GTapestry: A Locality-Aware Overlay Network for High Performance Computing*

Hai Jin, Fei Luo, Qin Zhang, Xiaofei Liao, Hao Zhang
Cluster and Grid Computing Lab
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract

A locality-aware overlay network for high performance computing, GTapestry, is proposed in this paper. GTapestry assembles physically neighboring nodes in the Internet into self-organized groups. The routing mechanism of GTapestry is divided into inter-group routing, used by groups that communicate with others through their leaders, and intra-group routing, used by group members to communicate with each other directly. Compared with Tapestry, the performance of GTapestry, including the routing efficiency and the maintenance cost for stability, is evaluated in theory and via simulation. It indicates that GTapestry can be the substrate of peer-to-peer based HPC systems for applications with task-level parallelism.

1. Introduction

It is a new trend to cluster many computers together to construct *high performance distributed computing* (HPDC) platforms. As one of HPDCs, *peer-to-peer* (P2P) fashion can construct supercomputers far beyond the power of any current computing center.

Current P2P-based HPDC systems are mainly in the style of volunteer computing. There is a limitation for the range of their applications which must be divided into independent subtasks. To support applications with data dependence between subtasks in such systems, two fundamental issues have to be considered. First, their substrates have to provide communication manners for nodes to interchange information between their subtasks. Second, the communication latency must be reduced to enhance the high performance.

* This paper is supported by National Science Foundation of China under grant 60433040, China CNGI project under grant CNGI-04-12-2A and CNGI-04-12-1D.

A new P2P-based platform for HPC, *P2HP*, has been proposed to provide a communication mode for a large range of applications. The substrate of P2HP, GTapestry, is a locality-aware overlay network described in this paper. Physically neighboring nodes in GTapestry are clustered into a medium-size scale work groups to collaborate for subtasks. With the help of the communication mechanism of GTapestry, P2HP extends its high performance applications, which avoids the limitation of independence between subtasks. According to theory analysis and simulation, GTapesty provides better performance than Tapestry.

Combining the structured P2P network with the unstructured one, GTapestry is the first step to utilize locality-aware overlay network in HPC areas. The rest of the paper is organized as follows. Related work is presented in Section 2 and we will describe the basic communication model of GTapestry in Section 3. To evaluate its performance, the theoretical analysis and simulations are presented in Section 4 and 5. Finally, conclusions are drawn in Section 6.

2. Related Work

Current volunteer computing systems exploit unstructured P2P networks, such as SETI@home [1], StremWeb [2] and OurGrid [3]. They just support applications with independent subtasks. P2HP exploits structured P2P network, GTapestry, to extend its applications without the limitation of independence.

Several scalable P2P proposals based on *distributed hash tables* (DHT) have been put forth, including CAN [4], Chord [5], Pastry [6] and Tapestry [7][8]. Their routing is independent of the underlying physical networks, which provides great communication latency, and is not suitable for HPC substrate.

It is brought forward in [9][10][11] that better performance can be gained when physically neighboring peers assemble together to collaborate for tasks in a P2P network, compared with a single peer to work alone. To reduce communication latency for P2HP, GTapestry assembles physically neighboring nodes to form medium-size groups, which exploits both the metrics of the dynamic landmark in [12] and the IP-to-location mapping information in [13].

3. Construction of GTapestry

Physically neighboring nodes adjoin each other in a group, or in close groups. The ID of each group is coded according to its domain and ISP. The IDs of subtasks are attained in the same namespace with that of the groups, and the subtasks are stored in the groups with the same or most matching IDs, where the groups are correspondingly called root groups or surrogate ones.

3.1 Membership management

The members of a group are assigned with different roles. The most stable one is elected as the leader, and the others are its workers and candidates. When the leader fails, one of its candidates will be elected to take its position. When a worker leaves the system, others in the same group will relay its work.

The group is an unstructured overlay network, where the leader maintains dispatched subtasks' states, and preserves a value of the *communication delay* (CD) which is the threshold, allowed maximum latency between a member and the leader.

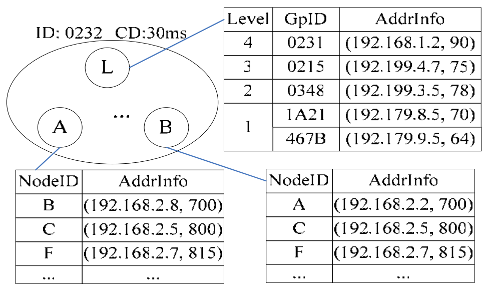
3.2 Routing and location

The routing in GTapestry is divided as intra-group routing according to the member neighbor table, and inter-group routing according to the group neighbor table. The member neighbor table consists of all other members' address information in its group, while the group neighbor table is a map with multiple levels. Each level contains links to groups matching a prefix up to a digit position in the ID and contains a number of entries equal to the ID's base. The closest group with an ID that begins with prefix *(N, j-1) + "i"* is the primary *i*th entry in the *j*th level.

Figure 1 shows the outgoing of links of the group with ID 0232, where the neighbor groups' information of each level is stored in the leader (*L*), and members' information is in each one (*A*, *B*) of this group. The backpointers to others are also stored.

Subtasks' IDs and their references are stored in the leader of a group, while their data are in workers, shown in Figure 2-a. To locate the needed subtask in a group, the member attempts through intra-group routing which passes through 2 hops. It first inquires its leads' references, and then communicates with the member who hosts the task, e.g., node *A* locates subtask with ID 0238 in Figure 2-a.

If the node fails to locate needed subtasks in its group, it will continue through inter-group routing. The router for the *n*th hop shares a prefix whose length is no less than *n* with the target ID, and the current group looks in its (n+1)*th* level neighbor table for the entry matching the next digit with the target ID. Its root or surrogate group will be reached at most log_bG logical hops, where *G* is the namespace size of the groups' IDs, and *b* is the base of ID. As shown in Figure 2-b, node *S* locates subtask 0238 through inter-group routing.

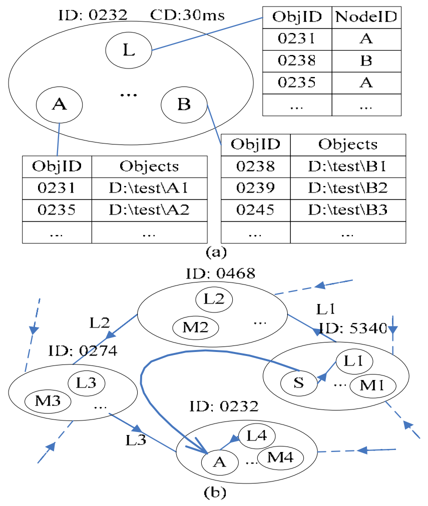


Level	GpID	AddrInfo
4	0231	(192.168.1.2, 90)
3	0215	(192.199.4.7, 75)
2	0348	(192.199.3.5, 78)
1	1A21	(192.179.8.5, 70)
	467B	(192.179.9.5, 64)

NodeID	AddrInfo
B	(192.168.2.8, 700)
C	(192.168.2.5, 800)
F	(192.168.2.7, 815)
...	...

NodeID	AddrInfo
A	(192.168.2.2, 700)
C	(192.168.2.5, 800)
F	(192.168.2.7, 815)
...	...

Figure 1. Neighbor tables in Group 0232



ObjID	NodeID
0231	A
0238	B
0235	A
...	...

ObjID	Objects
0231	D:\test\A1
0235	D:\test\A2
...	...

ObjID	Objects
0238	D:\test\B1
0239	D:\test\B2
0245	D:\test\B3
...	...

Figure 2. Routing to an object with ID 0238

3.3 Dynamic maintenance

Local dynamics is one of the basic properties of GTapestry. Metrics for ensuring the system's relative stability must be taken to enhance its usability.

3.3.1 Dynamics of nodes The joining node *N* gets the ID of its candidate group (C_{ID}) according to the IP-to-location mapping library. It finds a bootstrap node and requests its leader to route to the candidate

G_C with C_{ID}. Then N sends a joining request to G_C's leader with the communication delay. If the delay is less than the threshold of G_C, N joins it, or its nearest neighbor is elected as the new candidate. At the end of the iterated joining process, it will form a new group with the ID that shares a prefix with C_{ID} when there is no candidate for N to join.

If the member number in a group is more than 1, the leaving of a member will incur the change of the group's state and other members will take the node's work. The event is notified by the node if voluntary leaving detected by other members. If only one member exists in a group, the leaving will incur voluntary or involuntary deletion of a group in GTapestry.

3.3.2 Dynamics of groups The joining group G_N begins at G_N's surrogate S. G_N fills its group neighbor table from the level p, which is the length of the longest prefix of the ID that S shares with G_N. G_N sends multicast message to all groups sharing the same prefix. As the group leaders receive the message, they add G_N to their group neighbor tables and transfer references of local root pointers and then contact G_N. All the contacted groups become an initial group neighbor set used in its routing table construction. G_N selects at most b nearest groups from the group neighbor set to fill routing level p, and requests these groups to send their backpointers at that level. The corresponding groups to those backpointers become the next neighbor set. Then G_N decreases p, and repeats the process until p=1.

A group G_D will notify the event of its deletion to all groups in its neighbor table if the event is voluntary. If the event is involuntary, it will be detected by other groups. Groups that get the event will delete the entry relative to G_D in its group neighbor table and notify the groups responding to their backpointers to regulate the neighbor tables, and the process is iterated until GTapestry is stable.

3.3.3 Self-organization of groups If a group G_1 detects that its neighbor G_2 is idle and the number of its members is small, G_1 requests G_2 to unite with it. The members of G_2 first incur voluntary node and group deletion events, and then join G_1. The threshold of new G_1 will be tuned as Eq. 1, where *DT.max* is the system maximal *DT*.

$$DT=min(2*DT, DT.max) \qquad (1)$$

Continuous nodes joining in a group will make the group size very large. To maintain a medium size for a group, the over-scale one will be split into two. One of them inherits the original properties. While the other joins GTapestry as a new one.

4. Characteristics Analysis

To evaluate GTapestry performance, we compare it with Tapestry which has many novel performances. In GTapestry, it is assumed that the group number is g, the ID base is b, and the ID namespace is G. A node routes to another cost at most log_bG inter-group routing hops and 2 intra-group routing hops.

With the same number of nodes of GTapestry, Tapestry's corresponding namespace is $g\times G$, and the expected routing hops from a node to another is at most $log_b(g\times G)$. The difference $Diff_H$ (Eq. 2) between their maximal routing hops is small.

$$Diff_H=log_bG+2-log_b(g\times G)=2-log_bg \qquad (2)$$

Each node in GTapestry keeps a group neighbor table with $blog_bG$ entries and a member neighbor table with $(g-1)$ entries. Correspondingly the Tapestry node keeps $blog_b(g\times G)$ entries in its neighbor table. The difference $Diff_O$ (Eq. 3) between their routing costs is small too, where b and g are medium constant.

$$Diff_0=blog_bG+g-1-blog_b(g\times G)=g-(1+blog_bg) \qquad (3)$$

Nodes in Tapestry reconstruct their neighbor tables for dynamic maintenance, and the modification of neighbor tables is the operation for reconstruction. A Tapestry node's deletion will cause at most $blog_b(g\times G)$ other nodes to reconstruction its neighbor table. If it is assumed that a node's deletion probability is p, the average operations AV_T is shown as Eq. 4.

$$AV_T=pblog_b(g\times G) \qquad (4)$$

The group reconstruction will be incurred only when all its members leave GTapestry and the probability is p^g. For g>1, p^g is much less than p. A leader's deletion brings at most $blog_bG$ groups and $(g-1)$ members to reconstruct neighbor tables, while a member's deletion incurs at most $(g-1)$ members to reconstruct their tables. The average operations AV_{GT} (Eq. 5) is much more than AV_{GT}. It indicates that GTapestry is more stable than Tapestry.

$$\begin{aligned} AV_{GT} &= \frac{pb\log_b G + g - 1 + (g-1)p(g-1)}{g} \\ &= \frac{pb\log_b G}{g} + 1 - 1/g + p(g - 2 + 1/g) \end{aligned} \qquad (5)$$

5. Simulations

GTapestry routing efficiency and maintainence cost are further evaluated through simulations.

5.1 Routing efficiency

To compare their routing efficiency, the routing hops for locating an object are evaluated. All objects are stored in roots or their surrogates in Tapestry or

GTapestry. The environment is set as below (the namespace base (Base) and level (Level), the nodes' number (NodeNum), the group size (GroupSize) and the objects' number (ObjNum)):

Base=8; Level=10; GroupSize=5; NodeNum=5000; ObjNum=5000 (6)

The circumstances are changed with the variety of one parameter, and the routing hops are presented in Figure 3 to Figure 6. The curve of "Tapestry" represents the routing of Tapestry. GT-AVG, GT-INTRA, and GT-INTER denotes GTapestry's total routing, intra-routing and inter-routing, respectively.

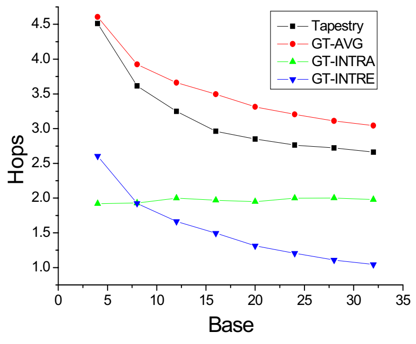


Figure 3. Routing efficiency with Base

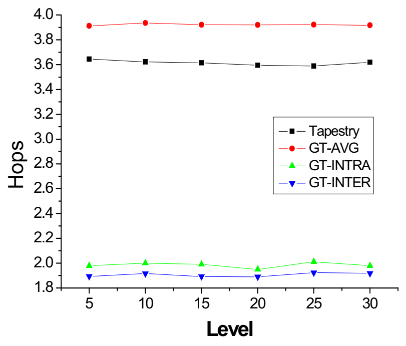


Figure 4. Routing efficiency with Level

Although the total routing hops of GTapestry are a little more than those of Tapestry, the intra-routing hops almost hold invariable and the inter-routing hops are much less than those of Tapestry, shown in Figure 3 to Figure 6. Because the location in a physically neighboring network is much more efficient than that between two physically disjoint nodes in the Internet, the routing cost of GTapestry mainly depends on its inter-routing. Therefore, its routing cost is much less than that of Tapestry, which indicates that GTapestry is more efficient than Tapestry.

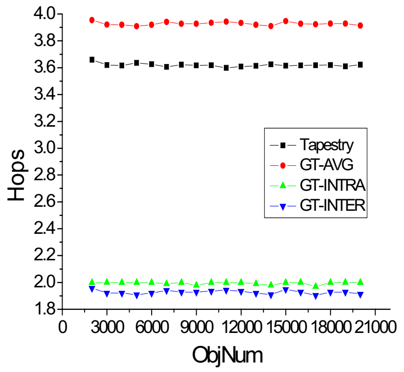


Figure 5. Routing efficiency with ObjNum

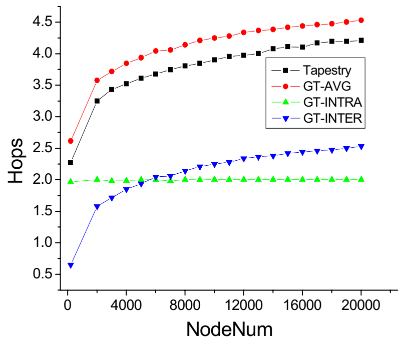


Figure 6. Routing efficiency with NodeNum

Some other properties of GTapestry can also be inferred. Its routing hops decrease with the increase of Base (Figure 3) and hold invariable with the change of Level (Figure 4) or ObjNum (Figure 5). They also increase with the increase of NodeNum (Figure 6) and decreases with the rise of GroupSize (Figure 7). It demonstrates that GTapestry's routing efficiency only depends on density of the system construction.

5.2 Maintenance for stability

The maintenance cost for nodes leaving is further evaluated, where the metric is to measure the number of maintaining operations. The primary operation for a node's leaving in Tapestry is to delete items in neighbor tables (Tp-DEL), and the neighbor table is reconstructed as necessary.

The leaving of a member in GTapestry causes related items deletion in member neighbor tables (Intra-delete). The leader leaving causes update of relative group neighbor tables and deletion of its members' member neighbor table. The group deletion incurs that other groups delete related items (Inter-delete) in their group neighbor tables. We distinguish the maintenance operations (G-OP) of GTapestry as inter-operations (G-INTER) between groups and intra-operations (G-INTRA) in a group.

With the environment set below, where DeleteNum is the maximal number of leaving nodes, the operation curves are shown in Figure 8 to Figure 11, which indicates that the maintenance cost of GTapestry is much less than that of Tapestry.

Base=8; Level=10; GroupSize=5; NodeNum=5000; DeleteNum=5000 (7)

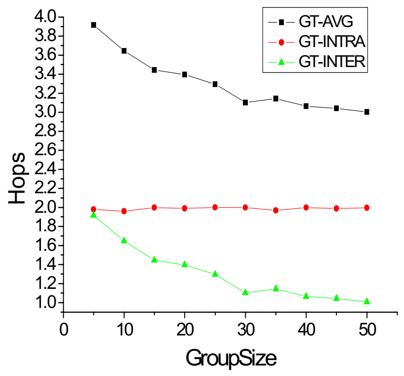


Figure 7. Routing efficiency with GroupSize

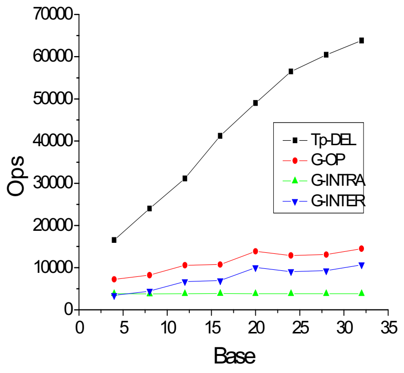


Figure 8. Maintenance with Base

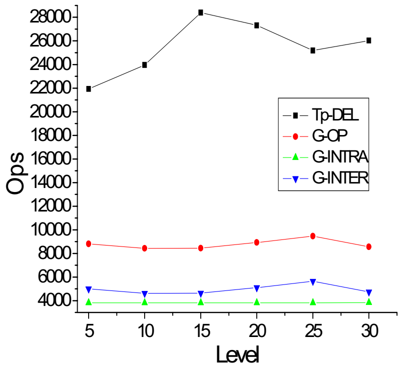


Figure 9. Maintenance with Level

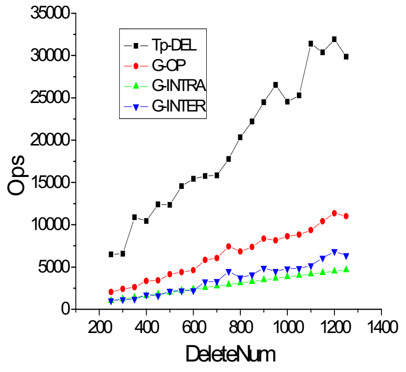


Figure 10. Maintenance with DeleteNum

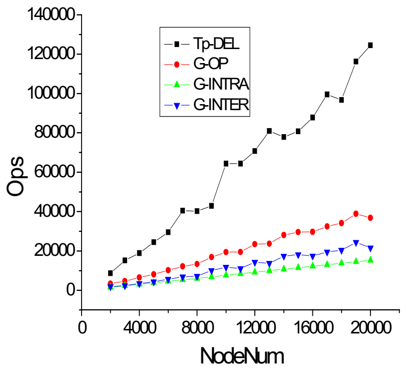


Figure 11. Maintenance with NodeNum

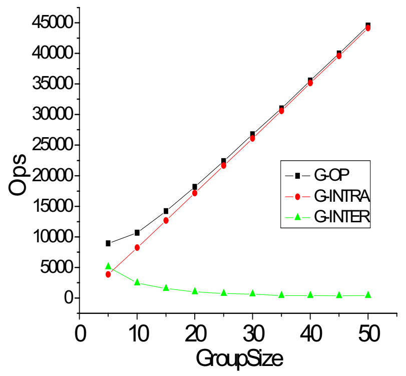


Figure 12. Maintenance with GroupSize

It also demonstrates that the maintenance cost of GTapestry is mainly affected by the network complexity, such as the Base, NodeNum, DeleteNum and GroupSize (Figure 8, 10, 11), while not the network scale (Figure 9). Furthermore, Figure 12 tells that advisably increasing the group size can reduce the maintenance cost of GTapestry.

6. Conclusions

GTapestry is a locality-aware overlay network and it is the substrate of a P2P-based HPC platform, P2HP. It assembles physically neighboring nodes in the Internet into medium-size groups and all groups are self-organized. Each group is responsible for one or some subtasks and all members collaborate to execute subtasks and exchange data with others. The routing in GTapestry is distinguished as intra-group routing and inter-group routing. Metrics to maintain the stability and enhance the usability of the system is adopted.

Compared with Tapestry, GTapestry is a scalable and relatively stable network through analysis and simulation. Some basic parameters can be leveraged to get better performance. It shows that GTapestry is able to provide an efficient and scalable substrate for P2HP, where the communication mechanism supports various applications with task-level parallelism.

References

[1] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An experiment in public-resource computing", *Communications of the ACM*, Vol.45, No.11, 2002, pp.56~61.

[2] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: A generic global computing system", *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, New York, 2001, pp.582~587.

[3] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg, "OurGrid: an approach to easily assemble grids with equitable resource sharing", *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'03)*, 2003, pp.61~86.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network", *Proc. of SIGCOMM'01*, San Diego, CA, Aug. 2001, pp.161-172.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", *Proc. of SIGCOMM'01*, San Diego, CA, Aug. 2001, pp.149-160.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", *Proceedings of the 18th IFIP/ACM Int'l Conf. on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001, pp.329-350.

[7] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing", Univ. California, Berkeley, CA, *Tech. Rep. CSD-01-1141*, Apr. 2001.

[8] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment", *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, January 2004, pp.41~53.

[9] K. Shin, S. Lee, G. Lim, H. Yoon, and J. S. Ma, "Grapes: Topology-based hierarchical virtual network for peer-to-peer lookup services", *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, 2002, pp.1456~1471.

[10] L. Garcés-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-Keller, "Hierarchical peer-to-peer systems", *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003.

[11] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection", *Proceedings of IEEE INFOCOM'02*, Vol.3, 2002, pp.1190~1199.

[12] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance", *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, Jan. 2004, pp.18~28.

[13] V. Kata, N. Padmanabhan, and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts", *Proc. of SIGCOMM'01*, San Diego, CA, Aug. 2001, pp.173~185.