AN ARTIFICIAL INTELLIGENCE TECHNIQUES BASED ON DISTRIBUTED OPERATING SYSTEM

Manisha Verma Assistant Professor**,** Department of Computer Science & Engineering, Rama University, Kanpur, Uttar Pradesh, India
Dr. Ravindra Nath Associate Professor, University Institute of Engineering and Technology, C S J M University Kanpur Uttar Pradesh, India

Abstract—
This paper presents and assesses a few artificial intelligence techniques, various computing technology, Quantum or Digital Gates computing are made microchip mechanism and worked on identical principles with different computing produces that practically and theoretically improved or facilitated some major elements of a distributed operating system, that is feasible for Quantum and Digital gates computing technologies with less execution and ability.
The distributed operating system elements impacted were mainly resource management, fault-tolerance, user-interface, security, resource discovery and naming. Featuring on Artificial Intelligence acquirement skill limitations dependency on hardware/software devices and technologies interaction. Some of the artificial intelligence techniques evaluated include Genetic algorithms, simulated annealing, A* search, neural networks, reinforcement learning and Surprise-Based Learning. The paper concludes that a merger between artificial intelligence and distributed operating systems will indeed yield more autonomous and powerful frameworks.

Keywords—Artificial Intelligent Distributed Operating System (AIDOS), Artificial Intelligence (AI), Intelligent Operating System(IOS), Distributed Operating Systems, Artificial Intelligence.

INTRODUCTION

Advances in distributed operating systems have positively influenced the field of artificial intelligence (AI).. The availability of extra resources such as multiple processors and larger memory have improved the turnaround time of AI applications, while the abstractions in communication and file storage have reduced the design overheads significantly. The potential to vastly improve AI application performance by using distributed systems fueled an entire field of research called "Distributed Artificial Intelligence" [1]", which has now largely been super ceded by multi-agent systems [2]. These AI subfields focused on creating algorithms and consequently applications that could run in parallel to achieve a common goal. Distributed operating systems have made the development and deployment of such applications much simpler.
Distributed Natural Language Summarization and Parallel Search using Machine Learning are a few such applications that have benefited from the availability of distributed operating systems. This provides the required information at the right time to aid decision making in a particular domain. It consists of mobile intelligent agents designed for information retrieval, extraction, assimilation and knowledge discovery using AI techniques such as neural networks, Bayesian networks and evolutionary techniques.
These agents are controlled and coordinated by the distributed operating system. Similarly, in DNLS distributed agents provide summaries of information found from different sources for a specific topic. In PSML the search problem is broken into smaller parts using either Parallel Window Search or Distributed Tree Search and executed in parallel on several nodes under the control of the distributed operating system (OS).
Distributed operating systems have influenced AI, several researchers have tried to use AI techniques to boost or facilitate distributed operating systems. the primary contribution of this paper was work the success of such research. In particular, the impact on several major elements of a distributed OS equivalent to resource management (processor programming), fault-tolerance Associate in Nursingd user-interface was discovered. The investigation commenced from the well

summarized literature on omnipresent computing, that highlighted the importance of autonomy in distributed systems. Then, the following practical experiences were considered. Nikravanand Kashani [7] explored the feasibility of process scheduling with genetic algorithms, while Lin and Hsu [8] used simulated annealing for task scheduling and Shen and Tsai [9] used A* search. In contrast, Xie et al. [11] focused on developing an AI based operating system to manage the distributed resources while providing a friendly user interface.

The final contribution of this paper was the evaluation of the theoretical applicability of several AI techniques to either improve or facilitate various aspects of distributed operating systems. The potential benefits of each approach towards resource (processor, file and memory) management, fault- tolerance, security, resource discovery and naming were reviewed. In particular, the most novel analysis was the feasibility evaluation of applying a new learning paradigm known as Surprise-Based learning to autonomously construct models of the distributed system with which the overall system performance can be adjusted dynamically.

A computing technology should perform basic arithmetic operations and describe on\off or up\down states. Nature of computing technology works on the different principle of mathematics which simulates Computing in Measurement Level, Binary System and applied coding scheme on them to get maximum result from the optimal processing. The whole development of new computing technology must have the capability to solve existing problems already solved by computing technology. By passage of time, advancement in computing took place, and computing technology changed its shape, size, material, stability properties and mechanism for characteristics of faster computation ability and Performance.

Different types of computing technology changed with advancement in ages:

- Computing with Mechanical
- Computing with Water
- Computing with Machines
- Computing with Digital Gates
- Computing with Quantum
- Computing with Organic

Abacus was the primary mechanical process device. Mechanical computing devices were solely performance restricted and only addiction price calculation once a lot of improvement in mechanical machines this device capable of the performance subtraction, multiplication, division more advancement the operational operate includes root of variety and quadratic equation. once probably progress gained in mechanical computing. In 1936 somebody Vladimir Lukyanov introduced water computing technology. it's world' first analog stream pc and one in every of its kind, ready to solve partial differential equations. The journeyman resolved the equations by "playing around" with a series of interconnected tubes crammed with water. Water computing technology wasn't obtaining famed as a result of its limitations and capability. Somebody refocuses on mechanical computing technology and starts developing Hybrid Computing Technology of Mechanical and Machine Computing prototypes. In hybrid computing prototype, automaton driven by machine motor. Advancement in machine computing, mechanical computing ultimately born-again to computers devices that increased the performance, stability, and capability. In 1942, John Vincent Atanasoff and his assistant Clifford E. Berry unreal Atanas off-Berry pc which based mostly on digital gates Computing technology and its next steps incremented in Computing technology. Digital Computing is that the most booming computing technology ever as a result of It will implement with all type’s advancement electrical semiconductor technology. Improvement in Traditionally, All Computing Technology Non-Biological and lack of correctives Self-configuration, Self-optimization, Self-protection, Self-explanation, Self-healing, and Self-describing. These capabilities solely found in Organic living thing. Duke University laboratory designed initial nuclear computer but it not entirely in proper shape, however somebody confirmed that organic computing would replace quantum computing because of human bio-capabilities.

Latest Computing Technologies are Quantum and Organic. Both technologies are touching nanoscale, faster computing. As compared to Quantum computing, Organic computing is quicker and present in living Organism and more form of molecular .Current Digital Gate Computing technology solves a haul in sixteen repetitions whereas Quantum Computing has property to unravel an equivalent problem in one cycle and Organic Computing additionally resolve that question within the amount and more difficult higher aspects matching. Organic computing systems comprise of freelance and collaborating subsystems, organization is predicated on adaptational and context-sensitive behaviour.

Biological laptop computing systems have self-properties as:-

• Self-configuration
• Self-optimization
• self-defense
• Self-explanation
• Self-healing
• Self-describing

Organic Computing is that the best mate with AIDOS.AI solely makes selections; however its decisions depend on analysis, definition, identity, info, and knowledge that entirely rely upon Computing Technology and peripheral devices. Quantum computing still employing a positional notation wherever Organic Computing has additional Decision-making power by judgement atmosphere and scenario on binary. For Example, Bio-Devices may be more correct than Electronic Devices. Bio-Devices sensing more in less process and more information send to a ADPS for system software system whereas Quantum Production is simply too big-ticket and solely specific purpose worked. Quantum computing production desires the extremely developed environment and plenty of ample resources. Whereas Organic Production is sleek and it's not necessary of teeming resources. Quantum Computing Production is extremely pricey as compare to Organic Computing. Quantum computing programming languages are Java, Python, C++ give libraries, plug-in, syntax editor. Quantum Computing desires a show interaction for sharing info whereas organic devices can patch with the bod or transmit generated an indication that communicates with Human Mind directly. The nuclear {computing device|computer|computing machine data processor electronic computer information process system machine} will produce by completely different technologies. person discovered it might be develop for any material which contains molecular structure which has to evolve within the correct mechanism. every computing technology has its material properties, operating mechanism, processing power that management by algorithms direct the computing flow control to achieve necessary results. Algorithms ordered steps build a needed outcome that we all know as software system. The software has to be in physical or logical form before golf shot it within the process mechanism which synchronizes and regulates the flow of computing in the right direction.

Human AI Convert to Machine AI (AI): changing human level kinds of intelligence in algorithmic shape to regulate in computing technologies. therefore Computing Learn and acquire the expertise by playacting a task each cycle till it absolutely understands and done the duty in whole produce. Act because the human call within the crucial scenario and do the job with additional accuracy by doing obtaining task experience. Albert Einstein one in all the famed person in the history. He is that the creator of the equation E = mc2 nearly same mechanism of this equation applying on AI by Dr. Alexander D. Wissner Gross. He discovers an equation that is learning $F = T \nabla S\tau$ automatically. He created an AI algorithmic rule on following rules of $F = T \nabla S\tau$ which applies to completely different Fields of AI Applications. Definition of Intelligence

• Ability to Analyse, creating the definition of an object by properties, applying Sequence of expertises that are learned by object properties to grasp object characteristics and dealing Functions.
• A collected information of definitions which is employed to outline an object and its specific functions are known as intelligence

• The experience gaining by receiving knowledge from completely different essential components of sense by size, shape, logically, physical, waves, touch, taste, smell to operational acts on objects respectively, mastering to understanding its working properties and limitations.
Techniques for Artificial Intelligence created on logic base analysis, and this philosophy established some principle that is pre-defined sort of a living thing. By doing work, the coordination increased between dominant mechanisms (and therefore the and additionally the specific organs)it also will increase intelligence logic on the bottom of configuration by the record within the memory. each living species on earth have a unique characteristic supported the biological structure and have varied kinds of intelligence as their species type. People at large contain 9 types of intelligence
• Naturalist
• Musical
• Logically
• Existential
• social
• Linguistic
• Intra-person
• spatial
The scientists have found that each species contains different intelligence and characteristics, however the human is that the most intelligent and characterised living species on earth. As delineated earlier that assortment of logics in specific order makes intelligence. From the start once man began to understand the facility of mathematics, a person's would understand math supported logics if it applied with authentication. AI is predicated entirely on math and might work with all sorts of sciences. whereas operating in several professions to reinforce ability and outcomes, innovation of mathematical algorithms become a reality. AN algorithmic rule may be a sequence of distinct steps that takes some input and apply the answer (which outlined in measures to manage the input parameters) on the input provides output. within the algorithm, mathematical equations enclosed that leads USA to needed result. algorithmic rule describe the logic and combination of logics makes intelligence. The brain is that the most subtle organ in the plant.

Artificial Software system

Main 2 kinds of software,i.e., System software system and Application software system. Latest Application software system is AI (Artificial Intelligence) Software. AN AI Application Software is nothing over software which is aware of some learning algorithms. It will some already programmed work known as Task, once it's applying the instruction on specific work scheduled (is called Performance), the result of worked task knowledge for { higher stronger a additional robust an improved} supervisor (called experienced).AI program updates Itself endlessly on every excellent detection occurs. AI Application software got a massive quality in each profession like Education, Business, Medical, and significant operating Intelligent Machines, and lots of others fields. Currently the planet is wanting forward to an unnaturally intelligent package (AIOS), A System software system As compared to Artificial Intelligent Application software system development is more challenging. Our proposed AIOS design is going to be initial package supported organic computing. it'll be capable of self-learning. No operating system architecture of this nature nonetheless exists.
AI Techniques can be implemented on this artificial intelligence distributed OS design will be as complex as the human brain. AI in computing is a collection of following units.

Artificial Intelligence (AI)				
Machine Learning	Expert Systems	Speech	Natural Language Processing (NLP)	Planning, scheduling & optimization

Robotics	AI Network	Guided Systems	Vision	AI Real-time embedded
Healthcare	Searching	Engineering	Automobile	Finance
Scientific				

Machine learning
- Deep learning
- Predictive analytics

Expert System
- Missile System
- Radar System
- Drone System
- Custom Build System on base Linux kernel
- Home and Office Appliance Systems
-

Natural Language Processing (NLP)
- Translation
- Classification & Cluster
- Data Extraction

Robotics
- Algorithms for Functions
- Management and Perform Tasks
- Sensors
- Motions

Speech
- Speech to Text
- Text to Speech

Planning, Scheduling& Optimization
- Data Analysis Approaches
- Achieve new level of efficiency
- Best input associated with higher outcome
- Decision on Timelines

Using Artificial Intelligence techniques in distributed operating system loads as cache memory but no need to force the brain for recalling. It additionally checks, sense, error, detection report, diagnosis, repairing, limitation, practical and performance of physique attachments. Similarly, once AIDOS starts up, it remembers and loads all parts of AIDOS Systems from intelligent Start-up. AIDOS is an intelligent package that supports even AIDOS in depth Endless analysis and better achievements in future. Hardware specification for such operational systems isn't needed, however its properties and have matter. These operating systems get input by interacting with connected devices and move operational space when corroboratory and validating the input analysis of Data type. practical area decides that section or combination of sections of Sciences dealt the input and applied various algorithmic rule sections. Outcomes of the input are once more compare and validate. Outcome generated terribly showing intelligence to store, and AIOS update its Operation's results and send update core files to an information centre. A team of pros check the core files and after their improved update core files enclosed in next upgraded version/early access version/developer version/testing version or updates. the total heroic tale of AIDOS becomes higher when every cycle, and new installation will cowl info change in few minutes. The user of this

AIDOS is secured and protected, and there's a mechanism style for the user that user will access Devices and his sensitive knowledge like on-line banking, ID accounts, Social, another website that supports AIOS API. This API design works with system hardware and user input code. Unauthorized misuse is going to be not possible from AIOS. there'll be no hacking possible like access to system Camera, Mic, identity verification or recorded activities. The End-User of AIOS gains additional knowledge, Increment in productive. AIOS can run on the transportable computer as Arduino, Intel gello, Raspberry Pi, and Banana. The framework starts its service to complete the outlined module in an operational state, so AISP intelligent Start-up (Kernel) alter operating load with best performance on the low demand of hardware. Framework load base part and good load another connected hardware components. INTELLIGENCE in an exceedingly DISTRIBUTED OS within the late 80s Mark Weiser [13] pictured a future full of omnipresent computing. He expressed that the physical world would be instrumented with pervasive networks of sensor-rich, embedded computation, within which devices were alert to their relationship with users moreover as different devices within the environment. Estrin et al. [6] explained that omnipresence may well be earned by injecting computation into the physical world, which physical property should be achieved by having nodes and their collectives operate autonomously. They argued that the degree of autonomy or the extent of intelligence has the foremost important and varied long consequences for system designs. the upper the intelligence, less human involvement is needed. However, it comes at the price of requiring additional subtle processing. Keeping in mind that this is often affectively imbuing intelligence in massive distributed system architectures, the authors counsel that this could be achieved by interactions between 2 disciplines admire computing and operational systems. In contrast, the agent's community has targeted on 'brain': autonomous drawback solvers that may act flexibly in unsure and dynamic environments. Yet because the scale and ambition of each Grid and agent deployments increase, we tend to see a convergence of interests, with agent systems requiring robust infrastructure and Grid systems requiring autonomous, flexible behaviors". In a more recent publication, this author describes grids and clouds as specialized forms of distributed systems. Therefore, it is reasonable to conclude that the above statement should still be applicable in the broader fields of distributed systems and AI.

Based on such reasoning it is intuitive that artificial intelligence can be used in distributed operating systems to minimize human involvement required in optimizing or facilitating crucial elements of the OS. Traditional AI research has resulted in a large number of algorithms that aid in searching distributed data quickly, enable learning from distributed data and facilitate fast planning with large amounts of information. Some of these techniques can be utilized in distributed operating systems, to optimize resource management, increase fault-tolerance, automate security, facilitate more natural user-interfaces and automate resource discovery together with naming; but is certainly not restricted to this list of benefits. The rest of this paper will contain practical as well as theoretical examples to support this fact.

I. CURRENT TECHNIQUES OF AI IN A DISTRIBUTED OS

Scheduling / Resource Management

It is the responsibility of a distributed OS to efficiently manage all available resources. These resources include processors, memory, network bandwidth, disks and other devices. A distributed system affords parallelism, yet speedup can only be achieved through concurrent execution of tasks. Since a task may consist of many processes it is important to execute them on the distributed processors in an appropriate order. This is known as process scheduling, where the main objective is minimizing the total execution time, possibly by maximizing the utilization of processors. Load balancing involves spreading the load on processors equally, thereby maximizing the utilization of processors and minimizing the total execution time, yet it may incur memory management and communication costs. Hence, scheduling in a distributed system is as an NP-complete problem. Therefore, this is an important distributed OS topic in which several AI techniques have been applied successfully.

Scheduling with Genetic Algorithms

Even in the best conditions heuristic based search methods are appropriate as they yield optimal or suboptimal solutions. Genetic Algorithms (GA) are classified under the probabilistic optimization heuristics category. GAs are a class of evolutionary algorithms designed to perform a search over a large set of data to find the best solution matching some predefined criteria. In brief, the process could be outlined as follows. Initially, a GA starts with a generation of individuals, which are encoded as strings known as chromosomes. A chromosome corresponds to a solution to the problem. A fitness function is used to evaluate the suitability or fitness of each individual. Individuals above a certain level of fitness are selected to reproduce offspring or the next generation of individuals through two operations known as mutation and crossover. The entire process repeats until the termination condition is reached, at which time the fittest solution found is returned [16].

Their algorithm mapped each schedule with a chromosome that showed the execution order of all existing processes on processors. They assumed that the distributed system is non-uniform, meaning that the processors may be different and that they are non-preemptive. In addition, their load balancing mechanism scheduled processes without process migration and was centralized.

A chromosome was encoded as an array of n digits, where n was the number of processes. The indexes showed process numbers and a digit could take any one of 1 to m values corresponding to the processors that the process was assigned to. When more than 1 process was assigned to the same processor, the left-to-right order determined their execution order on that processor. Each individual in the first generation or initial population was generated by randomly selecting an unscheduled process and assigning it to a processor selected circularly until all processes were assigned. This meant that processors were selected respectively from first to last and then back to first again. A predefined number of individuals were generated by repeating this method.

The fitness function captured the objective of finding a schedule with optimal cost, while considering load balancing, processors utilization and cost of communication.

The next generation was formed by selecting the fittest individuals from the current population as well as the newly created individuals. The entire process was repeated until the desired termination criterion was reached. This could be set to a maximum number of generations, algorithm convergence, or equal fitness for fittest selected chromosomes in respective iterations etc. At the end the fittest individual in the population was returned as the best schedule.

The experimental results were obtained through simulation. The simulated results indicated that as the number of processes increased, higher utilization was obtained. Consequently, the algorithm was able to always find a schedule such that the overall completion time would remain linear with respect to the number of processes, as opposed to exponential if the scheduling was not optimal. It was also noted that increasing the number of generations and the population size increased the quality of the best schedule found.

Scheduling with Simulated Annealing

The name and inspiration for simulated annealing (SA) come from a process in metallurgy used to strengthen a metal. In this process known as annealing the motion of atoms in metals are controlled through heating and cooling. Analogously, in SA each step replaces the current solution by a random neighboring solution, chosen with a probability that depends on the difference between the corresponding function values and on the current temperature, which is gradually

Since task scheduling is concerned with finding the least execution time, how SA could be used in a distributed OS for this purpose. A task assignment vector mapped module within each task to the processors. They argued that the total cost of an assignment in the system is equal to the maximum workload of the critical processor. Therefore, the optimal assignment had the minimum cost in all of the possible assignments subjected to constraints with memory capacities, real-time deadlines and associated communication costs based on the system topology. The execution cost of a module on a particular processor, the communication cost between modules while executing

on different processors, the connection information about the processors, the minimum storage needed for a module, the maximum capacity of local memory associated with a processor, the minimum processing time required by a module and the real-time deadline for a processor in a given task were all expressed as vectors and matrices that were combined to evaluate the cost of an assignment. In addition, a penalty was added to this cost if an assignment violated any constraints.
Given a particular assignment a neighboring solution was calculated using three strategies. They started with strategy 1 which attempted to decrease the cost by reassigning a module, if this was not successful then strategy 2 attempted to interchange two modules to decrease the cost, and if this was also not successful then in strategy 3 a detrimental swap was performed to intentionally break away from any locally optimal solutions. The initial temperature was decided based on the acceptable range of costs by satisfying the hot enough condition. The number of iterations at each temperature was bounded to a finite number and the temperature was decreased gradually to guarantee convergence. The annealing curve of cost versus temperature was considered for determining a more efficient stopping criterion.
The results obtained indicated that SA yielded near-optimal solutions for task scheduling in a distributed system in a reasonable amount of time given a set of good parameters. Despite being able to easily incorporate the system constraints, the performance of SA is heavily dependent on the number of tasks and the range of SA parameters. This makes it difficult for practical deployment within a real distributed OS. Nevertheless, SA and GA could be combined to efficiently solve the problem of process scheduling. More recently, successfully used a merger of GA and SA for task scheduling by a distributed OS.

Scheduling with A* Search

The A* algorithm is an extremely popular graph search technique that finds the least cost path from a given node to a goal node. In a state-space search problem each state is represented by a node in the graph. The cost of a node f (n) is the sum of, the distance from the start node to the given node g(n), and the estimated distance from the given node to a goal h(n). This estimate is known as a heuristic, which must remain consistent for the search to find the optimal solution path. Conceptually the algorithm works by following the lowest cost path keeping a sorted queue of alternate paths and switching to them if the path being followed has a higher cost until a goal is reached.

Shen and Tsai [9] were the first to demonstrate the use of A*search for task scheduling in a distributed system. The processor turnaround time for each processor under a particular task assignment was computed by adding the total time spent for module execution and the total time for inter processor communication delay in that processor. Therefore, the total time required to complete the whole task according to a particular assignment was the maximum processor turnaround time. Hence, the optimal task assignment was the one that minimized the maximum processor turnaround time, which was aptly named the mini-max criterion.
The processors and their interconnections were represented in the processor graph. Similarly, a task graph represented the modules and the intermodule communications of the particular task. Graph matching based on weak homomorphism was applied to match the task graphs to the processor graph, effectively facilitating task assignment. A node in the state space represented a partially developed homomorphic mapping, in which some modules of a task were assigned to processors. The A* search was performed using g(n) calculated with the mini-max criterion and h(n) was either set to 0 where pruning was minimal or set to a value based on a special cost calculation. The search terminated when all modules where assigned to processors, thereby yielding an optimal task assignment.
The experimental results have successfully validated the applicability of a distributed OS using A* search for task scheduling.

Fault-Tolerance

In distributed systems faults or failures are fairly common, they can occur due to problems in hardware as well as software. Hardware failures can occur at various locations on a distributed system, e.g. memory, processor, network, power supply etc. Software failures could manifest in poorly designed applications, protocols etc. It is the responsibility of a distributed OS to ensure that the system continues to operate gracefully in the presence of such failures. Most distributed operating systems are engineered to provide fault-tolerance using redundancy, error detection and error correction mechanism at various layers in the system. The use of redundant arrays of inexpensive disks for storage is an example of fault-tolerance using redundancy. The problem with most of these existing techniques is that they are explicitly engineered to handle a finite number of failure scenarios and are limited by the engineer's foresight. This could result in certain failures not being trapped and certain failures not being dealt with appropriately. Therefore, fault-tolerance is an important distributed OS topic in which a few AI techniques have been applied successfully.

How AI learning techniques could be used to provide fault-tolerance through on-line hardware reconfiguration. A tuning agent was designed to handle real-time reconfigurations of the distributed system. The agent would detect changes in the processing resources, memory availability and other resources such as the number of database connections. When the workload changed it would then decide whether to alter the configuration based on a model learned through statistical data acquisition. The experimental results proved that AI learning could be incorporated into a distributed OS to provide fault-tolerance.

User Interface

The user interface of a computing system defines the primary mode of information exchange between users and the system. In traditional operating systems the user interface has historically been one of the main factors influencing its popularity among users and its subsequent survival. Similarly, the support for a friendly user interface may benefit a distributed operating system. Since a distributed operating system has access to more resources than a traditional operating system, more sophisticated interface methods using AI techniques can be utilized.

A hierarchical approach was adopted to construct the NL interface. In the first phase natural language understanding was performed using a linguistic knowledgebase, which consisted of vocabulary knowledge (formation, usage and semantic meaning of words) and sentence pattern knowledge. Next, this initial translated request was further refined with the aid of the domain knowledgebase, which contained concepts related to domains of interest, e.g. the word "position" may mean location of a file in the name space or the coordinates of the cursor on the screen in a graphical application. In the final phase, the refined request was converted into application command sequences with the aid of the command knowledgebase, which described the applicable commands.

In addition to using NL for the user-interface, this OS learned belief networks (Bayesian networks) and used the hill-climbing algorithm to optimize task scheduling. The successful experimental results of this intelligent distributed operating system proved without any further doubt that AI techniques could be used to facilitate a distributed OS.

II. Theoretical Use of AI Techniques (Based on Algorithm)In Distributed OS

Search Algorithms

AI search techniques such as GA, SA, hill-climbing, Bayesian networks and informed searches including A* search can be used in distributed operating systems to improve processor management, file management, memory management and the management of all other resources. A file management technique involving the use of intelligent caching was demonstrated. Some of these processor management techniques. It is important to note that by changing a few parameters in each algorithm the entire behavior of the distributed OS can be altered without requiring any additional design considerations or further development. For example, the fitness function and

terminating criteria of a GA can be changed at any time to switch the system from meeting real-time demands to being more fault-tolerant.

At present, most of these strategies have been evaluated on their own rather than through integration into a real distributed operating system. It could be argued that these techniques can readily be integrated into microkernel architectures.

Planning and Learning Algorithms

Neural networks, decision trees, reinforcement learning using Markov decision processes and model learning algorithms such as Surprise-Based Learning (SBL) are a few powerful algorithms amongst many AI learning techniques that can be used in distributed operating systems to improve resource management, better user-interfaces, increase fault- tolerance, automate security and automate resource discovery together with naming.

In resource management, for example, decision trees can be used to classify statistics on communications between machines, which could subsequently be used to quickly distribute data to parts of the network. Reinforcement learning could also be used for similar purposes, especially if the status of the system can be represented as discrete states. Some of the feasible techniques for user-interfaces and fault-tolerance where seen in section III. Alternatively, the distributed OS can dynamically construct models of the system using model learning so as to detect and deal with resource or communication failures. A neural network can be deployed on a distributed OS and trained on known security attacks. Since a neural network is able to generalize, it may be able to cope with known attacks as well as previously unknown attacks.

Naming in a distributed system is to assign and uniquely identify users, groups, hosts, services, files, printers and other objects. Resource discovery is concerned with searching for named resources based on certain attributes. This is where model learning algorithms are well suited as they attempt to construct and maintain a model of the system which could be used in many ways.

SBL is able to autonomously construct and maintain a model by periodically querying the distributed system. Typically, the model is comprised of several rules, where each rule consists of a set of conditions, an action, and a set of predictions rules. The learner must be bootstrapped with all the possible actions, which could be a list of valid OS commands. A query would be the execution of a command such as ping with a certain number. SBL would monitor all activity in the system and record the changes it believes where caused by the execution of the command as rules in the model. The next time the same command is executed SBL has a prediction that it can test, if the prediction comes true then its model is correct, else it will split and refine the rules until the model matches the true situation of the system.

After a model is learned imagine that a new host was added or an existing host failed or a new user was added, then even if the learner is not immediately informed it will eventually find out and update the model accordingly. Thus, the distributed OS could easily use this model to perform resource management and facilitate fault-tolerance. Naturally the process of querying will also be able to handle resource discovery literally, simultaneously it could be used to resolve naming issues to some extent as the model can detect conflicts. Furthermore, a hierarchical approach could be used to distribute the model by maintaining smaller models for sub systems on separate hosts.

Knowledge about the distributed system may be represented using logic as seen in SBL. Large amounts of information about the system can be collected and stored as logic rules. In such cases there are several AI planning techniques that a distributed OS could use to make efficient decisions. Planning with propositional logic including constraint satisfaction, reasoning with first order logic, frame systems and semantic networks are a few well known AI planning techniques that could be employed by a distributed OS to ensure concurrency, prevent deadlocking etc.

Comparison of planned AI techniques for Distributed software system with existing operating Systems

Stability: mackintosh, UNIX operating system, and Windows aren't the terribly stable owing to the

operating system operate loads additional resources, hardware architects, communication that create a system cut down and crash applications. Linux Ubuntu is exceptionally very little stable as a result of of employing a lot less process power. Mac more stable because own style design hardware engineered to run with efficiency with capability. Whereas AIDOS Manage Resources, perceive hardware architecture, on Behave of economical usage of them by expertise gaining with Progression of your time on all sorts of systems Risk Protection: Linux, mackintosh's are loads less susceptible to malware than a Windows system. they're additional closed system and far lower users. However, there's additionally much less of a alternative once considering anti-malware programs. Windows vital target of malware another software system owing to larger end-users. Malware attack to infect all Windows os computers. Anti-Malware programs are accessible even Microsoft as well as Windows defender whereas AIDOS has Protection System Manage by Security Researchers and self-diagnosis, repairing, healing and risk managing intelligent mechanism. Hardware Support: Mac OS run on own system design Hardware, currently doing some Patches it put in on alternative Architecture, Virtual Machines. World's most Hardware Manufacturer style Device for Windows owing to Windows while support and add capability in redo Whereas AIOS Develop, Update Drivers for time period Use to figure on All update's version of AIOS to boot enhance, update drivers and device operative package feature. Application package Support: UNIX operating system utilized by Professionals, and Its development is adamant however terribly less support for package class have less convenience for application software because of the less high range of the user base than Windows. Additionally, there are less Objective-C developers. Windows or Linux system computers capable of interpreting. Windows support all sorts of applications package Even AI application package whereas AIDOS develop own application software as analysis. It can not be simply done however want Time and AIDOS Engineers. valuation of OS: UNIX operating system is freed from value and open supply, mackintosh OS is incredibly big-ticket and not accessible source currently Apple distributes Paid SDK, Windows less Expense and It additionally Open source now Whereas AIOS SDK Remained to the Researchers and restricted access to developers. AIDOS style structure not almost like Linux, Windows, mackintosh Os owing to Self-Learning Capable System Software. Every OS manufacturer has its architecture, User target, and dealing Limitations whereas AIDOS can mechanically produce amendment consistent with User and System usage. Specific functions continually want specific architecture. one structure doesn't cowl most of the principles for AIOS. AIDOS Limitations Dependency: - AIDOS learning depends on hardware/biological devices that are connect with its system. The hardware property determined the limitation of AIOS. AIOS Intelligence depends on the algorithmic rules which are programmed by the someone and Manufacturer. One algorithmic rule style for a particular problem. it's potential that one Algorithm disturbs another algorithm's actions and results.

CONCLUSION

Artificial intelligence research holds some promise in improving or facilitating distributed operating systems. This paper investigated several AI techniques that were practically and theoretically able to impact some major elements of a distributed OS. Genetic algorithms, simulated annealing and A* search was practically able to improve the performance of resource scheduling, while model learning provided fault- tolerance and natural language processing delivered a better user-interface. The KZ2 intelligent operating system was practical proof that a distributed OS could be facilitated with several AI techniques.

A number of AI search techniques including hill-climbing, Bayesian networks and informed searches were deemed theoretically feasible for improving processor management, file management and resource management. Suggestions were made for integrating and testing these on real distributed operating system kernels. Neural networks, decision trees, reinforcement learning using Markov decision processes and model learning algorithms were recommended for facilitating resource management, user-interfaces, fault-tolerance, security and resource discovery together with naming, in a distributed OS. The application and benefits of using Surprise-Based Learning

within a distributed OS was discussed. Finally, the theoretical applicability of several logic-based planning algorithms, frames and semantic networks was mentioned.

It can be concluded that the merger between AI and distributed operating systems will indeed yield more autonomous and robust systems, because AI provides the brain to the distributed OS to control the brawn that is the distributed system.

REFERENCES

[1] G. O'Hare and N. Jennings, "Foundations of Distributed Artificial Intelligence", John Wiley & Sons Ltd, New York, USA, April1996.
[2] M. Wooldridge, "An Introduction to Multi Agent Systems", John Wiley & Sons Ltd, New York, USA, May2002.
[3] V. Honavar, L. Millers and J. Wong, "Distributed Knowledge Networks", IEEE Information Technology Conference. Syracuse, NY, USA,1998.
[4] R. Dragomir, "AnArchitecture ForDistributed Natural Language Summarization", Workshop on Natural Language Generation, Herstmonceux, UK, June 1996,45-48.
[5] D. Cook and R. Varnell, "Maximizing the Benefits of Parallel Search Using Machine Learning", National Conference on Artificial Intelligence, 1997.
[6] D. Estrin, D. Culler, K. Pister and G. Sukhatme, "Connecting the Physical Worlds with Pervasive Networks", IEEE Pervasive Computing, vol. 1, January 2002, 59-69.
[7] M. Nikravan and M. Kashani, "A genetic algorithm for process scheduling in distributed operating systems considering load balancing", 21st European Conference on Modeling and Simulation, June2007.
[8] F. Lin and C. Hsu, "Task assignment scheduling by simulated annealing", IEEE Region 10 Conference on Computer and Communication Systems, vol. 1, September 1990,279-283.
[9] C. Shen and W. Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using a Minimax Criterion", IEEE Transactions on Computers, vol. 34, March 1985, 197- 203.
[10] J. Wildstrom, P. Stone, E. Witchel and M. Dahlin, "Machine Learning for On-Line Hardware Reconfiguration", 20th International Joint Conference on Artificial Intelligence, January 2007,1113-1118.
[11] L. Xie, X. Du, J. Chen, Y. Zheng and Z. Sun, "An introduction to intelligent operating system KZ2", ACM SIGOPS Operating Systems Review, ACM New York, USA, vol. 29, January 1995,29-46.
[12] N. Ranasinghe and W-M. Shen, "Surprise-Based Learning for Developmental Robotics", ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems, Edinburgh, UK, August2008.
[13] M. Weiser, "The Computer for the 21st Century", Scientific American, vol. 265, September 1991,94-104.
[14] I. Foster, N. Jennings and C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other", International Conference on Autonomous Agents and Multiagent Systems, New York, USA, July 2004,8-15.
[15] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", Grid Computing Environments Workshop, November 2008, 1-10.
[16] L. Schmitt, "Fundamental Study Theory of Genetic Algorithms", International Journal of Modelling and Simulation Theoretical Computer Science, vol. 259, May 2001,1-61.
[17] A. Omara and M. Arafa, "Genetic Algorithms for Task Scheduling Problem", Studies in Computational Intelligence, Springer Berlin, Heidelberg, Germany, vol. 203, May 2009,479-507.
[18] A. Elleuch, R. Kanawati, T. Muntean and E-G. Talbi, "Dynamic Load Balancing Mechanisms for a Parallel Operating System Kernel", Lecture Notes InComputer Science, Springer Berlin, Heidelberg, Germany, vol. 854, January 1994, 866-877.
[19] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing", Science, vol. 220, May 1983,671-680.
[20] Karp RM (1972) Reducibility among combinatorial problems. Complexity of ComputerComputations, eds Miller RE, Thatcher JW (Plenum, New York), pp 85–103
[21] C. Price and M. Salama, "Scheduling of Precedence-Constrained Tasks on Multiprocessors", The Computer Journal, vol.33, February 1990, 219-229.

