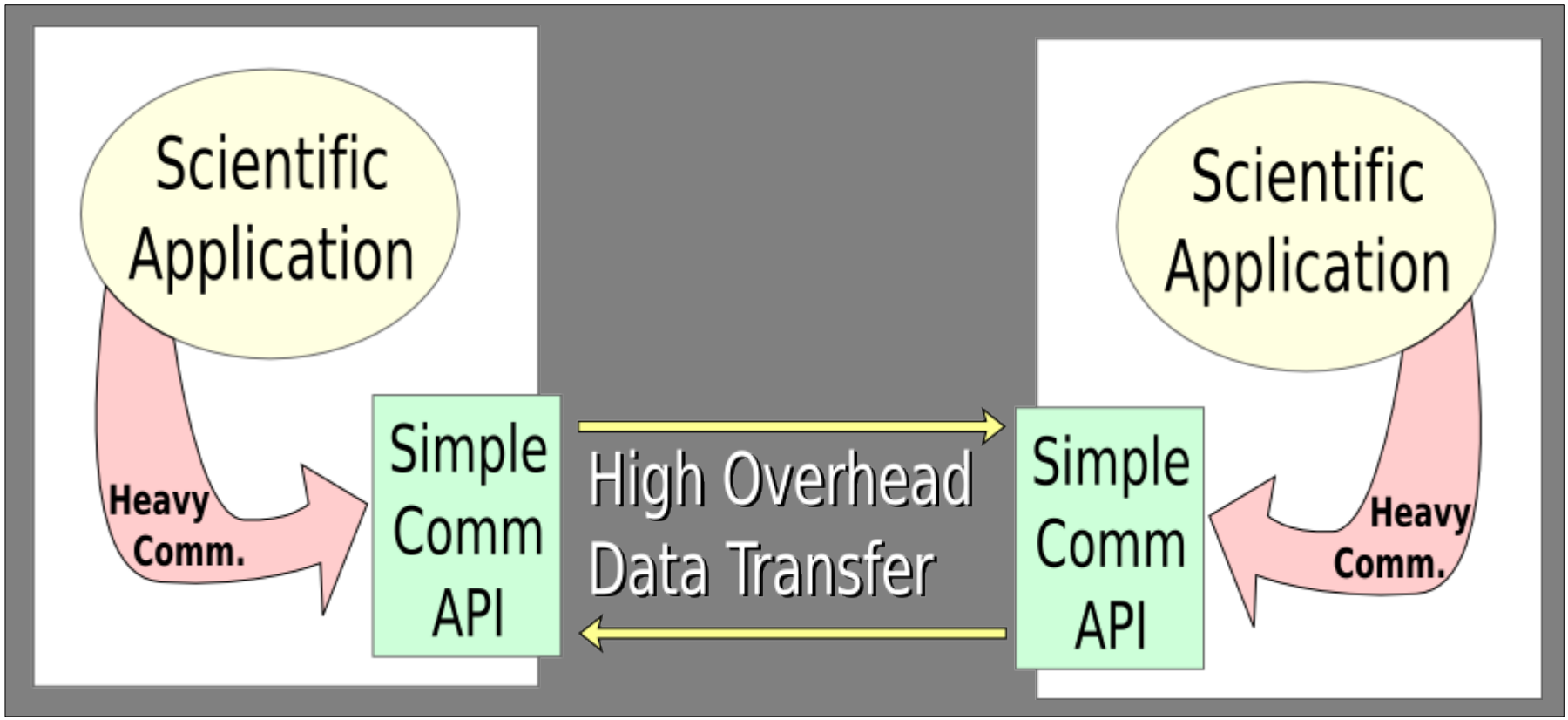
Implementing an **Open64**-based tool for improving the performance of **MPI** programs

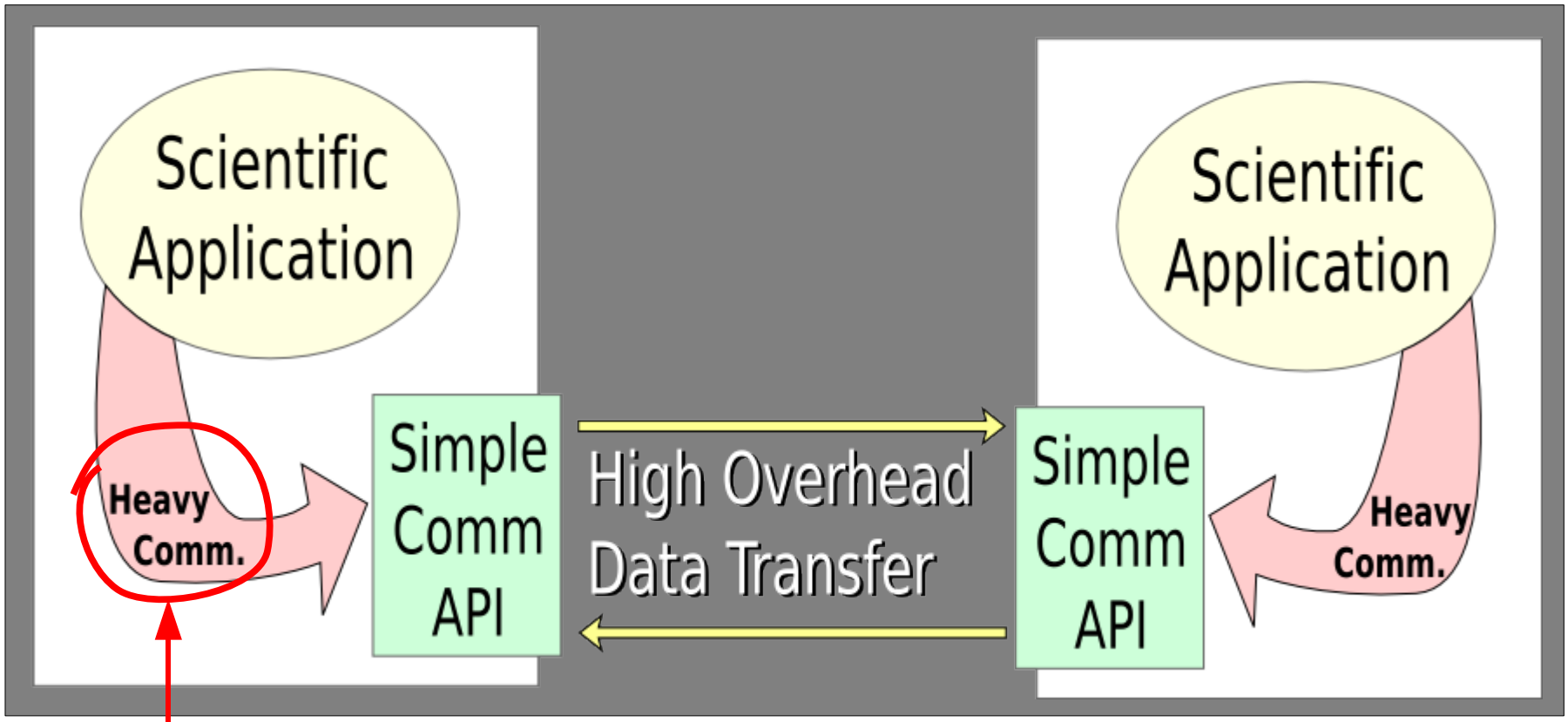
Anthony Danalis **Lori Pollock** **Martin Swany** **John Cavazos**

University of Delaware

Problem:
Communication Overhead in Cluster Computing

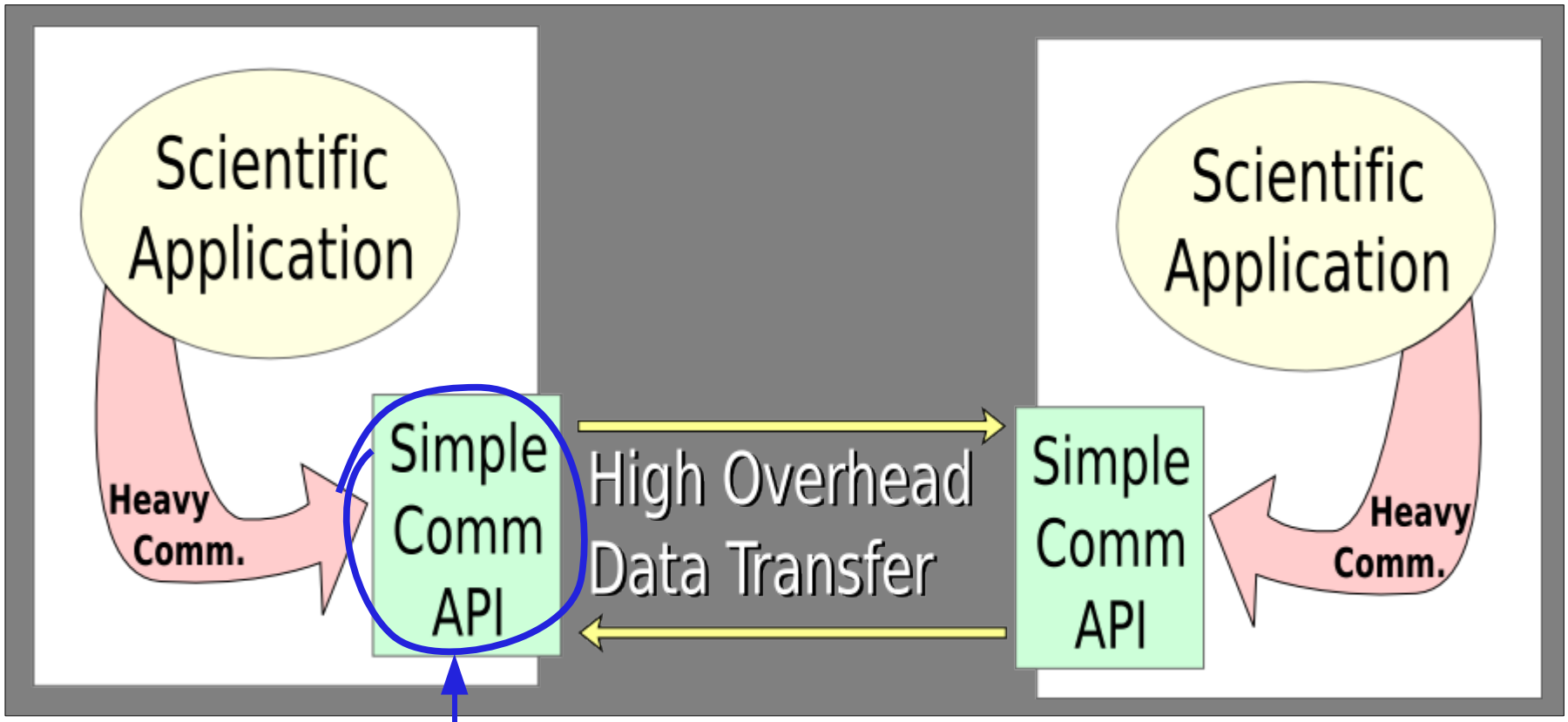


Problem:
Communication Overhead in Cluster Computing



- Inherent to the Application
- Hard or impossible to change

Problem:
Communication Overhead in Cluster Computing



- Communication Library & Developer Expertise dependent
- Amendable

Overall Research Goal

Requirements:

- ✔ Achieve high-performance communication
- ✔ Simplify the MPI code developers write

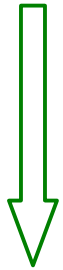
Have your cake
+
Eat your cake

Overall Research Goal

Requirements:

- ✔ Achieve high-performance communication
- ✔ Simplify the MPI code developers write

Have your cake
+
Eat your cake



Automatic cake making machine

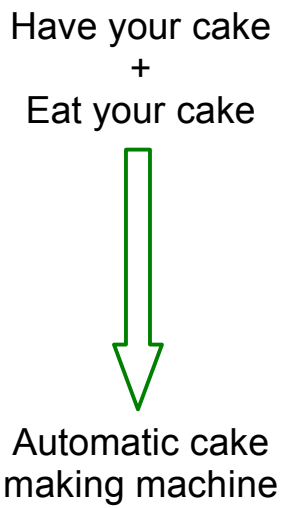
Overall Research Goal

Requirements:

- ✔ Achieve high-performance communication
- ✔ Simplify the MPI code developers write

Proposed Solution:

An **automatic** system that **transforms** simple communication code into efficient code.

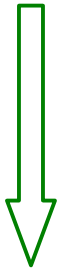


Overall Research Goal

Requirements:

- ✔ Achieve high-performance communication
- ✔ Simplify the MPI code developers write

Have your cake
+
Eat your cake



Automatic cake making machine

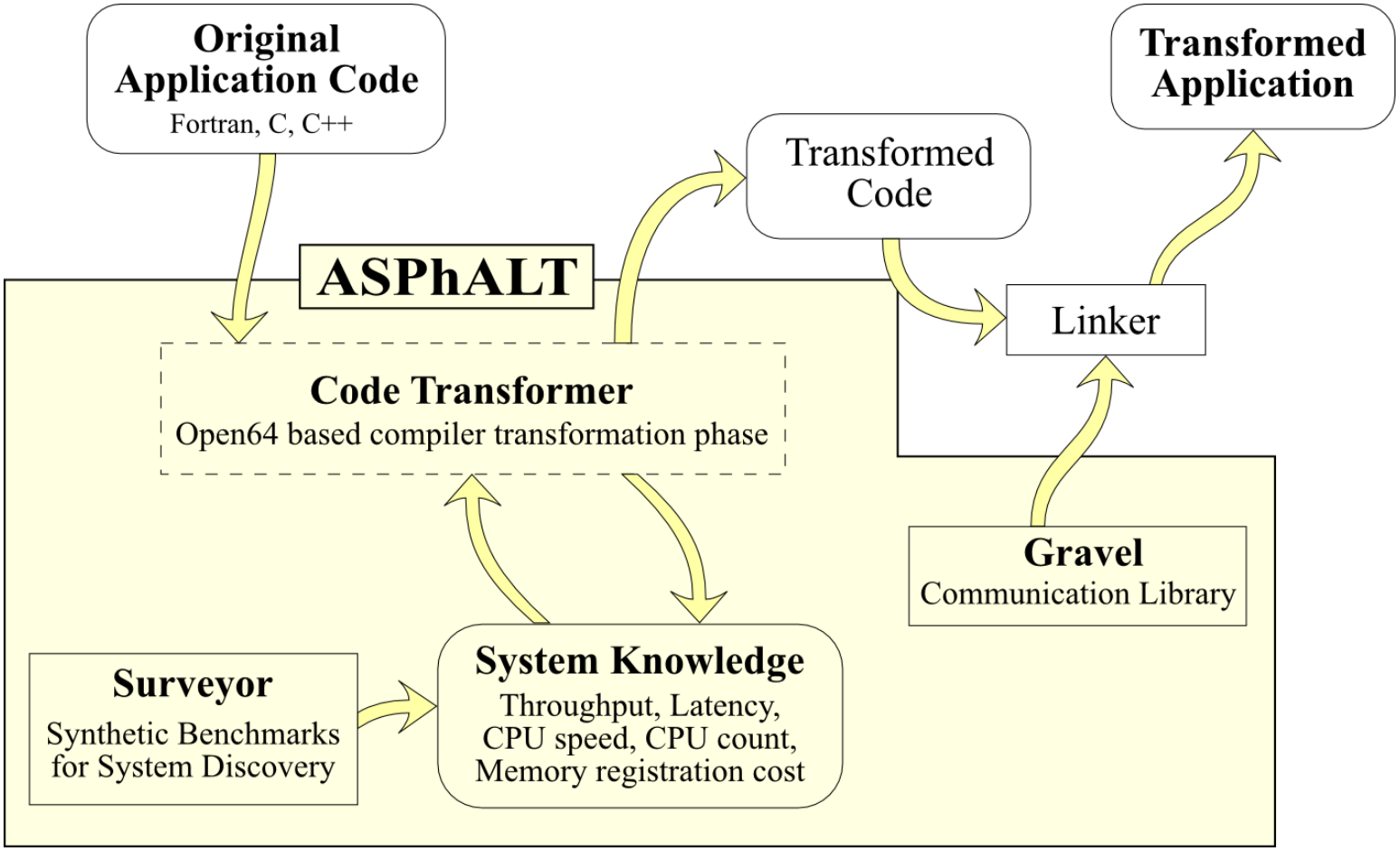
Proposed Solution:

An **automatic** system that **transforms** simple communication code into efficient code.

Side-effect:

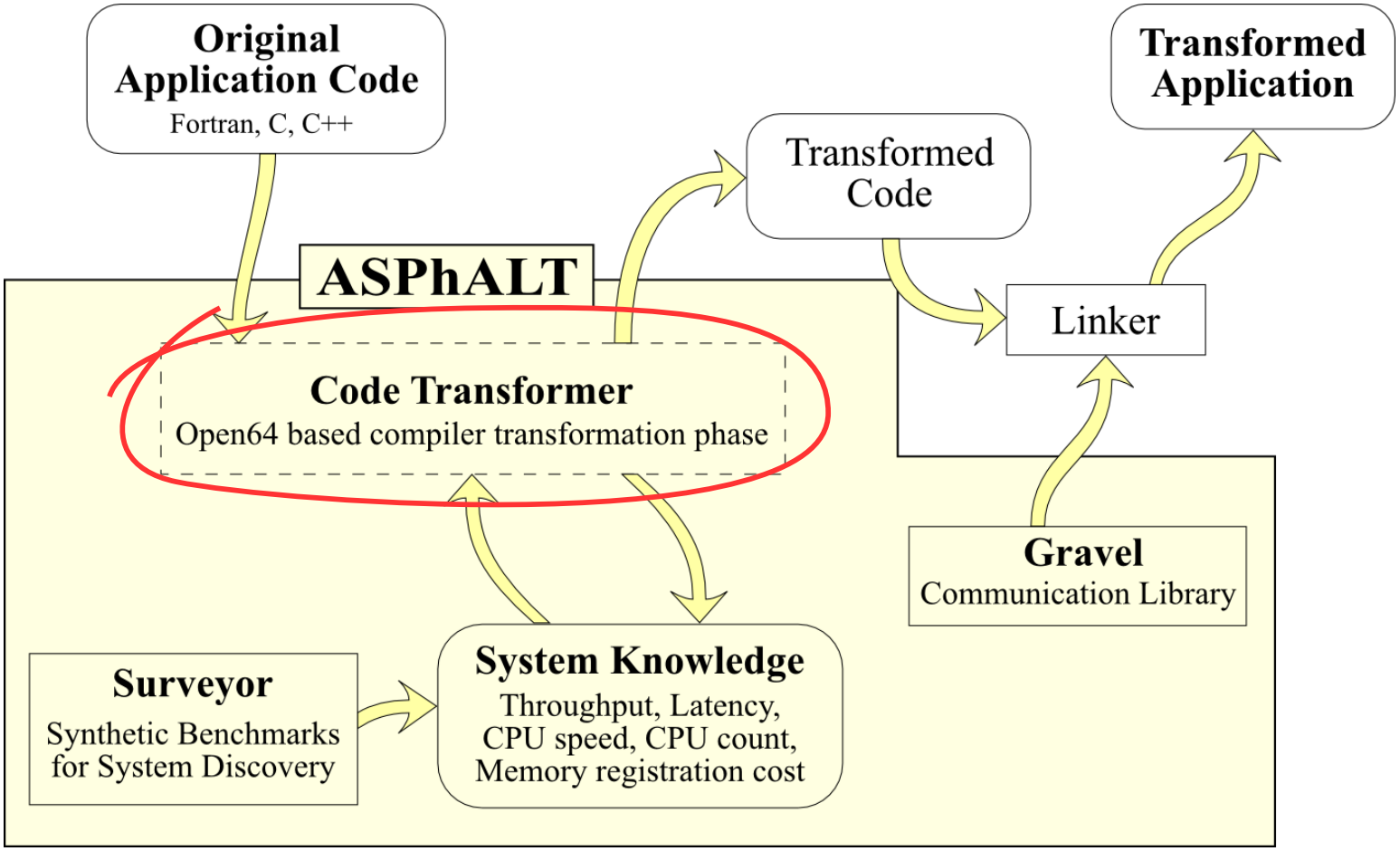
Enables **legacy parallel MPI applications** to perform better, even if written **without any knowledge** of this system

Our Framework : ASPhALT*



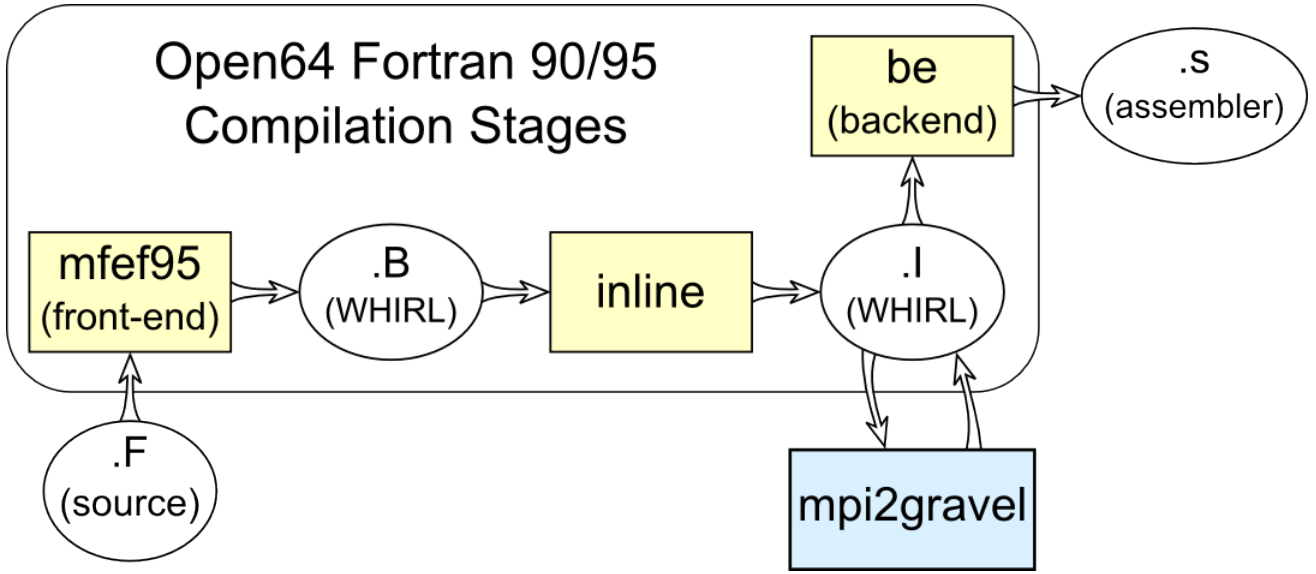
*Automated System for Parallel AppLication Transformation

Our Framework : ASPhALT*



*Automated System for Parallel AppLication Transformation

Integration with Open64



Pros:

- Portability
- Rapid Prototyping

Cons:

Underutilization of existing:

- Program analysis
- Program transformations

Challenges & Opportunities

Gravel enables explicit memory registration & rendezvous handshaking at the application layer

- Memory Registration Location
  - ✔ Smart placement can reduce registration cost
  - ✗ Choosing location requires inter-procedural analysis
- Rendezvous Protocol Choice
  - ✔ Early handshake initiation can overlap the control messages
  - ✔ Advanced rendezvous can reduce control messages & increase overlap
  - ✗ Control flow analysis required to map MPI code to Gravel protocol

Memory Registration:
context awareness matters

Frequent registering & unregistering message buffers induces significant cost

Library Function: Not context aware

mpi_isend() →
```
handshake()
register_msg_buffer()
init_data_transfer()
...
```

mpi_wait() →
```
block_until_completion()
unregister_msg_buffer() ??
...
```

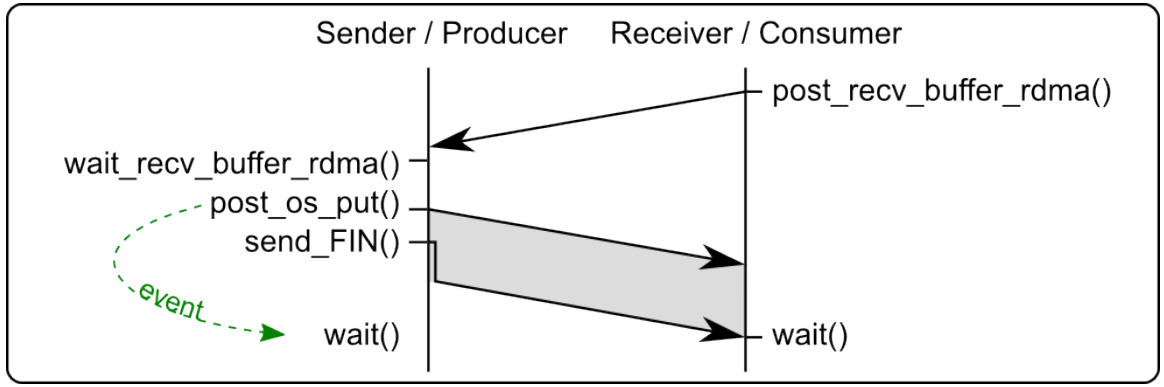
Compiler Tool: Fully context aware

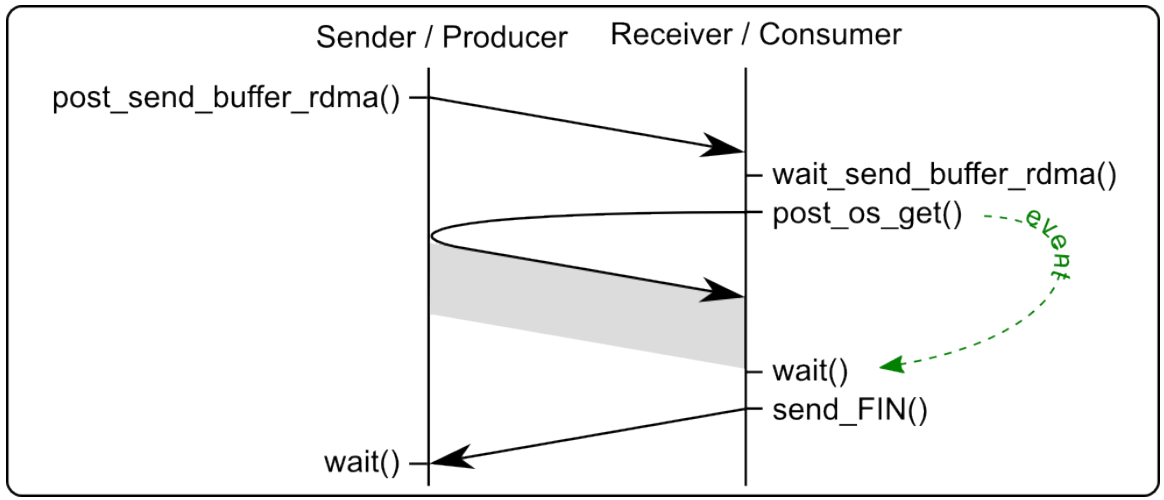
```
foo(){
    register_msg_buffer()
    do i=1, N
        bar()
    enddo
    unregister_msg_buffer()
}

bar(){
    ...
    isend()
    ...
    wait()
    ...
}
```

Gravel rendezvous protocols

Use of wrong rendezvous protocol would lead to deadlock





Mapping MPI to Gravel rendezvous

Trivial:
execution order = textual order

```
mpi_irecv()

do i=1,N
    sbuf[ i ] = ...
enddo

mpi_isend( sbuf )
mpi_waitall()
```

More complex:
execution order = reverse textual order

```
do i=1,N
    if( i > 1 ) then
        mpi_irecv()
        ...
        mpi_waitall()
    endif

    do j=1,N
        sbuf[ i ][ j ] = ...
    enddo

    if( i < N ) then
        mpi_isend( sbuf )
    endif

enddo
```

Example: NAS SP {x,y,z}_solve.f

Further Optimizations (future work)

Before

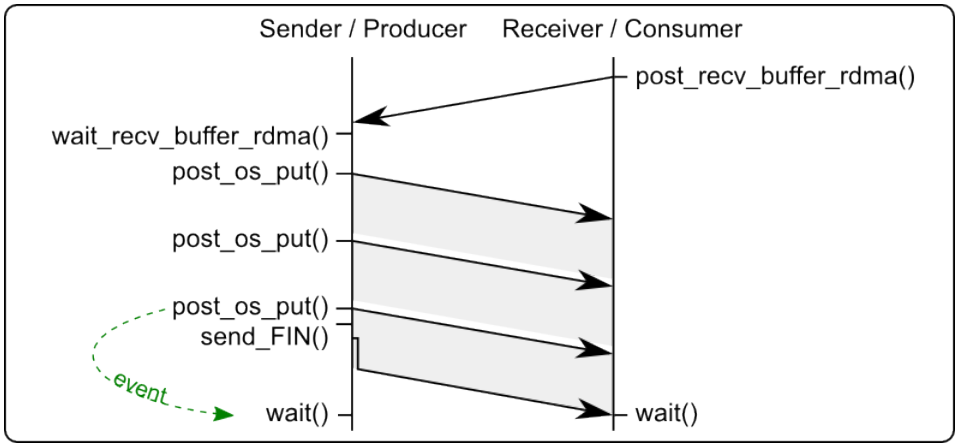
```
mpi_irecv()

do i=1,N
   sBuf[ i ] = ...
enddo

mpi_isend()
mpi_waitall()
```

After

```
gravel_post_recv_buffer_rdma()
do T=1, N, K
   do i=T, min(T+K-1, N)
      sBuf[ i ] = ...
   enddo
   if( T == 1 ) then
      gravel_wait_recv_buffer_rdma()
   endif
   gravel_post_os_put()
enddo
gravel_send_fin(next, ierr)
gravel_waitall()
```

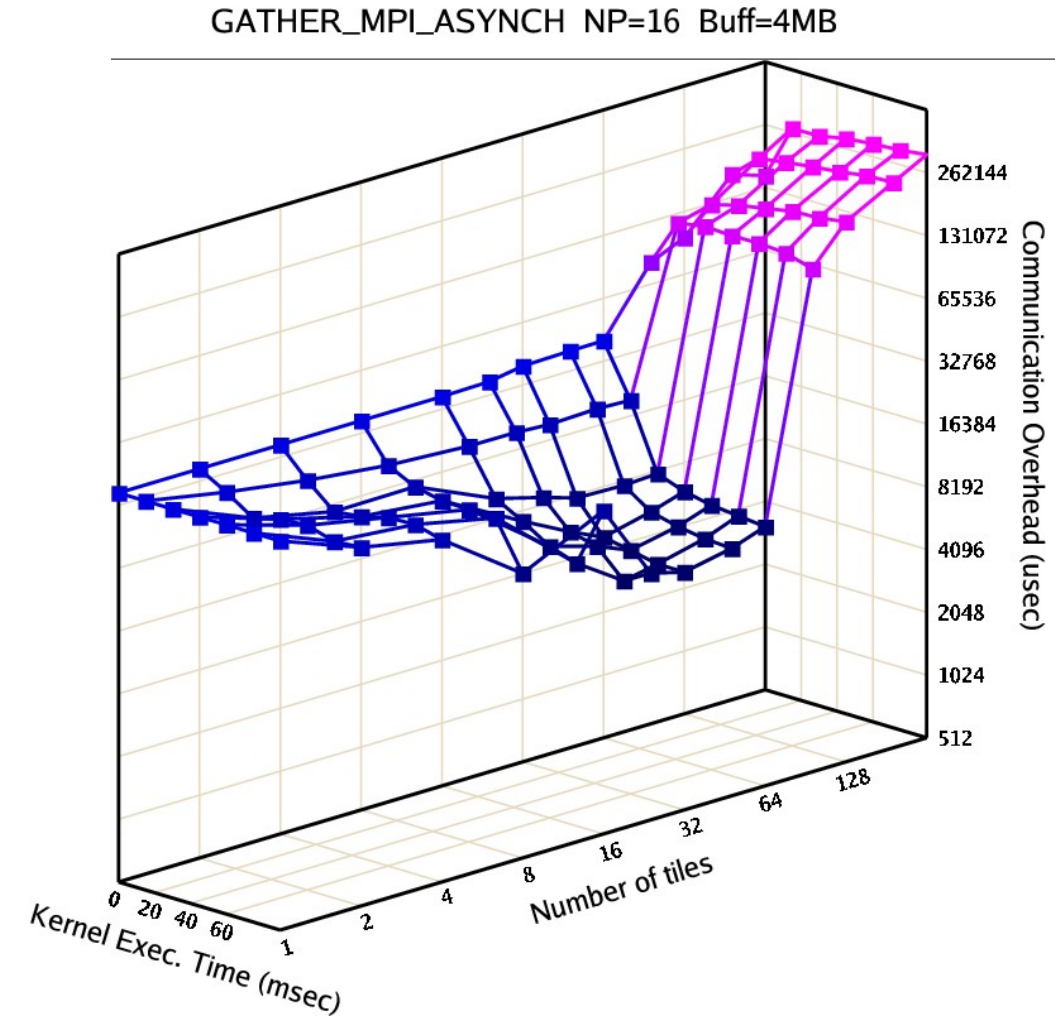


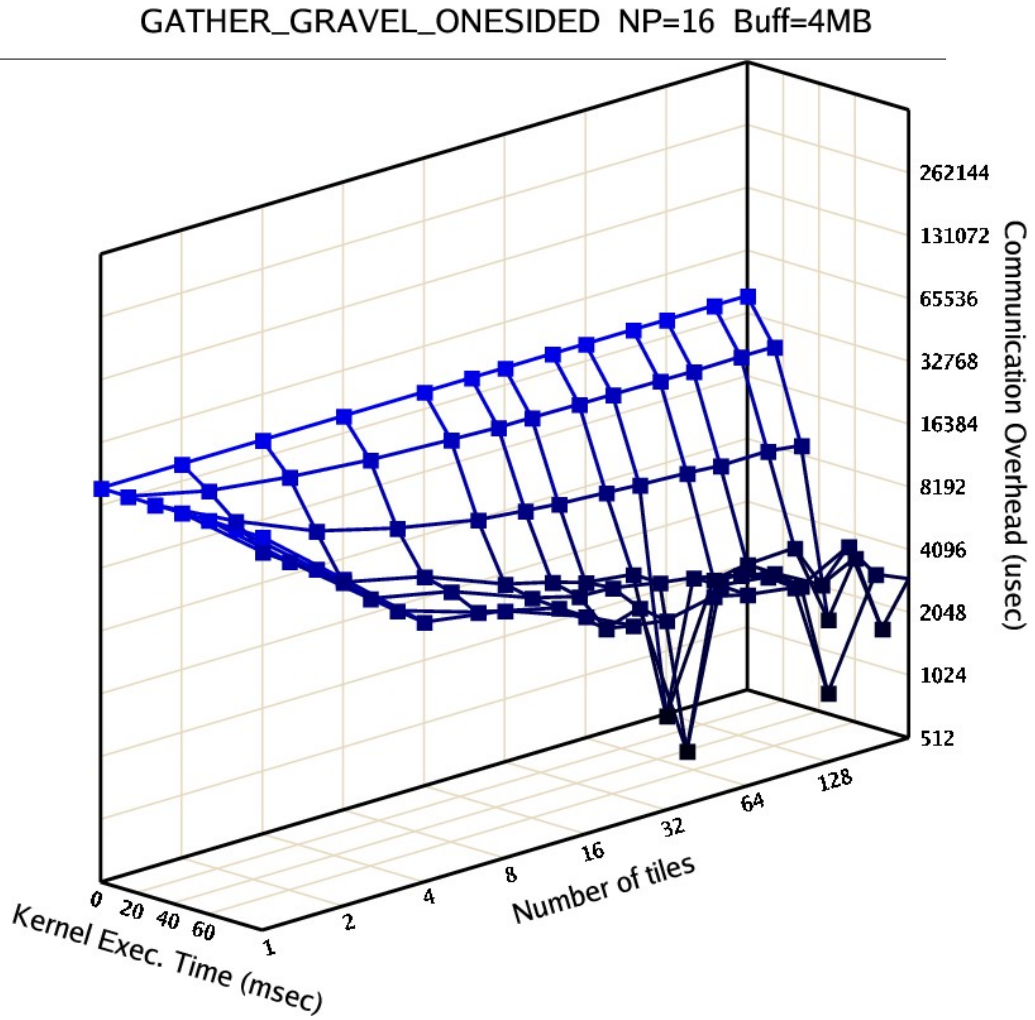
Current & Future Directions

- Break MPI collectives into loops of send()/recv()
- Use **L**oop **N**est **O**ptimizations to increase overlap
- Integrate code into Open64 backend (**be)**

Questions ?

Gravel: Performance graphs (1)





Gravel: Performance graphs (2)