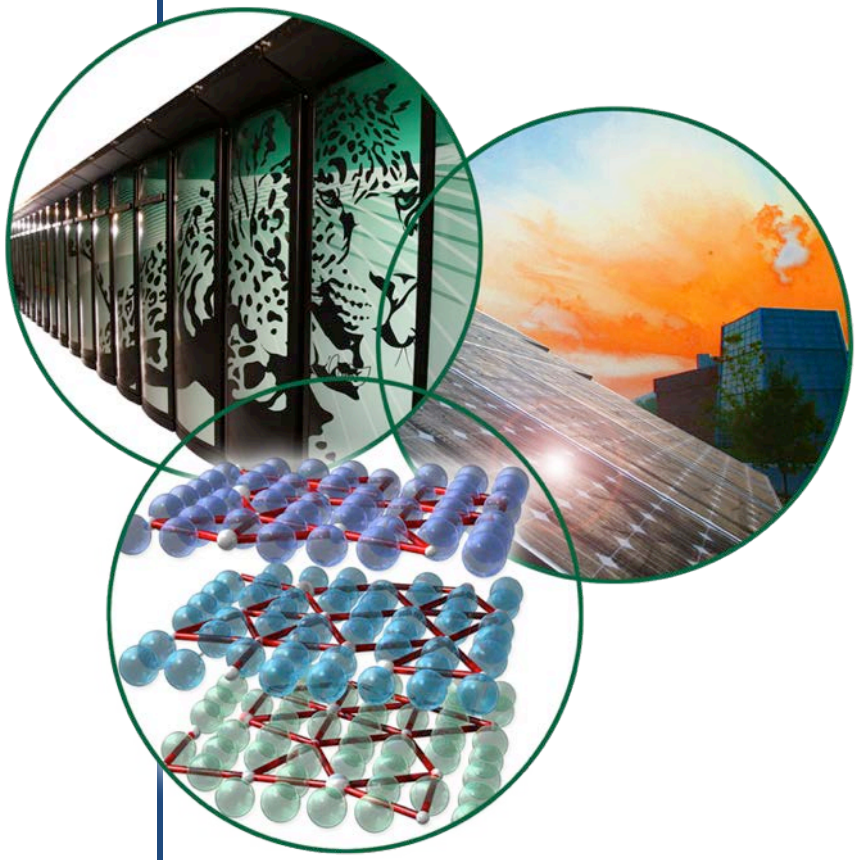
Hybrid Programming with OpenSHMEM

Matthew Baker



U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

What is hybrid programming?

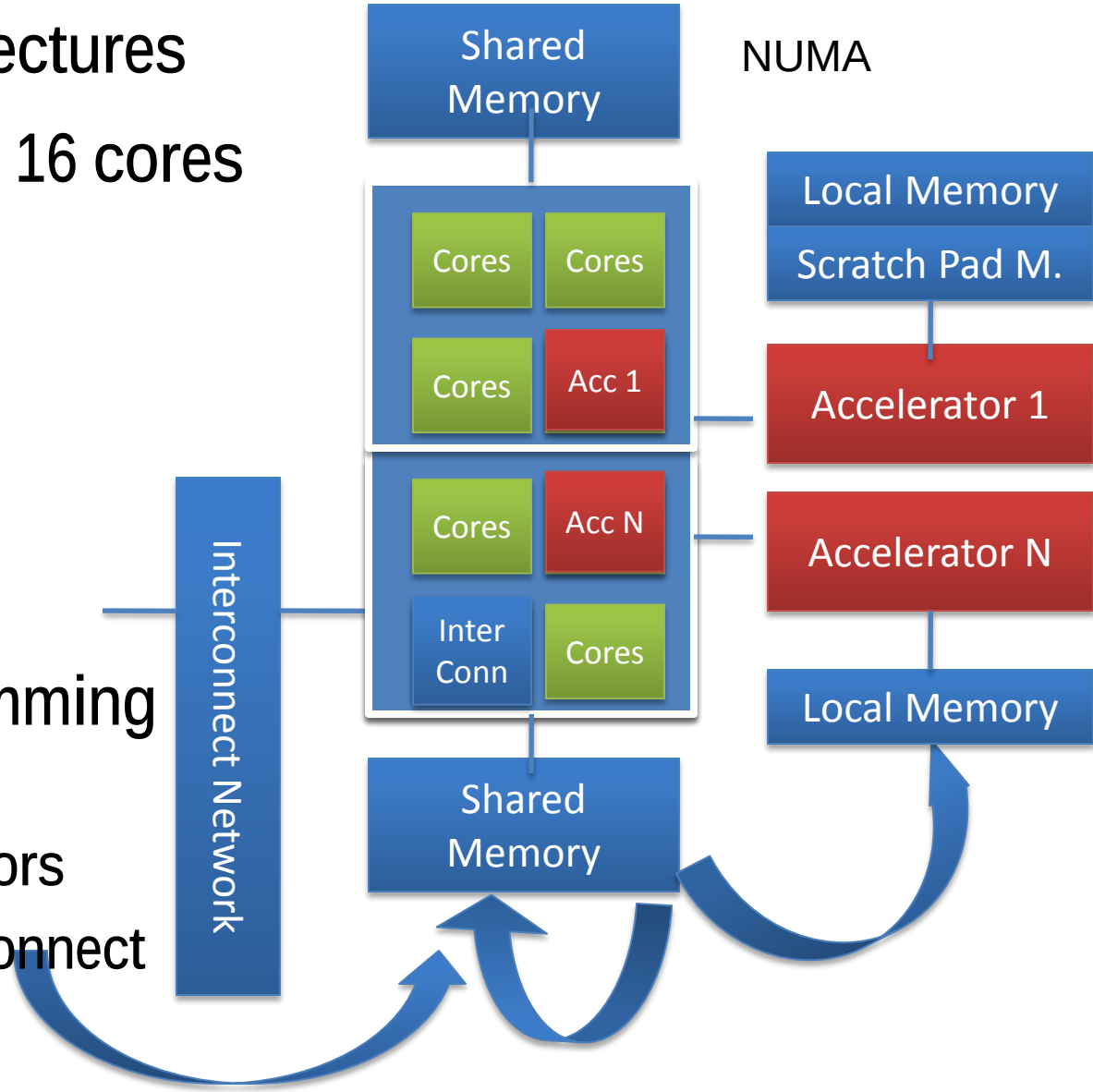
- Using more than one programming model
  - OpenSHMEM/OpenACC
- Different programming models to access different hardware

Why hybrid programming?

- Clusters are increasingly complex
  - Hardware limits demanded changes in computer organization
    - From single computers to network connected clusters
    - From single core per socket to multiple cores per socket
    - Now adding accelerators too
  - 4 of the top 10 Top500 super computers have accelerators
    - Including #1 and #2
    - HPL on titan used accelerators

Today's HPC nodes

- Multicore NUMA architectures
- Titan has 1 socket with 16 cores and attached GPU
- Three memory spaces
  - CPU
  - Accelerator
  - Remote cores
- Three different programming models
  - OpenACC for accelerators
  - OpenSHMEM for interconnect
  - OpenMP for cores



Current technologies

- OpenSHMEM
  - Efficient one sided communication library
- OpenACC
  - Simplest way to start accelerator programming
  - Emerging specification for accelerators
    - It's a step ahead of OpenMP and can work with PGAS languages
- Why these two?
  - We are in the process of extending OpenSHMEM
  - We have a strong presence in OpenACC.
  - We have a strong presence in both at ORNL

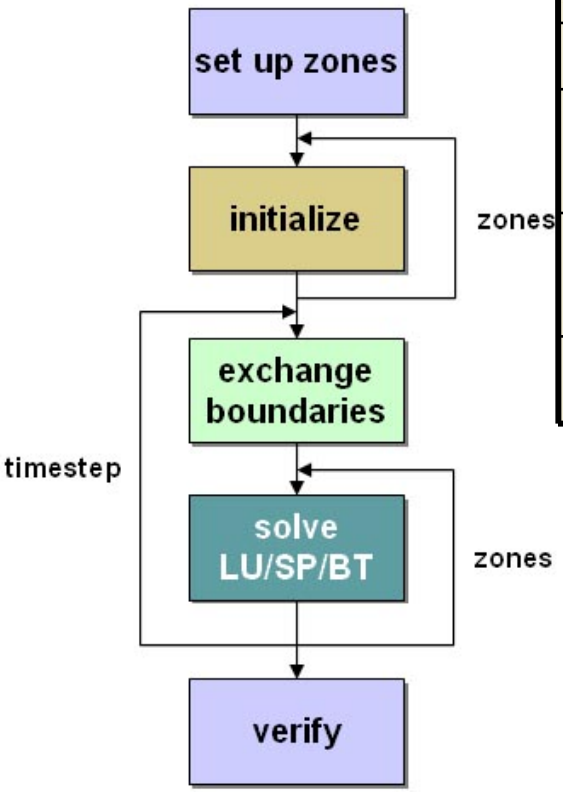
Principal focus

- Models need to be orthogonal
  - Different technologies addressing different problems
  - Models should strives to not interfere with each other
  - Must address sharing resources
  - Each model has different types of optimizations
    - How can we make them work together?

Hybrid Programming Bechmarks: NAS-BT-MZ

- Multizone block tri-diagnal solver
  - Selected because it is a hybrid benchmark
    - Uses MPI/OpenMP hybrid model, OpenMP only
  - Provides a nice split between communication and computation
  - Limited zone adjustability
  - Uses master-only mode for communication
    - Communication happens outside parallel region

NAS BT-MZ Benchmark



	MPI/OpenMP	OpenSHMEM/OpenMP	OpenSHMEM/ OpenACC
Time step	sequential	sequential	sequential
inter-zones	MPI Processes	SHMEM PEs	SHMEM PEs
exchange boundaries	MPI calls	SHMEM Put/Gets	SHMEM Put/Gets
intra-zones	OpenMP	OpenMP	OpenACC

- Multi-zone versions of the NAS Parallel Benchmarks LU,SP, and BT
- Two hybrid sample implementations
- Load balance heuristics part of sample codes
- www.nas.nasa.gov/Resources/Software/software.html

Porting BT-MZ to OpenSHMEM / OpenACC

- Data allocation in accelerator:

```
#pragma acc data create(forcing[0:size5],rho_i[0:size],u[0:size5]), &
#pragma acc      create(us[0:size],vs[0:size],ws[0:size]), &
#pragma acc      create(square[0:size],qs[0:size],rhs[0:size5])
{
      for (iz = 0; iz < proc_num_zones; iz++) {
        zone = proc_zone_id[iz];

        initialize(&u[start5[iz]],
                    nx[zone], nxmax[zone], ny[zone], nz[zone]);
        exact_rhs(&forcing[start5[iz]],
                   nx[zone], nxmax[zone], ny[zone], nz[zone]);
 }
```

Porting BT-MZ to OpenSHMEM / OpenACC

- Accelerating x_solve, mirroring data between cpu/gpu

```
#pragma acc data copy(rho_ip[0:size],up[0:size5],qsp[0:size],rhsp[0:size5]), &
#pragma acc      copy(squarep[0:size]) pcreate(lhsX,fjacX,njacX)
{
  #pragma acc kernels loop independent private(tmp1, tmp2, tmp3)
  for (k = 1; k <= gp22; k++) {
    #pragma acc loop independent
    for (j = 1; j <= gp12; j++) {
      for (i = 0; i <= isize; i++) {
        tmp1 = rho_i(i,j,k);
        tmp2 = tmp1 * tmp1;
        tmp3 = tmp1 * tmp2;
        //-----------------------------------------------------------------
        //
        //-----------------------------------------------------------------
        fjacX[0][0][i][j][k] = 0.0;
        fjacX[0][1][i][j][k] = 1.0;
        fjacX[0][2][i][j][k] = 0.0;
        fjacX[0][3][i][j][k] = 0.0;
        fjacX[0][4][i][j][k] = 0.0;
```

Porting BT-MZ to OpenSHMEM /OpenACC

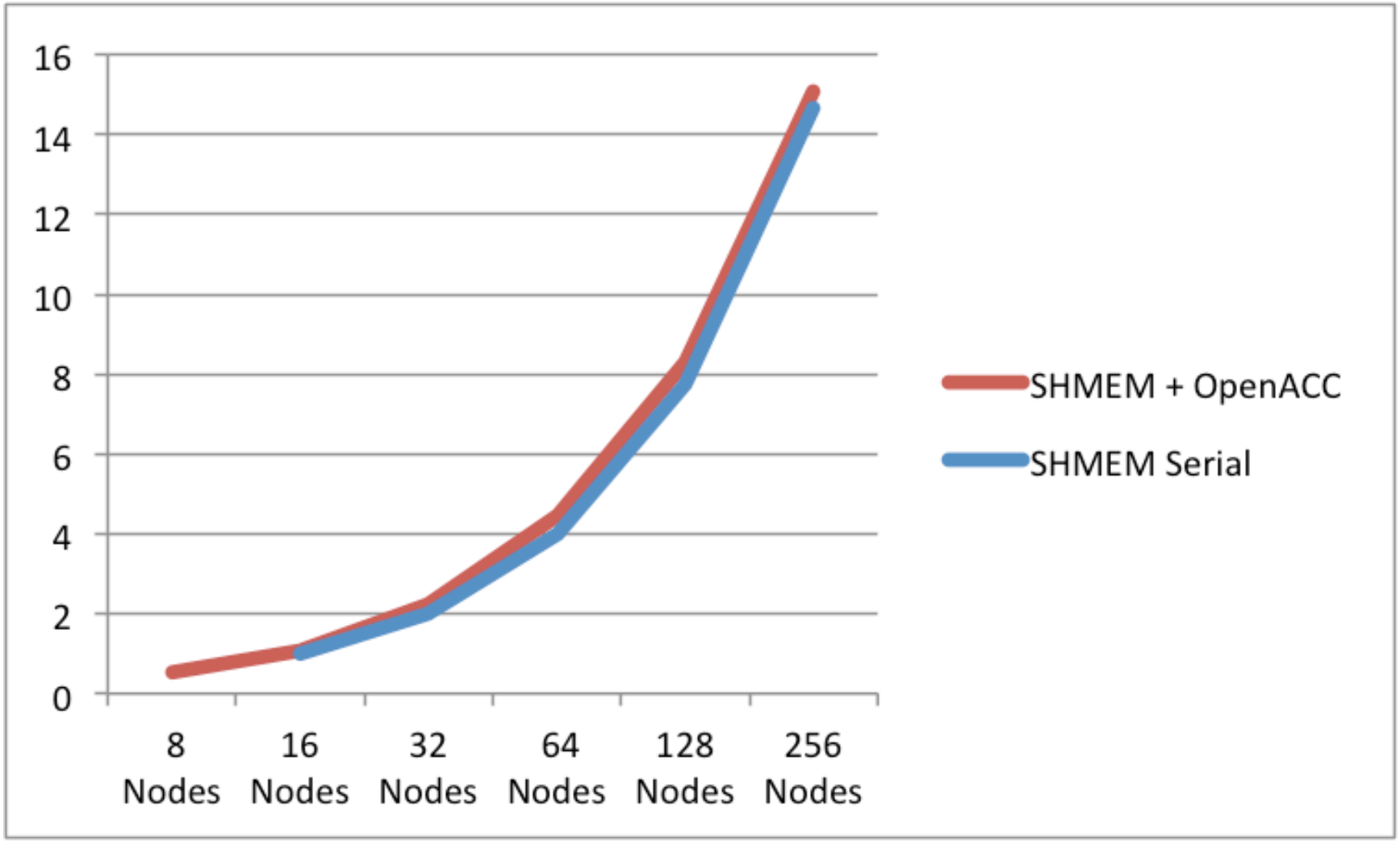
- Boundary exchange with OpenSHMEM

```
for (n = 0; n < num_msgs; n++) {
    if (nr >= MAX_REQS) {

    shmem_quiet();
    nr = 0;
    tag = MSG_TAG;
    }

    if (qoffset+m_size > qcomm_size[ip]) {
    m_size = qcomm_size[ip] - qoffset;
    }
    int iterator =0;
    // PE will have to send the data as well as the offset
    // sent so that the destination PE may calculate the new offset.
    shmem_putmem(&dest_qoffset,&qoffset,sizeof(idx_t), ip);
    shmem_fence();
    long x = 1;

     …..

   shmem_double_put(&qbc_in[dest_qoffset], &qbc_ou[qoffset], m_size, ip);
    shmem_quiet();
        nr = nr + 2;
    qoffset = qoffset + m_size;
    tag = tag + NUM_PROCS;
}
```

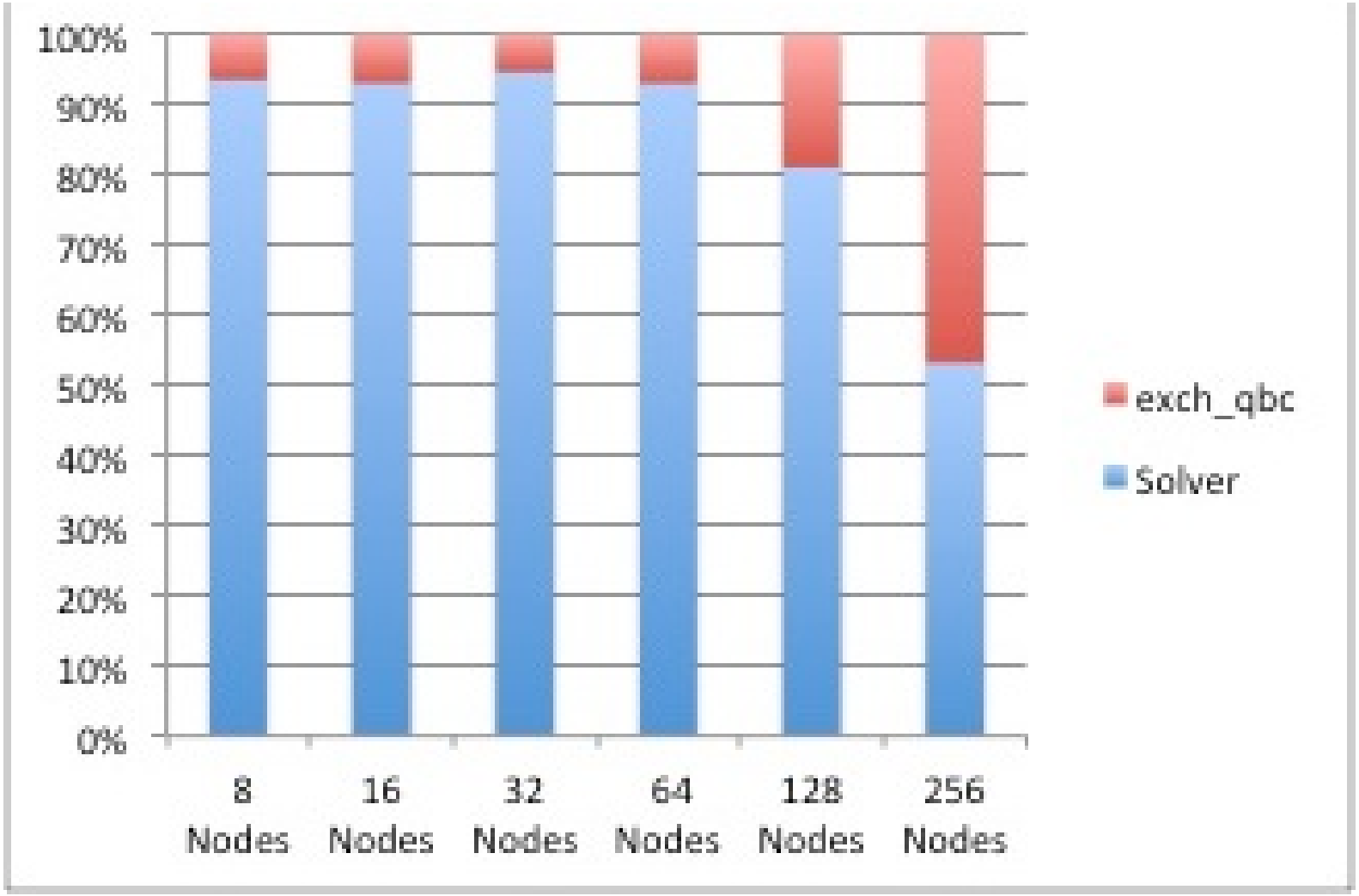
Conclusions

- Some gains in performance
  - 1 zone 1 node with accelerator saw performance improvement
    - Up to 4.3x faster
  - OpenSHMEM with no OpenACC saw consistent performance improvement
  - OpenSHMEM with OpenACC saw 10% performance gain
    - Large amounts of time spent transferring memory to and from accelerators
    - 256 nodes with OpenSHMEM without OpenACC
      - Time spent exchanging boundaries was 3% of run time
    - 256 nodes with OpenSHMEM and OpenACC
      - Time spent exchanging boundaries was 46% of run time

Graphs


16
14
12
10
8
6
4
2
0
8 Nodes
16 Nodes
32 Nodes
64 Nodes
128 Nodes
256 Nodes
SHMEM + OpenACC
SHMEM Serial

Communication vs compute



Conclusions

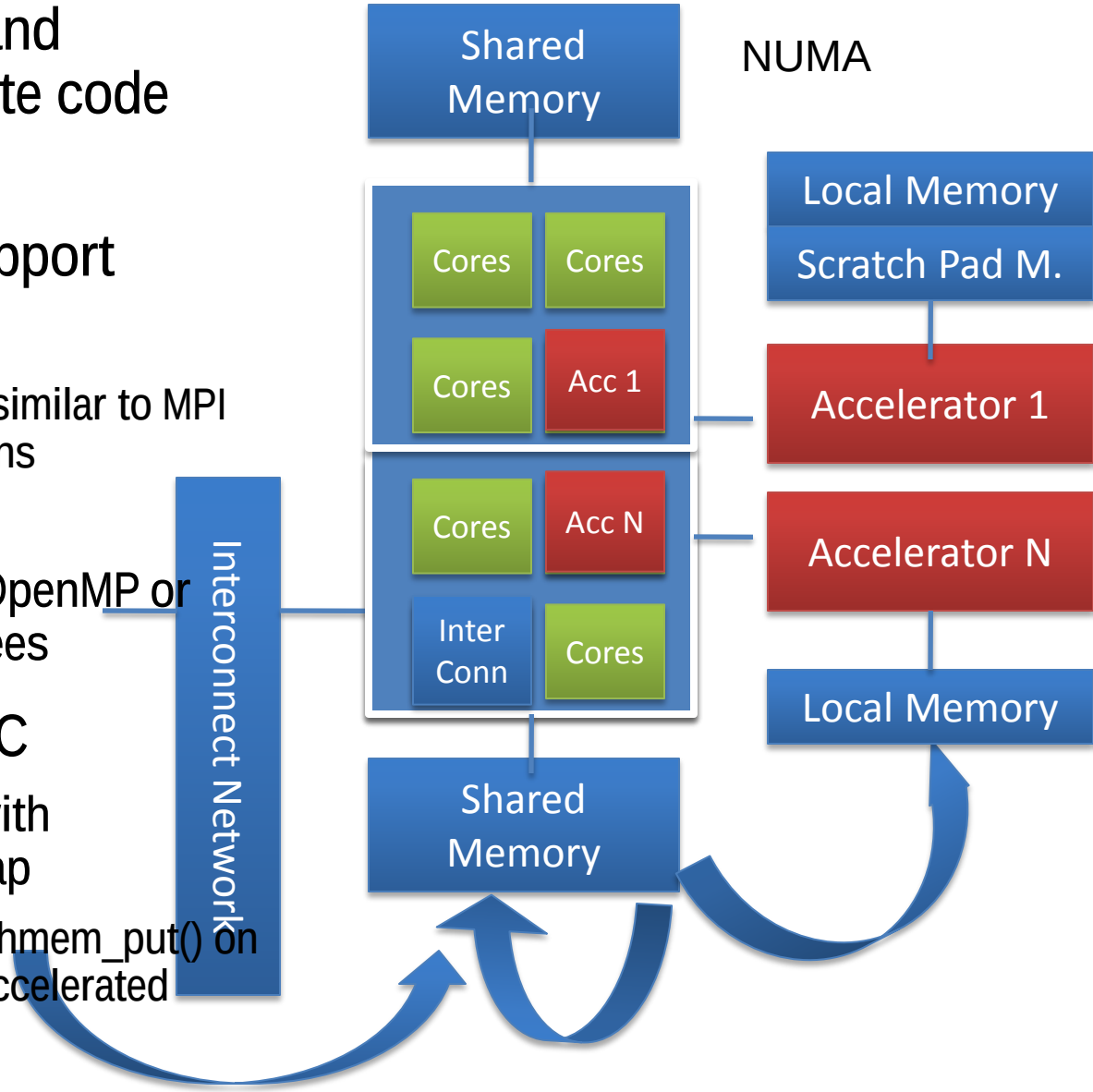
- Problems
  - Lots of small transfers
  - Irregular transfers
  - 13x8x28 to 57x28x38 element zones (all under 500KB)
- Solutions
  - APU
    - Shared memory between accelerator and host would eliminate PCIe bus transfers
  - Network access for GPU
    - Remove need to transfer memory to host

Conclusions

- None of the standards specified interactions
  - Solved in this benchmark by having separate steps
    - OpenSHMEM in boundary exchanges
    - OpenACC in compute sections
    - No threading

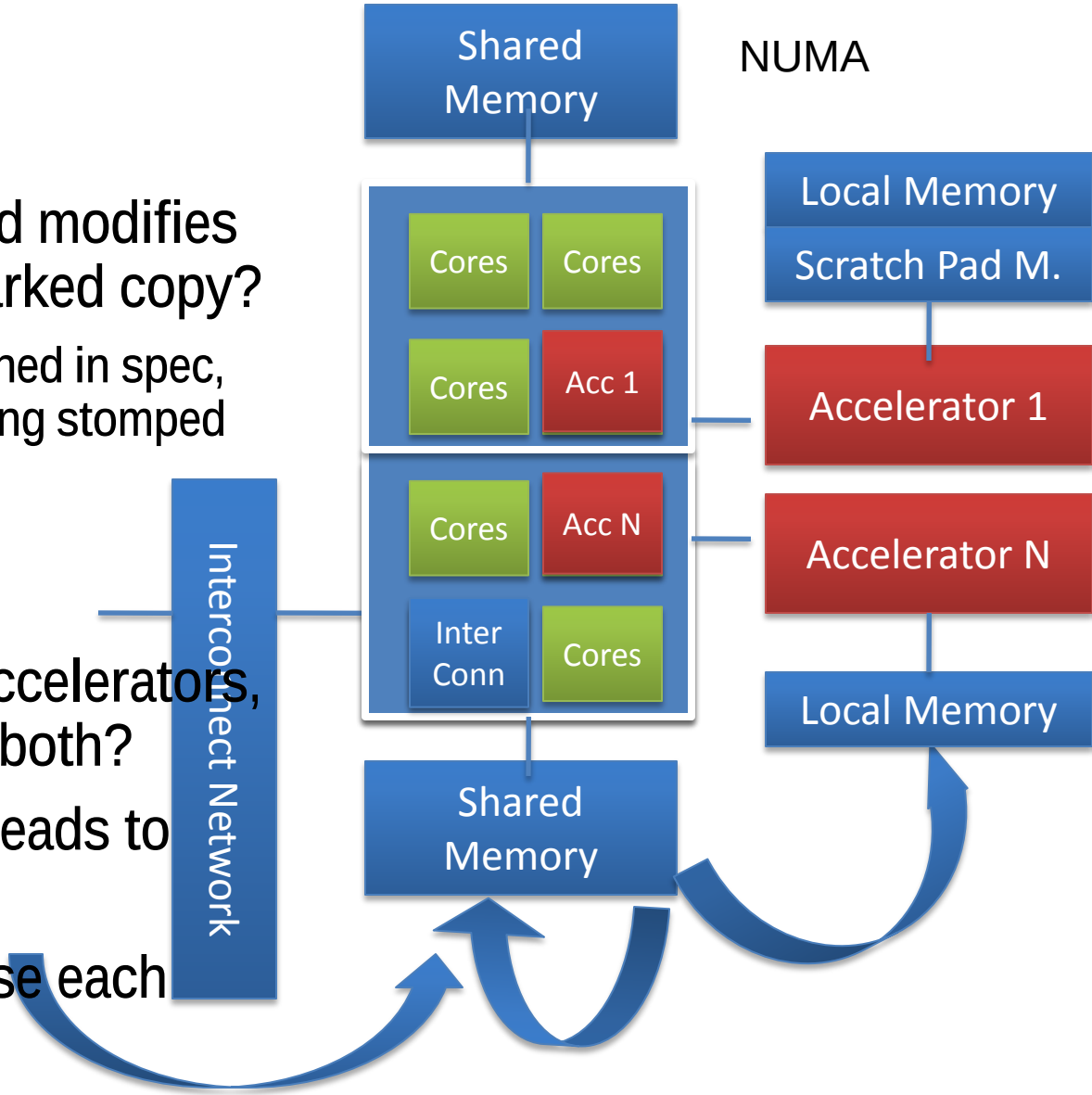
Open questions

- Model works if OpenACC and OpenSHMEM are in separate code regions.
- OpenSHMEM threading support
  - Currently: Nothing
    - Cray has a proposal that is similar to MPI spec with additional functions
    - Is this sufficient?
  - Can't use with pthreads or OpenMP or OpenACC with any guarantees
- OpenSHMEM and OpenACC
  - OpenACC parallel regions with OpenSHMEM symmetric heap
    - What happens if there is a shmem_put() on a variable in an OpenACC accelerated region?



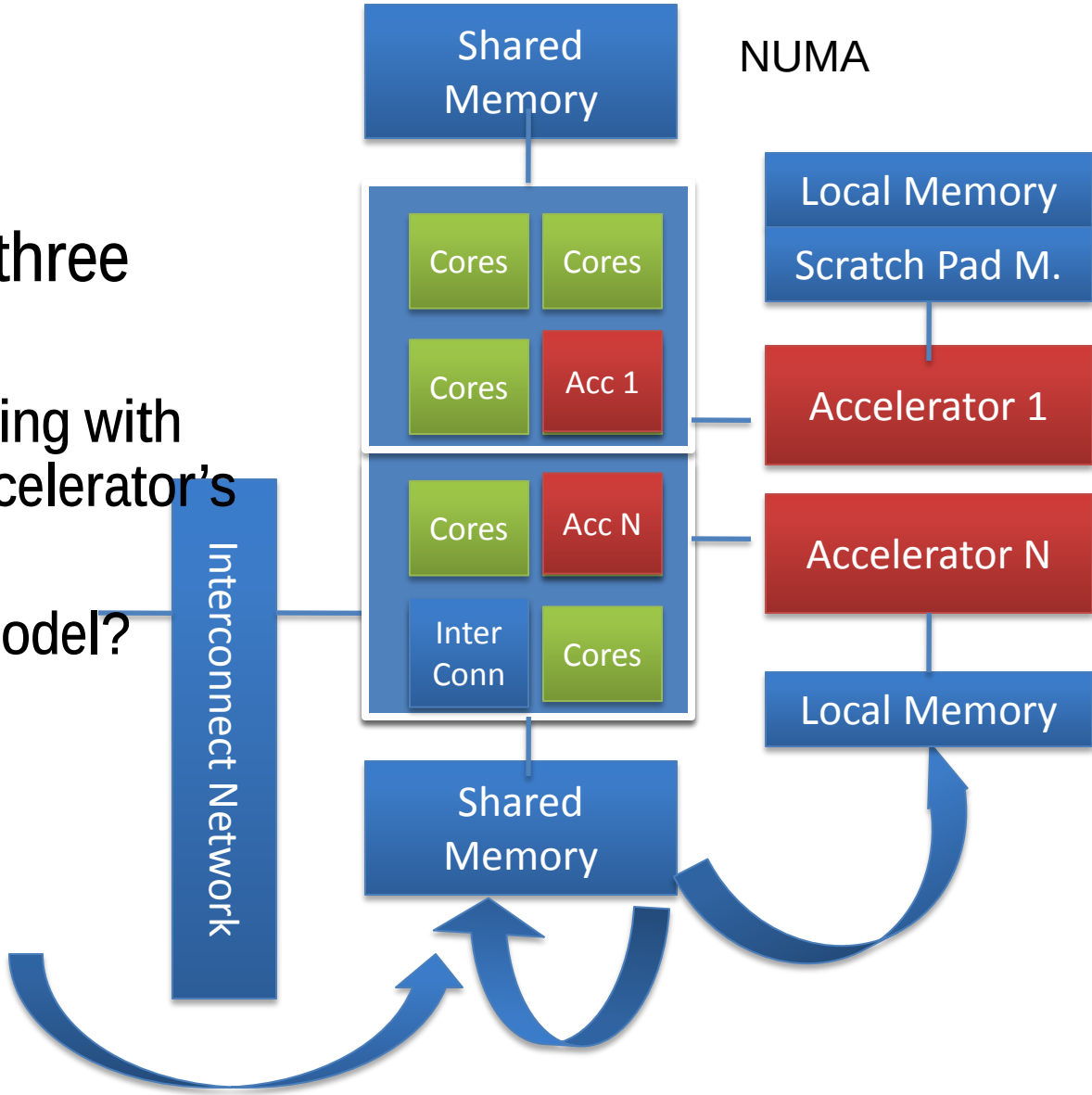
Open questions

- OpenACC threading
  - Currently: Nothing
  - What happens if a thread modifies data in a data region marked copy?
    - Could be explicitly undefined in spec, with an expectation of being stomped
- OpenACC with multiple accelerators
  - Suppose you had two accelerators, can a single thread use both?
  - Must you spawn two threads to use each one?
  - Must you fork/exec to use each one?



Open questions

- OpenMP is still relevant
  - Multiple cores per node
- How do we integrate all three into a triple hybrid?
  - Multiple threads contending with shmem puts/gets and accelerator's separate memory
  - Chimera programming model?



Additional Future Work

- All standards need to interact in well defined ways
  - Needs some definition of memory consistency
- Possibilities of standards working together?
  - Can we get OpenACC to call shmem functions?
    - Like MPI on GPUs?
    - Accelerators tapping straight into networking hardware
      - GPUs with networking hardware?
  - Can we do OpenSHMEM put/gets to accelerator memories?
    - Extensions are needed to define different memory contexts
    - Puts/Gets from GPU to GPU
    - Puts/Gets from Host to remote GPU

Questions?