EFFICIENT PROCESSING AND CLASSIFICATION OF WAVE ENERGY SPECTRUM DATA WITH A DISTRIBUTED PIPELINE[1]

I.G. Gankevich, A.B. Degtyarev

Saint Petersburg State University
igankevich@cc.spbu.ru, deg@csa.ru

Processing of large amounts of data often consists of several steps, e.g. pre- and post-processing stages, which are executed sequentially with data written to disk after each step, however, when pre-processing stage for each task is different the more efficient way of processing data is to construct a pipeline which streams data from one stage to another. In a more general case some processing stages can be factored into several parallel subordinate stages thus forming a distributed pipeline where each stage can have multiple inputs and multiple outputs. Such processing pattern emerges in a problem of classification of wave energy spectra based on analytic approximations which can extract different wave systems and their parameters (e.g. wave system type, mean wave direction) from spectrum. Distributed pipeline approach achieves good performance compared to conventional "sequential-stage" processing.

Introduction

The problem of classification of wave energy spectra is both data- and compute-intensive which makes it on one hand amenable to data-centric programming approaches like Hadoop and on the other hand to parallel programming techniques. In the "mapping" phase spectra should be pre-processed and converted to some convenient format and in the "reduction" phase resulting spectra are classified using genetic optimisation algorithm. These steps represent general algorithm for data processing with Hadoop, however, classification algorithm is itself parallel which makes it difficult to program in Java (the language in which Hadoop programmes are usually written). Therefore, we feel that Hadoop is not the most efficient way to solve the problem and a distributed programme which mimics useful Hadoop behaviour should be used instead.

This work is a short preview of a alternative technological framework being developed and it is compared to Hadoop implementation only.

1 Implementation

The NDBC dataset[2] consists of spectra which are sorted by year and station where measurements were made. Data for each spectrum is stored in five variables which are used to reconstruct original frequency-directional spectrum with the following formula:

$$S(\omega,\theta)=\frac{1}{\pi}\left[\frac{1}{2}+r_1\cos(\theta-\alpha_1)+r_2\sin(2(\theta-\alpha_2))\right]S_0(\omega).$$

Here ω denotes frequency, ϑ – wave direction, $r_{1,2}$ and $\alpha_{1,2}$ are parameters of spectrum decomposition and S_0 is the non-directional spectrum [1]; values of $r_{1,2}$, $\alpha_{1,2}$, S_0 are acquired through measurements.

[1] The research was carried out using computational resources of Resource Center Computational Center of Saint Petersburg State University (T-EDGE96 HPC-0011828-001) and partially supported by Russian Foundation for Basic Research (project No. 13-07-00747) and Saint Petersburg State University (project No. 9.38.674.2013 and 0.37.155.2014).

[2] http://www.ndbc.noaa.gov/dwa.shtml

Detailed properties of the dataset used in evaluation are listed in Table 1.

The algorithm of processing spectra is as follows. First, current directory is recursively scanned for input files. All directories are recursively distributed to processing queues of each machine in the cluster. Processing begins with joining corresponding measurements for each spectrum variables into a tuple which is subsequently classified by a genetic algorithm (this algorithm is not discussed in the paper and in fact can be replaced by any other suitable classification algorithm). While processing results are gradually copied back to the machine where application was executed and when the processing is complete the programme terminates. The resulting implementation is shown in Figure 1.

Directory structure can be arbitrary and the only thing it serves is to distribute the data, however, files containing corresponding measurements should be placed in a single directory so that no joining of variables residing in different machines can happen. In this test spectra were naturally sorted into directories by year and station.

Dataset size	144MB
Dataset size (uncompressed)	770MB
Number of wave stations	24
Time span	3 years (2010–2012)
Total number of spectra	445422

Table 1. Dataset properties.

The feature which makes this implementation different from other similar approaches is that both processors and disks work in parallel throughout the programme execution. Such behaviour is achieved with assigning a separate thread (or thread pool) for each device and placing tasks in the queue for the corresponding device in this pool. As tasks that read from the disk complete they produce tasks for CPUs to process this data and place them into the processor task queue. In similar way when data processing tasks complete they place tasks to write the data into the disk task queue. In similar vein via a separate task queue network devices transmit the data to a remote node. So, each device has its own thread (or thread pool) and all of them work in parallel by placing tasks in each other's task queues. Since tasks “flow” from one queue to another and queues can reside on different machines this approach is called distributed pipeline.

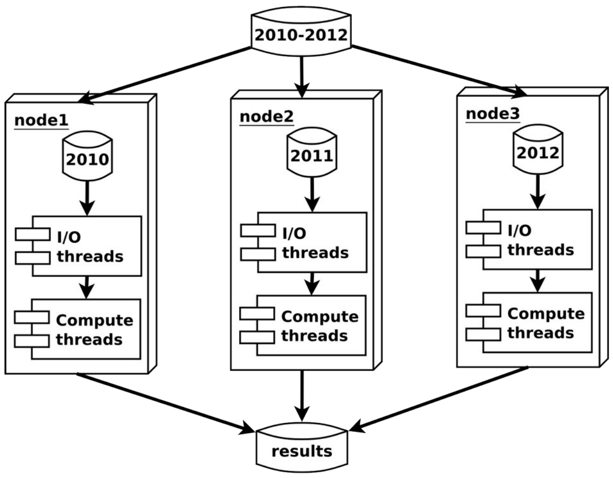

Figure 1. Implementation diagram for distributed pipeline.

2 Evaluation

The system setup which was used to test the implementation consisted of commodity hardware and open-source software (Table 2) and evaluation was divided into two stages. In the first stage Hadoop was installed on each node of the cluster and was configured to use host file system as a source of data so that performance of parallel file system which is used by default in Hadoop can be factored out from the comparison. To make this possible the whole dataset was replicated on each node and placed in the directory with the same name. In the second stage Hadoop was shut down and replaced by newly developed application and dataset directories were statically distributed to different nodes to nullify the impact of parallel file system on the performance.

Component	Details
CPU model	Intel Q9650
CPU clock rate (GHz)	3.0
No. of cores per CPU	4
No. of CPUs per node	1
RAM size (GB)	4
Disk model	ST3250318AS
Disk speed (rpm)	7200
No. of nodes	3
Interconnect speed (Mbps)	100

Component	Details
Operating system	Debian Linux 7.5
Hadoop version	2.3.0
GCC version	4.7
Compile flags	-std=c++11

Table 2. Hardware and software components of the system.

In the test it was found that Hadoop implementation has low scalability and maximum performance of approx. 1000 spectra per second and alternative implementation has higher scalability and maximum performance of approx. 7000 spectra per second (Figure 2). The source of Hadoop inefficiency was found to be temporary data files which are written to disk on each node. These files represent sorted chunks of the key-value array and are part of implementation of merge sort algorithm used to distribute the keys to different nodes. For NDBC dataset the total size of these files exceeds the size of the whole dataset which appears to be the consequence of Hadoop not compressing intermediate data (the initial dataset has compression ratio of 1:5, see Table 1). So, the sorting algorithm and careless handling of compressed data led to performance degradation and inefficiency of Hadoop for NDBC dataset.

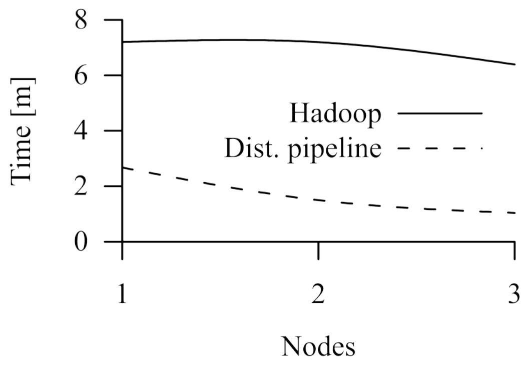

Figure 2. Performance of Hadoop and distributed pipeline implementations.

The sorting is not needed to distribute the keys and in the alternative implementation directory hierarchy is used to determine machine for reduction. For each directory a separate task is created which subsequently creates tasks for each sub-directory and each file. Since each task can interact with its parent when the reduction phase is reached reduction tasks are created on the machines where parents were executed previously.

Conclusions and future work

No redundant sorting nor any kind of temporary files are used in the alternative implementation which allows it to scale well and show better performance compared to Hadoop approach. The future work is to incorporate dynamic distribution of files to hosts and fault tolerance into the implementation.

References

[1] Marshall D. Earle. Nondirectional and Directional Wave Data Analysis Procedures (NDBC Technical Document 96-01), 1996,
URL: http://www.ndbc.noaa.gov/wavemeas.pdf.