A Service Replic Distribution Scheme Based on Group Structure

Yang Wangli [1] Wang Huiying[2]

（School of Computer Science Engineering Daqing Petroleum Institute Daqing, Heilongjiang 163318, China ywl008@126.com）

（Thermal Power Plant of Electric Power Daqing Petroleum Administratrive Bureau Daqing, Heilongjiang 163314, China）

Abstract

How to effectively distribute service replicas to its *n* nodes is of much importance in distributed and parallel systems. In many cases, the service replicas are randomly placed, and some service replicas have no relation to any nodes in which they are. So these service replicas are very difficultly accessed, and the flexibility of the system is poor. This paper presents such a novel service replicas distribution scheme. Instead of traditional method, this scheme is mainly based on the group structure that makes every node relate to the characteristics of the service replicas. The proposed scheme is very flexible to change the structures of the service applications. So the scheme can be employed to many scenarios where service replicas are needed.

1. Introduction

The Internet has experienced an explosive growth over the last decade. Making information available to a rapidly growing user population with a high service quality is quickly becoming a very important and challenging problem. Wide-area distributed systems often replicate entities in order to improve reliability, access latency, or availability[1,2]. But most replica services only provide Content replica. These services are mostly passive and they only wait for the requests of clients. Based on the mechanism, the applications of clients must account for some problems such as replica access latency, replica location and transparency. Recently we have seen some proposals giving solutions to these problems.

The rest of the paper is organized as follows. Section Ⅱ gives preliminary concepts and group basics. Section Ⅲ presents the new service replicas distributing scheme based on group structure. Section Ⅳ presents some previous schemes and applications. Section Ⅴ denotes a simulation to compare our scheme with a traditional one. Section Ⅵ concludes the paper and present future work.

2. Preliminary

In this section, some related concepts of semigroup and group [1] are given:

Definition 1 (Semigroup): Given a non-empty set S on which a binary operation $S \otimes S \to S$ is defined and "$\otimes$" is a mapping:

$$\forall a,b \in S; a \otimes b = c \in S$$

The cardinal number $|S|$ is called the order of the semigroup *S*. we can write $(S,\otimes)$ simply as *S*.

Definition 2 (Subsemigroup). If $(S,\otimes)$ is a semigroup, then a non-empty subset *A* of *S* is called a subsemigroup of *S* if it is closed with respect to multiplication:

$$\forall a,b \in A, a \otimes b = c \in A$$

Definition 3 (Group). If $(S,\otimes)$ is a semigroup, then $(S,\otimes)$ is called a group if it has the following properties:

$$\exists e \in S, \forall a \in S, a \otimes b = e \otimes a = a$$

$$\forall a \in S, \exists a^{-1} \in S, a \otimes a^{-1} = a^{-1} \otimes a = e$$

Espcially, e is called as the identity element of the group *S*.

Definition 4 (Subgroup). If $(S,\otimes)$ is a group, then a non-empty subset *A* of *S* is called a subgroup of *S* if it is closed with respect to multiplication:

$$\forall a,b \in A, a\otimes b = c \in A$$

Definition 5 (Generating Set): If $\{U_i \mid i\in I\}$ is a non-empty family of subgroups of a group *S*, then it is easy to see that $\cap\{U_i \mid i \in I\}$ is either empty or is itself a subgroup of *S*. If *A* is an arbitrary non-empty subset of *S*, then the family of subgroups of *S* containing *A* is non-empty. Hence the intersection of the family is a subgroup of *S* containing *A*. We denote it by $\langle A\rangle$, the semigroup $\langle A\rangle$ consists of all elements of *S* that can be expressed as finite products of elements in *A*. If $\langle A\rangle = S$ we shall say that *A* is set of generators for *S* or a generating set of *S.*

Definition 6 (Monogenic group). If *A* is a finite set $\{a^1, a^2, a^3, \ldots, a^n\}$, we shall write $\langle A\rangle$ as $\langle a^1, a^2, a^3, \ldots, a^n\rangle$. Especially it is the case where $A=\{a\}$, when $\langle a\rangle=\langle a^1, a^2, a^3, \ldots, a^n\rangle$. We refer to $\langle a\rangle$ as the monogenic subgroup of *S* generated by the element a. The order of *a* is defined as the order of the subgroup $\langle a\rangle$. If a group *S* has the property that *S* $=\langle a\rangle$ for some *a* in *S*, we say that *S* is a monogenic group. *a* is called by the generator of the monogeric group *A*.

Definition 7 (Group Homomorphism Mapping). If $f: S \to T$ is a mapping from a group $(S,\otimes)$ into a group $(T,\oplus)$, we say that *f* is a homomorphism if it has the following properties:

$$\forall a,b \in S, f(a\otimes b) = f(a)\oplus f(b)$$

Definition 8 (Coset):If *H* is a subgroup of a group *S* and $a\in S, Ha=\{ha \mid h\in H\}$, we say that $Ha=\{ha \mid h\in H\}$ is a Coset of *S* and *a* is the representative of Ha .

Definition 9 (Normal subgroup): Given *H* is a subset of *S*, A is called a normal group of *S* if it has the following property:

$$\forall a \in S, Ha = aH$$

Definition 10(Abelian group): Given $(A,\otimes)$ is a group, it is called *a* Abelian group if it has the following property:

$$\forall a_i, a_j \in A; a_i \otimes a_j = a_j \otimes a_i \in A$$

Lemma 1. The subgroup of Abelian group is Normal subgroup.

3. Previous schemes and applications

In this section, we introduce some previous schemes and application based on group. It is importance of multicast communication in distributed and parallel systems. Recently, the technique has been employed in many fields[2-5]:

First, some applications employ Active Replication, Passive Replication or multi-version to copy the important services as multi-replica and distribute to the different nodes in distributed and parallel system. When a client sends a request to an important service of the system, it actually sends a message to the group of replica. Because each of the group can provide the same service, if a node breakdowns then it will not reflect other nodes. Only one node running, the requests of clients can be replied.

Second, CSCW has been employed in many business fields. These applications always are consisted of many nodes and the nodes must correspond with each other. For many partions of the nodes, multicast communication must be employed to serve for these partions.

Third, distributed objects can be located in a distributed and parallel system. For example, clients probably search a file in a distributed operation system V-Kernel, and they broadcast a message to all servers for it. All servers receive the message, but only file servers need to reply. It is easy for clients to find a object in a system which can provide multicast communication service.

Fourth, multicast communication can be employed to balance load in a distributed system. When a server overruns, a distributed system should select other servers to replace it or subtract some load to them. Multicast communication is import to select a favorable subgroup.

But many problems are found in the schemes and applications:

Firstly, to manage the relation of the service replica in a distributed system is difficult. When the scale of a system has been changed, each node of it has difficult in adapting itself to the system.

Secondly, Multicast communication has difficult in select a favorable subgroup that will accept a message multicasted. And broadcast will waste much brandwidth and reflect those nodes that will not reply specific requests.

Thirdly, failure detection brings on a series of faults. When a node recovers from a system breakdown, it will probably "forget" all services that it shall provide. That is to say it is "Lost Memory". But in most cases, they will not adapt themselves to the system and search those service distributed to them. For infinite search the service belonged to them, they also will go into an infinite cycle.

As presented above, we know that these restrictions have narrowed the application areas base on group. We must present a new scheme to solve these problems.

4. The prosed scheme

The proposed service replica distribution scheme is based on the group Theorem.

Let S be $\{s_0, s_1, s_2 \ldots s_m\}$, s_i is a kind of service in a distributed and parallel system that has m nodes. Each s_i has some replica to serve for our system exterior request. The service replica distribution scheme is as follows:

Every s_i has some replica distributed in our system. s_i is defined by a ternary group s_i :($sname_i$, sid_i , srn_i). sname_i is the name of. s_i , sid_i is the indentity of s_i and srn_i is the number of the replicas which s_i has in our system. And every s_i has a set of nodes A_i.

Let A_i is a set of nodes $\{a_1, a_2 \ldots a_{n,}$ $\varnothing\}$ and a_i is the identity of one node. $\varnothing$ is the null that any service can be provide. We can access to the replica of s_i in the set A_i if we have an identity a_i. And these nodes work independently, but a_i is relative to each other. Especially in our scheme, a_1 is equal to the sid_i of the s_i of which it have replica. And the srn_i saves $|A|$.

In following section, we simplify A_i as A.

Definition 12 (Selection Operation): "$\odot$" is a binary operation on A and the operation is defined as:

(1) $a_i \odot a_j = a_{(i+j)}$

if $0 \leqslant i,j \leqslant |A|$ and $0 \leqslant i+j < |A|$;

$a_i \odot a_j = \varnothing$

if $0 \leqslant i,j \leqslant |A|$ and $i+j \geqslant |A|$;

(2) $a_i \odot \varnothing = \varnothing \odot a_i = a_i$

(3) $\varnothing \odot \varnothing = \varnothing$

We call the operation "$\odot$" by selection operation. The set A on which a binary operation $A \odot A \to A$ and the mapping "$\odot$" are consist of a group. The proof is following as:

(1) A is a non-empty set. If n=0 , $A=\{\varnothing\}$.

(2) $a_i \odot a_j = a_{(i+j)} \ni A$

if $0 \leqslant i,j \leqslant |A|$ and $0 \leqslant i+j < |A|$;

$a_i \odot a_j = \varnothing \ni A$

if $0 \leqslant i,j \leqslant |A|$ and $i+j \geqslant |A|$;

So $(A, \odot)$ is a semigroup.

(3) $a_i \odot \varnothing = \varnothing \odot a_i = a_i$ and $\varnothing \odot \varnothing = \varnothing$, so $\varnothing$ is the identity element e.

(4) $a_i \odot a_j = \varnothing$ if $0 \leqslant i,j \leqslant |A|$ and $i+j \geqslant |A|$.

And $\varnothing \odot \varnothing = \varnothing$, so every element a of *A* has a converse element.

So $(A, \odot)$ is a group.

(5) Every element can be attained from a_1:

$$a_i = a_1 \prod_{1}^{i-1} a_1 \quad \text{if } i<|A|\,;$$

$$\text{And } \varnothing = a_1 \prod_{1}^{i-1} a_1 \quad \text{if } i=|A|.$$

So $(A, \odot)$ is a monogenic group. And a_1 is the generator of the monogenic group.

According to the definitions, we have prove that $(A, \odot)$ is a monogenic group. The group *A* provides a kind of service to request. And its subgroups and every its element can provide the same service. So the scheme based on group structure can solve the problems presented above. The rest of the section discusses of solving the problems presented above.

A. Service Access Probablity

When a distributed and parallel system based on group runs well, all requests of clients will be averagely distributed to those nodes. If a node breakdown, those clients who send requests to it will generally repeat to send requests to it until overtime. Then those clients don't receive responses and notify the system what has happened. The system perhaps provide anther node to clients and clients will repeat to send requests.

Instead of traditional schemes, our scheme is based on group and the nodes received these requests sent to the group in equal probability. So a request for a kind of service is sent to a node, and the next request is not always sent to the same node. Given a node breakdowns, and it doesn't reply the request. Then, the next request is sent for the same kind of service and it will probably be sent to anther node instead of being continuously sent to the same node. Given k requests will be sent for the same kind of service from clients. For example, the service access probability is $1-(\frac{1}{n})^k$ when one node breakdowns. There, $(\frac{1}{n})^k$ is the probability that all requests are unfortunately sent to the fault node.

According to our scheme, the requests for a kind of service will not always sent to a node that has break downed. And the reply will be achieved before being conscious of the error. So the latency caused by repeated requests decreased. A stimulation of the scheme will be given in the section Ⅴ.

B. Recovery of "Lost Memory"

When one node recovers from a system breakdown, it will not find these services that it should provide to the distributed and parallel system. It is called "Lost Memory" as present above. In our scheme, the problem is solved by the properties of the monogenic group.

The identity a_i of every node is fixed and it will not be lost. When a node recovers from a system breakdown, it only has the identity a_i of itself. Then it requests all s_i from our system. If it has received a s_i, the node gets a ternary group s_i:($sname_i$, sid_i, srn_i), and the sid_i saves the generator of the set that provide the service s_i. The node a_i checks whether the sid_i is the generator of the group that it belongs to. If the sid_i is its generator, it starts up the service. It actively gets the service replica from our system if it does not have the service replica.

We only need to prove that every node can always be or not be generated by a s_i:($sname_i$, sid_i, srn_i) in finite steps. The proof is following as:

sid_i is the generator of the set which includes the node $a_i.srn_i$ is equal to the order $|A|$ of the node set. And each element of *A* has the properties:

$$a_i = a_1 \prod_{1}^{i-1} a_1 \quad \text{if } i<|A|\,;$$

$\varnothing = a_1 \prod_1^{i-1} a_1$ if i=|A|.

If a_i belongs to A, then there must be i(i<|A|) and $a_i = a_1 \prod_1^{i-1} a_1$ If a_i does not belong to A, then $\varnothing = a_1 \prod_1^{i-1} a_1$ when i=|A|. However, the calculation for a_i will not go beyond the limits of srn_i. So a_i can start up these services in finite steps.

C. Load Balancing Based on Subgroup

Group structure has some characteristics to classify the elements of groups. In our scheme, we employ coset to realize load balancing. Normal subgroup is a set of elements that have the same characteristics. If a subgroup of nodes can consist of a normal subgroup, we can distribute load to every node of the set and balance load. When the load of a node overruns its limitation, our system can classify the nodes that provide the same service to balance load by normal subgroup. We select the node whose load overrun as the representative of the normal subgroup. Our system can dynamically generate and balance load, because every kind of service has its set of replica and we can select some nodes including the overrunning node to generate a subgroup to share in the load.

Given H is a subgroup of A, and if a Ha_i or a_iH is also a subgroup of A, then we can distribute the load of a_i to the subgroup H. The proof is following as:

There, A is a group, a is randomly selected from A and H is a subgroup of A. Because the load is distributed from a specific node a, H includes a. We must prove that Ha and aH is the subgroups of A.

Firstly, $\forall a_i, a_j \in A$ $a_i \odot a_j = a_{(i+j)} = a_j \odot a_i \ni A$, so A is a Abelian group;

Secondly, according to Lemma 1, H is a normal group and $Ha=aH$;

Thirdly, if $a \ni H$ and H is a group, then $a^{-1} \ni H$.

$aH \subseteq H \otimes H = H$;

$H = (a \otimes a^{-1}) \otimes H = a \otimes (a^{-1} \otimes H) \subseteq aH$;

So $Ha = aH = H$

So Ha and aH are subgroups of A. That is to say the system can distribute load to a subgroup.

5. Simulation results

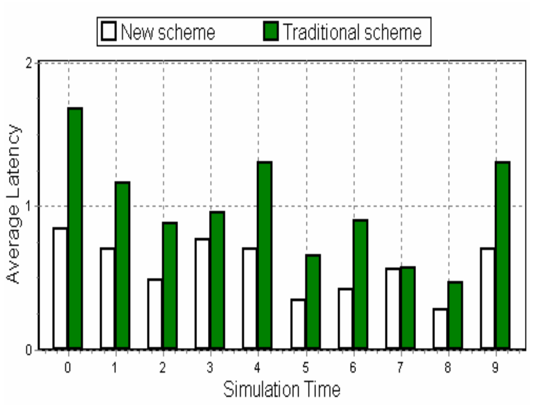

Fig. 1 Average latency of two schemes.

In order to evaluate the scheme, we developed a simulator based on it. This simulator provides a modular simulation framework through which we can compare our scheme with these traditional methods. In the simulator, we assumed a group, which provide services according to our scheme. According to the traditional methods [6], if a request is not be replied, then the next two requests will be sent to the same node. The node will be regarded as a node break downed if the three requests are not replied. And clients will request to the next nodes. There are 30 nodes in the simulator for the two schemes. Within these models, some nodes that are randomly selected will breakdown to evaluate them.

Figure 1 shows the average response time of them. Every pair of data shows the comparation of the two schemes that has the same number of the nodes that has broken down. We plot the simulation time on X-axis and the average response time on Y-axis.

According to the different requests and the number of fault nodes, the average latency is different. As we see above, the average latency of our scheme is shorter than that of the traditional scheme. Because in the traditional scheme, some indexed sequential nodes break down, and clients repeat to request to them and the latency is greatly increase. Within 0th and 9th pairs of data, the latency of ours is almost 1/2 shorter than that of the traditional scheme. Within 7th pair of data, the latency of them is the same without fault nodes. As we can see, the performance of our scheme is higher than the traditional especially in indexed sequential nodes.

6. Conclusions

Group is a great field of algebra. It has some promising properties to improve the performance of service replicas distribution schemes. In the paper, we address some problems in traditional schemes and present a novel scheme based on group structure. And it can be used to improve service access time, service availability and so on.

Our simulation results are very promising, and showed that the performce of the novel scheme is greatly improved. We note that our scheme has significant advantages. First, it can improve serivce access probablity, and second it can recover from "Lost Memory". At last, it can generate a normal group to balance load intead of the entire system.

7.References

[1] Aigner M, *Combinational Theory*, *M*, New York, *Springer-Verlag*, 1979, pp. 134-145.

[2] Hu Liang, "A dynamic load balancing system based on data migration", *M*. Computer Supported Cooperative Work in Design, The 8th International Conference, 2004,5 , pp. 26-28.

[3] Legrand, "Mapping and load-balancing iterative computations", *J. Parallel and Distributed Systems*, 2004,6, pp. 546-558.

[4] Tanenbaum, A.S, *Distributed Operating System*, *M*. Prentice Hall, 1995, pp. 212-231.

[5] Youn. C, Chung L, "An efficient load balancing algorithm for cluster system",*C*. Proceeding of Network and Parallel Computing - IFIP International Conference, 2005, pp. 176-179.

[6] Porter. G, Katz. H, "Effective web service load balancing through statistical monitoring," *J. Communications of the ACM*, vol. 4, no. 3, pp. 49-54, 2006.

PROFILE

Yang Wangli. An assistanted professor of Daqing Petroleum Institute. Her research interest is database application.