Scaling Nutch

Doug Cutting
doug@archive.org

5th International Web Archiving Workshop
9/22/05, Vienna

Nutch is...

- A young open-source project;
- Web search application software;
- Small but growing group of users and developers;
- Behind a few sites;
- An Apache project.;
- Built on Lucene.

Nutch isn't...

- A business;
- A search site;
  - But want to power lots of search sites;
  - From domain-specific, to whole-web.
- A research project.
  - But want to be platform for research.

Nutch Technical Goals

- Scale to entire web
  - pages on millions of different servers
  - billions of pages
  - complete crawl takes weeks
  - very noisy
- Support high traffic
  - thousands of searches per second
- State-of-the-art search quality

Scalability

- To meet scalability goals:
  - multiple simultaneous fetches
    (100+ pages/second / CPU, ~10M / day)
  - parallel, distributed db update
    (100M pages @ 100 pages/second / CPU)
  - distributed search
    (2-20M pages, 1-40 searches/second / CPU)

Initial Scalability

- Initial implementation is scalable...
  - parallel processes on multiple machines
  - some serial bottlenecks, but w/ plans to resolve
  - 100M web pages demonstrated

... but not to billions of pages

- scales better than other open source options
- but large installations are operationally onerous
  - manually monitoring multiple machines is painful
  - data-interchange and space-allocation difficult
- with single operator
  - hard to use more than a handful of machines
  - effectively limited to ~100M pages

Need a Distributed File System

- single, shared namespace
- fault-tolerant
  - disk failures
  - node failures
- use disks already on computing nodes
- scales
  - easy to add & remove disks from live system
  - (with 1000 disks, expect disk failure daily)

nutch

Google publishes GFS paper

- meets all our needs (not surprisingly)
- single *namenode*
  - maps name → <blockId>*
  - maps blockId → <host:port>$^{replication\_level}$
- many *datanodes*, one per disk generally
  - map blockId → <byte>*
  - poll namenode for replication, deletion, etc. requests
- client code talks to both

Nutch Distributed File System (NDFS)

- if it's good enough for Google...
- open-source, Java implementation of GFS
- part of Nutch project
- first implemented in 2003 by Mike Cafarella
- currently maintained in branches/mapred

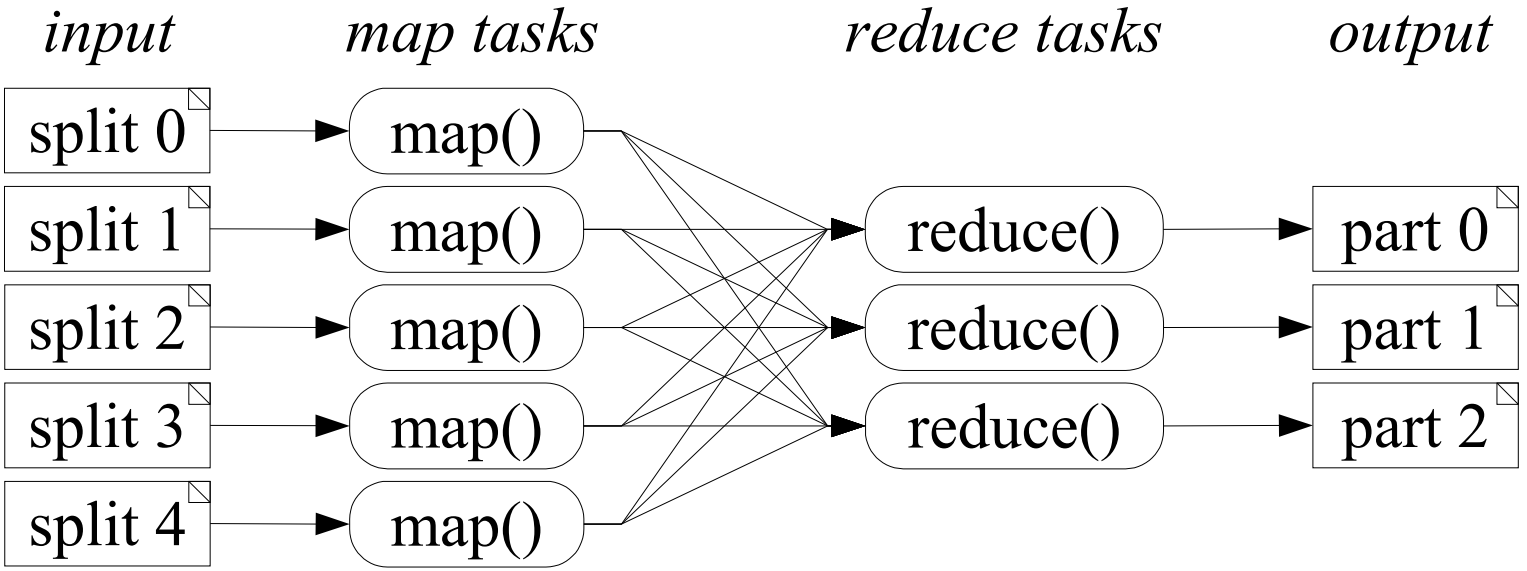
Need Distributed Computing Platform

- partition stages (*jobs)* into work units (*tasks)*
- for each task
  - allocate host
  - start
  - monitor
  - kill when hung
  - re-start when fails
- sequencing
  - start next job when one job is complete

Google publishes MapReduce paper

- Platform for reliable, scalable computing.
- All data is sequences of <key,value> pairs.
- Programmer specifies two primary methods:
  - map(k, v) → <k', v'>*
  - reduce(k', <v'>*) → <k', v'>*
  - also partition(), compare(), & others
- All v' with same k' are reduced together, in order.
  - bonus: built-in support for sort/merge!

MapReduce job processing



Use MapReduce in Nutch

- if it's good enough for Google...
  - trust that, while not ultimate arch., workable arch.
- not all Nutch computations are sort/merge
  - but all can easily be made to fit MapReduce
- not optimal for all computations
  - but not far off, and perhaps cheaper in the end
- reliable distributed platform tricky to develop
  - best to focus on a single implementation

Nutch on MapReduce

- Nutch's major algorithms converted in 2 weeks.
- Before:
  - several were undistributed scalabilty bottlenecks
  - distributable algorithms were complex to manage
  - collections larger than 100M pages impractical
- After:
  - all are scalable, distributed, easy to operate
  - code is substantially smaller & simpler
  - should permit multi-billion page collections

Nutch @ Internet Archive

- given 40-node Capricorn rack July 1
  - 4 x 400GB drives per node
  - 1Ghz Via processor, 512MB RAM
- currently using for testing & debugging
- hope to demonstrate 1B page index soon
  - crawling, and/or
  - indexing Archive's pre-crawled data

NDFS benchmark

- using 30-nodes, one drive only
- 10,000 files with random data
  - average 100MB, total 1TB
- 4 hours to create w/ replication=2
- 2 hours to read
- average 5MB/s per node, 1Gb/s overall
  - around 50% of optimal
  - 100Mb network is bottleneck

MapReduce status

- under active development
- usable today
  - have crawled, inverted links & indexed 50M in 72hrs
  - have parsed & indexed 100M archive pages in 96hrs
  - each w/ single command-line!
- in branch:
  - https://svn.apache.org/repos/asf/lucene/nutch/branches/mapred/

Thanks!

Questions?

http://lucene.apache.org/nutch/