Evaluating Error Detection Capabilities of UPC Run-time Systems

Glenn R. Luecke, James Coyle, James Hoekstra, Marina Kraeva, Ying Xu, Elizabeth Kleiman and Olga Weiss
High Performance Computing Group
Iowa State University
Ames, Iowa, 50011
Email: grl@iastate.edu

***Abstract*—The ability of system software to detect run-time errors and issue messages that help programmers quickly fix these errors is an important productivity criterion for developing and maintaining application programs. To evaluate this capability for Unified Parallel C (UPC), over two thousand run-time error tests and a run-time error detection (RTED) evaluation tool have been developed. For each error message issued, the RTED evaluation tool assigns a score from 0 to 5 based on the usefulness of the information in the message to help a programmer quickly fix the error. The RTED evaluation tool calculates averages over each error category and then prints the results. All tests and the RTED evaluation tool are freely available at the RTED web site http://rted.public.iastate.edu/UPC. The Cray, Berkeley, HP and GNU UPC compilers have been evaluated and results posted on this same web site.**

I. Introduction

Unified Parallel C is an extension of C for parallel execution on shared or distributed memory parallel machines [1], [2]. Some researchers prefer to write scientific applications in UPC rather than using C with MPI [3] since UPC is easier to use and can provide good performance [4], [5]. Providing a productive programming environment for UPC will encourage new scientific applications to be written in UPC instead of MPI. Since debugging UPC programs can be time consuming, it is important to have UPC compilers, tools and run-time systems that can detect run-time errors and issue messages that help programmers quickly fix the errors. Having high-quality error messages can greatly increase programmer productivity when developing and maintaining application programs by reducing debugging time.

With funding from the US Department of Defense, from DARPA's High Productivity Computing Initiative and from the Extreme Scale System Center at Oak Ridge National Laboratory, 2247 tests have been developed by Iowa State University's High Performance Computing Group for evaluating parallel run-time error detection capabilities of UPC implementations. A run-time error detection (RTED) evaluation tool has been developed for running these tests, assigning a score from 0 to 5 based on the quality of the error message, calculating the averages of these scores for each error category and reporting results.

Supported by the DoD and in conjunction with The Extreme Scale System Center at Oak Ridge National Laboratory.

These tests and the RTED evaluation tool provide an easy way to evaluate and compare run-time error detection capabilities of different UPC implementations and could be used as part of a computer procurement process. In addition, vendors could use these tests, recommended error messages and the RTED evaluation tool to evaluate and improve the run-time error detection capabilities of their UPC implementations.

UPC run-time test results, tests, recommended error messages and the RTED evaluation tool are freely available at http://rted.public.iastate.edu/UPC. As new UPC compilers/releases become available, vendors are encouraged to send results to rted.project@iastate.edu so they can be posted on this web site.

II. Background

Unified Parallel C (UPC) is an extension of the ISO C 99 programming language designed for shared and distributed memory high performance parallel machines. UPC uses a single shared, partitioned address space, where variables may be directly read and written by any processor, but each variable is physically associated with a single processor [6]. UPC combines the programmability advantages of the shared memory programming paradigm and the control over data layout and performance of the message passing programming paradigm.

The UPC Compiler Group at the University of California Berkeley/Lawrence Berkeley National Laboratory has developed the Berkeley UPC compiler. The goal of the Berkeley UPC compiler group is to develop an open-source, portable, high performance implementation of UPC for large-scale multiprocessors, PC clusters, and clusters of shared memory multiprocessors [6].

The UPC working group at the High Performance Computing Lab (HPCL) at George Washington University is involved in the UPC specification, UPC testing strategies, UPC documentation, UPC testing suites, UPC benchmarking, and UPC collective and Parallel I/O specification [7].

UPC work at Michigan Technological University includes the release of the MuPC run time system for UPC, the development of the UPC collectives specification, memory model research, programmability studies, and test suite development [8].

Researchers at the University of Florida's High Performance Computing and Simulation Laboratory are currently involved in the research and development of a next-generation performance analysis tool supporting UPC. This tool will facilitate users in identifying bottlenecks in their programs and will serve as a testbed for advanced analysis techniques aimed at increasing programmer productivity [9].

Currently, Cray, Berkeley, HP and GNU have UPC compilers and a UPC compiler is under development at IBM. However, GNU's and HP's UPC do not currently support UPC IO and GNU's UPC also does not support UPC collective utilities.

III. Methodology

This section summarizes the methodology used to develop run-time error tests and the RTED evaluation tool. For each error, a program has been written that contains the specified error and no other errors (each program contains one and only one run-time error). Tests were written so that the information needed to detect the error is not available at compile time. For each test a file with a recommended error message was created that contains the error name, the line number and the file name where the error occurs along with any additional information that would assist a programmer to find and correct the error. The recommended error message for each test was created based on the evaluation the error messages produced and the experience of the people in ISU's High Performance Computing Group.

The UPC run-time error tests have been written to cover a wide range of errors. The following are the UPC run-time error categories:

- out-of-bounds shared memory access using indices
- out-of-bounds shared memory access using pointers
- out-of-bounds shared memory access in UPC functions
- argument errors in UPC functions
- wrong order of UPC statements and function calls
- uninitialized variables
- deadlocks
- race conditions
- memory related errors
- undefined UPC operations
- warnings

The "warnings" category includes tests where programmers should be warned of likely errors. For example, it makes no sense to have the "nelems" argument in reduction functions be zero even though the UPC specification allows this. Thus, some tests have nelems=0 and the RTED evaluation tool checks whether good warning messages are produced.

The RTED evaluation tool mentioned above is a collection of scripts for running the tests, comparing actual messages with expected messages and then assigning a score of 0, 1, 2, 3, 4 or 5 to the message generated by each tests:

- A score of 5 is given for a detailed error message that will assist a programmer to fix the error.
- A score of 4 is given for error messages with more information than a score of 3 and less than 5. This is tailored for each test.
- A score of 3 is given for error messages with the correct error name, line number and the name of the file where the error occurred.
- A score of 2 is given for error messages with the correct error name and line number where the error occurred but not the file name where the error occurred.
- A score of 1 is given for error messages with the correct error name.
- A score of 0 is given when the error was not detected.

Different compilers, tools and run-time systems may issue different messages (with different error names) for the same run-time error. The RTED evaluation tool has a list of synonymous phrases for each error so that equivalent error messages will be evaluated appropriately. Additional synonymous phrases may need to be added as new compilers/releases become available. Error messages are evaluated by the RTED evaluation tool as follows:

- For each test and score a scoring script was created.
- Error messages are reduced to a canonical form for easy comparison with the recommended error messages by first changing all text to lower case and then replacing selected phrases with standard phrases. Blanks, hex addresses, and integers longer than three digits are removed to reduce false matches.
- Scoring scripts are applied to the canonical form of each error message for evaluation.

IV. Results

This section contains the results of running the UPC run-time error tests and the RTED evaluation tool on the Berkeley, Cray, HP and GNU UPC compilers. Results for each category of run-time error are in Table 1 and also on the web site, http://rted.public.iastate.edu/UPC .

The following example is one of the run-time error tests for wrong order of UPC calls. The example shows the message produced by the Berkeley, HP and Cray UPC compilers and gives our recommended error message. Notice that the evaluation tool does not take into consideration verbose error messages such as the one produced by Cray and listed below.

```
 c_I_3_b_D.upc:
 . . .
int main()
{
   . . .
   if(zero()==0) upc_notify;
   callCollectiveFunction();
   if(zero()==0) upc_wait;
   upc_barrier;
   . . .
}

c_I_3_b_D_s.upc:
. . .
void callCollectiveFunction()
{
 upc_all_scatter(dstB,srcA,sizeof(double),
 UPC_IN_ALLSYNC|UPC_OUT_ALLSYNC);
```

```
}
```

Berkeley's UPC gave the following message. This message was given a score of 1 since it did not give the line number where the error occurred.

```
*** FATAL ERROR: gasnet_barrier_notify() called
twice in a row
*** FATAL ERROR: gasnet_barrier_notify() called
twice in a row
*** FATAL ERROR: gasnet_barrier_notify() called
twice in a row
*** FATAL ERROR: gasnet_barrier_notify() called
twice in a row
```

HP's UPC gave the following message. This message was given a score of 1 since it did not give the line number where the error occurred.

```
0: UPCRTS - Notify 0 attempted prior to wait 0.
0: UPCRTS - Fatal error detected during execution,
            exiting
3: UPCRTS - Notify 0 attempted prior to wait 0
```

Cray's UPC gave the following message. This message was given a score of 5 since it contains all the information in our recommended error message.

```
Error in thread 0 - upc_notify or upc_barrier
called while previous call to upc_notify is
still pending with no matching call to upc_wait

Error in thread 1 - upc_notify or upc_barrier
called while previous call to upc_notify is
still pending with no matching call to upc_wait

Error in thread 3 - upc_notify or upc_barrier
called while previous call to upc_notify is
still pending with no matching call to upc_wait

Error in thread 2 - upc_notify or upc_barrier
called while previous call to upc_notify is
still pending with no matching call to upc_wait

Traceback for process 904531(ssp mode) apid
904531.0 on node 0 _kill+0x0030 (0x1195B30) at
kill.c

    _raise+0x00A8 (0x115FC08) at raise.c:27
abort+0x00B8 (0x115D438) at abort.c:44
     ___upc_notify+0x0734 (0x1145F34) at
upc_notify_wait.c:175
     upc_all_scatter+0x02C0 (0x1047900) at
upc_all_scatter.c:56
     callCollectiveFunction+0x013C (0x1002DFC)
at c_I_3_b_D_s.upc:34
main+0x0254 (0x1003734) at c_I_3_b_D.upc:46
Fault: Killed by user id 26414 process 904531
./limit_real_time_nums_a[4]: 904531 Abort
```

Our recommended error message for this test is:

```
ERROR: invalid order
Function 'upc_all_scatter' is called at line 34
in file 'c_I_3_b_D_s.upc' by all processes
while a prior invocation of function
'upc_notify' is still pending on the same
processes.
```

Table 1 presents the scores when running the 2247 UPC tests using Cray's, Berkeley's, HP's and GNU's UPC compilers. The section "Undefined UPC Operations" contains situations where the outcome of certain UPC statements is stated as being undefined by the UPC specification. The UPC "warnings" category is described in section 3. GNU's and HP's UPC do not support UPC IO so these tests were skipped when using these compilers and scores calculated on the reduced set of tests. In addition, GNU's UPC does not support UPC collective utilities so these tests were also skipped when using GNU' s UPC compiler and scores calculated on the reduced set of tests. Notice that Cray's UPC compiler scored better than all the others in some categories.

TABLE I
UPC RESULTS FOR EACH ERROR CATEGORY.

UPC run-time error category	Cray	Berkeley	HP	GNU
Out-of-bounds shared memory access using indices	1.30	0.00	0.03	0.20
Out-of-bounds shared memory access using pointers	1.04	0.00	0.00	0.21
Out-of-bounds shared memory access using UPC functions	0.91	0.00	0.02	0.01
Argument errors in UPC functions	0.38	0.04	0.00	0.00
Wrong order of UPC calls	0.84	0.20	0.53	0.89
Uninitialized variables	0.08	0.02	0.57	0.25
Deadlocks	0.00	0.58	0.36	0.27
Race conditions	0.01	0.00	0.00	0.00
Memory related errors	0.18	0.00	0.16	0.37
Undefined UPC operations	0.19	0.21	0.15	0.41
Warnings	0.27	0.00	0.00	0.00
AVERAGES	0.47	0.10	0.17	0.24

V. CONCLUSIONS

The ability of system software to detect run-time errors and issue messages that help programmers quickly fix these errors is an important productivity criterion for developing and maintaining application programs. To evaluate this capability for Unified Parallel C (UPC), over two thousand run-time error tests and a RTED evaluation tool have been developed. For each error message issued, the RTED evaluation tool assigns a score from 0 to 5 based on the usefulness of the information in the message to help a programmer quickly fix the error. The RTED evaluation tool calculates averages over each error category and then prints the results. All tests and the RTED evaluation tool are freely available at the RTED web site http://rted.public.iastate.edu/UPC.

The Cray, Berkeley, HP and GNU UPC compilers have been evaluated and results posted on this same web site. Even though the technology for detecting and reporting many run-time errors is known, scores for the Cray, Berkeley, HP and GNU compilers are low (with the exception of Cray's UPC compiler which provided better messages in some error categories).

It is hoped that these tests and recommended error messages will be used by vendors to evaluate and improve the run-time error detection capabilities of their UPC compilers, tools and run-time systems. We also hope that these tests will be used by high performance computing centers as part of their procurement process to reward vendors whose UPC implementations provide excellent run-time error messages.

A productive programming environment would include not only the detection of run-time errors but also detection of

errors at compile-time when it is possible. Compile-time error tests and a compile-time error detection evaluation tool for UPC are currently under development. These will be freely available by November 2009 at http://hpcgroup.public.iastate.edu/cted.

REFERENCES

[1] T. El-Ghazawi, W. Carlson, T. Sterling and K. Yelick, *UPC: Distributed Shared Memory Programming*, Wiley-Interscience, 2005.

[2] Unified Parallel C (Wikipedia), http://en.wikipedia.org/wiki/Unified_Parallel_C

[3] M. Snir, S Otto, S. Huss-Lederman, D. Walker, J. Dongarra, *MPI-The Complete Reference*, The MIT Press, 1998.

[4] Thomas Dunigan, Jeffrey Vetter, James White III, Patrick Worley, *Performance Evaluation of the Cray X1 Distributed Shared Memory Architecture*, IEEE Micro, Jan/Feb 2005

[5] Richard Barrett1, Yiyi Yao2, and Tarek El-Ghazawi, *Evaluation of UPC on the Cray X1E*, Cray Users Group, 2006.

[6] The Berkeley Unified Parallel C: http://upc.lbl.gov/

[7] Unified Parallel C at George Washington University, http://upc.gwu.edu/

[8] UPC Projects at Michigan Technological University: http://www.upc.mtu.edu/

[9] High Performance Computing and Simulation Laboratory, University of Florida: http://www.hcs.ufl.edu/upc/

[10] High Performance Computing Group, Iowa State University: http://hpcgroup.public.iastate.edu