OptimizingCrawls

Here are the things that could potentially slow down fetching

1) DNS setup

2) The number of crawlers you have, too many, too few.

3) Bandwidth limitations

4) Number of threads per host (politeness)

5) Uneven distribution of urls to fetch and politeness.

6) High crawl-delays from robots.txt (usually along with an uneven distribution of urls).

7) Many slow websites (again usually with an uneven distribution).

8) Downloading lots of content (PDFS, very large html pages, again possibly an uneven distribution).

9) Others

Now how do we fix them

1) Have a DNS setup on each local crawling machine, if multiple crawling machines and a single centralized DNS it can act like a DOS attack on the DNS server slowing the entire system. We always did a two layer setup hitting first to the local DNS cache then to a large DNS cache like OpenDNS or Verizon.

2) This would be number of map tasks * fetcher.threads.fetch. So 10 map tasks * 20 threads = 200 fetchers at once. Too many and you overload your system, too few and other factors and the machine sites idle. You will need to play around with this setting for your setup.

3) Bandwidth limitations. Use ntop, ganglia, and other monitoring tools to determine how much bandwidth you are using. Account for in and out bandwidth. A simple test, from a server inside the fetching network but not itself fetching, if it is very slow connecting to or downloading content when fetching is occurring, it is a good bet you are maxing out bandwidth. If you set http timeout as we describe later and are maxing your bandwidth, you will start seeing many http timeout errors.

4) Politeness along with uneven distribution of urls is probably the biggest limiting factor. If one thread is processing a single site and there are a lot of urls from that site to fetch all other threads will sit idle while that one thread finishes. Some solutions, use fetcher.server.delay to shorten the time between page fetches and use fetcher.threads.per.host to increase the number of threads fetching for a single site (this would still be in the same map task though and hence the same JVM ChildTask process). If increasing this > 0 you could also set fetcher.server.min.delay to some value > 0 for politeness to min and max bound the process.

5) Fetching a lot of pages from a single site or a lot of pages from a few sites will slow down fetching dramatically. For full web crawls you want an even distribution so all fetching threads can be active. Setting generate.max.per.host to a value > 0 will limit the number of pages from a single host/domain to fetch.

6) Crawl-delay can be used and is obeyed by nutch in robots.txt. Most sites don't use this setting but a few (some malicious do). I have seen crawl-delays as high as 2 days in seconds. The fetcher.max.crawl.delay variable will ignore pages with crawl delays > x. I usually set this to 10 seconds, default is 30. Even at 10 seconds if you have a lot of pages from a site from which you can only crawl 1 page every 10 seconds it is going to be slow. On the flip side, setting this to a low value will ignore and not fetch those pages.

7) Sometimes, manytimes websites are just slow. Setting a low value for http.timeout helps. The default is 10 seconds. If you don't care and want as many pages as fast as possible, set it lower. Some websites, digg for instance, will bandwidth limit you on their side only allowing x connections per given time frame. So even if you only have say 50 pages from a single site (which I still think is to many). It may be waiting 10 seconds on each page. The ftp.timeout can also be set if fetching ftp content. 8) Lots of content means slower fetching. If downloading PDFs and other non-html documents this is especially true. To avoid non-html content you can use the url filters. I prefer the prefix and suffix filters. The http.content.limit and ftp.content.limit can be used to limit the amount of content downloaded for a single document.

9) Other things that could be causing slow fetching:

- Max the number of open sockets/files on a machine. You will start seeing IO errors or can't open socket errors.
- Poor routing. Bad routers or home routers might not be able to handle the number of connections going through at once. An incorrect routing setup could also be causing problems but those are usually much more complex to diagnose. Use network trace and mapping tools if you think this is happening. Upstream routing can also be a problem from your network provider.
- Bad network cards. I have seen network cards flip once they reach a certain bandwidth point. This was more prevalent on, at the time, newer gigabit cards. Not usually my first thought but always a possibility. Use tcpdump and network monitoring tools on the single interface.

That is about it from my perspective. Feel free to add anything if anybody else thinks of other things.