IOWA

CS5630

Foundation (4): Compute in the Cloud

Prof. Supreeth Shastri
Computer Science
The University of Iowa

Course updates

Lecture recordings

I will make audio recordings available (in addition to the lecture slides).

This is to accommodate unexpected travel/health issues.

But, only those attending in-person will be able to take spot quizzes.

Research papers

We will start the student-led paper discussions from 9/28.

I have advanced this by a week to keep our topics sequential (foundation, cloud infrastructure, and then data analytics).

We will discuss more on this.

Course updates

Guest lecture modalities

Guest lectures are live talks delivered via zoom at regular lecture hours

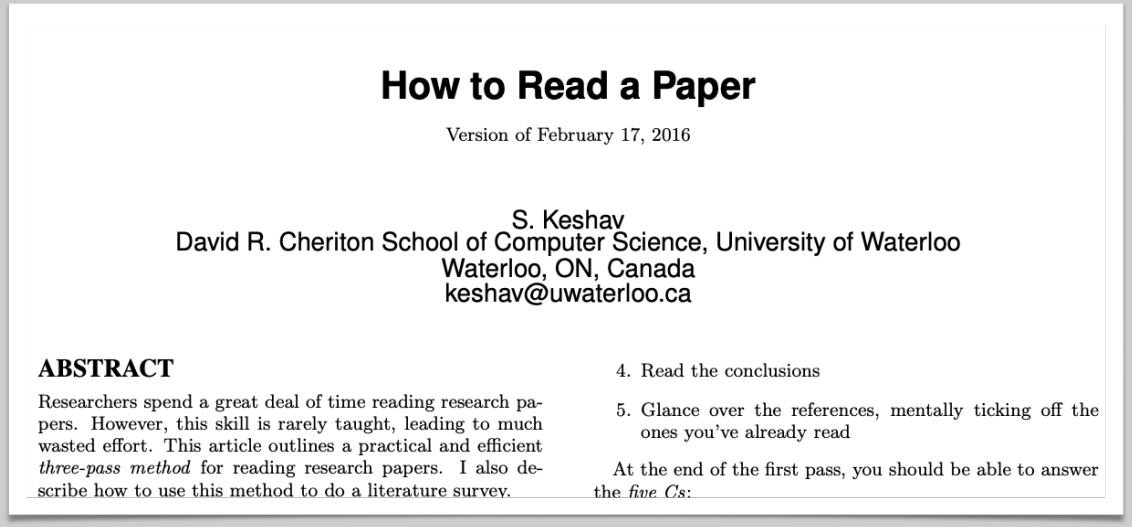
Q1. Would you prefer to have an in-person watch party here? Or join the talk via zoom from your home?

This is true for all guest lectures except for Prof. Renata Borovica-Gajic, who is in Melbourne, Australia! She will speak to us at 9pm central time.

Q2. Does Oct-28 (Thursday) work for you all?

How to read and review research papers

We have papers that help you read more papers!



How to Read a Paper

Version of February 17, 2016

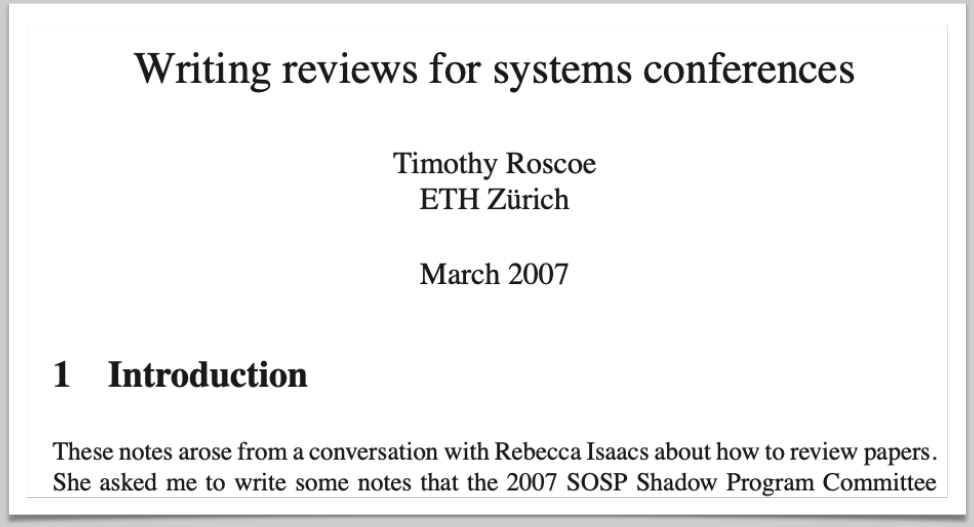
S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@uwaterloo.ca

ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

4. Read the conclusions
5. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:



Writing reviews for systems conferences

Timothy Roscoe
ETH Zürich

March 2007

1 Introduction

These notes arose from a conversation with Rebecca Isaacs about how to review papers. She asked me to write some notes that the 2007 SOSP Shadow Program Committee

Tips for reading a paper

- Read the abstract, introduction, and conclusion first
- Read the entire paper in two/three passes
  - *First pass is to get a birds eye view of the thesis, methods, and findings [10 mins]*
  - *Second pass is for a deeper and fuller understanding of all things [2 hours]*
  - *Third pass is for those reviewing/presenting the paper [5-10 hours]*
- Underline/highlight the important parts of the paper
- Keep notes on the margins about issues/questions
  - *Key insights, questionable claims, relevance to other topics, etc.*
- Look up references that seem to important (or missing)
  - *You may also want to check who references this paper and how*

What to look for in a paper

- The challenge addressed by the paper
- The key insights & original contributions
  - *Real or claimed, you have to check*
- Critique: the major strengths & weaknesses
  - *Claims and assumptions; methodology; analysis/evaluation; presentation style*
- Future work: extensions & improvements
  - *Can we apply the methodology to other problems?*
  - *What are the broader implications?*

How to write a good review

- The first step is to read the paper (and read it well)

- Writing down your review
  - *First, summarize the paper in your own words (neutral not judgmental)*
  - *Next, identify and evaluate the key contributions of the paper*
  - *Third, give specific comments on the content of the paper including its novelty, technical strength, limitations, and writing style*
  - *Fourth, describe the impact of the paper on the field (a retrospective view)*
  - *Finally, write your conclusion (brief; could include subjective opinions)*

- Few more advice
  - *Read sample reviews linked on the course site*
  - *Keep the writing style and tone professional*
  - *Ideal review length is between 500-800 words*

How to prepare for your research talk

- You assume the role of the primary authors of the paper
  - *This is not as hard as it sounds 'cause you're not inventing the work, just developing expertise on it*

- Explore and learn from existing resources
  - *Most conferences archive their talk slides and multimedia*
  - *Many of these papers are classics ==> other students/faculty may have given talks on them*
  - *Authors usually create extra content such as websites, Q&A, expanded tech reports etc*
  - *Don't be shy to engage with the original authors; they love hearing from well-read readers*

- Few more advice
  - ***DO NOT copy slides** from other talks; it is ok to be inspired by them*
  - *Use Iowa branded slide template*
  - *Practice your talk several times*

Teams and presentation

17

teams

- Look for a teammate (e.g., on Slack)
- Select a paper that you find interesting
- Sign up for a slot to present in the class

30 mins **presentation** + **15** mins **discussion**

- Both team members will participate equally and actively in all tasks
- Both members get the same grade (unless there are discrepancies)
- Submit paper reviews and talk slides to the instructor at least a day in advance

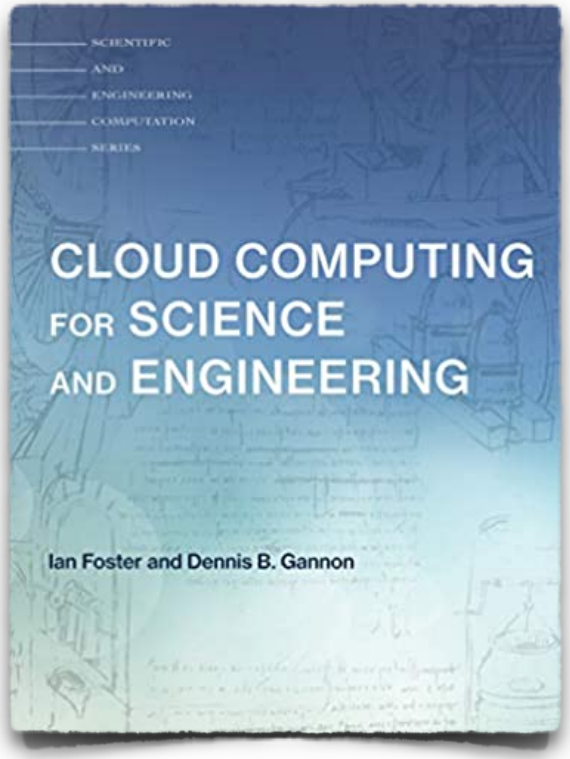
Tasks	Weight
Paper review	25%
Presentation	50%
Handling Q&A	25%

Remember that this activity makes up 25% of your final grade

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***



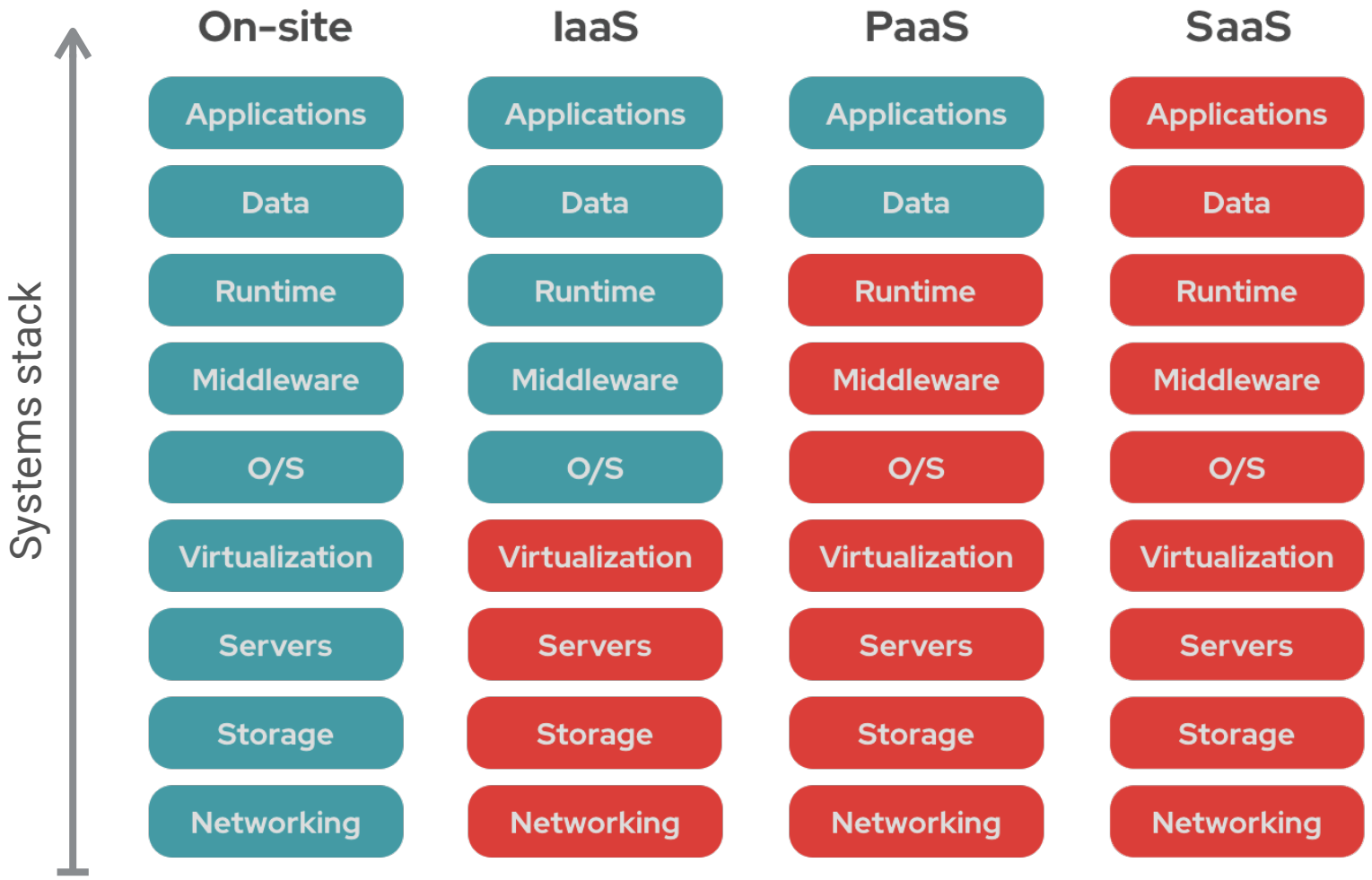
Chapter 8

aws

Google Cloud

IOWA

Visualizing compute in the cloud



You manage

Service provider manages

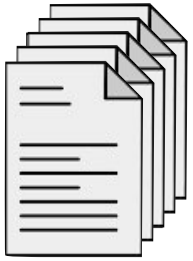
Image courtesy: Redhat

MapReduce: 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.
}

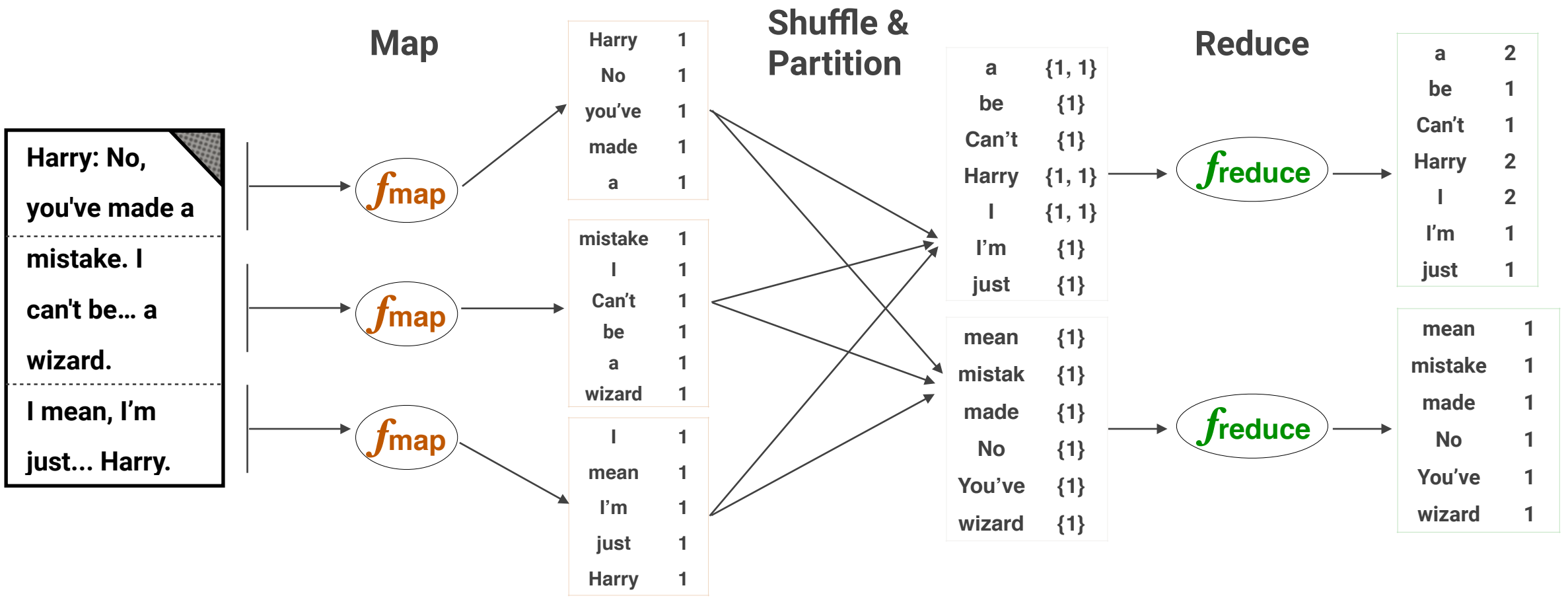
Output file

Word count problem

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

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

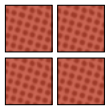
Parameters
Map: 3
Reduce: 2



MapReduce Under the Hood: *System Internals* ‡

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

User job

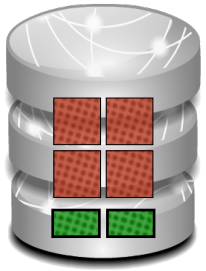


Input data

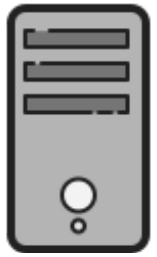
f_{map}

f_{reduce}

1 Job initiated



Distributed File System



MapReduce Master

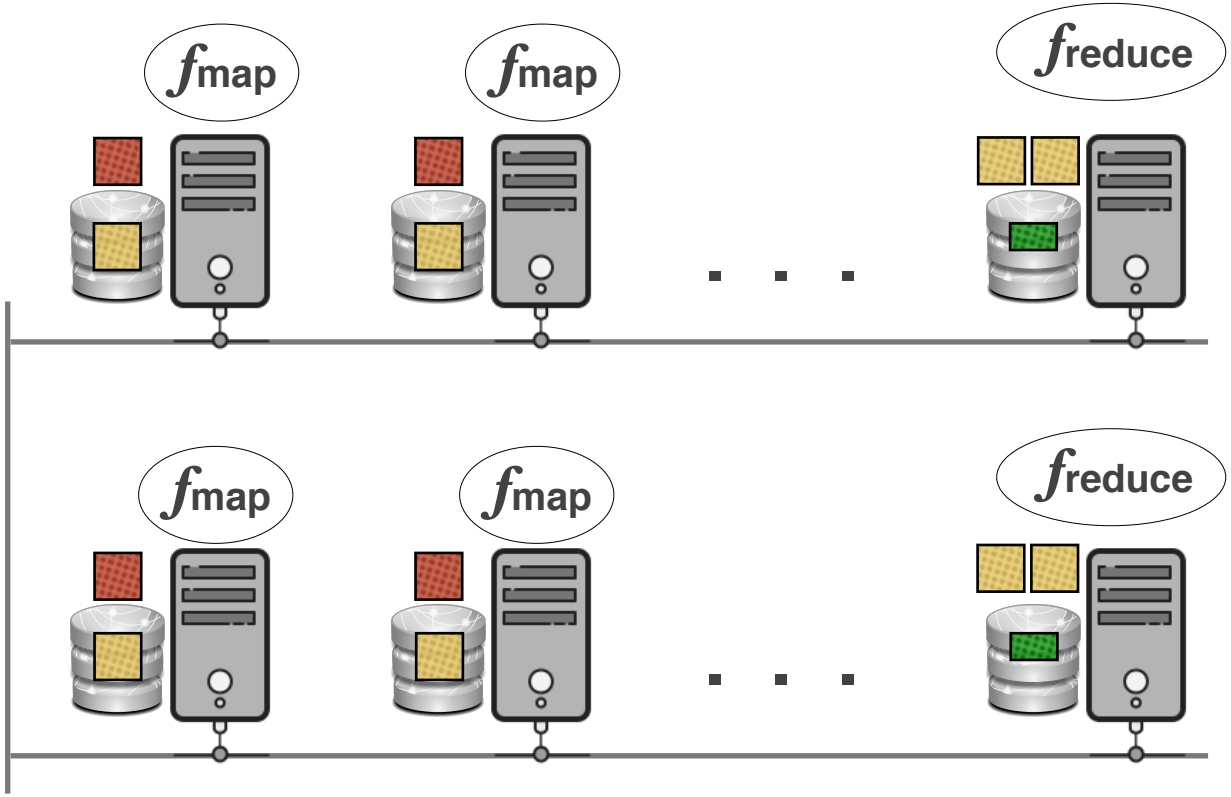
2 Distribute data (**M**-ways)

3 Launch **M** map tasks

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

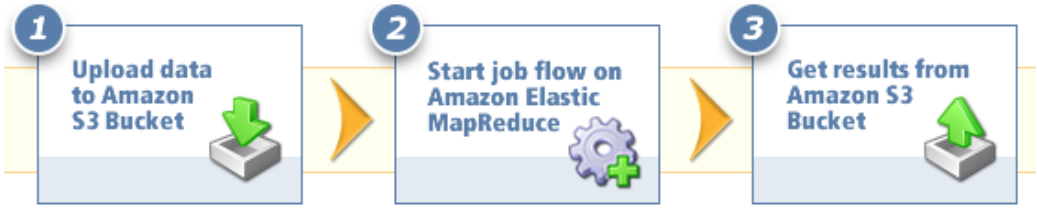
5 Launch **R** reduce tasks

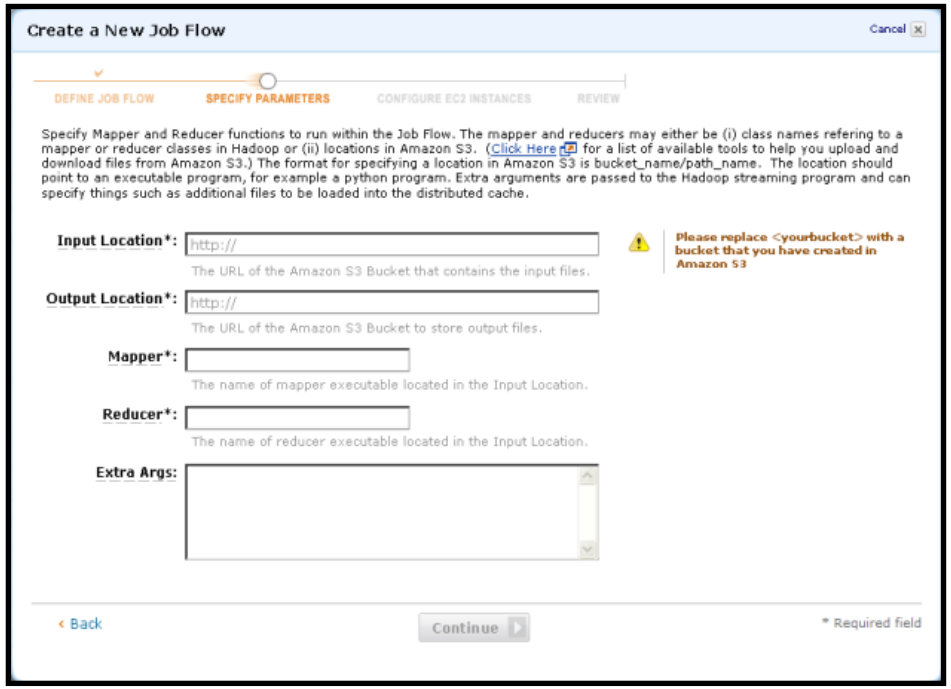
6 Gather output; return

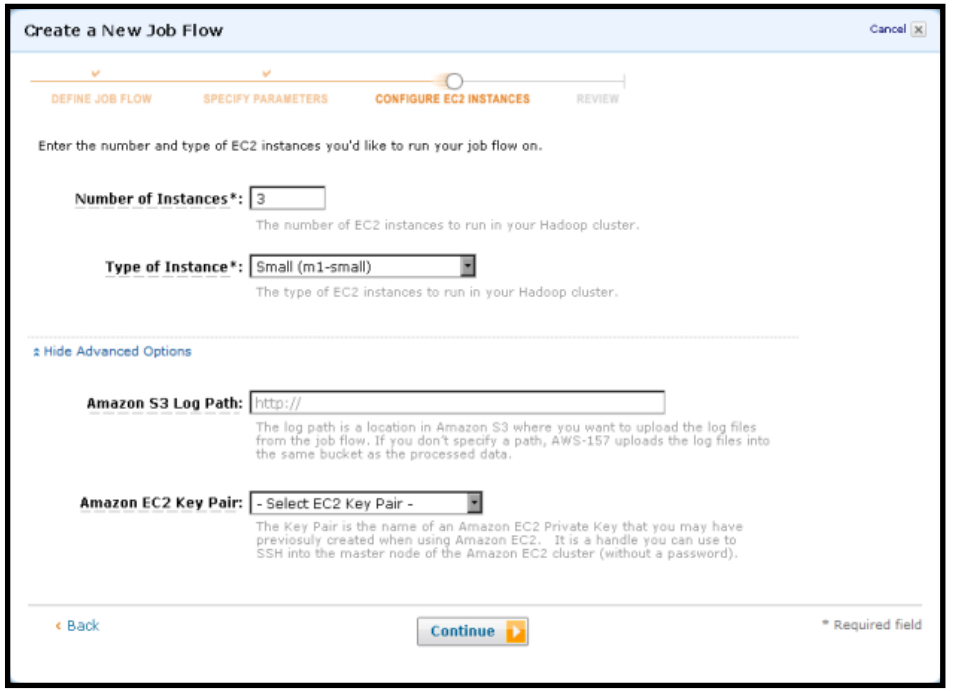


Platform as a Service (PaaS)

- 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, where the users can't see or touch it. Thus it is impossible for them 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*
- Even-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)