IOWA

CS5630

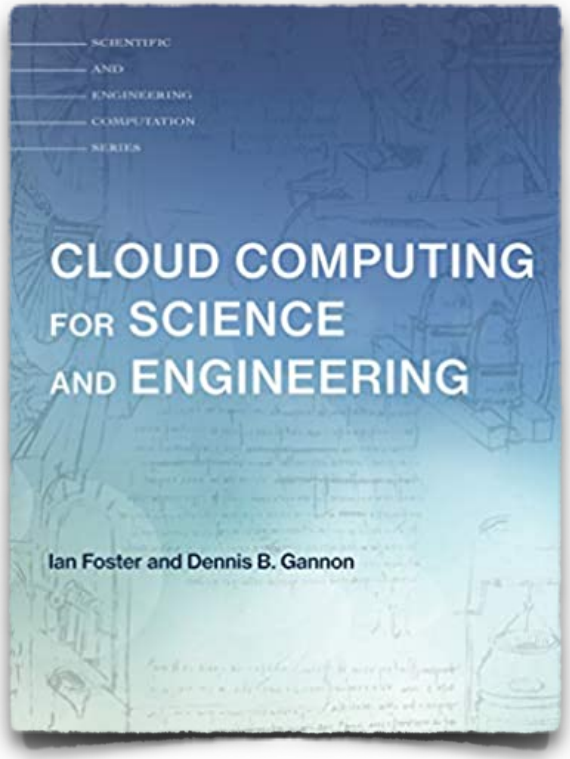
Foundation (4): Compute in the Cloud

Prof. Supreeth Shastri
Computer Science
The University of Iowa

Lecture goals

Technical introduction to computing resources on the cloud

- *Infrastructure as a Service (IaaS)*
- ***Platform as a Service (PaaS)***
- ***Software as a Service (SaaS)***
- ***Other emerging models***



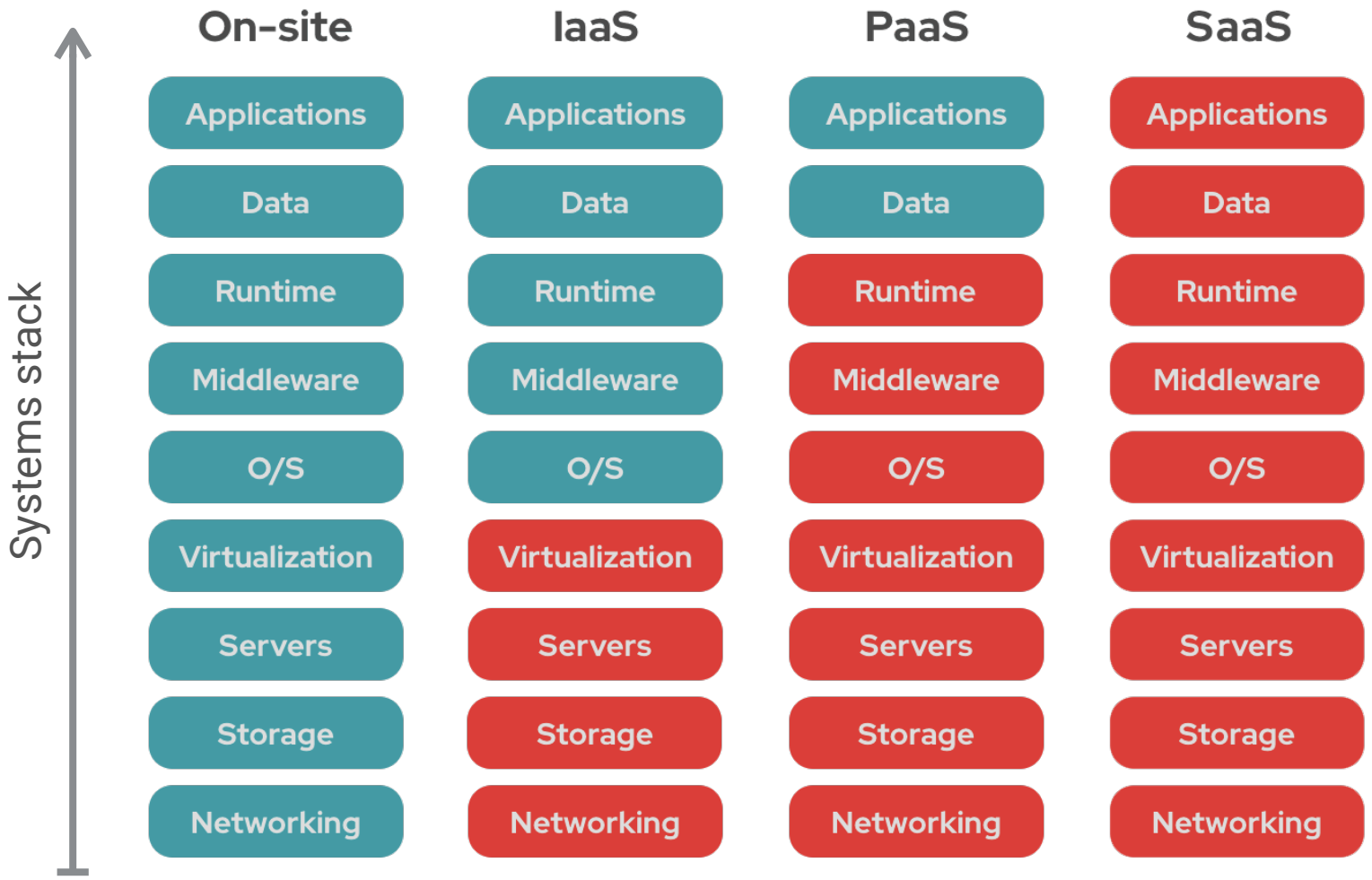
Chapter 8

aws

Google Cloud

IOWA

Visualizing compute in the cloud



You manage

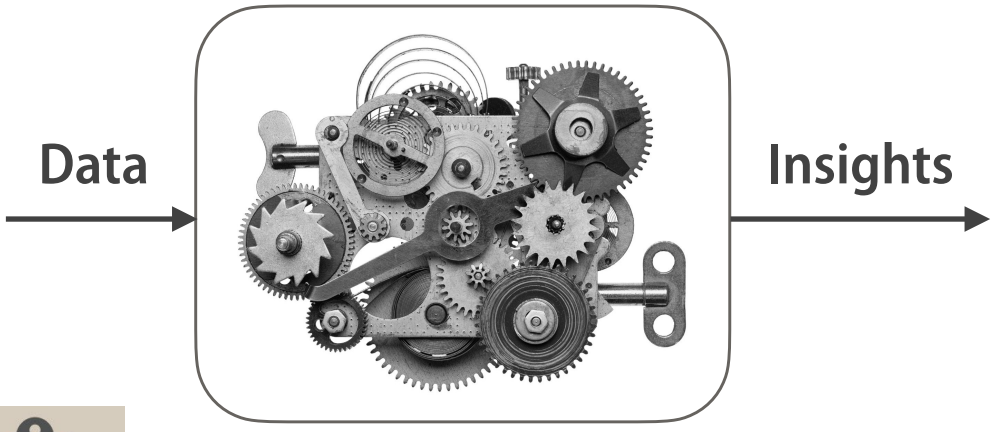
Service provider manages

Image courtesy: Redhat

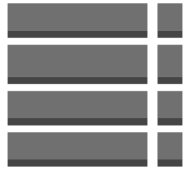
MapReduce Programming

A Primer on Large-Scale Data Analytics

Data Analytics at Large-Scale



Count the number of occurrences of each word in the document



10 TB

300 PB

~ ZB

How to meet the needs of storage and compute infrastructure at large-scale?

- ***Vertical Scaling**: increasing the capacity of the existing nodes.*
  E.g.: supercomputers
- ***Horizontal Scaling**: adding new compute and storage nodes.*
  E.g.: Web servers

Data Analytics at Large-Scale

*At **large-scale**, both storage and computation need to be **distributed** across a large number of nodes*

Think of servers in a cluster or datacenter

*Why solving this problem is **not trivial***

- ☑ *Parallelize the analytics job*
- ☑ *Distribute and manage data*
- ☑ *Handle scheduling and load balancing*
- ☑ *Implement fault-tolerance*

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in

USENIX Symposium on Operating Systems Design and Implementation **(OSDI) 2004**

22728
Citations

WWW index + 900
other projects within Google

Dozens
of clones

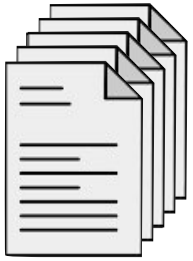
Photo courtesy: https://www.wired.com/2012/08/google-as-xerox-parc/

MapReduce: Programmer's View

Specify the input/output files

Implement the job via map() and reduce()

Flow of control and data



Input files

map(Key$_{in}$, Value$_{in}$) {
Filter/transform input data.Produce intermediate key-value pairs
}

Intermediate files

reduce(Key$_{out}$, list(Value)) {
Summarize intermediate values of particular key. Produce merged output.
}

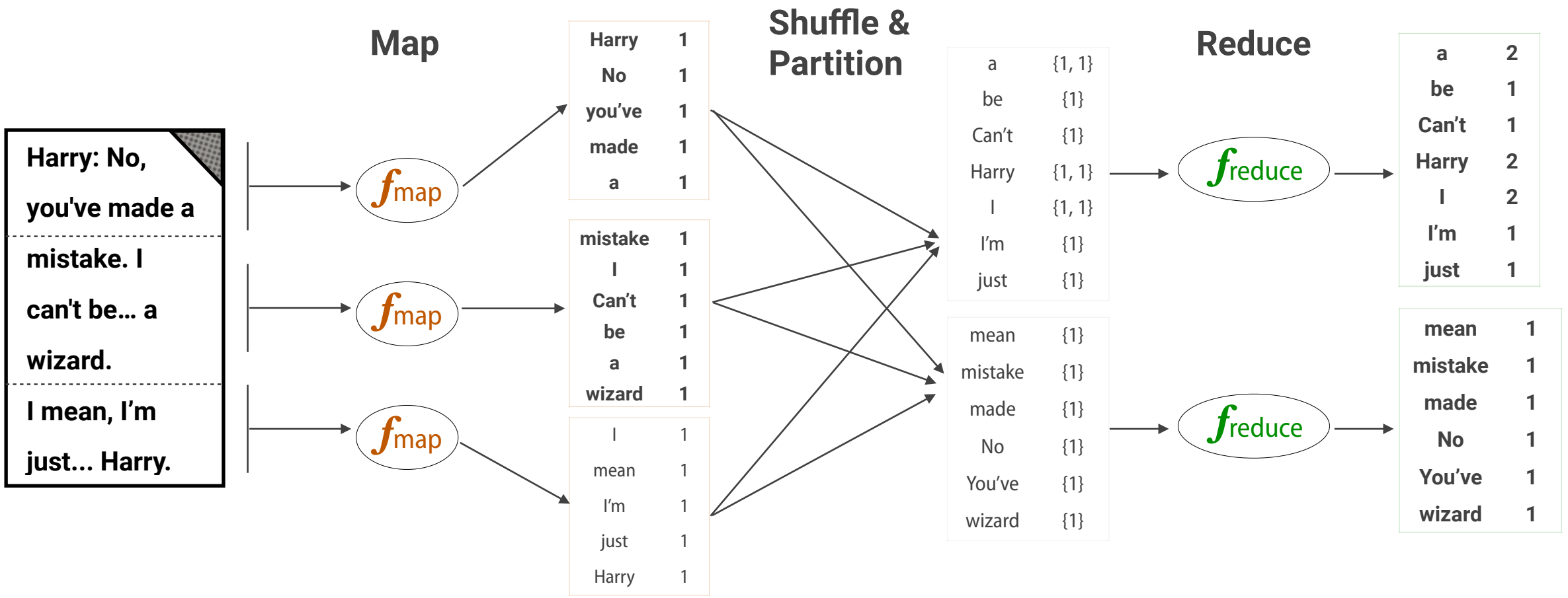
Output file

Word count problem

```
map(fileName, fileData) {
    foreach word in fileData:
        emit(word, 1)
}
```

```
reduce(word, list(values)) {
    int result = count(values)
    emit(word, result)
}
```

Parameters
Map: 3
Reduce: 2



MapReduce: Generality

	Map	Reduce
Distributed grep	Emit <line, document-id> if a line matches the given pattern	Copy all intermediate data to output
Inverted index	Emit a sequence of <word, document-id> pairs	Combine all pairs for a given word, sort that list based on document-id, output <word, list(document-id)>
Distributed sort	Extract key from each record and emit <key, record>	Gather all the pairs in the specified key range, sort them based on key and output list(<record>)

Works well for parallelizable **batch processing** jobs

Not useful for

Interactive jobs
∵ *high latency*

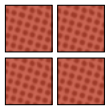
Iterative workloads
∵ *lack of data reuse*

Graph analytics
∵ *tasks are acyclic*

MapReduce Under the Hood: *System Internals* ‡

‡ *Not shown here: Load balancing; Fault tolerance; Straggler mitigation*

User job

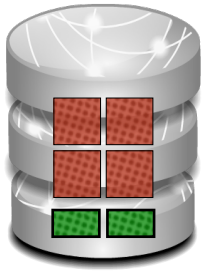


Input data

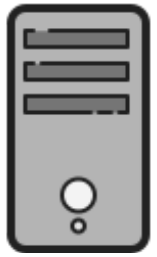
f_{map}

f_{reduce}

1 Job initiated



Distributed File System



MapReduce Master

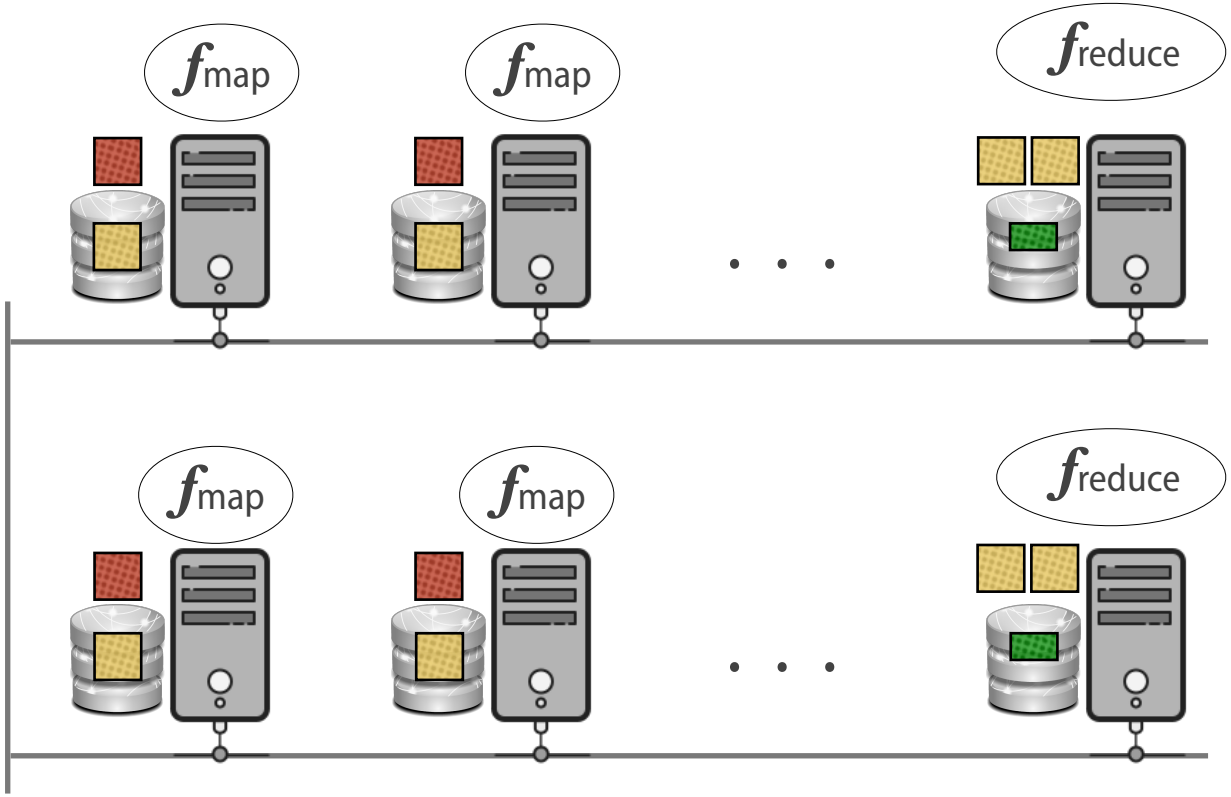
2 Distribute data (**M**-ways)

3 Launch **M** map tasks

4 Partition intermediate data (**R**-ways)

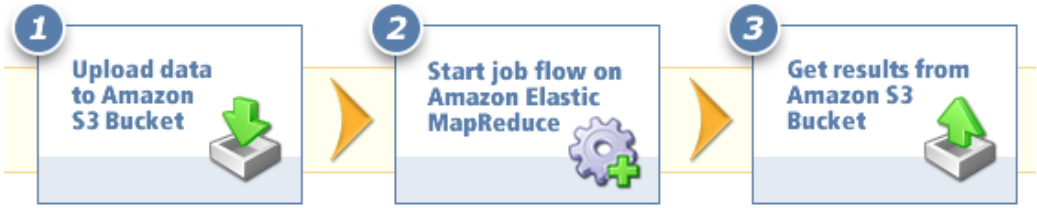
5 Launch **R** reduce tasks

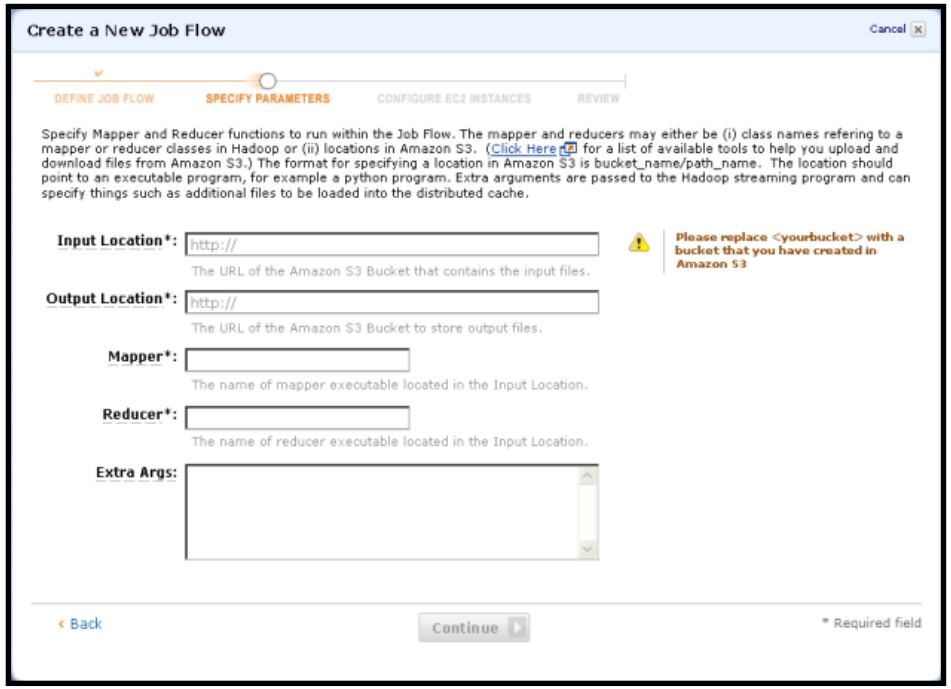
6 Gather output; return

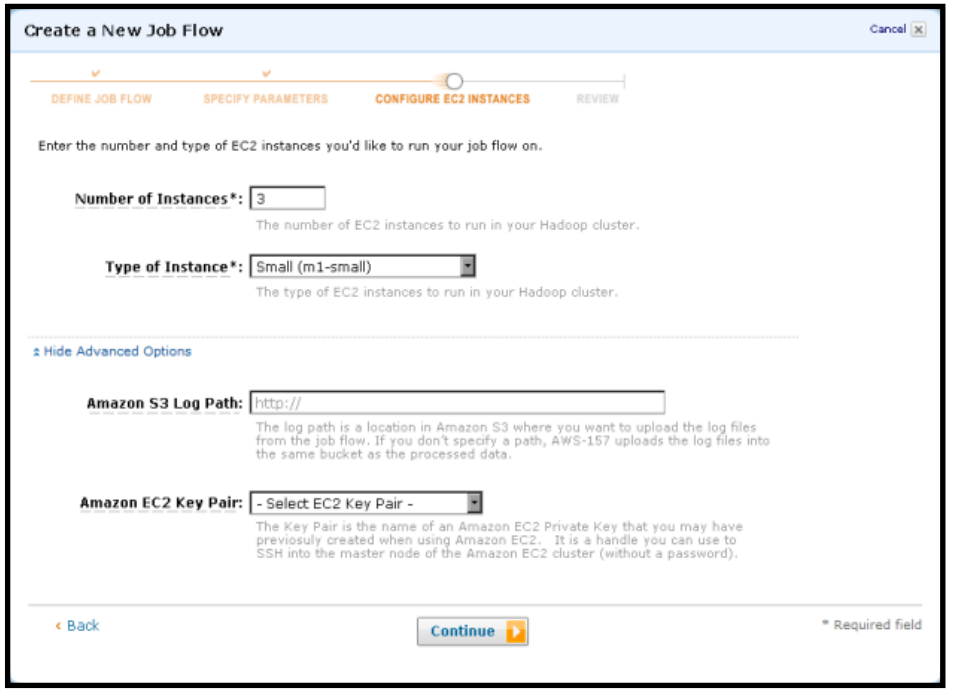


Platform as a Service (PaaS)

- How to accomplish the same on AWS Elastic MapReduce (EMR)?







Software as a Service (SaaS)

- Most pervasive form of computing on the cloud
- Two broad categories of software in the SaaS model
  - *Apps that bring economy and ease of maintenance. E.g., think of UI's Outlook emails and office suite; ICON*
  - *Apps that are hard/impossible to develop and run locally. E.g., Google translate and other AI/ML services*

Segment	2019 Revenue	Market Share
IaaS	$49.0	21.0%
PaaS	$35.9	15.4%
SaaS*	$148.5	63.6%
Total	**$233.4**	**100%**

"With SaaS, the users do not have even the executable file that does their computing: it is on someone else's server. It is impossible for users to ascertain what it really does, and impossible to change it."

— Richard Stallman

Table courtesy: IDC; Worldwide revenue in $B

Function as a Service (FaaS)

- Ability to “write custom code and run it on the cloud” without managing VMs (aka *serverless computing*)
- Functions can be written in languages including Python, Java, Go, Ruby, Node.js, C# and so on
- E.g., Amazon lambda, Google cloud functions, Azure functions

Key characteristics of FaaS functions

- Stateless *i.e., have to use persistent data stores like S3, EFS to store state between invocations*
- Event-driven: responding to an HTTP request, a change in a cloud data store, or a new pub-sub notification
- Sandboxed: executed in a dedicated VM/container spawned for this purpose

- Use cases: simple web service with sporadic traffic, processing data from IoT devices etc.,
- Billing model: provider will charge based on “all the cloud resources” the function consumed

More on this from **Prof. Shahrad**, who has conducted leading research on this topic at Princeton, Microsoft, and UBC

Spot Quiz (ICON)