Abstract—MapReduce is a domain-independent programming model for processing data in a highly parallel. The MapReduce program can be automatically concurrent executed in large-scale commodity machines. Apache’s Hadoop is an implementation of MapReduce and has been applied successfully for file based datasets. This paper proposes to utilize the parallel and distributed processing capability of Hadoop MapReduce for handling the data assimilation using Sequential Monte Carlo methods in wildfire spread simulation. The experiment results demonstrate that our approach significantly boosts the performance.

1. INTRODUCTION

Every year, wildfires incur sudden and rapid damages to and losses of natural forest resources, endangered species, human lives, and properties. Approximately 11,000 communities close to federal land have a high likelihood of being subject to threat from wildfire [1]. During 2007’s wildfire season, over 85,500 fires across the U.S. burned at least 9.3 million acres of land. It cost 1.8 billion dollars in the effort to fight wildfires and a potential 2.5 billion dollars in insured losses in California alone [2]. Therefore, it is necessary to develop the wildfire model to study and predict the wildfire spread. Over the years, several major models were developed, such as FARSITE [3], BehavePlus [4], and DEVS-FIRE [5] and Hfire [6]. The accuracy of those models always depends on the correctness of the model inputs, including the GIS data, the fuel data and the weather data. Unfortunately, due to the dynamic and complex nature of wildfire, it is impractical to obtain all these data with no error. For example, the weather data usually come from the local weather stations in a time-based manner; it is unchanged until the next weather data arrive. The GIS data and fuel data are constrained by their spatial resolutions [7].

To improve the accuracy of wildfire spread simulations, data assimilation methods that assimilate sensor data from real wildfires are needed. Previous work developed a data assimilation method using Sequential Monte Carlo (SMC) methods based on the DEVS-FIRE spread model [7][8], which showed that the data assimilation system was able to improve the accuracy of wildfire simulation by assimilating real time data from sensors. Therefore, in order to improve the data assimilation performance, we want to develop a parallel and distributed computing method in DEVS-FIRE spread simulation based on SMC methods.

On the other hand, for the purpose of processing big data, Dean and Ghemawat from Google firstly presented a parallel programming model, MapReduce, which was a framework for processing huge data sets on certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster [9][10]. The briefly described the MapReduce programming model: The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: Map and Reduce.

In this paper, we propose a parallel and distributed framework use Hadoop MapReduce to handle the data assimilation in wildfire simulation based on SMC methods. The rest of this paper is organized as follows. Section 2 introduces the work related to MapReduce combine with the traditional methods. Section 3 formalizes the new MapReduce framework application of particle filters in DEVS-FIRE. Experimental analysis is given in Section 4 and Section 5 draws the conclusions and points out future work.

2. RELATED WORK

Followed by Google’s work, many implementations of MapReduce emerged and lots of traditional methods combined with MapReduce have been presented until now [11].

• Implementations of MapReduce

Apache Hadoop is a software framework that helps constructing the reliable, scalable, distributed systems
Phoenix is a shared-memory implementation of Google's MapReduce model for data-intensive processing tasks [13]. Mars is a MapReduce framework on graphic processors (GPUs) [14]. Twister is a lightweight and Iterative MapReduce runtime system [15].

- Traditional methods combined with MapReduce

Apache Mahout can help to produce implementations of scalable machine-learning algorithms on Hadoop platform [16]. Menon et al. gave a rapid parallel genome indexing with MapReduce [17]. Blanas et al. proposed crucial implementation details of a number of well-known join strategies for log processing in MapReduce [18]. Ene et al. developed fast clustering algorithms using MapReduce with constant factor approximation guarantees [19]. Lin et al. presented three design patterns for efficient graph algorithms in MapReduce [20].

Moreover, MapReduce is rarely employed in the field of Systems Biology. In [21], the authors investigate whether a MapReduce approach utilizing on-demand resources from a Cloud is suitable to perform simulation tasks in the area of Metabolic Flux Analysis (MFA). Also, the authors introduced an implementation of a simple MapReduce method for performing fault-tolerant Monte Carlo computations in a massively-parallel cloud computing environment shown in [22].

3. **DEVS-FIRE & SMC MAPREDUCE APPROACH**

The previous works have already presented the data assimilation system in DEVS-FIRE to improve wildfire spread prediction by utilizing SMC methods. The results showed that the data assimilation system was able to improve the accuracy of wildfire simulation by assimilating real time data from sensors. More details about overview of SMC methods, DEVS-FIRE Spread simulation and data assimilation using SMC methods in DEVS-FIRE simulation can be found in [5] [7] [8] and are omitted in this paper. In summary, the major steps of particle filters based on sampling importance resampling are described below:

- Step 1: initialize N particles.
- Step 2: calculate importance weights.
- Step 3: normalize importance weights.
- Step 4: resampling.
- Step 5: predict new particles for future use.

Step 6: go to Step 2 to execute the next time step.

Based on the dynamic system, in the particle filter algorithm, step 1 initializes particles. With time advances, step 2 to step 5 are executed as shown in Figure 1.

![Figure 1: Particle filters algorithms of case study](image)

The MapReduce architectural pattern has evolved as a generic, domain-independent processing method for large amounts of data. Two functions: map and reduce, are required to be implemented by the user with the following prototypes [10]:

- `map(k1, v1) • list(k2, v2)`
- `reduce(k2, list(v2)) • list(v2)`

Which list denotes a list of objects, k1 and k2 represent key types, v1 and v2 are value types. The input key/value pairs (k1, v1) are pairwise independent, thus, map can be invoked in parallel for all pairs, yielding an intermediate list of mapped (k2, v2) pairs. As an outstanding feature, MapReduce jobs may be defined by using native libraries such as C++
and Java. For this paper, all the experiments use Java. More information about MapReduce can be found in [9] [10]. Based on the major steps of particle filters and the basic MapReduce prototypes, we introduce our new definition of MapReduce framework application of particle filters in DEVS-FIRE. The Algorithm 1 shown the map part, which key is the index of the particle, the value include all the necessary data, such as the GIS data, weather data (wind speed and wind direction), ignition points and sensor data. The Algorithm 2 shown the reduce part, in our framework, we use reduce part do nothing, that means the put the run DEVFS-FIRE simulation part and the weight computation part in Map, then do nothing in Reduce and put weights normalization part and resampling part in HDFS. The reason why we have to put weights normalization and resampling in HDFS is for the systematic resampler, the sampler require information of all the particles. And the reason why we do nothing in reduce part is we will continue to find the sampling way which does not require information of all the particles, this is considered as a topic for future work.

Algorithm 1: Map (key, value)

Input:
//key: Particle index
//value: S= {GIS data, Weather data, Ignition Points, Sensor Data}

Output:
//key': Particle index
//value': {Fire front, weight}

begin
1 let key' = key = Particle index
2 let value as the input of DEVFS-FIRE run a simulation
3 value' = the fire sharp and the particle weight
4 output.collect (key0, value0);
end

Algorithm 2: Reduce (key, V)

Input:
//key': Particle index
//value': {Fire front, weight}

Output:
//key': Particle index
//value': {Fire front, weight}

Do nothing

Figure 2: MapReduce SMC algorithms of case study

Figure 2 shown the MapReduce SMC algorithm: run the DEVFS-FIRE spread simulation in different node (computer), also add the graph noises and compute the weight in same node, all those parts are parallel worked. Then as we mentioned before, since the resampling part have to get the information about all the particles, we put the weights normalization and resampling parts in HDFS.

4. EXPERIMENTS AND ANALYSIS

First of all, we continue used the identical-twin experiment, which is widely used in data assimilation research, to evaluate the data assimilation system of DEVFS-FIRE. More details about identical-twin experiment can be found in [7]. In this experiment, we chose to use the incorrect wind conditions as the “error” data. The real wind speed and direction are 8 (m/s) and 180 degrees (from south to north) with random variances added every 10 minutes. The variances for the wind speeds are in the range of –2 to 2(m/s) and the wind direction to be exactly the same as the real wind direction (Table 1). For the
sensor deployment, we employed a regular sensor deployment schema and design our experiment as follow, uses a uniform fuel model (fuel model 7) and zero slope and aspect. The simulations are run for 5 steps (hours), the weather changed every 30 mins.

<table>
<thead>
<tr>
<th>Case</th>
<th>Speed Error</th>
<th>Direction Error</th>
<th>Speed Real</th>
<th>Direction Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>6±2</td>
<td>No error</td>
<td>8±2</td>
<td>180±20</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Experiment sets of wind factor

Secondly, all experiments run under the super computer named Cheetah, which has 14 nodes, 160 computing cores, 32 CPUs and 264 GB system memory. 7 nodes equipped with Nvidia GTX 285, 485, or Tesla c2075 Graphic processing units for CUDA development 6TB disk storage. The software package which we use is Apache Hadoop Cloud Computing Software. Hadoop version 1.0.1 and Java 1.6.0.12 are used as MapReduce system.

Finally, In order to test the performance, we use four nodes for MapReducePF and one of those four nodes for CentralizedPF, we use the particle number is: 50 particles, 100 particles and 200 particles.

The Figure 3 display the filtered fires (displayed in yellow) after 5 steps of simulation, compared with the real fire (displayed in red), and the simulated fires (displayed in blue), the particle number is 100 particles.

Figure 3 Comparisons of real fire, simulated fires, and filtered fires

Figure 4 display the result performance for the single step (1 hour) DEVS-FIRE spread simulation based on SMC method use different particle numbers: 50 particles, 100 particles and 200 particles. We can see the simulation time almost same when we just use 50 particles (CentralizedPF: 120s and MapReducePF 122s), but with increase the number of particle, the MapReducePF worked better and better.

Figure 4 the performance of single step

Figure 5 display the result performance for the five steps (5 hour) DEVS-FIRE spread simulation based on SMC method use different particle numbers: 100 particles and 200 particles. The result shown the MapReducePF did a great work when we apply 200 particles (CentralizedPF: 4132s and MapReducePF 1436s)

Figure 5 the performance of five steps

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a parallel and distributed processing capability of Hadoop MapReduce for handling the data assimilation using Sequential Monte Carlo methods in wildfire spread simulation. The experiment results showed the MapReducePF get the great performance when user applies large number particles. Based on the theory of MapReduce model, we believe we can get the better performance result when we apply more particles. This work builds a foundation where future work can be carried out. Future work includes find a new sampler way which does not require information for all the particles and then use reduce part to finish it and continue to build the MapReducePF, make MapReducePF not only just for DEVS-FIRE spread.
simulation, but also can apply to other SMC based methods.

REFERENCES


[22] Monte Carlo simulation of photon migration in a cloud computing environment with MapReduce. Guillem Pratx and Lei Xing, Journal of Biomedical Optics 16(12), 125003 (December 2011)