Toward Abstracting the Communication Intent in Applications to Improve Portability and Productivity

Tiffany M. Mintz, Oscar Hernandez, Christos Kartsaklis, David E. Bernholdt, Markus Eisenbach, Swaroop Pophale
Content

• Motivation
• Related Work
• Approach
• Design & Implementation
• Experimental Results
• Next Steps
Why This Matters

• Communication missing from compiler’s static analysis
  – Opportunities for automatic static optimization lost

• High level abstractions:
  – provide greater portability
  – enhances productivity
  – Easier to maintain
What Others Have Done: Bamboo

- Annotations for MPI library calls
- Provides mechanism for static analysis and communication/computation overlap
What Others Have Done: OpenMPI (not the library implementation)

- OpenMP-like directives for incremental parallelization
- Express collective communication

```c
#pragma ompi sync_sleeve
for (i = 1; i <= YSIZE; i++)
#pragma ompi for
for (j = 1; j <= XSIZE; j++)
    nu[i][j] = (u[i-1][j] + u[i+1][j] + u[i][j-1] + u[i][j+1]) / 4.0;

#pragma ompi for reduction(+:res2)
for (j = 1; j <= XSIZE; j++){
    tmp = (nu[i][j] - u[i][j]) / u[i][j];
    res2 += tmp * tmp;
}
#pragma ompi single
fprintf(stderr, "itr=%d res=%g\n", itr, res1);
```
Our Solution

- Use directive language extensions to assert communication
- Translate directives at compile time to communication calls

Assertion:
A complete sentence which expresses the point.
Asserting Communication

- **Directives:** `comm_p2p`, `comm_parameters`
- **Clauses:**
  - `sender`, `receiver`, `sbuf`, `rbuf`
  - `sendwhen`, `receivewhen`, `count`, `place_sync`, `max_comm_iter`, `target`
Examples

Ring communication pattern

```
prev = (rank-1+nprocs) % nprocs;
next = (rank+1) % nprocs;
#pragma comm_p2p sender(prev) receiver(next)\
   sbuf(buf1) rbuf(buf2)
```

Communication scoping and parameter inheritance

```
#pragma comm_params sender(rank-1)\
   receiver(rank+1) sendwhen(rank%2==0)\
   receivewhen(rank%2==1) count(size)\
   max_comm_iter(n) place_sync(END_PARAM_REGION)
{
    for(p=0; p < n; p++)
      #pragma comm_p2p sbuf(&buf1[p]) rbuf(&buf2[p])
}
```
Static Analysis and Optimizations

- Communication scoping
- Communication/computation overlap
- Flexible Implementation
- Data type handling
- Synchronization reduction

Open64 Compilation Phases
Scientific Application

WL-LSMS

• Wang-Landau (WL)
  – Monte-Carlo calculation

• Locally Self-Consistent Multiple Scattering (LSMS)
  – First principles electronic structure calculation

WL-LSMS Organizational View
WL-LSMS Communication

• Local Interaction Zone (LIZ)
  – Within each LSMS instance
  – Master-Worker process topology
  – Point-to-point communication

LIZ Communication Pattern
Single Atom Data Communication

Original communication source code

Communication using directives

```c
if(comm.rank==from)
{
    int pos=0;
    MPI_Pack (local_td,1, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.jwa,1, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.xstart,1, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.header, 80, MPI_CHAN, buf, &pos, comm.comm);
    MPI_Pack (atom.alat,1, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.formi,1, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.valf,1, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.ztot,1, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.evec,3, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.navc,3, MPI_PNT, buf, &pos, comm.comm);
    MPI_Pack (atom.nucl,1, MPI_INT, buf, &pos, comm.comm);
    t=atom.vr.n_row();
    MPI_Pack(t,1, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.vr(0,0), 2*t, MPI_DOUBLE, buf, &pos, comm.comm);
    MPI_Pack (atom.ztot(0,0), 2*t, MPI_DOUBLE, buf, &pos, comm.comm);
    t=atom.ec.n_row();
    MPI_Pack(t,1, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.ec(0,0), 2*t, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.nc(0,0), 2*t, MPI_INT, buf, &pos, comm.comm);
    MPI_Pack (atom.kc(0,0), 2*t, MPI_INT, buf, &pos, comm.comm);
    MPI_Send (buf, size, MPI_PACKED, to, 0, comm.comm);
}
if(comm.rank==to)
{
    MPI_Status status;
    MPI_Recv (buf, size, MPI_PACKED, from, 0, comm.comm, &status);
    int pos=0;
    MPI_Unpack (buf, &pos, &local_td,1, MPI_INT, comm.comm);
    MPI_Unpack (buf, &pos, &atom.jwa,1, MPI_INT, comm.comm);
    MPI_Unpack (buf, &pos, &atom.xstart,1, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.header, 80, MPI_CHAN, comm.comm);
    MPI_Unpack (buf, &pos, &atom.alat,1, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.formi,1, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.valf,1, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.ztot,1, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.evec,3, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.navc,3, MPI_PNT, comm.comm);
    MPI_Unpack (buf, &pos, &atom.nucl,1, MPI_INT, comm.comm);
    if(t<atom.vr.n_row())
        atom.realizePotential(t-50);
    MPI_Unpack (buf, &pos, &atom.vr(0,0), 2*t, MPI_DOUBLE, comm.comm);
    MPI_Unpack (buf, &pos, &atom.dzhotot(0,0), 2*t, MPI_DOUBLE, comm.comm);
    if(t<atom.ec.n_row())
        atom.realizeCore(t);
}
```
Performance Comparison

- Experiments using MPI and SHMEM translations
- Performance comparable to original source code
Spin Configurations Communication

Original communication source code

communication/computation overlap using directives
Communication Comparison

- Original synchronization caused poor performance
- After modifying synchronization:
  - MPI translation 1.4x speedup
  - SHMEM translation 14.5x speedup
Communication/Computation Overlap

- Current Computation/Communication ratio: 19 to 1
- Estimate 10x speedup with GPU acceleration
Looking Forward

• **Summary**
  – Higher abstraction for message passing communication
  – Communication aware compiler
  – Static analysis and optimization for message passing

• **What’s Next**
  – Develop assertions for many-to-one, one-to-many patterns
  – Extend data flow analysis
  – Implement cost model for automated selection of communication calls
QUESTIONS?