P2P-MPI main features

Communication Library

- Runs Java programs (OS independent)
- MPJ compliant communication library (passes the JavaGrande Forum Benchmark)
- Single or Multiple ports implementations available
- Fault-tolerance using replication of processes (no checkpoint server)
- Transparent file staging for execution

Middleware

- Peer-to-peer network of resources
- Transparent resource selection and allocation mechanism
- Simple strategies to guide allocation
- Platform visualization tool
- Failure detection in bounded time O(log(n))

P2P-MPI

P2P-MPI is an integrated development and execution environment for message-passing parallel programs.

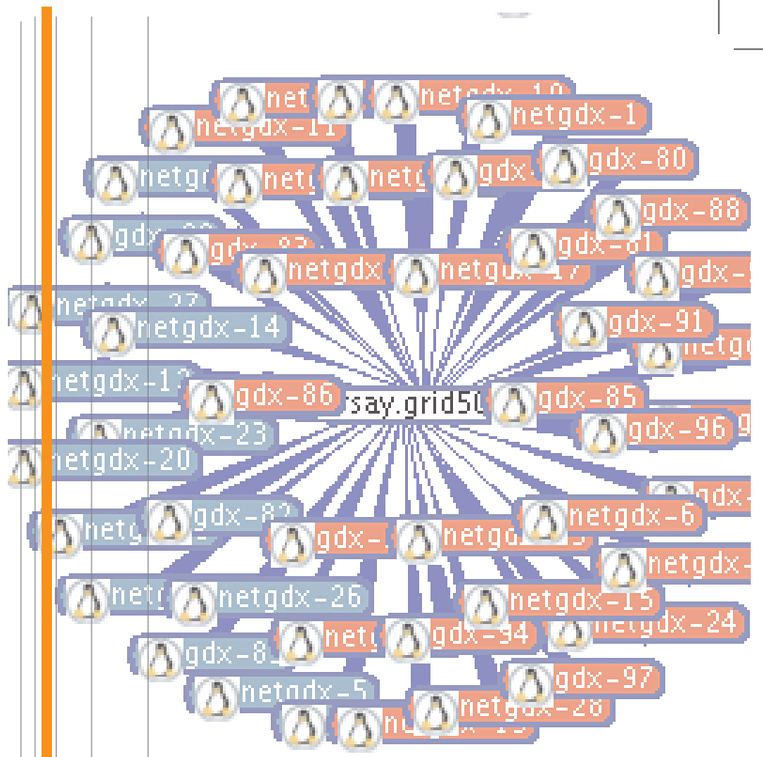
The environment supports MPJ (Message Passing for Java), the pendant of MPI for Java. The framework allows users to register within a network of P2P-MPI started computers.
The system is organized as a peer-to-peer network: a user joining the network has access to others' CPUs and accepts to share its own CPU for parallel program executions.

ALGORILLE Research Team

http://www.loria.fr/equipes/algorille

INRIA Nancy – Grand Est
Campus Scientifique
615 rue du jardin botanique
54600 Villers-lès-Nancy
FRANCE

http://www.inria.fr/nancy/



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

P2P-MPI

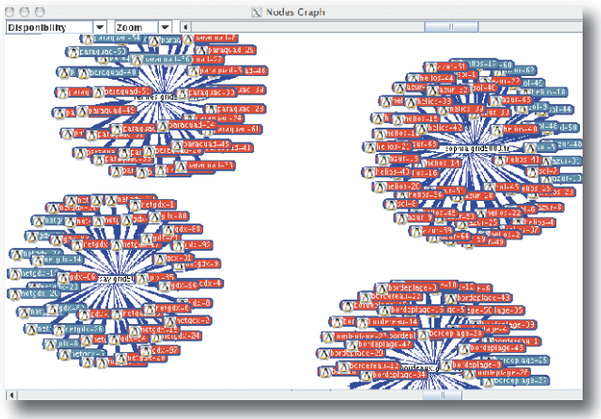
Integrated middleware and communication library for message passing parallel programs

Scalability

The system can scale up to hundreds of nodes, and over geographically distributed areas. Application deployments have been successfully tested using up to 600 processes.

Thanks to its P2P based resource organization the system has no single point of failures regarding resource discovery.

The fault-detection service is based on the principle of failure detectors (gossiping) in order to scale. The user can choose between two deterministic protocols (a quicker and a safer) which both guarantee that a failure is detected in a bounded and known time, logarithmic in the number of processes used.



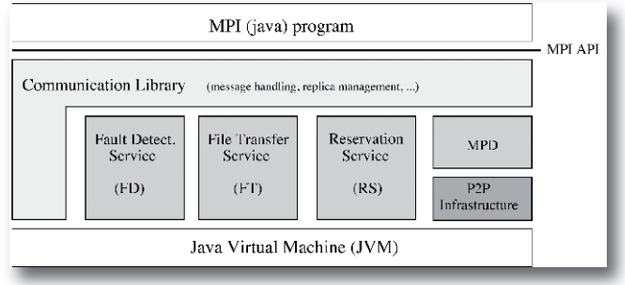
Fault-tolerance

```
import p2pmpi.mpi.*;

public class Pi {
        public static void main(String[] args) {
        int rank, size, i;
        double PI25DT = 3.14159265358979323846264 3;
        double h, sum, x;
        MPI.Init(args);
        size = MPI.COMM_WORLD.Size();
        rank = MPI.COMM_WORLD.Rank();
        int[] n = new int[1];
        double[] mypi = new double[1];
        double[] pi   = new double[1];

        if(rank == 0)
           n[0] = 1000000; //number of intervals
        MPI.COMM_WORLD.Bcast(n, 0, 1, MPI.INT, 0);
        h = 1.0 / (double)n[0];
        sum = 0.0;
        for(i = rank + 1; i <= n[0]; i+= size) {
           x = h * ((double)i - 0.5);
           sum += (4.0/(1.0 + x*x));
        }
        mypi[0] = h * sum;
        MPI.COMM_WORLD.Reduce(mypi, 0, pi, 0, 1, MPI.DOUBLE, MPI.SUM, 0);
        if(rank == 0) {
           System.out.println("Pi is approximately " + pi[0]);
           System.out.println("Error is " + (pi[0] - PI25DT));
        }
        MPI.Finalize();
    }
}
```

An example MPJ program for computing an approximation of π

The communication library implements fault-tolerance through replication of processes. A number of copies of each process may be asked to run simultaneously at runtime.
So, contrarily to an MPI application that crashes as soon as any of its processes crash, a P2P-MPI using replication will be able to continue as long as at least one copy of each process is running.

Resource Discovery



Contrarily to most MPI implementations, P2P-MPI does not assume a static description of resources. P2P-MPI has adopted a peer-to-peer architecture to adapt to the volatility of resources.
When a user joins the P2P-MPI grid its CPU becomes available to others. At each job request, the middleware handles a discovery of available resources.

A subset of available resources is chosen, possibly guided by simple strategies indicated by the user.