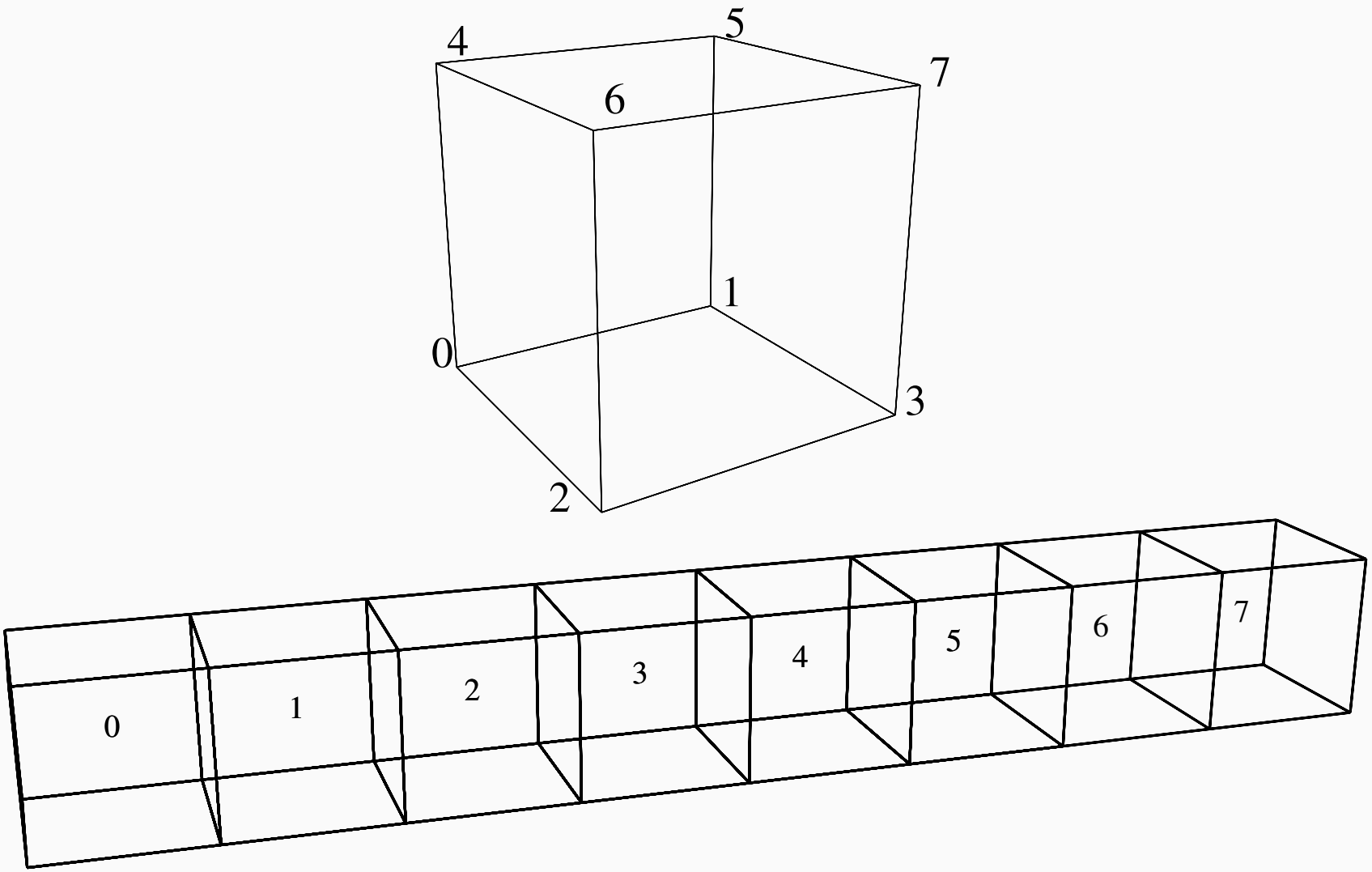
Initialization and finalization of INMOST modules

```
#include "inmost.h"
using namespace INMOST;
int main(int argc, char *argv[])
{
  // Initialize INMOST activities
  Solver::Initialize(&argc,&argv,"");
  Mesh::Initialize(&argc,&argv);
  Partitioner::Initialize(&argc,&argv);
  ...
  // Finalize INMOST activities
  Partitioner::Finalize();
  Solver::Finalize();
  Mesh::Finalize();
  return 0;
}
```

MINIMALISTIC PARALLEL MESH GENERATION

```
Mesh *m = new Mesh();
int rank = m->GetProcessorRank(); // Get PC rank
int size = m->GetProcessorsNumber(); // Get number of PCs
ElementArray<Node> verts(m); // Create local vertices
for(int k=0; k<2; k++)
  for(int j=0; j<2; j++)
    for(int i=0; i<2; i++) {
      double xyz[3]={rank+i, j, k}; // Define  coordinates
      verts.push_back(m->CreateNode(xyz)); // Add new vertex
    }
// Define face connectivity template for cubic cell
const int face_nodes[24] = {0,4,6,2, 1,3,7,5, 0,1,5,4,
                            2,6,7,3, 0,2,3,1, 4,5,7,6};
const int num_nodes[6]   = {4, 4, 4, 4, 4, 4};
m->CreateCell(verts,face_nodes,num_nodes,6); // Cubic cell
m->ResolveShared(); // Resolve duplicate nodes
m->Save("mesh.pvtk"); // Export mesh to parallel VTK format
delete m;
```

MINIMALISTIC PARALLEL MESH GENERATION



Minimalistic parallel cubic grid

ASSEMBLE MATRIX AND SOLVE LINEAR SYSTEM

```
m->ExchangeGhost(1,FACE);
Mesh::GeomParam table;
table[BARYCENTER] = CELL;
table[MEASURE] = CELL | FACE;
m->PrepareGeometricData(table);
Solver::Matrix A;
Solver::Vector x, b;
...  // Iterate over all cells and compute A and b
Solver S(Solver::K3BIILU2); // Specify BIILU2 solver
S.SetMatrix(A); // Compute the preconditioner
S.Solve(b,x);   // Solve the linear system
Tag phi = m->CreateTag("Solution",DATA_REAL,CELL,NONE,1);
for(Mesh::iteratorCell c = m->BeginCell();
    c != m->EndCell(); ++c)
{
        if( c->GetStatus() != Element::Ghost )
            c->Real(phi) = x[cell->GlobalID()];
}
```

ASSEMBLE MATRIX AND SOLVE LINEAR SYSTEM

```
for(Mesh::iteratorFace f = m->BeginFace();
    f != m->EndFace(); ++f)
{
  Cell r1 = f->BackCell(),  r2 = f->FrontCell();
  ... // Check cell states r1->GetStatus(), r2->GetStatus()
  double area = f->Area();
  double r1_cnt[3], f_cnt[3];
  r1->Barycenter(r1_cnt);  // Get the barycenter of the cell
  face->Barycenter(f_cnt); // Get the barycenter of the face
  ... // Compute matrix coefficients
  // Get global IDs
  int id1 = r1->GlobalID(),  id2 = r2->GlobalID();
  A[id1][id2] += ...; // Fill matrix A coefficients
  b[id1] += ...; // Adjust RHS vector
}
```

CONCLUSION

Integrated
Numerical
Modeling and
Object-oriented
Supercomputing
Technologies

Arbitrary polyhedral meshes
Parallel platform: ghost-cells, ParMETIS, Zoltan
Linear solvers: PETSc, BIILU2

www.inmost.org