GDC MapReduce Notes

Josh Snider

October 12, 2015

FP Review

Data structures are not modified, new data structures are made instead. This means we don't need to lock data structures. It also means that since we don't have side effects, we make everything a separate thread and put them together at the end.

Functional programming also allows you to use functions as arguments as the name implies. Functional programming is also good for generic yet typesafe programming, but languages like OCaml can still screw this up.

Standard functional programming functions:

- Map - Apply a function to each element of a list and return a list made my concatenating the results together.
- Fold - Track the result of calling f on each element and an accumulator, return the final accumulator. Can be either a left or right fold.

These are basically the Map and Reduce in MapReduce.

MapReduce

Motivated by large scale data processing, parallelize across thousands of computers, and make it easy for programmers to use. MapReduce has built-in fault tolerance and network monitoring. The user basically has two functions to provide

- map (in_key, in_value) -> (out_key, intermediate_value) list - Input consists of records from various sources.
- reduce (out_key, intermediate_value list) -> out_value list

After the map phase happens, we combine all the intermediate values for a given key into one list and then start the reduce phase. It's good practice for reduce to only have out_value. We use barrier synchronization, to prevent people from going to the reduce step while people are still mapping.

Optimization

Each map function runs parallelly and the reduction for each key runs in parallel. This means we have a bottleneck waiting for mapping to finish before we start reduction. How do we optimize this?

- Slow-moving "map" tasks are run on multiple hosts and then we take the first one to finish.
- Google considered making reducers lazy, but that has some design flaws that made it not used.
- We can have "combiners" which run mini-reduce phases before the actual reduce in order to save bandwidth. If our reduce is both associative and commutative, then we can use it as our combine as well.

Written in C++, with Java and Python bindings. The dividing program tries to divide up map tasks so that mappers are on the same computer as the data. Tasks are chunked in 64MB which is the same size as the Google File System's chunks.

Ensuring fault tolerance

- We give up on tasks if certain key-value pairs reliably crash.
- Mappers that fail before reporting results are redone.
- Reducers don't claim to be done, until their results are reliably backed up.

Impact

Has had great results at Google and is a shining jewel of functional programming. Greatly simplifies large-scale computations. Lets programmer focus on problem and let library handle messy details.

File Systems