Argonne NATIONAL LABORATORY

... for a brighter future

U.S. Department of Energy

UChicago ▶ Argonne LLC

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

Hybrid Parallel Programming with MPI and Unified Parallel C

James Dinan

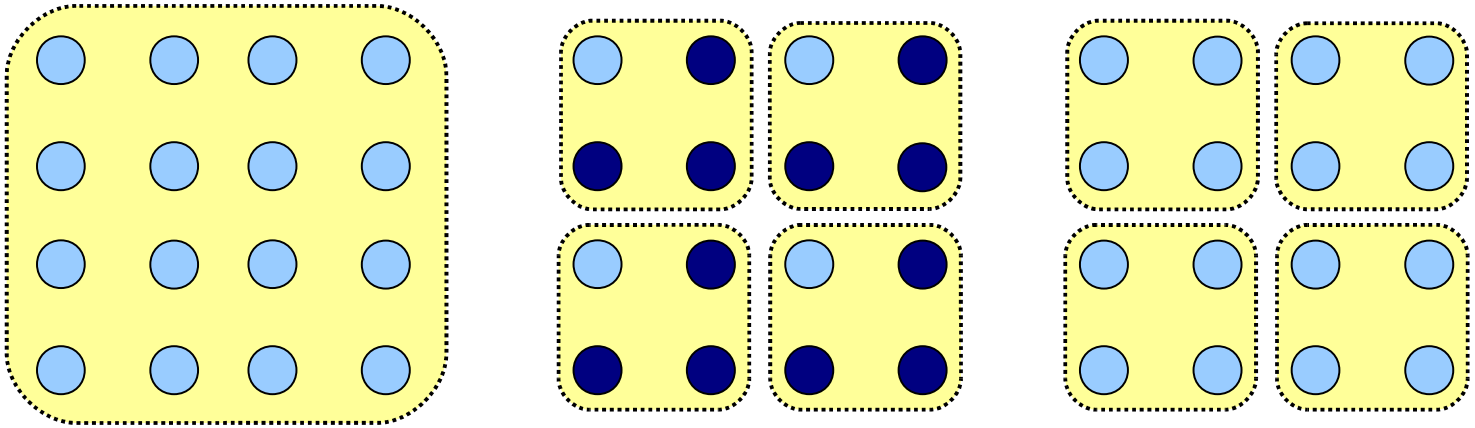
PhD Student, The Ohio State University
Advisor: Prof. Sadayappan

Intern, MCS Division
Host: Pavan Balaji

Hybrid MPI+UPC Execution Model

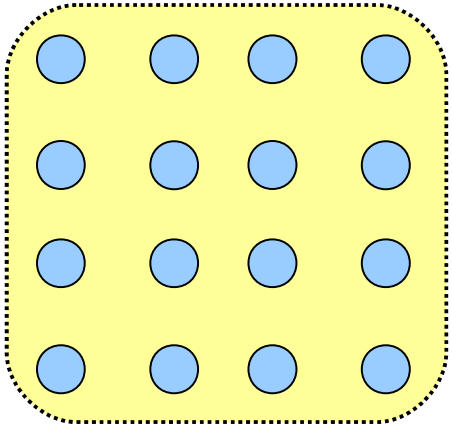
- Want to launch multiple UPC groups
- How many groups?
- How many MPI ranks per group?

Hybrid MPI+UPC Process
UPC Process



SPMD 1:1 MPMD N:1 MPMD 1:1

SPMD Hybrid Model



- UPC Threads ↔ MPI Ranks 1:1
  - Every process can do both UPC and MPI
- Benefit: Use UPC and MPI features in the same program
- Some support from Berkeley UPC for this model
  - "upcc -uses-mpi" tells BUPC to initialize/play nice with MPI
- UPC Thread IDs and MPI ranks may differ
  - `MPI_Comm_split(key = MYTHREAD)`

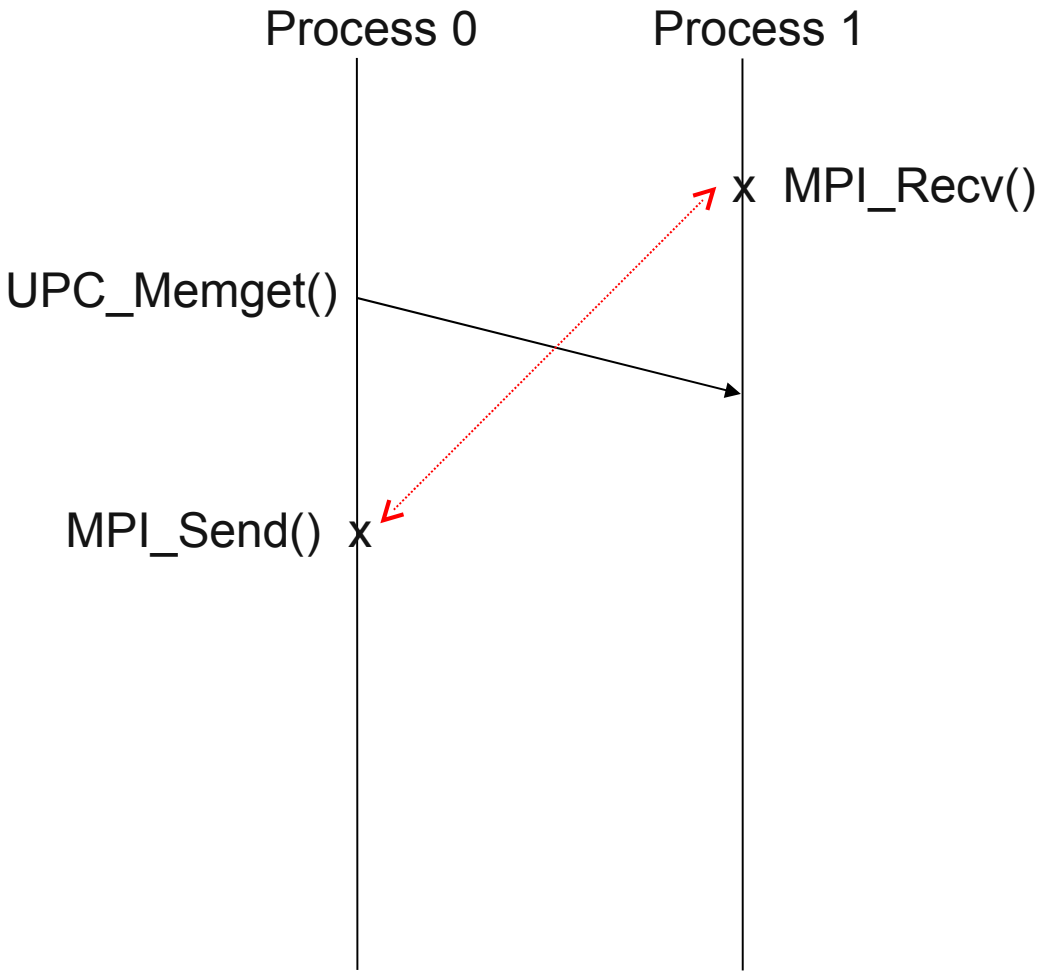
SPMD Hybrid Example: Vector dot product

```
#include <upc.h>
#include <mpi.h>
#define N 100*THREADS

shared double v1[N], v2[N];

int main(int argc, char **argv) {
  int i, rank;
  double sum = 0.0, dotp;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  upc_forall(i = 0; i < N; i++, i)
    sum += v1[i]*v2[i];


  MPI_Reduce(&sum, &dotp, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
  if (rank == 0) printf("Dot product = %f\n", dotp);.

  MPI_Finalize();
  return 0;
}
```

Caveat: Communication Deadlock

- MPI only guarantees progress if you make MPI calls
- UPC spec not specific on progress model
  - Berkeley UPC needs you make UPC calls to make progress
    - *Compiler/user may inject bupc_poll() calls*
- Mixing MPI/UPC introduces deadlock situations
- Could be solved by enforcing independent progress
  - Needs to be done for both MPI and UPC
  - Has performance implications for non-hybrid codes
- Workaround: Barrier synchronization between phases
  - Ensure completion of communication

Deadlock Example

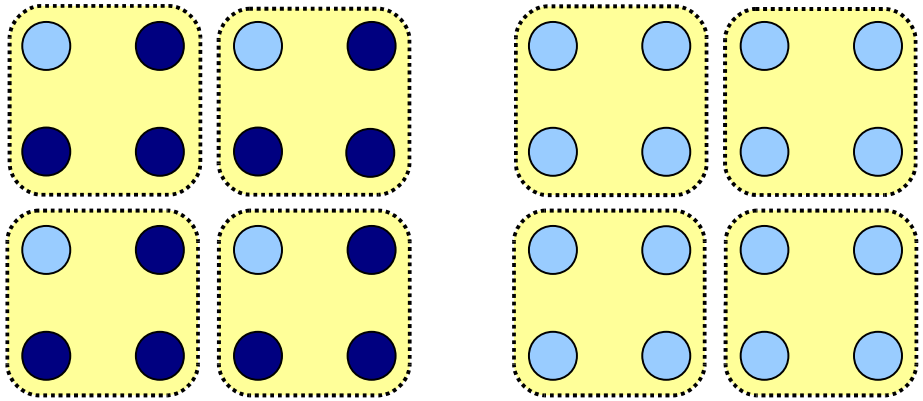


Passing Buffers Between UPC and MPI

- MPI doesn't know how to handle shared pointers
  - Can't change this in MPI
  - Don't want to require UPC compiler to build MPI

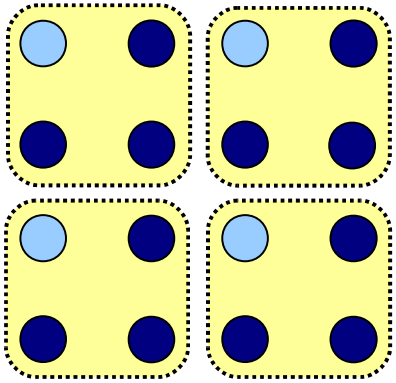
- User gives MPI a local buffer
  - Cast away sharedness if buf is local
  - or Get/Put remote data to/from a local buffer

MPMD Hybrid Model

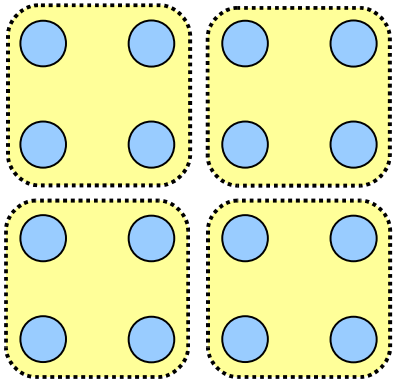


- Launch multiple UPC groups
  - Multiple global address spaces connected by MPI
- Useful for:
  - Scaling a UPC program that suffers from low locality
  - Scaling problem size of an MPI program
- Consider two cases:
  - Only thread 0 may perform MPI communication
  - All threads may perform MPI communcation

Mapping UPC Thread Ids to MPI Ranks



- How to identify a process?
  - Group ID
  - Group rank
- Group ID = MPI rank of thread 0
- Group rank = MYTHREAD



- Thread IDs not contiguous
  - Must be renumbered
- MPI_Comm_split(0, key)
- Key = MPI rank of thread 0 * THREADS + MYTHREAD
- Result is contiguous renumbering
  - MYTHREAD = MPI rank % THREADS
  - Group ID = Thread 0 rank = MPI rank/THREADS

Launching MPMD Hybrid Applications

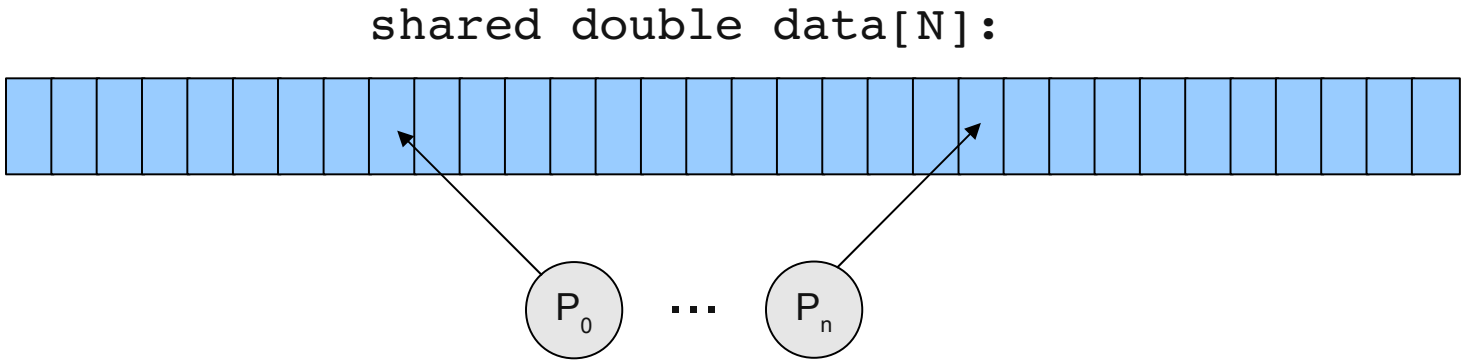
- Example: launch hybrid app with two UPC groups of size 8

```
$ mpiexec -env HOSTS=hosts.0 upcrun -n 8 hybrid-app
  : -env HOSTS=hosts.1 upcrun -n 8 hybrid-app
```

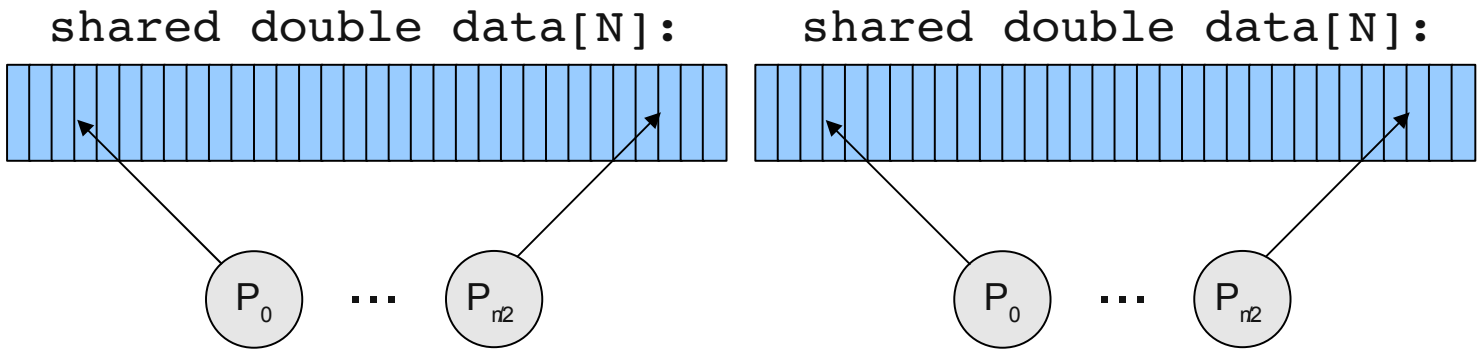
- Mpiexec launches two tasks
  - Each MPI task runs UPC's launcher
  - Provide different arguments (host file) to each task
- MPMD with all hybrid processes
  - Each instance of hybrid_app calls MPI_Init(), requests a rank
- *Problem:* MPI thinks it launched a two-task job!
- *Solution:*
  - Flag: --ranks-per-proc=8
  - Added to Hydra process manager in MPICH2

Random Access Benchmark

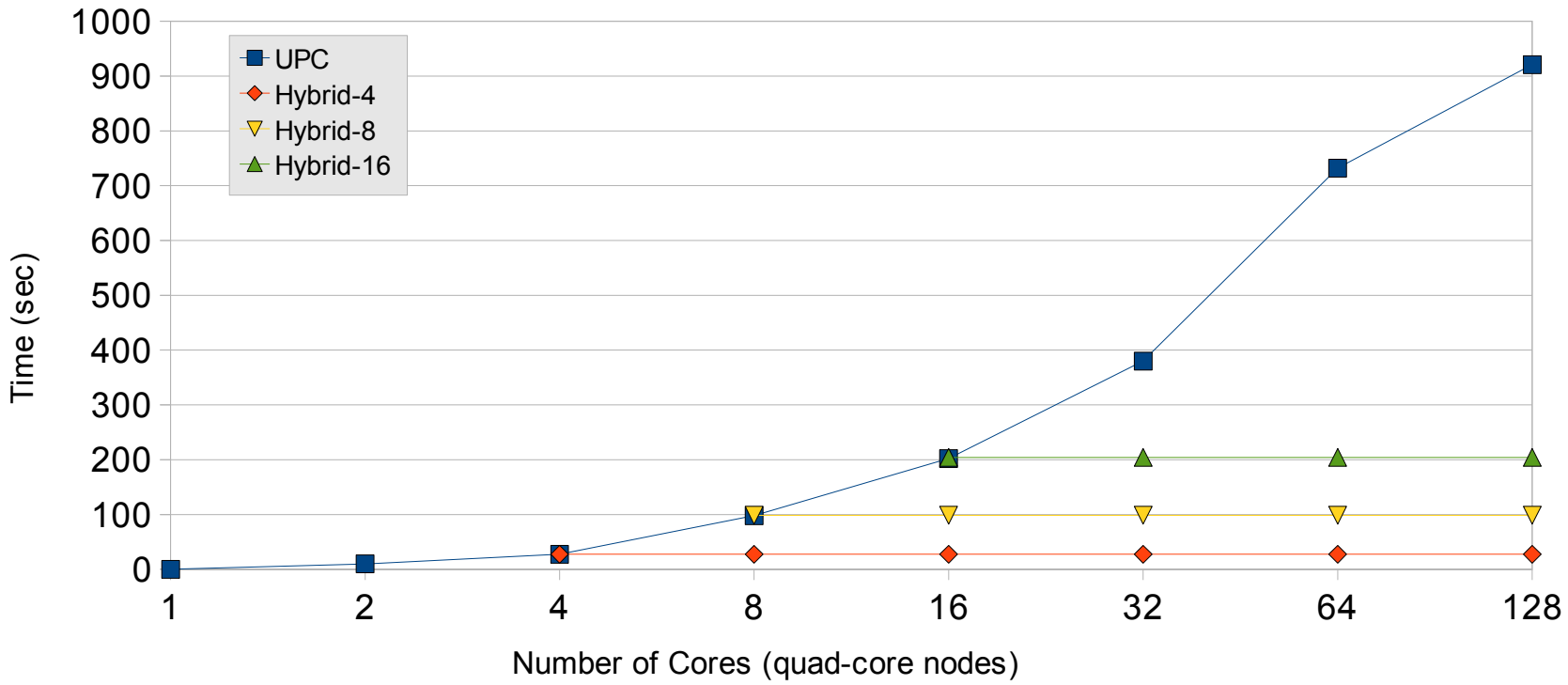
- UPC: Threads access random elements of distributed shared array



- Hybrid: Array is replicated on every group

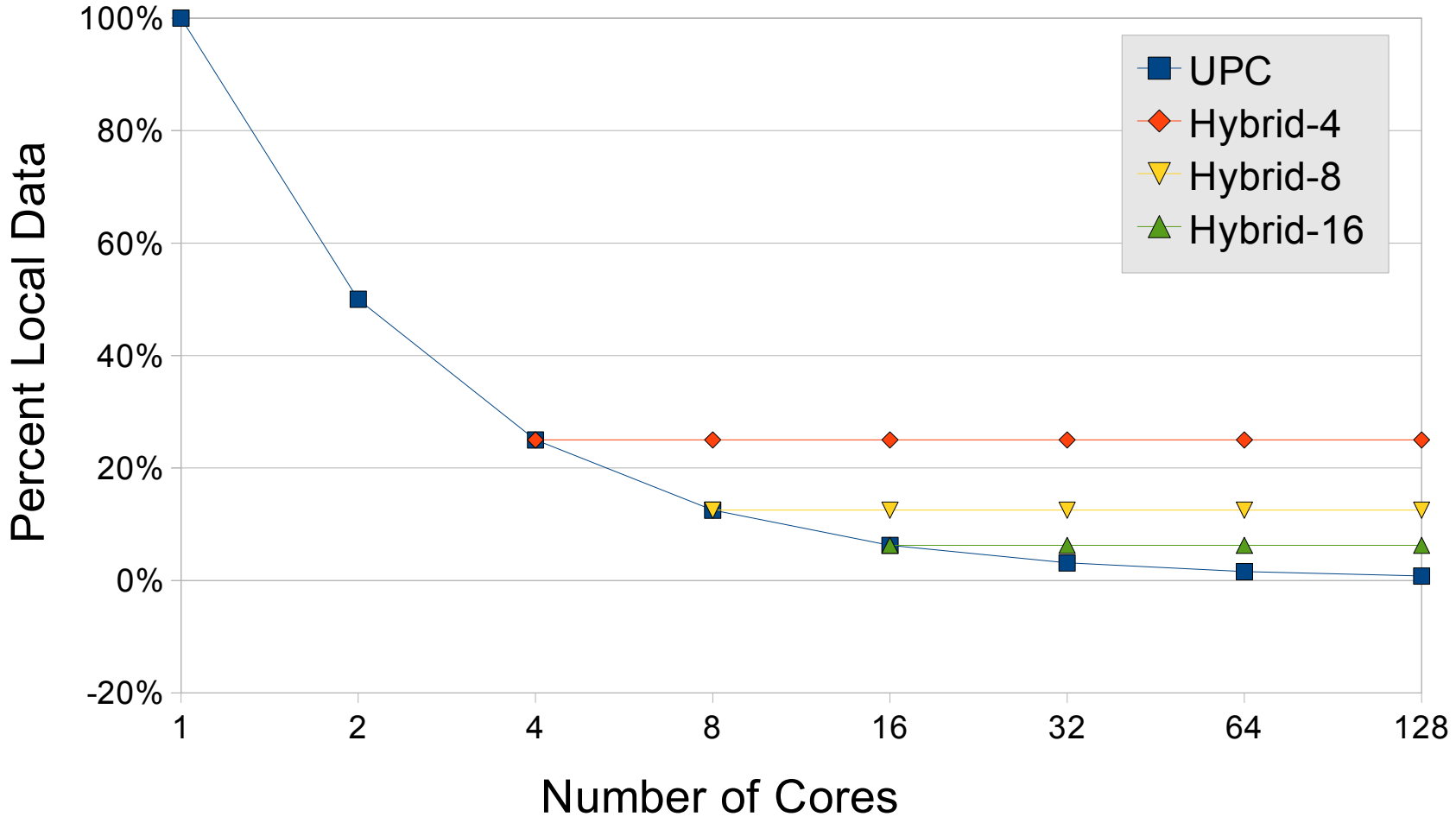


Impact of Data Locality on Performance



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

Percent Local References



Random Access Benchmark Takeaway

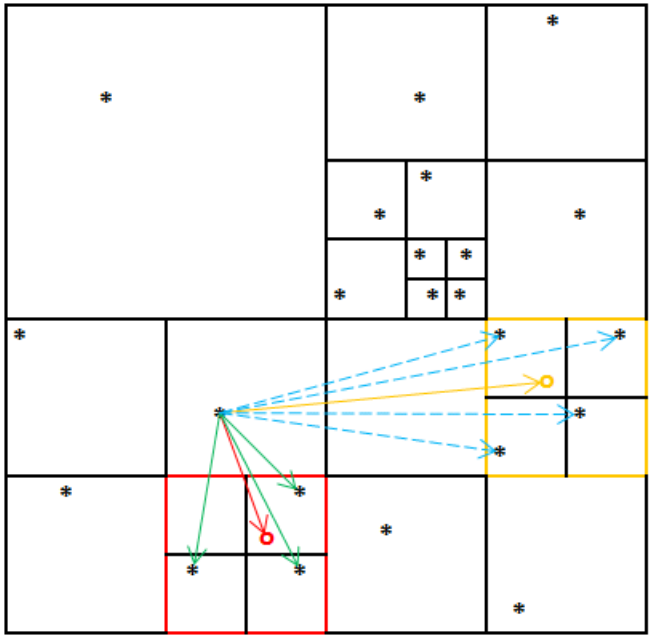
- Hybridization creates UPC groups
  - Improves locality, decreases communication
  - Replicate shared data on each group
  - Data replication is controlled by UPC group size

- Gap: UPC does not provide groups
  - UPC Teams have been proposed
  - Only in context of proposed collectives
  - Challenge: Teams are dynamic but data is static
    - *e.g. shared double data[N];*
  - Hybrid Model: Creates *static* groups, allowing grouping of *static* structures

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)

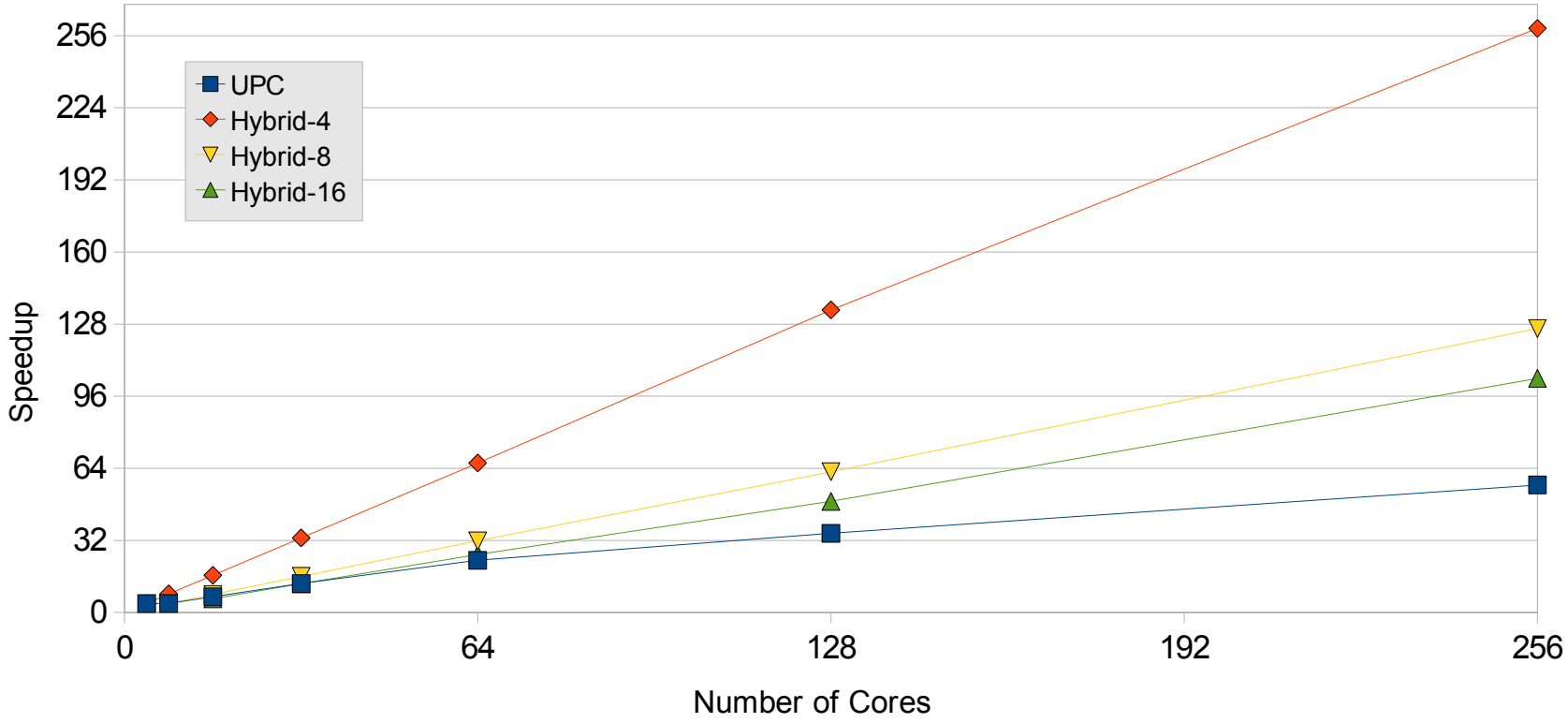
Hybrid Barnes Algorithm

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

  forall b in bodies
    insert(t, b)

  summarize_subtrees(t)
  our_bodies <- partion(group id, bodies)

  forall b in our_bodies
    compute_forces(b, t)

  forall b in bodies
    advance(b)

  Allgather(bodies)
```

Tree is distributed across group

Smaller distribution improves O(our_bodies) tree traversals

Barnes Force Computation



- Strong scaling: 100,000 body system