Cloud computing based bushfire prediction for cyber–physical emergency applications

Saurabh Garg\textsuperscript{a}, Jagannath Aryal\textsuperscript{b}, Hao Wang\textsuperscript{a}, Tejal Shah\textsuperscript{e}, Gabor Kecskemeti\textsuperscript{c}, Rajiv Ranjan\textsuperscript{a,*}

\textsuperscript{a} School of Engineering and ICT, University of Tasmania, Hobart, Australia
\textsuperscript{b} Discipline of Geography and Spatial Sciences, School of Land and Food, University of Tasmania, Hobart, Australia
\textsuperscript{c} Department of Computer Science, Liverpool John Moores University, United Kingdom
\textsuperscript{d} Chinese University of Geosciences, China
\textsuperscript{e} Newcastle University, United Kingdom

HIGHLIGHTS

- A novel cloud based framework to deploy/process fire models within a deadline.
- A novel scheduling mechanism integrating user’s req. and minimising resource usage.
- A case study using Tasmania Bushfire Model for evaluating the Cloud based framework.

ABSTRACT

In the past few years, several studies proposed to reduce the impact of bushfires by mapping their occurrences and spread. Most of these prediction/mapping tools and models were designed to run either on a single local machine or a High performance cluster, neither of which can scale with users’ needs. The process of installing these tools and models their configuration can itself be a tedious and time consuming process. Thus making them, not suitable for time constraint cyber–physical emergency systems. In this research, to improve the efficiency of the fire prediction process and make this service available to several users in a scalable and cost-effective manner, we propose a scalable Cloud based bushfire prediction framework, which allows forecasting of the probability of fire occurrences in different regions of interest. The framework automates the process of selecting particular bushfire models for specific regions and scheduling users’ requests within their specified deadlines. The evaluation results show that our Cloud based bushfire prediction system can scale resources and meet user requirements.

1. Introduction

Due to human activities and climate changes, bushfires have increased dramatically in the last few years \cite{1,2}. Every year thousands of acres of forest area is destroyed that includes not only loss of several animal and plant species but also human lives and properties. For example, during the Black Saturday 2009 fire, one of the most significant disasters in Australian history, 173 people lost their lives and 2298 homes were destroyed along with several other environmental losses. Therefore, forest fires are considered to have serious environmental and socioeconomic effects that are aggravated due to increase in climatic temperatures.

In response to this, several fire prediction and behaviour models have been developed during the last four decades to reduce the after-effects of bushfires. Several desktop based fire simulation tools are available that incorporate such models. Some well known tools are SiroFire simulator \cite{3}, BehavePlus \cite{4}, FARSITE \cite{5}, Spark \cite{6} and HFire \cite{7}.

In general, the estimation of fire risk and fire spread are dependent on several geospatial input data sources, some of which are dynamic and change with time. For example, weather data changes with time and space. Furthermore, each user may want to do computation for a different geographic extent and at different spatial resolutions which defines the amount of input data, storage
and computational resources required. Due to the complexity of 
computation involving data of different formats, sizes and from 
different sources, the data processing is not a trivial task and may 
involve expensive investment in terms of computational hardware, 
software and deep computing skills. Furthermore, although most 
of these simulators help us to understand in an efficient way and in 
an accurate form, it is still quite manual and time consuming 
from the perspective of a user who has little knowledge about underlying infrastructure. 

Some of these drawbacks were addressed in fire management 
systems such as Virtual Fire [8] which allows an easy to use web 
to interface to access and visualise different data sets including on-
demand fire behaviour simulations. Most of these fire prediction tools and technologies are designed to either work on single desk-
top machines, clusters or limited high performance computing. 
Thus, these systems suffer from low scalability and availability [9]. 

Recently, several researchers have begun to see Cloud comput-
ing technology as a cost-effective and highly scalable solution to 
Big Data problems in different domains such as geospatial sciences and 
threat management [10]. Cloud computing provides elastic and 
on-demand access to an almost infinite amount of storage, net-
work and computational resources [11]. Due to the pay-as-you-go 
motivation, Cloud computing resources, users do not have to maintain 
expensive computing facilities or face up-front cost. Thus, Cloud 
computing infrastructure allows elastic storage and computational 
capabilities for managing a fluctuating number of user requests. 
Some researchers have already showed the benefits of Cloud comput-
ing which provides dynamic and scalable computing and stor-
age infrastructure [12,13]. 

Despite so many benefits offered by Cloud computing, the solu-
tions available for tackling real geo-spatial science problems are 
limited. Some studies used Cloud computing for storing and man-
aging a large amount of geo-spatial data but using their infra-
structure with a strong manual component [14]. Others only used 
Cloud computing to increase computing capacity [15,16]. Most of 
this work does not offer an effective solution as it neglects either 
user requirements (e.g. deadline) or still has a large manual compo-
nent. During emergency situations such as bushfires, even a small 
delay can result in the loss of many lives. Thus, making these solu-
tions unpractical for time constraint cyber–physical systems [17]. 

Over the last several decades, there have been several deadline 
based scheduling algorithms for scheduling applications in a Cloud 
computing environment [18,19]. As they are developed for specific 
application domains, they cannot be applied directly to scheduling 
of bushfire prediction application. 

To overcome the limitations of previous bushfire prediction 
systems, we propose a Cloud based fire prediction service 
framework that not only allows access for multiple users simulaneously but also considers the requirements of each individual user. The proposed service also minimises the cost 
by keeping Cloud resource usage to a minimum. The proposed 
framework also allows users to use different bushfire models 
according to their area of interest. We also evaluated the proposed 
framework using a bushfire case study from Tasmania, Australia. 

In summary, the main contributions of this work are:

• A novel architectural framework which can allow deployment of 
  fire models considering users’ requirements in terms of area 
  and time. The framework allows integration of new fire models. 
• A novel deadline based scheduling algorithm for efficient 
  bushfire prediction. 
• A case study using the Tasmania Bushfire Model for evaluating 
  the Cloud based framework. 

In the next section, we discuss requirements for a fire prediction 
service. Then in the subsequent sections, we describe the design 
and implementation of the proposed framework with evaluation 
and results. Then we discuss related work on fire prediction 
services and their comparison with the architecture of the proposed framework. Finally, we present conclusions and future 
directions.

2. Scenario and requirements

Our aim is to design a framework that allows deployment of 
fire-prediction models with acquisition of data from different web-
services in order to satisfy users’ quality of service in terms of a 
deadline at minimal possible cost (i.e. number of machines used). 
In the current scenario, most of the acquisition and processing of 
data for fire prediction is done manually. Such computations are 
also done either on a user’s own desktop computer or on a local 
cluster which is limited in size and shared with many other users 
that further slow down the process. Sometimes, one has to deploy 
different models for different regions of interest. Such challenges 
slow down not only many critical research studies but also, in real 
life, can result in loss of public resources and even lives. Therefore 
we aim to facilitate such studies and on-demand fire prediction 
using scalable Cloud computing resources. 

Based on the user’s needs in terms of fire-predictions, the 
following further requirements of a Cloud computing software 
service are identified:

• Scalability: As the service may be accessed by several users 
  across the globe, it needs to scale accordingly to keep response 
time of accessing the service to a minimum. The response time 
threshold for accessing the service should be limited by the 
maximum response time experienced by users themselves. 

• Cost and time effective: The main aim of the service is to 
decrease the overall time for users who have to download 
large files from the different repositories and pre-process before 
extracting their real benefit. Given that most environmental 
data products are free, the services should be offered in a cost 
effective manner so that users see value in using such services. 

• Context aware and on-demand service: Depending on a user’s 
  context, different processing will be selected by the system. For 
  example, if a user needs the processed data for a certain region 
in a certain amount of time, then processing applications, input 
images (resolutions) and parallelisation is used accordingly to 
decrease the computation time. Different fire prediction models 
need to be utilised [20]. 

• Support of massive data storage and processing: Given that 
environmental processes need large amounts of data to be 
downloaded, an appropriate scalable storage service needs to be 
selected so that the time taken for data transfer, and read and 
write operation can be minimised. Based on user requirements 
and data, the required amount of computational resources 
should be acquired on-demand. 

• Security: To avoid spamming or denial of service attacks, there 
  should be an appropriate security mechanism for accessing 
different services of the system. All services must be accessed 
only by registered users.

3. Proposed system framework

3.1. Usage scenario

The system aims to provide Cloud based Fire Prediction (CFP) 
services required by the end user after acquiring data sets from 
different web services such as NASA. A typical scenario of the 
proposed CFP service is given in Fig. 1 with high level steps 
for one cycle of service provided by the proposed system to a 
user. The proposed service is designed to work in a master–slave 
manner where FirePredict Broker acts as a master node while Local 
FireWorker service nodes act as slave/worker nodes. 

A user will send a request to FirePredict Broker which 
analyses all the metadata provided by the user with his/her 
time constraints. Users provide details such as area of interest 
and processing required. Users might give a deadline by which
they would like to get processing completed and results. The FirePredict broker service will interact with the data service to get the pre-processed data needed to fulfil the user interest. In general the pre-processed data is much smaller than the original ones which contain much more information than required for processing. Thus, data preparation is essential before it can be processed. Other than data preparation, this component of the system keeps track of which data have been downloaded from different data repositories and by which Cloud service site. Data services pass the urls (data location) to the FirePredict Broker. Local FireWorker Service Nodes are hosted geographically at different Cloud computing sites. This component is responsible for interacting with different environmental data services to acquire data based on the user requirements. This component also deploys the required fire prediction application in the Cloud environment and sends the results location back to the FirePredict broker which passes this information to the user with the cost incurred in the request processing.

3.2. Architecture and design

The full component details of the CFP service are given in Fig. 2. The CFP service has mainly two types of service, i.e. the user services and the core services. The user services includes the user interface, authorisation/authentication service and accounting service. The core services consist of FirePredict Broker, Request Analyser service, Data Service, Local FireWorker services, request allocation and management service. Each of the services can run on different machines independently. FirePredict Broker service is the key component of the system that derives all other components of the system. Its main functionality is to interact with users and understand their requirements and pass the request over to other components after deciding the most appropriate Cloud site to download and process the data based on users’ time constraints.

3.2.1. User services

The user services hide all the internal components of the CFP service and implement all the services that are needed by users to interact with the system. To use the system services, the user has to first login with username and password which are checked by authentication and authorisation services. By interacting with this service, the user interface has responsibility for checking whether a user is authenticated or not. The user’s historical usage of the CFP services and processing cost incurred to each user is maintained by the Accounting Service. Using the Accounting Service, the user can also know the status of each request. The Accounting Service also does the cost analysis where cost is computed based on the amount of Cloud resources that are needed to be leased for downloading, storing and processing data. In each request, the user passes the details such as the area of interest and deadline through the User Interface to the Accounting Service which is passed to the FirePredict service for further processing. At the end of the processing, the url for downloading the processed data will be sent to the user with a bill for incurred cost.

3.2.2. Core services

FirePredict Broker Service has responsibility similar to that of a typical Cloud broker, i.e. to interact with users, understand their requirements and schedule processing based on users’ time constraints [21]. The FirePredict Broker service is hosted as a software service on Cloud infrastructure. All the requirements and constraints are checked by the broker using the Request Analyser service. This service first checks what data is needed for the processing required by the user. This service then checks whether the data or part of the data has already been downloaded by interacting with Data Service. If data has already been downloaded, this layer will check at which Local FireWorker service data exits and then forward these details to the FirePredict Broker which passes them to the Request Allocation and Management service for further processing. Fig. 3 further illustrates the interaction between different entities (aka. services).

The Request Allocation and Management service controls the distribution of requests across multiple Local FireWorker Cloud service sites. This service can be integrated with different allocation policies which takes into account the time taken to download the data for processing and cost incurred in storage and processing. By default, the request will be sent to the service site which has minimum data download time. The Request Allocation and Management service also monitors the progress of each request and passes this information to the Accounting Service.

The Data Service is a directory service which maintains the meta-data of actual geospatial data including the url from where data can be downloaded. If the data is already downloaded and stored in a Cloud processing site, it will also maintain this information. In case data is not downloaded, this service interacts with different data repositories to prepare the data for download and forwards the final url to the request analyser. This service helps the system to avoid multiple processing of data by different users. This will indirectly reduce the load on data services by acting as another layer of caching. As it will also track where pre processed data is located, it will help in avoiding the cost of processing the
Local FireWorker Cloud Services are software services hosted on different Cloud Infrastructure (aka IaaS) which are geographically distributed. They will receive the information from the Request Allocation service about user requirements. FireWorker services check how much Cloud resource is available and how much to lease to fulfill the end user request. These services will use advanced scheduling mechanisms to minimise the infrastructure cost and computation time. They will regularly monitor the resource usage and application processing to minimise any case of failure which can cause unnecessary delays. They can decide which resource should be leased depending on its load. For example, if there are many processing requests with limited time availability, then these services can decide to lease larger Cloud virtual machines with much more memory.

Local FireWorker Cloud Service consists of the following components:

- **FireModel Catalogue** is a directory that maintains meta-data of different fire prediction models and virtual machine images. The meta-data helps in deciding which fire prediction model should be used for a particular geographical location in which the user is interested. The meta-data also consists of the execution profile of different fire-prediction models which help in predicting their processing requirements.

- **Data Acquisition** component helps in downloading the data required for processing the user request and storing at the local Cloud site.

- **Request Scheduler** decides when and where each request will be executed. It makes the decision based on the processing requirements of a fire-prediction model, the user’s time constraints and available virtual machines. It also decides how many virtual machines should be utilised for processing a user’s request.

- **VM Manager** is responsible for initiating and stopping virtual machines.

- **Job Manager** is responsible for the deployment and the execution of a fire prediction model on a virtual machine.

Fig. 4 illustrates how requests are processed by each Local FireWorker. Based on the request, a FireWorker downloads the required data for processing using DataAcquisition if it is not already stored within the local Cloud storage. After data download is done, the FireWorker will forward the user’s request with location of downloaded data to the RequestScheduler component which decides when and on which Virtual Machines (VMs) the request will be processed. To make this decision, RequestScheduler requires the resource requirements and performance profile of the fire model which needs to be run to fulfill a user’s request. This information is sent by FireModelCatalogue. Based on the scheduling decision, RequestScheduler initiates the required VMs which will execute Fire Models in the form of parallel jobs. The parallel jobs are managed by JobManager which monitors the execution of the jobs and redeploy if a VM fails.
4. Case study: Tasmanian bushfire prediction model

To show applicability of the proposed Cloud based software service architecture for the Fire Prediction service, this section presents a short case study where a bushfire prediction Cloud service is built to serve multiple users. To evaluate the performance of the CFP service and provide a proof of concept of its architecture, we implemented a prototype with Nectar Cloud as the Local FireWorker cloud site.

In this case study, users submit their requests for fire prediction in a certain area of Tasmania with their time constraints in terms of a deadline to the FirePredict Broker through a user interface. More details are given in the following sections.

4.1. Prototype implementation

CFP has been implemented in Java in order to be portable over different platforms such as Windows and Unix operating systems. As our aim in this case study is to give a proof of concept, we just consider limited functionality of FirePredict Broker’s services and one Cloud processing site. It consists of three layers: user interface (user service), FirePredict Broker and one Local FireWorker service. The Local FireWorker service is responsible for managing and scheduling fire prediction requests (job) to different virtual machines where a slave daemon is running to handle actual execution of the job. The slave nodes process the requests on a first-come-first serve basis. The slave nodes do not interact with each other but only with the FireWorker service. The communication between virtual machines and the FireWorker service is implemented using Java sockets. The connections are kept active only when both FireWorker service and a slave are active; this feature keeps the FireWorker and slaves loosely coupled and independent. The FireWorker regularly checks the status of slaves. The user interface is built using Java Swing library. The details of the Fire Prediction Model (application) and scheduling algorithm utilised by the system are discussed in the following sections.

4.2. Bushfire prediction model

We develop a simple fire model for the Tasmania region based on a binary logistic regression as a proof of concept. This model assesses the probability of fire occurrence using the non-linear relationships among fire danger indices considered in this study. The topographic characteristics for a period of one year (July, 2014–July 2015) are used in developing the model. In this model, the Forest Fire Danger Index (FFDI) and Fire Weather Index (FWI) are considered, which incorporate climatic conditions data e.g. weather, temperature, relative humidity, wind speed and precipitation. Topographic characteristics of the study area, e.g. elevation, slope, and aspect, are considered as explanatory variables in developing the model. These data are extracted from the ASTER Global Digital Elevation Model (ASTER GDEM) with 30 m spatial resolution. Climatic conditions data are obtained from the Bureau of Meteorology, Australia’s national weather, climate and water agency.

The logistic regression model is expressed as:

\[
P = \frac{1}{1 + e^{-\left(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n\right)}}
\]

(1)

Where, \(P\) = Probability of the event, \(\beta_0 = \text{Intercept, } \beta_1 \ldots \beta_n = \text{Regression coefficients.}\)

Correlations among the variables were observed before developing the model. Considering occurrence of fire as \(P = 1\) and non-occurrence as \(P = 0\), the probability of fire occurrences is given by:

\[
P = \frac{1}{1 + e^{-21.610 + 0.198 \times FFDI - 0.028 \times FWI - 0.001 \times Ap + 0.064 \times SL + 19.903 \times Elv - 0.108 \times Lc}}
\]

(2)

In the equation, \(P\) is the probability that a point corresponds to a fire ignition. Ap, Sl, Elv, Lc represent Aspect, Slope, Elevation and Land cover, respectively. FFDI is the forest fire danger index and FWI is the fire weather index. The obtained logistic regression model showed that the most influential variable explaining the spatial patterns of fire was Elevation (\(\alpha = 19.903\)) Slope (\(\beta = 0.604\)), followed by FFDI (\(\gamma = 0.198\)), Land cover, and FWI. The details on FFDI and FWI are available in works by Noble et al. [22] and Beccari et al. [23]. Upon request source codes for the developed model can be made available from the authors.

4.3. Scheduling algorithm

As discussed in the previous section, the main function of the FireWorker Service is to map requests to slave nodes based on their capacity and user requirements. Within the scheduling module of FireWorker Service the following functionalities are achieved:

- The splitting of the user’s request into several partitions or jobs, which is determined by the capacity and the size of input data.
- Machines are added only if the number of machines is not enough, which means machines should be added one by one based on the requests’ requirements to avoid wastage of resources.
- If the capacity available on the currently used machines is enough to complete a request within its deadline, then the request is queued for processing in the currently available slave nodes.

The pseudo code of the scheduling algorithm is given below:
Firstly, the capacity of each computer is assumed to be known, and how many jobs can be executed by each machine in this time. This number of jobs depends on the capacity of the machines. Therefore, for partitioning the request, the area of interest will be divided into different subareas where each subarea’s fire prediction model will be computed. As shown in Fig. 5, in order to finish parallel computing, the request (for an area of interest) should be divided into several jobs (for each subarea) that do not need to communicate any data for processing and thus can run independently on different processors.

Jobs in the figure indicate how many sub-tasks should be created to finish the fire probabilities for a given area. For example, the size of this area above is L*L. Let a user want to get this size of this area above is L*L. Let a user want to get this size of this area above is L*L. Let a user want to get this size of this area above is L*L. Let a user want to get this size. Therefore, based on the terminology, the pseudo code for partitioning each request is described in Algorithm 2.

### Algorithm 1: Bushfire-Prediction Request Scheduling Algorithm

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: User Request list = RList; // details of the area of interest in terms of latitude and longitude, and deadline</td>
<td></td>
</tr>
<tr>
<td>RList = Collect user requests in current time; // Sort the requests by deadline</td>
<td></td>
</tr>
<tr>
<td>SortedReqList = Sort(RList);</td>
<td></td>
</tr>
<tr>
<td>for ri ∈ SortedReqList do</td>
<td></td>
</tr>
<tr>
<td>// find out the area for which data needs to be processed</td>
<td></td>
</tr>
<tr>
<td>CalculateAreaReq(ri);</td>
<td></td>
</tr>
<tr>
<td>Based on the area, calculate number of jobs (or partitions) i.e. NumJobs(ri);</td>
<td></td>
</tr>
<tr>
<td>RemainTime = Deadline(ri) - CurrentTime; // find the time remaining for returning results to user</td>
<td></td>
</tr>
<tr>
<td>// check whether time available is sufficient to process the job</td>
<td></td>
</tr>
<tr>
<td>if RemainTime &gt; 0 &amp; RemainTime &gt; MinExecutionTime(Job(ri)) then</td>
<td></td>
</tr>
<tr>
<td>for j ∈ {1, NumJobs(ri)} do</td>
<td></td>
</tr>
<tr>
<td>VM_withSpace = Find an existing virtual machine that can process the job before deadline;</td>
<td></td>
</tr>
<tr>
<td>if VM_withSpace exists then</td>
<td></td>
</tr>
<tr>
<td>submit the job VM_withSpace;</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>initiate a new machine and submit the job to this machine;</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
<tr>
<td>Add the resulting allocation to AllocationList;</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>

### Algorithm 2: NumJobs(Request Ri)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: User Request = Ri; // details of the area of interest in terms of latitude and longitude, and deadline</td>
<td></td>
</tr>
<tr>
<td>X = Remaining area for which processing has to be done;</td>
<td></td>
</tr>
<tr>
<td>M[i] = Capacity of each computer;</td>
<td></td>
</tr>
<tr>
<td>T = Deadline for the user;</td>
<td></td>
</tr>
<tr>
<td>Y = area for which fire probability will be computed on a worker node;</td>
<td></td>
</tr>
<tr>
<td>while X &gt; 0 do</td>
<td></td>
</tr>
<tr>
<td>Y = M[i]*T;</td>
<td></td>
</tr>
<tr>
<td>X = L*L - Y;</td>
<td></td>
</tr>
<tr>
<td>create a job to process Y amount of area and add to job list;</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.4. Partitioning algorithm for bushfire prediction model

The fire prediction model considered for this case study computes the probability of fire at a given point and the probability of fire occurrence at a given point is independent of another point in a region of interest. In other words, to compute fire probabilities for a given area of interest, each point in the area can be considered separately. Therefore, for partitioning the request, the area of interest will be divided into different subareas where each subarea’s fire prediction model will be computed. As shown in Fig. 5, in order to finish parallel computing, the request (for an area of interest) should be divided into several jobs (for each subarea) that do not need to communicate any data for processing and thus can run independently on different processors.

Jobs in the figure indicate how many sub-tasks should be created to finish the fire probabilities for a given area. For example, the size of this area above is L*L. Let a user want to get this computation done within T time. If a Local FireWorker service has to finish the whole area calculation in T time (the user’s deadline), we need to compute how many machines are needed for this area and how many jobs can be executed by each machine in this T time. This number of jobs depends on the capacity of the machines. Firstly, the capacity of each computer is assumed to be known, and we mark it as M[i]. The whole area of this map is L*L (the total number of jobs). Therefore, based on the terminology, the pseudo code for partitioning each request is described in Algorithm 2.

#### 4.5. Nectar cloud infrastructure

Nectar Cloud is a community research Cloud environment which provides flexible scalable computing power to all Australian researchers. The infrastructure is implemented and managed using the OpenStack cloud computing framework. To create virtual machines and run the experiments, we utilised application EC2 APIs. The details of virtual machines initiated are given in subsequent individual experimental sections.

#### 4.6. Profiling fire model

To meet the user’s time constraints in regard to the processing of the request, the FireWorker’s scheduler should know the execution time of the fire model for the given data. Thus, we need to profile the execution time of the fire model on multiple parallel (distributed) machines. For the experiments, the daily weather data was collected from July 2014 to July 2015 for Hobart weather observation stations. Local noon measurements of temperature (C), relative humidity (%), wind speed (km/h) and daily total precipitation (mm) were used to calculate the component codes and the Fire Weather Index (FWI) for each station. The Drought factor index was collected as well to calculate the Forest Fire Danger Index (FFDI) for each station. A digital elevation model (DEM) was used to get the topographic information such as height. We chose the area located near Hobart (Tasmania) for computing different requests and amount of data to be processed. For example, 8 MB means the data source about Hobart within a range of 30 km; 20 MB means the data source about Hobart within a range of 50 km; 40 MB means the data source about Hobart within a range of 65 km; 60 MB means the data source about Hobart within a range of 75 km; 80 MB means the data source about Hobart within a range of 82 km. Fig. 6 shows the execution time taken for processing requests with the size of interested area and number of machines utilised. The experiments are repeated 10 times and average values are presented for each scenario. The experiments were conducted on a small size virtual machine having 1 VCPU, 4 GB Ram, and 30 GB disc size. The deadlines are generated between 0 and 10 s using uniform distribution.

---

5. Evaluation

In this section, we will focus on the evaluation of our Cloud service. As the main objective of the algorithm is to meet users’ deadlines and minimise number of machines to process their requests, these are the main metrics that are used for evaluation: (a) Average Waiting Time and (b) Number of Machines utilised indicating the usage cost. The scheduling algorithm utilised by our CFP service is compared with two other usage strategies that are currently used:

- Single Machine: single machine is utilised by the user. It processes the requests based on a First Come First Serve (FCFS) basis and does not consider the deadline.
- Parallel Model: In this case, parallel computing machines are utilised by the user to process the area of interest and requests are served on a FCFS basis. For each request, the minimum number of machines required is computed so that the request can be processed just before the deadline specified by the user.

In the experiments, for the second criteria, i.e. the number of machines used, the proposed algorithm is only compared with the second strategy i.e. parallel computing machines are utilised by the user. To ensure accuracy, the experiments are repeated 10 times and the average time is presented. The capacity of each slave machine is assumed to be the same as used for profiling the execution times presented in the previous section and the results do not present data download times.

5.1. Experimental results

Fig. 7 shows the comparison results of different scheduling strategies against the one proposed. Fig. 7(a) compares the average waiting time of different techniques utilised to process the bushfire prediction model. In Fig. 7(a), we can clearly see that the average waiting time spent on Cloud based service is the smallest, which is about 50% lower than when the user only utilises parallel computing. It is obvious single machine or desktops have very limited processing capacities in comparison to clusters of parallel machines. For this reason in the parallel machines case, the average time is around 4, much better than that on a single machine. However, the reason behind the higher waiting time in the parallel machine case over the Cloud service is much deeper. It is due to the limitation of parallel machines in terms of expandability. Most parallel machines or clusters in different organisations have limited storage and processors which need to be shared between several users. Moreover, the workload of each user is processed on a First Come First Serve (FCFS) basis irrespective of the urgency of their work. Due to this, waiting time is much longer in privately owned clusters than in Cloud based systems. From Fig. 7(a), it can also be observed that the average waiting time is nearly the same in most of the cases. In summary, we can conclude that running requests on a Cloud based service has the best performance, shortening the waiting time for users in comparison with single machine and parallel machines.

Fig. 7(b) compares the number of machines utilised in each scenario. This factor is important to understand the cost effectiveness of the Cloud service based scheduling strategy. For the comparison of number of machines used, we only need to compare the number of machines used on two strategies not with a strategy when a single machine is utilised for each user request. The reason for this is that the result for a single machine strategy will obviously be very low and remain the same.

From Fig. 7(b), we can observe that the number of machines used for the requests of 25 and 75 are nearly equal to the Cloud service; however in cases 50, 100, and 125 requests the Cloud service performs better than the parallel model. The reason for this is the sharing model of the Cloud service based strategy. Users’ requests can be scheduled on the machines where other jobs are running. Thus, resource utilisation is much more compact than parallel machines which in general run the jobs in a more exclusive manner.

From the figure, we can also conclude that if the number of requests from users is increasingly large, the number of machines used on the Cloud service would be lower than the parallel model, which means the Cloud service scheduling would help the server in saving more computing resources when handling the same number of requests.

6. Related work

As discussed earlier, with the emergence of Cloud computing, several researchers are working to solve several geospatial science problems using Cloud environments. In this section, we point out some the most relevant work in this context and compare it with our proposed framework.

Before Cloud computing, many researchers worked on utilising parallel computing technologies to handle computational requirements of visualisation and analysis of large spatial datasets [24–27]. Thus, many research projects focused on developing CyberGIS frameworks [28,29] which integrate GIS with parallel and distributing computing architectures to solve computationally intensive problems. For example, Wang et al. [30] evaluated the performance of GISolve in a distributed environment. Huang et al. [31] proposed the CyberGIS framework that can support multiple data sources. In their work, the Hadoop platform is used to scale the processing of social media data for emergency situations. Yin et al. [32] proposed a model knowledge database to enable utilisation of parallel computing resources for computing GIS models. Chen et al. [33] proposed the efficient evacuation simulator.
using parallel computing principles. Liu et al. [34] proposed GPU based parallel algorithms to improve the efficiency of image processing. [9] proposed a Software as a Service (SaaS) to utilise Cloud computing for a wildfire risk and a wildfire spread simulation service. Bhat et al. [35] proposed a multi-tiered architecture for GIS cloud systems. Srinivas et al. [14] proposed a distributed architecture for building spatial information geoportals based on Cloud computing. In Cui et al. [36], the authors describe a cloud computing model for image processing of remote sensing data. Zhong et al. [37] proposed a geospatial data storage and processing framework for a large-scale WebGIS based Hadoop platform. Miao et al. [38] proposed a Web 2.0-based Science Gateway for Massive Remote Sensing Image Processing using Cluster computing nodes. Huang et al. [39] deployed GEOSS Clearinghouse which is a Metadata Catalog System on an Amazon EC2 Cloud virtual machine. Schnase et al. [40] developed a climate-analytics-as-a-service system (MERRA/AS) using a MapReduce platform. Shao et al. [41] developed a geo-processing service based on Amazon EC2 Cloud.

Morshed et al. [42] recommended environmental knowledge as a linked open data cloud using semantic machine learning. Dutta et al. [43] investigated deep cognitive imaging systems in estimating fire incidence at a continental scale for Australia.

Most of these works do not utilise the autoscaling feature of Clouds. Riteau et al. [15] proposed a Cloud based architecture for CyberGIS analytics with autoscaling features. Wang et al. [44] proposed pipsCloud system to manage data and processing of remote sensing data. Their solutions do not consider the user requirements in terms of deadline and also they do not focus on minimising the number of machines. Yue et al. [16] compared the geospatial data processing in the Microsoft Azure and Google cloud computing environments. They recommend a hybrid Cloud model to get benefits from different Cloud environments.

There has been several work in the area of scheduling and resource allocation [19]. Some of these algorithms also considers quality of service requirements such as time and cost. However, these work either consider very general application model or a specific application. Scheduling algorithms designed for specific applications are not directly applicable to the context of bushfire as each application differ significantly from others. Other scheduling approaches that have been designed for general application models cannot achieve limited amount of performance as they consider application as blackbox without detailing how application should be divided into different tasks.

In summary, our contribution is unique and novel because our proposed framework provides a Cloud based fire prediction service, it takes into consideration users’ time requirements and also utilises the Cloud computing environment in such a way that minimal amount of resources are utilised in addition to leverage the elasticity of the Cloud resources. Our proposed framework also utilises multiple Cloud datacenters to minimise the data download time and also reuses previous processing that further minimises the processing requirements. It allows integration of different fire prediction models which are selected automatically based on users’ requirements.

7. Conclusion and future works

The Cloud computing paradigm has changed the way we utilise computing power for solving data and computationally intensive problems. Thus, due to computational and fluctuating user requirements, geospatial scientists have started to explore scalable frameworks that utilise Cloud computing environments. In this context, fire prediction and behaviour modelling is one of the important areas of research which is gaining a lot of attention due to huge losses of lives and properties that occur during seasonal bushfires. We identified the various technical and user requirements and challenges in designing such a system. We proposed a novel framework for a Cloud based Fire Prediction service that not only leverages the elastic feature of Cloud infrastructure to handle dynamic user requirements in terms of processing needs and time constraints but also minimises resource usage which helps in reducing cost. We also proposed a scheduling algorithm for mapping user requests for fire prediction of a certain region within a certain deadline to Cloud computing resources. The experimental study using the Tasmanian region fire model showed the efficacy of the proposed framework in addition to superiority over previous usage models. The current prototype is applied in the study area of the Tasmania, Australia but its flexibility enables integration of several fire prediction models for different regions.

In future, we plan to do the experiments with a larger setup in terms of number of machines, different fire prediction models and different Cloud environments.
Acknowledgements

We would like to thank Mr. Tuan Do for his assistance in spatial data processing. We would also like to thank Joanne Allison for proofreading the manuscript.

References


Saurabh Garg is currently working as a lecturer in the Department of Computing and Information Systems at the University of Tasmania, Hobart, Tasmania. He was one of the few Ph.D. students who completed in less than three years from the University of Melbourne in 2010. He has published more than 40 papers in highly cited journals and conferences with H-index 25. He received the 2nd place in the Australian National Big Data Challenge in 2016 for devising novel and innovative market-oriented meta-scheduling mechanisms for distributed systems under conditions of concurrent and conflicting resource demand. He has gained about three years of experience in the Industrial Research while working at IBM Research Australia and India.

Jagannath Aryal is currently working as a Senior Lecturer in Surveying and Spatial Sciences with the School of Surveying and Spatial Sciences, University of Tasmania, Hobart, Australia. He received the Ph.D. degree in optimization and systems modelling from Centre for Advanced Computational Solutions (C-ACS), Lincoln University, Lincoln, New Zealand in 2010. He worked for Morgan Stanley and France for his research. His research focuses on advancing the knowledge in Geographic Information (GI) Science and Earth Observation data modelling with an emphasis on spatial and spatio-temporal analysis. Application
areas include terrestrial and extend to marine environments. He is in the editorial board of Journal of Spatial Science of Taylor and Francis Group.

Hao Wang completed his Masters with thesis from University of Tasmania, Australia. He specialized in web development, Java basic programming and Oracle database management and programming, PHP with web development, basic C# language. He has done 5 months internship in the company named Tempus innovative solutions.

Tejal Shah is a Postdoctoral researcher at Newcastle University. She completed her Ph.D. from the School of Computer Science and Engineering at the University of New South Wales, Australia. The focus of her research is on the development and application of Semantic Web Technologies for analyzing Big Data across various disciplines such as healthcare, remote sensing, and smart homes.

Gabor Kecskemeti (Ph.D., University of Westminster, 2011) has been a lecturer in the Department of Computer Science at Liverpool John Moores University, UK since 2016. In the past, he worked as a research fellow at MTA SZTAKI, Hungary, as well as a postdoctoral researcher at University of Innsbruck, Austria. He has been involved in several EU funded projects like: ePerSpace, S-Cube, EDGeS, ENTICE. His research interests include modeling energy efficient and autonomous distributed systems (e.g., clouds and IoT) as well as virtual machine/container image delivery optimization. He has published over 60 scientific papers, and he has also co-edited a few journal special issues and books.

Rajiv Ranjan is an Associate Professor (Reader) in Computing Science at Newcastle University, United Kingdom. Prior to that, he was a Senior Research and Julius Fellow at CSIRO, Canberra, where he was working on projects related to Cloud and big data computing. He has been conducting leading research in the area of Cloud and big data computing developing techniques for: (i) Quality of Service based management and processing of multimedia and big data analytics applications across multiple Cloud data centers (e.g., CSIRO Cloud, Amazon and GoGrid); and (ii) automated decision support for migrating applications to data centers. He has published about 110 papers that include 60+ journal papers. He serves on the editorial board of IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Cloud Computing, and Future Generation Computer System Journals. According to Google Scholar Citations his papers have received about 3450+ citations and he has an h-index of 24.