Bachelor Thesis Project

Using the IBM Watson™ Dialog Service for Assisting Parallel Programming

Author: Adrián Calvo Chozas
Supervisor: Sabri Pllana
Semester: VT/HT 2015
Subject: Computer Science
Abstract

IBM Watson is on the verge of becoming a milestone in computer science as it is using a new technology that relies on cognitive systems. IBM Watson is able to understand questions in natural language and give proper answers. The use of cognitive computing in parallel programming is an open research issue. Therefore, the objective of this project is to investigate how IBM Watson can help in parallel programming by using the Dialog Service. In order to answer our research question an application has been built based on the IBM Watson Dialog Service and a survey has been carried out. The results of our research demonstrate that the developed application offers valuable answers to the questions asked by a programmer and the survey reveals that students would be interested in using it.

Keywords: Parallel Programming, Cognitive Computing, IBM Watson Dialog Service, Natural Language Processing
Preface

I would like to thank Sabri Pllana (LNU, SE) for the opportunity, help and support that he has given to me when I was doing this thesis project. Also, I would like to thank Diego Martín (UCM, ES) for convincing me to come to Sweden as an Erasmus student.
# Table of Contents

1. Introduction ........................................................................................................... 6
   1.1 Background ....................................................................................................... 6
   1.2 Previous research ............................................................................................ 9
   1.3 Problem formulation ....................................................................................... 11
   1.4 Motivation ....................................................................................................... 11
   1.5 Research Question ......................................................................................... 11
   1.6 Scope/Limitation ............................................................................................. 11
   1.7 Target Group ................................................................................................... 11

2. Method .................................................................................................................. 12
   2.1 Scientific Approach ......................................................................................... 12
   2.2 Method Description ......................................................................................... 12
   2.3 Reliability and Validity .................................................................................... 12
   2.4 Ethical Considerations ..................................................................................... 12

3. Implementation ..................................................................................................... 13
   3.1 Dialog Service Overview ............................................................................... 13
   3.2 Dialog Service Implementation ....................................................................... 13
   3.3 Application Overview ..................................................................................... 17

4. Results .................................................................................................................. 19

5. Analysis .................................................................................................................. 22

6. Discussion ............................................................................................................. 23
   6.1 Research Question .......................................................................................... 23

7. Conclusion ............................................................................................................. 24
   7.1 Future Research .............................................................................................. 24

8. References ............................................................................................................ 25

9. Appendix A: Survey ............................................................................................. 28

10. Appendix B: XML Descriptors ........................................................................... 29
List of Tables

Table 1.1 Logical Errors ........................................................................................................ 8
Table 1.2 Performance Errors .................................................................................................. 9
Table 4.1: Questions ................................................................................................................ 19
Table 4.2: Points of each star and total points ......................................................................... 20
Table 5.1: Percentage of each question .................................................................................... 22

List of Figures

Figure 1.1: IBM Watson computing system (credit IBM) .............................................................. 6
Figure 3.1: Settings of the IBM Watson Dialog Service ............................................................... 13
Figure 3.2: Dialog tags ............................................................................................................... 14
Figure 3.3: Folders (Credit IBM) ................................................................................................ 15
Figure 3.4: Welcome message ................................................................................................... 15
Figure 3.5: Input ........................................................................................................................ 15
Figure 3.6: Dollar sign ............................................................................................................... 16
Figure 3.7: Asterisk ..................................................................................................................... 16
Figure 3.8: Output ...................................................................................................................... 16
Figure 3.9: getUserInput ............................................................................................................ 16
Figure 3.10: Default .................................................................................................................. 17
Figure 3.11: Application overview ............................................................................................ 17
Figure 4.1: Survey Result ......................................................................................................... 19
Figure 4.2: Survey Example ...................................................................................................... 20
Figure 9.1: Survey ..................................................................................................................... 28
Figure 9.2: Survey ..................................................................................................................... 28
Figure 10.1: Welcome message of the application .................................................................. 29
Figure 10.2: Library folder to ask questions ............................................................................. 29
Figure 10.3: Questions and Answers in English ..................................................................... 32
Figure 10.4: Questions and Answers in Spanish ....................................................................... 34
Figure 10.5: Mandatory settings ............................................................................................. 35
1 Introduction
A brief introduction about cognitive systems, DeepBlue, DeepQA, IBM Watson, Jeopardy! and OpenMP is presented. Then, previous research is shown to support this thesis and problem formulation and motivation is presented. Finally, the research question, scope/limitation and target group are presented.

1.1 Background
Cognitive systems are on the verge of becoming a milestone since natural language is used when interacting with humans. This new technology allows professionals, in a wide range of areas, to make better decisions that involve Big Data [1]. Besides, cognitive systems are able to learn through education, and can understand Natural Language; thus cognitive systems enable computers to judge and understand questions [2].

Deep Blue is the predecessor of IBM Watson and it was the first machine built by IBM capable of beating a world championship chess player by using complex mathematical calculations. Hence, due to this mathematical capability, the machine was used in order to perform the calculations that were needed in many scientific fields [3] [4]. Following this idea IBM wanted to go further and build a machine capable of understanding questions in natural language and this kind of AI (Artificial Intelligence) has been interesting for researchers for a long time. Due to this, DeepQA team wanted to build a machine capable of using natural language in order to help in fields such as finance, health care and bank. To do that they decided to use QA (Question and Answer). The result of that project was IBM Watson, which is able to process natural language to receive and answer questions.

Watson is a cognitive system that uses natural language to receive and answer questions. It works on unstructured data, which composes 80% of the data available in the Web. Thus IBM Watson learns a new subject by storing all the documents related to the topic on its own database, in which it later searches for the most suitable answer [5].

IBM Watson will help people to make better decisions by looking for valuable information in the web [2]. Also, “Watson is able to help users to find questions that they are not thinking to ask” [Rob High] [2].

Figure 1.1: IBM Watson computing system (credit IBM)

The general timeline evolution of Watson is as follows: in 2011 Watson won at Jeopardy! by using its Dialog Service, which was the first Watson’s Service. Then, in 2012 Watson helped oncologists, in 2013 Watson became commercial. In 2014 Watson
was expanded with new features such as knowledge extraction and graph visualization, Watson explorer and Watson Ecosystem. One year later Bluemix was built and Watson developer cloud was released. In 2016 Watson was expanded by enhancing human engagement focusing on emotional detection and expression. Also, robotics [2].

In 2011, to show how powerful the machine was, IBM decided to compete at Jeopardy. Jeopardy has been the most challenging task that Watson has done, since it is a game in which the players must have a broad knowledge of many fields. Watson was able to win the game beating the best two players in the history of the program.

Watson was fed with literature and QA (Question and Answer which is nowadays called Dialog Service), which was then used to answer the questions that the presenter asked.

In order for Watson to learn, it first needs an expert to include suitable information inside its corpus by adding updated literature. Second, Watson needs to be trained in order to interpret the information given in form of questions and answers. Third, Watson will learn through interaction by users. The experts will take care of the new information which is being given to the corpus by controlling and evaluating the correctness of the information.

When a question is asked, Watson takes the language patterns and creates hypotheses. Then, Watson looks for evidence to support or refuse the question. Finally, Watson estimates a confidence in the answer before giving it [6].

IBM Watson Health was built after helping oncologists in 2012 since the machine was able to give valuable information as to the treatment of patients. According to [7], 80% of the health-care data is invisible, since it can be found either in an unstructured form or is inaccessible. IBM Watson Health is able to read 200 million documents in three seconds, which allows professionals to make better decisions and thus take into account the aforementioned 80% of the data.

For this paper we are going to create a cognitive application related to parallel programming powered by IBM Watson. This application is focused on the retrieval and solving of common mistakes in OpenMP.

OpenMP (Open Multi-Processing) is an Application Program Interface (API) that collects a set of compiler directives, variables and library functions that may be used in shared memory parallelisms. The programming languages that can be used are: C, C++ and Fortran [8] [9].

Major OpenMP contructions are divided into: parallel constructions, work-sharing constructions, SIMD (Single Instruction Multiple Data) constructions, device constructions and tasking constructions [10].

Parallel constructions: Parallel constructions creates a team of threads and initiate parallel execution. It works following this construction: #pragma omp parallel + clause. Clauses can be: num_threads, default, private, shared… [10].

Work-sharing constructions for loops: here each thread executes a part of the program. It works following #pragma omp for + clause. Clauses for a loop can be: private, firstprivate, reduction, schedule, nowait [10].

Work-sharing constructions for sections: structure blocks are executed in parallel. It works following #pragma omp sections + clause. Clauses can be: private, firstprivate, reduction, nowait… [10].

Work-sharing constructions for single: a structured block is executed by one single thread. It works following #pragma omp single + clause. Clauses can be: private, firstprivate, copyprivate and nowait [10].

SIMD (Single Instruction Multiple Data) Constructions: SIMD Constructions specify that a loop can be transformed into a SIMD loop. It works following #pragma omp simd + clause. Clauses can be: safelen, linear, private, reduction… [10].

Device constructions target data: Device constructions target data create the execution on the device targeted. It works following #pragma omp target data + clause. Clauses can be: device, map and if [10].
Device constructions target: Device constructions target create the device target environment. It works following #pragma omp target + clause. Clauses can be: device, map and if [10].

Device constructions target update: Device constructions target update ensure consistency between host and data. It works following #pragma omp target update + clause. Clauses can be: device, to, from, if [10].

Tasking constructions: tasking constructions define a task. It works following #pragma omp task + clause. Clauses can be: if, final, private, mergeable, shared… [10].

Many problems can be found when it comes to program in OpenMP. Those errors can be split up into: Logical Errors and Performance Errors. Main problems are described in tables below [11].

<table>
<thead>
<tr>
<th><strong>Logical errors</strong></th>
<th><strong>Reasons</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing openmp</td>
<td>If OpenMP is not enabled, its directives will be ignored.</td>
</tr>
<tr>
<td>Missing “parallel”</td>
<td>If the programmer forgets to point out parallel, the code will not work as expected.</td>
</tr>
<tr>
<td>Missing “omp”</td>
<td>If omp is forgotten the entire pragma will be ignored.</td>
</tr>
<tr>
<td>Missing “for”</td>
<td>If the programmer wants to divided a loop into n threads and “for” is forgotten, the program will not split up the work into those n threads.</td>
</tr>
<tr>
<td>The usage of ordered is incorrect</td>
<td>If the ordered work is not correctly indicated, the compiler will decide the order randomly.</td>
</tr>
<tr>
<td>Change the number of threads inside a parallel region</td>
<td>Number of threads cannot be changed inside a parallel region since the programmer will have run-time errors.</td>
</tr>
<tr>
<td>Lock a variable without initializing it</td>
<td>According to the specification, to lock a variable it first needs to be initialized.</td>
</tr>
<tr>
<td>Local variables without initializing them</td>
<td>When a thread starts, the variable is copied and any attempt to use it without initializing it is incorrect.</td>
</tr>
<tr>
<td>Parallel array without order</td>
<td>If the result depends on previous iterations, order clause is compulsory in order for it not to have unexpected behaviour.</td>
</tr>
<tr>
<td>Access to a share memory without protection</td>
<td>When several threads are modifying a variable the result is unpredictable.</td>
</tr>
</tbody>
</table>

Table 1.1 Logical Errors

<table>
<thead>
<tr>
<th><strong>Performance errors</strong></th>
<th><strong>Results</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unnecessary flush</td>
<td>If flush directive is used without parameters, it can reduce the performance of the program.</td>
</tr>
<tr>
<td>Using critical instead of atomic</td>
<td>The atomic directive is faster than critical. When atomic cannot be used, the compiler will not allow the programmer to use it.</td>
</tr>
<tr>
<td>Overwork in a critical region</td>
<td>It is known that critical regions reduce the performance of the program so</td>
</tr>
</tbody>
</table>
1.2 Previous research

Goel et al. [12] developed six applications that use different services that Watson provides to understand the functionality and capabilities of Watson. Six groups were created in order to build applications that use one or more of the services that were currently available. The results have been applications in which Watson was trained so as to retrieve answers about the chosen fields. These applications are:

*The Erasmus Project:* The Erasmus Project consists of supporting and enhancing human-computer co-creativity, the project takes the advantage of Watson to reduce the amount of time that designers have to spend gaining knowledge of the issue in order to be able to determine whether a biological concept contains enough insights for a particular problem [12].

Watson represents the core of the architecture by providing natural language and uses AlchemyAPI to retrieve valuable and suitable information for the designer from its corpus in text form after a question in natural language is asked [12].

When the user asks a question it is split up following grammatical function (e.g., subject, verb, object) and then Watson retrieves the information which is being searched for through the AlchemyAPI service [12].

AlchemyAPI: is a set of three services that allows users to build cognitive applications and understand contents and context [15]. The three services are:

- **AlchemyLanguage:** it is a collection of APIs that offers text analysis by using Natural Language Processing.
- **AlchemyData:** it provides news and blog content by using natural language.
- **AlchemyVision:** it understands and analyzes data from visual scenes.

*Watson BioMaterial:* The project is focused on materials in biological designs. It is an Android app that allows a user to look for relevant materials. The user can submit the query in two ways:

- Unstructured query that searches for a material based on a feature.
- Structured query that searches for a material based on one or more features [12].

*Ask Jill:* Jill is an interactive website that supports literature reviews for researchers. They can highlight text in order to ask Watson for papers related to the highlighted text. Watson would return information from relevant papers in its corpus and users would then be able to add them into their investigation [12].

*SustArch:* SustArch is based on the field of biological designs for temperature control, which can be used as a research tool or a community space to buy, sell and discuss designs previously created by users. It is an Android application that allows the user to research this topic using Watson. Watson was trained to return biological systems as results for a given high-level search [12].

*Twenty Questions:* In this application, the user asks a question to the system, the system returns snippets of information from the best five articles in Watson's corpus as answers. The user may select one of the articles and continue playing or stop. If the user continues the system will offer keywords that they can select and use for future searches. The application is based on the 20 Questions game [12].

*Watsabi:* Watsabi is an interactive website dedicated to agricultural issues. The application allows users to ask questions related to agriculture. If Watson fails answering a question the user is allowed to share it with other users to improve the application through a forum [12].
1.2.1 Intelligent Personal Assistant (IPA)

An Intelligent Personal Assistant (IPA) can perform tasks or services for the user. These tasks or services are based on user input such as weather conditions, traffic, schedules and many kind of information available online. IOS Siri, Google Now and Microsoft Cortana are excellent examples of IPAs [14]. Watson is considered to be an intelligent personal assistant since it shares common features with the aforementioned IPAs. The main characteristics of these IPAs are:

**iOS Siri:** Siri is an intelligent assistant that enables Apple device users to speak through natural language [15]. The user can ask a question to Siri and get an answer related to the topic. Siri is able to help the user by activating alarms, writing messages and searching for information. Also, Siri offers entertainment by making jokes and playing music [16].

Nowadays Siri is being integrated into Apple's HomeKit Framework in which the user will be able to turn the lights off, lock the garage and reduce the temperature [17].

**Google now:** Google now is a personal assistant that allows the user to search for information through natural language [18]. Involved features are:
- Knowing the weather in real-time.
- Avoid traffic.
- Retrieving interesting information for the user.

**Microsoft Cortana:** Cortana is a virtual assistant for Microsoft Windows Phone and Windows 10 that allows the user to search data stored in the user's phone and PC. Cortana can manage the calendar, tell jokes and find files in natural language [19] [20].

1.2.2 Application Programming Interface (API)

Nowadays, machine learning is everywhere from smartphones cars. Applications that take advantage of this technology are able to learn through patterns and can be very useful for both companies and customers. IBM Watson is, right now, one of the major competitors in this field. IBM relies on Watson to become the best option in the market. According to [21] ranking, IBM Watson is the second best. Its main current competitors are:

**AT&T:** AT&T is designed to allow developers to create web and mobile applications adding speech recognition capabilities. Just as IBM Watson does, natural language processing is the core of those applications that use AT&T speech API. This API consists of: Speech To Text, Speech To Text Custom and Text to Speech [21].

**Google Prediction:** Google Prediction is known to be one of the most popular machine learning APIs in the market. It includes: NLP (Natural Language Processing), pattern recognition and prediction, spam detection and, like Watson, sentiment analysis [21].

**Wit.ai:** like AT&T, Wit.ai allows developers to build applications for mobile phones and webs that use features such as intelligent voice interfaces [21].

**Diffbot:** Diffbot is a platform that extracts data from web-pages in form of videos, text, comments, images and product information. This platform utilizes AI (Artificial Intelligence), machine learning and NLP (Natural Language Processing) [21].

**Microsoft Azure Machine Learning:** Microsoft Azure Machine Learning is able to process tons of data and build predictive applications. This platform provides: NLP (Natural Language Processing), computer vision, recommendation engine, pattern recognition and predictive modeling [21].

**Amazon Machine Learning:** Amazon Machine Learning offers the possibility of building intelligent applications that rely on pattern recognition and prediction. The developer will be able to build an application with features such as: fraud detection, document classification and customer prediction [21].
1.3 Problem formulation
The goal of this thesis project is to use IBM Watson with its Dialog service as an assistant in parallel programming by building an application that can help a programmer to avoid common mistakes in OpenMP [22]. *The Common Mistakes in OpenMP and How To Avoid Them* paper [22] is used to create questions and answers for this thesis project.

1.4 Motivation
IBM Watson offers a new concept in terms of programming: cognitive systems. As a new technology in ongoing development, many fields can have an advantage of using Watson, fields such as finance or health. It is interesting for me to create an application that can be useful for parallel programmers using this technology.

1.5 Research Question

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>How can we use the IBM Watson™ Dialog service for assisting parallel programming?</td>
</tr>
</tbody>
</table>

This project will result in an application that uses Dialog service and will be useful for novice programmers in parallel computing.

1.6 Scope/Limitation
In this project, we focus on using the IBM Watson Dialog Service in parallel programming. An application capable of retrieving valuable information for the users so that they avoid typical mistakes when using OpenMP is going to be created. The application will be focused on answering questions that an user may ask. However, the application will neither generate code nor find code errors.

1.7 Target Group
Our target group is novice programmers of parallel computing systems [23]. Getting acknowledged with parallel programming is difficult. Having an application that can help a novice parallel programmer by using natural language to reach the answers that they are searching for could potentially improve the results.
2 Method
Scientific Approach, Method Description, Reliability and Validity and Ethical considerations are presented.

2.1 Scientific Approach
A qualitative research methodology has been used in order to answer the question “How can we use the IBM Watson™ Dialog service for assisting parallel programming?” According to [24], “qualitative research questions tend to be open and probative in nature.” Consequently, our research question is exploring an innovative method to assist in parallel programming.

2.2 Method Description
An application has been built, using the Dialog Service, to answer the research question. The platform used has been IBM Bluemix [25] as it is the only currently available platform that supports applications that use IBM Watson.

In order to measure whether the application has been successful a survey has been carried out. The survey is composed by four pre-defined questions, given that the respondents do not need to create an account in IBM Bluemix and they do not ask irrelevant questions to the application.

The respondents are eight students, novice parallel programmers, in the field of computer science. They were selected bearing in mind that to use this application a computer science background is compulsory.

2.3 Reliability and Validity
A survey has been made to evaluate if the application might be useful for students so the external validity is restricted to students and it is neither applicable in companies nor researchers.

The application would work exactly as it is currently working if other researchers use it.

Qualitative research methodology is open and probative in nature and owing to that the research question has been successfully answered as an innovative application related to parallel programming has been successfully built using the Dialog Service. Then, a survey has been carried out to measure if students would be willing to use this kind of application, which, according to the results of the survey, they would.

2.4 Ethical Considerations
The issue of privacy or confidentiality in the survey is addressed by anonymizing the data.
3 Implementation
Dialog service overview, Dialog Service Implementation and Application overview are presented.

3.1 Dialog Service Overview
IBM Watson Dialog service allows the developer to build an application that interacts with the user through natural language offering automated responses related to the issue at hand.

This service has been chosen for our application because it is the best example of natural language. Also, it was used to play Jeopardy with the name: Question and Answer.

3.2 Dialog Service Implementation
In the section the implementation of the application is presented.

3.2.1 Settings
In order to build an application that uses IBM Watson Dialog service the developer needs specify the mandatory settings for the service as it is shown in the figure below. The developer must not change these settings as shown on Figure 3.1: Settings of the IBM Watson Dialog Service below.

```
<settings>
  <setting name="DISPLAYNAME" type="USER">test</setting>
  <setting name="RETURNTOCACHENODEID" type="USER">0</setting>
  <setting name="PERSONALITYTYPEID" type="USER">6</setting>
  <setting name="SENDCHATEMAIL" type="USER">false</setting>
  <setting name="CACHE" type="USER">true</setting>
  <setting name="AUTOLEARN" type="USER">false</setting>
  <setting name="LANGUAGE" type="USER">en</setting>
  <setting name="RESPONSEITEMS" type="USER">2</setting>
  <setting name="MAXAUTOLEARNTITEMS" type="USER">4</setting>
  <setting name="NUMAUTOSTERELATED" type="USER">0</setting>
  <setting name="TIMEZONEID" type="USER">Europe/Prague</setting>
  <setting name="AUTOSTERELATEDNODEID" type="USER">0</setting>
  <setting name="INPUTMASKTYPE" type="USER">0</setting>
  <setting name="CONCEPTMATCHING" type="USER">true</setting>
  <setting name="USECONCEPTS">true</setting>
  <setting name="AL:noneLabel">None of the above</setting>
  <setting name="DYNAMIC_MESSAGING">false</setting>
  <setting name="DEFAULT_DNR_RETURN_POINT_CANDIDATE">1</setting>
  <setting name="ENTITIES_SCOPE">3</setting>
  <setting name="MULTISENT">true</setting>
  <setting name="REPORT_SCRIPT_ID">0</setting>
  <setting name="DNR_NODE_ID">16</setting>
  <setting name="DEFAULT_POINT_NODE_ID">2</setting>
</settings>
```

Figure 3.1: Settings of the IBM Watson Dialog Service

DISPLAYNAME and RETURNTOCACHENODEID are compulsory settings whose value is 0.
PERSONALITYTYPEID is a compulsory setting whose value is 6.
SENDCHATEMAIL is a compulsory setting whose value is false

CACHE is a compulsory setting whose value is true

AUTOLEARN when an input does not match the dialog can suggest another node if AUTOLEARN is activated. The values can be true or false.

LANGUAGE this setting allows the user to determine the account language. It can be:
- en-US English (United States)
- es-ES Spanish (Spain)
- pt-BR Portuguese (Brazil) [26]

MAXAUTOLEARNTITEMS when AUTOLEARN has a true value MAXAUTOLEARNTITEMS will display the items learned with a maximum of 10 and a minimum of 1. The default value is 4.

TIMEZONEID sets the time zone.
\textit{INPUTMASKTYPE} it controls whether to mask user input. The value must not be empty. Values:
\begin{itemize}
  \item 0 which means false. It is the default value
  \item 1 which means true.
\end{itemize}
\textit{CONCEPTMATCHING} sets the dialog concept matching. The value must not be empty. Values:
\begin{itemize}
  \item 0 Matching that uses auto select
  \item 1 Strict matching
  \item 2 Fuzzy matching
\end{itemize}
\textit{USE_CONCEPTS} determine whether to use concepts. Values:
\begin{itemize}
  \item 0. Concepts are not used.
  \item 1. Local concepts are used
  \item 2. Concepts from parent account are used
  \item 3. Concepts from both 1 and 2 are used.
\end{itemize}
\textit{AL_NONE_LABEL} specified the text that must be displayed when auto learn feature is activated.
\textit{DYNAMIC_MESSAGING} controls if dynamic messaging is on. Values:
\begin{itemize}
  \item True
  \item False
\end{itemize}
\textit{DEFAULT_DNR_RETURN_POINT_CANDIDATE} sets the behavior during a check of a DNR return point. Values:
\begin{itemize}
  \item 0. False
  \item 1. True. Each node will be considered as a DNR candidate. Also, it is the default value.
\end{itemize}
\textit{ENTITIES_SCOPE} controls entities. Values:
\begin{itemize}
  \item 0. Entities for local account and system are off.
  \item 1. Local account entities are on
  \item 2. System entities are on
  \item 3. Local account and system entities are on
\end{itemize}
\textit{MULTISENT} controls multiple-sentence matching. Values:
\begin{itemize}
  \item 0. Multiple-matching is off.
  \item 1. Multiple-matching is on
  \item 2. Sub-matching is on
\end{itemize}
\textit{REPORT_SCRIPT_ID} this is a compulsory setting. The value is 0. \textit{DNR_NODE_ID} establishes the type of node to which the DNR should return. Values:
\begin{itemize}
  \item Blank. Default value is -15 if nothing is previously configured. DNR returns to the getUserInput nodes and display text to the closest output node.
  \item -16. DNR returns only to the previous getUserInput.
\end{itemize}
\textit{DEFAULT_POINT_NODE_ID} the ID of the default DNR return point [27].

\textbf{3.2.2 Dialog Tags}
The main characteristic of this service is the dialog tag (Figure 3.2: Dialog tags), without it, the developer cannot use the IBM Watson. In order for the dialog to work the flow tags inside dialog tags are mandatory. Flow tags take care of the content of the dialog and make it work since inside of it the main libraries are included.
3.2.3 Folders

The folders are used to allocate, organize and maintain the information in the Dialog. There are four types of folders (Figure 3.3: Folders (Credit IBM)) [28].

```
<folder label="Main"></folder>
<folder label="Library"></folder>
<folder label="Global"></folder>
<folder label="Concepts"></folder>
```

Figure 3.3: Folders (Credit IBM)

*Global*: when the Dialog is about to perform a function, Global folder is the first in being called.

*Concepts*: this node stores all the concepts in the Dialog

*Main*: this node starts when the application does it. Welcome messages are stored inside in order to welcome the user (Figure 3.4: Welcome message).

```
<item>Hello, welcome to this parallel assistant. Type down your question and I will try to do my best so as to help you.</item>
```

Figure 3.4: Welcome message

*Library*: the entire application is in that node. Library folder contains the core of the Dialog; it stores structures, input, output and stores valuable information.

3.2.4 Input

The input tag allows the user to type down those questions that are going to be formulated. The input node must have a child node which is called grammar (see Figure 3.5: Input) [29].

```
<input>
    <grammar>
        <item>Can I change a variable inside a pragma omp loop?</item>
        <item>$ Change a variable inside a loop? </item>
        <item>change * variable * loop </item>
    </grammar>
</input>
```

Figure 3.5: Input

*Grammar*: here the question is on the verge of being asked. In order to ask a question the developer must add the item node (Figure 3.5: Input).

*Item*: inside this tag the questions are formulated by the user (Figure 3.5: Input).

*Wildcards*: as a natural language one single question can be formulated in several different ways. Wildcards help the developers to control this situation (Figure 3.5: Input). On the one hand, when dollar sign is used the question becomes more open as the user can type down the question as he or she wants but it must be followed by the first verb after the dollar sign. In the example below (Figure 3.6: Dollar sign) one can appreciate that the question is totally open and the next verb is “to change” which means that the question can start as:

- *Is it possible* to change a variable inside a loop?
- *May I* change a variable inside a loop?
On the other hand, when the asterisk is in use means that the place corresponding to the asterisk can be changed for another suitable word (Figure 3.7: Asterisk).

In this particular case the user can ask the question in many ways such as:

- “Can I change the number of threads…”
- “Is it possible to change all of the threads…”

3.2.5 Output
The output tag is being used to answer the question asked by the user (Figure 3.8: Output). Also, the output node is a child of the input node.

```
<output>
    <prompt selectionType="RANDOM">
        <item>It is explicitly forbidden to change the loop variable from inside the loop.</item>
    </prompt>
</output>
```

**selectionType**: selectionType has two values:
- RANDOM. When random is established the output node will randomly display these variations.
- SEQUENTIAL. When an output node contains multiple variations and SEQUENTIAL has been established the output node will display these variations by the chosen order [30].

3.2.6 getUserInput
After the welcome message, getUserInput takes the question that the user asks so as to give a suitable feedback (Figure 3.9: getUserInput) [31].

```
<output>
    <prompt selectionType="RANDOM">
        <item>Hello, welcome to this parallel assistant. Type down your question and I will try to do my best so as to help you.</item>
    </prompt>
</output>
```

**Goto**: Goto node tells the system where to find the answer (Figure 3.9: getUserInput).

**Search**: this statement sends then question asked to the correct folder (Figure 3.9: getUserInput).
3.2.7 Default
Default is used when the system does not understand one question or the user has typed down something that cannot be understood (Figure 3.10: Default) [32].

```
<default>
  <output>
    <prompt selectionType="RANDOM">
      <item>
        I am sorry, I did not understand your question. Please try asking another one.
      </item>
    </prompt>
  </output>
</default>
```

Figure 3.10: Default

3.3 Application Overview
The final application will result as it is shown in Figure 3.11: Application overview.

```
USER
   ↓
Question
   ↓
Dialog service
   ↓
Corpus
   ↓
Answer
```

Figure 3.11: Application overview

User: the user is interested in asking a question related to parallel programming language. That user will presumably be a parallel programmer.

Question: the question is on the brink of being asked. As is usual in natural language, one question can be formulated in several different ways

Dialog Service: among all the available services that IBM Watson™ offers, the Dialog service is in use for this application

Corpus: the corpus composes the core of the application, at that point the question is being processed by IBM Watson™ Dialog Service and it is searching for the most suitable answer to retrieve for the user.

Answer: the question has already been processed. Here, the application has the answer for the user. There are two scenarios:
The question has been understood: if the question has been understood by IBM Watson™ with its Dialog Service a proper answer will be retrieved for the user.

The question has not been understood: if the question has not been understood by IBM Watson™ with its Dialog Service the answer retrieved will be the default message: “I am sorry, I did not understand your question. Please try asking another one”.
4 Results

A survey was carried out to evaluate the usefulness of our application. The number of participants has been eight people.

In table (Table 4.1: Questions) the questions in the survey are shown.

<table>
<thead>
<tr>
<th>Question 1</th>
<th>Do you think an application that can help you in parallel programming by using natural language would be useful, more so than searching engines or paper-based resources?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>Do you think that the answer is accurate?</td>
</tr>
<tr>
<td>Question 3</td>
<td>Do you find it useful that the application is multi-lingual?</td>
</tr>
<tr>
<td>Question 4</td>
<td>Would you find it useful that the application could retrieve papers as an answer?</td>
</tr>
</tbody>
</table>

Table 4.1: Questions

The degree of agreement chosen for this survey has been a Likert scale model in which:

- One star: strongly disagree.
- Two stars: somewhat disagree.
- Three stars: neither agree nor disagree.
- Four stars somewhat agree.
- Five stars: strongly agree.

The figure below (Figure 4.1: Survey Result) shows the survey respondents’ answers:

- x-axis: represents stars
- y-axis represents points collected

![Figure 4.1: Survey Result](image)

We are convinced that the application has been successful as the majority of the respondents have punctuated the application with four stars or more in the majority of the questions asked.

The figure shows the points gotten for each question:
• **Question 1** “Do you think an application that can help you in parallel programming by using natural language would be useful, more so than searching engines or paper-based resources?“: the points collected have been 1 point for 3-star punctuation, 3 points for 4-stars punctuation and 4 points for 5-stars punctuation.

• **Question 2** “Do you think that the answer is accurate?“: the points collected for this question have been 1 point for 2-stars punctuation, 2 points for 3-stars punctuation, 2 point for 4-stars punctuation and 3 points for 5-stars punctuation.

• **Question 3** “Do you find it useful that the application is multi-lingual?“: the points collected for this question have been 1 point for 2-star punctuation, 1 points for 4-stars punctuation and 6 points for 5-stars punctuation.

• **Question 4** “Would you find it useful that the application could retrieve papers as an answer?“: the points collected for this question have been 2 points for 2-stars punctuation, 1 point for 3-stars punctuation, 2 points for 4-stars punctuation and 3 points for 5-stars punctuation.

The table below (Table 4.2: Points of each star and total points) shows the total punctuation for each star and the total amount of points.

<table>
<thead>
<tr>
<th></th>
<th>1-star</th>
<th>2-stars</th>
<th>3-stars</th>
<th>4-stars</th>
<th>5-stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Question 2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Question 3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Question 4</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Percentage</td>
<td>0%</td>
<td>12.5%</td>
<td>12.5%</td>
<td>25%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 4.2: Points of each star and total points

The respondents highly agree with the application as 75% of the total points are pointing out 4 and 5 stars. 12.50% respondents neither agree nor disagree (3 stars) with the application and 12.5% of the respondents disagree with the application pointing out 1 or 2 stars.

Here one of the pre-defined questions in the survey is shown Figure 4.2: Survey Example

**Question-Answer**

Do you think that the answer is accurate?

★★★★★

Can I change a variable inside a loop?

It is explicitly forbidden to change the loop variable from inside the loop.

Type a response Send

Figure 4.2: Survey Example
The title for this question in the survey is “Question-Answer”. The question asked to the participants is whether they think that the answer is accurate. The stars represent the punctuation; from 1-strongly disagree to 5-strongly agree. Finally, a part of the functionality of the application itself is shown in which the user has already asked a question related to a variable inside a loop and the answer that the application retrieves. Also, the space to ask the question in natural language is shown at the bottom of the figure.
5 Analysis

The Table below (Table 5.1: Percentage of each question) shows the percentage of each question related to the respondents' answers.

<table>
<thead>
<tr>
<th>Question</th>
<th>1-star</th>
<th>2-stars</th>
<th>3-stars</th>
<th>4-stars</th>
<th>5-stars</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>0%</td>
<td>0%</td>
<td>12.5%</td>
<td>37.5%</td>
<td>50%</td>
<td>100%</td>
</tr>
<tr>
<td>Question 2</td>
<td>0%</td>
<td>12.5%</td>
<td>25%</td>
<td>25%</td>
<td>37.5%</td>
<td>100%</td>
</tr>
<tr>
<td>Question 3</td>
<td>0%</td>
<td>12.5%</td>
<td>0%</td>
<td>12.5%</td>
<td>75%</td>
<td>100%</td>
</tr>
<tr>
<td>Question 4</td>
<td>0%</td>
<td>25%</td>
<td>12.5%</td>
<td>25%</td>
<td>37.5%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5.1: Percentage of each question

According to the survey, the application would be successful among students as they would like to have an application that can help them in parallel programming by using natural language.

- **Question 1** “Do you think an application that can help you in parallel programming by using natural language would be useful, more so than searching engines or paper-based resources?”: 87.5% of the total score corresponds to 4 stars or more which means that the respondents highly agree with the usefulness of the application.
- **Question 2** “Do you think that the answer is accurate?”: 62.5% of the total score corresponds to 4 stars or more which means that the respondents highly agree with the usefulness of the application.
- **Question 3** “Do you find it useful that the application is multi-lingual?”: 87.5% of the total score corresponds to 4 stars or more which means that the respondents highly agree with the usefulness of the multi-language support in this application.
- **Question 4** “Would you find it useful that the application could retrieve papers as an answer?”: 62.5% of the total score corresponds to 4 stars or more which means that the respondents would highly agree with the usefulness of retrieving papers as answers in this application.
6 Discussion

Cognitive systems, like IBM Watson, are able to learn through user's input and be taught by experts. An application that uses this technology can become a milestone in the field in which it is being used due to the fact that a community can be created to improve applications, share new ideas and discover features that no one has thought of before. The general usefulness of these systems is the advantages they offer such as solutions to common problems. In health care, for instance, doctors can treat their patients better owing to previous experiences with common diseases.

This thesis project has the aim of searching for a way to use IBM Watson in parallel computing in order to tackle common mistakes in OpenMP. The result has been an application that uses the Dialog Service.

Such an application that used natural language could be important for those who want to learn parallel programming, since starting from scratch without help is always difficult.

Therefore, an application that can be taught through usage while an expert evaluates whether the information put in by students is relevant or not can be extremely beneficial for all parties involved in the process. And not only that, but with time the application will be bigger, more useful and more accurate.

6.1 Research Question

This section offers an answer to the Research Question.

- RQ: How can we use the IBM Watson™ Dialog service for assisting parallel programming?

The results show that building an application that uses natural language, retrieve answers and support multi-language based on the IBM Watson Dialog service is feasible to help a novice parallel programmer. Survey results show that a novice parallel user would be more than willing to use the application built and would find it useful to improve his or her knowledge of the subject. Furthermore, it shows that retrieving papers as answers will significantly improve the application.
7 Conclusion

IBM Watson is in ongoing development, from universities to companies it is currently being used. Companies such as Macy's have created a mobile service in which customers can ask questions regarding products, services and facilities in English and Spanish. Universities such as Imperial College of London are using Watson to solve challenging problems. Banks such as Standard Bank are in need of using Watson to solve customers' queries thanks to its high speed. Under Armour, which is a fitness brand, is using Watson to act like a personal trainer though an application. The American Cancer Society is using Watson to provide better answers to patients [33].

For this thesis project, one of the services that Watson can provide has been used following the main idea of these aforementioned projects. The project shows an emerging cognitive computing technology adapted to parallel programming to develop an application for assisting novice parallel programmers.

The application can be an educational resource with which students could develop applications avoiding common errors in OpenMP. Since natural language learns from the user’s collected input, the application can be improved as it is used.

The application might have been improved if more users had tested the application since better results could appear related to their own ideas, opinions and help in order to get more accurate answers and more parallel languages. Despite that, the results of the survey have shown that an application that relies on Dialog service shall be useful for novice programmers.

7.1 Future Research

The current application can be improved by adding more services, connecting it to a database and feeding it information to retrieve more and better answers.
8 References


[10] Sabri Plllana. 2DV605 OpenMP. Parallel Computing course. Linneuniversitetet


9 Appendix A: Survey

1* Do you think an application that can help you in parallel programming by using natural language would be useful, more so than searching engines or paper-based resources?

Welcome message
Hello, welcome to this parallel assistant. Type down your question and I will try to do my best so as to help you.

★★★★★

2* Do you think that the answer is accurate?

Question
Can I change a variable inside a loop?

It is explicitly forbidden to change the loop variable from inside the loop.

Type a response Send

★★★★★

Figure 9.1: Survey

3* Do you find it useful that the application is multi-lingual?

Spanish
¿Puedo cambiar una variable dentro de un bucle?

Está totalmente prohibido cambiar una variable dentro de un bucle

Type a response Send

★★★★★

4* Would you find it useful that the application could retrieve papers as an answer?

Papers as answers
★★★★★

Figure 9.2: Survey
10 Appendix B: XML Descriptors

Welcome message:

```xml
<xml version="1.0" encoding="UTF-8">
  <folder label="main">
    <output>
      <prompt selectionType="RANDOM">
        <item>hello, welcome to this parallel assistant. Type down your question and I will try to do my best to help you.</item>
      </prompt>
      <prompt>
        <folder id="folder_2449011"/>
        <default/>
        <prompt>
          <folder id="folder_2449011"/>
          <default/>
        </prompt>
      </output>
    </folder>
  </folder>
</xml>
```

Figure 10.1: Welcome message of the application

Library folder to ask questions:

```xml
<folder label="Library">
  <folder label="Live Content.id="folder_2449011">
  </folder>
</folder>
```

Figure 10.2: Library folder to ask questions

Corpus: Questions and Answers

```xml
<grammar>
  <item>Can I change a variable inside a pragma omp loop?</item>
  <item>Change a variable inside a loop?</item>
  <item>change * variable * loop</item>
</grammar>

<output>
  <prompt selectionType="RANDOM">
    <item>It is explicitly forbidden to change the loop variable from inside the loop.</item>
  </prompt>
</output>

<grammar>
  <item>when should I use ordered clause?</item>
  <item>ordered clause</item>
</grammar>

<prompt selectionType="RANDOM">
  <item>You cannot put an ordered clause into a for worksharing contract without specifying a separate ordered clause inside the enclosed for loop since this is supposed to be carried out in order.</item>
</prompt>
```

Figure 10.3: Corpus: Questions and Answers
Can use locks without flush?

- Locks without flush:
- Locks are flushed:

There are special cases when synchronisation can be achieved with a simple atomic construct. Not using it in this case leads to potentially slower programs and is therefore a performance mistake.

Use of orphaned construct outside parallel region:

- Orphaned construct outside parallel region:
- Orphaned construct:

- When using a combined worksharing construct, students forget to put down the parallel, producing an orphaned construct.

- When should flush be used?
- Flush be used:
- Flush used:

-Flush explicitly included in certain positions of the code by the compiler. Explicitly specifying a flush immediately before of after these positions is considered a performance mistake.

Mistakes when critical is used:

- Critical is used:
- Critical used:

The mistake is to protect memory accesses with a critical construct, although they need no protection.
<input>
  <grammar>
    <item>Access to shared variables not protected</item>
    <item>$ access to shared variables not protected</item>
  </grammar>
  <output>
    <prompt selectionType="RANDOM">
      <item>This is a severe mistake. You should avoid trying to access the same memory location.
      OpenMP provides several contracts for protecting critical regions, such as the critical construct, the atomic construct and locks</item>
    </prompt>
  </output>
</input>

<input>
  <grammar>
    <item>Could you recommend a book about memory model</item>
    <item>$ recommend a book about memory model</item>
  </grammar>
  <output>
    <prompt selectionType="RANDOM">
      <item>The OpenMP memory model. Hoeflinger, J.P</item>
    </prompt>
  </output>
</input>

<input>
  <grammar>
    <item>Private variables</item>
    <item>$ private variables</item>
  </grammar>
  <output>
    <prompt selectionType="RANDOM">
      <item>C and C++ both allow variables to be declared inside a parallel region. These variables will be private (except in static variables) and it is therefore not necessary to explicitly mark them as such. Also, default clause should be used since it will force each variable to be explicitly declared in a data-sharing attribute clause, or else the compiler will complain</item>
    </prompt>
  </output>
</input>

<input>
  <grammar>
    <item>Problems with omp_unset_lock()</item>
    <item>$ omp_unset_lock()</item>
  </grammar>
  <output>
    <prompt selectionType="RANDOM">
      <item>The OpenMP specification clearly states: "The thread which sets the lock is then said to own the lock. A thread which owns a lock may unset that lock, returning it to the unlocked state. A thread may not set or unset a lock which is owned by another thread. So my recommendation is to use locks when absolutely necessary."</item>
    </prompt>
  </output>
</input>

<input>
  <grammar>
    <item>Why my loop is not working</item>
    <item>$ for loop is not working</item>
  </grammar>
  <output>
    <prompt selectionType="RANDOM">
      <item>Sometimes, a parallel programmer forgets to include the word "for" inside #pragma omp parallel for and, because of that, the for loop will have an unexpected behaviour</item>
    </prompt>
  </output>
</input>
Figure 10.3: Questions and Answers in English
Figure 10.4: Questions and Answers in Spanish
Mandatory settings:

```xml
<settings>
  <setting name="DISPLAYNAME" type="USER">test</setting>
  <setting name="RETURNTOCHATNODEID" type="USER">0</setting>
  <setting name="PERSONALITYTYPEID" type="USER">6</setting>
  <setting name="SENDCHATEMAIL" type="USER">false</setting>
  <setting name="CACHE" type="USER">true</setting>
  <setting name="AUTOLEARN" type="USER">false</setting>
  <setting name="LANGUAGE" type="USER">EN</setting>
  <setting name="RESPONSETIME" type="USER">-2</setting>
  <setting name="MAXAUTOLEARNITEMS" type="USER">4</setting>
  <setting name="NUMAUTOSETRELATED" type="USER">0</setting>
  <setting name="TIMEZONEID" type="USER">Europe/Prague</setting>
  <setting name="AUTOSETRELATEDNODEID" type="USER">0</setting>
  <setting name="INPUTMASKTYPE" type="USER">0</setting>
  <setting name="CONCEPTMATCHING" type="USER">0</setting>
  <setting name="USE_CONCEPTS">3</setting>
  <setting name="AL_NONE_LABEL">None of the above</setting>
  <setting name="DYNAMIC_MESSAGING">false</setting>
  <setting name="DEFAULT_DNR_RETURN_POINT_CANDIDATE">1</setting>
  <setting name="ENTITIES_SCOPE">3</setting>
  <setting name="MULTISENT">0</setting>
  <setting name="REPORT_SCRIPT_ID">0</setting>
  <setting name="DNR_NODE_ID">16</setting>
  <setting name="DEFAULT_POINT_NODE_ID">2</setting>
</settings>
```

Figure 10.5: Mandatory settings