

OpenSHMEM Tools Eco-system

1990s

- SHMEM library introduced by Cray Research Inc. (T3D systems)
- Adapted by SGI for products based on the Numa-Link architecture and included in the Message Passing Toolkit (MPT).
- Vendor specific SHMEM libraries emerge (Quadrics, HP, IBM, gpSHMEM, SiCortex etc.).

- Extreme Scale Systems Center (funded by DOD), located at Oak Ridge National Laboratory, along with the University of Houston come together to address the differences between various SHMEM implementations.
- **OpenSHMEM** is born.
- OpenSHMEM Specification 1.0 finalized.

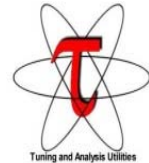
2012

- Development of **OpenSHMEM Reference Library, Validation and Verification Suite**.
- OpenSHMEM Specification 1.0d released to community.
- Specification supported by some Vendors.
- Compiler based (**OpenSHMEM Analyzer**), debuggers (**DDT**) and performance tools (**TAU**, **SHMEM Tracer**) support OpenSHMEM.

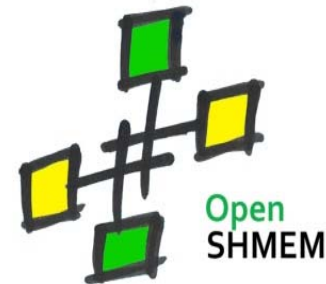
2013

- Development of the OpenSHMEM programming environment gains momentum.
- Tracing Tools support OpenSHMEM (**Vampir / Vampir Trace**)
- UCCS lower level communication library integration into OpenSHMEM Reference implementation.
- OpenSHMEM Specification 1.0e released to community.
- **OpenSHMEM Specification 1.1 will be released in March 2013**

OpenSHMEM Programming Environment



OpenSHMEM
Reference
Library



Vampir



- **OpenSHMEM** is a community driven effort.
- Website: www.oprnshmem.org

Managed by UT-Battelle for the
U. S. Department of Energy



Introduction

- **State-of-the-art tools for OpenSHMEM**
- **Tools are part of the OpenSHMEM programming environment**
- **We will present tools for:**
 - **Program Development**
 - **Error Checking**
 - **Performance Analysis**
 - **Performance Modeling**
 - **Debugging**

Introduction (cont)

We will cover the following tools:

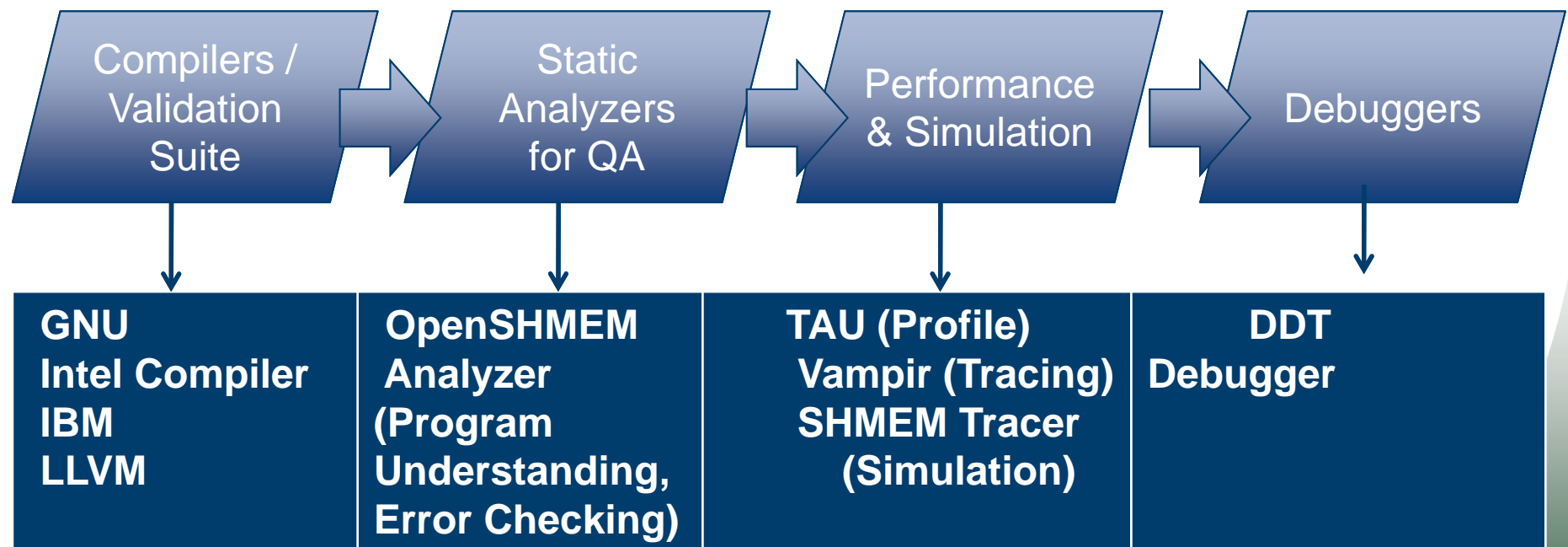
- **TAU performance system**
- Vampir / Score-P
- DDT debugger
- OpenSHMEM Analyzer

Why use OpenSHMEM tools?

- **Help understanding your application**
- **Fast prototyping to OpenSHMEM**
- **Increases productivity**
- **Improves performance and quality of software**
- **Facilitates the maintenance of the code**
- **Use leading technologies in the field**

Software Development Cycle

- **Tools are part a the programming environment for OpenSHMEM**



Static Analyzers: OpenSHMEM Analyzer

- **Understanding of an OpenSHMEM application**
 - Callgraph, Control flow graphs, Def-Use info
- **Helps to produce high quality code:**
 - Prevents common errors in OpenSHMEM
 - Type checks, out-of-bounds, pointer and alias analysis
 - Provides information for optimization
- **Provides Feedback to the user**
 - Command based, User Based
- **We will cover this tool in the lab session**

The OpenSHMEM Analyzer

- **Analysis that the tool can provide:**
 - SHMEM-aware callgraphs, callsites
 - Visualization of synchronizations, atomics, memory I/O,
 - Mappings to the source code, source code browser.
- **Errors and Semantic Checks:**
 - Initialization / finalization calls are present in the code
 - PEs are initialized
 - Redundant PE initializations
 - Correct type information of OpenSHMEM calls
 - Data types and sizes check in put/get operations
 - Enforcing accesses to symmetric data structures
 - Data/Memory Management, advanced analysis
 - Buffer allocations / de-allocations
 - Check the memory allocation library used.

Implementation

- **Based on a state-of-the-art compiler, that work on large applications.**
 - Open64 AMD 4.2.5.2 compiler release.
 - Handles C/C++/Fortran + OpenSHMEM 1.0a Spec
- **Relies heavily on Inter-procedural analysis.**
- **OpenSHMEM library is recognized by compiler and its semantics is used to perform checks**
 - **Analysis is exported to different formants:**
 - Command line messages
 - Hyper-texted graphs/images that can be visualized with a web browser.
- **Can be easily integrated to application Makefiles.**

Evaluation of the OpenSHMEM

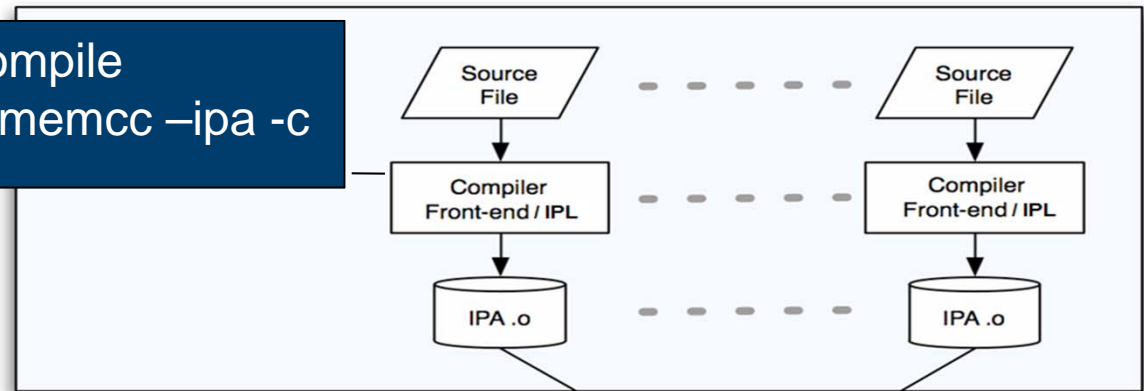
- **We validated the tool with the OpenSHMEM validation suite 1.0 provided by openshmem.org**
- **This includes the validation tests that comes with 1.0a**
- **We were able to reproduce the bugs intended for the validation suite.**
- **In one case we found an unintended bug in one of the cases.**

OpenSHMEM Analyzer Implementation

Compilers:

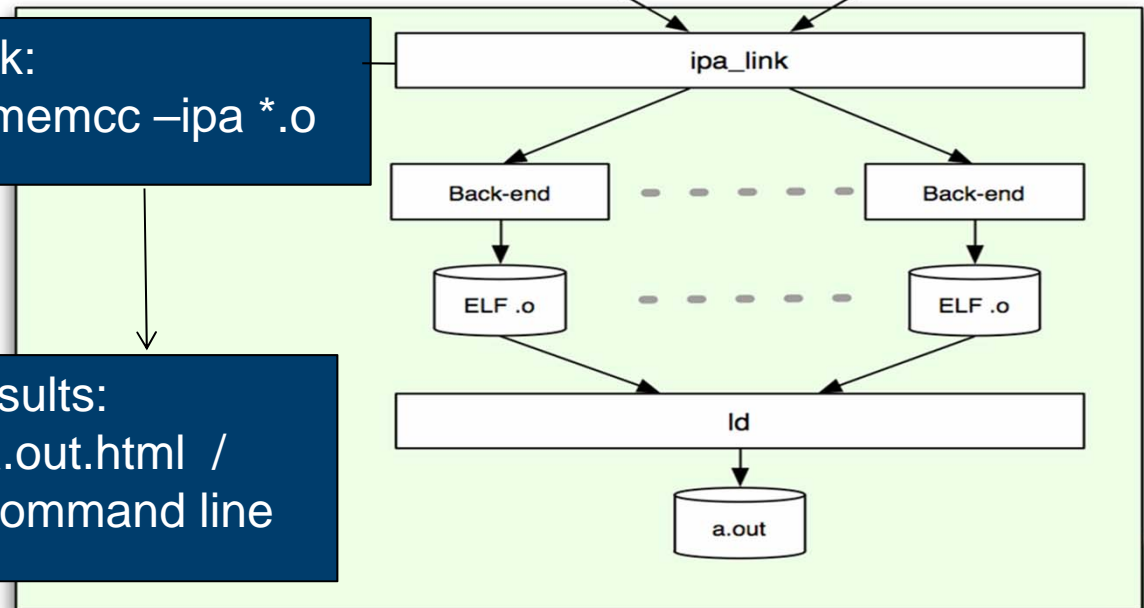
- Summarization & local analysis

Compile
`shmemcc -ipa -c`

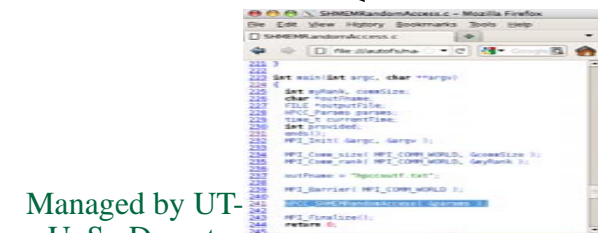
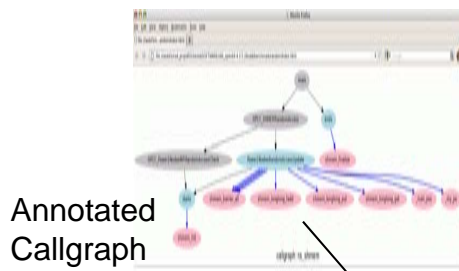


- Analysis
- Detection / Optimization
- Visualization of results:

Link:
`shmemcc -ipa *.o`

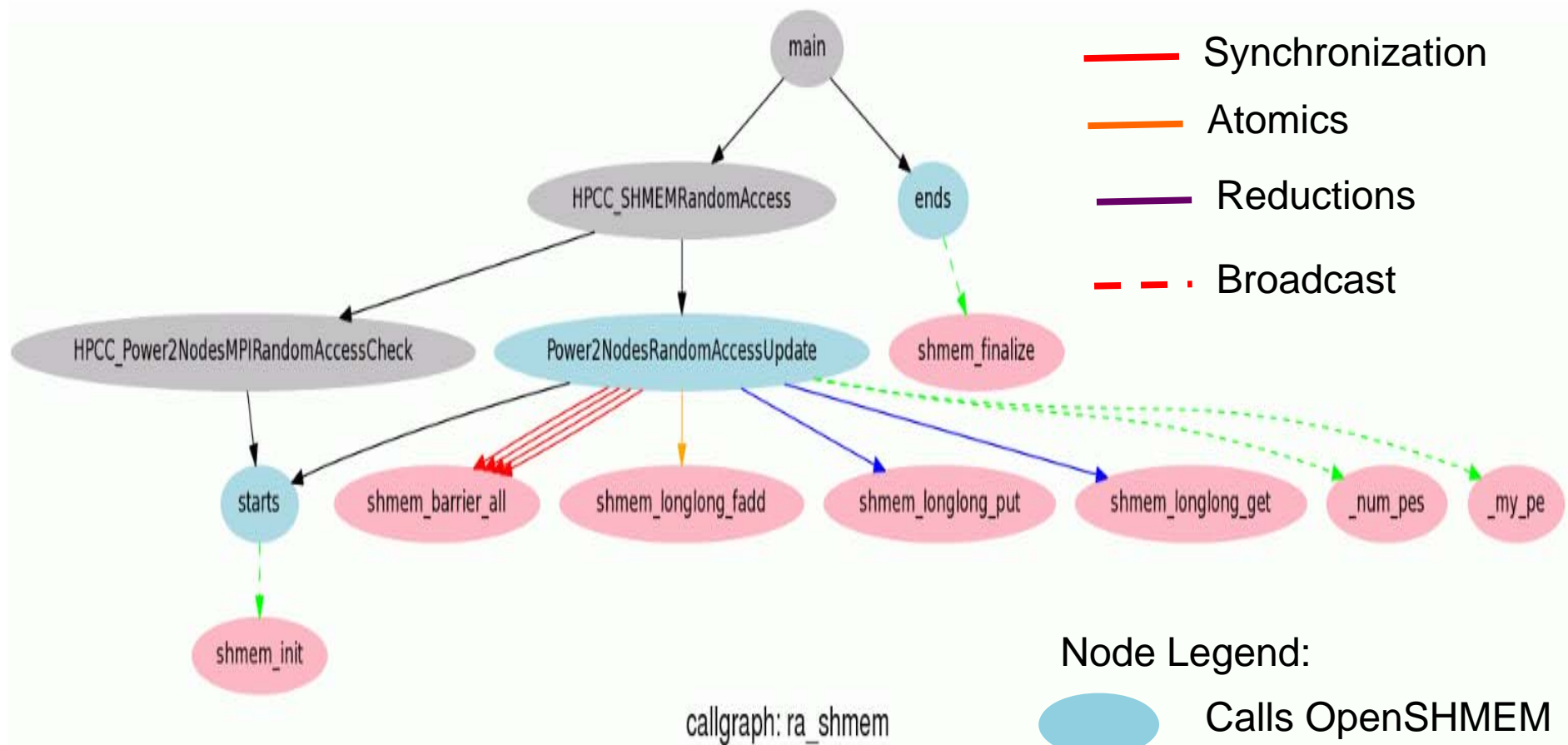


Results:
`a.out.html /`
`command line`



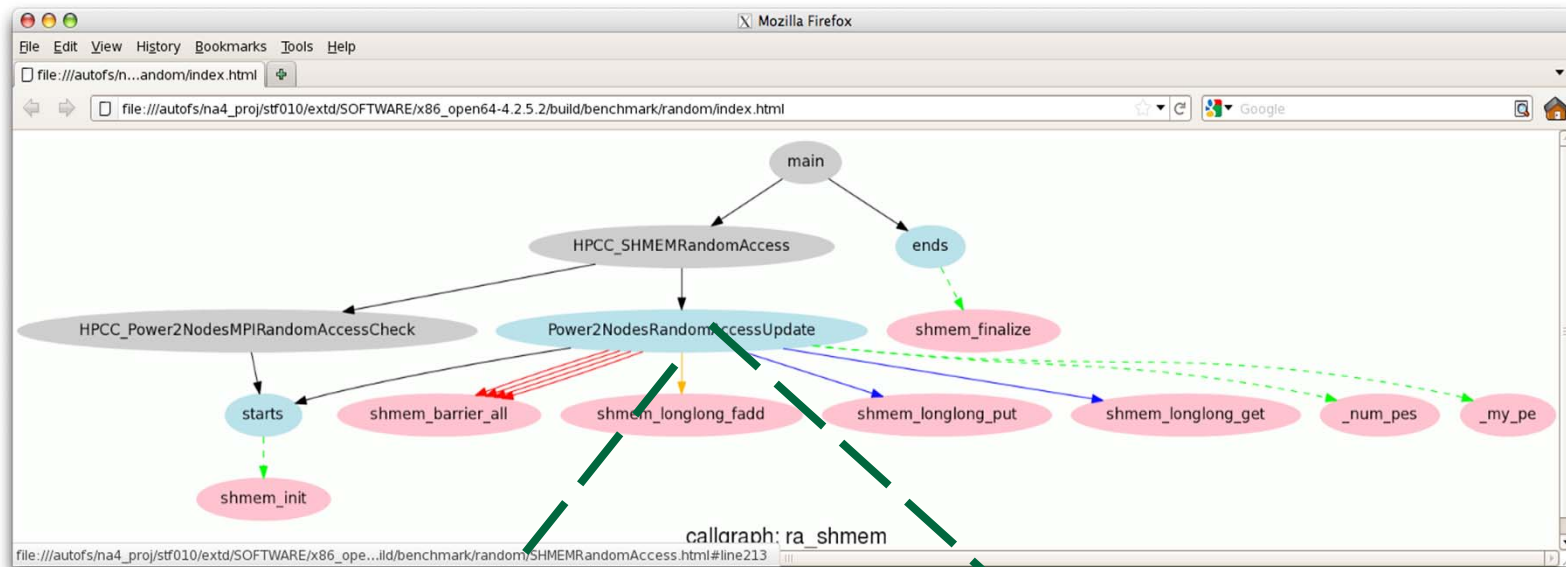
OpenSHMEM analysis

- **Callgraph generated:**



OpenSHMEM analysis

- Hypertexted OpenSHMEM callgraph



```
121 remote_proc = (rank >> log2(numNodes) & numNodes - 1);
122 shmem_longlong_get(&remote_count, &count[thisPe], remote_proc);
123 shmem_longlong_put(&updates[thisPeId][remote_count], &rank, 1, remote_proc);
124 shmem_longlong_fadd(&count[thisPeId], 1, remote_proc);
125
126 shmem_barrier_all();
127
128 for(i = 0; i < numNodes; i++){
129     count2 = 0;
130     while (count[i]){
131         datum = updates[i][count2];
132         index = datum & nlocal1;
133         HPCC_Table[index] ^= datum;
134         updates[i][count2] = 0;
135         count2++;
136         count[i]--;
137     }
138     shmem_barrier_all();
139 }
140 shmem_barrier_all();
141
142 shmem_barrier_all();
143
144 shmem_barrier_all();
145
146 shmem_barrier_all();
147
148 shmem_barrier_all();
149
150 shmem_barrier_all();
151
152 shmem_barrier_all();
153
154 shmem_barrier_all();
155
156 shmem_barrier_all();
157
158 shmem_barrier_all();
159
160 shmem_barrier_all();
161
162 shmem_barrier_all();
163
164 shmem_barrier_all();
165
166 shmem_barrier_all();
167
168 shmem_barrier_all();
169
170 shmem_barrier_all();
171
172 shmem_barrier_all();
173
174 shmem_barrier_all();
175
176 shmem_barrier_all();
177
178 shmem_barrier_all();
179
180 shmem_barrier_all();
181
182 shmem_barrier_all();
183
184 shmem_barrier_all();
185
186 shmem_barrier_all();
187
188 shmem_barrier_all();
189
190 shmem_barrier_all();
191
192 shmem_barrier_all();
193
194 shmem_barrier_all();
195
196 shmem_barrier_all();
197
198 shmem_barrier_all();
199
200 shmem_barrier_all();
201
202 shmem_barrier_all();
203
204 shmem_barrier_all();
205
206 shmem_barrier_all();
207
208 shmem_barrier_all();
209
210 shmem_barrier_all();
211
212 shmem_barrier_all();
213
214 shmem_barrier_all();
215
216 shmem_barrier_all();
217
218 shmem_barrier_all();
219
220 shmem_barrier_all();
221
222 shmem_barrier_all();
223
224 shmem_barrier_all();
225
226 shmem_barrier_all();
227
228 shmem_barrier_all();
229
230 shmem_barrier_all();
231
232 shmem_barrier_all();
233
234 shmem_barrier_all();
235
236 shmem_barrier_all();
237
238 shmem_barrier_all();
239
240 shmem_barrier_all();
241
242 shmem_barrier_all();
243
244 shmem_barrier_all();
245
246 shmem_barrier_all();
247
248 shmem_barrier_all();
249
250 shmem_barrier_all();
251
252 shmem_barrier_all();
253
254 shmem_barrier_all();
255
256 shmem_barrier_all();
257
258 shmem_barrier_all();
259
260 shmem_barrier_all();
261
262 shmem_barrier_all();
263
264 shmem_barrier_all();
265
266 shmem_barrier_all();
267
268 shmem_barrier_all();
269
270 shmem_barrier_all();
271
272 shmem_barrier_all();
273
274 shmem_barrier_all();
275
276 shmem_barrier_all();
277
278 shmem_barrier_all();
279
280 shmem_barrier_all();
281
282 shmem_barrier_all();
283
284 shmem_barrier_all();
285
286 shmem_barrier_all();
287
288 shmem_barrier_all();
289
290 shmem_barrier_all();
291
292 shmem_barrier_all();
293
294 shmem_barrier_all();
295
296 shmem_barrier_all();
297
298 shmem_barrier_all();
299
300 shmem_barrier_all();
301
302 shmem_barrier_all();
303
304 shmem_barrier_all();
305
306 shmem_barrier_all();
307
308 shmem_barrier_all();
309
310 shmem_barrier_all();
311
312 shmem_barrier_all();
313
314 shmem_barrier_all();
315
316 shmem_barrier_all();
317
318 shmem_barrier_all();
319
320 shmem_barrier_all();
321
322 shmem_barrier_all();
323
324 shmem_barrier_all();
325
326 shmem_barrier_all();
327
328 shmem_barrier_all();
329
330 shmem_barrier_all();
331
332 shmem_barrier_all();
333
334 shmem_barrier_all();
335
336 shmem_barrier_all();
337
338 shmem_barrier_all();
339
340 shmem_barrier_all();
341
342 shmem_barrier_all();
343
344 shmem_barrier_all();
345
346 shmem_barrier_all();
347
348 shmem_barrier_all();
349
350 shmem_barrier_all();
351
352 shmem_barrier_all();
353
354 shmem_barrier_all();
355
356 shmem_barrier_all();
357
358 shmem_barrier_all();
359
360 shmem_barrier_all();
361
362 shmem_barrier_all();
363
364 shmem_barrier_all();
365
366 shmem_barrier_all();
367
368 shmem_barrier_all();
369
370 shmem_barrier_all();
371
372 shmem_barrier_all();
373
374 shmem_barrier_all();
375
376 shmem_barrier_all();
377
378 shmem_barrier_all();
379
380 shmem_barrier_all();
381
382 shmem_barrier_all();
383
384 shmem_barrier_all();
385
386 shmem_barrier_all();
387
388 shmem_barrier_all();
389
390 shmem_barrier_all();
391
392 shmem_barrier_all();
393
394 shmem_barrier_all();
395
396 shmem_barrier_all();
397
398 shmem_barrier_all();
399
400 shmem_barrier_all();
401
402 shmem_barrier_all();
403
404 shmem_barrier_all();
405
406 shmem_barrier_all();
407
408 shmem_barrier_all();
409
410 shmem_barrier_all();
411
412 shmem_barrier_all();
413
414 shmem_barrier_all();
415
416 shmem_barrier_all();
417
418 shmem_barrier_all();
419
420 shmem_barrier_all();
421
422 shmem_barrier_all();
423
424 shmem_barrier_all();
425
426 shmem_barrier_all();
427
428 shmem_barrier_all();
429
430 shmem_barrier_all();
431
432 shmem_barrier_all();
433
434 shmem_barrier_all();
435
436 shmem_barrier_all();
437
438 shmem_barrier_all();
439
440 shmem_barrier_all();
441
442 shmem_barrier_all();
443
444 shmem_barrier_all();
445
446 shmem_barrier_all();
447
448 shmem_barrier_all();
449
450 shmem_barrier_all();
451
452 shmem_barrier_all();
453
454 shmem_barrier_all();
455
456 shmem_barrier_all();
457
458 shmem_barrier_all();
459
460 shmem_barrier_all();
461
462 shmem_barrier_all();
463
464 shmem_barrier_all();
465
466 shmem_barrier_all();
467
468 shmem_barrier_all();
469
470 shmem_barrier_all();
471
472 shmem_barrier_all();
473
474 shmem_barrier_all();
475
476 shmem_barrier_all();
477
478 shmem_barrier_all();
479
480 shmem_barrier_all();
481
482 shmem_barrier_all();
483
484 shmem_barrier_all();
485
486 shmem_barrier_all();
487
488 shmem_barrier_all();
489
490 shmem_barrier_all();
491
492 shmem_barrier_all();
493
494 shmem_barrier_all();
495
496 shmem_barrier_all();
497
498 shmem_barrier_all();
499
500 shmem_barrier_all();
501
502 shmem_barrier_all();
503
504 shmem_barrier_all();
505
506 shmem_barrier_all();
507
508 shmem_barrier_all();
509
510 shmem_barrier_all();
511
512 shmem_barrier_all();
513
514 shmem_barrier_all();
515
516 shmem_barrier_all();
517
518 shmem_barrier_all();
519
520 shmem_barrier_all();
521
522 shmem_barrier_all();
523
524 shmem_barrier_all();
525
526 shmem_barrier_all();
527
528 shmem_barrier_all();
529
530 shmem_barrier_all();
531
532 shmem_barrier_all();
533
534 shmem_barrier_all();
535
536 shmem_barrier_all();
537
538 shmem_barrier_all();
539
540 shmem_barrier_all();
541
542 shmem_barrier_all();
543
544 shmem_barrier_all();
545
546 shmem_barrier_all();
547
548 shmem_barrier_all();
549
550 shmem_barrier_all();
551
552 shmem_barrier_all();
553
554 shmem_barrier_all();
555
556 shmem_barrier_all();
557
558 shmem_barrier_all();
559
560 shmem_barrier_all();
561
562 shmem_barrier_all();
563
564 shmem_barrier_all();
565
566 shmem_barrier_all();
567
568 shmem_barrier_all();
569
570 shmem_barrier_all();
571
572 shmem_barrier_all();
573
574 shmem_barrier_all();
575
576 shmem_barrier_all();
577
578 shmem_barrier_all();
579
580 shmem_barrier_all();
581
582 shmem_barrier_all();
583
584 shmem_barrier_all();
585
586 shmem_barrier_all();
587
588 shmem_barrier_all();
589
590 shmem_barrier_all();
591
592 shmem_barrier_all();
593
594 shmem_barrier_all();
595
596 shmem_barrier_all();
597
598 shmem_barrier_all();
599
600 shmem_barrier_all();
601
602 shmem_barrier_all();
603
604 shmem_barrier_all();
605
606 shmem_barrier_all();
607
608 shmem_barrier_all();
609
610 shmem_barrier_all();
611
612 shmem_barrier_all();
613
614 shmem_barrier_all();
615
616 shmem_barrier_all();
617
618 shmem_barrier_all();
619
620 shmem_barrier_all();
621
622 shmem_barrier_all();
623
624 shmem_barrier_all();
625
626 shmem_barrier_all();
627
628 shmem_barrier_all();
629
630 shmem_barrier_all();
631
632 shmem_barrier_all();
633
634 shmem_barrier_all();
635
636 shmem_barrier_all();
637
638 shmem_barrier_all();
639
640 shmem_barrier_all();
641
642 shmem_barrier_all();
643
644 shmem_barrier_all();
645
646 shmem_barrier_all();
647
648 shmem_barrier_all();
649
650 shmem_barrier_all();
651
652 shmem_barrier_all();
653
654 shmem_barrier_all();
655
656 shmem_barrier_all();
657
658 shmem_barrier_all();
659
660 shmem_barrier_all();
661
662 shmem_barrier_all();
663
664 shmem_barrier_all();
665
666 shmem_barrier_all();
667
668 shmem_barrier_all();
669
670 shmem_barrier_all();
671
672 shmem_barrier_all();
673
674 shmem_barrier_all();
675
676 shmem_barrier_all();
677
678 shmem_barrier_all();
679
680 shmem_barrier_all();
681
682 shmem_barrier_all();
683
684 shmem_barrier_all();
685
686 shmem_barrier_all();
687
688 shmem_barrier_all();
689
690 shmem_barrier_all();
691
692 shmem_barrier_all();
693
694 shmem_barrier_all();
695
696 shmem_barrier_all();
697
698 shmem_barrier_all();
699
700 shmem_barrier_all();
701
702 shmem_barrier_all();
703
704 shmem_barrier_all();
705
706 shmem_barrier_all();
707
708 shmem_barrier_all();
709
710 shmem_barrier_all();
711
712 shmem_barrier_all();
713
714 shmem_barrier_all();
715
716 shmem_barrier_all();
717
718 shmem_barrier_all();
719
720 shmem_barrier_all();
721
722 shmem_barrier_all();
723
724 shmem_barrier_all();
725
726 shmem_barrier_all();
727
728 shmem_barrier_all();
729
730 shmem_barrier_all();
731
732 shmem_barrier_all();
733
734 shmem_barrier_all();
735
736 shmem_barrier_all();
737
738 shmem_barrier_all();
739
740 shmem_barrier_all();
741
742 shmem_barrier_all();
743
744 shmem_barrier_all();
745
746 shmem_barrier_all();
747
748 shmem_barrier_all();
749
750 shmem_barrier_all();
751
752 shmem_barrier_all();
753
754 shmem_barrier_all();
755
756 shmem_barrier_all();
757
758 shmem_barrier_all();
759
760 shmem_barrier_all();
761
762 shmem_barrier_all();
763
764 shmem_barrier_all();
765
766 shmem_barrier_all();
767
768 shmem_barrier_all();
769
770 shmem_barrier_all();
771
772 shmem_barrier_all();
773
774 shmem_barrier_all();
775
776 shmem_barrier_all();
777
778 shmem_barrier_all();
779
780 shmem_barrier_all();
781
782 shmem_barrier_all();
783
784 shmem_barrier_all();
785
786 shmem_barrier_all();
787
788 shmem_barrier_all();
789
790 shmem_barrier_all();
791
792 shmem_barrier_all();
793
794 shmem_barrier_all();
795
796 shmem_barrier_all();
797
798 shmem_barrier_all();
799
800 shmem_barrier_all();
801
802 shmem_barrier_all();
803
804 shmem_barrier_all();
805
806 shmem_barrier_all();
807
808 shmem_barrier_all();
809
810 shmem_barrier_all();
811
812 shmem_barrier_all();
813
814 shmem_barrier_all();
815
816 shmem_barrier_all();
817
818 shmem_barrier_all();
819
820 shmem_barrier_all();
821
822 shmem_barrier_all();
823
824 shmem_barrier_all();
825
826 shmem_barrier_all();
827
828 shmem_barrier_all();
829
830 shmem_barrier_all();
831
832 shmem_barrier_all();
833
834 shmem_barrier_all();
835
836 shmem_barrier_all();
837
838 shmem_barrier_all();
839
840 shmem_barrier_all();
841
842 shmem_barrier_all();
843
844 shmem_barrier_all();
845
846 shmem_barrier_all();
847
848 shmem_barrier_all();
849
850 shmem_barrier_all();
851
852 shmem_barrier_all();
853
854 shmem_barrier_all();
855
856 shmem_barrier_all();
857
858 shmem_barrier_all();
859
860 shmem_barrier_all();
861
862 shmem_barrier_all();
863
864 shmem_barrier_all();
865
866 shmem_barrier_all();
867
868 shmem_barrier_all();
869
870 shmem_barrier_all();
871
872 shmem_barrier_all();
873
874 shmem_barrier_all();
875
876 shmem_barrier_all();
877
878 shmem_barrier_all();
879
880 shmem_barrier_all();
881
882 shmem_barrier_all();
883
884 shmem_barrier_all();
885
886 shmem_barrier_all();
887
888 shmem_barrier_all();
889
890 shmem_barrier_all();
891
892 shmem_barrier_all();
893
894 shmem_barrier_all();
895
896 shmem_barrier_all();
897
898 shmem_barrier_all();
899
900 shmem_barrier_all();
901
902 shmem_barrier_all();
903
904 shmem_barrier_all();
905
906 shmem_barrier_all();
907
908 shmem_barrier_all();
909
910 shmem_barrier_all();
911
912 shmem_barrier_all();
913
914 shmem_barrier_all();
915
916 shmem_barrier_all();
917
918 shmem_barrier_all();
919
920 shmem_barrier_all();
921
922 shmem_barrier_all();
923
924 shmem_barrier_all();
925
926 shmem_barrier_all();
927
928 shmem_barrier_all();
929
930 shmem_barrier_all();
931
932 shmem_barrier_all();
933
934 shmem_barrier_all();
935
936 shmem_barrier_all();
937
938 shmem_barrier_all();
939
940 shmem_barrier_all();
941
942 shmem_barrier_all();
943
944 shmem_barrier_all();
945
946 shmem_barrier_all();
947
948 shmem_barrier_all();
949
950 shmem_barrier_all();
951
952 shmem_barrier_all();
953
954 shmem_barrier_all();
955
956 shmem_barrier_all();
957
958 shmem_barrier_all();
959
960 shmem_barrier_all();
961
962 shmem_barrier_all();
963
964 shmem_barrier_all();
965
966 shmem_barrier_all();
967
968 shmem_barrier_all();
969
970 shmem_barrier_all();
971
972 shmem_barrier_all();
973
974 shmem_barrier_all();
975
976 shmem_barrier_all();
977
978 shmem_barrier_all();
979
980 shmem_barrier_all();
981
982 shmem_barrier_all();
983
984 shmem_barrier_all();
985
986 shmem_barrier_all();
987
988 shmem_barrier_all();
989
990 shmem_barrier_all();
991
992 shmem_barrier_all();
993
994 shmem_barrier_all();
995
996 shmem_barrier_all();
997
998 shmem_barrier_all();
999
1000 shmem_barrier_all();
```

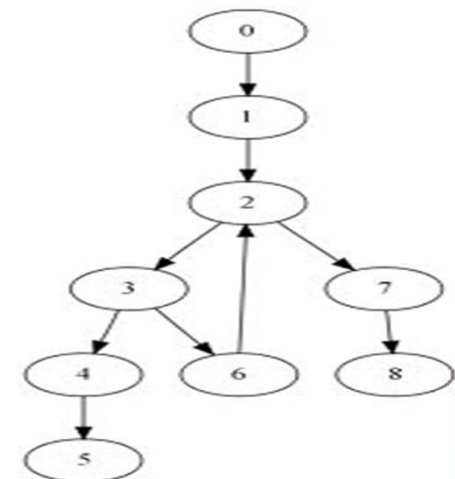
Callsite

```
135 s64Int updates[MAXTHREADS][MAXTHREADS];
136 s64Int rank;
137 void
138 Power2NodesRandomAccessUpdate(u64Int logTableSize,
139                               u64Int TableSize,
140                               s64Int LocalTableSize,
141                               u64Int MinLocalTableSize,
142                               u64Int GlobalStartMyProc,
143                               u64Int Top,
144                               int logNumProcs,
145                               int NumProcs,
146                               int Remainder,
147                               int MyProc,
148                               s64Int ProcNumUpdates,
149                               MPI_Datatype INT64_DT,
150                               MPI_Status *finish_Statuses,
151                               MPI_Request *finish_req)
152 {
153     int i,j,k;
154     int logTableLocal, ipartner, iterate, niterate;
155     int ndata, nkeep, nsend, nrecv, index, nlocal1;
156     int numthrs;
157     s64Int datum, procmask;
158     s64Int *data, *send;
159     MPI_Status status;
160     void *tstatus;
161     MPI_Win win;
162     int remote_proc, offset;
163     u64Int *tb;
164     int thisPeId;
165     int numNodes;
166     int count2;
167 }
```

Procedure

SHMEM Callgraph Analysis

- **Is start_pes() / shmem_init present in the application?**
- **If yes, is it present before other OpenSHMEM calls?**
- **start_pes should be a dominator of other OpenSHMEM calls.**
- **What about inter-procedural files?**
 - Flow insensitive analysis
 - Traverse callgraph nodes in pre-order



OpenSHMEM Initializations checks

```
• main (..) {  
    ...  
    sub();  
    ...  
}  
  
• sub (..) {  
    sub1();  
    sub2();  
}  
  
    sub1 (..) {  
        shmem_finalize();  
    }  
    sub2 (..) {  
        shmem_init()  
    }
```

- Tool will detect out of order calls inter-procedural
- Multiple instances of calls
- OpenSHMEM called outside

How-to-use: multi-init.c

```
int main (int argc, char **argv)
{
    start_pes (0);
    start_pes (0);
    printf ("Hello from multi-init test\n");

    return 0;
}
```

- Multiple start_pes()
- No shmem_finalize()

How-to-use:

```
$cd startpes-ipa
```

```
$ shmemcc -ipa test-startpes-ipa.c
```

```
*** OpenSHMEM Warning: more than one  
OpenSHMEM initialization call found ***
```


Examples: Symmetric Errors

- **Storage checking:**

```
#ifdef N 64
```

```
int main () {
```

```
    long dest[N], src[N]; //error: dest non-  
symetric
```

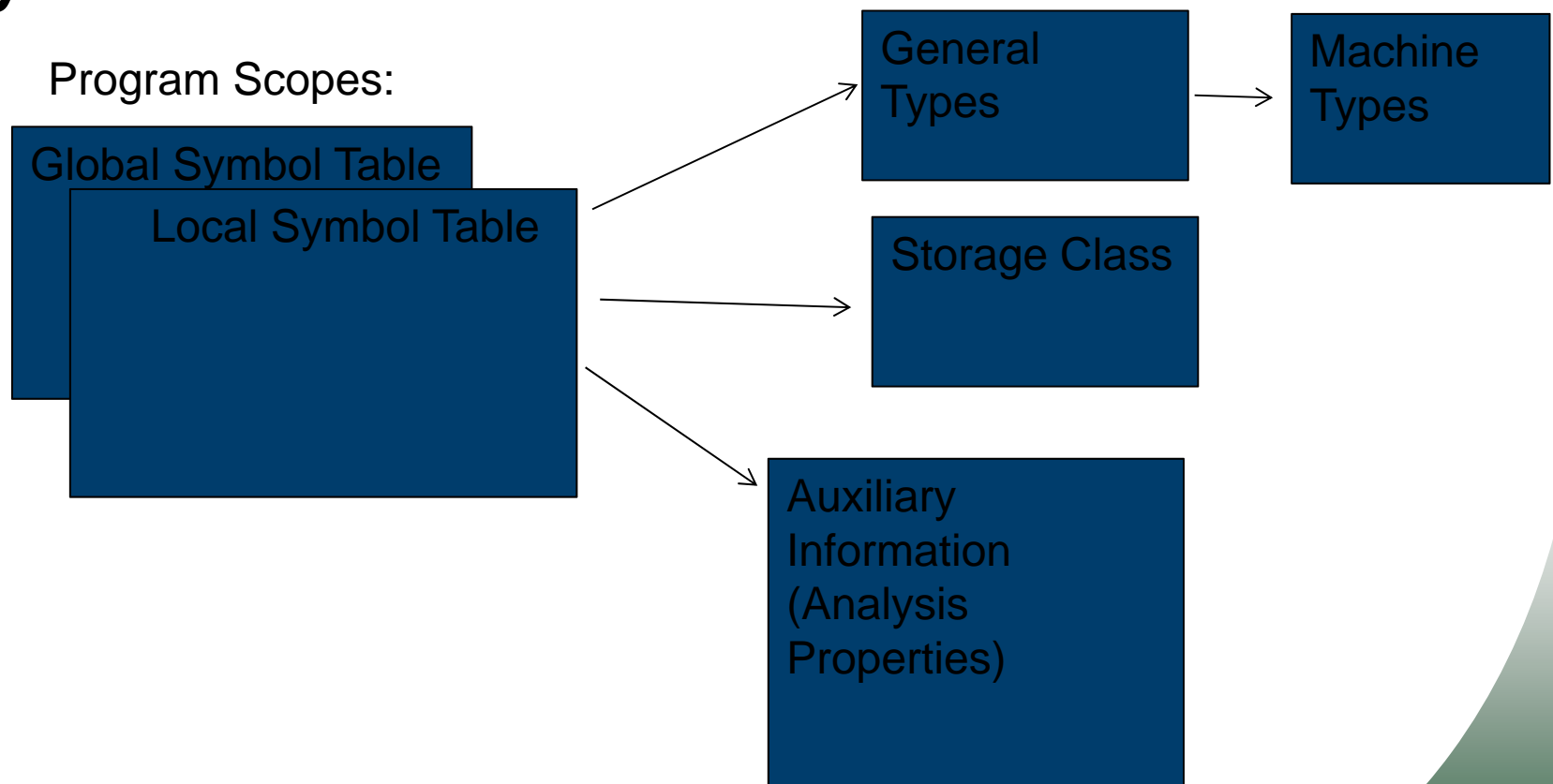
```
<....>
```

```
    shmem_long_put(dest, src, 64, 1);
```

- **Check if variables are symmetric.**
 - Check within storage tables of compiler
 - Differentiate if variables are pointers or arrays
 - Querying compiler's symbol tables.

SHMEM storage/type checks

- **A variable is represented in a compiler this way:**



Example: badget.c

```
Int main (int argc, char **argv) {  
    long dest, src;  
    int me, npes;  
    start_pes (0);  
  
    me = _my_pe ();  
    npes = _nu_pes ();  
    src = 42;  
    shmem_barrier_all ();  
    if (me == 0) {  
        shmem_long_get (&dest, &src, 1, 1);  
    }  
    shmem_barrier_all ();  
    return 0;  
}
```

Evaluation

```
$ cd budget
```

```
$ shmemcc -ipa budget.c
```

```
*** OpenSHMEM Warning: non-symmetric  
variable in arg2 of shmem_long_get (line=65,  
file=budget.o) ***
```

Example: Out of bounds checking

```
#ifdef N 64
```

```
int main () {
```

```
    long dest[N], src[N]; //error: dest non-  
symetric
```

```
<....>
```

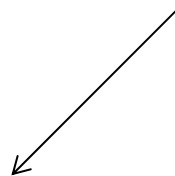
```
    shmem_long_put(dest, src, N+2, 1);
```

- **Check if variables are accessed within bounds.**
 - Works for array accesses for both dest and src
 - Depends if constant propagation is known for length of accesses

Evaluation: test-bounds.c

```
int main(void) {  
    int i, src[N];  
    long lget[N];  
    static int targ_static[N];  
  
    start_pes(0);  
  
    for(i=0; i< N; i++) {  
        src[i] = my_pe() + i;  
        froml[i] = my_pe() + i*i;  
    }  
  
    shmem_int_put(targg, srcg, N+M, 2);  
    shmem_int_put(targ_static, src, N+M, 3);  
    shmem_long_get(lget, froml, N+M, 4);  
  
    shmem_barrier_all(); /* sync sender and receiver */  
    return 1;  
}
```

Out of bounds accesses
of arrays of size N



Evaluation

- **%cd bounds**
- **%shmemcc -ipa test-bounds.c**

***** OpenSHMEM Warning: out of bounds access of
shmem_int_put arg1 of 8 elements with access of 18 elements
(line=20, file=test-bounds.o) *****

***** OpenSHMEM Warning: out of bounds access of
shmem_int_put arg2 of 8 elements with access of 18 elements
(line=20, file=test-bounds.o) *****

•

Example: Out of bounds checking with constant propagation

- **Since out of bounds checks relies on constants being propagated correctly.**
- **The OpenSHMEM analyzer can perform:**
 - **Constant Folding**
 - **Inter-procedural constant propagation**
 - **Common expression elimination that evaluate to constants**

Evaluation: test-bounds-const.c

```
int main(void) {  
    int i, src[N], targ[N], len=N;  
    long lget[N];  
    static int targ_static[N];  
    start_pes(0);  
  
    for(i=0; i< N; i++) {  
        src[i] = my_pe() + i;  
        froml[i] = my_pe() + i*i;  
        len++;  
    }  
    len++;  
    shmem_int_put(targg, srcg, len, 2);  
    shmem_int_put(targ_static, src, len, 3);  
    shmem_long_get(lget, froml, len, 4);  
    ....  
}
```

- len is a variable that is initialized to N
- len is increased by N
- len is increased by 1

Out of bounds access with len= 2N+1

Evaluation: test-bounds-const.c

- **%cd bounds-constprog**
- **%shmemcc -ipa test-bounds-const.c**

***** OpenSHMEM Warning: out of bounds access of shmem_int_put arg1 of 16 elements with access of 33 elements (line=23, file=test-bounds-constprog.o) *****

***** OpenSHMEM Warning: out of bounds access of shmem_int_put arg2 of 16 elements with access of 33 elements (line=23, file=test-bounds-constprog.o) *****

***** OpenSHMEM Warning: out of bounds access of shmem_int_put arg1 of 16 elements with access of 33 elements (line=24, file=test-bounds-constprog.o) *****

Examples: Out of bounds in strided data accesses

```
#ifdef N 10
```

```
int main () {
```

```
    static long dest[N],
```

```
    long src[N];
```

```
    shmem_long_iput(dest, src, 100, 2, 5 1);
```

- **Check if variables are within range**
 - Check if access to variables are out of range
 - Perform Constant propagation and check if len is constant

Evaluation

```
#define N 7  
short src1[N];  
int src2[N];  
long src3[N];  
...
```

```
Int main () {
```

```
....
```

```
    shmem_barrier_all();
```


```
    shmem_iget32(dest2, src2, 1, 2, N, npes-1);
```

```
    shmem_iget64(dest3, src3, 1, 2, N, npes-1);
```

```
    shmem_iget128(dest4, src4, 1, 2, N, npes-1);
```

```
....
```

Out of bounds accesses of
for strided accesses



Evaluation (Bug in Validation Suite)

- **%cd iget-global**
- **%shmemcc -ipa test_shmem_get_globals.c**

*** OpenSHMEM Warning: out of bounds access of shmem_iget32 arg2 of 7 elements with access of 14 elements (line=297, file=test_shmem_get_globals.o) ***

*** OpenSHMEM Warning: out of bounds access of shmem_iget64 arg2 of 7 elements with access of 14 elements (line=298, file=test_shmem_get_globals.o) ***

*** OpenSHMEM Warning: out of bounds access of shmem_iget128 arg2 of 7 elements with access of 14 elements (line=299, file=test_shmem_get_globals.o) ***

*** OpenSHMEM Warning: out of bounds access of shmem_short_iget arg2 of 7 elements with access of 14 elements (line=395, file=test_shmem_get_globals.o) ***

OpenSHMEM Data Flow problems

- **The tool will provides static analysis to respond questions:**
 - **How data is being used and initialized?**
 - **Framework for data flow information.**
 - Intra and inter-procedural data flow
 - **How to construct use-def chains that are OpenSHMEM aware?**
 - For example locate uninitialized variables in OpenSHMEM calls
 - Pointers that get propagated to constants or invalid addresses (i.e. Null)
 - How to bind use-define of variables with correct memory allocators?
 - **How to make the analysis accurate to be flow sensitive, process sensitive, etc.**

Check for Allocation of Data

```
int main () {  
    src = (long *) shmalloc (N * sizeof (long));  
    <...>  
    shmem_long_get(targ, src, N, 1);  
    <...>  
}
```

- Have data been allocated?
- Have data been initialized?
- Static vs. Dynamic testing

IPA Dataflow analysis

- **Traverse intermediate representations of each compilation unit**
- **Mark variables accessed via OpenSHMEM calls with extra field in symbol tables**
- **Aggregate global tables and symbols procedures at “ipa_link” time**
- **Perform Constant Propagation**
- **Check for allocation calls of variables and mark them**
- **Use data flow analysis to check use-def chains**

Constructing Use-Def Chains for OpenSHMEM calls

- Each use must be defined by 1 and only 1 def
- Straight-line code trivially single-assignment
- Uses-to-defs: many-to-1 mapping
- Each def dominates all its uses

Pointer becomes undefined

```
long *a = shmalloc(..)
      ↑
      |
shmem_long_put(a,..)
      |
      |
shmem_long_get(...,a)

shfree(a)
a = shmalloc(..)
      ↑
      |
shmem_long_put(a,..)
```

Challenges for Pointer Analysis

- **Pointers are represented by scalars in compilers**
- **Accesses in OpenSHMEM calls for pointers can be complex expressions with address calculations**
- **Common expression elimination might be needed to simplify analysis**
 - This is sometimes used for re-computation of addresses
- **Pure scalars variables can affect the address calculation and might be uninitialized**

Example: Complex pointer usage

....

```
float *x, *y;
```

```
static float xs[100],ys[100];
```

```
x = (float*)shmalloc((n_local1 - n_local0 + 2)*sizeof(float));
```

Initialization of pointers
*Wrong initialization of y

```
y = (float*)malloc((n_local1 - n_local0 + 2)*sizeof(float));
```

```
x -= (n_local0 - 1);
```

```
y -= (n_local0 - 1);
```

```
shmem_barrier_all();
```

Address calculation, address is
subtracted by $n_local0 - 1$

```
//... // fill x, y
```

```
// fill ghost zone
```

```
if (_my_pe() > 0)
```

```
shmem_float_get(ys,xs,n1,1); // extra code
```

.....

```
if (!*_my_pe() < _num_pes()-1 *!(int)y[2])
```

```
shmem_float_put(&y[n_local0-1], &y[n_local1-1], 1, _my_pe()+1);
```

```
shmem_barrier_all();
```

Address accessed by adding
 $n_local0 - 1$

We need to check also if
 n_local0 has been initialized
and if it is part of
a pointer expression.

How pointer is represented in IR

- **shmem_float_put(&y[n_local0-1], ...);**
- **IR representation:**

INTCONST(-1)

LOAD n_local

Need to check:

If variable is part of a pointer address calculation
If variable has been properly initialized

SUB

LOAD y

ADD

PARAM 0

Need to check if this
Variable is of type pointer

CALL shmem_float_put

Managed by UT-Battelle for the
U. S. Department of Energy

Example: Complex pointer usage

....

```
float *x, *y;
```

```
static float xs[100],ys[100];
```

UD n_local1
UD n_local0

```
y = (float*)malloc((n_local1 - n_local0 + 2)*sizeof(float));
```

```
x -= (n_local0 - 1);
```

```
y -= (n_local0 - 1);
```

UD y1 (SSA)
UD n_local0

```
shmem_barrier_all();
```

```
//... // fill x, y
```

```
// fill ghost zone
```

UD y2 (SSA)

```
if (_my_pe() > 0)
```

```
shmem_float_get(ys,xs,n1,1);
```

UD n_local0

.....

```
if (/*_my_pe() < _num_pes()-1 */(int)y[2])
```

```
shmem_float_put(&y[n_local0-1], &y[n_local1-1], 1, _my_pe()+1);
```

```
shmem_barrier_all();
```

Evaluation:

- **%cd heap**
- **%shmemcc -ipa test-shmem_heap.c**

***** OpenSHMEM Warning: Local pointer in arg2 of OpenSHMEM call (line=24, file=test-shmem_heap.c) initialized with malloc *****

***** OpenSHMEM Warning: Local pointer in arg1 of OpenSHMEM call (line=26, file=test-shmem_heap.c) initialized with malloc *****

Global pointers

- **For the case of global pointers, we keep track of all global pointer initializations per procedures**
- **We summarize their local access in a table and mark them initialized.**
- **If global pointer is not in the table and accessed in an OpenSHMEM call a warning is generated.**

Evaluation:

```
#include <shmem.h>
#include <stdlib.h>
int n1=101;
float *x, *y;
void variable_allocation(int n) {
    int nn = (n-1) / _num_pes();
    int n_local0 = 1 + _my_pe() * nn;
    int n_local1 = 1 + (_my_pe()+1) * nn;
    // y = (float*) shmalloc((n_local1 - n_local0 + 2)*sizeof(float)); // forgot to initialize pointer

}

int main(...) {
    m = ..
    variable_allocation(m)
    shmem_float_put(y,x, 1, _my_pe()-1);

}
```


Evaluation:

- **%cd heap-global**
- **%shmemcc -ipa test-shmem_heap-global.c**

***** OpenSHMEM Warning: global variable arg1
of call shmem_float_put is uninitialized
line=20, file=shmem_heap-global.0) *****

Other Data Flow problems: going beyond scalars or array single accesses

- **How can we summarize data accesses at different granularity levels:**
 - Loop data accesses
 - Basic Block Level at the Control Flow
 - Procedure level
- **Alias analysis information: The tool is able to check if a symmetric variable is aliased.**

How to use OpenSHMEM Analyzer

- **Text-Based Analysis:**

- Use like another compiler with the option **–ipa**
 - If more aggressive analysis is require use **–ipa –O3**
- **shmemcc –ipa mytest.c**
- Use the default **shmem.h** provided by the Analyzer

- **Callgraph Analysis**

- **shmemcc –ipa mytest.c -o test**
- **firefox test.html**

Makefile Example:

- Driver: shmemcc

%cat Makefile

CC = shmemcc

CXX = shmemCC

CFLAGS = -ipa

LDFLAGS = \$(CFLAGS)

TARGET = ra_shmem

OBJECTS = RandomAccess.o SHMEMRandomAccess.o verification.o

.SUFFIXES: .c

.c.o:

\$(CC) \$(INCDIR) \$(CFLAGS) -c \$<

code2html -l C -n -N \$< \$*.html

- Converts sources to HTML format

all: RASHMEM

RASHMEM: \$(OBJECTS)

\$(CC) \$(INCDIR) \$(CFLAGS) \$(OBJECTS) -o \$(TARGET) \$(LDFLAGS)

callgraph ra_shmem &> index.html

firefox index.html

- Provides error analysis

- Builds graphical callgrph

- Makefile modifications
- Use the tool as a regular compiler
- It includes OpenSHMEM (shmem.h)
- Uses a special version of SHMEM library that can used for IPA.
- It will not generate executables

How to Install

- **Installation instructions:**
- **Install code2html 0.9.1**
- **<http://www.palfrader.org/code2html/>**
- **Use current .tar.gz file)**
- **Note: Make sure to add the path to 'code2html' to your \$PATH**
- **Install graphviz 2.28.0**
- **<http://www.graphviz.org/>**
- **Use current .tar.gz file**
- **Recommended Configuration: `./configure --prefix=<install directory> --enable-python=no`**
- **Note: Make sure to add the path to 'dot' to your \$PATH**

How to Install (2)

- Go to www.openshmem.org/OSA
- **tar -xzvf openshmem-analyzer-1.0.tar.gz**
- **cd openshmem-analyzer-1.0/build**
- **../configure --prefix=<openshmem analyzer install directory> --disable-host_bdver1-support --with-build-optimize=DEBUG**
- **gmake**
- **gmake install**
- **Add <openshmem analyzer install directory>/bin to your \$PATH**

Supported Environments

- **Processors: IA32, X86_64 with**
- **Supports 32/64-bits builds**
- **SLES 11 SP1, RHEL 6 (GLIBC 2.11 and above) [Development]**
- **SLES 10 SP2, SLES 10 SP3, RHEL 5.5**

Acknowledgements



This work was supported by the United States Department of Defense & used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory.