



... for a brighter future

Hybrid Parallel Programming with MPI and PGAS (UPC)

P. Balaji (Argonne), R. Thakur (Argonne), E. Lusk (Argonne)
James Dinan (OSU)



U.S. Department
of Energy

UChicago ►
Argonne_{LLC}



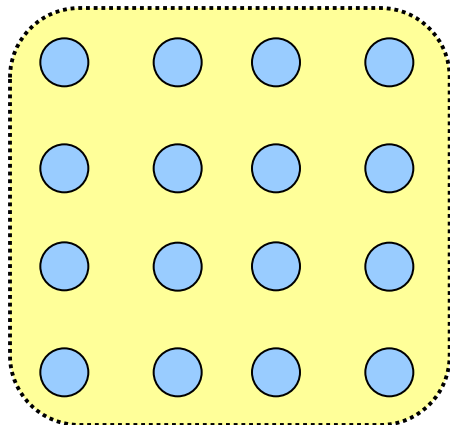
A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

Motivation

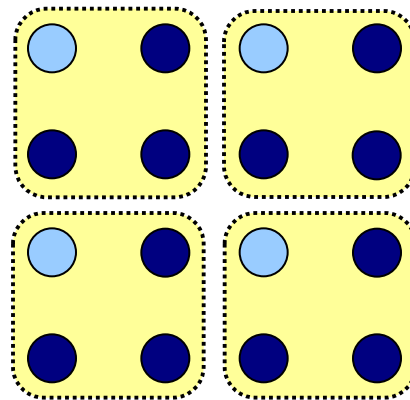
- MPI and UPC have their own advantages
 - UPC:
 - Distributed data structures (arrays, trees)
 - Implicit and explicit one-sided communication
 - Good for irregular codes
 - Can support large data sets
 - Multiple virtual address spaces joined together to form a global address space
 - MPI:
 - Groups
 - Topology-aware functionality (e.g., Cart functionality)

Extending MPI to work well with PGAS

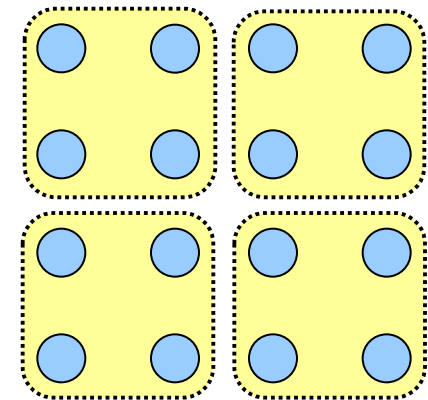
- MPI can handle some parts and allow PGAS to do handle others
 - E.g., MPI can handle outer-level coarse-grained parallelism, scalability, fault tolerance and allow PGAS to handle inner-level fine-grained parallelism



Flat Model



Nested Funneled



Nested Multiple

Description of Models

- Nested Multiple
 - MPI launches multiple UPC groups of processes
 - Note: Here “one process” refers to all entities that share one virtual address space
 - Each UPC process will have an MPI rank
 - Can make MPI calls
- Nested Funneled
 - MPI launches multiple UPC groups of processes
 - Only one UPC process can make MPI calls
 - Currently not restricted to the “master process” like with threads
 - Applications can extend address space without affecting other internal components

Description of Models (contd.)

- Flat Model
 - Subset of Nested-Multiple
 - ... but might be easier to implement

What does MPI need to do?

- Hybrid initialization
 - `MPI_Init_hybrid(&argc, &argv, int ranks_per_group)`
- When MPI is launched, it needs to know how many processes are being launched
 - Currently we use a flat model
 - If 10 processes are being launched, we know that world size is 10
 - Hybrid launching can be hierarchical
 - 10 processes are launched, each of which might launch 10 other processes → world size can be 100 (in the case of Nested-Multiple)

Other Issues with Interoperability

- No mapping between MPI and UPC ranks
 - Application needs to explicitly figure out
 - Can be done portably with enough number of MPI_Alltoall and Allgather calls
- Communication Deadlock
 - In some cases deadlocks can be avoided by implicit progress done by either MPI or UPC
 - Being handled as ticket #154
 - Might get voted out
 - Application might need to assume the worst case

Other Issues with Interoperability (contd.)

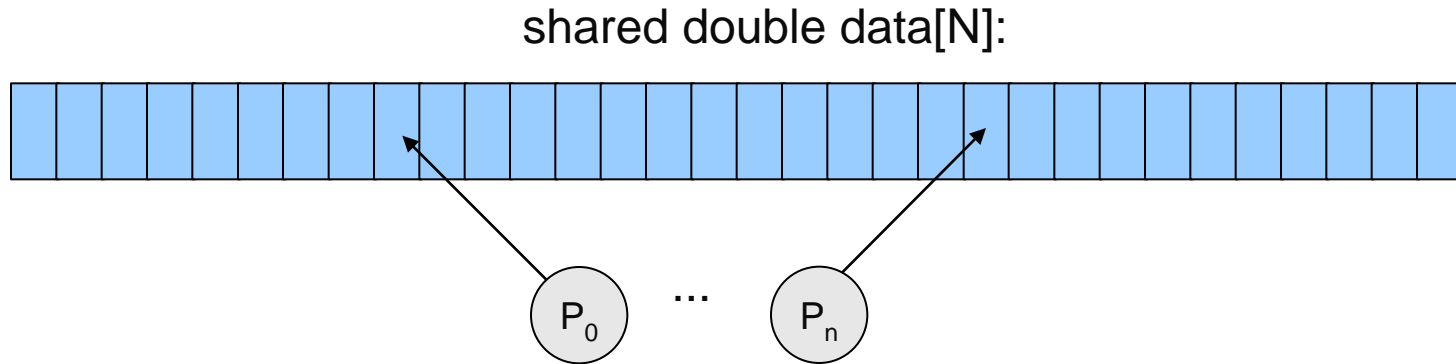
- There is no sharing of MPI and UPC objects
 - MPI does not know how to send data from “global address spaces”
 - User has to provide the data in its virtual address space
 - UPC cannot perform RMA into MPI windows

Implementation in MPICH2

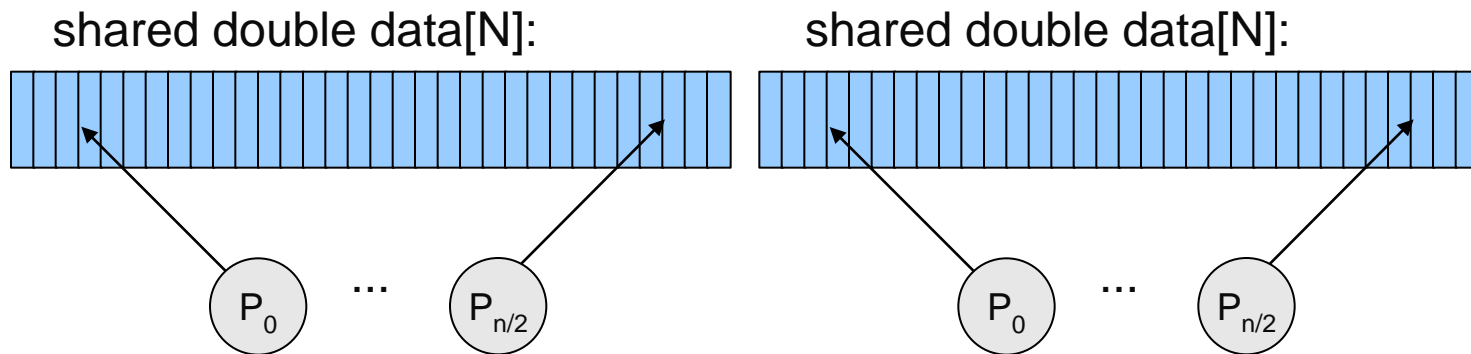
- Rough implementation available
 - Will be corrected once the details are finalized

Random Access Benchmark

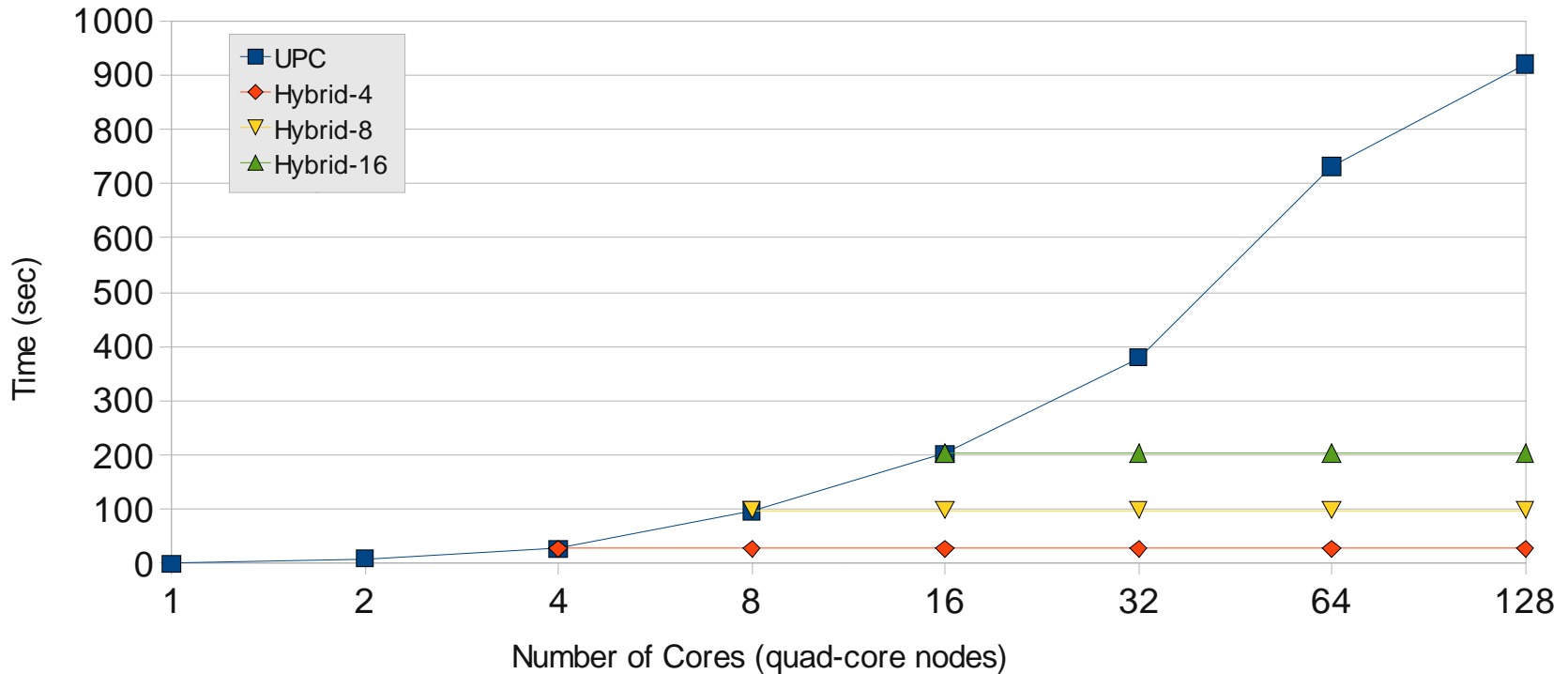
- UPC: Threads access random elements of distributed shared array



- Hybrid: Array is replicated on every group

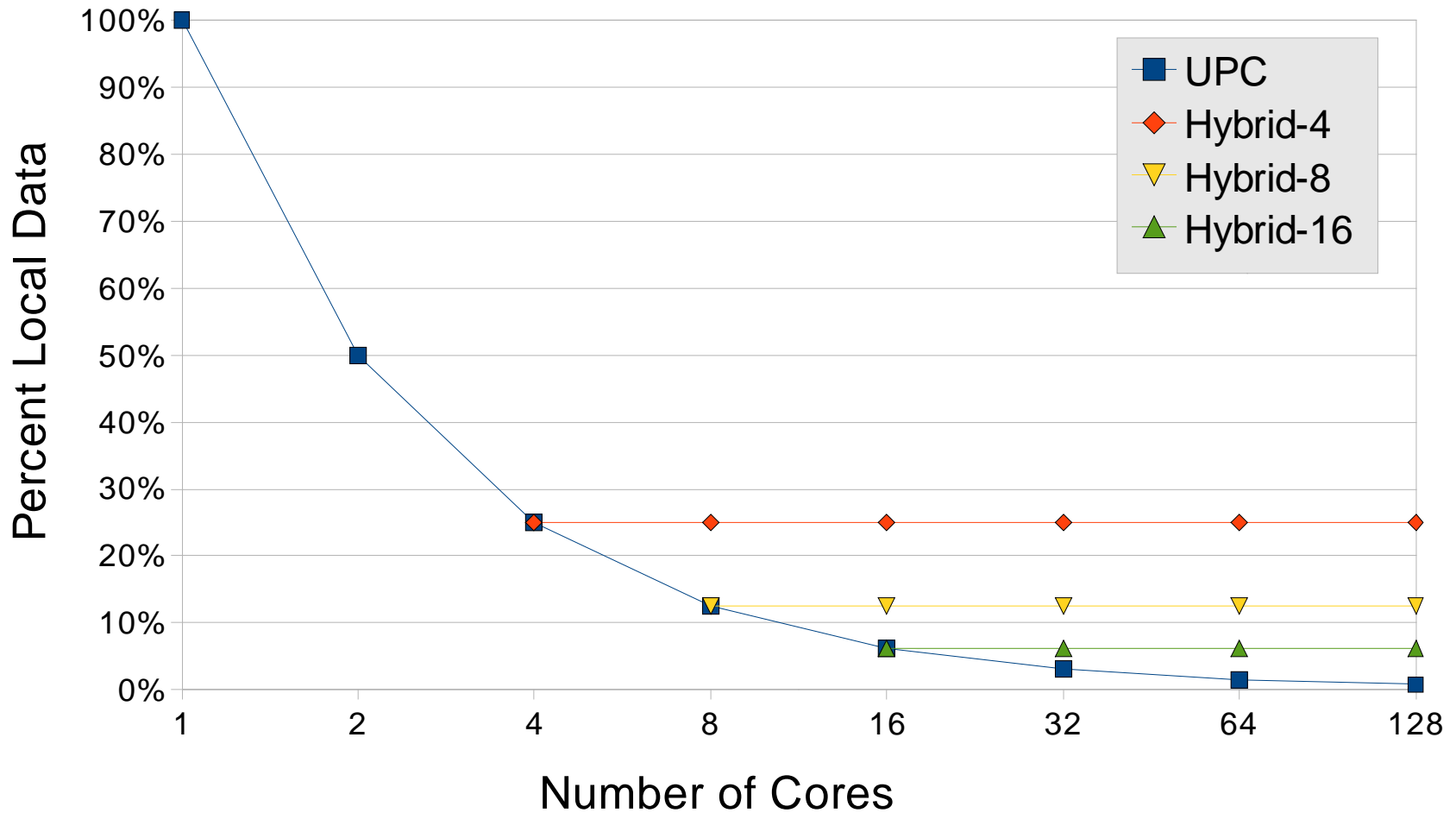


Impact of Data Locality on Performance



- Each process performs 1,000,000 random accesses
- Weak scaling ideal: Flat line

Percentage Local References



Barnes-Hut n-Body Cosmological Simulation

- Simulates gravitational interactions of a system of n bodies
- Represents 3-d space using an oct-tree
- Summarize distant interactions using center of mass

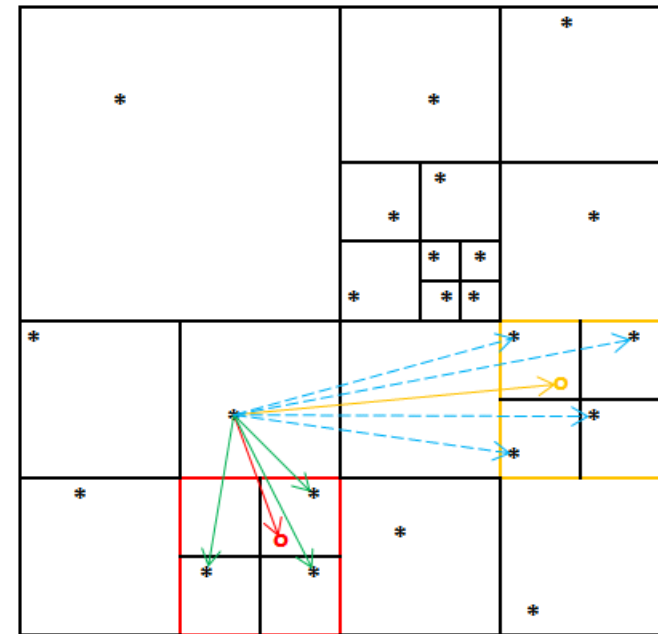
```
for i in 1..t_max
  t <- new octree()

  forall b in bodies
    insert(t, b)

  summarize_subtrees(t)

  forall b in bodies
    compute_forces(b, t)

  forall b in bodies
    advance(b)
```



Credit: Lonestar Benchmarks (Pingali et al)

Hybrid Barnes Algorithm

```
for i in 1..t_max  
  t <- new octree()
```

```
forall b in bodies  
  insert(t, b)
```

Tree is distributed across group

```
summarize_subtrees(t)  
our_bodies <- partition(group id, bodies)
```

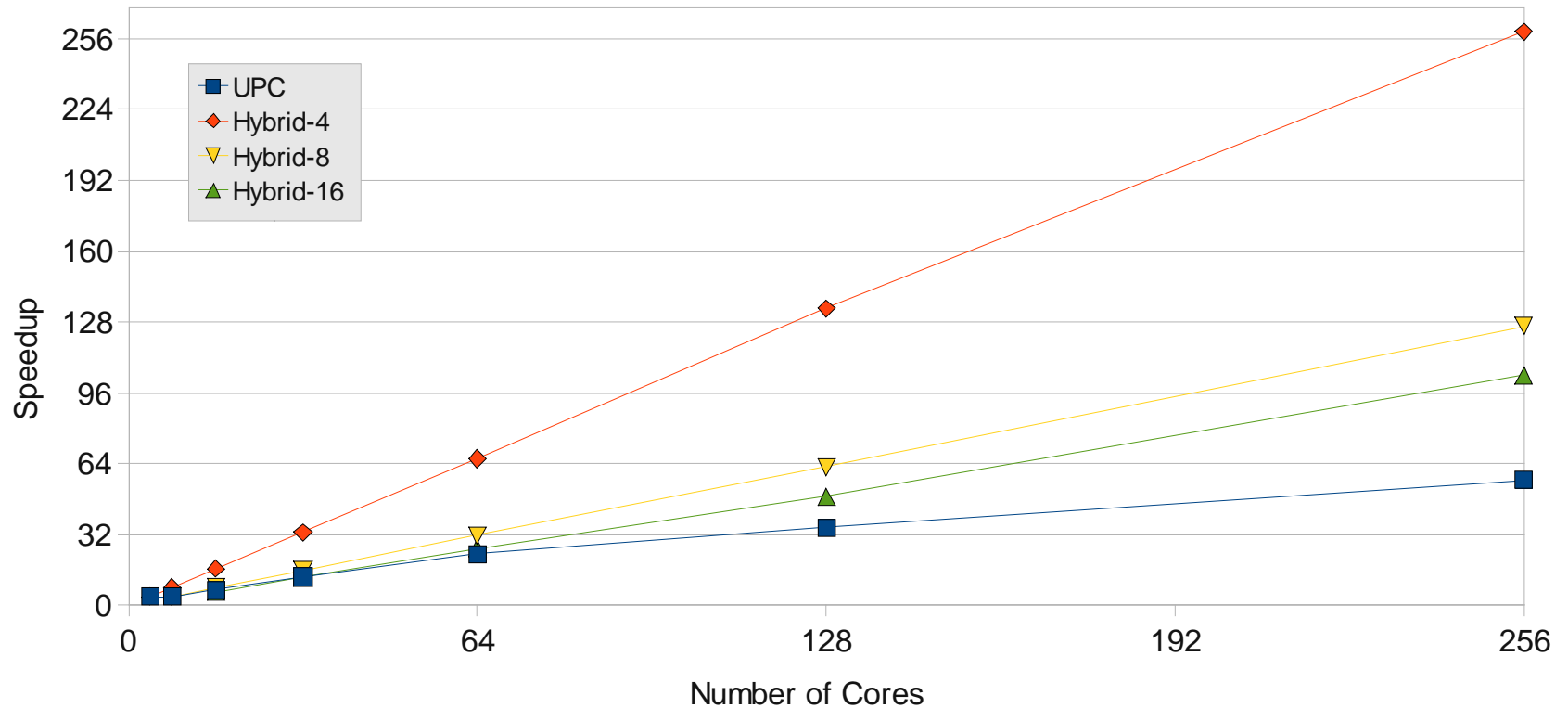
```
forall b in our_bodies  
  compute_forces(b, t)
```

Smaller distribution improves
 $O(\text{our_bodies})$ tree traversals

```
forall b in bodies  
  advance(b)
```

```
Allgather(bodies)
```

Barnes Force Computation



- Strong scaling: 100,000 body system