Massively Parallel Computing on Peer-to-Peer Networks

Team Timeout
Jon Ludwig
Prashant Gahlowt
Young Suk Moon

Summary

- Effectively distribute a set of computing tasks to a peer-to-peer network
  - All peers want the finished product
  - Peers may join and drop freely
  - Decentralized and self-organizing

Overview

- Summary of the project
- “Peer to Peer Computing”
- “ Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”
- “Dynamic Load Balancing in Parellel Processing on Non-Homogeneous Clusters”
- Progress

Peer-to-Peer Systems

- What types of peer-to-peer systems are available?
  - Presents a survey of existing P2P systems
- Which models are best for which environments?
  - Compare and contrast systems

Characteristics of P2P

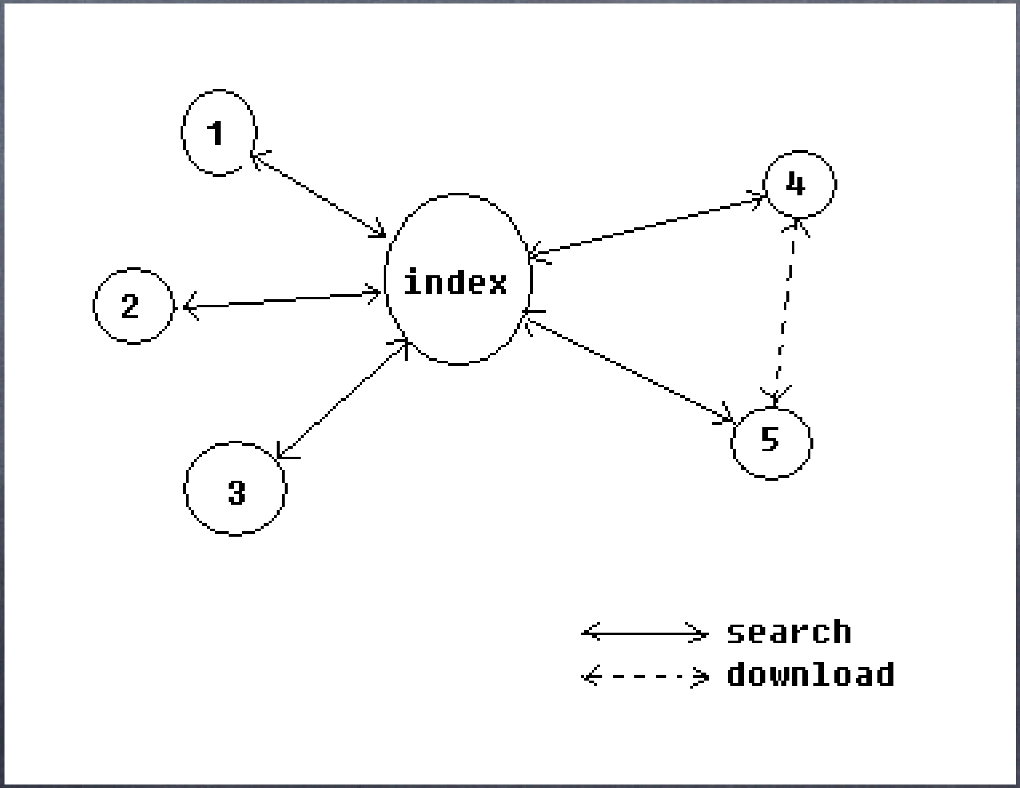
- Decentralization
- Scalability
- Anonymity
- Self-Organization
- Cost of Ownership

Characteristics of P2P

- Ad-Hoc Connectivity
- Fault Resilience
- Performance
- Transparency

P2P Algorithms

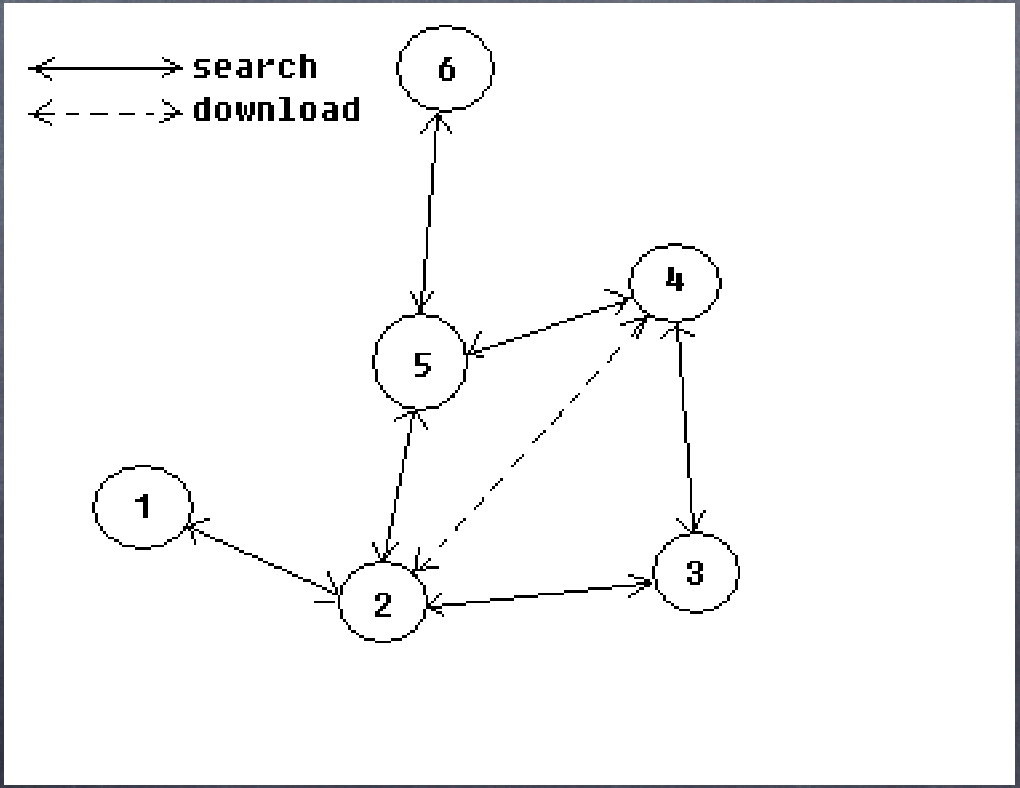
- Centralized directory model



[2]

P2P Algorithms

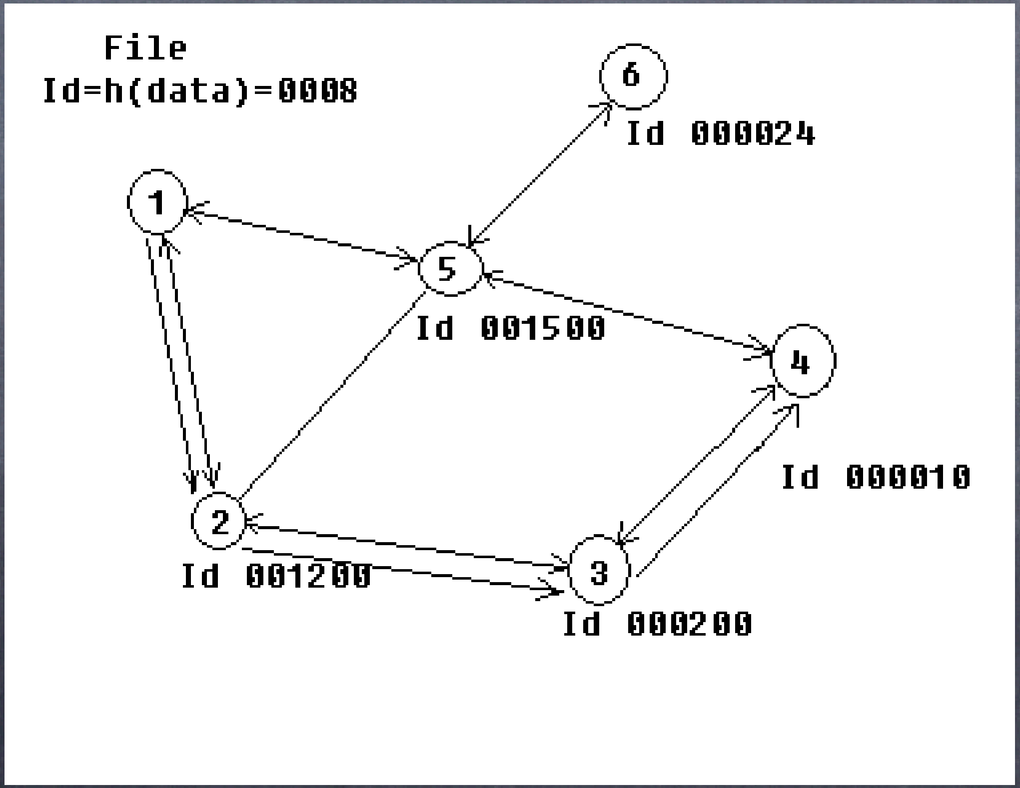
- Flooded requests model



[2]

P2P Algorithms

- Document routing model



[2]

Categories of P2P Systems

- Distributed Computing
- File Sharing
- Collaboration
- Platforms

P2P Systems

- Gives us useful factors to consider when evaluating the performance of our system
- Try to use the advantages from other systems and avoid the disadvantages

Research Paper:

"Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems"

– Antony Rowstron

– Peter Druschel

Pastry: Quick Recap

- Completely decentralized, fault resilient, scalable and reliable with good locality properties.
- Intended as general substrate for variety of P2P Internet apps like file sharing, file storage, etc.
- Consistent hashing: 128 bit circular id   N*odeIds* (uniform random)

  Message keys (uniform random)
- NodeId randomly assigned from {0, .., 2^{128}-1}, |L|, |M| are configuration parameters
- Expected number of routing steps is **O(log N);** N=No. of Pastry nodes in the network
- Under normal conditions: A pastry node can route to the numerically closest node to a given key in less than $\mathbf{log_{2b} N}$ steps.
- Despite concurrent node failures, delivery is guaranteed unless more than |L|/2 nodes with adjacent NodeIds fail simultaneously
- Invariant: node with numerically closest nodeId maintains objectMsg with key *X* is

Pastry Design: Node State

- Each node maintains: routing table-R, neighborhood set-M, leaf set-L.
- **Routing table** is organized into $[\log_{2^b} N]$ rows with 2^b-1 entry each.Each entry contains the IP address of a close node with appropriate prefix. Choice of b - tradeoff between size of routing table and length of route.
- **Neighborhood set** - nodeId , IP addresses of ｜M｜ closest nodes , useful for maintains locality properties.
- **Leaf set set of** ｜L｜ nodes with closest nodeId to current node.L - divided into 2 : ｜L｜ /2 closest larger, ｜L｜ /2 closest smaller.

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

[3]

Routing

- The routing procedure is executed whenever a message arrives at a node.
- First check if key is in the range of the leaf set.
  - **If** yes destination node is at most one hop away.
  - **Else** - forward the message to the node (from the routing table) with shared prefix that is longer in one then the current. Destination is reached in $[\log_{2^b} N]$ steps.
  - **Else** - In case entry is empty forward to a node with at least shared prefix like current node but it is numerically closer. The probability of the third case is less then 0.006 for $|L| = 2*2^b$.

Pastry API

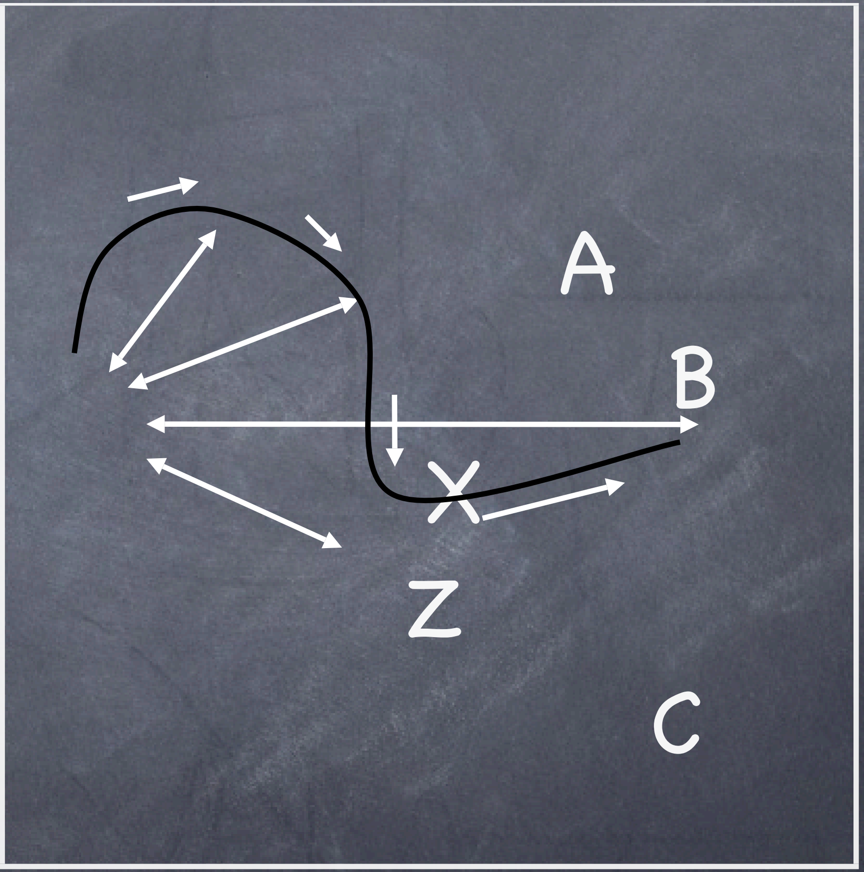
- *Pastry exports the following operations:*
  - **nodeId = PastryInit(Credentials, application)**
    - Local node join to Pastry network,init state , and return nodeId to application.
  - **Route(msg,key)**
    - Causes Pastry to route the given message to the node with NodeID numerically closest to the key.
- *Application layered on top of Pastry must export the following operations:*
  - **Deliver(msg,key)**
    - Called by Pastry when a message is received and the local node NodeID is numerically closest to key.
  - **Forward(msg,key)**
    - Called by Pastry just before a message is forwarded to the node with NodeID=nextID.The application may change the contents of the message or the value of nextID
  - **newLeafs(leafset)**
    - Called by Pastry whenever there is a change in the local node leaf set.This provides the application with an opportunity to adjust application specific invariants based on the leaf set.

Self-organization: Node Arrival

- Arriving Node X knows nearby node A
- X asks A to route a "join" message with key = NodeId(X)
- Message targets Z, whose NodeId is numerically closest to NodeId(X)
- All nodes along the path A, B, C, Z send state tables to X
- X initializes its state using this information
- X sends its state to concerned nodes

State Initialization

- X borrows A's Neighborhood Set
  - X_0 set to A_0
  - X_1 set to $B_{1,}$
  - X_2 set to $C_{2,}$
- X's leaf set derived from Z's leaf set



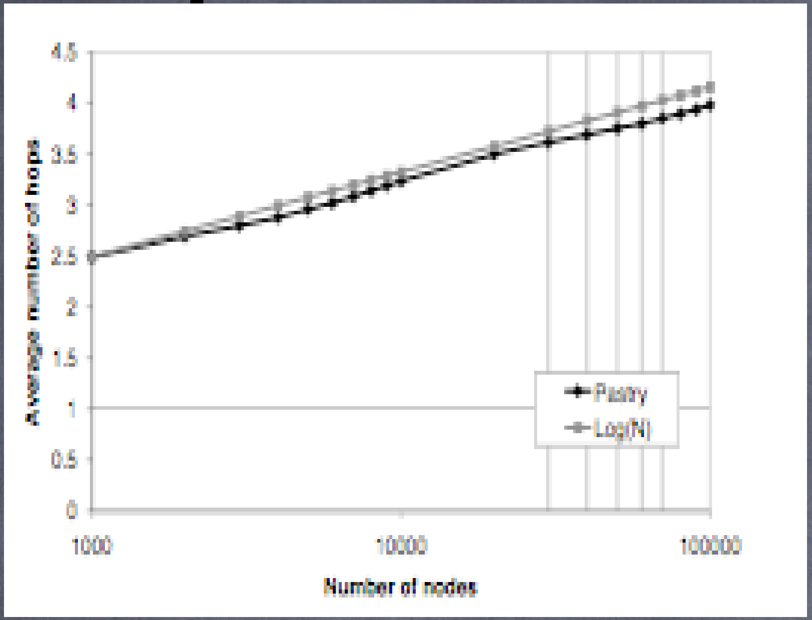
Self-organization: Node Failure

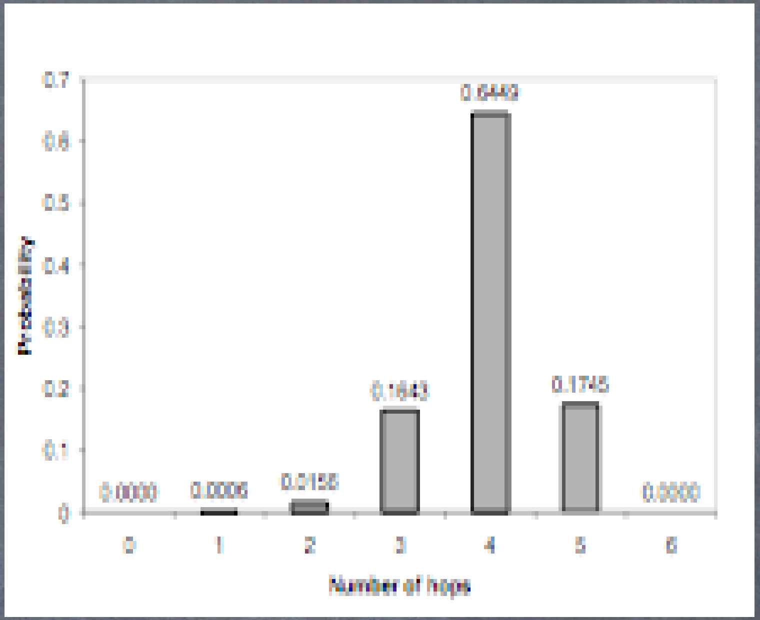
- Detected when a live node tries to contact a failed node
- Updating Leaf set
  - Asks the neighbor Node with largest index on the side of the failed node.
- Updating routing table
  - Node contacts other Nodes in the same row for an entry of the failed Node.

Locality

- Application provides the “distance” function, less distance is more desirable.
- Invariant: “All routing table entries refer to a node that is near the present node, according to the proximity metric, among all live nodes with an appropriate prefix”.
- Invariant maintained on self-organization.

Experimental results: I



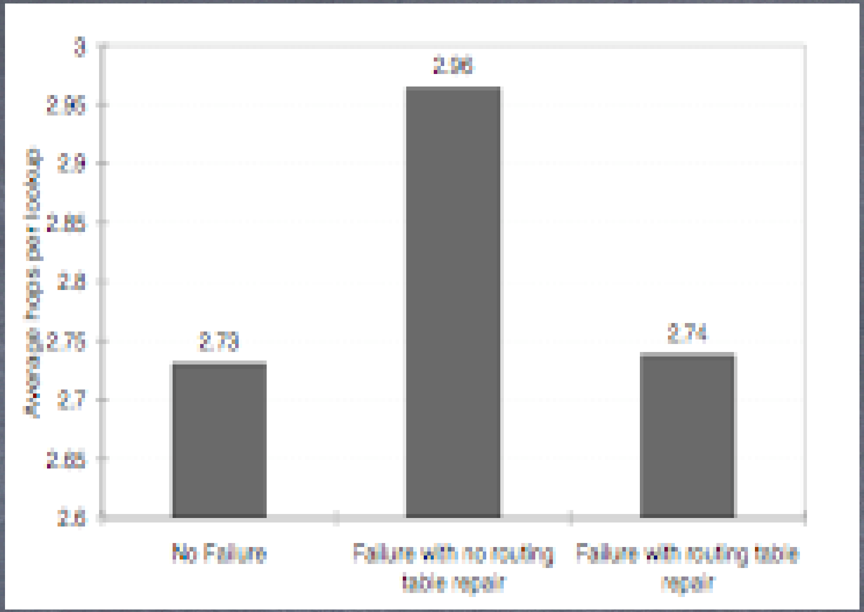


- L=16,M=32

[3]

- Number of nodes vary from 1,000 to 100,000.
- 200,000 trials - 2 nodes are selected randomly, and a message is routed between.
- Results:
  - Fig 1: Expected number of routing steps is **O(log N)**;
  - Fig 2: maximum route length is [$\log_{2^b} N$] (for N=100,000) = 5.

Experimental results: II



[3]

- L=16,M=32,*k*=5, N=5,000, 10% (500) randomly selected nodes fail silently.
- 2 nodes are chosen randomly, a message is routed between these 2 nodes to 200,000 lookups

Summary and Application

- Pastry is self-organizing, completely decentralized, scalable and reliable for routing a message.
- Routes to any node in the overlay network in O (logN) steps.
- Has locality properties, and maintain Neighboring and Leaf set which could be used for job replication and fault recovery
- Building block in construction.

Load Balancing - Research Paper

- "Dynamic Load Balancing in Parallel Processing on Non-Homogeneous Clusters"
    - De Guisti A. E., Naiouf M. R., De Giusti L. C., Chichizola F.

Load Balancing - Problems

- How do you distribute parallel processing tasks across a cluster of non-homogeneous nodes?
  - What methods are possible?
  - Which methods give the best performance?
    - Under what circumstances?

Load Balancing - Experiments

- Considers two general types of load balancing:
  - Static - Workload is divided up before processing
  - Dynamic - Workload may be adjusted during processing

Load Balancing - Experiments

- Direct Static Distribution (DSD)
  - Each node gets the same amount of work
- Predictive Static Distribution (PSD)
  - Each node gets an amount of work proportional to its computing power
- Dynamic Distribution upon Demand (DDD)
  - Each node demands work as needed

Load Balancing - Experiments

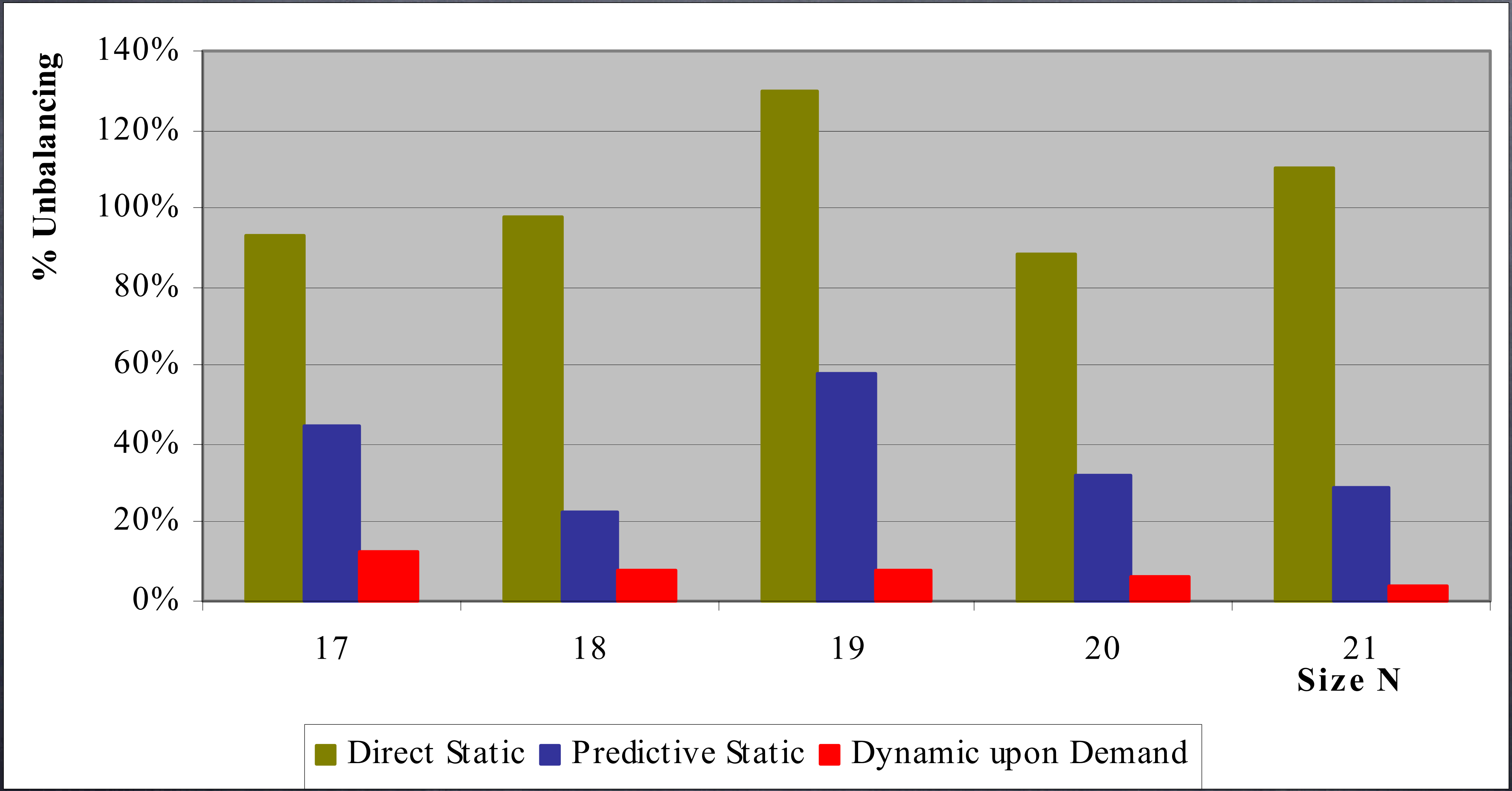
- 3 clusters of 8 compute nodes
- Each cluster contains a different type of node
- Performed a sample parallel problem with all 3 forms of load balancing

Load Balancing - Metrics

$$Unbalance = \frac{\max_{i=1..B}(W_i) - \min_{i=1..B}(W_i)}{\operatorname{avg}_{i=1..B}(W_i)}$$

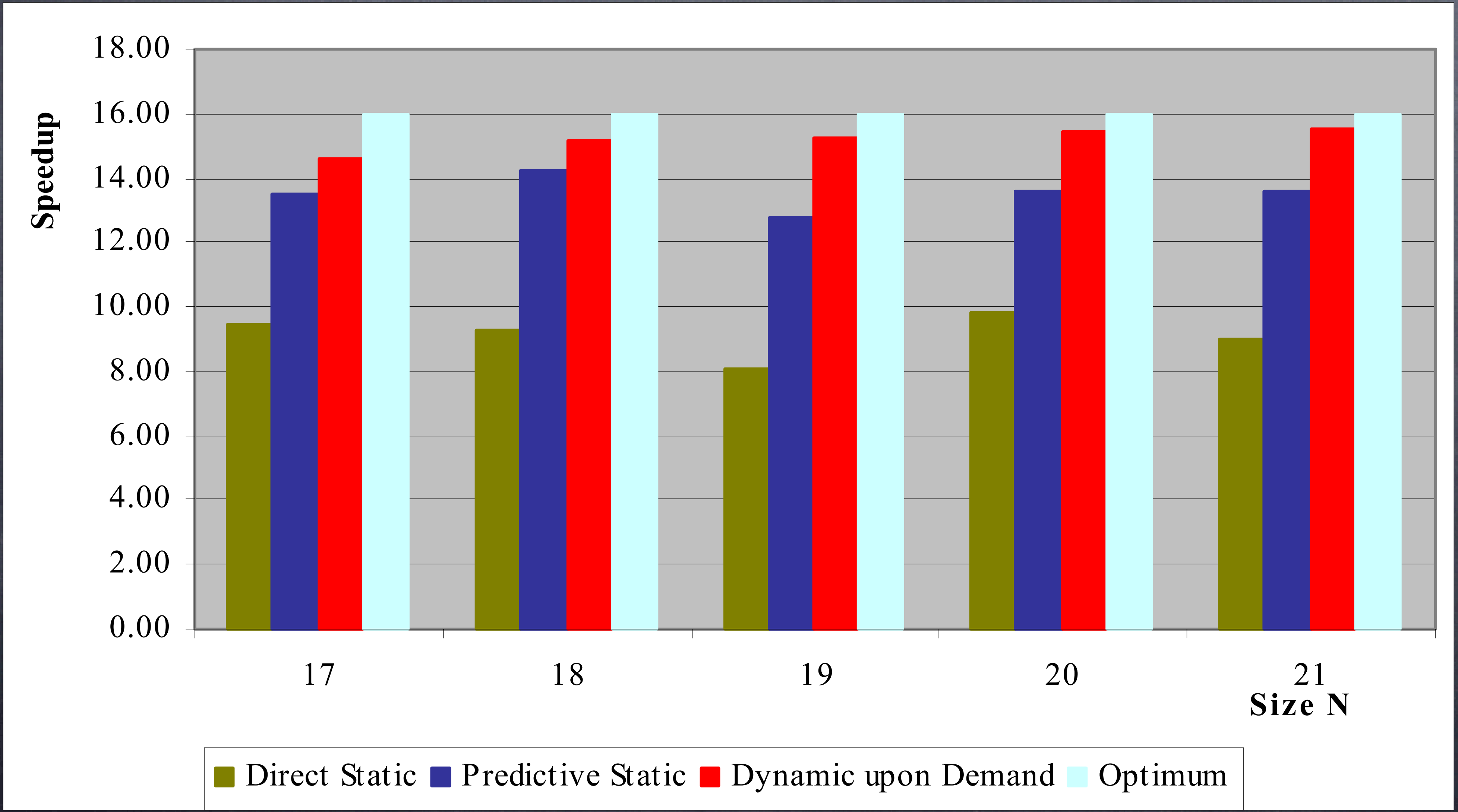
$$Speedup = \frac{SequentialTime}{ParallelTime}$$

Load Balancing - Results



[1]

Load Balancing - Results



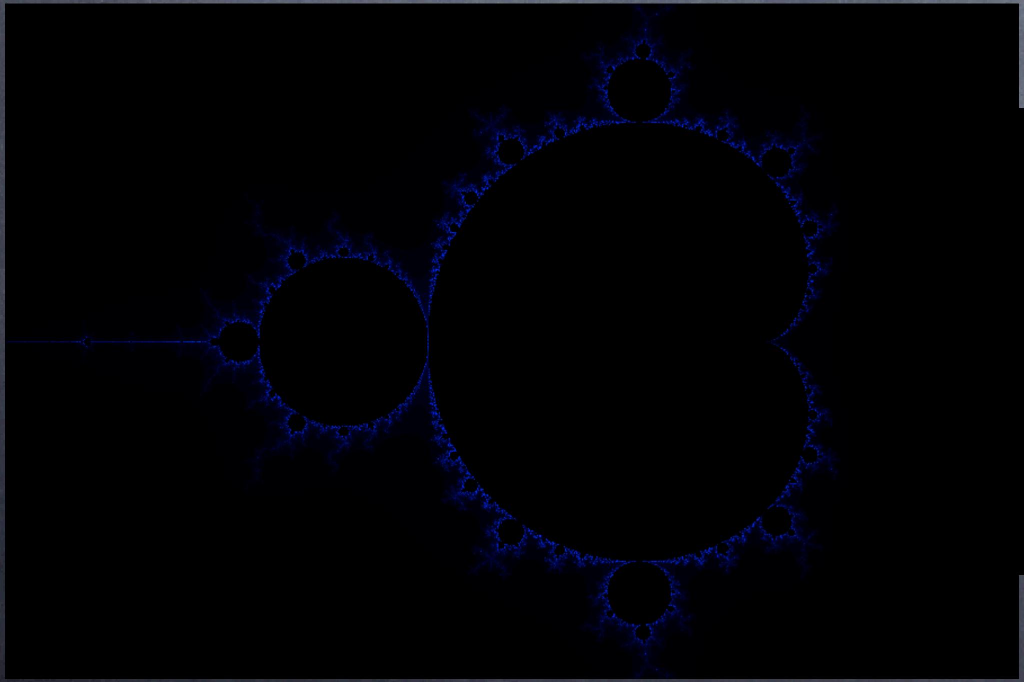
[1]

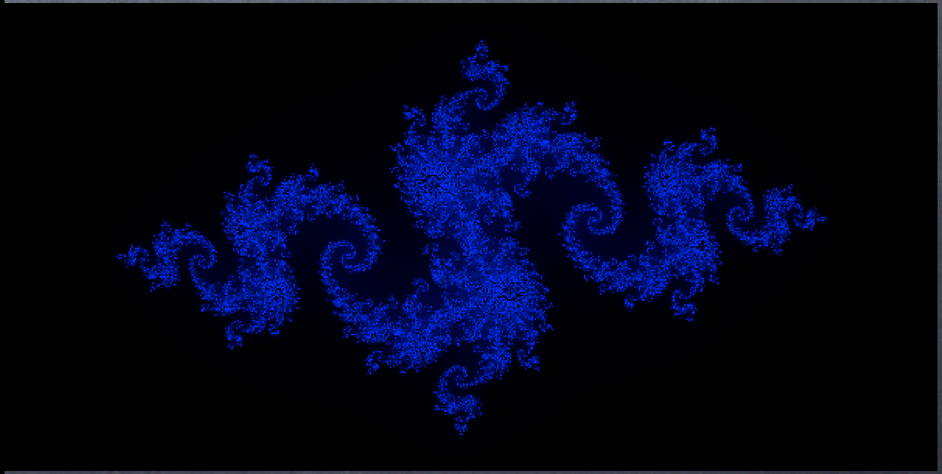
Load Balancing

- Plan to use a demand-driven scheme
- Modify the algorithm to address the problems that the network characteristics pose
  - Job owner (master) may change
  - Nodes may drop out or become available

Progress

- Constructed a simple test application which generates fractal images





References

(1) "Dynamic Load Balancing in Parallel Processing on Non-Homogeneous Clusters". De Guisti A. E., Naiouf M. R., De Giusti L. C., Chichizola F. JCS&T Vol. 5, No 4. December, 2005.

(2) D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, "Peer-to-Peer Computing". HP Labratories, Palo Alto, March, 2002.

(3) A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.