Dynamic Analysis to Support Program Development with the Textually Aligned Property for OpenSHMEM Collectives

OpenSHMEM Workshop 2015, Annapolis, 2015-08-05

Andreas Knüpfer (andreas.knuepfer@tu-dresden.de)

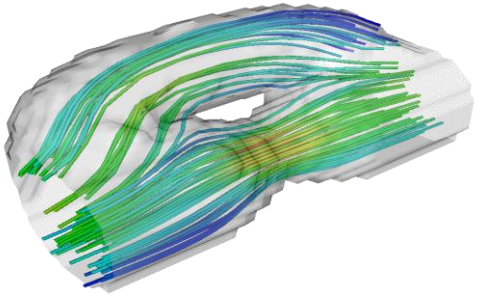
Tobias Hilbrich (tobias.hilbrich@tu-dresden.de)

Joachim Protze (protze@itc.rwth-aachen.de)

Joseph Schuchart (joseph.schuchart@tu-dresden.de)

Technische Universität Dresden, Germany

Motivation – A Medical Application (1)



Calculation of blood flow with 3D Lattice-Boltzmann method

- MPI parallelization
- Could be done with OpenSHMEM as well
- Around 6500 lines of code
- Different input files describe geometry of artery: tube, tube-stenosis, bifurcation

Motivation – A Medical Application (2)

- Tube-stenosis geometry: just a tube with varying radius
- Running the application in parallel:

```
mpirun -np 3 B_Stream 500. tube-stenosis
```

⇒ **Application hangs**

Motivation – A Medical Application (3)

What's happening:

```
PE 0
timestamp= 9321: MPI_Reduce(comm = MPI_COMM_WORLD)
PE 1
timestamp= 9325: MPI_Bcast(*buffer, count = 3, datatype = MPI_DOUBLE,
   root = 0, comm = MPI_COMM_WORLD)
PE 2
timestamp= 9327: MPI_Bcast(*buffer, count = 3, datatype = MPI_DOUBLE,
   root = 0, comm = MPI_COMM_WORLD)
```

⇒ **Collective mismatch (PE 0: MPI_Reduce vs other PEs: MPI_Bcast)**

Motivation – A Medical Application (4)

```
main {
    …
    num_iter = calculate_number_of_iterations();
    for (i=0; i < num_iter; i++) {
        computeBloodflow();
    }
    writeResults();
    ….
}
```

```
if (radius < x) num_iter = 50;
if (radius >= x) num_iter = 200;
// ERROR: it is not ensured here that all
// procs do the same (maximal) number
// of iterations
```

```
CalculateSomething();
// exchange results with neighbors
MPI_Reduce(…);
```

```
// communicate results with
neighbours MPI_Bcast(…);
```

Introduction – SPMD (1)

- SPMD is the dominating parallel programming style in HPC
- even MPMD codes really use few sets of SPMD codes
- Pure functional decomposition is lacking enough distinct functions to scale
- Example:

```
int main (…) {
    shmem_init();
    me = shmem_my_pe ();
    npes = shmem_n_pes ();
    …
    shmem_barrier(0,0,npes,...) ;
    …
}
```

(0,0,npes)

=> Stride expression identifies participating PEs, here all PEs

Ideal SPMD (textually aligned):

If you see a collective in code, then all PEs enter it

=> Easy to read, maintain, refactor, …

Introduction – SPMD (2)

- HPC codes frequently violate the SPMD idea … intentionally so!

```
if ( 0 == shmem_my_pe() ) {
    read_input_files( … );
    do_initialization( … );
    print_status_output();
}
```

```
if ( ! is_border_element() ) {
    shmem_double_put( …,
        neighnor_pe );
    …
}
```

Introduction – Textual Alignment

- Definition **textually aligned:**
  - Strict SPMD
  - Every PE executes same operations in same routine (function) from same call stack

```
int main ( void ) {
  ...
  shmem_barrier(0,0,npes,...) ;
  if ( me % 2 == 0) {
    shmem_barrier(0,0,npes,...) ;
  } else {
    shmem_barrier(0,0,npes,...);
  }
```

Textually Aligned

Violation to textually Aligned (Called from two distinct callstacks, differing in their line numbers)

Contributions – Textual Alignment Levels for OpenSHMEM

- Here, limit the scope of textual aligned programing to collective operations
  - they have a well-defined notion of "simultaneous"
- Classification of textual aligned strategies:
  - **Full textual alignment:** only global collective calls (all PEs) allowed, need to be from identical call stacks
  - **Subset tolerant alignment:** non-global collective calls allowed, full textual alignment required within the group of matching calls ("active set")
  - **Weak textual alignment:** source lines may differ, but must be from the same routine (also the call stacks may be different, e.g., recursion levels)
  - **Weak subset tolerant alignment:** Both generalizations apply

Contributions – Textual Alignment Levels for OpenSHMEM, Examples (1)

- Examples for the alignment classes:
  - All examples are valid OpenSHMEM codes sniplets

Full textual alignment:

```
int main (...) {
  ...
  shmem_barrier_all();
  ...
}
```

Weak textual alignment:

```
int main (...) {
  ...
  if ( 0 == me%2 ) // odd/even
    shmem_barrier_all();
  else
    shmem_barrier_all();
  ...
}
```

Contributions – Textual Alignment Levels for OpenSHMEM, Examples (2)

Subset tolerant alignment:

```
int main (...) {
  ...
  shmem_barrier( me % 2, 1,
                 (npes + (me+1)%2) / 2,
                 ...);
  ...
}
```

Example: 5 PEs

- Group A: PEs 0, 2, 4
- Group B: PEs 1, 3

Weak subset tolerant alignment:

```
...
// root has special handling
if ( 0 == me )
  shmem_broadcast32( 0, 1,
                     (npes + (me+1)%2) / 2, …);
else
  shmem_broadcast32( me % 2, 1,
                     (npes + (me+1)%2) / 2, ...);
...
```

Contributions – Textual Alignment Levels for OpenSHMEM, Examples (3)

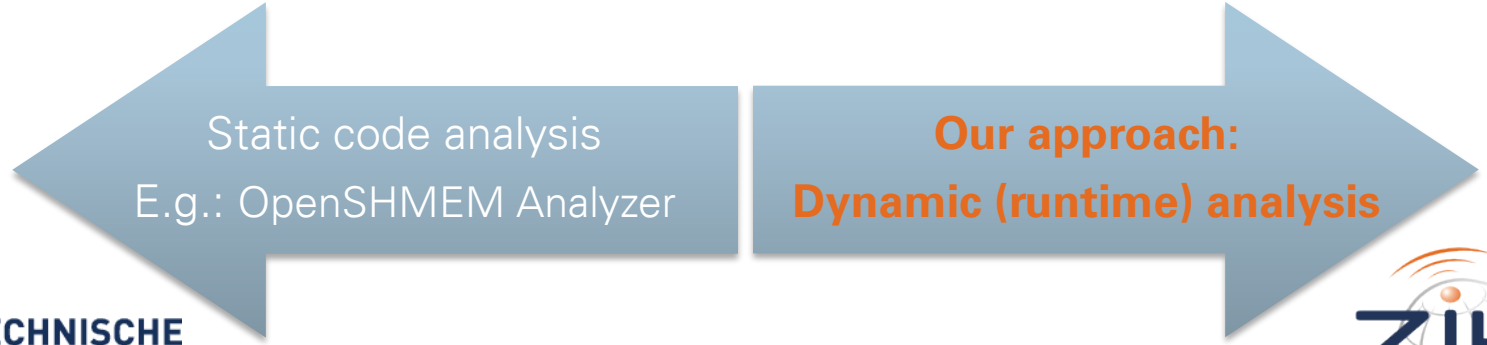
Violation of weak textual alingment:

```
int main (...) {
  ...
  // odd/even
  if ( 0 == me % 2 )
    foo ();
  else
    bar ();
  ...
}
```

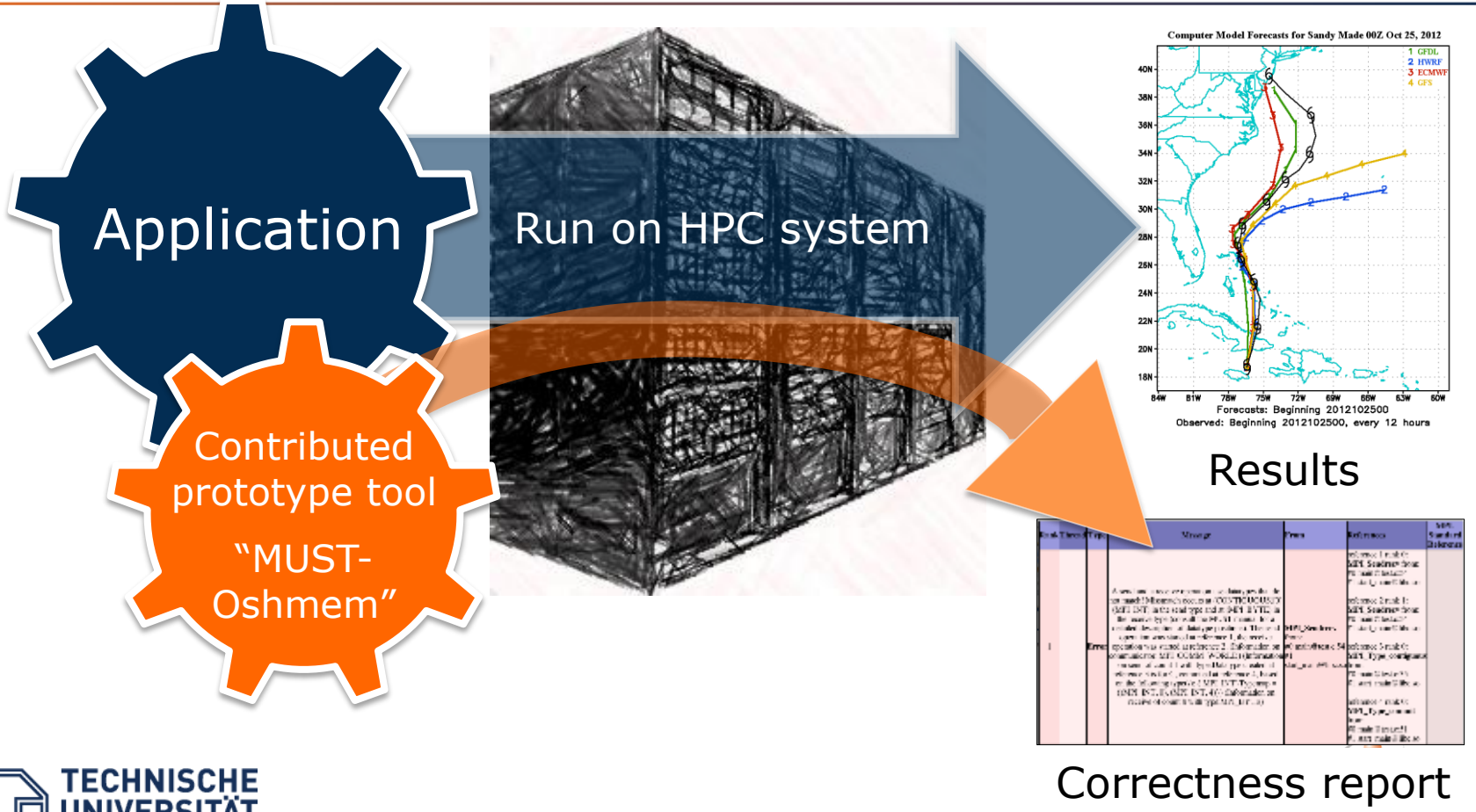
```
void foo () {
  shmem_barrier_all();
}

void bar () {
  shmem_barrier_all();
}
```

Challenge

- OpenSHMEM applications with high level of textual alignment
  - ⇒ Easier to maintain, read, extend, …
- How to find out:
  - What overall alignment level is?
  - Where are (possibly unintended) violations?
  - Are there other correctness issues, e.g., collective mismatch?



Approach


Application
Contributed prototype tool
"MUST-Oshmem"
Run on HPC system
Computer Model Forecasts for Sandy Made 00Z Oct 25, 2012
1 GFDL
2 HWRF
3 ECMWF
4 GFS
Forecasts: Beginning 2012102500
Observed: Beginning 2012102500, every 12 hours
Results
Correctness report

Approach – Example

```
if ( 0 == me%2 ) // odd/even
   shmem_broadcast32 (…, 0, 0, npes,…);
 else
   shmem_broadcast32 (…, 0, 0, npes,…);
```

MUST Output, starting date: Tue Jun 30 09:24:45 2015.

Rank(s)	Type	Message
1, 3	**Warning**	An OpenSHMEM collective uses a weak textual alignment. This happens if the cal…

Details:

Message	From	References
An OpenSHMEM collective uses a weak textual alignment. This happens if the call stacks that issue matching collective operations differ, but still result from the same program function. The OpenSHMEM standard allows this behavior, but it may highlight a defect if you did not intend this behavior. The PE group of the collective is (start=0, logStride=0, count=4). The first collective (A) and its call stack is detailed in the source location for this error message. The second collective (B) and its call stack is detailed as reference 1 (rightmost column).	Representative location: **shmem_broadcast32** (1st occurrence) called from: #0 main@coll-weak-aligned-warning.c:80	References of a representative process: reference 1 rank 2: **shmem_broadcast32** (1st occurrence) called from: #0 main@coll-weak-aligned-warning.c:87

Approach – Prototype Functionality

Basic OpenSHMEM Correctness

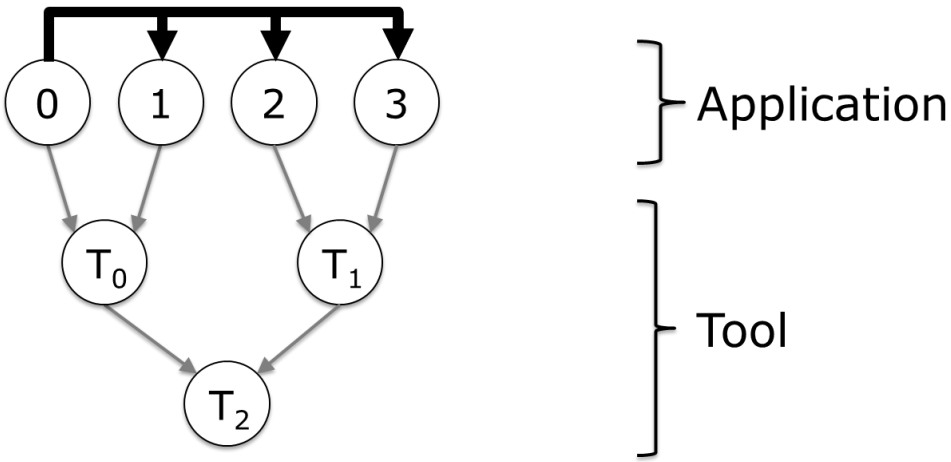
- Collective type mismatch (motivation example)
- Root PE mismatch
- „nreduces“ mismatch
- „nelems“ mismatch

Textual Alignment

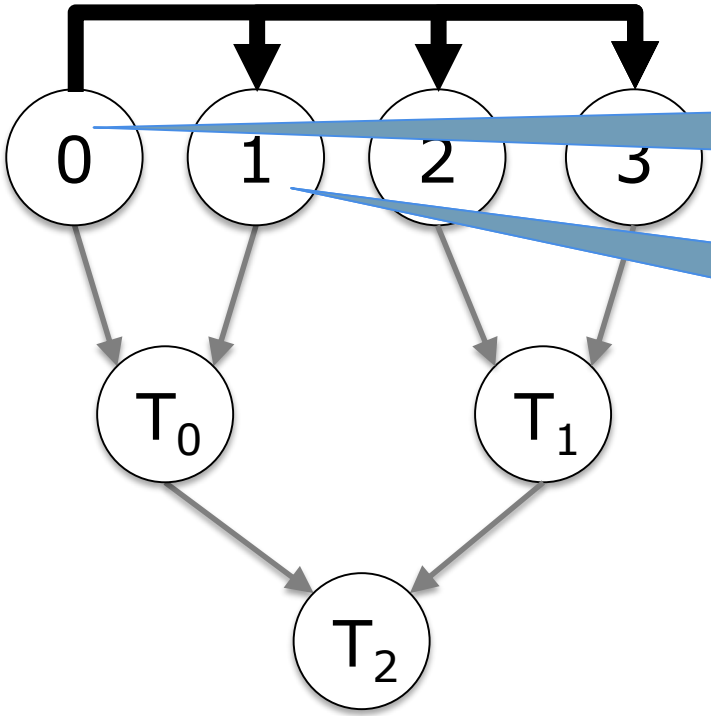
- Overall alignment level
- Instances of weak [subset] alignments or unaligned [subset] collectives

Approach – Tool Architecture

- Goal: Scalability
  - Introduce additional tool PEs
  - Arrange all PEs in a tree: Tree Based Overlay Network (TBON)
  - Events about collective calls are passed to parent nodes
  - Every tool node joins event objects for matching collective operations



Approach – Scalable Correctness Analysis

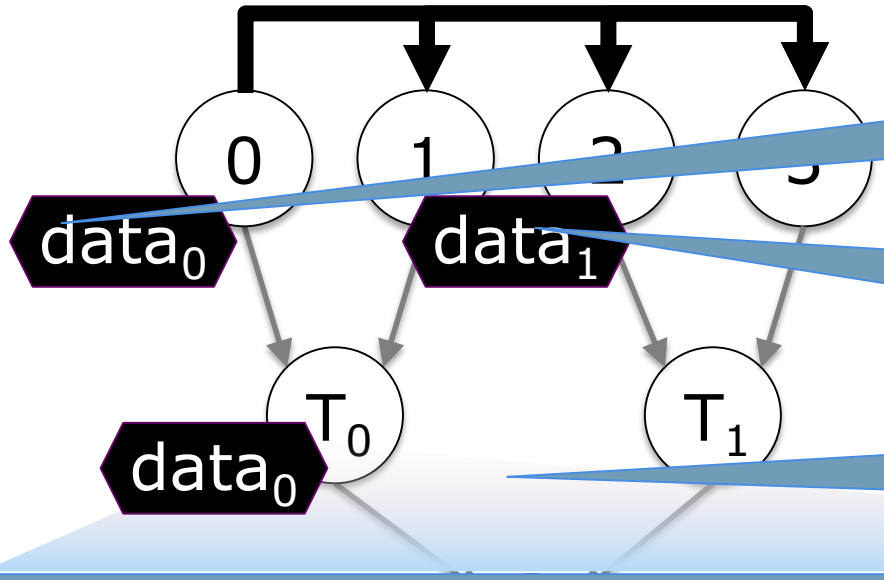


Colective:
shmem_broadcast32 (<$data_0$>)

Colective:
shmem_broadcast32 (<$data_1$>)

Tool

Approach – Scalable Correctness Analysis



Event for:
shmem_broadcast32 (<$data_0$>)

Event for:
shmem_broadcast32 (<$data_1$>)

Single event for:
$data_0$ and $data_1$

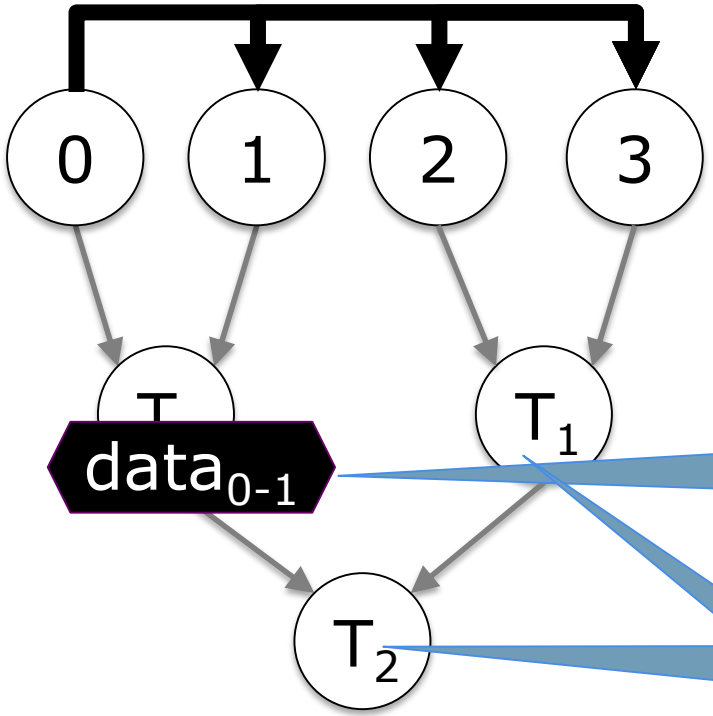
1) Input events:

Participant events available to T_0

2) Run checks

3) Output Event = representative event: $data_{0\text{-}1}$

Approach – Scalable Correctness Analysis

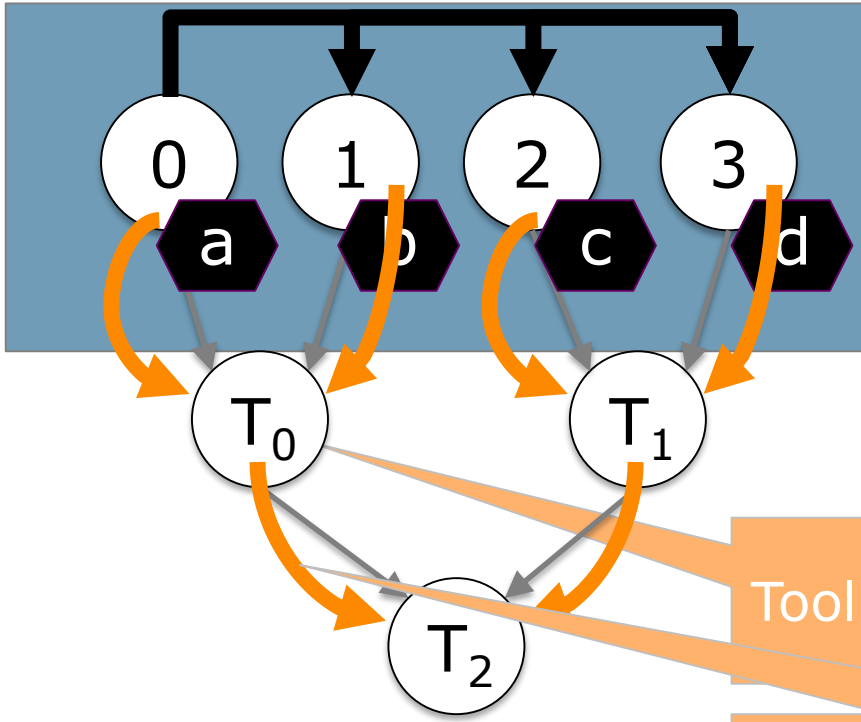


Single event for:

$data_0$ and $data_1$

Recursive repetition on all layers

Approach – Scalable Correctness Analysis



Collective:
shmem_broadcast32

Communication load:
$O(p \bullet k)$

p … #PEs

k … payload

Tool analysis cost can be $O(1)$

Communication cost $O(1)$

Proposal Beyond Textual Alignment Checks

```
if ( 0 == me%2 ) // odd/even
   shmem_broadcast32 (…, 0, 0, npes,…);
 else
   shmem_broadcast32 (…, 0, 0, npes,…);
```

MUST Output, starting date: Tue Jun 30 09:24:45 2015.

Rank(s)	Type	Message
1, 3	**Warning**	An OpenSHMEM collective uses a weak textual alignment. This happens if the cal…

Details:

Message	From	References
An OpenSHMEM collective uses a weak textual alignment. This happens if the call stacks that issue matching collective operations differ, but still result from the same program function. The OpenSHMEM standard allows this behavior, but it may highlight a defect if you did not intend this behavior. The PE group of the collective is (start=0, logStride=0, count=4). The first collective (A) and its call stack is detailed in the source location for this error message. The second collective (B) and its call stack is detailed as reference 1 (rightmost column).	Representative location: **shmem_broadcast32** (1st occurrence) called from: #0 main@coll-weak-aligned-warning.c:80	References of a representative process: reference 1 rank 2: **shmem_broadcast32** (1st occurrence) called from: #0 main@coll-weak-aligned-warning.c:87

Highlight intended cases of reduced alignment in the code

⇒ Documents them

⇒ Filter tool output

Proposal Beyond Textual Alignment Checks

```
if ( 0 == me%2 ) { // odd/even
    annonshmem_weak_coll ();
    shmem_broadcast32 (…, 0, 0, npes,…);
} else {
    annonshmem_weak_coll ();
    shmem_broadcast32 (…, 0, 0, npes,…);
}
```

MUST Outputfile

, starting date: Wed Feb 5 09:16:34 2014.

Rank(s)	Type	Message	From	References
	Information	MUST detected no MPI usage errors nor any suspicious behavior during this application run.		

MUST has completed successfully, end date: Wed Feb 5 09:16:34 2014.

TECHNISCHE UNIVERSITÄT DRESDEN

Conclusions

- Parallel programming is hard
- OpenSHMEM applications with textually aligned collectives simplify access to code
- Contributions:
  - Definitions of textual alignment levels for OpenSHMEM
  - Prototype tool to detect violations to these levels
    - ⇒Based on Dyninst and MUST
      https://doc.itc.rwth-aachen.de/display/CCP/Project+MUST
  - Scalable implementation of checks
  - Basic correctness checks for OpenSHMEM collectives

Future Work

- Deeper evaluation (beyond small examples)
- User feedback
- What can we do with one-sided communication?
- Development beyond research prototype
- Deadlock detection

Thank you!

Questions?