Efficiency of Searching Knowledge Bases for Robotic Control

Maurice Eggen  
Department of Computer Science  
Trinity University  
San Antonio, TX 78212  
meggen@cs.trinity.edu

Roger Eggen  
College of Computing and Information Science  
University of North Florida  
Jacksonville, FL 32216  
ree@unf.edu

Abstract: Autonomous intelligent robotic systems that can survive in dynamic changing environments have yet to be realized. The robot must process incoming information, make decisions, and learn from previous actions concurrently and in real time. Reliance on one all-encompassing system with one overburdened processor, no matter how fast, makes this goal unreachable. One of the primary reasons the goal is unreachable is that the knowledge base that the robot must access to make its decisions grows so large over time that simply searching the knowledge base for a decision becomes too time consuming for acceptable robotic behavior. One solution is to distribute the knowledge base among several processors and search in parallel. An experiment was contrived to test parallel searching algorithms which must search large databases of information, and provide some insight on performance of the underlying programming environment.

Keywords: Intelligent agents, robotic control, distributed programming

Introduction

Autonomous intelligent robotic systems that can survive in dynamic changing environments have yet to be realized. The robot must process incoming information, make decisions, and learn from previous actions concurrently and in real time. Reliance on one all-encompassing system with one overburdened processor, no matter how fast, makes this goal unreachable. One of the primary reasons the goal is unreachable is that the knowledge base that the robot must access to make its decisions grows so large over time that simply searching the knowledge base for a decision becomes too time consuming for acceptable robotic behavior. One solution is to distribute the knowledge base among several processors and search in parallel. An experiment was contrived to test parallel searching algorithms which must search large databases of information. The databases might be thought of as the knowledge bases that must be searched in order to provide the intelligent agent the information to be able to move about in
and adapt to a changing environment autonomously.

Research goals

Our research focuses on finding a system that is able to learn and act autonomously as well as function in a real time. Several questions arise as one attempts a general solution to such a problem. On a recent episode of “Star Trek: Enterprise” the Romulan empire was experimenting with a battle drone which was unmanned, but controlled by a telepath from a remote location. In 1996 the US Air Force announced that they were entering a new era of unmanned flight with the development of the Predator. The Predator classification of unmanned air vehicle suggests that it is capable of completing its mission without the need for a pilot. However, the Predator does require a pilot—an individual who operates the vehicle from the safety of a remote command center. Recently NASA sent Spirit and Opportunity rovers to Mars to explore. While exciting and successful, all of the above suffer from one major flaw. The robotic agents do not operate autonomously. All must be controlled via human intervention. Whether fantasy or reality, the goal is to eliminate the need for interaction with a human controller or pilot. When Spirit wishes to move across the surface of Mars, it must wait for a directive from Earth to do so. Envision a situation where a boulder is dislodged from a precipice and threatens the safety of Spirit. Will the human controller have time to react to the situation or will Spirit be damaged? Considering the time it takes for transmission of a signal from Earth to Mars, Spirit would most likely be lost, or at least incapacitated, reduced in its ability to function. The goal is to have Spirit observe its environment and take evasive action in real time, thus avoiding the need for a directive from a remote location to do so.

Background

The following definitions are accepted by the research community studying the design and implementation of agents, intelligent agents and distributed intelligent agents:

- An agent is an autonomous computational unit that observes its environment through the use of sensors, compares that data to a set of predefined rules, and interacts with its environment as defined by those rules.
- An agent is intelligent if it can modify its set of rules during operation in such a way as to optimize its interactions with the environment.
- A distributed intelligent agent is an intelligent agent that is composed of multiple agents with incomplete information about their environment. In distributed intelligent agents the system control is distributed, data is decentralized, and computation and communication among the agents is asynchronous.

Why should an agent be intelligent? No programmer, no matter how good, can anticipate all of the possible situations that an agent could encounter when operating in a foreign environment. An agent must be intelligent, that is, it must be able to modify its rules, its decision
making capabilities in real time in order to interact with its environment.

Assuming the robot survives for a period of time in the foreign environment, modifies its knowledge base of information and is successful at staying alive, it is clear that the knowledge base will continue to grow the longer the robot survives. A human infant has little knowledge of the environment it so suddenly enters at birth. The longer the human infant survives, the more knowledge it acquires. Soon fundamental abilities to eat and cry are supplanted by language abilities and intelligent and creative play. The infant’s knowledge base grows at a staggering rate during the first years of its life.

So it must be for an autonomous intelligent agent if it is to survive in its potentially hostile environment. The problem this research addresses is a simple yet extremely important one. Namely, that of searching the potentially massive knowledge base for an appropriate next move. The search must be able to be accomplished in real time as well, since if the robot must make decisions, it can not wait for extended processing to take place.

The experiment

An experiment was contrived to test searching algorithms where large databases of information were involved. A large database of random 10 letter words was created. The database varied in size to test the efficiency of various algorithms for performing the search. A random 10 letter test key was also generated, and the database was searched for all matches of the key. Naturally, because of the size of the database, the database was distributed among the workstations in the distributed cluster. Various programming language environments were also used, primarily C-MPI versus Java. The algorithms were implemented and times were taken, providing some insight on programming interfaces and efficiency when robotic searching of large knowledge bases is undertaken.

The equipment: hardware

The hardware for the experiment consisted of a cluster of Intel(R) Pentium(R) 4 machines with CPU at 2.4 GHz. The machines were connected with gigabit fast ethernet. Each of the machines had a 512Kb cache with 256Mb of ram.

The equipment: software

The machines in the experimental cluster were all Linux based, utilizing Fedora Core 2 with kernel version 2.6.7-1.494.2.2. The java version used was j2sdk1.4.2_04. The MPI version used was LAM 7.0.3/MPI 2. The experimental environment consisted of a cluster of as many as 40 workstations all configured exactly the same. However, the cluster is not a dedicated cluster. The timing experiments were done at 3:30am to minimize the effect of the multi-user environment. The experiment was performed five times at each data point and an average is the value recorded.

The algorithm

The assumptions of the experiment are such that the robot’s knowledge base has exceeded the size that can be held in memory. Whether or not that is actually the case for certain configurations
recorded is not relevant. The researchers wished to compare the performance of various sizes of databases with various numbers of distributed processors and test the efficiency and effectiveness of two modern and popular programming environments.

In addition the distribution of the knowledge base among the processors was timed, since the robot’s knowledge base was assumed to be managed in a distributed manner. The assumption is that, while one of the distributed agents may have partial information concerning its course of action, the desired reaction to the environment may not explicitly reside in the portion of the knowledge base it possesses. Thus, it must communicate with the other agents as they search in order to determine the optimal course of action. A person listens, smells, sees and feels its environment in order to interact with it. Moreover, all of these human actions are performed in parallel. So it must be for the distributed intelligent agent.

The size of the distributed databases and the ability of the programming environment to perform the searches is key, since the outcome depends on real-time behavior.

A master process acts as host for the search. It determines the number of workers available and performs the initial distribution of the knowledge base among the workers. It also transmits the key to each of the workers. The workers search their portion of the knowledge base, and when a match is found, it transmits a pointer to that location back to the master process. The master process resolves the pointer to one it understands and records it. For example, if process 1 finds a match at location 2 in a database if size 10, the master process must record a hit at location 12 in the original database. The master process knows the workers have finished when it receives a null pointer from each of the workers, indicating that all distributed searches have been completed. If there is additional searching to be done, and if one of the workers finishes first, then that worker is given more work to be done. In this way, all of the workers can stay busy, independent of the progress of the others. Moreover, if some of the processors in the process group are faster than others, they can request additional items to search.

**MPI**

MPI, the message passing interface, has become the standard for message passing distributed computing. MPI is designed for both high-performance computing as well as on distributed clusters. MPI is widely available, and free versions exist. MPI evolved over a period of time, developed by researchers as well as users. MPI exists as a collection of library function calls with C, C++, and Fortran bindings. The real attractiveness of MPI is that one can perform significant experiments in parallel computing with a small subset of commands. MPI uses a single program multiple data (SPMD) approach to parallelism. Using this approach, master-slave models as well as work pool models of parallelism may be implemented.

The simplicity of MPI is illustrated by the fact that only six fundamental commands may allow the user to perform parallel experiments.
MPI_Init();
MPI_Comm_rank();
MPI_Comm_size();
MPI_Send();
MPI_Recv();
MPI_Finalize();

MPI_Init() initializes the environment. All message passing commands follow this function call. MPI_Comm_rank() tells who you are in the process group. MPI_Comm_size() tells the number of cooperating processes in the process group. MPI_Finalize() gracefully terminates the MPI program. Numerous tutorials and implementations are available on the web.

**JAVA**

The Java programming language has enjoyed explosive growth in recent years. Java has evolved into a general purpose programming language with many features supporting a variety of application areas. In particular, Java makes socket programming straightforward, and Java’s Remote Method Invocation (RMI) allows the invocation of methods on remote machines, thus supporting a distributed programming environment.

**Results**

We therefore find ourselves with a dilemma. We have two very viable and popular programming environments in which to implement the application and control of our distributed intelligent agents. With the requirement of real-time performance constantly in mind, native performance of our programming environment is an issue. The study performed in this paper answers some of these important questions. When searching the knowledge-base, which programming environment has the best performance?

(To the reviewer: We are still gathering data for this section. Hard and fast results will be available for the final copy).

**Summary and Conclusions**

We have tested various file sizes with our distributed searching algorithms, up to several gigabytes in size. Even though the machines used for the testing were not dedicated machines, tests were run when usage of the linux machines was lightest, most often in the middle of the night.

Preliminary results show that if speed of processing is the issue, then the C-MPI environment is superior to the Java environment. Apparently there is less overhead with using the C programming language. Others have reported similar results.

**Directions for future work**

Accessing the knowledge base is only one aspect of robotic control. Database design, inference engines, learning engines and their design and implementation are all problems which will need to be solved to realize our goal.

If speed is the overall goal then C-MPI is probably the environment of choice. However, Java with its highly developed API has programming advantages to great to overlook. If robust code with ease of maintainability is the goal Java has its own advantages.
Bibliography


