \quad B \quad C] \begin{bmatrix} \alpha & o_{1,2} & \alpha & o_{1,4} \\ o_{2,1} & o_{2,2} & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & o_{4,3} & o_{4,4} \end{bmatrix} \begin{bmatrix} \beta & 0 & 0 & 0 \\ 0 & t_2 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & t_4 \end{bmatrix}$$