Journal of Engineering and Applied Sciences 13 (1): 38-42, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

On Balanced Deployment of Dynamic Objects in Distributed Environments

Lung Pin Chen and Hsin Ta Chiao
Department of Computer Science, Tunghai University, Taichung, Taiwan

Abstract: Typical modern internet applications rely on the collaboration of software components deploying in network hosts to deliver application services. As the previous literatures considered the total deploying cost, e.g., communication and computation costs of the service objects this researcher takes into account that how evenly the objects are distributed over the network. We demonstrate that evenly distributing the application components over the network can prevent bottlenecks and therefore, improves both of fault tolerance and system response time. In this study, we transform the minimum cost balanced deployment problem to the well known minimum cut problem and develop an efficient polynomial time deployment algorithm.

Key words: Distributed system, rich internet application, graph partition, deployment, polynomial, algorithm

INTRODUCTION

Modern internet applications rely on the collaboration of computational objects that are distributed in the network hosts to deliver the application services (Choi *et al*., 2013; Lee *et al*., 2013; Ambursa *et al*., 2017). The application performance is bounded by the network latency as well as the time required to compute the computational objects (Labrinidis *et al*., 2010; Viktorov, 2007; Samydurai and Shanmugam, 2014). Many research efforts have been devoted to partitioning and deploying a given set of computational objects over the hosts in varies distributed environments. For example, the literatures (Chen *et al*., 2010; Tang and Chanson, 2004; Vyavahare *et al*., 2016) considered the total deploying cost, e.g., communication and computation costs of the distributed objects.

This researcher investigates the object partition and deployment problem by considering not only the sum of the deploying costs but also how balanced the objects are distributed over the network. A balanced object deployment strategy tries to partition the objects to the network hosts evenly. The main benefit of evenly distributing the objects is that, we can enhance the performance via high parallelism and concurrency using multiple processors. Also, this strategy can prevent network bottlenecks and therefore, ensure the users share the network bandwidth fairly. Balanced object deployment also improves the degree of fault tolerance as there are no host take most objects that can be crashed at a same time.

In this study, we formally define the minimum cost balanced deployment problem and develop an efficient polynomial time algorithm. This proposed algorithm is based on the transformation of the minimum cut problem that has been extensively studied and efficiently solved (Goldberg and Tarjan, 1988; Shah *et al*., 2013).

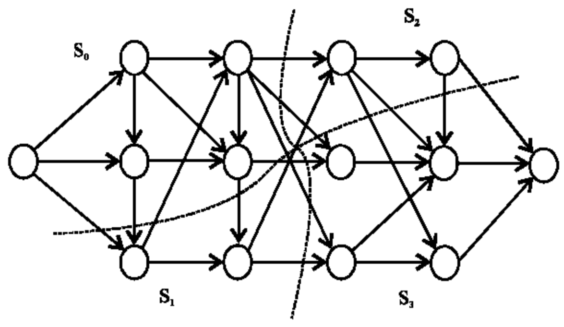

Fig. 1: An example of deploying an object computation graph on a 4-node mesh network

Problem definition: An object computation graph is represented as a DAG (Direct Acyclic Graph) G = (V, E), V = {v_1, v_2, ..., v_n} where each $v_i \in V$ refers to a data object and each link $(v_i, v_j) \in E$ refers to the dependency relation from object v_i to object v_j, $1 \le i, j \le n$.. We assume that G is a connected and the dependency relation induced by E is acyclic. Also, in the graph G those nodes without predecessors are called initial nodes (or initial objects) and those nodes without successors are called final nodes (or final objects).

In this study, an object deployment is a partition on a set of objects with each part assigned to a different network host. For example, Fig. 1 illustrates a deployment of objects on a mesh network with four hosts.

For a directed graph G, a bisection (L, R) partitions G into two parts L and R = V\L in terms of the precedence

Corresponding Author: Lung Pin Chen, Department of Computer Science, Tunghai University, Taichung, Taiwan relation induced by E. Specifically, (L, R) is a valid bisection if each object is partitioned along with all of its predecessor objects, i.e., $(u\in L)\wedge((u', u)\in E)\Rightarrow u'\in L$. Let the set of all bisections of G be denoted by B(G).

Let weight $(u)\in Z^+$ be the weight of node u of graph G. Also, let $w(S) = \sum_{u\in S}$ weight (S) for node set S. A bisection is said balanced if it equally partitions the node set as defined in definition 1.

Definition 1: For a mutually exclusive partition (L, R) on Graph G, the balancing factor is defined as $|w(L)-w(R)|$. The partition (L, R) is called a most balanced bisection if its balancing factor is minimum among B(G).

MATERIALS AND METHODS

Basic most balanced algorithm: This study employs the arithmetic and geometric means formulas to transform the MBP problem to the well-known minimum cut problem that can be efficiently solved in polynomial time.

Recursive bisection: Recursively bisecting an object computation graph to adapt to various network topologies is an approach commonly employed for partitioning and deploying objects. The recursive bisecting is a divide-and-conquer procedure. It first bisects a partition of objects P into two balanced parts P_L and P_R and then solves the smaller problems on P_L and P_R recursively until k partitions are obtained where k is the number of network hosts.

Figure 2 presents the divide and conquer procedure performed on three typical network topologies. In the linear structure shown in Fig. 2a, P_L is assigned to hosts 1 to k/2 and P_R is assigned to hosts k/2+1 to k. In the tree structure shown in Fig. 2b, P_L and P_R are assigned to the left and right subtrees, respectively. Also, in the mesh structure shown in Fig. 2c, P_L and P_R are assigned to the left (or upper) and right (or lower) submeshes, respectively.

Partition using arithmetic and geometric means: Given a list of real numbers $x_1, x_2, ..., x_d$ the arithmetic mean and geometric mean are defined as $(x_1+x_2+...+x_d)/d$ and $\sqrt[d]{x_1 x_2 \cdots x_d}$, respectively. The fundamental Arithmetic and Geometric Mean inequality (AM-GM inequality) states that the arithmetic mean must greater than or equal to the geometric mean and further the equal case occurs at $x_1 = x_2 = ... = x_d$.

A simple application of the AM-GM inequality is that among all the rectangles of perimeter 2n (we assume that, n is an integer and is even), the rectangle with width equals to height (i.e., the square shape) has the greatest area. For a rectangle of height x_1 and width x_2 satisfying $x_1+x_2 = n$, the AM-GM inequality states that:

$$(x_1+x_2)/2 \geq \sqrt{x_1x_2} \quad (1)$$

Taking the square root of both sides of Eq. 1 obtains:

$$((x_1+x_2)/2)^2 \geq x_1x_2 \quad (2)$$

Further, the equal case occurs at $x_1 = x_2$. We transform the balanced bisection problem using this AM-GM property as follows. Figure 3 demonstrates the relation between AM-GM property and balanced bisection. For a bisection (L, R) those edges across L and R are termed cross edges while the others (i.e., the edges with their end points both in L or both in R) are termed non-cross edges. As Fig. 3 illustrates, the rectangle of area corresponds to

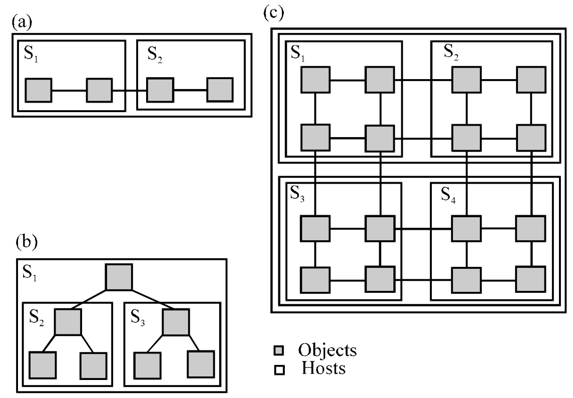

Fig. 2: Hierarchical object deployment: a) Linear structure; b) Tree structure and c) Grid structure

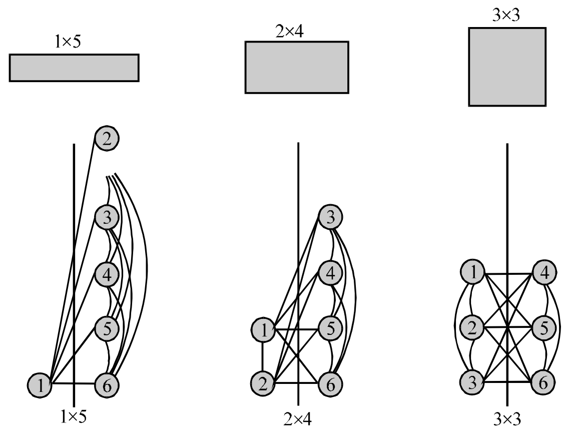

Fig. 3: The 2-dimensional rectangles and their corresponding partitions

the bisection (L, R) with |L| = x_1 and |R| = x_2. In this example, $x_1+x_2 = 6$. According to in Eq. 2 and AM-GM property, the largest rectangle area $x_1 \times x_2$ occurs at $x_1 = x_2 = 3$, the bisection (L, R) with maximum number of cross edges must have |L| = |R| = 3.

For mathematical convenient, we assume that the number of nodes n is even which is divisible by 2. Given an object computation graph G, we construct a transformation graph H as follows.

Given G = (V, E), the transformation graph H = (V, E') is a complete graph where E' is the set of n(n-1) edges that connect every pair of nodes. Theorem 3 proves the correctness of the above transformation based on AM-GM inequality.

Theorem 3: For an object computation graph G and its transformation graph H. Assume that weight (u) = 1 for all node u. The pair (L, R) is a balanced bisection of G if and only if (L, R) incurs maximum number of cross edges in H.

Proof: Clearly, since, H is a complete graph, each bisection (L, R) of H incurs |L|×|R| cross edges. Based on AM-GM inequality, since, |L|+|R| = n where n is the number of nodes and is even, the maximum value of |L|×|R| occurs at |L| = |R|. That is the bisection with maximum number of cross edges corresponds to the balanced bisection.

Weighted MBP: Theorem 3 discusses the unweighted balanced bisection. For the weighted MBP problem, each node can have different weight and the partition is said balanced if the total weights of the partitions are as equal as possible. For example, in Fig. 4, partition (a) is balanced since, both sides incur total weight 4 while partition (b) is unbalanced as the left and right side incur unequal weights 5 and 3, respectively.

Although, the AM-GM inequation is defined on real numbers, it can be applied to the discretized case that the node weights cannot be fully equally divided. Lemma 4 shows that for two bisections, more balanced partition will incur a greater multiplication value of the weights of the two partitions.

Lemma 4: For an object computation graph G and its transformation graph G. The pair (L, R) is the weighted MBP of G if and only if w(L)×w(R) = w(L')×w(R') for all (L', R')∈B(G).

Proof: This property can be proved by reducing the weighted MBP problem to the unweighted MBP problem. Given an object computation graph G with weight (v_i) = k_i

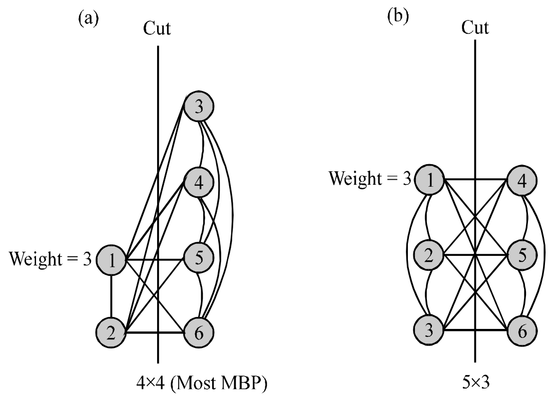

Fig. 4: Partitioning the weighted nodes: all nodes have weight of 1 except that weight node 1 = 3; a) Balanced partition and b) Unbalanced partition

for each node v_i. We construct a graph G' by duplicating k_i nodes in G' (with weight 1) for each node v_i in G. Formally, for each node in G with weight k_i, add k cloned nodes $d_{i,1}$, $d_{i,2}$, ..., $d_{i,k}$ to G'. Also, for each edge (v_i, v_j) in G, add ($d_{i,p}$, $d_{j,q}$) edges in G' for all p,q, $1 \le p \le k_i$, $1 \le q \le k_j$. Therefore, for an edge (v_i, v_j) in G there are $k_i \times k_j$ edges in G'.

Applying AM-GM inequality on G', the most balanced partition (L', R') must satisfy |L'| = |R'|. Now reducing from G' to G, the weighted maximum number of cross edges of G' corresponds to the weighted most balanced partition G. Note that in order to reducing G' to G correctly we restrict a group of cloned nodes in G' must be partitioned in a bundle manner. The proof is ignored in this study.

RESULTS AND DISCUSSION

Transformation algorithm: This subsection discusses the weighted MBP algorithm which solves the problem by reducing to the well-known minimum cut problem. Unlike the MBP problem which deals with weights on nodes, the minimum cut problem is to find the bisection with minimum weights on edges. Let cap denote a capacity function mapping edges to integer values. A cut of a graph is a bisection (L, R) of the graph and the cost of the cut is defined as the sum of capacity values of edges across L and R.

A similar problem is the maximum cut problem which is intend to find a cut with maximum cut cost among all cuts. Finding maximum cut is known an NP-complete problem. In this subsection, we first transform the MBP problem to the maximum cut problem. Since, an object computation graph is acyclic and we restrict the graph must be partitioned under the precedence relation, we than further transform the minimum cut to the minimum cut which can be solved in polynomial time. The main algorithm most balanced bisection is listed in Algorithm 1.

Algorithm 1; Most balanced bisection:

```
1: let G = (V, E) be the input object computation graph
2: // construct transformation graph H = (V', E')
3: for each node $u_i \in V$ do
4:     let $k_i$ = weight($v_i$)
5:     add a group of clone nodes $d_{i,p}$, P = 1, 2, ..., $k_i$, to V'
6:     add an ∞-capacity edge between every pair of clone nodes of $u_i$
7: end for
8: for each pair of $d_{i,p} \in V'$, $d_{j,q} \in V'$ do
9:     add ($d_i$, $d_j$) to E'
10:    let cap ($u_i$, $u_j$) = 1
11: end for
12: //transfer max-cut to min-cut
13: let (L', R') = max-cut(H)
14: //reduce to the most balanced partition
15: let (L, R) be the original node of the clone nodes in (L', R')
16: return (L, R)
```

According to the AM-GM property, the most balanced partition has the greatest value of multiplication of the number of nodes on the left and right sides. In Algorithm most balanced bisection, given an object computation graph G with weight $(v_i) = k_i$ for each node v_i. It constructs a group of clone nodes in G' for each node v_i in G. In algorithm most balanced bisection line 6 there are ∞-capacity edge between each pair of clone nodes. Since, an ∞-capacity edge cannot be cut by a minimum cut, a group of clone nodes must be aggregated in one side of the bisection.

In algorithm most balanced bisection line 12-13, it invokes the maximum cut procedure to find the partition with maximum total capacity of the cross edges. For the special application of object computation graph, the maximum cut can be transformed to the minimum cut and solved in polynomial time (Chen *et al*., 2010).

In algorithm most balanced bisection line 4-5, each object v_i with weight k_i in G is mapped to a set of k_i clone nodes in G'. Thus, for a bisection (L, R) of G, the corresponding bisection of G' will have w(L) clone nodes in the left side and w(R) clone nodes in the right side. Further, the corresponding bisection of G' must have $w(L)\times w(R)$ cross edges. According to the AM-GM property, the most balanced partition has the greatest value of $w(L)\times w(R)$. Thus, algorithm most balanced bisection finds such most balanced partition correctly.

Enhanced transformation algorithm: In the above MBP algorithm, one object in G of weight k is mapped to k clone nodes in the transformation graph H. Furthermore, in graph H it incurs all-to-all edges connecting the nodes.

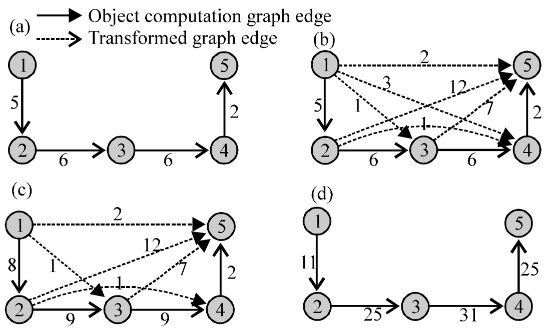

Fig. 5: a) Original object computation graph; b) Transformation graph; c) Merge the transformation edge (1, 4) along path (1-4) and d) Enhanced transformation graph

Both these properties lead to a transformation graph with large number of edges. This subsection discusses an approach to reduce the number of edges in H.

Recall that in study problem and definition, the object computation graph is a DAG and its nodes must be partitioned under the constraints of the precedence relation. Let us first consider a simple case that all nodes are arranged in a sequence of precedence relation as shown in Fig. 5a. In Fig. 5, the precedence relation is represented as bold arcs while the transformed graph edges are represented as thin arcs. The numbers beside arcs represent the capacity value of the arcs. Since, the predecessor nodes of a node v_i must be partitioned along with v_i in Fig. 5a there are only three valid bisections: ({1},{2, 3, 4}), ({1, 2},{3, 4}) and ({1, 2, 3},{4}).

Now, let us examine the transformed edge (1, 4) with capacity 3 in Fig. 5b. Every bisection cuts edge (1, 4) must also cut the path (1-4) exactly once. Thus, we can propagate the capacity value 3 along path (1-4) as shown in in Fig. 5c. Through this propagation procedure, the capacity of edge (1,4) is transferred to the edges on path (1-4) without changing the total cut cost of the valid bisection. Figure 5d shows the result of merging all the transformation graph edges to the precedence relation edges.

CONCLUSION

This researcher presented the minimum cost balanced deployment problem and develop an efficient polynomial time algorithm. This proposed algorithm is based on the transformation of the minimum cut problem that has been extensively studied and efficiently solved. This researcher presented the enhanced algorithm to significantly reduce the number of edges in the transformation graph by merging the transformation edges.

ACKNOWLEDGEMENT

This researcher was supported in part by the Ministry of Science and Technology of the Republic of China (Taiwan) under Contract MOST-105-2221-E-029-018.

REFERENCES

Ambursa, F.U., R. Latip, A. Abdullah and S. Subramaniam, 2017. Probabilistic reliability prediction models for task scheduling in distributed systems: A review. J. Eng. Appl. Sci., 12: 644-652.

Chen, L.P., I.C. Wu, W. Chu, J.Y. Hong and M.Y. Ho, 2010. Incremental digital content object delivering in distributed systems. IEICE. Trans. Inf. Syst., 93: 1512-1520.

Choi, J., C. Choi, B. Ko, D. Choi and P. Kim, 2013. Detecting web based DDoS attack using MaPreduce operations in cloud computing environment. J. Internet Serv. Inf. Secur., 3: 28-37.

Goldberg, A.V. and R.E. Tarjan, 1988. A new approach to the maximum-flow problem. J. ACM. JACM., 35: 921-940.

Labrinidis, A., Q. Luo, J. Xu and W. Xue, 2010. Caching and materialization for web databases. Found. Trends Databases, 2: 169-266.

Lee, T., H. Kim, K.H. Rhee and S.U. Shin, 2013. Implementation and performance of distributed text processing system using hadoop for E-discovery cloud service. J. Internet Serv. Inf. Secur. JISIS., 4: 12-24.

Samydurai, A. and A. Shanmugam, 2014. Fault tolerant middleware framework to improve QoS in distributed systems. Res. J. Appl. Sci., 9: 154-159.

Shah, V., B.K. Dey and D. Manjunath, 2013. Network flows for function computation. IEEE. J. Sel. Areas Commun., 31: 714-730.

Tang, X. and S.T. Chanson, 2004. Minimal cost replication of dynamic web contents under flat update delivery. IEEE. Trans. Parallel Distrib. Syst., 15: 431-439.

Viktorov, O., 2007. Providing fault-tolerance of distributed systems by process-to-processor reassignment. J. Eng. Applied Sciences, 2: 1563-1564.

Vyavahare, P., N. Limaye and D. Manjunath, 2016. Optimal embedding of functions for in-network computation: Complexity analysis and algorithms. IEEE. ACM. Trans. Networking, 24: 2019-2032.