A Distributed Agent Implementation of Multiple Species Flocking Model for Document Partitioning Clustering

Xiaohui Cui and Thomas E. Potok

Oak Ridge National Laboratory
Oak Ridge, TN 37831-6085
{Cuix, potokte}@ornl.gov

Abstract. The Flocking model, first proposed by Craig Reynolds, is one of the first bio-inspired computational collective behavior models that has many popular applications, such as animation. Our early research has resulted in a flock clustering algorithm that can achieve better performance than the K-means or the Ant clustering algorithms for data clustering. This algorithm generates a clustering of a given set of data through the embedding of the high-dimensional data items on a two-dimensional grid for efficient clustering result retrieval and visualization. In this paper, we propose a bio-inspired clustering model, the Multiple Species Flocking clustering model (MSF), and present a distributed multi-agent MSF approach for document clustering.

Keywords: Swarm, Bio-inspired, Clustering, Agent, Flocking, VSM.

1 Introduction

Currently, more and more digital document data is being generated as part of the ubiquitous and pervasive use of computing systems, information systems, and sensor systems. It is a challenge to efficiently and effectively analyze this data. Clustering analysis is a descriptive data mining task, which involves dividing a set of objects into a number of clusters. The motivation behind clustering a set of data is to find inherent structure inside the data and expose this structure as a set of groups [1]. The data objects within each group should exhibit a large degree of similarity while the similarity among different clusters needs be minimal [9]. Document clustering is a fundamental operation used in unsupervised document organization, automatic topic extraction and information retrieval. It provides a structure for organizing a large body of text for efficient browsing and searching. There are two major clustering techniques: partitioning and hierarchical [9]. Many document clustering algorithms can be classified into these two groups. In recent years, it has been recognized that the partitioning techniques are well suited for clustering a large document dataset due to their relatively low computational requirements [18]. The best-known partitioning algorithm is the K-means algorithm and its variants [17]. This algorithm is simple, straightforward and based on the firm foundation of analysis of variances. One drawback of the K-means algorithm is that the clustering result is sensitive to the selection of the initial cluster centroids and may converge to the local optima, instead of the global one. The other limitation of the K-means algorithm is that it generally requires a prior knowledge of the probable number

M. Klusch, M. Rovatsos, and T. Payne (Eds.): CIA 2006, LNAI 4149, pp. 124 – 137, 2006.

of clusters for a document collection. Therefore, there is a demand for more efficient algorithms for document clustering.

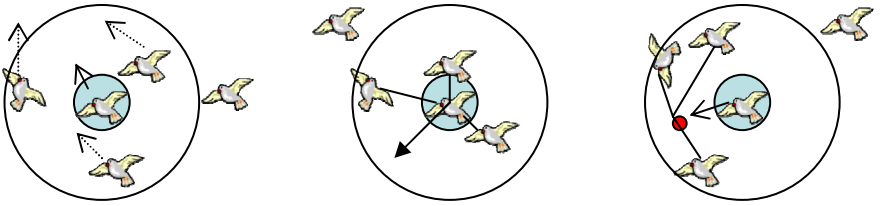
New algorithms based on biological models, such as ant colonies, bird flocks, and swarm of bees etc., have been invented to solve problems in the field of computer science. These algorithms are characterized by the interaction of a large number of agents that follow the same rules and exhibit complex, emergent behavior that is robust with respect to the failure of individual agents. The Flocking model is one of the first collective behavior models that have been applied in popular applications, such as animation. In addition to being used to simulate group motion, which has been used in a number of movies and games, The Flocking model has already inspired researches in time varying data visualization [12, 20] and spatial cluster retrieval [6, 7]. In this paper, we propose a bio-inspired clustering model, the Multiple Species Flocking clustering model (MSF), and present a distributed multiple agent MSF approach for dynamic updated text clustering.

The remainder of this paper is organized as follows: Section 2 provides a general overview of the basic Flocking model. A new multiple species flocking (MSF) model is proposed and a MSF model clustering algorithm is described in section 3. In section 4, a Multi-Agent Scheme for Distributed Dynamic Document Clustering is presented. Section 5 provides detailed experimental design, setup and results in comparing the performance of the multi-agent implementation for clustering the dynamic updated document collection on the cluster computer and a single processor computer. Section 6 describes the related works in the traditional and bio-inspired document clustering area. The conclusion is in Section 7

2 Modeling of Flocking Behavior

Social animals or insects in nature often exhibit a form of emergent collective behavior known as '*flocking*'. The Flocking model is a bio-inspired computational model for simulating the animation of a flock of entities. It represents group movement as seen in the bird flocks and the fish schools in nature. In this model, each individual makes its movement decisions on its own according to a small number of simple rules that it reacts to its neighboring members in the flock and the environment it senses. These simple local rules of each individual generate a complex global behavior of the entire flock. The basic Flocking model was first proposed by Craig Reynolds [14], in which he called each individual as "boid". This model consists of three simple steering rules that each boid need to execute at each instance over time: (1) Separation: Steering to avoid collision with other boids nearby; (2) Alignment: Steering toward the average heading and match the velocity of the neighbor flock mates (3) Cohesion: Steering to the average position of the neighbor flock mates.

As shown in Figure 1, in the circled area of Figure 1(a), 1(b) and 1(c), the boid's (located in the center of the small circle with grey background) behavior shows how a boid reacts to other boids' movement in its local neighborhood. The degree of locality is determined by the range of the boid's sensor (The semi-diameter of the big circle). The boid does not react to the flock mates outside its sensor range because a boid steers its movement based only on local information. These rules of Reynolds's boid flocking behavior are sufficient to reproduce natural group behaviors on the computer.



(a) Alignment (b) Separation (c) Cohesion

Fig. 1. The three basic rules in the boid

3 The Multiple Species Flocking (MSF) Model

Our early experiments [3] indicate these three rules in Reynolds's flocking model will eventually result in all boids in the simulation forming a single flock. It can not reproduce the real phenomena in the nature: the birds or other herd animals not only keep themselves within a flock that is composed of the same species or the same colony creatures, but also keep two or multiple different species or colony flocks separated. To simulate this nature phenomenon, we propose a new Multiple Species Flocking (MSF) model to model the multiple species bird flock behaviors. In the MSF model, in addition to these three basic action rules in the Flocking model, a fourth rule, the feature similarity rule, is added into the basic action rules of each boids to influence the motion of the boids. Based on this rule, the flock boid tries to stay close to these boids that have similar features and stay away from other boids that have dissimilar features. The strength of the attracting force for similar boids and the repulsion force for dissimilar boids is inversely proportional to the distance between the boids and the similarity value between the boids' features.

In the MSF model, we use the following mathematical equations to illustrate these four action rules for each boid:

Alignment Rule:

$$d(P_x, P_b) \le d_1 \cap (P_x, P_b) \ge d_2 \Rightarrow \vec{v}_{ar} = \frac{1}{n}\sum_{x}^{n} \vec{v}_x \ . \tag{1}$$

Separation Rule:

$$d(P_x, P_b) \le d_2 \Rightarrow \vec{v}_{sr} = \sum_{x}^{n} \frac{\overline{\vec{v}_x + \vec{v}_b}}{d(P_x, P_b)} \ . \tag{2}$$

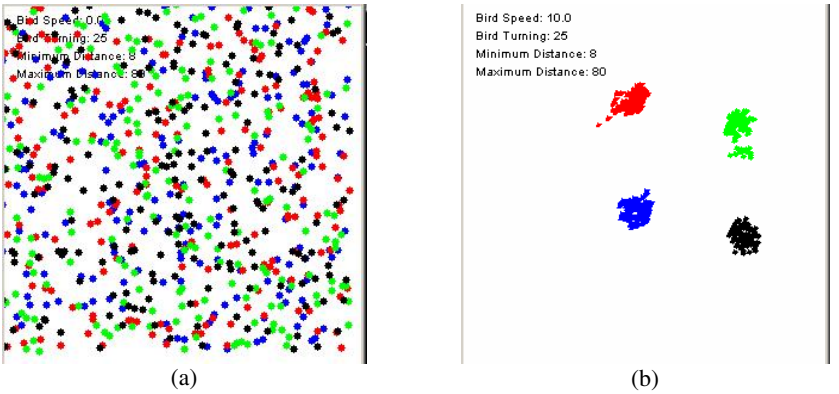
Cohesion Rule:

$$d(P_x, P_b) \le d_1 \cap (P_x, P_b) \ge d_2 \Rightarrow \vec{v}_{cr} = \sum_{x}^{n} \overrightarrow{(P_x - P_b)} \ . \tag{3}$$

Feature Similarity Rule:

$$v_{ds} = \sum_{x}^{n} \frac{(S(B,X) - T) * \overrightarrow{(P_x - P_b)}}{d(P_x, P_b)} \ . \tag{4}$$

where $\boldsymbol{v_{ar}}$, $\boldsymbol{v_{sr}}$, $\boldsymbol{v_{cr}}$ and v_{ds} are velocities driven by the four action rules, $d(P_x,P_b)$ is the distance between boid B and its neighbor X, n is the total number of the boid B's local neighbors, v_b and v_x is the velocity of boid B and X, d_1 and d_2 are pre-defined distance values and $d_1 \succ d_2$, $\overrightarrow{P_x - P_b}$ calculates a directional vector point. $S(B,X)$ is the similarity value between the features of boid B and X. T is the threshold for separating similarity and dissimilarity boids.



(a) (b)

Fig. 2. Multiple species bird flocking simulation

To achieve comprehensive flocking behavior, the actions of all four rules are weighted and summed to give a net velocity vector demanded for the active flock boid.

$$v = w_{sr}.v_{sr} + w_{ar}.v_{ar} + w_{cr}.v_{cr} + w_{ds}.v_{ds} \ . \tag{5}$$

where v is the boid's velocity in the virtual space and $w_{sr}, w_{ar}, w_{cr}, w_{ds}, w_{dd}$ are pre-defined weight values.

Figure 2 shows the result of our multiple species bird flock simulation by using multiple agents system in which the MSF model is implemented in each simulation agent. In this simulation, there are four different boid species and each species have 200 boids. We use four different colors, green, red, blue and black, to represent different species. All together, 800 boids are simulated in the environment. At the initial stage, each boid is randomly deployed in the environment as shown in Figure 2(a). Each color dot represents one boid agent. There is no central controller in the simulation. Each boid agent can only sense other boids within a limited range and move in the simulation environment by following these four action rules of the MSF model. Although there is no intention for each boid to form a same species group and to separate different species from each other, after several iterations, as shown in Figure 2(b), the boids in the same species (shown as in same color) are grouped together and different species are separated. This phenomenon represents an emergent clustering behavior.

4 The MSF Clustering Algorithm

The MSF model could offer a new way to cluster datasets. We applied the MSF model to developing a document collection clustering algorithm called MSF Clustering algorithm. The MSF clustering algorithm uses a simple and heuristic way to cluster input data, and at the same time, maps the data to a two-dimensional (2D) surface for easy retrieval and visualization of the clustering result, processing both tasks simultaneously. In the MSF clustering algorithm, we assume each document vector is projected as a boid in a 2D virtual space. Each document vector is represented as a feature of the boid. Following the simple rules in MSF model, each boid determines its movement by itself in the virtual space. Similar to the bird in the real world, the boids that share similar document vector features (same as the bird's species and colony in nature) will automatically group together and became a boid flock. Other boids that have different document vector features will stay away from this flock. In this algorithm, the behavior (velocity) of each boid is only influenced by the nearby boids. The boid's four MSF action rules react to this influence and generate the boid's new moving velocity. Although this influence on each bird is locally, the impacts on the entire boid group is global. After several iterations, the simple local rules followed by each boid results in generating a complex global behavior of the entire document flock, and eventually a document clustering result is emerged.

We evaluated the efficiency of the MSF algorithm and the K-means algorithm on document collection that includes 112 recent news articles collected from the Google news. This news article collection has been categorized by human and manually clustered into 11 categories. For the purpose of comparing, the Ant document clustering [8] and the K-means clustering algorithms were implemented by Java language and applied to the same real document collection dataset, respectively. The K-means algorithm implementation was given the exact clustering result number as the prior knowledge. Our early research [3] shows that the Ant clustering algorithm can not come out any useful result if the algorithm only given a limited number of iteration (300 iterations) for refining the result. In this experiment, each algorithm was given 100 fixed iterations to refine the clustering result and only the MSF clustering algorithm and K-means algorithm can generate reasonable results. As shown in Figure 2(b), the clustering results generated by the MSF clustering algorithm can be easily recognized by human eyes because of their visual characteristic. In our experiments, the clustering result of the MSF clustering algorithm is retrieved by human looking at the visual flock picture that generated by the virtual boids on the screen. We compared the average results of these two algorithms from ten separate experiments. The results of the clustering algorithm were evaluated by comparing it with the prior knowledge of the classification of the document collection. The F-measure was used as the quality measure. The results are listed in Table 1. The results indicate that the flocking algorithm achieves better result compared to the K-means for document clustering although the K-means algorithm has prior knowledge of the exact cluster number.

Table 1. Performance results of the K-means and MSF clustering algorithms

Algorithms	Average cluster result number	Average F-measure value
MSF	9.105	0.7913
K-means	(11)	0.5632

5 Distributed Agent Implementation of MSF Clustering Algorithm

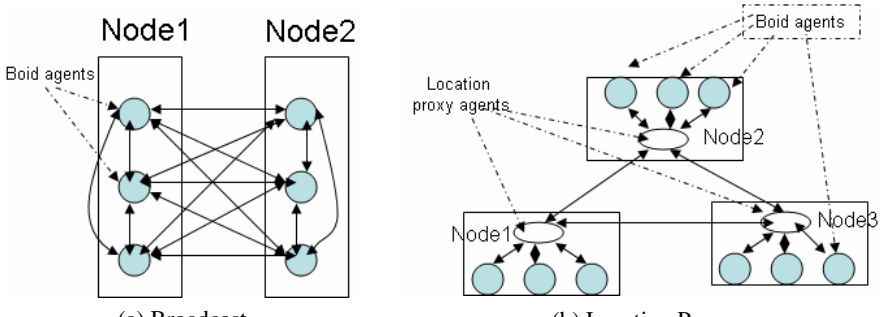
The MSF clustering algorithm can achieve better performance in document clustering than the K-means and the Ant clustering algorithm. This algorithm can continually refine the clustering result and quickly react to the change of individual data. This character enables the algorithm suitable for clustering dynamic changed document information, such as the text information stream. However, the computational requirement for real-time clustering a large amount of text collection is high. In the information society of today, tremendous amounts of text information are continuously accumulated. Inevitably, the MSF clustering algorithm approach of using single processor machine to cluster the dynamic text stream requires a large amount of memory and a faster execution CPU. Since the decentralized character of this algorithm, a distributed approach is a very natural way to improve the clustering speed of this algorithm. In this paper, we present a distributed multi-agent based flocking approach for clustering analysis of dynamic documents and balance the computation load on cluster nodes.

5.1 Distributed Agent Scheme for Document Clustering

In the MSF clustering algorithm, the document parse, similarity measurement and boid moving velocity calculation are the most computational consumption parts. The distributed implementation can divide these computational tasks into smaller pieces that may be scheduled to concurrently run on multiple processors. In order to achieve better performance using distributed computing, several issues must be examined carefully when designing a distributed solution. First is the load balance. It is important to keep load balancing among processing nodes to make sure each node have approximately the same workload. The environment state synchronization is the second issue need to be considered. It is very important for a distributed implementation to develop a synchronization algorithm, which is capable of maintaining causality. Third is reducing the communication between nodes, including communication overhead of the environment state synchronization and control of message exchange between nodes. Based on these requirements, we developed a distributed agent based implementation of the MSF clustering algorithm for clustering analysis of the text datasets. In this distributed agent based implementation, boids are modeled and implemented in terms of agents, which makes boids pro-active, adaptive and communicable. The distributed agent based implementation supports distributed load balance in a very natural way. Since each boid agent is implemented to perform document retrieval, parse, similarity comparison and moving velocity calculation independently, it is straight-forward to have different agents run on different machines to achieve a load balance. Since agent can be added, removed or moved to other machine without interrupting other agent's running, the system can be scalability and pro-activity to the change of work load.

One major concern in designing this distributed agent based MSF implementation is how to ensure agents be synchronized at any time when they must interact or exchange data. In a distributed system, environment information is spread out among the processors involved in the system. An agent doesn't know other agent's information if it is not informed, it has to commute with other agents to collect enough information, does an exhaustive search to find out which agents are located

within its range, and calculates the force that it is pushed to travel based on it's neighbor agents' information. All these require that each agent in the system have a global view of other agents' status information. As such, it is necessary to develop a communication schemes to update the agent's information on different processors. One easy communication scheme is broadcast. As shown in Figure 3(a), each agent in the system broadcast its status information to all other agents wherever they are located in the same node or different nodes. Each agent will also use the information it received from other agents' broadcast to find out its neighbor boid mates and calculate the next moving velocity. In this scheme, each agent has a global view of the entire system status. However, the broadcast will use so much bandwidth that makes the network bandwidth in a computer cluster become a bottleneck of the system when the agent number increased. In this report, we proposed an environment status sharing scheme by using location proxy agent. As shown in Figure 3(b), there is a location proxy agent on each node. Each agent will only inform its status to the location proxy agent in the same node. The agent also inquires the location proxy agent to find out its neighbor mates. At every time step, after collecting the status of all agents that located in the same host, location proxy agents will broadcast this information to other proxy agents that located on different nodes, which enable the location proxy agent on each node to have global view of the whole system.



(a) Broadcast (b) Location Proxy

Fig. 3. The architectures of different communication schemes

5.2 Datasets

The document dataset used in this study is derived from the TREC-5, TREC-6, and TREC-7 collections [10] and represented as a set of vectors $X=\{x_1, x_2, \ldots, x_n\}$, where the vector x_i corresponds to a single object and is called "*feature vector*" that contains proper features to represent the object. The feature value is represented using the Vector Space Model (VSM) [16]. In this model, the content of a document is formalized as a point in a multi-dimensional space and represented by a vector x, such as $x=\{w_1, w_2, \ldots.. w_n\}$, where $w_i(i = 1,2,\ldots,n)$ is the term weight of the term t_i in one document. The term weight value w_i represents the significance of this term in a document. To calculate the term weight, the occurrence frequency of the term within a document and in the entire set of documents needs to be considered. The most

widely used weighting scheme combines the Term Frequency with Inverse Document Frequency (TF-IDF) [15]. The TF-IDF weight w_{ij} of term i in document j is given in following equation:

$$w_{ji} = tf_{ji} * idf_{ji} = \log_2(1 + tf_{ji}) * \log_2(\frac{n}{df_{ji}}) \ . \quad (6)$$

Where tf_{ji} is the number of occurrences of term i in the document j; df_{ji} indicates the term frequency in the document collections; and n is the total number of documents in the collection.

Calculation of the TF-IDF weight value needs the knowledge of word frequency in the entire document collection and the total number of documents in the collection. If a single document is added or removed from the document collection, the TF-IDF scheme will need recalculate the TF-IDF value of all documents processed. It is difficult to use the TF-IDF scheme to convert streaming textual information into vectors. To address these issues, a modified TF-IDF scheme, Term Frequency / Inverse Corpus Frequency (TF-ICF) [13], is adopted to calculate the term weight value of each term in the document vector. In TF-ICF scheme, the TF portion is same as the TF portion in TF-IDF. The IDF calculation that uses document collection in TF-IDF is replaced with information gathered from a large, static corpus of documents in TF-ICF. The corpus includes more than 250,000 documents that contain almost all of the typically used English words. The weight w_{ij} of term i in the document j can be calculated by the following TF-ICF equation:

$$w_{ji} = \log_2\left(1 + tf_{ji}\right) * \log_2(\frac{n+1}{C_i + 1}) \ . \quad (7)$$

where C_i is the number of documents in the corpus C where term i occurs.

Before translating the document collection into TF-ICF VSM, the very common words (e.g. function words: “a”, “the”, “in”, “to”; pronouns: “I”, “he”, “she”, “it”) are stripped out completely and different forms of a word are reduced to one canonical form by using Porter’s algorithm [11].

As we indicated in the previous session, the nature of the MSF clustering algorithm enable the algorithm continually refine the clustering results and quickly react to the change of the document contents. This character makes the algorithm suitable for cluster analyzing dynamic changed document information. In this report, the performance of these algorithms on clustering dynamic updated document collections is studied. To simulate the dynamic updated document collection, the document vector of each agent is periodically updated with a new document vector and the old document vector is considered as expired. To easily compare the performance of different scenario, in this study, each agent’s document feature will be updated for ten times during the entire life of the system execution. In each experiment, the system will run 1000 cycles and the average document update gap is 100 time-steps.

5.3 Multi-agent Platform

The distributed MSF clustering algorithm is implemented on a (Java Agent DEvelopment Framework (JADE) agent platform. JADE is a software framework

fully implemented in the Java language and is a FIPA compliant agent platform. As a distributed agent plate form, the JADE agent can be split on several hosts. The OS on each host is not necessary same. The only required environment is a Java virtual machine (JVM). Each JVM is a basic container of agents that provide a complete run time environment for agents and allow several agents to concurrently execute on the same container, JVM.

5.4 Experimental Design and Results

The simulation experiment in this study is to illustrate the performance enhancement by comparing the run time of executing the MSF clustering distributed agent implementation on a three-node cluster machine and a single processor machine.

In the MSF clustering distributed agent implementation, each boid is implemented as a Jade agent. Each agent has the ability to calculate its moving velocity based on the four actions rules as we discussed in the previous session. Each agent carries a feature vector representing a document vector. The environment used in the experiment consists of a continuous 2D plane, in which boid are placed randomly on a grid within a 4000×4000 squire unit area. All experiments were carried out on an experiment Linux computer cluster machine. The cluster machine consists of one head node, ASER and three cluster nodes, ASER1, ASER2, and ASER3, which are connected with a Gigabit Ethernet switch. Each node contains a single 2.4G Intel Pentium IV processor and 512M memory. To compare the performance, we utilize a *starter* agent that initiates the boid agent process and measures time. The running times for different number of agents is recorded using java's System.currentTimeMillis() method and the unit is milliseconds.

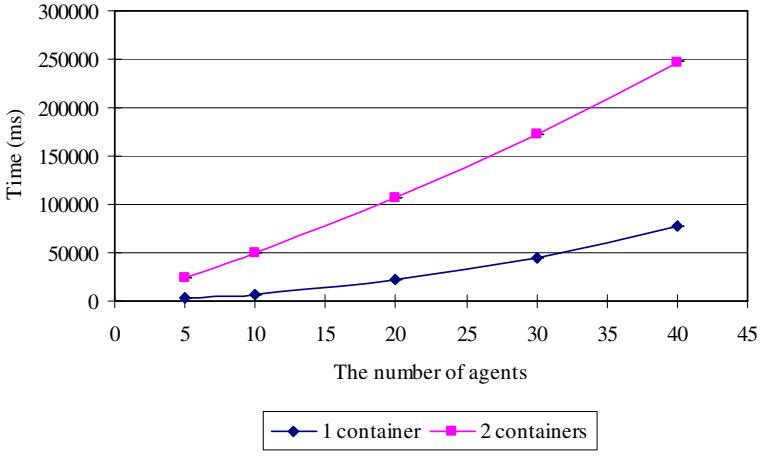
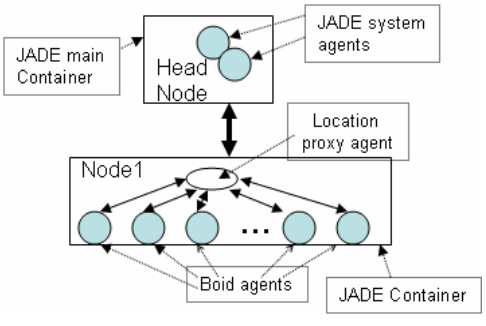
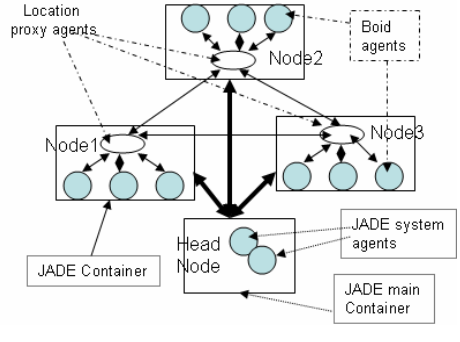


Fig. 4. The running time for boid agents deployed in one JADE container and two JADE containers

JADE allows multiple JADE containers (JVM) running on the same host while agents can be deployed in different containers. Our preliminary experiment is to test the performance impact when the boid agents running in multiple JADE containers. The running time of a different number of boid agents executed in one container or two containers are measured and recorded, separately. The experiment result is shown in Figure 4. As shown in this figure, the running time for the same amount of agents in a single container is much less than that in two containers. The main reason is that the communication between agents located in different JADE containers is much slower than the communication between agents located in the same JADE container. To reduce the communication delay, in the following experiments, all agents located in same host are assigned in the same container. At the same time, to reduce the impact of the JADE system computational requirement, in all simulation experiments, the main JADE system container runs on the head node of the cluster, which is not counted in the simulation nodes. Every simulation experiment will be executed for ten times. Reported results are the average time over 10 simulation runs of 1000 cycle each. The running time does not include the time for starting and finishing agents. It only counts the time consumed during boid agents start moving in the 2D space and stop moving after 1000 cycles.



(a)



(b)

Fig. 5. The architecture of the single processor model and the distributed model

In the single processor model, all boid agents are executed on one cluster node. In the distributed model, the boid agents are equally distributed on three nodes, each node has one location proxy agent to collect the agent position and the location proxy agent on each node will exchange agent position information at every step. The architecture of the single processor model and the distributed model are represented in Figure 5(a) and 5(b), respectively. Different numbers of boid agents are tested on both simulation and the boid agent's execution time to finish 1000 circle is recorded. Because the distributed model requires three processes to simulate the document clustering, the time is the average time consumption for all agents running on different node after 1000 cycles. The experiment results are shown in Figure 6. The "Three nodes" curve line in Figure 6 indicates the time consumption of the document clustering simulation executed on the three nodes cluster machine. The "One node" curve line indicates the time consumption of the document clustering simulation executed on the single node machine.

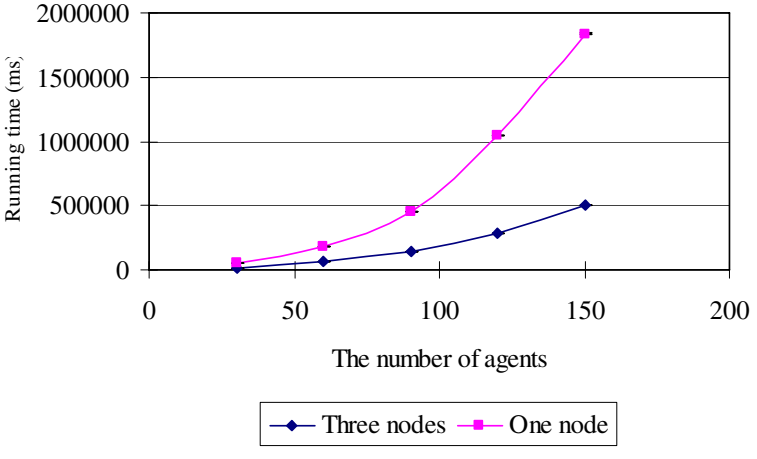


Fig. 6. The running time for 3 node cluster machine and one single processor machine

As shown in Figure 6, when the number of agents is 30, there is no significant difference on consumption time between the three node cluster machine and the single processor machine. When the number of agents is more than 60, it takes the three node cluster machine much less time than the single node machine. Before the total boid agent number reach 120, the three nodes simulation didn't cut the total running time into one third of the total time of the single node machine because of the communication overhead when location proxy agent updating status with other location proxy agents located on the other nodes. However, the running time consumption on the single node machine increases faster than that on the three node machine. Once the total boid agent number researches 120, the time required for running on the single node machine is more than three times of that on the three node machine. One possible reason is each node having limited memory (512M). In single node model, when more than 120 agents running on single node, depending on the documents that these agents represent, the memory requirement for the simulation

may be larger than the actual memory of the computer node, which cause the computer system use the virtual memory (hard disk space) and the time requirement for finishing the simulation is largely increased. In the distributed model, the boid agents are evenly deployed on three different cluster nodes. Each node only have one third of the total boid agents and the memory requirement is related smaller than single node model. This will avoid the agent system exceed the node's physical memory limitation.

6 Related Works

To deal with the limitations existed in the traditional partition clustering methods, in recent years, a number of computer scientists have proposed several approaches inspired from biological collective behaviors to solve the clustering problem, such as Genetic Algorithm (GA) [2], Particle Swarm Optimization (PSO) [4, 19], Ant clustering [8, 22] and Self-Organizing Maps (SOM) [21]. Within these clustering algorithms, the Ant clustering algorithm is a partitioning algorithm that does not require a prior knowledge of the probable number to clusters or the initial partition. The Ant clustering algorithm was inspired by clustering of corpses and eggs observed in the real ant colony. Deneubourg et al [5] proposed a "Basic Model" to explain the ants' behavior of piling corpses and eggs. In their study, a population of ant-like agents randomly moved in a 2D grid. Each agent only follows one simple rule: randomly moving in the grid and establishing a probability of picking up the data object it meets if it is free of load or establishing a probability of dropping down the data object if it is loading the data object. After several iterations, a clustering result emerges from the collective activities of these agents. Wu [22] and Handl [8] proposed the use of the Ant clustering algorithms for document clustering and declared that the clustering results from their experiments are much better than those from the K-means algorithm. However, in the Ant clustering algorithm, clustered data objects do not have mobility by themselves. The movements of data objects have to be implemented through the movements of a small number of ant agents, which will slow down the clustering speed. Since each ant agent, carrying an isolated data object, does not communicate with other ant agents, it does not know the best location to drop the data object. The ant agent has to move or jump randomly in the grid space until it finds a place that satisfies its object dropping criteria, which usually consumes a large amount of computation time. In this paper, we present a novel MSF clustering approach for document clustering analysis. Similar as the Ant clustering algorithm, the MSF clustering algorithm is a partitioning algorithm and does not require a prior knowledge of the cluster number in the datasets. It generates a clustering of a given set of data through projecting of the high-dimensional data items on a two-dimensional grid for easy retrieval and visualization of the clustering result. However, the MSF clustering algorithm is more efficient than the Ant clustering algorithm because each document object in the collection is projected as an agent moving in a virtual space, and each agent's moving activity is heuristic as opposed to the random activity in the Ant clustering algorithm.

7 Conclusion

In this study, we proposed a new multiple species flocking (MSF) model and presented a distributed multi-agent approach for the MSF clustering algorithm. In this algorithm, each document in the dataset is represented by a boid agent. Each agent follows four simple local rules to move in the virtual space. Agents following these simple local rules emerge complex global behaviors of the whole flock and eventually the agents that carrying document belong to the same class will gradually merge together to form a flock. All agents are evenly deployed on different nodes in a distributed computing environment for load balancing purposes. On each node, a location proxy agent is introduced for maintaining the agents' location and synchronizing the status between nodes in the cluster machine.

The advantage of the MSF clustering algorithm is the heuristic principle of the flock's searching mechanism. This heuristic searching mechanism helps bird agents quickly form a flock and reactive to the change of any individual document. Since the bird agent in the algorithm continues fly in the virtual space and join the flock it belongs to, new results can be quickly re-generated when the information stream is continually feed into the system.

Acknowledgments. Prepared by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285, managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725.

References

1. Anderberg, M.R.: Cluster Analysis for Applications. Academic Press, Inc., New York (1973)
2. Casillas, A., De Gonzalez Lena, M.T., Martinez, R.: Document clustering into an unknown number of clusters using a genetic algorithm. 6th International Conference, TSD 2003, Sep 8-12 2003, Vol. 2807. Springer Verlag, Heidelberg, D-69121, Germany, Ceske Budejovice, Czech Republic 43-49
3. Cui, X., Gao, J., Potok, T.E.: A Flocking Based Algorithm for Document Clustering Analysis. Journal of System Architecture (2006)
4. Cui, X., Potok, T.E.: Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm. Journal of Computer Sciences Special Issue (2005) 27-33
5. Deneubourg, J.L., Goss, S., SendovaFranks, N., Detrain, C., Chretien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats. MIT Press, Cambridge, MA, USA 356-363
6. Folino, G., Forestiero, A., Spezzano, G.: Discovering clusters in spatial data using swarm intelligence. 7th European Conference, ECAL 2003, Sep 14-17 2003, Vol. 2801. Springer Verlag, Heidelberg, Germany, Dortmund, Germany 598-605
7. Folino, G., Spezzano, G.: Sparrow: A Spatial Clustering Algorithm using Swarm Intelligence. 21st IASTED International Multi-Conference on Applied Informatics, Feb 10-13 2003, Vol. 21. Int. Assoc. of Science and Technology for Development, Calgery - Alberta, T3B OM6, Canada, Innsbruck, Austria 50-55

8. Handl, J., Knowles, J., Dorigo, M.: Ant-based clustering and topographic mapping. Artificial Life **12** (2006) 35-61
9. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31 (1999) 264-323
10. NIST: TREC (Text Retrieval Conference). http://trec.nist.gov (1999)
11. Porter, M.F.: An algorithm for suffix stripping. Program 14 (1980) 130-137
12. Proctor, G., Winter, C.: Information flocking: data visualisation in virtual worlds using emergent behaviours. Virtual Worlds First International Conference, VW'98 Proceedings, 1-3 July 1998. Springer-Verlag, Paris, France 168-176
13. Reed, J.: TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams. Technical Report. Oak Ridge National Laboratory (2006)
14. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics (ACM) 21 (1987) 25-34
15. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management 24 (1988) 513-523
16. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Cornell Univ., Ithaca, NY, USA (1974) 34
17. Selim, S.Z., Ismail, M.A.: K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6 (1984) 81-87
18. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. KDD Workshop on Text Mining
19. Van D. M., D.W., Engelbrecht, A.P.: Data clustering using particle swarm optimization. 2003 Congress on Evolutionary Computation, 8-12 Dec. 2003, Vol. Vol.1. IEEE, Canberra, ACT, Australia 215-220
20. Vande Moere, A.: Information flocking: time-varying data visualization using boid behaviors. Proceedings. Eighth International Conference on Information Visualization, 14-16 July 2004. IEEE Comput. Soc, London, UK 409-414
21. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE Transactions on Neural Networks 11 (2000) 586-600
22. Wu, b., Shi, Z.: A clustering algorithm based on swarm intelligence. 2001 International Conferences on Info-tech and Info-net. Proceedings, 29 Oct.-1 Nov. 2001, Vol. vol.3. IEEE, Beijing, China 58-66