Department of Computer Science

Submitted in part fulfilment for the degree of MEng.

Swarm Memory

Harry Burge

Version 1.0, 2020-November

Supervisor: Simon O'Keefe

Acknowledgements

Contents

List of Figures

List of Tables

Executive Summary

1 Introduction

Swarm intelligence/mechanics is an increasingly more important area of research for society, as the world moves towards a distributed technology future. Swarm intelligence can be viewed as distributed problem solving[2, 5], this is ever becoming more relevant as computer systems start to level out in terms of individual performance [7] and parallelism is embraced, satisfying the demand of the age of big data [9]. Swarm mechanics/robotics are on the rise in the industry, as society's pace increases and manual labor is automated out, whether its drone delivery to inpatient customers or mapping areas in dangerous environments [1].

An area of swarm intelligence research is distributed and local memory of swarm-like agents. This has gone down a route more to do with the optimization of distributed problem-solving algorithms rather than practical applications of storage of abstract ideas as a collective. There is a relative lack of research into collective memory on swarm like agents, within the scope of a practical setting. This would be invaluable to applications like mapping of a dangerous area [2], being able to handle the loss of agents and the collection of data on agents with limited memory. An explanation for this to be a less developed area of study is the existence of subjects like cloud-based and raid based storage systems.

Storing of data on an ever-changing network of storage devices is a hard task to complete, handling the loss of connection between different servers, reliability to access of the data and handling loss of services, whether it be non-correlated or correlated failures [9]. This is very applicable to swarm memory, with the handling of data across a swarm. However, must be adapted due to current algorithms such as RAID not being designed for highly dynamic systems such as swarm. There are promising papers in this field of cloud storage that suggest approaches that can be adapted with few modifications to apply to a swarm-based system [10], and optimized through RAID.

The objectives of this report are to merge three different areas of study into one by using knowledge of each, to create a suitable storage policy for swarm like agents to store collective memories of abstract ideas, then to perform analysis on a variety of simulations to explore the capability of the said storage policy.

1 Introduction

// Talk about the sections of the report once completed.

2 Literature Review

This chapter will review two areas of relevancy to the project proposed, these are Cloud/Backup storage policies/schemes and Swarm robotics. Ideas and concepts from both areas will have to be relied upon for the completion/design of the storage policy.

2.1 Cloud/Backup storage policys/schemes

Like most things in computer science, this area of study used to be simple, with small data-warehouses and backups on to a medium like magnetic tape, following a grandfather, father, son backup policy. As the years have progressed, technology has become greatly complex requiring larger files to be stored and accessed frequently. Leading to the need for complex backup systems to provide availability and longevity of data stored, across a network or even locally. A component to the lead of complexity of these algorithms other than providing a service better than competitors is the Legal Services Act. 2007 [11]. This enforces that a company's cloud storage solution has to be reliable and fast in data collection for users.

Most algorithms used in production are called random replication policies [9], this is where data is partitioned and randomly distributed among other storage devices usually on different racks of the datacenter. This is an efficient design policy for handling non-correlated errors, however, lacks the robustness against correlated errors. These algorithms are substantial for long-term data storage with average popularity of collection/use of that data.

A non-correlated error is when devices go down randomly for example in a swarm an agent's battery might explode, destroying that singular agent, therefore it can be modeled by a random chance of happening on each member. A correlated error happens when multiple agents go down due to a reason unbenounced to us. Following on from our swarm example, let us say a tsunami hits a certain section of the swarm destroying those agents therefore meaning that those failures were connected by some sort of event. In data-warehouses, this is usually a failure of power on a server or a bug, not as permanent damage as a tsunami.

Some of the problems arisen from random replication policy has spawned new replication policies, which can handle correlated and non-correlated errors, more effectively whilst also taking into account the demand of such items stored[9, 10]. Tackling these problems, two approaches were undertaken. The first approach is to come from a higher level of control where you have a manager which can choose items to replicate and where based on demand, knowledge of other replications and outside factors [9, 12]. This does not only apply to data but also schema changes of said servers or databases [13]. Working versions of these over a cloud service are tied in/together with a "Distributed key-value store" [14] where you have these key-value pairs on multiple devices on a network where duplication only leads to more fault tolerance of the data stored.

Another way to handle this which doesn't rely on a more privileged controller, and creates a distributed system is by having something like "SKUTE" as proposed in [10]. Each individual key-value replication has its own manager and can choose what it as a singular entity can do on that distributed system. The policy as described in "SKUTE" are as follows; Migration, Suicide, Replication, and Nothing. Migration is the moving of its data to lower costing and more redundant servers. Suicide is the removal of itself usually based on the number of duplicates and uses something like Paxos [15] to decide if it should suicide. Replication decides that it is being used enough to warrant the need to be duplicated.

An algorithm like "SKUTE" e.g. scattered key-value store [10, 16] will be best suited for swarm like agents due to the distributed nature of a swarm. This is less relevant in heterogeneous swarms with something like hivemind or hierarchal control however still relevant. For homogeneous swarms this is ideal due to not wanting to have a static/temporary leader because of the issues like communication bottleneck, power loss near the leader due to flow of information to the leader, loss of a leader in a hybrid static swarm [1, 5]. Also, one of the fundamental philosophies of swarm robotics is the inherently parallel nature that they bring to the table. When creating leader-based algorithms they do not fit within the spirit of the design of homogeneous swarms, unlike a heterogeneous swarm might.

An area of study which is stagnated is the storage of data on local disks. How to keep either backup for disk failure and/or improve write performance onto disks, rather than just duplicating data like as described above. An example schema for this is RAID, which has different levels based on the type of attributes that you may need for your array of storage devices [**?**]. In terms of cloud-based storage RAID arrays are used commonly internally rather than externally to a different NAS or storage server. This is because we will have the guarantees of RAID for internal disk failures if they occur, for example, a server goes down we have the above replication schemes to be able to handle that loss of storage. Leading to RAID arrays across multiple nodes to be sort of redundant. However, by using a parity [8] based higher-level schema you can get space savings on the duplication of data. This is harder to implement though and most likely not needed due to servers that are lost due to power outages usually coming back online pretty soon. An example of this is might be a power issue causing a server to restart.

// TODO: Say what a correlated and non correlated failure is

2.2 Swarm robotics

Within research about swarms there is a split between practical soloutions of the agents and ue of agents as an algorithm, this split can be seen as swarm robotics and swarm intelligence. Swarm intelligence is where we use swarm like behavoiur to a problem, for example traveling salesman problem [5], this means that we use agent like code to compute a task. These tasks have usally been solved using a diffrent algorithm for example for the TSP using a genetic algorithm to solve, and the swarm algorithms like AS-TSP [5] are alternatives to that algorithm. These provide benefits and drawbacks to there counterparts, an area which these algorithms could excell and are researched into is the networking space due to the natrual parrellelism that can be expoited.

However this is not the route of research that will be needed for this project, rather we will be looking at swarm robotics. Swarm robotics has the same idea as swarm intelligence however it focuses on tasks of which are usally desiged to have an agent/agents complete, this is mainly designed for the practical space like moving objects or mapping an area, whether simulated or not. This swarms come in three types: hetrogenus, homogenues and a subcatergoriy of homogenus being hybrid swarms [1]. These three types can be mapped onto both the controls of the swarm and the agents body/abilitys.

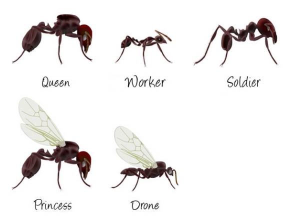

Figure 2.1: Example of a hetrogenus ant colony. https://www.pinterest.co.uk/pin/777363585651532845/

A hetrogenus swarm is where there are diffrences between the agents as in Figure 2.1, this is most commonly aquiring in nature and not usally studied into due to the diffrences in the agents, being a rarely needed property in research based problems. In real world soloutions hetrogenus soloutions can be of great use for example as desribed in [1] with a mother ship being a navy boat and a swarm of quadracopters. The reason for less research into this area id due to some key drawbacks of havin a hetrogenus swarm. Usaully you will have a hivemind like system if you have a hetrogenus swarm where you have leaders giving commands to subordinates or even one leader commanding the entire swarm. This is less desirable due to if there is a loss of those leaders you lose the ability to control the swarm, in are example this doesn't matter so much due to if you have loss of the mothership something has gone significantly wrong already. Due to the diffrences in the swarm agents it allows for greater efficency of the swarm for example having a robot that can mine and one that can farm, however the major loss of one type can lead to the loss of the colony. With swarms like these designs need to be taken so that agents can interchange between the tasks or that if agents fail there is no impact on the colony. Ants usally fit into this type of swarm where you have a queen, worker and major ants, however some ants like Leaf-Cutter Ants also have subcatgorey of workers like a fungus farmer. If there is a significant lose of workers, majors start doing tasks that normally workers would do [5], and with the farmer subcategory of the workers if a significant loss of them happens other workers can take over that job and learn how to do it, however that leads us more towards homgenus agents within a hetrogenus swarm.

A homogenus swarm is where you have each agent being the same, this is less often in nature and is more towards man-made agents, this is due to nature taking to the more efficent approach and due to having learning in the agents can adapt between roles [1, 5]. In homogenus swarms we get sugnificant redundancy due to if an agent goes down we have a swarms worth of replacements for that agent. With this redundancy we gain possible losses of effiecney due to either agents being to simple therefore losing specialism or each agent has parts for speclim but may never need a part. An example of this is a robot that has hands can farm and dig, however if we want them to be more effiecent we would have to give them both a hoe and pickaxe if we were using a homogenus model and if they couldn't switch between hoe and pickaxe during runtime of the swarm usage. In a homogenus style of swarm usally we will have homogenus control, this is where all agents decide what they want to do based of what other agents are doing and internal parameters, this can be equivilent to something like an emergernt swarm. This follows a distributed problem solving/commmunication design compared to leader based design like hivemind or structered/heirachical controlled swarms.

With swarm robotics everything gets a bit messy, usally there is no clear cut name or design that can be assigned to swarm models and behavouirs. This is where hybrid approaches come into play. An hybrid approach is mixtures between both hetrogenus and homogenus natures in both communications and agent design. In terms of communication usally when taking a hybrid approcach to a swarm usally you will have a swarm leader of leaders designated by a swarm of robots, this is handled with a concensus algorithm like paxos [**?**]. Also in hybrid models if we want to gain the efficency of hetrogenus models and the adaptabilty/reliabilty of homogenus swarms we can use something like tools. Humans themseleves are a great example of a hybrid based swarm. Though humans have variations in charateristics they can be seen as pretty homogenus in terms of the tasks that they perform, obvusily removing edge case actions that humans do. Tools and knowledge can be spread between humans to make the swarm more effiecient and an agent can specialize in a certain area however if some agents are lost other agents/humans can replace them by using the same tools and learning from the remaining agents of that task. Also the natural power based structure of humans fits a hybrid model in terms electorship of some kind, however the leaders aren't needed for every single action so fits into a usally heirachal power structure, compared to something of a hivemind model for swarm like agents.

3 Motiviation

The reason why I want to undertake this project of merging two/three research areas into one is because I beilieve that swarm robotics hasn't had much research into areas like this for storage of data on the swarm. The application of such a technolgy and algorithm is widly applicable to robots and cloud storage on a volitile network. There are lots of applications for this technology ranging from survailence, cooperative task completetion and the future of human space exploration. The push for an optimile distrubuted storage system seems to me to be a key for the future of man kind, being able to parrellise programs effecintly, this is espically true in quatum computers being able to handle parrellised algortihms efficently, obvisouly this is very simplified.

Examples of such applications that the merger of these two fileds bring together are mainly either just new algorithms for already handled cloud storage which is super needed but could be made more effiecnt with algoithms that are distributed, same sort of idea as block chain however that focuses more on security. Useful applications of swarms fit into useally the military, survalience, deleivery or exploration space. If talking future, future technologys you could be talking medical, terraforming however this is very vorbatium and specualrtive.

4 Methodology/Design

4.1 Design

5 Conclusion

A Some apendix

B Another apendix

Bibliography

[1] J. C. Barca and Y. A. Sekercioglu, “Swarm robotics reviewed,” Robotica, vol. 31, no. 3, pp. 345–359, 2013.

[2] V. Kumar and F. Sahin, "Cognitive maps in swarm robots for the mine detection application," SMC’03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483), Washington, DC, 2003, pp. 3364-3369 vol.4, doi: 10.1109/ICSMC.2003.1244409.

[3] H. Wang, D. Wang and S. Yang, “Triggered Memory-Based Swarm Optimization in Dynamic Environments,” in Applications of Evolutionary Computing, M. Giacobini, Ed. Berlin, Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2007, pp. 637–646.

[4] D. A. Lima and G. M. B. Oliveira, "A probabilistic cellular automata ant memory model for a swarm of foraging robots," 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, 2016, pp. 1-6, doi: 10.1109/ICARCV.2016.7838615.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems. Cary, NC, USA: Oxford University Press, 1999.

[6] L. Xiang, Y. Xu, J. Lui, Q. Chang, Y. Pan, and R. Li, ‘A Hybrid Approach to Failed Disk Recovery Using RAID-6 Codes: Algorithms and Performance Evaluation’, Association for Computing Machinery, vol. 7, p. 11, 2011

[7] C. Mims, ‘Why CPUs Aren’t Getting Any Faster’, MIT Technology Review, 2010. [Online]. Available: https://www.technologyreview.com/2010/10/12/199966/why-cpus-arent-getting-any-faster/. [Accessed: 01-Dec-2020].

[8] U. Troppens, W. Müller-Friedt, R. Wolafka, R. Erkens, and N. Haustein, ‘Appendix A: Proof of Calculation of the Parity Block of RAID 4 and 5’, in Storage Networks Explained: Basics and Applic-

ation of Fibre Channel SAN, NAS, ISCSI, InfiniBand and FCoE, U. Troppens, Ed. Chichester: Wiley United Kingdom, 2009, pp. 535–536.

[9] J. Liu and H. Shen, "A Low-Cost Multi-failure Resilient Replication Scheme for High Data Availability in Cloud Storage," 2016 IEEE 23rd International Conference on High Performance Computing (HiPC), Hyderabad, 2016, pp. 242-251, doi: 10.1109/HiPC.2016.036.

[10] N. Bonvin, T. G. Papaioannou, and K. Aberer, A Self-Organized, Fault-Tolerant and Scalable Replication Scheme for Cloud Storage. New York, NY, USA: Association of Computing Machinery, 2010.

[11] Legal Services Act. 2007.

[12] A. Prahlad, M. S. Muller, R. Kottomtharayil, S. Kavuri, P. Gokhale, and M. Vijayan, ‘Cloud gateway system for managing data storage to cloud storage sites’, 20100333116A1, 2010.

[13] B. Czejdo, K. Messa, T. Morzy, M. Morzy, and J. Czejdo, ‘Data Warehouses with Dynamically Changing Schemas and Data Sources’, in Proceedings of the 3rd International Economic Congress, Opportunieties of Change, Sopot, Poland, 2003, p. 10.

[14] ‘Key-Value Scores Explained’, HazelCast. [Online]. Available: https://hazelcast.com/glossary/key-value-store/. [Accessed: 02-Dec-2020].

[15] L. Lamport, ‘The Part-Time Parliament’, in Concurrency: The Works of Leslie Lamport, New York, NY, USA: Association of Computing Machinery, 2019, pp. 277–317.

[16] D. Agrawal and A. E. Abbadi. The tree quorum protocol: An efficient approach for managing replicated data. In VLDB’90: Proc. of the 16th International Conference on Very Large Data Bases, pages 243–254, Brisbane, Queensland,Australia, 1990.

[17] S. Lynn, ‘RAID Levels Explained’, PC Mag, 2014. [Online]. Available: https://uk.pcmag.com/storage/7917/raid-levels-explained. [Accessed: 06-Dec-2020].

[18] J. Hu et al., Eds., HiveMind: A Scalable and Serverless Coordination Control Platform for UAV Swarms. ArXiv, 2020.

[19] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review. Cham, Switzerland: Springer Interna-

tional Publishing, 2018.