International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

An Approach to Adaptive Load Balancing for RFID Middlewares

Heung Seok Chae, and Jaegeol Park

***Abstract*—** Recently, there have been an increasing interest in RFID system and RFID systems have been applied to various applications. Load balancing is a fundamental technique for providing scalability of systems by moving workload from overloaded nodes to under-loaded nodes.

This paper presents an approach to adaptive load balancing for RFID middlewares. Workloads of RFID middlewares can have a considerable variation according to the location of the connected RFID readers and can abruptly change at a particular instance. The proposed approach considers those characteristics of RFID middlewares to provide an efficient load balancing.

***Keywords*—** RFID middleware, Adaptive load balancing.

I. Introduction

In recent years, RFID technology has again attracted great attention due to technology advancements, heightened security concerns and a competitive business environment with emphasis on cost control and affordable RFID tag costs [1]. By identifying and tracking each product using RFID technology, it is possible to collect accurate, immediate information about the location and the history of products. RFID systems have been applied to various applications such as retail, healthcare, logistics, automotive, food industry, etc. Wal-Mart, Tesco, Metro together with the Department of Defense are even demanding the implementation at pallet level for warehouse and logistics operations.

Fig. 1 shows a schematic view of an RFID system. In a typical RFID system, RFID tags, are attached to objects, which will identify themselves when detecting a signal from a RFID reader by emitting a radio frequency transmission. An RFID middleware, located between readers and RFID applications, manages the flow of information from a multitude of readers to RFID applications. An RFID middleware collects tag data from the connected readers and passes on only relevant information to the RFID applications. RFID application performs some appropriate business logic by using the collected information from the RFID middleware.

As one of the essential features of distributed systems, the issue of load balancing should be addressed to increase in availability and scalability. Load balancing is a well-established technique to improve the utilization efficiency of computing resources by distributing the load dynamically according to load distribution strategies. A collection of servers, so called a cluster, is used as a pool of replicated resources to process concurrent services. The incoming requests can be distributed to the servers according to specific load distribution strategies and thus the requests can be processed quickly. In other words, the load on an overloaded server should be transferred to an under-loaded server to enhance the system throughput. Thus, the whole system resources can get full utilization.

This work was supported in part by the Brain Korea 21 Project in 2006 and the Regional Research Centers Program(Research Center for Logistics Information Technology), granted by the Korean Ministry of Education & Human Resources Development.

Heung Seok Chae and Jaegeol Park are with the Department of Computer Science and Engineering, Pusan National University, 30, Jangjeon-dong, Geumjeong-gu, Busan 609-735, Korea (e-mail:{hschae,jgpark} @pusan.ac.kr).

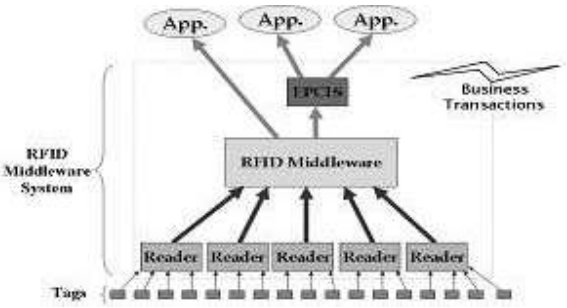

Fig. 1 An overview of RFID system

As in the case of conventional distributed systems, load balancing is an important feature which should be supported in RFID middlewares [2]. In clustering approach, each RFID middleware works together to process a considerable amount of information from many readers. Each RFID middleware receives EPC information from the readers connected to itself. The information quantity depends on the number of readers connected to it and the number of tags collected from those readers. If a RFID middleware has many connected readers or collects many tags from them, it may result in the RFID middleware response delayed or even broken down. Since some other RFID middlewares may keep idle or have enough available to process the workload resulting from extra readers, it is desirable to resolve this situation by redistributing the readers connected to the overloaded middleware to other under-loaded middlewares.

For example, consider a situation illustrated in Fig. 2, where six readers are connected to RFID middleware MW_1, and only two to MW_2. Assuming each of the readers transmit the same amount of EPC data to the middleware, each of two middlewares can not be used evenly. For better system resource utilization reason and finally for more scalability of a system, it is recommended to reallocate two of MW_1's readers to MW_2. This paper propose an adaptive load balancing strategy appropriate for RFID middlewares.

The rest of the paper is organized as follows: Section 2

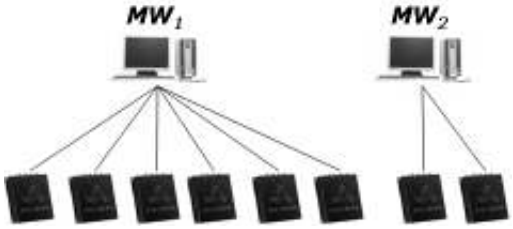

Fig. 2 RFID middlewares with the connected RFID readers

briefly describe the importance of load balancing in RFID middlewares. Section 3 presents an adaptive approach to load balancing for RFID systems. The related works and conclusion are given in Section 4 and 5, respectively.

II. Load Balancing in RFID Middlewares

A. Overview of RFID Systems

The EPC Network, originally developed by the Auto-ID Center [3] with its standards now managed by EPCglobal [4], was designed and implemented to enable all objects in the world to be linked via the Internet. Figure 3 shows the high-level components of the EPC Network Architecture.

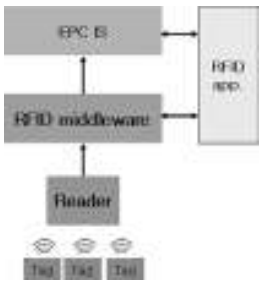

Fig. 3 A data-flow view of EPC Network Architecture

- Tags transmit EPC data using radio frequency. The Electronic Product Code (EPC) is a unique number that identifies a specific item. The EPC is stored on a RFID tag. Once the EPC is retrieved from the tag, it can be associated with dynamic data such as its location of origin and the date of its production via EPC IS.
- Readers are devices responsible for detecting when tags enter their read range. They are used to extract the EPCs from the tags. Along with tags, readers enable the automated identification of tagged objects. The read range of a tag depends on the power of the reader and the frequency the reader and tag use to communicate.
- RFID middleware(formerly called Savant) is a middleware software designed to process the streams of tag coming from one or more reader devices. RFID middleware performs filtering, aggregation, and counting of tag data, reducing the volume of data prior to sending to RFID Applications. These functions are required in order to handle the extremely large quantities of data that RFID systems can generate through the continuous interrogation of tags.
- The EPC Information Service makes EPC Network related data available in XML format to requesting services. Data available through the EPC Information Service may include tag read data collected from middleware, instance-level data such as date of manufacture, expiry date, and object class-level data such as product catalog information.
- RFID Application. Due to the automatic identification and collection of information, an RFID system can be used for various applications such as retail, healthcare, logistics, automotive, food industry, etc. These RFID applications obtain the necessary information from RFID middleware(tag read event) and EPC IS(its associated item data).

This paper focuses on the load balancing of RFID middlewares. The workload distribution of RFID systems can be very irregular with respect to both the location of the connected readers and the time.

Except mobile readers (i.e., handheld RFID readers), most of RFID readers are fixed at a particular location. Since reader can sense all the tags around itself, the workload of the reader depends on the number of tags around the reader. When a reader is installed in position where many RFID tags exist such as shelf in warehouse, the reader senses a lot of tags and then transmit them to RFID middleware. On the contrary, a reader installed at a gate in warehouse can sense only the tags which are moved in and out, the reader generally senses a small number of tags, which has less impact on the load of the connected RFID middleware.

However, the workload of an RFID middleware can abruptly change at a particular time. For example, when a couple of trucks with many products pass the gate, the reader instantaneously senses a huge number of tags, which accordingly can cause the workload of the RFID middleware to increase substantially.

Therefore, adaptive load balancing approach is more desirable in RFID middlewares. Adaptive load balancing strategies use run-time system state information (e.g., host workload), to make load balancing decisions, whereas non-adaptive load balancing strategies do not. In other words, adaptive load balancing strategies make changes to the distribution of work among hosts at run-time; they use current or recent load information when making distribution decisions. As a result, adaptive load balancing strategies can provide a significant improvement in performance over non-adaptive strategies.

B. Load Balancing Policies for RFID Middlewares

To address load balancing, a suite of polices are generally considered including the strategies of load information gathering, the conditions for activating load balancing operations, and the metrics for server selection and job reallocation [5].

- Information gathering policy specifies the strategy for the collection of load information including the frequency and method of information gathering. There is a tradeoff between having accurate information and minimizing the

overhead. It also includes the estimation and specification of workload including processor load, length of queue, storage utility [6]. The number of the readers connected to the RFID middleware or the number of tags read from those readers are candidate of workload in RFID middlewares.

- Initiation policy determines who initiates the process of load balancing. The initiator can be the source node, the destination node, or both (symmetric initiations). Load balancing can be initiated by each overloaded middleware (in decentralized approach) [7], [8], [9] and a central load balancer which distribute the workload among groups of middlewares (in centralized approach) [7].
- Job transfer policy decides when the initiator should consider reallocate some requests to other node. The decision can be made based on only local state or by exchanging global processor load information. The number of tags from RFID readers connected to an RFID middleware can be a good choice as criteria. Once workload exceeds the capability of the RFID middleware, we begin to distribute load of the middleware to others.
- Selection policy determines which particular job to reallocate. We need to maintain the load information of each reader connected to an RFID middleware. Once the load of an RFID middleware exceeds its allowed capability, we should decide an optimal set of RFID readers such that the reallocation of them leads to better performance with a minimum overhead due to the reader reallocation.
- Location policy determines to which node the jobs should be reallocated. To reduce load of overloaded RFID middleware, we need to reallocate the readers chosen according to the selection policy to other appropriate RFID middlewares. The location policy is concerned with the destination middleware to which the selected readers are reallocated. Naturally, the most under-loaded RFID middleware is a good candidate as the destination.

Among the policies mentioned above, this paper focuses on job transfer policy, selection policy, and location policy. That is, we propose an load balancing approach to when to start load balancing, which readers to be reallocated to which RFID middlewares.

III. The Load Balancing Strategy in RFID Middlewares

A. Basic Idea of the Approach

Most distributed system load sharing research focuses on improving some performance metrics such as mean response time where transfer overhead is assumed to be negligible. However, transfer overhead can seriously degrade performance of the distributed system. In the case of RFID middlewares, a set of chosen readers should be disconnected to the source middleware and then reconnected to the target middleware during reader reallocation. We should try to minimize reader reallocation overhead because the reallocation of readers surely incurs the cost of disconnection and reconnection, and there is a possibility of losing tags information collected by the reallocated readers. Therefore, we need to focus on minimizing overhead involved in reader reallocation.

Specifically speaking, reader reallocation overhead depends on two factors: (1) the number of reallocated readers and (2) the frequency of reader reallocation. When we reallocate more readers, there will be more time to disconnect them from the source middleware and reconnect them to the destination middleware. In addition, the more number of readers are reallocated, the more number of tags could be lost during reader reallocation. In addition to the overhead of actual reader reallocation, scheduling for reader reallocation can degrade performance. In other words, the number of times the scheduler is called to make load sharing decisions can be a factor that degrades performance. Therefore, we need to reduce the frequency of reader allocation.

B. Basic Definitions

Definition 1: For an RFID middleware systems, three sets M, CR_k and R are defined as follows:

- $M = \{m_1, m_2, \ldots, m_n\}$. M denotes the set of n RFID middlewares, named m_1, m_2, ..., m_n.
- $CR_k = \{r_1^k, r_2^k, \ldots, r_l^k\}$, $1 \leq k \leq n$. CR_k denotes a set of RFID readers that are connected to RFID middleware m_k.
- $R = \bigcup_{1 \leq k \leq n} CR_k$. R denotes the set of all the readers which are equivalent to the union of the readers of each middleware. Generally, each reader is connected to only one middleware. Thus, $CR_i \cap CR_j = \emptyset$, where $1 \leq i, j \leq n$ and $i \neq j$.

Figure 4 shows the basic definitions. There are n middlewares, named m_1, m_2, ...E, m_n. Middleware m_k has a set of the connected readers, named r_1^k, r_2^k, ..., r_l^k, which is denoted by CR_k.

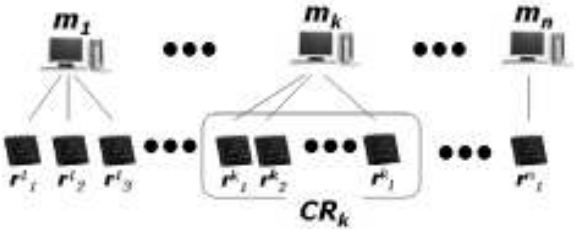

Fig. 4 Basic definitions

Definition 2: Concerning the workload of each reader and middleware, the followings are defined:

- $WL_R(r)$: the workload of reader r. *The workload of a reader will be imposed on the middleware that the reader is connected to. The number of the tags which the reader processes would be a proper choice for the workload of a reader.*

- $WL_M(m_i)$: The workload of middleware m_i. *Assuming that the RFID middleware is dedicated to processing its connected readers, the workload of a middleware can be computed by summing up the workload of its connected*

readers. That is, $WL_M(m_i) = \sum_{r \in CR_i} WL_R(r)$.

- $WL_M^U(m)$ and $WL_M^L(m)$: the maximum and the minimum workload of middleware m which are allowed. *They are used as threshold for load balancing strategy. That is, a load balancer should try to maintain the workload of a middleware* m *between* $WL_M^L(m)$ *and* $WL_M^U(m)$. *The load balancer in addition starts load balancing on the basis of* $WL_M^U(m)$.

- $WL_M^M(m)$: the average workload of middleware m. *Simply, the average workload of a middleware is defined to be the arithmetic mean of the maximum and the minimum workloads of the middleware. That is,* $WL_M^M(m) = \frac{WL_M^U(m)+WL_M^L(m)}{2}$.

C. Load Balancing Strategy

Based on the definition mentioned above, load balancing strategy for RFID middlewares are presented. Fig. 5 shows an algorithm of load balancing. The load balancing strategy for RFID middlewares is explained by each load balancing policy.

- Job transfer policy: RFID middleware should determine whether load balancing is needed on the basis of its workload. Our policy is that load balancing get started when its workload exceeds the maximally allowed workload, i.e., WL_M^U. It also may be applicable to initiate load balancing when its workload goes beyond the mean, not the maximum. However, the use of WL_M^M instead of WL_M^U can lead to more frequent load balancing. Therefore, WL_M^M is adopted as a threshold workload in order to reduce the load balancing overhead.
  Module $Get\_Most\_Overloaded\_Middleware()$ in Fig. 5 determines the most overloaded middleware by computing the difference between the current workload and the maximum workload of a middleware. There may be two or more overloaded middlewares at the same time. In this case, the most overloaded middleware is first tried. After the readers of the middleware are reallocated, load balancing will be applied to the next middleware. Module $Load\_Balancer$ continuously performs load balancing while there is any overloaded middleware.

- Selection Policy: Once we determine a particular middleware to be load balanced, the next step is to choose a proper set of readers to be transferred. To minimize the overhead of reallocation, readers are considered in decreasing order of its workload. That is, the reader with the most workload is first reallocated to other under-loaded middleware. The sequence S in module $Reallocate\_Readers()$ is the sequence of the readers that are connected to the source middleware in the decreasing order of their workloads.
  After transferring the most loaded reader to the destination middleware, the next reader should be transferred if the source middleware is still overloaded. We repeatedly perform this reallocation to the other connected readers in the decreasing order of their workload, which is suggested by the for loop **for** $i = 1$ to $i \leq p$ in module $Reallocate\_Readers()$.
  The reallocation of readers will be repeated until the resulting workload of the middleware goes below the average workload specified, which is expressed by **until** $WL_M(m_k) < WL_M^M(m_k)$. As a load balancing termination criterion we select the mean workload instead of the maximum workload. If the maximum workload is used, then the workload may be still high after load balancing, which can lead to a frequent load balancing.

- Location Policy: The least loaded middleware is chosen as the target to which the selected readers are reallocated. This is to prevent the situation where the target middleware itself can be overloaded right after load balancing. In addition, by moving workload to the lighter middleware, we can decrease the frequency of load balancing.

```
// get the most overloaded middleware
module Get_Most_Overloaded_Middleware() returns m_k begin
  find k such that
    WL_M(m_k) > WL_M^U(m_k) and
    ∀i(≠ k) · [WL_M(m_k) − WL_M^U(m_k) ≥ WL_M(m_i) − WL_M^U(m_i)]
end module

module Reallocate_Readers(m_k) begin
  // sort CR_k in decreasing order of WL_R(r), r ∈ CR_k
  S =< r_1, r_2, ..., r_p >, where p =| CR_k | such that
    r_i ∈ CR_k and WL_R(r_i) ≥ WL_R(r_{i+1}) where 1 ≤ i < p
 do
   for i = 1 to i ≤ p
    // choose the leat loaded middleware for the target
    find m_t such that
      WL_M(m_t) + WL_R(r_i) < WL_M^U(m_t) and
      ∀l(≠ t) · [WL_M(m_t) + WL_R(r_i) < WL_M(m_l) + WL_R(r_i)]

    // reallocate r_i from m_k to m_t
    CR_k = CR_k − {r_i}
    CR_t = CR_t ∪ {r_i}

    // update the workload after reallocation
    WL_M(m_k) = WL_M(m_k) − WL_R(r_i)
    WL_M(m_t) = WL_M(m_t) + WL_R(r_i)
   end for
 until WL_M(m_k) < WL_M^M(m_k)
end module

module Load_Balancer() begin
 do
  // find the most overloaded middleware
   m = Get_Most_Overloaded_Middleware()
   if no such overloaded middleware break

  // Perform reader reallocation
   Reallocate_Readers(m)
 while true
end module
```

Fig. 5 Load balancing strategy for RFID middlewares

IV. Related Works

Load balancing is a well-established technique for utilizing available computing resources more effectively in distributed systems. By overcoming the limitation to the capability of a single system, a set of homogenous or heterogenous computing resources are grouped into a cluster. In addition, with grid computing, an individual can unite pools of servers, storage systems and networks into a large system. End users and applications see this environment as a big virtual computing system.

Load balancing issues is fundamental for any type of distributed systems and accordingly there is several works to provide load balancing appropriate for the type of distributed systems. Payli et al. proposed a dynamic load balancing approach to provide application level load balancing for individual parallel jobs in grid computing environment [10]. In network of workstations environment, there is also work concerning load balancing [11]. Agent-based approaches have been tried to provide load balancing [12], [13]. Although there are numerous works on load balancing strategies, the existing strategies have some assumption to provide an optimized load balancing. However, the assumption may not be appropriate for RFID middlewares. In this paper, we have proposed a load balancing approach by considering the characteristics of RFID middlewares.

V. Conclusion and Future Work

In contrast to other conventional distributed systems, RFID middlewares have their own characteristics which should be considered and exploited for more efficient load balancing. We note that workload of RFID middlewares depends mainly on the workload of the RFID readers that are connected to itself. And, since the workload of each reader can be considerably variable, load balancing strategy which can be efficiently adapted to the variation of readers' workloads. By considering those characteristics, we have proposed a load balancing strategy for RFID middlewares, especially focusing on job transfer policy, selection policy, and destination policy.

In the future work, we'll implement the proposed load balancing strategy and explore its effectiveness by applying it to real RFID middleware systems. Recently, our research center have developed an RFID middleware, RFID readers, and tags. The RFID middleware conforms to the standard proposed by EPCglobal [4] and supports various types of application development [14], [15], [16], [17]. The proposed load balancing approach will be incorporated into the middleware so that the middleware can be used for large scale applications.

References

[1] Michielsen, E.: RFID Middleware Market Competition Heats Up. RFID International Newsletter 2(16) (2004).
[2] Leaver, S.: Evaluating RFID Middleware. Forrester Research, Inc. (2004).
[3] Auto-ID Center, http://www.autoidcenter.org
[4] EPCglobal Inc, http://www.epcglobalinc.com
[5] Cao, J. et al.: A Framework of Using Cooperating Mobile Agents to Achieve Load Sharing in Distributed Web Server Groups: In: Proc. of the fifth Conf. on Algorithms and Architectures for Parallel Processing (2002).
[6] Balasubramanian, J. et al.: Evaluating the Performance of Middleware Load Balancing Strategies. In: Proc. of IEEE Conf. on EDC (2004).
[7] Zhu, W., et al.: An Experimental Study of Load Balancing on Amoeba. In: Proc. of Aizu Int. Symp. on Parallel Algorithms/Architecture Synthesis (1995) 220-226.
[8] Theimer, M. M., Lantz K. A.: Two Adaptive Location Policies for Global Scheduling Altorithms. In: Proc. of Conf. Distribued Computer Systems (1990) 502-509.
[9] Zhou, S., et al.: Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems. Software-Practice & Experience 23(12) (1993) 1305-1336.
[10] Payli, R. Ul, et al.: DLB . A Dynamic Load Balancing Tool for Grid Computing. In: Proc. of Parallel CFD (2004).
[11] Zaki, M. J, et al.: Customized Dynamic Load Balancing for a Network of Workstations. Journal of Parallel and Distributed Computing 43 (1997) 156-162.
[12] Cao, J., et al.: Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling. In: Proc. of IPDPS (2003).
[13] Myint, C. C., et al.: A Framework of Using Mobile Agent to Achieve Efficient Load Balancing in Cluster. In: Proc. of APSITT (2005) 66-70.
[14] H. Lee and M. Chung, "RFID-Based ALE Application Framework using Hierarchical Architecture," *Proc. of IASTED Conference on Advances in Computer Science and Technology*, 2006.
[15] D.W. Park and H.C. Kwon, "RFID-based Logistics Information Service with Semantic Web," *Proc. of IASW 2005*, 2005.
[16] Y. Kim, et al., "A Framework for Rapid Development of RFID Applications," *Proc. of ICCSA 2006*, 2006.
[17] H.S. Chae, et al., "A Framework based Design for Supporting Availability of Layered Distributed Applications," *Proc. of NGITS 2006*, 2006.