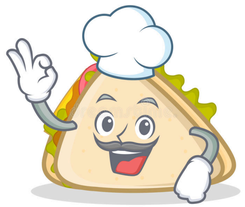
MapReduce

Ryan Eberhardt
August 16, 2021

Ryan’s Sandwiches

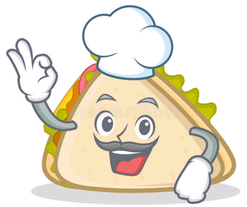
- Let’s say Ryan has developed an amazing new sandwich recipe



ryan

Ryan’s Sandwiches

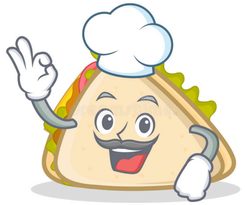
- Let’s say Ryan has developed an amazing new sandwich recipe
- This involves prepping some ingredients and combining them in a special way to produce a final product



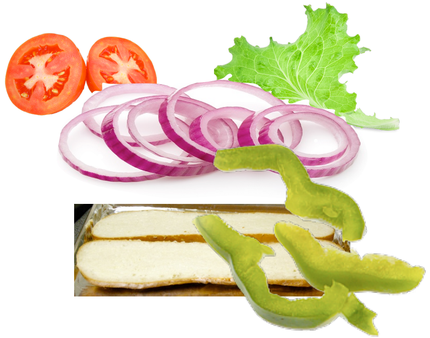
ryan

Ryan’s Sandwiches

- Let’s say Ryan has developed an amazing new sandwich recipe
- This involves prepping some ingredients and combining them in a special way to produce a final product

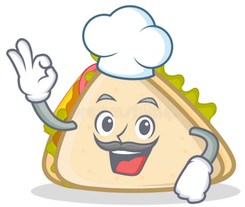


ryan



Ryan’s Sandwiches

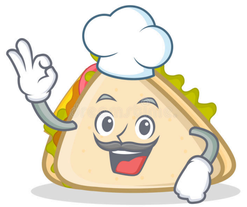
- Let’s say Ryan has developed an amazing new sandwich recipe
- This involves prepping some ingredients and combining them in a special way to produce a final product



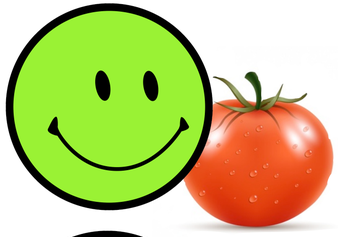
ryan

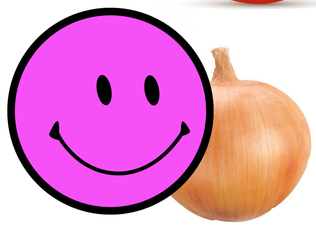
Ryan’s Sandwiches

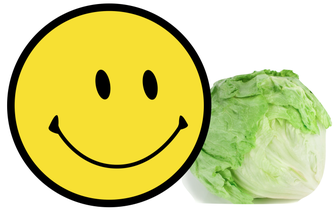
- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive



ryan

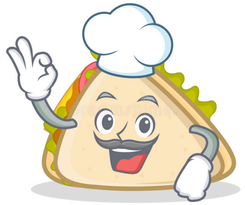




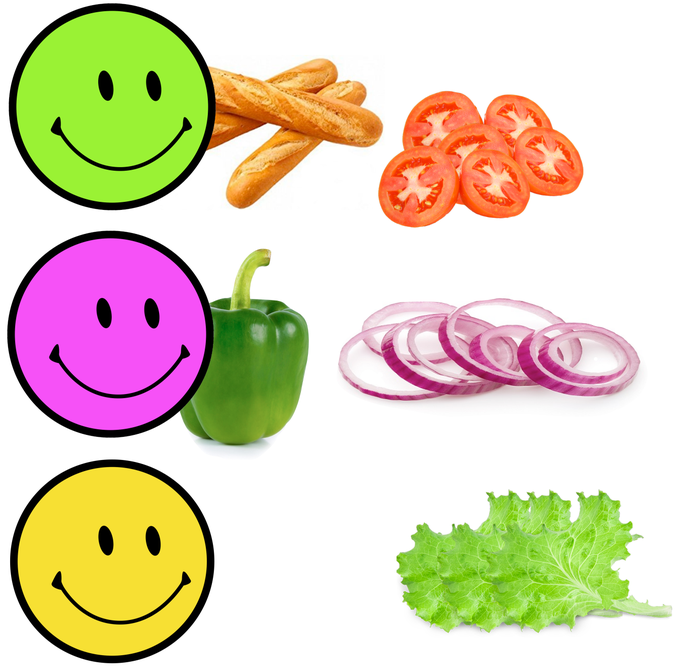


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

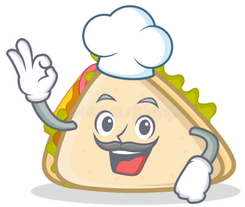


ryan

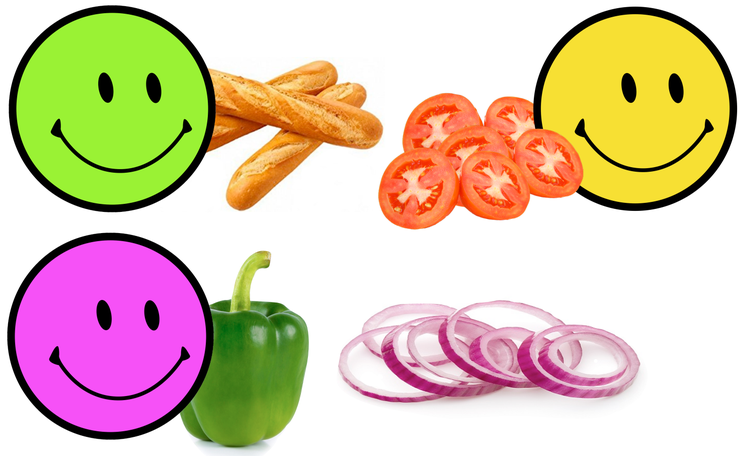


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

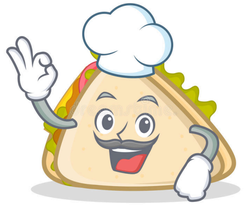


ryan

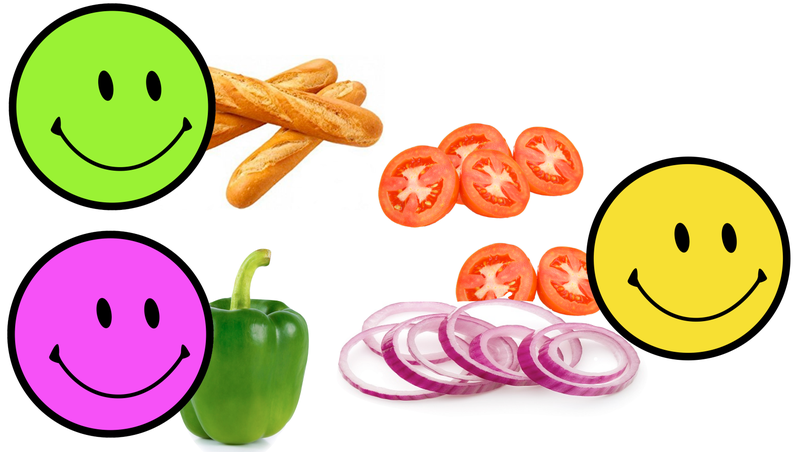


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

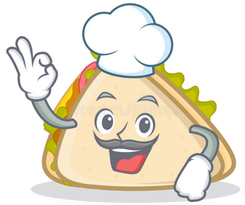


ryan

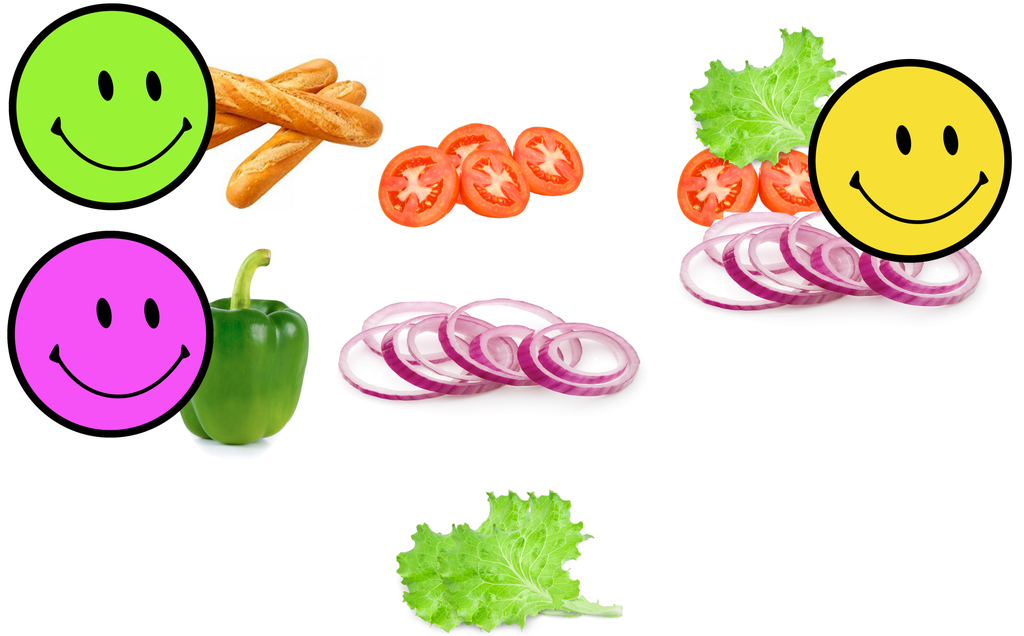


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

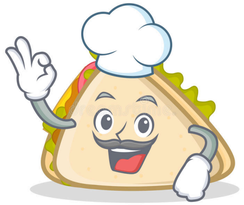


ryan

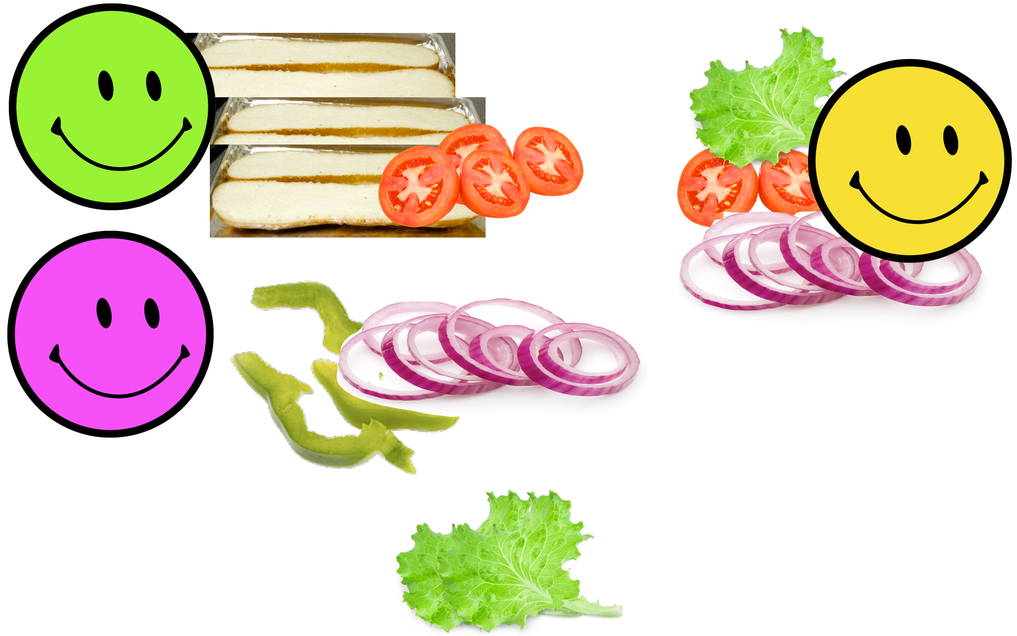


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

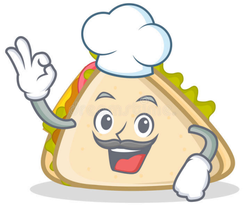


ryan

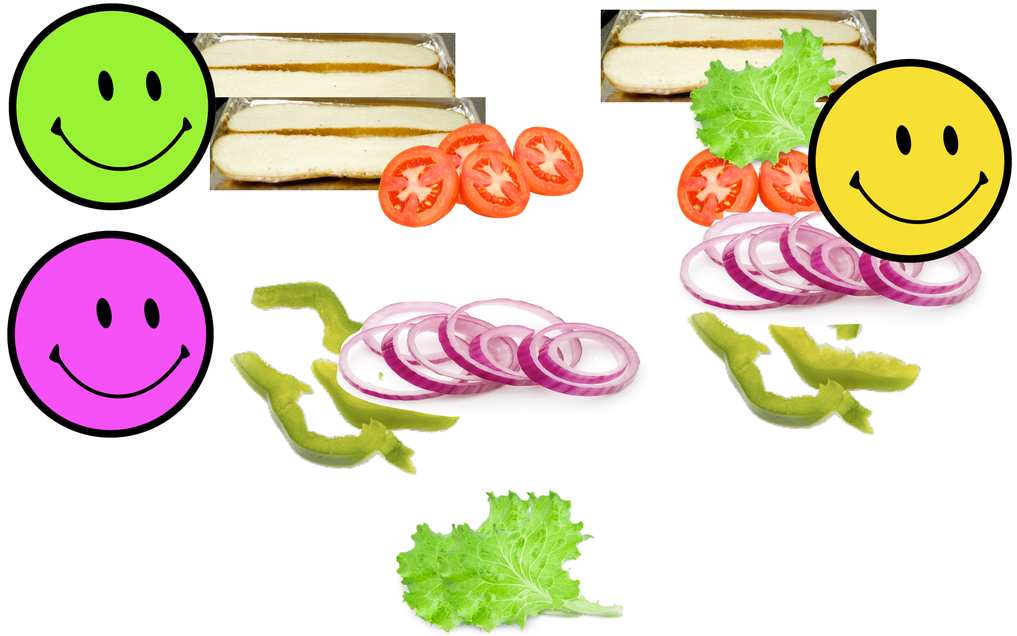


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive

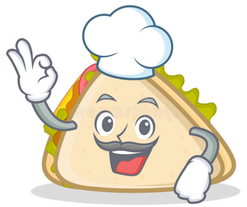


ryan

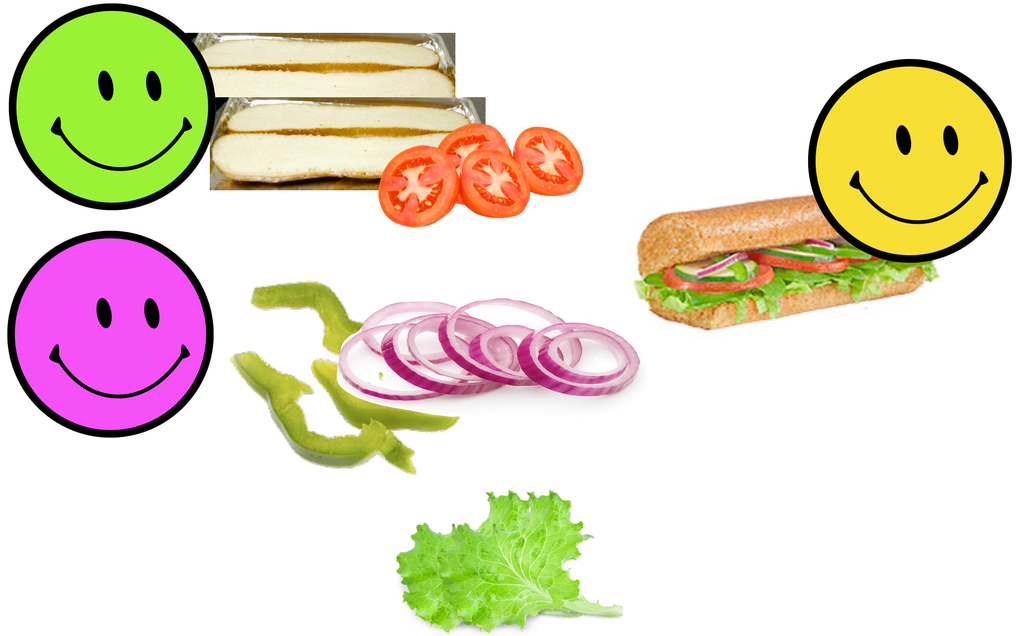


Ryan’s Sandwiches

- Ryan’s restaurant is increasing in popularity, but he can’t keep up with the demand
- On top of honing his recipe, now he needs to worry about hiring/training employees and designing a process to make everyone productive



ryan



Pam’s Pottery

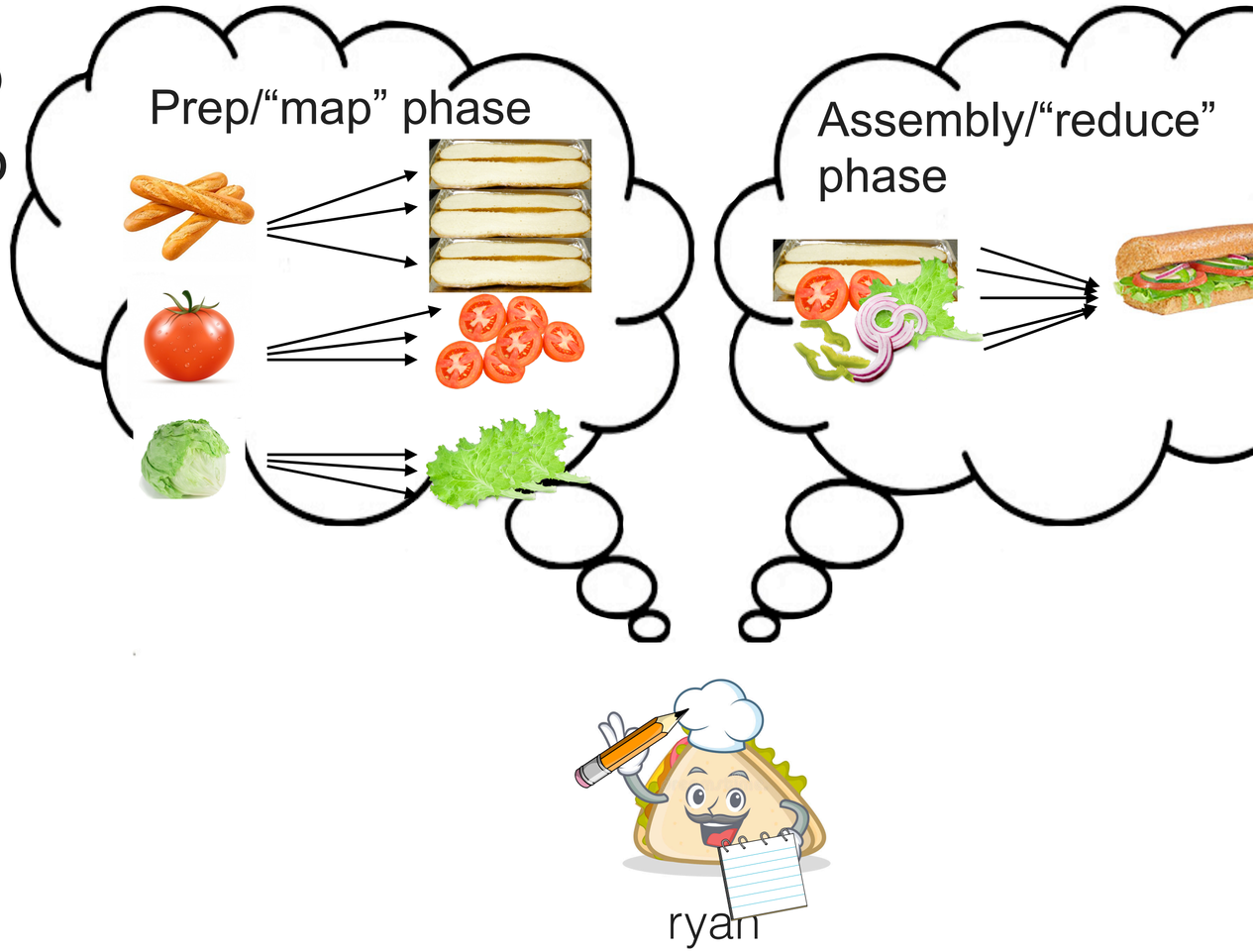
- Next door, Pam has designed a new mug that is a smashing success
- Now she needs to figure out how to scale her process

Designing for scalability

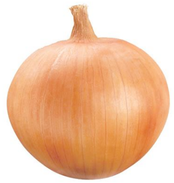
- There are two separate problems here:
  - Doing some domain-specific work (e.g. making a killer sandwich)
  - Scaling that work up: coordinating many people, handling edge cases (e.g. workers get sick or fail to do good work)
- It would be better to separate these problems:
  - Let domain experts focus on what they're good at without needing to think about how to manage hundreds of employees
  - Hire a "coach" or "orchestrator" who specializes in coordinating employees, without any domain knowledge

MapReduce

- We can ask Ryan the Chef to break up the process of making a sandwich into two phases: a “map” (prep) phase, and a “reduce” (assembly) phase
  - Ryan writes instructions for each step in a program
- The orchestrator then handles hiring employees, assigning them to tasks within each phase, handling failures, etc

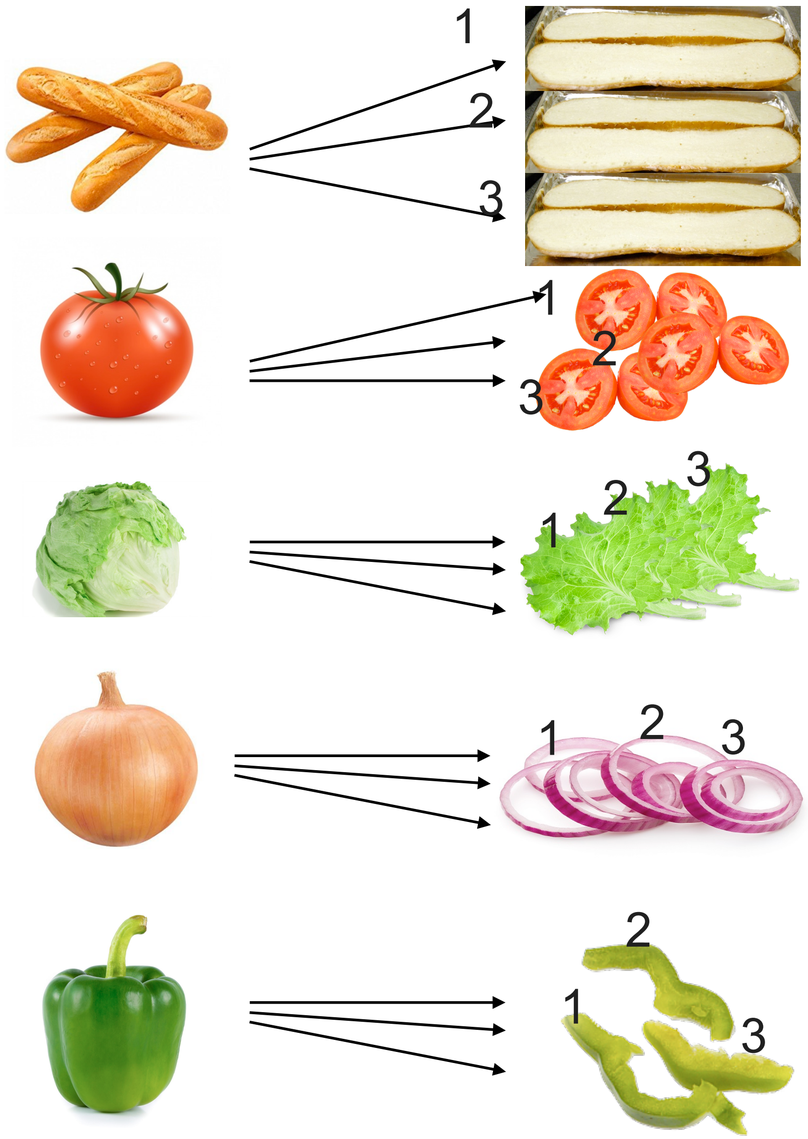


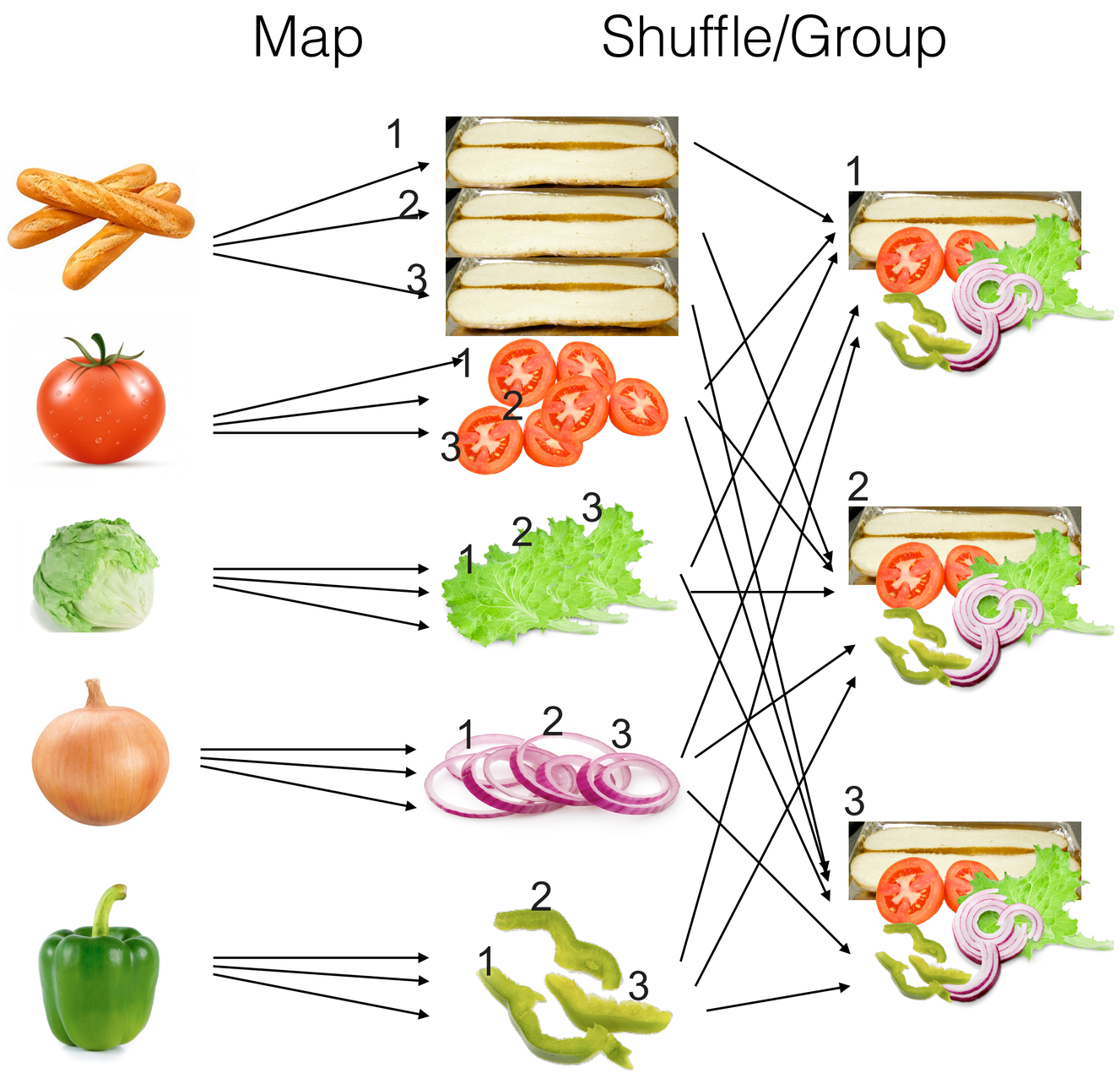
Inputs

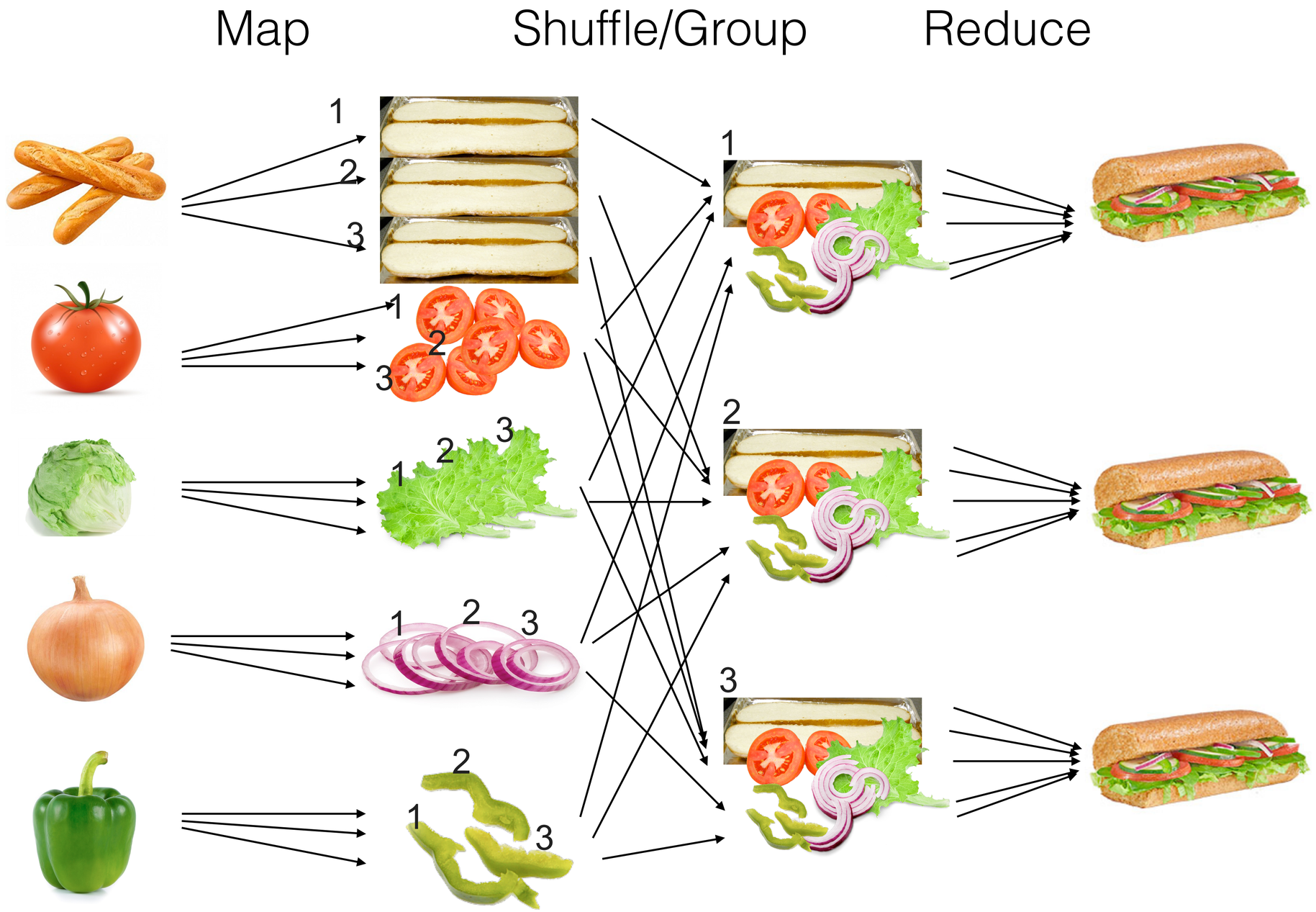


Outputs

Map




Map
Shuffle/Group
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3


Map
Shuffle/Group
Reduce
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3

MapReduce is versatile!

- Once this framework is in place, it can be used for any task that can be broken into “map” and “reduce” steps
- MapReduce was originally introduced to build Google’s search index
  - CS 106B-level search index:
    - Create map<search term, list of documents>. When looking up a search term, you can easily get a list of documents matching that term
    - May also include a term frequency in the document, or sort the list of documents by frequency
    - Populating this sequentially: for every document, for every term: map[term].push_back(document)
  - Take CS 124 to learn about better search indexes :)

Inputs

index.html

lecture-1.html

lab-3.html

midterms.html

Outputs

CS110 index.html:1 lecture-1.html:1

filesystems lecture-1.html:1 midterms.html:1

processes lab-3.html:1 midterms.html:1

quicksort index.html:1

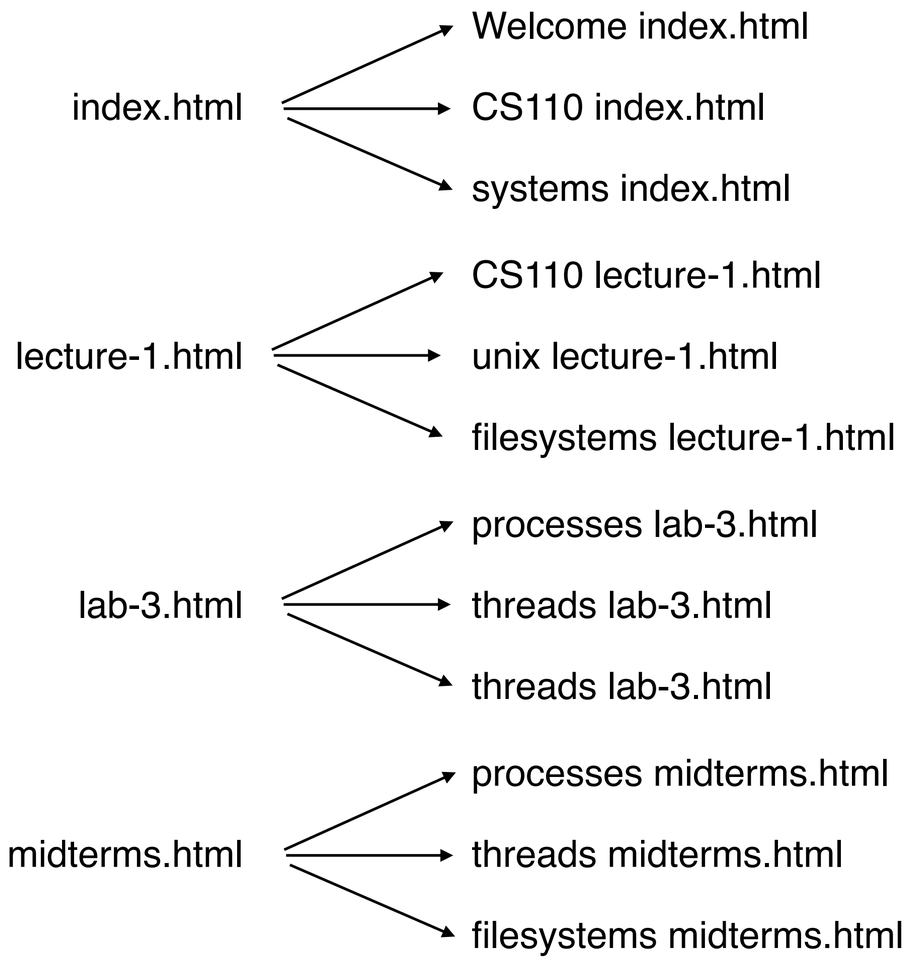
systems index.html:1

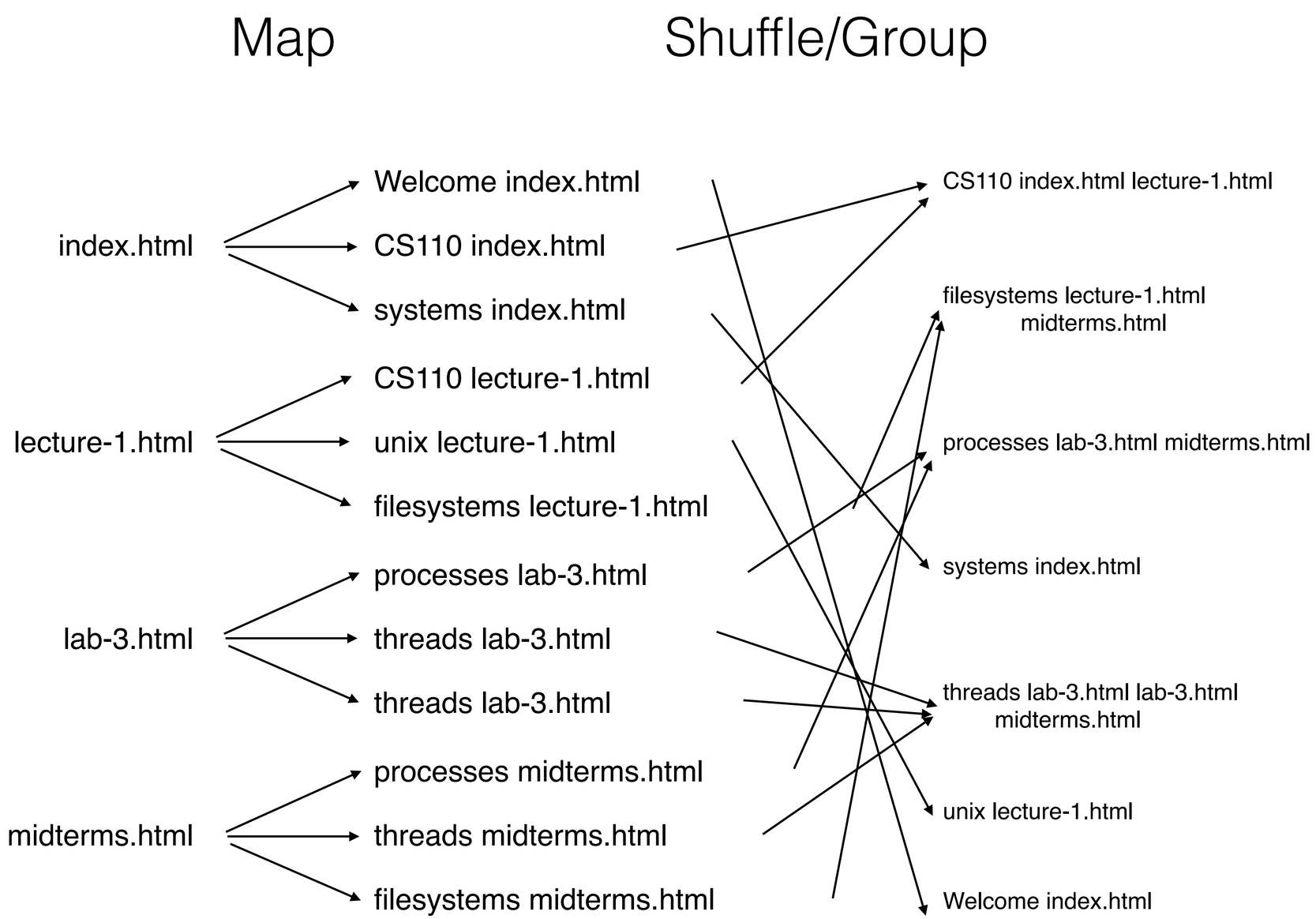
threads lab-3.html:2 midterms.html:1

unix lecture-1.html:1

Welcome index.html:1

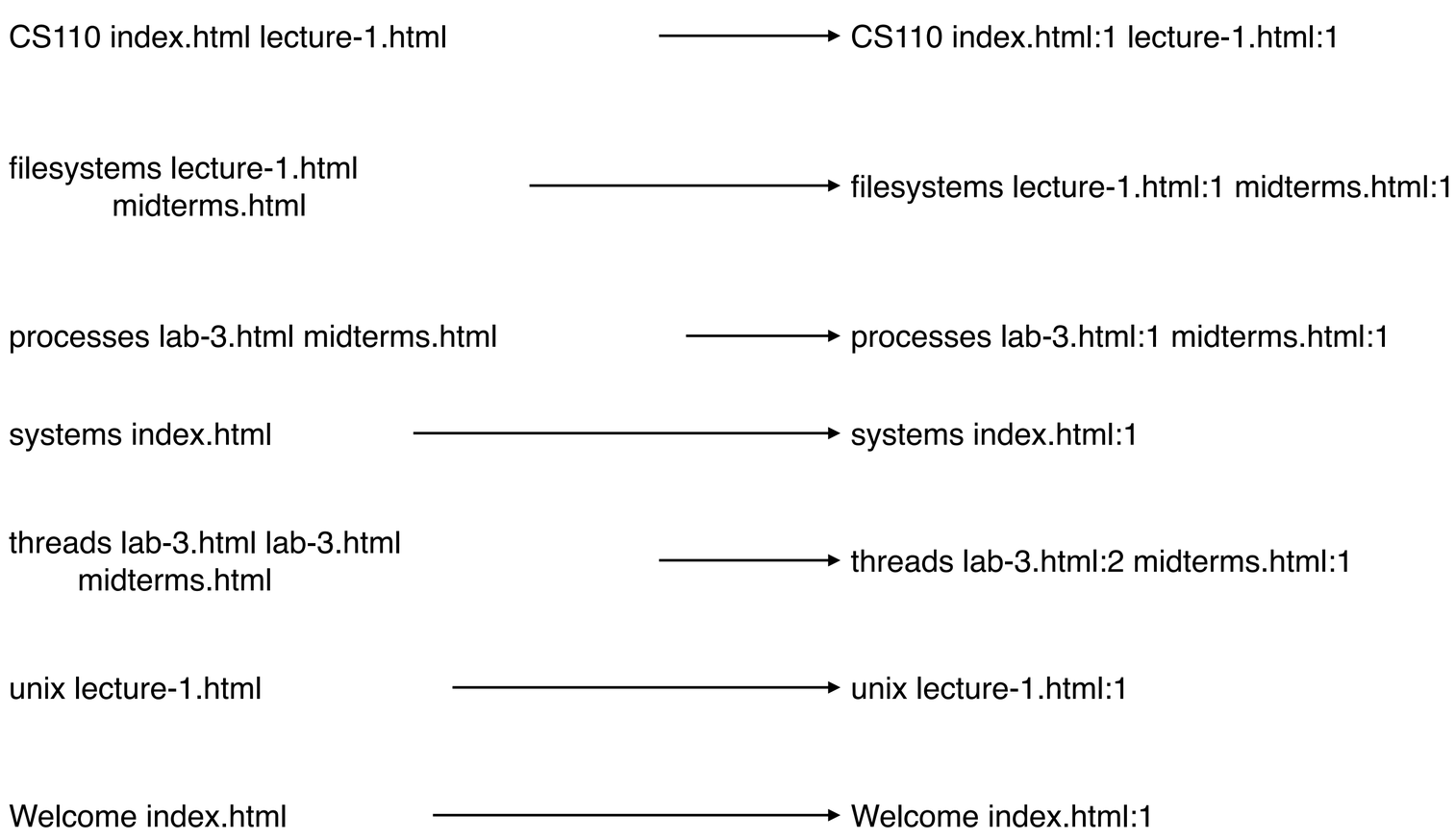
Map


index.html
Welcome index.html
CS110 index.html
systems index.html
lecture-1.html
CS110 lecture-1.html
unix lecture-1.html
filesystems lecture-1.html
lab-3.html
processes lab-3.html
threads lab-3.html
threads lab-3.html
midterms.html
processes midterms.html
threads midterms.html
filesystems midterms.html


Map
Shuffle/Group
index.html
lecture-1.html
lab-3.html
midterms.html
Welcome index.html
CS110 index.html
systems index.html
CS110 lecture-1.html
unix lecture-1.html
filesystems lecture-1.html
processes lab-3.html
threads lab-3.html
threads lab-3.html
processes midterms.html
threads midterms.html
filesystems midterms.html
CS110 index.html lecture-1.html
filesystems lecture-1.html
midterms.html
processes lab-3.html midterms.html
systems index.html
threads lab-3.html lab-3.html
midterms.html
unix lecture-1.html
Welcome index.html

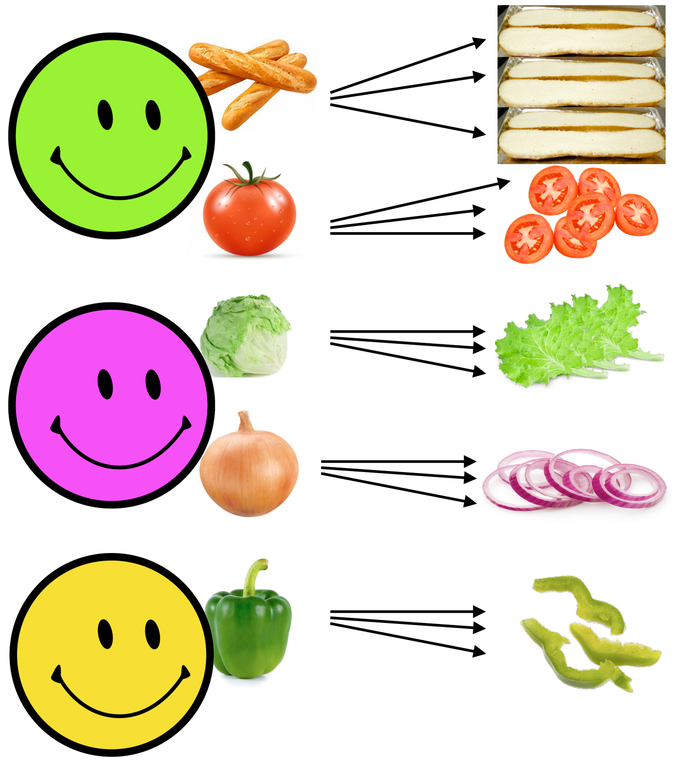
Reduce

(shuffle output)



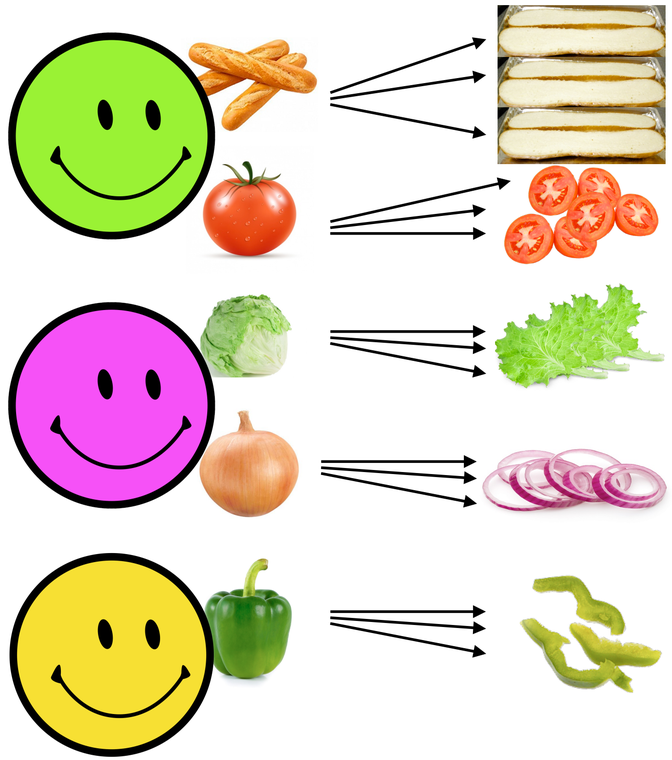
Parallelization

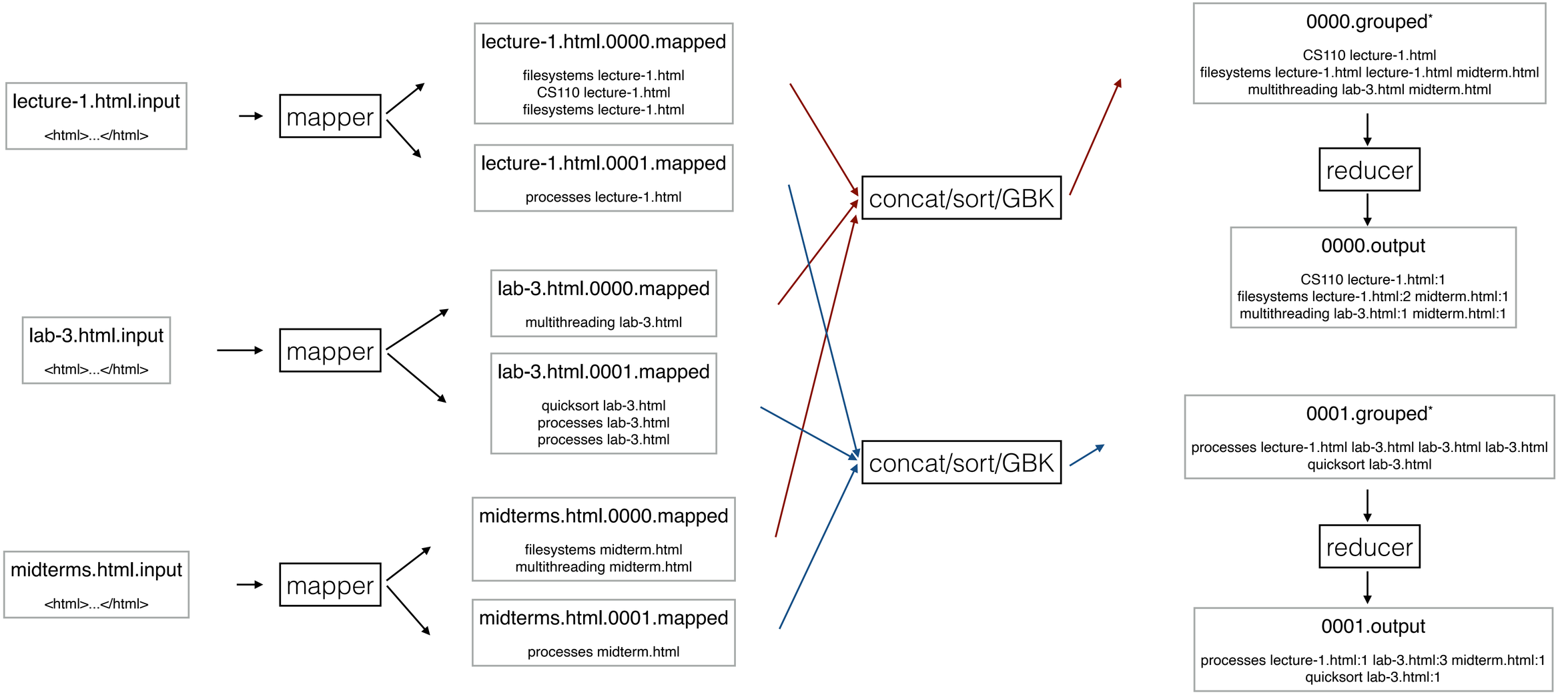
- Parallelizing the “map” step is easy
  - If we have n “ingredients” and m “mappers”/workers, have each mapper process n/m ingredients



Parallelization

- Parallelizing the “map” step is easy
  - If we have n “ingredients” and m “mappers”/workers, have each mapper process n/m ingredients
- How should we now parallelize the “shuffle/ group” step?
  - If there are 50 ingredient piles, and every worker has to go to every pile to get some ingredients, it’s going to be a mess!
  - Better idea: have the mapper workers split their output into separate piles, pre-organized for each reducer





* You won't actually create **.grouped** files in Assignment 6; this is done in memory for better performance.

More MapReduce applications

- Log analysis: which IP addresses are suspicious?
  - Mapper: log line → (IP : content accessed)
  - Reducer: (IP : all content accessed) → probability of being malicious
- Nearby gas stations
  - Mapper: gas station → (POI : distance to station)
  - Reducer: (POI : distance to all nearby stations) → (POI : closest station)

Limitations

- MapReduce is highly disk-based
- Fixed architecture: problem must be broken down into a single map stage and a single reduce stage
- Oriented around batch processing