



pig



web-scale
processing

Christopher Olston and many others
Yahoo! Research



Example Data Analysis Task

Find users who tend to visit “good” pages.

Visits

user	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00

⋮

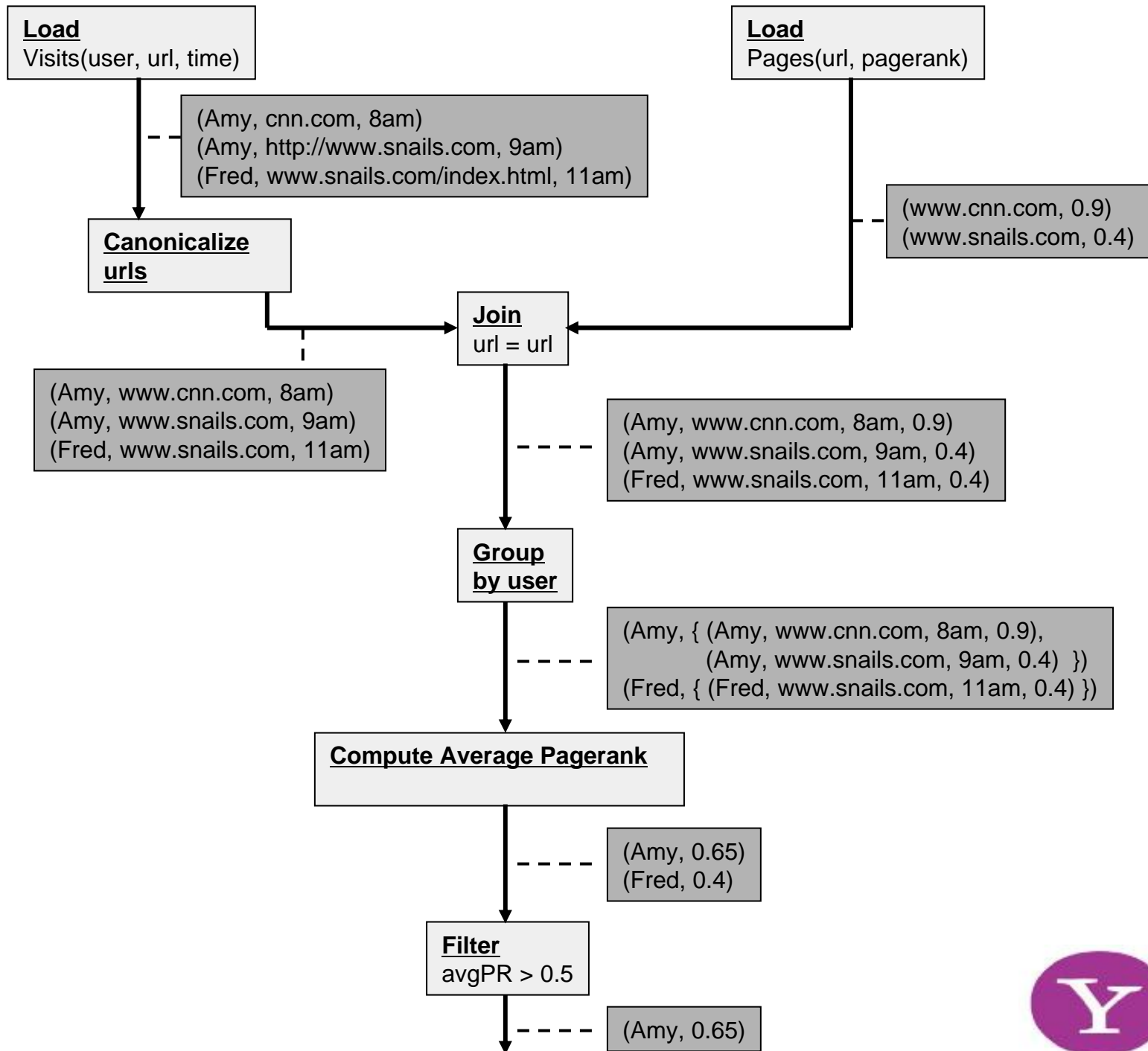
Pages

url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2

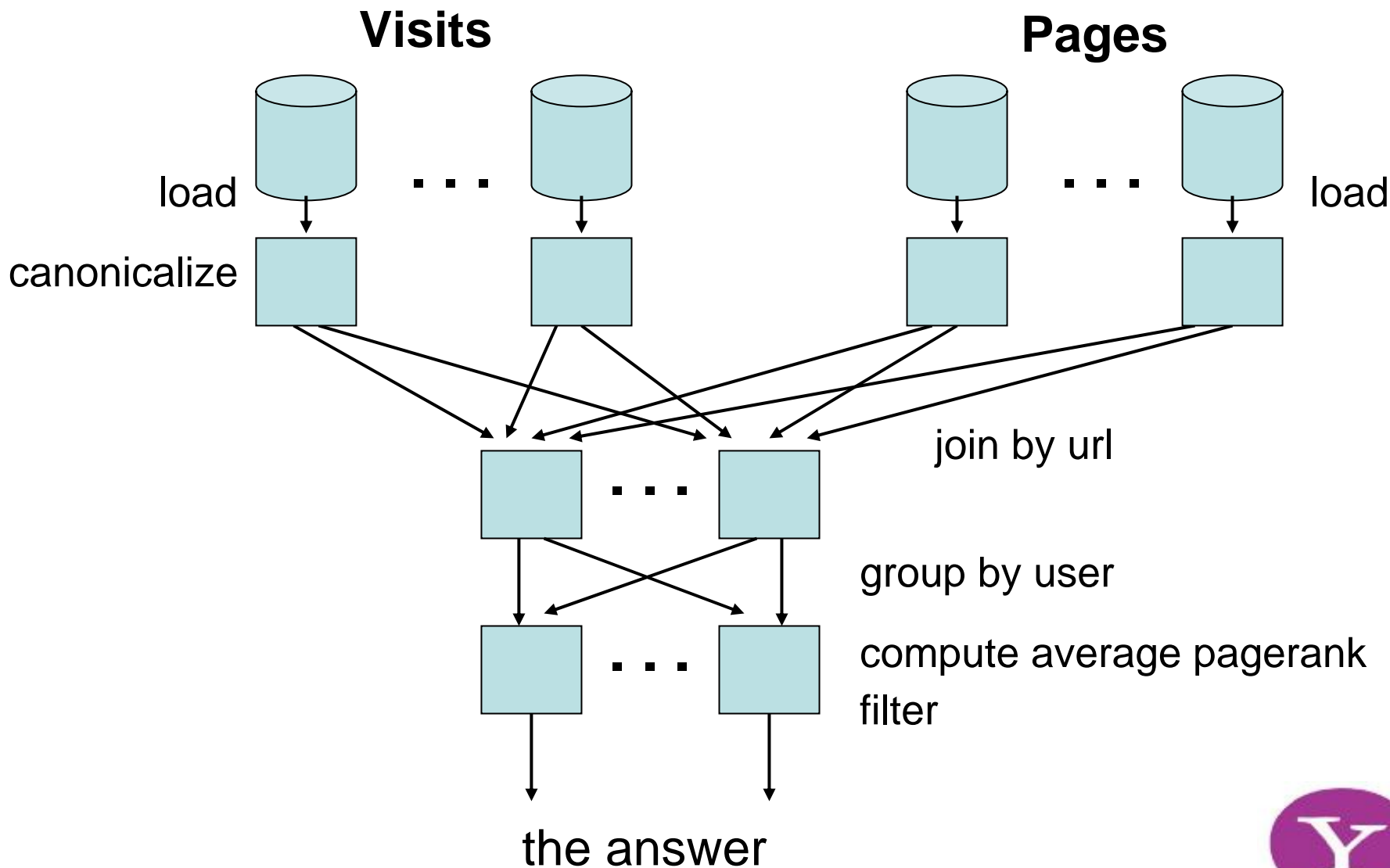
⋮



Conceptual Dataflow



System-Level Dataflow



Simple, right?

But ... using map-reduce:

- Write join code yourself
- Exploit data size, ordering properties
- Glue together 2 map-reduce jobs

⇒ Do low-level stuff by hand

⇒ Hard to understand, maintain code



Need a *Dataflow Language* + compiler into map-reduce

```
Visits = load '/data/visits' as (user, url, time);  
Visits = foreach Visits generate user, Canonicalize(url), time;
```

```
Pages = load '/data/pages' as (url, pagerank);
```

```
VP = join Visits by url, Pages by url;
```

```
UserVisits = group VP by user;
```

```
UserPageranks = foreach UserVisits generate user, AVG(VP.pagerank) as avgpr;
```

```
GoodUsers = filter UserPageranks by avgpr > '0.5';
```

```
store GoodUsers into '/data/good_users';
```



Pig Latin Dataflow Language

- transformations on sets of records
- **easy for users**
 - high-level, extensible data processing primitives
- **easy for the system**
 - exposes opportunities for parallelism and reuse

operators:

- FILTER
- FOREACH ... GENERATE
- GROUP

binary operators:

- JOIN
- COGROUP
- UNION



Related Languages

- **SQL**: declarative all-in-one blocks
- **NESL**: lacks join, cogroup
- **Map-Reduce**: special case of Pig Latin
- **Sawzall**: rigid map-then-reduce structure



Pig Latin vs. SQL

SQL

declarative (**what**, not **how**);
bundle many aspects into one statement

"I much prefer writing in Pig [Latin] versus SQL. The step-by-step method of creating a program in Pig [Latin] is much cleaner and simpler to use than the single block method of SQL. It is easier to keep track of what your variables are, and where you are in the process of analyzing your data."

-- *Jasmine Novak, Engineer, Yahoo!*

- semantic order of operations is obvious
- incremental construction
- debug by viewing intermediate results



Pig Latin vs. Map-Reduce

- Map-reduce welds together 3 primitives:
process records → create groups → process groups

```
a = FOREACH input GENERATE flatten(Map(*));  
b = GROUP a BY $0;  
c = FOREACH b GENERATE Reduce(*);
```

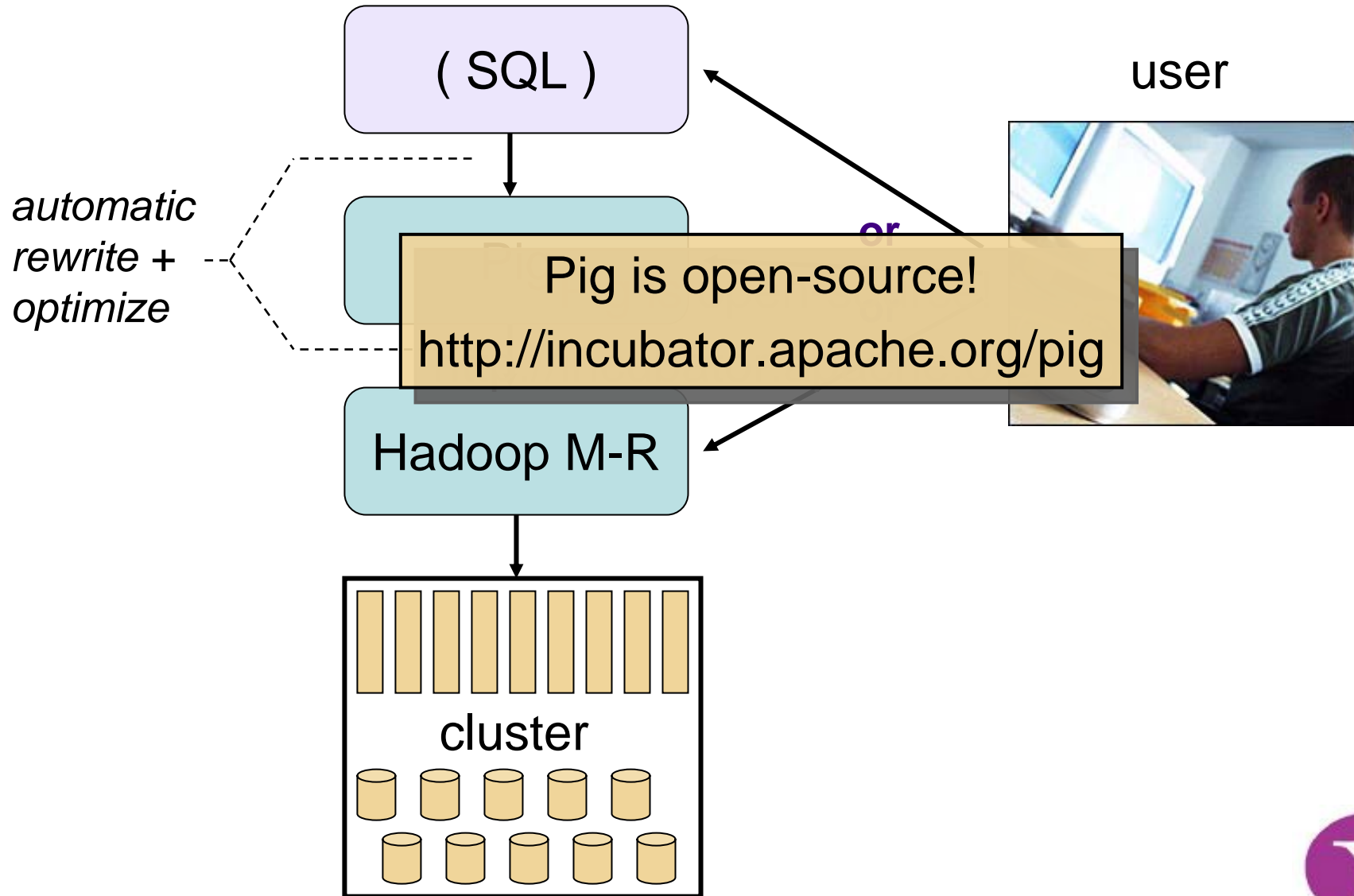
- In Pig, these primitives are:
 - explicit
 - independent
 - fully composable
- Pig adds primitives for:
 - filtering tables
 - projecting tables
 - combining 2 or more tables

more natural programming model

optimization opportunities



Map-Reduce as Backend



Is Pig+Hadoop a DBMS?

DBMS

Pig+Hadoop

workload

**data
representation**

**programming
style**

**customizable
processing**



Ways to Run Pig

- Interactive shell
- Script file
- Embed in host language (e.g., Java)
- *soon*: Graphical editor



Coming Soon to a Pig Near You

- External executables (“streaming”)
- Static type checking
- Error handling (partial evaluation)
- Development environment (Eclipse plugin)



Credits



Shubham Chopra
Alan Gates
Antonio Magnaghi
Shravan Narayanamurthy
Olga Natkovich

Chris Olston
Utkarsh Srivastava
Ben Reed
Amir Youssefi
Xu Zhang

