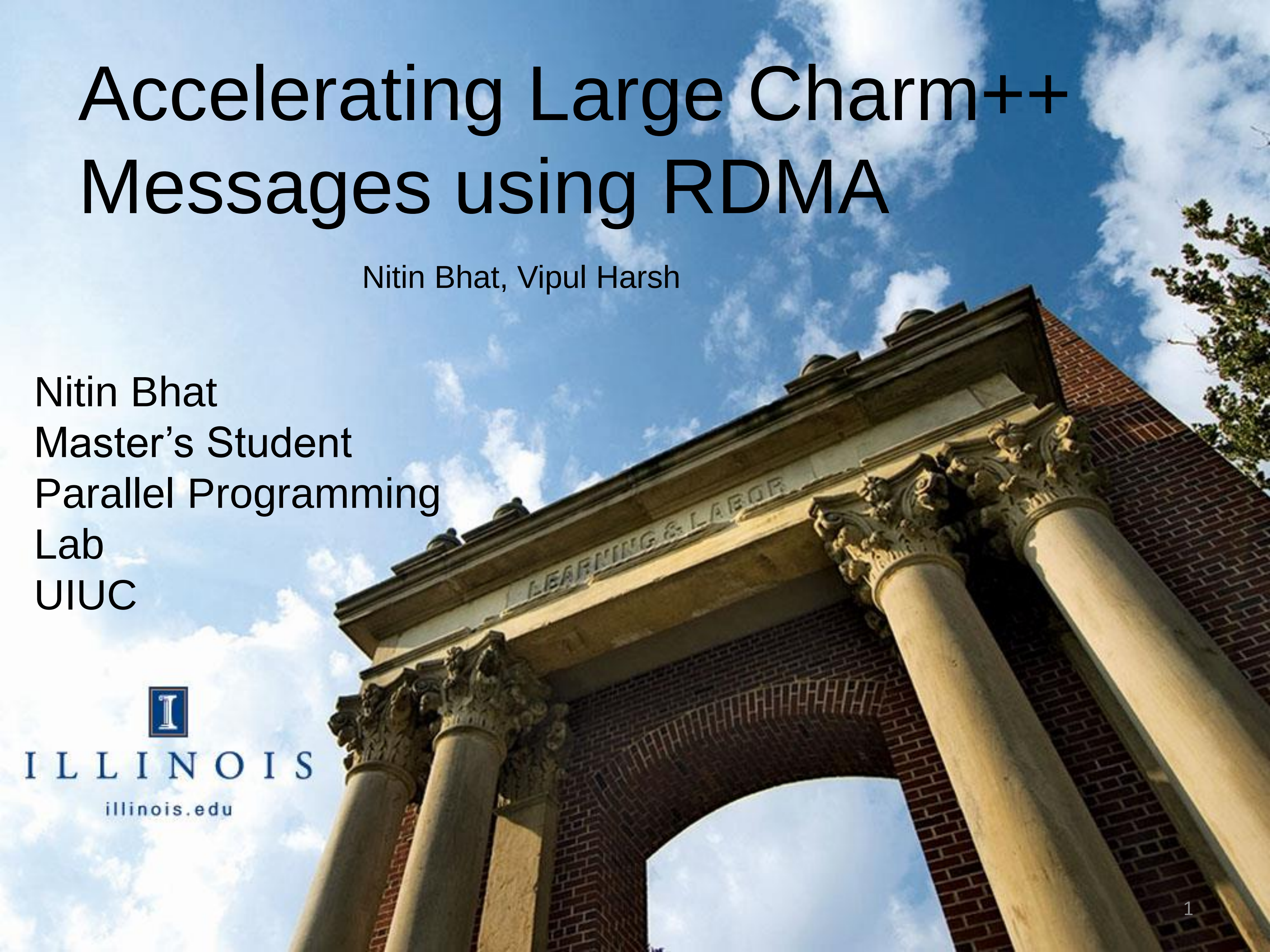
Accelerating Large Charm++ Messages using RDMA

Nitin Bhat, Vipul Harsh

Nitin Bhat
Master's Student
Parallel Programming Lab
UIUC

ILLINOIS
illinois.edu



Motivation

- Major bottleneck in HPC Applications – Communication

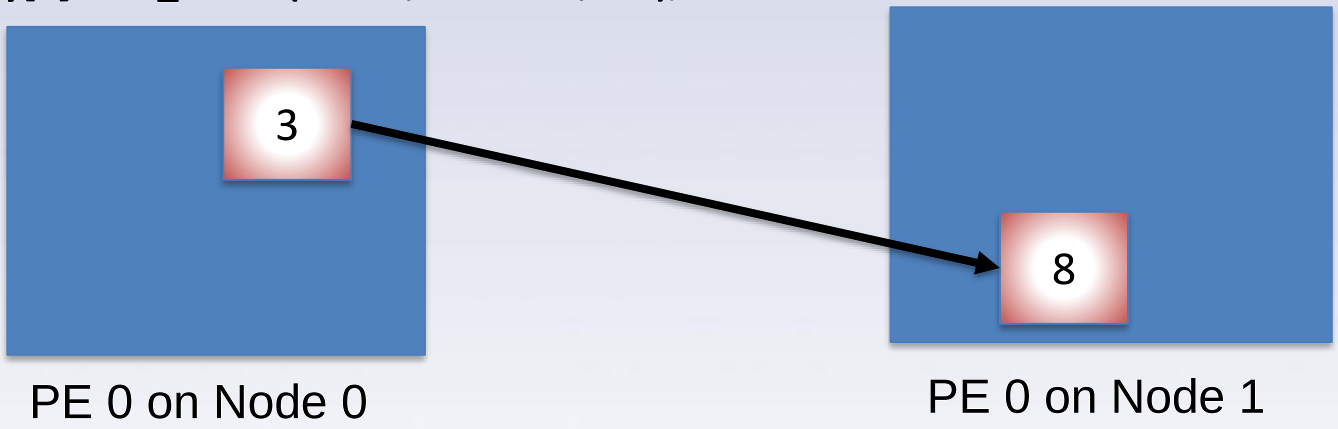
- Strategies to address communication bottlenecks
  - Overlap communication and computation
  - Topology aware mapping
  - **Reduce message sending times**
    - **Avoiding copying for large messages**

illinois.edu

Charm++ Programming Model

- Asynchronous Message Driven Execution
- Naturally One-sided

```
Cell_Proxy[8].recv_forces(forces, 1000000, 4.0);
```



illinois.edu

forcecalculations.ci

```
Module forcecalculations{
    ……
    array [1D] Cell {
        entry forces( ) ;
        entry void recv_forces (double forces [size], int size, double value);
    }
    ……..
}
```

Charm Interface File - Declarations

forcecalculations.C

```
void recv_forces(double * forces, int size, double value){
        ….
}
```

C++ Code File – Entry method

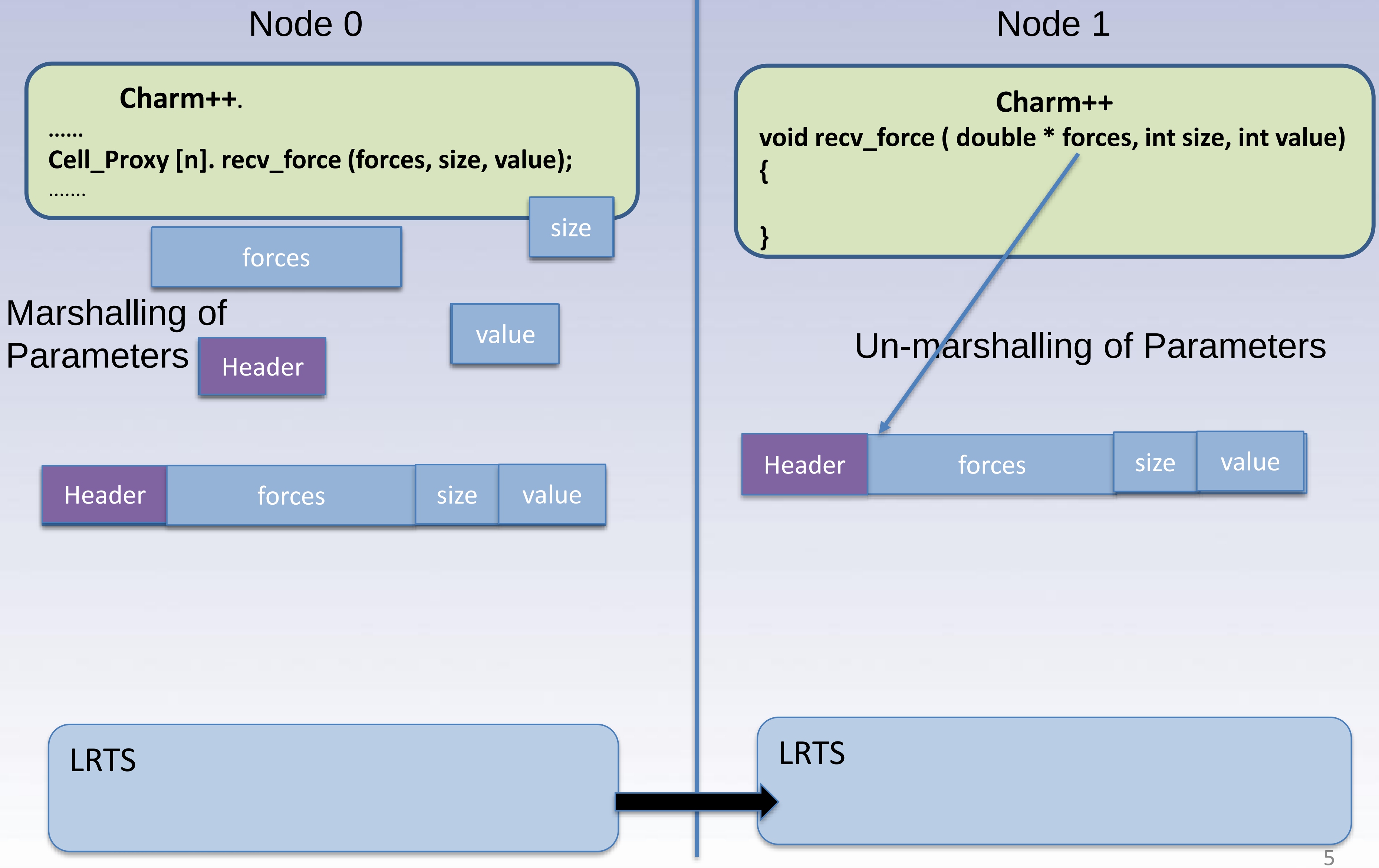
forcecalculations.C

```
Cell_Proxy[n].recv_forces(forces, 1000000, 4.0);
```

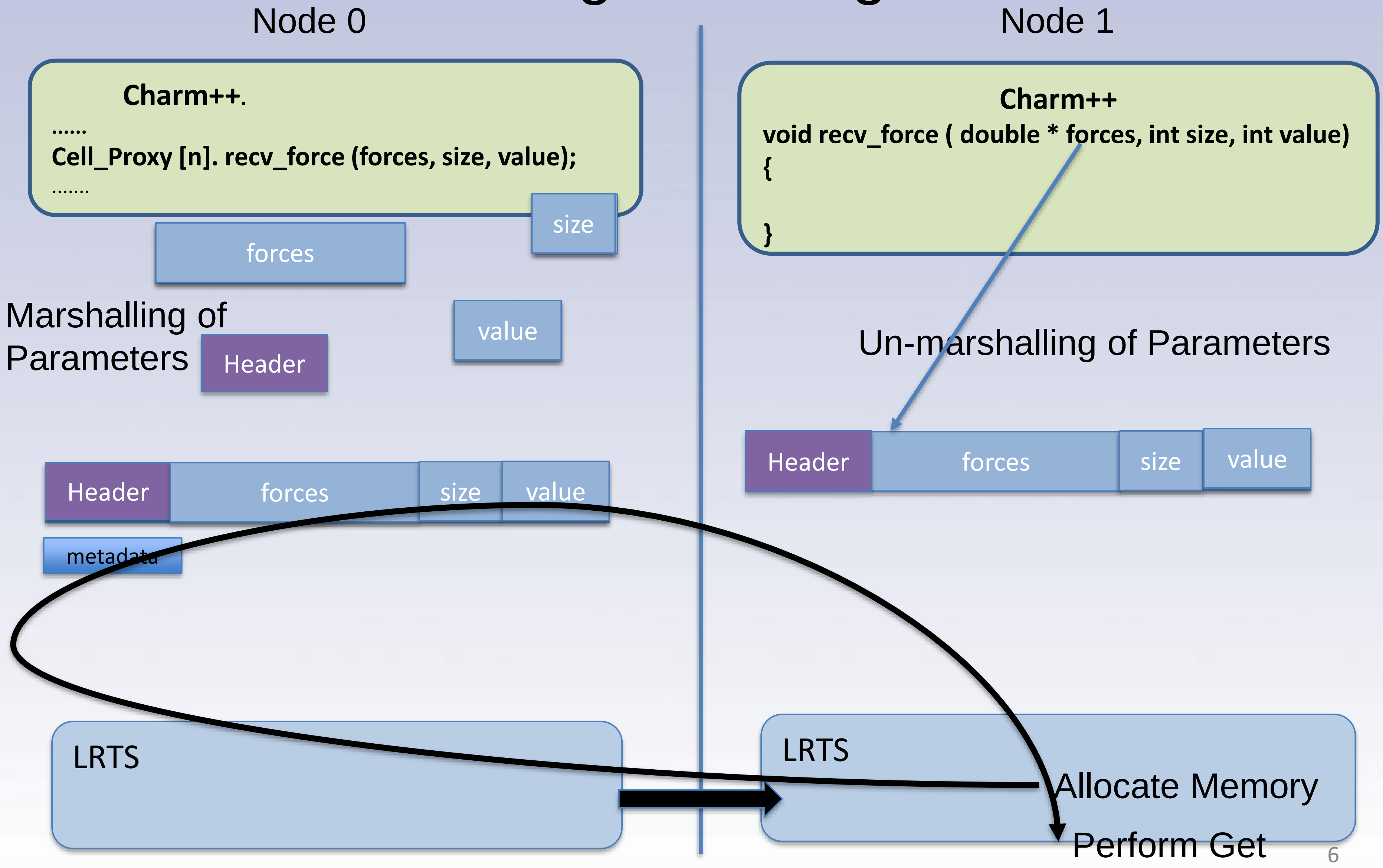
C++ Code File – Call site

illinois.edu

What happens under the hood?


Node 0
Node 1
Charm++.
……
Cell_Proxy [n]. recv_force (forces, size, value);
…….
size
forces
Marshalling of Parameters
value
Header
Header
forces
size
value
Charm++
void recv_force (double * forces, int size, int value)
{
}
Un-marshalling of Parameters
Header
forces
size
value
LRTS
LRTS

In Rdma enabled networks for large messages:


Node 0
Node 1
Charm++.
……
Cell_Proxy [n]. recv_force (forces, size, value);
…….
size
forces
Marshalling of Parameters
Header
value
Header
forces
size
value
metadata
LRTS
Charm++
void recv_force (double * forces, int size, int value)
{
}
Un-marshalling of Parameters
Header
forces
size
value
LRTS
Allocate Memory
Perform Get

How to accelerate large messages?

- Avoid sender side copy of a large messages
  - Small parameters will be marshalled into contiguous memory and sent.
  - Large arrays will be sent through Rdma Get Operations.

illinois.edu

Regular Charm++

forcecalculations.ci

```
Module forcecalculations{
      ……
      array [1D] Cell {
            entry forces( ) ;
            entry void recv_forces (double forces [size], int size, double value);
      }
      ……..
}
```

No copy Rdma API

forcecalculations.ci

```
Module forcecalculations{
      ……
      array [1D] Cell {
            entry forces( ) ;
            entry void recv_forces (Rdma double forces [size], int size, double value);
      }
      ……..
}
```

illinois.edu

Regular Charm++

forcecalculations.C

```
Cell_Proxy[98].recv_forces(forces, 1000000, 4.0);
```

C++ Code File – Call site

No copy Rdma API

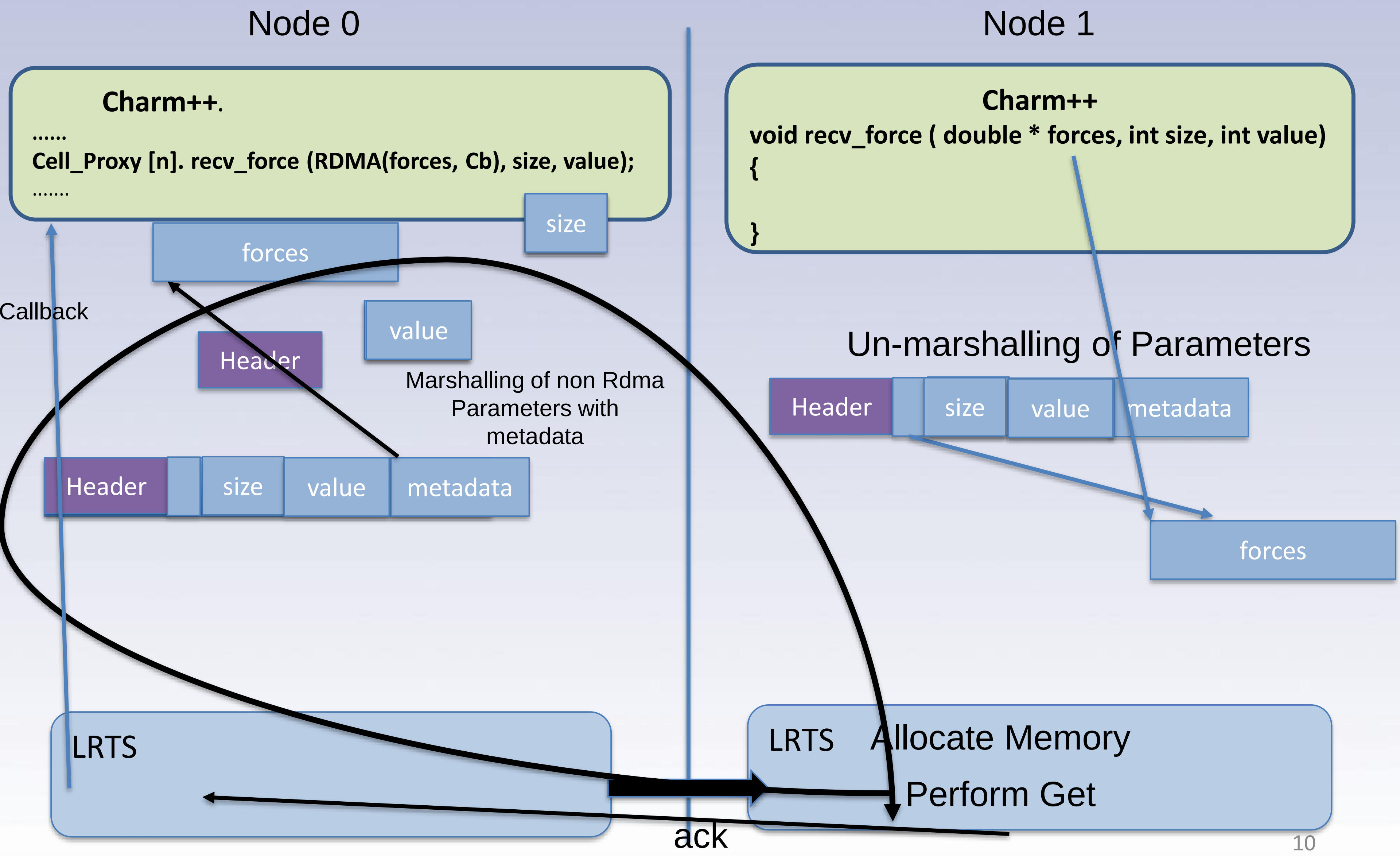
forcecalculations.C

```
Callback Cb = new Callback(CkIndex_Cell::completed, cellArrayID);

Cell_Proxy[98].recv_forces( RDMA(forces, Cb), 1000000, 4.0);
```

C++ Code File – Call site

illinois.edu

No Copy One-sided API



```
Charm++.
……
Cell_Proxy [n]. recv_force (RDMA(forces, Cb), size, value);
…….
```

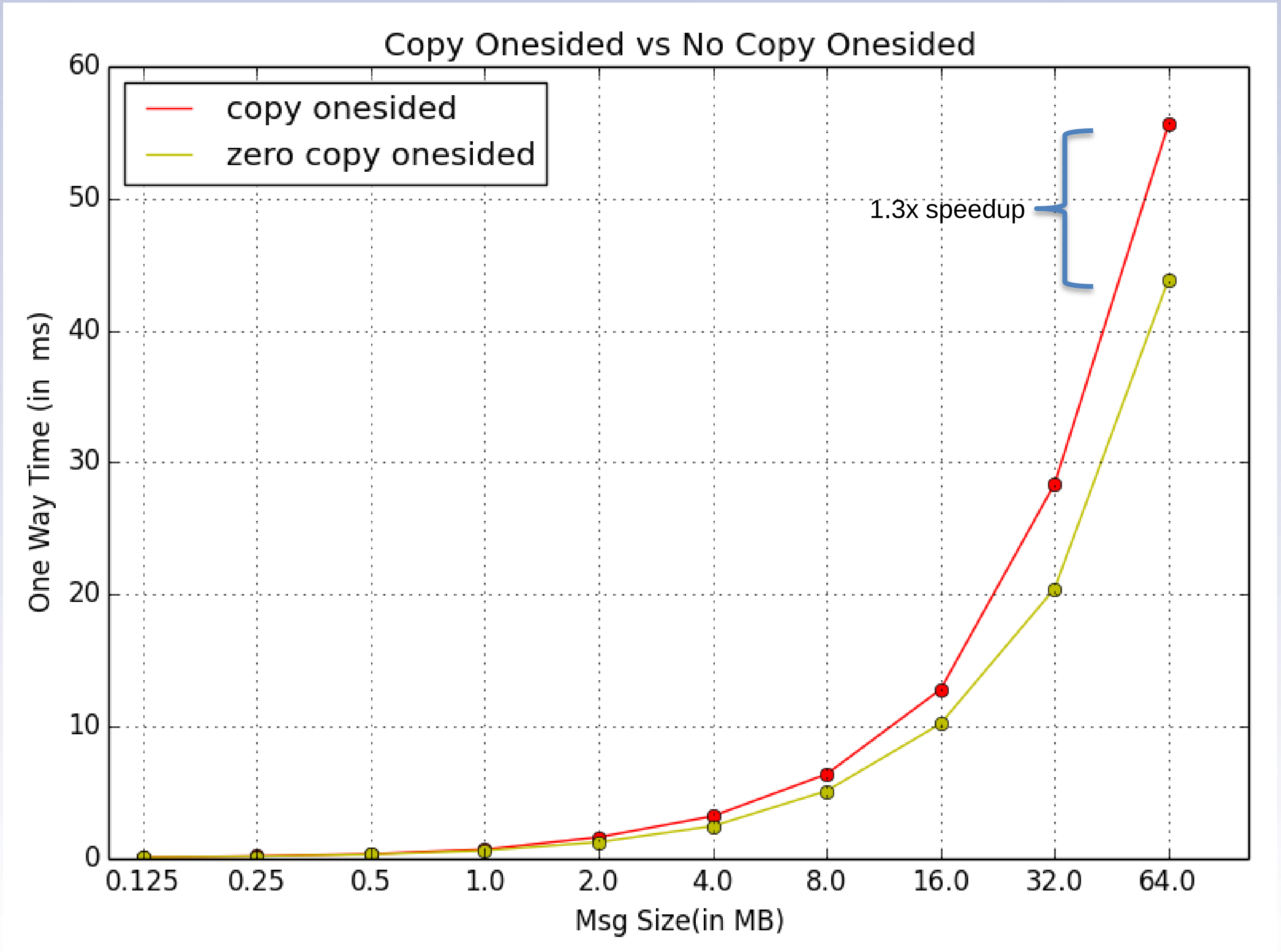
```
Charm++
void recv_force ( double * forces, int size, int value)
{

}
```

Results on Bluegene/Q Vesta – Pingpong Benchmark

Message Size (MB)	Existing One-sided Paradigm (ms)	No-copy One-sided Paradigm (ms)	Speed Up
0.125	0.1040	0.1036	1.01
0.25	0.19	0.18	1.07
0.5	0.36	0.32	1.12
1	0.70	0.61	1.14
2	1.62	1.25	1.30
4	3.21	2.46	1.31
8	6.40	5.13	1.25
16	12.81	10.22	1.25
32	28.38	20.44	1.39
64	55.62	43.87	1.27

illinois.edu

Performance Improvement



illinois.edu

Conclusions and Future Work

- Saving copy for large messages in RDMA supported networks improves performance

- On the receiver side, the user can pre-allocate a buffer and post a receive.
- Persistent RDMA
- Use cases in :
  - Charm++ with a posted receive
  - Charm++ sdag when clause
  - AMPI non blocking receive

illinois.edu

Questions?

illinois.edu