Massively Parallel Computing on Peer-to-Peer Networks

Team Timeout
Jon Ludwig
Prashant Gahlowt
Young Suk Moon

Agenda

- Summary of Presentation I and II
- Software Design and Architecture
- Pastry code and Demo
- Application code and Demo

Summary of Overall Project

- **Application: Distributed rendering of randomized fractal images**
  - Render slices of fractal images in parallel
  - No dependence between pixel values (highly parallelized)
  - Many possible variations on parameters (independent tasks)
- **Underlying Architecture: Pastry**
  - Self Organizing
  - Completely Decentralized
  - Scalable
  - Reliable and Fault Resilient

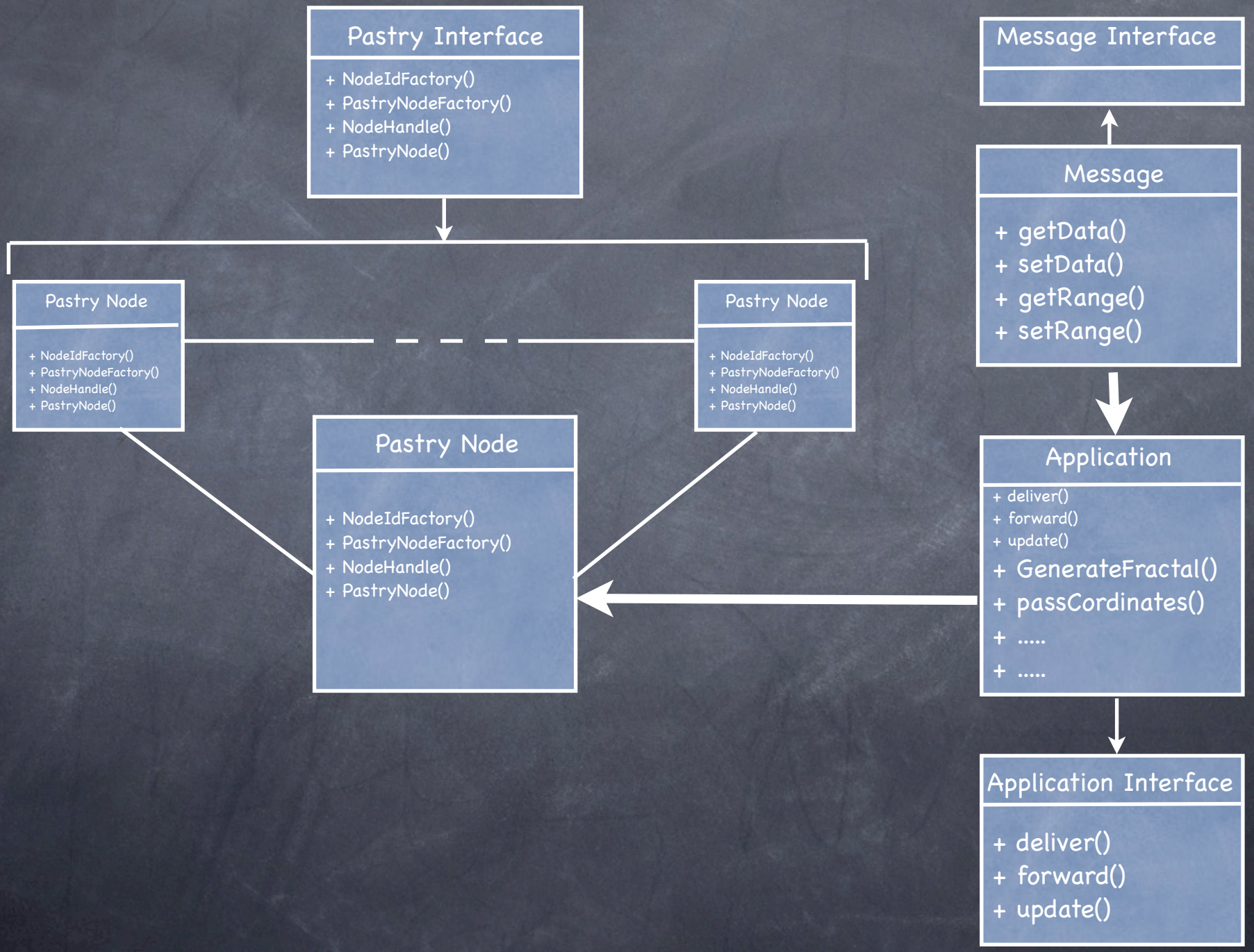
Summary of Presentation I

- Introduced the Peer-to-Peer Architecture
- Introduced the application and Fractal Images
- A brief overview to Pastry

Summary of Presentation II

- Analyzed some P2P Algorithms
  - Centralized Directory Model
  - Flooded Request Model
  - Document Routing Model
- In-depth analysis of Pastry Architecture
- Analyzed some Load-Balancing techniques and Architectures
  - Static
  - Dynamic

Software Design and Architecture



Pastry Code and Demo

```
// Constructor
public Pastry(int bindport, InetSocketAddress bootaddr, Environment env){
        // Generate the NodeIds Randomly
        NodeIdFactory nidFactory = new RandomNodeIdFactory(env);

        // Construct the PastryNodeFactory
        PastryNodeFactory factory = new SocketPastryNodeFactory(nidFactory, bindport, env);

        // Find a bootstrap node
        NodeHandle bootHandle = ((SocketPastryNodeFactory)factory).getNodeHandle(bootaddr);

        // Create the PastryNode
        PastryNode node = factory.newNode(bootHandle);

        // Wait for the node to fully boot into the Pastry ring
        synchronized(node){
            while(!node.isReady() && !node.joinFailed()){
                node.wait(500);
                if(node.joinFailed()){
                    throw new IOException("Could not join the Pastry ring."+
                                    " Reason:"+node.joinFailedReason());
                }
            }
        }
}
```

Pastry Code and Demo

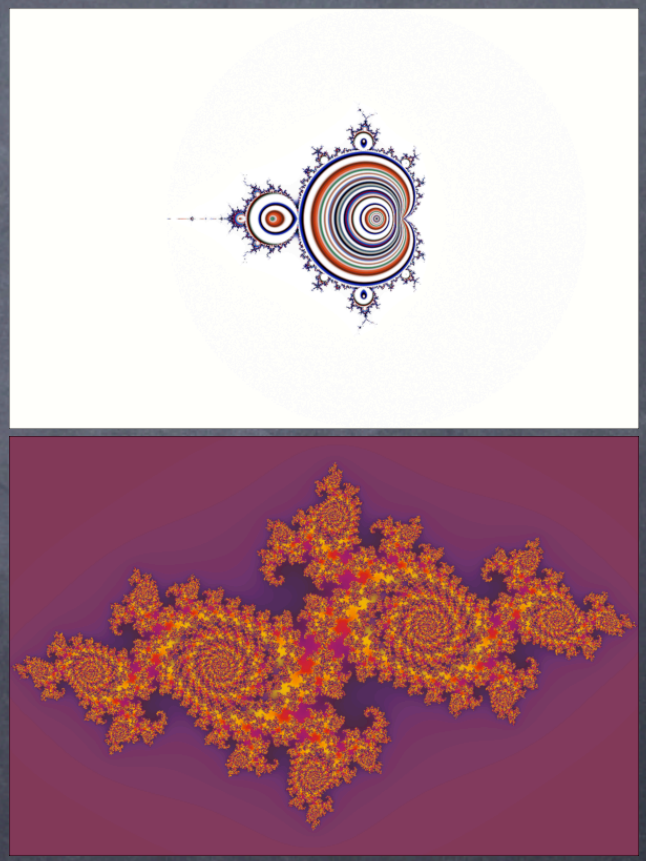
```
public class FractalImageApplication implements Application {
    // Endpoint to send messages
    protected Endpoint endpoint;

    // Message that will be received
    private myMessage receivedMsg;

    // Some example data
    private int x, y;

    public FractalImageApplication(Node node){
        endpoint = node.buildEndpoint(this,"fractalapp");
        endpoint.register();
    }

    public void deliver(Id id, Message message){
            receivedMsg = (myMessage)message;
                ......
    }

    ......

    public void routeMsg(Id destId){
                ......

                Message msg = new myMessage(endpoint.getId(), destId, x, y);
                ......

                endpoint.route(destId, msg, null);
    }
}
```

Pastry Code and Demo

```
public class myMessage implements Message {
    private Id from, to;
    private int x, y;

    public myMessage(Id from, Id to, int x, int y){
        this.from = from;
        this.to = to;
          this.x = x;
          this.y = y;
    }

    public int getPriority(){
        return Message.LOW_PRIORITY;
    }

    ......

    public String getData(){
         return "x = "+x+", y = "+y;
    }
}
```

Parallel Fractal Generation

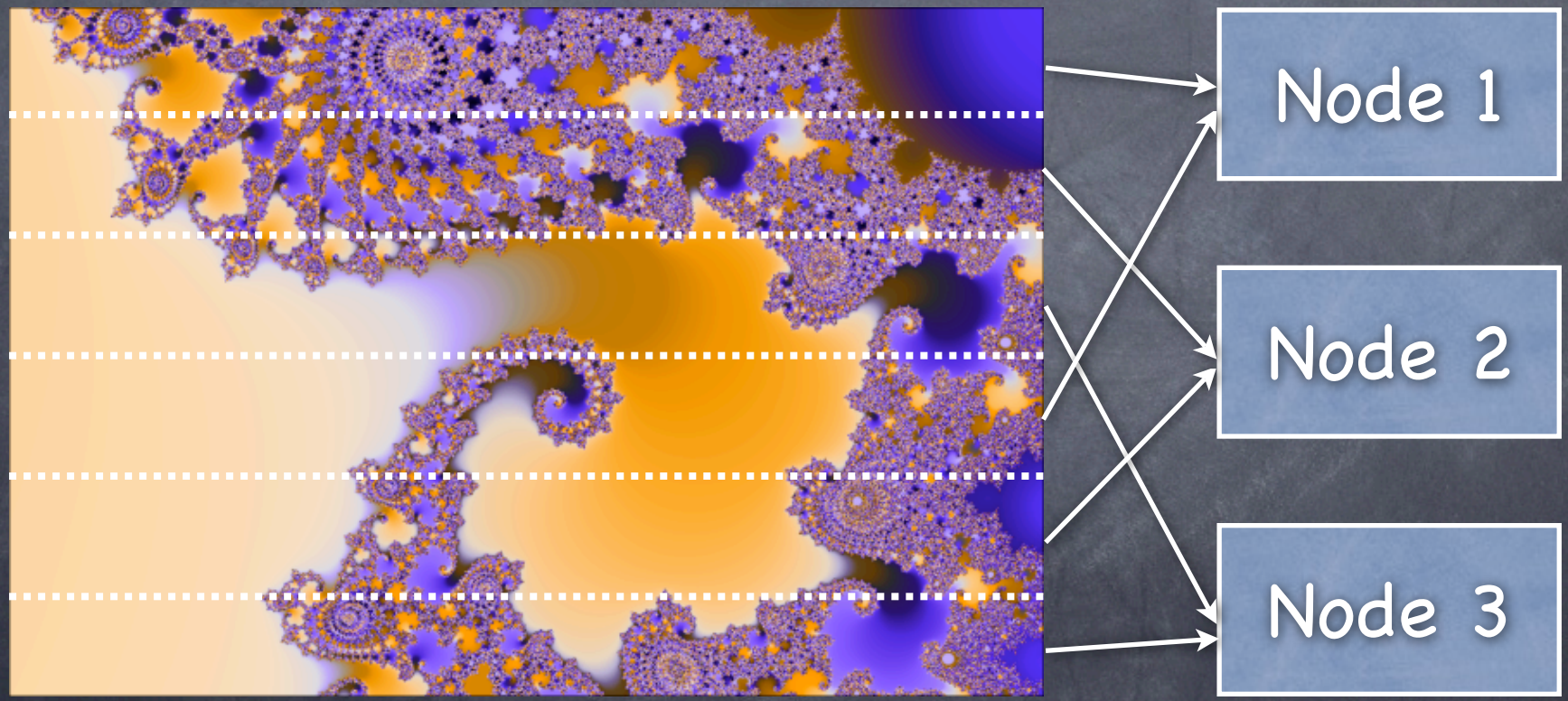
- Two common types of fractal systems
  - Mandelbrot Set
  - Julia Set



Parallel Fractal Generation

- The value of each pixel is calculated independently of other pixels.
- Calculating the value of each pixel is computationally intensive and cpu-bound.
- The algorithm requires only a small amount of information to generate pixel values.

Parallel Fractal Generation



Parallel Fractal Generation

- Size of the desired image is known.
- Divide the image space into a set of pixel ranges.
- Distribute each pixel range to a node in the network.
- Gather the generated pixel values from the nodes into the final image space.

Distributed Fractal Generation

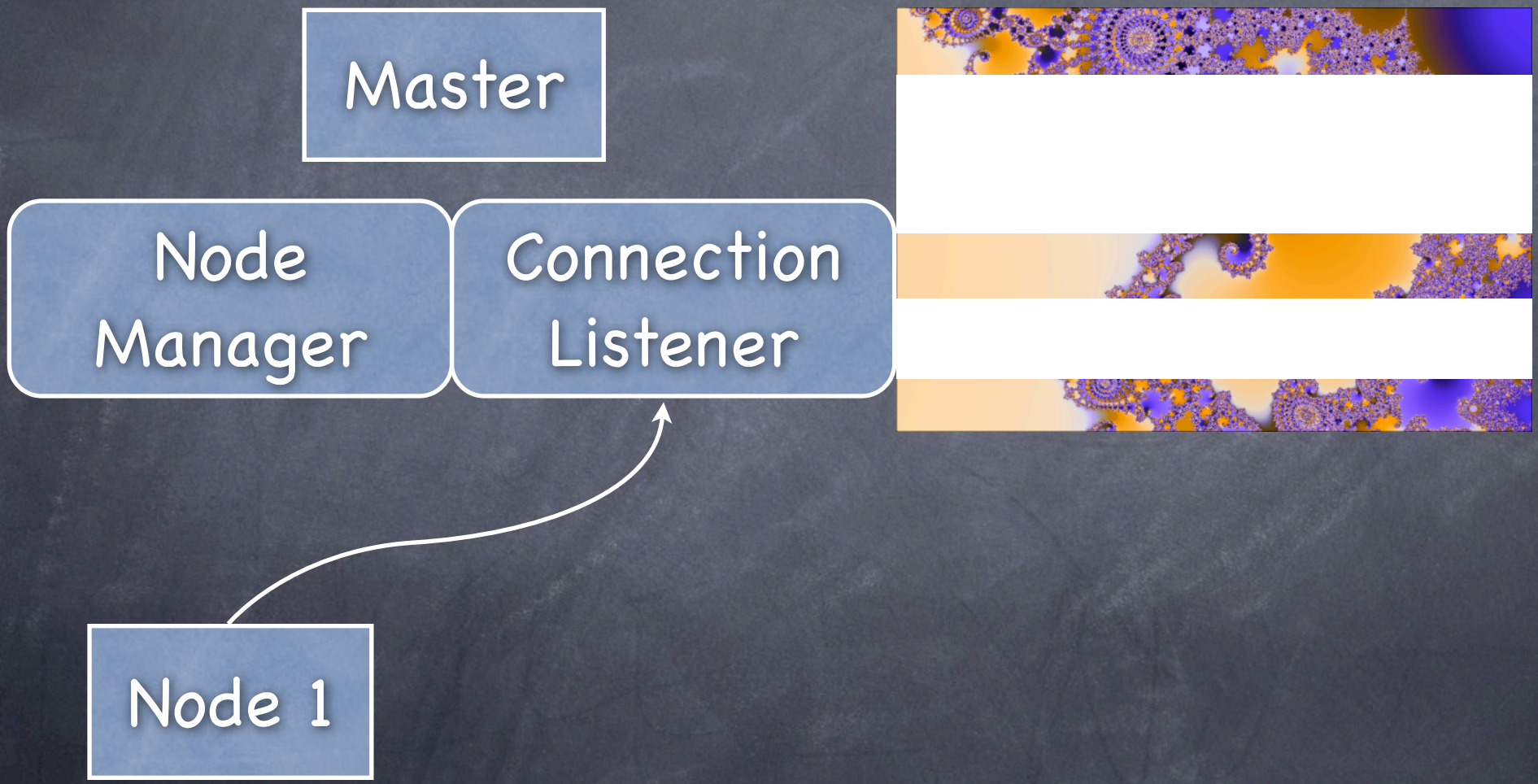
Master

Node Manager

Connection Listener

Node 1

Distributed Fractal Generation



Distributed Fractal Generation

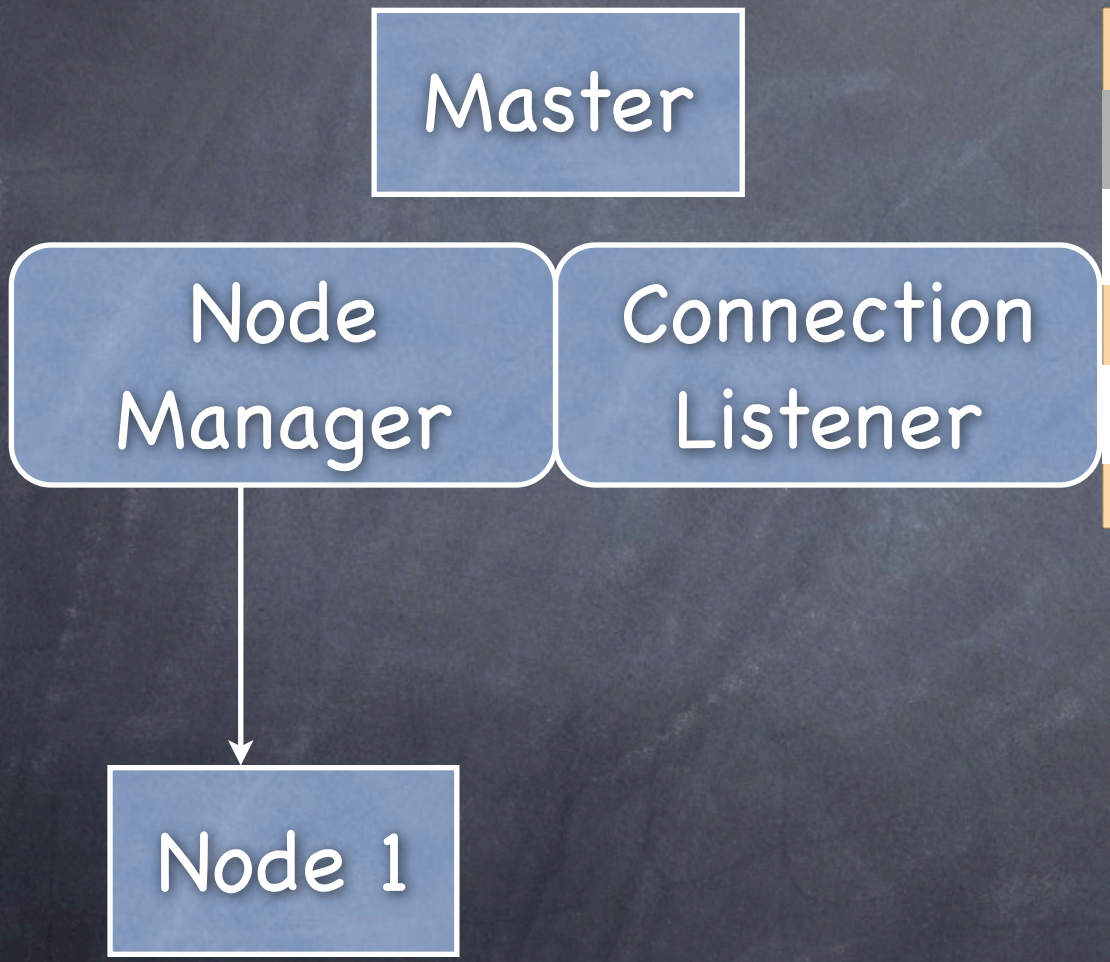
Master

Node Manager

Connection Listener

Node 1

Distributed Fractal Generation



Distributed Fractal Generation

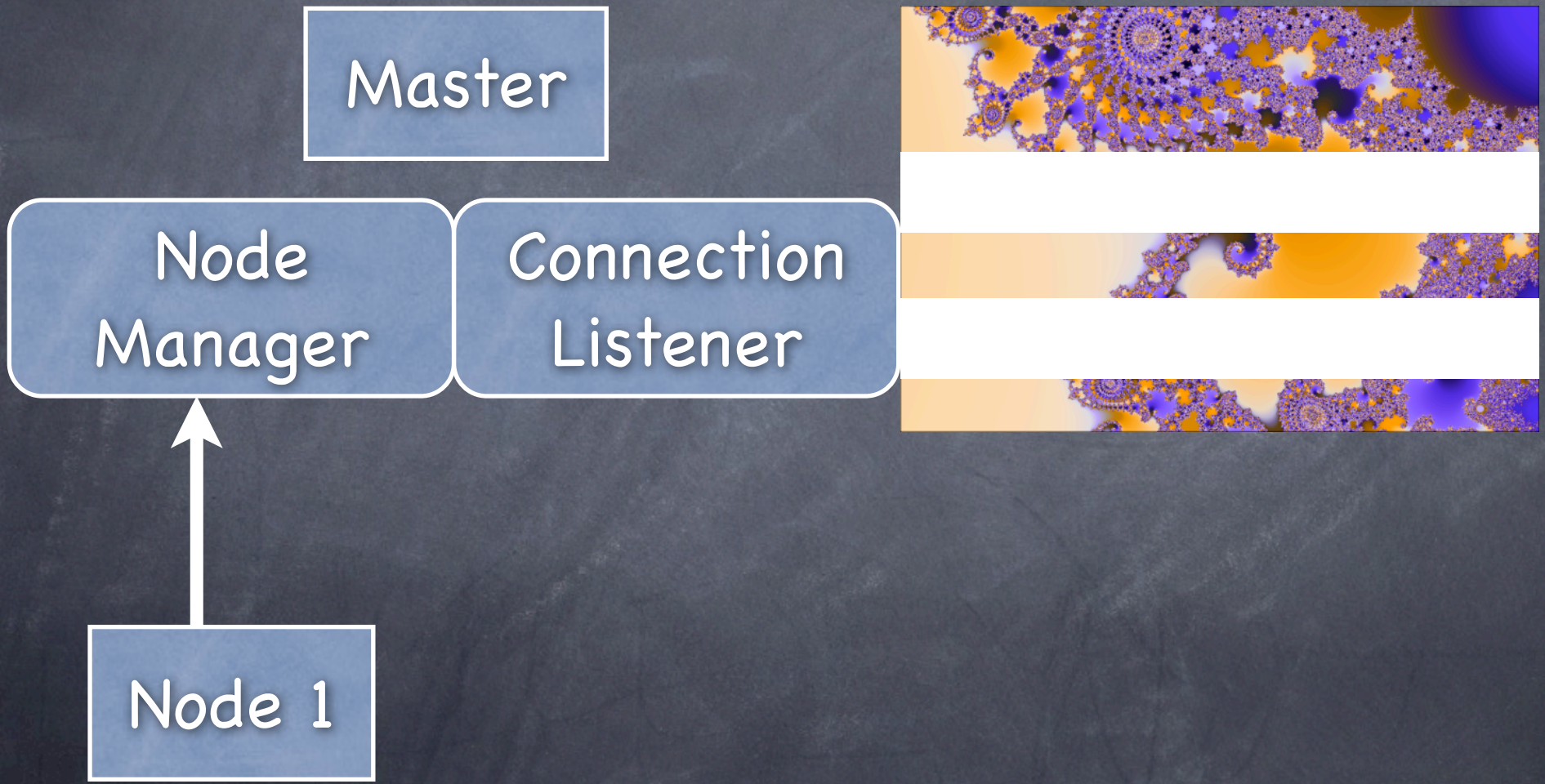
Master

Node Manager

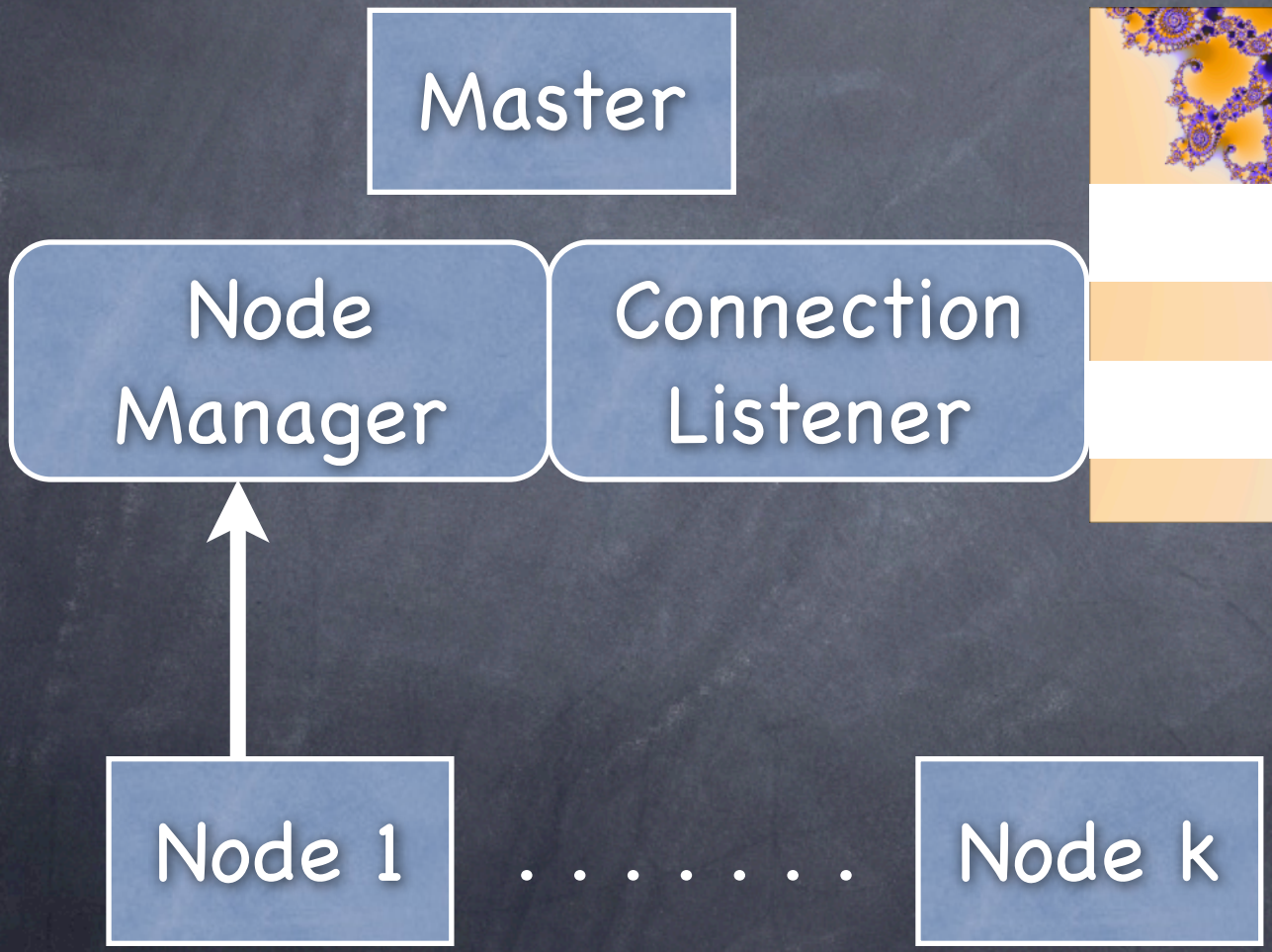
Connection Listener

Node 1

Distributed Fractal Generation



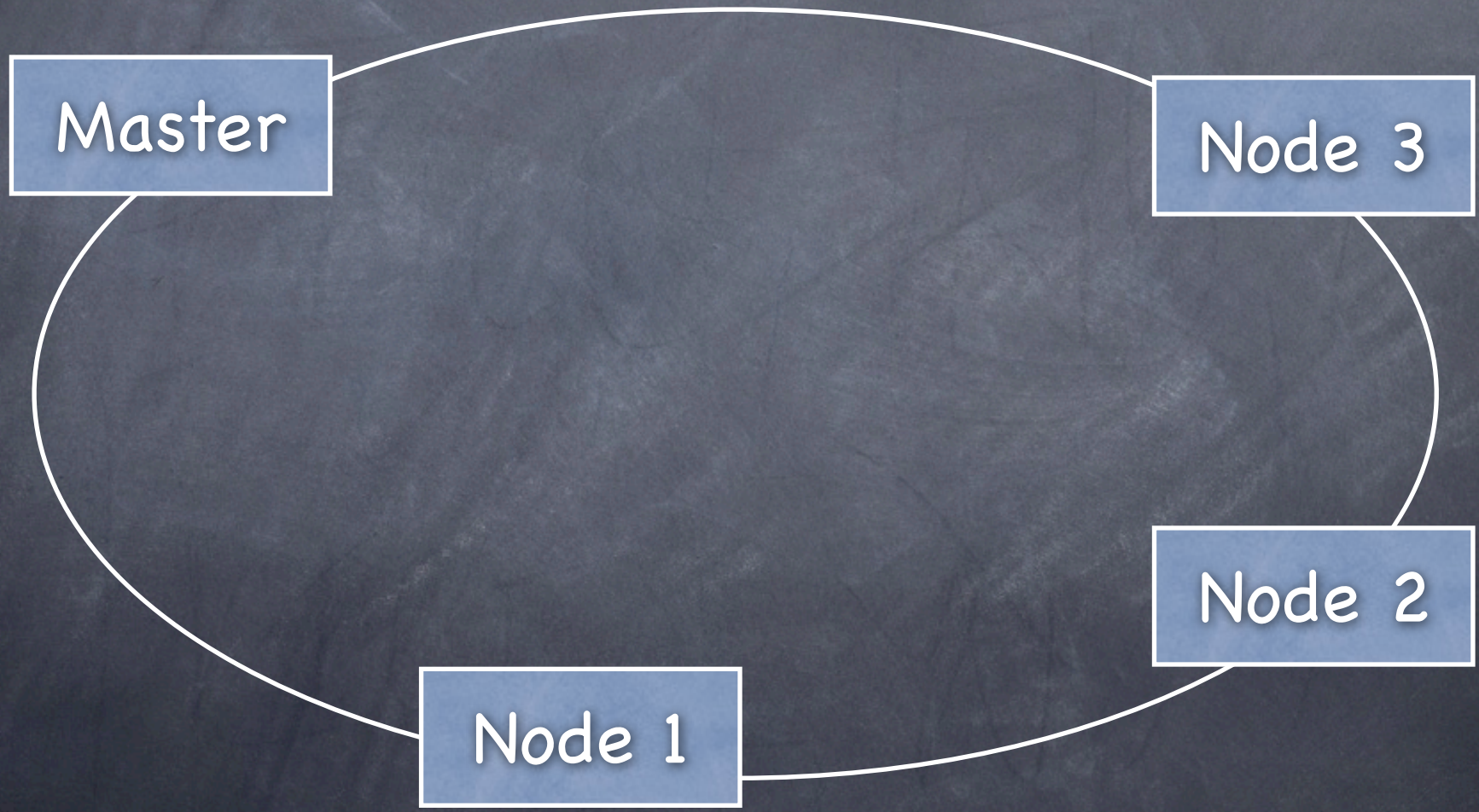
Distributed Fractal Generation



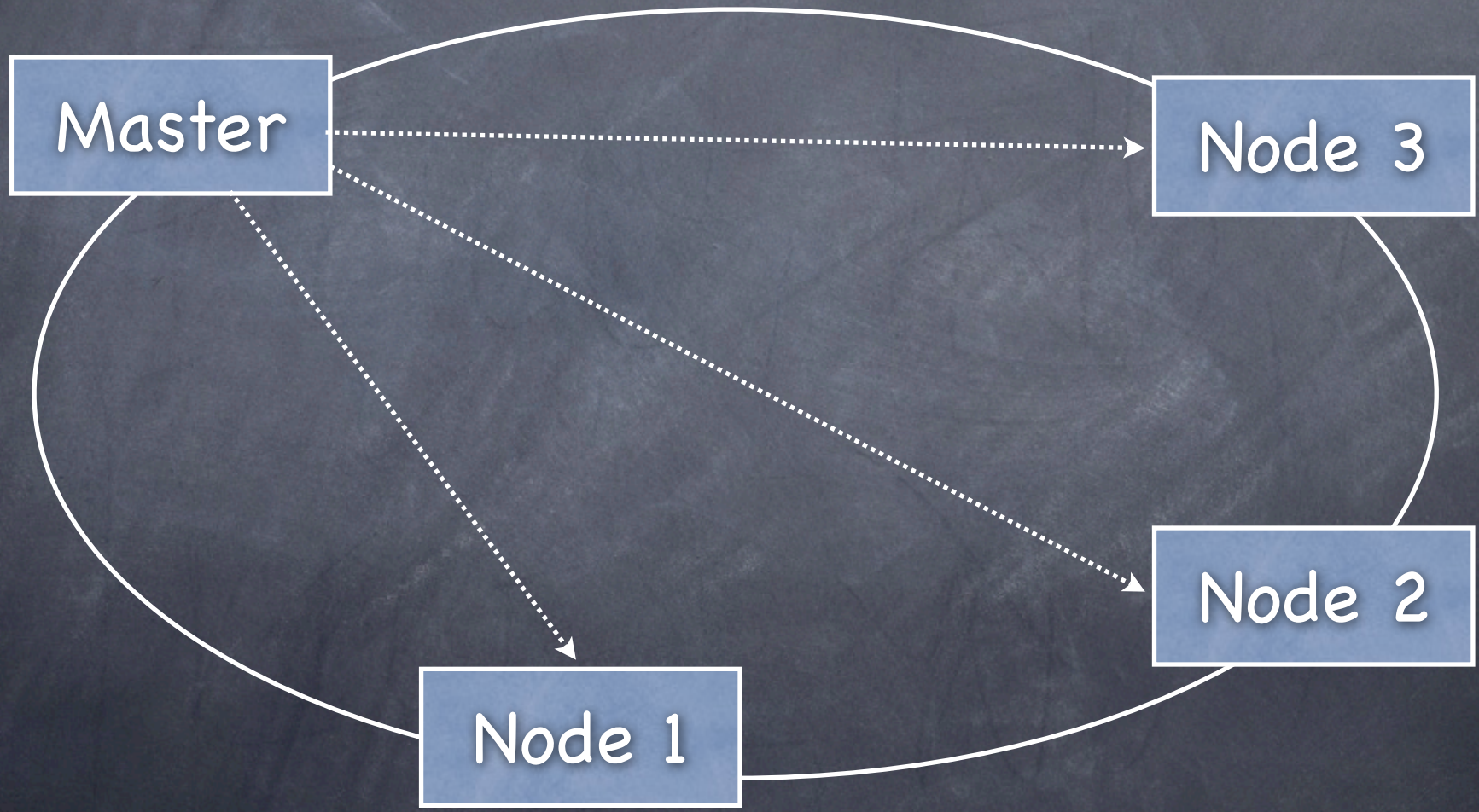
Distributed Fractal Generation

- Sequences of fractal images can be compiled into animations...

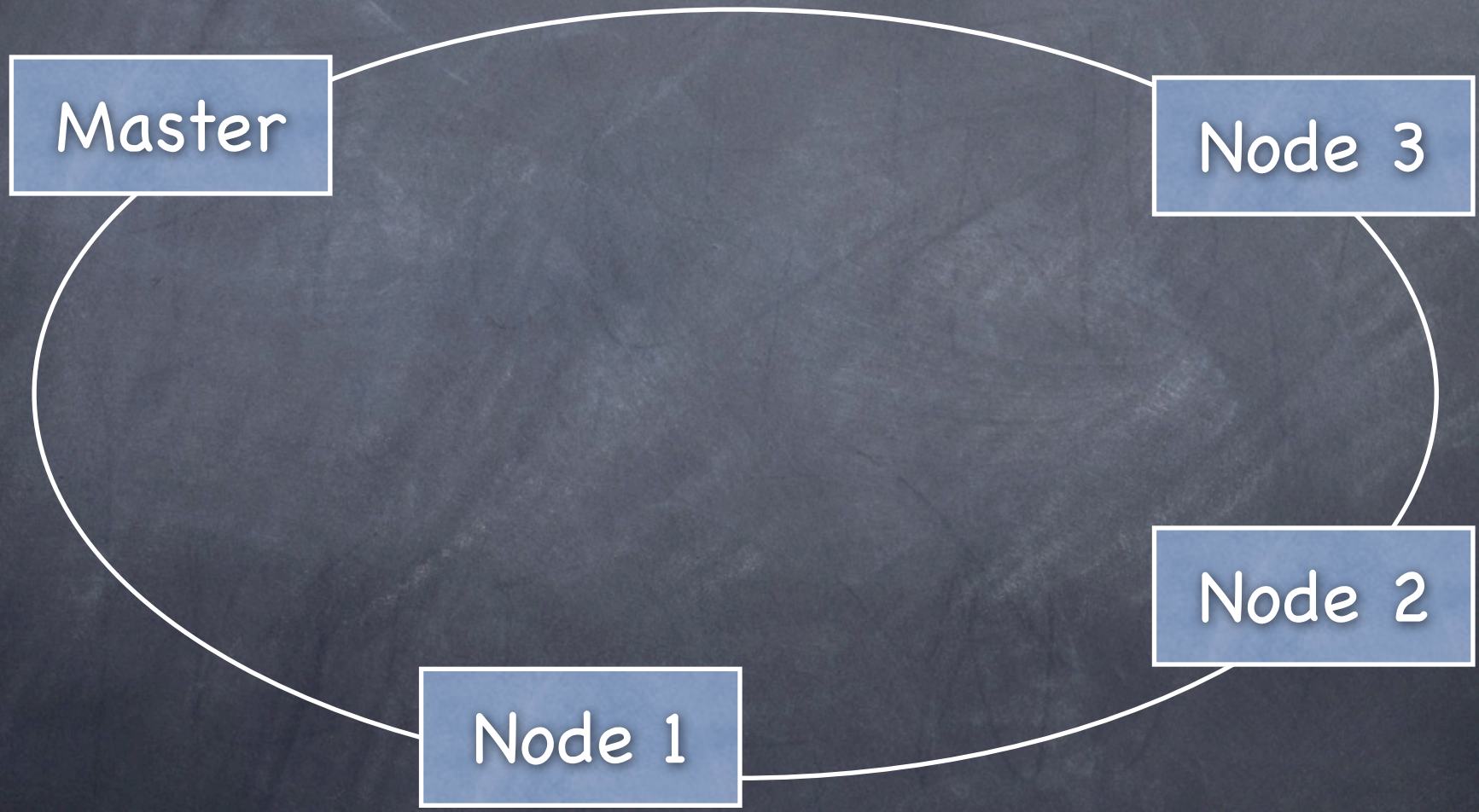
Distributed Fractal Generation with Pastry



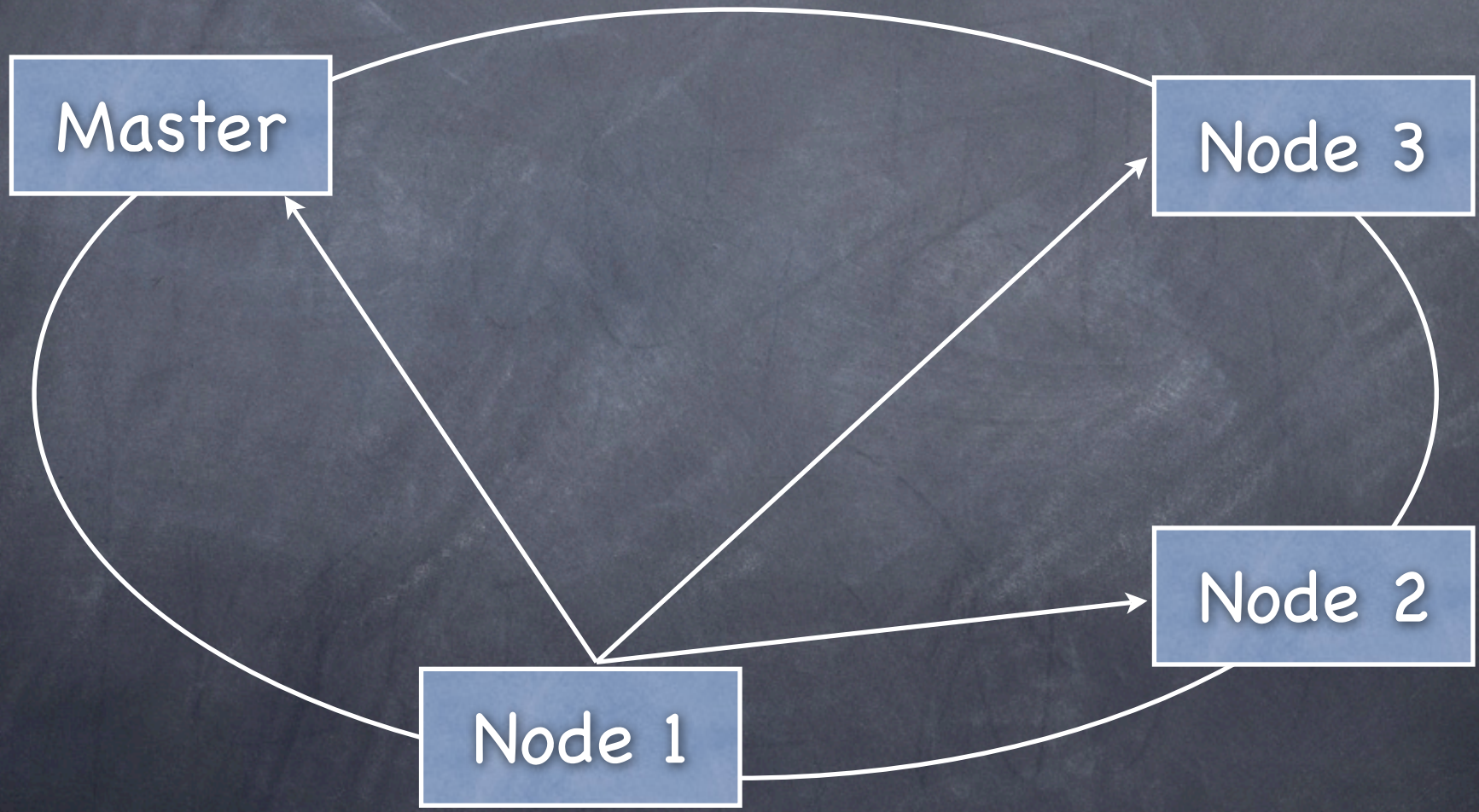
Distributed Fractal Generation with Pastry



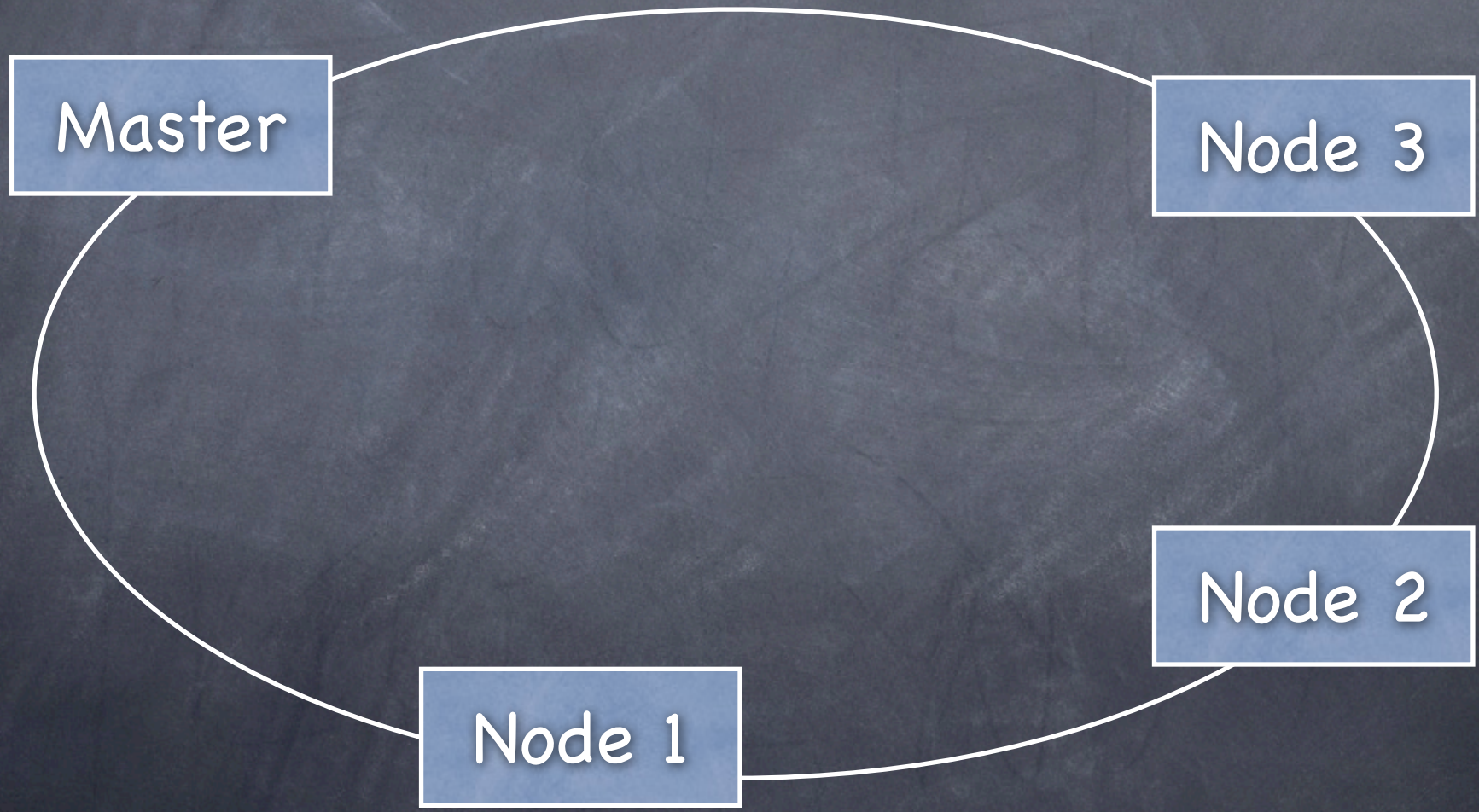
Distributed Fractal Generation with Pastry



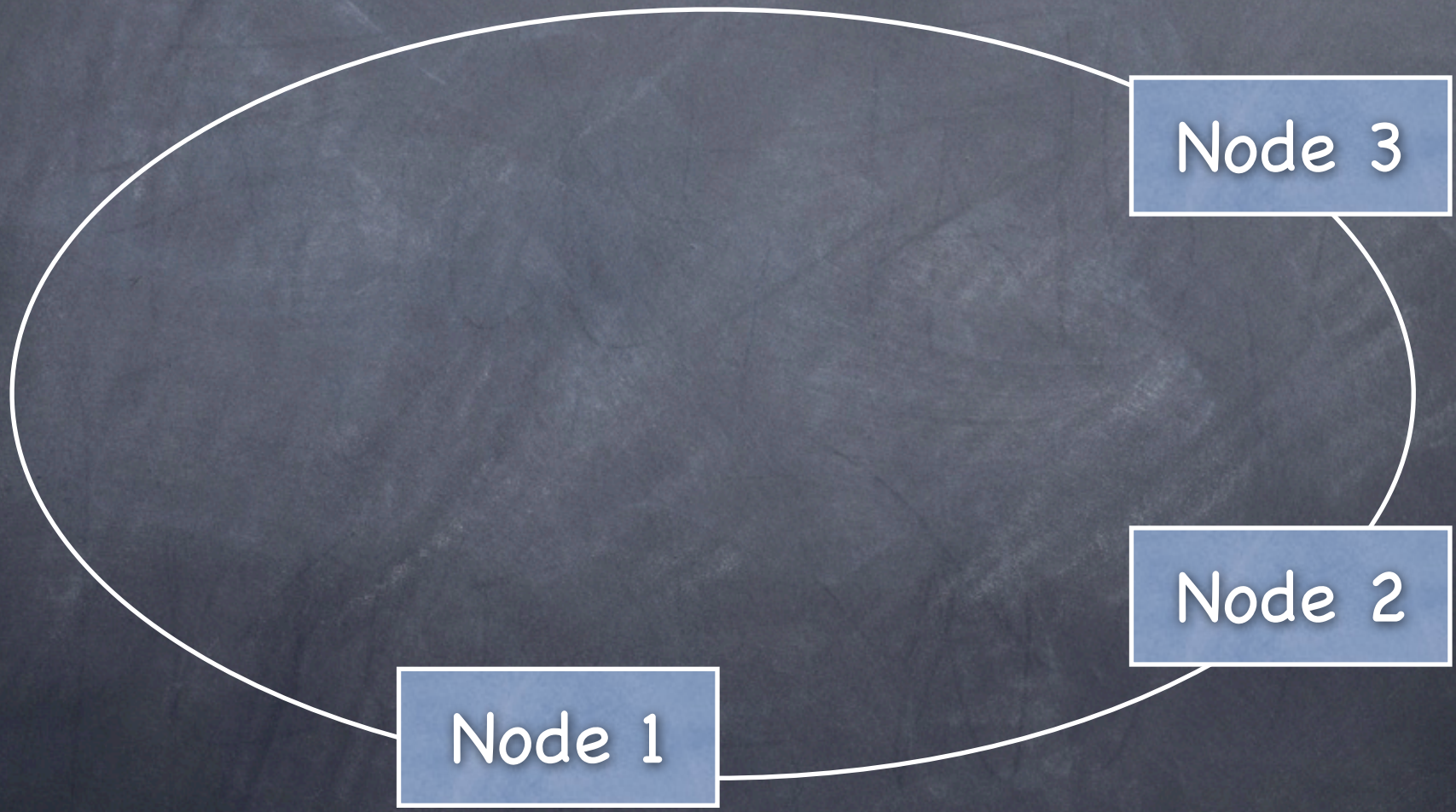
Distributed Fractal Generation with Pastry



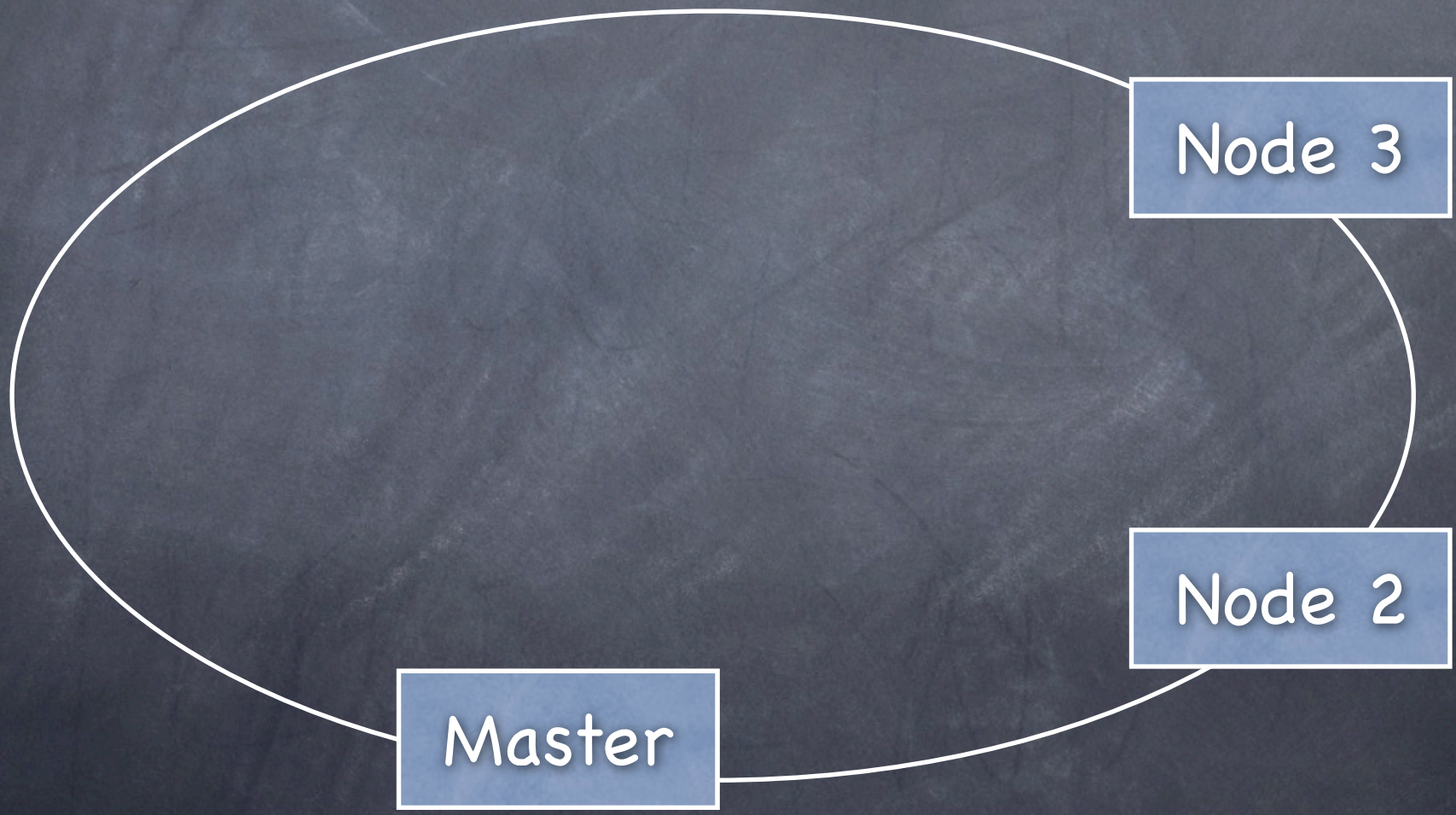
Distributed Fractal Generation with Pastry



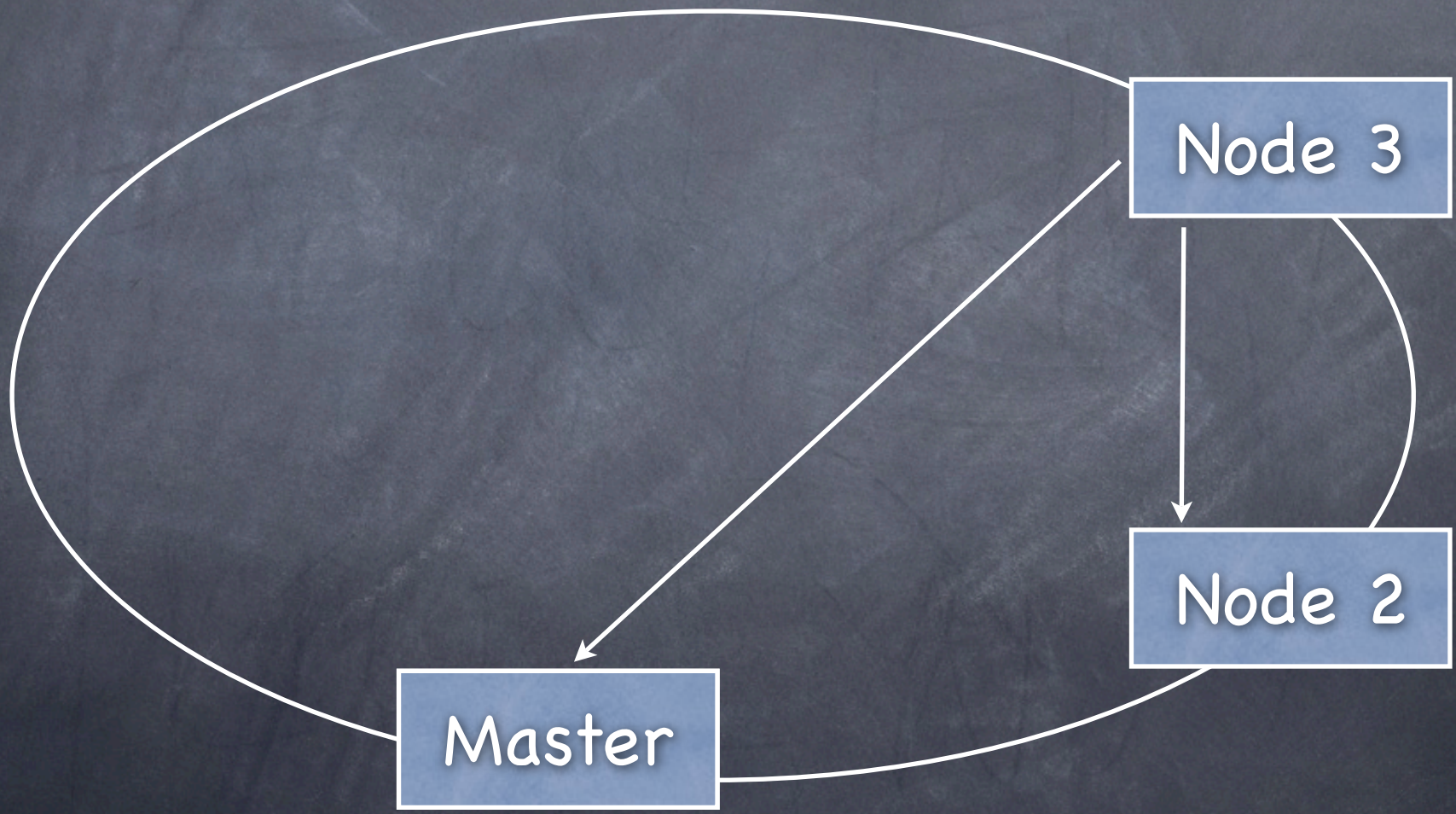
Distributed Fractal Generation with Pastry



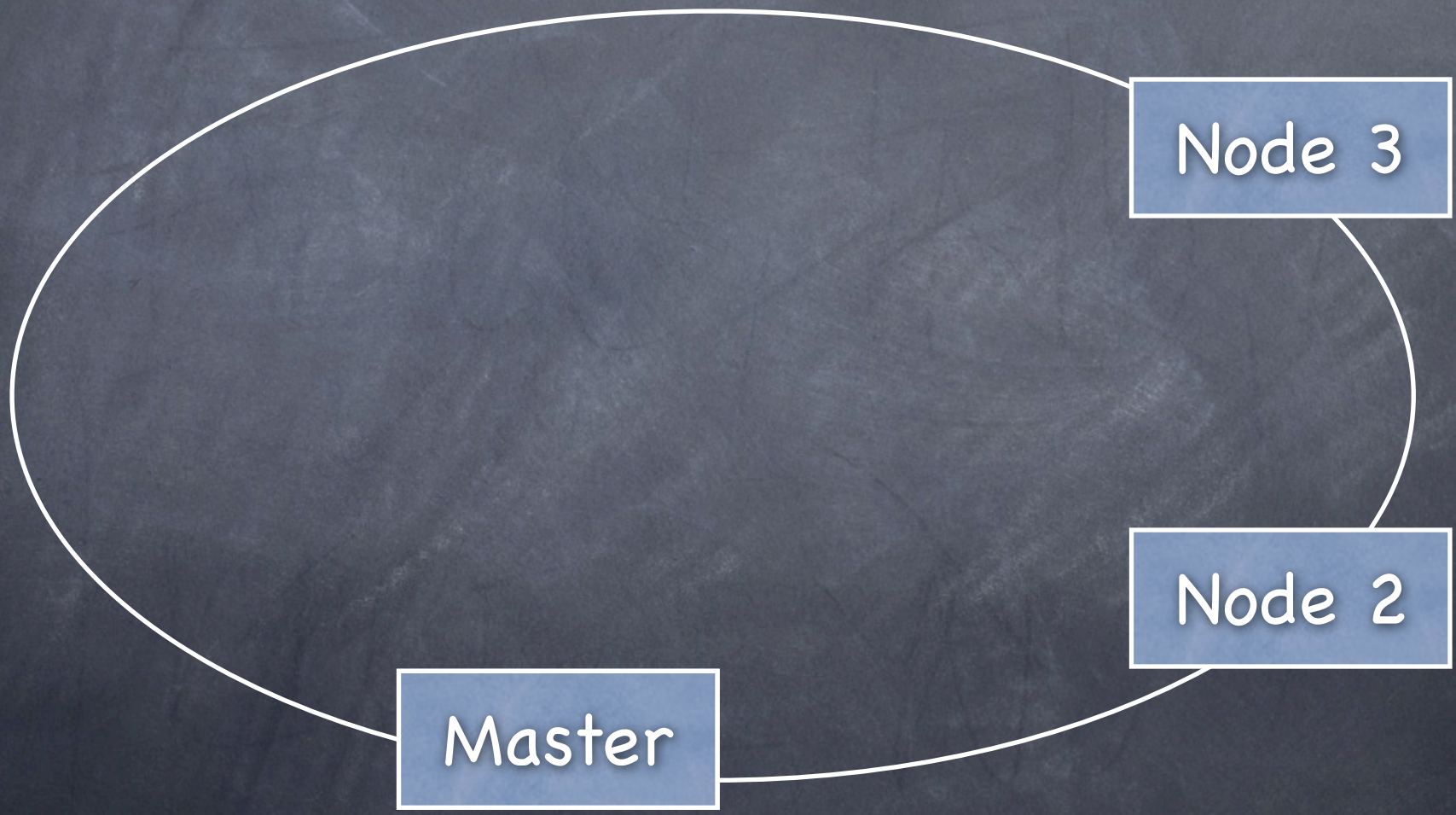
Distributed Fractal Generation with Pastry



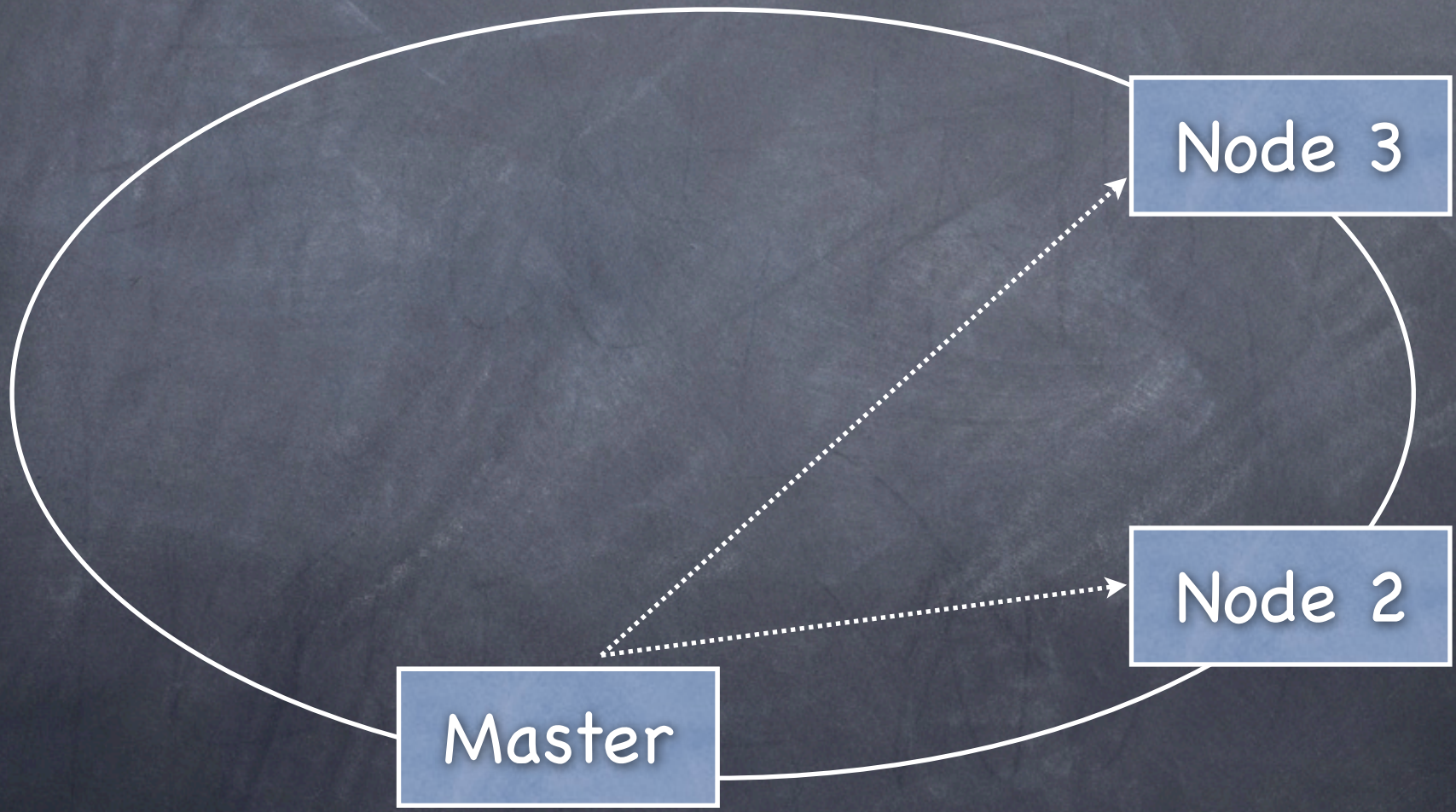
Distributed Fractal Generation with Pastry



Distributed Fractal Generation with Pastry



Distributed Fractal Generation with Pastry



Work in Progress

- Currently working on:
  - Implementing the distributed fractal generation application on top of Pastry.
  - Notification of failed nodes.
  - Migration of roles (master/worker).
  - High-bandwidth multicast using Scribe.

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.