

A Flexible and Portable Approach for Communication in Distributed Computing Systems

Erik Zenker

Professor: Prof. Dr. Wolfgang E. Nagel

Tutors: Dr. Michael Bussmann, Dr. Guido Juckeland, Andy Georgi

17th December 2014



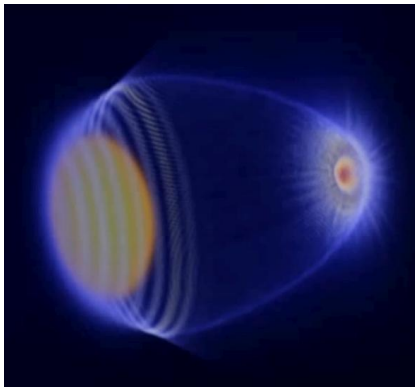
**TECHNISCHE
UNIVERSITÄT
DRESDEN**

hzdr



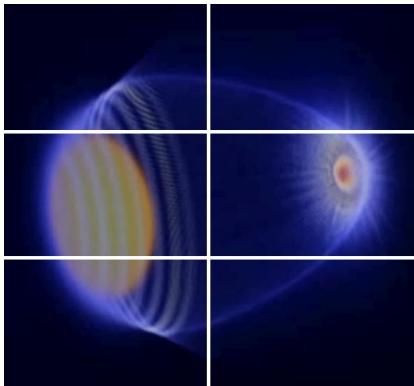
**HELMHOLTZ
ZENTRUM DRESDEN
ROSSENDORF**

Developing a Distributed Simulation is Simple



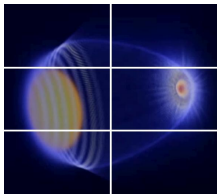
simulation
domain

Developing a Distributed Simulation is Simple

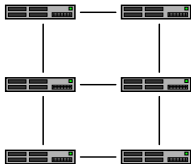


decomposed
domain

Developing a Distributed Simulation is Simple

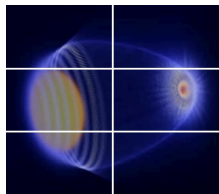


decomposed
domain

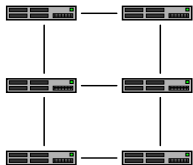


homogeneous
cluster

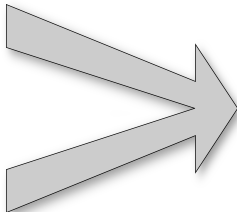
Developing a Distributed Simulation is Simple



decomposed
domain



homogeneous
cluster

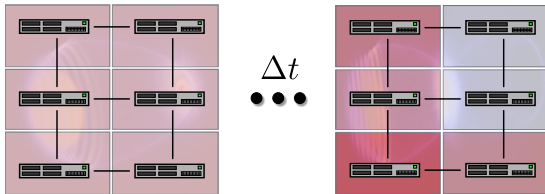


one-to-one
mapping

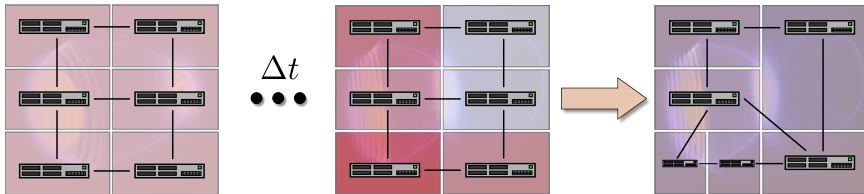
Or Maybe not?



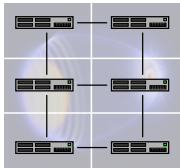
Or Maybe not?



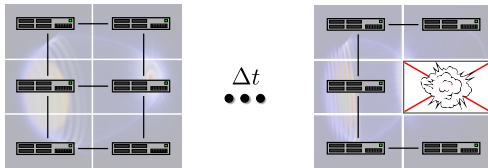
Or Maybe not?



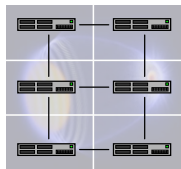
Or Maybe not?



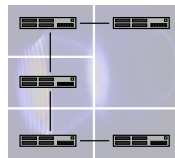
Or Maybe not?



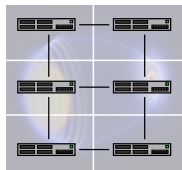
Or Maybe not?



Δt
• • •



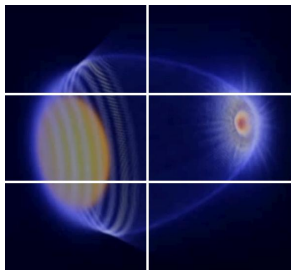
Or Maybe not?



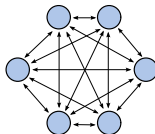
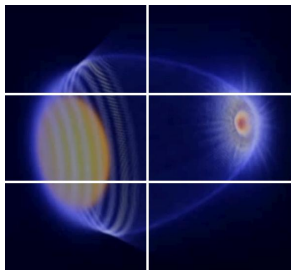
Δt
• • •



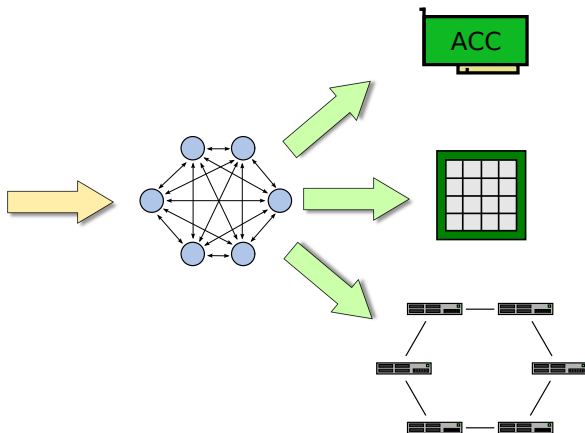
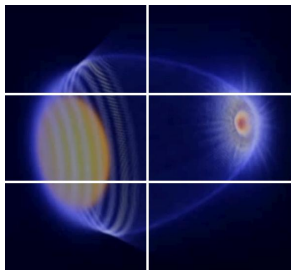
Or Maybe not?



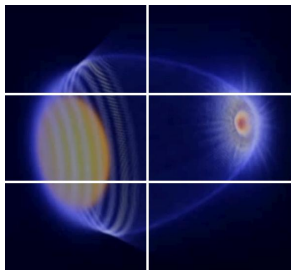
Or Maybe not?



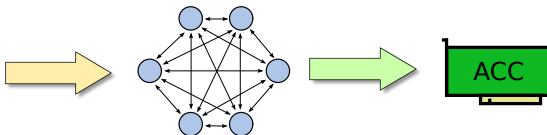
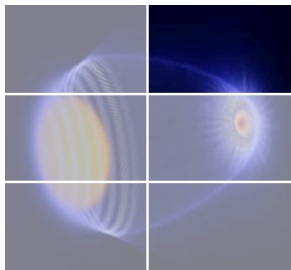
Or Maybe not?



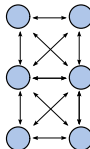
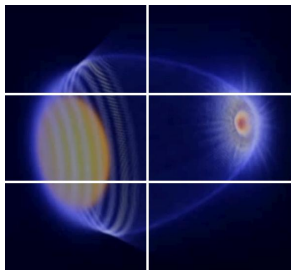
Or Maybe not?



Or Maybe not?



Or Maybe not?



A More Flexible Approach is Required

- **Abstraction** from data transfer methods of hardware topologies
 - Abstraction from MPI send, memcpy, offload etc.

A More Flexible Approach is Required

- **Abstraction** from data transfer methods of hardware topologies
 - Abstraction from MPI send, memcpy, offload etc.
- **Modeling** of communication topologies
 - Next neighbor
 - 3D torus
 - All-to-all

A More Flexible Approach is Required

- **Abstraction** from data transfer methods of hardware topologies
 - Abstraction from MPI send, memcpy, offload etc.
- **Modeling** of communication topologies
 - Next neighbor
 - 3D torus
 - All-to-all
- **Mapping** of communication topologies to hardware topologies

What is this Approach?

- It is **not** ...
 - a communication framework

What is this Approach?

- It is **not** ...
 - a communication framework
 - a load balancing approach

What is this Approach?

- It is **not** ...
 - a communication framework
 - a load balancing approach
 - a fault tolerance approach

What is this Approach?

- It is **not** ...
 - a communication framework
 - a load balancing approach
 - a fault tolerance approach
- But, it **is** ...
 - a foundation to solve problems such as load balancing, fault tolerance, and computations in heterogeneous system

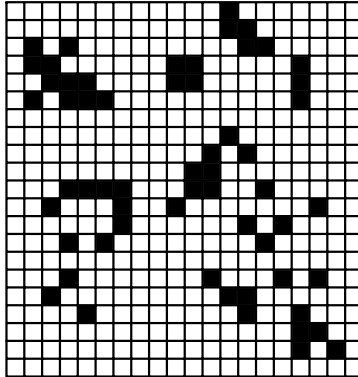
What is this Approach?

- It is **not** ...
 - a communication framework
 - a load balancing approach
 - a fault tolerance approach
- But, it **is** ...
 - a foundation to solve problems such as load balancing, fault tolerance, and computations in heterogeneous system
 - a mapping from decomposed domain to communication topologies

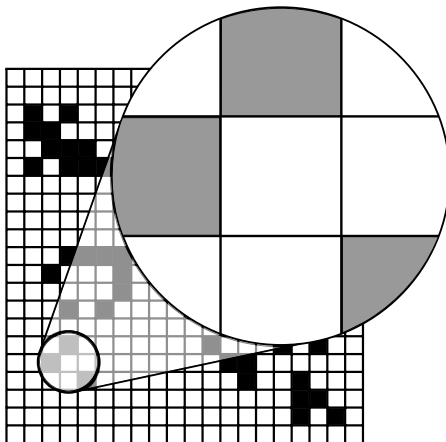
What is this Approach?

- It is **not** ...
 - a communication framework
 - a load balancing approach
 - a fault tolerance approach
- But, it **is** ...
 - a foundation to solve problems such as load balancing, fault tolerance, and computations in heterogeneous system
 - a mapping from decomposed domain to communication topologies
 - a mapping from communication topologies to hardware topologies

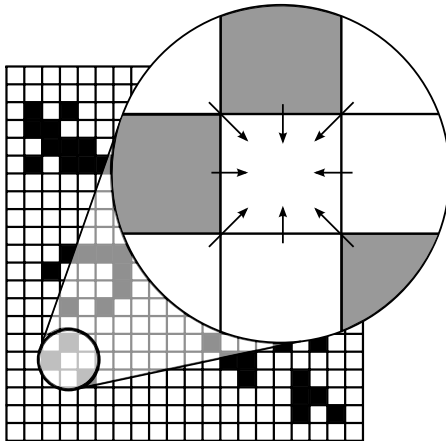
Conway's Game of Life



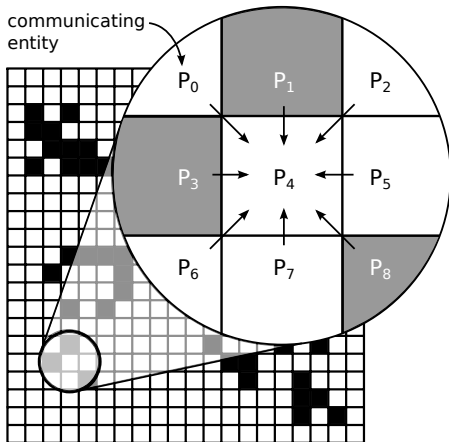
Conway's Game of Life



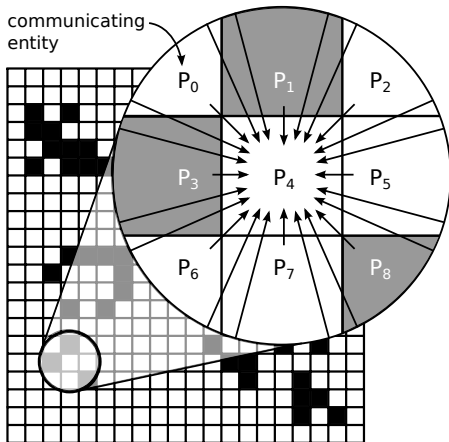
Zenker's Game of Life

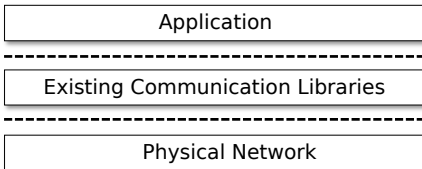


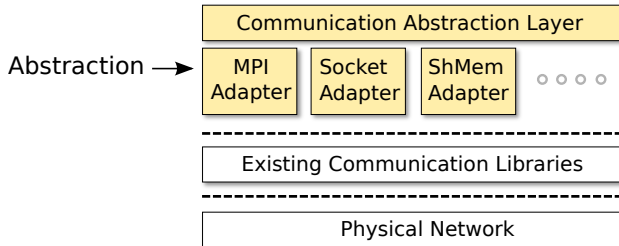
Zenker's Game of Life



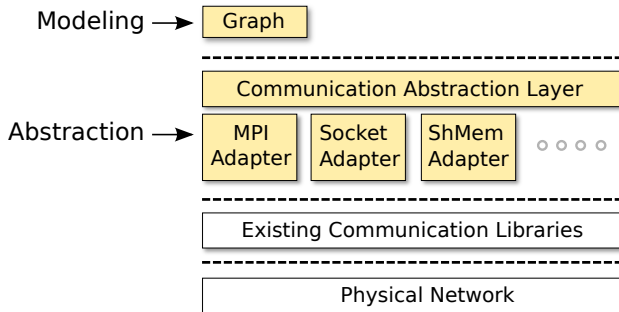
Zenker's Game of Life



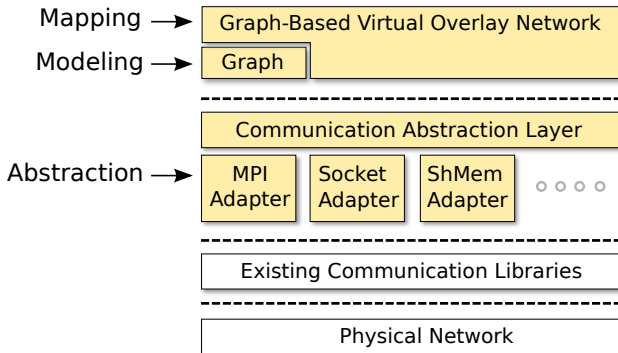




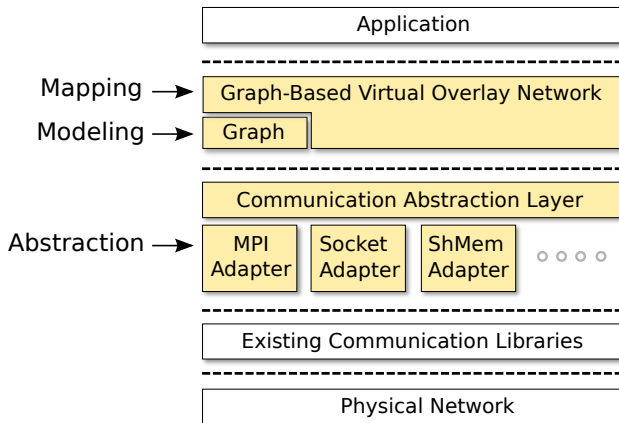
Design Overview



Design Overview



Design Overview



An Abstract Communication Layer is the Basis

peer



- A peer is an instance participating on communication

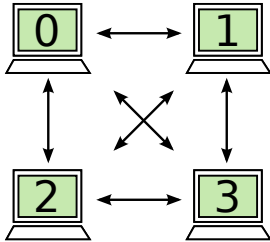
An Abstract Communication Layer is the Basis



- A peer is an instance participating on communication
- Each peer is identified by a virtual address

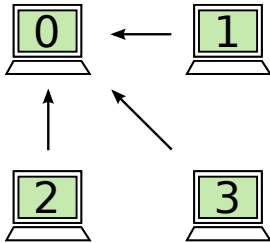


An Abstract Communication Layer is the Basis



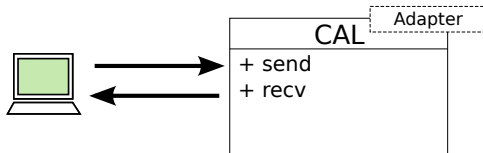
- A peer is an instance participating on communication
- Each peer is identified by a virtual address
- Peers exchange data
 - Point-to-point operations
 - Collective operations

An Abstract Communication Layer is the Basis

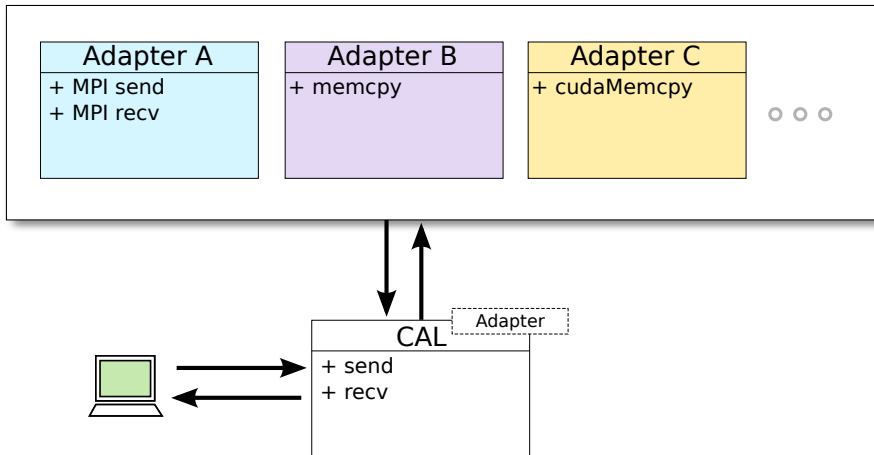


- A peer is an instance participating on communication
- Each peer is identified by a virtual address
- Peers exchange data
 - Point-to-point operations
 - Collective operations

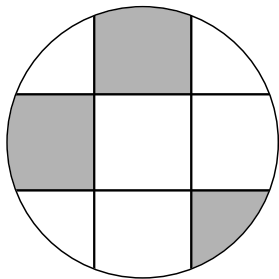
Communication is not Newly Invented



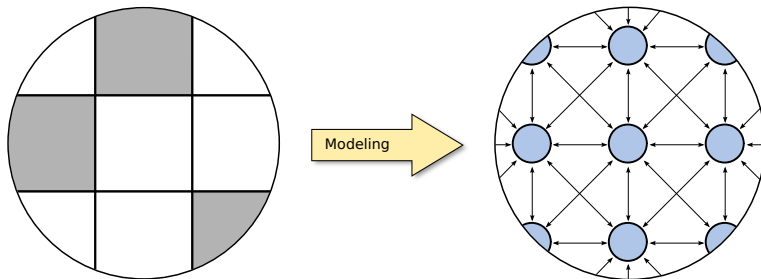
Communication is not Newly Invented



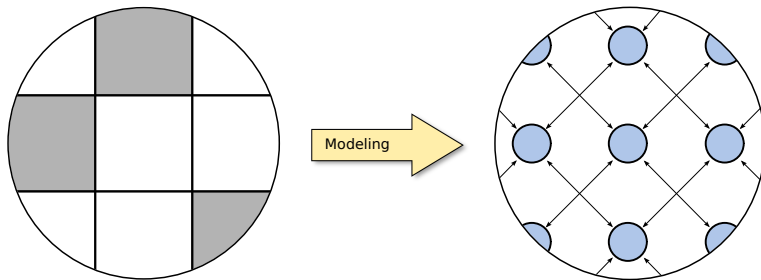
A Graph, the Most General Description



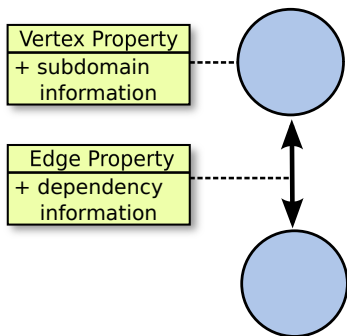
A Graph, the Most General Description



Lets Change my World

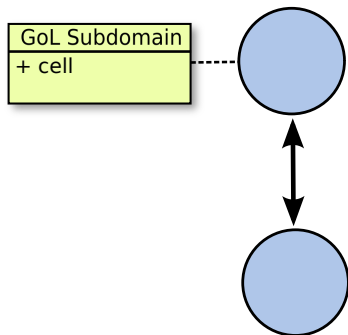


A Graph Can Provide More Information!



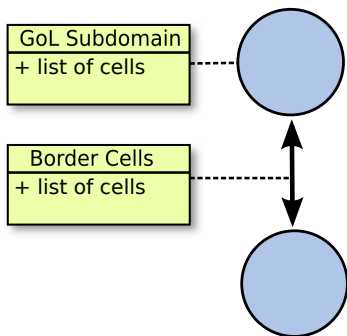
- Properties cover simulation domain specific information
- Properties are bounded to vertices and edges

A Graph Can Provide More Information!



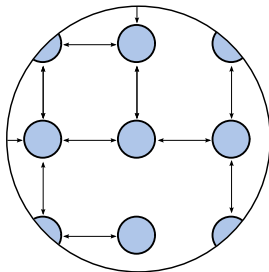
- Properties cover simulation domain specific information
- Properties are bounded to vertices and edges

A Graph Can Provide More Information!



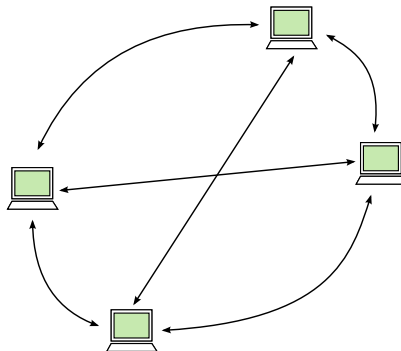
- Properties cover simulation domain specific information
- Properties are bounded to vertices and edges

Emergence of Flexibility by Combination



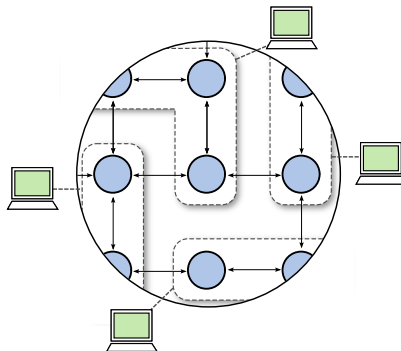
- The **Graph** provides the communication topologie

Emerge the Flexibility by Combination



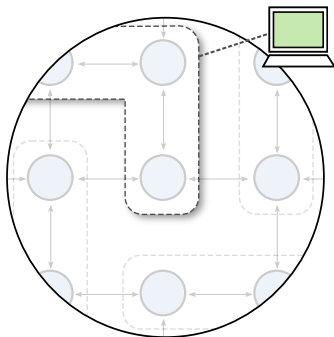
- The **Graph** provides the communication topologie
- The **CAL** provides communicatin abilities

Emergence of Flexibility by Combination



- The **Graph** provides the communication topologie
- The **CAL** provides communicatin abilities
- Together it creates a **Graph-based Virtual Overlay Network**

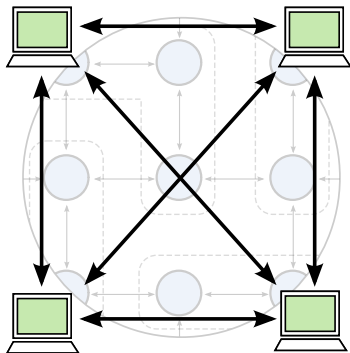
The GVON Maps Vertices to Peers



1 Distribution

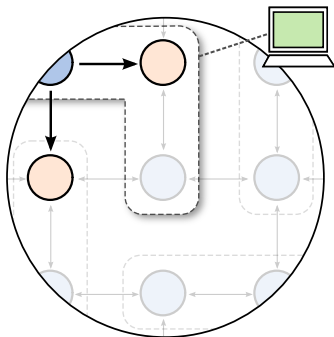
→ Distribute vertices to peers

The GVON Maps Vertices to Peers



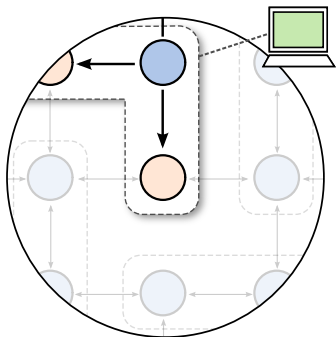
- 1 Distribution
→ Distribute vertices to peers
- 2 Announcement
→ Update mapping

The GVON Maps Vertices to Peers



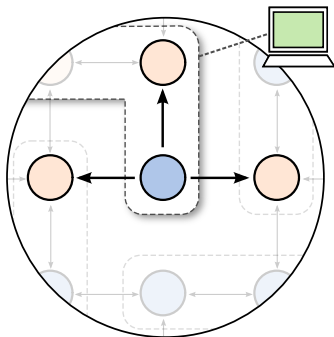
- 1 Distribution
→ Distribute vertices to peers
- 2 Announcement
→ Update mapping
- 3 Communication
→ Point-to-point operations
→ Collective operations

The GVON Maps Vertices to Peers



- 1 Distribution
→ Distribute vertices to peers
- 2 Announcement
→ Update mapping
- 3 Communication
→ Point-to-point operations
→ Collective operations

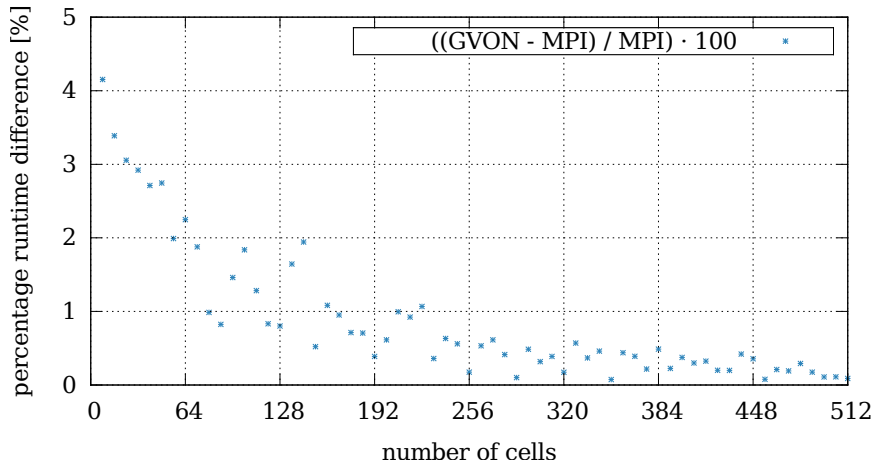
The GVON Maps Vertices to Peers



- 1 Distribution
→ Distribute vertices to peers
- 2 Announcement
→ Update mapping
- 3 Communication
→ Point-to-point operations
→ Collective operations

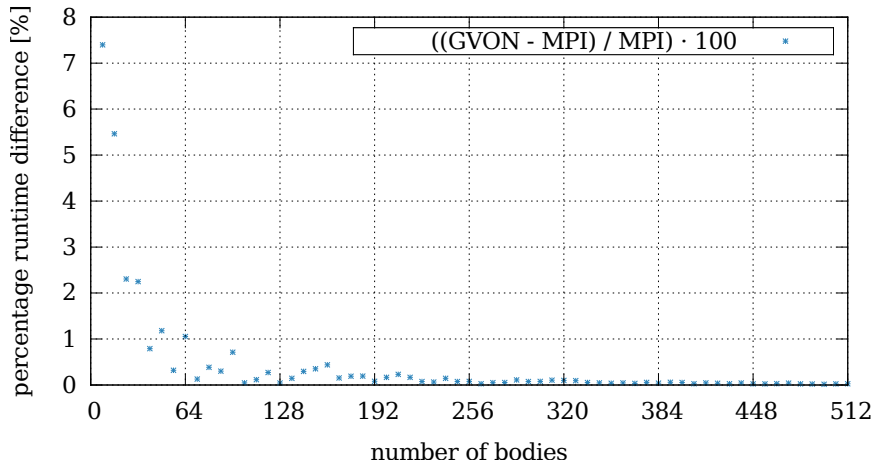
Abstraction is not a Source of Serious Overhead

GoL : relative runtime difference with respect to native MPI

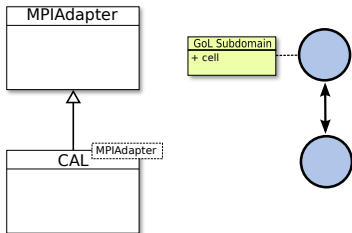


Abstraction is not a Source of Serious Overhead

N-body : relative runtime difference with respect to native MPI



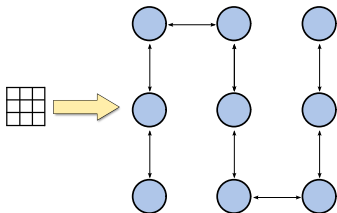
A Simulation in Three Steps



1 Configuration at compile-time

- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

A Simulation in Three Steps



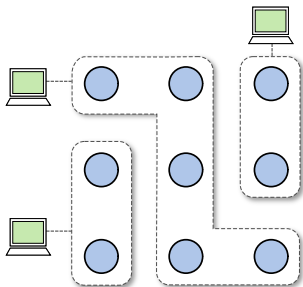
1 Configuration at compile-time

- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

2 Initialization

- Create Gol graph
- Distribute vertices to peers
- Announce vertices

A Simulation in Three Steps



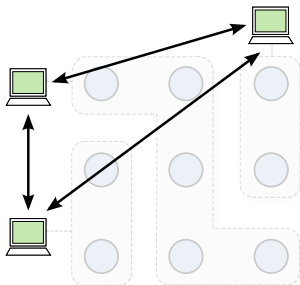
1 Configuration at compile-time

- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

2 Initialization

- Create Gol graph
- Distribute vertices to peers
- Announce vertices

A Simulation in Three Steps



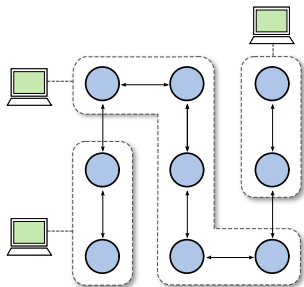
1 Configuration at compile-time

- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

2 Initialization

- Create Gol graph
- Distribute vertices to peers
- Announce vertices

A Simulation in Three Steps



1 Configuration at compile-time

- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

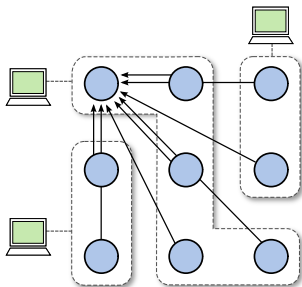
2 Initialization

- Create Gol graph
- Distribute vertices to peers
- Announce vertices

3 Communication

- Exchange data of neighboring cells
- Collect state information of all cells

A Simulation in Three Steps



1 Configuration at compile-time

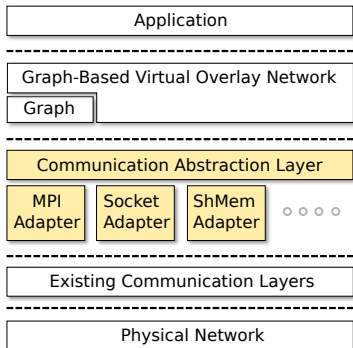
- CAL with MPI adapter
- Graph with cell property
- GVON with specific Graph and CAL

2 Initialization

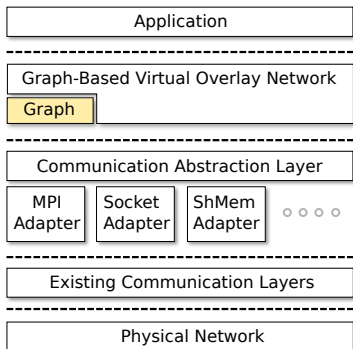
- Create Gol graph
- Distribute vertices to peers
- Announce vertices

3 Communication

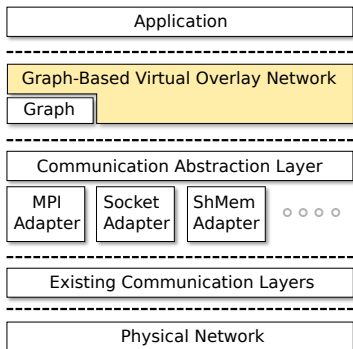
- Exchange data of neighboring cells
- Collect state information of all cells



- By using adapters, the **CAL** abstracts from existing communication libraries



- By using adapters, the **CAL** abstracts from existing communication libraries
- The **Graph** models communication topologies



- By using adapters, the **CAL** abstracts from existing communication libraries
- The **Graph** models communication topologies
- The **GVON** maps graphs to peers and provides communication methods on graph-basis

Example Code - Configuration

```
1 // CAL configuration
2 typedef CAL<MPIAdapter>           MyCAL;
3 typedef typename MyCAL::Context   Context;
4
5 // Graph configuration
6 typedef Graph<Cell>               MyGraph;
7 typedef typename MyGraph::Vertex  Vertex;
8 typedef typename MyGraph::Edge    Edge;
9
10 // GVON configuration
11 typedef GVON<MyGraph, MyCAL>      MyGVON;
```

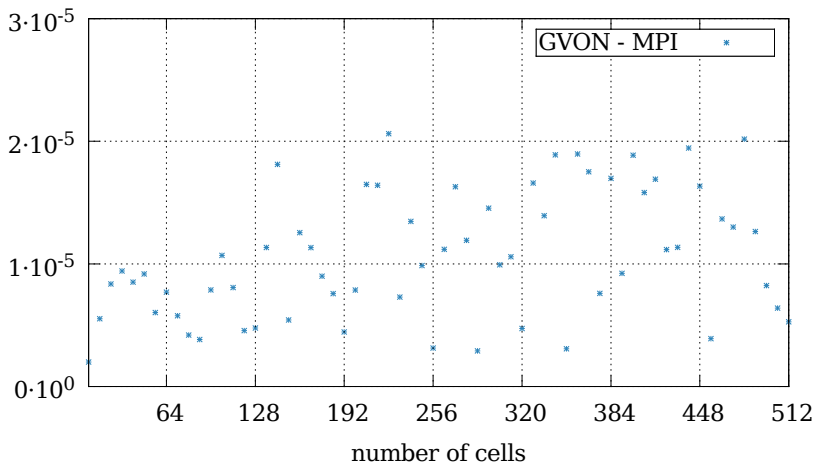
Example Code - Initialization

```
1 using namespace std;
2
3 // Graph generation and
4 vector<Vertex> vertices;
5 auto edges = generateGraph<MyGraph>(vertices);
6
7 // Object instantiation
8 MyGraph graph(vertices, edges);
9 MyCAL    cal;
10 MyGVON   gvon(cal);
11
12 // Mapping
13 Context context      = cal.getGlobalContext();
14 auto hostedVertices = mapping(graph, context);
15
16 // Announcement
17 gvon.announce(graph, hostedVertices);
```

Example Code - Communication

```
1 // Communication
2 for(Vertex v : hostedVertices){
3     auto outEdges = graph.getOutEdges(v);
4     for(std::pair<Vertex, Edge> outEdge : outEdges){
5         // Data to send
6         vector<unsigned> data(1, v.id);
7         // Prepare destination
8         Vertex destVertex = edge.first;
9         Edge     edge      = edge.second;
10        // Send data
11        gvon.send(graph, destVertex, edge, data));
12    }
13 }
```

GoL : absolute runtime difference with respect to native MPI



N-body : absolute runtime difference with respect to native MPI

